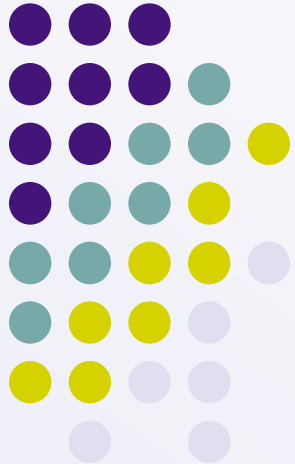
Introdução à Computação Gráfica

Marcel P. Jackowski
mjack@ime.usp.br

Aula #8: Iluminação

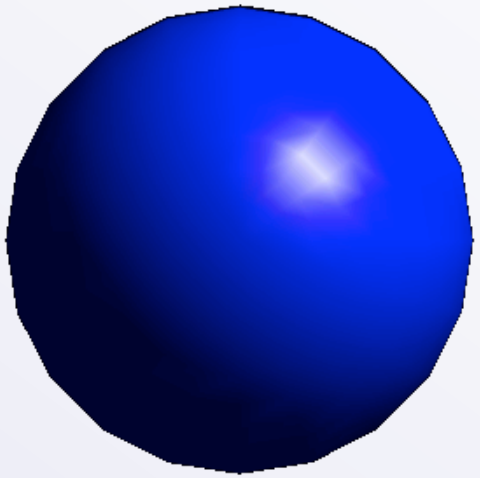


Objetivos

- Introdução à iluminação
  - Tonalização (“sombreamento”)
  - Local vs. global
- Iluminação em OpenGL
  - Componentes

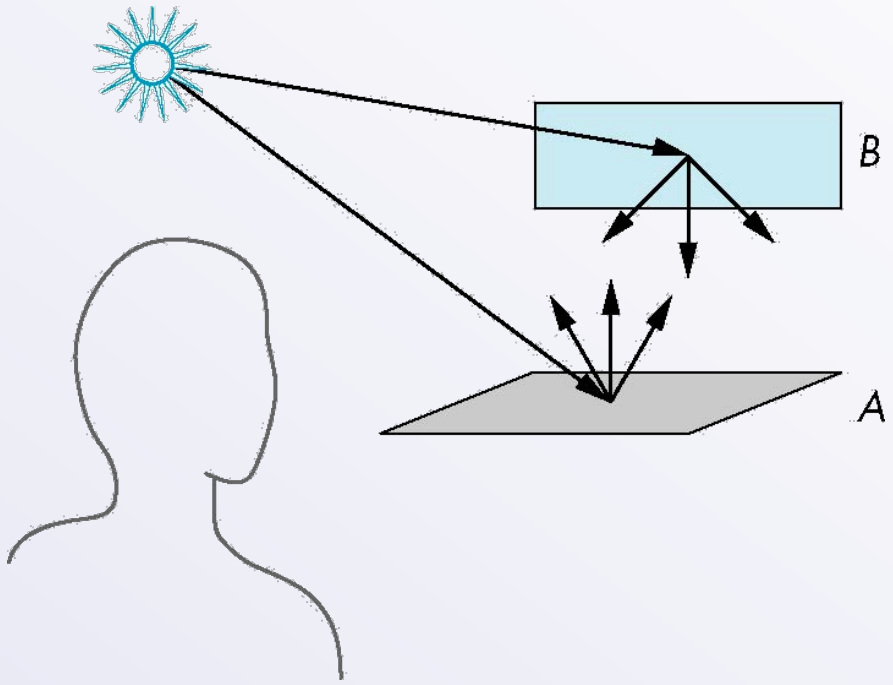
Tonalização

- Como a aparência de uma esfera é formada ?
  - Interações entre luz-material causam cada ponto ter uma cor ou tonalização diferente.
- Precisamos considerar
  - Fontes de luz
  - Propriedades do material
  - Localização do observador
  - Orientação da superfície



Espalhamento (“Scattering”)

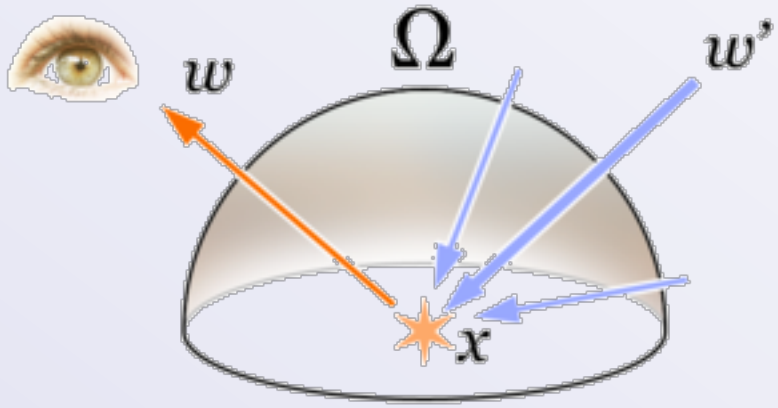
- Luz atinge superfície A
  - Parte é refletida
  - Parte é absorvida
- Parte da luz refletida atinge superfície B
  - Parte é refletida
  - Parte é absorvida
- E assim por diante…



Equação de renderização

- Esse espalhamento infinito e absorção de luz pode ser descrito através da *equação de renderização*.

$$L_o(\mathbf{x}, \omega, \lambda, t) = L_e(\mathbf{x}, \omega, \lambda, t) + \int_\Omega f_r(\mathbf{x}, \omega', \omega, \lambda, t) L_i(\mathbf{x}, \omega', \lambda, t)(-\omega' \cdot \mathbf{n}) d\omega'$$



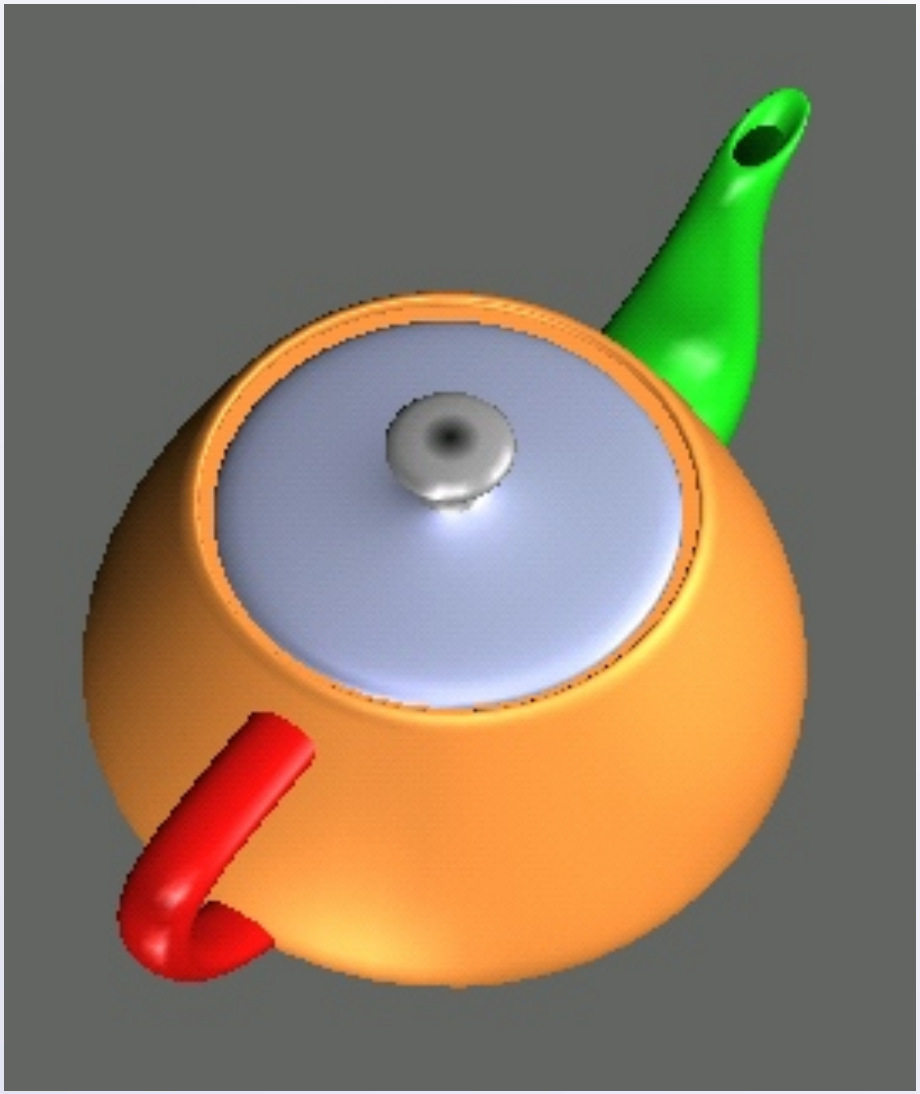
Kajiya, James T., The rendering equation", *SIGGRAPH 1986*

Equação de renderização

- Normalmente não possui solução
- Ray tracing é uma das aproximações para o caso de superfícies refletoras perfeitas
- A equação de renderização tem natureza global e automaticamente gera:
  - Sombras
  - Espalhamentos múltiplos entre objetos

Ray-tracing (global)

Renderização direta (local)



Renderização global e local

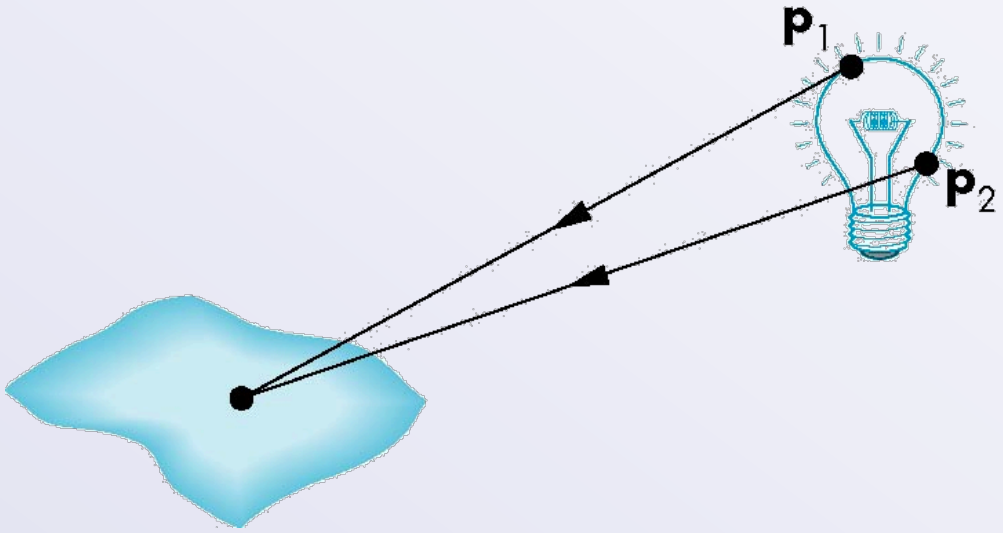
- A tonalização correta requer um cálculo global que envolve todos os objetos e fontes de luz
  - Incompatível com o modelo em pipeline que sombreia cada polígono independentemente (renderização local)
- Todavia, em CG, e especialmente em aplicações em tempo real, nos contentamos que a nossa cena “pareça” realística
  - Técnicas para aproximação de efeitos globais

Interação entre luzes e materiais

- A luz que atinge um objeto é parcialmente absorvida e parcialmente espalhada (refletida)
- A quantidade de luz refletida determina a cor e a intensidade do objeto
  - Uma superfície possui cor vermelha sob luz branca, porque o componente vermelho da luz é refletido e o resto é absorvido.
- A reflexão da luz depende da orientação e suavidade da superfície.

Fontes de luz genéricas

- São mais difíceis de modelar, pois temos que integrar todos os raios provenientes da fonte emissora
- Descritas por **seis** componentes: posição, direção e intensidade



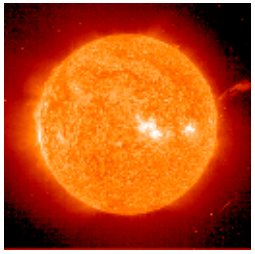
Fontes de luz

- Ponto de luz (ideal)
  - Modelo com posição e cor
  - Fonte pode estar distante = distância infinita (paralela)
  - Intensidade de iluminação inversamente **proporcional** à distância até o objeto sendo iluminado
- Spots
  - Restrição do ponto ideal de luz
- Ambiente
  - Mesma quantidade de luz em qualquer lugar da cena
  - Modela a contribuição de várias fontes de luz e superfícies refletoras

LÂMPADA

fonte de luz pontual

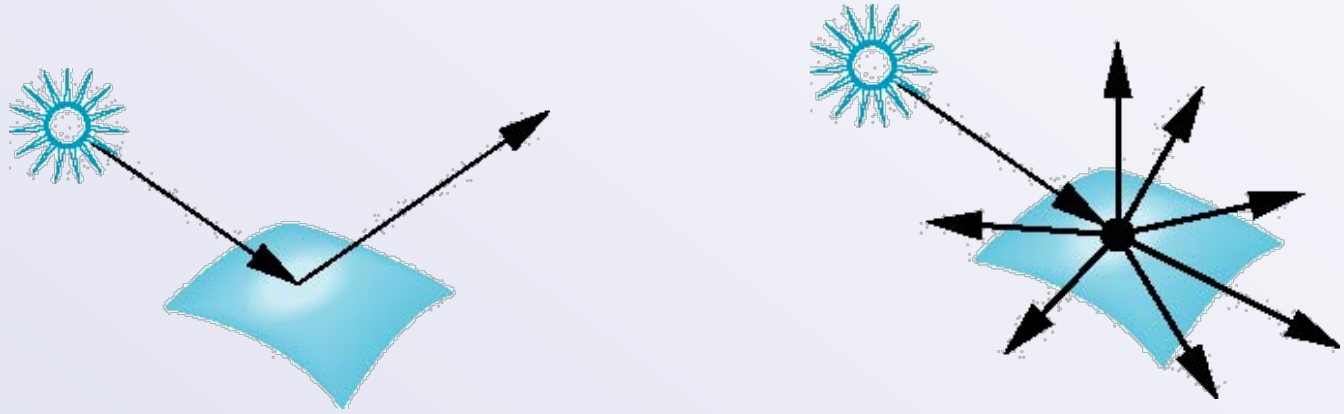
SOL



HOLOFOTE

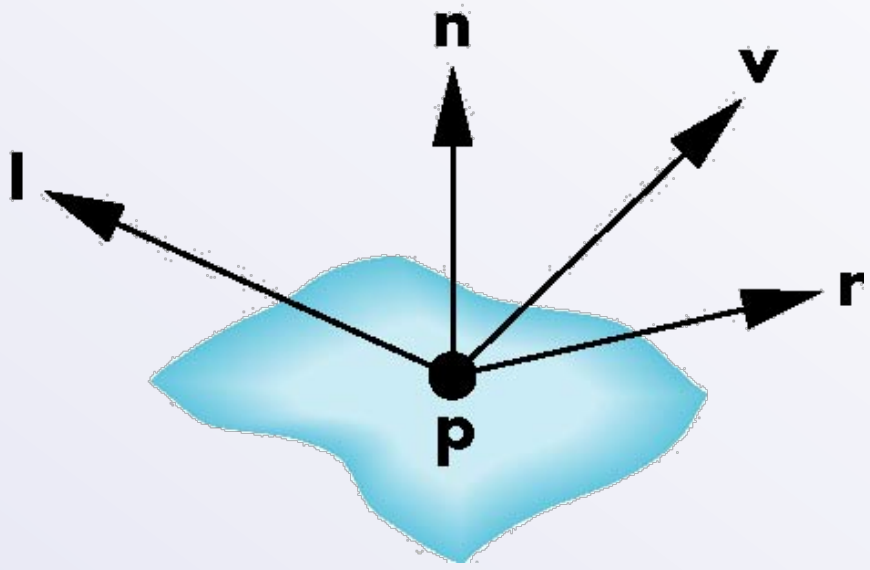
Tipos de superfície

- Quanto mais suave for a superfície, mais concentrada será a luz refletida nela
- Uma superfície irregular espalhará os raios em diversas direções.



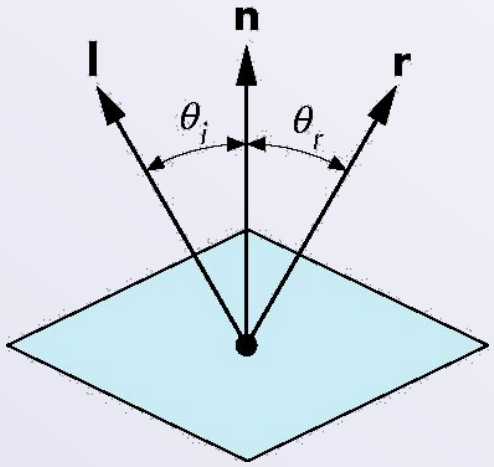
Modelo Phong

- Modelo simples que pode ser calculado rapidamente. Possui três componentes:
  - Difusivo
  - Especular
  - Ambiente
- Usa quatro vetores
  - Fonte de luz (**l**)
  - Observador (**v**)
  - Normal (**n**)
  - Refletor ideal (**r**)



O refletor ideal (r)

- O vetor normal é determinado pela orientação da superfície
- Ângulo de incidência = ângulo de reflexão
- Estes três vetores são coplanares e podem ser calculados

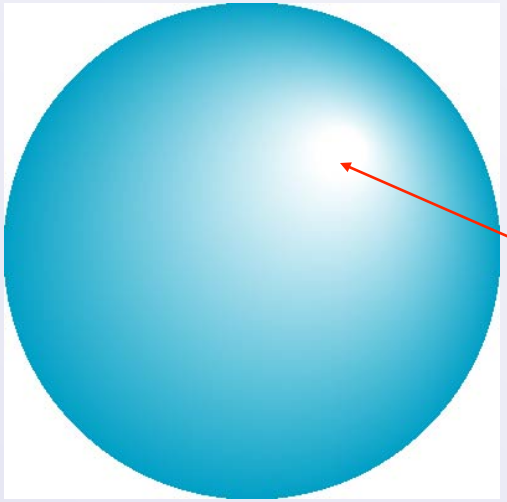


Superfícies lambertianas

- Superfícies **difusoras** perfeitas
- Refletem luz igualmente em todas as direções
- Quantidade de luz refletida é proporcional ao componente vertical da luz incidente
  - Luz refletida ~ $\cos \theta_i = \mathbf{l} \cdot \mathbf{n}$
- Coeficientes $\{k_R, k_G, k_B\}$ representam a quantidade refletida de cada componente de cor

Superfícies especulares

- A maioria das superfícies não são difusoras perfeitas.
- Superfícies suaves mostram um brilho especular porque a luz de entrada é refletida em direções concentradas na direção da reflexão ideal.



Reflexões especulares

- Phong propôs usar um termo que regula a especularidade dependendo do ângulo entre o observador e o vetor de reflexão ideal.

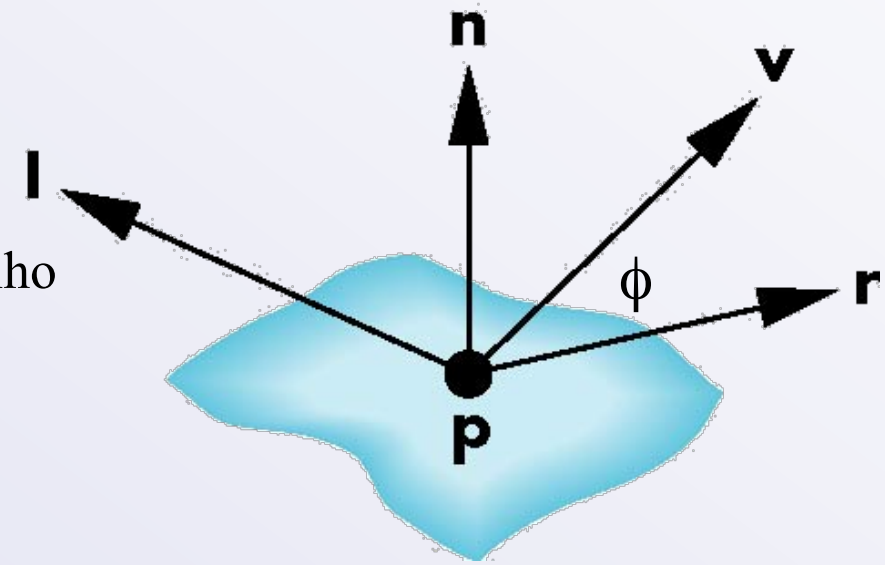
$$\mathbf{I}_s \sim k_s \mathbf{I} \cos^{\alpha}\phi$$

Intensidade refletida

Coeficiente de absorção

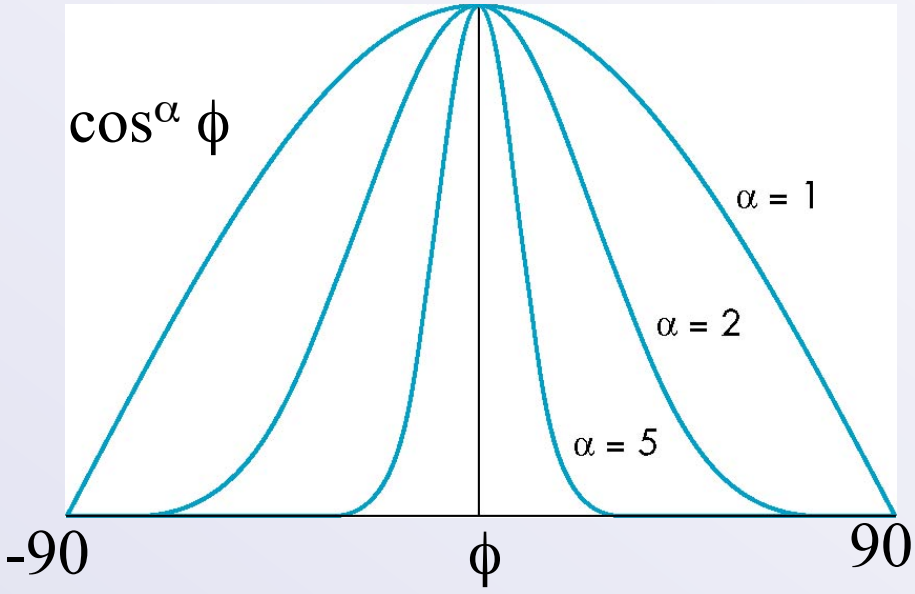
Intensidade de entrada

Coeficiente de brilho (metal, plástico)



Coeficiente de brilho

- Valores de α entre 100 e 200 correspondem aos metais (lim α infinito -> espelho)
- Valores entre 5 e 10 deixam a superfície com uma aparência plástica



Luz ambiente

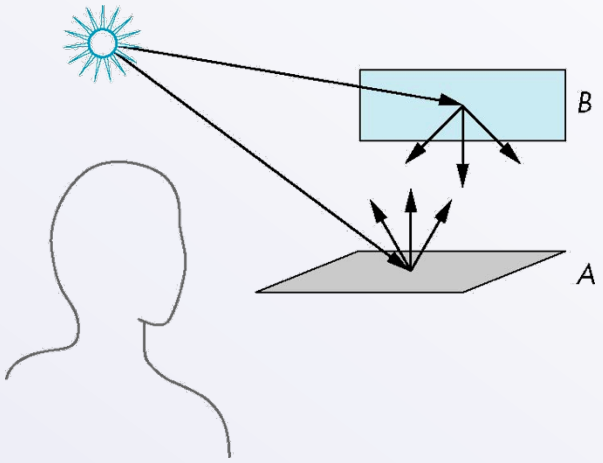
- Resultado de múltiplas interações entre fontes (grandes) de luz e objetos no ambiente
- Quantidade e cor dependem da cor das luzes e das propriedades dos materiais
- Adicionar k_a I_a aos termos difusivo e especular

Coeficiente de reflexão

Intensidade da luz ambiente

Termo de distância

- A intensidade da luz que chega à uma superfície é inversamente proporcional à distância até a fonte de luz
- Podemos adicionar o termo $1/(a + bd + cd^2)$ aos termos difusivo e especular.
- Os termos constante e linear suavizam o efeito do ponto de luz



Fontes de luz

- No modelo Phong, adicionamos os resultados de cada fonte luz
- Cada luz possui um termo difusivo, especular e de ambiente
  - Não possui justificação física!
- Três componentes: vermelho, verde e azul.
- Total de 9 coeficientes para cada ponto de luz:
  - I_{dR}, I_{dG}, I_{dB}, I_{sR}, I_{sG}, I_{sB}, I_{aR}, I_{aG}, I_{aB}

Materiais

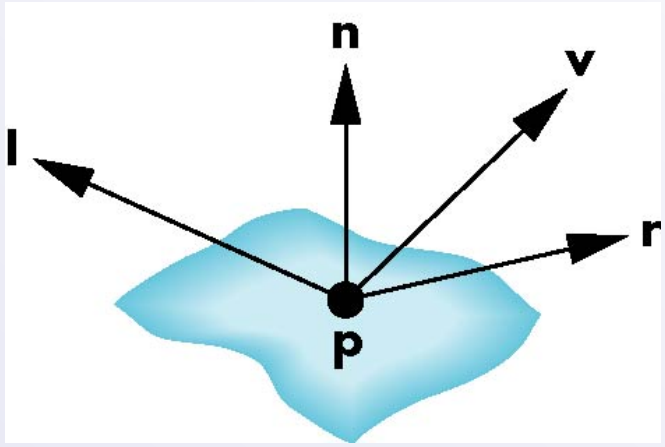
- Modelamos propriedades de materiais analogamente
  - 9 coeficientes de absorção
    - k_{dR}, k_{dG}, k_{dB}, k_{sR}, k_{sG}, k_{sB}, k_{aR}, k_{aG}, k_{aB}
- Coeficiente de brilho α

Adicionando os componentes

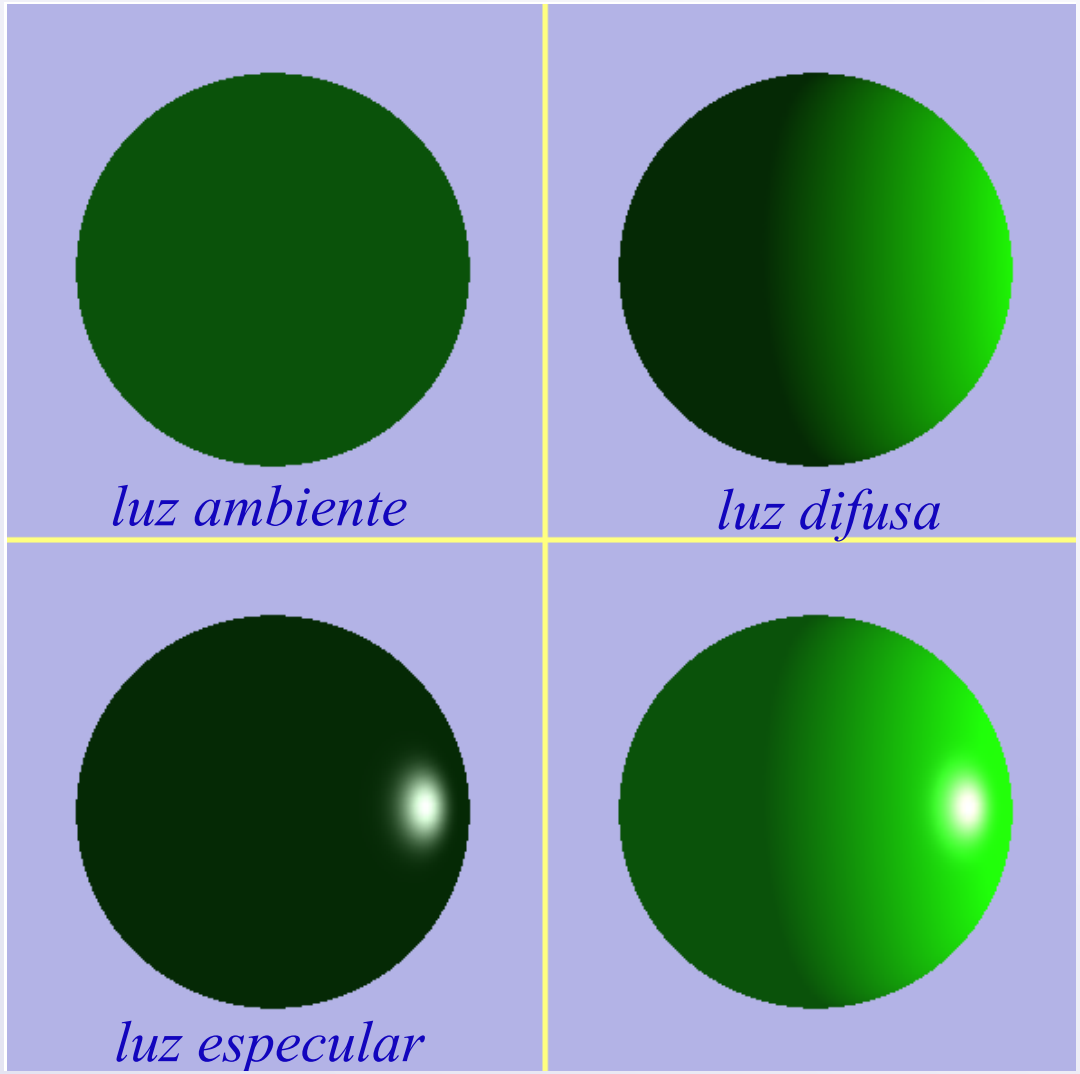
Para cada fonte de luz e componente de cor, o modelo Phong pode ser descrito como:

$$I = k_d I_d \, \mathbf{l} \cdot \mathbf{n} + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_a I_a$$

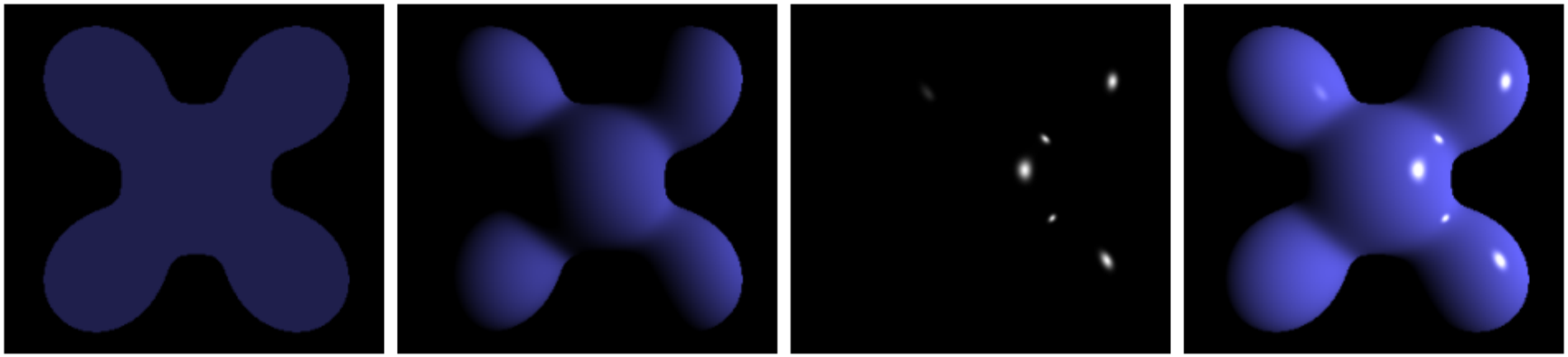
Para cada componente de cor, adicionamos contribuições de todas as fontes



Componentes do modelo Phong



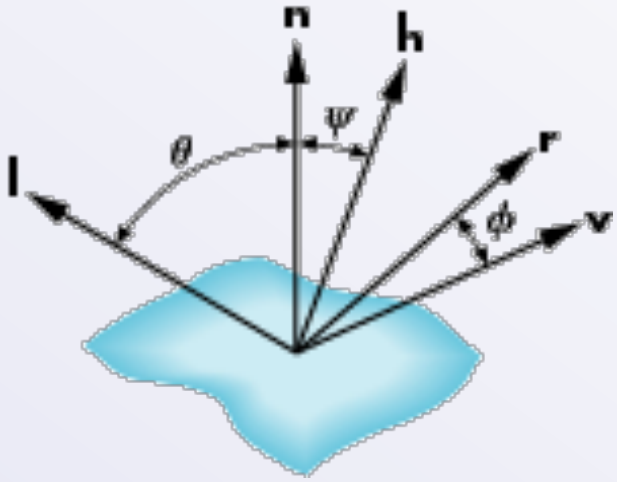
Componentes do modelo Phong



Modelo Phong modificado

- O termo especular implica que calculemos um novo vetor de reflexão ideal e o vetor do observador para cada vértice
- Blinn sugeriu uma aproximação

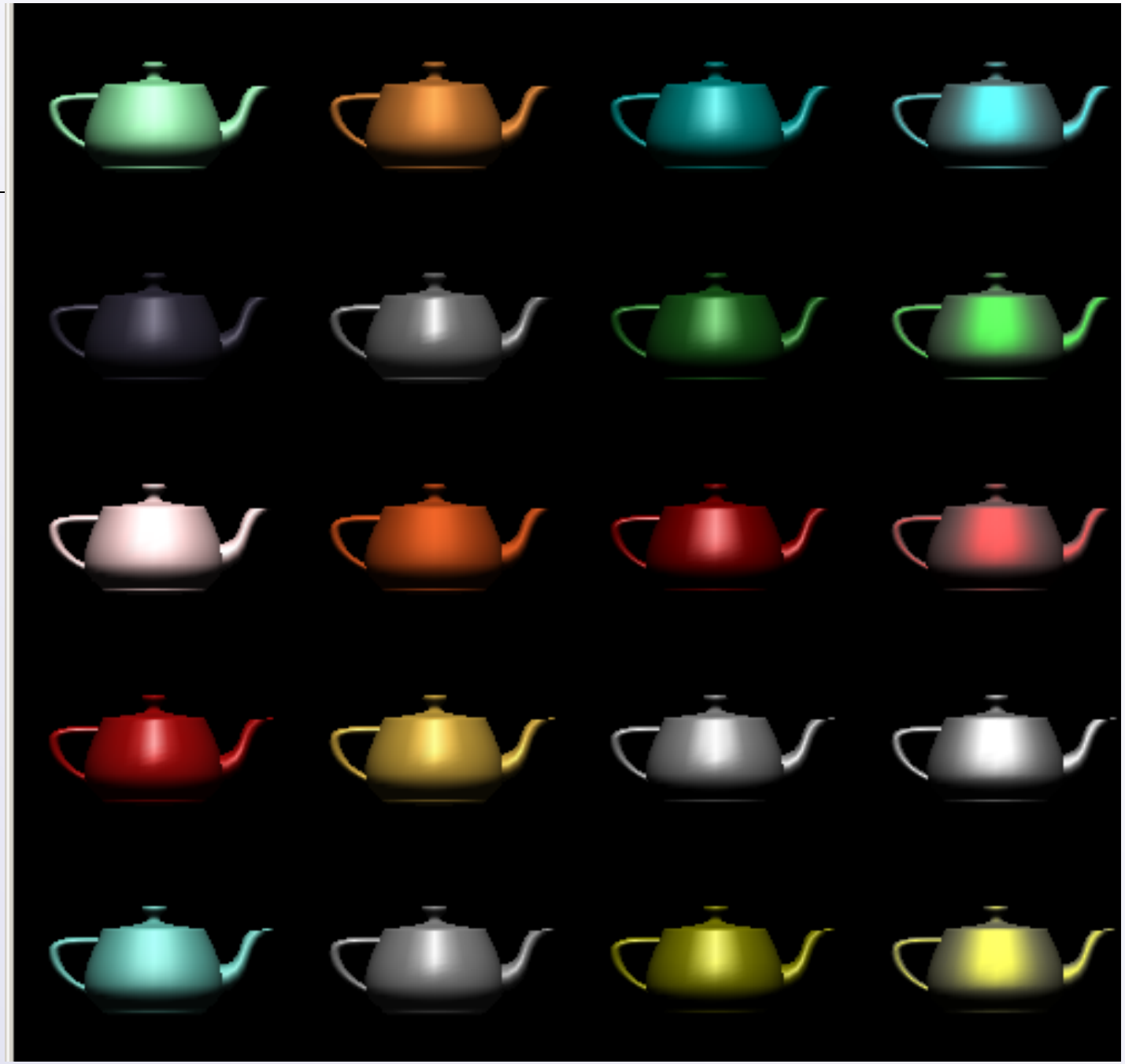
$$\mathbf{h} = (\mathbf{l} + \mathbf{v}) / |\mathbf{l} + \mathbf{v}|$$



Usando o vetor médio

- Trocar $(\mathbf{v} \cdot \mathbf{r})^{\alpha}$ por $(\mathbf{n} \cdot \mathbf{h})^{\beta}$
- β é escolhida dependendo a brilhosidade desejada
- O ângulo obtido é metade do ângulo entre $\mathbf{r}$ e $\mathbf{v}$ caso todos os vetores forem coplanares
- Este modelo de iluminação é chamado de Phong modificado ou modelo Blinn
  - Modelo padrão de iluminação do OpenGL

Exemplo



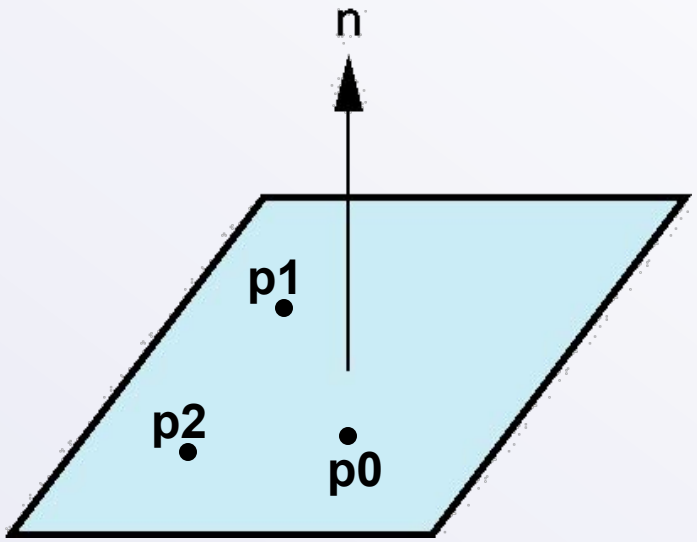
Cálculo dos vetores

- $\mathbf{l}$ e $\mathbf{v}$ são especificados pela aplicação
- Podemos calcular $\mathbf{r}$ de $\mathbf{l}$ e $\mathbf{n}$
- O problema é determinar $\mathbf{n}$
- Para superfícies simples, pode ser determinado, varia de acordo com a representação.
- OpenGL deixa a determinação das normais para a aplicação
  - Exceções: superfícies quádricas (GLU) e Bezier

Vetor normal ao plano

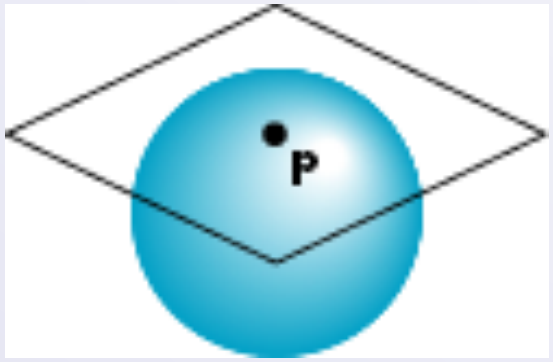
- Equação implícita do plano: $ax+by+cz+d = 0$
  - $\mathbf{n}\,(p - p_0) = 0$
- Um plano é determinado por 3 pontos p_0, p_1, p_2 ou um ponto p_0 e uma normal $\mathbf{n}$
- O vetor normal pode ser obtido através de

$$\mathbf{n} = (p_2\text{-}p_0) \times (p_1\text{-}p_0)$$



Vetor normal da esfera

- Função implícita $f(x,y,z)=0$
  - $f(x,y,z) = x^2+y^2+z^2-1 = 0$
  - $f(p) = p \cdot p - 1 = 0$
- Normal dada pelo gradiente de f
  - $n = [\partial f/\partial x, \partial f/\partial y, \partial f/\partial z]^T = 2\mathbf{p}$



Caso geral

- Podemos calcular as normais de forma paramétrica para casos simples
  - Superfícies quádricas
  - Superfícies polinomiais paramétricas
- Em superfícies complexas, que normalmente são feitos com faces triangulares, calculamos as normal de cada face.

Tonalização em OpenGL

1. Abilitar tonalização e selecionar o modelo;
2. Especificar vetores normais;
3. Especificar propriedades dos materiais;
4. Especificar luzes;

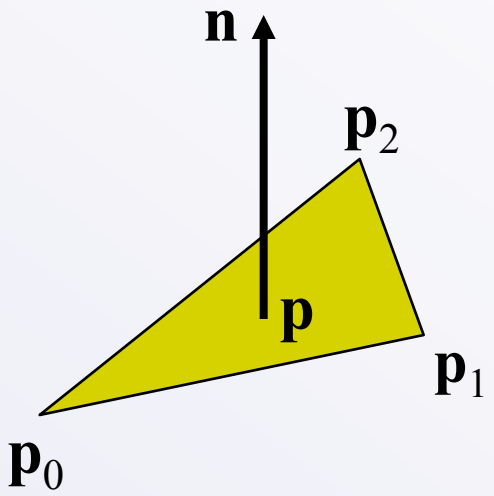
Normais

- Especificado através de `glNormal*()`
  - `glNormal3f(x, y, z);`
  - `glNormal3fv(p);`
- Comprimento deve ser unitário, |**n**|=1
  - Transformações podem afetar o comprimento
  - `glEnable(GL_NORMALIZE)` resulta na degradação na performance
- O vetor normal é parte do estado;

Normal de um triângulo

$$\vec{n} = (p_2 - p_0) \times (p_1 - p_0)$$

$$\vec{n}' = \frac{\vec{n}}{|\vec{n}|}$$

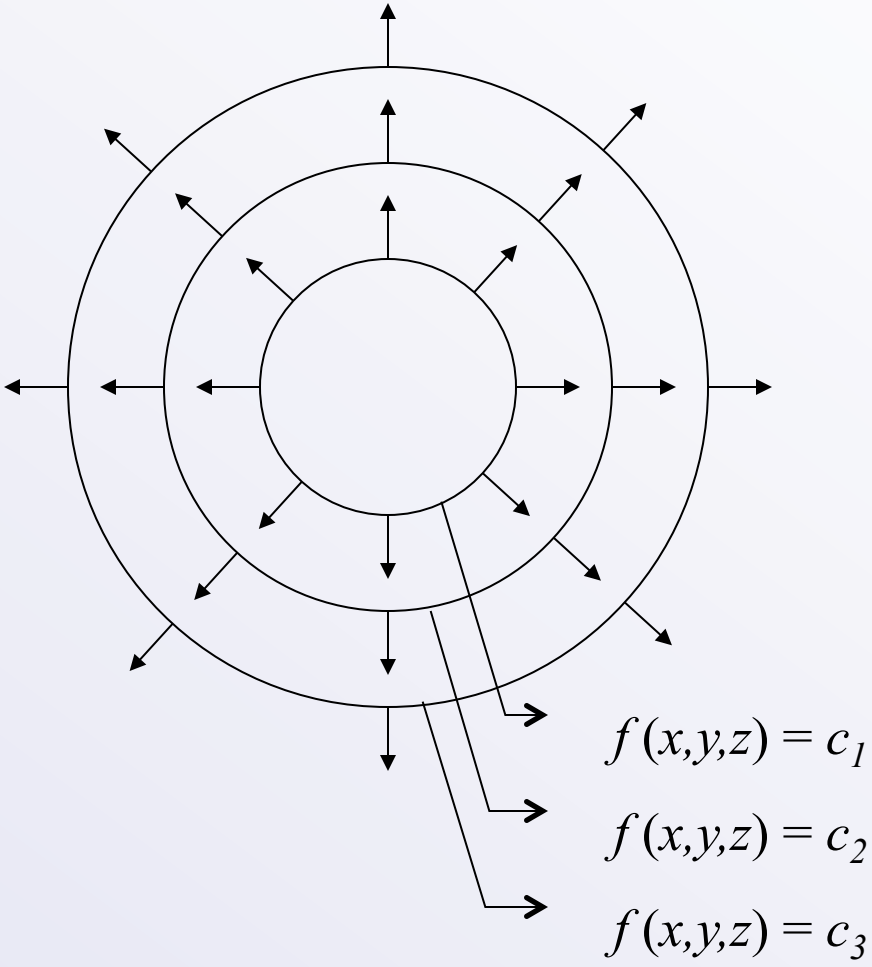


Calculando o Vetor Normal de Superfícies Implícitas

- Normal é dada pelo vetor gradiente

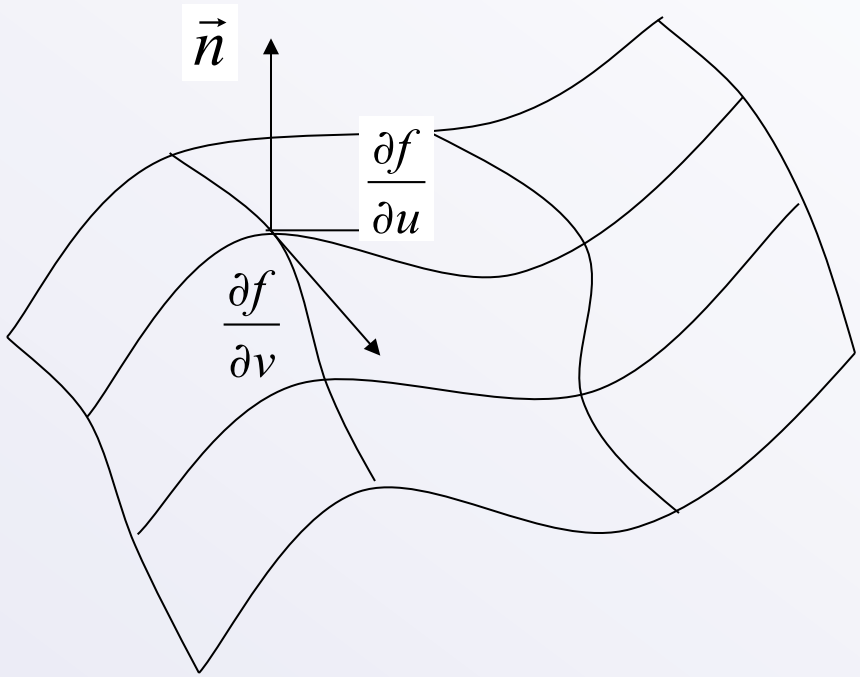
$$f(x,y,z) = 0$$

$$\vec{n} = \begin{pmatrix} \partial f / \partial x \\ \partial f / \partial y \\ \partial f / \partial z \end{pmatrix}$$



Calculando o Vetor Normal de Superfícies Paramétricas

- Normal é dada pelo produto vetorial dos gradientes em relação aos parâmetros *u* e *v*



$$P = \begin{pmatrix} f_x(u,v) \\ f_y(u,v) \\ f_z(u,v) \end{pmatrix}$$

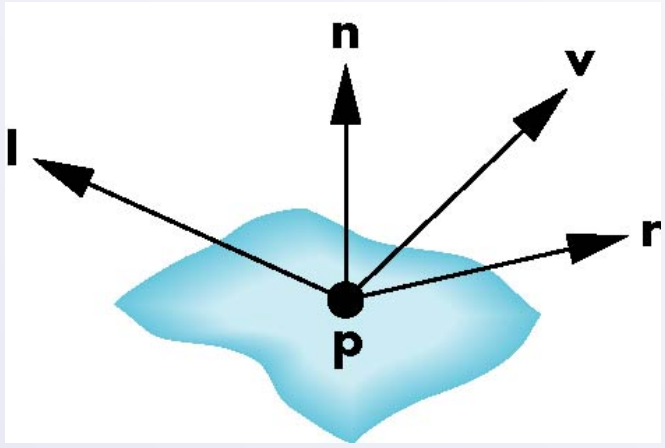
$$\vec{n} = \frac{\partial f}{\partial u} \times \frac{\partial f}{\partial v} = \begin{pmatrix} \partial f_x / \partial u \\ \partial f_y / \partial u \\ \partial f_z / \partial u \end{pmatrix} \times \begin{pmatrix} \partial f_x / \partial v \\ \partial f_y / \partial v \\ \partial f_z / \partial v \end{pmatrix}$$

Modelo de Phong

Para cada fonte de luz e componente de cor, o modelo Phong pode ser descrito como:

$$I = k_d I_d \, \mathbf{l} \cdot \mathbf{n} + k_s I_s (\mathbf{v} \cdot \mathbf{r})^{\alpha} + k_a I_a$$

Para cada componente de cor, adicionamos contribuições de todas as fontes



Tonalização

- Habilitação através do comando
  - `glEnable(GL_LIGHTING)`
  - Comandos `glColor()` são ignorados
- Cada luz deve ser ligada individualmente
  - `glEnable(GL_LIGHTi)` i=0,1,…,7
- Parâmetros do modelo de iluminação
  - `glLightModeli(parameter, GL_TRUE)`
    - `GL_LIGHT_MODEL_LOCAL_VIEWER` observador não está posicionado no infinito;
    - `GL_LIGHT_MODEL_TWO_SIDED` tonaliza ambos os lados dos polígonos independentemente

Ponto de luz

- Para cada fonte de luz, especificamos intensidades RGBA para cada um dos componentes: difuso, especular e ambiente, e sua posição/direção

```
GLfloat luz0difusa[]={1.0, 0.0, 0.0, 1.0};
GLfloat luz0ambiente[]={1.0, 0.0, 0.0, 1.0};
GLfloat luz0espec[]={1.0, 0.0, 0.0, 1.0};
Glfloat luz0posicao[]={1.0, 2.0, 3,0, 1.0};

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glLightv(GL_LIGHT0, GL_POSITION, luz0pos);
glLightv(GL_LIGHT0, GL_AMBIENT, luz0ambiente);
glLightv(GL_LIGHT0, GL_DIFFUSE, luz0difusa);
glLightv(GL_LIGHT0, GL_SPECULAR, luz0espec);
```

Distância

- O OpenGL também permite especificar a rapidez com que a intensidade diminui em relação a distância da fonte

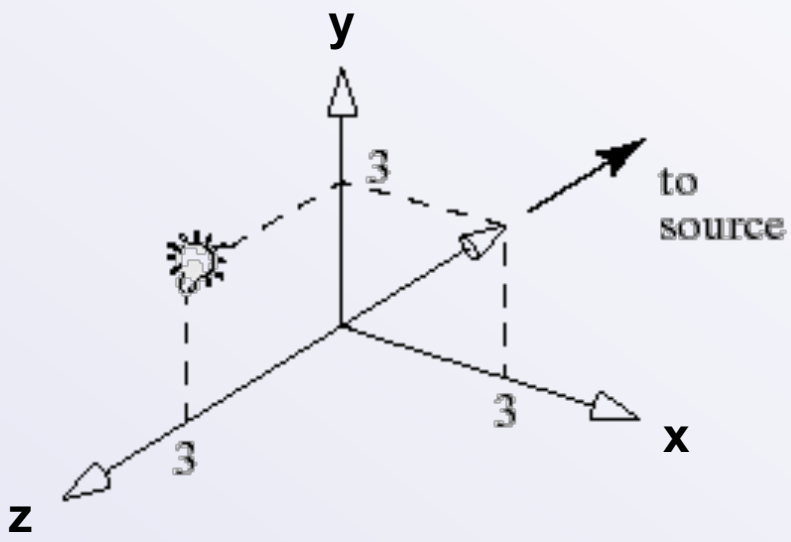
$$atenuação = \frac{1}{a + bx + cx^2}$$

- Onde a,b e c são os coeficientes e x é distância entre posição da luz e do vértice em questão (default: a=1, b=c=0)

```
glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, a);
glLightf(GL_LIGHT0, GL_LINEAR_ATTENUATION, b);
glLightf(GL_LIGHT0, GL_QUADRATIC_ATTENUATION, c);
```

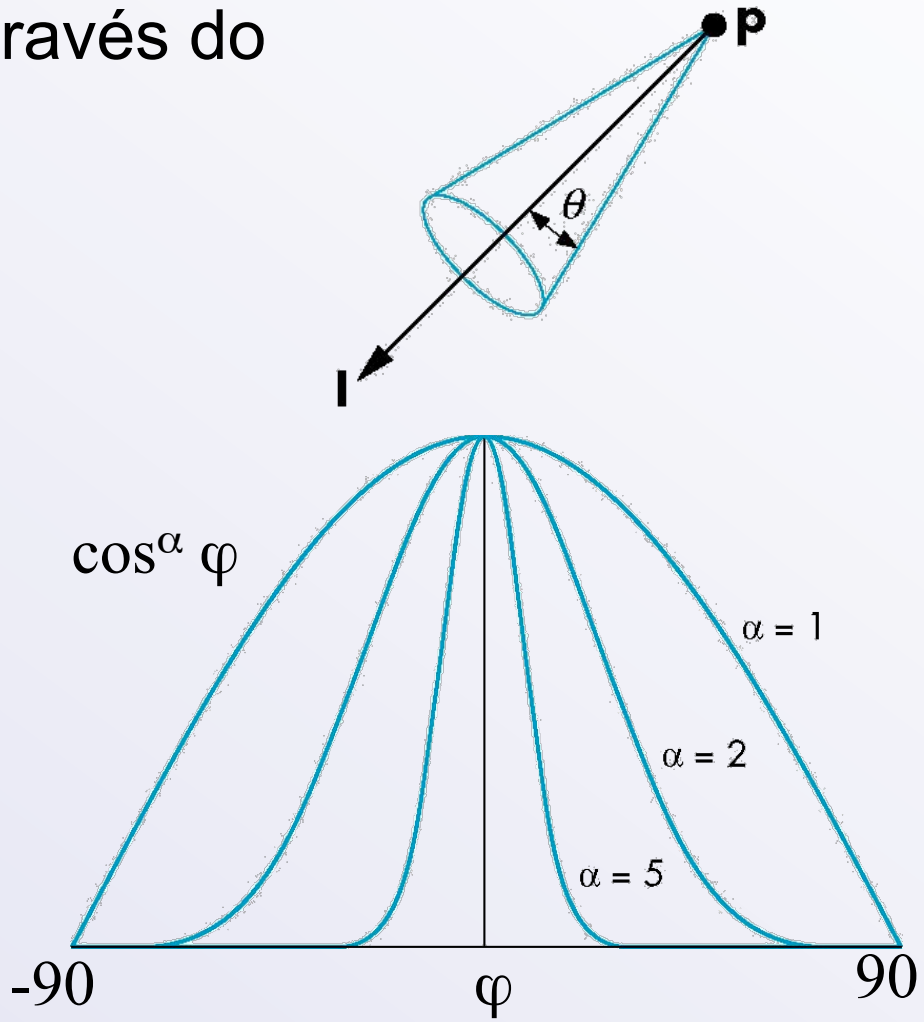
Luzes direcionais

- A figura mostra uma fonte local em (0, 3, 3, 1) e uma fonte distante apontada pelo vetor (3, 3, 0, 0).
- Fontes infinitamente distantes são chamadas “direcionais”
- A direção **l** nos cálculos dos termos difusivo e especular se mantém constantes para todos os vértices da cena.
- As luzes direcionais nem sempre são a melhor opção para conseguir os melhores efeitos visuais.



Luzes spot

- Também é especificada através do comando `glLightv()`
- Direção:
  - `GL_SPOT_DIRECTION`
- Ângulo de corte:
  - `GL_SPOT_CUTOFF`
  - Default: 180º
- Concentração:
  - `GL_SPOT_EXPONENT`
  - Proporcional a $\cos^{\alpha} \varphi$



Usando luzes spot

- Valores default **d** = {0, 0, 0, 1}, φ = 180º, α = 0
- Exemplo:
  - `glLightf (GL_LIGHT_0, GL_SPOT_CUTOFF, 45.0); // (45.0 é` φ `em graus)`
  - `glLightf (GL_LIGHT_0, GL_SPOT_EXPONENT, 4.0); // (4.0 é` α`)`
  - `glLightfv (GL_LIGHT_0, GL_SPOT_DIRECTION, d);`

Luz ambiente

- Um termo global de luz ambiente está presente mesmo se nenhuma luz for criada. A sua cor default é {0.2, 0.2, 0.2, 1.0}.
- Não depende da geometria
- Em OpenGL podemos especificar um termo global de ambiente através da função
  - **`glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient)`**
- Os valores default para termos ambiente são {0, 0, 0, 1} para todas as luzes

Posição do observador

- OpenGL calcula as reflexões especulares utilizando o vetor médio **h**
- As direções verdadeiras do observador **v** e da luz **l** serão diferentes para cada vértice
- Para aumentar a velocidade de renderização, o OpenGL utiliza o vetor **v** = (0, 0, 1)
- Para utilizar o vetor verdadeiro, devemos executar

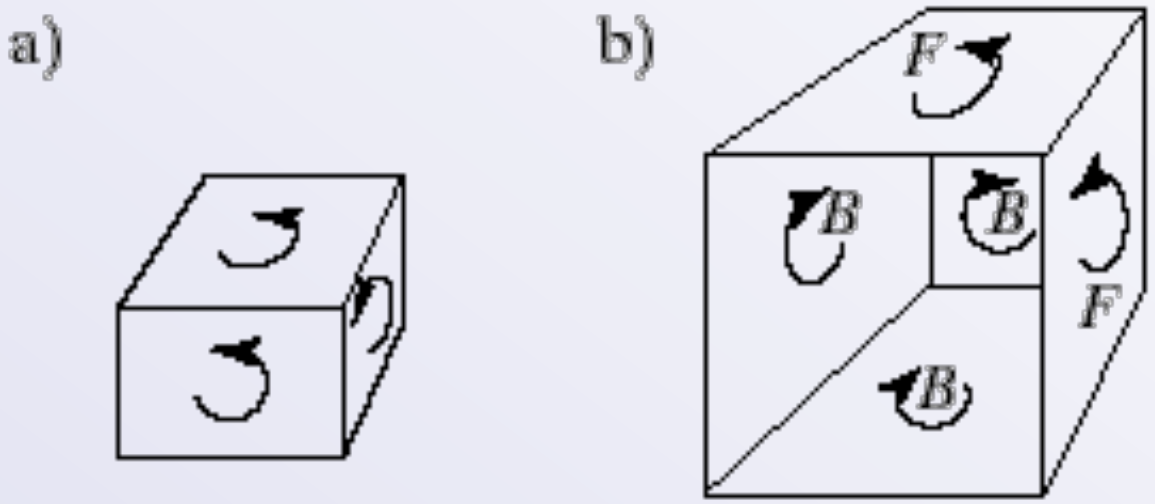
```
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,
GL_TRUE);
```

Modelo de iluminação

- Cada face poligonal possui dois lados
- O OpenGL não entende o que é “interior” ou “exterior”. Pode somente distinguir entre faces frontais e faces traseiras
- Assumimos que os vértices são definidos no sentido anti-horário
- Uma face é dita como face frontal se os seus vértices são definidos no sentido anti-horário em relação ao observador

Modelo de iluminação

- `glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);`



Transformações

- Fontes de luz também são afetadas pela matriz de modelo;
- Dependendo do tipo da transformação, podemos
  - Mover as luzes com os objetos
  - Mover somente as luzes
  - Mover somente os objetos
  - Mover as luzes e os objetos independentemente

Movimento independente

```
void display()
{
  GLfloat position[] = {2,1,3,1}; // posição inicial
   // limpa buffers
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   glPushMatrix();
   glRotated(...); glTranslated(...); // move luz
   glLightfv(GL_LIGHT0, GL_POSITION, position);
   glPopMatrix();
   gluLookAt(...); // seleciona câmera
    // Desenha objeto
    glutSwapBuffers();
}
```

Movimento dependente

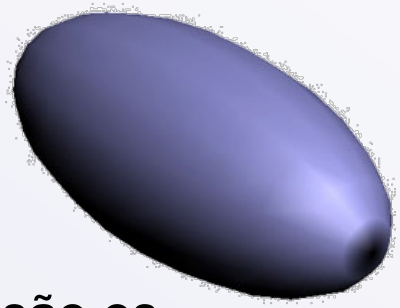
```
GLfloat pos[ ] = {0,0,0,1};
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
// luz posicionada em (0,0,0)
glLightfv(GL_LIGHT0, GL_POSITION, position);
// movimenta luz e câmera
gluLookAt(...);
// desenha objeto
```

Demo lightposition.c

Demo movelight.c

Tarefa de casa

- Uma elipsóide é descrita pelas seguintes equações paramétricas:
  - $x = a \cos(\phi) \cos(\theta)$
  - $y = b \cos(\phi) \operatorname{sen}(\theta)$
  - $z = c \operatorname{sen}(\phi)$



Onde $\phi \in [-\pi/2, +\pi/2]$ e $\theta \in [-\pi, +\pi]$, e a,b,c são os comprimentos de seus semi-eixos

  - Calcule a forma paramétrica do seu vetor normal **n**