UFABC

Computação Gráfica

André Brandão

Aula 08

Ray Tracing

(Traçados de Raios)

CONTINUAÇÃO

Sumário

- ~~Algoritmo básico de Ray Tracing~~
- ~~Perspectiva~~
- ~~Computação dos raios de visualização~~
- **Interseção raio-objeto (continuação)**
- Sombreamento (*Shading*)
- Um programa de Ray Tracing
- Sombras
- Reflexão especular ideal

Interseção raio-objeto

- ~~Interseção raio-esfera~~
- **Interseção raio-triângulo**
- Interseção raio-polígono

Interseção raio-triângulo

- Para detectarmos a interseção do raio com uma superfície paramétrica, usamos o sistema de equações em que as coordenadas Cartesianas satisfazem:

$$\left.\begin{aligned} x_e + t x_d &= f(u,v) \\ y_e + t y_d &= g(u,v) \\ z_e + t z_d &= h(u,v) \end{aligned}\right\} \text{ou} \qquad \mathbf{e} + t\mathbf{d} = \mathbf{f}(u,v)$$

Interseção raio-triângulo

- No caso em que a superfície paramétrica é um plano paramétrico, a equação pode ser escrita na forma de vetor.
- Se os vértices do triângulo forem **a, b** e **c**, então a interseção ocorrerá quando:

$$\mathbf{e} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

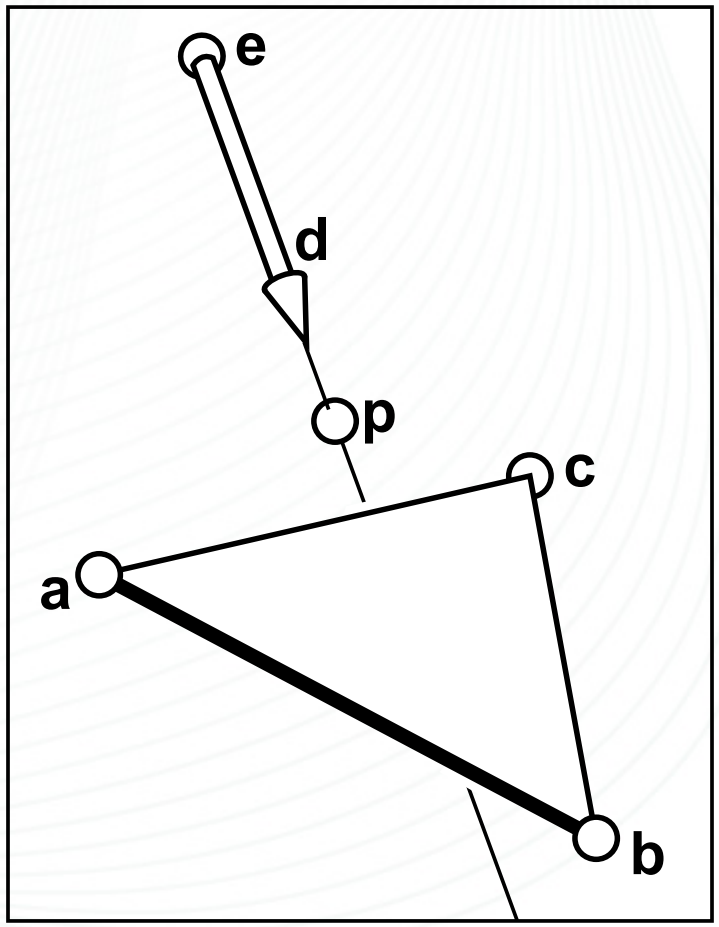
Interseção raio-triângulo

$$\mathbf{e} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

para algum **t, β** e **γ**. A interseção **p** estará em

e + td como apresentado na figura, ao lado.

- A interseção estará dentro do triângulo se
- **β > 0; γ > 0 e β + γ < 1**
- Caso contrário, o raio intercepta o plano, porém, fora do triângulo.



Interseção raio-triângulo

- Se a equação não tiver solução, então o raio não intercepta o plano, portanto, o raio é paralelo ao plano.
- Para resolver a equação, que está em forma de vetor, pode-se expandir a mesma para a forma de coordenadas:

$$x_e + tx_d = x_a + \beta(x_b - x_a) + \gamma(x_c - x_a),$$
$$y_e + ty_d = y_a + \beta(y_b - y_a) + \gamma(y_c - y_a),$$
$$z_e + tz_d = z_a + \beta(z_b - z_a) + \gamma(z_c - z_a).$$

Interseção raio-triângulo

$$\begin{bmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} x_a - x_e \\ y_a - y_e \\ z_a - z_e \end{bmatrix}$$

Interseção raio-triângulo

$$\beta = \frac{\begin{vmatrix} x_a - x_e & x_a - x_c & x_d \\ y_a - y_e & y_a - y_c & y_d \\ z_a - z_e & z_a - z_c & z_d \end{vmatrix}}{|\mathbf{A}|}$$

Interseção raio-triângulo

$$\gamma = \frac{\begin{vmatrix} x_a - x_b & x_a - x_e & x_d \\ y_a - y_b & y_a - y_e & y_d \\ z_a - z_b & z_a - z_e & z_d \end{vmatrix}}{|\mathbf{A}|}$$

Interseção raio-triângulo

$$t = \frac{\begin{vmatrix} x_a - x_b & x_a - x_c & x_a - x_e \\ y_a - y_b & y_a - y_c & y_a - y_e \\ z_a - z_b & z_a - z_c & z_a - z_e \end{vmatrix}}{|\mathbf{A}|}$$

Interseção raio-triângulo

$$\mathbf{A} = \begin{bmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{bmatrix}$$

Interseção raio-triângulo

- Após a resolução da equação, teremos o ponto **p**, que intercepta o triângulo em questão.

Interseção raio-objeto

- ~~Interseção raio-esfera~~
- ~~Interseção raio-triângulo~~
- **Interseção raio-polígono**

Interseção raio-polígono

- Dado um polígono planar com *m* vértices (p_1, p_2 ... p_m) e uma normal à sua superfície **n**.
- Primeiro, computamos os pontos de interseção entre o raio **e+td** e o plano que contém o polígono pela equação implícita:

$$(\mathbf{p} - \mathbf{p}_1) \cdot \mathbf{n} = 0.$$

Interseção raio-polígono

- Ao fazer as substituições, e isolarmos a variável **t**, teremos:

$$t = \frac{(\mathbf{p}_1 - \mathbf{e}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}$$

- Isso permitirá que **p** seja calculado.
- Se **p** estiver dentro do polígono, então o raio intercepta o mesmo.
- Como saber se **p** está dentro do polígono?

Interseção raio-polígono

- Podemos responder essa pergunta por meio da projeção do ponto e os vértices que constituem o polígono até o plano *xy*.
- Emite-se um raio que sai de **p** e conta-se o número de interseções entre o raio e as arestas dos os limites do polígono.
- Se o número de interseções for ímpar, então o ponto está dentro do polígono. Isso porque o raio que entra, deve sair, o que cria um número par de interseções.
- Apenas o raio que começa dentro do polígono gerará um número ímpar de interseções.

Sumário

- ~~Algoritmo básico de Ray Tracing~~
- ~~Perspectiva~~
- ~~Computação dos raios de visualização~~
- ~~Interseção raio-objeto~~
- **Sombreamento (Shading)**
- Um programa de Ray Tracing
- Sombras
- Reflexão especular ideal

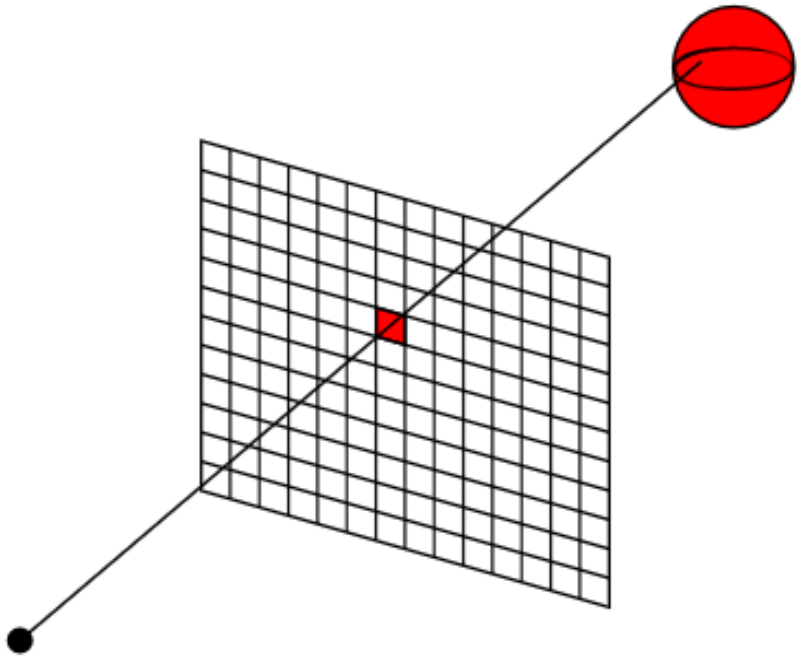
Algoritmo básico de ray tracing

- O traçado de raios contém três etapas:
  1. **Geração de raio: computa a origem e direção de cada ponto de vista de cada pixel, baseado na geometria de câmera;**
  2. **Interseção de raio: encontra a interseção mais próxima do raio com um objeto na cena; e**
  3. **Shading: computa a cor do pixel, baseado nos resultados da interseção de raio.**

Algoritmo básico de ray tracing

Para cada **pixel** faça

- compute o raio de visualização
- encontre o 1º objeto interceptado pelo raio e a sua normal $\vec{n}$
- atribua a cor do pixel o mesmo valor do ponto interceptado pelo raio, considerando a luz e, $\vec{n}$



Shading

- Uma vez que se sabe a superfície que é visível a um pixel, o valor deste pixel é computado pela avaliação do modelo de sombreamento – *Shading Model* – **Tonalização**.
- Muitos modelos de tonalização são desenvolvidos para capturar o processo de reflexão de luz, enquanto superfícies são iluminadas por fontes de luz e refletem parte da luz até a câmera (observador).

Shading

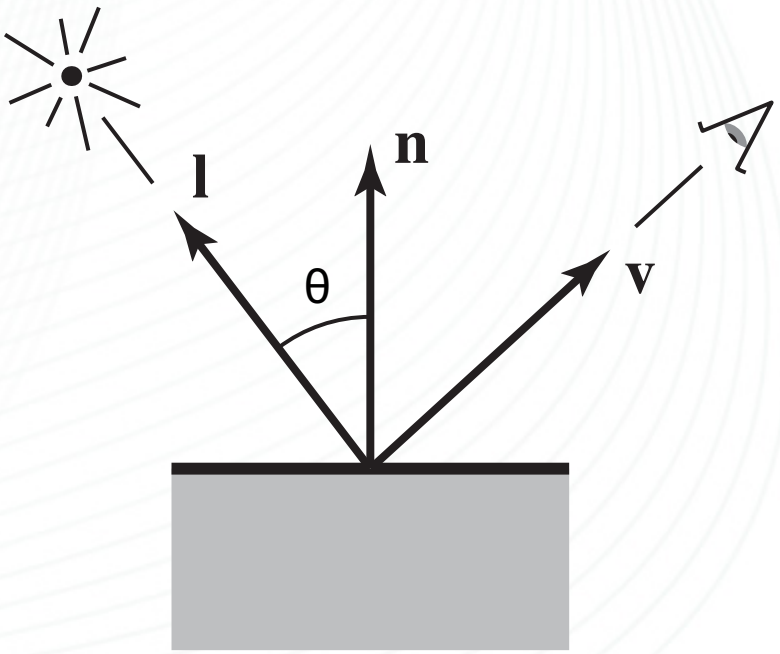
- Modelos de tonalização simples são definidos em termos da iluminação a partir de uma fonte de luz.
  - Deve-se saber a direção de luz ***l***, que é um vetor unitário que se origina a partir da fonte de luz.
  - A direção de visualização **v**, que é um vetor unitário que se origina a partir do observador ou da câmera.
  - A normal da superfície, que é um vetor unitário perpendicular à superfície, no ponto onde a reflexão acontece, bem como suas características, como: cor, brilho, entre outras.

Lambertian Shading

- A quantidade de energia de uma fonte de luz que está em uma área de uma superfície depende do ângulo entre a superfície e a luz.
- A superfície que está voltada para a luz recebe a iluminação máxima.
- A superfície que está tangente à luz não recebe qualquer iluminação

Lambertian Shading

- Entre um caso e outro, a iluminação é proporcional ao cosseno do ângulo θ entre a normal da superfície e a fonte de luz.

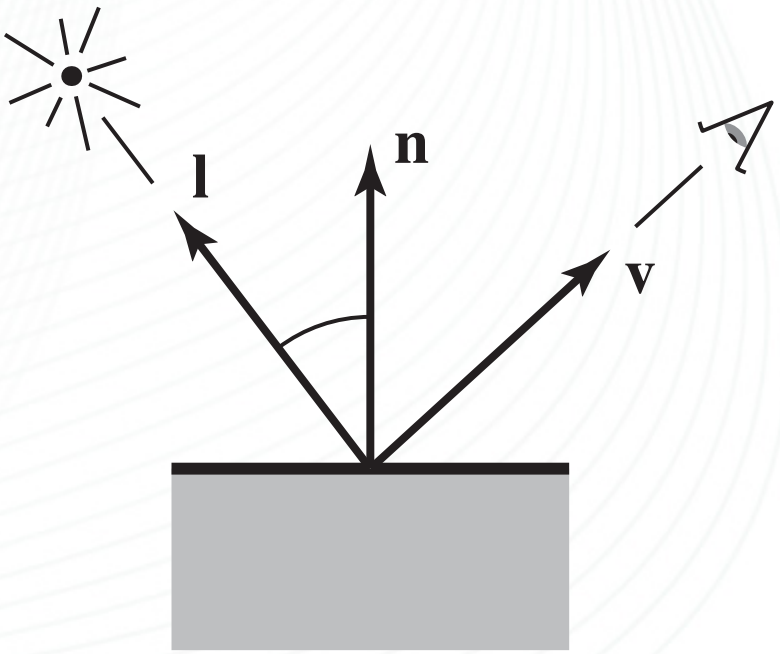


Lambertian Shading

- O modelo de tonalização de Lambert é definido por:

$$L = k_d \, I \max(0, \mathbf{n} \cdot \mathbf{l})$$

onde, *L* é a cor do pixel, k_d é o coeficiente de difusão (cor da superfície), ***I*** é a intensidade da fonte de luz.



Blinn-Phong Shading

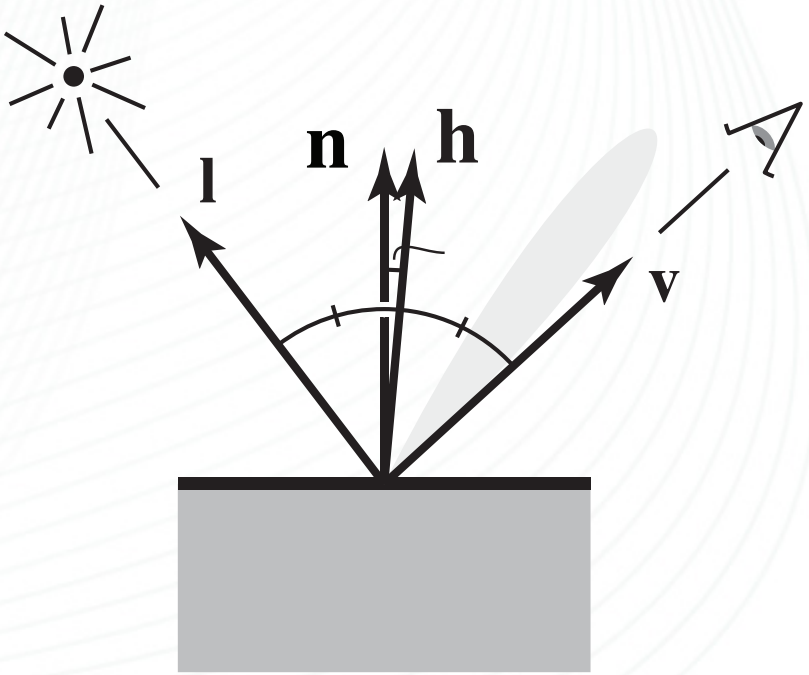
- No Modelo de Lampert, a cor da superfície não depende da direção de onde está o observador e produz imagens com aparências foscas.
- Muitos modelos de tonalização adicionam um componente especular ao modelo de Lampert. Dessa forma, o modelo de Lampert é o componente difuso.

Blinn-Phong Shading

- O modelo de Phong foi proposto em 1975 e atualizado por Blinn, em 1976.
- A ideia básica é que produzir reflexão que produz o máximo de brilho, quando **v** e **l** estão simetricamente posicionados, em relação à normal. Isso ocorre quando ocorreria em uma reflexão de espelho.
- O reflexo diminui, suavemente, a medida em que os vetores se distanciam da configuração de espelho.

Blinn-Phong Shading

- Pode-se saber a proximidade de algo em relação ao espelho, de acordo com o vetor **h**, que divide o ângulo entre **l** e **v**.
- Se **h** estiver próximo da normal da superfície, a componente especular deve ser brilhante. Senão, deve ser fraca.



Blinn-Phong Shading

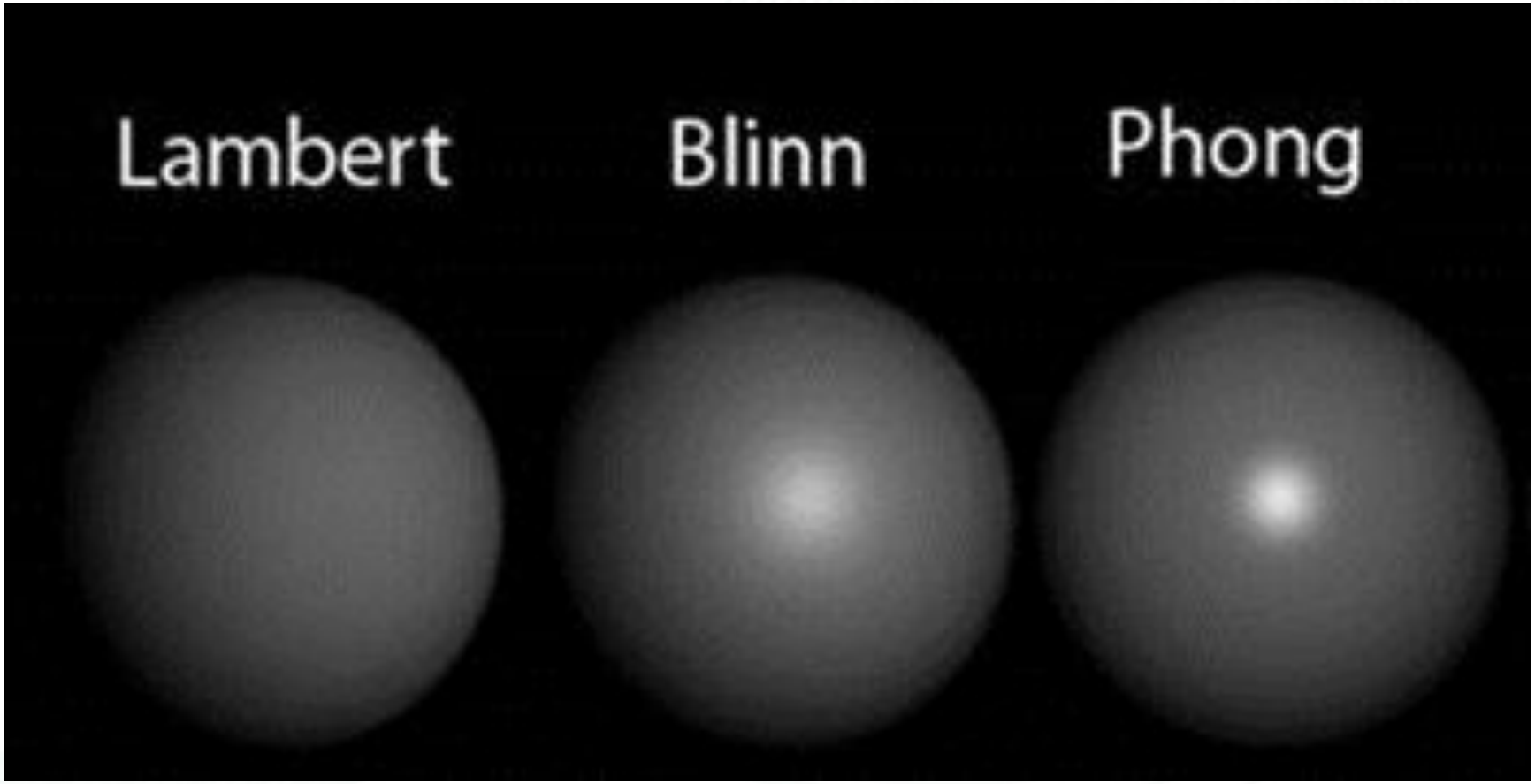
- Desta forma, o modelo de tonalização Blinn-Phong é definido da seguinte forma:

$$\mathbf{h} = \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|},$$

$$L = k_d \, I \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s \, I \max(0, \mathbf{n} \cdot \mathbf{h})^p$$

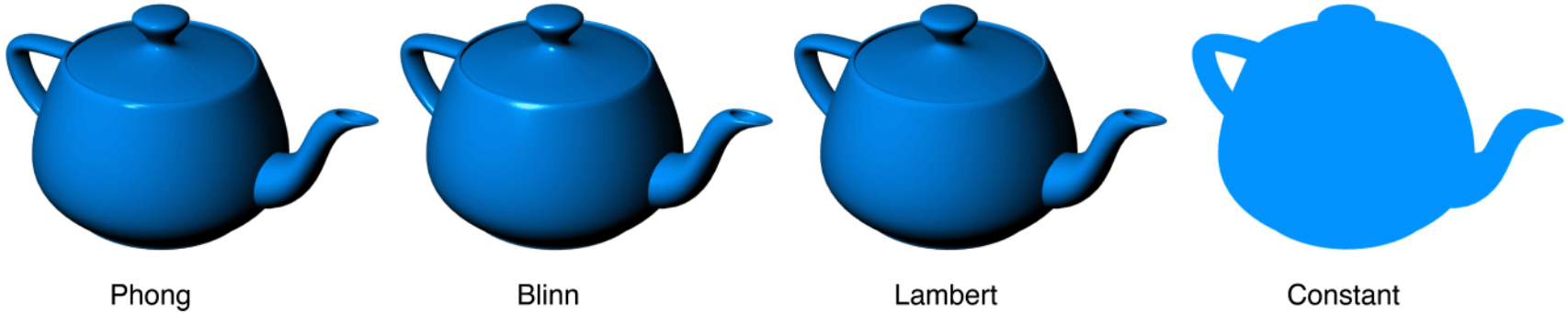
onde, k_s é o coeficiente especular, ou a cor especular da superfície.

Blinn-Phong Shading



https://thatgamedesignblog.files.wordpress.com/2014/09/blinn-phong-lambert.jpg

Blinn-Phong Shading



https://developer.apple.com/reference/scenekit/scnmaterial/1655321-lighting_models

Sumário

- ~~Algoritmo básico de Ray Tracing~~
- ~~Perspectiva~~
- ~~Computação dos raios de visualização~~
- ~~Interseção raio-objeto~~
- ~~Sombreamento (Shading)~~
- **Um programa de Ray Tracing**
- Sombras
- Reflexão especular ideal

Um programa de *ray-tracing*

- Nós sabemos como gerar um raio, dado um pixel, e como encontrar a interseção mais próxima deste raio com um objeto e como tonalizar o pixel, de acordo com a interseção e características da superfície.
- Isso é tudo que precisamos para criar um programa que produz imagens tonalizadas com superfícies oclusas removidas.

Um programa de *ray-tracing*

for each pixel **do**
    compute viewing ray
    if (ray hits an object with $t \in [0, \infty)$) **then**
        Compute **n**
        Evaluate shading model and set pixel to that color
    else
        set pixel color to background color

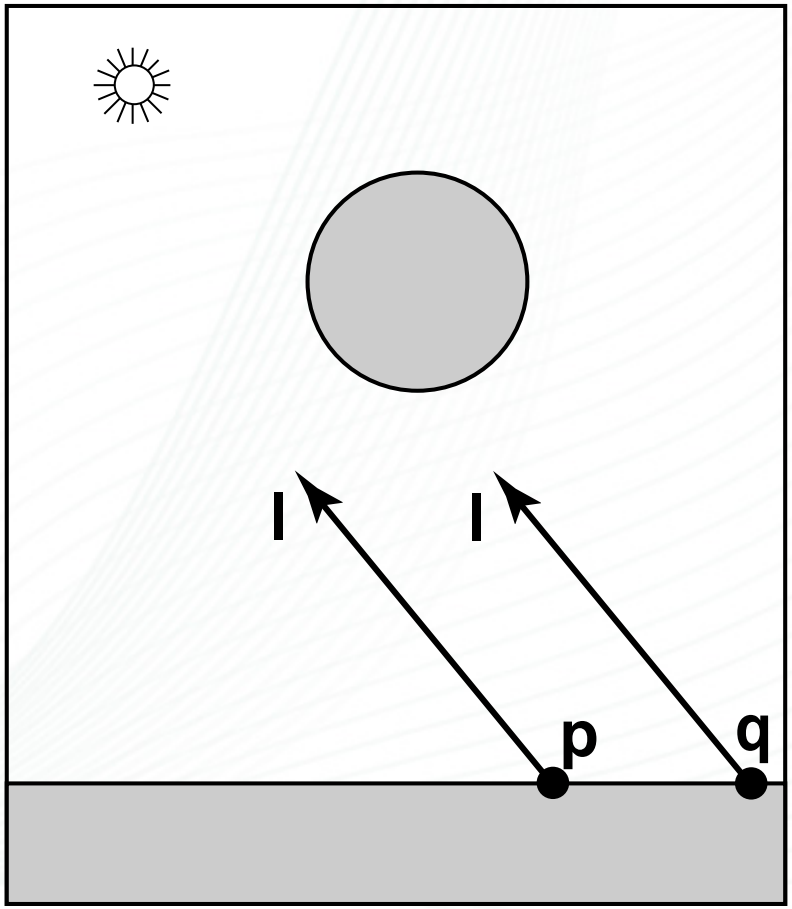
Sumário

- ~~Algoritmo básico de Ray Tracing~~
- ~~Perspectiva~~
- ~~Computação dos raios de visualização~~
- ~~Interseção raio-objeto~~
- ~~Sombreamento (Shading)~~
- ~~Um programa de Ray Tracing~~
- **Sombras**
- Reflexão especular ideal

Sombras (*Shadows*)

- Um objeto está “na sombra”, se ele está ocluso por um objeto.
- A luz tem uma direção **l**. Se nós nos imaginarmos em um ponto **p** da superfície que está tonalizada, o ponto estará em uma sombra se traçarmos um raio que tem origem em **p**, direção de **l**, e esse raio intercepta o objeto.
- Se não houver objetos entre **p** e a fonte de luz, então o objeto está visível, portanto, não está ocluso.

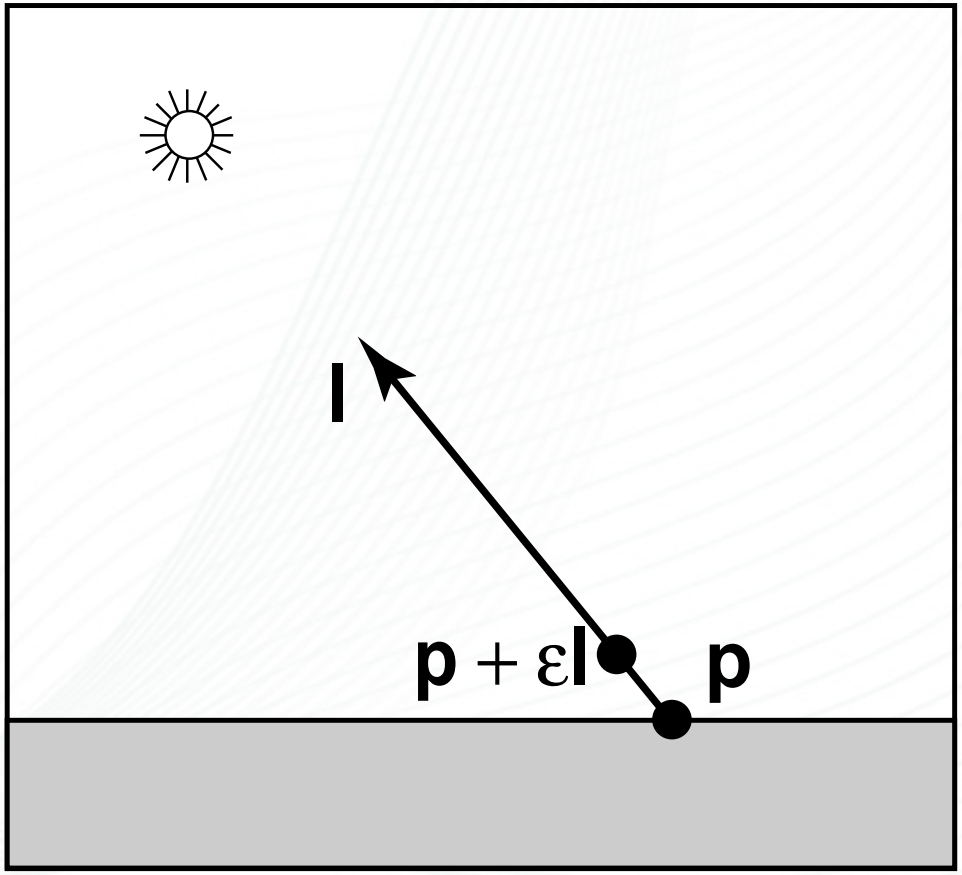
Sombras (*Shadows*)



Sombras (*Shadows*)

- Os raios que definem se um ponto está ocluso ou não são chamados de *shadow rays*.
- Para adicionarmos oclusão (*shadow*) no algoritmo, devemos inserir uma condição “if”.
- Em uma implementação ingênua, o teste do “if” verificaria **t** de zero até o infinito.
- Porém, verificou-se que, em alguns casos, o raio interceptaria a própria superfície onde o ponto **p** está localizado.
- Portanto, foi adicionado um baixo valor positivo constante (ε), para evitar que o raio intercepte a própria superfície.

Sombras (*Shadows*)



Sombras (*Shadows*)

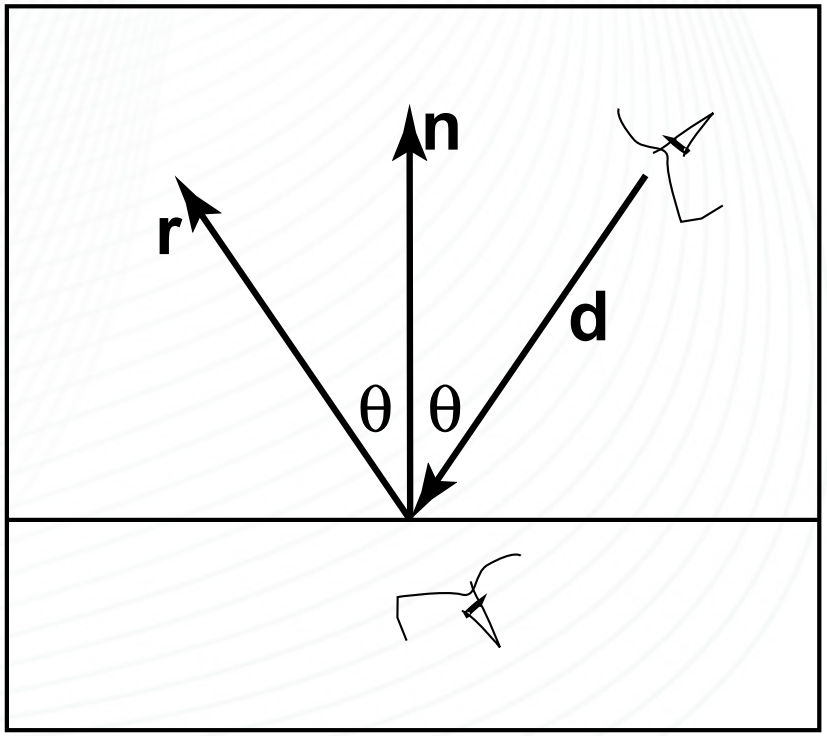
function raycolor(ray $\mathbf{e} + t\mathbf{d}$, real t_0, real t_1)
hit-record rec, srec
if (scene→hit($\mathbf{e} + t\mathbf{d}, t_0, t_1,$ rec)) **then**
    $\mathbf{p} = \mathbf{e} + (\text{rec}.t)\,\mathbf{d}$
    color $c = \text{rec}.k_a \;\; I_a$
    if (not scene→hit($\mathbf{p} + s\mathbf{l}, \epsilon, \infty,$ srec)) **then**
        vector3 $\mathbf{h} = \text{normalized}(\text{normalized}(\mathbf{l}) + \text{normalized}(-\mathbf{d}))$
        $c = c + \text{rec}.k_d \; I \max(0, \text{rec}.\mathbf{n} \cdot \mathbf{l}) + (\text{rec}.k_s) \; I \; (\text{rec}.\mathbf{n} \cdot \mathbf{h})^{\text{rec}.p}$
    return c
else
    return background-color

Sumário

- ~~Algoritmo básico de Ray Tracing~~
- ~~Perspectiva~~
- ~~Computação dos raios de visualização~~
- ~~Interseção raio-objeto~~
- ~~Sombreamento (Shading)~~
- ~~Um programa de Ray Tracing~~
- ~~Sombras~~
- **Reflexão especular ideal**

Reflexão especular ideal

- É bastante simples de se adicionar a reflexão especular ideal (reflexão de espelho) ao algoritmo de ray tracing.
- O observador que está em **e** vê o que está na direção **r**, por meio da superfície que reflete.



Reflexão especular ideal

- O vetor **r** é encontrado por meio da equação de iluminação e reflexão de Phong (Aulas 12 e 13).
- Há troca de sinal, pois o vetor **d** aponta por meio da superfície, então:

$$\mathbf{r} = \mathbf{d} - 2(\mathbf{d} \cdot \mathbf{n})\mathbf{n}$$

Reflexão especular ideal

- No mundo físico, alguma energia é perdida quando a luz é refletida em uma superfície e alguma perda pode ocorrer.
- Por exemplo: dourado pode ser refletido como amarelo.
- Isso pode ser implementado por uma chamada recursiva da função *raycolor*.

$$\text{color } c = c + k_m \text{raycolor}(\mathbf{p} + s\mathbf{r}, \epsilon, \infty)$$

Reflexão especular ideal

$$\text{color } c = c + k_m \text{raycolor}(\mathbf{p} + s\mathbf{r}, \epsilon, \infty)$$

onde k_m é a cor RGB.

- Assim como tivemos de realizar um teste para o raio de visualização, um teste semelhante deve ser feito para o raio de oclusão. Isso ocorre porque não queremos que o raio de oclusão intercepte a própria superfície reflexiva.

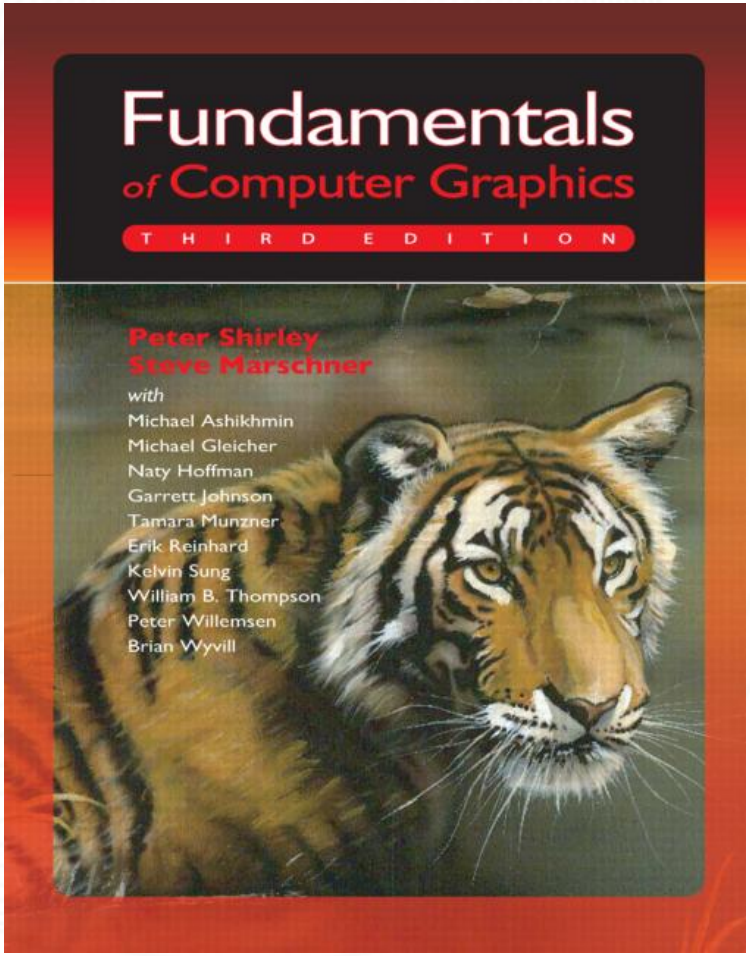
Sumário

- ~~Algoritmo básico de Ray Tracing~~
- ~~Perspectiva~~
- ~~Computação dos raios de visualização~~
- ~~Interseção raio-objeto~~
- ~~Sombreamento (Shading)~~
- ~~Um programa de Ray Tracing~~
- ~~Sombras~~
- ~~Reflexão especular ideal~~

Aula de hoje

Shirley, Peter, Michael Ashikhmin, and Steve Marschner. Fundamentals of computer graphics. CRC Press, 3rd Edition, 2009.

- **Capítulo 4**



Fim da Aula 08

André Luiz Brandão