Universidade Federal Fluminense

JOSÉ RICARDO DA SILVA JUNIOR

Simulação computacional em tempo real de fluídos utilizando o método SPH em ambiente heterogêneo CPU/GPU

NITERÓI

2010

JOSÉ RICARDO DA SILVA JUNIOR

Simulação computacional em tempo real de fluídos utilizando o método SPH em ambiente heterogêneo CPU/GPU

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

Orientador:

Prof. Esteban Walter Gonzalez Clua, D.Sc.

Universidade Federal Fluminense

NITERÓI

2010

Simulação computacional em tempo real de fluídos utilizando o método SPH em ambiente heterogêneo CPU/GPU

Jose Ricardo da Silva Junior

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre.

Aprovada por:

Prof. Esteban Walter Gonzalez Clua, D.Sc. / IC-UFF
(Orientador)

Prof. Paulo Aristarco Pagliosa, D.Sc. / UFMS
(Co-Orientador)

Prof. Marcelo Dreux, Ph.D. / PUC-RIO

Prof. Anselmo Antunes Montenegro, D.Sc. / IC-UFF

Niterói, 15 de Junho de 2010.

Dedico este trabalho aos meus pais pelo apoio, carinho e dedicação que sempre tiveram por mim.

Agradecimentos

Ao meu orientador, professor Esteban Clua pelo seu incentivo e apoio nos momentos difíceis, e pela confiança depositada em mim desde o meu ingresso no mestrado.

Ao meu co-orientado, professor Paulo Pagliosa pela ajuda, explicações e incentivo para a realização deste trabalho.

À minha esposa Giselle pelo amor, paciência e compreensão da minha ausência e por ser uma das pessoas que sempre me apoiou e me deu forças para continuar esta jornada.

Agradeço a minha família, pais e irmãos pois sempre me apoiaram e me ajudaram durante os momentos difíceis.

Aos professores Otton Teixeira e Anselmo Antunes pelas suas excelentes aulas, apoio e contribuição no desenvolvimento deste trabalho.

Ao pessoal do MediaLab pela ajuda e amizade que foi desenvolvida.

Ao amigo Erick Passos, pela suas excelentes explicações e ajuda durante o desenvolvimento do meu primeiro paper.

Ao amigo Marcelo Zamith pela sua disponibilidade e atenção durante as vezes em que precisei de sua ajuda no desenvolvimento deste trabalho.

À CAPES pelo apoio financeiro para o desenvolvimento deste trabalho.

A todo o pessoal do Instituto de Computação pelo apoio e amizade desenvolvida ao longo desse tempo.

Resumo

Fenômenos físicos vem sendo estudados desde a década de 70 devido a sua importância para a humanidade. Em particular, o sangue que circula nas artérias, a força do vento que balança a copa das árvores e a força de arrasto que é causada pelo ar ao redor das asas de um avião são exemplos de fenômenos fisicos estudados em dinâmica de fluidos e que desempenham um papel importante na vida das pessoas. Inicialmente esses fenômenos só eram possiveis de serem estudados através de experimentos reais de baixa escala ou estudos teóricos.

Com o advento da computação uma nova ferramenta surgiu para auxiliar neste estudo: a simulação numérica do comportamento físico de fluidos, dando origem a uma nova área do conhecimento chamada *dinâmica de fluidos computacionais*, responsável pelo desenvolvimento de algoritmos numéricos que possibilitam sua resolução utilizando a computação.

Através do crescimento do poder computacional da GPU, a simulação numérica em computador encontrou um ambiente capaz de resolver equações e apresentar os seus resultados em tempo real, além de permitir interações complexas como troca de calor e interação de fluidos com o ambiente. O trabalho desenvolvido nesta dissertação tem por objetivo construir uma arquitetura heterogênea inovadora entre gpu e cpu nunca desenvolvida anteriormente que permita o processamento da simulação de fluidos e corpos rígidos com dupla interação entre as partículas e destas com o ambiente onde estão inseridos. Consequentemente, também é desenvolvida uma nova estrutura de dados computacional capaz de permitir a comunicação entre processadores diferentes e memórias localizadas em espaços independentes de endereçamento em um menor tempo de acesso e processamento.

Abstract

Physical phenomena is being studied since 70's due its importance to humanity. Particularly, the blood that flows in the arteries, the wind's forces that moves trees and the drag force caused by the air over a plane's wings are examples of physical phenomena studied by fluid dynamics that plays an important role in people's life. At first, these phenomena were only possible to be studied through small scaled real experiments or theoretically.

After the computation advent, a new tool grow up to help these research: the numerical simulation of the physical fluid's behaviour, originating a new research field called *computational fluid dynamics*, responsible for development of numerical algorithms to solve the fluid equation computationally.

Through the GPU's processing capability, the numerical simulation using computers found a new ambient capable of solving these equations and presents the results of it in real time beyond allowing for complex interactions like heat exchange and fluid interactions within the ambient where they live. The objective of the work developed in this master dissertation is to develop an innovative heterogeneous architecture using gpu and cpu never seen before to allows for the simulation processing of fluids and rigid bodies and these with the ambient where they are live in. Beyond this, a new data structure is developed to allow the communication between different processors and memory space located at different address in the smallest accessing time and processing.

Palavras-chave

1. *Smoothed Particle Hydrodynamics*
2. Dinâmica de Fluidos Computacional
3. Simulação de Fluidos
4. Balanceamento de Carga GPU/CPU

Glossário

CPU	:	Unidade de Processamento Central
CUDA	:	Computer Unified Device Architecture
DFC	:	Dinâmica de Fluidos Computacional
FBO	:	Frame Buffer Object
Fragment Shader	:	Programa executado em GPU para cada fragmento renderizado
GLSL	:	OpenGL Shader Language
GPGPU	:	Unidade de processamento gráfico para propósito geral
GPU	:	Unidade gráfica de processamento
Shader	:	Programa que é executado em GPU
SIMD	:	Single Instruction Multiple Data
SPH	:	Smoothed Particle Hydrodynamics

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Animações de fenômenos naturais tais como explosões nucleares, interação de fótons com superfície de aspecto rugoso entre outras tem atraído a atenção de muitos pesquisadores, conforme podemos observar na qualidade de imagens e filmes gerados por computador vistos atualmente. Dessa forma podemos corretamente supor que investimentos tanto de cunho intelectual como financeiro vem sendo aplicados na simulação desses fenômenos naturais, principalmente pela indústria de entretenimento.

Especificamente fenômenos naturais tais como o escoamento de água sobre uma superfície, a fumaça emitida por veículos, a ação do vento sobre as asas de um avião são classificados como fluidos com diferentes tipos de comportamentos. Tais fenômenos, por mais simples que possam parecer possuem um alto grau de complexidade.

Antes do advento da computação, o estudo desses fenômenos eram feito através de estudos práticos com a simulação real do fenômeno a ser ser estudado em ambiente controlado ou de estudos teóricos, aprimorando as equações de movimento de cada tipo de escoamento. Porém, em alguns casos, nem sempre o tratamento teórico e/ou experimental são satisfatórios devido a duas razões: (1) o fenômeno a ser observado nem sempre é passível de reprodução em laboratório mesmo em escala reduzida; e (2) proibitivos custo de tempo e de montagem. Como exemplo de (1) podemos citar a previsão do tempo que trata dos movimentos do ar atmosférico com influência direta no tempo e o movimento do sangue nas artérias. Como exemplo de (2) temos o estudo da aerodinâmica em túneis de vento de alta velocidade que são caros de ser operado, geralmente fornecendo amostra pequenas das quantidades físicas tais como pressão e velocidade.

Com o advento da computação, surgiu uma nova área do conhecimento chamada dinâmica de fluidos computacionais (dfc), responsável pelo estudo de métodos numéricos

que permitem a simulação do comportamento de cada tipo de fluído por computador. A simulação numérica computacional de forma alguma surgiu para colocar de lado os experimentos reais e estudos teórico mas sim complementá-los permitindo que seus resultados possam ser utilizados para a escolha de melhores experimentos para o primeiro caso e a validade de novos modelos utilizando resultados da simulação numérica e experimentos para o segundo caso.

A dinâmica de fluidos computacionais estuda métodos numéricos para a resolução das equações de Navier-Stokes, as quais representam o modelo matemático do comportamento de um fluido. Como essas equações não possuem uma solução analítica, utilizam-se métodos numéricos tais como Diferenças Finitas, Volumes Finitos ou qualquer outro método de discretização dessas equações para sua resolução.

A simulação por computador de escoamento de fluidos deve levar em consideração o resultado a ser atingido. Simulações onde são prezados resultados precisos exigem maior esforço computacional, geralmente sendo executadas *offline*. Porém, em simulações onde se deseja resultado em tempo real, deve-se efetuar algumas simplificações nas equações de Navier-Stokes, o que pode tornar seus resultados inaceitável do ponto de vista científico porém visualmente convincente.

Um outro aspecto importante durante a simulação de fluidos computacional é a interação destes com os objetos ao seu redor. Em fenômenos naturais de escoamento de fluidos, é possível observar que estes interagem com os corpos rígidos aos quais tenham contato, gerando forças nesses. Por sua vez, estes mesmo corpos rígidos também exercem forças nas partículas de fluido, gerando o que chamamos *acoplamento de dupla interação*. Devido a sua complexidade, muitas vezes esse tipo de interação não é considerado durante a simulação de escoamento de fluido em tempo real.

1.1 Visão geral do problema

Muitos trabalhos realizam a simulação de fluidos utilizando cpu e mais recentemente utilizando gpu de forma isolada. O mesmo pode-se observar quando se trata de simulação independente de corpos rígidos ou em conjunto com fluidos. Isso ocorre devido a degradação de desempenho que pode ocorrer durante a utilização da cpu em conjunto com a gpu, caso não seja executada de forma apropriada. Muitas vezes essa degradação ocorre devido as constantes trocas de informações entre espaços de memórias de gpu e cpu bem como o grau de paralelismo suportado por cada processador. Além disso, também

podem ocorrer em casos de tarefas mal distribuídas durante entre cpu e gpu durante o processamento.

Durante a simulação de fluidos e corpos rígidos, poder-se-ia adotar um modelo de distribuição de tarefas conforme proposto em [40], obtendo uma distribuição de carga entre gpu e cpu de forma automática. Porém, um fator primordial aqui é a dependência que existe durante a simulação de fluidos e corpos rígidos com dupla interação. Nesse caso, fluidos interagem com corpos rígidos e vice-versa, através de forças repulsivas. Dessa forma, podemos notar que existe uma dependência de informações entre fluidos e corpos rígidos e consequentemente uma dependência de tarefas que devem ser resolvidas a fim de evitar problemas de inconsistências durante o acesso/armazenamento dessas informações.

Além disso, simulação de corpos rígidos e fluídos podem existir em ambiente separados, cada um com sua estrutura de dados que mais lhe seja conveniente. Com a dupla interação entre corpos rígidos e fluidos, é necessário que essas diferentes estruturas de dados interajam entre sim, podendo as mesmas serem completamente diferentes e ainda estar localizadas em diferentes espaços de memória.

Dessa forma, a simulação de fluidos e corpos rígidos em ambiente heterogêneo de cpu e gpu com dupla interação deve considerar a troca de informações entre espaços de memórias diferentes, a distribuição de tarefas entre gpu e cpu e a dependência dessas tarefas durante sua execução. Com isso, pode-se utilizar o poder computacional da gpu paralelamente a cpu, maximizando a carga de processamento de cada um destes processadores.

1.2 Contribuições

A principal contribuição deste trabalho é o desenvolvimento de uma arquitetura heterogênea original entre cpu e gpu para a simulação de fluidos e corpos rígidos com dupla interação, explorando os vários núcleos disponíveis na cpu. Para isso, deve-se desenvolver uma arquitetura que permita a ocupação de ambos os processadores e evite uma interdependência entre os mesmos durante a simulação. Após consulta na literatura, não é de conhecimento do autor o tratamento específico da simulação desenvolvida em ambiente heterogêneo de cpu e gpu.

Outra contribuição relevante deste trabalho consiste na elaboração de uma estrutura de dados capaz de lidar com interações entre corpos rígidos e partículas de fluido. Durante a simulação, várias estruturas de dados em gpu são utilizadas, exigindo comunicação entre as mesmas. Com isso, essa estrutura de dados deve ser capaz de possibilitar a comunicação

com outras estruturas no menor tempo possível, além de evitar possíveis problemas de sincronização que possam ocorrer durante seu acesso.

Após comparações com outros trabalhos disponíveis na literatura, resultados favoráveis destas contribuições foram observados.

1.3 Organização da dissertação

Esta dissertação está composta por sete capítulos. O primeiro capítulo contem a introdução e motivação deste trabalho. O segundo capítulo apresenta os fundamentos da dinâmica computacional de fluidos, bem como suas equações governantes e sua discretização numérica e do domínio do problema. Além disso, também apresenta os fundamentos do *Smoothed Particle Hydrodynamics* para simulação numérica de fluidos baseado em partículas. O terceiro capítulo apresenta a metologia de discretização utilizada neste trabalho enquanto o quarto capítulo a estrutura de dados desenvolvida. No quinto capítulo é apresentado a arquitetura do ambiente heterogêneo de cpu e gpu para a simulação de fluidos e corpos rígidos. Finalmente o sexto capítulo apresenta os resultados obtidos enquanto o sétimo capítulo conclui o trabalho e propõe pesquisas futuras.

Capítulo 2

Fundamentos

Neste capítulo são dados os primeiros passos para o entendimento dos elementos envolvidos na simulação de fluidos computacional. Primeiramente é feita uma introdução as simulações numéricas (Seção 2.1). Após essa introdução são apresentadas as formas de discretização do domínio do problema (Seção 2.2) e posteriormente a teoria da dinâmica de fluidos (Seção 2.3). Em seguida é apresentado o método SPH (Seção 2.4) e finalmente os trabalhos relacionados a este (Seção 2.5).

2.1 Simulações numéricas

Com o advento do poder computacional, as simulações numéricas se tornaram uma poderosa ferramenta para investigação científica. Fenômenos físicos que antes eram estudados através de experimentos reais e suposições teóricas encontram nas simulações numéricas computacional uma valiosa ferramenta para validação de modelos matemáticos desenvolvidos, muitas vezes sem a necessidade de recriação do experimento real para observação.

Inicialmente são desenvolvidos modelos matemáticos utilizando um conjunto de equações governantes que representem o comportamento do fenômeno físico em estudo com condições de contorno e/ou condições iniciais adequadas. Essas equações são em sua maior parte formadas por equações diferenciais ordinárias e parciais que não possuem solução analítica.

Para serem resolvidas numericamente em computador, o domínio do problema precisa ser dividido em componentes discretos bem como as equações de governo que são dadas geralmente em sua forma contínua. A partir desta discretização, são utilizados algoritmos numéricos para o cálculo das variáveis de campo do problema nesses componentes

discretos, criando-se vários cenários de simulação para posterior análise dos resultados obtidos.

Aqui é importante observar que a fidelidade de resultados do método numérico está intimamente relacionada com a resolução da discretização do domínio do problema, isto é, quanto maior for a resolução do espaço discreto, mais pontos de interpolação existirão e consequentemente mais cálculos precisarão ser efetuados. Além disso, a resolução das equações governantes do fenômeno físico despedem do método utilizado para a discretização espacial.

2.2 Discretização espacial

Após a discretização das equações governantes de um fenômeno físico, é necessário a discretização do espaço ou domínio da simulação. Nesta seção, serão apresentadas duas maneiras de efetuar esta discretização do domínio da simulação, de acordo com tipo de algoritmo numérico a ser utilizado para a resolução das equações governantes do fenômeno em estudo.

2.2.1 Métodos baseados em malha

Nos métodos baseados em malha, o espaço de domínio da simulação é discretizado em uma grade composta por um conjunto de células onde são aplicadas os métodos numéricos para o cálculo das variáveis do fenômeno físico em estudo.

Estas malhas podem ser de dois tipos [28]: malhas estruturadas e não estruturadas. Nas malhas estruturadas as células vizinhas de cada célula são conhecidas enquanto que nas malhas não estruturadas é necessário o armazenamento explicito da vizinhança de cada célula, utilizando uma lista encadeada por exemplo.

Conforme descrito em [11], as equações governantes de fenômenos físicos baseados em malhas podem ser divididas em dois tipos: abordagem espacial Euleriana e abordagem material Lagrangeana.

No processo de discretização Lagrangeana, a grade é fixa no material, movendo-se com ele, podendo resultar em expansões e contrações das células durante a deformação do material simulado conforme observado na figura 2.1. Com isso, percebe-se os seguintes benefícios [11]:

- Utilização de malha somente no domínio do problema, evitando possíveis processamento desnecessário em outras células que não fazem parte do problema;
- Utilização de malha irregular para o tratamento de geometria irregular ou complexa;
- Obtenção facilitada do histórico de todas as variáveis do material em um determinado ponto, visto que todos os pontos se movem junto com o material.

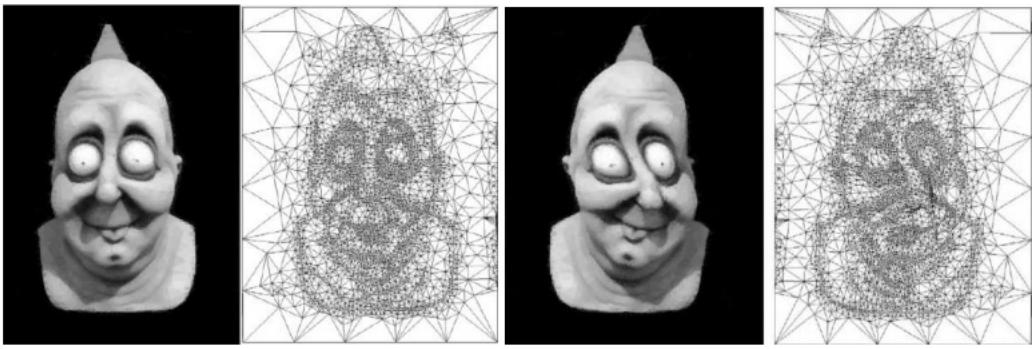

Figura 2.1: Simulação Lagrangeana: a esquerda a malha sem nenhuma deformação; a direita a malha deformada após a simulação utilizando o algoritmo *Imesh* [27].

Porém, podem-se encontrar problemas em trabalhar com métodos Lagrangeanos. Visto que estes se adaptam ao material em que são simulados, grandes deformações podem ocorrer na malha. Quando essas deformações ocorrem, a eficiência do método matemático e até os resultados da simulação podem ser prejudicados. Uma das soluções seria a remalhagem do domínio do problema, utilizando métodos Eulerianos para o cálculo das novas variáveis. Porém, esse processo pode ser computacionalmente caro e tornar-se inviável em aplicações interativas.

Na abordagem Euleriana, ao contrário da Lagrangeana, a grade é fixa no domínio da simulação e o material simulado desloca-se sobre essa grade conforme pode ser visto na figura 2.2. Com isso, deformações muito grandes no material não interferem no resultado numérico da simulação. Apesar desta vantagem, a discretização Euleriana apresenta as seguintes desvantagens:

- Dificuldade do cálculo da superfície livre entre dois fluidos distintos;
- Necessidade de conversão de geometrias irregulares e complexas do domínio do problema em um domínio regular, requerendo muitas vezes um processo de mapeamento numérico computacionalmente complexo;

- Devido a sua posição fixa no espaço, o método Euleriano requer a discretização de todo o domínio da simulação onde o material poderá estar localizado. Com isso, a fim de manter a eficiência para grandes domínios, geralmente diminui-se a precisão do método numérico através da utilização de uma discretização com menor resolução;

- Maior quantidade de memória necessária visto a necessidade de representação de todo o espaço do domínio da simulação.

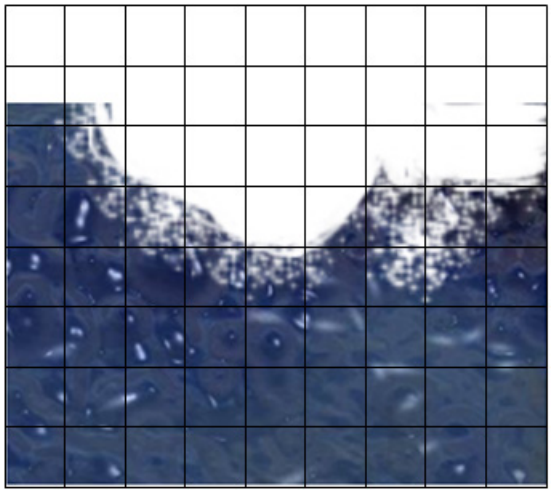

Figura 2.2: Exemplo de simulação Euleriana: o fluido desloca-se sobre a grade fixa no domínio da simulação.

Além dos métodos Lagrangeanos e Eulerianos para resolução de equações governantes, existem os métodos mistos, chamados *ALE (Arbitrary Lagrangian-Eulerian)*[31], os quais utilizam os métodos Lagrangeanos e Eulerianos em conjunto para resolução das equações governantes.

2.2.2 Métodos baseados em partículas

Os métodos baseados em partículas, conhecidos também como *Meshfree Methods* (MM) são métodos de discretização das equações governantes utilizando partículas no domínio da simulação. Possuem as seguintes vantagens em relação aos métodos baseados em malha [11]:

- Possibilitam grandes deformações do domínio da simulação, visto que as informação de conectividade entre os pontos faz parte inerente da computação;

- Podem representar objetos com boa precisão;

- Devido a possibilidade de inserção de novos pontos no domínio da simulação, permitem facilmente refinamento adaptativo para regiões do domínio da simulação que necessitam de resultados mais precisos;
- Fácil representatividade em sua estrutura de dados, podendo ser utilizado um array de pontos, por exemplo.

Dentre as desvantagens, podemos citar as seguintes [29]:

- Necessidade de um cálculo da vizinhança para cada interação devido a falta de informação de conectividade nos métodos baseados em partículas;
- Esforço computacional necessário para o cálculo da vizinhança, podendo ser superior aos métodos baseados em malha devido ao grande número de partículas necessário para simulações com boa precisão;
- Configuração do estado inicial das partículas, que tem forte influencia nos métodos numéricos sem malha.

Como exemplo dos métodos numéricos sem malha podemos citar o *Smoothed Particle Hydrodynamics* que será discutido na seção 2.4 e o método de elementos difusos [34].

Para este trabalho, é adotado o método sph para simulação de fluidos devido ao seu mapeamento mais direto para gpu, atribuindo a cada thread a responsabilidade pelo processamento de uma partícula do sph.

2.3 Dinâmica dos fluidos

As dificuldades da simulação numérica do escoamento de fluidos é devido ao comportamento complexo do fluido, resultado da interação entre vários fenômenos como convecção, difusão superficial, difusão e turbulência [30].

O comportamento de um fluido pode ser classificado da seguinte maneira [35], conforme observado na figura 2.3:

- *Escoamentos Laminares*: são aqueles nos quais camadas finas de fluido parecem deslizar uns sobre os outros

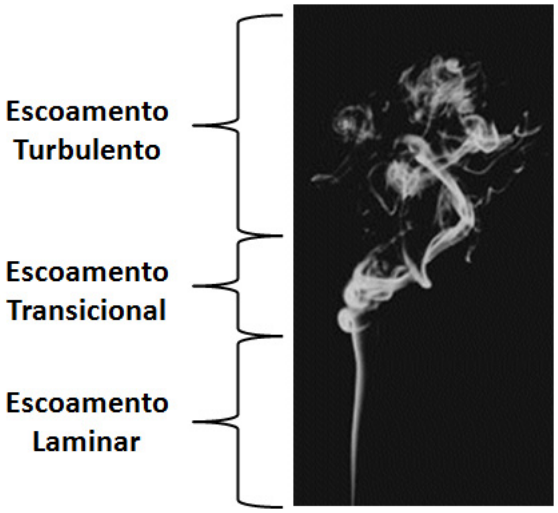

Figura 2.3: Diferentes tipos de escoamento. Inicialmente a fumaça começa com um escoamento laminar e após algum tempo passa para o turbulento através do escoamento transicional. Figura adaptada de http://content.answers.com.

- *Escoamentos Turbulentos*: são aqueles nos quais o movimento das partículas de fluido se dá caoticamente ou desordenadamente.

Além desse comportamentos, tem-se também o comportamento *transicional*, sendo este a fase de transição entre o escoamento laminar e o turbulento. É importante salientar que esses movimentos não são propriedades intrínsecas do fluido mas sim de seu estado.

2.3.1 Equações governantes

Descobertas por Claude Navier em 1822 e George Stokes em 1845, as equações de Navier-Stokes são um conjunto de equações que modelam matematicamente o movimento de um fluido, sendo consideradas o melhor modelo matemático.

Existem diferentes formas para sua representação. As equações

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

e

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho\mathbf{v}) = 0 \tag{2.2}$$

apresentam o modelo para fluidos Newtonianos e não compressíveis. Nessas equações ρ representa a densidade do material, $\mathbf{v}$ representa o campo de velocidade, p o campo de pressão, $\mathbf{g}$ representa as forças externas tal como força da gravidade e μ a constante de viscosidade do fluido.

A equação 2.1 é chamada de *equação da quantidade de movimento* e impõe que a alteração no momento da velocidade deve ser igual a soma das forças que atuam no sistema, sendo resultado da segunda lei de Newton. O termo convectivo $\mathbf{v}.\nabla\mathbf{v}$ representa a variação de uma propriedade do elemento de fluido que se move de uma posição para outra. Nas simulações que utilizam métodos Euleriano, esse termo é necessário visto que a posição da grade não muda com o movimento do fluido enquanto que nos métodos baseados em partículas esse termo pode ser desconsiderado, devido ao próprio material que se move.

A equação 2.2 é chamada de *equação de continuidade* ou *conservação de massa* e impõe que na ausência de sumidouros ou sorvedouros a massa do sistema deve ser constante. Para os métodos baseado em partículas esta equação não é necessária pois é implicitamente satisfeita já que cada partícula carrega uma quantidade de massa constante [4].

O termo ∇p representa a força de pressão por volume, causando o deslocamento de partículas localizadas em campo de alta pressão para campo de baixa pressão. Em um gás ideal, a pressão está relacionada a densidade através da equação

$$p = k\rho \tag{2.3}$$

onde k é uma constante que depende da temperatura. Na ausência de forças, essa relação causa a expansão indefinida de um fluido no espaço, muitas vezes sendo um comportamento indesejado. Para a simulação de fluidos incompressíveis, Desbrun [15] sugere a seguinte adaptação:

$$p = k(\rho - \rho_0) \tag{2.4}$$

Com esta adaptação no cálculo da pressão, a densidade do fluido tende a se aproximar de ρ_0, o que representa a densidade de um fluido em repouso. Como exemplo, [4] configura este valor para 1000 kg/m^3 a fim de simular água.

2.4 Smoothed particle hydrodynamics

Smoothed Particle Hydrodynamics (sph) pertence a classe de métodos Lagrangeanos sem malha. Nesse método numérico o fluido é representado por um conjunto de partículas e o cálculo de variáveis tais como pressão e densidade é realizado através de interpolação destas variáveis nas partículas vizinhas. Foi introduzido na comunidade científica em

1977 através dos trabalhos de Lucy [16] e Gingold e Monaghan [17] para modelagem de problemas astrofísicos.

Apesar de permitir que o fluido seja tratado como um espaço contínuo, o cálculo das quantidades físicas é geralmente efetuado na localização das partículas. A equação utilizada pelo sph para calcular uma quantidade escalar A do fluido na partícula i é dada por [38]

$$A(\mathbf{r}_i) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.5}$$

onde o subscrito j percorre todas as partículas vizinhas de i, $\mathbf{r}$ representa a posição da partícula, ρ a densidade e W a função de suavização de raio h. Nota-se que massa sobre densidade representa o volume ν. Dessa forma, a equação 2.5 pode ser reescrita na forma

$$A(\mathbf{r}_i) = \sum_j \nu_j A_j W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.6}$$

.

Conforme visto na equação 2.1, as variáveis de força de pressão e viscosidade do fluido requer o gradiente e Laplaciano da função, respectivamente. Usando sph, o gradiente da função é obtido através do cálculo do gradiente sobre a função núcleo W

$$\nabla A(\mathbf{r}_i) = \sum_j \nu_j A_j \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.7}$$

O Laplaciano é calculado da mesma forma, através do cálculo do Laplaciano sobre a função núcleo W

$$\nabla^2 A(\mathbf{r}_i) = \sum_j \nu_j A_j \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.8}$$

2.4.1 Equação de forças usando o sph

Calcular as forças internas de pressão e viscosidade que agem em um fluido utilizando o método sph pode ocorrer de várias maneiras. Monaghan [38], por exemplo, utiliza a constante da velocidade do som na equação de viscosidade do material a ser simulado.

Neste trabalho, o cálculo das forças internas de pressão e viscosidade serão baseados nas equações propostas em [4]. Utilizando a formulação sph, para o cálculo da força de pressão substitui-se o termos da pressão da equação 2.1 por 2.7, originando a equação

$$f^{pressão}(\mathbf{r}_i) = -\sum_j \nu_j p_j \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.9}$$

onde p representa a pressão na posição $\mathbf{r}$. Da mesma forma, a viscosidade é obtida através da substituição da força de viscosidade da equação 2.1 por 2.8 originando

$$f^{viscosidade}(\mathbf{r}_i) = \mu \sum_j \nu_j \mathbf{v}_j \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.10}$$

onde μ é a viscosidade do material e $\mathbf{v}$ sua velocidade.

Porém, conforme pode-se observar para o caso de duas partículas, essas forças não são simétricas, violando a terceira lei de Newton. Para contornar este problema, a simetria da força da pressão é obtida através da equação

$$f^{pressão}(\mathbf{r}_i) = -\sum_j \nu_j \frac{p_i + p_j}{2} \nabla W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.11}$$

enquanto que a força de viscosidade é dada por

$$f^{viscosidade}(\mathbf{r}_i) = \sum_j \frac{\mu_i + \mu_j}{2} \nu_j (\mathbf{v}_j - \mathbf{v}_i) \nabla^2 W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.12}$$

2.4.2 Funções de suavização

Todo cálculo em sph requer a utilizacão de uma função de suavização, ou *kernel*, para a interpolação das quantidades físicas do fluido. Essa função atribui pesos as quantidades físicas das partículas vizinhas baseada em sua distância durante o cálculo. Partículas que se encontram fora do raio h, chamado de "suporte compacto"por Monaghan [38], não são consideradas.

Müller et al. afirma que a estabilidade, precisão e velocidade do método sph depende fortemente da escolha da função núcleo utilizada [4].

De acordo com [38], a função de suavização precisa possuir as propriedades

$$\int_\Omega W(\mathbf{r}, h) d\mathbf{r} = 1 \tag{2.13}$$

e

$$\lim_{h \to 0} W(\mathbf{r}, h) = \delta(\mathbf{r}) \tag{2.14}$$

onde δ é a função delta de Dirac

$$\delta(\mathbf{r}) = \begin{cases} \infty & r = 0 \\ 0 & \textit{caso contrário} \end{cases} \tag{2.15}$$

A equação 2.13 define que a função de suavização precisa ser normalizada. Além disso,

a função de suavização também precisa ser positiva

$$W(\mathbf{r}, h) \geq 0 \tag{2.16}$$

a fim de garantir um comportamento de interpolação [36]. Para garantir invariância sobre rotação do sistema de coordenadas, a função de interpolação deve possuir a propriedade

$$W(\mathbf{r}, h) = W(-\mathbf{r}, h) \tag{2.17}$$

Caso as equações 2.13 e 2.17 sejam satisfeitas, a interpolação tem precisão de segunda ordem [38].

O uso de três funções de suavização é aconselhável [4], utilizado neste trabalho. Dessa forma, para o cálculo da densidade a função de suavização

$$W(\mathbf{r}, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3 & 0 \leq r \leq h \\ 0 & \textit{caso contrário} \end{cases} \tag{2.18}$$

é utilizada, onde r representa a magnitude do vetor $\mathbf{r}$. Uma das vantagens desta função de suavização é o não requerimento do cálculo de raiz quadrada, conforme pode ser observado. O gráfico desta função pode ser observado na figura 2.4.

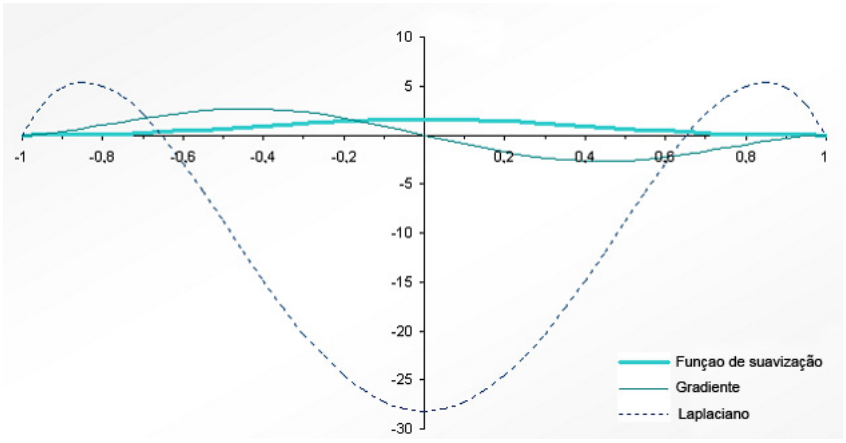

Figura 2.4: Gráfico da função de suavização utilizando h=1. Figura adaptada de [39].

Conforme observado na figura 2.4, o gradiente desta função de suavização tende a *0* a medida que as partículas se aproximam. Com isso, a utilizacão desta função para o cálculo da força de pressão acarretaria no agrupamento de partículas, um comportamento indesejado para simulação de escoamento de fluidos. Para resolver este problema, [4]

utiliza a função de suavização

$$W^{pressão}(\mathbf{r},h) = \frac{15}{\pi h^6}\begin{cases}(h-r)^3 & 0 \leq r \leq h \\ 0 & caso\ contrário\end{cases} \tag{2.19}$$

com o gradiente definido por

$$\nabla W^{pressão}(\mathbf{r},h) = -\frac{45}{\pi h^6}\frac{\mathbf{r}}{r}(h-r)^2 \tag{2.20}$$

e Laplaciano definido por

$$\nabla^2 W^{pressão}(\mathbf{r},h) = -\frac{90}{\pi h^6}\frac{1}{r}(h-r)(h-2r) \tag{2.21}$$

cujo gráfico é apresentado na figura 2.5.

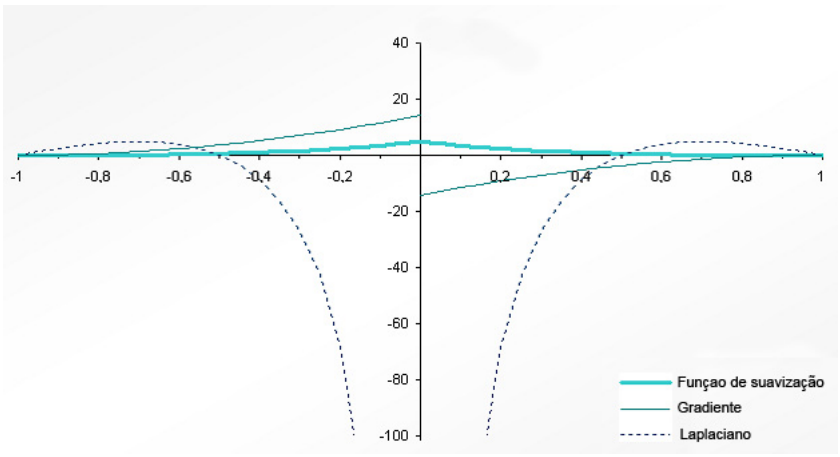

Figura 2.5: Gráfico da função de suavização utilizando h=1. Para uma melhor visibilidade, o gráfico está escalonado diferente. Figura adaptada de [39].

Finalmente, a última função de suavização para o cálculo da viscosidade foi desenvolvida após ser verificado que utilização da equação 2.18 diminui a estabilidade da simulação. Com isso a função de suavização

$$W^{viscosidade}(\mathbf{r},h) = \frac{15}{2\pi h^3}\begin{cases}\frac{-r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1 & 0 \leq r \leq h \\ 0 & caso\ contrário\end{cases} \tag{2.22}$$

é utilizada, cujo gradiente é definido por

$$\nabla W^{viscosidade}(\mathbf{r},h) = \frac{15}{2\pi h^3}\mathbf{r}(-\frac{3r}{2h^3} + \frac{2}{h^2} - \frac{h}{2r^3}) \tag{2.23}$$

e Laplaciano definido por

$$\nabla^2 W^{viscosidade}(\mathbf{r},h) = \frac{45}{\pi h^6}(h-r) \tag{2.24}$$

A figura 2.6 apresenta o comportamento desta função de suavização.

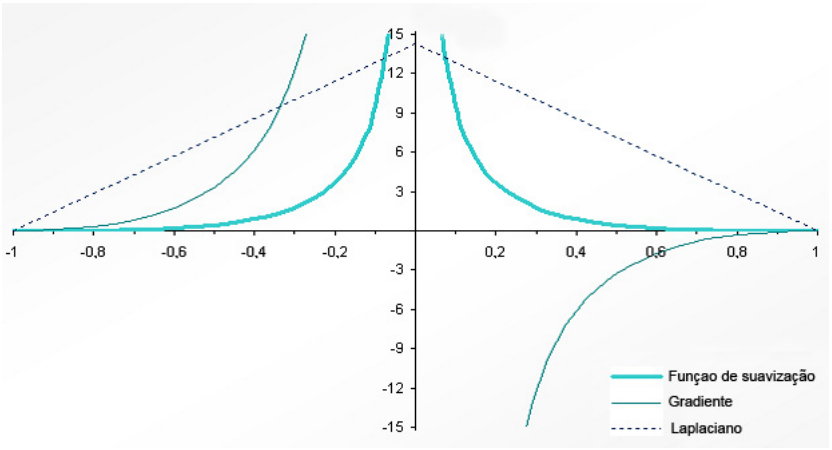

Figura 2.6: Gráfico da função de suavização utilizando h=1. Para uma melhor visibilidade, o gráfico está escalonado diferente. Figura adaptada de [39].

2.4.3 XSPH

Tradicionalmente o sph permite que partículas movimentem-se livremente pelo domínio do problema, sem verificação de colisão, podendo inclusive as mesmas assumirem a mesma posição. Desta forma, inconsistências numéricas podem surgir, visto que partículas na mesma posição podem possuir diferentes atributos tais como pressão e temperatura, ou divisão por zero na avaliação da equação 2.7. Monaghan [38] resolveu este problema desenvolvendo o XSPH onde pondera a velocidade de uma partícula com a dos seus vizinhos através da equação

$$\mathbf{v_i} = \varepsilon \sum_j \frac{m_j}{\rho_j}(\mathbf{v_i} - \mathbf{v_j})W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{2.25}$$

onde ε é uma constante no intervalo $0 \leq \varepsilon \leq 1$. Dessa forma, partículas tendem a mover-se com maior coerência, obtendo uma velocidade média entre as interações.

2.5 Trabalhos relevantes

A simulação do escoamento de fluidos pode ser divida em simulações utilizando cpu e simulações que utilizam gpu. Primeiramente serão abordados os trabalhos que utilizam cpu para simulação.

Um trabalho clássico na simulação de fluidos computacionais foi proposto Kass [8]. Nesse trabalho, os autores introduziram um método rápido e estável para simulação de

água baseado nas equações de águas rasas, as quais são simplificações das equações de Navier-Stokes. Nesta abordagem, os autores utilizam uma função de altura para representar a superfície do fluido e colunas de fluido para modelar o fluxo e transporte de propriedades de elementos do fluido. A discretização do domínio é feita seguindo uma abordagem Euleriana, utilizando-se uma grade cartesiana, onde é aplicado o método de Euler implícito a fim de obter um método numericamente estável. Devido a abordagem proposta, não é possível efetuar a simulação de escoamento de fluidos turbulentos.

Karl Sims [9] utiliza uma arquitetura paralela para simulação de vários fenômenos físicos tais como fogo e queda d'água baseado na utilização de partículas e na aplicação da segunda lei de Newton $f = ma$ para o movimento das partículas. Em seu trabalho, cada partícula é atribuída a um processador virtual, o qual é responsável pela dinâmica da partícula durante o seu tempo de vida. Na simulação da queda d'água o autor simula a espuma gerada na queda variando a tonalidade da cor das partículas. A figura 2.7 ilustra o resultado obtido.

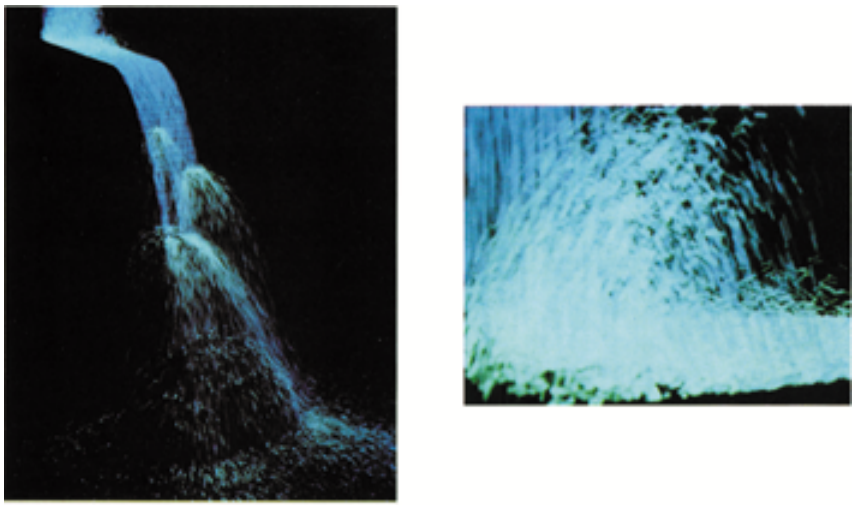

Figura 2.7: Simulação de queda d'água utilizando partículas [9].

O'Brien e Hodgins [13] extenderam a técnica apresentada por Kass [8] para permitir o borrifo e a geração de ondas durante a interação de objetos rígidos com a superfície do fluido. Para alcançar este objetivo, foram desenvolvidos três subsistemas. Um sistema responsável pela geração dos borrifos durante o impacto com corpos rígidos utilizando um sistema de partículas caso a variação de altura de uma coluna de fluido ultrapassasse um valor limite. Para o tratamento das forças externas na superfície, um outro subsistema foi desenvolvido, o qual é responsável pela verificação de colisão entre objetos rígidos e a superfície do fluido, gerando forças para o sistema responsável pela simulação do volume do fluido.

Stam [23] pode ser considerado um dos trabalhos pioneiros na simulação de fenômenos físicos utilizando a equação de Navier-Stokes. Neste trabalho, o autor apresenta um método incondicionalmente estável para simulação de gases utilizando uma abordagem semi-Lagrangeana. O resultado alcançado pode ser visto na figura 2.8. Mais tarde apresenta um trabalho [24] sobre simulação de fluidos utilizando equação da variação da densidade para jogos de maneira incondicionalmente estável.

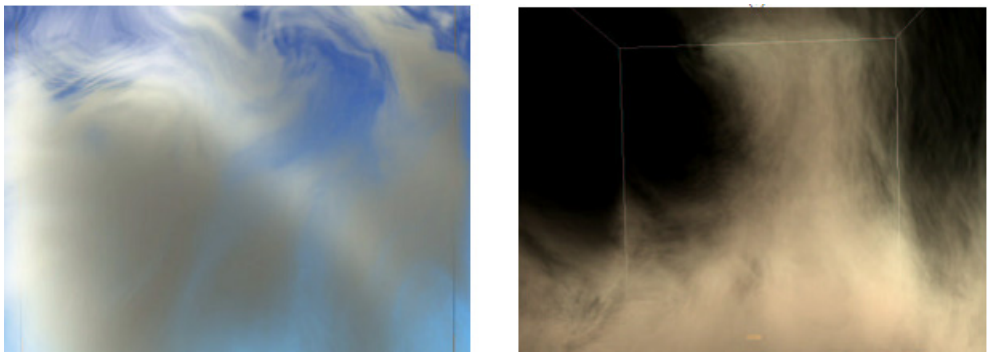

Figura 2.8: Simulação de gases utilizando método semi-Lagrangeano proposto em [23].

Desbrun [15] apresenta a utilização do método sph para simulação de objetos deformáveis com uma extensão em [22] para simulação de lava através do acoplamento da viscosidade à temperatura.

Müller [4] utiliza o método sph para simulação de fluidos em tempo real, derivando as forças de densidade e viscosidade diretamente da equação de Navier-Stokes. O resultado desse trabalho é mostrado na figura 2.9. Em trabalhos futuros, Müller acrescenta a seu trabalho interação de fluidos com corpos rígidos [20] para simulação de cirurgias virtuais. Para isso, Müller utiliza quadratura-Gaussiana para distribuir partículas virtuais na superfície dos polígonos e utiliza o método de Movimento de Partículas semi-implícita para simulação do fluido. Em [21], Müller extende seu modelo para simulação de interação de estruturas fluido-fluido.

Figura 2.9: Enchimento de um copo d'água utilizando sph [4].

Abordando simulações em gpu, Kipfer [48] utiliza sph para simular o escoamento de fluidos em terrenos com deformações utilizando *shaders*. Neste trabalho, interações com outros objetos na cena não são abordados.

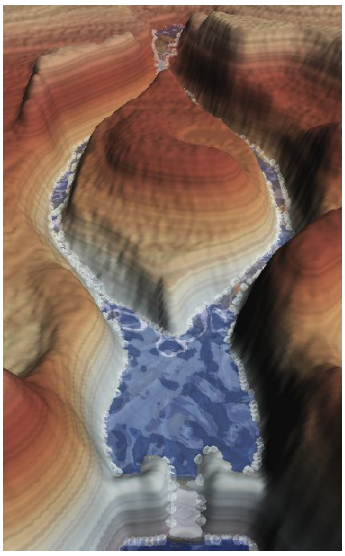

Figura 2.10: Simulação de escoamento de fluido sobre terrenos acidentados [48].

Posteriormente, Zhang [49, 50] também utilizou sph para simulação e renderização de fluidos em gpu através do uso de *shaders*.

Kurose [25] trata da interação entre fluidos e corpos rígidos através da discretização dos últimos em um conjunto de partículas. Após a verificação de colisão, as forças de contato são calculadas resolvendo o método de Problema de Complementaridade Linear e as devidas forças são aplicadas nos corpos rígidos e partículas de fluidos. O resultado alcançado pelo autor pode ser visto na figura 2.11.

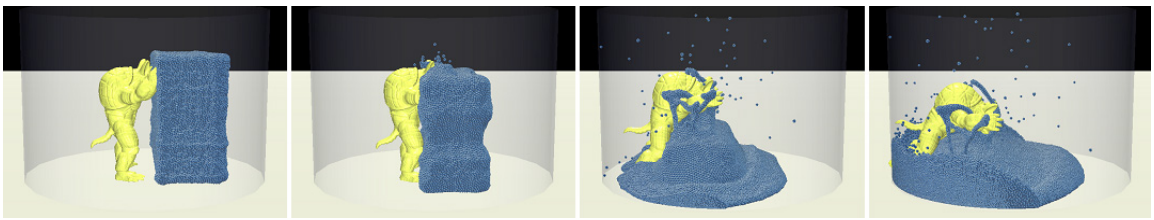

Figura 2.11: Interação de fluidos com corpos rígidos [25].

Capítulo 3

Simulação de corpos rígidos

Um aspecto muito importante durante simulação de fluidos computacionais é o tratamento da interação entre estes com os elementos do ambiente no qual estejam inseridos, sejam elementos estáticos ou dinâmicos. Quando estas interações não são executadas de forma correta, a simulação de fluídos pode apresentar comportamentos não verificados em uma simulação real tal como a penetração do fluido no interior de corpos rígidos. Dessa forma, quanto melhor for a técnica de tratamento de colisão utilizada melhor será a precisão dessa interação quando comparadas a realidade porém com exigência de um maior esforço computacional.

Neste capítulo serão discutidas diversas metodologias que podem ser utilizados para a verificação de colisão entre corpos rígidos (Seção 3.1) e destes com fluidos (Seção 3.2). Posteriormente será discutido formas de discretização de corpos rígidos (Seção 3.3) e o método de *Depth Peeling* utilizado neste trabalho (Seção 3.4). Finalmente é apresentado o algoritmo de discretização desenvolvido utilizando *shaders* (Seção 3.5).

3.1 Detecção de colisão

A verificação de colisão entre corpos rígidos é tratada na literatura utilizando vários algoritmos. Uma lista extensa desses algoritmos pode ser encontrada em [41, 37].

A verificação de colisão entre objetos geralmente é feita através de duas fases: *broad phase* e *narrow phase*. Na *broad phase* é feita uma verificação grosseira de colisão entre objetos, utilizando geralmente uma malha mais simples daquela utilizada durante a renderização. Um algoritmo extensivamente utilizado nesta fase é chamado de *axis aligned bounding box* (AABB) onde é utilizado um retângulo, em duas dimensões, ou um cubo

em 3 dimensões envolvente do objeto. Caso o retângulo ou o cubo se intersectem então ocorreu uma colisão e esses objetos passam para a fase de *narrow phase*. Nesta fase, é feita uma verificação mais precisa de colisão entre os objetos que colidiram na fase anterior, geralmente a nível de triângulos, gerando informações como ponto de contato, normal do ponto de contato, distância de penetração entre outras.

Com isso fica claro a necessidade de representação de uma malha de um corpo rígido em malhas mais simples a fim de acelerar o processo de verificação de colisão entre corpos rígidos com um menor esforço computacional.

3.2 Colisão de fluidos e corpos rígidos

Durante a simulação de fluidos computacional, é necessário o tratamento da interação entre corpos rígidos e o fluido simulado. Isso ocorre devido a necessidade do confinamento de um fluido dentro de um contêiner para sua eficiente simulação [4]. Neste caso, o mínimo necessário é o tratamento da interação entre o fluido e as fronteiras deste contêiner.

Diversos métodos podem ser aplicados no tratamento de colisão entre corpos rígidos e fluidos, classificados em métodos analíticos ou geométricos. Enquanto que nos métodos analíticos os corpos rígidos são representados utilizando uma função matemática para representação da superfície, nos métodos geométricos a representação é feita utilizando geometria, geralmente mais simples que a geometria de renderização a fim de acelerar o processo de verificação de colisão. A avaliação de representações analíticas matemáticas são, em geral, executadas mais rapidamente, devido ao fato destas requererem apenas a avaliação de uma função. Nos métodos geométricos, existe a necessidade de avaliar a colisão geometricamente, muitas vezes sendo esta verificada a nível de triângulos na *narrow phase*.

Durante a simulação de fluidos computacional, este recebe o foco principal na maioria das vezes. Com isso o tratamento da interação entre fluidos e corpos rígidos deve ser o mais simples e eficiente possível a fim de alocar um maior tempo de processamento a simulação do fluido. Neste caso, devido a sua maior eficiência, neste trabalho são utilizadas representações matemáticas de superfícies de objetos para o processamento da interação destes com os fluidos bem como na interação que deve ser avaliada entre os próprios corpos rígidos. Com isso, além de um ambiente para a simulação da dinâmica de fluidos computacional, é necessário também um ambiente para a simulação de corpos rígidos, executado do forma concorrente ou paralela a simulação principal [42].

3.3 Metodologias de discretização

Conforme discutido anteriormente, durante a simulação de fluidos computacional geralmente existe a necessidade do tratamento da interação entre estes e corpos rígidos. Com isso, representações minimalistas e, ao mesmo tempo, representativas desses modelos tanto de natureza analítica como geométrica devem ser buscadas para sua simulação.

Em [1] o autor utiliza método Euleriano para a simulação de fluidos enquanto utiliza o método Lagrangeano para simulação da interação dos mesmos com corpos rígidos. Neste trabalho o autor considera cada vértice do modelo como um ponto de massa ligado a vários outros pontos de massa utilizando uma conexão elástica, considerando também o componente viscoso. Para o tratamento de colisão, são utilizados somente primitivas implícitas.

A utilização da dinâmica de fluidos para simulação de explosões e sua propagação é apresentada em [2] onde o autor utiliza a metologia Euleriana para simulação do fluido. Corpos rígidos possuem duas formas de representação, sendo uma poligonal, utilizada para aplicação das forças provenientes do fluido e outra baseada em voxels, utilizando o volume do objeto para definir a interação dos corpos rígidos nos fluidos. A geração desses voxels é feita em tempo real, dependendo apenas da posição do corpo rígido.

3.4 Depth peeling

Conforme dito anteriormente, o método analítico matemático para representação das superfícies de um objeto é utilizado neste trabalho devido a sua maior eficiência. Assim, a partir de um conjunto de polígonos de um corpo rígido é extraída sua função implícita matemática adequada para sua representação, levando em consideração seu impacto durante a simulação.

Para efetuar esta discretização, é utilizado uma técnica chamada *Depth Peeling* [5] com algumas diferenças a serem discutidas posteriormente. *Depth Peeling* foi originalmente uma técnica utilizada para o tratamento de objetos transparentes. Nesta técnica, a cena é renderizada N vezes, onde N é a resolução de profundidade desejada. Em cada passo de renderização i, a profundidade de cada pixel é armazenada na camada i dependendo do seu valor e o armazenado na camada *i-1*. Após essa etapa, os objetos transparentes são renderizados N vezes a fim de atribuir o valor correto a posição do pixel renderizado. Este algoritmo possui ordem de complexidade $O(N^2)$, sendo N também chamado de

depth complexity.

A partir deste trabalho surgiram vários outros com a mesma ideia, porém, outras aplicabilidades. Em [3], o autor efetua vários passos de renderização a fim de gerar texturas classificadas em ordem de profundidade, ou *layered depth images* (LDI). Dessa forma, é possível obter os pixels mais próximos da câmera, depois os pixels mais próximos da câmera maiores que os primeiros, depois os pixels mais próximos considerando o segundo plano de profundidade e assim sucessivamente, conforme mostrado na figura 3.1. Neste trabalho, o autor utiliza esta técnica para verificação de interseção entre um ponto e uma geometria, transformando o ponto do espaço 3D para o espaço de textura, normalmente compreendido entre o intervalo [0,1].

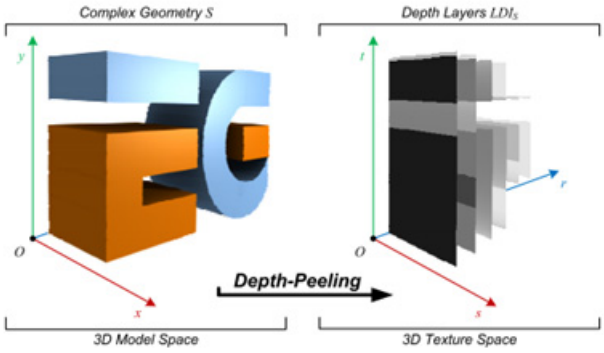

Figura 3.1: Um exemplo de geometria e seus mapas de profundidade. Extraído de [3].

3.5 Algoritmo de discretização

Nesta etapa, foi buscado uma solução que fosse eficiente para ser processada em paralelo pela gpu e, simultaneamente, permitir que objetos mais complexos pudessem ser utilizados na simulação. Para isso, um pré-processamento é efetuado na geometria do objeto a fim de gerar um conjunto de primitivas implícitas. Neste trabalho, a primitiva utilizada para discretização foi um conjunto de esferas de mesmo raio. Dessa forma, os seguintes benefícios foram obtidos:

- Utilização de métodos analíticos em geometrias complexas. Nesse caso, foi utilizado a equação da esfera, sendo esta equação considerada uma das mais rápidas de ser avalizada matematicamente dentro do conjunto das primitivas implícitas disponíveis;
- A arquitetura CUDA é otimizada para um modelo de execução do tipo *uma in-*

strução para múltiplos dados [1], ou seja, uma instrução computacional é executada para um mesmo conjunto de dados. Nesse caso, o teste de colisão sempre será executado utilizando a mesma função matemática com diferentes tipos de dados (raios) obtendo uma maior performance, visto que não é necessário testes de colisão entre primitivas diferentes;

- O sph utiliza partículas para simulação de fluidos. Estas partículas podem ser consideradas esferas primitivas, possibilitando que corpos rígidos possam também ser considerados um conjunto de partículas durante a simulação.

Porém a utilização desta técnica para discretização de corpos rígidos em esferas apresenta as seguintes desvantagens:

- Corpos rígidos discretizados em esferas de raio elevado pode gerar interpenetrações de outros objetos discretizados com um raio menor, tal como as partículas de fluído. Nesse caso, deve-se escolher um valor de raio das partículas que não prejudique a detecção de colisão. Esse problema pode ser visto na figura 3.2, onde um corpo rígido é discretizado utilizando um raio muito superior ao raio da partícula de fluido.

- Apesar de funcionar muito bem para modelos complexos, com grandes curvatura, essa técnica é pouco eficiente para corpos rígidos que possuem grandes áreas de polígonos no eixo de projeção. Isso ocorre devido a perda de informação desses polígonos durante a projeção ortogonal, já que estes polígonos paralelos ao eixo de projeção não são discretizados pois não são renderizados;

- Necessidade de uma etapa de pré-processamento da malha dos corpos rígidos utilizados durante a simulação.

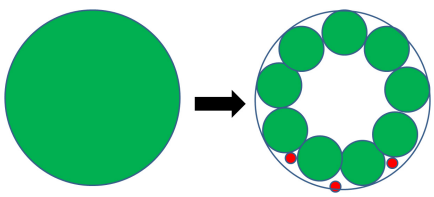

Figura 3.2: Esquerda: corpo rígido a ser discretizado; direita: corpo rígido discretizado utilizando raio elevado, causando interpenetração de partículas de fluido (em vermelho).

[1] Do inglês single instruction multiple data (SIMD)

3.5.1 Processamento utilizando depth peeling

A discretização da geometria em primitivas implícitas esféricas utiliza o *Depth Peeling* como base. Em [7], este método é utilizado para discretização da geometria durante o tratamento de colisão entre corpos rígidos, subdividindo seu volume em um conjunto de células ou *voxels* e obtendo-se a partir daí as células que pertencem a geometria do corpo rígido sendo discretizado utilizando traçado de raios.

No método desenvolvido neste trabalho, o número de renderizações a ser executado já é conhecido a *priori*, antes de sua execução, baseado no raio R parametrizado na simulação. Além disso, aqui não é necessário o armazenamento da profundidade do fragmento do pixel renderizado na textura de saída. Somente uma *flag* é utilizada a fim de indicar que este fragmento pertence a geometria renderizada. Todo o controle dos fragmentos que devem ser considerados na geração dos voxels no layer L são executadas pelo *fragment shader*, conforme pode ser visto na implementação em *GLSL (OpenGL Shader Language)* na listagem 3.1.

```
uniform float radius; // Raio da esfera
uniform vec3  minBB;  // Vertex Minimo do Bounding Box
uniform vec3  maxBB;  // Vertex Maximo do Bounding Box
uniform float layer;  // Depth layer que está sendo renderizado

void main()
{
  int numlayers = (int) ceil((maxBB.z - minBB.z) / (radius * 2.0));

  float unitLayer  = 1.0 / (float) numlayers;
  float unitRadius = 1.0 / ((maxBB.z - minBB.z) / (radius * 2.0));
  float currentLayer = unitLayer * layer;
  float layerEnd = currentLayer + unitRadius;

  if ((gl_FragCoord.z /< currentLayer) || (gl_FragCoord.z /> layerEnd))
    discard;

  gl_FragColor = vec4( 1.0, 1.0, 1.0, 1.0);
}
```

Listagem 3.1: Código para extração do volume da geometria

Neste código da listagem 3.1, a variável *layer* indica o layer a ser renderizado, enquanto que a variável *radius* indica o raio do voxel nos eixos *X, Y e Z*. Estas variáveis

estão definidas entre os valores *[0,1]* onde 0 indica o *near plane* e *1* indica o *far plane* do volume de visualização. As varíaveis *minBB* e *maxBB* são responsáveis por armazenar o *bounding box* do objeto.

Previamente a geração das informações de profundidade do objeto, é definida uma projeção capaz de capturar todas as informações deste objeto. Para isso, dado um objeto O que possui um *bounding box* de O_{bb}, é definida uma projeção ortográfica que compreende os valores da tabela 3.1, transformando o modelo em um volume unitário compreendido entre os valores $[-1, 1]$ obtendo um bom aproveitamento da textura utilizada na renderização sem distorção da perspectiva.

Tabela 3.1: Especificação do volume de visualização

Esquerda	Direita	Superior	Inferior	Plano Near	Plano Far
$O_{bb.min.x}$	$O_{bb.max.x}$	$O_{bb.max.y}$	$O_{bb.min.y}$	$O_{bb.max.z}$	$O_{bb.min.z}$

Um fator importante a ser considerado é o tamanho da memória de textura utilizada. Nesta metodologia, apenas um canal de cor é utilizado, ao contrário do método de *Depth Peeling* original, onde quatro canais de cor são necessários (rgba) visto que o mesmo é utilizado para o tratamento de pixels transparentes. Com essa modificação é obtido um menor consumo de memória daquele utilizado no método original. Na tabela 3.2 é apresentada algumas comparações com possíveis configurações da utilização da memória de vídeo em ambos os métodos, considerando que uma variável do tipo *float* consome *4 bytes* de memória.

Tabela 3.2: Comparação de utilização de memória de vídeo

Resolução	Depth Peeling (RGBA)	Método Proposto (R)	Eficiência
800x600x50	384.000 Kbytes	96.000 Kbytes	75.0%
800x600x60	460.800 Kbytes	115.200 Kbytes	75.0%
1024x768x50	629.145 Kbytes	157.286 Kbytes	75.0%

No processo de geração dos voxels que representam o volume do objeto processado O na textura volumétrica O_{Tex}, as informações presentes nesta textura volumétrica precisam ser convertidas do espaço de textura para o espaço do mundo. Sabendo-se que na projeção ortográfica a distribuição do espaço do mundo para o espaço de textura é linear, esta conversão pode ser efetuada utilizando

areaPixels.x = (int) ceil(($O_{Tex.width}$ / ($O_{bb.max.x}$ - $O_{bb.min.x}$)) * R * 2.0f);
areaPixels.y = (int) ceil(($O_{Tex.height}$ / ($O_{bb.max.y}$ - $O_{bb.min.y}$)) * R * 2.0f);

Com o espaço do modelo subdividido em voxels de tamanho *2R*, o processamento de cada voxel é efetuado através do processamento do subvolume da textura que corresponde a posição do voxel, de tamanho *areaPixels*. Nesse processamento, verifica-se em cada texel do subvolume selecionado se a *flag* de ativação está presente, indicando se este voxel faz parte do volume do objeto *O*. No momento em que a primeira *flag* é encontrada, o processamento deste subvolume pode cessar. Com isso, além da redução no consumo de memória de vídeo para o processamento deste método, existe a possível redução do número de *texels* a serem verificado, no melhor caso, de $R * R * R$, o que não acontece quando é utilizado o método de *depth peeling* original. A figura 3.3 apresenta algumas imagens dessa transformação aplicando este processo.

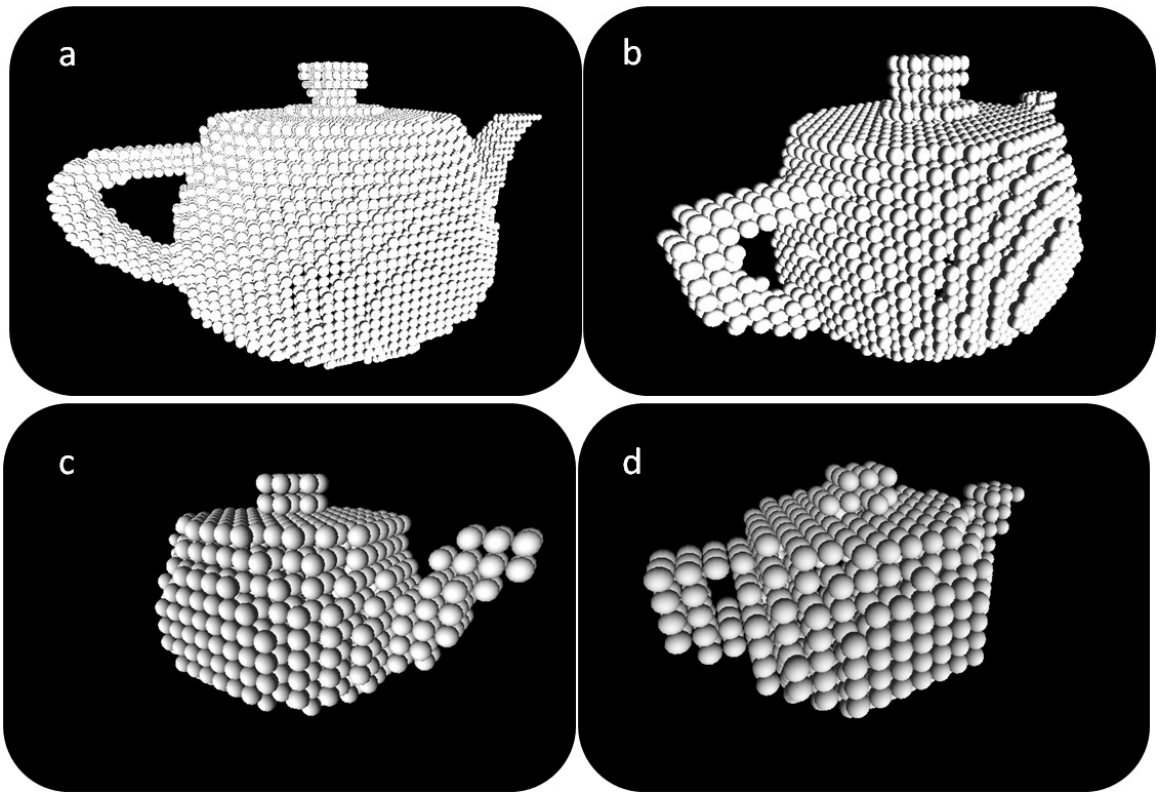

Figura 3.3: Processamento do bule utilizando vários raios como parâmetro: (a) raio 0.3 com 7337 partículas; (b) raio 0.5 com 2674 partículas; (c) raio 0.9 com 600 partículas; (d) raio 1 com 668 partículas.

Capítulo 4

Estrutura de dados

Neste capítulo será apresentada uma proposta inédita de estrutura de dados utilizada para a simulação de fluidos e corpos rígidos estáticos e dinâmicos. Inicialmente é feita uma introdução aos métodos de subdivisão espacial (Seção 4.1) para posterior apresentação da estrutura de dados desenvolvida (Seção 4.1.1) com o objetivo de permitir a interação entre partículas de fluidos e corpos rígidos que encontram-se em espaços diferentes de memória de gpu e cpu. A seguir será apresentado a solução implementada para permitir a interação entre estruturas localizadas em diferentes espaços de memória 4.2. Finalmente serão apresentadas os dados necessários utilizados na simulação para processamento de fluido (Seção 4.3) e os dados utilizados para o processamento de corpos rígidos (Seção 4.4) e suas complexidades.

4.1 Subdivisão espacial

Conforme discutido anteriormente, o sph é um método baseado em partículas sem informação de conectividade entre as partículas vizinhas. Porém, a integração numérica é discretizada utilizando o somatório das variáveis das partículas vizinhas dentro do suporte compacto, podendo estas partículas mudar de posição com frequência durante a simulação. Utilizando força bruta para a localização das partículas vizinhas sem nenhuma estrutura de aceleração é obtida uma complexidade de $O(n^2)$, onde n é o número de partículas total do sistema. Mesmo para um pequeno número de partículas, essa etapa pode exigir um esforço computacional muito grande, degradando consideravelmente o desempenho do sistema. Dessa forma, faz-se necessária uma estrutura de subdivisão espacial com objetivo de diminuir esta complexidade.

Além disso, devido a representação proposta dos corpos rígidos por meio de partículas,

o número de interações entre partículas pode aumentar consideravelmente dependendo da granularidade da discretização dos mesmos, bem como o número de corpos rígidos e partículas de fluído, visto que estamos tratando dupla interação entre fluidos e corpos rígidos.

Um outro fator importante a ser considerado é o tamanho das partículas utilizada na simulação dos fluidos e aquelas utilizadas na discretização dos corpos rígidos, sendo as últimas geralmente maiores que as utilizadas na representação do fluidos. Assim, deve-se escolher uma estrutura que melhor acomode essas diferenças apresentando boa performance durante as buscas utilizando baixo consumo de memória.

Na literatura, uma vasta gama de técnicas pode ser encontrada para estruturas de subdivisão espacial. Em [43] o autor apresenta a análise de implementação de vários métodos utilizados para subdivisão espacial. Como exemplos, podemos citar as estrutura de*octree*, *kd-tree* e *grade regular*.

Inicialmente, é estudado a viabilidade da utilização de uma estrutura de subdivisão espacial do tipo octree [44]. Esta estrutura é composta de nós que possuem cada um, oito nós filho. Cada nó filho é responsável por armazenar $\frac{1}{8}$ do volume representado pelo nó pai, sendo esta divisão geralmente realizada utilizando os planos definidos pelo sistema de coordenadas do espaço global. Devido a essa divisão linear feita a cada nível da árvore, essa estrutura deve levar em consideração o volume médio correspondente de cada elemento a fim de garantir a geração de uma árvore balanceada e com um baixo nível de profundidade. Porém, caso seja necessário vários níveis de profundidade, é desejável que cada nível seja representativo para a simulação, ou seja, cada nível de profundidade da árvore possua dados que justifiquem seu processamento.

Quando fala-se da simulação de fluidos, fala-se de partículas microscópicas chamadas *elementos de fluidos* [10] que são representadas utilizando um conjunto de esferas no método sph. Com isso, a fim de manter uma relação entre o real e o computacional, o suporte compacto das partículas do sph, h deve ser escolhido cuidadosamente a fim de obter uma simulação fisicamente correta. Esse valor deve ser escolhido de forma que cada partícula tenha em média 5, 21 e 27 partículas vizinhas em uma, duas e três dimensões respectivamente [11]. Além disso, é necessário um numero significante de partículas para que as discretização das equações de Navier-Stokes apresente resultados numéricos satisfatórios [10].

Por outro lado, ao discretizar corpos rígidos em um conjunto de esferas de mesmo raio verifica-se a necessidade de interação de elementos macroscópicos que devem interagir com

elementos microscópicos. Devido a esse fator, o processamento desses elementos em uma mesma estrutura de octree gera um grande número de níveis na estrutura que não serão utilizados, desperdiçando tempo de processamento e possivelmente de memória. Além disso, as partículas utilizadas na simulação e na discretização do corpo rígido possuem tamanhos não variáveis ou possuem uma variação controlada, como ocorre no caso de *level of detail* (lod) de colisão. Nesse caso, a utilização de uma estrutura de dados desse tipo torna-se ineficiente para representação de geometrias que possuem volumes distintos.

Devido a regularidade no tamanho das partículas que são utilizadas tanto na simulação do fluido quanto na discretização do corpo rígido em esferas, uma estrutura regular apresenta maiores benefícios. Dessa forma, uma estrutura de subdivisão baseado em uma tabela de hash [45] é utilizada neste trabalho para representação das partículas utilizadas na simulação de fluídos e corpos rígidos.

4.1.1 Definição da estrutura de dados

Durante a simulação foram utilizadas três grades regulares diferentes e independentes. Uma grade regular foi utilizada para classificar as partículas de fluídos, com tamanho de célula de $2K_h$, onde K_h representa o raio do suporte compacto utilizado para o cálculo de contribuição das partículas vizinhas. A partículas são inseridas nas células com base em sua posição. Com isso, durante o cálculo de partículas vizinhas, faz-se necessário somente a verificação das partículas que encontram-se na célula da partícula processada bem como nas células adjacentes, obtendo-se um total de vinte e sete células processadas em três dimensões, conforme pode-se observar na figura 4.1.

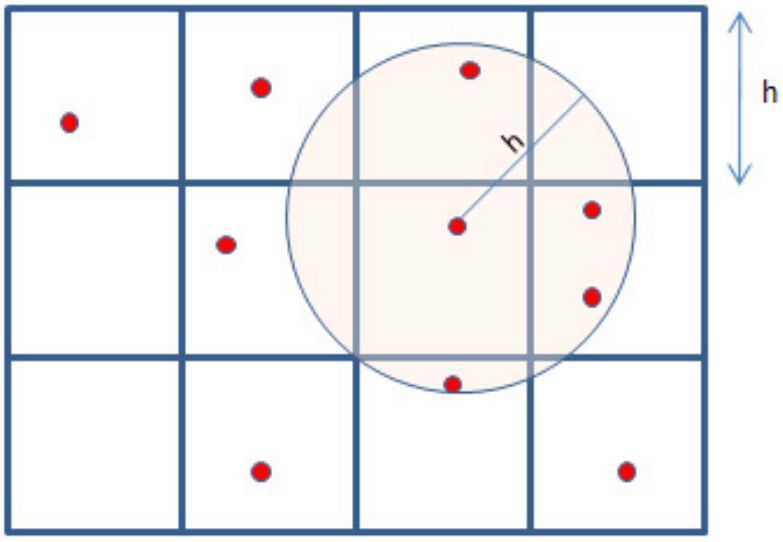

Figura 4.1: Configuração de célula utilizada no sph.

Além desta grade, também é utilizada outra para representar os corpos rígidos estáticos em separado dos corpos rígidos dinâmicos, utilizando um tamanho de célula de $2R_r$, onde R_r, representa o raio da esfera utilizada na discretização do corpo rígido. Essa separação de partículas de corpos rígidos estáticos e corpos rígidos dinâmicos foi efetuada baseando-se na frequência de atualizações, sendo executada somente no momento de inicialização para os corpos rígidos estáticos.

Finalmente, utiliza-se também uma grade regular com as mesmas configurações da grade utilizada na representação de corpos rígidos estáticos, podendo variar somente o número de *buckets* utilizado. Devido a natureza destes corpos rígidos, esta grade é atualizada frequentemente com as novas posições dos corpos rígidos. Na listagem 4.1, é apresentado o algoritmo responsável pelo cálculo do valor de hash. Esse algoritmo é baseado em [14], com ajuste de simetria, visto que na proposta original posições simétricas são mapeadas no mesmo código de hash, podendo gerar um número maior de partículas a serem verificadas.

```
unsigned int HashKey(float3 pos, float3 cell_size, int num_buckets)
{
  int x = (int) ((pos.x + world_limits.x) / cell_size.x);
  int y = (int) ((pos.y + world_limits.y) / cell_size.y);
  int z = (int) ((pos.z + world_limits.z) / cell_size.z);

  return (unsigned int) ((x * p1) xor (y * p2) xor (z * p3)) mod
      num_buckets;
}
```

Listagem 4.1: Método para geração de código de Hash

Na listagem 4.1, os parâmetros *world_limits* representa uma heurística com a menor coordenada do mundo onde a simulação ocorre. *p1*, *p2*, *p3* são grandes números primos, usados aqui com os valores *73856093*, *19349663*, *83492791* [14], respectivamente.

4.2 Mapeamento entre grades

Devido a dupla interação entre corpos rígidos e fluídos, existe a necessidade de interação entre partículas de fluido e partículas utilizadas na representação dos corpos rígidos. Mais ainda, existe a necessidade de interação entre partículas localizadas em grades de células com tamanhos diferentes localizadas em espaço de memória distintos.

A fim de solucionar este problema e permitir essa interação, é efetuado o mapeamento de posição entre grades distintas. Esse mapeamento é efetuado utilizando uma constante de proporcionalidade entre tamanhos de célula, conforme pode ser observado na imagem 4.2.

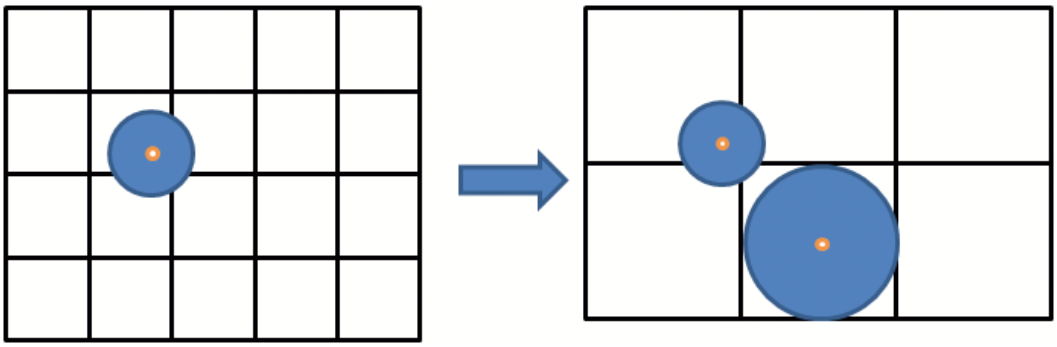

Figura 4.2: Mapeamento de uma esfera entre grades de diferentes tamanhos.

Conforme observado na imagem 4.2, essa constante é especificada através da fórmula $G_{FR} = Ceil(\frac{R_r}{K_h})$ para o mapeamento de partículas de fluido em grades de corpos rígidos e $G_{CR} = Ceil(\frac{K_h}{R_r})$ para o mapeamento de partículas de corpos rígidos em grades de fluidos. Com isso, um passo simples de processamento pode ser utilizado para efetuar a interação entre partículas de fluidos e corpos rígidos que encontra-se em diferentes grades na memória, conforme pode ser visto no trecho de algoritmo 4.2.

```
for (int x = -G_FR; x <= G_FR; x++)
{
  for (int y = -G_FR; y <= G_FR; y++)
  {
    for (int z = -G_FR; z <= G_FR; z++)
    {
      float3 rb_map_pos = float3(rb_cell_size.x * x, rb_cell_size.y * y,
          rb_cell_size.z * z);
      int rb_cell_bucket = HashKey( rb_map_pos, rb_cell_size,
          rb_num_buckets);
      ProcessRBParticles(rb_cell_bucket);
    }
  }
}
```

Listagem 4.2: Interação entre fluidos e corpos rígidos

Com a utilização de várias grades, obtém-se uma melhor performance devido a utilização de uma grade específica para cada caso, permitindo simplificações de inserção e

busca que seriam mais complexas caso fosse utilizado apenas uma grade.

4.3 Informações relativas ao processamento de escoamento de fluidos

Neste trabalho é realizado a simulação de fluido Newtoniano isotérmico quasi-incompressível utilizando o método Lagrangeano baseado em partículas sph. Para isso, as informações necessária bem como seu tipo de armazenamento é apresentado na tabela 4.1.

Tabela 4.1: Campos e seus respectivos tipos utilizados na representação das partículas de fluido.

Campo	Tipo de Dado
Posição	vector4
Força	vector4
Massa	float
Velocidade	vector4
Densidade/Pressão	vector2

Devido ao fato de não existir nenhuma informação de conectividade entre as partículas de fluido, esses campos podem ser processados separadamente em cada partícula. Dessa forma, o processamento da densidade, pressão, aceleração e integração de forças podem ser executados independentemente em cada partícula de fluido.

4.4 Informações relativas ao processamento de corpos rígidos

Conforme discutido anteriormente, corpos rígidos são representados através de partículas com raio constante. Campos necessários ao processamento do corpo rígido são apresentados na tabela 4.2, assim como o tipo de dados que utilizam.

Tabela 4.2: Campos e seus respectivos tipos utilizados na representação de partículas de corpos rígidos.

Campo	Tipo de Dado
Posição	vector4
Massa	float
Velocidade	vector4
Orientação	vector4

Além das informações apresentadas, um corpo rígido possui um conjunto k de partículas. Essas partículas possuem uma posição relativa ao centro de massa do corpo rígido,

conforme apresentado na figura 4.3. Para possibilitar a interação com outras partículas dentro do sistema, suas posições absolutas precisam ser calculadas a cada frame. Partículas de corpos rígidos estáticos inseridos na cena são armazenadas com posições absolutas e inseridas em uma grade diferente das partículas de fluido e de corpos rígidos dinâmicos. Isso deve-se ao fato dessas partículas nunca precisarem ser atualizadas durante a simulação.

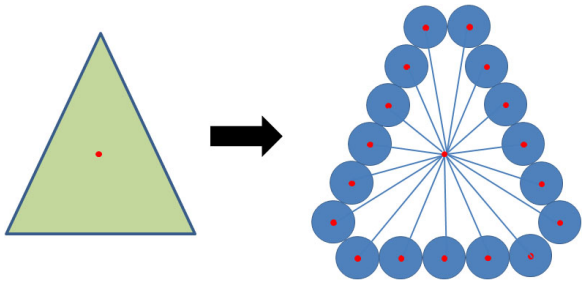

Figura 4.3: Relação entre as partículas e o centro de massa do corpo rígido.

Partículas de corpos rígidos dinâmicos possuem posições que variam com a posição e orientação do centro de massa do corpo rígido ao qual pertence. Dessa forma, a fim de evitar um grande tráfego entre gpu e cpu, as posições das partículas de corpos rígidos dinâmicos são armazenadas na gpu, assim como a posição do centro de massa do corpo rígido e sua orientação. A cada interação, a posição absoluta de cada partícula é atualizada e seu processamento segue normalmente.

Cada partícula p_x pertencente ao corpo rígido Y necessita acesso as informações de orientação RY_{ori} e posição do centro de massa RY_{cm} do respectivo corpo rígido. Pode-se observar através da figura 4.4 que existe um relacionamento $N \rightarrow 1$ entre as partículas e seu respectivo corpo rígido. Além disso, um corpo rígido A é discretizado em um conjunto de P_A, o que não é garantido ser o mesmo número para todos os corpos rígidos na simulação. Como essas partículas encontram-se na gpu, uma solução para esse problema seria a duplicação dessas informações para cada partícula, exigindo um maior consumo de memória. Porém, uma outra solução é adotada neste trabalho. A posição w de cada partícula relativa armazena o índice do corpo rígido ao qual esta partícula pertence. Esse índice é utilizado para localizar as informações de posição de centro de massa e orientação de seu respectivo corpo rígido durante a transformação da partícula para posição absoluta, o que evita duplicação de dados e consequentemente menor consumo de memória.

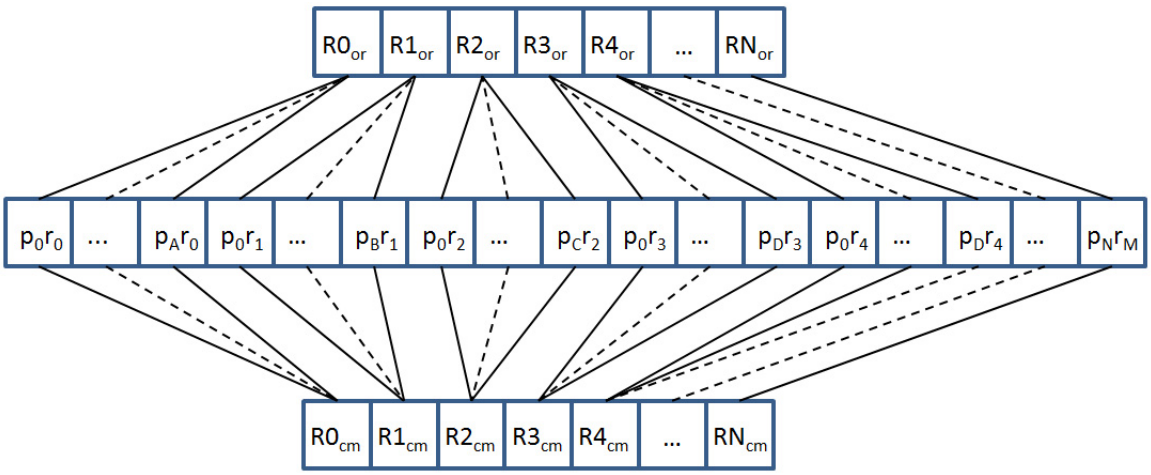

Figura 4.4: Dependência de dados entre partículas e corpos rígidos.

Capítulo 5

Ambiente heterogêneo (cpu-gpu)

Neste capítulo será apresentado um modelo inédito de distribuição tarefas para processamento de fluidos e corpos rígidos com dupla interação utilizando ambiente heterogêneo de gpu em conjunto com vários núcleos de cpu. Para evitar a ociosidade entre os processadores, este modelo é desenvolvido utilizando baixo acoplamento entre tarefas de gpu e cpu. Para isso é necessário a identificação das tarefas e suas dependências (Seção 5.1). Após esta identificação, é apresentado o modelo desenvolvido (Seção 5.2) bem como sua distribuição entre os processadores durante a simulação (Seção 5.3) e o detalhamento destas para simulação de fluidos e corpos rígidos.

5.1 Identificação das tarefas

Durante a simulação de fluidos e corpos rígidos, tarefas bem definidas podem ser observadas, bem como sua ordem de execução, conforme apresentado na figura 5.1. Na utilização em simulações de fluidos e corpos rígidos onde não ocorre interação entre estes, o processamento de tarefas inerentes a simulação do escoamento do fluido poderia ser executado concorrentemente com o processamento de tarefas inerentes a simulação de corpos rígidos. Esse paralelismo poderia ser obtido em gpu utilizando uma arquitetura FERMI [1] e em cpu utilizando um processador com vários núcleos.

Porém, o modelo desenvolvido neste trabalho considera a interação entre fluidos e corpos rígidos. Dessa forma, pode-se identificar algumas dependências durante o processamento das tarefas de fluidos e corpos rígidos, conforme pode ser observado na figura 5.2 através das setas tracejadas em vermelho.

[1] http://www.nvidia.com/object/fermi_architecture.html

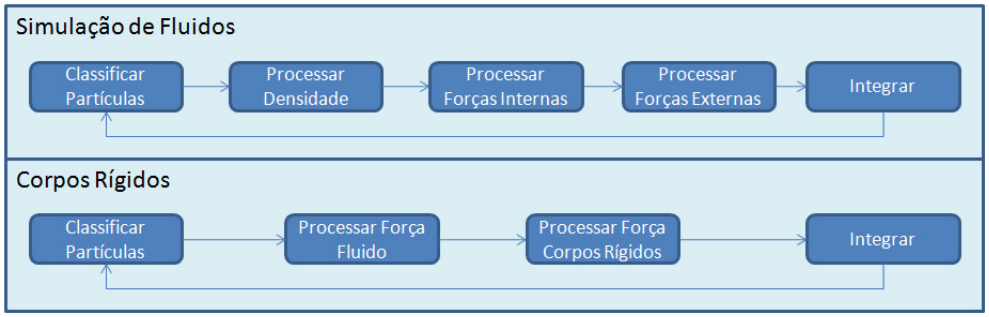

Figura 5.1: Tarefas necessárias para a simulação de fluidos e corpos rígidos.

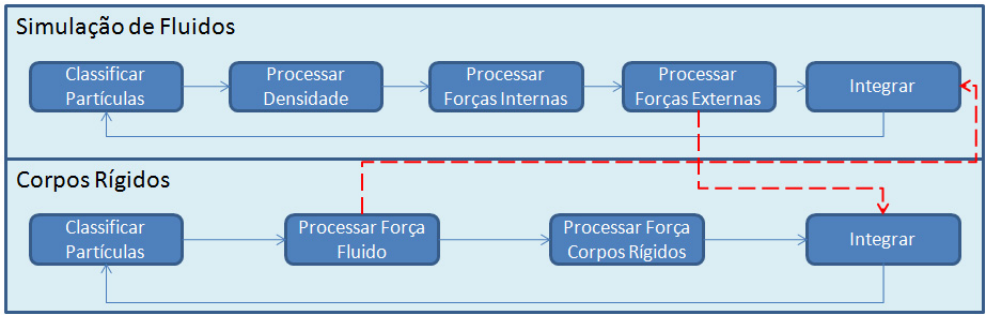

Figura 5.2: Dependência entre as tarefas, representada pela seta tracejada.

Utilizando uma arquitetura heterogênea de gpu e cpu, além da dependência de dados existente durante o processamento destas tarefas, também deve ser resolvida a distribuição das mesmas nestes processadores de forma a minimizar ao máximo qualquer ociosidade que possa ocorrer. Para efetuar este desacoplamento de tarefas, as informações utilizadas do fluído na simulação de corpos rígidos e dos corpos rígidos na simulação de fluidos são obtidas do *frame* anterior, técnica similar ao *double buffer* em computação gráfica. Nesse caso, durante o processamento da simulação de fluidos e corpos rígidos, informações tais como posição, velocidade e orientação são armazenadas em um espaço de memória diferente daquele acessado pela simulação oposta.

5.2 Modelo de tarefas

Conforme dito anteriormente, a gpu possui uma arquitetura *simd*, ou seja, os dados de entrada são processados simultaneamente por várias threads que executam a mesma função sobre esses dados. Com isso, as tarefas atribuidas a gpu devem seguir esse modelo de execução a fim de obter melhor performance.

Por sua vez, o paralelismo da cpu é inferior se comparado ao da gpu, devendo esse fato ser levado em consideração durante a atribuição de tarefas entre estes processadores.

A simulação de fluidos e corpos rígidos em trabalhos apresentados na literatura utilizando gpu [50, 49, 25, 7] efetuam o processamento de todas as tarefas identificadas na figura 5.1 a cada *frame*, independentemente destes estarem interagindo ou não durante seu processamento.

Porém, conforme podemos observar na figura 5.3, pode-se identificar três fatos importantes que permite simplificar ou até excluir a necessidade de processamento de algumas destas tarefas durante o processamento de cada *frame*: (1) existem corpos rígidos que não colidem com corpos rígidos; (2) existem corpos rígidos que não colidem com fluidos e (3) existem corpos rígidos que não colidem com fluidos e outros corpos rígidos. Com esta observação, um modelo de tarefas mais eficiente é apresentado na figura 5.4.

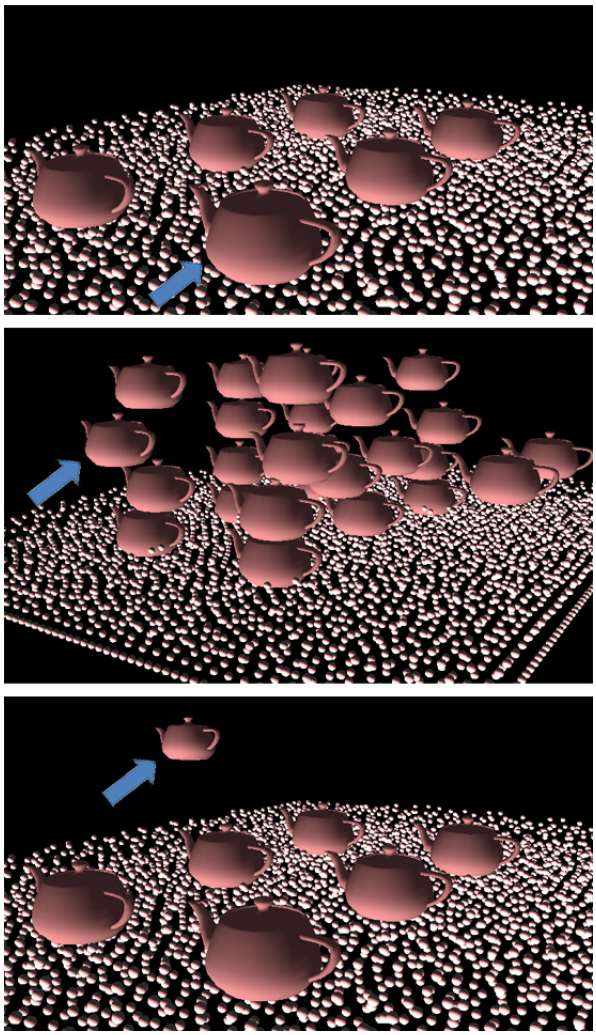

Figura 5.3: Comportamentos durante a simulação. De cima para baixo: corpo rígido sem interação outro corpo rígido; corpo rígido sem interação com fluido e corpo rígido sem interação com fluido e outro corpo rígido.

Conforme observado no modelo apresentado na figura 5.4, em alguns casos pode-se omitir o processamento de determinadas tarefas durante cada *frame* da simulação e as

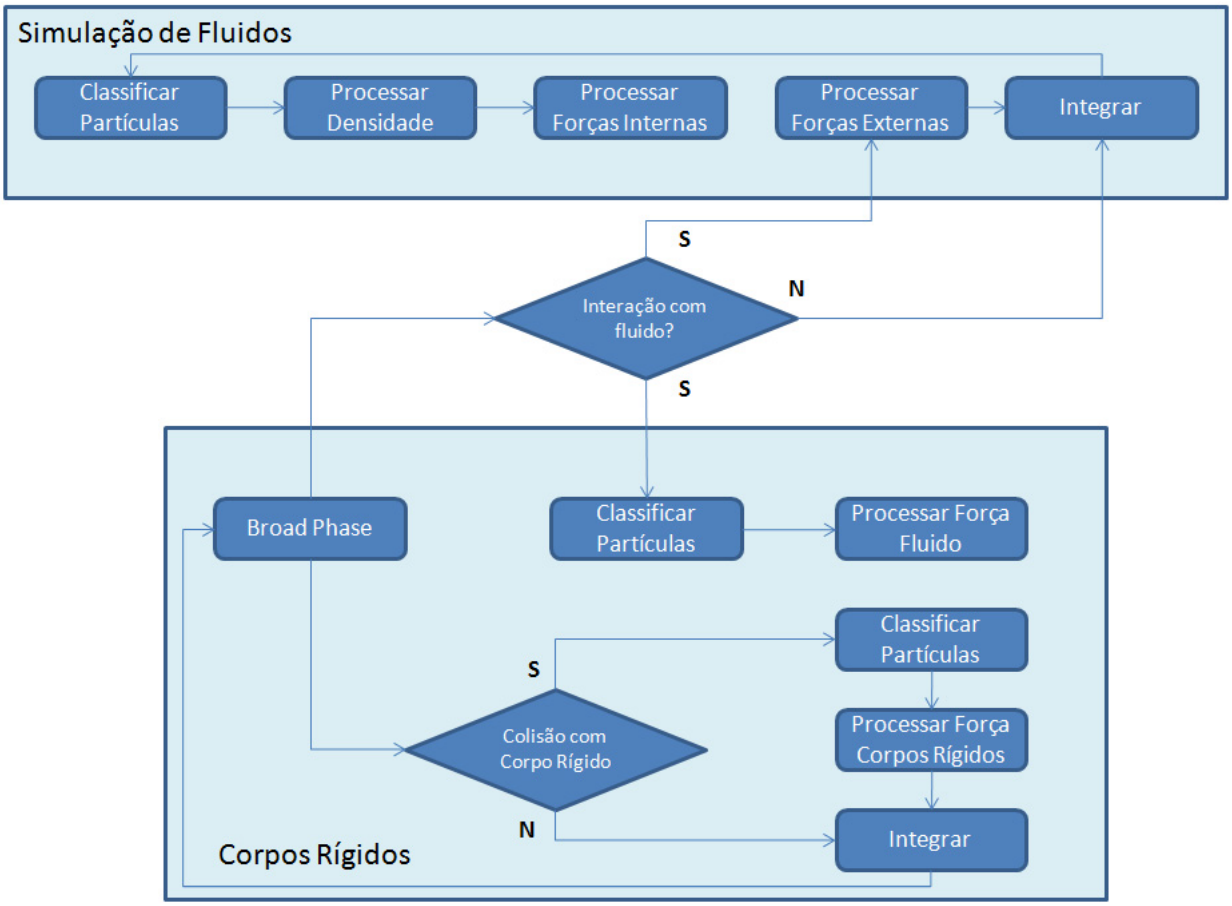

Figura 5.4: Modelo implementado para execução da simulação de corpos rígidos e fluidos com dupla interação.

interações entre corpos rígidos e fluidos ou entre corpos rígidos somente precisam ser processadas para aqueles que potencialmente colidiram.

Dessa forma, baseado no modelo apresentado, a figura 5.5 apresenta a distribuição de tarefas a serem processadas em gpu e cpu, cujo funcionamento será detalhado na próxima seção.

5.3 Distribuição de tarefas

Com o objetivo de manter o processamento contínuo em ambos os processadores, o processamento das tarefas são dividido em quatro blocos, conforme pode ser visto na figura 5.5.

No modelo apresentado, o *bloco 1* inicia a classificação das partículas de fluido enquanto que no *bloco 3*, ao mesmo tempo, a cpu inicia a etapa de *broad phase*. Após esta verificação, em caso de colisão, o *bloco 2* é iniciado classificando partículas que colidiram

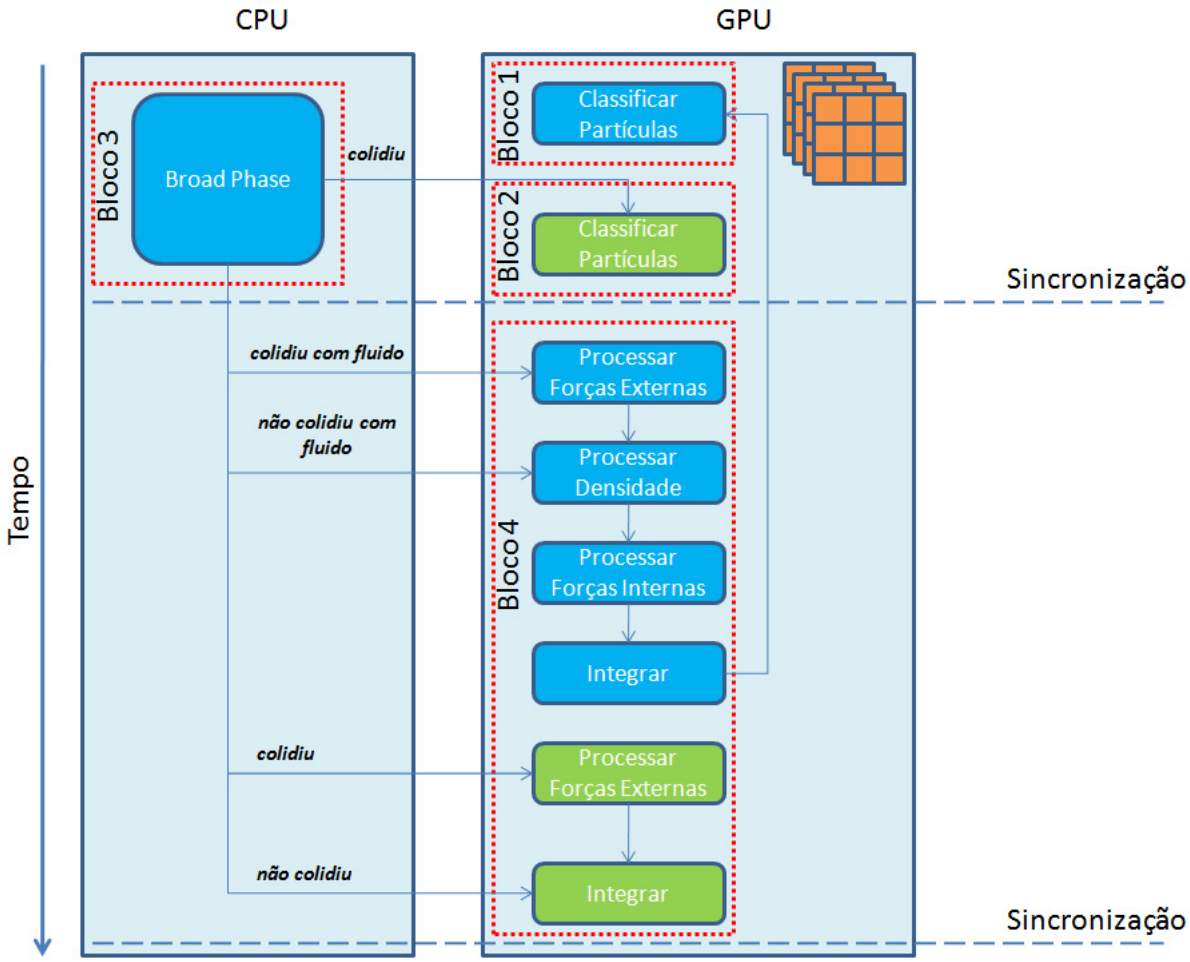

Figura 5.5: Diagrama de distribuição de tarefas entre gpu e cpu. No bloco da gpu em azul as tarefas inerentes ao simulação de fluidos enquanto verde indica as tarefas inerentes a simulação de fluidos.

com o fluido e partículas que colidiram com corpos rígidos.

Após a etapa de *broad phase*, as tarefas do *bloco 4* são executadas de acordo com o resultado do teste de colisão executado. Somente partículas de corpos rígidos que potencialmente colidiram são avaliadas, ao contrário de vários trabalhos que utilizam discretização por partículas[46, 25, 7], os quais enviam todas as partículas presentes na cena para processamento de colisão em gpu.

Nas seções seguintes, o funcionamento destas tarefas serão apresentadas mais detalhadamente.

5.3.1 Classificação de partículas

A etapa de classificação de partículas é responsável pelo agrupamento das partículas em suas respectivas células, utilizando uma função de *hash* baseada em sua posição.

Para a simulação de fluídos, esta tarefa deve ser executada a cada *frame*, visto que métodos Lagrangeanos necessitam de informações das partículas vizinhas para a computação de variáveis tais como pressão e densidade.

Porém, para simulação de corpos rígidos, suas partículas somente são utilizadas para o cálculo de colisão, sendo necessário sua classificação somente nos casos onde ocorram colisão com fluidos ou outros corpos rígidos.

5.3.2 Calculo de densidade

Após a classificação de partículas, deve-se calcular a densidade, utilizando as informações processadas previamente para a localização das partículas vizinhas. Devido a dependência do campo de densidade para os cálculos posteriores, essa etapa deve ser executada em um *kernel* diferente durante o processamento do sph.

5.3.3 Cálculo das forças internas

Nesta etapa do processo, são calculadas as forças internas do fluido. Conforme discutido anteriormente, essas forças internas são devido a pressão e viscosidade, calculadas utilizando a formulação sph.

5.3.4 Broad phase

A simulação de fluidos e corpos rígidos com dupla interação requer acesso a espaços de memória distintos e independentes. Mesmo utilizando uma estrutura de dados eficiente para esta comunicação, esta interação ainda pode-se apresentar custosa para processamento, visto o número de partículas de fluido e corpos rígidos que necessitam de processamento.

Durante a simulação de fluidos e corpos rígidos com dupla interação, os corpos rígidos são discretizados em partículas. Estas partículas são utilizadas somente para efeito de colisão sendo desnecessário o seu processamento para corpos rígidos que não apresentem colisão com fluido e/ou outros corpos rígidos. Assim, nesta etapa verifica-se corpos rígidos

que colidiram com outros corpos rígidos e/ou fluidos utilizando a técnica de *bouding box*. Essa verificação é executada em paralelo pelo número de núcleos disponível na cpu, utilizando a biblioteca *OpenMP* [2], e os *flags* de colisão armazenados em um array, conforme pode ser observado na figura 5.6.

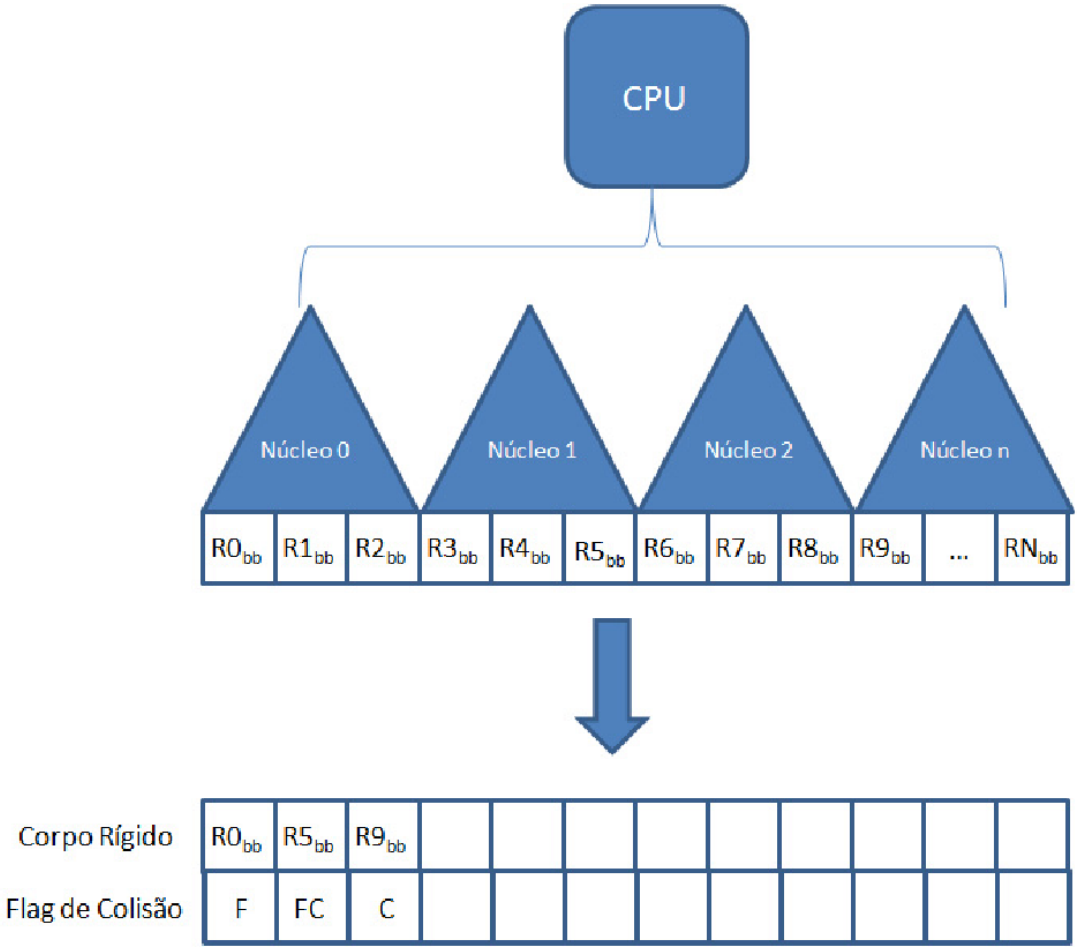

Figura 5.6: Execução da *broad phase* utilizando múltiplos núcleos em cpu. Rk_{bb} representa o *bouding box* do corpo rígido k. F representa colisão com fluido e C com corpo rígido.

Após este processamento, tem-se um array indicando os corpos rígidos que potencialmente colidiram assim como os *flags*, indicando se colidiram com outros corpos rígidos e/ou fluido. Essa identificação é importante pois permite que o processamento do fluido prossiga caso não exista nenhuma interação com corpos rígidos. Neste caso, apenas as forças internas e gravitacional precisam ser processadas. Esta pré-seleção é bastante importante pois verifica-se que em muitas simulações o fluido ocupa uma pequena parte do espaço em comparação com os corpos rígidos. Além disso, podemos ter uma pilha de corpos rígidos sobre um fluido, ocorrendo interação somente dos fluidos que encontram-se na base desta pilha.

[2]Disponível em http://www.openmp.org

Utilizando este array de *flags*, são gerados dois outros arrays, caso necessário: um array contendo as posições absolutas das partículas de corpos rígidos que colidiram com outros corpos rígidos e um outro array com as mesmas informações porém contendo partículas de corpos rígidos que colidiram com fluidos. Também são gerados um array com o número de partículas de cada corpo rígido e offset, responsável pela localização das partículas de cada corpo rígido no array de posição, conforme pode ser observado na figura 5.7.

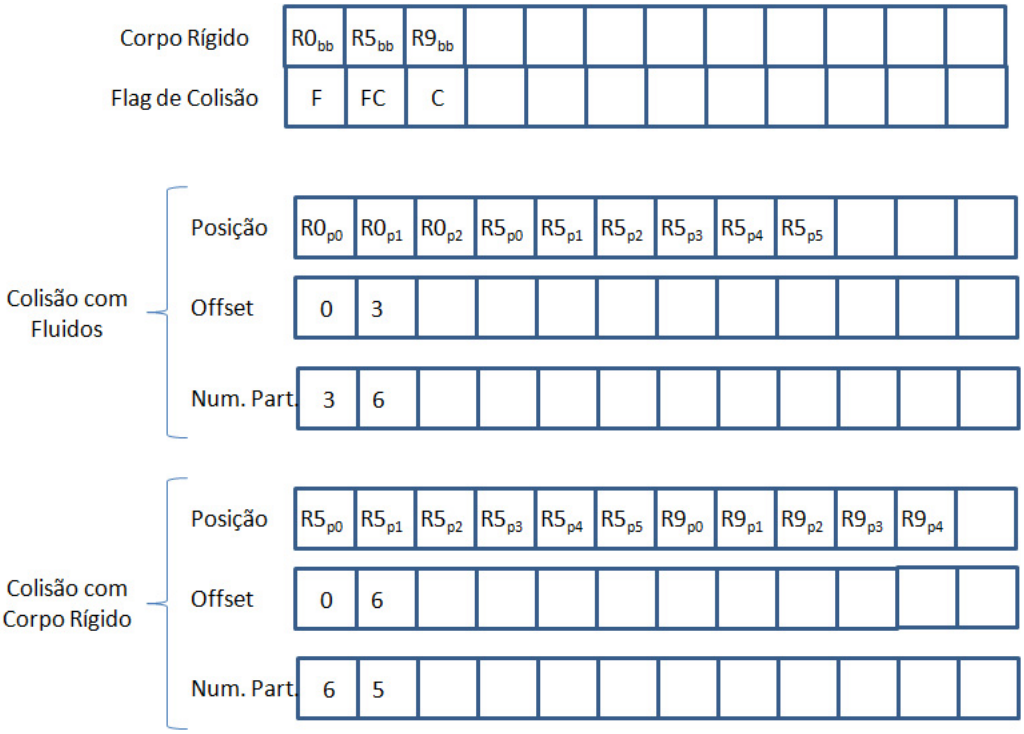

Figura 5.7: Informações geradas após o processamento da etapa de *broad phase*. Partículas que colidiram com corpos rígidos são armazenadas independentemente daqueles que colidiram com fluidos.

Após esta etapa, somente as partículas de corpo rígido que potencialmente possam colidir são processadas, efetuando-se a classificação e dupla interação entre fluidos e corpos rígidos.

5.3.5 Cálculo das forças externas

Durante esta etapa são calculados as forças de fluido nos corpos rígidos e destes nos fluidos somente para as partículas de corpos rígidos que foram selecionadas na etapa anterior. Além disso, a interação de fluidos com corpos rígidos será processada independentemente da interação entre corpos rígidos.

Para o cálculo das forças entre estas partículas, utiliza-se o método de elementos discretos (dem) utilizado para simulação de materiais granulares [12] tal como areia. A

força repulsiva f_{ij}, agindo na partícula i através da interação com uma partícula j é modelada utilizando uma força de elasticidade $f_{i,s}$

$$f_{i,s} = -k(t_{rad} - |r_{ij}|)\frac{r_{ij}}{|r_{ij}|} \tag{5.1}$$

e uma força de amortecimento $f_{i,d}$

$$f_{i,d} = \eta v_{ij} \tag{5.2}$$

onde k é o coeficiente de elasticidade, η é o coeficiente de amortecimento, r_{ij}, v_{ij} e t_{rad} representa a distância relativa, velocidade relativa, e soma dos raios das partículas i e j, respectivamente.

Durante a interação de partículas de fluído com partículas de corpos rígidos ou entre estes, as forças geradas devem obedecer a terceira lei de Newton, a qual afirma que para toda ação resulta uma reação de mesma intensidade porém direções opostas. Dessa forma, o mesmo cálculo de forças apresentado é utilizado durante a interação de fluidos com corpos rígidos e entre corpos rígidos. Assim, cada partícula de corpo rígido armazena o total de forças acumulada para posterior integração.

A utilização de corpos rígidos discretizados em partículas permite o cálculo das forças aplicadas fora do centro de massa com maior facilidade pois não existe a necessidade de cálculos para a localização do ponto exato da colisão, como acontece quando se utiliza triângulos.

Um aspecto muito importante a ser notado aqui é o conceito de pares de corpos rígidos que são gerados durante a colisão. Geralmente, durante o teste de colisão, esses pares de colisão entre objetos são gerados a fim de evitar que um corpo rígido que se encontre no estado de colisão precise verificar uma colisão mais precisa com todos os outros corpos rígidos. Para isso, geralmente utiliza-se uma matriz triangular a fim de efetuar o armazenamento destes pares de colisão.

Nessa arquitetura isto não é necessário. Como uma partícula de um corpo rígido está localizada em uma célula, o teste de colisão é executado somente nesta célula e nas célula adjacentes, não sendo necessário portanto o armazenamento de pares de colisão, conforme pode ser visto na figura 5.8.

5.3.6 Integração

Finalmente, após a identificação de todas as forças que agem sobre a partícula i, $i < N$, é realizado a integração dessas forças para o cálculo da aceleração e posteriormente

velocidade e posição. Para essa integração, foi utilizado o método de Euler explícito com um passo de tempo constante. Nesta etapa também é efetuada a verificação de colisão entre um conjunto de superfícies implícitas, utilizadas neste trabalho para representar os bordos do contêiner do fluido. Para o seu tratamento optou-se pelo reposicionamento da partícula no interior do contânier e a alteração da componente normal da velocidade [4].

Na simulação de fluidos, o cálculo da velocidade final das partículas utiliza o XSPH, conforme discutido na seção 2.4.3.

Para a simulação de corpos rígidos, a integração é feita diretamente na posição do centro de massa. Porém, para os casos onde ocorre colisão é necessário que as forças aplicadas em suas partículas seja transformada em uma força resultante a ser aplicada diretamente no centro de massa. Dessa forma, deve-se efetuar uma operação de somatório dessas forças em paralelo, conhecida como *scan* [19].

Conforme apresentado na seção 5.3.5, existe a necessidade do armazenamento das forças resultantes durante a interação de cada partícula de corpos rígidos selecionados na etapa de *broad phase*. Essas forças são armazenadas em um array contíguo para todas as partículas de todos os corpos rígidos que potencialmente colidiram. Dessa forma, a aplicação do *scan* em gpu no array de forças resultaria em um somatório das forças de todas as partículas de todos os corpos rígidos dinâmicos que colidiram, comportamento este não desejado. A fim de contornar este problema, devemos aplicar um *scan segmentado* [18], onde utiliza-se um array auxiliar indicando com o valor *1* o início de cada segmento. Dessa forma, aplicando um scan segmentado do tipo *backwarded inclusive scan* no array de forças utilizando um array auxilar para indicar início de cada corpo rígido, o somatório desta operação para cada corpo rígido será armazenado na mesma posição onde encontra-se o valor *1* no array de *flags*.

Para essa operação, foi utilizado a biblioteca *CUDPP* [3], que implementa o *scan* segmentado e o não segmentado. Um dos problemas encontrados nesta biblioteca é a implementação destas operações somente para os tipos de dados primitivos, como ponto flutuante e inteiro, não sendo possível utilizar vetores, como é o caso do tipo de dado utilizado no array de armazenamento das forças aplicadas nas partículas. Dessa forma, uma alternativa seria a segmentação desse array por componentes *x, y e z* e execução desta operação três vezes, uma para cada componente. Infelizmente essa opção não se apresentou favorável visto a sua complexidade de desenvolvimento em se trabalhar com os componentes em separado. Com o objetivo de contornar este problema, efetuou-se

[3]Disponível em http://gpgpu.org/developer/cudpp

uma modificação no código da biblioteca através da modificação dos núcleos de *kernels* da biblioteca a fim de permitir a operação com vetores.

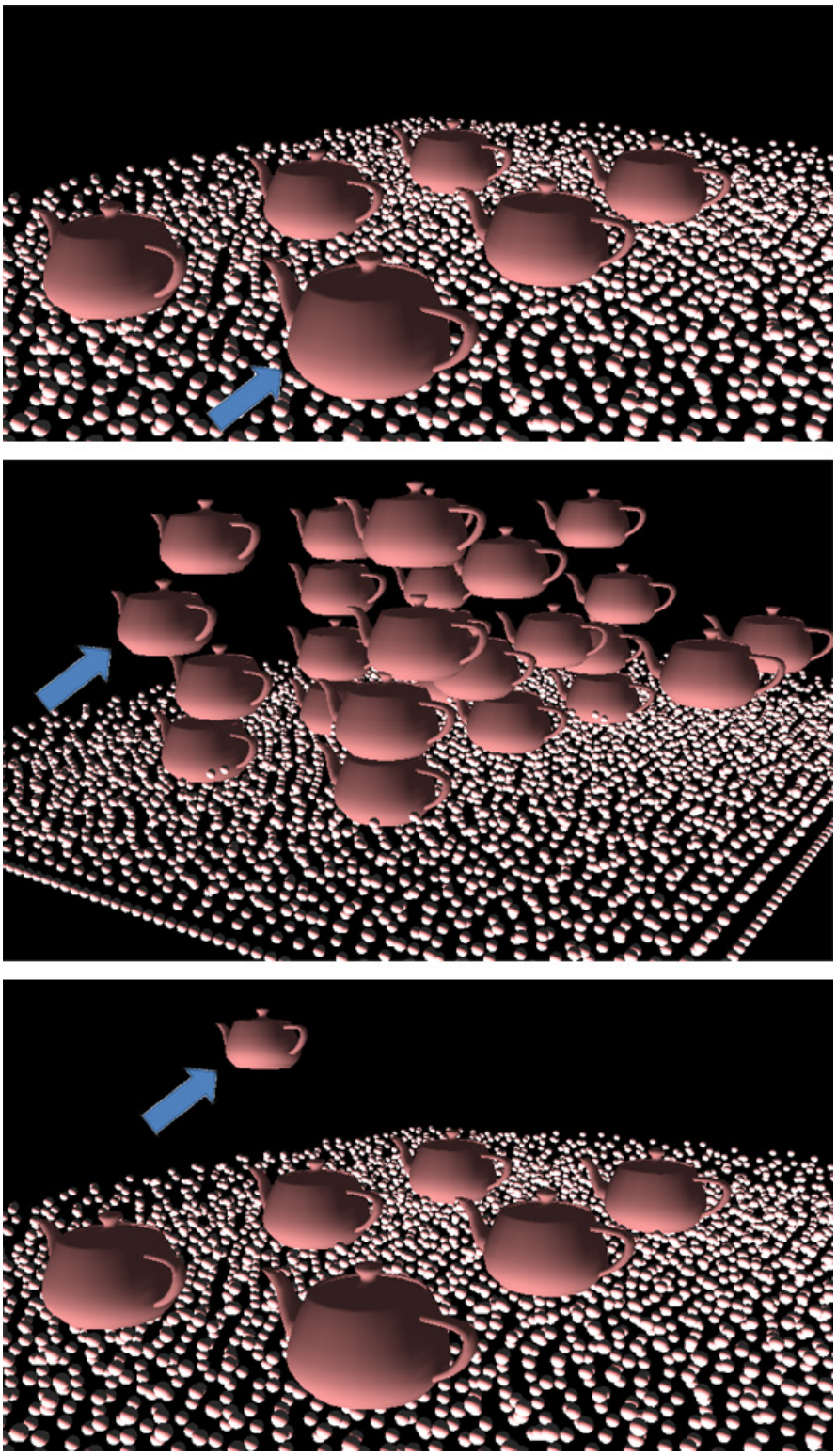

Figura 5.8: Verificação de colisão da partículas em amarelo com as células adjacentes.

Capítulo 6

Resultados

Neste capítulo serão apresentados os resultados obtidos com o trabalho desenvolvido. Primeiramente serão apresentados os resultado obtidos com a inédita estrutura de dados desenvolvida (Seção 6.1). Posteriormente serão apresentados os resultados da simulação de fluidos e corpos rígidos utilizando a arquitetura original heterogênea de gpu e cpu (Seção 6.2).

Para a realização dos testes foi utilizado um PC com processador Intel Core 2 Quad Q6600 com 4GB de memória ram e placa de vídeo NVidia 9600 GT com 512 MB de memória de vídeo.

Para a renderização da cena, é utilizado o *framework* guff (games uff) [47], desenvolvido como um projeto de pesquisa anterior utilizando a linguagem C++ com a incorporação de algumas outras bibliotecas para manipulação de recursos.

6.1 Estrutura de dados

Devido a utilização de partículas de fluidos e corpos rígidos com raios e objetivos diferentes, este trabalho utiliza três grades regulares durante a simulação.

Dessa forma, para permitir a dupla interação entre fluidos e corpos rígidos estáticos e dinâmicos, é necessários o compartilhamento de informações entre essas grades.

Para a realização dos testes, foi utilizada uma cena contendo 50 corpos rígidos, totalizando 3800 partículas, variando o número de partículas do fluido. Para comparação, foi utilizado grades com células capaz de armazenar o maior objeto da cena assim como o método desenvolvido utilizando várias grades.

A tabela 6.1, apresenta o resultado do tempo de acesso de uma partícula a todas as suas células adjacentes utilizando apenas uma grade com tamanho de célula capaz de armazenar o maior objeto, e o método desenvolvido, utilizando duas grades, sendo uma para fluido e outra para corpos rígidos dinâmicos.

Tabela 6.1: Tempo de acesso em segundos utilizando uma e duas grades regulares.

Partículas	4098	8196	16392	32784
Duas grades	0,053	0,101	0,199	0,775
Uma grade	0,062	0,126	0,288	1,062

Na tabela 6.2 é apresentado o resultado do tempo de acesso de uma partícula de fluido utilizando apenas uma grade e o tempo de acesso desta mesma partícula utilizando três grades regulares, sendo uma para fluido, corpos rígidos estáticos e dinâmicos.

Tabela 6.2: Tempo de acesso em segundos utilizando uma e três grades regulares.

Partículas	4098	8196	16392	32784
Três grades	0,101	0,187	0,372	1,381
Uma grade	0,154	0,298	0,661	2,013

Conforme pode-se observar no gráfico apresentado na figura 6.1, a utilização do método desenvolvido oferece um ganho de performance superior a utilizacão de uma grade apenas.

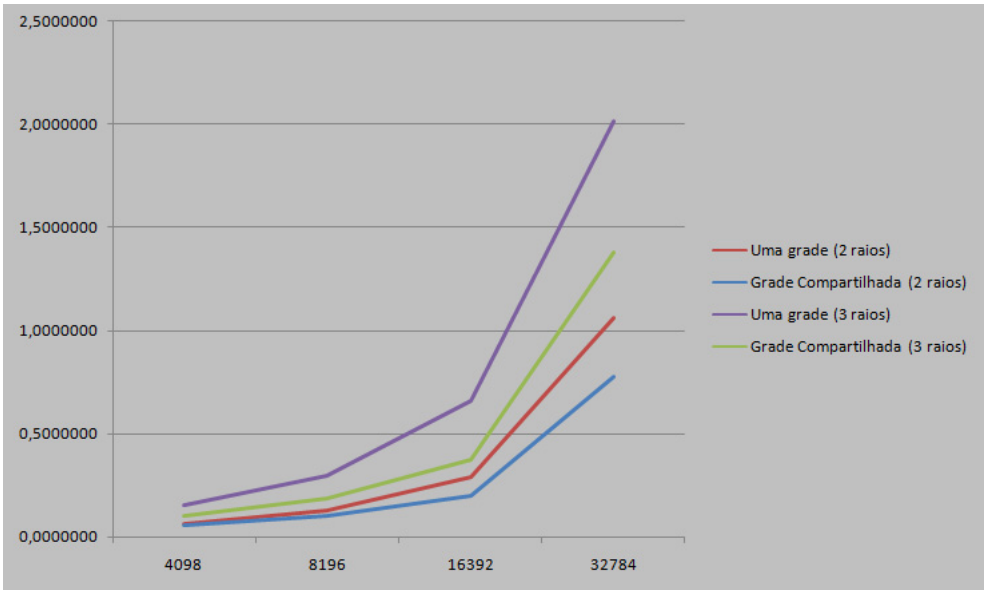

Figura 6.1: Gráfico com o ganho de performance do método desenvolvido.

6.2 Simulação de fluidos e corpos rígidos

Nesta seção são apresentados os resultados obtidos na simulação de fluidos e corpos rígidos utilizando várias configuraçõees de execução para análise.

Primeiramente é apresentado na tabela 6.3 a simulação de fluidos individualmente na gpu e cpu utilizando cenas com diferentes números de partículas. O campo da tabela *IPS* representa o *número de interações por segundo* executados pela simulação, sem a contabilização do tempo necessário para renderização. Devido ao foco do trabalho na simulação de fluidos e corpos rígidos, otimizações não foram feitas na etapa de renderização. Porém, para efeito de comparação, a tabela também exibe o *fps* (frames por segundo) da simulação. Conforme pode-se observar, existe uma grande superioridade da simulação em gpu em relação a cpu. A figura 6.2 apresenta o gráfico dos resultados desta tabela.

Tabela 6.3: Resultado da simulação de fluidos em gpu e cpu.

	GPU				CPU			
Partículas	FPS	Tempo	IPS	Tempo	FPS	Tempo	IPS	Tempo
2048	242,3	0,0041	410,5	0,0024	28,8	0,0346	28,9	0,0345
4096	96,8	0,0103	167,0	0,0059	9,7	0,1020	9,8	0,1017
8192	34,0	0,0293	43,5	0,0229	3,0	0,3288	3,0	0,3281
16394	13,9	0,0718	15,1	0,0661	1,4	0,7062	1,4	0,7027
32768	4,5	0,2181	4,7	0,2095	0,5	1,8939	0,5	1,8867

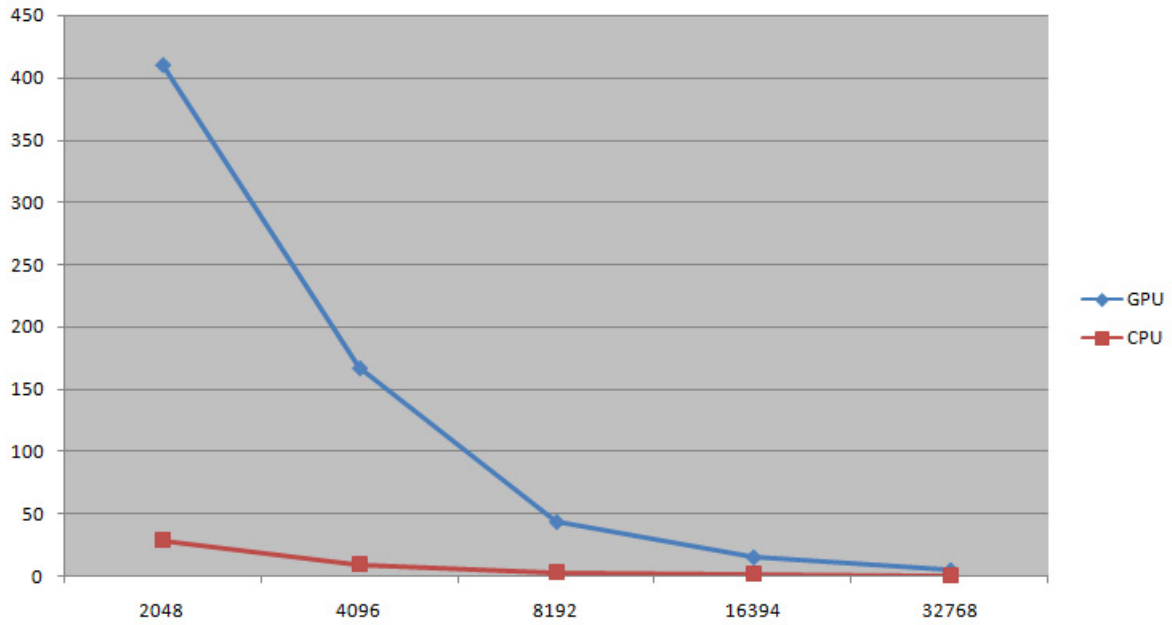

Figura 6.2: Gráfico comparativo da simulação de fluidos em gpu e cpu.

Na tabela 6.4 é apresentado a simulação de corpos rígidos utilizando diferentes cenários

em gpu e cpu, assim como seu gráfico comparativo na figura 6.3. Para este resultado foram utilizados um conjunto de objetos *teapot* discretizado utilizando 76 partículas com raio de *0,5*, conforme pode ser observado na figura 6.4, onde esta cena é simulada.

Tabela 6.4: Resultado da simulação de corpos rígidos em gpu e cpu. CR: corpos rígidos.

		GPU				CPU			
CR	Partículas	FPS	Tempo	IPS	Tempo	FPS	Tempo	IPS	Tempo
26	1976	220,7	0,0045	227,5	0,0043	19,0	0,0525	19,0	0,0524
53	4028	210,8	0,0047	223,6	0,0044	8,6	0,1160	8,6	0,1157
107	8132	159,9	0,0062	174,7	0,0057	4,1	0,2435	4,1	0,2431
215	16340	102,2	0,0097	114,6	0,0087	2,0	0,5017	2,0	0,5010
430	32680	63,0	0,0158	72,6	0,0137	0,9	1,0214	0,9	1,0204

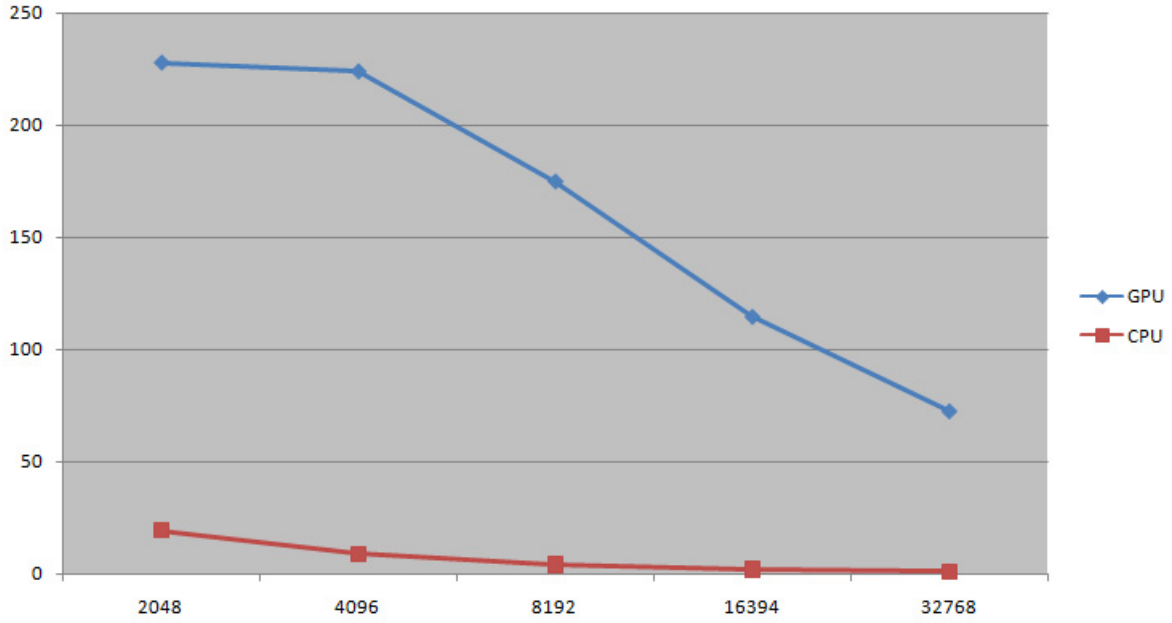

Figura 6.3: Gráfico comparativo da simulação de corpos rígidos em gpu e cpu.

Os resultados da simulação de fluidos e corpos rígidos simultaneamente são executados em duas etapas. Primeiramente a tabela 6.5 apresenta os resultados da simulação de fluidos e corpos rígidos simultaneamente em gpu e cpu, porém sem a interação entre estes. A figura 6.5 apresenta o gráfico desta simulação.

Finalmente são apresentados nas tabelas 6.6, 6.7 e 6.8 o resultado da simulação de fluidos e corpos rígidos com dupla interação em gpu, cpu e modo heterogêneo (cpu e gpu), respectivamente. A figura 6.6 apresenta o gráfico deste resultado. Conforme pode-se observar na figura 6.7, os fluidos exercem forças nos objetos, fazendo os mesmos flutuarem sobre o fluido, através de um empuxo, o qual é baseado no peso do corpo rígido. Na

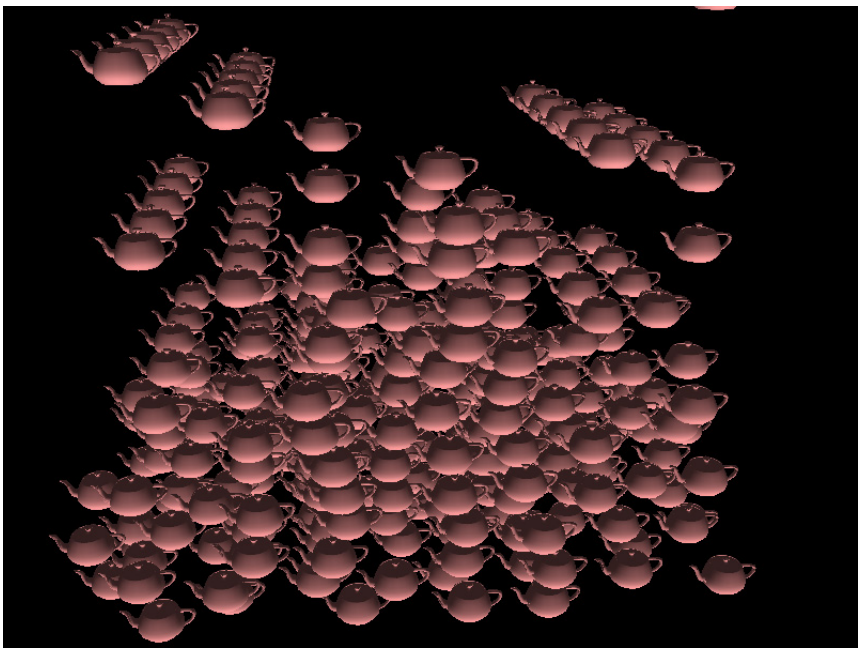

Figura 6.4: Renderização de uma cena da simulação de 430 corpos rígidos totalizando 32680 partículas.

Tabela 6.5: Resultado da simulação de fluidos e corpos rígidos em gpu e cpu sem interação. CR: corpos rígidos; CRP: numero de partículas de corpos rígidos; CF: número de partículas de fluido; TPS: total de partículas no sistema.

				GPU				CPU			
CF	CR	CRP	TPS	FPS	Tempo	IPS	Tempo	FPS	Tempo	IPS	Tempo
4096	200	15200	19296	49,5	0,0202	57,4	0,0174	1,6	0,6169	1,6	0,6028
4096	400	30400	34496	43,3	0,0231	50,2	0,0199	0,8	1,1261	0,9	1,0941
8192	200	15200	23392	25,8	0,0388	27,7	0,0361	1,2	0,8190	1,2	0,8045
8192	400	30400	38592	23,1	0,0433	25,1	0,0397	0,7	1,3477	0,7	1,3158

figura 6.8 pode-se verificar a força exercida pelo corpo rígido no fluido, causando seu deslocamento.

Conforme pode-se observar pelo resultado apresentado, a utilização desta arquitetura heterogênea de cpu e gpu proporcionou um ganho satisfatório durante a simulação quando comparada somente a gpu. Analisando a a tabela 6.9, percebe-se que foi obtido um ganho maior que 50% sobre a mesma cena simulada utilizando apenas a gpu.

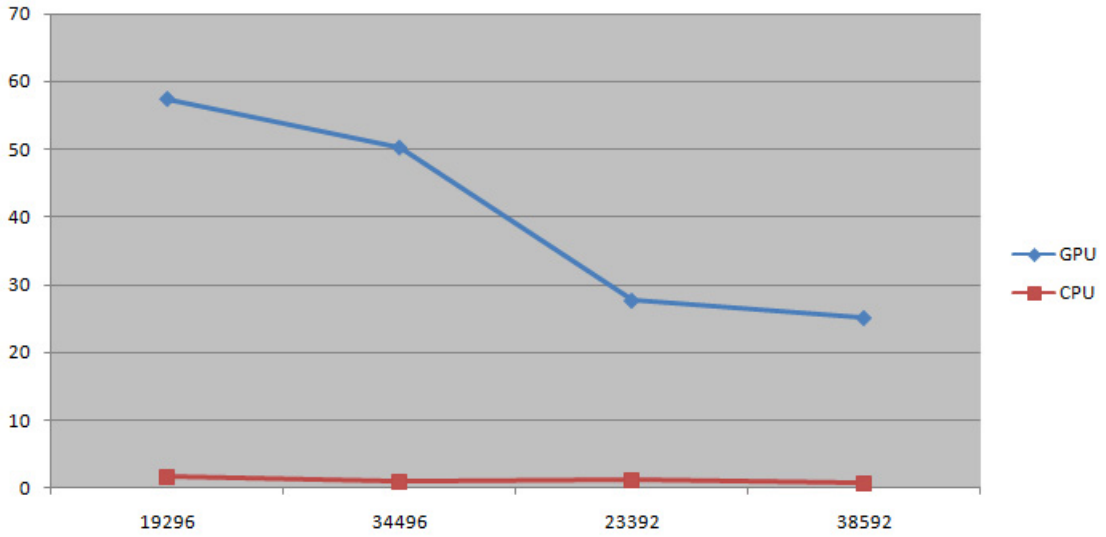

Figura 6.5: Gráfico comparativo da simulação de fluidos e corpos rígidos em gpu e cpu sem interação.

Tabela 6.6: Resultado da simulação de fluidos e corpos rígidos em gpu com dupla interação. CR: corpos rígidos; CRP: numero de partículas de corpos rígidos; CF: número de partículas de fluido; TPS: total de partículas no sistema.

				GPU			
CF	CR	CRP	TPS	FPS	Tempo	IPS	Tempo
4096	200	15200	19296	27,7	0,0361	29,5	0,0339
4096	400	30400	34496	20,6	0,0485	22,0	0,0455
8192	200	15200	23392	18,7	0,0534	19,7	0,0508
8192	400	30400	38592	14,0	0,0711	14,9	0,0672

Tabela 6.7: Resultado da simulação de fluidos e corpos rígidos em cpu com dupla interação. CR: corpos rígidos; CRP: número de partículas de corpos rígidos; CF: número de partículas de fluido; TPS: total de partículas no sistema.

				CPU			
CF	CR	CRP	TPS	FPS	Tempo	IPS	Tempo
4096	200	15200	19296	0,6	1,6077	0,6	1,5924
4096	400	30400	34496	0,3	2,9674	0,3	2,9240
8192	200	15200	23392	0,5	2,0202	0,5	2,0080
8192	400	30400	38592	0,3	3,4130	0,3	3,3784

Tabela 6.8: Resultado da simulação de fluidos e corpos rígidos em modo heterogêneo de cpu e gpu com dupla interação. CR: corpos rígidos; CRP: número de partículas de corpos rígidos; CF: número de partículas de fluido; TPS: total de partículas no sistema.

				Heterogêneo (cpu + gpu)			
CF	CR	CRP	TPS	FPS	Tempo	IPS	Tempo
4096	200	15200	19296	27,9	0,0359	53,4	0,0187
4096	400	30400	34496	18,8	0,0593	52,6	0,0190
8192	200	15200	23392	18,3	0,0545	28,2	0,0355
8192	400	30400	38592	12,2	0,0817	27,6	0,0363

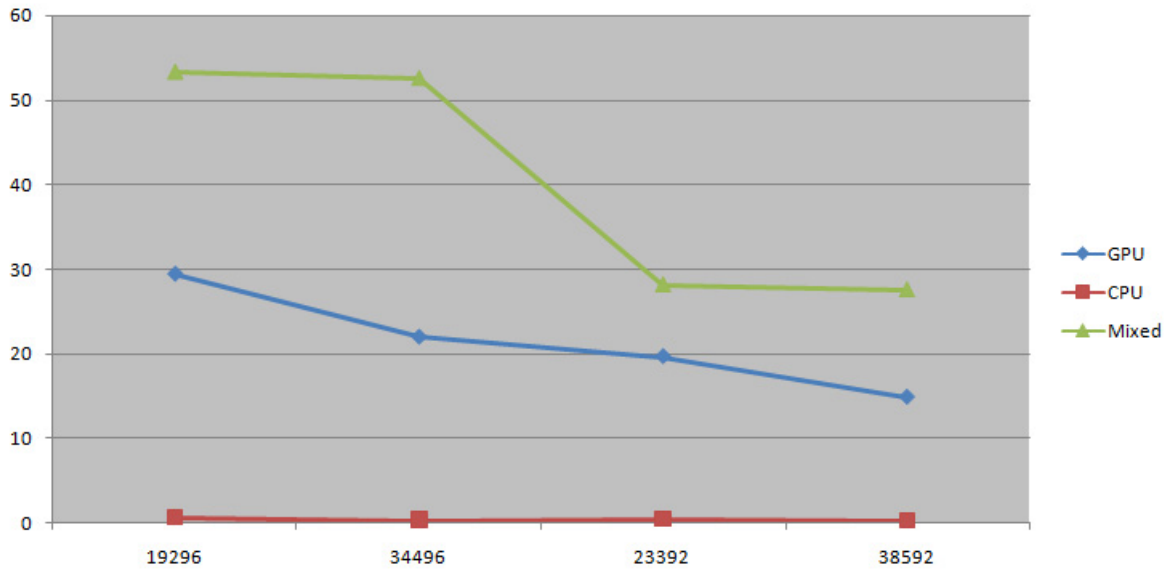

Figura 6.6: Gráfico comparativo da simulação de fluidos e corpos rígidos em gpu, cpu e modo heterogêneo(gpu e cpu) com dupla interação.

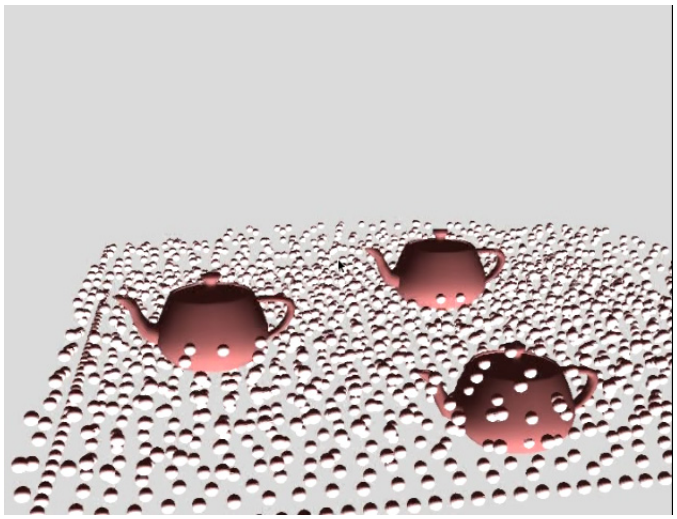

Figura 6.7: Renderização de uma cena onde verifica-se o comportamento do corpo rígido através da aplicação de forças pelo fluido.

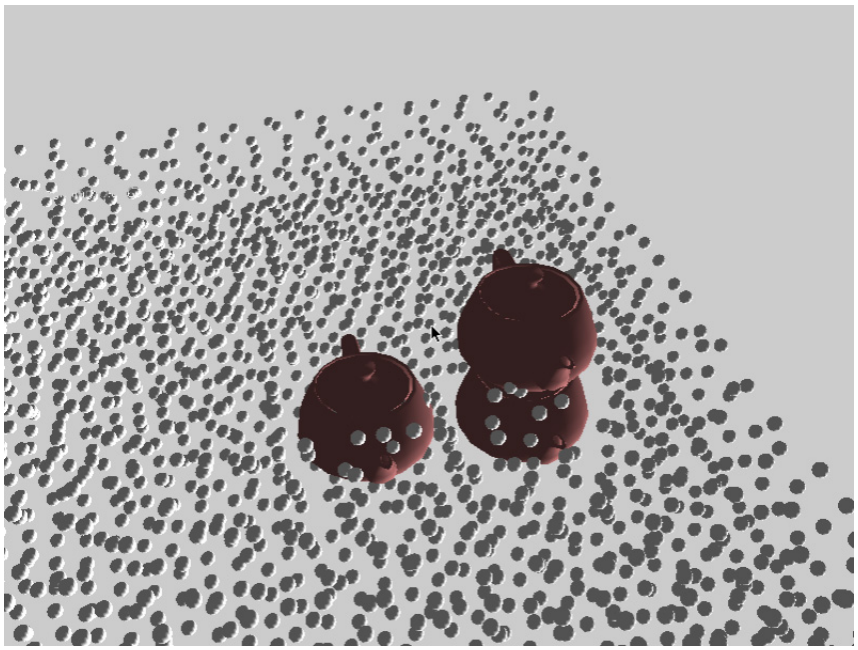

Figura 6.8: Renderização de uma cena onde verifica-se o comportamento do fluido através da aplicação de forças pelo corpo rígido.

Tabela 6.9: Ganho obtido com a arquitetura heterogênea de cpu e gpu em relação a utilização apenas de gpu. CR: corpos rígidos; CRP: número de partículas de corpos rígidos; CF: número de partículas de fluido; TPS: total de partículas no sistema.

				gpu		Heterogêneo (cpu + gpu)		
CF	CR	CRP	TPS	IPS	Tempo	IPS	Tempo	Eficiência
4096	200	15200	19296	29,5	0,0339	53,4	0,0187	44,79%
4096	400	30400	34496	22,0	0,0455	52,6	0,0190	58,20%
8192	200	15200	23392	19,7	0,0508	28,2	0,0355	30,02%
8192	400	30400	38592	14,9	0,0672	27,6	0,0363	45,05%

Capítulo 7

Conclusão e trabalhos futuros

Conforme pode-se observar nos resultados apresentados, o desempenho da estrutura de dados desenvolvida neste trabalho apresentou-se bastante satisfatória para comunicação entre espaços de memórias individuais e independentes, a qual foi utilizada neste trabalho para permitir a interação entre corpos rígidos estáticos, dinâmicos e fluidos, cada uma utilizando grades diferentes. Dessa forma, este trabalho mostrou que o tempo necessário para o acesso a espaços de memória diferentes é melhor realizado utilizando a estrutura de dados desenvolvida em relação a uma única estrutura de dados capaz de armazenar todos os objetos em uma única célula, utilizando neste caso um tamanho de célula capaz de armazenar o maior dos objetos da cena.

A simulação de corpos rígidos apresentou um comportamento mais linear conforme aumenta-se o número de partículas. Ao contrário, a simulação de fluidos apresentou um comportamento próximo ao quadrático, com o aumento do número de partículas, conforme os resultados apresentados.

Com essa observação, a simulação de fluidos e corpos rígidos tende a apresentar um comportamento linear ao aumentar o número de corpos rígidos e quadrático com o aumento do número de partículas de fluido. Esses resultados podem ser observados nos resultados das simulações de corpos rígidos e fluidos com e sem dupla interação.

A nova arquitetura heterogênea de cpu e gpu, apesar de não modificar este comportamento, apresentou um resultado global da simulação muito satisfatório. Conforme apresentado nos resultados, a performance obtida com esta nova arquitetura obteve um ganho próximo de 60% em comparação com a mesma simulação executada somente pela gpu.

Com o resultado apresentado, a investigação da utilização da cpu e gpu mostrou-se

válida, instigando o uso de cpu's com um maior número de núcleos para processamento da simulação.

Como trabalho futuro, alguns pontos na simulação e arquitetura desenvolvida podem ser melhorados e extendidos. O primeiro deles seria o tratamento da colisão entre corpos rígidos. No tipo de tratamento desenvolvido existe a necessidade do ajuste fino de parâmetros de elasticidade e amortecimento na geração de forças além de sua modificação para diversos tipos de simulações a fim de evitar possíveis interpenetrações entre essas partículas, tornando-o um método muito sensível a parâmetros com grande dificuldade de configuração.

Além disso, com a nova arquitetura FERMI de gpu's que permite a execução de múltiplos *kernels*, o trabalho desenvolvido pode ser extendido para tirar proveito dessa nova arquitetura com o objetivo de permitir um maior número de corpos rígidos e partículas de fluidos na simulação.

Devido a utilização de partículas para detecção e tratamento de colisão de corpos rígidos assim como a utilização de múltiplas estruturas de grades regular, um sistema de LOD de colisão pode ser desenvolvido, onde corpos rígidos fora do foco da simulação utilizam discretizações com menor número de partículas, enquanto aquelas dentro do foco de visão são processadas mais detalhadamente através de um número maior de partículas.

Referências

[1] FELDMAN, B. E. et al. Fluids in deforming meshes. In: *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM, 2005. p. 255–259. ISBN 1-7695-2270-X.

[2] YNGVE, G. D.; O'BRIEN, J. F.; HODGINS, J. K. Animating explosions. In: *Proceedings of ACM SIGGRAPH 2000*. ACM, 2000. p. 29–36. Disponível em: <http://graphics.cs.berkeley.edu/papers/Yngve-AEX-2000-07/>.

[3] TRAPP, M.; DÖLLNER, J. Real-time volumetric tests using layered depth images. In: *Eurographics 2008*. [S.l.]: The Eurographics Association, 2008. p. 235–238.

[4] MüLLER, M.; CHARYPAR, D.; GROSS, M. Particle-based fluid simulation for interactive applications. In: *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003. p. 154–159. ISBN 1-58113-659-5.

[5] EVERITT, C., Interactive order-independent transparency, NVidia Corporation, 2001.

[6] LIU, B.; WEI, L.-Y.; XU, Y.-Q., Multi-layer depth peeling via fragment sort, Microsoft Research, 2006.

[7] HARADA, T. Real-time rigid body simulation on gpus. In: NGUYEN, H. (Ed.). *GPU Gems 3*. [S.l.]: Addison-Wesley, 2007. p. 611–631.

[8] KASS, M.; MILLER, G. Rapid, stable fluid dynamics for computer graphics. In: *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1990. p. 49–57. ISBN 0-89791-344-2.

[9] SIMS, K. Particle animation and rendering using data parallel computation. In: *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1990. p. 405–413. ISBN 0-89791-344-2.

[10] PAIVA, A. et al. Simulação de fluidos sem malha: Uma introdução ao método sph. In: *27º Colóquio Brasileiro de Matemática*. Rio de Janeiro, RJ, Brasil: IMPA, 2009. p. 198.

[11] LIU, G. R.; LIU, M. B. *Smoothed Particle Hydrodynamics: A Meshfree Particle Method; electronic version*. Singapore: World Scientific, 2003.

[12] MISHRA, B. K. A review of computer simulation of tumbling mills by dem part i - contact mechanics. In: *International Journal of Mineral Processing, Vol. 71(1-4)*. [S.l.: s.n.], 2003. p. 73–93.

[13] O'BRIEN, J. F.; HODGINS, J. K. Dynamic simulation of splashing fluids. In: *CA '95: Proceedings of the Computer Animation*. Washington, DC, USA: IEEE Computer Society, 1995. p. 198. ISBN 0-8186-7062-2.

[14] TESCHNER, M. et al. Optimized spatial hashing for collision detection of deformable objects. In: *In Proceedings of VMV'03*. [S.l.: s.n.], 2003. p. 47–54.

[15] DESBRUN, M.; GASCUEL, M. paule. Smoothed particles: A new paradigm for animating highly deformable bodies. In: *In Computer Animation and Simulation 96 (Proceedings of EG Workshop on Animation and Simulation*. [S.l.]: Springer-Verlag, 1996. p. 61–76.

[16] LUCY, L. B. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, v. 82, p. 1013–1024, 1977.

[17] GINGOLD, R. A.; MONAGHAN, J. J. Smoothed particle hydrodynamics - theory and application to non-spherical stars. In: *Royal Astronomical Society, Monthly Notices, vol. 181*. [S.l.: s.n.], 1977. p. 375–389.

[18] SENGUPTA, S. et al. Scan primitives for gpu computing. In: ACM. *Graphics Hardware 2007*. [S.l.], 2007. p. 97–106.

[19] HARRIS, M.; SENGUPTA, S.; OWENS, J. D. Parallel prefix sum (scan) with cuda. In: NGUYEN, H. (Ed.). *GPU Gems 3*. [S.l.]: Addison Wesley, 2007.

[20] MüLLER, M. et al. Interaction of fluids with deformable solids. *Comput. Animat. Virtual Worlds*, John Wiley and Sons Ltd., Chichester, UK, v. 15, n. 3-4, p. 159–171, 2004. ISSN 1546-4261.

[21] MüLLER, M. et al. Particle-based fluid-fluid interaction. In: *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM, 2005. p. 237–244. ISBN 1-7695-2270-X.

[22] STORA, D. et al. Animating lava flows. In: *Proceedings of the 1999 conference on Graphics interface '99*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. p. 203–210. ISBN 1-55860-632-7.

[23] STAM, J. Stable fluids. In: *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999. p. 121–128. ISBN 0-201-48560-5.

[24] STAM, J. Real-time fluid dynamics for games. In: *Proceedings of the Game Developer Conference*. [S.l.: s.n.], 2003.

[25] KUROSE, S.; TAKAHASHI, S. Constraint-based simulation of interactions between fluids and unconstrained rigid bodies. In: *Proceedings of Spring Conference on Computer Graphics*. [S.l.: s.n.], 2009. p. 197–204.

[26] GILAT, A.; SUBRAMANIAM, V. *Métodos numéricos para engenheiros e cientistas.* Porto Alegre: Bookman, 2008.

[27] CUADROS-VARGAS, A. et al. Imesh: An image based quality mesh generation technique. In: *IEEE Proceedings SIBGRAPI.* [S.l.: s.n.], 2005. p. 341–348.

[28] SHEWCHUK, J. R. *Delaunay refinement mesh generation.* Tese (Doutorado) — Carnegie Mellon University, 1997.

[29] FIRES, T.; MATTHIES, H. G., Classification and overview of meshfree methods, Technical University Braunschweig, Brunswick, Alemanha, 2004.

[30] QUEIROZ, T. E. *Animação computacional de escoamento de fluidos utilizando o método SPH.* Dissertação (Mestrado), 2008.

[31] MARGOLIN, L. G. Introduction to "an arbitrary lagrangian-eulerian computing method for all flow speeds". *J. Comput. Phys.*, Academic Press Professional, Inc., San Diego, CA, USA, v. 135, n. 2, p. 198–202, 1997. ISSN 0021-9991.

[32] KOSHIZUKA, S.; NOBE, A.; OKA, Y. Numerical analysis of breaking waves using the moving particle semi-implicit method. *International Journal for Numerical Methods in Fluids.*, v. 26, p. 751–769, 1998.

[33] CUNDALL, P. A. Distinct element models of rock and soil structure. In: *Brown (Ed.) Analytical and Computational Methods in Engineering Rock Mechanics.* [S.l.: s.n.], 1987. p. 129–163.

[34] NAYROLES, B.; TOUZOT, G.; VILLON, P. Generalizing the finite element methods: diffuse approximation and diffuse elements. *Computational Mechanics*, n. 10, p. 307–318, 1992.

[35] FORTUNA, A. d. O. *Técnicas computacionais para dinâmica dos fluidos.* São Paulo: Editora da Universidade de São Paulo, 2000.

[36] ERLEBEN, K. et al. *Physics based animation.* San Diego, CA, USA: Charles River Media, 2005.

[37] ERLEBEN, C. *Real-time collision detection.* San Francisco, CA, USA: Elsevier, 2005.

[38] MONAGHAN, J. J. Smoothed particle hydrodynamics. *Annual Reviews Astronomy and Astrophysics*, v. 30, p. 543–573, 1992.

[39] KELAGER, M. *Lagrangian fluid dynamics using smoothed particle hydrodynamics.* Dissertação (Mestrado), 2006.

[40] JOSELLI, M. et al. An adaptative game loop architecture with automatic distribution of tasks between cpu and gpu. *Comput. Entertain.*, ACM, New York, NY, USA, v. 7, n. 4, p. 1–15, 2009. ISSN 1544-3574.

[41] MOORE, M.; WILHELMS, J. Collision detection and response for computer animationr3. In: *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques.* New York, NY, USA: ACM, 1988. p. 289–298. ISBN 0-89791-275-6.

[42] CARLSON, M.; MUCHA, P. J.; TURK, G. Rigid fluid: animating the interplay between rigid bodies and fluid. In: *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*. New York, NY, USA: ACM, 2004. p. 377–384.

[43] SAMET, H. *The design and analysis of spatial data structures*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990. ISBN 0-201-50255-0.

[44] PUECH, C.; YAHIA, H. Quadtrees, octrees, hyperoctrees: a unified analytical approach to tree data structures used in graphics, geometric modeling and image processing. In: *SCG '85: Proceedings of the first annual symposium on Computational geometry*. New York, NY, USA: ACM, 1985. p. 272–280. ISBN 0-89791-163-6.

[45] LEFEBVRE, S.; HOPPE, H. Perfect spatial hashing. In: *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*. New York, NY, USA: ACM, 2006. p. 579–588. ISBN 1-59593-364-6.

[46] BELL, N.; YU, Y.; MUCHA, P. J. Particle-based simulation of granular materials. In: *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM, 2005. p. 77–86. ISBN 1-7695-2270-X.

[47] VALENTE, L. *Guff: um framework para desenvolvimento de jogos*. Dissertação (Mestrado) — Universidade Federal Fluminense, 2005.

[48] KIPFER, P.; WESTERMANN, R. Realistic and interactive simulation of rivers. In: *GI '06: Proceedings of Graphics Interface 2006*. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2006. p. 41–48. ISBN 1-56881-308-2.

[49] ZHANG, Y.; SOLENTHALER, B.; PAJAROLA, R. Gpu accelerated sph particle simulation and rendering. In: *SIGGRAPH '07: ACM SIGGRAPH 2007 posters*. New York, NY, USA: ACM, 2007. p. 9.

[50] ZHANG, Y.; SOLENTHALER, B.; PAJAROLA, R. Adaptive sampling and rendering of fluids on the gpu. In: *Eurographics/IEEE VGTC Symposium on Point-Based Graphics*. New York, NY, USA: [s.n.], 2008. p. 137–146.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo