Modelos de Iluminação

SCC0250 - Computação Gráfica

Profa. Maria Cristina F. Oliveira

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)

4 de novembro de 2023

Sumário

1. Introdução
2. Fontes de Luz
3. Efeitos de Luz em Superfícies
4. Modelos Básicos de Iluminação
   - Programação OpenGL (Iluminação)
5. Superfícies Transparentes
   - Programação OpenGL (Transparência)

Sumário

1. Introdução
2. Fontes de Luz
3. Efeitos de Luz em Superfícies
4. Modelos Básicos de Iluminação
   - Programação OpenGL (Iluminação)
5. Superfícies Transparentes
   - Programação OpenGL (Transparência)

Introdução

- Imagens **realísticas** são criadas a partir da descrição da cena após as alterações introduzidas na geometria pela projeção perspectiva e mapeamento para o volume canônico
- Tenta-se simular os efeitos de uma ou mais fontes de iluminação às superfícies visíveis por meio de um **modelo de iluminação** (*shading model*)

- Na realidade, modelar os **efeitos da luz** sobre a superfície um objeto é um processo **complexo**, que envolve princípios físicos e psicofísicos
- Modelos simples de iluminação: **simplificam muito** o que de fato ocorre
- Modelos empíricos vs modelos físicos

Introdução

- Os modelos **físicos** de iluminação consideram múltiplos fatores, como propriedades dos **materiais**, **posição** do objeto em relação à fonte de luz e aos outros objetos, e propriedades das **fontes de luz**
  - Um material pode ser opaco ou (parcialmente) transparente, pode ser lisos ou rugoso
  - Fontes de luz podem ter diferentes formatos, cores e posições

- Os modelos de iluminação utilizados no pipeline gráfico são, em geral, modelos empíricos
- Ou seja, **aproximações** *ad hoc* que permitem replicar os efeitos da luz sobre as superfícies, sem de fato computar as leis físicas que descrevem esses efeitos

Sumário

1 Introdução

2 Fontes de Luz

3 Efeitos de Luz em Superfícies

4 Modelos Básicos de Iluminação
- Programação OpenGL (Iluminação)

5 Superfícies Transparentes
- Programação OpenGL (Transparência)

Fontes de Luz

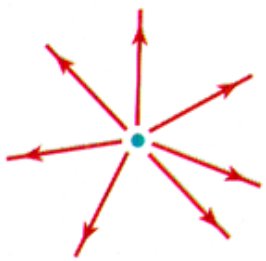
- Qualquer objeto que emite energia luminosa é uma **fonte de luz** que contribui para a iluminação dos outros objetos na cena e efeitos decorrentes

- **Fontes de luz** podem assumir diferentes **formas** e **propriedades** (posição, cor, direção de emissão, etc.), podendo emitir luz (emissora) ou refletir luz (refletora)
  - P.ex., um lustre de vidro redondo ao redor de uma lâmpada tanto emite como reflete luz

- Em aplicações gráficas em **tempo real**, normalmente aplica-se um modelo simples de iluminação
- O **custo computacional** de aplicar um modelo físico é alto demais para tempo real
  - Propriedades da emissão de luz são definidas considerando cada componente de cor RGB, descrevendo suas "intensidades"

Fontes de Luz Puntuais

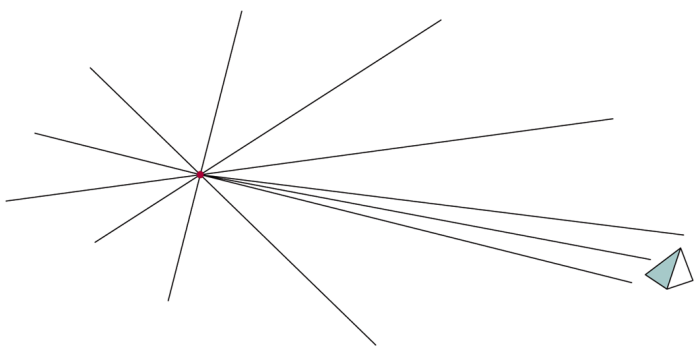
- O modelo de emissão de luz mais simples é o da **fonte de luz puntual**
  - Assume que a luz emitida tem uma única cor, em geral branca
  - Fonte é especificada em termos da sua **posição** e da **cor** da luz que emite



- Fonte gera 'raios´ de luz em **direções radiais divergentes** a partir da sua posição
- Fonte representada como um ponto de luz emissor
  - Modelo 'aproxima´ os efeitos de iluminação observados no mundo real, quando se tem uma **fonte de luz muito pequena** relativamente ao tamanho dos objetos da cena
  - P. ex., o sol!!

Fontes de Luz no Infinito

- Uma fonte de luz grande, mas posicionada muito distante da cena (p.ex., o sol) pode ser aproximada como um ponto emissor
  - Nesse caso, considera-se que a iluminação chega em uma única direção
  - Como se a fonte emitisse luz em uma única direção



- Essa fonte de luz distante é simulada definindo a **cor** da luz emitida e a **direção** de emissão dos raios
- Não é necessário especificar uma posição
- Esse tipo de fonte é chamada de (**fonte direcional**)

Atenuação Radial da Intensidade

- A energia de radiação emitida por uma fonte de luz posicionada a uma distância d_l da cena tem sua **amplitude atenuada** por um fator $1/d_l^2$
  - Uma superfície mais próxima da fonte recebe mais intensidade de luz do que uma mais distance
  - Esse efeito de atenuação precisa ser considerado para gerar uma iluminação mais realística

- Entretanto, nos modelos empíricos com fontes de luz puntuais os efeitos de adotar uma atenuação de $1/d_l^2$ não são muito realísticos
  - Tende a introduzir uma variação muito grande da intensidade quando d_l é pequeno (objetos próximos da fonte) e uma variação muito pequena quando d_l é grande

Atenuação Radial da Intensidade

- Para produzir efeitos mais realísticos com fontes puntuais adota-se

$$f_{radatten}(d_l) = \frac{1}{a_0 + a_1 d_l + a_2 d_l^2}$$

- Em que a_0, a_1 e a_2 são constantes cujos valores podem ser ajustados para produzir os efeitos desejados
  - Pode-se atribuir valores altos a a_0 quando d_l é muito pequeno, evitando que o valor computado por $f_{radatten}(d_l)$ seja muito grande

Atenuação Radial da Intensidade

- Não é possível aplicar esse cálculo de atenuação quando a fonte puntual está no "infinito", porque nesse caso a distância à fonte é indeterminada

- Outro ponto é que nesse caso todos as superfícies da cena estão praticamente a uma mesma distância da fonte de luz
- Isso é bem diferente do que ocorre de fato: mesmo para uma fonte distante, os objetos estão a diferentes distâncias da fonte

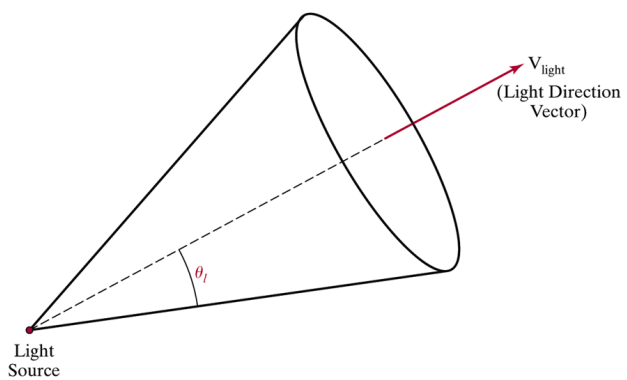
- Para tratar essas questões, faz-se uma adaptação no modelo empírico

$$f_{l,radatten} = \begin{cases} 1.0, & \text{se a fonte está no infinito} \\ \frac{1}{a_0 + a_1 d_l + a_2 d_l^2}, & \text{se a fonte é local} \end{cases}$$

Fontes de Luz Direcional e Efeitos de Holofote

- Pode-se modificar o modelo da fonte de luz local para produzir o efeito de uma **fonte direcional**, ou holofote
  - Se um objeto está fora dos limites direcionais ele não recebe iluminação dessa fonte

- Pode-se definir uma **fonte de luz direcional** especificando sua **posição**, a **cor** da luz emitida, um vetor de **direção** e um **limite angular** θ_l a partir desse vetor

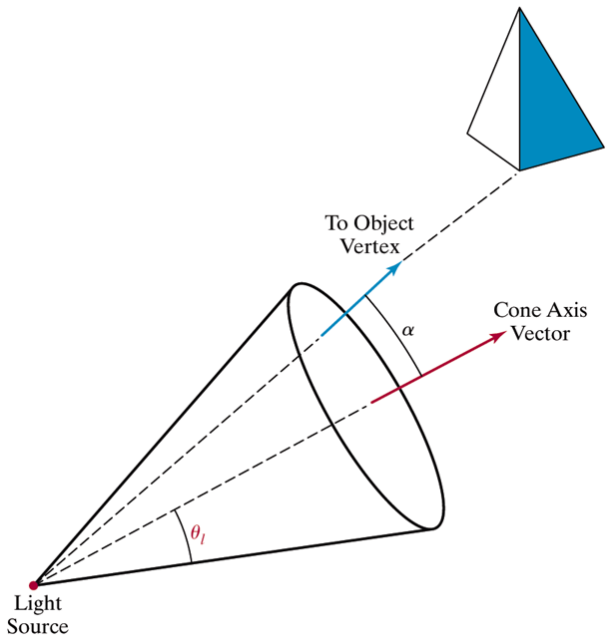


Fonte de Luz Direcional e Efeitos de Holofote

- Podemos especificar dois vetores:
- Um vetor unitário $\mathbf{V}_{light}$ dá a direção central de emissão de luz pela fonte (a partir da qual é definido o limite angular θ_l)
- Um vetor unitário $\mathbf{V}_{obj}$ dá a direção de emissão a partir da posição da luz até a superfície do objeto

- Considerando que $\cos\alpha = \mathbf{V}_{light} \cdot \mathbf{V}_{obj}$ e limitando $0^0 \leq \theta_l \leq 90^0$, um ponto está dentro da região que recebe luz da fonte se $\cos\alpha \geq \cos\theta_l$

Fonte de Luz Direcional e Efeitos de Holofote



Atenuação Angular de Intensidade

- Nesse tipo de fonte pode-se considerar que a energia emitida é atenuada tanto **radialmente** a partir da sua posição, mas também **angularmente**
  - Introduzir o efeito de atenuação angular permite simular melhor o efeito de holofote (cones de luz)
  - A emissão acontece com mais intensidade ao longo do eixo do cone
- Para modelar esse efeito pode-se usar a função

$$f_{angatten}(\phi) = \cos^{a_l} \phi,\ 0^0 \leq \phi \leq \theta$$

- A constante $a_l > 0$ define um expoente de atenuação
- ϕ é o ângulo medido a partir do eixo do cone
  - Ao longo do eixo do cone $\phi = 0^0$ e $f_{angatten}(\phi) = 1.0$
  - Quanto maior o valor da constante a_l, menor o valor de $f_{angatten}(\phi)$, dado $\phi > 0^0$

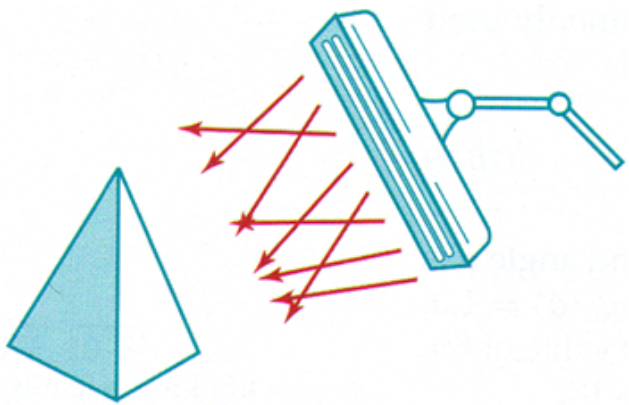
Atenuação Angular de Intensidade

- Considerando os vetores $\mathbf{V}_{light}$ e $\mathbf{V}_{obj}$ e assumindo $0^0 < \theta_l \leq 90^0$ a equação geral de atenuação angular é definida por

$$f_{l,angatten} = \begin{cases} 1.0, & \text{se a fonte de luz não é direcional} \\ 0.0, & \text{se } \mathbf{V}_{light} \cdot \mathbf{V}_{obj} = \cos\alpha < \cos\theta_l \\ & \text{(o objeto está fora do cone de luz)} \\ (\mathbf{V}_{light} \cdot \mathbf{V}_{obj})^{a_l}, & \text{caso contrário} \end{cases}$$

Fontes de Luz Estendidas e o Modelo de Warn

- As fontes de luz reais não são puntuais
- Pode-se simular os efeitos de uma fonte que tem uma área grande e está posicionada próxima aos objetos da cena como uma superfície que emite luz



- Isso pode ser feito modelando a superfície como uma *grade* de múltiplas fontes direcionais

Sumário

1 Introdução

2 Fontes de Luz

3 Efeitos de Luz em Superfícies

4 Modelos Básicos de Iluminação
- Programação OpenGL (Iluminação)

5 Superfícies Transparentes
- Programação OpenGL (Transparência)

Efeitos de Luz em Superfícies

- Um modelo de iluminação computa os **efeitos da luz** sobre uma superfície considerando várias **propriedades óticas** do material

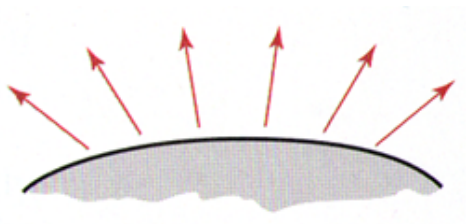
- Quando o material da superfície é **opaco**, parte da luz que incide é **refletida** e parte é absorvida
  - A quantidade de luz refletida depende do material
  - Existem dois tipos de reflexão: reflexão difusa e reflexão especular

- Se o material da superfície é **transparente**, parte da luz incidente também é **transmitida** através do material

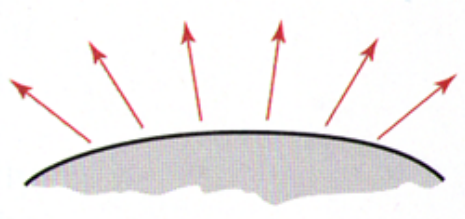
Efeitos de Luz em Superfícies

Reflexão Difusa

- Materiais com superfícies irregulares tendem a refletir a luz em todas as direções
- Parecem igualmente brilhantes a partir de qualquer ponto de observação



Efeitos de Luz em Superfícies

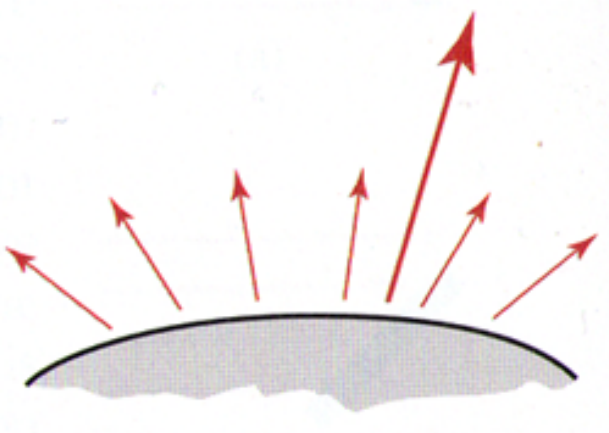


- A reflexão difusa determina a cor percebida de um material: p.ex. um objeto iluminado por luz branca vai ter a cor da luz que é refletida pela reflexão difusa
  - Um material é percebido como azul porque reflete a componente azul da luz branca
  - Um objeto de material azul iluminado com luz vermelha seria percebido como preto
    - a luz incidente só tem energia na faixa vermelha do espectro, o material só reflete a luz na faixa azul do espectro (que no caso não está presente), portanto ele não reflete
    - Se o objeto não reflete nenhuma energia luminosa, ele é percebido como preto.

Efeitos de Luz em Superfícies

Reflexão Especular

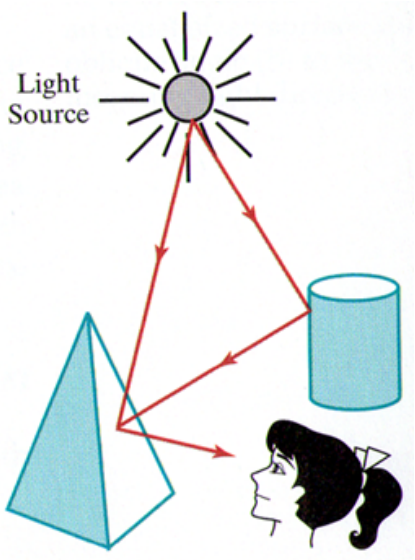
- Em materiais mais polidos, a reflexão da luz é concentrada em uma região
- Isso determina a percepção do material como sendo mais brilhante



Efeitos de Luz em Superfícies

Luz Ambiente

- Um ambiente tem uma iluminação distribuída, que introduz uma luz de fundo, ou luz ambiente
- Esse efeito é resultado da luz refletida pelas múltiplas superfícies na cena
- - A luz total refletida por uma superfície é a soma das contribuições da luz refletida pelas demais superfícies, inclusive a fonte



Sumário

1 Introdução

2 Fontes de Luz

3 Efeitos de Luz em Superfícies

4 Modelos Básicos de Iluminação
- Programação OpenGL (Iluminação)

5 Superfícies Transparentes
- Programação OpenGL (Transparência)

Modelos Básicos de Iluminação

- **Modelos físicos** de iluminação computam toda a **interação** entre a **radiação luminosa** e o **material** dos objetos
- Simular os efeitos dessa **interação** é, normalmente, um processo **computacionalmente muito caro**

- Os **modelos empíricos** buscam aproximar o efeito visual resultante, sem fazer cálculos complexos
- É possível obter bons resultados com estratégias *ad hoc*

Luz Ambiente

- Uma estratégia para incorporar o efeito da **luz ambiente** consiste em definir uma **intensidade luminosa global** para a cena
  - Gera uma luz ambiente uniforme, que tem o mesmo efeito em todos os objetos, aproximando o efeito do conjunto das reflexões difusas geradas por todas superfícies da cena
- A quantidade (intensidade) de luz refletida dependerá das propriedades óticas dos materiais de que são feitos as superfícies

- A intensidade de luz ambiente em uma cena é definida por um parâmetro I_a (constante)

Reflexão Difusa

- O efeito da **reflexão difusa** pode ser modelado assumindo que a luz incidente é **espalhada com igual intensidade** em todas as direções, independente da direção de observação
  - Materiais que apresentam esse comportamento são chamados de **refletores difusos ideais** (refletores Lambertinianos)

Reflexão Difusa

- Assumindo que toda superfície é um refletor difuso ideal (Lambertiniano), pode-se utilizar um parâmetro k_d (**coeficiente de reflexão difusa**) para determinar a fração da luz incidente que será refletida difusamente
  - A reflexão difusa é constante, proporcional à luz incidente multiplicada por k_d

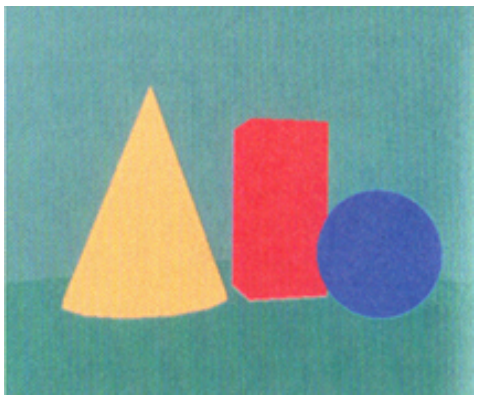
- Para fontes de luz monocromáticas, $0.0 \leq k_d \leq 1.0$
  - Materiais brilhantes apresentam valores de k_d altos – a intensidade da reflexão é próxima da intensidade da luz incidente
  - Valores baixos de k_d definem materiais que absorvem a maior parte da luz incidente, e refletem pouco

Reflexão Difusa

- Para incorporar os efeitos da luz distribuída no ambiente, todas as superfícies são iluminadas pela luz ambiente I_a definida para a cena, e a contribuição dessa iluminação para reflexão difusa é dada por

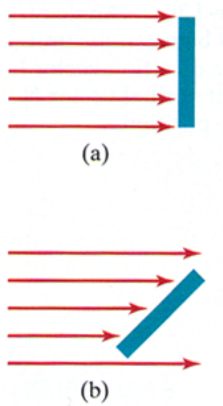
$$I_{ambdiff} = k_d \cdot I_a$$

- Considerando somente a luz ambiente, o efeito de iluminação é pouco interessante



Reflexão Difusa

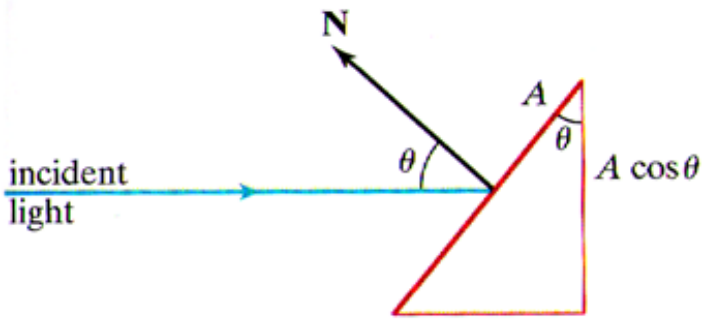
- Quando uma superfície é iluminada com intensidade I_l, a quantidade de **luz incidente** depende de como a **superfície está orientada** relativamente à direção da luz
- Quando a superfície está posicionada perpendiculamente à direção da luz incidente, ela recebe mais quantidade de luz



- A quantidade de luz que atinge uma superfície é proporcional à área da superfície projetada na direção perpendicular à direção da luz incidente

Reflexão Difusa

- Seja **ângulo** θ entre a direção da luz incidente e a normal à superfície



- A área projetada é proporcional à $\cos\theta$
- Ou seja, a intensidade da **iluminação** recebida diretamente da fonte de luz **diminui** conforme o **ângulo com que a luz incide na superfície aumenta**,

Reflexão Difusa

- Suponha que a cena inclui uma fonte de luz que emite com intensidade I_l
- Podemos modelar a intensidade da luz dessa fonte incidente a uma superfície como

$$I_{l,incident} = I_l \ \cos\theta$$

Reflexão Difusa

- Para uma superfície de um determinado material, podemos modelar a intensidade da luz que ela reflete difusamente como

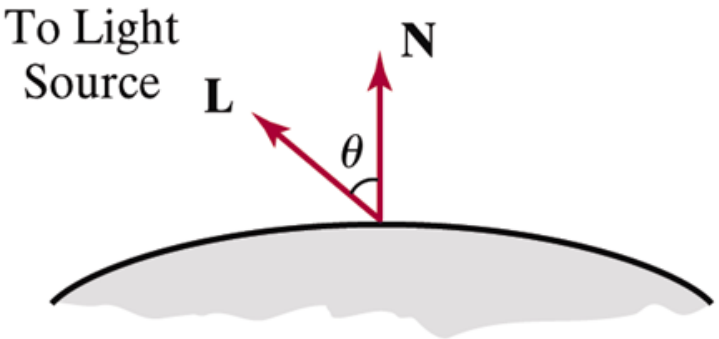
$$I_{l,diff} = k_d \; I_{l,incident}$$

$$I_{l,diff} = k_d \; I_l \; \cos\theta$$

- A superfície só vai ser iluminada pela luz de uma fonte quando $0^0 \leq \theta \leq 90^0$
- Se $\theta \geq 90^0$ $(\cos\theta < 0.0)$ a fonte de luz está posicionada atrás da superfície, e a iluminação não incide sobre a face visível da mesma

Reflexão Difusa

- Considerando $\mathbf{N}$ como o vetor unitário normal à superfície e $\mathbf{L}$ o vetor unitário que dá a direção da luz, então $\cos\theta = \mathbf{N} \cdot \mathbf{L}$



- E a equação de reflexão difusa para uma única fonte de luz puntual é dada por

$$I_{l,diff} = \begin{cases} k_d \; I_l(\mathbf{N} \cdot \mathbf{L}), & \text{se } \mathbf{N} \cdot \mathbf{L} > 0 \\ 0.0, & \text{se } \mathbf{N} \cdot \mathbf{L} \leq 0 \end{cases}$$

Reflexão Difusa

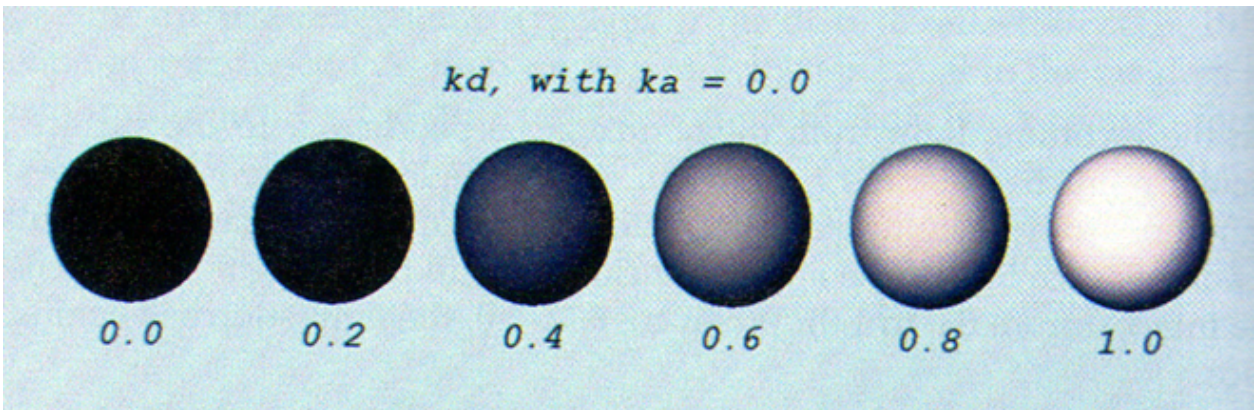
- Pode-se calcular o vetor unitário $\mathbf{L}$ em função da posição do ponto no qual está sendo calculada a iluminação (um vértice em uma superfície) e da posição da fonte de luz na cena

$$\mathbf{L} = \frac{\mathbf{P}_{source} - \mathbf{P}_{surf}}{|\mathbf{P}_{source} - \mathbf{P}_{surf}|}$$

- Uma fonte de luz no "infinito" não tem posição, nesse caso define-se uma direção de emissão
  - Emprega-se o negativo da direção de emissão para definir a direção do vetor de iluminação $\mathbf{L}$

Reflexão Difusa

- Exemplo do resultado de aplicar a equação de iluminação difusa, com valores da constante k_d variando no intervalo $[0, 1]$
  - Uma única fonte de luz puntual
  - Sem componente de luz ambiente



Reflexão Difusa

- Podemos associar os cálculos da luz ambiente e da fonte de luz puntual para determinar a reflexão difusa em uma posição da superfície
- Vamos introduzir o **coeficiente de reflexão ambiente** k_a para cada superfície, o que permite variar a intensidade I_a da luz ambiente

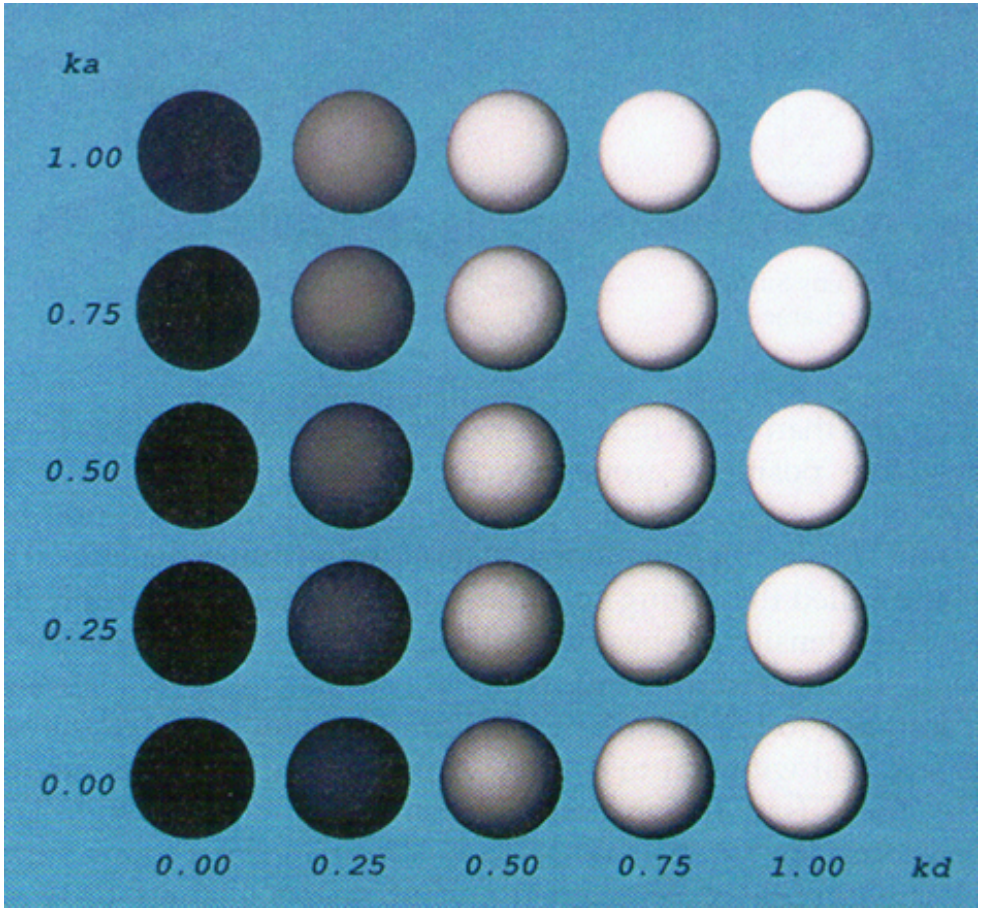
- Podemos escrever a equação de reflexão difusa total devida à luz ambiente e à luz emitida por uma única fonte puntual como

$$I_{diff} = \begin{cases} k_a\ I_a + k_d\ I_l(\mathbf{N} \cdot \mathbf{L}), & \text{se } \mathbf{N} \cdot \mathbf{L} > 0 \\ k_a\ I_a, & \text{se } \mathbf{N} \cdot \mathbf{L} \leq 0 \end{cases}$$

- em que k_a e k_d são constantes, usadas para modelar as propriedades do material da superfície

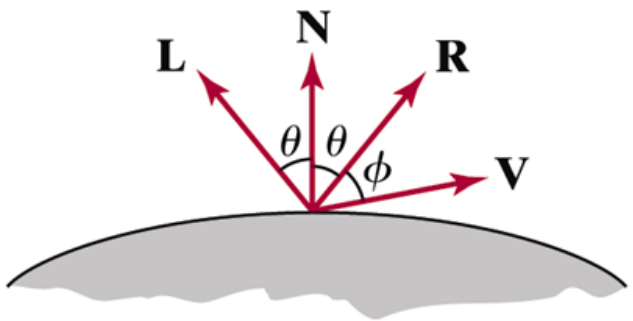
Reflexão Difusa

- Exemplo de iluminação difusa variando os parâmetros k_d e k_a



Reflexão Especular e Modelo de Phong

- O efeito da **reflexão especular** em uma superfície brilhante resulta da reflexão total (ou quase) da luz incidente em uma **área concentrada** ao redor de um **ângulo de reflexão especular**

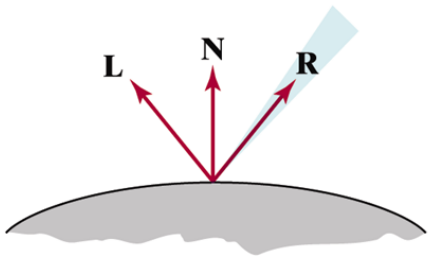


- O ângulo de reflexão especular θ é igual ao ângulo de incidência da luz, definido em função da normal à superfície $\mathbf{N}$, mas na direção simétrica
- O vetor unitário $\mathbf{R}$ dá a direção de reflexão especular
- O vetor unitário $\mathbf{L}$ dá a direção da fonte de luz puntual
- O vetor unitário $\mathbf{V}$ dá a direção do observador

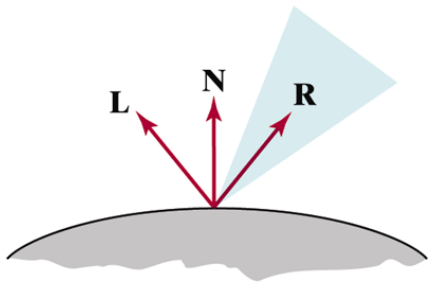
Reflexão Especular e Modelo de Phong

- Um material refletor ideal (espelho) reflete toda a luz incidente na direção de reflexão especular. O reflexo especular será visível somente quando $\mathbf{V}$ e $\mathbf{R}$ coincidem ($\phi = 0^0$)

- Materiais que não são refletores ideais exibem reflexão especular em uma região finita ao redor de $\mathbf{R}$
  - Superfícies brilhantes apresentam reflexão especular mais concentrada espacialmente
  - Superfícies foscas apresentam reflexão especular mais espalhada espacialmente



Shiny Surface
(Large n_s)

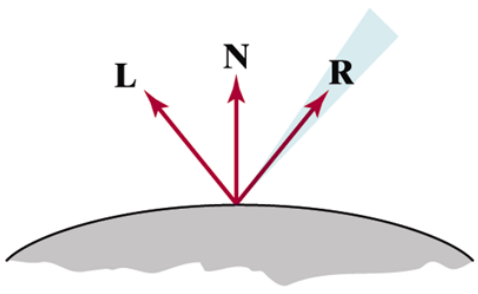


Dull Surface
(Small n_s)

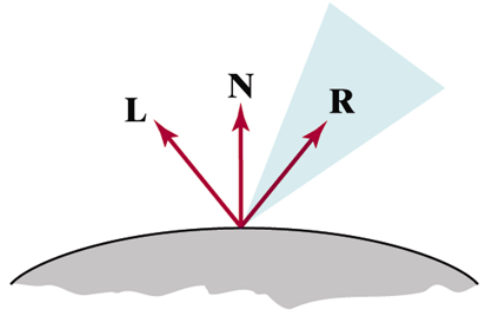
Reflexão Especular e Modelo de Phong

- O **Modelo de Phong** define a intensidade da reflexão especular como sendo proporcional a $\cos^{n_s} \phi$, com $0^0 \leq \phi \leq 90^0$

- O **expoente de reflexão especular** é uma constante n_s usada para determinar o comportamento da reflexão especular da superfície
  - Valores mais altos de n_s (p. ex. ≥ 100) simulam o comportamento de superfícies mais brilhantes
  - Para materiais refletores ideais $n_s \to \infty$



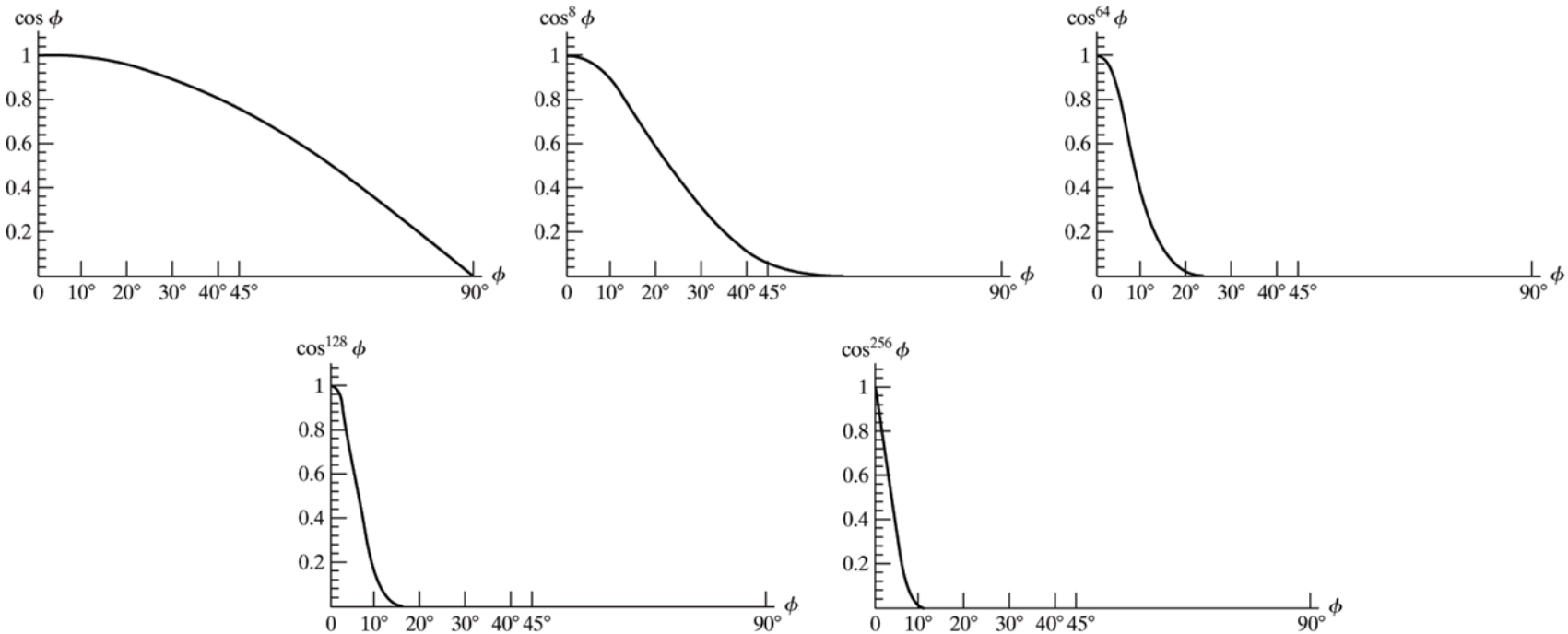
Shiny Surface
(Large n_s)



Dull Surface
(Small n_s)

Reflexão Especular e Modelo de Phong

- Curvas de $\cos^{n_s}(\phi)$ versus ϕ considerando diferentes valores do expoente especular n_s

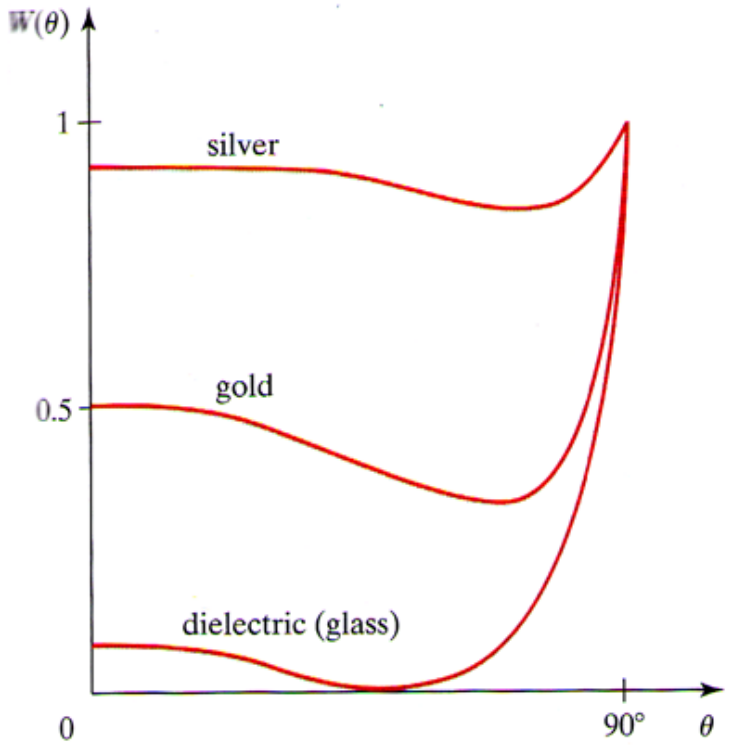


Reflexão Especular e Modelo de Phong

- A **intensidade da reflexão** especular depende das propriedades dos **materiais** e também do ângulo com que a luz incide na supefície (θ)
- Pode-se aproximar as variações da intensidade da reflexão especular utilizando um **coeficiente de reflexão especular** $W(\theta)$

- A variação da intensidade da reflexão especular é descrita pela **Lei de Reflexão de Fresnel**

Reflexão Especular e Modelo de Phong



- Para a maioria dos materiais, $W(\theta)$ aumenta quando o ângulo de incidência aumenta

Reflexão Especular e Modelo de Phong

- O modelo de Phong de reflexão especular pode ser escrito em termos da função $W(\theta)$ como

$$I_{l,spec} = W(\theta)\ I_l\ \cos^{n_s}\phi$$

- Em que I_l é a intensidade da fonte de luz e ϕ é o ângulo entre os vetores $\mathbf{V}$ (direção de observação) e $\mathbf{R}$ (direção da reflexão especular)

Reflexão Especular e Modelo de Phong

- Para a maioria dos materiais opacos, a reflexão especular é quase constante para todos os ângulos de incidência (θ)
  - Assim, substitui-se a função $W(\theta)$ por coeficiente de reflexão especular constante k_s
- O mesmo não ocorre para materiais transparentes

Reflexão Especular e Modelo de Phong

- Como $\mathbf{V}$ e $\mathbf{R}$ são vetores unitários, então $\cos\phi = \mathbf{V} \cdot \mathbf{R}$
- Se $\mathbf{V}$ e $\mathbf{L}$ estiverem do mesmo lado da normal $\mathbf{N}$ não é necessário calcular os efeitos especulares (o observador não vai percebê-los)

- Assumindo que o coeficiente de reflexão especular é constante para qualquer material, podemos calcular

$$I_{l,spec} = \begin{cases} k_s \; I_l \; (\mathbf{V} \cdot \mathbf{R})^{n_s}, & \text{se } \mathbf{V} \cdot \mathbf{R} > 0 \\ 0.0, & \text{se } \mathbf{V} \cdot \mathbf{R} \leq 0 \end{cases}$$

- A direção de $\mathbf{R}$ pode ser obtida a partir de $\mathbf{L}$ e $\mathbf{N}$

Reflexão Especular e Modelo de Phong

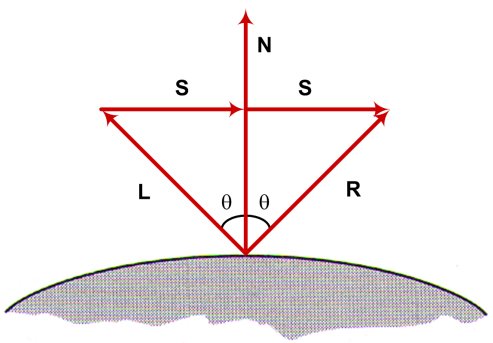
- A projeção do vetor $\mathbf{L}$ na direção do vetor $\mathbf{N}$ é $(\mathbf{N}\ \cos\theta)$ (projeção escalar), então

$$\begin{aligned} \mathbf{R} - \mathbf{S} &= \mathbf{N}\cos\theta \\ \mathbf{R} &= \mathbf{N}\cos\theta + \mathbf{S} \end{aligned}$$

$$\begin{aligned} \mathbf{L} + \mathbf{S} &= \mathbf{N}\cos\theta \\ \mathbf{S} &= \mathbf{N}\cos\theta - \mathbf{L} \end{aligned}$$

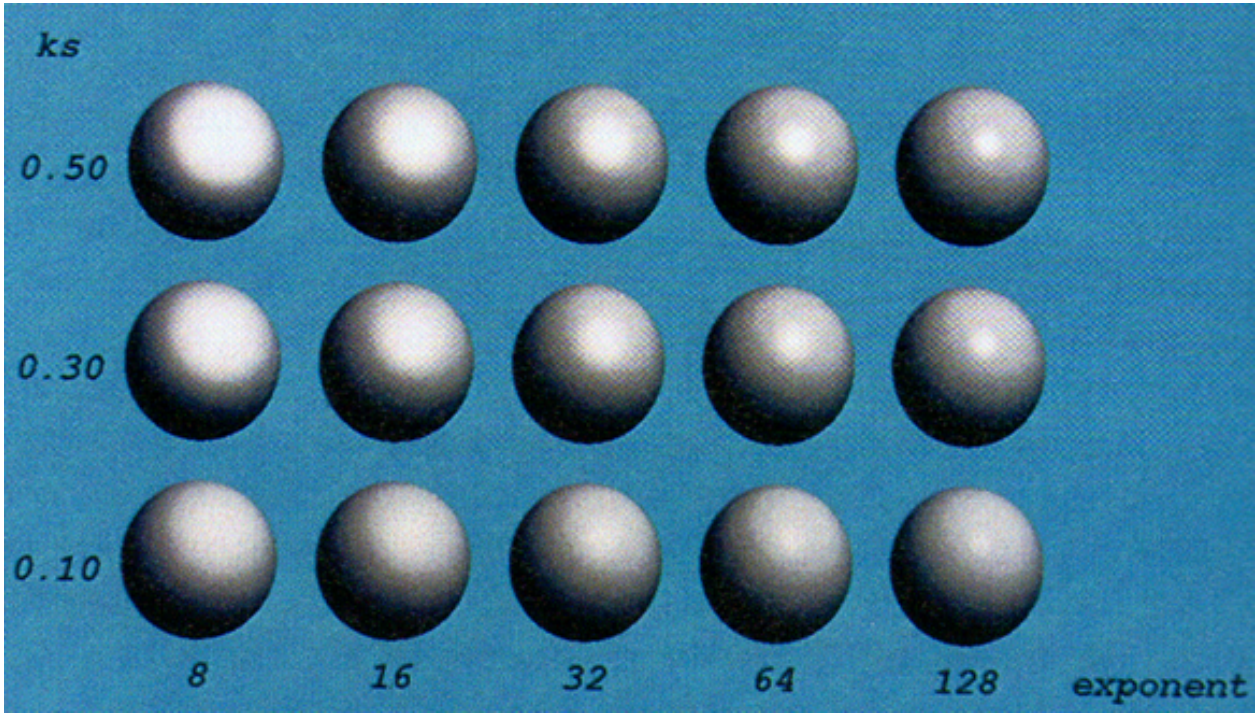
- De forma que o vetor de reflexão especular é obtido fazendo

$$\begin{aligned} \mathbf{R} &= \mathbf{N}\cos\theta + \mathbf{N}\cos\theta - \mathbf{L} \\ &= \mathbf{N}\ 2\cos\theta - \mathbf{L} \\ &= \mathbf{N}(2\mathbf{N}\cdot\mathbf{L}) - \mathbf{L} \end{aligned}$$



Reflexão Especular e Modelo de Phong

- Exemplos de reflexão especular variando os parâmetros k_s e n_s em uma superfície esférica iluminada por uma única fonte de luz puntual



Reflexão Especular e Modelo de Phong

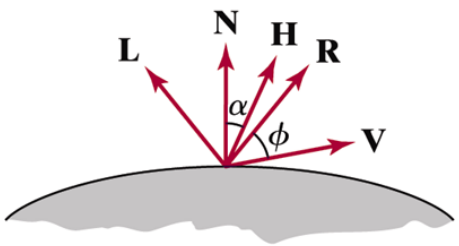
- O vetor $\mathbf{L}$ é dado em função dos parâmetros da cena, o vetor $\mathbf{V}$ é calculado em função da posição do observador e da posição do ponto na superfície para o qual está sendo calculado o modelo de iluninação

- Pode-se considerar **uma direção de observação** única para todos os pontos (vértices)
- Isso acelera cálculo da iluminação especular (porque?)
  - Porém, a qualidade visual da cena renderizada é pior

Reflexão Especular e Modelo de Phong

- Uma versão simplificada do modelo de Phong pode ser obtida usando o **vetor intermediário** $\mathbf{H}$ entre $\mathbf{L}$ e $\mathbf{V}$

$$\mathbf{H} = \frac{\mathbf{L} + \mathbf{V}}{|\mathbf{L} + \mathbf{V}|}$$



- O cálculo $\mathbf{N} \cdot \mathbf{H}$ pode ser usado como uma aproximação para $\mathbf{V} \cdot \mathbf{R}$
- Se $\mathbf{V}$ for coplanar a $\mathbf{L}$ e $\mathbf{R}$, tem-se $\alpha = \phi/2$
- Substitui-se o cálculo de $\cos\phi$ pelo de $\cos\alpha$

Reflexão Especular e Modelo de Phong

- Para **superfícies não planares**, é mais simples calcular $\mathbf{N} \cdot \mathbf{H}$ do que $\mathbf{V} \cdot \mathbf{R}$ porque o cálculo de $\mathbf{R}$ em cada ponto da superfície envolve o vetor $\mathbf{N}$

- Se tanto o **observador como a fonte de luz estiverem muito distantes** da superfície, $\mathbf{V}$ e $\mathbf{L}$ são constantes, e $\mathbf{H}$ **é constante** para todos os pontos da superfície

- $\mathbf{H}$ é a direção em que a reflexão especular da luz pela superfície é máxima na direção de observação

Reflexão Especular e Difusa Combinadas

- Para uma única fonte de luz puntual, podemos modelar a combinação das reflexões difusa e especular como

$$\begin{aligned} I &= I_{diff} + I_{spec} \\ &= (k_a\ I_a + k_d\ I_l\ (\mathbf{N} \cdot \mathbf{L})) + k_s\ I_l\ (\mathbf{N} \cdot \mathbf{H})^{n_s} \end{aligned}$$

Reflexão Especular e Difusa Combinadas

- Para uma única fonte de luz puntual, podemos modelar a combinação das reflexões difusa e especular como

$$\begin{aligned} I &= I_{diff} + I_{spec} \\ &= (k_a\ I_a + k_d\ I_l\ (\mathbf{N} \cdot \mathbf{L})) + k_s\ I_l\ (\mathbf{N} \cdot \mathbf{H})^{n_s} \end{aligned}$$

- A superfície será iluminada somente pela luz ambiente quando a fonte de luz estiver atrás da superfície ($\mathbf{N} \cdot \mathbf{L} \leq 0$)

Reflexão Especular e Difusa Combinadas

- Para uma única fonte de luz puntual, podemos modelar a combinação das reflexões difusa e especular como

$$\begin{aligned} I &= I_{diff} + I_{spec} \\ &= (k_a\ I_a + k_d\ I_l\ (\mathbf{N} \cdot \mathbf{L})) + k_s\ I_l\ (\mathbf{N} \cdot \mathbf{H})^{n_s} \end{aligned}$$

- A superfície será iluminada somente pela luz ambiente quando a fonte de luz estiver atrás da superfície ($\mathbf{N} \cdot \mathbf{L} \leq 0$)

- Não ocorrerão efeitos especulares se $\mathbf{V}$ e $\mathbf{L}$ estiverem do mesmo lado de $\mathbf{N}$ ($\mathbf{N} \cdot \mathbf{H} \leq 0$)

Reflexão Especular e Difusa Combinadas

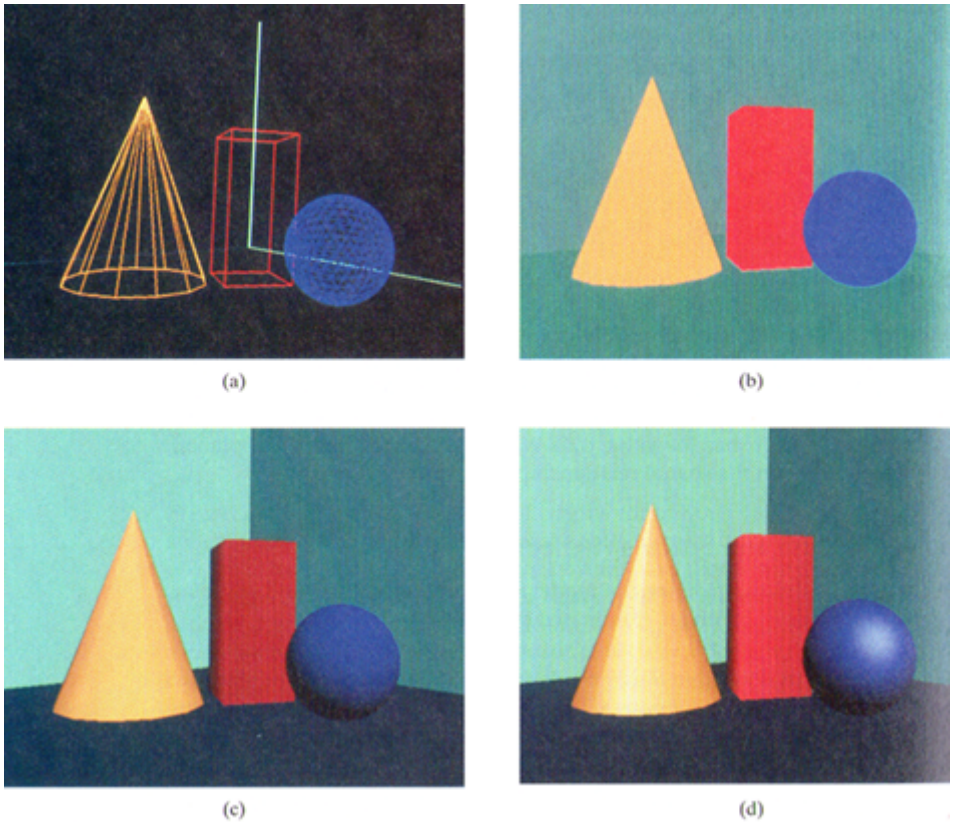


Figura: Cena wireframe (a), somente luz ambiente (b), reflexão difusa com luz ambiente e uma única fonte de luz puntual (c), reflexão difusa e especular com luz ambiente e uma única fonte de luz puntual (d).

Reflexões Difusa e Especular de Múltiplas Fontes de Luz

- É possível considerar múltiplas fontes de luz, basta somar as contribuições de reflexão difusa e especular devidas a cada fonte

$$
\begin{aligned}
I &= I_{ambdiff} + \sum_{l=1}^{n} [I_{l,diff} + I_{l,spec}] \\
&= k_a \; I_a + \sum_{l=1}^{n} I_l [k_d(\mathbf{N} \cdot \mathbf{L}) + k_s(\mathbf{N} \cdot \mathbf{H})^{n_s}]
\end{aligned}
$$

Superfícies Emissoras de Luz

- Em uma cena, algumas **superfícies** podem **emitir** e **refletir** luz
  - Esse efeito pode ser modelado adicionando um termo de emissão $I_{surfemission}$ à iluminação de fontes de luz diferentes e à iluminação ambiente

- Podemos **simular** esse efeito colocando uma **fonte de luz atrás da superfície** ou posicionando um conjunto de pontos de luz sobre uma superfície
  - Na prática, superfícies emissoras de luz são pouco usadas devido ao **custo computacional alto**

Modelo de Iluminação Básico com Atenuação da Intensidade e Holofotes

- O modelo de iluminação monocromático geral pode ser modelado como

$$I = I_{surfemission} + I_{ambdiff} + \sum_{l=1}^{n} f_{l,radatten}\ f_{l,angatten}(I_{l,diff} + I_{l,spec})$$

Modelo de Iluminação Básico com Atenuação da Intensidade e Holofotes

- Em que

$$I_{l,diff} = \begin{cases} 0.0, & \text{se } \mathbf{N} \cdot \mathbf{L}_l \leq 0.0 \\ & \text{(fonte de luz atrás do objeto)} \\ k_d \; I_l(\mathbf{N} \cdot \mathbf{L}_l), & \text{caso contrário} \end{cases}$$

- e

$$I_{l,spec} = \begin{cases} 0.0, & \text{se } \mathbf{N} \cdot \mathbf{H}_l \leq 0 \\ & \text{(fonte de luz atrás do objeto)} \\ k_s \; I_l \; \max\{0.0, (\mathbf{N} \cdot \mathbf{H}_l)^{n_s}\}, & \text{caso contrário} \end{cases}$$

Considerações sobre Cor RGB

- Para cores RGB, as intensidades são modeladas como vetores com 3 elementos que designam os componentes vermelho, verde e azul

$$I_l = (I_{lR}, I_{lG}, I_{lB})$$

- Similarmente, os coeficientes de reflexão são também especificados para cada uma das 3 componentes de cor

$$k_a = (k_{aR}, k_{aG}, k_{aB})$$

$$k_d = (k_{dR}, k_{dG}, k_{dB})$$

$$k_s = (k_{sR}, k_{sG}, k_{sB})$$

- Assim, calcula-se separadamente o modelo de iluminação para cada componente de cor da superfície

Considerações sobre Cor RGB

- Os coeficientes de reflexão são usados para definir a cor de uma superfície
  - Por exemplo, para definir uma superfície azul, escolhe-se um valor diferente de zero para a componente k_{dB}, enquanto as outras componentes são zeradas, i.e. $k_{dR} = k_{dG} = 0.0$
  - Isso corresponde ao efeito de que somente a luz azul é refletida, enquanto as outras componentes de cor são absorvidas pela superfície

Considerações sobre Cor RGB

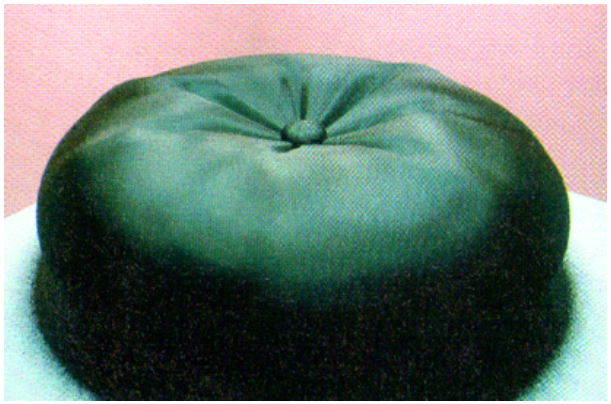
- No modelo original de Phong, k_s tinha valor constante
  - A reflexão especular tem a mesma cor da luz incidente
  - Pouco realismo, superfícies parecem plástico

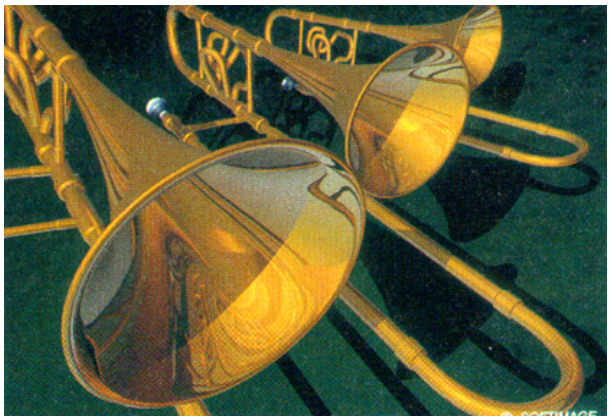
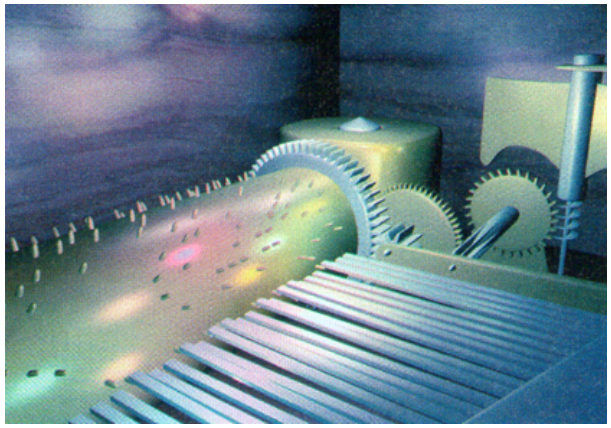
- Para maior realismo a cor da reflexão especular é função do material da superfície
  - Pode ser diferente da cor da luz incidente e da reflexão difusa

$$I_{l,spec} = k_{sB}\ I_{lB}\ \max\{0.0, (\mathbf{N} \cdot \mathbf{H}_l)^{n_s}\}$$

Considerações sobre Cor RGB

- Exemplos de reflexão considerando diferentes materiais e múltiplas fontes (coloridas) de luz



Sumário

1 Introdução

2 Fontes de Luz

3 Efeitos de Luz em Superfícies

4 Modelos Básicos de Iluminação
- Programação OpenGL (Iluminação)

5 Superfícies Transparentes
- Programação OpenGL (Transparência)

Funções de Iluminação e Rendering OpenGL

- As rotinas de iluminação são ativadas usando

```
1 gl.glEnable(GL.GL_LIGHTING);
```

- Múltiplas fontes de luz podem ser adicionadas a uma cena e várias propriedades podem ser associadas a cada fonte usando

```
1 gl.glLight*(light_name, light_property, property_value);
```

- Um sufixo i, iv, f ou fv é adicionado ao nome da função dependendo do tipo de dado do valor da propriedade

Funções de Iluminação e Rendering OpenGL

- O parâmetro *light_name* recebe um identificador
  - *GL.GL_LIGHT0*, *GL.GL_LIGHT1*, *GL.GL_LIGHT2*,...,*GL.GL_LIGHT7*

- Depois de todos os parâmetros de uma luz terem sido definidos, essa deve ser ligada usando

```
1 gl.glEnable(light_name);
```

Especificando a Posição e Tipo de uma Fonte de Luz

- Para designar a posição de uma fonte de luz usa-se o flag *GL.GL_POSITION* e passa-se um vetor de 4 elementos
  - Os primeiros 3 elementos do vetor definem sua posição em coordenadas do mundo

- Esse também é usado para definir o tipo de fonte de luz
  - Fonte próxima da cena (posição)
    - Quarto elemento do vetor diferente de 0.0
  - Fonte distante da cena (direção)
    - Quarto elemento do vetor igual a 0.0

Especificando a Posição e Tipo de uma Fonte de Luz

- O seguinte exemplo define duas fontes de luz, uma local e uma distante

```
1  float light1_pos[] = {2.0f,0.0f,3.0f,1.0f}; //fonte local
2  float light2_pos[] = {0.0f,1.0f,0.0f,0.0f}; //fonte distante
3
4  gl.glLightfv(GL.GL_LIGHT1,GL.GL_POSITION,light1_pos,0); //define posio da luz
5  gl.glEnable(GL.GL_LIGHT1);
6
7  gl.glLightfv(GL.GL_LIGHT2,GL.GL_POSITION,light2_pos,0); //define direo da luz
8  gl.glEnable(GL.GL_LIGHT2);
```

- Os valores padrão de uma fonte de luz são $(0.0, 0.0, 1.0, 0.0)$
  - Fonte de luz distante e luz na direção negativa de z

Especificando a Posição e Tipo de uma Fonte de Luz

- A posição da luz faz parte da descrição da cena, sendo transformada na transformação para o VCS

Especificando as Cores da Fonte de Luz

- A fonte de luz é definida especificando as diferentes cores RGBA
  - A componente *alpha* só é usada quando se ativa a transparência na cena
- A contribuição de cada fonte de luz para os efeitos de luz ambiente, difusa e especular deve ser especificada

```
1 float white[] = {1.0f,1.0f,1.0f,1.0f};
2 float black[] = {0.0f,0.0f,0.0f,1.0f};
3
4 gl.glLightfv(GL.GL_LIGHT1,GL.GL_AMBIENT,black,0); //contribuio ambiente
5 gl.glLightfv(GL.GL_LIGHT1,GL.GL_DIFFUSE,white,0); //contribuio difusa
6 gl.glLightfv(GL.GL_LIGHT1,GL.GL_SPECULAR,white,0); //contribuio especular
```

- Por padrão, a fonte de luz 0 tem a propriedade de luz ambiente preta e as propriedades difusa e especular branca
  - Para as demais fontes, todas as propriedades são pretas

Especificando as Cores da Fonte de Luz

```
1  public class IluminacaoFonteLuz implements GLEventListener {
2
3      public void init(GLAutoDrawable drawable) {
4          GL gl = drawable.getGL();
5          GLU glu = new GLU();
6
7          gl.glClearColor(1.0f, 1.0f, 1.0f, 1.0f); //define a cor de fundo
8          gl.glEnable(GL.GL_DEPTH_TEST); //habilita o teste de profundidade
9
10         gl.glMatrixMode(GL.GL_MODELVIEW); //define que a matrix  a model view
11         gl.glLoadIdentity(); //carrega a matrix de identidade
12         glu.gluLookAt(0.0, 0.0, 1.0, //posio da cmera
13                 0.0, 0.0, 0.0, //para onde a cmera aponta
14                 0.0, 1.0, 0.0); //vetor view-up
15
16         gl.glMatrixMode(GL.GL_PROJECTION); //define que a matrix  a de projeo
17         gl.glLoadIdentity(); //carrega a matrix de identidade
18         gl.glOrtho(-2.0, 2.0, -2.0, 2.0, -2.0, 2.0);
19
20         lighting(drawable); //defininido os parmetros de iluminao
21     }
22
23     ...
24 }
```

Especificando as Cores da Fonte de Luz

```
1  public class IluminacaoFonteLuz implements GLEventListener {
2
3      public static void main(String[] args) {
4          //cria o painel e adiciona um ouvinte GL
5          Renderer pp = new Renderer();
6          GLCanvas canvas = new GLCanvas();
7          canvas.addGLEventListener(pp);
8
9          //cria uma janela e adiciona o canvas
10         JFrame frame = new JFrame("Aplicacao JOGL Simples");
11         frame.add(canvas);
12         frame.setSize(400, 400);
13         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14
15         //inicializa o sistema e chama display() a 60 fps
16         Animator animator = new FPSAnimator(canvas, 60);
17         frame.setLocationRelativeTo(null);
18         frame.setVisible(true);
19         animator.start();
20     }
21
22     public void display(GLAutoDrawable drawable) {
23         GL gl = drawable.getGL();
24         GLUT glut = new GLUT();
25
26         //limpa o buffer
27         gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
28
29         //define que a matrix  a model view
30         gl.glMatrixMode(GL.GL_MODELVIEW);
31         glut.glutSolidSphere(1.5, 40, 40);
32
33         //fora o desenho das primitivas
34         gl.glFlush();
35     }
36
37     ...
38 }
```

Especificando as Cores da Fonte de Luz

```
1  private void lighting(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3
4    //define a posio e parmetros da luz 0
5    float position[] = {2.0f, 2.0f, 2.0f, 1.0f};
6    float white[] = {1.0f, 1.0f, 1.0f, 1.0f};
7    float black[] = {0.0f, 0.0f, 0.0f, 1.0f};
8
9    gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, position, 0);
10   gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, black, 0);
11   gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, white, 0);
12   gl.glLightfv(GL.GL_LIGHT0, GL.GL_SPECULAR, white, 0);
13
14   //ativa a iluminao
15   gl.glEnable(GL.GL_LIGHTING);
16   gl.glEnable(GL.GL_LIGHT0);
17 }
```

Especificando as Cores da Fonte de Luz

Especificando os Coeficientes de atenuação Radial

- É possível especificar os coeficientes de atenuação radial a_0, a_1 e a_2 usando

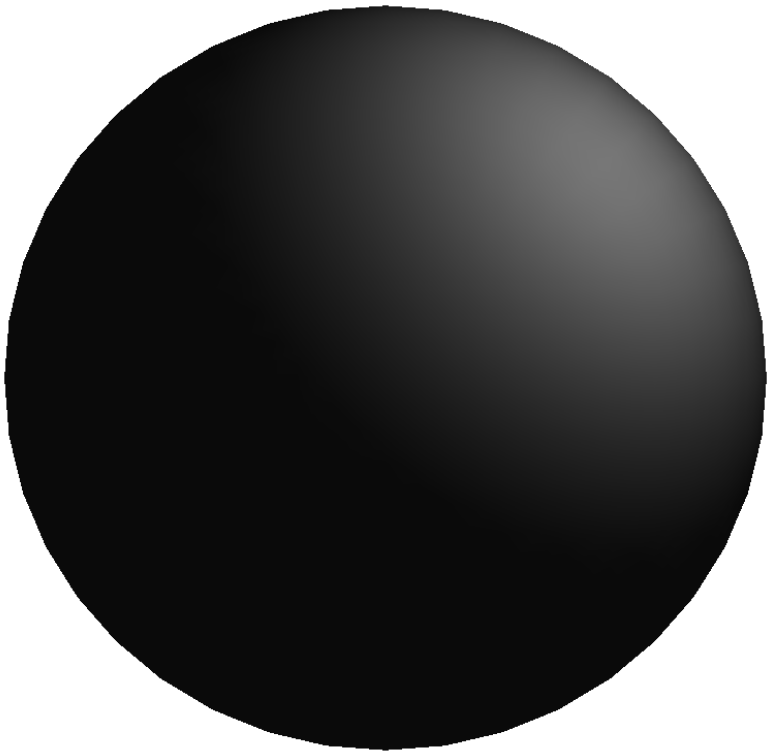
```
1 gl.glLightf(GL.GL_LIGHT0, GL.GL_CONSTANT_ATTENUATION, 0.5f); //define a0
2 gl.glLightf(GL.GL_LIGHT0, GL.GL_LINEAR_ATTENUATION, 0.15f); //define a1
3 gl.glLightf(GL.GL_LIGHT0, GL.GL_QUADRATIC_ATTENUATION, 0.1f); //define a2
```

- Os coeficientes podem assumir valores inteiros ou de ponto flutuante positivos
- Os valores padrão para a atenuação radial são $a_0 = 1$, $a_1 = 0$ e $a_2 = 0$ (atenuação desativada)

Especificando as Cores da Fonte de Luz

```
1  private void lighting(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3
4    //define a posio e parmetros da luz 0
5    float position[] = {2.0f, 2.0f, 2.0f, 1.0f};
6    float white[] = {1.0f, 1.0f, 1.0f, 1.0f};
7    float black[] = {0.0f, 0.0f, 0.0f, 1.0f};
8
9    gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, position, 0);
10   gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, black, 0);
11   gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, white, 0);
12   gl.glLightfv(GL.GL_LIGHT0, GL.GL_SPECULAR, white, 0);
13
14   //ativa a atenuao
15   gl.glLightf(GL.GL_LIGHT0, GL.GL_CONSTANT_ATTENUATION, 0.5f); //define a0
16   gl.glLightf(GL.GL_LIGHT0, GL.GL_LINEAR_ATTENUATION, 0.5f); //define a1
17   gl.glLightf(GL.GL_LIGHT0, GL.GL_QUADRATIC_ATTENUATION, 0.1f); //define a2
18
19   //ativa a iluminao
20   gl.glEnable(GL.GL_LIGHTING);
21   gl.glEnable(GL.GL_LIGHT0);
22 }
```

Especificando as Cores da Fonte de Luz



Fontes de Luz Direcionais (Holofotes)

- É possível especificar o efeito de holofote definindo o cone de atuação da luz direcional
  - Para isso define-se o vetor de direção da luz
  - O valor θ_l do espalhamento angular a partir do eixo do cone
  - O valor do exponente de atenuação angular a_l

```
1 float direction[] = {1.0f, 0.0f, 0.0f};
2 gl.glLightfv(GL.GL_LIGHT1, GL.GL_SPOT_DIRECTION, direction, 0); //vetor direo
3 gl.glLightf(GL.GL_LIGHT1, GL.GL_SPOT_CUTOFF, 45.0f); //espalhamento angular
4 gl.glLightf(GL.GL_LIGHT1, GL.GL_SPOT_EXPONENT, 0.5f); //atenuao angular
```

- O valor de θ_l deve estar entre 0^0 e 90^0, fazendo igual 180^0 a luz emite raios em todas as direções
- O valor de a_l é um inteiro ou ponto flutuante no intervalo de 0 a 128

Fontes de Luz Direcionais (Holofotes)

```
1  private void lighting(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3
4    //define a posio e parmetros da luz 0
5    float position0[] = {2.0f, 2.0f, 2.0f, 1.0f};
6    float white[] = {1.0f, 1.0f, 1.0f, 1.0f};
7    float black[] = {0.0f, 0.0f, 0.0f, 1.0f};
8
9    gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, position0, 0);
10   gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, black, 0);
11   gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, white, 0);
12   gl.glLightfv(GL.GL_LIGHT0, GL.GL_SPECULAR, white, 0);
13
14   //define a posio e parmetros da luz 1
15   float position1[] = {-2.0f, 0.0f, 0.0f, 1.0f};
16   gl.glLightfv(GL.GL_LIGHT1, GL.GL_POSITION, position1, 0);
17   gl.glLightfv(GL.GL_LIGHT1, GL.GL_DIFFUSE, white, 0);
18   gl.glLightfv(GL.GL_LIGHT1, GL.GL_SPECULAR, white, 0);
19
20   float direction[] = {1.0f, 0.0f, 0.0f};
21   gl.glLightfv(GL.GL_LIGHT1, GL.GL_SPOT_DIRECTION, direction, 0); //vetor direo
22   gl.glLightf(GL.GL_LIGHT1, GL.GL_SPOT_CUTOFF, 45.0f); //espalhamento angular
23   gl.glLightf(GL.GL_LIGHT1, GL.GL_SPOT_EXPONENT, 0.5f); //atenuao angular
24
25   //ativa a iluminao
26   gl.glEnable(GL.GL_LIGHTING);
27   gl.glEnable(GL.GL_LIGHT0);
28   gl.glEnable(GL.GL_LIGHT1);
29 }
```

Fontes de Luz Direcionais (Holofotes)

Parâmetros de Iluminação Global

- Vários parâmetros de iluminação podem ser definidos globalmente usando

```
1 gl.glLightModel*(param_name, param_value);
```

- O sufixo pode ser i, iv, f ou fv dependendo do tipo de parâmetro

- Podemos definir globalmente
  - O nível de luz ambiente
  - Como os brilhos especulares são calculados
  - Se o modelo de iluminação deve ser aplicado na parte de trás dos polígonos

Parâmetros de Iluminação Global

- Por exemplo, para definir uma luz ambiente independente das fontes de luz existentes usamos

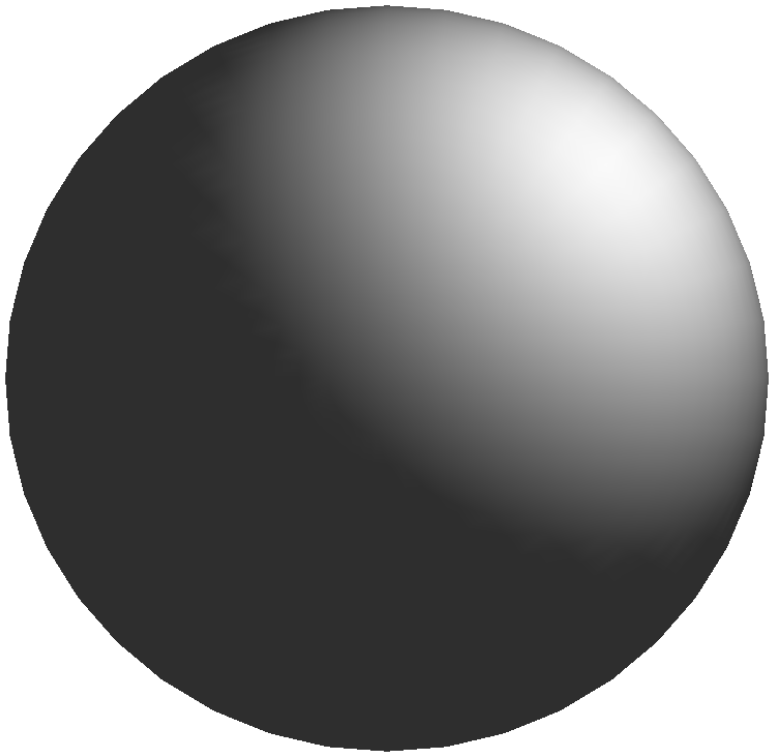
```
1 float global_ambient[] = {0.9f, 0.9f, 0.9f, 1.0f};
2 gl.glLightModelfv(GL.GL_LIGHT_MODEL_AMBIENT, global_ambient, 0);
```

- Por padrão essa cor é branca de baixa intensidade $(0.2, 0.2, 0.2, 1.0)$

Parâmetros de Iluminação Global

```
1  private void lighting(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3
4    //define a posio e parmetros da luz 0
5    float position[] = {2.0f, 2.0f, 2.0f, 1.0f};
6    float white[] = {1.0f, 1.0f, 1.0f, 1.0f};
7    float black[] = {0.0f, 0.0f, 0.0f, 1.0f};
8
9    gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, position, 0);
10   gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, black, 0);
11   gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, white, 0);
12   gl.glLightfv(GL.GL_LIGHT0, GL.GL_SPECULAR, white, 0);
13
14   //ativando luz ambiente global
15   float global_ambient[] = {0.9f, 0.9f, 0.9f, 1.0f};
16   gl.glLightModelfv(GL.GL_LIGHT_MODEL_AMBIENT, global_ambient, 0);
17
18   //ativa a iluminao
19   gl.glEnable(GL.GL_LIGHTING);
20   gl.glEnable(GL.GL_LIGHT0);
21 }
```

Parâmetros de Iluminação Global



Parâmetros de Iluminação Global

- Para o cálculo da reflexão especular é necessário determinar o vetor $\mathbf{V}$ (da superfície para a posição de observação)
  - Podemos acelerar o processamento fazendo $\mathbf{V}$ constante independente da posição da superfície

- O valor padrão para $\mathbf{V}$ é a direção de z $(0.0, 0.0, 1.0)$, mas podemos usar a posição de observação corrente fazendo

```
1 gl.glLightModeli(GL.GL_LIGHT_MODEL_LOCAL_VIEWER, GL.GL_TRUE);
```

- Isso tornará o processo computacionalmente mais caro, mas o resultado será mais realístico
  - Para desabilitar o cálculo de $\mathbf{V}$ fazer essa função igual a *GL.GL_FALSE*

Parâmetros de Iluminação Global

- Por padrão somente as faces frontais dos polígonos são iluminadas, para habilitar os cálculos de iluminação para ambas as faces usamos

```
1 gl.glLightModeli(GL.GL_LIGHT_MODEL_TWO_SIDE, GL.GL_TRUE);
```

- As normais das faces traseiras serão o reverso das faces frontais
- Para desligar essa opção, usar *GL.GL_FALSE*, que é o padrão

Propriedades da Superfície

- As propriedades óticas das superfícies são definidas usando

```
1 gl.glMaterial*(surface_face, surface_property, property_value);
```

- O sufixo pode ser i, iv, f ou fv dependendo do tipo de parâmetro

- O parâmetro *surface_face* indica a qual face o material se designa e pode ser
  - *GL.GL_FRONT*, *GL.GL_BACK* e *GL.GL_FRONT_AND_BACK*

- O parâmetro *surface_property* identifica o parâmetro da superfície e pode ser
  - I_{surf}, k_a, k_d, k_s ou n_s

Propriedades da Superfície

- Por exemplo, os valores RGBA para uma superfície emissora de luz (I_{surf}) podem ser especificados usando

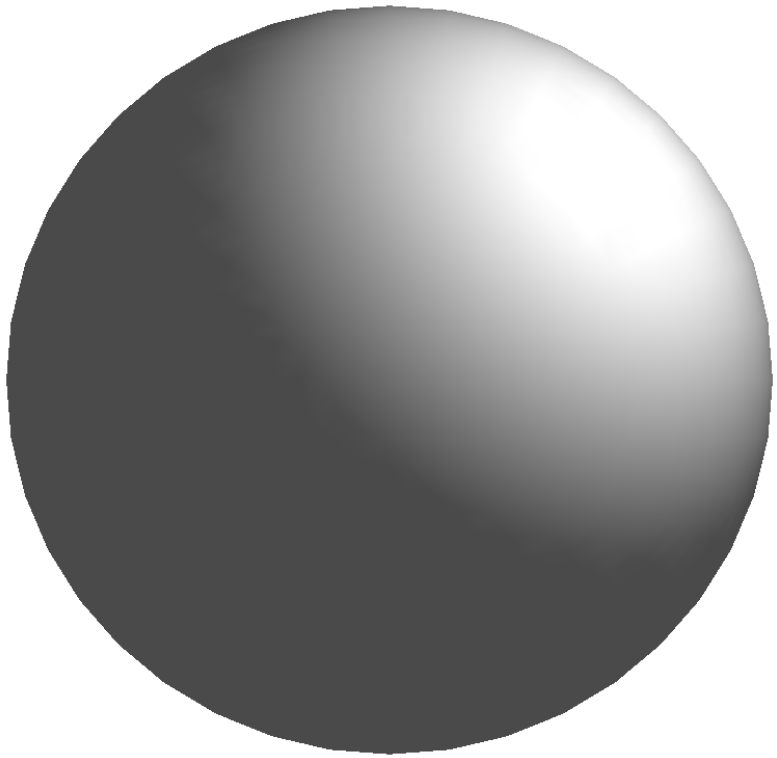
```
1 float surface_emission[] = {0.25f, 0.25f, 0.25f, 1.0f};
2 gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, surface_emission, 0);
```

- Por padrão o valor de emissão é $(0.0, 0.0, 0.0, 1.0)$
- A emissão de luz de uma superfície não é usada para iluminar outras superfícies

Propriedades da Superfície

```
1  public void display(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3    GLUT glut = new GLUT();
4
5    //limpa o buffer
6    gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
7
8    //superficie vai emitir luz
9    float surface_emission[] = {0.25f, 0.25f, 0.25f, 1.0f};
10   gl.glMaterialfv(GL.GL_FRONT, GL.GL_EMISSION, surface_emission, 0);
11
12   //define que a matrix  a a model view
13   gl.glMatrixMode(GL.GL_MODELVIEW);
14   glut.glutSolidSphere(1.5, 40, 40);
15
16   //fora o desenho das primitivas
17   gl.glFlush();
18 }
```

Propriedades da Superfície



Propriedades da Superfície

- Os flags *GL.GL_AMBIENT*, *GL.GL_DIFFUSE* e *GL.GL_SPECULAR* são usados para definir os coeficientes de reflexão da superfície
  - Normalmente, os valores dos coeficientes difuso e ambiente são os mesmos, para isso usamos GL.GL_AMBIENT_AND_DIFFUSE

- Os valores padrão para os coeficientes são
  - Luz ambiente (k_a) $(0.2, 0.2, 0.2, 1.0)$
  - Luz difusa (k_d) $(0.8, 0.8, 0.8, 1.0)$
  - Luz especular (k_s) $(1.0, 1.0, 1.0, 1.0)$

- Para definir o expoente de reflexão especular (n_s) usamos *GL.GL_SHININESS* entre 0 e 128
  - O valor padrão é 0

Propriedades da Superfície

- Por exemplo, para definir os coeficientes de reflexão ambiente, difusa e especular podemos fazer

```
1 float diffuse[] = {0.65f, 0.65f, 0.0f, 1.0f};
2 float specular[] = {0.9f, 0.9f, 0.9f, 1.0f};
3 float shininess = 65.0f;
4
5 gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT_AND_DIFFUSE, diffuse, 0);
6 gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, specular, 0);
7 gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, shininess);
```

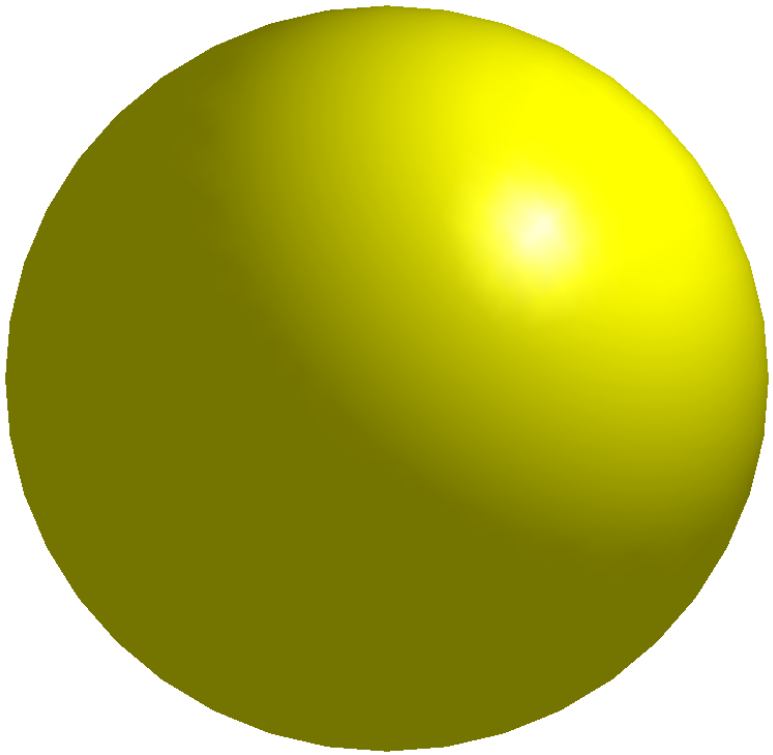
Propriedades da Superfície

```
1  public void display(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3    GLUT glut = new GLUT();
4
5    //limpa o buffer
6    gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
7
8    //define material da superfcie
9    float kd[] = {0.65f, 0.65f, 0.0f, 1.0f};
10   float ks[] = {0.9f, 0.9f, 0.9f, 1.0f};
11   float ns = 65.0f;
12
13   gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT_AND_DIFFUSE, kd, 0);
14   gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, ks, 0);
15   gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, ns);
16
17   //define que a matrix  a a model view
18   gl.glMatrixMode(GL.GL_MODELVIEW);
19   glut.glutSolidSphere(1.5, 40, 40);
20
21   //fora o desenho das primitivas
22   gl.glFlush();
23 }
```

Propriedades da Superfície

```
1  private void lighting(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3
4    //define a posio e parmetros da luz 0
5    float position[] = {2.0f, 2.0f, 2.0f, 1.0f};
6    float white[] = {1.0f, 1.0f, 1.0f, 1.0f};
7    float black[] = {0.0f, 0.0f, 0.0f, 1.0f};
8
9    gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, position, 0);
10   gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, black, 0);
11   gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, white, 0);
12   gl.glLightfv(GL.GL_LIGHT0, GL.GL_SPECULAR, white, 0);
13
14   //ativando luz ambiente global
15   float global_ambient[] = {0.7f, 0.7f, 0.7f, 1.0f};
16   gl.glLightModelfv(GL.GL_LIGHT_MODEL_AMBIENT, global_ambient, 0);
17
18   //ativa a iluminao
19   gl.glEnable(GL.GL_LIGHTING);
20   gl.glEnable(GL.GL_LIGHT0);
21 }
```

Propriedades da Superfície



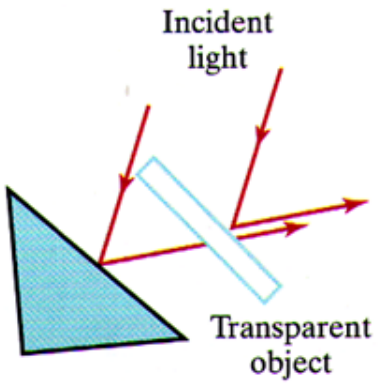
Sumário

1. Introdução
2. Fontes de Luz
3. Efeitos de Luz em Superfícies
4. Modelos Básicos de Iluminação
   - Programação OpenGL (Iluminação)
5. Superfícies Transparentes
   - Programação OpenGL (Transparência)

Luminância

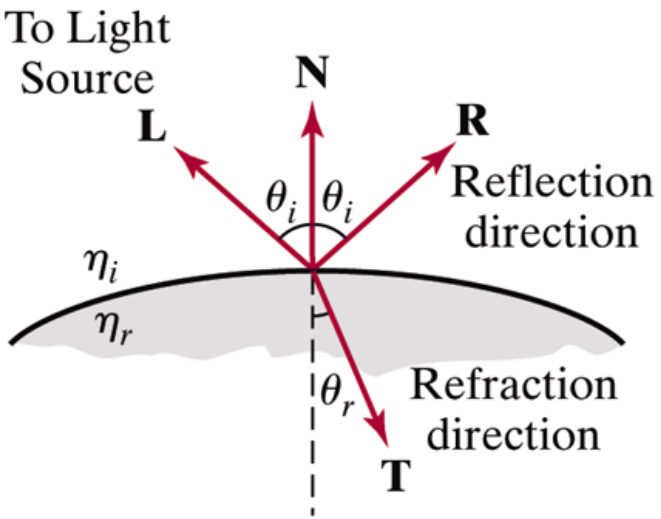
Superfícies Transparentes

- Materiais **transparentes** (como vidro) permitem ver outros objetos através deles, ao contrário dos materiais **opacos**
- Materiais intermediários, em que a luz é transmitida e difundida em todas as direções, são conhecidos como **translúcidos** (p.ex. plástico)



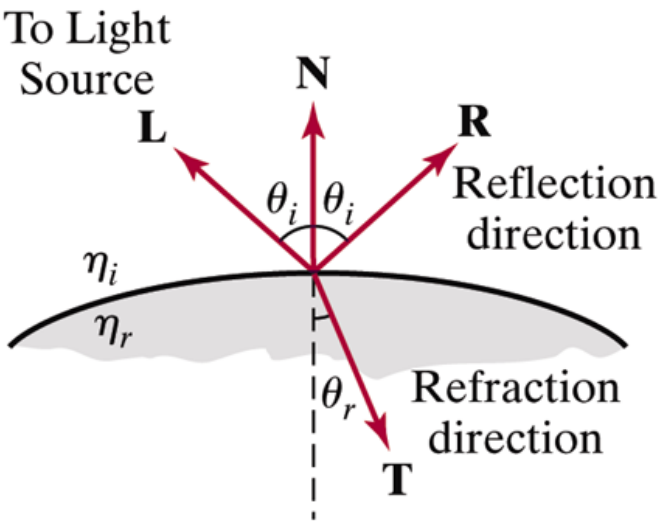
Refração da Luz

- **Imagens realísticas** de materiais transparentes podem ser obtidas modelando o **caminho de refração** de um raio de luz através do material
  - Parte da luz é **refletida** e parte é **refratada**
  - Como a propagação da luz depende do material, a direção da luz refratada é diferente da direção da luz incidente



Refração da Luz

- A direção da refração, determinada pelo **ângulo de refração** (θ_r), é função do **índice de refração** do material e da direção da luz incidente
  - O índice de refração é a razão entre a velocidade de propagação da luz no vácuo e a velocidade de propagação da luz no material

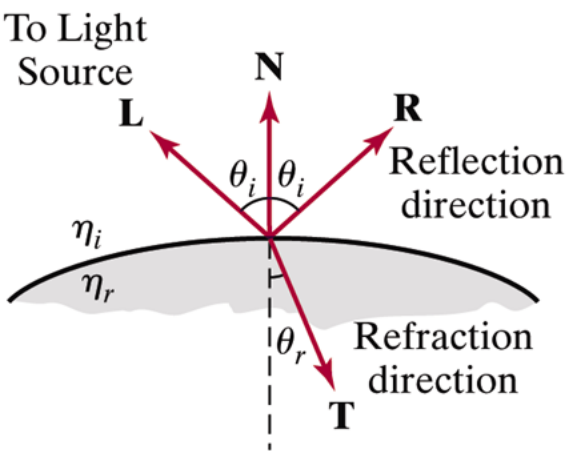


Refração da Luz

- A **lei de Snell** permite calcular o ângulo de refração θ_r

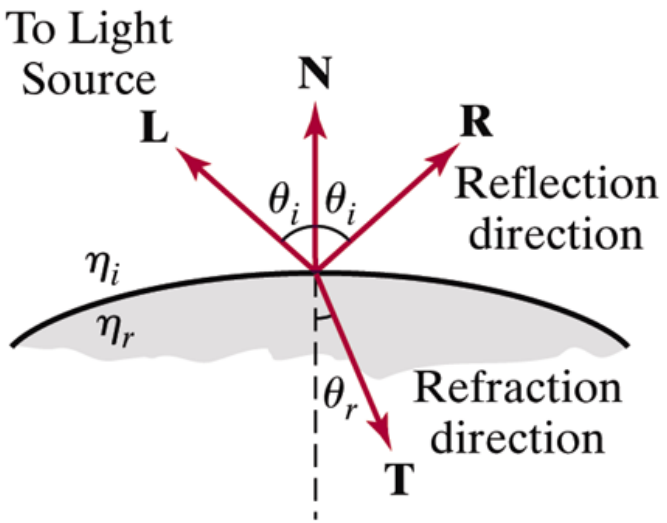
$$\operatorname{sen}\theta_r = \frac{\eta_i}{\eta_r}\operatorname{sen}\theta_i$$

- Em que θ_i é o ângulo de incidência da luz, η_i é o índice de refração do material no meio incidente e η_r é o índice de refração do material translúcido



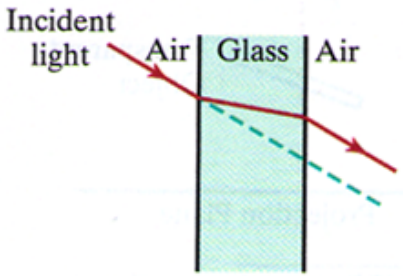
Refração da Luz

- Por exemplo, luz incidindo em $\theta_i = 30^0$ sobre um vidro espesso ($\eta_r \approx 1.61$) a partir do ar ($\eta_i \approx 1.0$), será refratada em um ângulo de $\theta_r \approx 18^0$



Refração da Luz

- O efeito da refração é alterar a direção da luz incidente para uma direção um pouco diferente, mas paralela à original



- A equação de refração é computacionalmente cara de ser avaliada, então normalmente utiliza-se aproximações

Refração da Luz

- O vetor de transmissão $\mathbf{T}$ na direção de refração θ_r é calculado como

$$\mathbf{T} = \left(\frac{\eta_i}{\eta_r} \cos\theta_i - \cos\theta_r \right) \mathbf{N} - \frac{\eta_i}{\eta_r} \mathbf{L}$$

- Em que $\mathbf{N}$ é a normal a superfície, $\mathbf{L}$ é a direção da fonte de luz

- $\mathbf{T}$ pode ser usado para localizar intersecções do caminho de refração com objetos atrás da superfície transparente

Modelo Básico de Transparência

- Um modelo simples de refração pode ignorar a mudança na direção da luz
  - Não existe mudança de índice de refração entre materiais diferentes
  - O ângulo de incidência será igual ao ângulo de refração

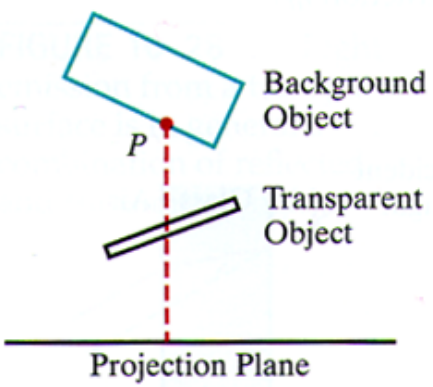
- Acelera o processamento e produz bons resultados para superfícies finas

Modelo Básico de Transparência

- Podemos combinar a intensidade transmitida I_{trans} através da superfície transparente de um objeto de fundo com a intensidade refletida I_{refl} usando um **coeficiente de transmissão** k_t

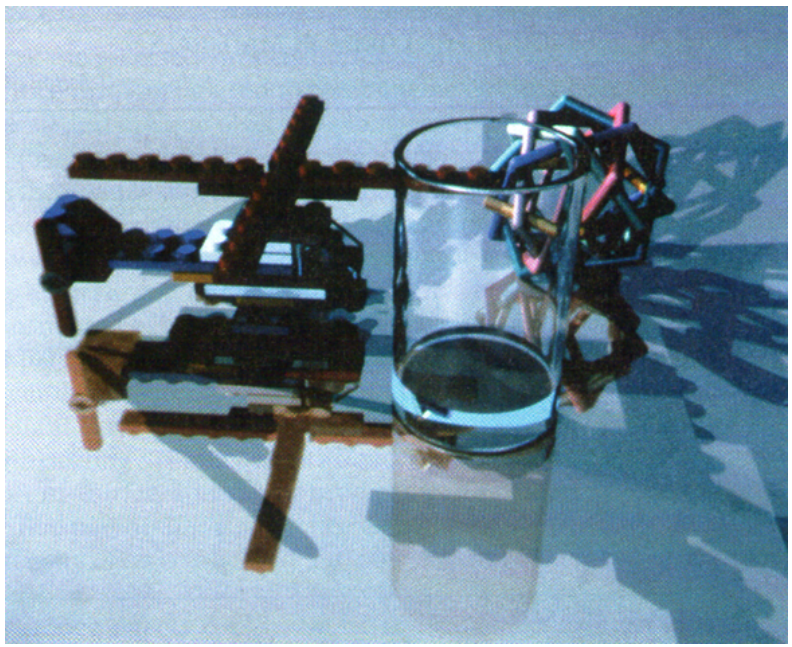
$$I = (1 - k_t)\ I_{refl} + k_t\ I_{trans}$$

- Em que o termo $(1 - k_t)$ é o fator de opacidade



Modelo Básico de Transparência

- Esse procedimento pode ser usado com um número arbitrário de objetos transparentes e opacos
  - Processa primeiro os objetos de trás e depois os da frente



Sumário

1. Introdução
2. Fontes de Luz
3. Efeitos de Luz em Superfícies
4. Modelos Básicos de Iluminação
   - Programação OpenGL (Iluminação)
5. Superfícies Transparentes
   - Programação OpenGL (Transparência)

Funções de Transparências

- O uso de transparência não é direto na OpenGL, é necessário combinar a cor dos objetos e usar um parâmetro de *alpha blending*
  - Ignora efeitos de refração
  - Em cenas complexas com várias fontes de iluminação pode ser difícil

- Não existe suporte para materiais translúcidos (refração e reflexão)
  - Para simular esse efeito funções devem ser providas

Funções de Transparências

- Materiais transparentes são definidos usando o parâmetro *alpha* RGBA em comandos como *gl.glMaterial(...)* e *gl.glColor(...)*

- Por exemplo, usando a função

```
1 gl.glColor4f(R, G, B, A)
```

- O valor $A = k_t$, onde uma superfície transparente terá $A = 1.0$ e opaca $A = 0.0$

Funções de Transparências

- Uma vez definido os valores de transparência essa deve ser ativada

```
1 gl.glEnable(GL.GL_BLEND);
2 gl.glBlendFunc(GL.GL_ONE_MINUS_SRC_ALPHA, GL.GL_SRC_ALPHA);
```

- Todas as componentes de cor da superfície corrente (o objeto "origem") são multiplicadas por $(1 - A) = (1 - k_t)$ e todas as componentes de cor da posição correspondente no *frame buffer* (o "destino") são multiplicadas por $A = k_t$

Funções de Transparências

```
1  private void lighting(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3
4    //define a posio e parmetros da luz 0
5    float position[] = {2.0f, 2.0f, 2.0f, 1.0f};
6    float white[] = {1.0f, 1.0f, 1.0f, 1.0f};
7    float black[] = {0.0f, 0.0f, 0.0f, 1.0f};
8
9    gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, position, 0);
10   gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, black, 0);
11   gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, white, 0);
12   gl.glLightfv(GL.GL_LIGHT0, GL.GL_SPECULAR, white, 0);
13
14   //ativa a iluminao
15   gl.glEnable(GL.GL_LIGHTING);
16   gl.glEnable(GL.GL_LIGHT0);
17
18   //ativa transparncia
19   gl.glEnable(GL.GL_BLEND);
20   gl.glBlendFunc(GL.GL_ONE_MINUS_SRC_ALPHA, GL.GL_SRC_ALPHA);
21 }
```

Funções de Transparências

```
1  public void display(GLAutoDrawable drawable) {
2    GL gl = drawable.getGL();
3    GLUT glut = new GLUT();
4
5    //limpa o buffer
6    gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
7
8    //define que a matrix  a de modelo
9    gl.glMatrixMode(GL.GL_MODELVIEW);
10
11   //desenho a esfera de dentro
12   float diffuse1[] = {0.0f, 0.65f, 0.0f, 0.25f};
13   float specular1[] = {0.9f, 0.9f, 0.9f, 0.25f};
14   float shininess1 = 65.0f;
15   gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT_AND_DIFFUSE, diffuse1, 0);
16   gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, specular1, 0);
17   gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, shininess1);
18   glut.glutSolidSphere(0.75, 40, 40);
19
20   //desenho a esfera de fora
21   float diffuse2[] = {0.0f, 0.0f, 0.65f, 0.5f};
22   float specular2[] = {0.9f, 0.9f, 0.9f, 0.5f};
23   float shininess2 = 65.0f;
24   gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT_AND_DIFFUSE, diffuse2, 0);
25   gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, specular2, 0);
26   gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, shininess2);
27   glut.glutSolidSphere(1.5, 40, 40);
28
29   //fora o desenho das primitivas
30   gl.glFlush();
31 }
```

Funções de Transparências