8. Síntese de Imagens: Cálculo de Cor

O processo de determinar a aparência (ou seja, a textura e a cor) das superfícies visíveis é chamado de *shading*. Essa "aparência" da superfície depende das **propriedades** da superfície (textura, reflectância, etc.), do **tipo de fonte de luz** que ilumina o objeto, e da sua **posição relativa** à fonte de luz e ao observador.

Apesar da existência de diversos algoritmos, não existe uma solução geral para o problema do realismo visual, incluindo-se aí o ocultamento das faces e o *shading*. Devemos buscar soluções específicas, em função dos requisitos das aplicações, evitando a busca de soluções generalíssimas.

• Cor e Iluminação

A iluminação é uma técnica importante em objetos gráficos criados através do computador. Sem iluminação os objetos tendem a não apresentar características realistas. O princípio da iluminação consiste em simular como os objetos refletem as luzes. Alguns modelos de iluminação de uma cena envolvem basicamente três aspectos:

-**propriedades dos materiais** que formam as superfícies dos objetos
-**propriedades de luzes**, definidas a partir das fontes de luz inseridas na cena
-**parâmetros globais de iluminação**, que depende do ambiente no qual os objetos estão inseridos.

Apesar de o estudo e descrição de cor ser um assunto amplo, podemos simplificar uma cor com sendo matematicamente um vetor 3D formado por três componentes R, G e B que indicam respectivamente a quantidade de vermelho, verde e azul que compõem tal cor. Esta definição de cor será necessária pois todo o cálculo de iluminação envolve o uso de cores.

O modelo de iluminação aqui descrito é baseado em um modelo chamado de *iluminação de Phong*. Em cada vértice da primitiva, uma cor é calculada usando as **propriedades dos materiais** junto com as **propriedades das luzes** envolvidas. Para este cálculo é utilizado um modelo geométrico semelhante ao da figura 8.3 para a incidência e reflexão da luz sobre os objetos.

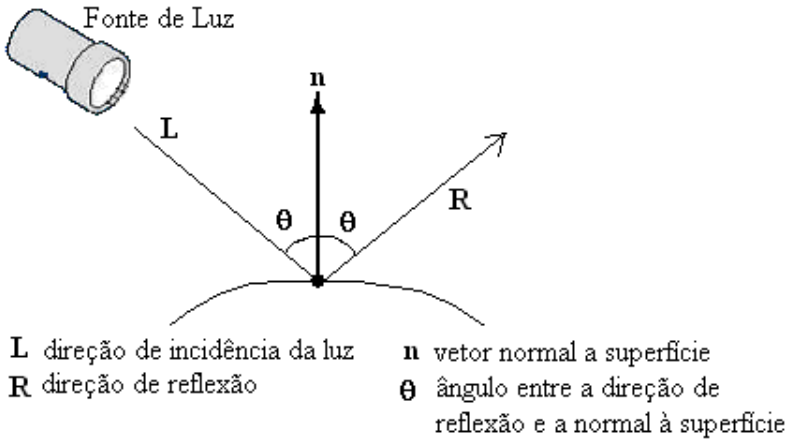

Fig. 8.3 Geometria do Modelo de Iluminação

A cor para o vértice é calculada como sendo a soma de quatro outras cores. Estas quatro cores que contribuem para a cor final do vértice são: Iluminação Ambiente, Componente Difusa, Luz Especular e Emissão.

A) Iluminação Ambiente: representa a luz do ambiente no qual se encontram os objetos. Não é resultante de nenhum fonte de luz especificamente, mas é o resultado global de toda iluminação do ambiente (luz solar, reflexão indireta de todos os objetos da cena, etc). Com isto ela **vem de todas as direções**. Normalmente é muito difícil calcular esta luz computacionalmente. Uma solução simples é fornecer o valor desta luz como um dado de entrada.

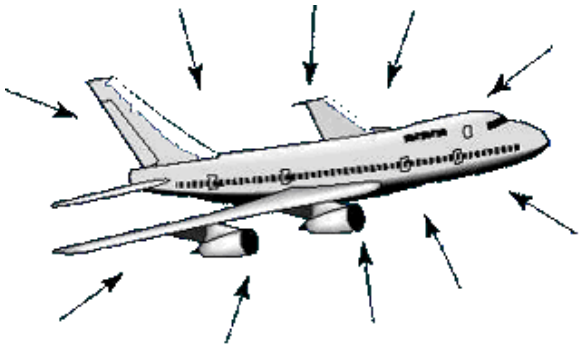

Fig. 8.4 Iluminação Global

B) Luz Difusa: é a luz que **vem de uma direção**, ou seja, de uma determinada fonte de luz que se encontra na cena. Uma vez que esta luz incida em uma superfície, é **difundida igualmente em todas as direções**, assim a superfície aparece igualmente luminosa, não importando de onde é visualizada. Qualquer luz vindo de uma posição particular ou direção tem provavelmente uma componente difusa.

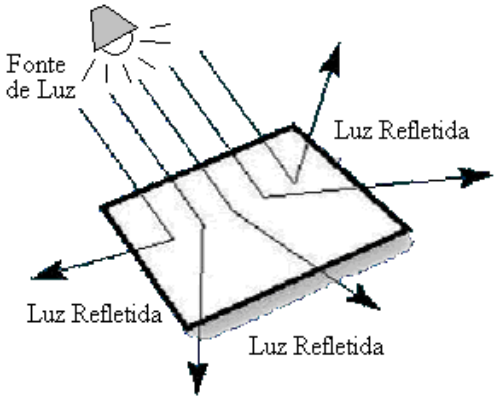

Fig. 8.5 Iluminação Difusa.

C) Luz Especular: luz que **vem de uma direção**, ou seja de uma fonte de luz, e tende a ser **refletida numa única direção**;

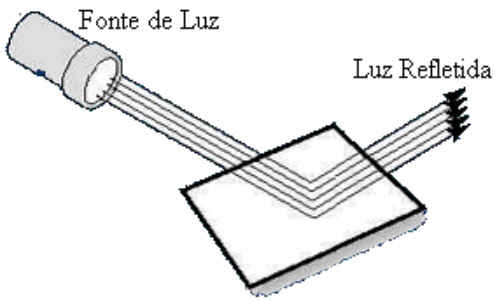

Fig. 8.6 Iluminação Esécular.

D) Luz Emissiva: simula a luz que se origina do próprio objeto; a cor emissiva de uma superfície adiciona intensidade ao objeto, mas não é afetada por qualquer fonte de luz; ela também não introduz luz adicional da cena.

É importante destacar que as **propriedades globais de iluminação** incluem a definição da luz ambiente resultante (independente das fontes de luz que se encontram na cena). Já as **propriedades locais de cada fonte de luz** podem incluir a sua cor (contribuição) ambiental, sua contribuição difusa e sua contribuição especular, dentre outras propriedades.

As **propriedades locais do material** que forma os objetos também indicam como eles refletem a luz ambiente, a luz difusa e a luz especular, além de definir como deve ser este brilho especular, e como eles emitem sua própria luz, etc.

8.2.1. Como o OpenGL simula a Iluminação

Para adicionar luz a uma cena são necessários os seguintes passos :

- Definir os vetores normais para cada vértice do objeto
- Criar, selecionar, e posicionar uma ou mais fontes de luz.
- Criar e selecionar um modelo de iluminação que define a luz ambiente global e a localização efetiva do ponto de visão.
- Definir as propriedades dos materiais que compõem os objetos na cena

A) Definição do Vetores Normais

Os vetores normais devem ser definidos para cada vértice de todos os objetos. Estes vetores determinam a orientação do objeto relativo às fontes de luz. A última seção deste capítulo mostra como calcular os vetores normais.

Uma vez calculada a normal de um vértice, ela dirá ao OpenGL como o objeto reflete luz ao redor deste vértice. Imaginando que exista um espelho pequeno no vértice, a normal descreve como o espelho é orientado, e por conseguinte como a luz é refletida. A função glNormal3f recebe as três coordenadas do vetor normal que é usada no calculo de iluminação para todos os vertices até uma nova normal ser configurada.

```
void glNormal3f (float nx, flota ny, floatr nz);
```

Este função deve ser chamada antes de desenhar o vértice (glVertex). Lembre-se que OpenGl é uma máquina de estado, logo se um próximo vértice tiver outra normal, então deve-se fazer outra chamada de glNormal.

B) Criação, Seleção e Posicionamento de Fontes de Luz

Lembremos que cada fonte de luz possui propriedades locais como contribuição ambiental, difusa e especular. Estas contrinuições na verdade são cores (vetor 3D). Outras propriedades incluem posição da fonte de luz e atenuação da luz à medida em que se afasta da fonte.

A função **glLightf** é usada para fixar essas propriedade ou parâmetros para uma fonte de luz. O OpenGL permite a implementação de até oito luzes que são nomeadas como GL_LIGHT0 até GL_LIGHTn onde n é um menos o número máximo suportado. As propriedades de cores (ambiental, difusa, especular) permitem interações separadas com as diferentes propriedades dos materiais. Propriedades de posicionamento controlam a localização e o tipo da luz e a atenuação controla a tendência natural da luz deteriorar-se sobre a distância. A função glLightf tem a seguinte definição:

```
void glLightf(int light, int pname, float param);
```

ou

```
void glLightfv(int light, int pname, float *param);
```

A diferença entre glLightf e glLightfv é que enquanto o parâmetro *param* do primeiro é um float, o parâmetro *param* do segundo deve ser um vetor de floats. Ambas as funções criam uma luz especificada pelo parâmetro *light* que pode ser GL_LIGHT0 GL_LIGHT1,..., ou GL_LIGHT7. A propriedade da luz que está sendo setada é definida por *pname* que determina uma propriedade nomeada conforme tabela abaixo. *param* indica os valores para os quais a propriedade *pname* é fixada; *param* é um ponteiro para um grupo de valores se a versão de vetor é usada (glLightfv, com v no final), ou o próprio valor se a versão não vetorial for usada (glLightf, sem v no final).

Nome do Parâmetro	Valor Padrao	Significado
GL_AMBIENT	(0.0, 0.0, 0.0, 1.0)	Intensidade RGBA da luz ambiente
GL_DIFFUSE	(1.0, 1.0, 1.0, 1.0)	Intensidade RGBA da luz difusa
GL_SPECULAR	(1.0, 1.0, 1.0, 1.0)	Intensidade RGBA da luz especular
GL_POSITION	(0.0, 0.0, 1.0, 0.0)	Posição da luz (x, y, z, w)
GL_SPOT_DIRECTION	(0.0, 0.0, -1.0)	direção da luz spotlight (x, y, z)
GL_SPOT_EXPONENT	0.0	Expoente spotlight
GL_SPOT_CUTOFF	180.0	Ângulo do spotlight
GL_CONSTANT_ATTENUATION	1.0	Fator de constante de atenuação
GL_LINEAR_ATTENUATION	1.0	Fator de atenuação linear
GL_QUADRATIC_ATTENUATION	0.0	Fator de atenuação quadrática

Perceba, na tabela, que as propriedades GL_AMBIENT, GL_DIFFUSE,GL_SPECULAR, GL_POSITION e GL_SPOT_EXPOENT requerem um vetor de floats, logo devem ser especificadas com aversão vetorial da função, ou seja , com glLightfv (com v no final). Já as demias propriedades requerem apenas um valor e desvem ser setadas com a versão não vetorial, ou seja, com glLightf.

Exemplo: Neste exemplo uma cor possui 4 parametros: alem do R,G e B, há um parametro para inidicar transparencia, mas nao se preocupe com este ultimo agora.

```
float luz_ambiente[]= {0.0,0.0,0.0,1.0}; //Cor preta R=0.0,G=0.0,B=0.0
float luz_difusa[] = {1.0,1.0,1.0,1.0};  //Cor branca R=1.0,G=1.0,B=1.0
float luz_epecular[] = {1.0,1.0,1.0,1.0};//Cor branca R=1.0,G=1.0,B=1.0
float luz_posicao[] = { 1.0, 1.0, 1.0, 0.0 };

glLightfv(GL_LIGHT0, GL_AMBIENT, luz_ambiente);
glLightfv(GL_LIGHT0, GL_DIFFUSE, luz_difusa);
glLightfv(GL_LIGHT0, GL_SPECULAR, luz_epecular);
glLightfv(GL_LIGHT0, GL_POSITION, luz_posicao);
```

C) Criação e Seleção de Modelo de Iluminação

Estas propriedades incluem as que não são diretamente conectadadas com materiais ou luzes. Estas propriedades incluem :

- **Cor ambiente global**, que inicia a contribuição global do ambiente na equação de iluminação.

- **Modo de visualização local**, que desabilita a otimização mas que provê mais rapidamente os cálculos de iluminação.
- **Iluminação de um ou dos dois lados** de cada face do objeto.

O comando usado para especificar todas as propriedades do modelo de iluminação é **glLightModelf**. Esta função tem dois argumentos: a propriedade do modelo de iluminação e o valor desejado para aquela propriedade.

```
void glLightModelf(int pname, float param);
ou
void glLightModelfv(int pname, float *param);
```

A versão terminada em v indica que o segundo parâmetro é um vetor de floats em vez de um só valor. Estas funções especificam as propriedades do modelo de iluminação . O parâmetro *pname* indica uma das propriedades especificadas na tabela abaixo. O parâmetro *param* indica os valores que a propriedade de *pname* assume; *param* é um ponteiro a um grupo de valores se a versão de vetor for usada (glLightModelfv, terminada em v), ou o próprio valor se a versão não vetorial for utilizada (glLightModelf, sem v). A versão não vetorial pode ser usada para fixar as propriedades GL_LIGHT_MODEL_LOCAL_VIEWER e GL_LIGHT_MODEL_TWO_SIDE, que requerem apenas um valor. Mas a propriedade GL_LIGHT_MODEL_AMBIENT requer a versão vetorial (com v) pois esta propriedade requer um vetor com as coordenadas da luz ambiente.

Nome do Parâmetro	Valor Padrão	Significado
GL_LIGHT_MODEL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	Intensidade RGBA ambiente de toda a cena
GL_LIGHT_MODEL_LOCAL_VIEWER	0.0 or GL_FALSE	Como o angulo de reflexão especular e calculado
GL_LIGHT_MODEL_TWO_SIDE	0.0 or GL_FALSE	Define entre a iluminação de um ou dois lados.

D) Propriedades de Materiais

Estas propriedades descrevem a cor e propriedades de superfície de um material . OpenGL suporta que as propriedades de materiais sejam definidas na frente e nas costas dos objetos. Algumas das propriedades definidas pelo OpenGL são:

- GL_DIFFUSE - cor básica de objeto
- GL_SPECULAR - cor de destaques do objeto
- GL_AMBIENT - cor do objeto quando não diretamente iluminado
- GL_EMISSION - cor emitida pelo objeto
- GL_SHININESS - concentração de brilhos em objetos. Valores variam de 0 (superfície muito áspera - nenhum destaque) até 128 (muito brilhante) .

A tabela a seguir mostra todas as propriedades.

A função responsável por setar as propriedades do material é:

```
void glMaterialf(int face, int pname, float param);
ou
void glMaterialfv(int face, int pname, float *param);
```

O parâmetro *face* indica se a propriedade é aplicada na parte frontal, na parte de trás ou em ambos os lados das faces do objeto. Para tal, este parâmetro deve assumir respectivamente os valores GL_FRONT, GL_BACK ou GL_FRONT_AND_BACK. O parâmetro *pname* indica uma das propriedades da tabela abaixo e o parâmetros *param* indica os valores que estas propriedades podem assumir. Perceba que *param* pode ser um valor (glMaterialf) ou um vetor de valores reais (glMaterialfv).

Nome do Parâmetro	Valor Padrão	Significado
GL_AMBIENT	(0.2, 0.2, 0.2, 1.0)	Cor ambiente do material
GL_DIFFUSE	(0.8, 0.8, 0.8, 1.0)	Cor difusa do material
GL_AMBIENT_AND_DIFFUSE		Cores ambiente e difusa do material.
GL_SPECULAR	(0.0, 0.0, 0.0, 1.0)	Cor especular do material
GL_SHININESS	0.0	Expoente especular
GL_EMISSION	(0.0, 0.0, 0.0, 1.0)	Cor emissiva do material
GL_COLOR_INDEXES	(0,1,1)	Índice de cores, ambiente, difusa e especular.

• Vetores Unitários e Produto Vetorial (Revisão)

Produto Vetorial

Chama-se produto vetorial de dois vetores a um outro vetor, cujo comprimento ou módulo, é igual ao produto dos módulos dos dois vetores pelo seno do ângulo por eles formado, cuja direção é perpendicular ao plano determinado pelos dois vetores e o sentido é o determinado pela regra da mão direita.

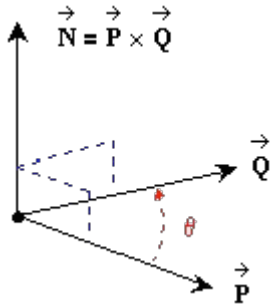

Fig. 8.7 O Produto Vetorial.

Vetores Unitários

Dado um vetor $\mathbf{a}=(a_x,a_y,a_z)$, definimos o vetor unitário **u** como sendo o vetor de comprimento igual a uma unidade e tendo a mesma direção e sentido de **a**. O vetor **u** é definido como

$$\mathbf{u} = (1/||\mathbf{a}||).\ \mathbf{a},$$

onde $||\mathbf{a}||$ é o comprimento ou norma do vetor **a.**

Vetores Fundamentais Unitários: sejam (X,Y,Z) as coordenadas de um vetor **a**. Chamemos de

$$(\vec{i}\ \ \vec{j}\ \ \vec{k})$$

os três vetores unitários localizados sobre os eixos fundamentais de um sistema de coordenadas cartesianas (estes vetores também são representados por $\mathbf{e_1}$=(1,0,0), $\mathbf{e_2}$=(0,1,0) e $\mathbf{e_3}$=(0,0,1) respectivamente).

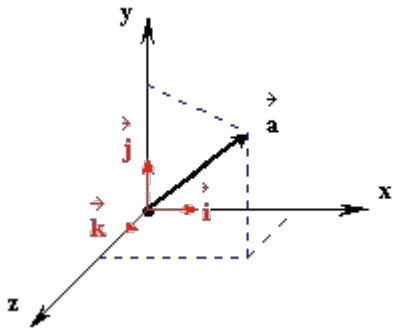

Fig. 8.8 Os Vetores Fundamentais Unitários.

Então,

$$X\cdot\vec{i}, Y\cdot\vec{j}, Z\cdot\vec{k}$$

são vetores localizados sobre os eixos fundamentais e de comprimento igual a X, Y, Z respectivamente. Sendo o vetor **a**, a soma desses três vetores, podemos escrever:

$$\vec{a} = X\cdot\vec{i} + Y\cdot\vec{j} + Z\cdot\vec{k}$$

o que constitui a representação analítica do vetor.

Voltando ao produto vetorial, devemos lembrar que são válidas as regras ordinárias do cálculo algébrico (exceto a comutativa) e que o produto vetorial de dois vetores unitários fundamentais é sempre igual ao outro vetor unitário, com sentido determinado pela regra da mão direita.

$$\vec{i}\times\vec{j} = \vec{k} \qquad \vec{j}\times\vec{i} = -\vec{k}$$
$$\vec{k}\times\vec{i} = \vec{j} \qquad \vec{i}\times\vec{k} = -\vec{j}$$
$$\vec{j}\times\vec{k} = \vec{i} \qquad \vec{k}\times\vec{j} = -\vec{i}$$

Assim, considerando dois vetores na forma analítica,

$$\vec{P} = P_X\cdot\vec{i} + P_Y\cdot\vec{j} + P_Z\cdot\vec{k} \qquad \vec{Q} = Q_X\cdot\vec{i} + Q_Y\cdot\vec{j} + Q_Z\cdot\vec{k}$$

efetuamos o produto vetorial dos dois vetores,

$$\vec{P} \times \vec{Q} = \left(P_x \cdot \vec{i} + P_y \cdot \vec{j} + P_z \cdot \vec{k}\right) \times \left(Q_x \cdot \vec{i} + Q_y \cdot \vec{j} + Q_z \cdot \vec{k}\right)$$

e que desenvolvendo algebricamente, obedecidas as propriedades relativas aos produtos vetoriais dos vetores unitários, obtemos:

$$\vec{N} = \vec{P} \times \vec{Q} = \left(P_y \cdot Q_z - P_z \cdot Q_y\right) \cdot \vec{i} + \left(P_z \cdot Q_x - P_x \cdot Q_z\right) \cdot \vec{j} + \left(P_x \cdot Q_y - P_y \cdot Q_x\right) \cdot \vec{k}$$

e que se pode resumir, em termos de fórmulas, para o produto vetorial 3D em um sistema de coordenadas da mão direita, como se segue:

$$N_x = P_y \cdot Q_z - P_z \cdot Q_y$$

$$N_y = P_z \cdot Q_x - P_x \cdot Q_z$$

$$N_z = P_x \cdot Q_y - P_y \cdot Q_x$$

O que se pode também escrever sob a forma matricial, em termos de determinantes:

$$\vec{N} = \vec{P} \times \vec{Q} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ P_x & P_y & P_z \\ Q_x & Q_y & Q_z \end{vmatrix}$$

$$\vec{N} = \begin{vmatrix} P_y & P_z \\ Q_y & Q_z \end{vmatrix} \cdot \vec{i} + \begin{vmatrix} P_z & P_x \\ Q_z & Q_x \end{vmatrix} \cdot \vec{j} + \begin{vmatrix} P_x & P_y \\ Q_x & Q_y \end{vmatrix} \cdot \vec{k}$$

Definição de Vetor Normal a uma Face e um Vértice

Vimos que os objetos são modelados como um conjunto de faces, mais especificamente triângulos ou quadriláteros. Por exemplo, podemos modelar um cubo unitário através das seguintes listas de vértices e faces (fig 8.9):

$V_{1=}(0,0,0), V_{2=}(1,0,0), V_{3=}(1,0,1), V_{4=}(0,0,1), V_{5=}(1,1,0), V_{6=}(1,1,0), V_{7=}(1,1,1), V_{8=}(0,1,1);$

$F_1=V_1V_2V_3V_4$, $F_2=V_2V_6V_7V_3$, $F_3=V_1V_5V_6V_2$, $F_4=V_1V_4V_8V_5$, $F_5=V_3V_7V_8V_4$, $F_6=V_5V_8V_7V_5$,

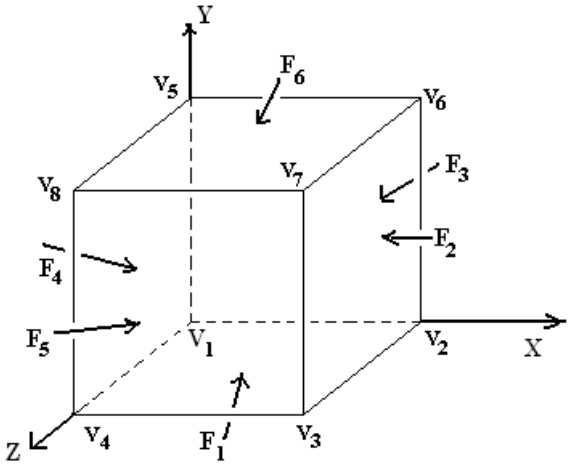

Fig. 8.9 Modelagem do Cubo Unitário.

É importante perceber que a definição de cada face da lista requer que os vértices estejam ordenados. Neste exemplo a ordenação está no sentido anti-horário. Esta ordenação é essencial para possibilitar a cálculo correto das normais às faces. Vimos que as normais são necessárias para o cálculo da iluminação. E como calcular o vetor normal a uma face F? Basta tomar dois vetores não colineares da face F e calcular o produto vetorial destes vetores. Por exemplo, a normal **n** à face F2 do cubo pode ser calculada a partir do produto vetorial

$$\mathbf{n} = \mathbf{u} \times \mathbf{v},$$

onde os vetores **u** e **v** são

$$\mathbf{u}=V_6-V_2 \quad \text{e} \quad \mathbf{v}=V_2-V_3.$$

Perceba que há outras possibilidades de se calcular os vetores **u** e **v**.
O cálculo do vetor normal a um vértice pode ser simulado computacionalmente fazendo-se a média dos vetores normais às faces que compartilham aquele vértice, ponderada pelas áreas destas faces.