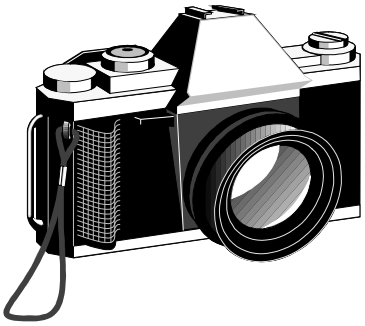
Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação

Rendering: *Ray Tracing*

Introdução

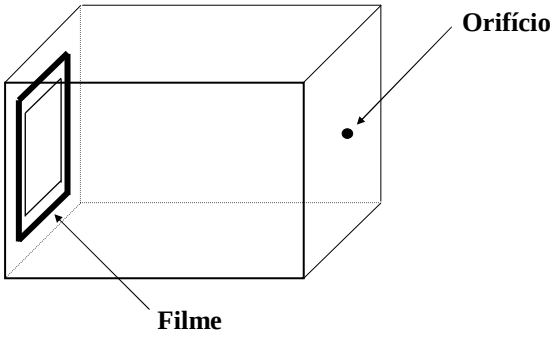


- Princípio: síntese de imagens 3D
- Simulação fotográfica

2

Introdução

O princípio da câmera de orifício (*Pinhole Camera*)

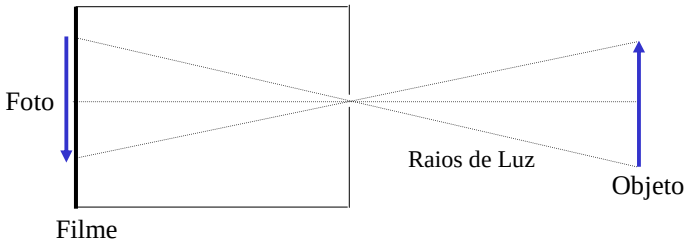


3

Introdução

Mundo Real

Objetos + Luz + Filme = Fotografia

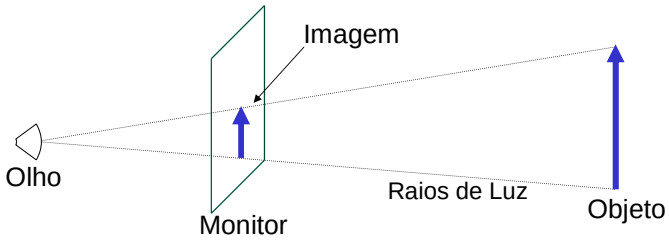


4

Introdução

Simulação por Computador

Objetos + Luz + Monitor = Imagem



5

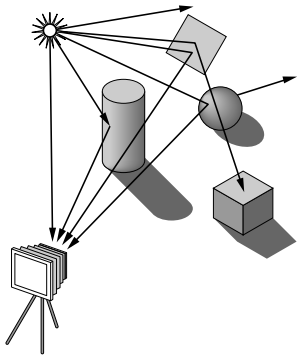
Introduction

- OpenGL is based on a pipeline model in which primitives are rendered one at time
  - No shadows (except by tricks or multiple renderings)
  - No multiple reflections
- Global approaches
  - Rendering equation
  - Ray tracing
  - Radiosity

6

Ray Tracing

- Follow rays of light from a point source
- Can account for reflection and transmission



7

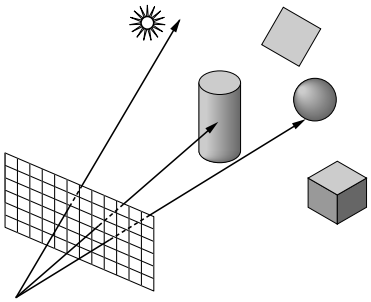
Computation

- Should be able to handle all physical interactions
- Ray tracing paradigm is not computational
- Most rays do not affect what we see
- Scattering produces many (infinite) additional rays
- Alternative: ray casting

8

Ray Casting

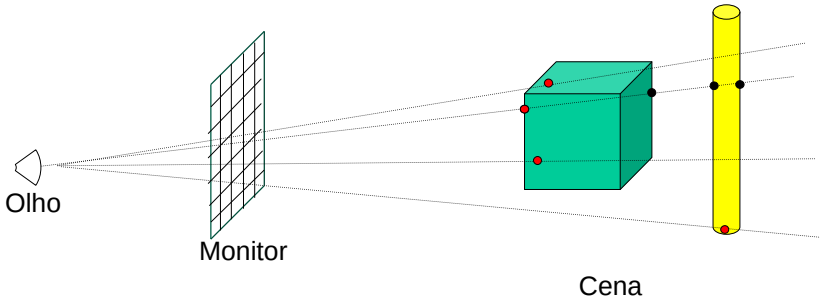
- Only rays that reach the eye matter
- Reverse direction and cast rays
- Need at least one ray per pixel



9

Ray Casting

Disparo de Raios (*Ray Casting*)



10

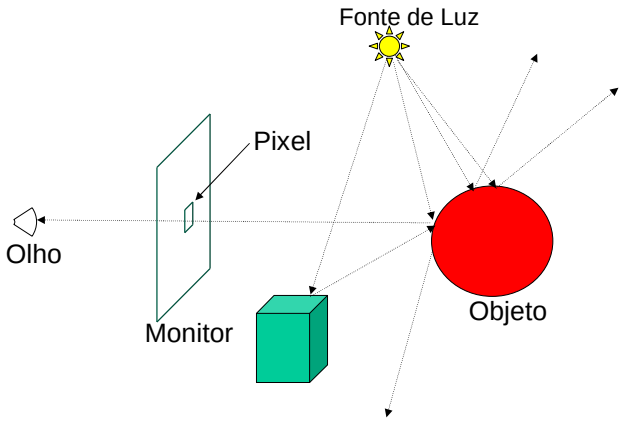
Ray Tracing

- Dispara um raio que sai do olho do observador, passa pelo pixel e atinge a cena
- Se o raio intercepta objeto da cena, calcula a cor resultante da **iluminação direta** nesse ponto
- Entretanto, se objetos são de material parcialmente reflexivo, ou parcialmente transparente (ou ambos), a cor da superfície no ponto é influenciada por raios que atingem o ponto refletidos/transmitidos por outras superfícies
- Estes raios são traçados de volta à sua origem, para determinar sua contribuição

11

Ray Tracing

***Forward* (traçado progressivo)**



12

Ray Tracing

- Traçado de um raio termina quando ele intercepta o fundo da cena, ou quando existem tantas intersecções entre o início do raio e o observador que a contribuição adicional desse raio para a cor final da imagem no ponto é desprezível
- Algoritmo dispara pelo menos um raio por pixel: *image order*, ou *image space*

13

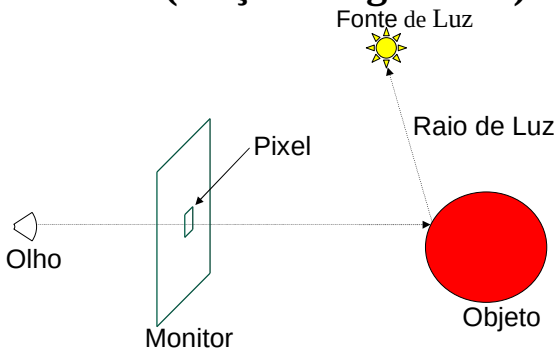
Ray Tracing Regressivo

- Por questão de eficiência, traçamos o caminho inverso do raio
  - Saindo do pixel em direção à cena, determinando as contribuições de todos os raios da cena para a cor desse pixel
- Raios saem da posição de observação, passam pelo centro de cada pixel da tela, e são traçados de volta a medida em que interagem com as superfícies dos objetos da cena

14

Ray Tracing

Backward (traçado regressivo)



15

Ray Tracing

- Para cada pixel
  - dispara o raio na cena, e determina as superfícies interceptadas por ele
  - para cada superfície interceptada, calcula a distância do ponto de intersecção ao pixel
  - a menor distância identifica a superfície visível para o observador
  - em seguida, o raio é refletido de volta (para a sua origem) e transmitido de volta (para a sua origem), gerando raios secundários
  - o procedimento de traçado é repetido para cada raio secundário

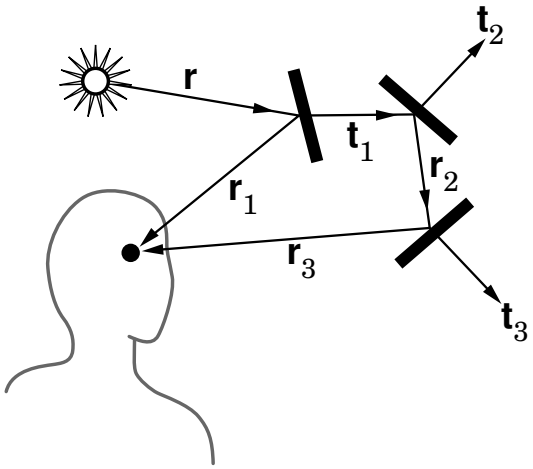
16

Ray Tracing

- Árvore de raios
  - Cada nó representa uma intersecção, e tem 1 ou 2 filhos
  - Traçado de um raio termina quando ele atinge uma fonte de luz, ou depois de um número máximo de intersecções

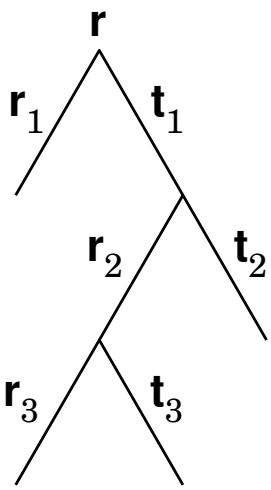
17

Ray Trees



18

Ray Tree



19

Ray Tracing

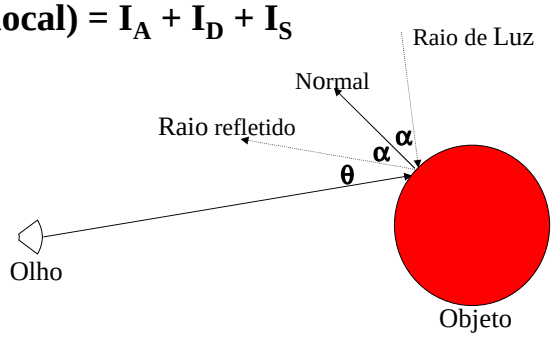
- Intensidade atribuída a um pixel
  - Determinada acumulando-se as intensidades na sua árvore, a partir dos nós terminais até a raiz
  - Em cada nó, a intensidade calculada da superfície é atenuada segundo a distância à superfície anterior (nó pai) e acumulada à intensidade dessa superfície
  - **A intensidade do pixel é dada pela soma das intensidades atenuadas no nó raiz**

20

Modelo de Iluminação

Modelo Local Completo + contribuição da luz devido aos outros objetos da cena

$$I(local) = I_A + I_D + I_S$$



21

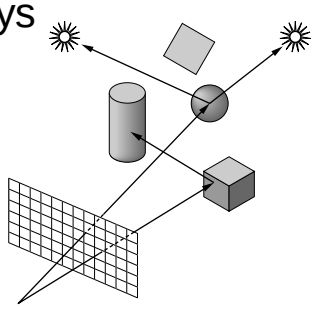
Modelo de Iluminação

- Modelo Global
  - modelo local completo +
  - sombras
  - reflexões múltiplas
  - transparência
  - texturas
- ver programa demo em http://www.siggraph.org/education/materials/HyperGraph/raytrace/rt_java/raytrace.html

22

Shadow Rays

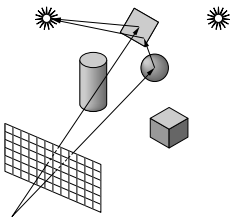
- Even if a point is visible, it will not be lit unless we can see a light source from that point
- Cast shadow or feeler rays



23

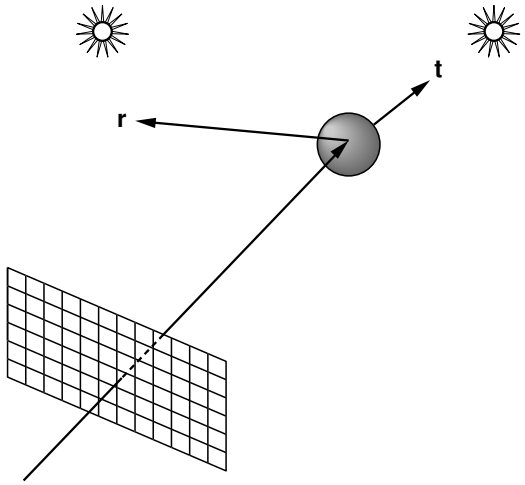
Reflection

- Must follow shadow rays off reflecting or transmitting surfaces
- Process is recursive



24

Reflection and Transmission



25

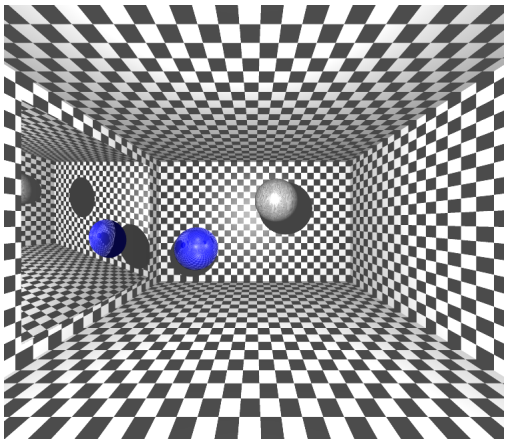
Diffuse Surfaces

- Theoretically the scattering at each point of intersection generates an infinite number of new rays that should be traced
- In practice, we only trace the transmitted and reflected rays but use the Phong model to compute shade at point of intersection
- Radiosity works best for perfectly diffuse (Lambertian) surfaces

26

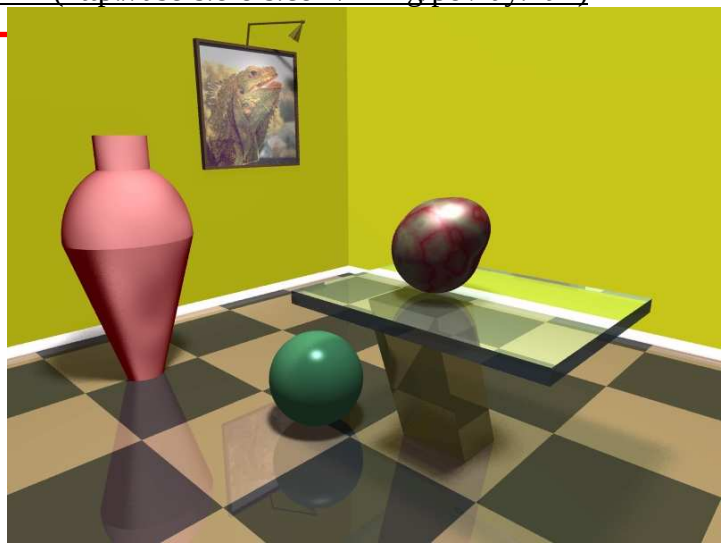
Modelo de Iluminação Global

Modelo Global



27

Figura gerada por Neal Ziring's usando o POV-RAY (http://users.erols.com/ziring/povray.htm)



28

Ray Tracing

- Para cada pixel da imagem:
  - Determina qual é a superfície visível nesse ponto, e qual a cor e intensidade desta superfície
- Como?
  - calcular as intersecções
  - Aplicar modelo de iluminação local no ponto visível
  - Rastrear o raio de volta às suas origens (+ de uma): disparar novos raios e repetir o processo para cada um

29

Cálculo de Intersecções

- Intersecções raio-objeto podem representar até 95% do tempo de processamento de um *ray tracer*
  - Tempo procurando interseções com objetos que sequer estão no caminho do raio
  - envoltórios (*bounding volume*): agrupar grupos de objetos adjacentes em um volume envoltório (caixa ou esfera) => teste inicial com o envoltório

30

Cálculo de Intersecções

- Métodos de sub-divisão do espaço: cena incluída dentro de um cubo
- Cubo é sucessivamente sub-dividido, até que cada sub-região (célula) contenha no máximo um número pré-definido de superfícies (uma ou duas)
- Sub-divisão pode ser armazenada em uma *octree*... Subdivisão adaptativa (apenas as regiões que contém objetos) => coerência espacial

31

Octrees (decomposição espacial)

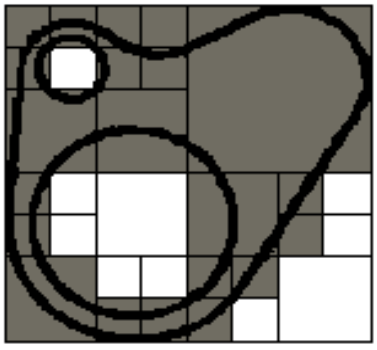
"*Dividir para Conquistar*": explora coerência espacial

Quadtree: 4 quadrantes

- **Vazio**
- **Parcialmente Cheio**
- **Cheio**

Octree: 8 octantes

Representação por árvores hierárquicas



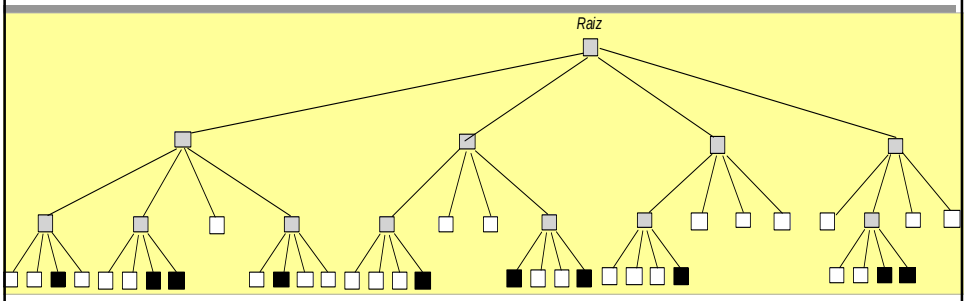
32

Quadtrees (decomposição espacial)

- Indexação da região: 0 a 3 no sentido horário, de cima para baixo
- Cada nó da quadtree tem 4 campos, associados a cada quadrante da região
  - Quadrante homogêneo (pixels da mesma cor): flag indicativo setado
  - quadrante heterogêneo: subdividido em 4, flag setado, campo correspondente armazena o ponteiro para o nó filho

33

Quadrtree



34

Octrees (decomposição espacial)

- Regiões do espaço 3D divididas em cubos
- Estrutura volumétrica, com informação sobre o interior: muito usada em aplicações que requerem a exibição de seções do objeto (por ex., medicina)
- Indexação da região: 0 a 7, cima para baixo, frente para trás
- Um octante heterogêneo é subdividido em 8, e cada nó da árvore contém 8 campos

35

Octrees (decomposição espacial)

- *Voxels*: elementos individuais do espaço 3D
- Teste de homogeneidade verifica 'material' contido no voxel
- Procedimento para gerar *octree*: voxels em cada *octante* são testados (homogêneo?), e subdivisões são efetuadas até que a região do espaço contenha apenas voxels do mesmo tipo (homogêneos)

36

Building a Ray Tracer

- Best expressed recursively
- Can remove recursion later
- Image based approach
  - For each ray …….
- Find intersection with closest surface
  - Need whole object database available
  - Complexity of calculation limits object types
- Compute lighting at surface
- Trace reflected and transmitted rays

37

When to stop

- Some light will be absorbed at each intersection
  - Track amount left
- Ignore rays that go off to infinity
  - Put large sphere around problem
- Count steps

38

Recursive Ray Tracer

```
color c = trace(point p, vector d,
 int step)
{
  color local, reflected,
    transmitted;
  point q;
  normal n;
  if(step > max)
    return(background_color);
```

39

Recursive Ray Tracer

```
q = intersect(p, d, status);
if(status == light_source)
 return(light_source_color);
if(status == no_intersection)
 return(background_color);

n = normal(q);
r = reflect(q, n);
t = transmit(q,n);
```

40

Recursive Ray Tracer

```
local = phong(q, n, r);
reflected = trace(q, r, step+1);
transmitted = trace(q,t, step+1);

return(local+reflected+
transmitted);
```

41

Computing Intersections

- Implicit Objects
  - Quadrics
- Planes
- Polyhedra
- Parametric Surfaces

42

Implicit Surfaces

Ray from $\mathbf{p}_0$ in direction $\mathbf{d}$

$$\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$$

General implicit surface

$$f(\mathbf{p}) = 0$$

Solve scalar equation

$$f(\mathbf{p}(t)) = 0$$

General case requires numerical methods

43

Quadrics

General quadric can be written as

$$\mathbf{p}^T\mathbf{A}\mathbf{p} + b^T\mathbf{p} + c = 0$$

Substitute equation of ray

$$\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$$

to get quadratic equation

44

Sphere

$(\mathbf{p} - \mathbf{p}_c) \bullet (\mathbf{p} - \mathbf{p}_c) - r^2 = 0$

$\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$

$\mathbf{p}_0 \bullet \mathbf{p}_0\, t^2 + 2\,\mathbf{p}_0 \bullet (\mathbf{d} - \mathbf{p}_0)\, t + (\mathbf{d} - \mathbf{p}_0) \bullet (\mathbf{d} - \mathbf{p}_0) - r^2 = 0$

45

Planes

$\mathbf{p} \bullet \mathbf{n} + c = 0$

$\mathbf{p}(t) = \mathbf{p}_0 + t\,\mathbf{d}$

$t = -(\mathbf{p}_0 \bullet \mathbf{n} + c) / \mathbf{d} \bullet \mathbf{n}$

46

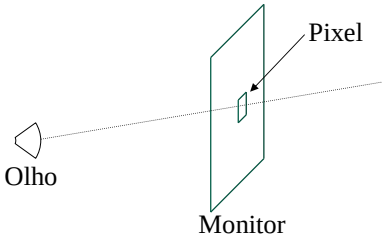
Polyhedra

- Generally we want to intersect with closed objects such as polygons and polyhedra rather than planes
- Hence we have to worry about inside/outside testing
- For convex objects such as polyhedra there are some fast tests

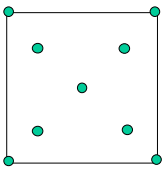
47

Ray Tracing: Aliasing

Raios versus Pixels



Supersampling



Visão Frontal de um Pixel

48

Supersampling

Sem *supersampling*

Com *supersampling*

Aliasing

- Re-amostragem
- amostragem adaptativa: traça múltiplos raios por pixel, mas os raios são espaçados de forma desigual em diferentes regiões da área do pixel
  - por exemplo, mais raios em regiões próximas às arestas dos objetos.
- outras técnicas mais sofisticadas, como *ray tracing* distribuído...

Bibliografia

- GLASSNER, Andrew S. (Edited) - An Introduction to Ray Tracing, Academic Press, 1989
- BAKER, M. Pauline e HEARN, Donald - Computer Graphics, Prentice Hall Ed, 1997
- FOLEY, James D., VAN DAM, Andries, FEINER, Steven e HUGHES, John - Computer Graphics: Principles and Practice - Addison-Wesley Ed., 1990
- Angel, E. Interactive Computer Graphics, Addison-Wesley, 2005
- Curso de CG da ACM SIGGRAPH: www.education.siggraph.org/materials/HyperGraph/hypergraph.htm