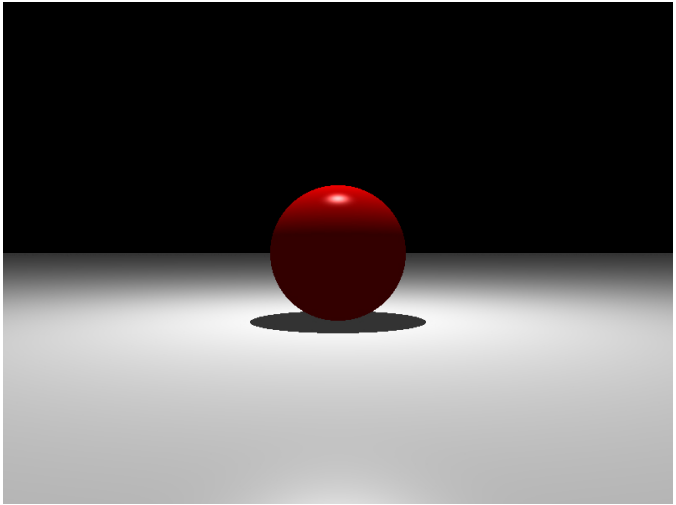
Traçado de Raios - Parte II

INF2604 – Fundamentos de Computação Gráfica

Waldemar Celes
celes@inf.puc-rio.br

Departamento de Informática, PUC-Rio

Algoritmo de traçado de raios



Algoritmo de traçado de raios

Tópicos

- *Antialising*
- Instanciação de objetos
- Fontes de luz de área
- Objeto "caixa"
- Reflexão
- Refração

Antialiasing

Geração de múltiplos raios por pixels

- Amostras aleatórias uniformemente distribuídas
- Gerador "canônico" de números aleatórios

$$\xi \in [0, 1)$$

- Em C++:

```
inline float Random ()
{
  return (float)rand()/RAND_MAX;
}
```

Antialising

Associação de um *sampler* a um objeto $Film$

- Amostra no centro do pixel

$Film.SampleCount()$
return 1

$GetSample(i, j)$
return $\left(\frac{i + 0.5}{w}, \frac{j + 0.5}{h}\right)$

Antialising

Associação de um *sampler* a um objeto $Film$

- Amostra no centro do pixel

$Film.SampleCount()$
return 1

$GetSample(i, j)$
return $\left(\frac{i + 0.5}{w}, \frac{j + 0.5}{h}\right)$

- Amostras com distribuição uniforme

$Film.SampleCount()$
return $nsamples$

$Film.GetSample(i, j)$
return $\left(\frac{i + \xi}{w}, \frac{j + \xi}{h}\right)$

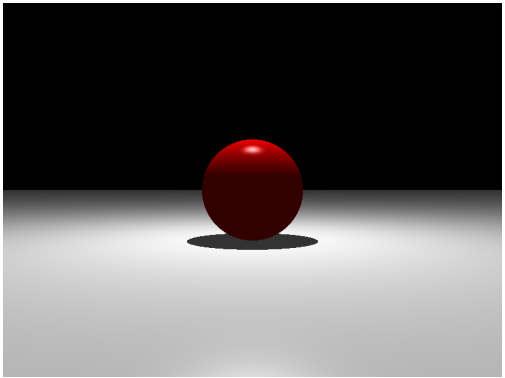
Traçado de raios

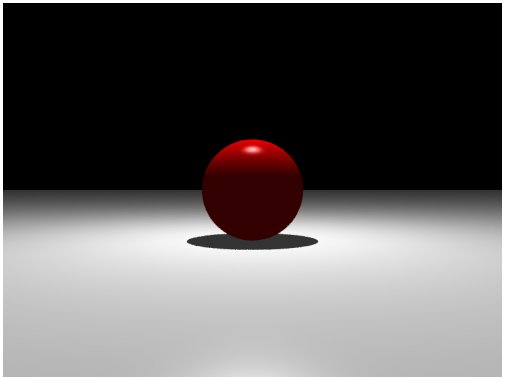
Múltiplas amostras por pixel

$$
\begin{aligned}
&Render(film, camera, scene)\\
&\quad n = film.SampleCount()\\
&\quad \textbf{for } each\ pixel\ (i,j)\ in\ film\\
&\qquad \mathbf{c} = \mathbf{0}\\
&\qquad \textbf{for } i = 1, n \textbf{ do}\\
&\qquad\quad x_n, y_n = film.GetSample(i,j)\\
&\qquad\quad ray = camera.GenerateRay(x_n, y_n)\\
&\qquad\quad \mathbf{c} \mathrel{+}= TraceRay(ray, scene)\\
&\qquad film.SetValue(i, j, \mathbf{c}/n)
\end{aligned}
$$

Antialising

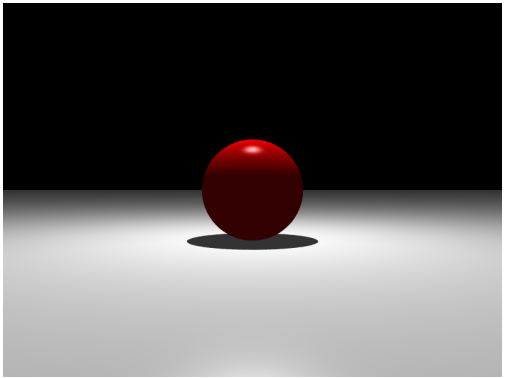
Múltiplas amostras por pixel: 1 e 4

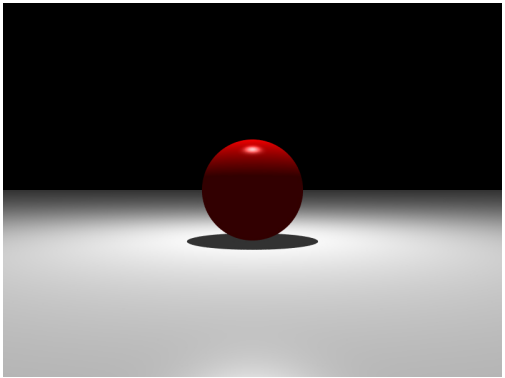




Antialising

Múltiplas amostras por pixel: 16 e 64

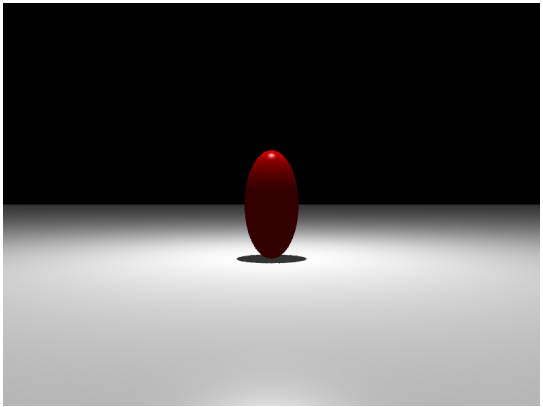




Instanciação de objetos

Cena com elipsóide

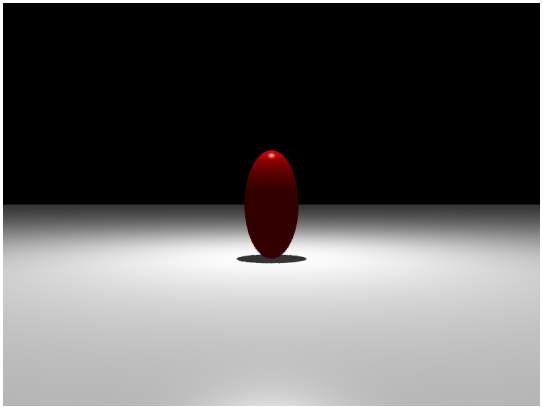
- É necessário interseção com elipsóide?



Instanciação de objetos

Cena com elipsóide

- É necessário interseção com elipsóide?



- Não, basta trabalhar com transformações

Transformação de objetos

Transformações usuais

- Translação: `glm::translate(tx,ty,tz)`
- Scala: `glm::scale(sx,sy,sz)`
- Rotação: `glm::rotate(a,rx,ry,rz)`

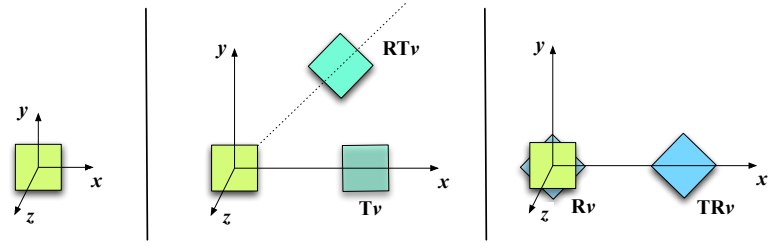
Transformação de objetos

Transformações usuais

- Translação: `glm::translate(tx,ty,tz)`
- Scala: `glm::scale(sx,sy,sz)`
- Rotação: `glm::rotate(a,rx,ry,rz)`

Combinação de transformações

- Ordem das transformações alteram o resultado



Transformação de objetos

Coordenadas homogêneas

$$\mathbf{p} = \begin{bmatrix} x & y & z & w \end{bmatrix}^T$$

Matrizes de transformação: espaço local $\rightarrow$ espaço global

$$\mathbf{M} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{bmatrix}$$

Transform
- **M** (4x4)
- **M**inv (4x4)

Instance
- shape
- material / light
- transform

Transformação de objetos

Coordenadas homogêneas

$$\mathbf{p} = \begin{bmatrix} x & y & z & w \end{bmatrix}^T$$

Matrizes de transformação: espaço local $\rightarrow$ espaço global

$$\mathbf{M} = \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{bmatrix}$$

Transform
- **M** (4x4)
- **M**inv (4x4)

Instance
- shape
- material / light
- transform

Transformação de normais

$$\mathbf{p} = \mathbf{M}\,\mathbf{p}'$$
$$\mathbf{n} = \mathbf{M}^{-\mathsf{T}}\mathbf{n}'$$

- onde $\mathbf{n}$ é a equação do plano suporte:

$$\mathbf{n} = \begin{bmatrix} n_x & n_y & n_z & 0 \end{bmatrix}^T$$

Transformação de objetos

Interseção raio-objeto: sempre no espaço local do objeto

1. Transforma raio para espaço local

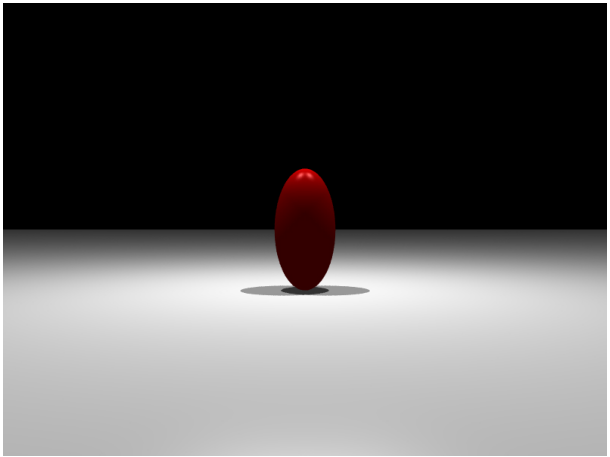
$$\mathbf{r}(t) \longrightarrow \mathbf{r}'(t)$$
$$\mathbf{r}(t) = f(\mathbf{o}, \hat{\mathbf{d}}) \longrightarrow \mathbf{r}'(t) = f(\mathbf{o}', \hat{\mathbf{d}}')$$
$$\mathbf{o}' = \mathbf{M}^{-1}\,\mathbf{o}$$
$$\hat{\mathbf{d}}' = normalize(\mathbf{M}^{-1}(\mathbf{o} + \hat{\mathbf{d}}) - \mathbf{o}')$$

2. Calcula interseção raio-objeto no espaço local: $hit = f(\mathbf{p}', \mathbf{n}')$
3. Transforma ponto e normal da interseção para espaço global: $hit = f(\mathbf{p}, \hat{\mathbf{n}})$

$$\mathbf{p} = \mathbf{M}\,\mathbf{p}'$$
$$\mathbf{n} = \mathbf{M}^{-\mathsf{T}}\,\mathbf{n}'$$

Múliplas fontes de luz

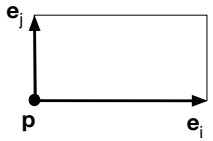
Duas fontes de luz pontual



Fonte de luz de área

Exemplo: fonte de luz retangular

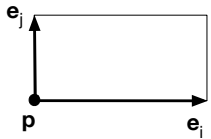
- Representação
  - $\mathbf{p}$: origem
  - $\vec{\mathbf{e}}_i$: aresta i
  - $\vec{\mathbf{e}}_j$: aresta j



Fonte de luz de área

Exemplo: fonte de luz retangular

- Representação
  - $\mathbf{p}$: origem
  - $\vec{\mathbf{e}}_i$: aresta i
  - $\vec{\mathbf{e}}_j$: aresta j



- Normal da fonte de luz
  - $\vec{\mathbf{n}} = normalize(\vec{\mathbf{e}}_i \times \vec{\mathbf{e}}_j)$
- Área da fonte de luz
  - $A = ||\vec{\mathbf{e}}_i \times \vec{\mathbf{e}}_j||$

Fonte de luz de área

Exemplo: fonte de luz retangular

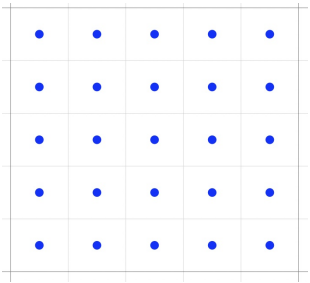
- Como integrar a contribuição da fonte?

Fonte de luz de área

Exemplo: fonte de luz retangular

- Como integrar a contribuição da fonte?

Amostragem regular



Fonte de luz de área

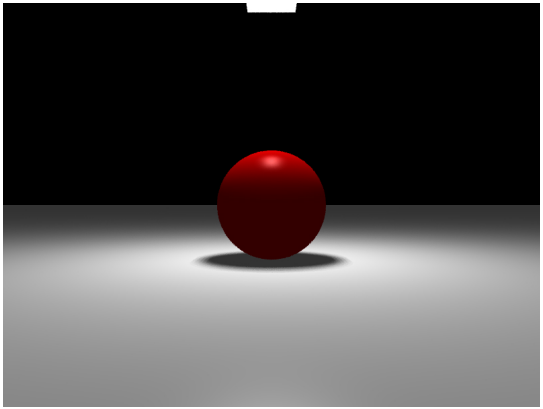
Revisitando função que captura radiância da fonte de luz

$$
\begin{aligned}
&Light.SampleRadiance(scene, \mathbf{p}) \\
&\quad \mathbf{s}, \hat{\mathbf{n}}_s = Light.GetSample(\mathbf{p}) \\
&\quad \hat{\mathbf{l}} = normalize(\mathbf{s} - \mathbf{p}) \\
&\quad ray = Ray(hit.\mathbf{p}, \hat{\mathbf{l}}) \\
&\quad hit_s = scene.ComputeIntersection(ray) \\
&\quad \textbf{if } hit_s.light == this \textbf{ then} \\
&\qquad r = ||\mathbf{p} - \mathbf{s}|| \\
&\qquad \mathbf{L}_i = \frac{this.P\ (-\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}_s)}{r^2} \frac{GetArea()}{SampleCount()} \\
&\qquad \textbf{return } \mathbf{L}_i, \hat{\mathbf{l}} \\
&\quad \textbf{else} \\
&\qquad \textbf{return } \mathbf{0}, \mathbf{0}
\end{aligned}
$$

Fonte de luz de área

Amostragem regular: *banding effects*

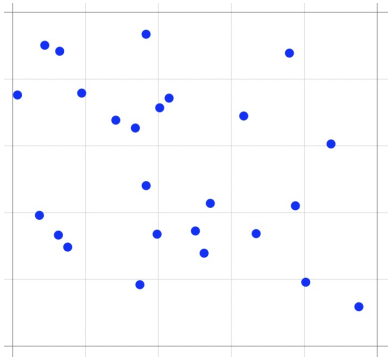
- Simula uma série de fontes de luz pontuais



Renderização com 4 amostras por pixel e 25 amostras por fonte de luz

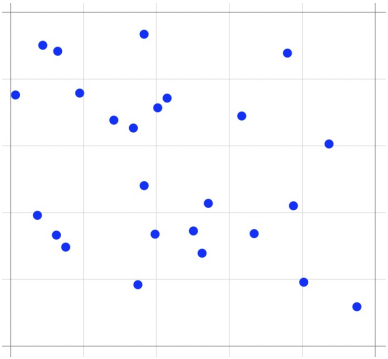
Fonte de luz de área

Amostragem uniforme



Fonte de luz de área

Amostragem uniforme



Geração de amostras

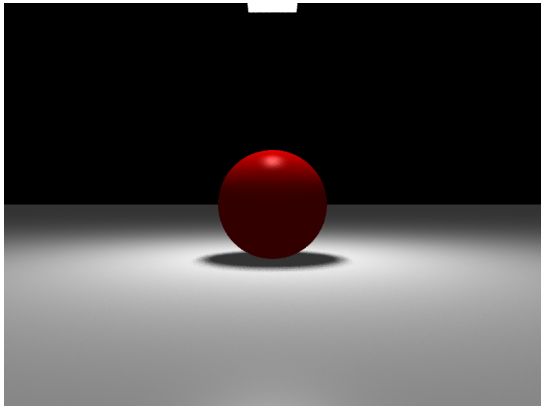
$$AreaLight.SampleCount()$$
return $nsamples$

$$AreaLight.GetSample()$$
return $\mathbf{p} + \xi_1 \vec{\mathbf{e}}_i + \xi_2 \vec{\mathbf{e}}_j$

Fonte de luz de área

Amostragem uniform: ruído elevado

- Grande variância

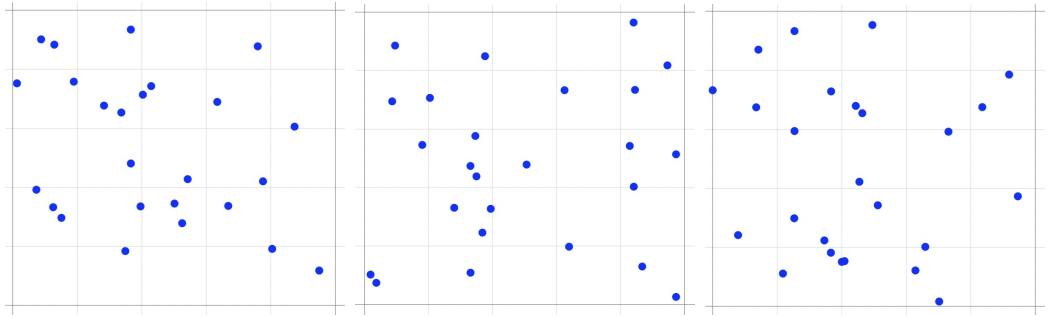


Renderização com 4 amostras por pixel e 25 amostras por fonte de luz

Fonte de luz de área

Amostragem uniforme

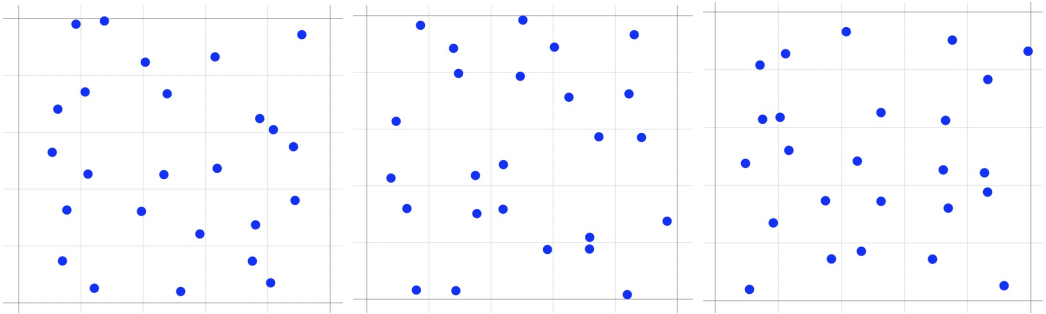
- Grande variância



Fonte de luz de área

Amostragem estratificada

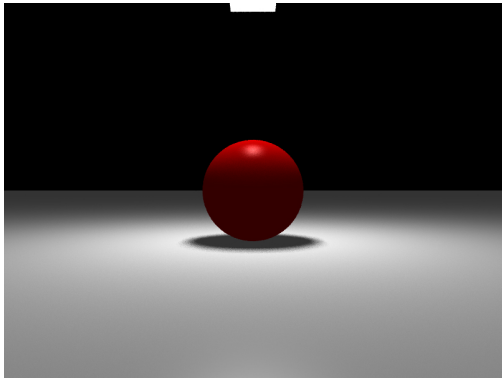
- Matém aleatoriedade com menor variância

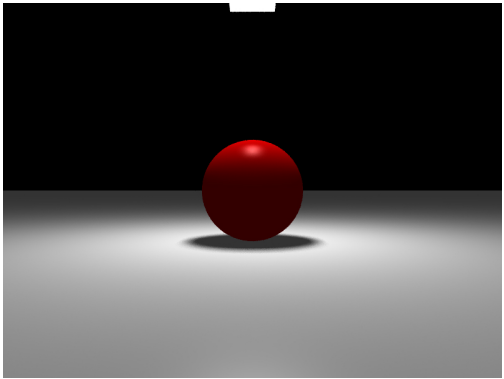


Fonte de luz de área

Amostragem estratificada: diminuição do ruído

- Menor variância



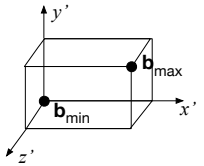


Renderização com 4 amostras por pixel e 25 amostras por fonte de luz

Caixa alinhada com objeto

Representação de caixa

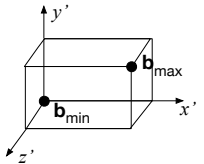
- Caixa alinhada com os eixos principais
  - $\mathbf{b}_{min}$: ponto de mínimo
  - $\mathbf{b}_{max}$: ponto de máximo



Caixa alinhada com objeto

Representação de caixa

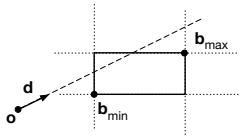
- Caixa alinhada com os eixos principais
  - $\mathbf{b}_{min}$: ponto de mínimo
  - $\mathbf{b}_{max}$: ponto de máximo



Interseção raio-caixa

$$\mathbf{t}_0 = \frac{\mathbf{b}_{min} - \mathbf{o}}{\hat{\mathbf{d}}} \qquad \mathbf{t}_{near} = \min_i(\mathbf{t}_0, \mathbf{t}_1) \qquad t_{min} = \max_i \mathbf{t}_{near}$$

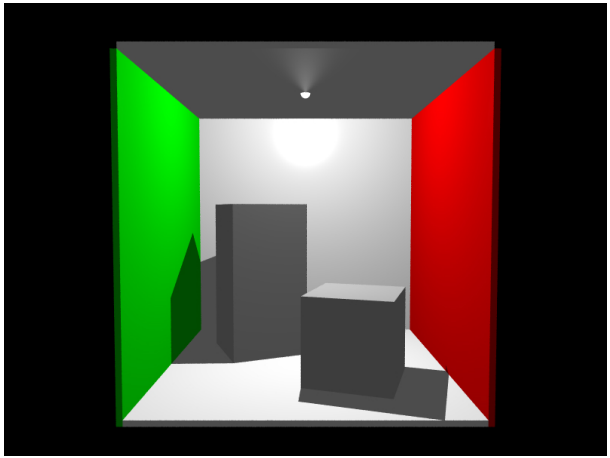
$$\mathbf{t}_1 = \frac{\mathbf{b}_{max} - \mathbf{o}}{\hat{\mathbf{d}}} \qquad \mathbf{t}_{far} = \max_i(\mathbf{t}_0, \mathbf{t}_1) \qquad t_{max} = \min \mathbf{t}_{far}$$



Instanciação de caixas

Versão simplificada do modelo "Cornell box"

- Iluminação com luz pontual



Reflexão

Materiais metálicos reflexivos

- Metais, espelhos etc.

Reflexão

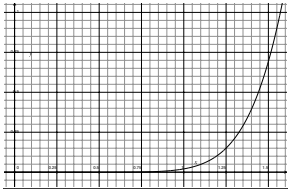
Materiais metálicos reflexivos

- Metais, espelhos etc.

Quanto de luz é refletida?

- Equações de Fresnel
  - Modelo simplificado de Schlick
    - $R_0(\lambda)$: reflectância à incidência normal (mínima)

$$R(\theta, \lambda) = R_0(\lambda) + (1 - R_0(\lambda))(1 - \cos\theta)^5$$



$f(x) = (1 - \cos x)^5$

Reflexão

Avaliação da cor refletida

$PhongMaterial.Eval(scene, hit, \mathbf{o})$
$\mathbf{c} = \mathbf{0}$
$\hat{\mathbf{v}} = normalize(\mathbf{o} - hit.\mathbf{p})$
for *each light source* l_s *in scene*
$\mathbf{L}_i, \hat{\mathbf{l}} = l_s.Radiance(scene, hit.\mathbf{p})$
$\mathbf{c} \mathrel{+}= \mathbf{m}_{dif} * \mathbf{L}_i * \hat{\mathbf{n}} \cdot \hat{\mathbf{l}}$
$\hat{\mathbf{r}} = reflect(-\hat{\mathbf{l}}, \hat{\mathbf{n}})$
$\mathbf{c} \mathrel{+}= \mathbf{m}_{spe} * \max(0, \hat{\mathbf{r}} \cdot \hat{\mathbf{v}})^{shi}$
return c

Reflexão

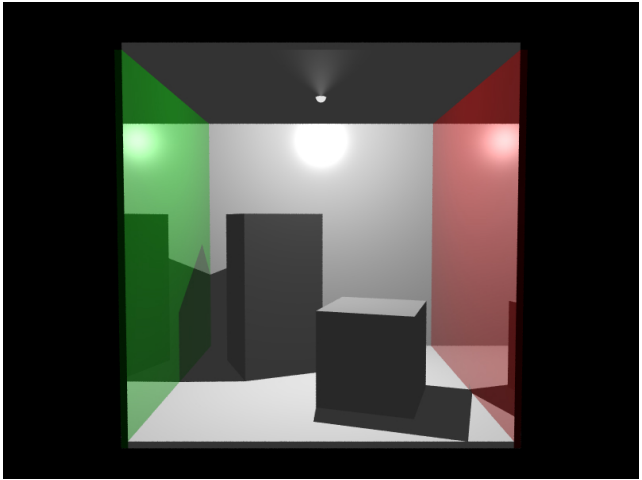
Avaliação da cor refletida

$PhongMaterial.Eval(scene, hit, \mathbf{o})$
$\mathbf{c} = \mathbf{0}$
$\hat{\mathbf{v}} = normalize(\mathbf{o} - hit.\mathbf{p})$
for *each light source* l_s *in scene*
$\mathbf{L}_i, \hat{\mathbf{l}} = l_s.Radiance(scene, hit.\mathbf{p})$
$\mathbf{c} \mathrel{+}= \mathbf{m}_{dif} * \mathbf{L}_i * \hat{\mathbf{n}} \cdot \hat{\mathbf{l}}$
$\hat{\mathbf{r}} = reflect(-\hat{\mathbf{l}}, \hat{\mathbf{n}})$
$\mathbf{c} \mathrel{+}= \mathbf{m}_{spe} * \max(0, \hat{\mathbf{r}} \cdot \hat{\mathbf{v}})^{shi}$
return c

$PhongMetal.Eval(scene, hit, \mathbf{o})$
$\mathbf{p} = hit.\mathbf{p}$
$\hat{\mathbf{n}} = hit.\hat{\mathbf{n}}$
$\hat{\mathbf{v}} = normalize(\mathbf{o} - \mathbf{p})$
$R = R_0 + (1 - R_0)(1 - \hat{\mathbf{v}} \cdot \hat{\mathbf{n}})^5$
$\mathbf{c} = (1 - R)\ PhongMaterial.Eval(scene, hit, \mathbf{o})$
$\hat{\mathbf{r}} = normalize(reflect(-\hat{\mathbf{v}}, \hat{\mathbf{n}}))$
$ray = Ray(\mathbf{p}, \hat{\mathbf{r}})$
$\mathbf{c} \mathrel{+}= R\ scene.TraceRay(ray)$
return c

Reflexão

Exemplo: "Cornell box" com paredes reflexivas



Refração

Materiais transparentes

- Água, vidro, diamante etc.

Refração

Materiais transparentes

- Água, vidro, diamante etc.

Propriedades importantes

- Quanto de luz é refletida, dado um ângulo de incidência
- Quanto de luz é absorvida quando se propaga para dentro do meio
- Quais as direções dos raios de reflexão e refração

Materiais transparentes

Reflexividade dos materiais

$$R_0(\lambda) = \left(\frac{\eta(\lambda) - 1}{\eta(\lambda) + 1} \right)^2$$

- $\eta(\lambda)$ é o índice de refração
- A quantidade de luz refratada é o que não foi refletido (conservação de energia)

Materiais transparentes

Reflexividade dos materiais

$$R_0(\lambda) = \left(\frac{\eta(\lambda) - 1}{\eta(\lambda) + 1}\right)^2$$

- $\eta(\lambda)$ é o índice de refração
- A quantidade de luz refratada é o que não foi refletido (conservação de energia)

Valores comuns do índice de refração

- Água: $\eta = 1.33$
- Vidro: $\eta \in [1.4, 1.7]$
- Diamante: $\eta = 2.4$
- Simulação de dispersão exige variação por comprimento de onda

Materiais transparentes

Direção do raio refratado

- Lei de Snell

$$\eta \sin\theta = \eta_t \sin\phi$$

- Trabalhando com cosseno $(\sin^2\theta + \cos^2\theta = 1)$

$$\cos^2\phi = 1 - \frac{\eta^2(1-\cos^2\theta)}{\eta_t^2}$$

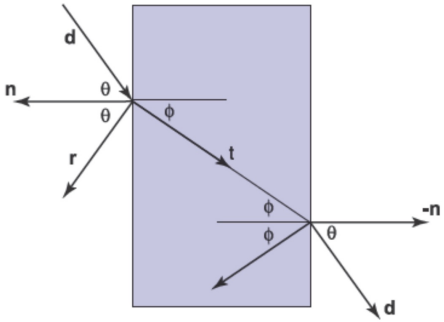
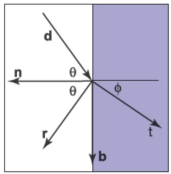


Figura extraída de Fundamentals of Computer Graphics, S. Marschner and P. Shirley

Materiais transparentes

Vetor de refração

- Da figura ao lado



$$\hat{\mathbf{t}} = \hat{\mathbf{b}}\ \sin\phi - \hat{\mathbf{n}}\ \cos\phi$$

$$\hat{\mathbf{d}} = \hat{\mathbf{b}}\ \sin\theta - \hat{\mathbf{n}}\ \cos\theta \therefore \hat{\mathbf{b}} = \frac{\hat{\mathbf{d}} + \hat{\mathbf{n}}\ \cos\theta}{\sin\theta}$$

- Como $\sin\theta = \eta_t \sin\phi/\eta$, podemos então obter $\hat{\mathbf{t}}$

$$\hat{\mathbf{t}} = \frac{\eta(\hat{\mathbf{d}} + \ \hat{\mathbf{n}}\cos\theta)}{\eta_t} - \hat{\mathbf{n}}\ \cos\phi = \frac{\eta(\hat{\mathbf{d}} + \hat{\mathbf{n}}\ (-\hat{\mathbf{d}}\cdot\hat{\mathbf{n}}))}{\eta_t} - \hat{\mathbf{n}}\ \sqrt{1 - \frac{\eta^2(1-(-\hat{\mathbf{d}}\cdot\hat{\mathbf{n}})^2)}{\eta_t^2}}$$

- Se o radicando for negativo, não tem refração
- Função $glm :: refract(\hat{\mathbf{d}}, \hat{\mathbf{n}}, \eta/\eta_t)$, nesse caso, retorna $\mathbf{0}$

Figura extraída de Fundamentals of Computer Graphics, S. Marschner and P. Shirley

Materiais transparentes

Materiais homogêneos impuros (ex. vidros coloridos)

- Atenuação da intesidade do raio quando atravessa o material

Lei de Beer

$$I(s) = I_0 \, a(\lambda)^s$$

- onde $a(\lambda)$ é a constante de atenuação

Materiais transparentes

Materiais homogêneos impuros (ex. vidros coloridos)

- Atenuação da intesidade do raio quando atravessa o material

Lei de Beer

$$I(s) = I_0 \, a(\lambda)^s$$

- onde $a(\lambda)$ é a constante de atenuação

Implicações

- Raios refratados devem ter intensidade atenuada
- Raios para captura de iluminação direta devem ter intensidade atenuada
  - Tratamos de forma aproximada com sendo raios sem desvios de refração

Materiais transparentes

Avaliação da interação luz-matéria

$PhongDieletrics.Eval(scene, hit, \mathbf{o})$

$\mathbf{p} = hit.\mathbf{p}$

$\hat{\mathbf{n}} = hit.\hat{\mathbf{n}}$

$\hat{\mathbf{v}} = normalize(\mathbf{o} - \mathbf{p})$

$R_0 = ((\eta - 1) \,/\, (\eta + 1))^2$

$R = R_0 + (1 - R_0)(1 - \hat{\mathbf{v}} \cdot \hat{\mathbf{n}})^5$

$\hat{\mathbf{r}} = normalize(reflect(-\hat{\mathbf{v}}, \hat{\mathbf{n}}))$

$ray = Ray(\mathbf{p}, \hat{\mathbf{r}})$

$\mathbf{c} = R\ scene.TraceRay(ray)$

if $hit.IsBackfacing()$ **then**

  $\mathbf{I} = \mathbf{a}^{||\mathbf{o}-\mathbf{p}||}$

  $ratio = 1/\eta$

else

  $\mathbf{I} = \mathbf{1}$

  $ratio = \eta/1$

$\hat{\mathbf{r}} = normalize(-\hat{\mathbf{v}}, \hat{\mathbf{n}}, ratio)$

if $\hat{\mathbf{r}}$ **then**

  $ray = Ray(\mathbf{p}, \hat{\mathbf{r}})$

  $\mathbf{c}$ += $(1 - R)\ scene.TraceRay(ray)$

return $I * \mathbf{c}$

Materiais transparentes

Radiância da fonte de luz

$Light.SampleRadiance(scene, \mathbf{p})$
  $\mathbf{s}, \hat{\mathbf{n}}_s = Light.GetSample(\mathbf{p})$
  $\hat{\mathbf{l}} = normalize(\mathbf{s} - \mathbf{p})$
  $\mathbf{I} = \mathbf{1}$
  $ray = Ray(hit.\mathbf{p}, \hat{\mathbf{l}})$
  $hit_s = scene.ComputeIntersection(ray)$
  while $hit_s.material.IsTransparent()$**do**
    if $hit_s.IsBackfacing()$ **then**
      $I = I\ hit_s.material.a^{||\mathbf{p} - hit_s.\mathbf{p}||}$
    $ray = Ray(hit_s.\mathbf{p}, \hat{\mathbf{l}})$
    $hit_s = scene.ComputeIntersection(ray)$
  if $hit_s.light == this$ **then**
    $r = ||\mathbf{p} - \mathbf{s}||$
    $\mathbf{L}_i = I * \frac{this.P\ (-\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}_s)}{r^2} \frac{GetArea()}{SampleCount()}$
    return $\mathbf{L}_i, \hat{\mathbf{l}}$
  else
    return $\mathbf{0}, \mathbf{0}$

Materiais transparentes

Cena com vidro ($\mathbf{a} = (0.8, 0.9, 0.8)$)