**ESCOLA POLITÉCNICA DA
UNIVERSIDADE DE SÃO PAULO**

EDUARDO TOLEDO SANTOS

**AVALIAÇÃO DO ALGORITMO DE
"RAY TRACING" EM MULTICOMPUTADORES**

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

São Paulo
1994

EDUARDO TOLEDO SANTOS

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

**AVALIAÇÃO DO ALGORITMO DE
"RAY TRACING" EM MULTICOMPUTADORES**

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

Área de Concentração:
Engenharia de Computação e Sistemas

Orientador:
Prof. Dr. JOÃO ANTONIO ZUFFO

São Paulo
1994

Santos, Eduardo Toledo

Avaliação do Algoritmo de "Ray Tracing" em
Multicomputadores. São Paulo, 1994.

181 p.

Dissertação (Mestrado) - Escola Politécnica da
Universidade de São Paulo. Departamento de Engenharia
de Computação e Sistemas.

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

Aos meus pais, que
sempre me apoiaram
incondicionalmente.

AGRADECIMENTOS

Ao meu orientador Prof. Dr. João Antonio Zuffo, pelo apoio constante e orientação para elaboração deste trabalho.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, pelo financiamento parcial desta dissertação.

Aos meus colegas e amigos do Laboratório de Sistemas Integráveis e seu Grupo de Computação Gráfica. Em especial a Marcelo K. Zuffo e Roseli de D. Lopes, pelas sugestões e pelo indispensável incentivo recebido durante o desenvolvimento desta pesquisa.

Ao Prof. Sérgio Takeo Kofuji, pelo suporte e confiança que depositou em mim.

À minha mãe, pela inspiração e exemplo. Ao meu pai, pelo incentivo e orientação de vida.

À Maria Laura Martinez, pelo carinho, apoio e compreensão.

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

SUMÁRIO

Lista de Figuras	
Lista de Tabelas	
Lista de Abreviaturas	
Lista de Símbolos	
Resumo	
"Abstract"	

1. INTRODUÇÃO	2
1.1. Motivação.....	3
1.2. Objetivos	4
1.3. Contribuições do Trabalho.....	4
1.4. Organização do Trabalho	5
2. SÍNTESE TRIDIMENSIONAL DE IMAGENS	8
2.1. Definições Básicas	9
2.2. Síntese por Emulação.....	11
2.2.1. O Dadoduto de Síntese de Imagens.....	12
2.2.2. Efeitos Especiais: Sombras, Reflexões, Transparência e Refração	17
2.3. Síntese por Simulação.....	19
2.3.1. "Ray Tracing"	19
2.3.2. Radiosidade	24
2.4. "Ray Tracing"	26
2.4.1. Modelamento Geométrico e Intersecções	27
2.4.1.1. "Ray Tracing" no Modelamento CSG	27
2.4.1.2. Cálculos de Intersecção e Vetor Normal para Esfera ..	29
2.4.1.3. Estimativas do Número de Operações.....	31
2.4.1.4. Outros Modelamentos	33
2.4.2. Técnicas para Aumento do Realismo.....	33
2.4.3. Técnicas de Aceleração	34
2.4.3.1. Aceleração do Cálculo de Intersecções	36
2.4.3.1.1. Envoltórios.....	36
2.4.3.1.2. Técnicas para Modelamentos Específicos	38
2.4.3.2. Redução do Número de Intersecções.....	39
2.4.3.2.1. Particionamento do Espaço 3D.....	39
2.4.3.2.2. Árvore de Envoltórios	40

2.4.3.2.3. Cache de Sombras / Cache da Árvore de Raios	43
2.4.3.2.4. Memorizador de Itens (Item Buffer).....	44
2.4.3.3. Redução do Número de Raios	45
2.4.3.3.1. Controle Adaptativo da Profundidade da Árvore de Raios	45
2.4.3.3.2. Super-amostragem Adaptativa.....	46
2.4.3.3.3. Seleção de Pontéis para "Ray Tracing"	47
2.4.3.3.4. Raios Generalizados	49
2.4.3.4. Técnicas Direcionais	50
2.4.3.4.1. Memorizador de Luz (Light Buffer).....	51
2.4.3.4.2. Coerência de Raios	52
2.4.3.4.3. Classificação de Raios	53
3. CONCEITOS E DEFINIÇÕES BÁSICAS EM COMPUTAÇÃO PARALELA	55
3.1. Memória Compartilhada x Memória Distribuída.....	55
3.2. Taxonomia de Flynn	55
3.3. Multiprocessadores x Multicomputadores	56
3.3.1. Por que Multicomputadores ?	56
3.4. Considerações sobre Topologia	57
3.5. Definições Básicas	58
3.5.1. Nó de Processamento	58
3.5.2. Sistema	58
3.5.3. Topologia	58
3.5.4. Elo ("Link").....	58
3.5.5. Conectividade.....	59
3.5.6. Granularidade	59
3.5.7. Latência de Comunicação	59
3.5.8. Memória Cache	59
3.5.9. Balanceamento de Carga	60
3.5.10. Sobrecarga ("Overhead").....	60
3.5.11. Redundância de Dados	60
3.5.12. Fazenda de Processadores ("Processor Farm")	60
3.5.13. Ganho de Velocidade	60
3.5.14. Eficiência.....	61
3.5.15. Lei de Amdahl.....	61
4. "RAY TRACING" PARALELO: CONCEITUAÇÃO, AVALIAÇÃO E TAXONOMIA.....	63
4.1. Conceituação do Problema de "Ray Tracing" Paralelo.....	63

4.2. Soluções de Referência	64
4.3. Uma Ferramenta Gráfica para Avaliação de Soluções.....	67
4.3.1. Eficiência Incremental.....	67
4.3.2. Redundância Incremental	68
4.3.3. Lucro de um Sistema de "Ray Tracing"	69
4.3.4. Índice de Lucro.....	71
4.3.5. O Gráfico Eficiência x Redundância Incrementais	72
4.4. Parâmetros para Avaliação do Alg. de "Ray Tracing" em Multicomputadores	76
4.4.1. Sobrecargas de Processamento.....	76
4.4.2. Balanceamento	77
4.4.3. Comunicação.....	78
4.4.4. Uso de Memória.....	79
4.4.5. Ganho de Velocidade e Eficiência	79
4.4.6. Restrições Algorítmicas	79
4.5. Taxonomias de Soluções.....	80
4.5.1. Classificações de Sistemas para "Ray Tracing" Paralelo	81
4.5.1.1. A Classificação de S. Gaudet et alli	81
4.5.1.2. A Classificação de M. McNeill et alli	82
4.5.1.3. A Classificação por Fluxo de Dados	82
4.5.2. A Taxonomia de Green	84
4.5.3. Uma Nova Taxonomia	87
4.5.3.1. Critérios de Classificação	92
4.5.3.2. Classificação de Propostas Publicadas	93
5. O DESEMPENHO DO ALGORITMO DE "RAY TRACING" EM MULTICOMPUTADORES	97
5.1. Particionamento do Espaço.....	97
5.1.1. Subdivisões Uniforme, Adaptativa e Dinâmica	99
5.1.2. Alocações em Bloco, Distribuída e Dinâmica.....	104
5.2. Particionamento de Objetos	107
5.2.1. Subdivisões Uniforme e Adaptativa.....	108
5.2.2. Replicação de Dados	111
5.3. Particionamento de Imagem.....	114
5.3.1. Replicação de Dados	114
5.3.2. Alocações em Bloco, Distribuída e Dinâmica.....	118
5.4. Particionamento de Quadros	122
5.5. Particionamento de Raios.....	126
5.6. Particionamento Estrutural.....	128

5.6.1. Representação CSG.....	130
5.6.2. Representação por Octárvore ("Octree").....	134
5.6.3. Representação por Enumeração Espacial.....	137
5.7. Particionamento Funcional	140
5.8. Particionamentos Híbridos.....	143
5.8.1. Espaço + Objetos.....	144
5.8.2. Espaço + Raios.....	144
5.8.3. Estrutural + Imagem.....	145
5.8.4. Funcional + Imagem.....	146
5.8.5. Funcional + Espaço	148
6. DISCUSSÃO E APLICAÇÕES	151
6.1. Conclusões sobre Desempenho.....	151
6.2. A Melhor Solução Paralela para "Ray Tracing".....	153
6.3. Novas Alternativas para "Ray Tracing" Paralelo	157
6.3.1. Particionamento de Espaço	158
6.3.1.1. Sem Replicação ou com Replicação Total de Dados ..	159
6.3.1.2. Subdivisão Adaptativa/Dinâmica com Alocação Distribuída/Dinâmica	159
6.3.2. Particionamento de Objetos	160
6.3.2.1. Técnicas hierárquicas	160
6.3.2.2. Técnicas não hierárquicas.....	160
6.3.3. Particionamento de Imagem.....	160
6.3.4. Particionamentos Híbridos	161
6.3.5. Particionamento de Quadros	162
7. CONCLUSÕES E TRABALHOS FUTUROS.....	164
7.1 Conclusões Finais	164
7.2 Trabalhos Futuros	165
REFERÊNCIAS BIBLIOGRÁFICAS	168

Apêndice I - Dicionário

Apêndice II - Descrição e Créditos das Imagens

LISTA DE FIGURAS

FIG. 2.1 - Definição de raio	11
FIG. 2.2 - As principais etapas do dadoduto de síntese de imagens	12
FIG. 2.3 - Matriciação incremental de um triângulo.....	14
FIG. 2.4 - O Algoritmo Memorizador Z ('Z-Buffer')	15
FIG. 2.5 - Algoritmo de tonalização com interpolação de normais (Phong)	16
FIG. 2.6 - Elementos do cenário de 'ray tracing'	21
FIG. 2.7 - Cálculo do raio refletido.....	23
FIG. 2.8 - Cálculo do raio transmitido.....	23
FIG. 2.9 - O algoritmo de 'ray tracing'	24
FIG. 2.10 - Sistema de equações para radiosidade.....	25
FIG. 2.11 - Um sólido modelado segundo a técnica CSG	28
FIG. 2.12 - Diagrama de Roth.....	29
FIG. 2.13 - Geometria da intersecção raio-esfera	30
FIG. 2.14 - Classificação das técnicas de aceleração de 'ray tracing'.....	35
FIG. 2.15 - Objeto com envoltório de intersecção de nacos	38
FIG. 2.16 - Fragmentação e falsa intersecção	40
FIG. 2.17 - Hierarquia de envoltórios	41
FIG. 2.18 - Intersecções com envoltórios	42
FIG. 2.19 - Intersecções nos particionamentos de objetos e de espaço	43
FIG. 2.20 - Super-amostragem.....	46
FIG. 2.21 - Seleção de Pontéis para 'Ray Tracing'	47
FIG. 2.22 - O cubo direcional	50
FIG. 2.23 - O teorema da coerência de raios.....	52
FIG. 3.1 - Topologias.....	59
FIG. 4.1 - As soluções de referência no espaço eficiência x redundância	72
FIG. 4.2 - As regiões do espaço eficiência x redundância incrementais.....	74
FIG. 4.3 - A classificação de Gaudet et alli'88.....	81
FIG. 4.4 - A classificação de McNeill et alli'92.....	82
FIG. 4.5 - A classificação por fluxo de dados para RT paralelo	83
FIG. 4.6 - Taxonomia de soluções paralelas para RT proposta por S. Green.....	85
FIG. 4.7 - Nova proposta de taxonomia de soluções paralelas para RT	87
FIG. 4.8 - Correspondência entre as taxonomias para 'ray tracing' paralelo	91
FIG. 5.1 - Influência da distribuição de objetos para partic. do espaço	98
FIG. 5.2 - Subdivisão no particionamento do espaço	99
FIG. 5.3 - Efeito do tipo de subdivisão no particionamento do espaço	100
FIG. 5.4 - Partic. do espaço adaptado à topologia do multicomputador.....	102

FIG. 5.5 - Célula com nove vizinhos	102
FIG. 5.6 - Alocação de células aos processadores para partic. do espaço	104
FIG. 5.7 - Efeito do tipo de alocação no particionamento do espaço	105
FIG. 5.8 - Balanceamento de carga em função do tamanho da sub-região	106
FIG. 5.9 - Particionamento de objetos - técnicas não hierárquicas	108
FIG. 5.10 - Efeito da subdivisão no particionamento de objetos	109
FIG. 5.11 - Replicação de dados em hierarquia de envoltórios	112
FIG. 5.12 - Efeito da replicação de dados no particionamento de objetos	112
FIG. 5.13 - Balanceamento de carga em função do NÍVEL de distribuição	113
FIG. 5.14 - Efeito da replicação de dados no particionamento de imagem	115
FIG. 5.15 - Eficiência no partic. de imagem com replicação parcial de dados	117
FIG. 5.16 - Efeito do tipo de alocação no particionamento de imagem	118
FIG. 5.17 - A fragmentação de imagem	119
FIG. 5.18 - Influência da granularidade na eficiência do partic. de imagem	121
FIG. 5.19 - Desempenho do particionamento de quadros	124
FIG. 5.20 - Desempenho do particionamento de raios	127
FIG. 5.21 - Particionamento estrutural para representação CSG	130
FIG. 5.22 - Desempenho do particionamento estrutural - CSG	131
FIG. 5.23 - Alocação de nós da octárvore aos processadores	135
FIG. 5.24 - Desempenho do partic. estrutural - representação por octárvore	135
FIG. 5.25 - Desempenho do particionamento estrutural - enumeração espacial	138
FIG. 5.26 - Particionamento funcional puro	141
FIG. 5.27 - Desempenho do particionamento funcional	142
FIG. 5.28 - Particionamento híbrido: Espaço + Raios	144
FIG. 5.29 - Particionamento híbrido: Estrutural + Imagem	146
FIG. 5.30 - Particionamento híbrido: Funcional + Imagem	147
FIG. 5.31 - O sistema LINKS-1 (Part. Híbrido Funcional + Imagem)	147
FIG. 5.32 - Arquitetura do sistema de Kobayashi'87	148
FIG. 6.1 - Abrangência das propostas de 'ray tracing' paralelo	158

LISTA DE TABELAS

TAB. 2.1 - Ref. sobre 'ray tracing' para diversos tipos de modelamento geométrico ..	33
TAB. 2.2 - Referências sobre técnicas para aumento de realismo em 'ray tracing'	34
TAB. 3.1 - A taxonomia de Flynn	55
TAB. 4.1 - Soluções de Referência para	66
TAB. 4.2 - Comparação entre Eficiência e Eficiência Incremental	68
TAB. 4.3 - A mem. tot. usada por uma solução em função de sua redundância	69
TAB. 4.4 - Eficiência e redundância incrementais das Soluções de Referência	69
TAB. 4.5 - Parâmetros de desempenho de sistemas paralelos para 'ray tracing'	70
TAB. 4.6 - Lucro das Soluções de Referência	71
TAB. 4.7 - Índices de Lucro das soluções de referência	72
TAB. 4.8 - As regiões do gráfico eficiência x redundância	75
TAB. 4.9 - Classificação de propostas na nova taxonomia	94
TAB. 5.1 - Comparação entre partic. de quadros e de imagem para redes de comp. ...	123
TAB. 6.1 - Comparação entre particionamento de objetos e de espaço	151
TAB. 6.2 - Comparação entre particionamentos de raios, quadros e funcional	151
TAB. 6.3 - Comparação entre particionamento de imagem e de espaço	152
TAB. 6.4 - Comparação entre características da replicação de dados	152
TAB. 6.5 - Comparação entre características da alocação de partições	152
TAB. 6.6 - Comparação entre características da subdivisão	153

LISTA DE ABREVIATURAS

BD	Banco de Dados
BSP	Particionamento Binário do Espaço ("Binary Space Partitioning")
CAD	Projeto Auxiliado por Computador ("Computer Aided Design")
CSG	Geometria de Sólidos Construtivos ("Constructive Solid Geometry")
c_{te}	Constante
ISO	"International Organization for Standardization"
MFLOPS	Milhões de Operações de Ponto Flutuante por Segundo ("Millions of Floating Point Operations per Second")
PI	Processador de Intersecção
PS	Processador de Sobreamento
RGB	Vermelho-Verde-Azul ("Red-Green-Blue")
RT	"Ray Tracing"

LISTA DE SÍMBOLOS

I	intensidade de um ponto amostrado no plano de imagem
k_d	coeficiente de reflexão difusa do material (componentes R, G e B)
$\vec{N}$	vetor unitário normal à superfície do objeto
$\vec{L}_j$	vetor na direção da j-ésima fonte de luz; módulo igual à sua intensidade
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
$\vec{L}_j$	vetor na direção da j-ésima fonte de luz, $ \vec{L}_j = \text{intensidade da 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	coeficiente de refração
B	radiosidade de uma superfície
E	emissão de uma superfície (luz própria)
ρ	reflexividade
F	fator de forma
NP	núm. de superf. discretas ou "retalhos" (<i>patches</i>) que compõem o cenário
E_c	centro de esfera
X_c, Y_c, Z_c	coordenadas do centro de esfera
E_r	raio de esfera
R_o	origem do raio
$\vec{r}$	vetor unitário na direção do raio
$R(t)$	raio
R_o	origem do raio
P_m	ponto do raio mais próx. de E_c
d	distância de P_m até E_c
P_{int}	ponto de intersecção
d_{oc}	distância de R_o até E_c
C_{tot}	custo total de uma intersecção
b	número de vezes que um envoltório é testado
C_b	custo de intersecção de um raio com o envoltório
i	número de raios que atingem o envoltório
C_{int}	custo de intersecção de um raio com um objeto

P	passo da amostragem inicial em seleção de pontéis para "ray tracing"
Q	limiar de diferença de intensidade de cores (sel. de pontéis para RT)
R	redundância de dados
m_N	memória total utilizada num multicomputador com N nós
M	memória utilizada pelo algoritmo seqüencial
N	número de nós do multicomputador
T	tempo de processamento com um nó
t_N	tempo de processamento com N nós
G	ganho de velocidade
E	eficiência
σ	proporção de uma tarefa que é inerentemente seqüencial
α	eficiência incremental
β	redundância incremental
L	lucro do sistema
C_a	custo de animação
C_p	custo do processador
C_m	custo de memória
C_i	carga do processador i
γ_i	complexidade média dos objetos alocados ao nó i
n_i	número de objetos alocados ao nó i
r_i	número de raios que chegam às células do nó i
T_r	tempo em execução
T_w	tempo em espera
m_i	memória local do nó i
ϕ	fração dos dados totais alocados em memória cache
N_m	número de mensagens
N_p	número de primitivos CSG
N_c	número de combinações CSG
N_r	número de raios
<i>tamanho</i>	largura da fatia do volume de dados atribuída a um nó de processamento
l_x	largura do volume de dados na direção x
$t_{\text{médio}}$	tempo médio de processamento dos processadores
$t_{\text{total } i}$	tempo de processamento total do processador i

RESUMO

A Computação Gráfica, área em franco desenvolvimento, têm caminhado em busca da geração, cada vez mais rápida, de imagens mais realísticas. Os algoritmos que permitem a síntese de imagens realísticas demandam alto poder computacional, fazendo com que a geração deste tipo de imagem, de forma rápida, requeira o uso de computadores paralelos.

Hoje, a técnica que permite gerar as imagens mais realísticas é o "ray tracing"¹. Os multicomputadores, por sua vez, são a arquitetura de computadores paralelos mais promissora na busca do desempenho computacional necessário às aplicações modernas.

Esta dissertação aborda o problema da implementação do algoritmo de "ray tracing" em multicomputadores. A paralelização desta técnica para uso em computadores paralelos de memória distribuída pode ser feita de muitas formas diferentes, sempre envolvendo um compromisso entre a velocidade de processamento e a memória utilizada.

Neste trabalho conceitua-se este problema e introduz-se ferramentas para a avaliação de soluções que levam em consideração a eficiência de processamento e a redundância no uso de memória. Também é apresentada uma nova taxonomia que, além de permitir a classificação de propostas para implementações de "ray tracing" paralelo, orienta a procura de novas soluções para este problema. O desempenho das soluções em cada classe desta taxonomia é avaliado qualitativamente.

Por fim, são sugeridas novas alternativas de paralelização do algoritmo de "ray tracing" em multicomputadores.

¹Decidiu-se manter este termo no original em inglês, segundo sua denominação consagrada, devido ao fato de que será repetido centenas de vezes neste trabalho e as traduções correntes (lançamento de raios, traçado de raios luminosos, etc.) têm grafia demasiado longas.

"ABSTRACT"

Computer Graphics is headed today towards the synthesis of more realistic images, in less time. The algorithms used for realistic image synthesis demand high computer power, so that the synthesis of this kind of image, in short periods of time, requires the use of parallel computers.

Nowadays, the technique that yields the most realistic images is ray tracing. On its turn, multicomputers are the most promising parallel architecture for reaching the performance needed in modern applications.

This dissertation is on the problem of implementing the ray tracing algorithm on multicomputers. The parallelization of this technique on distributed memory parallel computers can take several forms, always involving a compromise between speed and memory.

In this work, this problem is conceptualized and tools for evaluation of solutions that account for efficiency and redundancy, are introduced. It is also presented a new taxonomy that can be used for both the classification of parallel ray tracing proposals and for driving the search of new solutions to this problem. The performances of entries in each class of the taxonomy are qualitatively assessed.

New alternatives for parallelizing the ray tracing algorithm on multicomputers, are suggested.

CAPÍTULO 1

INTRODUÇÃO

1. INTRODUÇÃO

A *Computação Gráfica* é um campo que está recebendo cada vez mais a atenção da comunidade científica internacional. Da mesma forma, o interesse na utilização comercial de técnicas de computação gráfica também é crescente. Este interesse é devido aos enormes avanços nas áreas de microeletrônica e arquitetura de computadores que permitiram que muitas aplicações antes inviáveis computacionalmente pudessem finalmente ser atendidas. Restrições quanto à resolução espacial e de cores nos monitores e velocidade ou memória disponível nos computadores estão bastante menores do que eram há alguns anos atrás.

Atualmente a Computação Gráfica já é parte integrante do dia-a-dia da sociedade como se vê em comerciais de TV, cinema, vídeo-games, etc. Com a disponibilidade de equipamentos de alta desempenho e custo mais baixo, rapidamente surgiram aplicações em campos como publicidade, desenho industrial, visualização de modelos tridimensionais complexos, estudo de iluminação, CAD, animação, efeitos especiais, realidade virtual, visualização médica, editoração, apresentações e simulação de vôo, entre tantos outros. Em muitas destas aplicações cada vez mais é necessário gerar imagens mais sofisticadas, mais realísticas e mais rapidamente.

Genericamente, a Computação Gráfica procura transformar dados em imagens. Um segmento particular da Computação Gráfica conhecido por *Síntese de Imagens Foto-realísticas* procura gerar imagens que se aproximem daquelas que seriam obtidas fotografando-se a realidade². Estas imagens, normalmente, apresentam sombras, reflexos, transparências, texturas, etc. Dentre as técnicas usadas neste segmento destaca-se aquela denominada "ray tracing" pois é a que permitiu gerar as imagens mais realísticas até hoje produzidas por computador e inclui todos os efeitos mencionados. O "ray tracing" surgiu em 1968 (APPEL, 1968) mas permaneceu no esquecimento até praticamente 1980 pois, até então, não havia máquinas com poder de processamento suficiente para gerar imagens sofisticadas num período de tempo viável.

Apesar da grande evolução na capacidade de processamento dos equipamentos, a demanda computacional das aplicações é crescente. A síntese de imagens de objetos tridimensionais é muito cara computacionalmente e o uso de técnicas mais realísticas, com cenas de alta complexidade amplia ainda mais o custo por imagem.

Além da Computação Gráfica, outro campo que vem apresentando resultados bastante promissores é a *Computação Paralela*. Esta área já vem sendo estudada a bastante tempo, contando com vários congressos especializados realizados anualmente em todo o

² Veja discussão sobre o termo "imagem realística" no capítulo 2.

mundo. Apesar de menos desenvolvida comercialmente devido a fatores relacionados à dificuldade de programação dos computadores paralelos, vem paulatinamente se firmando com a presença de várias máquinas multiprocessadoras no mercado e a decisão de gigantes como *IBM* e *Cray* de montarem laboratórios e/ou grupos para pesquisar e desenvolver computadores maciçamente paralelos (BELL, 1992), tendo esta última recentemente lançado um modelo nesta linha.

A computação paralela tem-se mostrado como uma solução alternativa aos supercomputadores com um ou poucos processadores ultra rápidos já que os computadores maciçamente paralelos apresentam relação desempenho/custo mais favorável. Isto permite que se construam computadores com desempenho muito alto a custos comparativamente baixos. Também tem sido aplicada em conjunto com a computação gráfica com sucesso (BURKE & LELER, 1990).

1.1. Motivação

É no contexto anteriormente colocado que surge a motivação para a presente pesquisa. Este trabalho aborda duas áreas tecnológicas em ascensão e de grande relevância. Procura-se estudar a utilização da Computação Paralela para viabilizar o uso mais amplo da técnica de "ray tracing" na síntese de imagens realísticas.

Apesar da enorme redução no tempo de processamento proporcionada pelas técnicas de aceleração de "ray tracing", o processamento paralelo ainda é necessário para atingir-se taxas interativas (McNEILL et alli, 1992).

O algoritmo de "ray tracing" é inerentemente paralelo, o que torna esta aplicação especialmente interessante para implementação em máquinas com mais de um processador. Além disso, sua grande demanda de processamento justifica o emprego de equipamentos deste tipo.

A motivação inicial ao estudo desenvolvido nesta dissertação foi a necessidade de identificar-se qual a melhor forma de implementar-se o algoritmo de "ray tracing" em computadores paralelos de memória distribuída (multicomputadores). Acreditamos que este seja o melhor algoritmo de síntese de imagens foto-realísticas disponível e que a classe dos multicomputadores representa uma das mais importantes e auspiciosas tendências de arquitetura de computadores hoje em desenvolvimento.

O estudo das alternativas de paralelização de "ray tracing" nos mostrou que este é um problema de alta complexidade devido ao grande número de diferentes fatores envolvidos e sua complexa inter-relação, dificultando uma visão geral do problema e suas

soluções. A necessidade de compreensão do problema motivou-nos a tentar sistematizar os enfoques adotados e a generalizar soluções e conclusões espalhadas por dezenas de artigos publicados em livros e periódicos internacionais.

Conforme (BOOTHE, 1989) " '*ray tracing*' em máquinas de memória compartilhada deve ser considerado um problema praticamente resolvido. [...] '*Ray tracing*' em máquinas de memória distribuída permanece ainda não resolvido..."³.

Todos estes fatores nos motivaram e nos desafiam a caminhar em direção à solução do problema de "ray tracing" em multicomputadores.

1.2. Objetivos

O que se pretende neste trabalho é, além de estudar-se propostas de implementação de "ray tracing" em multicomputadores, desenvolver-se ferramentas que auxiliem este estudo, permitindo uma melhor compreensão do problema e visualização de novas alternativas para solucioná-lo.

1.3. Contribuições do Trabalho

Pode-se destacar, como sendo as contribuições de cunho original mais importantes apresentadas neste trabalho, as seguintes:

- Proposta de uma nova taxonomia para classificação e estudo de algoritmos paralelos de "ray tracing", tendo como características inovadoras:
 - identificação das características chave para classificação de propostas;
 - distinção entre *subdivisão* e *alocação*, conceitos antes tratados indistintamente, na literatura de "ray tracing", como "particionamento";
 - identificação das subclasses *uniforme*, *adaptativa* e *dinâmica* para a subdivisão e as subclasses *em bloco*, *distribuída* e *dinâmica* para a alocação;
 - identificação da classe de *particionamento estrutural* cujas propostas anteriormente eram classificadas como particionamento do espaço ou de objetos;

³ T. do A.

- introdução da denominação *particionamento de raios*;
- introdução da denominação *particionamento funcional* no contexto de "ray tracing".
- Proposta de soluções de referência de desempenho para "ray tracing" paralelo;
- Proposta dos conceitos de *eficiência e redundância incrementais*.
- Proposta do gráfico *Eficiência x Redundância Incrementais* para qualificar propostas de "ray tracing" paralelo. Análise de suas regiões;
- Proposta de um índice de mérito para qualificar propostas de "ray tracing" paralelo em multicomputadores (*Índice de Lucro*);
- Introdução do conceito de *fragmentação de imagem*;
- Seleção de critérios para análise de propostas de paralelização de "ray tracing";
- Apresentação do estado-da-arte em "ray tracing" paralelo para multicomputadores; Síntese de suas características de desempenho;
- Conceituação clara e direta do problema de "ray tracing" em multicomputadores, estabelecendo sua relação com tempo de processamento e memória utilizada;
- Apresentação, em forma mais sintética e atualizada que o clássico (ARVO & KIRK, 1989), das principais técnicas de aceleração de "ray tracing";
- Coletânea de aplicações de "ray tracing";
- Detalhamento e análise crítica da taxonomia proposta por Green (GREEN, 1991);
- Sugestão de novas alternativas de paralelização do algoritmo de "ray tracing";
- Afirmação da importância das restrições algorítmicas sobre o desempenho de algoritmos de "ray tracing".

1.4. Organização do Trabalho

Logo após esta introdução, o Capítulo 2 aborda tópicos de Computação Gráfica que procuram embasar a escolha do algoritmo de "ray tracing" como o ponto central deste trabalho. Dentro dos paradigmas de síntese de imagens por simulação e emulação, procura-se caracterizar, subjetivamente, a complexidade dos algoritmos tradicionais (emulação) e,

em oposição, a simplicidade e elegância algorítmica da técnica "ray tracing". Também neste capítulo é apresentada uma descrição completa do algoritmo clássico de "ray tracing" e referências a extensões e melhoramentos propostos na literatura.

O Capítulo 3 contextualiza o jargão de processamento paralelo dentro do escopo em que é utilizado neste trabalho e justifica algumas decisões desta pesquisa em relação ao processamento paralelo.

No Capítulo 4 são introduzidas as principais contribuições desta dissertação. O problema que nos propusemos a estudar é conceituado. São introduzidas algumas definições e ferramentas que auxiliarão este estudo. Finalmente, é apresentada uma taxonomia para classificação e desenvolvimento de soluções paralelas para "ray tracing".

O Capítulo 5 traz, organizadas segundo a taxonomia introduzida, discussões e análises sobre as várias alternativas de paralelização do algoritmo de "ray tracing" em multicomputadores, apresentando o estado-da-arte nesta tecnologia.

No capítulo 6 resumem-se conclusões sobre o desempenho das soluções. Discute-se qual a melhor solução. São apresentadas novas sugestões de "ray tracing" paralelo, ilustrando o uso da taxonomia para este fim.

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

CAPÍTULO 2

**SÍNTESE TRIDIMENSIONAL
DE IMAGENS**

2. SÍNTESE TRIDIMENSIONAL DE IMAGENS

Segundo a definição da ISO (International Organization for Standardization), a Computação Gráfica é "o conjunto de métodos e técnicas de converter dados para um dispositivo gráfico, via computador". Pode-se deduzir, portanto, que a atividade central da Computação Gráfica é a síntese de imagens. Por este motivo, desde o seu início, muitas técnicas foram desenvolvidas para este fim. Algumas destas técnicas são capazes de gerar imagens de grande qualidade visual, geralmente denominadas *realísticas* ou *foto-realísticas*.

A definição de "imagem realística" é causa de polêmica, variando no tempo e de acordo com a cultura e aplicação. (BLAKE, 1990) define "realístico" como "*o que quer que o espectador aceite como tal - qualquer coisa que convença o espectador o suficiente para que ele acredite na imagem*"⁴. Este autor mostra ainda, que mesmo exemplos normalmente tomados como representações realísticas (fotografia, por exemplo) podem causar incertezas.

Alguns importantes autores defendem que a síntese de imagens deveria, assim como outras tecnologias, avançar baseada nas leis da Física (GREENBERG, 1988). Os algoritmos deveriam incorporar modelos que descrevessem fenômenos físicos para que se possa atingir resultados visuais "realísticos".

Por outro lado, muitos pesquisadores entendem que o requisito básico para os algoritmos de síntese de imagens realísticas é apenas que gerem a imagem pretendida, mesmo que sem qualquer conexão com a realidade física (REEVES, 1987), (FRENKEL, 1988).

(MILLS, 1985) denomina o resultado do enfoque baseado em conceitos físicos de *identidade óptica* enquanto o outro é chamado de *simulação óptica*, tomando como base o estímulo luminoso que chega aos olhos de um observador. Outros trabalhos (ZUFFO et alli, 1990) (LOPES, 1993) usam as denominações *síntese por simulação* e *síntese por emulação* respectivamente. Adotou-se esta última terminologia neste trabalho por ser mais familiar ao jargão da Engenharia Eletrônica.

Num cenário futuro, é possível que ambas as estratégias possam ser unificadas sob um mesmo paradigma, como propõe (BLAKE, 1990) e que também sejam levados em conta aspectos psico-sensoriais como ilustrado por (MILLS, 1985) para que se possa desenvolver algoritmos mais eficientes e realísticos.

⁴ T. do A.

Acreditamos que tanto a simulação quanto a emulação tem suas qualidades e defeitos. Ambos os enfoques têm aplicações onde um ou outro é mais favorável. No entanto, até o momento, as técnicas de síntese de imagens por simulação têm obtido resultados visuais mais próximos do "realismo" do que as rápidas técnicas de emulação, razão pela qual optou-se aqui pelo estudo de um algoritmo desta classe, descrito detalhadamente na seção 2.3.1.

Apresenta-se, a seguir, uma rápida descrição de algumas técnicas de emulação orientada ao realismo e as técnicas de simulação. Pretende-se que o leitor observe a complexidade dos algoritmos de emulação em contraste com a relativa simplicidade da técnica "ray tracing". A qualidade das imagens geradas por esta técnica está exemplificada nas ilustrações de abertura de cada capítulo deste trabalho⁵.

2.1. Definições Básicas

A seguir são apresentadas definições de termos com significado particular, dentro do escopo deste trabalho.

A) Imagem, Animação e Quadro

Uma *imagem* é conjunto de pontos alinhados segundo uma grade plana, normalmente retangular, dos quais se conhece a cor. Um conjunto de imagens que formam uma sequência temporal é denominado *animação*. Cada imagem de uma animação é denominada *quadro*.

B) Objetos

Objetos são cada uma das entidades geométricas, definidas num processo anterior à síntese da imagem (modelamento), que compõem o cenário. Têm associadas a eles informações geométricas (forma, tamanho e posição) e ópticas (cor, transparência, reflexividade, textura, etc.).

Deve ficar claro que *objeto*, no sentido empregado neste trabalho, é uma referência a elementos individuais, mesmo que agrupados para formar entidades mais complexas. Assim, por exemplo, cada polígono utilizado para modelamento de uma cadeira será considerado um objeto independente. Uma esfera modelada através de representação implícita (centro e raio) é um único objeto. A mesma esfera, representada por polígonos, constituirá um conjunto de vários objetos.

⁵ Descrição e créditos encontram-se no apêndice II

C) Cenário

É um conjunto de dados contendo informações sobre objetos e fontes de luz (tipo, posição e intensidade).

D) Observador

Uma posição no espaço 3D correspondente ao centro de projeção dos objetos do cenário num plano onde se forma a imagem.

E) Cena e Cena-Teste

Uma *cena* é constituída por um cenário, um observador e o plano de imagem onde serão projetados os objetos do cenário, segundo a posição do observador.

Uma cena-teste é uma cena escolhida para testar o desempenho de um algoritmo de síntese de imagens. Deve, de preferência ser padronizada. Vide, por exemplo, (HAINES, 1987).

F) Pontel ("Pixel")

O menor elemento de uma imagem. A resolução de uma imagem é definida pelo número de pontéis nas suas duas dimensões.

G) Memória de Quadro ("Frame Buffer")

É uma região especial de memória que contém informações representando uma imagem. Geralmente existem, associado a este banco de memória, circuitos especiais que transformam estas informações em sinal de vídeo, permitindo sua exibição em monitores ou registro em gravadores de vídeo.

H) Raio

Raio é o segmento de reta que representa um trecho do caminho percorrido pela luz no cenário. É descrito por uma origem (ponto no espaço) e uma direção (vetor no espaço).

Deseja-se destacar que, no sentido usado neste texto, raio denomina apenas um segmento da trajetória e não a trajetória completa da luz, desviada por reflexões e refrações. Veja figura 2.1

FIG. 2.1 - Definição de raio

I) Coerência

Existe *coerência* quando se pode compartilhar informações entre diferentes computações (GLASSNER, 1989c). Diversos tipos de coerência (GREEN, 1991) podem ser exploradas para aceleração de algoritmos de síntese de imagens:

- *coerência de objetos*: objetos ocupam vizinhanças locais do espaço. Objetos distintos são disjuntos neste espaço.
- *coerência de raios*: raios cuja origem e direção são quase iguais tendem a interceptar os mesmos objetos.
- *coerência de imagem*: a constância local dos objetos origina uma constância similar na imagem, onde mudanças só ocorrem gradualmente.
- *coerência de dados*: se é feito um acesso aos dados de descrição de um certo conjunto de objetos, é provável que brevemente os mesmos dados sejam acessados novamente.
- *coerência temporal ou de quadro*: geralmente, a aparência de um quadro não é muito distinta daquela que imediatamente o segue, numa animação.

2.2. Síntese por Emulação

As técnicas de síntese de imagens por emulação procuram obter imagens retratando a realidade porém sem preocupar-se com os fenômenos ópticos reais. São técnicas que procuram sintetizar imagens usando artifícios algorítmicos sem conexão com a física óptica da cena.

A vantagem de tal enfoque é que pode-se fazer amplo uso de coerência e gerar imagens com rapidez. Sua desvantagem é que os algoritmos tornam-se complexos do ponto

de vista da implementação, principalmente quando deseja-se obter imagens com efeitos de realismo. Estes algoritmos normalmente não conseguem atingir o chamado foto-realismo.

A seguir serão abordadas as técnicas mais populares para síntese por emulação, incluindo algoritmos essenciais para geração de imagens tonalizadas (transformações, matriciação, visibilidade e tonalização) e também alguns algoritmos para geração de efeitos de realismo (sombras, reflexão, refração e transparência). O objetivo é ilustrar a complexidade algorítmica destas técnicas em comparação com o "ray tracing", a tema central deste trabalho.

2.2.1. O Dadoduto de Síntese de Imagens

O processo de gerar uma imagem a partir de dados, a síntese de imagens, é constituído por uma série de etapas. As principais etapas que constituem este processo são as *transformações* (geométricas, de visualização, etc), a *matriciação* (também chamada rasterização), a *determinação de visibilidade* e a *tonalização* (também denominada iluminação).

A combinação destas etapas constitui o chamado dadoduto (*pipeline*) de síntese de imagens (LOPES, 1993) (fig. 2.2). A ordem das etapas pode variar de acordo com a metodologia empregada e os algoritmos escolhidos para cada fase. Em alguns casos as etapas podem até se confundir. Nos algoritmos, em geral, existem trechos intercalados das várias etapas (por exemplo, vide fig. 2.3).

FIG. 2.2 - As principais etapas do dadoduto de síntese de imagens

A geração da imagem só pode ser feita com a implementação completa dos algoritmos de todas as etapas do dadoduto. Apesar da grande complexidade para

implementação do processo como um todo, sua demanda computacional é mais baixa que a requerida por técnicas de simulação.

É importante notar que a maioria destes algoritmos foi desenvolvida para operar sobre objetos descritos por representação poligonal. Assim, em caso de o cenário estar representado segundo outras técnicas de modelamento, pode ser necessário um pré-processamento para conversão da representação geométrica.

Exemplifica-se abaixo, técnicas populares para cada etapa. A prevalência destas técnicas geralmente advém da facilidade de sua implementação em circuitaria ("hardware").

A) Transformações: A Transformação Perspectiva

As transformações, em geral, podem ser expressas por uma matriz de transformação que é aplicada aos vértices que definem os objetos do cenário, usualmente em coordenadas homogêneas:

$$P = M.P'$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix}$$

As transformações podem ser geométricas (translação, rotação, escalamento e cisalhamento) ou de visualização (projeção paralela ou perspectiva).

O objetivo da transformação perspectiva é projetar os objetos do cenário no plano correspondente à tela virtual, segundo um centro de projeção posicionado junto ao observador. Após a aplicação desta transformação, desprezando-se a componente z, tem-se as coordenadas das projeções dos vértices dos objetos no plano de projeção (tela virtual).

B) Matriciação: O Algoritmo de Varredura Incremental

O processo de matriciação consiste na enumeração dos pontéis da tela onde se projetam os objetos do cenário.

O algoritmo de varredura incremental consiste de dois laços (*loops*): um, mais externo, varre verticalmente todas as linhas da tela virtual. O outro laço, interno, varre cada

Um algoritmo de matriciação incremental de um triângulo é mostrado na fig. 2.3 (adaptado de GOMES & VELHO, 1990). Note-se como as etapas de determinação de visibilidade e tonalização estão "incrustadas" entre o código da matriciação.

As operações de incremento podem ser realizadas em aritmética inteira, acelerando o algoritmo.

```
matriciação(t) #faz a matriciação do triângulo t
• topo = vértice_superior(t);
• arest_esq = cria_aresta(t[topo], t[topo-1]);      # cria uma aresta entre os pontos indicados
• arest_dir = cria_aresta(t[topo], t[topo+1]);
• arest_inf = cria_aresta(t[topo-1], t[topo+1]);

Enquanto não fim_esq e não fim_dir faça
  • seg = cria_seg(arest_esq, arest_dir);      # cria um segmento horizontal (span) na linha
                                              # corrente entre as arestas indicadas;

  Enquanto não fim_seg faça
    • Se seg é visível então      # VISIBILIDADE
      tonaliza pontel;           # TONALIZAÇÃO
    • fim_seg = incr_s(seg);      # ativa próximo pontel do segmento. Se for o último
                                # pontel do segmento retorna VERDADE;
  • fim_esq = incr_a(arest_esq);  # ativa próxima linha de varredura. Se a aresta
                                # termina nesta linha retorna VERDADE;

  • Se fim_esq então
    • arest_esq = arest_inf;

  • fim_dir = incr_a(arest_dir);
  • Se fim_dir então
    arest_dir = arest_inf;

fim
```

FIG. 2.3 - Matriciação incremental de um triângulo

Existem algoritmos para algumas outras formas de representação diferentes da representação por fronteiras (B-Rep) poligonais: (LANE et alli, 1980) apresentam três algoritmos para matriciação de superfícies parametrizadas.

C) Visibilidade: O Algoritmo Memorizador Z (Z-Buffer)

A etapa de determinação de visibilidade deve estabelecer qual é, dentre todos os objetos que se projetam sobre um dado pontel, aquele mais próximo do observador, isto é, qual o objeto exergue pelo observador através daquele pontel.

Sutherland (SUTHERLAND et alli, 1974) sumaria dez diferentes algoritmos de determinação de visibilidade. (LOPES, 1993) cita outros sete e ainda mais outros existem, já que o problema de eliminação de superfícies escondidas é quase tão antigo quanto a computação gráfica.

O mais simples destes algoritmos é o Memorizador-Z (*Z-Buffer*). Este algoritmo foi introduzido na tese de doutoramento de Catmull (CATMULL, 1974) e é largamente implementado em circuitaria ("hardware") especializada.

Consiste em manter-se, além do espaço em memória reservado à imagem, também uma outra área equivalente onde se armazena a profundidade (ou coordenada Z) do ponto do objeto projetado em cada pontel da tela. Durante a matriciação de um ponto, a cor do pontel só será atualizada se a profundidade armazenada antes para aquele pontel for maior do que aquela do ponto recém calculado, isto é, se o novo ponto estiver mais próximo do observador. Caso contrário é ignorado. O memorizador z deve conter inicialmente o máximo valor de z em todos os pontos. Veja seu algoritmo na fig. 2.4.

A maior desvantagem deste algoritmo é que, já que apenas um valor de intensidade e profundidade é armazenado para cada pontel os problemas relativos à amostragem (serrilhamento) são de difícil solução. Outra desvantagem é a memória gasta para armazenamento das profundidades. É necessário alocar-se pelo menos 2 bytes para cada pontel para evitar-se problemas adicionais de serrilhamento.

```
• memorizador_z()
  • inicializa todo o memorizador-z com o máximo z (mais distante do observador)
  • Para cada objeto do cenário faça
    • Para cada pontel na projeção do objeto faça
      • calcula z do ponto projetado;
      • Se (z calculado < z anterior) então
        atualiza cor do pontel (memória de quadro);
        atualiza z do pontel (memorizador z);
fim
```

FIG. 2.4 - O Algoritmo Memorizador Z ("Z-Buffer")

A principal qualidade deste algoritmo é a sua grande simplicidade, o que o torna adequado para implementação em circuitaria. Não é necessária nenhuma ordenação nem comparações entre objetos. Dispondo-se de uma memória de quadro distribuída, pode-se fazer eficientes implementações paralelas.

A complexidade do algoritmo em função do número de objetos é $O(c^{\frac{1}{2}})$ pois o número médio de pontéis cobertos pela projeção de cada objeto decresce com o aumento do número de objetos (FOLEY et alli, 1990).

D) Tonalização: O Algoritmo de Interpolação de Normais

A tonalização consiste em determinar-se qual a cor do ponto da superfície visível em cada pontel da tela virtual (plano de projeção).

A tonalização consiste em determinar-se qual a cor do ponto da superfície visível em cada pontel da tela virtual (plano de projeção).

Existem diversos algoritmos de iluminação que podem ser aplicados no processo de tonalização. A escolha do modelo de iluminação é fator importante no realismo da imagem gerada.

Os modelos de iluminação utilizáveis na síntese por emulação (técnicas incrementais) são os chamados *modelos empíricos* (HALL, 1986):

- Constante (BOUKNIGHT, 1970);
- Interpolação de Intensidades (GOURAUD, 1971);
- Interpolação de Normais (PHONG, 1975).

A adoção de modelos transicionais ou analíticos exige técnicas como "ray tracing" recursivo ou radiosidade e se enquadram na classe de síntese por simulação.

O algoritmo de Phong, apesar de menos implementado em circuitaria do que o modelo de Gouraud devido à sua maior complexidade, permite a obtenção de imagens mais realísticas do que este último pois emula efeitos locais de reflexão especular da luz.

A fig. 2.5 apresenta um algoritmo implementando tonalização com modelo de Phong.

```

phong(t) # faz a tonalização do triângulo t
Para cada segmento horizontal seg faça
    normal_e = interpola_normal(seg[min], arest_esq);
    normal_d = interpola_normal(seg[max], arest_dir);
    arest_hor = cria_aresta(seg[min], seg[max]);

    Para cada ponto de seg faça
        normal_p = interpola_normal(p, arest_hor);
        cor(p) = eq_ilum(normal_p, luz, observ);
fim

```

FIG. 2.5 - Algoritmo de tonalização com interpolação de normais (Phong)

Este algoritmo determina a cor de um ponto da superfície de um objeto (polígono plano) através da aplicação da seguinte equação de iluminação em cada ponto:

$$I = k_d (\bar{N} \cdot \bar{L}) + k_s (\bar{R} \cdot \bar{V})^n + I_a \quad (2.1)$$

onde:

I	Intensidade do ponto amostrado no plano de imagem;
k_d	coeficiente de reflexão difusa do material (componentes R, G e B);
$\bar{N}$	vetor unitário normal à superfície do objeto;

k_s	coeficiente de reflexão especular do material;
$\bar{R}$	vetor unitário na direção de reflexão da fonte de luz;
$\bar{V}$	vetor unitário na direção do observador;
n	expoente que representa o polimento da superfície;
I_a	Intensidade da iluminação ambiente global;

A determinação do vetor normal de um ponto do objeto é feita pela interpolação linear das normais dos vértices do polígono. A normal num vértice de polígono é determinada pela média das normais de todas as faces que compartilham o vértice.

2.2.2. Efeitos Especiais: Sombras, Reflexões, Transparência e Refração

As técnicas básicas usadas em emulação geram imagens bastante simples. Outras técnicas mais sofisticadas devem ser empregadas conjuntamente para aumentar-se o grau de realismo das imagens geradas (AMANATIDES, 1987).

A) Sombras

Existem diversos métodos para a emulação de sombras que podem ser aplicados em algum ponto do dadoduto gráfico. Cada um apresenta vantagens e restrições (GOMES & VELHO, 1990):

a) Sombra Projetada: método bastante simples que modela a sombra como sendo a projeção do objeto do ponto de vista da fonte de luz. É adequado quando a sombra se projeta em uma única superfície plana (BLINN, 1988).

b) Volume de Sombra: o volume de sombra de um objeto em relação a uma fonte de luz é o conjunto dos pontos do cenário que não são visíveis por um observador naquela fonte de luz. Ao calcular-se a iluminação de um ponto verifica-se se ele se encontra no volume de sombra de algum objeto do cenário. Este algoritmo é mais genérico do que o anterior porém exige muito mais processamento e memória. É descrito em (CROW, 1977).

c) Algoritmo de Atherton-Weiler-Greenberg: este algoritmo cria um modelo constituído por recortes dos objetos do cenário que não são visíveis da fonte de luz. Ao rasterizar-se e tonalizar-se um polígono, leva-se em consideração se este é um "polígono de sombra" ou não. A principal restrição deste algoritmo é que só se aplica a cenários constituídos de objetos poligonais (ATHERTON, WEILER & GREENBERG, 1978).

d) Algoritmo de Mapeamento: o algoritmo de mapeamento trata as sombras como um problema de visibilidade a partir das fontes de luz. Esta visibilidade é resolvida usando

um método semelhante ao "memorizador-z". Ao calcular-se a iluminação da cena, os dados deste "memorizador" são usados para determinação das sombras. Este método foi proposto por Lance Williams (WILLIAMS, 1978).

e) Algoritmo de Appel: Durante a rasterização, este algoritmo traça um raio de sombra para cada fonte de luz para testar se há obstrução. Em caso positivo, é determinada a parte da linha de varredura afetada pela sombra. Para aumentar a eficiência do algoritmo, é criada uma lista de pares de polígonos relacionando quais polígonos causam sombra em quais. A implementação original deste algoritmo só é utilizável para modelamentos poligonais. Foi proposta em (APPEL, 1968) e posteriormente em (BOUKNIGHT & KELLEY, 1970).

B) Reflexões

Para a emulação de reflexões do ambiente nos objetos do cenário pode ser utilizado uma técnica denominada mapeamento de reflexão ou de ambiente (*environment mapping*).

Segundo esta técnica introduzida em (BLINN & NEWELL, 1976), faz-se um mapeamento, na superfície do objeto, de uma textura que é uma imagem do ambiente ao redor do objeto. Este mapeamento é feito considerando-se que o espaço de textura está na superfície de uma esfera que circunda o objeto. O ponto de intersecção do vetor de reflexão com esta esfera determina o ponto (e a cor) do mapa de textura a ser mapeado no objeto.

A esfera pode ser substituída por um cubo, simplificando a função de mapeamento e a criação das texturas.

Esta técnica apresenta a vantagem de poder incluir efeitos de reflexão difusa não facilmente disponíveis em "ray tracing". Por outro lado exige a confecção de muitos mapas de reflexão se é desejada uma cena realística. Efeitos de inter-reflexão de vários níveis ou complexos são difíceis de obter.

C) Transparência e Refração

Efeitos de transparência podem ser emulados usando algoritmos (NEWELL et alli, 1973) que rasterizam superfícies em ordem de profundidade invertida. Ao rasterizar uma superfície transparente, faz-se uma combinação de cores, permitindo que partes da imagem anteriormente "pintadas" possam ser vistas através da nova superfície. Este algoritmo não permite efeitos de refração.

Um trabalho desenvolvido posteriormente por Kay (KAY & GREENBERG, 1979) permite obter uma aproximação dos efeitos de refração. O algoritmo proposto faz várias

simplificações tais como considerar que os raios de luz sempre entram e saem dos objetos do cenário na direção paralela ao vetor normal do plano de projeção e que sempre percorrem uma distância fixa dentro de cada objeto transparente. Estas simplificações reduzem sobremaneira a demanda computacional do algoritmo porém prejudicam o efeito de realismo em cenas onde a transparência e refração geram efeitos complexos que devam ser modelados corretamente.

2.3. Síntese por Simulação

As técnicas de síntese por simulação são baseadas em fenômenos físicos reais. Entre estas técnicas destacam-se o "ray tracing" e a radiosidade.

O algoritmo de "ray tracing" simula (ou "rastrea") alguns dos raios de luz que transitam no espaço e chegam até o observador, formando a imagem. Este "rastreamento" dos raios de luz é feito segundo equações da Óptica (Lei de Snell, etc.).

A técnica de radiosidade, baseada na conservação de energia num ambiente fechado, originou-se de disciplinas ligadas à Termodinâmica e portanto também tem um embasamento físico.

2.3.1. "Ray Tracing"

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.

Além de se constituir em um algoritmo bastante simples e elegante para geração de imagens foto-realísticas, o algoritmo de "ray tracing" é utilizado também em várias outras aplicações, entre elas:

- Cálculo de volume, massa, centróide, centro de massa, área de superfície, momentos e produtos de inércia de sólidos em aplicações de CAD (DYKSTRA, 1989).
- Auxílio ao projeto na indústria óptica e de lentes (ROBB & PAWLOWSKI, 1990);
- Conversão entre modelamentos (por exemplo, de secções cônicas para octárvore ("octree")) (FRANKLIN & AKMAN, 1985);
- Cálculo dos fatores de forma em algoritmos de radiosidade para síntese de imagens e cálculo de transferência de calor (WALLACE et alli, 1989);

- Cálculo dos fatores de forma em algoritmos de radiosidade para síntese de imagens e cálculo de transferência de calor (WALLACE et alli, 1989);
- Análise de antenas, sonares e radares (PLUNKETT & BAYLEY, 1986);
- Modelamento molecular, sismologia, processamento de imagens e sinais (PASSMORE, 1987);
- Geração de hologramas por computador (STEIN et alli, 1992);
- Base para "benchmark" de computadores de várias classes (ROBB & PAWLOWSKI, 1990) e (MUUSS et alli, 1991);
- É freqüentemente usado como exemplo de aplicação para computadores paralelos por ser muito simples variar-se a granularidade de suas tarefas (MAY & SHEPHERD, 1987), (ZUFFO, 1993).

Sua capacidade de sintetizar imagens diretamente de objetos modelados segundo quase qualquer técnica de modelamento o torna um algoritmo ímpar e, às vezes, a única opção disponível para visualização direta de alguns modelos.

Todo processo de síntese de imagens resume-se a determinar a cor de cada pontel de uma imagem. O algoritmo de "ray tracing" faz esta determinação através do modelo de iluminação proposto por Whitted (WHITTED, 1980), (HALL & GREENBERG, 1983):

$$I = k_d \sum_{j \in \Omega} (\bar{N} \cdot \bar{L}_j) + k_s \sum_{j \in \Omega} (\bar{N} \cdot \bar{H}_j)^n \cdot I_j + k_r I_r + k_t I_t + I_a \quad (2.2)$$

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 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 2.2 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 contribuem para a formação da imagem vista por um observador. Apenas os raios que são refletidos por objetos e atingem o "olho" do observador 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 pontel da imagem. Seu primeiro passo é a determinação do ponto do cenário, visto através do pontel a processar, onde serão avaliadas as parcelas locais da iluminação (fig. 2.6).

FIG. 2.6 - 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 pontel 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 (vide seção 2.4.3) 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. 2.2) 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 um rotina muito semelhante àquela para determinação de visibilidade.

Nesta etapa do algoritmo já se pode determinar completamente a componente local da luz que determina a cor de um determinado pontel, incluindo suas parcelas difusa e especular. A imagem resultante deste cálculo para todos os pontéis é semelhante à obtida por algoritmos de emulação (seção 2.2) com 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 seguinte equação, 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. 2.7):

FIG. 2.7 - 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 e lançando-se um raio nesta direção (fig. 2.8).

FIG. 2.8 - Cálculo do raio transmitido

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 pontel. 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 pontel.

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. 2.9). 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), (SCHIMMITT, 1988), (KUCHKUDA, 1988) e (GLASSNER, 1989a). (SPEER, 1992) apresenta uma extensa coletânea de referências sobre "ray tracing".


```

ray_tracing()
  para cada pontel faça
    • calcule direção do raio primário (do observador ao pontel), prim ;
    • cor do pontel = 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. 2.9 - O algoritmo de "ray tracing"

2.3.2. Radiosidade

Introduzida na Computação Gráfica pelo trabalho de Cindy Goral (GORAL et alli, 1983), a radiosidade surgiu de modelos físicos de transferência de calor.

Esta técnica é baseada no equilíbrio do balanço de energia num ambiente fechado e na hipótese de que todos os processos de emissão e reflexão envolvidos são idealmente difusos. Assim, somente cenários com estas características são imageados realisticamente.

A radiosidade de uma superfície, isto é, a energia (luz) que a deixa, consiste da luz própria por ela emitida mais a luz refletida ou transmitida por ela. É usualmente modelada pela equação abaixo:

$$B_i = E_i + \rho_i \sum_{j=1}^{NP} B_j F_{ij}, \quad p/ 1 < i < N \quad (2.3)$$

onde: B Radiosidade: taxa total de energia saindo de uma superfície $\left(\frac{J}{s.m^2} \right)$;

E Emissão: a taxa de energia emitida por uma superfície (luz própria) ;

ρ Reflexividade: a fração da luz incidente numa superfície que é refletida de volta ao ambiente;

F Fator de Forma: a fração de energia que parte de uma superfície e atinge outra;

NP Número de superfícies discretas ou "retalhos" (*patches*) que compõem o cenário.

A aplicação da fórmula de radiosidade para cada superfície do ambiente produz um sistema de equações lineares (fig. 2.10) que, resolvido, indicará a radiosidade de cada superfície.

$$\begin{bmatrix} 1 - \rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1 - \rho_2 F_{22} & \cdots & -\rho_2 F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N F_{N1} & \cdots & \cdots & 1 - \rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix}$$

FIG. 2.10 - Sistema de equações para radiosidade

O parâmetro ρ , reflexividade, varia em função do comprimento de onda da luz incidente. Assim, para obter-se uma imagem colorida, o sistema da figura. 2.10 deve ser calculado e resolvido para cada reflexividade correspondente aos componentes de cor desejados, usualmente próximos ao vermelho, ao verde e ao azul.

A radiosidade é um método para determinar-se a cor final das superfícies numa cena. Um algoritmo qualquer de visualização (incremental ou "ray tracing", por exemplo) deverá ser usado para a geração da imagem a partir das informações de radiosidade, usualmente interpolando-se as cores para cada ponto das superfícies, como na técnica de Gouraud (GOURAUD, 1971).

Como somente a luz difusa é modelada pela técnica de radiosidade, os resultados obtidos são independentes da posição do observador. Desta forma, uma vez calculados os fatores de forma e resolvido o sistema de equações para determinação das radiosidades, pode-se gerar imagens de qualquer ponto de vista sem recorrer-se novamente à este algoritmo.

A técnica de radiosidade está sendo especialmente aplicada à Arquitetura onde os cenários têm predominantemente luz difusa e são fechados. Como as etapas de visualização através de algoritmos incrementais e de visibilidade são atualmente implementadas em circuitaria, pode-se fazer os chamados "passeios interativos" (*walk-throughs*) através de cenários realísticos cuja iluminação foi calculada previamente com o algoritmo de radiosidade.

Além da ausência de luz especular, dois outros problemas são enfocados pelas pesquisas em radiossidade: a resolução do sistema de equações lineares e a memória ocupada pelos fatores de forma.

Como a cada superfície é atribuída apenas um valor de radiossidade, é necessário que mesmo as superfícies planas sejam subdivididas para que se obtenha um efeito de sombreado realístico, notadamente nos pontos onde o gradiente de iluminação é grande. Assim, um cenário de complexidade média conterá um grande número de "retalhos". Cada retalho implica numa equação linear no sistema da figura 2.10. A cada par de retalhos corresponderá um fator de forma que precisa ser armazenado.

Michael Cohen (COHEN et alli, 1988) dá um exemplo da ordem de grandeza num cenário complexo (50.000 retalhos): supondo que cada fator de forma ocupe 4 bytes de memória (variável real de precisão simples) e que a matriz de coeficientes, geralmente esparça, tenha 90% de valores nulos:

- A matriz da figura 2.10 terá dimensão 50.000 x 50.000;
- O armazenamento dos fatores de forma não nulos exigirá quase 1Gbyte (~954 Mbytes) de memória.

Melhoramentos no algoritmo original permitem que se calculem as radiossidades progressivamente e os fatores de forma somente quando necessários, aliviando ambos os problemas.

Finalmente, deve-se mencionar que uma variação do algoritmo de "ray tracing" é usada com sucesso para cálculo dos fatores de forma (WALLACE et alli, 1989).

2.4. "Ray Tracing"

Atualmente a pesquisa sobre o algoritmo de "ray tracing" se faz em três áreas básicas:

- Aplicação do algoritmo para novas formas de modelamento geométrico;
- Técnicas para aumento de realismo;
- Técnicas de aceleração do "ray tracing".

Esta seção contém informações básicas e referências sobre as três áreas mencionadas. Maior destaque será dado às técnicas de aceleração do "ray tracing" já que a presente dissertação representa pesquisa nesta área e várias técnicas de paralelização do "ray tracing" são adaptações de métodos de aceleração do algoritmo.

As informações apresentadas a seguir também têm importância à medida em que se deseja fornecer subsídios para avaliação das restrições algorítmicas impostas por soluções paralelas para "ray tracing".

2.4.1. Modelamento Geométrico e Intersecções

Uma das mais importantes características do "ray tracing" é sua capacidade de gerar imagens a partir de objetos descritos segundo praticamente qualquer técnica de modelamento geométrico. A única condição para que se possa aplicar o algoritmo a um modelo qualquer é que seja possível determinar-se o(s) ponto(s) de intersecção de uma reta com o objeto em questão. Adicionalmente, para cálculo da iluminação, geralmente é necessária a determinação do vetor normal à superfície nestes pontos.

As técnicas por emulação e até a radiosidade geralmente exigem a "poligonalização" dos objetos, isto é, que sua representação passe a ser feita por meio de polígonos planos. Esta operação acarreta maior uso de memória e, dependendo do algoritmo, "facetamento" do modelo.

2.4.1.1. "Ray Tracing" no Modelamento CSG

O algoritmo de "ray tracing" aplicado a cenários modelados segundo a técnica CSG (Geometria de Sólidos Construtivos) será aqui detalhado por se tratar de técnica relativamente diferente daquela aplicada aos demais modelamentos geométricos.

Como se sabe (MÄNTYLÄ, 1988), objetos no modelamento CSG são representados por árvores binárias onde cada nó contém uma operação booleana regularizada e seus nós filhos, se houver, serão objetos primitivos (esfera, paralelepípedo, cilindro, cone, etc.) ou outros objetos compostos. Os nós-folha sempre serão objetos primitivos (fig. 2.11).

Para determinar-se o ponto de intersecção de um raio ($R(t) = R_o + t \cdot \bar{r}$) com um objeto composto pode-se usar o chamado diagrama de Roth (ROTH, 1982). Neste diagrama são marcados parametricamente os pontos de entrada e saída do raio em relação a cada objeto primitivo que forma o sólido composto (fig. 2.12).

FIG. 2.11 - Um sólido modelado segundo a técnica CSG

As linhas do diagrama são combinadas usando-se as respectivas operações de conjunto que eram aplicadas para formar o sólido em questão. Pontos internos ou externos são descartados, permanecendo apenas aqueles sobre a superfície dos objetos. A combinação final indicará os pontos de entrada e saída do raio no sólido composto, representados por valores de seu parâmetro t . O valor mínimo deste parâmetro indica o primeiro ponto de intersecção do raio com o objeto CSG composto.

O cenário inteiro pode ser definido como um único objeto modelado em CSG.

Existem diversas otimizações para esta técnica. Veja-se, por exemplo (HANRAHAN, 1989).

FIG. 2.12 - Diagrama de Roth

2.4.1.2. Cálculos de Intersecção e Vetor Normal para Esfera

A seguir será apresentado um algoritmo de intersecção de raio com esfera para que o leitor tenha noção da complexidade deste tipo de algoritmo (HAINES, 1989).

A esfera é um dos primitivos mais simples para este tipo de cálculo e por isso foi utilizada para teste das primeiras implementações de "ray tracing", além de servir como um bom envoltório (vide seção 2.4.3.1.1).

Para modelamentos mais complexos serão apenas citados artigos onde a referida técnica de intersecção é descrita em detalhes (seção 2.4.1.4). (HANRAHAN, 1989) apresenta uma coleção destas técnicas.

A geometria envolvida no cálculo de intersecção de raio com esfera está ilustrada à fig. 2.13.

FIG. 2.13 - Geometria da intersecção raio-esfera

A esfera será representada implicitamente por seu centro e raio:

$$(X - X_c)^2 + (Y - Y_c)^2 + (Z - Z_c)^2 - E_r^2 = 0 \quad (2.4)$$

onde: $E_c = [X_c \ Y_c \ Z_c]$ centro da esfera;
 E_r raio da esfera.

O raio de luz é representado na forma paramétrica:

$$R(t) = R_o + t \cdot \bar{r} \quad (2.5)$$

onde: R_o origem do raio;
 $\bar{r} = [r_x \ r_y \ r_z]$ vetor unitário na direção do raio.

O cálculo do ponto de intersecção de raio com esfera e a normal neste ponto pode ser resumido em 8 passos, descritos a seguir:

Passo 1) Descobrir se a origem do raio está dentro da esfera:

$$\begin{aligned} \vec{OC} &= E_c - R_o && \text{(vetor da origem do raio até o centro da esfera);} \\ d_{oc}^2 &= \vec{OC} \cdot \vec{OC} && \text{(comprimento, ao quadrado, de OC);} \end{aligned}$$

Se $d_{oc}^2 < E_r^2$, então a origem do raio está dentro da esfera e vai interceptá-la;
 senão a origem está fora e o raio pode não interceptar a esfera.

Passo 2) Achar o ponto do raio mais próximo do centro da esfera (P_m):

$$t_m = \vec{OC} \cdot \vec{r} \quad (\text{distância da origem do raio até o ponto } P_m);$$

$$P_m = R(t_m) \quad (\text{ponto do raio mais próximo do centro da esfera})$$

Passo 3) Se a origem do raio está fora da esfera e aponta em sentido contrário ao centro desta, então o raio não a intercepta (encerrar cálculos):

Se $t_m < 0$ o raio aponta no sentido contrário ao centro da esfera.

Passo 4) Senão, achar o quadrado da distância do ponto de intersecção P_{int} até P_m (t_a^2):

$$t_a^2 = E_r^2 - d^2 \quad (\text{distância de } P_m \text{ até } P_{int});$$

$$d^2 = d_{oc}^2 - t_m^2 \quad (\text{distância do centro da esfera até } P_m)$$

$$\therefore t_a^2 = E_c^2 - d_{oc}^2 + t_m^2$$

Passo 5) Se o valor de t_a é negativo, o raio não intercepta a esfera.

$t_a < 0$: raio não intercepta a esfera. Encerrar cálculos.

Passo 6) Senão, achar distância da origem do raio até o ponto de intersecção I:

$$t_i = t_m - t_a$$

Passo 7) Calcular as coordenadas do ponto de intersecção:

$$I = R(t_i) = [x_i \quad y_i \quad z_i] = [X_o + r_x \cdot t_i \quad Y_o + r_y \cdot t_i \quad Z_o + r_z \cdot t_i]$$

Passo 8) Calcular o vetor unitário normal à esfera, no ponto de intersecção:

$$\vec{n} = \left[\frac{(x_i - X_c)(y_i - Y_c)(z_i - Z_c)}{E_r} \right]$$

2.4.1.3. Estimativas do Número de Operações

Tomando como exemplo os oito passos do algoritmo de cálculo de intersecção e normal de esfera mostrado anteriormente, tem-se o seguinte número de operações para o pior caso (HAINES, 1989):

- Passo 1: 5 somas/subtrações e 3 multiplicações;
 Passo 2: 2 somas e 3 multiplicações;

- Passo 3: 2 comparações (1 se a origem está dentro da esfera);
 Passo 4: 2 somas/subtrações e 1 multiplicação;
 Passo 5: 1 comparação (0 se a origem está dentro da esfera);
 Passo 6: 1 soma/subtração e uma raiz quadrada;
 Passo 7: 3 somas e 3 multiplicações;
 Passo 8: 3 subtrações e 3 multiplicações.

Total (ESFERA): 16 somas/subtrações;
no pior caso 13 multiplicações;
 1 raiz quadrada;
 3 comparações.

No caso de intersecção com plano, o número de operações é um pouco menor. No entanto, deve-se notar que o plano é uma entidade geométrica infinita e portanto, por si só, de pouca utilidade para o modelamento:

Total (PLANO): 8 somas/subtrações;
no pior caso 9 multiplicações;
 3 comparações.

Polígonos planos (de n lados), não necessariamente regulares, são entidades geométricas bastante utilizadas em modelamento. No cálculo de intersecção de raio com polígono, em geral calcula-se a intersecção com seu plano base e depois verifica-se se o ponto de intersecção encontrado está no interior do polígono. Somando-se as operações anteriores com aquelas para determinação da equação do plano (vetor normal) e as necessárias para classificação do ponto de intersecção, chega-se aos seguintes valores, aproximadamente, para o pior caso:

Total (POLÍGONO - n vértices): $19 + 5n$ somas/subtrações;
no pior caso $14 + n$ multiplicações;
 $n - 1$ divisões;
 $3 + 5n$ comparações.

As quádricas (cilindros, cones, elipsóides, parabolóides, hiperbolóides, etc) são também uma família de sólidos bastante usada, cujo cálculo de intersecção é razoavelmente simples:

Total (QUÁDRICAS): 44 somas/subtrações;
no pior caso 53 multiplicações;
 3 divisões;

2 raízes quadradas;

4 comparações.

Note-se que para alguns objetos procedurais, como superfícies fractais por exemplo, é necessário usar-se algoritmos iterativos que requerem inúmeros passos para chegar-se aos pontos de intersecção com precisão aceitável. Estes algoritmos podem exigir até várias centenas de operações sendo muito mais custosos que as outras intersecções mostradas aqui.

2.4.1.4. Outros Modelamentos

Na tabela 2.1 são indicadas referências onde o leitor interessado poderá encontrar detalhes para implementação de algoritmos de intersecção de raio com objetos representados segundo as mais variadas técnicas de modelamento geométrico.

MODELAMENTO	REFERÊNCIA
<i>Esfera, Plano, Polígonos (Implícitos)</i>	(HAINES, 1989)
<i>Geometria de Sólidos Construtivos (CSG)</i>	(ROTH, 1982)
<i>Fractais</i>	(KAJIYA, 1983)
<i>Superfícies Paramétricas</i>	(KAJIYA, 1982)
<i>Campos Escalares</i>	(SABELLA, 1988)
<i>Campos de Alturas (Height Fields)</i>	(MAX, 1981)
<i>Superfícies B-Spline</i>	(SWEENEY & BARTELS, 1986)
<i>Superfícies de Revolução</i>	(BURGER & GILLIES, 1989)
<i>Superfícies de Steiner (Bézier Biquadrática)</i>	(SEDERBERG & ANDERSON, 1984)
<i>Superfícies Bicúbicas</i>	(PULLEYBLANK & KAPENGA, 1987)
<i>Superfícies Algébricas</i>	(HANRAHAN, 1983)

TAB. 2.1 - Referências sobre "ray tracing" para diversos tipos de modelamento geométrico

2.4.2. Técnicas para Aumento do Realismo

Apesar do realismo alcançado pela técnica básica de "ray tracing" ser superior ao atingível por muitas outras metodologias, a pesquisa de melhoramentos no algoritmo para conseguir-se novos efeitos também tem sido intensa.

A tabela 2.2 apresenta referências que descrevem algoritmos para aumentar o foto-realismo das imagens geradas por "ray tracing", que o leitor interessado é convidado a consultar.

EFEITO	REFERÊNCIA
<i>Chamas (fogo)</i>	(KAJIYA & HERZEN, 1984)
<i>Foco e Profundidade de Campo (Lentes)</i>	(POTMESIL & CHAKRAVARTY, 1981)
<i>Miragens</i>	(BERGER, TROUT & LEVIT, 1990)
<i>Cor de Interferência (Bolha de Sabão)</i>	(DIAS, 1991)
<i>Cáusticas (Piscina)</i>	(INAKAGE, 1986)
<i>Pelagem, Pelúcia</i>	(KAJIYA & KAY, 1989)
<i>Nuvens</i>	(KIRK, 1987)
<i>Borrão de Movimento</i>	(POTMESIL & CHAKRAVARTY, 1983)
<i>Transluc., Sombras Difusas, Refl. Nubladas</i>	(AMANATIDES, 1984)
<i>Efeitos de Polarização da Luz</i>	(WOLFF & KURLANDER, 1990)
<i>Penumbra</i>	(COOK et alli, 1984)
<i>Água, Perturbação da Normal</i>	(WYVILL, PEARCE & WYVILL, 1986)

TAB. 2.2 - Referências sobre técnicas para aumento de realismo em "ray tracing"

2.4.3. Técnicas de Aceleração

Apesar de todas as características e aplicações interessantes deste algoritmo vistas anteriormente, sua talvez única desvantagem, a alta demanda de processamento, o manteve afastado de muitas potenciais aplicações. Desta forma, a pesquisa em técnicas de aceleração do "ray tracing" iniciou-se muito cedo e hoje já se dispõe de muitas metodologias com esta finalidade.

Whitted reporta em seu artigo clássico (WHITTED, 1980) que, para cenários complexos, até mais de 95% do tempo total de processamento pode ser necessário para o cálculo de intersecções de raios com os objetos do cenário. Desse modo, a maioria das técnicas de aceleração atuam no sentido de otimizar a etapa de cálculo de intersecção do algoritmo.

Uma noção mais exata da dimensão do problema de cálculo de intersecções em "ray tracing" pode ser adquirida através de um exemplo. Supondo-se uma cena com as seguintes características cotidianas:

- resolução da imagem: 1024 x 768 pontéis;
- probabilidade de geração de raios de reflexão: 50%;
- probabilidade de geração de raios de transmissão: 10%;
- profundidade média da árvore de raios: 5 níveis;
- sem super-amostragem (1 raio primário por pontel);

- cenário com 500 polígonos.

O uso do chamado "ray tracing exaustivo" (sem otimizações) vai requerer o cálculo de aproximadamente 900 milhões de intersecções para a geração desta imagem. Para polígonos, objetos simples, cada intersecção requer, em média, aproximadamente 25 operações de ponto flutuante, resultando num total de 22,5 bilhões de operações deste tipo. Supondo um computador moderno com capacidade para 10 MFLOPS, a geração desta imagem comum exigiria aproximadamente 40 minutos apenas para cálculo de intersecções. Esta grandeza de tempo inviabiliza a geração de animações muito longas e descarta qualquer aplicação interativa em tempo real.

As técnicas de aceleração desenvolvidas permitem que este tempo seja reduzido em ordem de grandeza, atualmente viabilizando o "ray tracing" para muitas aplicações antes inconcebíveis. Algumas técnicas tornam o tempo de processamento independente do número de objetos na cena fazendo com que este algoritmo seja, para cenas complexas, até mais rápido que técnicas convencionais consideradas rápidas.

A seguir serão descritas sucintamente as principais técnicas de aceleração de "ray tracing", classificadas segundo o esquema da fig. 2.14. Outras técnicas simples, que permitem pequenos ganhos são apresentadas em (GLASSNER, 1990) e (ARVO, 1991).

FIG. 2.14 - Classificação das técnicas de aceleração de "ray tracing"

Técnicas envolvendo coerência temporal também podem ser classificadas como técnicas de aceleração, porém seu uso é restrito a animações. (BADT, 1988), (GLASSNER, 1988), (CHAPMAN et alli, 1990), (MURAKAMI & HIROTA, 1990), (GRÖLLER & PURGATHOFER, 1991), (MAISEL & HEGRON, 1992) e (MAUREL et alli, 1993) são referências nesta área.

O chamado "*ray tracing*" parametrizado (SÉQUIN & SMYRL, 1989), (MARTINS & SANTOS, 1993) permite que, após a geração de uma imagem matriz, obtenham-se versões com alterações dos parâmetros ópticos (e só destes) gastando-se apenas uma pequena fração do tempo do "ray tracing" convencional usado na primeira imagem. (SANTOS, 1994a) e (SANTOS, 1994c) apresentam uma alternativa que permite alterar também alguns parâmetros geométricos do cenário: a posição das fontes de luz e a normal das superfícies (textura). O "ray tracing" parametrizado não é uma técnica de aceleração de uso genérico mas pode agilizar enormemente o ajuste de materiais e iluminação numa cena.

As técnicas de paralelização naturalmente também enquadram-se como metodologias de aceleração do algoritmo e são objeto de estudo detalhado deste trabalho.

2.4.3.1. Aceleração do Cálculo de Intersecções

As técnicas baseadas em acelerar o cálculo de intersecções procuram diminuir o custo computacional médio destes cálculos.

2.4.3.1.1. Envoltórios

No artigo introdutório do "ray tracing" moderno (WHITTED, 1980), Whitted já advogava o uso de volumes envolventes ("*bounding volumes*") ou envoltórios ("*extent*" , "*enclosure*") como forma de aceleração do "ray tracing". O envoltório proposto era a esfera por ser a forma mais simples para cálculo de intersecção com raio.

O uso de objetos simples, auxiliares, envolvendo outros mais complexos que pertencem ao cenário, permite acelerar o processamento de "ray tracing" porque pode-se calcular primeiro a intersecção do raio com o envoltório simples e, no caso de não haver intersecção, evitar-se este cálculo para o objeto mais complexo em seu interior. No caso de detectar-se um intersecção com o envoltório, será necessário repetir-se o cálculo para o objeto envolvido, tornando o cálculo de intersecção com o envoltório uma sobrecarga. Na média, o uso desta técnica será amplamente compensadora já que apenas pequena parcela dos testes resulta em detecção de intersecção.

Apesar de o envoltório acelerar o cálculo de intersecção, ele não proporciona nenhuma redução no número total de intersecções a serem efetuadas para síntese da cena. Assim, o tempo total de execução pode ser reduzido de um fator constante, porém a complexidade linear do algoritmo exaustivo em relação ao número de objetos do cenário é mantida (ARVO & KIRK, 1989).

Um bom envoltório é caracterizado por dois fatores:

- ajuste às formas do objeto envolvido;
- simplicidade para cálculo de intersecção com raio.

Em geral tem-se um compromisso entre estes dois fatores. Além das esferas, outras formas propostas para servirem de envoltório foram paralelepípedos com faces ortogonais aos eixos principais (RUBIN & WHITTED, 1980) e cilindros (WEGHORST et alli, 1984). No entanto, apesar de não imporem grande sobrecarga para cálculo de intersecções, estes envoltórios em geral não se ajustam bem aos objetos, resultando em muitos testes com "falso-positivo", diminuindo a eficiência da técnica.

O custo total dos testes de intersecção para um dado objeto é dado por (WEGHORST et alli, 1984):

$$C_{tot} = b.C_b + i.C_{int} \quad (2.6)$$

onde: C_{tot} custo total da intersecção;
 b número de vezes que o envoltório é testado;
 C_b custo de intersecção de um raio com o envoltório;
 i número de raios que atingem o envoltório;
 C_{int} custo de intersecção de um raio com o objeto.

Na equação acima, b e C_{int} são constantes para uma dada cena e C_b e i deverão ser manipulados para obter-se custo ótimo. Como não se pode determinar, a priori, o número de raios que interceptarão um envoltório, uma aproximação pode ser feita considerando-se este número proporcional ao volume do envoltório (WEGHORST et alli, 1984) ou à área de sua superfície (ARVO & KIRK, 1989), que procurar-se-á minimizar.

Um envoltório com melhor ajuste aos objetos, mantendo simples o cálculo de intersecção, foi proposto por Timothy Kay e James Kajiya (KAY & KAJIYA, 1986). Este envoltório consiste na intersecção de nacos ("slabs"), semi-espacos contidos entre dois planos, com orientação previamente selecionada (fig. 2.15).

FIG. 2.15 - Objeto com envoltório de intersecção de nacos

A vantagem destes envoltórios é que podem ser arbitrariamente ajustados aos objetos, bastando para isto aumentar o número de planos que delimitam o volume. Selecionando previamente a orientação destes planos pode-se pré-calculer alguns fatores que serão repetidos inúmeras vezes nos cálculos de intersecção, aumentando sua eficiência.

2.4.3.1.2. Técnicas para Modelamentos Específicos

Estas técnicas se confundem com os métodos padrão para a visualização direta dos modelos aos quais se aplicam. Por sua falta de generalidade têm aplicação bastante restrita e serão apenas indicadas aqui referências onde se podem encontrar detalhes de implementação para vários tipos de modelagem:

- *Superfícies B-Spline*: (SWEENEY & BARTELS, 1986);
- *Superfícies Algébricas*: (HANRAHAN, 1983);
- *Superfícies Bézier Biquadráticas*: (SEDERBERG & ANDERSON, 1984);
- *Superfícies Fractais, prismas e superfícies de revolução*: (KAJIYA, 1983);
- *Superfícies Paramétricas*: (JOY & BHETANABHOTLA, 1986), LISCHINSKI & GONCZAROWSKI, 1990);
- *Enumerações Espaciais ("voxels")*: (YAGEL et alli, 1992);
- *Modelos CSG (Geometria de Sólidos Construtivos)*: (BRONSVOORT et alli, 1984), (GERVAUTZ, 1986), (BOUATOUCH et alli, 1987);
- *Superfícies Bicúbicas (β -Splines, Bézier e Hermite)*: (LEVNER et alli, 1988).

2.4.3.2. Redução do Número de Intersecções

Estas técnicas atuam no sentido de diminuir o número total de raios que devem ser lançados e interseccionados com o ambiente. Isto inclui tanto raios primários quanto raios secundários (reflexões e refrações) e raios de sombra.

A otimização mais imediata desta classe é o uso de uma rotina de lançamento de raios ligeiramente modificada para os raios de sombra. O processamento destes para uma fonte de luz pode ser interrompido imediatamente quando se encontra uma intersecção com qualquer objeto pois, ao contrário dos outros tipos de raio, não interessa saber qual o objeto mais próximo da origem mas apenas se há ou não algum objeto na trajetória do raio de sombra.

Outras técnicas mais sofisticadas são apresentadas a seguir.

2.4.3.2.1. Particionamento do Espaço 3D

As metodologias de particionamento do espaço 3D atuam da seguinte forma: inicialmente, numa fase de pré-processamento, o espaço do cenário é subdividido em regiões disjuntas denominadas células ou "voxels". A cada célula é associada uma lista dos objetos cuja superfície intercepta as fronteiras da célula ou nela estão totalmente contidos.

Ao iniciar-se o lançamento de raios, determina-se qual célula será penetrada pelo raio. Somente os objetos constantes da lista daquela célula são testados para intersecção com aquele raio. Havendo intersecções, a mais próxima da origem do raio é selecionada e encerra-se o processamento de intersecções para aquele raio. Em caso contrário, a próxima célula na trajetória do raio é determinada e o processamento continua de forma análoga até a detecção de uma intersecção ou a saída do raio do espaço do cenário.

A diminuição do número de intersecções neste algoritmo se dá por duas razões:

- a cada passo, somente um pequeno subconjunto de objetos, potencialmente na trajetória do raio, são testados, ignorando-se os demais.
- os objetos são naturalmente testados aproximadamente em ordem de proximidade da origem do raio, encerrando-se o processamento ao encontrar-se intersecção válida.

Cuidado deve ser dado ao fato de que uma intersecção encontrada só é válida se o ponto de intersecção estiver contido dentro da célula testada. Caso contrário, pode-se desconsiderar objetos na trajetória do raio, mais próximos de sua origem, caracterizando

uma falsa intersecção como aquela da célula 4 (objeto C), determinada erroneamente em cálculo efetuado na célula 3 (fig. 2.16).

FIG. 2.16 - Fragmentação e falsa intersecção

Outro ponto a considerar é o cálculo de repetidas intersecções de um objeto com o mesmo raio, em células diferentes (fig. 2.16 - objeto A é testado nas células 1 e 2). Este fenômeno (*fragmentação*) pode ser evitado armazenando-se junto a cada objeto (em uma "caixa postal" - *mailbox*) uma identificação do último raio com o qual foi testado e o ponto de intersecção, se houver. Antes de calcular-se uma intersecção, a caixa postal do objeto é verificada, aproveitando-se o cálculo anterior se for o caso (ARNALDI et alli, 1987).

O particionamento do espaço pode ser feito de forma uniforme ou adaptativa. No primeiro caso, a subdivisão é feita sem levar em conta a geometria ou o posicionamento dos objetos no cenário, causando grande fragmentação e má distribuição dos objetos pelas células. Com o particionamento adaptativo, pode-se controlar o processo de subdivisão de modo a evitar-se, pelo menos parcialmente, estes problemas. Por outro lado, a travessia no espaço uniformemente particionado geralmente é mais eficiente.

O particionamento adaptativo pode ser feito segundo estruturas em octárvore ("octree") (GLASSNER, 1984), BSP tree (JANSEN, 1986) ou ainda EN-tree (HSIUNG & THIBADEAU, 1992). Cada metodologia apresenta aspectos diferentes quanto a capacidade de adaptação ao cenário e desempenho da fase de travessia dos raios.

2.4.3.2.2. Árvore de Envoltórios

O agrupamento de envoltórios por outros envoltórios e destes por outros dá origem à chamada árvore hierárquica de envoltórios ("*hierarchical tree of extents*") ou, simplesmente, hierarquia de envoltórios. O nó na raiz da árvore engloba todos os outros e define o espaço do cenário (fig. 2.17).

FIG. 2.17 - Hierarquia de envoltórios

Na síntese de imagens por "ray tracing" usando aceleração por hierarquias de envoltórios, ao disparar-se um raio, inicialmente calcula-se sua intersecção com o nó da raiz da árvore. Se não for detectada intersecção, encerra-se o processamento deste raio ("cor-de-fundo"). Caso contrário, verifica-se a intersecção deste raio com os nós filhos da raiz e assim sucessivamente até atingir-se nós-folha que contêm objetos.

Como mencionado anteriormente, o uso de envoltórios simples mantém a complexidade do "ray tracing" linearmente proporcional ao número de objetos do cenário. Porém, sua organização segundo uma hierarquia faz com que esta complexidade se reduza a uma proporção logarítmica, acelerando ainda mais o "ray tracing". Esta aceleração se dá porque, ao detectar-se a não intersecção com um envoltório, todos os sub-envoltórios e objetos contidos no envoltório mais externo podem ser descartados como candidatos a intersecção, poupando muitos testes de intersecção.

Para uma determinada cena, muitas árvores diferentes podem ser construídas, variando-se a escolha de agrupamentos. O tempo de processamento do "ray tracing" é bastante dependente da forma como a árvore é construída. Para uma cena simples, o tempo para a geração da imagem pode facilmente variar por um fator de 50 vezes devido apenas a diferentes escolhas de árvore (GOLDSMITH & SALMON, 1987).

A construção de boas árvores de envoltórios é, dessa forma, fator determinante para o sucesso desta técnica de aceleração. Weghorst (WEGHORST et alli, 1984) sugere que se use a própria hierarquia/ordenação introduzida pelo usuário ao criar um cenário, já que esta naturalmente agruparia objetos próximos. No entanto, como observou Goldsmith (GOLDSMITH & SALMON, 1987) hierarquias criadas durante o modelamento geralmente têm muitos ramos por nível, diminuindo os ganhos ao se realizar a poda de cada ramo.

Uma proposta que parece oferecer bons resultados se baseia em heurística que procura sempre minimizar a área dos envoltórios criados ou expandidos (GOLDSMITH & SALMON, 1987). A ordem em que os objetos são selecionados para inserção na árvore sendo construída também influencia o desempenho da hierarquia. Os melhores resultados foram obtidos quando esta ordem é gerada aleatoriamente. A seleção de diversas tentativas aleatórias pode ser tentada já que esta fase de pré-processamento, realizada uma única vez, pode proporcionar grande economia de tempo durante os testes de intersecções, realizados milhões de vezes.

Kay e Kajiya (KAY & KAJIYA, 1986) sugerem que se armazene numa fila de prioridade ("*heap*") os vários envoltórios irmãos para os quais detectou-se intersecção, na ordem de maior proximidade com a origem do raio. Os cálculos de intersecção com estes envoltórios serão realizados segundo esta ordem, com o auxílio da fila de prioridade.

Desta forma, pode-se encerrar a procura da intersecção tão logo se encontre uma, com um objeto, que esteja mais próxima da origem do raio do que o envoltório ocupando a cabeça da fila de prioridade. Uma intersecção só pode ser considerada válida neste caso (fig. 2.18) ou quando a fila de prioridade estiver vazia.

FIG. 2.18 - Intersecções com envoltórios

As metodologias de aceleração por hierarquias de envoltórios, são, de certa forma, duais àquelas de particionamento do espaço. Enquanto estas últimas selecionam *regiões do espaço* baseadas em *conjuntos de objetos* as primeiras selecionam *conjuntos de objetos*, baseadas em *regiões do espaço*. Outro aspecto, é que as técnicas de particionamento espacial atuam de forma "top-down", partindo do cenário completo e recursivamente dividindo mais e mais o espaço 3D, enquanto o particionamento de objetos pode partir de objetos individuais que vão sendo agregados em conjuntos e estes a outros, sucessivamente, segundo uma estratégia "bottom-up".

Quanto ao desempenho das implementações baseadas em hierarquias de envoltórios, comparadas àquelas que usam particionamento adaptativo do espaço, pode-se dizer que para cada tipo de cena, uma ou outra terá desempenho superior. De qualquer forma, o desempenho geral é influenciado por vários fatores como complexidade dos envoltórios e do tipo de modelagem usada no cenário. (KAY & KAJIYA, 1986) reportam experimentos com sua implementação de hierarquias de envoltórios onde foram obtidos tempos 2.6 vezes mais rápidos que os mencionados em (GLASSNER, 1984) que implementa particionamento espacial por octárvore ("octree").

As técnicas baseadas em octárvore (particionamento adaptativo) testam muito mais intersecções com polígonos do que as baseadas em hierarquias de envoltórios, devido ao fenômeno da fragmentação (objetos "cortados", por fronteiras e, portanto, duplicados em mais de uma sub-região). Por outro lado, estas últimas fazem mais cálculos de intersecção com envoltórios do que o número de células atravessadas nas metodologias por octárvore. Acompanhe-se a fig. 2.19 para uma explicação deste fato. No exemplo ilustrado nesta figura, está representada uma hierarquia em árvore binária.

Supondo-se uma divisão semelhante do mesmo cenário para particionamento do espaço, e que o primeiro objeto interceptado esteja na célula/envoltório número 3, tem-se que o particionamento de objetos testa 7 envoltórios (15, 14, 13, 10, 9, 4 e 3, nesta ordem) enquanto o particionamento do espaço testa apenas 3 células (1, 2 e 3) para obter o mesmo resultado (SCHERSON & CASPARY, 1987).

FIG. 2.19 - Intersecções nos particionamentos de objetos e de espaço

2.4.3.2.3. Cache de Sombras / Cache da Árvore de Raios

Técnica bastante simples, introduzida por Eric Haines (HAINES, 1986), o cache de sombras consiste em, ao lançar-se raios de sombra, armazenar-se para cada fonte de luz e nível de recursão, qual o primeiro objeto (se algum) que causa sombra em relação à cada fonte de luz, formando uma memória cache de objetos que causam sombra.

Desta forma, ao lançar-se um raio de sombra em direção a uma dada fonte de luz, o primeiro objeto a ser selecionado para teste de intersecção deve ser aquele armazenado na posição do cache correspondente àquela fonte de luz e nível de recursão na árvore em que se esteja. Como o processamento de raios de sombra pode ser imediatamente interrompido ao encontrar-se qualquer objeto entre a fonte de luz e o ponto iluminado, grande parte dos raios de sombra serão interceptados com somente um objeto, aquele indicado no cache.

Esta técnica, fazendo uso da coerência de objetos, permite que se encontre mais rapidamente um objeto entre a fonte de luz e o ponto de intersecção, reduzindo o número de intersecções testadas e acelerando, desta forma, o algoritmo. O armazenamento desta informação para vários níveis de recursão, permite melhor aproveitamento dos dados armazenados, melhorando a taxa de acertos do cache.

O ganho obtido com esta técnica é relativamente restrito pois somente acelera o processamento dos raios de sombra, porém pode ser significativo em caso de cenários com muitas fontes de luz.

Sua implementação é simples e introduz pouca sobrecarga.

Pode-se estender a idéia desta técnica para toda a árvore de raios, tirando-se proveito da coerência de raios: se, ao traçar-se cada raio da árvore de um pontel, armazenar-se os objetos interceptados, pode-se, ao traçar-se a árvore do pontel seguinte, usar-se a informação armazenada para otimizar, da mesma forma que o cache de sombras, o número de intersecções calculadas. Como a árvore de raios de pontéis próximos tende a ser parecida (coerência de raios) esta técnica deveria apresentar resultados positivos (HANRAHAN, 1986). No entanto, a extensão da técnica, para outros raios que não os de sombra, exige que se crie e mantenha estruturas de dados adicionais já que, para outros raios, é importante determinar-se qual objeto é interceptado primeiro e não apenas se existe intersecção. A manutenção destas estruturas pode mostrar-se demasiadamente custosas (SPEER et alli, 1985).

2.4.3.2.4. Memorizador de Itens (Item Buffer)

A técnica de memorizador de itens (WEGHORST et alli, 1984) procura explorar a coerência de imagem disponível aos raios primários. É aplicada como uma etapa de pré-processamento ao algoritmo de "ray tracing".

Basicamente, esta fase de pré-processamento executa os mesmos passos do algoritmo memorizador-z ("z-buffer") (CATMULL, 1974), porém, ao invés de armazenar-se a cor do objeto na memória de quadro, registra-se um índice indicativo do objeto visível

em cada pontel. O algoritmo de "ray tracing" a seguir, simplesmente substitui os testes de intersecção para os raios primários por uma operação de acesso ao objeto indexado no memorizador de ítems, economizando assim, muitos testes de intersecção.

O algoritmo memorizador-z é bastante rápido fazendo com que a sobrecarga desta fase de pré-processamento seja facilmente compensada pelos ganhos posteriores. Estes ganhos serão tanto maiores quanto o forem o número de objetos no cenário e a resolução da imagem (número de raios primários) calculada.

2.4.3.3. Redução do Número de Raios

As técnicas descritas nesta seção procuram reduzir o número total de raios lançados para sintetizar uma imagem do cenário, acelerando o algoritmo.

2.4.3.3.1. Controle Adaptativo da Profundidade da Árvore de Raios

A cada nível que se desce na árvore de raios, menor é a contribuição dos raios para a cor final do pontel. Isto se deve ao fato de que as contribuições dos raios refletidos e transmitidos são escaladas por coeficientes (k_s e k_t , respectivamente) menores que 1. A multiplicação acumulativa destes coeficientes faz com que a contribuição dos raios em níveis muito profundos da árvore sejam desprezíveis visualmente ou até mesmo escapem da precisão de cores da memória de quadro utilizada.

Um primeiro passo para acelerar o algoritmo de "ray tracing" é limitar a profundidade máxima da árvore. Tipicamente escolhe-se um valor igual a 5 níveis, obtendo-se bons resultados.

Uma técnica um pouco mais sofisticada nesta mesma direção é o chamado controle adaptativo da profundidade da árvore de raios (HALL et alli, 1983). Nesta técnica, ao invés de fixar-se a priori uma profundidade máxima para a árvore de raios, fixa-se uma contribuição mínima para os raios. Se a produtória dos coeficientes a serem aplicados à contribuição de um dado raio for menor que um limiar estabelecido, o raio não é lançado. Desta forma, em ambientes muito reflexivos, pode-se processar a árvore de raios até níveis mais profundos, evitando-se isto onde não seja necessário.

Esta técnica adaptativa produz resultados apreciáveis, mesmo em ambientes com muitas superfícies reflexivas ou transparentes e é bastante simples de implementar-se, produzindo pouca sobrecarga de processamento ou memória.

Naturalmente, até mesmo o mais simples algoritmo de "ray tracing" não deve disparar raios de reflexão para objetos totalmente difusos ($k_s = 0$) ou raios de transmissão para objetos opacos ($k_t = 0$).

2.4.3.3.2. Super-amostragem Adaptativa

Uma das maneiras mais tradicionais de atacar-se o problema de serrilhamento (*aliasing*) em imagens geradas por "ray tracing" é o uso de super-amostragem. Segundo esta metodologia, dispara-se mais de um raio por pontel, obtendo-se assim uma estimativa mais precisa da(s) cor(es) que seria(m) vista(a) através de um dado pontel. Dispara-se por exemplo 9 raios através de um pontel (fig. 2.20) e a cor final deste é calculada através da média das cores resultantes destes raios (GLASSNER, 1989b).

Uma das grandes desvantagens da super-amostragem é o aumento do número de raios gerados e, conseqüentemente, do número de intersecções a calcular.

FIG. 2.20 - Super-amostragem

Uma forma de reduzir a sobrecarga de processamento imposta por esta técnica é usar a chamada super-amostragem adaptativa, com a qual só se disparam muitos raios nas regiões da imagem onde isto é necessário. Iniciando-se a super-amostragem com 5 raios (um em cada canto do pontel e outro em seu centro) pode-se recursivamente aplicar a técnica a subdivisões do pontel, determinadas pela comparação das cores dos raios lançados. Nas regiões em que a diferença entre as cores retornadas for maior que um certo limiar, ocorre nova super-amostragem (fig. 2.20). A cor final é uma composição ponderada das cores obtidas por todos os raios lançados.

O uso da super-amostragem adaptativa otimiza o lançamento de novos raios, acelerando a geração da imagem.

A amostragem estocástica (COOK, 1986) pode ser usada para o mesmo fim, eliminando também efeitos colaterais da amostragem uniforme.

2.4.3.3.3. Seleção de Pontéis para "Ray Tracing"

A técnica de seleção de pontéis para "ray tracing" (*pixel-selected ray tracing*) (AKIMOTO et alli, 1991) procura acelerar a geração da imagem fazendo um sub-amostragem adaptativa da cena. Aproveitando a coerência de imagem esta técnica usa "ray tracing" convencional apenas para o cálculo de alguns pontéis da imagem. Os pontéis restantes são calculados por interpolação ou por "ray tracing" acelerado (coerência de raios).

O algoritmo inicialmente faz uma sub-amostragem da imagem (tipicamente amostrando de 4 em 4 pontéis). A cor destes pontéis iniciais é calculada por "ray tracing" convencional (fig. 2.21).

FIG. 2.21 - Seleção de Pontéis para "Ray Tracing"

A seguir, é verificada a homogeneidade dos quatro pontéis calculados de cada região quadrada da imagem (pontéis de A a D na fig. 2.21). A homogeneidade é classificada em quatro níveis:

Nível 0) As árvores de raios dos pontéis contém objetos diferentes ou seja, a trajetória dos raios não é semelhante.

Nível 1) As trajetórias dos raios são semelhantes porém as sombras são diferentes.

Nível 2) As trajetórias dos raios e as sombras são semelhantes, porém os objetos têm texturas e/ou as diferenças de intensidade dos pontéis é muito grande.

Nível 3) As trajetórias dos raios e as sombras são semelhantes. Não existem objetos com mapeamento de textura e as diferenças de intensidade dos pontéis são pequenas.

A seguir, calcula a intensidade do pontel do centro do quadrado (pontel E) usando uma das seguintes técnicas, de acordo com o nível de homogeneidade verificado:

A) "Ray tracing" convencional: a homogeneidade nível 0 indica que existem bordas de objetos ao redor dos quatro pontéis iniciais, impossibilitando a previsão da trajetória dos raios do pontel central.

B) Cálculo de intensidade com "ray tracing" acelerado: esta técnica, na qual se aproveita as informações de trajetória dos raios, deve ser empregada quando se verifica homogeneidade níveis 1 e 2. Para nível 1 deve-se determinar os pontos exatos de intersecção para cálculo correto de sombras. Se a homogeneidade é de nível 2 deve-se usar "ray tracing" acelerado pois a cor dos pontéis vizinhos tende a ser bastante diferente do central (casos de textura ou reflexão especular, por exemplo).

C) Cálculo de intensidade com interpolação: usada somente em caso de homogeneidade nível 4. Este nível indica que provavelmente se trata de uma região da imagem que mostra uma parte de um único objeto, com variação suave de intensidade, permitindo o uso de simples interpolação.

Se o pontel central não foi calculado com "ray tracing" convencional, deve-se repetir os mesmos procedimentos (classificação de homogeneidade e cálculo do pontel central) para os quatro pontéis formando um losango à direita do quadrado examinado (pontéis B, E, D e F da fig. 2.21). Estas etapas são realizadas recursivamente até que todos os pontéis da região tenham sido calculados por algum método.

A seguir, para detectar-se erros de avaliação devido a presença de objetos com arestas ou pontas agudas, recalcula-se por "ray tracing" convencional, qualquer pontel em vértice de região que tenha sido calculado por interpolação e tenha intensidade diferente do pontel do centro da região. Se a intensidade recalculada for diferente da anterior, as homogeneidades das três regiões as quais pertence o pontel devem ser reavaliadas e seus pontéis centrais recalculados se necessário. Este procedimento deve também ser aplicado recursivamente.

Retorna-se a seguir para os passos anteriores, até que toda a imagem esteja calculada.

Resultados experimentais desta técnica mostraram aumento de velocidade de 2 até 9 vezes sobre o algoritmo convencional (aquele sem sub-amostragem). O desempenho da técnica varia em função dos parâmetros P e Q :

- P : passo da amostragem inicial.
- Q : limiar de diferença de intensidade de cores.

Os valores $P = 4$ e $Q = 16$ (para faixa de 0 a 255) mostraram bons resultados.

A velocidade do algoritmo se mostrou praticamente insensível ao parâmetro T porém o erro da imagem gerada em relação a que seria obtida com "ray tracing" convencional aumenta proporcionalmente a este parâmetro, especialmente se P é grande. Já o parâmetro Q mostrou grande influência na velocidade do algoritmo, com variação diretamente proporcional a esta, e também no erro médio da imagem (também com variação diretamente proporcional).

2.4.3.3.4. Raios Generalizados

As técnicas que usam raios generalizados substituem a noção tradicional de raio (reta ou raio de espessura infinitesimal) por outras entidades como feixes (*beams*) cones ou lápis (*pencils*).

A maioria destas técnicas além de servir para reduzir o número de raios necessário à geração da imagem, também permitem a obtenção de efeitos especiais como translucência, sombras difusas (*fuzzy shadows*) e reflexões nubladas (*dull reflections*).

Dentro desta categoria foram propostas basicamente 3 técnicas: traçado de feixes (*beam tracing*) (HECKBERT & HANRAHAN, 1984), traçado de cones (*cone tracing*) (AMANATIDES, 1984) e traçado de lápis (*pencil tracing*).

Todas as técnicas, ao estender a área amostrada pelo raio, possibilitam um tratamento anti-serrilhado que normalmente exigiria o lançamento de grande número de raios (super-amostragem), acelerando assim a geração da imagem.

A técnica de traçado de cones pode ser usada como o algoritmo de Warnock (WARNOCK, 1969) (FOLEY et alli, 1990) para eliminação de superfícies escondidas (subdivisão de área e estratégia "dividir para conquistar"): regiões homogêneas da imagem só precisam ser traçadas por um único cone. A vantagem desta técnica depende muito da coerência da imagem.

As técnicas de raios generalizados implicam em algumas restrições. Por exemplo, o traçado de feixes só pode ser usado com polígonos planos e apenas aproxima os efeitos de refração; o traçado de cones só pode ser feito sobre objetos constituídos de esferas, planos e polígonos.

2.4.3.4. Técnicas Direcionais

Os métodos mais recentes de aceleração de "ray tracing" são as chamadas técnicas direcionais (ARVO & KIRK, 1989).

As técnicas direcionais são aquelas que usam estruturas de dados especiais (cubo direcional) para manter informações sobre aspectos ligados à direção dos raios, em geral coletadas numa fase de pré-processamento, para acelerar a execução do algoritmo.

Operações envolvendo raios com aproximadamente a mesma direção são efetuadas em bloco ao invés de individualmente, poupando processamento. Por outro lado, estas técnicas implicam em grande sobrecarga no uso de memória.

O cubo direcional é uma estrutura de dados construída imaginando-se um cubo de lado 2, posicionado na origem do sistema de coordenadas. As coordenadas de cada face variam, portanto, no intervalo $[(-1,-1),(1,1)]$ de cada par de planos xy, xz e yz (fig. 2.22).

FIG. 2.22 - O cubo direcional

As faces do cubo direcional são subdivididas de forma uniforme ou adaptativa (árvore quaternária ("quadtree"), árvore de particionamento binário do espaço (BSP-tree), etc.). Dado um raio, pode-se usar técnicas análogas (2D) àquelas de travessia de raios em particionamento do espaço 3D para determinar-se qual célula do cubo direcional contém a intersecção com o raio.

Cada célula do cubo armazenará informações usadas posteriormente para aceleração do algoritmo de "ray tracing".

Entre os métodos direcionais, serão apresentados o memorizador de luz, e as técnicas de coerência e classificação de raios.

2.4.3.4.1. Memorizador de Luz (Light Buffer)

A técnica denominada Memorizador de Luz (*Light Buffer*) (HAINES & GREENBERG, 1986) usa o cubo direcional para acelerar o processamento de raios de sombra.

A cada fonte de luz pontual é associado um cubo direcional, uniformemente dividido. As células destes cubos contêm listas dos objetos do cenário visíveis através delas. As listas são ordenadas segundo distância crescente em relação à fonte de luz. Estes cubos direcionais são preparados no pré-processamento.

Durante a execução do "ray tracing", ao necessitar-se verificar se um dado ponto está na sombra em relação a uma fonte de luz, calcula-se a célula do cubo direcional atravessada pelo raio de sombra correspondente. Os objetos desta célula são então testados para intersecção com o raio de sombra, até que se encontre um ponto de intersecção ou que chegue-se a um objeto na lista mais distante que o ponto da superfície testada para sombra. No primeiro caso o objeto está na sombra e neste último, iluminado.

Algumas otimizações importantes podem ser feitas no algoritmo:

- polígonos com faces orientadas em direção oposta à fonte de luz e pertencentes a objetos opacos podem ser descartados.
- listas do cubo direcional com apenas um polígono podem ser descartadas já que um polígono não pode fazer sombra a si mesmo (ou já foi descartado no caso anterior).
- se algum objeto oclui totalmente uma célula, não é necessário verificar-se nenhum outro objeto mais distante. Qualquer outro objeto além deste estará na sombra daquela fonte de luz.
- antes de usar-se o memorizador de luz, verifica-se se o polígono está orientado em sentido oposto à fonte de luz, caso em que pode-se imediatamente supor-se que está em sua sombra.

Apesar da grande aceleração no processamento de raios de sombra, esta técnica é limitada por só aplicar-se a este tipo de raio.

2.4.3.4.2. Coerência de Raios

Esta técnica de aceleração é baseada no Teorema da Coerência de Raios (OHTA & MAEKAWA, 1987):

Seja S_1 uma esfera com centro o_1 e raio r_1 e S_2 uma esfera com centro o_2 e raio r_2 onde $|o_1 o_2| > r_1 + r_2$. Então, se o ponto P está dentro de S_1 e Q está dentro de S_2 , a seguinte inequação é válida:

$$\frac{P\vec{Q}}{|P\vec{Q}|} \cdot \frac{o_1\vec{o}_2}{|o_1\vec{o}_2|} > \sqrt{1 - \left(\frac{r_1 + r_2}{|o_1\vec{o}_2|} \right)^2} \quad (2.7)$$

O que o teorema da coerência de raios estabelece é que um objeto pode ser visto a partir dos outros objetos, somente através de uma direção (ângulo sólido) limitada (fig. 2.23).

FIG. 2.23 - O teorema da coerência de raios

O algoritmo para aceleração usando este teorema é implementado de maneira semelhante ao memorizador de luz, porém, usado também para objetos e o observador, além das fontes de luz.

O preenchimento das listas do cubo direcional uniforme é feito através da aplicação do teorema da coerência de raios ao invés de usar uma projeção como no caso do memorizador de luz.

O processamento se faz de forma a testar todos os objetos apontados pela lista da célula relevante do cubo direcional, até que a lista se esgote ou surja um objeto mais distante que a origem do raio sendo processado (o ponto de intersecção na superfície).

Resultados obtidos por Ohta e Maekawa mostram que o uso deste algoritmo torna o processamento de "ray tracing" independente do número de objetos do cenário (OHTA & MAEKAWA, 1987).

2.4.3.4.3. Classificação de Raios

Esta técnica, descrita por Arvo e Kirk (ARVO & KIRK, 1987), baseia-se na observação de que um raio no espaço tridimensional têm 5 graus de liberdade correspondendo aos pontos de $R^3 \times S^2$, onde S^2 é a esfera unitária no espaço 3D (conjuntos das direções).

Esta técnica particiona o espaço 5D, durante a execução do algoritmo, armazenando para cada célula um cubo direcional contendo listas de objetos visíveis através de cada face do cubo.

O particionamento é feito de forma adaptativa segundo uma hiper-octárvore ("hiper-octree"), que é percorrida durante o processamento para encontrar-se o nó correspondente à origem e direção de cada raio. A célula encontrada conterá os objetos candidatos a intersecção.

Esta técnica é vantajosa porque se aplica a todos os tipos de raio (primários, secundários e de sombra). Sua desvantagem é a grande sobrecarga no uso de memória.

CAPÍTULO 3

CONCEITOS E DEFINIÇÕES BÁSICAS EM COMPUTAÇÃO PARALELA

3. CONCEITOS E DEFINIÇÕES BÁSICAS EM COMPUTAÇÃO PARALELA

Dado que a literatura básica em computação paralela é bastante diversificada e que freqüentemente termos são utilizados com mais de um sentido, apresenta-se este capítulo com o objetivo de estabelecer qual significado adotou-se para os termos, definições e conceitos ligados ao processamento paralelo, dentro do escopo deste trabalho. Também são apresentadas algumas considerações acerca de decisões tomadas na condução da presente pesquisa, assim como certas particularizações para o processamento paralelo quando aplicado ao algoritmo de ray tracing.

3.1. Memória Compartilhada x Memória Distribuída

A busca de computadores cada vez mais poderosos para atender a demanda de processamento das aplicações modernas levou a dois caminhos principais: os supercomputadores com apenas um ou poucos processadores ultra rápidos e os computadores maciçamente paralelos com muitos processadores de capacidade moderada.

A arquitetura dos primeiros em geral emprega bancos de memória compartilhados por todos os processadores através de barramentos (*bus*) ou chaves (*switches*), limitando sua escalabilidade devido à contenção de memória e aspectos construtivos.

Os computadores maciçamente paralelos, por sua vez, adotam memória fisicamente distribuída, sendo por isso escaláveis, isto é, geralmente pode-se acrescentar uma grande quantidade de processadores ao sistema antes que algum fator limitante ("gargalo") venha impedir novos incrementos. Esta característica, aliada ao fato de cada processador individualmente ter preço comercial acessível, faz com que se possa construir máquinas de alto desempenho a custos reduzidos em relação a outras soluções.

3.2. Taxonomia de Flynn

Uma taxonomia bastante popular para classificar arquiteturas paralelas é apresentada em (FLYNN, 1972), e resumida na tabela 3.1:

FLUXO DE DADOS	FLUXO DE INSTRUÇÕES	
	<i>SIMPLES</i>	<i>MÚLTIPLO</i>
<i>SIMPLES</i>	SISD	MISD
<i>MÚLTIPLO</i>	SIMD	MIMD

TAB. 3.1 - A taxonomia de Flynn

3.3. Multiprocessadores x Multicomputadores

Os computadores paralelos com arquitetura MIMD, podem ser classificados em dois tipos (WILSON, 1993):

- *Multiprocessadores*: compartilham o mesmo espaço de endereçamento lógico. Podem ser programados como uma máquina paralela de memória compartilhada. Também denominados fortemente acoplados.
- *Multicomputadores*: cada processador tem um espaço de endereçamento lógico independente. Um processador não tem acesso direto à memória dos demais. Também denominados fracamente acoplados.

3.3.1. Por que Multicomputadores ?

A classe de equipamento focalizada neste trabalho foi escolhida levando-se em consideração os seguintes aspectos:

- *relevância*: a implementação do algoritmo de ray tracing em máquinas de memória compartilhada não representa um desafio já que ele se comporta, nestas máquinas, praticamente da mesma forma que outros algoritmos sem código vetorial explícito. A implementação em máquinas fracamente acopladas apresenta interessantes aspectos dadas as muitas formas conceituais possíveis para a realização prática deste algoritmo.
- *arquitetura promissora*: entre todas as classes de máquinas paralelas a única que parece mais promissora hoje do que a selecionada é a dos multiprocessadores escaláveis (máquinas MIMD de memória distribuída com espaço de endereçamento único) (BELL, 1992) que são uma evolução dos multicomputadores. A escolha ainda assim se justifica porque estas máquinas permanecem pouco disponíveis e têm custo bem maior, utilizando tecnologias proprietárias. As máquinas SIMD foram abandonadas dada a dificuldade de programá-las e sua pequena generalidade quanto às aplicações possíveis. As arquiteturas MIMD com memória compartilhada apresentam a grande desvantagem de não serem escaláveis limitando seriamente o desempenho máximo atingível num determinado nível tecnológico, além do seu alto custo. Os multicomputadores, no entanto, são escaláveis e, no presente momento, detêm o recorde de desempenho (INTEL Paragon).

- *caráter estratégico*: ainda existem restrições à venda de supercomputadores a países como o Brasil impostas por potências militares do primeiro mundo. Há poucas perspectivas de se desenvolverem imediatamente em nosso país supercomputadores convencionais pois não existe investimento suficiente para pesquisa em tecnologias como GaAs que são aplicadas neste tipo de computador. Por outro lado, já existe competência nacional na área de computadores maciçamente paralelos e a fabricação desta classe de máquinas pode ser feita a partir de componentes comerciais, de custo acessível e sem restrições de uso. Portanto, desenvolver aplicações para esta classe de computadores é sobremaneira mais conveniente e estratégico para o Brasil.
- *aproveitamento de recursos*: estudos mostram (NIELSEN, 1991) que as taxas de utilização de estações de trabalho são baixas (15%). Por outro lado também sabe-se que há uma forte tendência no sentido de reduzirem-se os ambientes centralizados e caminhar-se em direção a ambientes de processamento distribuído ("down-sizing") conduzindo ao aumento do tamanho e número de redes de estações de trabalho. O aproveitamento destes recursos se faz uma necessidade econômica inegável. A arquitetura de multicomputadores pode ser mapeada integralmente numa rede como a mencionada de modo que os estudos desenvolvidos para esta classe de equipamento pode ser extrapolada para estes ambientes de processamento distribuído. Apesar de bastante semelhante aos multicomputadores, esta característica ainda não existe para os chamados "multiprocessadores" pois a velocidade das redes e o sistema operacional destas máquinas não permite um mapeamento desta arquitetura para as redes de estações de trabalho.

3.4. Considerações sobre Topologia

Neste trabalho não serão feitas maiores considerações sobre topologia, devido às seguintes razões:

- os multicomputadores disponíveis comercialmente e desenvolvidos em laboratórios têm topologias bastante diversificadas. Alguns permitem reconfiguração da rede de interconexão, e alguns até o fazem dinamicamente.
- com o aumento do número médio de nós nos multicomputadores, as alternativas de interconexão se multiplicam e é cada vez mais difícil a escolha adequada da melhor topologia para uma dada aplicação, sendo favorecidas topologias genéricas, usadas para todas as aplicações.

- o roteamento de mensagens implementado em circuitaria (*hardware*) será uma característica comum de todos os multicomputadores de segunda geração (ATHAS & SEITZ, 1988), assim como a latência se tornará bastante independente da distância percorrida, liberando o programador de preocupações quanto à topologia do sistema (HOLLIMAN et alli, 1993).

3.5. Definições Básicas

Nesta seção define-se o significado de outros termos e conceitos relacionados à computação paralela, que serão usados no restante deste trabalho.

3.5.1. Nó de Processamento

Uma unidade de processamento de um computador paralelo. Num multicomputador significa um processador e sua memória local.

3.5.2. Sistema

Denominaremos sistema ao conjunto computador + algoritmo.

3.5.3. Topologia

Uma família de grafos criados usando a mesma regra geral ou que compartilham certas propriedades. Num multicomputador indica como os vários nós de processamento estão conectados entre si.

As topologias citadas neste trabalho são:

- *Malha (Mesh)*: uma topologia na qual os nós formam uma grade regular acíclica, n-dimensional, onde cada aresta é paralela a um eixo da grade e liga dois nós adjacentes ao longo daquele eixo (fig. 3.1a).
- *Hipercubo*: uma topologia na qual cada nó é um vértice de um cubo n-dimensional (figura 3.1b)
- *Toroidal*: uma topologia na qual os nós formam uma grade regular cíclica, n-dimensional, onde cada aresta é paralela a um eixo da grade e liga dois nós adjacentes ao longo daquele eixo (fig. 3.1c).

3.5.4. Elo ("Link")

Uma conexão um-para-um entre dois processadores, através da qual se comunicam.

FIG. 3.1 - Topologias

3.5.5. Conectividade

O número de elos necessário a um nó de processamento para que toda a comunicação seja realizada com nós diretamente conectados a ele. É uma característica dependente do algoritmo implementado.

3.5.6. Granularidade

O tamanho ou quantidade de operações feitas por um processador entre dois eventos de comunicação.

Granularidade Fina (fine-grain): um processo realiza poucas operações entre duas comunicações.

Granularidade Espessa (coarse-grain): um processo realiza muitas operações entre duas comunicações.

3.5.7. Latência de Comunicação

A parcela do tempo necessário para enviar uma mensagem, que é independente do tamanho da mensagem (o tempo necessário para enviar uma mensagem de tamanho zero).

3.5.8. Memória Cache

Memória de alta velocidade, local a um nó de processamento, usada para acelerar referências à memória principal.

3.5.9. Balanceamento de Carga

O grau de homogeneidade de distribuição de trabalho entre os processadores disponíveis num sistema.

3.5.10. Sobre carga ("Overhead")

Refere-se ao processamento adicional requerido devido a uma modificação num algoritmo. Esta modificação pode ser sua paralelização.

Sobre carga de memória: uso adicional de memória induzido por uma modificação num algoritmo.

3.5.11. Redundância de Dados

Redundância é uma medida do uso de memória de um algoritmo num multicomputador. É definida por:

$$R = \frac{m_N}{M \cdot N} \quad (3.1)$$

onde: R redundância de dados;
 m_N memória total utilizada num multicomputador com N nós;
 M memória utilizada pelo algoritmo seqüencial;
 N número de nós do multicomputador.

3.5.12. Fazenda de Processadores ("Processor Farm")

É uma técnica onde um processo chamado mestre ou fonte gera um conjunto de tarefas que vão sendo solicitadas e processadas por um grupo de processos denominados trabalhadores ou escravos. Se o número de tarefas é muito maior que o de trabalhadores esta técnica pode ser uma maneira eficaz de balancear a carga num sistema.

3.5.13. Ganho de Velocidade

É a razão entre os tempos de execução de um programa em dois sistemas com número de processadores diferentes. Se T é o tempo de execução usando-se um

processador e t_N é o tempo de execução em N processadores então o ganho de velocidade é dado por:

$$G = \frac{T}{t_N} \quad (3.2)$$

Ganho de velocidade linear: ganho diretamente proporcional ao número de processadores utilizado.

Ganho de velocidade unitário (HELMBOLD & McDOWELL, 1990):

$$\lim_{N \rightarrow \infty} \frac{G_N}{N} = 1 \quad (3.3)$$

Ganho de velocidade superlinear:

$$\lim_{N \rightarrow \infty} \frac{G_N}{N} > 1 \quad (3.4)$$

É importante considerar, para fins deste trabalho, que o aumento no número de nós (N) não implica em correspondente aumento na memória total do sistema.

3.5.14. Eficiência

É uma medida do uso dos processadores de um sistema. É calculada por:

$$E = \frac{T}{N \cdot t_N} \cdot 100\% = \frac{G}{N} \cdot 100\% \quad (3.5)$$

onde: E Eficiência;
 T Tempo de processamento com um nó;
 t_N Tempo de processamento com N nós;
 G Ganho de velocidade.

3.5.15. Lei de Amdahl

Regra enunciada por G. Amdahl (AMDAHL, 1967) que estabelece que se σ é a proporção de uma tarefa que é inerentemente sequencial e $1-\sigma$ é a parte que pode ser paralelizada, então o máximo ganho de velocidade que pode ser atingido com N processadores é :

$$G = \lim_{N \rightarrow \infty} \frac{1}{\sigma + \frac{1-\sigma}{N}} = \frac{1}{\sigma} \quad (3.6)$$

CAPÍTULO 4

'RAY TRACING' PARALELO: CONCEITUAÇÃO, AVALIAÇÃO E TAXONOMIA

4. "RAY TRACING" PARALELO: CONCEITUAÇÃO, AVALIAÇÃO E TAXONOMIA

O uso de computadores paralelos para o processamento de "ray tracing" é uma idéia já presente no artigo de Turner Whitted (WHITTED, 1980) que introduziu esta técnica. Desde então, inúmeras propostas de paralelização deste algoritmo têm surgido, demonstrando claramente sua adequação ao processamento paralelo.

As pesquisas nesta área já vão muito além da exploração do paralelismo evidente no fato do processamento de cada pontel ser totalmente independente dos demais. As taxonomias de soluções apresentada neste capítulo mostra a grande variedade de enfoques aplicados a este problema.

Neste capítulo é apresentada, inicialmente, uma conceituação original do que seja o problema " 'ray tracing' paralelo em multicomputadores". São propostas algumas ferramentas para avaliação de soluções. A seguir, são apresentadas classificações encontradas na literatura para enquadrar as diversas alternativas de paralelização desta técnica. Fechando o capítulo, é proposta uma nova taxonomia para "ray tracing" paralelo, que tem como finalidades classificar soluções e orientar a pesquisa de novas alternativas a este problema.

4.1. Conceituação do Problema de "Ray Tracing" Paralelo

A forma mais simples de se implementar o algoritmo de "ray tracing" num multicomputador é usando-se uma técnica genérica para processamento paralelo denominada "fazenda de processadores" (*processor farm*) (MAY, 1987), (HOLLIMAN et alli, 1989), (PACKER, 1987), descrita no capítulo 3 deste trabalho e no item 5.3.1 para o caso específico de "ray tracing".

Implementações desta técnica obtiveram eficiência de 95.5% para multicomputadores com 80 nós (PACKER, 1987) e ganho de velocidade linear para até 128 nós (HOLLIMAN et alli, 1989). Estes resultados são os melhores obtidos para implementação de "ray tracing" em multicomputadores e exemplares dentre todas as aplicações executadas nesta classe de equipamento.

A fazenda de processadores aplicada ao "ray tracing" está muito próxima de atingir os objetivos de qualquer implementação paralela de um algoritmo, já que obtém ganho de velocidade quase linear, é escalável e não impõe restrições algorítmicas adicionais àquelas da versão seqüencial.

Surge então, naturalmente, a seguinte pergunta: se a implementação paralela mais trivial do "ray tracing" é assim tão boa, qual a razão do presente trabalho e de tantas outras pesquisas sobre o mesmo assunto ?

Pesquisam-se novas formas de paralelizar o "ray tracing" porque, se a implementação segundo a fazenda de processadores é quase imbatível em termos de ganho de velocidade ela também é, pelo fato de fazer replicação *total* da descrição da cena em *todos* os nós do sistema, uma das que exigem a maior quantidade de memória local de cada nó. Este é o ponto central das pesquisas na área. As soluções pesquisadas são, em geral, um compromisso entre tempo de processamento e memória utilizada pelo algoritmo.

Considerando-se a complexidade das imagens, a tendência atual e futura é produzir-se imagens cada vez mais realísticas. Isto é conseguido atualmente com o refinamento dos algoritmos de iluminação e principalmente com um maior detalhamento no modelamento da cena para que esta não tenha um aspecto simplista, evidenciando sua origem artificial. Este maior detalhamento implica em descrições geométricas que ocupam cada vez mais memória.

Sabe-se que, hoje, apesar da densidade cada vez maior dos circuitos de memória, este ainda é um recurso escasso que deve ser utilizado com eficiência, já que as aplicações também têm continuamente demandado mais memória. Por esta razão, de pouca utilidade prática são os algoritmos paralelos de "ray tracing" que exigem total replicação da descrição geométrica da cena em todos os nós de um sistema paralelo. Algoritmos deste tipo têm aplicação somente no processamento de cenas relativamente simples ou em casos em que pode-se pagar qualquer preço em busca de velocidade de processamento, como pode ser o caso de simuladores de voo e aplicações militares.

Este problema se agrava no caso de computadores maciçamente paralelos, com centenas e até milhares de nós, onde a redundância pode significar centenas de Mbytes de memória com dados duplicados.

A seção seguinte propõe alguns conceitos úteis para o estudo e caracterização dos algoritmos paralelos de "ray tracing".

4.2. Soluções de Referência

Conforme depreende-se da seção anterior, o "ray tracing" é um algoritmo sujeito ao compromisso velocidade x memória. Assim, ao examinar-se a qualidade das soluções propostas, o desempenho em ambos os itens deve ser considerado.

Para auxiliar este estudo, propõe-se, a seguir, uma nomenclatura de soluções de referência para "ray tracing" paralelo (tab. 4.1). Por soluções de referência, entende-se aqui, algoritmos hipotéticos dos quais se conhecem características de desempenho que serão comparadas àquelas de algoritmos propostos, obtendo-se assim, uma estimativa de sua qualidade.

A chamada **Solução Seqüencial** será aquela do algoritmo seqüencial para "ray tracing", sendo executado em um equipamento contendo um único processador do mesmo tipo utilizado no multicomputador de referência. Esta solução usa memória M e T unidades de tempo para a geração completa da imagem. Esta não é uma solução paralela mas sempre é usada para avaliar-se o desempenho comparativo de algoritmos paralelos. É conveniente notar que é difícil indicar qual o *melhor* algoritmo seqüencial de "ray tracing" já que o desempenho de uma solução qualquer é dependente de características dos objetos do cenário, resolução e coerência da imagem, etc. Considera-se que não existe qualquer tipo de sobrecarga ("*overhead*") nesta implementação; qualquer nova técnica de aceleração pode ser implementada; quaisquer técnicas anti-serrilhado ("*antialiasing*") e para aumento de realismo podem ser implementadas (i.e., não existem restrições a algoritmos).

Adota-se, como hipótese, que a memória mínima necessária para armazenar a descrição completa do cenário é M .

Segundo esta hipótese, propõe-se a definição das soluções ótima, dual e ideal, como segue.

A **Solução Ótima** é aquela em que obtém-se ganho de velocidade linear e unitário porém com replicação total dos dados em todas as memórias locais (i.e., memória total $N.M$). Esta solução pode ser praticamente atingida usando-se N computadores independentes ou com a implementação em fazenda de processadores, desde que a sobrecarga de comunicação seja baixa e o número de processadores não seja demasiado grande. Como o ganho é linear e unitário, o tempo de processamento desta solução é N vezes menor que o da solução seqüencial: T/N . Esta denominação preserva a noção consagrada de otimalidade em processamento paralelo (WILSON, 1993) - (o menor produto "tempo x processador" possível)

Note-se que a Solução Ótima representa uma otimização em relação ao tempo de processamento. Chamaremos de **Solução Dual** à solução para otimização em relação ao uso de memória. Nesta solução usa-se N processadores, cada um com M/N de memória (i.e., memória total M) e tempo de processamento igual a T . Este é exatamente o desempenho da Solução Seqüencial, com a desvantagem de usar-se N processadores.

Assim, dada a inconveniência de tais características, reservou-se a denominação "Solução Ótima" para a solução ótima em processamento e "Solução Dual" para a sua dual, ótima em relação ao uso de memória.

A **Solução Ideal** se refere à meta a ser atingida pelos algoritmos paralelos de "ray tracing": ganho de velocidade linear e unitário, sem nenhuma redundância de dados em memória. Como todo modelo ideal, este também é provavelmente impossível de implementação prática.

Além destas características, nenhuma das três soluções paralelas impõe quaisquer restrições à realização de técnicas de aceleração passíveis de implementação na versão sequencial. Isto é para garantir que o ganho de velocidade continuará linear nas soluções ótima e ideal, mesmo em casos em que se melhore o desempenho sequencial por meio de novas técnicas de aceleração. Estas técnicas, aplicadas às versões paralelas, tornariam o desempenho novamente linear. Adicionalmente, as soluções ideal e ótima são (infinitamente) escaláveis quanto ao número de processadores, isto é, o sistema manterá desempenho linear com qualquer quantidade de processadores.

	SOLUÇÃO SEQÜENCIAL	SOLUÇÃO ÓTIMA	SOLUÇÃO DUAL	SOLUÇÃO IDEAL
Número de processadores	1	N	N	N
Tempo de Processamento	T	T/N	T	T/N
Eficiência	100%	100%	$1/N$	100%
Requisitos de Memória	M	$N.M$	M	M
Restrições Algorítmicas	Não impõe nenhuma	Não impõe nenhuma	Não impõe nenhuma	Não impõe nenhuma
Escalável	-	SIM	NÃO	SIM

TAB. 4.1 - Soluções de Referência para "ray tracing" paralelo

Do exposto fica claro que o desenvolvimento de novos algoritmos paralelos de "ray tracing" em geral tentarão aproximar-se da Solução Ideal. Já se mencionou que as cenas que se deseja processar são cada vez mais complexas, requerendo sempre mais memória, o que torna o uso redundante desta altamente indesejável. Somente em aplicações muito

específicas pode-se negligenciar este fato. Por outro lado, quando a memória total do equipamento de que se dispõe é maior do que a necessária para processar a cena com um algoritmo "econômico", pode-se lançar mão de um outro algoritmo que exija mais memória porém retribuindo com melhor desempenho em velocidade.

4.3. Uma Ferramenta Gráfica para Avaliação de Soluções

Nas sub-seções seguintes introduziremos uma série de definições (de cunho original) que nos permitirão a criação de uma ferramenta gráfica para a avaliação comparativa de soluções de "ray tracing".

4.3.1. Eficiência Incremental

A eficiência incremental (α) indica o aumento no ganho de velocidade proporcionado por processadores que se adicionam a um sistema seqüencial, isto é, indica a eficiência dos processadores acrescentados. É definida por:

$$\alpha = \frac{E \cdot N - 1}{N - 1} = \frac{G - 1}{N - 1} = \frac{T - t_N}{t_N - 1} \quad (4.1)$$

onde: α eficiência incremental;
 G ganho de velocidade;
 E eficiência do sistema;
 N número de processadores;
 T tempo de processamento seqüencial;
 t_N tempo de processamento com N processadores.

Assim como a eficiência, pode-se escrever a eficiência incremental em forma decimal (p. ex., $\alpha = 0.75$) ou porcentual (p. ex., $\alpha = 75\%$)

Exemplificando, um sistema seqüencial onde é acrescentado 1 (um) processador apresentará ganho $G = 1.5$ se a eficiência incremental for $\alpha = 50\%$ e ganho $G = 1.0$ se a eficiência for $E = 50\%$. A tabela 4.2 apresenta outros exemplos.

N	GANHO DE VELOCIDADE (G)					
	$\alpha = 0\%$	$E = 0\%$	$\alpha = 50\%$	$E = 50\%$	$\alpha = 100\%$	$E = 100\%$
1	1.0	0.0	1.0	0.5	1.0	1.0
2	1.0	0.0	1.5	1.0	2.0	2.0
3	1.0	0.0	2.0	1.5	3.0	3.0
...	...	...	...	...	...	...
10	1.0	0.0	5.5	5.0	10.0	10.0

TAB. 4.2 - Comparação entre Eficiência e Eficiência Incremental

Segundo esta definição, as Soluções Ideal e Ótima apresentam eficiência incremental $\alpha = 100\%$ enquanto a Solução Dual tem $\alpha = 0\%$.

4.3.2. Redundância Incremental

A redundância incremental (β) indica o percentual de dados que são replicados num sistema paralelo. É definida por:

$$\beta = \frac{R \cdot N - 1}{N - 1} = \frac{m_N - M}{M(N - 1)} \quad (4.2)$$

onde: β redundância incremental;
 R redundância de dados;
 N número de processadores;
 m_N memória de dados total utilizada no sistema paralelo com N nós;
 M memória de dados utilizada pelo algoritmo seqüencial.

Um sistema com redundância incremental $\beta = 100\%$ replica todos os dados em todos os nós de processamento, utilizando memória total $m_N = N \cdot M$ como a Solução Ótima. As Soluções Ideal e Dual têm redundância incremental nula ($\beta = 0\%$) e sempre usam memória total $m_N = M$ (tabela 4.3).

N	MEMÓRIA TOTAL DE DADOS (m_N)		
	$\beta = 0\%$	$\beta = 50\%$	$\beta = 100\%$
1	M	M	M
2	M	$1.5 M$	$2.0 M$
3	M	$2.0 M$	$3.0 M$
...	...	...	...
10	M	$5.5 M$	$10.0 M$

TAB. 4.3 - A mem. tot. usada por uma solução em função de sua redundância incremental

A tabela 4.4 resume a eficiência e redundância incrementais para as soluções de referência:

SOLUÇÃO	EFICIÊNCIA INCR. (α)	REDUNDÂNCIA INCR. (β)
SEQÜENCIAL	-	-
IDEAL	100%	0%
ÓTIMA	100%	100%
DUAL	0%	0%

TAB. 4.4 - Eficiência e redundância incrementais das Soluções de Referência

4.3.3. Lucro de um Sistema de "Ray Tracing"

O problema de "ray tracing" paralelo em multicomputadores, conforme estabelecido anteriormente, se diferencia de problemas mais genéricos de implementação de algoritmos em multicomputadores pela importância do uso eficiente da memória.

Dessa forma, deve-se estabelecer uma métrica para a qualidade das soluções que possa levar em conta, além da eficiência de processamento, tradicionalmente adotada, também a redundância de dados, condensando em um só número estes dois parâmetros. Adicionalmente, esta métrica deve considerar que o valor que se dá ao uso eficiente de memória pode variar, de acordo com a aplicação.

Assim, definiremos o parâmetro *Lucro* (L) como sendo o lucro (ou valor) que se obtém com um dado sistema, a cada T unidades de tempo de processamento:

$$\begin{aligned}
 L &= E \cdot N \cdot C_a - N \cdot C_p - C_m \cdot [1 + \beta \cdot (N - 1)] = \\
 &= C_a \cdot [1 + \alpha \cdot (N - 1)] - N \cdot C_p - C_m \cdot [1 + \beta \cdot (N - 1)]
 \end{aligned} \tag{4.3}$$

onde: L lucro do sistema;

- E eficiência;
 N número de nós de processamento;
 α eficiência incremental;
 β redundância incremental.
 C_a custo de animação: indica o valor da imagem ou animação produzida pelo sistema seqüencial em T unidades de tempo;
 C_p custo do processador: indica o custo, ou aluguel, de um processador do tipo utilizado no multicomputador em questão, durante um período de T unidades de tempo.
 C_m custo de memória: indica o custo, ou aluguel, de um banco de memória de capacidade M , durante um período de T unidades de tempo.

O custo da memória utilizada para armazenamento de programas e sistema operacional não está inclusa no parâmetro C_m pois é inerente à adição de novos processadores. Deve, por isso, ser incluído no custo C_p .

Esta definição simplificada de Lucro leva em conta o valor do resultado produzido, isto é, uma ou mais imagens (ou uma animação), e os custos dos processadores e da memória do multicomputador utilizado, durante um período de tempo.

O Lucro proporcionado por um sistema varia de acordo com os custos adotados. Veja-se os exemplos a seguir.

Considerando-se os sistemas I e II (multicomputador + algoritmo), com características de desempenho indicadas na tabela 4.5 abaixo, calcular-se-á o lucro para dois casos (A e B), a seguir.

SISTEMA	EFICIÊNCIA INCR.	REDUNDÂNCIA INCR.
I	$\alpha = 50\%$	$\beta = 80\%$
II	$\alpha = 45\%$	$\beta = 50\%$

TAB. 4.5 - Parâmetros de desempenho de sistemas paralelos para "ray tracing"

A) *Produtora de Comerciais*: uma empresa produzindo animações para comerciais de TV deseja obter o máximo de imagens no menor tempo. Qual dos sistemas (I ou II) deve escolher? Supondo-se custos hipotéticos de processadores e memória $C_p = 15$ e $C_m = 10$, e $C_a = 50$, tem-se:

$$L_I = 2N + 23 \leq L_{II} = 2.5N + 22.5 \quad \therefore \text{O sistema II é mais adequado no caso A.}$$

B) *Simulador de vôo*: uma empresa produz um simulador de vôo para jato de passageiros de grande porte. Considerando os mesmos custos para os componentes eletrônicos (no mesmo período T) porém, agora com custo da animação $C_a = 80$, qual dos sistemas deve escolher ?

$$L_I = 17 \cdot N + 38 \geq L_{II} = 16N + 39 \quad \therefore \text{O sistema I é mais adequado no caso B.}$$

A tabela 4.6 traz o lucro das soluções de referência.

SOLUÇÃO	LUCRO (L)
SEQUENCIAL	$L_{sequ} = C_a - C_p - C_m$
IDEAL	$L_{ideal} = NC_a - NC_p - C_m$
ÓTIMA	$L_{ótima} = NC_a - NC_p - NC_m$
DUAL	$L_{dual} = C_a - NC_p - C_m$

TAB. 4.6 - Lucro das Soluções de Referência

4.3.4. Índice de Lucro

O *Lucro* de um sistema de "ray tracing" é uma medida que varia muito com os parâmetros de custo, e não fornece uma indicação direta de quão boa é uma solução. Uma alternativa mais interessante é comparar-se o lucro de uma solução com o máximo lucro possível, aquele da solução ideal. Assim, definimos o seguinte *Índice de Lucro* (D):

$$D = \frac{L}{L_{ideal}} = \frac{C_a \cdot [1 + \alpha \cdot (N - 1)] - N \cdot C_p - C_m \cdot [1 + \beta \cdot (N - 1)]}{N \cdot C_a - N \cdot C_p - C_m} \quad (4.4)$$

Este parâmetro indica quão próximo uma solução qualquer está da solução ideal. Leva em consideração a eficiência de processamento e de uso da memória, segundo os pesos (custos) atribuídos pelo usuário do índice.

O *Índice de Lucro*, assim como o *Lucro*, é função do número de nós do sistema já que a eficiência de soluções não escaláveis vai degenerando com o aumento do número de processadores, isto é, vai se distanciando da solução ideal.

Em alguns casos, para verificar-se a escalabilidade de uma solução, pode ser interessante fazer a definição do índice de lucro para $N \rightarrow \infty$:

$$D_{\lim_{N \rightarrow \infty}} = \lim_{N \rightarrow \infty} \frac{L}{L_{ideal}} = \frac{C_a \cdot \alpha - C_p - C_m \cdot \beta}{C_a - C_p} \quad (4.5)$$

A tabela 4.7 apresenta o índice de lucro das soluções de referência.

SOLUÇÃO	D	$D_{\lim N \rightarrow \infty}$
IDEAL	1	1
ÓTIMA	$N \cdot \frac{C_a - C_p - C_m}{N \cdot C_a - N \cdot C_p - C_m}$	$\frac{C_a - C_p - C_m}{C_a - C_p}$
DUAL	$\frac{C_a - N \cdot C_p - C_m}{N \cdot C_a - N \cdot C_p - C_m}$	$\frac{-C_p}{C_a - C_p}$
SEQÜENCIAL	$\frac{C_a - C_p - C_m}{N \cdot C_a - N \cdot C_p - C_m}$	0

TAB. 4.7 - Índices de Lucro das soluções de referência

4.3.5. O Gráfico Eficiência x Redundância Incrementais

Introduziremos agora uma ferramenta gráfica que permite visualizar rapidamente o mérito de soluções de "ray tracing" paralelo em multicomputadores.

Os parâmetros que indicam de forma mais imediata o desempenho de uma solução de "ray tracing" paralelo são o tempo de processamento e a memória total utilizada, ambos em função do número de processadores e do desempenho da solução seqüencial. Estes parâmetros são indicados de forma concisa pela eficiência e a redundância. Assim, propõe-se uma representação gráfica para um espaço do tipo *eficiência x redundância* para que se possa fazer comparações rápidas do mérito de soluções. É conveniente que se represente neste gráfico a eficiência e a redundância nas formas incrementais definidas anteriormente.

Cada solução será indicada por um ponto (α, β) neste espaço, cujas coordenadas corresponderão à sua eficiência e redundância incrementais. As posições das soluções de referência estão indicadas na figura 4.1, e não variam com N (número de processadores). Apesar disto, o gráfico deve ser construído para valores específicos de N , nos quais se tenha interesse.

FIG. 4.1 - As soluções de referência no espaço eficiência x redundância incrementais

Ao escrever-se a *eficiência incremental* (α) em função do *índice de lucro* (D), observa-se que o lugar geométrico das soluções com *índice de lucro* constante é uma reta com inclinação $\theta = \arctan\left(\frac{C_m}{C_a}\right)$ e que, portanto, não varia com N :

$$\alpha = \beta \cdot \tan \theta + k = \beta \cdot \frac{C_m}{C_a} + \frac{D \cdot L_{ideal} + N \cdot C_p + C_m - C_a}{(N-1) \cdot C_a} \quad (4.6)$$

O lugar geométrico das soluções que apresentam índice de lucro igual ao da solução ótima é dado por:

$$\alpha_{D_{ótimo}} = \beta \cdot \frac{C_m}{C_a} + \frac{(C_a - C_m)}{C_a} \quad (4.7)$$

Portanto, é uma reta que cruza o eixo α em $\left(1 - \frac{C_m}{C_a}, 0\right)$ passando pelo ponto (1,1).

Soluções que têm índice de lucro equivalente ao da solução seqüencial, constituem o seguinte lugar geométrico:

$$\alpha_{D_{sequ}} = \beta \cdot \frac{C_m}{C_a} + \frac{C_p}{C_a} \quad (4.8)$$

A solução dual está localizada no ponto (0, 0) e, portanto, soluções com índice de lucro equivalente têm lugar geométrico:

$$\alpha_{D_{dual}} = \beta \cdot \frac{C_m}{C_a} \quad (4.9)$$

Para soluções com $D = D_{ideal}$, tem-se:

$$\alpha_{D_{ideal}} = \beta \cdot \frac{C_m}{C_a} + 1 \quad (4.10)$$

Resumindo, o gráfico resultante (figura 4.2) tem as seguintes propriedades:

- As soluções *dual*, *ótima* e *ideal* assumem posições destacadas, delimitando regiões;
- As posições das soluções de referência são invariantes com o número de processadores utilizados (N);
- Os lugares geométricos definidos por soluções com $D = cte$ são retas com inclinação $\arctan(C_m/C_a)$;

- O lugar geométrico das soluções com $D = D_{\text{ótimo}}$ é definido pela reta que passa pelos pontos $(1,1)$ e $(1 - C_m/C_a, 0)$;
- O lugar geométrico das soluções com $D = D_{\text{sequ}}$ é definido pela reta que passa pelo ponto $(C_p/C_a, 0)$ e tem inclinação $\arctan(C_m/C_a)$;

FIG. 4.2 - As regiões do espaço eficiência x redundância incrementais

A qualidade de uma solução pode ser imediatamente verificada quando inserida no gráfico. As linhas inclinadas determinam faixas de índice de lucro crescentes de baixo para cima.

A única solução teoricamente possível com o mesmo índice de lucro da solução ideal é a própria solução ideal: outras soluções com $D = 1$ deveriam apresentar ganho superlinear ou redundância de dados negativa, o que são condições impossíveis segundo as hipóteses adotadas.

As linhas que indicam soluções com índice de mérito constante serão tanto mais inclinadas quanto for o custo da memória em relação ao processamento. Quando este custo for desprezível, estas linhas serão horizontais e o mérito das soluções será caracterizado exclusivamente pela sua eficiência de processamento.

A tabela 4.8 mostra as características das soluções em cada uma das regiões do espaço eficiência x redundância.

REGIÃO	COMENTÁRIOS
A	Solução superlinear ⁶ . Usa menos tempo e menos memória que a Solução Ideal. Totalmente impossível segundo as hipóteses adotadas.
B	Solução superlinear. Usa menos tempo e mais memória que a Solução Ideal. Pode indicar a existência de solução sequencial melhor que a escolhida.
C	Superlinear. É mais rápida que a solução ideal porém usa mais memória do que N Seqüenciais. Indica que se pode implementar um algoritmo seqüencial de melhor desempenho fazendo uso de mais memória.
D	Usa menos memória que a solução seqüencial porém leva mais tempo. Implementável com técnicas de <i>swapping</i> ⁷ eficientes.
E	Soluções disponíveis que apresentam interesse. A solução ótima é seu limite inferior.
F	Soluções possíveis porém de pouco interesse já que a solução ótima é melhor e pode ser praticamente atingida com N computadores independentes.
G	Soluções sem interesse prático pois têm índice de lucro menor que a solução seqüencial.
H	Soluções nesta região não justificam o uso de um multicomputador. Podem ser atingidas com um computador seqüencial usando <i>swapping</i> .
I	Soluções inaceitáveis pois têm desempenho pior que o da solução seqüencial em todos os aspectos.

TAB. 4.8 - As regiões do gráfico eficiência x redundância

Soluções fora da região E do gráfico não apresentarão grandes atrativos pois, segundo os custos adotados, têm índice de lucro menor que a solução ótima, passível de implementação.

⁶ Qualquer solução que apresente comportamento superlinear é teoricamente impossível e indicará que o algoritmo da Solução Serial pode ser melhorado tornando seu ganho novamente linear, no melhor caso.

⁷ Memória virtual implementada em memória de armazenamento secundário. Geralmente, um serviço fornecido pelo sistema operacional.

Soluções abaixo da linha $D = D_{sequ}$ são piores que a solução sequencial, indicando que a solução paralela não deve ser implementada.

4.4. Parâmetros para Avaliação do Alg. de "Ray Tracing" em Multicomputadores

Para que se possa analisar em mais detalhes o mérito de uma proposta de paralelização de "ray tracing" é necessário que se observe algumas características do algoritmo que determinarão seu comportamento em uma dada arquitetura. As características estudadas neste trabalho são relacionadas a seguir.

4.4.1. Sobrecargas de Processamento

Tanto as sobrecargas (*overheads*) de processamento quanto as de pré-processamento podem se constituir em aspectos determinantes do tempo total de execução do algoritmo e portanto devem ser analisadas.

O *pré-processamento* se refere a todo o processamento realizado antes de iniciar-se a execução do algoritmo de "ray tracing" em si. Somente o pré-processamento inerente ao algoritmo paralelo (ausente no algoritmo da Solução de Referência Sequencial) constitui sobrecarga. Em geral, refere-se a etapas de particionamento do espaço ou da imagem, pré-amostragens, construção de hierarquias de volumes envoltórios, etc., segundo cada caso.

Outro aspecto importante do pré-processamento é que frequentemente trata-se de uma etapa não paralelizável, podendo implicar em graves conseqüências para o desempenho global do algoritmo - Lei de Amdahl (AMDAHL, 1967). Por outro lado, deve-se considerar também que a etapa de pré-processamento é realizada uma única vez no algoritmo enquanto outras, como cálculo de intersecção, são repetidas milhões de vezes. Desta forma, a sobrecarga introduzida pelo pré-processamento pode economizar enormes parcelas de tempo se conseguir melhorar a eficiência de fases repetitivas.

Sobrecargas de processamento podem manifestar-se nas etapas de travessia de raios, cálculos de intersecção redundantes (fragmentação), manutenção de estruturas de dados auxiliares, etc. Em geral ocorrem em etapas paralelas do algoritmo.

A travessia de raios ("*ray traversal*") é a fase de processamento em que realiza-se cálculos para dar continuidade à trajetória de um raio. No caso de particionamento do espaço, corresponde a cálculos de intersecção com fronteiras e determinação do nó responsável pela região penetrada. Para partição de objetos consiste nos procedimentos para determinar-se a qual nó deve ser enviada mensagem com indicação de raio. É uma fase que pode ou não ser considerada sobrecarga, dependendo de sua utilização ou não no

algoritmo seqüencial que se está adaptando. De qualquer forma, não contribui diretamente para a geração da imagem e pode implicar em maior ou menor sobrecarga dependendo do tipo de particionamento utilizado.

4.4.2. Balanceamento

Quando todos os processadores num sistema (homogêneo) são uniformemente submetidos à mesma quantidade de carga computacional estes devem terminar seus respectivos processamentos aproximadamente ao mesmo tempo resultando em mínimo tempo total de processamento para a aplicação paralela.

Se algum nó permanece ocioso em alguma parcela de tempo enquanto outros nós ainda estão ocupados, o ganho de velocidade exibido pelo sistema tende a se reduzir aumentando o tempo total para finalização do trabalho. Desta forma, balanceamento de carga uniforme é uma característica a ser buscada em algoritmos paralelos em geral, e também em "ray tracing".

A carga computacional de um processador para processamento paralelo de "ray tracing" pode ser definida da seguinte forma, para particionamento de espaço (DIPPÉ & SWENSEN, 1984):

$$C_i = \gamma_i \cdot n_i \cdot r_i \quad (4.11)$$

onde: C_i Carga do processador i ;
 γ_i Complexidade média dos objetos alocados ao nó i ;
 n_i Número de objetos alocados ao nó i ;
 r_i Número de raios que chegam às células do nó i .

A carga não pode ser determinada antes da execução do algoritmo de "ray tracing" devido à dependência do parâmetro r , o que dificulta a perfeita uniformização em técnicas de balanceamento estático.

Outra métrica para carga de um processador é aquela adotada por Nemoto e Omachi (NEMOTO & OMACHI, 1986):

$$C_i = \frac{T_r}{T_w} \quad (4.12)$$

onde: T_r tempo em execução;
 T_w tempo em espera.

Esta métrica também não pode ser calculada previamente.

Em relação ao balanceamento de carga, as soluções para "ray tracing" paralelo enquadram-se em três categorias:

- *Sem Balanceamento*: a solução não prevê nenhum mecanismo para tentar uniformizar a carga entre os processadores;
- *Com Balanceamento Estático*: a alocação de carga é feita antes do início do processamento do "ray tracing" e segue alguma heurística para uniformização de carga;
- *Com Balanceamento Dinâmico*: a alocação de carga é feita durante o processamento ou alterada durante o mesmo. Também o particionamento pode ser alterado dinamicamente em busca de balanceamento da carga no sistema.

4.4.3. Comunicação

Para analisar-se as características de comunicação de um algoritmo para multicomputador é importante que se possa quantificar os seguintes parâmetros:

- Número total de mensagens geradas;
- Tamanho médio das mensagens;
- Distância média (em número de nós) percorrida pelas mensagens.

Estes parâmetros permitem avaliar a influência da topologia e da circuitaria de comunicação no desempenho do algoritmo.

Em geral, a obtenção destes parâmetros em forma analítica é complexa e podem resultar mais facilmente de ensaios ou simulações.

Outro aspecto relacionado ao padrão de comunicação do algoritmo é a conectividade. A conectividade indica quantos elos de comunicação ("*links*") deve haver em cada nó do multicomputador para que praticamente toda a comunicação no sistema seja feita entre nós vizinhos durante o processamento do algoritmo. Uma exceção é a comunicação com o processador que tem acesso à memória de quadro, usualmente acessado por todos os outros nós.

Quanto menor a conectividade requerida, melhor será a portabilidade (mantendo o desempenho) do algoritmo entre arquiteturas de diferentes topologias.

4.4.4. Uso de Memória

Este é um parâmetro importante e diz respeito à quantidade total (somatória em todos os nós) de memória do sistema requerida para processar o algoritmo para uma dada cena. Este parâmetro indica em alguma medida a redundância (devido à replicação) de dados armazenados. Pode também ser afetado pelo armazenamento de grandes estruturas de dados eventualmente necessárias a algum algoritmo.

4.4.5. Ganho de Velocidade e Eficiência

Estes são parâmetros diretamente ligados à outros como balanceamento e às diversas sobrecargas de processamento características de um algoritmo.

Tal como definidos no capítulo 3 deste trabalho, o ganho de velocidade ("*speed-up*") mede quantas vezes o algoritmo paralelo, sendo executado num multicomputador com N nós, é mais rápido em relação ao mesmo algoritmo sendo executado em um único processador. A eficiência indica qual porcentagem da capacidade total de processamento do sistema esta sendo utilizada pelo algoritmo paralelo para execução do trabalho. Estes parâmetros são importantes para determinar-se até que ponto a adição de novos processadores no sistema conduz à uma melhora correspondente no tempo de processamento. A adição de novos processadores não implica necessariamente em aumento proporcional da memória total do sistema.

4.4.6. Restrições Algorítmicas

É importante que, ao paralelizar-se um algoritmo, não se introduzam restrições a melhorias que se possa fazer no algoritmo original. No caso do "ray tracing", estas melhorias se enquadram em três classes:

- *Aumento de realismo*: preferencialmente, o algoritmo paralelo não deve impedir a implementação de técnicas como traçado de cones ("*cone tracing*"), borrão de movimento ("*motion-blur*"), antiserrilhado ("*anti-aliasing*") ou outras mencionadas no capítulo 2, que têm como objetivo melhorar a qualidade de imagem, do ponto de vista do "foto-realismo".
- *Aceleração*: novas técnicas que permitam acelerar a geração de imagens por "ray tracing" devem, de preferência, ter sua implantação facilitada nas versões paralelas do algoritmo. Grande parte das técnicas de aceleração existentes se baseiam em aspectos de coerência (de objetos, imagem, dados, etc.) que não podem ser explorados em algumas técnicas paralelas.

- *Modelamento*: uma das características importantes do "ray tracing" é a possibilidade de geração de imagens diretamente a partir de objetos modelados geometricamente segundo as mais diferentes técnicas, conforme visto no capítulo 2, bastando para isto que se possa calcular a intersecção do objeto com uma reta e a normal da superfície neste ponto. É indesejável que o algoritmo paralelo restrinja esta característica.

Poucos autores tratam deste item pela dificuldade inerente em se caracterizar estas restrições frente à diversidade de soluções propostas para as três classes acima. O tratamento a ser dado neste trabalho também será limitado.

4.5. Taxonomias de Soluções

Em seu artigo clássico sobre algoritmos para eliminação de linhas escondidas (SUTHERLAND, SPROULL & SCHUMACKER, 1974), os pioneiros da Computação Gráfica propuseram uma taxonomia para *"categorizar, comparar e contrastar os métodos usados por dez autores para solucionar o problema de linhas escondidas e assim aprender algo fundamental sobre o problema"*⁸. Este enfoque parece bastante válido também para o problema aqui abordado.

Assim como a eliminação de linhas escondidas, o "ray tracing" paralelo é um problema complexo que admite várias abordagens diferentes. Cada uma delas tem influência diversa sobre as características básicas do sistema resultante. As várias propostas sempre representam compromissos entre aspectos de desempenho. Devido à diversidade de soluções e parâmetros envolvidos, torna-se difícil a análise e escolha da solução adequada a cada caso.

A proposta de taxonomias representa um esforço de organização do conhecimento teórico e experimental adquirido sobre um dado assunto. De acordo com Roman e Cox (ROMAN & COX, 1993) *"os autores de uma taxonomia de sistemas são desafiados a selecionar princípios de classificação que diferenciem claramente entre contribuições reconhecidas ao campo de conhecimento, que propiciem uma intuição sobre o modo de funcionamento de vários sistemas e que ajudem a identificar caminhos para futuros desenvolvimentos"*⁸.

Apresenta-se a seguir, algumas propostas de classificações de sistemas paralelos para "ray tracing" e duas taxonomias. As classificações devem ser entendidas apenas como subsídio didático usado por seus autores para organização de pequenos apanhados das técnicas, usualmente apresentadas em seções introdutórias de artigos.

⁸ T. do A.

As taxonomias, conforme já mencionado, têm objetivos mais ambiciosos. Stuart Green, autor de uma das taxonomias apresentadas a seguir, utilizou-se dela para identificar uma lacuna para a qual apresentou um algoritmo original. A outra taxonomia, contribuição inédita deste trabalho, pretende corrigir algumas deficiências daquela proposta por Green e assim, se constituir em alternativa mais interessante para os pesquisadores na área de processamento paralelo de "ray tracing".

4.5.1. Classificações de Sistemas para "Ray Tracing" Paralelo

Verificou-se na literatura três diferentes classificações para propostas de processamento paralelo do algoritmo de "ray tracing": as classificações de Gaudet, de McNeill e a Classificação por Fluxo de Dados.

4.5.1.1. A Classificação de S. Gaudet et alli

Severin Gaudet e outros (GAUDET et alli, 1988) agrupam os sistemas para "ray tracing" paralelo em três classes, de acordo com o aspecto de concorrência explorado por elas (figura 4.3):

FIG. 4.3 - A classificação de Gaudet et alli'88

- *Subdivisão do Espaço Imagem*: sistemas nesta categoria alocam grupos de pontéis para cada processador.
- *Subdivisão do Espaço Objeto*: o espaço 3D é subdividido em sub-regiões e estas são alocadas aos processadores do sistema.
- *Objetos Inteligentes*: esta classe é assim denominada porque objetos são alocados aos processadores, dotando-os de "inteligência" para que calculem sua intersecção com raios, em paralelo com outros objetos.

4.5.1.2. A Classificação de M. McNeill et alli

McNeill e outros (McNEILL et alli, 1992) advogam a classificação dos enfoques ao "ray tracing" paralelo em três categorias (figura 4.4):

FIG. 4.4 - A classificação de McNeill et alli'92

- *Dadoduto ("Pipelining")*: as diferentes tarefas do algoritmo de "ray tracing" são distribuídas entre os processadores.
- *Paralelismo de Pontéis*: a computação dos pontéis é realizada por diferentes processadores.
- *Paralelismo Espacial*: o cenário é dividido em diversos volumes que são distribuídos entre os processadores.

Sistemas das classes "*Subdivisão do Espaço Objeto*" e "*Objetos Inteligentes*" da classificação de Gaudet são categorizados como "*Paralelismo Espacial*" por McNeill. Por outro lado, a classe "*Dadoduto*" de McNeil não encontra equivalente na classificação de Gaudet. A "*Subdivisão do Espaço Imagem*" deste é análoga ao "*Paralelismo de Pontéis*" da classificação de McNeill.

Ambas as classificações são bastante simplistas e incompletas e por isso classificam sistemas bastante diversos na mesma categoria, limitando sua utilidade.

4.5.1.3. A Classificação por Fluxo de Dados

Didier Badouel, Thierry Priol e Kadi Bouatouch (PRIOL & BOUATOUCH, 1989), (BADOUEL et alli, 1990) apresentam uma outra classificação dos algoritmos para "ray tracing" implementados em máquinas MIMD. Esta classificação também divide os algoritmos em três classes, agora de acordo com o fluxo de dados no sistema (fig. 4.5):

- *Processamento sem fluxo de dados*;
- *Processamento com fluxo de objetos (ou por movimentação de objetos)*;
- *Processamento com fluxo de raios (ou por movimentação de raios)*.

FIG. 4.5 - A classificação por fluxo de dados para RT paralelo

Algoritmos *sem fluxo de dados* são aqueles que exigem o armazenamento completo dos dados na memória local de cada nó do sistema. Tipicamente enquadram-se nesta classe algoritmos do tipo "fazenda de processadores" (*processor-farm*).

Os algoritmos da segunda classe acima (*fluxo de objetos*) são aqueles que, sendo a memória local de cada nó insuficiente para armazenar a descrição completa do cenário, enviam partes destes dados, conforme necessário, a outros nós através de troca de mensagens. Corresponde basicamente às técnicas de particionamento do espaço-imagem das demais taxonomias descritas adiante. (GREEN et alli, 1989) é citada como exemplo desta classe.

A outra classe, *com fluxo de raios*, corresponde a algoritmos de particionamento do espaço-objeto. Nesta classe, os raios percorrem o espaço de processadores, sendo passados de um ao outro por troca de mensagens. Esta classe é subdividida em duas subclasses, *Árvore de Envoltórios* e *Subdivisão Espacial*. Estas classe e subclasses encontram equivalentes exatos na taxonomia apresentada na seção seguinte. Exemplos de implementações destas subclasses são (SCHERSON & CASPARY, 1988) e (DIPPÉ & SWENSEN, 1984) respectivamente.

A classificação por fluxo de dados também é bastante simples e destina-se somente a classificar sistemas existentes, sem a pretensão de servir como subsídio à procura de novas soluções. Deve-se notar que existem implementações que fazem uso tanto de fluxo de raios quanto de objetos (para balanceamento de carga), tornando as denominações empregadas nesta classificação um tanto inadequadas. De qualquer forma, a classificação apresentada é mais completa que as anteriores e enfoca um aspecto diferente e importante, o fluxo de dados no sistema.

4.5.2. A Taxonomia de Green

(GREEN, 1991) propôs uma taxonomia para classificar as soluções paralelas para "ray tracing".

A taxonomia completa está esquematizada à figura 4.6, assim como a classificação de diversas propostas de solução, segundo enquadramento do próprio autor (GREEN, 1991).

A taxonomia de Green, inicialmente, procura dividir as soluções de acordo com o tipo do espaço em que é feito o particionamento do processamento (i.e. espaço-imagem ou espaço-objeto). A seguir classifica conforme o esquema de divisão deste espaço ou com o método de alocação dos dados. Uma última subclassificação observa o tipo de balanceamento de carga proposto ou esquema de partição de objetos.

A seguir estão enunciadas definições que conceituam e explicam os termos e critérios adotados na taxonomia de Green⁹ :

- ***Espaço-Objeto***: espaço tridimensional onde estão definidos os objetos que compõem uma cena. Relaciona-se ao espaço e aos objetos do cenário.
- ***Espaço-Imagem***: espaço bidimensional do plano (ou tela) virtual onde se forma a imagem (plano de projeção).
- ***Partição de imagem***: soluções que adotam particionamento do espaço-imagem são aquelas em que cada pontel é calculado por um único processador.
- ***Partição de objetos***: técnica na qual os objetos da cena são distribuídos aos processadores de forma que cada um seja processado por um único processador.
- ***Partição do espaço***: técnica na qual o espaço-objeto é dividido em sub-regiões, cada uma atribuída a um único processador (ou aglomerado ("*cluster*") de processadores). Cada um processará os objetos contidos nas sub-regiões a ele alocadas.
- ***Balanceamento de carga***: técnica usada para fazer com que as cargas de processamento impostas a cada processador de um sistema sejam equivalentes, de forma que não se tenham nós desocupados ou sobrecarregados. Pode ser ***Dinâmico***, com alocação de tarefas mudando durante a execução do algoritmo, ou ***Estático***, se esta alocação é feita previamente e não pode ser alterada durante o

⁹ Conforme interpretados por nós, já que seu autor não explicitou todos os termos empregados.

processamento da cena. O balanceamento pode, ainda, não ser levado em consideração pela proposta.

FIG. 4.6 - Taxonomia de soluções paralelas para RT proposta por S. Green

- ***Duplicação do Banco de Dados***: os dados referentes à descrição geométrica e fotométrica de um cenário estão replicados totalmente em todos os processadores de um sistema.
- ***Banco de Dados Distribuído***: os dados de descrição da cena encontram-se distribuídos pelos vários nós do sistema de forma que um processador *não* contém, em sua memória local, a descrição completa do cenário.
- ***Dadoduto "Pipelining"***: os raios atravessam um duto onde, a cada estágio, é calculada sua intersecção com um objeto diferente da cena.

- **Hierarquia Distribuída:** os objetos são divididos e agrupados segundo uma estrutura hierárquica. Agrupamentos de níveis superiores em geral contém outros agrupamentos menores que, por sua vez, contém subgrupos de objetos.
- **Divisão Quadro-a-Quadro:** cada processador calcula um quadro de uma animação.
- **Vetorização:** a paralelização é feita para uma máquina vetorial. Geralmente, um "feixe" de raios é testado para intersecção com um único objeto, i.e., as intersecções de vários raios são calculadas simultaneamente para um único objeto num processador vetorial.

Até onde alcançou a pesquisa bibliográfica do presente trabalho, esta é a única taxonomia proposta até então para a classificação de soluções de "ray tracing" paralelo. Apesar de seu mérito comprovado, acreditamos que esta proposta apresenta algumas deficiências, discutidas a seguir.

Dentro da concepção de utilizar-se a taxonomia como subsídio à procura de novas soluções, a proposta de Green é deficiente por duas razões: primeiro porque adota o balanceamento como critério de classificação (sete das onze classes). Existem diversas formas para atuar sobre o balanceamento do sistema, seja ele estático ou dinâmico. Assim, dois sistemas que utilizem técnicas distintas de balanceamento estático poderão ser enquadrados na mesma classe apesar de apresentarem características bastante diferentes, mesmo quanto ao próprio desempenho do balanceamento. Além disto, a simples menção de, por exemplo, "balanceamento dinâmico", não deixa claro, de forma alguma, como deverá ser implementado o algoritmo para a obtenção de balanceamento dinâmico.

A segunda razão pela qual a taxonomia de Green é deficiente como auxílio à procura de novas soluções é o fato de que ela é fechada, isto é, todas as classes já têm propostas representantes. É portanto uma taxonomia que surgiu para classificar os sistemas propostos, não admitindo ramificações. A única lacuna existente foi preenchida pelo próprio autor, no trabalho que a introduziu.

A inadequação da taxonomia de Green como guia para a busca de novas soluções parece-nos causada principalmente porque suas origens foram as próprias propostas de "ray tracing" paralelo existentes quando de sua criação, e não as características fundamentais deste tipo de algoritmo paralelo.

Em nossa visão, outros pontos negativos da proposta de Green são a heterogeneidade das classes e o fato de que algumas propostas (p. ex. Salmon &

Goldsmith, 1988 e Scherson & Caspary, 1988) sejam classificadas como pertencentes a mais do que uma categoria (fig. 4.6).

4.5.3. Uma Nova Taxonomia

Procurando desenvolver uma alternativa mais adequada para a classificação dos sistemas paralelos de "ray tracing", introduz-se aqui uma nova proposta de taxonomia que deverá cumprir os seguintes objetivos:

- Classificar enfoques algorítmicos significativamente distintos em classes diferentes;
- Contrastar as diferentes técnicas aplicáveis a cada aspecto do algoritmo;
- Permitir que a classe informe as características fundamentais de uma proposta;
- Permitir que novas propostas sejam sugeridas pela combinação de características;
- Mostrar onde estão possíveis soluções nunca antes experimentadas.

Para atingir-se os objetivos propostos, criou-se uma taxonomia que caracteriza os aspectos algorítmicos das soluções e não suas características de desempenho como balanceamento de carga ou ganho de velocidade (figura 4.7)

FIG. 4.7 - Nova proposta de taxonomia de soluções paralelas para RT

Assim, foram selecionados quatro aspectos algorítmicos para caracterizar propostas de "ray tracing" paralelo - particionamento, subdivisão, alocação e replicação de dados, cada um com as respectivas subclasses a seguir:

- **PARTICIONAMENTO**: designa qual elemento será alvo de subdivisão entre os processadores do sistema. Apresenta as seguintes subclasses:
 - *Espaço*: o espaço 3D do cenário é particionado entre os processadores;
 - *Objetos*: os objetos do cenário são particionados entre os processadores;
 - *Estrutural*: partes constituintes da estrutura do modelo geométrico do cenário são distribuídas entre os processadores. Tipicamente aplicada às modelagens CSG, octárvore ("octree") e enumeração espacial;
 - *Imagem*: grupos de pontéis são processados por diferentes processadores;
 - *Raios*: raios, mesmo que contribuindo para o mesmo pontel, podem ser processados por processadores diferentes;
 - *Funcional*: as várias tarefas constituintes do algoritmo podem ser realizadas por processadores diferentes;

- *Quadros*: numa animação, em cada processador são processados quadros diferentes;
 - *Híbrido*: sistemas que apresentam alguma combinação de duas ou mais das características anteriores.
- *SUBDIVISÃO*: indica o modo pelo qual o elemento escolhido será particionado. Apresenta as seguintes subclasses:
- *Uniforme*: os elementos são subdivididos uniformemente;
 - *Adaptativa*: os elementos são divididos de forma irregular, procurando adaptar a subdivisão a alguma característica de interesse;
 - *Dinâmica*: a subdivisão dos elementos pode ser realizada durante o processamento.
- *ALOCAÇÃO*: se refere ao modo como os elementos particionados são alocados aos processadores do sistema. Apresenta as seguintes subclasses:
- *Em Bloco*: partições vizinhas são alocadas ao mesmo processador;
 - *Distribuída*: partições "distantes" podem ser alocadas a um mesmo processador;
 - *Dinâmica*: a alocação de um elemento pode ser alterada durante o processamento.
- *REPLICAÇÃO DE DADOS*: indica qual é o nível de replicação das estruturas de dados do cenário. Apresenta as seguintes subclasses:
- *Total*: a descrição completa do cenário é replicada na memória local de cada nó de processamento ($\beta = 1$);
 - *Parcial*: apenas uma parcela dos dados é duplicada ($0 < \beta < 1$);
 - *Nenhuma*: cada processador armazena informações únicas ($\beta = 0$);

As características selecionadas constituem algumas das principais decisões que o projetista do sistema deverá tomar ao criar um novo algoritmo de "ray tracing" paralelo. Assim, a existência de alguma combinação nunca antes tentada poderá rapidamente sugerir uma nova proposta de algoritmo paralelo de "ray tracing".

Algumas características inovadoras da taxonomia aqui apresentada são:

- O particionamento de objetos é explicitamente diferenciado do particionamento do espaço e colocado no mesmo nível hierárquico deste. Como será observado no capítulo 5, as características dos particionamentos de espaço e de objetos são

bastante distintas, não se justificando que fiquem sob a mesma classificação mais geral ("Particionamento do Espaço-Objeto").

- Uso da denominação "particionamento de espaço" para designar a divisão do espaço 3D do cenário em substituição ao termo "espaço-objeto" que se aplica ao espaço tridimensional e aos objetos do cenário.
- Introduzida a denominação "particionamento estrutural", que tem características peculiares. Soluções desta classe são normalmente enquadradas como "particionamento do espaço-objeto" em outras classificações.
- Introduzida a denominação "particionamento funcional" no contexto do "ray tracing" paralelo.
- Introduzida a denominação "particionamento de raios" para adequadamente classificar propostas que se enquadram neste paradigma, porém sem necessariamente fazer uso de equipamentos vetoriais.

Nem todas as combinações têm significado ou relevância. Por exemplo, replicação de dados em particionamento de objetos ou subdivisão adaptativa em particionamento funcional, são classes para as quais dificilmente será proposta alguma alternativa, devido à impossibilidade teórica ou às desvantagens do algoritmo resultante.

Apesar da nova taxonomia ter sido criada tendo em mente implementações em multicomputadores, poderá ser usada também, com alguma eventual adaptação, para classificar propostas para sistemas multiprocessadores.

Comparando-se a nova taxonomia com aquela proposta por Green, pode-se fazer a correlação mostrada na figura 4.8. Note-se que várias categorias distintas na nova taxonomia são enquadradas na mesma classe pela taxonomia anterior, demonstrando a maior especialização da primeira.

O particionamento híbrido normalmente é classificado por Green considerando a parte mais importante da solução. Por exemplo, (GAUDET et alli, 1988) apresenta uma proposta em que claramente ocorrem particionamentos funcional e de imagem, sendo classificado por Green como "subdivisão do espaço-imagem". Assim, existem várias propostas que a nova taxonomia classifica como "híbridas" e que devem ser analisadas levando em conta as duas ou mais classes em que se enquadrariam, pois normalmente apresentam características de todas estas classes.

O balanceamento não é utilizado como critério de classificação na nova taxonomia. Os mecanismos para obtenção de balanceamento estático ou dinâmico é que determinam a classe onde será enquadrada uma proposta. Assim, propostas sem balanceamento geralmente se utilizam de subdivisão uniforme e alocação em bloco. Balanceamento estático é obtido por subdivisão adaptativa e/ou alocação distribuída. Subdivisão e/ou alocação dinâmicas proporcionam o balanceamento dinâmico.

A classe "vetorização" de Green foi ampliada, recebendo a denominação original "particionamento de raios", de forma a abrigar também propostas não necessariamente orientadas à circuitaria vetorial. Existem, por exemplo, propostas para máquinas SIMD nesta classe.

A nova classe "particionamento estrutural", contribuição inédita, realça a existência, praticamente desapercibida, de soluções antes classificadas como particionamento do espaço-objeto, que exigem o uso de certo tipo de modelamento para o cenário e que, como classe, exibem algumas características comuns entre si.

Finalmente, deve-se notar que a taxonomia proposta resume, de forma concisa, as inúmeras formas de subdivisão presentes na literatura e estrategicamente adota o critério de alocação de partições para diferenciar entre soluções que, de outra forma, seriam enquadradas na mesma classe, a despeito de apresentarem características de desempenho bastante diferentes.

FIG. 4.8 - Correspondência entre as taxonomias para "ray tracing" paralelo

Os conceitos de subdivisão e alocação de partições têm sido tratados na literatura de "ray tracing" paralelo de forma relativamente indistinta, pois quase sempre as partições são enxergues sob a forma dos dados que as representam e não como conceitos autônomos como procura-se identificar neste trabalho. Este novo enfoque traz como principal vantagem a separação de conceitos, criando classes que podem auxiliar a produção de novas soluções se usadas com "técnicas de fluência" (*brainstorming, etc*) (KIELY, 1993).

Optou-se por não restringir a combinação de categorias de cada classe para que a busca de soluções que se enquadrem em classes hoje "vazias" por serem consideradas "exóticas", possam eventualmente levar, ainda que indiretamente, a vislumbrar-se alternativas interessantes, antes ocultas. Pode-se perguntar, por exemplo, como seria uma solução para "ray tracing" paralelo que se enquadrasse na classe "particionamento funcional com alocação dinâmica". Uma implementação de particionamento funcional puro onde os nós pudessem mudar de "função" de acordo com a demanda, durante o processamento, seria uma solução inédita para esta classe.

4.5.3.1. Critérios de Classificação

Publicada uma nova proposta de paralelização do algoritmo de "ray tracing", é interessante classificá-la na taxonomia. Apresenta-se a seguir um pseudo-algoritmo para classificar propostas na nova taxonomia introduzida.

Os critérios seguintes devem ser válidos para qualquer cena processada pelo algoritmo a ser classificado. Assim, pode haver casos particulares em que mais de um critério são verdadeiros para a proposta analisada. Estes casos particulares devem ser descartados em favor dos casos (cenas) mais gerais.

A) Critérios para classificação do particionamento:

- A.1)** Se cada objeto do cenário é processado *exclusivamente* por um certo nó **então**
⇒ PARTICIONAMENTO DE OBJETOS;
senão
- A.2)** Se cada região do espaço é processada *exclusivamente* por um certo nó **então**
⇒ PARTICIONAMENTO DO ESPAÇO;
senão
- A.3)** Se cada pontel da imagem é processado *exclusivamente* por um certo nó **então**
⇒ PARTICIONAMENTO DE IMAGEM;
senão
- A.4)** Se cada raio da árvore de raios de um pontel pode ser processado por um nó diferente **então**
⇒ PARTICIONAMENTO DE RAIOS;
senão
- A.5)** Se cada quadro de uma animação é calculado por um *único* nó **então**
⇒ PARTICIONAMENTO DE QUADROS;
senão
- A.6)** Se cada elemento que compõe os objetos do cenário é processado *exclusivamente* por um nó **então**
⇒ PARTICIONAMENTO ESTRUTURAL;
senão
- A.7)** Se cada processador executa apenas uma das etapas do "ray tracing" **então**
⇒ PARTICIONAMENTO FUNCIONAL;
senão
- A.8)** o particionamento é uma combinação das formas anteriores
⇒ PARTICIONAMENTO HÍBRIDO.

B) Critérios para classificação da subdivisão:

- B.1) Se** a divisão da entidade particionada *deve* ser feita durante a execução do "ray tracing" **então**
⇒ SUBDIVISÃO DINÂMICA;
senão
- B.2) Se** a entidade particionada é sempre subdividida de maneira uniforme **então**
⇒ SUBDIVISÃO UNIFORME;
senão
- B.3) existe** alguma heurística governando a subdivisão da entidade particionada
⇒ SUBDIVISÃO ADAPTATIVA.

C) Critérios para classificação da alocação:

- C.1) Se** a alocação de subdivisões é feita durante o "ray tracing" **então**
⇒ ALOCAÇÃO DINÂMICA;
senão
- C.2) Se** a cada processador são alocadas subdivisões contíguas **então**
⇒ ALOCAÇÃO EM BLOCO;
senão
- C.3) são** alocadas subdivisões não contíguas a cada processador
⇒ ALOCAÇÃO DISTRIBUÍDA.

D) Critérios para classificação da replicação de dados:

- D.1) Se** todos os processadores têm todos os dados de descrição do cenário **então**
⇒ REPLICAÇÃO TOTAL;
senão
- D.2) Se** cada dado de descrição do cenário é alocado exclusivamente a um nó **então**
⇒ NENHUMA REPLICAÇÃO DE DADOS;
senão
- D.3) existe** alguma replicação de dados
⇒ REPLICAÇÃO PARCIAL.

4.5.3.2. Classificação de Propostas Publicadas

Para o desenvolvimento da presente dissertação foram analisadas mais de 40 propostas de processamento paralelo de "ray tracing". Estas propostas foram classificadas na nova taxonomia (tabela 4.9), segundo os critérios do item anterior.

Na classificação da tabela 4.9 alguns trabalhos apresentam mais de uma proposta, podendo, cada uma, ser classificada em classes diferentes. Em geral, deu-se preferência por classificar o trabalho na categoria em que o autor obteve o melhor resultado ou em categoria que não apresentava nenhuma outra solução.

Nas classes de particionamento híbrido, por razões de concisão e espaço, classificou-se as características de subdivisão, alocação e replicação de dados de apenas um dos particionamentos adotados na solução.

PARTIC.	SUBDIV.	ALOC.	REPLIC.	PROPOSTAS
ESPAÇO	UNIFORME	BLOCO	PARCIAL	(Cleary et alli, 1983)
		DISTRIBUÍDA	PARCIAL	(Pitot, 1993)
		DINÂMICA	PARCIAL	(Nemoto & Omachi, 1986)
	ADAPTATIVA	BLOCO	PARCIAL	(Boothe, 1989), (Priol & Bouatouch, 1988,89)
	DINÂMICA	BLOCO	PARCIAL	(Dippé & Swensen, 1984)
OBJETOS	UNIFORME	DISTRIBUÍDA	NENHUMA	(Pulleyblank & Kapenga,1987), (Ullner, 1983), (Williams et alli, 1991)
	ADAPTATIVA	BLOCO	PARCIAL	(Caspary & Scherson, 1989)
			NENHUMA	(Salmon & Goldsmith, 1988)
		DINÂMICA	PARCIAL	(Scherson & Caspary, 1988)
ESTRUTURAL	UNIFORME	BLOCO	NENHUMA	(Marsh, 1988)
		DISTRIBUÍDA	NENHUMA	(Kedem & Ellis, 1984)
IMAGEM	UNIFORME	BLOCO	TOTAL	(Holliman et alli, 1989)
			PARCIAL	(Badouel et alli, 90)
		DISTRIBUÍDA	TOTAL	(Woodwark, 1984)
			PARCIAL	(Green, 1991)
			NENHUMA	(Crow, 1989)
		DINÂMICA	TOTAL	(Barton, 1988), (Bloomer, 1992), (May & Shepherd, 1987), (Muus, 1991), (Packer, 1987), (Ransen, 1990), (Kung et alli, 1991)
			PARCIAL	(Green et alli, 1988, 89), (Holliman et alli, 1993)
RAIOS	UNIFORME	DISTRIBUÍDA	NENHUMA	(Williams et alli, 1988)
		DINÂMICA	PARCIAL	(Passmore, 1987), (Plunket & Bailey, 1985)
FUNCIONAL	UNIFORME	BLOCO	NENHUMA	(Kunii et alli, 1989)
QUADROS	UNIFORME	DINÂMICA	TOTAL	(Stöber et alli, 1988)
HÍBRIDO	UNIFORME	DISTRIBUÍDA	PARCIAL	ESPAÇO+RAIOS: (Kobayashi et alli, 1988) IMAGEM+ESTRUT.: (Montani et alli, 1992)
			NENHUMA	FUNC.+IMAGEM: (Gaudet et alli, 1988)
	UNIFORME	DINÂMICA	TOTAL	FUNC.+IMAGEM: (Deguchi et al., 1984), (Nishimura et al., 1983), (Naruse et al., 1987)
			PARCIAL	ESTRUT.+IMAGEM: (Montani et alli, 1992)
	ADAPTATIVA	BLOCO	PARCIAL	ESPAÇO+FUNC.: (Kobayashi et alli, 1987)
		DINÂMICA	PARCIAL	ESPAÇO+FUNC.: (Lefer, 1993)
	DINÂMICA	DINÂMICA	TOTAL	FUNC.+RAIOS: (Morris & Dew, 1989)

TAB. 4.9 - Classificação de propostas na nova taxonomia

Se combinadas, as 8 classes de particionamento e as 3 classes de subdivisão, alocação e replicação de dados, chega-se a um total de 216 possíveis subclasses onde pode-se enquadrar uma solução paralela de "ray tracing", sem contar as infindáveis combinações possíveis através de particionamentos híbridos. Conforme será abordado no capítulo 6, as propostas hoje existentes abrangem uma fração relativamente pequena deste universo, expondo a grande oportunidade existente para a criação de novas soluções para o "problema de 'ray tracing' paralelo em multicomputadores".

Esperamos que o uso e análise da taxonomia apresentada aqui possa auxiliar outros pesquisadores da área a explorar novas alternativas que se aproximem ainda mais da Solução Ideal.

CAPÍTULO 5

O DESEMPENHO DO ALGORITMO DE 'RAY TRACING' EM MULTICOMPUTADORES

5. O DESEMPENHO DO ALGORITMO DE "RAY TRACING" EM MULTICOMPUTADORES

Neste capítulo serão apresentadas características de desempenho de implementações do algoritmo de "ray tracing" em multicomputadores.

As informações deste capítulo são resultado da compilação e análise de dezenas de artigos que apresentam propostas de solução ao problema de "ray tracing" paralelo. Cada um destes artigos introduz alguma nova característica, que o posiciona como original, porém mantendo uma origem comum com as técnicas básicas, apresentadas aqui, e classificadas na nova taxonomia. Por razões de espaço e concisão, apresenta-se neste capítulo somente uma síntese do texto completo publicado em (SANTOS, 1994b).

As seções seguintes contêm gráficos de barras para comparação das características das várias classes de propostas. É importante explicitar que os valores constantes nestes gráficos têm apenas caráter subjetivo e qualitativo já que o desempenho de uma proposta qualquer é dependente de grande número de parâmetros envolvendo a complexidade da cena e características do multicomputador entre outras, como será visto nos próximos capítulos. Sua função é permitir ao leitor uma avaliação rápida do desempenho das soluções em cada classe.

Grande parte das técnicas de paralelização de "ray tracing" são baseadas em metodologias usadas para aceleração de algoritmos sequenciais. Muitas destas metodologias adaptam-se bem em arquiteturas de memória distribuída, como será visto a seguir.

5.1. Particionamento do Espaço

Propostas que se enquadram na categoria de particionamento do espaço são, geralmente, implementações paralelas da técnica de aceleração relatada no item 2.4.3.2.1 que toma-se como base para a descrição da implementação paralela.

O conjunto de células, resultante da subdivisão do espaço mencionado àquele item, agora é distribuído aos processadores do multicomputador.

A propagação de raios entre células alocadas a diferentes processadores é feita através do envio de mensagens. Estas mensagens devem conter, além da indicação da direção e origem do raio, as coordenadas do pontel a cuja árvore de raios pertencem para posterior composição de contribuições.

Ao determinar uma intersecção, o processador realiza cálculos de iluminação que são enviados juntamente com os raios refletido e/ou transmitido se houverem. O processador que identifica a última intersecção de um raio, envia o resultado final da composição de contribuições de iluminação ao nó que tem acesso à memória de quadro ou ao sistema de arquivos.

As sobrecargas de processamento associadas com o particionamento do espaço concentram-se nas fases de particionamento, feita como pré-processamento, e na travessia de raios.

Balancear a carga dos processadores nesta classe de soluções é mais difícil que nos algoritmos que adotam particionamento de imagem. Soluções classificadas como particionamento do espaço com subdivisão *uniforme* e alocação *em bloco*, não dispõem de nenhum mecanismo de balanceamento. Balanceamento estático pode ser obtido pela subdivisão *adaptativa* ou com a adoção da alocação *distribuída* na subdivisão *uniforme*. Subdivisão e/ou alocação *dinâmicas* proporcionam balanceamento dinâmico.

As mensagens comunicadas em sistemas com particionamento de espaço geralmente são locais, curtas e em grande quantidade (milhões) pois equivalem aos raios que atravessam o espaço particionado.

A fragmentação (cf. item 2.4.3.2.1), que em implementações sequenciais causa apenas sobrecarga de processamento, provoca também replicação de dados em soluções paralelas de particionamento do espaço. Por isso, praticamente qualquer solução desta classe apresentará replicação *parcial* dos dados.

As metodologias que subdividem o espaço 3D se adaptam melhor a cenas nas quais os raios interceptam rapidamente os objetos, sem tráfegar por demais entre eles, conforme exemplificado na fig. 5.1 (SCHERSON & CASPARY, 1987).

FIG. 5.1 - Influência da distribuição de objetos para partic. do espaço

Os resultados apresentados em trabalhos que propõem o particionamento do espaço (BOOTHE, 1989), (PRIOL & BOUATOUCH, 1988, 1989), (BADOUEL et alli, 1990), (PITOT, 1993) mostram que a eficiência deste tipo de metodologia é insatisfatória (de 12% a 71%), dificilmente atingindo valores acima de 50% quando se usam todos os processadores do multicomputador disponível.

5.1.1. Subdivisões Uniforme, Adaptativa e Dinâmica

O particionamento do espaço admite as três formas de subdivisão da taxonomia: *uniforme*, *adaptativa* e *dinâmica*. A figura 5.2 exemplifica os três casos.

FIG. 5.2 - Subdivisão no particionamento do espaço

A subdivisão *uniforme* basicamente divide o espaço do cenário em células com forma de paralelepípedos. O tamanho da célula é normalmente definido pelo número máximo de objetos em cada célula (CLEARY et alli, 1983).

A subdivisão *adaptativa* reparte o espaço em células disjuntas chamadas "cubos genéricos" que não precisam manter a perpendicularidade entre suas arestas (DIPPÉ & SWENSEN, 1984) ou adota algum tipo de hierarquia no particionamento, por exemplo, em forma de octárvore (GLASSNER, 1984). (BOOTHE, 1989) apresenta um trabalho no qual é feita subdivisão *adaptativa*, dividindo de forma uniforme em dois níveis. A heurística adotada é o número de objetos por célula. (PRIOL & BOUATOUCH, 1988, 1989) fazem subdivisão *adaptativa* usando particionamento binário do espaço (BSP) e uma pré-amostragem da carga das sub-regiões para guiar o particionamento. Além destas, ainda

outras heurísticas são possíveis para determinar nova subdivisão, tal como área da célula, etc. (MacDONALD & BOOTH, 1990).

As fronteiras das células que demarcam o particionamento com subdivisão *dinâmica* são movimentadas durante a execução do "ray tracing" sob o controle de algum algoritmo de monitoramento e redistribuição de carga (DIPPÉ & SWENSEN, 1984).

A técnica de subdivisão adotada para o particionamento do espaço influencia de forma diferente ao desempenho final do sistema, conforme mostra a figura 5.3:

FIG. 5.3 - Efeito do tipo de subdivisão no particionamento do espaço

Para efeito dos comentários apresentados nos sub-ítems a seguir, supõe-se implementações de particionamento do espaço com alocação em bloco e replicação parcial de dados.

A) Sobrecarga de Processamento

O pré-processamento requerido para *subdivisão uniforme* é bastante pequeno pois consiste apenas na verificação de quais células do particionamento são penetradas pelos objetos do cenário ($\approx 1\%$ do tempo total de processamento - (CLEARY et alli, 1983)). A *subdivisão adaptativa* pode requerer pré-processamento mais custoso, principalmente se a heurística usada para a subdivisão for sofisticada. Normalmente, o particionamento inicial da *subdivisão dinâmica* é uniforme impondo a mesma sobrecarga deste tipo de subdivisão, na fase de pré-processamento.

Quanto a sobrecargas de processamento, pode-se dizer que aquela apresentada pela subdivisão *uniforme* é bastante baixa pois podem ser usados algoritmos do tipo 3D-DDA - *3D Digital Differential Analyzer* (FUJIMOTO & IWATA, 1985) para a travessia de raios. Esta técnica é uma adaptação para 3 dimensões do algoritmo de traçado de linhas de Bressenham (BRESSENHAM, 1982), e requer apenas aritmética incremental inteira.. Por outro lado, pelo fato de a subdivisão *uniforme* não levar em conta o posicionamento dos

objetos, pode ocorrer grande fragmentação, aumentando em várias vezes o número lógico de objetos no cenário. Outras técnicas alternativas para travessia de raios são apresentadas em (AMANATIDES & WOO, 1987), (CLEARY & WYVILL, 1988), (MONTANI & SCOPIGNO, 1990), (HSIUNG & THIBADEAU, 1992) e (SLATER, 1992).

As subdivisões *adaptativas*, implicam por um lado, em maior sobrecarga pois a travessia dos raios é mais complexa e, por outro, em alívio de sobrecarga pois os raios percorrem menor número de células. Além disso, a fragmentação é reduzida pois a subdivisão se adapta ao posicionamento dos objetos no cenário.

A subdivisão *dinâmica* requer monitoramento e sofisticado gerenciamento de redistribuição de carga para que seja eficiente como técnica de balanceamento, evitando oscilações e outros problemas que incidem nesta técnica. Assim, normalmente são produzidas grandes sobrecargas como resultado da tentativa de balanceamento de carga objetivada por este tipo de subdivisão.

B) Balanceamento de Carga

O balanceamento proporcionado pela subdivisão *uniforme* é bastante insatisfatório (considerando-se alocação em bloco). (BADOUEL et alli, 1990) verificaram carga média de apenas 20% em implementação desta classe. O mesmo trabalho, agora usando um algoritmo com subdivisão *adaptativa*, relata obtenção de carga média de 82.5%, mostrando a maior adequação desta técnica para fins de balanceamento.

A finalidade da subdivisão *dinâmica* é favorecer o balanceamento de carga *durante* o processamento já que é difícil prever a carga a priori. Infelizmente, a única referência à subdivisão *dinâmica* em particionamento do espaço, pesquisada neste trabalho (DIPPÉ & SWENSEN, 1984), não foi implementada nem simulada por seus autores, de forma que não se tem nenhum resultado conclusivo sobre seu desempenho para balanceamento de carga. De qualquer forma, pela dificuldade em se obter métricas precisas para a carga e heurísticas que permitam redistribuí-la adequadamente, não se vêem boas perspectivas de sucesso para esta técnica.

C) Comunicação

Desde que a subdivisão seja adaptada à topologia do multicomputador (fig. 5.4), a comunicação apresentada por algoritmos de particionamento do espaço com subdivisão *uniforme* tende a ser local, já que os raios saem de uma célula e penetram em outra, sempre vizinha.

FIG. 5.4 - Partic. do espaço adaptado à topologia do multicomputador

A adoção de subdivisão *adaptativa* na forma hierárquica, faz com que o número de células adjacentes a outra possa ser muito maior do que quatro ou seis (ex. na fig. 5.5). Isto impede que a comunicação se faça apenas entre nós vizinhos, em multicomputadores com conectividade limitada, aumentando a sobrecarga imposta pela comunicação no sistema.

FIG. 5.5 - Célula com nove vizinhos

A maior desvantagem no emprego de subdivisão *dinâmica* é o grande aumento na sobrecarga de comunicação. Este aumento se deve à geração de longas mensagens para transmissão de descrição de objetos, quando se movimenta as fronteiras do particionamento. Também maior número de mensagens curtas são geradas para fins de monitoramento de carga e controle.

D) Memória

A subdivisão *uniforme* causa maior fragmentação de objetos e por isso apresenta maior replicação de dados do que as subdivisões *adaptativa* e *dinâmica*. Por outro lado, as subdivisões *adaptativas* em geral requerem o armazenamento de estruturas de dados auxiliares como a octárvore, por exemplo, aumentando a demanda de memória. A subdivisão *dinâmica* exige o armazenamento temporário de dados relativos a carga do sistema.

E) Ganho de Velocidade e Eficiência

Resultados publicados indicam que a eficiência de implementações de particionamento do espaço é, em geral, baixa.

Para subdivisão *uniforme*, (Kobayashi et alli, 1988) obtiveram eficiência monotonicamente decrescente com o aumento do número de processadores do sistema, demonstrando a pouca escalabilidade desta técnica.

A subdivisão *adaptativa* também não mostrou resultados satisfatórios com relação à eficiência: (BADOUEL et alli, 1990) obteve 53% de eficiência como melhor resultado processando cenas em multicomputador com 32 processadores. Não se tem resultados para a subdivisão *dinâmica*.

As razões para a baixa eficiência das implementações de particionamento do espaço são (BADOUEL et alli, 1990):

- *sobrecarga de travessia*: o aumento do número de subregiões também faz com que aumente a sobrecarga para travessia dos raios.
- *fragmentação*: a fragmentação aumenta demasiadamente com o refinamento da subdivisão.
- *aumento na razão comunicação/processamento*: com o aumento do número de processadores as células da partição tornam-se menores, reduzindo a carga associada à cada uma. Por outro lado aumenta a comunicação inter-processadores.
- *pontos quentes (hot spots)*: nós que contém fontes de luz recebem mais mensagens que os outros. Isto causa desbalanceamento de carga e congestionamento da rede próximo a estes nós diminuindo a eficiência do sistema.

F) Restrições Algorítmicas

Independentemente do esquema de subdivisão adotado, o particionamento do espaço só é adequado para técnicas de aceleração baseadas neste tipo de particionamento. A implementação de algoritmos que façam uso de coerência de imagem ou raios é bastante dificultada com a adoção deste particionamento.

5.1.2. Alocações em Bloco, Distribuída e Dinâmica

As células ou subregiões resultantes do particionamento do espaço podem ser alocadas aos nós de processamento do sistema de forma *distribuída*, *em bloco* ou *dinamicamente*.

A alocação *em bloco* é feita segundo o exemplo mostrado na figura 5.6a, para um sistema em topologia em malha 3x3. Cada processador (numerados de 0 a 8) recebe um bloco contíguo de células. Na alocação *distribuída* (fig. 5.6b) cada bloco de nove células é processado por todo o conjunto de processadores.

Na alocação *dinâmica*, células inicialmente alocadas a um determinado nó podem ser transferidas a outros, durante o processamento, com a finalidade de balancear a carga no sistema. A subdivisão do espaço não sofre modificação neste processo; somente a alocação de células do particionamento é alterada.

FIG. 5.6 - Alocação de células aos processadores para partic. do espaço

A figura 5.7 ilustra a influência relativa das técnicas de alocação *em bloco*, *distribuída* e *dinâmica* sobre o desempenho do sistema.

FIG. 5.7 - Efeito do tipo de alocação no particionamento do espaço

A) Sobrecarga de Processamento

A sobrecarga de processamento causada pela fragmentação internamente a um nó pode ser parcialmente evitada no caso de alocação *em bloco*, com o uso da técnica "caixa postal" (ARNALDI et alli, 1987) usada também em implementações sequenciais. Esta técnica permite que não sejam duplicados cálculos de intersecção com objetos que ocupem mais de uma célula, desde que as células estejam alocadas ao mesmo processador. Na alocação *distribuída*, células vizinhas são sempre alocadas a processadores diferentes, inviabilizando o uso desta técnica.

A alocação *dinâmica*, assim como a subdivisão *dinâmica*, exige monitoramento de carga para controle da alocação. No entanto, a sobrecarga de processamento imposta pela alocação *dinâmica* é menor do que a exigida pela subdivisão *dinâmica* por que menos decisões precisam ser tomadas e pode ser implementada com um algoritmo mais simples.

B) Balanceamento de Carga

A alocação *distribuída* pode ser usada como técnica de balanceamento estático. Resultados simulados por Kobayashi (KOBAYASHI et alli, 1988) mostram uma melhora de 2 a 3 vezes no tempo de processamento quando troca-se a alocação *em bloco* pela *distribuída*.

A alocação *distribuída* só será eficiente como mecanismo de balanceamento de carga se o grau de subdivisão do espaço for adequado. A divisão do espaço em poucas células (i.e., células grandes) faz com que a alocação *distribuída* torne-se equivalente à alocação *em bloco*. A fig. 5.8 ilustra resultados obtidos por Paul Pitot (PITOT, 1993), variando-se a dimensão lateral das células em particionamento do espaço alocado a 32 nós.

FIG. 5.8 - Balanceamento de carga em função do tamanho da sub-região

A alocação *em bloco*, quando não acompanhada de outros mecanismos de balanceamento como subdivisão *adaptativa* ou *dinâmica*, ocasiona balanceamento muito ruim (< 20%).

O trabalho de Nemoto e Omachi (NEMOTO & OMACHI, 1986) é o único que se pôde encontrar enquadrado na classe de alocação *dinâmica* com particionamento do espaço. Não se tem resultados da sua eficácia como estratégia de balanceamento.

C) Comunicação

A adoção da alocação *distribuída* gera maior número de mensagens pois um raio percorrerá maior número de processadores do que na alocação *em bloco*.

Quanto à conectividade, a alocação *em bloco* exige unicamente comunicação local. A alocação *distribuída* requer topologia toroidal para manter esta característica.

A alocação *dinâmica* de células requer a transmissão das informações geométricas e ópticas que descrevem os objetos contidos numa célula do nó ao qual estavam alocados originalmente para um novo nó que terá sua carga aumentada. Esta transmissão em geral implica na geração de poucas mensagens, porém longas.

D) Memória

Na alocação *em bloco*, apesar dos objetos serem seccionados pelo particionamento nas mesmas posições, um bloco inteiro de células estará alocado a um único nó de processamento, fazendo com que vários objetos estejam inteiramente contidos na memória local de um único nó, não causando, assim, redundância no uso de memória.

No caso da alocação *distribuída*, células contíguas, que provavelmente contêm partes da mesma superfície, estão em nós diferentes causando replicação de dados no sistema (fragmentação).

Kobayashi (KOBAYASHI et alli, 1988) observou que se os objetos estão uniformemente distribuídos pelo cenário o aumento no uso de memória pela alocação *distribuída* em relação à alocação *em bloco* é pequeno mas pode chegar ao dobro se existirem grandes objetos no cenário.

E) Ganho de Velocidade e Eficiência

Com a melhora no balanceamento provocada pela alocação *distribuída* conseqüentemente melhoram o ganho de velocidade e a eficiência no sistema. Simulações demonstraram, consistentemente, a superioridade da alocação *distribuída* neste aspecto, com eficiência da ordem de 70% (PITOT, 1993).

F) Restrições Algorítmicas

A alocação *distribuída* não impõe, intrinsecamente, outras restrições algorítmicas além aquelas já presentes pelo uso de técnicas de particionamento do espaço 3D. Os problemas causados pela alocação de células vizinhas a processadores diferentes têm de ser enfrentados para as células nas fronteiras das subregiões na alocação *em bloco*. A alocação *distribuída* só agrava quantitativamente esta questão.

5.2. Particionamento de Objetos

Existem duas estratégias principais para o particionamento de objetos: uma, em forma hierárquica, representa uma implementação paralela da técnica de aceleração por árvores de envoltórios (vide seção 2.4.3.2.2). A outra, não hierárquica, é principalmente usada em computadores com capacidade vetorial ou circuitaria dedicada.

As implementações hierárquicas são enquadradas na classe de *particionamento do espaço com subdivisão adaptativa* já que o agrupamento dos objetos se dá levando em conta sua posição no espaço para que se agrupem objetos próximos. No caso não hierárquico, os objetos são normalmente uniformemente distribuídos aos processadores.

Na técnica não hierárquica são comuns duas topologias: com hospedeiro centralizador (fig. 5.9a) onde um mesmo raio (ou pacote de raios) é testado em paralelo para interceptação com todos os objetos (WILLIAMS et alli, 1991) ou com topologia em

duto (*pipeline*) onde os raios percorrem sequencialmente os processadores, sendo testados com um objeto diferente em cada um deles (Ullner, 1983) (fig. 5.9b).

FIG. 5.9 - Particionamento de objetos - técnicas não hierárquicas

Nenhuma das publicações examinadas neste trabalho abordam em maiores detalhes a questão da alocação no particionamento de objetos, normalmente sendo adotado um esquema *em bloco* no caso hierárquico ou *distribuído* no caso não hierárquico, por serem os mais naturais.

A grande vantagem do particionamento de objetos é a total ausência de fragmentação, evitando replicação de dados. No entanto, algumas estratégias apresentam duplicação de estruturas auxiliares para favorecer o balanceamento de carga no sistema.

5.2.1. Subdivisões Uniforme e Adaptativa

A grande desvantagem da subdivisão *uniforme* no particionamento de objetos é que as soluções geralmente executam o chamado "ray tracing" exaustivo onde todos os objetos são testados para intersecção com todos os raios. A subdivisão *adaptativa* levando a técnicas hierárquicas, permitem redução considerável no número de objetos testados por raio.

A figura 5.10 contrasta o desempenho de implementações de particionamento de objetos com subdivisão *uniforme* (não hierárquica) e com subdivisão *adaptativa* (árvore de envoltórios).

FIG. 5.10 - Efeito da subdivisão no particionamento de objetos

Nos itens a seguir, é comparada a subdivisão adaptativa sem replicação de dados com a subdivisão uniforme.

A) Sobrecargas de Processamento

A sobrecarga de pré-processamento, relativa à construção da árvore de envoltórios para subdivisão *adaptativa*, equivale de 3% a 4% do tempo total de processamento em implementação sequencial (SCHERSON & CASPARY, 1987). Esta sobrecarga é ausente na subdivisão *uniforme*.

Quando a técnica de aceleração por árvore de envoltórios é implementada sequencialmente, pode-se ordenar os volumes interceptados por ordem de proximidade e testá-los assim, de forma a parar o processamento mais cedo em caso de detecção de uma intersecção válida. Quando os volumes estão alocados a processadores diferentes esta ordenação é apenas parcial, gerando cálculos desnecessários. A sobrecarga induzida é de aproximadamente 5% (SCHERSON & CASPARY, 1988). Por outro lado, conforme mencionado, o uso de subdivisão *uniforme* (técnica não hierárquica) pode significar a execução de "ray tracing" exaustivo, aumentando extraordinariamente a demanda computacional do algoritmo.

B) Balanceamento de Carga

É relativamente fácil a obtenção de balanceamento de carga adequado quando se utiliza a subdivisão *uniforme* no particionamento de objetos pois a carga pode ser conhecida a priori já que esta será função do número e complexidade dos objetos alocados a cada processador (eq. 4.11). Como todos os raios deverão ser testados para todos os objetos, um fator de escala comum é aplicado às cargas de todos os nós.

Em particionamento de objetos, a subdivisão *adaptativa* tem a finalidade de agrupar objetos próximos para otimizar-se o processamento e não o balanceamento de carga. Isto

faz com que seja difícil obter balanceamento neste tipo de implementação, sem recorrer-se a replicação de dados. (SCHERSON & CASPARY, 1988) apresentam um exemplo onde a carga máxima de um nó em multicomputador com 20 processadores chegou a ser 18 vezes maior que a carga média no sistema, comprovando a dificuldade de balanceamento neste esquema.

C) Comunicação

Nas propostas publicadas, a adoção de subdivisão *uniforme* implica na transmissão de pacotes de tarefas, contendo raios a processar e pacotes de resultados, contendo os respectivos pontos de intersecção com objetos. Dessa forma, partes intermediárias do processamento são transmitidas entre os processadores, aumentando o número de mensagens. O tamanho dos pacotes deve ser cuidadosamente escolhido para que a granularidade das tarefas não prejudique o balanceamento de carga no sistema.

A conectividade requerida por propostas deste tipo é baixa, tanto na configuração em duto como para aquela com hospedeiro centralizador. O fato de este último demandar elos com todos os outros nós ocorre em qualquer algoritmo para o nó conectado à memória de quadro.

Em algoritmos que implementam a subdivisão *adaptativa* sem replicação de dados, todo o processamento é dirigido por dados, gerando grande comunicação. A conectividade depende do tipo de árvore adotada, sendo igual a 3 (três) para árvores binárias.

D) Uso de Memória

O particionamento de objetos não apresenta fragmentação, evitando a redundância de dados causada por este fenômeno.

A estrutura de dados da árvore de envoltórios usada na subdivisão *adaptativa* consome, em média, mais 30% da quantidade de memória já ocupada pela descrição dos objetos do cenário (GOLDSMITH & SALMON, 1987).

E) Ganho de Velocidade e Eficiência

Não se dispõe de dados experimentais acerca da eficiência de implementações de particionamento de objetos, porém pode-se fazer algumas considerações.

Em técnicas de subdivisão *uniforme* (não hierárquicas) o gargalo do sistema estará no nó centralizador ou na sobrecarga da capacidade de comunicação do sistema.

Devido ao mau balanceamento exibido pela implementação de subdivisão *adaptativa* (árvore de envoltórios) sem replicação de dados, pode-se prever baixa eficiência desta técnica com o aumento do número de nós do multicomputador. Este aumento faz com que o número de objetos alocados a cada processador seja reduzido, agravando o problema.

F) Restrições Algorítmicas

As implementações com subdivisão *uniforme* restringem o uso da maioria das técnicas baseadas em coerência pois os objetos individuais são alocados aos processadores de forma aleatória.

As técnicas *hierárquicas* implicitamente exigem a implementação, pelo menos a nível global, de um algoritmo de árvore de envoltórios. Outra restrição algorítmica desta técnica é a impossibilidade de uso completo da estratégia de fila de prioridade para processamento dos envoltórios em ordem adequada (cf. 2.4.3.2.2), já que na implementação paralela esta fila não é única.

5.2.2. Replicação de Dados

A replicação de dados no particionamento de objetos será abordada considerando-se a subdivisão *adaptativa*, onde é possível replicarem-se dados para favorecer o balanceamento e a comunicação. Implementações com subdivisão *uniforme* implicitamente não apresentam replicação de dados.

Considerando-se uma implementação em árvore de envoltórios, a replicação de dados é feita escolhendo-se uma certa profundidade na árvore (NÍVEL) até a qual todos os envoltórios são replicados em todos os nós de processamento (SCHERSON & CASPARY, 1988) - figura 5.11. Deste nível em diante, as sub-árvores são alocadas somente a processadores específicos.

(SALMON & GOLDSMITH, 1988) implementam um sistema hierárquico, sem replicação de dados.

FIG. 5.11 - Replicação de dados em hierarquia de envoltórios

A figura 5.12 compara os efeitos da replicação de dados sobre os parâmetros de desempenho do sistema.

FIG. 5.12 - Efeito da replicação de dados no particionamento de objetos

A) Sobrecarga de Processamento

Praticamente não há diferenças entre as sobrecargas de processamento das duas implementações. A replicação de dados permite reduzir um pouco o efeito negativo da distribuição das filas de prioridade.

B) Balanceamento

O balanceamento de carga na implementação *com replicação de dados* é muito dependente do nível da árvore em que se faz a replicação de envoltórios. A figura 5.13 mostra o balanceamento em função deste parâmetro (SCHERSON & CASPARY, 1988) para sistema com 20 processadores.

Com o aumento do número de processadores, o balanceamento pode ficar prejudicado por haverem poucas sub-árvores para cada nó. Na ocorrência deste fenômeno,

é sugerido o balanceamento dinâmico através da transferência de sub-árvores (e dos objetos em seus nós-folha), para processadores menos sobrecarregados. Scherson assume que esta transferência implique em sobrecarga de aproximadamente 20% da carga do processador mais sobrecarregado.

FIG. 5.13 - Balanceamento de carga em função do NÍVEL de distribuição

De forma geral, pode-se dizer que o balanceamento de carga obtido no sistema é bastante bom devido à existência de dois processos em cada processador, um dirigido por dados (árvores inferiores) e outro por demanda (árvore superior). Se um processador estiver sobrecarregado com o processamento de sua sub-árvore, pouco processamento será feito por ele sobre a árvore superior. Outros processadores deverão efetuar estes cálculos. Outro aspecto positivo desta proposta é que qualquer processador pode iniciar os cálculos de intersecção de raios com a raiz da árvore, gerando uma demanda de processamento, sem sobrecarregar necessariamente nenhum nó do sistema.

C) Comunicação

A comunicação exibida pelo algoritmo *com replicação* de dados é inversamente proporcional ao nível de replicação e, portanto, sempre menor do que no algoritmo *sem replicação*.

D) Uso de Memória

Pode-se estimar a replicação dos dados em memória em função do parâmetro NÍVEL, considerando-se a heurística de "corte mediano" que gera árvores binárias. A partir da copa da árvore, a cada nível que se desce em direção à raiz, o dobro (menos 1) de nós são replicados nas memórias locais dos processadores, resultando em crescimento exponencial.

E) Ganho de Velocidade e Eficiência

Resultados de simulações indicam ganho de velocidade linear para até 200 processadores usando a técnica *com replicação de dados* (SCHERSON & CASPARY, 1988). A partir deste ponto, ocorre desbalanceamento de carga porque o número de sub-árvores torna-se pequeno em comparação com o de processadores, diminuindo a eficiência. Este número é maior para o processamento de cenas mais complexas.

F) Restrições Algorítmicas

O uso de redundância de dados não impõe nenhuma nova restrição algorítmica. Ao contrário, permite melhorar o uso da coerência de objetos.

5.3. Particionamento de Imagem

Neste tipo de implementação, os pontéis que constituirão a imagem a ser sintetizada são particionados entre os vários nós de processamento do sistema. Cada nó deverá realizar o processamento total dos pontéis a ele atribuídos e enviar um pacote com os pontéis calculados de volta a um processador "gerente" que, ligado à memória de quadro ou ao sistema de arquivos do multicomputador, deverá exibir ou armazenar a imagem completa.

As análises a seguir consideram subdivisão uniforme.

5.3.1. Replicação de Dados

A forma mais imediata de paralelização do algoritmo de "ray tracing" segue a estratégia "fazenda de processadores", e resulta na classe de particionamento de imagem com subdivisão uniforme, alocação dinâmica e *replicação total de dados*. Paradoxalmente, esta mesma implementação apresenta os melhores resultados em todos os parâmetros de desempenho aqui estudados, com exceção do uso de memória, onde o resultado é o pior possível - replicação *total* dos dados em *todos* os nós. Isto é necessário pois qualquer nó pode processar intersecções com qualquer objeto. Nesta implementação, um processo gerente envia aos processos escravos mensagens especificando tarefas a serem cumpridas - o cálculo de um bloco de pontéis. Os processos escravos, terminando suas tarefas, enviam ao processo gerente um pacote de pontéis já calculados, requisitando ao mesmo tempo novas tarefas.

Para tentar manter as características positivas desta implementação, reduzindo sua única desvantagem, (GREEN, 1989) e (BOOTHE, 1989) propuseram, de forma independente, uma solução com a simulação de uma memória cache para armazenar parte

dos dados do cenário. O uso de parte da memória local do nó de processamento para simulação deste cache de objetos gera soluções com replicação *parcial* de dados. Os dados de descrição do cenário ficam distribuídos pelos vários nós do multicomputador.

Uma comparação entre os parâmetros de desempenho obtidos com o uso de replicação total ou parcial de dados é mostrada na figura 5.14.

FIG. 5.14 - Efeito da replicação de dados no particionamento de imagem

A) Sobrecarga de Processamento

O uso de replicação *parcial* de dados exige uma etapa de pré-processamento para alocação dos dados aos nós do multicomputador, de forma que se possa saber, durante o processamento, onde encontrar a descrição de qualquer objeto que seja necessário para cálculo de uma intersecção.

A replicação *parcial* causa pequena sobrecarga de processamento devido às rotinas de comunicação (requisição de dados e atendimento de requisições).

B) Balanceamento

O balanceamento de carga é pouco afetado pela escolha do grau de replicação de dados no sistema. Pode ocorrer um pequeno desbalanceamento se algum nó estiver armazenando um conjunto de dados muito requisitado pelos outros nós.

C) Comunicação

É possível fazer uma implementação *sem replicação de dados* em que praticamente não haja nenhuma comunicação entre processadores durante a execução do algoritmo, atingindo o desempenho ideal neste parâmetro.

Implementações *com replicação de dados* são forçadas a realizar comunicação para obtenção de dados não presentes na memória local de um nó. A quantidade de mensagens

geradas será dependente da porcentagem dos dados do cenário disponíveis na memória de cada nó.

D) Uso de Memória

A replicação de dados em implementações com uso de cache varia de acordo com a complexidade do cenário e da memória disponível em cada nó. No caso de cenários simples, em que todas as estruturas de dados podem residir completamente na memória local de cada nó, a implementação recai no caso da *replicação total*. A replicação de dados inexistente quando as estruturas de dados requerem $N.m_i$ bytes de memória, onde N é o número de nós do sistema e m_i é a memória disponível em cada nó. Neste caso, deve-se esperar desempenho extremamente pobre em função da alta demanda de comunicação exibida pelo sistema.

Não ocorre problema de fragmentação de dados como no particionamento do espaço 3D. De certa forma, pode-se considerar que a implementação de particionamento de imagem com *replicação parcial* dos dados faz sempre o melhor uso da memória disponível, implicando em redundância somente quando existe memória disponível para tal. O compromisso velocidade x memória é automaticamente ajustado pela própria natureza da implementação.

E) Ganho de Velocidade e Eficiência

Em termos de ganho de velocidade e eficiência, a implementação de particionamento de imagem com *replicação total de dados* é praticamente imbatível. Sendo sua sobrecarga de processamento muito baixa e a demanda de comunicação mínima, é natural que os resultados de ganho de velocidade sejam praticamente lineares e unitários mesmo para número elevado de processadores.

(PACKER, 1987) relata resultados atingindo eficiência de 95.5% com 80 processadores do tipo Transputer. (BARTON, 1988) relata a obtenção de ganho de velocidade linear em um sistema semelhante ao de Packer porém com 311 nós de processamento. (RANSEN, 1990) relata eficiência de 99.5% em sistema com 16 processadores. Todos com *replicação total de dados*.

Com a adoção de metodologias de cache de objetos para reduzir o problema de replicação de dados, os resultados em termos de ganho de velocidade começam a sofrer os efeitos das sobrecargas de comunicação e processamento. Ainda assim, os resultados são bastante satisfatórios.

A fig. 5.15 mostra resultados obtidos em ensaio (GREEN, 1991) com um multicomputador de 8 nós e implementação de particionamento de imagem com *replicação parcial de dados*. Neste caso, o desempenho do sistema varia em função do tamanho do cache alocado, isto é, da porcentagem de dados replicados.

FIG. 5.15 - Eficiência no partic. de imagem com replicação parcial de dados

Com replicação total de dados o desempenho foi semelhante ao das implementações relatadas anteriormente obtendo-se eficiência de 99.6% para 8 nós de processamento. Com 10% das estruturas de dados replicadas em cada nó, obteve-se, no pior caso, eficiência de 78.9%. Para apenas 0.5% de replicação de dados por nó, ainda pôde-se obter eficiência de 50.5%.

É importante notar que a memória total utilizada na implementação acima é dada pela equação a seguir:

$$m_N = \phi \cdot M \cdot N + M \quad (5.1)$$

onde: m_N memória total para sistema com N processadores;
 ϕ fração dos dados alocados em memória cache;
 N número de nós de processamento do sistema;
 M memória total mínima, necessária para armazenar os dados do cenário.

Assim, 10% de replicação num sistema com 8 nós significa uso de 80% mais memória que a consumida pelo algoritmo seqüencial, assim como 0.5% de replicação indicam o uso de apenas 4% mais memória que o modelo ideal.

Badouel (BADOUEL et alli, 1990) fizeram experimentos com um multicomputador maior, obtendo, em média, eficiência de 89% para 64 nós de processamento. No pior caso, relatam eficiência estimada de 74% para uma cena complexa. Não é mencionada a fração de dados disponíveis localmente a cada nó.

F) Restrições Algorítmicas

Uma das características mais interessantes das soluções com particionamento de imagem é a facilidade de implementação pois poucas modificações precisam ser feitas no algoritmo sequencial para adaptá-lo ao ambiente paralelo. Desta forma, também este tipo de particionamento não impõe praticamente nenhuma restrição algorítmica exceto aquelas relacionadas com a fragmentação de imagem (veja a seguir).

5.3.2. Alocações em Bloco, Distribuída e Dinâmica

Na alocação *em bloco*, após a subdivisão da imagem, grupos de pontéis contíguos são alocados a cada nó de processamento. O número de blocos iguala o de processadores.

A alocação *distribuída* distribui a cada processador pontéis individuais espacialmente distantes ou, para aproveitamento da coerência, blocos de pontéis não contíguos.

A alocação *dinâmica* faz a alocação de carga durante o processamento, por demanda dos nós do sistema. Neste caso e na alocação *distribuída*, a subdivisão deve gerar um número de partições maior que o de nós de processamento do sistema.

A figura 5.16 compara o desempenho das várias formas de alocação no particionamento de imagem.

FIG. 5.16 - Efeito do tipo de alocação no particionamento de imagem

A) Sobrecargas de Processamento

No particionamento de imagem ocorre um fenômeno semelhante àquele da fragmentação em implementações com particionamento do espaço 3D. Denominar-se-á este fenômeno de *fragmentação de imagem*. A fragmentação de imagem reduz o aproveitamento da coerência de imagem e causa replicação de processamento dos raios

localizados nas arestas ou vértices dos pontéis, na fronteira de sub-regiões, quando se utiliza técnicas de superamostragem para evitar serrilhamento. Nestes casos, os raios disparados nestas posições são normalmente aproveitados para os pontéis vizinhos em implementações sequenciais. Na implementação paralela, estes raios deverão ser recalculados (fig. 5.17).

No exemplo da figura 5.17 a geração da imagem com resolução 4 x 4 pontéis e superamostragem (9 raios/pontel) exige o lançamento de 81 raios primários. Se fragmentada em 4 sub-regiões para processamento em 4 nós diferentes, serão lançados 100 raios primários (e todos os raios secundários correspondentes) já que a informação de alguns raios compartilhados agora estará distribuída entre os processadores, exigindo o recálculo. A sobrecarga de processamento envolvida é tanto maior quanto for a fragmentação da imagem.

Da mesma forma que a fragmentação de objetos, a fragmentação de imagem também aumenta com o uso da alocação *distribuída*.

FIG. 5. 17 - A fragmentação de imagem

B) Balanceamento

A adoção de alocação *em bloco* produz balanceamento de carga insatisfatório. Green (GREEN, 1991) verificou diferenças de carga de 30.8% a 87.5%, nos melhores casos, em um sistema com 16 processadores, usando alocação *em bloco*.

Tal como no particionamento do espaço 3D, também no particionamento de imagem o uso de alocação *distribuída* pode proporcionar balanceamento estático. Tal estratégia foi usada por vários autores (WOODWARD, 1984), (SALMON & GOLDSMITH, 1988) e (GREEN, 1991). Com a alocação *distribuída*, Green (GREEN, 1991) obteve para o mesmo sistema anterior, diferenças de carga variando de 0.1% a 1.9%, nos melhores casos de cada imagem, mostrando a eficácia desta técnica para balanceamento da carga em implementações de particionamento de imagem.

Deve ser notado que a alocação *distribuída* só é eficiente quando a granularidade das tarefas é pequena, isto é, quando o nível de subdivisão da imagem é grande, degenerando para o caso de alocação *em bloco* quando o número de sub-regiões é igual ao de processadores.

Balanceamento dinâmico no particionamento de imagem pode ser obtido através da alocação *dinâmica* de carga. Green (GREEN, 1991) verificou variações de apenas 0.3% a 1.3%, demonstrando o balanceamento quase ótimo proporcionado pela alocação *dinâmica* no particionamento de imagem.

C) Comunicação

As alocações *distribuída* e *dinâmica* reduzem a coerência de dados em sistemas com *replicação parcial*, aumentando a comunicação. Como estas alocações só são eficientes para níveis razoavelmente elevados de subdivisão da imagem a coerência de dados é bastante prejudicada pois quanto maior o nível de subdivisão da imagem, maior o tamanho do conjunto de dados de trabalho ("*working set*") necessário para o processamento das tarefas de um nó. Este fenômeno causa elevação na taxa de faltas ("*miss ratio*") do cache, aumentando a comunicação para transferência de dados.

O uso de alocação *dinâmica* implica na geração de grande número de mensagens curtas para solicitação de tarefas e envio de resultados. Esta característica é muito ruim, principalmente em sistemas com alta latência de comunicação. Pode-se fazer uso de memorizadores (*buffers*) para evitar o envio constante de pacotes de resultados. Eles seriam agrupados e enviados como uma só mensagem evitando a repetição da latência.

A granularidade das tarefas influencia de forma decisiva no desempenho global do sistema e deve ser escolhida de tal forma que haja balanceamento entre o tempo gasto em comunicação e processamento. Zuffo (ZUFFO, 1993) obteve os resultados de eficiência apresentados na figura 5.18 para o processamento de "ray tracing" com *particionamento de imagem*, alocação *dinâmica* de carga e *replicação total* de dados, evidenciando a

importância da granularidade no particionamento de imagem (eficiência x número de processadores x tamanho do pacote de pontéis).

FIG. 5.18 - Influência da granularidade na eficiência do partic. de imagem

Como a comunicação entre processadores neste tipo de implementação é inexistente ou apenas para fins de transferência de dados, nenhuma topologia específica é requerida e portanto não é exigido explicitamente nenhum grau de conectividade.

D) Uso de Memória

O tipo de alocação escolhida tem pouco efeito sobre o uso de memória do sistema.

Considerando a coerência de dados na memória cache de objetos, pode-se dizer que a *alocação em bloco* é mais favorável do que as *alocações distribuída e dinâmica*.

E) Ganho de Velocidade e Eficiência

Os resultados referentes ao balanceamento de carga obtidos com uso de *alocação em bloco* mostram que não se poderá conseguir implementações escaláveis através desta técnica.

As *alocações distribuída e dinâmica* mostram resultados mais satisfatórios. Grande quantidade de trabalhos publicados favorecem a *alocação dinâmica* como a melhor escolha para esquema de alocação de tarefas no particionamento de imagem.

F) Restrições Algorítmicas

Algoritmos anti-serrilhado que utilizam super-amostragem têm seu desempenho ligeiramente prejudicado para os pontéis pertencentes às bordas das sub-regiões da imagem, pois terão de ser calculados por mais de um processador. O uso de *alocação distribuída* ou *dinâmica* agrava este problema.

5.4. Particionamento de Quadros

O particionamento de quadros é caracterizado pelo processamento de imagens completas em cada nó do sistema. Dessa forma, este tipo de particionamento se distingue dos demais por ser aplicável apenas às situações onde se deseja gerar um conjunto de imagens e não apenas uma. Este é o caso das animações, uma aplicação bastante comum, onde cada imagem é denominada quadro. Tradicionalmente, para cada segundo de animação, é necessária a geração de 30 quadros¹⁰.

Na geração de animações é comum a movimentação dos objetos do cenário, incluindo fontes de luz e observador, de quadro para quadro. Ao iniciar-se o processamento de nova cena (cinematográfica) pode ocorrer total mudança de cenário, requerendo um banco de dados totalmente diferente. Dessa forma, os nós do sistema estarão sempre operando com dados distintos (apesar de, às vezes, bastante semelhantes). Esta característica torna necessária a disponibilidade de memória local, em cada nó, suficiente para o armazenamento completo do banco de dados do cenário de trabalho.

Apesar de os dados contidos na memória local de cada nó serem distintos, esta implementação será considerada como apresentando replicação total dos dados pois, para fins práticos, apresenta inconvenientes típicos de outras implementações com esta mesma característica.

A implementação desta técnica se faz geralmente num esquema "fazenda de processadores", com um nó gerenciando a execução das tarefas.

Deve-se notar que o particionamento de quadros não é um caso particular do particionamento de imagens com o tamanho do pacote de pontéis (partição) igual ao tamanho da imagem. Diferentemente do primeiro caso, neste último apenas uma imagem estaria sendo processada em dado instante (com todos os nós, exceto um, ociosos !) implicando no pior balanceamento de carga possível.

O particionamento de quadros é especialmente interessante para uso em multicomputadores distribuídos, interligados por rede, pelas seguintes razões:

- processadores de estações de trabalho tipicamente passam 90 a 95% do tempo em estado ocioso (edição de texto, leitura de correspondência, etc.) (GREEN, 1991);

¹⁰ Se a animação for transferida para película cinematográfica, 24 quadros/s são suficientes. Dependendo da finalidade da animação pode-se repetir duas vezes cada quadro reduzindo estes valores à metade.

- estações de trabalho modernas dispõem de boa capacidade de memória (> 16 Mbytes) permitindo a replicação de dados ou uso de memória virtual;
- suportam mecanismo de comunicação que permite fácil transferência de dados e resultados de imagem;

A tab. 5.1 apresenta uma comparação das características dos particionamentos de quadros e imagem, com especial atenção para o caso de usar-se um ambiente de processamento distribuído (rede de computadores) (BLOOMER, 1992).

	PARTICIONAMENTO DE QUADROS	PARTICIONAMENTO DE IMAGEM
VISUALIZAÇÃO INTERATIVA	não acelera	acelera
ANIMAÇÃO	acelera	acelera
SOBRECARGA PARA CONTROLE DA REDE	baixa	maior
SOBRECARGA COM BANCO DE DADOS	baixa, bancos de dados diferentes	maior, mesmo banco de dados
TOLERÂNCIA A FALHAS	granularidade espessa	granularidade fina
RECUPERAÇÃO DE FALHAS	granularidade espessa	granularidade fina

TAB. 5.1 - Comparação entre partic. de quadros e de imagem para redes de comp.

Conforme destacado na tabela 5.1 o particionamento de quadros, por não acelerar (via paralelismo) a geração individual dos quadros, não se presta como alternativa para visualização interativa. Independentemente do número de processadores disponíveis, a geração de um quadro será executada sempre no mesmo tempo.

As características de tolerância e recuperação de falhas, bastante importantes na geração de longas animações (Stöber et alli, 1988), são negativas no particionamento de quadros pois implica geralmente em desperdício de toda ou parte de uma imagem enquanto o particionamento de imagem reduz esta perda a apenas um pacote de pontéis, isto é, uma pequena fração da imagem.

Finalmente, é importante ressaltar que os resultados do particionamento de quadros devem ser avaliados considerando-se seu ambiente típico de processamento, estações de trabalho em rede, e nunca um multicomputador, mais adequado às outras modalidades de

particionamento já apresentadas. Sob este enfoque, esta metodologia pode se constituir em eficiente ferramenta para aproveitamento do grande poder de processamento disponível em redes de estações de trabalho cujas taxas de utilização são bastante baixas.

As qualidades do particionamento de quadros são mostradas na figura 5.19, abaixo:

FIG. 5.19 - Desempenho do particionamento de quadros

A) Sobrecargas de Processamento

Na prática, as implementações de "ray tracing" paralelo por particionamento de quadros equivalem a computadores executando tarefas totalmente independentes (sem cooperação entre si) porém sob uma coordenação central. A única forma de sobrecarga está relacionada com esta coordenação, e pode ser considerada bastante pequena apesar de algumas implementações alocarem um nó somente para esta atividade.

Em ambientes de processamento distribuído, normalmente o processamento de uma animação por "ray tracing" se fará como tarefa de fundo (*background*) para aproveitamento do período ocioso dos processadores da rede.

B) Balanceamento

É mais difícil obter-se bom balanceamento no particionamento de quadros pois a granularidade das tarefas é muito elevada.

Neste tipo de implementação, geralmente é feita alocação *dinâmica* e o particionamento é *uniforme* (um quadro). Pode-se prever um balanceamento com características parecidas ao de particionamento de imagens com granularidade muito grande, portanto, ruim.

Próximo ao fim do processamento de todas as imagens, quando o estoque de imagens a gerar tiver se esgotado, grande parte dos processadores tende a ficar ociosa enquanto apenas um ou alguns nós completam suas tarefas. Sendo a granularidade elevada,

a execução de uma tarefa (geração de uma imagem completa) pode levar até horas significando horas de ciclo ocioso em quase todos os processadores do multicomputador.

C) Comunicação

A comunicação em implementações deste tipo de particionamento tende a ser pequena se todos os nós têm acesso direto ao sistema de arquivos para armazenamento das imagens geradas e para a carga das estruturas de dados dos cenários de trabalho.

As mensagens são em pequeno número e muito curtas (controle) ou muito longas (envio de imagens prontas ao nó com acesso ao sistema de arquivos se este não é local).

Não é requerido nenhum grau de conectividade explícito. A comunicação se faz exclusivamente com o nó controlador.

D) Uso de Memória

Conforme mencionado, este tipo de implementação implica em disponibilidade de memória, em todos os nós, suficiente para o armazenamento completo das estruturas de dados correspondentes a um cenário, sendo equivalente a *replicação total dos dados*.

E) Ganho de Velocidade e Eficiência

Green (GREEN, 1991) relata o processamento de uma animação com duração de 2 segundos (3600 quadros) em uma rede com 30 estações de trabalho. O processamento durou 24 dias e 10 horas com tempo efetivo de processamento de 24 horas, resultando em eficiência aproximada de 80%. Este é o único resultado numérico que dispomos e, portanto, não deve ser tomado como estimativa geral.

F) Restrições Algorítmicas

Como o algoritmo executado em cada nó é praticamente idêntico ao da versão seqüencial, não existe qualquer tipo de restrição algorítmica imposta pelo particionamento de quadros, com uma exceção. Técnicas para aceleração de animações, baseadas na coerência temporal das imagens são prejudicadas pois não se têm, localmente, informações sobre outros quadros adjacentes da animação.

A alternativa de alocar-se pacotes de tarefas com várias imagens consecutivas a cada nó pode atenuar este problema, porém com graves efeitos sobre o balanceamento de carga no sistema, pois a granularidade das tarefas, já bastante alta, seria aumentada ainda mais.

5.5. Particionamento de Raios

O particionamento de raios caracteriza-se pela distribuição aleatória dos raios que trafegam no cenário para processamento pelos nós do sistema.

Esta estratégia, assim como o particionamento de quadros, inerentemente implica em replicação do banco de dados.

Apesar de o particionamento de raios ser uma metodologia de sucesso em computadores vetoriais, o mesmo não se pode dizer quanto a sua utilização em multicomputadores.

Na implementação vetorial, os raios gerados vão sendo armazenados em uma fila até que esta atinja o comprimento do vetor do computador utilizado. Neste momento, todos os raios desta fila são interceptados, simultaneamente, com cada objeto do cenário (PLUNKETT & BAYLEY, 1984). Implementações CSG também podem ter a fase de combinação de intersecções vetorizadas bastando para isto que sejam feitas na mesma ordem para todos os raios.

Conforme relatado por David Plunkett, o código vetorial exige mais operações do que o seqüencial, apesar de ser executado mais rapidamente em circuitaria vetorial. Em multicomputadores não haveria esta vantagem em velocidade e o código executaria ainda mais lentamente.

Talvez por perceber-se rapidamente a inadequação desta metodologia para uso em multicomputadores é que a literatura não traz, até onde pode verificar esta pesquisa, qualquer referência a implementação deste tipo em máquinas puramente MIMD. Preferiu-se classificar a proposta de Kobayashi (KOBAYASHI, 1988) na categoria de particionamento híbrido. Dessa forma, nos comentários que se seguem não consta nenhum resultado experimental, muito menos quantitativo; sua finalidade é apenas a completeza deste trabalho.

O desempenho do particionamento de raios é resumido na figura 5.20.

FIG. 5.20 - Desempenho do particionamento de raios

A) Sobrecargas de Processamento

A implementação vetorial desta metodologia tem a grande desvantagem de processar o "ray tracing" exaustivo já que todos os objetos são testados para intersecção com todos os raios, sendo difícil a adoção de técnicas de aceleração.

Uma implementação em multicomputador poderia ser feita num esquema "fazenda de processadores" onde os nós poderiam usar algumas das técnicas de aceleração mencionadas no item 2.4.3 deste trabalho.

Não há pré-processamento, fragmentação ou sobrecargas relacionadas a travessia de raios nesta implementação.

B) Balanceamento

Bom balanceamento de carga pode ser obtido com relativa facilidade, dada a baixa granularidade das tarefas envolvidas.

O balanceamento é do tipo dinâmico pois as tarefas (raios a processar) são alocadas dinamicamente, por demanda. O particionamento também é dinâmico pois as tarefas não são conhecidas a priori.

C) Comunicação

A comunicação neste esquema, implementado em multicomputador, representaria uma sobrecarga insuportável já que a granularidade das tarefas é muito baixa.

As mensagens são curtas e em grande quantidade. Não é requerida nenhuma conectividade explícita.

D) Uso de Memória

Cada processador deve ser capaz de processar qualquer raio. Desta forma, deve conter em sua memória local a descrição completa do cenário. Em outras palavras, esta implementação requer a replicação total das estruturas de dados e representa o pior caso.

A implementação de um esquema de cache de objetos semelhante ao proposto por Green (GREEN, 1991), poderia atenuar o problema de replicação total, porém, como não haveria qualquer tipo de coerência nos raios a processar (e portanto nos dados), as sobrecargas de comunicação e de processamento inviabilizariam esta solução.

E) Ganho de Velocidade e Eficiência

O desempenho de implementações com particionamento de raios poderia ser estimado a partir da observação do comportamento de sistemas com particionamento de imagens onde o tamanho do pacote é mínimo: 1 (um) pontel. Neste caso, como indica a figura 5.18, a eficiência fica seriamente prejudicada devido à baixa granularidade. O particionamento de raios corresponde a um pacote ainda menor: 1 (um) raio, apenas uma fração do processamento de um pontel, piorando ainda mais o ganho e a eficiência do sistema.

F) Restrições Algorítmicas

Qualquer algoritmo que faça uso de coerência de imagem, da árvore de raios, ou de acesso a dados fica prejudicado em implementações de particionamento de raios. A coerência de objetos ainda pode ser aproveitada.

5.6. Particionamento Estrutural

Denominar-se-á particionamento estrutural àquelas técnicas de "ray tracing" paralelo baseadas na estrutura dos modelos do cenário. Tipicamente, são técnicas que só podem ser empregadas quando os objetos do cenário são representados segundo alguma técnica específica de modelamento.

Identificam-se, basicamente, três técnicas de modelamento (MÄNTYLÄ, 1988) que impõem uma estrutura ao modelo, passíveis de processamento segundo o particionamento estrutural :

- *Geometria de Sólidos Construtivos (CSG)*: esta técnica de modelamento representa um objeto através de um conjunto de operações booleanas regularizadas, aplicadas em sólidos primitivos (tipicamente a esfera, paralelepípedo, cone, cilindro e toro) (figura 2.11).

- *Representação por octárvore (octree)*: o objeto é representado por uma discretização do espaço em forma de octárvore, onde cada célula, ou "*voxel*", é "cheia" ou "vazia" se contém ou não volumes pertencentes ao objeto descrito.
- *Enumeração espacial*: esta técnica de representação é semelhante à representação por octárvore a menos do fato de a discretização ser feita através de um particionamento uniforme do espaço.

Sendo a classificação, aqui introduzida, de escopo original, é importante que fique absolutamente clara a diferença entre algoritmos paralelos de "ray tracing" que podem gerar imagens a partir de cenários modelados segundo as técnicas mencionadas ou mesmo unicamente a partir delas e os algoritmos que aqui se deseja classificar. Existem, por exemplo, implementações que somente sintetizam imagens de cenários modelados por CSG mas adotam *particionamento de imagem* como estratégia de paralelização (HOLLIMAN et alli, 1993). Estas implementações, ao contrário daquelas aqui destacadas, não fazem uso explícito da estrutura de modelamento, como por exemplo o sistema descrito em (KEDEM & ELLIS, 1989) que também apenas admite a metodologia CSG como forma de modelamento do cenário, mas *particiona* os *primitivos* e *operações booleanas* pelos vários processadores do sistema, organizados segundo a árvore CSG que descreve o cenário.

Uma característica dos algoritmos de particionamento estrutural é a possível dependência do seu ganho de velocidade em relação à complexidade do cenário. Assim, tipicamente, as curvas de ganho são lineares se considerar-se um aumento da complexidade ou tamanho do cenário de acordo com o incremento do número de processadores. Esta característica pode limitar a escalabilidade deste tipo de solução.

Outra desvantagem implícita nas técnicas de particionamento estrutural é a restrição quanto às técnicas de modelamento permitidas. Por exemplo, não se pode usar uma solução CSG se o cenário foi modelado por enumeração espacial. Esta particularidade restringe uma das importantes características do algoritmo de "ray tracing", a sua generalidade quanto a modelos de representação.

As propostas nesta categoria tipicamente têm melhor desempenho quando a topologia do multicomputador combina com o tipo de modelamento utilizado. Assim, a metodologia CSG favorecerá topologias em árvore binária enquanto que a enumeração espacial terá melhor desempenho em malhas 3D.

De qualquer forma, em sistemas especializados, estas técnicas encontram boa utilização já que seu desempenho pode compensar suas desvantagens.

Analisa-se nas seções seguintes, as características das subclasses citadas.

5.6.1. Representação CSG

As técnicas de particionamento estrutural para representação CSG em geral dividem os nós de processamento em dois tipos: processadores de intersecção e processadores de combinação. Os primeiros realizam o cálculo de intersecção dos raios com os sólidos primitivos que constituem o cenário, transmitindo aos segundos os pontos de entrada e saída do raio no primitivo. Os processadores de combinação combinam os dados recebidos de dois processadores de intersecção, segundo a operação booleana definida na representação do sólido (figura 2.12).

O resultado da operação booleana realizada é passado a outro processador de combinação que combinará com os dados de outro processador do mesmo tipo. Este encadeamento, em forma de árvore binária, se repete até atingir o nó raiz da árvore CSG, encerrando o processamento do raio inicial. Este resultado pode gerar novos raios (secundários) reiniciando um processo semelhante. A fig. 5.21 ilustra a arquitetura lógica do algoritmo.

FIG. 5.21 - Particionamento estrutural para representação CSG

A alocação de objetos é bastante simples. Os processadores em nós folha da árvore CSG são os processadores de intersecção e cada um deve armazenar os parâmetros correspondentes a um primitivo sólido. Os demais processadores são processadores de combinação e memorizam a operação booleana correspondente à sua posição na árvore. Um processador especial, ou o hospedeiro do sistema, recebe os resultados gerados na raiz da árvore e gera os raios primários e secundários que alimentam o sistema.

A cada ciclo, um mesmo raio é apresentado a todos os processadores de intersecção nos nós folha da árvore. Cada "linha horizontal" de processadores sempre trabalha no mesmo raio. Após ter sido completado o cálculo de intersecção simultâneo realizado pelos processadores de intersecção, o raio seguinte lhes é enviado. Ocorre um processamento concorrente do tipo "duto" ("*pipeline*"), aumentando o paralelismo do algoritmo.

Dois trabalhos propõem implementações de particionamento estrutural para representação CSG: (MARSH, 1988) e (KEDEM & ELLIS, 1989). Ambas as propostas são semelhantes.

A figura 5.22 resume qualitativamente as características de desempenho desta classe de solução.

FIG. 5.22 - Desempenho do particionamento estrutural - CSG

A seguir são analisadas algumas características de desempenho do particionamento estrutural - CSG. Exposição muito mais detalhada pode ser encontrada em (SANTOS, 1994b).

A) Sobrecargas de Processamento

Não existe pré-processamento nesta implementação. Apenas deve-se fazer a alocação da árvore CSG aos nós de processamento.

O maior problema desta proposta em relação à sobrecarga de processamento é o fato de que não é utilizada nenhuma técnica de redução do número de intersecções a calcular, isto é, é feito o "ray tracing" exaustivo, calculando-se a intersecção de todos os raios com todos os objetos (primitivos, no caso) do cenário. Outras propostas, também orientadas ao modelamento CSG porém utilizando outros tipos de particionamento, conseguem muito melhor eficiência neste aspecto (HOLLIMAN et alli, 1993).

Não estão presentes aspectos de fragmentação ou travessia que normalmente implicam em sobrecargas importantes nas outras modalidades de particionamento.

B) Balanceamento

Os sólidos primitivos usados em CSG podem ser representados por superfícies quádricas, que têm a equação geral abaixo:

$$ax^2 + by^2 + cz^2 + dxy + eyz + fxz + gx + hy + iz + k = 0 \quad (5.2)$$

Desta forma, independentemente do primitivo associado a um processador, o tempo para cálculo de intersecção com raio será aproximadamente constante, facilitando o balanceamento. Como é calculado o "ray tracing" exaustivo, a carga relativa de qualquer nó de processamento pode ser determinada a priori dependendo apenas do número de primitivos associado ao nó em caso de processador de intersecção.

Por outro lado, existem alguns fatores que prejudicam o balanceamento:

- Os tempos para cálculo de intersecção e combinação (operação booleana) são bastante diferentes.
- Pode haver não casamento entre o número de nós da árvore CSG e o número de processadores disponíveis no multicomputador, causando ociosidade.
- Se a razão entre o número de árvores de processadores e o número de raios não for inteira, árvores inteiras de processadores podem ficar ociosas próximo ao fim do processamento.
- O processador hospedeiro terá carga diferente dos demais nós do sistema.

C) Comunicação

O padrão de comunicação neste tipo de implementação é constituído de mensagens curtas e em grande quantidade.

O número de mensagens geradas no processamento de uma cena é dado por:

$$N_m = (2 \cdot N_p + N_c) \cdot N_r \quad (5.3)$$

onde: N_m Número de mensagens
 N_p Número de primitivos
 N_c Número de combinações

N_r Número de raios

Esta equação inclui as mensagens geradas pelos processadores de intersecção ($N_p \cdot N_r$) e de combinação ($N_c \cdot N_r$) além daquelas enviadas pelo hospedeiro aos processadores de intersecção ($N_p \cdot N_r$). Estas últimas, sendo as mesmas para todos os processadores de intersecção, podem ser transmitidas por propagação ("*broadcasting*").

Para maximizar o desempenho do sistema, a comunicação deve ser síncrona. O multicomputador funcionaria quase como um sistema sistólico (KUNG, 1982) (SANTOS, 1993).

A conectividade requerida por esta implementação é bastante baixa: apenas 3 elos (com exceção do hospedeiro). Uma topologia em árvore binária é a indicada neste caso.

Dependendo dos circuitos e estratégias de comunicação empregadas no multicomputador em uso, estas características de comunicação podem ser favoráveis ou desfavoráveis ao desempenho global do sistema. Se considerado um sistema de comunicação onde a latência é significativa e existe previsão para roteamento de mensagens então pode-se prever fraco desempenho do sistema. Se este mecanismo for adaptado para comunicação totalmente síncrona, o resultado é o oposto.

D) Uso de Memória

O uso de memória é bastante otimizado nesta implementação já que não existe fragmentação.

Se o número de processadores for maior que o de nós da árvore CSG haverá replicação de dados (várias árvores de processadores, cada uma com dados idênticos). Esta situação deve ser bastante restrita para a média das cenas e dos multicomputadores.

O uso de memória não é uniforme entre os processadores: nós que realizam intersecções deverão armazenar os parâmetros das equações dos sólidos primitivos pelos quais são responsáveis. Nós de combinação podem ter memória restrita a apenas 2 bits por operação booleana que devem realizar. A complexidade dos programas (e portanto o tamanho do código) é também bastante diferente para os dois tipos de processadores.

Para árvores CSG balanceadas, metade mais um dos processadores calculam intersecções enquanto os demais realizam combinações.

E) Ganho de Velocidade e Eficiência

Esta metodologia apresenta a surpreendente capacidade de ganho de velocidade linear unitário e, portanto, eficiência muito próxima de 100% para quase todos os casos (combinações de número de processadores, nós CSG e número de raios) (SANTOS, 1994b).

F) Restrições Algorítmicas

Não permite a adoção de praticamente nenhuma técnica de aceleração e restringe o modelamento ao CSG.

5.6.2. Representação por Octárvore ("Octree")

O único trabalho encontrado apresentando proposta puramente nesta categoria é (MARSH, 1988). (FUJIMOTO & IWATA, 1985) propõe uma técnica que pode ser vista como mista entre particionamento do espaço e particionamento estrutural, dependendo do tipo de modelamento do cenário.

Marsh propõe a alocação de cada nó da octárvore a um elemento de processamento. Processadores virtuais poderiam ser simulados em caso de haver mais nós na árvore do que processadores.

Esta implementação é diferente daquelas do tipo particionamento adaptativo do espaço, em forma de octárvore. No particionamento estrutural, uma mesma região do espaço é alocada a vários processadores pois cada nó da octárvore representa um elemento alocável e não somente os nós-folha, como no caso do particionamento do espaço.

Nesta técnica de particionamento estrutural, o cenário, assim como a divisão do espaço, é modelado através de representação por octárvore, de forma que o particionamento do espaço corresponde ao particionamento do objeto. Não existe a fragmentação do particionamento do espaço nem as intersecções redundantes das hierarquias de envoltórios do particionamento de objetos. Há uma combinação das vantagens das duas técnicas proporcionada pelo tipo de modelamento geométrico utilizado.

A figura 5.23 ilustra a alocação de nós da octárvore aos processadores (MARSH, 1988).

FIG. 5.23 - Alocação de nós da octárvore aos processadores

A figura 5.24 resume as características de desempenho desta implementação

FIG. 5.24 - Desempenho do partic. estrutural - representação por octárvore

A) Sobrecargas de Processamento

Não há pré-processamento, apenas alocação de nós da octárvore aos elementos de processamento do sistema.

A travessia dos raios é bastante eficiente pois requer apenas a determinação da próxima célula penetrada. As fronteiras podem ser feitas ortogonais aos eixos principais.

O cálculo de intersecção não existe pois cada célula-folha é definida como totalmente cheia ou vazia.

B) Balanceamento

Nesta implementação, hierárquica, o processador responsável pelo nó da raiz da octárvore fica submetido à maior carga pois todos os raios têm de ser processados por ele em primeiro lugar. Inevitavelmente ocorre desbalanceamento.

Não se tem nenhum resultado, experimental ou simulado, do comportamento deste tipo de implementação, sendo difícil avaliar o impacto real do desbalanceamento no desempenho desta metodologia.

C) Comunicação

A comunicação é semelhante àquela observada em sistemas com particionamento adaptativo do espaço por octárvore. Na proposta de Marsh um processador ao receber um raio e calcular a iluminação no ponto de intersecção, deve enviar um pacote com a informação de iluminação local de volta ao processador que originou o raio, para fins de composição de cor. Este tipo de implementação aumenta ainda mais o fluxo de mensagens no sistema.

A conectividade requerida é, no máximo de grau 9.

D) Uso de Memória

Não ocorre fragmentação já que existe total coincidência das células do particionamento com as do objeto.

Cada nó de processamento deve armazenar os raios recebidos (temporariamente), os endereços dos processadores responsáveis por suas 8 células filhas (se houverem), informação sobre a ocupação da célula (cheia, vazia ou subdividida) e suas características ópticas. Se um processador é responsável por mais de uma célula (i.e., está simulando também processadores virtuais), então deve conter informações referentes a estas outras células também.

A representação por octárvore sabidamente consome muita memória, principalmente se deseja-se representar o objeto com grande precisão. No entanto, como este aspecto é comum ao algoritmo sequencial também, não se deve considerá-lo como sobrecarga no uso de memória.

E) Ganho de Velocidade e Eficiência

Uma determinação analítica do ganho, conforme desenvolvida para a metodologia de particionamento estrutural de CSG (MARSH, 1988) (SANTOS, 1994b), é difícil de ser feita para o caso da representação por octárvore.

Não se tem resultados experimentais, nem simulados.

F) Restrições Algorítmicas

Esta metodologia está restrita ao modelamento geométrico do tipo representação por octárvore.

5.6.3. Representação por Enumeração Espacial

O modelamento geométrico por enumeração espacial é pouco utilizado pela dificuldade em criar-se diretamente um modelo. Por outro lado, a visualização científica de dados volumétricos é um campo em grande desenvolvimento atualmente, no qual os dados a serem visualizados se apresentam sob uma forma de enumeração espacial.

Assim, volumes de dados obtidos por simulações de fenômenos físicos ou através da aquisição por tomógrafos se constituem em "cenários" cada vez mais comuns.

O "ray tracing", com pequenas modificações, em geral é a técnica escolhida para visualizar-se este tipo de dados porque permite a visualização volumétrica direta, evitando a introdução de "artefatos", falsos contornos que ocorrem com o uso de métodos baseados na detecção de iso-superfícies, por exemplo. Além disso, permite a visualização de partes internas do volume através da transparência, recurso este indispensável em aplicações médicas. No campo de visualização volumétrica, "ray tracing" otimizado é o algoritmo seqüencial mais rápido conhecido (NIEH & LEVOY, 1992).

As análises apresentadas aqui são baseadas em trabalhos para visualização científica porém são aproximadamente válidas também para síntese de imagens de objetos geométricos descritos por enumeração espacial.

O particionamento estrutural para enumeração espacial faz uso do fato de os dados serem homogêneos e terem distribuição espacial regular. Tal como nas outras variantes de particionamento estrutural, também não ocorre nesta o fenômeno da fragmentação de dados.

Montani (MONTANI et alli, 1992) propõe um algoritmo híbrido, baseado em particionamento estrutural para enumeração espacial e particionamento de imagem. O volume de dados é particionado entre os processadores que compõem o multicomputador, segundo uma divisão em fatias.

Os processadores são divididos em aglomerados ("*clusters*"). Cada aglomerado recebe uma cópia completa do volume de dados. Partições da imagem são alocadas à cada um deles e o conjunto de aglomerados trabalha segundo a metodologia de particionamento de imagem. Dentro de cada aglomerado ocorre particionamento estrutural, alocando-se a cada nó uma fatia do volume.

A figura 5.25 resume as características de desempenho desta classe de solução.

FIG. 5.25 - Desempenho do particionamento estrutural - enumeração espacial

A) Sobrecargas de Processamento

Pode-se fazer implementação desta técnica sem pré-processamento. O uso de balanceamento estático requer uma fase de pré-amostragem para determinação da carga relativa de cada região. Os resultados dessa pré-amostragem são aproveitados na imagem final, não caracterizando sobrecarga.

A travessia dos raios pelo volume é feita através de uma varredura incremental bastante semelhante ao 3D-DDA de Fujimoto (FUJIMOTO & IWATA, 1985) e tem a finalidade de identificar as células atravessadas pelos raios. É, portanto, pouco custosa.

Cada processador deve gerar todos os raios primários para verificar quais deles penetram o volume de dados em sua fatia. Esta técnica implica em sobrecarga relativa à geração de raios e aos cálculos de intersecção repetidos por todos os processadores. Um controle centralizado desta tarefa elimina esta redundância porém implica em maior comunicação entre os nós. Além disso, não favorece a prevenção de travamentos perpétuos ("deadlocks") e balanceamento de carga.

B) Balanceamento

O balanceamento é feito através da movimentação das fronteiras que definem cada fatia. Pode ser feito dinamicamente, durante a execução do algoritmo, com a implementação de um mecanismo de monitoração de carga ou estaticamente através de uma pré-amostragem dos tempos para cálculo de alguns pontéis.

Montani et alli (MONTANI et alli, 1992) sugerem uma heurística para determinar os novos tamanhos das fatias atribuídas a cada nó, supondo uma alocação inicial uniforme.

Os processadores calculam em paralelo um conjunto de raios de pré-amostragem. Os novos tamanhos das fatias serão determinados de acordo com a equação abaixo:

$$tamanho[i] = tamanho[i] + l_x \cdot \frac{t_{médio} - t_{total\ i}}{\sum t_{total\ i}} \quad (5.4)$$

onde: $tamanho[i]$ largura da fatia atribuída ao nó de processamento i ;
 l_x largura do volume de dados na direção x ;
 $t_{médio}$ tempo médio de processamento dos processadores;
 $t_{total\ i}$ tempo de processamento total do processador i .

Cada processador transfere ou recebe dados de seus processadores vizinhos. Esta técnica é eficiente porque o tempo de redistribuição é muito menor que o tempo para geração da imagem. Montani obteve redução de 7 a 37% no tempo total de processamento, com seu uso.

C) Comunicação

A comunicação entre processadores é bastante simples devido ao modo como os dados são alocados (em fatias). Cada processador só precisa comunicar-se com outros dois, responsáveis por fatias de dados adjacentes. A conectividade requerida é de 2 elos por nó de processamento.

Neste sistema, as mensagens tendem a ser curtas, em grande quantidade e totalmente locais.

D) Uso de Memória

O algoritmo proposto por Montani é híbrido com particionamentos estrutural e de imagem. Variando-se o número de processadores contidos nos aglomerados pode-se ir de um particionamento puramente estrutural (um único aglomerado contendo todos os processadores do multicomputador) até um particionamento de imagem puro (aglomerados com um único processador cada).

Quando tem-se particionamento puramente estrutural, observam-se as características típicas desta metodologia: ausência de fragmentação ou duplicação de dados. Ao contrário, quando o sistema caminha em direção ao particionamento de imagem, começa a crescer mais e mais a duplicação de dados, característica do particionamento de imagem.

No particionamento estrutural não há fragmentação de dados já que a divisão é feita junto às fronteiras das células. O particionamento deste tipo de dados em forma de fatias é bastante simples e natural.

E) Ganho de Velocidade e Eficiência

A dependência da eficiência em relação à complexidade do modelo, típica do particionamento estrutural, também foi detectada no trabalho de Montani que verificou grande dificuldade para obter ganhos de velocidade aumentando o número de processadores e mantendo a resolução do volume de dados fixo. A razão disto é que ao subdividir-se mais o volume (para alocar-se fatias a novos processadores), o custo de comunicação de cada raio aumenta (tem de atravessar mais fatias) enquanto o custo de processamento, já relativamente baixo, é dividido entre mais processadores.

F) Restrições Algorítmicas

O modelamento é restrito à enumeração espacial (com dados modelados, simulados ou adquiridos).

5.7. Particionamento Funcional

A operação de determinação da cor de um pontel segundo a metodologia de "ray tracing" é composta de várias etapas, que podem diferir de acordo com a variação do algoritmo utilizada. Algumas etapas, no entanto, são fundamentais:

- Cálculo da direção de raios (primários, secundários e de sombra);
- Intersecção de raio com objeto;
- Visibilidade (escolha da intersecção mais próxima da origem do raio);
- Cálculo da normal no ponto de intersecção;
- Cálculo de iluminação local;
- Composição de contribuições de iluminação.

Quando as etapas do "ray tracing" são realizadas por nós diferentes, isto é, quando existe, no sistema, nós que realizam apenas alguma(s) das tarefas que constituem o "ray tracing" tem-se o particionamento funcional.

Raramente o particionamento funcional é encontrado em forma pura devido a problemas de escalabilidade. O algoritmo de "ray tracing" é constituído de poucas etapas e é difícil subdividir-se uma etapa para execução em diversos processadores. Algumas

poucas possibilidades são apresentadas na figura 5.26, que ilustra um exemplo de sistema com particionamento funcional puro.

FIG. 5.26 - Particionamento funcional puro

A operação de visibilidade pode ser paralelizada como um algoritmo de ordenação ou mínimo de conjuntos. A etapa de iluminação pode ser realizada cooperativamente por alguns processadores (cálculo de termos da equação de iluminação) ou vários pontos podem ser iluminados em paralelo por eles.

Normalmente, para aproveitamento de mais processadores de um sistema, o particionamento funcional é combinado com outros, enquadrando a metodologia no particionamento híbrido.

Os particionamentos funcionais normalmente induzem a arquiteturas do tipo duto (*pipeline*). Assim, também esta forma de processamento concorrente pode ser explorada com esta metodologia.

A única proposta de particionamento funcional puro classificada nesta dissertação é (KUNII et alli, 1989) que adota um enfoque diferente do mencionado. Nesta proposta, as operações atômicas que constituem cada etapa do "ray tracing" são executadas por processadores diferentes.

A combinação de outros tipos de particionamento juntamente com o funcional é proposta bastante atrativa, simples e decisiva para superação de algumas desvantagens inerentes desta metodologia.

Resumo das características gerais de desempenho de implementações de particionamento funcional é apresentado no gráfico da figura 5.27.

FIG. 5.27 - Desempenho do particionamento funcional

A) Sobrecargas de Processamento

A impossibilidade de utilização de resultados de outras etapas do processo de "ray tracing" pode levar a algumas sobrecargas de processamento. Estas restrições são bastante dependentes do esquema específico de particionamento funcional implementado, havendo alguns sem restrições que causem sobrecarga.

B) Balanceamento

Este certamente é um dos pontos críticos das implementações de particionamento funcional devido ao fato de que as diversas etapas podem requerer tempos de processamento muito diferentes. Como a maioria dos sistemas é configurado segundo uma arquitetura de duto, todo o processo deve acompanhar o estágio mais lento.

C) Comunicação

Resultados intermediários têm de ser comunicados entre os processadores, aumentando a comunicação. As mensagens em geral são curtas, sofrendo forte influência da latência de comunicação.

Estes fatos tornam o particionamento funcional pouco adequado a sistemas fracamente acoplados.

D) Uso de Memória

Em geral as operações de natureza geométrica (cálculo de intersecção e normal, por exemplo) são alocadas a processadores diferentes daquelas de natureza óptica (cálculo de iluminação). Assim, também os dados geométricos e ópticos podem ser alocados separadamente.

Deve-se mencionar que, num esquema de particionamento puro, raramente se pode fazer as operações de cálculo de intersecção em mais de um nó. Uma exceção seria o caso de um aglomerado ("*cluster*") de nós para processar pedidos de intersecção por demanda. Neste caso, deverá haver replicação total dos dados geométricos ou implementado um esquema de cache de objetos, obtendo-se replicação parcial à custa de aumento de comunicação.

E) Ganho de Velocidade e Eficiência

É difícil aplicar-se os conceitos de ganho de velocidade e eficiência para implementações de particionamento funcional pois os resultados obtidos com a adição de nós ao sistema dependem de qual tarefa é atribuída aos nós adicionados. Assim, por exemplo no sistema da figura 5.26, se mais um processador for adicionado ao aglomerado que calcula visibilidade, o ganho será provavelmente diferente daquele obtido com a inclusão no aglomerado de cálculo de iluminação. Os resultados serão variáveis também de acordo com a proporção de processadores alocados a cada tarefa.

F) Restrições Algorítmicas

Dada a grande variedade de implementações possíveis dentro de um esquema de particionamento funcional, não se pode estabelecer restrições algorítmicas gerais para esta classe.

5.8. Particionamentos Híbridos

Os particionamentos híbridos são aqueles que apresentam alguma combinação dos particionamentos mencionados anteriormente. Normalmente existem dois ou mais níveis organizacionais no algoritmo, com um tipo de particionamento diferente em cada um.

Em geral, estas implementações requerem multicomputadores com arquiteturas especiais.

As implementações híbridas costumam apresentar características combinadas dos particionamentos em que estão baseadas.

A seguir serão exemplificadas algumas implementações de particionamentos híbridos. Não se pode fazer generalizações quanto ao desempenho destas subclasses de particionamento híbrido pois são muito particulares e existem poucas implementações representativas.

5.8.1. Espaço + Objetos

Scherson e Caspary (SCHERSON & CASPARY, 1987) apresentam uma estrutura de dados para aceleração de "ray tracing" que, se adaptada como algoritmo paralelo, resultaria em particionamento híbrido entre particionamento de espaço e de objetos.

Baseado em análises de desempenho de algoritmos de particionamento do espaço (octárvore) e hierarquias de envoltórios, foi sugerido um particionamento adaptativo que, até determinado nível, fosse feito segundo uma octárvore. Os objetos contidos dentro de cada célula da octárvore são então agrupados formando uma hierarquia de envoltórios. Este algoritmo (versão seqüencial) mostrou melhor desempenho, para as cenas testadas, que o particionamento em octárvore puro ou a hierarquia de objetos apenas.

5.8.2. Espaço + Raios

A proposta completa de Kobayashi (KOBAYASHI et alli, 1988) deve ser classificada como particionamento híbrido pois propõe um sistema com arquitetura em dois níveis onde ocorrem particionamentos diferentes: o primeiro nível, composto por um conjunto de aglomerados (*clusters*) de nós, realiza processamento segundo um particionamento do espaço com alocação distribuída. O segundo nível, dentro de cada aglomerado, processa os raios que chegam à subregião do espaço correspondente àquele aglomerado (fig. 5.28).

FIG. 5.28 - Particionamento híbrido: Espaço + Raios

No exemplo da figura 5.28 tem-se 8 aglomerados de 3 processadores cada um, que processam o espaço dividido em 64 sub-regiões. Estas sub-regiões são alocadas de forma distribuída aos 8 aglomerados, cada um recebendo 8 sub-regiões. Dentro de cada aglomerado, os raios que penetram cada uma das 8 sub-regiões são processados pelos 3 processadores do aglomerado segundo alocação dinâmica de raios (por demanda). Cada processador do aglomerado tem a descrição dos objetos contidos nas 8 sub-regiões pelas quais seu aglomerado é responsável.

A vantagem desta implementação é o bom balanceamento estático conseguido no primeiro nível, através da alocação distribuída, sem causar grandes problemas de fragmentação pois o grau de subdivisão é relativamente pequeno. No segundo nível, o balanceamento é dinâmico, obtido pela alocação dinâmica dos raios que penetram a subregião, aos vários nós que compõem um aglomerado.

A replicação de dados também segue um padrão misto entre os características exibidas pelos dois tipos de particionamento: no nível extra-aglomerado não ocorre replicação, a não ser devido à fragmentação, como no particionamento do espaço. Já ao nível intra-aglomerado, a replicação é total, característica do particionamento de raios.

5.8.3. Estrutural + Imagem

O algoritmo proposto por Montani (MONTANI et alli, 1992) é híbrido, com particionamento estrutural de enumeração espacial e particionamento de imagem.

Aqui novamente os processadores formam aglomerados. Estes aglomerados vão processar partições da imagem. Internamente aos aglomerados, ocorre particionamento de um volume de dados. A cada processador do aglomerado corresponderá uma fatia da estrutura de dados (figura 5.29).

Cada aglomerado possui uma cópia completa do volume de dados. Estes dados estão distribuídos entre cada processador do aglomerado.

Em um nível ocorre balanceamento estático pela alocação distribuída das partições de imagem e em outro, balanceamento dinâmico devido à alocação dinâmica das partições da estrutura de dados (fatias do volume).

FIG. 5.29 - Particionamento híbrido: Estrutural + Imagem

5.8.4. Funcional + Imagem

Duas propostas relatam implementações de sistemas híbridos na forma funcional + imagem: (NISHIMURA et alli, 1983) e (GAUDET et alli, 1988).

A proposta de Severin Gaudet particiona a operação de "ray tracing" em três tarefas distintas:

- Cálculo de raios primários e secundários, iluminação e acumulação de contribuições de iluminação (ILUMINAÇÃO);
- Cálculo de intersecção com árvore de envoltórios (ENVOLTÓRIOS);
- Cálculo de intersecção com objetos e respectivo vetor normal (OBJETOS).

A vantagem deste particionamento de tarefas é que cada uma requer um conjunto diferente de dados: parâmetros ópticos, envoltórios e dados de geometria dos objetos, respectivamente.

O sistema paralelo proposto é constituído por aglomerados de 3 processadores que formam um anel em particionamento funcional, conforme mostra a figura 5.30. Estes anéis formam conjuntos que devem processar partições da imagem.

FIG. 5.30 - Particionamento híbrido: Funcional + Imagem

Na proposta original (GAUDET et alli, 1988), os processadores de cada tipo são conectados a um barramento comum para recebimento de dados, caracterizando um sistema multiprocessador, classe que não se analisa neste trabalho.

No sistema denominado LINKS-1 (NISHIMURA et alli, 1983), (DEGUCHI et alli, 1994) o processo de "ray tracing" é dividido nas mesmas etapas descritas para a proposta anterior.

A arquitetura do sistema é composta por 64 microcomputadores individuais, além de outro para controle do sistema, que podem ser agrupados formando dutos (*pipelines*). A conexão entre os nós é feita por memória compartilhada, caracterizando, como no caso anterior, um sistema fortemente acoplado (fig. 5.31).

FIG. 5.31 - O sistema LINKS-1 (Part. Híbrido Funcional + Imagem)

Estes exemplos demonstram a inadequação do particionamento funcional para uso em multicomputadores, mesmo quando em configuração híbrida, devido à grande demanda de comunicação.

5.8.5. Funcional + Espaço

Kobayashi apresenta em seu trabalho de 1987 (KOBAYASHI et alli, 1987) uma arquitetura paralela para "ray tracing" e um algoritmo para o mesmo fim, que podem ser enquadrados na classe de particionamento híbrido.

A arquitetura proposta (fig. 5.32) é especializada e consta de um computador hospedeiro e dois diferentes tipos de processadores; um responsável pelo cálculo de intersecções (PI) e o outro pela determinação da iluminação (sombreamento) (PS).

FIG. 5.32 - Arquitetura do sistema de Kobayashi'87

Os Processadores de Intersecção (PI) são conectados entre si segundo uma topologia hipercubo e implementam particionamento do espaço segundo uma octárvore.

A função dos Processadores de Intersecção é verificar se ocorre intersecção entre os raios que lhes são enviados e os objetos mantidos em suas memórias locais. Em caso de intersecção, calculam a iluminação local e a enviam aos Processadores de Sombreamento, juntamente com informações para reconstrução da árvores de raios associada. Se um raio não intercepta nenhum objeto, uma mensagem é enviada a outro PI para dar continuidade à trajetória do raio. O mesmo ocorre para os raios de sombra e os possíveis raios secundários (reflexão / transmissão).

Os Processadores de Sombreamento (PS) são conectados aos PIs através de uma rede tipo Ômega (FENG, 1981). Estes processadores devem processar pacotes enviados pelos PIs. Estes pacotes contém informações sobre intensidades locais e resultados de raios

de sombra. Com estes dados é possível determinar-se a cor final de cada pixel que é então enviada à memória de quadro.

CAPÍTULO 6

DISCUSSÃO E APLICAÇÕES

6. DISCUSSÃO E APLICAÇÕES

Este capítulo discute o material apresentado nesta dissertação, resume conclusões e mostra como podem ser aplicadas as novas contribuições introduzidas.

6.1. Conclusões sobre Desempenho

As tabelas 6.1 a 6.6 resumem as principais conclusões obtidas revendo a literatura gerada pelas pesquisas realizadas até esta data em soluções paralelas para "ray tracing" em multicomputadores.

PARTICIONAMENTO DE OBJETOS x ESPAÇO		
PART. OBJETOS	PARTICIONAMENTO DE ESPAÇO	
(<i>HIER. DE ENVOLT.</i>)	ADAPTAT. (<i>OCTÁRV.</i>)	UNIFORME (<i>GRADE</i>)
• testa mais envoltórios que a octárvore.	• testa mais objetos que a hierarquia de envoltórios.	• testa mais objetos ou células que a octárvore.
• não apresenta fragmentação.	• causa fragmentação.	• causa muita fragmentação.
• apresenta pouca sobrecarga de memória.	• usa mais memória que a hierarquia de envoltórios.	• usa mais memória que a octárvore.
• cálc. de intersecção com envoltório é mais complexo.	• int. com célula é um pouco mais complexa que na grade.	• cálc. de intersecção com célula é rápido.
• difícil estabelecer boa heurística para agrupamento.	• ruim para objetos pequenos e espalhados no cenário.	• custo total de travessia de raios é maior que na octárvore.

TAB. 6.1 - Comparação entre particionamento de objetos e de espaço

PARTICIONAMENTO DE RAIOS, QUADROS E FUNCIONAL		
RAIOS	QUADROS	FUNCIONAL
• acelera imagens.	• só acelera animações.	• acelera imagens.
• adequado para computadores vetoriais.	• adequado para redes de computadores.	• adequados para implementação em circuitos dedicados.
• balanceamento dinâmico.	• balanceamento dinâmico.	• difícil balancear.
• em geral, replicação total.	• em geral requer replic. total.	• divide os tipos de dados.

TAB. 6.2 - Comparação entre particionamentos de raios, quadros e funcional

PARTICIONAMENTO DE IMAGEM x ESPAÇO	
PARTICIONAMENTO DE IMAGEM	PARTICIONAMENTO DE ESPAÇO
• balanceamento dinâmico automático.	• difícil obter bom balanceamento de carga.
• granularidade das tarefas facilmente ajustável.	• soluções não apresentam boa escalabilidade.
• todas as tarefas a executar estão disponíveis desde o início do processamento.	• tarefas são geradas a medida que os raios percorrem o espaço.
• poucas restrições algorítmicas	• algumas restrições algorítmicas
• simples de implementar.	• exige pré-processamento para subdivisão.
• difícil não replicar dados.	• só replica dados por fragmentação.

TAB. 6.3 - Comparação entre particionamento de imagem e de espaço

REPLICAÇÃO DE DADOS		
REPLICAÇÃO TOTAL	REPLICAÇÃO PARCIAL	SEM REPLICAÇÃO
• implementação mais simples.	• exige gerenciador de cache (part. imagem).	• dependente do particionamento.
• pouca comunicação.	• mensagens longas.	• mensagens curtas.
• usa muita memória.	• uso regular de memória.	• faz bom uso da memória.
• poucas restrições algorítmicas.	• poucas restrições algorítmicas.	• algumas restrições algoritm.

TAB. 6.4 - Comparação entre características da replicação de dados

ALOCÇÃO DE PARTIÇÕES		
EM BLOCO	DISTRIBUÍDA	DINÂMICA
• pouca fragmentação.	• alta fragmentação.	• alta fragmentação.
• pouca comunicação.	• maior comunicação.	• maior comunicação.
• conectividade reduzida.	• favorece topologia toroidal.	• conectividade variável.
• é simples.	• é simples.	• exige monitoramento.
• favorece coerência.	• dificulta uso de coerência.	• dificulta uso de coerência.
• prejudica balanceamento.	• balanceamento estático.	• balanceamento dinâmico.

TAB. 6.5 - Comparação entre características da alocação de partições

SUBDIVISÃO		
UNIFORME	ADAPTATIVA	DINÂMICA
• é simples.	• exige heurística.	• exige heurística dinâmica.
• balanceamento pobre.	• balanceamento estático.	• balanceamento dinâmico.
• pré-processamento simples	• pré-processamento complexo	• exige monitoramento
• pouca mem. para estruturas.	• pode requerer estrut. auxiliar.	• pode requerer estr. auxiliares.
• comunicação local	• comunicação nem sempre local	• muita comunicação
• baixa conectividade	• conectividade pode ser alta	• conectividade variável

TAB. 6.6 - Comparação entre características da subdivisão

6.2. A Melhor Solução Paralela para "Ray Tracing"

O capítulo 5 deste trabalho apresentou informações referentes ao desempenho de várias soluções ao problema de "ray tracing" paralelo em multicomputadores. A seção anterior resumiu conclusões sobre desempenho. Porém, uma importante questão ainda não respondida é "Qual a melhor solução ?". Nesta seção procura-se discutir a resposta a esta pergunta.

O que se pode concluir sobre a resposta a esta questão, após os estudos realizados, é que ela não é simples nem única. A seguir, procura-se expor argumentos que justificam esta conclusão.

O desempenho de uma solução qualquer para "ray tracing" paralelo pode ser caracterizado, em última instância, pelo tempo de processamento total requerido pelo algoritmo e pela somatória da memória em todos os nós, utilizada durante o processamento, num multicomputador.

O tempo total de processamento é função de um conjunto de parâmetros do algoritmo e de características do multicomputador utilizado. Os parâmetros dos algoritmos, usados para as análises do capítulo 5, podem ser resumidos em:

- *sobrecarga de processamento;*
- *balanceamento de carga;*
- *sobrecarga de comunicação;*
- *eficiência e ganho de velocidade;*
- *restrições algorítmicas.*

As principais características do multicomputador, que influenciam o desempenho de uma solução são:

- *velocidade dos processadores (para cálculo inteiro e ponto flutuante);*
- *topologia/conectividade da rede de interconexão;*
- *velocidade dos elos de comunicação e latência;*
- *política de roteamento de mensagens;*
- *quantidade de memória local em cada nó.*

Cada um destes parâmetros sofre influência de outros fatores de forma diferenciada, em função da solução adotada, conforme foi visto no capítulo anterior.

Com frequência, a adoção de um esquema diferente de subdivisão ou alocação de partições melhora alguns fatores ao mesmo tempo em que piora outros. Nestes casos, a escolha deverá levar em consideração o peso relativo destes fatores na composição de determinado parâmetro de desempenho. Por exemplo, a sobrecarga de *pré-processamento* tem menor influência na sobrecarga total de processamento, pois atua uma única vez, enquanto que uma *travessia de raios* ineficiente, sendo executada milhões de vezes, provoca muito maior sobrecarga.

Pode ocorrer também, que a adoção de um certo esquema de particionamento, subdivisão, alocação ou replicação de dados favoreça determinado parâmetro e prejudique outro. Então, características do multicomputador em uso devem ser levadas em conta para que se possa apontar a melhor alternativa. Por exemplo, no particionamento de imagem, a replicação total de dados reduz a comunicação ao menor valor possível. Induz, no entanto, ao péssimo uso de memória. Num multicomputador com elos de comunicação muito rápidos ou nós compostos (i.e., com um processador dedicado à comunicação e outro ao processamento) a adoção de replicação parcial de dados pode trazer grande alívio ao uso de memória sem ocasionar impacto significativo na comunicação.

Outras influências das características do multicomputador serão vistas a seguir.

Algumas soluções de particionamento do espaço com subdivisão uniforme podem fazer uso de algoritmo 3D-DDA para a travessia de raios. Neste tipo de algoritmo, esta etapa é responsável pela maior parte do processamento e é realizada com *aritmética diferencial inteira*, favorecendo processadores com bom desempenho nesta modalidade de cálculo, em detrimento ao desempenho em *ponto flutuante*. Em outras implementações ocorre o oposto.

A *topologia* do multicomputador tem papel destacado quanto à escolha do algoritmo a paralelizar, pois este deve, preferencialmente, combinar com a topologia do

multicomputador disponível. Particionamento do espaço com subdivisão uniforme é adequado para topologias em grade. Adoção de alocação distribuída favorece a topologias toroidais; particionamento estrutural CSG é ideal para multicomputador com nós conectados em árvore binária, e assim por diante.

Multicomputadores com *alta latência de comunicação* prejudicam soluções que enviam muitas mensagens curtas (por exemplo, mensagens de raio em particionamento do espaço). Grande velocidade de comunicação favorece mensagens longas como as usadas para transferência de objetos na subdivisão dinâmica.

A *política de roteamento* de mensagens também tem influência sobre o tamanho e quantidade de mensagens enviadas pela solução adotada, principalmente no que se refere a colisões e tempo de comunicação (ZUFFO, 1993).

A *memória disponível* em cada nó é parâmetro decisivo no desempenho de uma solução. Pouca disponibilidade de memória pode até mesmo impedir o processamento de uma cena através de uma solução que adote replicação total de dados, por exemplo. Por outro lado, conforme relatado no capítulo anterior, soluções de particionamento de imagens com replicação parcial de dados têm desempenho proporcional a este parâmetro, até o limite da replicação total.

Não sendo já suficiente a complexidade de inter-relacionamentos dos parâmetros que influenciam o desempenho das soluções paralelas para "ray tracing" em multicomputadores, ainda outra característica externa, além do multicomputador, atua de forma a que não se possa dar uma resposta única à pergunta formulada no início desta questão: a complexidade da cena a sintetizar.

Vários autores já mencionaram que, dados dois algoritmos distintos, sempre é possível encontrar cenas-teste em que o desempenho de um é superior ao outro e vice-versa. Esta afirmação é verdadeira para algoritmos sequenciais de "ray tracing" e mais ainda em implementações paralelas.

Na figura 5.1 foi exemplificada a influência da distribuição dos objetos no cenário sobre o desempenho do particionamento de espaço. Outros parâmetros que caracterizam a complexidade da cena e que influenciam de forma diversa o desempenho das soluções são:

- *quantidade de objetos no cenário;*
- *posição dos objetos no cenário, entre si e em relação ao observador;*
- *forma e tamanho dos objetos;*
- *características especulares/transmissivas dos materiais dos objetos;*
- *técnica de modelamento geométrico utilizada;*

- *quantidade, tipo e posição das fontes de luz;*
- *resolução da imagem a ser gerada;*
- *porcentagem da imagem mostrando a cor-de-fundo;*
- *número de raios gerados;*
- *profundidade média das árvores de raio;*
- *comprimento médio percorrido pelos raios;*
- *coerência de imagem, raios, objetos, etc.*

De acordo com o tipo de particionamento, subdivisão e alocação escolhidos, outros fatores intervêm:

- *número de objetos por célula no particionamento de espaço;*
- *heurística de agrupamento para formação de hierarquias de envoltórios no particionamento de objetos;*
- *forma de subdivisão da imagem e tamanho dos pacotes de pontéis no particionamento de imagem, etc.*

Dessa forma, não há resposta única e definitiva à questão "Qual o melhor algoritmo de 'ray tracing' ?", nem em implementação sequencial, muito menos na forma paralela. As conclusões a que se pode chegar estão quase sempre restritas à certa classe de equipamento e cena-teste.

Pode-se, no entanto, examinando-se todas as comparações anteriores e o material apresentado no capítulo 5, chegar a uma conclusão genérica importante:

O particionamento de imagens com subdivisão uniforme, alocação dinâmica e replicação parcial de dados é, desconsideradas cenas particulares e multicomputadores específicos, uma das melhores soluções ao problema de "ray tracing" paralelo em multicomputadores.

Justificativa:

- *Sobrecarga de processamento baixa:* o pré-processamento necessário para fazer a alocação inicial de dados é bastante menor que a requerida para fazer subdivisão de objetos ou do espaço.
- *Bom balanceamento de carga:* resultados experimentais mostraram que o particionamento de imagens com alocação dinâmica proporciona os melhores resultados em termos de balanceamento de carga (variação de apenas 1.3%). Este parâmetro é um dos mais importantes para garantia de bom ganho de velocidade no sistema.

- *Baixa comunicação*: se adotada replicação total de dados pode-se obter o melhor desempenho teórico em relação ao perfil de comunicação. A replicação parcial causa razoável aumento de comunicação, variável com a coerência de imagem e tamanho do cache de objetos disponível.
- *Replicação de dados ajustável*: sabe-se que a melhor solução, desconsiderado o uso de memória, é a "processor-farm", com replicação total de dados. A adoção de replicação parcial de dados permite que, dependendo da quantidade de memória disponível (função do cenário e do multicomputador) faça-se automaticamente uso ótimo da memória, desde o extremo da replicação total até o caso de nenhuma replicação de dados. A sobrecarga de comunicação é o parâmetro que se contrapõe, neste caso.
- *Ótimo ganho de velocidade e eficiência*: os resultados experimentais de implementações de particionamento de imagem com replicação total de dados são os melhores (até 95.5% para 80 processadores). A eficiência cai se houver pouca disponibilidade de memória (eficiência = 79% a 50.5% para 0.5% de replicação de dados), porém permanecendo em patamares aceitáveis.
- *Poucas restrições algorítmicas*: ao que se pôde observar, apenas algumas técnicas para aceleração de animações sofrem restrições devido ao particionamento de imagem, restrições estas também presentes em outros tipos de particionamento. Adicionalmente, soluções baseadas em particionamento de imagem são as mais simples de implementar e adaptar a partir de soluções sequenciais.

6.3. Novas Alternativas para "Ray Tracing" Paralelo

A finalidade da taxonomia criada e apresentada neste trabalho é, como já foi dito, além de classificar propostas, auxiliar a sugestão de novas soluções.

A figura 6.1 abaixo mostra a abrangência das propostas de "ray tracing" paralelo publicadas na literatura internacional e classificadas segundo a nova taxonomia. Os círculos brancos indicam classes para as quais nenhuma proposta se encontrou, dentro da pesquisa realizada neste trabalho. Vê-se que pouco mais de 10% das classes foram exploradas, mostrando o enorme potencial ainda existente para pesquisas, num campo que, considerando o grande número de trabalhos já publicados, parecia próximo da exaustão.

Nas subseções a seguir discute-se soluções alternativas que poderiam ser enquadradas nas classes inexploradas. Não se pretende apresentar estas discussões como sendo propostas formais de novas soluções pois não se fez implementações ou simulações

de tais sugestões e também não são detalhadas o suficiente. No entanto, são todas sugestões originais, inspiradas pelas classes vazias da taxonomia. Devem ser encaradas apenas como um passo inicial para verificação da viabilidade de se desenvolver novas soluções de "ray tracing" paralelo, tendo como motivação a nova taxonomia.

FIG. 6.1 - Abrangência das propostas de "ray tracing" paralelo

6.3.1. Particionamento de Espaço

As soluções existentes para particionamento de espaço são as que se apresentam melhor distribuídas na taxonomia, abrangendo os vários tipos de subdivisão e alocação. De qualquer forma, ainda existe lugar para novas propostas *sem replicação de dados* ou com

replicação total. Outras classes vazias incluem *subdivisão adaptativa* ou *dinâmica* combinada com *alocação distribuída* ou *dinâmica*.

6.3.1.1. Sem Replicação ou com Replicação Total de Dados

Uma solução *sem replicação de dados para particionamento do espaço* deve fundamentalmente evitar a fragmentação de objetos. Isto é possível para cenários particulares, porém inviável de forma genérica se adotado qualquer tipo de subdivisão uniforme. Talvez mereça pesquisa a idéia de criar-se uma transformação que, localmente, deformasse as fronteiras ou os objetos do cenário de forma a evitar fragmentação. Provavelmente os raios deveriam sofrer uma transformação também, antes do cálculo de intersecção.

Subdivisões adaptativas ou dinâmicas poderiam evitar a fragmentação, porém, provavelmente, incorrendo em grandes sobrecargas de processamento já que as etapas de pré-processamento e travessia de raios seriam mais complexas devido à generalidade das fronteiras.

Outro enfoque alternativo seria o armazenamento do objeto fragmentado em apenas um nó, enquanto os outros conteriam apenas referências ao mesmo objeto, indicando onde sua descrição pode ser encontrada. Esta solução implica em maior sobrecarga de comunicação, para troca de informações ou objetos.

Soluções de particionamento de espaço com replicação total não nos parece interessante já que uma das poucas vantagens do particionamento de espaço é a baixa replicação de dados. Se fosse implementada, havendo memória suficiente, apresentaria sobrecarga de comunicação bastante baixa. De qualquer forma, uma solução com particionamento de imagem e replicação total de dados seria superior a tais propostas em todos os aspectos.

6.3.1.2. Subdivisão Adaptativa/Dinâmica com Alocação Distribuída/Dinâmica

Parece-nos que uma solução envolvendo *subdivisão e alocação dinâmicas* exigiria uma heurística muito sofisticada e de difícil controle, ocasionando grande sobrecarga de processamento. Alta sobrecarga de comunicação também seria característica desta solução já que pacotes de objetos deveriam ser transferidos de um nó para outro.

Subdivisão adaptativa com alocação distribuída (ou vice-versa) poderia produzir um mecanismo de balanceamento estático interessante. A conectividade requerida, no entanto, seria aumentada.

6.3.2. Particionamento de Objetos

Observando a figura 6.1, identificam-se grandes oportunidades para trabalhos inovadores que apresentem propostas de particionamento de objetos com *alocação distribuída* ou até *dinâmica*. A *subdivisão dinâmica* também não foi explorada.

Propostas que impliquem em replicação total de dados não são interessantes já que a principal característica do particionamento de objetos é a facilidade de evitar a replicação de dados.

6.3.2.1. Técnicas hierárquicas

Na técnica hierárquica (árvore de envoltórios) os objetos agrupados num mesmo envoltório poderiam ser alocados a processadores diferentes favorecendo o balanceamento. Este enfoque (*alocação distribuída*) seria desfavorável no aspecto comunicação pois o número de mensagens aumentaria assim como a conectividade requerida.

Subdivisão dinâmica poderia ser experimentada num sistema para geração de animações onde o agrupamento de objetos para formação de hierarquias fosse variado durante o processamento em busca da melhor organização para favorecer o balanceamento de carga.

A *alocação dinâmica* para particionamento de objetos foi unicamente proposta em (SCHERSON & CASPARY, 1988) e ainda é pouco explorada.

6.3.2.2. Técnicas não hierárquicas

Particionamento de objetos com *subdivisão uniforme* e *alocação dinâmica* pode ser implementado através de técnica não hierárquica na configuração em duto.

Os processadores vizinhos trocam informações sobre carga e transferem objetos entre si. A heurística de balanceamento é simples pois só opera com dois vizinhos, assim como a transferência de objetos. Esta solução parece bastante interessante e merece ser melhor examinada em trabalho futuro.

6.3.3. Particionamento de Imagem

O particionamento de imagem já foi bastante explorado em todas as suas possibilidades, considerando a *subdivisão uniforme*, conforme fica claro na figura 6.1. Por outro lado, não se identificou um único trabalho sugerindo *subdivisão adaptativa* ou *dinâmica* para particionamento de imagem, em ambiente paralelo.

Uma *subdivisão adaptativa* poderia, em fase de pré-processamento, lançar raios exploratórios em diferentes partes da imagem para levantar estatísticas sobre a complexidade de cada região, adaptando-se o tamanho de cada partição a estes dados. Note-se que este pré-processamento não implica em sobrecarga pois os raios lançados são aproveitados na imagem final.

Subdivisão dinâmica poderia ser implementada observando-se o tempo de retorno de pacotes de pontéis processados. Este tempo representaria uma estimativa da complexidade da tarefa e poderia ser usado para calibrar-se o tamanho das novas partições. Este esquema seria especialmente interessante em ambientes heterogêneos onde pacotes mais complexos poderiam ser atribuídos a processadores mais poderosos, balanceando adequadamente a carga no sistema. Em outra implementação desta classe, após uma alocação inicial, os processadores pouco carregados requisitariam carga aos outros. Estes, subdividiriam (dinamicamente) seus pacotes de pontéis, cedendo parte ao nó requisitante.

6.3.4. Particionamentos Híbridos

Os particionamentos híbridos são especialmente interessantes na medida em que podem combinar características de dois ou mais esquemas de particionamento, às vezes exibindo vantagens de ambos.

As soluções examinadas durante este trabalho apresentaram as seguintes combinações:

- ESPAÇO + RAIOS;
- IMAGEM + ESTRUTURAL;
- FUNCIONAL + IMAGEM;
- ESPAÇO + FUNCIONAL;
- FUNCIONAL + RAIOS.

(SCHERSON & CASPARY, 1987) propõem um esquema seqüencial onde, num nível existe particionamento de espaço e em outro, dentro de cada célula, particionamento de objetos segundo uma hierarquia de envoltórios. Conforme se mencionou no capítulo 5, uma implementação paralela de tal proposta não nos parece complicada e representaria uma solução com particionamento híbrido ESPAÇO + OBJETOS. O desempenho da solução seqüencial revelou-se melhor que os esquemas puros e, talvez, o mesmo ocorra em implementação paralela.

6.3.5. Particionamento de Quadros

O particionamento de quadros é naturalmente restrito, porém pode-se colocar algumas sugestões.

Uma implementação de *subdivisão adaptativa* poderia ser realizada fazendo-se uma subdivisão dos quadros de uma animação de forma que cada nó se encarregue, aproximadamente, de uma cena (no sentido cinematográfico) da animação. Isto permitiria aproveitamento da maioria dos dados geométricos já carregados em memória e também eliminaria restrições algorítmicas relacionadas à coerência temporal.

CAPÍTULO 7

CONCLUSÕES E TRABALHOS FUTUROS

7. CONCLUSÕES E TRABALHOS FUTUROS

Apresenta-se neste capítulo as conclusões finais e sugestões para futuros desenvolvimentos.

7.1 Conclusões Finais

Neste trabalho constatamos a grande diversidade de soluções propostas para o problema de "ray tracing" em multicomputadores. Pudemos verificar como decisões de projeto dos algoritmos influenciam em parâmetros chave do desempenho de sistemas paralelos de memória distribuída. Estas constatações foram possíveis devido a adoção de uma abordagem simples, porém sistemática, para relatar as características das inúmeras soluções dispersas na literatura.

Propusemos uma taxonomia com dupla finalidade: primeiro, classificar soluções, evidenciando pontos em comum. Segundo, expor oportunidades para a criação de novas soluções de "ray tracing" paralelo.

O primeiro objetivo foi preenchido ao criar-se a taxonomia pela seleção de alguns poucos parâmetros, e valores que podem assumir, dentre os inúmeros possíveis para cada classe de solução, que pudessem diferenciar propostas fundamentalmente diferentes e ainda assim se constituir numa base comum para todas elas. Preenchemos a segunda finalidade escolhendo parâmetros relacionados à estrutura dos algoritmos e não às suas características conseqüentes como balanceamento, por exemplo. As técnicas de subdivisão e alocação de partições determinam as características do balanceamento e não o contrário. Desta forma, o simples exame das combinações destes parâmetros é poderosa semente para a sugestão de novas soluções, antes ocultas sob a multiplicidade de parâmetros e alternativas.

O gráfico Eficiência x Redundância Incrementais, na forma apresentada, se constitui em ferramenta interessante para rápida visualização, comparativa, do mérito de soluções ao problema de "ray tracing" paralelo. Apesar de seu uso requerer dados só passíveis de obtenção através de ensaio ou simulação, permite comparar soluções entre si e em relação às Soluções de Referência, também introduzidas neste trabalho. As Soluções de Referência têm posicionamento destacado neste gráfico, delimitando regiões de características semelhantes. A escolha dos custo de animação, processador e memória permite calibrá-lo para configurações específicas de cada equipamento e aplicação, privilegiando poder de processamento ou memória, conforme o caso.

Durante a fase inicial do desenvolvimento deste trabalho implementamos uma versão seqüencial do algoritmo de "ray tracing". Esta atividade foi de fundamental

importância para a total compreensão do algoritmo, como programa de computador que é, e não somente como entidade abstrata. Conhecer detalhes de implementação propiciou-nos uma abordagem prática, de Engenharia, ao problema que decidimos estudar.

Também uma implementação paralela (particionamento de imagem, subdivisão uniforme, alocação dinâmica e replicação parcial de dados) foi por nós desenvolvida, a partir de programa de domínio público. Apesar de não totalmente depurada, esta nova implementação nos proporcionou o conhecimento de novos problemas, não encontrados em programas seqüenciais. Balanceamento, comunicação, sincronização e alocação de dados (e desenvolvimento de gerenciador de cache, no caso particular implementado) são todos componentes somente presentes nas implementações paralelas do algoritmo. Ambientes de programação, avaliação ("profiling") e depuração bastante diversos do cotidiano "seqüencial" também puderam ser experimentados.

Apesar de não se constituírem como contribuições científicas e nem terem sido referenciados no corpo desta dissertação, relatamos estes pontos como contribuições importantes para o desenvolvimento técnico do autor, resultantes do desenvolvimento deste trabalho.

Iniciamos o desenvolvimento desta pesquisa à procura do "*melhor* algoritmo paralelo de 'ray tracing' ". Concluímos que ele não existe. O que existe são algoritmos com melhor desempenho para um certo conjunto de cenas. Mesmo que deste trabalho resultasse a *total* compreensão de *todos* os fatores envolvidos neste problema e seus inter-relacionamentos, ainda assim não se poderia mudar o fato acima.

Acreditamos, porém, que a sistematização e as ferramentas introduzidas nesta dissertação permitirão determinar quais são possivelmente *as melhores* soluções e que também propiciarão a criação de soluções melhores, isto é, soluções que gerem imagens com mais eficiência e/ou soluções que apresentem o melhor desempenho em um conjunto maior de cenas. Os trabalhos futuros, sugeridos a seguir perseguem esta meta.

7.2 Trabalhos Futuros

Esta dissertação de mestrado pode se constituir no preâmbulo, nos estudos preliminares, para um trabalho que vise responder a pergunta "*Qual o melhor algoritmo de 'ray tracing' para processar uma determinada cena em um dado multicomputador ?*". Uma meta ambiciosa seria a obtenção de uma equação que modelasse o desempenho de certa solução, parametrizada em função de medidas de complexidade da cena e parâmetros do multicomputador.

Sugerimos o emprego de redes neurais para chegar-se, ainda que indiretamente, à resposta desta questão. Existem trabalhos indicando a adequação do emprego de redes neurais para modelamento de fenômenos complexos, especialmente de natureza não linear. Diversas redes poderiam ser treinadas, cada uma especializada em certo tipo de algoritmo. O conjunto de redes seria aplicado sempre ao mesmo multicomputador. Alimentadas com os parâmetros de complexidade de uma cena, as redes neurais treinadas emitiriam um valor de alguma forma proporcional à uma estimativa do tempo de processamento da cena em cada uma das soluções treinadas.

Para que este trabalho possa ser desenvolvido é necessário, em primeiro lugar, um estudo aprofundado de parâmetros de complexidade de cenas para que se possa determinar quais são representativos. Esta matéria ainda é muito pouco explorada, sendo tratada por um número muito restrito de trabalhos. Estes parâmetros constituirão os dados de entrada das redes neurais.

Para a fase de treinamento será necessário que se implemente os algoritmos selecionados e que se façam ensaios para a geração dos vetores de saída das redes. Estes vetores conteriam alguma indicação do tempo de processamento requerido pelos algoritmos.

Os resultados obtidos nesta dissertação podem orientar a seleção de algoritmos com características bastante diversas, favorecendo os diferentes parâmetros e assim aumentando a chance de determinação do melhor algoritmo para determinado caso.

Outro trabalho futuro promissor é a implementação e comparação, em bases as mais homogêneas possíveis, de diversas propostas de "ray tracing" paralelo. É patente a pouca utilidade dos resultados numéricos de desempenho de soluções paralelas de "ray tracing" apresentados pela literatura. Isso ocorre porque inúmeros parâmetros, além da estratégia de paralelização em sí, são alterados em cada relato, influenciando os resultados apresentados:

- modelo e configuração do computador utilizado nos ensaios;
- cenas-teste ensaiadas;
- rotinas de intersecção, iluminação, etc. utilizadas;
- parâmetros de teste incluindo resolução da imagem, superamostragem, profundidade da árvore de raios, etc.;
- qualidade do código implementado (habilidade do programador).

O resultado final disto é que torna-se praticamente impossível comparar-se, diretamente, os resultados de duas propostas quaisquer pois todas as variações mencionadas afetam sobremaneira os tempos de processamento apresentados.

Sugere-se então, como trabalho futuro: partir-se de uma implementação de "ray tracing" conhecida e de domínio público internacional e modificá-la nos pontos essenciais para se implementar cada uma de um conjunto de propostas representativas previamente selecionadas. Dispondo-se deste conjunto de implementações, submetê-las a ensaios usando as mesmas cenas-teste, também consagradas para este uso e de domínio público internacional, no mesmo equipamento paralelo, e medir-se o desempenho de cada uma segundo os mesmos critérios, para várias configurações dos parâmetros.

Apesar da limitação dos resultados a serem obtidos, já que se referem a um multicomputador em particular, sabe-se do valor que tais dados teriam pois representariam, se não para toda uma classe de máquinas, pelo menos a situação real, para um determinado equipamento, do desempenho de várias propostas importantes de implementação paralela do algoritmo de "ray tracing". A comunidade de computação paralela reconhece o valor deste tipo de informação (CYBENKO & KUCK, 1992) pois pode proporcionar aumento no uso mais generalizado da computação paralela.

Estes resultados seriam importantes também para pesquisas em processamento paralelo em máquinas fracamente acopladas. O estudo de hipóteses de modelamento do comportamento destas arquiteturas pressupõem a comparação com resultados experimentais nem sempre fáceis de obter (ZUFFO, 1993). Como já foi mencionado anteriormente, o fato do "ray tracing" ser uma aplicação de granularidade controlável torna o conjunto de dados a ser produzido ainda mais atraente para este fim.

Ainda outros trabalhos que poderiam se seguir a esta dissertação são:

- refinar, implementar e testar as sugestões apresentadas no capítulo 6;
- criar novas formas de paralelização inspiradas pelos resultados aqui apresentados;
- realizar estudo aprofundado das restrições algorítmicas impostas pelas várias classes de "ray tracing" paralelo.

REFERÊNCIAS BIBLIOGRÁFICAS

- AKIMOTO, T. A. ; MASE, K. ; SUENAGA, Y. Pixel-Selected Ray Tracing. **IEEE Computer Graphics and Applications**, v.11, n. 4, p.14-22, 1991.
- AMANATIDES, J. ; WOO, A. A Fast Voxel Traversal Algorithm for Ray Tracing. In: Maréchal, G. (ed.) **EUROGRAPHICS'87 - Proceedings**, p.3-10, 1987.
- AMANATIDES, J. Ray Tracing With Cones. **Computer Graphics**, v.18, n.3, p.129-135, 1984.
- AMANATIDES, J. Realism in Computer Graphics: A Survey. **IEEE Computer Graphics and Applications**, v. 7, n.1, p.44-56, 1987.
- AMDAHL, G. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In: **AFIPS Conference Proceedings**, 30 (April), pp.483-485. Thompson Books, Washington, D.C., 1967.
- APPEL, A. Some Techniques for Shading Machine Renderings of Solids. in: **Spring Joint Computer Conference Proceedings**, AFIPS, p. 37-45, 1968.
- ARNALDI, B. ; PRIOL, T. ; BOUATOUCH, K. A New Space Subdivision Method For Ray Tracing CSG Modelled Scenes. **The Visual Computer**, v.3, n.2, p.98-108, 1987.
- ARVO, J. ; KIRK, D. Fast Ray Tracing by Ray Classification. **Computer Graphics**, v.21, n.4, p.55-64, 1987.
- ARVO, J. ; KIRK, D. A Survey of Ray Tracing Acceleration Techniques. In: **An Introduction to Ray Tracing**. ed. Andrew Glassner. Academic Press, London, 1989.
- ARVO, J., ed. **Graphic Gems II**. Boston, Academic Press, 1991.
- ATHAS, W. C. ; SEITZ, C. L. Multicomputers: Message-Passing Concurrent Computers. **IEEE Computer**, v.21, n.8, p.9-24, 1988.
- ATHERTON, P. ; WEILER, K. ; GREENBERG, D. P. Polygon Shadow Generation. **Computer Graphics**, v. 12, n.3, p.275-281, 1978.
- BADOUEL, D. ; BOUATOUCH, K. ; PRIOL, T. Ray Tracing on Distributed Memory Parallel Computers: strategies for distribtbing computations and data. In: **SIGGRAPH'90 Course Notes #28 - Parallel Algorithms and Architectures for 3D Image Generation**. p.185-198, 1990. Também como **Technical Report 508 - IRISA**, 1990.
- BADT, S. Two Algorithms for Taking Advantage of Temporal Coherence in Ray Tracing. **The Visual Computer**, v.4, p.123-132, 1988.

- BARTON, E. Data Concurrency on the Meiko Computing Surface. **Proceedings of the Conference on Parallel Processing for Computer Vision and Display**, p.402-407, Univ. of Leeds, 1988.
- BELL, G. Ultra Computers - A Teraflop Before It's Time. **Comm. of the ACM**, v.35, n.8, p.27-47, 1992.
- BERGER, M. ; TROUT, T. ; LEVIT, N. Ray Tracing Mirages. **IEEE Computer Graphics and Applications**, v.10, n.3, p.36-41, 1990.
- BLAKE, E. H. The Natural Flow of Perspective: Reformulating Perspective Projection for Computer Animation. **Leonardo**, v.23, n.4. p.401-409, 1990.
- BLINN, J. F. ; NEWELL, M. E. Texture and Reflection in Computer Generated Images. **Comm. of the ACM**, v.19, n.10, p.542-547, 1976.
- BLINN, J. F. Me And My (Fake) Shadow. **IEEE Computer Graphics and Applications**, v.8, n.1, p.82-86, 1988.
- BLOOMER, J. **Power Programming with RPC**. Sebastopol, O'Reilly & Associates, Inc., 1992.
- BOOTHE, B. **Multiprocessor Strategies for Ray-Tracing**. 60p. University of California at Berkeley, Report No. UCB/CSD 89/534, September 1989.
- BOUATOUCH, K. ; MADANI, M. O. ; PRIOL, T. ; ARNALDI, B. A New Algorithm of Space Tracing Using a CSG Model. In: Maréchal, G. (ed.) **EUROGRAPHICS'87 - Proceedings**, p.65-78, 1987.
- BOUATOUCH, K. ; PRIOL, T. Parallel Space Tracing: An Experience on an iPSC Hypercube. in: Magnenat-Thalmann, N. ; Thalmann D. (ed.). **New Trends in Computer Graphics**, Springer-Verlag, p.170-187, 1988.
- BOUKNIGHT, W. J. A Procedure for Generation of Three-dimensional Half-toned Computer Graphics Representations. **Comm. of the ACM**, v.13, n.9, 1970.
- BOUKNIGHT, W. J. ; KELLEY, K. An Algorithm for Producing Half-Tone Computer Graphics Presentations With Shadows and Movable Light Sources. **Spring joint Computer Conference, AFIPS Proceedings**, v.36, p.1-10, 1970.
- BRESENHAM, J. E. Incremental Line Compaction. **Computer Journal**, p.116-120, v.25, n.1, 1982
- BRONSVOORT, W. F. ; VAN WIJK, J. J. ; JANSEN, F. W. Two Methods for Improving the Efficiency of Ray Casting in Solid Modeling. **Computer Aided Design**, v.16, p.51-55, 1984.
- BURGER, P. ; GILLIES, D. Rapid Raytracing of General Surfaces of Revolution. **New Advances in Computer Graphics - Proc. of Computer Graphics International'89**. R. A. Earnshaw & B. Wyvill (ed.) Springer-Verlag, p.523-32, 1989.

- BURKE, A. ; LELER, W. Paralellism and Graphics: An Introduction and Annotated Bibliography. In: **SIGGRAPH'90 Course Notes #28 - Parallel Algorithms and Architectures for 3D Image Generation**. p.111-140, 1990.
- CASPARY, E. ; SCHERSON, I. D. A Self-Balanced Parallel Ray Tracing Algorithm. in: Dew, P. M. et al. (ed.) **Parallel Processing for Computer Vision and Display**. p. 408-419, Addison-Wesley, 1989.
- 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.
- CHAPMAN, J. ; CAVERT, T. W. ; DILL, J. Exploiting Temporal Coherence in Ray Tracing. **Proc. of Graphics Interface'90**, p.196-204, 1990.
- CLEARY, J. G. ; WYVILL, G. Analysis of an Algorithm for Fast Ray Tracing Using Uniform Space Subdivision. **The Visual Computer**, v.4, n.2, p.65-83, 1988.
- CLEARY, J. G. ; WYVILL, B. M. ; BIRTWISTLE, G. M. ; VATTI, R. Multiprocessor Ray Tracing. Technical Report n. 83/128/17, Dept. of Computer Science, Univ. of Calgary, 1983. Também em **Computer Graphics Forum**, v.5, n.1, p.3-12, 1986.
- COHEN, M. F. ; CHEN, S. E. ; WALLACE, J. ; GREENBERG, D. P. A Progressive Refinement Approach to Fast Radiosity Image Generation. **Computer Graphics**, v.22, n.4, p.75-84, 1988.
- COOK, R. L.; PORTER, T. ; CARPENTER, L. Distributed Ray Tracing. **Computer Graphics**, v.18, n.3, p.137-145, 1984.
- COOK, R. L. Stochastic Sampling in Computer Graphics. **ACM Trans. on Graphics**, v.5, n.1, 1986.
- CROW, F. C. Shadow Algorithms for Computer Graphics. **Computer Graphics**, v.13, n.2, p.242-248, 1977.
- CROW, F. C. The Origins of the Teapot. **IEEE Computer Graphics and Applications**, v.7, n.1, 1987.
- CROW, F. C. ; DEMOS, G. ; HARDY, J. ; McLAUGHLIN, J. ; SIMS, K. 3D Image Synthesis on the Connection Machine. in: Dew, P. M. et al. (ed.) **Parallel Processing for Computer Vision and Display**. p. 254-269, Addison-Wesley, 1989.
- CYBENKO, G. ; KUCK, D. J. Revolution or evolution? **IEEE Spectrum**, v.29, n.9, p.39-41, 1992.
- DEGUCHI, H. ; NISHIMURA, H. ; YOSHIMURA, H. ; KAWATA, T. ; SHIRAKAWA, I. ; OMURA, K. A Parallel Processing Scheme for Three-Dimensional Image Creation. **Proc. of IEEE International Symposium on Circuits and Systems**, v.3, p.1285-1288, 1984.

- DIAS, M. L. Ray Tracing Interference Color. **IEEE Computer Graphics and Applications**, v.11, n.2, p.54-60, 1991.
- DIPPÉ, M. ; SWENSEN, J. An Adaptative Algorithm and Parallel Architecture for Realistic Image Synthesis. **Computer Graphics**, v.18, n.3, 149-158, 1984.
- DYKSTRA, P. C. RTPHYS - A Physical Properties Ray Tracer. In: **The Ballistic Research Laboratory CAD Package - Release 4.0 - Vol. III - The BRL-CAD Applications**. Advanced Computing Systems - The U.S. Army Ballistic Research Laboratory (BRL-CAD Symposium'89), Maryland, 1991.
- FENG, T. F. A Survey of Interconnection Networks. **IEEE Computer**, v. 14, n. 12, p.12-27, 1981.
- FLYNN, M. J. Some Computer Organizations and Their Effectiveness **IEEE Trans. on Computers**, v. C-21, n.9, 1972.
- FOLEY, J. D. ; DAM, A. V. ; FEINER, S. K. ; HUGHES, J. F. **Computer Graphics - Principles and Practice**. 2nd. Ed. Addison-Wesley Publishing Company, 1990.
- FRANKLIN, W. R. ; AKMAN, V. Octree Data Structures and Creation by Stacking. In: **Computer-Generated Images - The State of the Art (Proc. of Graphics Interface'85)** ed. N. Magnenat-Thalmann e D. Thalmann, Springer-Verlag, Tokyo, p.176-185, 1985.
- FRENKEL, K. A. Physically Based Modeling versus "Faking It". **Communications of the ACM**, v.31, n.2, p.116-117, 1988.
- FUJIMOTO, A. ; IWATA, K. Accelerated Ray Tracing. In: Kunii, T. L. (ed.) **Computer Graphics: Visual Technology and Art**. p.41-65, 1985.
- GAUDET, S. ; HOBSON, R. ; CHILKA, P. ; CALVERT, T. Multiprocessor Experiments for High-Speed Ray Tracing. **ACM Transactions on Graphics**, v.7, n.3, p. 151-179, 1988.
- GERVAUTZ, M. Three Improvements of the Ray Tracing Algorithm for CSG Trees. **Computer Graphics**, v.20, n.4, p.333-339, 1986.
- GLASSNER, A. S. Space Subdivision for Fast Ray Tracing. **IEEE Computer Graphics and Applications**, v. 4, n. 10, p.15-21, 1984.
- GLASSNER, A. S. Spacetime ray tracing for animation, **IEEE Computer Graphics and Applications**. v.8, n.2, p.60-70, 1988.
- GLASSNER, A. S., ed. **An Introduction to Ray Tracing**. Londres, Academic Press, 1989a.
- GLASSNER, A. S. An Overview of Ray Tracing. In: **An Introduction to Ray Tracing**. ed. Andrew Glassner. London, Academic Press, 1989b.

- GLASSNER, A. S. A Ray Tracing Glossary. In: **An Introduction to Ray Tracing**. ed. Andrew Glassner. London, Academic Press, 1989c.
- GLASSNER, A. S., ed. **Graphic Gems**. Boston, Academic Press, 1990.
- GOLDSMITH, J. ; SALMON, J. Automatic Creation of Object Hierarchies for Ray Tracing. **IEEE Computer Graphics and Applications**, v. 7, n. 5 , p.14-20, 1987.
- GOMES, J. M. ; VELHO, L. C. **Conceitos Básicos de Computação Gráfica**. São Paulo, VII Escola de Computação, IME-USP, 1990.
- GORAL, C. ; TORRANCE, K. E. ; GREENBERG, D. P. ; BATTAILE, B. Modeling the Interaction of Light Between Diffuse Surfaces. **Computer Graphics**, v.18, n.3, p.212-222, 1983.
- GOURAUD, H. Continuous Shading of Curved Surfaces. **IEEE Trans. on Computers**, v.20, n.6, p. 623-628, 1971.
- GREEN, S. A. , PADDON, D. J. ; LEWIS, E. A Parallel Algorithm and Tree-Based Computer Architecture for Ray Traced Computer Graphics. **Proceedings of the Conference on Parallel Processing for Computer Vision and Display**. Univ. of Leeds, 1988.
- GREEN, S. A. ; PADDON, D. J. Exploiting Coherence for Multiprocessor Ray Tracing. **IEEE Computer Graphics and Applications**, v. 9, n. 6, p. 12-26, 1989.
- GREEN, S.A. **Parallel Processing for Computer Graphics**. Cambridge, MIT Press, 1991.
- GREENBERG, D. P. Coons Award Lecture. **Communications of the ACM**, v.31, n. 2, p. 123-129,151, 1988.
- GRÖLLER, E. ; PURGATHOFER, W. Using Temporal and Spatial Coherence for Accelerating the Calculation of Animation Sequences. **Proc. of Eurographics'91**, Vienna, 1991.
- 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.
- HAINES, E. Essencial Ray Tracing Algorithms. In: **An Introduction to Ray Tracing**. ed. Andrew Glassner. Academic Press, London, 1989.
- HALL, R. A Characterization of Illumination Models and Shading Techniques. **The Visual Computer**, v.2, p.268-277, 1986.
- 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.

- HANRAHAN, P. Ray Tracing Algebraic Surfaces. **Computer Graphics**, v.17, n.3, p.83-89, 1983.
- HANRAHAN, P. Using Caching and Breadth-First Search to Speed-up Ray Tracing. **Proc. of Graphics Interface'86**. Vancouver, B. C., 56-61, 1986.
- HANRAHAN, P. A Survey of Ray-Surface Intersection Algorithms. In: **An Introduction to Ray Tracing**. ed. Andrew Glassner. Academic Press, London, 1989.
- HECKBERT, P. ; HANRAHAN, P. Beam Tracing Polygonal Objects. **Computer Graphics**, v.18, n.3, p.119-145, 1984.
- HELMBOLD D. P. ; McDOWELL, C. E. Modeling Speedup (n) Greater than n. **IEEE Trans. on Parallel and Distributed Systems**, v. 1, n.2, p.250-256, 1990.
- HOLLIMAN, N. S. ; MORRIS, D. T. ; DEW, P. M. ; PENNINGTON, A. An Evaluation of the Processor Farm Model for Visualizing Constructive Solid Geometry. in: Dew, P. M. et al. (ed.) **Parallel Processing for Computer Vision and Display**,. p. 443-451, Addison-Wesley, 1989.
- HOLLIMAN, N. S. ; WANG, C. M. ; DEW, P. M. Mistral-3: Parallel Solid Modeling. **The Visual Computer**, v.9, n.7, p.356-370, 1993.
- HSIUNG, P.-K. ; THIBADEAU, R. Accelerating ARTS. **The Visual Computer**, v.8, n.8, p.181-190, 1992.
- INAKAGE, M. Caustics and Specular Reflection Models for Spherical Objects and Lenses. **The Visual Computer**, v.2, n.6, p.379-383, 1986.
- JANSEN, F. W. Data Structures for Ray Tracing. In: **Data Structures for Raster Graphics, Proceedings Workshop**, (eds. L.R.A. Kessener, F. J. Peters, M. L. P. Lierop), Eurographics Seminars, Springer-Verlag, p.57-73, 1986.
- JEVANS, D. A. J. Optimistic Multi-Processor Ray Tracing. in: Earnshaw, R. A. ; Wyvill, B. **New Advances in Computer Graphics**, p.507-522. Springer-Verlag, 1989.
- JOY, K. I. ; BHETANABHOTLA, M. N. Ray Tracing Parametric Surface Patches Utilizing Numerical Techniques and Ray Coherence. **Computer Graphics**, v.20, n.4, p.279-85, 1986.
- KAJIYA, J. T. Ray Tracing Parametric Patches. **Computer Graphics**, v.16, n.3, p.245-254, 1982.
- KAJIYA, J. T. New Techniques for Ray Tracing Procedurally Defined Objects. **ACM Transactions on Graphics**, v.2, n.3, p.161-81, 1983. Também em **Computer Graphics**, v.17, n.3, p.91-102, 1983.
- KAJIYA, J. ; von HERZEN, B. Ray Tracing Volume Densities. **Computer Graphics**, v.20, n.3, p.165-174, 1984.

- KAJIYA, J. ; KAY, T. Rendering Fur with Three-Dimensional Textures. **Computer Graphics**, v.23, n.3, p.271-280, 1989
- KAPLAN, M. ; GREENBERG, D. P. Parallel Processing Techniques for Hidden Surface Removal. **Computer Graphics**, v.13, n.3, p.300-307, 1979.
- KAY, T. L. ; GREENBERG, D. P. Transparency for Computer Synthesized Images. **Computer Graphics**, v. 13, n.2, p.158-164, 1979.
- KAY, T. L. ; KAJIYA, J. T. Ray tracing Complex Scenes. **Computer graphics**, v.20, n.4, p.269-278, 1986.
- KEDEM, G. ; ELLIS, J. L. The Ray-Casting Machine. **Proc. of the IEEE Conference on Computer Design'84**, p.533-538, 1984. Também em: Dew, P. M. et al. (ed.) **Parallel Processing for Computer Vision and Display**,. p. 378-401, Addison-Wesley, 1989.
- KIELY, T. Os Construtores de Idéias. Trad. de Maria R. V. do Conde. **Revista Politécnica**. n.211, out./dez., p.66-75, São Paulo, Escola Politécnica da USP, 1993.
- KIRK, D. B. The Simulation of Natural Features Using Cone Tracing. **The Visual Computer**, v.3, n.1, p.63-71, 1987.
- KOBAYASHI, H. ; NAKAMURA, T. ; SHIGEI, Y. Parallel Processing of an Object Space for Image Synthesis using Ray Tracing. **The Visual Computer**, v.3, p.13-22, 1987.
- KOBAYASHI, H. ; HORIGUCHI, S. ; KUBOTA, H. ; NAKAMURA, T. Effective Parallel Processing for Synthesizing Continuous Images. in: R. A. Earnshaw e B. Wyvill (ed.), **New Advances in Computer Graphics**, p. 343-352, Springer-Verlag, 1989.
- KOBAYASHI, H. ; NISHIMURA, S. ; KUBOTA, H. ; NAKAMURA, T. ; SHIGEI, Y. Load Balancing Strategies for a Parallel Ray-Tracing System Based on Constant Subdivision. **The Visual Computer**, v.4, p.197-209, 1988.
- 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.
- KUNII, T. L. ; NISHIMURA, S. ; NOMA, T. The Design of a Parallel Processing System for Computer Graphics. in: Dew, P. M. et al. (ed.) **Parallel Processing for Computer Vision and Display**,. p. 353-377, Addison-Wesley, 1989.
- KUNG, H. T. Why Systolic Architectures ? **IEEE Computer**, v.15, n. 1, p. 37-46, 1982.

- KUNG, H. T. ; STEENKISTE, P. ; GUBITOSO, M. D. ; KHAIRA, M. Parallelizing a New Class of Large Applications over High Speed Networks. **Proceedings of the Third ACM SIGPLAN Symposium on Principles & Practice of Parallel Programming - PPOPP'91**, Williamsburg, Virginia, p.167-177, April 1991. Também publicado como RT-MAC-9108, IME-USP, Junho 1991.
- LANE, J. M. ; CARPENTER, L. C. ; WHITTED, T. ; BLINN, J. F. Scan Line Methods for Displaying Parametrically Defined Surfaces. **Communications of the ACM**, v.23, n.1, 1980.
- LEISTER, W. ; MAUS, T. ; MÜLLER, H. ; NEIDECKER, B. ; STÖßER, A. Occursus Cum Novo - Computer Animation by Ray Tracing in a Network. in: N. Magnenat-Thalmann e D. Thalmann (ed.) **New Trends in Computer Graphics**, p. 83-92, Springer-Verlag, 1988.
- LEVER, W. An Efficient Parallel Ray Tracing Scheme for Distributed Memory Parallel Computers. **Parallel Rendering Symposium Proceedings**. San Jose, IEEE/ACM, Oct. 1993.
- LEVNER, G. ; TASSINARI, P. ; MARINI, D. A Simple, General Method for Ray Tracing Bicubic Surfaces. In: Earnshaw, R. A. (ed.) **Theoretical Foundations of Computer Graphics and CAD**, Springer-Verlag Berlin Heidelberg, NATO ASI Series, vol. F40, p.805-820, 1988.
- LISCHINSKI, D. ; GONCZAROWSKI, J. Improved Techniques for Ray Tracing Parametric Surfaces. **The Visual Computer**, v.6, n.6, p.134-152, 1990.
- LOPES, R. D. **O Multicomputador TRGR e a Paralelização da Síntese de Imagens**, Dissertação (mestrado), Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.
- MacDONALD, J. D. ; BOOTH, K. S. Heuristics for Ray Tracing Using Space Subdivision. **The Visual Computer**, v.6, p.153-166, 1990.
- MAISEL, E. ; HEGRON, G. A Realistic Image Synthesis of Animation Sequences Based on Temporal Coherence. **In: Proc. Third Eurographics Workshop on Animation and Simulation**. Cambridge, 1992.
- MÄNTYLÄ, M. **Introduction to Solid Modeling**, Computer Science Press, Rockville, MD, 1988.
- MARSH, S. C. Fine Grain Parallel Architectures and the Creation of High Quality Images. In: Earnshaw, R. A. (ed.) **Theoretical Foundations of Computer Graphics and CAD**, Springer-Verlag Berlin Heidelberg, NATO ASI Series, vol. F40, p.728-754, 1988.

- MARTINS, F. F. D. ; SANTOS, E. T. Ray Tracing Parametrizado - uma técnica para tornar mais rápido o processo de ajuste na síntese de imagens 3D realísticas. **Anais do XII Congresso de Iniciação Científica e Tecnológica em Engenharia - CICTE'93**, S.Carlos - SP, dezembro 1993.
- MAUREL, H. ; DUTHEN, Y. ; CAUBET, R. A 4D Ray Tracing. **Computer Graphics Forum**, v.12, n.3, p.285-294, (Proc. of Eurographics'93), 1993.
- MAX, N. Vectorized Procedural Models for Natural Terrain: Waves and Islands in the Sunset. **Computer Graphics**, v.15, n.3, p.317-324, 1981.
- MAY, D. ; SHEPHERD, R. Communicating Process Computers. **Inmos Technical note 22**, Bristol, Inmos Ltd., 1987.
- McNEILL, M. ; SHAH, B. ; HÉBERT, M. ; LISTER, P. ; GRIMSDALE, R. Performance of Space Subdivision Techniques in Ray Tracing. **Computer Graphics Forum**, v.11, n.4, p.213-222, 1992.
- MILLS, M. I. Image Synthesis - Optical Identity or Pictorial Communication. in: Magnenat-Thalmann, N. ; Thalmann, D. (ed) **Computer Generated Images - The State of the Art. Proc. of Graphics Interface'85**, Springer-Verlag, Tokyo, p.3-10, 1985.
- MONTANI, C. ; SCOPIGNO, R. Rendering Volumetric Data Using the STICKS Representation Scheme. **Computer Graphics**, v.24, n.5, p.87-102, 1990.
- MONTANI, C. ; PEREGO, R. ; SCOPIGNO, R. Parallel Volume Visualization on a Hypercube Architecture. **Proceedings of Workshop on Volume Visualization**, ACM, Boston, p. 9-16, 1992.
- MORRIS, D. T. ; DEW, P. M. Dynamic Dataflow Algorithms for Ray Tracing CSG Objects. in: Dew, P. M. et al. (ed.) **Parallel Processing for Computer Vision and Display**, p. 452-460, Addison-Wesley, 1989
- MURAKAMI, K. ; HIROTA, K. Incremental Ray Tracing. **Proc. of Eurographic's Workshop on Photosimulation, Realism and Physics in Computer Graphics**, p.15-29, 1990.
- MUUSS, M. J. RT & REMRT Shared Memory Parallel and Networked Distributed Ray-Tracing Programs. In: **The Ballistic Research Laboratory CAD Package - Release 4.0 - Vol. V - The BRL-CAD Analyst's Manual**. Advanced Computing Systems - The U.S. Army Ballistic Research Laboratory, Maryland (BRL-CAD Symposium'88), 1991.
- MUUSS, M. J. ; KENNEDY, C. M. ; BUTLER, L. A. The BRL-CAD Benchmark Test. In: **The Ballistic Research Laboratory CAD Package - Release 4.0 - Vol. V - The BRL-CAD Analyst's Manual**. Advanced Computing Systems - The U.S. Army Ballistic Research Laboratory, Maryland (BRL-CAD Symposium'91), 1991.

- NARUSE, T. ; YOSHIDA, M. ; TAKAHASHI, T. ; NAITO, S. SIGHT - A Dedicated Computer Graphics Machine. **Computer Graphics Forum**, v.6, n.4, p.327-334, 1987.
- NEMOTO, K. ; OMACHI, T. An Adaptative Subdivision by Sliding Boundary Surfaces for Fast Ray Tracing. in: **Proceedings of Graphics Interface'86**, p.43-48, 1986
- NEWELL, M. E. ; NEWELL, R. G. ; SANCHI, T. L. A New Approach to the Shaded Picture Problem. in: **Proceedings of the ACM National Conference**, 1973.
- NIEH, J. ; LEVOY, M. Volume Rendering on Scalable Shared-Memory MIMD Architectures. **Proceedings of Workshop on Volume Visualization**, ACM, Boston, p. 17-24, 1992.
- NIELSEN, D. E. A Strategy for Smoothly Transitioning to Massively Parallel Computing. **Energy and Technology Review**, (Nov. 1991), p.20-32, 1991.
- NISHIMURA, H. ; OHNO, H. ; KAWATA, T. ; SHIRAKAWA, T. ; OMURA, K. LINKS-1: A Parallel Pipelined Multimicrocomputer System for Image Creation. In: **10th Symposium on Computer Architecture, SIGARCH'83 - Proceedings**, p.387-94, 1983.
- OHTA, M. ; MAEKAWA, M. Ray Coherence and Constant Time Ray Tracing Algorithm. **Computer Graphics 1987 (Proc. of CG International'87)**. ed. T. L. Kunii, p. 303-314, 1987.
- PACKER, J. Exploiting Concurrency: A Ray Tracing Example. **INMOS Technical note 7**, Inmos Ltd., Bristol, 1987. Também em **The Transputer Applications Notebook - Architecture and Software**, 1st Edition, Inmos Ltd., Bristol, 1989.
- PASSMORE, G. Advanced Ray Tracing. **Computer Graphics World**, v.10, n.7, p.42-44, 1987.
- PETERSON, J. W. Distributed Computation for Computer Animation. **Technical Report UUCS-87-014**, Department of Computer Science, The University of Utah, 1987.
- PHONG, B. T. Illumination for Computer Generated Images. **Communications of the ACM**, v.18, n.6, p.311-317, 1975.
- PITOT, P. The Voxar Project. **IEEE Computer Graphics and Applications**, v.13, n.1, p. 27-33, 1993.
- PLUNKETT, D.J. ; BAILEY, M. J. The Vectorization of a Ray-Tracing Algorithm for Improved Execution Speed. **IEEE Computer Graphics and Applications**, v. 5, n.4, p.52-60, 1985.
- POTMESIL, M. ; CHAKRAVARTY, I. A Lens and Aperture Camera Model for Synthetic Image Generation. **Computer Graphics**, v.15, n.3, p.297-305, 1981.

- POTMESIL, M. ; CHAKRAVARTY, I. Modeling Motion Blur in Computer Generated Images. **Computer Graphics**, v.17, n.3, 389-399, 1983.
- PRIOL, T. ; BOUATOUCH, K. Experimenting with a Parallel Ray-Tracing Algorithm on a Hypercube Machine. In: Duce, D. A. ; Jancene, P. (ed.) **EUROGRAPHICS'88. Proceedings**, p.243-259, 1988.
- PRIOL, T. ; BOUATOUCH, K. Static Load Balancing for a Parallel Ray Tracing on a MIMD Hypercube. **The Visual Computer** , v.5, p.109-119, 1989.
- PULLEYBLANK, R. ; KAPENGA, J. The Feasibility of a VLSI Chip for Ray Tracing Bicubic Patches. **IEEE Computer Graphics and Applications**, v. 7, n. 3, p.33-44, 1987.
- RANSEN, O. F. The Art of Ray Tracing. **Byte**, v. 15, n. 2, p.238-242, 1990.
- REEVES, W. T. The Physical Simulation and Visual Representation of Natural Phenomena. **Computer Graphics**, v.21, n.4, p.335-336, 1987.
- ROBB, P. ; PAWLOWSKI, B. Computer Ray Tracing Speeds. **Applied Optics**, v.29, n.13, p.1933-1939, 1990.
- ROMAN, G. C. ; COX, K. C. A Taxonomy of Program Visualization Systems. **IEEE Computer**, v.26, n.12, p.11-24, December 1993.
- ROTH , S. D. Ray Casting for Modeling Solids. **Computer Graphics and Image Processing**, v.18, n.2, p.109-144, 1982.
- RUBIN, S. M. ; WHITTED, T. A 3-Dimensional Representation for Fast Rendering of Complex Scenes. **Computer Graphics**, v.14, n. 3, p. 110-116, 1980.
- SABELLA, P. A Rendering Algorithm for Visualizing 3D Scalar Fields. **Computer Graphics**, v.22, n.4, p.51-58, 1988.
- SALMON, J. ; GOLDSMITH, J. A Hypercube Ray-Tracer. in: Fox, C. G. (ed.) **Proceedings of the Third Conference on Hypercube Computer and Applications**, 1988.
- SANTOS, E. T. Um Sistema Sistólico para Interpolação de Imagens. **Anais do VI SIBGRAPI - Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens**, Recife, p.133-138, 1993.
- SANTOS, E. T. Movimentação de Fontes de Luz em Ray Tracing Parametrizado. **Anais do Congresso Internacional de Computação Gráfica - CICOMGRAF'94**, São Paulo, 1994a.
- SANTOS, E. T. O Algoritmo de Ray Tracing em Multicomputadores. **Relatório Técnico LSI/EPUSP**, n.36, 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, (submetido), 1994c.
- SCHERSON I. D. ; CASPARY, E. Data Structures and the Time Complexity of Ray Tracing. **The Visual Computer**, v.3, p.201-13, 1987.
- SCHERSON, I. D. ; CASPARY, E. Multiprocessing for Ray Tracing: a Hierarchical Self-Balancing Approach. **The Visual Computer**, v.4, p.188-196, 1988.
- SCHIMMITT, 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.
- SEDERBERG, T. W. ; ANDERSON, D. C. Ray Tracing of Steiner Patches. **Computer Graphics**, v.18, n.3, p.159-164, 1984.
- SÉQUIN, C. ; SMYRL, E. Parameterized Ray Tracing. **Computer Graphics**, v.23, n.3, p.307-314, 1989.
- SLATER, M. Tracing a Ray Through Subdivided N-Dimensional Space. **The Visual Computer**, v.8, n.9, p.39-46, 1992.
- SPEER, L. R. ; DeROSE, T. D. ; BARSKY, B. A. A Theoretical and Empirical Analysis of Coherent Ray Tracing. **Graphics Interface'85**, p.1-8, Canadian Information Processing Society, 1985. Também em **Computer Generated Images: The State of the Art**, N. Magnenat-Thalmann & D. Thalmann (ed.), Springer-Verlag, 1985.
- SPEER, L. R. An Updated Cross-Indexed Guide to the Ray Tracing Literature. **Computer Graphics**, v.26, n.1, p. 41-72, 1992
- STEIN, A. D. ; WANG, Z. ; LEIGHT Jr., J. S. Computer-Generated Holograms: A Simplified Ray-Tracing Approach. **Computers in Physics**, v.6, n.4, p.389-392, Jul/Aug. 1992.
- STÖßER, A. ; SCHMITT, A. ; NEIDECKER, B. "Occursus Cum Novo" - Tools for Efficient Photo-Realistic Computer Animation. **Proc. of Eurographics'88**, (ed) D. A. Duce ; P. Jancene, North Holland, p.31-41, 1988.
- SUTHERLAND, I. E. ; SPROULL, R. F. ; SCHUMAKER, R. A. A Characterization of Ten Hidden-Surface Algorithms. **ACM Computing Surveys**, v.6, n.1, 1974. Também in: **Tutorial: Computer Graphics**, Beatty, J. C. ; Booth, K. S. (ed.), IEEE Computer Society, 2nd. Edition, p.387-441, 1982.
- SWEENEY, M. A. J. ; BARTELS, R. H. Ray Tracing Free-Form B-Spline Surfaces. **IEEE Computer Graphics and Applications**, v.6, n.2, p.41-49, 1986.

- ULLNER, M. K. **Parallel Machines for Computer Graphics**. PhD Thesis, California Institute of Technology, 1983. Report Number 5112:TR:83.
- VANDEWETTERING, M. T. **The MTV RayTracer**. Documentação do programa. Disponível na rede Internet em cs.uoregon.edu (128.223.4.13) por ftp anônimo, 1988.
- WALLACE, J. R. ; ELMQUIST, K. A. ; HAINES, E. A. A Ray Tracing Algorithm for Progressive Radiosity. **Computer Graphics**, v.23, n.3, p.335-344, 1989.
- WARNOCK, J. A Hidden-Surface Algorithm for Computer Generated Half-Tone Pictures. **Univ. of Utah Computer Science Dept. TR 4-15**, NTIS AD-753 671, 1969.
- 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.
- 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.270-274, 1978.
- WILLIAMS, N. S. ; BUXTON, B. F. ; BUXTON, H. Distributed Ray Tracing Using an SIMD Processor Array. In: Earnshaw, R. A. (ed.) **Theoretical Foundations of Computer Graphics and CAD**, Springer-Verlag Berlin Heidelberg, NATO ASI Series, vol. F40, p.703-725, 1988.
- WILLIAMS, G. N. ; NELSON, E. L. ; BARNETT, D. M. ; PARMLEY, K. Visualization of Molecular Dynamics via Ray Tracing and Animation in a Vectorized Environment. **IBM Journal of Research and Development**, v.75, n.1/2, p.108-118, Jan/Mar. 1991.
- WILSON, G. V. A Glossary of Parallel Computing Terminology. **IEEE Parallel & Distributed Technology - Systems and Applications**, v.1, n.1, p.52-67, 1993.
- WOLFF, L. D. ; KURLANDER, D. J. Ray Tracing with Polarization Parameters. **IEEE Computer Graphics and Applications**, v.10, n.6, p.44-55, 1990.
- WOODWARK, J. R. A Multiprocessor Architecture for Viewing Solid Models. **Displays Journal**, v. 5, n.2, p. 97-103, 1984.
- WYVILL, G. ; PEARCE, A. ; WYVILL, B. The Representation of Water. **Proc. of Graphics Interface'86**, Canadian Information Processing Society, p.217-222, 1986.
- YAGEL, R. ; COHEN, D. ; KAUFMAN, A. Discrete Ray Tracing. **IEEE Computer Graphics and Applications**, v.12, n.5, p.19-28, 1992.

ZUFFO, M. K. ; NETTO, M. L. ; LOPES, R. D. Introdução à Computação Gráfica. In: **Adendo aos Anais da Jornada EPUSP/IEEE em Computação Visual**, LSI, São Paulo, 1990.

ZUFFO, M. K. **Um Multicomputador para Aplicações Gráficas**, Dissertação (mestrado), Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.

183 refs.

APÊNDICE I - DICIONÁRIO

Adotou-se nesta dissertação a tradução de termos em inglês. Algumas traduções são consagradas; outras são propostas introduzidas por este trabalho.

aglomerado	<i>cluster</i>
anti-serrilhado	<i>antialiasing</i>
artefato	<i>artifact</i>
árvore hierárquica de envoltórios	<i>hierarchical tree of extents</i>
árvore quaternária	<i>quadtree</i>
barramento	<i>bus</i>
borão de movimento	<i>motion blur</i>
campos de alturas	<i>height fields</i>
chave	<i>switch</i>
circuitaria	<i>hardware</i>
conjunto de dados de trabalho	<i>working set</i>
cor de interferência	<i>interference color</i>
cubo direcional	<i>direction cube</i>
dadoduto	<i>pipeline</i>
elo	<i>link</i>
envoltório	<i>extent, enclosure</i>
fazenda de processadores	<i>processor farm</i>
fila de prioridade	<i>heap</i>
granularidade espessa	<i>coarse-grain</i>
granularidade fina	<i>fine-grain</i>
hospedeiro	<i>host</i>
laço	<i>loop</i>
linhas escondidas	<i>hidden lines</i>
mapeamento de ambiente	<i>environment mapping</i>
matriciação	<i>rasterization</i>
memória de quadro	<i>frame buffer</i>
memorizador	<i>buffer</i>
memorizador de itens	<i>item buffer</i>
memorizador de luz	<i>light buffer</i>
memorizador-z	<i>z-buffer</i>
naco	<i>slab</i>
octárvore	<i>octree</i>
particionamento binário do espaço	<i>BSP - binary space partitioning</i>
passeio interativo	<i>walk-through</i>
pontel	<i>pixel</i>
ponto quente	<i>hot spot</i>
propagação	<i>broadcasting</i>
"ray tracing" direto	<i>forward ray tracing</i>
"ray tracing" exaustivo	<i>exhaustive ray tracing</i>
"ray tracing" reverso	<i>backward ray tracing</i>
reflexão nublada	<i>dull reflection</i>

representação por fronteiras	<i>boundary representation (B-Rep)</i>
retalho	<i>patch</i>
serrilhamento	<i>aliasing</i>
sobrecarga	<i>overhead</i>
sombra difusa	<i>fuzzy shadow</i>
taxa de acertos	<i>hit ratio</i>
taxa de faltas	<i>miss ratio</i>
tonalização	<i>shading</i>
traçado de cones	<i>cone tracing</i>
traçado de feixes	<i>beam tracing</i>
traçado de lápis	<i>pencil tracing</i>
travamento perpétuo	<i>deadlock</i>
travessia de raios	<i>ray traversal</i>
volume envolvente	<i>bounding volume</i>

APÊNDICE II - DESCRIÇÃO E CRÉDITO DAS IMAGENS

A maioria das imagens apresentadas na abertura de cada capítulo desta dissertação foram geradas com um programa de "ray tracing" paralelizado a partir de um programa de domínio público chamado MTV RayTracer (VANDEWETTERING, 1988). Algumas imagens foram modeladas pelo autor desta dissertação usando a linguagem de descrição NFF (Neutral File Format). Outras foram geradas pelo pacote denominado SPD - Standard Procedural Databases (HAINES, 1987), conforme descrito abaixo.

Capítulo 1: Introdução

Bule de chá. Modelo clássico em computação gráfica (CROW, 1987).
Modelado através do SPD (*com alterações*).
Contém 4096 retalhos triangulares e 2 fontes de luz.

Capítulo 2: Síntese Tridimensional de Imagens

Novamente o bule, agora transparente, mostrando todo o potencial da técnica "ray tracing": refração, reflexões múltiplas, sombras.
Modelado através do SPD (*teapot 4*).
Contém 1008 retalhos triangulares e 2 fontes de luz.

Capítulo 3: Conceitos e Definições Básicas em Computação Paralela

Imagem representando o processamento paralelo (grade 2D).
Modelado pelo autor usando a linguagem NFF.
Contém 1024 esferas e 1921 cilindros e 3 fontes de luz.

Capítulo 4: "Ray Tracing" Paralelo: Conceituação, Avaliação e Taxonomia

Imagem representando a taxonomia.
Modelada através do SPD (*com adaptações*).
Contém 3969 retalhos triangulares e 2 fontes de luz.

Capítulo 5: O Desempenho do Algoritmo de "Ray Tracing" em Multicomputadores

Imagem mostrando um "fractal" de esferas (*sphereflake*)
Modelada através do SPD (*balls 4*),
Contém 7381 esferas, 1 plano e 3 fontes de luz.

Capítulo 6: Discussão e Aplicações

Imagem de plastimodelo de mercedes-benz.
Obtida na rede Internet em <http://hoohoo.ncsa.uiuc.edu/docs/info/raytraced>
(World Wide Web) de robm@ncsa.uiuc.edu, autoria desconhecida.

Capítulo 7: Conclusão e Trabalhos Futuros

Imagem de árvore "fractal", representando a semente e o futuro.
Modelada através do SPD (*tree 12, com adaptações*),
Contém 16383 cones e esferas e 7 fontes de luz.