



Rodrigo Marques Almeida da Silva

**Um Método Otimizado de Renderização
Fotorealista com Distribuição Estatística e
Seleção Automática de Técnicas**

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção
do grau de Doutor pelo Programa de Pós-graduação em
Informática do Departamento de Informática da PUC-Rio

Orientador: Prof. Bruno Feijó

Rio de Janeiro
Março de 2015



Rodrigo Marques Almeida da Silva

**Um Método Otimizado de Renderização
Fotorealista com Distribuição Estatística e
Seleção Automática de Técnicas**

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

Prof. Bruno Feijó

Orientador

Departamento de Informática – PUC-Rio

Prof. Hélio Cortes Vieira Lopes

Departamento de Informática – PUC-Rio

Prof. Markus Endler

Departamento de Informática – PUC-Rio

Prof. Soraia Raupp Musse

Departamento de Informática – PUC-RS

Prof. Luiz Eduardo Azambuja Sauerbronn

Departamento de Informática – UFRJ

Prof. José Eugênio Leal

Coordenador(a) Setorial do Centro

Técnico Científico - PUC-Rio

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

Rodrigo Marques Almeida da Silva

Graduou-se em Engenharia de Computação pela Universidade Federal do Espírito Santo em 2008, obteve o título de Mestre em Informática pela PUC-Rio em 2010. Desde 2009 trabalha no VisionLab desenvolvendo sistemas de realidade virtual e aumentada e visualização científica, e desde 2010 trabalha como Pesquisador na equipe de P&D em Efeitos Visuais do Grupo Globo.

Ficha Catalográfica

Silva, Rodrigo Marques Almeida da

Um método otimizado de renderização fotorealista com distribuição estatística e seleção automática de técnicas / Rodrigo Marques Almeida da Silva ; orientador: Bruno Feijó. – 2015.

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

Tese (doutorado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2015.

Inclui bibliografia

1. Informática – Teses. 2. Renderização Fotorealista. 3. Fila de Renderização. 4. Traçado de Raios. 5. Computação em Nuvem. 6. Árvores de Shade. I. Feijó, Bruno. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

*A meu pai (in memorian), a minha mãe, minha esposa, meus irmãos (inclui
Pablo), meus avós e tios e primos.*

Agradecimentos

Ao meu pai, Clésio Marques da Silva (in memoriam), que sempre apostou no estudo e me proporcionou a força e a vontade de realizar um objetivo tão grande.

À minha família e amigos, pelo grande apoio, sem os quais este trabalho não poderia ter sido realizado.

À minha esposa Ana Paula Sperandio, por sempre me dar força para lutar.

Aos companheiros de luta, em especial a Pablo Bioni, irmão que me apoiou desde o início nessa jornada.

Ao meu orientador Bruno Feijó, pela grande dedicação, apoio e incentivo à pesquisa, sem o qual, esse trabalho não seria possível.

Ao VisionLab/PUC-Rio pelos auxílios concedidos e as oportunidades dentro do projeto com a TV Globo.

À TV Globo e, em destaque, ao departamento de P&D em VFX, pelo suporte e acesso a equipamentos especiais, sem os quais o trabalho não teria alcançado ganho prático.

Ao CNPQ pelo apoio financeiro.

À FINEP pelas facilidades de infraestrutura na PUC-Rio.

Resumo

Silva, Rodrigo Marques Almeida da; Feijó, Bruno (Orientador). **Um Método Otimizado de Renderização Fotorealista com Distribuição Estatística e Seleção Automática de Técnicas**. Rio de Janeiro, 2015. 180p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

O processo de renderização fotoreal para cinema e TV demanda, cada vez mais, poder de processamento, necessitando não só de algoritmos paralelos, bem como sistema de distribuição de tarefas. No entanto, mesmo em sistemas de produção, o tempo necessário para avaliar uma animação pode chegar a vários dias, dificultando a melhoria da qualidade artística e limitando alterações. Neste trabalho foca-se na otimização de três processos inerentes à renderização, a renderização local, na qual o sistema trabalha para renderizar um conjunto de pixels de forma otimizada, aproveitando os recursos de hardware disponíveis e aproveitando dados de renderizações previamente realizadas, pelo nó ou teste. O processo de gerenciamento, extremamente crítico para o resultado, é alterado para não só distribuir, mas analisar toda a infraestrutura de renderização, otimizando o processo de distribuição e permitindo o estabelecimento de metas como prazo e custo. Além disso, o modelo de gerenciamento é expandido para a nuvem, utilizando-a como transbordo de processamento. Ainda, um novo processo foi criado para avaliar a renderização de forma colaborativa, onde cada nó comunica resultados parciais e assim otimiza a renderização de outros. Por fim, diversas técnicas inovadoras foram criadas para melhorar o processo como um todo, removendo desperdícios e reaproveitando trabalho.

Palavras-chave

Renderização Fotorealista; Fila de Renderização; Traçado de Raios; Computação em Nuvem; Árvores de Shade

Abstract

Silva, Rodrigo Marques Almeida da; Feijó, Bruno (Advisor). **An Optimized Photorealistic Rendering Method with Statistic Distribution and Automatic Rendering Technique Selection.** Rio de Janeiro, 2015. 180p. DSc. Thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The photorealistic rendering process for cinema and TV increasingly demands processing power, requiring fast parallel algorithms and effective task distribution systems. However, the processes currently used by the academia and by the industry still consume several days to evaluate an animation in super-resolution (typically 8K), what makes difficult the improvement of artistic quality and limits the number of experiments with scene parameters. In this work, we focus on the optimization of three processes involved in photorealistic rendering, reducing the total time of rendering substantially. Firstly, we optimize the local rendering, in which the system works to render a set of pixels optimally, taking advantage of the available hardware resources and using previous rendering data. Secondly, we optimize the management process, which is changed not only to distribute frames but also to analyze all the rendering infrastructure, optimizing the distribution process and allowing the establishment of goals as time and cost. Furthermore, the management model is expanded to the cloud, using the cloud as a processing overflow. Thirdly, we propose a new optimized process to evaluate the rendering task collaboratively, where each node communicates partial results to other nodes, allowing the optimization of the rendering process in all nodes. Altogether, this thesis is an innovative proposal to improve the whole process of high-performance rendering, removing waste of resources and reducing rework.

Keywords

Photorealistic Rendering; Render Queue; Ray Tracing; Cloud Computing; Shade Tree

Sumário

Sumário	8
Lista de figuras	11
Lista de tabelas	16
Lista de equações	17
1. Introdução	19
1.1. Objetivos	22
1.2. Estrutura da Tese	26
2. Conceitos e Trabalhos Relacionados.....	28
2.1 Elementos Básicos para Renderização Fotorealista.....	28
2.2 Aceleração de Renderização em Alta Resolução	33
2.3 Metodologia e Gerenciamento de Renderização	41
3. Otimização em Nível de Execução	48
3.1. O Processo de Renderização	48
3.1.1. Complexidade Geométrica.....	51
3.1.2. Complexidade de Material e Luzes.....	52
3.1.3. Subdivisão baseada em custo	54
3.1.4. Aproveitamento de Custos.....	56
3.1.5. Otimização na Renderização	62
3.1.6. Otimização de Shade Tree	63
3.1.6.1. Geração de Material	65
3.1.7. Renderização Multitécnica	69

3.1.7.1.	Combinação de Técnicas	73
3.1.8.	Cache.....	74
3.2.	Arquitetura do Renderizador	77
3.2.1.	Rotina Principal do Renderizador	81
4.	Otimização em Nível Gerencial	84
4.1.	Análise de Renderização	84
4.1.1.	Inventário de Nós	86
4.2.	Arquitetura de Renderização em Cluster	87
4.2.1.	Gerenciador	90
4.2.2.	Processo de Inicialização de Renderização.....	92
4.2.3.	Distribuição sem restrição temporal	94
4.2.4.	Distribuição com restrição temporal	98
4.3.	Arquitetura de Renderização em Nuvem	103
4.3.1.	Sincronização de Bases.....	105
4.3.2.	Distribuição com restrição temporal	107
4.3.3.	Distribuição com restrição de custos	110
4.4.	Múltiplos Trabalhos	112
5.	Otimização Integrada	114
5.1.	Compartilhamento de Informações	115
5.1.1.	Método em Pré-Cálculo	117
5.1.2.	Método em Execução	118
5.1.3.	Dump de dados.....	119
6.	Resultados	120
6.1.	Metodologia de Teste de Desempenho.....	120
6.2.	Metodologia de Análise Estatística	127

6.3.	Testes	129
6.3.1.	Desempenho do Renderizador	129
6.3.2.	Qualidade de Redução de Técnica	139
6.3.3.	Desempenho do Gerenciador	141
6.3.4.	Desempenho do Gerenciador com Colaboração	147
6.3.5.	Desempenho de Transbordo para Nuvem	150
6.3.6.	Desempenho GPU x CPU.....	153
6.4.	Finalização	156
7.	Conclusão e Trabalhos Futuros	158
7.1	Conclusões Finais.....	158
7.2	Trabalhos Futuros	161
	Referências Bibliográficas	163
	Glossário	174

Lista de figuras

Figura 1: A referência “Utah Fairy”(Wald,I. 2010) nas versões:

(a) ray tracing em tempo real , 180K triângulos, 1024 x 1024,

1 única fonte de luz e aprox. 4 *fps* em dual core 64 bits

(figura extraída de Wald *et al.*, 2007);

(b) ray tracing de imagem HDTV, 1280 x 720, wide-screen,

com desfocagem e motion blur, aprox. 6M micropolígonos,

aprox. 60 seg em GPU (para um único frame)

(figura extraída de Hou *et al.*, 2010).....20

Figura 2: Cena complexa com qualidade de cinema,

com 678 milhões de triângulos, 106 minutos de rendering,

em um Apple G5 com 2 processadores PowerPC de 2 GHz

(figura extraída de Christensen et al. (2006)).....21

Figura 3: Inter-reflexão com uso de ray tracing no filme “Cars

(Pixar, 2006)”: (a) reflexo dos olhos e vidro frontal ;

(b) reflexão entre partes cromadas. Figura composta a

partir de Christensen et al. (2006).....30

Figura 4: Sombras bem definidas e detalhadas geradas

por ray tracing, do filme “Cars (Pixar, 2006)” contendo

1000 fontes de luz (figura extraída de Christensen et al. (2006)).....31

Figura 5: Oclusão de ambiente gerado por ray tracing

em imagens de alta qualidade (figura extraída de

Christensen et al. (2006)).....32

Figura 6: Exemplo da utilização do *Fakeosity* (exemplo produzido).....33

Figura 7: Comparação entre o resultado do *RayTracing*

e *Virtual Point Light* a) resultado obtido com *ray tracing*

com iluminação global. b) resultado obtido com VPL.

(Imagens extraídas de Walter et al (2012))34

Figura 8: Virtual Point Light simula a influencia de uma

luz através de pontos adicionais de luz, já o Virtual Ray

Light utiliza uma interpolação de todo o trajeto entre os

pontos para simular a interação, sendo assim, muito mais denso. (imagem de Novàk et al (2012)).....	35
Figura 9: Estruturas Hierárquicas:	
a) Grade Regular (Fernando, 2004)	
b) Kd-Tree (exemplo produzido) c) Octree (Pharr, 2005).....	37
Figura 10: Sombras suaves utilizando amostragem de Monte Carlo (a) baixa amostragem (b) alta amostragem (imagens produzidas)	
	39
Figura 11: Shade tree e código correspondente na RenderMan (Pixar, 1970)	
	42
Figura 12: Sistema de criação de Shade Trees desenvolvido nesta pesquisa	
	44
Figura 13: Espectro de cor visível e o gamut de câmeras e televisores (Gaggioni et al, 2012).....	
	45
Figura 14: Workflow de Renderização (Rasterização) (imagem produzida)	
	48
Figura 15: Problemas na subdivisão da renderização. Cena do Maracanã, gerado pelo autor para a novela Avenida Brasil (TV Globo, 2012). Imagem reproduzida sob a política de “fair use”	
	49
Figura 16: Projeção de nós da BVH (imagem produzida)	
	51
Figura 17: Subdivisão utilizada por Abraham et al (2004).....	
	54
Figura 18: Subdivisão utilizada pelo sistema (imagem produzida (algoritmo))	
	56
Figura 19: Imagem de Performance: cada pixel de um frame tem as informações de desempenho armazenadas (#Rays, CPU Time, #MC Samples, Stack Size, #Texture Samples, Memory, #Ray misses). Estas informações permitem a subdivisão otimizada do frame (exemplificada com as opções Quadtree e kd-tree).	
(imagem produzida (algoritmo))	57
Figura 20: Compensação da Imagem de Performance (imagem produzida (algoritmo))	
	58

Figura 21: Sistema de Criação de Shade Trees Desenvolvido	64
Figura 22: Criação de Bloco Composto.....	66
Figura 23: Algoritmo de Geração de Material (image produzida)	67
Figura 24: Diferenças entre técnicas, usando a cena “San Miguel” (Barringer R. et al, 2014).....	71
Figura 25: Blend Zones (imagem produzida)	73
Figura 26: Múltiplas técnicas técnicas usadas em uma cena de Saramandaia (TV Globo, 2014). Imagem reproduzida sob a política de <i>fair use</i>	74
Figura 27: <i>Irradiance Cache</i> (Krivaneck et al., 2009).....	75
Figura 28: Amostragem dos Irradiance Caches (Krivaneck et al., 2009)	76
Figura 29: Arquitetura do sistema local proposto (imagem produzida)	78
Figura 30: Modelo básico de dados do banco de dados (imagem produzida)	78
Figura 31: Interface <i>MaxScript</i> , usando a cena “San Miguel” (Barringer R. et al, 2014).....	80
Figura 32: Fluxograma de operação do sistema proposto	82
Figura 33: Processo convencional de distribuição de frames	85
Figura 34 : Arquitetura de alto nível do Gerenciador e do Sistema de Controle dos Nós (imagem produzida)	88
Figura 35: Interface Web do Gerenciador (imagem do sistema).....	90
Figura 36: Estrutura de Dados do Servidores SQL (imagem produzida)	91
Figura 37: Campos de Submissão (imagem do sistema).....	92
Figura 38: Inicialização da renderização de cenas (imagem produzida)	93
Figura 39: Classificação e categorização de frames (imagem produzida)	95
Figura 40: Ordem de execução do bloco de frames (imagem produzida)	96

Figura 41: Fluxo de gerenciamento dos <i>frames</i> (sem restrições) (imagem produzida)	98
Figura 42: a) Esquemático do algoritmo de subdivisão de frame. b) Imagem de Performance sendo subdividida pelo BSP. c) Máscaras de renderização (branco habilita renderização, preto impede renderização)	100
Figura 43: Algoritmo de gerenciamento com restrição temporal (imagem produzida).....	102
Figura 44: Arquitetura convencional de transbordo para a nuvem	104
Figura 45: Cloudberry, sistema de sincronização de bases.	107
Figura 46: Processo de distribuição na nuvem com restrição temporal.....	110
Figura 47: Alocação de nós com restrição de custo (imagem produzida)	111
Figura 48: Exemplo de escalonamento usando o EDF – Earliest Deadline First, extraído de (Tanenbaum et al, 2004)	113
Figura 49: Grafo de frames proposto para a renderização colaborativa (imagem produzida).....	116
Figura 50: Intervalo de análise em execução (imagem produzida)	118
Figura 51: Arquitetura de testes usando o Deadline e o MPI do HPCPack (imagem produzida).....	122
Figura 52: Cenas do teste a) cena “San Miguel” (Barringer R. et al, 2014) b) Cena do Maracanã, gerado pelo autor para a novela Avenida Brasil (TV Globo, 2012). Imagem reproduzida sob a política de “fair use”	124
Figura 53: Opções de linha de comando do sistema de avaliação	125
Figura 54: Hardware empregado nos testes, cedidos pela TV Globo para testes. a) Intel Xeon Phi b) Nós de renderização c) Storage Equallogic e NVidia VCA	126
Figura 55: Resultado Gráfico de Frames – Teste 1.1	130

Figura 56: Resultado Gráfico de Frames – Teste 1.2.....	131
Figura 57: Resultado Gráfico de Frames – Teste 2.1.....	132
Figura 58: Resultado Gráfico de Frames – Teste 2.2.....	133
Figura 59: Resultado Gráfico de Frames – Teste 2.3.....	134
Figura 60: Resultado Gráfico de Frames – Teste 2.4.....	135
Figura 61: Resultado Gráfico de Frames – Teste 3.1.....	136
Figura 62: Resultado Gráfico de Frames – Teste 3.2.....	137
Figura 63: Resultado Gráfico de Frames – Teste 4.1.....	140
Figura 64: Resultado Gráfico de Frames – Teste 5.1.....	142
Figura 65: Resultado Gráfico de Frames – Teste 5.2.....	144
Figura 66: Resultado Gráfico de Frames – Teste 6.1.....	145
Figura 67: Resultado Gráfico do Número de Servidores Alocados (Servidor x Tempo) – Teste 6.1	146
Figura 68: Resultado Gráfico de Frames – Teste 7.1.....	146
Figura 69: Resultado Gráfico do Número de Servidores Alocados (Servidor x Tempo) – Teste 7.1	147
Figura 70: Resultado Gráfico de Frames – Teste 8.1.....	148
Figura 71: Resultado Gráfico de Frames – Teste 9.1.....	149
Figura 72: Resultado Gráfico do Número de Servidores Alocados (Servidor x Tempo) – Teste 9.1	149
Figura 73: Resultado Gráfico de Frames – Teste 10.1.....	151
Figura 74: Alocação de Servidores – Teste 10.1.....	151
Figura 75: Resultado Gráfico de Frames – Teste 11.1.....	152
Figura 76: Alocação de Servidores – Teste 11.1.....	153
Figura 77: Resultado Gráfico de Frames – Teste 12.1.....	154
Figura 78: Resultado Gráfico de Frames – Teste 13.1.....	155

Lista de tabelas

Tabela 1: Resultados do tempo de renderização do Teste 1.1	130
Tabela 2: Resultados do tempo de renderização do Teste 1.2	131
Tabela 3: Resultados do tempo de renderização do Teste 2.1	132
Tabela 4: Resultados do tempo de renderização do Teste 2.2	133
Tabela 5: Resultados do tempo de renderização do Teste 2.3	134
Tabela 6: Resultados do tempo de renderização do Teste 2.4	135
Tabela 7: Resultados do tempo de renderização do Teste 3.1	136
Tabela 8: Resultados do tempo de renderização do Teste 1.2	137
Tabela 9: Resultados Gráficos Teste 1, 2 e 3 – Cena San Miguel.....	138
Tabela 10: Resultados Gráficos Teste 1, 2 e 3 – Cena Avenida Brasil.....	138
Tabela 11: Resultados do tempo de renderização do Teste 4.1	140
Tabela 12: Resultados Gráficos Teste 4.1	140
Tabela 13: Resultados do tempo de renderização do Teste 5.1	143
Tabela 14: Resultados do tempo de renderização do Teste 5.2	144
Tabela 15: Resultados do tempo de renderização do Teste 6.1	146
Tabela 16: Resultados do tempo de renderização do Teste 7.1	147
Tabela 17: Resultados do tempo de renderização do Teste 8.1	148
Tabela 18: Resultados do tempo de renderização do Teste 9.1	149
Tabela 19: Resultados do tempo de renderização do Teste 10.1	151
Tabela 20: Resultados do tempo de renderização do Teste 11.1	153
Tabela 21: Resultados do tempo de renderização do Teste 12.1	154
Tabela 22: Resultados do tempo de renderização do Teste 13.1	155

Lista de equações

Equação 1: Fórmula de custo inicial de um <i>pixel</i>	50
Equação 2: Fórmula de atualização de <i>kgeometria</i>	50
Equação 3: Fórmula de atualização de <i>kmaterial</i>	51
Equação 4: Fórmula de custo geométrico de um <i>pixel</i>	52
Equação 5: Fórmula de custo de material de um <i>pixel</i>	53
Equação 6: Fatores de Refração e Reflexão.....	53
Equação 7: Fórmula da computação da imagem de performance resultante	59
Equação 8: Fórmula de custo de um pixel considerando a imagem de performance.....	60
Equação 9: Fórmula da atualização de <i>kperfmagem</i>	60
Equação 10: Limite da atualização de <i>kperfmagem</i>	61
Equação 11: Fórmula da atualização de <i>kimg</i>	62
Equação 12: Estimativa de Consumo de Memória.....	94
Equação 13: Estimativa de performace dos servidores	97
Equação 14: Estimativa do número de servidores a serem alocados na nuvem	108
Equação 15: Estimador de tempo por pixel no VRay	123
Equação 16: Estimador de media	127
Equação 17: Estimador de variância.....	128
Equação 18: Cálculo do intervalo de confiança.....	129

“Há momentos em que a maior sabedoria é parecer não saber nada.”

Sun Tzu, A Arte da Guerra.