Mestrado em
Engenharia Informática

Ray Tracing Distribuído

Visualização e
Iluminação

Luís Paulo Peixoto dos Santos

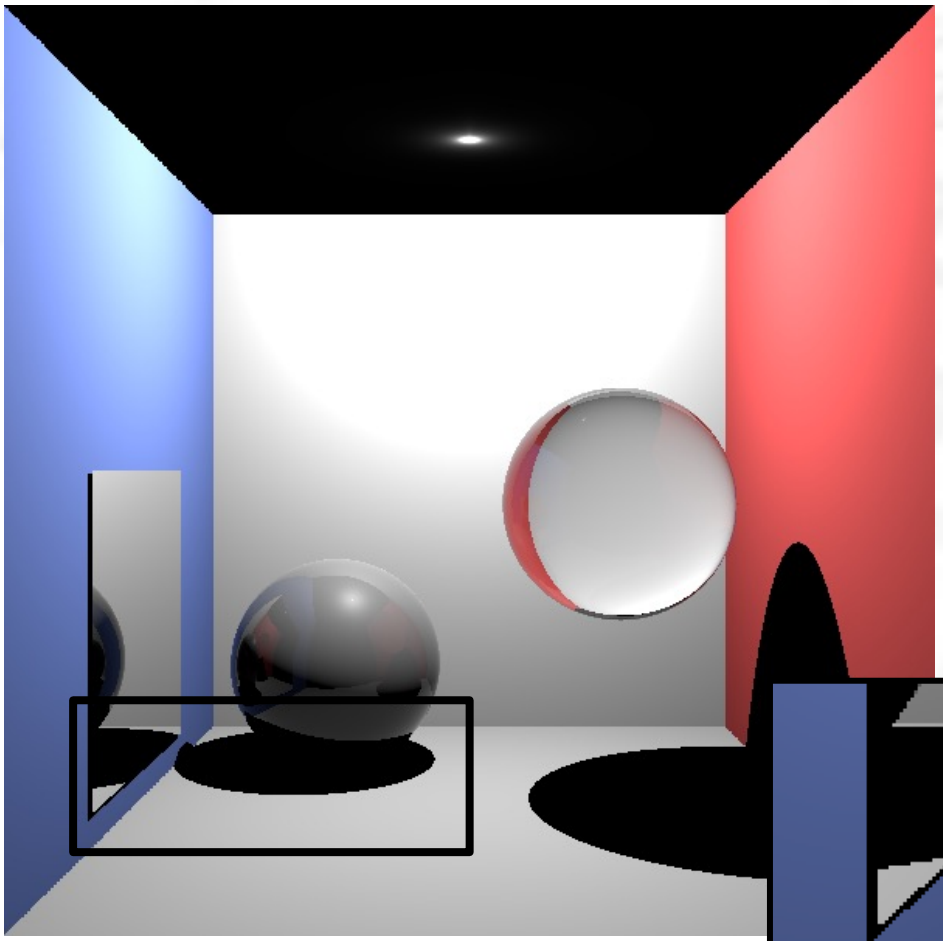
Ray Tracing clássico: artefactos e *aliasing*

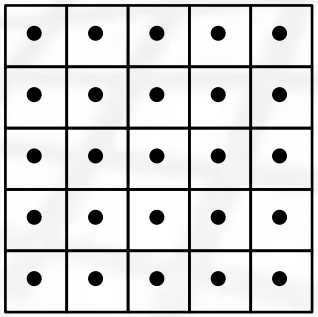
- O *rendering* implica estimar vários sinais ao longo da cena. Exemplos:
  - Estimar a radiância incidente em cada *pixel* do plano da imagem;
  - Estimar a radiância incidente directa (das fontes de luz) em pontos do mundo 3D;
  - Entre muitos outros

- Os algoritmos de iluminação global baseados em *ray tracing* estimam estes sinais **amostrando-os** em pontos bem definidos. Exemplos:
  - Disparando um raio primário através do centro da área de cada pixel;
  - Disparando um *shadow ray* através do centro de cada fonte de luz

- O carácter determinístico, uniforme e regular desta amostragem resulta em fenómenos de *aliasing* e artefactos estruturados que não têm plausabilidade física.

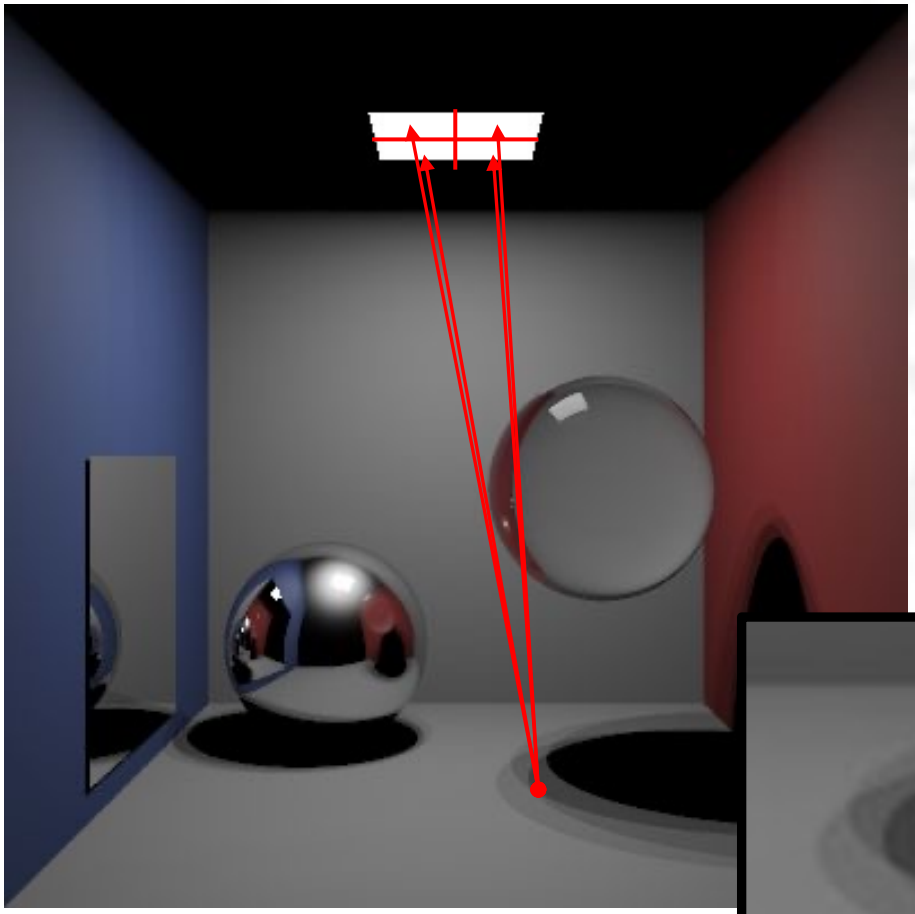
Ray Tracing clássico: artefactos e *aliasing*



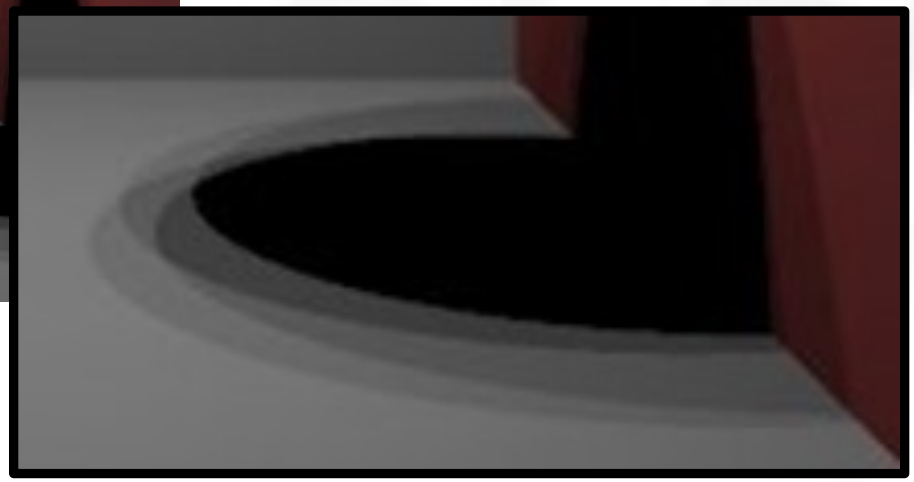


A amostragem uniforme do plano da imagem resulta em artefactos estruturados

Ray Tracing clássico: artefactos e *aliasing*



A amostragem uniforme e regular das fontes de luz resulta em sombras com contornos muito definidos, sem penumbra: artefactos estruturados



Ray Tracing clássico: artefactos e *aliasing*

A amostragem uniforme do plano da imagem resulta em *aliasing*:

- O padrão axadrezado é constante
- Quanto mais distante está da câmara, maior a frequência espacial das variações
- Quando essa frequência ultrapassa metade da frequência de amostragem do plano da imagem as variações preto/branco não são bem capturadas
- Tal acontece quando 1 ciclo preto/branco projecta em menos do que 2 pixéis: **Teorema de Nyquist**: frequência de amostragem >= 2 * frequência do sinal
- As altas frequências mascaram-se como baixas frequências: ***aliasing***

Teoria da amostragem

- Selecção de pontos de amostragem de funções contínuas e utilização das amostras para construir novas funções semelhantes à original

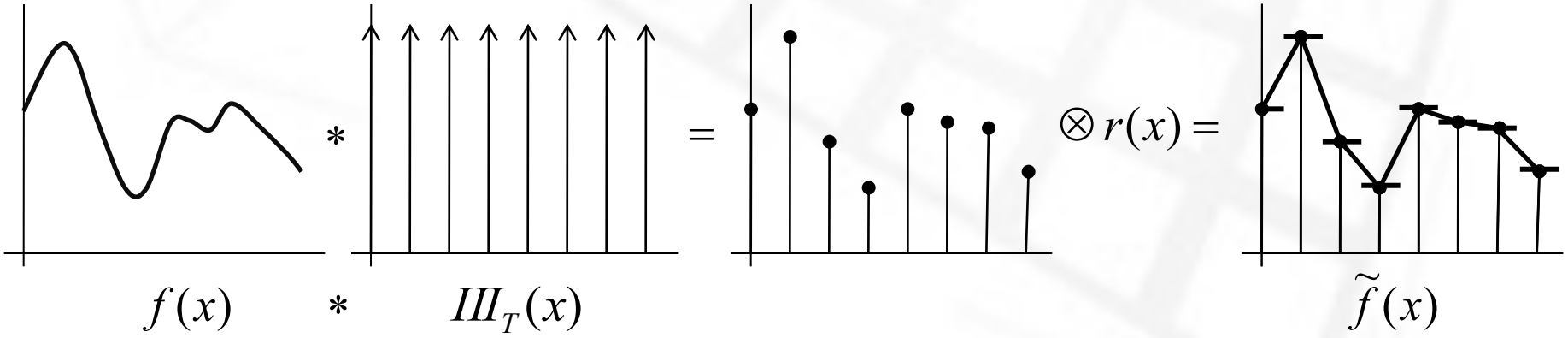
$f(x)$ - função contínua

$x' \in D_f$ - Posição da amostra

$f(x')$- amostra

$r(x)$ - filtro reconstrução

$\tilde{f}(x)$ - função reconstruída

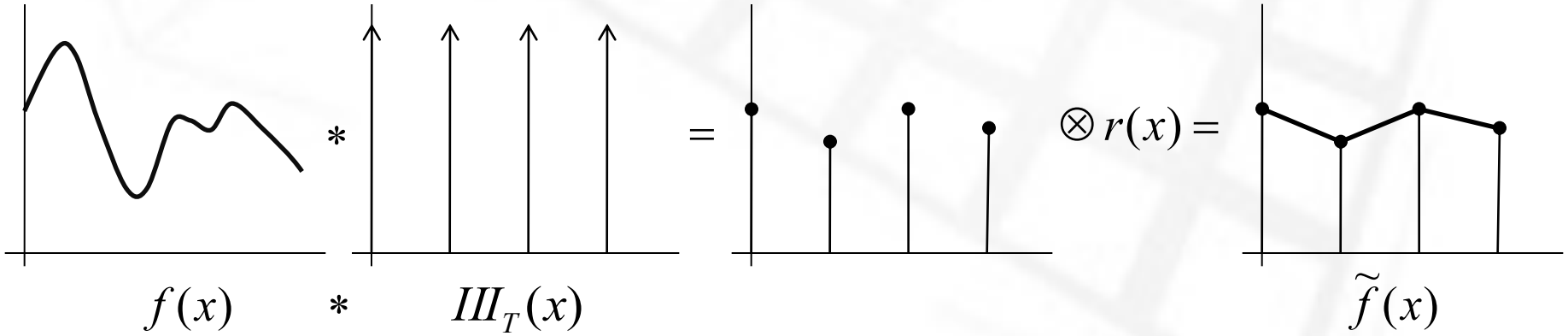


Limite de Nyquist

Limite de Nyquist

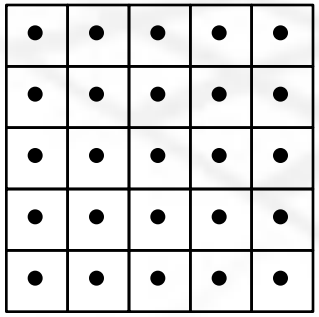
Se a frequência de amostragem, f_s, é maior ou igual ao dobro da maior frequência presente no sinal (função contínua), f_{max}, então é possível reconstruir perfeitamente o sinal original.

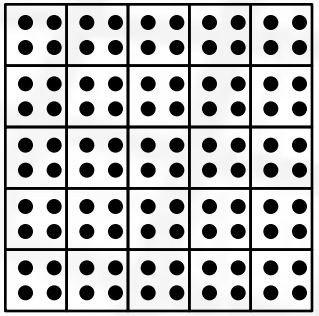
$$\text{se } f_s \geq 2 * f_{\max} \text{ então } \widetilde{f}(x) = f(x)$$

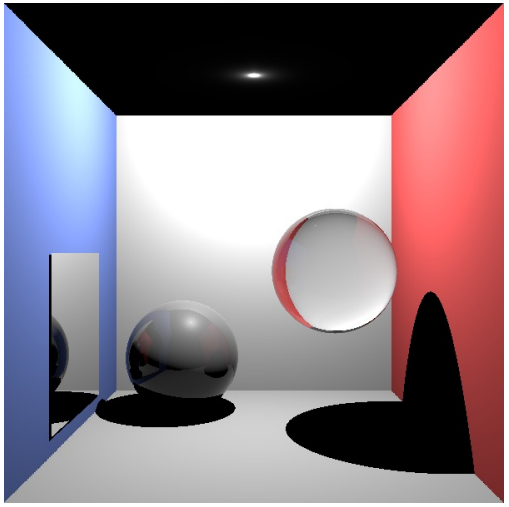


Antialiasing: Sobreamostragem

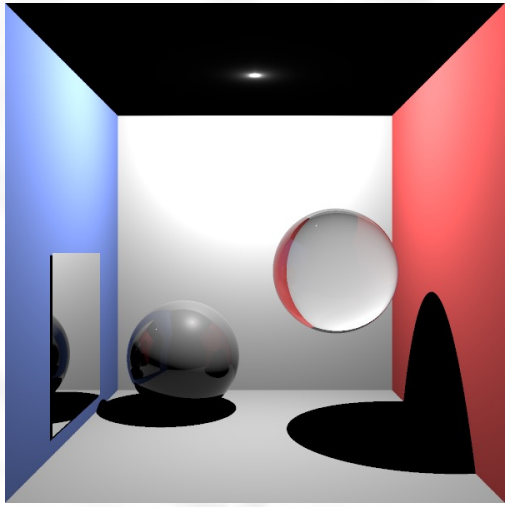
- A frequência de amostragem pode aumentar disparando mais do que um raio primário por pixel







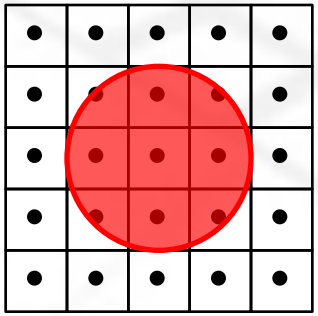
0.5 seg

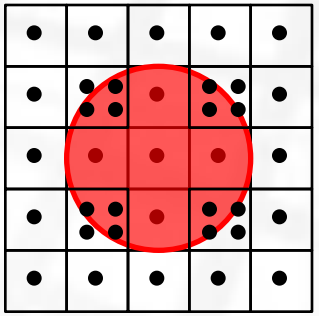


1.3 seg

Antialiasing: Sobreamostragem adaptativa

- Para evitar o aumento linear do tempo de execução aumenta-se a frequência de amostragem apenas quando a diferença entre 2 amostras vizinhas ultrapassa um determinado limite





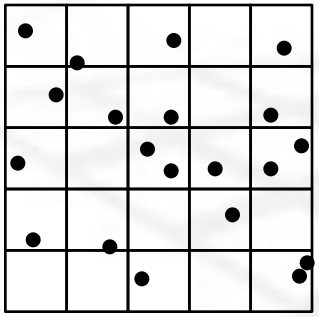
- Uma vez que uma imagem com descontinuidades não é limitada em banda, o aumento da frequência de amostragem nunca elimina o fenómeno de *aliasing*, apenas reduz o erro
- Esta redução pode resultar num erro abaixo do limite percepcionável pelo Sistema Visual Humano

Antialiasing: Amostragem não uniforme

- O impacto visual de *aliasing* pode ser reduzido variando o espaçamento entre amostras
  - Por exemplo seleccionando o ponto de amostragem estocasticamente
- O sinal reconstruído continua a ser incorrecto, mas é percepcionado como ruído e não como *aliasing*
- O ruído é a variância introduzida pelas variáveis aleatórias usadas no processo de selecção das amostras
- O sistema visual humano é mais tolerante a ruído aleatório do que a *aliasing* estruturado

Antialiasing: Amostragem não regular

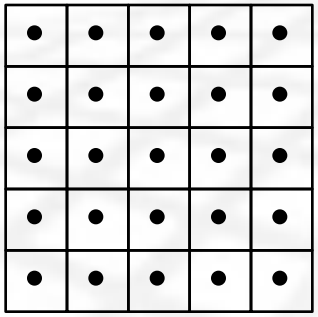
Pseudo-aleatório



$$x = \xi_1 * W$$

$$y = \xi_2 * H$$

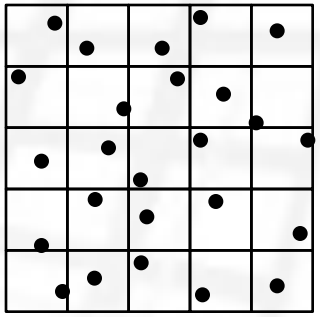
Estratificado



$$x = x_p$$

$$y = y_p$$

Jittered

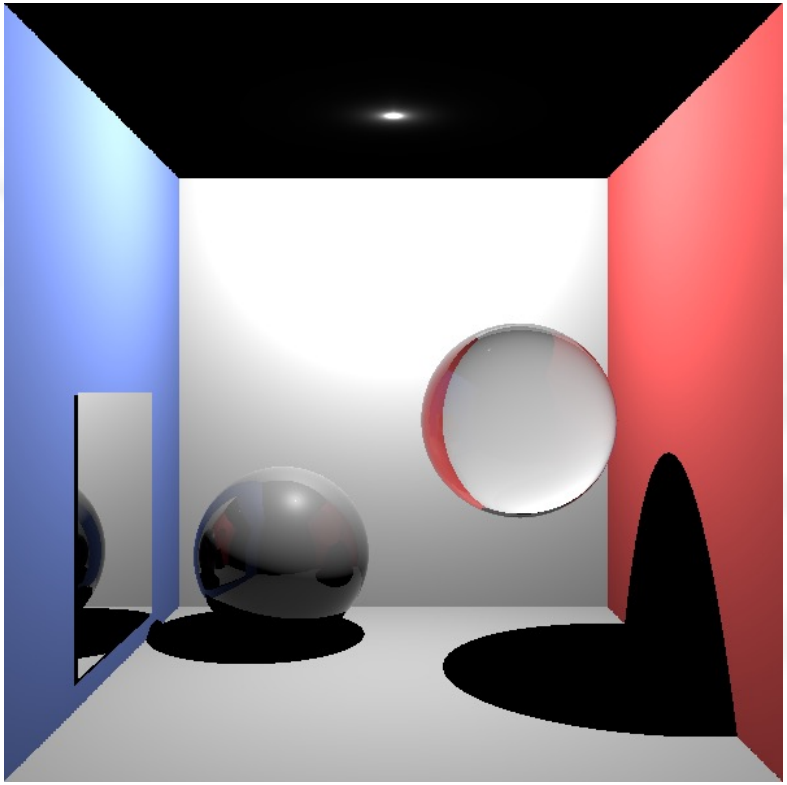


$$x = x_p - 0.5 + \xi_1$$

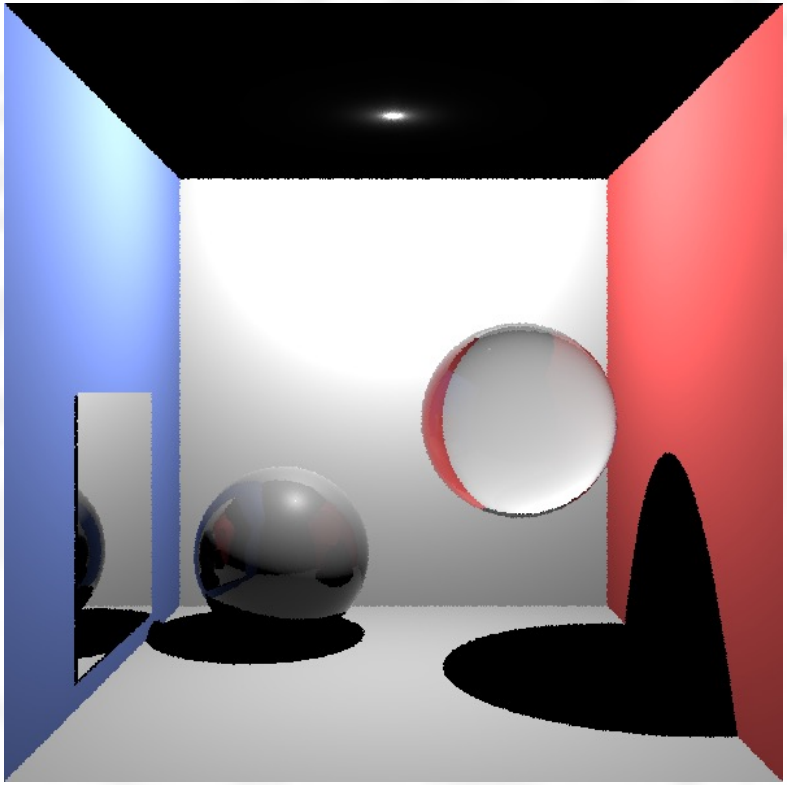
$$y = y_p - 0.5 + \xi_2$$

- ξ_1 e ξ_2 são variáveis aleatórias com distribuição uniforme no domínio [0 .. 1[

Antialiasing: Amostragem não regular



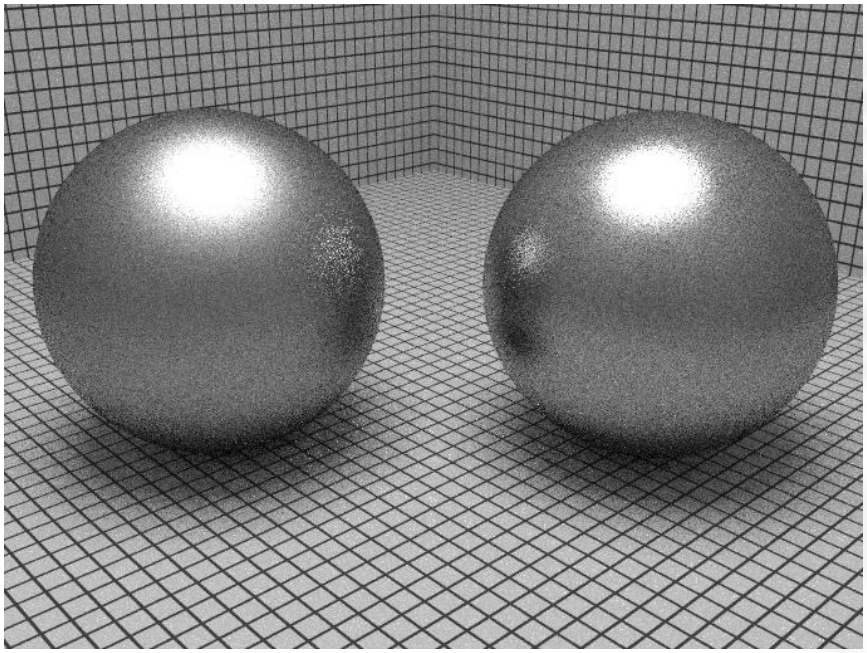
Uniform (1spp) – 0.5 seg

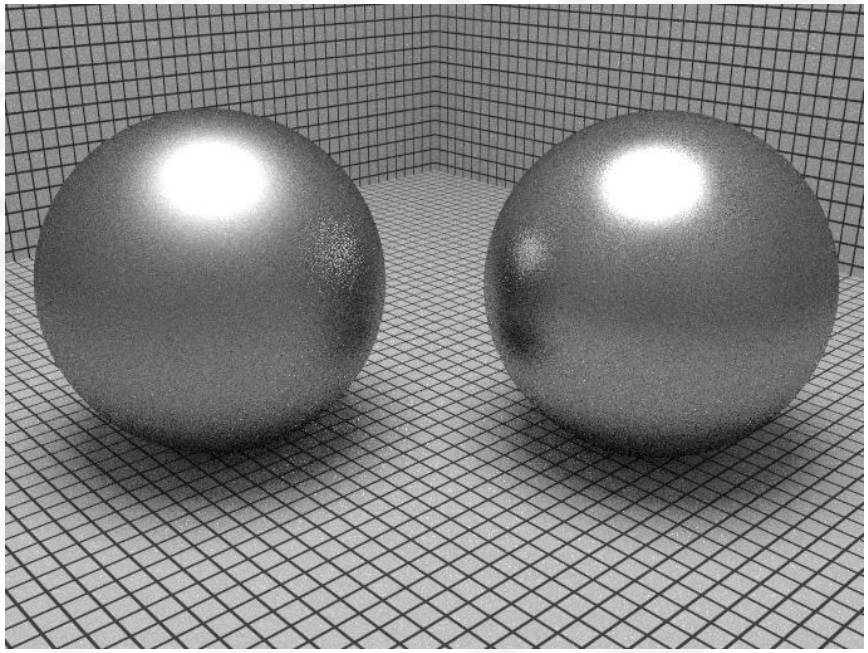


Jittered (1spp) – 1.6 seg

- O *aliasing* é removido introduzindo *jittering* e mantendo 1 amostra por pixel (spp: *samples per pixel*)
- É introduzido ruído no resultado; tempo de execução mantém-se constante

Antialiasing: Amostragem não uniforme





Imagens obtidas com o mesmo número de raios e diferentes distribuições
[Pharr & Humphreys, 2004]

- A utilização de distribuições de amostragem apropriadas permite obter melhores resultados (menor variância) com o mesmo número de raios

Ray tracing distribuído: introdução

- O domínio do sinal é amostrado com ***N*** amostras, sendo estas **amostras escolhidas de forma estocática**

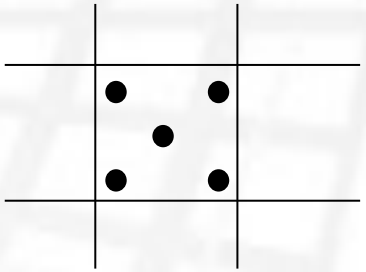
- Cada amostra ω_i é escolhida com probabilidade $p(\omega_i)$
- Se a distribuição da probabilidade é uniforme então $p(\omega_i)$ igual ao inverso da extensão do domínio:
  - Exemplo: se o domínio é a área de um pixel A_p, então a probabilidade $p(\omega_i)$ de seleccionar um ponto ω_i nessa área é igual para todos os pontos: $p(\omega_i) = 1 / A_p$

- A estimativa do valor do sinal é dada por **integração de Monte Carlo:**

$$I \approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(w_i)}{p(w_i)}$$

Ray tracing distribuído: *antialiasing*

- Consideremos um pixel com área A_p
- O ray tracing clássico amostra este pixel com n amostras distribuídas de forma regular



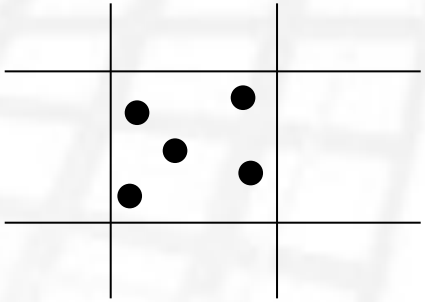
- e calcula o seu valor final como

$$I_p = \frac{A_p}{N} \sum_{i=0}^{N-1} L(\omega_i)$$

- Esta solução corresponde a sobreamostragem e não elimina o *aliasing*

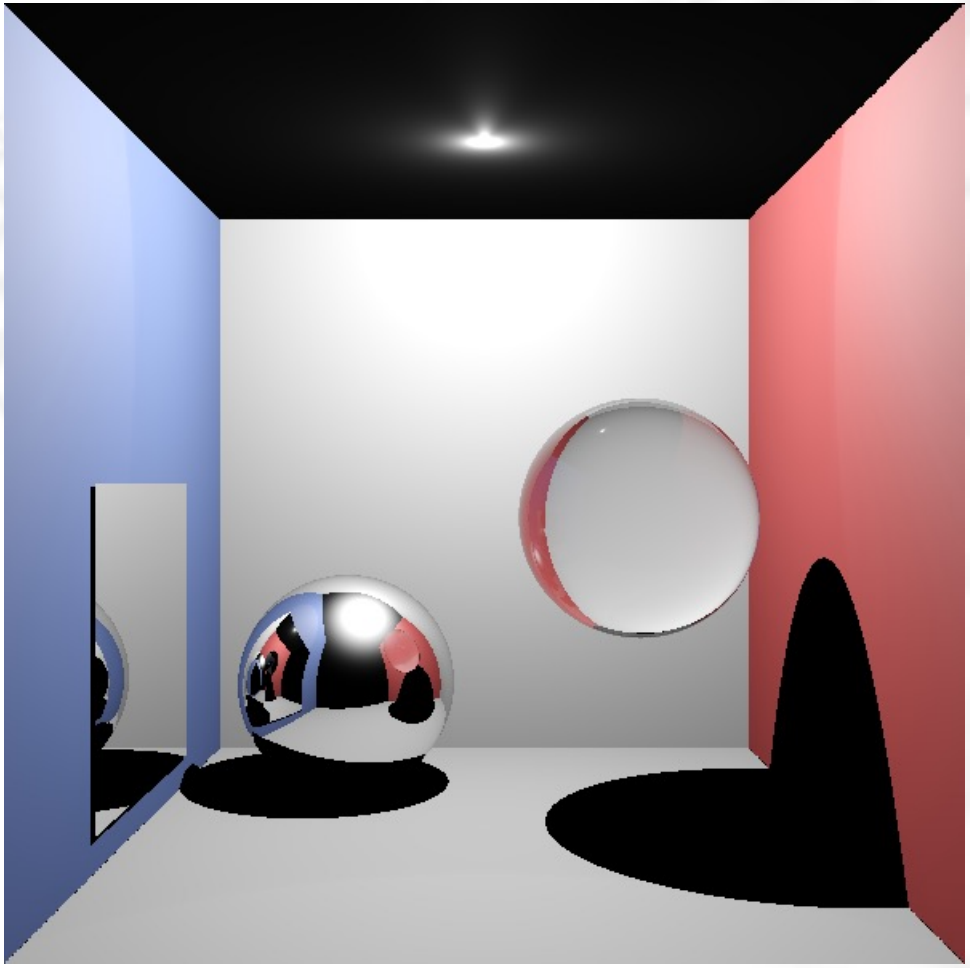
Ray tracing distribuído: *antialiasing*

- Consideremos um pixel com área A_p
- O ray tracing distribuído amostra este pixel com n amostras seleccionadas com probabilidade $p(\omega_i)$



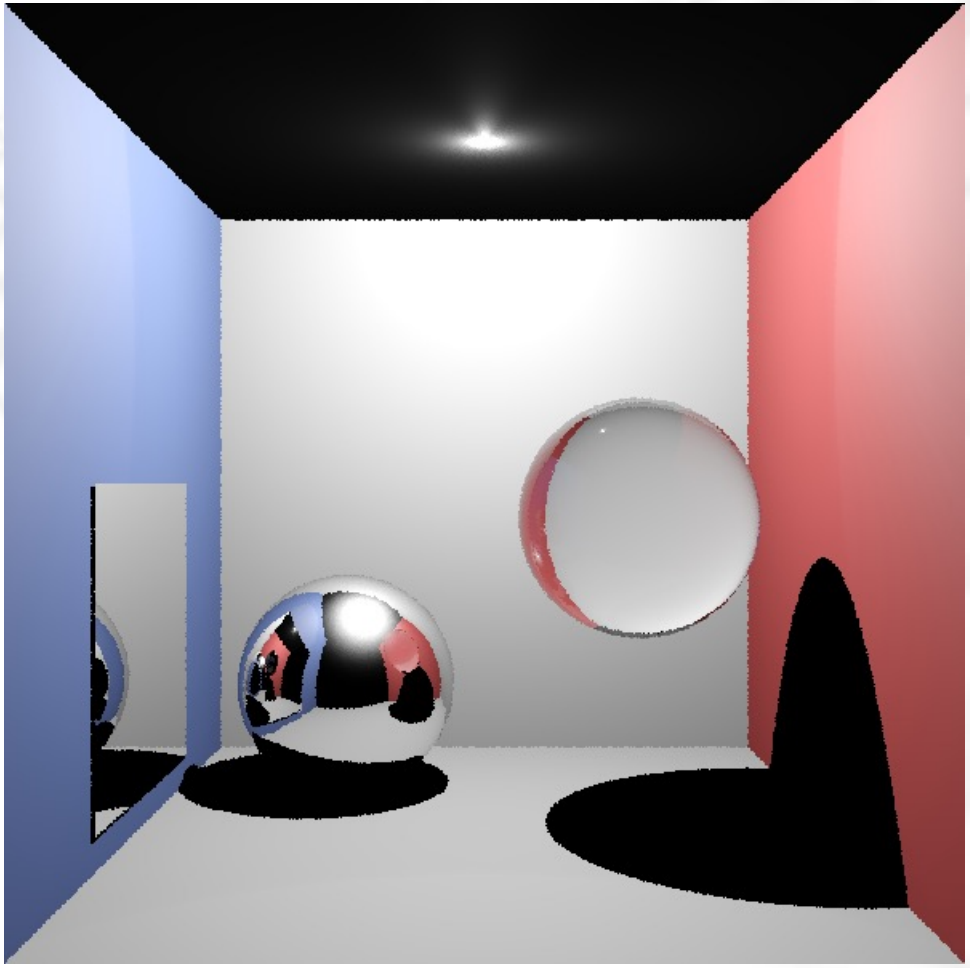
- e calcula o seu valor final como $$I_p \approx \frac{1}{N}\sum_{i=0}^{N-1}\frac{L(\omega_i)}{p(\omega_i)}$$
- O *aliasing* é substituído por ruído aleatório, que o Sistema Visual Humano aceita melhor

Ray tracing distribuído: *antialiasing*



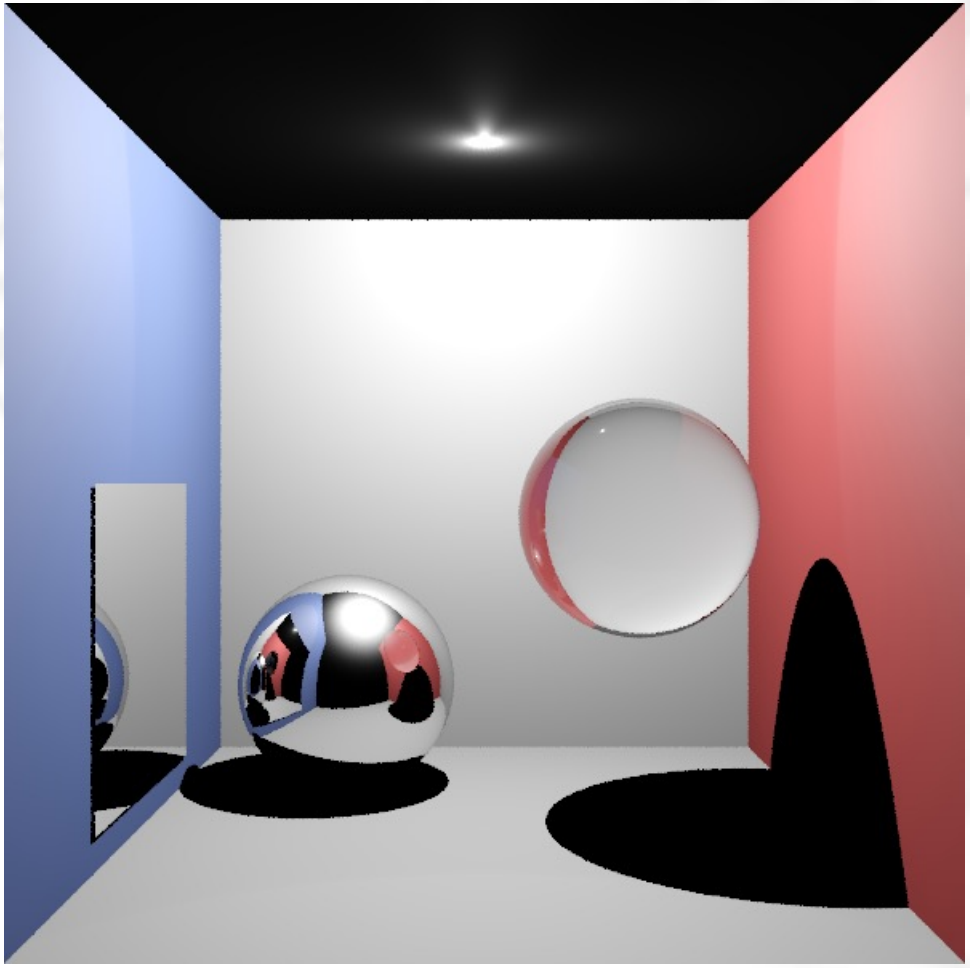
spp=1; regular

Ray tracing distribuído: *antialiasing*



spp=1; jitt

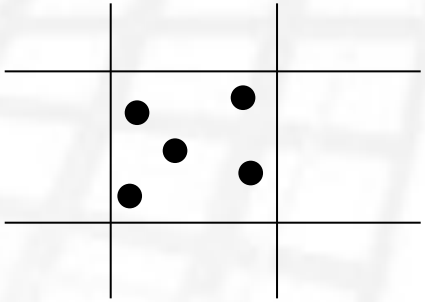
Ray tracing distribuído: *antialiasing*



spp=3; jitt

Ray tracing distribuído: *antialiasing*

- Consideremos um pixel com área A_p
- A probabilidade $p(w_i)$ com que um ponto é escolhido na área do pixel pode ser qualquer desde que o seu integral sobre o domínio (área) seja igual a 1
- Probabilidade uniforme implica $p(\omega_i) = 1/A_p$



$$I_p \approx \frac{1}{N}\sum_{i=0}^{N-1}\frac{L(\omega_i)}{p(\omega_i)} = \frac{1}{N}\sum_{i=0}^{N-1}\frac{L(\omega_i)}{1/A_p} = \frac{1}{N}\sum_{i=0}^{N-1}A_p * L(\omega_i) = \frac{A_p}{N}\sum_{i=0}^{N-1}L(\omega_i)$$

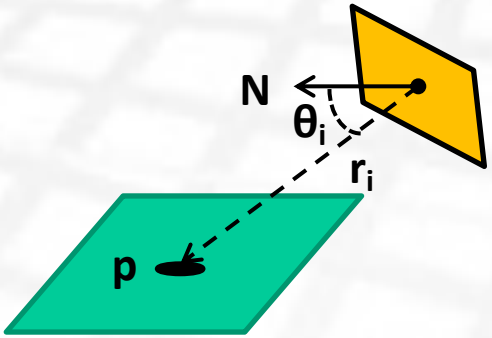
A probabilidade pode ser não uniforme, privilegiando determinadas regiões do domínio

RT Clássico: Fontes de luz e *soft shadows*

- A contribuição de cada fonte de luz para a radiância incidente no ponto *p* deve ser o produto da radiância emitida pelo ângulo sólido, ω_i , que a fonte de luz subentende em *p*.

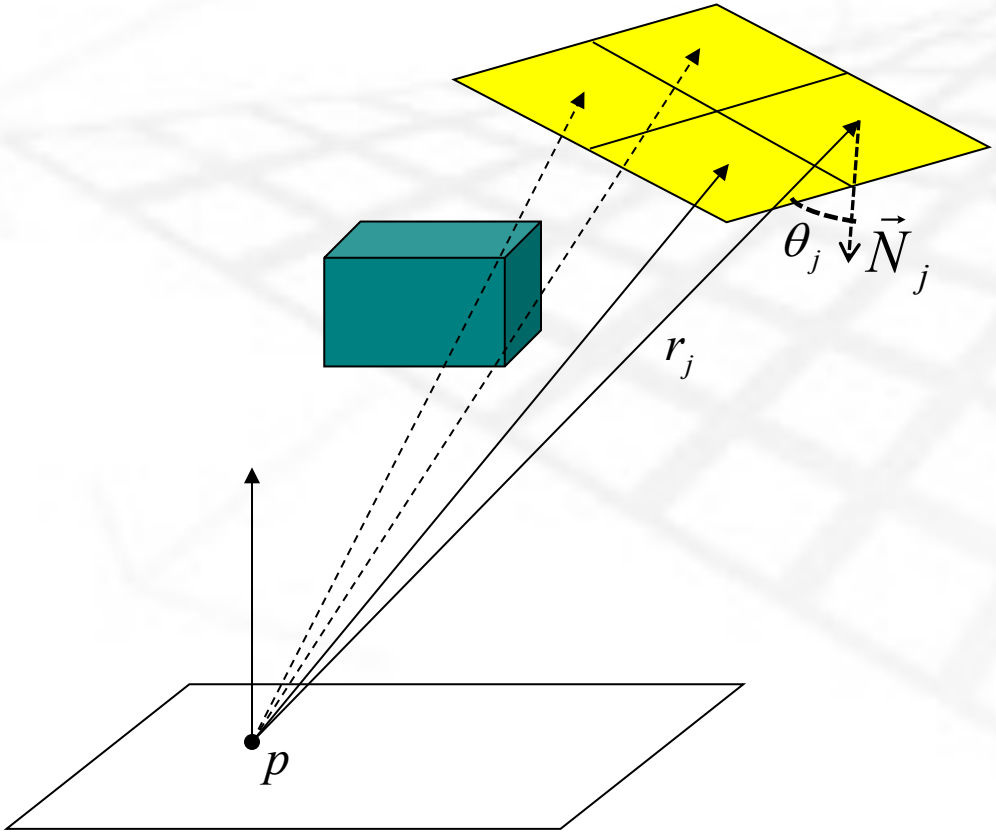
$$\omega_i = \frac{A_i \cos\theta_i}{r_i^2}$$

$$L(p \leftarrow \omega_i) = L_i * \omega_i = L_i * \frac{A_i \cos\theta_i}{r_i^2}$$



- No ray tracing clássico a direcção dos *shadow rays* é determinística e regular, correspondendo sempre ao mesmo ponto na área da fonte de luz.

RT Clássico: Fontes de luz (*hard shadows)*



São escolhidos N pontos na área da fonte de luz

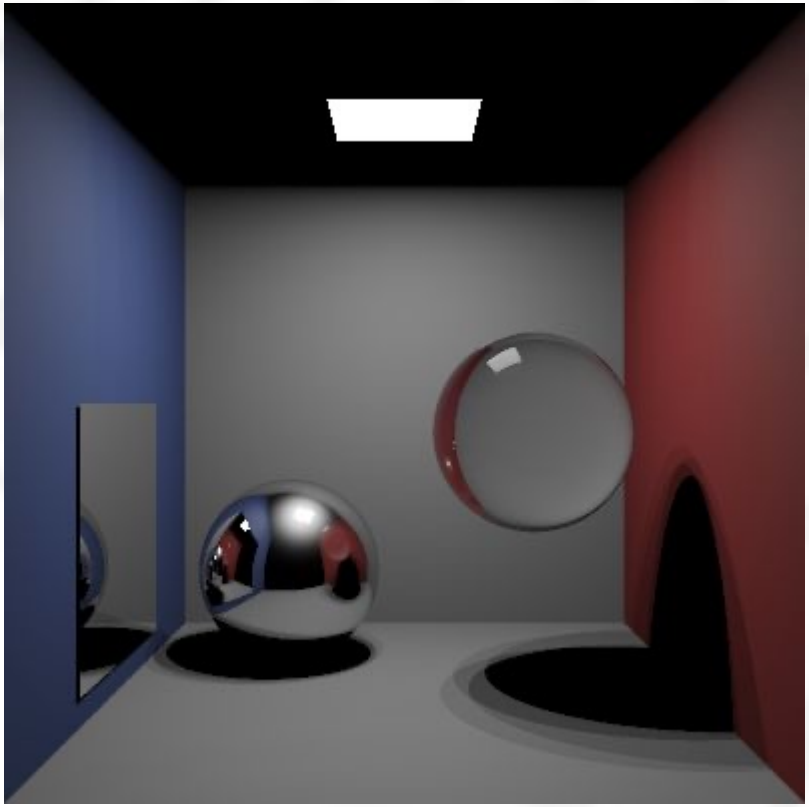
Cada *shadow ray* avalia a visibilidade de p_j a partir de *p*: V(*p*, p_j)

A radiância directa incidente em *p* devido à fonte de luz *i* é dada por:

$$\frac{L_i * A_L}{N} \sum_j V(p, p_j) * \frac{\cos\theta_j}{r_j^2}$$

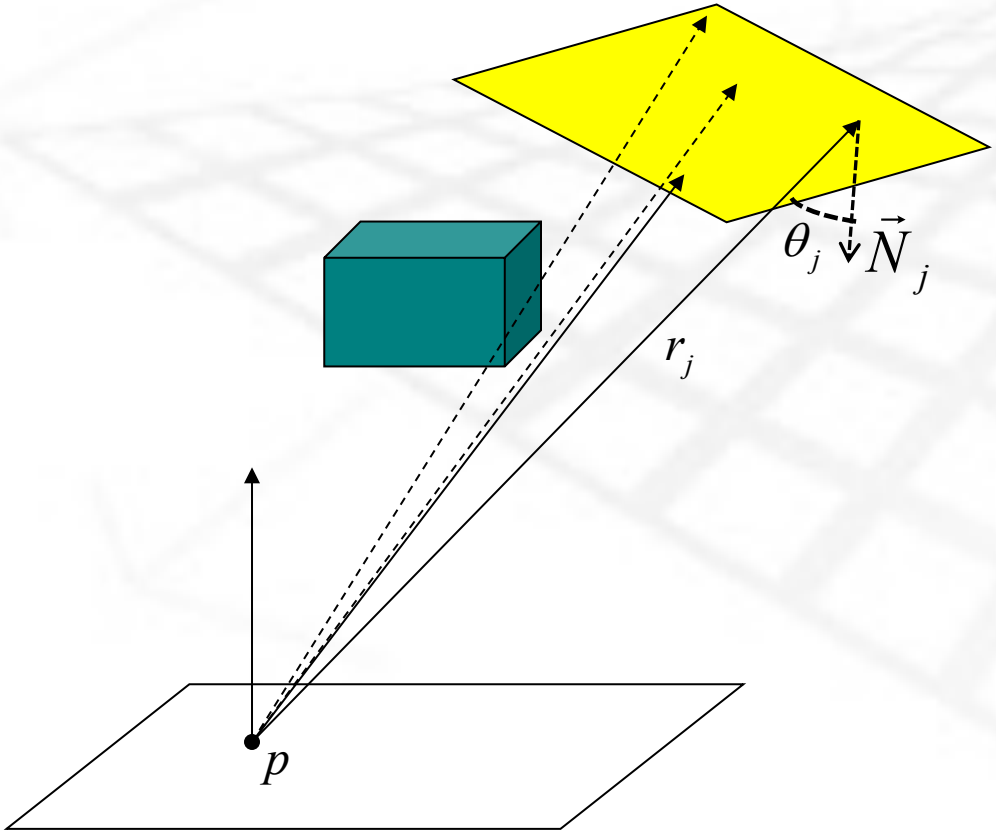
RT Clássico: Fontes de luz (*hard shadows)*

- Esta configuração equivale a ter N pontos de luz



`spp=1; spl=4`

Ray tracing distribuído: *soft shadows*



São escolhidos aleatoriamente N pontos na área da fonte de luz com probabilidade (uniforme)

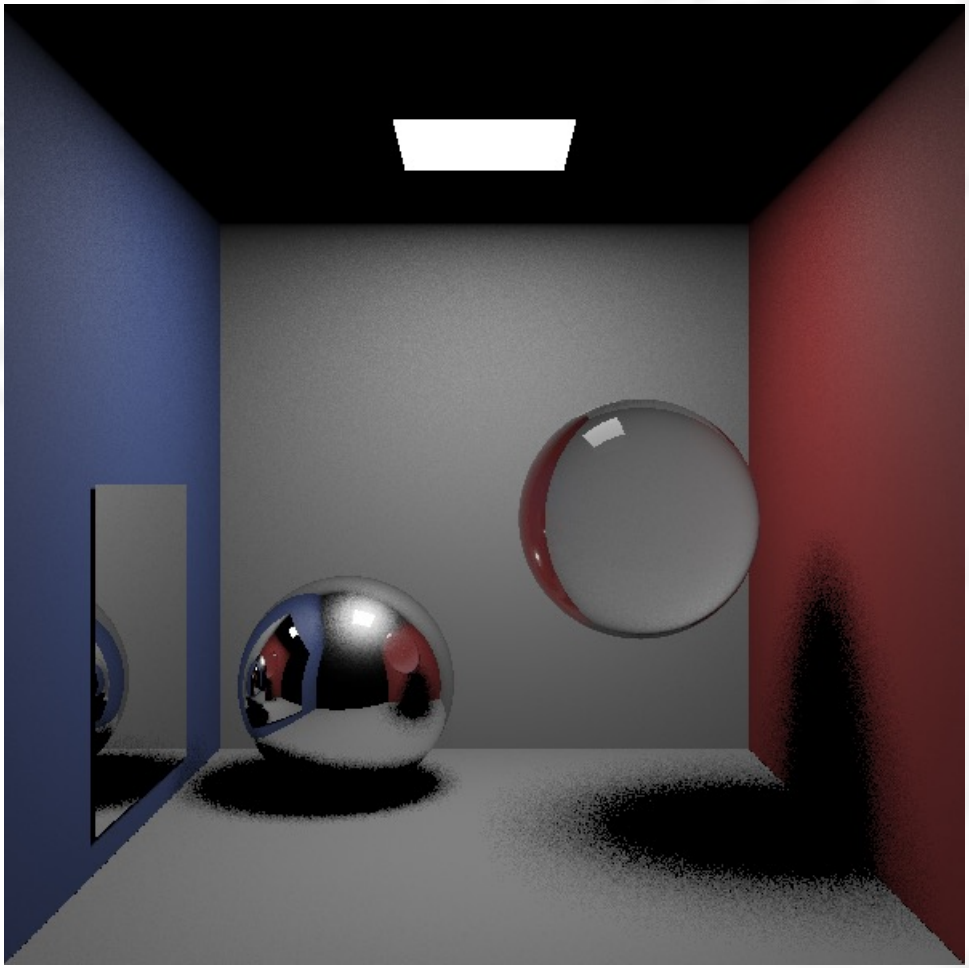
$$p(p_j) = \frac{1}{A_L}$$

Cada *shadow ray* avalia a visibilidade de p_j a partir de *p*: V(*p*, p_j)

A radiância directa incidente em *p* devido à fonte de luz *i* é dada por:

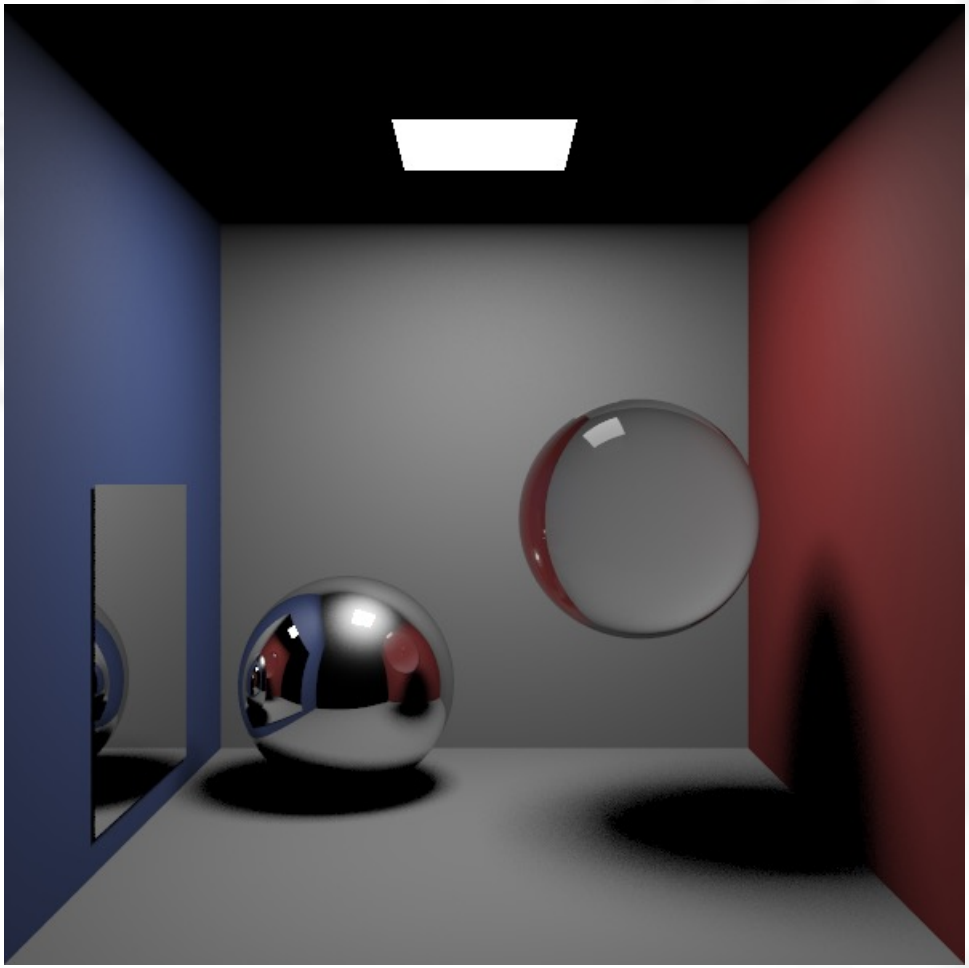
$$\frac{L_i * A_L}{N} \sum_j V(p, p_j) * \frac{\cos\theta_j}{r_j^2}$$

Ray tracing distribuído: *soft shadows*



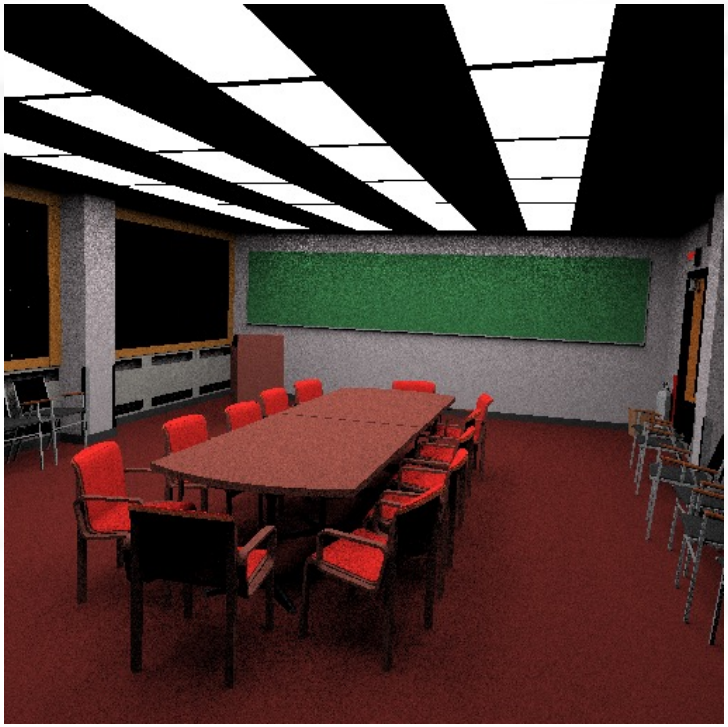
`spp=1; spl=4`

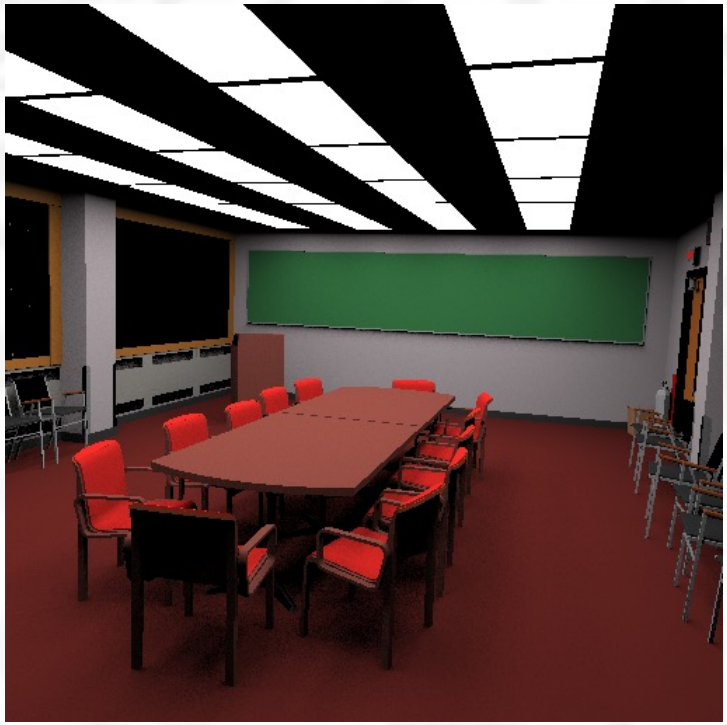
Ray tracing distribuído: *soft shadows*



`spp=64; spl=4`

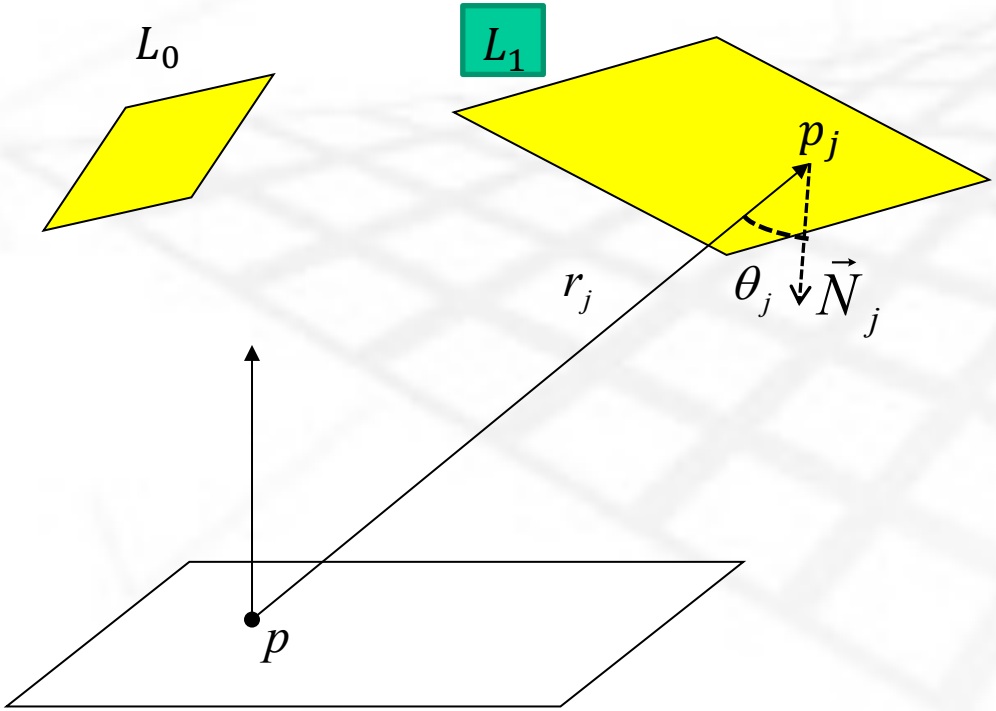
Ray tracing distribuído: *soft shadows*





36 *shadow rays* mas com diferentes pdf's

Ray tracing distribuído: múltiplas fontes de luz



Escolher aleatoriamente uma fonte de luz L_j entre as N_L existentes com probabilidade (uniforme): $p(L_j) = \frac{1}{N_L}$

É escolhido aleatoriamente 1 ponto na área da fonte de luz L_j com probabilidade (uniforme): $p(p_j) = \frac{1}{A_{L_j}}$

Um *shadow ray* avalia a visibilidade de p_j a partir de *p*: V(*p*, p_j)

A radiância directa incidente em *p* devido à fonte de luz *L* é dada por:

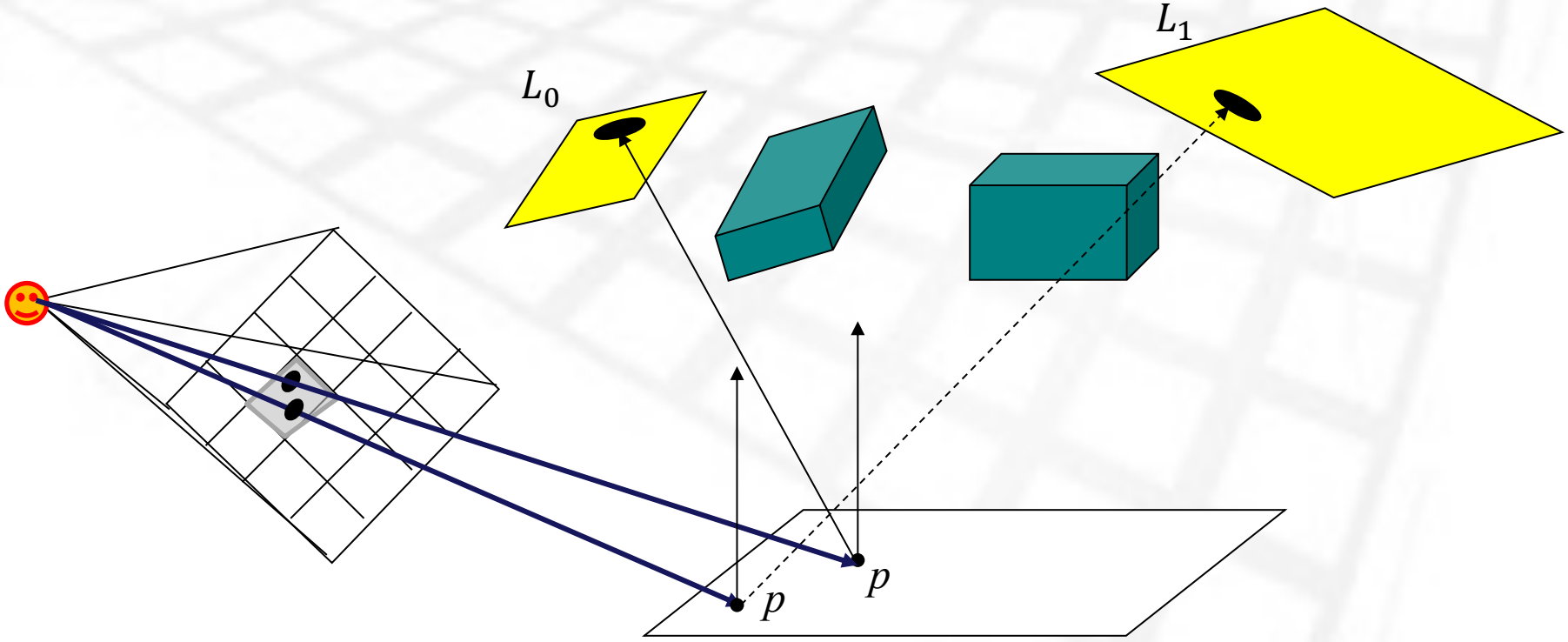
$$\frac{L_j * V(p, p_j) * \cos\theta_j}{\frac{1}{N_L} * \frac{1}{A_{L_j}} * r_j^2}$$

$$= \frac{N_L * A_{L_j} * L_j * V(p, p_j) * \cos\theta_j}{r_j^2}$$

Repetindo o processo N vezes:

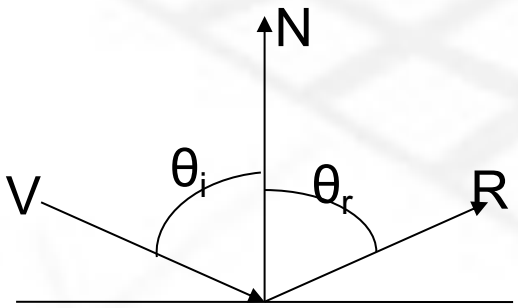
$$\frac{N_L}{N} \sum_j \frac{A_{L_j} * L_j * V(p, p_j) * \cos\theta_j}{r_j^2}$$

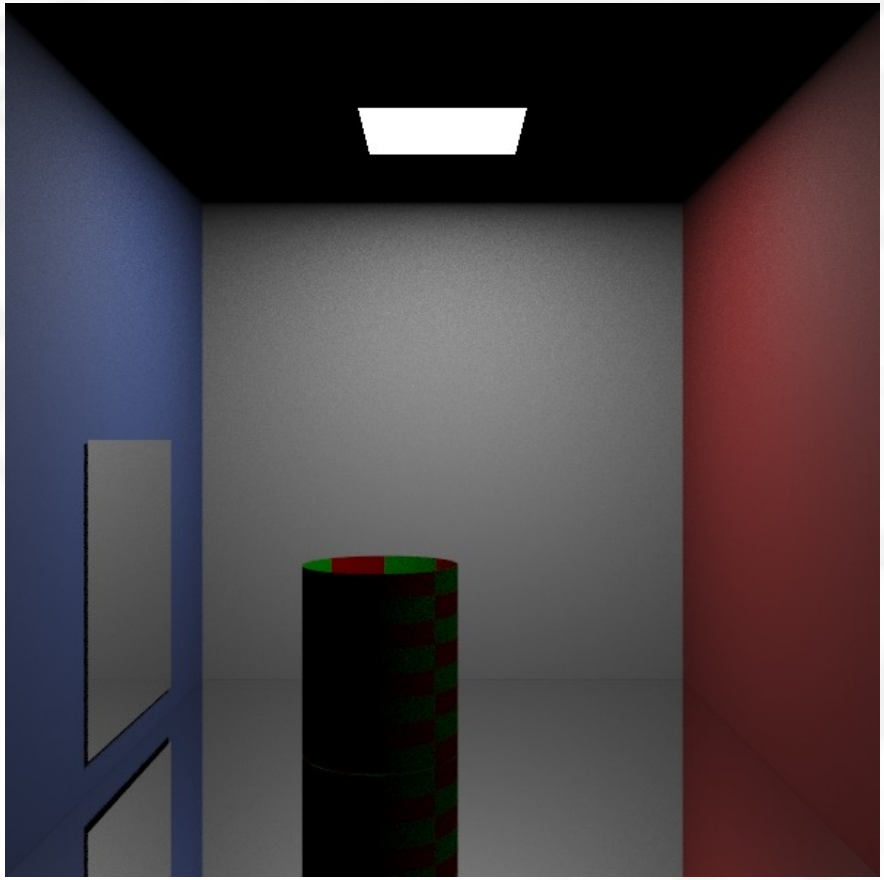
Ray tracing distribuído: múltiplas fontes de luz



Ray tracing distribuído: *glossiness*

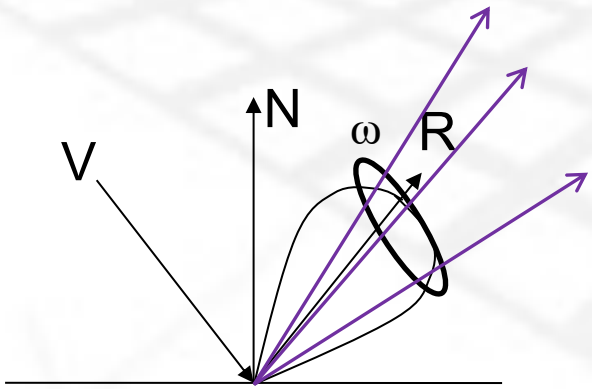
- RT clássico amostra apenas as direcções especulares perfeitas





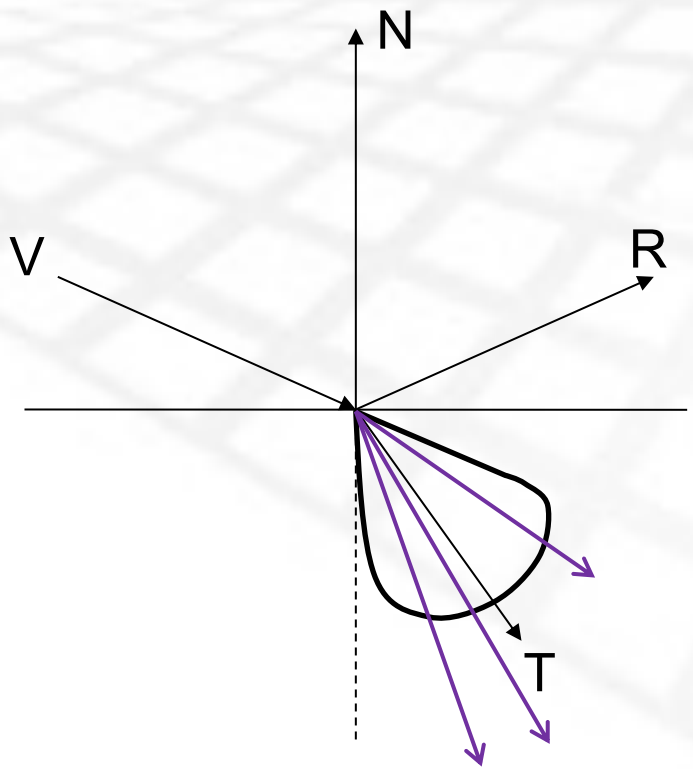
Ray tracing distribuído: *glossiness*

- Os materiais reais exibem *gloss*



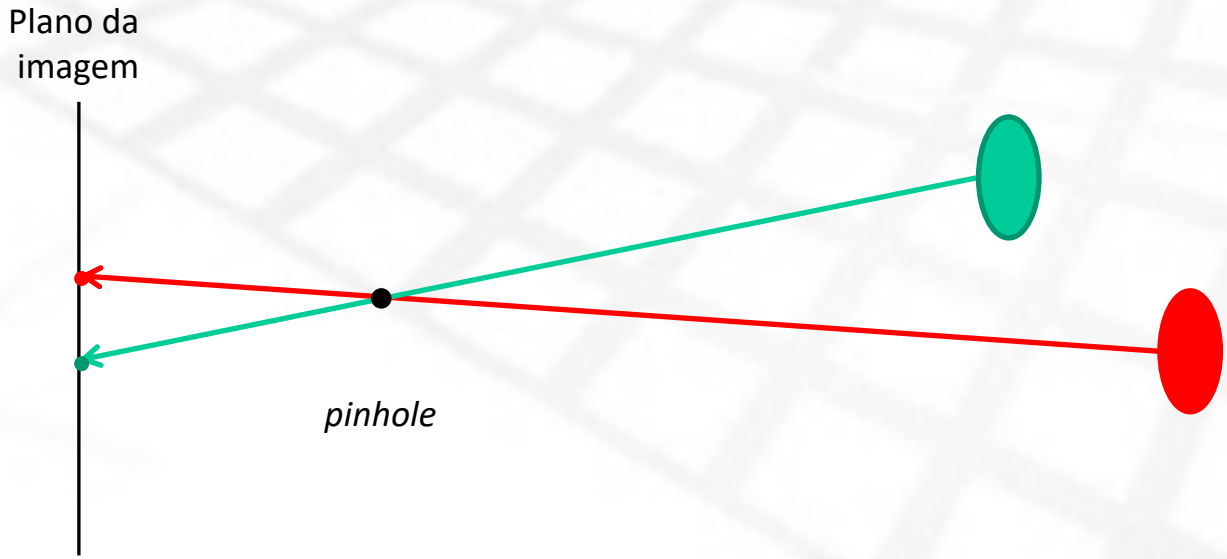
- Múltiplos raios de reflexão são distribuídos no cone (ângulo sólido ω) centrado em R

Ray tracing distribuído: translucência



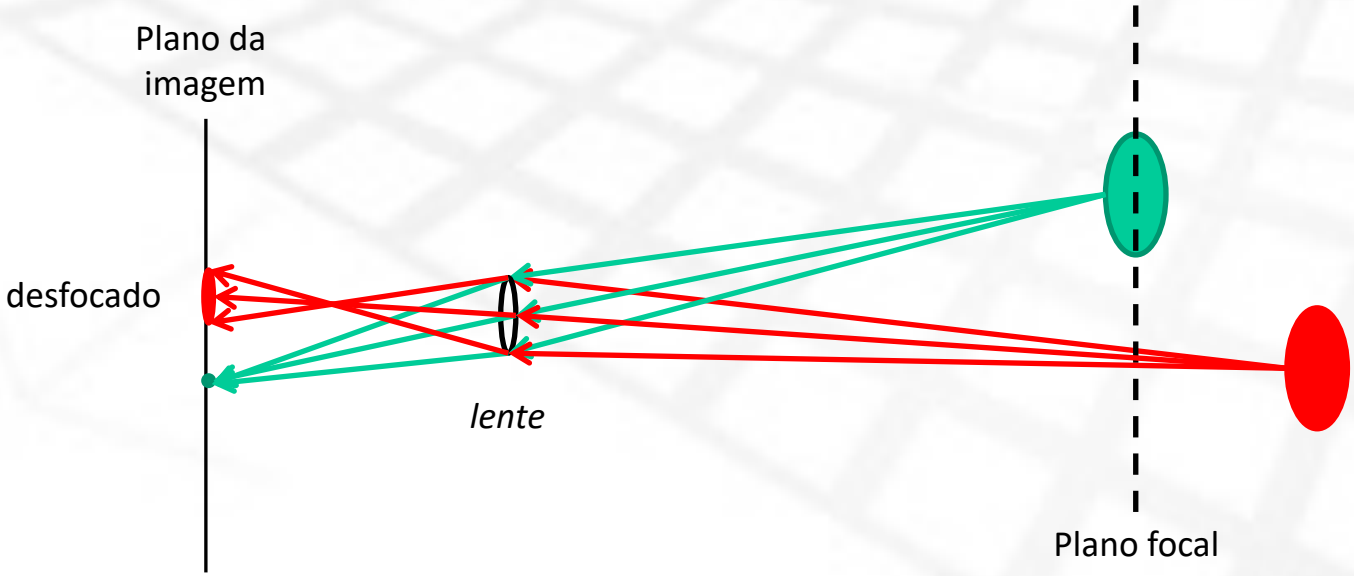
Ray tracing distribuído: *depth of field*

- Câmaras *pinhole*: abertura infinitesimal
  todos os objectos em foco independentemente da distância



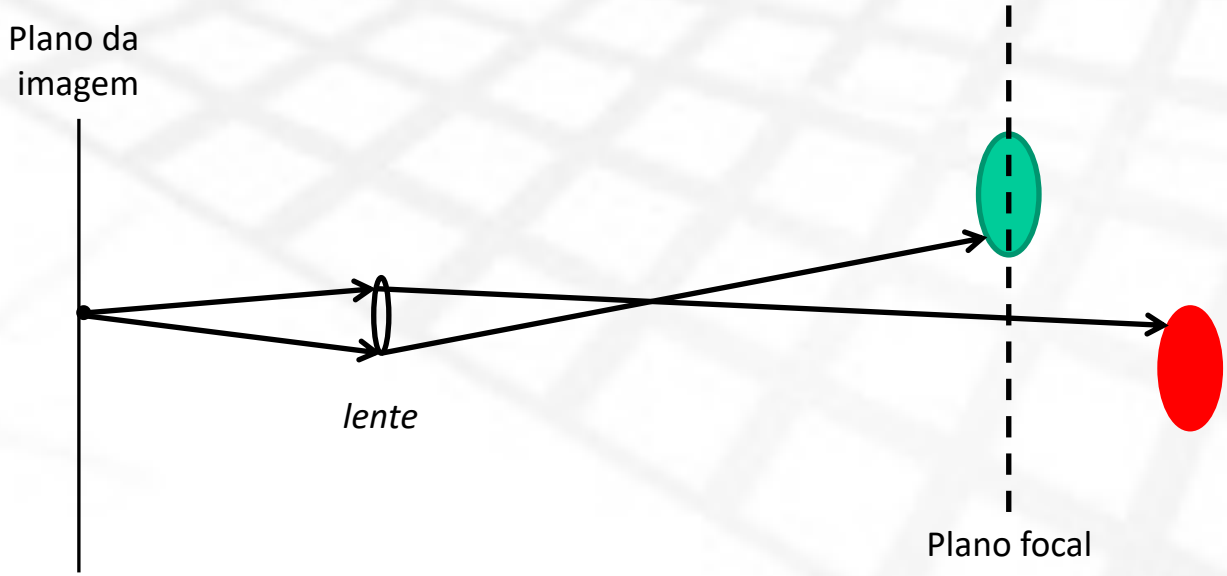
Ray tracing distribuído: *depth of field*

- Câmara com abertura finita
  Apenas os objectos situados no plano focal estão em foco

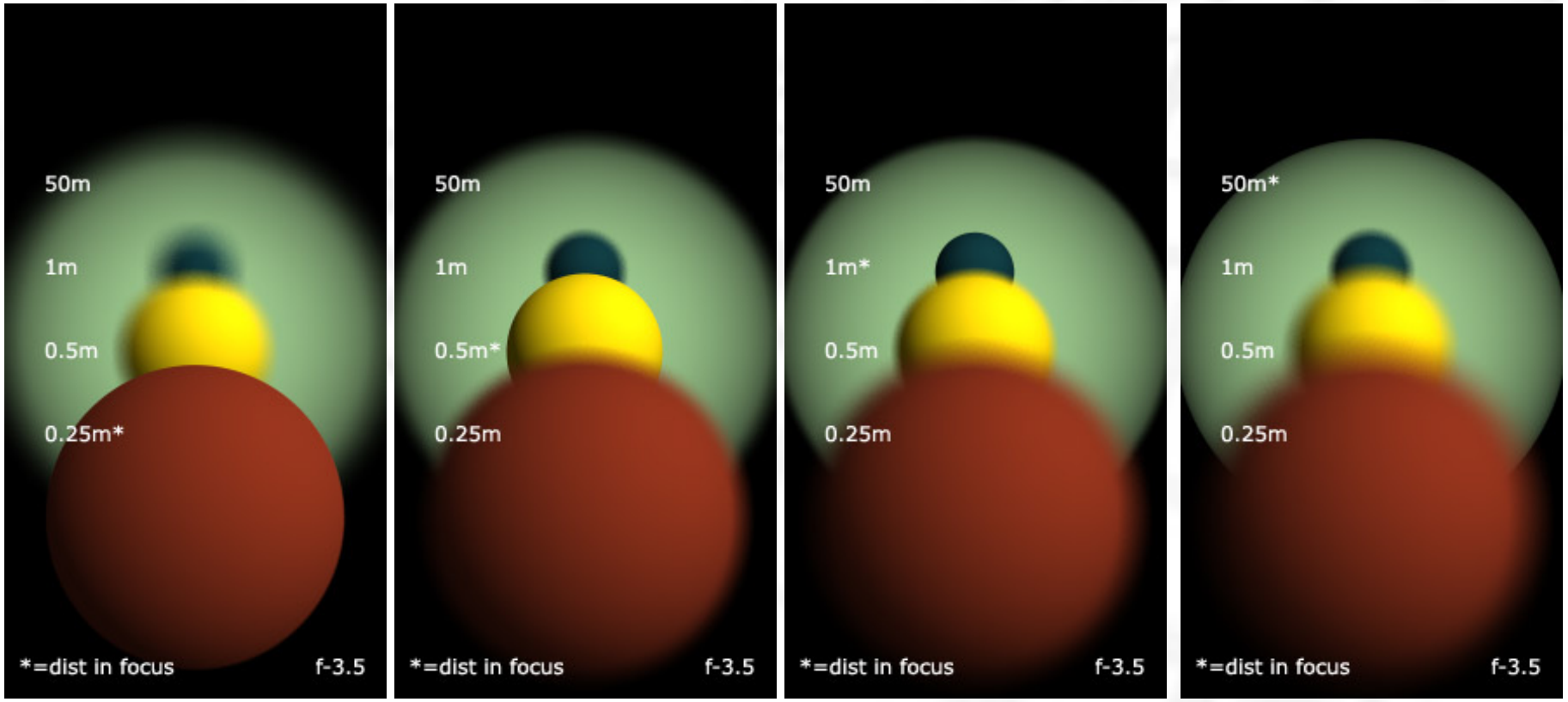


Ray tracing distribuído: *depth of field*

- Distribuir os raios primários (do mesmo pixel) no ângulo sólido subentendido pela lente



Ray tracing distribuído: *depth of field*



144 raios primários por pixel

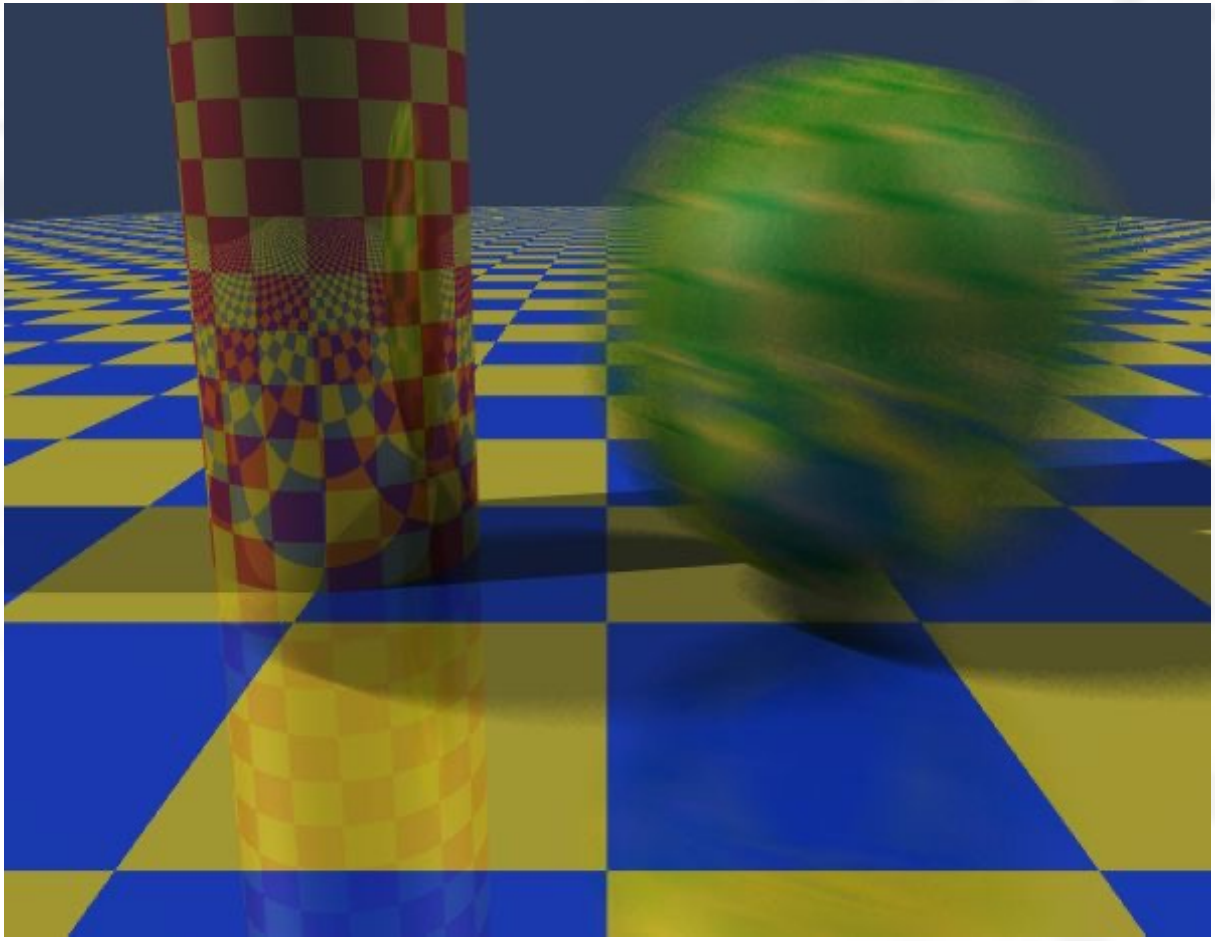
Ray tracing distribuído: *motion blur*

- A câmara virtual de um *ray tracer* modela uma câmara com um tempo de exposição infinitesimalmente pequeno
- Os objectos, mesmo que deslocando-se rapidamente relativamente à câmara, aparecem perfeitamente definidos.
- Uma câmara real apresenta os objectos com grande velocidade mal definidos, podendo-se mesmo ver parte da cena por trás desses objectos
- Este efeito é conhecido como *motion blur*

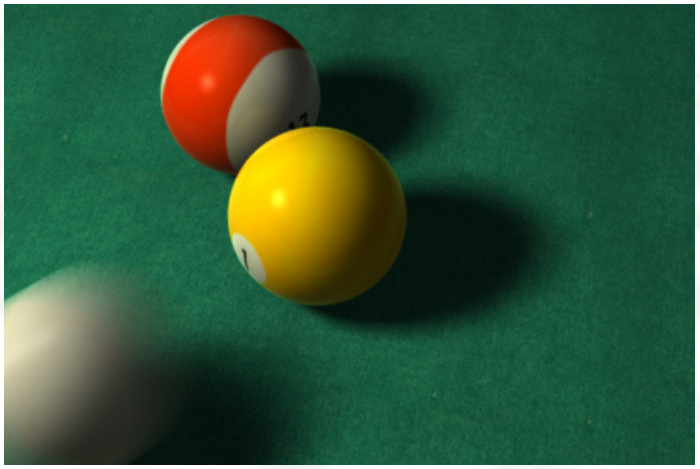
Ray tracing distribuído: *motion blur*

- Este efeito pode ser conseguido distribuindo os raios primários no tempo.
- Para cada raio, correspondente a um instante de tempo é necessário determinar a posição dos objectos em movimento
- A integração da contribuição dos vários raios reproduz com fidelidade o efeito de *motion blur*

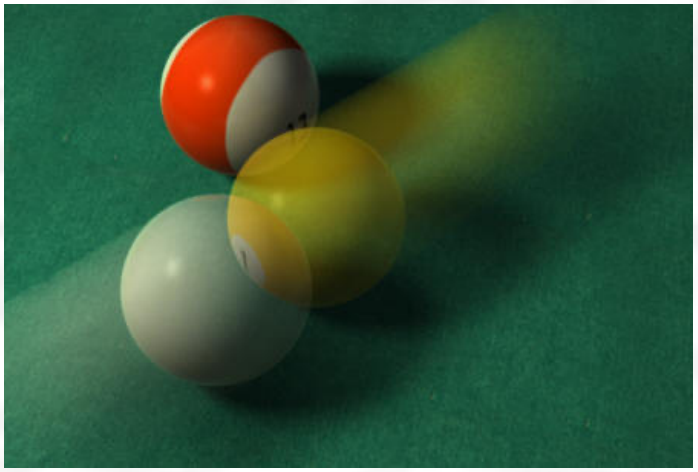
Ray tracing distribuído: *motion blur*



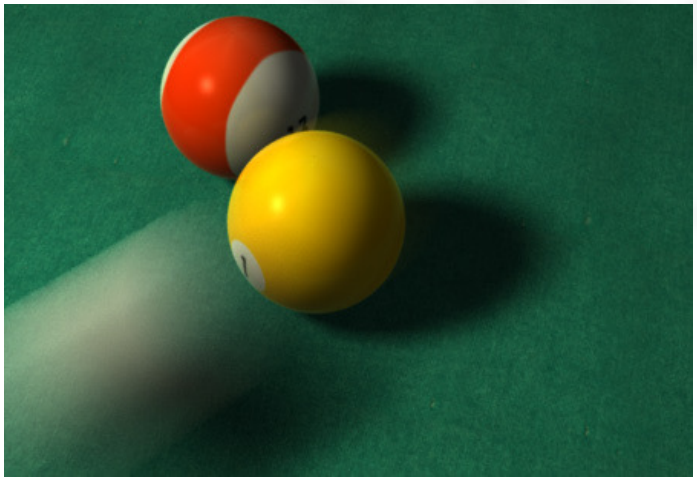
Ray tracing distribuído: *motion blur*



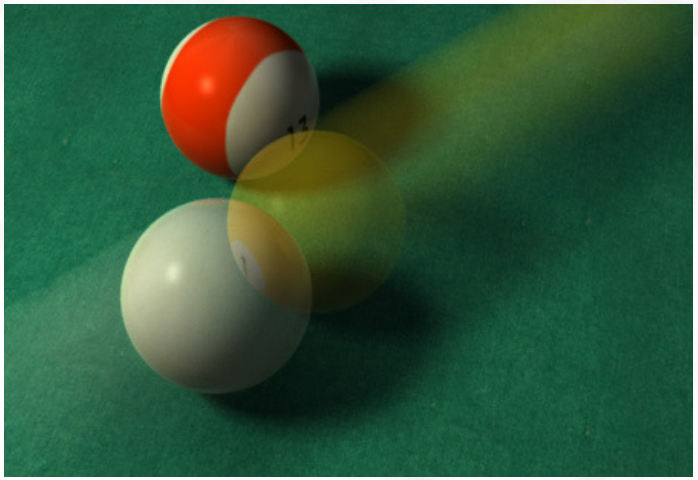
0.1 seg



0.5 seg



0.25 seg



0.75 seg

Ray tracing distribuído: corolário

- Distribuir as amostras no domínio de integração

Fenómeno	Domínio
Antialiasing	Plano da imagem
Soft shadows	Fonte de luz
Glossiness	Cone da BRDF
Depth of field	Lente
Motion blur	Tempo

- Seleccionar as amostras estocasticamente
- Usar Integração de Monte Carlo

$$I \approx \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(w_i)}{p(w_i)}$$

Ray tracing distribuído: corolário

- Combinar para cada trajecto (iniciado no raio primário) a distribuição sobre todos os domínios relevantes. Exemplo: estocasticamente

1. Seleccionar o **instante** e ajustar a cena;
2. Seleccionar o **ponto na área do pixel**;
3. Seleccionar o **ponto na lente**;
4. Disparar o raio primário e determinar o ponto de intersecção ;
5. Seleccionar o **ponto na fonte de luz** e disparar *shadow ray*;
6. Seleccionar **direcção no *glossy lobe***, disparar raio secundário e entrar recursividade (a partir do ponto 4);
7. Seleccionar **direcção no *lobe* translucente**, disparar raio secundário e entrar recursividade (a partir do ponto 4);
8. **Integrar** contribuições usando quadratura de **Monte Carlo.**