Animação de Fluidos 3D via Autômatos Celulares do Tipo FHP

Sicilia F. Judice e Gilson A. Giraldi

[1]National Laboratory of Scientific Computing
Ave Getúlio Vargas, 333, 25651-075, Petrópolis, RJ, Brasil
{sicilia,gilson}@lncc.br

Abstract. *In this work, we propose a three dimensional fluid animation model, based on a lattice cellular automata and simple interpolation methods. The fluid is rendering by a volume rendering technique, the single scattering method, available in the PBRT library. The obtained model is used to animated a set of macroscopic particles following the particle tracing methodology. We also discuss details of the implementation of the proposed model, and a software developed for fluid animation that incorporates this model.*

Resumo. *Neste trabalho, propomos um modelo para animação de fluidos tridimensionais, baseado em um autômato celular do tipo lattice e em métodos simples de interpolação. O rendering do fluido é executado através de uma técnica de render volumétrico, o método single scattering, disponível na biblioteca PBRT. O modelo obtido é utilizado para animar um conjunto de partículas macroscópicas seguindo a metodologia do traçado de partículas (particle tracing). Discutimos também detalhes da implementação do modelo proposto, assim como um aplicativo desenvolvido para animação de fluidos que incorpora este modelo.*

1. Introdução

Nas últimas décadas, observou-se um interesse crescente por aplicações de técnicas de dinâmica de fluidos computacional (DFC) na geração de efeitos visuais para a indústria cinematográfica e de jogos para computador [Witting 1999, DreamWorks 1998, Giraldi et al. 2005a].

As pesquisas em animação de fluidos se dividem basicamente em três etapas. Primeiramente, temos a busca de novos modelos em DFC que sejam mais eficientes do ponto de vista da computação gráfica. Uma vez resolvidas numericamente as equações de fluidos, passa-se à fase de rendering, onde técnicas de computação gráfica são aplicadas sobre os campos gerados, com o objetivo de criar efeitos visuais, tais como transparência e imagens refletidas na superfície de um líquido. Finalmente, as técnicas adotadas devem ser incorporadas a um software, com interfaces gráficas convenientes, tornando possível o uso destes recursos por artistas gráficos e animadores (ver [Witting 1999], como exemplo). Neste trabalho vamos nos concentrar na etapa de modelagem do fluido e demonstrar a potencialidade de um aplicativo em desenvolvimento para animação de fluidos $3D$.

Os métodos numéricos em DFC, tais como Diferenças Finitas e Elementos Finitos [Hughes 1987], procuram descrever um sistema contínuo através da discretização das equações que o representam. Uma alternativa ao uso dos Métodos Numéricos é o uso de

Autômatos Celulares [Wolfram 1994, Sarkar 2000, Kari 2005]. Autômatos celulares são formados por um conjunto de células interconectadas localmente umas as outras. A cada uma dessas células está associado um estado, o qual é atualizado a cada instante de acordo com uma regra que considera os estados das células vizinhas.

Enquanto a modelagem tradicional em DFC tenta representar um meio contínuo partindo de variáveis macroscópicas e equações diferenciais parciais, a modelagem via autômatos celulares segue o caminho inverso, ou seja, a partir da descrição microscópica baseada em regras simples, procura-se obter o comportamento macroscópico [Chopard and Droz 2005], sem, no entanto, resolver explicitamente nenhum sistema de equações diferenciais parciais. O comportamento macroscópico é obtido via simulação computacional, baseada no conjunto de regras que regem o sistema, e interpolações simples. Assim, estas simulações são, em geral, eficientes do ponto de vista computacional. Esta é uma vantagem que pretendemos explorar neste trabalho para a geração de efeitos visuais envolvendo fluidos $3D$.

Dentre a variedades de autômatos do tipo (LGA) optamos pelo modelo FHP pela sua simplicidade e eficiência computacional [Chopard and Droz 2005, Frisch et al. 1986]. O FHP foi desenvolvido para simular fluidos bidimensionais. Neste trabalho, propomos modelo de simulação de fluidos $3D$, baseado no FHP e em métodos simples de interpolação. O rendering do fluido é executado através de uma técnica de render volumétrico, o método *single scattering*, disponível na biblioteca PBRT (http://www.pbrt.org/). O modelo obtido é utilizado para animar um conjunto de partículas macroscópicas seguindo a metodologia do traçado de partículas (*particle tracing*). Discutimos também detalhes da implementação do modelo proposto, assim como um aplicativo desenvolvido para animação de fluidos que incorpora este modelo.

O texto está organizado da seguinte forma. Na seção 2 apresentamos uma revisão das técnicas de animação de fluidos. Na seção 3 descrevemos o FHP e o método de simulação proposto. Na seção 4 explicamos detalhes de implementação e apresentamos os resultados obtidos. Na seção 5 apresentamos as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Nesta seção faremos uma revisão dos métodos para animação computacional de fluidos diretamente relacionados ao nosso trabalho. Animação computacional é uma subárea da computação gráfica que procura reproduzir a dinâmica dos objetos de uma cena com o realismo necessário [Kerlow 1996, Taylor 1996, Watt and Watt 1991]. Para isso ela usa técnicas de rendering, modelagem geométrica e modelagem de sistemas físicos, dentre outras [Iglesias 2004, Nealen et al. 2005, Deusen et al. 2004]. No caso específico da animação de fluidos, este processo envolve: modelagem geométrica da cena; modelo matemático dos fluidos em questão; simulação computacional deste modelo; e rendering.

As técnicas de modelagem geométrica podem envolver modelos poligonais da fronteira do domínio por onde o fluido irá escoar (paredes, objetos rígidos, etc), superfícies paramétricas e texturas para a representação da superfície visível do fluido e malhas regulares ou irregulares para discretização do domínio e solução numérica das equações.

Os métodos encontrados na literatura para animação de fluidos via modelos de DFC, são fundamentados nas equações de Navier-Stokes, com técnicas

de discretização baseadas em diferenças finitas implícitas [Stam 1999] e explícitas [Foster and Metaxas 1997], bem como em métodos Lagrangeanos tais como Método das Características [Stam 1999], *Smoothed Particle Hydrodynamics* (SPH) [Liu et al. 2003, Müller et al. 2003] e *Moving-Particle Semi-Implicit* (MPS) [Premoze et al. 2003]. Os campos gerados nas simulações devem receber um tratamento gráfico via métodos de visualização científica e computação gráfica, para a geração da seqüencia de imagens.

O trabalho de Foster e Metaxas [Foster and Metaxas 1997] propõe um modelo para gases aquecidos fundamentado em uma versão simplificada da equação de Navier-Stokes, resolvida pelo método de diferenças finitas. Este trabalho pode ser considerado o precursor ao algoritmo de Jos Stam [Stam 1999], o qual propõe uma solução para o custo computacional elevado do método de Foster e Metaxas para malhas com alta resolução e/ou velocidades altas. O método de Foster e Metaxas foi também o precursor do trabalho de Patrick Witting [Witting 1999], o qual considera um modelo matemático para fluidos (gases) bem mais completo que os dois anteriores. Neste caso, o fluido (gás) é tratado como compressível e o modelo matemático envolve também equações termodinâmicas. Este método usa diferenças finitas no espaço e Runge-Kutta para interação no tempo.

No trabalho [Müller et al. 2003], encontramos resolução das equações de Navier-Stokes via SPH. Uma vantagem deste método com relação aos anteriores é sua independência de malhas previamente definidas. Mais recentemente, demonstramos as vantagens de modelos do tipo LGCA para animação de fluidos bidimensionais, bem como para simulação de fluxo superficial da água em modelos digitais de terrenos [Giraldi et al. 2005b, Xavier 2006].

Na fase de rendering, os trabalhos citados fazem uso de render volumétrico [Foster and Metaxas 1997], *splatting* e *marching-cubes* [Müller et al. 2003]. Para obter maior nível de realismo visual, tem-se utilizado também métodos de iluminação global, tais como *path tracing*, *path tracing* bidirecional [Heckbert 1990], *Metropolis light transport* [Veach and Guibas 1997] e mapa de fótons (*photon mapping*) [Jensen 1996].

3. FHP

Introduzido por Frisch, Hasslacher and Pomeau [Frisch et al. 1986] em 1986, o FHP é um modelo de um fluido bidimensional que pode ser visto, em uma escala microscópica, como uma abstração desse fluido. O modelo FHP descreve o movimento das partículas ao longo de um espaço discreto, assim como as colisões entre elas. O espaço é discretizado em uma malha hexagonal.

As partículas do FHP se movem em passos discretos de tempo, com velocidade constante em módulo, direcionada ao longo de uma das seis direções da malha. A dinâmica ocorre de maneira que não mais do que uma partícula ocupa o mesmo nó no mesmo instante de tempo com a mesma velocidade. Tal restrição é chamada de *princípio de exclusão*, e assegura que seis variáveis booleanas para cada nó da malha são suficientes para representar as microdinâmicas.

O módulo da velocidade é tal que, em um dado instante de tempo, cada partícula percorre uma unidade de espaço da malha, alcançando um nó vizinho. Quando exatamente duas partículas ocupam o mesmo nó com velocidades opostas, ambas são desviadas em um ângulo de 60 graus, de forma que após a colisão tenha-se uma nova configuração

também com momento nulo. O desvio pode ocorrer para a direita ou para a esquerda, como mostrado na Figura 1. Por questões de simetria, as duas possibilidades de desvio são escolhidas aleatoriamente, com igual probabilidade.

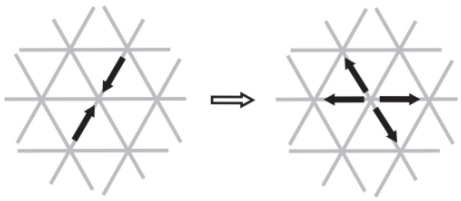


Figure 1. Colisão entre duas partículas no FHP.

Quando exatamente três partículas colidem com um ângulo de 120 graus entre elas, elas retornam para seus nós de origem (fazendo com que o momento após a colisão seja zero, sendo portanto conservado). As duas colisões descritas acima são necessárias para evitar leis de conservação extras. Para todas as outras configurações não mencionadas, não ocorre colisão e as partículas seguem seus caminhos como se fossem transparentes entre si.

A microdinâmica completa do modelo FHP pode ser expressa por uma equação de evolução para os números de ocupação definidos como o número, $n_i\left(\mathbf{r},t\right)$, de partículas entrando no nó $\mathbf{r}$ no instante t com velocidade na direção $\vec{c}_i$

$$\vec{c}_i = \left(\cos\frac{2\pi i}{6}, \sin\frac{2\pi i}{6}\right), \tag{1}$$

onde $i = 1, 2, \ldots, 6$ representa uma das seis direções da malha.

Os números n_i podem ser 0 ou 1. É definido também o passo de tempo como Δ_t e o deslocamento entre os nós da malha como Δ_r. Desta forma, as seis possíveis velocidades $\vec{v}_i$ das partículas estão relacionadas com suas direções de movimento por

$$\vec{v}_i = \frac{\Delta_r}{\Delta_t}\vec{c}_i. \tag{2}$$

A microdinâmica de um LGCA é escrita como

$$n_i\left(\mathbf{r} + \Delta_r\vec{c}_i, t + \Delta_t\right) = n_i\left(\mathbf{r},t\right) + \Omega_i\left(n\left(\mathbf{r},t\right)\right) \tag{3}$$

onde Ω_i é chamado de termo de colisão [Chopard and Droz 2005].

É proposto uma extensão para o 3D, usando o modelo bidimensional do FHP explicado acima. Na prática, o sistema de uma dada regra de autômato celular não pode lidar com uma malha infinita, ela deve ser finita e ter fronteiras [Chopard and Droz 2005]. Assim, primeiro é preciso definir um domínio no espaço tridimensional. A proposta consiste em distribuir regularmente planos ao longo dos eixos x e z, como mostrado na Figura 2 (a). Cada um desses planos é um sistema cuja regra de autômato celular é o modelo FHP bidimensional.

Uma vez feita a simulação do FHP 2D em cada plano separadamente, é feito uma interpolação simples para gerar um fluxo macroscópico $2D$ em cada plano. Neste

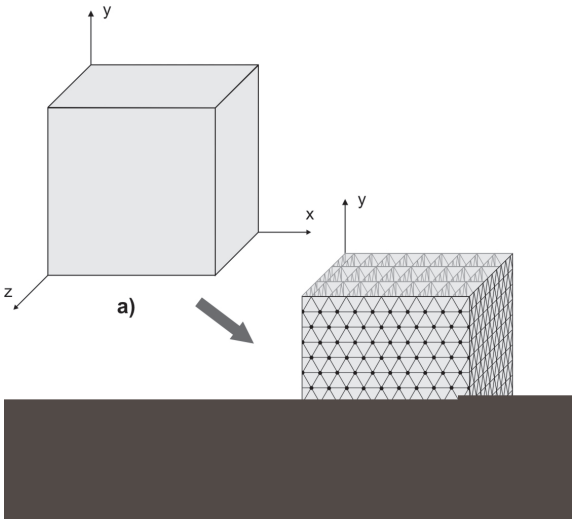


Figure 2. (a) Domínio no espaço 3D e (b) distribuição dos planos FHP 2D.

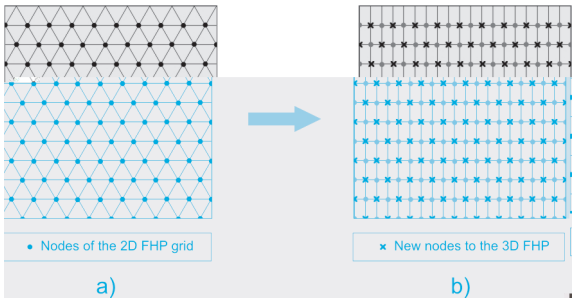


Figure 3. Extensão (a) da malha do FHP para gerar (b) uma malha retangular.

passo, novos nós são adicionados à malha do FHP com o intuito de completar uma malha retangular em cada plano, como mostra a Figura 3 (b).

Seguindo a definição usual da mecânica estatística, computa-se a densidade macroscópica em cada nó (x_i, y_i, z_i) do plano $x = x_i$ através da expressão:

$$\rho_x\left(x_i, y_i, z_i, t\right) = \sum_{j \in V} \sum_{k=1}^{6} n_k\left(x_j, y_j, z_j, t\right), \tag{4}$$

onde V é uma vizinhança do ponto (x_i, y_i, z_i).

Uma expressão análoga pode ser usada para o plano $z = z_i$. Agora é preciso renderizar um fluxo macroscópico $3D$. Deve-se observar que cada nó (x_i, y_i, z_i) na Figura 2 (a) pertence aos planos $x = x_i$ e $z = z_i$. Dessa forma, a densidade $3D$ pode finalmente ser obtida através de uma média simples dos valores correspondentes nos planos FHP:

$$\rho\left(x_i, y_i, z_i, t\right) = \frac{\rho_x\left(x_i, y_i, z_i, t\right) + \rho_z\left(x_i, y_i, z_i, t\right)}{2}. \tag{5}$$

Computa-se também o campo de velocidades através de amostragens macroscópicas do comportamento do FHP. Para tanto, utilizamos um conjunto de partículas de teste cujas posições iniciais no domínio 3D são aleatórias. Para cada

partícula macroscópica p_{mac}, toma-se uma vizinhança $V\left(p_{mac}, raio\right)$, onde $raio$ é um raio de influência, e computa-se a soma das velocidades $\vec{c}_i$ (equação (1)) das partículas microscópicas p_{mic} interiores à vizinhança:

$$\vec{v}\left(p_{mac}\right) = \sum_{p_{mic} \in V(p_{mac}, raio)} \vec{c}_i\left(\vec{r}, t\right). \tag{6}$$

Uma vez calculado o campo de velocidades, dada uma partícula de teste p, na posição $\vec{x}\left(p, t\right)$ no instante t, a posição da partícula p no instante seguinte é dada por:

$$\vec{x}\left(p, t + \Delta t\right) = \vec{x}\left(p, t\right) + \sigma \vec{v}(p), \tag{7}$$

onde σ é uma constante referente ao passo macroscópico no tempo.

4. Implementação e Resultados

O modelo proposto na seção 3 para simulação de fluidos $3D$ foi implementado em linguagem C/C++, com a metodologia Orientada à Objetos. A visualização foi implementada utilizando o *OpenGL* [Shreiner 2004], que é uma *API (Application Programmers Interface)* aberta para o desenvolvimento de aplicações gráficas, independente de hardware e que pode ser incorporada a qualquer sistema de janelas (*MS Windows*, *X Window*, etc.).

Para a interface com o ambiente gráfico usamos a biblioteca *GLUT (OpenGL Utility Toolkit)* [Kilgard 1996, Baker 2006], que possui funções para gerenciamento de janelas, geração de vários objetos gráficos 3D e dispositivos de entrada de dados. Uma grande vantagem da GLUT é permitir o uso de todas as funções gráficas do OpenGL e ainda tornar padronizado o acesso a características específicas de cada ambiente de janelas disponíveis para sistemas operacionais como Unix e Windows.

A interface gráfica, constituída de botões, *checkboxes*, teclas de rádio, entre outros, foi desenvolvida com a biblioteca *GLUI* [Rademacher 1999, Stewart 2006], uma extensão da GLUT e, portanto, independente do sistema usado. Para a etapa de rendering usou-se a biblioteca PBRT (http://www.pbrt.org/).

Todas as ferramentas de software utilizadas no processo de desenvolvimento são multi-plataforma e de código não proprietário. Dessa forma, o programa desenvolvido pode ser executado em qualquer ambiente com suporte para OpenGL. Em nosso caso, tal programa foi desenvolvido e testado em Windows XP.

A Figura 4 mostra o aplicativo em desenvolvimento. Nela vemos a janela de visualização, à esquerda, onde podemos ver a simulação sobre vários aspectos: tornar visível a limitação do domínio, as partículas microscópicas, as macroscópicas, os vetores de velocidade, dentre outras. E temos também a janela de comando, à direita, por onde se torna a possível a comunicação do usuário com o aplicativo.

Os resultados gerados usaram o modelo FHP 3D proposto na seção 3. Foram visualizados os campos de densidade e de velocidade. Todos os resultados foram gerados em um domínio $3D$ de dimensão $20 \times 20 \times 20$ com planos $120 \times 120 \times 120$ distribuídos ao longo dos eixos x e y. Os vídeos correspondentes aos resultados obtidos podem ser encontrados em `http://www.lncc.br/~sicilia/assuntosAfins_pesquisa.htm`.

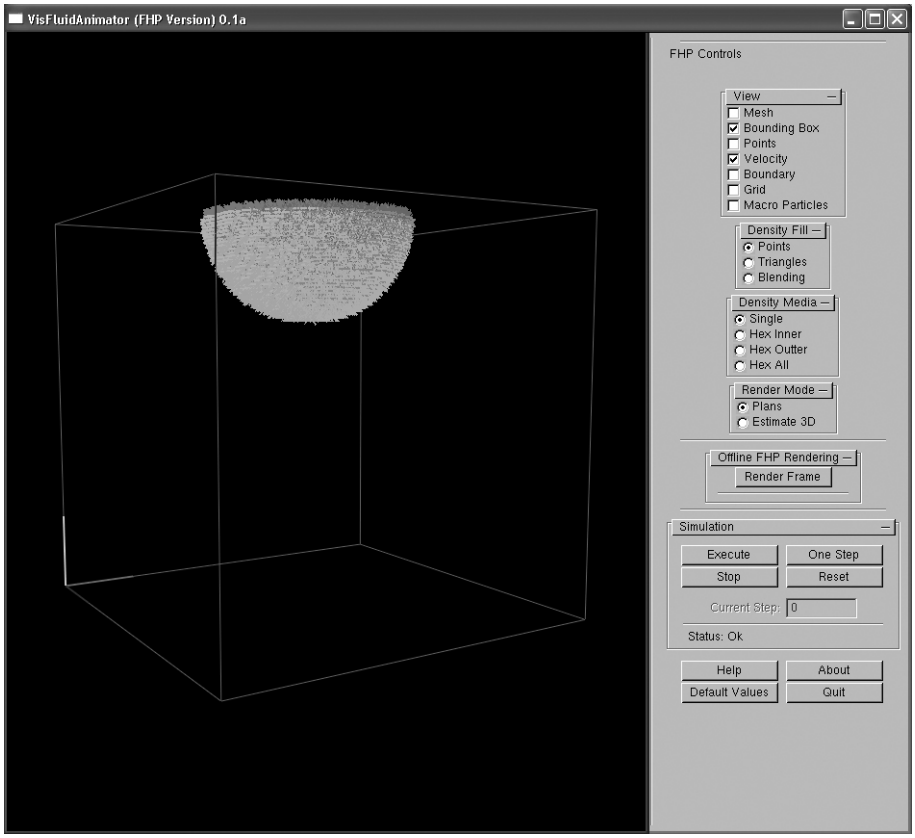
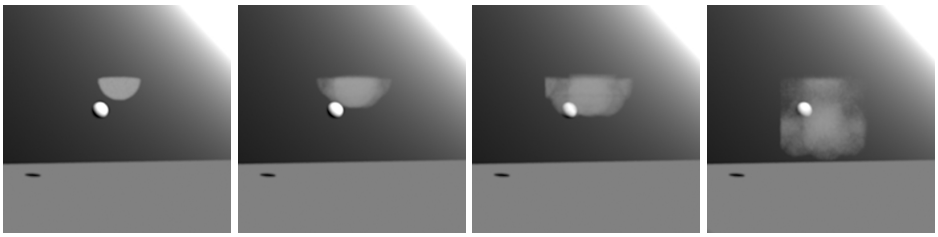


Figure 4. Screenshot do aplicativo em desenvolvimento.

4.1. Visualização do Campo de Densidade

Para visualização do campo de densidade usamos uma configuração inicial das partículas microscópicas do FHP como mostrado na Figura 5 (a): uma semi-esfera no topo do domínio. O processo de inicialização ocorre da seguinte forma: primeiro, os nós pertencentes à semi-esfera são detectados; então, para cada nó, o algoritmo escolhe aleatoriamente a quantidade de partículas microscópicas e suas direções. Para destacar os efeitos de transparência posicionamos uma esfera sólida externa ao domínio de evolução do gás.



(a) (b) (c) (d)

Figure 5. Configuração do volume de gás com uma esfera sólida externa ao domínio do gás: (a) no instante de tempo inicial; (b) depois de 10; (c) 20 e (d) 70 passos de simulação.

4.2. Visualização do Campo de Velocidade

A configuração inicial das partículas microscópicas do FHP ocorre como mostrado na Figura 6 (a): todas são inicializadas com suas velocidades seguindo a mesma direção. À

medida que a simulação evolui, novas partículas microscópicas são geradas na malha do FHP, alimentando o fluxo do gás.

Para melhor visualização do campo de velocidade, usamos um conjunto de 6 mil partículas macroscópicas (representadas pelas pequenas esferas) posicionadas aleatoriamente numa semi-esfera no domínio. Ao longo da evolução essas partículas ocupam novas posições de acordo com a equação 7, que depende do campo de velocidade e de uma constante referente ao passo macroscópico. Para gerar os resultados mostrados na Figura 6 usamos $\sigma = 0,0015$ e um raio de influência de $3,0$ para calcular o campo de velocidade. O tempo em média para simular e renderizar cada frame foi de $6,172$ minutos.

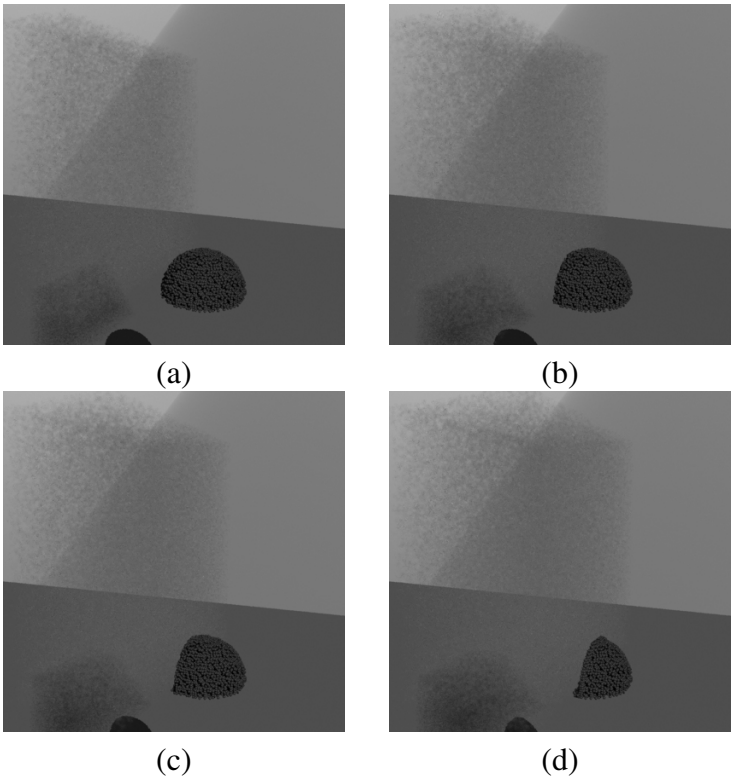


Figure 6. Partículas macroscópicas sob influência do campo de velocidade: (a) no instante de tempo inicial; (b) depois de 20 passos; (c) depois de 25 passos; e (d) depois de 30 passos de simulação.

5. Conclusão e Trabalhos Futuros

Neste trabalho foi apresentado um modelo do tipo LGCA para a animação de fluidos $3D$ para computação gráfica. Nosso trabalho é motivado pelo interesse crescente por aplicações de técnicas de simulação de fluidos na geração de efeitos visuais. Foram também discutidos detalhes de implementação de um aplicativo para animação de fluidos em desenvolvimento.

Como trabalhos futuros, pretendemos implementar técnicas de iluminação global, tais como mapa de fótons [Jensen 1996], explorando possíveis otimizações para diminuir o custo computacional da etapa de rendering. Por outro lado, pretendemos comparar o método proposto com o *Lattice Boltzman* [Benzi et al. 1992] para a etapa de simulação do fluido.

References

[Baker 2006] Baker, S. (2006). *The Freeglut Project*. http://freeglut.sourceforge.net/index.php. Último acesso em 25/06/2008.

[Benzi et al. 1992] Benzi, R., Succi, S., and Vergassola, M. (1992). The lattice boltzmann equation: Theory and applications. *Physics Reports*, 222:145–197.

[Chopard and Droz 2005] Chopard, B. and Droz, M. (2005). *Cellular Automata Modeling of Physical Systems*. Cambridge University Press.

[Deusen et al. 2004] Deusen, O., Ebert, D. S., Fedkiw, R., Musgrave, F. K., Prusinkiewicz, P., Roble, D., Stam, J., and Tessendorf, J. (2004). The elements of nature: Interactive and realistic techniques. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*.

[DreamWorks 1998] DreamWorks (1998). Prince of egypt.

[Foster and Metaxas 1997] Foster, N. and Metaxas, D. (1997). Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 181–188. ACM Press/Addison-Wesley Publishing Co.

[Frisch et al. 1986] Frisch, U., Hasslacher, B., and Pomeau, Y. (1986). Lattice-gas automata for the navier-stokes equation. *Phys. Rev.*, 56:1505.

[Giraldi et al. 2005a] Giraldi, G. A., JR., A. L. A., Oliveira, A. A. F., and Feijóo, R. A. (2005a). Animação de fluidos via técnicas de visualização científica e mecânica computacional. Technical report, Laboratório Nacional De Computação Científica, Petrópolis, RJ, Brasil.

[Giraldi et al. 2005b] Giraldi, G. A., Xavier, A. V., Apolinario, A. L., and Rodrigues, P. S. (2005b). Lattice gas cellular automata for computational fluid animation. Technical report, National Laboratory of Scientific Computing.

[Heckbert 1990] Heckbert, P. S. (1990). Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90*, pages 145–154, Dallas, TX, USA.

[Hughes 1987] Hughes, T. J. R. (1987). *The Finite Element Method: Linear Static And Dynamic Finite Element Analysis*. Prentice-Hall, Inc.

[Iglesias 2004] Iglesias, A. (2004). Computer graphics for water modeling and rendering: a survey. *Future Gener. Comput. Syst.*, 20(8):1355–1374.

[Jensen 1996] Jensen, H. W. (1996). Global illumination using photon maps. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 21–30, London, UK. Springer-Verlag.

[Kari 2005] Kari, J. (2005). Theory of cellular automata: A survey. *Theoretical Computer Science*, 334(1):3–33.

[Kerlow 1996] Kerlow, I. V. (1996). *The Art Of 3-D Computer Animation And Imaging*. Van Nostrand Reinhold, New York, NY.

[Kilgard 1996] Kilgard, M. J. (1996). *The OpenGL Utility Toolkit (GLUT): Programming Interface*. http://www.opengl.org/resources/libraries/glut/glut-3.spec.pdf. Último acesso em 12/07/2008.

[Liu et al. 2003] Liu, M., Liu, G., and Lamb, K. (2003). Constructing smoothing functions in smoothed particle hydrodynamics with applications. *Journal Of Computational And Applied Mathematics*, 155.

[Müller et al. 2003] Müller, M., Charypar, D., and Gross, M. (2003). Particle-based fluid simulation for interactive applications. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 154–159, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Nealen et al. 2005] Nealen, A., Muller, M., Keiser, R., Boxerman, E., and Carlson, M. (2005). Physically based deformable models in computer graphics. In *Eurographics 2005, State of The Art Report (STAR)*.

[Premoze et al. 2003] Premoze, S., Tasdizen, T., Bigler, J., Lefohn, A., and Whitaker, R. (2003). Particle-based simulation of fluids. In *EUROGRAPHICS*, volume 22, pages 401–410.

[Rademacher 1999] Rademacher, P. (1999). *GLUI: A GLUT-Based User Interface Library*. http://web.mit.edu/glut/glui-2.1b/glui_manual.pdf. Último acesso em 25/05/2006.

[Sarkar 2000] Sarkar, P. (2000). A brief history of cellular automata. Technical Report 1, Indian Statistical Institute.

[Shreiner 2004] Shreiner, D. (2004). *OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.4*. Addison-Wesley, Boston, MA, USA.

[Stam 1999] Stam, J. (1999). Stable fluids. In *Siggraph 1999*, pages 121–128. Addison Wesley Longman.

[Stewart 2006] Stewart, N. (2006). *GLUI User Interface Library*. Último acesso em 25/05/2006.

[Taylor 1996] Taylor, R. (1996). *The Encyclopedia Of Animation Techniques*. Quarto Inc., London, England.

[Veach and Guibas 1997] Veach, E. and Guibas, L. J. (1997). Metropolis light transport. In *SIGGRAPH '97*, pages 65–76.

[Watt and Watt 1991] Watt, A. and Watt, M. (1991). *Advanced Rendering And Animation Techniques: Theory And Practice.* Addison-Wesley.

[Witting 1999] Witting, P. (1999). Computational fluid dynamics in a traditional animation enviroment. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 129–136, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Wolfram 1994] Wolfram, S. (1994). *Cellular Automata And Complexity : Collected Papers*. AddisonWesley Pub. Co., http://www.stephenwolfram.com/publications/articles-/ca/83-cellular/index.html, 1 edition.

[Xavier 2006] Xavier, A. V. (2006). Animação de fluidos via autômatos celulares e sistemas de partículas. Master's thesis, Laboratório Nacional de Computação Científica.