UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA

MARCOS PAULO BERTELI SLOMP

Rendering em Tempo Real com Iluminação Baseada em Imagens em Alta Faixa Dinâmica e Harmônicas Esféricas

Projeto de Diplomação.

Prof. Dr. Manuel Menezes de Oliveira Neto
Orientador

Porto Alegre, Dezembro de 2005.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitor: Prof. José Carlos Ferraz Hennemann
Vice-reitor: Prof. Pedro Cezar Dutra da Fonseca
Pró-Reitora Adjunta de Pós-Graduação: Profa. Valquiria Link Bassani
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

AGRADECIMENTOS

Primeiramente, agradeço aos meus pais, Ivo e Nilvonei, por sempre colocarem minha educação e formação em primeiro plano. Exemplos de dedicação, inteligência e cultura, com uma simplicidade e atenção sem iguais, agradeço-os por nunca interferirem em minhas decisões e por sempre acreditarem e confiarem nos meus objetivos e na minha competência.

Agradeço aos meus irmãos, Marcelo e Mônica, pela amizade e confiança sem limites estabelecidas comigo. Agradeço pelo incentivo que depositaram em mim nas horas de desespero e pelas críticas sobre mim lançadas, para que minha formação sempre continuasse progredindo sempre para melhor.

Gostaria de agradecer aos colegas e amigos Rodrigo Gheller Luque, Diego Patrício por toda a ajuda que me ofereceram na depuração de programas, pelas sugestões oferecidas para o aprimoramento do meu trabalho e também pelas discussões construtivas desencadeadas.

Agradeço à toda a equipe da Singular Studios pelo constante processo de aprendizado e formação profissional que obtive ao longo desses últimos anos. Certamente, sem a ajuda e competência dessa equipe, inúmeras outras dificuldades impossibilitariam o bom andamento de meu trabalho.

Aos meus amigos de longa data, Léo Idalgo Barbosa e Rafael Slomp Masiero pela companhia e amizade inigualáveis estabelecidas comigo. Perseverança e determinação, essas são as palavras que resumem os aspectos positivos que mais me foram passados por vocês.

Quero agradecer aos inúmeros colegas e amigos que conheci durante toda a minha graduação dentro da universidade, em especial às inúmeras amizades que firmei com os integrantes dos grupos PET da universidade. Todos contribuíram para a minha formação e são improtantes para o meu sucesso.

Agradeço ao prof. João Luiz Dihl Comba por ter despertado fortemente em mim o interesse pela computação gráfica, durante uma disciplina ministrada em 2003. Sem esse interesse induzido, esse trabalho jamais teria se concretizado.

Agradeço (e muito!) ao prof. Manuel Menezes de Oliveira Neto, meu orientador, por toda a atenção, conhecimento e esforço oferecidos a mim. Agradeço por sempre ter acreditado em minha competência e por procurar manter-me sempre no limite de minha capacidade, aumentando-a cada vez mais.

Por fim, agradeço à todos os meus amigos e amigas, próximos ou distantes, pelo carinho e atenção sempre oferecidos, pelas festas, pelas risadas, pela compreensão e não menos importante, pela credibilidade e confiança em mim depositados. Sem pessoas como vocês, eu jamais teria a determinação e incentivo necessários para vencer sempre.

SUMÁRIO

RESUMO

O uso de algoritmos de iluminação global permite a síntese de imagens fotorrealísticas, mas o elevado custo computacional geralmente associado com estes algoritmos não permite sua utilização em aplicações em tempo real. Uma alternativa comumente empregada é o uso de algoritmos que implementam apenas parcialmente o transporte de luz definido pela Equação de *Rendering*. Nesses casos, uma solução para as trocas de energia envolvendo superfícies difusas pode ser pré-calculada para cenas estáticas e então utilizadas para obtenção de *rendering* em tempo real. Este é o caso, por exemplo, de algoritmos de radiosidade e, mais recentemente, de uma técnica conhecida como transferência de radiância pré-computada. Ao mesmo tempo, técnicas recentes de *rendering* que exploram o uso de luz natural para a simulação da interação da luz com objetos sintéticos vêm permitindo a obtenção de um elevado nível de realismo. O presente trabalho explora essas técnicas de iluminação com luz natural e transferência de radiância pré-computada através de harmônicas esféricas e o uso do hardware gráfico programável para reproduzir efeitos de iluminação global na síntese de imagens em tempo real.

ABSTRACT

Global illumination algorithms are important to photorealistic image synthesis, but the higher cost usually involved with them don't allow their use in real time applications. A way to avoid this cost is to employ algorithms that only implement partially the light transport defined in the Rendering Equation. In these cases, a solution for the transport of energy between diffuse surfaces in static scenes can be pre-computed and then used for real time rendering. This is the case of radiosity methods and, more recently, pre-computed radiance transfer techniques. At the same time, recent rendering techniques explores the use of natural light to simulate lighting with synthetic objects. This work explores pre-computed radiance transfer methods with spherical harmonics, image based lighting with natural light and the programmable graphics hardware to perform global illumination effects in real time image synthesis.

LISTA DE ABREVIATURAS E SIGLAS

BRDF – ***B****idirectional* ***R****eflectance* ***D****istribution* ***F****unction* (Função Bidirecional de Distribuição de Refletância).
HDR – ***H****igh* ***D****ynamic* ***R****ange* (Alta faixa dinâmica).
PRT – ***P****re-computed* ***R****adiance* ***T****ransfer* (Transferência de Radiância Pré-computada).
SH – ***S****pherical* ***H****armonics* (Harmônicas Esféricas).

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

Capítulo 1

Introdução

Um fator importante para obtenção de realismo em imagens geradas por computador é o modelo de iluminação utilizado ou, mais precisamente, a maneira como é simulada a interação da luz com as propriedades ópticas dos objetos que compõem uma cena. Quanto mais fiel for essa simulação, mais realistas serão as imagens produzidas. O uso de iluminação com luz natural foi introduzido recentemente em computação gráfica [Debevec98], permitindo realizar essa simulação de forma ainda mais convincente e, conseqüentemente, contribuindo para uma melhoria do realismo das imagens produzidas.

Um grande desafio dentro da computação gráfica é a síntese de imagens realistas em tempo real, uma vez que implementações de modelos de iluminação sofisticados tendem a afetar negativamente o desempenho da aplicação. Esse desafio é constante dentro de empresas de desenvolvimento de jogos e motores gráficos, onde a qualidade e eficiência do produto deve refletir a expectativa do público, de acordo com as capacidades tecnológicas disponíveis e, ainda, inovar e apresentar vantagens sobre a concorrência.

Recentemente, importantes avanços no hardware gráfico vêm possibilitando a realização de implementações eficientes de técnicas de *rendering* mais sofisticadas. A flexibilidade oferecida pelo hardware gráfico programável aos desenvolvedores permite que cenas complexas sejam sintetizadas, de maneira realista e detalhada, em tempo real. O custo relativamente baixo desse hardware é um fator importante para a aceitação dessa tecnologia por parte dos usuários de jogos de computador, e o aumento da qualidade das imagens que podem ser geradas em tempo real está sendo amplamente explorado pelos desenvolvedores de jogos.

Todos esses avanços deixam um caminho aberto para a exploração de implementações eficientes, em tempo real, de sofisticados modelos de iluminação. Com as recentes técnicas de aquisição e representação de luz natural e o suporte do hardware gráfico programável para essas novas representações, o presente trabalho explora técnicas de iluminação baseada em imagens para produzir imagens realistas em tempo real.

1.1 Objetivo

O objetivo deste trabalho é a geração de imagens realistas em tempo real, combinando o uso de luz natural e técnicas de iluminação baseada em imagens, com transferência de radiância pré-computada [Sloan-Kautz-Snyder02]. O hardware gráfico programável será utilizado para implementar, de forma eficiente, as técnicas exploradas, visando a obtenção de interatividade.

1.2 Estrutura do Texto

Este trabalho encontra-se organizado do seguinte modo:

O **Capítulo 2: Iluminação Global** apresenta uma descrição do processo de simulação de iluminação na computação gráfica. Será discutida a importância dos modelos de iluminação para realizar essa simulação, a diferença entre modelos de iluminação locais e globais e modelos de *shading*, a interação da luz com a matéria e os aspectos ópticos das superfícies (refletividade). No final do capítulo, serão apresentados dois conceitos importantes para discutir técnicas de iluminação global: a função bidirecional de distribuição de refletância (ou BRDF, que serve como base para a construção de modelos de refletividade) e a Equação de *Rendering*.

O **Capítulo 3: Iluminação Baseada em Imagens** detalha técnicas que utilizam imagens para determinar a simulação de iluminação em objetos sintéticos. A iluminação com luz natural e o conceito de alta faixa dinâmica serão introduzidos nesse capítulo, bem como um breve estudo sobre os formatos para representar imagens com essas características. O capítulo ainda descreve os operadores de *tone mapping*, importantes para reproduzir imagens em dispositivos com faixas dinâmicas distintas.

O **Capítulo 4: Transferência de Radiância Pré-computada** descreve uma técnica que procura resolver a Equação de *Rendering* em tempo real, através de procedimentos pré-computados e aproximações de funções. Será apresentado todo o processo de derivação da Equação de *Rendering* para superfícies difusas e algumas abordagens para especificação de algoritmos de transporte de luz. O capítulo ainda detalha uma família de funções, as harmônicas esféricas, útil para aproximar funções de iluminação, usualmente definidas em coordenadas esféricas.

O **Capítulo 5: Implementação e Resultados** apresenta o ambiente utilizado e os detalhes de implementação do trabalho. Será discutida a implementação, no hardware gráfico programável, dos algoritmos descritos nos capítulos anteriores, bem como as etapas de pré-computação envolvidas no processo de PRT difusa, realizadas na CPU. O capítulo também discute decisões de projeto tomadas ao longo do desenvolvimento do trabalho, assim como as dificuldades e limitações encontradas. O final do capítulo apresenta uma série de imagens, sintetizadas através do protótipo implementando, ilustrando os resultados obtidos.

O **Capítulo 6: Conclusão e Trabalhos Futuros** resume as impressões e conclusões obtidas durante e após a realização do trabalho. Uma breve discussão de melhorias que poderiam ser incorporadas ao protótipo e sugestões de trabalhos futuros a serem explorados após a análise da implementação e dos resultados também serão apresentadas no capítulo.

Capítulo 2

Iluminação Global

Este capítulo discute a importância de modelos de iluminação global para a síntese de imagens realistas. Esses modelos representam simplificações do complexo sistema de transporte de luz e sua interação com a matéria que ocorre no mundo real. O capítulo também discute a diferença entre **modelos de iluminação** e **modelos de *shading***. Será apresentada uma rápida revisão sobre o transporte de luz e os fenômenos que ocorrem quando esta atinge superfícies (matéria). Ao final, de modo a generalizar algoritmos de iluminação global, serão apresentadas as noções de **BRDFs** (funções de dstribuição bidirecional de refletância) e a **Equação de *Rendering***. Uma BRDF serve como modelo para especificar o aspecto de refletância de materiais quando há energia incidente sobre os estes. A Equação de *Rendering*, apresentada no final desse capítulo, representa de maneira compacta o transporte de luz em uma cena. A última seção do capítulo apresenta um sumário com os conceitos mais importantes que devem ser compreendidos ao longo do capítulo, de forma a não comprometer o entendimento dos demais capítulos do trabalho. Sugere-se, também, a leitura do Anexo I, apresentando as definições das medidas mais importantes em **Radiometria** que se encaixam no contexto do trabalho, como o conceito de **fluxo radiante** e **radiância**.

2.1 Introdução

Uma imagem sintética (gerada por computador) é dita **fotorrealística** quando esta leva seu observador a acreditar tratar-se de uma fotografia obtida de uma cena real. O interesse por fotorrealismo em computação gráfica é intenso nas áreas da publicidade, *design*, arquitetura, e, também, na indústria de entretenimento (*Figura 2.1*). Nessa última, a indústria de jogos está interessada na possibilidade de geração de imagens realistas, que possam ser produzidas em tempo real. Um dos fatores mais importantes para obtenção desse tipo de imagens é a simulação do processo de interação da luz com a cena de maneira realista. Esse processo é representado por meio de **modelos de iluminação**, e serão discutidos a seguir.

Figura 2.1: Exemplo de imagem fotorrealística - cena do filme Final Fantasy VII Advent Children (2005). Embora artisticamente os personagens estejam estilizados, a imagem apresenta diversos aspectos realísticos, como sombras suaves e profundidade de campo. Os materias se comportam de maneira diferenciada em resposta à luz recebida (roupas, cabelos, pele e armas). Extraído de: http://www.adventchildren.net/

2.2 Modelos de Iluminação

Um modelo de iluminação descreve como um determinado ponto em uma superfície é iluminado. O nível de realismo de uma imagem sintética está diretamente relacionado com a capacidade do modelo de representar e simular a interação da luz com a cena.

Os modelos de iluminação podem ser classificados em dois grupos: **modelos de iluminação local** e **modelos de iluminação global**. Modelos de iluminação local consideram apenas informações isoladas das componentes da cena: cada ponto de uma superfície é iluminado independentemente do que está ao seu redor, isto é, sem levar em conta o contexto em que este se insere na cena. A avaliação do modelo considera apenas a posição do observador, propriedades do ponto a ser iluminado (vetor normal e aspectos ópticos, por exemplo) e propriedades das fontes de luz. Um exemplo de imagem obtida com o uso de modelo de iluminação local é mostrado na *Figura 2.2*.

Os modelos de iluminação global, por outro lado, procuram abranger todo o contexto da cena para determinar a iluminação de cada ponto em uma superfície. Isso possibilita a obtenção de efeitos importantes, como sombras (oclusão de luz) e transferência de luz entre superfícies, que, obviamente, resultam em cenas muito mais realistas, como pode ser observado na *Figura 2.3*.

Figura 2.2: Imagem do jogo James Bond: Nightfire (PC – 2002/2003). Um modelo de iluminação local foi utilizado. Observe a ausência de sombras entre os personagens e o cenário. Além disso, a luz que interage com as paredes não contribui para a iluminação de nenhum outro objeto da cena. Extraído de: http://pc.ign.com

Figura 2.3: Imagem sintética obtita através da aplicação de um modelo de iluminação global. Podemos perceber regiões com sombras suaves. A parede ao lado da escada receba luz indireta, refletida de outros pontos da cena. Extraído de: http://www.yafray.org

A qualidade dos efeitos de iluminação produzidos por um modelo de iluminação local é muito inferior àquela obtida com um modelo de iluminação global. Entretanto, o custo computacional para avaliar um modelo de iluminação local é muito mais baixo (tanto em processamento quanto em memória, visto que não há necessidade de conhecer outros elementos da cena). Esse é o grande motivo pelo qual os modelos de iluminação

local são tão utilizados para realizar o *rendering* de cenas em tempo-real em jogos de computador.

O modelo de iluminação local mais conhecido e utilizado atualmente é o modelo de iluminação proposto por Phong [Phong73], utilizado na síntese da imagem mostrada na *Figura 2.2*. O hardware gráfico apresenta suporte para implementação do modelo de Phong em tempo real. Os algoritmos de *ray-tracing* [Whitted80] (*Figura 2.4b*) e o método da radiosidade [Goral84] (*Figura 2.4a*) implementam modelos de iluminação global, mas ainda existem limitações no hardware atual que dificultam suas implementações em tempo real. A *Seção 2.6* discute a **Equação de *Rendering***, uma expressão que generaliza as técnicas de iluminação global [Kajiya86].

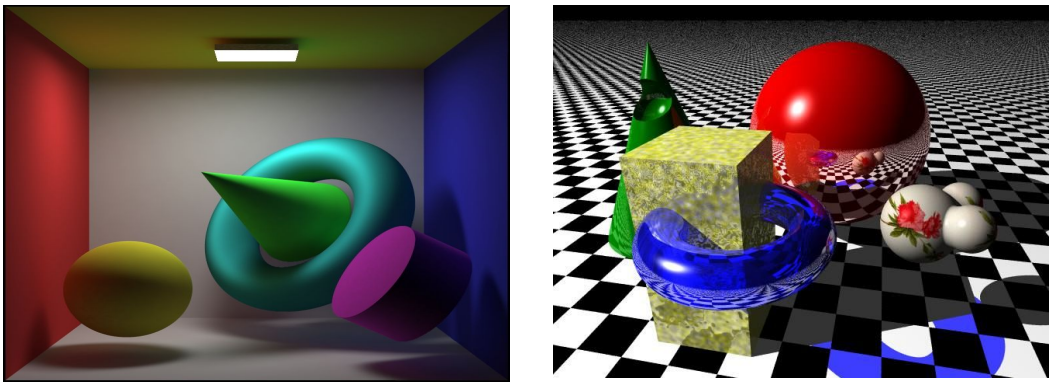


Figura 2.4: (a) Imagem gerada utilizando o método da radiosidade. (b) Imagem sintetizada através do algoritmo de ray-tracing. Extraído de: http://www.tjhsst.edu/~dhyatt/supercomp

2.3 Modelos de *Shading*

O termo *shading* define a cor percebida em cada ponto de uma superfície. Ele estabelece a freqüência com que um determinado modelo de iluminação é avaliado em uma superfície.

Os modelos de *shading* mais comuns são: Flat, Gouraud [Gouraud71] e Phong [Phong75]. Assim como os modelos de iluminação, o emprego de um determinado modelo de *shading* está relacionado com a qualidade desejada e custo computacional envolvido.

No flat *shading* (*Figura 2.5a*) o modelo de iluminação é avaliado uma única vez para cada face do modelo poligonal, e todos os *pixels* gerados por essa face receberão o mesmo valor de iluminação (toda a face recebe o mesmo valor de iluminação). O modelo de *shading* de Gouraud (*Figura 2.5b*) avalia a função de iluminação para cada vértice do modelo, preenchendo o interior das faces através de interpolação dos valores obtidos nos vértices. O modelo de *shading* de Phong (*Figura 2.5c*) aplica o modelo de iluminação para cada pixel obtido pelo processo de rasterização do modelo geométrico. Na primeira abordagem, a primitiva de iluminação é uma face; no segundo, um vértice; no terceiro, um pixel. É importante não confundir o **modelo de iluminação de Phong** com o **modelo de *shading* de Phong**.

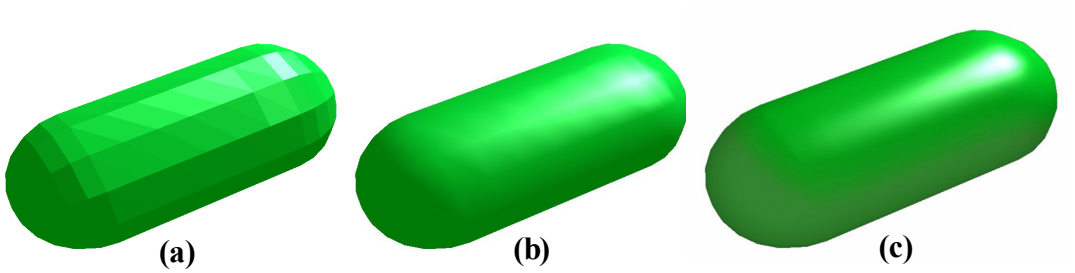


Figura 2.5: Cápsula renderizada em três modelos de shading distintos. Todos os modelos de shading utilizam o mesmo modelo de iluminação, o modelo de Phong. (a) Flat Shading; (b) Gouraud Shading; (c) Phong Shading.

2.4 Interação entre luz e cena

O cerne de um modelo de iluminação está no processo de simular o transporte de luz na cena. Todas as superfícies da cena podem receber luz **diretamente** e/ou **indiretamente** (*Figura 2.6*). Quando a luz viaja, sem obstáculos, até uma superfície, temos iluminação direta. Se a luz atinge uma superfície, e parte dessa luz é refletida sobre outro objeto, temos iluminação indireta. Para avaliar a iluminação recebida (direta e indireta) por uma superfície, os modelos de iluminação exploram métodos empíricos e métodos analíticos.

Os métodos empíricos extraem quantidades (medidas) de luz refletida a partir da luz incidente, em **todas** as direções. Isso dificulta seu uso na prática, porque há um grande custo envolvido no armazenamento e recuperação dessas medidas. A maioria dos modelos de iluminação, principalmente os locais, baseiam-se em métodos analíticos, modelando a refletância de uma superfície através de parâmetros e funções que melhor se adaptam a um determinado comportamento esperado.

Para entender como funcionam os métodos analíticos e empíricos, e a interação entre luz e matéria, são necessários alguns conceitos de **radiometria** e **fotometria** e, também, algum estudo sobre refletância de superfícies. Há outros conceitos envolvidos na interação de luz com uma cena real, como polarização, difração e transmissão, mas estão além dos objetivos deste trabalho. O Anexo I apresenta os principais conceitos de radiometria e fotometria, e alguns deles, como **fluxo**, **potência radiante** e **radiância**, serão utilizados no decorrer do trabalho.

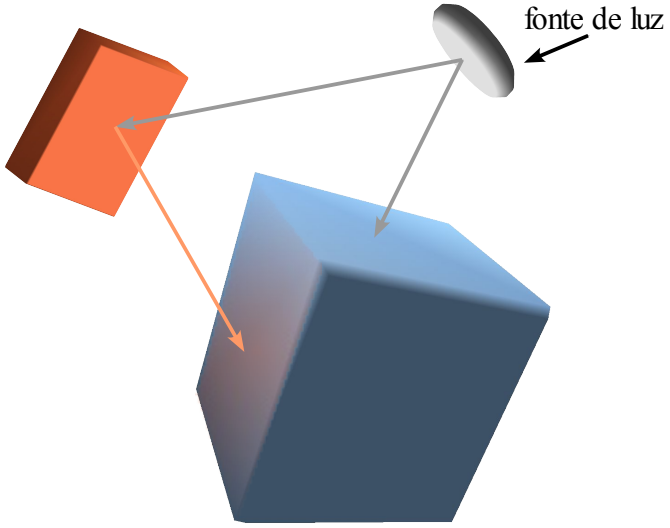


Figura 2.6: A fonte de luz ilumina, diretamente, a face superior do cubo azul e uma face lateral do sólido laranja. A luz que incide nesse último é refletida e atinge uma das faces laterais do cubo azul. As setas cinzas indicam iluminação direta, enquanto a seta laranja indica iluminação indireta.

2.5 Aspectos Ópticos das Superfícies

Quando a luz (energia) atinge uma superfície (matéria), podem ocorrêr três fenômenos: parte (ou toda) a luz será **refletida**, **transmitida** e/ou **absorvida**, e quando a luz é refletida, pelo menos um raio novo é gerado. Pela Lei da Conservação de Energia, é válida a seguinte igualdade (*Equação 2.1*):

$$I_i = I_r + I_t + I_a$$

Equação 2.1: Energia (luz) incidente sobre uma superfície.
I_i é a energia incidente sobre a superfície.
I_r é a energia refletida na superfície.
I_t é a energia transmitida através da superfície.
I_a é a energia absorvida pela superfície.

A luz refletida e a luz transmitida irão determinar o aspecto final da iluminação da superfície do objeto, embora, para materiais opacos, a luz refletida seja mais relevante. A luz absorvida também colabora para o resultado final da iluminação, uma vez que não há retorno luminoso. A luz trasmitida pode ser re-emitida, fenômeno que pode ser simulado pela técnica de s*ubsurface scattering* [Hanrahan93], mas esse tópico, assim como a trasmissividade e a absorção, não será discutido nesse trabalho.

2.5.1 Refletividade e Refletância

O conceito de **refletividade** define, basicamente, a capacidade de uma superfície

refletir a energia (luz) incidente, e está relacionado diretamente com as duas leis da reflexão:

Primeira lei da reflexão: os raios incidente, refletido e o vetor normal da superfície encontram-se, todos, no mesmo plano.

Segunda lei da reflexão: o ângulo de refletância é igual ao ângulo de incidência, para uma superfície refletora ideal.

Refletores ideais (*Figura 2.7a*) são incomuns na prática. Nas superfícies em geral, há um fenômeno de difusão da reflexão, fazendo com que a luz incidente não seja refletida, unicamente, em uma direção mas, sim, espalhando-se (difundindo-se) em diversas direções.

Superfícies ditas **difusores ideias** (ou superfícies **lambertianas**) são aquelas onde a luz é refletida, com **mesma intensidade**, em todas as direções (*Figura 2.7b*). Um observador, independente do ponto que enxerga a superfície, perceberá sempre a mesma quantidade de luz refletida. Assim como os refletores ideais, na prática, não possuímos difusores ideais.

Uma superfície, portanto, pode variar, em termos de refletividade, entre um difusor ideal e um refletor ideal. De modo a modelar essa propriedade óptica da superfície, a luz refletida é analisada através de duas componentes distintas: **componente difusa**, relativa à reflexão que ocorre em todas as direções, e **componente especular**, correspondendo à reflexão ocorrida nas proximidades da direção de reflexão ideal.

A capacidade de refletir luz é medida através da **refletância**, uma relação entre o fluxo refletido e o fluxo incidente em uma superfície. A refletância em superfícies difusas também é chamada de ***albedo*** (ρ).

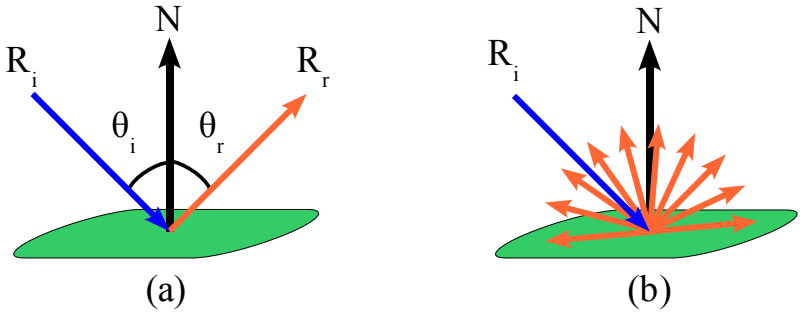


Figura 2.7: (a) Refletor ideal: o raio incidente (R_i) é refletido, gerando um novo raio (R_r), ambos formam um mesmo ângulo ($\theta_i = \theta_r$), com respeito ao vetor normal (N) da superfície. (b) Em um difusor ideal, o raio incidente (R_i) é refletido igualmente em todas as direções.

2.5.2 Função Bidirecional de Distribuição de Refletância (BRDF)

Uma função bidirecional de distribuição de refletância (do inglês, *Bidirectional Reflectance Distribution Function* - **BRDF**) define a refletância de um ponto em uma superfície em função dos raios incidente e refletido (*Equação 2.2*). Ela é determinada através das propriedades ópticas da superfície (sombreamento mútuo, transmissividade, refletividade, emissividade, orientação, comprimentos de onda da luz incidente, etc) e, no mundo real, objetos são iluminados de maneiras distintas quando observados de diferentes pontos de vista ou quando recebem luz proveniente de diferentes direções (*Figura 2.8*). Em outras palavras, uma BRDF expressa um modelo de interação da luz com a matéria.

$$f_r(x, \vec{\omega}_i, \vec{\omega}_r) = \frac{L_r}{E_i} = \frac{L_r(x, \vec{\omega}_r)}{L_i(x, \vec{\omega}_i)\cos(\theta_i)\, d\,\vec{\omega}_i}$$

Equação 2.2: função bidirecional de distribuição de refletância (BRDF).
***x** é um ponto sobre a superfície.*
ω_i é a direção da luz incidente.
ω_r é a direção da luz refletida.
***L(x,ω)** representa a medida de radiância no ponto x, na direção ω (incidência ou saída).*
θ_i é o ângulo de incidência sobre a superfície.

A função BRDF expressa a razão entre a quantidade de luz refletida na direção de reflexão especificada e a quantidade de luz que incide sobre a superfície.

Figura 2.8: fotografia de uma plantação de soja, em condições de iluminação e ponto de vista distintos: (a) o Sol está atrás do observador; (b) o Sol está na frente observador. Note que a luz refletida percebida depende da posição da fonte de luz. Fotografia de: Don Deering (Goddard Space Flight Center)

Existem duas classes de BRDFs: **isotrópicas** e **anisotrópicas**. Uma BRDF é dita

isotrópica quando suas propriedades de refletância são invariantes com respeito à rotação da superfície em torno do vetor normal da mesma (*Figura 2.9b*). Se suas propriedades se alterarem com respeito a essa rotação, a BRDF é dita anisotrópica (*Figura 2.9c*). A grande parte dos materiais existentes no mundo real tem comportamento anisotrópico.

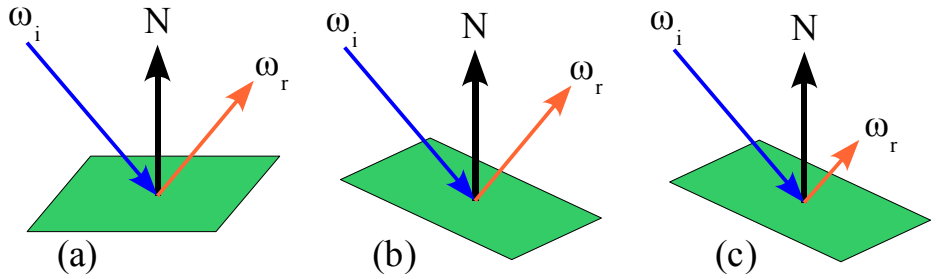


Figura 2.9: (a) superfície com uma rotação específica. Uma BRDF isotrópica (b) não altera a refletância (representada através do comprimento do vetor ω_r) quando a superfície for rotancionada em torno de seu vetor normal (N). Já uma BRDF anisotrópica (c) não mantém o valor de refletância quando a superfície é rotacionada.

Há, ainda, duas propriedades importantes para que uma BRDF seja fisicamente plausível: ela deve obedecer a Lei da Reciprocidade (*Equação 2.3* e *Figura 2.10*) e a Lei de Conservação de Energia (*Equação 2.4*).

$$f_r(x, \vec{\omega}_i, \vec{\omega}_r) = f_r(x, \vec{\omega}_r, \vec{\omega}_i)$$

Equação 2.3: Lei da Reciprocidade em BRDFs. Mantendo-se as direções inalteradas , mas invertendo o sentido dos fluxos, a igualdade deve ser satisfeita, para que a Lei da Reciprocidade seja obedecida.

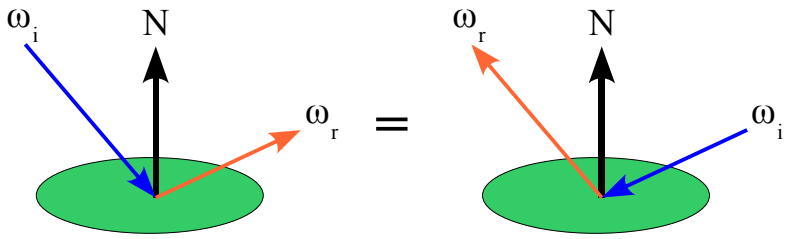


Figura 2.10: Lei da Reciprocidade. A avaliação da função (BRDF) para uma dada direção de incidência e direção de reflexão (ω_i e ω_r), independe da ordem na qual estas são aplicadas na função BRDF.

$$\int_{\Omega_r} f_r(x, \vec{\omega}_i, \vec{\omega}_r)\, d\omega_r \leq E_i$$

Equação 2.4: Lei da Conservação de Energia em BRDFs. Para superfície não emissoras, a quantidade de energia que deixa a superfície, em todas as direções possíveis ($\boldsymbol{\Omega_r}$), não pode ser superior à quantidade energia que incide sobre a mesma ($\boldsymbol{E_i}$).

2.6 A Equação de *Rendering*

A Equação de *Rendering* [Kajiya86] descreve o transporte de luz em um ambiente ou, em outras palavras, o comportamento do fluxo de energia luminosa presente em uma cena. É um modelo que prevê resultados teoricos ideais e é capaz de generalizar (abstrair) qualquer algoritmo de iluminação global (que nada mais são do que métodos numéricos para resolver a equação de *rendering*).

A Lei de Conservação de Energia é o cerne da equação. Intuitivamente, a contribuição luminosa de um ponto, em uma cena, numa determinada direção, é resultado da luz emitida diretamente por este, naquela direção, somado a tudo que esse ponto reflete na direção especificada, baseado na contribuição luminosa que recebe de todas as direções que o cercam (*Equação 2.5*). Quando se fala em resolver a equação de *rendering*, deseja-se descobrir a radiância que chega ao observador da cena (determinar os valores dos pixels da imagem a ser sintetizada).

$$L(x,\vec{\omega}) = L_e(x,\vec{\omega}) + L_r(x,\vec{\omega})$$

Equação 2.5: Representação compacta da Equação de Rendering.
$L_e(x,\omega)$ *representa a luz emitida pelo ponto x na direção ω.*
$L_r(x,\omega)$ *representa toda a luz incidente no ponto x que é refletida na direção ω.*

Para pontos que não pertencem à fontes de luz, a emissividade destes será usualmente zero. A luz refletida pelo ponto, na direção especificada, depende, como visto anteriormente, da contribuição luminosa que este recebe das direções que o cercam. Combinando essa intuição com a definição de BRDF (*Equação 2.2*), podemos definir essa componente de luz refletida, através da (*Equação 2.6*):

$$L_r(x,\vec{\omega}_r) = \int_{\Omega_i} f_r(x,\vec{\omega}_i,\vec{\omega}_r)\, L_i(x,\vec{\omega}_i) \cos(\theta_i)\, d\vec{\omega}_i$$

Equação 2.6: A contribuição de luz refletida no ponto ***x****, na direção* $\boldsymbol{\omega_r}$ *considera a contribuição de luz incidente proveniente no conjunto de todas as direções que o cercam (*$\boldsymbol{\Omega_i}$*).*

Voltando à *Equação 2.5* e substituindo nela o termo previamente definido na *Equação 2.6*, chega-se à forma geral da Equação de *Rendering* (*Equação 2.7*):

$$L(x,\vec{\omega}_r) = L_e(x,\vec{\omega}_r) + \int_{\Omega_i} f_r(x,\vec{\omega}_i,\vec{\omega}_r)\, L_i(x,\vec{\omega}_i) \cos(\theta_i)\, d\vec{\omega}_i$$

Equação 2.7: A Equação de Rendering.

A forma geral da Equação de *Rendering* baseia-se apenas em medidas de radiância e nas propriedades espaciais (orientação) e ópticas (BRDF) da superfície. Ela ainda pode ser reescrita, considerando o aspecto geométrico e de visibilidade dos pontos do espaço que geram luz incidente sobre o ponto de interesse (*Equação 2.8*):

$$L(x,\vec{\omega}_r) = L_e(x,\vec{\omega}_r) + \int_{\Omega_i} f_r(x,\vec{\omega}_i,\vec{\omega}_r)\, L_i(x,\vec{\omega}_i)\, V(x,x')\, G(x,x')\, d\vec{\omega}_i$$

Equação 2.8: Outra possível representação da Equação de Rendering.
$\boldsymbol{\omega_i}$ *representa a direção incidente de um ponto x' sobre o ponto x.*
V(x,x') *é uma função de visibilidade: seu valor será 0 se houver algum obstáculo entre x e x';*

caso contrário, seu valor será 1.
***G(x,x')** define uma relação geométrica entre ambos os pontos.*

Nos algoritmos de iluminação global, a função de visibilidade costuma ser a operação mais cara, visto que ela deve ser computada inúmeras vezes para cada ponto e implica em percorrer os modelos geométricos da cena, calculando possíveis intersecções com a malha poligonal desses. Esse processo pode ser acelerado utilizando estruturas de dados inteligentes [Luque05] para realizar o particionamento do espaço, evitando que raios sejam testados desnecessariamente contra polígonos.

2.7 Sumário

Este capítulo discutiu a importância dos **modelos de iluminação global** na síntese de imagens fotorrealísticas. Também foram citados alguns algoritmos de iluminação global (ray-tracing e o método da radiosidade) e eles nada mais são do que soluções numéricas de modelos de iluminação global.

A seguir foi discutida a maneira pela qual a luz interage com elementos de uma cena. Objetos podem receber **iluminação direta** de uma fonte de luz e, ainda, receber **iluminação indireta**, proveniente da luz refletida por outros objetos. A luz incidente em uma superfície pode ser **refletida**, **transmitida** ou **absorvida**, e deve respeitar a Lei da Conservação de Energia. As superfícies, de modo geral, apresentam características **especulares** e **difusas** e para descrever a refletância de superfícies foi discutido o conceito de **função bidirecional de distribuição de refletância** (**BRDFs**), que expressa a refletância de materias sob diversas condições. Uma BRDF pode ser **isotrópica** ou **anisotrópica**, e deve obedecer a Lei da Reciprocidade e a Lei da Conservação de Energia para que seja fisicamente plausível.

O final do capítulo apresentou a **Equação de *Rendering***, uma fórmula que descreve o transporte de luz, generalizando algoritmos de iluminação global. A Equação de *Rendering* é baseada em medidas de **radiância** (conceito discutido no Anexo I) e utiliza, em seu cerne, a definição de BRDF.

Capítulo 3

Iluminação Baseada em Imagens

Nesse capítulo, serão discutidas algumas técnicas de iluminação, focadas na utilização de fotografias para realizar a simulação do transporte de luz em objetos sintéticos. Serão apresentadas duas representações de imagens panorâmicas (***cube-maps*** e ***light probes***), capazes de representar luz incidente em todas as direções do espaço, e que podem ser utilizadas em técnicas de iluminação baseada em imagens. A definição de **faixa dinâmica**, muito importante para a síntese de imagens fotorrealísticas utilizando **luz natural**, e o mapeamento entre diferentes faixas dinâmicas, obtido através do processo de ***tone mapping*** serão discutidos no capitulo. *Tone mapping* é utilizado para que imagens capturadas ou geradas de maneira realista possam ser exibidas em dispositivos cujas faixas dinâmicas suportadas sejam menores que a faixa dinâmica da imagem. O capítulo oferece, ainda, um breve estudo sobre diferentes formatos de imagem para representar e armazenar imagens com **alta faixa dinâmica**. Ao final, a última seção apresenta um sumário das informações mais importantes que devem ser compreendidas nesse capítulo, de forma a facilitar a compreensão dos demais capítulos do trabalho.

3.1 Introdução

Um dos recursos comumente utilizandos na geração de imagens realistas é a utilização de informações presentes na realidade. Fotografias tiradas de diferentes materiais podem ser utilizadas como texturas e aplicadas diretamente sobre as superfícies de modelos sintéticos. É possível, ainda, capturar a iluminação de uma cena real por meio de fotografias e utilizá-las para iluminar cenas sintéticas. Esse processo é conhecido como **iluminação baseada em imagens**. A *Figura 3.1* mostra uma imagem onde a cena foi iluminada utilizando **luz natural**.

A técnica procura determinar de que maneira os objetos que compõem uma cena são iluminados com base em uma ou mais imagens de referência. A iluminação baseada em imagens é uma ferramenta muito eficaz para a integração de objetos sintéticos em cenas reais como acontece, por exemplo, nos filmes de ficção científica. As origens dessa técnica estão associadas com a técnica de ***reflection mapping***.

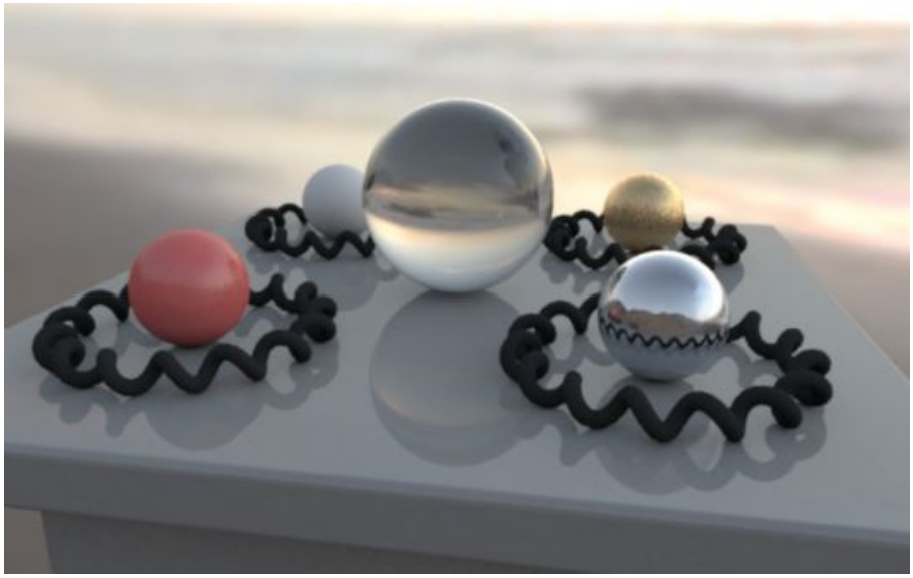


Figura 3.1: A cena acima foi produzida utilizando iluminação baseada em imagens. Todos os objetos da cena são sintéticos, mas a imagem utilizada para realizar a iluminação foi retirada (fotografada) de uma cena real. Cortesia de Paul Debevec.

3.2 *Reflection Mapping*

A técnica de *reflection mapping* [Blinn76] [Debevec/ReflectionMap], assim como ***environment mapping***, procura demonstrar a maneira pela qual superfícies são iluminadas quando imersas em um ambiente. Esses ambientes são representados através de imagens panorâmicas, comumente representadas por meio de ***cube maps*** (*Figura 3.2*).

Na técnica de *environment mapping* tradicional, o vetor normal à superfície é utilizado como entrada para uma rotina de acesso a textura, que irá retornar a informação de iluminação contida na direção especificada pelo vetor normal. A posição do observador não é relevante (*view-independent*). Se a superfície for rotacionada, o seu vetor normal também será, alterando sua iluminação. A *Figura 3.3a* ilustra este processo.

A técnica de Reflection mapping funciona de modo semelhante, mas considerando o ponto de vista do observador; um vetor de observação é obtido a partir da posição do observador e da posição sobre a superfície, sendo utilizado para o cálculo do vetor relfletido com respeito ao vetor normal da superfície. Esse vetor relfletido servirá como entrada para a rotina de acesso a textura, conforme ilustrado na *Figura 3.3b*.

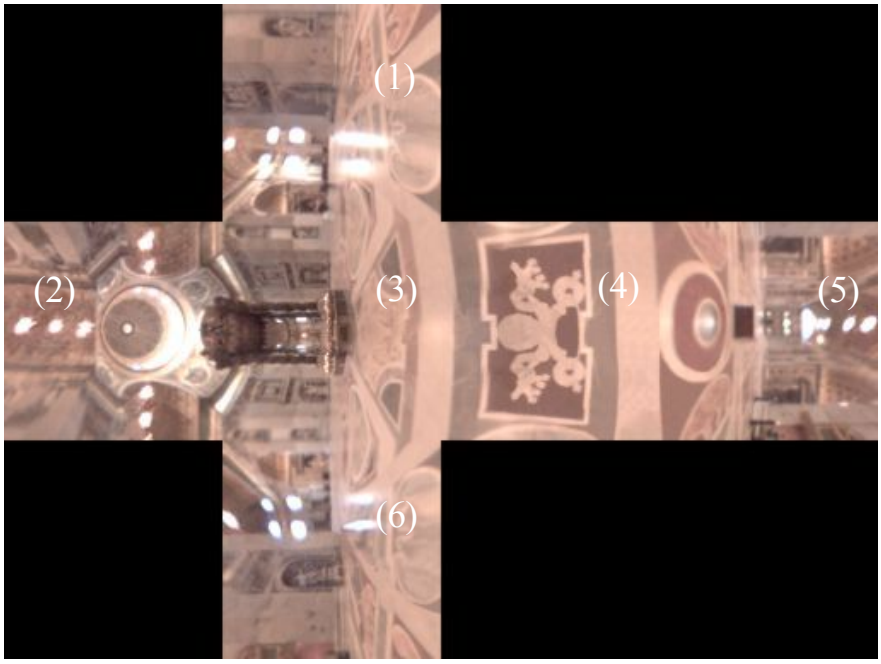


Figura 3.2: Um cube-map é uma imagem panorâmica. (1), (3), (5) e (6) são as faces lateriais do cubo, enquanto (2) e (4) representam, respectivamente, a face superior e inferior. Extraído de: http://www.debevec.org/Probes

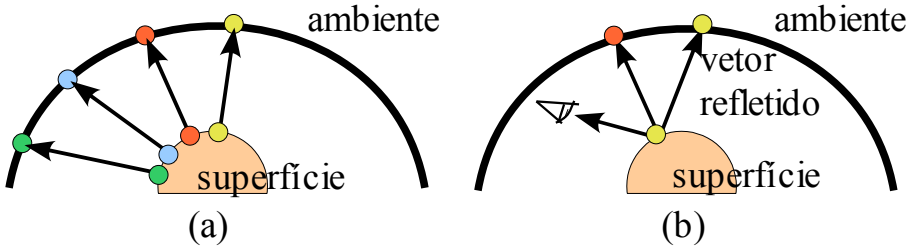


Figura 3.3: (a) Environment mapping considera apenas o vetor normal à superfície. Já o reflection mapping (b) considera, ainda, a posição do observador para então calcular a direção de reflexão em torno do vetor normal ao ponto.

As superfícies iluminadas por meio de *evironment mapping* e *reflection mapping* apresentam aspecto metálico ou de espelho (refletores ideais), porque um único vetor de direção é utilizado para determinar a iluminação final de cada ponto da superfície.

Apesar da característica fortemente especular, é possível extrair componentes difusas dessas imagens panorâmicas, atráves de processamento especial sobre as mesmas. O Capítulo 4 apresenta algumas abordagens de como utilizar essas imagens para avaliar contribuições difusas às superfícies.

As técninas de *environment/reflection mapping* são bastante eficientes e de simples implementação em hardware, e produzem resultados muito interessantes, que podem ser produzidos em tempo real e sua aplicação é cada vez mais freqüente em jogos de computador (*Figura 3.4*).

Figura 3.4: Imagem do jogo Need for Speed Underground (PC – 2003/2004). A equipe de desenvolvimento utilizou reflection mapping para intensificar a aparência metálica dos veículos do jogo, como podemos perceber na parte traseira do carro. Extraído de: http://pc.ign.com

3.3 *Light Probes*

Light probes (*Figura 3.5*), assim como os *cube-maps*, são imagens omnidirecionais, que armazenam informação de luz incidente de diversas direções do espaço. Sua aplicação popularizou-se com os trabalhos de Paul Debevec [Debevec98]. *Light probes* usualmente armazenam valores em **alta faixa dinâmica**, um conceito que será discutido na próxima seção.

Embora a apresentação da imagem, mostrada na *Figura 3.5*, possa não enfatizar, ela é capaz de representar informação de todas as direções do espaço (4π estrerradianos). As bordas do círculo nesta imagem contém uma quantidade muito grande de informação.

O processo para a obtenção de *light probes* de ambientes reais envolve o processamento de uma série de fotografias (denominadas ***radiance maps*** [Debevec97]), capturadas com diversos tempos de exposição e em determinados ângulos da cena. Um tutorial explicando como construir *light probes* pode ser encontrado em [Debevec/HDRShopTutorial]. apresenta um tutorial de como obter fotografias e combiná-las, através de um *software* [Debevec/HDRShop], para a construção de *light probes*.

Figura 3.5: Light probe: representa, assim como um cube map, informação panorâmica da cena e também pode ser utilizada na iluminação baseada em imagens. Extraído de: http://www.debevec.org/Probes

3.4 Alta Faixa Dinâmica (HDR)

A luz, no mundo real, varia em grandes escalas, em termos de luminância. Para verificar o fenômeno, basta comparar a intensidade de luz percebida ao observar uma cena escura com aquela precebida olhando-se diretamente para uma lâmpada acesa. Esse contraste (faixa dinâmica) pode ser medido analisando a distribuição das regiões mais claras e mais escuras da cena.

Sejam *m* e *M*, respectivamente, a menor e a maior luminância de uma cena, a faixa dinâmica dessa cena é denotada por *M*:*m*. Exemplificando, se uma cena possui como menor luminância o valor 1 e possui 10.000 como maior luminância, então a faixa dinâmica dessa cena é 10.000:1.

O termo alta faixa dinâmica (do inglês, *high dynamic range* - **HDR**) se aplica a imagens que possuem uma faixa dinâmica muito maior do que aquela disponível em um monitor de vídeo ou impressora para sua exibição. O conceito também se aplica a fotografias reais, onde a faixa dinâmica da cena é muito maior do que a faixa dinâmica suportada pelo filme.

Em dispositivos digitais, na prática, imagens em alta faixa dinâmica são armazenadas utilizando representações de ponto flutuante, variando, em faixas numéricas contínuas, de zero até valores muito maiores do que um milhão. Por outro lado, formatos de imagens tradicionais costumam armazenar cada canal de cor (RGB) de seus pixels no intervalo discreto de 0 a 255 (um byte). Nas próximas seções, serão descutidos alguns formatos para representar e armazenar imagens com alta faixa

dinâmica. A *Figura 3.6* compara a representação de imagens em um formato tradicional com aquelas que suportam alta faixa dinâmica.

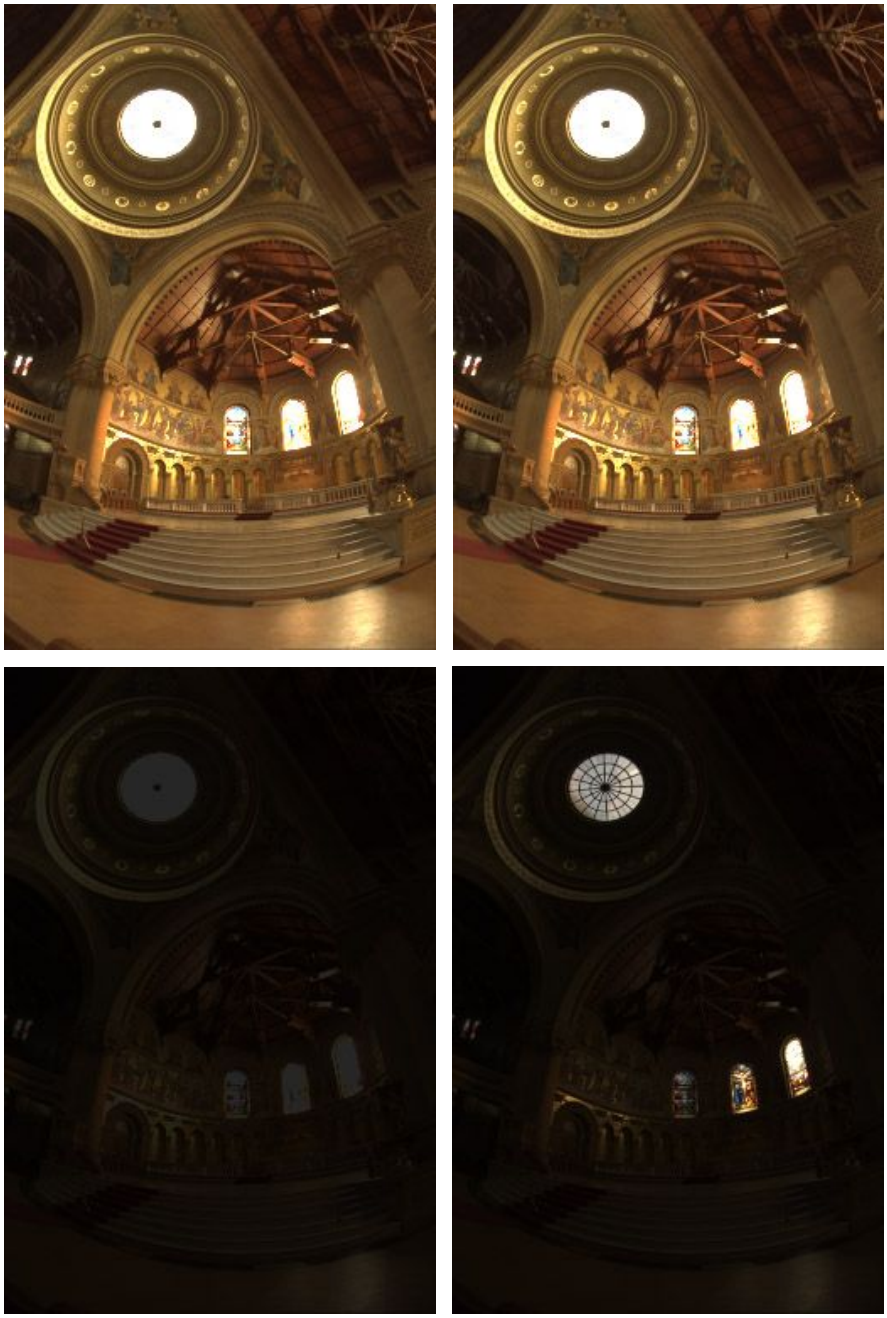


Figura 3.6: Acima, imagens em baixa faixa dinâmica (esquerda) e em alta faixa dinâmica (direita). Abaixo, as mesmas imagens, com seus valores de luminância reduzidos 64 vezes. Imagens extraídas de: [Debevec/HDRShop]

Para ser possível reproduzir toda essa informação em um dispositivo com menor faixa dinâmica, é necessário realizar um mapeamento, utilizando um operador, conhecido como **operador de *tone mapping***.

3.5 *Tone Mapping*

A função do *tone mapping* [Tumblin-Rushmeier91] é mapear uma determinada faixa dinâmica para outra faixa mais estreita. Estima-se que o sistema visual humano seja capaz de perceber variações de luminância de até 5 ordens de magnitude, adaptando-se gradualmente a até 9 ordens de magnitude [Ward97]. No mundo real, a luz natural varia em até 10 ordens de magnitude [Ferwerda96], enquanto dispositivos para exibição gráfica suportam apenas 2 ordens de magnitude [Seetzen04].

Tone mapping surgiu, originalmente, como uma técnica fotográfica, na qual um determinado filme revelado adaptava a faixa dinâmica capturada do mundo real para a faixa dinâmica na qual o papel de revelação era capaz de suportar. Além disso, o fotógrafo ainda precisava configurar sua câmera para uma determinada situação desejada (ajustando o tempo exposição do filme, por exemplo). Descrições mais detalhadas de técnicas fotográficas relacionadas a *tone mapping* podem ser encontradas em [London-Upton1998].

Para ilustrar a necessidade e a importância do processo de *tone mapping*, vamos analisar o seguinte exemplo: você está tirando férias em uma praia e a janela de seu quarto fica diretamente na direção em que o Sol nasce. Amanhece e você acorda, no seu quarto, com a janela fechada. Você levanta, abre-a e, imediatamente, sua visão fica ofuscada, devido ao excesso de luz. Aos poucos, o seu sistema visual adapta-se à nova faixa de iluminação e você consegue ver o mar, a areia, os quiosques e as pessoas caminhando na praia. Como o tempo está bonito, você resolve ir até o porão da casa buscar sua bicicleta para dar uma volta. O porão, por sua vez, está muito escuro e você não consegue enxergar nada, mas, aos poucos, a pouca iluminação proveniente da porta de acesso ao porão é suficiente para você discernir os objetos que lá se encontram (mais uma vez, o sistema visual adaptou-se à uma condição de iluminação). Se você resolvesse tirar uma fotografia da paisagem ensolarada ou do porão escuro, sem o devido ajuste no tempo de exposição, a primeira ficaria saturada (*over-exposed*) enquanto que a segunda ficaria completamente escura (*under-exposed*).

Existem diversos operadores de tone map. O presente trabalho irá discutir apenas um deles [Reinhard02], porque esse operador leva em consideração a cena inteira, não apenas informações locais. Além disso, outros operadores necessitam de ajustes diretos (como a determinação do fator de exposição), enquanto esse depende apenas da informação de luminância contida na cena.

3.5.1 Operador de *tone mapping* de Reinhard

O operador de *tone mapping* proposto por [Reinhard02] é dedicado ao processamento de imagens digitais, e procura corrigir alguns problemas encontrados nos demais operadores (seção 3.5.2). Para atingir seus objetivos, a técnica procura considerar muito mais do que apenas a variação de luminância; a luminância média e a condição de observação também são consideradas [Fairchild98].

3.5.1.1 Sistema de Zonas (*Zone System)*

Essa técnica fotográfica consiste em utilizar informação medida na cena fotografada para aumentar as chances de obter uma revelação de boa qualidade, e é denominada *Zone System* [White84]. Uma zona representa uma faixa de luminância da cena (assim como de um papel fotográfico). As zonas são identificadas através do número zero e de algarismos romanos (I a X), onde 0 representa a menor luminância (preto puro) e X representa a maior luminância (branco puro), totalizando onze zonas, e cada zona tem seu valor dobrado com respeito à zona anterior (*Figura 3.7*).

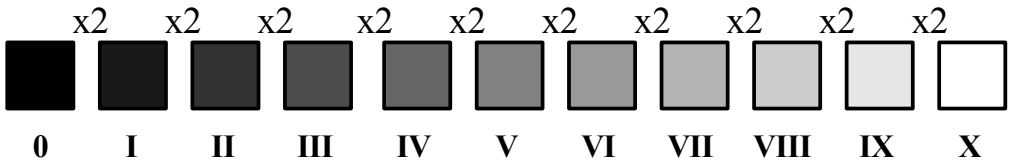


Figura 3.7: Ao todo, no sistema de zonas, existem 11 faixas. A cada faixa, o valor de luminância é dobrado, com respeito à faixa anterior.

Uma dessas faixas (zonas) é escolhida para ser a luminância média da cena. Usualmente, escolhe-se a zona V, mas esta escolha não é obrigatória. A zona V corresponde à 18% da refletância do papel fotográfico. Para cenas muito luminosas (o exemplo da foto do nascer do Sol), zonas mais escuras podem ser escolhidas como luminância média, enquanto que em cenas muito escuras (a foto do porão), uma zona mais brilhante é desejada. Esse valor também é chamado de *middle-grey* e sua determinação fica a critério do fotógrafo. A medida da luminância subjetiva da cena (muito clara, muito escura, etc) é denominada **chave** (*key)*. A *Figura 3.8* ilustra o resultado obtido mapeando o mesmo valor chave para zonas médias diferentes.

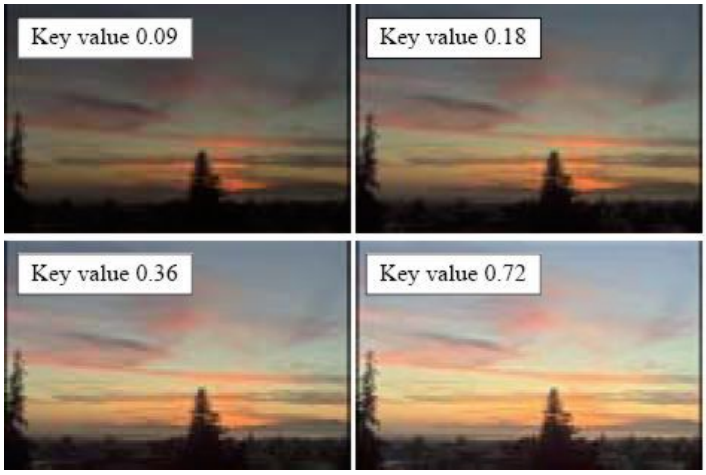


Figura 3.8: A mesma fotografia, revelada utilizando quatro valores distintos para o middle-grey, que é onde será mepeada a luminância média da imagem. Extraído de: [Reinhard02]

No próximo passo, o fotógrafo deve extrair a faixa dinâmica da cena, analisando as regiões mais escuras e as mais brilhantes. Se a faixa dinâmica da cena não exceder à nove zonas, então o fotógrafo pode garantir que conseguirá uma zona intermediária (*middle-grey*) que preservará todos os detalhes da cena. Se ela exceder nove zonas, isso significa que alguns valores, obrigatoriamente, irão ser mapeados para preto puro ou

branco puro.

O fotógrafo ainda pode desejar que algumas partes de sua fotografia sejam mapeadas para uma dessas zonas limite, mesmo não excedendo à nove zonas. Para isso, ele pode utilizar uma técnica conhecida como *dodge-and-burning*. Essa técnica está fora do escopo do presente trabalho, e o leitor interessado em obter maiores detalhes pode consultar [Reinhard02].

3.5.1.2 Algoritmo de *tone-mapping* de Reinhard

Para reproduzir os passos que um fotógrafo realiza ao utilizar o sistema de zonas, incialmente deve ser computada a luminância média da cena (chave). Essa luminância chave é obtida através da *Equação 3.1*:

$$\hat{L} = \exp\left(\frac{1}{N}\sum_{x,y}\log\left(\delta + L(x,y)\right)\right)$$

Equação 3.1: cálculo da luminância média de uma imagem (frame).
***N** é o número de pixels da imagem.*
***L(x,y)** avalia a luminância de um pixel individual.*
***δ** é uma constate de valor muito pequeno, utilizada para evitar a ocorrência de log(0) quando a luminância do pixel for zero.*

Uma vez encontrada a luminância média, esta será mapeada linearmente para a zona média (*middle-grey*), usando a *Equação 3.2*:

$$\tilde{L}(x,y) = \frac{\alpha}{\hat{L}} L(x,y)$$

Equação 3.2: cálculo da luminância mapeada de um pixel da imagem.
***α** é o valor escolhido como zona média (middle-grey).*

Essa equação produz uma escala linear das luminâncias da cena com respeito à luminância média. Esses valores escalados, entretanto, ainda não correspondem à luminâncias aceitas pela faixa dinâmica suportada pelo dispositivo. É necessário, então, aplicar o operador de *tone mapping*, através da *Equação 3.3*:

$$T(x,y) = \frac{\tilde{L}(x,y)}{1 + \tilde{L}(x,y)}$$

Equação 3.3: operador de tone mapping. Ao final, os valores obtidos poderão ser utilizados para representar a luminância na faixa dinâmica de qualquer dispositivo.

Esse operador garante que os valores serão representáveis da faixa dinâmica do dispositivo. Valores altos de $\tilde{L}$ serão mapeados para $\tilde{L}/(1+\tilde{L}) \approx 1$ enquanto que valores baixos serão mapeados para $\tilde{L}/(1+\tilde{L}) \approx 0$, implicando que a saída do operador sempre estará no intervalo [0, 1).

3.5.2 Outros operadores de *tone mapping*

Existem diversos outros operadores e, entre eles, podemos citar: [Stockham72], [Schlick94], [Seetzen04], [Ferwerda96], [Ward97]. O leitor interessado pode encontrar uma comparação entre esses operadores e o operador proporsto por Reinhard em [Reinhard02].

3.6 Formatos de codificação de *HDR*

Como visto anteriormante, imagens com ampla faixa dinâmica podem apresentar uma diferença considerável entre a sua tonalidade mais escura e a mais brilhante. As diversas codificações existentes levam em conta a largura da faixa dinâmica representada (que pode ser expressa em ordens de magnitude), o erro produzido pelo processo de quantização e a diferença de luminância entre valores próximos. Maiores detalhes sobre esses aspectos podem ser encontrados em [Ward02]. Cada formato apresenta vantagens e desvantagens, ajustando-se aos objetivos pelos quais foram criados.

Existem diversos formatos para codificação de imagens HDR. Essa seção irá discutir apenas alguns formatos, apresentando suas vantagens e limitações, mas será dada maior ênfase ao formato RGBE, utilizado na implementação desse trabalho.

3.6.1 IEEE 32bit float

A representação de números em ponto flutuante parece adequada para representar valores de alta faixa dinâmica. As APIs gráficas utilizam ponto flutuante para especificação de cores (embora a maioria dos dispositivos gráficos interpretem valores superiores a 1.0 como sendo o próprio valor 1.0). A representação numérica em ponto flutuante é suportada em hardware, não havendo custos para codificação/decodificação. Entretando, a grande desvantagem é a quantidade necessária para armazenar um pixel (96 bits, considerando 3 canais de cor com 32 bits cada). Devido à natureza da representação de ponto flutuante, os algoritmos de compressão não são capazes de compactar significativamente imagens que utilizam o formato. Além disso, nem todo o hardware gráfico implementa *frame buffers* em ponto flutuante de 32 bits.

Para algumas aplicações, a precisão dessa representação é fundamental e os custos envolvidos são menos importantes do que a qualidade do resultado esperado, como é o caso da obtenção de mapas geográficos de profundidade e sensoramento remoto via satélite.

3.6.2 Pixar Log Encoding

A equipe da Pixar é uma das pioneiras no uso de imagens com alta faixa dinâmica. O interesse da companhia era armazenar as imagens em seu próprio gravador

de filmes. Uma película de filme é capaz de representar uma faixa dinâmica muito maior do que um monitor convencional (4 ordens de magnitude da película, contra duas do monitor). Sua implementação foi feita através de rotinas de codificação e decodificação, dentro da biblioteca TIFF [Leffler2003].

O formato utiliza 3 canais de cores (vermelho, azul e vermelho), codificados logaritmicamente, e cada canal utiliza 11 bits, totalizando 33 bits por pixel (os formatos tradicionais utilizam 3 canais de cores, de 8 bits cada, totalizando 24 bits por pixel). Através dele, a Pixar conseguiu obter uma faixa dinâmica de 3.600:1 (aproximadamente 4 ordens de magnitide), com incrementos de 0.4% entre cada valor de luminância (o maior valor aceitável para que o olho humano não perceba diferenças bruscas é 1%), suficiente para os objetivos da empresa. Entretanto, esse formato é pouco utilizado em computação gráfica, devido à falta de documentação sobre o formato, restrito apenas ao código fonte disponível na biblioteca TIFF.

3.6.3 Industrial Light and Magic OpenEXR

Recentemente, a Industrial Light and Magic disponibilizou e documentou código fonte (em C++) para ler e gravar o formato de imagem HDR desenvolvido por eles, o [IML/OpenEXR]. A empresa já vinha utilizando esse formato para na geração de efeitos especiais há anos.

O formato utiliza uma representação em ponto flutuante de 16 bits (*half*), sendo 1 bit para o sinal, 5 bits para expoente e 10 bits para mantissa (também chamado de S5E10). Esse mesmo formato vem sendo utilizado pela NVIDIA e pela ATI para implementar *frame buffers* de ponto flutuante. Cada canal de cor (RGB) é representado por um *half*, totalizando 48 bits por pixel. A faixa dinâmica representada pelo formato é superior a 10 ordens de magnitude, com uma precisão relativa de 0.1%.

OpenEXR é um código aberto, e sua licença de uso é gratuita. Sua especificação prevê a adição de canais extras (*alpha* e *depth*, por exemplo) e também prevê o uso de mais bits por canal. Além de cobrir uma faixa dinâmica muito maior do que o olho humano é capaz de perceber simultaneamente, seu formato (*half*) já é suportado diretamente nos dispositivos gráficos atuais.

3.6.4 Radiance RGBE8 Encoding

O formato RGBE8 [Ward91] foi desenvolvido para representar imagens HDR no sistema [Ward/Radiance], muito antes de surgir o formato OpenEXR. Radiance é um sistema fisicamente correto para síntese de imagens, construído para computar quantidades fotométricas. Por isso a necessidade de usar alta faixa dinâmica.

Como o nome sugere, o formato utiliza 3 canais de cores (RGB) e um quarto canal (E), todos utilizando 8 bits, totalizando 32 bits por pixel da imagem. Cada canal de cor armazena uma mantissa e o canal extra armazena um expoente comum às mantissas, que serve como fator de escala para as mesmas. O formato, como foi

proposto, garante que o maior valor das componentes RGB estará entre 128 e 255, enquanto os outros dois canais podem variar de 0 à 255. O trecho de código (C-*like*) demonstra como codificar e decodificar o formato RGBE:

```
struct
{
  BYTE R;
  BYTE G;
  BYTE B;
  BYTE E;
} RGBE;

struct
{
  FLOAT R;
  FLOAT G;
  FLOAT B;
} FLOAT3;

FLOAT3 Decode(RGBE encoded)
{
  FLOAT3 decoded;
  INT exp = (encoded.E * 255) - 128;
  FLOAT scale = exp2(exp);
  decoded.R = encoded.R * scale;
  decoded.G = encoded.G * scale;
  decoded.B = encoded.B * scale;
  return(decoded);
}

RGBE Encode(FLOAT3 decoded)
{
  RGBE encoded;
  BYTE maxRG = MAX(decoded.r, decoded.g);
  BYTE maxRGB = MAX(maxRG, decoded.b);
  INT exp = ceil(log2(maxRGB));
  FLOAT scale = exp2(exp);
  encoded.R = decoded.R / scale;
  encoded.G = decoded.G / scale;
  encoded.B = decoded.B / scale;
  encoded.A = (exp + 128) / 255;
  return(encoded);
}

OBSERVAÇÕES:
1. exp2(X) = 2^X
2. log2(X) = log2 X
3. ceil(X) = maior inteiro próximo de X
```

O formato RGBE8 compreende uma faixa dinâmica muito grande, cerca de 76 ordens de magnitude, com uma **precisão média** próxima de 1% (o limite máximo aceitável de modo a "enganar" o olho humano). Embora a faixa representada seja muito grande, a precisão média está muito próxima do máximo acietável (1%) e, na prática, essa faixa dinâmica é muito maior do que aquela que qualquer aplicação precisaria usar, subutilizando muitas ordens de magnitude. Seria melhor se o formato suportasse, na mesma quantidade de bits, uma menor ordem de magnitude, mas com uma precisão melhor (já que, em alguns passos de quantização, a diferença pode chegar à 5%).

Entretanto, o formato, por utilizar números inteiros de 8 bits em cada canal, pode ser perfeitamente acomodado nas representações atuais em RGBA. Isso permite utilizar *frame buffers* tradicionais para armazenar os pixels da imagem HDR, com o acréscimo

de um pouco de computação para codificar/decodificar as cores.

3.7 Sumário

Este capítulo apresentou uma breve descrição das técnicas de **iluminação baseada em imagens**. *Cube maps* e ***light probes*** são duas representações possíveis para imagens utilizadas nessas técnicas.

A captura e o uso de **luz natural** para iluminar objetos sintéticos proporcionou um grande aumento no realismo de imagens sintetizadas [Debevec/FiatLux], e é ideal para realizar a integração de objetos sintéticos em cenas reais. Entretanto, os formatos tradicionais de imagens digitais não suportam toda a faixa de luminância presente no mundo real. Para representar e armazenar esses valores de luminosidade de **alta faixa dinâmica** surgiram novos formatos de imagem, como o **formato RGBE** e o formato OpenEXR.

Dispositivos de exibição gráfica, como monitores e *data-shows*, geralmente não suportam uma faixa de luminosidade muito ampla. Para que imagens em alta faixa dinâmica possam ser exibidas nesses dispositivos tradicionais, a faixa de luminosidade representada na imagem deve ser mapeada para a faixa de luminosidade suportada pelo dispositivo. Esse processo é realizado através de operadores de ***tone mapping***, como é o caso do **operador de *tone mapping* de Reinhard** apresentado na Seção 3.5.1.

Capítulo 4

Transferência de Radiância Pré-computada (PRT)

A **transferência de radiância pré-computada** é uma abordagem para superar o desafio de reproduzir **iluminação global em tempo real**. Técnicas recentes [Sloan-Kautz-Snyder02], originadas do trabalho de [Ramamoorthi-Hanrahan01] para a geração de **mapas de irradiância**, apresentadas nesse capítulo, permitem derivar a equação de *Rendering* para uma fórmula bastante simplificada, que pode ser avaliada em tempo real e implementada no hardware gráfico. Para esse processo, é necessário simplificar a representação da iluminação da cena e avaliar o **transporte de luz** sobre o modelo poligonal. As **harmônicas esféricas**, apresentadas nesse capítulo, representam um conjunto de funções de base que podem ser utilizadas para aproximar/simplificar a **função de iluminação** original. A transferência de radiância pré-computada apresenta vantagens e desvantagens sobre outros algoritmos para simulação iluminação e transporte de luz e uma breve discussão sobre elas será oferecida ao final do capítulo. A última seção sumariza todos os conceitos e detalhes importantes apresentados no capítulo.

4.1 Introdução

A complexidade da iluminação e do processo de transporte de luz em cenas reais faz com que sua simulação implique em elevado esforço computacional, inviabilizando o uso destas em aplicações de tempo real. A **transferência de radiância pré-computada** (do inglês, *precomputed radiance transfer* – **PRT**) procura resolver o processo de transporte de luz em uma etapa de pré-computação para posteriormente utilizar os resultados obtidos na síntese de imagens em tempo real.

A geração de ***irradiance maps*** [Miller-Hoffman84] é uma solução possível para simular o transporte de luz em uma cena, sobre um determinado material. A técnica envolve a geração de *reflection maps* distintos para as componentes difusa e especular de uma BRDF que aproxima as propriedades ópticas do material. Essa técnica ainda é muito explorada em jogos de computadores (*Figura 4.1*), onde esses *irradiance maps* são usualmente transformados em texturas e a serem mapeadas diretamente sobre as superfícies de um modelo geométrico, em vez de serem acessados através de uma

função que recebe um vetor normal. Essas texturas são denominadas *light maps*.

Figura 4.1: À esquerda, imagem do jogo TimeSplitters (Nintendo GameCube – 2002); à direita imagem do jogo God of War (Sony Playstation2 – 2005). Ambos os jogos utilizam light maps para simular o transporte de luz nos elementos estáticos do cenário (paredes, chão, teto, etc.). Extraído de: http://cube.ign.com e http://ps2.ign.com

Embora o uso de *light maps* simule a interação dessa luz com as superfícies de forma eficiente, essa técnica possui limitações. Ela não permite, por exemplo, que as fontes de luz originalmente utilizadas para realizar a pré-computação sejam aletradas ou rotacionadas. Além disso, os objetos não podem ser movidos ou rotacionados. Recentes progressos nas técnicas de transferência de radiância pré-computada permitiram isolar o processo de transporte de luz da condição de iluminação (fontes de luz). Ainda existem, entretanto, limitações quanto à movimentação de objetos na cena, e os primeiros estudos para tratar animação e PRT estão surgindo [Sloan05] [Annen04].

Como mencionado anteriormente, é possível realizar o procedimento de transferência de radiância pré-computada para as componentes especular e difusa. O presente trabalho irá discutir a PRT para a componente difusa, que independe da posição de observação e possui equações mais simples e rápidas de serem calculadas. O resultado obtido pode ser combinado com a técnica tradicional de *reflection mapping* (Seção 3.2) para acrescentar aos objetos características especulares.

4.2 PRT Difusa

O objetivo de realizar PRT difusa é determinar a transferência de luz entre superfícies de objetos difusos. A equação utilizada no cálculo de PRT pode ser derivada a partir da Equação de *Rendering* (*Equação 2.8*):

$$L(x,\vec{\omega}_r) = L_e(x,\vec{\omega}_r) + \int_{\Omega_i} f_r(x,\vec{\omega}_i,\vec{\omega}_r)\, L_i(x,\vec{\omega}_i)\, V(x,x')\, G(x,x')\, d\vec{\omega}_i$$

Equação 4.1: A Equação de Rendering, apresentada na Seção 2.6.

Uma superfície difusora ideal (*Figura 2.7b*) reflete a luz incidente igualmente em todas as direções do hemisfério de luz que a cerca, sendo que a radiância que deixa a superfície não depende da posição do observador. Isso significa que a BRDF dessa superfície pode ser representada por meio da seguinte constante:

$$f_r(x, \vec{\omega}_i, \vec{\omega}_r) = \frac{\rho_d}{\pi}$$

Equação 4.2: BRDF para superfícies difusoras ideais.

onde ρ_d representa a componente de refletividade difusa do material, variando dentro do intervalo [0, 1]. A demonstração da derivação da BRDF constante acima apresentada pode ser encontrada em [Cohen-Wallace93].

Assumindo que a superfície em questão não emite luz, podemos eliminar o termo de emissividade da equação. A BRDF, por ser um valor constante, pode ser retirada de dentro da integral e como trata-se de uma abordagem independente de ponto de vista, a direção de reflexão não precisa ser especificada. Assim, Equação de *Rendering* pode ser reescrita como:

$$L(x) = \frac{\rho_d}{\pi} \int_{\Omega_i} L_i(x, \vec{\omega}_i) V(x, x') G(x, x') d\vec{\omega}_i$$

Equação 4.3: Equação de Rendering reescrita, ignorando o termo de emissividade e assumindo superfícies idealmente difusas.

A equação acima sugere que precisamos integrar a **radiância incidente** ($L_i(x,\omega_i)$), o **termo de visibilidade** ($V(x,x')$) e o **fator geométrico** ($G(x,x')$). Visualmente, tem-se algo semelhante ao que é mostrado na *Figura 4.2*:

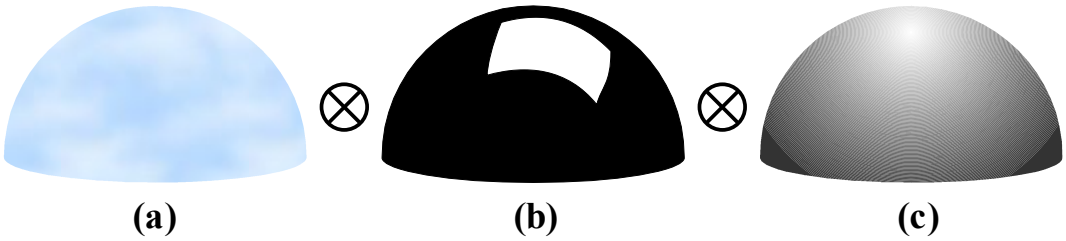


Figura 4.2: (a) a radiância incidente, (b) a visibilidade e (c) o fator geométrico, integrados,são suficientes para determinar como a superfície difusa será iluminada.

A técnica de PRT nos permite combinar o termo de visibilidade com o fator geométrico, separando-os da radiância incidente, conforme mostrado na *Figura 4.3*. Essa combinação é denominada **função de transferência**.

A função de iluminação determina a radiância incidente em uma direção especificada e uma boa representação para esse tipo de função são as *light probes*, que permitem, ainda, o uso de luz natural em alta faixa dinâmica.

Surgem, então algumas questões importantes: a radiância incidente, também chamada de **função de iluminação**, deve poder ser avaliada apenas em função de um vetor de direção; o comportamento da função de transferência precisa ser especificado e resolvido; e a integral precisa ser resolvida em tempo real. As soluções para essas questões serão apresentadas nas próximas seções.

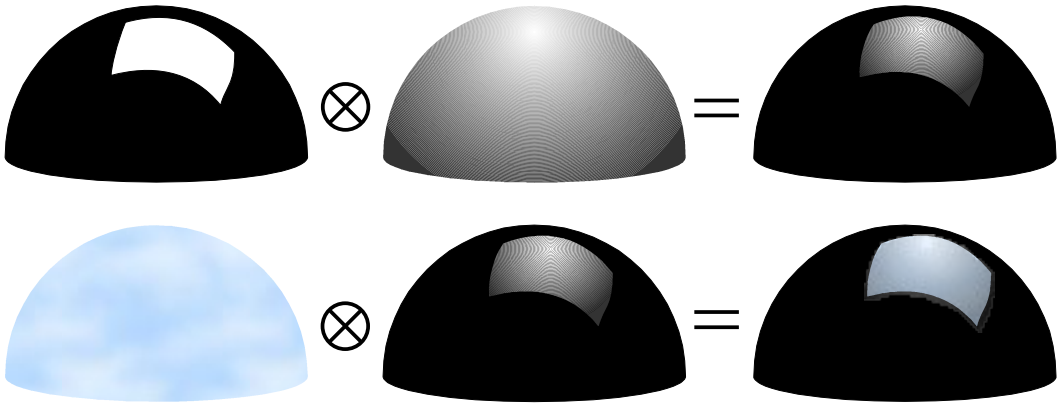


Figura 4.3: O termo de visibilidade é combinado com o fator geométrico, determinando um fator de transferência. Esse termo obtido será, então, combinado com a radiância incidente para determinar a iluminação da superfície.

4.2.1 A Função de Iluminação

Ainda é necessário realizar uma integração em todas as direções (no domínio da função). Devido ao tamanho desse domínio, é interessante criar uma aproximação da função de iluminação. Esse processo pode ser realizado projetando a função de iluminação sobre um conjunto de **funções de base**. Uma aproximação da função de iluminação original pode então ser obtida por meio de uma combinação linear envolvendo apenas alguns termos de mais baixa ordem obtidos com tal projeção. Para obter esses coeficientes, correspondentes à projeção de uma função qualquer sobre uma família de funções de base, pode-se recorrer a métodos numéricos como o **Método de Monte Carlo** [Green03] [Hammersley-Handscomb64].

Utilizando uma função de base para aproximar o sinal luminoso original, a função de iluminação pode ser redefinida:

$$L_i(\vec{\omega}_i) \approx \sum_k l_k \beta_k(\vec{\omega}_i)$$

Equação 4.4: Função de iluminação aproximada, onde os coeficientes (l_k) são obtidos através da projeção da função original sobre as funções de base (β_k).

Substituindo a função de iluminação pela função aproximada (*Equação 4.4*) na *Equação 4.3* obtém-se:

$$L(x) = \frac{\rho_d}{\pi} \int_{\Omega_i} \sum_k l_k \beta_k(\vec{\omega}_i) V(x, x') G(x, x') d\vec{\omega}_i$$

Equação 4.5: Derivação da Equação 4.3 utilizando uma função de iluminação aproximada.

Através da propriedade do cálculo integral que garante que a integral de somas é equivalente à soma de integrais, e respeitando o domínio de integração ($d\omega_i$), podemos re-escrever a *Equação 4.5* como:

$$L(x)=\frac{\rho_d}{\pi}\sum_k I_k \int_{\Omega_i} \beta_k(\vec{\omega}_i) V(x,x') G(x,x') d\vec{\omega}_i$$

Equação 4.6: Derivação da Equação 4.5, aplicando propriedades do cálculo integral.

A integral restante representa a **função de transferência**, que pode ser pré-computada, e será discutida na próxima seção.

4.2.2 A Função de Transferência

A função de transferência irá determinar a complexidade do transporte de luz. Através dela é possível simular sombras, interreflexões e cáusticas, por exemplo. Computacionalmente, o cálculo da função de transferência corresponde à etapa de maior custo em toda a técnica de PRT. Conforme a *Equação 4.6* podemos escrever a função de transferência como:

$$T(x)=\int_{\Omega_i} \beta_k(\vec{\omega}_i) V(x,x') G(x,x') d\vec{\omega}_i$$

Equação 4.7: A função de transferência para superfícies difusas.

Da *Equação 4.7* precebe-se que o produto do termo de visibilidade pelo fator geométrico é **projetado** sobre as mesmas funções de base e no mesmo domínio de integração no qual a função de iluminação também foi projetada. Como resultado dessa projeção, tem-se um outro conjunto de coeficientes. Esses coeficientes, juntamente com àqueles obtidos na projeção da função de iluminação, irão determinar a iluminação final do ponto (x) desejado.

Nas próximas sub-seções, serão apresentadas duas funções de transferência, **sem auto-sombreamento** (*unshadowed*) e **com auto-sombreamento** (*self-shadowed*) e, ainda, uma breve discussão sobre a determinação de outras funções de transferência mais complexas.

4.2.2.1 Função de Transferência sem Auto-somberamento

A função de transferência sem auto-sombreamento considera apenas iluminação direta, desconsiderando a possibilidade do raio incidente (direção) não atingir atingir a superfície, devido a algum obstáculo, como é mostrado na *Figura 4.4.*

Essa configuração pode ser obtida a partir da *Equação 4.7*, simplesmente ignorando o termo de visibilidade, conforme a equação:

$$T(x)=\int_{\Omega_i} \beta_k(\vec{\omega}_i) G(x,x') d\vec{\omega}_i$$

Equação 4.8: A função de transferência sem auto-sombreamento desconsidera o termo de visibilidade.

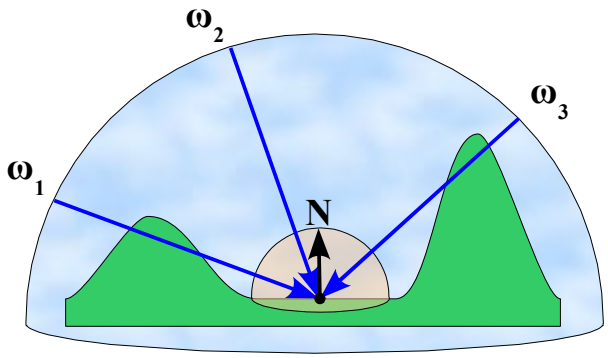


Figura 4.4: a função de transferência sem auto-sombreamento ignora o termo de visibilidade; obstáculos que impediriam a incidência de raios sobre o ponto são desconsiderados.

4.2.2.1 Função de Transferência com Auto-somberamento

Em contraste com a função de transferência sem auto-sombreamento, podemos definir uma outra função que utiliza o termo de visibilidade. Dessa forma, é possível a geração de sombras, devido a oclusão de radiância sobre o ponto na superfície a ser iluminado (*Figura 4.5*). Na prática, a função de visibilidade pode ser implementada através de um traçador de raios (*ray-tracing*). Um raio bloqueado, nessa abordagem, terá sua participação **completamente anulada** no valor de iluminação final do ponto.

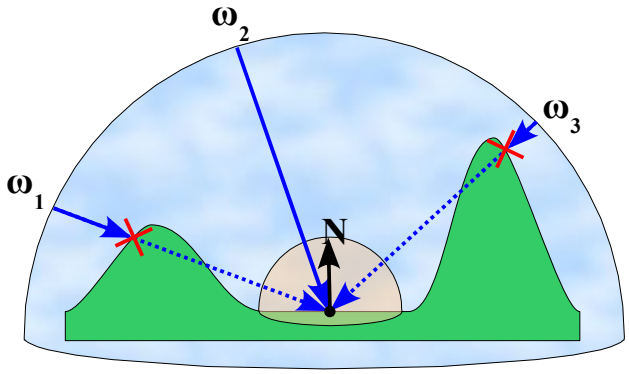


Figura 4.5: a função de transferência com sombreamento mútuo utiliza o termo de visibilidade; raios incidentes são bloqueados por obstáculos, não contribuindo para a iluminação do ponto.

Pode-se, então, denotar a função de transferência com sombreamento mútuo através da equação:

$$T(x) = \int_{\Omega_i} \beta_k(\vec{\omega}_i) V(x, x') G(x, x') d\vec{\omega}_i$$

Equação 4.9: A função de transferência com auto-sombreamento.

4.2.2.3 Outras Funções de Transferência

Diversas outras funções de transferência podem ser definidas para simular

transportes mais detalhados de luz sobre superfícies. Como demonstra a equação geral da função de trasferência (*Equação 4.7*), há apenas duas componentes que podem ser modificadas: o termo de visibilidade e o fator geométrico.

Para a simulação de interreflexões, o termo de visibilidade pode ser o mesmo apresentado na seção anterior, anulando completamente a contribuição de raios bloqueados. Entretanto, raios bloqueados, embora não contribuam diretamente para a iluminação do ponto, podem contribuir de forma indireta. Sendo assim, raios bloqueados darão origem à novos raios, que poderão transportar contribuições indiretas. É importante, ainda, definir condições de parada para as interreflexões para que o tempo de computação não seja exageradamente elevado com pouca contribuição para a iluminação da cena. O fator geométrico também pode ser modificado, modelando relações mais complexas entre as características das superfícies envolvidas.

Refrações e *subsurface scattering*, de modo semelhante às interreflexões, também geram novos raios, mas esses raios serão transmitidos ao invés de refletidos.

4.3 A Equação de *Rendering* para PRT Difusa

Recapitulando a derivação da Equação de *Rendering* iniciada na Seção 4.2 (*Equação 4.3*), e substituindo nela a função de iluminação e a função de transferência anteriormente apresentadas, deriva-se uma fórmula simples e eficiente que pode ser avaliada em tempo real:

$$L(x)=\frac{\rho_d}{\pi}\sum_k l_k T(x)$$

Equação 4.10: derivação da Equação 4.3, utilizando uma função de iluminação aproximada e uma função de transferência.

Como mostrado anteriormente, a função de transferência produz como resultado um vetor de coeficientes que representam a projeção do termo de visibilidade e do termo geométrico sobre as **mesmas funções de base** utilizadas para realizar a projeção da função de iluminação, garantindo que ambas são representadas com o **mesmo número de coeficientes**. Isso significa que o processo de combinação dos coeficientes da função de iluminação aproximada e os coeficientes da função de transferência é o produto escalar entre esses dois vetores:

$$\{\ \vec{L}=(l_0,\ldots,l_k)\mid k\geq 0\ \}$$
$$\{\ \vec{T}=(t_0,\ldots,t_k)\mid k\geq 0\ \}$$
$$L(x)=\frac{\rho_d}{\pi}\vec{L}\cdot\vec{T}$$

Equação 4.11: A Equação 4.10 pode ser expressa através de um produto escalar entre os vetores de coeficientes obtidos através da projeção da função de iluminação e da função de transferência.

O valor constante da BRDF pode ser avaliado juntamente com a pré-computação da função de transferência, simplificando ainda mais a equação de *rendering* para o caso

de PRT difusa:

$$L(x)=\vec{I}\cdot\vec{T}$$

Equação 4.12: A BRDF constante pode ser aplicada diretamente à função de transferência, simplificando ainda mais a Equação 4.11.

A próxima seção irá apresentar uma família de **funções ortogonais** que podem ser utilizadas como funções de base para a projeção da função de iluminação e da função de transferência: as **harmônicas esféricas**.

4.4 Harmônicas Esféricas (SH)

As harmônicas esféricas (do inglês, *Spherical Harmonics* – **SH**) representam um sistema matemático semelhante à transformada de Fourier, mas definido sobre a superfície de uma esfera. As funções de SH utilizam números complexos, mas para PRT, é suficiente aplicar as harmônicas esféricas no domínio real (*Real Spherical Harmonics*). Todas as definições e equações dessa seção referem-se às harmônicas esféricas sobre o domínio real.

O cerne desse sistema são os **polinômios de Legendre**, uma família de polinômios que apresentam a propriedade da **ortonormalidade** (Equação 4.13). Essa propriedade garante que, ao integrar o produto de dois desses, o resultado será 1 se eles forem iguais ou zero, caso contrário:

$$\int P_m(x)\,P_n(x)\,dx=\begin{cases}0 & para\ \ m\neq n\\ 1 & para\ \ m=n\end{cases}$$

Equação 4.13: A propriedade da ortonormalidade de funções.

Intuitivamente, é como se cada uma das funções não interferisse nas demais, ainda que ocupando o mesmo espaço (domínio). Essa propriedade é fundamental para a determinação de funções que podem ser utilizadas como funções de base.

Os polinômios de Legendre são caracterizados por meio de dois parâmetros, ***l*** e ***m***. Usualmente, o parâmetro ***l*** é chamado de **banda**. Os polinômios são definidos através da seguinte expressão:

$$\{\, l\in\mathbb{N}\,\}$$
$$\{\, m\in\mathbb{N}\mid m\leq l\,\}$$
$$\{\, x\in\mathbb{R}\mid -1\leq x\leq 1\,\}$$
$$P_l^m(x)=\begin{cases}x(2m+1)\,P_m^m & para\ \ l=m+1\\ (-1)^m(2m-1)!!(1-x^2)^{m/2} & para\ \ l=m\\ \dfrac{x(2l-1)\,P_{l-1}^m-(l+m-1)\,P_{l-2}^m}{(l-m)} & caso\ contrário\end{cases}$$

*Equação 4.14: Os polinômios de Legendre, identificados pelos parâmetros **l** (banda) e **m**.*

onde o operador *!!* é o **fatorial duplo**, definido recursivamente através da fórmula:

$$x!! = \begin{cases} 1 & para \quad x \leq 1 \\ x(x-2) & para \quad x > 1 \end{cases}$$

Equação 4.15: O fatorial duplo.

Através da definição dos polinômios de Legendre, as harmônicas esféricas são definidas através do seguinte sistema:

$$\{ l \in \mathbb{N} \}$$
$$\{ m \in \mathbb{Z} \mid -l \leq m \leq l \}$$
$$y_l^m(\theta, \phi) = \begin{cases} \sqrt{2} K_l^m \cos(m\phi) P_l^m(\cos(\theta)) & para \quad m > 0 \\ \sqrt{2} K_l^m \sin(-m\phi) P_l^{-m}(\cos(\theta)) & para \quad m < 0 \\ K_l^0 P_l^0(\cos(\theta)) & para \quad m = 0 \end{cases}$$

Equação 4.16: Definição de harmônicas esféricas. Os polinômios de Legendre são o cerne do sistema de equações.

Note que as harmônicas esféricas permitem valores negativos para ***m***, e isso sugere que uma aproximação de função na base das SH utilizando ***n*** bandas terá ***n²*** coeficientes. O termo ***K*** serve apenas como um fator de escala, de forma a normalizar as funções:

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

Equação 4.17: Termo de escala para harmônicas esféricas.

É possível, ainda, mapear as variáveis ***l*** e ***m*** em uma seqüência, indexada por ***i***:

$$\{ y_l^m(\theta, \phi) = y_i(\theta, \phi) \mid i = l(l+1) + m \}$$

*Equação 4.18: As harmônicas esféricas podem ser enumeradas em seqüência, através de um íncide **(i)** em função de **l** e **m**.*

Ao utilizar-se harmônicas esféricas para aproximar funções, a qualidade da aproximação está diretamente relacionada ao número de bandas utilizadas, conforme pode ser observado na *Figura 4.6.*

As harmônicas esféricas garantem que a integração de duas funções projetadas na base SH é equivalente à somar o produto de todos os coeficientes (*Equação 4.19*), devido a propriedade da ortonormalidade:

$$\int_S L(s) T(s) ds \approx \int_S \tilde{L}(s) \tilde{T}(s) = \sum_{i=0}^{l^2} L_i T_i = \vec{L} \cdot \vec{T}$$

Equação 4.19: A propriedade da ortonormalidade garante que a soma do produto dos coeficientes de duas funções projetadas é equivalente à integração das mesmas.

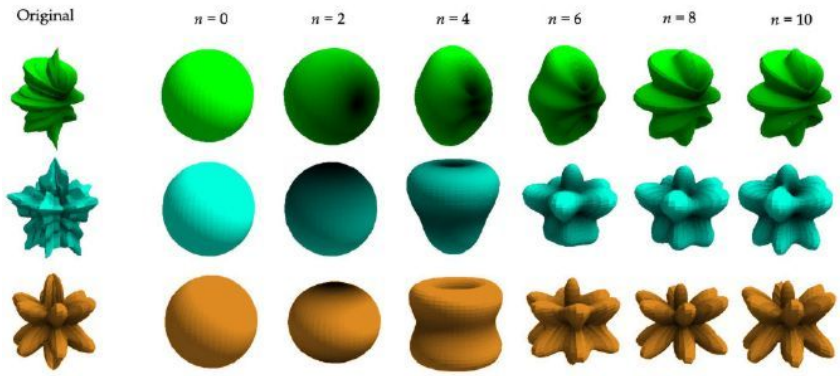


Figura 4.6: Aproximação de funções usando SH: à esquerda temos as funções originais; à direita, a função reconstruída, conforme o número de bandas (n) adotado, que está relacionado à fidelidade da função aproximada com a função original. Extraído de: [Green03]

Note a semelhança da *Equação* com a derivação da Equação de *Rendering* para PRT (*Equação 4.12*). As harmônicas esféricas, garantem, ainda, que a rotação de uma função na base das harmônicas esféricas é equivalente à projeção da função original rotacionada (*Equação 4.20*). O processo de rotação de funções na base das harmônicas esféricas ainda é muito complexo e é difícil inferir uma solução ótima geral para implementá-la [Choi99].

$$R(f(s)) \approx \tilde{f}(R(s))$$

Equação 4.20: Propriedade da rotação das harmônicas esféricas.

4.5 Vantagens e Desvantagens da PRT

Através da PRT é possível utilizar fontes de luz bastante complexas e ela não está limitada apenas ao uso de iluminação capturada de ambientes reais: é possível definir modelos analíticos [Mardaljevic99]. Além disso, o uso de um elevado número de fontes de luz representadas na função de iluminação não adiciona custo computacional algum no procedimento de avaliação em tempo real na Equação (*Equação 4.12*). O uso de PRT ainda permite avaliar modelos sofisticados de transporte de luz, processo que costuma ser o mais caro em algoritmos de iluminação global, e utilizar os resultados de maneira eficiente em tempo real.

Entretanto, há uma série de limitações e idéias a serem exploradas dentro da técnica. Funções de base, como as harmônicas esféricas, são ideais, apenas, para representar as baixas freqüências da função de iluminação; para conseguir capturar, também, componentes de mais alta freqüência, torna-se necessário o uso de funções de base mais flexíveis, como as *wavelets* [Ramamoorthi03], exploradas recentemente no contexto de PRT. Os modelos geométricos ainda possuem pouca flexibilidade dentro da cena, e os primeiros avanços na aplicação de PRT em animação estão surgindo [Sloan05] [Annen04]. A realização de PRT sobre materiais com BRDFs mais complexas envolve procedimentos que ainda não se adaptam facilmente ao hardware gráfico atual, assim como a rotação na base das harmônicas esféricas.

As figuras abaixo exemplificam a técnica de PRT difusa sem auto-sombreamento e com auto-sombreamento, sintetizadas através do protótipo implementando no trabalho.

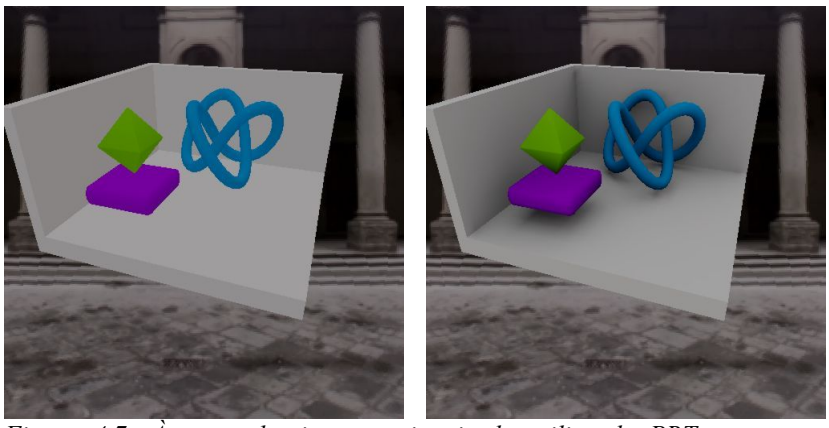


Figura 4.7: À esquerda, imagem sintetizada utilizando PRT sem auto-sombreamento; à direita, a mesma cena renderizada, mas utilizando a função de transferência com auto-sombreamento. As sombras suaves na segunda imagem fornecem um aspecto muito mais realista à imagem.

4.6 Sumário

A técnica de **transferência de radiância pré-computada** utiliza uma versão simplificada da Equação de *Rendering* que pode ser avaliada em tempo real. Para isso, é necessário representar a **função de iluminação** de maneira simplificada, através de sua projeção sobre **funções de base**, como é o caso da família de funções das **harmônicas esféricas**. Para realizar essa projeção, métodos numéricos como **integração de Monte Carlo** podem ser utilizados. A técnica de PRT separa a complexidade de iluminação da complexidade de transporte, através da **função de transferência**.

A técnica de PRT pode ser aplicada tanto com iluminação natural capturada, como com modelos analíticos que definem fontes de luz. A complexidade dessas fontes de luz não acrescenta custos na avaliação da equação. Funções de base como as harmônicas esféricas são ideais apenas para aproximar funções de iluminação com baixas freqüências. A técnica vem ganhando, aos poucos, mais flexibilidade na complexidade de geometria (movimentar e animar modelos) e, também na complexidade de material (BRDFs mais elaboradas).

Capítulo 5

Implementação e Resultados

Este capítulo discute a construção de um protótipo que incorpora as técnicas descritas nos capítulos anteriores, todas implementadas de modo a explorar a programabilidade do hardware gráfico disponível atualmente. Estas técnicas incluem o uso de imagens com alta-faixa dinâmica, transferência de radiância pré-computada para superfícies difusas, *reflection mapping* e o operador de *tone mapping* de Reinhard. O capítulo apresenta um conjunto de imagens que demonstram os resultados obtidos com a implementação realizada e uma breve descrição dos problemas e limitações encontrados.

5.1 Ambiente de Desenvolvimento

O trabalho foi implementado na linguagem de programação C++. Os algoritmos implementandos no hardware gráfico programável foram desenvolvidos na linguagem Cg. A biblioteca gráfica adotada foi o OpenGL e algumas extensões desta foram utilizadas de modo a permitir o uso de *pixel buffers* e *render targets*. Os modelos poligonais foram exportados utilizando NVB Exporter.

5.2 *Tone Mapping*

A implementação do operador de *tone mapping* discutido no Capítulo 3 pode ser resumida nas seguintes etapas, executadas a cada novo quadro (*frame*):

1. Geração da imagem em alta faixa dinâmica;
2. Determinação da luminância média da imagem;
3. Mapeamento dos valores de luminância da imagem original para a faixa dinâmica suportada pelo dispositivo.

A primeira etapa consiste em renderizar a geometria da cena utilizando *environment maps* que armazenam valores em alta faixa dinâmica. Sem o devido mapeamento, os valores de luminância que excedem o limite do dispositivo não serão

representados corretamente. Entretanto, a cena não é renderizada diretamente no *frame buffer*, mas sim em um *pixel buffer*, que será acessado, em forma de textura, nas próximas etapas do algoritmo.

A segunda etapa envolve visitar cada um dos pixels da imagem sintetizada, de modo a resolver a *Equação 3.1*, através de uma filtragem utilizando um processo de redução em $\log_2(N)$ passos. Para isso, o *pixel buffer* utilizado para renderizar a geometria é utilizado como uma textura a qual é mapeada sobre um quadrilátero. Esse quadriláterio será renderizado em outro *pixel buffer* que possui metade do tamanho do *pixel buffer* original. O processo é repetido até que o quadrilátero seja renderizado em um *pixel buffer* de dimensões 1x1, que representa a luminância média da cena. A primeira redução é responsável por transformar as componentes RGB do pixel em valores de luminância e os passos subseqüentes apenas realizam a média dos valores obtidos na primeira etapa.

A última etapa do algoritmo irá realizar o mapeamento dos valores de luminância para o sistema de zonas, de acordo com a luminância média obtida (*Equação 3.2*). A zona intermediária, entretanto, deve ser especificada pelo usuário ou diretamente no programa de fragmento. Com a luminância mapeada para o sistema de zonas, a equação de *tone mapping* (*Equação 3.3*) é aplicada, fornecendo um valor apropriado para ser exibido no monitor de vídeo.

Os programas de vértices e fragmentos abaixo foram desenvolvidos para implementar todos os passos necessários do algoritmo de *tone mapping*:

```
//--------------------------------------------------------------------------
struct app2vertex
{
   float4 f4Position : POSITION;
   float4 f4TexCoord : TEXCOORD;
};
//--------------------------------------------------------------------------
struct vertex2fargment
{
   float4 f4ProjPos  : POSITION;
   float4 f4TexCoord : TEXCOORD0;
};
//--------------------------------------------------------------------------
struct fragment2screen
{
   float4 f4Color    : COLOR;
};
//--------------------------------------------------------------------------
float2 vf2Offsets [4] =
{
   float2( 0.0f,   0.0f),
   float2( 0.0f,  -1.0f),
   float2(-1.0f,  -1.0f),
   float2(-1.0f,   0.0f),
};
//--------------------------------------------------------------------------
float3 f3LumVec = float3(0.2125f, 0.7154f, 0.0721f);
float fDelta = 0.0001f;
float fMiddleGray = 0.72f;
//--------------------------------------------------------------------------
vertex2fargment VQuad
(
   app2vertex IN,
   uniform float4x4 mxModelViewProj
)
{
   vertex2fargment OUT;
```

```
        OUT.f4ProjPos = mul(mxModelViewProj, IN.f4Position);
        OUT.f4TexCoord = IN.f4TexCoord;
        return(OUT);
     }
     //--------------------------------------------------------------------------
     fragment2screen FLuminanceStart
     (
        vertex2fargment IN,
        uniform samplerRECT samTexture
     ) : COLOR
     {
        fragment2screen OUT;
        float fLumAvg = 0.0f;
        for (int i = 0; i < 4; i++)
        {
          float4 f4Color = texRECT(samTexture, IN.f4TexCoord.xy +
vf2Offsets[i]);
          float3 f3Color = rgbe2float(f4Color);
          float fLum = dot(f3Color, f3LumVec);
          fLumAvg += log(fLum + fDelta);
        }
        fLumAvg /= 4.0f;
        OUT.f4Color = float2rgbe(float3(fLumAvg, fLumAvg, fLumAvg));
        return(OUT);
     }
     //--------------------------------------------------------------------------
     fragment2screen FLuminanceIter
     (
        vertex2fargment IN,
        uniform samplerRECT samTexture
     ) : COLOR
     {
        fragment2screen OUT;
        float3 f3Average = float3(0.0f, 0.0f, 0.0f);
        for (int i = 0; i < 4; i++)
        {
          float4 f4Color = texRECT(samTexture, IN.f4TexCoord.xy +
vf2Offsets[i]);
          float3 f3Color = rgbe2float(f4Color);
          f3Average += f3Color;
        }
        f3Average /= 4.0f;
        OUT.f4Color = float2rgbe(f3Average);
        return(OUT);
     }
     //--------------------------------------------------------------------------
     fragment2screen FLuminanceFinal
     (
        vertex2fargment IN,
        uniform samplerRECT samTexture
     ) : COLOR
     {
        fragment2screen OUT;
        float3 f3Average = float3(0.0f, 0.0f, 0.0f);
        for (int i = 0; i < 4; i++)
        {
          float4 f4Color = texRECT(samTexture, IN.f4TexCoord.xy +
vf2Offsets[i]);
          float3 f3Color = rgbe2float(f4Color);
          f3Average += f3Color.r;
        }
        f3Average /= 4.0f;
        float fLumAvg = exp(f3Average.r);
        OUT.f4Color = float2rgbe(float3(fLumAvg, fLumAvg, fLumAvg));
        return(OUT);
     }
     //--------------------------------------------------------------------------
     fragment2screen FragToneMap
     (
        vertex2fargment IN,
        uniform samplerRECT samScene,
        uniform samplerRECT samBloom,
        uniform sampler2D samLuminance
     ) : COLOR
     {
```

```
   fragment2screen OUT;
   float4 f4Scene = texRECT(samScene, IN.f4TexCoord.xy);
   float4 f4Pixel = tex2D(samLuminance, float2(0.5f, 0.5f));
   float3 f3Scene = rgbe2float(f4Scene);
   float3 f3Pixel = rgbe2float(f4Pixel);
   float fLumAvg = f3Pixel.r;
   f3Scene *= (fMiddleGray/(fLumAvg+0.0001f));
   f3Scene /= (1.0f + f3Scene);
   OUT.f4Color = float4(f3Scene, 1.0f);
   return(OUT);
}
//--------------------------------------------------------------------------
```

Quando há uma variação muito grande na luminância média entre um *frame* e outro, um esquema de **luminância adaptativa** pode ser adotado. Ele aproxima gradualmente a luminância do *frame* anterior com a do *frame* atual. O efeito é semelhante àquele realizado pelo sistema visual humano, que se adapta gradativamente à luminosidade das cenas. Para sua implementação, dois novos *pixel buffers* são necessários, ambos com dimensões 1x1. Dessa maneira, a luminância média sempre será avaliada em um *pixel buffer*, enquanto o segundo irá armazenar a luminância do último *frame* renderizado e o terceiro irá adaptar a luminância do *frame* anterior com o atual. Esses dois últimos são trocados a cada novo frame (a luminância adaptada passa a ser a luminância do último *frame* e a luminância antiga passa a armazenar a nova luminância adaptada). O programa de fragmento abaixo implementa esse processo.

```
//--------------------------------------------------------------------------
struct vertex2fargment
{
   float4 f4ProjPos  : POSITION;
   float4 f4TexCoord : TEXCOORD0;
};
//--------------------------------------------------------------------------
struct fragment2screen
{
   float4 f4Color    : COLOR;
};
//--------------------------------------------------------------------------
fragment2screen FragAdaptation
(
   vertex2fargment IN,
   uniform sampler2D samOldLum,
   uniform sampler2D samNewLum,
   uniform float fFrameTime
) : COLOR
{
   fragment2screen OUT;

   float4 f4NewLum = tex2D(samNewLum, float2(0.5f, 0.5f));
   float4 f4OldLum = tex2D(samOldLum, float2(0.5f, 0.5f));

   float3 f3NewLum = rgbe2float(f4NewLum);
   float3 f3OldLum = rgbe2float(f4OldLum);

   float fAdaptation = f3OldLum.r + (f3NewLum.r - f3OldLum.r) * (1.0f -
                       pow(0.98f, 60.0f*fFrameTime));

   return(OUT);
}
//--------------------------------------------------------------------------
```

5.3 Codificação e Decodificação RGBE

Neste trabalho foram utilizados *pixel buffers* no formato RGBA (8 bits por canal).

Para que esses *buffers* possam armazenar valores em alta faixa dinâmica, o formato de imagem RGBE é o que melhor se adapta às características desejadas. Os programas de fragmentos envolvidos no processo de *tone mapping* necessitam realizar a codificação e decodificação dos valores armazenados para o seu correto funcionamento. As duas rotinas abaixo fornecem o suporte à codificação/decodificação do formato RGBE, no hardware gráfico programável:

```
//------------------------------------------------------------------------
float3 rgbe2float(float4 f4Encoded)
{
   float3 f3Decoded;
   float fExp = f4Encoded.a * 255 - 128;
   f3Decoded = f4Encoded.rgb * exp2(fExp);
   return(f3Decoded);
}
//------------------------------------------------------------------------
float4 float2rgbe(float3 f3Decoded)
{
   float4 f4Encoded;
   float fMaxChannel = max(max(f3Decoded.r, f3Decoded.g), f3Decoded.b);
   float fExp = ceil(log2(fMaxChannel));
   f4Encoded.rgb = f3Decoded / exp2(fExp);
   f4Encoded.a = (fExp + 128) / 255;
   return(f4Encoded);
}
//------------------------------------------------------------------------
```

5.4 PRT Difusa

Para implementar PRT difusa no hardware gráfico programável, os coeficientes da função de iluminação aproximada e da função de transferência devem ser informados ao programa de vértices (isto é necessário porque a transferência de radiância foi pré-computada para cada vértice do modelo poligonal). O programa de vértices irá, então, resolver a Equação de *Rendering* para PRT Difusa (*Equação 4.12*) através do produto escalar entre os coeficientes informados. Cada canal de cor (tanto para a função de iluminação quanto para a função de transferência) terá seu próprio vetor de coeficientes. O resultado do produto escalar define a cor difusa do vértice, que será interpolada durante o processo de rasterização. Esta interpolação é aceitável porque a função de iluminação aproximada através das harmônicas esféricas é suave, sendo caracterizada apenas pelas baixas freqüências da função original. O programa de vértice abaixo implementa PRT difusa para $l^2 = 9$ coeficientes.

```
//------------------------------------------------------------------------
struct app2vertex
{
   float4 f4Position : POSITION;
};
//------------------------------------------------------------------------
struct vertex2fargment
{
   float4 f4ProjPos  : POSITION;
   float4 f4Color    : COLOR;
};
//------------------------------------------------------------------------
vertex2fargment VertexRefMap
(
   app2vertex IN,
   float3 vf3TransferCoeffs [9],
   uniform float3 vf3LightCoeffs [9],
   uniform float4x4 mxModelViewProj
```

```
    )
    {
       vertex2fargment OUT;
       OUT.f4ProjPos = mul(mxModelViewProj, IN.f4Position);
       OUT.f4Color = float4(0.0f, 0.0f, 0.0f, 1.0f);
       for (int i = 0; i < 9; i++)
       {
          OUT.f4Color.r += vf3TransferCoeffs[i].r * vf3LightCoeffs[i].r;
          OUT.f4Color.g += vf3TransferCoeffs[i].g * vf3LightCoeffs[i].g;
          OUT.f4Color.b += vf3TransferCoeffs[i].b * vf3LightCoeffs[i].b;
       }
       return(OUT);
    }
    //-------------------------------------------------------------------------
```

5.5 *Reflection Mapping*

A técnica de *reflection mapping* é simples de ser implementada no hardware gráfico programável. A linguagem Cg oferece, em sua biblioteca padrão, rotinas paa calcular vetores refletidos a partir de um vetor incidente e um vetor normal. O trabalho utiliza *light probes* para representar *environment maps*, e é necessária uma rotina para acesso a textura representada através de uma *light probe* [Debevec02]. Os programas abaixo implementam, respectivamente, a função de acesso a textura e o programa de fragmento que realiza *reflection mapping*.

```
    //-------------------------------------------------------------------------
    float4 texPROBE(sampler smpProbe, float3 f3Direction)
    {
      float4 f4Color;
      float d = sqrt(f3Direction.x*f3Direction.x +
f3Direction.y*f3Direction.y);
      float r = 0;
      if (d != 0)
        r = (1.0f/3.141592654f/2.0f)*acos(f3Direction.z)/d;
      float2 f2TexCoord;
      f2TexCoord.x = 0.5 + f3Direction.x * r;
      f2TexCoord.y = 0.5 + f3Direction.y * r;
      f4Color = tex2D(smpProbe, f2TexCoord);
      return(f4Color);
    }
    //-------------------------------------------------------------------------
    struct vertex2fargment
    {
      float4 f4ProjPos   : POSITION;
      float4 f4WorldPos  : TEXCOORD0;
      float4 f4Normal    : TEXCOORD1;
      float4 f4Color     : COLOR;
    };
    //-------------------------------------------------------------------------
    struct fragment2screen
    {
      float4 f4Color           : COLOR;
    };
    //-------------------------------------------------------------------------
    fragment2screen FragmentRefMap
    (
      vertex2fargment IN,
      uniform float3 f3EyePos,
      uniform sampler2D samProbe
    ) : COLOR
    {
      fragment2screen OUT;
      float3 f3WorldPos = IN.f4WorldPos.xyz;
      float3 f3Normal = normalize(IN.f4Normal.xyz);
      float3 f3LookDir = normalize(f3WorldPos - f3EyePos);
      float3 f3Reflected = reflect(f3LookDir, f3Normal);
```

```
    float4 f4ReflectColor = texPROBE(samProbe, f3Reflected);
    float3 f3ReflectColor = rgbe2float(f4ReflectColor);
    OUT.f4Color = float2rgbe(f3ReflectColor, 1.0f);
    return(OUT);
}
//---------------------------------------------------------------------
```

5.6 Projeção da Função de Iluminação e Transferência

Para projetar a função de iluminação e a função de transferência na base das harmônicas esféricas é necessário integrá-las cada uma das funções de base, em todo o domínio das mesmas. Dessa forma, para obter um coeficiente específico, a seguinte expressão deve ser avaliada:

$$c_l^m = \int_{\Omega} f(\vec{\omega})\, y_l^m(\vec{\omega})\, d\,\vec{\omega}$$

Para que a projeção seja independente da definição da função envolvida, uma solução para essa integração é a **Integral de Monte Carlo** [Green03] [Hammersley-Handscomb64], permitindo realizar essa projeção através de um procedimento genérico, que independe da definição da função, avaliando a expressão para um conjunto de direções especificadas. Esse conjunto de direções é obtido através da geração de amostras de pontos sobre a superfície de uma esfera de raio unitário, centrada na origem. Cada uma dessas direções possui associada a si uma coordenada esférica *(θ, φ)*, uma coordenada cartesiana (x, y, z) e uma lista de valores (obtidos através da avaliação das harmônicas esféricas para aquela direção, nas bandas desejadas), conforme apresentado abaixo:

$$A_i = \{\, (\theta_i, \phi_i),\ (Dx_i, Dy_i, Dz_i),\ (y_0^0(\theta_i, \phi_i), \ldots, y_n^n(\theta_i, \phi_i)) \,\}$$

onde n é o número de bandas desejado. Essas amostras são geradas sobre um plano, que representa a superfície da esfera no espaço bidimensional. As coordenadas dessa superfície 2D são transformadas em coordenadas esféricas e estas em coordenadas cartesianas em 3D:

$$A_i = (a_i, b_i)$$
$$(\theta_i, \phi_i) = (2\,acos(\sqrt{(1-a_i)}), 2\pi b_i)$$
$$(Dx_i, Dy_i, Dz_i) = (\sin(\theta_i)\cos(\phi_i), \sin(\theta_i)\sin(\phi_i), \cos(\theta_i))$$

O código abaixo implementa a geração de uma grade de ***NxN*** amostras sobre a superfície de uma esfera de raio unitário:

```
int Sampler::samples;
int Sampler::bands;
vector<Sample> Sampler::vSamples;
Sampler(int N, int bands)
{
   samples = N*N;
   bands = bands;
   for (int i = 0; i < N; i++)
   {
      for (int j = 0; j < N; j++)
```

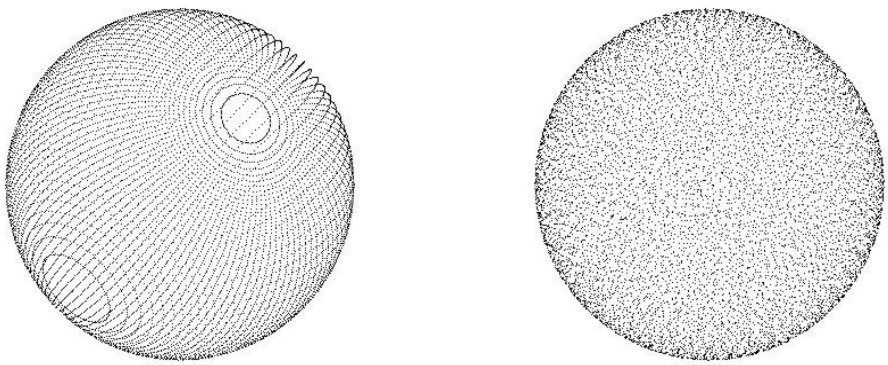
```
        {
          float a = (float) i / (float) N;
          float b = (float) j / (float) N;
          vSamples.push_back(new Sample(a, b, bands));
        }
      }
    }

    float Sample::theta;
    float Sample::phi;
    float Sample::Dx;
    float Sample::Dy;
    float Sample::Dz;
    vector<float> Sample::vSHEval;
    Sample (float a, float b, int bands)
    {
      float theta = 2.0f*acos(sqrt(1.0f - a));
      float phi = 2.0f*PI*b;
      Dx = sin(theta)*cos(phi);
      Dy = sin(theta)*sin(phi);
      Dz = cos(theta);
      for (int l = 0; l < bands; l++)
      {
        for (int m = -l; m <= l; m++)
        {
          vSHEval.push_back(y(l, m, theta, phi));
        }
      }
    }
```

As amostras geradas com o código acima possuem uma variância constante entre si. Para fornecer às amostras um caráter mais aleatório, é possível acrescentar aos pares *(a, b)* um valor arbitrário (r_a, r_b), diminuindo a variância entre as amostras:

```
    float a = (i + random()) / (float) N;
    float b = (j + random()) / (float) N;
```

O par (r_a, r_b) a ser adicionado não deve interferir no espaço das amostras vizinhas, o que sugere que a seguinte propriedade deve ser obedecida:

$$(a_i, b_i)+(r_{ai}, r_{bi})<(a_{i+1}, b_{i+i})$$



Amostras geradas com variância constante (esquerda) e com variância reduzida (direita).

Essa propriedade é garantida se a rotina de geração de números randômicos produzir valores no intervalo [0,1). O código abaixo apresenta uma possível implementação dessa rotina de modo a respeitar a propriedade especificada:

```
float random()
{
  int r = rand();
  r = r % 100000;
  return((float) r / 100000.0f);
}
```

Com as amostras geradas, é possível resolver a **Integral de Monte Carlo**:

$$\int f(x)\,dx \approx \frac{1}{T}\sum_{i=1}^{T}\frac{f(x_i)}{p(x_i)}$$

e ***p(x)*** representa uma função de probabilidades associada à ocorrência da variável x. A probabilidade de amostrar um ponto qualquer na superfície de uma esfera de raio unitário é a mesma para todas as amostras, implicando que:

$$p(x)=\frac{1}{4\pi}$$

onde 4π é a área da superfície da esfera (4π esterradianos). A função de probabilidade pode ser reescrita como:

$$w(x)=\frac{1}{p(x)}=\frac{1}{\frac{1}{4\pi}}=4\pi$$

Isso sugere que a seguinte expressão deve ser avaliada:

$$\frac{1}{T}\sum_{i=1}^{T} f(x_i)\,4\pi=\frac{4\pi}{T}\sum_{i=1}^{T} f(x_i)$$

O trecho de código abaixo implementa essa expressão:

```
vector<float> MonteCarlo::vResult;
MonteCarlo(Function& f, Sampler& sampler)
{
  float weight = 4*PI;
  int T = sampler.samples;
  float n = sampler.bands;
  vResult.resize(n*n);
  for (int i = 0; i < vResult.size(); i++)
  {
    vResult[i] = 0.0f;
  }
  for (int i = 0; i < T; i++)
  {
    for (int i = 0; i < n*n; i++)
    {
      Sample& sample = *(sampler.vSamples[i]);
      vResult[i] += f(sample) * sample.vSHEval[i];
    }
  }
  for (int i = 0; i < vResult.size(); i++)
```

```
    {
      vResult[i] *= weight/T;
    }
  }
```

No código acima, Function define uma interface abstrata para a implementação de uma função que seja capaz de avaliar o resultado para uma determinada amostra. O trecho de código abaixo define a declaração dessa interface:

```
class Function
{
  public:
  virtual float operator () (Sample&)=0;
};
```

Para uma função de acesso a uma *light probe*, a função de iluminação não retorna um valor único mas sim uma tripla, a cor correspondente, naquela direção. Para isso, a interface pode ser ampliada do seguinte modo:

```
class Function
{
  public:
  virtual float operator () (Sample&) { return(0.0f) };
  virtual float* operator () (Sample&, float*) { return(NULL) };
};
```

Uma implementação da função de acesso à *light probes* pode ser realizada de modo semelhante ao código abaixo:

```
class LightProbeFunction : virtual public Function
{
  private:
    HDRImage image

  public:
  float* operator () (Sample& sample, float* out)
  {
    float direction [3] = {sample.Dx, sample.Dy, sample.Dz};
    image.GetPixel(out, direction);
    return(out);
  }
};
```

A rotina de acesso a *light probes* (**image.GetPixel()**) na CPU é implementada de modo análogo à rotina implementada na GPU, apresentada na Seção 5.5.

Para resolver, então, a Integral de Monte Carlo em funções de iluminação representadas através de *light probes*, deve-se utilizar o seguinte trecho de código, que resulta em um vetor de coeficientes para cada canal de cor:

```
for (int i = 0; i < n*n; i++)
{
  Sample& sample = *(sampler.vSamples[i]);
  float yi = sample.vSHEval[i];
  float color [3];
  f(sample, color);
  vResult[i][0] += color[0] * yi; // R
  vResult[i][1] += color[1] * yi; // G
  vResult[i][2] += color[2] * yi; // B
}
```

A função de transferência utiliza uma abordagem análoga. A Integral de Monte

Carlo deve ser avaliada para cada ponto onde for avaliada a função de transferência e o presente trabalho avalia a função a cada vértice do modelo geométrico e. Para a função de transferência sem auto-sombreamento, o seguinte código deve ser executado para cada vértice do modelo:

```
for (int i = 0; i < T; i++)
{
  Sample& sample = *(sampler.vSamples[i]);
  float direction [3] = {sample.Dx, sample.Dy, sample.Dz}
  float G = dot(direction, vertex.normal);
  if (G > 0.0f)
  {
    for (int i = 0; i < n*n; i++)
    {
      float yi = sample.vSHEval[i];
      float transfer [3] = { brdf.r*G, brdf.g*G, brdf.b*G };
      vResult[i][0] += transfer[0] * yi;
      vResult[i][1] += transfer[1] * yi;
      vResult[i][2] += transfer[2] * yi;
    }
  }
}
```

onde G é a avaliação do termo geométrico descrito na equação da função de transferência, implementado, aqui, através do coseno entre a direção de incidência e a normal do vértice (produto escalar - ***dot***) e a BRDF é avaliada juntamente com a função de transferência. O teste realizado (G > 0) ignora a contribuição de direções incidentes que não pertencem ao hemisfério de luz do vértice.

Para a função de transferência com auto-sombreamento, o termo de visibilidade é avaliado na função de transferência. O código acima é então ampliado da seguinte maneira:

```
if (G > 0.0f)
{
  if (Visibility(scene, vertex.position, direction))
  {
    for (int i = 0; i < n*n; i++)
    {
      float yi = sample.vSHEval[i];
      float transfer [3] = { brdf.r*G, brdf.g*G, brdf.b*G };
      vResult[i][0] += transfer[0] * yi;
      vResult[i][1] += transfer[1] * yi;
      vResult[i][2] += transfer[2] * yi;
    }
  }
}
```

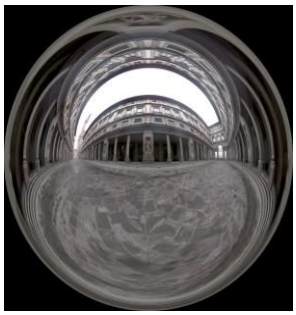
A implementação do termo de visibilidade utilizada no trabalho determina se uma direção é bloqueada percorrendo toda a cena e testando a ocorrência de intersecções entre a direção incidente e cada um dos triângulos que compõem a cena. A implementação do teste de intersecção de triângulos e raios é baseada no algoritmo proposto por [Möller-Trumbore]. O código abaixo realiza esse teste:

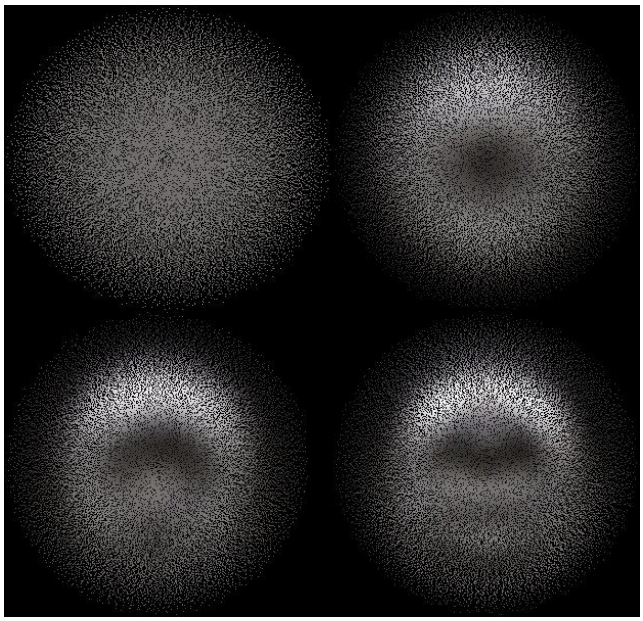
```
bool RayIntersectsTriangle
(float* p, float* d, float* v0, float* v1, float* v2)
{
  float e1 [3] = { v1[0] - v0[0],  v1[1] - v0[1], v1[2] - v0[2] };
  float e2 [3] = { v2[0] - v0[0],  v2[1] - v0[1], v2[2] - v0[2] };
  float h [3];
  cross(h, d, e2);
  float a = dot(e1, h);
  if (a > -0.00001f && a < 0.00001f)
```

```
        return(false);
      float f = 1.0f / a;
      float s [3] = { p[0] - v0[0],  p[1] - v0[1], p[2] - v0[2] };
      float u = f * dot(s, h);
      if (u < 0.0f || u > 1.0f)
        return(false);
      float q [3];
      cross(q, s, e1);
      float v = f * dot(d, q);
      if (v < 0.0f || u + v > 1.0f)
        return(false);
      float t = dot(e2, q)*f;
      if (t < 0.0f)
        return(false);
      return(true);
    }
```

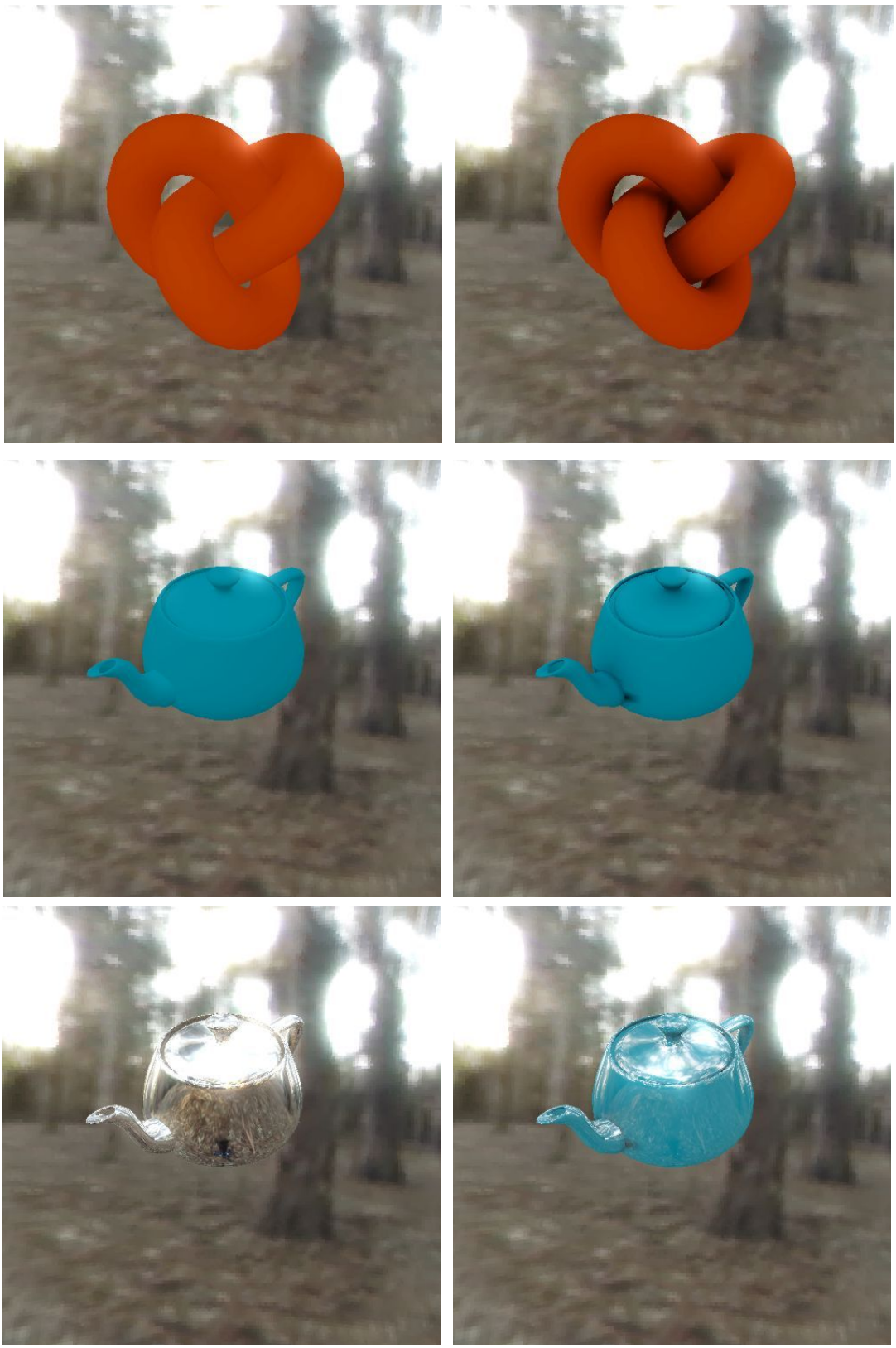
5.7 Exemplos de Imagens

Esta seção apresenta uma série de imagens obtidas com o protótipo implementado que ilustram a aplicação das várias técnicas discutidas.

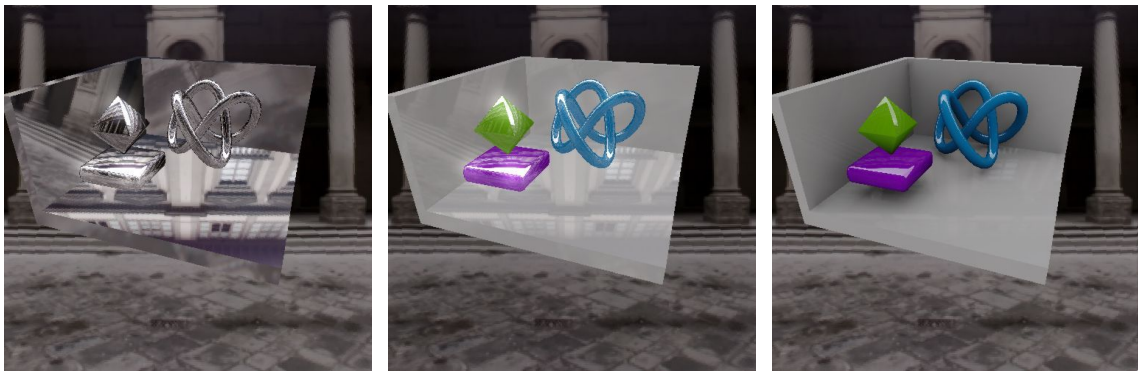




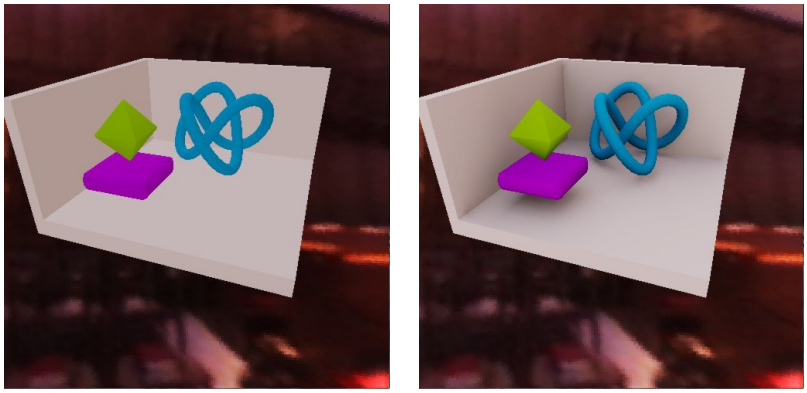
As figuras acima representam a aproximação da função de iluminação original (superior), utilizando 200x200 amostras reconstruídas, respectivamente, utilizando 1, 3, 5 e 7 bandas.



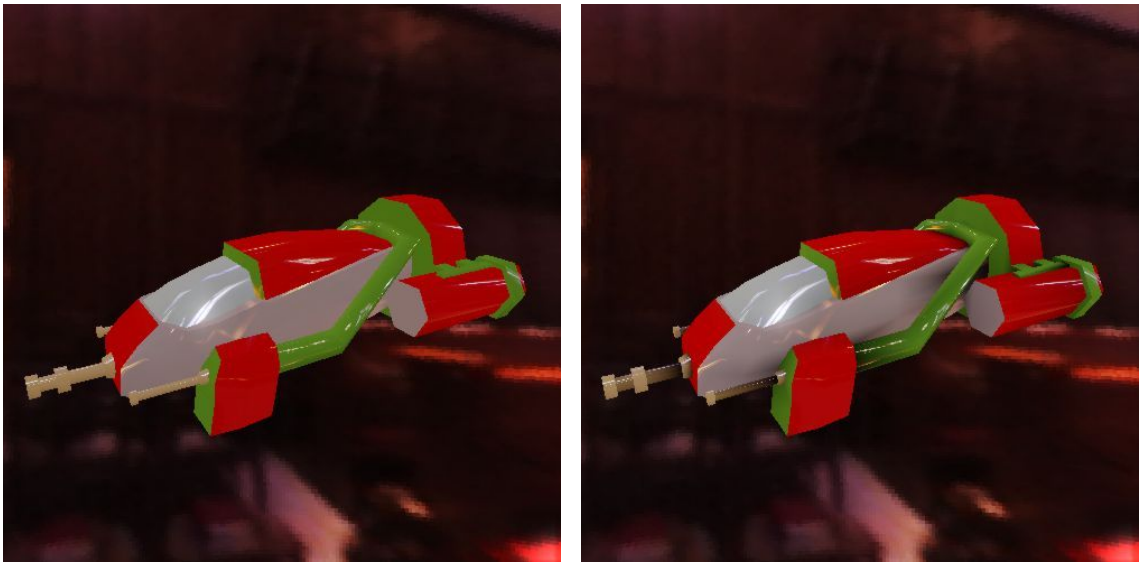
No topo, um toróide renderizado sem considerar o termo de visibilidade (à esquerda) e o mesmo toróide considerando o termo de visibilidade (à direita). No centro, uma *teapot* renderizada sem e com auto-sombreamento (à esquerda e à direita, respectivamente). Abaixo, a mesma *teapot* renderizada apenas com *reflection mapping* (à esquerda) e combinando PRT com auto-sombreamento e *reflection mapping* (à direita).



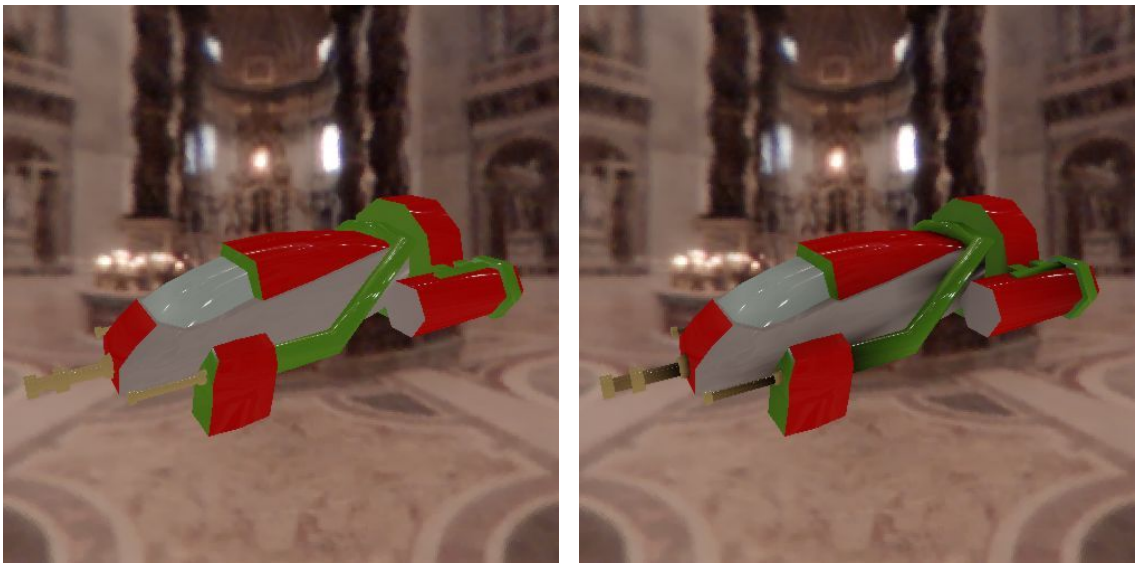
A primeira imagem (à esquerda) foi renderizada apenas com *reflection mapping*. A segunda (ao centro) foi sintetizada mesclando PRT difusa (sem auto-sombreamento) e *reflection mapping*. A última (à direita) apresenta auto-sombreamento e considera menos contribuição especular que as demais.



A mesma cena renderizada anteriormente, mas em ambiente distinto. As imagens foram sintetizadas considerando apenas a contribuição difusa. À esquerda, sem auto-sombreamento; à direita, com auto-sombreamento.



Um *hovercraft* imerso em um ambiente. Na primeira imagem (à esquerda) ele foi renderizado sem auto-sombreamento. Nas segunda imagem (à direita) foi considerado o termo de visibilidade, produzindo sombras suaves devido à oclusões.



O mesmo *hovercraft* renderizado anteriormente, nas mesmas condições (funções de transferência), mas em um ambiente diferente. Note como a iluminação das superfícies do modelo é muito diferente daquela apresentada anteriormente.

5.8 Dificuldades e Limitações Encontradas

Um dos problemas mais marcantes durante o desenvolvimento deste trabalho foi a falta de documentação das componentes de software utilizadas. A documentação oferecida para as extensões do OpenGL não é satisfatória e muito tempo foi dispendido tentando obter seu correto funcionamento. Além disso, decisões de projeto foram tomadas devido à falta de documentação disponível, como foi o caso da adoção do formato RGBE, visto que, no período de projeto, não foi encontrada nenhuma especificação ou exemplo para a adoção de *pixel buffers* em ponto flutuante. De modo semelhante, modelos geométricos carregados no protótipo através da biblioteca NVB Exporter apresentavam comportamento inesperado e a falta de documentação acarretou em dificuldades para apontar os problemas.

O pacote Cg 1.3 apresentou erros de ligação quando utilizado com o compilador gcc/g++ (GNU), o que forçou a escolha do compilador C da Microsoft. De modo semelhante, uma *Console Project* no Microsoft Visual Studio .NET 2003 produzia erros de ligação com a biblioteca Cg (*Could not load type _CGcontext from assembly*). Esse problema foi resolvido criando-se uma *Win32 Console Project*, mas nenhma documentação previa esse comportamento anormal e nenhum fórum apresentou soluções ou explicações para o problema, embora outras pessoas já tivessem relatado esse erro.

Durante a implementação do processo de luminância adaptativa, *frames* totalmente escuros apareciam a cada 2 *frames* renderizados e isso, obviamente, atrapalhava enormemente o algoritmo de luminância adaptativa. Nenhum erro era acusado pela biblioteca OpenGL ou pelo Cg *Runtime*, e o problema desapareceu após uma longa investigação no uso dos *pixel buffers* e *render targets*. A solução foi reiniciar

o canal de multitextura (GL_ARB_TEXTURE0) sempre que um desses canais fosse utilizado.

Outro problema que consumiu um tempo considerável no andamento do trabalho foi o fato de que o *target* de textura GL_TEXTURE_RECTANGLE_NV deve ser referenciado utilizando coordenadas de texturas absolutas em contraste com as coordenadas de textura relativas (valores entre 0 e 1) como comumente é referenciado.

Durante o desenvolvimento do programa de vértice para avaliar a PRT difusa, uma limitação no ambiente Cg foi encontrada: as entradas do programa são alocadas em interpoladores (*interpolators*), mas estes podem ser especificados apenas em um número limitado (*error C5041: cannot locate suitable resource to bind parameter "<null atom>"*). A associação de semânticas (TEXCOORD, COLOR, PSIZE, ...) para o parâmetro dos coeficientes resultava no *rendering* incorreto dos modelos geométricos. A solução foi manter o número de coeficientes dentro do limite aceitável (9 coeficientes, o que, para PRT difusa, é aceitável, visto que por ser uma aproximação em baixa freqüência, as componentes mais relevantes encontram-se nesta faixa). Outra abordagem, que também não obteve êxito, foi codificar esses coeficientes em uma textura e acessá-la no programa de vértices ou fragmentos. A rotina de acesso a essa textura não parecia estar retornando os valores esperados, provavelmente por alguma aproximação nos valores da coordenada de textura especificados ao realizar o acesso.

Capítulo 6

Conclusões e Trabalhos Futuros

A técnica de transferência de radiância pré-computada para superfície difusas proporciona a obtenção de um elevado grau de realismo em imagens sintéticas geradas em tempo real. A função de transferência incorporando sombreamento é capaz de reproduzir sombras suaves, característica presente em cenas iluminadas por fontes de luz de área. A técnica também permite o uso de fontes de luz complexas sem qualquer custo adicional em sua avaliação em tempo real. Além disso, se comparada ao modelo de iluminação local proposto por Phong, ela elimina a necessidade da utilização da componente ambiente.

A mesma luz natural capturada e utilizada para realizar PRT difusa pode ser aplicada em outros algoritmos, de modo a acrescentar um caráter especular às superfícies. Tais efeitos podem ser obtidos com o uso da técnica de *reflection mapping*. Entretanto, o uso de luz natural exige procedimentos para sua correta reprodução em monitores de vídeo convencionais, obtidos através do uso de algoritmos de *tone mapping*. O algoritmo proposto por Reinhard considera todo o contexto da imagem a ser exibida para determinar o resultado do procedimento de *tone mapping*, e acaba apresentando resultados práticos bastante realistas. Este algoritmo pode ser implementado no hardware gráfico programável realizando *rendering* em múltiplos passos, juntamente com o esquema de luminância adaptativa, produzindo resultados ainda mais realistas para a imagem final sintetizada. Também é possível mapear para o hardware algoritmos para tratamento de PRT difusa, embora existam limitações na abordagem empregada, devido ao limite de parâmetros do programa de vértices.

Embora a técnica de PRT difusa seja capaz de adicionar um nível elevado de realismo às imagens, a função de iluminação aproximada, através do uso de harmônicas esféricas, não é ideal para reproduzir as altas freqüências da função de iluminação original. Sem essas freqüências, não é possível reproduzir sombras mais abruptas produzidas por pontos muito luminosos existentes na função de iluminação. Além disso, ainda não é possível movimentar e animar livremente modelos geométricos na cena.

Como trabalhos futuros, pode ser realizada a substituição das harmônicas esféricas por outras funções capazes de extrair as altas freqüências da função de iluminação, como as *wavelets*. A rotação da função de iluminação projetada pode ser implementada, complementando o trabalho. Além disso, a função de transferência pode ser aprimorada

de modo a realizar a simulação de cáusticas, interreflexões, refrações e *subsurface scattering*. A função de visibilidade, responsável pela maior parte da computação de todo o pré-processamento realizado na função de transferência pode ser melhorado através da adição de estruturas de dados inteligentes para realizar o particionamento do espaço, de modo a não testar raios com polígonos desnecessáriamente. Em termos de melhorias nos algoritmos implementados no hardware gráfico, seria interessante obter uma maneira mais eficiente e flexível de transferir os coeficientes da função de iluminação e da função de transferência para a GPU, através, por exemplo, do uso de uma codificação em textura. Outra melhoria possível seria a substituição de *pixel buffers* e texturas no formato RGBA (que suporta diretamente o formato RGBE8 para imagens de HDR) para ponto flutuante, evitando o custo, por pixel, da codificação e decodificação de cores.

Anexo I

Radiometria e Fotometria

Radiometria é a ciência que estuda a medida de radiação, dentro do espectro eletromagnético. As freqüências do espectro envolvidas encontram-se na faixa de 3×10^{11}Hz à 3×10^{16}Hz (comprimentos de onda entre 0.01μm e 1000μm). Em termos prátcos, envolve luz infra-vermelha, luz visível e luz ultravioleta. Fotometria direciona seu estudo para a parte visível do espectro. Essa região visível é determinada pela sensitividade do olho humano. Os comprimentos de onda envolvidos na fotometria encontram-se entre 360nm e 830nm (1μm = 1000nm).

O objetivo desse anexo é mostrar como a luz pode ser medida e quais são as unidades envolvidas para representar essas medidas.

Luz é radiação eletromagnética. O espectro magnético varia de sinais com baixas frequências, como as ondas de rádio, até raios X e raios gama, de alta freqüência (*Figura I.1*). O sistema visual humano, entretanto, responde a apenas uma pequena fração desse espectro, a luz visível.

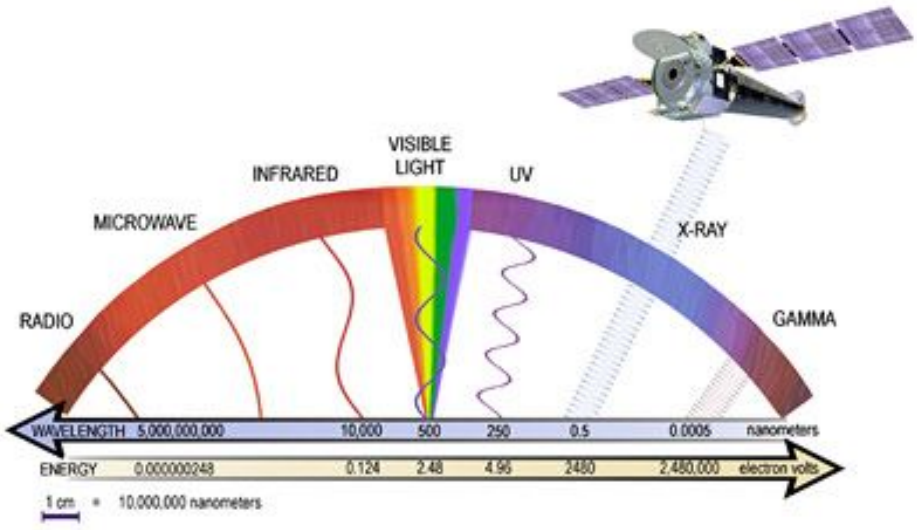


Figura I.1: Espectro eletromagnético. A fotometria direciona seus estudos apenas para uma pequena fatia deste, a luz visível.

I.1 Energia Radiante

A radiação eletromagnética transporta energia pelo espaço, e pode ser definida tanto como uma onda, quanto como uma partícula, e essa energia irá, possivelmente, interagir com a matéria. Quando essa interação ocorre, a energia incidente na matéria é convertida em outra forma de energia, o que ocorre, por exemplo, quando um aparelho eletrodoméstico de microondas é utilizado para aquecer um prato de sopa: a energia transportada pelas microondas emitidas atingem as moléculas da sopa, que, ao absorvem essa energia, transformam-na em energia térmica. **Energia radiante**, então, é a energia carregada pela radiação eletromagnética, e é denotada por *Q*, e é medida em joules (J).

I.2 Fluxo Radiante

A quantidade de energia, em uma intervalo de tempo, define o **fluxo radiante**, ou **potência radiante** (*Equação I.1* e *Figura I.2*), medida em joules por segundo (*J/s*), ou watt (*W*):

$$\Phi = \frac{dQ}{dt}$$

Equação I.1: Fluxo radiante (medido em watts – W).

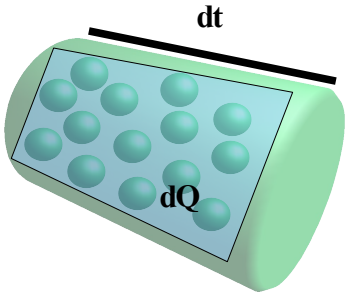


Figura I.2: Fluxo radiante: quantidade de energia por unidade de tempo. As esferas representam quantidades de energia radiante, que estão contidas em uma fatia do espaço, representada pelo cilindro.

I.3 Densidade de Fluxo Radiante

A **densidade de fluxo radiante** é fundamental para estudar modelos de iluminação. Ela define o fluxo radiante por unidade de área, em um ponto sobre uma superfície. Esse fluxo pode estar incidindo sobre o ponto (**irradiância**) ou pode estar deixando o ponto (**radiosidade**). Ele pode incidir/deixar o ponto em qualquer direção acima da superfície, conforme mostra a *Figura I.3*. O conjunto de todas as direções "visíveis" por uma superfície é chamado de **hemisfério de luz**.

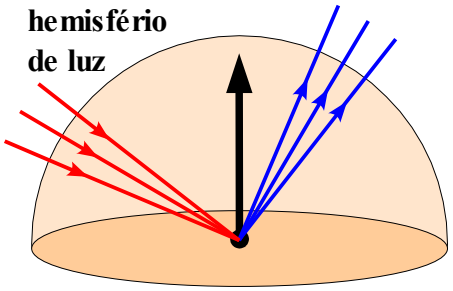


Figura I.3: Os raios vermelhos representam a irradiância, enquanto os azuis, a radiosidade. O vetor preto é o vetor normal à superfície.

A irradiância é denotada por (*E*), enquanto a radiosidade é denotada por (*B*). A densidade de fluxo radiante é medida em watts por metro quadrado (*W/m²*):

$$E = \frac{d\Phi}{dA} \qquad B = \frac{d\Phi}{dA}$$

Equação I.2: Densidade de fluxo radiante (irradiância – E; e radiosidade – B) (medida em watts por metro quadrado – W/m²).

Pode-se medir a densidade de fluxo radiante em qualquer lugar do espaço tri-dimensional, incluindo superfícies de objetos, materiais transparentes (vidro, água, gelo, ar) e vácuo.

I.4 Intensidade Radiante

Enquanto a densidade de fluxo mede a potência radiante sobre uma área, a **intensidade radiante** mede a potência radiante em uma direção, e é mais fácil de ser compreendida quando visualizada. Imagine um ponto em uma superfície e imagine, também, um raio incidindo (ou deixando) esse ponto. Agora transforme esse raio em um pequeno cone, onde o ponto na superfície representa seu ápice. O raio que une o ápice com o centro da base do cone coincide com o raio do fluxo (*Figura I.4)*:

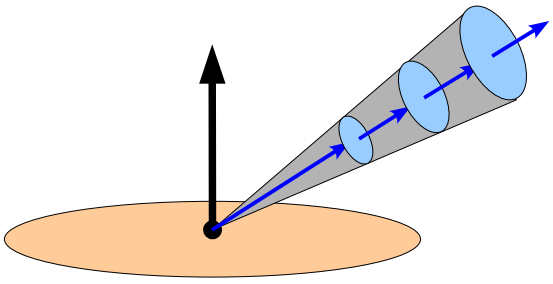


Figura I.4: Ponto em uma superfície emitindo energia: o ápice do cone é o próprio ponto sobre a superfície. O raio de energia em azul coincide com o raio que une o ápice com o ponto central da circunferência que é a base do cone.

Se imaginarmos, agora, o hemisfério de luz acima do ponto e da superfície, o cone irá intersecioná-lo, destacando uma área sobre a superfície do mesmo, como mostra a

Figura I.5:

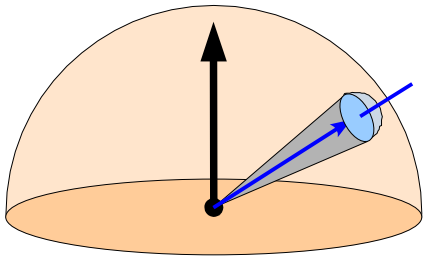


Figura I.5: A área em azul é a região na qual o cone interceptou a superfície do hemisfério de luz da superfície.

Matematicamente, essa área representa um **ângulo sólido**. Um ângulo sólido é a generalização, para três dimensões, de um ângulo ordinário em duas dimensões. Em 2D, um ângulo representa uma relação entre um trecho de circunferência (arco) e seu raio (*Figura I.6a*); um ângulo sólido representa uma relação entre uma região na superfície de uma esfera e o quadrado do raio da mesma (*Figura I.6b*). Um ângulo ordinário é medido em radianos (rad) e um ângulo sólido é medido em **esterradianos** (sr). O formato da área do ângulo sólido não implica em nada, o importante é tão somente o valor da área. Uma circunferência possui 2π radianos, enquanto uma esfera possui 4π esterradianos de área de superfície.

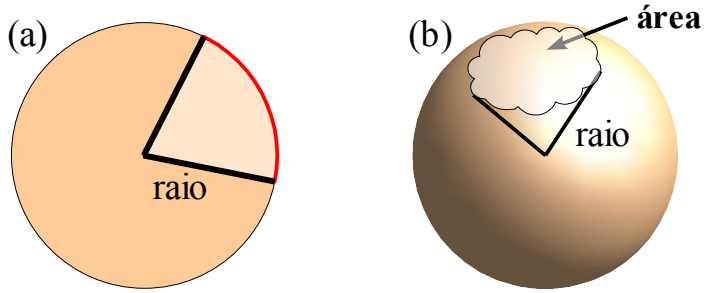


Figura I.6: (a) ângulo ordinário (2D); (b) ângulo sólido (3D) – o formato da área não é relevante, o que importa é o valor da área.

$$\hat{a}ngulo\,(\alpha) = \frac{arco}{raio}$$

Equação I.3: Ângulo ordinário, usualmente denotado por α, medido em radianos (rad).

$$\hat{a}ngulo\ s\acute{o}lido\,(\omega) = \frac{\acute{a}rea}{raio^2}$$

Equação I.4: Ângulo sólido, usualmente denotado por ω e medido em esterradianos (sr).

Assim, definimos intensidade radiante *I* como sendo o fluxo radiante presente nessa área (ângulo sólido), medida em watts por esterradiano (W/sr), conforme a equação abaixo e a *Figura I.7*:

$$I = \frac{d\Phi}{d\omega}$$

Equação I.5: Intensidade radiante (watts por esterradiano – W/sr).

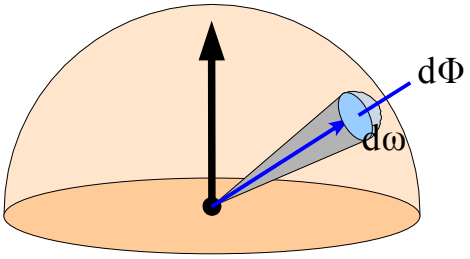


Figura I.7: Intensidade radiante é fluxo (dΦ) radiante sobre ângulo sólido (dω).

I.5 Radiância

O último conceito da radimetria a ser apresentado é a **radiância**. Ela representa uma a densidade do fluxo em uma superfície, em uma direção. Entretanto, essa densidade não é medida em relação a área da superfície, mas, sim com respeito a **área projetada** dessa superfície, na direção do fluxo. Esse conceito pode parecer confuso, e é mais fácil entendê-lo visualizando o que ele representa geometricamente.

Primeiramente, vamos pensar em apenas duas dimensões. Imagine uma superfície e seu vetor normal, imagine, ainda, um determinado fluxo saindo da superfície, em uma direção qualquer. Esses dois vetores acabam formando um ângulo (*Figura I.8*):

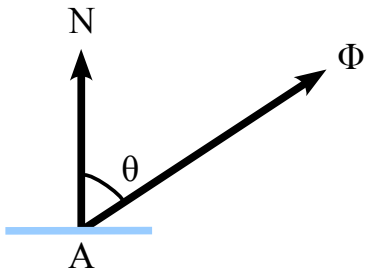


Figura I.8: O fluxo (Φ) incide na superfície (A), formando um ângulo (θ) entre o vetor normal à superfície (N) e a direção do fluxo (Φ).

Sabemos que a área dessa superfície está contida em um plano, e sabemos, também, que existem planos que são perpendiculares à direção do fluxo. Se projetarmos a área da superfície sobre um desses planos perpendiculares à direção do fluxo, encontraremos uma nova área, chamada de área projetada (*Figura I.9*):

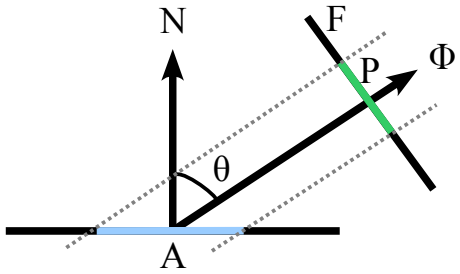


Figura I.9: A área em verde (P) representa a área da superfície (A) projetada em um plano perpendicular (F) à direção do fluxo (Φ).

Existe uma relação importante entre o ângulo e a área projetada: quanto maior o ângulo, menor a área projetada (*Figura I.10*):

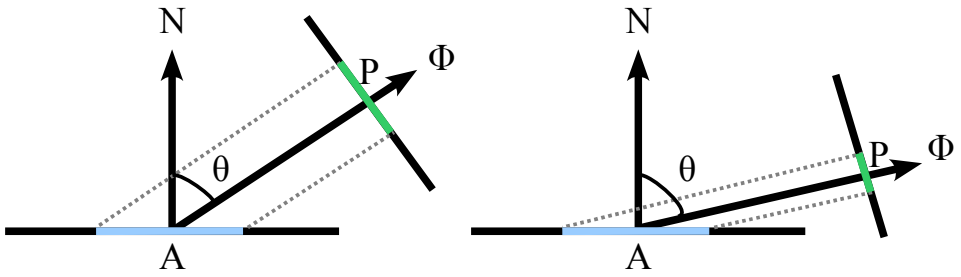


Figura I.10: Conforme aumentamos o ângulo (θ), a área projetada (P) diminui.

Essa relação pode ser expressa matematicamente através da equação:

$$dP = dA\cos(\theta)$$

Equação I.6: Relação entre a área da superfície e a área projetada, em uma direção.

A partir da *Equação 6*, temos que a densidade de fluxo radiante (*Equação I.2)*, para a área projetada, é dada por:

$$E = \frac{d\Phi}{dP} = \frac{d\Phi}{dA\cos(\theta)} \qquad B = \frac{d\Phi}{dP} = \frac{d\Phi}{dA\cos(\theta)}$$

Equação I.7: Densidade de fluxo radiante para a área projetada dP.

A radiância *L*, então, é definida como a intensidade radiante (*Equação I.5*) dividida pela área projetada (*Equação I.6*), e é medida em watts por metro quadrado por esterradiano (W/m².sr):

$$L = \frac{dI}{dP} = \frac{d\Phi}{d\omega\, dP} = \frac{d\Phi}{d\omega\, dA\cos(\theta)}$$

Equação I.8: Radiância (watts por metro quadrado por esterradiano – W/m².sr).

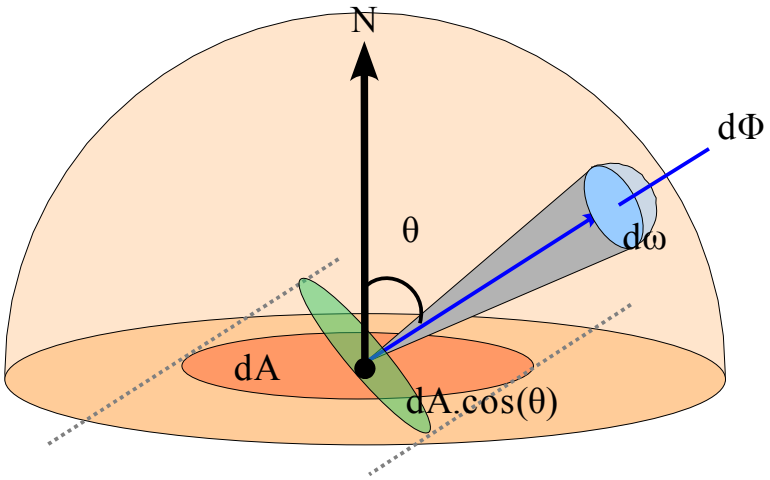


Figura 2.18: A radiância é a relação entre fluxo (dΦ) e a área projetada (dA.cos(θ)) da superfície (dA) pelo ângulo sólido (dω).

REFERÊNCIAS

[Annen04] Annen, T., Kautz, J., Durand F., and Seidel H. 2004. Spherical harmonic gradients for mid-range illumination. In EuroGraphics Symposium on Rendering, 2004.

[Blinn76] Blinn, J.F. Newell, M.E. 1976. Texture and reflection in computer generated images. Communications of the ACM 19, 542–546.

[Cohen-Wallace93] Cohen, M. F., Wallace, J., and Hanrahan, P. 1993. Radiosity and realistic image synthesis. Academic Press Professional, Inc, 1993, pp 32-33.

[Choi99] Choi, Choel Ho et al. 1999. "Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion.", J. Chem. Phys. Vol 111, No 19, 1999, pp 8825-8831.

[Debevec/FiatLux] Debevec, P. 1999. FiatLux. http://www.debevec.org/FiatLux/

[Debevec/HDRShop] Debevec, P. High Dynamic Range Image Processing and Manipulation. http://gl.ict.usc.edu/HDRShop/

[Debevec/HDRShopTutorial] Debevec, P. Creating a Light Probe. http://gl.ict.usc.edu/HDRShop/tutorial/tutorial5.html

[Debevec/ReflectionMap] Debevec, P. The Story of Reflection Mapping. http://www.debevec.org/ReflectionMapping/

[Debevec02] Debevec, P. 2002. Image-Based Lighting. Computer Graphics and Applications. March/April 2002. Pages 26-34.

[Debevec97] Debevec, P. E., and Malik, J. 1997. Recovering high dynamic range radiance maps from photographs. In SIGGRAPH 97 Conference Proceedings, AddisonWesley, T. Whitted, Ed., Annual Conference Series, ACM SIGGRAPH, 369–378.

[Debevec98] Debevec, P. 1998. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography, SIGGRAPH`98, 189-198.

[Fairchild98] FAIRCHILD, M. D. 1998. *Color appearance models*. Addison-Wesley, Reading, MA.

[Ferwerda96] James A. Ferwerda, Sumanta N. Pattanaik, Peter Shirley, and Don Greenberg. A Model of Visual Adaptation for Realistic Image Synthesis. SIGGRAPH 1996. Pages 249-258.

[Goral84] Goral, C. Torrance, K.E. Greenberg, D.P. Battaile, B. 1984. Modelling the interaction of light between diffuse surfaces. Computer Graphics (Proceedings of SIGGRAPH 1984) 18 (3), 213–222.

[Gouraud71] Gouraud, H. 1971. Computer display of curved surfaces. IEEE Transactions on Computers 20 (6), 623–629.

[Green03] Green, R. 2003. Spherical Harmonic Lighting: The Gritty Details. GDC 2003.

[Hammersley-Handscomb64] Hannerslay, J. M., Handscomb, D. C., 1964. Monte Carlo Methods.

[Hanrahan93] Hanrahan P., and Krueger W. 1993. "Reflection from Layered Surfaces due to Subsurface Scattering," Computer Graphics, Proc. of SIGGRAPH 93, ACM Press, New York, 1993, pp. 165-174.

[Hanrahan93] Hanrahan, P. Krueger, W. 1993. Reflection from layered surfaces due to subsurface scattering. Computer Graphics (Proceedings of SIGGRAPH 1993) 27 (), 165–174.

[ILM/OpenEXR] Industrial Light and Magic OpenEXR HDR Image Format. http://www.openexr.com/

[Kajiya86] Kajiya, J.T. 1986. The rendering equation. Computer Graphics (Proceedings of SIGGRAPH 1986) 20 (4), 143–150.

[Leffler2003] Leffler, Samuel, 2003. "LibTIFF – TIFF Library and Utilities," http://remotesensing.org/libtiff

[London-Upton98] London, B., and Upton, J. 1998. *Photography*, sixth ed. Longman.

[Luque05] Luque, R. 2005. Árvores BSP Semi-Ajustáveis. Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação, Instituto de Informática, UFRGS (Julho 2005).

[Mardaljevic99] Mardaljevic, J. 1999. Daylight Simulation: Validation, Sky Models and Daylight Coefficients. A thesis submitted in partial fulfilment of the requirements of the De Montfort University for the degree of Doctor of Philosophy.

[Miller-Hoffman84] G. Miller and C. Hoffman. Illumination and reflection maps: Simulated objectsin simulated and real environments. *SIGGRAPH 84 Advanced Computer Graphics Animation seminar notes*, 1984.

[Möller-Trumbore97] Möller, T., and Trumbore, B. 1997. "Fast, minimum storage ray-triangle intersection". Journal of graphics tools, 2(1):21-28, 1997.

[Phong73] Phong, B.T., "Ilumination for Computer Generated Surfaces", Doctoral Thesis, University of Utah, 1973.

[Phong75] Phong, B-T. 1975. Illumination for computer generated pictures. Communications of the ACM 18 (6), 311–316.

[Ramamoorthi-Hanrahan01] Ramamoorthi, R. and Hanrahan, P. 2001. An efficient representation for irradiance environment maps, SIGGRAPH'01, 497-500

[Ramamoorthi03] Ramamoorthi, R., Ng, R., and Hanrahan, P. 2003. All-Frequency Shadows Using Non-linear Wavelet Lighting Approximation. SIGGRAPH`03.

[Reinhard02] Reinhard, E., Stark, M., Shirley, P., and Ferwerda, J. 2002. Photographic tone reproduction for digital images. ACM Transactions on Graphics 21, 3, 267-276.

[Schlick94] Schlick, C. 1994. Quantization techniques for the visualization of high dynamic range pictures. In *Photorealistic Rendering Techniques*, Springer-Verlag Berlin Heidelberg New York, P. Shirley, G. Sakas, and S. M¨uller, Eds., 7–20.

[Seetzen04] Seetzen, Helge, Wolfgang Heidrich, Wolfgang Stuerzlinger, Greg Ward, et. Al. High Dynamic Range Display Systems. 2004 ACM Transactions on Graphics, Volume 23 Number 3. SIGGRAPH 2004. Pages 760-768.

[Sloan-Kautz-Snyder02] Sloan, P., Kautz, J., and Ssyder, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. ACM Transactions on Graphics 21, 3, 527–536.

[Sloan05] Sloan, P.P., Luna, B., and Snyder J. 2005. Local, Deformable Precomputed Radiance Transfer. 2005. To appear in the Proceedings of SIGGRAPH 2005, July, 2005.

[Stockham72] Stockham, T. 1972. Image processing in the context of a visual model. *Proceedings of the IEEE 60*, 7, 828–842.

[Tumblin-Rushmeier91] Tumblin, J., and Rushmeier, H. 1991. Tone reproduction for realistic computer generated images. Tech. Rep. GIT-GVU-91-13, Graphics, Visualization, and Useability Center, Georgia Institute of Technology.

[Ward/HDREncodings] Ward, G. High dynamic range encodings.
http://www.anyhere.com/gward/hdrenc/hdr_encodings.html

[Ward/Radiance] Ward, G. Radiance Synthetic Imaging System.
http://radsite.lbl.gov/radiance/

[Ward91] Ward, G. "Real Pixels" Graphics Gems II, edited by James Arvo, Academic Press, 1991.

[Ward97] Ward, G., Rushmeier, H., and Piakto, C. 1997. A visibility matching tone reproduction operator for high dynamic range scenes. IEEE Transactions on Visualization and Computer Graphics 3, 4 (December).

[White84] White, M., Zakia, R., and Lorenz, P. 1984. The new zone system manual. Morgan & Morgan, Inc.

[Whitted80] Whitted, T. 1980. An improved illumination model for shaded display. Communications of the ACM 23 (6), 343–349.