Pontifícia Universidade Católica
do Rio de Janeiro

Bianca Faria Dutra Fragoso

Implementação de um Renderizador com Traçado de Caminhos

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio.

Orientador: Prof. Waldemar Celes Filho

Rio de Janeiro
Dezembro de 2022

Pontifícia Universidade Católica
do Rio de Janeiro

Bianca Faria Dutra Fragoso

Implementação de um Renderizador com Traçado de Caminhos

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós–graduação em Informática da PUC-Rio . Aprovada pela Comissão Examinadora abaixo:

Prof. Waldemar Celes Filho
Orientador
Departamento de Informática – PUC-Rio

Prof. Marcelo Walter
Universidade Federal do Rio Grande do Sul - UFRGS

Prof. Alberto Barbosa Raposo
Departamento de Informática - PUC-Rio

Rio de Janeiro, 14 de Dezembro de 2022

Bianca Faria Dutra Fragoso

Graduado em engenharia da computação pela Pontificia Universidade Católica do Rio de Janeiro.

Ficha Catalográfica

Fragoso, Bianca

Implementação de um Renderizador com Traçado de Caminhos / Bianca Faria Dutra Fragoso; orientador: Waldemar Celes Filho. – 2022.

84 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2022.

Inclui bibliografia

1. Informática – Teses. 2. Traçado de caminhos. 3. BRDF. 4. Transmitância. 5. Dispersão. 6. Ricochetear. 7. Amostragem de Importância. 8. Superfície. 9. Volume. 10. Função de Fase. 11. Fotorrealismo. 12. Monte Carlo. 13. Rastreamento. 14. Renderização Cinematográfica.
I. Celes, Waldemar. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Dedico esta dissertação ao meu professor preferido, meu querido pai, Marcelo.

Agradecimentos

Primeiramente gostaria de agradecer a toda a minha família que foi fundamental para que eu pudesse avançar e concluir esta dissertação. Minha mãe querida que sempre esteve comigo nesta jornada e me deu forças nos momentos mais difíceis. Meu pai que é meu orgulho, minha maior inspiração, meu imortal 2 vezes. Seu exemplo e ensinamentos foram essenciais para que eu pudesse concluir este trabalho. Minhas 3 irmãs, Andrea, Paula e Patricia que me ouviam e me incentivavam sempre.

Gostaria de agradecer também ao meu orientador Waldemar que sempre foi muito presente durante todo esse período. Nossas discussões e trocas de ideias foram muito importantes para que eu conseguisse avançar na pesquisa.

Queria agradecer também a pessoas que me ajudaram em algum momento desta pesquisa. Ao Quatrin, que me tirou várias dúvidas, sempre foi muito solícito e me forneceu volumes com funções de transferência, o que foi muito importante para este trabalho. Também ao Rustam e André que me ajudaram em momentos da minha implementação.

Agradeço também à minha equipe de trabalho do Tecgraf, o J5 Team, Suellen, Fábio, Ricardo e Anderson. Todos foram sempre muito compreensivos comigo e o ótimo ambiente de trabalho com certeza foi muito importante para que eu tivesse tranquilidade e conseguisse terminar esta dissertação. Em especial, nossa líder Suellen, que sempre me apoiou e ajudou durante esse processo.

Também gostaria de agradecer a todos os meus amigos que me apoiaram neste período. Daniel, Victor, Thiana, Helena, Rapha, Bruno e muitos outros que me ajudaram e trouxeram leveza para os meus dias mais estressantes.

Finalmente, gostaria de agradecer ao CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico, pela bolsa de estudos e auxílio financeiro.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Resumo

Fragoso, Bianca; Celes, Waldemar. **Implementação de um Renderizador com Traçado de Caminhos**. Rio de Janeiro, 2022. 84p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Graças aos avanços significativos nas técnicas de renderização e ao aumento do poder computacional, o algoritmo de traçado de caminhos, também chamado de traçado de raios com Monte Carlo, tornou-se o método de renderização predominante usado em campos que priorizam o realismo, como na produção de filmes e na visualização fotorrealista de exames médicos. Esse algoritmo simula precisamente o comportamento da luz possibilitando a simulação de alguns efeitos de iluminação, como sombras suaves, reflexos brilhantes e iluminação indireta. Esta dissertação trata do problema de renderização fotorrealista de cenas envolvendo tanto superfícies como volumes. Ela tem como contribuição a apresentação e explicação dos principais métodos que podem ser utilizados no desenvolvimento do algoritmo de traçado de caminhos, além de ter como produto final a implementação desse algoritmo resultando em um framework que renderiza cenas híbridas que podem envolver superfícies e volumes.

Palavras-chave

Traçado de caminhos; BRDF; Transmitância; Dispersão; Ricochetear; Amostragem de Importância; Superfície; Volume; Função de Fase; Fotorrealismo; Monte Carlo; Rastreamento; Renderização Cinematográfica;.

Abstract

Fragoso, Bianca; Celes, Waldemar (Advisor). **Implementation of a Path Tracing Render**. Rio de Janeiro, 2022. 84p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Thanks to significant advances in rendering techniques and increased computational power, the path tracing algorithm, also called Monte Carlo ray tracing, has become the predominant rendering method used in fields that prioritize realism, such as in film production and photorealistic visualization of medical exames. This algorithm accurately simulates the behavior of light, making it possible to simulate some lighting effects, such as soft shadows, glossy reflections and indirect lighting. This dissertation deals with the problem of photorealistic rendering of scenes involving both surfaces and volumes. It has as contribution the presentation and explanation of the main methods that can be used in the development of the path tracing algorithm, besides having as a final product the implementation of this algorithm resulting from a framework that renders hybrid scenes that may involve surfaces and volumes.

Keywords

Path Tracing; BRDF; Transmittance; Scattering; Bounce; Importance Sampling; Surface; Volume; Phase Function; Photorealism; Monte Carlo; Tracking; Cinematic Rendering; .

Sumário

Lista de figuras

Lista de tabelas

Lista de algoritmos

Lista de Abreviaturas e Símbolos

BSDF – Bidirectional Scattering Distribution Function

BRDF – Bidirectional Reflectance Distribution Function

pdf – Probability Density Function

HDR – High Dynamic Range

CDF – Cumulative Distribution Function

RTE – Radiative Transfer Equation

VRE – Volume Rendering Equation

MIS – Multiple Importance Sampling

CPU – Central Processing Unit

NEE – Next Event Estimation

Tabela 1: Nomenclatura

Símbolo	Descrição
x	Posição
t	Parâmetro do Raio
ω	Direção do Raio
x_t	Posição Parametrizada ao Longo do Raio: $x_t = x + t\omega$
$\sigma_a(x)$	Coeficiente de Absorção
$\sigma_s(x)$	Coeficiente de Dispersão
$\sigma_t(x)$	Coeficiente de Extinção $= \sigma_a(x) + \sigma_s(x)$
$\alpha(x)$	Albedo de Dispersão Única $= \sigma_s(x)/\sigma_t(x)$
$f_r(x, \omega, \omega')$	BRDF
$f_p(x, \omega, \omega')$	Função de fase
d	Comprimento do raio/Domínio da integração do volume: $0 < t < d$
ξ, ζ	Números aleatórios
$L(x, \omega)$	Radiância em x na direção ω
$L_e(x, \omega)$	Radiância emitida
$L_s(x, \omega)$	Radiância de Dispersão de Entrada em x vinda da direção ω

Se uma pessoa fizesse só o que entende,
jamais avançaria um passo.

Clarice Lispector.

1
Introdução

Renderização é o processo de síntese de uma imagem a partir da descrição de uma cena. Obviamente, esta é uma tarefa muito ampla, e há muitas maneiras de abordá-la. Técnicas de renderização baseadas em física (physically based rendering) tentam simular a realidade; isto é, elas usam princípios da física para modelar a interação da luz e da matéria [2] . Embora uma abordagem baseada em física possa parecer a maneira mais óbvia de abordar a renderização, ela só foi amplamente adotada na prática há pouco mais de 20 anos. Um dos principais algoritmos de renderização baseada em física é o traçado de caminhos que será explorado com detalhes neste trabalho.

O algoritmo de traçado de caminhos foi introduzido em 1986 com o artigo de Kajiya [6] que também introduziu a equação de renderização. Desde então, esse algoritmo vem ganhando espaço em áreas que prezam pelo realismo de imagens. A indústria cinematográfica é uma das que mais tem aproveitado esse algoritmo. A Walt Disney Animation Studios utiliza para renderizar seus filmes, desde 2014, o software Hyperion [7], que tem como algoritmo principal o traçado de caminhos. O filme Big Hero 6 (2014) foi completamente renderizado utilizando esse algoritmo, assim como os filmes Zootopia (2016) e Moana (2016).

A Pixar também tem seu próprio software de renderização, RenderMan [8], que anteriormente utilizava o algoritmo de Reyes [9], mas que depois do sucesso de Big Hero 6, foi modificado para ser um renderizador baseado em física e utilizar o algoritmo de traçado de caminhos [10]. Filmes como Procurando Dory (2016), Carros 3 (2017) e Viva–A Vida É uma Festa (2017) foram renderizados utilizando essa técnica.

A área médica é outra que utiliza amplamente o algoritmo de traçado de caminhos para renderização de exames. A chamada renderização cinematográfica (cinematic rendering) foi introduzida por Engel [5] e tem como foco a renderização fotorrealista de dados de exames médicos como ressonâncias magnéticas e tomografias computadorizadas.

No entanto, a tarefa de simular o mundo real já não é tão simples mesmo quando tratamos apenas de superfícies. Para se construir uma cena que possa ser renderizada de forma fotorrealista, uma série de técnicas devem ser implementadas para que o comportamento da luz seja devidamente simulado. Além disso, a cena deve ser organizada e construída de maneira adequada. Cada objeto da cena deve ter um material e fontes de luz devem ser posicionadas de

forma que iluminem a cena do jeito desejado.

Quando começamos a pensar em inserir volumes às nossas cenas, a complexidade aumenta, pois, segundo Fong et al [10], a renderização de volumes ainda oferece seu próprio conjunto de desafios únicos que, a primeira vista, podem ser assustadores para desenvolvedores e pesquisadores de traçado de caminhos. Um desses desafios é a escolha da técnica de integração para resolver a integral volumétrica, um problema que é caro e complexo.

Em 2009, Banks e Beason [11] relataram que a inserção da iluminação baseada em física na visualização científica foi quase zero no mercado em 2008, apesar de muitas pesquisas indicarem que ela trazia vantagens perceptuais. Uma das razões pelas quais isso ocorria, segundo eles, era porque os cientistas que geravam as visualizações não tinham acesso fácil a implementações de iluminação global em seu fluxo de trabalho padrão. Desde essa época, o número de trabalhos e cursos sobre o assunto aumentaram e o uso de traçado de caminhos para renderização de volumes vem se popularizando, com cada vez mais trabalhos sendo publicados nessa área. No entanto, por envolver muitos conceitos, ainda é raro encontrar um trabalho que explique detalhadamente sobre todos os conceitos envolvidos nesse tipo de renderização e que fale diretamente sobre as questões práticas de implementação que também não são triviais.

Na literatura, é comum vermos trabalhos focados apenas em renderização de superfícies ou apenas renderização de volumes. Quando os dois são tratados juntamente, o foco, geralmente, é em cenas que possuem superfícies e volumes que simulam efeitos naturais (participating media), como nuvens, explosões, fog etc. Raramente vemos a união de cenas que contém superfícies e dados volumétricos de exames, por exemplo.

Neste trabalho, procuramos fazer uma síntese coerente de todos os conceitos e ferramentas necessárias para a implementação tanto de traçado de caminhos para superfícies quanto para volumes. Descrevemos os métodos mais importantes de cada etapa da implementação de um framework desenvolvido para renderizar cenas que contém esses dois tipos de dados.

Esta dissertação está divida da seguinte forma: No segundo capítulo, alguns trabalhos anteriores sobre renderização de superfícies e volumes são citados. No terceiro capítulo são apresentados conceitos importantes, ferramentas e técnicas para compreender melhor o algoritmo de traçado de caminhos. No quarto capítulo apresentamos o algoritmo de traçado de caminhos para superfícies e é feita uma comparação entre o algoritmo de traçado de caminhos e outros algoritmos de renderização. No quinto capítulo são introduzidos alguns conceitos e técnicas de traçado de caminhos para volumes. No sexto capítulo

apresentamos o método proposto para implementar o framework, falando sobre as técnicas escolhidas, dando uma visão mais prática sobre a implementação e deixando claro as dificuldades de se juntar superfícies e volumes. No sétimo capítulo alguns resultados da renderização são apresentados. Finalmente no oitavo capítulo é feita uma conclusão com algumas ideias para trabalhos futuros.

2
Trabalhos relacionados

Na literatura, trabalhos foram publicados focando em algum dos temas que serão tratados nesta dissertação. A seguir citamos alguns trabalhos relacionados que abordam os seguintes temas: (1) renderização baseada em física focada em superfícies; (2) métodos de renderização de volumes; (3) traçado de caminhos para volumes; (4) traçado de caminhos envolvendo superfícies e volumes.

2.1
Renderização baseada em física de superfícies

Abordagens baseadas na física em renderização começaram a ser seriamente consideradas por pesquisadores na década de 1980. O artigo de Whitted [12] introduziu a ideia de usar traçado de raios para efeitos de iluminação global, abrindo portas para simular com precisão a distribuição de luz nas cenas. As imagens renderizadas produzidas por sua abordagem eram diferentes de todas as vistas antes, o que gerou entusiasmo por essa abordagem [2].

Outro avanço notável na renderização baseada em física foi o modelo de reflexão de Cook e Torrance [13], que introduziu modelos de reflexão de microfacetas. Entre outras contribuições, eles mostraram que modelar com precisão a reflexão de microfacetas tornou possível renderizar superfícies metálicas com precisão; o metal não havia sido bem renderizado em abordagens anteriores.

Pouco tempo depois, Goral et al [14] fizeram conexões entre a literatura de transferência térmica e a renderização, mostrando como incorporar efeitos globais de iluminação difusa usando uma aproximação com base física do transporte de luz. Este método foi baseado em métodos de elementos finitos, onde áreas de superfícies na cena trocam energia umas com as outras. Essa abordagem passou a ser chamada de "radiosidade", em homenagem a uma unidade física relacionada.

Embora a abordagem de radiosidade fosse fortemente baseada em unidades físicas e conservação de energia, com o tempo ficou claro que não levava a algoritmos de renderização viáveis: a complexidade computacional assintótica era difícil de gerenciar; os pesquisadores tiveram dificuldade em desenvolver algoritmos robustos e eficientes para esse fim e a adoção da radiosidade na prática foi limitada.

Durante os anos da radiosidade, um pequeno grupo de pesquisadores buscou abordagens físicas para renderização baseada em traçado de raios e integração de Monte Carlo. Na época, muitos olhavam para seu trabalho com ceticismo; o ruído nas imagens devido à variação da integração de Monte Carlo parecia inevitável, enquanto os métodos baseados em radiosidade rapidamente davam resultados visualmente agradáveis, pelo menos em cenas relativamente simples.

Em 1984, Cook et al [15] introduziram o traçado de raios distribuído, que generalizou o algoritmo de Whitted para calcular o desfoque de movimento e o desfoque de câmeras, reflexos embaçados de superfícies brilhantes e iluminação de fontes de luz de área, mostrando que o traçado de raios era capaz de gerar uma série de efeitos de iluminação importantes.

Pouco tempo depois, em 1986, Kajiya [6] introduziu o traçado de caminhos. Ele estabeleceu uma formulação para o problema de renderização (a equação de renderização) e mostrou como aplicar a integração de Monte Carlo para resolvê-lo. Esse trabalho exigia uma quantidade imensa de computação: para renderizar uma imagem 256x256 de duas esferas com traçado de caminhos, eram necessárias 7 horas de computação em um computador IBM 4341. Nos anos subsequentes, trabalhos importantes com Monte Carlo para síntese de imagens realistas foram descritos em artigo de Arvo e Kirk [16] e na tese de Shirley [17]. Essas contribuições foram importantes para a pesquisa em Monte Carlo.

Um outro trabalho importante de renderização baseada em física foi o trabalho de Veach, descrito em detalhes em sua dissertação [18]. Ele avançou nos principais fundamentos teóricos da renderização de Monte Carlo, ao mesmo tempo em que desenvolveu novos algoritmos, como a múltipla amostragem por importância e traçado de caminhos bidirecional, que melhoraram muito sua eficiência.

2.2 Métodos de Renderização de Volumes

Max [19] apresentou diferentes modelos usados para calcular o transporte de luz. Esses modelos levam em conta ou não eventos que podem ocorrer quando um raio de luz atinge uma partícula do volume. Eles podem ser classificados em seis grupos em ordem crescente de realismo: somente absorção, somente emissão, emissão e absorção combinada, dispersão única da iluminação externa sem sombras, dispersão única com sombras e dispersão múltipla.

Jönsson et al [1] discutiram várias técnicas de iluminação para renderização de volumes. Embora a maioria das técnicas seja baseadas na formulação

de Max [19] , os resultados visuais podem mudar drasticamente. A diferença visual mais proeminente está na intensidade das sombras, bem como na sua frequência, que é visível com base no desfoque das bordas da sombra.

As técnicas de renderização podem ser divididas em seis grupos principais de acordo com Jönsson: Iluminação baseada em regiões locais, iluminação baseada em fatias, iluminação baseada no espaço da luz, iluminação baseada em grades, iluminação baseada em funções e iluminação baseada em traçado de raios.

Iluminação baseada em regiões locais: Como o nome já diz, para o cálculo da iluminação, os modelos nesse grupo consideram apenas o voxel atual, voxels adjacentes ou ainda regiões maiores, mas que ainda são consideradas regiões localizadas. Esta é a abordagem mais simples para adicionar iluminação em renderização de volume, pois aplica o modelo de iluminação de Blinn-Phong, composto por iluminação ambiente, difusa e termos especulares. Uma técnica representativa desse tipo de iluminação é a baseada em gradiente [20], que usa o gradiente do campo escalar como vetor normal para fazer o cálculo de iluminação.

Iluminação baseada em fatias: Essa técnica usa uma série de fatias retangulares como geometria para retratar dados volumétricos. No entanto, hoje em dia, com maior memória e suporte a programação em GPU, é mais comum simplesmente mapear os dados volumétricos para uma textura 3D. Uma técnica representativa desse tipo de iluminação é a de oclusão direcional [21], que não leva em consideração o termo de emissão e suporta apenas uma fonte de luz fixa na posição da câmera.

Iluminação baseada no espaço da luz: Essa técnica considera que a iluminação é propagada direcionalmente levando em consideração a posição corrente da fonte de luz; no entanto, é limitada, pois suporta apenas uma fonte de luz. Uma das técnicas representativas desse tipo de iluminação é a de volume de varredura do plano da imagem [22], que suporta uma fonte de luz pontual ou direcional e dipersão única e múltipla.

Iluminação baseada em grades: Essa técnica usa diretamente a grade para calcular a iluminação por voxel. Uma técnica representativa desse tipo de iluminação é a de propagação de sombra do volume [23], que distribui a iluminação em uma fase de pré-processamento, armazenando seu cálculo em uma grade de dados volumétricos.

Iluminação baseada em funções: Essa técnica usa funções como base para representar a radiância da fonte de luz e a transparência. Uma das técnicas representativas desse tipo de iluminação é a dos harmônicos esféricos [24], que usa uma estrutura semelhante a um mipmap para armazenar várias resoluções

do conjunto de dados volumétricos, exigindo muito armazenamento. Nessa técnica, fontes de luz pontuais e de área são suportadas.

Todos os cinco grupos mencionados estão presentes na Figura 2.1, representados por uma de suas técnicas. Para um mesmo volume é possível notar que há uma grande diferença principalmente quanto à qualidade da sombra.

Figura 2.1: Grupos principais de técnicas de renderização de volumes. Da esquerda para direita as técnicas utilizadas são: Iluminação baseada em gradiente; iluminação com oclusão direcional; iluminação de volume de varredura do plano da imagem; iluminação de propagação de sombra do volume; iluminação dos harmônicos esféricos. Imagem tirada de [1].

2.3 Traçado de raios com Monte Carlo para volumes

O último grupo de técnicas de renderização, como já mencionado, é o baseado em traçado de raios. Esta dissertação é baseada, principalmente, nos fundamentos deste grupo.

Assim como o traçado de raios tem sido amplamente utilizado na renderização de superfícies para simular imagens fisicamente corretas e, portanto, realistas, alguns autores também investigaram essa direção na área de renderização de volumes.

Uma dessas abordagens foi proposta por Stegmaier et al [25]. Com essa abordagem, foi possível simular reflexões e refração semelhantes a espelhos, lançando um raio e calculando sua deflexão. Abordagens de refração mais sofisticadas foram apresentadas por outros [26], [27].

Métodos de traçado de raios baseado em Monte Carlo também foram aplicados na área de renderização interativa de volume para produzir imagens realistas. A vantagem dessas abordagens é que elas são acoplados diretamente com o transporte de luz baseado em física.

A primeira abordagem baseada em Monte Carlo foi apresentada por Rezk-Salama [28]. Essa abordagem suporta dispersão única e múltipla, em que

a interatividade é alcançada restringindo os cálculos de dispersão caros apenas a algumas superfícies. No entanto, devido a essa restrição, não era possível renderizar materiais translúcidos, como nuvens.

Mais recentemente, Kroes et al [29]. apresentaram um método de traçado de raios baseado em Monte Carlo, que não limita a dispersão a superfícies. Sua abordagem simula vários fatores do mundo real que influenciam a iluminação volumétrica, como fontes de luz com múltiplos formatos, fontes de luz vindas de texturas, câmeras do mundo real com lentes e abertura, bem como propriedades complexas de materiais.

Um método que foi desenvolvido pela Siemens Healthineers, e que é importante ser citado, pois tem trazido uma série de contribuições em renderização fotorrealista de volumes, é a Renderização Cinematográfica (cinematic rendering) (ver [30] e [5]). Esse método de renderização de volumes calcula em tempo real renderizações 3D fotorrealistas de dados médicos, como aquisições de ressonância magnética e tomografia computadorizada e tem sido uma importante área de pesquisa que utiliza traçado de caminhos.

2.4 Traçado de caminhos envolvendo superfícies e volumes

Uma série de trabalhos trazendo renderizações que incluem superfícies e volumes também tem sido publicados ao longo dos anos. No entanto, a maioria desses trabalhos foca principalmente em dados volumétricos que preenchem um meio e simulam efeitos naturais (participating media), como nuvens, atmosfera, fumaça, fog, explosões etc.

Fong et al [10], em um curso sobre renderização em produção (production rendering), mostram como inserir a teoria de volumes em um framework que já tem o traçado de caminhos para superfícies implementado. Nesse trabalho são discutidas as principais técnicas utilizadas para o traçado de caminhos em volumes e é mostrado como a integração do volume pode ser incorporada à integração da luz.

Christensen et al [8] descrevem a versão moderna do RenderMan, um dos maiores softwares de traçado de caminhos desenvolvido e usado pela Pixar. Eles apresentam uma nova arquitetura para um software de traçado de caminhos que é extensível e programável com muitos recursos que são essenciais para lidar com as cenas extremamente complexas na produção de filmes.

Clarberg et al [31] da NVIDIA mostram resultados visuais e discutem como construiram um software de traçado de caminhos em tempo real. Além disso, eles discutem os problemas de pesquisa ainda em aberto nessa área, focando principalmente no que será necessário para alcançar o sonho de

renderização fotorrealista em tempo real de conteúdo verdadeiramente fiel à realidade. Nesse trabalho volumes são brevemente citados.

3
Técnicas e ferramentas

Tanto a renderização de superfícies quanto a renderização de volumes são regidas por equações integrais. No entanto, essas equações integrais geralmente não têm soluções analíticas, então é necessário recorrer aos métodos numéricos. Neste capítulo, esses métodos numéricos e ferramentas matemáticas utilizadas, assim como alguns conceitos importantes, serão sucintamente apresentados.

3.1
Conceitos básicos de probabilidade e notações

Para entender melhor os métodos utilizados para a resolução das equações integrais, é importante que se tenha em mente alguns conceitos básicos de probabilidade. Isso porque o método de Monte Carlo, que é o principal método numérico utilizado, é um método probabilístico.

Considere um espaço de probabilidade $(\Omega, \mathcal{F}, \mathbb{P})$, onde Ω denota o espaço amostral, $\mathcal{F}$ o conjunto de todos os eventos possíveis e $\mathbb{P} : \mathcal{F} \to [0, 1]$ a medida de probabilidade. O conjunto $\mathcal{F}$, é uma coleção de subconjuntos de Ω, incluindo o conjunto vazio. Ele é fechado para operações contáveis de união, interseção e complemento de conjuntos.

Um dos primeiros conceitos importantes é o de **variável aleatória**. Grosso modo, uma variável aleatória é uma função que mapeia os resultados de um processo aleatório (estocástico) em valores numéricos. Geralmente elas são representadas por letras maiúsculas (X) ou letras gregas (ξ). Por exemplo, quando jogamos um dado, o resultado desse lançamento é uma variável aleatória, X_i, amostrada de um espaço de eventos simples $\Omega = \{1, 2, 3, 4, 5, 6\}$ (espaço amostral). Se o dado não é viciado, cada evento desses tem probabilidade $\frac{1}{6}$ e a soma de todas as probabilidades é necessariamente igual a 1. De maneira formal, uma variável aleatória é uma função $X : \Omega \to \mathbb{R}$ tal que para todo $a \in \mathbb{R}$ ao evento $\{X \leq a\} = \{w \in \Omega : X(w) \leq a\}$ pode-se atribuir uma probabilidade, i.e., $\{X \leq a\} \in \mathcal{F}$ (ver [32]).

Um outro conceito importante é o de **função de probabilidade cumulativa (CDF)**. A CDF $P(x)$ de uma variável aleatória X é a probabilidade de um valor escolhido ser menor ou igual a um valor x, i.e., $P(x) = \mathbb{P}(X \leq x)$. Por exemplo, no caso do lançamento de dados, temos que $P(2) = \mathbb{P}(X \leq 2) = \frac{1}{3}$, pois a chance de amostrarmos um número menor ou igual a 2, nesse caso, é a soma das probabilidades de sair o número 1 ou o número 2. Vale salientar que

o conjunto $\{X \leq 2\}$ é um evento em $\mathcal{F}$, como mencionado anteriormente (ver em [32] e [33]).

No entanto, temos que pensar que, em renderização, estamos lidando com variáveis aleatórias contínuas e não com variáveis aleatórias discretas como no caso de lançamento de dados. Quando vamos amostrar uma direção no hemisfério por exemplo, o espaço amostral que estamos lidando tem um conjunto não-enumerável de possibilidades de eventos que podem ser amostrados.

Considere, a título de ilustração, um caso simples de amostragem de um número entre 0 e 2, i.e. um número no intervalo $[0, 2]$. A probabilidade de amostrar um valor x específico nesse intervalo é zero, pois existe um infinito não-enumerável de valores entre 0 e 2, já que estamos no domínio de números reais. No entanto, podemos definir qual a probabilidade de amostrar um número x em um dado intervalo $[a, b]$. Para esse propósito, usamos a **função de densidade de probabilidade (pdf)**, que é um conceito muito importante para os métodos que serão utilizados, e será muito mencionado neste trabalho.

A pdf é uma função, $p(x)$, que define como a probabilidade de uma variável aleatória se comporta em um determinado intervalo (define a densidade da probabilidade no intervalo). Considere, por exemplo, o intervalo $\Delta(x) = [x - h, x + h]$, uma vizinhança de x. Grosso modo, se considerarmos que $p(x)$ é contínua, podemos obter uma aproximação da probabilidade em uma vizinhança de x da seguinte maneira:

$$\mathbb{P}(X \in \Delta(x)) \approx p(x)\Delta(x) \tag{3-1}$$

Para determinarmos a probabilidade de uma variável aleatória em um intervalo $[a, b]$, somamos então a probabilidade para pequenos intervalos, e na condição limite temos que a probabilidade no intervalo $[a, b]$ é igual à integral de $p(x)$ nesse intervalo, como mostrado na Equação (3-2). É importante salientar que, para uma função $p(x)$ ser uma pdf é necessário que satisfaça as seguintes condições: (1) $p(x) \geq 0$; (2) $\int_{\mathbb{R}} p(x)d(x) = 1$.

$$P(a \leq X \leq b) = \int_a^b p(x)d(x) \tag{3-2}$$

Podemos entender melhor a pdf tentando imaginar como poderíamos usá-la na prática. Considere o espaço amostral $\Omega = [0, 2]$, i.e., um espaço amostral não-enumerável. Como a integral da pdf em todo espaço amostral deve ser igual a 1, então a área entre 0 e 2 deve ser igual a 1. Se considerarmos que a probabilidade associada a qualquer subconjunto de Ω de mesmo tamanho (mesmo comprimento) são iguais, i. e., a área associada a qualquer subconjuntos de Ω de mesmo comprimento são iguais, então para as condições acima

serem satisfeitas, $p(x) = \frac{1}{2}, \forall x \in \Omega$, como no gráfico de $p(x)$ (Figura 3.1). Essa pdf é normalmente denominada na literatura de **distribuição uniforme**. Observe que essa situação é o equivalente para o caso contínuo ao que acontece no caso discreto quando consideramos eventos equiprováveis (o caso do lançamento de dados descrito acima). A diferença no caso contínuo é que ao invés da probabilidade para cada elemento ser a mesma, como no caso discreto, a probabilidade aqui é mesma para intervalos de mesmo tamanho.

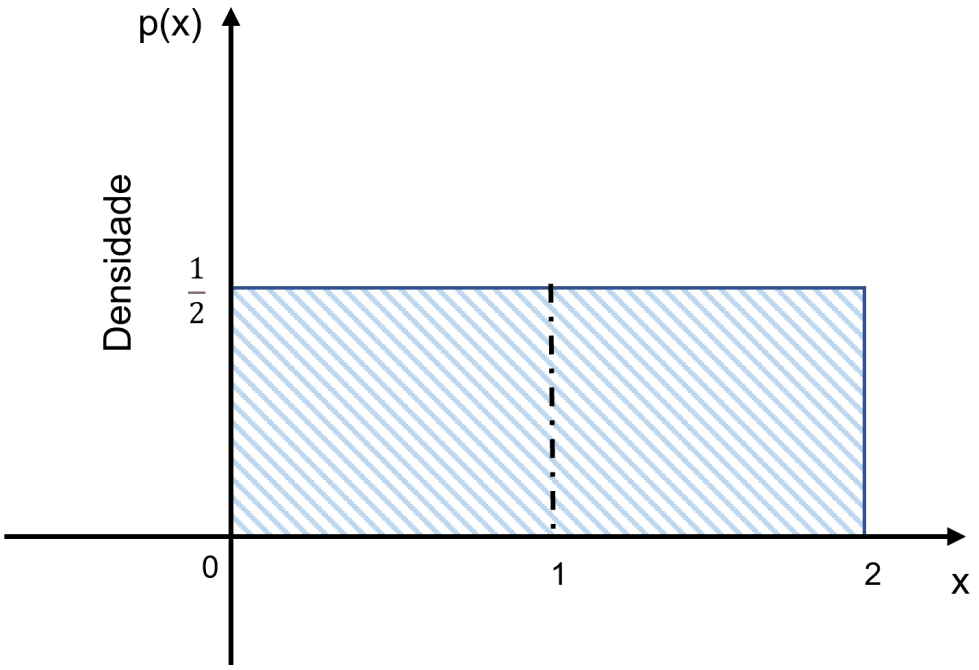


Figura 3.1: Gráfico da pdf de uma distribuição uniforme em um domínio $[0, 2]$. Como a área em todo o domínio deve ser igual a 1.0, podemos inferir que a pdf será $\frac{1}{2}$ para todo o intervalo, inclusive para $x = 1$.

Além da distribuição uniforme, existem diversas outras distribuições importantes, cuja utilidade depende do tipo de aplicação (ver em [34]). Dentre essas distribuições, destaca-se a **Distribuição Gaussiana** (também conhecida na literatura como Distribuição Normal). A importância dessa distribuição está associada ao Teorema Central do Limite que afirma que a soma de N variáveis aleatórias independentes (X), com qualquer distribuição e variâncias semelhantes, é uma variável com distribuição que se aproxima da distribuição de Gauss (distribuição normal) quando N aumenta (ver, p.ex. [32, 33, 34]). A Figura 3.2 ilustra a forma de uma Gaussiana. De acordo com a Figura 3.2, a pdf de 10 é 0.2.

Em renderização, temos que amostrar uma direção no hemisfério unitário. Podemos usar uma distribuição uniforme para fazer isso. Nesse caso, o tamanho do nosso espaço amostral é a área do hemisfério que é igual a 2π. Como a

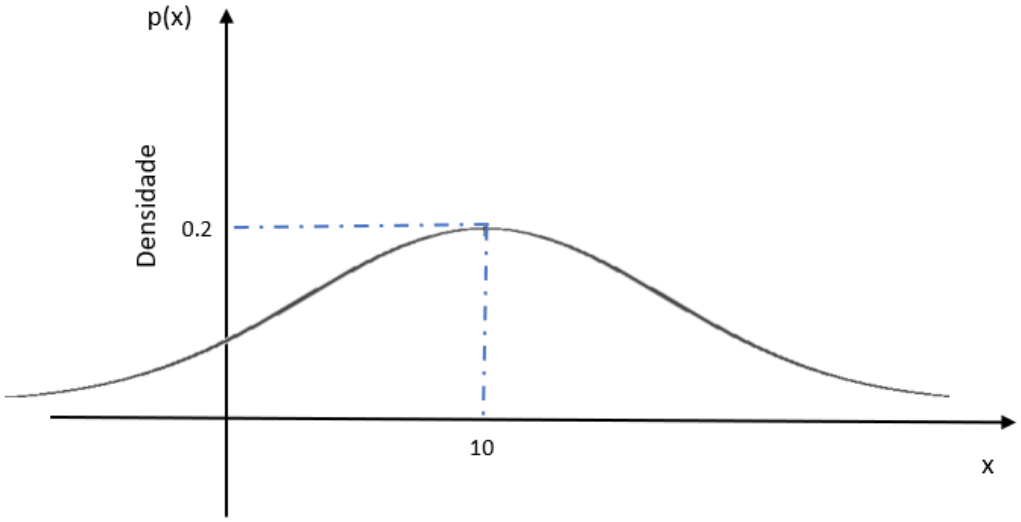


Figura 3.2: Gráfico da pdf de uma distribuição normal em que para um valor 10 amostrado, a pdf é igual a 0.2.

distribuição é uniforme a pdf para qualquer valor dentro do espaço amostral é constante e igual a $\frac{1}{2\pi}$.

Seguindo nos conceitos de probabilidade, é importante falar sobre o **valor esperado**. O valor esperado de uma função $f(X)$, de uma variável aleatória X, denotada aqui por $E_p[f(X)]$, é o valor médio dessa função amostrada de acordo com alguma pdf $p(x)$ (pode ser visto também como um valor médio ponderado pela pdf). O valor esperado em um domínio D é definido como:

$$E_p[f(X)] = \int_D f(x)p(x)d(x) \tag{3-3}$$

A título de ilustração, vamos pegar a função cosseno como exemplo. Num intervalo $[0, \pi]$, se usarmos uma distribuição uniforme (ponderação igual) o valor esperado é igual a zero, o que faz sentido, já que olhando para o trecho de 0 a π do gráfico da função cosseno, e considerando que a pdf é uniforme, as áreas ponderadas (negativas e positivas) se anulam, ou seja:

$$E_p[cos(X)] = \int_0^{\pi} cos(x)\frac{1}{\pi}d(x) = \frac{\sin\pi - \sin 0}{\pi} = 0 \tag{3-4}$$

O último conceito importante para os métodos utilizados nessa dissertação é o de **variância**. A variância $V[f(X)]$ de uma função $f(X)$, de uma variável aleatória X, é o desvio médio quadrático dessa função com relação ao seu valor esperado. Ela é a principal forma de medir erros para alguns métodos como o de Monte Carlo que veremos a seguir. A variância é matematicamente definida como:

$$V[f(X)] = E\Big[\Big(f(X) - E[f(X)]\Big)^2\Big] \tag{3-5}$$

3.2 Monte Carlo

As aplicações de técnicas denominadas de Monte Carlo são numerosas e cobrem uma ampla gama de áreas, incluindo finanças, física, química, biologia computacional, geologia, radiologia computacional, dentre outras. Nesta seção, faremos uma breve discussão sobre o método de Monte Carlo, com foco nas técnicas que serão importantes nesta dissertação.

O método de Monte Carlo é uma das principais ferramentas para resolver as equações de renderização. Ele é muito importante para o algoritmo de traçado de caminhos. Podemos, inclusive, encontrar esse algoritmo também com o nome traçado de raios com Monte Carlo.

Esse método tem uma longa história. Ele se originou no Laboratório Nacional de Los Alamos logo após a Segunda Guerra. Em 1947 John Von Newman usou simulações de Monte Carlo em seu trabalho associado a bomba atômica [35]. Mas foi em 1949 que Metropolis e Ulam publicaram o artigo que apresentou e nomeou esse método [36]. O nome vem da cidade de Monte Carlo em Mônaco, famosa pelos seus cassinos. Para uma introdução ao método de Monte Carlo ver, p.ex., [37, 34]. Para questões referentes a geração de algoritmos, análise de erros e aplicações ver, p.ex., [38].

Os métodos de Monte Carlo calculam os resultados com base em números aleatórios. Historicamente são métodos de natureza probabilística.

O estimador F_N de Monte Carlo para uma integral $\int_a^b f(x)d(x)$, é baseado no conceito de valor esperado. É possível provar que essa integral é o valor esperado do estimador (ver em [2]). Para variáveis aleatórias $X_i \in [a, b]$ uniformemente distribuídas, por exemplo, o estimador dessa integral estaria de acordo com a Equação (3-6).

$$F_N = \frac{b-a}{N} \sum_{i=1}^{N} f(X_i) \tag{3-6}$$

No entanto, é possível generalizar a fórmula do estimador de Monte Carlo para qualquer distribuição e ainda assim a integral será o valor esperado desse estimador. Logo, a fórmula mais geral desse estimador leva em conta a pdf da distribuição utilizada de acordo com a Equação (3-7).

$$F_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)} \tag{3-7}$$

Portanto, a ideia da integração de Monte Carlo consiste em avaliar a integral usando amostragem. Em sua forma básica, isso é feito por amostragem de N variáveis aleatórias X_1,.....,X_n de acordo com alguma função de densidade conveniente $p(X_i)$, e então calculando uma estimativa de acordo com o

estimador (3-7).

Muitas integrais que surgem na renderização são difíceis ou impossíveis de avaliar diretamente. Por exemplo, veremos nos próximos capítulos que, para calcular a quantidade de luz refletida por uma superfície em um ponto, devemos integrar a radiância incidente sobre o hemisfério. Fazer isso não é imediatamente claro, pois a função de radiância incidente quase nunca está disponível de forma fechada devido ao efeito complexo e difícil de prever da visibilidade do objeto em cenas realistas.

A integração de Monte Carlo permite estimar a radiância refletida simplesmente escolhendo um conjunto de direções sobre a esfera, computando a radiância incidente ao longo delas e aplicando um termo de ponderação.

Uma das grandes vantagens do método de Monte Carlo é sua simplicidade. É preciso apenas de uma estratégia de amostragem em que seja possível obter amostras e suas respectivas pdf's.

3.3 Amostragem por Importância

Para que o estimador de Monte Carlo seja mais próximo do valor da integral com menos amostras, é necessária alguma técnica de amostragem que reduza a variância. Dentre as técnicas que usam o método da redução de variância (variance-reduction technique), destaca-se, no contexto de renderização, a técnica de **amostragem por importância** (importance sampling) (ver em [37] e [38]).

A amostragem por importância [39] se baseia na ideia de que o estimador de Monte Carlo (Equação (3-7)) converge mais rápido se a pdf $p(x)$ da distribuição utilizada for similar à função $f(x)$, de forma que a expressão de dentro do somatório se torne algo próximo a uma constante.

A amostragem por importância é uma das técnicas mais úteis e poderosas do método de integração de Monte Carlo [18]. Ela é particularmente útil para integrandos que têm grandes valores em um parte relativamente pequena do domínio. Ao concentrar o trabalho onde o valor do integrando é relativamente alto, uma estimativa precisa é calculada de forma mais eficiente.

Em renderização, a amostragem por importância pode ser usada quando amostramos direções no hemisfério. A amostragem uniforme poderia ser utilizada, porém ela não é tão eficiente. A técnica de amostragem por importância mais conhecida para esse caso é a amostragem com ponderação por cosseno (Cosine-Weighted Hemisphere Sampling). Na equação de renderização, o integrando possui um cosseno do ângulo entre a normal da superfície e a direção do raio de luz. A amostragem com ponderação por cosseno, faz uma amostragem

levando em conta esse cosseno (ver em [2]) e como consequência, a fórmula da pdf acaba contendo esse cosseno. Com isso é feita uma amostragem por importância, já que conseguimos aproximar a função f da função p na fórmula do estimador de Monte Carlo (3-7).

Uma outra razão para utilizar a amostragem por ponderação do cosseno está associado ao fato do estimador convergir mais rápido. Se amostrássemos, por exemplo, uniformemente, poderíamos ter amostras em que o cosseno seria zero e portanto não contribuiriam em nada para o resultado. Já com a amostragem por cosseno, amostramos direções que maximizem o cosseno entre a normal e a direção escolhida e assim contribuam mais para a radiância final. Na Figura 3.3, podemos ver um exemplo de uso da amostragem por importância em renderização.

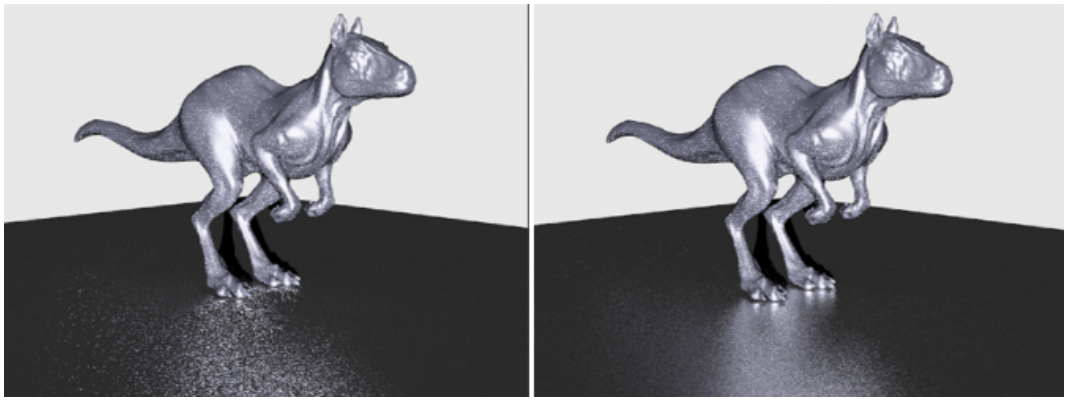


Figura 3.3: Imagem retirada de [2] que compara a amostragem uniforme (esquerda) com a amostragem com ponderação por cosseno (direita).

3.4 Múltipla Amostragem por Importância

Em alguns casos, é necessário fazer a amostragem de um produto de funções, ou seja, pode ser que uma integral seja da forma $\int f(x)g(x)dx$. No entanto, muitas vezes, temos uma estratégia de amostragem para a função f e outra diferente para a função g e não temos uma técnica eficiente para amostrar o produto das duas. Poderíamos usar a estratégia de amostragem da função f ou da função g, a questão é qual seria a melhor opção.

Um exemplo prático em renderização é o de estimar a luz direta para um ponto (next event estimation). Nesse caso temos um produto de funções na integral a ser calculada (Equação 3-8). Uma das duas amostragens não vai ser tão boa quanto a outra para determinados casos [2].

$$L_o(p, \omega_o) = \int_{S^2} f_r(p, \omega_o, \omega i) L_d(p, \omega_i) |\cos \theta_i| d\omega_i \tag{3-8}$$

Se considerarmos o caso de uma superfície que é um espelho, as direções amostradas pela função (f_r) são limitadas à direção de reflexão especular perfeita. Apenas para essa direção a função f_r não é zero. Por isso, se amostrarmos apenas a fonte de luz (L_d), a amostragem não será eficiente. Se temos uma fonte de luz de área com uma área grande, a situação ainda piora, pois há uma maior possibilidade de direções a serem amostradas para a fonte de luz e isso diminui as chances da direção desejada ser amostrada. Nesse caso específico, a amostragem ideal é a da função f_r.

No entanto, para superfícies com materiais difusos ou plásticos e fontes de luz pequenas, uma amostragem da função L_d seria a mais adequada, pois amostrar uma direção com (f_r) geraria mais variância já que a direção teria grandes chances de não atingir a fonte de luz.

Veach em sua tese de doutorado [18], propôs um método chamado de **múltipla amostragem por importância (MIS)** para combinar as duas amostragens, de forma que a melhor amostragem tivesse um peso maior. Esse método consiste em extrair amostras de ambas distribuições de amostragem, na esperança de que pelo menos uma delas corresponda razoavelmente bem à forma do integrando, mesmo que não saibamos qual será.

O novo estimador de Monte Carlo proposto por Veach, unindo as duas amostragens para colocar a MIS em prática, consiste numa ponderação das duas técnicas de amostragem por um peso ω, como mostrado na Equação (3-9) abaixo.

$$F_N = \frac{1}{n_f}\sum_{i=1}^{n_f}\frac{f(X_i)g(X_i)\omega_f(X_i)}{p_f(X_i)} + \frac{1}{n_g}\sum_{j=1}^{n_g}\frac{f(Y_j)g(Y_j)\omega_g(Y_j)}{p_g(Y_j)} \tag{3-9}$$

Porém, precisamos saber como calcular esse peso ω. Existem algumas formas de calcular esse termo; no entanto, a que na prática reduz mais a variância e é uma boa escolha para os casos de renderização [2] é a **heurística de poder** (power heuristic). Nela, uma constante β é utilizada, e foi empiricamente testada por Veach, que constantou que 2 seria um bom valor para ela. Se considerarmos que faremos uma amostragem de cada função chegamos na Equação (3-10)

$$\omega_s(x) = \frac{p_s(x)^\beta}{\sum\limits_i p_i(x)^\beta} \tag{3-10}$$

A Figura 3.4 mostra uma comparação de testes com fontes de luz e superfícies diferentes. A imagem mais à esquerda testa uma amostragem apenas da fonte de luz e a imagem do meio, uma amostragem apenas da função (f_r). As superfícies que estão mais acima se aproximam mais de um

espelho perfeito e as de baixo são mais rugosas. A amostragem da luz direta, como já mencionamos, não é ideal para o caso da fonte de luz maior com uma superfície de espelho. Já a amostragem com (f_r) não é ideal para fontes de luz menores com superfícies difusas ou mais rugosas. Quando as duas amostragens são unidas utilizando a técnica de MIS (Figura 3.4 mais a direita), o melhor das duas amostragens é aproveitado.

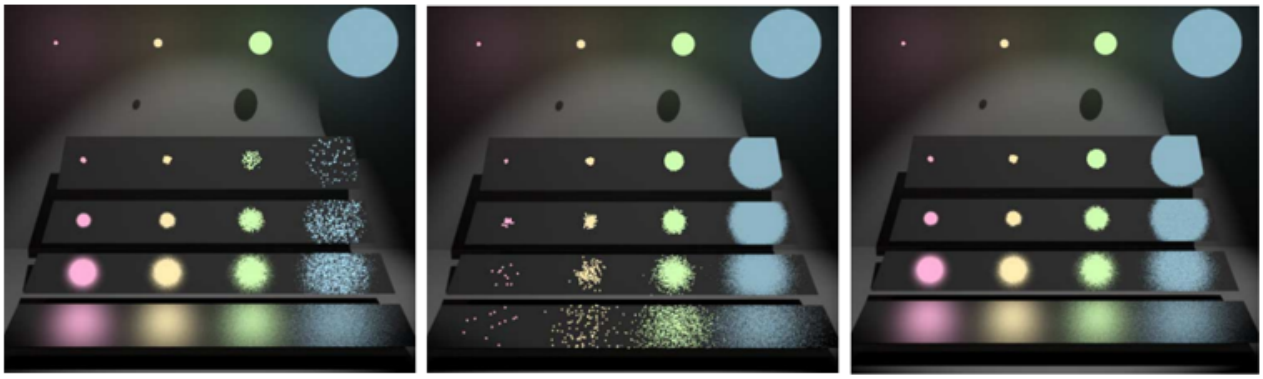


Figura 3.4: Imagem retirada de [3], que compara as diferentes amostragens. A primeira imagem a esquerda é resultado da amostragem só utilizando a fonte de luz. A imagem do meio é resultado de uma amostragem apenas da função (f_r). A imagem mais a direita é a união das duas amostragem utilizando a técnica de MIS.

4
Renderização de Superfícies

Neste trabalho, tratamos de superfícies representadas por malhas de triângulos tridimensionais. Podemos interpretar superfícies como volumes quase infinitamente densos, ou seja, diferentemente de volumes, quando vamos renderizar esse tipo de dado, não precisamos nos preocupar com o que há no interior, pois não consideramos que a luz entra nesses objetos. No entanto, é importante frisar que é possível ter objetos que possuem uma superfície que é preenchida com volume por dentro, mas não consideramos esse caso neste trabalho.

4.1
Equação de Renderização

Em 1986, Jim Kajiya publicou o artigo que introduziu uma equação integral que generalizava uma variedade de algoritmos de renderização conhecidos, chamada de **Equação de Renderização** [6]. Nesse artigo, além de introduzir a nova equação, ele também propunha um novo algoritmo, variante do traçado de raios, que recebeu o nome de traçado de caminhos.

$$L(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} f_r(x, \omega_i, \omega_o)(\omega_i \cdot n) L(x, \omega_i)\, d\omega_i \qquad (4\text{-}1)$$

A equação de renderização (4-1) calcula a radiância $L(x, \omega_o)$ chegando ao ponto x e incidindo na direção ω_o. A superfície pode funcionar também como uma fonte de luz, emitindo uma quantidade de luz $L_e(x, \omega_o)$. A integral da equação representa o acúmulo de luz vinda de todas as direções possíveis no hemisfério Ω em torno da normal da superfície.

A função f_r é chamada de **função de distribuição de refletância bidirecional (BRDF)**. Esta função descreve o tipo de material com o qual estamos lidando (metal ou dielétrico, claro ou escuro, brilhante ou fosco). A BRDF define qual proporção de luz proveniente de ω_i é refletida na direção ω_o. Essa função será discutida com um pouco mais de detalhes na Seção 4.3.

O produto $\vec{\omega_i} \cdot \vec{n}$ representa o $cos(\theta)$ onde θ é o ângulo entre a luz incidente e a normal da superfície. Se imaginarmos um feixe de luz paralelo atingindo uma superfície de frente (90°) e compararmos com o mesmo feixe atingindo a superfície em um ângulo próximo a 0°, veremos que na segunda situação a luz se espalhará por uma área maior, mas cada ponto nessa área parecerá mais escuro do que na primeira situação em que a luz se concentrará em uma área menor. O cosseno é responsável por esse controle.

4.2 Traçado de caminhos vs outros algoritmos de renderização

O traçado de caminhos é uma melhoria do algoritmo de traçado de raios. Ele permite a implementação de outros efeitos como: inter-reflexão difusa, emissão e sombras suaves. Esses efeitos são possíveis graças à formulação da equação de renderização [6].

No algoritmo de traçado de raios, um raio é lançado para cada pixel e a cor desse pixel é calculada levando em conta todas as fontes de luz da cena. Já no algoritmo de traçado de caminhos, há uma simulação mais realista do movimento da luz no ambiente, ou seja, cada raio lançado ricocheteia pelos objetos e acumula as contribuições de cor e iluminação durante toda a trajetória desse raio. O raio só para de ricochetear quando encontra uma fonte de luz ou quando ricocheteia um número máximo de vezes.

No algoritmo de traçado de caminhos, o raio lançado para o pixel pode não atingir nenhuma fonte de luz e por isso as imagens geradas por esse algoritmo podem apresentar ruídos. Por isso, é necessário traçar mais de um raio por pixel. Quanto mais raios, menor a quantidade de ruído.

Para fins de representação de caminhos de raio em algoritmos de traçado de raios, Heckbert [40] introduziu uma notação usando expressões regulares. Nessa notação, consideramos o E como sendo a câmera (eye) e o L como sendo a fonte de luz (light). Além disso, também usamos letras para representar cada tipo de interação com a superfície. A letra D representa uma reflexão ou transmissão difusas, ou seja, nesse tipo de evento o raio de luz tem a mesma probabilidade de seguir em qualquer direção. A letra S representa uma reflexão ou refração especulares, ou seja, há apenas uma direção possível para o raio de luz seguir, baseada na direção de entrada desse raio. A letra G representa uma reflexão ou transmissão brilhosas (glossy) que tem um comportamento intermediário entre difuso e especular.

De acordo com Arvo et al [41], algoritmos que usam traçado de raios podem ser classificados pelos possíveis caminhos de raios que eles consideram, como demonstrado na tabela abaixo, que exemplifica alguns dos principais algoritmos.

Tabela 4.1: Comparação de algoritmos

Autor	**Algoritmo**	**Caminho**
Appel	Ray Casting	$E(D\|G)L$
Whitted	Ray Tracing	$E[S^*](D\|G)L$
Kajiya	Path Tracing	$E[(D\|G\|S) + (D\|G)]L$

Considerando que todo caminho deve envolver uma luz L, o olho E e

pelo menos uma superfície, um caminho tem comprimento mínimo igual a 3. Com essa notação, fica claro quando certos tipos de caminhos não são rastreados e, portanto, quando certos tipos de transporte de luz não são considerados pelo algoritmo. Por exemplo, o algoritmo de projeção de raios (Ray Casting) apenas traça caminhos de comprimento 3, ignorando caminhos mais longos. Já o algoritmo de traçado de raios (ray tracing) pode traçar caminhos de qualquer comprimento, mas todos os caminhos começam com uma sequência de 0 ou mais reflexões ou refrações de espelho (especular). Assim, esse algoritmo ignora caminhos como $EDSDSL$ ou $E(D|G)^*L$. O algoritmo de traçado de caminhos, que é nosso foco neste trabalho, possibilita múltiplos ricocheteamentos envolvendo dispersões não especulares como o caminho $E(D|G)^*L$, e assim, considera iluminação indireta.

4.3 BRDF

A BRDF descreve a refletância da superfície para determinada combinação de direções de entrada e saída da luz. Em outras palavras, ela determina quanta luz é refletida em determinada direção quando uma certa quantidade de luz incide de outra direção, dependendo das propriedades da superfície. Por exemplo, para superfícies difusas, a BRDF especifica uma pequena quantidade de reflexão em todas as direções, enquanto que para espelhos, toda a luz que atinge essa superfície é refletida em uma única direção. Por isso, a BRDF é parte fundamental para que se tenha uma renderização fotorrealista, visto que ela é responsável pela simulação do comportamento dos materiais.

BRDF difuso de Lambert: Uma das BRDFs mais simples é a que segue o modelo lambertiano. Ela modela uma superfície difusa perfeita que espalha a iluminação incidente igualmente em todas as direções. Embora este modelo de reflexão não seja fisicamente plausível, é uma aproximação razoável para muitas superfícies do mundo real, como tinta fosca. Sua fórmula (Equação (4-2)) leva em conta o albedo k_d da superfície.

$$f_{Lambert} = \frac{k_d}{\pi} \tag{4-2}$$

BRDF especular de Phong: Outro modelo de reflexão comumente usado com o lambertiano no passado foi introduzido por Phong [42] para fornecer um destaque especular computacionalmente barato para renderização em tempo real. O modelo de reflexão de Phong não deve ser confundido com o modelo de iluminação de Phong.

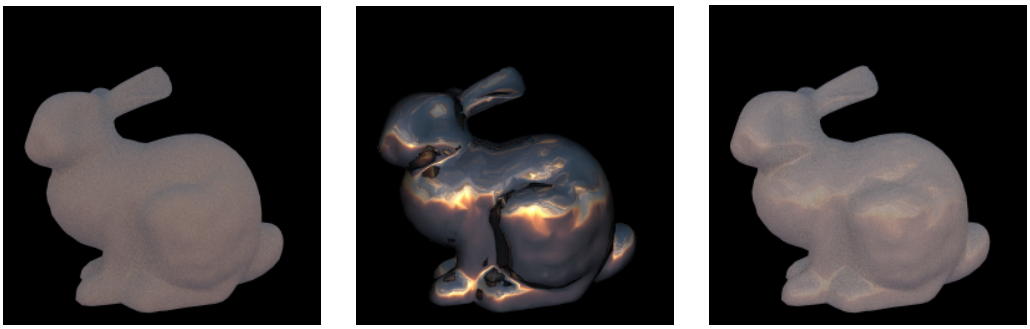
Enquanto o artigo original de Phong afirma que seu modelo de reflexão é baseado fisicamente (derivado da observação de materiais), ele na verdade

não é baseado em física para os padrões de hoje, pois carece de fenômenos como reflexões de Fresnel e outros. Além disso, ele não conserva energia nem é recíproco; no entanto, houve tentativas de corrigir essas deficiências e torná-lo adequado para ser utilizado como BRDF. Mesmo com algumas mudanças com relação à formulação original, sua fórmula continua simples, levando a uma implementação direta [43].

A Equação (4-3) leva em conta a especular k_s da superfície e uma variável $\alpha \in [0, 1]$, que está relacionada à suavidade da superfície.

$$f_{Phong} = k_s \frac{(\alpha + 2)}{2\pi} (\omega_r \cdot \omega_o)^\alpha \tag{4-3}$$

Existem diversas outras BRDFs utilizadas dependendo da situação e do material que se deseja simular (ver em [2] e [44]). Além disso, é possível fazer um mix de BRDFs usando roleta russa para que seja possível simular outros materiais. Um material brilhoso (glossy), por exemplo, pode ser simulado unindo a BRDF de Lambert com a BRDF de Phong. Na Figura 4.1, é feita uma comparação entre a mesma malha de coelho com materiais que usam diferentes BRDFs.



(a) BRDF difusa de Lambert

(b) BRDF especular de Phong com $\alpha = 1.0$

(c) BRDF difusa de Lambert junto com o especular de Phong

Figura 4.1: Malhas de triângulos de coelhos renderizadas usando diferentes tipos de materiais com BRDFs diferentes e iluminadas por um mapa de luz. As imagens foram renderizadas com o framework desenvolvido.

4.4 Fontes de luz

Uma parte importante para se obter uma renderização realista com o efeito desejado é a escolha das fontes de luz. Pode-se ter uma ou mais fontes de luz em uma cena que devem ser amostradas (ver em [2]). É necessário que se tenha, para todas as fontes de luz: uma forma de amostragem de uma posição aleatória na fonte de luz; a radiância L_i para uma amostra; a pdf dessa amostra e a pdf_{eval} que é a pdf para uma direção avaliada qualquer. Nesta dissertação

5 tipos de fontes de luz foram exploradas: luz pontual, luz direcional, luz de área, luz de área infinita (ambiente) e mapa de luz.

4.4.1 Luz pontual

A luz pontual é uma das mais simples, pois a amostra da posição é sempre a mesma. Logo, a pdf para essa amostra é igual a 1. A radiância L_i depende da intensidade I definida para a fonte e da distância ao quadrado do ponto de luz p_{Light} para o ponto p, de acordo com a Equação (4-4).

$$L_i = \frac{I}{||p - p_{light}||^2} \tag{4-4}$$

$$pdf = 1 \tag{4-5}$$

Precisamos ser capazes de calcular a radiância e a pdf dessa luz para qualquer direção. No caso da fonte pontual, como estamos lidando com um ponto no espaço é muito improvável que uma direção recebida intercepte esse ponto. Logo, podemos considerar que a pdf_{eval} de uma direção avaliada qualquer será sempre zero. É importante sermos capazes de avaliar e calcular essa pdf, pois ela é útil quando implementamos a amostragem por importância múltipla que foi vista na Seção 3.4.

$$pdf_{eval} = 0 \tag{4-6}$$

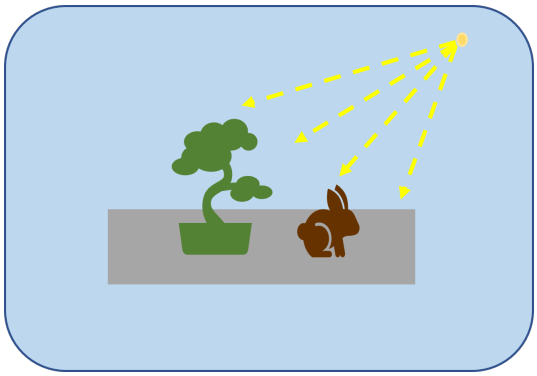


Figura 4.2: Esquema de fonte pontual em cena

4.4.2 Luz direcional

A luz direcional, também chamada de luz distante, simula uma fonte que emite luz em apenas uma direção em todos os pontos do espaço. Por exemplo, o sol pode ser considerado como uma fonte de luz direcional. Embora ele seja na verdade uma fonte de luz de área, a iluminação chega efetivamente à Terra em feixes paralelos porque está muito distante.

Esse tipo de fonte de luz tem apenas uma opção de amostra, pois só há uma direção possível para cada ponto analisado. Logo, assim como na luz pontual, a pdf da amostra é sempre igual a um.

A radiância L_i desse tipo de luz é constante e depende apenas da intensidade definida para a fonte de luz. Assim como no caso da luz pontual, é muito improvável que a direção avaliada recebida seja exatamente a direção da fonte, sendo que as possibilidades são infinitas. Logo, podemos considerar também que a pdf_{eval} é zero.

$$L_i = I \tag{4-7}$$

$$pdf = 1 \tag{4-8}$$

$$pdf_{eval} = 0 \tag{4-9}$$

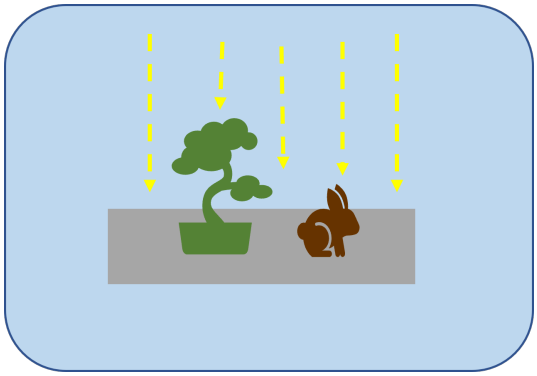


Figura 4.3: Esquema de fonte direcional em cena. Com direção escolhida sendo a direção vertical. Todos os raios de luz partem da fonte com a mesma direção.

4.4.3 Luz de área

A luz de área tem uma complexidade maior, visto que depende da forma da fonte de luz. Nesta dissertação a forma escolhida para os testes foi a fonte retangular, no entanto, é possível ter fontes de qualquer forma, desde que se tenha uma técnica de amostragem de pontos nessa fonte (ver em [45]).

Shirley et al [45] propõem uma técnica para amostrar pontos em uma fonte de luz de formato retangular. O ponto pode ser amostrado de acordo com a Equação (4-10), sendo p_{min} o ponto inferior esquerdo do retângulo, v_1 e v_2 dois vetores que formam o retângulo e ξ_1 e ξ_2 dois números aleatórios.

A *pdf* para essa amostra é inversamente proporcional à área do retângulo. No entanto, precisamos levar em conta também a distância ao quadrado entre o ponto corrente p e o ponto amostrado da fonte de luz p_{light} e o ângulo entre a

normal n da fonte de luz e a direção do raio indo do ponto de referência até o ponto amostrado da fonte ω, (ver em [2]), como mostrado na Equação (4-11).

$$p_{light} = p_{min} + \xi_1 * \vec{v_1} + \xi_2 * \vec{v_2} \tag{4-10}$$

$$pdf = \frac{||p - p_{light}||^2}{area\ |\vec{n} \cdot \vec{\omega}|} \tag{4-11}$$

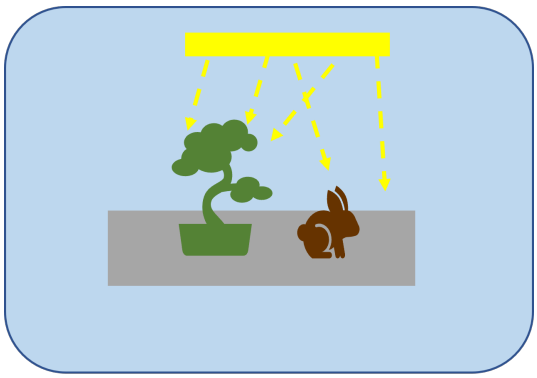


Figura 4.4: Esquema de fonte de área em cena. A fonte tem formato retangular. Pontos aleatórios são amostrados dessa fonte e o raio parte desse ponto até o ponto da cena corrente.

4.4.4 Luz de área infinita (ambiente)

Nesse tipo de fonte de luz, consideramos que a cena está envolvida por uma grande esfera de luz como mostrado na Figura 4.5. Essa fonte se assemelha à luz ambiente, pois ela vem de todas as direções. Para amostrar uma direção ω_i nessa fonte de luz, é primeiramente feita uma amostragem de pontos na esfera unitária (ver em [2]). A posição p da amostragem da fonte de luz é calculada usando a função do raio, considerando a posição do ponto p_{scene} da cena em questão como a origem, e uma distância l maior do que o raio da cena inteira, como na Equação (4-12).

Como a esfera envolve a cena inteira, sempre teremos um L_i constante. A pdf também será constante, pois podemos amostrar uniformemente direções da esfera unitária.

$$p = p_{scene} + \omega_i l \tag{4-12}$$

$$L_i = I \tag{4-13}$$

$$pdf = pdf_{eval} = \frac{1}{4\pi} \tag{4-14}$$

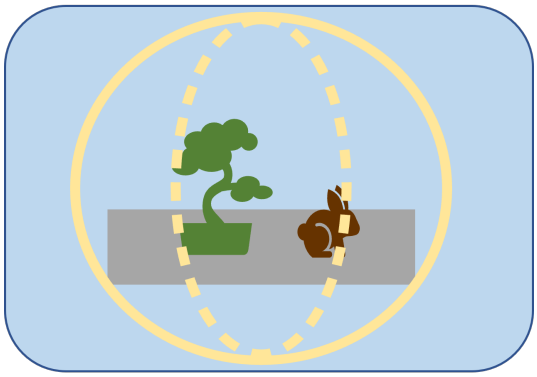


Figura 4.5: Esquema de fonte de área infinita em cena. Uma esfera envolve toda a cena e a ilumina com mesma intensidade de todas as direções.

4.4.5 Mapa de luz

O mapa de luz também considera que há uma esfera envolvendo a cena. No entanto, ao invés de todos os pontos emitirem a mesma intensidade de luz, há uma textura projetada na esfera que dita quanto de radiância é emitida e qual a cor dessa radiância, para determinado ponto amostrado da esfera.

As texturas utilizadas como mapas de luz devem ser derivadas de imagens panorâmicas para que não haja distorções ao fazermos o mapeamento para a esfera. Esse tipo de fonte leva a uma iluminação natural dos dados renderizados, em contraste com fontes de luz sintéticas como as apresentadas anteriormente.

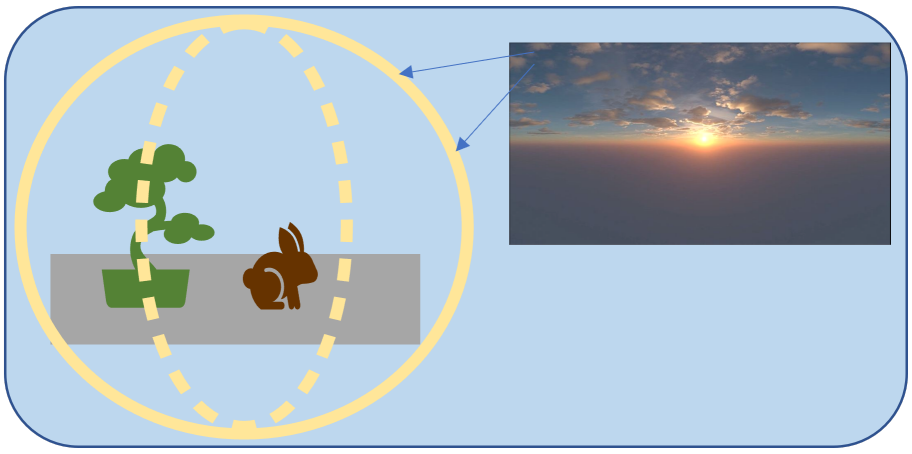


Figura 4.6: Esquema de mapa de luz em cena. Uma imagem HDR é utilizada e mapeada para o interior da esfera que envolve a cena.

Na Figura 4.7, uma esfera é iluminada por diferentes tipos de fontes

de luz. Os dois primeiros tipos (pontual e direcional), produzem sombras escuras (dark shadows), enquanto os outros tipos produzem sombras leves (soft shadows).

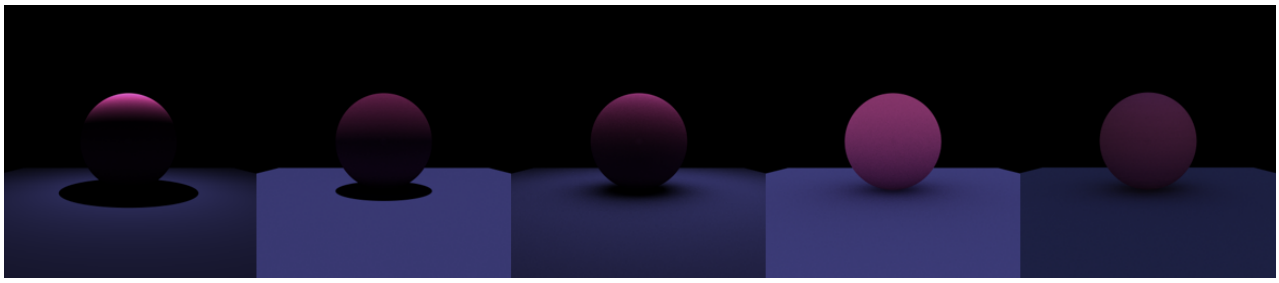


Figura 4.7: Uma mesma cena de uma esfera em cima de um plano é iluminada por diferentes tipos de fontes de luz. Da esquerda para a direita: Fonte pontual, fonte direcional, fonte de área, fonte de área infinita e mapa de luz. Imagem renderizada pelo framework desenvolvido.

5
Renderização de Volumes

Volumes são coleções de partículas, variando de átomos e moléculas para qualquer tamanho de partícula, incluindo estrelas, por exemplo. Esse tipo de dado é muito comum na computação científica e uma representação bastante utilizada é uma grade tridimensional de pontos amostrais.

Matrizes tridimensionais de dados digitais representando volumes surgem em muitas aplicações científicas. Tomografia computadorizada (TC) e ressonância magnética (RM) podem ser usadas para criar um volume através da união de imagens de uma série de cortes transversais. Medições astrofísicas, meteorológicas e geofísicas, e simulações de computador usando modelos de elementos finitos de tensão, fluxo de fluido etc. também geram naturalmente um conjunto de dados de volume [46]. Além disso, volumes são essencias em produção de filmes, principalmente quando se trata de representação de efeitos naturais (participating media), como por exemplo: nuvens, explosões, fog etc.

A seguir descrevemos: (1) as unidades de volumes e como eles são representados e estruturados; (2) como a luz se propaga em volumes e quais os tipos de colisões com suas partículas; (3) como volumes podem ser categorizados; (4) a equação de renderização volumétrica e seus termos; (5) o termo trânsmitância que é fundamental para a equação de renderização; (6) a técnica utilizada para amostrar pontos no volume e calcular a radiância nesses pontos; (7) como e quando funções de fase ou BRDFs são utilizadas em volumes; (8) como a dispersão de entrada pode ser contabilizada no cálculo de radiância.

5.1
Estrutura de dados

Um dado volumétrico consiste em um conjunto de elementos, contendo algum campo escalar ou vetorial, representando alguma propriedade. Cada um desses elementos tem uma localização 3D, podendo possuir ou não uma correlação espacial entre seus vizinhos.

Para visualizar corretamente um dado volumétrico, é necessário ter uma estrutura que possibilite acessar os seus respectivos elementos de maneira coerente. Normalmente, essas estruturas são chamadas de **malhas estruturadas** e **malhas não-estruturadas**. Neste trabalho, lidamos com malhas estruturadas apenas.

Nas malhas estruturadas, cada elemento pode ser acessado através de uma indexação lógica espacial (i, j, k). A Figura 5.1 ilustra o caso mais simples

de malha estruturada, com formato de cubo, que é a usada pelos volumes presentes nesta dissertação.

As malhas estruturadas podem ser interpretadas como um conjunto de voxels ou um conjunto de células. Essas duas estruturas são ilustradas pela Figura 5.2. No caso dos voxels, cada um deles é definido como um ponto de malha no espaço 3D que possui uma região de influência com valor constante. Essa região possui um formato de hexaedro que não possui variação em torno do seu ponto central de malha. Em alguns algoritmos, a contribuição do voxel diminui à medida em que se distancia do ponto central. Em outros casos, o voxel tem uma contribuição constante em toda a região de influência [47].

Já as células em um volume são normalmente interpretadas como um conjunto de hexaedros cujos vértices são pontos de malha e seu respectivo campo escalar é variável. Estima-se o campo escalar dentro de cada célula a partir da interpolação dos valores determinados nos seus vértices [47].

Os dados de volume usados nesta dissertação são grades tridimensionais escritas em arquivos com os formatos .raw e .pvm. Nesses arquivos de volumes, se encontram: a dimensão da grade e os valores escalares para cada vértice das células que compõem o volume.

No entanto, mesmo assim, é necessário que se definam atributos para visualização desses dados. Por isso, cada volume possui uma **Função de Transferência** que define uma cor e uma opacidade para cada valor escalar do volume.

Como a implementação do renderizador desta dissertação foi feita em CPU, foi necessário fazer uma interpolação trilinear dos valores escalares de cada vértice de uma célula para que se obtivesse o valor correto da densidade em determinado ponto de interesse dentro dessa célula.

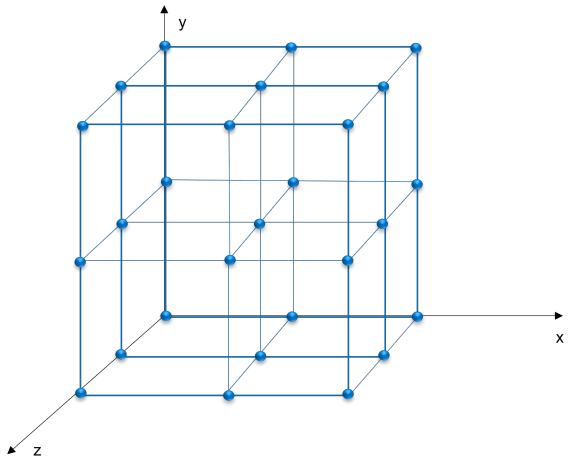


Figura 5.1: Malha estruturada

(a) Voxel (b) Célula

Figura 5.2: Unidades de volume.

5.2 Propagação da luz em volumes

À medida que um fóton de luz percorre um volume, ele pode colidir com as partículas que o compõe. Essas colisões definem a distribuição de radiância ao longo do volume. Já que não é viável modelar cada partícula, elas são tratadas como campos de probabilidade de colisão. Essencialmente, colisões de partículas são instanciadas estocasticamente.

A chance de colisão de um fóton é definida por um coeficiente $\sigma(x)$, que representa a densidade de probabilidade de colisão por unidade de distância percorrida dentro do volume. A unidade de um coeficiente de colisão é o inverso do comprimento. Em geral, consideramos o caso em que os coeficientes são funções da posição que variam quando apropriado [10].

Existem 4 tipos principais de eventos que podem ocorrer quando fótons colidem com partículas do volume:

Absorção: Esse evento ocorre quando um fóton colide com uma partícula e essa partícula absorve parte da luz para si, deixando o raio de luz seguir com menos energia. Para que os eventos de absorção sejam levados em conta na radiância final, o coeficiente de absorção σ_a é utilizado.

Dispersão de saída (Out-Scattering): O raio de luz também perde energia na direção ω que está seguindo quando ocorre o evento de dispersão de saída. Nesse evento, a luz é dispersada, como o nome sugere, para outras direções e uma intensidade menor de radiância segue no caminho inicial do raio.

Emissão: Eventos de emissão ocorrem quando a própria partícula do volume emite luz e adiciona energia no raio de luz amostrado. A emissão é um campo de radiância separado $L_e(x, \omega)$ que define a radiância adicionada ao campo de radiância final $L(x, \omega)$.

Dispersão de entrada (In-Scattering): Nesse tipo de evento, raios de luz de diversas direções chegam à partícula do volume e uma parcela de energia é adicionada ao raio de luz que segue na direção ω inicial.

Na Figura 5.3, esses eventos são ilustrados. Todos eles são importantes para o cálculo da radiância final, para que se obtenha uma imagem com maior fotorrealismo.

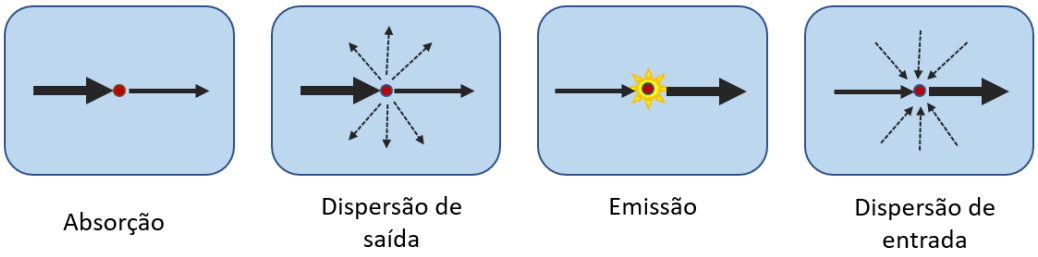


Figura 5.3: Eventos de volume

Em muitos casos, é desejável escolher uma parametrização diferente dos coeficientes de absorção e dispersão. É possível definir o coeficiente de **extinção** $\sigma_t = \sigma_a + \sigma_s$ como a soma dos coeficientes de absorção e dispersão. A extinção define a perda de radiância devido a absorção e a dispersão, e podemos modelar essa perda com um único coeficiente. Pode-se, então, interpretar uma colisão de extinção como uma colisão onde tanto a dispersão quanto a absorção estão envolvidas.

Na prática, podemos interpretar σ_t como a opacidade de determinado ponto no volume , ou seja, é possível interpretar a opacidade definida na função de transferência como sendo o coeficiente de extinção [19]. Além da extinção, outra parametrização que pode ser utilizada é o **albedo de dispersão única** (single scattering albedo) $\alpha = \sigma_s/\sigma_t$. Ele tem significado similar ao albedo de superfície, pois ambos são uma medida que determina a quantidade de radiância que será dispersada. Quando $\alpha = 0$ toda a radiância é absorvida (por exemplo carvão preto) e quando $\alpha = 1$ nenhuma absorção ocorre (nuvens). Na prática, o albedo é importante para levar em conta a cor da partícula no cálculo da radiância.

5.3 Categorias

Novák et al [4] divide os volume em categorias dependendo de algumas de suas características. Eles são diferenciados pelas suas **dependências espaciais**, ou seja, as propriedades do volume (σ_a, σ_s, etc) podem ou não variar conforme a posição. Se essas propriedades não mudam, ele é chamado de **homogêneo**, mas se elas mudam ele é chamado de **heterogêneo**.

Outra classificação de volumes está relacionada às suas **dependências direcionais**, ou seja, quando os coeficientes de colisão σ_a e σ_s não dependem da direção de propagação da luz, então o volume é dito **isotrópico**. Se os

coeficientes de colisão dependem da direção do raio incidente ou luz dispersada, ou seja, a resposta do material varia com a direção de propagação, o volume é referido como **anisotrópico**.

5.4 Equação de Renderização Volumétrica

A distribuição de radiância em volumes é definida pela Equação (5-1). Ela é chamada de **equação de transferência radiativa** (Radiative Transfer Equation (RTE)) e é parametrizada pela posição x e pela direção ω.

Essa equação leva em consideração todos os eventos já mencionados na Seção 5.2, que podem ocorrer quando um raio atinge uma partícula do volume: absorção, dispersão de saída, dispersão de entrada e emissão, porém os termos de absorção e dispersão de saída foram combinados no termo de extinção, como já sugerido na Seção 5.1.

$$(\omega\cdot\nabla)L(x,\omega) = \underbrace{-\sigma_t(x)L(x,\omega)}_{\text{termo de extinção}} + \underbrace{\sigma_a(x)L_e(x,\omega)}_{\text{termo de emissão}} + \underbrace{\sigma_s(x)\int_{S^2} f(x,\omega,\omega')L(x,\omega')\,d\omega}_{\text{termo de dispersão de entrada}} \tag{5-1}$$

A RTE formula a distribuição de radiância usando gradientes, definindo o que acontece com a radiância enquanto o raio viaja da câmera até a fonte de luz, ou seja, ela trata a viagem do raio de forma progressiva. Já a Equação (5-2) de renderização volumétrica (VRE) (ver em [48]) é obtida ao integrarmos os dois lados da RTE. Ela provê uma formulação que pode ser usada no algoritmo de traçado de caminhos. Nessa equação, o termo de dispersão foi reduzido para L_s (Equação (5-3)).

$$L(x,\omega) = \int_{t=0}^{d} exp\Big(-\int_{s=0}^{t}\sigma_t(x_s)ds\Big)\Big[\sigma_a(x_t)L_e(x_t,\omega) + \sigma_s(x_t)L_s(x_t,\omega)\Big]dt \tag{5-2}$$

$$L_s(x,\omega) = \int_{S^2} f(x,\omega,\omega')L(x,\omega')\,d\omega \tag{5-3}$$

A VRE descreve a radiância recursivamente, em termos de onde o raio vem, olhando para trás e acumulando todas as contribuições através de integrais. Para simplificar ainda mais essa equação, pode-se agrupar o termo exponencial em uma variável $T(t)$ que representa a transmitância (Equação (5-4)) e que será melhor explicada na Seção 5.5. Como resultado, temos a equação de renderização volumétrica final usada nesta dissertação (Equação (5-5)).

$$T(t) = exp\Big(- \int_{s=0}^{t} \sigma_t(x_s)ds \Big) \tag{5-4}$$

$$L(x,\omega) = \int_{t=0}^{d} T(t)\Big[\sigma_a(x)L_e(x_t,\omega) + \sigma_s(x)L_s(x_t,\omega)\Big]dt \tag{5-5}$$

A Figura 5.4 representa a construção da VRE com as contribuições dos eventos da Equação (5-5). A equação de renderização de volume (VRE) pode ser interpretada como uma generalização da equação de renderização [6] para incluir estruturas volumétricas, já que superfícies são volumes quase infinitamente densos com funções de fase complexas (BRDFs).

Existem duas abordagens estocásticas principais em Monte Carlo que podem ser usadas para resolver a VRE. O método de Tracking, que será explicado na Seção 5.6 e o de pdf que não será aprofundado nesta dissertação (ver em [10]).

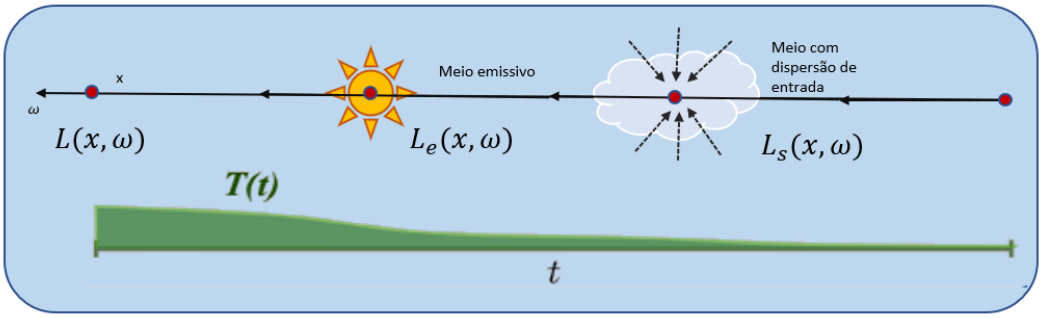


Figura 5.4: Eventos que compõem a Equação de Renderização Volumétrica

5.5 Transmitância

A transmitância $T(x)$ é o fator de redução da absorção e da dispersão de saída entre um ponto x e x_t. A radiância L do ponto x é atenuada pela transmitância, pois no caminho que o raio percorre parte da radiância é absorvida (absorção) e/ou perdida para outras direções (dispersão). A transmitância está sempre entre 0 e 1. A integral da exponencial na Equação (5-4) é chamada de espessura óptica entre dois pontos. Sabendo que a transmitância é uma função de dois pontos, muitas vezes vamos simplificar a notação como uma função de apenas uma distância escalar t.

Uma propriedade muito importante e válida para todos os meios é que a transmitância é multiplicativa ao longo dos pontos de um raio, ou seja, a transmitância da origem a um ponto x(trajetória da Figura 5.5), $T(o \rightarrow x)$, pode ser calculada tomando o produto da transmitância para um ponto

anterior $T(o \rightarrow x')$ e a transmitância do segmento entre o ponto anterior e o atual $T(x' \rightarrow x)$, como mostra a Equação (5-6).

$$T(o \rightarrow x) = T(o \rightarrow x')T(x' \rightarrow x) \tag{5-6}$$

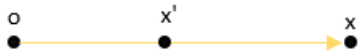


Figura 5.5: Caminho do raio de luz

Para volumes homogêneos, σ_t não é espacialmente variável, logo, a mudança na radiância ao longo do raio é proporcional ao coeficiente de extinção σ_t. Este fato nos permite derivar a transmitância ao longo do raio, conhecida como Lei de Beer-Lambert (ver em [49] e [50]).

$$T(t) = e^{-\sigma_t t} \tag{5-7}$$

Já para volumes heterogêneos, o coeficiente de extinção se altera; portanto, a expressão para a transmitância vira uma integral (Equação (5-4)). Essa integral deve ser calculada andando pelo volume e acumulando as transmitâncias de cada passo. Existem algumas técnicas para realizar esse cálculo que serão exploradas na Seção 5.6.2.

5.6 Rastreamento (Tracking)

A abordagem de rastreamento emprega estratégias de roleta russa e amostragem por rejeição para decidir sobre um único tipo de colisão que está sendo amostrada, em vez de tentar estimar todas elas ao mesmo tempo.

Nessa abordagem estocástica para resolver a VRE, não permitimos que o feixe de radiância se divida em contribuições fracionárias. Escolhemos apenas qual tipo de colisão deve ser modelado. O problema que surge é o de como amostrar as distâncias de caminho livre (free-path distances) que são percorridas por um fóton entre diferentes tipos de colisões para modelar o processo físico. Além disso, para podermos resolver a equação de renderização volumétrica, como pode ser observado na Equação (5-5), é preciso que se tenha uma técnica para o cálculo da transmitância total para uma dada distância.

5.6.1 Amostragem de distância

Existem algumas técnicas utilizadas para amostragem de distância dentro da abordagem de rastreamento. Além dessas que serão citadas, existem outras

técnicas que não serão comentadas nesta dissertação (ver em [51] e [52]).

Rastreamento de forma fechada (closed-form tracking): Em meios homogêneos, onde o coeficiente de extinção σ_t é espacialmente invariante, distâncias podem ser amostradas usando amostragem de transformação inversa. Para fins de amostragem, podemos definir uma função de densidade de probabilidade (pdf) normalizando a transmitância (Equação (5-4)). Se a função de distribuição cumulativa correspondente (cumulative distribution function (CDF)) for analiticamente invertível, então a distância t_0 pode ser amostrada analiticamente usando um número aleatório ξ.

Para o rastreamento, o interesse principal é ser capaz de amostrar a distância t_0 de forma que a transmitância nessa distância seja constante. Pode-se então, fazer uma amostragem por importância de uma função exponencial, para que se possa amostrar essa distância t_0. A pdf para essa amostragem pode ser obtida, normalizando a Equação (5-4) de transmitância e obtendo a fórmula .

$$p(t) = \sigma_t exp(-\sigma_t t) \tag{5-8}$$

Pode-se amostrar perfeitamente essa pdf através do método de amostragem para uma distribuição exponencial discutida por Pharr e Humphreys em [2], obtendo-se:

$$t_0 = -ln(1-\xi)/\sigma_t \tag{5-9}$$

Essa distância t_0 amostrada, pode ser considerada como a distância que um fóton viaja no volume até um novo evento ocorrer em decorrência da colisão com alguma partícula. Pode-se substituir a fórmula obtida da pdf na Equação (5-2), obtendo a Equação (5-10).

$$L(x,\omega) = \int_{t=0}^{d} p(t)\Big[\frac{\sigma_a}{\sigma_t}(x_t)L_e(x_t,\omega) + \frac{\sigma_s}{\sigma_t}(x_t)L_s(x_t,\omega)\Big]dt \tag{5-10}$$

As frações ($P_a = \frac{\sigma_a}{\sigma_t}$ e $P_s = \frac{\sigma_s}{\sigma_t}$) que surgiram nessa fórmula, podem ser interpretadas como probabilides de cada evento do tipo acontecer. O Algoritmo 1, *closed-form tracking*, é derivado da Equação (5-10). Esse método é simples e muito utilizado, no entanto, é limitado para volumes homogêneos.

Algoritmo 1: Closed-Form Tracking.

```
1  Function ClosedFormTracking(x, ω, d):
2      enquanto true faça
3          t ← ln(1−ζ) / σ_t
4          se t > d então        // Um limite do volume foi atingido
5              retorna
6          fim
7          x ← x − tω
8          se ξ < σ_a / σ_t então        // Uma colisão absorção/emissão
               ocorreu
9              retorna L_e(x, ω)
10         senão                 // Uma colisão de dispersão ocorreu
11             ω ← amostra f_p(ω)
12             d ← novo raio terminado em ω
13         fim
14     fim
15 fim
```

Rastreamento regular (regular tracking): Para volumes heterogêneos que possuem partes homogêneas, essa técnica consiste em utilizar o *closed-form tracking* em partes homogêneas do volume para fazer a amostragem da distância (Ver em [4]).

Marcha em raio (ray marching): Para reduzir o custo do rastreamento regular, pode-se ignorar os limites e marchar ao longo do raio utilizando passos com tamanho fixo. Isso simplifica significativamente a implementação. A cada passo o algoritmo consulta o coeficiente de extinção do meio local e então avança por uma distância fixa. No entanto, apesar de ser mais fácil de implementar esse método não é muito preciso e pode introduzir viés.

Rastreamento delta (delta tracking): Essa técnica tem vários nomes: Rastreamento de Woodcock, algoritmo de colisão nula ou pseudo-dispersão. Ela é uma das técnicas mais utilizadas para amostragem de distâncias em volumes heterogêneos e foi a técnica utilizada nesta dissertação.

Essa técnica é baseada em amostragem de rejeição e consiste em introduzir uma colisão fictícia, chamada de colisão nula que homogeniza a densidade total de colisão de forma que a estratégia de amostragem veja um volume homogêneo (constante), da mesma forma que acontecia com o *closed-form tracking*.

Neste novo tipo de colisão, o volume se dispersa na mesma direção que a direção de entrada e também é necessário que esse tipo de colisão

tenha seu coeficiente σ_n como os outros tipos de colisão. Como o intuito é homogeinizar o volume, escolhemos esse coeficiente de forma que a soma de todos os coeficientes (o coeficiente $\bar{\sigma}$) seja constante, como na Equação (5-11).

$$\bar{\sigma} = \sigma_t + \sigma_n \tag{5-11}$$

Uma consequência dessa equação é que o $\bar{\sigma}$ escolhido precisa ser igual ou maior do que o maior valor que σ_t pode assumir no volume. Assim, como feito no *closed-form tracking* (Equação (5-8)), chegamos a uma nova fórmula que inclui probabilidades de cada evento ocorrer, inclusive o novo evento de colisão nula.

$$L(x_j, \omega_j) = \int_{t=0}^{\infty} p_n(t_j)\Big[P_a(x)L_e(x_{j+1}, \omega_j) + P_s(x)L_s(x_{j+1}, \omega_j) + P_n(x)L(x_{j+1}, \omega_j)\Big]dt \tag{5-12}$$

A diferença é que ao invés de usar o σ_t para o cálculo de $p(t_j)$, temos o $p_n(t_j)$ que é calculado utilizando o $\bar{\sigma}$. A implementação do *delta tracking* é muito parecida com a do *closed-form tracking* como mostrado a seguir no Algoritmo 2. A maior diferença é que no rastreamento delta devemos considerar esse novo tipo de colisão. Como mostrado no Algoritmo 2, a distância t_0 utilizada para andar no raio, é igual a calculada no rastreamento de forma fechada (Equação (5-9)).

Algoritmo 2: Delta Tracking.

```
1  Function DeltaTracking(x, ω, d):
2      enquanto true faça
3          t ← ln(1−ζ) / σ̄
4          se t > d então        // Um limite do volume foi atingido
5              retorna
6          fim
7          x ← x − tω
8          se ξ < σ_a / σ̄ então        // Uma colisão absorção/emissão
9              retorna L_e(x, ω)
10         senão se ξ < 1 − σ_n / σ̄ então  // Uma colisão de dispersão
11             ω ← amostra f_p(ω)
12             d ← novo raio terminado em ω
13         senão                                  // Uma colisão nula
14             d ← d − t
15         fim
16     fim
17 fim
```

A Figura 5.6 a seguir mostra uma comparação entre os últimos métodos de rastreamento citados, que podem ser utilizados em volumes heterogêneos. O volume possui pedaços homogêneos o que também permite que métodos como rastreamento regular sejam utilizados.

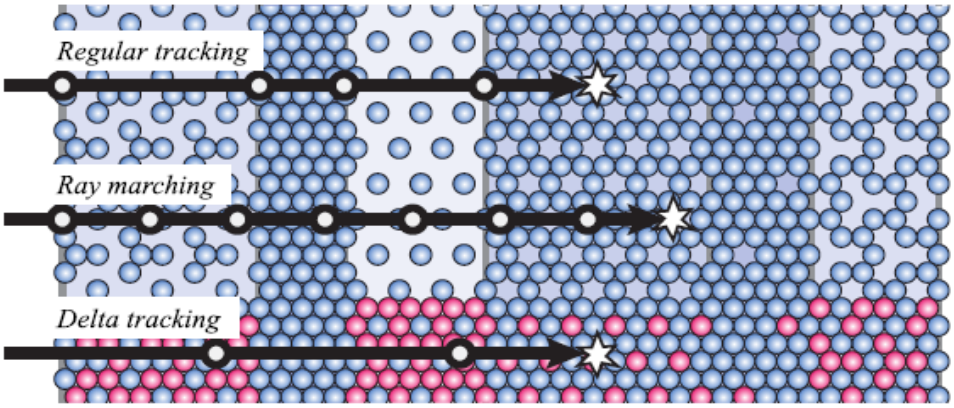


Figura 5.6: Figura tirada de [4] que mostra a comparação entre 3 métodos de rastreamento. O volume possui pedaços homogêneos. Partículas azuis e vermelhas representam matéria real e fictícia respectivamente. O *regular tracking* encontra todos cruzamentos de fronteira e resolve a posição analiticamente. Abaixo é mostrado o algoritmo de *ray marching* com um passo constante. Por último é mostrado o algoritmo de *delta tracking* que primeiro homogeniza o volume usando matéria fictícia e depois compensa isso explicitamente lidando com as colisões nulas resultantes.

5.6.2 Estimadores de Transmitância

Também é necessário estimar estocasticamente a transmitância $T(t)$ transformando algumas das técnicas de amostragem de distância da subseção anterior em estimadores de transmitância. Essa estimativa é importante para o cálculo da VRE que utiliza esse termo.

Estimador de transmitância com marcha em raio (ray marching transmittance estimator): Como já dito na seção anterior, essa técnica consiste em andar pelo raio com um passo constante. Esse método também pode ser utilizado para cálculo da transmitância, acumulando as transmitâncias a cada passo. No entanto, isso introduz viés se a transmitância for subamostrada. Uma vantagem da marcha em raio é que ela não precisa de dados adicionais como os outros métodos sem viés, que precisam de $\bar{\sigma}$, por exemplo. Além disso, com passos bem pequenos esse método pode ser muito preciso, porém bem custoso.

Estimador de transmitância com rastreamento delta (delta tracking transmittance estimator): Podemos usar o mesmo algoritmo de

delta tracking já apresentado para estimar a transmitância. Este estimador produz uma transmitância de $T(t) = 1$, quando um ponto de dispersão não é encontrado ou $T(t) = 0$ quando há um ponto de dispersão devido à roleta russa, portanto, esse é um estimador binário e não muito preciso.

Estimador de rastreamento de proporção (ratio tracking transmittance estimator): Esse estimador que tem relação com o delta tracking foi introduzido por Novák et al [53]. A roleta russa do rastreamento delta é substituída por um peso de Monte-Carlo T que é igual à probabilidade de uma colisão nula em relação a extinção em um ponto de colisão amostrado, resultando no estimador de transmitância. Este método é demonstrado no Algoritmo 3 a seguir e foi utilizado nas implementações desta dissertação para cálculo do feixe de transmitância (beam transmittance) que será melhor explicado na Seção 6.3.

Algoritmo 3: Ratio Tracking.

```
1  Function RatioTracking(x_0, ω_0, d):
2      T ← 1
3      enquanto true faça
4          t ← t − ln(1−ζ)/σ̄
5          se t > d então        // Um limite do volume foi atingido
6              retorna T
7          fim
8          x ← x_0 − tω_0
9          T ← T(1 − σ_t(x)/σ̄)
10     fim
11 fim
```

5.7
Função de fase e BRDFs para volumes

A função de fase $f_p(x, \omega, \omega')$ é a densidade direcional da luz, dispersada em um ponto do volume [4]. Ela é geralmente modelada como uma função 1D do ângulo θ entre as duas direções ω e ω'. Em certo sentido, a função de fase tem a mesma função que uma BRDF para a renderização de superfícies. Uma propriedade importante dessas funções é a reciprocidade: as duas direções podem ser trocadas e o valor da função permanece inalterado [10].

Funções de fase precisam ser normalizadas sobre a esfera, senão elas adicionariam ou subtrairiam radiância em uma colisão de dispersão:

$$f_p(x, \omega, \omega')d\theta = 1 \tag{5-13}$$

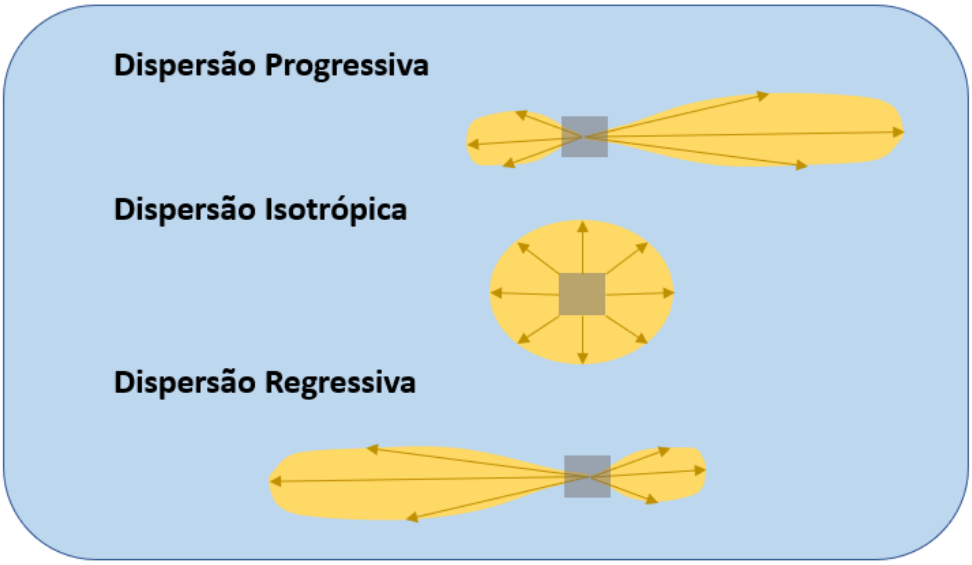


Figura 5.7: Função de Henyey-Greenstein com diferentes valores de g. Em cima o diagrama que mostra como a função se comporta quando g > 0, amostrando direções que estão a frente no volume, no meio demonstrando a dispersão isotrópica (g = 0) e por último a dispersão regressiva quando g < 0.

Se o meio dispersa a luz uniformemente, a função de fase é isotrópica, ou seja, a densidade de probabilidade para qualquer direção é a mesma:

$$f_p(x,\theta) = \frac{1}{4\pi} \tag{5-14}$$

Existem muitas funções de fase anisotrópicas que podem ser modeladas de acordo com o meio, como as de Rayleigh e de Lorenz-Mie [4], no entanto, elas podem ser bem caras e complicadas. Como uma alternativa a essas funções, a função de Henyey-Greenstein [54] é uma função muito utilizada:

$$f_p(x,\theta) = \frac{1}{4\pi} \frac{1-g^2}{(1+g^2-2g\cos\theta)^{\frac{3}{2}}} \tag{5-15}$$

onde $-1 < g < 1$. Um único parâmetro g (chamado de parâmetro de assimetria) controla a distribuição da luz espalhada e pode ser entendido como o cosseno médio das direções de espalhamento.

A função pode modelar a **dispersão regressiva** (backward scattering) ($g < 0$), onde a luz é dispersada de volta para a direção de onde ela vem, a **dispersão isotrópica** (isotropic scattering) ($g = 0$) e a **dispersão progressiva** (forward scattering), onde a luz incidente é dispersada principalmente para frente no volume($g > 0$). Um diagrama mostando cada um desses perfis de dispersão é mostrado na Figura 5.7.

Funções de fase são muito mencionadas quando se trata de volumes, pois para renderização de dados volumétricos de efeitos naturais (participating

media), elas são amplamente utilizadas. No entanto, para outros tipos de dados volumétricos, como dados médicos que envolvem ossos por exemplo, ou dados que devem ter aparência mais opaca como o bonsai apresentado nesta dissertação, BRDFs também podem ser utilizadas, inclusive são mais indicadas.

Na área de renderização cinematográfica, que foca em exames médicos, funções de fase e BRDFs são utilizadas, muitas vezes simultaneamente. Isso porque os diferentes tecidos encontrados em exames, podem ser interpretados como diferentes tipos de materiais. Como ossos são muito opacos, podemos interpretá-los como sendo superfícies e atribuir uma BRDF para eles. Já alguns tecidos moles ou fluidos têm uma certa transparência e uma função de fase pode ser utilizada fornecendo bons resultados.

Kroes et al [29] propõem uma solução para o problema de ter que ficar alterando entre funções de fase e BRDFs. A chamada **dispersão híbrida** junta a dispersão de superfície e a dispersão volumétrica. O desafio se dá por não se terem superfícies bem definidas no volume, assim como há para malhas de superfícies.

Na dispersão híbrida, é calculada a probabilidade P_{brdf} da BRDF ser utilizada ao invés da função de fase. Essa probabilidade leva em conta o coeficiente de extinção σ_t (opacidade) do ponto no volume, a magnitude do gradiente nesse ponto do volume $|\nabla\tau_n(x)|$ e um fator gradiente g, que define o quanto de dispersão de superfície e volumétrica serão utilizadas.

$$P_{brdf} = \sigma_t(x) \cdot (1 - e^{-25 \cdot g^3 \cdot |\nabla\tau_n(x)|}) \tag{5-16}$$

$$P_{phase} = 1 - P_{brdf} \tag{5-17}$$

Dessa forma, o uso da BRDF ou da função de fase na Equação (5-3) é determinado por essas probabilidades e usando também uma roleta russa, sorteando um número ξ, como mostrado na Equação (5-18). A Figura 5.8 mostra como a dispersão híbrida é usada em volumes, dependendo do $sd = 25 \cdot g^3$ escolhido.

$$f = \begin{cases} f_r, \ \xi < P_{brdf} \\ f_p, \ \text{caso contrário.} \end{cases} \tag{5-18}$$

5.8 Dispersão de entrada

Como apresentado na Equação (5-1) há um termo integral para a dispersão de entrada, ou seja, a luz pode vir de direções diferentes e é preciso

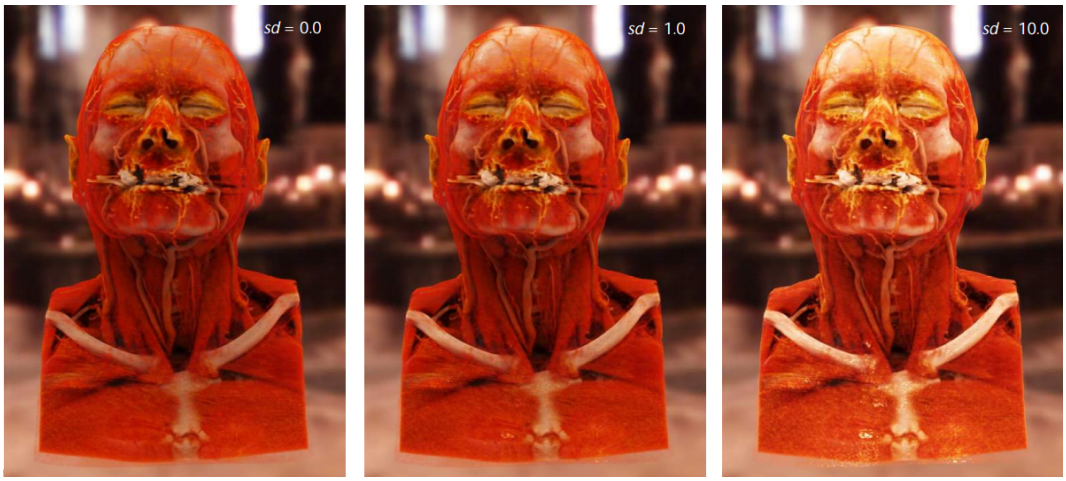


Figura 5.8: Exame médico renderizado utilizando dispersão híbrida com diferentes valores de *sd*. Imagem retirada de [5].

levar essa contribuição em conta ao calcular a radiância final do raio de luz amostrado. Essa contribuição pode ser calculada amostrando direções para apenas um ponto no volume (single scattering) ou para alguns pontos no volume (multiple scattering). Na computação gráfica clássica, a **dispersão única** é responsável pela luz emitida de uma fonte de luz em uma superfície (luz direta) [1].

Para renderização volumétrica, é útil considerar o caso limitado de dispersão onde a radiância dispersada leva em consideração apenas a iluminação direta para um ponto no volume (daí o termo dispersão única: a luz se dispersa apenas uma vez no caminho). Essa técnica também pode ser encontrada na literatura com o nome de **estimativa de próximo evento (next event estimation)** [49].

Dispersão única é uma aproximação útil para volumes mais escuros que têm baixo albedo, ou mais opacos, onde efeitos de dispersão múltipla de ordem superior são de menor importância [10]. Revisitando a Equação (5-5), temos o termo que corresponde à dispersão de entrada, apresentado a seguir:

$$L(x,\omega) = \sigma_s(x) \int_{S^2} f_p(x,\omega,\omega')L(x,\omega)d\omega' \tag{5-19}$$

Esse termo pode ser calculado fazendo uma amostragem da função de fase ou BRDF e também da fonte de luz. Como são duas amostragens, é necessário combiná-las usando a teoria da múltipla amostragem por importância apresentada na Seção 3.3, assim como é feito na estimativa de luz direta para superfícies.

Já na **dispersão múltipla** vários pontos são amostrados dentro do volume. A cada ponto amostrado, é feita uma estimativa da luz direta para o ponto juntamente com a função de fase, assim como ocorre para o ponto da

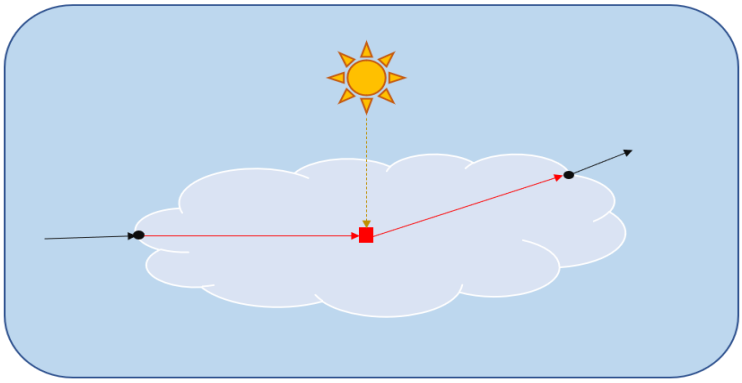


Figura 5.9: Esquema de dispersão única. Dentro do volume um ponto de dispersão é amostrado e é feito o cálculo da luz direta para esse ponto.

dispersão única. Nesse tipo de dispersão, o raio de luz continua ricocheteando dentro do próprio volume com a direção amostrada pela função de fase. Assim, a iluminação indireta pode ser calculada dentro do próprio volume, como mostrado na Figura 5.10.

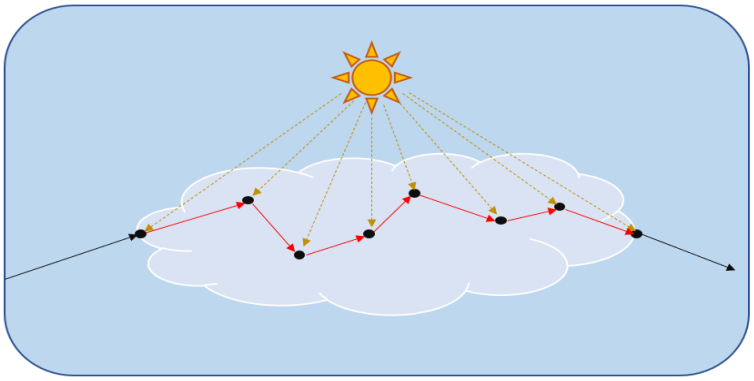


Figura 5.10: Esquema de dispersão múltipla. Dentro do volume alguns pontos de dispersão são amostrados e é feito o cálculo da luz direta para esses pontos.

6
Renderizador com traçado de caminhos

Esta dissertação teve como resultado a implementação de um framework que renderiza superfícies e volumes usando o algoritmo de traçado de caminhos. O trabalho foi implementado na CPU, utilizando a linguagem C++ e o framework Qt. O programa foi desenvolvido de modo que pudesse ser estendido com a adição de mais tipos de BRDFs, materiais, volumes etc. Um diagrama de classes reduzido é apresentado na Figura 6.1.

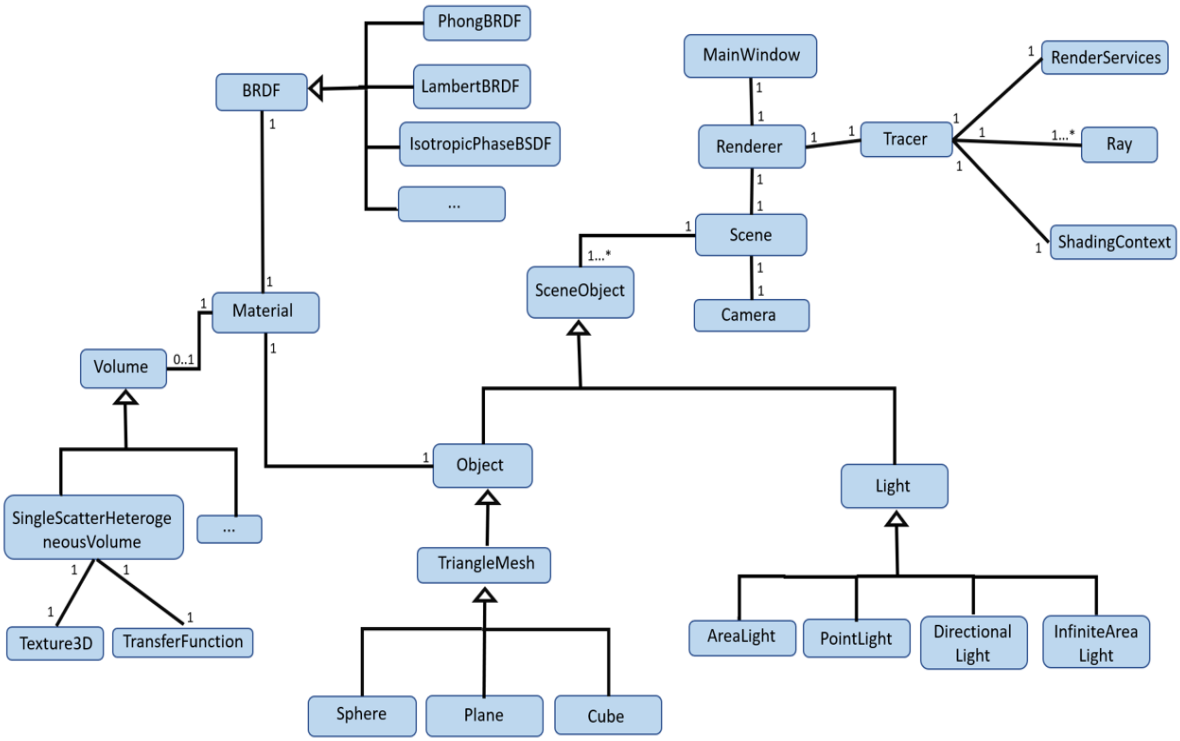


Figura 6.1: Diagrama de classes do software de renderização.

6.1
Algoritmo de traçado de raios

O algoritmo de traçado de caminhos é uma variação do algoritmo de traçado de raios. No entanto, diferentemente do algoritmo clássico de traçado de raios, precisamos ser capazes de traçar vários raios para um mesmo pixel. Portanto, é necessário implementar um laço que passe por todos os pixels da tela e um outro laço interno para que mais de um raio seja traçado para um mesmo pixel.

No Algoritmo 4, é apresentada uma implementação do traçado de raios que recebe um número de raios e as dimensões da tela. Cada raio traçado é

definido por uma origem e uma direção. A origem será sempre a posição p_{eye} do olho, mas é preciso calcular a direção d que vai do olho até o pixel da posição (x, y).

Com esses dados, é possível começar a integração da luz que será aprofundada na seguinte seção. O somatório de todas as amostras de cores para o mesmo pixel deve ser feito. Para definir a cor L resultante, é feita uma média, dividindo esse valor L pelo número de raios por pixel.

É importante ressaltar que cada raio traçado atinge um ponto diferente do pixel para evitar serrilhados (aliasing) na renderização. Apesar de não ser a melhor estratégia, usamos uma amostragem uniforme de pontos dentro do pixel.

Algoritmo 4: Traçado de raios

```
1  Função Tracing(p_eye, w, h, n):
2      nRaios ← n
3      L ← (0, 0, 0)
4      para cada pixel (x, y) na tela de dimensões (w, h) faça
5          para cada raio faça
6              d ← directionFromEyeToPixel()
7              ray ← Ray(p_eye, d)
8              L ← L + LightIntegration(ray)
9          fim
10         L ← L / nRaios
11         image.setPixels(x, y, L)
12     fim
13 fim
14
```

A câmera possui parâmetros que a definem, como a sua posição p_{eye}, a posição para onde ela está olhando *center* e o vetor *up* que define sua rotação para cima . Ambos estão representados na Figura 6.2. O sistema de coordenadas (x_e, y_e, z_e) da câmera pode ser calculado usando esses parâmetros (vide Equação (6-1), Equação (6-2) e Equação (6-3)) (ver em [55]).

$$z_e = \frac{1}{||p_{eye} - center||}(p_{eye} - center) \tag{6-1}$$

$$x_e = \frac{1}{||up \times z_e||}(up \times z_e) \tag{6-2}$$

$$y_e = x_e \times z_e \tag{6-3}$$

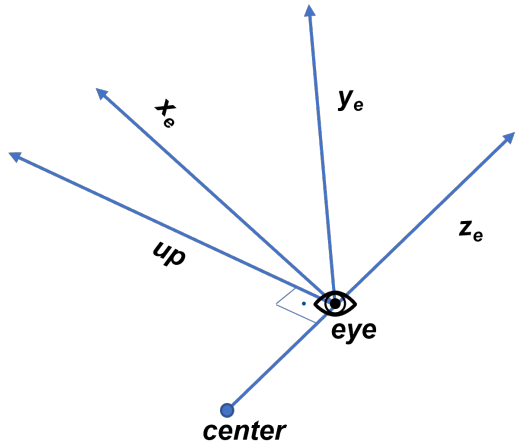


Figura 6.2: Esquema de funcionamento do algoritmo de traçado de caminhos para superfícies e volumes.

A função $directionFromEyeToPixel()$, calcula d com base no sistema de coordenadas da câmera. Pela Figura 6.3 (a), podemos tirar a Equação (6-4). Dessa equação, se isolarmos o d, obtemos a Equação (6-5).

$$p(t) = p_{eye} - td \tag{6-4}$$

$$d = p_{xy} - p_{eye} \tag{6-5}$$

É possível chegar à Equação (6-6), também olhando para a Figura 6.3(a). Nessa equação, o_1 é a origem do sistema de eixos do plano de projeção uv, que é alinhado com a imagem. Os vetores unitários $\hat{u}$ e $\hat{v}$ correspondem aos unitários $\hat{x_e}$ e $\hat{y_e}$, respectivamente. Assim, a posição o_i da origem pode ser determinada pela Equação (6-7). Nessa equação, f é a distância da câmera para o plano $near$.

$$p_{xy} = o_1 + u(x)\hat{u} + v(y)\hat{v} \tag{6-6}$$

$$o_1 = p_{eye} - f\hat{z_e} - \frac{a}{2}\hat{y_e} - \frac{b}{2}\hat{x_e} \tag{6-7}$$

As funções $u(x)$ e $v(y)$ são escalas para acomodar as coordenadas da imagem (x,y) dadas em pixels nas coordenadas da janela. Como ilustra a Figura 6.3(b), estas escalas podem ser calculadas como mostrado na Equação (6-8a) e na Equação (6-8b).

$$u(x) = \frac{x}{w}b \tag{6-8a}$$

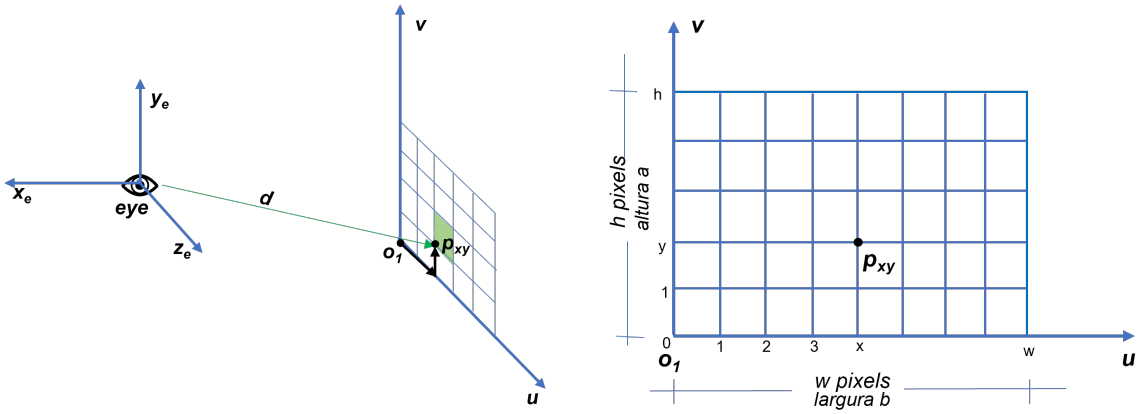
$$v(y) = \frac{y}{h}a \tag{6-8b}$$

Finalmente, podemos chegar à Equação (6-9a) usando o resultado das equações (6-8), (6-6) e (6-5). Substituindo a expressão de o_1 da Equação (6-7)

nessa equação, chegamos à Equação (6-9b), que é a formulação final para d.

$$d = \left(o_1 + b\frac{x}{w}\hat{x_e} + a\frac{y}{h}\hat{y_e}\right) - p_{eye} \tag{6-9a}$$

$$d = -f\hat{z_e} + a\left(\frac{y}{h} - 0.5\right)\hat{y_e} + b\left(\frac{x}{w} - 0.5\right)\hat{x_e} \tag{6-9b}$$



(a) Esquema de raio lançado da câmera (p_{eye}) para um pixel p_{xy} na tela.

(b) Representação de um pixel p_{xy} na tela com altura a com h pixels e largura b com w pixels.

Figura 6.3: Lançamento de raios por uma câmera.

6.2 Renderização de superfícies

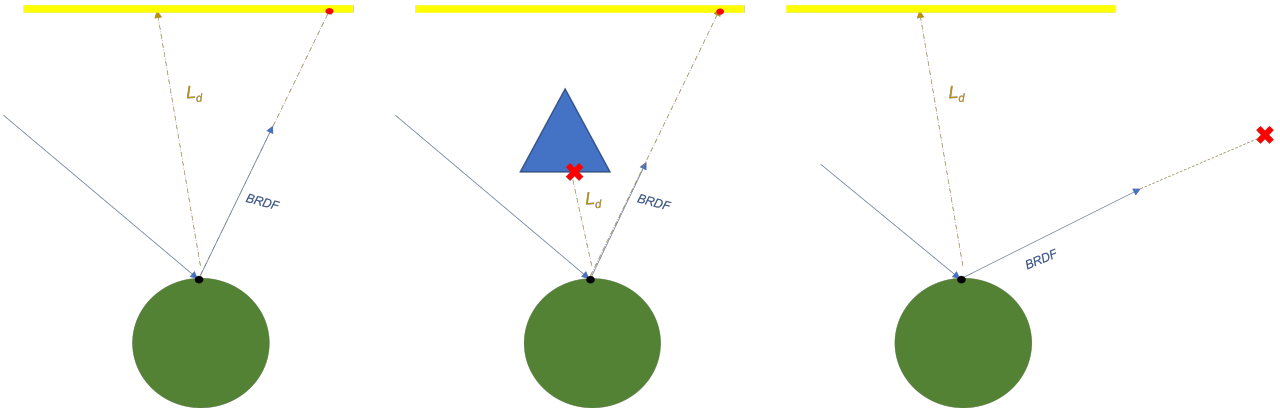
Assim que um raio é traçado da câmera para o pixel uma busca é feita para achar o objeto mais próximo que esse raio intersecta. Assim que o objeto é encontrado, a integração do raio de luz é iniciada. A integração começa com o cálculo da luz direta para o ponto atingido.

Sempre que um raio atinge um objeto, calculamos uma estimativa para a luz direta (next event estimation). Esse cálculo é feito utilizando a amostragem por importância múltipla (MIS) que, como já mencionado na Seção 3.4, faz mais de uma amostragem e as une. Nesse caso, é feita uma amostragem da direção utilizando a BRDF no pixel e uma amostragem de uma direção indo do pixel para a fonte de luz resultando na radiância L_d (Figura 6.4(a)). Essas duas amostragens são unidas utilizando a fórmula de poder heurístico (Equação (3-10)), já mencionada na Seção 3.4. Assim, temos uma amostragem da luz direta para aquele ponto, como mostrado no Algoritmo 5.

É importante ressaltar que situações como a da Figura 6.4 (b) podem ocorrer. Na amostragem da fonte de luz para cálculo da luz direta, o raio amostrado pode, no caminho do ponto pra fonte de luz, se chocar com outro objeto. Com isso, a contribuição L_d da fonte de luz para esse ponto deve ser zero. Por isso, é importante sempre fazer o teste de visibilidadde para ver se

os raios amostrados atingem ou não outro objeto antes de atingir a fonte de luz.

Além disso, na amostragem da BRDF, por exemplo, é possível que a direção amostrada não atinja a fonte de luz da cena, resultando em uma radiância zero para essa amostra (Figura 6.4 (c)). O mesmo pode acontecer para uma amostra da fonte de luz que não tem nenhuma possibilidade de ser amostrada pela BRDF do material.



(a) Amostragem da luz direta e da BRDF.

(b) Amostragem da luz direta e amostragem da BRDF, com teste de visibilidade falhando.

(c) Amostragem da luz direta e amostragem da BRDF falhando por não atingir fonte de luz.

Figura 6.4: Estimativa da luz direta

Voltando ao algoritmo de traçado de caminhos para superfícies, precisamos ainda calcular e guardar a parcela de luz indireta que é um dos detalhes mais importantes quando se fala de traçado de caminhos. A luz indireta é obtida, calculando uma estimativa para a integral da equação de renderização (4-1) utilizando Monte Carlo. Uma variável β é utilizada para armazenar a luz resultante do ricocheteamento do raio de luz nos objetos.

Na equação de renderização (4-1) temos dentro da integral a multiplicação do resultado da BRDF no ponto x para uma direção de entrada ω_i e uma direção de saída ω_o multiplicado pelo produto interno entre a luz de entrada e normal no ponto, multiplicado pela radiância de entrada desse raio. Esse integrando parece naturalmente recursivo, porém pode ser implementado iterativamente sempre amostrando a BRDF do ponto x e acumulando o resultado dos ricocheteamentos do raio na variável β. Assim, essa variável vai ser resultado de um produtório das contribuições de cada ponto.

Um pseudocódigo da integração do raio de luz é mostrado no Algoritmo 5. Ele consiste em um laço que calcula a radiância resultante do ricocheteamento do raio traçado para um pixel que atinge um ponto x inicialmente. Um

pseudocódigo do cálculo da luz direta para um ponto também é apresentado no Algoritmo 6.

Algoritmo 5: Integração do raio de luz na cena

```
1  Função LightIntegration(ray):
2      L ← (0, 0, 0)
3      β ← (1, 1, 1)
4      se !getNearestHit(obj, ray, P) então
5          pare
6      fim
7      para cada ricocheteamento do raio faça
8          m ← obj.getMaterial()
9          bsdf ← m.getBSDF()
10         L ← L + β * directLight()
11         bsdf.generateSample(sampleDirection, Ls, pdf)
12         β ← β * (Ls/pdf)
13         nextRay.origin ← P
14         nextRay.direction ← sampleDirection
15         se !getNearestHit(obj, nextRay, P) então
16             pare
17         fim
18         ray ← nextRay
19     fim
20     retorna L
21 fim
```

Algoritmo 6: Luz direta

Função `directLight()`:
  $L_d \leftarrow (0, 0, 0)$
  para *cada fonte de Luz* **faça**
    $generateLightSample(w_i, light, directLight, pdfLight)$
    $bsdf.evaluateSample(w_i, brdfLight, pdfbrdf)$
    $misLight = powerHeuristic(pdfLight, pdfbrdf)$;
    $L_d \leftarrow L_d + \frac{directLight * brdfLight * misLight}{pdfLight}$
    $bsdf.generateSample(sampleDirection, L_s, pdf)$
    $evaluateLightSample(light, sampleDirection, resultLight, pdfl)$
    $misBRDF = powerHeuristic(pdf, pdfl)$;
    $L_d \leftarrow L_d + \frac{L_s * resultLight * misBRDF}{pdf}$
  fim
  retorna L_d
fim

6.3 Renderização de Volumes

Com a integração da luz para superfícies já implementada, podemos adicionar volumes na mesma cena e fazer a diferenciação dos dois casos. No caso dos volumes, é necessário que se tenha uma função de integração volumétrica que é chamada sempre que o raio entra em um volume.

Os volumes podem ser integrados ao renderizador de superfícies considerando eles como sendo parte de um material. Assim, um cubo pode ser criado com as dimensões do volume e um "material de volume" pode ser adicionado a esse cubo. Isso funciona, visto que a maioria dos volumes utilizados neste trabalho são malhas estruturadas no formato de hexaedro. Porém, com essa estrutura, é possível adicionar volume a qualquer geometria.

Neste trabalho usamos, em sua maioria, volumes heterogêneos com dispersão única. Portanto, como já mencionado na Seção 5.4, o método escolhido para resolver a integral volumétrica foi o método de rastreamento (tracking). Nesse método, é necessário que se defina uma distância que o raio de luz percorrerá no volume até uma colisão com uma partícula. Na Seção 5.6.1 foram apresentados alguns métodos para se definir essa distância, porém o rastreamento delta foi o escolhido para ser implementado por ser o mais comumente usado para volumes heterogêneos e não ter viés nem ser tão custoso como a marcha em raio.

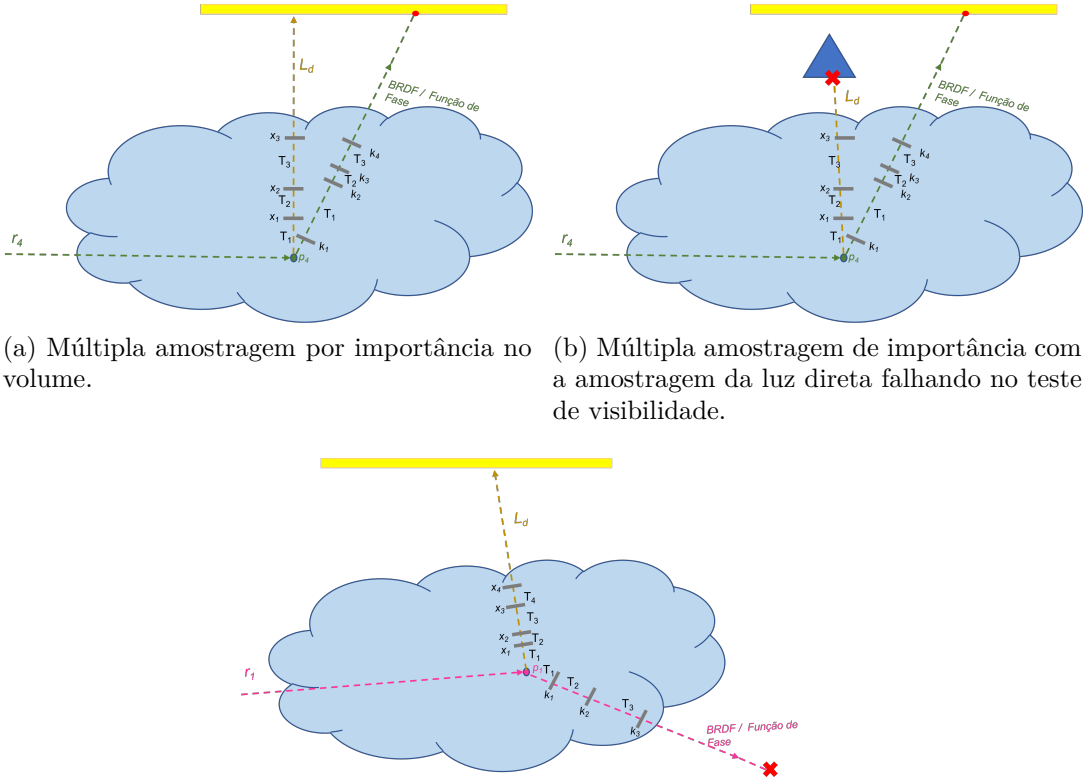
Para a implementação do laço do rastreamento delta, no cálculo da distância t a cada passo, foi utilizado um número randômico e o coeficiente máximo de extinção do volume em questão. Em seguida, é preciso analisar o ponto em que o raio alcança ao andar por essa distância. A roleta russa é utilizada para escolher se o raio para ou não em um ponto. Para escolher, utilizamos a divisão entre o próprio coeficiente de extinção do ponto e o $\bar{\sigma}$, como previsto pelo Algoritmo 2.

Se o raio cair em um ponto fora do volume, consideramos que ele passou por um vácuo em que não houve eventos de dispersão nem absorção. No entanto, se o ponto amostrado ainda está dentro do volume temos duas possibilidades. Se o número randômico for menor do que a divisão, consideramos que uma colisão de dispersão aconteceu e o algoritmo de rastreamento delta é finalizado. Se não for menor, consideramos que um evento nulo ocorreu e o laço continua em busca de um novo ponto para analisar e aplicar a roleta russa. Por questões de simplificação, não consideramos eventos de emissão.

Nesse algoritmo, é possível perceber que um ponto que tenha um coeficiente de extinção mais alto tem uma probabilidade maior de ser escolhido na roleta russa como um ponto de dispersão. O coeficiente de extinção pode ser considerado como a opacidade do ponto, logo pontos com uma opacidade maior tem mais chance de serem escolhidos.

Para cálculo do resultado da integração do volume, utilizamos esse único ponto amostrado, já que estamos lidando com dispersão única. Se o ponto amostrado está dentro do volume, assumimos que houve um evento de dispersão ali. Calculamos a luz direta para esse ponto utilizando múltipla amostragem por importância. Assim como para superfícies, uma amostragem da fonte de luz e da BRDF ou função de fase são feitas. A Figura 6.5 ilustra as duas amostragens. Na Figura 6.5(a), as duas amostragens são feitas e contabilizadas. Já a Figura 6.5(c) ilustra o caso em que a amostragem da BRDF não é aproveitada, pois a direção amostrada não atinge a fonte de luz. Essa imagem mostra a importância da múltipla amostragem por importância, pois apesar da amostragem da BRDF não impactar na radiância nesse caso, a amostragem L_d da fonte de luz impacta. Além disso, é importante ressaltar que, assim como para superfícies, é importante fazer o teste de visibilidade,pois casos como o da Figura 6.5(b) também podem ocorrer.

É importante ressaltar que mesmo implementando a dispersão única, ainda temos uma amostragem de vários pontos na imagem final, visto que diferentes pontos podem ser amostrados para diferentes raios que entram no volume. A Figura 6.6 ilustra como as amostras de um ponto do volume podem ser bem diferentes dependendo do raio. Isso faz com que o volume todo



(a) Múltipla amostragem por importância no volume.

(b) Múltipla amostragem de importância com a amostragem da luz direta falhando no teste de visibilidade.

(c) Múltipla amostragem de importância com amostragem da BRDF não intersectando a fonte de luz.

Figura 6.5: Opções de interação de raios com superfícies que contém volume.

seja contemplado. Pontos com maior opacidade têm maior probabilidade de serem amostrados; porém, é possível que qualquer ponto dentro do volume seja amostrado.

No entanto, precisamos calcular o que chamamos de feixe de transmitância, para que possamos calcular corretamente a sombra, causada por outras partes do próprio volume, no ponto do volume amostrado. Para isso, utilizamos o método de rastreamento de proporção (ratio tracking) já mencionado na Seção 5.6.2.

Nesse estimador de transmitância, traçamos um raio do ponto dentro do volume até a fronteira do volume, na direção amostrada. Uma distância t_0 é amostrada (Equação (5-9)). Calculamos a transmitância do ponto do volume inicial até um segundo ponto distante t_0 na direção amostrada. Fazemos isso consecutivamente, até a fronteira do volume, como no Algoritmo 3. Como mencionado na Seção 5.5, a transmitância tem a propriedade multiplicativa (5-6), em que a transmitância total é igual ao produtório das transmitâncias do caminho. Assim, calculamos o feixe de transmitância utilizando essa propriedade e multiplicando as trasmitâncias de cada trecho. Na Figura 6.5, são ilustradas as transmitâncias T_i que devem ser calculadas e adicionadas ao feixe

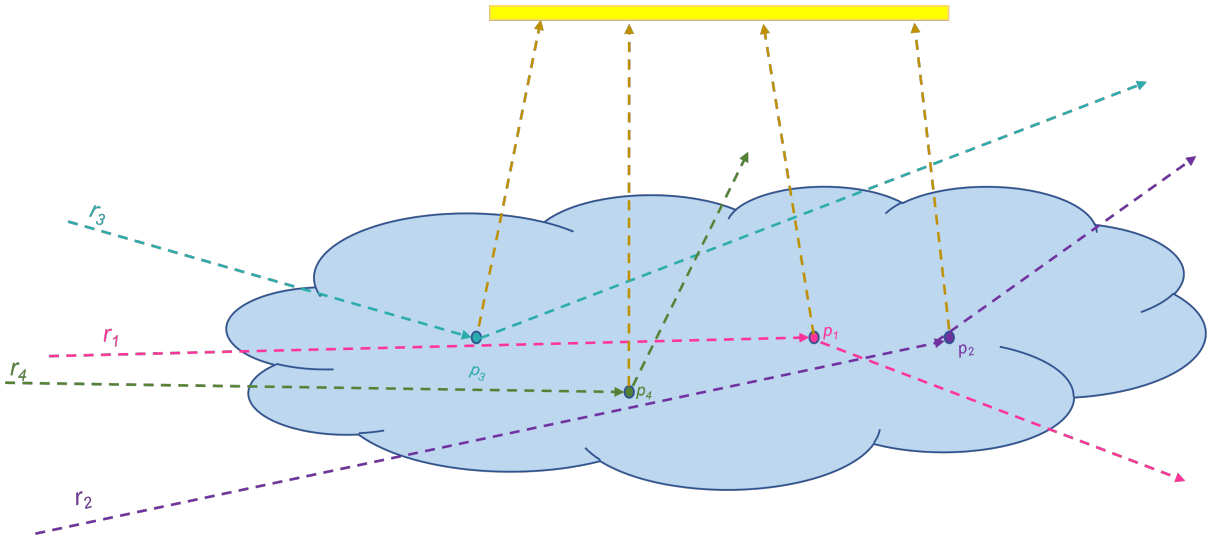


Figura 6.6: Esquema que mostra os diferentes raios que atingem o volume. Para cada raio uma amostragem é feita e pontos diferentes podem ser amostrados dentro do volume.

de transmitância no algoritmo de rastreamento de proporção.

A radiância final da integração volumétrica é multiplicada pelo β, e também por um peso que é calculado usando o coeficiente de extinção do ponto amostrado. Esse peso é igual a 1 quando lidamos com superfícies, pois a chance de calcularmos uma dispersão naquele ponto da superfície para um dado raio é igual a 1, enquanto que todos os outros pontos tem probabilidade zero de ocorrerem. No caso do volume é diferente, pois há chance de calcularmos a dispersão em todos os pontos do raio em que há volume. Por isso, devemos atribuir um peso à nossa amostragem de ponto no volume. Esse peso pode ser calculado como sendo o inverso da extinção do ponto.

Finalmente, assim como no caso da superfície, devemos atualizar a variável β que guarda as informações dos ricocheteamentos do raio. Se o raio passou pelo volume sem atingir nenhum ponto de dispersão, a variável não é alterada, visto que é como se o raio tivesse ultrapassado um vácuo. No entanto, se houve dispersão no volume, essa variável precisa ser multiplicada pela radiância dada pela amostragem da BRDF ou função de fase, dividida pela sua respectiva pdf, assim como ocorre com superfícies.

Nesta dissertação, usamos tanto BRDFs quanto Funções de fase para volumes, dependendo de qual o nosso objetivo final. Se o objetivo é deixar o volume com um material específico, como um metal, plástico etc, usamos a BRDF, porém ela é indicada para volumes que tenham maior opacidade. Para volumes que possuem transparência (como nuvens, água, fluidos), optamos por funções de fase.

Na implementação da dispersão híbrida, antes de fazermos a amostragem

da fonte de luz e da BRDF/Função de fase, calculamos a probabilidade da Equação (5-17) e sorteamos um número para decidir se usaremos uma BRDF ou uma Função de fase nos nossos cálculos de radiância do volume.

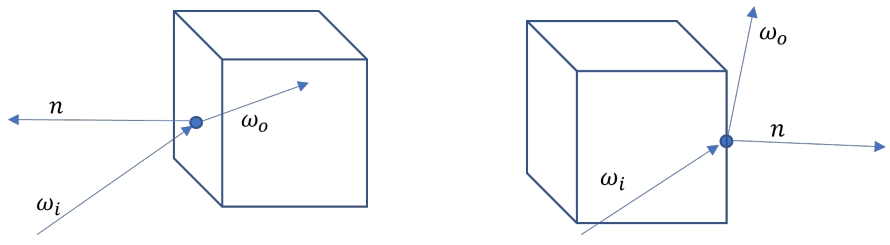
6.4 Renderização conjunta (Superfícies + Volumes)

Quando juntamos as duas renderizações, o ideal é separar a integração da luz da integração volumétrica, visto que as duas tem suas próprias complexidades e são independentes. A integração volumétrica pode ser encapsulada como uma função do objeto volume, como já foi dito.

Em cada laço da integral da luz, checamos se o objeto intersectado pelo raio de luz contém ou não um volume. Se ele contém, é preciso investigar se o raio de luz entrará ou sairá do volume e se um evento que ocorre ali é um evento de transmissão ou de reflexão. Para isso, é calculado o produto interno entre a normal desse objeto e a direção de entrada do raio de luz e o produto interno entre a normal e a direção de saída amostrada. Se esses dois produtos tiverem sinais distintos, quer dizer que houve uma transmissão e se o segundo produto interno mencionado for menor do que zero, significa que o raio está entrando no volume, caso contrário ele está saindo.

A Figura 6.7 ilustra as possibilidades de interação do raio com o cubo que contém volume. Na Figura 6.7 (a), o produto interno entre a normal e ω_i é positivo, pois o ângulo entre esses vetores é agudo, logo, $cos(\theta_i) > 0$. Nessa mesma figura, o ângulo entre a normal e ω_o é obtuso ($cos(\theta_o) < 0$), o que indica que o raio está entrando no volume. Além disso, como os dois produtos internos têm sinais distintos há uma transmissão, como já dito anteriormente.

Já na Figura 6.7 (b), o ângulo θ_i é obtuso e θ_o é agudo. Portanto, como os dois produtos internos têm sinais distintos, houve um evento de transmissão e não de reflexão, como é mostrado na figura. Porém, como θ_o é agudo, sabemos que o raio está saindo do volume.



(a) Entrada no volume (b) Saída do volume

Figura 6.7: Opções de interação de raios com superfícies que contém volume.

São situações que são importantes de serem previstas, pois vão determinar o que precisamos calcular em seguida. Caso haja transmissão e o raio esteja entrando no volume, o algoritmo calcula a integração do volume e acumula na variável de radiância. Caso o raio esteja saindo do volume, o raio continua até atingir novamente algum outro objeto. A Figura 6.8, mostra um esquema geral de como o algoritmo funciona, numa cena que tenha superfícies e volumes.

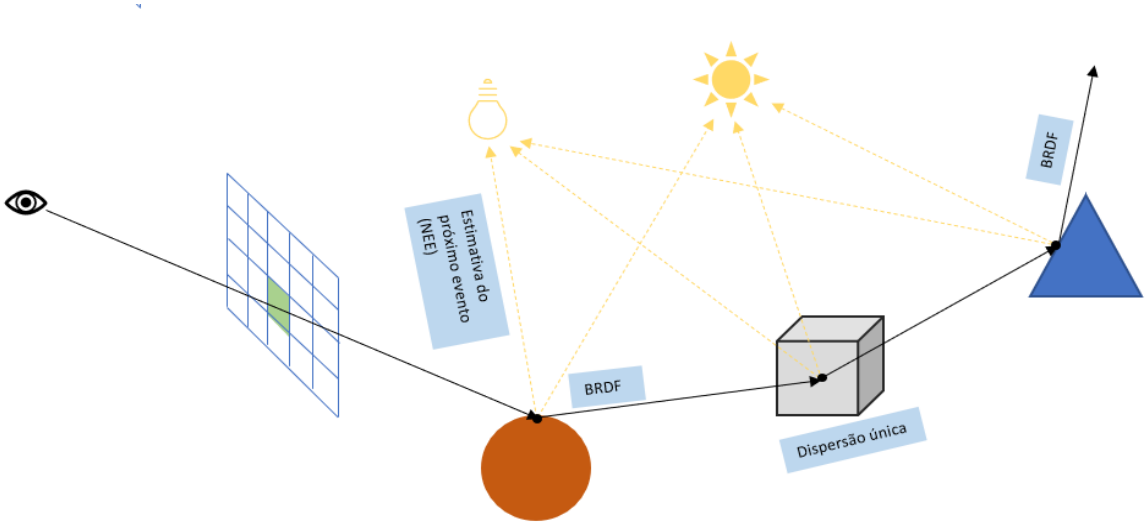


Figura 6.8: Esquema de funcionamento do algoritmo de traçado de caminhos para superfícies e volumes.

7
Resultados

Alguns testes foram feitos para demonstrar algumas funcionalidades do programa implementado neste trabalho. Além disso, testes foram realizados para ilustrar o papel desempenhado por alguns elementos chaves que aparecem na abordagem teórica discutida nos capítulos anteriores.

Na Figura 7.1, temos uma sala renderizada com diferentes números máximos de richocheteamentos. Quando aumentamos o número máximo de ricocheteamentos, os raios de luz contribuem para a radiância final de mais pixels da cena e é possível ver que a luminosidade de cada ponto parece aumentar.

A cena é a mesma, com a mesma fonte de luz, porém, um raio que é lançado para um pixel na figura mais à esquerda só poderá acumular contribuições de até 3 ricocheteamentos, enquanto que no caso da figura mais à direita, um raio poderá acumular contribuição de até 7 ricocheteamentos.

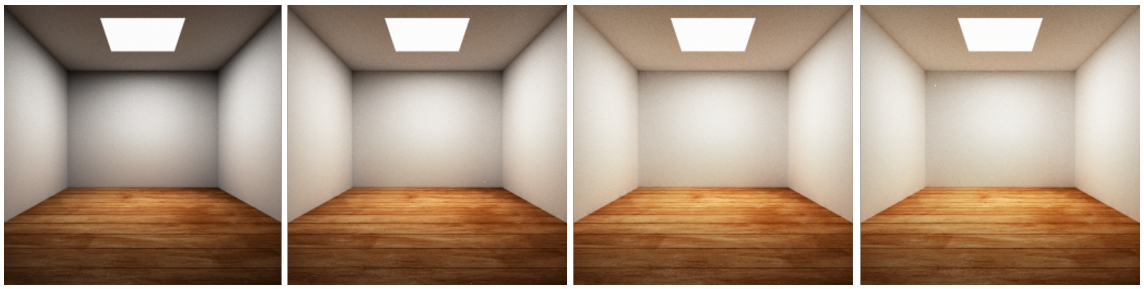


Figura 7.1: Da esquerda pra direita temos como números máximos de ricocheteamentos: 2, 3, 5 e 7. Todas as figuras foram renderizadas com 1000 raios por pixel. Cada tela foi renderizada com um tamanho de 256x256 pixels.

Na Figura 7.2, temos uma comparação entre um mesmo volume, renderizado utilizando a função de fase de Henyey-Greenstein [54] com diferentes valores de g. Na figura, uma luz de área foi posicionada atrás do volume. Como já dito na Seção 5.7, o g, chamado de parâmetro de assimetria, é responsável por modelar uma dispersão regressiva ($g < 0$), dispersão isotrópica ($g = 0$) ou dispersão progressiva ($g > 0$).

No caso da figura mais à esquerda o g é negativo e portanto, a luz que vem de trás do volume é dispersada para trás, ocorrendo uma dispersão regressiva. Já na figura do meio o g é zero e portanto, a luz que vem de trás é dispersada igualmente para frente e para trás do volume, por causa da dispersão isotrópica. Na figura mais à direita a função de fase utilizada possui g positivo, então a

maior parte da luz emitida pela fonte de luz atrás do volume o atravessa, iluminando mais sua parte da frente.

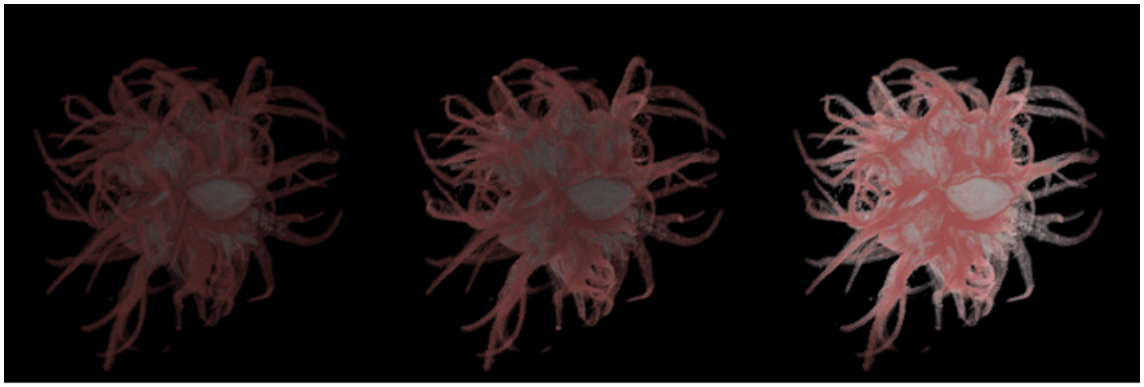


Figura 7.2: O Dataset Hazelnut é renderizado usando a função de fase de Henyey-Greenstein com $g = -0.9$ (esquerda), $g = 0$ (meio) e $g = 0.9$ (direita). Todas as figuras foram renderizadas com 300 raios por pixel. Cada tela foi renderizada com um tamanho de 256x256 pixels.

Na Figura 7.3, é feito um teste utilizando dispersão híbrida. Como mencionado na Seção 5.7, a dispersão híbrida calcula uma probabilidade para decidir entre uma função de fase ou uma BRDF para ser utilizada no cálculo da luz direta em algum ponto no volume. A probabilidade de uma BRDF ser calculada segue a Equação (5-16). Uma roleta russa também é utilizada para decidir qual função usar.

As imagens mais à esquerda do pé e do joelho foram renderizadas com $g = 0$, ou seja, usam apenas a função de fase. Dessa forma, dão mais destaque ao que envolve o osso e menos destaque ao osso. Já nas imagens mais à direita, $g = 1$, o que significa que apenas o BRDF é utilizado para renderizar o volume. O BRDF escolhido para o volume foi a mistura do de Lambert com o de Phong. É possível perceber que o osso se destaca mais do que as imagens da esquerda que utilizam função de fase. Isso acontece porque o BRDF reflete a luz que chega ao osso tornando-o mais brilhoso. As imagens do meio variam entre usar a função de fase e a BRDF. Essa figura também junta volume e superfícies (chão e parede). A fonte de luz utilizada é a fonte de área infinita e uma sombra suave causada pelo volume na superfície pode ser percebida.

Na Figura 7.4, é feita uma comparação entre volumes homogêneos em forma de esfera e uma superfície em forma de esfera. Os volumes (lado esquerdo) possuem diferentes σ_t, ou seja, diferentes opacidades. Na imagem mais à esquerda, o volume que preenche a esfera possui $\sigma_t = 0.5$, tendo uma maior transparência quando comparada à superfície do lado direito. Na imagem do meio $\sigma_t = 0.8$, tendo uma opacidade maior. Quando atribuímos $\sigma_t = 1$ ao

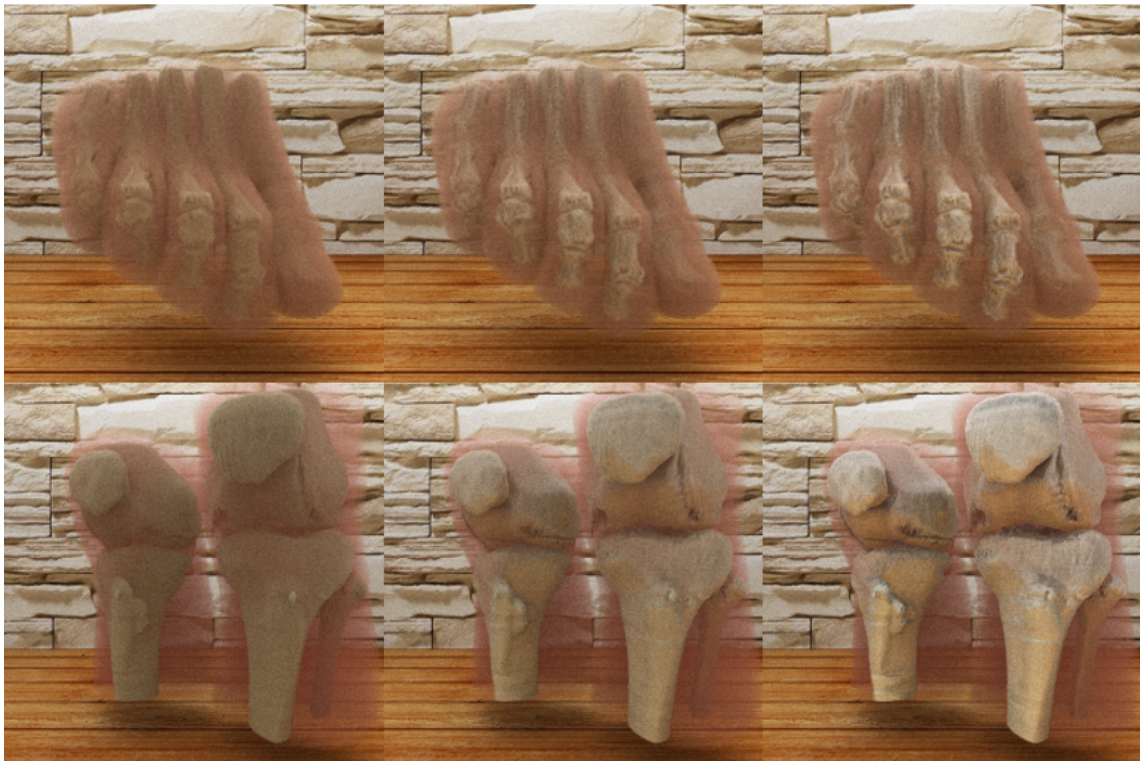


Figura 7.3: Dataset CT-Foot e CT-Knee são renderizados utilizando diferentes valores de g na Equação (5-16) de dispersão híbrida. Na esquerda usou-se $g = 0$, ou seja, apenas a função de fase é utilizada. No meio, $g = 0.3$ usa em alguns momentos a função de fase e em outros a BRDF difusa unida com a de Phong. Na direita, $g = 1$, ou seja, apenas a BRDF é utilizada. Todas as figuras foram renderizadas com 300 raios por pixel. Cada tela foi renderizada com um tamanho de 256x256 pixels.

volume, como na imagem mais à direita, ele se comporta como uma esfera totalmente densa e se torna visualmente igual à superfície de esfera.

Na Figura 7.5, são utilizados diferentes BRDFs para um mesmo volume, para demonstrar que é possível atribuir diferentes materiais para volumes, assim como acontece com superfícies. Porém, diferentemente das funções de fase, as BRDFs, muitas vezes, necessitam da normal para amostrar direções. No entanto, o cálculo de normais para volumes não é tão direto como é para as superfícies. Como alternativa, utilizamos no lugar das normais, os gradientes do volume, calculados utilizando o método de diferenças finitas central (ver em [56]).

As imagens mais à esquerda da Figura 7.5, foram renderizadas utilizando a BRDF de Lambert para os volumes, e portanto, simulando um material difuso para eles. Já as imagens do meio foram renderizada usando uma BRDF que mistura as BRDFs de Lambert e de Phong dando um aspecto mais brilhoso ou plástico para os volumes. As imagens mais à direita foram renderizadas utilizando a BRDF de Phong, e portanto, simulando um material especular.

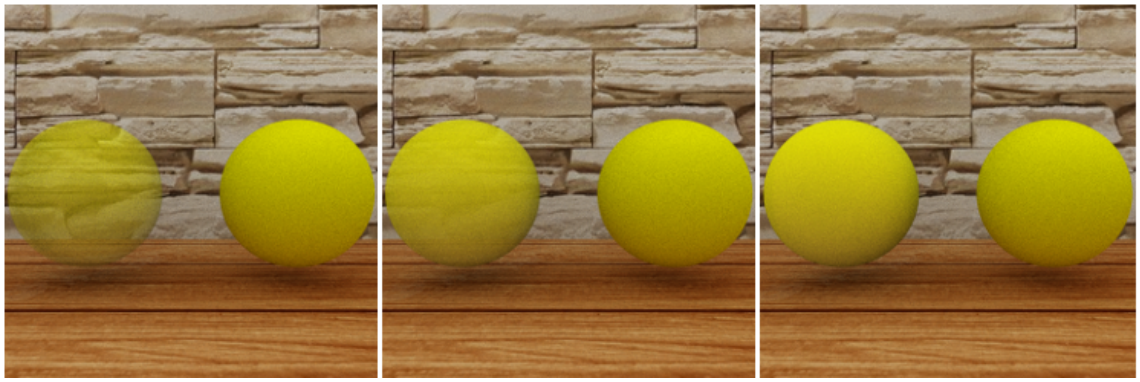


Figura 7.4: Comparação entre volumes de esfera e superfície de esfera. Na figura da esquerda, temos um volume homogêneo com forma de esfera e $\sigma_t = 0.5$ na esquerda e uma superfície ao lado. Na figura do meio, o volume tem $\sigma_t = 0.8$ e também uma superfície ao lado. Na figura da direita, temos um volume com $\sigma_t = 1$ e uma superfície do lado direito. Todas as figuras foram renderizadas com 1000 raios por pixel. Cada tela foi renderizada com um tamanho de 256x256 pixels.

Todas as imagens foram renderizadas em uma cena com superfícies e uma fonte de luz de área infinita, ou seja, uma esfera de luz que engloba a cena. Apesar de a cena ser totalmente iluminada, ainda podemos perceber sombras suaves produzidas pelos volumes na superfície do chão.

Na Figura 7.6, diferentes tipos de fonte de luz iluminam uma cena contendo um volume. O feixe de transmitância mencionado na Seção 6.3 é o termo que faz com que a sombra seja calculada para o volume. Assim como para superfícies, as fontes de luz pontual e direcional produzem no próprio volume, sombras escuras (dark shadows) e fontes de luz de área e de área infinita produzem sombras suaves (soft shadows).

A cena da Figura 7.7, envolve diferentes tipos de dados com diferentes materiais. O chão, a parede e o teto são superfícies com materiais difusos (BRDF de Lambert). Já o bonsai é um volume heterogêneo que está com um material também difuso. Ao lado do bonsai, temos um espelho, ou seja, uma superfície com material especular (BRDF de Phong). O espelho reflete a imagem do volume, da fonte de luz e das outras superfícies.

Figura 7.5: Datasets Bonsai, VismaleHead e Engine são renderizados usando diferentes BRDFs. Na esquerda a BRDF de Lambert é utilizada, no meio a BRDF misturando a de Lambert e a de Phong é utilizada. Na direita, a BRDF de Phong é utilizada. Todas as figuras foram renderizadas com 300 raios por pixel. Cada tela foi renderizada com um tamanho de 256x256 pixels.

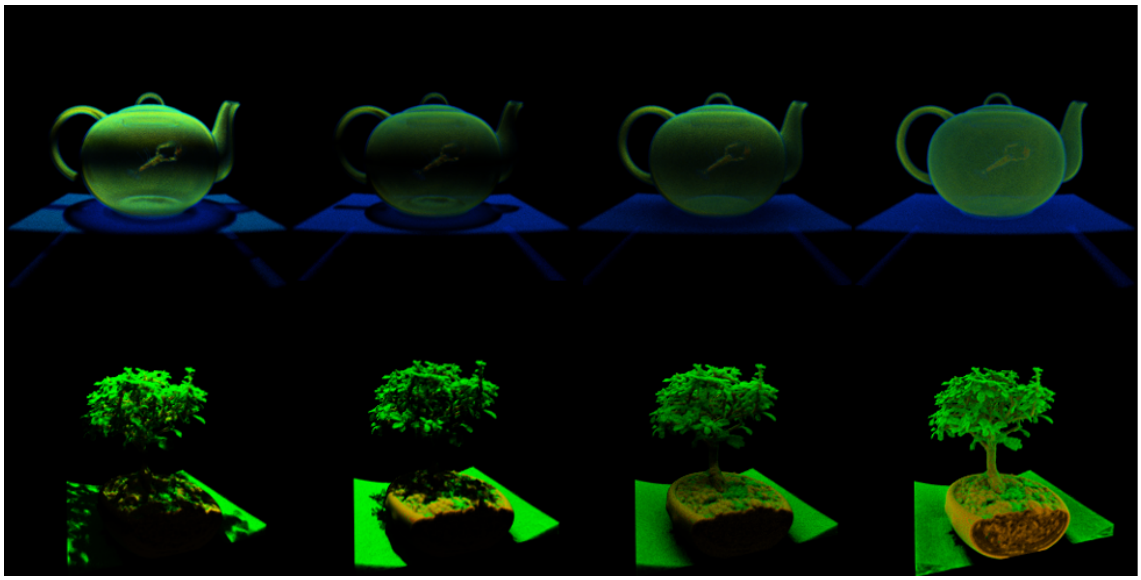


Figura 7.6: Volumes dos Datasets Boston Teapot e Bonsai renderizados utilizando diferentes tipos de fontes de luz. Da esquerda para a direita: Fonte pontual, fonte direcional, fonte de área e fonte de área infinita. Todas as figuras foram renderizadas com 300 raios por pixel. Cada tela foi renderizada com um tamanho de 256x256 pixels.

Figura 7.7: Volume de Bonsai em sala com espelho e fonte de luz de área. A imagem foi renderizada com 300 raios por pixel e com um tamanho de 256x256 pixels.

8
Conclusão e Trabalhos futuros

Um extenso estudo sobre o algoritmo de traçado de caminhos foi feito ao longo de todo este trabalho. Alguns métodos e conceitos matemáticos foram expostos para ajudar na compreensão do algoritmo de traçado de caminhos, tanto para superfícies quanto para volumes. Além disso, foi feita uma implementação do algoritmo de traçado de caminhos que junta em uma mesma cena dados volumétricos e superfícies.

Esta dissertação tem como uma das suas contribuições mais relevantes a explicação das técnicas e conceitos utilizados na renderização de superfícies e volumes e os devidos cuidados que se deve ter ao aplicá-los. No entanto, uma contribuição mais importante ainda se dá no detalhamento da interação desses dois tipos de renderização, dando a mesma importância para ambos, o que raramente acontece em outros trabalhos e que é muito importante quando se deseja ter um renderizador que abrange simultaneamente esses dois tipos de dado em uma mesma cena.

Além da parte teórica detalhada, o trabalho contém uma extensa explicação de questões de implementação que não são normalmente tratadas em outros trabalhos. Para isso, apresentamos alguns pseudocódigos para cada etapa e chamamos a atenção para situações que devem ser previstas pelo algoritmo, como, por exemplo, o teste de visibilidade.

No framework de traçado de caminhos, os objetos presentes nas cenas devem possuir algum material que segue uma BRDF. No mundo real os materiais podem ser de infindáveis tipos. O ideal seria ter um programa em que todos eles fossem abrangidos, porém neste trabalho foi possível incluir apenas alguns tipos de BRDFs para construção dos materiais. Como trabalho futuro, seria importante agregar mais BRDFs ao programa, como a BRDF que segue o modelo de multifacetas, permitindo a adesão de outros materiais e diversificando os objetos nas cenas. Além disso, eventos de emissão não foram considerados na implementação do algoritmo e poderiam ser adicionados futuramente.

Como mencionado anteriormente, renderizações cinematográficas utilizam a dispersão híbrida para oscilar entre dispersão volumétrica e dispersão de superfície. Com isso, além da função de fase, uma BRDF também é utilizada. Esta técnica foi implementada neste trabalho, porém, outras técnicas que contribuiriam muito para o realismo das imagens, como dispersão de subsuperfície, não foram implementadas e futuramente poderiam ser adicionadas.

Além do estudo dos conceitos importantes para o desenvolvimento do algoritmo, o foco do trabalho foi a qualidade da imagem final, que deveria se aproximar da realidade, apesar das limitações de materiais. No entanto, alguns detalhes importantes como o tempo de renderização não foram levados em conta. Com a implementação feita na CPU, se a cena contiver muitos objetos, ou um objeto com uma malha muito refinada, ou o tamanho da imagem gerada for muito grande, o tempo de renderização se torna muito alto. Por isso, uma possível melhoria nesse sentido, seria a utilização de algum tipo de estrutura como a kd-tree para armazenar os objetos da cena e otimizar a busca pelo objeto intersectado. Uma outra solução, seria migrar o algoritmo para a GPU. Nesse caso, um grande problema seria passar todas as informações de objetos e materiais, BRDFs etc para os shaders. Linguagens de shader como HLSL e GLSL não são tão indicadas para lidar com um programa extenso como esse. Em seu Keynote, Clarberg [31] apresenta a linguagem open source que foi desenvolvida especialmente para ser usada nesse tipo de situação chamada de Slang (ver em [57]) e poderia ser utilizada. Essa linguagem é extensível e capaz de criar códigos modulares em shaders. Uma outra opção seria usar o OptiX [58], que é a API de traçado de raios da NVIDIA.

Neste trabalho, ainda é preciso traçar uma quantidade muito alta de raios para que se tenha uma cena com menos ruído. Com a grande quantidade de raios, o tempo de renderização também aumenta. A principal solução para esse problema seria a implementação de um denoiser que diminuiria consideravelmente a quantidade necessária de raios para que se tivesse uma renderização com menos ruídos.

Por último, ainda poderiam ser implementadas melhorias relacionadas à integração de volumes, como a implementação da dispersão múltipla. Essa melhoria permitiria uma renderização com mais qualidade de dados volumétricos semelhantes à efeitos naturais, ou dados com alguma transparência, como participating media. A iluminação de volumes teria uma qualidade superior, considerando iluminação indireta dentro do volume.

9
Referências bibliográficas

[1] JÖNSSON, D. et al. A survey of volumetric illumination techniques for interactive volume rendering. In: WILEY ONLINE LIBRARY. *Computer Graphics Forum*. [S.l.], 2014. v. 33, n. 1, p. 27–51.

[2] PHARR, M.; JAKOB, W.; HUMPHREYS, G. *Physically based rendering: From theory to implementation*. [S.l.]: Morgan Kaufmann, 2016.

[3] VEACH, E.; GUIBAS, L. J. Optimally combining sampling techniques for monte carlo rendering. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 1995. p. 419–428.

[4] NOVÁK, J. et al. Monte carlo methods for volumetric light transport simulation. In: WILEY ONLINE LIBRARY. *Computer Graphics Forum*. [S.l.], 2018. v. 37, n. 2, p. 551–576.

[5] ENGEL, K. Real-time monte-carlo path tracing of medical volume data, gpu technology conference, 4–7 apr 2016. In: *San Jose Convention Center, CA, USA*. [S.l.: s.n.], 2016.

[6] KAJIYA, J. T. The rendering equation. In: *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 1986. p. 143–150.

[7] BURLEY, B. et al. The design and evolution of disney's hyperion renderer. *ACM Transactions on Graphics, Vol. 37, No. 3, Article 33. Publication date: July 2018*, v. 37, n. 33, p. 1–22, 2018.

[8] CHRISTENSEN, P. et al. Renderman: An advanced path-tracing architecture for movie rendering. *ACM Transactions on Graphics (TOG)*, ACM New York, NY, USA, v. 37, n. 3, p. 1–21, 2018.

[9] COOK, R. L.; CARPENTER, L.; CATMULL, E. The reyes image rendering architecture. *ACM SIGGRAPH Computer Graphics*, ACM New York, NY, USA, v. 21, n. 4, p. 95–102, 1987.

[10] FONG, J. et al. Production volume rendering: Siggraph 2017 course. In: *ACM SIGGRAPH 2017 Courses*. [S.l.: s.n.], 2017. p. 1–79.

[11] BANKS, D. C.; BEASON, K. Decoupling illumination from isosurface generation using 4d light transport. *IEEE Transactions on Visualization and Computer Graphics*, IEEE, v. 15, n. 6, p. 1595–1602, 2009.

[12] WHITTED, T. An improved illumination model for shaded display. In: *ACM Siggraph 2005 Courses*. [S.l.: s.n.], 2005. p. 4–es.

[13] COOK, R. L.; TORRANCE, K. E. A reflectance model for computer graphics. *ACM Transactions on Graphics (ToG)*, ACM New York, NY, USA, v. 1, n. 1, p. 7–24, 1982.

[14] GORAL, C. M. et al. Modeling the interaction of light between diffuse surfaces. *ACM SIGGRAPH computer graphics*, ACM New York, NY, USA, v. 18, n. 3, p. 213–222, 1984.

[15] COOK, R. L.; PORTER, T.; CARPENTER, L. Distributed ray tracing. In: *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 1984. p. 137–145.

[16] ARVO, J.; KIRK, D. Particle transport and image synthesis. In: *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 1990. p. 63–66.

[17] SHIRLEY, P. S. *Physically based lighting calculations for computer graphics*. [S.l.]: University of Illinois at Urbana-Champaign, 1991.

[18] VEACH, E. *Robust Monte Carlo methods for light transport simulation*. [S.l.]: Stanford University, 1998.

[19] MAX, N. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, IEEE, v. 1, n. 2, p. 99–108, 1995.

[20] LEVOY, M. Display of surfaces from volume data. *IEEE Computer graphics and Applications*, IEEE, v. 8, n. 3, p. 29–37, 1988.

[21] SCHOTT, M. et al. A directional occlusion shading model for interactive direct volume rendering. In: WILEY ONLINE LIBRARY. *Computer Graphics Forum*. [S.l.], 2009. v. 28, n. 3, p. 855–862.

[22] SUNDÉN, E.; YNNERMAN, A.; ROPINSKI, T. Image plane sweep volume illumination. *IEEE Transactions on Visualization and Computer Graphics*, IEEE, v. 17, n. 12, p. 2125–2134, 2011.

[23] BEHRENS, U.; RATERING, R. Adding shadows to a texture-based volume renderer. In: IEEE. *IEEE symposium on volume visualization (cat. no. 989EX300)*. [S.l.], 1998. p. 39–46.

[24] RITSCHEL, T. Fast gpu-based visibility computation for natural illumination of volume data sets. The Eurographics Association, 2007.

[25] STEGMAIER, S. et al. A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In: IEEE. *Fourth International Workshop on Volume Graphics, 2005.* [S.l.], 2005. p. 187–241.

[26] LI, S.; MUELLER, K. Accelerated, high-quality refraction computations for volume graphics. In: IEEE. *Fourth International Workshop on Volume Graphics, 2005.* [S.l.], 2005. p. 73–229.

[27] RODGMAN, D.; CHEN, M. Refraction in volume graphics. *Graphical Models*, Elsevier, v. 68, n. 5-6, p. 432–450, 2006.

[28] SALAMA, C. R. Gpu-based monte-carlo volume raycasting. In: IEEE. *15th Pacific Conference on Computer Graphics and Applications (PG'07)*. [S.l.], 2007. p. 411–414.

[29] KROES, T.; POST, F. H.; BOTHA, C. P. Exposure render: An interactive photo-realistic volume rendering framework. *PloS one*, Public Library of Science San Francisco, USA, v. 7, n. 7, p. e38586, 2012.

[30] PALADINI, G. et al. Optimization techniques for cloud based interactive volumetric monte carlo path tracing. *Industrial Talk, EG/VGTC EuroVis*, 2015.

[31] CLARBERG, P. et al. *Real-Time Path Tracing and Beyond*. 2022. HPG 2022 Keynote.

[32] HOEL, P. G.; PORT, S. C.; STONE, C. J. *Introduction to Probability Theory*. [S.l.]: Houghtion Mifflin, 1972.

[33] HOEL, P. G.; PORT, S. C.; STONE, C. J. *Introduction to Stochastic Processes*. [S.l.]: Houghtion Mifflin, 1972.

[34] BRÉMAUD, P. *Markov Chains, Gibbs Fields, Monte Carlo Simulation, and Queues*. [S.l.]: Springer-Verlag, New York, 1998.

[35] ECKHARDT, R. Stan ulam, john von neumann, and the monte carlo method. *Los Alamos Science*, v. 15, n. 131-136, p. 30, 1987.

[36] METROPOLIS, N.; ULAM, S. The monte carlo method. *Journal of the American statistical association*, Taylor & Francis, v. 44, n. 247, p. 335–341, 1949.

[37] FISHMAN, G. S. *A First Course in Monte Carlo*. [S.l.]: Duxbury Press, 2005.

[38] FISHMAN, G. S. *Monte Carlo: Concepts, Algorithms, and Applications*. [S.l.]: Springer-Verlag, 1996.

[39] KLOEK, T.; DIJK, H. K. V. Bayesian estimates of equation system parameters: an application of integration by monte carlo. *Econometrica: Journal of the Econometric Society*, JSTOR, p. 1–19, 1978.

[40] HECKBERT, P. S. Adaptive radiosity textures for bidirectional ray tracing. In: *Proceedings of the 17th annual conference on Computer graphics and interactive techniques*. [S.l.: s.n.], 1990. p. 145–154.

[41] ARVO, J. et al. State of the art in monte carlo ray tracing for realistic image synthesis. Citeseer, 2001.

[42] PHONG, B. T. Illumination for computer generated pictures. *Communications of the ACM*, ACM New York, NY, USA, v. 18, n. 6, p. 311–317, 1975.

[43] KURI, D. *GPU Path Tracing in Unity – Part 2*. 2018. Disponível em: <http://three-eyed-games.com/2018/05/12/gpu-path-tracing-in-unity-part-2/>.

[44] BOKSANSKY, J. *Crash Course in BRDF Implementation*. 2021.

[45] SHIRLEY, P.; WANG, C.; ZIMMERMAN, K. Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics (TOG)*, ACM New York, NY, USA, v. 15, n. 1, p. 1–36, 1996.

[46] DREBIN, R. A.; CARPENTER, L.; HANRAHAN, P. Volume rendering. *ACM Siggraph Computer Graphics*, ACM New York, NY, USA, v. 22, n. 4, p. 65–74, 1988.

[47] ELVINS, T. T. A survey of algorithms for volume visualization. *ACM Siggraph Computer Graphics*, ACM New York, NY, USA, v. 26, n. 3, p. 194–201, 1992.

[48] CHANDRASEKHAR, S. *Radiative transfer*. [S.l.]: Courier Corporation, 2013.

[49] FASCIONE, L. et al. Path tracing in production-part 1: Production renderers. In: *ACM SIGGRAPH 2017 Courses*. [S.l.: s.n.], 2017. p. 1–39.

[50] LAMBERT, J. H. *Photometria sive de mensura et gradibus luminis, colorum et umbrae*. [S.l.]: sumptibus vidvae E. Klett, typis CP Detleffsen, 1760.

[51] KUTZ, P. et al. Spectral and decomposition tracking for rendering heterogeneous volumes. *ACM Transactions on Graphics (TOG)*, ACM New York, NY, USA, v. 36, n. 4, p. 1–16, 2017.

[52] CARTER, L.; CASHWELL, E.; TAYLOR, W. Monte carlo sampling with continuously varying cross sections along flight paths. *Nuclear science and engineering*, Taylor & Francis, v. 48, n. 4, p. 403–411, 1972.

[53] NOVÁK, J.; SELLE, A.; JAROSZ, W. Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph.*, v. 33, n. 6, p. 179–1, 2014.

[54] HENYEY, L. G.; GREENSTEIN, J. L. Diffuse radiation in the galaxy. *The Astrophysical Journal*, v. 93, p. 70–83, 1941.

[55] GATTASS, M. Apostila de computação gráfica. p. 121–127, 2004.

[56] KAUFMAN, A. E.; MUELLER, K. Overview of volume rendering. *The visualization handbook*, v. 7, p. 127–174, 2005.

[57] HE, Y.; FATAHALIAN, K.; FOLEY, T. Slang: language mechanisms for extensible real-time shading systems. *ACM Transactions on Graphics (TOG)*, ACM New York, NY, USA, v. 37, n. 4, p. 1–13, 2018.

[58] PARKER, S. G. et al. Optix: a general purpose ray tracing engine. *Acm transactions on graphics (tog)*, ACM New York, NY, USA, v. 29, n. 4, p. 1–13, 2010.