2 Trabalhos Relacionados

2.1 SPH

Sistemas de partículas tê m se mostrado úteis em aplicações de computação gráfica (William, 1983). Em particular, o uso de partículas para modelar líquidos pode ser visto em (Gavin e Andrew, 1989) e (Demetri *et al*, 1989). O método de simulação por partículas SPH (*Smoothed-particle Hydrodynamics*) foi desenvolvido por Lucy (1977) e Gingold e Monaghan (1977) com o objetivo de modelar e simular problemas astrofísicos, sendo posteriormente estendidos para aplicações em mecânica dos fluidos. Dado que o movimento coletivo das partículas é similar ao movimento de líquidos, este pode ser modelado utilizando equações que governam a hidrodinâmica Newtoniana clássica.

Como a gama de problemas que podem ser resolvidos com SPH é grande, muitos estudos foram focados em extensões e melhoramentos importantes do método original. Os aspectos numéricos têm se tornado cada vez mais robustos, uma vez que algumas técnicas foram modificadas e a correção de alguns métodos foram propostos. O trabalho de Morris (1996) identificou o problema da inconsistência das partículas, podendo causar uma baixa acurácia da solução.

Com o passar dos anos, diferentes modificações no método foram estudadas para restaurar a consistência e aumentar a acurácia do SPH. Monaghan e Gingold (1983) aplicaram o método de SPH para simular o problema do tubo de choque em uma dimensão. Neste trabalho, os autores propuseram uma nova forma de viscosidade artificial, com a finalidade de eliminar os problemas de oscilação excessiva da frente da onda de choque.

No final da década de 80, Monaghan (1988) aplicou o método SPH para simular diversos tipos de problemas de fluidos compressíveis. Neste trabalho surgiu a dificuldade de modelagem de fronteiras sólidas e a necessidade de se utilizar uma velocidade média local para realizar o movimento das partículas.

Em 1994, o escoamento de fluido incompressível foi estudado utilizando o método SPH (Monaghan, 1994). Neste trabalho, as fronteiras sólidas foram

modeladas utilizando partículas que exerciam uma força de repulsão nas partículas de fluido.

A melhoria da acurácia da aproximação da simulação por partículas foi apresentada no trabalho de Liu e Chen (1996), através do método "Reproducing Kernel Particle Method" (RKPM). Este método foi aprimorado por Chen (1999) através de um método corretivo, denominado "Corrective Smoothed Particle Method" (CSPM), que aumentou a confiabilidade dos dados, tanto dentro do problema de domínio quanto nas áreas vizinhas.

Na década de 90 surge o primeiro trabalho aplicado de SPH na computação gráfica com Desbrun e Gascuel (1996) realizando simulações de substâncias altamente deformáveis. Resultados em tempo real foram apresentados por Muller (2003), e incluem o cálculo da tensão superficial do fluido (Figura 2). No ano seguinte, Muller (2004), continuou seu trabalho estudando as forças de repulsão, adesão e atrito entre o fluido e corpos sólidos utilizando a simulação SPH.

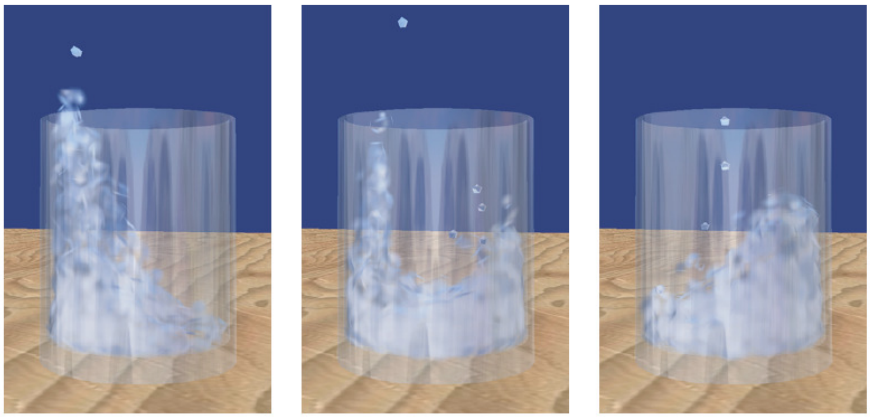

Figura 2 - Simulação de SPH realizada no trabalho de Müller (2003)

Dois anos depois, baseado no trabalho de Morris (2000), Muller *et al*. (2005) apresentaram uma proposta de simulação e interação entre fluidos, estendendo o trabalho anterior (Müller, 2003) com o acréscimo de forças de interface entre fluidos distintos com características diferentes.

Em 2007, para validar o método de SPH para simulação e animação de fluidos, Nakamura (2007) desenvolveu uma biblioteca que realiza simulação de fluidos incompressíveis baseada nos trabalhos de Muller *et al* (2003 e 2005). Essa biblioteca foi utilizada nesta dissertação para gerar uma melhor visualização do sistema através de uma malha de triângulos.

Alves (2008), em sua dissertação, realizou um estudo comparativo de eficiência e acurácia dos métodos MPS e SPH. Foi analisado e comprovado que para simulações de fluidos em tempo real com a finalidade de animação, em que apenas uma aproximação do comportamento real do escoamento é necessária, o método SPH pode ser considerado a melhor alternativa devido ao baixo custo computacional comparado com o método MPS.

2.2 Visualização Volumétrica

Podemos dividir os algoritmos de visualização volumétrica em duas classes: algoritmos de *rendering* direto e os de extração de superfície. Nos algoritmos da primeira classe, a imagem é gerada diretamente a partir dos dados volumétricos, sem passar por etapa intermediária alguma. Nos algoritmos de extração de superfície, as imagens são obtidas através da geração de representações geométricas dos dados, utilizando técnicas de *rendering* de polígonos. Dentre os membros desta segunda classe está o algoritmo de Marching Cubes usado na presente dissertação.

2.2.1 *Rendering* Direto

Na classe de rendering direto, podemos citar o algoritmo de Ray-Casting (Levoy, 1988) O algoritmo percorre todos os pixels da imagem, determinando a cor e a opacidade de cada um. Um raio é disparado de cada pixel através do volume de dados. As cores e as opacidades encontradas ao longo do raio são acumuladas até determinarmos a opacidade e a cor final daquele pixel (Figura 3). Este algoritmo possui um alto custo computacional, porém pode ser paralelizado sem muitas dificuldades devido à independência entre os raios lançados.

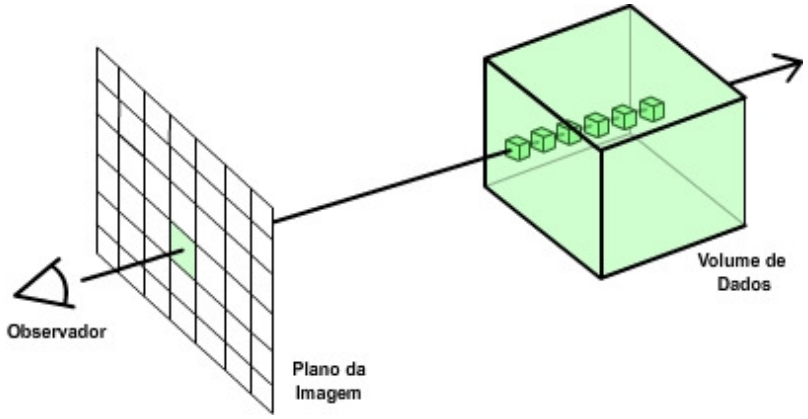

Figura 3 - Esquema do algoritmo de Ray-Casting. Os cubos menores representam as frações das cores e opacidades acumuladas do volume de dados

Posteriormente, Westover (1990) propôs uma técnica, chamada *Splatting*, que, numa primeira etapa, determina em qual ordem o volume de dados deve ser percorrido pelo algoritmo, ou seja, qual voxel está mais próximo do plano de visualização. Numa segunda etapa, a projeção dos voxels no plano de visualização é calculada de acordo com as suas distâncias deste plano. Para determinar a extensão da contribuição é utilizado um filtro de arredondamento. Quanto maiores o tamanho do volume e o tamanho da imagem a ser gerada, maior é a projeção do filtro no plano de visualização. As contribuições dos voxels são calculadas utilizando tabelas de mapeamento (*look-up tables*).

No ano seguinte, Laur e Hanrahan (1991) propuseram uma otimização do algoritmo de Westover, denominado *Hierarquical Splatting*, que utiliza um caminhamento hierárquico no volume de dados e um refinamento progressivo da imagem.

Similarmente ao algoritmo anterior, Upson e Keeler (1988) desenvolveram um trabalho chamado *V-Buffer*. A principal característica deste algoritmo é o fato de ser baseado em células ao invés de *voxels*. Ao percorrer o interior das células, ele interpola os valores dos vértices e projeta cada valor interpolado no plano de visualização.

2.3 Extração de Superfícies

Fazendo parte da classe de extração de superfícies, a conexão de contornos (Keppel, 1975) foi uma das primeiras técnicas de visualização volumétrica. Inicialmente sugerida por Keppel, o algoritmo traça uma "iso-linha" em cada fatia de dados e conecta as fatias adjacentes (Figura 4). Apesar da

velocidade para a geração e exibição da imagem final ser grande e utilizar pouco espaço para armazenar a estrutura, o método gera pedaços falsos da superfície e impossibilita a criação de imagens para volumes compostos de microestruturas.

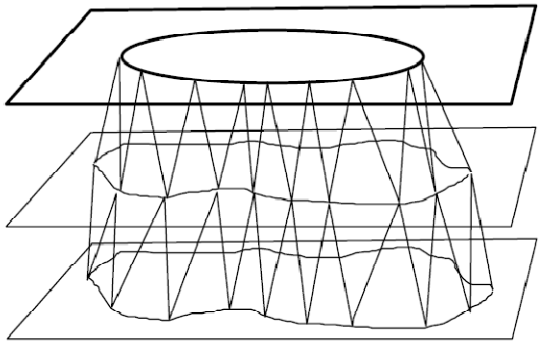

Figura 4 - Representação do algoritmo Conexão de Contonos. Conexão das iso-linhas entre os planos de corte ou "fatias"

Um outro algoritmo, desenvolvido por Herman (1983), denominado de Cubos Opacos (*Cuberille*) é dividido em duas fases. Na primeira fase o usuário determina um valor de limiarização e o algoritmo faz uma busca no volume de dados demarcando os valores que se encontram no nível deste limiar. Seis polígonos, um para cada face da célula, são gerados para estas células. Numa segunda fase, a descrição geométrica deste polígonos é transferida para um visualizador de superfícies que gerará a imagem final.

Apesar do algoritmo de Herman ser rápido e sua implementação não ser complexa, possui a deficiência em visualizar detalhes da superfície. Após a geração da superfície, esta apresenta uma aparência de pequenos blocos (Figura 5). Gordon e Reynolds (1985) chegaram a resultados mais realísticos desta técnica levando em consideração também o gradiente da superfície, calculado a partir dos cubos vizinhos de cada célula. Vale ressaltar que a primeira etapa do algoritmo pode ser paralelizada, uma vez que os polígonos de cada célula podem ser gerados independentemente.

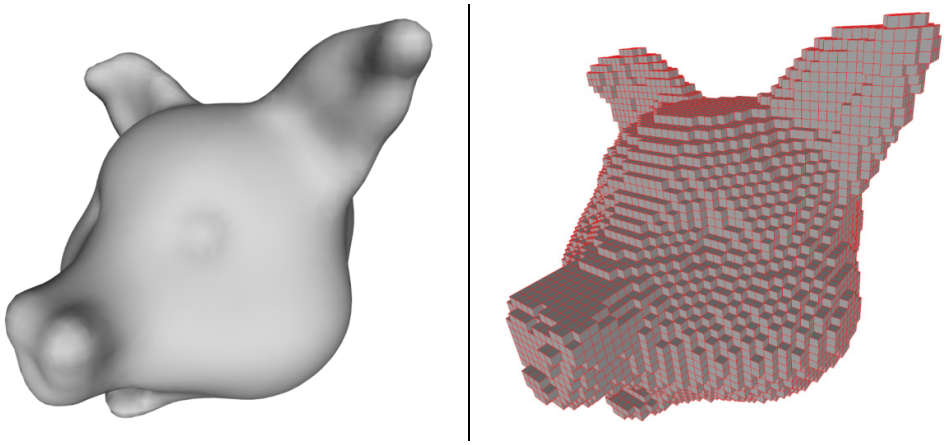

Figura 5 - Esquerda: Imagem de uma iso-superfície ideal. Direita: Imagem gerada pelo algoritmo de "Cuberille"

Um método de extração de superfícies que se tornou muito conhecido no mundo científico relacionado à visualização é o Marching Cubes. Na próxima seção serão descritos alguns trabalhos relacionados com este algoritmo.

2.4 Marching Cubes

O algoritmo de *Marching Cubes* foi desenvolvido independentemente por Wyvill e Mcpheeters (1986) e por Lorensen e Cline (1987). Wyvill utilizou este algoritmo para extrair iso-superfícies de um campo analítico definido por um conjunto de pontos. Já Lorensen aplicou a idéia à criação de modelos de superfícies de densidade constante, a partir de dados médicos 3D resultantes de exames de Tomografia Computadorizada (CT), Ressonância Magnética (MRI), etc.

Posteriormente, Dürst (1988) entre outros, descobriu que a aproximação original da superfície poderia produzir buracos na malha de triângulos da iso-superfície. Montani *et al.* (1994) propuseram uma ligeira alteração no esquema de geração da *lookup table* para previnir tais buracos separando os vértices positivos em faces ambíguas.

Natarajan (1994) e Chernyaev (1995) examinaram a interpolação trilinear no interior de uma célula e determinaram uma topologia da iso-superfície trilinear por seção. Contudo, o método de Natarajan não leva em consideração os pontos no interior de cada célula resultando em triangulações inválidas. Já o estudo de

Chernyaev apresenta o uso de pontos adicionais no interior de cada célula, contudo apresenta algumas triangulações inválidas.

Em seguida, Lopes (1999), em sua dissertação, discutiu métodos para aumentar a acurácia da visualização científica. Seu método analisa a interpolação trilinear para determinar a topologia da superfície e utiliza os pontos adicionais no interior e nas faces das células para melhorar a geração da iso-superfície. Com os pontos adicionais utilizados são obtidas malhas triangulares válidas.

Diferentemente dos estudos anteriores, que se baseavam na interpolação trilinear por seção da superfície, Gelder e Wilherlms (1994) realizaram um estudo considerando amostras de valores de uma função quadrática para a correção da topologia. Já Stander e Hart (1997) consideraram o uso de funções implícitas. Suas aproximações detectam todos os pontos críticos dessas funções e constroem a poligonização baseada na topologia da função implícita utilizada.

Lorensen e Cline descobriram posteriormente, que o tamanho de alguns triângulos gerados era menor que o tamanho de um pixel. Um novo algoritmo, denominado *Dividing Cube* (Cline et al, 1988), foi então desenvolvido para tirar vantagem desta observação. O algoritmo efetua a projeção das células na tela verificando somente aquelas em que a projeção é maior que um pixel. Se isso ocorrer, a célula é dividida em sub-células, cada uma das quais determinando um ponto. Se não, a célula toda é visualizada também como um ponto (Paiva *et al*, 1999).

As principais dificuldades do algoritmo são a geração de iso-superfícies com descontinuidades. Isso se deve ao fato dessa superfície ser indeterminada em certos cubos, levando em consideração apenas os valores dos vértices. Logo sua topologia é ambígua. Tal ambiguidade pode ocorrer tanto na face quanto dentro do cubo.

As ambiguidades faciais foram resolvidas por Nielson e Hamann (1991) supondo que o campo escalar fosse trilinear em cada cubo. Adicionalmente foi utilizada uma tabela de topologia (*look-up table*) diferenciada (Montani *et al*, 1994) . Da mesma maneira foi possível resolver as ambiguidades internas como foi feito em (Chernyaev, 1995).

Em 2003, Lewiner *et al* (2003) desenvolveram uma implementação mais robusta e eficiente do algoritmo apresentado por Chernyaev (1995). Este novo algoritmo garante um resultado topologicamente correto, completando, assim, o artigo anterior (Figura 6).

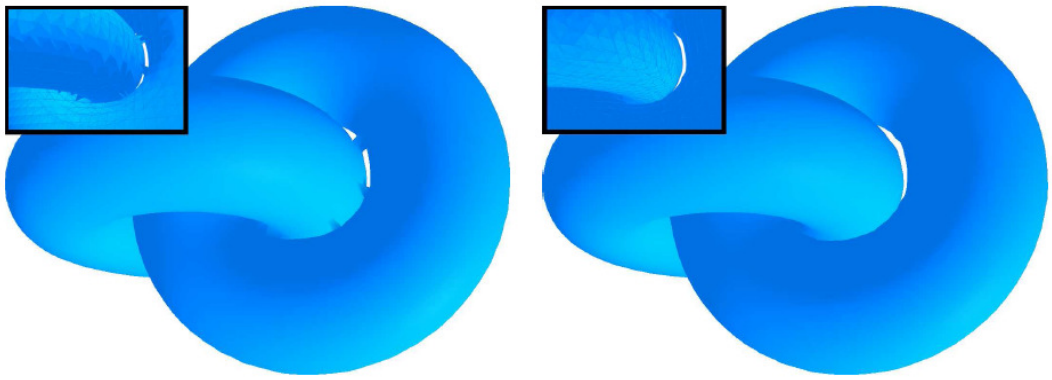

Figura 6 – Torus interligados gerados pelo algoritmo de Chernyaev (à esquerda) e pelo de Lewiner (à direita) – Imagem retirada de Lewiner et al (2003)