UNIVERSIDADE FEDERAL FLUMINENSE

ILDENIR DA COSTA BARBOSA

Um método de Iluminação Global Aproximada para Ambientes Complexos baseado em Radiosidade e Decomposição Célula-Portal

Niterói

2008

UNIVERSIDADE FEDERAL FLUMINENSE

ILDENIR DA COSTA BARBOSA

Um método de Iluminação Global Aproximada para Ambientes Complexos baseado em Radiosidade e Decomposição Célula-Portal

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de concentração: Computação Visual e Interfaces.

Orientador:
Anselmo Antunes Montenegro

Niterói
2008

Um método de Iluminação Global Aproximada para Ambientes Complexos baseado em Radiosidade e Decomposição Célula-Portal

Ildenir da Costa Barbosa

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de concentração: Computação Visual e Interfaces.

Aprovada por:

Prof. D.Sc. Anselmo Antunes Montenegro / IC-UFF
(Presidente)

Prof. D.Sc. Esteban Walter Gonzalez Clua / IC-UFF

Prof. Ph.D. Marcelo Gattas / PUC-RIO

Niterói, 12 de setembro de 2008

Resumo da Dissertação apresentada à UFF como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação (M.Sc.)

Um método de Iluminação Global Aproximada para Ambientes Complexos baseado em Radiosidade e Decomposição Célula-Portal

Ildenir da Costa Barbosa

Setembro/2008

Orientador: Anselmo Antunes Montenegro
Programa de Pós-Graduação em Computação

O presente trabalho propõe uma técnica de iluminação global capaz de simular, em tempo de execução, o transporte de luz para superfícies perfeitamente difusas em cenas complexas, com alto grau de oclusão. O método proposto combina uma estratégia de *culling* conhecida como decomposição *cell-portal*(células-portais) com o cálculo da radiosidade em hardware gráfico, explorando o alto grau de paralelismo disponibilizado pelas modernas arquiteturas de GPU.

O algoritmo inicia com uma decomposição *cell-portal* da cena na qual deseja-se simular o transporte de luz. A cada portal da célula anexamos uma fonte de luz artificial com mesma geometria. Essa fonte de luz artificial representa a radiosidade emitida por uma célula incidente à célula adjacente. A cada passo da simulação, executa-se o algoritmo de radiosidade em cada célula isoladamente. Em seguida, um passo de sincronização é executado nas fontes artificiais de modo que a radiosidade de um célula chegue na célula adjacente. Então, a cena é renderizada a partir do resultado obtido e um novo passo de sincronização é realizado. A medida que mais passos são realizados, a iluminação da cena fica mais acurada.

Abstract of Dissertation presented to UFF as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A method of Global Illumination for Complex Environment based on Radiosity and Cell-Portal Decomposition

Ildenir da Costa Barbosa

September/2008

Advisors: Anselmo Antunes Montenegro
Department: Computer Science

This work proposes a technique of global illumination that is able to simulate, at run-time, the light transport of perfect diffuse surfaces in complex scenes with high degrees of occlusion. The method combines a culling strategy known as cell-portal decomposition with the computation of the scene radiosity in graphics hardware, exploring the parallelism provided by modern GPUs.

The algorithm starts with a cell-portal decomposition of the scene where the light transport simulation is supposed to be computed. The portal of each cell is attached to a source of artificial light with the same geometry. This source of artificial light represents the radiosity emitted by a cell incident to an adjacent cell. In each step of the simulation, the radiosity algorithm is executed for each individual cell. Then, a synchronization step is run on the artificial light sources so that the radiosity of a cell reaches the adjacent cell. Finally, the scene is rendered based on the results obtained and a new syncronization step is started. As more steps are performed, the computed radiosity of the scene becomes more accurate.

Palavras-chave

1. Radiosidade
2. Iluminação global
3. Decomposição célula portal
4. GPU

Sumário

1 Introdução **8**

2 Trabalhos Relacionados **12**

3 O Problema de Iluminação **15**

4 Modelos de iluminação local **17**

4.1 Difusor Lambertiano 17

4.2 Refletor especular perfeito 19

4.3 Componente Ambiente 20

4.4 Superfície difusa-especular 20

5 Modelo de Iluminação Global **22**

5.1 Métodos de Raytracing 22

5.2 Métodos de Radiosidade 27

5.2.1 Hemi-cube 29

5.2.2 Refinamento Progressivo 31

5.2.3 Hierárquico 33

5.2.4 Elementos Finitos 34

5.3 Cálculo da iluminação global usando Esféricos Harmônicos 38

5.3.1 Esféricos Harmônicos 39

5.3.2 Iluminação de superfícies difusas 40

5.3.3 Sem sombreamento 41

5.3.4 Auto-sombreamento 41

5.3.5 Interreflexões 42

5.3.6 Rendering 44

6 Radiosidade em GPU **45**

7 Decomposição Cell-Portal **53**

8 Radiosidade integrada a decomposição Cell-Portal **57**

9 Resultados **63**

10 Conclusão **75**

11 Integração de Monte-Carlo **81**

12 Polinômios de Legendre **83**

Lista de Figuras

4.1 Espalhamento da energia incidente por uma superfícies difusa. 18

4.2 Lei de Lambert 18

4.3 Exemplo de superfícies difusas. 19

4.4 Reflexão especular 19

4.5 Exemplo de superfície especular com variação de n 20

5.1 Interação dos raios de luz de duas fontes de luz com os objetos de uma cena. 23

5.2 Algoritmo de raytracing 24

5.3 Algoritmo de raytracing 25

5.4 Imagem gerada pelo método de traçado de raios. 26

5.5 Derivação do fator de forma 28

5.6 Projeção da área do retalho P_j no hemisfério do retalho P_i 28

5.7 Hemi-cube 30

5.8 Discretização do hemi-cubo 30

5.9 Algoritmo para solucionar a equação de radiosidade. ΔB_i significa a radiosidade não lançada. 32

5.10 Imagem produzida pelo algoritmo *hemicube*. Extraida do programa Cornell University Program of Computer Graphics 32

5.11 Decomposição de um polígono em subpolígonos 33

5.12 Construção da hierarquia da matriz do fator de forma 35

5.13 Solução do sistema de equações utilizando o método de jacobi. 36

5.14 Exemplo de imagem gerada com o algoritmo de radiosidade hierárquica. 37

5.15 Visualização das 5 primeiras bandas das funções de base 39

5.16 Sombra 42
5.17 Interreflexões 43
5.18 Imagem produzida pelo algoritmo dos Esféricos Harmônicos extraída do artigo [23]. 44

6.1 Projeção estereográfica 49
6.2 Projeção estereográfica de uma retalho grosseiro. 49
6.3 Aproximação do fator de forma utilizando discos 50

7.1 Ilustração de uma cena com celulas e portais. 54
7.2 Grafo da cena com celulas e portais. 54
7.3 Diagrama com estrutura de objetos da decomposicao Cell-Portal. . . 55
7.4 Renderiza a decomposição célula-portal por meio de um percurso na estrutura de grafo. 55
7.5 Renderização da célula 55
7.6 Renderiza célula adjacente por meio do portal. 56

8.1 Diagrama com estrutura de objetos para o algoritmo de radiosidade com decomposicao Cell-Portal. 60

9.1 Imagem gerada para a cena 1 com textura 32x32 66
9.2 Imagem gerada para a cena 1 com textura 64x64 66
9.3 Imagem gerada para a cena 1 com textura 128x128 67
9.4 Imagem gerada para a cena 2 com textura 16x16 67
9.5 Imagem gerada para a cena 2 com textura 16x16 68
9.6 Imagem gerada para a cena 2 com textura 32x32 68
9.7 Imagem gerada para a cena 2 com textura 32x32 69
9.8 Imagem gerada para a cena 2 com textura 64x64 69
9.9 Imagem gerada para a cena 2 com textura 64x64 70
9.10 Imagem gerada para a cena 2 com textura 128x128 70
9.11 Imagem gerada para a cena 2 com textura 128x128 71
9.12 Imagem gerada para a cena 3 com textura 16x16 71
9.13 Imagem gerada para a cena 2 com textura 16x16 72
9.14 Imagem gerada para a cena 3 com textura 32x32 72

9.15 Imagem gerada para a cena 2 com textura 32x32 73

9.16 Imagem gerada para a cena 2 com textura 64x64 73

9.17 Imagem gerada para a cena 2 com textura 64x64 74

12.1 Polinômios de Legendre de ordem $n = 0, 1, 2, 3, 4, 5$ 84

12.2 Polinômios associados de Legendre de ordem $n = 1, 2$ onde $p11(x) = P_1^1(x)$, $p21(x) = P_2^1(x)$, $p22(x) = P_2^2(x)$ 85

Lista de Tabelas

9.1 Tempo das simulações 65
9.2 Número de quadriláteros das cenas 65

Capítulo 1

Introdução

Os modelos de iluminação assumem um papel preponderante no processo de síntese de imagens realísticas. A simulação de ambientes reais requer o uso de técnicas sofisticadas da matemática e da física, afim de aproximar ao máximo os efeitos de iluminação do mundo físico. Tais efeitos são tão sutis que muitas vezes passam desapercebidos pelos olhos de um observador desatento, porém, ao desconsiderá-los obtemos imagens distantes da realidade, o que é indesejável em determinadas situações.

Um destes fenômenos é a interação da energia luminosa refletida por todos os objetos do ambiente com um determinado objeto. Um ambiente iluminado dominantemente por iluminação indireta apresenta diversas dificuldades técnicas. Talvez a maior delas seja a determinação eficiente da visibilidade de um determinado objeto com relação aos outros. Junta-se a isto a necessidade de administrar cenas complexas em tempo real.

Os primeiros modelos para o cálculo da iluminação consideravam somente a iluminação direta de uma superfície por uma fonte de luz. Muitos deles se baseavam em aproximações sem qualquer compromisso com a realidade física, sendo puramente empíricos como o modelo de Phong [11]. Apesar de serem puramente empíricos, tinham a vantagem de serem facilmente implementáveis, serem simples conceitualmente e produzirem resultados visuais compatíveis com a realidade.

Contudo em aplicações específicas como, por exemplo, o planejamento da iluminação do interior de uma estrutura arquitetônica e a geração de imagens na indústria

cinematográfica, surge a necessidade de modelos mais reais. No caso do planejamento de interiores, a proximidade com a realidade física deve ser ainda maior, afim de possibilitar que um arquiteto faça todo o projeto da iluminação no computador e após construí-lo, obter o mesmo resultado planejado.

Nos anos 80, surgiram modelos mais sofisticados para a simulação de iluminação. A grande evolução de tais modelos foi a consideração do efeito de todos os elementos do ambiente em um objeto. Esses modelos ficaram conhecidos como *iluminação global* contrastando com os métodos anteriores conhecidos como *iluminação local*.

O modelo físico que generaliza a troca de energia luminosa foi descrito separadamente por Immel e Kajiya [16]. Eles apresentaram a mesma equação que descreve a intensidade luminosa que sai de uma superfície em uma dada direção em termos da intensidade produzida e da energia incidente proveniente de outros elementos do ambiente. Essa equação é a base dos modelos de iluminação e ficou conhecida como *Rendering Equation*.

Diante da complexidade do problema de troca de energia luminosa e da diversidade de materiais dos objetos, surgiram simplificações afim de permitir uma decomposição do problema em partes mais simples. Uma delas está relacionada com o tipo de material [11] de um objeto, o que levou a criação de algumas idealizações como as superfícies perfeitamente difusas e as perfeitamente especulares. As superfícies perfeitamente difusas seriam as superfícies rugosas que espalham energia luminosa incidente igualmente em todas as direções. Já as perfeitamente especulares seriam aquelas que se aproximam a um espelho. Após alguns estudos, observou-se que uma superfície arbitrária poderia ser aproximada pela combinação das idealizações difusa e especulares resultando em uma superficie *difuso-especular*. Essas idealizações não só simplificaram o problema, mas também permitiram um modo de especificar a característica do material através do grau de difusividade e especularidade.

Com o passar do tempo surgiram métodos que tratavam melhor superfícies especulares e outros, superfícies difusas, conhecidos, respectivamente, por Métodos de Traçado de Raios (*Ray Tracing*) e Métodos de Radiosidade (*Radiosity*). Em geral os métodos de radiosidade são independentes da posição do observador, enquanto que os de traçado de raios são dependentes, o que tem vantagens e desvantagens.

Uma integração de tais métodos, que considera a vantagem de cada um deles na simulação da iluminação global, foi proposta em [25].

No decorrer dos anos, muitos algoritmos de radiosidade foram apresentados. Um método classico foi descrito em [6]. Sua solução consistia basicamente no cálculo da equação de radiosidade, mais precisamente o fator de forma ou *Form-Factor*, utilizando princípios geométricos. Seguindo as diretrizes desse artigo, temos [5], [12] e [22]. Em essência, esses métodos se baseam na solução de um sistema de equações.

Outras abordagens como o método de elementos finitos [24], o Método de Galerkin [28] e o Método dos Esféricos Harmônicos (*Spherical Harmonics*) [13] [23] são baseadas na idéia de projeção da função descrita pela equação de iluminação (*rendering equation*) em um espaço de funções geradas por um conjunto de funções de base cuidadosamente selecionadas.

O objetivo deste trabalho é investigar uma uma técnica de iluminação global capaz de simular, em tempo de execução, o transporte de luz para superfícies perfeitamente difusas em cenas complexas, com alto grau de oclusão. O método proposto combina uma estratégia de *culling* conhecida como decomposição *cell-portal*(células-portais) com o cálculo da radiosidade em hardware gráfico, explorando o alto grau de paralelismo disponibilizado pelas modernas arquiteturas de GPU.

O algoritmo inicia com uma decomposição *cell-portal* da cena na qual deseja-se simular o transporte de luz. A cada portal da célula anexamos uma fonte de luz artificial com mesma geometria. Essa fonte de luz artificial representa a radiosidade emitida por uma célula incidente à célula adjacente. A cada passo da simulação, executa-se o algoritmo de radiosidade em cada célula isoladamente. Em seguida, um passo de sincronização é executado nas fontes artificiais de modo que a radiosidade de um célula chegue na célula adjacente. Então, a cena é renderizada a partir do resultado obtido e um novo passo de sincronização é realizado. A medida que mais passos são realizados, a iluminação da cena fica mais acurada.

O algoritmo de radiosidade em GPU empregado para realizar a simulação de radiosidade explora o paralelismo das GPUs por meio do cálculo da informação de radiosidade em vários pontos do retalho pelo processador de fragmentos das placas gráficas. Este mecanismo é diferente dos algoritmos dos algorimos tradicionais que

cálculam um valor de radiosidade para cada retalho e empregam a interpolação Gouraud para recontruir suavemente a solução.

No capítulo 4 é discutido o modelo mais simples de iluminação. Nele foi feita uma breve abordagem do modelo specular de *Phong*, o modelo difuso e outros detalhes referentes à iluminação local. O método de traçado de raios é apresentado no capítulo 5.1 onde encontra-se a descrição do algoritmo original e o método de radiosidade é apresentado no capítulo 5.2. Neste capítulo, os métodos mais influentes na area de radiosidade são descritos. Os algoritmos **hemi-cube**, refinamento progressivo, hierárquico e dos elementos finitos são detalhados nas seções 5.2.1, 5.2.2, 5.2.3 e 5.2.4, respectivamente. Na seção 5.3 é apresentado o método dos esféricos harmônicos, um dos mais difundidos para o cálculo da iluminação global. No capítulo 6 é apresentado o algoritmo de radiosidade em GPU utilizado nesse trabalho. O capítulo 7 faz uma breve explicação do funcionamento e estrutura de uma decomposição célula-portal. O algoritmo desenvolvido nesse trabalho é descrito no capítulo 8. Os resultado são apresentados no capítulo 9 e, por fim o trabalho é concluido no capítulo 10.

Capítulo 2

Trabalhos Relacionados

O algoritmo clássico de radiosidade por refinamento progressivo foi introduzido por Cohen, Wallace e Greenberg no artigo [5] onde se apresenta o método, otimizações e uma estratégia de subdivisão adaptativa. O algoritmo funciona basicamente lançando a energia luminosa de um retalho da cena para todos os outros. Um termo de grande importância no cálculo da radiosidade é o fator de forma que diz o quanto de energia deve ser transmitida entre os retalhos. A formulação clássica para o cálculo do fator de forma é o uso do hemi-cubo [6]. Apesar do hemi-cubo produzir bons resultados, o seu uso na *gpu* não é aconselhável devido à necessidade de realização de 5 renderizações, 4 para as laterais e 1 para o topo. Assim, é apresentado em [17] e [8] o cálculo baseado em projeções estereográficas.

Um outro algoritmo clássico de iluminação global é o algoritmo *ray tracing*. Nesse algoritmo, a iluminação em um determinado ponto de uma superficie é obtida pela absorção, transmissão e reflexão de um raio de luz determinado por sucessivos traçados de raios. Como o método é muito caro computacionalmente, o uso do hardware gráfico para otimizar o processo é desejavel. A operação de interseção entre um raio e um triângulo domina o processo de traçado de raios. Em [4] é proposta uma implementação em GPU da operação de interseção entre um raio e um triângulo.

Uma nova classe de algoritmos de iluminação global consiste nos métodos baseados em função de transferência pré-computada ou *Precomputed radiance transfer*. Nesses algoritmos, a simulação da iluminação é compactada em coeficientes, con-

hecidos como função de transferência, de funções de base para cada um dos vértices. Ou seja, cada vértice possui um conjunto de coeficientes que representa o quanto da energia incidente ele reflete. A etapa final de visualização se resume a interpolação dos coeficientes seguida do produto interno entre os coeficientes interpolados e os coeficientes da fonte de luz. O trabalho [13] ensina como aplicar a técnica usando *Spherical Harmonics* como função de base no desenvolvimento de um jogo. Mas o trabalho mais influente foi [23] que introduziu a técnica. Já existe, na literatura, versões que utilizam *wavelets* como funções de base.

Como a simulação da iluminação global é em geral cara computacionalmente, os renderizadores offline constumam executar os cálculos em múltiplos passos. Uma abordagem é a de dois passos que consiste na determinação de uma solução grosseira da iluminação global empregando técnicas como radiosidade ou *PhotonMap* no primeiro passo e no segundo passo, conhecido como *Final Gathering* , são completados os cálculos somente para os pontos visíveis. O artigo [14] propoe uma implementação em GPU para efetuar a etapa de Final Gathering.

O método de radiosidade apresentado por Chen-Chin Feng and Shi-Nine Yang [10] também utiliza uma decomposição célula-portal para calcular a radiosidade de uma cena complexa. O algoritmo funciona agendando as células visíveis para processamento paralelo. Diferentemente da abordagem empregada por nós, esse trabalho utiliza o conceito de buffer de visibilidade. O buffer de visibilidade contém as identificações do objetos das células adjacentes projetadas nos portais. Ou seja, os portais armazenam as identificações das células adjacentes, enquanto nosso método armazena a energia.

Outro problema fortemente relacionado com os métodos de iluminação global é a visibilidade. Como a visibilidade é um problema presente desde o início da computação gráfica, há um grande número de trabalhos relacionados ao tema. Inicialmente, o problema era determinar a visibilidade exata da cena do ponto de vista da câmera. De fato, antes do algoritmo z-buffer [11] a visibilidade exata era o problema principal.

Atualmente, o problema central é determinar e eliminar do pipeline gráfico partes da cena ocultas por outros objetos. Para cenários internos onde há um grande

número de oclusores foram desenvolvido algoritmos baseados na partição do ambiente. Os métodos mais utilizados são conhecidos como PVS [3] and [21]. Este método obtem uma estimativa da geometria visível, embora ainda possa haver geometria invisível que por ventura será eliminada pelo algoritmo de visibilidade exata. Geralmente, este método particiona o cenário em células e determina para cada célula qual célula da vizinhança é visivel.

O método baseado na decomposição cell-portal ou *Cell-Portal Graph* (*CPG*) emprega uma abordagem ligeiramente diferente. Em vez de pré-processar a geometria possivelmente visível, essa determinação é feita em tempo de execução utilizando a geometria do portal, como pode ser vista [20], [18] e [9]. Os primeiro trabalhos se preocupavam com a utilização do CPG e não com sua criação que era realizada manualmente. O processo de decomposição automática ainda está em um estágio inicial de desenvolvimento (veja [9] para maiores detalhes sobre o método baseado em uma representação volumétrica que emprega o algoritmo de segmentação watershed 3D).

Capítulo 3

O Problema de Iluminação

O processo de simulação da iluminação de um ambiente requer a determinação da energia emanada de um determinado ponto da superfície de um objeto. Intuitivamente, se pensarmos em um objeto sendo iluminado diretamente por uma fonte de luz, podemos supor que energia emanada de um ponto do objeto será uma porcentagem da energia incidente. E mais, se o objeto produzir energia luminosa, então a energia emanada do ponto será a soma da energia emitida e a porcentagem da energia incidente. É necessário considerar que a energia emanada pode atingir outras superfícies. Assim é razoável considerar o objeto como uma fonte de luz. Daí concluimos que a iluminação de um ambiente consiste em um grande sistema de troca de energias.

De fato, comprovou-se que tais suposições são verdadeiras e uma formulação que descreve tal sistema de transferência de energia foi desenvolvida. A fórmula, conhecida como *equação de iluminação*, determina a energia emanada de um ponto x na direção ω_o em termos da energia emitida e as energias incidentes. A equação de iluminação é dada por 3.1 onde N é a normal da superfície no ponto x, L_e a energia emitida e H o hemisfério superior de uma esfera unitária centrada em x. A função f determina o quanto a energia incidente contribuirá para a energia total. Essa função, conhecida como BRDF(*bidirecional reflectance distribuition function*), descreve a propriedade reflectiva do material e assume valores em $[0, \infty)$. A integração é feita sobre as direções incidentes ω_i em todo o hemisfério superior da esfera.

$$L(x \rightarrow \omega_o) = L_e(x \rightarrow \omega_o) + \int_H f(x, \omega_o \leftrightarrow \omega_i) L(x \leftarrow \omega_i)(N \cdot \omega_i) d\omega \tag{3.1}$$

O significado físico da energia L vai depender da aplicação. L pode ser radiosidade, irradiância ou radiância. Contudo, a grandeza usual é a radiância , pois, pode-se derivar as outras a partir desta. Para uma abordagem completa consulte [11], [1].

A formulação dada pela equação 3.1 não faz nenhuma menção a informação de cor. Para incluir cor no modelo, devemos colocar a radiância em função do comprimento de onda λ, que varia no espectro produzindo $L(x \rightarrow \omega_o, \lambda)$. Se λ está entre 400 a 800 nanômetros, então a radiância está na faixa visível. No entanto, devemos trabalhar a informação da cor no computador através de algum sistema de cor, tipicamente, o sistema RGB. Neste caso, consideramos a combinação das equações $L_r = L(x \rightarrow \omega_o, 700\text{nm})$, $L_g = L(x \rightarrow \omega_o, 546\text{nm})$ e $L_b = L(x \rightarrow \omega_o, 435\text{nm})$. Essa modificação também deve ser feita na função BRDF para cada canal de cor.

Capítulo 4

Modelos de iluminação local

Os modelos de iluminação local foram os primeiros modelos empregados no processo de visualização. Sua simplicidade, eficiência e a facilidade de implementação justificam sua predominância até mesmo nos dias atuais. Sua aplicabilidade, talvez, persista devido a grande quantidade de programas que necessitam somente da visualização de dados ou de um mundo virtual. Como exemplo, podemos citar um modelador de objetos, uma aplicação de visualição de moléculas, um plotador de superfícies. Nessas aplicações, a eficiência é um fator dominante e o realismo torna-se secundário.

A grande característica do modelo local está na consideração de somente interações entre superfícies e fontes de luz. Deste modo, temos somente três elementos a considerar: o objeto, as fontes de luz e o observador. Sobre essas condições, o problema é *determinar a energia luminosa proveniente de um ponto* $\boldsymbol{p}$ *do objeto na direção do observador.*

4.1 Difusor Lambertiano

Como dito anteriormente, o material de um objeto pode ser considerado como uma combinação de dois tipois de materiais ideais: o material difuso perfeito e o especular perfeito. Trataremos agora do difuso perfeito.

Uma superfície é um difusor perfeito quando a energia incidente é refletida em todas as direções. Esse fato está ilustrado na figura 4.1. O difusor perfeito pode

ser visto como uma superfície extremamente irregular em uma escala microscópica(macroscópicamente a superfície pode ser bem regular). Um exemplo que se aproxima muito do difusor ideal é o giz escolar.

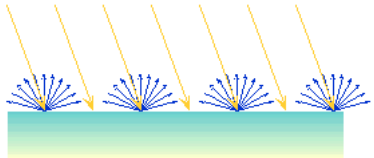


Figura 4.1: Espalhamento da energia incidente por uma superfícies difusa.

A relação matemática entre o energia incidente L_i e a energia refletida L_r em uma superfície difusa ideal é dada pela lei Lambert. A lei diz que "se um raio tem intensidade radiante L_i, então a intensidade radiante refletida será

$$L_r = L_i \cos\theta_i = L_i(\omega_i \cdot N)$$

onde ω_i é a direção de incidência no ponto e N a normal no ponto da superfície". A figura 4.2 mostra essa relação. O tamanho dos vetores azuis denota a energia refletida. Observe que a medida que o raio incidente se afasta da normal a energia refletida diminui proporcionalmento ao cosseno do ângulo com a normal.

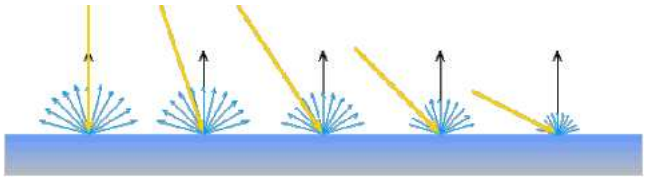


Figura 4.2: Lei de Lambert

Assim para o modelo local, a componente difusa, considerando somente uma fonte de luz, é dada por

$$I_d = k_d I(\omega \cdot N)$$

onde ω e I são, respectivamente, a direção de incidência e a intensidade da fonte de luz. A constante k_d corresponde a característica do material e pode assumir valores em $[0, 1]$. Veja na figura 4.3 exemplos de superfícies com essa componente. Se houver mais que uma fonte de luz, então a componente difusa será dada pela

soma da componente difusa de cada fonte de luz

$$I_d = k_d\Big[\sum_j I_j(\omega_j \cdot N)\Big]$$

Figura 4.3: Exemplo de superfícies difusas.

4.2 Refletor especular perfeito

Uma superfície especular perfeita satisfaz a lei de reflexão que diz:

"Para cada raio de incidência, o ângulo de incidência e reflexão são iguais. Isto é, $\theta_i = \theta_r$"

Essa lei corresponde a um espelho ideal. No entanto, um material real tende, devido a pequenas imperfeições na superfície, a espalhar os raios refletidos em uma pequena vizinhança do raio ideal, como na figura 4.4.

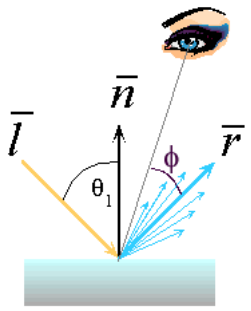


Figura 4.4: Reflexão especular

O modelo empregado para simular essas pequenas imperfeições é conhecido como modelo de Phong. Este é um modelo matemático puramente empírico. Ele relaciona a intensidade incidente e refletida pela expressão

$$I_r = I(cos\phi)^n$$

onde ϕ é o ângulo entre a direção do observador e o raio refletido como descrito na figura 4.4. Dessa relação, extraimos a componente especular

$$I_s = k_s I(\nu \cdot \bar{r})^n$$

onde ν é a direção do observador e $\bar{r}$ a direção do raio ideal refletido.

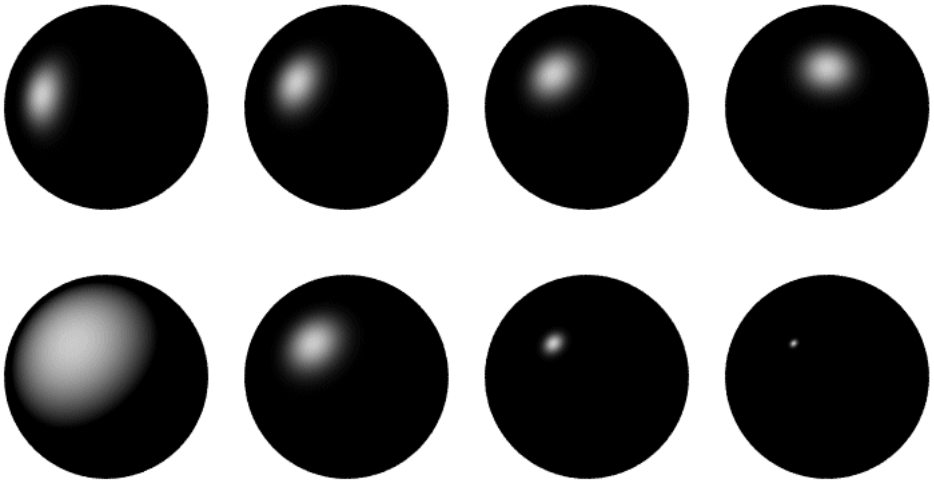


Figura 4.5: Exemplo de superfície especular com variação de n

4.3 Componente Ambiente

A componente ambiente corresponde a intensidade luminosa proveniente de inter-reflexões. Como o modelo local não calcula inter-reflexões, a componente ambiente é modelada por um valor constante I_a

4.4 Superfície difusa-especular

Uma superfície difusa-especular é modelada como sendo a combinação das superfícies ideais. Assim, a componente difusa-especular é igual componente difusa mais a componente especular mais a componente ambiente. Ou seja,

$$I = I\Big[k_d(\omega \cdot N) + k_s(\nu \cdot \bar{r})^n\Big] + k_a I_a$$

No caso de várias fontes de luz temos

$$I = k_a I_a + \sum_j I_j \left[k_d(\omega \cdot N) + k_s(\nu \cdot \bar{r})^n \right]$$

Capítulo 5

Modelo de Iluminação Global

5.1 Métodos de Raytracing

Traçado de Raios ou *Raytracing* é uma técnica capaz simular muitos fenômenos, como reflexões, refrações, *motion blur*, profundidade de campo e etc, que são quase impossíveis de serem produzidos com outras técnicas.

O método foi introduzido por Whitted [27] como uma extensão do modelo local de iluminação. Em seu modelo, a componente difusa do modelo local permanecia inalterada e determinava a componente especular através do lançamento de raios secundários. Em trabalhos anteriores ao de Whitted, os raios eram lançados de uma fonte de luz e percorriam o ambiente até atingir um pixel da tela. O problema com essa abordagem está no fato que nem todos os raios da fonte de luz irão atingir a tela virtual. Assim foi proposto o traçado de raios reverso onde o raio inicia-se em um pixel da tela virtual e percorre o ambiente até um número limitado de raios secundários.

Um exemplo da interação dos raios de luz em um ambiente é mostrado na figura 5.4. O processo é iniciado com o lançamento do raio E que parte do ponto de observação. No instante que o raio intersecta o objeto 3 há a bifurcação do mesmos em dois raios R_1 e T_1 , raio de reflexão e raio de transmissão, respectivamente, e adicionalmente lança-se dois raios auxiliares S_1 e S_2 em direção as fontes de luz para determinar pontos de sombra. O raio R_1 intersecta o objeto 9 e produz R_3, T_2 e, por sua vez, T_1 intersecta o objeto 6 que produz o raio R_2. Como R_3, T_2 e

R_2 não intersectam nenhum outro objeto, a luminosidade obtida nas direções R_1 e T_1 será dada pela componente difusa do objeto 9 e do objeto 6, respectivamente. A intensidade final refletida na direção E é dada pela soma da intensidade obtida no raio R_1 multiplicada pela constante especular, a intensidade do raio T_1 multiplicada pela constante de transmissão e a componente difusa do objeto 3.

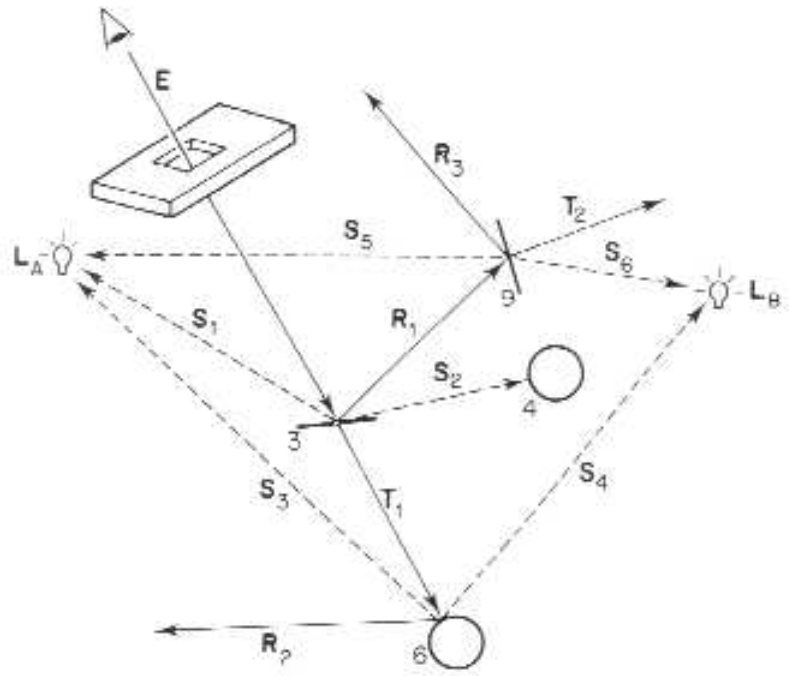


Figura 5.1: Interação dos raios de luz de duas fontes de luz com os objetos de uma cena.

O modelo era dado pela expressão

$$I = I_a + k_d \sum_j (\omega_j \cdot N) + k_s S + k_t T$$

onde S é a intensidade da luz incidente na direção ν, que é a direção do observador, T a intensidade de luz obtida da direção de transmissão.

A determinação da intensidade luminosa em um ponto é dada de modo recursivo pelo algoritmo 5.2 e 5.3. O processo consiste basicamente em lançar um raio partindo de cada pixel da tela e determinar o objeto que intersecta este raio. A intensidade do pixel será dada pela intensidade refletida pelo ponto de interseção do objeto, que por sua vez depende da intensidade proveniente da direção de incidência e transmissão. Note no algoritmo que há uma recursão implicita nas rotinas raytrace

e shade. Observe também que não foi dado um critério de parada para a recursão. Um critério razoável seria a limitação do número de raios traçados.

```
main(){
 para cada pixel (x,y) da imagem faça{
 raio= Criar raio que inicie no centro de projeção e
 passe pelo ponto (x,y);
 Intensidade(x,y)= raytrace(raio);
 }
}
raytrace(raio){
 p= Calcule a interseção de raio com o objeto mais próximo;
 material= material do objeto;
 return  shade(raio,p, material);
}
```

Figura 5.2: Algoritmo de raytracing

```
shade(raio, p, material){
  intensidade= 0;
  Para cada fonte de luz L faça{
   Se L for vísivel{
   intensidade= intensidade + intensidade(P);
   }senão{
   intensidade= intensidade + intensidade ambiente;
   }
   Se material == especular{
   raio= raio refletido;
   intensidade= intensidade + raytrace(raio);
   }
   Se material == tranlúcido{
   raio= raio refletido;
   intensidade= intensidade + raytrace(raio);
   }
  }
 return intensidade;
}
```

Figura 5.3: Algoritmo de raytracing

O problema com o método proposto por Whitted está nas imagens resultantes que possuem reflexões e refrações muito nítidas. Isto deve-se à precisão geométrica com que as direções dos raios são determinadas. Assim para obter sombras suaves necessita-se de um número muito grande de raios. Para contornar essa situação, Cook [7] propõe distribuir os raios, usando técnicas de amostragem estocástica, em vez de adicionar mais raios. O algoritmo de traçado de raios distribuídos também resolve o problema de *aliasing* inerente a amostragem pontual de raios, inclusive o *aliasing* de movimento cujo anti-aliasing é o *motion blur*.

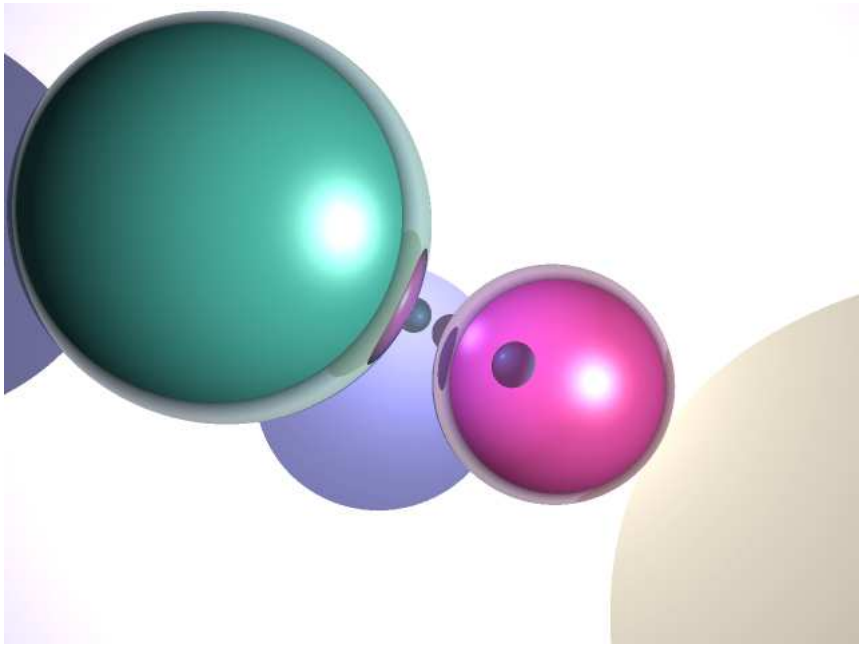


Figura 5.4: Imagem gerada pelo método de traçado de raios.

5.2 Métodos de Radiosidade

Os métodos de radiosidade são técnicas clássicas que simplificam o problema da iluminação global ao considerar somente superfícies difusas ou *Lambertianas*. Sobre essa consideração, a grandeza radiosidade $B(x)$ independe da direção incidente. A função BRDF é substituida pela função $\rho(x)$, conhecida como albedo, que pode ser tirada da integral. Assim, a equação de iluminação toma a forma

$$B(x) = E(x) + \rho(x) \int_{\Omega} B(x') \cdot G(x, x') \cdot \cos\theta d\omega \tag{5.1}$$

Onde $B(x)$ é a radiosidade e $E(x)$ a energia produzida pela superfície no ponto x da mesma. O termo geométrico $G(x, x') = V(x, x') \dfrac{\cos\theta_i \cos\theta_j}{||x - x'||^2}$ onde V é a visibilidade entre x e x', θ_i o ângulo entre $x \rightarrow x'$ e a normal da superfície em x', θ_j o ângulo entre $x \rightarrow x'$ e a normal da superfície em x. O ponto x' denota o primeiro ponto visível de x na direção ω.

A integral em 5.1 pode ser aproximada por uma soma finita se os objetos da cena são considerados como conjuntos de retalhos poligonais. Para cada retalho P_i assumimos um valor constante de radiosidade B_i. Isso tranforma a expressão 5.1 em

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j \tag{5.2}$$

O termo F_{ij} é conhecido como fator de forma. Ele indica a porcentagem de luz que parte do retalho P_i e chega em P_j. Se o retalho P_j não é visivel do ponto de vista de P_i, então $F_{ij} = 0$.

Para dois elementos infinitesimais de area dA_i e dA_j(figura 5.5) dos retalhos P_i e P_j, respectivamente, temos o fator de forma

$$F_{dA_i dA_j} = \frac{\cos\theta_i \cos\theta_j}{\pi r^2}$$

De fato, a projeção da área infinitesimal dA_j no hemisfério do retalho P_i é dada por

$$\bar{A}_j = \frac{dA_j \cos\theta_j}{r^2}$$

, pois dA_j é muito menor que a distância r entre os retalhos P_i e P_j. Pela lei de Lambert, a influência do retalho P_j sobre P_i é proporcional ao cosseno do ângulo de

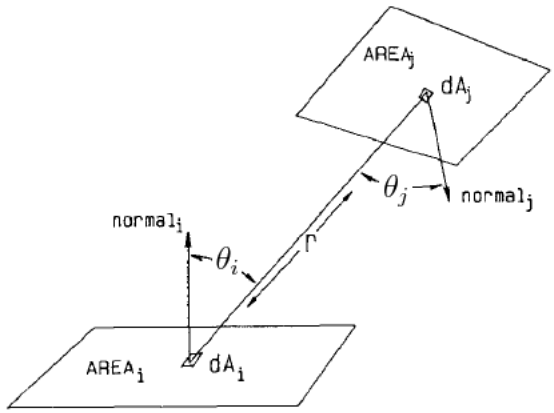


Figura 5.5: Derivação do fator de forma

incidência. Isto conduz a seguinte interpretação ilustrada na figura 5.6: O fator de forma é dado pela fração entre projeção ortogonal da área $\bar{A}_j$ na base do hemisfério e a área da base do hemisfério de raio unitário

$$F_{dA_i dA_j} = \frac{\bar{A}_j \cos\theta_i}{\pi} = \frac{\cos\theta_j \cos\theta_i}{r^2\pi} dA_j$$

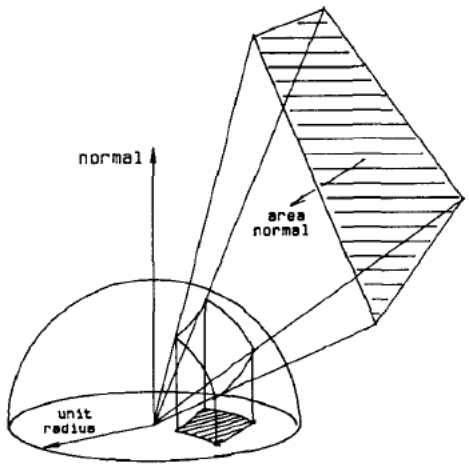


Figura 5.6: Projeção da área do retalho P_j no hemisfério do retalho P_i

Integrando sobre a área do retalho j e i obtemos

$$F_{ij} = \int_{A_i}\int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} dA_j dA_i$$

Rearranjando a expressão 5.2 obtemos o sistema de equações

$$(I - F)B = E \tag{5.3}$$

,ou mais precisamente,

$$\begin{pmatrix} 1 & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1 & \cdots & -\rho_2 F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \vdots & 1 \end{pmatrix} \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{pmatrix} = \begin{pmatrix} E1 \\ E_2 \\ \vdots \\ E_n \end{pmatrix}$$

A solução desse sistema pode ser obtida por meio de qualquer método de algebra linear computacional. Como $\sum F_{ij} \leq 1$ e $\rho_i < 1$, então o sistema é diagonal dominante e pode ser resolvido eficientemente por métodos iterativos como Gauss-Seidel.

Várias técnicas foram propostas para resolver a equação de radiosidade. Muitas delas se concentram no cálculo eficiente do fator de forma , pois, a grande complexidade computacional está no cálculo deste termo.

O primeiro método proposto [6] utilizava um conceito simplificado de um hemisfério, chamado *hemi-cube*, para calcular o fator de forma. Isto é, a integração da equação 5.2 era realizada de forma geométrica. Em vez de projetar o ambiente no hemisfério, o algoritmo projeta-o em um *hemi-cube* centrado no retalho. Este algoritmo permite a geração de efeitos como sombra e penumbra.

5.2.1 Hemi-cube

Este algoritmo utiliza uma aproximação do hemisfério chamado *hemi-cube* para calcular o fator de forma. Um *hemi-cube* em um retalho P_j é um cubo centrado no retalho. O fator de forma é obtido projetando todos os retalhos do ambiente nesse hemi-cubo, isto é, executa-se o mesmo procedimento descrito anteriormente para determinar o fator de forma só que substituindo o hemisfério pelo hemi-cubo. O retalho divide o *cubo* em duas metades, uma superior e a outra inferior como na figura 5.7.

As faces do cubo são discretizada em pixels, como na figura 5.8, e cada retalho P_j é projetado nesses pixels. Para cada pixel, um fator de forma é calculado. O fator de forma total será a soma do fator de forma de cada pixel. Também associamos um *z-buffer* em cada face para gerenciar a projeção de vários retalhos num mesmo pixel. Ou seja, para vários retalhos que projetam sobre um mesmo pixel, somente

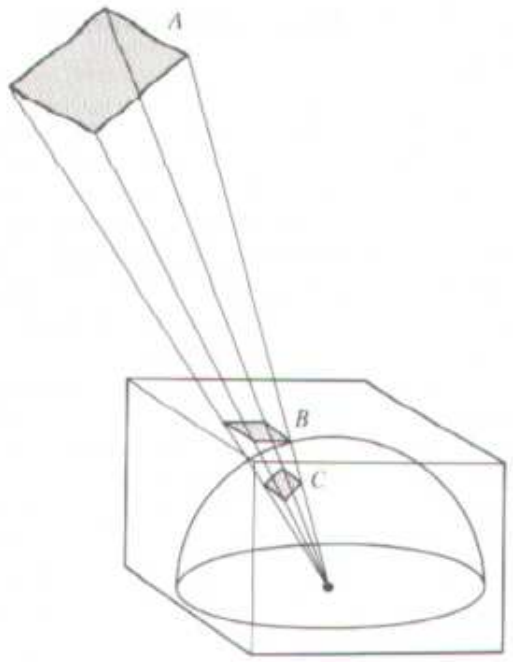


Figura 5.7: Hemi-cube

o retalho vísivel, ou equivalentemente o mais próximo, ao pixel será considerado no cálculo.

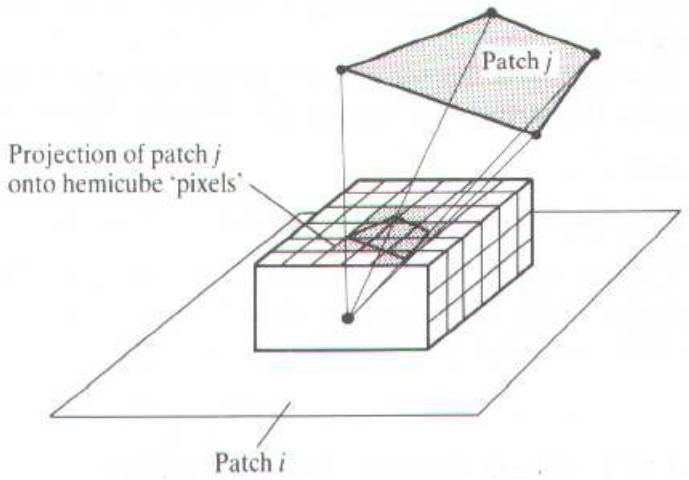


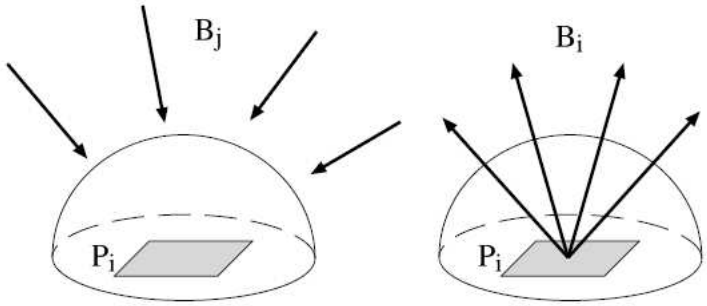
Figura 5.8: Discretização do hemi-cubo

Os resultados obtidos por esse algoritmo são de extrema qualidade. Entretanto, a complexidade do algoritmo é relativamente grande, $O(n^2)$ onde n é o número total de retalhos do ambiente, podendo ser um limitador para cenas complexas, além de poder haver problemas de aliasing devido a resolução da discretização do hemi-cube.

A vantagem do algoritmo está na simplicidade, facilidade de implementação e na possibilidade de utilizar recursos existentes em hardware.

5.2.2 Refinamento Progressivo

A diferença entre o algoritmo hemi-cube e o algoritmo de refinamento progressivo está na forma de distribuir a radiosidade. Enquanto no hemi-cube um retalho recebe energia de todos os outro do ambiente, no algoritmo de refinamento progressivo o retalho lança sua radiosidade para todos os outros retalhos (figura 5.2.2).



Baseado no teorema de reciprocidade $A_iF_{ij} = A_jF_{ji}$, a influência do retalho P_i no retalho P_j é dada por

$$B_{ji} = \rho_j B_i F_{ji} = \rho_j B_i F_{ij} \frac{A_i}{A_j}$$

Assim, um passo da iteração é dado por $B_j = B_j + \rho_j B_i F_{ij} \frac{A_i}{A_j}$. O fator de forma F_{ij} ainda é determinado empregando a técnica do hemi-cube. A solução da equação de radiosidade será dada pelo pseudocódigo 5.9 no qual ΔB_i significa a quantidade de energia radiante que ainda não foi distribuida, ou lançada, entre os retalhos. Inicialmente todos os B_i e ΔB_i estão zerados, exceto os referentes às fontes de luz que possuirão os valores de emissão. As iterações são executadas até obter a precisão desejada.

Note que se iniciarmos as iterações com retalhos que não emitem luz, estaremos distribuindo energias de valor zero, e portanto, realizando cálculos desnecessários. Intuitivamente, sabemos que os retalhos de maior energia radiante exercem maior influência na iluminação do ambiente. Isso indica que devemos iniciar as iterações de modo ordenado considerando primeiro os retalhos de maior energia radiante, ou seja, o retalho que possuir o maior produto B_iA_i.

1. Para cada iteração faça:
2. Para cada retalho P_i faça:
3. Calcular o fator de forma F_{ij}, $\forall j$, usando um hemi-cubo em P_i
4. Para cada retalho P_j faça:
5. $\Delta R = \rho_j \Delta B_i F_{ij} \frac{A_i}{A_j}$
6. $\Delta B_j = \Delta B_j + \Delta R$
7. $B_j = B_j + \Delta R$
8. Fim-Para
9. $\Delta B_i = 0$
10. Fim-Para
11. Fim-Para

Figura 5.9: Algoritmo para solucionar a equação de radiosidade. ΔB_i significa a radiosidade não lançada.

Figura 5.10: Imagem produzida pelo algoritmo *hemicube*. Extraida do programa Cornell University Program of Computer Graphics

5.2.3 Hierárquico

Em uma cena real, a energia radiante refletida por um objeto relativamente distante de um outro exercerá pouca influência sobre a radiosidade deste. Isso nos sugere a utilização de um valor padrão para o fator de forma quando a distância entre dois retalhos forem suficientemente grandes. Deste modo, para cada par de retalhos efetuamos uma estimativa do fator de forma considerando um como um elemento infinitesimal e o outro como um elemento finito, e comparamos com o valor padrão. Se a estimativa for menor que o valor padrão, então usamos o valor padrão como fator de forma.

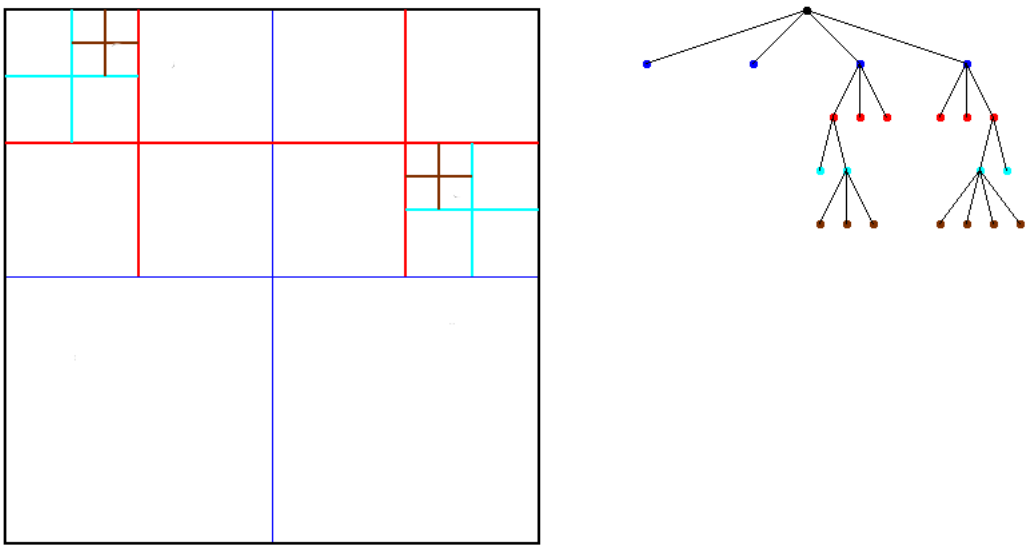


Figura 5.11: Decomposição de um polígono em subpolígonos

O algoritmo hierárquico, proposto em [15], emprega esse princípio para construir uma representação hierárquica da matriz do fator de forma. O algoritmo inicia com a conversão de polígonos em retalhos relativamente grandes a partir do qual constrói-se uma representação da matriz do fator de forma através de sucessivas subdivisões dos retalhos, como descrito no pseudocódigo 5.12, até obtermos a estimativa desejada do fator de forma ou chegarmos ao menor retalho possível chamado de elemento. Assumindo que cada polígono é um quadrilátero, o processo de subdivisão produz a árvore quaternária representada na figura 5.11. Com a hierarquia da matriz do

fator de forma, aplica-se o algoritmo progressivo ou o método de Jacobi para obter a solução do sistema de equações.

Após o processo de refinamento, a solução do sistema de equações é efetuada em duas etapas. Na primeira, obtém-se a radiosidade Bg de cada retalho na hierarquia da árvore quartenária isoladamente utilizando as ligações realizadas no refinamento empregando o algoritmo 5.13. A segunda etapa consiste em balancear a radiosidade de um retalho com a radiosidade de seus filhos. A energia recebida por um elemento será a energia recebida por ele diretamente somada a energia total de seu pai. Esse passo é realizado propagando a radiosidade Bg para os nós folha. Agora, devemos propagar a radiosidade dos filhos para os pais. Isso é feito subindo a árvore até chegar ao nó raiz. Durante a subida, a radiosidade de um sub-retalho é igual à média da radiosidade de seus filhos ponderada pela área.

Vale observar que o sistema de equações não é formado de forma explícita, como foi apresentado nos métodos anteriores. O algoritmo 5.13 resolve o sistema de equações usando o método de Jacobi empregando a estrutura hierárquica contruída. Também é descrito no artigo original [15] a solução por refinamento progressivo e usando métodos multigrid [26].

5.2.4 Elementos Finitos

Os algoritmos de radiosidade tradicionais apresentados, até o momento, calculam um valor constante de radiosidade para cada retalho. Isto significa que a radiosidade é obtida por uma aproximação constante por partes. Uma extensão razoável do algoritmo seria considerar aproximações de ordem superior como a linear, quadrática, cúbica ou superior. Deste modo esperamos obter soluções mais precisas com um número menor de retalhos.

Para alcançar este objetivo, empregamos a técnica dos elementos finitos para aproximar a solução por uma combinação linear de bases. Dentre vários método possíveis, como o da colocação, o de Galerkin será apresentado a seguir.

Inicialmente consideramos a equação de radiosidade na forma paramétrica, para podermos aplicar o método de Galerkin. Encapsulando toda a complexidade das interações entre as superfícies em uma única função núcleo $K_{ij}(s, t, u, v)$, a equação

```
Refinar( retalho p, retalho q, float Fe, float A){
    Fpq= EstimativaFatorForma( p, q);
    Fqp= EstimativaFatorForma( q, p);
    if( Fpq < Fe && Fqp < Fe  )
        Link(p,q);
    else{
        if( Fpq > Fqp ){
            if( subdividir(q, A) ){
                Refinar(p, q.ne, Fe, A);
                Refinar(p, q.nw, Fe, A);
                Refinar(p, q.se, Fe, A);
                Refinar(p, q.sw, Fe, A);
            }else
                Link( p, q);
        }else{
            if( subdividir(p, A) ){
                Refinar(q, p.ne, Fe, A);
                Refinar(q, p.nw, Fe, A);
                Refinar(q, p.se, Fe, A);
                Refinar(q, p.sw, Fe, A);
            }elsel
                Link( p, q);
       }
    }
}
```

Figura 5.12: Construção da hierarquia da matriz do fator de forma

de radiosidade 5.1 pode ser escrita como

$$B_i(s,t) = E_i(s,t) + \sum_j \int\int K_{ij}(s,t,u,v)B_j(u,v)dudv \tag{5.4}$$

```
Gather(Patch *p){
    Patch *q;
    float Fpq;
    if(p){
        p->Bg=0.0;
        ForAllElements(q,p->interactions){
            Fpq= FormFactor(p,q);
            p->Bg+= Fpq * p->Cd * q->B;
        }
    Gather(p->sw);
    Gather(p->se);
    Gather(p->nw);
    Gather(p->ne);
    }
}
```

Figura 5.13: Solução do sistema de equações utilizando o método de jacobi.

onde a função núcleo $K_{ij}(s,t,u,v)$ é dada pelo produto do fator de forma F_{ij}, a BRDF ρ_i, a área $A_i(s,t)$ e o termo de visibilidade $V_{ij}(s,t,u,v)$ produzindo

$$K_{ij}(s,t,u,v) = F_{ij}\rho_i A_i(s,t)V_{ij}(s,t,u,v)$$

A aproximação da radiosidade será dada pela combinação de funções de base. Denotando o conjunto de funções de base por $\{\Upsilon_k(s,t)|k = 0,1,\ldots\}$, onde k especifica uma determinada função de base, temos a aproximção

$$B_i(s,t) \approx \sum_k B_i^k \Upsilon_k(s,t)$$

onde B_i^k corresponde aos coeficientes da aproximação.

Considerando o produto interno entre duas funções f e g com função peso $\Psi(s,t)$

$$< f,g >_\Psi = \int_{-1}^{1}\int_{-1}^{1} f(s,t)g(s,t)\Psi(s,t)dsdt$$

Figura 5.14: Exemplo de imagem gerada com o algoritmo de radiosidade hierárquica.

assumimos que o conjunto de bases $\{\Upsilon_k(s,t)|k = 0, 1, \ldots\}$ é ortonormal, isto é,

$$\forall k, l < \Upsilon_k, \Upsilon_l >_\Psi = \delta_{kl}$$

sendo δ_{kl} um delta de Dirac. Deste modo, a projeção da função exata de radiosidade no espaço de funções $\{\Upsilon_k(s,t)|k = 0, 1, \ldots\}$ é dada por

$$B_i^k =< B_i, \Upsilon_k >_\Psi$$

O conjunto de bases pode ser formado por qualquer conjunto de funções que apresente as propriedade descritas. Contudo, utilizaremos os polinômios unidimensionais de Legendre. Assim, o conjunto de funções de base é obtido realizando o produto tensorial dos polinômios, ou seja, o produto dos polinômios aplicados a cada variável s e t.

Retornando a equação 5.4, projetaremos B_i no espaço de funções. Para isso, expandimos $B_j(u,v)$ em termos das funções de base

$$B_i(s,t) = E_i(s,t) + \sum_{jl} B_j^l \int\int K_{ij}(s,t,u,v)\Upsilon_l(u,v)dudv$$

Efetuando o produto interno em ambos os lados da expressão acima com a k-

ésima função de base Υ_k e usando as propriedade da funções de base obtemos

$$B_i^k = E_i^k + \sum_{jl} B_j^l < \int\int K_{ij}(s,t,u,v)\Upsilon_l(u,v)dudv, \Upsilon_k(u,v) >$$

Denotando o produto interno $< \int\int K_{ij}(s,t,u,v)\Upsilon_l(u,v)dudv, \Upsilon_k(u,v) >$ por K_{ij}^{kl} obtemos o sistema de equações

$$B_i^k - E_i^k = \sum_{jl} B_j^k K_{ij}^{kl}$$

que pode ser resolvido por qualquer método como, por exemplo, a eliminação gaussiana. O termo K_{ij}^{kl} é de difícil obtenção por ser formado por integrações. Ele pode ser calculado usando o método de quadratura gaussiana.

5.3 Cálculo da iluminação global usando Esféricos Harmônicos

Atualmente, grande parte do esforço de desenvolvimento dos métodos de radiosidade estão voltados para a simulação de iluminação global em aplicações em tempo real(*real-time*) ou interativas. Contudo, muitas das técnicas clássicas são impraticáveis do ponto de vista das aplicações de tempo real.

A técnica dos esféricos harmônicos ou *Spherical Harmonics(SH)* possibilita a síntese de imagens em tempo real de superfícies difusas em ambientes com iluminação de baixa frequência. Ele produz efeitos como sombras suaves, cáusticas e interreflexões. Outra vantagem desse método, com relação a aplicações de tempo real, é a facilidade de implementação em processadores gráficos.

O funcionamento do método começa com um pré-processamento que produz um conjunto de coeficientes para cada vértice da cena. Esses coeficientes codificam informações, como cor e sombra e de tranferência de energia. Em um segundo passo, a energia radiante incidente também é representada na forma de coeficientes. Para obtermos a energia radiante refletida, efetuamos o produto interno do coeficiente do vértice com o coeficiente da energia incidente.

5.3.1 Esféricos Harmônicos

Os esféricos harmônicos definem uma base ortonormal sobre a esféra unitária. Usando a parametrização

$$(x, y, z) \rightarrow (\text{sen}\theta \cos\phi, \text{sen}\theta\text{sen}\phi, \cos\theta)$$

a base de funções é definida como

$$y_l^m(\theta, \phi) = \begin{cases} \sqrt{2}K_l^m \cos(m\phi)P_l^m(\cos\theta) & , \quad m > 0 \\ \sqrt{2}K_l^m \text{sen}(-m\phi)P_l^{-m}(\cos\theta) & , \quad m < 0 \\ K_l^0 P_l^0(\cos\theta) & , \quad m = 0 \end{cases} \tag{5.5}$$

onde $l \in \mathbb{R}^+$, $-l \leq m \leq l$, P é o polinômio associado de Legendre da forma $P_{l+1}^m = x(2l-1)P_l^m($ veja o anexo 12 para maiores detalhes) e K um fator escalar para normalizar a função:

$$K_l^m = \sqrt{\frac{2l+1}{4\pi}\frac{(l-\|m\|)!}{(l-\|m\|)!}}$$

Valores baixos para l representam funções de base de baixa frequência sobre a esfera. Veja na figura 5.15 as funções de base para 5 bandas.

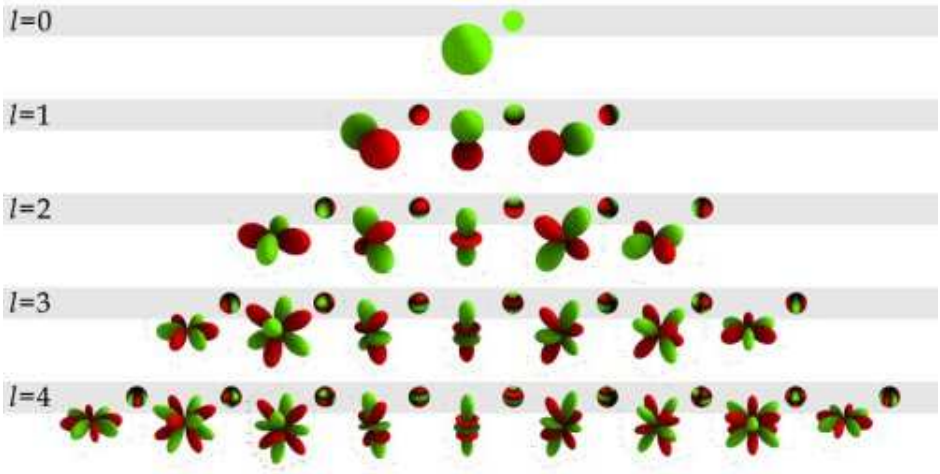


Figura 5.15: Visualização das 5 primeiras bandas das funções de base

A projeção de uma função escalar f definida na esfera S nas funções de base é dada por

$$c_l^m = \int_S f(s)y_l^m(s)ds$$

A recontrução de f na base 5.5 é obtida por

$$\tilde{f}(s) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} c_l^m y_l^m(s) = \sum_{i=1}^{n^2} c_i y_i(s)$$

onde $i = l(l+1)+m+1$. A aproximação melhora a medida que o número de bandas n cresce. Sinais com baixa-frequência podem ser precisamente aproximados com um número pequeno de bandas. Daí a necessidade do ambiente possuir iluminação de baixa frequência.

A ortonormalidade da base 5.5 conduz a uma propriedade básica. Dadas duas funções a e b definidas na esfera S, suas projeções satisfazem

$$\int_S \tilde{a}(s)\tilde{b}(s)ds = \sum_{i=1}^{n^2} a_i b_i$$

onde n corresponde ao número de bandas da expansão.

Com efeito, efetuando o produto de $\tilde{a}(s) = \sum_{i=1}^{n^2} a_i y_i(s)$ e $\tilde{b}(s) = \sum_{j=1}^{n^2} b_j y_j(s)$ obtemos $\tilde{a}(s)\tilde{b}(s) = \sum_{i=1}^{n^2} \sum_{j=1}^{n^2} a_i b_j y_i(s) y_j(s)$. Integrando essa expressão, temos

$$\int_S \tilde{a}(s)\tilde{b}(s) = \sum_{i=1}^{n^2} \sum_{j=1}^{n^2} \int_S a_i b_j y_i(s) y_j(s)$$

Como $y_i(s)$ e $y_j(s)$ são ortonormais, então $\int y_i(s)y_j(s)$ é igual a zero se $i \neq j$ e um caso contrário. Daí segue o resultado.

Ou seja, a integração é convertida em um produto interno entre os coeficientes. Outras propriedades podem ser encontradas em [13] , [23] e [19].

5.3.2 Iluminação de superfícies difusas

O processo de iluminação consiste na projeção da função de tranferência de cada vértice na base 5.5. A função de transferência é a função que multiplicada pela função de iluminação nos fornece uma aproximação da energia radiante no vértice, em outras palavras, o quanto de energia radiante proveniente da função de iluminação é refletida.

Para superfícies difusas, podemos considerar três tipos de função de tranferência: **sem sombreamento**, **auto-sombreamento** e **interreflexões** . A função de transferência **sem sombreamento** produz iluminação semelhante ao modelo de

iluminação local, considerando somente a iluminação direta por uma fonte de luz. A função do tipo **auto-sombreamento** é igual a anterior com um termo de visibilidade que produz sombras no modelo. Já o tipo **interreflexões** produz sombras entre os modelos.

5.3.3 Sem sombreamento

Retornando a equação de iluminação 3.1 e 5.1, consideramos somente o termo integral e que as únicas fontes de energia radiante são as fontes de luz, donde resulta a seguinte equação

$$L(x \rightarrow \omega_o) = \int_H f(x, \omega_o \leftrightarrow \omega_i) L(x \leftarrow \omega_i)(N \cdot \omega_i) d\omega$$

Como estamos supondo que a superfície é difusa, então podemos considerar a BRDF constante. Assim obtemos a expressão

$$L(x) = \frac{\rho(x)}{\pi} \int_H L(x \leftarrow \omega_i)(N \cdot \omega_i) d\omega$$

onde $\rho(x)$ é o albedo da superfície.

Daí vemos que a função de transferência deve ser $M = (N \cdot \omega_i)$. Então, para cada vértice x da cena, projetamos M na base 5.5 utilizando o método de integração de Monte Carlo e obtemos um conjunto de coeficientes. A energia radiante em x é dada pelo produto interno entre esses coeficiente e os coeficientes da fonte de luz.

5.3.4 Auto-sombreamento

O auto-sombreamento é idêntico ao caso anterior, só que com a adição de um termo de visibilidade $V(\omega_i)$ que retorna 0, se o raio ω_i é bloqueado pela superfície do próprio modelo, ou 1 caso contrário.

Essa modificação resulta na equação

$$L(x) = \frac{\rho(x)}{\pi} \int_H L(x \leftarrow \omega_i) V(\omega_i)(N \cdot \omega_i) d\omega$$

Assim tomamos a função de transferência

$$M = V(\omega_i) \max((N \cdot \omega_i), 0)$$

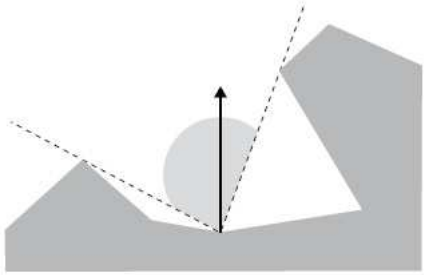


Figura 5.16: Sombra

O termo de visibilidade deve ser calculado utilizando o traçado de raios. Para cada direção ω_i, lança-se um raio partindo do ponto x e verifica-se se houve alguma interseção do raio com os polígonos do objeto. Contudo, não podemos simplesmente lançar os raios e encontrar a interseção do polígono com o raio. Veja na figura 5.16 que os raios internos ao modelo irão fazer interseção com faces do polígono de dentro para fora, resultando em um sombreamento errado. Portanto, devemos ter um cuidado especial na implementação desse termo.

Note que os coeficientes dessa função de transferência codificam a informação de visibilidade na vizinhança do ponto.

5.3.5 Interreflexões

Este tipo apresenta melhores resultados por considerar as interações entre vários objetos da cena. Nele, as fontes de energia radiante são as fontes de luz e as reflexões dos objetos da cena. A expressão da radiância total é dada pela soma da radiância obtida pelo auto-sombreamento e a radiância proveniente das reflexões. Produzindo a equação

$$L(x) = L_{\mathrm{ds}}(x) + \frac{\rho(x)}{\pi} \int_H \overline{L}(x \leftarrow \omega_i)(1 - V(\omega_i))(N \cdot \omega_i) d\omega$$

onde $L_{\mathrm{ds}}(x)$ é a radiância obtida no auto-sombreamento e o termo $\overline{L}$ a radiância proveniente de reflexões secundárias.

O cálculo dos coeficientes no caso da interreflexão é um pouco mais complicado que os outros casos. Inicialmente, calcula-se os coeficientes da função de tranferência auto-sombreamento, isto é, $(M_p)^0 = V(\omega)(N_p \cdot \omega)$ para todo ponto p do ambiente. Em seguida, para uma amostra do conjunto de direções ω no ponto p efetuamos os

cálculo

$$(M_p)^b = (M_p)^{(b-1)} + \frac{\rho_p}{\pi}(M_q)^{b-1}(1 - V(\omega))(N_p \cdot \omega)$$

onde $(M_p)^b$ é o vetor de coeficientes do ponto p na iteração b, $(M_q)^{b-1}$ é o vetor de coeficientes do ponto q(ponto mais próximo de interseção com o raio que parte de p na direção ω) na iteração anterior. A iteração é executada até um certo limite predefinido.

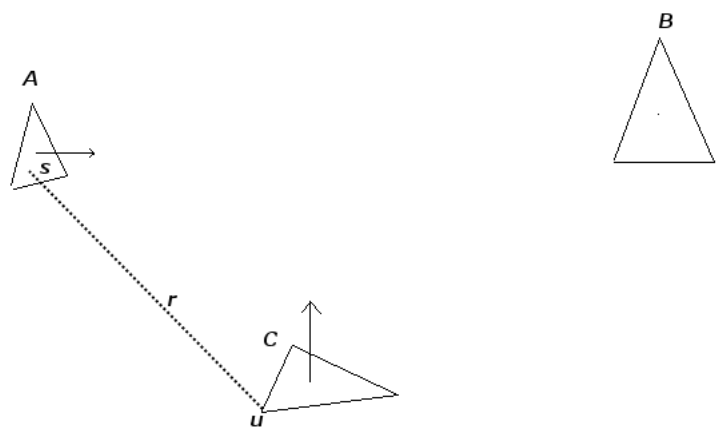


Figura 5.17: Interreflexões

Para ilustrar o funcionamento do algoritmo, veja a figura 5.17. Inicialmente os coeficientes M_{A}, M_{B} e M_{C} dos objetos A, B e C estão zerados. Calcula-se na primeira iteração os coeficiente da função de transferência empregando o auto-sombreamento. Na segunda iteração, para cada vértice do objeto são lançados raios nas direções do hemisfério superior e verificamos se houve interseção. Caso haja interseção, devemos somar os coeficientes da função de transferência do ponto de interseção aos coeficientes do vértice. No caso ilustrado na figura, o raio r que parte do vértice u do objeto C intersecta o ponto s no objeto A. Assim, devemos somar os coeficientes do ponto s que é obtido por interpolação dos coeficientes dos três vértices de A aos coeficientes do ponto u.

Observe que no caso da interreflexão, estamos incluindo o termo ρ_p dentro da função de tranferência. Isso significa que propriedade de cor do material do objeto

será embutida no vetor de coeficientes.

5.3.6 Rendering

Agora nós temos um conjunto de objetos cujos respectivos vértices possuem um vetor de coeficientes da função de transferência. Para visualizar os objetos da cena devemos para cada vértice do modelo:

1. Calcular a projeção $\tilde{L}$ da fonte de luz na base dos esféricos harmonicos.
2. Realizar a integração $\int_H \tilde{L}(x)\tilde{t}(s)ds = \sum_{i=0}^{n^2} L_i t_i$ onde t_i são os coeficientes da função de tranferência do vértice.
3. Guardar o valor da integração no vértice.

Após os passos anteriores, teremos a intensidade de luz para cada vértice. Para obter a iluminação em todo o objeto, aplicamos o algoritmo Gouraud Shading.

Figura 5.18: Imagem produzida pelo algoritmo dos Esféricos Harmônicos extraída do artigo [23].

Capítulo 6

Radiosidade em GPU

Refinamento progressivo em GPU

A grande vantagem do método de radiosidade por refinamento progressivo sobre a radiosidade por *gathering* está no cálculo do fator de forma. Enquanto o *gathering* exige o conhecimento prévio de todos os valores do fator de forma, requerendo espaço de armazenamento da ordem $O(n^2)$, onde n é o número de retalhos da cena, o refinamento progressivo determina o fator de forma durante o progresso da simulação.

O algoritmo inicia o processo de iluminação com o retalho que possui maior energia, geralmente os retalho fonte de luz, conhecidos como *shooters*. Em seguida, determina-se os retalhos que receberão a energia do shooter e para cada retalho é calculado o fator de forma shooter-retalho. Com o fator de forma calculado, a radiosidade de cada retalho é atualizada finalizando assim o primeiro ciclo de iterações. Agora, outro shooter é escolhido e o processo é repetido até alcançar um critério de convergência que pode ser um determinado número de interações. Os passos do algoritmo estão resumidos no pseudo-código 1.

No método clássico de radiosidade por refinamento progressivo, a radiosidade de um retalho é considerada constante por toda a superfície do retalho e representada por um número para cada retalho. A abordagem considerada neste trabalho, assume que cada retalho possui radiosidade variável ao longo de sua superfície e essa radiosidade é representada por um textura bidimensional. Essa estratégia também pode ser interpretada como uma subdivisão uniforme do retalho onde cada *texel*

Algoritmo 1 Algoritmo clássico para solucionar a equação de radiosidade. ΔB_i significa a radiosidade não lançada e B_j a radiosidade do retalho j.

1: **Para** cada iteração **faça**
2: **Para** cada retalho P_i **faça**
3: Calcular o fator de forma F_{ij}, $\forall j$, usando um hemi-cubo em P_i
4: **Para** cada retalho P_j **faça**
5: $\Delta R = \rho_j \Delta B_i F_{ij} \frac{A_i}{A_j}$
6: $\Delta B_j = \Delta B_j + \Delta R$
7: $B_j = B_j + \Delta R$
8: **Fim Para**
9: $\Delta B_i = 0$
10: **Fim Para**
11: **Fim Para**

represetaria um sub-retalho.

O algoritmo de radiosidade em GPU apresentado em [17] funciona ligeiramente diferente do apresentado no pseudo-código 1. Como a operação lançamento de energia do shooter para o retalho receptor não se adapta bem a arquitetura paralela da GPU, o novo algoritmo inverte esta operação. Isto é, o retalho receptor recebe energia do shooter, em vez do shooter lançar energia para o retalho receptor. Os passos do algoritmo radiosidade em GPU estão descritos no pseudo-código 2. A discussão de cada passo segue nas seções a seguir.

Visibilidade

A etapa de visibilidade é necessária para determinar os retalhos receptores de energia do shooter corrente. Esta etapa é realizada em dois passos. No primeiro, a cena é renderizada do ponto de vista do shooter usando projeção estereográfica (*stereographic projection*), veja a figura 6.1, através do *vertex shader* 6.0.1. A vantagem da projeção estereográfica sobre o hemi-cubo é que em um único passo determina-se a visibilidade, enquanto o hemi-cubo necessita de 5 passos. Por outro lado, se a malha de polígonos não for suficientemente refinada, a projeção sobre a esfera produzirá resultados incorretos (veja a figura 6.2). O resultado do primeiro passo é armazenado

Algoritmo 2 Algoritmo em GPU para solucionar a equação de radiosidade.

1: **Para** cada iteração **faça**
2: Selecione Shooter
3: renderize a cena do ponto de vista do shooter.
4: Guarde o resultado em um item buffer B
5: **Para** cada retalho receptor **faça**
6: **Para** cada texel **faça**
7: **Se** texel é visível em B **então**
8: Calcular o fator de forma F
9: $\Delta E = \rho F E$
10: Adicione ΔE a textura residual
11: Adicione ΔE a textura radiosity
12: **Fim Se**
13: **Fim Para**
14: **Fim Para**
15: Configure a energia do shooter $E = 0$
16: **Fim Para**

em um item buffer.

O *vertex shader* 6.0.1 realiza a projeção na esfera utilizando uma normalização e uma projeção ortográfica. A projeção ortográfica é obtida com a multiplicação de um ponto pela matriz

$$\begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

onde n é o plano near , f o plano far e r, l, t, b correspondem, respectivamente, a coordenada direita, esquerda, topo, e base do volume de visão. Como estamos considerando o volume canônico então $r = b = -1$ e $l = t = 1$, resultando na matriz

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Após normalizar o ponto *mpos* obtém-se *hemi_pt*, que é o ponto na esfera unitária. Multiplicando pela matriz acima mencionada, obtemos a expressão apresentada no vertex shader.

Shader 6.0.1 Vertex shader para efetuar projeção esterográfica.

```
uniform vec2 NearFar;


vec4 ProjPos;


void main(){
   vec3 mpos= vec3( gl_ModelViewMatrix
    * gl_Vertex);
   vec3 hemi_pt= normalize(mpos);
   ProjPos.xy= hemi_pt.xy *
    (NearFar.y - NearFar.x );
   ProjPos.z= (-2.0 * mpos.z
     - NearFar.x - NearFar.y);
   ProjPos.w= (NearFar.y - NearFar.x );
   gl_Position= ProjPos;
   gl_FrontColor= gl_Color;
}
```

No segundo passo, cada retalho receptor é renderizado ortograficamente num framebuffer com a mesma resolução da textura radiosity e residual. Essa operação cria uma correspondência biunívoca(um-para-um) entre os pixels do framebuffer e os texels das texturas radiosity e residual. Isso permite que um *fragment shader* referencie com as mesmas coordenadas um texel em radiosity e um pixel no frame-

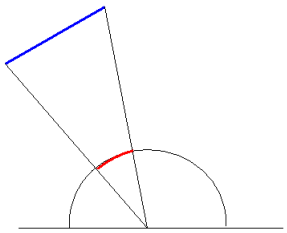


Figura 6.1: Projeção estereográfica

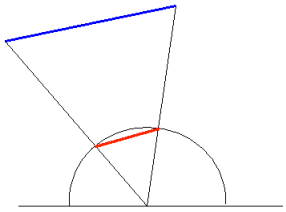


Figura 6.2: Projeção estereográfica de uma retalho grosseiro.

buffer. A visibilidade é determinada usando um *fragment shader* (veja 6.0.2) que projeta a posição do texel usando a projeção estereográfica e verifica se a ID do texel está no item buffer. Esse processo é semelhante ao funcionamento de um *shadow mapping* só que considerando ID em vez do z-buffer.

Shader 6.0.2 Fragment shader para determinar a visibilidade de um texel.

```
varying vec4 ProjPos;
float Visible() {
    vec3 proj= normalize(ProjPos.xyz);
    //Passa do intervalo [-1,1] para [0,1]
    proj.xy= proj.xy * 0.5 + 0.5;
    vec4 xtex= texture2D(ItemBuffer, proj.xy);
    // gl_Color eh a RecvID
    return xtex == gl_Color?1.0:0.0;
}
```

Fator de Forma

Agora que temos como determinar a visibilidade dos texels, podemos calcular o fator de forma entre o texel e o shooter. A formulação básica [6] para o fator de forma entre elementos infinitesimais de área assume que a área dos elementos infinitesimais é pequena comparada a distância entre eles. A violação dessa regra produz artefatos como *aliasing*, segundo [17]. Desta maneira, Lastra [17] sugere o uso da aproximação com discos ilustrado na figura 6.3. Essa aproximação calcula o fator de forma entre uma área finita j e uma área infinitesimal d_i pela subdivisão de j em m discos orientados.

$$F_{j \to d_i} = A_j \sum_{i=1}^{m} \frac{\cos\theta_j \cos\theta_i}{\pi r^2 + \frac{A_j}{m}}$$

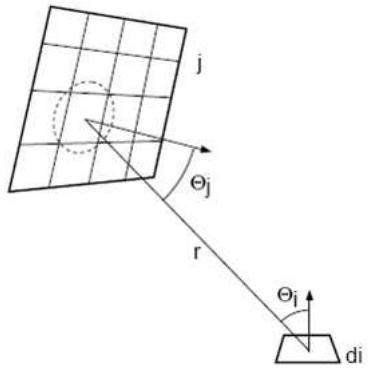


Figura 6.3: Aproximação do fator de forma utilizando discos

O *fragment shader* 6.0.3 calcula o fator de forma entre o texel do retalho receptor (área infinitesimal) e *shooter* (área finita) usando esta equação. Essa equação se ajusta bem às arquiteturas paralelas de hardware gráficos atuais, pois, não há a dependência entre dados.

Seleção do Shooter

O último passo de uma iteração do algoritmo é a seleção do próximo shooter. Cohen [6] observa que a convergência do algoritmo de radiosidade por refinamento progressivo converge mais rapidamente se a energia luminosa for distribuída de forma ordenada, ou seja, o retalho mais brilhante deve lançar energia antes que todos os

Shader 6.0.3 Fragment shader para calcular o fator de forma. ShootPos, ShootNormal, ShootEnergy e ShootDArea são fornecidos pela aplicação para cada retalho. Já RecvPos, RecvNormal são fornecidos pelo vertex shader por interpolação. ShootPos, ShootNormal,RecvPos e RecvNormal devem estar em coordenadas do mundo.

```
//Posicao projetada do ponto de vista do shooter
varying vec4 ProjPos;
// O prefixo wc significa que a variavel
// estah definida
// no sistema de coordenadas do mundo
varying vec3 wcRecvPos;
varying vec3 wcRecvNormal;



uniform vec3 wcShootPos;
uniform vec3 wcShootNormal;
uniform vec4 ShootEnergy;
uniform float ShootDArea;
// Reflexividade do patch
uniform vec4 RecvAlbedo;
uniform vec4 RecvID;

uniform sampler2D ItemBuffer;

const float PI= 3.14159265358979323846264338327 95;

float Visible(){
   vec4 proj= normalize(ProjPos);
   //Passa do intervalo [-1,1] para [0,1]
   proj.xy= proj.xy * 0.5 + 0.5;
   vec4 xtex= texture2D(ItemBuffer, proj.xy);
   return xtex == RecvID?1.0:0.05;
}

void main(){
   vec3 r= wcShootPos - wcRecvPos;
```

outros, em seguida o segundo mais brilhante e assim por diante. Deste modo, o próximo shooter deve ser o retalho receptor com maior energia luminosa.

Para empregar essa estratégia usando as capacidades do hardware gráfico, cada retalho receptor é renderizado em um framebuffer 1x1 usando um *fragment shader*. Este *fragment shader* calcula a energia residual (energia x area), coloca o inverso deste valor no z-buffer e a ID do retalho no backbuffer. Após renderizar todos os retalhos, o z-buffer automaticamente seleciona o próximo shooter que terá ID igual ao valor no backbuffer. A energia média é obtida gerando na camada da aplicação um mipmap da textura residual. O topo da piramide mipmap, que consite na energia residual média, é lido pelo *fragment shader*.

Capítulo 7

Decomposição Cell-Portal

O conceito de decomposição cell-portal foi concebido para tirar proveito de um tipo especial de cena conhecida como cenário interno ou *indoor environment*. O método pertence a classe de algoritmos de *culling*, isto é, ele se preocupa em renderizar somente a geometria visível ao observador.

Inicialmente, a cena é decomposta em células que são ligadas por portais. Uma cena típica seria uma casa, onde as células correspondem aos comodos e os portais as janelas e portas. As células e os portais formam um grafo orientado onde as células são os nós e os portais os arcos do grafo.

Como ilustração observe a cena na figura 7.1 e o correspondente grafo na figura 7.2 . A imagem contém 3 células A, B, C e dois portais. O desenho da cena inicia na célula A onde está o observador. Como o observador está olhando diretamente para um portal, a rotina de desenho é propagada para a célula adjacente B que por sua vez é propagada para a célula C. Não havendo mais células adjacentes, renderizamos a célula C com o volume de visão definido pela posição do observador e contorno do portal entre B e C. Em seguida, a célula B é renderizada com o volume de visão definido pelo contorno do portal entre A e B. E por fim a célula A é renderizada. Note que o processo de desenho é realizado de forma ordenada.

O diagrama de classes apresentado na figura 7.3 mostra a estrutura necessária para uma implementação. A classe CellMgr é o ponto de entrada da decomposição. Ele contém o conjunto de células que forma a decomposição. O método *Draw* (figura 7.4) renderiza toda a decomposição utilizando o algoritmo de decomposição célula-

portal (**cell-portal**). O método GetContainingCell é utilizado para obter a célula que contém o observador.

Cada célula é representada pela classe *Cell*. O método *DrawGeometry* renderiza a geometria da célula e *Draw* (figura 7.5) renderiza recursivamente as células adjacentes empregando os portais.

O portal representado pela classe *Portal* possui uma referência a célula adjacente. O método *Draw* (figura 7.6) renderiza a célula adjacente se o portal for visível ao observador. Nessa etapa, a célula adjacente é renderizada com o volume de visão reduzido.

O volume de visão está contido na classe *Camera*. O método *Push* adiciona um plano de corte adicional ao volume de visão e *Pop* retira um plano da pilha. O método *Culled* é usado para determinar se um portal é visível ao observador. Tal método retorna verdadeiro se o portal não for visível.

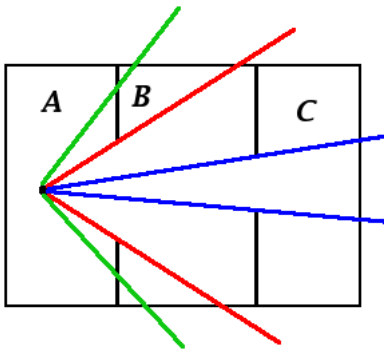


Figura 7.1: Ilustração de uma cena com celulas e portais.

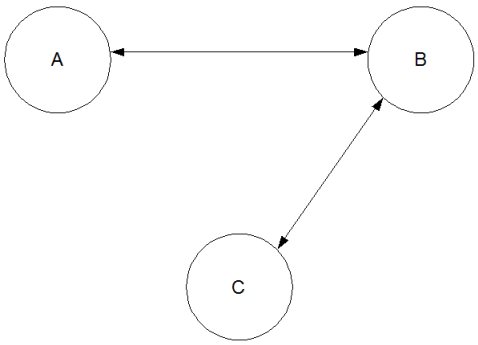


Figura 7.2: Grafo da cena com celulas e portais.

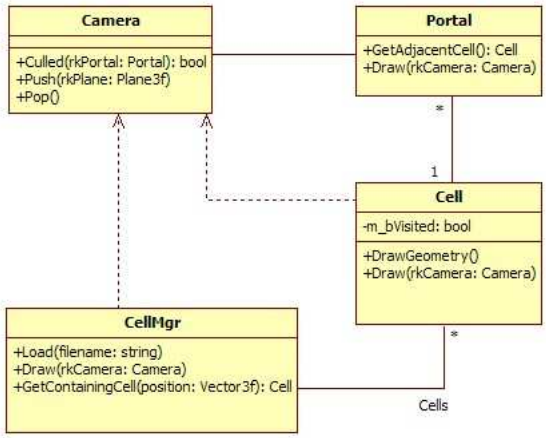


Figura 7.3: Diagrama com estrutura de objetos da decomposicao Cell-Portal.

```
CellMgr::Draw(rkCamera:Camera){
  Cell *CurrentCell;
  CurrentCell=GetContainingCell(rkCamera.position);
  If(CurrentCell)
    CurrentCell->Draw(rkCamera);
}
```

Figura 7.4: Renderiza a decomposição célula-portal por meio de um percurso na estrutura de grafo.

```
Cell::Draw(rkCamera:Camera){
  if(!m_bVisited){
    m_bVisited= true;
    for each portal p do
      p->Draw(rkCamera);
    DrawGeometry();
    m_bVisited= false;
  }
}
```

Figura 7.5: Renderização da célula

```
Portal::Draw(rkCamera:Camera){
  if(rkCamera.Culled(this)) return;
  Adicionar_Planos_Pilha();
  GetAdjacentCell()->Draw(rkCamera);
  Retirar_Planos_Pilha();
}
```

Figura 7.6: Renderiza célula adjacente por meio do portal.

Capítulo 8

Radiosidade integrada a decomposição Cell-Portal

O método de radiosidade é capaz de gerar imagens de alta qualidade para ambientes difusos. No entanto, as potenciais interações de um patch com o ambiente tornam o método extremamente caro. No método clássico, há um grande volume de interações que nunca ocorreram, como por exemplo, dois patchs com um oclusor entre eles. Embora o metodo trate corretamente o caso, o simples fato de testar a visibilidade entre os dois patchs adiciona um custo alto a simulação. Uma decomposição Cell-Portal simplifica o ambiente e possibilita uma redução do número da potenciais interações.

Em nossa abordagem, cada célula é considerada uma unidade isolada que interage entre si através de portais. Na interação entre duas células consideramos dois casos possíveis a incidência da luz direta de uma célula sobre a célula adjacente e a incidência da luz indireta sobre a célula adjacente. Essa distinção é feita, pois, a luz direta possui intensidade mais significativa que a indireta.

O algoritmo é apresentado na figura 3. O processo de iluminação começa com a distribuição da luz direta em cada célula e adjacências. Essa etapa inicializa a energia residual e radiosidade de todos os patches da cena. Ou seja, quando a simulação da radiosidade iniciar, quase todos os patches estão com energia diferente de zero, diferente do método clássico que inicia com os patches com energia zerada. Após inicializar a energia de cada patch, os patchs fonte de luz são eliminados do

processo de iluminação. Essa decisão foi tomada, pois, a energia da fonte de luz geralmente será sempre maior que a energia indireta que pode alcançá-la.

Em seguida, a iluminação indireta de cada célula é calcula empregando o algoritmo de radiosidade em GPU apresentado na seção 6. Nesse momento, os portais são considerados como demais patches da geometria da célula. A energia acumulada no portal consiste na maior energia dos *shooters*. Cada portal, somente recebe energia indireta que é acumulada no interior do portal para a etapa posterior. No fim há um passo de sincronização que consiste na distribuição da energia acumulada nos portais para as células adjacentes. De fato, os portais são tratados como fontes de luz artificiais.

Note que o algoritmo apresenta algumas características interessantes. A primeira é a natureza paralela do método. O fato da simulação de uma célula independer da simulação das células vizinhas possibilita a execução concorrente das simulações.

A segunda característica possibilitaria o cálculo da radiosidade por demanda. Em vez de calcular a radiosidade em um primeiro momento para todas as células, determinaríamos um conjunto de célula baseado na visibilidade do observador, ou em um outro critério, e calculamos a iluminação para essas células. A medida que o observador percorrer o ambiente, nos adicionaríamos ou removeríamos células ao conjunto de trabalho. Essa estratégia tem a vantagem de economizar processamento, pois, estaremos trabalhando com um grupo reduzido de informações e além de economizar uma quantidade significativa de memória, pois, as texturas das células fora do grupo não necessitariam ser alocadas.

Observe que a idéia de tratar os portais como fonte de luz artificiais não se baseia em princípios físicos. Contudo, podemos pensar no processo de sincronização como uma medida paliativa e imediata para a visualização da imagem, semelhante a componente ambiente apresentada no algoritmo de radiosidade progressiva [5].

A estrutura principal da implementação do algoritmo é a estrutura do retalho no fragmento de código 8.0.1. Os lightmaps para armazenar a radiosidade e a energia residual são referenciados pelos ponteiros radiosity e residual possuindo as dimensões 16x16, 32x32 etc, sendo os valores armazenados em ponto flutuante. A reflectividade do retalho é guardada na variável albedo que possui um valor para cada uma das

componentes R, G,B. As variáveis normal, vertex e indx descrevem a geometria do patch. A variável id contém a identificação de cada retalho e é utilizada no teste de visibilidade. As identificações são passadas para os shaders através da conversão do inteiro para as componentes RGBA por meio da operação *shift*.

A implementação do algoritmo de Radiosidade com Cell-Portal é representada pelo fragmento de código 8.0.2. Nesse código, um número pré-definido de iterações foi utilizado como teste de convergência. A função SimulateLightTransport calcula a radiosidade utilizando a GPU para a i-ésima célula. A estrutura do retalho é utilizada dentro desta função.

O teste de visibilidade gera artefatos nas arestas dos retalhos devido à projeção estereográfica. A solução utilizada para minimizar os artefatos foi a realização de uma amostragem de pontos na vizinhança do texel a ser testado como apresentado no fragmento de código 8.0.4. Foi utilizada uma vizinhança 3x3 que foi passada para o shader através da variável *offset*. Esse código retorna verdadeiro caso uma das identificações das vizinhanças seja igual a identificação do retalho.

Algoritmo 3 Pseudo-código para o algoritmo de Radiosidade com Cell-Portal

1: Crie uma decomposição Cell-Portal da cena.
2: **Para** cada célula da decomposição **faça**
3: Lançar luz direta na célula.
4: **Fim Para**
5: **Enquanto** não converge **faça**
6: **Para** cada celula da decomposição **faça**
7: Simular o transporte de luz na célula com o algoritmo de Radiosidade
8: **Fim Para**
9: **Para** cada celula da decomposição **faça**
10: Lançar a energia média dos Portais para as respectivas células adjacentes
11: **Fim Para**
12: **Fim Enquanto**

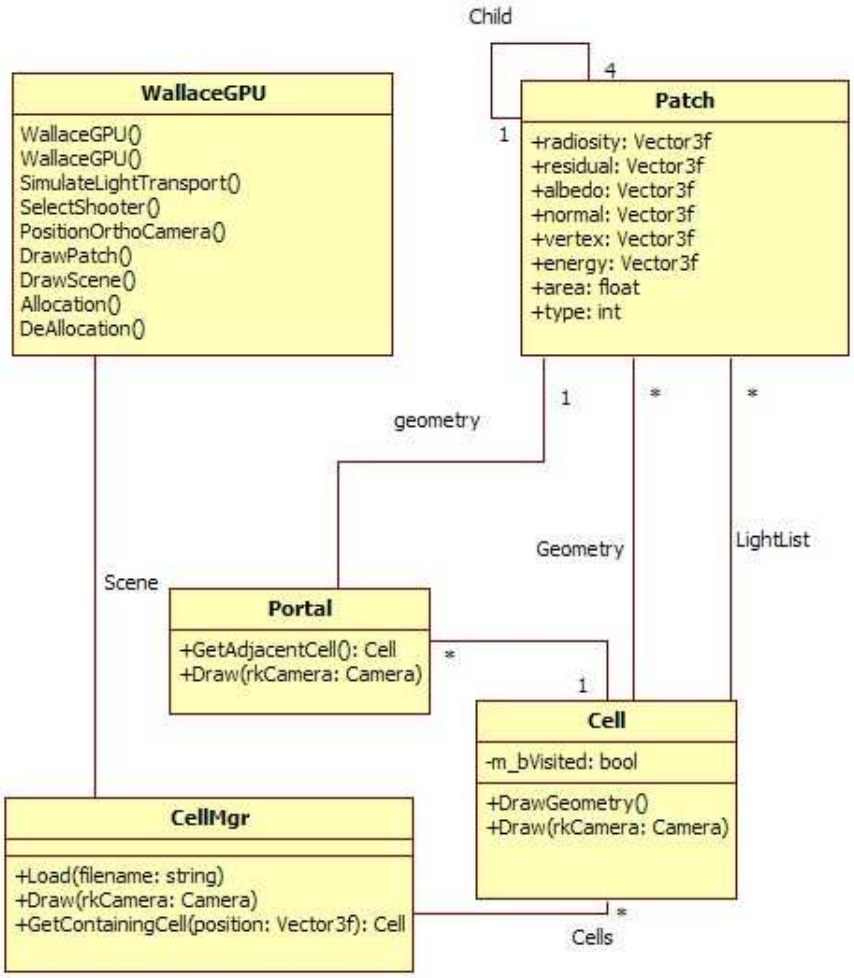


Figura 8.1: Diagrama com estrutura de objetos para o algoritmo de radiosidade com decomposicao Cell-Portal.

Código 8.0.1 Estrutura do retalho

```
class Patch {
public:
    Vector3f *radiosity;
    Vector3f *residual;
    Vector3f albedo;
    Vector3f normal;
    Vector3f *vertex;
    Vector3f energy;
    float area;
    int indx[4];
    int id;
    static int num_objects;
    void calculateArea();
    void calculateNormal();
};
```

Código 8.0.2 Fragmento de código para a simulação

```
void WallaceGPU::SimulateLightTransport() {
    int i;
    static int convergence=2;

    for (i=0; i < Scene.Cells.size() ; ++i)
        ShootDirectLight(i);

    while (convergence--) {
        for (i=0;  i < Scene.Cells.size();++i)
            SimulateLightTransport(i);
        for (i=0;  i < Scene.Cells.size();++i)
            ShootPortalsEnergy(i);
    }
}
```

Shader 8.0.4 Teste de Visibilidade para minimizar artefatos nas bordas dos retalhos.

```
uniform vec2 offset[9];
float Visible() {
    vec3 proj= normalize(ProjPos.xyz);
    proj.xy= proj.xy * 0.5 + 0.5;//Passa do intervalo [-1,1] para [0,1]
    vec4 xtex;
    for (k=0; k < 9; ++k) {
        xtex= texture2D(ItemBuffer, proj.xy + offset[k]);
        if (xtex == RecvID)
            return 1.0;
    }
    return 0.0;
}
```

Capítulo 9

Resultados

Os experimentos foram realizados em um computador com processador AMD Athlon 64 2.0GHz, L2 cache de 512KB, 512MB de memória RAM e placa de vídeo NVidia Geforce 6200 com 128MB de memória . O programa foi implementado em c++ e a biblioteca SDL foi utilizada como ponte entre o sistema de janelas e a API OpenGL. A versão do OpenGL foi 2.0 e a linguagem GLSL foi utilizada para a programação dos *shaders*.

Todas as cenas são constituidas por quadriláteros retangulares. A estrutura Cell-Portal foi obtida manualmente empregando o modelador Blender. O albedo de todos os polígonos de uma determinada célula é atribuído de forma aleatória. As texturas utilizadas possuem resolução de 16x16, 32x32, 64x64, 128x128 e o item buffer com resolução 512x512.

A primeira cena (cena 1) apresentada nas figuras 9.1, 9.2, 9.3 é formada por três células , dois portais e duas fontes de luz. Duas das células possuem fonte de luz de 100 unidades cada. Essa cena não possui oclusores dentro das células. Ela mostra a iluminação de uma célula sem fonte de luz por iluminação indireta.

A cena 2 apresentada na figura 9.4, 9.5, 9.6, 9.7, 9.8, 9.9, 9.10, 9.11 é semelhante à anterior só que com a inclusão de oclusores. Novamente, a célula do meio não possui fonte de luz.

A última cena (cena 3) mostrada nas figuras 9.12, 9.13, 9.14, 9.15, 9.16, 9.17 possui três células, dois portais e duas fontes de luz. Sendo que uma célula contem uma oclusão dada por um conjunto de colunas. A célula do meio também foi refinada.

Observe que a sombra projetada na célula sem fonte de luz possui artefatos como buracos. Note também que esses artefatos surgem na parte superior da parede. Esse problema aparece possivelmente devido a uma singularidade no momento da determinação da visibilidade. Como estamos projetando em uma esfera, a medida que nos aproximamos nas extremidades da esfera, a projeção se torna cada vez menor. Isso aumenta as chances de erro, pois, estaremos consultando um conjunto reduzido de pixels no ItemBuffer.

A tabela 9.1 apresenta o tempo necessário para calcular a radiosidade de cada cena empregando o algoritmo integrado à decomposição célula-portal e sem decomposição célula-portal. Em ambos os algoritmos foi utilizado o número de iterações como critério de convergência. No algoritmo integrado à decomposição célula-portal foram utilizados 2 iterações para a convergência no calculo das células separadas e 2 iterações na distribuição da radiosidade na célula. Na convergência do teste sem considerar a decomposição célula-portal foram utilizados 4 iterações mais 2 iterações para compensar as interações entre as salas.

A tabela 9.2 apresenta a relação entre a resolução das texturas e o número de quadriláteros de cada cena. Observe o numero reduzido de polígonos e o resultada final da radiosidade. As cena são constituídas basicamente por quadriláteros grandes, por exemplo, uma parede inteira é composta por um único quadrilátero. Caso a radiosidade fosse determinada por vértice, haveria a necessidade de subdividir os polígonos para haver uma correta interpolação por Gouraud.

Cena	Resolução da textura	Tempo(s) com CPG	Tempo(s) sem CPG
cena 1	16x16	0.879	1.127
cena 1	32x32	0.958	1.341
cena 1	64x64	1.497	2.662
cena 1	128x128	5.366	11.871
cena 2	16x16	1.570	2.362
cena 2	32x32	1.964	2.735
cena 2	64x64	3.281	5.834
cena 2	128x128	11.239	26.875
cena 3	16x16	27.858	31.189
cena 3	32x32	31.196	36.272
cena 3	64x64	51.092	77.298
cena 3	128x128	–	–

Tabela 9.1: Tempo das simulações

Cena	Resolução da textura	# quadriláteros
cena 1	16x16	42
cena 1	32x32	42
cena 1	64x64	42
cena 1	128x128	42
cena 2	16x16	90
cena 2	32x32	90
cena 2	64x64	90
cena 2	128x128	90
cena 3	16x16	1224
cena 3	32x32	1224
cena 3	64x64	1224
cena 3	128x128	1224

Tabela 9.2: Número de quadriláteros das cenas

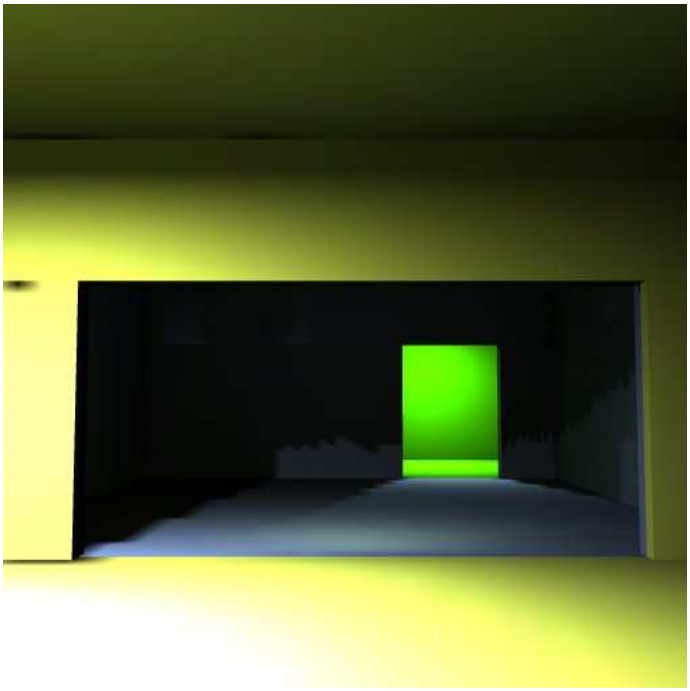


Figura 9.1: Imagem gerada para a cena 1 com textura 32x32

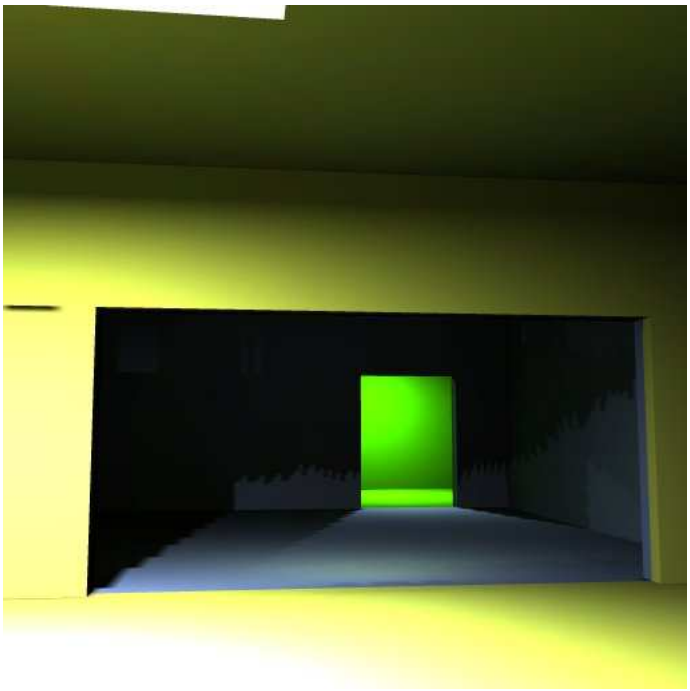


Figura 9.2: Imagem gerada para a cena 1 com textura 64x64

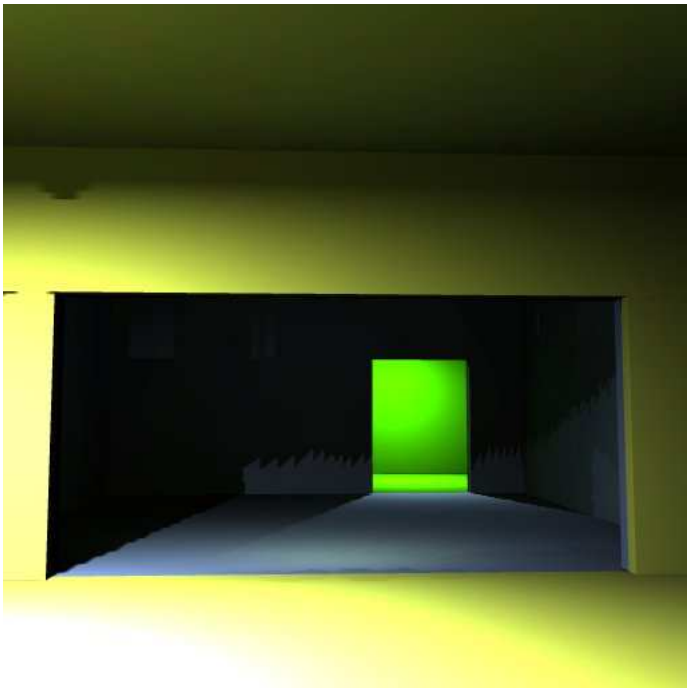


Figura 9.3: Imagem gerada para a cena 1 com textura 128x128

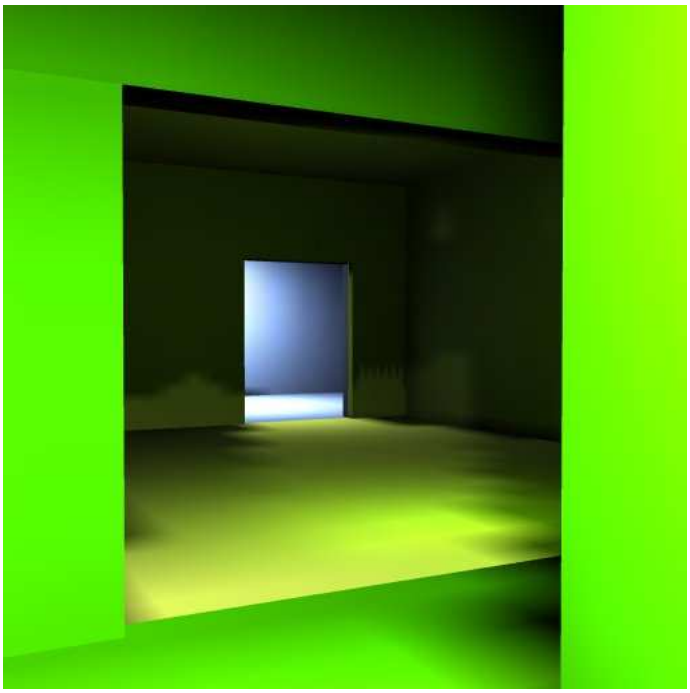


Figura 9.4: Imagem gerada para a cena 2 com textura 16x16

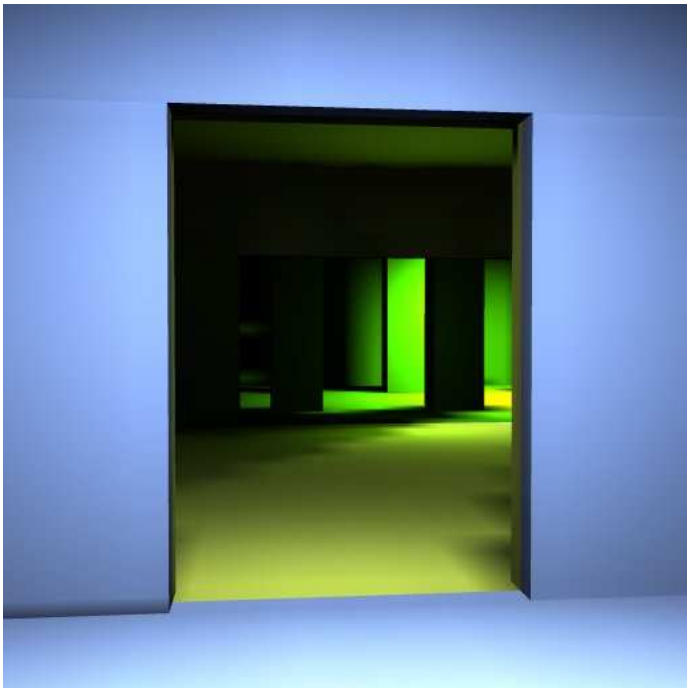


Figura 9.5: Imagem gerada para a cena 2 com textura 16x16

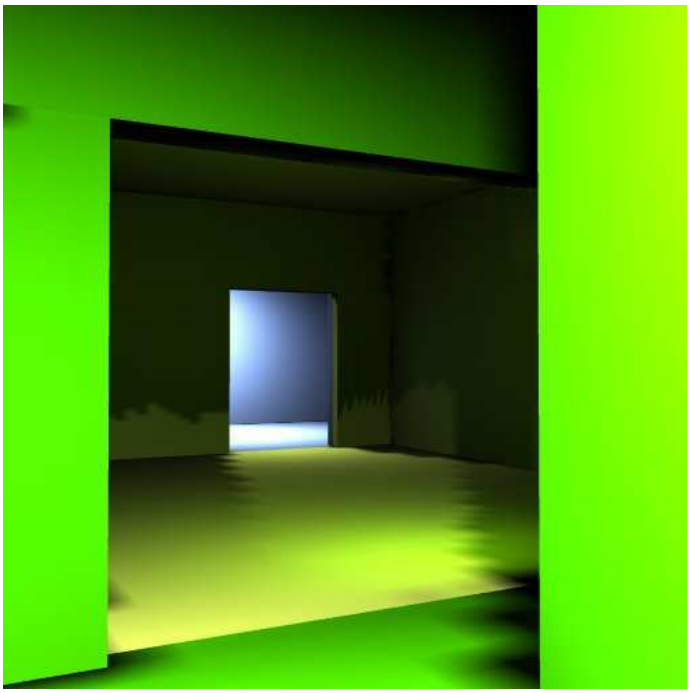


Figura 9.6: Imagem gerada para a cena 2 com textura 32x32

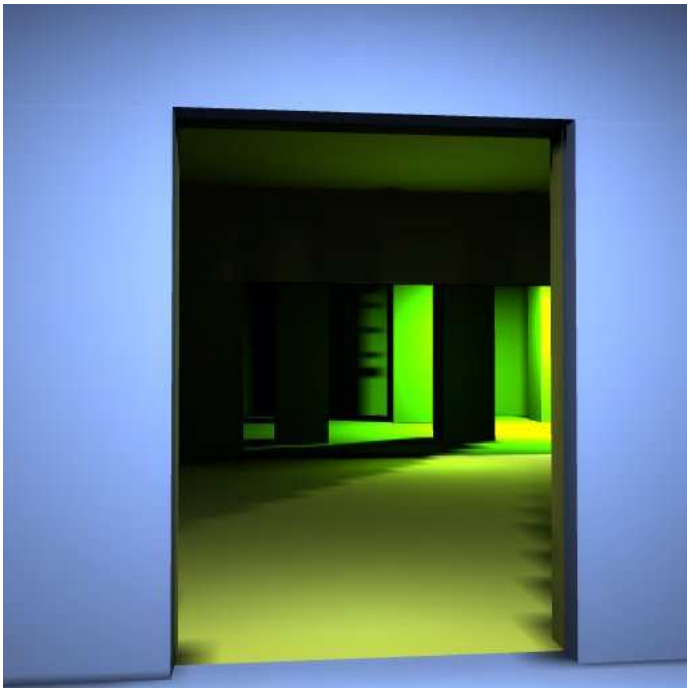


Figura 9.7: Imagem gerada para a cena 2 com textura 32x32

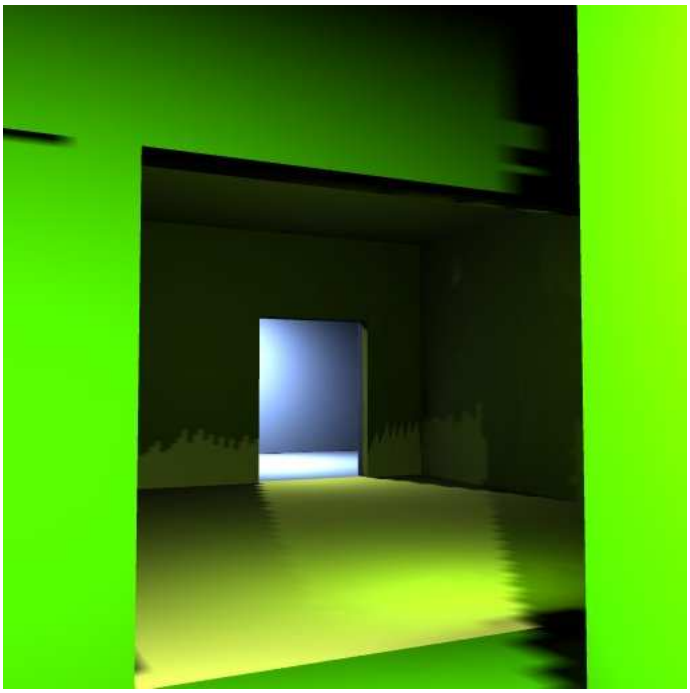


Figura 9.8: Imagem gerada para a cena 2 com textura 64x64

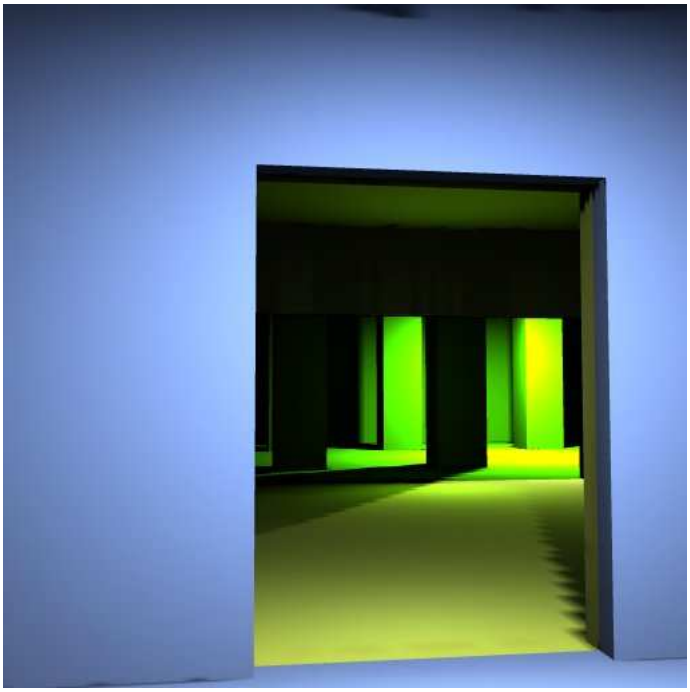


Figura 9.9: Imagem gerada para a cena 2 com textura 64x64

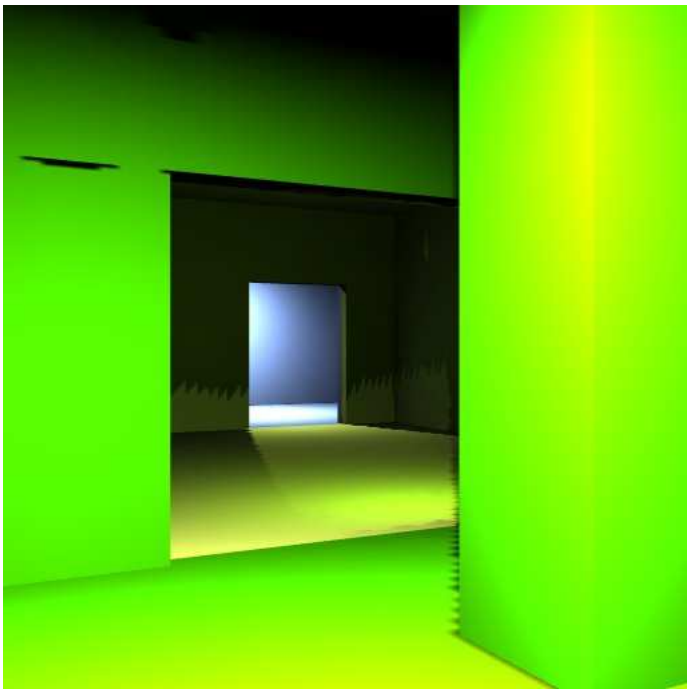


Figura 9.10: Imagem gerada para a cena 2 com textura 128x128

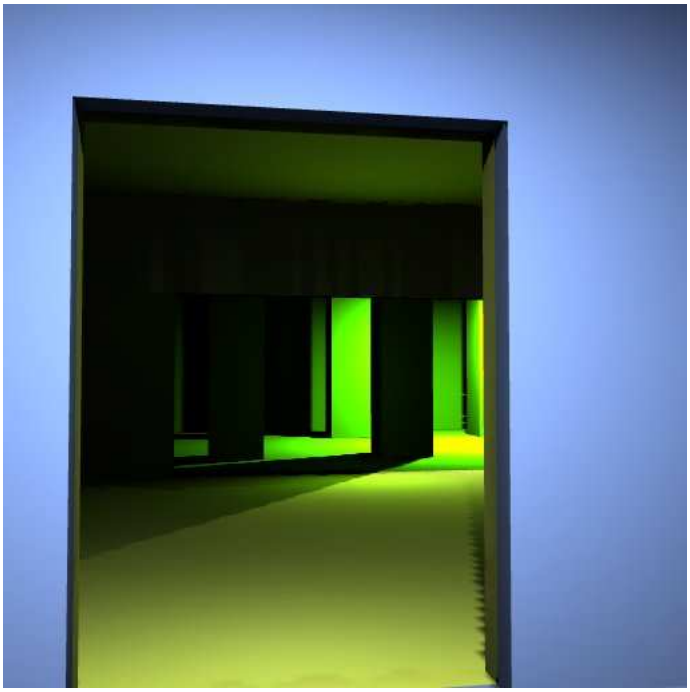


Figura 9.11: Imagem gerada para a cena 2 com textura 128x128

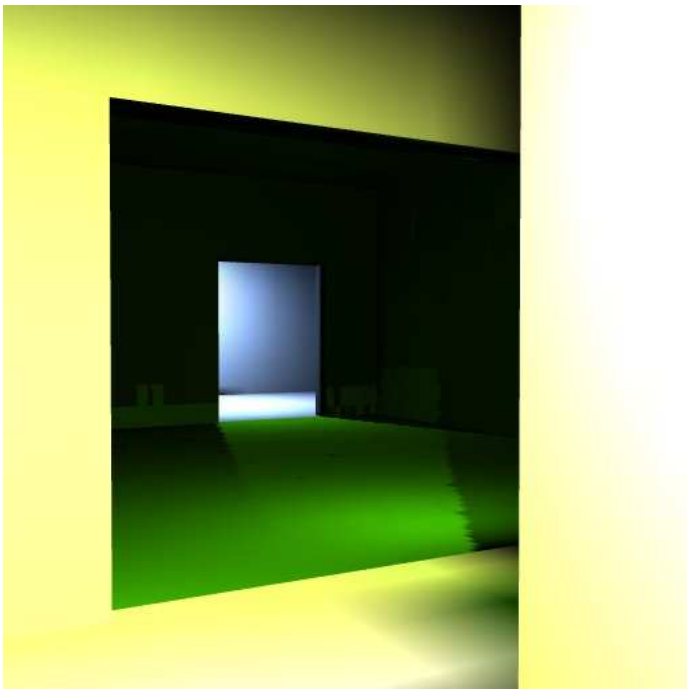


Figura 9.12: Imagem gerada para a cena 3 com textura 16x16

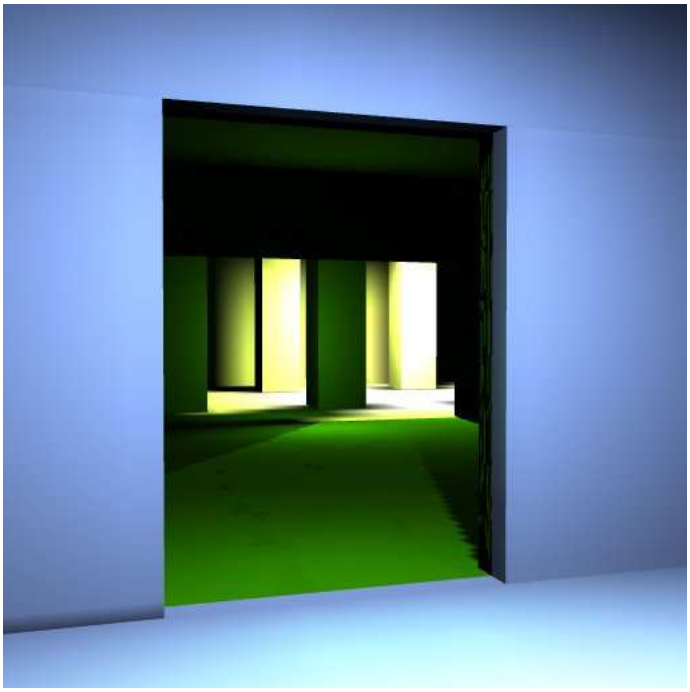


Figura 9.13: Imagem gerada para a cena 2 com textura 16x16

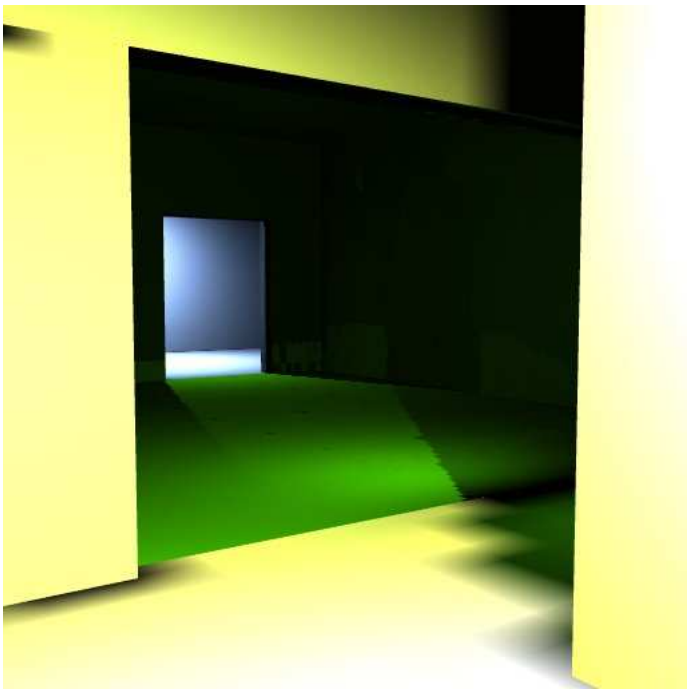


Figura 9.14: Imagem gerada para a cena 3 com textura 32x32

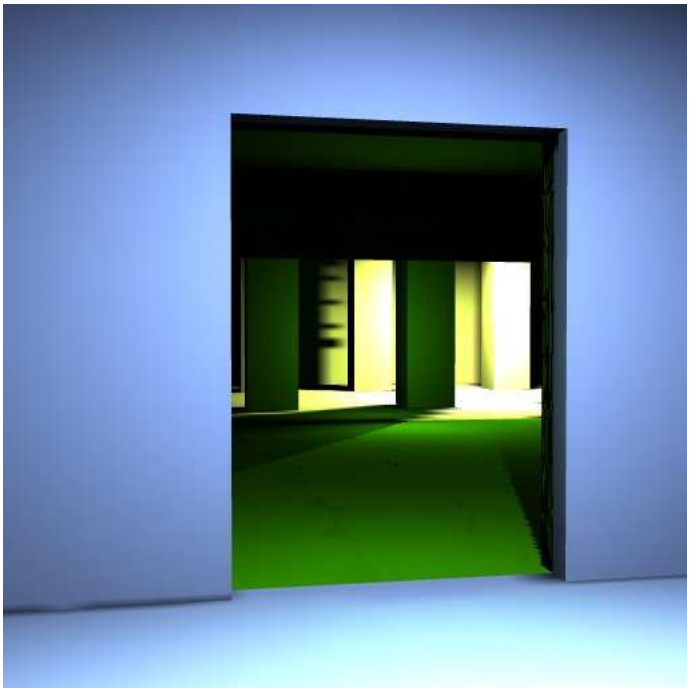


Figura 9.15: Imagem gerada para a cena 2 com textura 32x32

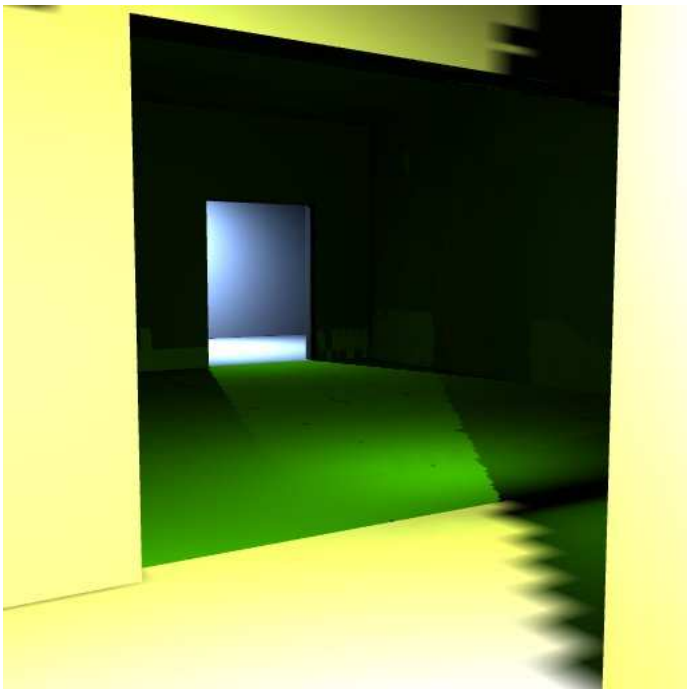


Figura 9.16: Imagem gerada para a cena 2 com textura 64x64

Figura 9.17: Imagem gerada para a cena 2 com textura 64x64

Capítulo 10

Conclusão

Neste trabalho foi apresentado um método de iluminação global baseado em uma decomposição Célula-Portal. A grande vantagem do método é a capacidade de determinar a iluminação global de superfícies difusas considerando interações em um número reduzido de objetos geométricos da cena.

O algoritmo inicia com a criação de uma decomposição Célula-Portal manual da cena. Em seguida, os valores de radiosidade de cada célula são inicializados com o lançamento da energia das fontes de luz. A partir daí, desconsideramos as fontes de luz na simulação.O algoritmo de radiosidade é aplicado em cada célula da decomposição, sem considerar as células vizinhas. Então um passo de sicronização é realizado, lançando as energias acumuladas nos portais.

Nos primeiros testes, os portais foram considerados como retalhos com texturas *residual* e *radiosity*. Isto é, se estamos trabalhando com texturas 16x16, teríamos para cada portal duas texturas 16x16. No passo de sincronização, a energia média das texturas eram calculadas e essa média era lançada para a célula adjacente. A desvantagem dessa estratégia está na necessidade de armazenar as texturas e o cálculo da média a cada passo da sicronização. Além disso o acumulo da energia nos portais requer o uso de *shaders*.

Para evitar o armazenamento adicional exigido pela estratégia anterior, representou-se a energia do shooter através dos 3 valores dos canais RGB. A energia do shooter é acumulada no portal, visto que será sempre a mais significativa.

Pelos testes realizados, observamos que as imagens são consistentes apesar do

método não ser baseado em princípios físicos. O método pode ser utilizado como uma solução intermediária, caso a acurácia seja um fator determinante ou uma solução aproximada.

No futuro, pode-se implementar uma solução inteiramente em GPU. Ou seja, toda a simulação e o processo de *rendering* seriam realizado em GPU. Na atual implementação não foi incluida nenhuma otmização como o recurso de *rendering to texture*, atlas de textura ou buffer da geometria da cena na memória da GPU. Uma estratégia para o gerenciamento do atlas de textura, seria considerar um atlas para cada célula da cena. Desta maneira, o atlas da célula e os atlas das células adjacentes ficariam armazenadas na memória do hardware gráfico. Um algoritmo de política de cache poderia ser utilizado para gerenciar os atlas a medida que o *rendering* das células progredisse.

Outro recurso ainda não implementado é a determinação da iluminação por demanda. No estado atual da implementação, a iluminação é determinada para todas as células. A idéia do cálculo por demanda, seria calcular a iluminação na célula que contém o observador e, em seguida, determinar as células adjacentes visíveis e calcular a iluminação. Esta estratégia reduziria bastante o tempo do cálculo, pois, o observador tende a ficar vários frames em uma mesma célula.

Uma implementação utilizando algum tipo de paralelismo, como multiplas threads, também pode ser incorporado na implementação. O cálculo da iluminação em cada célula pode ser realizado de modo independente em multiplas *threads*. Contudo, esta solução pode trazer dificuldade uma vez que o acesso ao hardware gráfico seria compartilhado, exigindo um esforço adicional para a sincronização das threads. Como o algoritmo utilizado faz uso intenso da GPU, tal estratégia poderia degradar a performance geral do sistema. No entanto, uma estratégia multi-GPU poderia produzir bons resultados.

Referências Bibliográficas

[1] Otávio P. F. Braga. Uma arquitetura para síntese de imagens fotorrealistas baseada em técnicas de monte carlos. Master's thesis, 2006.

[2] Edmundo Capelas and Martin Tygel. *Métodos Matemáticos para a Engenharia*, volume 1. Sociedade Brasileira de Matemárica Aplicada e Computacional, 2001.

[3] Derek W. Carr. Computation of potentially visible set for occluded three-dimensional environments, 2004.

[4] N. Carr, J. Hall, and J. Hart. The ray engine, 2002.

[5] Michael F. Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A progressive refinement approach to fast radiosity image generation. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 75–84, New York, NY, USA, 1988. ACM Press.

[6] Michael F. Cohen and Donald P. Greenberg. The hemi-cube: a radiosity solution for complex environments. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 31–40, New York, NY, USA, 1985. ACM Press.

[7] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 137–145, New York, NY, USA, 1984. ACM Press.

[8] Greg Coombe and Mark Harris. Global illumination using progressive refinement radiosity. In Matt Pharr, editor, *GPU Gems 2*, chapter 39, pages 635–647. Addison Wesley, March 2005.

[9] O. Debeir D. Haumont and F.Sillion. Volumetric cell-and-portal generation. In *EUROGRAPHICS 2003*, pages 303–312, USA, 2003. The Eurographics Association and Blackwell.

[10] Chen-Chin Feng and Shi-Nine Yang. A parallel hierarchical radiosity algorithm for complex scenes. In *PRS '97: Proceedings of the IEEE symposium on Parallel rendering*, pages 71–ff., New York, NY, USA, 1997. ACM.

[11] Jonas Gomes and Luiz Velho. *Fundamentos da Computação Gráfica*. IMPA, 2003.

[12] Steven J. Gortler, Michael F. Cohen, and Phillipp Slusallek. Radiosity and Relaxation Methods: Progressive Refinement is Southwell Relaxation. Technical Report CS-TR-408-93, Princeton, NJ, 1993.

[13] Robin Green. Spherical harmonic lighting: The gritty details. Technical report, Sony Computer Entertainment America, 2003.

[14] Toshiya Hachisuka. High-quality global illumination rendering using rasterization. In Matt Pharr, editor, *GPU Gems 2*, chapter 38, pages 615–633. Addison Wesley, March 2005.

[15] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 197–206, New York, NY, USA, 1991. ACM Press.

[16] James T. Kajiya. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, New York, NY, USA, 1986. ACM Press.

[17] G. C. M. J. H. A LASTRA. Radiosity on graphics hardware. graphics interface. 2004.

[18] Sylvain Lefebvre and Samuel Hornus. Automatic cell-and-portal decomposition, 2003.

[19] Nelson L. Max and Elizabeth D. Getzoff. Spherical harmonic molecular surfaces. *IEEE Computer Graphics and Applications*, 08(4):42–50, 1988.

[20] Tomas Akenine Moller and Eric Haines. *Real-Time Rendering*. A.K. Peters Ltd., 2002.

[21] Samuel Ranta-Eskola. Binary space partitioning trees and polygon removal in real time 3d rendering, 2001.

[22] Rodney J. Recker, David W. George, and Donald P. Greenberg. Acceleration techniques for progressive refinement radiosity. In *SI3D '90: Proceedings of the 1990 symposium on Interactive 3D graphics*, pages 59–66, New York, NY, USA, 1990. ACM Press.

[23] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 527–536, New York, NY, USA, 2002. ACM Press.

[24] Roy Troutman and Nelson L. Max. Radiosity algorithms using higher order finite element methods. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 209–212, New York, NY, USA, 1993. ACM Press.

[25] John R. Wallace, Michael F. Cohen, and Donald P. Greenberg. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 311–320, New York, NY, USA, 1987. ACM Press.

[26] P. Wesseling. *An Introduction to Multigrid Methods*. R.T. Edwards,Inc., 2004.

[27] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, 1980.

[28] Harold R. Zatz. Galerkin radiosity: a higher order solution method for global illumination. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 213–220, New York, NY, USA, 1993. ACM Press.

Capítulo 11

Integração de Monte-Carlo

Os métodos de Monte Carlo constituem uma vastas classe de algoritmos computacionais numéricos para a simulação do comportamentos de sistemas físicos e matemáticos. Tais métodos são úteis na modelagem de fenômenos com alto grau de incerteza de suas entradas, como o cálculo de risco de um investimento. Neste capítulo, estamos interessados no método de integração.

Desejamos resolver numericamente a integral

$$I = \int_a^b f(x)dx \tag{11.1}$$

Considerando x como uma variável aleatória temos

$$E[f(x)] = \int_a^b f(x)p(x)dx$$

onde $p(x)$ é uma função densidade de probabilidade.

Pela lei dos grandes números, para as amostras $x_1, x_2, \cdots, x_n$ uma aproximação do valor esperado é dada por

$$E[f(x)] \approx \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

Combinando os resultados anteriores obtemos uma aproximação da integral

$$I = \int_a^b f(x)dx = E\Big[\frac{f(x)}{p(x)}\Big] = \int_a^b \frac{f(x)}{p(x)}p(x)dx \approx \frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{p(x_i)}$$

Assim o estimador $I_n = \frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{p(x_i)}$ coincide com I 11.1 quando n tende para

o infinito. A variância do estimador é

$$
\begin{aligned}
V[I_n] &= V\Big[\frac{1}{n}\sum_{i=1}^{n}\frac{f(x_i)}{p(x_i)}\Big] \\
&= \frac{1}{n^2}\sum_{i=1}^{n} V\Big[\frac{f(x_i)}{p(x_i)}\Big] \\
&= \frac{1}{n} V\Big[\frac{f(x)}{p(x)}\Big] \\
&= \frac{1}{n}\left(E\Big[\Big(\frac{f(x)}{p(x)}\Big)^2\Big] - E\Big[\frac{f(x)}{p(x)}\Big]^2\right) \\
&= \frac{1}{n}\left(\int_a^b \Big(\frac{f(x)}{p(x)}\Big)^2 p(x)dx - I^2\right) \\
&= \frac{1}{n}\left(\int_a^b \frac{(f(x))^2}{p(x)}dx - I^2\right)
\end{aligned}
$$

Uma escolha apropriada da função de densidade de probabilidade $p(x)$ pode reduzir consideravelmente a variância do estimador, resultando em uma melhor aproximação com um número menor de amostras.

O método apresentado se estende de modo natural para a integração de funções de várias variáveis considerando $x \in \mathbb{R}^n$. Outra vantagem do método é que não precisamos de nenhum conhecimento a priori da função $f(x)$. Só precisamos saber realizar uma boa amostragem e avaliar a função $f(x)$ nessas amostras. Esse fato é altamente desejável do ponto de vista do encapsulamento em um módulo ou objeto.

Capítulo 12

Polinômios de Legendre

Os polinômios de Legendre constituem uma classe de polinômios muito aplicada em física-matemática e engenharia. Eles surgem da equação diferencial ordinária

$$(1-x^2)y'' - 2xy' + p(p+1)y = 0$$

conhecida como equação de Legendre de ordem p. O parâmetro p pode ser real ou complexo.

As soluções $P_n(x)$ desta equação são os polinômios de Legendre (para um abordagem completa veja [2]) e podem ser escritos como

$$P_n(x) = \sum_{k=0}^{N} \frac{(-1)^k (2n-2k)!}{2^n k!(n-k)!(n-2k)!} x^{n-2k}$$

onde

$$N = \begin{cases} \frac{n}{2} & \text{, se n é par} \\ \frac{(n-1)}{2} & \text{, se n é impar} \end{cases}$$

Veja a seguir alguns exemplos dos polinômios de Legendre de ordem $n = 0, 1, 2, 3, 4, 5$ e seus respectivos gráficos 12

$$\begin{aligned}
P_0(x) &= 1 \\
P_1(x) &= x \\
P_2(x) &= \tfrac{1}{2}(3x^2 - 1) \\
P_3(x) &= \tfrac{1}{2}(5x^3 - 3x) \\
P_4(x) &= \tfrac{1}{8}(35x^4 - 30x^2 + 3) \\
P_5(x) &= \tfrac{1}{8}(63x^5 - 70x^3 + 15x)
\end{aligned}$$

Os polinômios de Legendre possuem algumas propriedades bastente úteis como uma relação de recorrência, que facilita a implementação, e a ortogonalidade.

Os polinômios $P_n(x)$ e $P_m(x)$ satisfazem as relações de ortogonalidade no intervalo $x \in (-1, 1)$

$$\int_{-1}^{1} P_n(x)P_m(x) = \begin{cases} 0 & \text{, para } m \neq n \\ \dfrac{2}{2n+1} & \text{, para } m = n \end{cases}$$

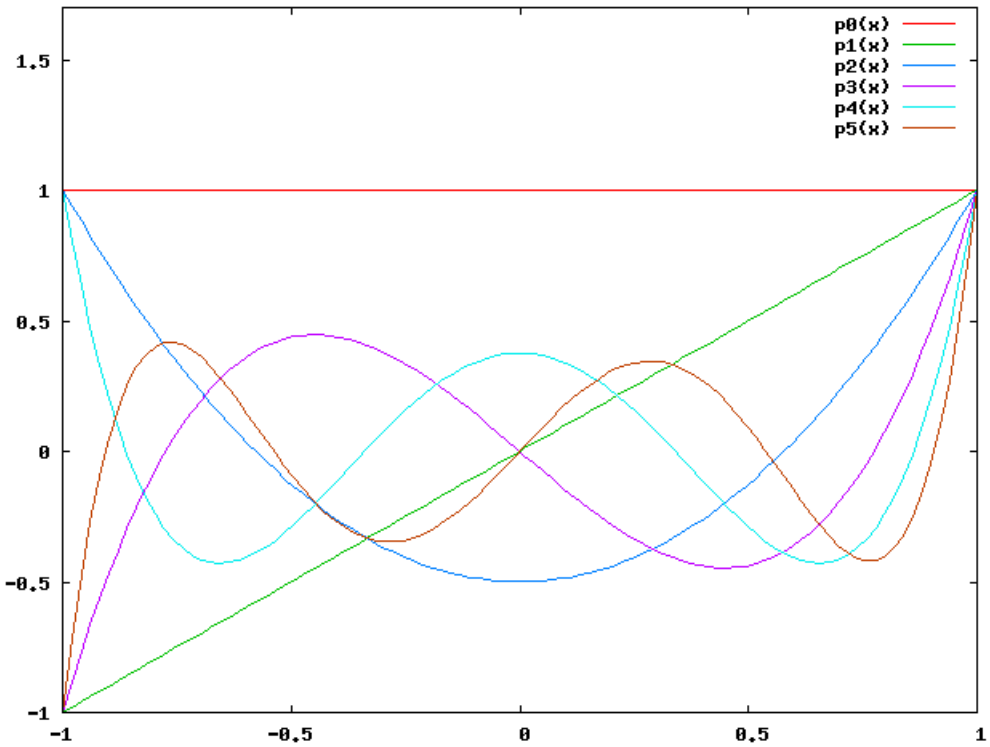


Figura 12.1: Polinômios de Legendre de ordem $n = 0, 1, 2, 3, 4, 5$

A relação de recorrência estabelece que se conhecemos $P_0(x)$ e $P_1(x)$ então podemos determinar $P_n(x)$ pela fórmula

$$(2n+1)xP_n(x) = (n+1)P_{n+1}(x) + nP_{n-1}(x)$$

Outra importante classe de polinômios são os polinômios associados de Legendre. Tais polinômios são as soluções da equação associada de Legendre

$$(1-x^2)y'' - 2xy' + \left[n(n+1) - \frac{m^2}{1-x^2}\right]y = 0$$

onde n e m são inteiros, $n \geq m$ e $n \geq 0$. Os dois argumentos n e m dividem4 a família de polinômios em bandas de funções onde n é o indice de banda e m um inteiro na faixa $[0, n]$.

Os polinômios associados também possuem relação de recorrêndia dada por

$$(n-m)P_n^m = x(2n-1)P_{n-1}^m - (n+m-1)P_{n-2}^m$$

Veja no gráfico a seguir os polinômios

$$\begin{aligned} P_1^1(x) &= (1-x^2)^{1/2} \\ P_2^1(x) &= 3x(1-x^2)^{1/2} \\ P_2^2(x) &= 3(1-x^2) \end{aligned}$$

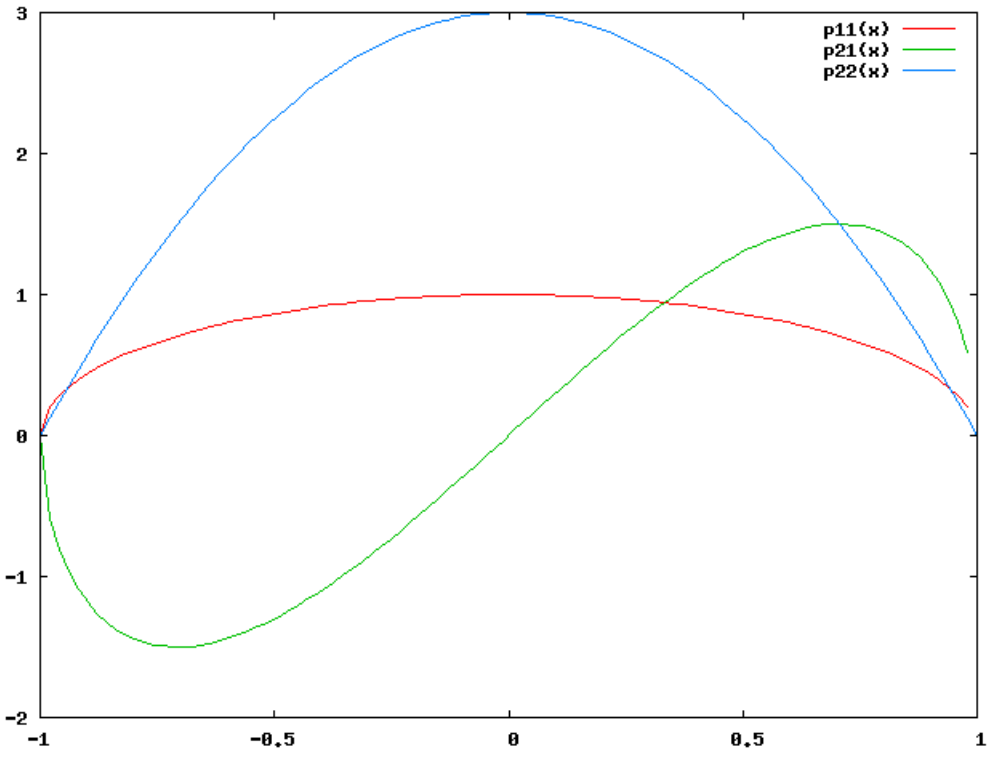


Figura 12.2: Polinômios associados de Legendre de ordem $n = 1, 2$ onde $p11(x) = P_1^1(x)$, $p21(x) = P_2^1(x)$, $p22(x) = P_2^2(x)$