Geração de Imagens Realistas
– Algoritmo de Rastreamento de Raios –

Este capítulo descreve o algoritmo de Rastreamento de Raios para geração de imagens a partir de um cenário modelado no computador. Este algoritmo é pouco eficiente quando comparado com outros, como o Zbuffer, que é usado nas placas gráficas, mas produz imagens que parecem reais como uma fotografia. Além de produzir imagens foto-realistas, o algoritmo de rastreamento de raios é simples e direto. Ele constitui uma boa base para a apresentação dos modelos de câmera, objetos, iluminação e textura. Estes modelos formam a base da Computação Gráfica 3D.

O algoritmo de Rastreamento de Raios procura simular uma câmera fotográfica e por isto este capítulo se inicia com uma apresentação da Câmera "*Pinhole*". Esta câmera é simples e contém os principais elementos da óptica necessários para o algoritmo. Para tornar a discussão mais interessante apresentamos alguns dados históricos da busca da humanidade pela aquisição e armazenamento de imagens do mundo que nos cerca.

5.1 A Câmera Obscura e a Câmera "Pinhole"

A busca por uma representação gráfica das cenas que enxergamos vem desde os tempos em que o homem habitava as cavernas. Pintores renascentistas, como o Canalleto (Fig. 5.1a), foram capazes de pintar cenas que se assemelham a fotografias sem os conhecimentos da geometria projetiva que temos hoje, graças ao auxílio de câmaras escuras como a ilustrada na Fig. 5.1b. Ou seja, o processo não se baseava em modelos geométricos, mas era artístico e dependente do talento do pintor.

A geração de imagens realistas na Computação Gráfica se baseia em modelos geométricos e ópticos de câmera, de objetos e de iluminação, mas, como procuraremos mostrar neste e nos próximos capítulos, estes modelos não dispensam uma certa criatividade tanto na sua formulação quanto na sua utilização. A atitude de "copiar o que observamos" permanece. O caráter artístico também continua importante. As ferramentas deste artista moderno, entretanto, deixam de ser o pincel e a tela e se tornam modelos matemáticos e programas de computador. Os magos de efeitos especiais de industria cinematográficas são estes novos artistas.

(a) Uma pintura de Canaletto (1697-1768).

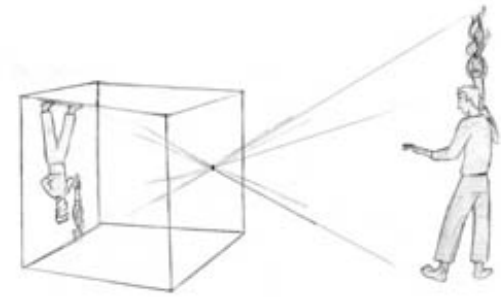
(b) Uma câmera obscura portátil.

Fig. 5.1 – Artifício para produzir uma pintura com foto-realismo.

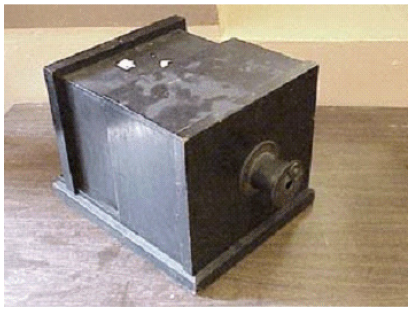
A câmera obscura não é uma invenção renascentista da época de Canaletto. A primeira referência conhecida a câmeras escuras, ou *camera obscura* (*latim*), foi do filósofo chinês Mo-Ti (V século antes de Cristo). Ele registrou a criação de uma imagem invertida formada por raios de luz que passavam através de um pequeno orifício na janela de um quarto escuro da forma ilustrada na Fig. 5.2a.

Aristóteles (384-322 AC) entendeu o princípio óptico da câmera escura a utilizou para observar uma eclipse parcial do sol. Outras menções antigas à câmera obscura podem ser encontradas no trabalhos do matemático árabe do século X, Alhazen de Basra, e de Leonardo da Vinci (XVI DC).

Um invento associado com o mesmo princípio óptico da câmera obscura foi a câmera *pinhole* ilustrada na Figura 5.2b. Esta figura mostra um esquema de um câmera amadora feita de lata com um furinho na tampa e uma foto de um dos primeiros modelos de câmera industrial feita por Luis-Jacques-Mandé Daguerre (1839).



(a) Câmera obscura

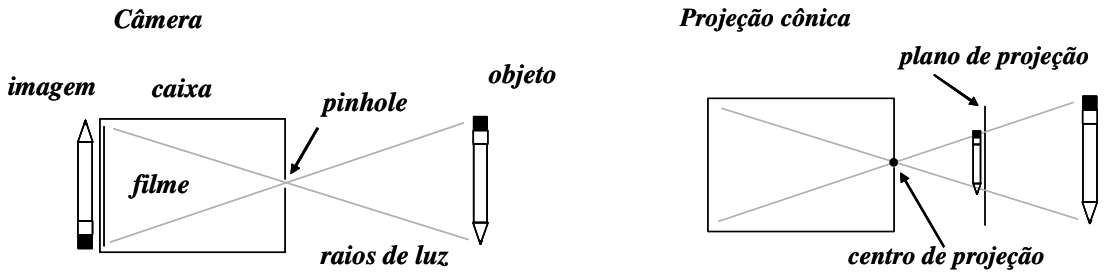


(b) Cameras *pinhole*

Fig. 5.2 – Câmera obscura e câmeras *pinhole*

Uma dos princípios ópticos observados na câmera obscura é que os raios de luz viajam em linha reta. Assim, a cor de um ponto na imagem projetada corresponde ao ponto de interseção da reta que passa por este ponto e pelo orifício da câmera com o objeto iluminado.

O processo geométrico de formação da imagem na câmera *pinhole* corresponde à projeção cônica dos objetos iluminados no plano que contém a imagem. O centro de projeção da projeção cônica é a posição do orifício da câmera. A imagem projetada é invertida como ilustra a Fig. 5.3a. Matematicamente podemos obter a mesma imagem, sem a inversão, se colocarmos o plano de projeção entre o centro de projeção e o objeto, como ilustra a Fig. 5.3b.



(a) Geometria da câmera (b) Projeção cônica equivalente

Fig. 5.3 – Inversão na posição do plano de formação da imagem.

A partir destes princípios ópticos simples podemos enunciar em idéias gerais o que é o algoritmo de Rastreamento de Raios. Considere a Fig. 5.4: para definir a cor de um *pixel* da imagem lançamos um raio que vai do centro de projeção (*eye* na figura) até uma posição correspondente ao *pixel* no plano de projeção. Calculamos a interseção deste segmento de reta com todos os objetos da cena tomando como visível aquele objeto que estiver mais próximo. A cor do *pixel* é a cor do objeto iluminado pelas fontes de luz no ponto de interseção.

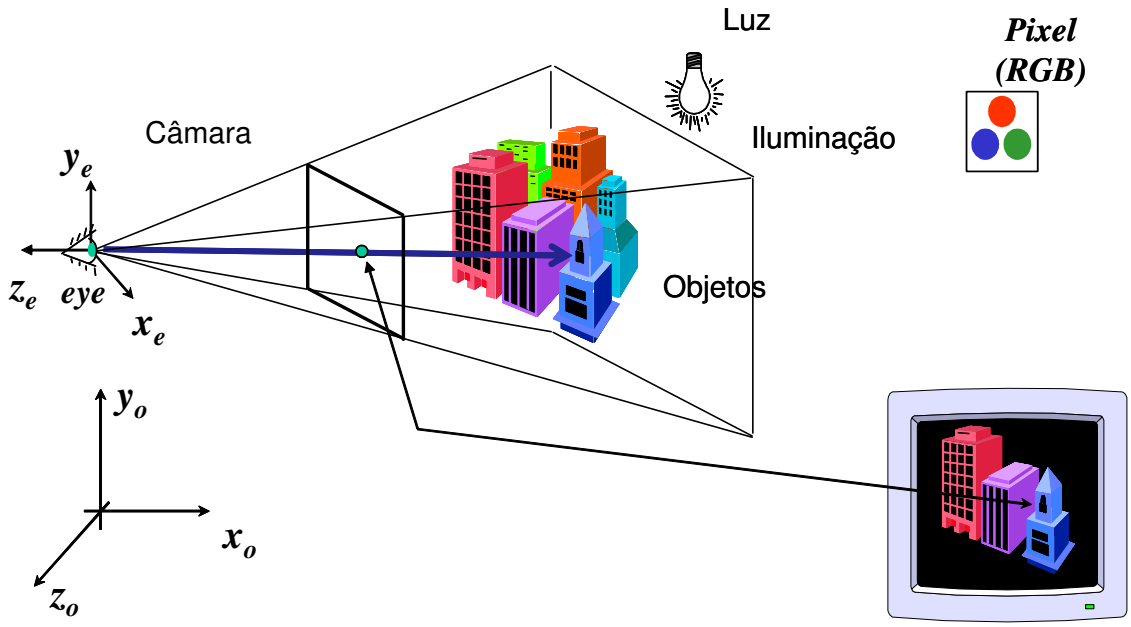


Fig. 5.4 – Idéia geral do algoritmo de Rastreamento de Raios.

O centro de projeção é denominado aqui de **eye** para utilizar a mesma nomenclatura do OpenGL, de forma a facilitar comparações. O quadro 5.1 ilustra uma forma simples do algoritmo de Rastreamento de Raios.

```
Para cada pixel da tela

      Lance uma raio;

      Para cada objeto da cena
            Calcule a interseção do raio com este o objeto;
            Armazene a interseção mais próxima;

      Se o raio interceptou algum objeto
            Calcule a contribuição das luzes na cor deste ponto;
            Pinte o pixel com esta cor.
```

Quadro 5.1 – Algoritmo Básico de Rastreamento de Raios.

Este algoritmo requer um modelo geométrico para a câmera que seja capaz de fornecer um segmento de reta (raio) para cada *pixel* da imagem. Necessita também de uma descrição dos objetos, como um conjunto de pontos do espaço com propriedades que descrevam como a luz reflete neles. Finalmente, temos que ter modelos para as luzes indicando as regiões do espaço que elas iluminam.

5.2 Modelo de Câmera

No algoritmo de Rastreamento de Raios a câmera é um objeto matemático que fornece raios para determinar a cor de cada um dos *pixels* da imagem digital gerada. Um raio é o conjunto de pontos da forma $\mathbf{p}(t)=\mathbf{o}+t\mathbf{d}$, com $t>0$. Isto porque só nos interessam as interseções com objetos que estejam à frente do olho.

O estudo de uma câmera virtual tem duas etapas. A primeira trata dos parâmetros que definem a câmera e onde ela está posicionada na cena. A segunda etapa determina qual o raio correspondente ao *pixel* (x,y) de uma dada câmera.

Definição da Câmera

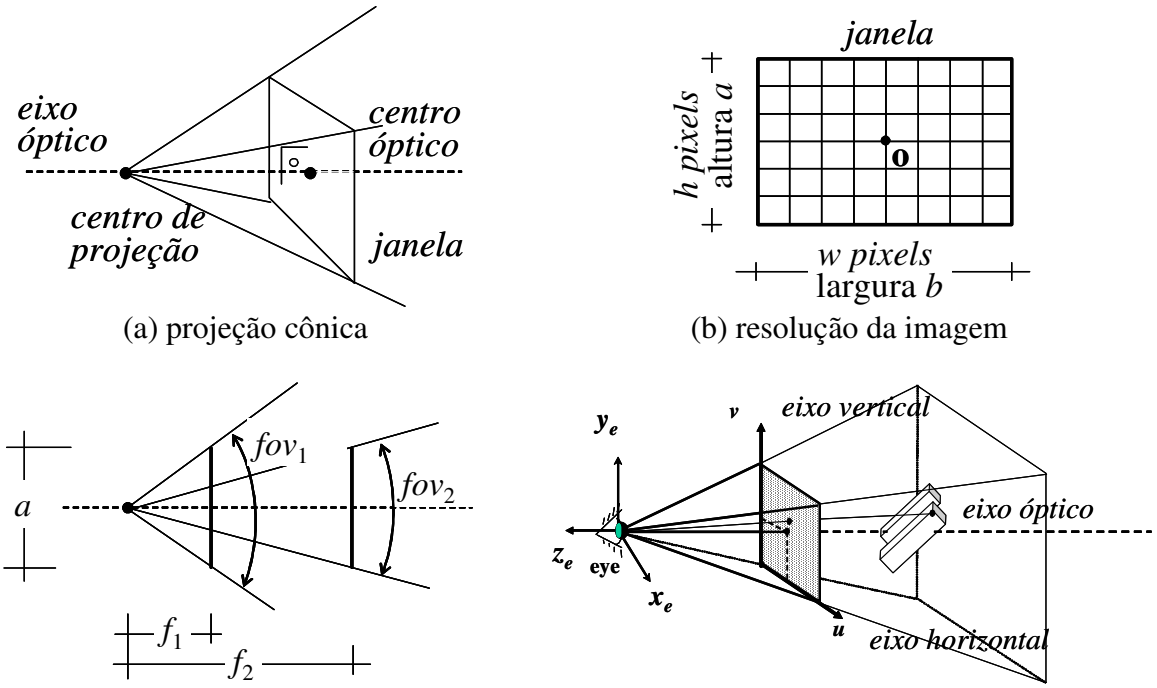
Assim como nas câmeras fotográficas reais, é natural nos referirmos a uma câmera virtual diferenciando os parâmetros que são **intrínsecos** a ela dos que definem o seu posicionamento na cena, ditos **extrínsecos**. Assim numa animação em que a se câmera move dentro de um cenário, podemos, por exemplo, definir os parâmetros intrínsecos da câmera uma só vez. Entre um quadro e outro apenas os parâmetros extrínsecos mudam. Numa outra situação, uma mesma câmera pode ser re-utilizada de um cenário para outro.

Uma câmera virtual simples deve procurar representar a essência da geometria de uma câmera *pinhole.* Se adotarmos o modelo de projeção cônica ilustrado nas Figs. 5.3b e 5.4,

cada câmera terá um centro de projeção, um eixo óptico e um retângulo (janela) onde se forma uma imagem de *w*×*h pixels*.

Nas câmeras normais o eixo óptico é perpendicular a este retângulo e o intercepta exatamente no seu centro, como ilustram as Figs. 5.5a e 5.5b. O tamanho do retângulo e a sua distância ao centro de projeção definem a abertura da câmera ou campo de visão, *fov* (*field of vision*), como ilustra a Fig. 5.5c.

O eixo óptico e as direções dos lados do retângulo da janela definem três direções que dão orientação aos ditos **eixos da câmera** x_e y_e z_e e **eixos da imagem** *uv* mostrados na Fig. 5.5c. A escolha do z_e voltado para trás do olho é feita para compatibilizar os requisitos de que o eixo x_e aponte para direita, o eixo y_e para cima e o sistema x_e y_e z_e seja dextrógiro. O OpenGL também adota a mesma convenção.



(a) projeção cônica (b) resolução da imagem

(c) abertura ou campo de visão (d) eixos da câmera

Fig. 5.5 – Parâmetros intrínsecos de uma câmera simples.

Os parâmetros *f*, *fov*, *a, b*, *w* e *h* mostrados na Fig. 5.5 não são independentes. As relações de dependência nos permitem escolher quais parâmetros devem definir uma câmera e quais ficam automaticamente especificados pelos primeiros.

Na maioria dos dispositivos atuais, por exemplo, os *pixels* são quadrados. Com isto, se o número de *pixels* na horizontal e na vertical é definido, temos automaticamente a razão de lados da janela através da equação:

$$\frac{b}{a} = \frac{w}{h} \qquad (5.1.a)$$

como mostra a Fig. 5.5b. Devemos notar que as imagem dos monitores atuais é 640×480, 800×600, 1024×768 ou 1600×1200, fixando que a janela tem a razão 4:3. Ou seja, o valor da altura de uma janela que deve ser exibida em um monitor padrão deve ter 75% do valor da base.

Outra relação importante pode ser deduzida da Fig. 5.5c:

$$\frac{a}{2f} = \tan\left(\frac{fov}{2}\right) \tag{5.1.b}$$

Podemos então escolher tanto a razão *a/f* quanto o ângulo *fov* para indicar a abertura da câmera como ilustra a Fig. 5.5c. Nas máquinas fotográficas normais o ângulo *fov* varia de 60º a 90º. Quando ele é pequeno a imagem é gerada com um efeito de teleobjetiva e quando ele é grande o efeito é o de grande angular.

Uma boa escolha de parâmetros são os parâmetros da função gluPerspective do OpenGL:

```
void gluPerspective( GLdouble fovy, Gldouble aspect,
                     Gldouble near, Gldouble far );
```

Onde *fovy* é o campo de visão em y_e (ver Fig. 5.5c e 5.5d) e *aspect* é a razão *w/h* (4/3 nos monitores padrão). Os demais parâmetros são particulares do algoritmo de Zbuffer, que só desenha os objetos cujas coordenadas z_e estejam entre *–near* e *–far* (o sinal negativo é necessário uma vez que *near* e *far* são distâncias e a parte visível é a que fica com $z_e<0$). Ou seja, diferente do algoritmo de Rastreamento de Raios, o ZBuffer não pode trabalhar sem especificar uma profundidade mínima e máxima. Apesar desses parâmetros não serem necessários no algoritmo de Rastreamento de Raios, vamos mantê-los na definição de nossa câmera simples. Assim uma mesma definição de cena pode ser utilizada para ambos os algoritmos e as imagens geradas podem ser comparadas, mesmo que o algoritmo de Rastreamento de Raios desenhe alguns objetos a mais.

Com base na discussão acima, os parâmetros intrínsecos de uma câmera podem ser definidos aqui por: *fov*, *w*, *h*, *near* e *far*. Como *w* e *h* são necessários (precisamos saber quantos raios lançar) a razão *aspect* deixa de ser um parâmetro primário e se torna um parâmetro calculado. Os demais parâmetros podem ser calculados por:

$$f = near \tag{5.2.a}$$

$$a = 2f\tan\left(\frac{fov}{2}\right) \tag{5.2.b}$$

$$b = \frac{w}{h}a \tag{5.2.c}$$

se colocarmos o plano de projeção na coordenada z=*–near*. Devemos notar ainda que, nesta escolha de parâmetros mostrada acima, está implícita a posição do centro óptico no meio da janela, por isto a posição do centro óptico não foi especificada explicitamente.

Para definir a posição da câmera na cena (parâmetros extrínsecos) temos que saber a posição do centro de projeção, a direção para onde a câmera está apontando e a rotação da câmera em torno de um eixo nesta direção. Definir a posição do centro de projeção é simples: para especificar uma direção basta definirmos na cena outro ponto para onde a

câmera esteja apontando. Definir o ângulo que produza o posicionamento desejado, entretanto, não é uma tarefa tão simples. A Fig. 5.6 ilustra o efeito de rotação de uma câmera em torno de se eixo óptico.

Fig. 5.6 – Fotos tiradas rodando a câmera em torno de seu eixo óptico.

Para escolher os parâmetros extrínsecos podemos novamente utilizar os parâmetros de uma função OpenGL, no caso a **gluLookAt**. Nela os parâmetros extrínsecos de uma câmera são: a posição do centro de projeção (***eye***), um ponto para onde a câmera esteja apontando (***center***) e um vetor que indique a direção "para cima" da câmera (***up***). Note que este vetor ***up*** pode ser qualquer vetor que esteja no plano $y_e\ z_e$ da Fig. 5.5c e não é, necessariamente, perpendicular a eixo óptico que vai de ***eye*** até o ***center***. Esta liberdade permite que, por exemplo, utilizemos a direção vertical de uma cena, y_0 na Fig. 5.7, para especificar uma câmera que passeie pela cena apontando para cima e para baixo sem, entretanto, sofrer rotação em torno de seu eixo.

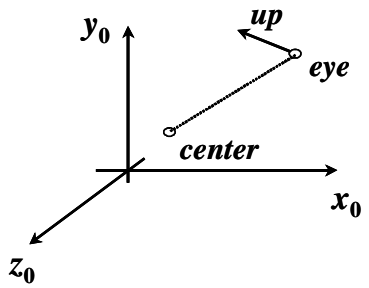


Figura 5.7 –Parâmetros extrínsecos de uma câmera.

Lançamento dos raios

Com a câmera definida pelos parâmetros indicados na seção anterior, que são propícios para uma interface humana, é necessário um certo pré-processamento para que a determinação de um raio se faça de forma eficiente.

Primeiramente precisamos determinar o sistema de coordenadas da câmera (ou do ***eye***) $\mathbf{x}_e\mathbf{y}_e\mathbf{z}_e$ e para isto considere a Fig. 5.8. Com o eixo óptico definido pela reta que passa pelo ***eye*** e pelo ***center*** o eixo **z** pode ser determinado por:

$$\mathbf{z}_e = \frac{1}{\|\mathbf{eye} - \mathbf{center}\|}(\mathbf{eye} - \mathbf{center}) \tag{5.3.a}$$

Como o vetor ***up*** está no plano $\mathbf{y}_e\mathbf{z}_e$ é mais fácil calcularmos o unitário $\mathbf{x}_e$ através de:

$$\mathbf{x}_e = \frac{1}{\|\mathbf{up} \times \mathbf{z}_e\|}(\mathbf{up} \times \mathbf{z}_e) \tag{5.3.b}$$

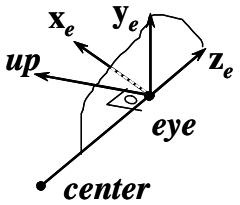


Fig. 5.8 – Relação entre os vetores da base $\mathbf{x}_e\mathbf{y}_e\mathbf{z}_e$

Com $\mathbf{x}_e$ e $\mathbf{z}_e$ determinados o unitário $\mathbf{y}_e$, que completa o sistema de eixos dextrógiro é dado por:

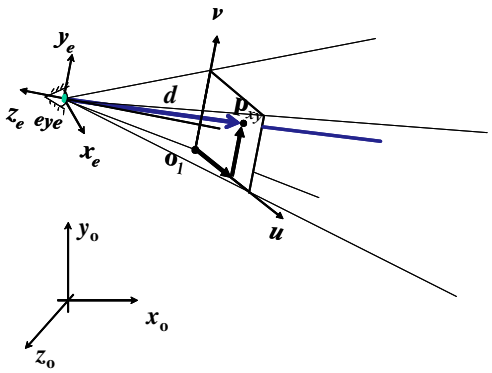
$$\mathbf{y}_e = \mathbf{z}_e \times \mathbf{x}_e \tag{5.3.c}$$

Com base na Fig. 5.9(a), vemos que a partir do sistema de eixos da câmera podemos determinar um raio correspondente ao *pixel* (x,y), $x = 0,1,...(w-1)$ e $y = 0,1,...(h-1)$, na forma:

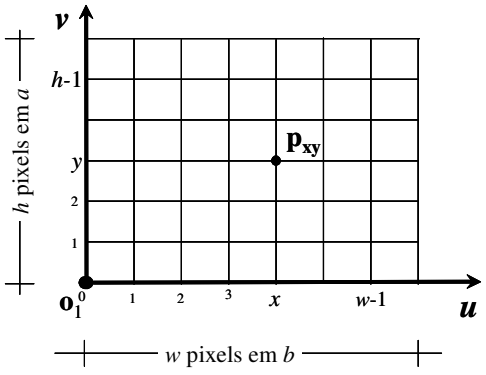
$$\mathbf{p}(t) = \mathbf{eye} - t\mathbf{d} \tag{5.4}$$

onde:

$$\mathbf{d} = \mathbf{p}_{xy} - \mathbf{eye} \tag{5.5}$$



(a) raio (**o**,**d**).



(b) escala de *pixel* para janela.

Figura 5.9 – Lançamento de raios por uma câmera.

A determinação do ponto $\mathbf{p}_{xy}$ é um pouco mais elaborada. A Fig. 5.9(b) mostra que ele pode escrito como sendo:

$$\mathbf{p}_{xy} = \mathbf{o}_1 + u(x)\hat{\mathbf{u}} + v(y)\hat{\mathbf{v}} \tag{5.6}$$

onde $\mathbf{o}_1$ é a origem do sistema de eixos do plano de projeção ***uv***, que é alinhado com a imagem. Os unitários $\hat{\mathbf{u}}$ e $\hat{\mathbf{v}}$ correspondem aos unitários $\hat{\mathbf{x}}_e$ e $\hat{\mathbf{y}}_e$, respectivamente, e a posição da origem $\mathbf{o}_1$ pode ser determinada por:

$$\mathbf{o}_1 = \mathbf{eye} - f\,\hat{\mathbf{z}}_e - \frac{a}{2}\hat{\mathbf{y}}_e - \frac{b}{2}\hat{\mathbf{x}}_e \tag{5.7}$$

As funções $u(x)$ e $v(y)$ são simples escalas para acomodar as coordenadas da imagem (x,y) dadas em *pixels* nas coordenadas da janela. Como ilustra a Figura 5.5b estas escalas podem ser calculadas por:

$$u(x) = \frac{x}{w}b \tag{5.8a}$$

$$v(y) = \frac{y}{h}a \tag{5.8b}$$

Com os resultados das equações (5.8) e (5.6), a equação (5.5) pode re-escrita como:

$$\mathbf{d} = \left(\mathbf{o}_1 + b\frac{x}{w}\hat{\mathbf{x}}_e + a\frac{y}{h}\hat{\mathbf{y}}_e\right) - \mathbf{eye} \tag{5.9}$$

Substituindo a expressão de $\mathbf{o}_1$ dada em (5.7) nesta equação, temos:

$$\mathbf{d} = -f\,\hat{\mathbf{z}}_e + a\left(\frac{y}{h} - \frac{1}{2}\right)\hat{\mathbf{y}}_e + b\left(\frac{x}{w} - \frac{1}{2}\right)\hat{\mathbf{x}}_e \tag{5.10}$$

Esta expressão também poderia ser deduzida diretamente da Fig. 5.9 se observarmos que **d** tem estas coordenadas no sistema da câmera.

O quadro abaixo resume o processamento do objeto câmera.

Dados: *fov, w, h, near, far,* ***eye, center, up***

Inicialização (pré-processamento):

$$f = near \qquad a = 2f\tan\left(\frac{fov}{2}\right) \qquad b = \frac{w}{h}a$$

$$\mathbf{z}_e = \frac{1}{\|\mathbf{eye}-\mathbf{center}\|}(\mathbf{eye}-\mathbf{center}) \qquad \mathbf{x}_e = \frac{1}{\|\mathbf{up}\times\mathbf{z}_e\|}(\mathbf{up}\times\mathbf{z}_e) \qquad \mathbf{y}_e = (\mathbf{z}_e\times\mathbf{x}_e)$$

Lançamento de raios: $\mathbf{o} + t\mathbf{d}$

$$\mathbf{o} = \mathbf{eye}$$

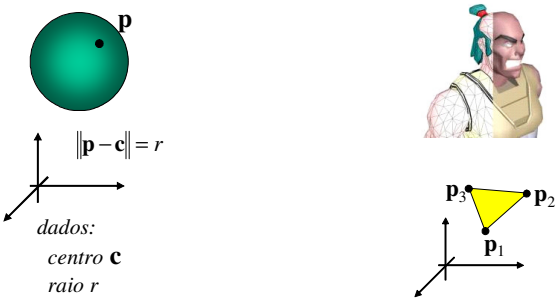
$$\mathbf{d} = -f\,\hat{\mathbf{z}}_e + a\left(\frac{y}{h}-\frac{1}{2}\right)\hat{\mathbf{y}}_e + b\left(\frac{x}{w}-\frac{1}{2}\right)\hat{\mathbf{x}}_e$$

Quadro 5.2 – O objeto câmera

5.3 Modelo Geométrico dos Objetos

A luz captada por uma câmera fotográfica é, geralmente, a luz refletida sobre a superfície dos objetos que compõem uma cena. Por isto, a descrição geométrica de um objeto foca na descrição de sua superfície externa. Nesta seção estudaremos a definição geométrica das superfícies e nas seções seguintes vamos tratar de suas propriedades ópticas. O que estamos estudando aqui é a forma de definirmos um objeto, de calcularmos a interseção de um raio com ele e de determinarmos a normal à superfície do objeto no ponto de interseção. Esta normal é necessária para os cálculos de iluminação que veremos nas próximas seções.

As superfícies dos objetos de uma cena são geralmente descritas através um dos dois tipos de modelos ilustrados na Fig. 5.10: (a) implícitos e (b) de fronteira. Nos modelos implícitos, ilustrados pela esfera na figura, as coordenadas dos pontos do objeto satisfazem uma equação. Nos modelos de superfície, ilustrados pelo guerreiro, a superfície do objeto é descrita diretamente. Em geral utilizamos malhas de triângulos para fazer esta descrição.



(a) implícita (b) pela fronteira

Fig. 5.10 – Formas de se descrever a geometria de um objeto.

Esferas

A determinação da interseção de um raio $\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$ com uma esfera de centro $\mathbf{c}$ e raio r pode ser obtida procurando-se os valores t_i que satisfaçam a equação implícita da esfera

$$\|\mathbf{p}(t_i) - \mathbf{c}\|^2 = r^2 \tag{5.11}$$

como ilustra a Fig. 5.11.

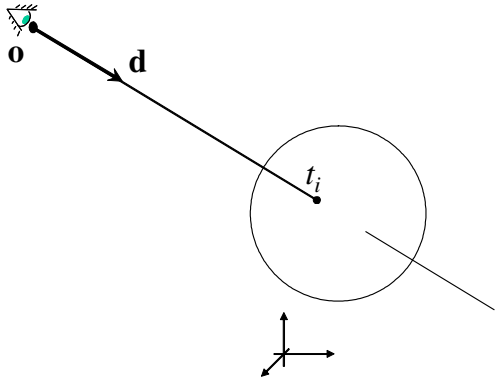


Fig. 5.11 – Interseção raio-esfera.

Substituindo a expressão do raio temos:

$$\|\mathbf{o} + t_i\mathbf{d} - \mathbf{c}\|^2 = r^2 \tag{5.12}$$

Escrevendo a norma como um produto interno chegamos a:

$$\big((\mathbf{o} - \mathbf{c}) + t_i\mathbf{d}\big) \cdot \big((\mathbf{o} - \mathbf{c}) + t_i\mathbf{d}\big) = r^2 \tag{5.13}$$

Expandindo e re-agrupando os termos, temos a expressão

$$\big[\mathbf{d} \cdot \mathbf{d}\big] t_i^2 + \big[2\mathbf{d} \cdot (\mathbf{o} - \mathbf{c})\big] t_i + \big[(\mathbf{o} - \mathbf{c}) \cdot (\mathbf{o} - \mathbf{c}) - r^2\big] = 0 \tag{5.14}$$

que é uma equação do segundo grau em t_i da forma:

$$a\, t_i^2 + b\, t_i + c = 0 \tag{5.15}$$

onde:

$$a = \mathbf{d} \cdot \mathbf{d} \tag{5.16a}$$

$$b = 2\mathbf{d} \cdot (\mathbf{o} - \mathbf{c}) \tag{5.16b}$$

$$c = (\mathbf{o} - \mathbf{c}) \cdot (\mathbf{o} - \mathbf{c}) - r^2 \tag{5.16c}$$

A solução desta equação tem a forma:

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{5.17}$$

que depende do valor do discriminante

$$\Delta = b^2 - 4ac \tag{5.18}$$

Se ele for menor que zero, a solução quadrática não tem solução, o que significa que o raio não intercepta a esfera. Se for maior ou igual a zero, o raio intercepta a esfera em t_i, dado por

$$t_i = \min(t_1, t_2) \tag{5.19}$$

onde:

$$\begin{aligned} t_1 &= \frac{-b - \sqrt{\Delta}}{2a} \\ t_2 &= \frac{-b + \sqrt{\Delta}}{2a} \end{aligned} \tag{5.20}$$

Note que o valor do discriminante igual a zero corresponde à situação geométrica na qual o raio apenas tangencia a esfera. Ou seja, este ponto está no contorno da esfera vista pelo centro de projeção.

Dado o valor da coordenada t_i a determinação do ponto de interseção pode ser feita através da equação do raio. Ou seja:

$$\mathbf{p}_i = \mathbf{p}(t_i) = \mathbf{o} + t_i \mathbf{d} \tag{5.21}$$

A determinação da normal ao ponto de interseção é bastante simples no caso da esfera. A normal em qualquer ponto da esfera está na direção que vai do seu centro até ele, como ilustra a Fig. 5.12. Dado o ponto $\mathbf{p}_i$ o unitário na direção da normal é, então:

$$\hat{\mathbf{n}}_i = \frac{1}{\|\mathbf{p}_i - \mathbf{c}\|} (\mathbf{p}_i - \mathbf{c}) \tag{5.22}$$

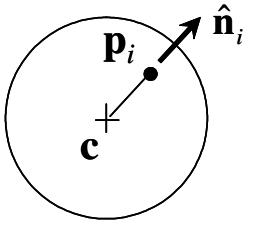


Fig. 5.12 – Normal na esfera.

Triângulos

A determinação da interseção de um raio $\mathbf{p}(t) = \mathbf{o} + t\mathbf{d}$ com um triângulo de vértices $\mathbf{p}_1\mathbf{p}_2\mathbf{p}_3$ pode ser obtida procurando-se primeiramente o valor t_i em que o raio atinge o plano que contém o triângulo, como ilustra a Fig. 5.13. Numa segunda etapa devemos determinar se este ponto é ou não interior ao triângulo.

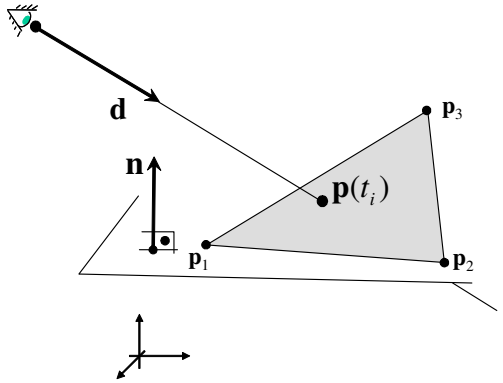


Fig. 5.13 – Interseção raio-triângulo.

A ordem dos vértices pode servir para definir o lado visível da superfície que contém o triângulo. Em objetos sólidos a superfície de fronteira tem dois lados: um que aponta para o exterior e portanto é visível e outro que aponta para dentro do objeto. Adotamos aqui a ordem trigonométrica para indicar o lado externo. Ou seja: no lado visível, quando caminhamos nas arestas visitando os vértices na ordem 1, 2 e 3, os pontos interiores ficam à esquerda. Nesta ordem, a normal que aponta para o lado visível pode ser calculada pela equação:

$$\hat{\mathbf{n}} = \frac{(\mathbf{p}_2 - \mathbf{p}_1)\times(\mathbf{p}_3 - \mathbf{p}_2)}{\left\|(\mathbf{p}_2 - \mathbf{p}_1)\times(\mathbf{p}_3 - \mathbf{p}_2)\right\|} \tag{5.23}$$

Um ponto que pertença ao raio e esteja no plano do triângulo deve satisfazer a seguinte condição de ortogonalidade:

$$\left(\mathbf{p}(t_i) - \mathbf{p}_1\right)\cdot\hat{\mathbf{n}} = 0 \tag{5.24}$$

Substituindo a equação do raio temos:

$$\left(\mathbf{o} + t_i\mathbf{d} - \mathbf{p}_1\right)\cdot\hat{\mathbf{n}} = 0 \tag{5.25}$$

Expandindo o produto interno chegamos a:

$$t_i\mathbf{d}\cdot\hat{\mathbf{n}} + \left(\mathbf{o} - \mathbf{p}_1\right)\cdot\hat{\mathbf{n}} = 0 \tag{5.26}$$

Que fornece o valor de t_i para todos os raios que não forem paralelos ao triângulo:

$$t_i = \frac{(\mathbf{p}_1 - \mathbf{o}) \cdot \hat{\mathbf{n}}}{\mathbf{d} \cdot \hat{\mathbf{n}}} \tag{5.27}$$

Quando o raio for paralelo, $\mathbf{d} \cdot \hat{\mathbf{n}} = 0$, consideramos que ele não pode interceptar o triângulo.

Com base no valor de t_i chegamos ao ponto de interseção do raio com o plano que contém o triângulo, dado por:

$$\mathbf{p}_i = \mathbf{p}(t_i) = \mathbf{o} + t_i \mathbf{d} \tag{5.28}$$

A verificação de se o ponto achado é ou não interior ao triângulo pode ser feita de diversas maneiras. Uma delas consiste em verificar se ele está sempre à esquerda das arestas. Se sempre estiver à esquerda é interior, se estiver à direita de alguma delas é exterior. A Fig. 5.14 ilustra esta idéia. O ponto $\mathbf{p}_i^{int}$ está à esquerda da aresta $\mathbf{v}_{12}$ porque:

$$\hat{\mathbf{n}} \cdot \left(\mathbf{v}_{12} \times (\mathbf{p}_i^{int} - \mathbf{p}_1) \right) > 0 \tag{5.29a}$$

O ponto $\mathbf{p}_i^{ext}$, também mostrado na figura, é exterior ao triângulo, porque:

$$\hat{\mathbf{n}} \cdot \left(\mathbf{p}_{12} \times (\mathbf{p}_i^{ext} - \mathbf{p}_1) \right) < 0 \tag{5.29b}$$

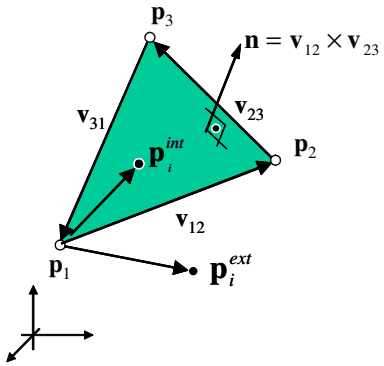


Fig. 5.14 – Teste de interior pela orientação das arestas do triângulo.

Uma outra maneira de verificar se um ponto é ou não interior a um triângulo pode ser obtida calculando-se as coordenadas naturais ou baricêntricas dele. Como ilustra a Fig. 5.15, as áreas de cada um dos três sub-triângulos A_1, A_2 e A_3 podem ser calculadas por:

$$\begin{aligned} A_1 &= \hat{\mathbf{n}} \cdot \left(\mathbf{v}_{23} \times (\mathbf{p}_i - \mathbf{p}_2) \right) / 2 \\ A_2 &= \hat{\mathbf{n}} \cdot \left(\mathbf{v}_{31} \times (\mathbf{p}_i - \mathbf{p}_3) \right) / 2 \\ A_3 &= \hat{\mathbf{n}} \cdot \left(\mathbf{v}_{12} \times (\mathbf{p}_i - \mathbf{p}_1) \right) / 2 \end{aligned} \tag{5.30}$$

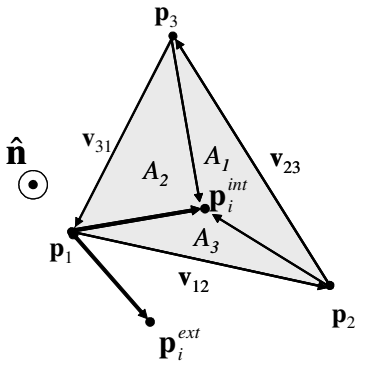


Fig. 5.15 – Teste de interior pelas coordenadas baricêntricas.

De posse destas áreas as coordenas baricêntricas são obtidas por:

$$L_1 = A_1 / A_T$$
$$L_2 = A_2 / A_T \qquad (5.31)$$
$$L_3 = A_3 / A_T$$

onde:

$$A_T = A_1 + A_2 + A_3$$

O ponto $\mathbf{p}_i$ é interior se L_1, L_2 e L_3 são todos maiores ou iguais a zero e menores ou iguais a um. Note que a soma das coordenadas baricêntricas é sempre um e que se uma coordenadas for negativa o ponto é, necessariamente, exterior.

Caixa alinhada com os eixos

Outro objeto elementar consiste numa caixa alinhada com os eixos definida pelos vértices de coordenadas mínimas e máximas, como ilustra a Fig. 5.16.

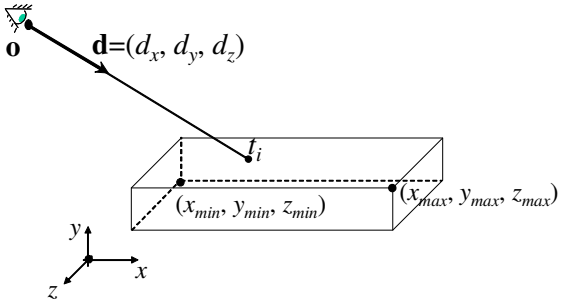


Fig. 5.16 – Caixa alinhada com os eixos.

O cálculo da interseção pode ser feito observando que, dadas as componentes d_x, d_y, d_z do raio, a caixa tem até três faces para o raio possivelmente entrar na caixa e até três faces para ele possivelmente sair.

Se $d_x>0$ o raio pode entrar pela face $x=x_{min}$ e sair pela face $x=x_{max}$. Se $d_x<0$ o raio pode entrar por $x=x_{max}$ e sair por $x=x_{min}$. Se $d_x=0$ as faces $x=x_{min}$ e $x=x_{max}$ não são passíveis de serem interceptadas pelo raio. O mesmo ocorre com as demais coordenadas.

Uma maneira simples de calcular o ponto de interseção de um raio com uma caixa alinhada consiste em determinar o ponto de entrada em x da forma indicada acima e, então, calcular as demais coordenadas e verificar se elas estão no intervalo $[y_{min}, y_{max}]\times [z_{min}, z_{max}]$. Caso estejam o ponto de interseção foi determinado. Caso contrario repetimos a tentativa em y e z.

5.4 Modelos de Iluminação

No algoritmo de Rastreamento de Raios discutido até agora, lançamos um raio para determinar a cor de um *pixel*. As seções anteriores tratam da determinação do ponto do primeiro objeto que é interceptado pelo raio. Esta seção discute da determinação da cor deste ponto para ela seja atribuída ao *pixel* correspondente.

A determinação fisicamente correta da energia luminosa que emana de um ponto na superfície de um objeto e chega até uma câmera fotográfica é um problema bastante complexo. Na base deste problema está a forma com que as superfícies refletem a luz que incide sobre elas, como a luz refletida de uma superfície afeta as outras e quanto dela chega à câmera. Os modelos utilizados na Computação Gráfica são geralmente aproximações computacionalmente eficientes que produzem bons resultados visuais.

Esta seção apresenta um modelo simples que foi primeiramente proposto na tese de doutorado de Bui Tuong Phong, na Universidade de Utah, em 1973, e por isto é conhecido na literatura como **modelo de iluminação de Phong**.

Para motivar o desenvolvimento de modelos de iluminação, propomos o problema de gerar uma imagem da cena simples, composta de uma esfera azul e duas caixas finas amarelas, que está ilustrada na Fig. 5.17.

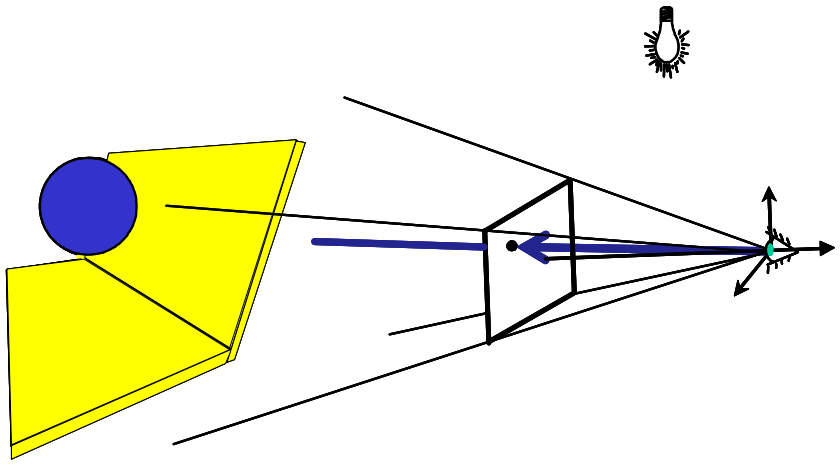


Fig. 5.17 – Esquema de uma cena que contém uma esfera e duas caixas finas.

Uma descrição textual inicial desta cena seria algo como a mostrada no Quadro 5.3.

Cena:
cor de fundo = (0.43,0.43,0.43)

Câmera:
eye = (100,40,40), **center** = (0,0,0), **up**=(0,1,0), near = 30, far=230,
w=230, *h*=230.

Esfera:
c = (0,20,0), *r* = 25, cor azul = (0,0,1)

Caixas alinhadas com os eixos:
$\mathbf{p}_0$ = (-80,-50,-50), $\mathbf{p}_1$ = (50,-45,50) e cor amarela = (0.7,0.7,0)
$\mathbf{p}_0$ = (-80,-50,-60), $\mathbf{p}_1$ = (50,50,-50) e cor amarela = (0.7,0.7,0)

Luz Pontual:
Posição=(60,120,40) e intensidade RGB *l*=(0.8,0.8,0.8)

Quadro 5.3 – Descrição inicial de uma cena com duas caixas e uma esfera.

A cor de fundo é simplesmente a cor que devemos atribuir aos raios que não atingirem nenhum objeto, simulando uma espécie de pano de fundo da cena.

Cor sem iluminação

Se não considerarmos a interação da luz com os objetos e simplesmente atribuirmos uma cor correspondente à cor do material, teríamos uma imagem chapada e sem realismo. A Fig. 5.18 mostra o resultado do algoritmo colocando simplesmente a cor do objeto mais próximo interceptado pelo raio no *pixel* correspondente.

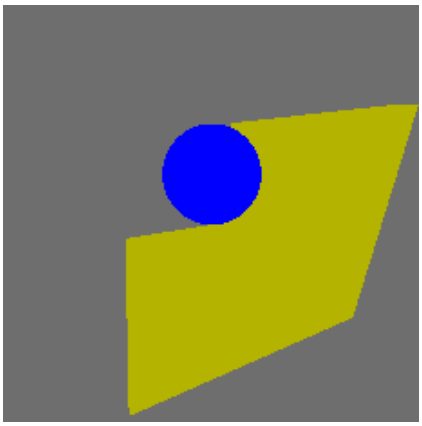


Fig. 5.18 – Imagem sem iluminação.

Podemos notar que os cálculos algébricos feitos com os modelos de câmera e de objetos, apresentados acima, resultam em uma imagem que tem a geometria e a oclusão de uma foto real. A esfera está à frente das caixas, o lado da caixa próximo à câmera aparece maior que o distante, e as linhas paralelas convergem para o ponto de fuga, tudo exatamente como esperado. A falta da interação da luz com o objeto, entretanto, confunde a forma dos objetos. A esfera fica parecendo um disco e as duas caixas se fundem num objeto de forma indeterminada.

Reflexão difusa

A cor que vemos num ponto de um objeto depende de vários fatores. O mais básico deles é a posição relativa entre a normal ao objeto naquele ponto e a direção de onde vem a luz. Superfícies iluminadas diretamente refletem mais luz que as superfícies iluminadas lateralmente. A Fig. 5.19 ilustra o conceito de área aparente no qual a luz refletida varia com o co-seno do ângulo entre a normal da superfície.

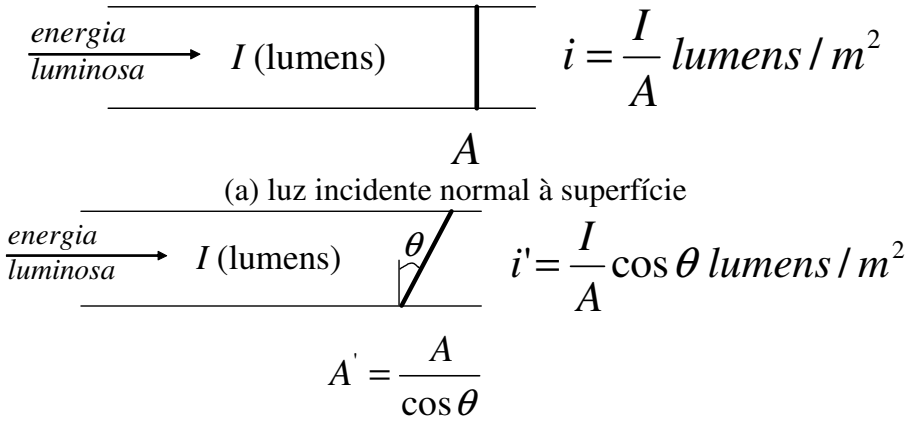


Fig. 5.19 – Área aparente.

Se olharmos para as superfícies à nossa volta vamos notar que, nas superfícies foscas, a cor praticamente não depende da posição do observador. Ou seja, a luz incidente reflete igualmente em todas as direções. As superfícies que satisfazem estes dois postulados são ditas **superfícies lambertianas** e o modelo de reflexão é chamado de **reflexão difusa**. A Fig. 5.20 enfatiza os dois postulados das superfícies Lambertianas: a reflexão se dá uniformemente em todas as direções e a intensidade é reduzida pelo co-seno do ângulo entre e luz e a normal à superfície.

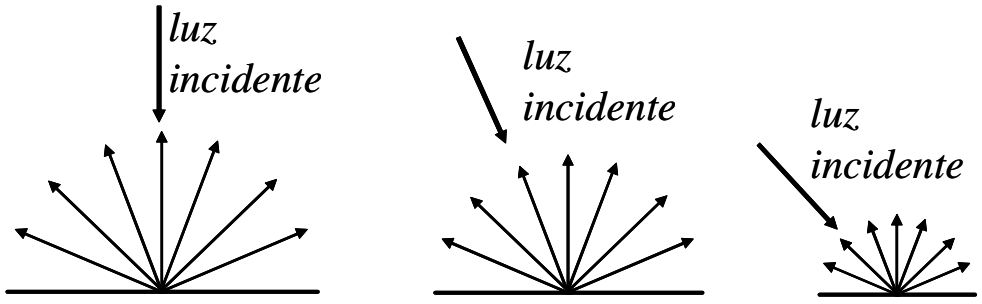


Fig. 5.20 – Reflexão em superfície Lambertiana.

No modelo de reflexão difusa ainda falta uma discussão sobre o matiz da cor refletida. No nosso cotidiano sabemos que se a luz é branca, ou seja, contém todas as componentes do espectro, o matiz da luz refletida é a das superfícies. Ou seja, uma superfície verde aparece como verde se exposta à luz branca. Se a luz não tiver componente azul, por exemplo, a bola azul de nosso exemplo deveria aparecer preta. Este modelo de reflexão é discutido no Capítulo 2, com relação a filtros ou processos subtrativos de formação de cores.

Toda a discussão do modelo de reflexão difusa pode ser formalizada na seguinte equação, que define a intensidade das componentes RGB da luz refletida como sendo:

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} l_r k_{dr} \cos\theta \\ l_g k_{dg} \cos\theta \\ l_b k_{db} \cos\theta \end{pmatrix} \tag{5.32a}$$

onde $(l_r,l_g,l_b)^T$ são as **intensidades RGB da luz**, $(k_{dr}, k_{dg}, k_{db})^T$ é chamada a **cor difusa do material** e θ é o ângulo entre a normal e a direção da luz. Na literatura da Computação Gráfica é comum encontramos esta equação escrita como:

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} l_r k_{dr} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) \\ l_g k_{dg} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) \\ l_b k_{db} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) \end{pmatrix} = \begin{pmatrix} l_r k_{dr} \\ l_g k_{dg} \\ l_b k_{db} \end{pmatrix} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) = \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) \tag{5.32b}$$

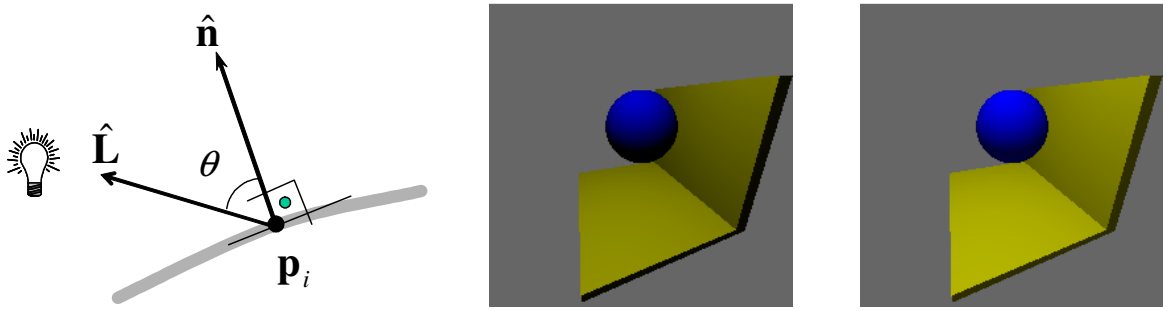
onde $\otimes$ representa um produto componente a componente, como mostra a fórmula. A Fig. 5.21 mostra o resultado da aplicação do modelo de iluminação difusa no exemplo desta seção. A Fig. 5.21a ilustra o ângulo θ e os vetores $\hat{\mathbf{n}}$ e $\hat{\mathbf{L}}$. Este último é o vetor unitário que, a partir do ponto de interseção do raio, aponta para a posição da luz.

A Fig. 5.21b ilustra a aplicação da equação (5.32) no exemplo desta seção. Apesar de não ser muito marcante no exemplo, podemos notar que na Fig. 5.21b a cor das caixas varia de um ponto a outro, uma vez que a luz está próxima ao objeto e o vetor $\hat{\mathbf{L}}$ varia de direção. Se a luz estivesse muito longe, este raio-vetor seria constante e a cor das superfícies planas seria uniforme.

Nem toda a luz que chega a uma superfície vem diretamente da fonte de luz. Uma parcela dela vem de reflexões múltiplas entre os objetos da cena. Para levar em conta este efeito seria necessário utilizarmos um método de iluminação global. Tais métodos são computacionalmente caros, uma vez que eles se baseiam no balanço da radiosidade de todas as superfícies da cena. Uma forma amplamente utilizada na Computação Gráfica de incorporar este efeito consiste em acrescentar para todos os pontos da cena uma **luz ambiente**, $(I_{ar}, I_{ag}, I_{ab})^T$, que independe de posição e normal. Com este efeito a equação de iluminação no nosso exemplo ficaria:

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} + \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) \tag{5.33}$$

A Fig. 5.21c ilustra a inclusão de uma luz ambiente cujas componentes RGB são (0.2,0.2,0.2).



(a) Relação da normal com a luz (b) Iluminação difusa (c) Ambiente mais difusa

Fig. 5.21 – Modelo de reflexão difusa ou Lambertiana.

Reflexão especular

A inclusão do modelo de reflexão difusa nos permite uma clara idéia da forma dos objetos, como ilustra a Fig. 5.21b, mas nem todas as superfícies são foscas como preconiza o modelo lambertiano. Um material metálico polido, por exemplo, não pode ser bem representado apenas com este modelo. Isto porque a luz reflete em superfícies polidas e, dependendo de sua posição com relação à fonte luminosa, um observador vê pontos brilhantes. A não inclusão destes pontos tira a nossa capacidade de identificar visualmente o tipo de material e, por isto, compromete o realismo da imagem. Nestes pontos brilhantes a superfície age como um espelho da luz e é esta a idéia do modelo mostrado na Fig. 5.22. Se a luz reflete na superfície como um espelho, os pontos brilhantes são vistos em torno da direção refletida $\hat{\mathbf{r}}$ mostrada na Fig. 5.22a. Esta direção é obtida refletindo-se o vetor que aponta para a fonte luminosa, $\hat{\mathbf{L}}$, em torno da normal $\hat{\mathbf{n}}$. Os pontos brilhantes estão situados nos raios para os quais o valor do ângulo α é pequeno, ou seja, para pontos em que o vetor de reflexão $\hat{\mathbf{r}}$ está próximo do vetor $\hat{\mathbf{v}}$. Este é o vetor unitário que, a partir do ponto de interseção, $\mathbf{p}_i$, aponta para a posição do centro de projeção, ***eye*** .

Um modelo matemático simples para esta reflexão especular pode ser dado por:

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix}_{especular} = \begin{pmatrix} l_r k_{sr} (\cos\alpha)^n \\ l_g k_{sg} (\cos\alpha)^n \\ l_b k_{sb} (\cos\alpha)^n \end{pmatrix} \tag{5.34}$$

onde n é um coeficiente, dependente do material, que caracteriza o maior ou menor espalhamento do ponto brilhante (veja a Fig. 5.22b).

A equação (5.34) também inclui um modelo para o matiz da luz do ponto brilhante. Assim como na reflexão difusa, a superfície na reflexão especular age como um filtro sobre a luz incidente. A redução das componentes RGB é especificada pelos coeficientes $(k_{sr}, k_{sg}, k_{sb})^T$. Note que os coeficientes especulares são independentes dos coeficientes difusos, uma vez que nas cenas que observamos no nosso cotidiano o brilho nas superfícies especulares tem cor distinta da cor difusa. É comum vermos, por exemplo, pontos brilhantes brancos em superfícies coloridas.

Incorporando a equação (5.34) na equação (5.33):

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} + \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) + \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{sr} \\ k_{sg} \\ k_{sb} \end{pmatrix} \left(\hat{\mathbf{r}} \cdot \hat{\mathbf{v}}\right)^n \qquad (5.35)$$

A Fig. 5.22c mostra o resultado da aplicação desta equação no exemplo deste capítulo, cuja propriedades da cena estão resumidas no Quadro 5.4.

Cena:

cor de fundo = (0.4,0.4,0.4), I_a=(0.2,0.2,0.2)

Câmera:

eye = (100,40,40), **center** = (0,0,0), **up**=(0,1,0), near = 30, far=230, w=230, h=230.

Esfera:

c = (0,20,0), r = 25, k_d = (0,0,1), k_s=(1,1,1) e n = 50

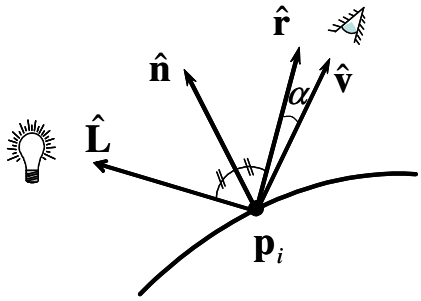
Caixas alinhadas com os eixos:

$\mathbf{p}_0$ = (-80,-50,-50), $\mathbf{p}_1$ = (50,-45,50), k_d = (0.7,0.7,0), k_s=(1,1,1) e n = 40
$\mathbf{p}_0$ = (-80,-50,-60), $\mathbf{p}_1$ = (50,50,-50), k_d = (0.7,0.7,0), k_s=(1,1,1) e n = 40

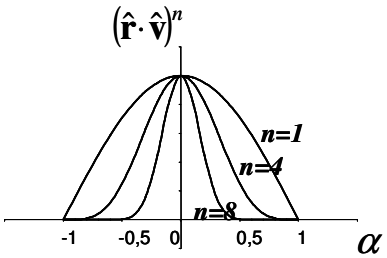
Luz Pontual:

Posição=(40,120,0) e intensidade RGB l=(0.8,0.8,0.8)

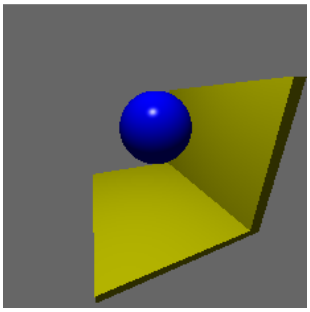
Quadro 5.4 – Descrição de uma cena com duas caixas e uma esfera com iluminação de Phong.



(a) Reflexão especular



(b) Modelo de decaimento



(c) Iluminação difusa e especular

Fig. 5.22 - Modelo de iluminação especular.

Cena com várias luzes

Se uma cena possuir várias fontes de luz, a equação (5.35) pode ser simplesmente estendida para:

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} + \sum_{luzes} \left(\begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) + \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{sr} \\ k_{sg} \\ k_{sb} \end{pmatrix} \left(\hat{\mathbf{r}} \cdot \hat{\mathbf{v}}\right)^n \right) \tag{5.36}$$

Dependendo da escolha dos dados de uma cena, os valores de intensidade calculados por este somatório podem resultar em valores maiores que 1.0. Como, por hipótese, as intensidades variam de 0 a 1, é comum as implementações do algoritmo de Traçado de Raios simplesmente limitarem o resultado a 1. Ou seja, se elas resultam em (1.3, 1.0, 0.9), por exemplo, o algoritmo limita para (1.0, 1.0, 0.9). Estas ações resultam em áreas brancas na imagem final, a exemplo do que aconteceria numa fotografia super-exposta.

Outras ações poderiam ser tomadas para manter as componentes RGB no intervalo [0,1]. Poderíamos, por exemplo, reduzir todas as componentes de todos os *pixels* das imagens de forma que maior delas seja 1.0. Esta ação evitaria a super-exposição do foto virtual, mas seria computacionalmente mais cara, além de prender o resultado da cor de um *pixel* até que toda a imagem fosse calculada.

Na equação (5.36), para cada fonte luminosa, temos que calcular o vetor $\hat{\mathbf{r}}$ corresponde ao vetor $\hat{\mathbf{L}}$ refletido em torno da normal $\hat{\mathbf{n}}$ para então fazer o produto $\hat{\mathbf{r}} \cdot \hat{\mathbf{v}}$. Podemos simplificar o algoritmo refletindo o vetor $\hat{\mathbf{v}}$ em relação à normal $\hat{\mathbf{n}}$ uma única vez e, para cada fonte luminosa, fazer o produto interno entre este vetor e o vetor $\hat{\mathbf{L}}$ correspondente. Ou seja, sendo

$$\hat{\mathbf{r}}_r = 2(\hat{\mathbf{v}} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} - \hat{\mathbf{v}} \tag{5.37}$$

o vetor $\hat{\mathbf{v}}$ refletido em relação à normal $\hat{\mathbf{n}}$ (ver a equação (4.40)), a equação (5.36) pode ser re-escrita como:

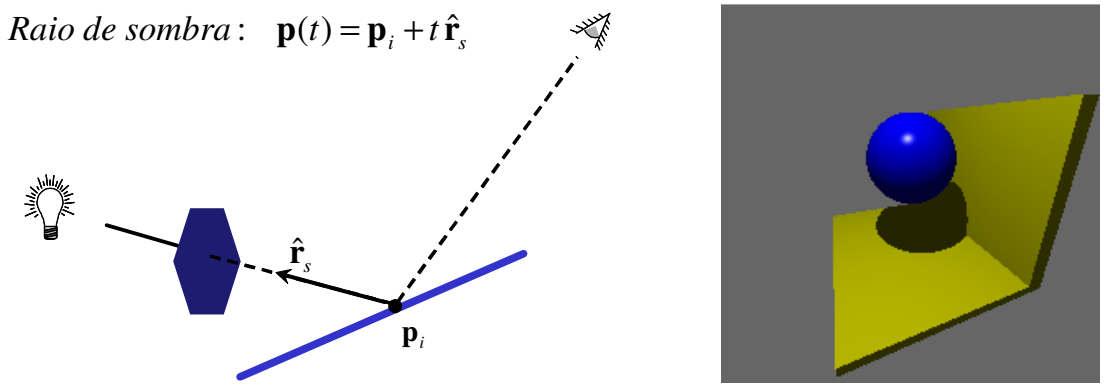
$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} + \sum_{luzes} \left(\begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) + \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{sr} \\ k_{sg} \\ k_{sb} \end{pmatrix} \left(\hat{\mathbf{r}}_r \cdot \hat{\mathbf{L}}\right)^n \right) \tag{5.38}$$

Luz e sombra

O jogo de luz e sombra realça a beleza de muitas paisagens. Incluir o efeito de sombra no algoritmo deste capítulo é fundamental para gerarmos imagens de boa qualidade. Tecnicamente, o fato de uma superfície da cena estar voltada para a posição de uma fonte luminosa não implica em que ela necessariamente receba a luz desta fonte, como supusemos até agora. É possível que exista um objeto opaco entre eles projetando uma sombra sobre o ponto $\mathbf{p}_i$ onde o raio intercepta a superfície do objeto mais próximo.

O algoritmo de Traçado de Raios pode incluir o efeito de sombra, simplesmente lançando um raio do ponto $\mathbf{p}_i$ até a posição da fonte luminosa, como ilustra a Fig. 5.23a. Se neste trajeto o raio intercepta algum objeto opaco, esta fonte não contribui para a iluminação

direta do ponto $\mathbf{p}_i$. A Fig. 5.23b mostra a imagem obtida incluindo-se este efeito no algoritmo aplicado ao exemplo descrito no Quadro 5.4.



(a) Oclusão da luz (b) Iluminação com sombra

Fig. 5.23 - Modelo de iluminação com sombra.

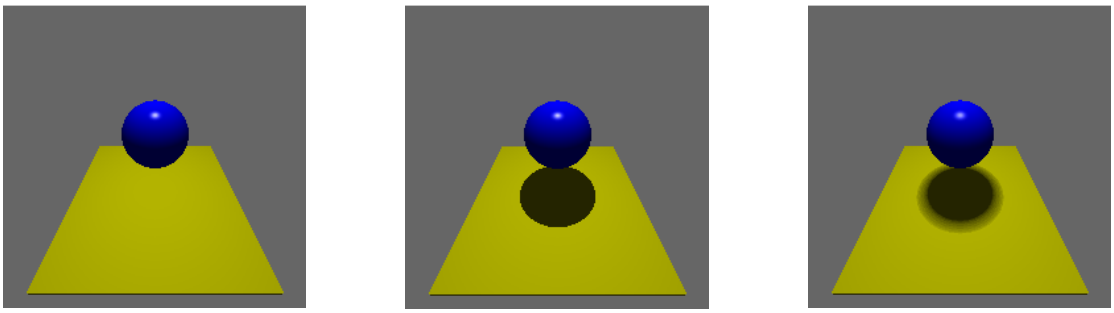
Podemos formalizar este procedimento incluindo na equação (5.36) um fator f_s que multiplica a contribuição difusa e especular de cada fonte luminosa da forma:

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} + \sum_{luzes} f_s \left(\begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) + \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{sr} \\ k_{sg} \\ k_{sb} \end{pmatrix} \left(\hat{\mathbf{r}}_r \cdot \hat{\mathbf{L}}\right)^n \right) \qquad (5.39)$$

Este fator f_s tem o valor igual a zero se existe algum objeto entre o ponto $\mathbf{p}_i$ e a luz pontual, caso contrário ele vale um.

Além da sua importância artística e de realismo visual, a sombra também contribui para incrementar a nossa percepção da posição relativa dos objetos. Considere a Fig. 5.24, em que o centro da esfera está na mesma vertical que o centro da caixa amarela. Sem sombras (Fig. 5.24a) esta posição não é óbvia. A Fig. 5.24b mostra a sombra para a luz pontual colocada na vertical do centro da esfera.

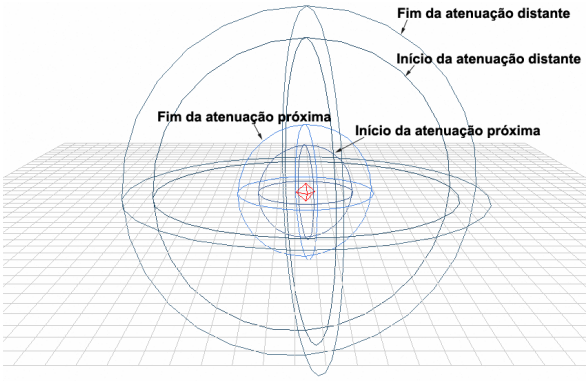
A Fig. 5.24c mostra o efeito de sombra suave, obtido dividindo a fonte luminosa em 16 fontes menores distribuídas num círculo de raio 10 na mesma altura da fonte pontual. Este efeito de sombra suave é comum na natureza devido ao tamanho finito das fontes de luz. No algoritmo de Rastreamento de Raios este efeito pode ser obtido com artifícios como este de aumentar o número de fontes luminosas. Estes artifícios, entretanto, tornam o cálculo de iluminação mais caro computacionalmente.



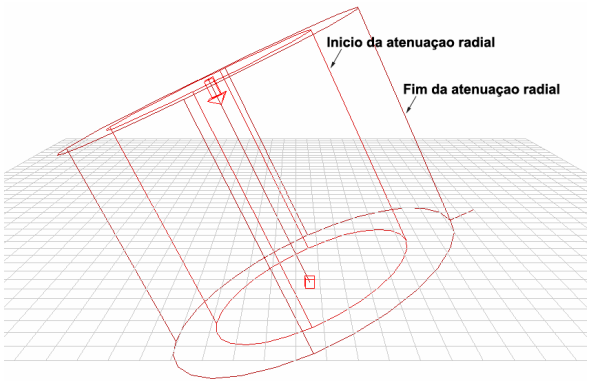
(a) sem sombra (b) sombra de luz pontual (c) sombras suaves

Fig. 5.24 – Efeitos da sombra no posicionamento de objetos e sombra suave.

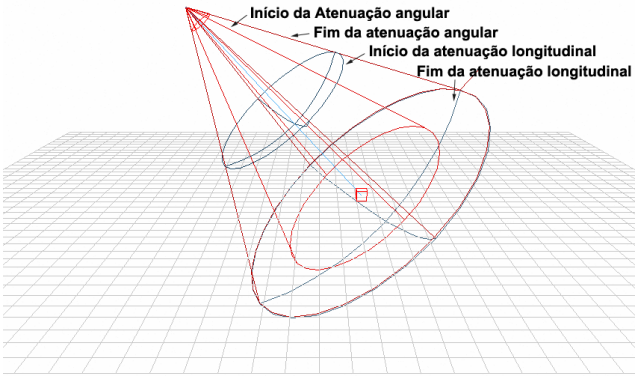
Mesmo se nos restringirmos a luzes pontuais, ainda podemos incluir controles na fonte luminosa que reproduzam os efeitos especiais de iluminação. A Fig. 5.25 ilustra três tipos de fontes luminosas, simples de serem implementadas, que são comuns na Computação Gráfica. A Fig. 2.25a mostra a fonte onidirecional, que é uma fonte de luz pontual com campo de influência em todo o espaço ao redor de sua posição no ambiente virtual, com os raios de luz divergindo a partir da sua posição. A Fig. 5.25b ilustra a fonte direcional, que é uma fonte de luz com raios paralelos. Pode ser modelada como estando posicionada em um ponto no infinito emitindo os raios em uma determinada direção e tendo como campo de influência todo o espaço ou uma região limitada em torno de um eixo que define a direção dos raios. Finalmente a Fig. 5.25c ilustra o farolete, que é um caso particular de uma fonte de luz onidirecional cujo campo de influência é limitado a um cone. Em todas estas fontes o valor da intensidade luminosa é modulado por uma função de atenuação que é função de distâncias e/ou ângulos.



(a) Fonte de luz onidirecional



(b) Fonte de luz direcional



(c) Fonte de luz tipo farolete

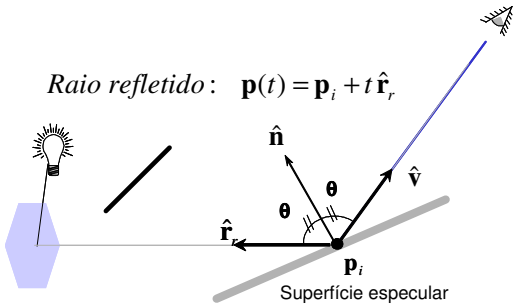
Fig. 5.25 – Fontes luminosas pontuais com efeitos especiais.

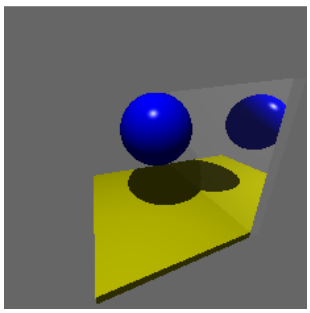
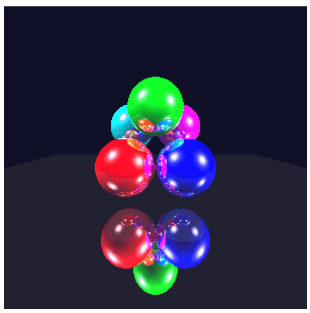
Superfícies refletoras e objetos transparentes

Algumas superfícies agem como espelhos e refletem a luz que incide sobre elas. Outras são fronteiras de objetos transparentes, como o vidro, que nos permitem ver através deles. Efeitos de reflexão e refração geram imagens complexas e interessantes que são exploradas em muitos filmes.

Podemos incorporar superfícies refletoras no algoritmo de Rastreamento de Raios se, ao determinarmos a cor de um ponto $\mathbf{p}_i$ sobre uma superfície refletora, lançarmos outro raio $\hat{\mathbf{r}}_r$ (ver a Fig. 5.26a) e procurarmos a cor do objeto interceptado pelo raio $\mathbf{p}_i + t\hat{\mathbf{r}}_r$. O raio $\hat{\mathbf{r}}_r$ corresponde à reflexão do vetor $\hat{\mathbf{v}}$ em torno da normal $\hat{\mathbf{n}}$ já calculado na equação (5.37). A cor encontrada por este raio é combinada com a cor do ponto $\mathbf{p}_i$.

A Fig. 5.26b ilustra o resultado deste efeito incluído no algoritmo de Rastreamento de Raios e aplicado ao exemplo desta seção se modificarmos a caixa vertical para torná-la uma superfície tipo espelho. A Fig. 5.26c mostra o resultado deste algoritmo para um exemplo de espelho horizontal e 5 bolas coloridas.



(a) raio refletido (b) espelho vertical (b) 5 bolas

Fig. 5.26 – Inclusão de superfícies refletoras.

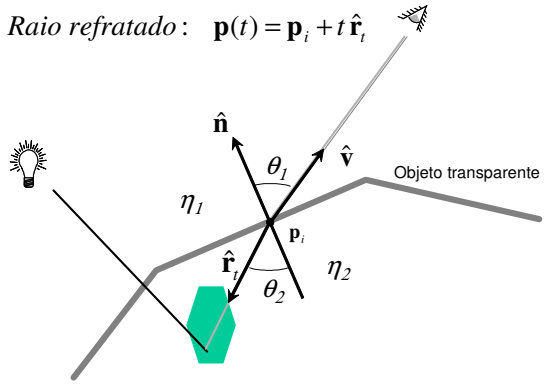
A definição do espelho da Fig. 5.26b é dada pelo Quadro 5.5. Note que, da forma que está definida, a contribuição das componentes difusa e especular é pequena. O fator redutor k multiplica a cor calculada para o raio refletido e soma esta cor na cor corrente do ponto $\mathbf{p}_i$. No caso, nenhuma redução foi aplicada (k=1).

Caixas alinhadas com os eixos: (espelho) $\mathbf{p}_0$ = (-80,-50,-50), $\mathbf{p}_1$ = (50,-45,50), k_d =(0.08,0.08,0.08), k_s=(0.04,0.04,0.04), n = 40 e k=1

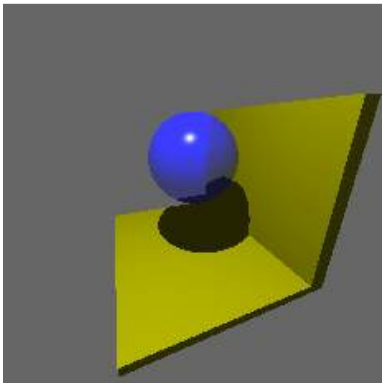
Quadro 5.5 – Descrição do espelho na Fig. 5.26b.

Analogamente, podemos incorporar o efeito de objetos transparentes no algoritmo de Rastreamento de Raios se, ao determinarmos a cor de um ponto $\mathbf{p}_i$ sobre a superfície de um objeto transparente, lançarmos outro raio $\hat{\mathbf{r}}_t$ (ver Fig. 5.27a) e procurarmos a cor do objeto interceptado pelo raio $\mathbf{p}_i + t\hat{\mathbf{r}}_t$. A cor encontrada por este raio é combinada com a cor do ponto $\mathbf{p}_i$.

A Fig. 5.27b ilustra o resultado da inclusão do efeito de transparência no algoritmo de Rastreamento de Raios no exemplo desta seção se modificarmos a esfera para torná-la transparente.



(a) Raio refratado



(b) Esfera transparente

Fig. 5.27 – Modelo de iluminação com objetos transparentes.

O raio refratado pode ser calculado utilizando a lei de Snell (1691) da Física, que diz que a razão entre os senos dos ângulos de refração e incidência é dada por:

$$\frac{\sin\theta_2}{\sin\theta_1} = \frac{\eta_1}{\eta_2} \tag{5.40}$$

onde η_1 e η_2 são os coeficientes de refração do material de onde vem a luz e do objeto transparente. Estes coeficientes correspondem à razão entre a velocidade da luz no vácuo e a velocidade da luz no material 1 e 2, respectivamente. De posse destes coeficientes, que devem fazer parte dos dados do problema, a Fig. 5.28 mostra um esquema para calcular o raio refratado.

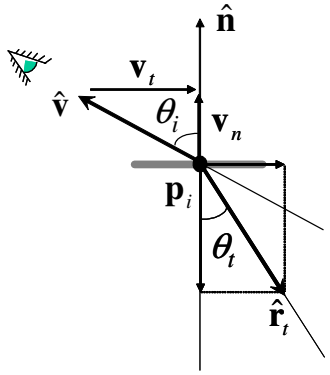


Fig. 5.28 – Cálculo do raio refratado.

Dado o vetor $\hat{\mathbf{v}}$ podemos calcular sua componente tangencial $\hat{\mathbf{v}}_t$ por (ver o Capítulo 4):

$$\mathbf{v}_t = (\hat{\mathbf{v}} \cdot \hat{\mathbf{n}})\hat{\mathbf{n}} - \hat{\mathbf{v}} \tag{5.41}$$

O seno do ângulo incidente pode ser obtido por:

$$\sin\theta_i = \|\mathbf{v}_t\| \tag{5.42}$$

uma vez que o vetor $\hat{\mathbf{v}}$ é unitário. O seno e o co-seno do ângulo refratado podem ser calculados por:

$$\sin\theta_t = \frac{\eta_i}{\eta_t}\sin\theta_i \text{ e } \cos\theta_t = \sqrt{1-\sin^2\theta_t} \tag{5.43}$$

A partir do seno e co-seno do ângulo refratado, o vetor unitário refratado é simplesmente dado por:

$$\mathbf{r}_t = \sin\theta_t\hat{\mathbf{t}} + \cos\theta_t(-\hat{\mathbf{n}}) \tag{5.44}$$

onde $\hat{\mathbf{t}}$ é o unitário da direção tangencial dada por $\hat{\mathbf{v}}_t$, como ilustra a Fig. 5.28.

O Quadro 5.6 mostra as propriedades da esfera transparente da Fig. 5.27b. O fator η é o coeficiente de refração do material da lei de Snell e o fator o indica a opacidade do material, sendo 0 quando é transparente e 1 quando é opaco. A cor obtida pelo raio refratado é multiplicada pelo fator (1-o) antes de ser somada à cor do ponto $\mathbf{p}_i$.

Esfera:
$\mathbf{c}$ = (0,20,0), r = 25, k_d = (0,0,1), k_s=(1,1,1) e n = 50, k = 0, η=1.2 e o=0.4

Quadro 5.5 – Descrição da esfera transparente.

A opacidade o é, em geral, arbitrária mas o valor do coeficiente de refração do material da lei de Snell segue os valores típicos de materiais reais. O Quadro 5.7 ilustra alguns destes valores.

Material	η
ar	1.0
vidro	1.5
água	1.3

Quadro 5.7 – Valores típicos de coeficiente de refração.

Incluindo os efeitos de superfícies refletoras e objetos transparentes na equação (5.39) temos:

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} + \sum_{luzes} f_s \left(\begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} (\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}) + \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{sr} \\ k_{sg} \\ k_{sb} \end{pmatrix} (\hat{\mathbf{r}}_r \cdot \hat{\mathbf{L}})^n \right) + k \begin{pmatrix} I_r(\mathbf{r}_r) \\ I_g(\mathbf{r}_r) \\ I_b(\mathbf{r}_r) \end{pmatrix} + (1-o) \begin{pmatrix} I_r(\mathbf{r}_t) \\ I_g(\mathbf{r}_t) \\ I_b(\mathbf{r}_t) \end{pmatrix} \tag{5.39}$$

A implementação desta equação no algoritmo de Rastreamento de Raios implica em calcularmos as cores dos raios refletidos e refratados antes de determinarmos a cor do ponto $\mathbf{p}_i$. Acontece que, ao buscarmos estas cores, os raios podem atingir outras superfícies refletoras ou transparentes, gerando novos raios, como ilustra a Fig. 5.28. Nesta figura os raios L buscam a luz, os raios R são os refletidos e os T os refratados. A árvore binária mostrada na figura representa esta composição. É comum implementarmos o algoritmo de Rastreamento de Raios da forma recursiva ilustrada no Quadro 5.8. O controle desta recursão é geralmente a altura da árvore ou a profundidade da recursão.

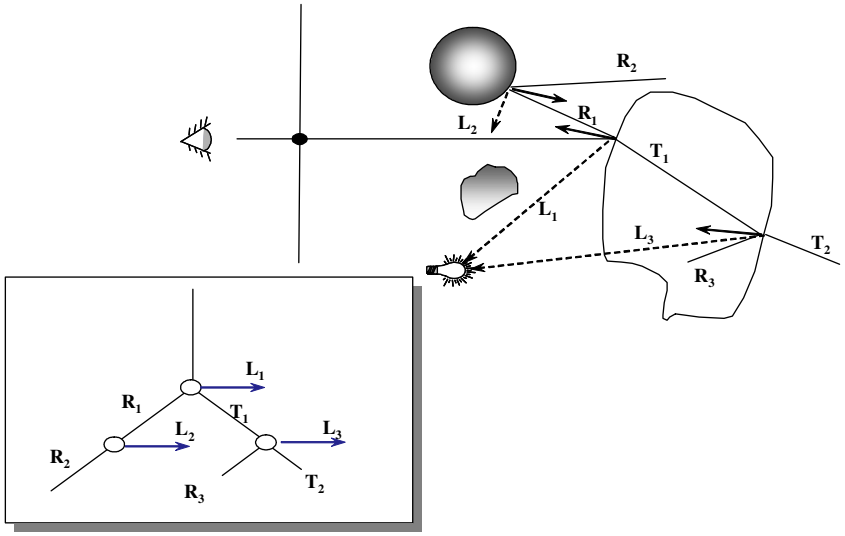


Fig. 5.29 – Natureza recursiva do algoritmo de Rastreamento de Raios.

```
for (cada pixel da tela)
 {
  determine o raio ray correspondente ao pixel em questão;
  pixel = trace ( ray, 1 );
  }
```

```
Color  trace (Scene scene, Vector eye, Vector ray, int depth)
{
 determine a interseção mais próxima com um objeto
 if (intercepta objeto)
 {
  calcule a normal no ponto de interseção
  return ( shade ( scene, object, ray, point, normal, depth ));
  }
 return BACKGROUND;
}
```

```
Color  shade (Scene scene, Object object, Vector ray,
              Vector point, Vector normal,  int  depth)
{
 color = cor ambiente sobre a cor difusa do material do objeto;

 for (cada luz)  {
   L = vetor unitário na direção de point  para a posição da luz;
   if  (L•normal>0) {
      if  (a luz não for bloqueada no ponto) {
          color += componente difusa + componente especular
    }
 }

 if (depth >= maxDepth)  return color;

 if (objeto é refletor) {
    rRay = raio na direção de reflexão;
```

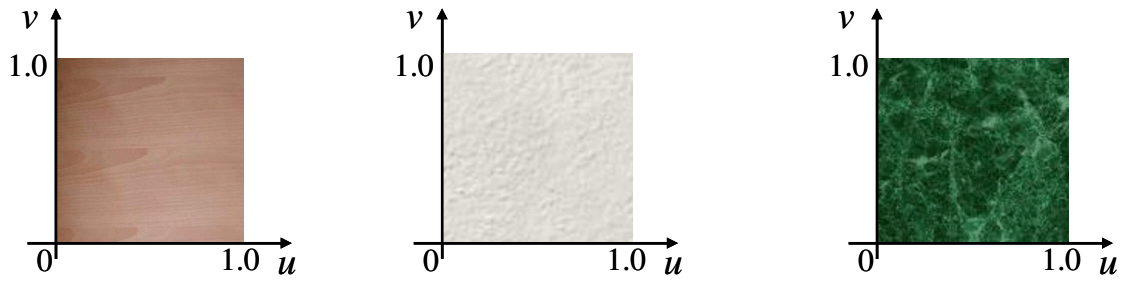
```
    rColor = trace(scene, point, rRay, depth+1);
    color += k* rColor;
 }

 if (objeto é transparente) {
    tRay = raio na direção de refração;
    tColor = trace(scene, point, tRay, depth+1);
    color += (1-o)*tColor;
 }

    return color;
}
```

Quadro 5.8 – Algoritmo recursivo de Rastreamento de Raios.

5.5 Textura

A cor da maioria dos materiais não pode ser bem descrita apenas através do modelo de uma cor difusa e outra especular descrito acima. Superfícies de materiais como o mármore, por exemplo, possuem padrões que para serem descritos necessitam de funções que descrevam como a cor varia de uma posição a outra. Estas funções são chamadas de **funções de textura** e podem ser descritas com base numa reta (1D), num retângulo (2D) ou num cubo (3D). As mais comuns são as texturas 2D.

A Fig. 5.30 ilustra três funções de textura 2D que descrevem a variação das propriedades ópticas de uma fórmica, de uma superfície enrugada e de um mármore verde. Ou seja, para cada par de valores (u,v), $u,v \in [0,1]$ estas funções mapeiam uma cor, dada por (r,g,b) ou (r,g,b,a).



(a) fórmica (b) enrugada (c) mármore

Fig. 5.30 – Texturas 2D.

Uma das maneiras mais comuns de se definir texturas consiste em fornecer uma imagem de um certa resolução $w \times h$. Além de associarmos a textura (0,0) à cor do *pixel* (0,0) e a textura (1,1) à cor do *pixel* (w-1, h-1), necessitamos definir um processo de interpolação para fornecer a textura para qualquer valor real no intervalo $[0,1] \times [0,1]$. Este processo é semelhante à re-definição da resolução de uma imagem discutida no Capítulo 3 e tem os

mesmos problemas de amostragem e reconstrução. As interpolações pela amostra mais próxima, bi-linear, bi-quadrática ou bi-cúbica, podem necessitar de um filtragem prévia para reduzir os problemas de *alias*.

Existem diversas maneiras de atribuir textura à superfície de um objeto. Para a caixa alinhada com os eixos, por exemplo, podemos simplesmente associar a cada face da caixa um sistema de coordenadas (u,v), $u,v \in [0,1]$, da forma mostrada na Fig. 5.31.

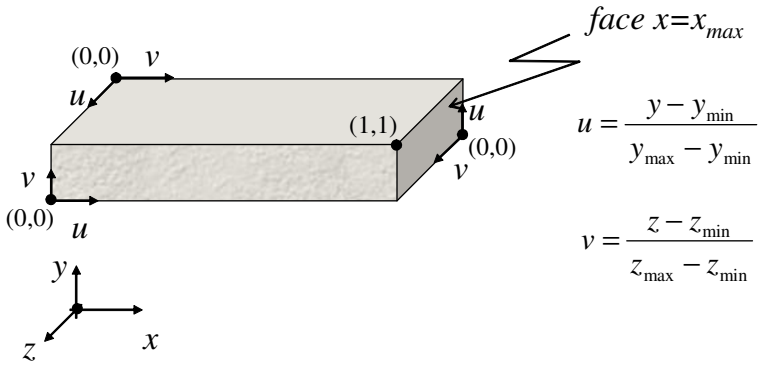


Fig. 5.31 – Sistemas de coordenadas de textura na caixa alinhada.

Para a esfera, por exemplo, podemos associar cada ponto $(x, y, z)^T$ da superfície às suas coordenadas esféricas (ρ, ϕ, θ) de acordo com as fórmulas ilustradas na a Fig. 5.32. Como na superfície da esfera ρ é constante e igual ao raio, as coordenadas de textura podem ser associadas a ϕ e θ. Como os valores de ϕ e θ calculados através da função `atan2` do C estão no intervalo $[-\pi,\pi]$ e $[0,\pi]$, respectivamente, as coordenadas (u,v) podem ser calculadas por:

$$u = \frac{1}{2}\left(1 + \phi/\pi\right) \tag{5.45a}$$

$$v = \theta/\pi \tag{5.45b}$$

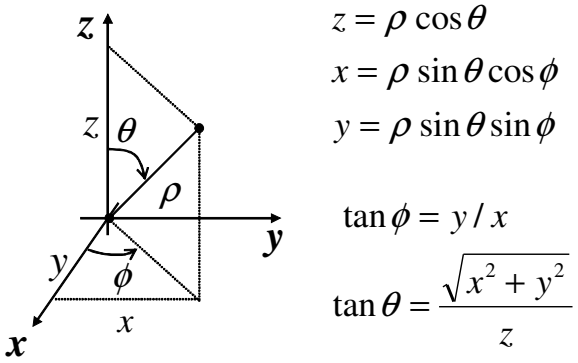


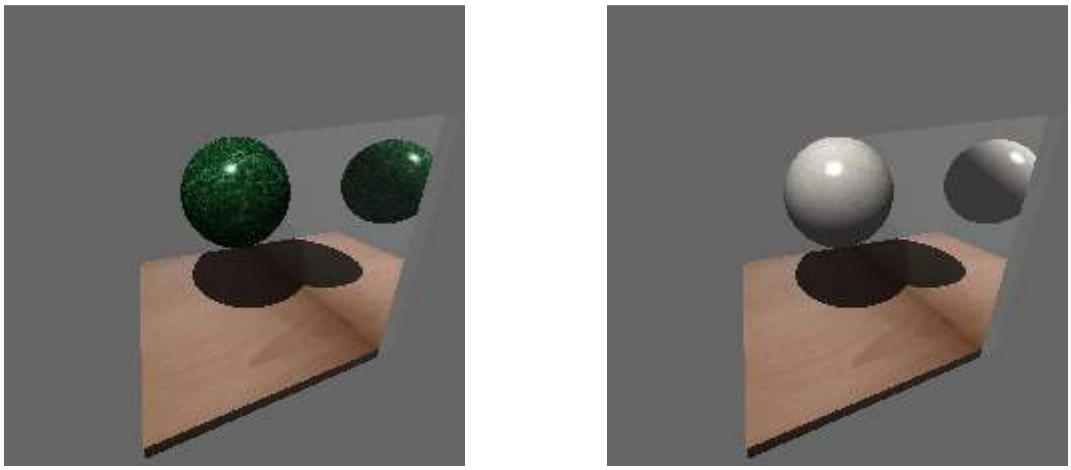
Fig. 5.32 – Sistemas de coordenadas de textura na esfera.

Conhecendo as coordenadas de textura dos pontos de interseção do raio com o objeto, $\mathbf{p}_i$, podemos incluir a informação de textura no cálculo da cor do raio de diversas maneiras, produzindo efeitos variados. Uma destas maneiras seria simplesmente atribuirmos ao raio a

informação (r,g,b) oriunda da textura. Esta estratégia pode dar bons resultados quando a textura já incorpora o cálculo da iluminação.

Outra maneira de determinarmos a cor do raio seria considerarmos a textura como sendo um decalque transparente colocado sobre a cor $(I_r, I_g, I_b)^T$ calculada pelo modelo de Phong apresentado acima (eq. 5.39). Neste caso, se a textura tiver componentes $(r, g, b, a)^T$, a cor resultante poderia ser calculada por um operador de *blending* de cores. O mais comum seria o operador de *over*, através do qual a cor resultante seria uma média ponderada de a vezes a cor da textura mais $(1-a)$ vezes a cor de Phong.

Uma terceira maneira de incorporarmos a informação de textura de um ponto $\mathbf{p}_i$ no cálculo da cor do raio seria utilizarmos as componentes (r, g, b) da textura como sendo a cor difusa do material, ou seja, como sendo os valores de $(k_{dr}, k_{dg}, k_{db})^T$. A Fig. 5.33 mostra o resultado desta estratégia no exemplo de duas caixas e uma esfera.



(a) fórmica e mármore (b) fórmica e enrugada

Fig. 5.33 – Texturas 2D.

A textura 2D aplicada na esfera sofre uma distorção grande nos pólos. A textura aplicada nas caixas também pode ter um problema na junção de uma face com outra. O natural nestes casos seria aplicarmos texturas 3D, em que o valor da cor seria uma função de três coordenadas (t,u,v), $t,u,v \in [0,1]$. Talvez por ainda serem menos disponíveis, as texturas 3D são menos utilizadas atualmente.

Uma outra maneira de especificarmos a textura, particularmente apropriada para domínios 3D é, ao invés de utilizarmos imagens ou valores em grades uniformes, utilizarmos funções randômicas, ou seja, procedimentos não determinísticos para determinarmos valores das componentes (r, g, b) da textura. Dentre estes procedimentos se destaca a proposta de ruído sólido, também chamado de ruído de Perlin em homenagem a seu inventor. Dos trabalhos de Perlin resultaram texturas que produziram efeitos especiais importantes para a indústria cinematográfica – contudo, o estudo destas funções foge ao escopo introdutório deste capítulo.

Existem ainda muitas outras maneiras de utilizarmos a textura no cálculo da cor de um ponto. Entre elas se destacam as **texturas de rugosidade** (*bump textures*), as **texturas de**

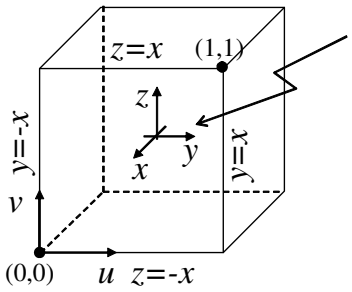
deslocamento (*displacement mapping*) e as **texturas de ambiente** (*environment maps*). As texturas de rugosidade são utilizadas para caracterizar uma variação da normal visando dar ao objeto uma aparência rugosa. Ou seja, mudam a normal e conseqüentemente mudam os cálculos das componentes difusa e especular. A Fig. 5.34a ilustra esta idéia.

Apesar de produzirem resultados razoáveis, as texturas de rugosidade não produzem sombras projetadas nem mudam o perfil do objeto, que continua lisa. Isto porque a geometria do objeto continua inalterada. Uma estratégia mais robusta é a textura de deslocamento. É comum utilizarmos uma textura escalar para mudarmos a posição dos pontos do objeto segundo a direção da normal, produzindo um objeto verdadeiramente rugoso. Este procedimento é fácil de implementarmos em malhas de triângulos, nas quais podemos aplicar a textura de deslocamento nos vértices numa etapa de pré-processamento. Porém, ela é complicada de ser implementada em objetos formulados implicitamente, como a esfera. A Fig. 5.34b ilustra a idéia de textura de deslocamentos.

(a) perturbação das normais (b) deslocamentos das superfícies

Fig. 5.34 – Texturas de rugosidade e deslocamentos.

As texturas de ambiente servem, principalmente, para dar o efeito de fundo nas imagens. A idéia é colocarmos uma caixa ou uma esfera texturizada em volta da cena que estamos renderizando. Dependendo da direção em que a câmera esteja apontando, trechos desta textura aparecem tanto nos raios que não atingem nenhum objeto quanto na reflexão das superfícies especulares da cena. É comum colocarmos a textura ambiente em uma caixa muito grande, de forma que a posição da câmera não importe e a cor da textura ambiente seja determinada apenas pela direção do raio direto ou refletido. A Fig. 5.35 ilustra o cálculo das coordenadas de textura de uma direção em que a coordenada x é maior que y e z.



face dos raios x>y e x>z

$$u = \frac{y + x}{2x}$$

$$v = \frac{z + x}{2x}$$

Fig. 5.35 – Cubo de texturas de ambiente.

5.6 Rastreamento de raios distribuído

No algoritmo descrito neste capítulo ficou implícita a idéia de lançarmos um raio para cada *pixel*. Esta amostragem uniforme não consegue captar bem regiões onde a imagem tem altas freqüências, como as arestas das caixas do exemplo da seção de iluminação. Daí surge o efeito de serrilhado que é particularmente visível em imagens de baixa resolução.

Uma maneira eficaz de combatermos o efeito de *alias* consiste em sub-dividir cada *pixel* em com conjunto de 2×2, 3×3 ou 4×4 *sub-pixels* e lançar 4, 9 ou 16 raios por *pixel* variando a posição de lançamento da forma mostrada na Fig. 5.36. Os deslocamentos dx e dy são variáveis aleatórias. A cor do *pixel* é a média das cores dos raios dos *sub-pixels*.

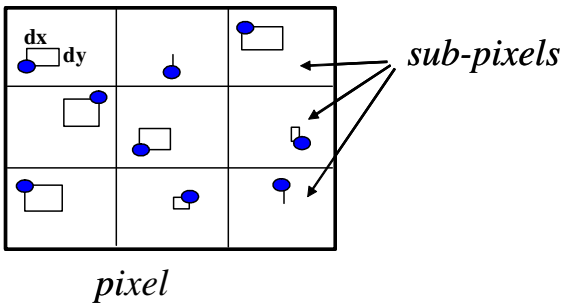
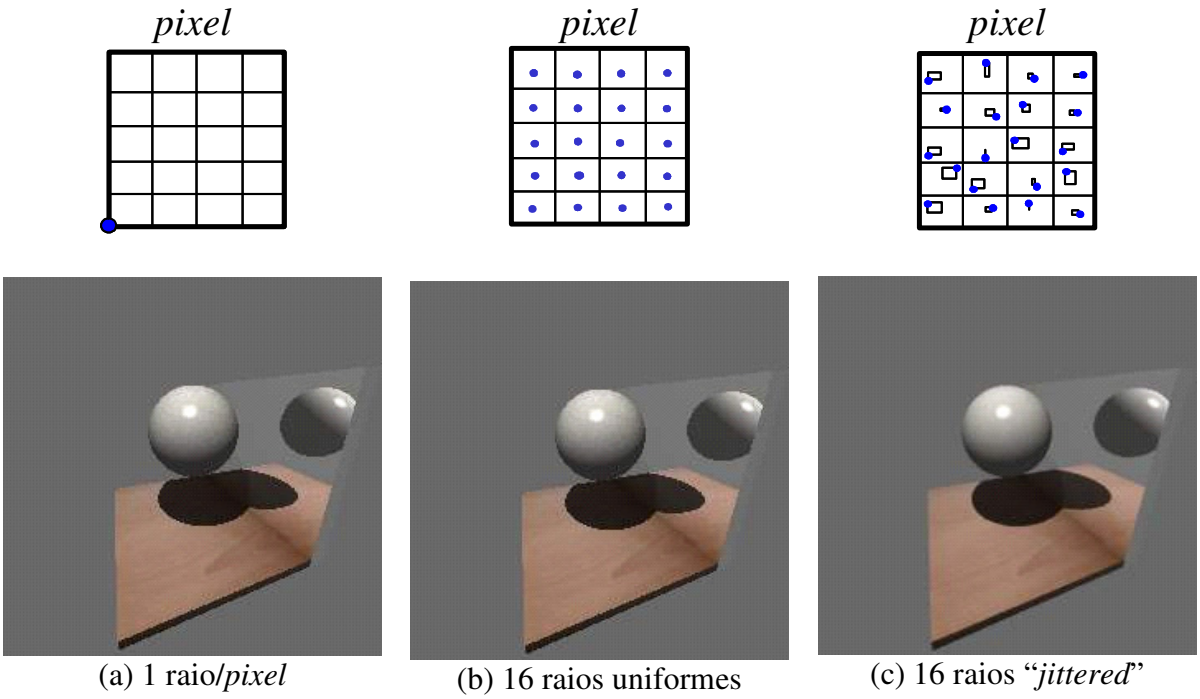


Fig. 5.36 – Tratamento anti-alias.

A subdivisão sem o tratamento aleatório minimiza mas não resolve o problema de *alias*. Ela é simplesmente equivalente a um aumento de resolução da imagem. O importante neste tratamento é o posicionamento aleatório, que desfaz a amostragem uniforme. A Fig. 5.37 ilustra o efeito deste tratamento no exemplo de duas caixas e uma esfera.



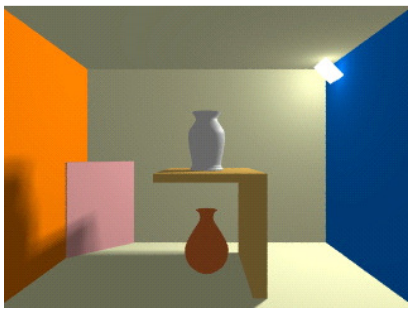
(a) 1 raio/*pixel* (b) 16 raios uniformes (c) 16 raios "*jittered*"

Fig. 5.37 – Efeitos do tratamento anti-alias.

5.7 Modelos de iluminação globais

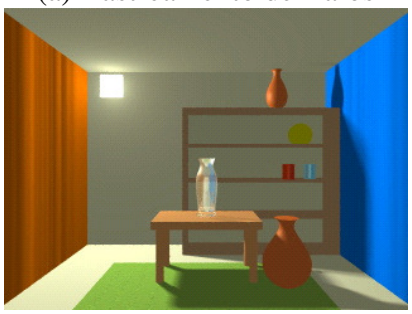
Toda a discussão apresentada anteriormente se baseia no chamado modelo de iluminação local. Isto porque, mesmo com o processo recursivo do algoritmo descrito no Quadro 5.8, o algoritmo não leva em conta a reflexão que existe entre as diversas superfícies que compõem a cena. Ou seja, a luz que chega em um ponto não é apenas aquela que vem diretamente das fontes luminosas. Uma superfície iluminada reflete a luz, que ilumina as demais. Esta interação entre a radiação luminosa de cada ponto de todas as superfícies pode ser computada com métodos globais, como o Método da Radiosidade, que fazem um balanço geral entre as energias geradas, recebidas e emitidas na cena como um todo.

A Fig. 5.38 mostra resultados para duas cenas comparando o Método de Rastreamento de Raios padrão e outro em que a iluminação difusa é calculada através do Método da Radiosidade.

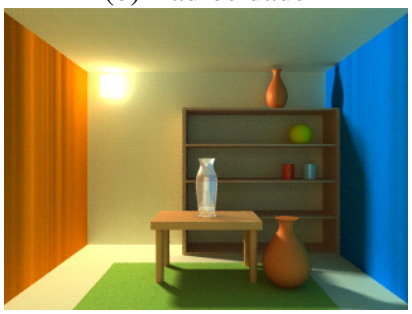


(a) Rastreamento de Raios

(b) Radiosidade



(c) Rastreamento de Raios



(d) Radiosidade

Fig. 5.38 – Radiosidade versus Rastreamento de Raios.
(tirado de Klaus Mueller, Stony Brook University, Computer Science (CSE 564))

5.8 Exercícios resolvidos

1) Determine a componente difusa da luz de uma lâmpada de intensidade (200, 100, 50) colocada na posição homogênea [0, 1, 0, 0]T sobre o ponto A do triângulo ABC. O material da face tem coeficiente kd=(0.6, 0.5, 0.3), o triângulo é orientado no sentido trigonométrico, e as coordenadas do vértice são: A=(3, 2, 1), B=(8, 2, 1) e C=(4, 8, -4).

   Resp.:

   Normal do triângulo:

$$\mathbf{n} = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 8-3 & 2-2 & 1-1 \\ 4-8 & 8-2 & -4-1 \end{bmatrix} = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 5 & 0 & 0 \\ -4 & 6 & -5 \end{bmatrix} = 25\mathbf{j} + 30\mathbf{k}$$

$$\hat{\mathbf{n}} = \frac{1}{\sqrt{25^2 + 30^2}}(25\mathbf{j} + 30\mathbf{k}) \approx \frac{1}{39}(25\mathbf{j} + 30\mathbf{k}) = 0.64\mathbf{j} + 0.768\mathbf{k}$$

$$\hat{\mathbf{L}} \cdot \hat{\mathbf{n}} = \mathbf{j} \cdot (0.64\mathbf{j} + 0.768\mathbf{k}) = 0.64$$

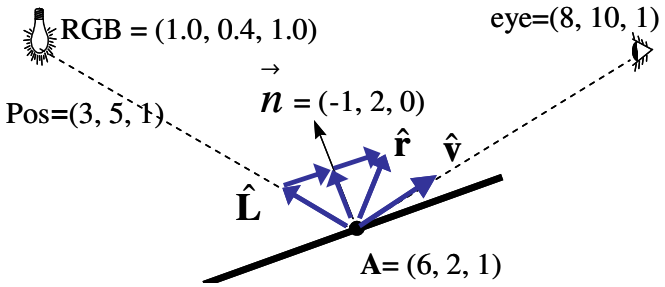
$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = 0.64 \begin{pmatrix} 0.6 \\ 0.5 \\ 0.3 \end{pmatrix} \otimes \begin{pmatrix} 200 \\ 100 \\ 50 \end{pmatrix} = \begin{pmatrix} 76.8 \\ 32 \\ 9.6 \end{pmatrix}$$

2) Determine os vetores $\hat{\mathbf{r}}$ e $\hat{\mathbf{v}}$ da equação:

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} + \sum_{luzes} \left(\begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} (\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}) + \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{sr} \\ k_{sg} \\ k_{sb} \end{pmatrix} (\hat{\mathbf{r}} \cdot \hat{\mathbf{v}})^n \right)$$

correspondentes ao ponto A da cena mostrada na figura abaixo.

Obs.: Desenhe na figura e calcule as coordenadas.



Resp.:

$$\mathbf{v} = \begin{pmatrix} 8 \\ 10 \\ 1 \end{pmatrix} - \begin{pmatrix} 6 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 8 \\ 0 \end{pmatrix} \Rightarrow \boxed{\hat{\mathbf{v}} = \frac{1}{\sqrt{68}} \begin{pmatrix} 2 \\ 8 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{17}} \begin{pmatrix} 1 \\ 4 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.243 \\ 0.970 \\ 0 \end{pmatrix}}$$

$$\hat{\mathbf{n}} = \frac{1}{\sqrt{5}} \begin{pmatrix} -1 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.447 \\ 0.894 \\ 0 \end{pmatrix}$$

$$\mathbf{L} = \begin{pmatrix} 3 \\ 5 \\ 1 \end{pmatrix} - \begin{pmatrix} 6 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -3 \\ 3 \\ 0 \end{pmatrix} \Rightarrow \hat{\mathbf{L}} = \frac{1}{\sqrt{18}} \begin{pmatrix} -3 \\ 3 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.707 \\ 0.707 \\ 0 \end{pmatrix}$$

$$\mathbf{L}_n = (\hat{\mathbf{L}} \cdot \hat{\mathbf{n}}) \cdot \hat{\mathbf{n}},$$

$$\hat{\mathbf{L}} \cdot \hat{\mathbf{n}} = \frac{1}{\sqrt{5}} \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 2 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} = \frac{3}{\sqrt{10}} = 0.949 \Rightarrow \mathbf{L}_n = \frac{3}{5\sqrt{2}} \begin{pmatrix} -1 \\ 2 \\ 0 \end{pmatrix} = \begin{pmatrix} -0.424 \\ 0.849 \\ 0 \end{pmatrix}$$

$$\mathbf{h} = \mathbf{L}_n - \hat{\mathbf{L}} = \frac{1}{5\sqrt{2}} \begin{pmatrix} -3 \\ 6 \\ 0 \end{pmatrix} - \frac{1}{5\sqrt{2}} \begin{pmatrix} -5 \\ 5 \\ 0 \end{pmatrix} = \frac{1}{5\sqrt{2}} \begin{pmatrix} 2 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.283 \\ 0.141 \\ 0 \end{pmatrix}$$

$$\hat{\mathbf{r}} = \mathbf{L}_n + \mathbf{h} = \frac{1}{5\sqrt{2}}\begin{pmatrix}-3\\6\\0\end{pmatrix} + \frac{1}{5\sqrt{2}}\begin{pmatrix}2\\1\\0\end{pmatrix} = \frac{1}{5\sqrt{2}}\begin{pmatrix}-1\\7\\0\end{pmatrix} = \begin{pmatrix}-0.141\\0.990\\0\end{pmatrix}$$

3) Considere no algoritmo de Traçado de Raios, o raio que passa no ponto P(i,j) no plano de projeção. Determine (se existir) o ponto P em que este raio intercepta a esfera de centro C e raio 1, e a normal unitária da esfera neste ponto. Considere apenas o ponto entrando na esfera.

	eye	P(i,j)	C
x	10	8	3
y	6	5	2
z	0	0	0

Resp.:

Ponto genérico do raio:

$$\mathbf{p}(t) = \begin{pmatrix}10\\6\\0\end{pmatrix} + t\begin{pmatrix}8-10\\5-6\\0-0\end{pmatrix} = \begin{pmatrix}10-2t\\6-t\\0\end{pmatrix}$$

Ponto na superfície da esfera:

$$(x-3)^2 + (y-2)^2 + (0-0)^2 = 1$$

Ponto na esfera e no raio:

$$(10-2t-3)^2 + (6-t-2)^2 = 1$$

$$(7-2t)^2 + (4-t)^2 = 1$$

$$4t^2 - 28t + 49 + t^2 - 8t + 16 - 1 = 0 \qquad 5t^2 - 36t + 64 = 0 \qquad t = \frac{36 \pm \sqrt{36^2 - 4\cdot 5\cdot 64}}{2\cdot 5}$$

$$t_i = \frac{36 \pm \sqrt{16}}{10} = \begin{cases}3.2 \leftarrow\\4\end{cases}$$

Ponto de interseção (entrando):

$$\mathbf{p}(t_i) = \begin{pmatrix}10-2t\\6-t\\0\end{pmatrix} = \begin{pmatrix}10-2\cdot 3.2\\6-3.2\\0\end{pmatrix} = \begin{pmatrix}3.6\\2.8\\0\end{pmatrix}$$

Normal no ponto de entrada:

$$\mathbf{n} = \begin{pmatrix}3.6\\2.8\\0\end{pmatrix} - \begin{pmatrix}3\\2\\0\end{pmatrix} = \begin{pmatrix}0.6\\0.8\\0\end{pmatrix}$$ (já está normalizada, uma vez que o raio é 1).

5.9 Exercícios

4) Descreva o princípio óptico da câmera *pinhole* que é utilizado no nosso modelo de câmera.

5) Por que no Algoritmo de Rastreamento de Raios não temos que levar em conta o tempo em que a luz demora para chegar nas superfícies? Existe uma fase transiente, isto é, que ocorre temporariamente?

6) Qual a distância focal de uma câmera que tem abertura *fovy*=90º e altura 35 mm?

7) Em que situações o centro óptico não está no centro de uma fotografia?

8) Explique, em linhas gerais, como fazemos para calcular a cor que deve ser atribuída a um *pixel* no algoritmo de Rastreamento de Raios, quando um raio atinge um ponto em um objeto.

9) Explique cada um dos termos da equação abaixo. Faça um desenho para explicar $\hat{\mathbf{n}}, \cdot \hat{\mathbf{L}}, \hat{\mathbf{r}}, \hat{\mathbf{v}}$.

$$\begin{pmatrix} I_r \\ I_g \\ I_b \end{pmatrix} = \begin{pmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} + \sum_{luzes} \left(\begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{dr} \\ k_{dg} \\ k_{db} \end{pmatrix} \left(\hat{\mathbf{n}} \cdot \hat{\mathbf{L}}\right) + \begin{pmatrix} l_r \\ l_g \\ l_b \end{pmatrix} \otimes \begin{pmatrix} k_{sr} \\ k_{sg} \\ k_{sb} \end{pmatrix} \left(\hat{\mathbf{r}} \cdot \hat{\mathbf{v}}\right)^n \right)$$

10) Dado um cilindro cujo centro de uma base se encontra na posição $\mathbf{b_1}$ e o centro da outra base se encontra na posição $\mathbf{b_2}$ e cujo diâmetro das bases é d, escreva uma rotina que calcule a interseção entre o raio e o cilindro. (Sugestão: Transformar o cilindro para a posição canônica, fazendo a mesma transformação no raio.)

11) Seja r = **o** + *t***d** o raio incidente numa superfície de vidro num ponto cuja normal é **n** e o índice de refração é n, calcule o raio refratado considerando que o índice de refração do ar é 1.

12) Considere agora o raio saindo do vidro de índice de refração **n** para o ar. Calcule o raio refratado. Sempre haverá um raio refratado possível?

13) Determine o ponto **p** em que o raio partindo do ponto **eye** na direção **d** intercepta o plano que contém o triângulo **a**, **b** e **c**. Descreva como você faria para determinar se o ponto encontrado é interior ao triângulo e se o raio o atinge pelo lado positivo do triângulo. (Obs.: o ponto **p** deve ser calculado explicitamente.)

	a	**b**	**c**	**d**	**eye**
x	3	4	2	-1	12
y	4	1	1	-2	18
z	2	5	6	-1	12

14) No algoritmo de Traçado de Raios, um raio atinge uma face em um ponto **p** onde a normal é **n**. As posições do **eye** e da única luz pontual presente na cena estão dadas, juntamente com as coordenadas de **p** e **n**, na tabela de Posições mostrada abaixo. Na tabela de RGB estão definidas as propriedades do material da superfície e a intensidade

da luz. A última tabela fornece as demais propriedades do material cuja superfície contém P. Sabendo-se que o objeto interceptado é convexo e é o único da cena, determine:

a) a componente difusa da reflexão da **luz** em **P**;
b) a componente especular da reflexão da **luz** em **P**;
c) a cor RGB a ser atribuída ao *pixel* correspondente incluindo estes dois efeitos e a luz ambiente.

	Posições			
	p	n	eye	luz
x	6.0	1.0	9.0	3.0
y	2.0	2.0	1.0	5.0
z	1.0	1.0	1.0	1.0

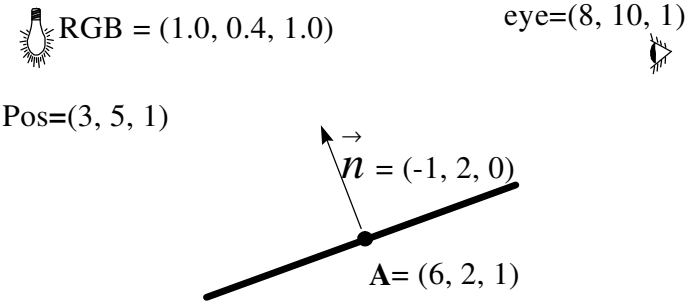
	Material			Luz
	Ia*Kd	Kd	Ks	I
r	0.2	0.2	0.9	1.0
g	0.2	0.3	0.8	1.0
b	0.2	1.0	1.0	1.0

n	30.0
K refletido (R)	0.0
K refratado (T)	0.0
Opacidade	1.0 (opaco)
Índice de refração	1.5

(contas com 3 algarismos significativos)

15) Determine a componente difusa da luz de uma lâmpada de intensidade (0.8, 0.4, 0.2) colocada na posição homogênea $[0, 1, 0, 0]^T$ sobre o ponto A do triângulo **abc**. O material da face tem coeficiente difuso k_d=(0.6, 0.5, 0.3), o triângulo é orientado no sentido trigonométrico e as coordenadas do vértice são: **a**=(3, 2, 1), **b**=(8, 2, 1) e **c**=(4, 8, -4).

16) Determine as componentes especulares da cor do ponto A considerando a luz e o observador mostrados na figura abaixo.



Material: Difusa: K_d = (0.6, 0.7, 0.6)

Especular: K_s = (1.0, 1.0, 1.0) e n=8

17) Determine a cor de um *pixel* correspondente à posição (5, 2, 2) do plano de projeção visto a partir do centro de projeção (10, 2, 1) considerando apenas a componente de reflexão difusa de uma luz de intensidade (0.3, 0.8, 0.6) e localização (10, 10, 10) numa esfera de raio 2 e centro (1, 2, 2) que tem como k_d = (0.8, 1.0, 0.3).