# CCI 36 – Computação Gráfica

# OpenGL – Parte 3

**Instituto Tecnológico de Aeronáutica**

**Prof. Carlos Henrique Q. Forster – Sala 121 IEC**

**forster@ita.br**

**Luiz Felipe Simões Hoffmann**

## Tópicos da Aula

- Texturas
- Modelos de Iluminação
- Sombreamento (*Shading*)
- Carregamento de Objetos
- Exemplos

## Referências

Dave Shreiner, Graham Sellers, John Kessenich, Bill Licea-Kane. OpenGL Programming Guide, 8th ed., Pearson Education, 2013.

Richard S. Wright Jr., Nicholas Haemel, Graham Sellers, Benjamin Lipchak. OpenGL Super Bible, 5th ed., Pearson Education, 2011.

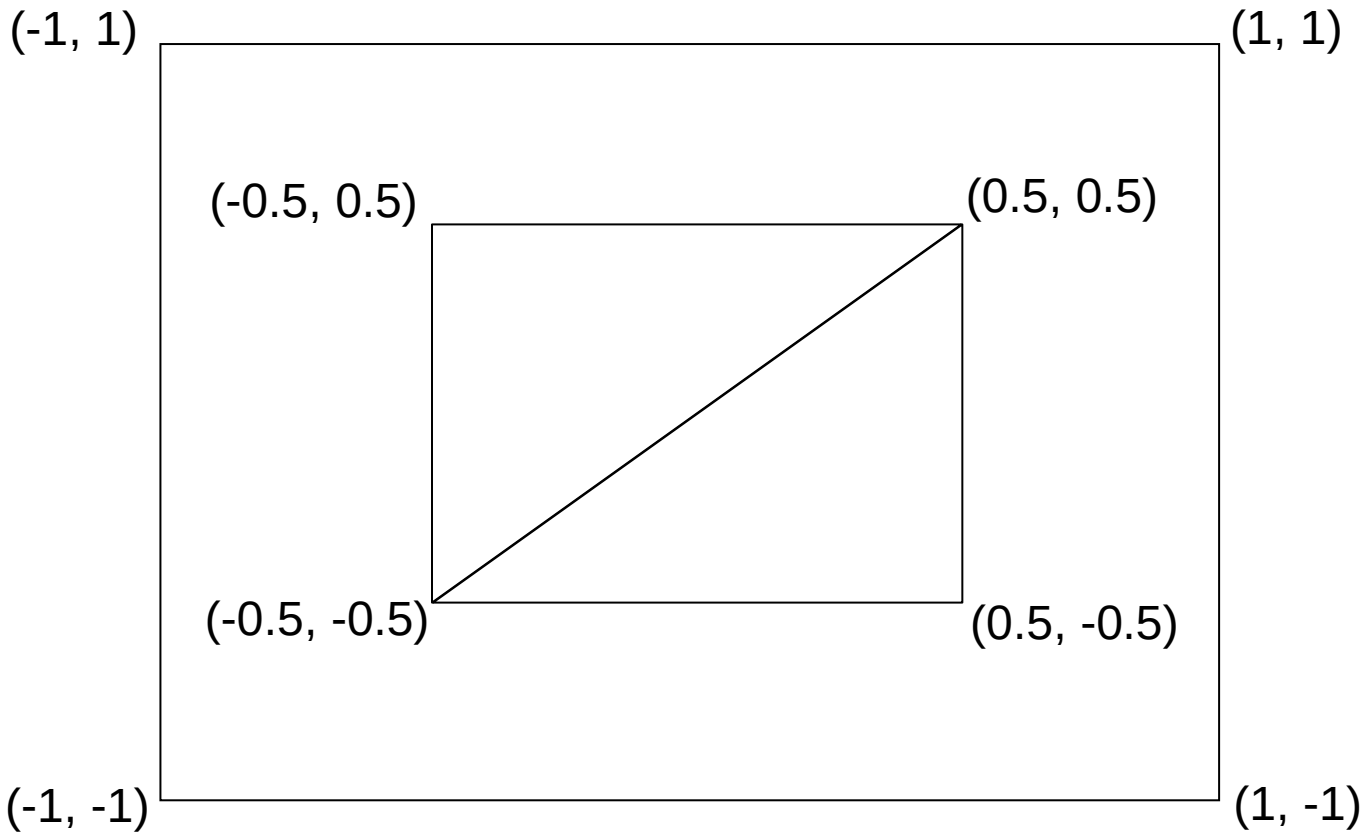
Joey de Vries. Learn OpenGL, Joey de Vries, 2015.

## Texturas

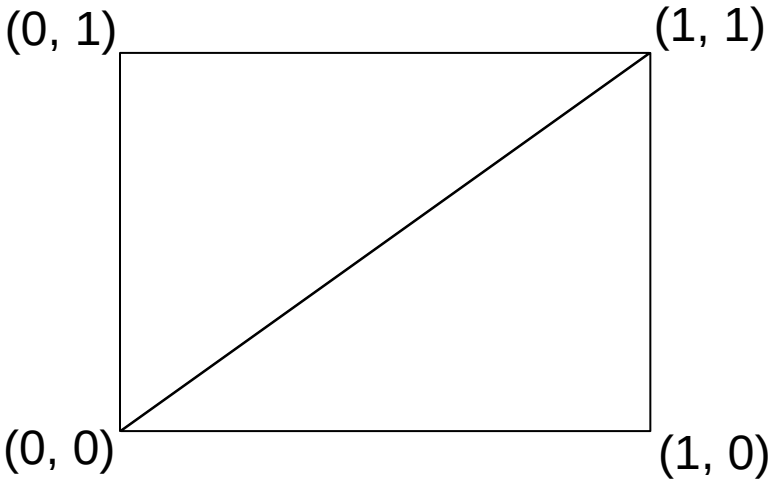
São imagens utilizadas para adicionar detalhes aos objetos.

As texturas também podem ser usadas para armazenar uma grande coleção de dados que deve ser enviada aos *shaders*.

## Coordenadas Normalizadas do Dispositivo (*Normalized Device Coordinates* – NDC)



## Coordenada de Textura (*x*, *y*, *z* -> *s*, *t*, *r*)



## Exemplo: Textura

Passo 1: definir os vértices, as coordenadas de textura e os índices.

```
#                       Posições    Texturas
vertices = np.array([-0.5,  0.5, 0.0, 1.0,  # Superior Esquerdo
                      0.5,  0.5, 1.0, 1.0,  # Superior Direito
                     -0.5, -0.5, 0.0, 0.0,  # Inferior Esquerdo
                      0.5, -0.5, 1.0, 0.0   # Inferior Direito
                    ], dtype=np.float32)

indices = np.array([0, 1, 2,                # Primeiro Triângulo
                    1, 2, 3                 # Segundo Triângulo
                   ], dtype=np.uint8)
```

## Exemplo: Textura

Passo 2: alocar memória na GPU.

```
# Cria um vertex buffer object
vbo = glGenBuffers(1)
...

# Cria um element buffer object
ebo = glGenBuffers(1)
...

# Cria um texture object
texture = glGenTextures(1)
glBindTexture(GL_TEXTURE_2D, texture)
```

## Exemplo: Textura

Passo 3: definir os parâmetros de repetição de padrões (*wrapping*).

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)
```

## Exemplo: Textura

Passo 4: definir os parâmetros de filtragem.

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)
```


GL_NEAREST GL_LINEAR

## Exemplo: Textura

Passo 5: carregar a imagem e especificar a textura.

```
image = Image.open('img/crate0.jpg')
img_data = np.array(list(image.getdata()), np.uint8)
glTexImage2D(GL_TEXTURE_2D,                # Tipo de textura
             0,                            # Nível
             GL_RGB,                       # Formato para a textura
             image.width,                  # Largura
             image.height,                 # Altura
             0,                            # Sempre 0
             GL_RGB,                       # Formato da imagem
             GL_UNSIGNED_BYTE,             # Tipo da imagem
             img_data                      # Imagem
            )
glEnable(GL_TEXTURE_2D)
```

## Exemplo: Textura

Passo 6: vincular os vértices e a textura.

```
# Posição
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE,
                      4 * vertices.itemsize,
                      None)
glEnableVertexAttribArray(0)

# Textura
glVertexAttribPointer(1, 2, GL_FLOAT, GL_FALSE,
                      4 * vertices.itemsize,
                      ctypes.c_void_p(2 * vertices.itemsize))
glEnableVertexAttribArray(1)
```

## Exemplo: Textura

Passo 7: criar o *vertex shader*.

```
VERTEX_SHADER = '''#version 330 core
layout (location = 0) in vec2 position;
layout (location = 1) in vec2 texcoord;
out vec2 Texcoord;
void main()
{
    Texcoord = texcoord;
    gl_Position = vec4(position, 0.0, 1.0);
}'''
```

## Exemplo: Textura

Passo 8: criar o *fragment shader*.

```
FRAGMENT_SHADER = '''#version 330 core
in vec2 Texcoord;
out vec4 color;
uniform sampler2D tex;
void main()
{
    color = texture(tex, Texcoord);
}'''
```

## Exemplo: Textura

Passo 9: desenhar o retângulo.

```
glDrawElements(GL_TRIANGLES,                    # Tipo de primitiva
               6,                               # Número de índices
               GL_UNSIGNED_BYTE,                # Tipo dos índices
               None                             # Ponteiro
              )
```

## Exemplo: Textura

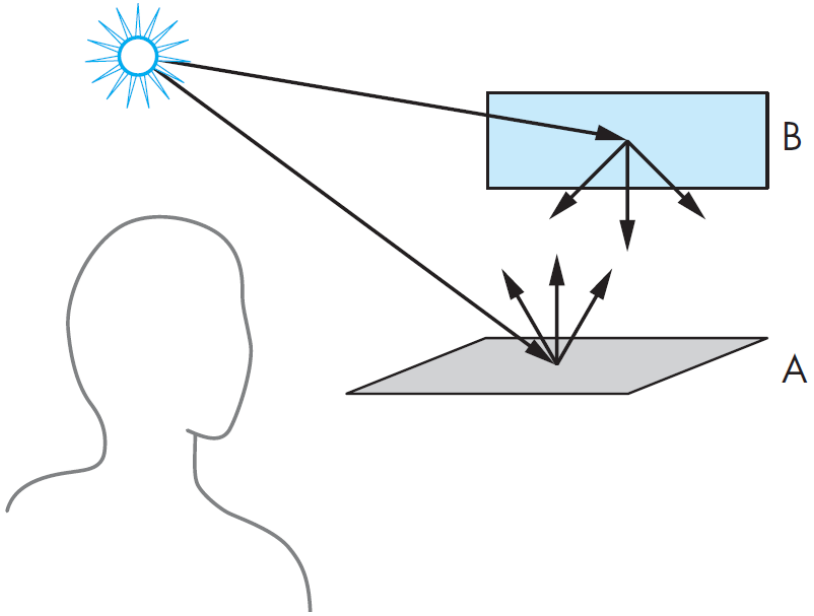
# Modelos de Iluminação

Os modelos de iluminação são utilizados para calcular a cor e intensidade da luz que atinge um ponto em uma superfície de um objeto. Servem para tornar as imagens mais realistas.

Em muitos casos, os modelos de iluminação são mais importantes do que a correta definição de perspectiva para tornar uma cena compreensível.

Dois tipos principais de modelos:

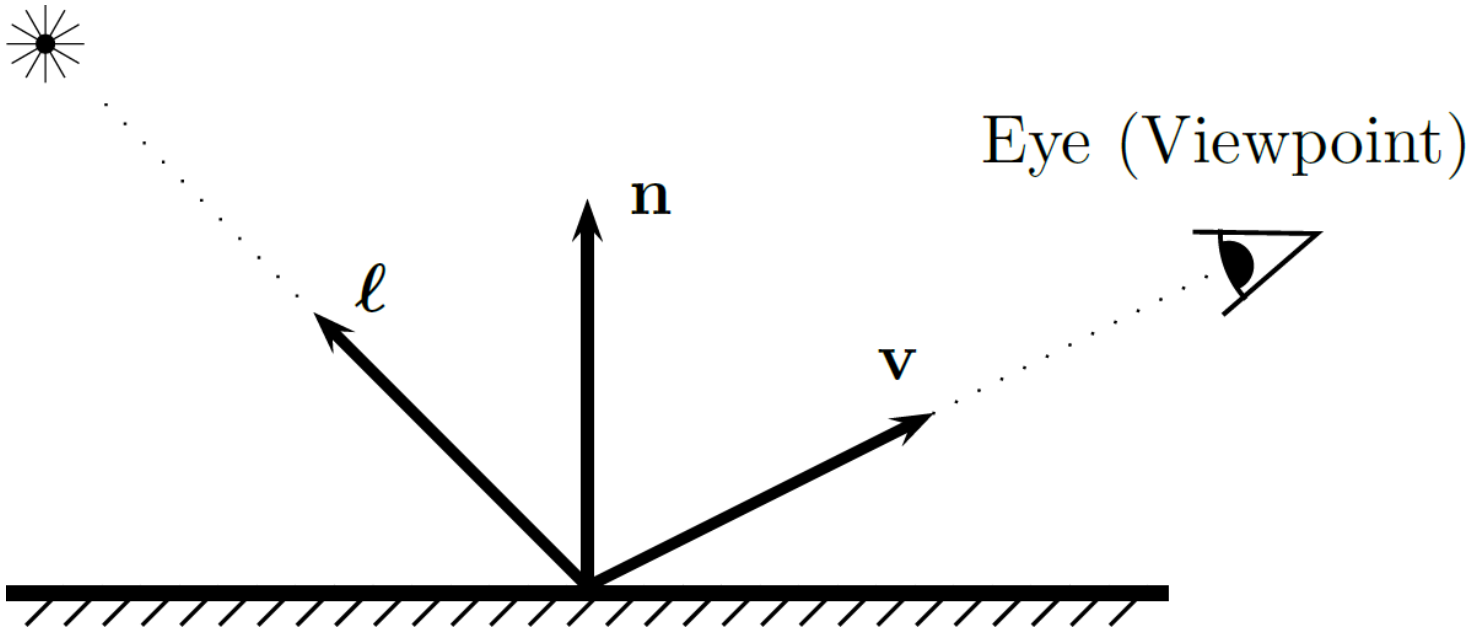
- Modelos de iluminação local: não considera múltiplas reflexões.
- Modelos de iluminação global: considera múltiplas reflexões.



## Modelos de Iluminação Local

Esses modelos utilizam os seguintes elementos para calcular a intensidade e cor de um ponto em uma superfície:

- Propriedades do material (cor e geometria da superfície).
- Propriedades da luz (cor e posição).
- Posições do ponto e do observador.



## Fontes de Luz

A cor e intensidade de um ponto em uma superfície resultam da soma das contribuições das fontes de luz que iluminam essa superfície.

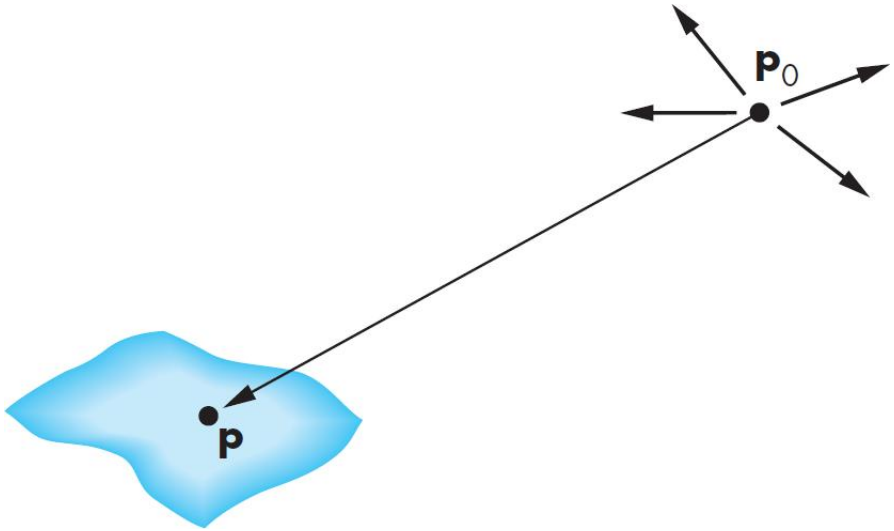
Quatro tipos padrão de fontes:

- Ambiente.
- Pontual.
- Direcional ou Holofote (*Spot*)
- Distante.

## Fontes de Luz

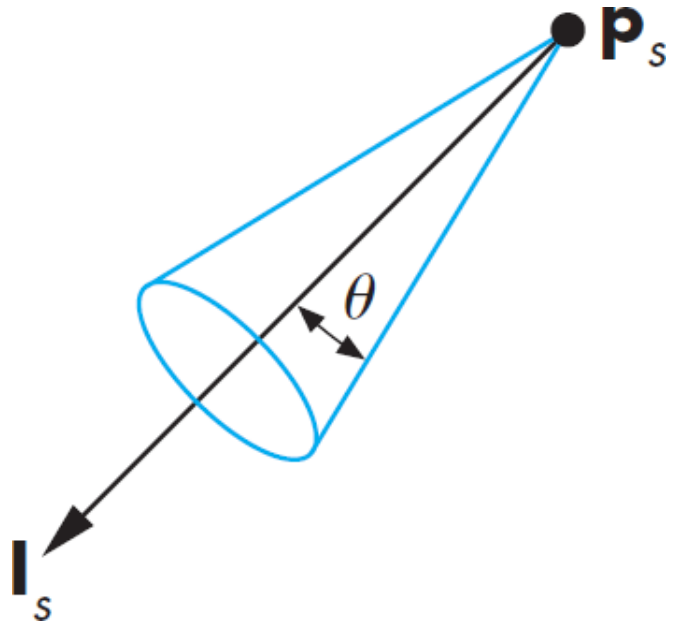
Luz Ambiente: é um tipo de fonte de luz, de baixa intensidade e uniforme, que atinge uma superfície igualmente, a partir de várias direções. Tem como objetivo modelar a luz que se espalhou em um ambiente, por meio de múltiplas reflexões.

Luz Pontual: irradia luz igualmente em todas as direções, a partir de um único ponto no espaço.



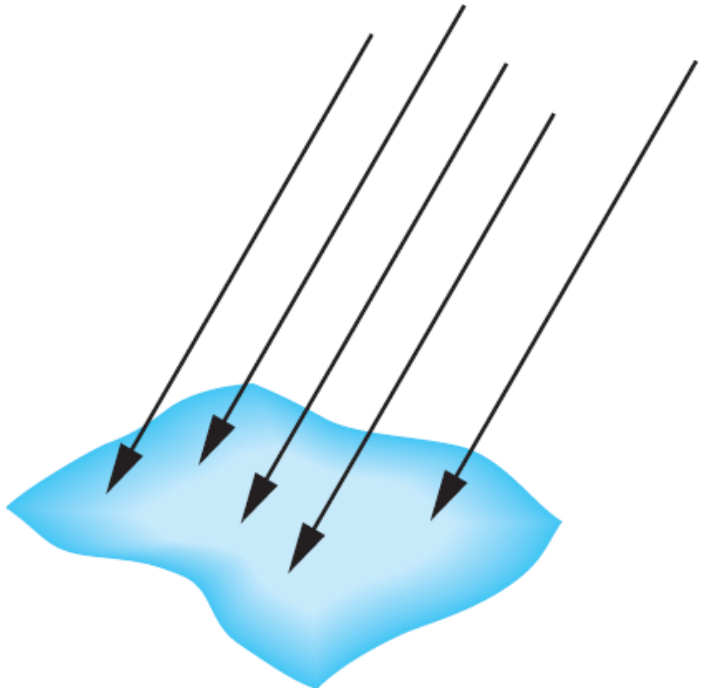
## Fontes de Luz

Luz Direcional ou Holofote (*Spot*): é caracterizada por uma faixa estrita (limite angular) na qual a luz é emitida. Pode ser construída a partir de uma fonte pontual, com limitação dos ângulos, em que a luz pode ser vista.



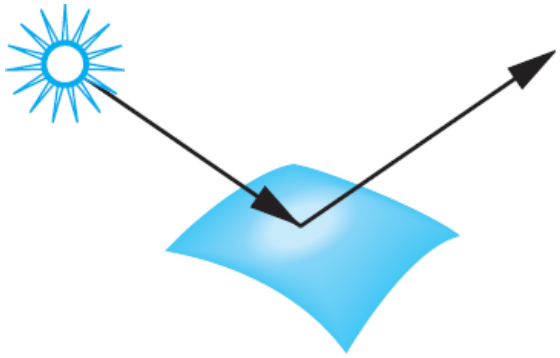
## Fontes de Luz

Luz Distante: é um tipo de fonte de luz disposta a certa distância de uma superfície, de modo que os raios de luz chegam paralelamente à mesma.

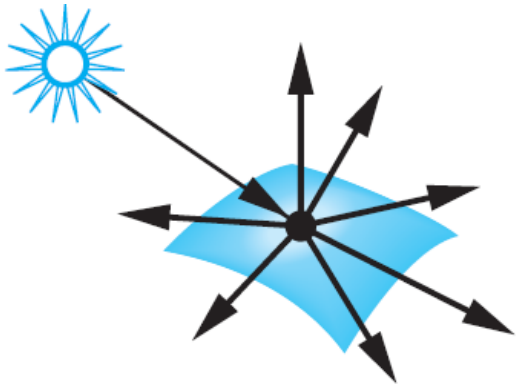


## Efeitos de Luz em Superfícies

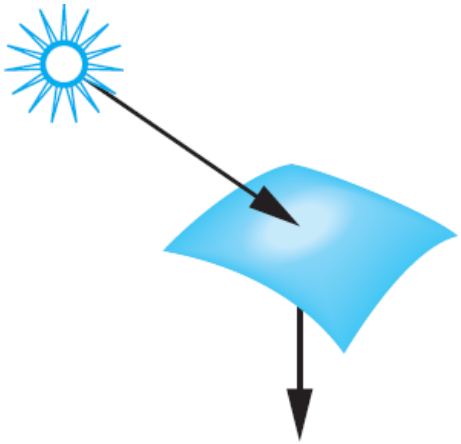
A luz que incide em uma superfície pode ser absorvida, refletida ou refratada.



Superfície Especular



Superfície Difusa



Superfície Translúcida

## Modelo de Iluminação Local de Phong

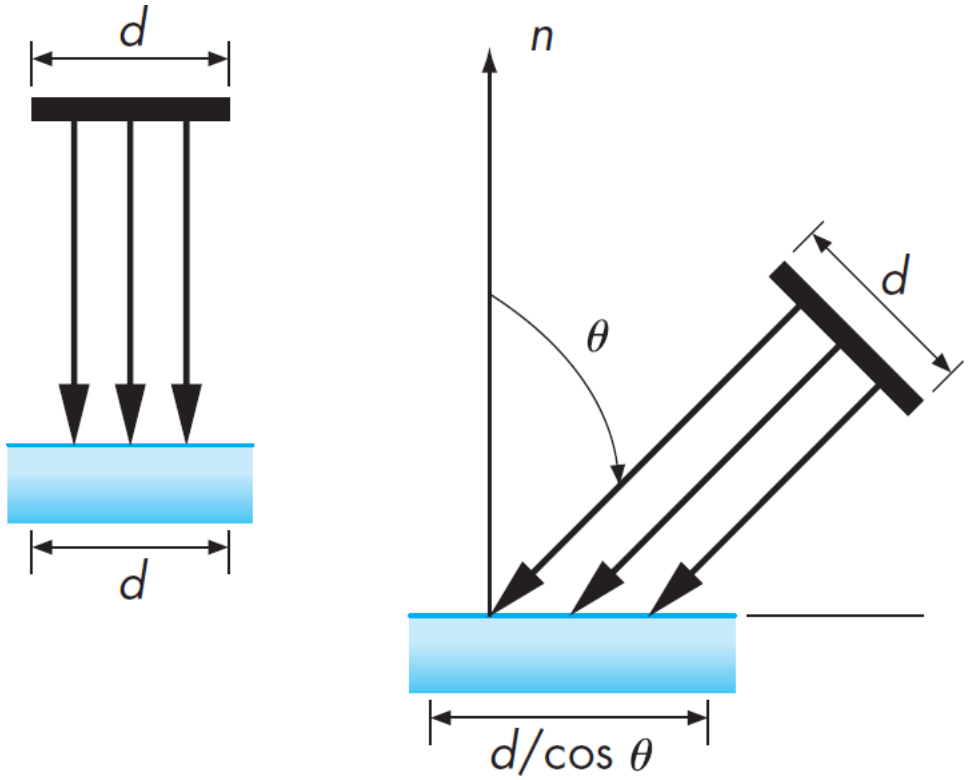
O modelo de iluminação local de Phong é o modelo mais popular utilizado em computação gráfica.

Esse modelo tem se mostrado eficiente e produz resultados com aproximações realistas que consideram diferentes condições de iluminação e propriedades de materiais.

Considera três tipos de reflexão: difusa, especular e ambiente.

## Reflexão Difusa ou Reflexão Lambertiana

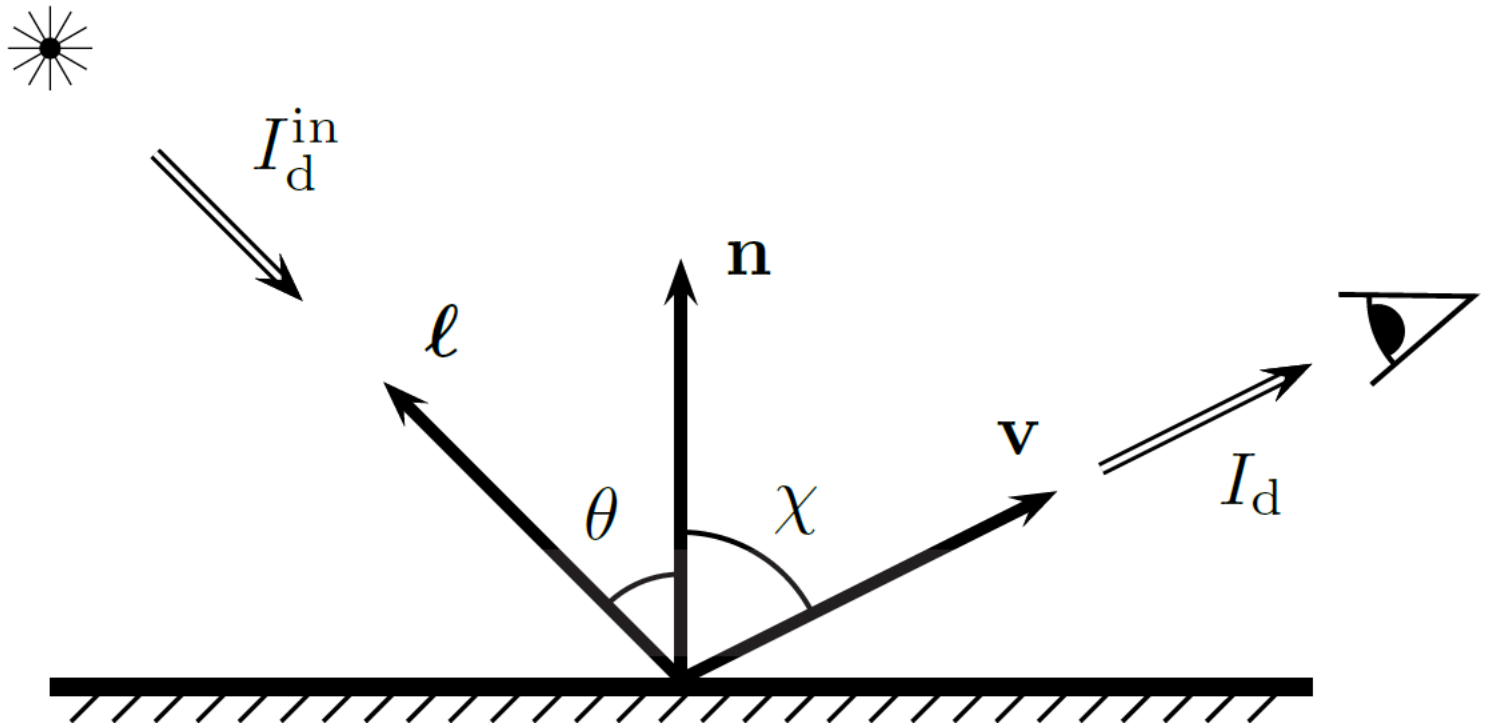
A luz é refletida igualmente em todas as direções, independentemente da posição do observador. No entanto, a quantidade de luz refletida depende do material e da posição da fonte. São caracterizadas por superfícies ásperas.



## Reflexão Difusa ou Reflexão Lambertiana

A intensidade da luz difusamente refletida pode ser modelada conforme abaixo. Nesse caso, $\rho_d$ é o coeficiente de reflexão difusa (constante que depende do material).

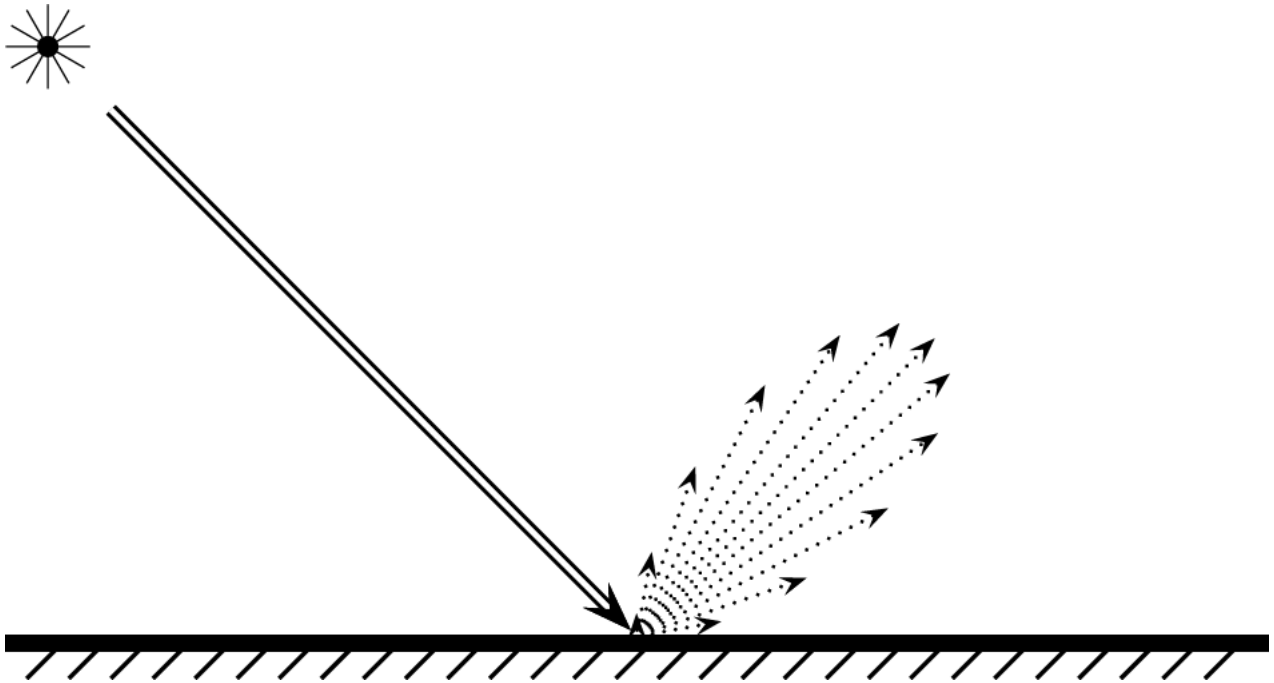
$$I_d = \rho_d I_d^{in} \cos\theta = \rho_d I_d^{in} (\boldsymbol{l} \cdot \mathbf{n})$$



## Reflexão Especular

Ocorre quando o ângulo de incidência da luz é igual ao ângulo de reflexão (espelho). É utilizada para modelar superfícies brilhantes.

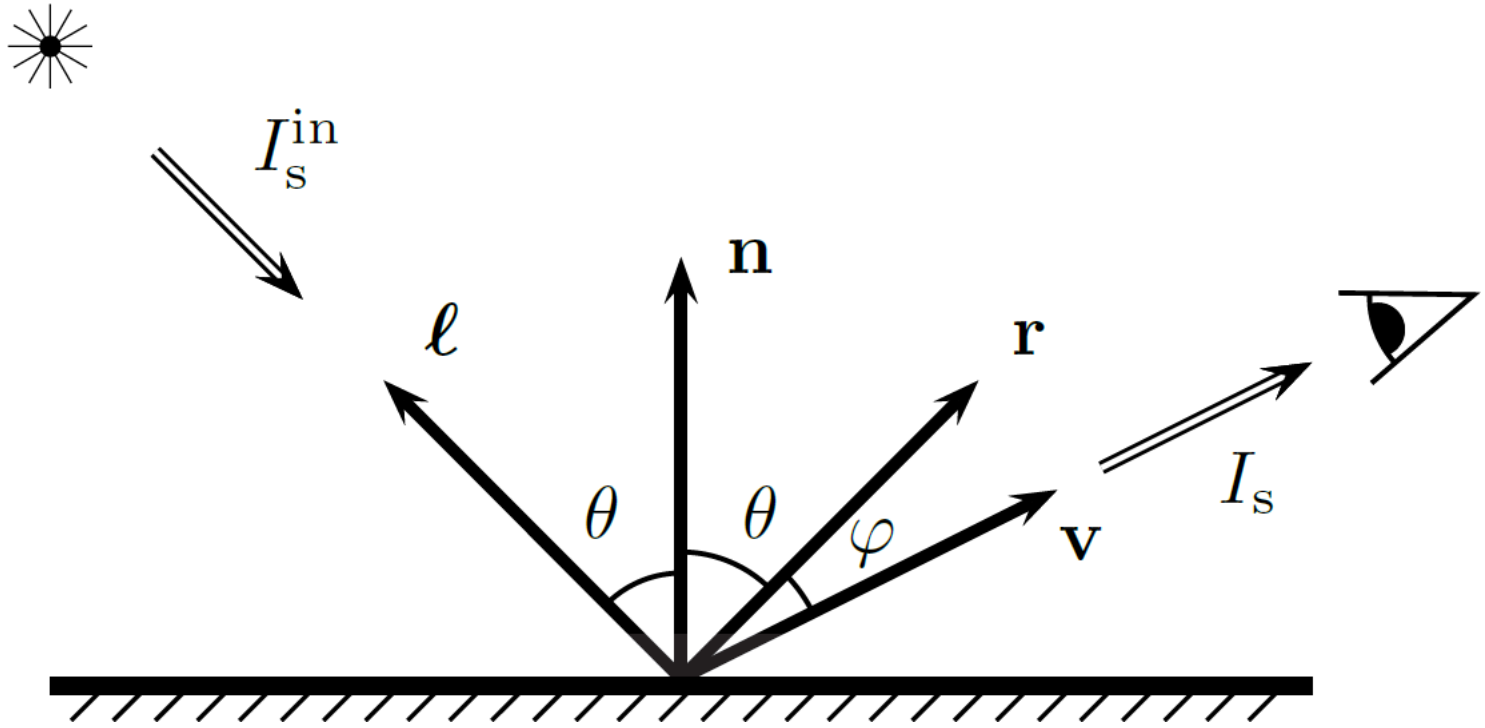
De modo geral, as superfícies brilhantes não refletem luz como o espelho. Sendo assim, a luz refletida especularmente se espalha em várias direções.



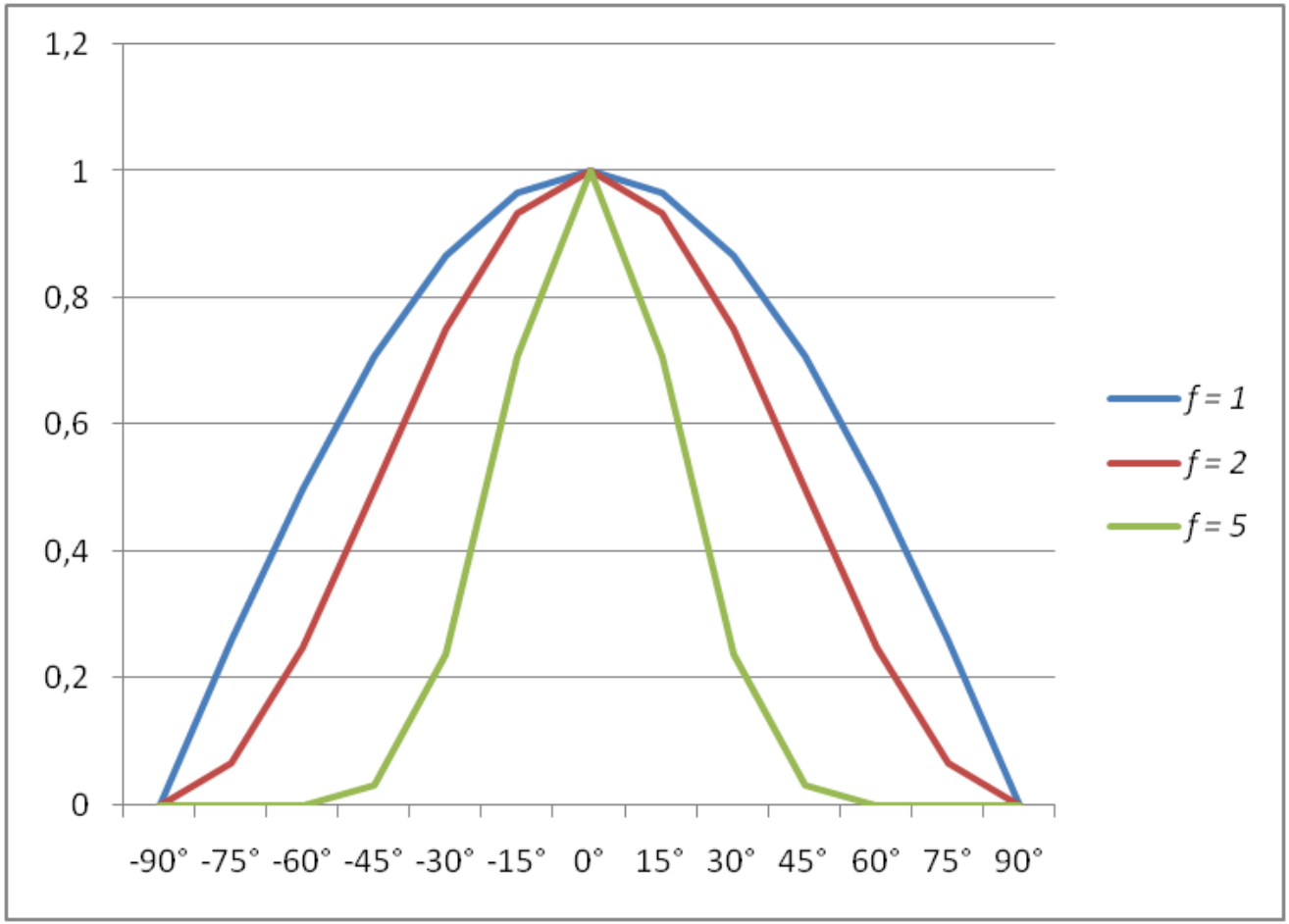
## Reflexão Especular

A intensidade da luz especularmente refletida pode ser modelada conforme abaixo. $\rho_s$ e *f* representam, respectivamente, o coeficiente e expoente de reflexão especular.

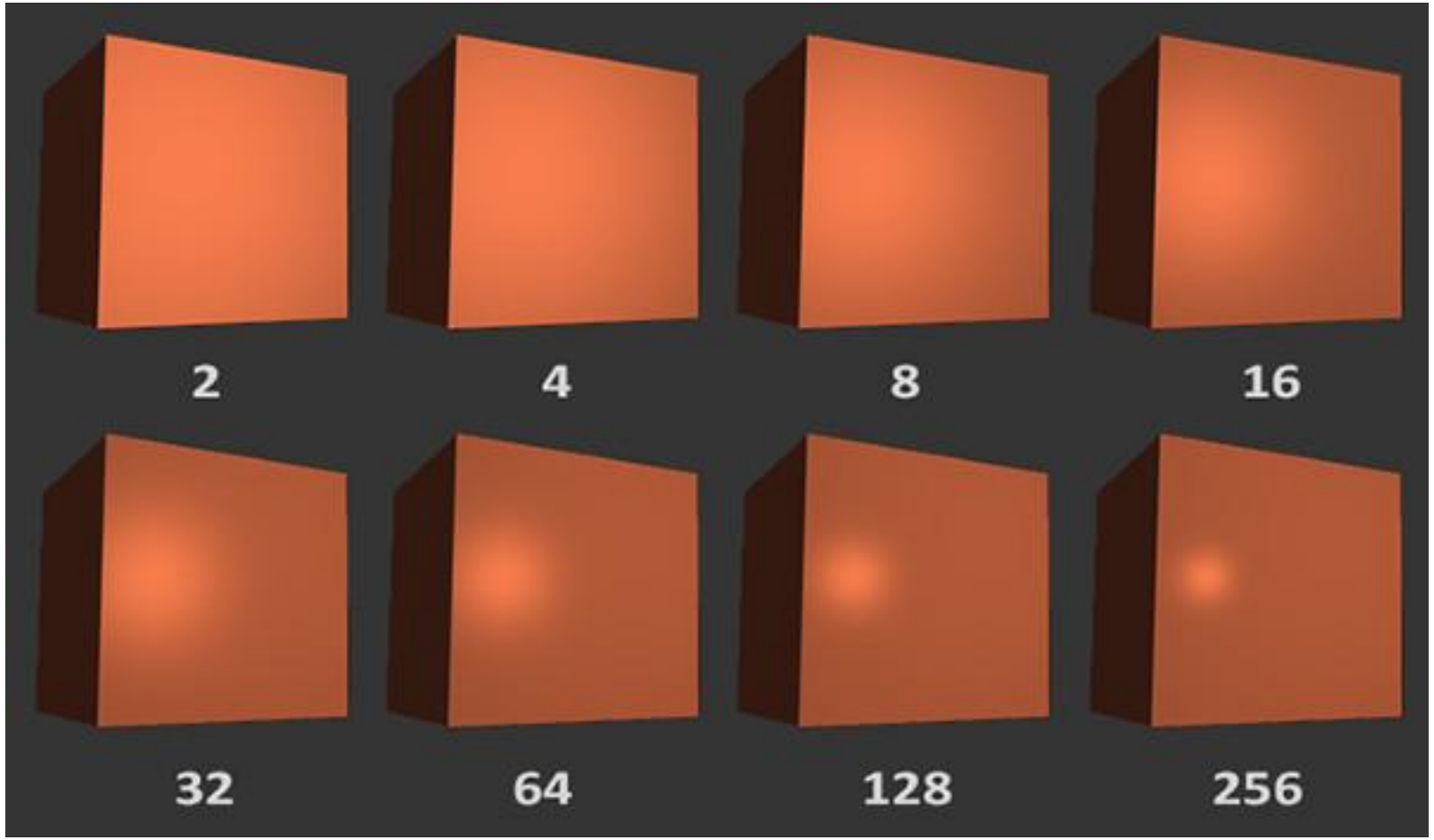
$$I_s = \rho_s I_s^{in} (\cos\varphi)^f = \rho_s I_s^{in} (\mathbf{v} \cdot \mathbf{r})^f \quad \therefore \quad \mathbf{r} = 2(\boldsymbol{l} \cdot \mathbf{n})\mathbf{n} - \boldsymbol{l}$$



# Efeito do Expoente de Reflexão Especular



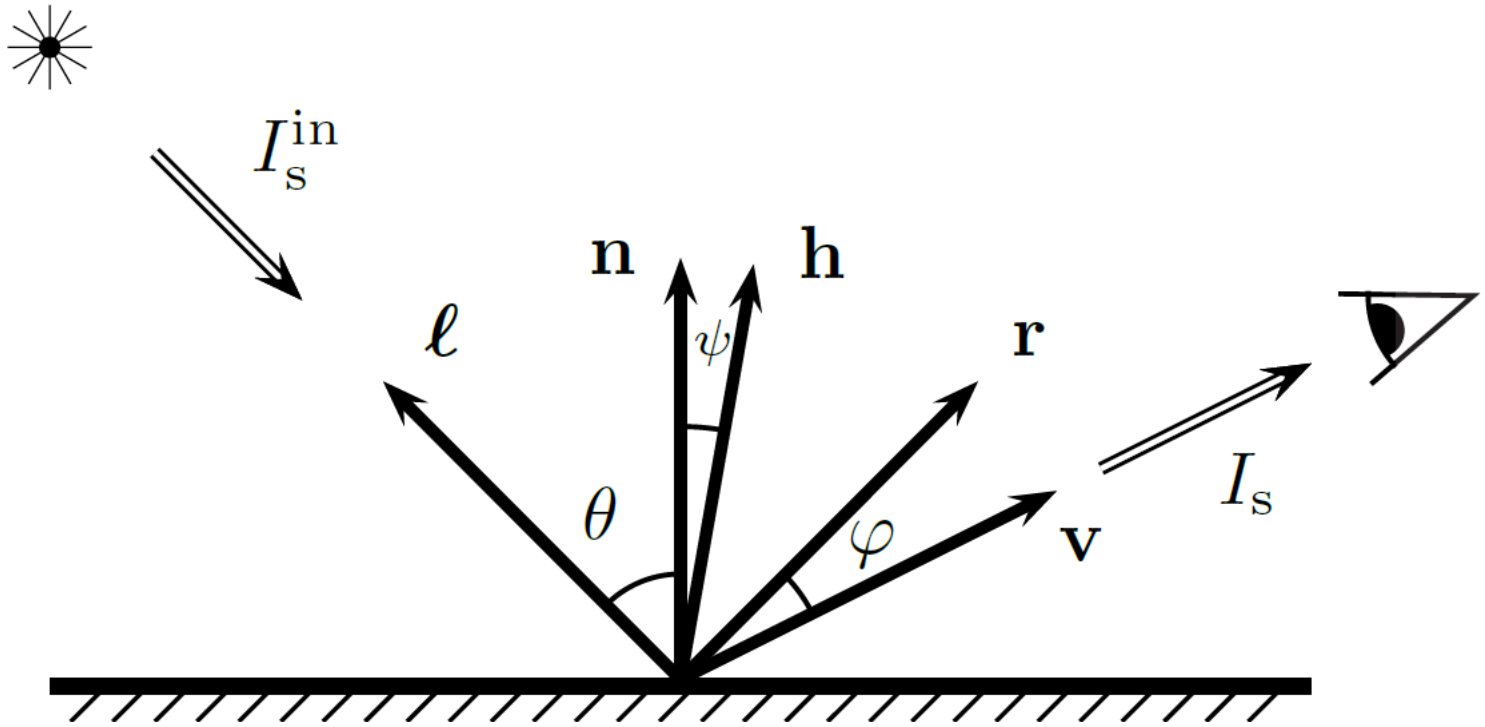
## Efeito do Expoente de Reflexão Especular



## Modelo de Iluminação Local de Blinn-Phong

A aproximação de Blinn utiliza o vetor *halfaway* (**h**):

$$I_s = \rho_s I_s^{in} (\cos\psi)^f = \rho_s I_s^{in} (\mathbf{h} \cdot \mathbf{n})^f \quad \therefore \quad \mathbf{h} = \frac{\boldsymbol{l} + \mathbf{v}}{\|\boldsymbol{l} + \mathbf{v}\|}$$



## Reflexão do Ambiente e Emissividade

Reflexão do Ambiente:

Luz uniforme resultante de múltiplas reflexões. Pode ser modelada como:

$$I_a = \rho_a I_a^{in}$$

$\rho_a$ é o coeficiente de reflexão do ambiente.

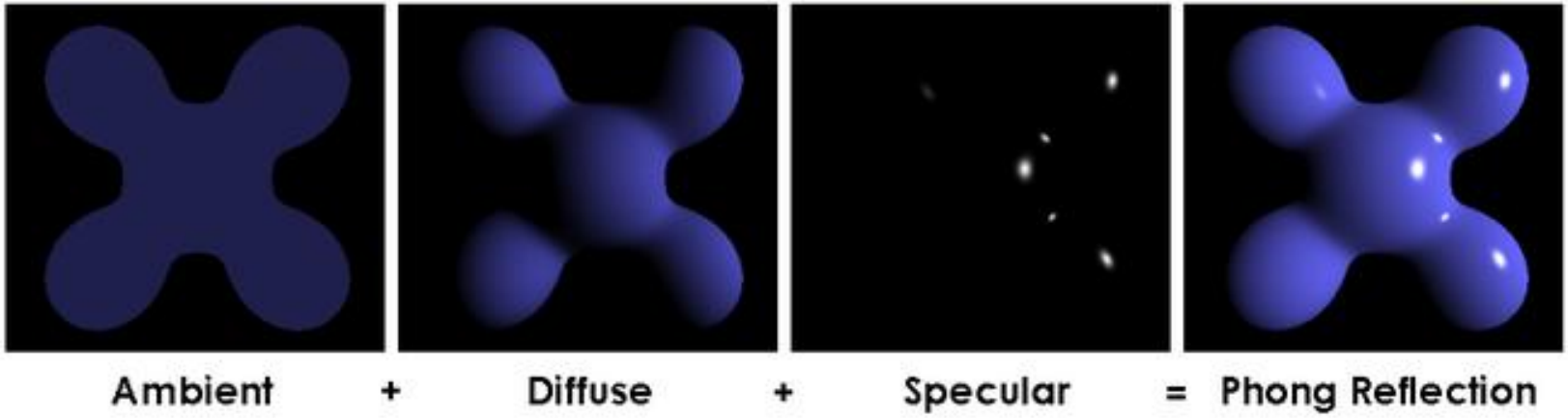
Emissividade:

É a intensidade de luz ($I_e$) emitida pela superfície, em acréscimo a luz refletida.

## Modelo de Iluminação Phong – Equação Final

Considerando apenas uma fonte de luz, a intensidade total refletida por uma superfície é:

$$I = I_a + I_d + I_s + I_e$$

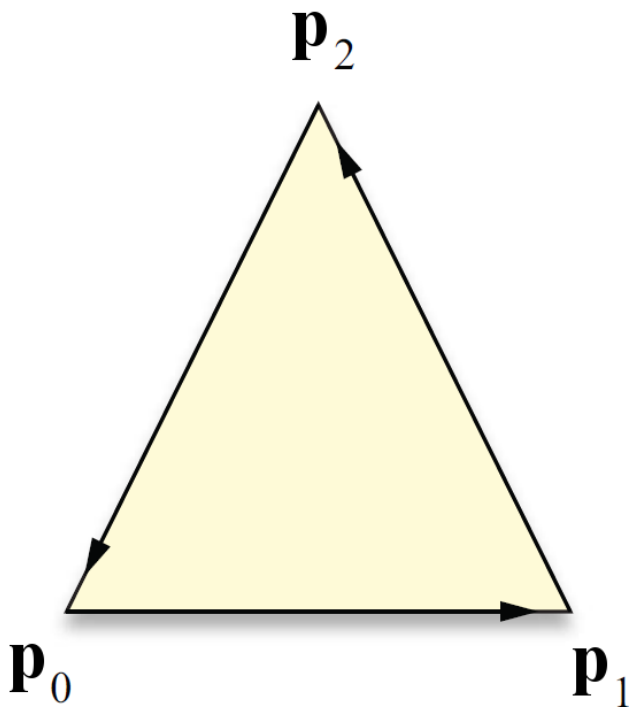


## Vetores Normais

É importante a definição correta dos valores dos vetores normais para a obtenção de bons resultados nos efeitos de iluminação.

A normal de um triângulo pode ser calculada por:

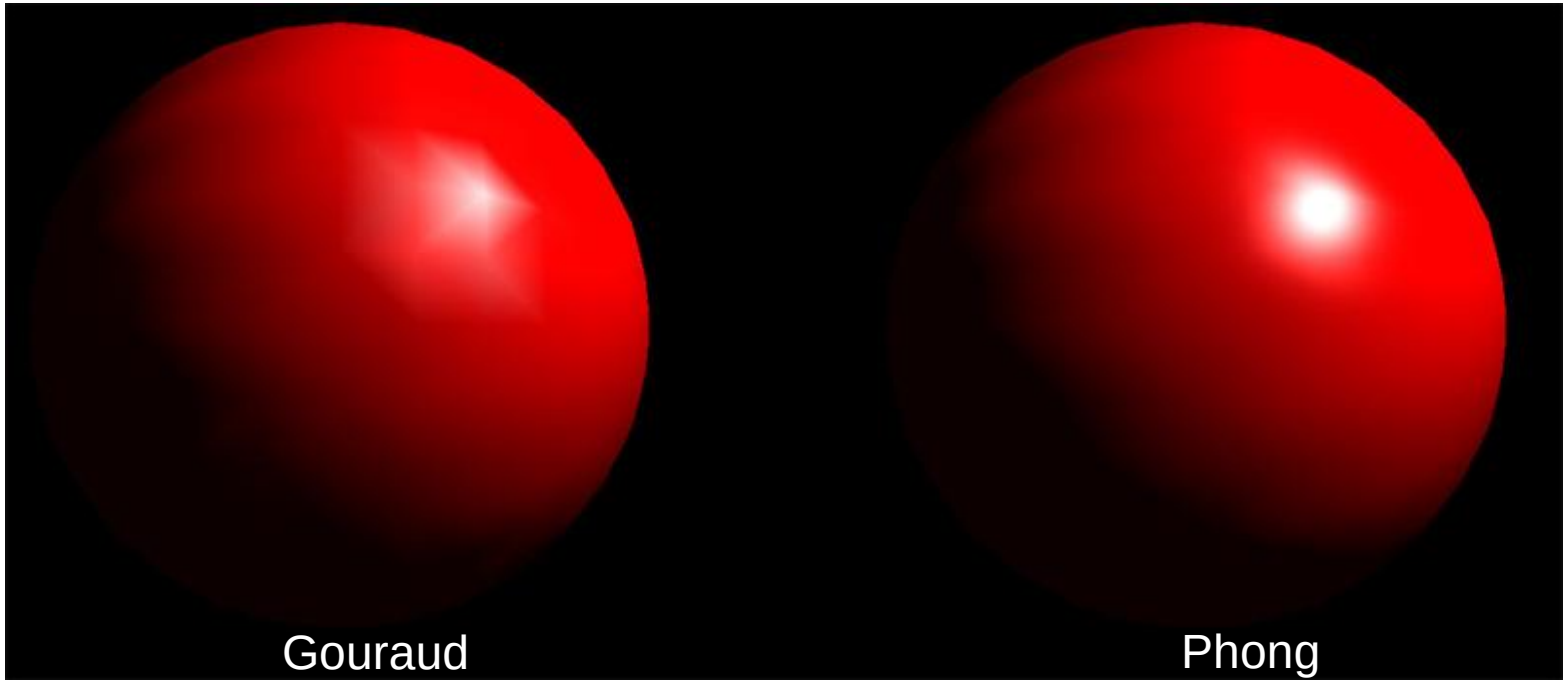
$$\mathbf{n} = \frac{(\mathbf{p_1} - \mathbf{p_0}) \times (\mathbf{p_2} - \mathbf{p_0})}{\|(\mathbf{p_1} - \mathbf{p_0}) \times (\mathbf{p_2} - \mathbf{p_0})\|}$$



## Sombreamento (*Shading*)

Sombreamento (*Shading*) é a técnica utilizada, por meio de interpolação, para criar um padrão de cor e brilho de variação suave nas superfícies de objetos.

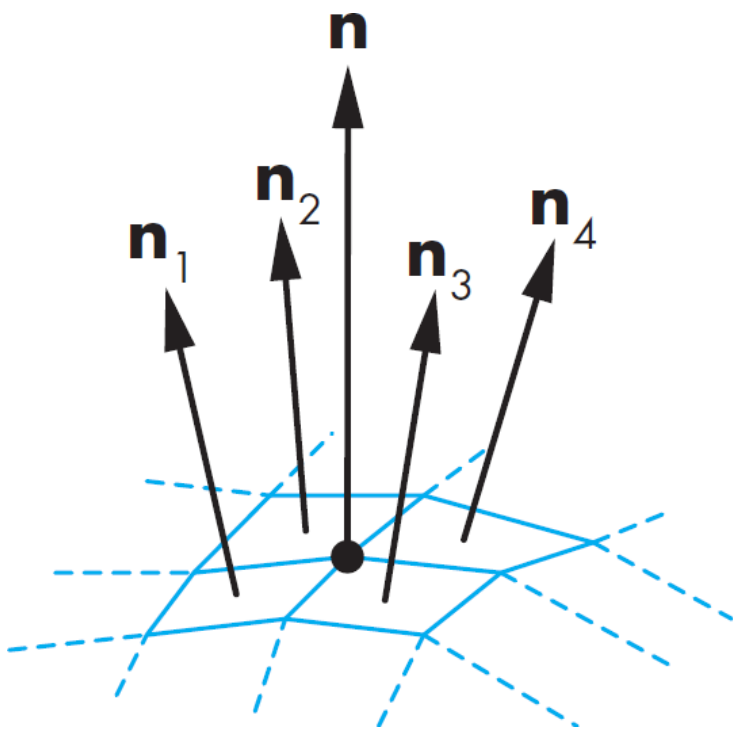
Dois tipos principais: Gouraud e Phong.



## Sombreamento de Gouraud

Vetores normais são definidos para cada vértice.

$$\mathbf{n} = \frac{\sum_i \mathbf{n}_i}{\|\sum_i \mathbf{n}_i\|}$$




Computacionalmente mais simples.

Perde alguns efeitos especulares.

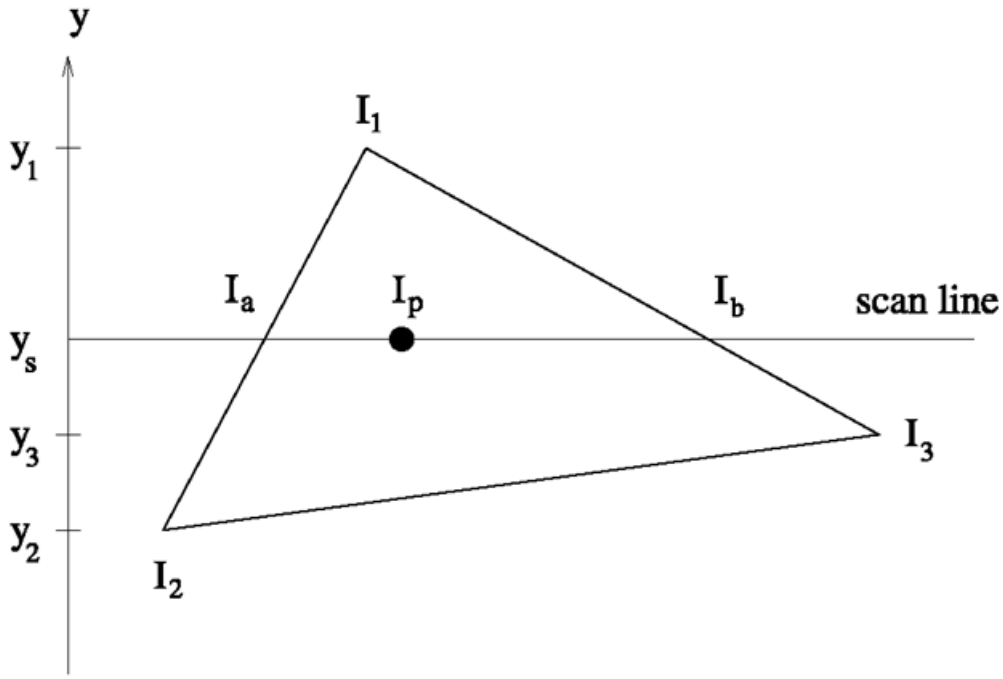
## Sombreamento de Gouraud

A intensidade da cor em um ponto é calculada por interpolação linear:

$$\mathbf{I}_a = \frac{\mathbf{I}_1(y_s - y_2) + \mathbf{I}_2(y_1 - y_s)}{y_1 - y_2}$$

$$\mathbf{I}_b = \frac{\mathbf{I}_1(y_s - y_3) + \mathbf{I}_3(y_1 - y_s)}{y_1 - y_3}$$

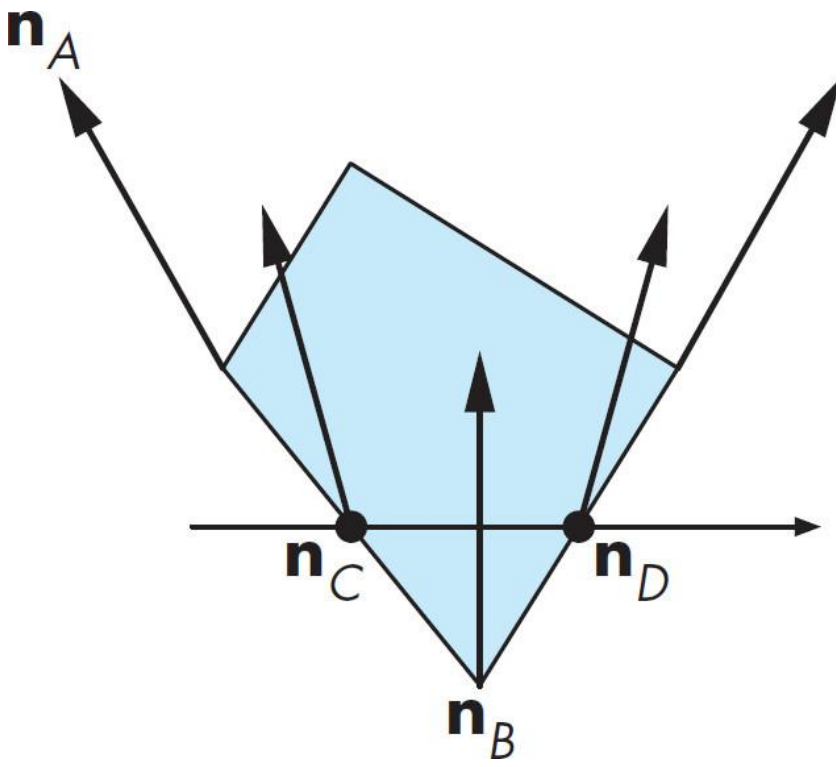
$$\mathbf{I}_p = \frac{\mathbf{I}_a(x_b - x_p) + \mathbf{I}_b(x_p - x_a)}{x_b - x_a}$$



## Sombreamento de Phong

Vetores normais são definidos para cada pixel por interpolação.

$$\mathbf{n}_C(\alpha) = \frac{(1-\alpha)\mathbf{n}_A + \alpha\mathbf{n}_B}{\|(1-\alpha)\mathbf{n}_A + \alpha\mathbf{n}_B\|}$$




As cores são calculadas com base nos vetores normais.

Computacionalmente mais caro.

Resultados mais suaves.

# Transformação do Vetor Normal

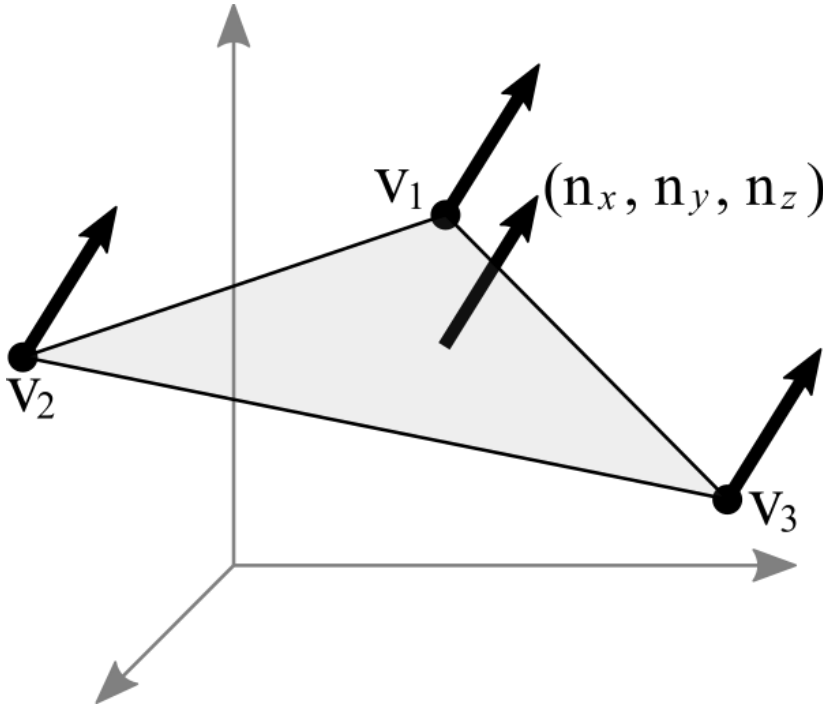
Considerar um vértice (0, 0, 0) e um vetor normal (1, 0, 0). Ao mover o vértice duas unidades ao longo do eixo *y* (0, 2, 0), o vetor normal deve permanecer o mesmo e não (1, 2, 0).

$$\mathbf{n} \cdot \mathbf{v} = 0$$

$$(n_x, n_y, n_z, n_w) \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = 0$$

$$(n_x, n_y, n_z, n_w) M^{-1} M \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = 0$$

Transformação da normal | Transformação do vértice



## Transformação do Vetor Normal

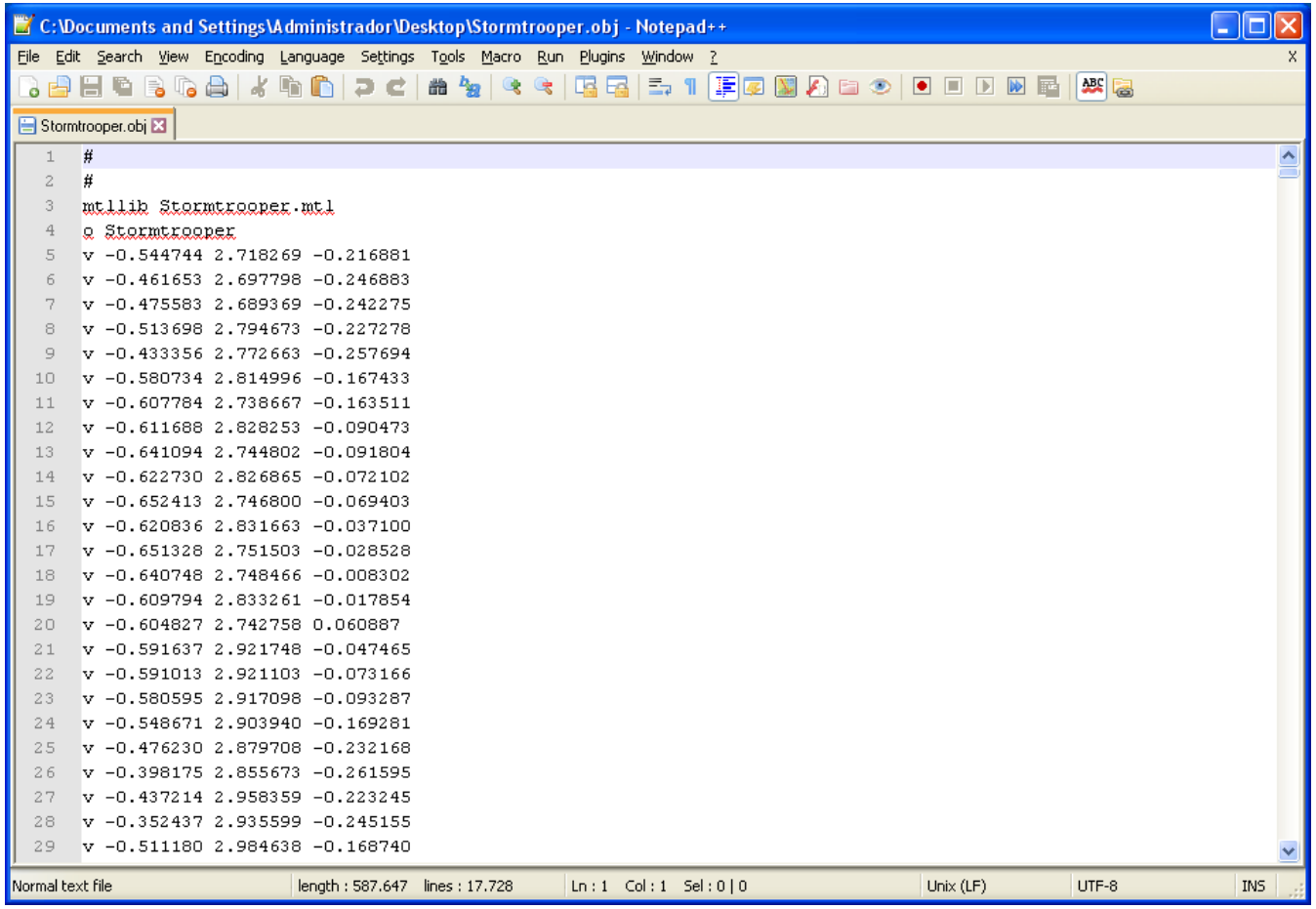
A transformação do vetor normal do espaço do objeto, para o espaço da câmera é:

$$\begin{pmatrix} nx_e \\ ny_e \\ nz_e \\ nw_e \end{pmatrix} = (nx_o, ny_o, nz_o, nw_o) M^{-1}$$

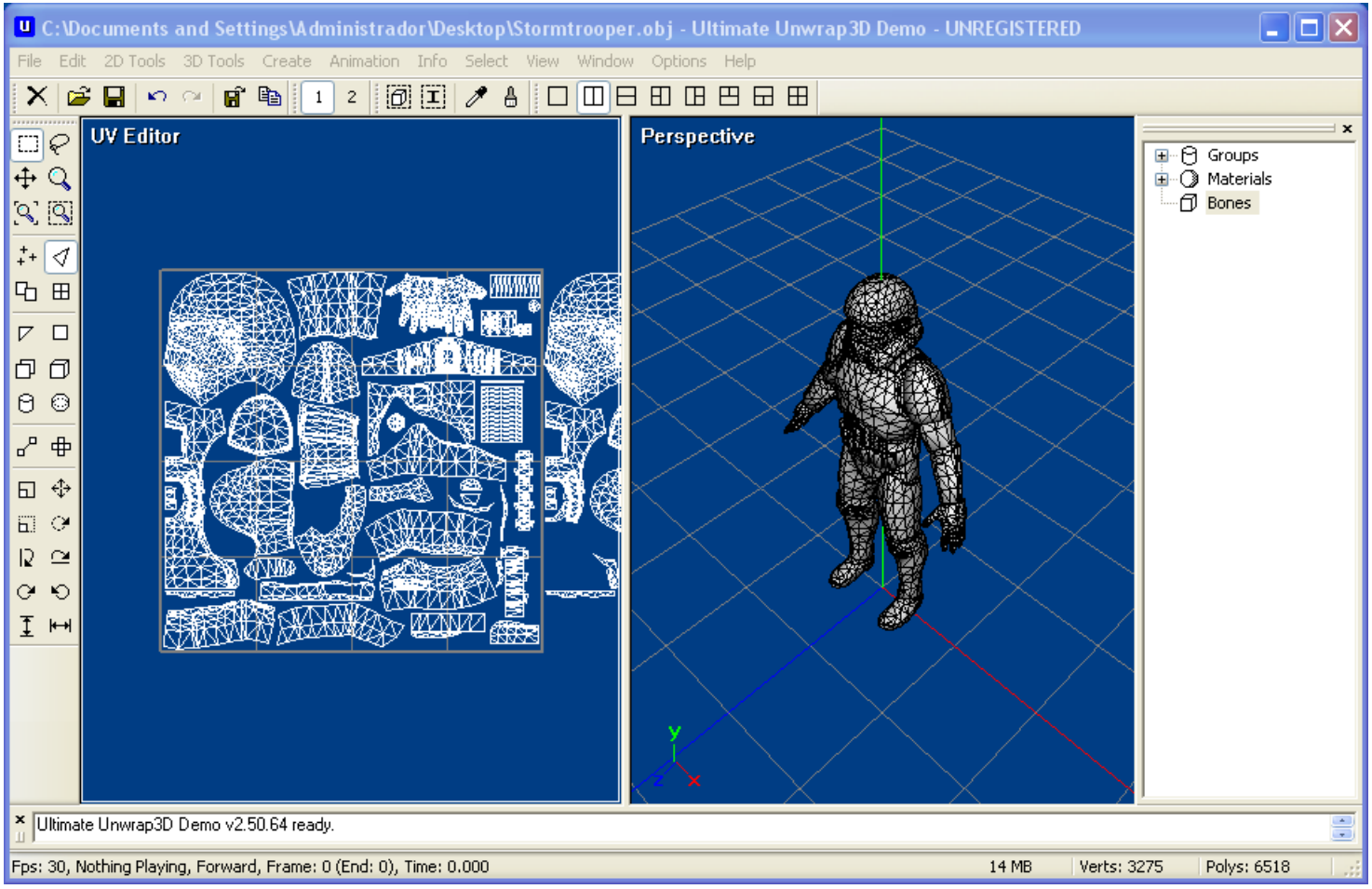
Ou:

$$\begin{pmatrix} nx_e \\ ny_e \\ nz_e \\ nw_e \end{pmatrix} = (M^{-1})^T \begin{pmatrix} nx_o \\ ny_o \\ nz_o \\ nw_o \end{pmatrix}$$

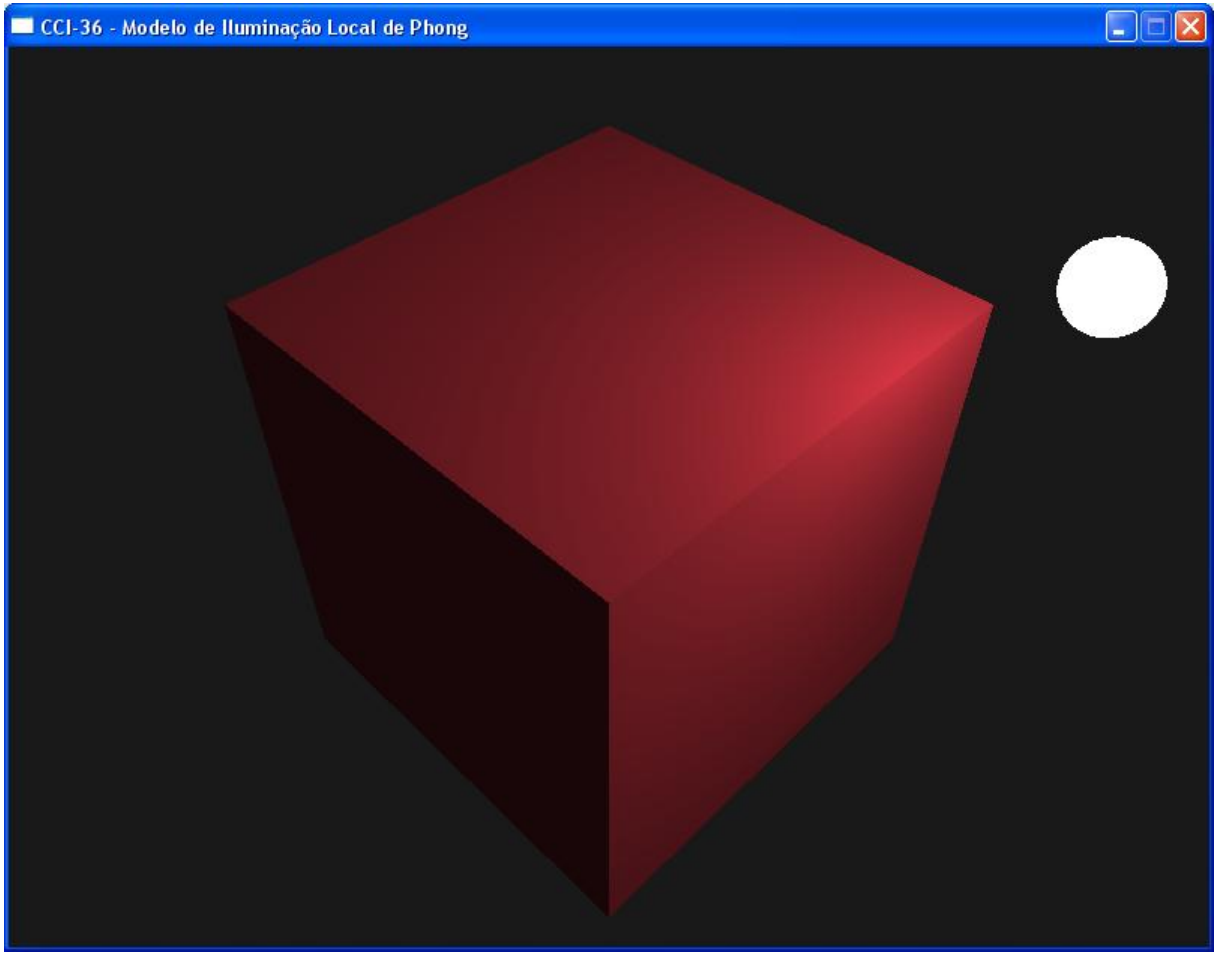
# Carregamento de Objetos – Arquivo OBJ



# Carregamento de Objetos – *Ultimate Unwrap3D Demo*



# Exemplo – Modelo de Iluminação Local de Phong



# Exemplo – Carregamento de Objetos