Comparação dos Métodos PIC/FLIP e SPH para Simulação de Ondas Marítimas

Palavras chaves: computação gráfica, simulação de ondas marítimas, processamento gráfico

Autores:
Luc Joffily Ribas, Instituto de Computação, UNICAMP
Hélio Pedrini (Orientador), Instituto de Computação, UNICAMP

1 Introdução

Líquidos, como água, são responsáveis por complexos fenômenos visuais, que muitas vezes são desejáveis para trazer realismo para filmes e videogames. Dessa maneira, simulá-los é um desafio muito pertinente na computação gráfica e que é amplamente estudado. Nesse contexto, duas principais vertentes surgem [3], os métodos lagrangianos e os métodos eulerianos.

A abordagem lagrangiana envolve a representação do fluido como um sistema de partículas, na qual cada uma delas representa uma porção do fluido que possui posição, velocidade e, possivelmente, outros parâmetros. Uma das técnicas mais comuns para esse método é a Hidrodinâmica de Partículas Suavizadas (SPH), que foi inicialmente proposta por por Lucy et al. [7] e Gingold e Monaghan et al. [4] no campo da astrofísica e, posteriormente, introduzida a fluidos por Muller et al. [12]. Esse mecanismo envolve a atualização dos atributos das partículas por meio da interação com seus vizinhos dentro de um certo raio.

Na abordagem euleriana, as propriedades do fluido e como elas se alteram são analisadas em pontos fixos do espaço. Dessa maneira, um fluxo de fluido com partes quentes e partes frias teria a temperatura em um ponto variando constantemente, mesmo que a temperatura de cada partículas não esteja sendo alterada [3].

Há também métodos híbridos que utilizam características de ambas as abordagens lagrangiana e euleriana, de maneira que os dados são constantemente transferidos entre partículas e grades que englobam o espaço de interesse.

Nesse contexto, torna-se pertinente compreender as limitações de cada abordagem nas diferentes situações na qual o fluido pode ser submetido. O principal objetivo deste trabalho é comparar as vantagens e desvantagens de dois métodos, com foco na simulação de ondas marítimas. O primeiro, denominado Fluido Baseado em Posição (PBF), faz uso de princípios lagrangianos e conceitos provenientes do SPH para manter a densidade do fluido uniforme. O segundo é uma junção de métodos híbridos, denominados Partícula em Célula (PIC) e Fluido de Partículas Implícitas (FLIP), que utiliza partículas para mover o fluido e uma grade para garantir que o fluido permanece incompressível.

2 Embasamento Teórico

Como cada uma das simulações realizadas utiliza uma abordagem diferente para representação do fluido, há pouca interseção na derivação teórica de cada um dos métodos.

2.1 PBF

A Simulação de Fluidos baseada em PBF é derivada do princípio da Dinâmica Baseada em Posições [13], no qual as posições das partículas são atualizadas conforme as forças externas e funções de vínculo entre partículas. Nesse contexto, um corpo sólido, por exemplo, poderia ter um vínculo de distância fixa entre partícu-

las vizinhas, de maneira que, quando as forças externas provocam um desequilíbrio, é aplicada uma correção na posição. Normalmente, essa correção é feita a partir da resolução de um sistema de equações lineares utilizando métodos como Jacobi.

Formalmente, dado um vínculo C, para partículas com posição $\mathbf{p}$ as correções de posição $\Delta\mathbf{p}$ devem ser tais que:

$$C(\mathbf{p} + \Delta\mathbf{p}) = 0 \tag{1}$$

A partir de uma expansão em séries de Taylor[6], tem-se:

$$C(\mathbf{p} + \Delta\mathbf{p}) \approx C(\mathbf{p}) + \nabla_{\mathbf{p}} C(\mathbf{p}) \cdot \Delta\mathbf{p} = 0 \tag{2}$$

Tomando $\Delta\mathbf{p}$ na direção de maior mudança na função C, ou seja, na direção do gradiente, obtém-se:

$$\Delta\mathbf{p} = \lambda \nabla_{\mathbf{p}} C(\mathbf{p}) \tag{3}$$

Com λ sendo um escalar, que, substituindo pode ser optido por:

$$\lambda = -\frac{C(\mathbf{p})}{|\nabla_{\mathbf{p}} C(\mathbf{p})|^2} \tag{4}$$

Para o PBF, a equação de vínculo adotada é que a densidade entre as partículas deve ser constante [8], de maneira que, caso sejam afastadas por forças externas, a correção tende a juntá-las com as demais, enquanto que, caso estejam muito aglomeradas, elas tendem a se espalhar, mantendo assim, um fluido coeso. Dessa maneira, a densidade de uma partícula i é definida pelas posições de seus vizinhos, tal que seu vínculo é função do conjunto de vizinhos V, ou seja:

$$C_i(V) = \frac{\rho_i}{\rho_0} - 1 \tag{5}$$

em que ρ_0 é a densidade base e ρ_i é a densidade da partícula calculada utilizando o princípio do SPH[11]:

$$\rho_i = \sum_j m_j W(\mathbf{p}_i - \mathbf{p}_j, h) \tag{6}$$

em que W é uma função *kernel* [12] e h é o raio máximo entre vizinhos. A massa pode ser ignorada assumindo que ela é constante entre todas as partículas.

Com isso, pode-se determinar a variação da posição através da formulação do gradiente do SPH [10] e levando em conta a correção dos vizinhos, fazendo:

$$\Delta\mathbf{p}_i = \frac{1}{\rho_0} \sum_j (\lambda_i + \lambda_j) \nabla W(\mathbf{p}_i - \mathbf{p}_j, h) \tag{7}$$

Os autores ainda aplicam uma correção de pressão para evitar efeitos visuais de aglomeração excessiva de partículas e termos adicionais para levar em conta a viscosidade e vorticidade dos fluidos.

No decorrer desse processo, o passo mais custoso é a busca pelos vizinhos que, caso seja feita de maneira ingênua, pode acabar tendo custo $O(n^2)$. Portanto, uma grade foi implementada em placa gráfica (do inglês, *graphics processing unit* - GPU) utilizando um algoritmo baseado no trabalho de Hoetzlein [5]. Esse método envolve ordenar as partículas por posição e criar uma grade que contém os índices de início e fim das que ocupam uma certa região. Dessa maneira, pode-se acessar partículas próximas por meio dos índices na grade.

Para colisão, utilizou-se o método baseado no trabalho de Macklinem et al. [9], no qual a superfície dos sólidos é coberta com partículas rígidas que interagem com as partículas de fluido resolvendo colisões.

2.1.1 Resolvedor

Originalmente, os autores do PBF utilizaram o método de Jacobi para resolver os sistemas de equações necessários para calcular a correção de posição. No entanto, apesar desse método ser facilmente paralelizável para GPU, ele tem uma convergência mais lenta do que outros como Gauss-Seidel ou SOR [14]. Dessa maneira, desenvolvemos neste trabalho um novo algoritmo para essa etapa baseado em coloração.

O princípio é "colorir" as partículas de maneira a formar duas categorias, as vermelhas e as pretas. A partir disso, as vermelhas são atualizadas primeiro, seguidas das pretas. Com isso, a segunda categoria interage com as partículas já atualizadas da outra, o que acelera a convergência do método. Vale ressaltar que essa coloração foi realizada para utilizar a GPU da melhor maneira possível, atualizando as partículas que possuem a mesma cor em uma mesma chamada.

2.2 PIC/FLIP

Em simulações eulerianas, um dos maiores desafios é modelar o movimento do fluido através dos pontos fixos no espaço [3]. Uma maneira de fazer isso é utilizar partículas marcadoras, cujo propósito é representar esse movimento. Portanto, esses métodos híbridos exigem uma constante troca de atributos entre a grade e as partículas.

O primeiro passo para essa simulação é avançar as partículas, o que é feito com o método Runge-Kutta de segunda ordem que garante mais precisão na integração do que o método de Euler. Com isso, os pontos fixos recebem como valor de velocidade a média das velocidade das partículas próximas ponderadas por 1 menos o valor da distância ponto-partícula.

O próximo passo é tornar o fluido incompressível. Isso envolve garantir que o divergente do campo vetorial de velocidades do fluido é zero e que a velocidade do fluido em contato com uma uma borda sólida é igual à velocidade dessa borda. O passo inicial para esse processo vem do fato de que a velocidade na $(n+1)$-ésima iteração é:

$$\vec{u}^{n+1} = \vec{u} - \Delta t \frac{1}{\rho} \nabla p \tag{8}$$

em que p é a pressão, ρ é a densidade do fluido (constante nesse caso) e Δt é o tempo da iteração.

A partir disso, pode-se calcular calcular as pressões em cada célula de maneira que:

$$\nabla \cdot \vec{u}^{n+1} = 0 \tag{9}$$

Esse cálculo é realizado através da resolução de um sistema linear positivo definido esparso, cujo tamanho é o quadrado no número de células na grade. Em virtude disso, métodos exatos como eliminação de Gauss, acabam sendo muito custosos [3]. Nesse contexto, utiliza-se o Gradiente Conjugado (CG), que é um método iterativo específico para esse tipo de matriz. Além disso, pode ser adicionado um precondicionador, que acelera o processo tornando a matriz mais próxima da identidade. Para isso, foi utilizado o método descrito por Ament et al. [1], que é conveniente para uso em GPU. Com isso, o método CG totalmente paralelo foi implementado como resolvedor.

A nova pressão calculada é então aplicada nas velocidades, as quais são transferidas às partículas. Apenas atribuir a velocidade da célula a uma partícula corresponde ao método PIC. Contudo, esse processo gera perdas de detalhes no fluido. Dessa maneira, uma combinação dele com FLIP é realizada, a qual adiciona à velocidade da partícula a diferença entre a velocidade atual e a antiga [3]. Tipicamente, utiliza-se 95% de FLIP e 5% de PIC.

Para a interface sólido-fluido, um campo de distâncias com sinal foi implementado, o qual gera uma grade com a distância de cada célula à superfície do objeto. Isso é feito calculando a distância das células próximas aos triângulos da malha do objeto rígido e propagando essa informação aproximada para toda a grade. Por fim, os sinais de pontos no interior da malha são tornados negativos [3].

2.3 Renderização

Como em ambos os métodos as partículas representam as posições nas quais o fluido está presente, a abordagem de renderização utilizada foi a Renderização de Fluido em Espaço de Tela [15] para ambos os casos.

As etapas visuais dessa técnica podem ser observadas na Figura 1. O primeiro passo é calcular as profundidades das partículas em relação à tela, que são então borradas para gerar uma superfície uniforme. Vale notar que, neste passo, foi utilizada a correção proposta por Bagar et al. [2]. A partir desses valores, as normais com diferenças parciais são então calculadas.

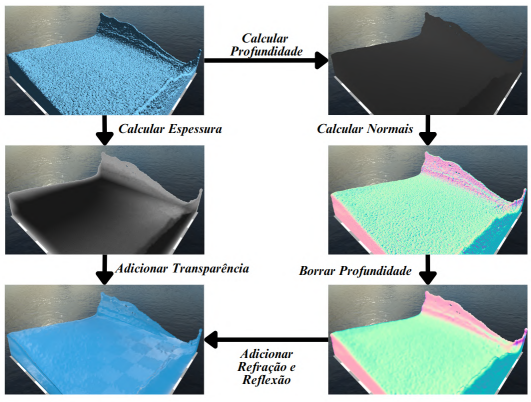


Figura 1: Etapas para Renderização de Fluido em Espaço de Tela.

Paralelamente, a espessura de fluido em cada direção é calculada e armazenada. Juntando es-

sas etapas, a difração, a refração e a iluminação fresnel são adicionadas para finalizar a renderização. Esse processo é realizado inteiramente em tempo real, em conjunto com a simulação.

3 Resultados

Apesar do foco inicial da pesquisa ser relacionado à geração de ondas e ao desenvolvimento do novo algoritmo de iteração para o PBF, também foram desenvolvidas cenas padronizadas para utilização em simulações de fluido diversas, com o propósito de verificar com elas se comportam em diferentes contextos.

As simulações foram desenvolvidas na linguagem de programação C++, utilizando a API gráfica Vulkan e executadas em uma NVIDIA GTX 1650 e Intel Core i7 10^a geração.

3.1 Convergência do PBF

Para testar a convergência do novo método para o cálculo das correções de posição do PBF, uma comparação com o Jacobi original foi realizada, variando-se o número de iterações do resolvedor, em uma simulação com 500.000 partículas com passos de 16 milissegundos ao longo de 800 iterações da simulação. O ponto de convergência (PC) foi definido como sendo alcançado quando o erro relativo da iteração atual e as 50 anteriores é menos de 0,5% e essa condição se mantém valida para todos os pontos futuros.

Os resultados, apresentados na Tabela 1, mostram que nosso método foi capaz de acelerar a convergência com pequena perda de desempenho, especialmente para um número baixo de iterações do resolvedor. Dessa maneira, ele pode se apresentar como alternativa para simulações em tempo real nas quais 2 a 4 iterações são recomendadas [8].

Tabela 1: Registro do Ponto de Convergência (PC) e do tempo médio em milissegundos por *frame* com o método Jacobi e com o novo método.

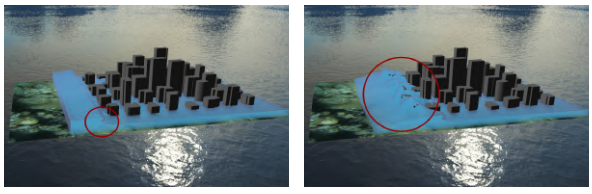
Iterações do	Jacobi		Proposto	
Resolvedor	Tempo	PC	Tempo	PC
2	23.73	737	24.54	636
3	28.22	603	29.35	449
4	32.80	439	34.47	291
5	37.44	287	39.65	168
6	42.31	251	44.68	165
7	47.13	165	49.99	163
8	51.93	162	55.23	161

3.2 Ondas

Para a geração de ondas, em ambos os métodos, o limite da simulação é alterado com uma certa velocidade, empurrando as partículas e fazendo uso das condições de borda já presentes na simulação. Isso resulta em acúmulos de líquido se movendo em uma mesma direção, o que cria um efeito similar às ondas que vêm do mar. Para testar e comparar como cada simulação se comporta nessa situação, um cenário foi criado com 200.000 partículas e com uma cidade em uma extremidade, para que a colisão ocorra e as ondas se quebrem.

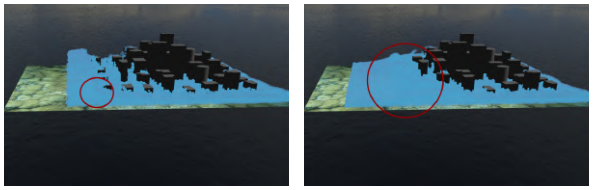
Vale notar que a resolução da grade do PIC/FLIP é de 125×63×75 pixels, que são valores muito maiores do que o necessário para bons efeitos visuais da simulação. Entretanto, para que o fluido tenha uma interação mais realista com a cidade, o tamanho das células teve que ser reduzido. Apesar disso, em ambos os casos, o tempo médio por *frame* dessa simulação foi de 24.27 milissegundos, enquanto para o PBF foi de 25.22 milissegundos.

As Figuras 2 e 3 apresentam os resultados visuais das cenas aplicadas ao PBF e ao PIC/FLIP, respectivamente. Pode-se observar que a onda formada na imagem da Figura 2(a) possui uma crista mais bem definida quando comparada com a imagem da Figura 3(a), como destacado na região em vermelho. Além disso, a imagem da Figura 3(b) não apresenta um *splash* bem definido e acaba "engolindo" os prédios, enquanto o espalhamento da onda é mais acentuado na imagem da Figura 2(b).



(a) Onda gerada (b) Contato com prédios

Figura 2: Simulação com PBF.



(a) Onda gerada (b) Contato com prédios

Figura 3: Simulação com PIC/FLIP.

4 Conclusões

Ambas as simulações foram capazes de gerar ondas similares à natureza. Em nossos experimentos, entretanto, a simulação lagrangiana foi capaz de trazer mais detalhes que enriquecem os efeitos visuais.

Os dois métodos foram implementados para funcionamento em tempo real. Dessa forma, os resultados podem ser diferentes para simulações *offline*, que são mais lentas, mas são capazes de fornecer mais detalhes.

Agradecimentos

Os autores gostariam de agradecer ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela concessão de bolsa de Iniciação Científica.

Referências

[1] M. Ament, G. Knittel, D. Weiskopf, and W. Straßer. A parallel preconditioned conjugate gradient solver for the poisson problem on a multi-gpu platform. *18th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 583–592, Feb. 2010.

[2] F. Bagar, D. Scherzer, and M. Wimmer. A Layered Particle-Based Fluid Model for Real-Time Rendering of Water. *Computer Graphics Forum*, 29:1383–1389, 06 2010.

[3] R. Bridson. *Fluid Simulation*. A. K. Peters, Ltd., Estados Unidos, 2008.

[4] R. A. Gingold and J. J. Monaghan. Smoothed Particle Hydrodynamics: Theory and Application to Non-Spherical Stars. *Monthly Notices of the Royal Astronomical Society*, 1977.

[5] R. Hoetzlein. Fast Fixed-Radius Nearest Neighbors: Interactive Million-Particle Fluids, 2014.

[6] M. Köster and A. Krüger. Adaptive Position-Based Fluids: Improving Performance of Fluid Simulations for Real-Time Applications. *International Journal of Computer Graphics & Animation*, 6(3): 1–16, June 2016.

[7] L. B. Lucy. A Numerical Approach to the Testing of the Fission Hypothesis. *Astronomical Journal*, 1977.

[8] M. Macklin and M. Müller. Position-Based Fluids. *ACM Transactions on Graphics*, 32 (4), July 2013.

[9] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim. Unified Particle Physics for Real-Time Applications. *ACM Transactions on Graphics*, 33(4), July 2014.

[10] J. Monaghan. SPH without a Tensile Instability. *Journal of Computational Physics*, 159:290–311, 4 2000.

[11] J. Monaghan. Smoothed Particle Hydrodynamics. *Reports on Progress in Physics*, 68:1703, 7 2005.

[12] M. Müller, D. Charypar, and M. Gross. Particle-Based Fluid Simulation for Interactive Applications. In *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, page 154–159, Goslar, Alemanha, 2003.

[13] M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. Position-Based Dynamics. *Journal of Visual Communication and Image Representation*, 18(2):109–118, 2007.

[14] O. Osadcha. Comparison of Two Stationary Iterative Methods. Technical report, Faculty of Applied Mathematics, Silesian University of Technology, Gliwice, Polônia, 2024.

[15] W. J. van der Laan, S. Green, and M. Sainz. Screen Space Fluid Rendering with Curvature Flow. In *Symposium on Interactive 3D Graphics and Games*, page 91–98, Boston, MA, Estados Unidos, 2009.