**ESCOLA POLITÉCNICA DA
UNIVERSIDADE DE SÃO PAULO**

EDUARDO TOLEDO SANTOS

**EXTENSÕES AO ALGORITMO DE
"RAY TRACING" PARAMETRIZADO**

Tese apresentada à Escola Politécnica
da Universidade de São Paulo para
obtenção do título de Doutor em
Engenharia.

São Paulo
1998

EDUARDO TOLEDO SANTOS

Eng. Eletricista, Mod. Eletrônica, Escola Politécnica da USP, 1988

Mestre em Engenharia Elétrica, Sistemas Digitais, Escola Politécnica da USP, 1994

EXTENSÕES AO ALGORITMO DE "RAY TRACING" PARAMETRIZADO

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Doutor em Engenharia.

Área de Concentração:
Sistemas Eletrônicos

Orientador:
Prof. Dr. JOÃO ANTONIO ZUFFO

Santos, Eduardo Toledo
Extensões ao Algoritmo de "Ray Tracing" Parametrizado.
São Paulo, 1998.
132p.

Tese (Doutoramento) - Escola Politécnica da Universidade
de São Paulo. Departamento de Engenharia Eletrônica.

1. Computação Gráfica 2. Computação Paralela I.
Universidade de São Paulo. Escola Politécnica. Depto. de
Engenharia Eletrônica II. t

À Laura, com amor.

AGRADECIMENTOS

Ao Prof. Dr. João Antonio Zuffo , meu orientador neste trabalho.

À FINEP pelo financiamento parcial deste projeto.

Ao Prof. Dr. Sérgio Takeo Kofuji pelo apoio e colaboração.

Aos estagiários Fernando Ferreira Dias Martins, Esteban Walter Gonzalez Clua, Jan Leduc de Lara e, especialmente, Márcio Grossi de Almeida, responsáveis por grande parte da programação do projeto RT(p).

À Tania Fraga pela colaboração, trabalho conjunto e construção do modelo geométrico usado nos testes.

Aos meus colegas e amigos do Laboratório de Sistemas Integráveis pelas sugestões e pelo indispensável incentivo recebido durante o desenvolvimento desta pesquisa.

Aos colegas e amigos do Departamento de Engenharia de Construção Civil da Escola Politécnica pelo grande apoio recebido durante o período de meu doutoramento.

Ao Laboratório de Sistemas Integráveis pelo suporte técnico e disponibilização da infra-estrutura.

Aos meus pais pelo infalível apoio.

À Laura , pelo carinho e companhia e por suportar o meu mal humor de fim de tese.

A todos que colaboraram com discussões, sugestões e incentivos para o sucesso deste trabalho, meu muito obrigado !

SUMÁRIO

LISTA DE FIGURAS	viii
LISTA DE TABELAS	x
LISTA DE ABREVIATURAS	xi
LISTA DE SÍMBOLOS	xii
RESUMO	xiii
"ABSTRACT"	xiv
1 Introdução	2
1.1 Motivação	3
1.2 Objetivos	5
1.3 Contribuições do trabalho	6
1.4 Organização do trabalho	6
2 Ray tracing parametrizado	9
2.1 Ray tracing	9
2.2 O algoritmo de ray tracing parametrizado	14
2.2.1 Otimização do algoritmo	15
2.2.1.1 Compartilhamento de expressões	15
2.2.1.2 Codificação "run lenght"	16
2.2.1.3 Atualização seletiva de expressões	16
2.2.1.4 Codificação "run lenght" 2D	16
2.2.1.5 Compartilhamento de expressões com "hashing"	16
2.2.1.6 Atualização unitária de expressões	17
2.3 Outras técnicas relacionadas	17
2.3.1 Ray tracing incremental	17
2.3.2 Estruturas hierárquicas para manipulação interativa	19
3 O Sistema RT(p)	21
3.1 Histórico	21
3.2 Arquitetura do sistema	26
3.2.1 Componentes	27
3.2.1.1 Applet rtpweb	27
3.2.1.2 Master	29
3.2.1.3 Slave	30
3.2.1.4 Sserver (Shadow Server)	31
3.2.1.5 Shader "gera_rtp"	31
3.2.1.6 Http Server	31
3.3 Exemplos	32
3.4 Conclusão	32
4 Movimentação de fontes de luz	35
4.1 Definição do problema	35
4.2 Cálculo rápido de produtos escalares para equação de iluminação	36
4.3 Determinação das fontes de luz incidentes no ponto de intersecção	38
4.3.1 O algoritmo "light buffer"	38
4.3.2 Um algoritmo auxiliado por acelerador gráfico	40
4.3.2.1 Demanda de memória	41
4.3.2.2 Sobrecarga de processamento	42
4.3.3 Um algoritmo melhorado	42
4.4 Implementação	43
4.5 Metodologia de avaliação	44
4.6 Resultados e análises	46
4.6.1 Tempo de recálculo de sombras	46
4.6.2 Tempo de comunicação para recálculo de sombras	50
4.6.3 Variação do limiar de comparação de profundidade (Z-buffer)	51
4.6.4 Influência da resolução espacial do Z-buffer na sombra projetada	55
4.6.5 Uso de acelerador gráfico	57
4.6.6 Comparação do tempo de reavaliação com o tempo de síntese	58
4.7 Conclusões	58

5 Efeitos especiais.....	61
5.1 Luzes spot.....	61
5.1.1 Parametrização de luzes spot.....	63
5.1.2 Implementação e resultados.....	63
5.2 Projetores.....	65
5.2.1 Parametrização de projetores.....	65
5.3 Mapeamento de Relevô.....	66
5.3.1 Parametrização de mapeamento de relevô.....	67
6 Otimizações.....	70
6.1 Paralelização.....	70
6.1.1 Estratégia.....	71
6.1.2 Metodologia de Avaliação.....	73
6.1.3 Resultados e Análises.....	75
6.1.3.1 Tempo de processamento.....	76
6.1.3.2 Tempo de comunicação.....	77
6.2 Reavaliação incremental.....	78
6.2.1 Implementação.....	79
6.2.1.1 Recálculo seletivo de píxeis.....	79
6.2.2 Resultados e análises.....	80
6.3 Conclusões.....	84
7 Conclusões e trabalhos futuros.....	86
7.1 Conclusões.....	86
7.2 Trabalhos futuros.....	86
7.2.1 Compressão de dados.....	86
7.2.2 Geração de imagens panorâmicas e sistemas de imersão.....	87
7.2.3 Extensão das idéias para outros algoritmos.....	88
7.2.4 Formato de imagem com compressão incremental.....	88
7.2.5 Melhoria da interface.....	88
7.2.6 Parametrização do coeficiente de refração.....	89
7.2.7 Reaproveitamento de resultados parciais nas equações.....	89
7.2.8 Implementação paralela e local do servidor de sombras.....	89
Referências Bibliográficas.....	90
Anexo A: Um “shader” RT(p) para o Softimage/mental ray.....	96
A.1 Introdução.....	96
A.2 Implementação.....	97
Anexo B: StereoRTP – Uma aplicação em estereografia.....	99
B.1 Introdução.....	99
B.2 Implementação.....	100
Anexo C: Dados coletados.....	103
ENSAIO 1 - Tempos de processamento para recálculo de sombra (Cena Bule).....	103
ENSAIO 2 - Tempos de processamento para recálculo de sombra (Cena Interior).....	106
ENSAIO 3 - Tempos de comunicação sserver / master (Cena Bule).....	107
ENSAIO 4 - Acelerador Gráfico (Cena: Bule300).....	108
ENSAIO 5 - Processamento paralelo (Cena: Bule500).....	109
ENSAIO 6 - Processamento paralelo (Cena: Bule1000x250).....	111
ENSAIO 7 - Processamento paralelo (Cena: Bule250x1000).....	112
ENSAIO 8 - Processamento paralelo (Cena: InteriorT4).....	113
ENSAIO 9 - Tempo de reavaliação completa (Cena: Interior).....	115
ENSAIO 10 - Tempo de reavaliação incremental (Cena: Interior).....	116
ENSAIO 11 - Total de píxeis influenciados por material (Cena: Interior).....	118

LISTA DE FIGURAS

FIG. 2.1 - Elementos do cenário de ray tracing	11
FIG. 2.2 - Cálculo do raio refletido	12
FIG. 2.3 - Cálculo do raio transmitido.....	13
FIG. 2.4 - O algoritmo de ray tracing	13
FIG. 2.5 - Árvore de raios do píxel da figura 2.1	14
FIG. 3.1 - Logotipo do Projeto RT(p).....	21
FIG. 3.2 - Interface Motif™ do sistema RT(p).....	22
FIG. 3.3 - Interface HTML do sistema RT(p)	23
FIG. 3.4 - Interface Java do sistema RT(p).....	24
FIG. 3.5 - Nova interface Java do sistema RT(p) - painel de cores	26
FIG. 3.6 - Nova interface Java do sistema RT(p) - painel de luzes spot.....	26
FIG. 3.7 - Relacionamento entre os componentes do sistema RT(p)	32
FIG. 3.8 - Sequência de imagens geradas com o RT(p)	33
FIG. 3.9 - Imagens demonstrando aplicação de texturas no RT(p)	33
FIG. 4.1 - Vetores da equação de iluminação em coordenadas esféricas	37
FIG. 4.2 - O cubo direcional.....	38
FIG. 4.3 - Cena "Bule"	44
FIG. 4.4 - Cena "Interior"	45
FIG. 4.5 - Cubo direcional rebatido.....	46
FIG. 4.6 - Tempo de recálculo de sombras x resolução Z-buffer (Cena "Bule").....	47
FIG. 4.7 - Tempo de recálculo de sombras x resolução Z-buffer (Cena "Interior")	47
FIG. 4.8 - Tempo de recálculo de sombras em função do num. de <i>hit points</i> (Cena "Bule").....	48
FIG. 4.9 - Tempo de recal. de sombras x num. de <i>hit points</i> (Cena "Interior")	48
FIG. 4.10 - Fluxo de processamento do componente sserver	49
FIG. 4.11 - Tempo de comunicação para envio de resultado do recálculo de sombra	50
FIG. 4.12 - Efeito de variação do limiar de comparação de profundidade (Eps_{abs}).....	53
FIG. 4.13 - Rel. entre a prof. relativa e o valor no Z-buffer para várias razões <i>far/near</i>	54
FIG. 4.14 - Efeito da resolução espacial do Z-buffer no serrilhamento da sombra	55
FIG. 4.15 - Efeito da res. do Z-buffer no serrilhamento da sombra (300x300).....	56
FIG. 4.16 - Avaliação do impacto no recálculo de sombra com uso de acelerador gráf.....	57
FIG. 4.17 - Comparação RT convencional x RT Parametrizado.....	58
FIG. 5.1 - Diagrama goniométrico de uma fonte de luz	61
FIG. 5.2 - Variação da intensidade da luz conforme equação 5.1	62
FIG. 5.3 - Variação da posição de fonte de luz spot.....	64
FIG. 5.4 - Variação da direção da fonte de luz spot	64
FIG. 5.5 - Variação do ângulo de abertura do cone de fonte de luz spot.....	64
FIG. 5.6 - Variação do ângulo de máxima intensidade - fonte de luz spot.....	65
FIG. 5.7 - Projecção gobo e padrão projetado	65
FIG. 5.8 - Efeito de projecção e padrão projetado	66
FIG. 5.9 - Mapeamento de relevo	66
FIG. 5.10 - O algoritmo de mapeamento de relevo (1D)	67
FIG. 6.1 - Partic. de imagem com subdivisão unif. e alocação distr. a 4 processadores.....	70
FIG. 6.2 - Tempo de processamento para várias resoluções da cena Bule (1).....	75
FIG. 6.3 - Tempo de processamento para várias resoluções da cena Bule (2).....	75
FIG. 6.4 - Tempo de processamento para várias resoluções da cena Bule (3).....	75
FIG. 6.5 - Tempo de processamento paralelo para a cena "Interior"	76
FIG. 6.6 - Ganho de processamento paralelo - cena "Interior"	77
FIG. 6.7 - Gráfico para análise de latência para várias quantidades de comunicação	77
FIG. 6.8 - Gráfico para análise da taxa de transmissão	78
FIG. 6.9 - Codificação RLE da estrutura de reavaliação incremental	79
FIG. 6.10 - Correlação entre área e tempo de reavaliação - cena InteriorT1	81
FIG. 6.11 - Correlação entre área e tempo de reavaliação - cena InteriorT2.....	81
FIG. 6.12 - Correlação entre área e tempo de reavaliação - cena InteriorT3.....	82
FIG. 6.13 - Correlação entre área e tempo de reavaliação - cena InteriorT4.....	82
FIG. 6.14 - Comparação de tempos reavaliação completa x incremental.....	83

FIG. 6.15 - Ganho de velocidade com reavaliação incremental	83
FIG. A.0.1 - Arquitetura de pré-processamento do sistema RT(p).....	98
FIG. B.0.1 – Apresentação do sistema StereoRTP no SC'97	99
FIG. B.0.2 - Arquitetura de software do sistema StereoRTP.....	100
FIG. B.0.3 - Cartaz explicativo do sistema StereoRTP apresentado no SC'97	102

LISTA DE TABELAS

TAB. 3.1 - Parâmetros modificáveis no sistema RT(p).....	28
TAB. 4.1 - Características das cenas de teste	44
TAB. 4.2 - Características da cena "Bule" em várias resoluções.....	45
TAB. 4.3 - Características da cena "Interior"	45
TAB. 4.4 - Condições para ensaio do componente sserver (shadow server).....	46
TAB. 6.1- Características do sistema empregado nos testes de paralelização	73
TAB. 6.2 - Matriz de ensaios realizados.....	74
TAB. 6.3 - Número total de píxeis transmitidos em cada ensaio.....	74
TAB. 6.4 - Tam. médio e compressão da estrutura para reaval. incremental (Bule)	80
TAB. 6.5 - Tam. médio e compressão da estrutura para reaval. incr. (InteriorT4).....	80
TAB. B.0.1 - Alocação de componentes de software no sistema Aquila	100

LISTA DE ABREVIATURAS

NFF	Neutral File Format
PLB	Pixel-Light-Buffer
RLE	"Run-Length Encoding" - Codificação por Comprimento de Corrida
RT	"Ray Tracing"
RT ²	"Real Time Ray Tracing" - Ray Tracing em Tempo Real
RT(p)	Ray Tracing Parametrizado

LISTA DE SÍMBOLOS

I	intensidade do ponto amostrado no plano de imagem (R, G, e B)
k_d	coeficiente de reflexão difusa do material (componentes R, G e B)
$\bar{N}$	vetor unitário normal à superfície do objeto
$\vec{L}_j$	vetor na direção da j-ésima fonte de luz
$ \vec{L}_j $	intensidade da fonte de luz j
Ω	subconjunto das fontes de luz que iluminam um ponto
k_s	coeficiente de reflexão especular do material
$\vec{H}_j$	vetor unitário na direção bissetriz entre o observador e $\vec{L}_j$
n	expoente que representa o polimento da superfície
I_j	intensidade da j-ésima fonte de luz
I_r	intensidade do raio refletido
k_t	coeficiente de transmissão do material
I_t	intensidade do raio transmitido (refratado)
I_a	intensidade da iluminação ambiente global
η_i	coeficiente de refração do material i
η	índice de refração
mix	porcentagem de cor de textura na cor do material;
$c_{luz j}$	cor da j-ésima fonte de luz;
c_{obj}	cor difusa do material;
c_{tex}	cor da textura do material;
$\tilde{s}$	cor especular do material;
$\theta_L, \theta_N, \varphi_L$ e φ_N	coordenadas esféricas dos vetores $\vec{L}$ e $\vec{N}$
ω	tamanho do conjunto Ω
Z_{hit} :	distância do ponto testado à fonte de luz;
Eps :	limiar de teste Z-buffer (ajustável);
res_sb :	número de píxeis do shadow buffer;
n_vert :	número de vértices projetados;
n_hits :	número de pontos testados;

RESUMO

Ray tracing é um algoritmo para a síntese de imagens por computador. Suas características principais são a alta qualidade das imagens que proporciona (incorporando sombras, reflexões e transparências entre outros efeitos) e, por outro lado, a grande demanda em termos de processamento.

O *ray tracing parametrizado* é um algoritmo baseado no *ray tracing*, que permite a obtenção de imagens com a mesma qualidade a um custo computacional dezenas de vezes menor, porém com restrições. Estas restrições são a necessidade de geração de um arquivo de dados inicial, cujo tempo de processamento é pouco maior que o do *ray tracing* convencional e a não possibilidade de alteração de qualquer parâmetro geométrico da cena. Por outro lado, a geração de versões da mesma cena com mudanças nos parâmetros ópticos (cores, intensidades de luz, texturas, reflexões, transparências, etc.) é extremamente rápida.

Esta tese propõe extensões ao algoritmo de *ray tracing parametrizado*, procurando aliviar algumas de suas restrições. Estas extensões permitem alterar alguns parâmetros geométricos como a posição das fontes de luz, parâmetros de fontes de luz spot e mapeamento de revelo entre outros, mantendo o bom desempenho do algoritmo original.

Também é estudada a paralelização do algoritmo e outras formas de aceleração do processamento.

As extensões propostas permitem ampliar o campo de aplicação do algoritmo original incentivando sua adoção mais generalizada.

"ABSTRACT"

Ray tracing is an image synthesis computer algorithm. Its main features are the high quality of the generated images (which incorporate shadows, reflections and transparency, among other effects) and, on the other hand, a high processing demand.

Parameterized ray tracing is an algorithm based on ray tracing which allows the synthesis of images with the same quality but tens of times faster than ray tracing, although with some restrictions. These restrictions are the requirement of generating a data file (which takes a little longer than standard ray tracing to create) and the fact that no geometric modifications are allowed. On the other side, the processing time for creating new versions of the image with changes only on optical parameters (colors, light intensities, textures, reflections, transparencies, etc.) is extremely fast.

This Ph.D. dissertation proposes extensions to the parameterized ray tracing algorithm for diminishing its restrictions. These extensions allow changing some geometric parameters like the light source positions, spotlight parameters and bump-mapping among others, keeping the processing performance of the original algorithm.

The parallelization of the algorithm is also focused as well as other performance enhancements.

The proposed extensions enlarge the field of application of the original algorithm, encouraging more general adoption.

CAPÍTULO 1

INTRODUÇÃO

1 Introdução

A geração de imagens fotorealísticas (veja discussão deste termo em (SANTOS, 1994a - p.8)) tem inúmeras aplicações em diversas áreas da atividade humana, desde o desenho industrial até o entretenimento, passando por medicina, arquitetura, moda, engenharia, simulação, etc.

Infelizmente, as técnicas conhecidas para a geração deste tipo de imagem, notadamente o ray tracing (GLASSNER, 1989) e a radiosidade (COHEN, 1993) estão entre as que demandam o maior poder de processamento, requerendo até mesmo horas para a geração de uma única imagem em computadores modernos. Não fosse esta restrição, seu uso provavelmente seria dominante na maioria das aplicações da Computação Gráfica, dadas as grandes vantagens destes algoritmos e a qualidade das imagens produzidas.

Neste trabalho será abordado particularmente o algoritmo de ray tracing devido à sua capacidade de gerar corretamente efeitos de reflexão, refração e sombras e operar diretamente com a maioria das técnicas de modelamento geométrico. Refinamentos deste algoritmo permitem a obtenção de penumbras, profundidade de campo e reflexão global difusa, entre outros recursos que ampliam o realismo. Por outro lado, ao nosso ver, a técnica de radiosidade não apresenta nenhuma vantagem sobre as técnicas mais avançadas de ray tracing (WARD, 1994), (ASHDOWN & HAINES, 1996).

O processo de produção de imagens ou animações por meio da Computação Gráfica envolve, além de aspectos técnicos, algumas fases em que predomina o senso estético. Tipicamente, a escolha das cores e acabamento superficial dos objetos do cenário (texturização) e a posição, tipo, cor e intensidade das fontes de luz (iluminação) são feitas nestas etapas, num processo idealmente interativo e iterativo. Este processo pode ser difícil e tedioso (MARKS, 1997) devido ao empirismo do método e a lentidão de resposta à experimentação.

Claramente, a característica de interatividade dos processos de texturização e iluminação são prejudicadas pela grande demanda computacional do ray tracing. Também é inviável que, após a modificação de alguns parâmetros da cena, seja necessário um tempo demasiado longo para verificação dos resultados obtidos com as alterações, limitando seriamente o número de iterações possível até a obtenção de parâmetros satisfatórios.

Sendo necessária a qualidade final das imagens de *ray tracing*, as alternativas disponíveis são a geração de imagens em baixa resolução ou drástica redução no número e profundidade dos raios normalmente exigidos para processamento da imagem. Técnicas alternativas de síntese de imagens podem ser aplicadas apenas enquanto não são requeridos ajustes nas características de refração e reflexão especular das superfícies, caso em que apenas o *ray tracing* satisfaz.

Em 1989 foi introduzida a técnica denominada *ray tracing parametrizado* (SÉQUIN & SMYRL, 1989) com a finalidade de acelerar a geração iterativa de imagens, onde apenas parâmetros ópticos das superfícies da cena são modificados, mantendo a geometria inalterada.

Com o uso desta técnica torna-se possível interativamente modificar cores e texturas de uma cena sem qualquer tipo de degradação relativamente às imagens produzidas pelo algoritmo original de ray tracing. Este algoritmo permite aceleração de dezenas de vezes sobre o tempo de execução do ray tracing convencional, porém com duas restrições principais:

- é necessária a geração de uma imagem inicial, sujeita ao tempo de execução convencional do ray tracing;
- nenhum parâmetro geométrico da cena pode ser modificado, incluindo também um parâmetro óptico (os índices de refração dos materiais transparentes).

1.1 Motivação

A escolha do tema desta tese foi motivado por vários fatores. Dentre eles destacam-se:

- a) ray tracing é um algoritmo extremamente elegante.* A simplicidade do algoritmo de ray tracing confrontada com a alta qualidade das imagens produzidas pelo mesmo conferem a esta técnica uma elegância compartilhada por poucos algoritmos da Computação Gráfica. A recursividade implícita na geração da árvore de raios, a repetição do mesmo algoritmo básico no processamento dos raios primários, secundários e de sombra e a execução do mesmo processo em cada píxel da imagem contribuem significativamente para esta característica. Além disso, os resultados alcançados por ray tracing, um algoritmo baseado na Óptica Física, só podem ser simulados normalmente a custo de grande conhecimento técnico e intervenção do usuário (uso de técnicas de mapeamento de ambiente, por exemplo).

b) impossibilidade atual de obter-se taxas interativas com ray tracing.

Mesmo equipamentos especialmente construídos para o processamento de ray tracing (por exemplo, o Advanced Rendering Technologies RD64, com capacidade de cálculo de 5 bilhões de interseções/seg) ainda não são capazes de gerar, em tempo real, animações de cenas relativamente simples por ray tracing. O algoritmo explorado nesta tese permite a obtenção, dentro de certas limitações, de novas imagens de alta complexidade em décimos de segundos.

c) apesar das restrições (geometria fixa) o ray tracing parametrizado tem potencialmente muitas aplicações. Duas áreas de grande aplicação são o design industrial e de interiores (apresenta-se lista mais completa no capítulo 2). A simulação visual nestas duas áreas tem grande valor por acelerar o desenvolvimento e permitir a obtenção de melhores soluções. As extensões propostas neste trabalho vão estender ainda mais o campo de aplicação desta técnica, passando a incluir, por exemplo, luminotécnica e cenografia, além de melhorar aquelas já existentes.

d) o alto consumo de memória, a grande desvantagem do ray tracing parametrizado, cada vez menos é obstáculo. Com o grande barateamento dos módulos de memória RAM e a tendência de contínuo aumento em sua densidade de integração, a grande demanda de memória do algoritmo de ray tracing parametrizado, uma de suas mais sérias limitações, torna-se cada vez menos restritiva.

e) o algoritmo é altamente paralelizável. Em consonância com nossa tradição de pesquisa em alto desempenho, esta característica do algoritmo de ray tracing parametrizado é bastante apreciada. Neste caso a adequação à implementação paralela é ainda maior que a do ray tracing convencional que exploramos em profundidade em dissertação de mestrado (SANTOS, 1994a), não apresentando nenhum problema de replicação de dados.

f) o algoritmo ainda é pouco explorado. O algoritmo de ray tracing parametrizado passou vários anos “esquecido” e ainda hoje é pouco divulgado. Poucos produtos comerciais implementam esta tecnologia (por exemplo, TDI Explore/Alias Maya e, mais recentemente o Autodesk Picture This Home™). Este trabalho procura divulgar este algoritmo que consideramos valioso.

1.2 Objetivos

O objetivo deste trabalho é estender o algoritmo de ray tracing parametrizado em várias direções:

- a) permitir a movimentação das fontes de luz da cena (parâmetro geométrico);
- b) permitir a parametrização completa de luzes spot e projetores;
- c) permitir a parametrização de mapeamento de relevo (bump-mapping);

Os objetivos a) e b) acima abrirão novas áreas de aplicação para este algoritmo, principalmente em luminotécnica e cenografia teatral. A parte este fato, tal extensão representa a primeira proposta de alteração de algum tipo de parâmetro geométrico no contexto de ray tracing parametrizado.

O objetivo c), parametrização de mapeamento de relevo, além de propiciar maior realismo às cenas sintetizadas, significa também a possibilidade de mudança de outro parâmetro geométrico que, como já foi mencionado, é característica inovadora desta tese. O mapeamento de relevo pode, em alguns casos, amenizar a impossibilidade de alterar-se a geometria real dos modelos da cena, já que seu efeito visual aproximadamente equivale ao do mapeamento de deslocamento.

Além destes objetivos, também procura-se melhorar o desempenho no processamento do algoritmo:

- e) realizar uma implementação paralela do algoritmo;
- f) realizar uma implementação de reavaliação incremental;

A paralelização, seja em máquinas de memória compartilhada ou distribuída, é a forma pela qual pode-se atingir mais facilmente desempenho interativo, importante para aceitação e uso generalizado deste algoritmo.

A reavaliação incremental, uma nova proposta, procura reavaliar apenas um parâmetro por vez com grande ganho na velocidade percebida pelo usuário.

Como objetivos secundários, pretende-se:

- g) aumentar a divulgação desta técnica ;
- h) disponibilizar tecnologias que incentivem sua maior disseminação.

1.3 Contribuições do trabalho

As contribuições mais importantes deste trabalho são:

- a) desenvolvimento de extensões práticas e eficientes para o algoritmo de ray tracing parametrizado de forma a permitir:
 - a movimentação de fontes de luz nas cenas resintetizadas;
 - a parametrização completa de fontes de luz de tipo spot;
 - a parametrização de efeitos de projeção;
 - a parametrização de mapeamento de relevo;
- b) a proposta e implementação de versão paralela do algoritmo, atingindo-se taxas de re-síntese interativas;
- c) a proposta e implementação de uma versão incremental;
- d) a divulgação do algoritmo através da Internet, Mostras de Arte e Exposições, assim como através do desenvolvimento de novas aplicações.
- e) a proposta e implementação do sistema RT(p).

1.4 Organização do trabalho

Esta tese está organizada da seguinte forma: logo após esta introdução, apresenta-se no capítulo 2 conceitos básicos sobre os algoritmos de ray tracing e ray tracing parametrizado. As melhorias propostas por seus criadores são sumarizadas. Mostra-se também o atual estado da arte no campo da geração rápida de imagens com algoritmos de traçagem de raios.

O capítulo 3 descreve de forma sucinta o sistema RT(p), um conjunto de programas que implementam as inovações propostas neste trabalho, demonstram e divulgam o algoritmo.

No capítulo 4 apresenta-se uma proposta de extensão do algoritmo que permite, pela primeira vez, a alteração de um parâmetro geométrico em cenas de ray tracing parametrizado. A implementação desta extensão é apresentada e são mostrados e discutidos os resultados. O mesmo é feito no capítulo 5, onde nossa contribuição agora é em relação à parametrização de fontes de luz de tipo spot, efeitos de projeção e mapeamento de relevo.

No capítulo 6, são apresentadas propostas para otimizações do algoritmo em termos de velocidade. Estas otimizações são alcançadas através da paralelização e da

proposta de reavaliação incremental. As implementações realizadas e os resultados obtidos são discutidos.

Finalmente, o capítulo 7 traz conclusões gerais sobre a pesquisa desenvolvida e aponta trabalhos futuros que podem se seguir a este e/ou que o tenham como ponto de partida e referência.

Como anexos a esta tese, apresentam-se três seções abordando, respectivamente:

1. um “shader” desenvolvido para permitir o uso de ray tracing parametrizado com o pacote Softimage;
2. a implementação estereoscópica do RT(p) que permite a síntese e exibição de imagens com efeito estéreo;
3. tabelas com os dados coletados nos ensaios.

CAPÍTULO 2

RAY TRACING PARAMETRIZADO

2 Ray tracing parametrizado

A possibilidade de geração de imagens fotorealísticas em poucos segundos abre muitas áreas para novas aplicações da Computação Gráfica. Entre as principais aplicações do algoritmo de ray tracing parametrizado, pode-se citar:

- arquitetura (escolha de pisos e revestimentos, pintura, combinação cromática, etc.);
- *design* industrial (automóveis, eletrodomésticos, embarcações, móveis, etc.);
- luminotécnica (posição, tipo, cor e intensidade de fontes luminosas);
- decoração de interiores (cores, padronagem, etc.);
- moda (cores, padronagem);
- ensino de computação gráfica;
- animação por computador;
- simulação visual;
- efeitos especiais;
- síntese de imagens;
- realidade virtual;
- comércio eletrônico personalizado (*show-room* virtual);
- cenários virtuais.

A técnica de ray tracing parametrizado foi introduzida formalmente em 1989 pelo artigo (SÉQUIN & SMYRL, 1989).

Neste capítulo descreve-se este algoritmo como proposto originalmente pelos seus autores e outros trabalhos dele derivados.

Como esta técnica é fundamentada pelo algoritmo de ray tracing, apresenta-se na próxima seção uma breve descrição do mesmo.

2.1 Ray tracing¹

Introduzido no final da década de 60 (APPEL, 1968), o algoritmo de ray tracing certamente é o que permitiu sintetizar as imagens mais realísticas até hoje produzidas. Mesmo utilizando-se apenas o algoritmo básico (WHITTED, 1980) pode-se obter cenas de alta qualidade com inter-reflexões múltiplas, transparências/refração e sombras de forma bastante simples.

Todo processo de síntese de imagens resume-se em determinar a cor de cada píxel de uma imagem. O algoritmo de ray tracing faz esta determinação através do

¹ Adaptado de (SANTOS, 1994a)

modelo de iluminação proposto por Whitted (WHITTED, 1980), (HALL & GREENBERG, 1983):

$$I = k_d \sum_{j \in \Omega} (\bar{N} \cdot \bar{L}_j) + k_s \sum_{j \in \Omega} (\bar{N} \cdot \bar{H}_j)^n \cdot I_j + k_r I_r + k_t I_t + I_a \quad (2.1)$$

onde:	I	intensidade do ponto amostrado no plano de imagem (R, G, e B);
	k_d	coeficiente de reflexão difusa do material (componentes R, G e B);
	$\bar{N}$	vetor unitário normal à superfície do objeto;
	$\bar{L}_j$	vetor na direção da j-ésima fonte de luz, $ \bar{L}_j = \text{intensidade da luz}$;
	Ω	subconjunto das fontes de luz que iluminam o ponto;
	k_s	coeficiente de reflexão especular do material;
	$\bar{H}_j$	vetor unitário na direção bissetriz entre o observador e $\bar{L}_j$;
	n	expoente que representa o polimento da superfície;
	I_j	intensidade da j-ésima fonte de luz;
	I_r	intensidade do raio refletido;
	k_t	coeficiente de transmissão do material;
	I_t	intensidade do raio transmitido (refratado);
	I_a	intensidade da iluminação ambiente global ;

A cor de cada píxel é determinada individualmente através do cálculo da luz (I) que chega até o ponto de observação passando através dele. Esta luz, segundo o modelo de Whitted, tem duas componentes: local e global.

A componente local é constituída pela parcela de iluminação que, partindo das fontes de luz do cenário, atinge diretamente uma superfície visível ao observador. É composta por duas parcelas, difusa e especular, representadas, respectivamente, pelo primeiro e segundo termos da equação acima.

A componente global é representada pelos demais termos da equação 2.1 e seu cálculo será explicado mais adiante.

Fisicamente, existem "infinitos" raios de luz, partindo de cada fonte de luz, num cenário. A simulação de todos os raios é inviável computacionalmente. Por outro lado, somente uma parcela destes efetivamente contribui para a formação da imagem vista por um observador. Apenas os raios que são refletidos por objetos e atingem o "olho" deste deverão ser simulados para que se obtenha a imagem percebida por ele.

Para a síntese computacional de imagens usa-se o chamado "*ray tracing*" *reverso* ("*backward ray tracing*") que simula os raios de luz no sentido inverso, isto é, partindo do observador em direção ao cenário. Sabe-se que, pelas leis da Óptica, esta reversão não provoca alteração alguma na geometria envolvida. Devido a esta inversão ser tão comum, raramente se usa o adjetivo *reverso* e o nome "*ray tracing*"

passou a designar o "*ray tracing*" reverso. A designação "*ray tracing*" direto ("*forward ray tracing*") pode ser aplicada quando se deseja referir ao algoritmo que traça raios no sentido físico.

O algoritmo de ray tracing encerra uma rotina recursiva, aplicada independentemente para cada píxel da imagem. Seu primeiro passo é a determinação do ponto do cenário, visto através do píxel a processar, onde serão avaliadas as parcelas locais da iluminação (fig. 2.1).

FIG. 2.1 - Elementos do cenário de ray tracing

A versão do algoritmo denominada "exaustiva" (*exhaustive ray tracing*) calcula a intersecção de uma semi-reta (originada no ponto de observação e direcionada a cada píxel da tela) com *todos* os objetos presentes no cenário. Esta semi-reta é denominada "raio" pois simula um raio de luz que, partindo da superfície de um objeto atinge o observador. Como mencionado anteriormente, o algoritmo processa os raios de luz em sentido inverso, possibilitando que se considere somente aqueles raios que atingem o observador. Otimizações no algoritmo (GLASSNER, 1989) evitam o cálculo de intersecção com todos os objetos, acelerando a geração da imagem.

Determinados todos os pontos de intersecção do raio com os objetos do cenário, o mais próximo do observador é selecionado pois indica a superfície visível. Neste ponto é calculada a normal à superfície do objeto para uso na equação de iluminação.

A equação de iluminação (eq. 2.1) contém somatórias que consideram apenas as fontes de luz que iluminam o ponto de intersecção (conjunto Ω). A contribuição de fontes de luz que provocam sombra naquele ponto é nula e, portanto, não deve ser considerada. Para isto, é necessária a determinação de quais fontes efetivamente iluminam o ponto interceptado. Esta determinação é feita pelo lançamento de novos raios, denominados *raios de sombra*, com origem no ponto de intersecção e em direção à cada fonte de luz. A interceptação de qualquer objeto por um raio de sombra indica que a respectiva fonte de luz provoca sombra no ponto considerado. Dessa

forma, as sombras presentes na cena são elegantemente determinadas pelo algoritmo, usando uma rotina muito semelhante àquela para determinação de visibilidade.

Nesta etapa do algoritmo já se pode calcular completamente a componente local da luz que determina a cor de um dado píxel, incluindo suas parcelas difusa e especular. A imagem resultante deste cálculo para todos os píxeis é semelhante à obtida por algoritmos de tonalização do tipo Phong, a menos das sombras, em geral ausentes nestes últimos.

A componente global da luz é o grande diferencial do ray tracing e é responsável pelos efeitos de reflexão e refração exibidos em suas imagens. O cálculo da componente global é obtido através da luz que chega ao ponto de intersecção do raio pelas direções de reflexão (I_r) e refração (I_t). Estas são determinadas pela aplicação recursiva dos mesmos passos anteriores.

O cálculo do parâmetro I_r , luz global especular, é feito através do lançamento de um novo raio (secundário) na direção de reflexão dada pela equação 2.2, baseada nas propriedades de que os raios incidente e refletido formam um plano com a normal da superfície do objeto e que os ângulos de incidência e reflexão são iguais (fig. 2.2):

FIG. 2.2 - Cálculo do raio refletido

Já a parcela da luz global que é transmitida (refratada) pelo objeto pode ser determinada calculando-se a direção de refração pela Lei de Snell (eq. 2.3) e lançando-se um raio nesta direção (fig. 2.3).

Pela aplicação recursiva dos mesmos procedimentos, pode-se determinar a iluminação em cada ponto de intersecção dos raios que vão "ricocheteando" pelo cenário, formando assim a chamada "árvore de raios", associada ao píxel. A aplicação dos coeficientes de reflexão especular (k_s) e de transmissão (k_t) permitem determinar a parcela de contribuição de todos os raios da árvore de determinado píxel.

FIG. 2.3 - Cálculo do raio transmitido

Observa-se que o algoritmo de ray tracing é bastante simples pois seu núcleo é constituído por uma rotina reentrante de "lançamento de raios" que calcula a direção destes e sua intersecção com os objetos do cenário (fig. 2.4). A mesma rotina pode ser utilizada para o cálculo dos raios de sombra.

Pode-se encontrar bom material introdutório ao algoritmo de ray tracing nas seguintes referências: (WHITTED, 1980), (SCHIMITT et al., 1988), (KUCHKUDA, 1988) e (GLASSNER, 1989). (SPEER, 1992) apresenta uma extensa coletânea de referências sobre ray tracing.

```

ray_tracing()
  para cada pixel faça
    • calcule direção do raio primário (do observador ao pixel), prim ;
    • cor do pixel = lança_raio(prim);
  fim

lança_raio(raio)
  • intersec = det_intersecção(raio);
  • calcule iluminação local (ilum_local) no ponto de intersecção intersec;
  • calcule direção do raio refletido (refl) para gerar a componente especular;
  • calcule direção do raio refratado (refr) para gerar a componente transmitida;
  • comp_espec = lança_raio(refl);
  • comp_trans = lança_raio(refr);
  • cor_final = ilum_local + ks.comp_espec + kt.comp_trans;
  • retorne cor_final;
fim

det_intersecção(raio)
  • para cada objeto faça
    • calcule a intersecção do raio com objeto;
  • retorne a intersecção mais próxima da origem do raio;
fim

```

FIG. 2.4 - O algoritmo de ray tracing

2.2 O algoritmo de ray tracing parametrizado

A figura 2.1 exemplifica o trajeto percorrido por um raio genérico numa cena: partindo do observador em direção a determinado píxel, ao interceptar o primeiro objeto, serão gerados raios secundários. Dependendo das características do material do objeto interceptado, será disparado um raio na direção especular e/ou um raio na direção de refração. Este processo continua até que os raios secundários não interceptem nenhum outro objeto ou que seja atingido um número máximo de interações pré-estabelecido.

Para cada píxel, o raio primário, juntamente com os raios secundários correspondentes, formam a chamada “árvore de raios”. A figura 2.5 mostra a árvore de raios correspondente ao píxel indicado na figura 2.1.

FIG. 2.5 - Árvore de raios do píxel da figura 2.1

A cada nó da árvore de raios corresponde uma equação semelhante à eq. 2.1. A cor final de um píxel é determinada pela contribuição acumulativa de cada raio de sua árvore, de acordo com esta equação.

A equação de iluminação tem dois tipos de parâmetros: **geométricos** e **ópticos**. Parâmetros geométricos são aqueles que podem se alterar com a modificação da geometria da cena, incluindo a posição do observador e das fontes de luz. Os parâmetros ópticos não são alterados pela geometria da cena e correspondem às características de reflexão das superfícies dos objetos.

Sabe-se (WHITTED, 1980) que o cálculo de intersecções é a parte mais custosa do algoritmo de ray tracing. Se a geometria da cena permanecer inalterada, o caminho percorrido pelos raios no cenário se mantém constante mesmo que os parâmetros ópticos² dos materiais empregados sejam mudados.

A idéia por trás do algoritmo de ray tracing parametrizado é armazenar-se as árvores de raios de cada píxel de uma imagem e fazer uma reavaliação da equação correspondente à cor de cada píxel para cada mudança nos parâmetros ópticos,

² O coeficiente de refração, apesar de genericamente ser um parâmetro óptico, será aqui considerado como parâmetro geométrico já que sua alteração é capaz de mudar a trajetória dos raios refratados.

obtendo-se uma nova imagem ajustada. Como não se altera nenhum parâmetro geométrico, os termos com produtos escalares da equação 2.1 podem ser pré-calculados.

2.2.1 Otimização do algoritmo

Séquin e Smyrl (1989) propuseram algumas formas para reduzir-se o tempo para recálculo de uma imagem parametrizada e para redução da demanda de memória do algoritmo. Curiosamente, o algoritmo de ray tracing parametrizado geralmente obtém simultaneamente ganhos em armazenamento quando se otimiza a velocidade, bem ao contrário do ray tracing convencional, como mostramos em (SANTOS, 1994a).

2.2.1.1 Compartilhamento de expressões

Esta otimização se baseia no fato de que, devido à coerência de imagem, píxeis adjacentes têm maior chance de serem semelhantes. No caso do ray tracing parametrizado não basta que suas cores sejam semelhantes. É necessário que toda a árvore de raios associada aos píxeis sejam equivalentes nó a nó (mesmo material e produtos escalares próximos, dentro de limiar estabelecido). Esta otimização permite poupar tempo de processamento.

Esta otimização não foi implementada em nosso sistema pois, ao contrário do sistema de Séquin e Smyrl, nossa implementação dispõe do recurso de mapeamento de textura, indispensável para dar maior realismo à imagem. A presença de textura torna impossível garantir que dois píxeis serão idênticos já que mesmo que tenham rigorosamente a mesma árvore de raios, os texeis³ mapeados em cada ponto de intersecção serão diferentes, podendo tornar a cor dos mesmos radicalmente distintas. Outro detalhe importante a ser observado é que o sistema apresentado em (SÉQUIN & SMYRL, 1989) se utiliza apenas de fontes de luz direcionais aumentando enormemente a coerência de imagem já que os vetores de luz terão a mesma direção em todos os pontos de intersecção. Nosso sistema só dispõe de fontes de luz locais (as fontes direcionais podem ser simuladas com uma luz local muito distante) que adicionam realismo à cena e flexibilidade ao sistema, porém reduzem a coerência da imagem.

³ Texel = Texture Element (elemento de textura)

2.2.1.2 Codificação “run lenght”

Esta otimização baseia-se em codificar segundo a técnica “run-length” seqüências de píxeis que tenham árvores semelhantes de acordo com o critério da seção anterior. Assim, seqüências de píxeis são substituídas por uma única árvore e o número de ocorrências da mesma. Esta otimização permite economia de processamento e memória.

Não podemos tirar proveito desta técnica pela mesma razão indicada na seção anterior.

2.2.1.3 Atualização seletiva de expressões

Esta otimização consiste em armazenar-se junto à cada nó da árvore de raios de cada píxel o valor calculado para a sub-expressão nele representada. Quando ocorre uma mudança em algum parâmetro, somente os nós que aquele parâmetro influencia devem ser recalculados; nos demais, o valor previamente armazenado é propagado na árvore de raios. Note-se que todos os nós devem ser examinados. Esta otimização poupa tempo mais requer uso de mais memória.

Implementamos a reavaliação incremental que oferece ganhos semelhantes.

2.2.1.4 Codificação “run lenght” 2D

Semelhante à codificação “run length” normal, porém agora são comparados também os três píxeis vizinhos na linha de cima do píxel corrente. Assim, áreas (ao invés de linhas) podem ser substituídas por uma única árvore. Esta otimização procura aproveitar a coerência 2D presente na imagem.

Novamente, devido ao uso de textura, não podemos implementar esta otimização.

2.2.1.5 Compartilhamento de expressões com “hashing”

Esta otimização consiste em expandir-se a idéia do compartilhamento de sub-árvores não só numa linha (“run length”) ou numa área (“run length” 2D), mas na imagem toda. Isto é feito através do armazenamento de expressões numa tabela de “hashing” para que uma sub-árvore já codificada anteriormente possa ser rapidamente encontrada. Esta otimização reduz mais ainda a demanda de memória e processamento e se sobrepõe às técnicas “run length” 1D e 2D.

Decidimos pela não implementação desta otimização pelo fato de que, por usarmos mapeamento de textura, teríamos muito pouca coerência de imagem e seu uso poderia causar efeito contrário, isto é, sobrecarga de processamento e memória.

2.2.1.6 Atualização unitária de expressões

Consiste em associar-se um código a cada reavaliação da imagem. Este código é armazenado junto com as expressões compartilhadas na técnica 2.2.1.5. Antes de se reavaliar uma sub-expressão o código armazenado é comparado ao da reavaliação atual. Se forem iguais, indica que aquela expressão já foi atualizada e o valor antigo é usado, proporcionando ganho de processamento.

Como esta otimização depende da implementação da técnica da seção anterior, também não a implementamos.

O ganho apresentado com a melhor das otimizações propostas por Séquin e Smyrl varia de 2 a 5 vezes. Considerando as particularidades de ausência de textura e fontes de luz locais, acreditamos que este ganho seria extremamente reduzido, de forma que estas otimizações não nos servem, já que consideramos estes recursos de fundamental importância nas principais aplicações do algoritmo de ray tracing parametrizado.

2.3 Outras técnicas relacionadas

Conforme foi mencionado anteriormente, o sonho do RT² ("Real Time Ray Tracing" - Ray Tracing em Tempo Real) ainda não pode ser realizado com nenhuma máquina ou algoritmo que conhecemos hoje. As propostas que mais procuram se aproximar desta conquista se baseiam em situações com restrições como o ray tracing parametrizado.

Foram propostos alguns poucos novos algoritmos que procuram eliminar a restrição quanto à movimentação de objetos do ray tracing parametrizado. Discute-se nas seções seguintes, duas destas propostas.

2.3.1 Ray tracing incremental

Em 1990 foi proposto um novo algoritmo denominado ray tracing incremental (MURAKAMI & HIROTA, 1990), usado para se recalcular rapidamente uma imagem gerada por ray tracing, quando são feitas apenas pequenas modificações na geometria da cena ou propriedades das superfícies, mantendo-se o observador fixo como no caso do ray tracing parametrizado.

As estruturas de dados usadas passam a armazenar também pontos de intersecção, vetores normais às superfícies e dados relacionados ao caminho percorrido pelo raio usando um esquema de particionamento por voxéis⁴.

A idéia central deste algoritmo é determinar rapidamente quais raios foram influenciados por uma mudança na geometria através da estrutura de voxéis e recalcular apenas estes. São usadas tabelas "hash" e algoritmos de travessia otimizados para reduzir custos de acesso e processamento.

Este algoritmo incorpora mudanças na posição das fontes de luz e mapeamento de relevo ("bump-mapping"), itens que apresentamos neste trabalho no contexto do ray tracing parametrizado. No entanto, cabe notar que o tratamento de sombras no caso de movimentação de fontes de luz descrito por Murakami e Hirota é completamente convencional. Segundo os autores,

*"As estruturas de dados para o recálculo de sombras não incluem históricos dos voxéis percorridos ou das intersecções pelas seguintes razões. Primeiro, já que o número de raios de luz é grande (equivalente ao produto do número de pontos visíveis pelo número de fontes de luz), uma grande quantidade de dados para o recálculo de sombras seria necessária. Segundo, nós assumimos que o usuário muda com frequência a posição da fonte de luz. Quando a posição da fonte de luz muda, a direção de cada raio de luz deve ser recalculada e assim qualquer dado histórico teria que ser descartado."*⁵

Os autores também informam que pode ser necessário fazer-se ray tracing adicional *"quando o vetor normal num ponto visível muda devido ao mapeamento de relevo"*.

As extensões que apresentaremos nesta tese lidam de forma elegante com este problema, evitando sobrecarga em demasia.

Os resultados apresentados para este algoritmo mostram que mudanças em parâmetros óticos são recalculadas 100 vezes mais rapidamente que o ray tracing convencional. Mudanças na posição de objetos apresentam ganho de 10 vezes e as mudanças na posição de fontes de luz são apenas 4 vezes mais rápidas que o recálculo completo da imagem.

Considerando que a restrição de manter o observador fixo ainda persiste e que as mudanças na posição de objetos devem ser incrementais (i.e., muito pequenas), este

⁴ Voxel = Volume Element (elemento de volume).

⁵ T. do A.

algoritmo têm pouca aplicação adicional em relação ao ray tracing parametrizado. Acreditamos que a movimentação rápida das fontes de luz é de extrema importância, pois abre novos campos de aplicação para o ray tracing parametrizado, principalmente em luminotécnica. Desta forma, julgamos que a extensão que proporemos, por permitir mudança rápida na posição de fontes de luz, é mais vantajosa que o "ray tracing incremental".

2.3.2 Estruturas hierárquicas para manipulação interativa

As estruturas de dados propostas em (BRIÈRE & POULIN, 1996) permitem a manipulação de vários parâmetros ópticos e geométricos de uma cena e re-síntese da cena em poucos segundos. Esta técnica engloba o ray tracing parametrizado e se parece com o ray tracing incremental, mas adota outro enfoque para determinar rapidamente quais os raios que devem ser re-computados.

Os resultados apresentados pelos autores são bastante parecidos aos números de Murakami e Hirota. O algoritmo também apresenta as mesmas restrições de forma que o consideramos equivalente para todas as finalidades.

CAPÍTULO 3

O SISTEMA RT(P)

3 O Sistema RT(p)

RT(p) é um sistema constituído por um conjunto de programas que desenvolvemos no decorrer deste projeto com a finalidade de experimentar as idéias propostas nesta tese.

FIG. 3.1 - Logotipo do Projeto RT(p)

3.1 Histórico

Este projeto iniciou-se em 1993 quando este autor tomou conhecimento dos recursos do sistema denominado IPR (Interactive Photorealistic Rendering) oferecido nos pacotes TDI Explore™ e TDImage™. Esta é uma tecnologia fechada de propriedade da Thomson Digital Image, empresa atualmente incorporada pela Alias/Wavefront/Silicon Graphics. Este sistema é capaz de alterar cores, texturas e parâmetros de acabamento de superfície, assim como intensidade e cores das fontes de luz de uma cena sintetizada (porém não permite qualquer tipo de movimentação na cena). Tudo em poucos segundos e com a qualidade das imagens de ray tracing, após a geração de uma primeira imagem.

Intrigados e, após algum tempo meditando sobre o problema, recriamos, de forma independente, o algoritmo de ray tracing parametrizado. Naturalmente descobrimos em seguida que o mesmo já havia sido publicado em 1989.

De qualquer forma, acreditando no potencial de tal recurso somente disponível em um único produto no mercado e, tendo passado pela tediosa experiência representada pela fase de definição de materiais para a produção de animações, resolvemos iniciar o desenvolvimento de um *ray tracer* parametrizado (MARTINS & SANTOS, 1993).

Partindo de um *ray tracer* de domínio público muito simples, para facilitar a implementação - MTV Ray Tracer (VANDEWETTERING, 1988) - o modificamos para que passasse a gerar, além da imagem, também um arquivo contendo as informações da árvore de raios de cada píxel na imagem. Ao mesmo tempo,

desenvolvemos um sistema com interface gráfica que permitisse ao usuário o controle dos parâmetros ajustáveis e a reavaliação da imagem (MARTINS & SANTOS, 1994a). Estes programas iniciais executavam em estações Sun sob sistema operacional SunOs e GUI OpenLook™.

Neste mesmo sistema iniciamos o desenvolvimento das extensões relativas a movimentação de fontes de luz (SANTOS, 1994b), (MARTINS & SANTOS, 1994b) e mapeamento de relevo (SANTOS, 1994c).

Devido à necessidade de executar o sistema RT(p) em uma plataforma com recursos de aceleração gráfica em hardware, construímos uma nova interface gráfica no sistema Motif™ (fig. 3.2). Esta versão foi portada para estações Silicon Graphics e sistema operacional IRIX, fazendo uso do acelerador gráfico VGX.

FIG. 3.2 - Interface Motif™ do sistema RT(p)

Como o MTV Ray Tracer não dispunha originalmente de recursos de mapeamento de textura e de relevo, estes foram também acrescentados ao código original (CLUA, SANTOS & ZUFFO, 1995). Ao mesmo tempo, iniciou-se estudos para melhorar a estrutura de dados original visando reduzir a demanda de memória do algoritmo (ALMEIDA, SANTOS & ZUFFO, 1995).

Com a grande popularidade da Internet e visando divulgar o algoritmo de ray tracing parametrizado, em 1996 (ALMEIDA & SANTOS, 1996) disponibilizamos na Internet uma página HTML onde o usuário, preenchendo um formulário com os parâmetros de uma cena simples, poderia interagir remotamente com o sistema RT(p) (MOURÃO et al., 1996), (RAMOS, 1996), recebendo de volta uma imagem reavaliada com os novos parâmetros escolhidos (fig. 3.3).

FIG. 3.3 - Interface HTML do sistema RT(p)

Este mesmo sistema, denominado RTpWeb, que é baseado em um servidor Web, interface CGI e comunicação inter-processos serviu de suporte a um trabalho de arte interativa apresentado na Mostra de Arte do Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens - SIBGRAPI'96, em parceria com a artista Tania Fraga (FRAGA & SANTOS, 1996).

Mais recentemente, para aproveitar-se o enorme poder de processamento e baixo custo dos novos processadores Intel, fez-se novo porte do sistema, agora para processadores PentiumPro e sistema operacional Linux. Considerando os desenvolvimentos mais recentes, a experiência positiva da versão RTpWeb e o histórico de evolução do sistema RT(p), decidimos separar completamente a interface gráfica do sistema de reavaliação de imagens. Para garantir total portabilidade da interface gráfica implementamos uma versão em linguagem Java (ARNOLD & GOSLING, 1998), destinada a executar na forma de um applet dentro de um navegador WWW (fig. 3.4).

FIG. 3.4 - Interface Java do sistema RT(p)

Esta interface possibilitou a interação remota ou local com o sistema a partir de plataformas Windows95/NT, Linux, Sun, Silicon Graphics e Apple MacIntosh. Por outro lado, exigiu o desenvolvimento de um pequeno servidor HTTP devido às restrições de segurança dos applets Java. Além disso, foi criado um protocolo de comunicação (via sockets) entre o applet e o servidor RT(p). O applet envia os parâmetros ajustados pelo usuário ao servidor RT(p) e este, após reavaliar a imagem, a envia de volta ao applet para exibição no navegador. Esta interface permite ao usuário escolher qualquer imagem (formato GIF) disponível na Internet e utilizá-la no mapeamento de textura remoto.

O sucesso desta implementação Java fomentou o desenvolvimento de novas aplicações para o sistema RT(p), especialmente para fins educacionais (GOUVEIA & SANTOS, 1997).

O “ray tracer” que adotamos inicialmente como base do sistema RT(p) recebia como entrada cenas descritas no formato NFF (Neutral File Format) que permite apenas o uso de objetos simples como esferas, cilindros, cones, etc. ou objetos descritos por triângulos, para modelar-se as cenas. Este fato limitava seriamente a aplicação do sistema RT(p) já que não estavam disponíveis modeladores ou conversores para NFF. Decidiu-se então criar um “shader” especial que, funcionando em conjunto com o ray tracer *mental ray*, parte do sistema Softimage, poderia gerar o arquivo de dados para o servidor RT(p) a partir de cenários modelados com este poderoso sistema (Softimage) ou em qualquer formato por ele importado. Este sistema foi desenvolvido e hoje pode servir ao usuário que deseje realizar estudos de cores e texturas a serem sintetizadas pelo Softimage. Este módulo está detalhado no Anexo A.

Ao mesmo tempo, iniciou-se a paralelização do algoritmo no sistema SPADE-II (ALMEIDA & SANTOS, 1997) com o que se atingiu a meta de reavaliação de imagens a taxas interativas ($< 0.3s$). Uma nova versão paralela do sistema foi desenvolvida para que pudesse reavaliar duas imagens simultaneamente, correspondendo a um par estereoscópico. Este sistema foi demonstrado em evento de supercomputação (SC’97) em San Jose - CA, USA, em 1997 (detalhado no Anexo B).

Finalmente, em 1998, com a nova implementação das extensões propostas nesta tese, foi desenvolvida uma nova interface Java para configuração e movimentação das fontes de luz spot (fig. 3.5 e 3.6)

FIG. 3.5 - Nova interface Java do sistema RT(p) - painel de cores

FIG. 3.6 - Nova interface Java do sistema RT(p) - painel de luzes spot

3.2 Arquitetura do sistema

Em acordo com a tradição de pesquisa em processamento de alto desempenho do Laboratório de Sistemas Integráveis, a filosofia de implementação do sistema RT(p) foi concebida para que pudesse ser facilmente adaptada a um ambiente de processamento paralelo, particularmente de memória distribuída. Dessa forma, o RT(p), em sua forma final, foi concebido segundo uma arquitetura cliente-servidor cujos componentes se comunicam através de "sockets". A maior parte do código foi

escrito em linguagem 'C' sob ambiente Unix. A interface com usuário foi implementada como um applet em linguagem Java, podendo executar em qualquer plataforma que suporte este ambiente.

Esta forma de implementação trouxe várias vantagens ao sistema:

- a) A interface fica isolada do restante do sistema podendo ser executada remotamente através da Internet e propiciando total portabilidade;
- b) O sistema pode ser executado em paralelo em arquiteturas de memória distribuída ou rede de estações, sem modificações no código;
- c) Cada parte do sistema pode ser executada no ambiente mais favorável ao tipo de processamento necessário.

Todos os componentes do sistema comunicam-se entre si utilizando protocolos proprietários bastante simples visando a redução da carga de comunicação.

Quando estão disponíveis vários processadores para a execução do sistema, pode-se fazer o processamento em paralelo segundo o esquema "fazenda de processadores" (MAY & SHEPHERD, 1987) no qual um processo mestre envia tarefas a um conjunto de processos escravos executando em paralelo.

3.2.1 Componentes

O sistema RT(p) é composto pelos seguintes componentes, detalhados nas seções seguintes: applet rtpweb, master, slave, sserver (shadow server), shader "gera_rtp" e http server.

3.2.1.1 Applet rtpweb

Este applet implementa a interface gráfica com o usuário do sistema. Através dele podem ser configurados os parâmetros da cena e visualizadas as imagens produzidas.

Os parâmetros configuráveis são mostrados na tabela 3.1:

Contexto	Parâmetro	Descrição
Materiais	cor _R	componente vermelha da cor para reflexão difusa.
	cor _G	componente verde da cor para reflexão difusa.
	cor _B	componente azul da cor para reflexão difusa.
	spec _R	componente vermelha da cor para reflexão especular local.
	spec _G	componente verde da cor para reflexão especular local.
	spec _B	componente azul da cor para reflexão especular local.

	amb _R	componente vermelha da cor para reflexão ambiente.
	amb _G	componente verde da cor para reflexão ambiente.
	amb _B	componente azul da cor para reflexão ambiente.
	tex	número da textura interna (0 se externa).
	tex _{URL}	URL da textura externa.
	xtile	replicação horizontal do padrão de textura.
	ytile	replicação vertical do padrão de textura.
	mix	porcentagem de combinação da cor difusa com textura.
	k _s	coeficiente de reflexão especular global.
	k _t	coeficiente de transmissão (transparência).
	n	coeficiente de polimento (expoente de Phong).
Luzes	l _R	componente vermelha da cor da luz.
	l _G	componente verde da cor da luz.
	l _B	componente azul da cor da luz.
	i	Intensidade da luz.
	x	coordenada x da posição da luz.
	y	coordenada y da posição da luz.
	z	coordenada z da posição da luz.
	lx	componente x da direção da luz spot.
	ly	componente y da direção da luz spot.
	lz	componente z da direção da luz spot.
	maxang	ângulo do cone de luz spot.
	minang	ângulo do cone de máxima intensidade de luz spot.
Luz Amb.	ia _R	componente vermelha da cor da luz ambiente global.
	ia _G	componente verde da cor da luz ambiente global.
	ia _B	componente azul da cor da luz ambiente global.
Fundo	back _R	componente vermelha da cor de fundo.
	back _G	componente verde da cor de fundo.
	back _B	componente azul da cor de fundo.
	back_tex	número da textura interna (0 se externa) do fundo.
	back_tex _{URL}	URL da textura externa do fundo.
	back_xtile	replicação horizontal do padrão de textura do fundo.
	back_ytile	replicação vertical do padrão de textura do fundo.
	back_mix	porc. de combinação da cor difusa com textura do fundo.

TAB. 3.1 - Parâmetros modificáveis no sistema RT(p)

Esta interface procura seguir parâmetros de ergonomia para que a interação com o usuário seja simples já que a grande quantidade de parâmetros configuráveis (aprox. 40) poderia torná-la complexa em demasia.

A tabela 3.1 menciona texturas "internas" e "externas". As primeiras são texturas cujos arquivos são carregados imediatamente ao se iniciar a operação do sistema, ficando sempre disponíveis ao usuário. As texturas "externas" são aquelas que podem ser carregadas a partir da Internet, durante a execução do sistema.

As informações relativas à cena (número de materiais e fontes de luz) e a configuração inicial dos parâmetros são solicitadas pelo applet ao *master* que, em resposta, envia a configuração corrente.

Em relação à visualização das imagens produzidas, o usuário pode escolher (através de parâmetro na página HTML) entre uma resolução fixa, com a imagem mostrada em área gráfica ao lado dos controles ou em uma janela externa cujo tamanho pode ser alterado. A imagem é sempre calculada em uma resolução fixa (geralmente alta) e depois é escalada de acordo com a resolução da janela externa. O processamento da imagem em resolução superior à de exibição serve para evitar-se serrilhamento (*aliasing*).

As imagens são calculadas com 24 bits de cor por píxel, mas são enviadas (juntamente com o *palette* calculado) após quantização de cores (8 bits) para redução do tempo de transmissão dos dados. Também foi desenvolvida uma versão do componente "master" que, ao invés de quantizar e enviar a imagem ao applet, simplesmente mostra-a em uma janela na máquina em que estiver sendo executado. Esta versão é denominada "master_local" e foi criada para melhorar a interação do sistema quando operado localmente.

Este componente está implementado em aproximadamente 2100 linhas de código Java.

3.2.1.2 Master

Este programa é responsável pela comunicação com o applet rtpweb, recebendo dele informações sobre os parâmetros a modificar e enviando-lhe a imagem reavaliada. O *master* recebe este nome pois atua como o processo mestre quando é feita execução em paralelo. Para isso deve comunicar-se com os processos escravos para receber todas as partes da imagem recalculadas, compondo uma imagem única a ser enviada ao applet.

O *master* é responsável por receber cada linha da imagem dos *slaves* e fazer o escalamento vertical compatível com a dimensão vertical atual da janela de imagem do rtpweb. Além disso, também faz a quantização de cores (de 24 para 8 bits) com o cálculo de *palette*, antes de enviar a imagem ao rtpweb.

O *master* também comunica-se com o servidor de sombras (*sserver*) quando necessário devido a alteração na posição de fontes de luz.

Para maior compatibilidade com o pacote Softimage, a equação de iluminação do RT(p) contém termos adicionais àqueles apresentados na equação 2.1, tomando a seguinte forma (o sinal $\sim$ indica que o parâmetro tem componentes R,G e B):

$$\begin{aligned} \tilde{I} = & \tilde{k}_a I_a + \\ & \left[(1 - mix) \cdot \tilde{c}_{obj} + mix \cdot \tilde{c}_{tex} \right] \sum_{j \in \Omega} \left[(\bar{N} \cdot \bar{L}_j) I_j \cdot \tilde{c}_{luz_j} \right] + \\ & \tilde{s} \cdot \sum_{j \in \Omega} \left[(\bar{N} \cdot \bar{H}_j)^n \cdot I_j \cdot \tilde{c}_{luz_j} \right] + k_s I_r + k_t I_t \end{aligned} \quad (3.1)$$

onde:	$\tilde{I}$	intensidade do ponto amostrado no plano de imagem;
	$\tilde{k}_a$	cor de reflexão de luz ambiente;
	I_a	intensidade da iluminação ambiente global ;
	mix	porcentagem de cor de textura na cor do material;
	c_{obj}	cor difusa do material;
	c_{tex}	cor da textura do material;
	$\bar{N}$	vetor unitário normal à superfície do objeto;
	$\bar{L}_j$	vetor unitário na direção da j-ésima fonte de luz;
	I_j	intensidade da j-ésima fonte de luz;
	c_{luz_j}	cor da j-ésima fonte de luz;
	Ω	subconjunto das fontes de luz que iluminam o ponto;
	$\tilde{s}$	cor especular do material;
	$\bar{H}_j$	vetor unitário na direção bissetriz entre o observador e $\bar{L}_j$;
	n	expoente que representa o polimento da superfície;
	k_s	coeficiente de reflexão de luz especular global do material;
	I_r	intensidade do raio refletido;
	k_t	coeficiente de transmissão do material;
	I_t	intensidade do raio transmitido (refratado);

Este componente está implementado em aproximadamente 2200 linhas de código em linguagem "C".

3.2.1.3 Slave

Este programa tem a função de receber parâmetros do programa *master* e do arquivo de dados ".rtp" e realizar os cálculos para reavaliação de parte da imagem, enviando ao master as linhas da imagem que recalculou.

Cada linha da imagem é processada na resolução horizontal em que foi originalmente gerada quando da criação do arquivo ".rtp". As linhas repassadas ao *master*, no entanto, têm a resolução horizontal da janela do applet rtpweb. Portanto, o escalamento horizontal da imagem (através de média) é feito pelo programa *slave*.

Este componente está implementado em aproximadamente 2900 linhas de código em linguagem "C".

3.2.1.4 Sserver (Shadow Server)

Este programa implementa o "servidor de sombras". Comunica-se com o master para receber a descrição geométrica da cena e o conjunto de pontos de intersecção determinados no pré-processamento da imagem por ray tracing. Após receber novas coordenadas do posicionamento de uma fonte de luz, recalcula a condição (na sombra / na luz) de cada ponto de intersecção, retornando esta informação ao *master* que a repassa adequadamente aos *slaves*. Este componente está implementado em aproximadamente 670 linhas de código em linguagem "C" e biblioteca OpenGL.

Existem também outros dois componentes auxiliares no sistema RT(p):

3.2.1.5 Shader "gera_rtp"

Este programa é um "shader" escrito para o programa *mental ray*, o ray tracer que acompanha o pacote SoftImage. Este "shader" tem a função de gerar o arquivo de dados (".rtp") que contém a descrição completa da árvore de raios da cena. A grande vantagem deste sistema é que pode-se usar qualquer cena modelada no sofisticado pacote SoftImage sem a necessidade de implementar um ray tracer capaz de processar entidades geométricas sofisticadas (superfícies NURBS, etc.). Estamos agora implementando uma interface para configuração deste shader dentro do pacote Softimage, aumentando sua integração. Este componente está implementado em aproximadamente 1650 linhas de código em linguagem "C".

3.2.1.6 Http Server

Este é um micro-servidor web que deve executar na mesma máquina onde se encontra o programa *master*.

A implementação deste programa foi necessária devido às restrições de segurança impostas aos applets Java⁶ pelo navegador em que são executadas. Desejamos evitar a instalação de um servidor web convencional devido à grande sobrecarga desnecessária que traria ao sistema.

Apesar de bastante simples, o servidor implementado conta com recursos de proteção (só serve arquivos em diretórios pré-configurados em arquivo próprio), reconhece *proxies* e faz cache de dados, aderindo ao protocolo HTTP 1.0.

⁶ Ao applet somente é permitida a conexão via *sockets* com a máquina da qual o próprio applet foi recebido.

Este componente está implementado em aproximadamente 1000 linhas de código em linguagem "C".

A figura 3.7, mostra a relação entre os vários componentes do sistema RT(p).

FIG. 3.7 - Relacionamento entre os componentes do sistema RT(p)

3.3 Exemplos

A figura 3.8 mostra uma seqüência de imagens gerada no RT(p) onde os vários parâmetros ópticos da cena vão sendo alterados sucessivamente. A primeira imagem (bule transparente) demora o tempo de geração convencional do ray tracing. As demais são geradas em menos de 1 segundo pelo sistema. A imagem da fig. 3.9 ilustra a aplicação de textura.

3.4 Conclusão

O sistema RT(p) é um projeto em andamento. Como será visto, esta tese ilustra seu estágio de desenvolvimento atual e indica algumas direções para encaminhamento futuro.

O aspecto mais importante que se deseja destacar é seu caráter evolutivo e o fato de que este trabalho não encerra uma linha de desenvolvimento; apenas apresenta um instantâneo do estágio presente.

O objetivo primordial do projeto é a aplicação da técnica de ray tracing parametrizado para servir à sociedade através da Engenharia. Sua próxima fase deverá contar com a participação de usuários potenciais da técnica, em aplicações reais e úteis, as quais guiarão as prioridades de desenvolvimentos. Acreditamos que terá frutos recompensadores.

FIG. 3.8 - Sequência de imagens geradas com o RT(p)

FIG. 3.9 - Imagens demonstrando aplicação de texturas no RT(p)

CAPÍTULO 4

MOVIMENTAÇÃO DE FONTES DE LUZ

4 Movimentação de fontes de luz

A posição das fontes de luz no cenário é considerado um parâmetro geométrico da cena e, portanto, não pode ser alterado de acordo com o algoritmo original de ray tracing parametrizado (SÉQUIN & SMYRL, 1989).

Neste capítulo introduzimos uma extensão ao algoritmo básico de ray tracing parametrizado, permitindo que as fontes de luz da cena sejam movimentadas, mantendo-se baixo o tempo de reavaliação.

A possibilidade de alterar-se a posição das fontes de luz de uma cena e recalcular-se rapidamente a imagem abre novas possibilidades de aplicação ao algoritmo de ray tracing parametrizado. As aplicações potenciais desta extensão encontram-se particularmente nas áreas de projeto de iluminação (luminotécnica) para Arquitetura (interiores e exteriores) e Cenografia, no projeto industrial de luminárias e na área de animação.

4.1 Definição do problema

Deseja-se alterar a posição das fontes de luz que iluminam um cenário sintético e obter-se uma nova imagem que reflita com razoável precisão tal alteração, num intervalo de tempo pequeno comparado com o requerido para o recálculo da imagem pelo algoritmo tradicional de ray tracing.

Ao modificar-se a posição das fontes de luz, os seguintes termos da equação de iluminação (eq. 2.1) são afetados:

$\vec{L}_j$: vetor na direção da j -ésima fonte de luz, função direta da posição desta;

Ω : subconjunto das fontes de luz que iluminam um dado ponto de intersecção;

$\vec{H}_j$: vetor unitário na direção bissetriz entre o observador e $\vec{L}_j$;

Conseqüentemente, os termos $\vec{N} \cdot \vec{L}_j$ e $\vec{N} \cdot \vec{H}_j$ também sofrem alteração.

Todos estes termos devem ser recalculados de forma rápida para que o algoritmo seja eficiente.

O armazenamento do ponto de intersecção em cada nó da árvore de raios permite um recálculo simples dos vetores $\vec{L}_j$ e $\vec{H}_j$. Os produtos escalares $\vec{N} \cdot \vec{L}_j$ e $\vec{N} \cdot \vec{H}_j$, antes armazenados como constantes, agora devem ser também recalculados a cada reavaliação da imagem. Propusemos conjuntamente em (LEITÃO et al., 1996) uma técnica de aceleração para o cálculo destes produtos escalares, resumida nesta

tese (seção 4.2), capaz de reduzir o impacto deste recálculo no tempo total de reavaliação da imagem em certos casos particulares.

Resta ainda determinar, de forma eficiente, o novo subconjunto Ω das fontes de luz que iluminam cada ponto de intersecção, já que a movimentação de fontes de luz pode fazer com que objetos que estavam à sombra sejam agora iluminados e vice-versa.

Em (SANTOS, 1994b) e (SANTOS, 1994c) propusemos o uso de um algoritmo híbrido, combinando as técnicas denominadas “item-buffer” (WEGHORST et al., 1984) e “light-buffer” (HAINES & GREENBERG, 1986) como forma de recalculá-lo rapidamente o subconjunto Ω . Este algoritmo será revisado na seção 4.3.2. Na seção 4.3.3 apresentamos um novo algoritmo que elimina algumas das deficiências da proposta anterior.

4.2 Cálculo rápido de produtos escalares para equação de iluminação

Em (LEITÃO, AYRES, FAGUNDES, SANTOS & ZUFFO, 1996) propusemos um algoritmo para o cálculo rápido da equação de iluminação onde, através da representação dos vetores da equação de iluminação por coordenadas esféricas, pode-se diminuir o processamento necessário ao cálculo dos produtos escalares envolvidos, reduzindo-o a algumas adições e acessos a tabelas.

Após alguma manipulação trigonométrica, pode-se mostrar que:

$$\vec{N} \cdot \vec{L} = \frac{1}{2} \cos(\Delta \varphi)(1 + \cos(\Delta \theta)) + \frac{1}{2} \cos(\Theta \varphi)(1 - \cos(\Delta \theta)) \quad (4.1)$$

onde:

$$\Delta \theta = \theta_L - \theta_N \quad (4.2)$$

$$\Theta \theta = \theta_L + \theta_N \quad (4.3)$$

$$\Delta \varphi = \varphi_L - \varphi_N \quad (4.4)$$

O significado dos ângulos $\theta_L, \theta_N, \varphi_L$ e φ_N é mostrado na figura 4.1.

Esta forma de representar o produto escalar dos vetores $\vec{N}$ e $\vec{L}$ permite que os termos da equação 4.1 possam ser representados em duas tabelas da seguinte forma:

$$F(\Delta \varphi, \Delta \theta) = 1/2 \cos(\Delta \varphi)(1 + \cos(\Delta \theta)) \quad (4.5)$$

$$G(\Theta \varphi, \Delta \theta) = 1/2 \cos(\Theta \varphi)(1 - \cos(\Delta \theta)) \quad (4.6)$$

FIG. 4.1 - Vetores da equação de iluminação em coordenadas esféricas

O produto escalar pode então ser calculado através de uma única adição e duas consultas a tabela (eq. 4.7):

$$\vec{N} \cdot \vec{L} = F(\Delta\varphi, \Delta\theta) + G(\Theta\varphi, \Delta\theta) \quad (4.7)$$

O produto escalar referente a parcela de luz especular $(\vec{N} \cdot \vec{H})^n$ pode ser calculado através do uso das mesmas tabelas (LEITÃO, AYRES, SANTOS & ZUFFO, -). A exponenciação pode ser calculada através de consulta a outra tabela pré-computada $H(x)$. O termo especular fica:

$$(\vec{H} \cdot \vec{N})^n = H(F(\Delta\varphi, \Delta\theta) + G(\Theta\varphi, \Delta\theta), n) \quad (4.8)$$

Devido à simetria das duas primeiras tabelas, utilizando-se 16 bits de precisão, pode-se armazenar todas as três tabelas em 1 Mbyte de memória.

É importante notar que o uso deste algoritmo só é eficaz quando se usam fontes de luz direcionais (vetor $\vec{L}$ constante) pois neste caso pode-se construir as tabelas supondo que o sistema de coordenadas aponta para a direção da luz ($\vec{L}$) dispensando-se o cálculo da diferença das equações 4.2 a 4.4.

Como este é um caso bastante restrito, optamos por não implementá-lo no sistema RT(p). Os resultados apresentados na seção 4.6 mostram que a sobrecarga do recálculo dos produtos escalares (cada produto escalar: 3 produtos e 2 somas) é perfeitamente tolerável nas aplicações alvo.

4.3 Determinação das fontes de luz incidentes no ponto de intersecção

No algoritmo tradicional de ray tracing, a determinação das fontes de luz que incidem sobre um determinado ponto é feita, conforme apresentado na secção 2.1, através do lançamento de raios de sombra. Este procedimento não poderá ser adotado para uso no algoritmo de reavaliação do ray tracing parametrizado visto que é muito custoso, devendo-se evitar cálculos de intersecção durante o procedimento de reavaliação da imagem.

Vários trabalhos propõem técnicas para acelerar-se o cálculo dos raios de sombra. A seguir descreve-se a técnica denominada “light buffer” que objetiva reduzir o número de objetos a serem testados para intersecção com os raios de sombra, acelerando o processamento deste tipo de raio.

4.3.1 O algoritmo “light buffer”⁷

Segundo esta técnica, pode-se acelerar o processamento dos raios de sombra envolvendo-se cada fonte de luz por um cubo direcional (ARVO & KIRK, 89), conforme mostra a figura 4.2.

FIG. 4.2 - O cubo direcional

O cubo direcional é uma estrutura de dados que pode ser representada por um cubo centrado na origem de um sistema de coordenadas (fig. 4.2). Este cubo tem dimensão lateral igual a 2 e suas faces são alinhadas com os eixos principais, de forma que seus lados apresentam coordenadas variando de -1 a +1. Esta convenção simplifica os cálculos como será visto adiante.

No algoritmo *light-buffer* (HAINES & GREENBERG, 1986), o cubo é centrado com a origem do sistema coincidindo com a fonte de luz. Suas faces são divididas em células segundo uma resolução especificada. A cada célula do cubo direcional associa-se uma lista de objetos (polígonos, normalmente). Estas listas indicam quais

⁷ Adaptado de (SANTOS, 94b) e (SANTOS, 94c)

objetos projetam-se sobre cada célula, tendo como centro de projeção a origem do sistema. Estas listas são ordenadas de forma crescente pela distância do polígono à fonte de luz. Polígonos orientados em direção oposta à fonte de luz não são incluídos nestas listas (rejeição trivial).

O preenchimento dos *light-buffers* é feito numa etapa de pré-processamento. A cada fonte de luz é associado um cubo direcional. Todos os objetos devem ser projetados em cada uma das seis faces de cada cubo direcional.

O procedimento de teste para determinar se um ponto da cena encontra-se à sombra em relação a uma dada fonte de luz é relativamente simples, como descrito a seguir.

Determina-se qual face do cubo direcional é interceptada pelo vetor associado à fonte de luz e ao ponto de intersecção, tomando-se a direção dominante deste vetor, isto é, a que apresentar a componente (x, y ou z) de maior módulo. Obtém-se as coordenadas do ponto de intersecção na face simplesmente dividindo-se as outras duas coordenadas pelo valor da coordenada da direção dominante.

Com a coordenada do ponto de intersecção na face do cubo direcional pode-se facilmente determinar a qual célula pertence este ponto. A lista de polígonos associada àquela célula é testada, em ordem, para intersecção com o raio de sombra. O polígono que contém a origem do raio de sombra naturalmente não deve ser considerado já que não pode fazer sombra a si mesmo. A procura na lista de polígonos continua até que seja encontrado um polígono que intercepta o raio de sombra (ponto à sombra) ou a distância testada seja maior que aquela do ponto de intersecção (ponto iluminado) ou ainda, termine a lista de polígonos candidatos (ponto iluminado).

Detalhes de implementação e otimização desta técnica podem ser encontrados em (HAINES & GREENBERG, 86).

A aceleração no cálculo de sombras proporcionada por este algoritmo varia em função das características do cenário, número de fontes de luz e resolução adotada para os cubos direcionais. Testes apresentados por seus autores mostram uma aceleração média de 7.6 vezes na fase de cálculo de sombra realizados num conjunto de 7 imagens, com máximo de 12.8 vezes e mínimo de 2.7, já considerando o tempo de pré-processamento para preenchimento dos *light-buffers*.

A desvantagem do uso desta técnica está na maior demanda de memória (para armazenamento das listas de objetos de cada célula) e no fato de que os cálculos de intersecção são reduzidos mas não eliminados.

4.3.2 Um algoritmo auxiliado por acelerador gráfico

Propusemos em (SANTOS, 94b) e (SANTOS, 94c) um algoritmo para determinação rápida de sombras, combinando as técnicas *light-buffer* (descrita na seção anterior) e *item-buffer* (WEGHORST et al., 1984) (descrita a seguir).

A técnica denominada *item buffer* foi proposta para a aceleração do cálculo de raios primários. Este algoritmo opera de forma muito semelhante ao *Z-buffer* (CATMULL, 74), com a exceção de que, ao invés de ser armazenada uma profundidade (coord. Z) em cada célula, armazena-se um ponteiro ou índice correspondente ao objeto visível em cada píxel. Assim, o cálculo de intersecção de cada raio primário limita-se ao objeto indicado na respectiva posição do *item-buffer*.

O algoritmo que propusemos consta basicamente do cálculo de um *item-buffer* colocado na posição da fonte de luz ao invés de junto ao observador, tal como é feito para o *light-buffer*. A rigor, é necessário um *item-buffer* para cada face do cubo direcional do *light-buffer*. Chamaremos *pixel-light-buffer* (PLB) à esta nova estrutura de dados, já que a “célula” do *light-buffer* original agora se reduz a um único píxel.

O teste de sombreamento é extremamente simples: se a identificação do objeto da célula do PLB de uma certa fonte de luz corresponder àquela do objeto que contém o ponto sendo testado, este estará sendo iluminado por aquela fonte de luz. Caso contrário, isto é, se a identificação encontrada no píxel correspondente do PLB for diferente da identificação do objeto testado, este estará à sombra em relação àquela fonte de luz.

A vantagem desta técnica sobre o *light-buffer* convencional é que pode ser acelerada pelo uso de hardware gráfico. Este tipo de dispositivo é bastante comum já que o algoritmo Z-buffer é comumente implementado em aceleradores gráficos para determinação rápida de visibilidade. Se a cada objeto for atribuída uma cor individual e for empregada tonalização constante (“flat-shading”), não é necessária nenhuma alteração nas rotinas normalmente usadas com tais dispositivos, já que a geração normal da imagem produz um PLB onde a cor de cada píxel é o código de identificação do objeto de cada célula. Com o grande avanço destes aceleradores, o cálculo dos PLB já pode ser realizado em poucos centésimos de segundo.

Por outro lado, a desvantagem desta técnica é que este algoritmo opera no espaço de imagem e, conseqüentemente, está sujeito a severos efeitos de serrilhamento (“aliasing”). Estes efeitos se manifestam na forma de serrilhamento nas bordas das sombras projetadas e em falso auto-sombreamento (polígonos adjacentes provocam sombras uns nos outros).

O serrilhamento nas sombras pode ser atenuado com o aumento da resolução dos *pixel-light-buffers*. De qualquer forma, como trata-se de processo iterativo, este fenômeno terá importância bastante reduzida não comprometendo de forma alguma sua utilidade para os fins a que se propõe o ray tracing parametrizado. Após o correto posicionamento das fontes de luz e escolha de cores e texturas, uma imagem em resolução adequada pode ser gerada com o uso do algoritmo de ray tracing convencional.

Os efeitos de falso auto-sombreamento ocorrem também devido a sub-amostragem da cena, já que o ray tracing é executado no espaço de objetos e o Z-buffer resulta no espaço imagem. Sua incidência também é reduzida com o aumento da resolução dos PLB, porém seu efeito, mesmo reduzido, é mais incômodo do que o serrilhamento das sombras. Estes efeitos são visualmente piores pois surgem dentro das superfícies, denunciando a discretização geométrica do modelamento. Uma alternativa para a atenuação destes efeitos seria a numeração dos polígonos da cena de forma a que polígonos adjacentes ou próximos recebessem identificação próxima. Ao fazer o teste de visibilidade, em caso de sombra, deve-se verificar se a identificação no píxel correspondente do PLB está dentro de uma certa tolerância, caso em que provavelmente trata-se de "aliasing" e deve-se acusar o ponto como iluminado. A dificuldade de implementação deste enfoque reside no algoritmo de numeração dos polígonos. Preferimos resolver este problema com a proposta apresentada na seção 4.3.3.

4.3.2.1 Demanda de memória

O uso da técnica PLB com *ray tracing* parametrizado pode impor ou não um outro custo adicional de memória para armazenamento de parâmetros, dependendo do número de fontes de luz presentes no cenário. Sendo ω o número de fontes de luz que iluminam um ponto (tamanho do conjunto Ω), deverão ser armazenadas $2 \cdot \omega$ constantes reais (“floats”), resultantes da avaliação dos termos $(\bar{N} \cdot \bar{L}_j)$ e $(\bar{N} \cdot \bar{H}_j)$, para cada nó da árvore de raios, resultando em:

$$2 \cdot \omega \text{ floats/nó}$$

A adoção da modificação proposta requer o armazenamento explícito do ponto de intersecção e dos vetores normal e de reflexão em cada nó, resultando em

$$3 \times 3 \text{ floats} = 9 \text{ floats/nó}$$

Assim, com o uso de 5 ou mais fontes de luz, o armazenamento requerido já pode ser menor se adotada a nova técnica. Caso contrário, a nova técnica demandará maior capacidade de armazenamento.

Vale notar que a memória exigida para processamento também pode ser maior devido ao armazenamento dos dados dos *pixel-light-buffers*. No entanto, se usados aceleradores gráficos para esta parte do processamento, pode-se aproveitar a memória disponível nestes dispositivos para armazenar os PLBs na forma de imagens. Por outro lado, com a rapidez atual destes aceleradores, pode-se até mesmo descartar os PLBs após cada uso, recalculando-os sempre que necessário.

4.3.2.2 Sobrecarga de processamento

O uso da técnica PLB em *ray tracing* parametrizado causa maior sobrecarga de processamento do que o algoritmo tradicional já que além do próprio processamento envolvido com seu uso, os produtos escalares da equação de iluminação terão de ser recalculados a cada reavaliação.

De qualquer forma, esta sobrecarga é um preço aceitável a ser pago pela possibilidade de movimentação das fontes de luz pois o recálculo de uma imagem completa por *ray tracing* é muito mais dispendioso em termos de tempo já que envolveria também todos os cálculos de intersecção associados com os raios primários e secundários.

4.3.3 Um algoritmo melhorado

Propomos aqui um algoritmo melhorado em relação ao apresentado na seção 4.3.2. Este algoritmo, baseado na técnica de determinação de sombras proposta por Williams (1978), substitui os PLB por simples Z-buffers (chamados "Shadow Buffers", neste caso).

O algoritmo é tão simples quanto o anterior. Neste caso, a profundidade armazenada para o píxel correspondente no Z-buffer é comparada com a profundidade (distância em relação à fonte de luz) do ponto de intersecção. Caso seja menor, indica que existe algum objeto entre o ponto de intersecção e a fonte de luz, provocando sombra. Caso contrário o ponto estará iluminado.

A vantagem deste algoritmo é que o auto-sombreamento pode ser eliminado atribuindo-se um limiar mínimo para a diferença de profundidades detectada entre o ponto de intersecção e a armazenada no Z-buffer. O teste de sombra neste caso toma a forma representada na equação 4.9:

$$\text{se } (Z_{hit} - Eps) < Z_{buffer} \begin{cases} \text{então : ponto na LUZ} \\ \text{senão : ponto na SOMBRA} \end{cases} \quad (4.9)$$

onde: Z_{hit} : distância do ponto testado à fonte de luz;

Eps : limiar de teste (ajustável);

Z_{buffer} : distância correspondente armazenada no Z-buffer;

Outra vantagem é que são dispensados quaisquer processamentos referentes ao sombreamento, acelerando a geração dos buffers. Por outro lado exigem o cálculo da profundidade de cada ponto de intersecção da cena em relação à fonte de luz. Esta não é uma grande desvantagem já que as coordenadas da projeção (x, y) dos pontos na área do Z-buffer devem ser calculada nos dois casos. A terceira coordenada (z) é obtida naturalmente pela mesma transformação aplicada às outras duas e pode ser acelerada por *hardware* gráfico.

4.4 Implementação

Implementamos o algoritmo da seção 4.3.3 na forma de um "servidor de sombras": um programa autônomo que se comunica via sockets com o sistema RT(p). Este programa foi implementado em linguagem 'C' em ambiente UNIX e usa a biblioteca gráfica OpenGL (NEIDER, DAVIS, WOO, 1993). Esta implementação está de acordo com a filosofia geral "distribuída" do sistema RT(p) e permite que este subsistema seja executado na arquitetura disponível que prover melhor desempenho, como, por exemplo, sistemas Silicon Graphics com aceleração do cálculo de Z-buffer por *hardware*.

O servidor de sombras reconhece diversos comandos pré-estabelecidos num protocolo de comunicação. Na fase de iniciação o servidor recebe a descrição da geometria da cena e um conjunto de coordenadas 3D referentes a todos os pontos de intersecção determinados no pré-processamento do ray tracing parametrizado.

Na fase de operação, o servidor de sombras recebe as coordenadas 3D de uma fonte de luz. São computados os Shadow Buffers necessários de acordo com a posição relativa da fonte de luz na cena. Cada ponto de intersecção é então projetado e sua profundidade comparada com a correspondente no Shadow Buffer.

Ao terminar os cálculos para todos os pontos, o servidor de sombras envia uma cadeia de bits ao programa *master* do sistema RT(p). Esta cadeia de bits contém um bit para cada ponto de intersecção, na ordem em que foram enviados ao servidor de sombras. O valor 1 denota ponto iluminado e 0, sombreado.

Demais modificações na posição da luz são solicitadas através do envio de apenas um nova coordenada 3D.

O limiar de comparação de profundidade e a resolução dos "shadow buffers" podem ser alterados dinamicamente através do envio de comandos ao servidor.

4.5 Metodologia de avaliação

Foram feitos ensaios com duas cenas de características distintas (tabela 4.1), em várias resoluções (tabelas 4.2 e 4.3).

A cena "Bule" (fig. 4.3) , ilustrando o famoso bule da equipe de computação gráfica da Universidade de Utah (CROW, 1987), tem pequena complexidade, com pouco mais de 1000 polígonos. Possui apenas uma fonte de luz e 3 materiais. Este objeto foi gerado com o pacote SPD - *Standard Procedural Database* (HAINES, 1987) no formato NFF e convertido para o formato MI do mental ray através de conversor que desenvolvemos. A cena é constituída apenas por polígonos em cujos vértices são definidos vetores normais.

FIG. 4.3 - Cena "Bule"

A cena "Interior", figura 4.4, ilustra uma sala de estar. É complexa e foi importada para o Softimage com o qual foi gerado o arquivo MI. Esta cena apresenta 103 objetos, 4 fontes de luz e 11 materiais somando ao todo mais de 32 mil polígonos.

Parâmetro	Bule	Interior
N. Polígonos	1.008	32.939
Num. Vértices	3.008	106.015
Área com cor de fundo	73.6%	0.0%

TAB. 4.1 - Características das cenas de teste

FIG. 4.4 - Cena "Interior"

Para avaliação do desempenho dos algoritmos implementados, estas cenas foram geradas em várias resoluções, resultando arquivos com as características mostradas nas tabelas 4.2 e 4.3 para a cena "Bule" e "Interior", respectivamente.

A cena "Interior" foi gerada em apenas uma resolução mas com controle da profundidade máxima da árvore de raios (*trace depth*) controlada, variando de 1 até 4.

Parâmetro	Bule100	Bule300	Bule500	Bule1000
Resolução (píxeis)	100x100	300x300	500x500	1000x1000
Núm. de píxeis	10.000	90.000	250.000	1.000.000
Num. hit points ⁸	12.339	61.052	309.294	1.237.650
Tam. retorno do sserver (bytes)	1.543	7.632	38.662	154.707
Tamanho arquivo RTP (bytes)	869.452	3.564.275	15.894.729	62.858.109

TAB. 4.2 - Características da cena "Bule" em várias resoluções

Parâmetro	InteriorT1	InteriorT2	InteriorT3	InteriorT4
Resolução (píxeis)	300x300	300x300	300x300	300x300
Núm. de píxeis	90.000	90.000	90.000	90.000
Num. hit points	268.599	585.636	1.132.847	2.040.931
Tam. retorno do sserver (bytes)	33.575	73.205	141.606	255.117
Tamanho arquivo RTP (bytes)	20.407.564	37.061.604	65.877.808	113.870.612

TAB. 4.3 - Características da cena "Interior"

⁸ hit point: ponto de intersecção de um raio com objeto (um nó da árvore de raios)

4.6 Resultados e análises

O Anexo C contém todos os dados dos quais derivaram os gráficos e resultados apresentados nesta seção.

As condições do ensaio estão descritas na tabela 4.4.

Parâmetro	Valor
Processador	Dual Intel Pentium Pro (P6), 200MHz
Memória RAM	64 Mbytes
Sistema Operacional	Red Hat Linux release 4.2 (Biltmore) Kernel 2.0.30
Rede	FastEthernet 100Mbps

TAB. 4.4 - Condições para ensaio do componente *sserver* (shadow server)

A figura 4.5 mostra um cubo direcional rebatido no plano, gerado pelo algoritmo proposto. As faces que ficam totalmente atrás da geometria (portanto que não "enxergam" nenhum elemento da cena) não são computadas pelo algoritmo.

FIG. 4.5 - Cubo direcional rebatido

4.6.1 Tempo de recálculo de sombras

Os gráficos das figuras 4.6 e 4.7 mostram os tempos de processamento do módulo *sserver* para recálculo de sombras das duas cenas-teste, em função da

resolução (número de píxeis) do Z-buffer. As figuras 4.8 e 4.9 mostram o mesmo gráfico, agora em função do número de pontos de intersecção (*hit points*) testados para sombra das duas cenas-teste. Estes tempos correspondem àquele necessário ao processamento das etapas destacadas na figura 4.10.

FIG. 4.6 - Tempo de recálculo de sombras x resolução Z-buffer (Cena "Bule")

FIG. 4.7 - Tempo de recálculo de sombras x resolução Z-buffer (Cena "Interior")

FIG. 4.8 - Tempo de recálculo de sombras em função do num. de *hit points* (Cena "Bule")

FIG. 4.9 - Tempo de recal. de sombras x num. de *hit points* (Cena "Interior")

Conforme se poderia prever, os gráficos apresentam comportamento praticamente linear tanto em função do aumento da resolução dos *Z-buffers* quanto em função do número de pontos a serem testados.

O processamento executado no servidor de sombras pode ser dividido em duas partes principais, conforme o diagrama da figura 4.10.

FIG. 4.10 - Fluxo de processamento do componente sserver

A primeira etapa trata de projetar todos os polígonos no(s) *shadow buffer(s)*, executando simultaneamente o algoritmo Z-buffer. Implica no cálculo da projeção dos vértices de cada polígono e na interpolação e comparação de valores de profundidade. A duração da primeira fase depende do número de vértices a serem projetados e também da resolução em que serão projetados nos *shadow buffers*. Para resoluções altas e poucos vértices, este processamento deve ser dominado pelo custo de interpolação e o tempo será aproximadamente linear com o número de píxeis do *shadow buffer*.

Na segunda etapa são testados cada um dos *hit-points*. Este teste implica apenas na transformação das coordenadas do ponto, do espaço interno para o espaço da fonte de luz e uma comparação das coordenadas *Z* resultantes. Claramente, deve ser linear com o número de pontos testados.

Dessa forma, pode-se modelar o tempo de processamento total para recálculo de sombras (sserver) através da seguinte equação:

$$t_p = res_sb \cdot a + n_vert \cdot b + n_hits \cdot c \quad (4.10)$$

onde:

<i>res_sb</i> :	número de píxeis do shadow buffer;
<i>n_vert</i> :	número de vértices projetados;
<i>n_hits</i> :	número de pontos testados;

Os fatores *a*, *b* e *c* acima podem ser calculados a partir dos gráficos levantados, resultando na seguinte equação⁹ para tempo de recálculo de sombras:

$$t_p = res_pbl \cdot 8.0E-07 + n_vert \cdot 1.3E-05 + n_hits \cdot 3.5E-06 \text{ (segundos)} \quad (4.11)$$

A equação 4.10 apresenta erro máximo menor que 10% e erro médio de 3.4% quando comparada aos valores experimentais coletados para a cena "Bule".

⁹ obviamente válida somente nas condições (plataforma, cena, etc.) desta avaliação.

Os resultados obtidos mostram também que o aumento da resolução do *shadow buffer* tem influência relativamente baixa no tempo total de processamento. O gráfico da figura 4.8 mostra pouca diferença no tempo de processamento para as resoluções 10x10, 32x32, 50x50, 100x100, 320x320 e 500x500 píxeis. Assim, é possível utilizar-se resoluções até 500x500 (250.000 píxeis → sobrecarga de 0.2 s, pela equação 4.11), obtendo-se tempos de processamento interativos no recálculo de sombras.

Vale observar também que o tempo de processamento para recálculo de sombras também varia um pouco em função da posição da fonte de luz, já que a área ocupada pelos polígonos projetados varia em função da posição relativa dos objetos da cena em relação à fonte de luz. Por sua vez, o tempo para interpolação das profundidades dos polígonos é proporcional às suas áreas projetadas.

4.6.2 Tempo de comunicação para recálculo de sombras

O tempo total de comunicação entre os componentes *sserver* e *master* é composto pelo tempo para envio da requisição de recálculo de sombra e pelo tempo de envio/recepção da resposta computada.

O primeiro termo é desprezível já que se trata apenas do envio da nova coordenada da fonte de luz (3 números em ponto flutuante), totalizando menos que 30 bytes.

O segundo termo depende do número de pontos de intersecção recalculados. O gráfico da figura 4.11 mostra este tempo¹⁰ em função do número de bytes transmitidos.

FIG. 4.11 - Tempo de comunicação para envio de resultado do recálculo de sombra

¹⁰ inclui também o tempo do envio dos dados do componente *master* aos *slaves*, operação realizada simultaneamente à transmissão do *sserver*.

Observa-se que o tempo de comunicação do servidor de sombra com o *master* é significativo, chegando a representar até um terço do tempo total de processamento.

4.6.3 Variação do limiar de comparação de profundidade (Z-buffer).

Conforme mencionado anteriormente, a grande vantagem da utilização de um Z-buffer em cada face do cubo direcional ao invés do *Pixel-Light-Buffer* é a possibilidade de eliminar-se falso auto-sombreamento pela utilização de um limiar, conforme a equação 4.9.

A utilização de tal limiar é necessária devido à precisão limitada (geralmente 16 bits) dos Z-buffers comumente implementados em software ou disponíveis em hardware.

É importante notar que as profundidades no Z-buffer são obtidas em forma normalizada, no intervalo [0.0, 1.0]. As distâncias de todos os pontos entre os planos de corte de visualização ("clipping") afastado (*far*) e próximo (*near*) são mapeadas para este intervalo. Assim, o valor do limiar utilizado foi definido de forma a se ajustar à distância absoluta representada no Z-buffer, de acordo com a equação abaixo:

$$Eps_{rel} = \frac{Eps_{abs}}{(Z_{far} - Z_{near})} \quad (4.12)$$

onde: Eps_{rel} : limiar relativo, utilizado na equação 4.9;
 Eps_{abs} : limiar absoluto, indicado pelo usuário;
 Z_{far} : distância¹¹ do ponto da cena mais distante;
 Z_{near} : distância do ponto da cena mais próximo;

Este limiar relativo deve ser calculado para cada face do cubo direcional.

As figuras a seguir mostram o efeito visual da variação deste limiar (absoluto). A cena "Bule" está contida aproximadamente dentro de um envoltório cúbico de 10 unidades de lado e a fonte de luz está a aproximadamente à distância de 15 unidades do centro deste envoltório.

Um $Eps_{abs} = 0.1$ significa uma tolerância na comparação de profundidades equivalente a 0.1 unidades no sistema de coordenadas utilizado. Para a posição da fonte de luz usada na geração das imagens da figura 4.12, implica num

¹¹ na verdade, a projeção da distância na direção Z (normal à face correspondente do cubo direcional).

$Eps_{rel} \cong 0.005$, ou 0.5% da distância do ponto mais distante da luz, na direção normal à face do cubo direcional considerado.

FIG. 4.12 - Efeito de variação do limiar de comparação de profundidade (Eps_{abs})

No caso da figura 4.12, o melhor resultado é obtido com $Eps_{abs} = 0.04$ ($Eps_{rel} \cong 0.002$). Assim, o limiar adotado deve ser bastante baixo geralmente, apenas o suficiente para resolver problemas de auto-sombreamento, sob pena de perder-se a eficácia da comparação.

Outra característica a ser notada é que a resolução numérica do Z-buffer só é linear quando se emprega projeção cilíndrica. Quando se emprega projeção perspectiva (centrada), tal como é necessário no caso de fonte de luz pontual, o mapeamento entre a profundidade do ponto ("eye z") e o valor armazenado no z-buffer ("window z") é bastante não linear.

A relação entre a profundidade (relativa) de um ponto e o correspondente valor no Z-buffer é mostrada na figura 4.13¹², para vários valores da relação entre as distâncias dos planos de corte próximo (*near*) e afastado (*far*).

FIG. 4.13 - Rel. entre a prof. relativa e o valor no Z-buffer para várias razões *far/near*

Assim, para obter-se a melhor precisão é necessário reduzir-se ao máximo a relação *near/far*. Isto se consegue aproximando ao máximo o plano de projeção (*near*) dos pontos a projetar. Esta otimização ainda não foi implementada no sistema.

¹² By David Blyte (<http://www.sgi.com/Technology/OpenGL/advanced97/notes/node18.html>)

4.6.4 Influência da resolução espacial do Z-buffer na sombra projetada

Assim como o controle do limiar de comparação de profundidades permite reduzir-se os erros de auto-sombreamento, o aumento da resolução espacial do Z-buffer permite diminuir-se os efeitos de serrilhamento nas sombras projetadas.

FIG. 4.14 - Efeito da resolução espacial do Z-buffer no serrilhamento da sombra

As imagens da figura 4.14 mostram o efeito da resolução espacial dos Z-buffers do cubo direcional no serrilhamento das sombras projetadas.

As imagens da figura 4.14 foram geradas à resolução de 1000x1000 píxeis. Já as imagens da figura 4.15 foram geradas em 300x300 píxeis. Percebe-se que a resolução espacial de Z-buffer necessária para obter-se sombras de boa qualidade varia pouco em função da resolução da imagem calculada.

FIG. 4.15 - Efeito da res. do Z-buffer no serrilhamento da sombra (300x300)

Obtém-se efeito razoável com resolução de Z-buffer de 320x320 para esta imagem. Conforme foi mencionado na seção 4.6.1, a sobrecarga de processamento, na máquina testada, é bastante tolerável para resoluções do Z-buffer até 500x500, cujo resultado é muito próximo do obtido para resolução de 1000x1000 no Z-buffer.

Deve-se mencionar também que, para as aplicações típicas do ray tracing parametrizado, o efeito de serrilhamento nas sombras não causa problemas críticos já que o mais importante é obter-se uma noção da posição das sombras projetadas e não uma imagem com qualidade final para utilização.

4.6.5 Uso de acelerador gráfico

Um estudo ideal do impacto do uso de acelerador gráfico para cálculo de sombras exige que se possa desativar este dispositivo em uma certa plataforma, deixando o cálculo de projeções, do Z-buffer e as interpolações totalmente a cargo da implementação em software da biblioteca OpenGL. Isto, no entanto, nas plataformas que temos disponíveis, não é totalmente possível.

Realizou-se alguns ensaios na tentativa de observar o ganho proporcionado por um acelerador gráfico de alto desempenho¹³ como o disponível na estação Indigo² Maximum Impact. Os resultados são apresentados na gráfico da figura 4.16.

FIG. 4.16 - Avaliação do impacto no recálculo de sombra com uso de acelerador gráf.

Nos ensaios realizados, os módulos *master* e *slave* sempre foram executados em uma quarta estação, distinta das demais empregadas nos testes. O módulo *sserver* foi executado em três estações diferentes, conectado a diferentes servidores de janelas (X-Server), que são os verdadeiros responsáveis pela execução dos comandos OpenGL através da extensão GLX.

No gráfico da figura 4.16, as máquinas denominadas R e M têm a mesma configuração: processador de médio desempenho e sem aceleração gráfica. A máquina H tem alto desempenho em processamento geral e gráfico.

Observa-se uma melhoria no desempenho do recálculo de sombra, porém não se pode concluir quantitativamente qual é este ganho pois a diferença no processador principal do equipamento também influencia fortemente o resultado. A comunicação pela rede para exibição remota também influencia nos resultados.

¹³ 240M pixels/sec fill rate

4.6.6 Comparação do tempo de reavaliação com o tempo de síntese

A comparação do tempo de reavaliação com o tempo requerido pelo algoritmo convencional de ray tracing exige que se utilize a mesma plataforma e equação de iluminação na síntese da imagem e do arquivo de dados do sistema RT(p).

Ocorre que não temos disponível hoje um "shader" para o programa mental ray que se utilize exatamente da mesma equação de iluminação que temos implementada no sistema RT(p). Outro agravante é o fato da necessidade de executar-se o sistema RT(p) em equipamento Silicon Graphics e sistema operacional IRIX, pois é a plataforma nativa do mental ray.

Assim, apresenta-se na figura 4.17 desempenho comparativo de uma versão antiga do sistema RT(p) (utilizando o MTV ray tracer), que ilustra o ganho de tempo proporcionado pelo algoritmo de ray tracing parametrizado, cerca de 25 vezes mais rápido.

FIG. 4.17 - Comparação RT convencional x RT Parametrizado

Comparando-se o tempo de reavaliação quando há ou não movimentação de fontes de luz (fig. 6.2), observa-se que o primeiro pode dobrar em relação ao segundo. Ainda assim, o tempo de reavaliação permanece muito inferior ao necessário ao ray tracing convencional.

4.7 Conclusões

Podemos concluir que o algoritmo proposto cumpre seu papel na medida em que o custo do processamento de sombra é relativamente pequeno se comparado ao re-processamento total da cena pelo ray tracing convencional.

O ajuste do limiar de comparação é bastante simples de ser feito interativamente pelo usuário apesar de ser fundamental ao cálculo correto das sombras.

As resoluções espaciais do Z-buffer que permitem boa qualidade nas sombras projetadas são compatíveis com tempos de processamento interativos.

A implementação utilizando arquitetura cliente-servidor para o cálculo de sombra não mostrou-se totalmente adequada pois apesar de permitir melhor distribuição da carga de processamento na reavaliação, a sobrecarga de comunicação é significativa. Isto sugere que uma implementação local possa trazer melhor desempenho, apesar do consumo de memória.

Outro fato a se destacar é que o algoritmo proposto para o servidor de sombras é altamente paralelizável já que os testes dos pontos de intersecção são totalmente independentes entre si. Pretende-se no futuro testar uma implementação paralela onde cada nó de processamento disponha de uma placa gráfica aceleradora, e os testes de sombra sejam realizados localmente em cada nó, em paralelo. Além do ganho em processamento, toda a sobrecarga de comunicação relacionada ao cálculo de sombra seria eliminada.

CAPÍTULO 5

EFEITOS ESPECIAIS

5 Efeitos especiais

A partir dos algoritmos PLB/Shadow Buffer desenvolvidos no capítulo anterior apresentamos aqui uma proposta para parametrizar luzes do tipo spot e projetores de diapositivos ou projetores “gobo”¹⁴ dentro do contexto de ray tracing parametrizado.

Este tipo de recurso não estava disponível no algoritmo original devido ao fato do versor $\bar{L}_i$ ser uma constante naquele caso.

Outra possibilidade nova que introduzimos neste capítulo é a parametrização do mapeamento de relevo (bump-mapping).

5.1 Luzes spot

Luzes do tipo spot são fontes de luz com origem definida mas cuja emissão não é uniforme em todas as direções tal como ocorre com as fontes de luz pontuais/omnidirecionais. Normalmente elas produzem um cone (ou prisma de base retangular) de luz cuja intensidade decai longe do centro (WATKINS, 1992).

A variação da intensidade da luz em função do ângulo de emissão de uma fonte é caracterizada por um diagrama goniométrico como ilustrado na figura 5.1 abaixo (SILLION & PUECH, 1994).

FIG. 5.1– Diagrama goniométrico de uma fonte de luz

A menos que a luminária tenha simetria cilíndrica, os diagramas goniométricos são dados para múltiplos planos que devem ser interpolados, se for o caso.

¹⁴ Veja definição de projetor gobo na seção 5.2

É tradicional em programas de computação gráfica simplificar-se a especificação completa de múltiplos diagramas goniométricos de forma a facilitar o controle de fontes de luz do tipo spot. Assim pode-se, por exemplo, adotar um decaimento linear da intensidade luminosa do centro para as bordas do cone, a partir de um ângulo máximo (WATKINS, 1992) como indicado na equação 5.1 e figura 5.2:

$$I = I_{\max} \cdot \begin{cases} 1, & \text{se } ang \leq ang_{\min} \\ (ang_{\max} - ang) / (ang_{\max} - ang_{\min}), & \text{se } ang_{\min} < ang < ang_{\max} \\ 0, & \text{se } ang \geq ang_{\max} \end{cases} \quad (5.1)$$

FIG. 5.2– Variação da intensidade da luz conforme equação 5.1

Outra alternativa é escolher uma interpolação cúbica, fazendo com que a maior parte do decaimento ocorra próximo à borda (ELLIOTT & MILLER, 1995).

Assim, a especificação completa de uma fonte de luz spot se faz pelos seguintes parâmetros:

- intensidade máxima;
- ângulo máximo;
- ângulo mínimo;
- tipo de decaimento angular (linear, cúbico, etc.)
- posição;
- direção ou alvo.

A disponibilidade de fontes de luz spot contribui para aumentar o realismo das cenas sintetizadas e são de fundamental importância na simulação visual para auxílio ao projeto de iluminação (luminotécnica). O uso de simulação visual realística para o projeto de iluminação vem crescendo (SHAKESPEARE, 1996) (ASHDOWN, 1996) e acreditamos que uma ferramenta baseada nas propostas desta tese tornará mais eficiente a atividade de projeto de iluminação com auxílio do computador.

5.1.1 Parametrização de luzes spot

No algoritmo original proposto por Séquin e Smyrl a utilização de luzes do tipo spot era permitida (apesar de não mencionada explicitamente), porém somente se podia variar a intensidade e cor de tais luzes. A parametrização completa de fontes de luz deste tipo passa a ser possível como uma decorrência das extensões que propusemos. Podem ser alterados a direção e ângulo de abertura do spot, além do tipo de decaimento de intensidade.

Como as coordenadas dos pontos de intersecção são armazenadas e as posições das fontes de luz são parâmetros no algoritmo proposto, pode-se calcular com facilidade o ângulo de emissão da luz em relação ao ponto iluminado. Assim, a intensidade da fonte de luz deve ser calculada de acordo com uma equação do tipo 5.1.

Uma especificação mais detalhada da fonte de luz spot, dada por um diagrama goniométrico, pode ser implementada de forma eficiente usando-se um algoritmo semelhante ao que propomos para a parametrização de efeitos de projeção, conforme mostraremos na secção 5.2, além de permitir a simulação de spots não circulares.

5.1.2 Implementação e resultados

No estágio atual do sistema RT(p) somente foram implementadas luzes spot com decaimento linear e formato circular.

Devido ao fato de termos optado pela implementação de um servidor de sombras (*sserver*), isto é, um componente separado do programa *master*, o cálculo de intensidade através do mapeamento no PLB pelo *sserver* exigiria que este comunicasse ao *master* não só a condição de iluminação de cada ponto mas também a intensidade em cada ponto testado. Como para o caso de fontes circulares e decaimento linear este cálculo é muito simples, optou-se por processá-lo no próprio *master*, economizando-se precioso tempo de comunicação.

A implementação realizada utiliza a equação 5.1, porém são testados os cossenos dos ângulos ao invés dos próprios ângulos para aproveitar-se o cálculo do produto escalar.

A figura 5.3 mostra o efeito da variação da posição da fonte de luz spot, mantendo fixo o alvo do spot.

FIG. 5.3 - Variação da posição de fonte de luz spot

A figura 5.4 ilustra mudança na direção do spot, mantendo-se fixos os outros parâmetros.

FIG. 5.4 - Variação da direção da fonte de luz spot

A mudança na abertura do cone do spot está mostrada na figura 5.5.

FIG. 5.5 - Variação do ângulo de abertura do cone de fonte de luz spot

A fig. 5.6 mostra alteração no ângulo de máxima intensidade.

FIG. 5.6 - Variação do ângulo de máxima intensidade - fonte de luz spot

A sobrecarga de tempo de processamento para cálculo de fonte de luz spot é desprezível já que trata-se apenas de calcular um produto escalar e comparar seu resultado com os limiares armazenados para cada fonte de luz, multiplicando a intensidade da fonte de luz pelo fator adequado, se o ponto for iluminado. O processamento de luz spot é feito apenas para os pontos potencialmente iluminados ("enxergues" pela fonte de luz) e somente para as fontes de luz com processamento spot habilitado, reduzindo ainda mais seu impacto.

5.2 Projetores

Projetores têm uma forte tradição de uso em teatro e em projetos de interiores e iluminação (ELLIOTT & MILLER, 1995). Além da projeção de diapositivos, outros efeitos especiais são também bastante usados. Quando a imagem projetada é opaca (preto sobre branco-transparente), será projetada uma sombra. Esta técnica é conhecida por projeção “gobo” (figura 5.7). Este tipo de recurso é usado em Computação Gráfica para a obtenção de vários efeitos especiais que, realizados de outra forma, demandariam grande esforço de modelamento.

FIG. 5.7 – Projeção gobo e padrão projetado

5.2.1 Parametrização de projetores

O algoritmo PLB que propusemos, conjugado com as luzes spot da secção anterior, sugere também uma nova extensão que permite que parametrize-se efeitos de projeção.

Se associarmos à cada face do PLB outros buffers que contêm o mapa de píxeis (“pixmap”) da imagem ou máscara que deseja-se projetar, poderemos obter efeitos especiais de projeção, bastando associar o controle da cor e intensidade da luz projetada em certa direção ao píxel correspondente nestes buffers (figura 5.8).

FIG. 5.8 – Efeito de projeção e padrão projetado

Conforme já mencionado, pode-se usar técnica semelhante para controlar com precisão a especificação dos diagramas goniométricos de uma fonte de luz spot. Para isso, basta usar-se uma imagem de controle em tons de cinza, mapeando-se a intensidade de luz desejada em cada direção.

O recurso de projetores será futuramente implementado no sistema RT(p).

5.3 Mapeamento de Relevô

Mapeamento de Relevô (“bump-mapping”) é uma técnica introduzida por James Blinn (BLINN, 1978) que permite obter-se a aparência de baixo ou alto relevô em uma superfície sem a necessidade de modelar-se geometricamente estes detalhes (figura 5.9a).

a) esfera com mapeamento de relevô

b) mapa de relevô utilizado

FIG. 5.9 – Mapeamento de relevô

A técnica consiste em perturbar-se a normal usada para cálculo de iluminação em cada ponto da superfície. Esta perturbação é controlada por um mapa, normalmente uma imagem com o padrão que se quer reproduzir na superfície (fig. 5.9b e fig. 5.10).

FIG. 5.10 – O algoritmo de mapeamento de relevo (1D)¹⁵

O mapeamento de relevo apresenta dois artefatos que denunciam o uso desta técnica nas imagens geradas. O primeiro é que o relevo não aparece nas arestas de silhueta quando o objeto, visto de perfil, deveria mostrar sulcos e / ou protuberâncias. Este problema pode ser evitado com a adoção da técnica de mapeamento de deslocamento (“displacement mapping”) (COOK, 84) que normalmente é incorporada na fase de modelamento e que, no entanto, efetivamente muda a geometria do modelo, impedindo seu uso em ray tracing parametrizado. O segundo é que as sombras projetadas não seguem a geometria percebida e sim a realmente modelada. Noma e Sumi (NOMA & SUMI, 94) apresentam uma proposta para correção deste problema.

5.3.1 Parametrização de mapeamento de relevo

A possibilidade de usar-se mapeamento de relevo no contexto de ray tracing parametrizado também deriva naturalmente das extensões que propusemos.

A necessidade de se recalcularem os produtos escalares da equação de iluminação ($\vec{N} \cdot \vec{L}$ e $\vec{N} \cdot \vec{H}$) traz a possibilidade de se parametrizar o vetor normal $\vec{N}$, a única operação necessária no mapeamento de relevo.

¹⁵ Figura adaptada de (WATT & WATT, 1992)

O mapeamento de relevo é implementado através do seguinte equacionamento:

$$\vec{N}' = \vec{N} + \vec{D} \quad (5.2)$$

A nova normal no ponto de intersecção ($\vec{N}'$) será obtida pela soma com um vetor de perturbação $\vec{D}$, dado pela equação 5.3 (BLINN, 1978):

$$\vec{D} = \frac{\partial F}{\partial u} (\vec{N} \times \vec{P}_v) - \frac{\partial F}{\partial v} (\vec{N} \times \vec{P}_u) \quad (5.3)$$

$F(u, v)$ é o mapa de relevo, isto é, uma matriz bidimensional cujos valores indicam a altura do relevo naquele ponto. Suas derivadas parciais podem ser aproximadas por diferenças simples:

$$\frac{\partial F}{\partial u} \cong \Delta F_u = [F(u + \varepsilon, v) - F(u - \varepsilon, v)] / 2\varepsilon \quad (5.4)$$

$$\frac{\partial F}{\partial v} \cong \Delta F_v = [F(u, v + \varepsilon) - F(u, v - \varepsilon)] / 2\varepsilon \quad (5.5)$$

$\vec{P}_u$ e $\vec{P}_v$ são vetores ortogonais ao vetor normal e, portanto, tangentes à superfície ($\vec{N} = \vec{P}_u \times \vec{P}_v$). São chamados de vetores base do relevo ("bump basis vectors") e definem a orientação do mapeamento de relevo na superfície. Tanto a normal quanto estes vetores são extraídos diretamente da geometria da cena.

A implementação deste recurso no sistema RT(p) é simples, porém, nesta fase dos trabalhos, ainda não foi implementado.

A disponibilização deste recurso implicará em maior demanda de memória já que os vetores base do relevo deverão ser armazenados também, juntamente com a geometria da cena.

CAPÍTULO 6

OTIMIZAÇÕES

6 Otimizações¹⁶

As otimizações propostas neste capítulo são de dois tipos:

- otimização de velocidade de execução através de implementação paralela do algoritmo;
- otimização de velocidade de resposta ao usuário através da reavaliação incremental.

6.1 Paralelização

A paralelização eficiente do algoritmo de ray tracing parametrizado, levando-se em conta apenas a fase de reavaliação da imagem, é um problema bastante diferente da paralelização do algoritmo convencional de ray tracing que abordamos detalhadamente em (SANTOS, 1994a). Neste último caso, o principal problema era evitar-se a replicação das estruturas de dados de descrição da cena em todos os nós do sistema, mantendo-se baixa a quantidade de dados transmitidos entre os nós e cuidando-se de manter o balanceamento de carga uniforme.

Na paralelização do ray tracing parametrizado a maior parte destes problemas desaparece.

Se não forem adotadas algumas técnicas de otimização tais como o compartilhamento de sub-expressões com “hashing” (secção 2.2.1.5) as estruturas de dados de cada píxel serão totalmente independentes. Por outro lado, pode-se garantir um excelente balanceamento estático de carga através do particionamento de dados no espaço de imagem com subdivisão uniforme e alocação distribuída (fig. 6.1).

FIG. 6.1 - Partic. de imagem com subdivisão unif. e alocação distr. a 4 processadores

¹⁶ Usamos o termo otimização no sentido de melhora de desempenho

Dessa forma, o principal problema envolvido com a paralelização do ray tracing parametrizado torna-se a comunicação de dados. Esta estará intrinsecamente relacionada à granularidade do processamento e à latência de comunicação as quais, no contexto deste trabalho, define-se a seguir:

Granularidade:

A quantidade de operações feitas por um nó entre dois eventos de comunicação. Denomina-se granularidade fina (*fine-grain*) quando são realizadas poucas operações entre duas comunicações ou espessa (*coarse-grain*) se um processo realiza muitas operações entre dois eventos de comunicação.

Latência:

A parcela do tempo necessário para enviar uma mensagem de um nó a outro que é independente do tamanho da mensagem (i.e., o tempo necessário para enviar uma mensagem de tamanho zero).

A latência é um atributo da plataforma utilizada. A granularidade pode, em certa medida, ser ajustada no algoritmo em questão em função da latência e do desempenho dos sistemas de processamento e comunicação da plataforma alvo.

Neste trabalho, escolheu-se o sistema SPADE-II (KOFUJI, 1994) para a implementação paralela deste algoritmo. Nos itens seguintes serão discutidas as estratégias de paralelização adotadas, a metodologia de ensaio e os resultados obtidos.

6.1.1 Estratégia

A estratégia básica adotada nesta paralelização do algoritmo de ray tracing parametrizado foi a “fazenda de processadores” (MAY & SHEPHERD, 1987). Neste enfoque, um processo mestre distribui tarefas para um grupo de processos escravos em execução em outros nós de processamento. Estes, ao completarem o processamento de suas tarefas, enviam os resultados ao processo mestre e recebem nova tarefa caso disponível.

Adotou-se particionamento de imagem com subdivisão uniforme e alocação distribuída, sendo cada tarefa constituída por uma linha completa da imagem e alocando-se ao i -ésimo processo escravo as linhas $\{i, i+N, i+2N, \dots\}$, onde N é o número total de processos escravos alocados (fig. 6.1).

Como adotou-se uma subdivisão uniforme com alocação distribuída, isto é, balanceamento estático de carga, os processos escravos já conhecem de antemão o

conjunto de tarefas, neste caso, pacotes de píxeis, a processar, reduzindo a comunicação necessária para reavaliação de uma imagem. Ao iniciar-se a carga das estruturas de dados, o processo mestre informa a todos os processos escravos qual o nome do arquivo de dados a ser processado e também qual o número total de processos alocados ao processamento.

Ao ser requisitada uma reavaliação de imagem, o processo mestre comunica a todos os processos escravos os novos parâmetros para reavaliação.

A granularidade do processo pode ser minuciosamente calibrada através do tamanho do pacote de dados comunicado a cada vez. Este pode variar de um único píxel até o conjunto completo das linhas atribuídas a um processo escravo. Na implementação atual é transmitida uma linha completa de píxeis em cada evento de comunicação.

A quantidade de processamento ligada a cada píxel varia em função da profundidade da árvore de raios a ele associada e do tipo de cada nó desta árvore. Assim, um píxel que mostre apenas a cor de fundo da cena terá associada uma árvore de profundidade 1 enquanto píxeis mostrando objetos reflexivos e transparentes poderão atingir a máxima profundidade permitida, implicando em maior tempo de processamento. Da mesma forma, árvores com nós texturizados e/ou com mapeamento de relevo ou ainda iluminados por muitas fontes de luz, demandarão maior tempo de processamento.

O processo de otimização do processamento paralelo deste algoritmo se resume a encontrar a granularidade ideal para a imagem reavaliada na plataforma alvo. Esta indicará qual a quantidade de píxeis que devem ser processados entre cada evento de comunicação.

O tempo de comunicação de um pacote pode ser modelado pela equação 6.1.

$$T_c = T_l + T_t \cdot n_bytes \quad (6.1)$$

onde: T_c : tempo de comunicação;
 T_l : tempo de latência;
 T_t : tempo de transmissão/byte;
 n_bytes : número de bytes a transmitir;

O tempo total de comunicação pode ser modelado pela equação 6.2.

$$T_{Tc} = T_l \cdot res_v + T_t \cdot res_v \cdot win_h \quad (6.2)$$

onde: T_{Tc} : tempo total de comunicação;
 T_l : tempo de latência;
 T_t : tempo de transmissão/byte;
 res_v : resolução vertical da imagem pré-processada;
 win_h : resolução horizontal da janela de apresentação;

6.1.2 Metodologia de Avaliação

As mesmas cenas usadas nas avaliações apresentadas nos capítulos anteriores foram utilizadas para testar-se o desempenho das implementações paralelas do sistema RT(p).

Utilizou-se uma plataforma da família SPADE-II nos testes de paralelização desenvolvidos, cujas características encontram-se na tabela 6.1 abaixo:

CARACTERÍSTICAS	AQUILA09-16
Número de nós	8
Número de proc./nó	2
Tipo do processador	Intel PentiumPro 200 MHz
Memória por nó	256 Mbytes
Rede de interconexão	FastEthernet + Switch
Largura de banda ¹⁷	100 Mbits/s
Protocolo comunicação	TCP/IP
Sistema Operacional	Linux Red Hat v.4.2

TAB. 6.1- Características do sistema empregado nos testes de paralelização

Foram feitos ensaios usando de 1 a 8 processadores com um processo *slave* por nó.

Conforme mencionado anteriormente, os módulos *slave* calculam todos os píxeis de uma linha, conforme a resolução original da imagem pré-processada, enviando os resultados ao módulo *master*, linha-a-linha. Porém, antes de enviar os píxeis ao *master*, o módulo *slave* faz um escalamento horizontal da linha, de acordo com a resolução atual da janela de apresentação da imagem. Assim, o número de bytes transmitidos em cada evento de comunicação depende exclusivamente da resolução horizontal da janela de apresentação. O escalamento vertical da imagem é feito no módulo *master* de forma que o número de eventos de comunicação é fixo e igual à resolução vertical da imagem original pré-processada.

¹⁷ Nominal

Desta forma, as cenas foram geradas em três resoluções distintas porém mantendo-se o mesmo número total de píxeis (250.000), pois é uma indicação do total de processamento requerido pela cena. Estas três versões foram computadas com a janela de apresentação também em três diferentes resoluções¹⁸, de forma a variar-se o total de comunicação em cada caso. As várias combinações ensaiadas estão indicadas na matriz da tabela 6.2, onde a quantidade e tamanho dos pontos é proporcional ao número e tamanho dos pacotes transmitidos.

Características dos pacotes de dados transmitidos			
Resolução de apresentação	Res. imagem pré-processada		
	1000x250	500x500	250x1000
640x480	●	●●	●●●
320x480	•	••	•••
160x480	•	••	•••

TAB. 6.2 - Matriz de ensaios realizados

Conforme a equação 6.2, a latência é proporcional ao número de pacotes transmitidos (resolução vertical da imagem pré-processada) sendo detectável pela comparação entre os testes numa mesma diagonal (esquerda para direita) da tabela 6.2/6.3 onde o número de bytes transmitidos é constante (tabela 6.3). O tempo de comunicação variável com o quantidade de bytes transmitidos poderá ser avaliado através da comparação entre testes numa mesma coluna da tabela 6.2.

Total de píxeis transmitidos em cada ensaio			
Resolução de apresentação	Res. imagem pré-processada		
	1000x250	500x500	250x1000
640x480	160.000	320.000	640.000
320x480	80.000	160.000	320.000
160x480	40.000	80.000	160.000

TAB. 6.3 - Número total de píxeis¹⁹ transmitidos em cada ensaio

A cena "Interior" foi processada apenas em resolução 300x300 pois é complexa demais para ser gerada em resoluções muito maiores.

¹⁸ A resolução vertical da janela de apresentação não tem nenhuma influência na comunicação.

¹⁹ Os píxeis são computados e transmitidos em 24bits/píxel. Portanto o número de bytes transmitidos é três vezes maior.

6.1.3 Resultados e Análises

Os resultados obtidos estão resumidos nos gráficos das figuras a seguir. Os dados numéricos encontram-se no Anexo C.

FIG. 6.2 - Tempo de processamento para várias resoluções da cena Bule (1)

FIG. 6.3 - Tempo de processamento para várias resoluções da cena Bule (2)

FIG. 6.4 - Tempo de processamento para várias resoluções da cena Bule (3)

FIG. 6.5 - Tempo de processamento paralelo para a cena "Interior"

6.1.3.1 Tempo de processamento

Analisando-se os gráficos de tempo de processamento em função do número de processadores percebe-se que sempre ocorre saturação, tornando o ganho de processamento nulo quando empregados mais que um número N de processadores. Esta saturação ocorre devido à comunicação entre os processos *master* e *slave*, e N é tanto maior quanto menor for a quantidade de comunicação realizada.

Para a cena "Bule", de menor complexidade, a saturação ocorre rapidamente já que o tempo de comunicação domina o processamento. Os dados obtidos para esta cena serão utilizados na seção 6.1.3.2 para analisar-se a influência do tempo de comunicação no tempo total de processamento.

A cena "Interior", muito mais complexa que a cena "Bule" permite observar um bom ganho de processamento paralelo, aproximadamente linear até 6 processadores conforme mostra o gráfico de ganho da figura 6.6.

A cena processada em resolução de apresentação de 640x480 satura mais rapidamente porque a quantidade de comunicação a ela associada é maior.

FIG. 6.6 - Ganho de processamento paralelo - cena "Interior"

6.1.3.2 Tempo de comunicação

Com as medidas tomadas para a cena Bule, procurou-se detectar qual a influência da latência no tempo total de computação. Observando-se qualquer um dos três gráficos de processamento da cena Bule (fig. 6.2 a 6.4) na região de saturação (na qual a carga de processamento de cada nó é muito pequena) percebe-se claramente que o tempo total de computação não é influenciado de forma significativa pela quantidade de pacotes.

FIG. 6.7 - Gráfico para análise de latência para várias quantidades de comunicação

Concluimos assim, que a latência é desprezível neste tipo de processamento e que a quantidade de bytes transmitidos é completamente dominante no tempo de comunicação.

A análise do tempo de comunicação em função do número de bytes transmitidos revela um gráfico praticamente linear, conforme mostra a figura abaixo. Os tempos que constam neste gráfico incluem, além do tempo de comunicação, o tempo de processamento. A taxa verificada é de aproximadamente 7 Mbits/s.

FIG. 6.8 - Gráfico para análise da taxa de transmissão

6.2 Reavaliação incremental

Observando a forma típica de utilização do sistema RT(p), propusemos o desenvolvimento de um modo de reavaliação em que apenas um parâmetro de cada vez fosse modificado em relação à última reavaliação solicitada.

Um usuário normalmente, ao interagir com o sistema RT(p), imediatamente solicita uma reavaliação da imagem após alterar o valor de um parâmetro para que possa observar os efeitos da modificação efetuada.

Vários fatores levam a este comportamento:

- a) a tendência normal é que o usuário concentre-se no ajuste de um material por vez;
- b) os efeitos da mudança dos parâmetros da equação de iluminação não são óbvios e o usuário precisa da realimentação do sistema para se orientar;

- c) há uma grande quantidade de parâmetros disponíveis para modificação. Alterando-se vários simultaneamente torna-se bastante difícil atingir o resultado desejado;
- d) como o tempo de reavaliação é bastante baixo, o custo (espera) de solicitar nova reavaliação é pequeno e estimula o usuário a fazê-lo.

A árvore de raios associada à cada píxel de uma imagem contém referências apenas aos materiais das superfícies atingidas por aqueles raios. Desta forma, nem todos os píxeis de uma imagem são afetados pela mudança de apenas alguns dos parâmetros. Assim, procuramos desenvolver um sistema onde apenas os píxeis afetados fossem reavaliados, reduzindo o tempo de resposta.

6.2.1 Implementação

Para a implementação da proposta de ray tracing parametrizado incremental propomos o uso de uma estrutura de dados que permite que só se recalculam os píxeis que são afetados pelo parâmetro modificado.

6.2.1.1 Recálculo seletivo de píxeis

Para que se possa identificar rapidamente quais são os píxeis afetados por uma alteração solicitada pelo usuário, foi associada à estrutura de dados de cada material um ponteiro para uma sequência de bits (1 bit/píxel) que indica se a árvore de raios daquele píxel contém ou não referências àquele material. Estas estruturas são todas criadas no pré-processamento, anterior à fase de reavaliações.

Estas seqüências de bits são posteriormente comprimidas através de codificação "run-length" (RLE) (Foley et. al., 1990), de acordo com o esquema da figura 6.9:

FIG. 6.9 - Codificação RLE da estrutura de reavaliação incremental

As imagens normalmente têm grande coerência, de forma que a compressão RLE será bastante eficiente neste caso. Como a coerência da imagem é 2D supõe-se que uma codificação RLE 2D ou *quad-tree* seja mais eficiente que o RLE 1D que adotamos. No entanto, a razão desta escolha é que o fato de usar-se alocação distribuída linha a linha (fig. 6.1) na paralelização dos algoritmos, impede a adoção de estruturas que englobem linhas contíguas.

Estas estruturas de dados só são criadas para alterações nos materiais já que alterações nas fontes de luz (cor ou posição) têm grande possibilidade de afetar quase todos os píxeis da imagem, sendo pouco eficiente.

6.2.2 Resultados e análises

As tabelas 6.4 e 6.5 mostram o tamanho médio das corridas da estrutura proposta e os fatores de compressão atingidos para as duas cenas-teste.

Bule 500x500 - Estrutura sem compactação: 31.250 x 3 = 91.5Kb			
<i>Material</i>	<i>Tam. médio das corridas</i>	<i>Tam. total</i>	<i>Compressão</i>
even_tile	56.7	4.408 bytes	1:7.0
odd_tile	61.3	4.079 bytes	1:7.7
teapot	102.3	2.443 bytes	1:12.8
Total	68.6	10.930 bytes	1:8.6

TAB. 6.4 - Tam. médio e compressão da estrutura para reaval. incremental (Bule)

InteriorT4 - Estrutura sem compactação: 11.250 x 11 = 120.8Kb			
<i>Material</i>	<i>Tam. médio das corridas</i>	<i>Tam. total</i>	<i>Compressão</i>
1	59.1	1.524 bytes	1:7.4
2	28.1	3.203 bytes	1:3.5
3	37.5	2.398 bytes	1:4.7
4	71.4	1.260 bytes	1:8.9
5	37.7	2.390 bytes	1:4.7
6	42.3	2.130 bytes	1:5.3
7	29.2	3.084 bytes	1:3.6
8	32.0	2.811 bytes	1:4.0
9	26.0	3.455 bytes	1:3.3
10	70.7	1.273 bytes	1:8.8
11	73.3	1.227 bytes	1:9.2
Total	40.0	24.755 bytes	1:5.0

TAB. 6.5 - Tam. médio e compressão da estrutura para reaval. incr. (InteriorT4)

As corridas têm tamanho médio bastante abaixo do máximo (128) de forma a confirmar o acerto na escolha do formato para armazenamento do comprimento da corrida: 7 bits são suficientes e 6 bits (máx. 65) fica abaixo da média da cena "Bule".

Os fatores de compressão obtidos para as cenas teste são razoáveis (1:8.6 e 1:5), e assim, concluímos que a estrutura proposta ocupa pouco espaço.

Os gráficos das figuras 6.10 a 6.13 mostram a correlação entre o tempo relativo de reavaliação incremental e a área relativa da imagem reavaliada. A correlação é bastante forte. Nota-se também que a distância entre as duas curvas é proporcional ao tempo de comunicação (relativo ao tempo total de processamento), que é maior na cena InteriorT1 e menor na InteriorT4.

FIG. 6.10 - Correlação entre área e tempo de reavaliação - cena InteriorT1

FIG. 6.11 - Correlação entre área e tempo de reavaliação - cena InteriorT2

FIG. 6.12 - Correlação entre área e tempo de reavaliação - cena InteriorT3

FIG. 6.13 - Correlação entre área e tempo de reavaliação - cena InteriorT4

A figura 6.14 mostra os tempos de reavaliação completa e incremental da imagem em função da resolução horizontal da janela de apresentação (que é diretamente proporcional à quantidade de comunicação realizada). São mostradas curvas para reavaliação completa e reavaliação incremental do material aplicado ao bule e a um dos ladrilhos da superfície da cena "Bule500", além da diferença entre esses tempos. Nota-se que o ganho absoluto de tempo permanece constante, independente da resolução da janela de apresentação, isto é, da quantidade total de comunicação, conforme esperado.

FIG. 6.14 - Comparação de tempos reavaliação completa x incremental

O gráfico da figura 6.15 mostra o ganho da reavaliação incremental para as cenas InteriorT1 a T4.

FIG. 6.15 - Ganho de velocidade com reavaliação incremental

Observa-se que, para a cena analisada, o ganho pode chegar a mais de 6 vezes. O ganho médio foi de 3.3 vezes. Por outro lado, o ganho também pode até ser negativo, caso a área atingida pela alteração ser muito próxima à área total da imagem. Isto ocorreu (-1%) para a reavaliação de área igual a 96% (material 1 - InteriorT4) da imagem e é devido à pequena sobrecarga decorrente da avaliação da estrutura compactada.

6.3 Conclusões

Nossa proposta de reavaliação incremental foi demonstrada bastante válida já que implica em sobrecarga de memória baixa devido à sua compressão e proporciona ganhos de velocidade consideráveis. Além disso, é adequada ao processamento paralelo do algoritmo.

Em relação ao processamento paralelo, concluímos que o tempo total de processamento é altamente dominado pela comunicação nesta aplicação. Este fato é comum quando se paraleliza aplicações de computação gráfica interativa. Como a grande massa de dados transmitidos se refere exclusivamente aos resultados computados, esta é difícil ser reduzida. Uma alternativa possível para amenizar o problema é a compressão dos dados, antes de enviá-los ao módulo *master*. Um algoritmo simples como o RLE (Run-Length Encoding) possivelmente traria algum ganho na comunicação sem causar demasiada sobrecarga de processamento.

Os ensaios realizados mostram que a latência tem pouca influência no tempo total de comunicação, descartando-se assim as possibilidades de ganho através da calibração do tamanho dos pacotes enviados.

A solução para este tipo de problema é o uso de *frame-buffers* distribuídos, tais como os propostos em (WEI et al., 1997) para o Intel Paragon e Sistema SHRIMP da Princeton University ou em (McDonald, 1993) para a J-Machine do MIT ou ainda outras propostas mais antigas (POTMESIL et al., 1989), (JANSEN, 1988), (COPPEN et. al., 1992).

Este tipo de solução permite que todos os nós de processamento tenham acesso direto ao *frame-buffer*, eliminando-se o tempo de comunicação o que garantiria, em nosso caso, ganho de velocidade linear para muitos processadores.

CAPÍTULO 7

CONCLUSÕES E TRABALHOS FUTUROS

7 Conclusões e trabalhos futuros

7.1 Conclusões

Ray tracing parametrizado é uma técnica ainda pouco utilizada assim como o é, em menor grau, o ray tracing. O desempenho dos processadores atuais já permite que estas técnicas possam fazer parte de aplicações disponíveis a grande número de usuários.

O trabalho que desenvolvemos no âmbito desta tese abre novas frentes de aplicações para estes algoritmos especialmente, como já mencionamos, na área de projeto de iluminação e melhora as aplicações de desenvolvimento de animações.

O algoritmo que propusemos e a implementação realizada obtiveram sucesso atingindo os objetivos de permitir a rápida movimentação das fontes de luz da cena, a parametrização de luzes spot, projetores e mapeamento de relevo.

A reavaliação incremental proposta é uma nova técnica no contexto do ray tracing parametrizado que mostrou-se extremamente adequada nesta aplicação, permitindo a redução da ociosidade do sistema e melhorando sensivelmente o tempo de resposta percebido pelo usuário.

Por fim, acreditamos que este trabalho serviu também para aumentar a divulgação do algoritmo de ray tracing parametrizado na medida em que a interface do sistema RT(p), implementada na forma de applet Java e disponibilizada na Internet, permite que o sistema seja experimentado por uma enorme audiência mundial, conforme confirma a realimentação que este autor tem recebido dos visitantes da home-page²⁰ do projeto RT(p).

7.2 Trabalhos futuros

Vislumbram-se muitos trabalhos decorrentes dos desenvolvimentos iniciais feitos no contexto do projeto RT(p). A seguir, citam-se alguns deles:

7.2.1 Compressão de dados

Um dos principais problemas do ray tracing parametrizado é seu enorme consumo de memória. Várias técnicas propostas no artigo que introduziu-o permitem a redução dos requisitos de memória do mesmo. A maioria não é aplicável no sistema desenvolvido, conforme mencionamos no capítulo 2.

²⁰ <http://www.lsi.usp.br/~rtp>

Apesar de não discutida nesta tese, devido à falta de tempo para a realização de análises e medidas precisas, a implementação que realizamos procura economizar memória através do emprego de tipos de dados com menor precisão numérica que a tradicionalmente utilizada.

Assim, ao invés de armazenar o resultado de produtos escalares em ponto flutuante de simples (*float*: 4 bytes) ou dupla (*double*: 8 bytes) precisão, utilizamos tipos inteiros de apenas 1 (*unsigned char*) ou dois bytes (*short int*) para a mesma função. Os resultados visuais obtidos são praticamente indistinguíveis. Isto é possível levando-se em consideração dois aspectos:

- a) por mais precisos que sejam os cálculos, o resultado final deverá ser armazenado na precisão de 8 bits, correspondente a uma componente de cor da imagem gerada;
- b) geralmente a imagem final sofre quantização de cores para compactação do arquivo (formato GIF (BORN, 1995), por exemplo).

Como trabalhos futuros, pretende-se desenvolver uma análise numérica que demonstre a validade destas simplificações e a realização de medidas do erro médio quadrático apresentado pelas imagens em relação àquelas calculadas com precisão convencional.

7.2.2 Geração de imagens panorâmicas e sistemas de imersão

Na tentativa de aliviar a restrição de posição fixa para o observador, propomos a geração de imagens panorâmicas parametrizadas e seu uso em sistemas onde o usuário pode observar a cena 360° ao seu redor como, por exemplo, QuickTime VR (CHEN, 1995) ou Surround Video.

Este tipo de imagem é facilmente sintetizável (apesar de custosa devido ao grande tamanho) e é ideal para aplicações em arquitetura de interiores onde o ray tracing parametrizado tem um nicho garantido.

Acoplado ao sistema de visualização de imagens panorâmicas, propomos também o uso de sistemas de Realidade Virtual imersivos como HMDs²¹ ou sistemas de projeção semi-esféricos ou do tipo CAVE (CRUZ-NEIRA et alli, 1992), tornando o sistema RT(p) em uma ferramenta de projeto interativo com interface imersiva.

²¹ Head Mounted Display

7.2.3 Extensão das idéias para outros algoritmos

Para que o ray tracing parametrizado estendido que propusemos aqui possa ter precisão realmente adequada para uso profissional em projeto de iluminação é necessário que modele também efeitos de reflexão difusa global (FELDMAN, 1996). Para isto é necessária a extensão do algoritmo para o campo do chamado "ray tracing distribuído" (COOK, PORTER & CARPENTER, 1984). Este algoritmo consiste basicamente em se lançar mais do que um único raio em direção à fonte de luz e nas reflexões de forma a se determinar, de forma estocástica, uma aproximação da contribuição global de iluminação.

Assim, pode-se realizar uma implementação direta do ray tracing parametrizado sobre ray tracing distribuído. O que é necessário é um estudo que otimize o processamento e consumo de memória que, neste caso, certamente seriam proibitivos.

Da mesma forma, pode-se procurar estender as idéias do ray tracing parametrizado para o algoritmo de radiosidade (COHEN & WALLACE, 1993).

7.2.4 Formato de imagem com compressão incremental

Um dos maiores problemas da utilização do sistema RT(p) na Internet hoje é que, apesar de as imagens serem reavaliadas em décimos de segundos, sua transmissão pela rede, devido à banda restrita, toma vários segundos e até minutos, frustrando o usuário.

Como a imagem transmitida ilustra sempre o mesmo cenário, do mesmo ponto de vista e preservando a geometria intacta, acreditamos ser possível o desenvolvimento de um formato de imagem que permita sua compactação baseada na disponibilidade de uma versão anterior, com parâmetros ópticos modificados.

Apesar de tal formato ter pouca utilidade em aplicações gerais, para o caso de ray tracing parametrizado seria de grande importância na aceleração da transmissão das imagens via Internet.

7.2.5 Melhoria da interface

Temos várias idéias para a melhoria da interface com usuário que o sistema RT(p) disponibiliza hoje e que implantaremos em futuro próximo.

A seleção do material/superfície a modificar poderia ser feita simplesmente clicando-se sobre a imagem num ponto onde o mesmo aparece. A posicionamento das

fontes de luz poderia ser feito através de realimentação usando-se a sombra de um elemento qualquer (um poste, por exemplo) como referência.

7.2.6 Parametrização do coeficiente de refração

Acreditamos que o coeficiente de refração possa ser parametrizado dentro de certas limitações. Apesar de este parâmetro não ser importante para a maioria das aplicações, nichos em aplicações educacionais e projeto de sistemas ópticos poderiam fazer uso de tal recurso.

7.2.7 Reaproveitamento de resultados parciais nas equações

É possível se poupar o processamento de milhares de operações armazenando alguns resultados intermediários que, dependendo dos parâmetros alterados, permanecem constantes nas reavaliações. Os resultados destas operações normalmente são descartados. Este tipo de otimização é possível e está em fase de implementação no projeto RT(p).

7.2.8 Implementação paralela e local do servidor de sombras

O servidor de sombras, hoje um módulo sequencial e remoto, poderia ser integrado ao código dos módulos *slave*. Esta implementação traria dois benefícios: o primeiro, a eliminação do tempo de comunicação para transferência dos resultados dos testes de sombra; o segundo, a execução em paralelo da fase de projeção dos pontos de intersecção. Como desvantagem, teríamos a necessidade de armazenamento da geometria da cena em todos os nós. Para cenas grandes, a maior demanda de memória poderia provocar redução de desempenho global.

O aspecto que nos atrai nesta implementação é a possibilidade de utilização de hardware gráfico local em cada nó, fazendo sua utilização em paralelo. Acreditamos que tal implementação tornaria quase desprezível o tempo de computação de sombras num sistema de processamento paralelo distribuído.

Referências Bibliográficas

- ALMEIDA, M. G. ; SANTOS, E. T. ; ZUFFO, J. A. Compactação de Dados em Ray Tracing Parametrizado. **Anais do 3º Simpósio de Iniciação Científica da USP**, p.138 (3.70), São Carlos-SP, 1995.
- ALMEIDA, M. G. ; SANTOS, E. T. RTPWEB: Ciência e Arte na Internet. In: **Anais do 4º Simpósio de Iniciação Científica – USP (EXATAS)**, p.282 (6.100), S.Carlos-SP, 1996.
- ALMEIDA, M. G. ; SANTOS, E. T. Uma Implementação em Máquina Paralela para o Sistema RTpWeb. **Anais do 5º Simpósio de Iniciação Científica da USP**, p.215 (6.30), São Paulo-SP, 1997.
- APPEL, A. Some Techniques for Shading Machine Renderings of Solids. in: **Spring Joint Computer Conference Proceedings**, AFIPS, p. 37-45, 1968.
- ARNOLD, K. ; GOSLING, J., **The Java Programming Language**, 2nd Edition, Addison-Wesley Publishing Co., Reading, Mass., USA, 1998.
- ARVO, J. ; KIRK, D. A Survey of Ray Tracing Acceleration Techniques. In: **An Introduction to Ray Tracing**. ed. Andrew Glassner. Academic Press, London, 1989.
- ASHDOWN, I. ; HAINES, E. Radiosity On-line: A Bibliography. **SIGGRAPH'96 Course Notes #7: Global Illumination in Architecture and Theater**, Appendix D, New Orleans, 1996.
- ASHDOWN, I. Global Illumination and Lighting Design. **SIGGRAPH'96 Course Notes #7: Global Illumination in Architecture and Theater**, New Orleans, 1996.
- BLINN, J. F. ; Simulation of Wrinkled Surfaces. **SIGGRAPH 78 Conference Proceedings**, p.286-92, 1978.
- BORN, G. **The File Formats Handbook**, International Thomson Computer Press, London, 1995.
- BRIÈRE, N. ; POULIN, P. Hierarchical View-dependent Structures for Interactive Scene Manipulation. . **SIGGRAPH 96 Conference Proceedings**, p.83-90, 1996.
- CATMULL, E. **A Subdivision Algorithm for Computer Display of Curved Surfaces**, PhD. Thesis, Report UTEC-CSc-74-133, Computer Science Department, University of Utah, Salt Lake City, UT, 1974.
- CHEN, S. E. QuickTime VR: An Image-Based Approach to Virtual Environment Navigation. **SIGGRAPH 95 Conference Proceedings**, p.29-38, 1995.
- CLUA, E. W. G. ; SANTOS, E. T. ; ZUFFO, J. A. Mapeamentos de Textura e Relevô em Ray Tracing Parametrizado. **Anais do 3º Simpósio de Iniciação Científica da USP**, p.138 (3.69), São Carlos-SP, 1995.

- COHEN, M. F. ; WALLACE, J. R. **Radiosity and Realistic Image Synthesis**. Cambridge, Academic Press, 1993.
- COOK, R. L. Shade Trees. **SIGGRAPH 84 Conference Proceedings**, p.223-31, 1984.
- COOK, R. L.; PORTER, T. ; CARPENTER, L. Distributed Ray Tracing. **Computer Graphics**, v.18, n.3, p.137-145, 1984.
- COPPEN, D. ; SLATER, M. ; DAVISON, A.; HAWES, D., Distributed Frame Buffer for Rapid Dynamic Changes to 3D Scenes, **Proceedings of the Hardware Workshop - Eurographics 92**, 1992.
- CROW, F. C. The Origins of the Teapot. **IEEE Computer Graphics and Applications**, v.7, n.1, 1987.
- CRUZ-NEIRA, C.; SANDIN, D. J.; DeFANTI, T. A., KENYON, R. V.; HART, J. C. The CAVE: Audio Visual Experience Automatic Virtual Environment. **Communications of the ACM**, v.35, n.6, p.64-72, 1992.
- ELLIOTT, S. D. ; MILLER, P. L. **Inside 3D Studio Release 4**. New Riders Publishing, Indianapolis, 1995.
- FELDMAN, S. Photorealism in Architecture and Entertainment. **SIGGRAPH'96 Course Notes #7: Global Illumination in Architecture and Theater**, New Orleans, 1996.
- FOLEY, J. D. ; DAM, A. V. ; FEINER, S. K. ; HUGHES, J. F. **Computer Graphics - Principles and Practice**. 2nd. Ed. Addison-Wesley Publishing Company, 1990.
- FRAGA, T. R. ; SANTOS, E. T. Mostra de Arte SIBGRAPI'96.
- GLASSNER, A. S., ed. **An Introduction to Ray Tracing**. Londres, Academic Press, 1989.
- GOUVEIA, F. A. ; SANTOS, E. T. Um Tutorial On-line para o Ensino de Modelos de Iluminação. **Anais do 5º Simpósio de Iniciação Científica da USP**, p.215 (6.29), São Paulo-SP, 1997.
- HAINES, E. A. ; GREENBERG, D. P. The Light Buffer: A Ray Tracer Shadow Testing Accelerator. **IEEE Computer Graphics and Applications**. v.6, n.9, p.6-16, 1986.
- HAINES, E. A Proposal for Standard Graphics Environments. **IEEE Computer Graphics and Applications**, v.7, n.11, p.3-5, 1987.
- HALL, R. A. ; GREENBERG, D. P. A Testbed for Realistic Image Synthesis. **IEEE Computer Graphics and Applications**, v.3, n.8, p.10-20, 1983.

- JANSEN, F.W., A Multi-Processor Workstation with a Logic-enhanced Distributed Frame Buffer, **Advances in Computer Graphics Hardware II**, Eurographics Seminars, Springer-Verlag, Berlin, p. 229-238, 1988.
- KOFUJI, S. T. **Considerações de Projeto e Análise do Spade: um Multiprocessador de Larga Escala Baseado no Padrão ANSI/IEEE-SCI**. Tese de Doutorado. São Paulo, EPUSP, 1994.
- KUCHKUDA, R. An Introduction to Ray Tracing . In: Earnshaw, R. A. (ed.) **Theoretical Foundations of Computer Graphics and CAD**, Springer-Verlag Berlin Heidelberg, NATO ASI Series, vol. F40, p.1039-1060, 1988.
- LEITÃO, R. B. V.; FAGUNDES, R. P.; AYRES, F. J.; SANTOS, E. T.; ZUFFO, M. K. Um Algoritmo Rápido para Iluminação Volumétrica. in: **Anais do Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens**, Caxambu-MG, p.9-14, 1996.
- LEITÃO, R. B. V.; AYRES, F. J.; SANTOS, E. T.; ZUFFO, M. K. Fast Volume Shading Through Lookup Tables (a ser publicado).
- MARKS, J. et alli Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. **SIGGRAPH 97 Conference Proceedings**, p.389-400, 1997.
- MARTINS, F. F. D. ; SANTOS, E. T. Ray Tracing Parametrizado. **Anais do XII Congresso de Iniciação Científica e Tecnológica em Engenharia - CICTE'93**, p.159, S. Carlos - SP, dezembro 1993.
- MARTINS, F. F. D. ; SANTOS, E. T. Rtp - Uma Implementação de Ray Tracing Parametrizado. **Anais do Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens - SIBGRAPI'94 - Comunicações**, p.31-4, Curitiba - PR, novembro 1994a.
- MARTINS, F. F. D. ; SANTOS, E. T. Hardware Light Buffering. **Anais do 2º Simpósio de Iniciação Científica da USP**, p.E-06, São Paulo, 1994b.
- MAY, D. ; SHEPHERD, R. Communicating Process Computers. **Inmos Technical Note 22**, Bristol, Inmos Ltd., 1987.
- McDONALD, E. A Video Controller and Distributed Frame Buffer for the J-Machine. S.M. Dissertation Project. MIT, June, 1993.
- MICROSOFT **SOFTIMAGE | SDK - mental ray Programmer's Reference Guide Version 1.7 for IRIX and Windows NT**, 1997.
- MOURÃO, L. ; MOREIRA, J. C. V. ; DEFENDI, M. C. C. ; CENTOLA, N. O Mundo Maravilhoso da Computação Gráfica. **PC MAGAZINE BRASIL**, v.6, n.10, Outubro 1996.
- MURAKAMI, K. ; HIROTA, K. Incremental Ray Tracing. **Proceedings of the Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics**. Rennes, p.15-29, June 1990.

- NEIDER, J. ; DAVIS, T. ; WOO, M. OpenGL Programming Guide. Addison-Wesley, Reading - Mass., 1993.
- NOMA, T. ; SUMI, K. Shadows on Bump-Mapped Surfaces in Ray tracing. **The Visual Computer**, v.10, p.330-6, 1994.
- POTMESIL, M. ; McMILLAN, L. ; HOFFERT, E. M. ; INMAN, J. F.; FARAH, R. L. ; HOWARD, M. A Parallel Image Computer with a Distributed Frame Buffer: System Architecture and Programming. **Proceedings of Eurographics '89**, W. Hansmann, F. R. A. Hopgood and W. Strasser (eds.), p. 197-208, 549. 1989.
- RAMOS, T. O. Aplicação Multimídia é destaque no sistema. **Folha de São Paulo, Caderno de Informática**, p. G-14, 10/06/1996.
- SANTOS, E. T. **Avaliação do Algoritmo de Ray Tracing em Multicomputadores**. Dissertação de Mestrado. São Paulo-SP, EPUSP, 1994a.
- SANTOS, E. T. Movimentação de Fontes de Luz em Ray Tracing Parametrizado. **Anais do Congresso Internacional de Computação Gráfica - CICOMGRAF'94**, São Paulo-SP, 1994b.
- SANTOS E. T. Changing Some Geometric Parameters in Parameterized Ray Tracing. **Anais do VII SIBGRAPI - Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens**, Curitiba-PR, 1994c.
- SCHIMITT, A. ; MÜLLER, H. ; LEISTER, W. Ray Tracing Algorithms - Theory and Practice. In: Earnshaw, R. A. (ed.) **Theoretical Foundations of Computer Graphics and CAD**, Springer-Verlag Berlin Heidelberg, NATO ASI Series, vol. F40, p.997-1030, 1988.
- SÉQUIN, C. ; SMYRL, E. Parameterized Ray Tracing. **Computer Graphics**, v.23, n.3, p.307-314, 1989.
- SHAKESPEARE, R. Applications of Theatrical Lighting Design. **SIGGRAPH'96 Course Notes #7: Global Illumination in Architecture and Theater**, New Orleans, 1996.
- SILLION, F. ; PUECH, C. **Radiosity and Global Illumination**. San Francisco, Morgan Kaufmann Publishers, 1994.
- SPEER, L. R. An Updated Cross-Indexed Guide to the Ray Tracing Literature. **Computer Graphics**, v.26, n.1, p. 41-72, 1992
- VANDEWETTERING, M. T. **The MTV Ray Tracer**. Documentação do programa. Disponível na Internet em cs.uoregon.edu por ftp anônimo, 1988.
- WARD, G. J. The RADIANCE Lighting Simulation and Rendering System. **Computer Graphics Annual Conference Series**, ACM, p. 459-72, 1994.
- WATKINS, C. ; COY, S. ; FINLAY, M. **Photorealism and Ray Tracing in C**. M & T Books, 1992.

- WATT, A. ; WATT, M. **Advanced Animation and Rendering Techniques**. Addison-Wesley / ACM Press, New York, 1992.
- WEGHORST, H. ; HOOPER, G. ; GREENBERG, D. P. Improved Computational Methods for Ray Tracing. **ACM Transactions on Graphics**, v.3, n.1, p.52-69, 1984.
- WEI, B. ; STOLL, G. ; CLARK, D. W. ; FELTEN, E. W., KAI Li RAIN: Supporting Parallel Graphics on Paragon Multicomputers. **The 13th Annual Conference - Intel Supercomputing Users Group**, Albuquerque, June, 1997.
- WHITTED, T. An Improved Illumination Model for Shaded Display. **Comm. of the ACM**, v. 23, n.6, p. 343-349, 1980.
- WILLIAMS, L. Casting Curved Shadows on Curved Surfaces. **Computer Graphics**, v. 12, n. 3, p.270-4, 1978.

Anexos

Anexo A: Um “shader” RT(p) para o Softimage/mental ray

A.1 Introdução

Softimage Creative Environment é um pacote integrado para modelamento, animação e síntese de imagens. O mental ray é um ray tracer independente que pode ser usado de forma integrada ao ambiente Softimage.

Somente no ano de 1997, os seguintes filmes fizeram uso intensivo do software Softimage: "O quinto elemento", "Men In Black", "Jurassic Park: O mundo perdido", "Velocidade Máxima 2", "Spawn", "Contacto", "Um lobisomem americano em Paris", "Alien: A ressurreição", "Anastásia", "Tropas Estelares", "Titanic", "Flubber", "O amanhã nunca morre", entre outros, demonstrando a alta qualidade e aceitação deste sistema.

Mental ray é um ray tracer de uso genérico capaz de criar imagens de excepcional qualidade e obter alto desempenho através da exploração de paralelismo tanto em máquinas multiprocessadoras quanto em redes de estações de trabalho (MICROSOFT, 1997).

Este software usa técnicas avançadas de aceleração tais como algoritmo de varredura de linhas (scanline) para determinação primária de superfícies visíveis, "classificação de raios" para raios secundários e classificação separada para acelerar raios de sombra, além de suportar o algoritmo BSP (Binary Space Partitioning).

Existem versões de ambos os programas para os sistemas Unix (IRIX - Plataforma MIPS-Silicon Graphics) e Windows NT 4.0 (Plataformas Intel).

A funcionalidade do mental ray pode ser estendida através da ligação (linking) dinâmica de subrotinas escritas em linguagem "C" denominadas "shaders". Os "shaders" podem ser usados para a criação de texturas procedurais, materiais, efeitos atmosféricos e de volume, lentes e fontes de luz.

Desenvolvemos um "shader" para permitir a utilização do mental ray em substituição ao MTV ray tracer que utilizávamos nas primeiras versões do sistema RT(p).

As vantagens de tal substituição são:

- a) possibilidade de utilizar-se o versátil modelador do pacote Softimage;

- b) possibilidade de importar modelos em outros formatos (dxf, 3ds, etc.). Antes tinha-se que restringir-se ao formato nff (Neutral File Format) que é muito limitado;
- c) possibilidade de utilizar-se modelos geométricos sofisticados, como superfícies NURB por exemplo, sem a necessidade de desenvolver o código de cálculo de intersecção correspondente;
- d) isolamento completo entre o código de traçado de raios e intersecções, (implementado no mental ray em si) do código correspondente ao armazenamento da estrutura de dados das árvores para pós-processamento no RT(p) (implementado no "shader"), conseqüentemente dispensando a manutenção do primeiro.
- e) rapidez de geração das estruturas de dados utilizadas pelo sistema RT(p) já que o mental ray é um ray tracer bastante otimizado e paralelo.

As desvantagens que percebemos foi a maior dificuldade de desenvolvimento de código porque não temos o controle do código fonte do mental ray e sua documentação é bastante pobre. O resultado foi uma sensível diminuição de nosso desempenho de programação. Com o domínio do programa mental ray e formatos associados acreditamos ter recuperado a facilidade anterior.

A.2 Implementação

A função do "shader" gera_rtp é gerar um arquivo no formato ".rtp" que contém, além das informações geométricas da cena, as árvores de raio geradas no processo de traçagem dos raios pelo cenário. Estas informações são usadas posteriormente para a reavaliação rápida da imagem.

A figura A.1 ilustra a arquitetura atual para geração do arquivo ".rtp", necessário ao sistema RT(p).

A implementação desenvolvida é compatível com o processamento paralelo em "threads" realizado automaticamente pelo mental ray, alcançando máximo desempenho.

FIG. A.0.1 - Arquitetura de pré-processamento do sistema RT(p)

Devido ao fato de as arquiteturas onde são executados o mental ray e o sistema RT(p) serem geralmente diferentes (Linux/Intel x IRIX/MIPS) tendo ordenação de bytes na palavra incompatíveis, a gravação do arquivo é sempre feita na arquitetura destino, onde será executado o RT(p). Assim, o "shader" gera_rtp, em acordo com a arquitetura cliente-servidor do restante do sistema, comunica-se via sockets com outro componente do sistema (srtp2rtp), responsável pela gravação do arquivo no formato final. Esta organização tem dois objetivos:

- a) gravar o arquivo .rtp com a ordenação de bytes adequada;
- b) gravar os dados de cada nó em ordem, já que o processamento dos píxeis pelo mental ray ocorre em qualquer ordem devido ao processamento paralelo ("threads").

O formato de entrada do mental ray é um arquivo em formato ".mi". Atualmente este arquivo precisa ser pré-processado (utilitário conv) para que os parâmetros sejam adequados ao "shader" gera_rtp. Além disso, um arquivo auxiliar (mental_rtp.conf) também é gerado simultaneamente para conter outros dados necessários ao shader. Está em desenvolvimento atualmente uma interface a ser incorporada ao pacote Softimage que permitirá que o usuário configure os parâmetros desejados dentro do próprio ambiente Softimage disparando também a execução do mental ray de forma a dispensar-se a geração explícita do arquivo ".mi" e seu pré-processamento. Esta interface substituirá todos os componentes dentro da região marcada em linha tracejada na figura A.1.

Anexo B: StereoRTP – Uma aplicação em estereografia

B.1 Introdução

StereoRTP é uma versão do sistema RT(p) desenvolvida para permitir a visualização em estéreo das imagens geradas pelo sistema. Esta versão foi desenvolvida com a finalidade de demonstrar a capacidade de processamento dos protótipos da família SPADE, para a visualização estéreo de "esculturas virtuais", e para demonstrar o sistema RT(p). Pretende-se desenvolver no futuro uma aplicação para projeto de interiores que utilize o sistema RT(p) e possibilite visualização em estéreo.

O sistema StereoRTP foi apresentado no estande da Universidade de São Paulo no evento "SC97: High Performance Networking and Computing", organizado pelo ACM SIGARCH e IEEE Computer Society em San Jose, CA - USA em novembro de 1997 (fig. B.1).

FIG. B.0.1 – Apresentação do sistema StereoRTP no SC'97

A técnica de estereoscopia utilizada é a projeção, em tela metalizada, de duas imagens projetadas com polarização da luz em "V" (direções de polarização rotacionadas de 90°). As duas imagens se superpõem na tela. O usuário deve usar óculos com lentes polarizadoras nas mesmas direções de polarização dos projetores de forma que enxergue apenas uma das imagens com cada olho.

A figura B.3 mostra a configuração completa do sistema apresentado. As cenas apresentadas foram criadas pela artista Tania Fraga.

B.2 Implementação

A arquitetura de software do sistema implementado basicamente duplica aquela do sistema RT(p), com um sistema individual para o recálculo da imagem de cada olho (fig. B.2). Devido à limitações de segurança nos applets Java que impede a conexão com computadores outros que não o que forneceu o próprio applet, um "middleware" (stereo_mdw) foi desenvolvido para fazer a comunicação entre o par de sistemas e o applet.

FIG. B.0.2 - Arquitetura de software do sistema StereoRTP

A alocação dos componentes de software nos nós de processamento é mostrada na tabela B.1.

Nó	Componentes de software
0 ²²	stereo_mdw, server
1	master_l
2	master_r
3	slave_l e slave_r
4	slave_l e slave_r
5	slave_l e slave_r
6	slave_l e slave_r
7	slave_l e slave_r
8	slave_l e slave_r

TAB. B.0.1 - Alocação de componentes de software no sistema Aquila

O micro-servidor web (server) e o middleware (stereo_mdw) são executados na estação de acesso conectada ao aglomerado por rede de alta velocidade. Este "middleware" tem a única finalidade de comunicar-se com o applet para receber os

²² Estação de acesso

dados da reavaliação e enviar duas cópias idênticas dos mesmos ao master_l e master_r.

Os programas master e slave são exatamente iguais ao normalmente usados no sistema RT(p), sendo o master em versão local (mostra imagem na tela ao invés de enviá-la ao cliente).

Os nós onde são executados os componentes "master" possuem uma placa gráfica para exibir, em tela cheia, o resultado da reavaliação. A saída de cada placa gráfica é conectada a um projetor com filtro polarizador sobre a lente.

Na configuração utilizada, obteve-se tempo de reavaliação de aproximadamente 0,5 segundo para as duas imagens simultaneamente, em resolução de 640x480 píxeis e 24 bits de cor.

Interactive Stereo Image Resynthesis – SC'97

FIG. B.0.3 - Cartaz explicativo do sistema StereoRTP apresentado no SC'97

Anexo C: Dados coletados

Os valores em negrito são a média dos tempos das várias execuções de um ensaio

ENSAIO 1 - Tempos de processamento para recálculo de sombra (Cena Bule)

Cena: Teapot100 (3 materiais)		
Resolucao: 100x100		
N. hit points: 12339		
Msg. result.: 1543 bytes		
Z-Buffer		Tp
10		0.081797
10		0.082101
10		0.081375
10		0.081909
10		0.082140
		0.081864

32		0.086765
32		0.087228
32		0.086166
32		0.0872
32		0.087269
		0.086926

50		0.091174
50		0.090348
50		0.089102
50		0.090699
50		0.090279
		0.09032

100		0.094919
100		0.09674
100		0.096744
100		0.096617
100		0.096123
		0.096229

320		0.177597
320		0.177182
320		0.178436
320		0.178469
320		0.177237
		0.177784

Cena: Teapot300 (3 materiais)		
Resolucao: 300x300		
N. hit points: 61052		
Msg. result.: 7632 bytes		
Z-Buffer		Tp
10		0.229403
10		0.228336
10		0.228606
10		0.228312
10		0.228446
		0.228621

32		0.235316
32		0.233682
32		0.233856
32		0.233738
32		0.233681
		0.234055

50		0.238925
50		0.236435
50		0.235644
50		0.235618
50		0.235911
		0.236507

100		0.243862
100		0.244022
100		0.244758
100		0.243956
100		0.244072
		0.244134

320		0.336831
320		0.343035
320		0.337278
320		0.337828
320		0.338016
		0.338598

500		0.295198
500		0.295233
500		0.298271
500		0.294333
500		0.295988
		0.295805

1000		0.877558
1000		0.877265
1000		0.877478
1000		0.877318
1000		0.872718
		0.876467

Cena: Teapot500 (3 materiais)		
Resolucao: 500x500		
N. hit points: 309294		
Msg. result.: 38662 bytes		

Z-Buffer		TP
10		1.115837
10		1.115993
10		1.116235
10		1.116044
10		1.115969
		1.116016

32		1.123871
32		1.124436
32		1.121724
32		1.123725
32		1.124116
		1.123574

50		1.132882
50		1.128202
50		1.138577
50		1.128132
50		1.130233
		1.131605

100		1.13569
100		1.137044
100		1.138784
100		1.135945
100		1.134333
		1.136359

320		1.168888
320		1.169345

500		0.462034
500		0.458386
500		0.457678
500		0.458067
500		0.459296
		0.459092

1000		1.069929
1000		1.111668
1000		1.045503
1000		1.053974
1000		1.066453
		1.069505

Cena: Teapot1000 (3 materiais)		
Resolucao: 1000x1000		
N. hit points: 1237650		
Msg. result.: 154707 bytes		

Z-Buffer		TP
10		4.040023
10		4.039530
10		4.038450
10		4.040118
		4.039530

32		4.043230
32		4.042330
32		4.041234
		4.042265

50		4.054375
50		4.058234
50		4.049763
50		4.052145
50		4.053629

100		4.087726
100		4.085476
100		4.089923
100		4.080904
100		4.09848
		4.088696

320		4.157552
320		4.155185
320		4.159006
320		4.167382
320		4.168519
		4.161529

320		1.16916
320		1.169435
320		1.168649
		1.169095

500		1.290413
500		1.290016
500		1.290571
500		1.29224
500		1.297096
		1.292067

1000		1.881846
1000		1.879315
1000		1.87906
1000		1.881757
1000		1.879609
		1.880317

500		4.382127
500		4.38713
500		4.383985
500		4.382227
		4.383867

1000		4.989141
1000		4.999903
1000		4.997439
1000		5.000755
1000		4.982603
		4.993968

ENSAIO 2 - Tempos de processamento para recálculo de sombra (Cena Interior)

INTERIOR T1		
hit points: 268599		
bytes sent: 33575		
Resolução Z-Buffer		
300x300	100x100	500x500
4.389029	3.127055	8.591487
4.367836	3.128998	8.563509
4.383463	3.117026	8.524768
4.373842	3.120148	8.674993
4.388707	3.120572	8.583864
4.380575	3.122760	8.587724
INTERIOR T2		
hit points: 585636		
bytes sent: 73205		
Resolução Z-Buffer		
300x300	100x100	500x500
5.585838	4.162241	9.848304
5.573654	4.151174	9.660178
5.606956	4.146802	9.727898
5.683471	4.161431	9.691256
5.562210	4.150295	9.665894
5.602426	4.154389	9.718706

INTERIOR T3		
hit points:1132847		
bytes sent:141606		
Resolução Z-Buffer		
300x300	100x100	500x500
7.702347	5.811110	11.794658
7.713204	5.825657	11.719885
7.765142	5.802970	11.776017
7.789941	5.811771	11.805785
7.721743	5.820547	11.861578
7.738475	5.814411	11.791585
INTERIOR T4		
hit points: 2040931		
bytes sent: 255117		
Resolução Z-Buffer		
300x300	100x100	500x500
11.200033	9.204221	15.005109
11.212990	9.244187	15.095510
11.194262	9.215194	15.242654
11.209841	9.224026	15.159636
11.312769	9.223062	15.049612
11.225979	9.222138	15.110504

ENSAIO 3 - Tempos de comunicação sserver / master (Cena Bule)

TEAPOT100 1543 bytes		
<i>Tproc</i>	<i>Ttot</i>	
0.413559	0.635879	
0.366121	0.574485	
0.516143	0.723108	
0.343831	0.55031	
0.375324	0.583909	
0.402996	0.613538	
TEAPOT300 7632 bytes		
<i>Tproc</i>	<i>Ttot</i>	
0.485434	1.085472	
0.537431	0.613801	
0.518053	1.094211	
0.505461	0.542639	
0.486631	1.075123	
0.531946	1.247778	
0.503105	0.525566	
0.509723	0.883513	

TEAPOT500 38662 bytes		
<i>Tproc</i>	<i>Ttot</i>	
1.592934	2.046617	
1.610694	2.079799	
1.660347	2.215153	
1.53642	2.013103	
1.526981	1.990522	
1.585475	2.069039	
TEAPOT1000 154707 bytes		
<i>Tproc</i>	<i>Ttot</i>	
5.448074	8.303679	
5.383328	8.188027	
5.338804	8.152715	
5.753167	8.757768	
5.489823	8.337961	
5.482639	8.34803	

ENSAIO 4 - Acelerador Gráfico (Cena: Bule300)

<i>master@polaris slave@polaris sserver@ravel, display@handel</i>
2.820415
2.52538
2.521419
2.667104
2.722573
2.6513782
<i>master@polaris slave@polaris sserver@ravel, display@mozart</i>
3.061792
3.048048
2.908729
2.883445
2.960893
2.9725814
<i>master@polaris slave@polaris sserver@handel, display@mozart</i>
2.377717
2.332367
2.282741
2.343135
2.416028
2.3503976
<i>master@polaris slave@polaris sserver@mozart, display@mozart</i>
0.956741
0.967145
0.958807
0.973905
0.954936
0.9623068
<i>master@polaris slave@polaris sserver@handel, display@handel</i>
0.188186
0.185881
0.186521
0.185324
0.183061
0.1857946

ENSAIO 5 - Processamento paralelo (Cena: Bule500)

TEAPOT500		
640x480	320x480	160x480
----- 1 processador -----		
1.269613	1.040994	0.965078
1.280765	1.044646	0.959337
1.355813	1.070961	0.963434
1.366269	1.035369	0.980342
1.304856	1.027346	0.974651
1.315463	1.043863	0.968568
----- 2 processadores -----		
1.103955	0.658648	0.515758
1.106008	0.657286	0.510584
1.083272	0.643958	0.521936
1.125253	0.616542	0.500731
1.081398	0.666824	0.500559
1.099977	0.648652	0.509914
----- 3 processadores -----		
1.054809	0.538526	0.363627
1.111619	0.476041	0.378936
1.113394	0.565141	0.379687
1.04296	0.57236	0.367854
1.06426	0.518732	0.371985
1.077408	0.53416	0.372418

TEAPOT500		
640x480	320x480	160x480
----- 4 processadores -----		
1.081283	0.543911	0.336446
1.036844	0.558	0.317174
1.025991	0.562577	0.295523
0.952399	0.510936	0.304325
1.00631	0.526572	0.311011
1.020565	0.540399	0.312896
----- 5 processadores -----		
1.067775	0.570944	0.256499
1.025805	0.495988	0.268152
1.101014	0.534572	0.282208
1.114968	0.592774	0.304806
1.080781	0.592774	0.266553
1.078069	0.55741	0.275644
----- 6 processadores -----		
1.025695	0.564698	0.257417
1.089756	0.569433	0.260173
1.152768	0.56226	0.299023
1.124641	0.543943	0.282423
1.000672	0.585581	0.276003
1.078706	0.565183	0.275008
----- 7 processadores -----		
1.015772	0.533627	0.27186
1.075041	0.570755	0.263694
1.098473	0.557377	0.271373
1.113212	0.537177	0.303525
1.106333	0.534578	0.308115
1.081766	0.546703	0.283713
----- 8 processadores -----		
1.045045	0.514122	0.274979
1.079519	0.493621	0.274303
1.115164	0.572001	0.276989
1.126478	0.549226	0.336578
1.148789	0.573961	0.292543
1.102999	0.540586	0.291078
----- 16 processadores / 8 nós -----		
1.145372	0.562762	0.304902
1.128026	0.754516	0.325864
1.268713	0.566963	0.312809
1.232877	0.612334	0.325952
1.176623	1.473.862	0.333139
1.190322	294772.9	0.320533

ENSAIO 6 - Processamento paralelo (Cena: Bule1000x250)

TEAPOT 1000x250		
640x480	320x480	160x480
----- 1 processador -----		
1.038856	0.964906	0.949503
1.034559	0.955567	0.944447
1.025496	0.977709	0.936215
1.044645	0.955833	0.931625
1.014818	0.961093	0.935624
1.023562	0.955263	0.945467
1.035344	0.959364	0.946152
1.026793	0.95477	0.935416
1.027748	0.957273	0.939125
1.036668	0.955103	0.937618
1.030849	0.959688	0.940119
----- 2 processadores -----		
0.637761	0.514358	0.496615
0.615336	0.515396	0.485917
0.648814	0.50603	0.500667
0.675079	0.504395	0.494458
0.623192	0.501466	0.491335
0.619129	0.507848	0.496422
0.570222	0.509848	0.501413
0.660898	0.543762	0.494277
0.594767	0.50864	0.49758
0.619991	0.5128	0.495008
0.626519	0.512454	0.495369
----- 3 processadores -----		
0.594896	0.376754	0.352456
0.563218	0.376126	0.355845
0.576293	0.39768	0.347722
0.647839	0.377791	0.35649
0.561359	0.377732	0.351671
0.570503	0.378457	0.353381
0.571613	0.378806	0.349554
0.602629	0.397189	0.354791
0.609109	0.377258	0.341154
0.565824	0.361843	0.336377
0.586328	0.379964	0.349944

----- 4 processadores -----		
0.63349	0.338967	0.270745
0.588934	0.326762	0.266973
0.620037	0.335295	0.276065
0.583517	0.324312	0.267965
0.590303	0.337813	0.276986
0.603256	0.33263	0.271747
----- 5 processadores -----		
0.576199	0.340473	0.221665
0.563238	0.370529	0.230095
0.612801	0.347451	0.226422
0.682434	0.364492	0.231637
0.595611	0.336676	0.226883
0.606057	0.351924	0.22734
----- 6 processadores -----		
0.658134	0.361097	0.208366
0.64954	0.349995	0.200769
0.681339	0.355889	0.195985
0.671976	0.355025	0.205971
0.601508	0.347729	0.207782
0.652499	0.353947	0.203775
----- 7 processadores -----		
0.716129	0.363489	0.190791
0.72815	0.361274	0.19004
0.738144	0.357078	0.184787
0.729644	0.362475	0.186427
0.680683	0.372645	0.188871
0.71855	0.363392	0.188183
----- 8 processadores -----		
0.608536	0.365456	0.188207
0.621618	0.444911	0.190556
0.721879	0.437998	0.189939
0.714778	0.429975	0.187473
0.681245	0.4101	0.187477
0.669611	0.417688	0.18873
----- 16 proc. (8 nos) -----		
0.623489	0.417181	0.212055
0.744354	0.435379	0.204228
0.814321	0.486478	0.191036
0.657167	0.380872	0.196643
0.708436	0.477197	0.199259
0.709553	0.439421	0.200644

ENSAIO 7 - Processamento paralelo (Cena: Bule250x1000)

TEAPOT250x1000		
640x480	320x480	160x480
----- 1 proc. -----		
1.943702	1.30005	1.037943
2.082943	1.321191	1.024235
1.993306	1.305896	1.042263
1.981802	1.305382	1.025056
1.957753	1.341771	1.044057
1.991901	1.314858	1.034711
----- 2 proc. -----		
2.07292	1.077173	0.640661
2.196063	1.091524	0.603881
2.112652	1.113491	0.657245
2.150647	1.080439	0.652373
2.167464	1.075419	0.631038
2.139949	1.087609	0.63704
----- 3 proc. -----		
2.063146	1.052705	0.586397
2.164141	1.011983	0.501
2.025477	1.048944	0.510683
2.124619	1.091566	0.496185
2.096191	1.07455	0.564464
2.094715	1.05595	0.531746
----- 4 proc. -----		
2.031793	1.066263	0.548309
2.086831	1.066878	0.515385
2.096268	1.071456	0.591627
2.1765	1.077437	0.517609
2.16211	1.035783	0.53713
2.1107	1.063563	0.542012

----- 5 proc. -----		
2.107737	0.989566	0.545112
2.045833	1.063264	0.530459
2.135225	1.065915	0.521112
2.094994	1.069038	0.501788
2.064409	1.083669	0.556912
2.08964	1.05429	0.531077
----- 6 proc. -----		
2.114203	1.116208	0.536381
2.122849	1.114764	0.584711
2.044958	1.124787	0.667756
2.166098	1.026574	0.638251
2.021612	1.044114	0.687631
2.093944	1.085289	0.622946
----- 7 proc. -----		
2.075983	1.069231	0.576657
2.051174	1.035287	0.590942
2.144142	1.086637	0.548022
1.976205	1.09235	0.53749
2.143998	1.020238	0.526878
2.0783	1.060749	0.555998
----- 8 proc. -----		
2.137426	1.000697	0.538712
2.200251	0.955477	0.529391
2.126885	1.103899	0.509083
2.19213	1.015886	0.542763
2.104935	1.102928	0.54124
2.152325	1.035777	0.532238

ENSAIO 8 - Processamento paralelo (Cena: InteriorT4)

INTERIOR_T4		
640x480	320x480	160x480
1 proc		
4.631028	4.415811	4.385578
4.506701	4.417256	4.385568
4.518529	4.418995	4.366442
4.507763	4.417042	4.384412
4.507011	4.416428	4.416356
4.50779	4.41763	4.376507
4.494946	4.426786	4.375126
4.50625	4.423849	4.374227
4.506544	4.43729	4.391245
4.507494	4.424223	4.382202
4.519406	4.421531	4.383766
2 proc		
2.269558	2.2637	2.198385
2.270661	2.236577	2.196685
2.269776	2.267731	2.213551
2.269681	2.234631	2.217379
2.270734	2.255496	2.213838
2.299646	2.235713	2.195336
2.287347	2.237795	2.215857
2.286404	2.234466	2.218155
2.276548	2.235937	2.209568
2.294992	2.243341	2.216059
2.279535	2.244539	2.209481
3 proc		
1.609557	1.540247	1.517241
1.665643	1.537711	1.517675
1.588361	1.547116	1.517176
1.577094	1.545814	1.515364
1.588353	1.541621	1.517815
1.568854	1.536691	1.515976
1.567514	1.547623	1.517415
1.584255	1.546169	1.515784
1.586177	1.536921	1.514731
1.586436	1.54903	1.525784
1.592224	1.542894	1.517496
4 proc		
1.186137	1.165531	1.144207
1.188097	1.165566	1.145871
1.189128	1.166742	1.155889
1.192358	1.1652	1.155427
1.187856	1.165611	1.155609
1.186348	1.166734	1.155167
1.188677	1.167037	1.156208

INTERIOR_T4		
640x480	320x480	160x480
5 proc		
0.976931	0.950356	0.935736
0.977539	0.949598	0.936317
0.976528	0.950321	0.944852
0.977649	0.947602	0.940064
0.977798	0.966249	0.936458
0.969344	0.947914	0.935917
0.969982	0.960503	0.935645
0.966486	0.952489	0.936238
0.969507	0.952957	0.9416
0.976221	0.94767	0.936011
0.973799	0.952566	0.937884
6 proc		
0.820903	0.794443	0.78679
0.818809	0.794056	0.785091
0.812649	0.794543	0.785336
0.82294	0.802793	0.787475
0.81642	0.809814	0.785365
0.820671	0.798912	0.785548
0.812233	0.801356	0.784229
0.819287	0.796587	0.787308
0.809425	0.794804	0.783993
0.817875	0.79536	0.791631
0.817121	0.798267	0.786277
7 proc		
0.755456	0.684609	0.685998
0.740595	0.694925	0.685328
0.774241	0.687972	0.68667
0.736251	0.69587	0.682271
0.747448	0.689109	0.683845
0.746549	0.689151	0.683413
0.744818	0.689748	0.685033
0.740371	0.686978	0.683173
0.760146	0.687551	0.681086
0.738815	0.687482	0.684659
0.748469	0.68934	0.684148
8 proc		
0.75365	0.61646	0.599911
0.739967	0.626974	0.59911
0.720298	0.635369	0.605516
0.73714	0.632251	0.606568
0.725012	0.63976	0.599423
0.747562	0.611064	0.604555
0.740029	0.617952	0.599555

1.187847	1.163639	1.156428
1.18881	1.167004	1.155266
1.186449	1.165783	1.155846
1.188171	1.165885	1.153592

0.73648	0.626636	0.605732
0.736578	0.615674	0.605614
0.758945	0.613929	0.610334
0.739566	0.623607	0.603632

ENSAIO 9 - Tempo de reavaliação completa (Cena: Interior)

<i>Tempo de reavaliação completa</i>			
<i>InteriorT1</i>	<i>InteriorT2</i>	<i>InteriorT3</i>	<i>InteriorT4</i>
2.999805	3.314205	3.654983	4.32915
3.011555	3.310811	3.638352	4.347151
2.988433	3.30543	3.63903	4.347429
3.012773	3.314368	3.637324	4.33077
2.996663	3.319935	3.637181	4.334143
3.001846	3.31295	3.641374	4.337729

ENSAIO 10 - Tempo de reavaliação incremental (Cena: Interior)

InteriorT1						
mat1:	2.101616	2.109619	2.097266	2.104509	2.095666	2.101735
mat2:	0.693041	0.683521	0.677796	0.681673	0.689798	0.685166
mat3:	0.698047	0.691448	0.688295	0.683305	0.691892	0.690597
mat4:	1.883589	1.889529	1.888922	1.889971	1.884249	1.887252
mat5:	0.681518	0.687869	0.696607	0.684268	0.689571	0.687967
mat6:	1.298202	1.250106	1.247324	1.284834	1.245165	1.265126
mat7:	1.764995	1.761893	1.76028	1.759537	1.761319	1.761605
mat8:	0.697075	0.705624	0.69805	0.681022	0.690432	0.694441
mat9:	0.823345	0.819476	0.816893	0.820464	0.813432	0.818722
mat10:	0.683135	0.693937	0.697744	0.680859	0.688217	0.688778
mat11:	0.697003	0.686011	0.676562	0.704042	0.696918	0.692107

InteriorT2						
mat1:	3.064755	3.07245	3.05209	3.054448	3.055416	3.059832
mat2:	1.044576	1.051292	1.038844	1.045135	1.053886	1.046747
mat3:	0.700866	0.715381	0.688353	0.681805	0.685791	0.694439
mat4:	2.377037	2.382265	2.394223	2.396324	2.370609	2.384092
mat5:	0.696463	0.673022	0.682624	0.682451	0.680949	0.683102
mat6:	1.498987	1.466389	1.480009	1.446866	1.469919	1.472434
mat7:	2.468298	2.47434	2.471694	2.469127	2.464847	2.469661
mat8:	0.699405	0.684236	0.685203	0.697691	0.700309	0.693369
mat9:	0.895274	0.919569	0.94463	0.886813	0.901685	0.909594
mat10:	0.682242	0.689916	0.691465	0.681091	0.689226	0.686788
mat11:	0.688617	0.671418	0.69103	0.721277	0.676628	0.689794

InteriorT3						
mat1:	3.516692	3.515635	3.513207	3.52014	3.532389	3.519613
mat2:	1.586741	1.570443	1.574974	1.57259	1.579159	1.576781
mat3:	0.685056	0.682835	0.683025	0.673972	0.673706	0.679719
mat4:	2.783552	2.769963	2.783467	2.771113	2.772156	2.77605
mat5:	0.67004	0.666602	0.677032	0.665998	0.694153	0.674765
mat6:	1.665475	1.662113	1.660123	1.645226	1.641088	1.654805
mat7:	3.19435	3.191029	3.194913	3.191069	3.203472	3.194967
mat8:	0.789807	0.780398	0.76446	0.778975	0.771031	0.776934
mat9:	1.007199	1.026566	0.982547	1.001137	0.991451	1.00178
mat10:	0.682107	0.680855	0.679347	0.679425	0.66462	0.677271
mat11:	0.688055	0.663537	0.681097	0.664414	0.660045	0.67143

InteriorT4						
mat1:	4.379701	4.382465	4.382222	4.377529	4.376194	4.379622
mat2:	1.958411	1.950323	1.969657	1.962494	1.952165	1.95861
mat3:	0.67569	0.655632	0.675428	0.674628	0.672432	0.670762
mat4:	3.409727	3.412405	3.411962	3.411972	3.421903	3.413594
mat5:	0.703335	0.701043	0.714763	0.697361	0.713122	0.705925
mat6:	2.048989	2.020531	2.043529	2.048408	2.034842	2.03926
mat7:	4.201937	4.212095	4.204847	4.219274	4.211331	4.209897
mat8:	0.937926	0.911434	0.890533	0.900236	0.901364	0.908299
mat9:	1.225567	1.250399	1.23679	1.246648	1.22726	1.237333
mat10:	0.673912	0.675006	0.672942	0.65175	0.675352	0.669792
mat11:	0.657779	0.666663	0.676551	0.658702	0.683448	0.668629

ENSAIO 11 - Total de píxeis influenciados por material (Cena: Interior)

	Total de píxeis afetados por material			
	InteriorT1	InteriorT2	InteriorT3	InteriorT4
mat1:	51851	77074	81750	86652
mat2:	447	13408	27673	32121
mat3:	1042	1441	1734	2827
mat4:	50748	58985	63002	65171
mat5:	1025	1709	2644	3742
mat6:	27347	30580	32070	33497
mat7:	38626	55870	69795	81742
mat8:	5941	6807	8076	9332
mat9:	9140	13192	14856	16673
mat10:	1669	1731	1967	2140
mat11:	399	1267	1906	2203