RENDERIZAÇÃO NÃO-FOTORREALISTA EM TEMPO-REAL DE LÍQUIDOS BASEADA EM PARTÍCULAS

Abel Bruno Nascimento Silva

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Ricardo Guerra Marroquim

Rio de Janeiro

Março de 2016

RENDERIZAÇÃO NÃO-FOTORREALISTA EM TEMPO-REAL DE LÍQUIDOS BASEADA EM PARTÍCULAS

Abel Bruno Nascimento Silva

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Ricardo Guerra Marroquim, D.Sc.

Prof. Claudio Esperança, Ph.D.

Prof. Esteban Walter Gonzalez Clua, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
MARÇO DE 2016

Silva, Abel Bruno Nascimento

Renderização Não-Fotorrealista em Tempo-Real de Líquidos Baseada em Partículas/Abel Bruno Nascimento Silva. – Rio de Janeiro: UFRJ/COPPE, 2016.

XV, 46 p.: il.; 29, 7cm.

Orientador: Ricardo Guerra Marroquim

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2016.

Referências Bibliográficas: p. 42 – 46.

1. Renderização Não-Fotorrealista. 2. Simulação de Fluidos. 3. NPR. 4. SPH. I. Marroquim, Ricardo Guerra. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A Deus por ter me dado força e sabedoria nessa caminhada. E a minha família pelo apoio incondicional.

Agradecimentos

A Deus, por tudo. Pois sem Ele eu nada conseguiria.

Aos meus pais, Laercio e Dilza, que sempre apoiaram e incentivaram os meus estudos e a minha caminhada acadêmica mesmo que isso significasse me mudar para outro estado. Embora não soubessem nada sobre simulação de fluidos ou renderização não-fotorrealista, saber que estavam ao meu lado sempre que eu precisasse foi fundamental para a conclusão deste trabalho.

A minha irmã Lana, minha companheira de filmes e videogame que sempre me incentiva a comer soja em favor dos animais. Suas correções ortográficas sempre fizeram com que eu me sentisse mais seguro ao escrever um artigo ou trabalho, nunca entregando a versão final sem que ela não tivesse corrigido antes.

Aos meus orientadores, professores Esperança e Marroquim, por terem acreditado em mim e me dado a oportunidade de estar inserido em um laboratório de pesquisa em computação gráfica. Este período de três anos que passei sob suas orientações foi essencial para o meu desenvolvimento, me proporcionando conhecimento e experiência.

À UFRJ e ao LCG por me fornecer o espaço e a infraestrutura necessária para o meu desenvolvido acadêmico. Agradeço aos professores que se esforçaram para transferir o seu conhecimento e àqueles que me desafiaram como aluno a sempre buscar mais conhecimento, para que assim eu pudesse me tornar um mestre em computação gráfica.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro.

A minha amiga Talita por todo o suporte que ofereceu em minha mudança para o estado do Rio de Janeiro. Sem a sua imensa ajuda essa etapa seria um grande obstáculo para o desenvolvimento deste trabalho.

Finalmente, aos meus colegas de laboratório, tanto os que conseguiram alcançar o final do mestrado quanto àqueles que ainda vão conseguir. Apesar de todos terem contribuído para essa experiência incomparável que é estar no LCG, destaco o meu amigo Lucas, que apesar de quase nunca concordar comigo, foi muito importante para mim, seja pelos trabalhos que realizamos juntos ou mesmo pelas conversas sobre cultura pop e a grande cidade de Itabira-MG.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

RENDERIZAÇÃO NÃO-FOTORREALISTA EM TEMPO-REAL DE LÍQUIDOS BASEADA EM PARTÍCULAS

Abel Bruno Nascimento Silva

Março/2016

Orientador: Ricardo Guerra Marroquim

Programa: Engenharia de Sistemas e Computação

O avanço tecnológico dos computadores possibilitou o desenvolvimento da simulação computacional de fluidos. Entre os diversas métodos de simulação, Smoothed-Particle Hydrodynamics (SPH) tornou-se um dos mais aplicados devido sua simplicidade teórica e facilidade de implementação, além disso, por ser uma abordagem altamente paralelizável, aproveita grande parte da capacidade das placas gráficas modernas.

Embora o surgimento da Renderização Não-Fotorealista (NPR) tenha permitido uma forma alternativa de visualização baseada em arte, abstração e estilização, a área de visualização de fluidos ainda tem focado-se no desenvolvimento de técnicas fotorealistas, havendo pouco estudo na renderição não-fotorealista de fluidos, o que seria de grande valor às indústrias cinematográfica, de animação e de jogos.

Este trabalho implementa um sistema de simulação de fluidos baseado em partículas. Além disso, desenvolve modelos de visualização não-fotorealista em tempo-real para a simulação de fluidos.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

REAL-TIME NON-PHOTOREALISTIC RENDERING OF SMOOTHED PARTICLE HYDRODYNAMICS

Abel Bruno Nascimento Silva

March/2016

Advisor: Ricardo Guerra Marroquim

Department: Systems Engineering and Computer Science

Technological advances has allowed for the development of computational fluid simulation. Among the several methods of simulation, Smoothed-Particle Hydrodynamics (SPH) has become one of the most applied due to its theoretical simplicity and ease of implementation. Furthermore, as it is a highly parallelizable approach it leverages many of the modern graphic cards' features.

The emergence of Non-Photorealistic Rendering (NPR) has pursued another method of rendering based on art, abstraction and stylization. Nevertheless, the fluid visualization field is still focused on the development of photorealistic techniques, thus there are few studies of fluid non-photorealistic rendering, which would be of great value to entertainment industries, for example.

This work implements a Smoothed-Particle Hydrodynamics system. In addition, it develops models for real-time non-photorealistic rendering for the simulated fluid.

Sumário

Lista de Figuras **xi**

Lista de Tabelas **xiii**

Lista de Símbolos **xiv**

Lista de Abreviaturas **xv**

1 Introdução **1**

1.1 Objetivo 1

1.2 Organização da tese 2

2 Simulação de Fluidos **3**

2.1 Teoria dos Fluidos 3

2.1.1 Equações de Navier-Stokes 4

2.2 Tipos de Abordagem 6

2.2.1 Método Euleriano 6

2.2.2 Método Lagrangiano 7

2.2.3 Método de Lattice Boltzmann 8

2.3 Simulação de Fluidos Baseada em Partículas 9

2.3.1 Núcleo de Suavização 9

3 Visualização de Fluidos **13**

3.1 Marching Cubes 13

3.2 Ray Tracing 14

3.3 Renderização Baseada em Pontos 14

3.4 Renderização no Espaço da Imagem 15

3.4.1 *Splats* . . . 16
3.4.2 Profundidade . . . 17
3.4.3 Suavização . . . 17
3.4.4 Normais . . . 18
3.4.5 Renderização . . . 19
3.4.6 Transparência . . . 20
3.5 Renderização Não-Fotorrealista . . . 20

4 Método Proposto **23**
4.1 Simulação do Fluido . . . 23
4.1.1 Estrutura da Partícula . . . 24
4.1.2 Busca em Grade . . . 24
4.1.3 Propriedades Físicas do Fluido . . . 25
4.1.4 Integração . . . 27
4.1.5 Resultados . . . 27
4.2 Visualização Não-Fotorrealista do Fluido . . . 29
4.2.1 Extração de Características . . . 31
4.2.2 Linhas de Expressão Baseadas em Quantização . . . 31
4.2.3 Espuma . . . 33
4.2.4 Linhas de Expressão Baseadas em Partículas . . . 34
4.2.5 Renderização do Fluido Utilizando *Brushes* . . . 35
4.2.6 Desempenho Computacional . . . 38

5 Conclusões **40**
5.1 Análise do Trabalho . . . 40
5.2 Dificuldades e Trabalhos Futuros . . . 40

Referências Bibliográficas **42**

Lista de Figuras

2.1 Abordagem Euleriana do fluido. Os círculos representam os elementos de fluido fixos na grade de simulação e as setas mostram o escoamento do fluido observado em alguns destes pontos. 7

2.2 Representação bidimensional de um fluido lagrangiano. Cada partícula carrega as quantidades físicas do fluido (massa, densidade, velocidade, etc.) ao longo da simulação. 8

2.3 Núcleo de suavização W. 10

2.4 Núcleos de suavização W_{poly6}, W_{spiky} e $W_{viscosity}$ (esquerda para a direita). O Núcleo de suavização, seu Gradiente e Laplaciano são representados pelas cores azul, vermelho e verde, respectivamente. Ilustração adaptada de [1]. 11

3.1 Representação da abordagem de visualização Ray Tracing. 15

3.2 Visualização de uma superfície implícita utilizando o método baseado em pontos. 16

3.3 Partículas renderizadas como esferas. 17

3.4 Profundidade do fluido obtida através do *Z-Buffer*. 18

3.5 Normais do fluido. 19

3.6 Superfície do fluido reconstruída pelo algoritmo de LAAN. 20

3.7 (a) Espessura do líquido. (b) Fluido transparente. 21

4.1 Divisão em grade do espaço da simulação 2D para otimizar a busca por partículas vizinhas. 25

4.2 Dois quadros do SPH 3D desenvolvido com base no trabalho de MÜLLER *et al.* 28

4.3 Desempenho computacional em FPS das simulações 2D e 3D conforme o aumento no número de partículas utilizadas. 29
4.4 *Framework* de processamento não-fotorrealista de imagens. Figura adaptada de [2] 30
4.5 Principais propriedades do fluido fornecidas pelo SPH. (a) Densidade e (b) Velocidade das partículas. 31
4.6 Extração das bordas suavizadas do fluido. (a) Partículas do SPH. (b) Profundidade obtida pelo *z-buffer*. (c) Bordas do fluido calculadas utilizando a Diferença das Gaussianas. (d) Aplicação de cor ao fluido e composição com as bordas. 32
4.7 Obtendo linhas de expressão do fluido. (a) Quantização da profundidade. (b) Bordas do fluido calculadas utilizando a Diferença das Gaussianas. (c) Aplicação de cor ao fluido e composição com as bordas. 32
4.8 (a) Regiões convexas e côncavas do fluido, representadas pelos ângulos vermelhos e amarelos, respectivamente. Sendo que apenas as partículas com alta curvatura são identificas como estando na crista (imagem adaptada de [3]). (b) Partículas classificadas como espuma. (c) Composição da espuma com as bordas e o fluido. 33
4.9 Sequência de uma simulação exibindo linhas de expressão do fluido que denotam sua direção e velocidade. 35
4.10 (a) Mapa de altura de um *brush* e as normais calculadas. (b) Máscara. (c) Aplicação do mapa de altura sobre a pincelada. 36
4.11 *Brushes* calculados a partir do campo velocidade das partículas. . . . 37
4.12 *Brushes* utilizados para preencher os espaços em branco do fluido. . . 37
4.13 Máscara do fluido. 38
4.14 Composição final dos *brushes* renderizados nos passos anteriores e recortados pela máscara do fluido. 39

Lista de Tabelas

4.1 Desempenho computacional, em FPS, dos algoritmos de renderização não-fotorrealista de SPH. 39

Lista de Símbolos

W	Núcleo de suavização, p. 9
μ	Coeficiente de viscosidade, p. 4
σ	Coeficiente de tensão superficial, p. 27
Ω_i	Operador de colisão, p. 8
∇^2	Operador de Laplace, p. 4

Lista de Abreviaturas

API	Application Programming Interface, p. 28
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico, p. v
CPU	Central Processing Unit, p. 24
DoG	Difference of Gaussians, p. 29
FPS	Frames Por Segundo, p. 22
GLSL	OpenGL Shading Language, p. 23
GPU	Graphics Processing Unit, p. 23
LBM	Lattice Boltzmann Method, p. 8
LCG	Laboratório de Computação Gráfica, p. v
NPR	Non-Photorealistic Rendering, p. vii
SPH	Smoothed-Particle Hydrodynamics, p. vii

Capítulo 1

Introdução

A simulação de fluidos baseada em partículas desperta grande interesse de parte da comunidade científica [4], sendo que muitas desses contribuições se focam na etapa de visualização dos fluidos simulados. Apesar de um número considerável de trabalhos envolvendo a etapa de renderização, a maioria destes se voltam para a visualização realista de um fluido ([4], [5] e [6]), enquanto que alguns poucos são focados na representação não-realista do líquido.

Dentre estes trabalhos envolvendo renderização não-fotorrealista, apenas uma quantidade ínfima busca se aprofundar na visualização não-realista de sistemas de partículas. Apesar disso, é grande a quantidade de animações e jogos que precisam representar líquidos de forma não-fotorreal, sendo que muitas dessas representações ainda são realizadas de forma manual pelo artísta. Essa falta de soluções de renderização não-fotorrealista de líquidos motivou o desenvolvimento da presente dissertação visando o preenchimento desta lacuna e o incentivo às pesquisas voltadas para a área.

1.1 Objetivo

Esta dissertação tem como objetivo a implementação de um simulador de fluidos Lagrangiano baseado em partículas e, a partir deste, desenvolver técnicas de visualização não-fotorrealistas que aproveitam as características distintas em sistemas SPH. Acreditamos que a aplicação desses algoritmos de renderização não-fotorrealista em tempo-real de líquidos poderá contribuir para a indústria

de animação e de jogos, auxiliando o artísta na etapa de criação de animações não-fotorreais que envolvam a utilização de simulação de líquidos.

1.2 Organização da tese

Com o propósito de uma representação mais formal e científica de um fluido, realizamos no Capítulo 2 uma breve explanação sobre os conceitos principais da mecânica de fluidos, além disso, no Capítulo 3 também é apresentada uma revisão sucinta sobre a visualização de fluidos. O Capítulo 4 expõe o método proposto neste trabalho e os resultados obtidos. Finalmente, as conclusões finais sobre o trabalho desenvolvido são realizadas no Capítulo 5 e, em seguida, são apresentadas as referências bibliográficas.

Capítulo 2

Simulação de Fluidos

Sendo o fluido parte fundamental deste trabalho, torna-se crucial um melhor entendimento sobre as características deste elemento, em vista disso, neste capítulo realizamos uma breve discussão sobre os conceitos básicos da mecânica dos fluidos e, em seguida, descrevemos as principais abordagens utilizadas para a simulação computacional de fluidos focando principalmente na abordagem lagrangiana, por se tratar da utilizada no desenvolvimento do trabalho.

O conhecimento dos princípios e conceitos básicos da mecânica de fluidos é fundamental para qualquer aplicação que trabalhe com este meio. Como exemplos podemos citar: a análise aerodinâmica no desenvolvimento de meios de transporte, abrangendo desde motocicletas a jatos supersônicos; produções cinematográficas e animações; energia hidráulica; aplicações biológicas e médicas; metereologia; e inclusive situações corriqueiras, como encher um copo com água ou a trajetória curva de uma bola de futebol ao se cobrar um escanteio, podem ser compreendidas pela mecânica dos fluidos.

2.1 Teoria dos Fluidos

Inicialmente deve-se caracterizar o que é um fluido. Apesar de possuírmos o senso comum de que o fluido é uma substância que escoa ao se interagir com ela, e que é oposta aos sólidos, que tendem a se deformar ou dobrar, para a ciência, é necessária uma descrição mais formal e precisa do que é um fluido. Segundo FOX *et al* [7], um fluido é uma substância que se deforma continuamente quando submetida a uma

tensão de cisalhemento por menor que esta seja. Devido a incapacidade de suportar esta tensão, os fluidos se caracterizam por não resistirem à deformação e possuírem a capacidade de tomar a forma de seus recipientes (também conhecida como fluir). Incluem-se nos fluidos os líquidos, os gases e os plasmas.

As leis básica aplicadas a qualquer fluido são:

1. Conservação de massa.
2. Princípio fundamental da dinâmica.
3. Princípio da quantidade de movimento angular.
4. Primeira lei da termodinâmica.
5. Segunda lei da termodinâmica.

Nem todas as leis básicas são imprescindíveis para se resolver um problema. Entretanto, em muitos problemas faz-se necessário analisar relações adicionais que descrevem o comportamento das propriedades físicas do fluido sobre determinadas condições.

2.1.1 Equações de Navier-Stokes

Um fluido isotérmico e incompressível é composto pelos campos velocidade v, pressão p e densidade ρ. Sendo estes governados pelas equações de conservação de massa (também conhecida como equação de continuidade) e conservação do momento. Como no método Lagrangiano a simulação consiste de um número constante de partículas que possuem uma massa fixa, ao adortarmos esta abordagem a conservação de massa é inerente, podendo então sua equação ser omitida [1].

A equação de conservação do momento é conhecida como Equação de Navier-Stokes e é utilizada para descrever o movimento e evolução do fluido. A Equação de Navier-Stokes para fluidos incompressíveis é descrita da seguinte forma:

$$\rho\left(\frac{\partial v}{\partial t} + v \cdot \nabla v\right) = -\nabla p + \mu\nabla^2 v + pg \tag{2.1}$$

Em que g são as forças externas, μ o coeficiênte de viscosidade e ∇^2 o Laplaciano da velocidade v.

Como as partículas se movem com o fluido carregando seus atributos, a expressão $\frac{\partial v}{\partial t} + v \cdot \nabla v$ pode ser substituída pela derivada material $\frac{Dv}{Dt}$ [1]. Deste modo, a Equação de Navier-Stokes Lagrangiana fica:

$$\rho \frac{dv}{dt} = -\nabla p + \mu \nabla^2 v + pg \tag{2.2}$$

A Equação de Navier-Stokes trata-se da aplicação da segunda Lei de Newton aos fluidos, desta forma, os três termos no lado direito da Equação (3.2) calculam as forças internas e externas sobre o fluido, e podem ser utilizados para obter a aceleração a e a velocidade das partículas.

$$f = -\nabla p + \mu \nabla^2 v + pg \tag{2.3}$$

$$a_i = \frac{dv_i}{dt} = \frac{f_i}{\rho_i} \tag{2.4}$$

O primeiro termo

$$-\nabla p \tag{2.5}$$

representa a força exercida pela pressão que é a principal responsável pelas forças de atração e repulsão entre partículas vizinhas. Em uma região do fluido onde a densidade de partículas é grande, a força da pressão aumentará gerando forças repulsivas que afastarão as partículas. Por outro lado, em regiões onde o fluido apresenta pouca densidade, a pressão será responsável por manter as partículas unidas. Este termo tem como principal objetivo equalizar as diferenças de pressão pelo fluido [8].

O segundo termo apresenta a força de viscosidade

$$\mu \nabla^2 v \tag{2.6}$$

que define a resistência interna do fluido com relação ao escoamento das partículas, sendo o termo μ responsável por indicar o quão forte ele age sobre o fluido. O Laplaciano do campo velocidade fornece a diferença entre a velocidade da partícula e o valor médio do campo, forçando uma suavização da diferença entre velocidades.

O terceiro e último termo

$$pg \tag{2.7}$$

engloba todas as forças externas que agem sobre o fluido tais como gravidade, vento, tensão superficial e colisões.

2.2 Tipos de Abordagem

2.2.1 Método Euleriano

Na abordagem euleriana, o domínio da simulação é discretizado em uma malha fixa em que os valores das propriedades do fluido somente podem ser avaliados em pontos discretos do domínio. Ou seja, o elemento de fluido não se move conforme o escoamento mas se mantém fixo em uma determinada coordenada espacial onde é realizada a leitura das quantidades físicas (densidade, temperatura, velocidade, etc.) relacionadas ao elemento do fluido em específico. Adotando essas premissas, o movimento de um fluido eulerino pode ser descrito pela equação:

$$\frac{\partial v}{\partial t} = -\left(v \cdot \nabla\right) v - \frac{1}{\rho}\nabla p + f \tag{2.8}$$

em que v representa a velocidade, ρ a densidade, p a pressão e f as forças externas que atuam sobre o fluido. O termo $\left(v \cdot \nabla\right) v$ é conhecido como *derivada convectiva* e mede a variação de uma quantidade física (neste caso a velocidade) de um elemento de fluido em movimento [9]. A Figura 2.1 representa a visão euleriana de um fluido bidimensional.

Embora a abordagem euleriana proporcione uma melhor descrição de algumas quantidades físicas como densidade e pressão, sua grande desvantagem está na natureza fixa da grade, fazendo com que o fluido fique confinado, não podendo existir fora desta. Tal característica pode tornar inviável a simulação de cenários em que o fluido escorre livremente [10]. Para uma leitura mais profunda sobre a teoria e implementação de fluidos eulerianos veja [11].

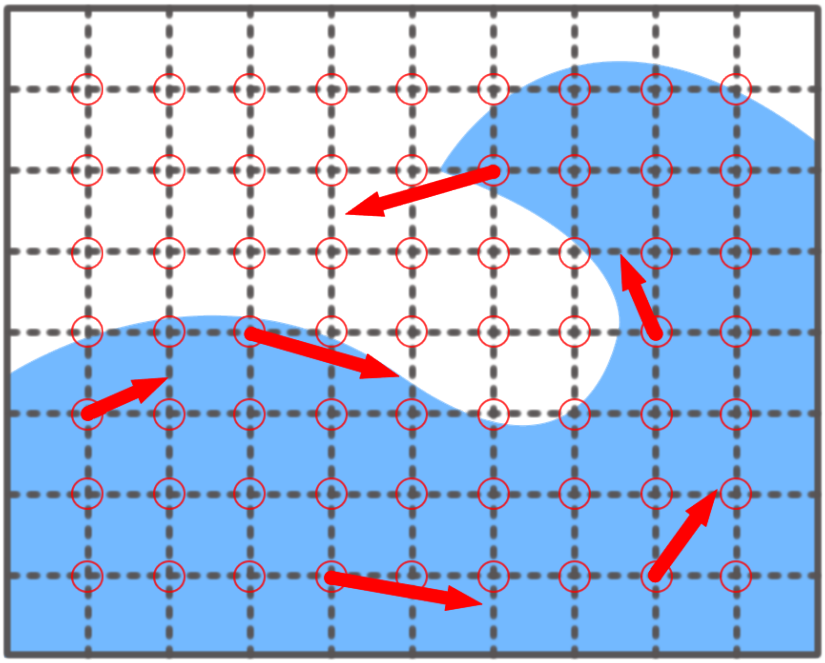

Figura 2.1: Abordagem Euleriana do fluido. Os círculos representam os elementos de fluido fixos na grade de simulação e as setas mostram o escoamento do fluido observado em alguns destes pontos.

2.2.2 Método Lagrangiano

Diferente do método euleriano, a abordagem lagrangiana descreve o fluido como um conjunto discreto de partículas em que cada partícula carrega as propriedades do fluido. Dependendo das forças aplicadas, as partículas se movem livremente e podem existir em qualquer lugar do domínio da simulação. A Figura 2.2 visualiza a representação bidimensional de um fluido lagrangiano.

Desenvolvido paralelamente por GINGOLD e MONAGHAN [12] e LUCY [13] para simular problemas astrofísicos, o método SPH tem sido usado de forma bem sucedida em uma grande variedade de problemas hidrodinâmicos e tornou-se uma abordagem popular para simular fluidos devido sua eficiência e implementação de baixa complexidade [8].

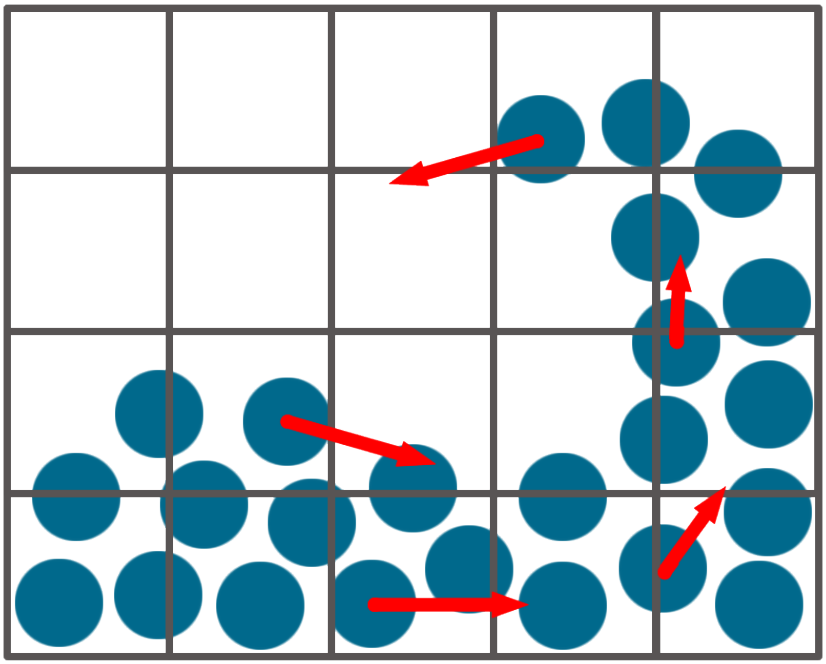

Figura 2.2: Representação bidimensional de um fluido lagrangiano. Cada partícula carrega as quantidades físicas do fluido (massa, densidade, velocidade, etc.) ao longo da simulação.

2.2.3 Método de Lattice Boltzmann

O método de Lattice Boltzmann (LBM) é considerado uma solução relativamente nova para aproximar a solução das equações fundamentais do fluido. Diferente dos métodos apresentados anteriormente, que resolvem as equações de Navier-Stokes para propriedades macroscópicas do fluido, como densidade e velocidade, o LBM é baseado em modelos microscópicos das equações [14].

A ideia fundamental é a construção de modelos cinéticos simplificados que utilizem os processos microscópicos e mesoscópicos de tal forma que a média das propriedades macroscópicas obedeça as equações determinadas, fazendo com que a equação de Lattice Boltzmann convirja para as equações de Navier-Stokes [15].

A equação de Lattice Boltzmann é a equação fundamental do LBM:

$$f_i(\mathbf{x} + \mathbf{e}_i \Delta \mathbf{x}, t + \Delta t) = f_i(\mathbf{x}, t) + \Omega_i(f(\mathbf{x}, t)) \tag{2.9}$$

Sendo f_i a função de distribuição de velocidade na i-ésima direção $\mathbf{e}_i$ e $\Omega_i(f(\mathbf{x}, t))$ o operador de colisão. Para uma explanação mais profunda sobre o método de Lattice Boltzmann indicamos a leitura de [16] em que os autores provam através de detalhadas derivações que a equação de Lattice Boltzmann é uma aproximação das equações de Navier-Stokes, quando observada macroscopicamente.

2.3 Simulação de Fluidos Baseada em Partículas

Devido à natureza finita das partículas Lagrangeanas não é possível representar o fluido de maneira contínua, por isso é necessária a utilização do método de interpolação SPH. Com o SPH, valores de campos, que antes eram definidos somente nas posições discretas das partículas, podem ser obtidos em qualquer ponto no fluido através de interpolação. Para isto, o SPH utiliza núcleos de suavização de tal maneira que a aproximação de uma grandeza A (escalar ou vetorial) em determinada posição r é dada pela soma ponderada das contribuições de todas as partículas vizinhas [8]:

$$A(r) = \sum_j m_j \frac{A_j}{\rho_j} W(r - r_j, h) \tag{2.10}$$

A distância h representa o alcance que o núcleo de suavização W possui e funciona como um fator de escala que controla a suavização do núcleo [10]. O fato de partículas que estão mais distantes que h não participarem dos cálculos será abordado mais à frente evidenciando sua importância no processo de otimização da performance computacional do sistema. A Figura 2.3 ilustra a atuação de um núcleo de suavização sobre partículas vizinhas.

2.3.1 Núcleo de Suavização

A precisão numérica e eficiência computacional de um sistema SPH está fortemente relacionada aos núcleos de suavização que serão implementados. Com base nisto, neste trabalho foram utilizados os núcleos desenvolvidos em [1] por serem simétricos, normalizados e apresentarem grande aceitação no meio acadêmico. Sendo que, devido às particularidades de cada quantidade física que será suavizada, foram implementados três núcleos de suavização: W_{poly6}, W_{spiky} e $W_{viscosity}$, responsáveis

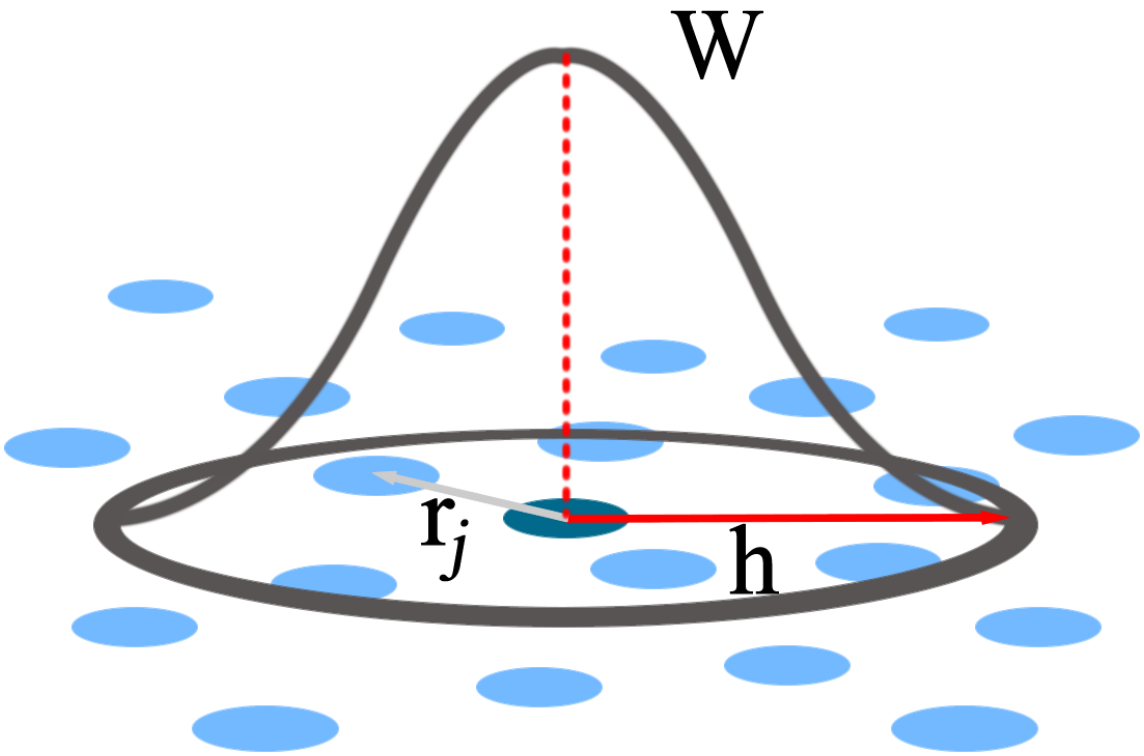

Figura 2.3: Núcleo de suavização W.

por suavizar os valores de densidade, pressão e viscosidade, respectivamente. Além disso, o núcleo W_{poly6} também suaviza a tensão superficial.

O principal núcleo apresentado por MÜLLER *et al* é:

$$W_{poly6}(r,h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \leq r \leq h \\ 0 & \text{caso contrário} \end{cases} \tag{2.11}$$

Uma das principais características deste núcleo é o fato de r estar elevado ao quadrado, fazendo com que não seja necessário o uso de raizes quadradas no cálculo de distâncias. Porém, como explanado em [1], quando este núcleo é utilizado na computação das forças de pressão as partículas tendem a se aglomerar em altas pressões. E, à medida que duas particulas se aproximam, a força de repulsão entre elas gerada pela pressão desaparece devido ao centro do núcleo se aproximar de zero, conforme observado na Figura 2.4.

Para contornar tal problema MÜLLER *et al* utiliza o núcleo *spiky* apresentado

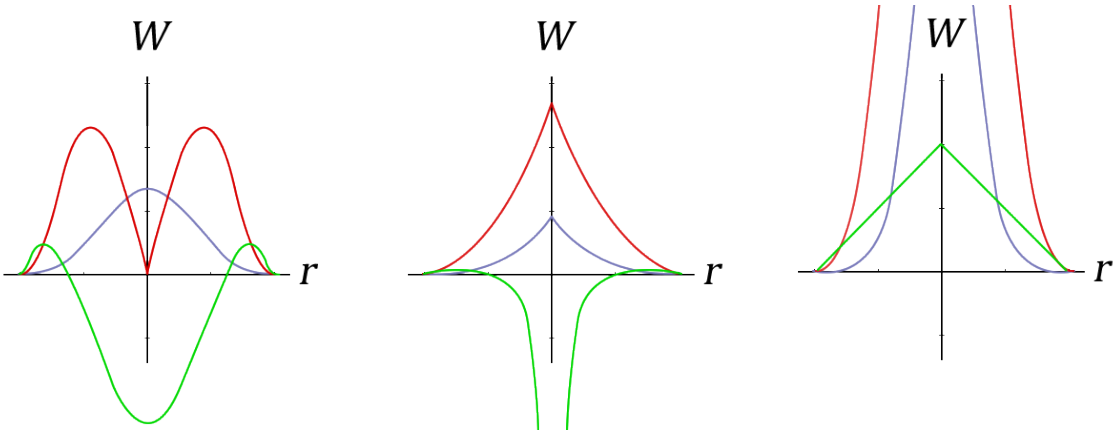

Figura 2.4: Núcleos de suavização W_{poly6}, W_{spiky} e $W_{viscosity}$ (esquerda para a direita). O Núcleo de suavização, seu Gradiente e Laplaciano são representados pelas cores azul, vermelho e verde, respectivamente. Ilustração adaptada de [1].

por [17]:

$$W_{spiky}(r,h) = \frac{15}{\pi h^6} \begin{cases} (h-r)^3 & 0 \leq r \leq h \\ 0 & \text{caso contrário} \end{cases} \tag{2.12}$$

MÜLLER *et al* observaram que a utilização de um destes dois núcleos de suavização no cálculo da viscosidade poderia ocasionar instabilidade no sistema, pois quando duas partículas se aproximam os valores do cálculo Laplaciano sobre o campo velocidade se tornam negativos. Problema este que se intensifica em aplicações em tempo real devido à baixa quantidade de partículas vizinhas. A solução encontrada pelos pesquisadores foi o desenvolvimento de um terceiro núcleo cujo Laplaciano é sempre positivo:

$$W_{viscosity}(r,h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1 & 0 \leq r \leq h \\ 0 & \text{caso contrário} \end{cases} \tag{2.13}$$

Além da utilização deste terceiro núcleo, também são necessárias as seguintes propriedades para a computação da viscosidade:

$$\nabla^2 W(r,h) = \frac{45}{\pi h^6}(h-r)$$
$$W(|r| = h, h) = 0$$
$$\nabla W(|r| = h, h) = 0$$

Capítulo 3

Visualização de Fluidos

Um dos principais objetivos da simulação computacional de fluidos encontra-se na visualização da superfície do fluido simulado, sendo que diversas abordagens vêm sendo apresentadas tanto no meio acadêmico quanto na indústria. Entretanto, a renderização de sistemas de fluidos baseados em partículas não é trivial, pois como apenas os valores das grandezas de uma partícula são informados pela simulação, a superfície do fluido não é obtida de forma natural.

Neste capítulo, descreveremos alguns métodos consolidados no meio científico para a visualização de sistemas SPH, focando nos métodos de renderização no espaço da imagem, devido esta ter sido a abordagem utilizada no desenvolvimento do trabalho.

3.1 Marching Cubes

Bastante utilizado pela área médica na visualização tridimensional de imagens de Ressonância Magnética e Tomografia Computadorizada, o algoritmo *Marching Cubes* desenvolvido por LORENSEN e CLINE [18], também é tradicionalmente utilizado na visualização de fluidos tanto lagrangianos quanto eulerianos. Para tal, o algoritmo necessita: de uma função de densidade que permita saber se um determinado ponto no espaço está dentro ou fora do líquido; e que o domínio seja dividido em uma grade de voxels ou células (no caso de simulações tridimensionais).

Para cada um dos oito cantos de um voxel é verificado, utilizando a função de densidade, se este está dentro ou fora da superfície. No caso de um mesmo

voxel possuir cantos dentro e fora do fluido, o algoritmo calcula os triângulos necessários para reconstruir a superfície. Uma vantagem dessa abordagem é o fato de que ao final do processamento é obtida uma malha tridimensional da superfície. Entretanto, a resolução da superfície está fortemente relacionada ao tamanho do voxel utilizado para subdividir o espaço da simulação. Quanto menor o voxel, maior a resolução, apesar disso, voxels muito pequenos fazem com que seja necessária uma quantidade muito grande de células para preencher o espaço da aplicação, o que afeta diretamente o desempenho computacional da abordagem. Outros pontos negativos do algoritmo são o fato de todos os voxels precisarem ser visitados para reconstruir a superfície e a necessedidade de reconstruir a superfície a cada iteração da simulação.

Diversas soluções foram propostas visando um melhor desempenho do algoritmo, como a aplicação de abordagens adaptativas, estruturas de dados baseadas em mapas, computação paralela, entre outros [19].

3.2 Ray Tracing

O *Ray Tracing* consiste em, a partir da posição do expectador, disparar raios em direção ao objeto a ser visualizado e, deste modo, obter os valores de profundidade e cor da superfície, preenchendo todos os pixels da imagem [20]. A Figura 3.1 apresenta uma representação do método.

Assim como o algoritmo *marching cubes*, o método de *ray tracing* não aproveita a coerência temporal intrínseca da simulação de fluidos, necessitando que a superfície seja recalculada a cada iteração. Apesar do alto custo computacional, a abordagem é uma das mais simples de serem implementadas, sendo uma boa opção para a renderização *offline* de superfícies [20].

3.3 Renderização Baseada em Pontos

A renderização baseada em pontos, apresentada por WITKIN e HECKBERT [21], consiste em criar e manter partículas que fluem livremente sobre a superfície para serem então utilizadas na visualização deste. Utilizando repulsão local, WITKIN e HECKBERT fazem com que as partículas flutuantes se espalhem e preencham a superfície de forma uniforme, como exemplificado na Figura 3.2.

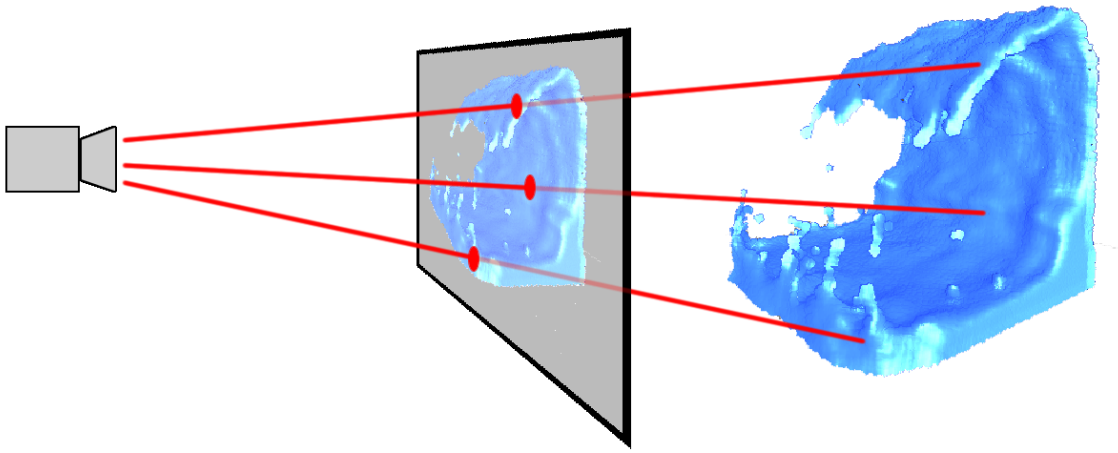

Figura 3.1: Representação da abordagem de visualização Ray Tracing.

Para que toda a superfície seja visualizada com alta qualidade, entretanto, é necessária uma grande quantidade de partículas flutuantes, o que aumenta o custo computacional da solução. Devido a esta característica e à complexidade de implementação do algoritmo, este método é pouco utilizado na visualização de superfícies implícitas, sendo preferíveis soluções igualmente custosas do ponto de vista computacional mas que geralmente são mais fáceis de serem implementadas como, por exemplo, o método de *ray tracing* [20].

Para um melhor entendimento sobre o método e uma implementação deste quase que totalmente em GPU, indicamos a leitura de [22].

3.4 Renderização no Espaço da Imagem

Diversas técnicas de visualização utilizando o conceito de *splatting* de partículas surgiram como alternativa para a renderização de superfícies. Dentre estas, o algoritmo desenvolvido por LAAN *et al* [4] apresenta grande vantagem na utilização de sistemas em tempo-real por ser totalmente implementado em placas gráficas. Esta propriedade fez com que o método fosse escolhido como passo inicial desta dissertação para a renderização do fluido simulado.

A abordagem de LAAN *et al* opera totalmente no espaço da imagem, sem geração

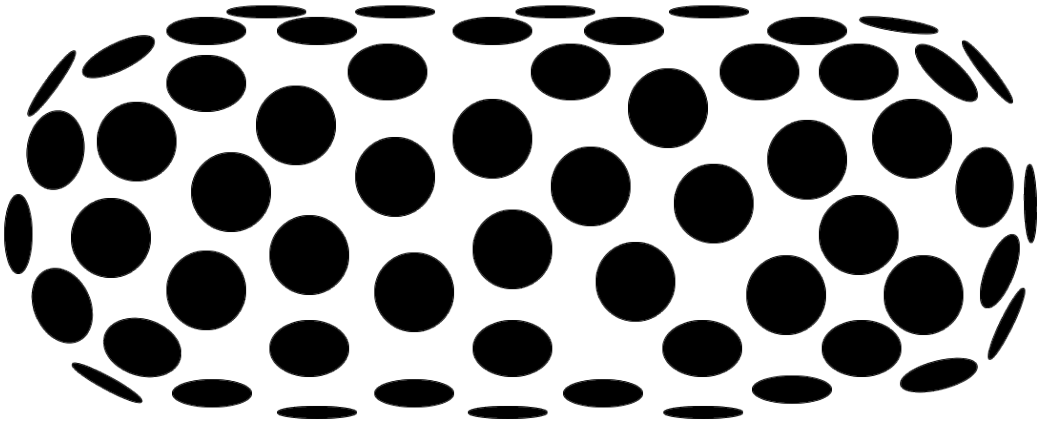

Figura 3.2: Visualização de uma superfície implícita utilizando o método baseado em pontos.

de malhas poligonais ou necessidade de divisão do espaço de simulação em uma grade. Desta forma, alcança grande performance computacional mas exige técnicas sofisticadas de processamento de imagens para se obter visualizações realísticas. Outra característica importante é a renderização somente das superfícies que são visualizadas pelo expectador, evitando o gasto computacional no processamento de áreas que não serão visualizadas (problema que ocorre na renderização por *marching cubes*).

As principais etapas de renderização do algoritmo de LAAN *et al* são explanadas nos subtópicos que seguem.

3.4.1 *Splats*

O primeiro passo necessário é a renderização das partículas como *point sprites*, também conhecidas como *splats*. Para isto, todas as partículas são enviadas ao *pipeline gráfico* para serem renderizadas como pontos.

Cada partícula determina o tamanho do ponto renderizado de acordo com a sua distância em relação ao observador. Desta forma, pontos que estão próximos ao usuário são visualizados como esferas grandes, enquanto que pontos mais afastados são visualizados como esferas menores, proporcionando assim uma sensação de

perspectiva da simulação.

Nesta etapa, verificando a densidade de cada partícula, pode-se eliminar a visualização daquelas que possuam poucas ou nenhuma partícula vizinha, o que geralmente causa instabilidade da partícula durante a simulação. A Figura 3.3 exibe a renderização da primeira etapa.

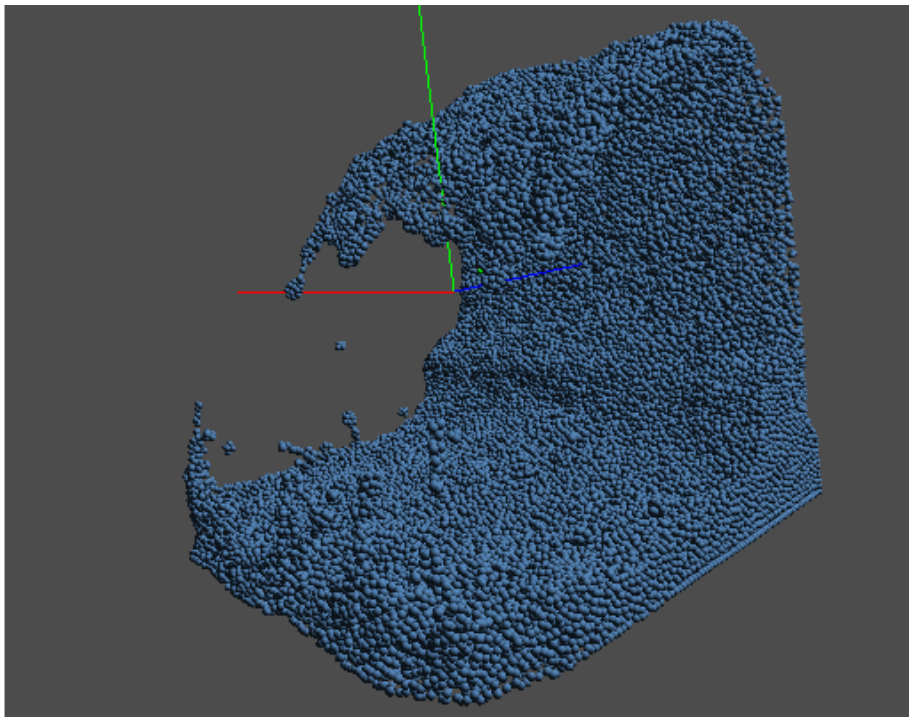

Figura 3.3: Partículas renderizadas como esferas.

3.4.2 Profundidade

Outra propriedade muito importante que auxilia as etapas seguintes é a profundidade das partículas. Para se obter a profundidade, é necessária a alocação de um *framebuffer* no qual o *pipeline gráfico* armazenará a imagem gerada pela técnica de *z-buffering*. A textura, vinculada ao *framebuffer*, contendo a profundidade poderá então ser utilizada por outras etapas. A Figura 3.4 apresenta o resultado do *z-buffer* recuperado em uma textura.

3.4.3 Suavização

Geralmente a superfície de um líquido, como a água, é idealizada como sendo plana e suave. Entretanto, a visualização das partículas como esferas acaba gerando uma

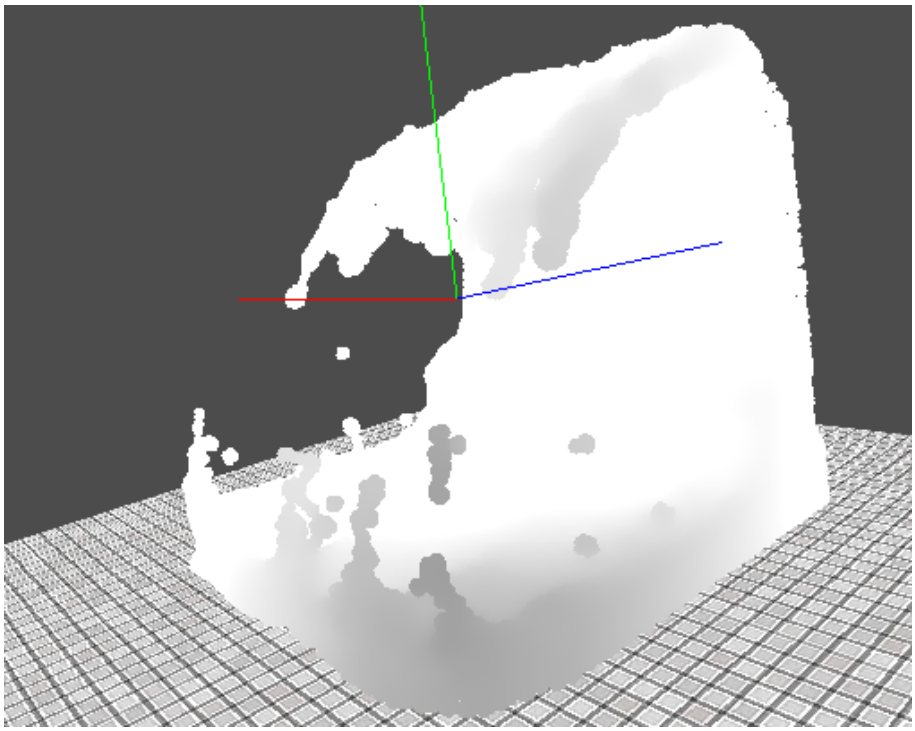

Figura 3.4: Profundidade do fluido obtida através do *Z-Buffer*.

superfície rugosa. Para contornar tal problema podemos recorrer a diversas técnicas de suavização aplicadas no espaço da imagem.

Uma possível solução é a utilização do filtro gaussiano de suavização. Porém, se aplicarmos o filtro gaussiano diretamente na imagem perderemos informações importantes na região das bordas do fluido [4], devido à sua ação uniforme sobre a imagem. Portanto, segundo GREEN [23], a aplicação do filtro gaussiano bilateral se torna interessante neste caso, por proporcionar uma suavização da imagem ao mesmo tempo em que preserva as bordas.

3.4.4 Normais

Outra propriedade muito importante calculada no algoritmo de LAAN *et al* é a normal das partículas, que é muito útil durante a etapa de renderização e na aplicação de efeitos como reflexão especular.

Para obter a normal de um pixel, LAAN *et al* utilizam diretamente a imagem suavizada da profundidade. Verificando-se os pixels vizinhos, a normal **n** em um

ponto P da imagem é obtida a partir do cálculo das derivadas parciais

$$\mathbf{n}(x, y) = \frac{\partial P}{\partial x} \times \frac{\partial P}{\partial y} \tag{3.1}$$

e, em seguida, armazenada em uma textura. Um exemplo das normais calculadas para um fluido pode ser observado na Figura 3.5.

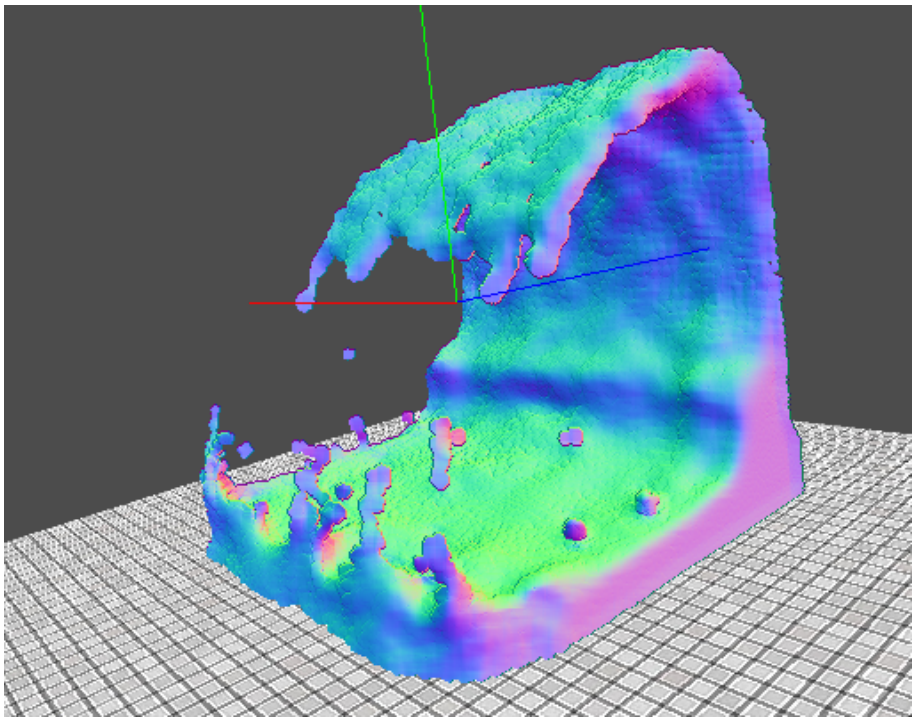

Figura 3.5: Normais do fluido.

3.4.5 Renderização

A renderização do fluido é a etapa final do algoritmo, em que os resultados calculados anteriormente são compostos a fim de visualizar a superfíce do líquido.

Esta etapa é bastante similar à aplicação de um modelo de iluminação, entretanto, para uma melhor verossimilhança com o tipo de reflexão de um líquido, LAAN *et al* se baseiam nas equações de Fresnel para modelar as propriedades ópticas do fluido, pois elas descrevem a quantidade de luz que é refletida e a quantidade que é refratada quando a luz se move entre meios com diferentes índices de refração (neste caso, o ar e o fluido). O resultado desse modelo de iluminação pode ser visualizado na Figura 3.6.

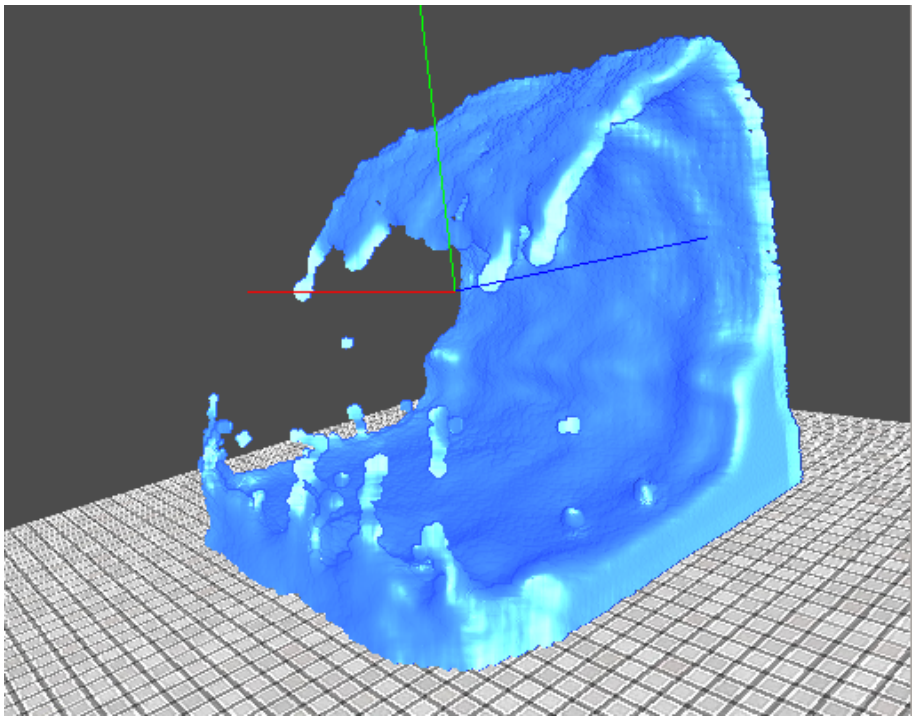

Figura 3.6: Superfície do fluido reconstruída pelo algoritmo de LAAN.

3.4.6 Transparência

Um efeito interessante que pode ser aplicado ao fluido é a transparência. Para alcançar tal efeito, LAAN *et al* renderizam as partículas utilizando *blending aditivo* sem teste de profundidade, ou seja, áreas que contenham mais partículas receberão uma contribuição maior, indicando regiões mais densas do fluido, portanto, menos transparentes.

Após ser suavizada, a imagem com a densidade do fluido é adicionada ao modelo de iluminação. A Figura 3.7 apresenta a textura com a densidade calculada e a superfície do fluido gerada a partir desta.

3.5 Renderização Não-Fotorrealista

Dentro da área de visualização de fluidos, a renderização não-fotorrealista destes ainda é pouco estudada pela comunidade. Enquanto a maioria dos trabalhos se voltam para a visualização realista de um fluido ([4], [5] e [6]), alguns poucos são focados na representação não-realista do líquido.

YOU *et al* [24], a partir de uma modificação no modelo Phong de iluminação,

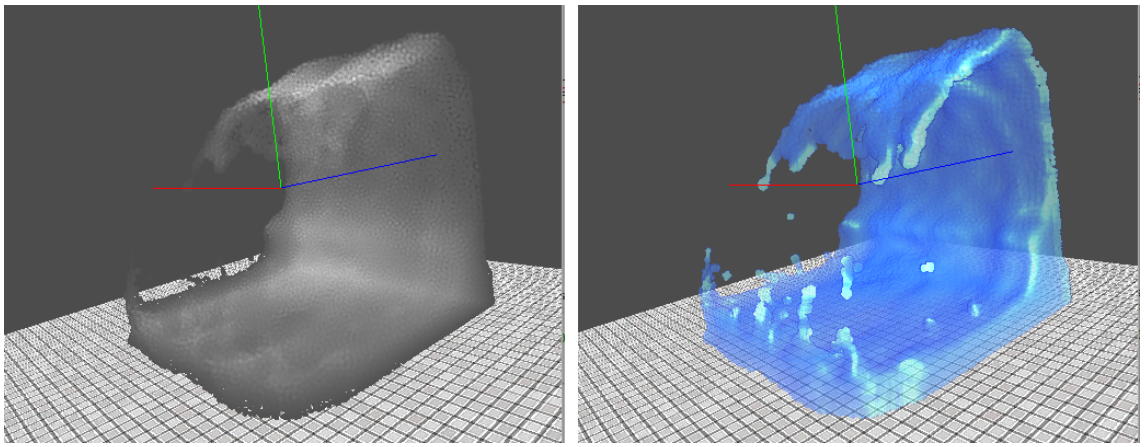

Figura 3.7: (a) Espessura do líquido. (b) Fluido transparente.

desenvolveram um *Cartoon Water Shader* que captura as principais características observadas em animações envolvendo água, tais como transparência, refração e reflexão. Apesar de YOU *et al* indicarem a equivalência de seus resultados com exemplos desenhados por artistas, este método não consegue reproduzir sprays d'água e bolhas. Além disso, a necessidade de se extrair a malha poligonal da superfície do fluido representa um grande empecilho no desenvolvimento de uma solução em tempo real.

O método proposto por EDEN *et al* [25] busca enfatizar diversas características do fluido, tais como descontinuidades na sua profundidade e também regiões próximas a silhuetas. Outra característica que é enfatizada são áreas onde o líquido é escasso, para tal o algoritmo utiliza *raycasting* para o cálculo dessas áreas. Por último, a fim de obter uma sensação de movimento na animação, o método realiza o tracking de texturas sobre a superfície do fluido. Apesar de apresentar resultados visualmente atrativos, a grande quantidade de computação envolvida no método dificulta a sua utilização em aplicações em tempo-real.

Diferente dos métodos anteriores que se baseiam na simulação física do líquido, YU *et al* [26] desenvolveram uma solução baseada em templates. Inicialmente o fluido é classificado entre diferentes tipos, e diferentes templates desenhados por artistas são então designados para cada tipo de fluido, mantendo a coerência temporal frame-a-frame. Alguns modelos de líquido são: ondas, jatos de água e correnteza. O sistema desenvolvido pelos autores pode ser aplicado tanto em 2D quanto em 3D, e também permite ao usuário interativamente alterar a posição,

tamanho e velocidade da animação do fluido.

Entre os estudos pesquisados nesta tese, o de NETO *et al* [27] é o que mais se aproxima do objetivo proposto neste trabalho. Utilizando como base uma simulação de fluidos baseada em partículas, NETO *et al* aplicam o método proposto em [4] para realizar a extração da superfície, que, por sua vez, é suavizada através de um filtro bilateral. A última etapa é a renderização do líquido aplicando um efeito não-realista. Para alcançar tal efeito, NETO *et al* basicamente verificam o ângulo entre as normais da superfície e a direção da luz, e aplicam um *Toon Shader* [28]. Como essas operações são realizadas no espaço da imagem, os autores conseguem alcançar uma performance na renderização de até 95 frames por segundo (FPS).

NETO e APOLINÁRIO [29] apresentam uma extensão do trabalho anterior, objetivando a renderização não-realista de um fluido com espuma. Para isto, na etapa de simulação, cada partícula do fluido contabiliza o número total de partículas vizinhas, com base neste número é gerada uma imagem contendo a espuma do líquido, que mais tarde é renderizada na etapa de composição final. Como no método anterior, este também trabalha no espaço da imagem, evitando técnicas de poligonização computacionalmente custosas e alcançando performance de até 42 FPS. Entretanto, o método ainda enfrenta problemas de flicker na transição entre partículas de fluido e de espuma.

Apesar de alguns trabalhos apresentados conseguirem alcançar o objetivo de renderização não-fotorealista em tempo-real a partir do processamento no espaço da imagem, acreditamos que esta abordagem limita a utilização do potencial da natureza particular de uma simulação SPH, e que o estudo de abordagens que aproveitem estas características do sistema de partículas pode originar o desenvolvimento de técnicas de renderização interessantes tanto para a pesquisa quanto para a indústria de animação.

Capítulo 4

Método Proposto

Neste capítulo apresentaremos as etapas e resultados do trabalho realizado durante o desenvolvimento desta dissertação. Para tal, a estrutura do capítulo será dividida em duas seções, sendo a primeira relacionada ao desenvolvimento do sistema SPH utilizado para simular o fluido, e a segunda relacionada à visualização não-fotorrealista da simulação.

4.1 Simulação do Fluido

O passo inicial para se visualizar um fluido é a sua simulação. Ao invés de adotarmos um dos diversos simuladores de fluidos que estão disponíveis ao público, optamos por implementar totalmente a abordagem SPH proposta por MÜLLER *et al* [1], desta forma, teremos total controle sobre a simulação, ajustando-a conforme as necessidades da visualização não-fotorrealista.

Como uma das propostas do trabalho é a renderização da simulação em tempo-real, precisamos que esta seja realizada dentro de um intervalo de tempo aceitável. Visando essa eficiência computacional, optou-se por utilizar o potencial da programação paralela em placas gráficas (GPU) na implementação do SPH. Deste modo, todo o código da simulação foi desenvolvido nas linguagens de programação C++ e GLSL, sendo esta última a linguagem de *shading* fornecida pela OpenGL para o desenvolvimento e controle direto do *pipeline de renderização*.

Para os trechos de código executados paralelamente na GPU, utilizamos o estágio de *Compute Shader* do *pipeline gráfico* (presente na OpenGL a partir da versão

4.3), que é utilizado para a computação de dados que não sejam necessariamente relacionados à renderização de pixels.

4.1.1 Estrutura da Partícula

No sistema, cada partícula é representada como um conjunto de *arrays* de *float* contendo as seguintes propriedades: posição, velocidade, densidade, pressão, viscosidade e *color field* (utilizado no cálculo da tensão superficial, como detalhado na Seção 4.1.3). Além das propriedades físicas do fluido, cada partícula possui um valor *int* único (identificador) que a diferencia de todas as outras partículas da simulação.

Inicialmente, os *arrays* das partículas são preenchidos na CPU com os valores padrão e, em seguida, enviados para a GPU, onde os cálculos serão realizados. Estes *arrays* são mantidos na placa gráfica durante toda a execução do programa, desta forma, evitamos o *overhead* causado pela transferência de memória entre a CPU e a GPU.

4.1.2 Busca em Grade

Uma caracterísa essencial de um sistema SPH é a interação que existe entre as partículas, sendo esta fundamental para o cálculo das forças que atuam sobre cada uma delas. Portanto, um algoritmo que permita acessar as propriedades das partículas de maneira eficiente é imprescindível para um sistema em tempo-real.

Como visto no Capítulo 2, o alcance do núcleo de suavização de cada partícula é limitado. Ou seja, somente as partículas que estão dentro deste alcance (determinado pelo usuário) contribuem para o cálculo das forças atuantes sobre uma partícula. Deste modo, evitamos que uma partícula precise acessar todas as outras existentes na simulação, visitando apenas as partículas vizinhas que atuam sobre esta. Entretanto, o acesso somente às partículas vizinhas não pode ser feito de forma trivial.

A abordagem utilizada para solucionar esta questão foi a apresentada por GREEN [23]. Nela, GREEN subdivide o espaço da simulação em uma grade de busca, como exemplificado na Figura 4.1. Assim, cada partícula verifica paralelamente a sua posição atual e informa a qual célula da grade ela pertence. Utilizando o identificador das células da grade como chave, ordenamos a lista de

identificadores das partículas e obtemos uma estrutura em que as partículas vizinhas são armazenadas próximas uma das outras no *array*.

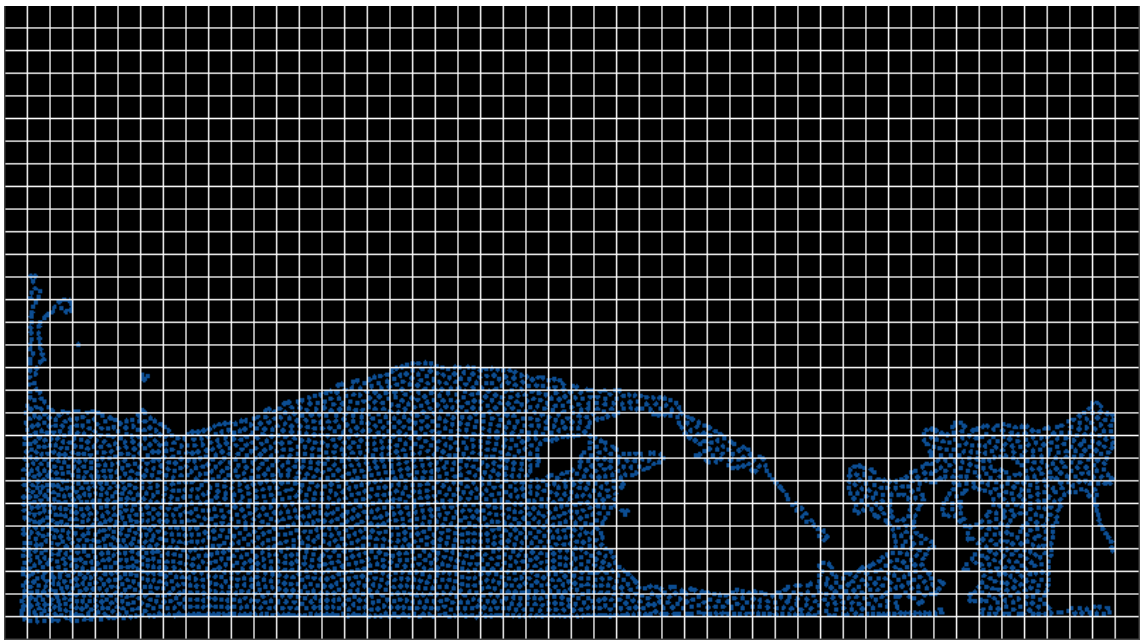

Figura 4.1: Divisão em grade do espaço da simulação 2D para otimizar a busca por partículas vizinhas.

O algoritmo de ordenação, então, torna-se o ponto crucial para uma busca em grade eficiente. Deste modo, decidimos implementar em GLSL o algoritmo de ordenação *radixsort* paralelo, assim como o algoritmo de soma do prefixo paralelo utilizado como uma etapa do *radixsort*. Para uma explicação mais aprofundada sobre os algoritmos paralelos de soma do prefixo e *radixsort* e sua implementação em CUDA, sugerimos a leitura de [30].

4.1.3 Propriedades Físicas do Fluido

A partir das partículas vizinhas, podemos calcular o valor das propriedades físicas do fluido em um ponto através dos núcleos de suavização. Utilizando os *buffers* armazenados na placa gráfica, calculamos todas as forças no *compute shader* com o objetivo de aumentar o desempenho computacional da aplicação, sendo que cada partícula é representada por uma *thread* que é executada em paralelo.

Como o cálculo de todas as forças utilizadas no SPH dependem da densidade, ela é a primeira propriedade obtida no sistema. Assim, a densidade ρ de uma partícula

i é calculada com base nas j-ésimas partículas vizinhas de i de acordo com a fórmula apresentada por MÜLLER *et al* [1]:

$$\rho_i = \sum_j m_j W_{poly6} \left(r_i - r_j, h \right) \tag{4.1}$$

A pressão p de uma partícula é obtida utilizando uma versão modificada da equação de estado do gás ideal proposta por DESBRUN e CANI [17] e, a partir da densidade e da pressão, calculamos a força de pressão da seguinte forma:

$$\mathbf{f}_i^{\text{pressão}} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \nabla W_{spiky} \left(r_i - r_j, h \right) \tag{4.2}$$

A próxima força obtida é a de viscosidade, que além de ser aplicada como uma força amortizante, também contribui para a estabilidade da simulação. Esse cálculo é realizado por MÜLLER *et al* com base nas diferenças de velocidade segundo a equação:

$$\mathbf{f}_i^{\text{viscosidade}} = \mu \sum_j m_j \frac{v_i - v_j}{\rho_j} \nabla^2 W_{viscosity} \left(r_i - r_j, h \right) \tag{4.3}$$

A última propriedade necessária para a etapa de integração do SPH é a tensão superficial, para isto, MÜLLER *et al* adicionam ao modelo um campo em que atribuem o valor 1 às posições que contenham partículas e 0 caso contrário, desta forma, atribuem um valor, ou "cor", para cada meio presente na simulação (geralmente ar e líquido). Este campo é identificado como *color field* e sua suavização é obtida através do seguinte modo:

$$c_s(\mathbf{r}) = \sum_j m_j \frac{1}{\rho_j} W_{poly6}(\mathbf{r} - \mathbf{r}_j, h) \tag{4.4}$$

Sendo que o gradiente $\mathbf{n}$ deste campo para uma determinada partícula i

$$\mathbf{n}_i = \sum_j m_j \frac{1}{\rho_j} \nabla W_{poly6}(r_i - r_j, h) \tag{4.5}$$

nos fornece a normal da superfície e o quão próxima a partícula i se encontra da superfície do fluido. Além da normal, também podemos obter a curvatura da

superfície através da equação:

$$k = -\frac{\nabla^2 c}{||\mathbf{n}||} \tag{4.6}$$

Deste modo, podemos finalmente calcular a tensão superficial:

$$\mathbf{f}_i^{\text{superfície}} = \sigma k \mathbf{n} = -\sigma \nabla^2 c_i \frac{\mathbf{n}_i}{||\mathbf{n}_i||} \tag{4.7}$$

Em que o coeficiente σ indica a força com que a superfíce se sustém. Uma vez que todas as forças sejam calculadas, inicia-se a etapa de integração destas.

4.1.4 Integração

Na etapa de integração do sistema, as propriedades de velocidade e posição das partículas são integradas a fim de mover as partículas pelo espaço de simulação. Inicialmente, foi utilizada a integração euleriana por motivo de simplicidade, entretanto, com o objetivo de aumentar a estabilidade da simulação, foi adotado o método de integração *leapfrog*.

O método *leapfrog* consiste em atualizar as velocidades nos intervalos entre as atualizações das posições. Este "salto"entre as atualizações é o que origina o nome do método de integração. Apesar de ser um tipo de integração simples como a integração euleriana, o *leapfrog* se mostra mais estável ao decorrer da simulação.

Nesta etapa também são adicionadas as forças externas, como a gravidade, e é realizada a colisão das partículas com as bordas que delimitam o espaço da simulação. Ao invés de simplesmente rebater a velocidade da partícula quando esta alcança um dos limites, optamos por utilizar uma força de repulsão que impede as partículas de sairem da área de simulação. Esta abordagem permite ajustarmos de forma mais precisa a força com que as partículas são amortizadas e repelidas.

4.1.5 Resultados

A fim de se testar somente o desempenho do SPH 3D desenvolvido no trabalho, utilizamos nos testes uma visualização simples baseada em *splatting* que não teria um impacto significativo no tempo de execução da aplicação. Um exemplo da execução do SPH desenvolvido pode ser visualizado na Figura 4.2.

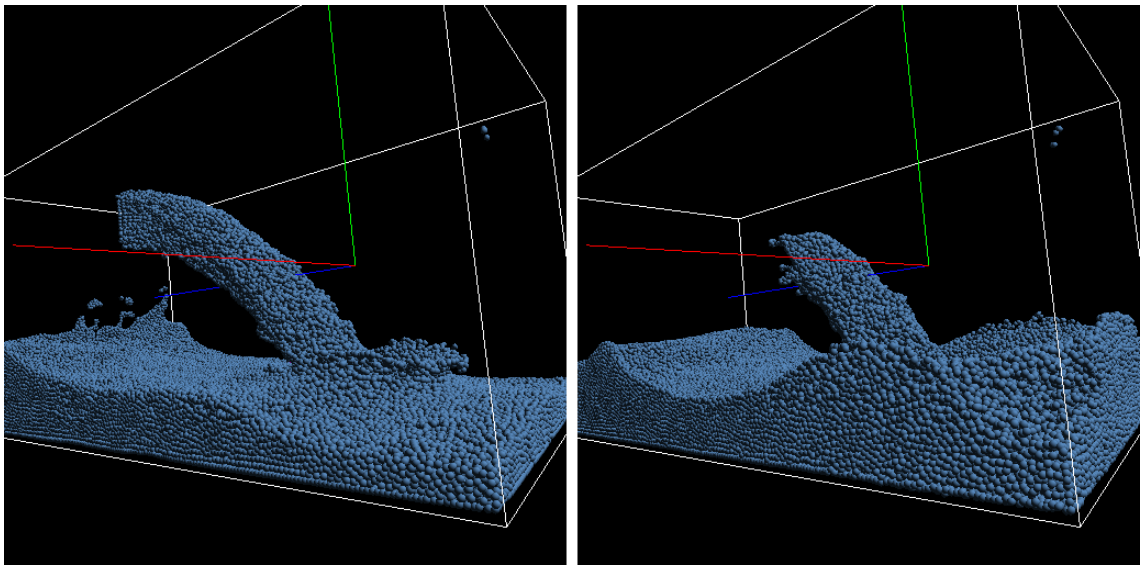

Figura 4.2: Dois quadros do SPH 3D desenvolvido com base no trabalho de MÜLLER *et al.*

O SPH foi executado em um computador com as seguintes configurações: sistema operacional Linux 64-bit, processador Intel Core i7 com frequência de 3.4 GHz e 16GB de memória RAM. Além disso, a placa gráfica utilizada nas simulações foi uma GeForce GTX 660 com 960 núcleos de processamento e 2048MB de memória, sendo que a OpenGL versão 4.3 foi a API utilizada para comunicação com a GPU.

Para o teste, foram elaborados dois cenários: o primeiro, em 2D, contendo até 16384 partículas e o segundo, em três dimensões, possuindo um máximo de 131072 partículas. Além disso, o espaço bidimensional da simulação foi dividido em 28 x 50 células e o tridimensional em 25 x 25 x 50 voxels. Após seguidas execuções, mesmo variando a quantidade de partículas, a média do tempo computacional do primeiro caso foi 58 FPS. No segundo caso, nota-se que a simulação também consegue manter um desempenho em torno de 58 FPS até uma quantidade de 65536 partículas, entretanto, conforme o aumento no número de partículas, há uma queda no desempenho computacional que chega até 28 FPS, evidenciando o impacto que a quantidade de partículas possui sobre o tempo de execução do SPH, provavelmente devido à um maior armazenamento de dados na memória global da GPU, que apesar de possuir um tamanho maior apresenta um acesso mais lento quando comparada aos outros tipos de memória da placa gráfica. Apesar da queda de desempenho no segundo cenário, acreditamos que os dois casos apresentaram tempos de execução

satisfatórios para uma aplicação em tempo-real. O desempenho computacional nos dois casos pode ser observado na Figura 4.3.

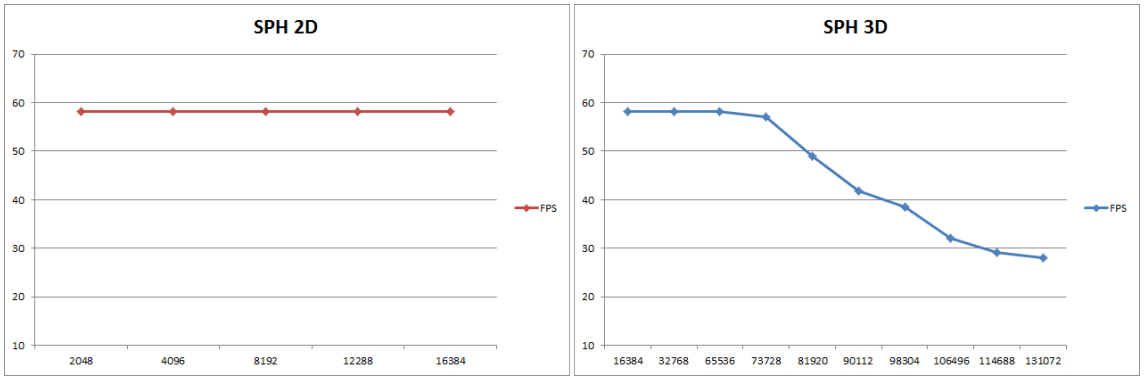

Figura 4.3: Desempenho computacional em FPS das simulações 2D e 3D conforme o aumento no número de partículas utilizadas.

4.2 Visualização Não-Fotorrealista do Fluido

O primeiro passo realizado na dissertação para a renderização não-fotorrealista do SPH construído na seção anterior foi a implementação de ferramentas básicas consolidadas na área de NPR, tais como extração de silhuetas, suavização de imagem, quantização de cores, etc. Para isto, foi implementado o sistema de abstração de imagens desenvolvido por KYPRIANIDIS e DÖLLNER [2], que aplica diversas técnicas de NPR que serão futuramente utilizadas neste trabalho para a visualização de fluidos. Outra característica importante da abordagem de KYPRIANIDIS e DÖLLNER é sua implementação em GPU, o que contribui para um melhor desempenho computacional da visualização.

Os principais filtros desenvolvidos no *framework* são o *Separated Orientation-aligned Bilateral Filter*, utilizado na suavização da imagem, e o *Separated Flow-based DoG Filter*, responsável pela extração de silhuetas suaves. Sendo que ambos são auxiliados pela Orientação Local (*Local Orientation Estimation*) da imagem calculada a partir dos autovalores do *Tensor de Estrutura*, conforme explanado em [2].

Utilizando a orientação local, o *Separated Orientation-aligned Bilateral Filter* trata-se de uma aproximação do filtro gaussiano bilateral (expensivo

computacionalmente, pois apresenta complexidade $O(n^2)$ [31]) que consiste em uma implementação separada em duas passadas, a primeira filtrando a imagem na direção do gradiente, e a segunda na direção da tangente, desta forma, alcançando bons resultados em um tempo computacional menor quando comparado ao filtro gaussiano bilateral simples. Similiar ao filtro anterior, o *Separated Flow-based DoG Filter* também consiste em uma abordagem separada em duas passadas: a primeira aplica o filtro DoG na direção do gradiente, e a segunda suaviza e aplica um *threshold* ao longo das curvas inferidas pelas tangentes.

Outra ferramenta apresentada no sistema de KYPRIANIDIS e DÖLLNER, também utilizada nesta dissertação, é a quantização de cor aplicada no canal de luminância segundo o trabalho de WINNEMÖLLER *et al* [32]. A Figura 4.4 apresenta uma visão geral sobre o método de abstração de imagens desenvolvido com base em [2].

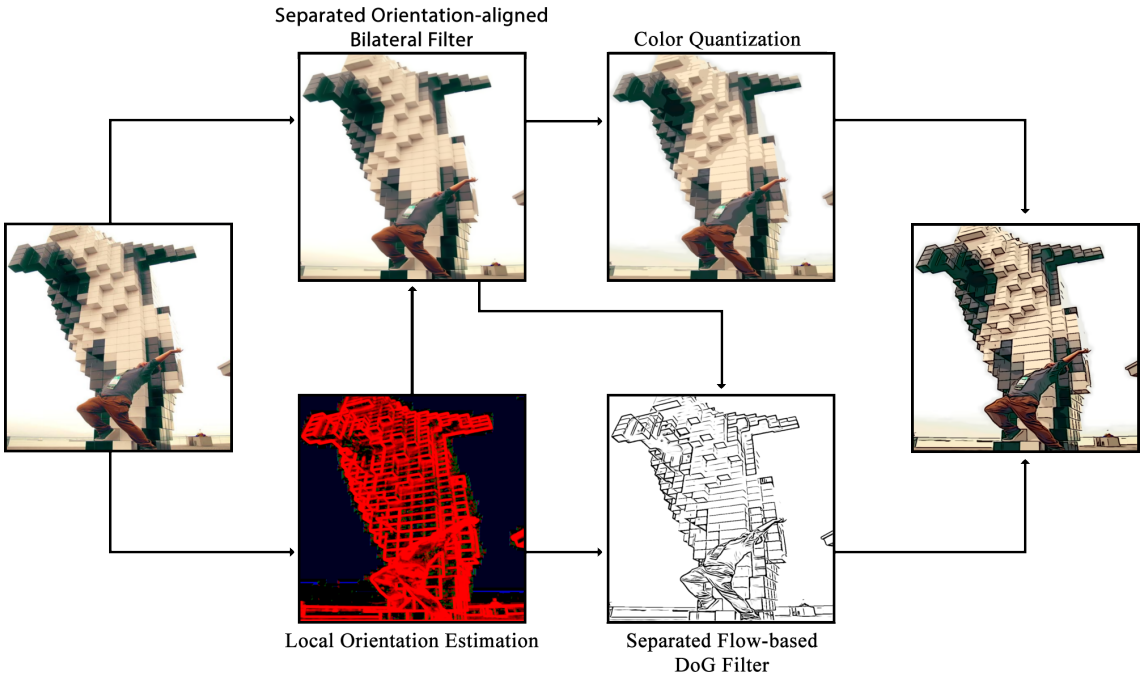

Figura 4.4: *Framework* de processamento não-fotorrealista de imagens. Figura adaptada de [2]

Como dito anteriormente no Capítulo 3, a renderização de sistemas de simulação de fluidos baseados em partículas não é trivial, pois as partículas do SPH fornecem somente os valores das propriedades do fluido, desta forma, a rendenderização não-fotorrealista da superfície será desenvolvida com base nessas propriedades.

Algumas delas são visualizadas na Figura 4.5.

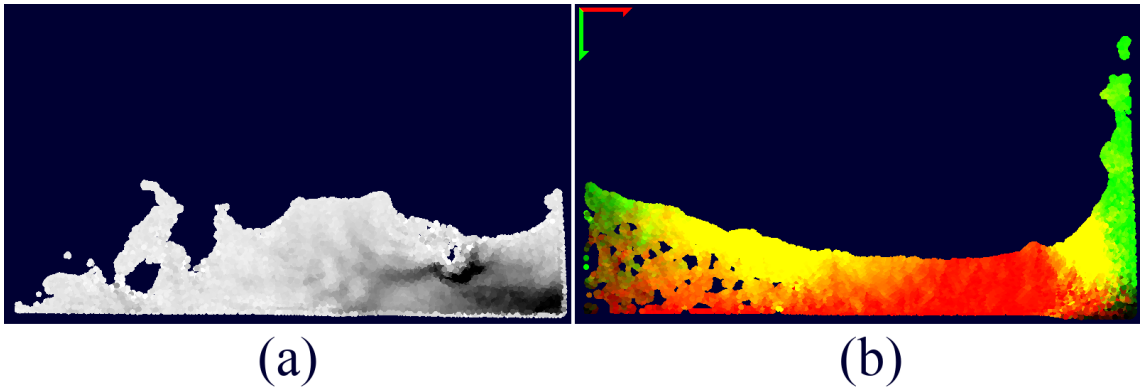

Figura 4.5: Principais propriedades do fluido fornecidas pelo SPH. (a) Densidade e (b) Velocidade das partículas.

4.2.1 Extração de Características

Por apresentarem grande quantidade de informação para o sistema visual humano sobre a representação da forma de um objeto, as bordas são bastante enfatizadas por artistas e designers [33]. Deste modo, a extração de silhuetas é um importante passo nas técnicas de NPR.

Com este objetivo, inicialmente obtemos a profundidade das partículas conforme explanado no Capítulo 3. Em seguida, a imagem é suavizada com o algoritmo *Separated Orientation-aligned Bilateral Filter* e, finalmente, extraímos as silhuetas do fluido utilizando o *Separated Flow-based DoG Filter*. Para exibirmos o resultado, renderizamos todas as partículas com a mesma cor e a imagem obtida é composta com as silhuetas conforme visualizado na Figura 4.6.

A fim de se evitar repetições no decorrer do texto, entende-se que todos os passos envolvendo suavização de imagens, extração de silhuetas e quantização nesta dissertação, foram realizados pelos algoritmos desenvolvidos em [2] e [32] citados anteriormente.

4.2.2 Linhas de Expressão Baseadas em Quantização

Utilizando como base as ferramentas dos passos anteriores, podemos desenvolver renderizações não-fotorrealistas. Para isto, foram analisadas diversas ilustrações

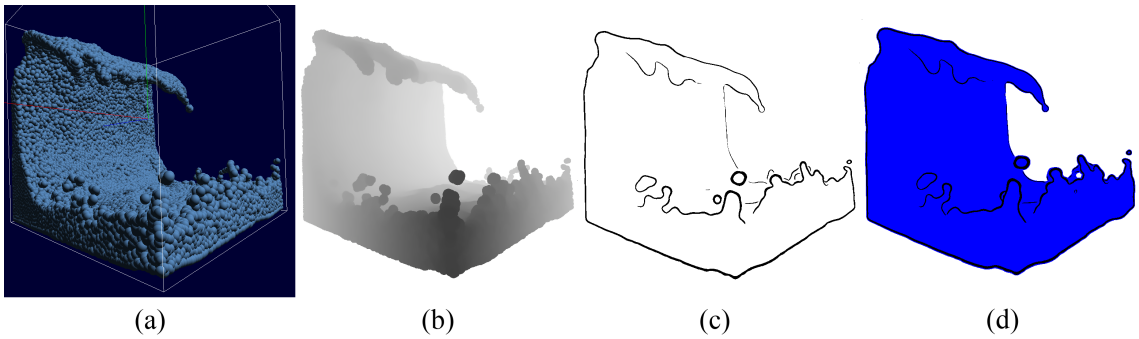

Figura 4.6: Extração das bordas suavizadas do fluido. (a) Partículas do SPH. (b) Profundidade obtida pelo *z-buffer*. (c) Bordas do fluido calculadas utilizando a Diferença das Gaussianas. (d) Aplicação de cor ao fluido e composição com as bordas.

produzidas por artistas a fim de se identificar propriedades que podem ser replicadas com o intuito de simular uma representação artística do fluido. Uma característica identificada, bastante recorrente nas ilustrações de líquidos, é a utilização de linhas desenhadas em favor do fluxo do fluido que expressam a ideia de movimento do líquido.

Para tentarmos reproduzir esse efeito, muito utilizado na representação de ondas, inicialmente quantizamos a imagem da profundidade das partículas. Assim, obtemos faixas de cores distintas com transições bruscas entre elas. A extração das silhuetas desta imagem resulta em linhas que percorrem toda a extensão do fluido, como mostrado na Figura 4.7.

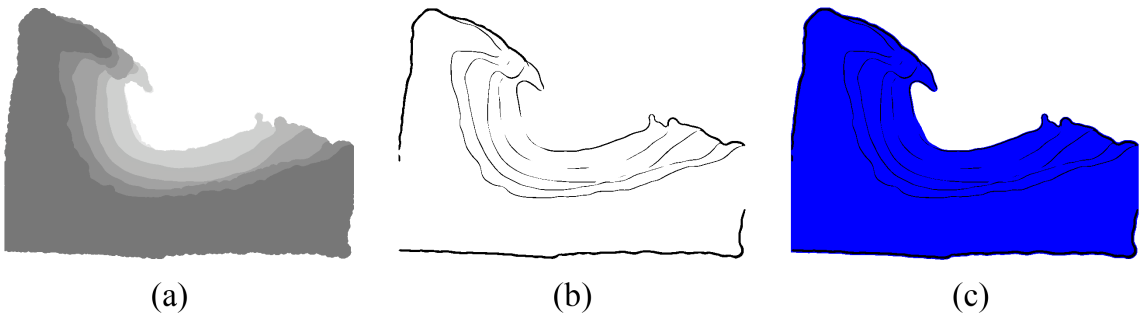

Figura 4.7: Obtendo linhas de expressão do fluido. (a) Quantização da profundidade. (b) Bordas do fluido calculadas utilizando a Diferença das Gaussianas. (c) Aplicação de cor ao fluido e composição com as bordas.

Apesar do efeito bastante interessante, esta abordagem é muito dependente da posição em que se encontra o obsevador e da direção em que o líquido está fluindo, pois estamos quantizando a profundidade obtida através do *z-buffer*, o que faz com que as linhas não respeitem o sentido do fluxo do líquido mas sim a posição da câmera. Além disso, como esta abordagem depende da profundidade, apenas a versão 3D foi implementada no trabalho.

4.2.3 Espuma

Nesta etapa, tentou-se reproduzir o efeito de espuma também bastante recorrente em ilustrações. Entretanto, determinar as regiões que conteriam espuma mostrou-se problemático. Como grande parte das representações artísticas posicionam a espuma sobre a crista de uma onda, voltamos nossa atenção para o problema de determinar quais partículas fazem parte da crista de uma onda na simulação. Segundo IHMSEN *et al* [3], a crista de uma onda em um SPH pode ser encontrada ao verificar se a superfície é convexa e a sua curvatura é alta, como ilustrado na Figura 4.8. Além disso, verificamos a densidade das partículas e identificamos como sendo espuma partículas com baixa vizinhança. A Figura 4.8 também apresenta os resultados obtidos nesta etapa.

Figura 4.8: (a) Regiões convexas e côncavas do fluido, representadas pelos ângulos vermelhos e amarelos, respectivamente. Sendo que apenas as partículas com alta curvatura são identificas como estando na crista (imagem adaptada de [3]). (b) Partículas classificadas como espuma. (c) Composição da espuma com as bordas e o fluido.

Um dos principais problemas enfrentados na renderização de espuma foi a

manutenção da coerência temporal desta, pois, devido à grande variação de densidade das partículas no decorrer da simulação, diversas partículas apresentaram o efeito de *flickering*, alterando entre os estados de "fluido"e "espuma". Tal efeito foi suavizado ao se manter um histórico de cada partícula, renderizando a partícula como espuma somente nos casos em que esta permanecesse no estado de"espuma"por uma quantidade determinada de frames. Como a implementação deste algoritmo é relativamente simples, optamos por desenvolver diretamente a versão 3D deste, sem a necessidade de uma versão 2D inicial.

4.2.4 Linhas de Expressão Baseadas em Partículas

Além do problema citado anteriormente relacionado às linhas de expressão criadas a partir da quantização da profundidade, esta abordagem torna-se limitada ao não utilizar as características oferecidas pelas partículas do SPH. Partindo deste princípio, procuramos desenvolver uma abordagem que aproveite as informações presentes nas partículas para criar as linhas de expressão do fluido.

Como a ideia dessas linhas é representar o fluxo do líquido, a escolha lógica é a utilização da velocidade das partículas. Com esta propriedade, podemos então estabelecer a direção em que uma partícula escorre e, deste modo, desenhar linhas que respeitem este fluxo, independente da posição da câmera. Na abordagem desenvolvida, cada partícula passa ao *Geometry Shader*, durante a execução do *pipeline gráfico*, um *GL_POINT* contendo a sua posição, e o *geometry shader* retorna um *GL_LINE_STRIP* contendo a linha de expressão da partícula.

No *geometry shader*, a partir da posição recebida, este se encarrega de verificar a velocidade da partícula e, utilizando o SPH, calcula o próximo ponto da linha baseado nesta velocidade. Esta computação é realizada por um número fixo de vezes determinado pelo usuário, mas apesar de todas as *GL_LINE_STRIP* possuirem a mesma quantidade de pontos, o tamanho das linhas de expressão varia por se basear na velocidade das partículas. Regiões do fluido que apresentam uma maior velocidade tendem a apresentar linhas de expressão longas, enquanto que regiões do líquido mais calmas renderizam linhas mais curtas.

Buscando um refinamento das linhas renderizadas, procuramos visualizar somente aquelas que mais se assemelham às produzidas por artistas. Sendo assim,

verificando os exemplos de desenhos manuais, estabelecemos que as linhas de expressão são aplicadas principalmente em regiões que apresentam mudanças na direção do fluxo (ondas, quedas d'água, entre outros) e velocidades mais altas (águas mais calmas, como lagos, geralmente são desenhadas com poucas linhas de expressão). Desta forma, antes de renderizar uma linha de expressão, verificamos o seu comprimento e a sua curvatura, dando prioridade para as linhas longas e curvas. A Figura 4.9 ilustra o resultado obtido após o refinamento das linhas de expressão.

Figura 4.9: Sequência de uma simulação exibindo linhas de expressão do fluido que denotam sua direção e velocidade.

Além de não depender da posição da câmera, utilizar a velocidade das partículas também fornece visualizações interessantes do fluido, como por exemplo, a criação de vórtices no fluido durante a simulação. Entretanto, esta versão possui um alto custo computacional quando comparada à anterior devido ao grande uso do SPH para calcular os pontos que formam uma linha de expressão. Após ser desenvolvida a versão 2D, implementamos a versão tridimensional do cálculo das linhas de expressão com o intuito de comparar, na Seção 4.2.6, o desempenho computacional do algoritmo em ambas as dimensões.

4.2.5 Renderização do Fluido Utilizando *Brushes*

Dentro da área de NPR, existem muitos trabalhos voltados para a renderização de imagens similares às pintadas manualmente. Dentre estes, utilizamos a pesquisa de HERTZMANN [34] como base para o desenvolvimento desta última abordagem. No trabalho citado, o autor "pinta" a imagem utilizando sucessivas pinceladas (*Brushes*) que são refinadas a cada passada do algoritmo até que toda a imagem seja finalizada.

Com base na pesquisa de HERTZMANN, tentamos renderizar o fluido de modo que a imagem final se assemelhe a uma pintura manual. Para isto, similar à abordagem anterior, utilizaremos as linhas originadas pelas velocidades das partículas. Porém, ao invés de renderizar um *GL_LINE_STRIP*, o *geometry shader* retornará um *GL_TRIANGLE_STRIP* em que serão renderizadas as texturas do *brush*. Apesar desta diferença, o cálculo no *geometry shader* é bastante parecido. A linha de expressão também é calculada a partir da velocidade da partícula, entretanto, sobre a linha, serão estabelecidos os triângulos que formam o *GL_TRIANGLE_STRIP*.

Após o processamento no *geometry shader*, triângulos calculados são enviados ao *fragment shader* junto com suas respectivas coordenadas de textura. Uma máscara contendo o formato da pincelada é então aplicada ao *GL_TRIANGLE_STRIP*. Em seguida, utilizando um mapa de altura, calculamos as normais da textura e aplicamos um modelo de iluminação para dar um aspecto mais realista ao *brush*, conforme mostrado na Figura 4.10.

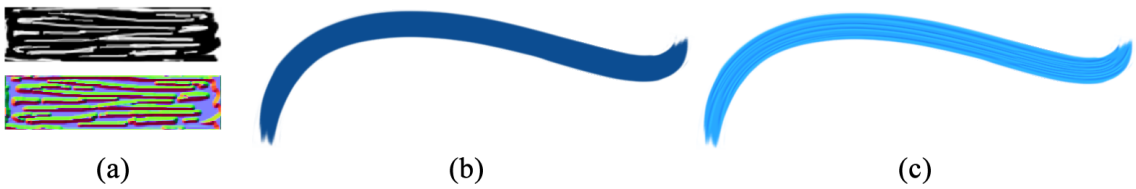

Figura 4.10: (a) Mapa de altura de um *brush* e as normais calculadas. (b) Máscara. (c) Aplicação do mapa de altura sobre a pincelada.

A cor do *brush* é determinada a partir da densidade da partícula originária, assim, áreas mais densas no fluido são renderizadas com pinceladas mais escuras, enquanto que áreas menos densas são representadas por cores mais claras. Utilizando esses *brushes*, tentamos preencher toda a área do fluido, entretanto, como algumas partículas apresentam velocidades próximas de zero, áreas no fluido deixam de ser preenchidas, como visualizado na Figura 4.11.

Para solucionar tal problema, verificamos as partículas que estão sobre regiões do fluido que não foram preenchidas pelo passo anterior. Em seguida, para cada uma delas, calculamos a média das velocidades de suas partículas vizinhas e renderizamos um *brush* de tamanho fixo de acordo com a direção da velocidade média obtida. A

Figura 4.11: *Brushes* calculados a partir do campo velocidade das partículas.

Figura 4.12 apresenta o resultado desta etapa.

Figura 4.12: *Brushes* utilizados para preencher os espaços em branco do fluido.

Como observado, esta etapa acaba preenchendo áreas fora do fluido. Deste modo, precisamos de uma máscara contendo a região total do líquido e, em seguida, realizar o corte dos preenchimentos excedentes. Esta máscara é obtida ao se renderizar todas as partículas utilizando a técnica de *splatting*. Um exemplo de máscara é apresentado na Figura 4.13.

Finalmente, compomos em uma imagem final as pinceladas baseadas na velocidade e as de preenchimento devidamente recortadas pela máscara. O resultado pode ser observado na Figura 4.14.

Figura 4.13: Máscara do fluido.

Como explicado no Capítulo 5, devido a problemas relacionados à memória da GPU, somente a versão 2D desta abordagem foi implementada no trabalho.

4.2.6 Desempenho Computacional

Os testes realizados foram executados na mesma máquina utilizada para testar o simulador SPH, e os resultados podem ser observados na Tabela 4.1.

Uma característica interessante que pode ser observada na tabela são os tempos de execução bastante similares mesmo quando a diferença no número de partículas utilizadas é grande. Esse fato se dá devido ao gargalo causado pelos algoritmos de suavização e extração de bordas que são aplicados em todas as abordagens. Como esses algoritmos operam no espaço da imagem, o número de partículas não possui nenhum tipo de influência sobre o tempo de execução destes, acarretando em desempenhos bastante similares.

Apesar disso, podemos perceber o impacto da busca em grade na versão tridimensional do algoritmo de Linhas de expressão baseadas em partículas. Como no espaço 3D o número de voxels vizinhos que precisam ser visitados pela partícula (em torno de 26) é maior comparado à versão 2D (cerca de 8), observamos que a quantidade de acessos que uma partícula faz à memória da GPU afeta o tempo de execução da aplicação.

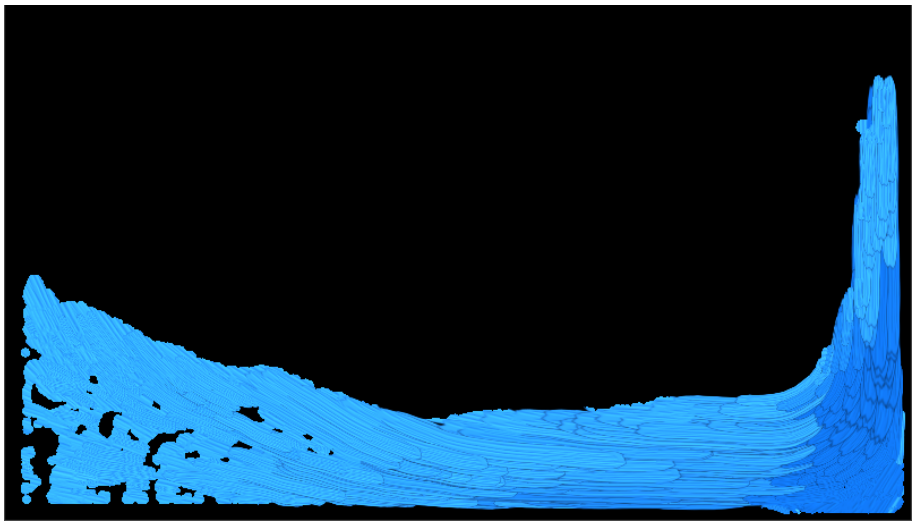

Figura 4.14: Composição final dos *brushes* renderizados nos passos anteriores e recortados pela máscara do fluido.

Tabela 4.1: Desempenho computacional, em FPS, dos algoritmos de renderização não-fotorrealista de SPH.

Algoritmo	Dimensão	N^o de partículas	FPS
Linhas de expressão (quantização)	3D	65536	28
Linhas de expressão (partículas)	2D	4096	29
Linhas de expressão (partículas)	3D	65536	13
Espuma	3D	65536	28
Brushes	2D	4096	28

Capítulo 5

Conclusões

Nesta dissertação, foi implementado um sistema de simulação baseado em partículas com o objetivo de se desenvolver e aplicar abordagens de visualização não-fotorrealistas no mesmo. Neste capítulo, discutiremos os objetivos alcançados durante o desenvolvimento do trabalho, juntamente com as dificuldades encontradas, melhorias que podem ser realizadas e possíveis trabalhos futuros.

5.1 Análise do Trabalho

Acreditamos que o objetivo principal a que o trabalho se propôs foi alcançado com o desenvolvimento de uma quantidade considerável de abordagens que exploram, tanto de forma superficial quanto profunda, as características fornecidas pelo método de simulação Lagrangiano de fluidos. Embora ainda sejam necessárias futuras pesquisas envolvendo a área de renderização não-fotorrealista de SPH para que essas abordagens possam ser aplicadas de forma consistente na indústria ou na área acadêmica, observamos a viabilidade em se aplicar técnicas de NPR na visualização de simulações SPH e as vantagens que animadores, designers e profissionais da área podem usufruir deste sistema NPR de renderização.

5.2 Dificuldades e Trabalhos Futuros

Durante este trabalho, conseguimos identificar os principais desafios encontrados na visualização não-fotorrealista de sistemas de partículas. Entre alguns destes,

podemos citar a grande limitação de dados que podem ser repassados pelo *pipeline gráfico*, sendo esta a principal dificuldade encontrada no desenvolvimento da versão 3D do algoritmo de renderização com *brushes*. Um melhor gerenciamento da memória e uma diminuição dos dados gerados no *geometry shader* são algumas das possíveis soluções para se contornar este problema. Além disso, alguns dos algoritmos de processamento de imagens utilizados no trabalho, como a detecção de silhuetas e a suavização da imagem preservando as bordas, ainda apresentam-se como um importante limitante no desempenho computacional das abordagens desenvolvidas. Contudo, embora a utilização de algoritmos de processamento de imagens mais eficientes seja uma forma plausível de melhorar o desempenho computacional, pensamos que uma melhor solução encontre-se no desenvolvimento de abordagens NPR que consigam contornar o uso destes algoritmos computacionalmente custosos.

Muitos dos desafios encontrados também envolvem a complexidade na programação de soluções paralelas, otimização do acesso à memória compartilhada, falta de informações mais claras no processo de *debbuging* do código executado em GPU, etc. Mas apesar das dificuldades encontradas e das limitações presentes no trabalho, reconhecemos que esta dissertação pode servir como uma base e um direcionamento para que futuras pesquisas possam se aprofundar na área de renderização não-fotorrealista de simulação de fluidos baseada em partículas.

Referências Bibliográficas

[1] MÜLLER, M., CHARYPAR, D., GROSS, M., "Particle-based Fluid Simulation for Interactive Applications". In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, *SCA '03*, pp. 154–159, Eurographics Association: Aire-la-Ville, Switzerland, Switzerland, 2003.

[2] KYPRIANIDIS, J. E., DÖLLNER, J., "Image Abstraction by Structure Adaptive Filtering". In: *Theory and Practice of Computer Graphics*, The Eurographics Association, 2008.

[3] IHMSEN, M., AKINCI, N., AKINCI, G., et al., "Unified Spray, Foam and Air Bubbles for Particle-based Fluids", *Vis. Comput.*, v. 28, n. 6-8, pp. 669–677, June 2012.

[4] VAN DER LAAN, W. J., GREEN, S., SAINZ, M., "Screen Space Fluid Rendering with Curvature Flow". In: *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, *I3D '09*, pp. 91–98, ACM: New York, NY, USA, 2009.

[5] GOSWAMI, P., SCHLEGEL, P., SOLENTHALER, B., et al., "Interactive SPH Simulation and Rendering on the GPU". In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, *SCA '10*, pp. 55–64, Eurographics Association: Aire-la-Ville, Switzerland, Switzerland, 2010.

[6] YU, J., TURK, G., "Reconstructing Surfaces of Particle-based Fluids Using Anisotropic Kernels". In: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, *SCA*

'10, pp. 217–225, Eurographics Association: Aire-la-Ville, Switzerland, Switzerland, 2010.

[7] FOX, R. W., MCDONALD, A. T., PRITCHARD, P. J., *Introduction to Fluid Mechanics*. 6th ed. Wiley, 2003.

[8] PERSEEDOSS, R., *Lagrangian Liquid Simulation using SPH*, Master's Thesis, Bournemouth University, 2011.

[9] PAIVA, A., PETRONETTO, F., LEWINER, T., et al., *Simulação de Fluidos sem Malha: Uma Introdução ao Método SPH*. IMPA: Rio de Janeiro, 2009, 27° Colóquio Brasileiro de Matemática.

[10] KELAGER, M., *Lagrangian Fluid Dynamics Using Smoothed Particle Hydrodynamics*, Master's Thesis, Univ. Copenhagen, 2006.

[11] CRANE, K., LLAMAS, I., TARIQ, S., "Real-Time Simulation and Rendering of 3D Fluids", In: NGUYEN, H. (ed), *GPU Gems 3*, pp. 633–675, Addison-Wesley, 2008.

[12] GINGOLD, R., MONAGHAN, J., "Smoothed Particle Hydrodynamics - Theory and Application to Non-Spherical Stars", *Monthly Notices of Royal Astronomical Society*, 1977.

[13] LUCY, L. B., "A Numerical Approach to the Testing of the Fission Hypothesis", *Astron. J.*, v. 82, pp. 1013–1024, 1977.

[14] LI, W., XIAOMING, W., KAUFMAN, A. E., "Implementing Lattice Boltzmann Computation on Graphics Hardware", *The Visual Computer*, v. 19, pp. 444–456, 2003.

[15] TAN, J., YANG, X., "Physically-Based Fluid Animation: A Survey", *Science in China Series F: Information Sciences*, v. 52, pp. 723–740, 2009.

[16] CHEN, S., DOOLEN, G. D., "Lattice Boltzmann Method for Fluid Flows", *Annual Review of Fluid Mechanics*, v. 30, n. 1, pp. 329–364, 1998.

[17] DESBRUN, M., CANI, M.-P., "Smoothed Particles: A new paradigm for animating highly deformable bodies", In: BOULIC, R.,

HÉGRON, G. (eds), *Computer Animation and Simulation 96*, pp. 61–76, *Eurographics*, Springer Vienna, 1996, Published under the name Marie-Paule Gascuel.

[18] LORENSEN, W. E., CLINE, H. E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", *SIGGRAPH Comput. Graph.*, v. 21, n. 4, pp. 163–169, Aug. 1987.

[19] DU, S., KANAI, T., "GPU-based Adaptive Surface Reconstruction for Real-time SPH Fluids", *WSCG 2014: Full Papers Proceedings: 22nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS Association*, pp. 141–150, 2014.

[20] NGUYEN, H., *Gpu Gems 3*. 1st ed. Addison-Wesley Professional, 2007.

[21] WITKIN, A. P., HECKBERT, P. S., "Using Particles to Sample and Control Implicit Surfaces". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, *SIGGRAPH '94*, pp. 269–277, ACM: New York, NY, USA, 1994.

[22] KOOTEN, K., BERGEN, G., TELEA, A., "Point-Based Visualization of Metaballs on a GPU", In: *Gpu Gems 3*, Addison-Wesley Professional, 2007, Relation: http://www.rug.nl/informatica/organisatie/overorganisatie/iwi Rights: University of Groningen. Research Institute for Mathematics and Computing Science (IWI).

[23] GREEN, S., "Screen Space Fluid Rendering for Games". In: *Proceedings for the Game Developers Conference*, 2010.

[24] YOU, M., PARK, J., CHOI, B., et al., "Cartoon Animation Style Rendering of Water", In: BEBIS, G., BOYLE, R., PARVIN, B., et al. (eds), *Advances in Visual Computing*, v. 5875, pp. 67–78, *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2009.

[25] EDEN, A. M., BARGTEIL, A. W., GOKTEKIN, T. G., et al., "A Method for Cartoon-Style Rendering of Liquid Animations". In: *Proceedings of Graphics Interface 2007*, pp. 51–55, May 2007.

[26] YU, J., JIANG, X., CHEN, H., et al., "Real-time Cartoon Water Animation", *Comput. Animat. Virtual Worlds*, v. 18, n. 4-5, pp. 405–414, Sept. 2007.

[27] NETO, L. D. S. R., GUIMARÃES, F. D., APOLINÁRIO, A. L., et al., "Real-Time Screen Space Rendering of Cartoon Water". In: *Brazilian Symposium on Computer Games and Digital Entertainment, SBGAMES 2013, São Paulo, Brazil, October 16-18*, 2013.

[28] LAKE, A., MARSHALL, C., HARRIS, M., et al., "Stylized Rendering Techniques for Scalable Real-time 3D Animation". In: *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering*, *NPAR '00*, pp. 13–20, ACM: New York, NY, USA, 2000.

[29] NETO, L. D. S. R., APOLINÁRIO, A. L., "Cartoon Water Rendering with Foam and Surface Smoothing". In: *Brazilian Symposium on Computer Games and Digital Entertainment, SBGAMES 2014, Porto Alegre, Brazil, November 12-14*, pp. 230–239, 2014.

[30] HARRIS, M., SENGUPTA, S., OWENS, J. D., "Parallel Prefix Sum (Scan) with CUDA", In: *Gpu Gems 3*, Addison-Wesley Professional, 2007.

[31] PARIS, S., KORNPROBST, P., TUMBLIN, J., et al., "Bilateral Filtering: Theory and Applications", *Foundations and Trends® in Computer Graphics and Vision*, v. 4, n. 1, pp. 1–73, 2008.

[32] WINNEMÖLLER, H., OLSEN, S. C., GOOCH, B., "Real-time Video Abstraction", *ACM Trans. Graph.*, v. 25, n. 3, pp. 1221–1226, July 2006.

[33] KALNINS, R. D., DAVIDSON, P. L., MARKOSIAN, L., et al., "Coherent Stylized Silhouettes". In: *ACM SIGGRAPH 2003 Papers*, *SIGGRAPH '03*, pp. 856–861, ACM: New York, NY, USA, 2003.

[34] HERTZMANN, A., "Painterly Rendering with Curved Brush Strokes of Multiple Sizes". In: *Proceedings of the 25th Annual Conference on*

Computer Graphics and Interactive Techniques, *SIGGRAPH '98*, pp. 453–460, ACM: New York, NY, USA, 1998.