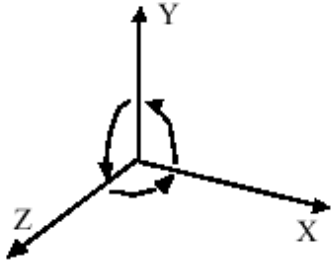
Transformações 3D

Soraia Raupp Musse

Transformações 3D

- Translação – glTranslatef(dx, dy, dz)
  - T(dx, dy, dz): $\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$

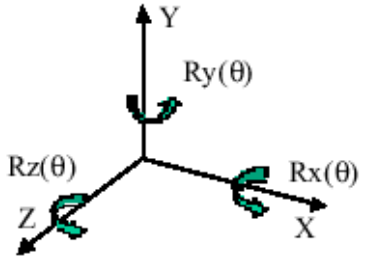


Sistema de Coordenadas
"mão direita"

- Escala – glScalef(Sx, Sy, Sz)
  - S(Sx, Sy, Sz): $\begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Transformações 3D

- Rotação – glRotatef(angle,x,y,z)



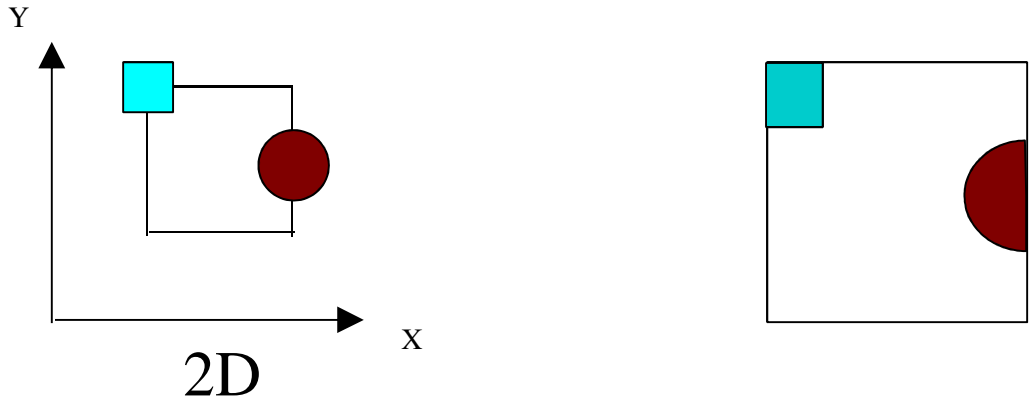
$$R_z(\theta): \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\theta): \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_x(\theta): \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

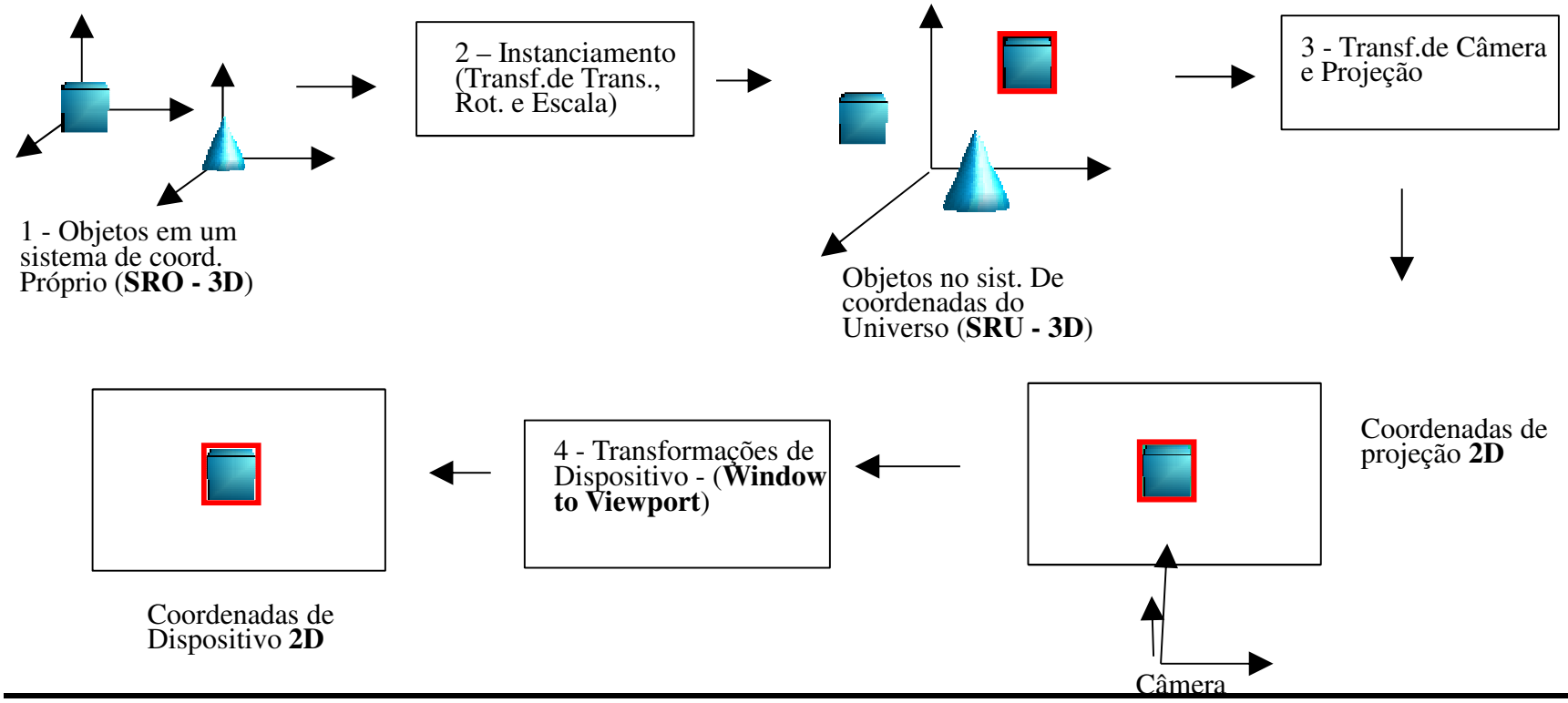
Pipeline de Visualização

- Em 2D as coisas são mais simples
  - Simplesmente especificar uma janela do mundo 2D e uma viewport na superfície de visualização
- A complexidade começa em 3D, pelo fato de termos uma dimensão a mais, mas também pelo fato do dispositivo de exibição ser 2D



Pipeline de Visualização 3D

- Para tirarmos uma foto virtual, precisamos executar os seguintes passos



Recorte 3D

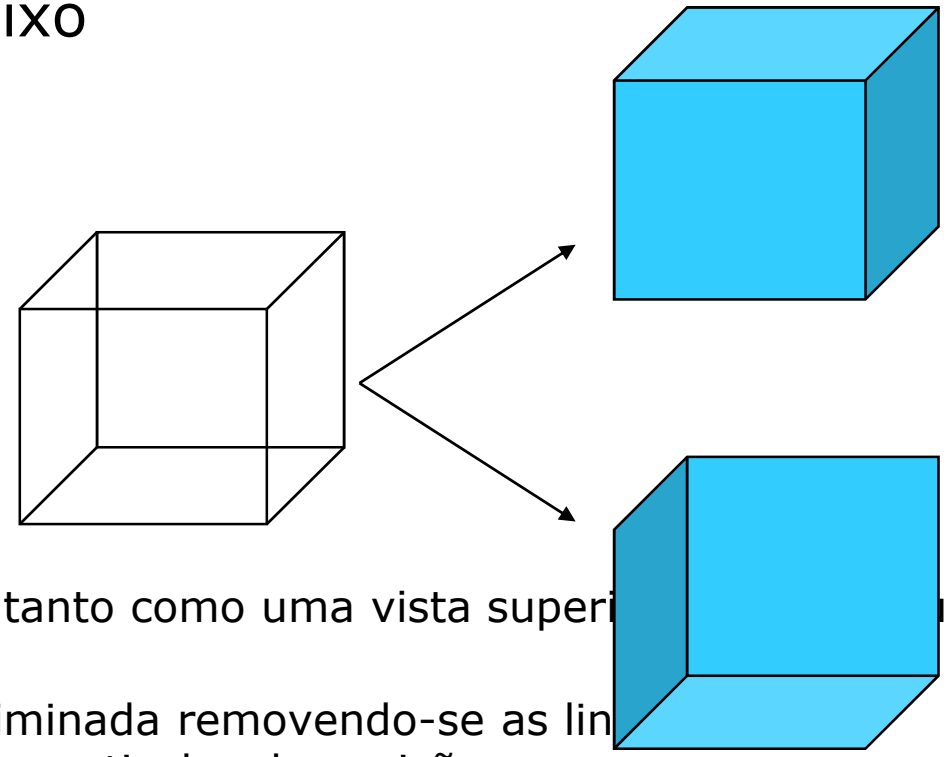
- Para o recorte em 3D, veremos a extensão do algoritmo de Cohen-Sutherland em 2D
- O algoritmo de Cohen-Sutherland classifica as extremidades das linhas de acordo com sua posição no espaço em relação ao volume de visualização
- Em 3D, esta classificação é formada por 6 bits de acordo com o seguinte
  - bit 1: ponto está acima do volume
  - bit 2: ponto está abaixo do volume
  - bit 3: ponto está à direita do volume
  - bit 4: ponto está à esquerda do volume
  - bit 5: ponto está atrás do volume
  - bit 6: ponto está à frente do volume

Remoção de Elementos Ocultos

1. Introdução
2. Remoção de Faces Traseiras
3. Algoritmo do Pintor
4. Algoritmo *Z-Buffer*

Introdução

- Uma das necessidades de eliminar superfícies escondidas está ilustrada na figura abaixo



  - O cubo pode ser interpretado tanto como uma vista superi… inferior/direita
  - Esta ambigüidade pode ser eliminada removendo-se as lin… superfícies que são invisíveis a partir das duas visões

Roteiro

- ✓ Introdução
- ✓ Remoção de Faces Traseiras
3. **Algoritmo do Pintor**
4. Algoritmo *Z-Buffer*

Algoritmo do Pintor (*Depth-Sorting Method*)

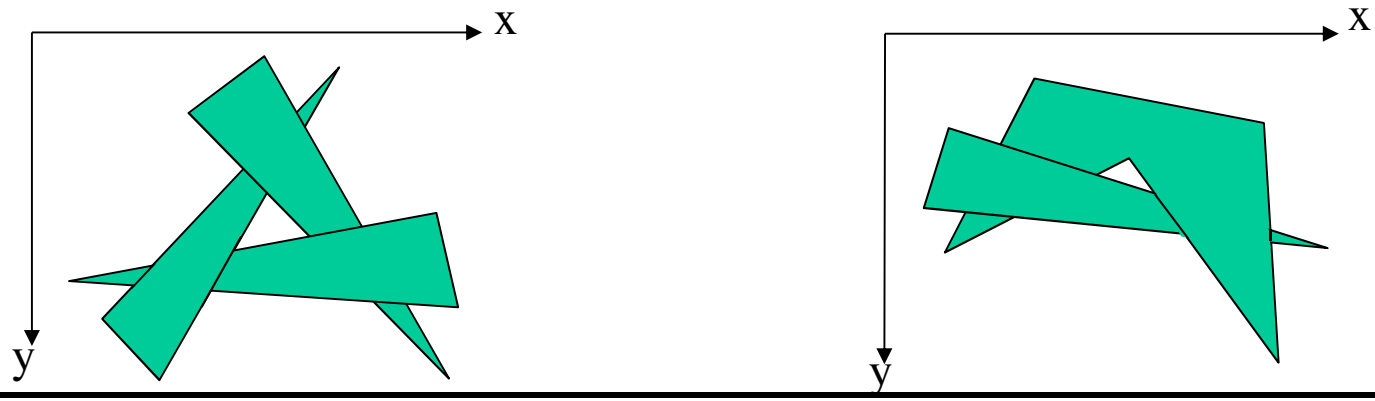
- Usa operações no espaço de imagem e no espaço de objeto
- Possui uma abordagem direta:
  1. Ordena todos os polígonos (faces) de acordo com a distância do observador (mais distante, maior coordenada Z)
  2. Resolve problemas de ambigüidade que podem ocorrer quando a extensão Z dos polígonos se sobrepõe (objetos se interseccionam)
  3. Pinta os polígonos na tela na ordem decrescente (mais distantes primeiro)
- Problema:
  - Ordenação não é trivial!

Algoritmo do Pintor

- Solução para o problema da ambigüidade
  - O teste é uma seqüência de cinco passos
    1. A extensão X dos polígonos não se sobrepõem, então os polígonos não se sobrepõem
    2. A extensão Y dos polígonos não se sobrepõem, então os polígonos não se sobrepõem
    3. *P* está totalmente atrás de *Q* , então os polígonos não se sobrepõem
    4. *Q* está totalmente atrás de *P*, então os polígonos não se sobrepõem
    5. As projeções dos polígonos no plano XY (tela) não se sobrepõem, então os polígonos não se sobrepõem
  - Se os cinco testes falharem, assume-se que P sobrepõe Q, e trocam-se suas posições na lista, marcando que Q foi movido para esta nova posição no fim da lista

Algoritmo do Pintor

- Idéia básica
  - Ordenar os polígonos pelas suas distâncias do observador, colocá-los em um *buffer* em ordem decrescente de distância e pintá-los de trás para frente
- Problema de ambigüidade quando há interseção entre os objetos da cena



Roteiro

- ✓ Introdução
- ✓ Remoção de Faces Traseiras
- ✓ Algoritmo do Pintor
4. **Algoritmo *Z-Buffer***
5. Árvores BSP

Algoritmo *Z-Buffer*

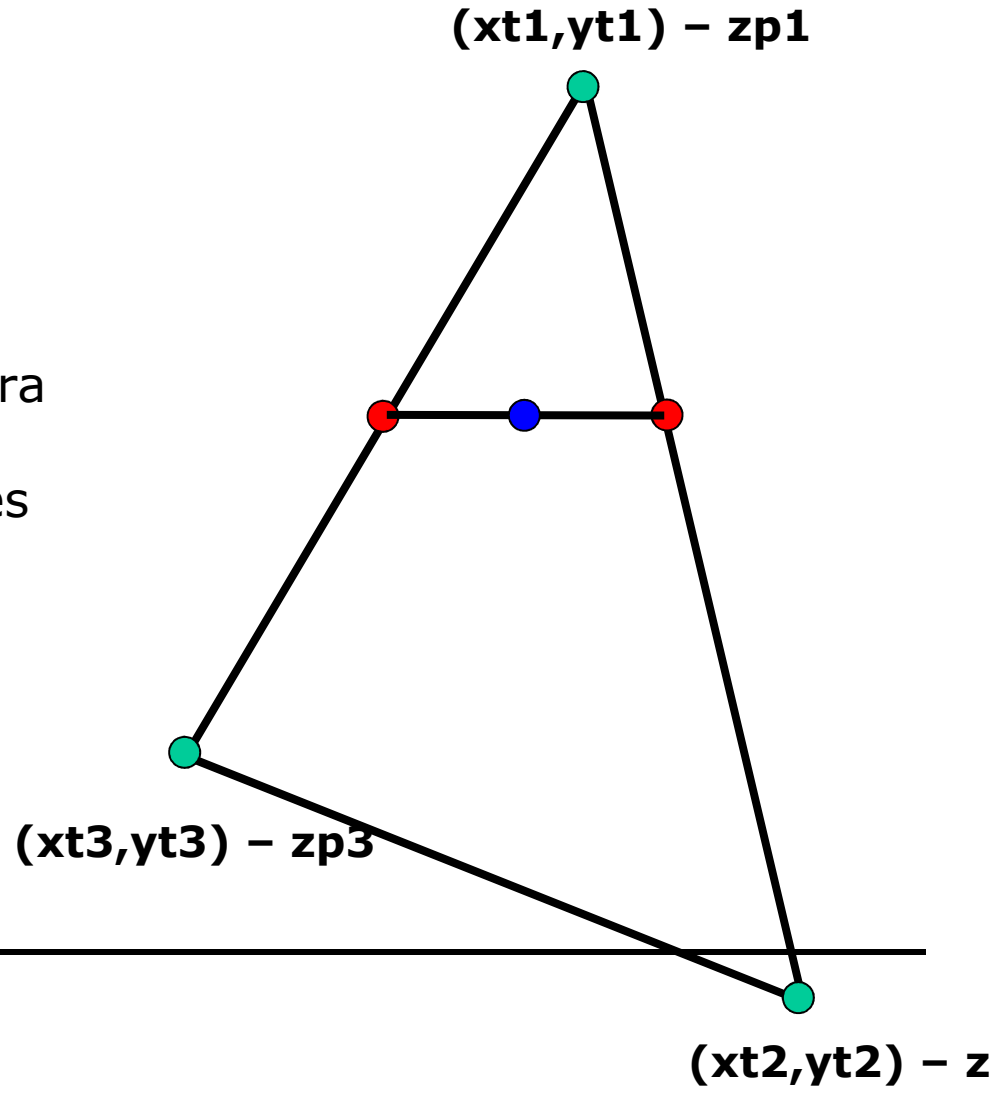
- Operações são realizadas no espaço de imagem
- Baseado no procedimento de preenchimento de polígonos tipo *scan-line*
- Polígonos
  - Já foram projetados (estão no SRP)
  - Já foram mapeados para o espaço de tela (SRT)
  - Porém mantém-se a coordenada Z (SRP) de cada vértice, de modo que seja possível recuperar a informação de profundidade

Algoritmo *Z-Buffer*

- Requer dois *buffers* (tamanho da tela)
  - *Buffer de Cor (Color Buffer*)
    - Armazena os valores de cor
    - Inicializado com a cor de fundo
  - *Buffer de Profundidade (Z-Buffer)*
    - Armazena os valores de Z (para cada pixel)
    - Inicializado com o maior valor possível para Z (depende da quantidade de bits por pixel no *z-buffer*)

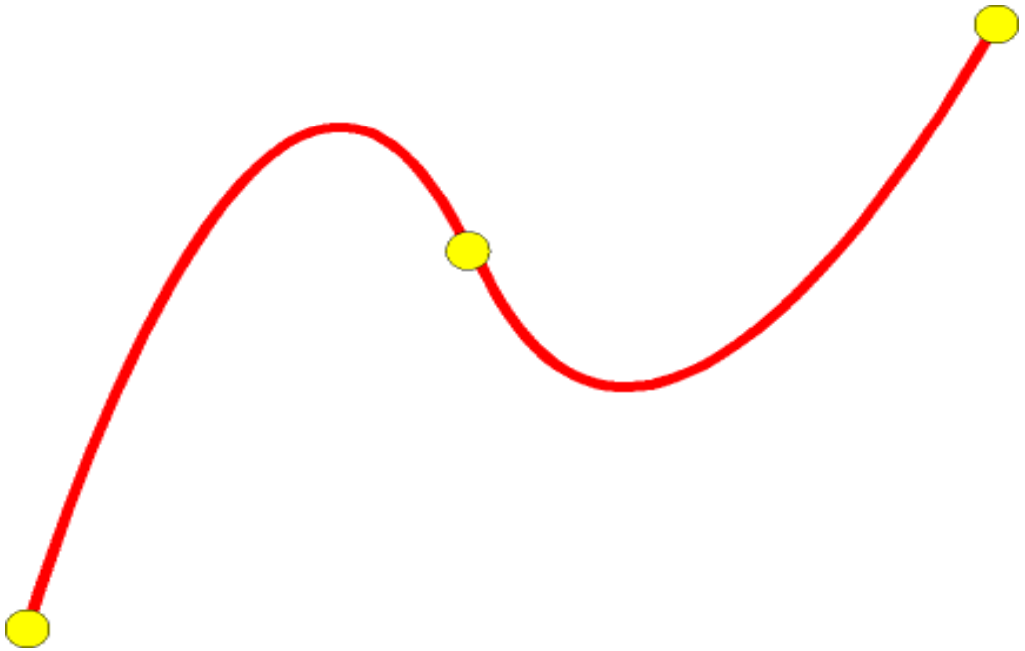
Algoritmo *Z-Buffer*

- Para cada polígono a ser exibido
  - Calcula os coeficientes da equação do plano do polígono
  - Coeficientes são usados para calcular os valores de Z no interior do polígono, através de interpolação

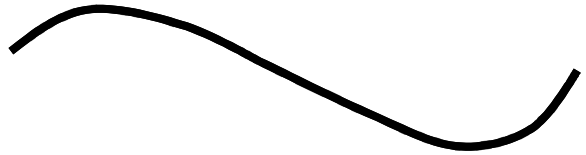


Curvas

- Curvas
  - Apenas comprimento

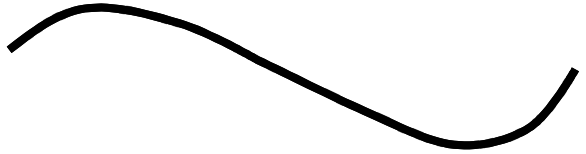


Interpolação X Aproximação



Na interpolação, a curva passa sobre todos os pontos definidos.

Hermite, Catmull-Rom spline

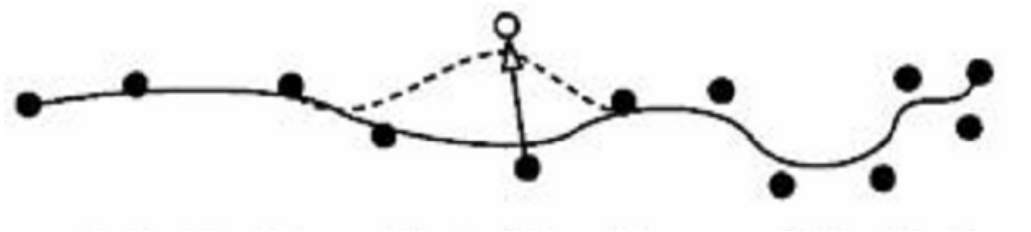


Na aproximação, a curva começa sobre o ponto inicial e termina sobre o final. Os demais pontos são aproximados.

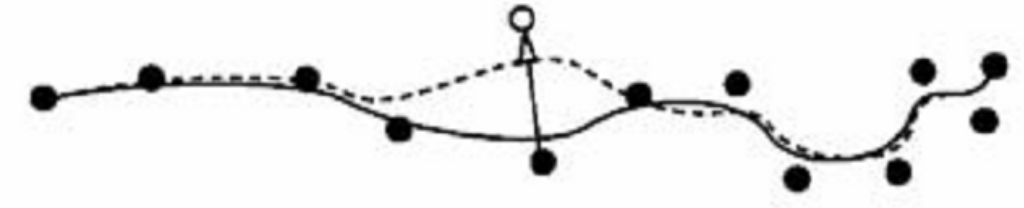
Bézier, curvas B-spline

Controle

- Local



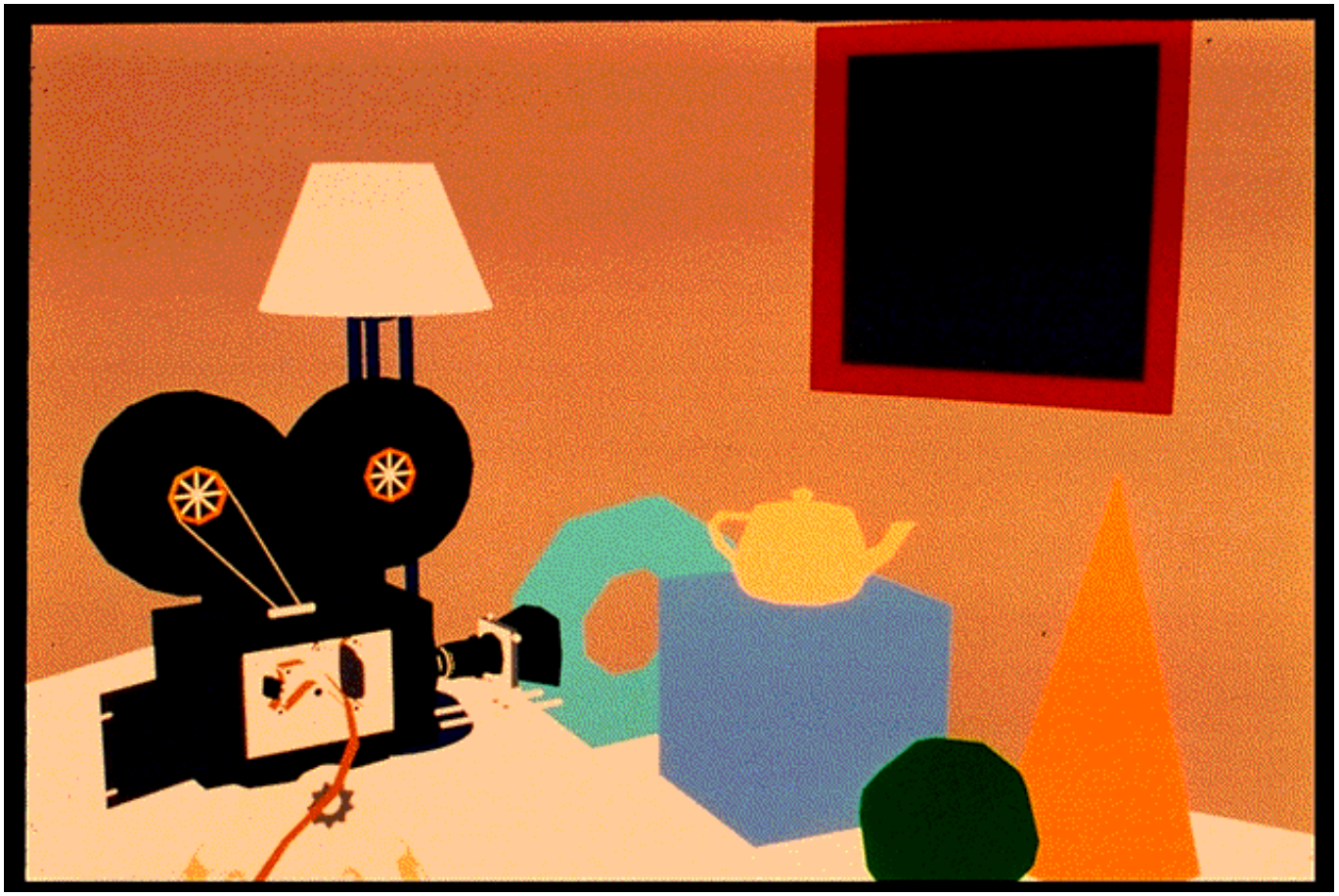
- Global



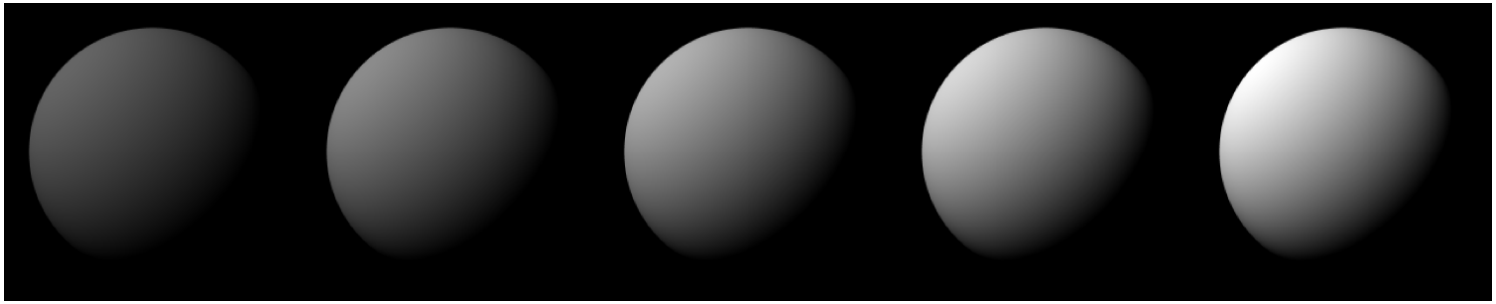
Métodos para representar curvas

- Hermite
- Bézier
- B-Spline

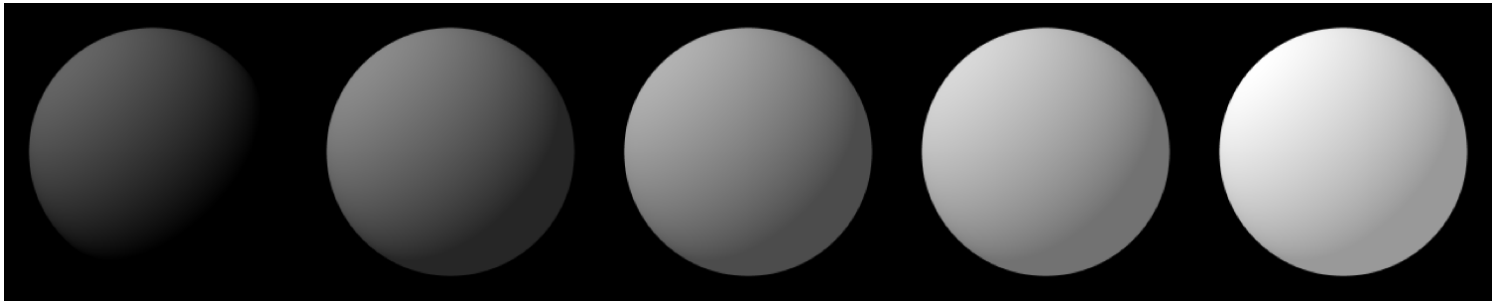
Luz Ambiente



Reflexão Difusa



Kd=0.4, 0.55, 0.7, 0.85, 1.0



Ia=Ip=1.0, Kd=0.4, Ka=0.0, .015, 0.3, 0.45, 0.6

Reflexão Especular de Phong

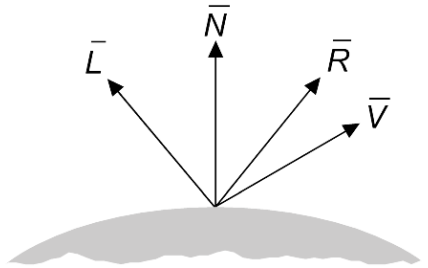
- Phong desenvolveu em 1975, um modelo para a parcela de reflexão especular que assumia que:
  - Máxima reflexão especular ocorre quando α^* é zero e cai rapidamente quando α cresce
  - Essa mudança é representada por $\cos^n \alpha$, onde n é o expoente de reflexão especular
    - Valores de n variam de 1 até centenas, dependendo do material simulado
- A equação de iluminação fica (O=cor)

considerando vetores normalizados,

$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} [K_d O_{d\lambda} (N.L) + K_s (R.V)^n]$$

incluindo a cor especular

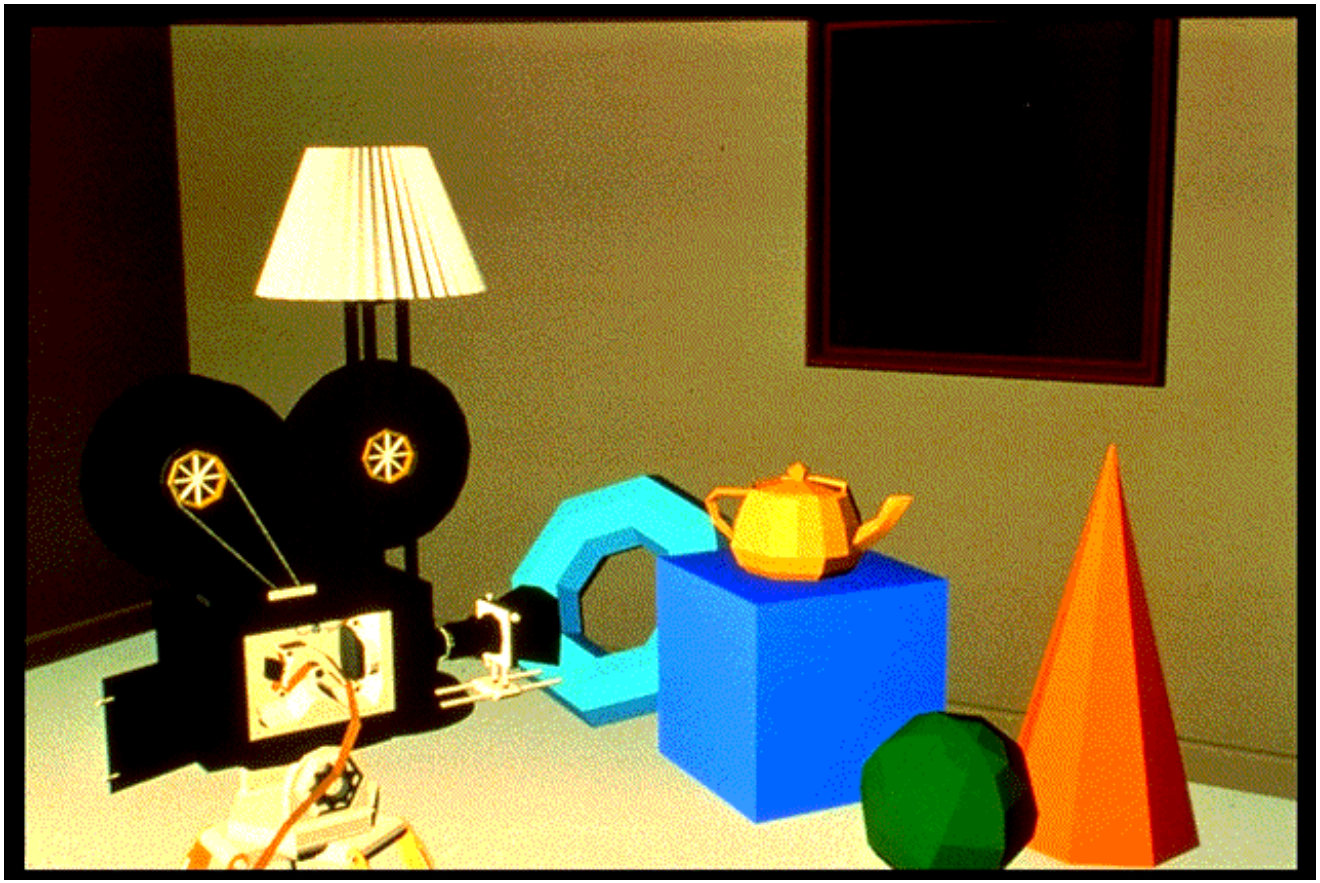
$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + f_{att} I_{p\lambda} [K_d O_{d\lambda} (N.L) + K_s O_s (R.V)^n]$$



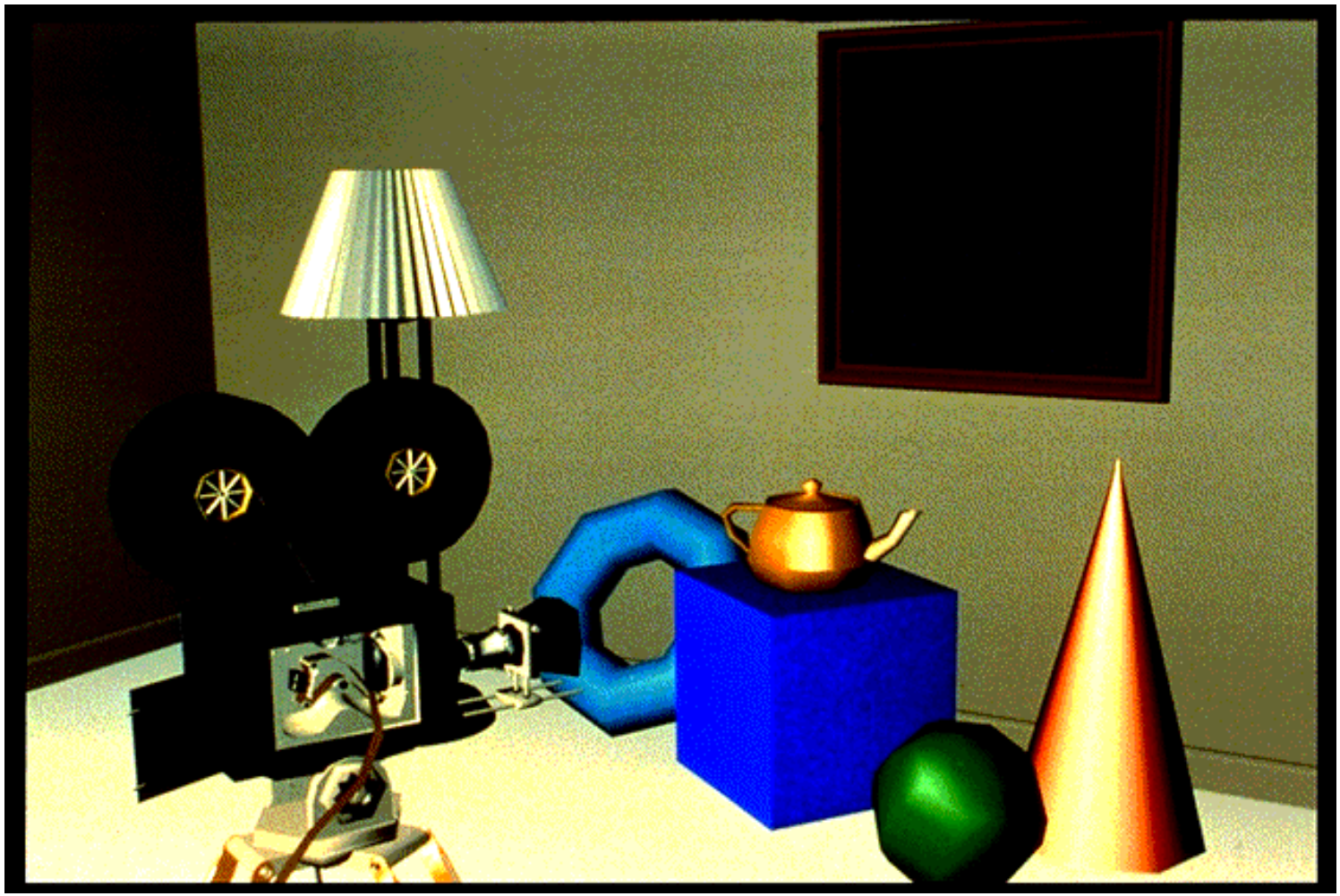
R = reflexão e V=viewpoint

*α é ângulo entre R e V

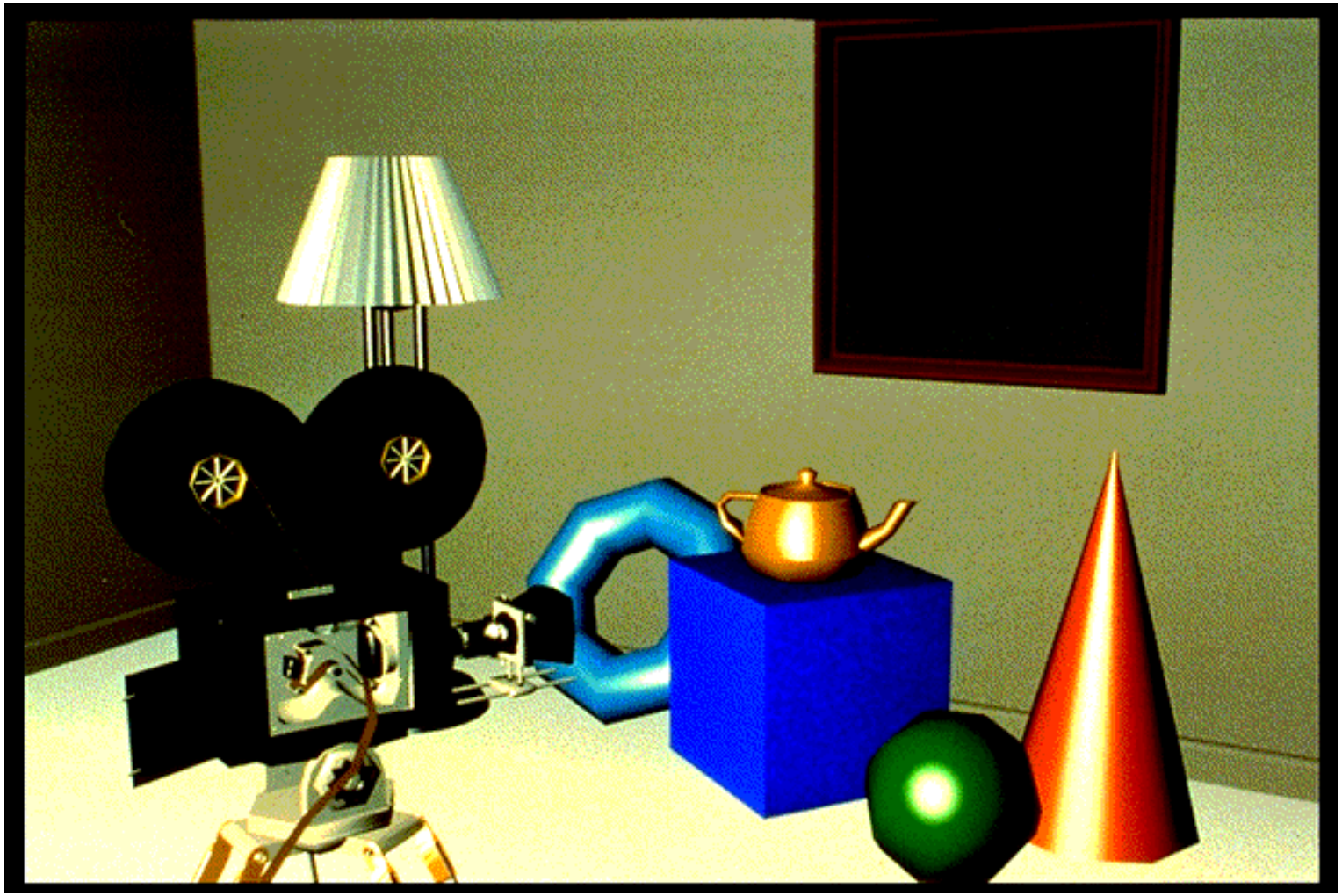
Sombreamento de Malhas Poligonais



Gouraud Shading

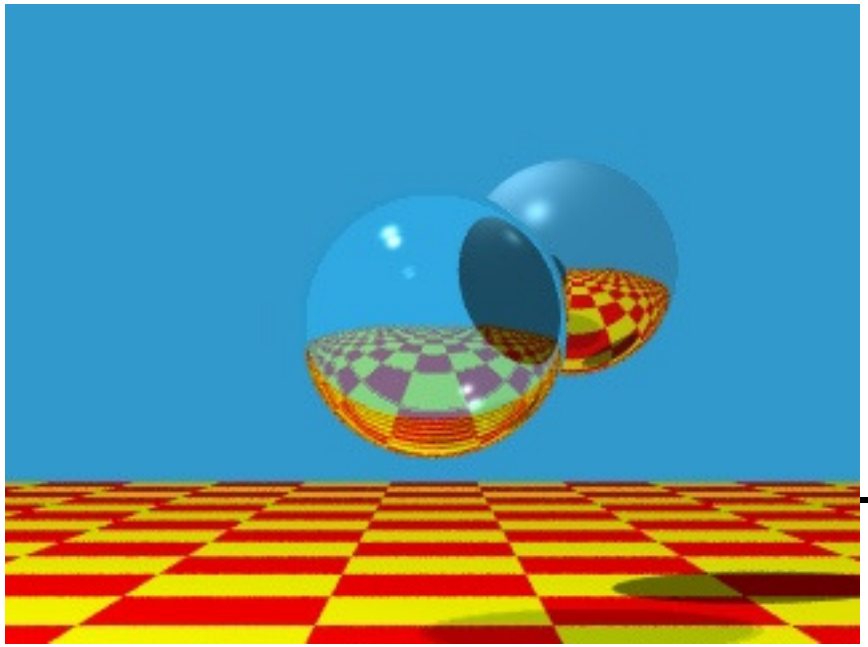


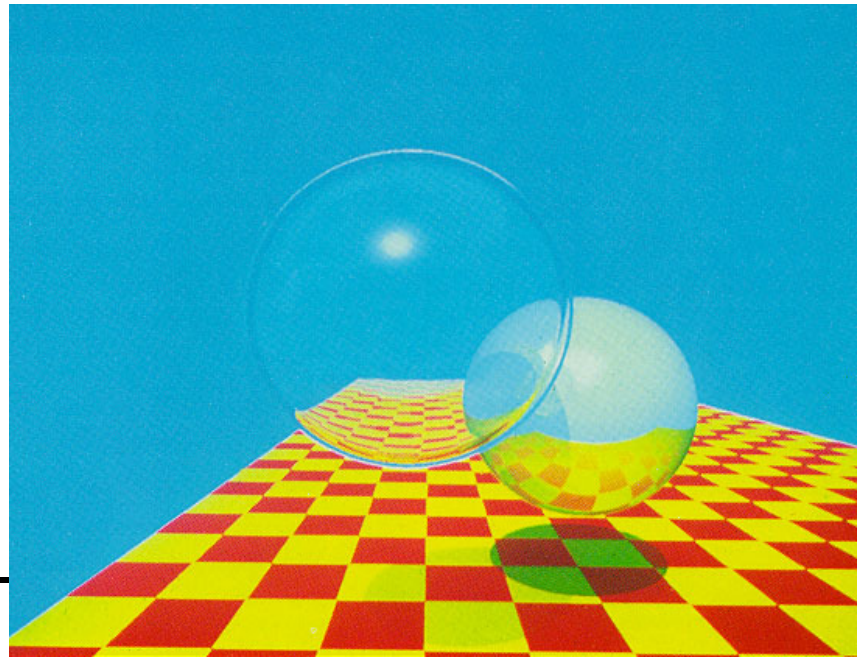
Phong Shading



Traçado de Raios

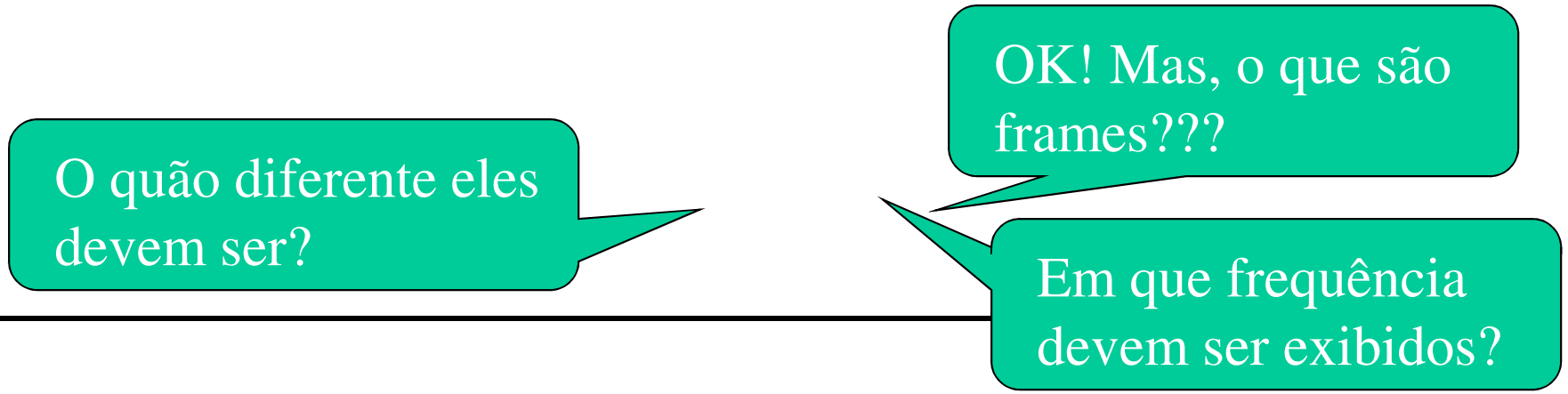
- Primeiras idéias em 1968 (*Ray Casting*)
- 1980 Turner Whitted
  - *Communications of the ACM* N. 23, V. 6, June 1980, p. 343-349)





O que significa o termo *"Animação Computadorizada?"*

- *"Geração de frames consecutivos que são exibidos numa freqüência suficiente para que o olho humano não consiga diferenciá-los e tenha então a sensação de movimento"*



Técnicas de Animação

Keyframe (Animação por quadros-chave)

- keyframes são especificados pelo animador e o computador gera os quadros intermediários
- Os quadros intermediários são gerados baseados numa lei de *interpolação*

Técnicas de Animação
Scripting Systems

- Visa oferecer uma linguagem de script (com comandos pré-definidos) que possa descrever a animação de objetos
  - Ex: #ACTORS 5
    - » ACTOR_1 FRAME=1 POSITION 10 10 10
    - » ACTOR_1 FRAME=100 POSITION 20 20 20

Classificação de técnicas

Diferenças	Low-level	High-level
Intervenção do usuário	Muita	Pouca
Nível de abstração	Pouca	Muita
Precisão do resultado em relação ao especificado	Muita	Pouca