UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
FACULDADE DE COMPUTAÇÃO

NOTAS DE AULA DE

COMPUTAÇÃO GRÁFICA

PAULO PAGLIOSA

CAMPO GRANDE, MS

CAPÍTULO 1

Introdução

1.1 O que é Computação Gráfica

Computação gráfica é o conjunto de técnicas e métodos matemáticos e computacionais envolvidos na criação, armazenamento e manipulação de *modelos* e de *imagens* de objetos reais ou imaginários. Dentre as várias disciplinas relativas à computação gráfica, o enfoque neste curso será a *síntese de imagens* em computador, ou *renderização*, isto é, a criação de imagens de objetos a partir dos modelos computacionais correspondentes.

Um modelo é uma *representação* das características principais de um objeto, construída com o propósito de permitir a visualização e a compreensão da estrutura e do comportamento do objeto. Em geral, um modelo não representa todas as características de um objeto, mas aquelas mais relevantes para determinado problema. O que define a validade de um modelo, portanto, é o propósito para o qual este foi construído.

Uma imagem é uma coleção discreta de pontos coloridos chamados *pixels*, organizados em um arranjo retangular chamado *mapa de pixels*. Um pixel é identificado por coordenadas (i, j), $0 \leq i < m$, $0 \leq j < n$, inteiros tomados em relação a um sistema de coordenadas cartesianas 2D chamado *sistema de coordenadas de imagem*, Figura 1.1.

Um pixel é caracterizado por uma cor. Em uma imagem monocromática, a cor de um pixel pode ser definida por um único bit (daí o nome *bitmap*). Em uma imagem com várias cores, uma cor pode ser um índice para uma paleta de cores ou os componentes da cor em relação a um *sistema de cores*. Adotaremos o sistema de cores RGB. Nesse modelo, uma cor é definida por três componentes: r (vermelho), g (verde) e b (azul), $0 \leq r, g, b \leq 1$. O modelo RGB é aditivo, ou seja, dadas duas cores (r_1, g_1, b_1) e (r_2, g_2, b_2), a combinação de ambas é a cor cujos componentes são $(r_1+r_2, g_1+g_2, b_1+b_2)$. O modelo RGB pode ser representado por um cubo unitário, Figura 1.2.

Além dos componentes RGB, cada pixel pode ter como atributo um valor real α, $0 \leq \alpha \leq 1$ (utilizado em diversas operações de composição de imagem). Portanto, um pixel (i, j) de uma imagem pode ser caracterizado por uma quádrupla RGBA de números reais ocupando 16 (precisão simples) ou 32 (precisão dupla) bytes de memória.

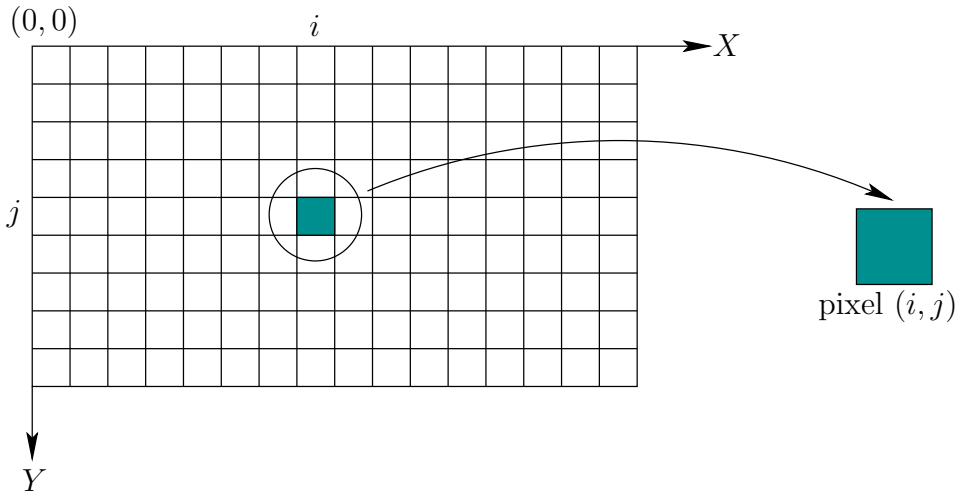

Figura 1.1: Imagem.

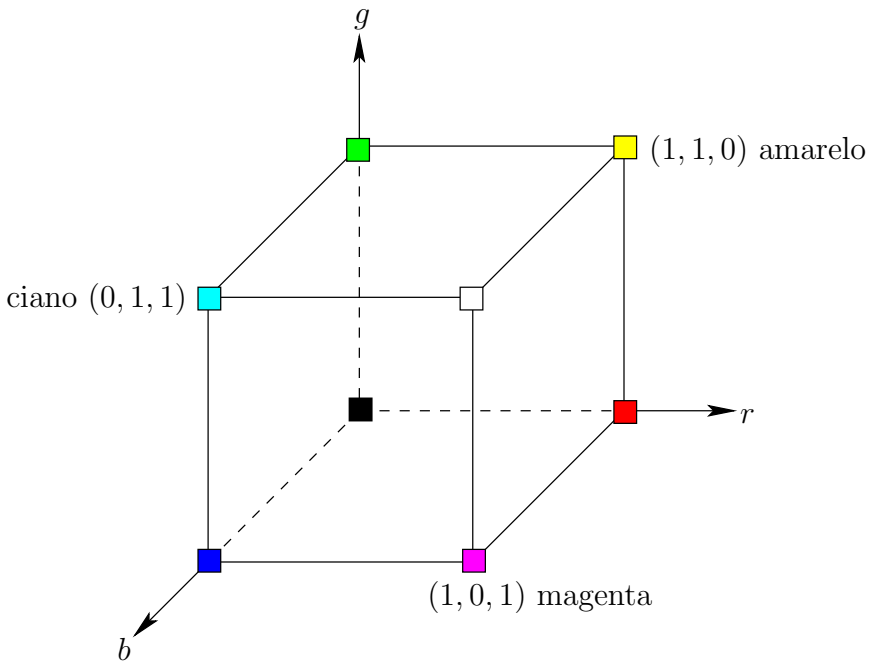

Figura 1.2: Cubo unitário representando as cores RGB.

O modelo RGB permite a especificação de muito mais cores que podem ser distinguidas pelo olho humano ou que podem ser exibidas em um dispositivo gráfico. As unidades de processamento gráfico, ou GPUs (*graphics processing units*), mais comuns permitem a exibição de pouco mais de 16 milhões de cores que podem ser especificadas utilizando-se componentes RGB de 8 bits cada. Portanto, para exibir uma imagem em um dispositivo gráfico, deve-se transformar os componentes RGB reais (r, g, b) em componentes RGB inteiros (r_i, g_i, b_i):

$$r_i = (\texttt{unsigned char})(r \times 255),$$
$$g_i = (\texttt{unsigned char})(g \times 255),$$
$$b_i = (\texttt{unsigned char})(b \times 255).$$

Uma imagem é gerada a partir da descrição de uma porção de um universo n-dimensional chamado *cena* (neste curso consideraremos $n = 3$). Uma cena é uma coleção de *atores* e *luzes*. Um ator representa um objeto visível da cena, sendo caracterizado por um *modelo geométrico* que define exata ou aproximadamente sua posição, formas e dimensões. O modelo geométrico de um ator possui, entre outros atributos, o *material* que constitui sua superfície. Uma luz representa uma fonte luminosa virtual cujos raios incidem sobre as superfícies dos modelos geométricos dos atores da cena. A porção visível da cena é determinada por uma *câmera* virtual que representa o ponto de vista do observador.

A transformação de uma cena em uma imagem se dá através de uma sequência de processos chamada *pipeline de renderização* (*rendering pipeline*). Cada categoria de algoritmo de síntese de imagens possui um pipeline distinto que envolve:

- Determinação de superfícies visíveis, isto é, a determinação de qual ponto de qual superfície de qual ator será mapeado para um pixel da imagem.
- Tonalização (*shading*). Tonalização é o processo de determinação da cor de cada ponto, ou pixel, de uma imagem da cena, a qual é fundamentada na interação da luz da cena com o material das superfícies dos modelos geométricos que definem os atores da cena.
- Texturização (*texturing*). Texturização é o método de determinação da variação, de ponto a ponto, das propriedades das superfícies dos modelos geométricos dos atores da cena. O objetivo da texturização é fornecer a uma superfície detalhes que não são definidos pela geometria da superfície.

A tonalização de uma superfície é baseada em um *modelo de iluminação*, um conjunto de equações matemáticas usualmente derivadas das leis da física que simula a interação da luz com os materiais das superfícies dos objetos. Embora derivados da física, os modelos de iluminação mais utilizados em computação gráfica levam em consideração um grande número de hipóteses simplificadoras em suas formulações, introduzidas para facilitar os cálculos e aumentar sua eficiência computacional.

Um dos modelos de iluminação mais simples, e um dos primeiros a ser utilizado em computação gráfica, é o modelo de iluminação difuso, também chamado de modelo Lambertiano (uma superfície difusa possui uma aparência fosca, sem brilho). O modelo difuso foi seguido por uma variedade de modelos de iluminação mais realísticos, os quais simulam também reflexão especular. Estes modelos são chamados *modelos locais de iluminação*, porque consideram a luz incidente em uma superfície como sendo oriunda diretamente das fontes de luz da cena. No começo da década de 80, a maioria dos esforços dos pesquisadores de modelos de iluminação recaiu sobre os chamados *modelos globais de iluminação*, os quais consideram não somente a luz emitida pelas fontes da luz da cena, mas também a iluminação indireta resultante dos efeitos de reflexão e refração da luz entre as superfícies de todos os atores da cena. As técnicas de síntese de imagens mais empregadas para simular efeitos de iluminação global são *traçado de raios* e *radiosidade*. Enquanto radiosidade admite somente a simulação da iluminação em objetos cujas superfícies sejam Lambertianas, a técnica de traçado de raios permite considerar, de forma simples e elegante, a reflexão especular e as superfícies transparentes, sendo atualmente uma das técnicas mais utilizadas em programas de síntese de imagens realísticas.

Traçado de raios distribuído, particularmente, embora muito mais intensivo em termos computacionais que o traçado de raios determinístico, permite a simulação de muitos outros efeitos tais como profundidade de campo (*depth of field*), translucência, *motion blur* e, mais importante, a atenuação do chamado *aliasing*, uma característica intrínseca e indesejável sempre presente em imagens geradas por técnicas de amostragem tais como o traçado de raios.

Exemplos de Métodos de Síntese de Imagens

Há vários métodos de síntese de imagens de uma cena. O mais simples deles consiste na geração de imagens *fio de arame*, ou seja, nas quais os atores são vistos pelas *arestas* que determinam os contornos das superfícies do objeto. Para a obtenção deste tipo de imagem, um modelo geométrico de um ator é geralmente representado por uma malha de polígonos que define exata ou aproximadamente a superfície do objeto, e o modelo deve ser capaz de informar ao renderizador da imagem quais são suas arestas (ou o renderizador deve ser capaz de extrair as arestas do modelo). Neste caso, o modelo pode ser definido por uma estrutura de dados simples que representa apenas as arestas dos polígonos que definem a superfície do objeto.

Outro tipo de imagem é aquela na qual os trechos de superfícies mais próximos do observador obstruem ou escondem aqueles trechos de superfície que estão atrás. Esse tipo de imagem é chamada imagem com *remoção de linhas* ou *superfícies escondidas*. Para tanto, um modelo geométrico deve prover ao renderizador capacidade para extração das *faces*, isto é, os próprios polígonos, que compõem a superfície do objeto, necessárias para o cálculo de profundidade e oclusão. Para modelos poligonais, as faces devem ser explicitamente definidas na estrutura de dados do modelo.

A Figura 1.3 ilustra imagens com fio de arame e remoção de linhas escondidas de um cubo. Na primeira, as faces podem ser consideradas transparentes e, na segunda, totalmente opacas, mas ainda sem cor.

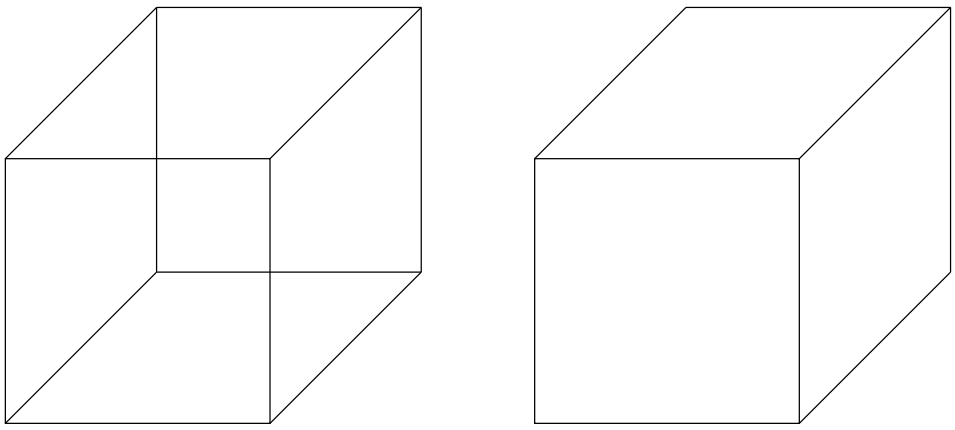

Figura 1.3: Imagem fio de arame e com linhas escondidas de um cubo.

A seguir, introduziremos um dos algoritmos mais empregados na renderização de imagens fotorrealísticas em computação gráfica, chamado *traçado de raios*.

1.2 Traçado de Raios

O traçado de raios é derivado do modelo de câmera mais fundamental da computação gráfica, chamado *buraquinho de alfinete*. Neste, a câmera é representada por uma caixa contendo em seu interior um filme fotográfico fixado em uma das faces. No centro da face oposta há um pequeno orifício, o "buraquinho de alfinete", protegido por um anteparo que impede que a luz exterior entre na caixa. Removendo-se o anteparo por um breve instante, a luz passa pelo orifício, entra na caixa e impressiona o filme, formando uma imagem (invertida).

Na prática, o filme é considerado como situado à frente do buraquinho de alfinete. O buraquinho representa o olho do observador e o filme representa a imagem. A Figura 1.4 mostra o volume de vista, chamado *frustum*, cuja forma é um tronco de pirâmide com ápice na posição do observador e com faces laterais formadas pelos chamados *projetores*. Estes são linhas que partem da posição do observador em direção às arestas de um retângulo denominado *janela de projeção*. O eixo formado pela posição do observador e o centro da janela de projeção define a *direção de projeção*. As faces da frente e do fundo, paralelas à janela de projeção, são definidas por planos de corte de frente (ou de perto) e de fundo (ou de longe), respectivamente.

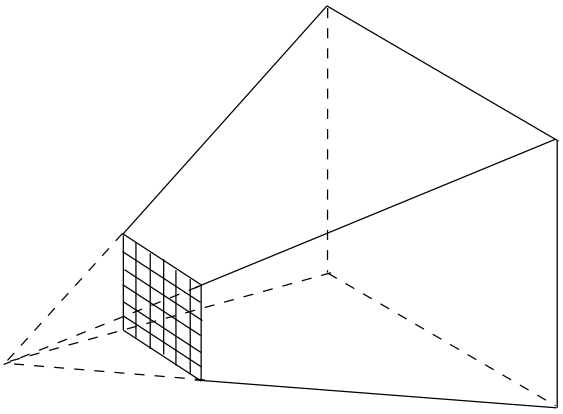

Figura 1.4: Frustum.

Os atores e luzes de uma cena são posicionados em relação a um sistema cartesiano de coordenadas chamado *sistema global* ou *mundial de coordenadas*, designado pelo acrônimo WC (*world coordinates*), Figura 1.5. O sistema tem origem em um ponto O qualquer do espaço Euclidiano $\mathbb{E}^3$ e eixos x, y, z nas direções dos *versores* $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$, respectivamente. Tais versores formam a *base canônica* do espaço vetorial $\mathbb{R}^3$, a partir da qual podemos definir as coordenadas de qualquer vetor $\mathbf{u}$, conforme veremos no Capítulo 2.

O traçado *progressivo* de raios segue todos os raios de luz emitidos de todas as fontes de luz da cena e verifica quais destes raios alcançam o olho do observador. Consideremos inicialmente cenas com objetos opacos somente (desconsideraremos a refração). Como exemplo, seja a cena da Figura 1.6.

O raio A não intercepta nenhum objeto da cena e não contribui para a formação da imagem. Da mesma forma, o raio B não contribui para a imagem, pois reflete no bloco menor e segue uma trajetória que não lhe conduz em direção à posição do observador.

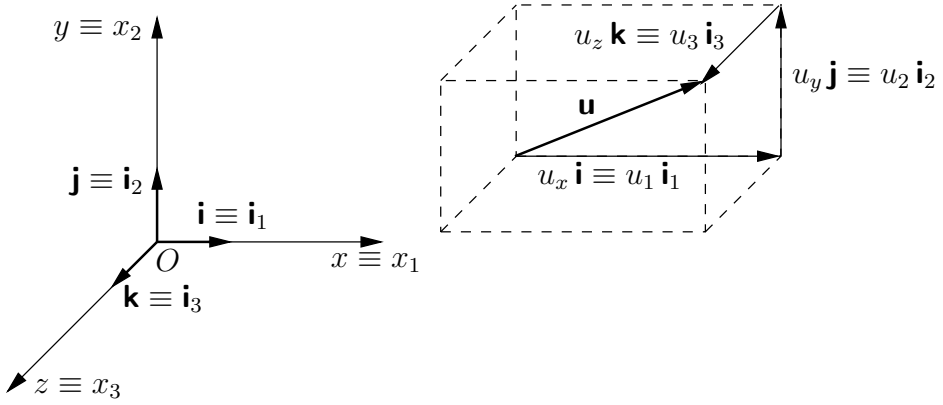

Figura 1.5: WC.

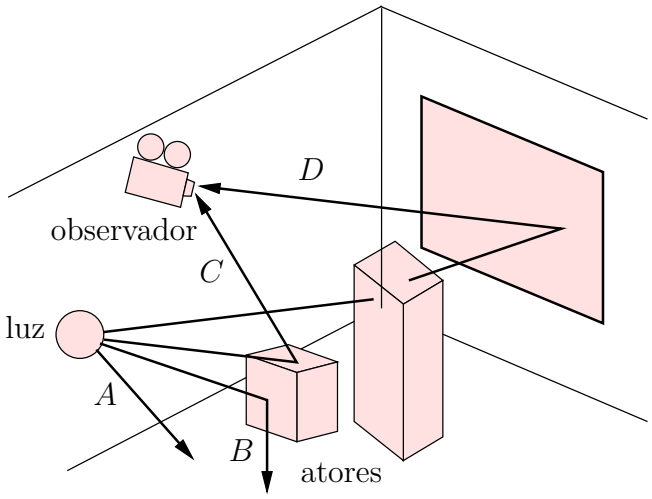

Figura 1.6: Traçado de raios progressivo.

O raio C, ao contrário, intercepta o bloco menor e por ele é refletido em direção ao olho do observador, passando por um ponto da imagem (ou seja, entrando no buraquinho de alfinete e impressionando o filme). A cor do pixel correspondente ao raio C é função da energia da fonte emissora, do decaimento dessa energia à medida que o raio trafega no espaço e das propriedades materiais da superfície do objeto (o bloco menor) no ponto de interseção. Observamos um ponto sobre a superfície do objeto como consequência da iteração de um raio de luz diretamente proveniente de uma fonte luminosa (ou seja, efeito da iluminação direta, a qual independe de quaisquer outros objetos presentes na cena). O raio D também contribui para a formação da imagem, mas chega ao olho do observador como decorrência da inter-reflexão entre o bloco maior e o espelho ao fundo. Note que a contribuição do raio D não depende apenas da luz e do bloco maior, mas também da presença do espelho no qual o bloco aparece refletido (ou seja, iluminação indireta). Traçado de raios, portanto, implementa um modelo global de iluminação.

No traçado de raios progressivo a imagem é formada seguindo-se a trajetória de todos os raios emitidos por todas as fontes e considerando-se apenas aqueles que chegam ao olho do observador, o que é inviável computacionalmente. Ao invés de seguir os raios da fonte até o olho do observador, na prática faz-se o oposto: traçam-se raios partindo do olho passando pela imagem, verifica-se se há interseção desses raios

com os atores da cena e então determina-se a cor nestes pontos de interseção. Este é o traçado de raios *regressivo*, ou simplesmente traçado de raios (*ray tracing*).

Para a geração de uma imagem $m \times n$, o filme é dividido em $m \times n$ áreas, também chamadas de pixels. Na versão mais simples do algoritmo, consideraremos que a cor de um pixel será determinada por somente um raio, disparado do olho na direção do centro do pixel, chamado *raio de pixel*, Figura 1.7.

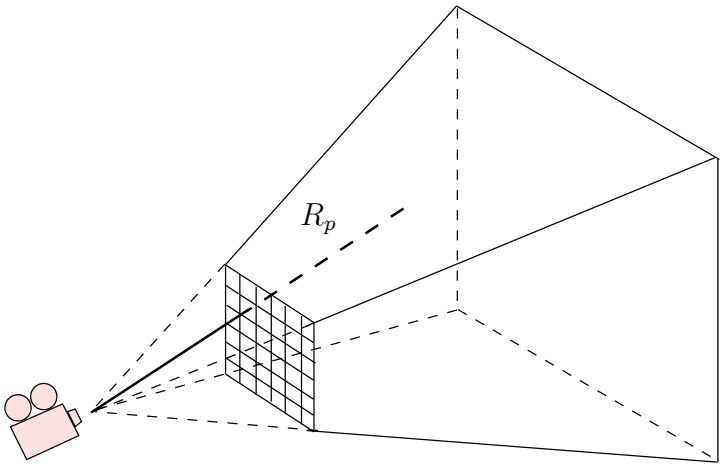

Figura 1.7: Raio de pixel.

O algoritmo do traçado de raios é sumarizado a seguir.

Passo 1 **P**ara cada pixel (i, j) da imagem, $0 \leq i < m$, $0 \leq j < n$, trace um raio de pixel R_p do olho do observador em direção ao centro do pixel. A cor do raio R_p será a cor do pixel (i, j).

Passo 2 **S**e R_p não interceptar nenhum ator, sua cor será a *cor de fundo* da cena. Caso contrário, seja I_p o ponto de interseção de R_p com a superfície do objeto O_p mais próximo da origem de R_p, Figura 1.8.

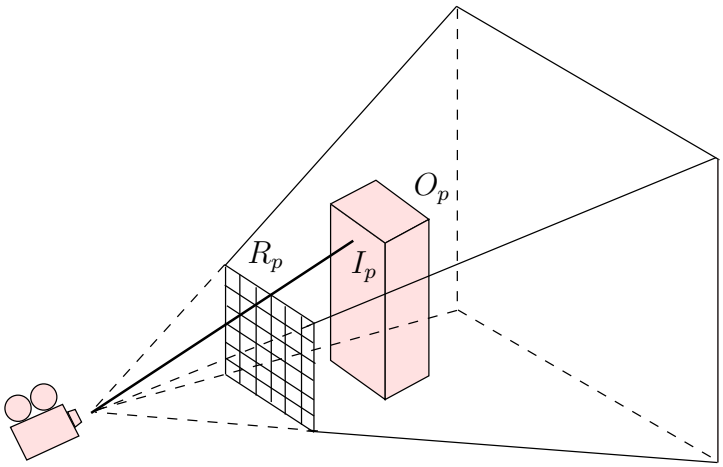

Figura 1.8: Interseção de R_p com O_p.

Passo 3 **D**o ponto I_p traçam-se raios em direção a cada uma das fontes de luz da cena. Esses raios são chamados de *raios de luz* (ou *raios de sombra*). Seja então uma fonte de luz l e o raio de luz R_l, Figura 1.9. Se R_l não interceptar nenhum ator da cena, então l ilumina diretamente I_p. A cor de R_l é determinada em função de um modelo local de iluminação e adicionada à cor de R_p. Caso contrário, R_l é um raio de sombra com cor preta.

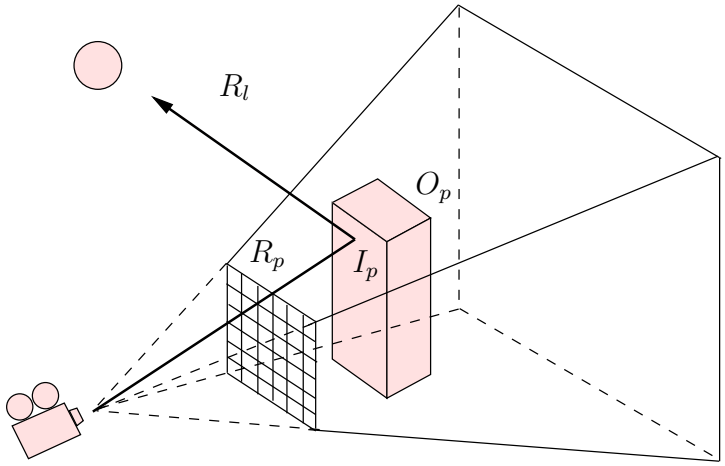

Figura 1.9: Raio de luz.

Passo 4 **S**e o material de O_p em I_p for reflexivo, traça-se um raio R_r partindo de I_p na direção de reflexão. R_r é chamado *raio de reflexão*. A cor de R_r é adicionada à cor de R_p. A determinação da cor de R_r é feita executando-se recursivamente o algoritmo considerando-se que R_r é um raio de pixel e que I_p é o olho do observador. A direção de reflexão é função da direção de incidência do raio de pixel em I_p e do *vetor normal* à superfície de O_p em I_p.

Os critérios de parada da recursão serão vistos no Capítulo 4 e detalhes do algoritmo no Capítulo 6.

1.3 Organização do Curso

Capítulo 2
Vetores e Transformações Geométricas 3D

Em computação gráfica, vetores 3D são usados, entre outros, para definir pontos, direções e normais a uma superfície no espaço. No Capítulo 2 são revistas as principais operações entre vetores 3D e algumas de suas aplicações em computação gráfica, e as transformações geométricas afins elementares de pontos, direções e normais.

Capítulo 3
Modelagem Geométrica

A geometria de um ator é representada por um modelo geométrico. Este contém informações sobre a posição, dimensões e forma do objeto sendo representado, além

de outros atributos (por exemplo, o material). A forma do objeto, ou sua *topologia*, pode ser caracterizada pela conectividade e relações de adjacência entre componentes do modelo tais como arestas, vértices e faces. No Capítulo 3 são estudados três tipos de modelos geométricos: modelos poligonais, quádricas simples e CSG (*constructive solid geometry*).

Capítulo 4
Modelos de Iluminação

No Capítulo 4 são vistos tipos simples de luzes virtuais usadas em computação gráfica e a formulação de um modelo de iluminação local denominado modelo de iluminação de Phong, empregado na determinação da cor de um ponto devida à contribuição da iluminação direta.

Capítulo 5
Câmera Virtual

No Capítulo 5 são descritos os parâmetros que definem uma câmera virtual e o volume de vista por ela representado, bem como algumas operações com a câmera. São estudadas ainda as transformações de normalização para dois tipos de projeção planar: perspectiva e paralela ortográfica.

Capítulo 6
Algoritmos de Síntese de Imagens

No Capítulo 6 são descritos, em adição ao algoritmo de traçado de raios introduzido na Seção 1.2, os pipelines de dois algoritmos de síntese de imagens de modelos poligonais: *spanning scan-line* e *Z-buffer* (este último implementado na biblioteca gráfica OpenGL).

1.4 Exercícios

1.1 Seja $\mathbf{I}$ um mapa de pixels $m \times n$. Descreva um algoritmo de *inversão* das cores RGB dos pixels de $\mathbf{I}$.

1.2 O que é frustum?

1.3 Descreva o algoritmo de traçado de raios.

1.4 Um ponto recebe iluminação direta de uma luz vermelha e uma luz verde. Qual a cor no ponto? E se as luzes forem branca e azul?