Introdução à Computação Gráfica

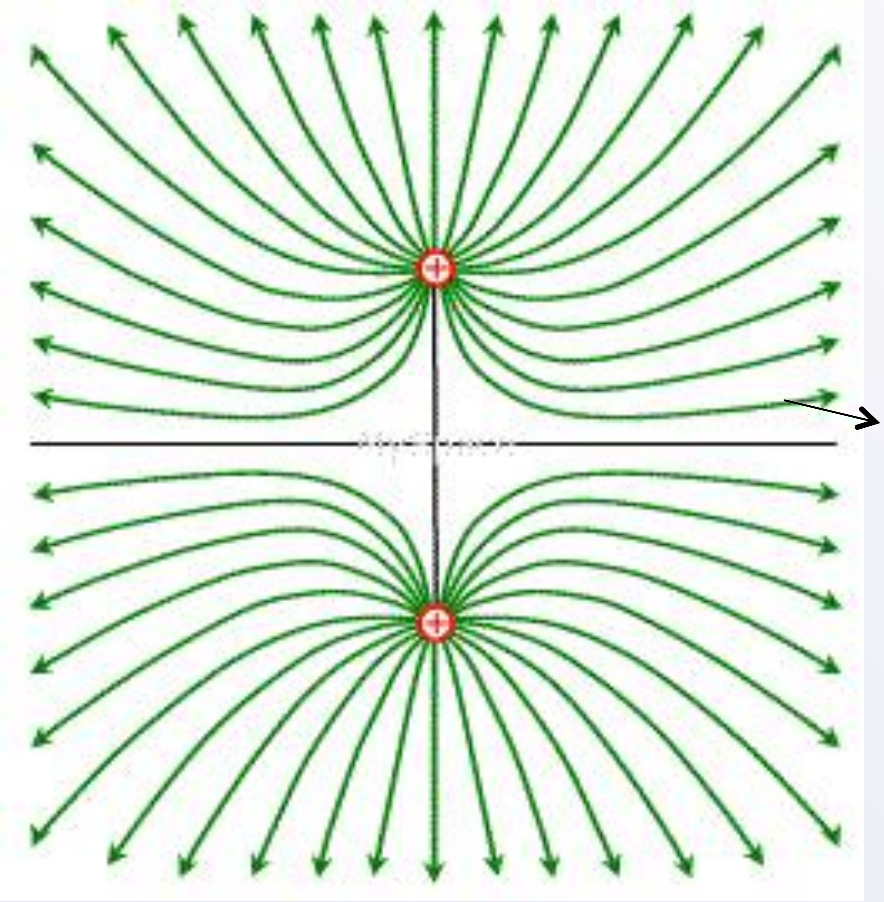
Marcel P. Jackowski
mjack@ime.usp.br

Aula #7: Mais sobre projeções

Objetivos

- EP #1
- Aplicação da projeção perspectiva
  - Construção de sombras (artificiais)
- Normalização de projeções
  - Deformar o volume de visualização
- Modelos articulados
  - Transformações compostas
- Introdução à tonalização

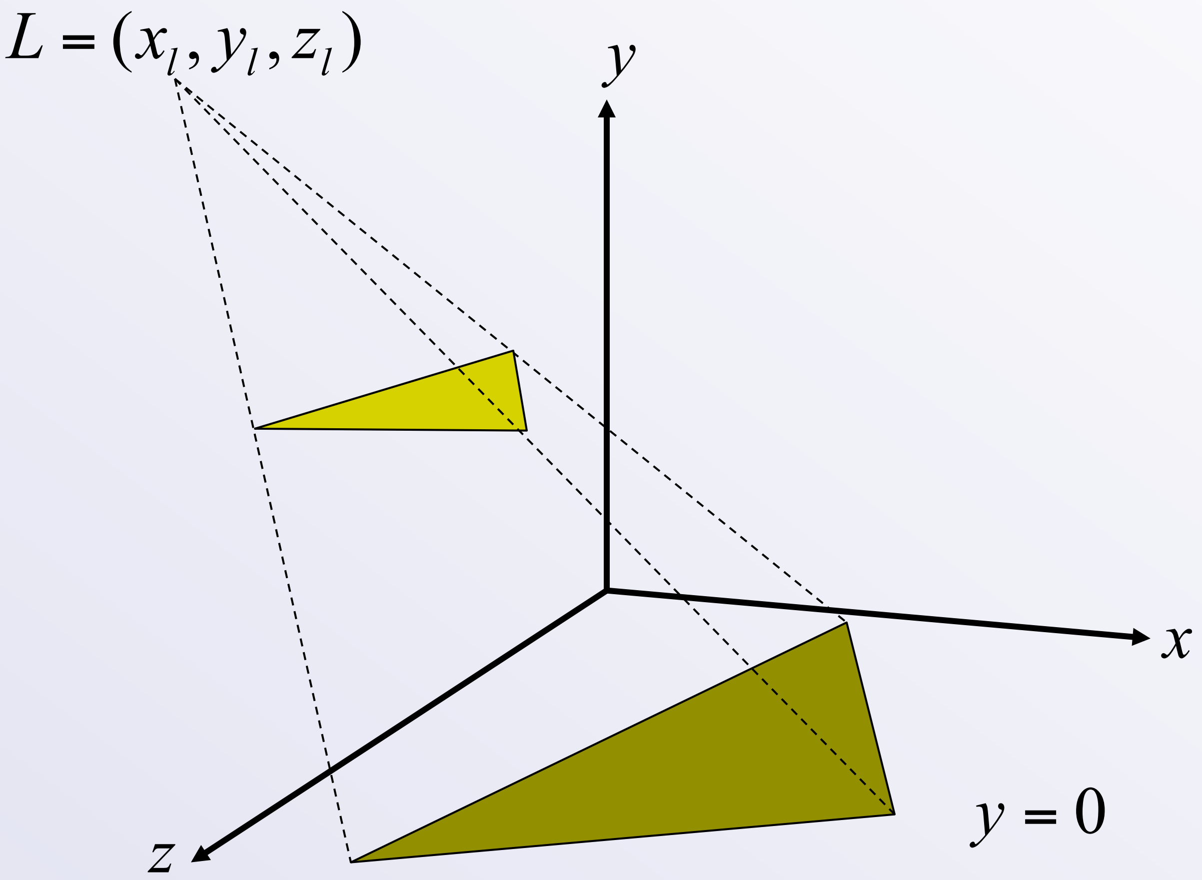
EP #1: Campo de Forças



Linhas de força

Simulação de movimentação de partículas em um campo de forças

Sombras artificiais

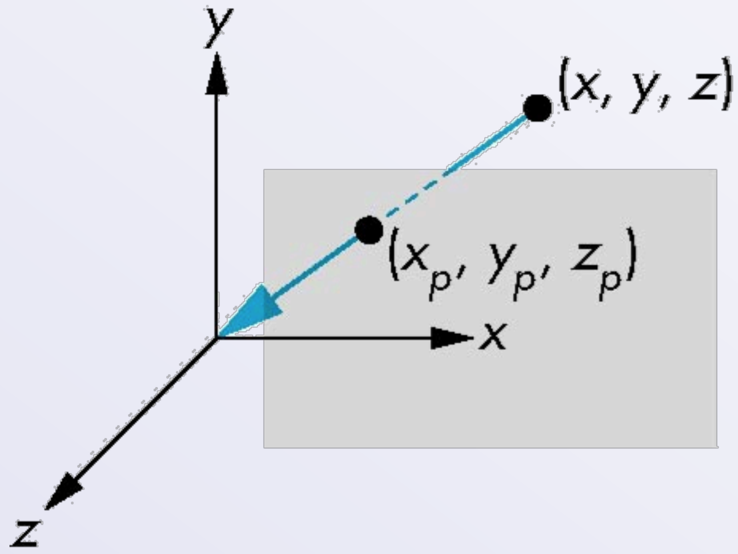


Polígono de sombra

- Escolher o plano de projeção (e.g. $y=0$)
- Escolher ponto de luz $L=(x_l,y_l,z_l)$
- Transformações:
  - Transladar a luz para a origem
  - Projeção perspectiva
  - Transladar a luz de volta ao seu lugar
- Desenhar o objeto duas vezes:
  - Primeira vez para desenhar o polígono
  - Segunda vez para desenhar a sua sombra

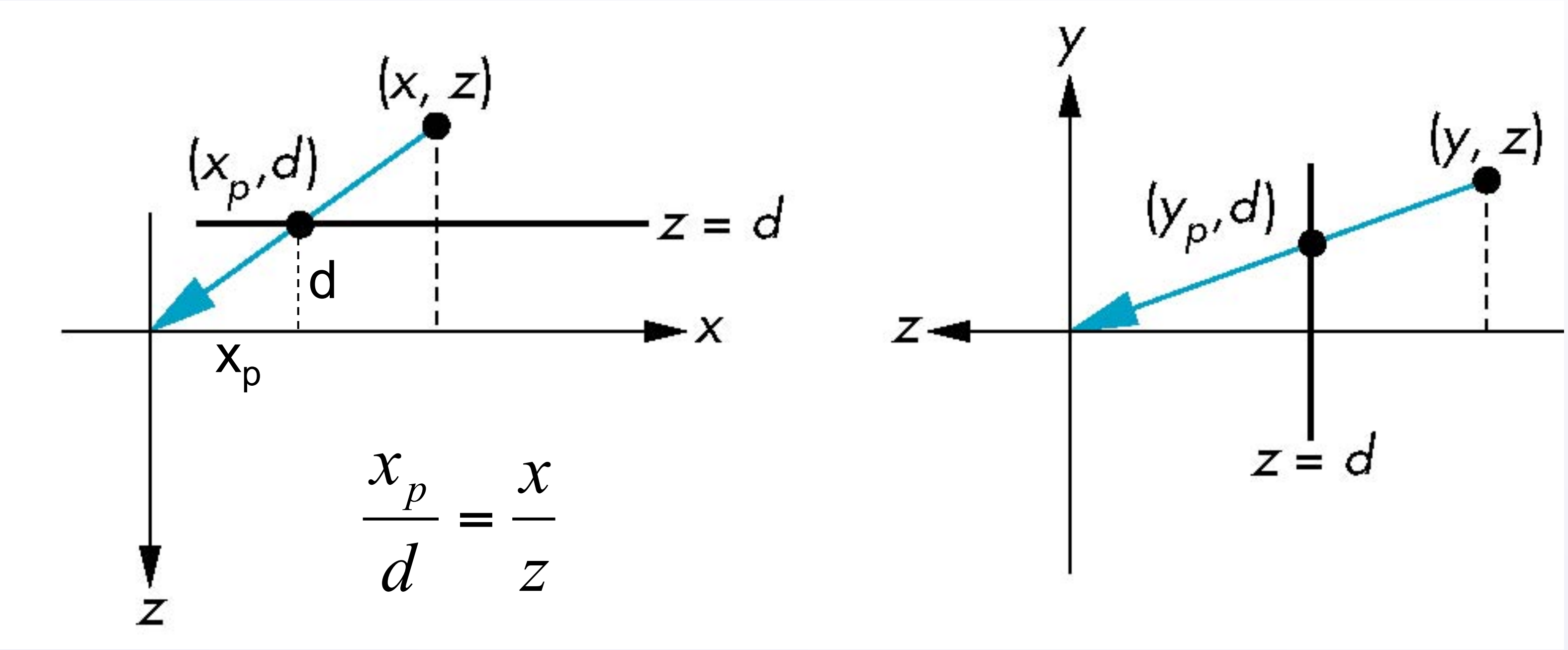
Projeção perspectiva

- Centro da projeção na origem
- Plano de projeção $z = d, d < 0$



Projeção perspectiva

Visões superior e lateral



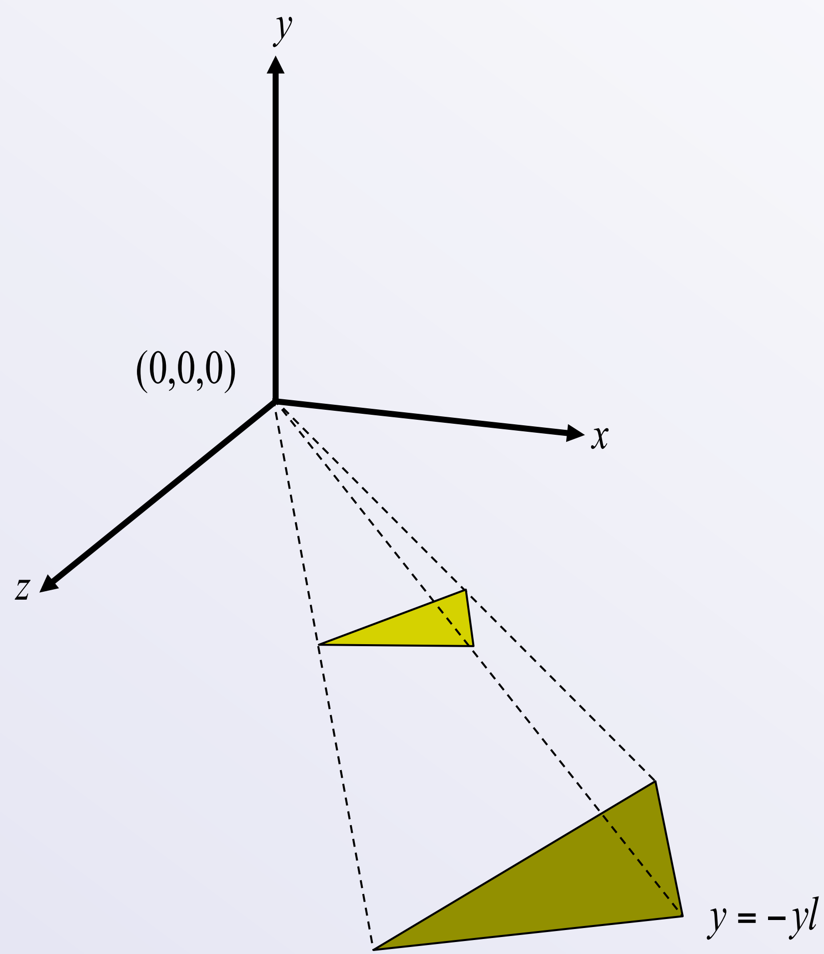
$$x_p = \frac{x}{z/d} \qquad y_p = \frac{y}{z/d} \qquad z_p = d$$

Perspectiva simples

considere $\mathbf{q} = \mathbf{Mp}$ onde $M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix}$

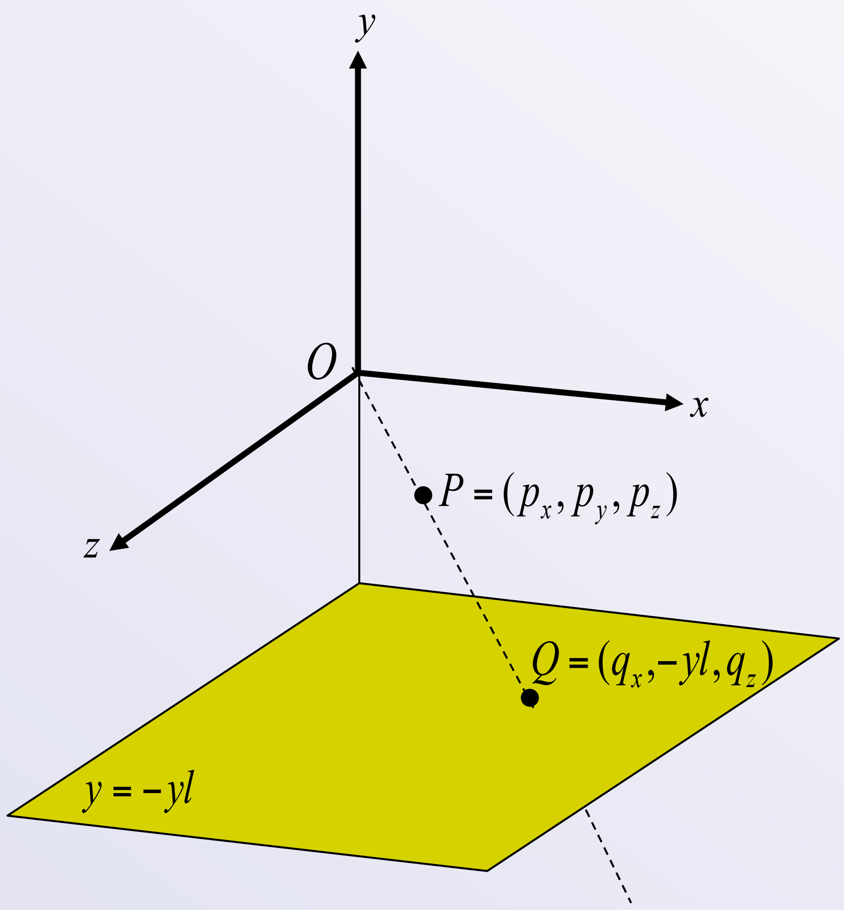
$$q = \begin{bmatrix} x \\ y \\ z \\ z/d \end{bmatrix} = M \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \xrightarrow[\text{Divisão de perspectiva}]{} q' = \begin{bmatrix} x/z/d \\ y/z/d \\ d \\ 1 \end{bmatrix}$$

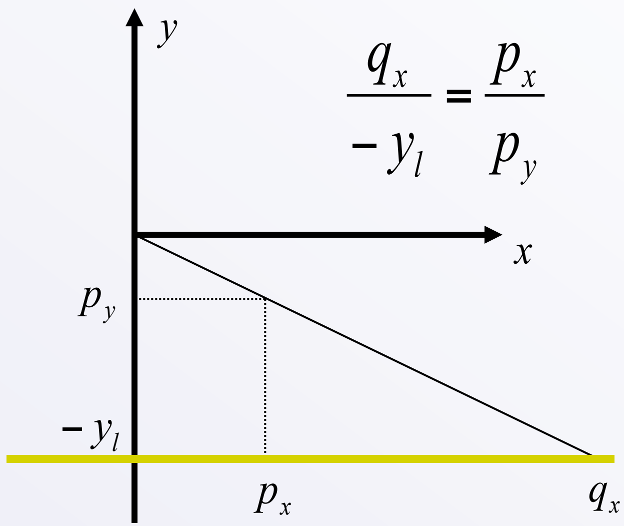
Transladando para a origem



$$T(-x_l, -y_l, -z_l)$$

Transformação de perspectiva





$$\frac{q_x}{-y_l} = \frac{p_x}{p_y}$$

$$q_x = \frac{p_x}{p_y / -y_l} \qquad q_z = \frac{p_z}{p_y / -y_l}$$

Em coordenadas homogêneas

considere $\mathbf{q} = \mathbf{Mp}$ onde $M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1/y_l & 0 & 0 \end{bmatrix}$

$$q = \begin{bmatrix} q_x \\ q_y \\ q_z \\ -p_y / y_l \end{bmatrix} = M \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \xrightarrow[\text{Divisão de perspectiva}]{} q' = \begin{bmatrix} -p_x / p_y / y_l \\ -y_l \\ -p_z / p_y / y_l \\ 1 \end{bmatrix}$$

Transformação da sombra

- A transformação total será:

$$q = T(x_l, y_l, z_l) \cdot M \cdot T(-x_l, -y_l, -z_l) \cdot p$$

- Após a divisão de perspectiva e translação final:

$$q_x = x_l - \frac{p_x - x_l}{(p_y - y_l)/y_l}$$

$$q_y = 0$$

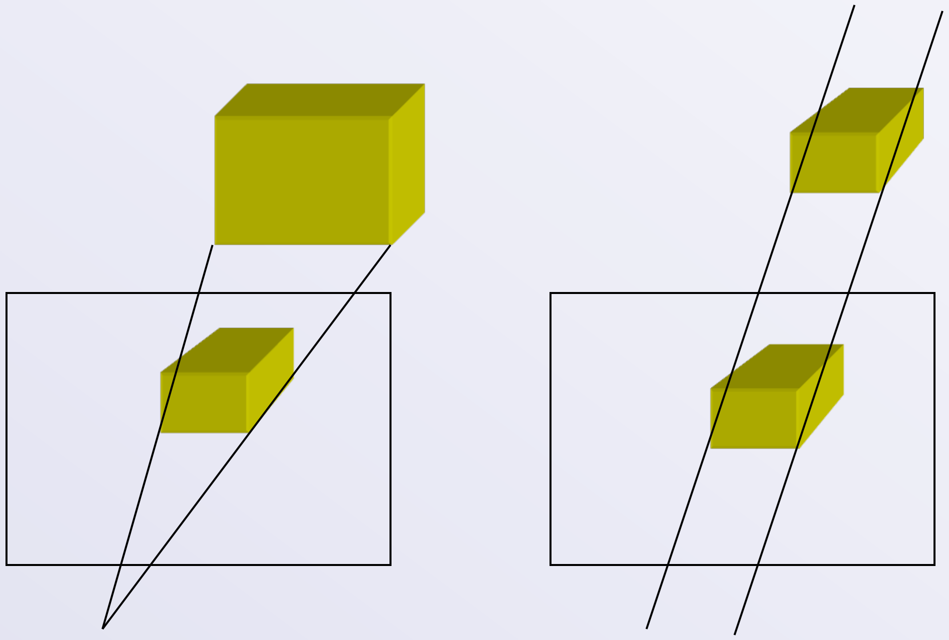
$$q_z = z_l - \frac{p_z - z_l}{(p_y - y_l)/y_l}$$

Demo shadow.c

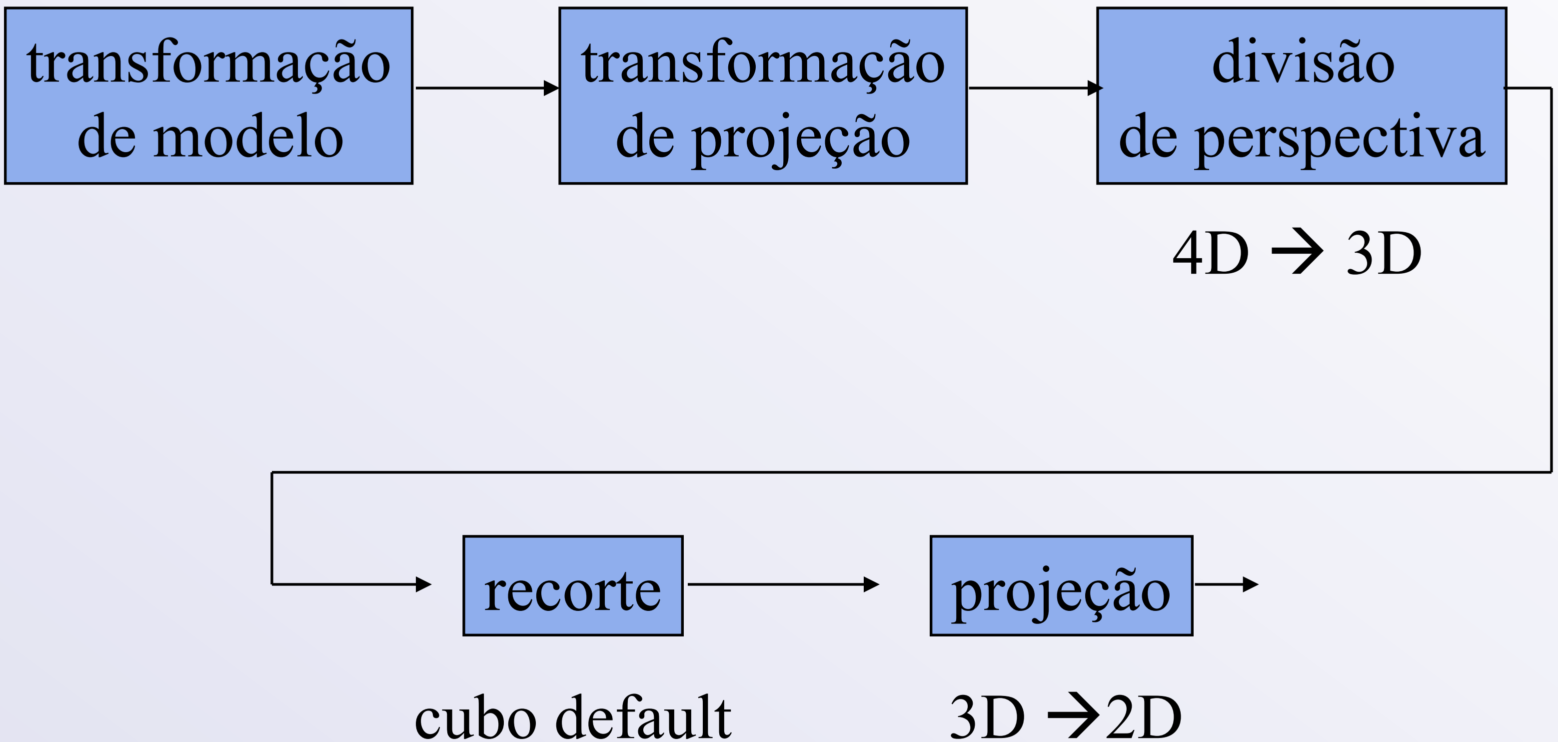
- Exemplifica o uso da transformação perspectiva como transformação de modelo.
- Sugestão: aprenda a usar o buffer de acumulação para deixar a sombra suave.

Normalização de projeção

- Primeiro, distorça a cena através de uma transformação afim.
- A projeção ortográfica da cena distorcida é a mesma que a projeção original.



Pipeline de visualização



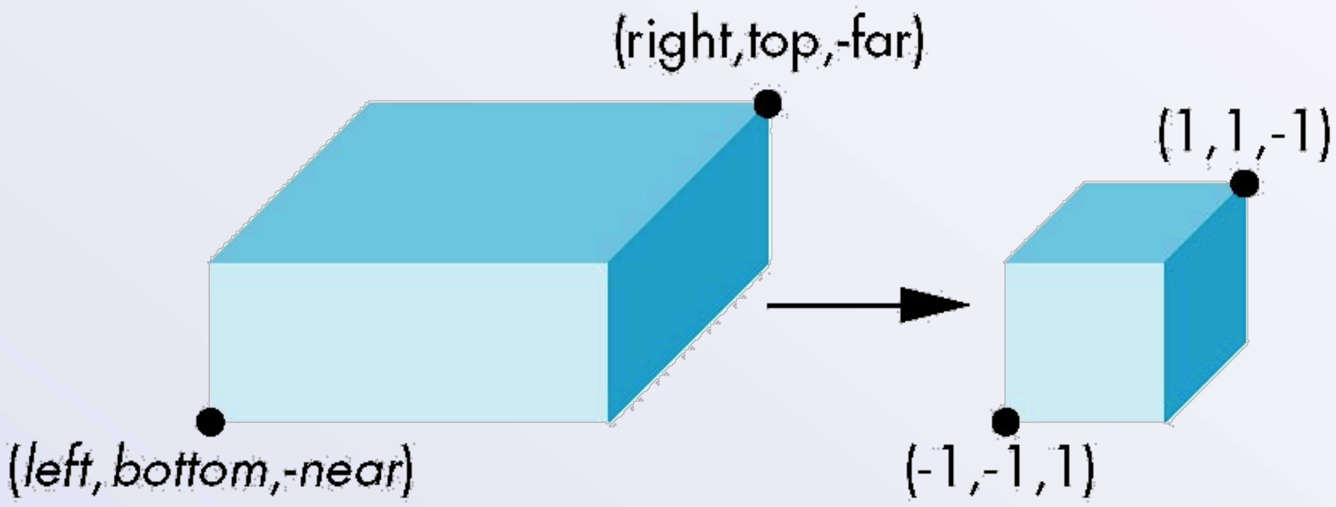
Características

- Permanecemos em coordenadas homogêneas (4D) durante as transformações de modelo e de projeção
  - Matriz identidade (visão ortogonal)
- A normalização faz que com o recorte seja em relação à um simples cubo independente do tipo de projeção
- Retardar a projeção ao máximo possível
  - Importante para reter informação sobre profundidade para o mecanismo de remoção de superfícies escondidas

Normalização ortogonal

`glOrtho(left,right,bottom,top,near,far)`

normalização → achar transformação para converter o volume de recorte para o default (volume canônico)



Matriz ortogonal

- Consiste em dois passos
  - Mover o centro para a origem

    T(-(left+right)/2, -(bottom+top)/2,(near+far)/2)**)**
  - Escalar para que os lados tenham comprimento 2

    S(2/(left-right),2/(top-bottom),2/(near-far)**)**

$$\mathbf{P} = \mathbf{ST} = \begin{bmatrix} \frac{2}{right-left} & 0 & 0 & -\frac{right+left}{right-left} \\ 0 & \frac{2}{top-bottom} & 0 & -\frac{top+bottom}{top-bottom} \\ 0 & 0 & \frac{2}{near-far} & \frac{far+near}{far-near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projeção ortogonal final

- Fazemos com que $z = 0$
- Equivalente à transformação de coordenadas homogêneas

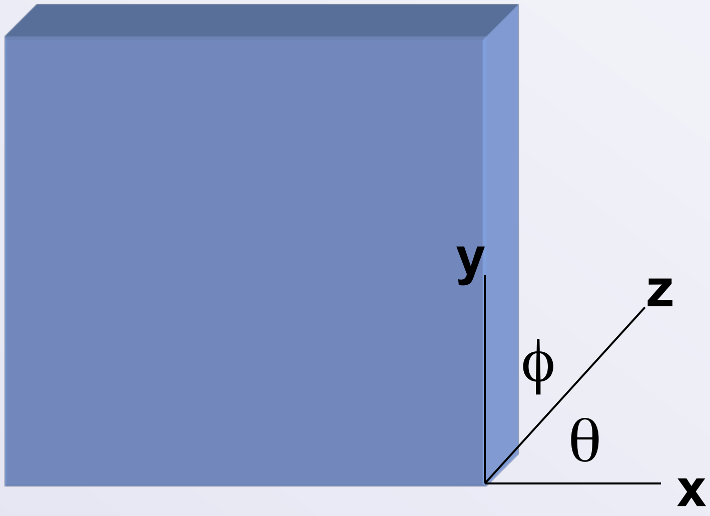
$$\mathbf{M}_{\text{orth}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Assim, uma projeção ortogonal generalizada em 4D é definida por

$$\mathbf{P} = \mathbf{M}_{\text{orth}}\mathbf{ST}$$

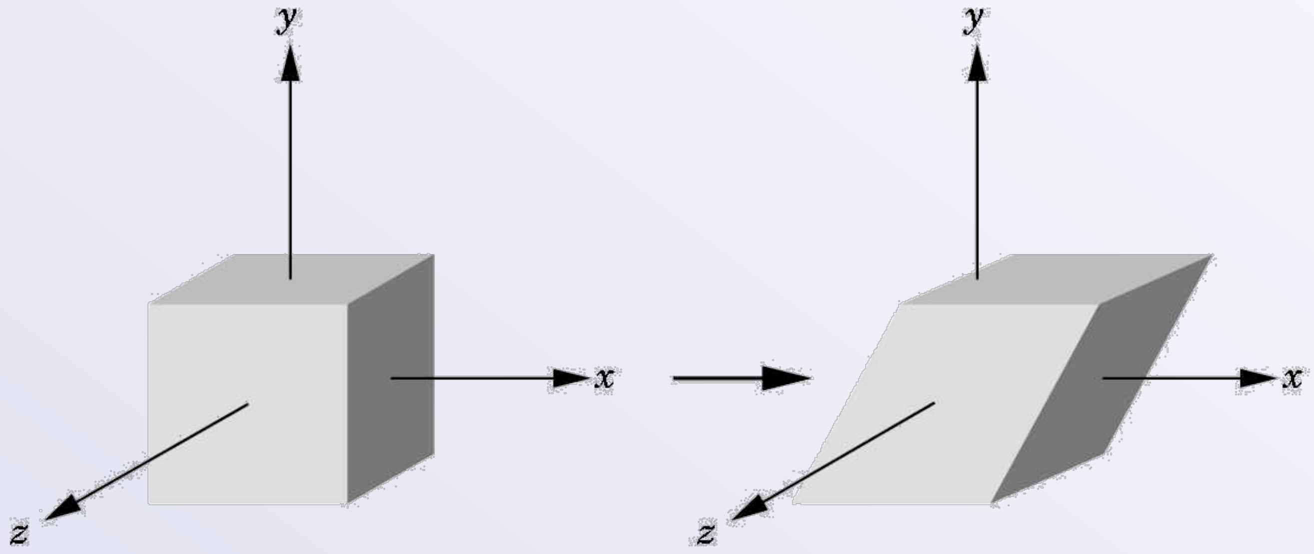
Projeções oblíquas

- As funções de projeção em OpenGL não produzem projeções paralelas como a vista abaixo.



Cisalhamento

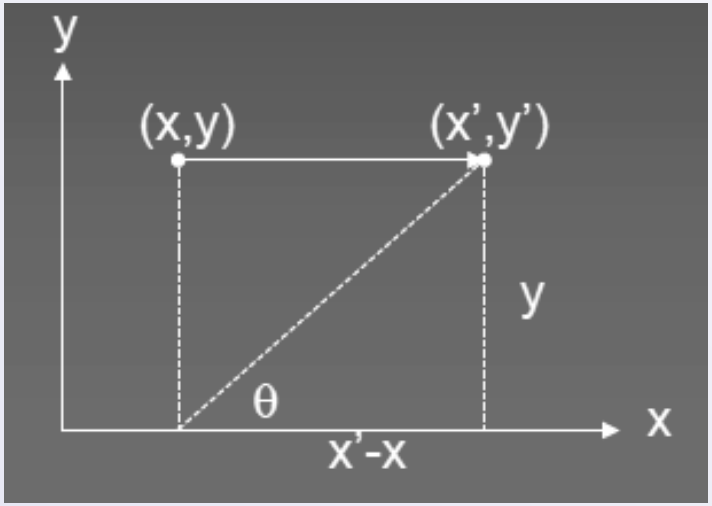
- Cisalhamento em x
  - Escala x proporcional a y.
  - Deixa y e z inalterados.



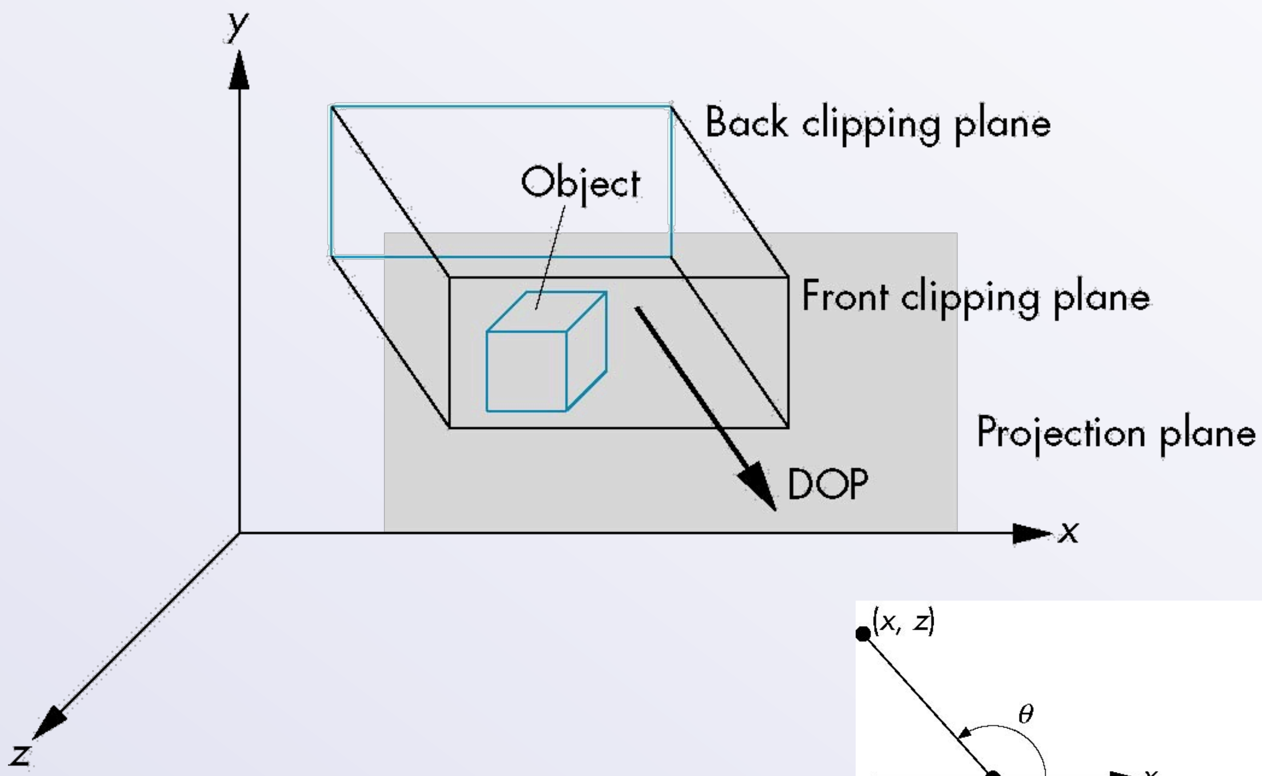
Cisalhamento

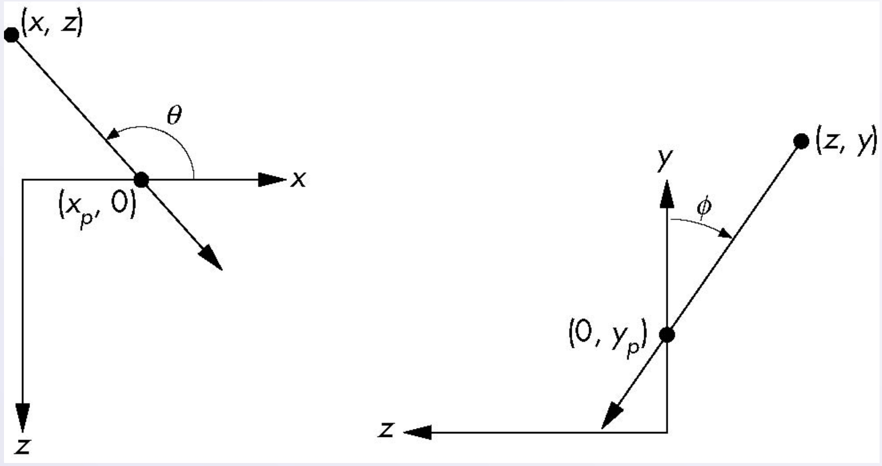
- $x' = x + y \cot(\theta)$
- $y' = y$
- $z' = z$

$$\mathbf{H}_x(\theta) = \begin{bmatrix} 1 & \cot(\theta) & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Cisalhamento genérico





Matriz de cisalhamento

Cisalhamento em xy (valores de z não mudam)

$$\mathbf{H}(\theta,\phi) = \begin{bmatrix} 1 & 0 & -\cot\theta & 0 \\ 0 & 1 & -\cot\varphi & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Matriz de projeção

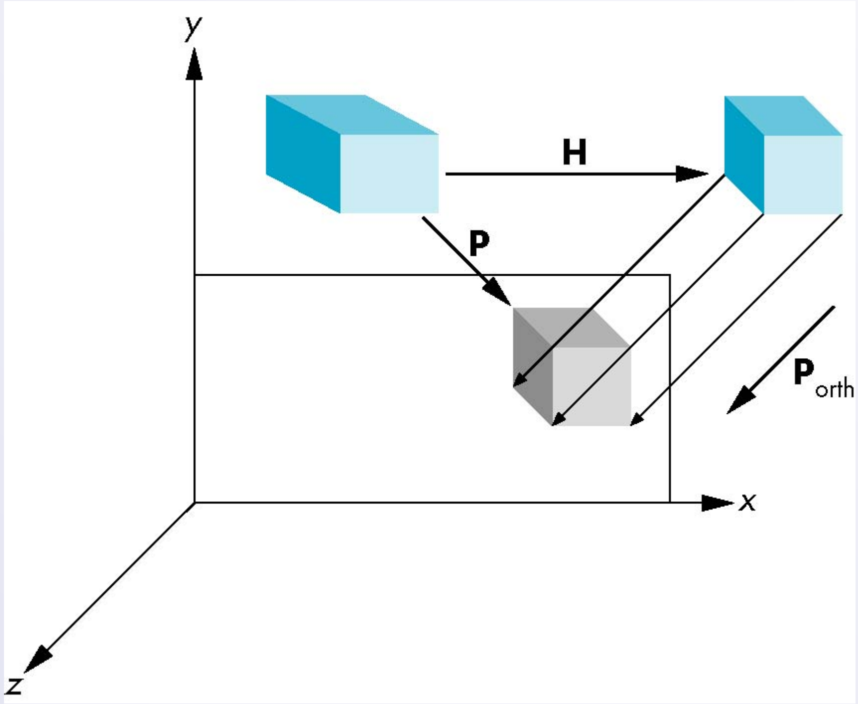
$$\mathbf{P} = \mathbf{M}_{\mathrm{orth}}\,\mathbf{H}(\theta,\phi)$$

Caso geral:

$$\mathbf{P} = \mathbf{M}_{\mathrm{orth}}\,\mathbf{STH}(\theta,\phi)$$

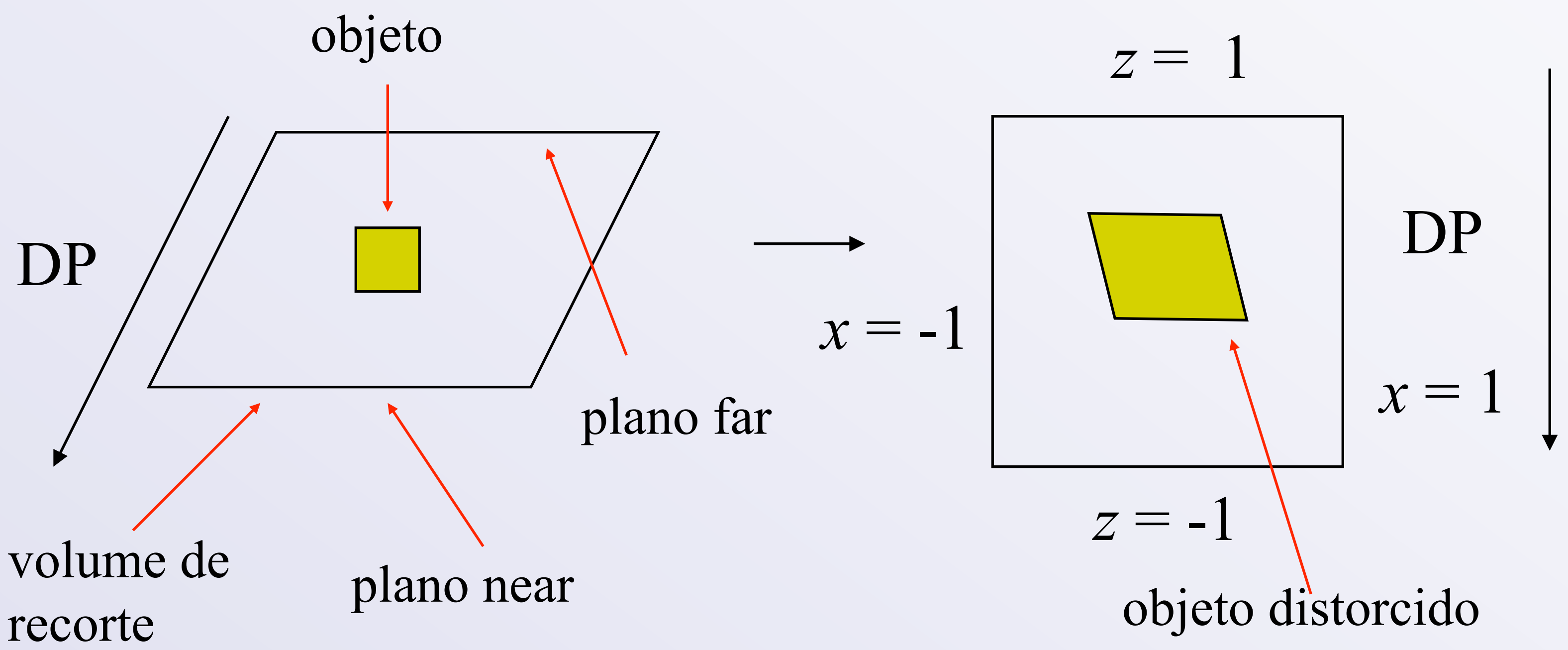
Equivalência

- Projeção original = Cisalhamento + Projeção Ortogonal



Efeito no recorte

- A matriz de projeção $\mathbf{P} = \mathbf{STH}$ transforma o volume de recorte original para o volume default



Instâncias

- Definição geométrica do objeto
  - Símbolo
- Cada aparência deste objeto no modelo é chamada de instância
- Transformação de instância:
  - Escala, rotação e translação do objeto

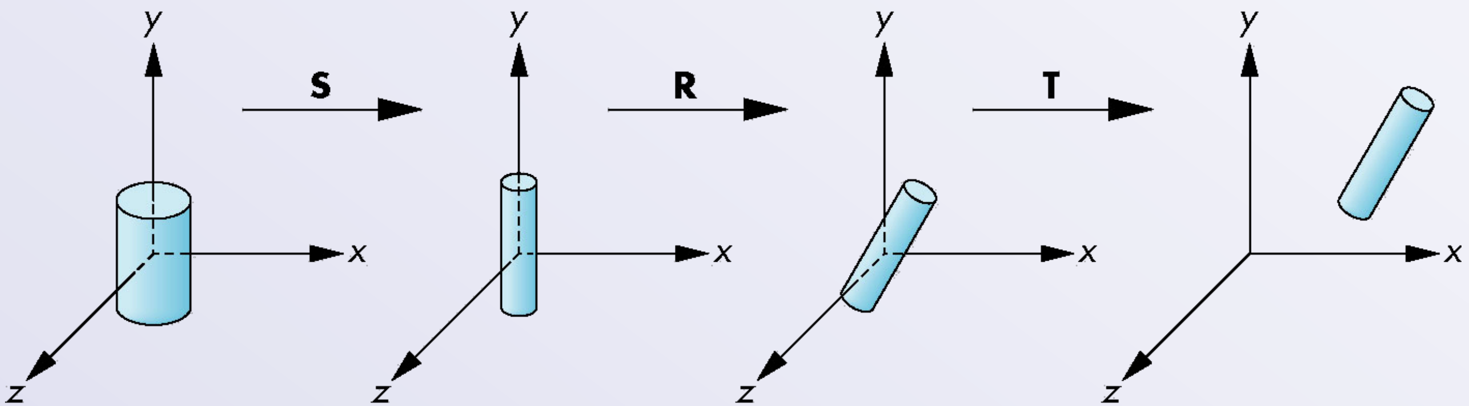
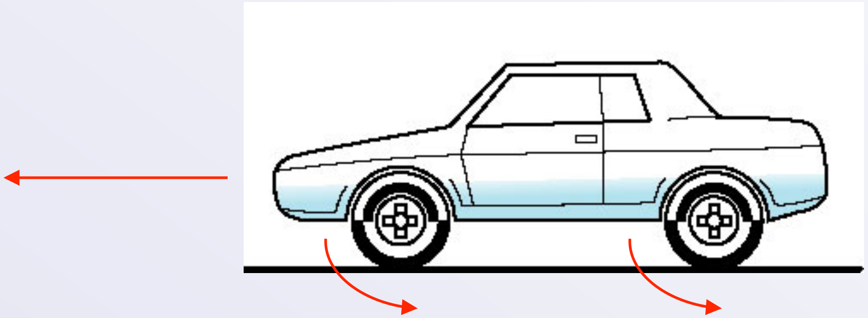


Tabela de símbolos-instâncias

Symbol	Scale	Rotate	Translate
1	s_x, s_y, s_z	$\theta_x, \theta_y, \theta_z$	d_x, d_y, d_z
2			
3			
1			
1			
.			
.			

Exemplo de modelo: carro

- A tabela símbolo-instância não mostra os relacionamentos entre as partes do modelo
- Modelo carro contém dois símbolos:
  - Chassi + 4 rodas



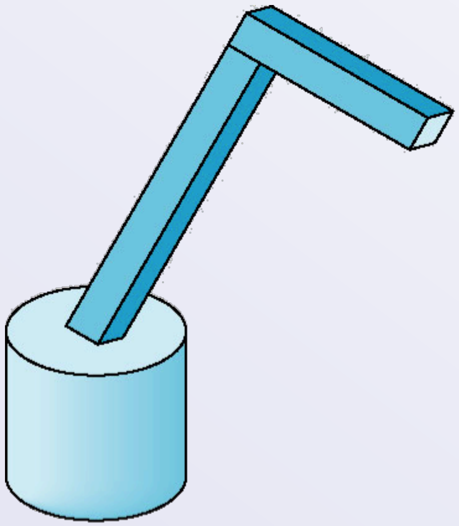
- Movimento é determinado pela velocidade rotacional das rodas

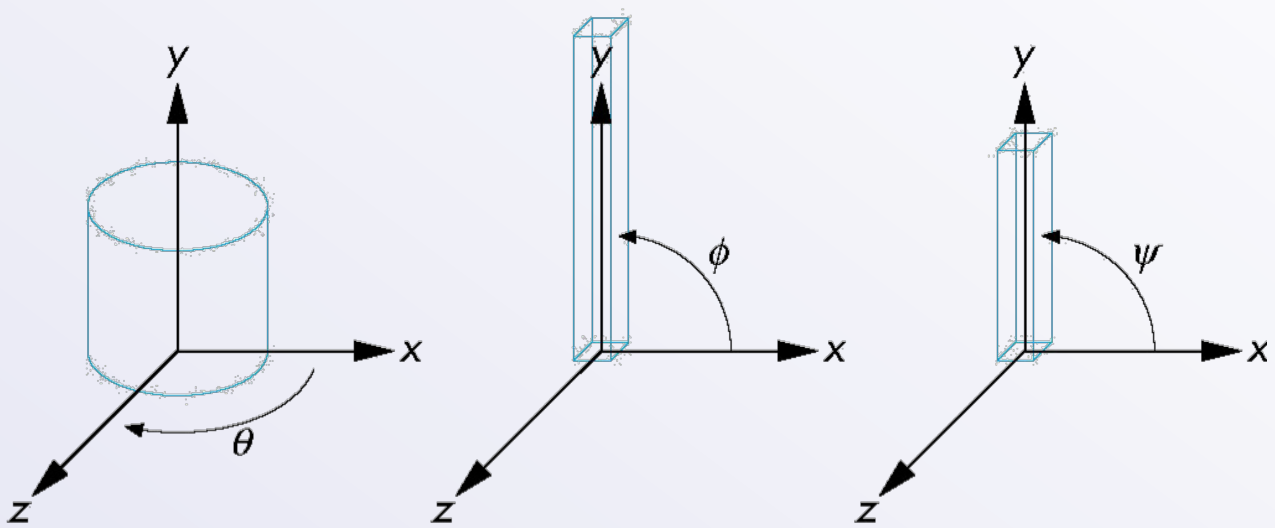
Estrutura de chamadas

```
carro(velocidade)
{
    desenha_chassi()
    desenha_roda(frente_direita);
    desenha_roda(frente_esquerda);
    desenha_roda(traseira_direita);
    desenha_roda(traseira_esquerda);
}
```

- Falha em representar o relacionamento entre as partes!

Robô



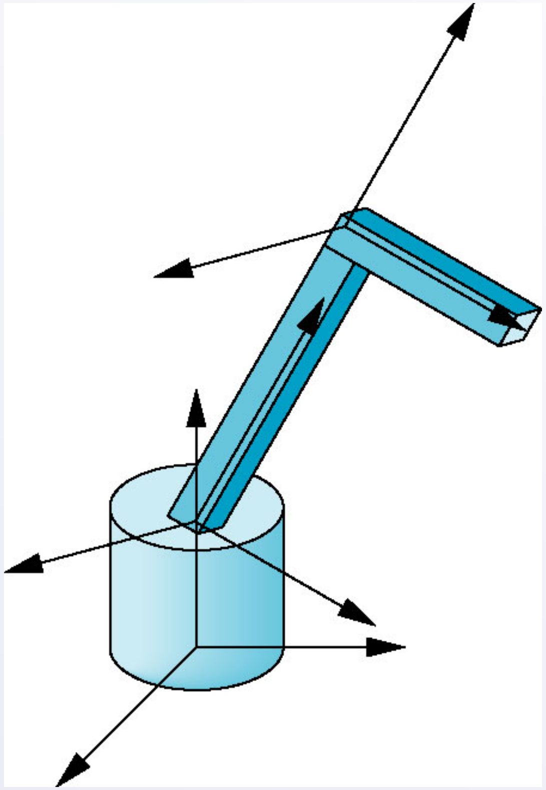


Robô

cada parte em seu sistema de coordenadas próprio

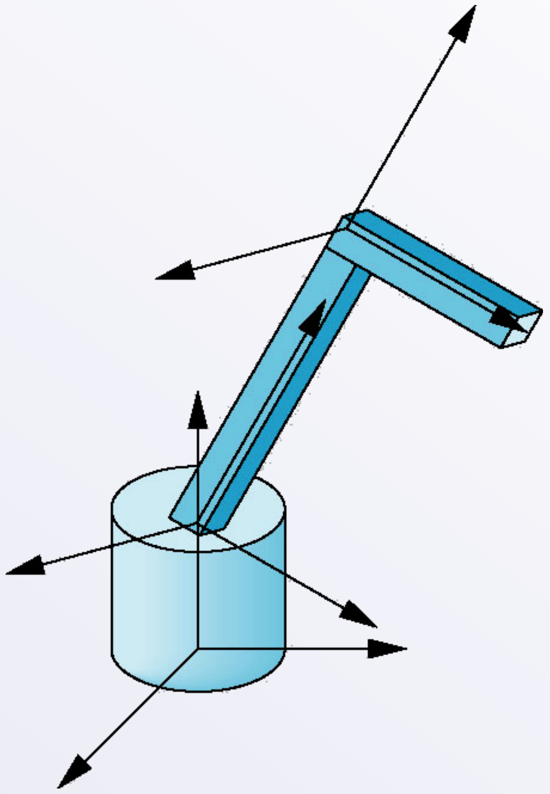
Modelos articulados

- O nosso robô é um exemplo de um *modelo articulado*
  - Partes são conectadas nas junções
  - Podemos especificar o estado do modelo através do ângulo de cada junção



Relacionamentos

- Base rotaciona de forma independente
  - Posição determinada por um único ângulo
- Braço (inferior) é conectado a base
  - Sua posição depende da rotação da base
  - Deve ser transladado em relação a base e rotacionado no ponto de junção
- Antebraço conectado ao braço
  - Sua posição depende da base e do braço
  - Deve ser transladado em relação ao braço e rotacionado no seu ponto de junção



Matrizes de transformação

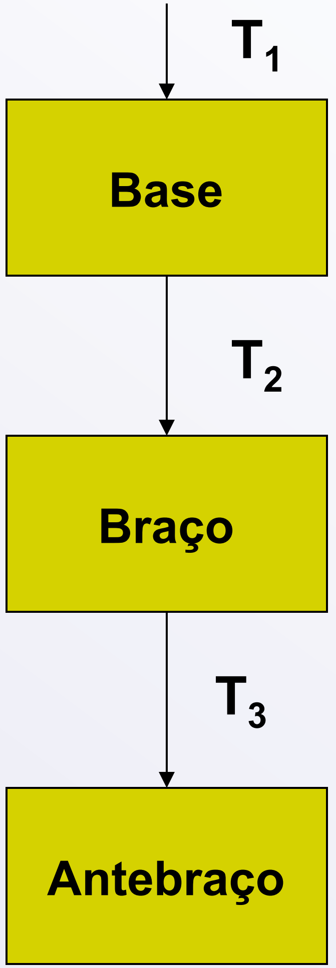
- Rotação da base: $\mathbf{R}_b$
  - Aplicar $\mathbf{M} = \mathbf{R}_b$ na base
- Transladar braço <u>relativo</u> a base: $\mathbf{T}_{lu}$
- Rotacionar braço na junção: $\mathbf{R}_{lu}$
  - Aplicar $\mathbf{M} = \mathbf{R}_b\, \mathbf{T}_{lu}\, \mathbf{R}_{lu}$ para o braço
- Transladar antebraço <u>relativo</u> ao braço: $\mathbf{T}_{uu}$
- Rotacionar antebraço: $\mathbf{R}_{uu}$
  - Aplicar $\mathbf{M} = \mathbf{R}_b\, \mathbf{T}_{lu}\, \mathbf{R}_{lu}\, \mathbf{T}_{uu}\, \mathbf{R}_{uu}$ para o antebraço

Código em OpenGL

```
desenha_robo()
{
    glRotate(theta, 0.0, 1.0, 0.0);
    desenha_base();
    glTranslate(0.0, h1, 0.0);
    glRotate(phi, 0.0, 1.0, 0.0);
    desenha_braço();
    glTranslate(0.0, h2, 0.0);
    glRotate(psi, 0.0, 1.0, 0.0);
    desenha_antebraço ();
}
```

Árvore de relacionamento

- Note que o código mostra a dependência entre as partes do modelo
  - Podemos facilmente mudar a "aparência" das partes sem alterar o relacionamento entre elas
- Simples estrutura de árvore

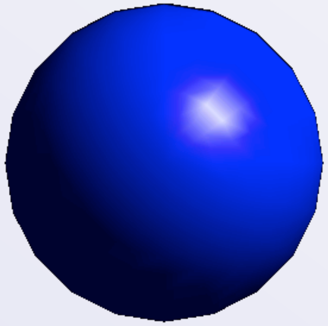


Demo robot.c

Tonalização

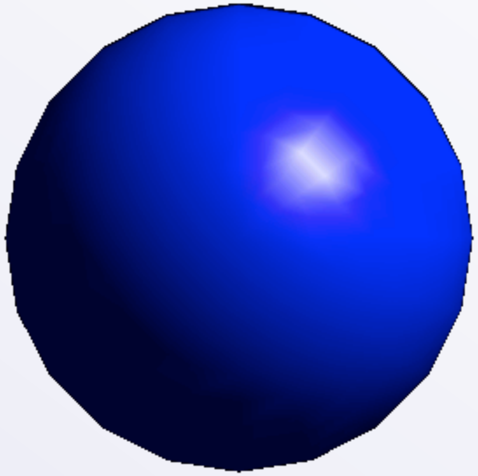
- Na subdivisão da esfera, percebemos que se usarmos somente **`glColor`**, teríamos como resultado algo como:

- Embora gostaríamos de ter visto:



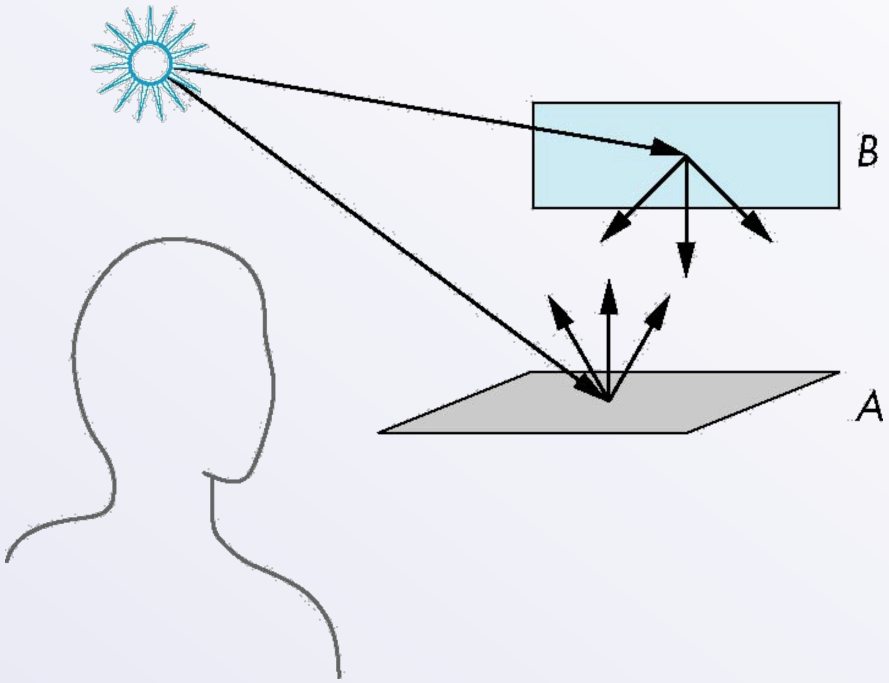
Tonalização

- 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”)

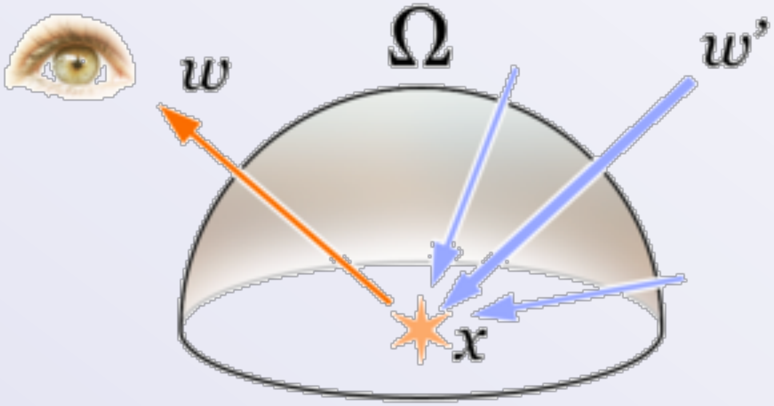
- Luz atinge superfície A
  - Parte é refletida
  - Parte é absorvida
- Parte da luz refletida atinge superfície B
  - Parte é refletida
  - Parte é absorvida
- E assim por diante…



Equação de renderização

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

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



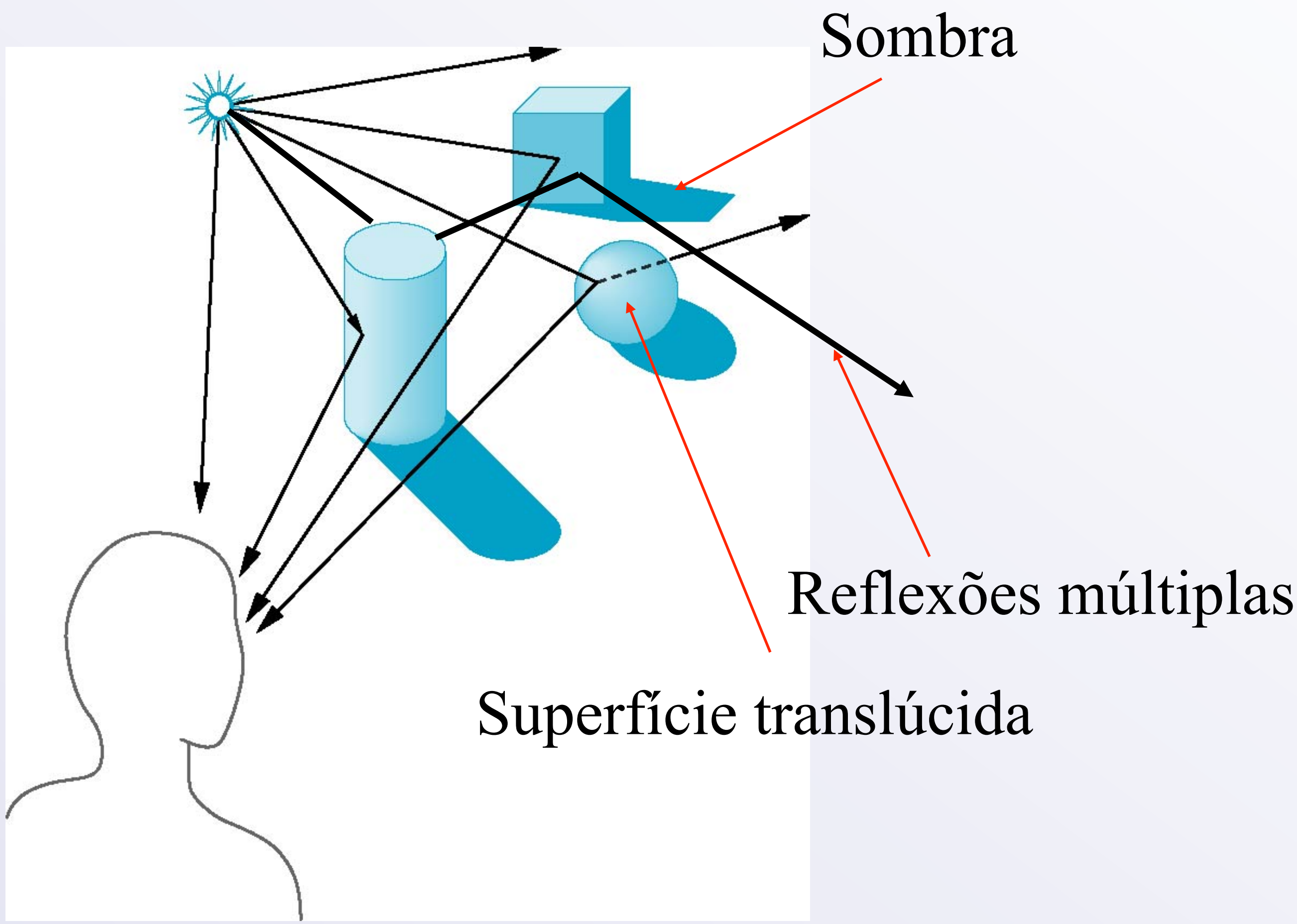
Kajiya, James T., The rendering equation", *SIGGRAPH 1986*

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
  - Espalhamento múltiplo entre objetos

Exemplo

Efeitos globais



Renderização global e local

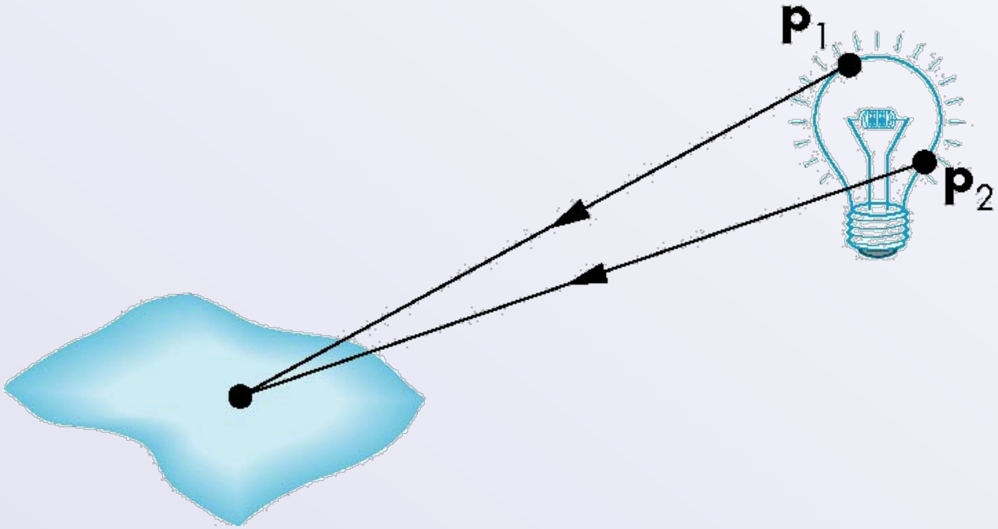
- A tonalização correta requer um cálculo global que envolve todos os objetos e fontes de luz
  - Incompatível com o modelo em pipeline que sombreia cada polígono independentemente (renderização local)
- Todavia, em CG, e especialmente em aplicações em tempo real, nos contentamos que a nossa cena “pareça”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

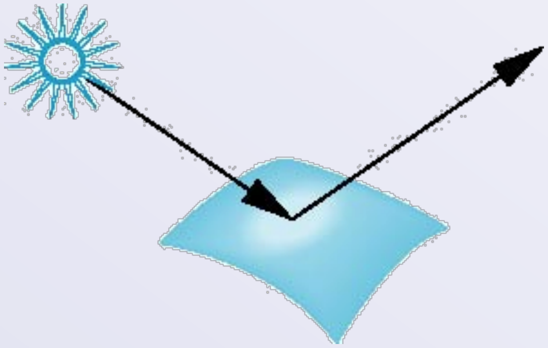


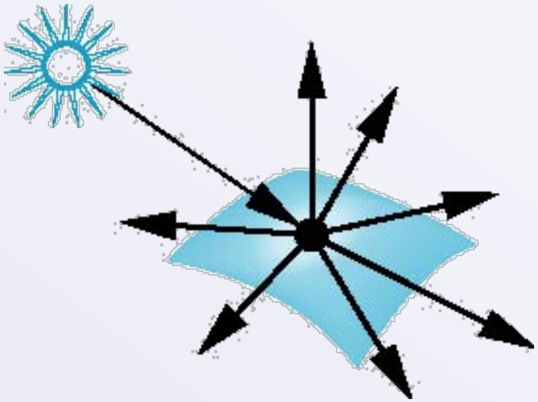
Fontes de luz

- Ponto de luz (ideal)
  - Modelo com posição e cor
  - Fonte 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

Tipos de superfície

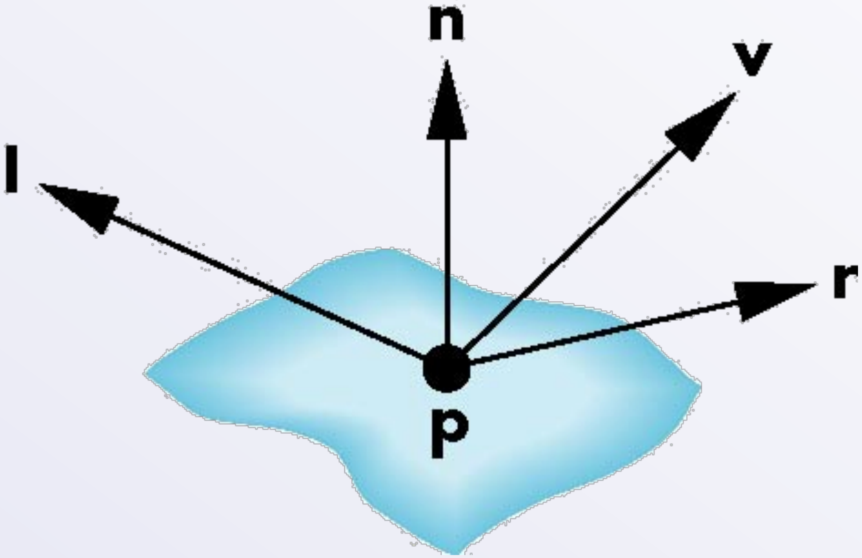
- Quanto mais suave for a superfície, mais concentrada será a luz refletida nela
- Uma superfície irregular espalhará os raios em diversas direções.





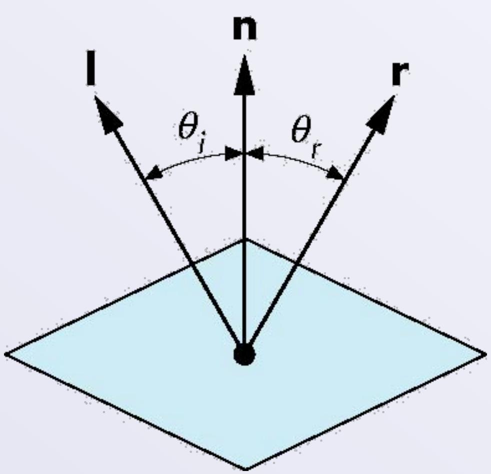
Modelo Phong

- 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

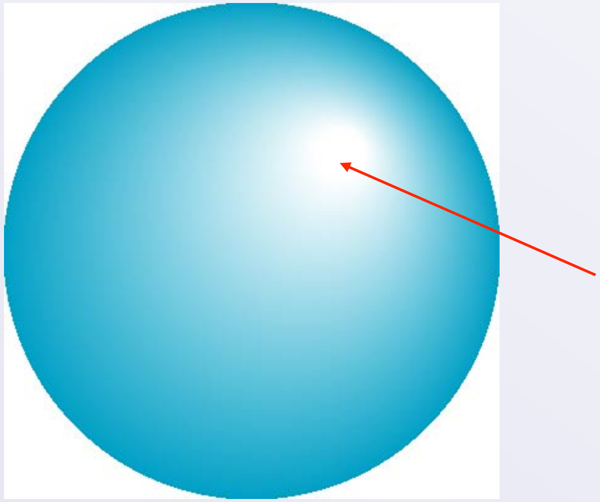


Superfície Lambertiana

- Superfícies difusoras perfeitas
- Reflete 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.

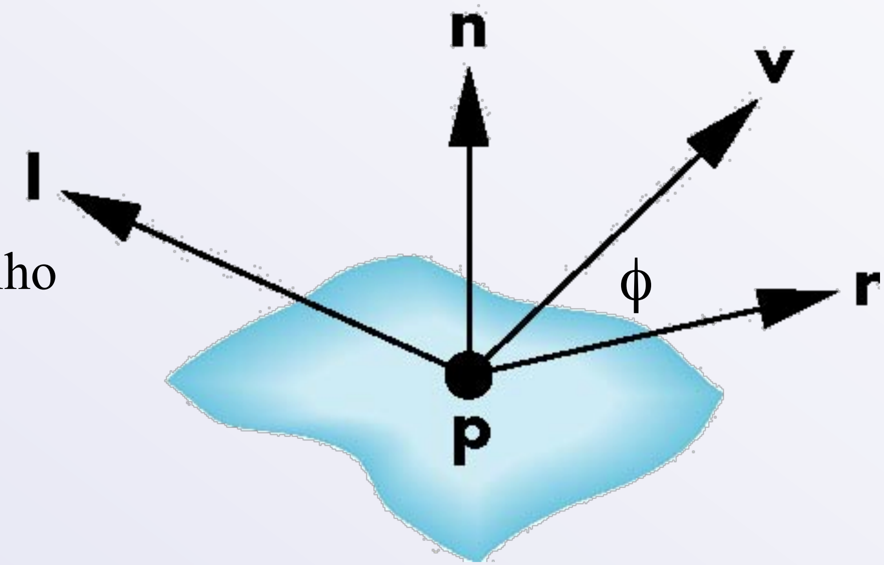
$$\mathbf{I}_s \sim k_s\, \mathbf{I} \cos^{\alpha}\phi$$

Intensidade refletida

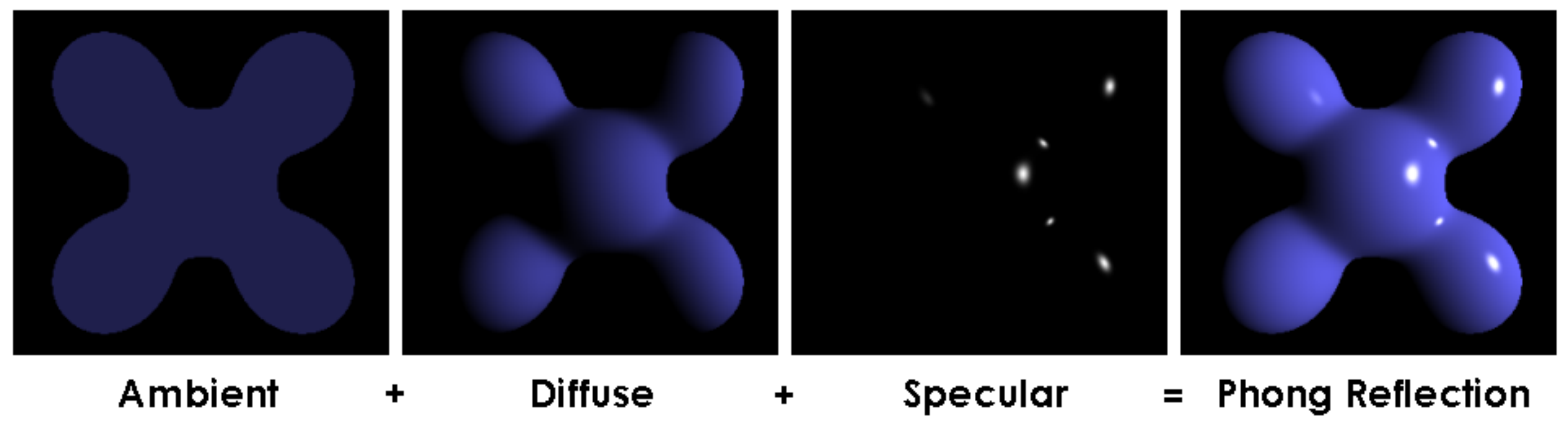
Coeficiente de absorção

Intensidade de entrada

Coeficiente de brilho (metal, plástico)



Modelo completo



Sugestão de Tarefa

- Descubra quais funções do OpenGL são responsáveis pelas definições dos tipos de fontes de luz, e como habilitá-las.
- Adicione uma fonte de luz direcional no exercício da esfera.