UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA

DEPARTAMENTO DE ENGENHARIA DA COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

SCANLINE:
Um Sistema para Visualização de Imagens Foto-Realistas

por: Tânia Martins Preto

orientador: Prof. Léo Pini Magalhães

Dissertação submetida à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do Título de Mestre em Engenharia Elétrica.

dezembro de 1992

Este exemplar corresponde à redação final da tese defendida por Tania Martins Preto
e aprovada pela Comissão
Julgadora em
Orientador

UNICAMP
BIBLIOTECA CENTRAL

Resumo

Este trabalho tem como objetivo apresentar o Sistema *Scanline*-ProSIm que consiste em um algoritmo do tipo *Scanline-Z-Buffer*, ao qual foram associadas algumas técnicas para melhorar o realismo.

Este sistema é um módulo do Projeto ProSIm, em desenvolvimento pelo Grupo de Computação de Imagens do DCA/FEE/UNICAMP, que tem como função fornecer um ambiente para a síntese de imagens foto-realistas e animação.

O fato deste algoritmo ser relativamente simples e rápido motivou o estudo das possíveis técnicas que podem ser a ele adaptadas, produzindo imagens de melhor qualidade, sem causar um aumento demasiado do custo computacional.

Este trabalho inicia-se com uma revisão dos algoritmos para a remoção de superfícies escondidas existentes na literatura, enfatizando os algoritmos do tipo *Scanline*. Posteriormente, citam-se as técnicas para a simulação de fontes de luz do tipo lâmpada, *spot* e sol; efeitos de tonalização segundo as estratégias de Bouknight, Gouraud e Phong; transparência; sombra e *antialiasing*. Algumas destas técnicas foram implementadas e estão comentadas. Finalizando o trabalho, apresenta-se a implementação do algoritmo *Scanline* juntamente com as técnicas para a simulação dos efeitos acima.

Abstract

This work presents some techniques which can be associated to the *Scanline-Z-Buffer* algorithm, improving realism.

This system is a part of the ProSIm Project, under development by the Image Computing Group at DCA/FEE/UNICAMP. It aims at providing an environment for images synthesis and animation.

Simplicity and speed of this algorithm led us to study the possible techniques that can be used with it to produce high quality images, without increasing its computational time too much.

An overview of the known hidden-surface algorithms in the literature, mainly *Scanline* method is given. Subsequently, techniques to simulate various light sources such as lamp, spot and sun; and other effects such as shading, transparency, shadow and antialiasing are discussed.

Some of these effects were implemented and the results obtained are presented.

Agradecimentos

Gostaria de agradecer a todas as pessoas que estiveram ao meu lado esse tempo todo e que, de algum modo, contribuíram para a realização deste trabalho.

Ao Professor Léo Pini, pelo incentivo, orientação e paciência.

Aos Professores do Departamento de Matemática Aplicada do IMECC/UNICAMP, João Frederico Meyer (Joni) e Maria A. Diniz Ehrhardt (Cheti), cujos exemplos motivaram o meu interesse pela carreira acadêmica.

Também gostaria de agradecer ao amigo Marcelo Hounsell pelas valiosas sugestões, críticas e orientação dos meus "primeiros passos" no *Scanline*.

Aos amigos Armando Delgado, Heraldo Madeira, Jorge Diz, José Olguín e Mário Queiroz pelas dicas e suporte computacional, que foram de fundamental importância.

Aos amigos Gonzalo Besvievsky e Marcus Ponce, pela ajuda nos slides e fotografias.

A todos os amigos do DCA e DENSIS, cuja convivência, auxílio e amizade foram essenciais ao meu crescimento profissional e pessoal.

Em especial, às amigas Ana Cláudia (Mulhé do céu!!), Débora (Dedé), Myriam (Minha filha!!) e Olga, pela amizade, apoio e compreensão nos momentos certos.

A meus pais, exemplos de carinho, determinação e honestidade.

A meu irmão Ari.

Conteúdo

RESUMO i

ABSTRACT ii

AGRADECIMENTOS iii

CONTEÚDO v

LISTA DE FIGURAS ix

1 INTRODUÇÃO 1

2 ALGORITMOS PARA REMOÇÃO DE SUPERFÍCIES ESCONDIDAS 3

2.1 INTRODUÇÃO 3

2.2 CLASSES DE ALGORITMOS 4

2.3 PRINCÍPIOS FUNDAMENTAIS 4

2.3.1 ORDENAÇÃO 5

2.3.2 COERÊNCIA 6

2.4 ALGORITMOS DE SUBDIVISÃO DE ÁREA 7

2.5 ALGORITMOS DE LISTA DE PRIORIDADE 8

2.6 ALGORITMOS Z-BUFFER 9

2.7 ALGORITMOS SCANLINE 10

2.8 ALGORITMOS RAY-TRACING 11
2.9 COMENTÁRIOS FINAIS 11

3 ALGORITMOS SCANLINE **13**
3.1 INTRODUÇÃO 13
3.2 GEOMETRIA DOS OBJETOS 14
3.3 CARACTERÍSTICAS GERAIS 14
3.4 ALGORITMO DE WYLIE E OUTROS 18
3.5 ALGORITMO DE BOUKNIGHT 19
3.6 ALGORITMO DE WATKINS 19
3.7 ALGORITMO DE MYER 21
3.8 ALGORITMO DE HAMLIN E GEAR 22
3.9 ALGORITMO DE INGRAM 24
3.10 ALGORITMO DE CROCKER 24
3.11 ALGORITMO DE DYER E WHITMAN 25
3.12 COMENTÁRIOS FINAIS 25

4 TÉCNICAS PARA OBTENÇÃO DE REALISMO **27**
4.1 INTRODUÇÃO 27
4.2 MODELO DE ILUMINAÇÃO 27
4.3 FONTES DE LUZ 30
4.3.1 INTRODUÇÃO 30
4.3.2 TIPOS DE FONTES DE LUZ 31
4.3.3 O CONTROLE DA DIREÇÃO DA LUZ 32
4.3.4 COMENTÁRIOS FINAIS 33
4.4 TONALIZAÇÃO 34
4.4.1 INTRODUÇÃO 34

4.4.2 MODELO DE BOUKNIGHT 35
4.4.3 MODELO DE GOURAUD 36
4.4.4 MODELO DE PHONG 39
4.4.5 COMENTÁRIOS FINAIS 40
4.5 TRANSPARÊNCIA 41
4.5.1 INTRODUÇÃO 41
4.5.2 ALGORITMOS LINEARES 42
4.5.3 ALGORITMOS NÃO-LINEARES 46
4.5.4 REFRAÇÃO 48
4.5.5 COMENTÁRIOS FINAIS 48
4.6 SOMBRA 49
4.6.1 INTRODUÇÃO 49
4.6.2 CONCEITO DE SOMBRA 49
4.6.3 TIPOS DE SOMBRA 51
4.6.4 PROBLEMAS E SOLUÇÕES NA OBTENÇÃO DA SOMBRA . . . 52
4.6.5 COMENTÁRIOS FINAIS 57
4.7 ANTIALIASING 58
4.7.1 INTRODUÇÃO 58
4.7.2 DESCRIÇÃO DO MÉTODO 59
4.7.3 COMENTÁRIOS FINAIS 63
4.8 COMENTÁRIOS FINAIS 63

5 IMPLEMENTAÇÃO E RESULTADOS OBTIDOS **64**
5.1 INTRODUÇÃO 64
5.2 O PROJETO ProSIm 64
5.3 O SISTEMA SCANLINE 66
5.3.1 ESTRUTURAS DE DADOS 68

5.3.2 ROTINAS 71
5.4 RESULTADOS OBTIDOS 85
5.5 COMENTÁRIOS FINAIS 90

6 CONCLUSÃO **92**

BIBLIOGRAFIA **95**

Lista de Figuras

2.1 Algoritmo Z-Buffer. 10

3.1 Um poliedro. 15
3.2 Descrição da geometria do poliedro. 15
3.3 Plano de rastreamento cortando a cena. 16
3.4 Algoritmo Scanline Padrão. 18
3.5 Tipos de *spans*. 20
3.6 Algoritmo Scanline-Z-Buffer. 23

4.1 Cálculo da intensidade. 35
4.2 Cálculo da normal no vértice. 37
4.3 Interpolação no modelo de Gouraud. 38
4.4 O problema dos polígonos côncavos. 39
4.5 Camadas transparentes. 43
4.6 Espessura da camada transparente. 47
4.7 Sombra gerada por uma fonte de luz pontual. 50
4.8 Partes da sombra: Umbra e Penumbra. 51
4.9 Sombra formada por uma fonte *Spot*. 53
4.10 Sombra formada por uma fonte direcional. 54
4.11 Volume de sombra delimitado por dois polígonos. 57

4.12 O problema de *aliasing*. 58
4.13 Subdivisão de um *pixel*. 61

5.1 Esquema geral do Projeto ProSIm. 65
5.2 Esquema geral do Scanline. 67
5.3 Rotina Principal 74
5.4 Rotina Inicializadora. 75
5.5 Preparação dos objetos. 75
5.6 Preparação dos objetos de sombra. 76
5.7 Criação dos objetos de sombra. 76
5.8 Determinação dos polígonos associados aos vértices. 77
5.9 Cálculo de informações de arestas de objetos. 78
5.10 Cálculo de informações de arestas de sombra. 79
5.11 Determinação de arestas de sombra em uma linha de rastreamento. 79
5.12 Determinação de arestas de objetos em uma linha de rastreamento. 80
5.13 Determinação da intensidade em função do tipo de tonalização. 80
5.14 Determinação da visibilidade de cada *pixel*. 81
5.15 Modelo de iluminação. 82
5.16 Determinação de sombra em cada *pixel*. 83
5.17 Cálculo da intensidade "misturada". 83
5.18 Cálculo das camadas em sombra. 84
5.19 Cena 1: Tonalização de Bouknight, Gouraud e Phong. 86
5.20 Cena 2: Fonte Pontual. 86
5.21 Cena 3: Fonte Direcional. 87
5.22 Cena 4: Fontes *Spot* com diferentes atuações dos raios de luz. 87
5.23 Cena 5: Transparência. 88

Capítulo 1

INTRODUÇÃO

A computação de imagens é uma importante área de conhecimento que envolve a síntese, o processamento de imagens e a visão computacional.

As pesquisas empenhadas em melhorar a relação CUSTO X BENEFÍCIO dos dispositivos gráficos através do desenvolvimento de *hardware* e *software*, têm contribuído muito para a difusão da computação de imagens possibilitando as aplicações nas diversas áreas como a educação, medicina, artes, engenharia, geociências, publicidade e outras.

A área de síntese de imagens refere-se ao tratamento de imagens geradas a partir de uma descrição formal e envolve a modelagem e o *rendering* da cena.

Entende-se o termo CENA , no contexto deste trabalho, como o conjunto de todos os elementos considerados na computação de uma imagem (objetos, observador, fontes de luz) e o termo *rendering* como o processo de criar imagens realistas a partir da posição do observador, considerando todos os elementos da cena e as características visuais dos objetos.

Algumas estratégias para a obtenção de realismo na etapa de *rendering* são:

- Remoção das superfícies escondidas, ou seja, das superfícies que são, do ponto de vista do observador, parcialmente ou totalmente obscurecidas por outras e que portanto, não devem aparecer.
- Uso de fontes de luz e efeitos como tonalização (*shading*), transparência, reflexão, sombra, *antialiasing* e textura às superfícies dos objetos da cena que são visíveis.

Neste trabalho citam-se as principais características dos algoritmos para remoção de superfícies escondidas (*hidden-surface*), dando ênfase aos algoritmos do tipo *Scanline* e também, apresentam-se algumas técnicas para a obtenção dos efeitos citados anteriormente.

O sistema de *Scanline* apresentado neste trabalho é um módulo do Projeto ProSIm (Prototipação e Síntese de Imagens Foto-Realistas e Animação) em desenvolvimento pelo grupo de Computação de Imagens do DCA-FEE-UNICAMP.

O algoritmo básico *Scanline* foi implementado em uma etapa anterior a esta [HM90]. A presente atividade concentra-se no estudo e associação de técnicas para a melhoria do realismo das imagens, conforme citado acima, sem um acréscimo excessivo do custo computacional. Assim, a contribuição deste trabalho situa-se em duas direções:

- Sistema de apoio à pré-visualização no ProSIm: O usuário pode criar as imagens rapidamente, a fim de verificar se estão como idealizou, para posteriormente visualizá-las através dos sistemas de *Ray-Tracing* [BMM89] ou de Radiosidade [QM91], que geram imagens de qualidade superior, porém são muito mais lentos.
- Técnica de visualização final, através da sofisticação do algoritmo.

O capítulo 2, a seguir, cita e fornece as principais características dos algoritmos para remoção de superfícies escondidas encontrados com maior freqüência na literatura consultada, situando o algoritmo *Scanline* dentro das variações existentes.

O capítulo 3 descreve os algoritmos *Scanline* destinados ao tratamento de objetos formados por superfícies planas, enfatizando o algoritmo do tipo *Scanline Z-Buffer* [Mye75].

O capítulo 4 descreve algumas técnicas que podem ser associadas ao algoritmo *Scanline Z-Buffer* na etapa de *rendering*, com o objetivo de melhorar a qualidade da imagem.

O capítulo 5 contém a especificação da implementação do módulo *Scanline* no projeto ProSIm, enfatizando as rotinas que contém as técnicas citadas acima. Também fornece alguns resultados relacionando o tempo de computação da imagem com as diversas opções disponíveis para o *rendering*, além de analisar os melhoramentos visuais obtidos.

O capítulo 6 conclui o trabalho com algumas conclusões e sugestões para trabalhos futuros e para finalizar, cita-se a bibliografia consultada.

Capítulo 2

ALGORITMOS PARA REMOÇÃO DE SUPERFÍCIES ESCONDIDAS

2.1 INTRODUÇÃO

Um problema existente em síntese de imagens é a remoção das superfícies escondidas. Dado um objeto tridimensional, um plano de projeção e a posição do observador, executa-se a projeção do objeto sobre o plano a partir de uma posição pré-definida do último. Para tornar o algoritmo mais eficiente, deve-se considerar apenas as partes visíveis dos objetos da cena ao realizar-se a projeção.

Surgiram várias soluções para resolver este problema, resultando numa diversidade de algoritmos. Estes podem ser classificados segundo o espaço em que operam e baseiam-se nos princípios de ordenação e coerência.

Este capítulo tem como objetivo citar os algoritmos encontrados com maior freqüência na literatura consultada ([FvD90], [Rog85] e [NS79]), bem como descrever a maneira como são classificados e os princípios em que se baseiam.

Os algoritmos *Z-Buffer* serão enfatizados pois nestes se baseia o algoritmo *Scanline-Z-Buffer* que é o mais importante para este trabalho.

2.2 CLASSES DE ALGORITMOS

Tem-se três classes de algoritmos para remoção de superfícies escondidas : os que operam no espaço-objeto, os que operam no espaço-imagem e os algoritmos que operam parcialmente no espaço-objeto e no espaço-imagem [Rog85].

Os algoritmos que operam no espaço-objeto determinam a visibilidade sobre áreas contínuas, geralmente abrangendo todo o plano da imagem. Todas as partes visíveis das superfícies são consideradas independentemente de seu tamanho e localização.

A imagem resultante é descrita em termos de linhas e pontos. O tempo de cálculo tende a crescer com o número de objetos da cena, visíveis ou não.

A maior parte dos algoritmos que operam no espaço-objeto servem para a remoção de linhas escondidas, como por exemplo, o algoritmo de Roberts, Appel, Loutrel e Galimbert. O algoritmo de Weiler-Atherton pode ser usado tanto para a remoção de linhas escondidas como para a remoção de superfícies escondidas.

Os algoritmos que operam no espaço-imagem fornecem uma solução aproximada para a visibilidade. Esta é determinada apenas em um conjunto finito de pontos da imagem, cujo número é determinado pela resolução do dispositivo de saída. São feitas aproximações quanto a visibilidade dos objetos localizados entre os pontos amostrados. Exemplos de algoritmos do tipo espaço-imagem são o algoritmo de Warnock, o *Z-Buffer*, *Ray-Tracing* e *Scanline*.

Os algoritmos que operam parcialmente no espaço-objeto e no espaço-imagem são chamados de algoritmos de lista de prioridade. A maior parte dos cálculos destes algoritmos é feita em coordenadas do objeto e somente a parte final é feita em coordenadas da imagem. Exemplos de algoritmos deste tipo são o algoritmo de Newell, o de Newell and Sancha, e o de Schumacker.

2.3 PRINCÍPIOS FUNDAMENTAIS

Os algoritmos para remoção de superfícies escondidas baseiam-se em dois princípios fundamentais ao analisarem uma cena: ordenação e coerência [SSS74].

2.3.1 ORDENAÇÃO

A maior parte dos algoritmos para remoção de superfícies escondidas executa ordenações sobre listas que contém informações sobre polígonos (faces) e (ou) arestas. Estas serão acessadas posteriormente para a obtenção e armazenamento de informações, e execução de cálculos comparativos. Existem diversos procedimentos destinados a realizar ordenação, tais como *Bubble Sort, Shell Sort, Quick Sort, Tree Sort* e *Radix Sort.*
Os algoritmos para remoção de superfícies escondidas executam uma ou mais ordenações ao longo dos eixos x, y e z; sendo que a maneira de fazê-las é um fator de distinção entre os diferentes tipos de algoritmos [Joy88]. A seguir essa diferença é mostrada para alguns algoritmos, onde os eixos x, y correspondem as coordenadas do plano de imagem, ou seja, do dispositivo de visualização.

Z-Buffer

Para todos os objetos:
    Para todos os *pixels*[1] do dispositivo de visualização:
        Compare o valor de z (profundidade).

Ray-Tracing

Para todos os *pixels* do dispositivo de visualização:
    Para todos os objetos:
        Compare o valor de z.

Algoritmo do Pintor

Ordene os objetos pelo valor de z.
Para todos os objetos (na ordem z):
    Para todos os *pixels* do dispositivo de visualização:
        "Pintar" no *frame-buffer.*

[1] Cada um dos elementos da matriz de pontos do dispositivo de visualização, também referido como *picture element.*

Algoritmo Scanline

Ordene os objetos pelo valor de y.

Para todo y:

Ordene objetos por x.

Para todo x:

Compare z.

2.3.2 COERÊNCIA

Coerência é a constância local de determinadas características da cena. A exploração da coerência é importante pois minimiza a quantidade de cálculos necessários para o processamento da mesma.

A exploração das várias formas de coerência torna o algoritmo mais eficiente, pois diminui a quantidade de dados a serem armazenados e o tempo necessário ao processamento.

Algumas formas de coerência [FvD90] são as seguintes:

- Coerência de objetos:

  Caso dois objetos não estejam se sobrepondo a partir do ponto de vista do observador, não é necessário determinar a visibilidade entre os mesmos.

- Coerência de faces:

  As características da superfície de uma face geralmente variam muito pouco ao longo da mesma, sendo facilmente determinadas por cálculos incrementais. Além disso, é freqüente o caso em que algumas características são constantes em toda a face.

- Coerência de arestas:

  Idem ao caso anterior, além disso a visibilidade de uma aresta só é alterada quando esta intercepta outra aresta visível ou penetra em uma face.

- Coerência de linhas:

  Os trechos visíveis em uma linha de rastreamento diferem pouco dos trechos visíveis da linha anterior.

- Coerência de área:

  Dois *pixels* vizinhos freqüentemente pertencem a mesma face visível, possuindo características comuns.

- Coerência de profundidade:

  A partir do valor de profundidade de um *pixel*, pode-se obter a profundidade dos outros *pixels* pertencentes a mesma face por cálculos incrementais.

- Coerência de quadros:

  Em seqüências de imagens para animação, são observadas poucas mudanças nas características de uma imagem em relação a imagem anterior, conseqüentemente torna-se possível a utilização de informações obtidas na imagem anterior na determinação da imagem corrente.

2.4 ALGORITMOS DE SUBDIVISÃO DE ÁREA

Estes algoritmos exploram a coerência de área, ou seja, aproveitam a semelhança que existe entre *pixels* pertencentes a uma determinada região do dispositivo de visualização chamada de "janela". Nesta classe destacam-se os seguintes algoritmos:

- ALGORITMO DE WARNOCK [War69]

  Este algoritmo pertence a classe espaço-imagem e as janelas são determinadas de acordo com a complexidade[2] da parte geométrica da cena.

  Sucessivas subdivisões são feitas até que seja possível determinar a visibilidade dentro da janela em questão, ou até que a janela se reduza a um único *pixel*, partindo-se então, para a análise das outras janelas.

  A eficiência deste algoritmo pode ser aumentada através do uso de critérios mais elaborados para a subdivisão de área.

[2] A complexidade de uma cena, no contexto deste trabalho, é determinada, no que diz respeito a parte geométrica, pela quantidade de objetos e de sobreposições destes entre si. A complexidade total da cena inclui a complexidade da parte geométrica juntamente com o número de fontes de luz, atributos de tonalização, transparência, sombras e texturas.

- ALGORITMO DE WEILER-ATHERTON [WA77]

  Este algoritmo pertence a classe espaço-objeto. Este algoritmo é uma variação do algoritmo anterior com a diferença de que a janela é um polígono no plano de projeção.

  Weiler e Atherton elaboraram um algoritmo para fazer o recorte de polígonos em relação a outros polígonos, que podem ser côncavos e possuir buracos.

2.5 ALGORITMOS DE LISTA DE PRIORIDADE

Estes algoritmos executam ordenação dos objetos ao longo de uma direção, como a dos valores crescentes de z (ordem de visibilidade do ponto de vista do observador) ou de grupos de objetos. Estes objetos são comparados entre si e caso exista sobreposição ou interpenetração dos mesmos, eles são subdivididos.

As etapas de ordenação, comparação e partição dos objetos são feitas no espaço-objeto e a exibição das partes visíveis da cena é feita na precisão do dispositivo de visualização, ou seja, no espaço-imagem.

Os algoritmos pertencentes a esta classe diferem entre si na maneira de executar a ordenação e a partição dos objetos da cena. Alguns destes algoritmos são:

- ALGORITMO DE NEWELL, NEWELL E SANCHA

  Este algoritmo é chamado de "Ordenação de Profundidade" (*Deph-Sort*) e possuí duas versões: uma destinada ao tratamento de polígonos [NNS72] e uma extensão para o tratamento de objetos tridimensionais [New74].

  A forma mais simples deste algoritmo, onde pressupõe-se a não existência de objetos interpenetrantes, é conhecida como o "Algoritmo do Pintor" onde o nome resulta da analogia feita entre a maneira que uma seqüência de objetos é "pintada" no *frame-buffer* do computador e a maneira que uma seqüência de objetos é "pintada" em um quadro. Nos dois casos os objetos são pintados na ordem de proximidade crescente do observador.

- ALGORITMO DE SCHUMACKER [SBS69]

  Este algoritmo, assim como o anterior, pode ser aplicado tanto para o tratamento de polígonos como de objetos tridimensionais, mas possui a restrição de permitir apenas polígonos convexos na cena.

  Fuchs, Kedem e Naylor [FKN80] desenvolveram um algoritmo baseado no algoritmo de Schumacker para calcular as relações de visibilidade de um conjunto de polígonos a partir de posições arbitrárias do observador. Este algoritmo destina-se ao pré-processamento de uma cena em seqüências de animação, onde o ambiente é estático e apenas o observador muda de posição.

2.6 ALGORITMOS Z-BUFFER

O algoritmo básico *Z-Buffer* foi inicialmente desenvolvido por Catmull [Cat75] e caracteriza-se por operar no espaço-imagem, sendo um dos mais simples algoritmos para a remoção de superfícies escondidas. Variações deste algoritmo surgiram com o objetivo de melhorar a qualidade das imagens obtidas.

O algoritmo de Catmull processa os objetos da cena em ordem arbitrária. Para cada *pixel* do dispositivo de visualização, armazenam-se no *frame-buffer* as informações de cor e de profundidade dos objetos mais próximos do observador. Se não houver objetos, armazenam-se as informações referentes ao fundo da cena.

Para cada objeto da cena, é feita a análise de todos os seus *pixels*. Caso o valor da profundidade de um objeto em uma determinada localização do *pixel* seja maior que o valor armazenado, então as informações de cor e profundidade devem ser substituídas pelos novos valores.

Entre as vantagens deste algoritmo destacam-se a simplicidade e a facilidade com que são tratadas as cenas geometricamente complexas. Uma desvantagem pode ser a necessidade de grande quantidade de armazenamento em alguns casos. Melhoramentos do algoritmo tem sido obtidos com o desenvolvimento de implementações em *hardware*. Outra desvantagem é a dificuldade e o custo para a associação de técnicas para *antialiasing* e transparência pelo fato dos *pixels* serem introduzidos em ordem arbitrária no *frame-buffer*.

A figura 2.1 contém o algoritmo básico *Z-Buffer*.

```
INICIALIZAR O FRAME-BUFFER COM
    INTENSIDADE DE FUNDO
    PROFUNDIDADE MÍNIMA

PARA CADA OBJETO DA CENA:

    PARA CADA PIXEL (X, Y) DO OBJETO

        COMPARAR O VALOR DE PROFUNDIDADE (Z) COM O VALOR ARMAZENADO:
        SE   Z > Zarmazenado

            ENTÃO

                ARMAZENAR OS NOVOS VALORES DE PROFUNDIDADE E
                INTENSIDADE
```

Figura 2.1: Algoritmo Z-Buffer.

Algumas extensões para o algoritmo *Z-Buffer* foram sugeridas por Cook [Coo86], Williams [Wil78] e Logatti [Log90]. O primeiro apresentou um *Z-Buffer*-Estocástico que produz imagens com *antialiasing* e de boa qualidade. O segundo adaptou o algoritmo para produzir o efeito de sombra. O último apresenta uma implementação em *hardware*.

2.7 ALGORITMOS SCANLINE

Estes algoritmos operam no espaço-imagem. Diferem dos algoritmos de Warnock, Z-Buffer e Lista de Prioridade pelo fato de não processarem os objetos da cena aleatoriamente e sim na ordem crescente das linhas de rastreamento (valores de y), ou seja, uma linha por vez. A análise da cena, feita no espaço tridimensional pela maioria dos algoritmos, passa a ser feita no espaço bidimensional.

São feitas ordenações ao longo dos eixos x, y e z, e a coerência entre linhas consecutivas é explorada através de cálculos incrementais.

Duas grandes vantagens destes algoritmos são a economia de tempo de processamento e de memória para armazenar informações.

Técnicas para a obtenção de efeitos como sombra, transparência, tonalização, textura e fontes de luz podem ser incorporadas aos algoritmos *Scanline*, fornecendo um grande aumento do realismo das imagem geradas, sem ocasionar um aumento exagerado do custo computacional.

Pelo fato deste trabalho concentrar sua atenção nestes algoritmos, os mesmos serão detalhados no próximo capítulo.

2.8 ALGORITMOS RAY-TRACING

O algoritmo *Ray-Tracing*, desenvolvido inicialmente por Appel [App68], considera que a intensidade que chega ao observador é proveniente dos efeitos de reflexão, refração e transmissão da luz, através das superfícies dos objetos.

Com o objetivo de tornar o algoritmo mais rápido e obter imagens de melhor qualidade, várias extensões [Dre89] têm sido desenvolvidas, concentrando sua atenção nos seguintes aspectos:

- Soluções eficientes para os testes de interseção do raio com os objetos.
- Restrição do espaço de busca, evitando assim, cálculos desnecessários de interseção.
- Introdução de novos modelos de iluminação.

2.9 COMENTÁRIOS FINAIS

Técnicas de pós-filtragem para a obtenção de *antialiasing* podem ser aplicadas às imagens obtidas pelos algoritmos de Schumacker, Warnock e Catmull melhorando a qualidade das imagens.

Dentre os algoritmos citados neste capítulo, os mais populares são o *Ray-Tracing*, devido a alta qualidade da imagem e o *Scanline*, devido à sua rapidez e ao fato de necessitar

de pouca memória para armazenamento. Baseado nestes dois algoritmos, Dreux em [Dre89], apresenta um algoritmo híbrido que busca unir as vantagens de ambos: rapidez e qualidade.

No Projeto ProSIm, os trabalhos de J. Diz [Diz], G. Besvievsky [Bes92] e C. Bannwart [BMM89] tratam da técnica de *Ray-Tracing* e o trabalho de M. Queiroz [Que] apresenta um algoritmo híbrido de *Ray-Tracing* e *Radiosidade.*

O próximo capítulo destina-se à descrição dos algoritmos *Scanline*, que constituem a base do presente trabalho.

Capítulo 3

ALGORITMOS SCANLINE

3.1 INTRODUÇÃO

Este capítulo apresenta as principais características dos algoritmos *Scanline*. Estes algoritmos subdividem-se em dois grupos em função da superfície dos objetos a serem tratados, são elas:

- Algoritmos para superfícies algébricas (curvas).
- Algoritmos para superfícies poligonais (facetadas).

Para o tratamento de superfícies algébricas pode-se utilizar os algoritmos de Blinn [Bli78a], Lane and Carpenter [LC79] e o de Clark [Cla79].

O tratamento de objetos sólidos (primitivas) referidos como CSG *(Constructive Solid Geometry)*, pode ser feito pelo algoritmo de Atherton [Ath83] que é uma modificação do *Spanning Scanline* apresentado no item 3.6.

Neste trabalho são citados apenas os algoritmos destinados a tratar superfícies poligonais, enfatizando o algoritmo de Myers (*Scanline Z-Buffer*) [Mye75], pois a este foram direcionadas as técnicas para a obtenção de realismo descritas no capítulo 4. O algoritmo *Spanning Scanline*, também apresenta facilidades para incorporar as mesmas técnicas de realismo.

Os demais algoritmos são citados sucintamente. Alguns destes são extensões de outros. É o caso dos algoritmos de Crocker e Dyer que se baseiam em modificações dos algoritmos de Atherton e Myers, respectivamente.

3.2 GEOMETRIA DOS OBJETOS

Como visto anteriormente, os algoritmos *Scanline* são elaborados em função do tipo de superfície dos objetos a serem tratados.

Utiliza-se neste trabalho a representação de objetos (primitivas) através de superfícies poligonais que proporciona facilidades como o rápido acesso às informações de polígonos e arestas, independentemente do conjunto.

A descrição dos objetos é feita por "malhas de polígonos" com algumas características da representação B-Rep [Rog85]. Cada objeto é descrito como um conjunto de polígonos. Estes são formados por arestas, cada uma das quais ligada por dois vértices.

A figura 3.1 ilustra um objeto em forma de tetraedro, onde:

- f1, f2, f3 e f4 são as faces,
- a1, a2, a3, a4, a5 e a6 são as arestas,
- v1, v2, v3 e v4 são os vértices.

A descrição deste objeto, da maneira que foi elaborada neste trabalho, pode ser observada na figura 3.2.

O sistema *Scanline*-ProSIm armazena estas informações em uma estrutura de dados chamada POLYHEDRON, descrita no item 5.3.1.

3.3 CARACTERÍSTICAS GERAIS

Os algoritmos *Scanline* resolvem o problema das superfícies escondidas linha por linha. O processamento inicia-se a partir da linha que possui o mais alto valor de y para a que possui o mais baixo valor.

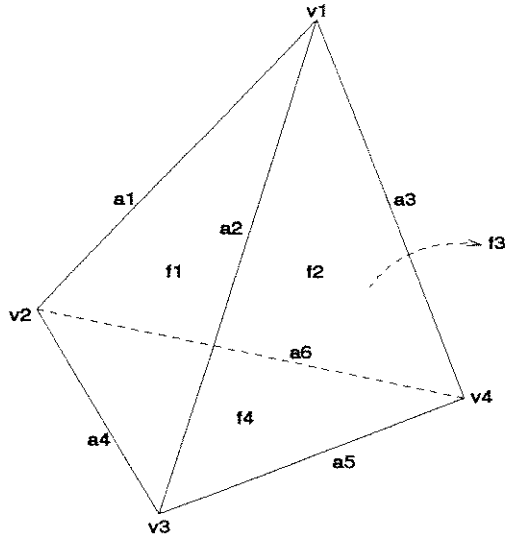

Figura 3.1: Um poliedro.

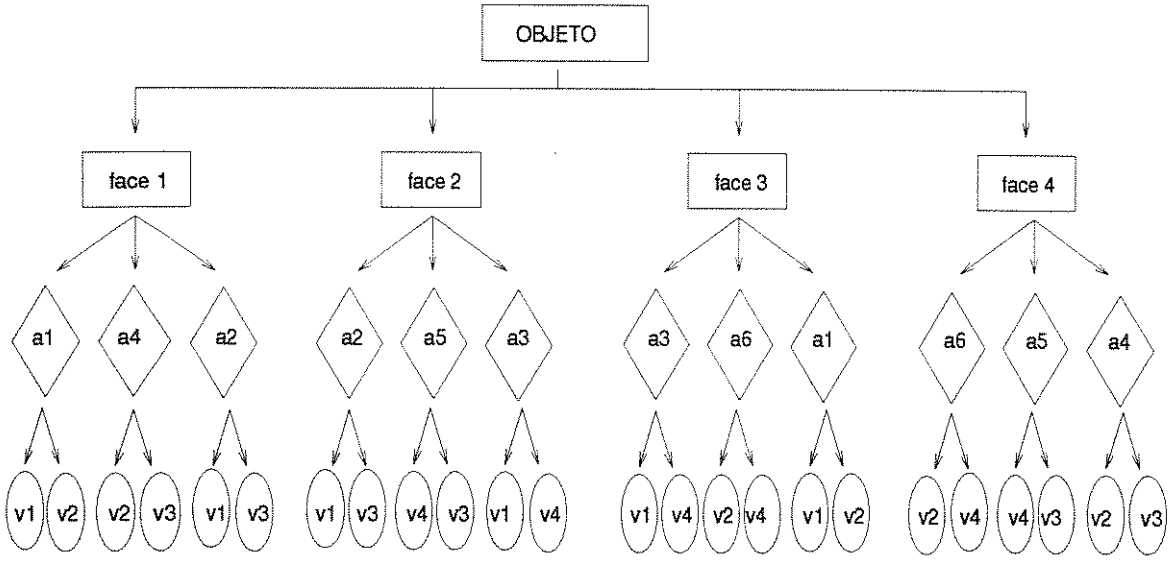

Figura 3.2: Descrição da geometria do poliedro.

Partindo do princípio de que é mais simples calcular a imagem para uma linha da cena do que para a cena inteira, estes algoritmos reduzem o problema de três dimensões para duas, determinando a cada linha um plano chamado plano de rastreamento. Este passa pela linha e pelo ponto onde o observador está localizado. Parte-se então, para o cálculo de todas as interseções deste plano com os objetos da cena, como ilustra a figura 3.3.

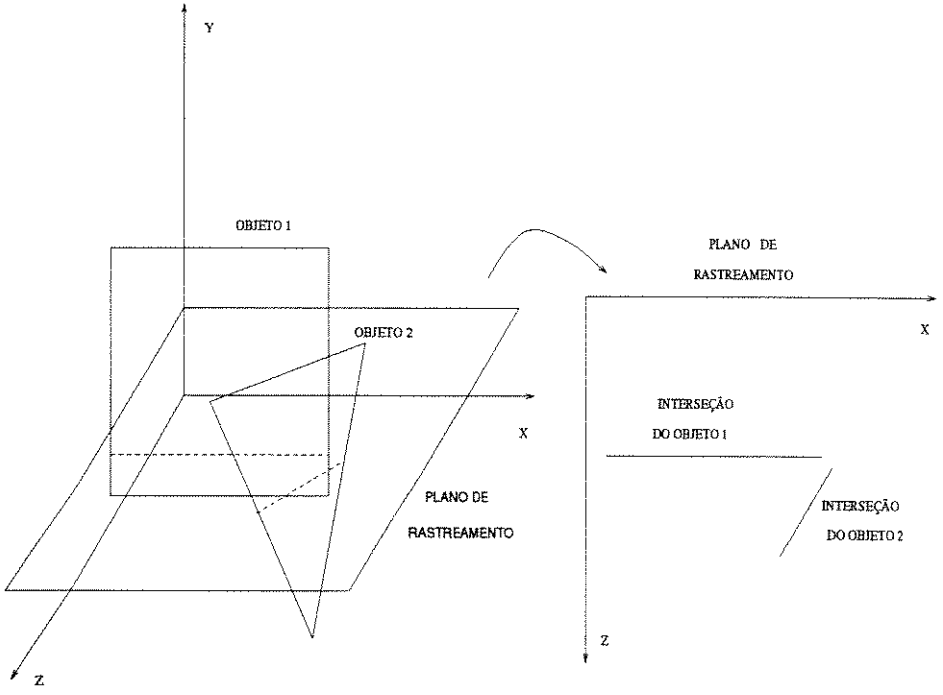

Figura 3.3: Plano de rastreamento cortando a cena.

Conforme visto na seção 2.3.1 os algoritmos *Scanline* fazem ordenações de listas ao longo dos eixos Y, X e Z, nesta seqüência, com os seguintes objetivos:

- ORDENAÇÃO EM Y:

  Identifica para cada posição y da lista, correspondente à uma linha de rastreamento, quais as arestas dos polígonos que estão se iniciando nesta posição.

- ORDENAÇÃO EM X:

  Para cada linha, ordenam-se as arestas interceptadas pela mesma, obedecendo a ordem crescente dos valores de x.

- ORDENAÇÃO EM Z:

  Para cada *pixel* da linha, todas as camadas, correspondentes a localização do mesmo, são armazenadas na ordem de proximidade do observador. Geralmente esta ordenação é feita para algoritmos que tratam de objetos transparentes e também sombra, os demais apenas fazem comparações para determinar as faces mais próximas do observador.

É possível explorar a coerência entre linhas de rastreamento consecutivas e entre *pixels* vizinhos pertencentes a mesma linha através, por exemplo, de cálculos incrementais.

No primeiro caso, as informações ao longo das arestas são atualizadas, evitando assim, cálculos mais complexos. No segundo caso, aproveita-se da tendência de *pixels* vizinhos pertencerem ao mesmo polígono e conseqüentemente possuírem as mesmas características, para utilizar valores já calculados.

Existem algumas estruturas de dados, utilizadas pelos algoritmos *Scanline*, que facilitam a manipulação das informações e aumentando a eficiência dos mesmos [Rog85]. São elas:

- Y-BUCKET:

  Lista de dimensão igual ao número de linhas de rastreamento da imagem. Em cada posição da lista, referente à uma linha, está contida a informação da primeira aresta do polígono que está se iniciando na correspondente posição. As outras arestas são encontradas através da primeira pois estão numa lista ligada.

- LISTA DE POLÍGONOS ATIVOS - LPA:

  Lista que contém o índice dos polígonos que estão sendo interceptados em uma linha, chamados de polígonos ativos. Esta lista é atualizada para cada linha de rastreamento.

- LISTA DE ARESTAS ATIVAS - LAA:

  Lista que contém o índice das arestas que estão sendo interceptadas em uma linha, chamadas de arestas ativas. Esta lista é atualizada a cada linha de rastreamento.

As diferenças básicas entre os algoritmos *Scanline* encontram-se na maneira de executar as comparações ao longo dos eixos x, y e z, no modo de determinar as partes visíveis da cena e de explorar a coerência. A figura 3.4 contém o algoritmo *Scanline* padrão.

```
PARA CADA OBJETO
  PARA CADA POLÍGONO
    PARA CADA ARESTA:
      INSERÍ-LA NA LISTA Y-BUCKET ATRAVÉS DE SEU MAIS ALTO VALOR

PARA CADA LINHA DE RASTREAMENTO
  VERIFICAR QUAIS SÃO AS ARESTAS INTERCEPTADAS PELA LINHA DE RASTREAMENTO
  PARA CADA INTERSEÇÃO DE ARESTA:
    CALCULAR OS PONTOS TERMINAIS (pontos de interseção)
    ORDENAR AS ARESTAS PELOS VALORES CRESCENTES DE X DESTES PONTOS
    PERCORRER A LAA E
     PARA CADA PIXEL LOCALIZADO ENTRE DUAS ARESTAS CONSECUTIVAS:
      CONSTRUIR A LISTA DE PROFUNDIDADE (Z)
      DETERMINAR AS CAMADAS VISÍVEIS DA CENA (mais próximas ao observador)
      DETERMINAR A INTENSIDADE DO PIXEL A PARTIR DAS CARACTERÍSTICAS DO OBJETO
  SE EXISTIREM ARESTAS TERMINANDO NESTA LINHA
    ENTÃO RETIRÁ-LAS DA LAA
```

Figura 3.4: Algoritmo Scanline Padrão.

3.4 ALGORITMO DE WYLIE E OUTROS

Este foi o primeiro algoritmo *Scanline* a ser proposto [WEE67]. Possui as características gerais comuns aos algoritmos deste tipo, sendo que processa apenas polígonos triangulares.

Os polígonos são ordenados pelo valor da coordenada y e z. Utiliza-se a lista LPA que contém triângulos interceptados pela linha de rastreamento.

Esta lista é atualizada a cada linha e executa-se também a ordenação dos triângulos ativos na ordem crescente das coordenadas x.

3.5 ALGORITMO DE BOUKNIGHT

Este algoritmo [BK70] possui características semelhantes ao algoritmo de Wylie, diferindo na maneira de manipular a lista de polígonos ordenada em x.

O algoritmo calcula, a cada linha de rastreamento, um conjunto contendo os valores das coordenadas x das partes visíveis da cena, chamados de *key-squares*.

A cada polígono da cena associa-se um índice de estado que identifica se o mesmo está sendo interceptado pela linha corrente.

Neste caso são feitas são feitas comparações de visibilidade ao longo do eixo z. com o objetivo de determinar as partes mais próximas ao observador.

A exploração da coerência entre as linhas é feita da seguinte maneira: Ordenam-se as arestas ao longo do eixo x somente se existe alguma aresta começando ou terminando na linha corrente, caso contrário, utiliza-se a mesma ordenação da linha anterior.

3.6 ALGORITMO DE WATKINS

Este algoritmo, chamado de *Spanning Scanline* [Wat70], executa ordenação de polígonos ao longo do eixo x, diferindo do anterior na maneira de determinar a visibilidade da linha de rastreamento.

A visibilidade é determinada a cada *span*[1], escolhido segundo um determinado critério e nos quais executam-se comparações de profundidade.

[1]Seqüência de *pixels* que determinam um trecho de um polígono.

O uso de *spans* representa uma forma unidimensional de coerência de área, pois a partir da comparação da profundidade em apenas um *pixel* do trecho, determina-se a visibilidade do trecho inteiro.

Cada *span* é tomado inicialmente como sendo o trecho entre duas arestas, se não for possível determinar sua visibilidade, muda-se o ponto de comparação ou subdivide-se o *span*. Têm-se três tipos básicos de ocorrência de *span* [NS79]:

- SPAN VAZIO : Nenhum polígono é visto pelo observador.
- SPAN COM UM SEGMENTO : Apenas um polígono intercepta o *span*, não são necessárias as comparações de profundidade.
- SPAN COM DOIS OU MAIS SEGMENTOS : É necessário calcular a profundidade de cada segmento e fazer comparações a fim de verificar quem está mais a frente (maior valor de z).

Estes tipos estão ilustrados na figura 3.5(a).

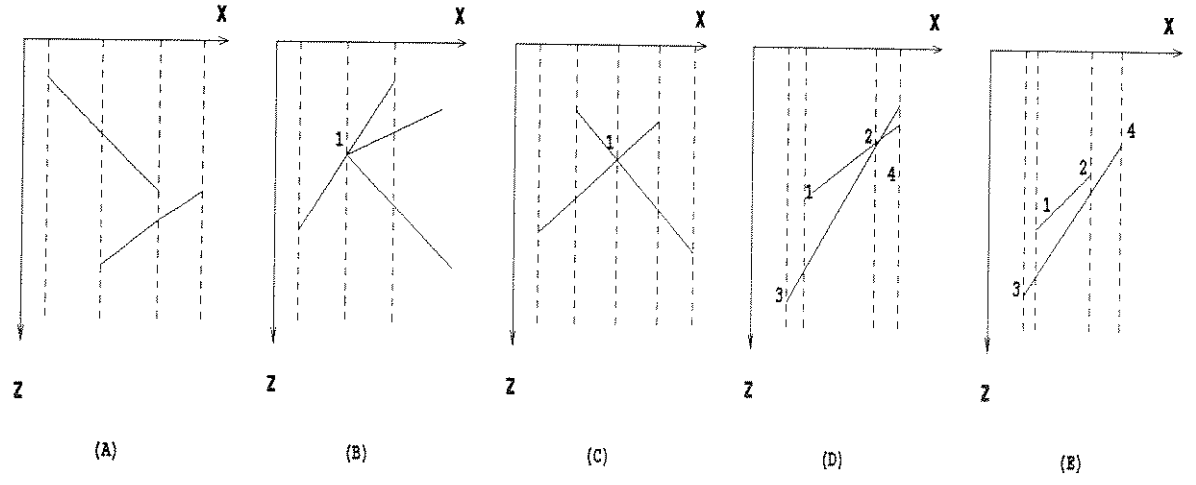

Figura 3.5: Tipos de *spans*.

A ocorrência de polígonos interpenetrantes determina o critério de comparação. Caso estes não existam, as comparações de profundidade podem ser feitas nos pontos extremos do *span*. No entanto, os mesmos podem ser coincidentes nas extremidades do *span*, partindo-se então, para as comparações no ponto médio (figura 3.5(b)).

Se há ocorrência de polígonos interpenetrantes, o *span* não apenas deve ser comparado no ponto médio, como também deve ser dividido no ponto de interseção (figura 3.5(c)).

Para assegurar a obtenção do resultado correto, Watkins sugeriu que a comparação de profundidade fosse feita no ponto médio do *span*. Mesmo no caso em que os polígonos não se interpenetram e nem são coincidentes nos extremos do *span*, é possível que se obtenha resultados inconsistentes. A figura 3.5(d) ilustra um caso em que a comparação feita nos extremos fornece o resultado correto. Porém, na figura 3.5(e), se a comparação fosse feita entre os pontos 2 e 4, o resultado seria incorreto.

Pode-se também, explorar a coerência entre *spans* pertencentes à linhas de rastreamento consecutivas. Em [Rog85] são apresentados os fluxogramas de várias versões deste algoritmo, que usam diferentes critérios para a determinação da visibilidade do *span*. A escolha adequada destes, influencia diretamente a eficiência do algoritmo.

As técnicas mais eficientes buscam reduzir o número de *spans* e conseqüentemente, diminuem a quantidade de cálculos.

3.7 ALGORITMO DE MYER

Este algoritmo é conhecido como *Scanline-Z-Buffer* [Mye75]. É um caso particular do algoritmo *Z-Buffer*, diferindo na maneira de processar os objetos da cena.

O *Z-Buffer* processa os objetos em ordem aleatória no dispositivo de visualização. Já no algoritmo de Myers [Mye75], os objetos são ordenados pelo valor da coordenada y e são processados nesta ordem, ou seja, uma linha por vez em ordem crescente.

A cada linha de rastreamento, é inicializada a estrutura de dados que contém o valor da intensidade de fundo e o valor mínimo de z para cada *pixel*.

O algoritmo difere do algoritmo de Watkins na maneira de calcular a visibilidade de trechos da linha. Para cada ponto pertencente a um trecho de polígono entre duas arestas consecutivas, executa-se a comparação de seu valor de profundidade com o valor armazenado na estrutura, sendo que dois casos podem ocorrer:

1. O valor armazenado corresponde ao valor de fundo *background*, ou seja, nenhum polígono foi armazenado nesta posição.

   Então as informações da nova camada serão armazenadas na posição da estrutura correspondente ao ponto.

2. A profundidade armazenada é diferente do valor de fundo, ou seja, algum polígono já está ocupando esta posição.

   Caso o valor da nova camada seja maior que o valor armazenado, então armazenam-se as novas informações na posição correspondente.

   Caso seja menor, então mantém-se os mesmos valores, pois a nova camada está atrás da camada armazenada na estrutura.

A visualização de objetos que se interceptam é feita corretamente, pois as comparações são feitas para todos os *pixels* de todas as camada visíveis. Polígonos côncavos também são representados de maneira correta.

Para a implementação deste algoritmo foram utilizadas as estruturas LAA e Y-BUCKET contendo o índice das arestas. O algoritmo *Scanline Z-Buffer* está ilustrado na figura 3.6.

3.8 ALGORITMO DE HAMLIN E GEAR

Este algoritmo [HG77] foi elaborado com o objetivo de reduzir o número de operações aritméticas durante a determinação da profundidade. Os objetos da cena são formados por polígonos planos convexos que não se interceptam. Assim como os algoritmos anteriores, este também realiza ordenação das arestas em y, armazenando-as na Y-BUCKET e em x, utilizando a LAA.

A coerência entre linhas consecutivas é explorada da seguinte maneira: Uma vez que os polígonos não se interceptam, considera-se então, que as profundidades relativas entre eles não mudam. Novos cálculos são necessários somente quando a ordem da LAA é alterada, ou seja, quando existem arestas começando ou terminando na linha de rastreamento corrente. Neste caso, a visibilidade pode ser determinada a partir da análise da mesma antes da mudança e na análise do tipo das arestas envolvidas na mudança.

```
PARA CADA OBJETO
    PARA CADA POLÍGONO
        PARA CADA ARESTA
            INSERÍ-LA NA LISTA Y-BUCKET COM SEU MAIS ALTO VALOR
                                        (construção da y-bucket)
PARA CADA LINHA DE RASTREAMENTO
    INICIALIZAR OS VALORES DO "FUNDO":
        VALOR MÍNIMO DE PROFUNDIDADE
        INTENSIDADE DE FUNDO
    CONSULTAR O Y-BUCKET REFERENTE A LINHA E
    SE (EXISTIREM ARESTAS INICIANDO-SE)
                (ou seja: y-bucket[linha] diferente de zero)
        ENTÃO
            COLOCÁ-LAS NA LAA
        SENÃO
            RETIRÁ-LAS DA LAA
    SE (ENTRARAM ARESTAS OU SAIRAM ARESTAS)
        ENTÃO
            ORDENAR A LAA NA ORDEM CRESCENTE DOS VALORES CRESCENTES DE X
            PERCORRER A LAA E
            PARA CADA PAR DE ARESTAS CONSECUTIVAS:
            SE (PERTENCEREM AO MESMO POLÍGONO)
                ENTÃO
                    PARA CADA PIXEL ENTRE AS DUAS ARESTAS:
                        VERIFICAR A VISIBILIDADE:
                        SE (Zpoligono > Zarmazenado)
                            ENTÃO
                                ATUALIZAR O VALOR DE Z
                                CALCULAR A INTENSIDADE DA NOVA CAMADA
```

Figura 3.6: Algoritmo Scanline-Z-Buffer.

Hamlin e Gear desenvolveram duas versões para este algoritmo que são *Stack* e *Cross*, diferindo entre si na maneira de determinar a visibilidade a partir da LAA.

3.9 ALGORITMO DE INGRAM

Este algoritmo [Ing84] explora a coerência entre conjuntos de elementos chamados de *clusters*. Este é o termo dado a um aglomerado ou grupo conectado de polígonos e arestas, após terem sido eliminados os polígonos que não estavam de frente para o observador.

A medida que um *cluster* é encontrado, determina-se a sua prioridade e esta será utilizada para as linhas seguintes. Esta ordem de prioridade é igual ao número de faces entre o *cluster* e o observador, portanto, quanto mais próximo está o *cluster* do observador, mais baixo será o seu valor.

Este algoritmo executa poucos cálculos para determinar a visibilidade mesmo em cenas complexas, exceto na determinação inicial da prioridade.

O tempo de processamento aumenta linearmente com o aumento da complexidade da cena.

3.10 ALGORITMO DE CROCKER

O algoritmo de Crocker [Cro84] é um *Scanline* modificado. Utiliza a técnica de coerência de invisibilidade que tem como objetivo de executar apenas uma quantidade mínima de cálculos sobre objetos que são ocultos por outros. O algoritmo executa um processamento para remover partes de objetos que provavelmente não serão visíveis.

Crocker adaptou esta técnica ao algoritmo de Atherton para CSG, no entanto, esta pode ser adaptada também a outros algoritmos *Scanline* que tratam tanto superfícies poligonais (Watkins, Bouknight), como algébricas (Carpenter, Whitted and Blinn).

Este algoritmo apresenta melhoramentos na ordenação e a técnica de coerência de invisibilidade é baseada na exploração da coerência entre linhas consecutivas, coerência

de profundidade e de entre *spans*.

3.11 ALGORITMO DE DYER E WHITMAN

Este algoritmo [DW87] é uma modificação do algoritmo de Myers. A arquitetura de um computador vetorial é explorada a fim de aumentar a eficiência do algoritmo.

O investimento em *hardware* possibilita o uso de paralelismo em várias etapas durante o *rendering*. Isto significa que uma seqüência de instruções pode ser aplicada à uma seqüência de elementos simultaneamente, ao invés de ser aplicado a dados individuais de elementos.

São executados os mesmos procedimentos do algoritmo de Myers, porém utiliza-se paralelismo nas seguintes etapas:

- Cálculo do vetor normal às superfícies,
- Interpolação de informações referentes às arestas (normal, posição, intensidade, etc),
- Interpolação de valores entre pares de arestas,
- Cálculo dos valores de intensidade,
- Cálculo da área dos *pixels* quando do uso da técnica de *antialiasing*.

3.12 COMENTÁRIOS FINAIS

Os algoritmos *Scanline* fazem freqüentes acessos às listas de dados e também executam ordenações sobre as mesmas. A utilização de um método eficiente para a ordenação e a adequada exploração da coerência, influencia diretamente o desempenho do algoritmo.

No entanto, deve-se tomar cuidado com a exploração da coerência. Como a execução de cálculos incrementais está totalmente baseada em resultados anteriores, então a probabilidade de haver acumulo de erros e alteração do resultado final é alta.

A utilização de implementações em *hardware* que permitem a execução de tarefas simultaneamente, como o algoritmo de Whitman e Dyer, contribui muito para aumentar a eficiência do algoritmo, sendo uma opção interessante.

O presente trabalho optou pelo uso de algoritmo *Scanline-Z-Buffer*, pelo fato deste ser um algoritmo relativamente simples e rápido, possuir poucas restrições quanto ao tipo de objetos tratados e permitir facilmente a incorporação de melhoramentos.

Capítulo 4

TÉCNICAS PARA OBTENÇÃO DE REALISMO

4.1 INTRODUÇÃO

Após a modelagem da cena, parte-se para a etapa de *rendering*. Os algoritmos para remoção das superfícies escondidas permitem a correta determinação das partes dos objetos visíveis ao observador.

Para a obtenção de imagens mais realistas, algumas técnicas podem ser associadas ao algoritmo *Scanline Z-Buffer* para obter-se a simulação de fontes de luz, efeitos de tonalização, transparência, sombra e *antialiasing*, sem contudo, comprometer exageradamente o tempo de processamento.

Este capítulo destina-se à descrever estas técnicas e, a fim de assegurar seu entendimento, o modelo de iluminação utilizado é descrito sucintamente no item 4.2.

4.2 MODELO DE ILUMINAÇÃO

O valor da intensidade em um ponto da superfície de um objeto é obtido através dos modelos de iluminação.

Estes modelos têm sido desenvolvidos com o objetivo de englobar o maior

número possível de aspectos do ambiente, considerando várias características, tais como a rugosidade ou polimento da superfície, cor, geometria da superfície e da fonte [FvD90].

A intensidade recebida pelo observador está em função da reflexão, transmissão e absorção da luz que incide sobre a superfície.

Os modelos de iluminação que consideram a interação entre vários componentes da cena, tais como fontes de luz e outros objetos, são chamados de globais. Por sua vez, os modelos que consideram apenas o efeito da fonte incidente sobre o objeto, são chamados de locais. O modelo utilizado no módulo *Scanline*-ProSIm é local e para simplificação, considera apenas os efeitos de reflexão.

Esta reflexão pode ser dividida em três componentes: ambiente, especular e difusa.

- COMPONENTE AMBIENTE:

  Também chamada de luz de fundo. Incide igualmente sobre todos os objetos da cena, sendo refletida em todas as direções. Para este trabalho não está sendo considerada a rugosidade da superfície dos objetos. Representa uma fonte de luz distribuída, emitida por todo os objetos da cena em maior ou menor escala, tais como paredes, mobília, etc.

- COMPONENTE DIFUSA:

  Luz proveniente de uma fonte cujos raios não alcançam igualmente a superfície dos objetos iluminados e que, ao incidir sobre as mesmas, espalha-se em diversas direções. A luminosidade da superfície depende da direção e da distância da fonte.

  Esta componente varia de material para material e provém, principalmente, de superfícies rugosas ou granuladas. A luz se origina de uma mesma fonte de luz.

- COMPONENTE ESPECULAR:

  Luz que determina o brilho de uma superfície simulando os *highlights*. Estes são luzes concentradas (brilhos) ao redor de pontos onde os raios incidentes, provenientes de uma fonte, têm direção igual ou próxima à direção do observador. Esta componente é bem evidente em superfícies polidas. A luz se origina de uma mesma fonte de luz, como na componente difusa.

O modelo de iluminação implementado foi o modelo de Phong [Pho75]. São utilizadas algumas aproximações para a simulação das componente da reflexão, uma vez que a simulação perfeita envolveria cálculos complexos e demorados.

A intensidade em um ponto de uma superfície é obtida através da seguinte equação:

$$Int_{total} = Int_{amb} + \sum_{j=1}^{n\_fon} (Int_{dif} + Int_{esp}) \tag{4.1}$$

onde,

- Int_{amb} é a componente ambiente da luz,
- Int_{dif} é a componente difusa,
- Int_{esp} é a componente especular,
- n_fon é o número de fontes de luz.

A componente ambiente é independente das fontes de luz, correspondendo a luz ambiente, sendo obtida através da multiplicação do valor desta (considerada uma constante) pelo valor do coeficiente de reflexão ambiente (k_a).

As componentes difusa e especular são consideradas para cada fonte de luz, sendo seus valores somados.

A componente difusa é função do cosseno de *beta*, ângulo entre a linha do ponto (da superfície) à fonte de luz e a normal no ponto, e também do coeficiente de reflexão difusa (k_d). Estes valores são multiplicados pela intensidade de cada fonte de luz.

A componente especular é função do cosseno de *alpha*, ângulo entre a direção de visão do observador e a direção de reflexão, e também do coeficiente de reflexão especular (k_s). Assim como a componente difusa, estes valores também são multiplicados pela intensidade de cada fonte de luz.

A distribuição espacial da luz refletida (*highlight*), pode ser controlada durante o cálculo da componente especular. Este controle é obtido elevando-se o valor do cosseno a

um expoente (n), determinando o quão concentrada ou espalhada é a região de brilho sobre a superfície. N é o índice que simula a rugosidade da superfície. Para um espelho perfeito, n deveria ser infinito e a direção de visão igual à direção de reflexão.

As componentes difusa e especular podem ser melhor representadas se for considerado o efeito da atenuação da luz em função da distância. Considera-se aqui, que a atenuação é linear, sendo determinada em função da distância da fonte ao ponto (d) e de uma constante arbitrária (k), usada para controlar o efeito da atenuação.

Os valores de k_a, k_d, k_s, n e k são determinados pelo usuário, assim como os valores da intensidade da luz ambiente e das demais fontes de luz, chamados respectivamente de I_A e I_{Lj}.

A equação 4.1 pode então, ser reescrita na seguinte forma:

$$Int_{total} = I_A \times k_a + \frac{\sum_{j=1}^{n_fontes} (I_{Lj} \times k_d \times cos\beta + I_{Lj} \times k_s \times cos^n\alpha)}{d \times k} \tag{4.2}$$

Tem-se a seguinte correspondência do lado direito da equação 4.1 em relação ao lado direito da equação 4.2:

$$Int_{amb} = I_A \times k_a$$

$$Int_{dif} = I_{Lj} \times k_d \times cos\beta$$

$$Int_{esp} = I_{Lj} \times k_s \times cos^n\alpha$$

Alguns efeitos citados nos próximos itens, são obtidos através da manipulação dos parâmetros desta equação.

4.3 FONTES DE LUZ

4.3.1 INTRODUÇÃO

A simulação de fontes de luz é de grande importância para a obtenção de mais realismo em uma cena. A fonte emite raios que determinam um valor de intensidade,

ao atingirem uma superfície, representando a quantidade de energia luminosa refletida e transmitida.

A determinação da intensidade de um ponto da superfície, é feita em função da orientação e propriedades da mesma, além da direção e geometria das fontes de luz que atuam sobre o objeto.

Este item descreve sucintamente alguns tipos de fontes de luz. A simulação destas é feita através da manipulação dos parâmetros da equação de iluminação (equação 4.2).

Maiores detalhes sobre a descrição dos atributos para a modelagem das fontes de luz são encontrados em [VG84].

4.3.2 TIPOS DE FONTES DE LUZ

As fontes de luz encontradas com maior freqüência no dia a dia, podem ser classificadas em quatro tipos básicos:

- FONTE DE LUZ AMBIENTE:

  Citada em 4.2, incide igualmente sobre a superfície dos objetos e não tem posição definida.

  O resultado da tonalização de objetos iluminados somente por esta fonte é um brilho constante em todo o objeto. Uma esfera, por exemplo, é visualizada como sendo um disco [Pop89].

- FONTE EXTENSA:

  Esta fonte é determinada por uma área de onde partem os raios de luz. Ocorre quando se tem uma fonte grande e próxima, tal como um fluorescente comprido. No caso de haver formação de sombras pela obstrução da passagem dos raios de luz originários desta fonte, estas possuirão umbra e penumbra (item 4.6.2).

- FONTE DIRECIONAL:

  A fonte de luz é considerada no infinito e geometricamente fica definida por um vetor unitário que determina sua direção.

Todos os raios provenientes desta fonte são paralelos atuando da mesma maneira sobre toda a superfície. Um exemplo típico desta fonte é o Sol. Devido a enorme distância em relação à Terra, os raios que aqui chegam, são aproximadamente paralelos.

- FONTE PONTUAL:

  As dimensões desta fonte são pequenas quando comparadas ao tamanho de um objeto. Por esta razão, sua posição é definida por um ponto no espaço.

  Os raios de luz partem da fonte e se espalham em todas as direções. Um exemplo deste tipo de fonte é a lâmpada.

  A representação destas fontes de luz oferece parâmetros que podem ser manipulados permitindo o controle da atuação dos raios de luz, limitando-se a região atingida pelos raios luminosos. Estas fontes modificadas são conhecidas como *Spot*, sendo descritas no item 4.3.3.

4.3.3 O CONTROLE DA DIREÇÃO DA LUZ

As fontes do tipo *Spot* podem ser simuladas através da determinação de uma direção de atuação, em torno da qual os raios de luz são emitidos. Só é iluminada a região próxima a essa direção de atuação.

As componentes difusa e especular da equação 4.2 são consideradas em função da proximidade do ponto calculado osiçção e direção de atuação da fonte *Spot*. Calcula-se um fator de atenuação que altera o valor dessas componentes obtendo-se o efeito desejado.

A seguir estão descritas duas estratégias para o controle da direção da luz [Rog85], uma proposta por Phong e a outra por Warn:

- MODELO DE PHONG:

  Considera-se que a quantidade de luz recebida num ponto depende do ângulo β, localizado entre a direção de atuação da fonte e a direção da fonte ao ponto . Este ângulo é elevado a um expoente c que controla a abertura do "cone" de luz.

Um valor alto para c produz um "cone" de luz estreito. Por sua vez, um valor pequeno produz um "cone" mais largo, permitindo que os raios de luz se espalhem mais.

- MODELO DE WARN:

  Em estúdios fotográficos limita-se a área de concentração de luz, adaptando-se objetos chamados "flaps" e "cones" nas fontes comuns. Warn em [War83] propõem a simulação desses efeitos da seguinte maneira:

  - FLAPS: Confina o efeito luminoso numa região limitada ao longo das coordenadas x, y ou z. A intensidade de um ponto da superfície só é calculada, caso este esteja localizado dentro do limite determinado para o "flap".
  - CONES: Confina o efeito luminoso numa região cônica. Determina-se um ângulo limite que representa a direção máxima de atuação dos raios de luz em relação à direção de atuação da fonte.

    Enquanto o valor do ângulo formado entre a direção do raio de luz no ponto e a direção da fonte for menor que o ângulo máximo de atuação, considera-se o efeito desta fonte sobre os objetos. Caso contrário, despreza-se.

4.3.4 COMENTÁRIOS FINAIS

A simulação de diferentes fontes de luz pode ser feita através da manipulação dos parâmetros das componentes difusa e especular da equação 4.2, sendo um procedimento relativamente simples e compensador em vista dos melhoramentos oferecidos.

A possibilidade de controlar-se a direção da luz, permite que detalhes da cena sejam evidenciados, fornecendo assim, excelentes resultados visuais, além de ampliar as opções do usuário.

4.4 TONALIZAÇÃO

4.4.1 INTRODUÇÃO

A tonalização de superfícies pode ser definida como o tratamento da distribuição da luz sobre uma superfície iluminada, proporcionando a suavização das cores [PM91].

O modelo de iluminação calcula a intensidade de cor de um ponto específico da superfície em função da normal. Se esta for conhecida em cada ponto da superfície, pode-se aplicar o modelo de iluminação a cada um destes pontos e obter um efeito bem realista de tonalização.

As técnicas de tonalização são elaboradas em função do tipo de superfície do objeto. Para superfícies poligonais, têm-se três modelos básicos desenvolvidos:

- Modelo de Bouknight.
- Modelo de Gouraud.
- Modelo de Phong.

Ao aplicar-se um destes modelo, deve-se levar em consideração os objetivos da aplicação e também, o tipo de objeto que se deseja simular.

No caso de objetos em que é desejável manter a aparência facetada de suas superfícies, como por exemplo um cubo, pode-se obter bons resultados utilizando apenas um valor de intensidade para cada superfície. Para este caso, o modelo de Bouknight é o mais apropriado.

Quando se utiliza um conjunto de polígonos para aproximar superfícies que não são planas, como por exemplo uma esfera, ocorre que o uso de uma intensidade a cada superfície não produz resultado satisfatórios, pois obtém-se aparência facetada.

Assim, torna-se necessário o uso de modelos mais elaborados, que considerem a variação da intensidade ao longo de toda a superfícies e não apenas em um ponto da mesma. Os modelos de Gouraud e Phong determinam esta variação de diferentes maneiras.

Este item tem como objetivo descrever as características principais de cada um destes modelos, bem como suas vantagens e desvantagens.

4.4.2 MODELO DE BOUKNIGHT

Este modelo foi desenvolvido por Bouknight [Bou70], sendo chamado de tonalização *Flat* ou Constante.

O cálculo da intensidade é feito somente uma vez para cada polígono, ou seja, o valor calculado em um ponto será o mesmo para todos os pontos do polígono, atribuindo a este, intensidade constante.

Para o cálculo do valor da intensidade (equação 4.2), é necessário o conhecimento do valor da normal e das coordenadas do ponto atingido pelo raio de luz.

No módulo *Scanline*-ProSIm utiliza-se o algoritmo de Martin-Newell [NS79] para o cálculo do vetor normal e toma-se o baricentro do polígono, como sendo o ponto atingido pelo raio de luz.

A figura 4.1 ilustra o cálculo da intensidade utilizando como parâmetros o valor do ponto P e a direção do vetor normal.

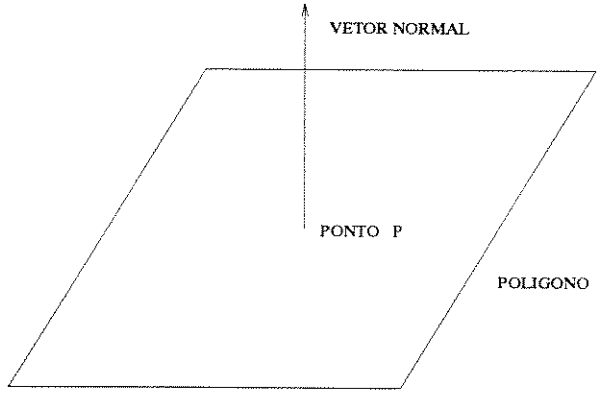

Figura 4.1: Cálculo da intensidade.

4.4.3 MODELO DE GOURAUD

Este modelo, criado por Gouraud [Gou71], procura eliminar as descontinuidades na intensidade, encontrando um valor aproximado para a mesma em cada ponto.

O cálculo da intensidade é feito a cada vértice dos polígonos e para os outros pontos, a intensidade é obtida através da interpolação dos valores dos vértices. Gouraud sugere que a intensidade de um vértice seja calculada em função das características dos polígonos adjacentes a ele, caso não esteja disponível a informação precisa do vértice.

Nem sempre é desejável que a intensidade entre polígonos vizinhos seja interpolada (ex.: cubo), portanto, é necessário estabelecer critérios que verifiquem a aplicação deste procedimento.

No *Scanline*-ProSIm estabeleceu-se um critério que compara o ângulo formado entre todas as normais de todos os polígonos adjacentes a um vértice. Se o ângulo entre duas normais for menor que 60 graus (parâmetro que pode ser modificado), então os correspondentes polígonos devem ser interpolados entre si, portanto, o valor da normal no vértice comum será compartilhado por ambos.

Como no modelo anterior, para o cálculo da intensidade, necessita-se de conhecer o valor da normal e das coordenadas do ponto, que neste caso é o próprio vértice. Para a obtenção do valor da normal no vértice, executa-se o seguinte procedimento:

- calcule o valor do vetor normal referente a cada polígono do objeto;
- verifique, para cada vértice, quais são os polígonos associados a ele e aplicar o critério citado;
- calcule a média aritmética das normais dos polígonos a serem suavizados (determinados pelo critério), para a obtenção da normal.

A figura 4.2 ilustra o cálculo da normal em um vértice a partir da normal dos polígonos adjacentes: A normal no ponto P (NP), é obtida a partir das normais N1, N2, N3 e N4 aos polígonos adjacentes pol1, pol2, pol3 e pol4, respectivamente.

O valor da normal NP, que posteriormente deve ser normalizado, pode ser aproximado como:

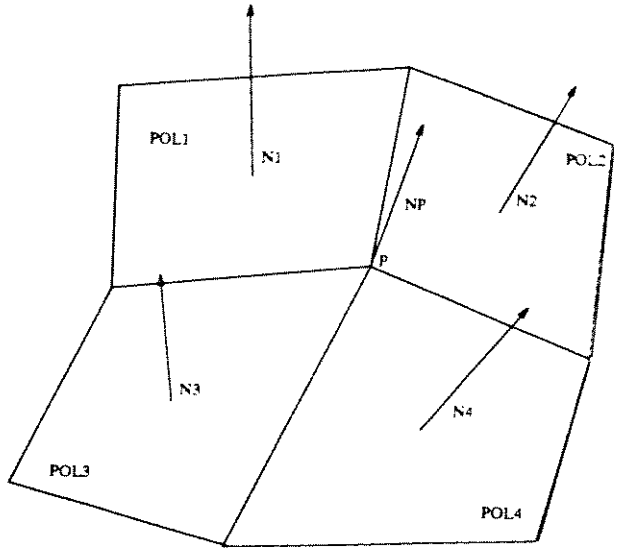

Figura 4.2: Cálculo da normal no vértice.

$$NP = \frac{(N1 + N2 + N3 + N4)}{4} \tag{4.3}$$

Conhecendo-se o valor da normal, pode-se calcular o valor da intensidade em cada vértice, através do modelo de iluminação.

O valor da intensidade, ao longo das arestas, é obtido através da interpolação dos valores das intensidades de seus vértices. Ao longo das linhas de rastreamento, o mesmo é calculado a partir da interpolação das intensidades dos pontos de interseção da linha com as arestas.

A figura 4.3 ilustra esta idéia: A intensidade ao longo da aresta 1 é calculada através da interpolação dos valores da intensidade nos vértices V1 e V3, analogamente para a aresta 2 e os vértices V2 e V4. A intensidade em qualquer ponto P, ao longo da linha de rastreamento, é obtida pela interpolação das intensidades dos pontos P1 e P2.

Este modelo remove a aparência de descontinuidade entre faces adjacentes, na representação de uma superfície, devido ao fato das intensidades nos limites (arestas) se igualarem [Hea86].

No entanto, existem algumas deficiências. O fato do valor da normal no interior

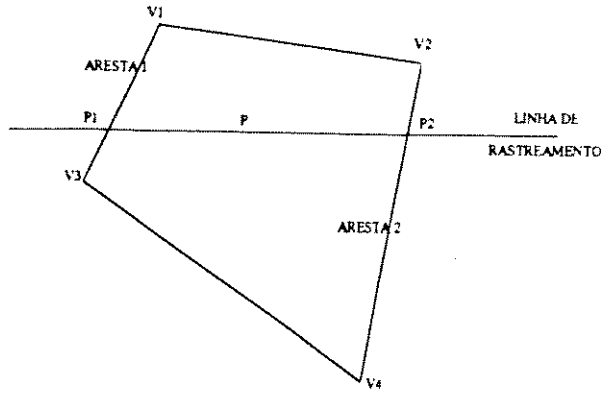

Figura 4.3: Interpolação no modelo de Gouraud.

do polígono não ser diretamente considerado durante o cálculo da intensidade, causa uma descontinuidade na curva de intensidade de cor.

Existe uma diferença entre a cor interpolada e a cor calculada tendo como parâmetro a normal interpolada. No primeiro caso ocorre uma mudança mais acentuada na variação da normal entre polígonos vizinhos o que já não ocorre no segundo caso.

Por essa razão, as regiões de alta reflexão especular, ou seja, regiões que contém os pontos de brilho, geralmente não são bem representadas. Observa-se uma diferença de intensidade entre polígonos vizinhos mais pronunciada, sendo percebida pelo olho humano. Este problema é conhecido como efeito *Mach-Band* [NS79].

Um outro problema pode aparecer ao tratar-se de polígonos côncavos. A intensidade dentro destes nem sempre é calculada da maneira adequada. Este fato é ilustrado pela figura 4.4. A intensidade no ponto P é calculada interpolando-se as intensidades dos pontos P1 e P2. No entanto, o ponto M também deveria ser levado em consideração pois está próximo de P. Como isto não ocorre, observa-se uma descontinuidade da intensidade na região próxima ao ponto M.

Uma maneira de contornar estes problemas seria subdividir os polígonos. Porém, isto causaria outros problemas, como a necessidade de mais memória para armazenamento, maior número de cálculos para remover as superfícies escondidas e determinar a intensidade; pois tem-se mais polígonos e conseqüentemente, mais vértices e arestas.

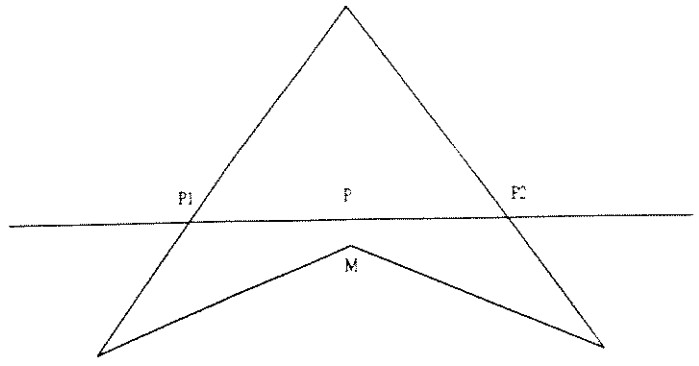

Figura 4.4: O problema dos polígonos côncavos.

4.4.4 MODELO DE PHONG

Este modelo [Pho75], considera as variações do vetor normal ao longo da superfície. É possível simular com mais exatidão o efeito da luz sobre cada ponto de incidência, obtendo-se informações que permitem uma boa representação das propriedades especulares dos objetos.

O vetor normal é calculado para todos os pontos da superfície de um objeto e em função deste, calcula-se a intensidade.

Como no modelo de Gouraud, o vetor normal é calculado para todos os vértices do polígono, em função dos vetores normais às faces adjacentes a estes (vide figura 4.2 e equação 4.3).

Posteriormente, a normal é calculada ao longo das arestas e das linhas de rastreamento através da interpolação de seus valores nos vértices. Este procedimento é análogo ao cálculo da intensidade interpolada no modelo de Gouraud.

Para o exemplo mostrado na figura 4.3 a interpolação é feita da seguinte maneira: O vetor normal ao longo da aresta 1 é calculado através da interpolação dos vetores normais nos vértices V1 e V3. Analogamente para a aresta 2 e os vértices V2 e V4. O vetor normal em qualquer ponto P, ao longo da linha de rastreamento, é obtido pela interpolação dos vetores normais nos pontos P1 e P2.

Após a determinação do vetor normal, realiza-se o cálculo do valor da intensidade em cada ponto, através da aplicação do modelo de iluminação.

Devido ao fato de ser considerado, em cada ponto, o vetor normal à superfície e ser possível calcular a direção do raio refletido, tem-se uma representação bem realista da reflexão especular. Como conseqüência, o efeito de Mach-Band, citado no item 4.4.3, é reduzido e os pontos de brilho são produzidos com mais nitidez, obtendo-se mais realismo que nos modelos anteriores (Bouknight e Gouraud).

Por outro lado, devido a necessidade de executar muitos cálculos, este modelo requer mais tempo para processamento que os anteriores. No entanto, este aumento é compensado pelo benefícios obtidos em termos de qualidade da imagem. Apesar disto, o modelo de Phong, não fornece ainda a solução para um problema existente no modelo de Gouraud: a descontinuidade da tonalização de polígonos côncavos.

4.4.5 COMENTÁRIOS FINAIS

Alguns algoritmos foram desenvolvidos com o objetivo de melhorar o desempenho dos modelos de tonalização.

Como foi dito, o modelo de tonalização de Phong produz imagens com alto grau de realismo. No entanto, este não é muito apropriado para aplicações em sistemas onde necessita-se produzir imagens em tempo real. Em vista deste problema, Tom Duff em [Duf79] propõe uma aproximação para o cálculo da tonalização, buscando reduzir o número de operações necessárias para a determinação da intensidade, acelerando assim, o processo de tonalização.

Bishop em [BW86] propõe uma nova formulação para o modelo de Phong, onde é introduzido o uso da série de Taylor na aproximação da equação da iluminação, buscando reduzir, ainda mais, o número de cálculos.

Um método híbrido que busca combinar a rapidez do modelo de Gouraud com a qualidade do modelo de Phong é apresentado em [Pop89], sendo chamado de *Highlight Shading*. A idéia básica deste método é subdividir os polígonos nas regiões de alta reflexão especular até que os valores de intensidade, obtidos pelo modelo de Gouraud, estejam próximos dos valores obtidos pelo modelo de Phong. Assim, a intensidade destas regiões é calculada com mais precisão.

O modelo de Phong proporciona um bom resultado para uma superfície curva aproximada por um conjunto de polígonos planos, fornecendo uma imagem de alta qualidade e representando realisticamente os pontos de brilho.

O uso da interpolação linear para aproximar a normal em cada ponto do polígono no modelo de Phong, não garante a continuidade da derivada primeira na curva de distribuição da intensidade. No caso onde ocorre uma mudança abrupta da orientação entre polígonos vizinhos, tem-se que o efeito de Mach-Band é visível ao longo da aresta que estes compartilham. Entretanto, este efeito é muito menos visível no modelo de Phong do que no modelo de Gouraud.

Na produção de seqüências de imagens para animação, tem-se que o modelo de Gouraud não é muito adequado. Neste caso, podem ser percebidas diferenças de intensidade em imagens consecutivas, devido ao efeito de Mach-Band, sendo mais adequado o uso do modelo de Phong [NS79].

Uma opção interessante para diminuir o tempo de processamento dos modelos de tonalização, é a associação de técnicas em *hardware* que permitem a execução de diferentes tarefas ao mesmo tempo. Em vista dos melhoramentos de qualidade da imagem obtida pelo uso do modelo de Phong[1], o gasto extra para a associação de *hardware* será bem compensado.

4.5 TRANSPARÊNCIA

4.5.1 INTRODUÇÃO

A representação de objetos transparentes contribui muito para melhorar o realismo de uma imagem, além de aumentar as opções do usuário.

A simulação deste efeito no *Scanline*-ProSIm é feita através do uso de uma função de transparência. Esta função altera o valor da intensidade de um *pixel* de acordo com a quantidade de camadas transparentes e o coeficiente de transparência das mesmas. Em alguns casos, outras informações relativas ao objeto podem ser levadas em consideração.

[1] Principalmente em aplicações onde há necessidade da representação da reflexão especular em sequências de animação.

Cada camada transparente atua como se fosse um filtro da intensidade, onde a quantidade desta que passa é proporcional ao coeficiente de transparência do objeto.

Para a obtenção destes efeitos, vários algoritmos foram desenvolvidos. Estes diferem na maneira de calcular a função de transparência e nos parâmetros considerados durante o cálculo. Neste item, serão descritos os seguintes algoritmos:

- Algoritmos lineares (não-refrativos).
- Algoritmos não-lineares (não-refrativos).
- Algoritmos que consideram a refração dos raios de luz.

Os primeiros algoritmos são usados com maior freqüência pois são de fácil implementação e o aumento do custo computacional devido a sua associação é baixo. A qualidade das imagens produzidas por estes algoritmos é satisfatória para muitas aplicações, porém não oferece alto grau de realismo.

Com o objetivo de melhorar a qualidade surgem os algoritmos não-lineares que procuram considerar maiores atenuações ocorridas nas arestas silhuetas de uma superfície transparente. Outra opção, criada com o mesmo objetivo, são os algoritmos que consideram a refração dos raios de luz, obtendo efeitos mais realistas.

Tanto os algoritmos lineares como os não-lineares são facilmente associados aos algoritmos *Scanline*, o que não ocorre com os algoritmos que consideram a refração. Os próximos itens destinam-se à descrever as principais características destes algoritmos.

4.5.2 ALGORITMOS LINEARES

Para a simulação da transparência, deve-se considerar todas as camadas transparentes visíveis ao observador. O algoritmo básico *Scanline*, que até então considerava apenas a camada mais próxima do observador, para o cálculo da intensidade final, deve ser modificado de maneira a considerar as várias camadas transparentes.

Em [Mam89] é apresentado um método onde executa-se um armazenamento de todas as camadas transparentes da cena, ordenando-as pelo valor crescente da coordenada z. Esta ordenação é feita a nível de *pixel*, ou seja, para cada posição x de uma linha de

rastreamento (no caso de algoritmos *Scanline*), são armazenadas informações referentes à todas as camadas que interceptam essa posição.

Para fins de otimização, estabelece-se que a camada opaca mais próxima ao observador será a camada limite de profundidade e as camadas situadas atrás desta, sejam opacas ou transparentes, serão desprezadas. No caso de não haver camadas opacas em uma determinada posição, considera-se o fundo da cena como a camada limite.

A figura 4.5 contém algumas camadas transparentes e uma camada opaca. Para o *pixel* localizado na posição Xa são armazenadas as camadas 1, 2 e 3, uma vez que a camada seguinte (última) é opaca, desprezam-se as camadas localizadas atrás desta (4 e 5). Para o *pixel* da posição Xb, armazenam-se as camadas 2 e 5 e como última camada considera-se o fundo da cena.

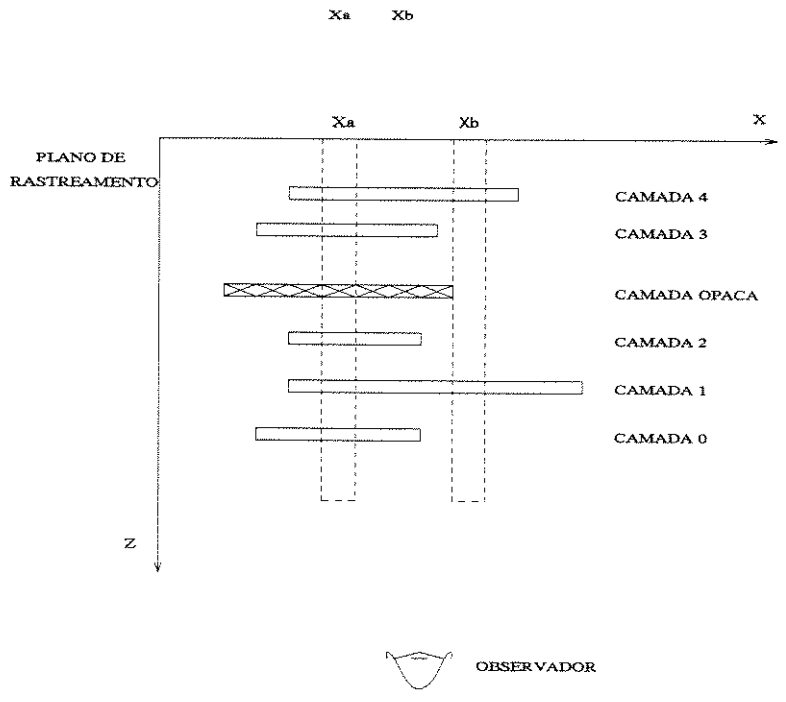

Figura 4.5: Camadas transparentes.

Após o armazenamento de todas as camadas de uma linha de rastreamento, parte-se para o cálculo da intensidade final de cada *pixel*. Para isto necessita-se das seguintes informações:

- Intensidade da última camada (camada opaca ou fundo).
- Intensidades das camadas transparentes ordenadas em z.
- Fator de transparência de cada camada.

Para este cálculo, aplica-se uma função linear, chamada de função de Blending (mistura), que usa como parâmetros os coeficientes de Blending que por sua vez são obtidos a partir dos fatores de transparência de cada camada.

Esta função é responsável pela "mistura" das intensidades, simulando o processo de filtragem da intensidade em cada camada transparente que o raio de luz atravessa.

O fator determina o quão transparente é o objeto, modelando algumas características dos materiais e geralmente varia entre 0 e 1.

A função de Blending opera entre duas camadas da seguinte maneira:

$$I_n = \alpha_n \times R_n + \beta_n \times I_{n-1} \tag{4.4}$$

onde,

- I_n é a intensidade total na camada n, correspondendo ao pixel que está sendo sucessivamente refinado.
- R_n é a intensidade da nova camada a ser incluída (camada n na posição p_n).
- I_{n-1} é a intensidade total das camadas anteriores.
- α e β são os coeficientes de Blending.

É desejável manter o valor da intensidade normalizado, ou seja, no mesmo limite numérico a cada passo, isto implica em impor:

$$\alpha_n + \beta_n = 1 \qquad com \qquad \beta_0 = 0 \tag{4.5}$$

O valor da intensidade também pode ser escrito como:

$$I = I_N = \sum_{n=0}^{N} (F_n \times R_n) \tag{4.6}$$

onde F_n é o coeficiente de filtragem da camada n,

Os coeficientes de filtragem também devem ser normalizados:

$$\sum_{n=0}^{N} F_n = 1 \tag{4.7}$$

A partir da equação 4.4, obtém-se o refinamento sucessivo para o pixel a cada camada, sendo I_k o valor da intensidade total na camada k:

$I_0 = \alpha_0 \times R_0 \qquad (\ \beta_0 = 0\ )$

$I_1 = \alpha_1 \times R_1 + \beta_1 \times I_0 = \alpha_1 \times R_1 + \beta_1 \times \alpha_0 \times R_0$

$I_2 = \alpha_2 \times R_2 + \beta_2 \times I_1 = \alpha_2 \times R_2 + \beta_2 \times \alpha_1 \times R_1 + \beta_2 \times \beta_1 \times \alpha_0 \times R_0$

generalizando, para camada n:

$$I_n = \alpha_n \times R_n + \beta_n \times \alpha_{n-1} \times R_{n-1} + \beta_n \times \beta_{n-1} \times \alpha_{n-2} \times R_{n-2} + \cdots + \beta_n \times \beta_{n-1} \times \beta_{n-2} \times \cdots \times \alpha_0 \times R_0 \tag{4.8}$$

Igualando-se as equações 4.6 e 4.8, obtém-se os coeficientes de filtragem (F_i) em função de α_n e β_n:

$F_n = \alpha_n$

$F_{n-1} = \beta_n \times \alpha_{n-1}$

$F_{n-2} = \beta_n \times \beta_{n-1} \times \alpha_{n-2}$

$\vdots \qquad \vdots$

$F_0 = \beta_n \times \beta_{n-1} \times \alpha_{n-2} \times \cdots \times \beta_0 \times \alpha_0$

ou seja, F_n multiplica R_n que é a intensidade da camada n, F_{n-1} multiplica R_{n-1} que é a intensidade da camada $n-1$ e assim sucessivamente. O somatório do resultado dessas multiplicações corresponde à intensidade final, ou seja, a intensidade do pixel refinado.

Freqüentemente ocorre que, para um dado objeto, pode-se ter mais de uma camada transparente armazenada. Um exemplo é uma esfera onde tem-se a parte da frente e a de trás. Caso não haja nenhuma camada opaca entre elas, são consideradas camadas distintas. A intensidade final é obtida aplicando-se a função de Blending à essas duas camadas.

4.5.3 ALGORITMOS NÃO-LINEARES

Uma outra estratégia na busca de mais realismo é considerar o fator de transparência como sendo função não-linear da superfície. Através da curvatura, simula-se a espessura. O fator de transparência é alterado de forma não-linear simulando ora mais opacidade, ora mais transparência.

Como a intensidade de luz transmitida é proporcional à espessura da camada que o raio atravessa, o fator de transparência deve ser determinado em cada ponto da camada.

Em uma superfície transparente, existem pontos onde o ângulo formado pela direção da normal e a direção do observador é próximo de zero, proporcionando assim, a passagem de mais intensidade de luz. Nestas regiões, a camada apresenta-se menos espessa ao observador, como mostra a figura 4.6.

Por outro lado, nos pontos onde estas direções são perpendiculares entre si, ou seja, o ângulo entre elas é próximo de 90^o, ocorre a passagem de menos intensidade. Nestas regiões, a camada transparente apresenta-se mais espessa ao observador, possuindo aparência opaca.

A figura 4.6 ilustra o problema da espessura da camada para um observador distante: No ponto $P1$, o vetor normal ($N1$) e a direção do observador (DO) estão próximos, logo a camada que o raio atravessa é pouco espessa (espessura $E1$). No ponto $P2$ a diferença entre as direções $N2$ e DO é mais acentuada que em $P1$, logo a camada que o raio atravessa é mais espessa ($E2$).

Isto é facilmente notado ao observar-se um copo de vidro transparente. As regiões próximas à borda lateral apresentam-se mais opacas.

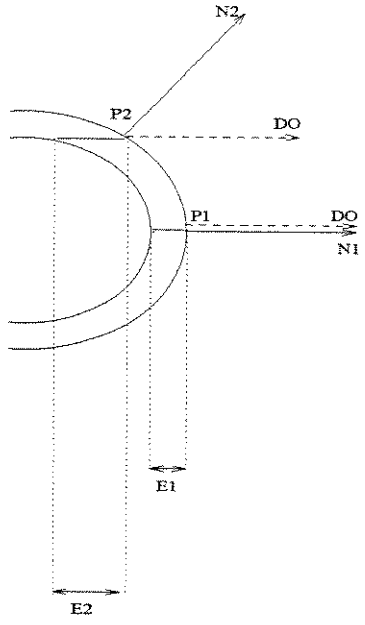

Figura 4.6: Espessura da camada transparente.

O cálculo do fator de transparência passa a ser feito em função não somente do coeficiente de transparência, como nos algoritmos lineares, mas também do cosseno do ângulo formado entre o vetor normal e a direção do observador. Pode-se controlar a influência do cosseno, elevando-o a um expoente que determina maior ou menor variação da transparência em função da curvatura da superfície.

Em [GK79] é apresentada uma solução para o cálculo deste fator através da seguinte equação:

$$\alpha = (MAXT - MINT) \times (1 - (1 - NORMZ)^{exp}) + MINT \tag{4.9}$$

onde,

- $MAXT$ é o valor máximo de transparência do objeto,
- $MINT$ é o valor mínimo de transparência do objeto,
- $NORMZ$ é a componente z (normalizada) do vetor normal à superfície no ponto,

- exp é o fator de influência do cosseno.

O expoente exp geralmente é 2 ou 3. No caso de uma esfera, por exemplo, estes valores produzem um efeito de transparência aproximadamente constante sobre a maior parte do objeto, reduzindo-se rapidamente perto das bordas, a medida que o ângulo entre a normal e a direção do observador aumenta.

Diferentes expoentes podem ser usados para simular a transparência de diferentes tipos de material. Por exemplo, vidros com pouca espessura podem ser obtidos aumentando-se o valor do expoente.

4.5.4 REFRAÇÃO

Uma outra maneira para obter-se mais realismo em uma imagem é simular a distorção natural dos objetos, que ocorre devido ao efeito de refração.

Este efeito ocorre quando um raio de luz atravessa meios de diferentes densidades, sofrendo mudança de direção. O efeito visual desta mudança é a aparência distorcida de objetos vistos através de superfícies transparentes. Esta distorção depende da quantidade e do tipo de material que o raio atravessa.

No caso de algoritmos *Scanline*, como o processamento é feito a cada linha da cena, não é possível considerar raios que atingem *pixels* localizados fora da linha corrente, impedindo assim, a simulação dos efeitos de refração.

Já os algoritmos *Ray-Tracing* simulam este efeito facilmente, pois baseiam-se no acompanhamento dos raios de luz, distorcendo-os naturalmente.

Em [GK79] é apresentado um método que simula a distorção da imagem devido ao efeito de refração, direcionando a sua associação ao algoritmo *Z-Buffer*.

4.5.5 COMENTÁRIOS FINAIS

Algoritmos de Lista de Prioridade e algoritmos *Scanline* (itens 2.5 e 2.7) podem ser facilmente adaptados para simular transparência.

O Z-Buffer apresenta algumas dificuldades pois os objetos são processados em ordem aleatória. Porém, se os objetos (camadas) forem ordenados em profundidade, torna-se possível simular a transparência.

Esta simulação é feita através do uso de algoritmos lineares e não-lineares. Ambos fornecem um considerável melhoramento da imagem. Os algoritmos lineares, por não considerarem a curvatura da superfície, são mais rápidos. Por sua vez, os algoritmos não-lineares, necessitam de mais tempo para o processamento, porém, produzem resultados melhores, principalmente em superfícies curvas aproximadas por um conjunto de polígonos planos.

De maneira geral, a compensação em termos de qualidade e a possibilidade de representar as características do material, são suficientes para justificar as alterações do algoritmo e o aumento do custo computacional.

4.6 SOMBRA

4.6.1 INTRODUÇÃO

Efeitos de sombra propiciam um grande aumento no realismo de uma cena. A simulação deste efeito facilita a compreensão das relações entre os objetos da cena e o melhoramento da percepção de profundidade.

Este item descreve um método que utiliza idéias dos trabalhos de Brotman [BB84], Crow [Cro77] e Blinn [Bli78b], direcionando-se para a incorporação no algoritmo *Scanline Z-Buffer.*

4.6.2 CONCEITO DE SOMBRA

O efeito de sombra é obtido quando os raios de luz emitidos de uma fonte encontram um obstáculo (objeto) que impede sua passagem, não atingindo assim, outros objetos ou partes de objetos, chamados de anteparo. Diz-se então, que estas regiões contém a sombra projetada ou ainda, estão em sombra (figura 4.7).

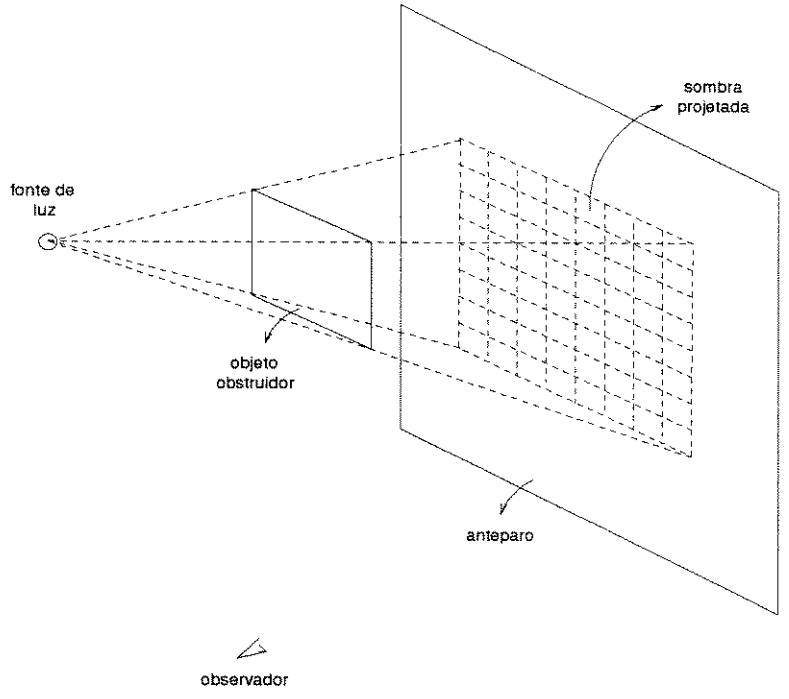

Figura 4.7: Sombra gerada por uma fonte de luz pontual.

A sombra é composta por duas partes: umbra e penumbra [Cro77]. A umbra é a região que não recebe nenhum raio de luz da fonte, também chamada de "sombra dura".

A penumbra recebe apenas alguns raios, sendo chamada de "sombra suave". Esta só existe no caso da fonte de luz ser uma fonte distribuída e é observada na região que rodeia a umbra.

A quantidade de raios de luz que atinge a penumbra, aumenta gradativamente ao distanciar-se da umbra, dando a aparência de suavidade ao contorno da sombra. A figura 4.8 ilustra este conceito.

A maior parte dos algoritmos concentram-se na determinação da umbra, porém ultimamente tem sidos desenvolvidos métodos que consideram os aspectos físicos do comportamento da luz e de recursos de síntese de imagens para a simulação da penumbra, fornecendo imagens com alto grau de realismo.

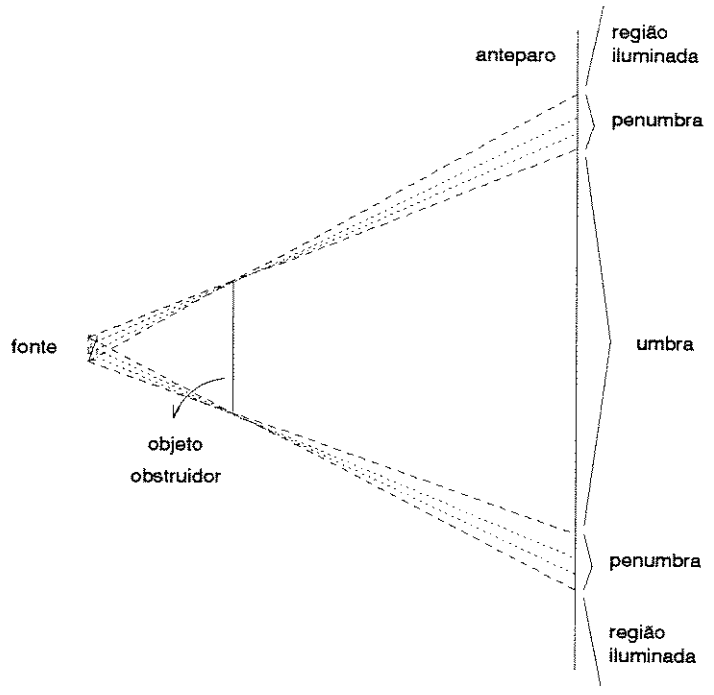

Figura 4.8: Partes da sombra: Umbra e Penumbra.

4.6.3 TIPOS DE SOMBRA

Para simular sombra, é necessário determinar as partes da cena que não estão recebendo os raios de luz e que são visíveis ao observador.

Os contornos da sombra estão em função do objeto obstruidor e da fonte, sendo obtidos através da projeção de cada polígono do objeto sobre o anteparo, determinando assim uma região de sombra.

A maneira de executar a projeção está relacionado com a geometria da fonte. Para as fontes de luz descritas no item 4.3.2, tem-se três tipos possíveis de formação de sombra:

- FONTE PONTUAL:

  As arestas de sombra (contornos) são definidas pelas interseções do plano determinado pelas arestas terminais do objeto obstruidor e pela posição da fonte, com o anteparo (projeção perspectiva). Qualquer ponto interno à região determinada por

estas arestas, está em sombra. A figura 4.7 contém uma sombra originária de uma fonte pontual.

- FONTE PONTUAL SPOT:

  Como a região iluminada por esta fonte é limitada, considera-se que a região de sombra não é determinada apenas pelos contornos do objeto obstruidor, mas também pelo limite de atuação dos raios de luz sobre o anteparo. A figura 4.9 mostra a sombra provocada por uma fonte *Spot* que ilumina parcialmente um objeto.

- FONTE DIRECIONAL:

  Como esta fonte emite raios paralelos que incidem igualmente sobre todos os objetos, os contornos da sombra são determinados através da projeção paralela do objeto obstruidor sobre outros objetos, feita na direção de atuação da fonte.

  Ao contrário das fontes pontuais, onde a sombra aumenta de tamanho a medida que o anteparo afasta-se da fonte (considerando-se fixos a fonte e o objeto obstruidor), a fonte direcional sempre gera sombras do mesmo tamanho, independentemente da distância da fonte ao anteparo. A figura 4.10 mostra uma sombra provocada por uma fonte direcional.

4.6.4 PROBLEMAS E SOLUÇÕES NA OBTENÇÃO DA SOMBRA

O primeiro passo é fazer uma análise da cena para cada fonte de luz. Isto pode ser feito tratando cada fonte de luz como se fosse um observador da cena. Como foi dito anteriormente, a partir da projeção de cada polígono, determinam-se os contornos da região de sombra, ou ainda, os polígonos de sombra.

Este procedimento é feito antes do cálculo da intensidade do ponto, que necessita saber quais fontes não estão atuando sobre o mesmo. É feito um pré-processamento a fim de determinar, a cada linha de rastreamento, quais os pontos em sombra e as respectivas fontes responsáveis. Depois de obtidas estas informações, parte-se para o cálculo da intensidade dos polígonos visíveis na linha, levando em consideração se os mesmos estão ou não em sombra.

O algoritmo *Scanline* apresenta um problema ao determinar a projeção de um polígono sobre um anteparo: Como a sombra muda de posição, em função da distância do

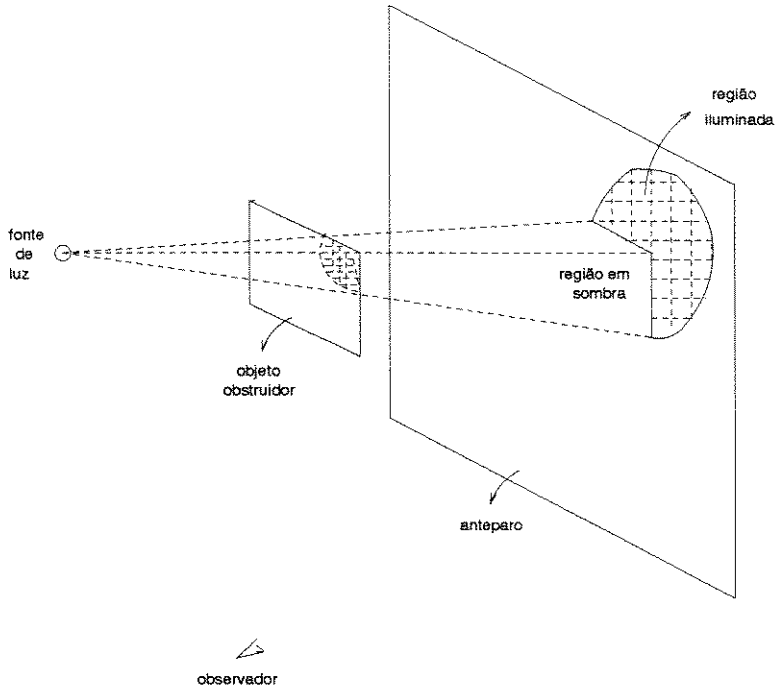

Figura 4.9: Sombra formada por uma fonte *Spot*.

anteparo ao observador localizado numa posição diferente da fonte, torna-se necessário o conhecimento desta distância, para determinar corretamente a sombra projetada.

No entanto esta distância só é conhecida após a transformação de câmera, onde a origem dos eixos passa a ser a posição do observador e sua direção de visão coincide com o eixo z (profundidade da cena). Portanto, somente ao determinar-se os polígonos visíveis é que obtém-se a informação de profundidade.

O objeto obstruidor (polígono) determina a região de sombra, ou seja, quaisquer objetos localizados atrás deste e na região de visão da fonte, não recebem raios de luz.

Precisa-se então de novas informações para poder determinar a sombra. Estas que usualmente seriam obtidas somente na fase de determinação da visibilidade, agora são necessárias para concretizar esta fase. Necessita-se então de saber previamente que polígonos ou parte destes estarão em sombra e qual a fonte associada.

Como foi dito anteriormente, a sombra é obtida quando os raios de luz encontram um obstáculo e não podem prosseguir. Pode-se entender a região do espaço determinada

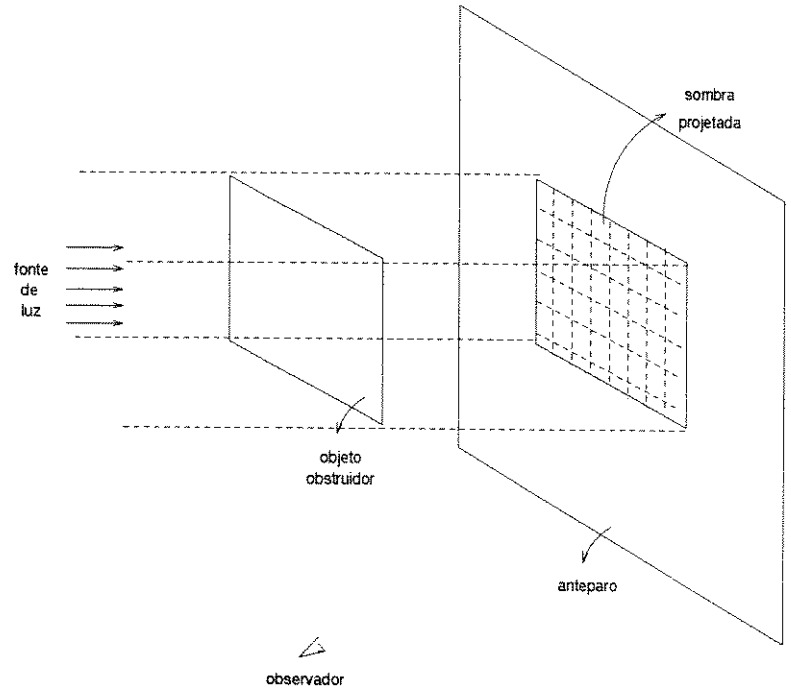

Figura 4.10: Sombra formada por uma fonte direcional.

pela fonte, contornos do objeto e anteparo como um volume de sombra [Cro77].

O efeito de sombra é percebido em qualquer parte dos objetos que são interceptados pelo volume de sombras. Se um objeto está totalmente dentro do volume de sombra, ele não recebe nenhuma luz da fonte associada. Se parte do objeto está dentro do volume, só esta estará em sombra.

É necessário então, conhecer o anteparo para obter-se o volume de sombra. Uma maneira de se fazer isto, seria comparar cada objeto com todos os outros, a partir da posição ou da direção da fonte para verificar onde os contornos do primeiro objeto estão sendo projetados. Este procedimento implica em um grande gasto computacional, pois envolve muitos cálculos comparativos além do custo de realizar a projeção sobre vários polígonos.

Uma maneira de contornar este problema é considerar como anteparo os planos que determinam o volume de visão do observador. Esta é uma solução muito mais econômica que a anterior. Porém, no caso de uma cena onde os objetos estão próximos uns dos outros

e distantes do plano de visão do observador ou do fundo da cena, seriam armazenados volumes de sombra de tamanho maior que o necessário, o que implicaria em um número de excessivo de cálculos e incrementos.

Apresenta-se aqui uma sugestão para limitar o volume de visão e consequentemente diminuir o tamanho dos volumes de sombra para cada objeto. Esta consiste em realizar um procedimento para obter o último objeto "visto" pela fonte e o vértice mais distante da fonte. Determina-se então um plano que passa por esse vértice e sua normal seja a direção de visão da fonte.

A partir da fonte e do objeto, determina-se a projeção dos contornos deste último sobre o plano determinado.

Este procedimento é interessante principalmente em objetos que seriam projetados no fundo da cena, que geralmente está a uma distância significativa dos objetos. Como estes planos foram determinados a partir de pontos extremos de objetos, pode-se considerá-los como os anteparos mais distantes para a sombra em questão. Assegura-se então, que as sombras provocadas por qualquer objeto da cena, estarão contidas na região limitada pelos referidos planos.

Estes volumes de sombra correspondem a novos objetos que são introduzidos na cena. Suas arestas são determinadas, ordenadas e tratadas como os outros objetos. No entanto, as únicas informações destas necessárias à determinação da sombra, são referentes à interseção com os objetos "reais" da cena, o índice do objeto interceptor e a fonte responsável pela sombra.

Algumas considerações são feitas a respeito deste método:

1. Objetos em sombra são aqueles que estão totalmente ou parcialmente dentro de um volume de sombra.
2. Objetos que, do ponto de vista do observador, estão atrás de um volume de sombra mas não se interceptam com o mesmo, não estão em sombra.
3. Objetos transparentes podem causar sombras (objeto interceptor) e receber sombras (anteparo). Se um objeto transparente estiver recebendo sombra, os objetos localizados atrás deste também estarão em sombra. Se um objeto transparente for o anteparo, ele causa uma sombra parcial, ou seja, filtra parte da luz.

4. Objetos que estão fora do volume de visão do observador não recebem sombras, no entanto podem provocar sombras.

5. Pode ser feita uma minimização no armazenamento das arestas da seguinte maneira: Considera-se apenas as arestas que fazem parte do contorno dos objetos, definidas a partir da posição da fonte (arestas silhuetas). Arestas interiores a estas regiões, ou seja, aquelas que são compartilhadas por mais de um polígono, devem ser desprezadas.

A figura 4.11 mostra o volume de sombra compreendido entre dois polígonos de sombra provocados por um polígono qualquer. Supõem-se não haver mais nenhum objeto atrás do anteparo. Tem-se:

- F é o polígono que determina o contorno (frente) do objeto de sombra,
- T é o contorno (traseiro) do objeto de sombra,
- $F1$, $F2$, $F3$ e $F4$ são vértices de F definidos por coordenadas x, y e z,
- $T1$, $T2$, $T3$ e $T4$ são os vértices de T, também definidos por coordenadas x, y e z,
- $d1$, $d2$, $d3$ e $d4$ correspondem às distâncias entre os vértices $F1$ e $T1$, $F2$ e $T2$,$F3$ e $T3$, $F4$ e $T4$, respectivamente. Definem as arestas das laterais do objeto de sombra e juntamente com as arestas dos polígonos F e T, definem os polígonos (faces) laterais.

Assim como é feito com os objetos da cena, deve-se armazenar as camadas de sombra por ordem de profundidade em relação ao observador.

A última consideração que utiliza apenas as arestas silhuetas para determinar a face, propicia uma boa economia, no entanto apresenta uma falha no seguinte caso: Como é apenas considerado o contorno de sombra do objeto, não é possível representar casos de sombra de objetos localizados dentro de (e causados por) objetos transparentes, onde os primeiros localizam-se antes da região determinada pelo contorno de sombra dos segundos objetos. Um bom exemplo é uma esfera, se seu contorno for considerado um polígono, objetos localizados na frente desse polígonos são considerados fora do volume de sombra.

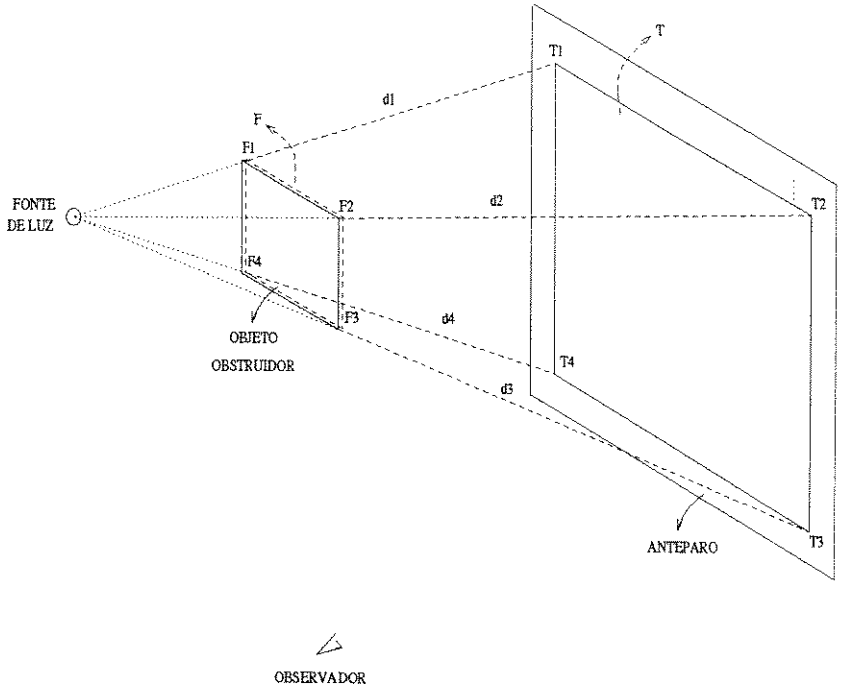

Figura 4.11: Volume de sombra delimitado por dois polígonos.

4.6.5 COMENTÁRIOS FINAIS

As arestas dos objetos de sombra devem ser armazenadas e ordenadas da mesma maneira que as arestas comuns. A coerência entre arestas e entre linhas também pode ser explorada.

A inclusão do efeito de sombra implica no aumento do número de operações aritméticas realizadas no algoritmo. No entanto, este número pode ser reduzido se não for feita a armazenagem todos os polígonos que determinam um volume de sombra, apenas a dos que determinam o contorno da sombra.

O aumento do realismo propiciado pela técnica de sombra compensa o trabalho de implementação e o tempo maior de processamento devido aos novos cálculos e armazenamentos.

4.7 ANTIALIASING

4.7.1 INTRODUÇÃO

Um problema existente na fase final de *rendering* é a representação da intensidade no dispositivo de visualização.

Um *pixel* pode corresponder a um conjunto infinito de pontos, porém, apenas um destes é tomado como referência para representá-lo e nem sempre sua intensidade, corresponde a intensidade real do conjunto. Em outras palavras, frente a necessidade de se fazer uma amostragem da função de intensidade, aparece inevitavelmente, o problema de *aliasing*. Entre *pixels* vizinhos que determinam fronteiras de objetos, a diferença entre as intensidades é bem acentuada, evidenciando o problema.

A figura 4.12 exemplifica este problema: O ponto de referência está localizado no centro do *pixel*. Caso este ponto esteja dentro da região delimitada pelo polígono (fig.4.12(a)), então sua intensidade total é considerada mesmo que parte do *pixel* esteja fora da região. A fig.4.12(b) mostra como é feito o preenchimento dos *pixels* dentro do dispositivo de visualização.

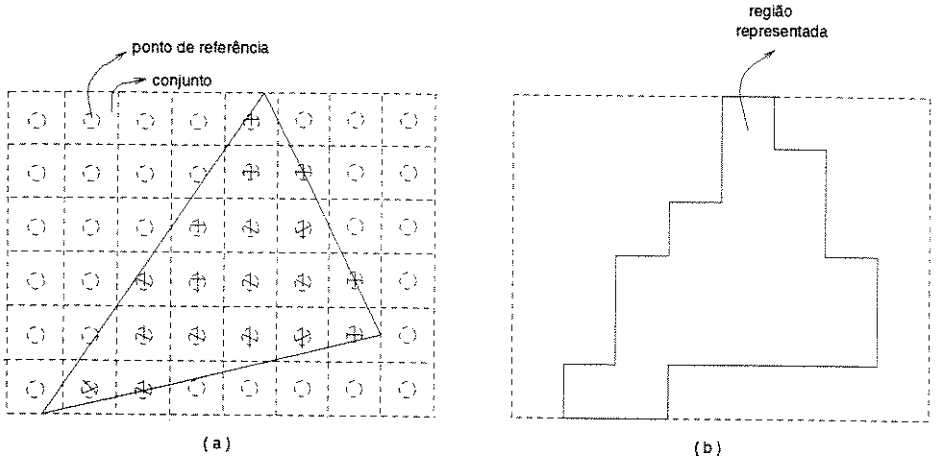

Figura 4.12: O problema de *aliasing*.

Para objetos muito pequenos, o problema de *aliasing* pode fazer com que estes "desapareçam" ao serem exibidos no dispositivo de visualização. Também pode ocorrer,

que estes mesmos objetos, sejam parcialmente exibidos, tendo sua forma original alterada.

O ideal seria que apenas partes dos *pixels* fossem preenchidas, no entanto, isto não é possível devido às limitações do *hardware*. Uma estratégia simples e interessante para solucionar este problema, é calcular a intensidade na proporção correspondente às partes do *pixel* cobertas pelo polígono.

Algumas sugestões são encontradas na literatura com o objetivo de contornar o problema de *aliasing* [FvD90], são os chamados métodos de *antialiasing*. O método apresentado aqui, baseia-se no algoritmo *A-Buffer* [Car84], podendo ser associado ao algoritmo *Scanline-Z-Buffer*.

4.7.2 DESCRIÇÃO DO MÉTODO

Uma maneira de calcular a intensidade de um *pixel*, em função das partes deste cobertas por um polígono, é através da determinação da área de abrangência do polígono no *pixel*.

Como o cálculo exato desta área causaria um grande aumento no custo computacional, parte-se então, para a obtenção de um valor aproximado.

Toma-se a intensidade do *pixel*, como sendo proporcional à área do mesmo coberta pelo polígono. Este procedimento contorna o problema de *aliasing*, pois assegura que não haverá mudanças bruscas da intensidade principalmente em objetos que possuem contornos inclinados (nem horizontal e nem vertical). A intensidade passa a variar gradativamente, proporcionando contornos mais suaves com aparência de continuidade.

No entanto, pode ser que partes de outros polígonos estejam localizadas neste *pixel*, além da possibilidade de haver polígonos transparentes sobrepondo-se. Tem-se então, o problema da determinação da visibilidade na região determinada por um *pixel*.

A solução é dividir o *pixel* em regiões chamadas de *subpixels* e para cada uma destas, determina-se quais são as partes visíveis.

Executando o procedimento acima, ou seja, considerando a intensidade do *pixel* proporcional à área abrangida pelo polígono, obtém-se um efeito mais realístico nas bordas

dos objetos. Este procedimento assemelha-se ao algoritmo *Z-Buffer* para determinar a visibilidade. É necessário, porém, fazer algumas considerações e executar alguns passos antes de iniciar a parte principal deste algoritmo, como descreve-se a seguir:

1- Verificar se a aresta que corta o *pixel* é terminal, ou seja, se determina fronteira com outros objetos ou o fundo da cena.

2- A região de recorte, correspondente ao contorno do *pixel*, é determinada em função das coordenadas x e y do ponto P de início ou de término da aresta.

Para qualquer ponto P, cujas coordenadas p_x, p_y e p_z pertencem ao conjunto dos números reais, a correspondente região do *pixel* será qualquer valor de x e y localizados, respectivamente, entre:

$$int(x_p) \leq x < int(x_p) + 1 \qquad e$$

$$int(y_p) \leq y < int(y_p) + 1$$

onde, $int(n)$ é a parte inteira do número n e a intensidade desta área, é representada na tela na posição $(int(x_p), int(y_p))$.

3- O número de subdivisões fica a critério do usuário. Quanto mais subdivisões forem feitas no *pixel*, mais preciso será o valor da intensidade, implicando porém, em maior custo computacional.

Supõe-se que a área do *pixel* é igual a 1. Este será subvidido em k regiões de área $1/k$, com "resolução" de $m \times n$, onde $k = m \times n$. A figura 4.13 mostra a subdivisão de um *pixel* em 20 *subpixels* (4×5).

4- A estrutura que contém informações sobre as arestas (LAA), fornece os valores de incremento das mesmas em x e z para cada linha de rastreamento, ou seja, ΔX e ΔZ para $\Delta Y = 1$. Como dentro do *pixel* tem-se m "sub-linhas" de rastreamento e n "sub-colunas", os valores de incremento dos mesmos são, respectivamente, ΔX_p e ΔZ_p, obtidos através do seguinte cálculo:

$$\Delta X_p = \frac{\Delta X}{m}$$

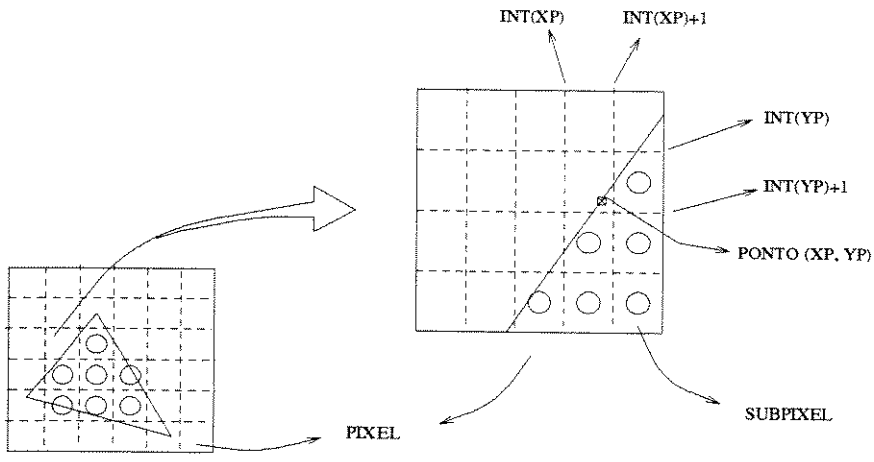

Figura 4.13: Subdivisão de um *pixel.*

$$\begin{aligned} \Delta Z_p &= \frac{\Delta Z}{m} \\ \Delta Y_p &= \frac{\Delta Y}{m} \end{aligned} \tag{4.10}$$

Para uma determinada "sub-linha", calcula-se os incrementos entre *subpixels* inicial (xi, zi) e final (xf, zf), chamados de ΔX_{pp} e ΔZ_{pp} ao longo dos eixos x e z, respectivamente. O número de *subpixels* a serem considerados em uma linha, chamado de *nspl*, determina o ΔZ_{pp}, como mostra as seguintes equações:

$$\begin{aligned} \Delta X_{pp} &= \frac{1}{n} \\ nspl &= \frac{xf - xi}{\Delta X_{pp}} \\ \Delta Z_{pp} &= \frac{zf - zi}{nspl} \end{aligned} \tag{4.11}$$

5- Calcular os valores das arestas direita e esquerda, entre as quais será feita a determinação da visibilidade. No caso de ter apenas uma aresta cortando o *pixel*, o limite direito ou esquerdo de análise será determinado pela borda vertical que representa o con-

torno do *pixel*.

Após a execução dos passos descritos acima, parte-se para a parte principal:

Para cada polígono dentro do *pixel*:

Para todos os seus *subpixels* (x, y) (usando as equações 4.10 e 4.11):

Se $Z(x, y) > Z_{armazenado}$, então:

Atualizar o índice do polígono.

Armazenar o valor de $Z(x, y)$.

Através dos passos descritos acima, determina-se o estado de visibilidade de cada *subpixel*. No caso de haver polígonos transparentes, os subpixels são ordenados em profundidade por camadas e o mesmo tratamento, descrito no item 4.5.2, pode ser aplicado.

O valor da intensidade final do *pixel* será determinado em função do resultado do procedimento descrito. Para cada polígono, calcula-se a área coberta pelo mesmo no *pixel*, através do seguinte procedimento:

Para cada polígono que intercepta o *pixel*:

Determinar o número de *subpixels* (ns) ocupados por ele.

Calcular a área: Área $= \frac{ns}{k}$

Determinar a intensidade final do *pixel*:

Intensidade =

Área do polígono 1 × Intensidade do polígono 1 +

Área do polígono 2 × Intensidade do polígono 2 + ...

$\vdots$ $\vdots$

Área do polígono n × Intensidade do polígono n

4.7.3 COMENTÁRIOS FINAIS

A introdução das técnicas de *antialiasing* é muito importante para o aumento do realismo de uma imagem. Os melhoramentos são percebidos principalmente no contorno de objetos que não seja nem vertical e nem horizontal, pois nestes casos, o problema de *aliasing* é bem evidente.

Objetos pequenos e seqüências de animação também são recompensados pela associação de técnicas de *antialiasing*, que permitem a correta visualização dos mesmos.

Além da técnica apresentada, onde aumenta-se a taxa de amostragem de um *pixel*; pode-se utilizar técnicas de pré-filtragem, onde o *pixel* é tratado como uma área finita [FvD90]. Em [TM87] é apresentado um resumo de vários algoritmos de *antialiasing* que podem ser associados aos algoritmos para a remoção de superfícies escondidas.

Como conseqüência da associação desta técnica, ocorre um aumento do tempo de processamento da cena. Porém, isto é recompensado pelos melhoramentos das imagens, que passam a ter contornos mais suaves e representam corretamente os objetos.

4.8 COMENTÁRIOS FINAIS

A associação de técnicas para a simulação de fontes de luz, efeitos de tonalização, transparência, sombra e *antialiasing*, provocam um aumento satisfatório da qualidade da imagem, compensando o acréscimo no tempo de processamento e utilização de memória. Porém esse tempo continua sendo bem inferior se o *Scanline* for comparado com outros algoritmos como por exemplo o *Ray-Tracing*.

É importante ressaltar, que estas técnicas simulam os efeitos de modelos globais, sendo portanto, uma aproximação para estes. Isto justifica a rapidez do algoritmo, uma vez que a representação de um modelo global, envolveria cálculos mais complexos, que por sua vez, implicaria em maior custo computacional.

Capítulo 5

IMPLEMENTAÇÃO E RESULTADOS OBTIDOS

5.1 INTRODUÇÃO

Este capítulo tem como objetivo descrever a implementação do módulo *Scanline* no contexto do Projeto ProSIm.

Apresenta-se inicialmente, uma visão geral deste projeto para melhor compreensão, por parte do leitor, dos objetivos que motivaram o seu desenvolvimento. Em seguida descreve-se a implementação do algoritmo *Scanline*, enfatizando a associação das técnicas descritas no capítulo 4. Finalizando, alguns resultados são apresentados, bem como a análise do desempenho do algoritmo.

5.2 O PROJETO ProSIm

O projeto - Prototipação e Síntese de Imagens Foto-Realistas e Animação - tem por objetivo fornecer uma ferramenta para modelagem, visualização e animação de imagens de alta qualidade. Este sistema pode ser aplicado em atividades didáticas, de pesquisa e apoio à indústria [BMM89].

A figura 5.1 mostra o esquema geral do ProSIm. Este constitui um ambiente

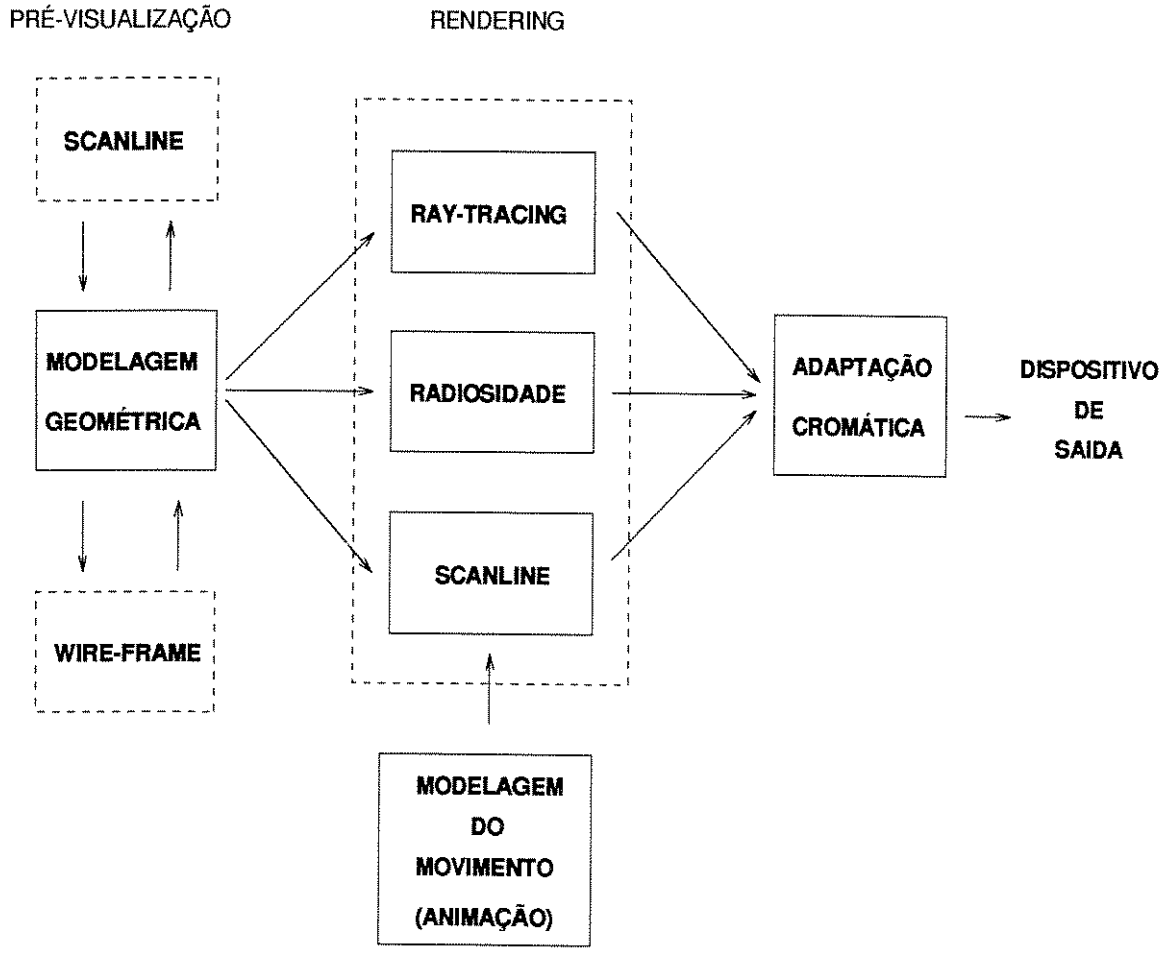

Figura 5.1: Esquema geral do Projeto ProSIm.

onde têm-se diversos módulos, destinados às seguintes tarefas:

- Modelagem geométrica,
- Animação,
- *Rendering*: *Ray-Tracing*, *Scanline* e Radiosidade,
- Pré-visualização (*Scanline*),
- Adaptação Cromática.

O ambiente de desenvolvimento possui uma biblioteca de funções que podem ser utilizadas por todos os módulos. A interação com o usuário é feita atualmente, por uma interface definida por procedimentos.

Todas as funções e módulos foram implementados na Linguagem C e no ambiente UNIX, podendo ser transportados para o ambiente DOS.

O sistema *Scanline*, como pode ser visto na figura 5.1, destina-se a realizar duas tarefas básicas no ProSIm:

- Servir de módulo pré-visualizador, ou seja, fornecer uma visualização inicial ao usuário, para que este verifique se a imagem está de acordo com o que idealizou. A visualização final, neste caso, será feita através de outros módulos.
- Executar o *rendering* final da imagem. Com este objetivo, foram associadas as técnicas descritas no capítulo anterior, proporcionando a geração de imagens de boa qualidade. O módulo de animação do ProSIm atualmente está utilizando o Sistema *Scanline* para gerar os quadros das seqüências.

5.3 O SISTEMA SCANLINE

A versão inicial do *Scanline*-ProSIm [HM90] foi alterada com a implementação das técnicas que permitem a obtenção dos seguintes efeitos:

- fontes de luz pontual (lâmpada e *spot*) e direcional,

ESQUEMA GERAL DO MÓDULO SCANLINE

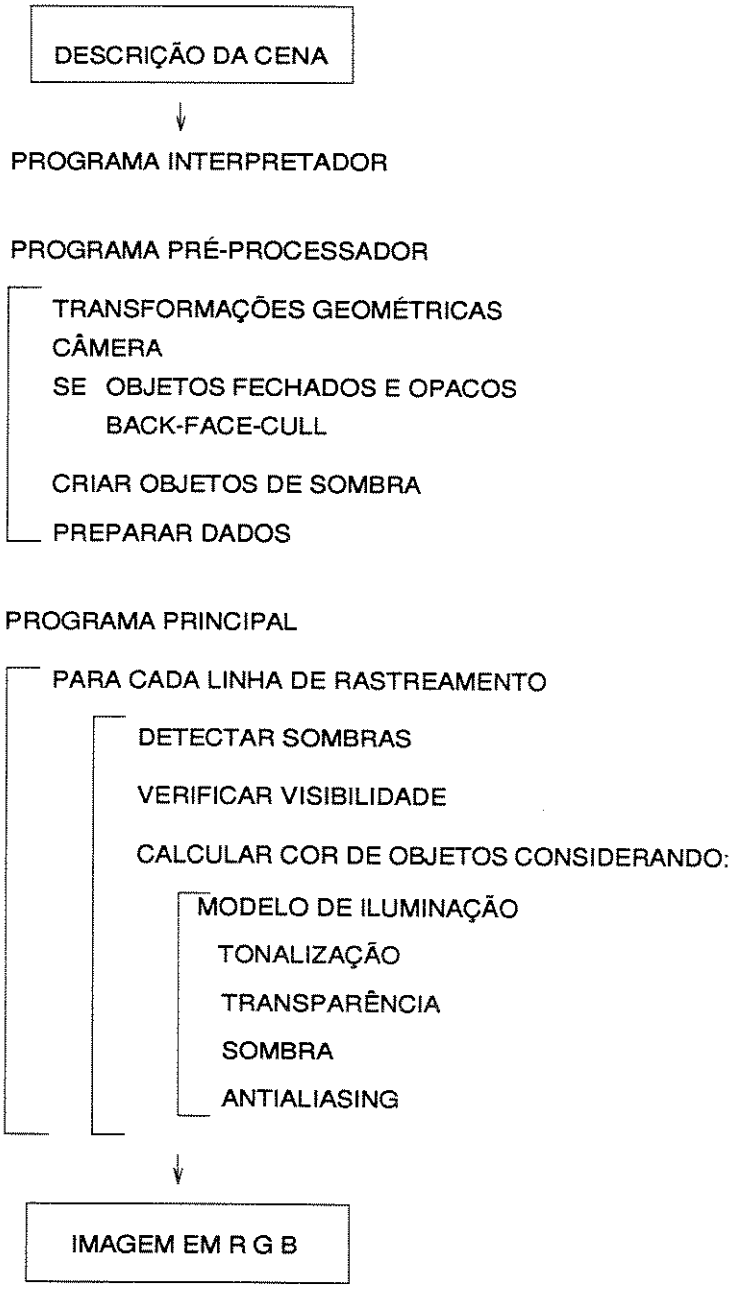

Figura 5.2: Esquema geral do Scanline.

- tonalização de Bouknight, Gouraud e Phong,
- transparência.

Para a construção deste sistema foi necessário a implementação de várias rotinas. Estas estão interligadas entre si, ou seja, rotinas que chamam outras rotinas. A figura 5.2 mostra as etapas principais do algoritmo *Scanline* implementado, onde aquelas que utilizam as técnicas descritas no capítulo 4, serão detalhadas em pseudo-código no item 5.3.2 segundo sua seqüência de acesso.

O item 5.3.1 descreve sucintamente algumas estruturas de dados utilizadas na implementação, a fim de facilitar o entendimento das rotinas descritas no item 5.3.2.

A técnica de sombra gerada por fonte de luz pontual está em fase de implementação, alguns procedimentos que já foram elaborados são descritos neste trabalho, no entanto, os resultados não estão incluídos aqui. A técnica de *antialiasing*, embora esteja especificada, não foi implementada. A sua introdução está sendo sugerida na figura 5.2 para uma possível extensão, em uma etapa posterior a este trabalho.

5.3.1 ESTRUTURAS DE DADOS

Este item descreve algumas estruturas de dados utilizadas na implementação do *Scanline*-ProSIm, além das estruturas LAA e Y-BUCKET descritas no item 3.3, principalmente as que estão diretamente envolvidas na associação das técnicas citadas. As demais estruturas de dados utilizadas neste algoritmo estão descritas em [HM90] e [PM91].

- Estrutura SCANIN:

  Destina-se a armazenar as características dos objetos. Possui cinco variáveis agregadas, onde todas têm dimensão igual ao número de objetos:

  1. Variável "geo" do tipo "Polyedron" que define as características geométricas dos objetos, sendo composta das seguinte variáveis:
     (a) npoints: tipo inteiro que representa o número de pontos do objeto,
     (b) nfaces: tipo inteiro que representa o número de polígonos do objeto,

(c) *p_list: tipo ponto que contém a lista dos pontos do objeto de dimensão "npoints",

Obs.: O tipo "ponto" é definido por três números reais referentes às coordenadas x, y e z, respectivamente.

(d) *f_list: tipo "Face" de dimensão "nfaces" que contém as seguintes variáveis:

- nvert: tipo inteiro que contém o número de pontos (vértices) de cada polígono,
- *v_list: tipo inteiro que contém a lista dos índices dos vértices para cada polígonos, cuja dimensão é "nvert".

2. Variável "viz" do tipo "Surface" que define a cor do objeto, sendo composta das seguinte variáveis:

   (a) ka, kd, ks e kt: tipo "Color" que contém, respectivamente, as contribuições ambiente, difusa, especular e o coeficiente de transparência,

   Obs.: O tipo "Color" é definido por três números reais referentes às componentes R, G e B, respectivamente.

   (b) n: tipo real que contém o coeficiente que determina o tamanho do *highlight*.

3. Variável "shade" do tipo enumerável "ShadeType", que pode possuir os valores FLAT, GOUR ou PHONG, identificando o tipo de tonalização do objeto.

4. Variável "ftype" do tipo enumerável "Type", que pode possuir o valor OPEN ou CLOSE. Este valor determina se o objeto é aberto ou fechado.

- Estrutura LINEBUFFER:

Destina-se a armazenar as informações referentes aos *pixels* de uma linha de rastreamento, tendo dimensão igual ao valor x da resolução da imagem. Possui cinco variáveis agregadas:

1. n: campo inteiro que representa o número de camadas (polígonos) localizadas na posição do *pixel*,
2. *z: campo real que compreende uma lista de dimensão "n" contendo os valores da profundidade de cada camada,
3. *pnumber: campo inteiro que compreende a lista de dimensão "n" dos índices dos polígonos que correspondem às camadas,

4. *layer_int: campo "Color" que compreende a lista de dimensão "n" contendo a intensidade de cada camada,
5. intensity: campo "Color" que representa o valor da intensidade final do *pixel* em função de todas as camadas.

Obs.: A dimensão "n" das listas acima só será maior que um no caso de haver camadas transparentes.

- Estrutura LINESHBUFFER:

  Destina-se a armazenar as informações de sombras para cada *pixel* de uma linha de rastreamento, tendo dimensão igual ao valor x da resolução da imagem. Possui cinco variáveis agregadas, cujas dimensões correspondem ao número de fontes de luz:

  1. n: campo inteiro que representa o número de objetos (camadas) que causam sombra no *pixel*,
  2. *z: campo real que compreende uma lista de dimensão "n" contendo os valores da profundidade do objeto que causa sombra (importante para o cálculo de sombra proveniente de objetos transparentes),
  3. *objeto: campo inteiro que corresponde a lista de dimensão "n" dos índices dos objetos que causam as sombras,
  4. *face: campo inteiro que compreende a lista de dimensão "n" dos índices dos polígonos que causam as sombras.

- Estrutura REFINV:

  Esta estrutura é alocada apenas para os objetos que possuem o tipo de tonalização de Gouraud ou Phong. Possui as seguintes variáveis:

  1. *tam: campo inteiro que, para cada vértice do objeto, indica o número de faces associadas a ele.
  2. *inv_list: campo "Inv" que, para cada vértice do objeto, contém as seguintes variáveis:
     (a) *inv_face: campo inteiro que corresponde a uma lista de dimensão "tam" contendo o índice dos polígonos associados,

(b) *inv_norm: campo "ponto" que corresponde a uma lista de dimensão "tam" contendo a normal do vértice em relação à face associada.

- Variável DO_SHADOW:

  Variável booleana que pode ter valor TRUE ou FALSE, determinando, respectivamente, se as sombras dos objetos serão ou não calculadas.

5.3.2 ROTINAS

O algoritmo inicia-se com a leitura dos dados contidos em um arquivo de entrada elaborado pelo usuário. Este arquivo segue um formato padrão comum a todos os módulos do ProSIm.

Os objetos contidos no arquivo de entrada estão em sua forma primitiva, isto é, centrados na origem dos eixos (x,y,z) e com dimensões dentro do cubo unitário, sofrem posteriormente várias transformações geométricas. Estas englobam escalamento, translação e rotação e buscam colocar os objetos em uma posição determinada a critério do usuário.

A seguir os objetos passam por outra transformação que simula uma câmera. A partir de informações como posição do observador, centro de interesse da cena e outras, executa-se a projeção perspectiva ou paralela dos objetos, colocando-os em um novo sistema de coordenadas, chamado de sistema de coordenadas de visualização [Hou92].

Estes dois tipos de transformação são executados por funções contidas no ProSIm. A figura 5.4 ilustra a chamada aos procedimentos de leitura e transformações que antecedem a rotina principal do algoritmo.

A rotina principal, ilustrada na figura 5.3, executa inicialmente, em um pré-processamento dos objetos, para posteriormente concentrar a atenção do algoritmo em uma linha de rastreamento por vez, determinando sua visibilidade.

Os procedimentos da figura 5.3, assinalados com asteriscos (**), foram enfatizados neste trabalho pois são responsáveis pela execução das técnicas descritas no capítulo anterior, sendo detalhados separadamente em outras figuras.

A seguir, descreve-se sucintamente, a função principal de cada um destes procedimentos:

- CONVERT_OBJETOS - figura 5.5:

  Determina quais polígonos estão de frente para o observador e no caso de ser desejável o cálculo da sombra, quais estão de frente para a "fonte de luz". Armazena para os mesmos, informações relativas a eles, bem como das arestas que os constituem.

- CONVERT_SH_OBJETOS - figura 5.6:

  Determina para um objeto, quais são os polígonos que estão de frente para a fonte de luz. Armazena na estrutura de dados ScanIn informações relativas ao objeto de sombra obtido a partir do objeto iluminado pela fonte. Também armazena informações sobre suas arestas.

- CRIAR_OBJ_SHADOW - figura 5.7:

  Finaliza a criação do objeto de sombra iniciada em convert_sh_objetos. O valor da profundidade da sombra é obtido, assim as arestas e faces laterais (do ponto de vista da fonte) são criadas. Os dados são armazenados na estrutura ScanIn completando a descrição do objeto.

- SOFT - figura 5.8:

  Determina, para cada vértice visível de um objeto, cuja tonalização é Gouraud ou Phong, quais os polígonos associados a ele e suas respectivas normais no vértice. Estas informações são armazenadas na estrutura RefInv.

- EDGE_Y_SORT - figura 5.9:

  Responsável pelo cálculo de diversas informações sobre as arestas, além de posicioná-las na lista Y-BUCKET, referente ao seu início e também convertê-las para as coordenadas do dispositivo de visualização.

- EDGE_SH_Y_SORT - figura 5.10:

  Executa a determinação de informações sobre as arestas dos objetos de sombra, assim como no procedimento EDGE_Y_SORT. Posiciona as arestas na lista Y_BUCKET_SHADOW, armazenando as informações relativas à posição.

- FILL_SH_LINE - figura 5.11:

  Percorre a lista de arestas ativas de sombra e faz a chamada ao procedimento SHADOW, tendo como parâmetros, duas arestas consecutivas. Após isto, atualiza a posição das arestas para a próxima linha.

- SHADOW - figura 5.16:

  Verifica a presença de sombra para cada *pixel* da linha localizado entre as duas aresta. Armazena informações sobre cada "camada" de sombra na estrutura LineShBuffer.

- FILL_LINE - figura 5.12:

  Percorre a lista de arestas ativas e executa a chamada ao procedimento SHADE, cujos parâmetros são duas arestas consecutivas. Após isto, os valores relativos ao cálculo da intensidade e posição das arestas são atualizados para a próxima linha. Depois de percorrer toda a LAA, executa a chamada ao procedimento CALC_SHADOW para o caso em que a sombra é calculada e finalmente, chama o procedimento CALC_COLOR.

- SHADE - figura 5.14:

  Determina as camadas visíveis para cada *pixel* da linha entre duas arestas, armazenando suas intensidades e profundidades na estrutura LineShBuffer.

- ESC_SHADE - figura 5.13:

  Determina a intensidade de um *pixel* de acordo com o tipo de tonalização do objeto.

- ILUM_MODEL - figura 5.15:

  Executa a aplicação do modelo de iluminação para o cálculo da intensidade de um ponto, determinando para cada fonte de luz e armazenando, separadamente, suas componentes ambiente, difusa e especular.

- CALC_COLOR - figura 5.17:

  Determina a intensidade final de cada *pixel* em função da intensidade de todas camadas e seus fatores de transparência, através das funções de *Blending* (item 4.5.2).

- CALC_SHADOW - figura 5.18:

  Executa a verificação, para cada camada visível da linha, se a mesma está em sombra. Se estiver, despreza-se a contribuição da fonte de luz responsável pela sombra.

```
CONVERT_OBJETOS **
SOFT **
EDGE_Y_SORT**
SE  DO_SHADOW
    ENTÃO  PARA CADA FONTE DE LUZ
                EDGE_SH_Y_SORT **
PARA CADA LINHA DE RASTREAMENTO  (SCLN)
    INICIALIZAR LINHA DE OBJETOS
    SE  DO_SHADOW
        ENTÃO
        INICIALIZAR LINHA DE SOMBRA
        PARA CADA FONTE DE LUZ
            SE ( EXISTIREM ARESTAS DE SOMBRA INICIANDO-SE  NA SCLN
                - Y_BUCKET_SHADOW[SCLN]  DIFERENTE DE ZERO ) OU
               ( EXISTIREM ARESTAS DE SOMBRA TERMINANDO NA SCLN)
                ENTÃO  ATUALIZAR LISTA DE ARESTAS DE SOMBRA ATIVAS (LAAS)
            SE ( EXISTIREM ARESTAS DE SOMBRA INTERCEPTANDO A SCLN)
                ENTÃO  FILL_SH_LINE **
    SE ( EXISTIREM ARESTAS DE OBJETOS INICIANDO-SE NA SCLN
        - Y_BUCKET[SCLN]  DIFERENTE DE ZERO ) OU
       ( EXISTIREM ARESTAS DE OBJETOS TERMINANDO NA SCLN )
        ENTÃO  ATUALIZAR A LISTA DE ARESTAS (LAA)
    SE ( EXISTIREM ARESTAS INTERCEPTANDO A SCLN)
        ENTÃO  FILL_LINE **
    ARMAZENAR LINHA
```

Figura 5.3: Rotina Principal

ROTINA INICIALIZADORA

```
LEITURA DE DADOS
EXECUTAR TRANSFORMAÇÕES GEOMÉTRICAS SOBRE AS PRIMITIVAS
TRANSFORMAÇÃO DE CÂMERA NOS OBJETOS (VISTA DO OBSERVADOR)
SE  DO_SHADOW
    ENTÃO
        PARA CADA FONTE DE LUZ
            TRANSFORMAÇÃO DE CÂMERA (VISTA DA FONTE DE LUZ)
            CONVERT_SH_OBJETOS
            CRIAR_OBJ_SHADOW
        TRANSFORMAÇÃO DE CÂMERA  NOS OBJETOS DE SOMBRA
                                 (VISTA DO OBSERVADOR)
```

Figura 5.4: Rotina Inicializadora.

CONVERT_OBJECTS

```
PARA TODOS OBJETOS

    PARA TODAS AS FACES

        CALCULAR  NORMAL

        SE ( FACE DE FRENTE  AO OBSERVADOR )    OU    ( FACE TRANSPARENTE)
            OU ( TIPO_OBJETO  IGUAL A ABERTO )

            SE  ( FACE DENTRO DO DISPOSITIVO DE VISUALIZAÇÃO)
                ENTÃO

                    ARMAZENAR A FACE E INFORMAÇÕES SOBRE A MESMA

                    ARMAZENAR SUAS RESPECTIVAS ARESTAS
```

Figura 5.5: Preparação dos objetos.

```
CONVERT_SH_OBJETOS

PARA TODAS AS FACES
    CALCULAR NORMAL
    SE (FACE DE FRENTE PARA FONTE ) OU (FACE TRANSPARENTE ) OU
                                (TIPO OBJETO ABERTO)
        ENTÃO
            ARMAZENAR SUAS ARESTAS NA LISTA DE ARESTAS DO OBJETO
            ARMAZENAR DADOS DA FACE NA ESTRUTURA DE DADOS ScanIn (NOVO OBJETO)
```

Figura 5.6: Preparação dos objetos de sombra.

```
CRIAR_OBJ_SHADOW

PARA TODOS OS OBJETOS
    PARA TODAS AS FACES
        CALCULAR NORMAL
        SE (FACE DE FRENTE PARA FONTE ) OU (FACE TRANSPARENTE ) OU
                                (TIPO OBJETO ABERTO)
            ENTÃO
                OBTER PONTO DE MAIOR COORDENADA Z (PROFUNDIDADE)
                ARMAZENAR EM COORDENADAS DO MUNDO O PONTO

PARA CADA ARESTA SILHUETA DO OBJETO EM QUESTÃO
    PARA CADA PONTO
        OBTER ARESTA LATERAL (DIREÇÃO DA FONTE)
        ARMAZENAR DADOS NA ESTRUTURA ScanIn
                        (NOVO OBJETO DE SOMBRA ESTÁ DEFINIDO)
```

Figura 5.7: Criação dos objetos de sombra.

SOFT

```
PARA TODOS OS OBJETOS
    SE (TIPO DE TONALIZAÇÃO FOR DIFERENTE DE FLAT)
        ENTÃO
            PARA TODAS AS FACES
                PARA TODOS OS VÉRTICES
                    ARMAZENAR PARA O OBJETO O E VÉRTICE V
                        ÍNDICE DA FACE ASSOCIADA N :
                        INV_REF [ O ] [ V ]._FACE_ASSOCIADA [ N ] = ÍNDICE DA FACE
            PARA TODOS OS VÉRTICES
                PARA TODAS AS FACES ASSOCIADAS
                    SE ( FACE ( F1 ) VISÍVEL )
                        ENTÃO
                            NORMAL_DO_VÉRTICE = NORMAL DE F1
                            SUM = 1
                            PARA TODAS AS FACES ASSOCIADAS ( F2 )
                                (exeto F1 )
                                SE (ÂNGULO ENTRE A NORMAL DE F1 E A NORMAL DE F2
                                    FOR MENOR QUE 60 graus )
                                    ENTÃO
                                        NORMAL_DO_VÉRTICE = NORMAL_DO_VÉRTICE + NORMAL DE F2
                                        SUM = SUM + 1
                            NORMAL_DO_VÉRTICE = NORMAL_DO_VÉRTICE / SUM
                            ARMAZENAR PARA O OBJETO O E VÉRTICE V
                                NORMAL DA FACE ASSOCIADA F1
                                INV_REF [ O ] [ V ]._NORMAL_ASSOCIADA [ N ] = NORMAL_DO_VÉRTICE
```

Figura 5.8: Determinação dos polígonos associados aos vértices.

EDGE_Y_SORT

PARA TODOS OBJETOS

PARA CADA ARESTA

CONVERTER PARA COORDENADAS DO DISPOSITIVO DE VISUALIZAÇÃO

COLOCAR A ARESTA NA *Y_BUCKET* NA POSIÇÃO DE INÍCIO DA MESMA

CALCULAR E ARMAZENAR AS SEGUINTES INFORMAÇÕES

- posição **x** e **z**
- incremento da coordenada **x** entre linhas consecutivas
- incremento da coordenada **z** entre linhas consecutivas
- objeto de origem
- número da linha de rastreamento de término
- faces adjacentes

CASO O TIPO DE SHADING FOR

GOURAUD ENTÃO

CALCULAR E ARMAZENAR

- valor da intensidade no vértice superior
- incremento de intensidade da aresta entre linhas consecutivas

PHONG ENTÃO

CALCULAR E ARMAZENAR

- valor da normal no vértice superior
- incremento da normal da aresta entre linhas consecutivas

Figura 5.9: Cálculo de informações de arestas de objetos.

EDGE_SH_Y_SORT

PARA TODOS OS OBJETOS

PARA TODAS AS FONTES DE LUZ

PARA CADA ARESTA DE SOMBRA

CONVERTER PARA COORDENADAS DO DISPOSITIVO DE VISUALIZAÇÃO

COLOCAR A ARESTA NA *Y_BUCKET_SHADOW* NA POSIÇÃO DE INÍCIO DA MESMA

CALCULAR E ARMAZENAR AS SEGUINTES INFORMAÇÕES

- posição **x** e **z** de início da aresta
- incremento da coordenada **x** entre linhas consecutivas
- incremento da coordenada **z** entre linhas consecutivas
- objeto de origem
- número da linha de rastreamento de término
- incremento das coordenadas **x** e **y** entre a menor e a maior sombra

Figura 5.10: Cálculo de informações de arestas de sombra.

FILL_SH_LINE

PERCORRER A LISTA DE ARESTAS ATIVAS DE SOMBRA E
PARA CADA PAR DE ARESTAS CONSECUTIVAS

SHADOW (aresta esquerda, aresta direita) **

PARA CADA ARESTA

ATUALIZAR VALOR DOS PONTOS

Figura 5.11: Determinação de arestas de sombra em uma linha de rastreamento.

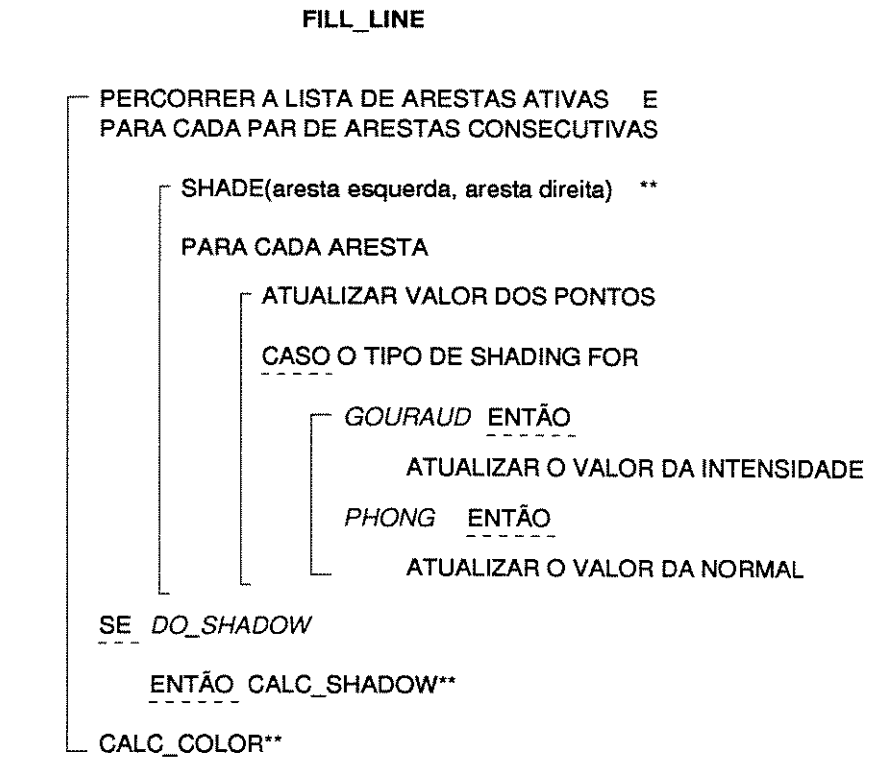

Figura 5.12: Determinação de arestas de objetos em uma linha de rastreamento.

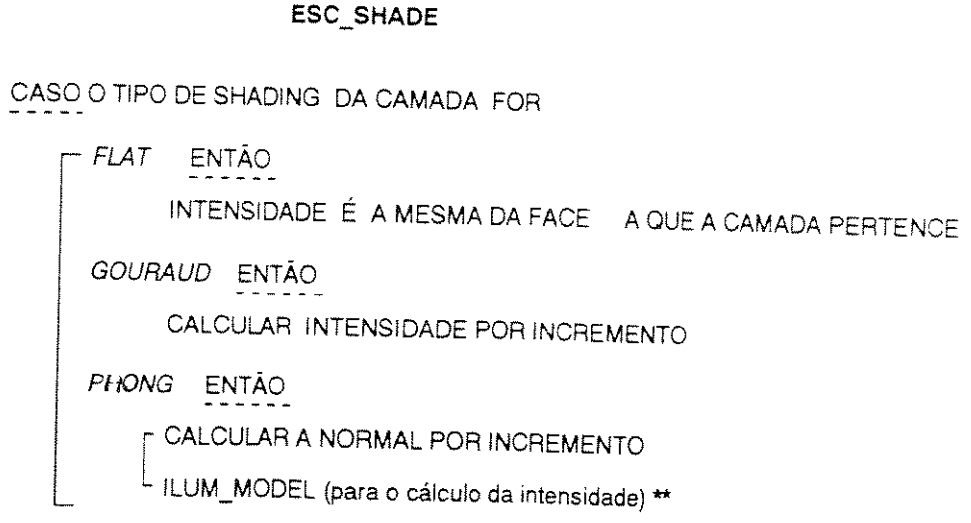

Figura 5.13: Determinação da intensidade em função do tipo de tonalização.

```
SHADE (ARESTA ESQUERDA , ARESTA DIREITA)

VERIFICAR QUAL É A FACE ENTRE AS ARESTAS

CASO O TIPO DE TONALIZAÇÃO FOR

    GOURAUD  ENTÃO

        CALCULAR INCREMENTO DE INTENSIDADE ENTRE AS ARESTAS

    PHONG  ENTÃO

        CALCULAR INCREMENTO DE NORMAL ENTRE AS ARESTAS

PARA TODOS OS PIXELS ENTRE AS DUAS ARESTAS    (da esquerda para a direita)

    SE  ( Z( x, y ) É MAIOR QUE  Z( x, y ) armazenado )

        ENTÃO

            SE  ( OBJETO TRANSPARENTE )

                ENTÃO     ARMAZENAR NA  MAIOR POSIÇÃO EM Z  (frente)    (ESTRUTURA LineBuffer)

                          ESC_SHADE**

                SENÃO     ARMAZENAR NA PRIMEIRA POSIÇÃO EM Z

                          ESC_SHADE**

                          ELIMINAR TODAS AS OUTRAS CAMADAS  (localizadas atrás)

        SENÃO

            EXECUTAR BUSCA EM Z DA POSIÇÃO  Zn ONDE SE INSERE A NOVA CAMADA

            SE  ( OBJETO TRANSPARENTE )

                ENTÃO     ARMAZENAR A CAMADA  NA POSIÇÃO Zn

                          ESC_SHADE**

                          PARA TODAS AS CAMADAS NA FRENTE DE Zn
                              AVANÇAR UMA POSIÇÃO

                SENÃO     ARMAZENAR NA PRIMEIRA POSIÇÃO EM Z

                          ESC_SHADE**

                          ELIMINAR TODAS AS OUTRAS CAMADAS  (localizadas atrás)
```

Figura 5.14: Determinação da visibilidade de cada *pixel*.

ILUM_MODEL

ARMAZENAR COMPONENTE AMBIENTE :
COMP_AMB = COMP_AMB + LUZ_AMB X KA

PARA CADA FONTE DE LUZ

CASO O TIPO DE FONTE FOR

SUN ENTÃO DIREÇÃO DA LUZ = DIREÇÃO DA FONTE

LAMP ENTÃO DIREÇÃO DA LUZ = PONTO - POSIÇÃO DA FONTE

SPOT ENTÃO DIREÇÃO DA LUZ = PONTO - POSIÇÃO DA FONTE

COS_ALFA = COSSENO ENTRE A NORNAL NO PONTO E A DIREÇÃO DA LUZ

SE A FACE FOR ILUMINADA (COS_ALFA < 0)

ENTÃO

CALCULAR A ATENUAÇÃO EM FUNÇÃO DA DISTÂNCIA

ARMAZENAR COMPONENTE DIFUSA :
COMP_DIF = COMP_DIF + LUZ_FONTE x ATENUAÇÃO x KD x COS_ALFA

DIREÇÃO DO OBSERVADOR = POSIÇÃO DO OBSERVADOR - PONTO

HALF_WAY = DIREÇÃO DA LUZ - DIREÇÃO DO OBSERVADOR

COS_TETA = COSSENO ENTRE A NORMAL E HALF_WAY

ARMAZENAR COMPONENTE ESPECULAR:
COMP_ESP = COMP_ESP + LUZ DA FONTE x ATENUAÇÃO x KS x $COS\_TETA^{func\_espa}$

COR FINAL = COMP_AMB + COMP_DIF + COMP_ESP

Figura 5.15: Modelo de iluminação.

```
SHADOW (ARESTA ESQUERDA , ARESTA DIREITA)

PARA TODOS OS PIXELS ENTRE AS DUAS ARESTAS   (da esquerda para a direita)
    PARA CADA FONTE DE LUZ
        SE FOR PRIMEIRA CAMADA DE SOMBRA DO PIXEL
        ENTÃO ARMAZENAR NA PRIMEIRA POSIÇÃO EM Z   (Estrutura LineBuffer)
        SENÃO
            SE ( Z( x, y ) É MAIOR QUE Z( x, y ) armazenado )
                ENTÃO
                    ARMAZENAR NA MAIOR POSIÇÃO EM Z (frente)
                SENÃO
                    EXECUTAR BUSCA EM Z DA POSIÇÃO Zn ONDE SE INSERE A NOVA CAMADA
                    ARMAZENAR A CAMADA NA POSIÇÃO Zn
                    PARA TODAS AS CAMADAS NA FRENTE DE Zn
                        AVANÇAR UMA POSIÇÃO
```

Figura 5.16: Determinação de sombra em cada *pixel*.

```
CALC_COLOR

PARA TODOS OS PIXELS DA LINHA DE RASTREAMENTO
    PARA CADA CAMADA
        CALCULAR O FATOR DE TRANSPARÊNCIA
        (utilizar funções de Blending descritas no capítulo 4)
    CALCULAR A INTENSIDADE FINAL EM FUNÇÃO DA INTENSIDADE DE CADA
    CAMADA E DE SEU FATOR DE TRANSPARÊNCIA
    ARMAZENAR A INTENSIDADE NA ESTRUTURA BUFF_LINE
```

Figura 5.17: Cálculo da intensidade "misturada".

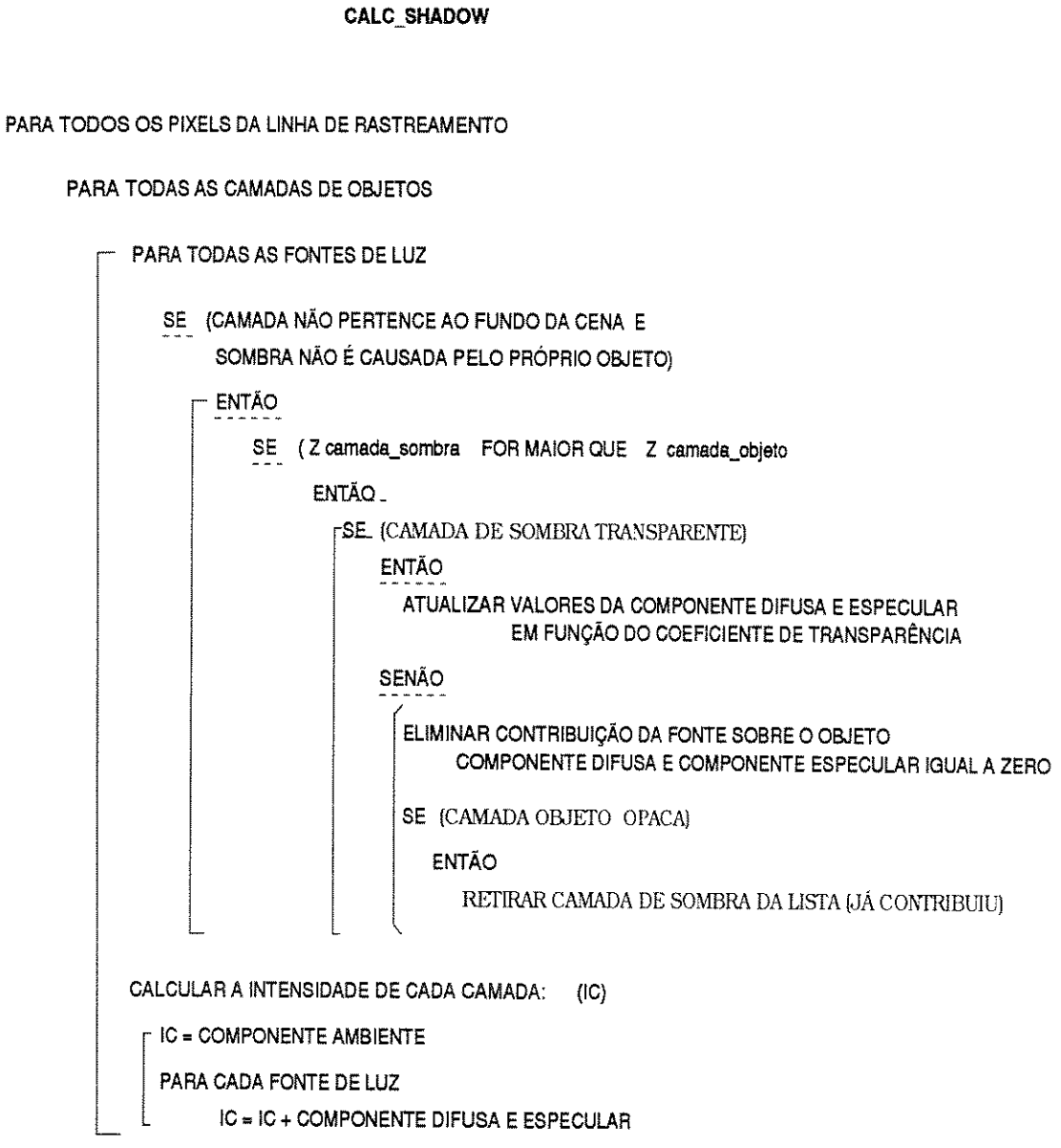

Figura 5.18: Cálculo das camadas em sombra.

5.4 RESULTADOS OBTIDOS

Este item destina-se a analisar os resultados obtidos com a implementação de algumas técnicas para melhorar o realismo, como descrito no item anterior.

O equipamento utilizado foi uma estação de trabalho SUN SPARC station 370 com placa gráfica GXP e capacidade para representar cores em 24 bits e um monitor colorido com resolução de 1152 X 900 *pixels.*

Alguns testes foram elaborados com o objetivo de analisar os seguintes aspectos:

1. Melhoramento visual da imagem perante as técnicas associadas.
2. Complexidade da cena, no que se refere a atributos visuais *versus* tempo de processamento.

Para analisar o primeiro aspecto, algumas imagens simples foram geradas através do sistema *Scanline*, ilustrando o resultado das técnicas associadas.

As imagens foram geradas com resolução de 900 X 900 *pixels* e as cenas contém as características descritas na tabela abaixo:

Cena	Objetos	n_o de pontos	n_o de faces	Atributos
1 fig. 5.19	3 esferas	684 (cada)	760 (cada)	Ton. de Bouknight Ton. de Gouraud Ton. de Phong
2 fig. 5.20	cilindro	40	22	Fonte Pontual Ton. Phong
3 fig. 5.21	cilindro	40	22	Fonte Direcional Ton. Phong
4 fig. 5.22	cilindro	40	22	Fonte *Spot* Ton. Phong
5 fig. 5.23	3 cubos	8 (cada)	6 (cada)	Transparência
	3 cilindros	40 (cada)	22 (cada)	Fonte Pontual

As imagens contidas nas cenas 1 a 5, foram analisadas em relação às seguintes características:

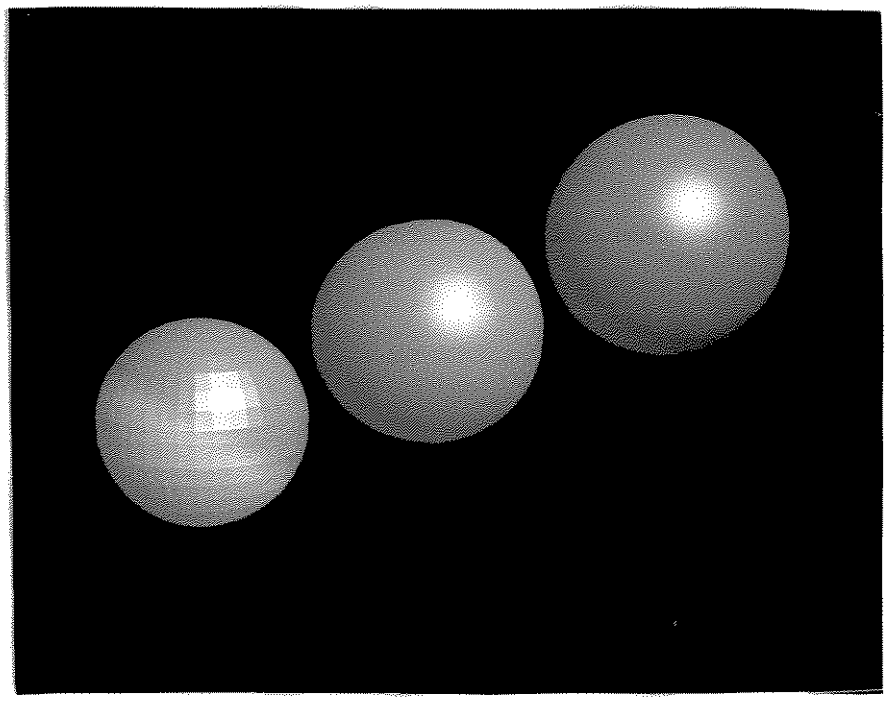

Figura 5.19: Cena 1: Tonalização de Bouknight, Gouraud e Phong.

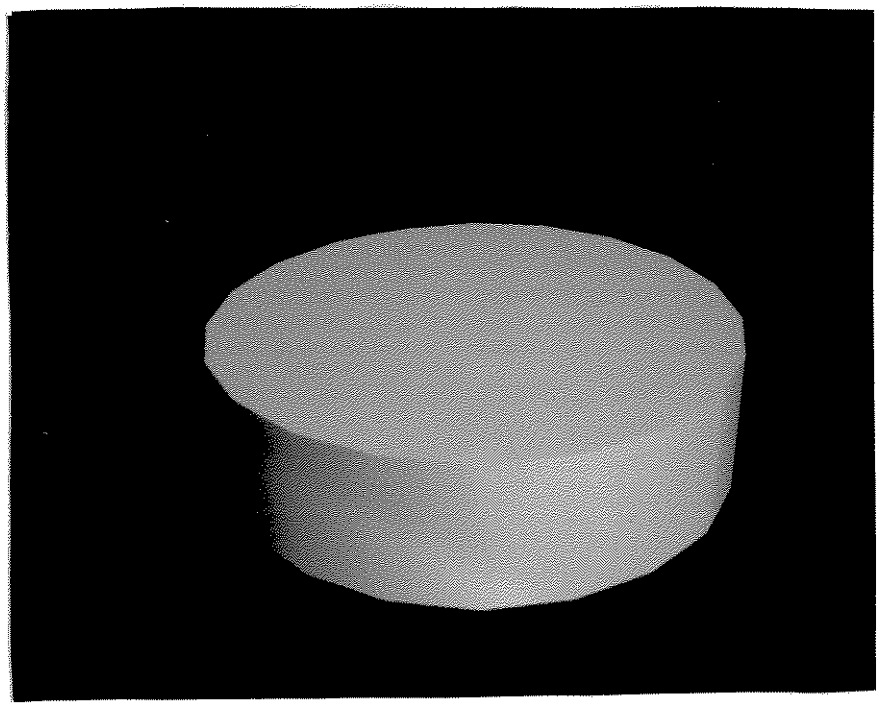

Figura 5.20: Cena 2: Fonte Pontual.

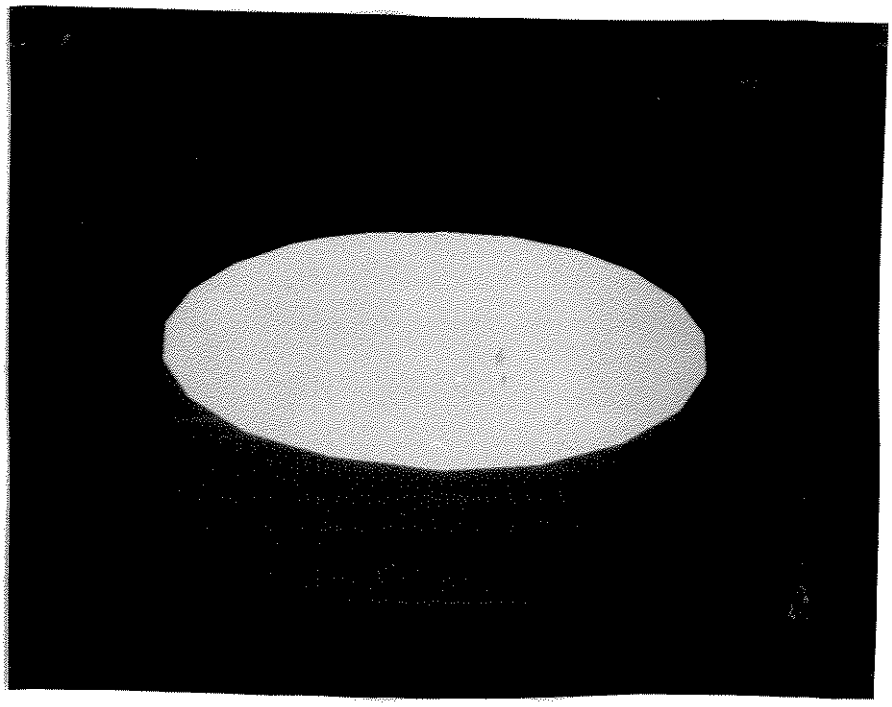

Figura 5.21: Cena 3: Fonte Direcional.

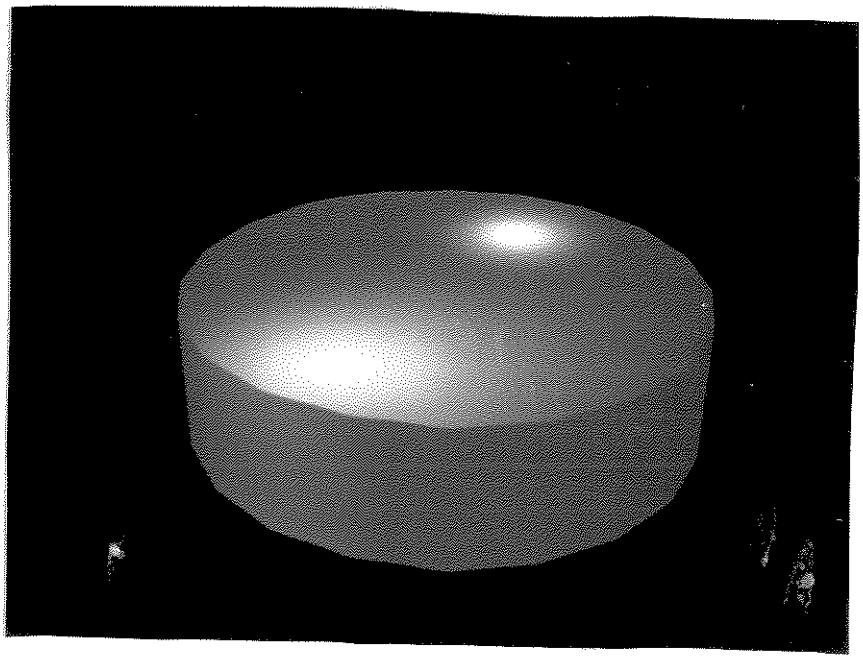

Figura 5.22: Cena 4: Fontes *Spot* com diferentes atuações dos raios de luz.

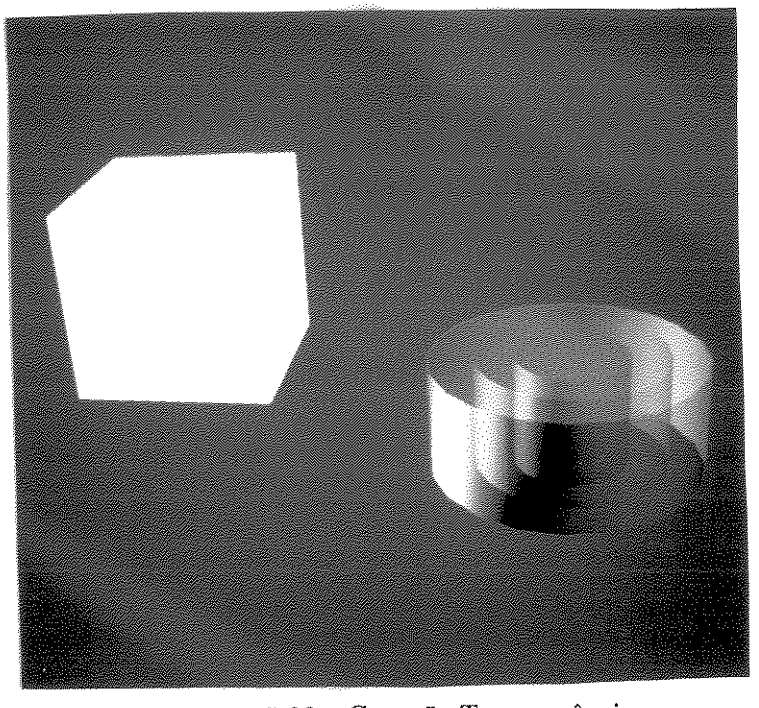

Figura 5.23: Cena 5: Transparência.

- TONALIZAÇÃO:

  A cena 1 (fig. 5.19) exibe a tonalização de três esferas (da esquerda para a direita), através dos modelos de Bouknight, de Gouraud e Phong respectivamente.

  Verifica-se que o modelo Flat não é adequado para tonalizar objetos que utilizam um conjunto de polígonos planos para aproximar superfícies curvas, como uma esfera. Isto já não ocorre em objetos onde deseja-se evidenciar as superfícies planas, como um cubo.

  Observa-se que os modelos de Gouraud e Phong são mais adequados para a aproximação de superfícies curvas, porém o segundo proporciona resultados melhores na representação das regiões que contém os pontos de brilho.

- FONTES DE LUZ:

  As cenas 2, 3 e 4 mostram os três tipos de fontes de luz implementados: pontual, direcional e *spot* usando baixo e alto índices de espalhamento; todas atuando sobre superfícies planas.

  Verifica-se que a fonte pontual (fig. 5.20), ilumina partes do objeto com maior ou

menor intensidade, em função de sua posição. A fonte direcional (fig. 5.21) ilumina igualmente todos os pontos de uma superfície, em função da posição destes em relação à direção da fonte.

A fonte *spot* (fig. 5.22), ilumina apenas uma determinada região do objeto. Observa-se a iluminação de duas fontes atuando sobre um cilindro, com alto e baixo índices de espalhamento, respectivamente.

- TRANSPARÊNCIA:

  A cena 5 (fig. 5.23) exibe alguns objetos transparentes. Do lado esquerdo da figura estão três cubos, onde o menor é opaco e os outros são transparentes. Do lado direito estão três cilindros, onde o menor é opaco e os maiores são transparentes.

Para analisar o segundo aspecto, ou seja, o tempo de processamento em função da complexidade da cena, foram feitos vários testes com diferentes atributos, definindo diferentes cenas. Estas foram geradas com resolução de 400 X 400 *pixels*, sendo compostas pelos seguintes objetos :

- Uma esfera formada por 760 faces e 1269 arestas,

Os seguintes atributos foram considerados como determinantes da complexidade da cena:

- Tonalização Flat,
- Tonalização Gouraud,
- Tonalização Phong,
- Número de fontes de luz
- Transparência,

Os resultados estão ilustrados na tabela abaixo, onde P, D e S referem-se a fontes de luz do tipo Pontual, Direcional e *Spot*, respectivamente.

Tonalização	Transparência	n_o de fontes	Tempo
Bouknight	não	1 P	14 seg.
	sim	1 P	24 seg.
Gouraud	não	1 P	16 seg.
	sim	1 P	26 seg.
Phong	não	1 P	19 seg.
	não	1 D	18 seg.
	não	1 S	20 seg.
	sim	1 P	34 seg.

Observa-se na tabela acima que o tempo de processamento varia muito pouco com o uso dos diferentes modelos de tonalização e também, de diferentes fontes de luz, sendo o modelo de Phong e a fonte do tipo *Spot* mais demorados que os outros, possuindo taxa de aumento pouco representativa.

O uso de diferentes fontes de luz também resulta em pouquíssimas variações do tempo de processamento.

Entre o processamento de objetos opacos e transparentes, a diferença é mais acentuada, ocorrendo um aumento aproximado entre 60% e 80% do tempo devido aos cálculos de transparência.

5.5 COMENTÁRIOS FINAIS

A implementação das técnicas para melhorar o realismo, descrita neste capítulo, não exigiu grandes alterações do algoritmo *Scanline* padrão, implementado em uma fase anterior a este trabalho [HM90]; baseando-se no acréscimo de novas rotinas.

As imagens obtidas apresentaram uma qualidade satisfatória, justificando a sua utilização como módulo de visualização final do Projeto ProSIm. Apesar do tempo de processamento variar muito em função dos atributos, aumentando significativamente em alguns casos, é considerado baixo em relação às imagens obtidas pelos módulos de *Ray-Tracing* e Radiosidade. Este fato, por sua vez, recomenda o uso deste sistema como módulo de pré-visualização, pois o mesmo possibilita uma antecipação do resultado final, com baixo custo computacional.

Como sugestão para a continuação deste trabalho, fica o término da implementação do processo de sombra, adicionando também os efeitos de sombra causados por fonte do tipo *Spot* e do tipo *Sun*. A implementação destes é semelhante e até mais simples que a da fonte do tipo Pontual, sugerida nos capítulo 4 e 5. Uma outra sugestão é a implementaão de *antialiasing*, que certamente produzirá bons resultados.

Capítulo 6

CONCLUSÃO

A popularização, cada vez maior, da utilização de recursos gráficos nas diversas áreas do conhecimento, vem exigindo o desenvolvimento de novas técnicas, bem como o aprimoramento das já existentes.

O algoritmo *Scanline* tem sido bem aceito e vem sendo usado freqüentemente, tanto em *softwares* desenvolvidos em ambientes universitários, como em *softwares* comerciais. Isto deve-se ao fato destes algoritmos serem relativamente rápidos e produzirem imagens com qualidade satisfatória para muitas aplicações.

Por não ser um algoritmo recente, diversas extensões já foram apresentadas, além do que, sua simplicidade facilita a associação de outros algoritmos destinados a melhorar a qualidade da imagem.

Este trabalho iniciou-se com um breve estudo dos algoritmos mais conhecidos para a remoção das superfícies escondidas, e em especial das variações dos algoritmos *Scanline*, com o objetivo de apresentar as características principais destes, bem como analisar os melhoramentos agregados aos mesmos.

Posteriormente, algumas técnicas foram estudadas para a obtenção de maior realismo, tais como:

- Simulação de fontes de luz,
- Tonalização,

- Transparência,
- Sombra,
- *Antialiasing.*

Com o objetivo de obter-se um sistema que fosse rápido, e apresentasse uma boa relação entre o tempo de processamento, os requerimentos de memória e a qualidade de imagens, optou-se por associar algumas destas técnicas a um algoritmo *Scanline Z-Buffer.*

Como exposto ao longo deste trabalho, a escolha do algoritmo *Scanline*, deveu-se às seguintes razões:

- facilidade para a manipulação e acesso às listas que armazenam informações sobre a geometria e características visuais dos objetos,
- possibilidade de explorar coerência entre faces, arestas e outros,
- exigir descrição simples da estrutura geométrica dos objetos,
- facilidade para a incorporação de novas técnicas.

A implementação realizada baseia-se na literatura de maneira geral e busca otimizar os cálculos e o armazenamento de informações, possuindo as seguintes características:

- Algoritmo *Scanline-Z-Buffer*,
- Tonalização de Bouknight, Gouraud e Phong,
- Fontes de luz dos tipos pontual, direcional e *spot*,
- Transparência,

O sistema *Scanline* é parte integrante do Projeto ProSIm e foi implementado na Linguagem C, estando disponível em um ambiente UNIX.

A associação das técnicas citadas acima, forneceram um considerável aumento da qualidade das imagens, sem contudo, causar um aumento exagerado do custo computacional para a obtenção das mesmas. Foram feitas implementações de novas rotinas e modificações em algumas rotinas já existentes, adaptando-as para incorporar os novos efeitos.

De maneira geral, os resultados obtidos foram satisfatórios e a contribuição principal deste trabalho foi no sentido de oferecer ao ProSIm, duas funcionalidades adicionais:

- Facilidades de pré-visualização,
- Facilidades de visualização final com baixo custo computacional.

As sugestões para trabalhos futuros, situam-se no aperfeiçoamento de alguns aspectos do algoritmo, através da incorporação dos seguintes efeitos:

- Sombra,
- *Antialiasing* (conforme proposto no item 4.7) e
- Texturas.

BIBLIOGRAFIA

[App68] A. Appel. Some techniques for shading machine rendering of solids. In *Spring Joint Computer Conference*. AFIPS, 1968.

[Ath83] P.R. Atherton. A scan-line hidden surface removal procedure for constructive solid geometry. *Computer Graphics ACM*, 17(3):73 – 82, 1983.

[BB84] L.S. Brotman and N.I. Badler. Generating soft shadows with a depth buffer algorithm. *IEEE Computer Graphics and Aplications*, (10):5 – 12, 1984.

[Bes92] S.G. Besvievsky. Cálculo da iluminação em síntese de imagens através de ray-tracing estocástico. Tese de Mestrado em Preparação., 12 1992.

[BK70] W.J. Bouknight and K. Kelley. An algorithm for producing half-toned computer graphics representations with shadows and movable light source. In *Spring Joint Computer Conference*. AFIPS, 1970.

[Bli78a] J. F. Blin. A scan-line algorithm for the display of parametrically defined surfaces. *Computer Graphics*, 12, 1978.

[Bli78b] J. F. Blinn. Me and my (fake) shadow. *IEEE Computer Graphics and Aplications*, 8(1):82 – 86, 1978.

[BMM89] C. Bannwart, H. Madeira, and L. P. Magalhães. Prosin - projeto em síntese de imagens. In *SIBGRAPI89*. SBC, 1989.

[Bou70] W. J. Bouknight. A procedural for generation of three-dimensional half-toned computer graphics representations. *Communications on ACM*, 13(9):527 – 536, 1970.

[BW86] G. Bishop and D.M. Weiner. Fast phong shading. *Computer Graphics ACM*, 20(4), 1986.

[Car84] L. Carpenter. The a-buffer, an antialiased hidden surface method. *Computer Graphics*, 18(3):103 – 108, 1984.

[Cat75] E. Catmull. Computer display of curved surfaces. *IEEE Conf. Computer Graphics, Pattern Recognition and Data Structures*, (5):11 – 17, 1975.

[Cla79] J.H. Clark. A fast scan-line algorithm for rendering parametric surfaces. *Computer Graphics*, 13, 1979.

[Coo86] R.L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51 – 72, 1986.

[Cro77] F. Crow. Shadow algorithms for computer graphics. *Computer Graphics*, 11(2):242 – 247, 1977.

[Cro84] G. Crocker. Invisibility coherence for faster scan-line hidden surface algorithms. *Computer Graphics*, 18(3):94 –102, 1984.

[Diz] J.A. Diz. Efeitos especiais em síntese de imagens. Tese de Mestrado em Preparação.

[Dre89] M. Dreux. *A Hybrid Scan-Line/Ray-Tracing Algorithm.* PhD thesis, Brunel University, February 1989.

[Duf79] T. Duff. Smoothy shaded rendering of polyhedral objects on raster display. *ACM on Computer Graphics*, 13(2):270 – 275, 1979.

[DW87] S. Dyer and S. Whitman. A vectorized scan-line z-buffer rendering algorithm. *IEEE Computer Graphics and Aplications*, (7):34 – 45, 1987.

[FKN80] H. Fuchs, Z.M. Kedem, and B.F. Naylor. A visible surface generation by a priori tree structures. In *SIGGRAPH'80*, 1980.

[FvD90] J. Foley and van Dam. *Computer Graphics Principles and Practice.* Addison Wesley, 1990.

[GK79] D. Greenberg and D. Kay. Transparency for computer synthesized images. *ACM Transactions on Graphics*, 4, 1979.

[Gou71] H. Gouraud. Continous shading of curved surfaces. *IEEE Transactions on Computers*, 20(6):623, 1971.

[Hea86] D. Hearn. *Computer Graphics.* Prentice-Hall, 1986.

[HG77] G. Hamlin and C.W. Gear. Raster-scan hidden surface algorithm tecniques. *Computer Graphics*, 11(2):206 – 213, 1977.

[HM90] M. Hounsell and L.P. Magalhães. Rt 5/90 - prosim-scanline, 1990.

[Hou92] M. Hounsell. Tookima: Um conjunto de ferramentas para descrição de animação modelada por computador. Master's thesis, Universidade Estadual de Campinas, 4 1992.

[Ing84] J.F.G. Ingram. *A linear growth image synthesis algorithm using object coherence.* PhD thesis, Brunel University, May 1984.

[Joy88] K. Joy. Tutorial 9- image synthesis. In *SIGGRAPH'88*, 1988.

[LC79] J.M. Lane and L.C. Carpenter. A generalized scan-line algorithm for computer display of parametrically defined surfaces. *Computer Graphics and Image Processing*, 11(3):290 – 297, 1979.

[Log90] L.A. Logatti. Linhas e superfícies escondidas: Toxonomia e proposta de implementação vlsi de um algoritmo z-buffer. Master's thesis, Universidade Estadual de Campinas, 4 1990.

[Mam89] A. Mammen. Transparency and antialiasing algorithms implemented with the virtual maps tecnique. *IEEE Transaction on Computer Graphics*, (5), 1989.

[Mye75] A.J. Myers. An efficient visible surface program, 1975.

[New74] M.E. Newell. *The utilization of procedural models in digital images synthesis.* PhD thesis, University of Utah, 1974.

[NNS72] M.E. Newell, R.G. Newell, and T.L. Sancha. A new approach to the shaded picture problem. In *Proceedings ACM National Conference*, 1972.

[NS79] W.N. Newman and R.F. Sproull. *Principles of Interactive Computer Graphics.* McGraw-Hill, 1979.

[Pho75] B.T. Phong. Ilumination for computer generated pictures. *Communications of the ACM*, 18(6):311 – 317, 1975.

[PM91] T.M. Preto and L.P. Magalhães. Rt 28/91 - prosim - scanline, 1991.

[Pop89] J. Popsel. Highlight shading: Lighting and shading in a phigs+/pex-environment. In *Eurographics'89 Award Paper*, 1989.

[QM91] M.S. Queiroz and L.P. Magalhäes. O método radiosidade em um ambiente de síntese de imagens foto-realísticas. In *SIBGRAPI91*. SBC, 1991.

[Que] M.S. Queiroz. Uma metodologia híbrida baseada na técnica de ray-casting. Tese de Mestrado em Preparação.

[Rog85] D.F. Rogers. *Procedural elements for computer graphics*. McGraw-Hill, 1985.

[SBS69] R.A. Schumacker, B. Brand, and W. Sharp. Study for applying computer generated images to visual simulation, 1969.

[SSS74] I.E. Sutherland, R.F. Sproull, and R.A. Schumacker. A characterization of ten hidden-surface algorithms. *Computing Surveys*, 6(1):1 – 55, 1974.

[TM87] D. Thalmann and N. Magnenat. *Image Synthesis*. Springer-Verlag, 1987.

[VG84] C. Verbeck and D. Greenberg. A comprehensive light source description for computer graphics. *IEEE Computer Graphics and Aplications*, 4(7):66 – 75, 1984.

[WA77] K. Weiler and P. Atherton. Hidden surface removal using polygon area sorting. In *SIGGRAPH'77*, 1977.

[War69] J.E. Warnock. A hidden surface algorithm for computer half-tone picture representation, 1969.

[War83] D.R Warn. Lighting controls for synthetic images. In *SIGGRAPH'83*, 1983.

[Wat70] G.S. Watkins. A real time visible surface algorithm, 1970.

[WEE67] R.C. Wylie, G.W. Evans, and A. Erdahl. Halftone perspective drawings by computer. In *AFIPS FJCC*, 1967.

[Wil78] L. Williams. Casting curved shadows on curved surfaces. In *SIGGRAPH'78*, 1978.