Rodrigo de Souza Lima Espinha

Visualização Volumétrica Interativa de Malhas Não-Estruturadas Utilizando Placas Gráficas Programáveis

DISSERTAÇÃO DE MESTRADO

DEPARTAMENTO DE INFORMÁTICA

Programa de Pós-Graduação em Informática

Rio de Janeiro, março de 2005

Pontifícia Universidade Católica
do Rio de Janeiro

Rodrigo de Souza Lima Espinha

Visualização Volumétrica Interativa de Malhas Não-Estruturadas Utilizando Placas Gráficas Programáveis

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio.

Orientador: Prof. Waldemar Celes Filho

Rio de Janeiro, março de 2005

Pontifícia Universidade Católica
do Rio de Janeiro

Rodrigo de Souza Lima Espinha

Visualização Volumétrica Interativa de Malhas Não-Estruturadas Utilizando Placas Gráficas Programáveis

Dissertação apresentada como requisito parcial para obtenção do título de Mestre pelo Programa de Pós-Graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Waldemar Celes Filho
Orientador
Departamento de Informática - PUC-Rio

Prof. Marcelo Gattass
Departamento de Informática - PUC-Rio

Prof. Paulo Cezar Pinto Carvalho
Instituto Nacional de Matemática Pura e Aplicada (IMPA)

Prof. Luiz Fernando Martha
Departamento de Engenharia Civil - PUC-Rio

Prof. José Eugênio Leal
Coordenador Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 18 de março de 2005

Rodrigo de Souza Lima Espinha

Graduou-se em Engenharia de Computação pela Pontifícia Universidade Católica do Rio de Janeiro, onde continuou seus estudos no programa de Mestrado em Informática. Durante a permanência nesta instituição, atuou em projetos voltados para a indústria do petróleo, no laboratório Tecgraf.

Ficha Catalográfica

Espinha, Rodrigo de Souza Lima

Visualização volumétrica interativa de malhas não-estruturadas utilizando placas gráficas programáveisl / Rodrigo de Souza Lima Espinha; orientador: Waldemar Celes Filho. – Rio de Janeiro: PUC, Departamento de Informática, 2005.

86 f. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática.

Inclui referências bibliográficas.

1. Informática – Teses. 2. Visualização volumétrica. 3. Programação em placas gráficas. 4. Malhas não-estruturadas. 5. Visualização interativa. 6. Função de transferência. I. Celes Filho, Waldemar. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Aos meus pais, meu irmão e minha namorada.

Agradecimentos

A Deus, pois sem Ele nada teria sido possível.

À minha família e minha namorada, Herisangela, pelo incentivo em todos os momentos.

Ao meu orientador, professor Waldemar Celes, pelo apoio, motivação e o conhecimento que me foi transmitido ao longo do curso.

À CAPES, Tecgraf e PUC-Rio, pelo financiamento e oportunidade de trabalhar e estudar em um lugar onde posso aprender todos os dias.

Aos meus colegas e amigos, especialmente Frederico Abraham (Fred), Antonio Calomeni, Luiz Gustavo, Márcio Pereira, Rodrigo Hermann, Michel e Manuel (peruanos), que também percorreram esse mesmo caminho ao longo dos últimos dois anos.

A Antonio Miranda e Ivan Menezes, do Tecgraf, pela paciência e pelas malhas de elementos finitos utilizadas neste trabalho.

A todos do Tecgraf e da PUC-Rio.

Resumo

Espinha, Rodrigo de Souza Lima. **Visualização Volumétrica Interativa de Malhas Não-Estruturadas Utilizando Placas Gráficas Programáveis.** Rio de Janeiro, 2005. 86p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A visualização volumétrica é uma importante técnica para a exploração de dados tridimensionais complexos, como, por exemplo, o resultado de análises numéricas usando o método dos elementos finitos. A aplicação eficiente dessa técnica a malhas não-estruturadas tem sido uma importante área de pesquisa nos últimos anos. Há dois métodos básicos para a visualização dos dados volumétricos: extração de superfícies e renderização direta de volumes. Na primeira, iso-superfícies de um campo escalar são extraídas explicitamente. Na segunda, que é a utilizada neste trabalho, dados escalares são classificados a partir de uma função de transferência, que mapeia valores do campo escalar em cor e opacidade, para serem visualizados. Com a evolução das placas gráficas (GPU) dos computadores pessoais, foram desenvolvidas novas técnicas para visualização volumétrica interativa de malhas não-estruturadas. Os novos algoritmos tiram proveito da aceleração e da possibilidade de programação dessas placas, cujo poder de processamento cresce a um ritmo superior ao dos processadores convencionais (CPU). Este trabalho avalia e compara dois algoritmos para visualização volumétrica de malhas não-estruturadas, baseados em GPU: projeção de células independente do observador e traçado de raios. Adicionalmente, são propostas duas adaptações dos algoritmos estudados. Para o algoritmo de projeção de células, propõe-se uma estruturação dos dados na GPU para eliminar o alto custo de transferência de dados para a placa gráfica. Para o algoritmo de traçado de raios, propõe-se fazer a integração da função de transferência na GPU, melhorando a qualidade da imagem final obtida e permitindo a alteração da função de transferência de maneira interativa.

Palavras-chave

visualização volumétrica, programação em placas gráficas, malhas não-estruturadas, visualização interativa, função de transferência

Abstract

Espinha, Rodrigo de Souza Lima;. **Interactive Volume Visualization of Unstructured Meshes Using Programmable Graphics Cards.** Rio de Janeiro, 2005. 86p. MSc. Dissertation - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Volume visualization is an important technique for the exploration of three-dimensional complex data sets, such as the results of numerical analysis using the finite elements method. The efficient application of this technique to unstructured meshes has been an important area of research in the past few years. There are two basic methods to visualize volumetric data: surface extraction and direct volume rendering. In the first, the iso-surfaces of the scalar field are explicitly extracted. In the second, which is the one used in this work, scalar data are classified by a transfer function, which maps the scalar values to color and opacity, to be visualized. With the evolution of personal computer graphics cards (GPU), new techniques for volume visualization have been developed. The new algorithms take advantage of modern programmable graphics cards, whose processing power increases at a faster rate than the one observed in conventional processors (CPU). This work evaluates and compares two GPU-based algorithms for volume visualization of unstructured meshes: view-independent cell projection (VICP) and ray-tracing. In addition, two adaptations of the studied algorithms are proposed. For the cell projection algorithm, we propose a GPU data structure in order to eliminate the high costs of the CPU to GPU data transfer. For the ray-tracing algorithm, we propose to integrate the transfer function in the GPU, which increases the quality of the generated image and allows to interactively change the transfer function.

Keywords

volume visualization, GPU programming, unstructured meshes, interactive visualization, transfer function.

Sumário

1 Introdução 14
1.1. Motivação 18
1.2. Objetivo 19

2 Classificação dos dados volumétricos 21
2.1. Função de transferência 22
2.2. Modelos ópticos 23
2.3. Aplicação da função de transferência 28
2.3.1. Pré-integração 29
2.3.2. Integração de segmentos lineares 32
2.3.3. Iso-superfícies na placa gráfica 35

3 Algoritmos para visualização volumétrica baseados na GPU 38
3.1. Projeção de tetraedros 39
3.1.1. Ordenação de células 45
3.2. Traçado de raios na placa gráfica 48
3.2.1. Estruturas de dados 53

4 Modificações realizadas 59
4.1. Estruturas de dados na placa gráfica 59
4.1.1. Projeção de células (VICP) 59
4.1.2. Traçado de Raios 62
4.2. Integração de segmentos lineares 63

5 Comparação e Resultados 67
5.1. Desempenho 71
5.2. Memória 75
5.3. Qualidade de imagem 76
5.4. Modificação interativa da função de transferência 79

6 Conclusão 81
6.1. Trabalhos futuros 82

7 Referências bibliográficas 83

Lista de figuras

Figura 1 – Dados volumétricos dispostos como uma grade tridimensional. 15

Figura 2 – Exemplos de malhas tridimensionais projetadas. As malhas são apresentadas hierarquicamente, da mais restrita à mais geral: (a) grade cartesiana ou uniforme, (b) grade regular, (c) grade retilínea, (d) malha estruturada, (e) malha não-estruturada de tetraedros. 15

Figura 3 – Visualização volumétrica dos resultados da análise de elementos finitos para uma malha de tetraedros, usando duas técnicas: (a) renderização direta de volumes, (b) extração de iso-superfícies. 16

Figura 4 – Etapas da renderização direta de volumes. 17

Figura 5 – Exemplos dos principais elementos finitos: (a) tetraedro linear (4 nós), (b) hexaedro linear (8 nós), (c) tetraedro quadrático (10 nós). 18

Figura 6 – (a) Valor de um campo escalar ao longo de um raio com parâmetro *t*. (b) Função de transferência definida para valores escalares. Os escalares são mapeados para valores de cor *(r,g,b)* e opacidade α. 22

Figura 7 – Volume diferencial composto por partículas que absorvem a intensidade de um raio *I(s)*, ao longo da espessura *ds*. 24

Figura 8 – Aproximação de uma função por segmentos constantes (a) e segmentos lineares (b). 28

Figura 9 – Composição das contribuições de três tetraedros ao longo de um raio em direção ao observador. 28

Figura 10 – Um raio que parte do observador e atravessa um tetraedro. Os escalares de entrada e saída, s_f e s_b, são interpolados a partir dos escalares associados aos vértices do tetraedro, enquanto que *l* é a distância entre a posição de entrada e saída do raio. 29

Figura 11 – Textura 3D contendo valores de cor associada e opacidade (C', α), resultantes da pré-integração da função de transferência, em função dos escalares de entrada e saída e a distância do raio (s_f, s_b, l). 30

Figura 12 – (a) Função de transferência definida por segmentos lineares. (b) "Fatia" de um tetraedro limitada pelos dois pontos de controle de um

segmento linear. 32
Figura 13 – Projeção de um tetraedro cortado por duas iso-superfícies definidas por pontos de controle da função de transferência. Os exemplos *a*, *b*, e *c* representam os casos fundamentais para um raio que atravessa o tetraedro. 33
Figura 14 – Textura 2D contendo os valores de cor e opacidade da primeira iso-superfície atravessada por um raio, para cada par de coordenadas (s_f, s_b). 36
Figura 15 – As áreas relativas às iso-superfícies representadas na textura 2D são aumentadas para corrigir falhas visuais. 37
Figura 16 – Classificação de tetraedros projetados de acordo com o número de triângulos necessários para a renderização. 40
Figura 17 – Raio partindo do observador que atravessa um tetraedro linear. 41
Figura 18 – Projeção lateral de um tetraedro que é atravessado por um raio que parte do observador. 42
Figura 19 – Raio que parte do observador e atravessa um tetraedro. Cada vértice do tetraedro possui um vetor (t_0, t_1, t_2, t_3) das respectivas distâncias às faces f_0, f_1, f_2 e f_3, na direção do raio. 44
Figura 20 – Exemplo de um ciclo de visibilidade. 46
Figura 21 – Ilustração do algoritmo MPVO. (a) As setas representam a classificação das faces internas como “entrando”, “saindo” ou “nenhum”, em relação ao observador. (b) Grafo orientado criado a partir da classificação das faces. 47
Figura 22 – Extensão das relações de adjacência do MPVO, para malhas não-convexas, representada pelas setas vermelhas. 48
Figura 23 – Propagação de raios que partem do observador e atravessam a malha de tetraedros. A cada passo de um raio, é atravessado um tetraedro. 49
Figura 24 – Interseção de um raio com as faces de um tetraedro. 49
Figura 25 – Os raios de cores diferentes representam as camadas necessárias para a visualização de uma malha não-convexa. 52
Figura 26 – (a) Vetor unidimensional indexado pelo índice *i*. (b) Textura 2D equivalente ao vetor. O índice *i* é decomposto em duas coordenadas de textura, *u* e *v*. As dimensões da textura são *w* e *h*. 54
Figura 27 – (a) Faixa com 3 tetraedros. (b) Representação da faixa, onde v_k é o índice global de um vértice e a_0-a_3 são as adjacências de um tetraedro da

faixa, armazenadas na posição do primeiro vértice do tetraedro. (c) Representação compactada da faixa. 56

Figura 28 – Textura 2D utilizada para a determinação das iso-superfícies definidas pelos pontos de controle da função de transferência. Em cada posição são armazenados *(s_{iso}; prox_s_{iso})*, relativos à primeira interseção de um raio entre s_f e s_b, e o valor da próxima iso-superfície. Se s_{iso} for igual a -1, então nenhuma iso-superfície é atravessada. 64

Figura 29 – (a) Imagem da grade de 16x16x16. (b) Visualização volumétrica da grade de 16x16x16. 69

Figura 30 – (a) Imagem da malha *Barra Fixa*. (b) Visualização volumétrica da malhas *Barra Fixa*. 69

Figura 31 – (a) Imagem da malha *Roda*. (b) Visualização volumétrica da malha *Roda*. 70

Figura 32 – (a) Imagem da malha *Bluntfin*. (b) Visualização volumétrica da malha *Bluntfin*. 70

Figura 33 – (a) Imagem da malha *Oxygen*. (b) Visualização volumétrica da malha *Oxygen*. 71

Figura 34 – Diferença entre a qualidade da imagem gerada pelo VICP (a) e Traçado de Raios (Pré-integração) (b), para grade 32x32x32, com a mesma função de transferência, armazenada em uma textura 3D de 128x128x128. 77

Figura 35 – Diferença entre a qualidade da imagem do Traçado de Raios (Pré-integração) e Traçado de Raios (Integração na GPU): (a) função de transferência utilizada; (b) Traçado de Raios (Pré-integração), com a função de transferência pré-integrada e armazenada em uma textura 3D de 128x128x128; (c) Traçado de Raios (Integração na GPU). 78

Figura 36 – *Aliasing* ocorrido em alguns casos do algoritmo de Traçado de Raios, para a malha Bluntfin, quando a precisão de 16 *bits* é utilizada para o parâmetro da equação do raio. 79

Lista de tabelas

Tabela 1 – Dados armazenados na textura 2D utilizada para a passagem de parâmetros para um passo da propagação de um raio. 55

Tabela 2 – Estrutura de dados utilizada para o Traçado de Raios na placa gráfica, originalmente usada por Weiler et al. (2003a). 55

Tabela 3 – Texturas 2D utilizadas para o armazenamento de faixas de tetraedros. 57

Tabela 4 – Texturas 2D utilizadas para o armazenamento de uma malha de tetraedros na placa gráfica, para projeção de células. 61

Tabela 5 – Estrutura de dados utilizada para a implementação do algoritmo de Traçado de Raios. 62

Tabela 6 – Características das malhas utilizadas para os testes de desempenho. 68

Tabela 7 – Comparação dos resultados, em número de quadros por segundo (qd/s) e tetraedros por segundo (tet/s), dos algoritmos VICP (CPU) e VICP (GPU). 72

Tabela 8 – Tempos, em segundos, necessários à ordenação de visibilidade das células de diversas malhas. 73

Tabela 9 – Comparação dos resultados, em número de quadros por segundo (qd/s) e tetraedros por segundo (tet/s), do VICP (GPU) e o Traçado de Raios (Pré-integração), para diversas malhas. 74

Tabela 10 – Comparação dos resultados, em número de quadros por segundo (qd/s) e tetraedros por segundo (tet/s), para o Traçado de Raios (Integração na GPU), com 10 e 255 segmentos, para diversas malhas. 75