2
Trabalhos relacionados

Este capítulo introduz alguns trabalhos importantes na área de animação de fluidos, destacando a simulação de escoamentos sobre terrenos e a utilização de sistemas de partículas.

2.1
Métodos para animação de fluidos

Os métodos para animação computacional de fluidos encontrados na literatura adotam técnicas de discretização baseadas em formulações Eulerianas, bem como em formulações Lagrangeanas. Nas formulações Eulerianas o domínio do problema é discretizado por um conjunto de pontos fixos no espaço. As propriedades do fluido são avaliadas a cada instante de tempo através do fluxo que passa por esses pontos. Por sua vez, nas formulações Lagrangeanas, o fluido é discretizado por um conjunto de partículas que acompanham o fluxo. A seguir, alguns dos trabalhos de maior relevância em animação de fluidos.

Foster e Metaxas (1997) apresentaram um modelo Euleriano para animação de gases turbulentos e aquecidos. Eles utilizaram um conjunto simplificado de equações para a modelagem do gás, que exigiu menos esforço computacional para simulação do que a modelagem tradicional. Esta modelagem, mostrou-se adequada para a representação dos efeitos desejados. A discretização das equações foi feita via Diferenças Finitas. As fronteiras e os elementos que compõem a cena foram modelados via voxels. A visualização foi feita via *Volume Rendering* e partículas de prova no interior do fluido. Este método apresenta a desvantagem de ser numericamente instável para grandes passos de tempo. Sendo assim, mesmo com a modelagem simplificada do gás, este método não é capaz de gerar animações em tempo real. Stam (1999) apresentou uma estratégia para solucionar o problema de instabilidade numérica apresentado no trabalho de Foster e Metaxas (1997). Foram utilizados métodos Lagrangeanos e esquemas implícitos para solução numérica das equações de Navier-Stokes, resultando em simulações mais estáveis mesmo com passos de tempo grandes, o que viabilizou a produção de animações em tempo real.

Witting (1999) apresentou um modelo para simulação de fluidos bem mais

completo. Ao contrário de Foster e Metaxas (1997) e Stam (1999), Witting mantém a modelagem do gás como um fluido compressível. Este método foi utilizado para gerar efeitos visuais no filme " O Príncipe do Egito".

Premoze e Ashikhmin (2000) apresentaram um método para geração de superfícies de água sob diferentes condições de vento e iluminação. Tal método é adequado à visualização tanto de águas profundas quanto de águas rasas, uma vez que é capaz de reproduzir a opacidade e coloração específicas em cada caso, pois modela o transporte de luz na água.

Em (Carlson et al., 2004) é apresentado um método para animação da interação entre sólidos e fluidos de superfície livre. Os objetos sólidos são modelados como se fossem feitos de fluido. A rigidez de um objeto é reproduzida restringindo-se o campo de velocidade da região do fluido que representa determinado objeto. Objetos de diferentes densidades podem ser modelados, tais como chumbo ou madeira. A influência entre os elementos do sistema se dá tanto no sentido fluido-sólido quanto no sentido sólido-fluido.

Kass e Miller (1990) propuseram um método para animação de fluidos no qual a superfície do fluido é representada por um campo de altura. A evolução do fluido é controlada por uma versão simplificada das equações de águas rasas. Tal método é adequado à simulação de grandes superfícies de água, como oceanos e escoamentos sobre terrenos.

Dentre os métodos Lagrangeanos baseados em partículas, destacam-se o método SPH (*Smoothed Particle Hydrodynamics*) e o MPS (*Moving Particle Semi-Implicit*).

O método SPH foi desenvolvido por Gingold e Monaghan (1977) e por Lucy (1977) para simulação de problemas astrofísicos. Em (Desbrun; Gascuel, 1996), o SPH foi empregado pela primeira vez em computação gráfica, tendo sido utilizado para animação de objetos deformáveis.

Em (Muller et al., 2003) o método SPH foi utilizado na simulação de fluidos para aplicações interativas. A visualização do fluido foi feita por dois métodos distintos para geração de superfícies: O *Surface Splatting* (Zwicker et al., 2001) e o *Marching Cubes* (Lorensen; Cline, 1987). O primeiro apresentou melhor performance. No entanto, este último apresentou resultados mais convincentes. Em (Muller et al.,), foi apresentada uma extensão do modelo proposto em (Muller et al., 2003), para animação da interação entre fluidos de diferentes tipos e fases, utilizando SPH.

(Nakamura, 2007) apresentou um estudo, sobre animação de fluidos via SPH. Com base nas técnicas descritas por (Muller et al., 2003) e (Muller et al.,), foram gerados diferentes efeitos com fluidos, tais como poças d'água, fluidos sob

confinamento e interação entre fluidos diferentes. Além disso, foram identificados os gargalos do método e os recursos computacionais necessários para produção de animações interativas utilizando tal abordagem.

O MPS foi apresentado inicialmente por (Koshizuka; Oka, 1996) e é uma variante do SPH. Dentre os trabalhos de animação de fluidos para computação gráfica utilizando este método, destaca-se o de (Premoze et al., 2003). Neste trabalho, não foram produzidas animações interativas. O cálculo da pressão via gradientes conjugados e a geração da superfície do fluido via Level Set para renderização foram identificados como gargalos da aplicação. Por outro lado, as simulações produzidas possuem alta resolução (80×10^3 - 150×10^3 partículas) e o fluido apresentou um comportamento bastante convincente.

Sistemas de partículas foram introduzidos na computação gráfica por (Reeves, 1983) como uma técnica para representação de objetos *fuzzy* tais como fogo, fumaça ou nuvens. Desde então, sistemas de partículas vêm sendo amplamente utilizados em simulação de fluidos. Para representar o comportamento do fluido de maneira realista, além de modelos matemáticos adequados, é necessária uma grande quantidade de partículas. Para isso, são necessários sistemas de partículas sofisticados, com algoritmos de detecção de interferência e colisão eficientes. A seguir, são apresentados alguns trabalhos que propõem implementações eficientes de sistemas de partículas.

Em (Celes; Calomeni, 2003) são apresentados alguns métodos para simulação e visulização de sistemas de partículas. Para aumento de desempenho, é proposta uma estratégia para distribuição eficiente do processamento entre CPU e GPU. No entanto, com o recente avanço do hardware gráfico, algumas implementações de sistemas de partículas foram concebidas explorando totalmente as GPU's. Devido ao alto grau de paralelismo nas GPU's, observa-se um aumento de performance considerável com relação às implementações tradicionais com simulação em CPU. Além disso, tal estratégia elimina a transferência de dados das partículas, em tempo de execução, entre CPU (para simulação) e GPU (para renderização).

Latta (2004) apresentou uma implementação totalmente em GPU de um sistema de partículas capaz de simular um milhão de partículas em tempo real. Kolb et al. (2004) apresentaram uma extensão deste trabalho, incluindo detecção de colisão de partículas com objetos de formas arbitrárias e um algoritmo de ordenação das partículas em GPU para renderização correta com *alpha blending* (Shreiner, 2004). Para detecção de colisão, foram utilizados mapas de profundidade, armazenados em texturas de 8 bits, contendo a forma externa do objeto e seus vetores normais. Para ordenar as partículas, foi utilizada uma textura adicional armazenando a distância das partículas para o observador, e o algoritmo de ordenação " odd-even merge sort"

proposto em (Batcher, 1968) é utilizado para ordenar as partículas em função dessa distância.

Kipfer et al. (2004) apresentaram outra implementação inteiramente em GPU, para animação e renderização, em tempo real, de grandes conjuntos de partículas. Kipfer utilizou, nessa implementação, *Memory Objects* OpenGL como *render targets* e para armazenar dados de geometria na placa gráfica. Assim como em (Kolb et al., 2004), Kipfer et al. também inclui uma ordenação das partículas em GPU para renderização correta com *alpha blending*, e adicionalmente trata intercolisões (colisões entre as partículas do sistema), o que também requer uma ordenação. A ordenação, em ambos os casos, foi realizada por uma variação do algoritmo "*Bitonic sort*", proposto em (Purcell et al., 2003), alterando-se apenas a chave de ordenação em cada caso. No caso de detecção de intercolisões, a chave de ordenação utilizada é o índice da célula associado a cada partícula, já para o caso da renderização com transparência através de *alpha blending*, a chave de ordenação é a distância da partícula para o observador.

Mais recentemente, Venetillo e Celes (2007) apresentaram um sistema de partículas baseado em GPU que lida com intercolisões e é capaz de simular um milhão de partículas a taxas interativas. Esse desempenho é obtido nas placas gráficas mais recentes (*GeForce 8 Series* em diante) e além de intercolisões o sistema trata a colisão contra objetos em cena e simulação de materiais granulares.

Vale destacar que, recentemente, diversos trabalhos vêm explorando recursos de GPUs para aceleração da simulação numérica na animação de fluidos. O resultado da exploração de GPUs para este tipo de processamento é, geralmente, um aumento considerável de performance na simulação frente às implementações em CPU. Sendo assim, apesar de não empregarmos este recurso, citamos a seguir alguns trabalhos nesta linha pelos benefícios que podem ser obtidos.

Em (Harris, 2004), é detalhada uma versão para GPU do método baseado em grid " Stable Fluids" proposto em (Stam, 1999).

Amada et al. (2004) propuseram uma implementação do método SPH em GPU. No entanto, a detecção da vizinhança de cada partícula, necessária ao cálculo das grandezas em questão, é executada em CPU. Um mapa com as informações de vizinhança é montado e transferido para GPU em uma textura a cada iteração. A montagem e a transferência do mapa de vizinhança para GPU foram identificadas como o gargalo da aplicação, mas ainda assim foi obtido um ganho de desempenho com relação à implementação em CPU apresentada para fins de comparação.

Kolb e Cuntz (2005) e, mais recentemente, Harada et al. (2007) propuseram implementações do SPH completamente em GPU, eliminando assim, a transferência de dados entre CPU e GPU.

Karsten e Trier (2004) apresentaram um versão em GPU do método proposto em (Kass; Miller, 1990).

Kipfer e Westermann (2006) utilizaram SPH em uma simulação, baseada em GPU, de rios e escoamentos em terrenos para aplicações interativas em computação gráfica. Para determinar as partículas vizinhas e detectar colisão, foi proposta uma estrutura baseada em três listas encadeadas. Além disso, foi apresentado um método para extração e visualização da superfície livre do fluido, que utiliza um campo de altura sobre as partículas. Tal método mostrou-se mais eficiente que os métodos *Marching Cubes* e *Surface Splatting* para aplicações que demandam tempo real.

Maes et al. (2006) propuseram uma outra abordagem baseada em GPU para simulação de escoamentos em terrenos para aplicações interativas. A proposta foi uma otimização do método baseado em colunas de água e variação de pressão hidrostática apresentado por Holmberg e Wunsche (2004). Além disso, um sistema de partículas foi utilizado nesse modelo, para adicionar mais detalhes à superfície do fluido.