

## Introdução à Computação Gráfica Ray Tracing

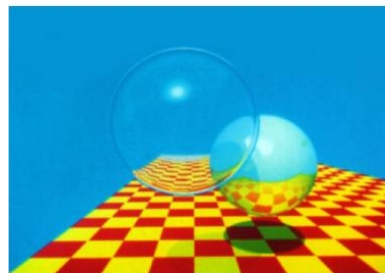
Claudio Esperança  
Paulo Roma Cavalcanti

### Características Principais

- Tipicamente implementado em Software
- Combina um modelo de iluminação com determinação de visibilidade
- Simula efeitos de iluminação global tais como
  - ♦ Sombras
  - ♦ Reflexão especular e refração recursivas
  - ♦ Acompanha vários caminhos da luz
- Desvantagens
  - ♦ Lento
  - ♦ Não simula reflexão difusa recursiva

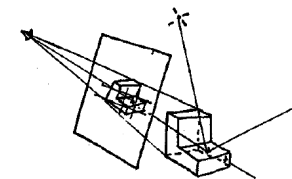
### Contexto Histórico

- Trabalhos Seminais
  - ♦ Appel 68
  - ♦ Whitted 80
- Pesquisa
  - ♦ Uso de diferentes primitivas geométricas
  - ♦ Técnicas de aceleração
- Pesquisa recente
  - ♦ Ray tracing em tempo real
  - ♦ Arquiteturas para Ray tracing em HW



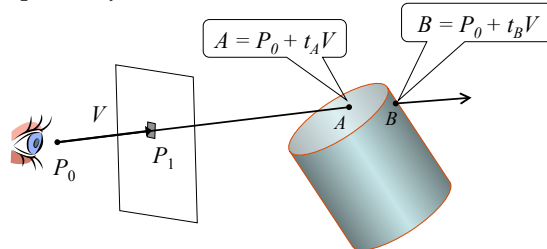
### Ray Casting

- Introduzido por Appel (1968)
- Raios são lançados passando pelo olho e por cada pixel da imagem
  - ♦ Teste de interseção entre cada objeto da cena e raio
  - ♦ Pixel é pintado com cor do objeto mais próximo (do olho)
  - ♦ Sombras são calculadas lançando raios desde o ponto do objeto até a fonte de luz



## Interseção Raio / Objeto

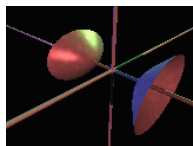
- Raio é modelado como uma reta em forma paramétrica:  $R(t) = P_0 + t(P_1 - P_0) = P_0 + tV$
- Computa-se para quais valores do parâmetro  $t$  a reta intercepta o objeto



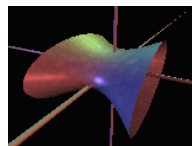
## Objetos Implícitos

- Objeto implícito é dado por uma equação da forma  $f(x, y, z) = 0$
- Muitas superfícies importantes podem ser modeladas como objetos implícitos principalmente os dados por equações polinomiais
  - ♦ Planos (grau 1)
  - ♦ Quádricas (grau 2)
    - elipsóides, cones, parabolóides, hiperbolóides
  - ♦ Quárticas (grau 4)
    - Toros

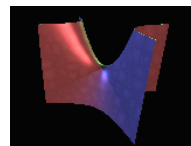
## Quádricas



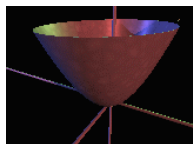
Hiperbolóide de duas folhas



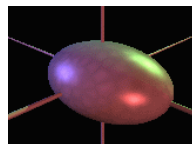
Hiperbolóide de uma folha



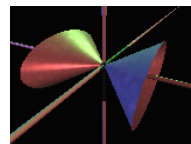
Parabolóide Hiperbólico



Parabolóide de revolução



Elipsóide



Cone (Hiperbolóide degenerado)

## Interseção Raio / Objeto Implícito

- Raio é modelado em forma paramétrica:  

$$R(t) = [R_x(t) \ R_y(t) \ R_z(t)]^T$$
- Logo, os pontos de interseção satisfazem  

$$f(R_x(t), R_y(t), R_z(t)) = 0$$
- Basta resolver a equação para determinar o(s) valor(es) de  $t$  que a satisfazem

### Exemplo: Interseção com Esfera

- Esfera de raio 1 centrada na origem:  
 $x^2 + y^2 + z^2 - 1 = 0$

- Raio parametrizado como:  
 $[V_x t + P_x \quad V_y t + P_y \quad V_z t + P_z]^T$

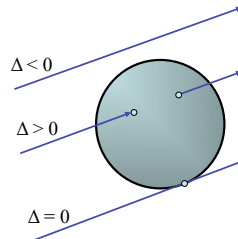
- Logo,  
 $(V_x t + P_x)^2 + (V_y t + P_y)^2 + (V_z t + P_z)^2 - 1 = 0$   
ou  
 $at^2 + bt + c = 0$   
onde

$$a = V_x^2 + V_y^2 + V_z^2$$

$$b = 2(V_x P_x + V_y P_y + V_z P_z)$$

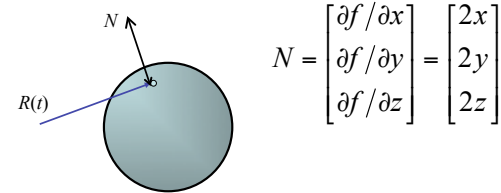
$$c = P_x^2 + P_y^2 + P_z^2 - 1$$

- Seja  $\Delta = b^2 - 4ac$ , então  $t = \frac{-b \pm \sqrt{\Delta}}{2a}$



### Computando a Normal no Ponto de Interseção

- Normal é dada pelo gradiente no ponto de interseção

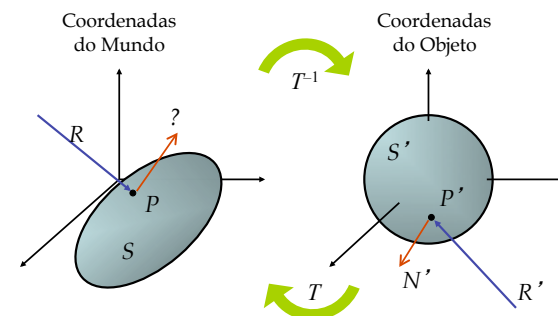


$$N = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \\ \partial f / \partial z \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \\ 2z \end{bmatrix}$$

### Interseção com Objetos Transformados

- As rotinas de interseção normalmente lidam com objetos primitivos de tamanho, posição e orientação fixas (ex.: esfera de raio unitário na origem)
- Para obter objetos genéricos, usa-se transformações lineares afim
- Para computar a interseção de um raio  $R$  com um objeto transformado  $S = T S'$ :
  - Leva-se o raio para o sistema de coordenadas da primitiva:  $R' = T^{-1} R$
  - Computa-se o ponto  $P'$  resultante da interseção  $R' \times S'$
  - O ponto de interseção é trazido de volta ao sistema de coordenadas do mundo:  $P = T P'$

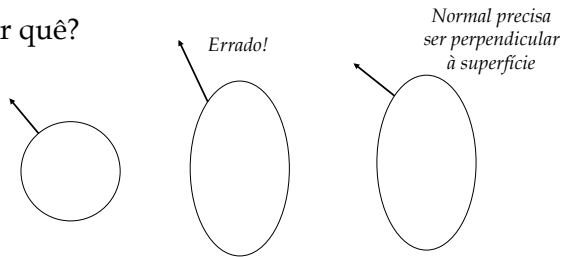
### Interseção com Objetos Transformados



## Transformando Normais

- Ao contrário do que nossa intuição indica,  $N' \neq T N$

- Por quê?



## Transformando Normais

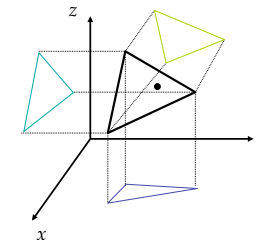
- Se a transformação não envolve deformação, isto é, é composta apenas de transformações rígidas e escalas uniformes, ela pode ser aplicada também à normal
- Para transformações afim genéricas, entretanto,  $N' = (T^{-1})^T N$
- Prova:
  - Queremos que  $N'$  seja perpendicular a qualquer vetor  $V'$  sobre o plano tangente à superfície:  $N' \cdot V' = 0$
- Sabemos que  $V' = T V$
- Então,  $N' \cdot (T V) = 0$  ou,  $(N' \cdot T) V = 0$
- Como o produto escalar de dois vetores  $A$  e  $B$  denotados por matrizes coluna pode ser escrito  $A^T B$ , então,  $N'^T T V = 0$
- Como  $A = A^{TT}$ , então  $N'^T T^T T V = 0$
- Lembrando que  $(AB)^T = B^T A^T$  então  $(T^T N')^T V = 0$  ou  $(T^T N') \cdot V = 0$
- Portanto,  $(T^T N') = N$
- Resolvendo para  $N'$  temos  $N' = (T^{-1})^T N$

## Interseção com Planos

- Plano em forma implícita  $Ax + By + Cz + D = 0$
- Se queremos um plano que passa por um ponto  $Q$  e tem normal  $N$  podemos escrever  $(P - Q) \cdot N = 0$
- Resolução da forma habitual
- Entretanto, normalmente não temos planos ilimitados, mas sim polígonos planares!

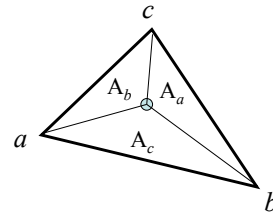
## Interseção com Triângulos

- Computa-se interseção com o plano que contém o triângulo
- O ponto de interseção está dentro do triângulo?
- O teste é feito sobre a projeção do triângulo sobre um dos planos coordenados ( $x$ - $y$ ,  $y$ - $z$  ou  $x$ - $z$ )
- Qual? Escolhe-se o plano para o qual a projeção tem maior área  $Ax + By + Cz + D = 0$ 
  - Se  $|A| > |B|, |C| \rightarrow$  plano  $y$ - $z$
  - Se  $|B| > |A|, |C| \rightarrow$  plano  $x$ - $z$
  - Se  $|C| > |A|, |B| \rightarrow$  plano  $x$ - $y$



## Interseção com Triângulos

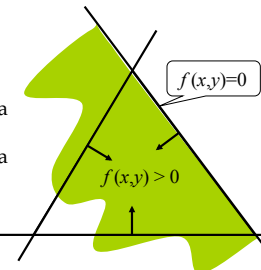
- Como determinar se o ponto está dentro do triângulo?
- Uma idéia é computar as coordenadas baricênticas do ponto de interseção:  
 $P = \alpha a + \beta b + \gamma c$ , onde  $\alpha + \beta + \gamma = 1$
- $P$  está dentro do triângulo sse  $P$  é uma combinação convexa de  $a, b, c$ , isto é,  $0 \leq \alpha, \beta, \gamma \leq 1$
- As coordenadas baricênticas correspondem às áreas relativas dos triângulos que unem o baricentro aos vértices



$$\begin{aligned}\alpha &= A_a / A \\ \beta &= A_b / A \\ \gamma &= A_c / A\end{aligned}$$

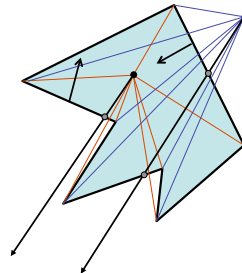
## Interseção com Polígonos Convexos

- Uma outra idéia que também funciona com qualquer polígono convexo é considerar o polígono a interseção de semiespaços planos em 2D
- Cada aresta é colinear com uma reta dada por  $f(x,y) = ax + by + c = 0$
- Pode-se escolher  $a, b$  e  $c$  de tal forma que o interior do polígono corresponda a  $f(x,y) > 0$
- Para saber se o ponto de interseção está no interior (ou na borda) do polígono, basta testar o ponto com relação a todas as arestas



## Interseção com Polígonos Quaisquer

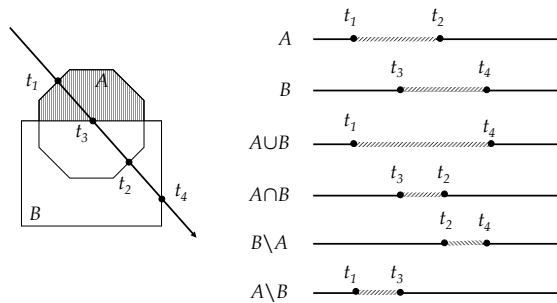
- Diversos métodos
  - ♦ Soma dos ângulos
    - Dentro:  $360^\circ$
    - Fora:  $0^\circ$
  - ♦ Regra de paridade (teorema de Jordan)
  - ♦ Ray-Casting em 2D
    - Semelhante à regra de paridade
    - Apenas a normal da aresta mais próxima é examinada



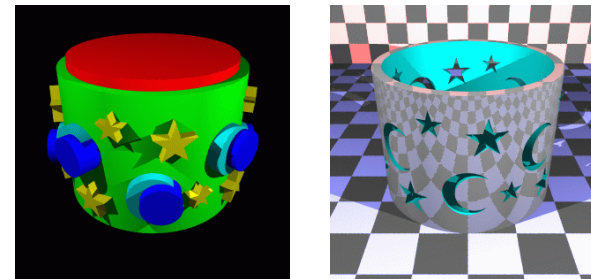
## Interseção com Sólidos CSG

- Ray-tracing provê um método direto de visualização de sólidos CSG (sem avaliação de bordo)
- A interseção com primitivas é feita como antes, mas todos os pontos interseção são guardados
  - ♦ O resultado é uma estrutura de dados que registra os intervalos em que o raio está dentro, fora, ou na fronteira da primitiva
- Para computar as operações de conjunto ( $\cap$ ,  $\cup$ ,  $\setminus$ ) os intervalos são combinados de maneira apropriada

## Interseção com Sólidos CSG



## Ray Tracing de Sólidos CSG



## Interseção com Superfícies Paramétricas

- Superfícies paramétricas são dadas por

$$S(u, v) = [S_x(u, v) \ S_y(u, v) \ S_z(u, v)]^T$$

- Raio é representado como a interseção de dois planos

$$A_1x + B_1y + C_1z + D_1 = 0$$

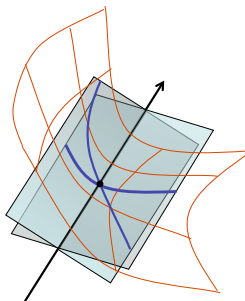
$$A_2x + B_2y + C_2z + D_2 = 0$$

- Substituindo, temos

$$A_1S_x(u, v) + B_1S_y(u, v) + C_1S_z(u, v) + D_1 = 0$$

$$A_2S_x(u, v) + B_2S_y(u, v) + C_2S_z(u, v) + D_2 = 0$$

- Cada equação representa uma curva de interseção



## Interseção com Superfícies Paramétricas

- Ponto de interseção computado resolvendo um sistema de 2 equações com 2 incógnitas
  - ♦ Se equações são polinomiais, pode-se usar eliminação ou outras técnicas algébricas
    - Exemplo: 2 equações cúbicas podem ser transformadas em uma equação de sexto grau [Kajiya]
  - ♦ Pode-se também usar métodos numéricos
    - Método de iteração de Newton [Toth]
- Procedimentos muito dispendiosos
  - ♦ Usar métodos de aceleração

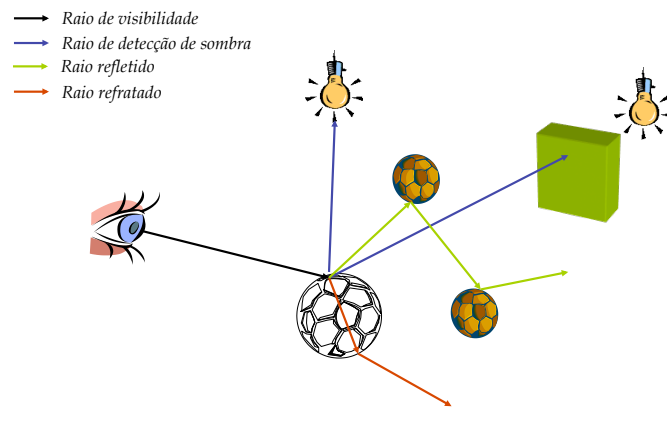
## Outros Objetos

- Superfícies de varredura (*sweep*)
  - ♦ Translação (cilíndrica / cônica)
  - ♦ Revolução
  - ♦ Varredura genérica
- Terrenos (*height fields*)
- Blobs (superposição de campos escalares exponenciais)

## Ray Tracing Recursivo

- Introduzido por Turner Whitted: “*An Improved Illumination Model for Shaded Display*” (1980)
- Idéia de traçar os raios de luz desde as fontes até o olho existia há muito tempo (*Forward Ray Tracing*)
- Whitted mostra como computar um modelo de iluminação aproximado acompanhando os raios de luz que chegam ao olho no sentido inverso (*Backward Ray Tracing*)

## Ray Tracing Recursivo



## Ray Tracing Recursivo

- Para cada pixel da imagem
  - ♦ Calcular raio que passa pelo pixel e pelo olho
  - ♦ Determinar objeto atingido pelo raio
    - Ponto de interseção
    - Normal
    - Propriedades de material
    - Propriedades de textura
  - ♦ Computar contribuição da iluminação ambiente
  - ♦ Para cada fonte de luz, determinar visibilidade (raios de detecção de sombra)
    - Se fonte visível, somar contribuição reflexão difusa
  - ♦ Se limite de recursão não foi atingido
    - Somar contribuição reflexão especular acompanhando raio refletido
    - Somar contribuição de transmissão acompanhando raio refratado

## Técnicas de Aceleração

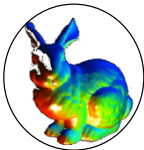
- O que faz com que o Ray Tracing seja lento?
  - ♦ Número de interseções entre raios e objetos
  - ♦ Custo de cada cálculo de interseção

## Técnicas de Aceleração

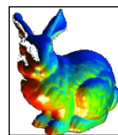
- Reduzir o número de interseções
  - ♦ Hierarquias de volumes limitantes
  - ♦ Subdivisão espacial
- Aceleração dos cálculos de interseções
  - ♦ Volumes limitantes
  - ♦ Especialmente superfícies paramétricas
- Determinar mais rapidamente o primeiro objeto interceptado (visibilidade)
  - ♦ Usar hardware gráfico (z-buffer)

## Volumes Limitantes

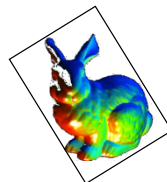
- Objetos complexos são envolvidos em objetos simples cujo cálculo de interseção é + fácil
- Serve para eliminar objetos que não interceptam raio



*Esfera*  
+Rápida, mas  
pouco justa



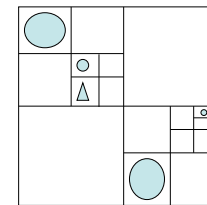
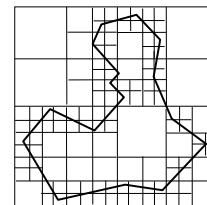
*Caixa Alinhada  
com Eixos:*  
Rápido, justo



*Caixa Orientada*  
Rápida, +justa

## Hierarquias de Volumes

- Podem ser aplicadas aos próprios objetos ou a cenas
- Raios são recursivamente testados com relação aos nós da árvore
  - ♦ Nós + próximos do olho são testados primeiro
  - ♦ Nós vazios são trivialmente rejeitados





### Usando Hardware para determinar Visibilidade

- Hardware é geralmente mais rápido que software
- Desenha-se os objetos usando acelerador gráfico
- O valor de profundidade armazenado no z-buffer pode ser usado para determinar as coordenadas do mundo dos objetos mais próximos
  - ♦ Um teste exato é feito apenas sobre esses objetos
- Uma outra idéia é usar uma cor para desenhar cada objeto (*item buffer*)
  - ♦ A cor vai designar qual o objeto mais próximo
- Essas técnicas podem ser usadas em qualquer estágio do ray tracing