



Henry Giovanni Gallegos Velgara

**Visualização de variedades implícitas de
dimensão 3 no \mathbb{R}^4**

Dissertação de Mestrado

Dissertação apresentada ao Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC–Rio como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Prof. Hélio Côrtes Vieira Lopes

Rio de Janeiro
Agosto de 2014



Henry Giovanni Gallegos Velgara

**Visualização de variedades implícitas de
dimensão 3 no \mathbb{R}^4**

Dissertação apresentada ao Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio como requisito parcial para obtenção do grau de Mestre em Informática. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Hélio Côrtes Vieira Lopes

Orientador

Departamento de Informática — PUC-Rio

Prof. Luiz Henrique de Figueiredo

IMPA

Prof. Sinésio Pesco

Departamento de Matemática — PUC-Rio

Prof. José Eugênio Leal

Coordenador Setorial do Centro Técnico Científico — PUC-Rio

Rio de Janeiro, 22 de Agosto de 2014

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador.

Henry Giovanni Gallegos Velgara

Graduou-se em Ingeniería de Sistemas na Universidad Nacional de San Agustín (UNSA) de Arequipa, Perú. Trabalhou na Cátedra CONCYTEC da UNSA como pesquisador e desenvolvedor em projetos de Gestão Inteligente de Recursos Hídricos. Trabalhou como pesquisador no Tecgraf da PUC-Rio em projetos de Engenharia Naval.

Ficha Catalográfica

Gallegos Velgara, Henry Giovanni

Visualização de variedades implícitas de dimensão 3 no \mathbb{R}^4
/ Henry Giovanni Gallegos Velgara; orientador: Hélio Côrtes
Vieira Lopes. — 2014.

55 f. : il. (color); 30 cm

1. Dissertação (mestrado) - Pontifícia Universidade
Católica do Rio de Janeiro, Departamento de Informática,
2014.

Inclui bibliografia.

1. Informática – Teses. 2. Visualização em R4. 3.
Visualização de variedades implícitas. 4. Ray Casting. 5.
Aritmética Intervalar. 6. 16-Tree. 7. GPU – GLSL. I. Lopes,
Hélio Côrtes Vieira. II. Pontifícia Universidade Católica do Rio
de Janeiro. Departamento de Informática. III. Título.

A Deus, a minha mãe, ao meu pai, aos meus irmãos, e a minha noiva Jakyta
este trabalho vai para vocês.

Agradecimentos

Agradeço a Deus pela oportunidade e por ficar a meu lado sempre.

A meu orientador pela paciência, apoio e incentivo para a realização deste trabalho, e a todos os professores com os quais aprendi ao longo destes dois anos.

À CAPES e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

A minha mãe Juana Velgara, ao meu pai Raúl Gallegos, aos meus irmãos Erika, Sofia e Raúl, pela compreensão e porque me apoiaram mesmo com minha ausência.

A minha quase esposa Jakelinne, pelas palavras de incentivo nos momentos mais difíceis, pela grande compreensão e amor.

Aos meus amigos pela torcida e incentivo.

Resumo

Gallegos Velgara, Henry Giovanni; Lopes, Hélio Côrtes Vieira. **Visualização de variedades implícitas de dimensão 3 no \mathbb{R}^4** . Rio de Janeiro, 2014. 55p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O principal objetivo deste trabalho é apresentar um novo método para visualização de variedades implícitas de dimensão 3 mergulhadas no \mathbb{R}^4 . Esse método consiste primeiramente de um pré-processamento em CPU utilizando uma árvore 16-Tree e a Aritmética Intervalar para encontrar as regiões do domínio onde a variedade se encontra. Esses dados são posteriormente processados em GPU para efetuar a visualização, e para isso foi utilizada uma generalização da técnica *Ray Casting*.

Palavras-chave

Visualização em \mathbb{R}^4 ; Visualização de variedades implícitas; Ray Casting; Aritmética Intervalar; 16-Tree; GPU – GLSL.

Abstract

Gallegos Velgara, Henry Giovanni; Lopes, Hélio Côrtes Vieira (Advisor). **Visualization of 3–dimension implicit manifolds in R^4** . Rio de Janeiro, 2014. 55p. MSc. Dissertation — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The main objective of this work is to present a new method for the visualization of implicit 3-manifolds in \mathbb{R}^4 . This method consists primarily of a preprocessing in the CPU using a 16-tree and Interval Arithmetic to detect regions of the domain where the variety is present. These data are then processed in the GPU to perform the visualization, and for this a generalization of Ray Casting technique was adopted.

Keywords

R4 visualization; Implicit Manifold visualization; Ray Casting; Interval Arithmetic; 16-Tree; GPU – GLSL.

Sumário

1	Introdução	12
1.1	Objetivos	12
1.2	Trabalhos relacionados	12
1.3	Organização	13
2	Fundamentos teóricos	14
2.1	Variedades implícitas	14
2.2	Aritmética intervalar	15
2.3	Ray casting implícito	16
2.4	Programação em GPU	19
2.5	Modelo de iluminação de Phong	23
3	Método proposto	27
3.1	Introdução	27
3.2	Descrição geral do método	27
3.3	Pré-processamento em CPU	28
3.4	Processamento em GPU	32
4	Resultados	41
4.1	Avaliação	42
4.2	Variando os ângulos	48
4.3	Testes em CPU	48
5	Conclusões e trabalhos futuros	52
6	Referências Bibliográficas	53

Lista de figuras

2.1	O processo do <i>Ray Tracing</i> . Fonte: (Suffern, 2007).	17
2.2	Definição do raio.	17
2.3	Encontrando a raiz com amostragem de pontos com bisseção.	19
2.4	O pipeline gráfico programável. Fonte: (Lighthouse3d).	22
2.5	Um exemplo do estágio de rasterização (esquerda) e interpolação (direita) do pipeline gráfico. Fonte: (Lighthouse3d).	22
2.6	Uma descrição visual dos estágios do pipeline gráfico, desde os vértices até os pixels. Fonte: (Engel et al., 2006).	23
2.7	Geometria do modelo de iluminação de Phong, à esquerda os vetores que intervêm e a direita o cálculo do raio de reflexão da luz (Díaz, 2008).	24
2.8	Exemplo da iluminação com o modelo de Phong. Esquerda, iluminando com o termo ambiente. Centro, adicionando o termo difuso. Direita, imagem final com o termo especular.	25
3.1	Diagrama da descrição geral do método proposto, as caixas azuis são processadas em CPU uma só vez, a caixa verde é processada em CPU em cada frame, e as caixas vermelhas são processadas em GPU em cada frame.	28
3.2	Vetores de projeção e observadores da Matriz A.	31
3.3	Hipercaixa e sua hipersfera envolvente.	32
3.4	Dois triângulos usados para ativar os pixels que serão processados.	34
3.5	Plano perpendicular que passa por cada pixel, formado pelos vetores \mathbf{n}_1 e \mathbf{n}_2 .	35
3.6	Raios aleatórios, e o cálculo da cor final.	36
3.7	Calculando os valores t_{min} e t_{max} .	37
3.8	Método de intervalos composto com o algoritmo de bisseção.	38
3.9	Iluminação da função $F(x, y, z, w) = w - (x^2 + y^2 + z^2 + R^2 - r^2)^2 + 4R^2(x^2 + y^2)$ com $R = 2$ e $r = 1$, nos ângulos: esquerda ($\theta_1 = 0.75, \theta_2 = 0.75, \theta_3 = 0.0, \theta_4 = 0.0, \theta_5 = 0.0, \theta_6 = 0.0$) e direita ($\theta_1 = 0.75, \theta_2 = 0.75, \theta_3 = 0.75, \theta_4 = 0.75, \theta_5 = 0.0, \theta_6 = 0.0$).	38
3.10	Silhueta da função $F(x, y, z, w) = w - (x^2 + y^2 + z^2 + R^2 - r^2)^2 + 4R^2(x^2 + y^2)$ com $R = 2$ e $r = 1$, nos ângulos: esquerda ($\theta_1 = 0.75, \theta_2 = 0.75, \theta_3 = 0.0, \theta_4 = 0.0, \theta_5 = 0.0, \theta_6 = 0.0$) e direita ($\theta_1 = 0.75, \theta_2 = 0.75, \theta_3 = 0.75, \theta_4 = 0.75, \theta_5 = 0.0, \theta_6 = 0.0$).	39
3.11	Bordo do domínio da função $F(x, y, z, w) = w - (x^2 + y^2 + z^2 + R^2 - r^2)^2 + 4R^2(x^2 + y^2)$ com $R = 2$ e $r = 1$, nos ângulos ($\theta_1 = 1.1, \theta_2 = 1.1, \theta_3 = 1.1, \theta_4 = 1.1, \theta_5 = 1.1, \theta_6 = 1.1$), com 10 raios.	40

- 3.12 Processo de iluminação, começando da esquerda: iluminação só com fontes de luz, só desenhando a silhueta, só desenhando o bordo e a iluminação completa, variedade da função $F(x, y, z, w) = w - (x^2 + y^2 + z^2 + R^2 - r^2)^2 + 4R^2(x^2 + y^2)$ com $R = 2$ e $r = 1$, nos ângulos $(\theta_1 = 1.13, \theta_2 = 1.13, \theta_3 = 1.13, \theta_4 = 0.75, \theta_5 = 0.75, \theta_6 = 0.75)$, com 100 raios. 40
- 4.1 Visualização das variedades implícitas da Tabela 4.1, em diferentes ângulos, para as funções 1 até 11, na ordem da esquerda para a direita. 42
- 4.2 Qualidade da função 1, variando a quantidade de raios: 10, 15, 20 e 30 raios por cada pixel. 44
- 4.3 Qualidade da função 7, variando a quantidade de raios: 10, 15, 20 e 30 raios por cada pixel. 44
- 4.4 Qualidade da função 5, variando a quantidade de intervalos (7,10,15,20) e de raios por cada pixel, 10 raios para a primeira fila e 30 raios para a segunda fila. 44
- 4.5 Qualidade da função 6, variando a quantidade de intervalos (7,10,15,20) e de raios por cada pixel, 10 raios para a primeira fila e 30 raios para a segunda fila. 46
- 4.6 Qualidade da função 10, variando o nível máximo do árvore e de raios por cada pixel. A primeira fila: nível 0,1 e 2 com 10 raios. Segunda fila, começando da esquerda: nível 0 e 1 com 20 raios, nível 0 e 1 com 30 raios. 46
- 4.7 Qualidade da função 11, variando o nível do árvore e de raios por cada pixel. A primeira fila: nível 0,1 e 2 com 10 raios. Segunda fila, começando da esquerda: nível 0 e 1 com 20 raios, nível 0 e 1 com 30 raios. 47
- 4.8 Qualidade da função 10, variando o nível da árvore, a quantidade de intervalos e o número de raios por cada pixel. A primeira fila: nível 1, da esquerda com 5 intervalos e as da direita com 7 intervalos para 10 e 20 raios. Segunda fila: nível 2, da esquerda com 2 intervalos e as da direita com 4 intervalos para 10 e 20 raios. 47
- 4.9 Sequencia de imagens geradas pela função 1, nos ângulos da Tabela 4.6, lançado com 10 raios por pixel. 50
- 4.10 Imagens geradas em CPU pela função 7 da Tabela 4.1, nos ângulos 8, 10 e 14 da Tabela 4.6. A primeira fila com 50 raios, a segunda fila com 75 raios, a terceira fila 100 raios. 51
- 4.11 Qualidade da função 11 com *Ray Casting* em CPU. A primeira fila com 50 raios e a segunda fila com 100 raios. 51

Lista de tabelas

4.1	Funções implícitas de teste.	41
4.2	Frames por segundo obtidos variando a quantidade de raios lançados desde cada pixel.	43
4.3	Frames por segundo obtidos variando a quantidade de intervalos, para 10 e 30 raios lançados desde cada pixel.	45
4.4	Frames por segundo obtidos variando o nível do árvore, para 10, 20 e 30 raios lançados desde cada pixel.	45
4.5	Frames por segundo obtidos variando o nível do árvore e a quantidade de intervalos, para 10 e 20 raios lançados desde cada pixel.	48
4.6	Ângulos usados na seção variando os ângulos 4.2.	49
4.7	Tempo de processamento em milissegundos do <i>Ray Casting</i> em CPU, para diferentes funções com diferentes níveis da árvore e diferente quantidades de raios.	50