UNIVERSIDADE FEDERAL FLUMINENSE

YAN RAMOS DA SILVA

SIMULAÇÃO DO ESCOAMENTO DE FLUIDOS BIFÁSICOS EM MEIOS POROSOS UTILIZANDO SPH

NITERÓI

2016

UNIVERSIDADE FEDERAL FLUMINENSE

YAN RAMOS DA SILVA

SIMULAÇÃO DO ESCOAMENTO DE FLUIDOS BIFÁSICOS EM MEIOS POROSOS UTILIZANDO SPH

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

Orientador:
MARCOS DE OLIVEIRA LAGE FERREIRA

NITERÓI

2016

YAN RAMOS DA SILVA

SIMULAÇÃO DO ESCOAMENTO DE FLUIDOS BIFÁSICOS EM MEIOS POROSOS UTILIZANDO SPH

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

Aprovada em Julho de 2016.

BANCA EXAMINADORA

Prof. Dr. Marcos de Oliveira Lage Ferreira - Orientador
Universidade Federal Fluminense

Prof. Dr. Sinésio Pesco
PUC-Rio de Janeiro

Prof. Dr. Waldemar Celes
PUC-Rio de Janeiro

Prof. Dr. Anselmo Montenegro
Universidade Federal Fluminense

Niterói
2016

A todos que queriam me fazer acreditar que eu não era capaz, mas, sobretudo, aos que me lembraram que eu era.

Agradecimentos

Agradeço ao prof. Marcos Lage pelo auxílio e orientação desde os últimos anos de graduação. Agradeço também aos meus pais Odete e Jorge Luiz e minha irmã Tayama por acreditarem em mim e investirem em minha educação em todos esses anos. Agradeço, sobretudo, aos meus amigos João Henrique, Nícholas e Paulo Roberto e à dra. Emília Lobato por toda a paciência e motivação e por nunca me deixarem nem sequer pensar em desistir.

This door... is more than it appears to be. It separates who you are from who you can be. You do not have to walk through it... you can run.

Jonathan Hickman

Resumo

A simulação computacional do escoamento de fluidos é uma tarefa não-trivial, pois envolve a definição de algoritmos computacionalmente intensivos para a realização do cálculo das soluções numéricas das equações de Navier-Stokes. Neste contexto, o "nível de realismo" e o tempo de execução de uma simulação estão intimamente relacionados com a precisão numérica da solução obtida e devem ser escolhidos de acordo com o tipo de aplicação. Uma das estratégias mais populares para a realização da simulação computacional do escoamento de um fluido é o método *Smoothed Particles Hydrodynamics* (SPH). O método trata o fluido como um conjunto de partículas e aproxima as grandezas físicas através de cálculos sobre partículas vizinhas. Este trabalho utiliza o método SPH para simular o escoamento de fluidos bifásicos através de meios porosos. Para isso, o domínio poroso é representado por um *grid* cujas células representam regiões sólidas ou poros. O tratamento das condições de contorno é realizado através do cálculo de colisões geométricas entre as partículas de fluido e a superfície das células sólidas.

Palavras-chave: meios porosos, fluidos bifásicos, Smoothed Particles Hydrodynamics, computação gráfica.

Abstract

The computational simulation of fluid flows is a a non-trivial task since it involves the definition of computationally intensive algorithms for carrying the calculation of the numerical solutions to the Navier-Stokes equations. In this context, the "level of realism" and runtime of a simulation are deeply related to the precision of the obtained numerical solution and must be chosen according to the application type. One of the most popular strategies for the computational simulation of fluid flows is the *Smoothed Particles Hydrodynamics* (SPH) method. It treats the fluid as a set of particles and approximates physical quantities through calculations over neighbor particles. This work uses the SPH method to simulate the flow of biphasic fluids through porous media. To achieve that, the porous domain is represented by a *grid* whose cells represent either solid regions or pores. The treatment of boundary conditions is made through the computation of geometric collisions between the particles of the fluid and the surface of the solid cells.

Keywords: porous media, biphasic fluids, Smoothed Particles Hydrodynamics, computer graphics.

Lista de Figuras

1.1 Simulação de fluidos (água e lava) no jogo eletrônico *From Dust* (2011). . . 2

1.2 Simulação de fluidos no filme *Além da Vida* (2010). 2

1.3 Exemplo de uso de partículas fantasma. Em (a), o suporte da partícula está incompleto devido à sua proximidade de um obstáculo. Ao adicionar as partículas fantasma em (b), o suporte passa a estar completo. 3

3.1 Exemplo de simulação monofásica utilizando o método SPH, gerada com o programa desenvolvido durante este trabalho. 11

3.2 Exemplo de função núcleo de suavização. 12

3.3 Exemplo de atribuição de identificadores a partículas pertencentes a cada uma das fases do fluido. 16

3.4 Exemplo de direção do vetor gradiente de densidade. 17

3.5 Particionamento arbitrário de um obstáculo poroso. 18

3.6 Exemplo de uso do particionamento do obstáculo para o tratamento da condição de contorno. Utilizando-se a posição da partícula (a), isola-se a partição à qual ela pertence (b) e testa-se a colisão geométrica apenas com as componentes da malha que também pertencem à mesma partição (c). . 18

3.7 Esquema da colisão geométrica de uma partícula com um polígono da malha do obstáculo. 19

3.8 Reflexão e amortecimento da velocidade causado pela colisão. 19

4.1 Representação bidimensional do *grid* de busca. 24

5.1 Técnica de de colisão utilizada. A figura A representa a posição antiga da partícula e seu vetor velocidade. B mostra a sua posição recém-calculada, que está numa célula preenchida, e C mostra sua nova posição e velocidade após o tratamento. 37

6.1 Cenário dos testes com fluido monofásico. 42

6.2 Capturas de tela de cenários do teste bifásico. 43

6.3 Porosidade do obstáculo $\times$ percentual de volume de fluido absorvido pelo obstáculo. 44

6.4 Cenário dos testes com fluido bifásico. 45

6.5 Corte realizado nos testes com fluido bifásico. 46

6.6 Capturas de tela de cenários do teste bifásico 47

6.7 Cortes transversais do obstáculo poroso, compreendendo a camada de células com coordenada $y \in [-2.0, 0.0]$. 48

6.8 Porosidade do obstáculo $\times$ razão entre volume de fluido removido e inserido no obstáculo. 48

Sumário

1 Introdução **1**

1.1 Motivação 1

1.2 Contribuições 3

1.3 Organização do texto 4

2 Trabalhos relacionados **5**

2.1 Simulação de fluidos monofásicos usando SPH 5

2.2 Simulação de fluidos bifásicos usando SPH 8

2.3 Escoamento de fluidos em meios porosos 9

3 Descrição do método matemático **11**

3.1 SPH monofásico 11

Funções núcleo de suavização. 12

Cálculo das forças. 14

3.2 SPH bifásico 15

3.3 SPH em meios porosos 17

4 Considerações computacionais **21**

4.1 *Loop* principal 21

4.2 Inicialização do fluido 22

4.3 Busca 23

4.4 Integração numérica 25

4.5 Detecção e tratamento de condições de contorno 26

5 Implementação **28**

5.1 Estrutura de classes 28

5.2 Arquivos de configuração 28

5.3 Estrutura da implementação 29

5.4 Inicialização dos elementos da simulação 30

Cena 30

Tanque 30

Grid de busca 30

Grid de obstáculo 31

Sistema de fluidos 33

5.4.1 *Loop* principal 36

5.4.2 Detecção e tratamento de colisões 37

6 Resultados **41**

6.1 Metodologia dos experimentos 41

6.2 Teste 1: escoamento de fluido monofásico 42

6.3 Teste 2: escoamento de fluido bifásico 45

7 Conclusão **51**

7.1 Desafios e trabalhos futuros 52

Referências **53**

Apêndice A – Diagrama de classes **55**

Apêndice B – Descrição das classes **56**

Capítulo 1

Introdução

Este trabalho tem como objetivo a criação de uma técnica para a simulação computacional do escoamento de fluidos bifásicos através de meios porosos baseada no método SPH (*Smoothed Particle Hydrodynamics*). Este capítulo introduzirá este trabalho, apresentando também motivações para a escolha deste tema de pesquisa.

1.1 Motivação

A dinâmica de fluidos computacional é de grande interesse da comunidade científica, tendo em vista a vasta gama de aplicações na qual ela pode ser utilizada. Técnicas desenvolvidas no estudo desta área são empregadas, por exemplo, na Astrofísica, para simular o comportamento de corpos celestes como estrelas, nebulosas e galáxias, assim como as interações entre eles. Na Engenharia, elas são utilizadas para realizar estimativas quanto a resistência de materiais suscetíveis à ação de fluidos, como barragens e fuselagens de aviões, além de também serem relevantes, por exemplo, para estudos de viabilidade de prospecção de petróleo.

Pesquisadores da área de Computação também têm bastante interesse em técnicas de simulação de fluidos. As indústrias de jogos eletrônicos, além de estúdios de animação e efeitos visuais, dentre outros, utilizam amplamente estas técnicas para simular não apenas substâncias nos estados líquido e gasoso, mas também outros materiais que comportam-se de maneira análoga a fluidos, como neve e areia, por exemplo. As Figuras 1.1 e 1.2 ilustram dois usos da simulação de fluidos na indústria do entretenimento.

Figura 1.1: Simulação de fluidos (água e lava) no jogo eletrônico *From Dust* (2011).

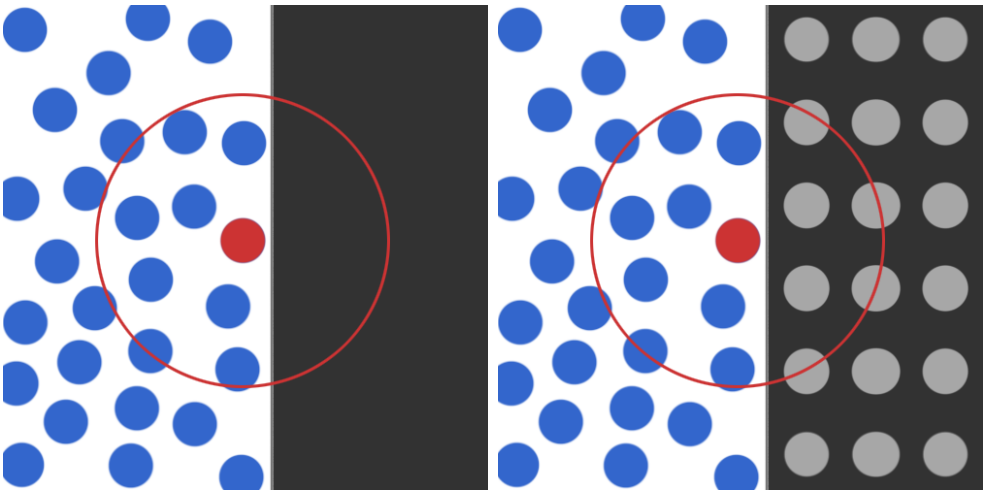
Figura 1.2: Simulação de fluidos no filme *Além da Vida* (2010).

Além de sua versatilidade, a complexidade intrínseca à dinâmica de fluidos também é uma característica que fomenta os estudos realizados nessa área. A simulação de escoamentos com um nível suficiente de realismo para causar a mesma sensação visual de uma cena real em um observador requer um extenso ferramental matemático e computacional, cuja modelagem é pouco trivial. De fato, os pesquisadores que trabalham na área buscam desenvolver técnicas que produzam resultados cada vez mais realistas de forma numericamente robusta e computacionalmente eficiente. Estes requisitos mostram-se de grande importância nas aplicações gráficas, nas quais o tempo necessário para o cálculo da solução é uma variável crítica, em especial naquelas cujas cenas são geradas em tempo real, como os jogos eletrônicos, por exemplo.

Os desafios são ainda maiores quando a cena é composta por fluidos bifásicos ou envolvem escoamentos em meios porosos, temas abordados neste trabalho. Isto se deve à complexidade inerente ao tratamento adequado da interação entre as fases imiscíveis que compõem o fluido e do mesmo com as paredes do obstáculo poroso, de geometria intricada, através do qual ele escoa.

1.2 Contribuições

A estratégia mais popular para o tratamento de condições de contorno em simulações de fluidos utilizando o método SPH é baseada no conceito de partículas fantasma. Elas são partículas que permanecem estacionárias durante toda simulação, preenchendo as áreas da cena intransponíveis pelo fluido. Isto é feito pois as partículas de fluido localizadas nas proximidades destas regiões possuem parte de sua vizinhança no interior destes obstáculos, nos quais não há fluido e, consequentemente, partículas. Esta incompletude do suporte da partícula faz com que seu movimento seja em direção ao espaço vazio, de forma a preenchê-lo, fazendo com o que o fluido atravesse a superfície do obstáculo, apresentando um comportamento diferente do desejado. A Figura 1.3 ilustra a utilização de partículas fantasma para o tratamento da condição de contorno.



(a) (b)

Figura 1.3: Exemplo de uso de partículas fantasma. Em (a), o suporte da partícula está incompleto devido à sua proximidade de um obstáculo. Ao adicionar as partículas fantasma em (b), o suporte passa a estar completo.

Esta técnica possui a vantagem de impor as condições de contorno automaticamente durante o cálculo da velocidade e da posição das partículas realizado a cada passo de simulação. Todavia, ela também possui desvantagens, como o aumento no número de

partículas da cena, em especial para obstáculos com geometria complexa. Além disso, o uso de partículas fantasma pode causar efeitos numéricos indesejados em simulações de fluidos bifásicos. Isto acontece pois, neste caso, é possível que uma mesma partícula fantasma seja utilizada para completar os suportes de partículas de fases diferentes do fluido, afetando-as de maneiras distintas visto que propriedades físicas não são iguais.

Este trabalho apresenta uma estratégia para a simulação de escoamentos de fluidos newtonianos bifásicos incompressíveis através de meios porosos utilizando o método SPH. Nela, o domínio de simulação é representado através de um *grid* regular cujas células representam tanto regiões sólidas, por onde não há o escoamento de fluido, quanto poros. Ainda, tendo em vista as limitações do uso de partículas fantasma citadas no parágrafo anterior, utiliza-se uma abordagem para o tratamento de condições de contorno baseada no cálculo de colisões geométricas entre as partículas e as regiões sólidas do domínio.

1.3 Organização do texto

No próximo capítulo serão apresentados trabalhos relevantes da literatura estudada e suas contribuições para a realização do trabalho proposto. No terceiro capítulo será descrito o método SPH. Em seguida, o capítulo quatro apresentará considerações importantes sobre as etapas vistas como críticas na implementação proposta, que, por sua vez, será mais detalhada no quinto capítulo desta dissertação. O capítulo seis, então, apresentará resultados de testes realizados utilizando a ferramenta implementada, seguido, finalmente, pelo sétimo capítulo, que concluirá o trabalho fazendo um apanhado de suas contribuições à literatura e apresentando desafios e possíveis trabalhos futuros.

Capítulo 2

Trabalhos relacionados

O escoamento de fluidos newtonianos, estudado amplamente na área da Engenharia Mecânica, pode ser descrito através de um conjunto de equações diferenciais, denominadas equações de Navier-Stokes. Baseadas em leis de conservação e na segunda lei de Newton, elas determinam que mudanças na aceleração de um fluido são consequências de alterações em sua densidade e pressão. Assim, analisando os gradientes destas quantidades no fluido, é possível obter um campo de velocidades como solução para tais equações, descrevendo o movimento do fluido em todos os pontos do espaço.

Segundo Bridson [1], as técnicas computacionais que utilizam estas equações para realizar a simulação do escoamento de fluidos podem ser divididas em duas grandes categorias. As técnicas chamadas eulerianas baseiam-se em um *grid* no espaço, em cujas posições são avaliadas as grandezas físicas do fluido que as cruzam em um dado instante no tempo. Já as abordagens lagrangeanas tratam o fluido como um conjunto finito de porções fluidas que movimentam-se no espaço e avaliam as grandezas de cada uma delas a cada passo da simulação. Assim, pode-se dizer que a principal diferença entre estas categorias de técnicas é que, enquanto as posições avaliadas permanecem fixas ao longo das simulações eulerianas, elas variam nas lagrangeanas.

2.1 Simulação de fluidos monofásicos usando SPH

O trabalho de Gingold e Monaghan [8] foi o primeiro a apresentar o método SPH, uma abordagem lagrangeana para a simulação computacional de fluidos. Simultaneamente, Lucy [13] discutiu uma técnica semelhante em um trabalho publicado no mesmo ano. Estes estudos foram motivados pela necessidade da área de Astrofísica de realizar modelagens simples e robustas de estrelas assimétricas utilizando uma pequena quantidade de pontos.

Viu-se que, enquanto uma estrela simétrica necessitava de cerca de 20 pontos ao longo de um eixo para ser modelada satisfatoriamente utilizando soluções para as equações de Navier-Stokes obtidas através diferenças finitas, uma estrela assimétrica exigia um número da ordem de 20^3 pontos para que fosse atingido o mesmo nível de precisão. Este fato tornava o uso destas técnicas inviável devido ao grande número de integrais complexas que precisavam ser calculadas.

A criação do método SPH por Gingold e Monaghan teve como base um trabalho anterior de Pasta e Ulam [19], no qual foi proposta uma técnica heurística para a simulação de escoamentos que se baseava na discretização do fluido, representando-o através de um conjunto finito de partículas ao invés de utilizar a representação contínua tradicional dos métodos eulerianos. A partir dessa premissa, houve uma análise das formas de se determinar as forças que agiam em uma posição arbitrária do fluido. Foram estudadas abordagens estatísticas para a obtenção de uma distribuição probabilística das grandezas do fluido a partir dos valores observados nos elementos discretos que o representavam, dentre as quais destacaram-se duas, baseadas em aproximações de integrais obtidas através do método estatístico de Monte Carlo. Uma destas aproximações faz uso das chamadas funções núcleo de suavização, que tornaram-se parte fundamental da formulação do método SPH ao longo dos anos. Esse mesmo trabalho também estudou o uso destas funções núcleo de suavização para obter distribuições de grandezas envolvidas nas equações de Navier-Stokes. Porém, por esse trabalho ser da área da Astrofísica, que lida com fluidos não-viscosos, a contribuição da viscosidade no campo de velocidades do fluido foi desconsiderada. Além disso, foram levadas em conta contribuições de forças adicionais importantes em aplicações de escala cósmica, como a atração gravitacional exercida pelos corpos celestes simulados e a ação de campos eletromagnéticos sobre eles.

Anos após o desenvolvimento do SPH, o trabalho de Desbrun e Cani [5] estudou as adaptações necessárias para adequar o método tradicional, utilizado para a simulação de fluidos cosmológicos, de forma que ele também pudesse ser empregado na área de Computação Gráfica para modelar corpos inelásticos com diferentes níveis de rigidez e viscosidade. Como, em simulações de fluidos incompressíveis, a densidade do fluido no repouso é constante, foi vista a necessidade de manter um certo grau de coesão de sua estrutura interna de forma a garantir esta característica. Isto requeria o balanceamento de forças de pressão tanto atrativas quanto repulsivas, o que não ocorria na modelagem clássica de Monaghan, tendo em vista que, em aplicações astrofísicas, as forças resultantes da pressão eram estritamente repulsivas, sendo equilibradas pela atração gravitacional do corpo celeste simulado. Para resolver este problema, propô-se utilizar uma versão

modificada da equação de estado dos gases, dada por

$$p = k(\rho - \rho_0), \tag{2.1}$$

que será discutida em mais detalhes no capítulo a seguir. Além disso, foi analisada também a função núcleo de suavização *splice* Gaussiana, tradicionalmente utilizada nos cálculos do SPH, descrita na Equação 2.1 abaixo:

$$W_h(\mathbf{r}) = \frac{1}{\pi h} \begin{cases} 1 - \frac{3}{2}\left(\frac{r}{h}\right)^2 + \frac{3}{4}\left(\frac{r}{h}\right)^3, & 0 \leqslant r \leqslant h \\ \frac{1}{4}\left(2 - \frac{r}{h}\right)^3, & h \leqslant r \leqslant 2h \\ 0, & \text{caso contrário,} \end{cases} \tag{2.2}$$

onde r é a distância entre um par de partículas e h é o núcleo de suavização, que determina o tamanho da vizinhança das partículas a ser considerada. Verificou-se que esta função era inadequada para o uso em simulações no campo da Computação Visual devido ao fato de gerar aglomerações de partículas. Isto era causado pelos termos envolvendo potências quadradas e cúbicas da razão $\frac{r}{h}$, proporcional à distância entre as partículas, o que fazia com que os valores avaliados diminuíssem rapidamente quanto mais próximas estivessem as partículas, acarretando em forças de repulsão de baixa magnitude nestes casos. Este efeito era desejado em aplicações astrofísicas por representar o processo de nascimento de estrelas através da aglomeração de nuvens de gás. Todavia, ele se mostrava indesejado para a simulação de fluidos incompressíveis, pois, por sua densidade e pela magnitude da força da gravidade exercida pela Terra sobre eles serem bastante inferiores em relação às grandezas envolvidas na simulação de corpos celestes, esta tendência de aglomeração das partículas não era observada.

O trabalho de Desbrun e Cani [5] concluiu que a utilização do método SPH para a modelagem de fluidos em Computação Gráfica era uma boa estratégia, visto que ele é computacionalmente semelhante a métodos anteriores baseados na discretização do fluido [6, 7, 10, 11, 12, 14, 20, 22, 23], além de utilizar equações de movimento similares aos métodos convencionais baseados em forças simétricas entre pares de partículas. Além disso, foram observadas outras vantagens do método, como uma representação implícita da superfície, coerente com o modelo físico, derivada da distribuição espacial das densidades no fluido, além da possibilidade de tratar a integração numérica de maneira robusta, visto que cada partícula utiliza um passo de tempo individual que pode ser alterado para garantir a estabilidade, e um conjunto de parâmetros intuitivos, como densidade e viscosidade, que facilitam a definição pelo usuário do material que deseja-se simular. Por

fim, através da simulação de Monte Carlo, verificou-se nesse trabalho que a estabilidade numérica do método é proporcional ao número de partículas usadas na simulação.

O trabalho de Müller, Charypar e Gross [18], usado como ponto de partida para esta dissertação, fez novas contribuições para a área de simulação de fluidos em Computação Gráfica utilizando SPH. Primeiramente, foram feitas alterações nos cálculos das forças de pressão e viscosidade para garantir sua simetria. Também foram introduzidas duas novas funções de suavização, totalizando, junto da proposta por Desbrun e Cani, três funções núcleo de suavização diferentes, cada uma delas empregada no cálculo de um dos termos da forma paramétrica das equações de Navier-Stokes usada pelo método. Além disso, foi apresentada uma estratégia para o rastreamento e a visualização da superfície livre utilizando uma representação baseada em uma malha de triângulos. Também foi proposta uma estratégia para aumentar ainda mais a eficiência da simulação ao armazenar cópias dos objetos das partículas nas células do *grid* ao invés de referenciar as estruturas, o que causa uma maior proximidade em memória das informações necessárias para os cálculos de interpolação, aumentando dramaticamente a taxa de acerto em *cache*. Esta técnica tem como desvantagem o aumento do gasto de memória, que passa a ser duas vezes maior. Devido a limitações do *hardware* utilizado neste trabalho, optou-se por não utilizar esta estratégia, mas seu uso é de grande utilidade caso haja disponibilidade de recursos para tal.

2.2 Simulação de fluidos bifásicos usando SPH

Uma primeira formulação da utilização do SPH para a simulação de fluidos bifásicos foi apresentada por Monaghan [16], motivada pela necessidade de simular numericamente fluxos piroclásticos, correntes de cinzas, poeira e gás aquecido causadas por erupções vulcânicas, fenômenos estes que, quando ocorridos em alto mar, podem acarretar na formação de tsunamis.

A escolha do SPH ao invés de técnicas tradicionais utilizando elementos finitos foi motivada por uma maior facilidade de lidar com superfícies livres e sistemas nos quais vários fluidos interagem entre si. Segundo o autor, a formulação original, pensada para a descrição do movimento de um fluido monofásico, pode ser facilmente adaptada para o caso bifásico. A única alteração, realizada na etapa de computação das forças, foi a mudança da equação de estado para uma capaz de garantir a condição de incompressibilidade do fluido, minimizando as flutuações de densidade em seu interior, visto que esta

grandeza é crucial para o estabelecimento das fases do fluido.

A comparação dos resultados numéricos obtidos neste trabalho com dados experimentais mostrou que o SPH é uma técnica adequada à simulação de fluidos bifásicos. Porém, notou-se que ela é estável para casos nos quais a diferença entre as densidades das duas fases do fluido é da ordem de até 50% [9], apresentando instabilidades para razões maiores, ou seja, simulações nas quais a diferença entre as densidades é grande. Estas conclusões foram reiteradas em trabalhos futuros de Monaghan *et al* [17].

Outros trabalhos disponíveis na literatura propõem adaptações do método SPH para casos específicos nos quais a utilização de sua formulação tradicional não gera bons resultados. Dentre eles, pode-se citar o trabalho de Landrini, Colagrossi e Tulin [9], que propôs alterações nos cálculos dos operadores de divergência e gradiente, além de uma mudança na computação da velocidade das partículas. Estes ajustes permitem a utilização desta técnica para a simulação de fluidos bifásicos nos quais a diferença entre as densidades das substâncias que compõem cada fase são grandes.

2.3 Escoamento de fluidos em meios porosos

Em 1990, um trabalho de Cassel, Brown e Johnson [3] apresentou uma abordagem para uso da tecnologia de tomografia computadorizada, utilizada amplamente na área médica, visando a obtenção de um modelo 3D que explicitasse a estrutura interna de materiais porosos. Este estudo abriu caminho para pesquisas sobre a simulação do escoamento de fluidos em meios porosos, que possuem aplicações nos campos da Geologia [4, 24] e Medicina [2, 15], dentre outros.

O trabalho de Zhu *et al* [25] apresentou uma primeira análise da utilização do SPH para a simulação de escoamentos em meios porosos. Este estudo foi motivado por características deste método, como a modelagem sem malhas e a adaptabilidade a superfícies livres, que são interessantes neste contexto. Através de comparações de simulações em duas dimensões do escoamento de um fluido por um meio poroso artificial geradas utilizando a técnica tradicional de elementos finitos e o SPH, observou-se que ambas ofereceram resultados robustos, havendo, porém, um ganho de eficiência na versão utilizando SPH.

Uma implementação para o escoamento de fluidos em meios porosos foi sugerida por Strozzi *et al.* Foram utilizados modelos de estruturas internas de rochas porosas obtidos por meio de tomografia computadorizada como os meios através dos quais o fluido,

modelado utilizando a formulação tradicional do SPH, escoa. Para otimizar o processo de busca por vizinhos de partículas na etapa de computação de forças do SPH, foi usada uma *octree*.

Este trabalho tem como diferencial em relação às outras contribuições na área de escoamento de fluidos em meios porosos através do método SPH a utilização de uma técnica de colisão geométrica para tratar a condição de contorno, substituindo a abordagem de partículas fantasma empregada na literatura estudada. O uso desta técnica tem como objetivo eliminar a necessidade de um *setup* das posições iniciais das partículas fantasma, que não é trivial para obstáculos de geometria complexa, e reduzir o número total de partículas envolvidas na simulação, proporcionando um aumento da eficiência. Seu uso também visa minimizar a ocorrência de efeitos indesejados no comportamento da simulação de fluidos bifásicos devido às influências distintas que as partículas fantasma possuem sobre partículas provenientes de fases diferentes do fluido. Além disso, o obstáculo poroso é representado neste trabalho como um *grid* 3D composto por células cúbicas com faces ortogonais aos eixos coordenados, o que simplifica as etapas de detecção e tratamento de colisão geométrica, acarretando num ganho de eficiência.

Capítulo 3

Descrição do método matemático

3.1 SPH monofásico

O SPH baseia-se na representação discreta do fluido através de um conjunto de elementos chamados partículas, como ilustra a Figura 3.1.

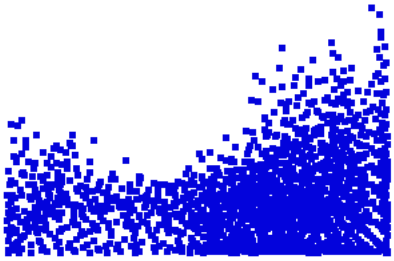


Figura 3.1: Exemplo de simulação monofásica utilizando o método SPH, gerada com o programa desenvolvido durante este trabalho.

O método consiste em calcular a força resultante sobre cada uma das partículas, que posteriormente é integrada numericamente para obter-se suas velocidades. As forças que atuam sobre as partículas são calculadas a partir das equações de Navier-Stokes, que descrevem o movimento de fluidos viscosos. Sua formulação lagrangeana para o escoamento incompressível de fluidos newtonianos é:

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2\mathbf{v} + \mathbf{g}. \tag{3.1}$$

Segundo Monaghan [8], uma grandeza escalar A pode ser interpolada em uma dada posição $\mathbf{r}$ no domínio do fluido através de um somatório ponderado das contribuições de todas as partículas que o constituem, através da equação:

$$A_s(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} W(\mathbf{r} - \mathbf{r}_j, h), \tag{3.2}$$

onde j é iterado sobre todas as partículas do fluido, m_j representa a massa, ρ_j a densidade e A_j o valor da quantidade calculada na posição $\mathbf{r}_j$. O gradiente de A, por sua vez, pode ser calculado usando a equação:

$$\bigtriangledown A_s(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \bigtriangledown W(\mathbf{r} - \mathbf{r}_j, h). \tag{3.3}$$

Ainda, a laplaciana de A é dada por:

$$\bigtriangledown^2 A_s(\mathbf{r}) = \sum_j m_j \frac{A_j}{\rho_j} \bigtriangledown^2 W(\mathbf{r} - \mathbf{r}_j, h). \tag{3.4}$$

Nas equações acima, o componente $W(\mathbf{r} - \mathbf{r}_j, h)$ é a chamada função núcleo de suavização e pondera a contribuição da partícula na posição $\mathbf{r}_j$ para o cálculo da grandeza escalar A em $\mathbf{r}$ em função de um raio de suavização h. A Figura 3.2 ilustra o formato de uma função núcleo de suavização. Observa-se que, quanto menor a distância entre as posições $\mathbf{r}$ e $\mathbf{r}_j$, maior será o valor de $W(\mathbf{r} - \mathbf{r}_j, h)$, indicando uma contribuição maior da partícula j para o cálculo. Já partículas a uma distância superior a h da posição $\mathbf{r}$, quando avaliadas pelo núcleo de suavização, produzem valores nulos, indicando que a partícula na posição $\mathbf{r}_j$ não contribui para o cálculo da grandeza em questão por estar muito distante de $\mathbf{r}$.

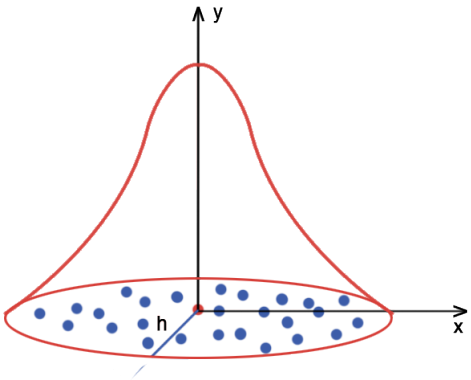


Figura 3.2: Exemplo de função núcleo de suavização.

Funções núcleo de suavização. A literatura de SPH apresenta diversas funções núcleo de suavização, cada uma adequada para o cálculo de diferentes grandezas físicas.

Neste trabalho, são utilizadas:

- A função de suavização $W_d(\mathbf{r} - \mathbf{r}_j, h)$ [18] é usada para o cálculo da densidade das partículas, possuindo a vantagem computacional de apresentar $\mathbf{r}$ apenas elevado ao quadrado, o que elimina a necessidade de computar a raiz quadrada no cálculo da distância, obtida por $r = \|\mathbf{r} - \mathbf{r}_j\|$. Esta função é dada por:

$$W_d(\mathbf{r} - \mathbf{r}_j, h) = \frac{315}{64\pi h^9} \begin{cases} (h^2 - r^2)^3, & 0 \leqslant r \leqslant h \\ 0, & \text{caso contrário.} \end{cases} \tag{3.5}$$

  Uma desvantagem desta função de suavização é que ela é inadequada para o cálculo das pressões. Isto ocorre pois seu gradiente aproxima-se de zero no centro, para vizinhos próximos à partícula de interesse, acarretando em forças de pressão nulas, o que gera aglomerações de partículas.

- A função $W_p(\mathbf{r} - \mathbf{r}_j, h)$, proposta por Desbrun [5], soluciona o problema do uso do núcleo de suavização $W_d(\mathbf{r} - \mathbf{r}_j, h)$ para a avaliação das pressões, visto que os maiores valores de sua derivada são obtidos quando a diferença $(h - r)$ tende a h, ou seja, quando r tende a zero, o que corresponde aos casos em que os vizinhos estão próximos da partícula. Isto faz com que as forças de repulsão geradas pela pressão sejam maiores para partículas próximas, o que está de acordo com o modelo físico. Ela é dada por:

$$W_p(\mathbf{r}, h) = \frac{15}{\pi h^6} \begin{cases} (h - r)^3, & 0 \leqslant r \leqslant h \\ 0, & \text{caso contrário,} \end{cases} \tag{3.6}$$

  com gradiente dado por:

$$\bigtriangledown W_p(\mathbf{r}, h) = -\frac{45}{\pi h^6}(h - r)^2. \tag{3.7}$$

- A função $W_v(\mathbf{r} - \mathbf{r}_j, h)$ [18] foi criada com o objetivo de ser utilizada durante a etapa de cálculo das forças viscosas. Sua laplaciana possui a propriedade de gerar valores estritamente positivos, o que é condizente com o efeito da viscosidade, que reduz a energia cinética do fluido através da geração de calor causado pela fricção entre lâminas fluidas. Esta função é dada por:

$$W_v(\mathbf{r}, h) = \frac{15}{2\pi h^3} \begin{cases} -\frac{r^3}{2h^3} + \frac{r^2}{h^2} + \frac{h}{2r} - 1, & 0 \leqslant r \leqslant h \\ 0, & \text{caso contrário,} \end{cases} \tag{3.8}$$

sendo sua laplaciana dada por:

$$\bigtriangledown^2 W_v(\mathbf{r}, h) = \frac{45}{\pi h^6}(h - r). \tag{3.9}$$

Cálculo das forças. O primeiro passo para o cálculo da força que atua sobre uma partícula é a avaliação do valor de sua densidade. Substituindo a grandeza escalar A por ρ na Equação 3.2, obtemos:

$$\rho_s(\mathbf{r}) = \sum_j m_j \frac{\rho_j}{\rho_j} W_d(\mathbf{r} - \mathbf{r}_j, h) = \sum_j m_j W_d(\mathbf{r} - \mathbf{r}_j, h). \tag{3.10}$$

Utilizando o valor da densidade avaliada através desta equação, é possível calcular a pressão do fluido na posição $\mathbf{r}$ através de uma versão modificada da equação de estado para os gases ideais [5], dada por:

$$p_s = k(\rho_s - \rho_0), \tag{3.11}$$

onde ρ_0 é o valor da densidade do fluido em repouso e k é uma constante relacionada à temperatura. Utilizando os valores calculados da pressão, é possível obter sua contribuição na força resultante que age sobre a partícula i através da avaliação do simétrico de seu gradiente, $-\bigtriangledown p$, visto que as forças agem da região com maior pressão em direção à com menor pressão. Utilizando a Equação 3.3, obtemos:

$$f_i^{pressure} = -\bigtriangledown p(\mathbf{r}_i) = -\sum_j m_j \frac{p_j}{\rho_j} \bigtriangledown W_p(\mathbf{r}_i - \mathbf{r}_j, h). \tag{3.12}$$

Todavia, esta equação não é simétrica. Isto pode ser verificado a partir do caso em que apenas duas partículas interagem, na qual o cálculo da contribuição da pressão na força resultante sobre a partícula i usará apenas o valor da pressão da partícula j e vice-versa. Como, em geral, as pressões observadas em duas posições distintas do fluido não são iguais, as forças não serão simétricas. Este problema pode ser solucionado com uma simples alteração na Equação 3.12 [18], substituindo o termo p_j pela média aritmética das pressões de ambas as partículas envolvidas em cada etapa do somatório, obtendo a equação:

$$f_i^{pressure} = -\sum_j m_j \frac{p_i + p_j}{2\rho_j} \bigtriangledown W_p(\mathbf{r}_i - \mathbf{r}_j, h). \tag{3.13}$$

A contribuição da viscosidade para a força resultante sobre a partícula i pode ser

calculada de forma semelhante, aplicando a Equação 3.4 ao termo de viscosidade $\mu \bigtriangledown^2 \mathbf{v}$:

$$f_i^{viscosity} = \mu \bigtriangledown^2 v(\mathbf{r}_i) = \mu \sum_j m_j \frac{\mathbf{v}_j}{\rho_j} \bigtriangledown^2 W_v(\mathbf{r}_i - \mathbf{r}_j, h). \tag{3.14}$$

Assim como a Equação 3.12, a expressão acima não é simétrica. Porém, como as forças de viscosidade dependem apenas das diferenças de velocidade, é possível simetrizar as forças de forma natural ao substituir o termo $\mathbf{v}_j$ por $\mathbf{v}_j - \mathbf{v}_i$, obtendo, finalmente, a equação:

$$f_i^{viscosity} = \mu \sum_j m_j \frac{\mathbf{v}_j - \mathbf{v}_i}{\rho_j} \bigtriangledown^2 W_v(\mathbf{r}_i - \mathbf{r}_j, h). \tag{3.15}$$

A força resultante que age sobre a partícula i é calculada através da soma das componentes calculadas nas equações 3.13 e 3.15 e da contribuição das forças externas, representadas por $\mathbf{f}_i^{external}$, como expresso abaixo pela Equação 3.16:

$$\mathbf{f}_i = \mathbf{f}_i^{pressure} + \mathbf{f}_i^{viscosity} + \mathbf{f}_i^{external}. \tag{3.16}$$

Utilizando, então, a segunda lei de Newton e levando em consideração que o volume de cada partícula é igual e unitário, temos que $\rho_i = \frac{m_i}{V_i} = \frac{m_i}{1} = m_i$, sendo possível assim obter a aceleração da partícula através da Equação 3.17:

$$\mathbf{a_i} = \frac{\mathbf{f}_i}{\rho_i}. \tag{3.17}$$

Após esta etapa, é possível realizar a integração numérica da aceleração para obter a velocidade e, por fim, a nova posição da partícula.

3.2 SPH bifásico

Conforme descrito por Monaghan [16], não é preciso alterar as equações listadas na seção anterior para adaptar o método SPH para a simulação de um fluido bifásico. É necessário, porém, identificar cada partícula da discretização de forma que seja possível determinar a qual fase de fluido ela pertence e para que, na etapa de cálculo das forças que agem sobre ela, seja possível utilizar as constantes físicas adequadas. A Figura 3.3 ilustra a rotulação das partículas pertencentes às fases de um fluido bifásico.

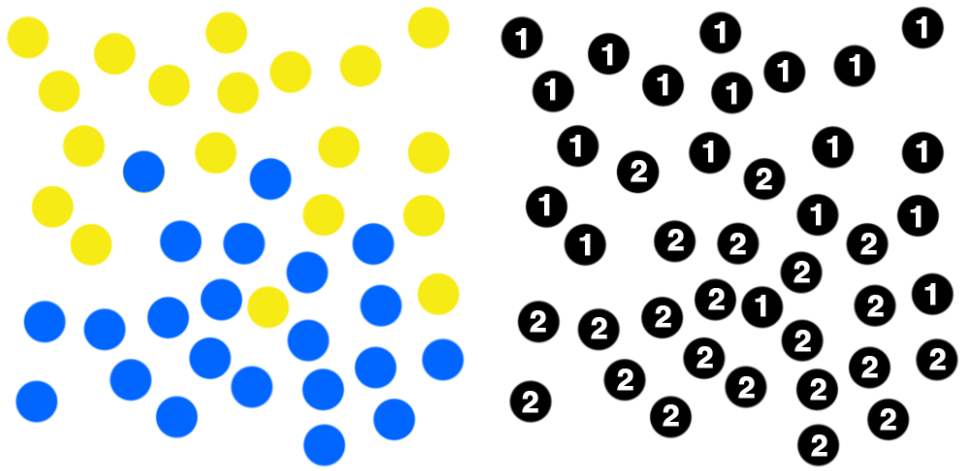


Figura 3.3: Exemplo de atribuição de identificadores a partículas pertencentes a cada uma das fases do fluido.

É importante notar que o uso do SPH para a simulação de fluidos bifásicos é estável apenas para diferenças de densidade da ordem de 50% entre as fases. Desta forma, é necessário avaliar a necessidade de realizar alterações nos cálculos do SPH tradicional, como as propostas por Landrini [9], caso os fluidos simulados possuam densidades muito diferentes no repouso. Levando em consideração que a maioria dos fluidos bifásicos, como os estudados neste trabalho, possui fases com densidades que diferem em menos de 50% entre si, o SPH tradicional possui estabilidade suficiente para simulá-los e, por isso, não foi necessário realizar alterações em seus cálculos.

Considerando que o método realiza seus cálculos individualmente para cada partícula utilizando contribuições ponderadas de suas vizinhas para obter sua nova posição, a separação das fases ocorre automaticamente. Isto se deve à influência da densidade de repouso dos fluidos no cálculo da pressão na Equação 3.11 e dos efeitos de repulsão causados pelo termo $-\frac{1}{\rho}\nabla p$ da Equação 3.1. De fato, o simétrico do gradiente de pressão em torno da interface entre os fluidos aponta o interior da região ocupada por cada fase de fluido, como mostra a Figura 3.4, e, assim, as partículas de mesmo tipo tendem a se manter agrupadas.

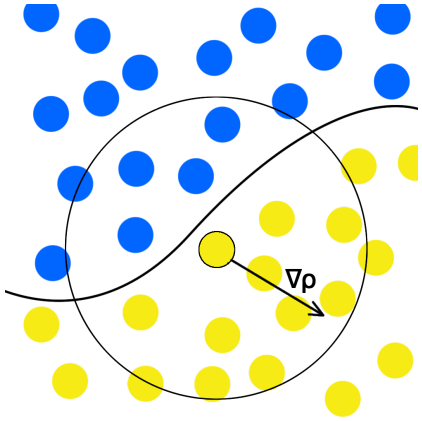


Figura 3.4: Exemplo de direção do vetor gradiente de densidade.

3.3 SPH em meios porosos

O fato de ser uma técnica lagrangeana, isto é, que não utiliza malhas, faz do SPH uma boa estratégia para a simulação de escoamentos em meios porosos, já que descreve naturalmente a superfície do fluido e torna o tratamento de condições de contorno mais simples e eficiente.

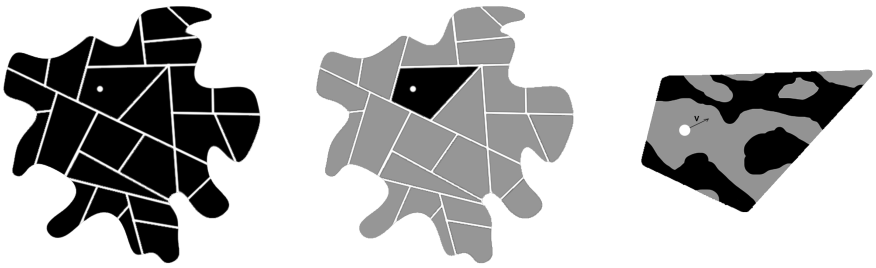
Neste trabalho, o tratamento das condições de contorno é feito através do calculo de colisão entre as partículas de fluido e o modelo geométrico que descreve o domínio de simulação. Esta abordagem é uma alternativa à técnica tradicional para o tratamento de condições de contorno que utiliza partículas fantasma [21], que pode ser computacionalmente custosa quando a geometria do domínio de simulação é complexa. Este fato se deve à necessidade do uso de um grande número de partículas fantasma para representar completamente a fronteira do domínio, além de técnicas sofisticadas que facilitem o posicionamento de tais partículas.

Primeiramente, é realizado um particionamento do espaço ocupado pelo obstáculo poroso, de forma que cada partição armazene informações da geometria da porção do obstáculo nela contida, como, por exemplo, uma lista de triângulos de uma malha poligonal. A Figura 3.5 ilustra este processo.

Durante a simulação do escoamento do fluido através do meio poroso, é possível utilizar a estrutura criada para determinar com quais polígonos da malha que representa o obstáculo serão realizados os testes de detecção de colisão ao verificar em que partição está contida a partícula analisada e considerar apenas os polígonos que também fazem parte dela, como mostra a Figura 3.6. Assim, ao eliminar a necessidade de realizar esta verificação com todos os polígonos da malha, a eficiência do programa aumenta

significativamente.

Figura 3.5: Particionamento arbitrário de um obstáculo poroso.



(a) (b) (c)

Figura 3.6: Exemplo de uso do particionamento do obstáculo para o tratamento da condição de contorno. Utilizando-se a posição da partícula (a), isola-se a partição à qual ela pertence (b) e testa-se a colisão geométrica apenas com as componentes da malha que também pertencem à mesma partição (c).

O tratamento da condição de contorno através da colisão geométrica é feito através da análise da interseção entre o vetor velocidade da partícula e cada polígono da malha contido na mesma partição do espaço que ela. Considerando $\mathbf{r}_0$ e $\mathbf{v}_0$ a posição e a velocidade da partícula, respectivamente, $\mathbf{n}$ a normal do polígono e $\mathbf{a}$ a posição de um de seus vértices, como ilustrado na Figura 3.7, pode-se calcular uma constante d através da Equação 3.18 abaixo:

$$d = \frac{(\mathbf{a} - \mathbf{r}_0) \cdot \mathbf{n}}{\mathbf{v}_0 \cdot \mathbf{n}}. \tag{3.18}$$

Um valor de d positivo indica que não houve colisão entre a partícula e o vértice. Porém, caso $d \leq 0$, utiliza-se seu valor para encontrar a interseção do vetor velocidade com o plano, que será a posição corrigida $\mathbf{r}$ da partícula, através da equação:

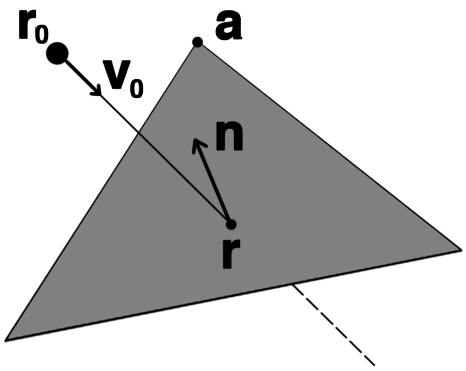


Figura 3.7: Esquema da colisão geométrica de uma partícula com um polígono da malha do obstáculo.

$$\mathbf{r} = d\mathbf{v} + \mathbf{r}_0. \tag{3.19}$$

É necessário também refletir o vetor v em relação à normal n de forma a obter a nova velocidade da partícula após a colisão. Também mutiplica-se o vetor por uma constante β tal que $0 < \beta < 1$ de forma a representar a perda de energia cinética causada pela colisão. Assim, tem-se a nova velocidade $\mathbf{v}$ dada por:

$$\mathbf{v} = \beta[-2\ (\mathbf{v}_0 \cdot \mathbf{n})\mathbf{n} + \mathbf{v}] \tag{3.20}$$

e mostrada na Figura 3.8.

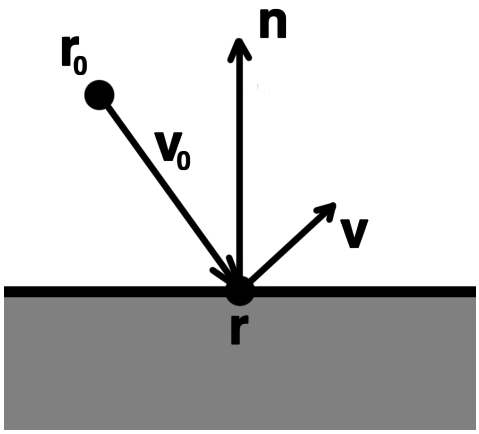


Figura 3.8: Reflexão e amortecimento da velocidade causado pela colisão.

Esta estratégia permite que as etapas de avaliação o tratamento de colisões sejam

inseridas logo após o cálculo de sua nova posição. Em comparação com as técnicas mais comuns empregadas em conjunto com o SPH, que utilizam partículas fantasma imóveis para representar os limites dos obstáculos da cena, o uso da técnica de colisão geométrica em conjunto a um particionamento do espaço mostra-se mais eficiente, em especial para obstáculos porosos de geometria complexa. Isto se deve ao fato desta técnica não requerer a inserção de mais partículas na cena, o que poderia gerar um efeito negativo na eficiência da simulação. Além disso, por não necessitar da utilização de partículas fantasma, que não representam nenhum fluido, como auxiliares para o tratamento da condição de contorno, a abordagem geométrica mostra-se mais adequada para escoamentos bifásicos, visto que a presença de terceiro tipo de partícula poderia influenciar os cálculos das interações entre as duas fases do sistema simulado devido às diferenças distintas entre as propriedades das partículas.

Capítulo 4

Considerações computacionais

Este capítulo tem como objetivo descrever computacionalmente o método SPH aplicado à simulação de fluidos em meios porosos, fazendo considerações mais detalhadas sobre algumas etapas importantes deste processo.

4.1 *Loop* principal

O Algoritmo 4.1 abaixo mostra as etapas da simulação de fluidos utilizando o método SPH.

Algoritmo 1 *Loop* principal

```
1:  Inicialize o fluido
2:  Inicialize o obstáculo poroso
3:  Para cada passo da simulação faça:
4:      Para cada partícula p do fluido faça:
5:          Obtenha a vizinhança de p
6:          Calcule a densidade de p
7:          Calcule a pressão de p
8:          Calcule a viscosidade de p
9:          Calcule as forças sobre p
10:         Calcule a aceleração de p
11:         Calcule a velocidade de p
12:         Calcule a posição de p
13:         Faça o tratamento da condição de contorno
14:         Atualize a vizinhança de p
15:         Renderize a cena
```

Durante a elaboração deste trabalho, observou-se que as etapas nas linhas $1, 2, 5, 11, 12, 13$ e 14 merecem atenção devido à existência de detalhes importantes a serem levados em consideração de forma a realizar sua implementação de forma eficiente e robusta. Além

disso, a etapa de tratamento de condições de contorno deve ser adaptada para se adequar aos meios porosos tratados neste trabalho, utilizando técnicas de colisão geométrica para tal, conforme proposto. Devido a sua importância, essas etapas serão tratadas em mais detalhes nas seções a seguir.

4.2 Inicialização do fluido

A etapa de inicialização do fluido consiste em determinar o número de partículas que devem ser utilizadas para representar o fluido simulado, assim como as suas posições no espaço da cena. A distribuição das partículas no volume de fluido é de grande importância para o método SPH, visto que ela determinará o número de vizinhos que cada partícula possui nas iterações iniciais, quantidade esta utilizada nas etapas de cálculo das grandeza físicas do escoamento nas linhas 6 a 9 do algoritmo do *loop* principal. Tendo em vista que o método SPH aproxima as grandezas de uma partícula através de contribuições ponderadas dos valores de seus vizinhos, o grau de esparsidade da vizinhança está relacionado com a precisão do resultado da simulação. Experiências na literatura sobre o método SPH indicam que simulações de escoamentos tridimensionais apresentam resultados ideias quando o número médio de vizinhos das partículas é próximo de 27. Por isto, busca-se realizar a distribuição das partículas no volume de maneira a obter vizinhanças próximas a este valor durante a simulação.

A realização da distribuição inicial de partículas de forma que o número de vizinhos médio seja próximo de 27 no repouso não garante, porém, bons resultados para a simulação. Isto se deve ao fato de que, de acordo com as propriedades físicas iniciais das partículas, elas poderão se reajustar nos primeiros passos da simulação de forma que as distâncias entre elas adequem-se às grandezas calculadas, acarretando numa mudança do número de vizinhos médio. Desta forma, é interessante determinar o número de partículas e o raio de busca, dados de entrada da simulação, de tal maneira que o tamanho da vizinhança média atinja valores próximos ao valor médio desejado de 27 vizinhos no decorrer de toda a simulação.

Outra etapa importante realizada durante a fase de inicialização de fluidos é o cálculo das densidades iniciais das partículas. Esta grandeza está relacionada à quantidade de partículas que representa o fluido e à sua distribuição no espaço, de forma que, quanto mais partículas houver, menor será a densidade inicial de cada uma delas. Visto que tais informações são sabidas apenas ao final da etapa de inicialização do fluido, é necessário

calculá-las antes de prosseguir com a simulação. Para obter estes valores, é realizado o mesmo cálculo feito na linha 6 do *loop* principal para todas as partículas do fluido ao final da distribuição de partículas na linha 1, atribuindo o valor obtido à densidade de repouso das mesmas.

4.3 Busca

Pode-se dizer que o maior gargalo computacional do SPH é a busca pelos vizinhos de uma dada partícula, visto que é necessário obtê-los a cada passo da simulação. Além disso, o nível de realismo da simulação é diretamente influenciado pelo número de partículas utilizadas, o que torna a busca pela vizinhança de cada uma delas uma tarefa ainda mais árdua. A complexidade desta etapa também está diretamente associada ao número de vizinhos por partícula, como abordado na seção anterior. Uma maneira ingênua de realizar este teste é calcular a distância r entre uma partícula i e cada uma das outras partículas j do fluido a cada iteração, para então comparar o valor encontrado com o raio de busca h utilizado e, caso $r \leq h$, incluir j na lista de vizinhos de i. Todavia, esta estratégia torna-se rapidamente ineficiente com o aumento do número de partículas do sistema.

Para otimizar este processo, é necessário utilizar técnicas e estruturas que diminuam a quantidade de partículas a serem verificadas, visto que não é possível reduzir a frequência com a qual esta verificação é feita. A estratégia escolhida neste trabalho foi a utilização de um *grid* tridimensional de busca.

Inicialmente, é estabelecido um domínio da cena grande o suficiente para englobar todo o espaço que o fluido ocupará durante o decorrer de toda a simulação. Em seguida, este espaço é subdividido em células cúbicas, cuja aresta possui tamanho igual ao raio de busca h da simulação, dado como entrada para o programa. Esta escolha se deve ao fato de que, desta forma, é possível garantir que todos os vizinhos de uma partícula estarão nas células imediatamente adjacentes àquela à qual ele pertence. A Figura 4.1 abaixo ilustra este fato através de uma simplificação bidimensional do *grid*. Observa-se que mesmo uma partícula posicionada no limite de uma célula tem vizinhos apenas no interior daquelas imediatamente adjacentes devido à escolha do valor do raio de busca para as arestas.

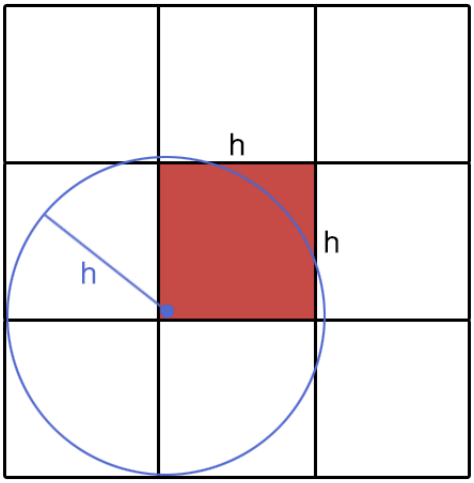


Figura 4.1: Representação bidimensional do *grid* de busca.

A busca por vizinhos de uma partícula α em um *grid* tridimensional pode ser então simplificada pelo seguinte algoritmo:

Algoritmo 2 Busca por vizinhos

1: $(l, m, n) \leftarrow$ célula da posição atual da partícula p
2: **para** $i \leftarrow l - 1$ **até** $l + 1$ **faça**
3: **para** $j \leftarrow m - 1$ **até** $m + 1$ **faça**
4: **para** $k \leftarrow n - 1$ **até** $n + 1$ **faça**
5: **se** a célula (i, j, k) existe **então**
6: adicione as partículas de (i, j, k) aos vizinhos de p
7: **fim se**
8: **fim para**
9: **fim para**
10: **fim para**

Observa-se que, nas linhas 2 a 4, o aninhamento de estruturas de repetição permite varrer todas as células cujos índices diferem em no máximo uma unidade em relação á célula (l, m, n), iterando assim por todas as células vizinhas à analisada. Considerando que pode-se garantir que todos os vizinhos de uma partícula p estarão em células vizinhas à célula onde p está contida, o Algoritmo 4.2 permite obter uma lista reduzida de vizinhos em comparação com o número total de partículas da cena, gerando um ganho de eficiência.

A atualização do *grid* é realizada após obtenção da nova posição de cada partícula ao final de cada passo da simulação, na linha 13 do Algoritmo 4.1 do *loop* principal. Isto pode ser feito ao comparar a célula à qual pertence a posição antiga da partícula p, calculada no passo anterior da simulação, com aquela onde encontra-se sua nova posição, recém-obtida na linha 13 do *loop* principal. Caso ambas sejam iguais, o deslocamento da partícula

ocorreu dentro da célula e, por isso, não é necessário realizar nenhuma alteração. Caso contrário, é feita a troca no *grid*. Este processo pode ser resumido pelo pseudo-código descrito no Algoritmo 4.3:

Algoritmo 3 Atualização do grid

1: $(i, j, k) \leftarrow$ célula da posição atual da partícula p
2: $(l, m, n) \leftarrow$ célula da nova posição da partícula p
3: **se** $(i, j, k) \neq (l, m, n)$ **então**
4: remova $p$de (i, j, k)
5: insira $p$em (l, m, n)
6: **fim se**

É importante realizar a alteração apenas para partículas que mudaram de célula devido a seu deslocamento, pois, enquanto a obtenção das coordenadas da célula na qual reside uma dada posição, feita nas linhas 1 e 2 do Algoritmo 4.3, possui complexidade $O(1)$, a atualização possui complexidade $O(n_{i,j,k})$, onde $n_{i,j,k}$ equivale ao número de partículas no interior de uma célula (i, j, k) do *grid* de busca. Isto se deve ao fato da etapa de remoção da partícula p da célula (i, j, k), feita na linha 4 do algoritmo acima, envolver uma varredura da lista de partículas nela contidas. Desta forma, uma atualização ingênua da localização no *grid* de busca de todas as partículas após o cálculo de sua nova posição acarretaria em uma grande perda de eficiência, sendo agravada ainda mais em simulações envolvendo uma grande quantidade de partículas.

4.4 Integração numérica

A partir da equação de Navier-Stokes, conseguimos calcular o campo de velocidades do fluido, utilizando-o para obter as novas posições das partículas através de técnicas de integração numérica. A literatura no campo de Métodos Numéricos apresenta diversas técnicas, dentre as quais foi escolhida para este trabalho a integração *leap frog*. Esta decisão foi feita com base no fato de que esta técnica é numericamente mais robusta que a integração de Euler, um método de primeira ordem, apesar de requerer o mesmo número de avaliações de funções a cada passo. Além disso, sua solução não introduz um grande *overhead* na simulação, diferente de outras opções com maior precisão, como o método de Runge-Kutta, por exemplo. Por fim, o *leap frog* é classificado como sendo de natureza simpléctica, o que significa que sua utilização conserva a energia de sistemas dinâmicos, como o modelado nesta aplicação, mantendo-o assim mais coeso em seu decorrer.

Dadas as propriedades de uma partícula p em um determinado passo i e um intervalo

de tempo Δt decorrido, pode-se calcular suas propriedades em $i+1$ pelo método *leap frog* através das seguintes equações:

$$v_{i+1} = v_i + \frac{1}{2}(a_i + a_{i+1})\Delta t \tag{4.1}$$

$$r_{i+1} = r_i + r_i \Delta t + \frac{1}{2} a_i \Delta t^2 \tag{4.2}$$

É possível observar que a obtenção dos valores de r_{i+1} e v_{i+1} não possui grandes desafios computacionais, tendo em vista que dependem da avaliação de equações simples realizada em tempo constante. Além disso, como o método SPH já necessita das informações do estado do fluido num passo i de forma a calcular seu estado em $i+1$, as grandezas r_i, v_i e a_i usadas nas equações podem ser facilmente obtidas sem a necessidade de um aumento adicional no consumo de memória de forma a armazená-las.

4.5 Detecção e tratamento de condições de contorno

Na simulação do escoamento de fluidos através de meios porosos, a estratégia adotada para detectar e tratar as condições de contorno da simulação é crucial para a obtenção de resultados satisfatórios. Uma das técnicas mais utilizadas na literatura emprega partículas fantasma, cujas posições são fixas durante toda a simulação e escolhidas de forma que preencham o suporte das partículas.

As características desta abordagem, porém, a tornam pouco ideal para a utilização para obstáculos porosos. Isto se deve ao fato de, nestas condições, serem necessárias muitas partículas fantasma para representar todos os poros do meio pelo qual o fluido escoa, o que aumenta o número de vizinhos a serem tratados durante o passo do SPH, podendo gerar uma diminuição sensível na eficiência da simulação. Além disso, a atribuição das posições das partículas fantasma não é uma tarefa trivial, sobretudo nos casos em que o obstáculo possui geometria complexa, como os considerados neste trabalho. Este algoritmo de distribuição de partículas fantasma, por ser custoso, pode acarretar em uma perda de eficiência da etapa de *setup* da simulação.

Buscando minimizar o problema descrito, este trabalho introduz uma técnica diferente para as colisões partícula-obstáculo, tratando-as através de uma abordagem geométrica. Isto é facilitado pela simplificação do obstáculo poroso, que é representado como um conjunto de elementos cúbicos em estrutura de *grid* criado de forma bastante similar

ao utilizado na etapa de busca, que indica se uma dada célula está preenchida ou vazia através de um dado booleano.

Capítulo 5

Implementação

Este capítulo tem como objetivo apresentar decisões de implementação relevantes para o trabalho realizado.

5.1 Estrutura de classes

Um diagrama de classes do aplicativo criado para este trabalho é apresentado no Apêndice A, acompanhado de uma tabela que descreve as funções de cada uma delas. Ele contempla as classes criadas para a implementação da simulação e das estruturas nela utilizadas.

As principais classes da simulação são *Scene*, *Particle*, *FluidSystem* e *Obstacle*, que representam, respectivamente, a cena, uma partícula, um fluido composto por um conjunto de partículas e um obstáculo.

5.2 Arquivos de configuração

A definição dos parâmetros do programa é feita a partir de arquivos de texto com extensão *.settings*. Cada um deles é tratado por uma classe de configuração. O arquivo é composto por linhas contendo a atribuição de um parâmetro, seguindo o formato:

$$chave = valor. \tag{5.1}$$

Comentários podem ser inseridos no arquivo e são precedidos do caracter #. O endereço em disco do arquivo deve ser passado como parâmetro para o construtor da classe de configuração correspondente. Após este processo, o valor do parâmetro poderá ser obtido

através da chamada de função

$$valor = getProperty(chave), \tag{5.2}$$

que retorna um *String*.

Os principais arquivos de configurações utilizados na simulação e seus parâmetros mais importantes são:

- *SceneSettings*: armazena valores de propriedades utilizadas em cálculos que referem-se à toda a cena, como, por exemplo, o raio de busca h, o intervalo de tempo Δt da integração numérica, o amortecimento β aplicado à velocidade após a colisão, a aceleração da gravidade g e o percentual de porosidade do obstáculo.

- *FirstFluidSettings e SecondFluidSettings*: armazena as propriedades de cada uma das fases do fluido simulado. Ambos os arquivos possuem parâmetros como a massa m, viscosidade μ e constante de gás k, que são utilizados nas etapas de avaliação da densidade, pressão e viscosidade das partículas que compõem o fluido. Além disso, eles armazenam também dois vetores que determinam as dimensões do volume de fluido, que são considerados paralelepípedos definidos através de um ponto mínimo e máximo, e um terceiro que indica o espaçamento entre as partículas uniformemente distribuídas que compõem a fase de fluido. É importante que haja arquivos distintos de forma a encapsular as propriedades de cada fase do fluido, permitindo que os cálculos das grandezas de uma partícula utilizem os parâmetros adequados, referentes à fase de qual ela e suas vizinhas são provenientes.

5.3 Estrutura da implementação

O sistema de processamento de malhas poligonais utilizado como suporte deste trabalho funciona através da leitura de arquivos de configuração. Ao iniciá-lo, é apresentada uma interface que permite escolher o arquivo desejado, neste caso *SPHAppSettings.settings*, e, pressionando a barra de espaço, o método *create()* da classe *SPHApp* é executado, dando início à simulação.

O programa criado para este trabalho pode ser resumido em duas etapas: na primeira é feita a inicialização das estruturas, seguida do *loop* principal da simulação, descrito no Algoritmo 4.1. A implementação destas etapas será detalhada nas seções a seguir.

5.4 Inicialização dos elementos da simulação

Cena Um objeto do tipo *Scene*, que conterá todas as informações sobre a simulação realizada e será responsável pelos cálculos envolvidos na mesma, é construído no momento da criação dao objeto da classe *SPHApp*. Sua inicialização é responsável pela construção de objetos do tipo *SceneGrid*, *ParticleSystem* e *Obstacle*, além de uma malha poligonal que representa o tanque no qual a cena está contida. Além disso, o construtor da cena também é responsável por atribuir valores muito utilizados nas etapas de cálculo, como as porções constantes das funções núcleo de suavização, a variáveis auxiliares, como estratégia para aumentar a eficiência do programa. Também é feita a alocação inicial das partículas criadas na construção do fluido ao *grid* de busca, além do cálculo de suas densidades iniciais e a atribuição de seus respectivos objetos *ParticleInfo* contendo suas propriedades particulares, como aceleração e velocidade, por exemplo. Por fim, nesta etapa também são criadas estruturas auxiliares importantes para a simulação, como um vetor que armazenará os índices das partículas que mudaram de célula no *grid* de busca de forma a facilitar a atualização desta estrutura ao fim do passo da simulação, assim como os arquivos que guardarão as informações utilizadas para cálculos de estatísticas da mesma.

Tanque O paralelepípedo que contém a cena é modelado como uma malha poligonal simples. A geometria do tanque é lida a partir de arquivos de extensão *.off*. Ele funciona como uma barreira, criando paredes invisíveis que impedem que o fluido escoe para além dos limites do domínio de busca, o que causaria um erro ao tentar obter a célula do *grid* de busca na qual uma partícula fora de seu domínio está contida. Para que o tanque seja capaz de exercer esta função, porém, é preciso garantir que ele esteja completamente contido no domínio de busca. Para isto, é executada uma rotina na etapa de inicialização do tanque que lê as informações contidas no arquivo *.off* de sua geometria para verificar se as posições de seus vértices e faces estão dentro dos limites da cena armazenados no arquivo *SceneSettings*, prosseguindo com a simulação em caso verdadeiro.

***Grid* de busca** O objeto do tipo *SceneGrid* criado de forma a otimizar a busca por vizinhos, considerada o principal gargalo do SPH, é representado por um vetor tridimensional de *arrays* de inteiros, de forma que cada posição $G_{i,j,k}$ represente uma célula do *grid* através da lista dos índices das partículas nela contidas. A inicialização do *grid* é descrita pelo algoritmo 4.

Algoritmo 4 Inicialização do *grid* de busca

1: **min** $\leftarrow$ coordenadas do ponto mínimo do domínio da cena
2: **max** $\leftarrow$ coordenadas do ponto máximo do domínio da cena
3: $h \leftarrow$ raio de busca
4: **dim** $\leftarrow$ **max** $-$ **min**
5: **cells** $\leftarrow \lceil \frac{\mathbf{dim}}{h} \rceil$
6: **para** $i \leftarrow$ *0* **até** $cells_x$ **faça**
7: **para** $j \leftarrow$ *0* **até** $cells_y$ **faça**
8: **para** $k \leftarrow$ *0* **até** $cells_z$ **faça**
9: $G_{i,j,k} \leftarrow$ *array vazio de inteiros*
10: **fim para**
11: **fim para**
12: **fim para**

Neste algoritmo, obtém-se os pontos mínimo e máximo do domínio da cena e o raio de busca, lidos do arquivo de configuração da cena. Através de uma subtração vetorial, realizada na linha 4, as dimensões da cena nas três coordenadas cartesianas são calculadas e utilizadas na linha seguinte para calcular o número de células do *grid* tridimensional de busca por uma divisão inteira do vetor **dim** pelo raio de busca, visto que este é o tamanho utilizado para o lado das células do *grid*, como discutido no capítulo anterior. É importante observar que utiliza-se o teto da divisão para evitar que o número de células e, consequentemente, o domínio de cena por elas representado seja menor do que o necessário para comportar o fluido e o obstáculo simulados. Finalmente, atribui-se um *array* vazio de inteiros de dimensão dinâmica a cada célula, que armazenará os índices das partículas nela contidas.

***Grid* de obstáculo** A detecção e o tratamento de colisões utilizando a técnica proposta neste trabalho baseia-se na construção de um *grid* tridimensional, cujas células representam um volume cúbico do espaço, que está ou não preenchido. Esta estrutura é criada através do Algoritmo 5 abaixo:

Algoritmo 5 Criação do *grid* de obstáculo

1: **min** $\leftarrow$ coordenadas do ponto mínimo do obstáculo
2: **max** $\leftarrow$ coordenadas ponto máximo do obstáculo
3: $side \leftarrow$ medida do lado da célula cúbica
4: **dim** $\leftarrow$ **max** $-$ **min**
5: **cells** $\leftarrow \lceil \frac{\mathbf{dim}}{side} \rceil$
6: $topography \leftarrow$ matriz booleana de dimensões dim
7: **para** $i \leftarrow 0$ **até** dim_x **faça**
8: **para** $j \leftarrow 0$ **até** dim_y **faça**
9: **para** $k \leftarrow 0$ **até** dim_z **faça**
10: $topography_{i,j,k} \leftarrow true$
11: **fim para**
12: **fim para**
13: **fim para**

Observa-se que a construção do *grid* que representa o obstáculo é feita de maneira bastante semelhante ao *grid* de busca, utilizando, neste caso, as coordenadas dos limites inferior e superior do obstáculo e a medida de seu lado, um dado de entrada da implementação contido no arquivo de configuração da cena, para determinar o número de células em cada direção através da divisão feita na linha 5. Na linha 10, o *grid*, implementado pela matriz *topografia* de booleanos, é inicializado com valores verdadeiros, que representam uma célula preenchida. A atribuição de poros ao obstáculo pode ser realizada através de dados topológicos armazenados em um arquivo de configuração do obstáculo ou aleatoriamente a partir de um valor percentual de porosidade, como expressado abaixo no Algoritmo 6:

Algoritmo 6 Atribuição de valores ao *grid* de obstáculo

$porosity \leftarrow$percentual do obstáculo ocupado por poros
$pores \leftarrow dim_x \times dim_y \times dim_z \times porosidade$
$i \leftarrow$ número aleatório entre 0 e dim_x
$j \leftarrow$ número aleatório entre 0 e dim_y
$k \leftarrow$ número aleatório entre 0 e dim_z
enquanto $pores > 0$ **faça**
  enquanto $topography_{i,j,k} = false$ **faça**
    $i \leftarrow$ número aleatório entre 0 e dim_x
    $j \leftarrow$ número aleatório entre 0 e dim_y
    $k \leftarrow$ número aleatório entre 0 e dim_z
  fim enquanto
  $topography_{i,j,k} \leftarrow false$
  $pores \leftarrow pores - 1$
fim enquanto

Neste algoritmo, obtém-se o número de células que representam poros do obstáculo ao multiplicar o número total, dado pelo produto das coordenadas do vetor **dim** calculado no Algoritmo 5, pelo percentual de porosidade do obstáculo, obtido do arquivo de configuração da cena, como feito na linha 2 do algoritmo. Em seguida, é feito um *loop*, a cada passo do qual é escolhida uma célula aleatória do *grid* do obstáculo, verificando se o valor nela armazenado é verdadeiro, indicando que ela está preenchida e, em caso positivo, alterando-o para falso, o que torna a célula em questão um poro. Este processo ocorre até que tenham sido atribuídos todos os poros ao obstáculo. Ao final da atribuição, a topografia do obstáculo é armazenada em um arquivo *ObstacleSettings.settings* para permitir que a mesma configuração de poros possa ser utilizada em outras simulações.

Sistema de fluidos A inicialização do sistema de fluidos é feita com base no tipo de cena simulada, determinada pelo atributo *scene* do arquivo de configuração $SceneSettings$. Através das informações sobre os limites que definem os volumes de fluido, armazenadas nos arquivos $FirstFluidSettings$ e $SecondFluidSettings$, cada uma das fases do fluido é criada utilizando o algoritmo.

No algoritmo acima, obtêm-se, primeiramente, os pontos mínimo e máximo do volume de fluido, subtraindo-os na linha 3 para obter um vetor com as dimensões do volume. Em seguida, na linha 5, divide-se este vetor pelo espaçamento entre as partículas em sua

Algoritmo 7 Inicialização de uma fase de fluido como volume livre

1: **min** $\leftarrow$ coordenadas do ponto mínimo do volume de fluido
2: **max** $\leftarrow$ coordenadas do ponto máximo do volume de fluido
3: **dim** $\leftarrow$ **max** $-$ **min**
4: $spacing \leftarrow$ espaçamento entre as partículas do fluido
5: **parts** $\leftarrow \lfloor \frac{\mathbf{dim}}{spacing} \rfloor$
6: $n \leftarrow parts_x \times parts_y \times parts_z$
7: **base** $\leftarrow$ **min** $+ \frac{spacing}{2} \times (1, 1, 1)$
8: **para** $z \leftarrow base_z$ **até** max_x **com** $z \leftarrow z + spacing$ **faça**
9: **para** $y \leftarrow base_y$ **até** max_y **com** $y \leftarrow y + spacing$ **faça**
10: **para** $x \leftarrow base_x$ **até** max_x **com** $x \leftarrow x + spacing$ **faça**
11: $createParticle(x, y, z, id, v_0)$
12: **fim para**
13: **fim para**
14: **fim para**

distribuição inicial, dado como parâmetro no arquivo de configurações da fase de fluido em questão, obtendo um vetor com as dimensões do *grid* usado como suporte para o posicionamento das partículas. Esta etapa considera o piso da divisão de forma a evitar um aumento do número de partículas, o que poderia acarretar no posicionamento de algumas do lado de fora do domínio da cena. Multiplicando-se as coordenadas do vetor encontrado, é possível obter na linha 6 o número total de partículas da fase do fluido criada, informação esta de grande importância para a simulação, pois permite a iteração correta sobre todas as partículas no *loop* principal. Calcula-se, então, um ponto de base, somando a metade do espaçamento às coordenadas do ponto mínimo do fluido, sobre o qual será posicionada a primeira partícula. Isto é feito para que as partículas criadas sejam posicionadas no centro de cada célula do *grid* de suporte, evitando que haja partículas contidas nos planos que delimitam a cena, o que poderia causar problemas ao atribuí-las às células do *grid* de busca. A partir deste ponto de base, as iterações nas linhas 8 a 10 obtém as posições da partícula ao somar o valor do espaçamento às coordenadas da base. Por fim, o método *createParticle* na linha 11 do algoritmo 7 atribui à partícula a posição (x, y, z) calculada, assim como um inteiro *id* que permite identificar a que fase ela pertence, informação importante na simulação de fluidos bifásicos para que se possa utilizar as propriedades físicas corretas, e uma velocidade inicial v_0, informações provenientes dos arquivos de configuração.

Uma outra opção de inicialização do fluido consiste em posicionar as partículas no interior dos poros do obstáculo ao invés de em um volume paralelepipedal inicial. Este processo está representado pelo algoritmo 8:

Algoritmo 8 Inicialização de uma fase de fluido no interior dos poros

1: $n \leftarrow 0$
2: **min** $\leftarrow$ coordenadas do ponto mínimo do obstáculo
3: **max** $\leftarrow$ coordenadas do ponto máximo do obstáculo
4: $side \leftarrow$ lado da célula obstáculo
5: **base** $\leftarrow$ **min** $+ \frac{h}{2} \times (1, 1, 1)$;
6: **para** $z \leftarrow base_z$ **até** max_z **com** $z \leftarrow z + side$ **faça**
7: **para** $y \leftarrow base_y$ **até** max_y **com** $y \leftarrow y + side$ **faça**
8: **para** $x \leftarrow base_x$ **até** max_z **com** $x \leftarrow x + side$ **faça**
9: **se** a posição (x, y, z) está em um poro do obstáculo **então**
10: $createParticle(x, y, z, id, v_0)$;
11: $n \leftarrow n + 1$;
12: **fim se**
13: **fim para**
14: **fim para**
15: **fim para**

Neste algoritmo, o ponto de base, sobre o qual verificar-se-á se é possível criar uma partícula, é definido como o limite mínimo do obstáculo poroso, acrescido de metade da medida do lado das células que compõem o *grid* que representa o obstáculo. Este ponto corresponderá ao centro da primeira célula do *grid* do obstáculo. A iteração desta posição feita nas linhas 6 a 8 acrescenta a ela o valor do espaçamento, obtendo, assim, uma posição (x, y, z) correspondente ao centro de cada célula do *grid* do obstáculo. Verifica-se, então, se a célula em questão corresponde a um poro ou uma porção preenchida. Caso ela indique uma região vazia do obstáculo, é criada uma partícula no centro desta célula utilizando o mesmo método *createParticle* utilizado no algoritmo 7, incrementando também o valor de n, que guarda o número de partículas da fase de fluido inicializada.

Por fim, é criada uma cópia do objeto *fluid*, chamda aqui de *nextFluid*, que é utilizada para salvar as informações do estado recém-calculado do fluido a cada passo da simulação. Isto é necessário para permitir que o cálculo do estado n do fluido considere apenas informações de partículas no passo $n - 1$, o que seria impossibilitado caso os resultados obtidos no *loop* principal sobrescrevessem os armazenados no objeto *fluid*, que são usados como parâmetro do mesmo.

5.4.1 *Loop* principal

A atualização da cena é possibilitada pelo *loop* de renderização da simulação, representada pela classe *SPHApp*, apresentado no algoritmo 9 abaixo.

Algoritmo 9 *Loop* de renderização da simulação

para $s \leftarrow 0$ **até** $maxSteps$ **faça**
  $updateScene()$
  $renderScene()$
fim para

O método *updateScene()* é responsável por avaliar as posições em um dado estado s das partículas do fluido a partir de seu estado anterior $s-1$, atribuir as informações de seu estado recém-calculado ao objeto $nextFluid$. Ao final da execução do mesmo, atribui-se ao objeto $fluid$ uma cópia de $nextFluid$, de forma que, na próxima chamada deste método, que calculará o estado do fluido no passo $s+1$, serão utilizadas corretamente as informações do estado s, agora armazenadas em $fluid$, para realizar os cálculos do SPH. O algoritmo 10 mostra este processo.

Algoritmo 10 *Loop* de atualização da cena - *updateScene()*

para $p \leftarrow 0$ **até** n **faça**
  $updateParticle(p)$;
fim para
$fluid = nextFluid$;

O procedimento de atualização das partículas, descrito pelo Algoritmo 11, utiliza a modelagem computacional do método SPH, apresentada no capítulo 3, para obter a nova posição de uma partícula p através da avaliação da equação de Navier-Stokes e de integrações numéricas.

Algoritmo 11 Atualização de partícula - *updateParticle(p)*

$computeDensity()$	▷ Equação 3.10
$computePressure()$	▷ Equação 3.11
$computePressureForce()$	▷ Equação 3.13
$computeViscosityForce()$	▷ Equação 3.15
$computeTotalForce()$	▷ Equação 3.16
$computeAcceleration()$	▷ Equação 3.17
$computeVelocity()$	▷ Equação 4.1
$computePosition()$	▷ Equação 4.2

5.4.2 Detecção e tratamento de colisões

Ao final da etapa de cálculo para cada partícula, são observadas sua posição anterior $\mathbf{p_{s-1}}$ e a nova $\mathbf{p_s}$, recém-calculada ao final do Algoritmo 11. Através de uma operação simples, descrita no Algortimo 13, é possível obter a que posição (i, j, k) do *grid* do obstáculo cada uma delas pertence. Caso a célula na qual está contida a nova posição $\mathbf{p_s}$ da partícula esteja preenchida, pode-se utilizar a posição anterior $\mathbf{p_{s-1}}$ para determinar qual face da célula cúbica do obstáculo foi atravessada e, utilizando seus limites máximo e mínimo, alterar a posição e a velocidade da partícula de forma a representar sua colisão com o meio. Este processo é uma simplificação do discutido na seção 3.3, possibilitada pelo fato das faces das células cúbicas serem paralelas aos planos xy, xz e yz espaço da cena. A figura 5.1 ilustra este processo de maneira simplificada bidimensional.

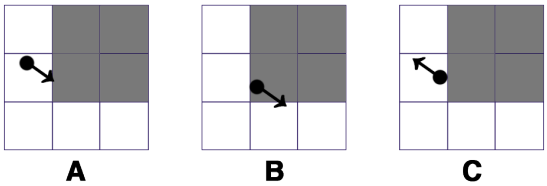


Figura 5.1: Técnica de de colisão utilizada. A figura A representa a posição antiga da partícula e seu vetor velocidade. B mostra a sua posição recém-calculada, que está numa célula preenchida, e C mostra sua nova posição e velocidade após o tratamento.

Nas terceira linha do algoritmo, a detecção da colisão é feita ao verificar se a célula do obstáculo poroso no qual a partícula está contida mudou e, em caso positivo, se a nova célula está preenchida, ou seja, não é um poro, o que é feito ao verificar o valor booleano armazenado na matriz *topography*, criada no Algoritmo 5. Caso isto ocorra, deve-se determinar qual face da célula cúbica preenchida foi atravessada pela partícula de forma a calcular a sua posição corrigida. Isto é feito ao comparar os índices da célula (i, j, k) na qual estava a posição $\mathbf{p_{s-1}}$ pelos da célula (l, m, n) na qual está nova posição $\mathbf{p_s}$. Caso os índices antigos sejam menores, a célula foi atravessada por das faces de seu limite inferior, sendo este valor é atribuído à sua coordenada adequada, considerando uma pequena margem, armazenada na variável $margin$ para evitar que a posição esteja contida na face da célula. De forma similar, caso os índices antigos sejam maiores, isto indica que a partícula atravessou a célula por uma face de seu limite superior e, portanto, este valor, somado à margem, será atribuído à coordenada adequada de sua posição corrigida. Em todos os casos, o valor da coordenada da velocidade é substituído por seu simétrico de forma a considerar o efeito de reflexão ocorrido. Além disso, o vetor velocidade é

Algoritmo 12 Detecção e tratamento de colisões

1: $(i, j, k) \leftarrow$ célula do obstáculo da posição antiga da partícula p
2: $(l, m, n) \leftarrow$ célula do obstáculo da posição nova da partícula p
3: **se** $(i, j, k) \neq (l, m, n)$ **e** $topography_{l,m,n} = true$ **então**
4: **se** $i < l$ **então**
5: $x \leftarrow \text{lower}(l, m, n)_x - margin$
6: $v_x \leftarrow -v_x$
7: **senão**
8: **se** $i > l$ **então**
9: $x \leftarrow \text{upper}(l, m, n)_x + margin$
10: $v_x \leftarrow -v_x$
11: **fim se**
12: **fim se**
13: **se** $j < m$ **então**
14: $y \leftarrow \text{lower}(l, m, n)_y - margin$
15: $v_y \leftarrow -v_y$
16: **senão**
17: **se** $j > m$ **então**
18: $y \leftarrow \text{upper}(l, m, n)_y + margin$
19: $v_y \leftarrow -v_y$
20: **fim se**
21: **fim se**
22: **se** $k < n$ **então**
23: $z \leftarrow \text{lower}(l, m, n)_z - margin$
24: $v_z \leftarrow -v_z$
25: **senão**
26: **se** $k > n$ **então**
27: $z \leftarrow \text{upper}(l, m, n)_z + margin$
28: $v_z \leftarrow -v_z$
29: **fim se**
30: **fim se**
31: $\mathbf{p_s} \leftarrow (x, y, z)$
32: $\mathbf{v} \leftarrow (v_x, v_y, v_z) * \beta$
33: **fim se**

multiplicado pela constante de amortecimento β, empregada para simular a perda de energia cinética ocorrida devido à colisão do fluido com o obstáculo.

Esta estratégia é capaz de realizar o tratamento de colisões em tempo constante para cada posição calculada sem a necessidade de inserir partículas fantasma. Isto a torna uma boa opção para simulações de fluidos bifásicos em meios porosos, nas quais este tipo de partícula pode interferir negativamente na eficiência da simulação e na qualidade de seu resultado final. A implementação realizada engloba apenas casos com um obstáculo paralelepipedal, porém, esta técnica pode ser utilizada para obstáculos múltiplos ou de geometrias mais complexas.

O Algoritmo 13 abaixo apresenta os passos necessários para obter a célula que contém uma dada posição no espaço, como feito nas duas primeiras linhas do algoritmo anterior:

Algoritmo 13 Obtenção de uma célula do obstáculo a partir de coordenadas espaciais

função POSTOCELL(**pos**)
  min $\leftarrow$ coordenadas do ponto mínimo do obstáculo
  side $\leftarrow$ lado da célula obstáculo
  cell $\leftarrow \lfloor \frac{\mathbf{pos}-\mathbf{min}}{side} \rfloor$
  devolve cell
fim função

Nele, subtrai-se o ponto mínimo do obstáculo do vetor avaliado, de forma a obter sua posição relativa ao limite inferior do obstáculo. Em seguida, ao realizar uma divisão inteira das coordenadas do resultado pela medida do lado da célula do *grid* que representa o obstáculo, obtém-se um vetor (i, j, k) com os índices da célula na qual a posição dada como parâmetro está contida.

É necessário também obter as coordenadas do limite inferior e superior de uma célula de forma a obter a posição corrigida da partícula após o tratamento da colisão, como visto nas linhas 5 e 9 do algoritmo 12, por exemplo. Isto é feito através do algoritmo 14 abaixo, cujas funções, dados os índices (i, j, k) de uma célula do *grid* do obstáculo, retornam as coordenadas dos limites inferior e superior que a definem:

Algoritmo 14 Obtenção de coordenadas espaciais a partir de uma célula do obstáculo

função LOWER(**cell**)
  min $\leftarrow$ coordenadas do ponto mínimo do obstáculo
  side $\leftarrow$ lado da célula obstáculo
  pos $\leftarrow$ **min** $+$ **cell** $\times$ *side*
  devolve pos
fim função

função UPPER(cell)
  min $\leftarrow$ coordenadas do ponto mínimo do obstáculo
  side $\leftarrow$ lado da célula obstáculo
  pos $\leftarrow$ **min** $+ (\mathbf{cell} + (1, 1, 1)) \times$ *side*
  devolve pos
fim função

Este algoritmo funciona de forma simétrica ao 13, multiplicando os índices da célula pela medida de seu lado e somando-a ao ponto mínimo do obstáculo. Para retornar o limite superior da célula, soma-se uma unidade a seus índices antes de realizar o cálculo. Isto se deve ao fato deste ponto coincidir com o limite inferior da célula vizinha encontrada ao realizar um deslocamento de uma unidade no sentido positivo de cada direção no *grid*.

Capítulo 6

Resultados

6.1 Metodologia dos experimentos

Os testes foram realizados em um *notebook MacBook Air* que conta com um processador *Intel Core i5 Mobile* de 1.7 GHz, 4GB de memória RAM DDR3 de 1600 MHz e placa gráfica integrada *Intel HD Graphics 4000* com 384 MB de memória de vídeo compartilhada. Em todos os casos, as simulações ocorreram por 1000 passos. O domínio da cena foi determinado como o cubo definido entre os pontos $(-24.0, -24.0, -24.0)$ e $(24.0, 24.0, 24.0)$, sendo o obstáculo poroso o paralelepípedo determinado pela metade inferior do domínio da cena, ou seja, abaixo do plano $y = 0.0$, com células de lado de tamanho $side = 2.0$. Para facilitar a observação dos resultados, foi renderizada apenas a porção da cena contida entre os planos $z = -12.0$ e $z = 12.0$.

O objetivo dos testes feitos neste cenário é observar a variação da taxa de volume absorvido por um obstáculo poroso causada pela mudança de sua porosidade, ou seja, o percentual de seu volume total que é ocupado por poros. Foram realizados dois casos de teste de forma a analisar o comportamento do escoamento de fluidos tanto monofásicos quanto bifásicos. Em ambos, a distribuição dos poros no obstáculo foi feita de maneira que, considerando dois conjuntos A e B de todos os poros do obstáculo em cenários de teste distintos cujos números de elementos são tais que $n(A) < n(B)$, tem-se que $A \subset B$. Foram usadas as mesmas configurações de poros nos obstáculos tanto no teste utilizando um fluido monofásico quanto no caso de um fluido bifásico. Isto foi feito para permitir uma observação do comportamento do escoamento de dois fluidos distintos através de um mesmo obstáculo, visto que, devido à aleatoriedade da distribuição de poros no obstáculo, a utilização de configurações distintas poderia acarretar em diferenças nos resultados que não estivessem relacionadas às características do fluido utilizado no teste.

A contabilização dos volumes de fluido inseridos ou removidos do obstáculo feita nos testes foi realizada através da contagem de partículas que ultrapassaram o plano $y = 0.0$, que determina o topo do obstáculo. Isto foi possível pois o método SPH considera que cada partícula contém uma mesma quantidade de volume.

6.2 Teste 1: escoamento de fluido monofásico

No primeiro teste realizado, foi posicionado um volume de fluido sobre o obstáculo poroso, de forma que, sob o efeito da força da gravidade, o fluido fosse absorvido por ele, escoando através de seus poros. Este cenário é ilustrado na Figura 6.1 a seguir:

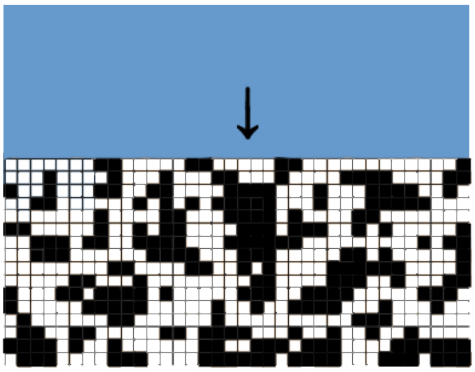
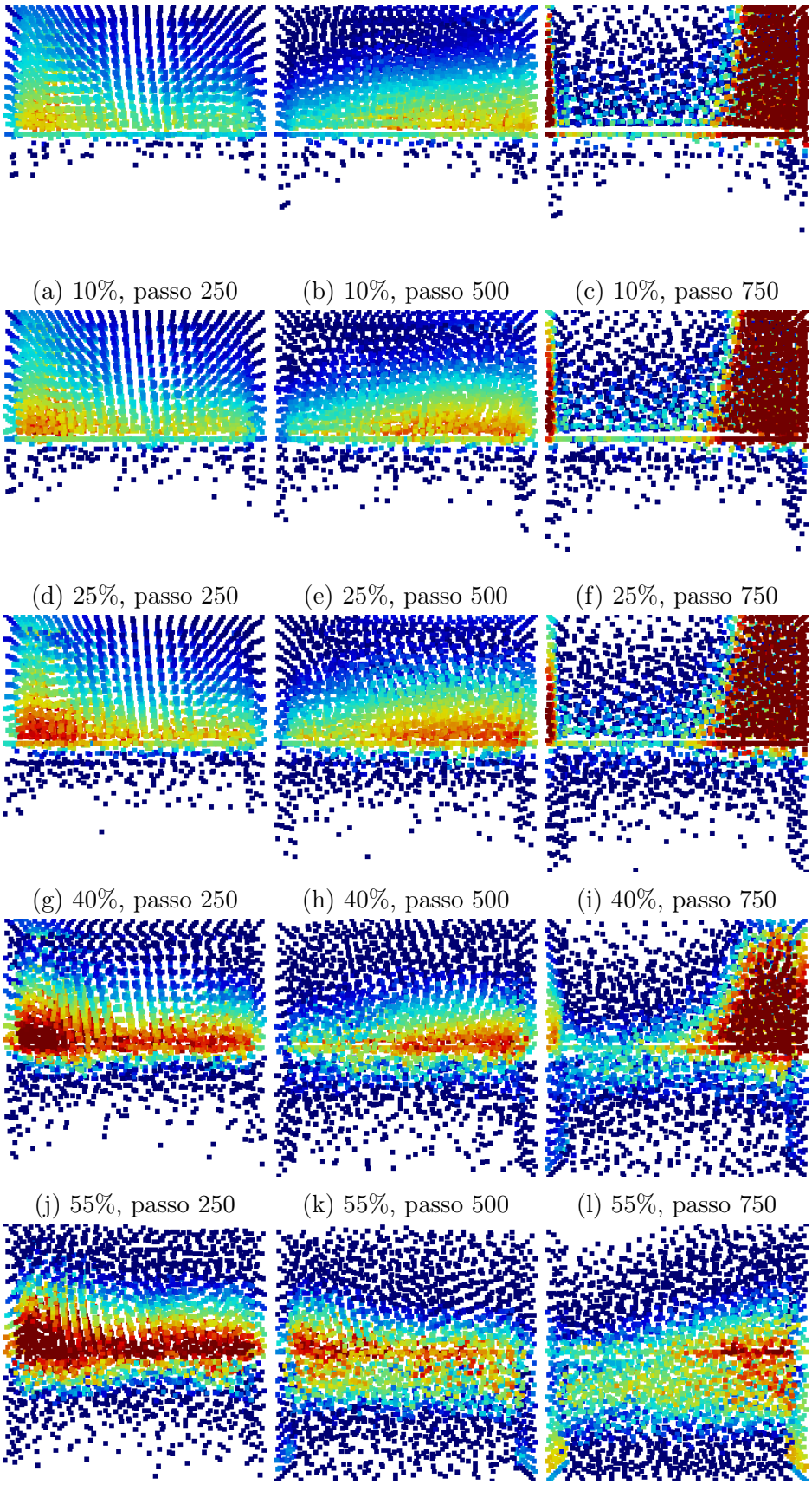


Figura 6.1: Cenário dos testes com fluido monofásico.

O fluido simulado possui como propriedades massa $m = 40$, viscosidade $mu = 80$ e constante de pressão $k = 80$. Foram realizados 15 cenários de teste, aumentando a porosidade do obstáculo de 5% até 75% em incrementos de 5%. A Figura 6.2 apresenta capturas de tela dos passos de número 250, 500 e 750 das simulações referentes às porosidades de $10, 25, 40, 55$ e 70%. Nos testes realizados, observou-se o percentual de volume de fluido que penetrou o obstáculo poroso, relacionando esta quantidade à porosidade em casa caso de forma a gerar o gráfico observado abaixo na figura 6.3.



(a) 10%, passo 250 (b) 10%, passo 500 (c) 10%, passo 750

(d) 25%, passo 250 (e) 25%, passo 500 (f) 25%, passo 750

(g) 40%, passo 250 (h) 40%, passo 500 (i) 40%, passo 750

(j) 55%, passo 250 (k) 55%, passo 500 (l) 55%, passo 750

(m) 70%, passo 250 (n) 70%, passo 500 (o) 70%, passo 750

Figura 6.2: Capturas de tela de cenários do teste bifásico.

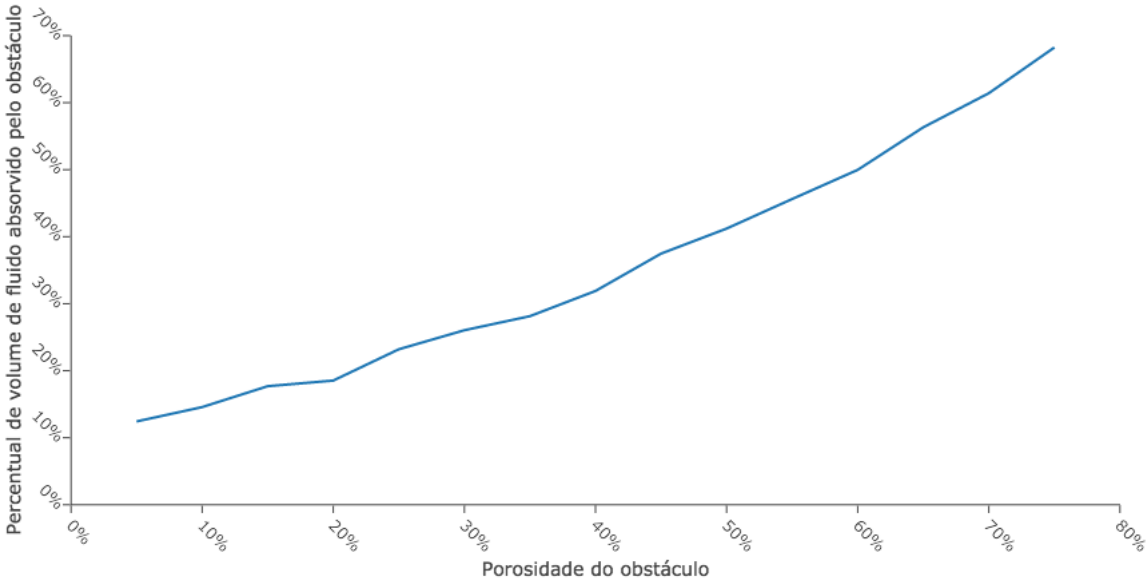


Figura 6.3: Porosidade do obstáculo × percentual de volume de fluido absorvido pelo obstáculo.

Pode-se observar nas imagens da Figura 6.2 que, com o aumento da porosidade do obstáculo, um volume maior de fluido é capaz de penetrar no obstáculo, como esperado. Observa-se também que há uma concentração maior de altas pressões, visível especialmente no lado do direto da cena no passo 750 nos casos de menor porosidade, como na Figura 6.2(c), por exemplo. Isto é uma consequência do movimento de ondulação do fluido que ocorre a partir do início da simulação devido ao ajuste natural das posições das partículas para se adequarem às propriedades físicas do fluido. Nos casos com porosidades menores, a falta de canais pelos quais o fluido possa escoar faz com que um volume maior do mesmo fique confinado sobre o obstáculo até o final da simulação, quando a ondulação do fluido atinge a parede da direita do tanque. Devido a este número maior de partículas aglomeradas em uma região da cena, observa-se um aumento de suas pressões consistente com o modelo físico. Percebe-se que, com o aumento da porosidade, o fluido passa a poder escoar pelo obstáculo poroso com mais facilidade, o que diminui a quantidade de partículas confinadas sobre ele e, consequentemente, reduz a concentração de altas pressões observada. Este efeito pode ser observado na variação de cores mais suave vista nos campos de pressão dos casos de teste realizados com um obstáculo de maior porosidade, como vê-se, por exemplo, na Figura 6.2(o).

No gráfico da Figura 6.3, é possível observar que há uma relação aproximadamente

linear entre a porosidade de obstáculo e a taxa de fluido absorvida pelo obstáculo. Este comportamento está de acordo com o esperado, visto que o aumento da porosidade indica um maior número de poros no obstáculo a serem preenchidos pelas partículas de fluido que nele penetram sob o efeito da força da gravidade, aumentando, assim, o percentual de volume de fluido absorvido. A variação observada na taxa de crescimento desta quantidade entre cada par consecutivo de porosidades do obstáculo utilizadas nos casos de teste pode ser justificada pela característica aleatória da distribuição de poros no obstáculo. Isto faz com que alguns conjuntos de poros adicionados a cada incremento de porosidade sejam mais beneficiais ao escoamento do fluido do que outros, acarretando nas pequenas variações vistas no gráfico.

6.3 Teste 2: escoamento de fluido bifásico

O segundo cenário analisado consistiu no caso de um obstáculo paralelepipedal cujos poros estavam preenchidos com fluido, sobre o qual foram posicionados dois canais, um deles injetando um fluido com diferentes propriedades no obstáculo de forma a pressurizar seu interior, permitindo o escoamento do fluido pelo segundo canal. O diagrama da Figura 6.4 ilustra o cenário modelado:

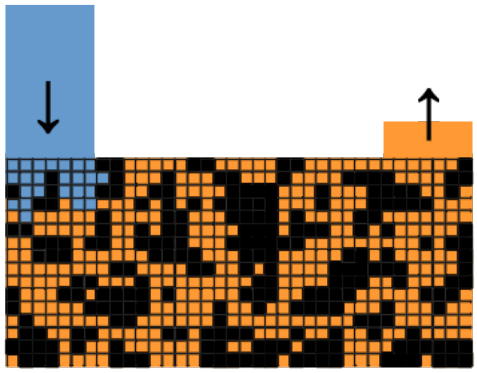


Figura 6.4: Cenário dos testes com fluido bifásico.

Observando de antemão o cenário da simulação, é possível concluir que a porosidade do obstáculo, ou seja, a porcentagem de volume do mesmo que corresponde a espaços vazios, influencia a proporção de volume de fluido analisada nos testes. Isto se deve ao fato de que uma quantidade menor de poros acarreta em maior dificuldade para o fluido penetrar no obstáculo, de forma a aumentar a pressão em seu interior e causar a ejeção de parte do fluido nele armazenado. Assim, espera-se observar um aumento na razão $\frac{V_r}{V_i}$ de forma linearmente proporcional à porosidade do obstáculo.

As propriedades dos fluidos envolvidos nesta simulação foram mantidas fixas em todas as instâncias dos testes e seus respectivos valores estão listados abaixo na tabela 6.3.

Propriedade	Fluido inserido	Fluido removido
Massa da partícula (m)	40.0	30.0
Viscosidade (μ)	80.0	90.0
Constante de pressão (k)	80.0	80.0

Foram realizados 15 cenários de teste, incrementando a porosidade da rocha em 5% em cada um deles. A Figura 6.6 mostra algumas capturas de tela do programa durante a execução do teste. Durante a execução das simulações envolvidas neste teste, foram obtidos também cortes transversais ao eixo Y, como ilustrado no esquema da Figura 6.5, entre os planos $y = -2.0$ e $y = 0.0$, contendo, assim, a primeira camada de células do *grid* que representa o fluido. Os cortes feitos durante os passos de número $250, 500$ e 750 do teste com um obstáculo de porosidade 70%, escolhidos por ilustrarem melhor o campo de pressões no plano observado devido ao maior número de partículas no interior do obstáculo, são apresentados na Figura 6.7. Os resultados obtidos estão representados no gráfico na Figura 6.8, que relaciona a porosidade do obstáculo com a razão entre o volume de fluido removido e inserido no obstáculo.

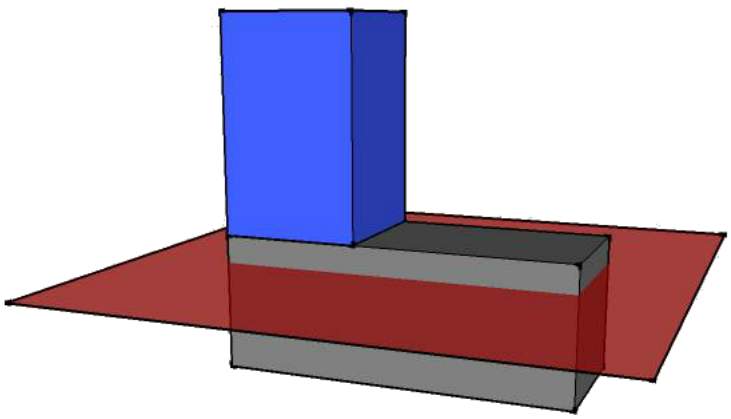
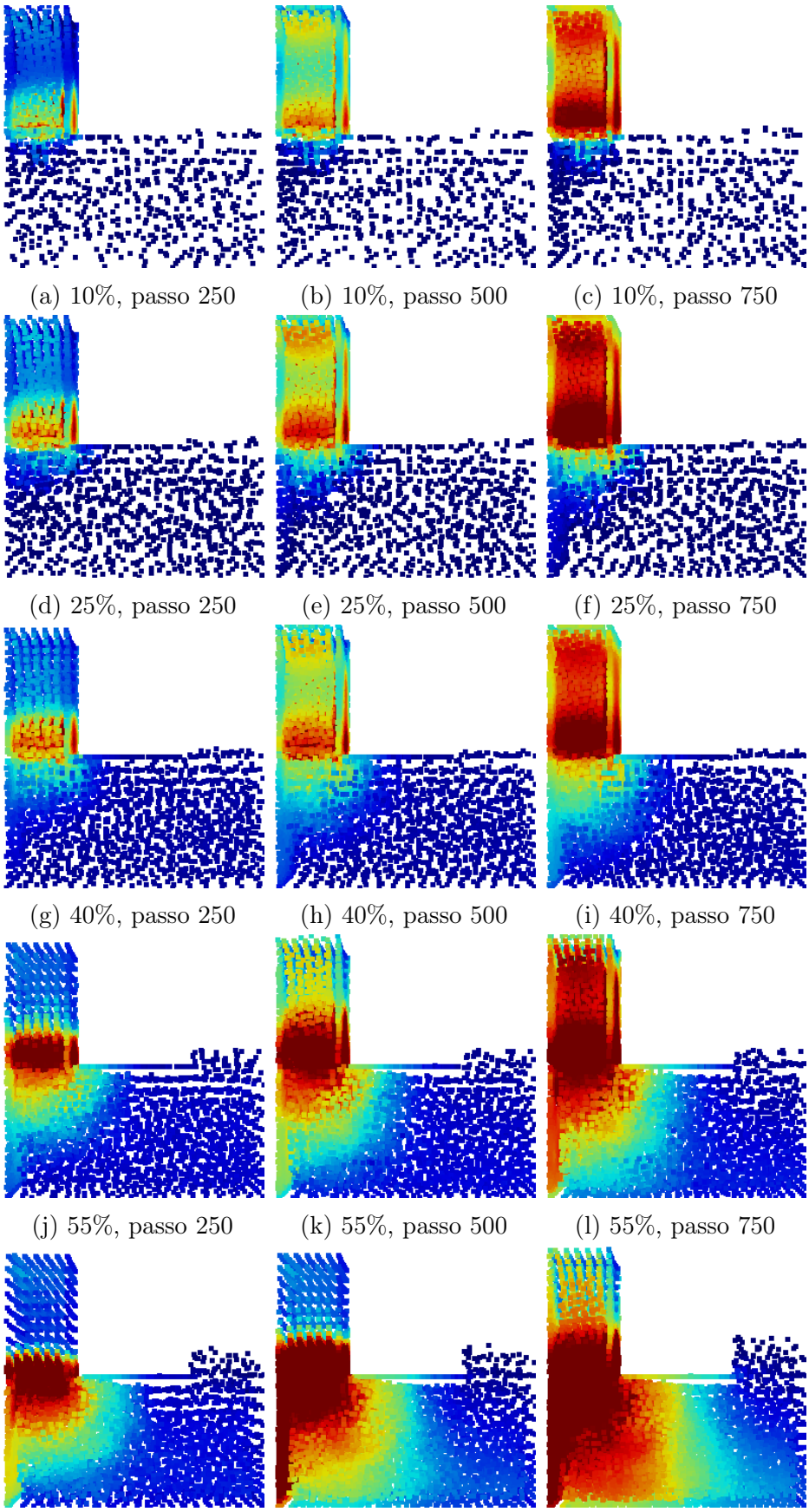


Figura 6.5: Corte realizado nos testes com fluido bifásico.



(a) 10%, passo 250 (b) 10%, passo 500 (c) 10%, passo 750

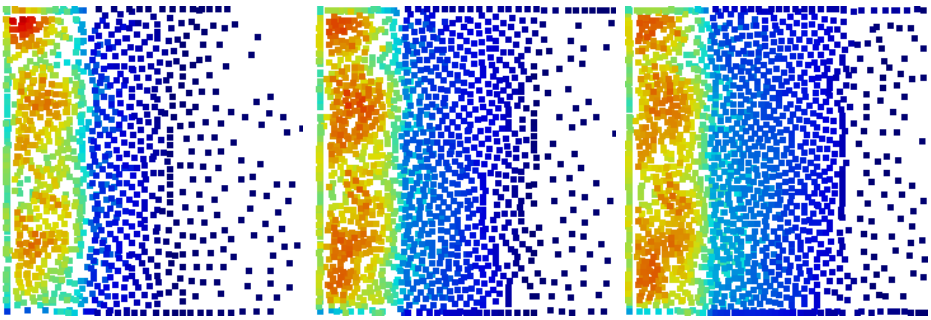
(d) 25%, passo 250 (e) 25%, passo 500 (f) 25%, passo 750

(g) 40%, passo 250 (h) 40%, passo 500 (i) 40%, passo 750

(j) 55%, passo 250 (k) 55%, passo 500 (l) 55%, passo 750

(m) 70%, passo 250 (n) 70%, passo 500 (o) 70%, passo 750

Figura 6.6: Capturas de tela de cenários do teste bifásico



(a) 70%, passo 250 (b) 70%, passo 500 (c) 70%, passo 750

Figura 6.7: Cortes transversais do obstáculo poroso, compreendendo a camada de células com coordenada $y \in [-2.0, 0.0]$.

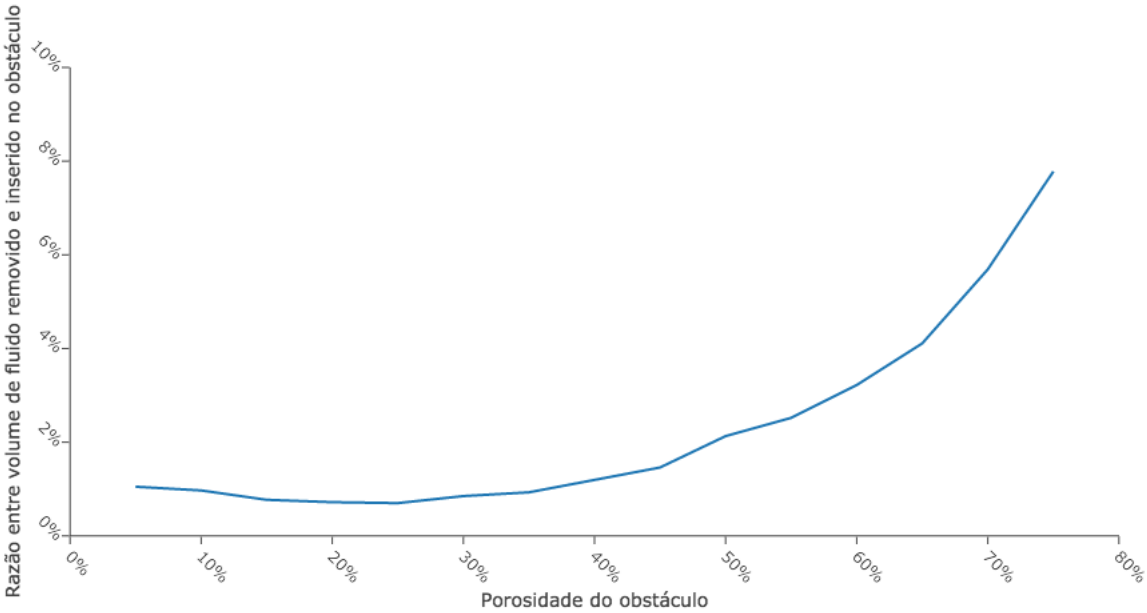


Figura 6.8: Porosidade do obstáculo × razão entre volume de fluido removido e inserido no obstáculo.

É possível notar nas capturas de tela da Figura 6.6 que, nos casos de menor porosidade, a falta de poros através dos quais o fluido inserido possa escoar faz com que ele fique confinado, causando uma concentração das pressões mais altas no interior do canal por onde ele é injetado no fluido, como ilustrado, por exemplo, nas Figuras 6.6(c), 6.6(f) e 6.6(i). Como consequência da impossibilidade do fluido inserido de escoar no interior do obstáculo, observa-se que, à direita, no canal de saída, o volume de fluido que conseguiu escapar do obstáculo é pequeno. Com o aumento da porosidade do obstáculo, porém, o maior número de poros faz com que o fluido injetado consiga escoar com mais facilidade, acarretando numa distribuição mais uniforme das pressões ao passo que ele percorre o

interior do obstáculo, como visto na Figura 6.6(o), por exemplo. Isto faz com que uma quantidade maior do fluido contido no interior dos poros seja expelida do obstáculo, como pode ser visto pelo maior número de partículas no interior do canal de saída nesta mesma figura.

Nos cortes da Figura 6.7, pode-se observar que algumas porções permanecem em branco mesmo com o decorrer do escoamento, indicando células do *grid* que indicam porções preenchidas do obstáculo, que são intransponíveis. Vê-se também uma região retangular à esquerda, no interior da qual há uma pressão média maior, que representa a parte do obstáculo por onde entra o fluido pressurizado proveniente do canal de entrada. Neste retângulo, pode-se observar algumas áreas nas quais há uma concentração de partículas com pressões maiores, o que se deve ao fato da distribuição aleatória tornar o escoamento mais fácil em algumas partes do obstáculo devido à maior quantidade de poros, enquanto, em outras, a falta de poros faz com que porções de fluido fiquem confinadas e, consequentemente, apresentem maior pressão. Por fim, é possível identificar também um segundo retângulo, à direita, no qual há uma menor concentração de partículas. Ele é referente à porção do obstáculo onde está posicionado o canal de saída, por onde as partículas expelidas em decorrência do escoamento do fluido injetado saem do obstáculo. Como as partículas possuem mais uma direção possível de movimento nesta região, podendo escoar pelo canal e, assim, ultrapassar o limite superior do obstáculo definido pelo plano $y = 0.0$, é esperado que haja menos partículas nela confinadas.

Observa-se também que o a taxa de crescimento da razão entre o volume de fluido removido e inserido é aproximadamente exponencial, indicando um aumento mais rápido do que o aproximadamente linear observado no teste anterior, realizado com um fluido monofásico. Isto se deve ao fato da razão observada neste teste ser calculada utilizando dois volumes de fluidos que são influenciados de maneiras distintas pelo aumento da porosidade do obstáculo. Enquanto o volume de fluido inserido aumenta apenas linearmente, assim como visto na seção anterior no resultado do teste monofásico, observa-se que o volume removido é mais fortemente influenciado pelo aumento da porosidade do fluido. Isto se deve ao fato de que um número de maior de poros influencia o volume de fluido expelido do obstáculo de mais de uma maneira:

- O fluido expelido consegue escoar no interior do obstáculo com mais facilidade, da mesma forma que ocorre com o fluido injetado e no teste monofásico;
- A maior facilidade de escoamento faz com que um volume maior do fluido inserido consiga percorrer o obstáculo, consequentemente pressurizando o fluido contido nos

poros e expelindo um volume maior do mesmo do obstáculo;

- Um número maior de poros acarreta em mais fluido confinado nos mesmos no interior do obstáculo, fazendo com que haja um volume maior para ser expelido devido à injeção de fluido pressurizado.

Assim, como a quantidade de fluido removido do obstáculo é influenciada tanto direta quanto indiretamente pelo aumento da porosidade, ao passo que o volume inserido é influenciado apenas diretamente, é esperado que a razão entre estas duas quantidades cresça mais rapidamente que a curva aproximadamente linear observada no teste anterior.

Capítulo 7

Conclusão

Neste trabalho, foi implementada uma simulação do escoamento de fluidos bifásicos através de meios porosos utilizando a técnica SPH. Seu diferencial consiste em uma abordagem alternativa ao tratamento das condições de contorno através de colisões geométricas das partículas que constituem o fluido com os polígonos que representam o obstáculo, substituindo o uso tradicional de partículas fantasma. A escolha desta estratégia foi motivada pela não-trivialidade da etapa de posicionamento destas partículas nos casos em que o suporte possui geometria complexa, como ocorre com obstáculos porosos. Além disso, nestes cenários, o grande número de partículas fantasma utilizadas causa um aumento na vizinhança média das partículas que constituem o fluido, o que pode acarretar numa perda de eficiência do algoritmo.

Observou-se que a utilização da técnica de colisão geométrica acarretou em um ganho de eficiência na implementação, tendo em vista a diminuição do número total de partículas envolvidas na simulação devido à não utilização de partículas fantasma para tratar a condição de contorno, além da remoção do *overhead* existente na etapa de inicialização e posicionamento das mesmas ao longo dos obstáculos da cena. Há também a vantagem de, no caso da simulação de fluidos bifásicos, não ocorrer casos em que partículas pertencentes a fases distintas incluem um mesmo conjunto de partículas fantasma em suas vizinhanças, o que poderia causar efeitos diferentes em cada uma delas devido às suas propriedades distintas. Esta estratégia também permite a utilização de modelos simplificados para representar o obstáculo, como o *grid* utilizado neste trabalho, causando uma diminuição do consumo de memória, visto que não há a necessidade de armazenar informações sobre faces ou vértices, além de simplificar a etapa de tratamento de colisões ao utilizar faces ortogonais aos eixos coordenados, o que acarreta em um ganho de eficiência.

7.1 Desafios e trabalhos futuros

A maior eficiência da simulação observada neste trabalho ao utilizar a técnica de colisão geométrica para tratar condições de contorno na simulação de fluidos está intrinsecamente ligada à representação do obstáculo. Em casos nos quais é necessário representar o escoamento através de um meio poroso de geometria complexa, envolvendo um grande número de faces e vértices que não podem ser simplificados sem que haja perda na qualidade da simulação, é necessário escolher estruturas de dados adequadas para armazenar suas informações geométricas, de forma a reduzir o tempo da busca pelas mesmas, necessárias para que se possa detectar e tratar colisões partícula-obstáculo. Além disso, também é importante utilizar técnicas de particionamento do obstáculo robustas de forma a diminuir o número de testes de detecção de colisão entre partículas e faces do obstáculo realizados, em especial quando sua geometria é complexa e a resolução da malha que a representa é alta.

Em trabalhos futuros na área de escoamento de fluidos bifásicos em meios porosos utilizando SPH, é interessante expandir a implementação realizada para suportar malhas tridimensionais quaisquer, utilizando técnicas de estruturas de dados e de particionamento especial para permitir a utilização de modelos mais complexos, representando obstáculos porosos de geometria mais intricada, como, por exemplo, os obtidos através da realização de escaneamento de rochas usando CT. Uma alternativa ao suporte de modelos 3D genéricos é a análise da geometria dos mesmos e sua subsequente simplificação em um *grid* similar ao utilizado neste trabalho, que represente o obstáculo da forma mais fidedigna possível através de componentes geometricamente simples. Desta forma, é possível usar as técnicas de detecção e tratamento de colisão simplificadas implementadas neste trabalho, que consideram o caso em que as faces das células do *grid* do obstáculo são ortogonais ao eixo, permitindo obter um resultado aproximado com maior eficiência. Finalmente, também é interessante estudar maneiras de representar bem não apenas a superfície do fluido, mas também a interface entre suas fases, permitindo introduzir a contribuição das tensões superficiais e interfaciais no cálculo das forças sobre as partículas, gerando resultados mais fiéis à realidade.

Referências

[1] Bridson, M. *Fluid Simulation For Computer Graphics.* A. K. Peters, 2009.

[2] Butler, S.; Kohles, S.; Thielke, R.; Chen, C.; Jr., R. V. Interstitial fluid flow in tendons or ligaments: a porous medium finite elements simulation. *Medical and Biological Engineering and Computing 35*, 6 (Novembro 1997), 742–746.

[3] Cassel, D.; Brown, J.; Johnson, G. A pore-scale numerical model for flow through porous media. *Theoretical and Applied Climatology 42*, 4 (1990), 223–228.

[4] Collins, R. *Flow of fluids through porous materials.* Petroleum Publishing Co., 1976.

[5] Desbrun, M.; Cani, M. Smoothed particles: A new paradigm for animating highly deformable bodies. In *Proceedings of EG Workshop on Animation and Simulation* (Agosto 1996), pp. 61–76.

[6] Desbrun, M.; Gascuel, M. Animating soft substances with implicit surfaces. In *Proceedings of SIGGRAPH'95* (Los Angeles, EUA, Agosto 1995), pp. 287–290.

[7] Gamito, M.; Lopes, P.; Gomes, M. Two-dimentional simulation of gasous phenomena using vortex particles. *6th Eurographics Workshop on Animation and Simulation* (Setembro 1995).

[8] Gingold, R.; Monaghan, J. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society 181* (Novembro 1977), 375–389.

[9] Landrini, M.; Colagrossi, A.; Tulin, M. A novel sph formulation for 2-phase flows. In *Proceedings of the 17th International Workshop on Water Waves and Floating Bodies* (2002).

[10] Lombardo, J.; Puech, C. Oriented particles: A tool for shape memory objects modelling. *Graphics Interface '95* (Maio 1995).

[11] Luciani, A.; Habibi, A.; Vapillon, A.; Duroc, Y. A physical model of turbulent fluids. *6th Eurographics Workshop on Animation and Simulation* (Setembro 1995).

[12] Luciani, A.; Jimenez, S.; Raoult, O.; Cadoz, C.; Florens, J. A unified view of mutitude behaviour, flexibilty, plasticity, and fractures: balls, bubbles and agglomerates. *IFIP WG 5.10 Working Conference* (Abril 1991).

[13] Lucy, L. A numerical approach to the testing of the fission hypothesis. *Astronomical Journal 82* (Dezembro 1977), 1013–1024.

[14] Miller, G.; Pearce, A. Globular dynamics: A connected particle system for animating viscous fluids. *SIGGRAPH'89 Courses 30 notes* (Agosto 1989), 305–309.

[15] Milosevic, M.; Fyle, A.; Hill, R. The relationship between elevated interstitial fluid pressure and blood flow in tumors: a bioengineering analysis. *International Journal of Radiation Oncology*Biology*Physics 43*, 5 (Março 1999), 1111–1123.

[16] Monaghan, J. Gravity currents and solitary waves. *Physica D - Special issue on nonlinear phenomena in ocean dynamics 98*, 2–4 (Novembro 1996), 523–533.

[17] Monaghan, J.; Cas, R.; Kos, A.; Hallworth, M. Gravity currents descending a ramp in a stratified tank. *Journal of Fluid Mechanics 379*, 1 (Janeiro 1999), 39–69.

[18] Müller, M.; Charypar, D.; Gross, M. Particle-based fluid simulation for interactive applications. In *Proceedings of ACM SIGGRAPH Symposium on Computer Animation* (San Diego, EUA, Julho 2003), pp. 154–159.

[19] Pasta, J.; Ulam, S. Heuristic numerical work in some problems of hydrodynamics. *Mathematical Tables and Other Aids to Computation 13*, 65 (Janeiro 1959), 1–12.

[20] Reeves, W. Particle systems–a technique for modeling a class of fuzzy objects. *Computer Graphics 17*, 3 (Julho 1983), 359–375.

[21] Schechter, H.; Bridson, R. Ghost sph for animating water. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2012) 31*, 4 (2012).

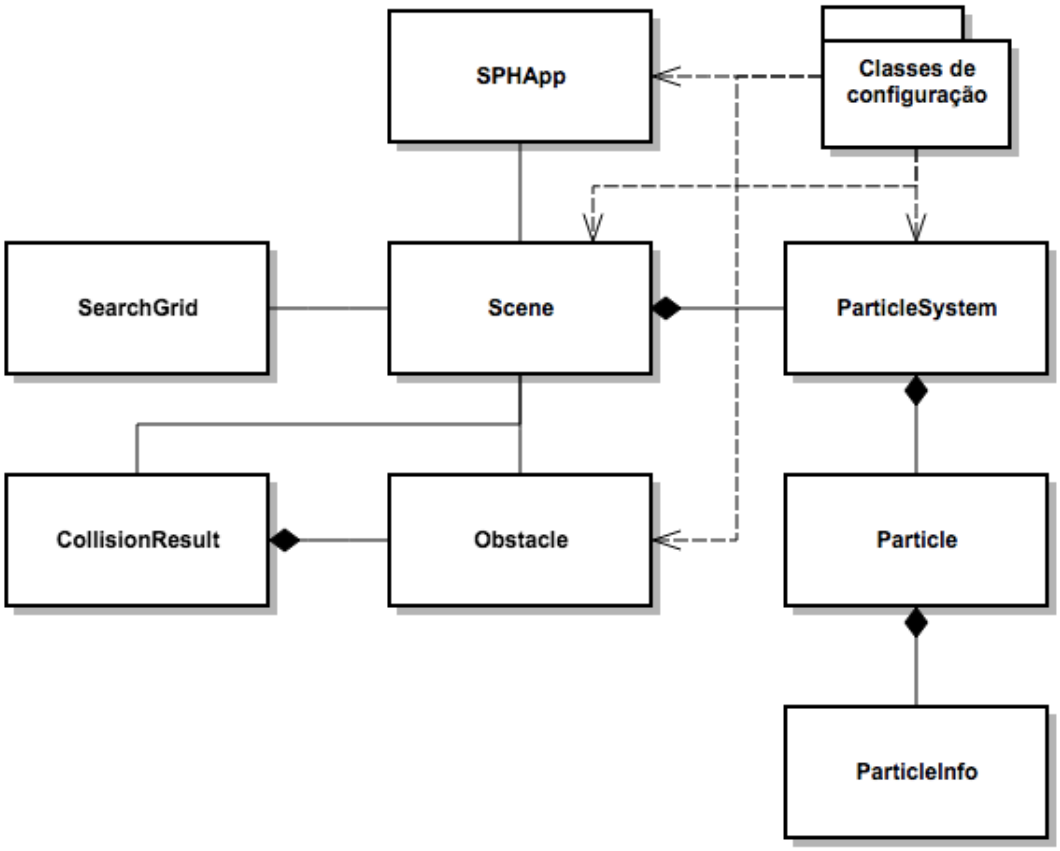
[22] Terzopoulos, D.; Platt, J.; Fleisher, K. Heating and melting deformable models (from goop to glop). *Graphics Interface '89* (Junho 1989), 219–226.

[23] Tonnensen, D. Modeling liquids and solids using thermal particles. *Graphics Interface '91* (Junho 1991), 255–262.

[24] Woods, A. *Flow in Porous Rocks - Energy and Environmental Applications*. Cambridge University Press, 2015.

[25] Zhu, Y.; Fox, P.; Morris, J. A pore-scale numerical model for flow through porous media. *International Journal for Numerical and Analytical Methods in Geomechanics 23*, 9 (Agosto 1999), 881–904.

APÊNDICE A – Diagrama de classes



APÊNDICE B – Descrição das classes

Classe	Descrição
CollisionResult	Armazena o resultado do teste de colisão de uma partícula com o interior do obstáculo que representa a rocha porosa.
Obstacle	Objeto que representa a rocha porosa através da qual o sistema de fluidos escoa.
Particle	Objeto que representa uma partícula da simulação através de sua posição.
ParticleInfo	Armazenado associado a um objeto Particle, armazena propriedades como velocidade, aceleração, densidade etc.
ParticleSystem	Representa o sistema de fluidos bifásico simulado através de um conjunto de objetos do tipo Particle associados a seus respectivos objetos ParticleInfo.
SPHApp	Classe principal, incializada pela ferramenta de processamento de malhas poligonais e responsável por dar início à simulação.
Scene	Objeto que representa a cena simulada, contendo o fluido (ParticleSystem), a rocha pelo qual ele escoa (Obstacle), além de diversos outros objetos auxiliares.
SearchGrid	Objeto que representa o *grid* de busca de partículas utilizado para otimizar a simulação.