

# Modelos de Iluminação e Métodos para Rendering de Superfície

SCC0250 - Computação Gráfica

Prof. Fernando V. Paulovich  
<http://www.icmc.usp.br/~paulovic>  
[paulovic@icmc.usp.br](mailto:paulovic@icmc.usp.br)

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

31 de maio de 2010



# 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)
- 6 Métodos de Rendering de Superfície
  - Programação OpenGL (Rendering)

# 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)
- 6 Métodos de Rendering de Superfície
  - Programação OpenGL (Rendering)

# Introdução

- Imagens **realísticas** são criadas usando projeções **perspectivas**, aplicando-se efeitos de **iluminação natural** às superfícies visíveis
- Um **Modelo de Iluminação** (*shading model*) é usado para calcular a cor de um ponto na superfície do objeto
- Um método de **Rendering de Superfície** então usa esses cálculos de cor para determinar as cores dos pixels

# Introdução

- O **rendering da superfície** pode obtido por meio de um modelo de iluminação aplicado a **cada ponto** dos objetos, ou por meio de **interpolações** a partir de um pequeno conjunto de cálculos
  - Algoritmos de *scan-line* tipicamente usam esquemas de interpolação
  - Algoritmos de *ray-tracing* podem empregar o modelo de iluminação para cada pixel

# Introdução

- De forma geral, modelar os **efeitos da luz** sobre um objeto é processo **complexo** que envolve princípios físicos e psicológicos
- Os modelos **físicos** envolvem vários fatores, como propriedades dos **materiais**, **posição** do objeto em relação a luz e outros objetos, e características das **fontes de luz**
  - Objetos podem ser opacos ou mais ou menos transparentes, podem ser finos ou grosseiros
  - Fontes de luz podem ter vários formatos, cores e posições
- Os modelos de iluminação em **computação gráfica** são na maioria das vezes **aproximações** das leis físicas que descrevem efeitos de luz sobre superfícies

# 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)

6 Métodos de Rendering de Superfície

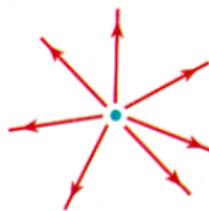
- Programação OpenGL (Rendering)

# Fontes de Luz

- Qualquer objeto que emite energia brilhante é uma **fonte de luz** que contribui para os efeitos de luz dos outros objetos na cena
- **Fontes de luz** podem ter diferentes **formas** e **características** (posição, cor, direção de emissão, etc.), podendo emitir luz ou refletir luz
  - Um lustre redondo de vidro ao redor de uma lâmpada vai emitir e refletir luz
- Em aplicações gráficas de **tempo real**, um modelo simples de iluminação normalmente é aplicado por causa do **custo computacional**
  - Propriedades da emissão de luz são definidas usando valores distintos para cada componente de cor RGB, descrevendo suas “intensidades”

# Fontes de Luz Puntuais

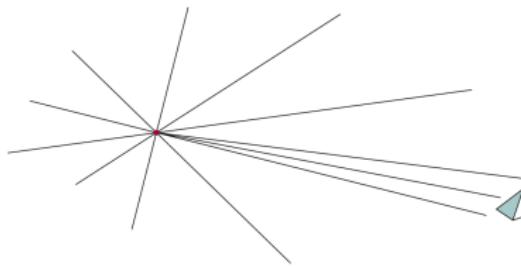
- O modelo mais simples de emissão de luz é **fonte de luz puntual** com uma única cor
  - Definida por uma **posição** e a **cor** da luz emitida



- Os raios de luz são gerados em **direções radiais divergentes** a partir do ponto de luz
  - Indicado para aproximar efeitos de luz quando essa é pequena em comparação com os objetos da cena

# Fontes de Luz Infinitamente Distantes

- Uma fonte de luz grande (p.ex. o sol) que está bem longe da cena pode ser aproximada como um ponto emissor bem distante dos objetos
  - A iluminação é provida em uma única direção



- Uma fonte de luz distante é simulada definindo sua **cor** e uma **direção** da emissão dos raios, não é necessário especificar uma posição

# Atenuação Radial da Intensidade

- A energia de radiação de uma fonte de luz a uma distância  $d_l$  da origem tem sua **amplitude atenuada** por um fator  $1/d_l^2$ 
  - Uma superfície próxima da fonte recebe uma maior intensidade de luz
  - Para uma **iluminação realística** essa atenuação deve ser levada em consideração
- Na prática uma atenuação de  $1/d_l^2$  para fontes de luz puntuais não produz efeitos realísticos
  - Tende a definir uma alta variação da intensidade para objetos que estão próximos da fonte de luz e pouca variação quando  $d_l$  é grande

# Atenuação Radial da Intensidade

- Para produzir efeitos mais realísticos com fontes puntuais usamos

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

- Os valores de  $a_0$ ,  $a_1$  e  $a_2$  podem ser ajustados para se produzir efeitos de atenuação desejados
  - Valores grandes podem ser assinalados a  $a_0$  quando  $d_l$  é muito pequeno para prevenir  $f_{radatten}(d_l)$  de ficar muito grande

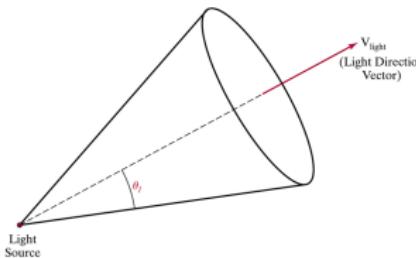
# Atenuação Radial da Intensidade

- Esse cálculo de atenuação não pode ser empregado quando a fonte puntual estiver no “infinito” porque a distância à fonte é indeterminada
- Outro problema é que quase todos os pontos da cena ficariam quase a mesma distância da fonte de luz (baixo realismo)
- Para resolver esses problemas podemos usar

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

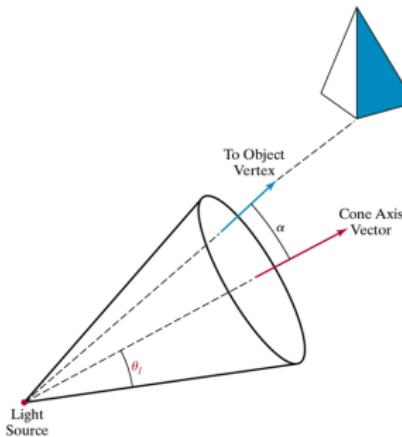
# Fontes de Luz Direcional e Efeitos de Holofote

- Uma fonte de luz local pode ser modificada para produzir uma **fonte direcional** ou holofote
  - Se um objeto está fora dos limites direcionais ele é eliminado da iluminação
- Uma **fonte de luz direcional** pode ser definida por meio de uma **posição**, da **cor** da luz, um vetor de **direção** e um **limite angular**  $\theta_l$  a partir desse vetor



# Fontes de Luz Direcional e Efeitos de Holofote

- Podemos usar um vetor unitário  $\mathbf{V}_{light}$  na direção da fonte de luz e um vetor unitário  $\mathbf{V}_{obj}$  na direção da posição da luz para um objeto
- Considerando que  $\cos \alpha = \mathbf{V}_{light} \cdot \mathbf{V}_{obj}$  e limitando  $0^0 \leq \theta_l \leq 90^0$ , então o objeto está dentro da região de luz se  $\cos \alpha \geq \cos \theta_l$



# Atenuação Angular de Intensidade

- Para uma fonte de luz direcional a atenuação pode ocorre **angularmente** e **radialmente** a partir da posição da fonte
  - Permite simular cones de luz que são mais intensos ao longo do eixo do cone

- Uma função usual de atenuação é

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

- Onde  $a_l > 0$  é um expoente de atenuação e  $\phi$  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 de  $a_l$ , menor o valor de  $f_{angatten}(\phi)$  dado  $\phi > 0^0$

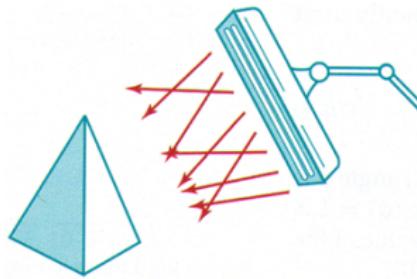
# Atenuação Angular de Intensidade

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

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

# Fontes de Luz Estendidas e o Modelo de Warn

- Para se incluir uma fonte de luz grande em uma posição próxima aos objetos da cena, podemos aproximar esse efeito como uma superfície que emite luz



- Podemos fazer isso modelando a superfície de luz como um *grid* de 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)
- 6 Métodos de Rendering de Superfície
  - Programação OpenGL (Rendering)

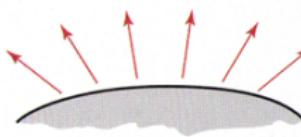
# Efeitos de Luz em Superfícies

- Um modelo de iluminação computa os **efeitos de luz** para uma superfície usando várias **propriedades óticas**
- Quando a superfície é opaca, parte da luz é refletida e parte absorvida
  - A quantidade de luz refletida depende do tipo de material da superfície
- Para superfícies transparentes, alguma luz é também transmitida através da superfície

# Efeitos de Luz em Superfícies

## Reflexão Difusa

- Superfícies irregulares tendem a refletir a luz em todas as direções
- Superfícies muito irregulares podem parecer igualmente brilhantes a partir de qualquer ponto de vista

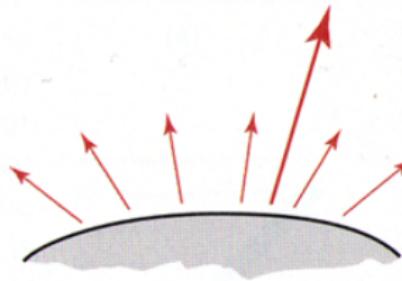


- A cor de um objeto é a cor da reflexão difusa com uma iluminação de luz branca
  - Objetos azuis refletem a componente azul da luz branca
  - Um objeto azul sobre luz vermelha ficará preto pois toda luz é absorvida

# Efeitos de Luz em Superfícies

## Reflexão Especular

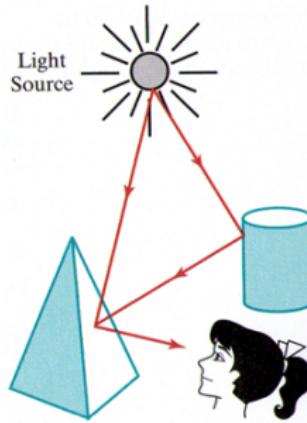
- Alguma parte da luz refletida é concentrada em uma região mais brilhante
- O efeito de realce é mais pronunciado em superfícies de materiais brilhantes



# Efeitos de Luz em Superfícies

## Luz de Fundo ou Ambiente

- Efeito de iluminação produzido pela luz refletida de várias superfícies na cena
  - A luz total refletida de uma superfície é a soma das contribuições da luz refletida pelas outras superfícies



# 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)
- 6 Métodos de Rendering de Superfície
  - Programação OpenGL (Rendering)

# Modelos Básicos de Iluminação

- **Modelos precisos** de iluminação computam toda a **interação** entre a radiação de **luz** e o **material** dos objetos
  - Esses efeitos podem ser aproximados usando modelos empíricos com bons resultados

## Luz Ambiente

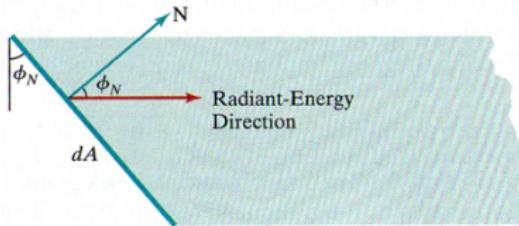
- **Luz de fundo** pode ser incorporada definindo um nível de **brilho geral** para a cena
    - Produz luz ambiente uniforme para todos os objetos, aproximando as reflexões difusas de todas superfícies da cena
  - A quantidade de luz refletida dependerá das propriedades óticas (material) das superfícies
- 
- O nível de luz ambiente em uma cena é definido por um parâmetro de intensidade  $I_a$

## Reflexão Difusa

- A reflexão difusa pode ser modelada assumindo que a luz incidente é espalhada com igual intensidade em todas as direções independente da direção de visão
  - Essas superfícies são chamadas de **refletores difusos ideais** (refletores Lambertianos)

# Reflexão Difusa

- A radiação de luz refletida é calculada com a **Lei do Cosseno de Lambert**
  - A quantidade de energia radiante vindo de qualquer superfície pequena com área  $dA$  na direção  $\phi_N$  relativa a normal da superfície é proporcional a  $\cos \phi_N$



# Reflexão Difusa

- A intensidade de luz nessa direção pode ser calculada fazendo

$$\text{Intensidade} = \frac{\text{energia radiante por unidade}}{\text{área projetada}}$$

$$\approx \frac{\cos \phi_N}{dA \cos \phi_N}$$

= constante

- Portanto a intensidade é igual em todas as direções de visão

# Reflexão Difusa

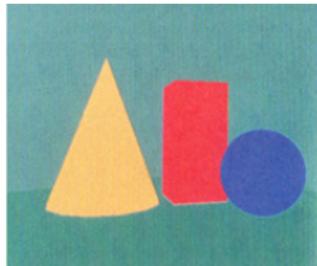
- Assumindo que toda superfície é um refletor difuso ideal (Lambertiniano), um parâmetro  $k_d$  (**coeficiente de reflexão difusa**) pode ser usado para determinar a fração da luz incidente que será espalhada como reflexões difusas
  - A reflexão difusa é constante, igual a luz incidente multiplicada por  $k_d$
- Para fontes de luz monocromáticas,  $0.0 \leq k_d \leq 1.0$ 
  - Superfícies brilhantes apresentam valores de  $k_d$  altos – intensidade de reflexão próxima da incidente
  - Valores de  $k_d$  próximos de 0 definem superfícies que absorvem a luz

# Reflexão Difusa

- Para efeitos de luz de fundo, as superfícies são completamente iluminadas pela luz ambiente  $I_a$  definida para a cena, e a contribuição dessa para reflexão difusa é

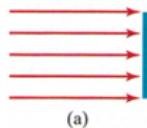
$$I_{ambdiff} = k_d \cdot I_a$$

- Se somente luz ambiente é considerada, o efeito de iluminação obtido em uma cena é pouco interessante

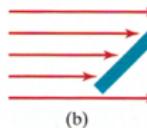


# Reflexão Difusa

- Quando uma superfície é iluminada com intensidade  $I_l$ , a quantidade de **luz incidente** depende da **orientação da superfície** relativa a direção da luz



(a)

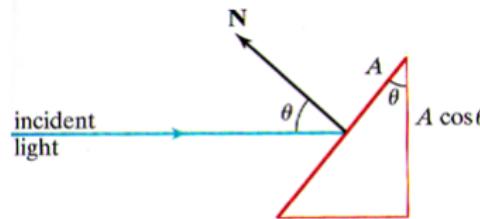


(b)

- O número de raios intersectando a superfície é proporcional a área da projeção perpendicular da superfície na direção da luz incidente

# Reflexão Difusa

- Considerando um **ângulo de incidência**  $\theta$  entre a direção da luz incidente e a normal da superfície



- A área projetada é proporcional a  $\cos \theta$

## Reflexão Difusa

- Assim podemos modelar a quantidade de luz incidente de uma fonte com intensidade  $I_l$  como

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

- Com isso podemos modelar a reflexão difusa de uma fonte de luz com intensidade  $I_l$  como

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

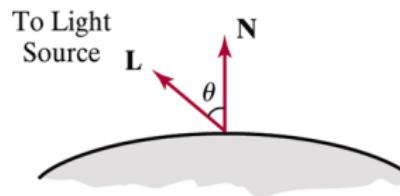
$$= k_d I_l \cos \theta$$

## Reflexão Difusa

- Conforme o **ângulo de incidência aumenta**, a **iluminação** a partir da fonte de luz **diminui**
- Uma superfície somente será iluminada quando  $0^0 \leq \theta \leq 90^0$ , quando  $\cos \theta < 0.0$ , a luz estará atrás da superfície

## Reflexão Difusa

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



- E a equação de reflexão difusa para uma única fonte de luz puntual fica

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

# Reflexão Difusa

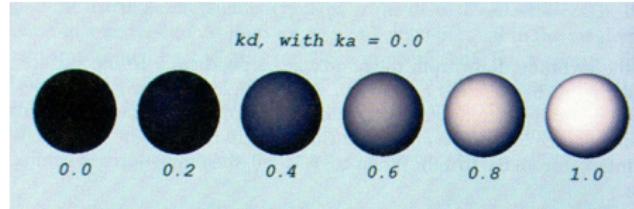
- O vetor unitário  $\mathbf{L}$  é calculado usando a posição da superfície e a posição da fonte de luz

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

- Uma fonte de luz no “infinito” não tem posição, somente a direção de propagação
  - Emprega-se o negativo da direção de emissão para a direção do vetor  $\mathbf{L}$

# Reflexão Difusa

- Exemplo de iluminação difusa variando  $k_d$  entre 0 e 1
  - Uma única fonte de luz puntual
  - Sem luz ambiente



## Reflexão Difusa

- Podemos combinar cálculos de fonte de luz ambiente e puntual para se obter a reflexão difusa em uma posição da superfície
- Introduz o **coeficiente de reflexão ambiente**  $k_a$  para cada superfície para modificar a intensidade  $I_a$  da luz ambiente

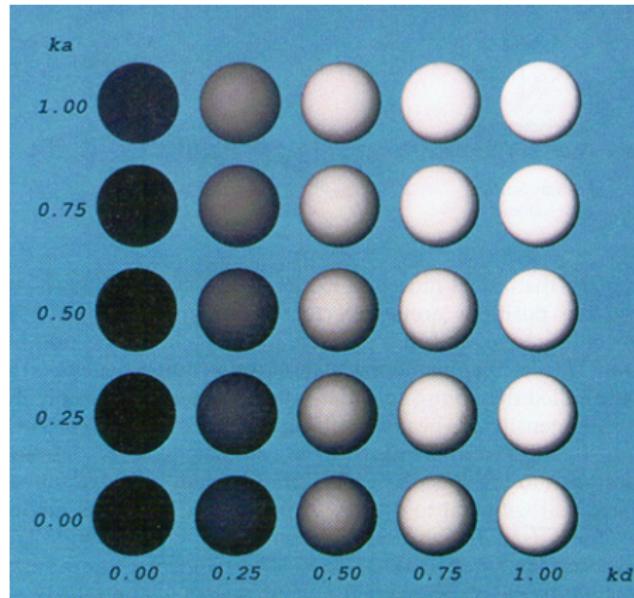
- Podemos escrever a equação de reflexão difusa total de uma única fonte puntual como

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

- Onde  $k_a$  e  $k_d$  dependem das propriedades do material da superfície

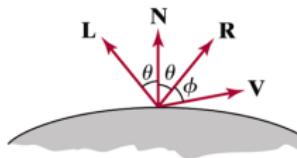
# Reflexão Difusa

- Exemplo de iluminação difusa variando  $k_d$  e  $k_a$



# Reflexão Especular e Modelo de Phong

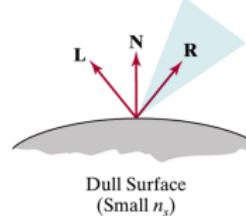
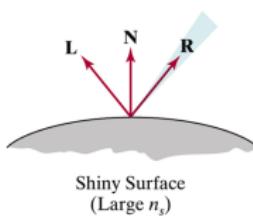
- A **reflexão especular** vista em uma superfície brilhante é o resultado da reflexão total (ou quase) da luz incidente em uma **área concentrada** ao redor de um **ângulo de reflexão especular**



- O ângulo de reflexão especular  $\theta$  é igual ao ângulo de incidência da luz, opostos a normal da superfície **N**
- O vetor unitário **R** representa a direção de reflexão especular
- O vetor unitário **L** aponta na direção da fonte de luz puntual
- O vetor unitário **V** aponta na direção do “visualizador”

# Reflexão Especular e Modelo de Phong

- Em um refletor ideal (espelho perfeito), a luz incidente é refletida somente na direção de reflexão especular, que ocorrerá somente quando  $\mathbf{V}$  e  $\mathbf{R}$  coincidirem ( $\phi = 0^\circ$ )
- Objetos que não são refletores ideais exibem reflexão especular ao redor de  $\mathbf{R}$  em uma região finita
  - Superfícies brilhantes tem um campo menor de reflexão especular
  - Superfícies foscas tem um campo maior de reflexão especular

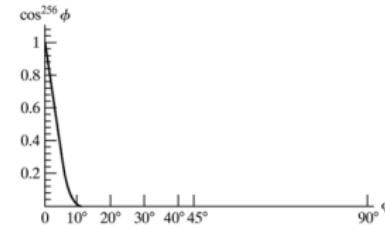
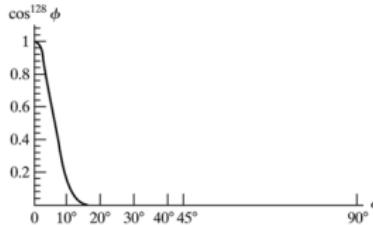
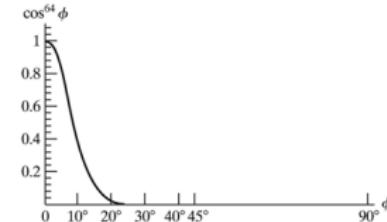
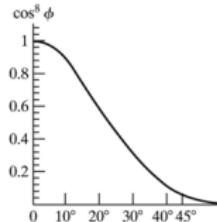
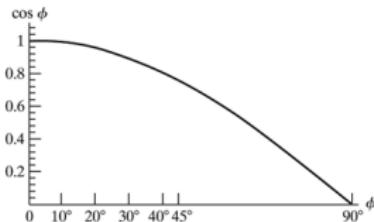


# Reflexão Especular e Modelo de Phong

- O **Modelo de Phong** define a intensidade da reflexão especular proporcional a  $\cos^{n_s} \phi$ , com  $0^0 \leq \phi \leq 90^0$
- O **expoente de reflexão especular**  $n_s$  é determinado pelo tipo de superfície
  - Superfícies brilhantes apresentam valores altos de  $n_s$  (p.ex. 100 ou mais)
  - Para refletores perfeitos  $n_s \rightarrow \infty$

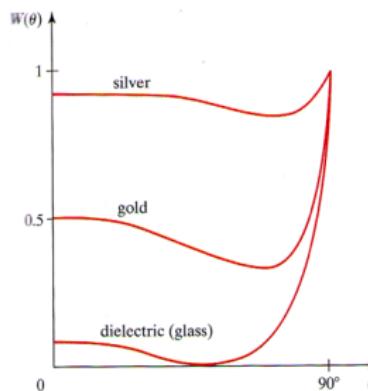
# Reflexão Especular e Modelo de Phong

- Diferentes gráficos de  $\cos^{n_s} \phi$  usando diferentes valores de expoente especular  $n_s$



# Reflexão Especular e Modelo de Phong

- A **intensidade da reflexão** especular depende das propriedades dos **materiais** da superfície e do **ângulo de incidência** ( $\theta$ )
  - É possível aproximar as variações da intensidade especular usando o **coeficiente de reflexão especular**  $W(\theta)$



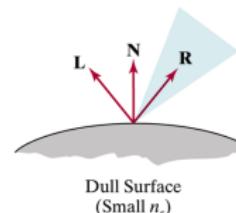
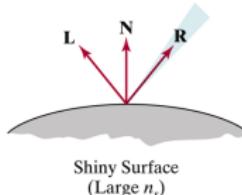
- Em geral  $W(\theta)$  aumenta quando o ângulo de incidência aumenta

# Reflexão Especular e Modelo de Phong

- A variação da intensidade especular é descrita pela **Lei de Reflexão de Fresnel**
- Usando a função  $W(\theta)$  podemos escrever o modelo de Phong de reflexão especular

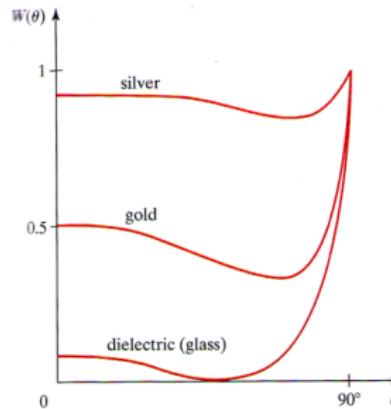
$$I_{l,spec} = W(\theta) I_l \cos^{n_s} \phi$$

- Onde  $I_l$  é a intensidade da fonte de luz e  $\phi$  é o ângulo de visão relativo a  $\mathbf{R}$



# Reflexão Especular e Modelo de Phong

- Para a maioria dos materiais opacos, a reflexão especular é quase constante para todos os ângulos de incidência ( $\theta$ )
  - Podemos substituir  $W(\theta)$  pelo coeficiente de reflexão especular  $k_s$
  - 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  $\mathbf{N}$  efeitos especulares não precisam ser calculados
- 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 de  $\mathbf{L}$  na direção de  $\mathbf{N}$  é  $(\mathbf{N} \cdot \mathbf{cos} \theta)$  (projeção escalar), então

$$\mathbf{R} - \mathbf{S} = \mathbf{N} \cos \theta$$

$$\mathbf{R} = \mathbf{N} \cos \theta + \mathbf{S}$$

$$\mathbf{L} + \mathbf{S} = \mathbf{N} \cos \theta$$

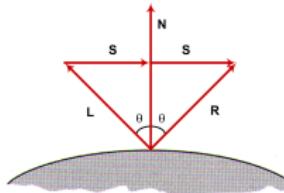
$$\mathbf{S} = \mathbf{N} \cos \theta - \mathbf{L}$$

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

$$\mathbf{R} = \mathbf{N} \cos \theta + \mathbf{N} \cos \theta - \mathbf{L}$$

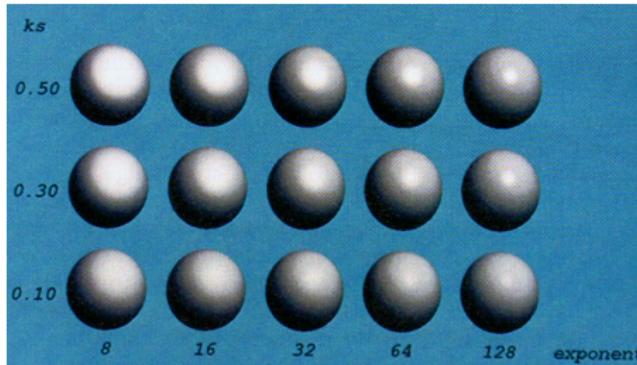
$$= \mathbf{N} 2 \cos \theta - \mathbf{L}$$

$$= \mathbf{N}(2\mathbf{N} \cdot \mathbf{L}) - \mathbf{L}$$



# Reflexão Especular e Modelo de Phong

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



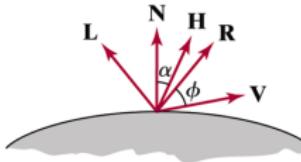
## Reflexão Especular e Modelo de Phong

- O vetor  $\mathbf{V}$  é calculado usando a posição da superfície e a posição de visão da mesma forma como o vetor  $\mathbf{L}$  foi obtido
  - Porém, se apenas uma posição de visão é usada para todas as posições na tela, o cálculo da iluminação especular é acelerado, mas o resultado final é pior

# Reflexão Especular e Modelo de Phong

- Uma simplificação do modelo de Phong é obtida usando o **vetor intermediário H** entre **L** e **V**

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



- Se  $\mathbf{N} \cdot \mathbf{H}$  for usado no lugar de  $\mathbf{V} \cdot \mathbf{R}$  então trocamos o cálculo de  $\cos \phi$  pelo de  $\cos \alpha$
- Se  $\mathbf{V}$  for coplanar a  $\mathbf{L}$  e  $\mathbf{R}$  entre  $\alpha = \phi/2$

# Reflexão Especular e Modelo de Phong

- Para **superfícies não planares**,  $\mathbf{N} \cdot \mathbf{H}$  requer menos cálculos do que  $\mathbf{V} \cdot \mathbf{R}$  porque o cálculo de  $\mathbf{R}$  em cada ponto da superfície envolve o vetor  $\mathbf{N}$
- Além disso, se a posição de **visão e a fonte de luz forem distantes** da superfície,  $\mathbf{V}$  e  $\mathbf{L}$  são constantes, então  $\mathbf{H}$  é constante para todos os pontos da superfície
- $\mathbf{H}$  é a direção da superfície que produzirá a reflexão especular máxima na direção de visão

# Reflexão Especular e Difusa Combinadas

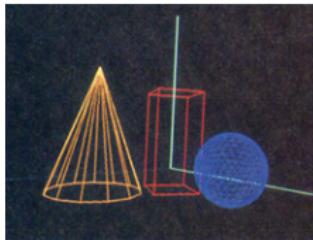
- Para uma única fonte de luz puntual, podemos modelar a combinação das reflexões difusa e especular como

$$I = I_{diff} + I_{spec}$$

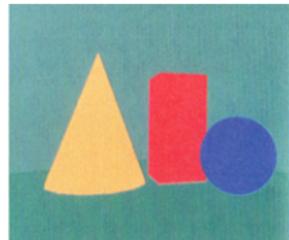
$$= (k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L})) + k_s I_l (\mathbf{N} \cdot \mathbf{H})^{n_s}$$

- A superfície será iluminada somente pela luz ambiente quando a fonte de luz estiver atrás da superfície
- Não existirão efeitos especulares se  $\mathbf{V}$  e  $\mathbf{L}$  estiverem do mesmo lado de  $\mathbf{N}$

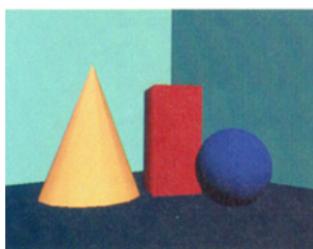
# Reflexão Especular e Difusa Combinadas



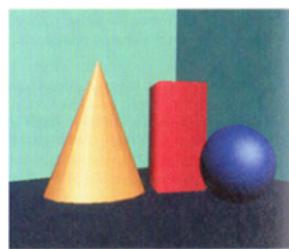
(a)



(b)



(c)



(d)

Figura: Cena wire-frame (a), usando somente luz ambiente (b), reflexão difusa com luz ambiente e uma única fonte de luz puntual (c) e 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 usar a quantidade de fontes de luz que se deseja, para isso soma-se as contribuições de reflexão difusa e especular de cada fonte

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

# Superfícies Emissoras de Luz

- Em uma cena algumas **superfícies** podem **emitir** e **refletir** luz
  - Podemos modelar esse efeito adicionando um termo de emissão  $I_{surfemission}$  a iluminação de fontes diferentes e a iluminação ambiente
- Podemos **simular** esse efeito colocando uma **fonte de luz atrás da superfície** ou por meio de um conjunto de pontos sobre a superfície
- Na prática superfícies emissoras de luz não são usadas devido o **custo computacional alto**

# Modelo de Iluminação Básico com Atenuação da Intensidade e Holofotes

- É possível formular uma modelo de iluminação monocromático geral 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

- Onde

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

(fonte de luz atrás do objeto)

- E

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

(fonte de luz atrás do objeto)

# Considerações sobre Cor RGB

- Para cores RGB, as intensidades são modeladas com vetores de 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 as 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 cada componente de cor da superfície é calculada separadamente

# Considerações sobre Cor RGB

- Para definir a cor de uma superfície, os coeficientes de reflexão são usados
  - Por exemplo, para a definição de uma superfície azul, um valor diferente de zero para a componente  $k_{dB}$  deve ser escolhido enquanto as outras componentes são zeradas  $k_{dR} = k_{dG} = 0.0$
  - Somente a luz azul é refletida, as outras são absorvidas pela superfície

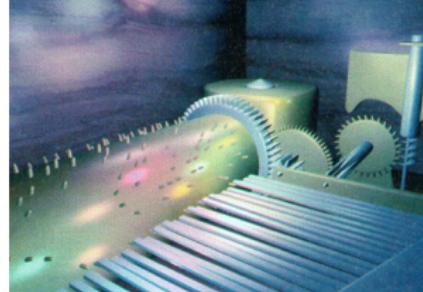
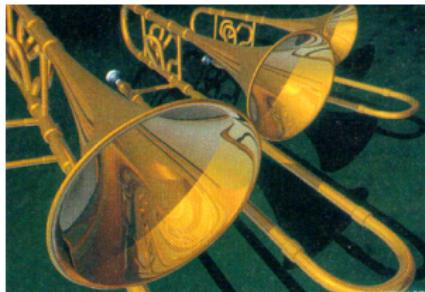
# Considerações sobre Cor RGB

- No modelo original de Phong,  $k_s$  tinha valor constante
  - 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)
- 6 Métodos de Rendering de Superfície
  - Programação OpenGL (Rendering)

# Funções de Iluminação e Rendering OpenGL

- As rotinas de iluminação são ativadas usando

```
1 glEnable(GL_LIGHTING);
```

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

```
1 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\_LIGHT0, GL\_LIGHT1, GL\_LIGHT2,...,GL\_LIGHT7*
- Depois de todos os parâmetros de uma luz terem sido definidos, essa deve ser ligada usando

```
1 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\_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 GLfloat light1_pos[] = {2.0,0.0,3.0,1.0}; //fonte local
2 GLfloat light2_pos[] = {0.0,1.0,0.0,0.0}; //fonte distante
3
4 glLightfv(GL_LIGHT1, GL_POSITION, light1_pos); //define posição da luz
5 glEnable(GL_LIGHT1);
6
7 glLightfv(GL_LIGHT2, GL_POSITION, light2_pos); //define direção da luz
8 glEnable(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 é incluída na descrição da cena, sendo transformada em coordenadas de visão junto com a descrição dos objetos

# Especificando as Cores da Fonte de Luz

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

```
1 GLfloat white[] = {1.0,1.0,1.0,1.0};  
2 GLfloat black[] = {0.0,0.0,0.0,1.0};  
3  
4 glLightfv(GL_LIGHT1,GL_AMBIENT,black); //contribuição ambiente  
5 glLightfv(GL_LIGHT1,GL_DIFFUSE,white); //contribuição difusa  
6 glLightfv(GL_LIGHT1,GL_SPECULAR,white); //contribuição especular
```

- Por padrão, para a luz 0 a propriedade de luz ambiente é preta e branca as as propriedades difusa e especular
  - Para as outras luzes todas as propriedades são pretas

# Especificando as Cores da Fonte de Luz

```
1 void init(void)
2 {
3     glClearColor(1.0,1.0,1.0,1.0); //define a cor de fundo
4     glEnable(GL_DEPTH_TEST); //habilita o teste de profundidade
5
6     glMatrixMode(GL_MODELVIEW); //define que a matrix é a model view
7     glLoadIdentity(); //carrega a matrix de identidade
8     gluLookAt(0.0,1.0,1.0, //posição da câmera
9                0.0,0.0,0.0, //para onde a câmera aponta
10               0.0,1.0,0.0); //vetor view-up
11
12    glMatrixMode(GL_PROJECTION); //define que a matrix é a de projeção
13    glLoadIdentity(); //carrega a matrix de identidade
14    glOrtho(-2.0,2.0,-2.0,2.0,-2.0,2.0);
15 }
```

# Especificando as Cores da Fonte de Luz

```
1 void display(void)
2 {
3     //limpa o buffer
4     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
5
6     //define que a matrix é a de modelo
7     glMatrixMode(GL_MODELVIEW);
8     glutSolidSphere(1.0, 40, 40);
9
10    //força o desenho das primitivas
11    glutSwapBuffers();
12 }
13
14 int main(int argc, char **argv)
15 {
16     glutInit(&argc, argv);
17     glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE | GLUT_DEPTH);
18     glutInitWindowSize(900, 900);
19     glutCreateWindow("3D");
20
21     init();
22     lighting();
23     glutDisplayFunc(&display); //registra função de desenho
24
25     glutMainLoop();
26
27     return 0;
28 }
```

# Especificando as Cores da Fonte de Luz

```
1 void lighting(void)
2 {
3     //define a posição e parâmetros da luz 0
4     GLfloat position[] = {2.0,2.0,2.0,1.0};
5     GLfloat white[] = {1.0,1.0,1.0,1.0};
6     GLfloat black[] = {0.0,0.0,0.0,1.0};
7
8     glLightfv(GL_LIGHT0,GL_POSITION,position);
9     glLightfv(GL_LIGHT0,GL_AMBIENT,black);
10    glLightfv(GL_LIGHT0,GL_DIFFUSE,white);
11    glLightfv(GL_LIGHT0,GL_SPECULAR,white);
12
13    //ativa a iluminação
14    glEnable(GL_LIGHTING);
15    glEnable(GL_LIGHT0);
16 }
```

# 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

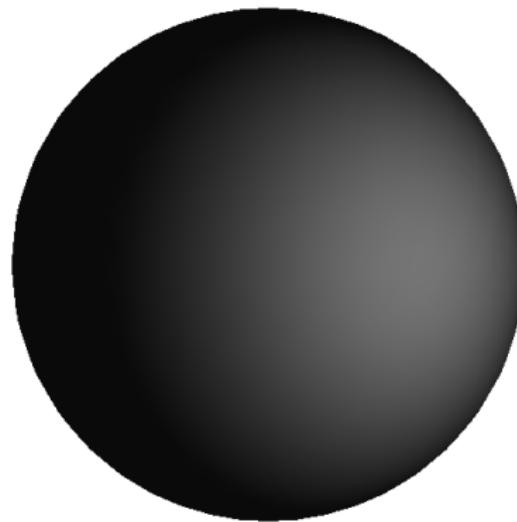
```
1 glLightf(GL_LIGHT0,GL_CONSTANT_ATTENUATION,0.5); //define o coeficiente a0  
2 glLightf(GL_LIGHT0,GL_LINEAR_ATTENUATION,0.15); //define o coeficiente a1  
3 glLightf(GL_LIGHT0,GL_QUADRATIC_ATTENUATION,0.1); //define o coeficiente a2
```

- Os valores dos coeficientes podem ser 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 void lighting(void)
2 {
3     //define a posição e parâmetros da luz 0
4     GLfloat position[] = {2.0,2.0,2.0,1.0};
5     GLfloat white[] = {1.0,1.0,1.0,1.0};
6     GLfloat black[] = {0.0,0.0,0.0,1.0};
7
8     glLightfv(GL_LIGHT0,GL_POSITION,position);
9     glLightfv(GL_LIGHT0,GL_AMBIENT,black);
10    glLightfv(GL_LIGHT0,GL_DIFFUSE,white);
11    glLightfv(GL_LIGHT0,GL_SPECULAR,white);
12
13    //ativa a atenuação
14    glLightf(GL_LIGHT0,GL_CONSTANT_ATTENUATION,0.5);
15    glLightf(GL_LIGHT0,GL_LINEAR_ATTENUATION,0.2);
16    glLightf(GL_LIGHT0,GL_QUADRATIC_ATTENUATION,0.15);
17
18    //ativa a iluminação
19    glEnable(GL_LIGHTING);
20    glEnable(GL_LIGHT0);
21 }
```

# Especificando as Cores da Fonte de Luz



# Fontes de Luz Direcionais (Holofotes)

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

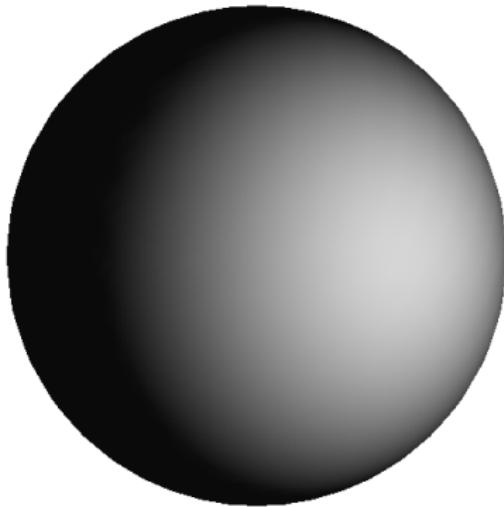
```
1 GLfloat direction[] = {1.0,0.0,0.0};  
2 glLightfv(GL_LIGHT0,GL_SPOT_DIRECTION,direction); //vetor direção  
3 glLightf(GL_LIGHT0,GL_SPOT_CUTOFF,30.0); //espalhamento angular  
4 glLightf(GL_LIGHT0,GL_SPOT_EXPONENT,2.5); //atenuação angular
```

- O valor de  $\theta_l$  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 void lighting(void)
2 {
3     //define a posição e parâmetros da luz 0
4     GLfloat position0[] = {2.0,2.0,2.0,1.0};
5     GLfloat white[] = {1.0,1.0,1.0,1.0};
6     GLfloat black[] = {0.0,0.0,0.0,1.0};
7
8     glLightfv(GL_LIGHT0,GL_POSITION,position0);
9     glLightfv(GL_LIGHT0,GL_AMBIENT,black);
10    glLightfv(GL_LIGHT0,GL_DIFFUSE,white);
11    glLightfv(GL_LIGHT0,GL_SPECULAR,white);
12
13    //define a posição e parâmetros da luz 1
14    GLfloat position1[] = {-2.0,0.0,0.0,1.0};
15    glLightfv(GL_LIGHT1,GL_POSITION,position1);
16    glLightfv(GL_LIGHT1,GL_DIFFUSE,white);
17    glLightfv(GL_LIGHT1,GL_SPECULAR,white);
18
19    GLfloat direction[] = {1.0,0.0,0.0,0.0};
20    glLightfv(GL_LIGHT1,GL_SPOT_DIRECTION,direction); //vetor direção
21    glLightf(GL_LIGHT1,GL_SPOT_CUTOFF,45.0); //espalhamento angular
22    glLightf(GL_LIGHT1,GL_SPOT_EXPONENT,2.5); //atenuação angular
23
24    //ativa a iluminação
25    glEnable(GL_LIGHTING);
26    glEnable(GL_LIGHT0);
27    glEnable(GL_LIGHT1);
28 }
```

# Fontes de Luz Direcionais (Holofotes)



# Parâmetros de Iluminação Global

- Vários parâmetros de luz podem ser definidos globalmente usando

```
1 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

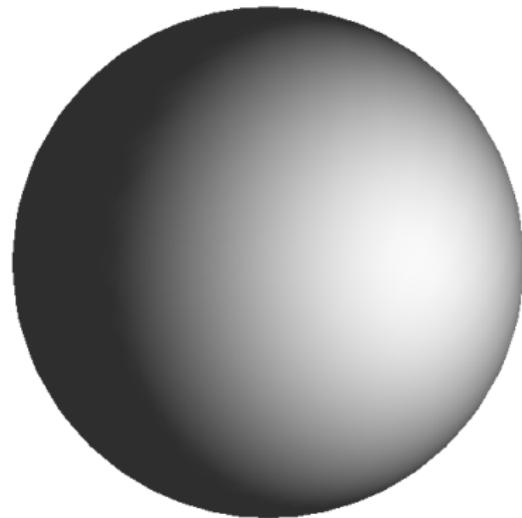
```
1 GLfloat global_ambient[]={0.9,0.9,0.9,1.0};  
2 glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
```

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

# Parâmetros de Iluminação Global

```
1 void lighting(void)
2 {
3     //define a posição e parâmetros da luz 0
4     GLfloat position[] = {2.0,2.0,2.0,1.0};
5     GLfloat white[] = {1.0,1.0,1.0,1.0};
6     GLfloat black[] = {0.0,0.0,0.0,1.0};
7
8     glLightfv(GL_LIGHT0,GL_POSITION,position);
9     glLightfv(GL_LIGHT0,GL_AMBIENT,black);
10    glLightfv(GL_LIGHT0,GL_DIFFUSE,white);
11    glLightfv(GL_LIGHT0,GL_SPECULAR,white);
12
13    //ativando luz ambiente global
14    GLfloat global_ambient[]={0.9,0.9,0.9,1.0};
15    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
16
17    //ativa a iluminação
18    glEnable(GL_LIGHTING);
19    glEnable(GL_LIGHT0);
20 }
```

# Parâmetros de Iluminação Global



# Parâmetros de Iluminação Global

- Para o cálculo da reflexão especular é necessário determinar o vetor **V** (da superfície para a posição de visão)
  - Podemos acelerar o processamento fazendo **V** constante independente da posição da superfície
- O valor padrão para **V** é a direção de *z* (0.0, 0.0, 1.0), mas podemos usar a posição de visão corrente fazendo

```
1 glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, GL_TRUE);
```

- Isso tornará o processo computacionalmente mais caro, mas o resultado será mais realístico
  - Para desabilitar o cálculo de **V** fazer essa função igual a *GL\_FALSE*

# Parâmetros de Iluminação Global

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

```
1 glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
```

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

# Propriedades da Superfície

- As propriedades óticas das superfícies são definidas usando

```
1 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\_FRONT*, *GL\_BACK* e *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

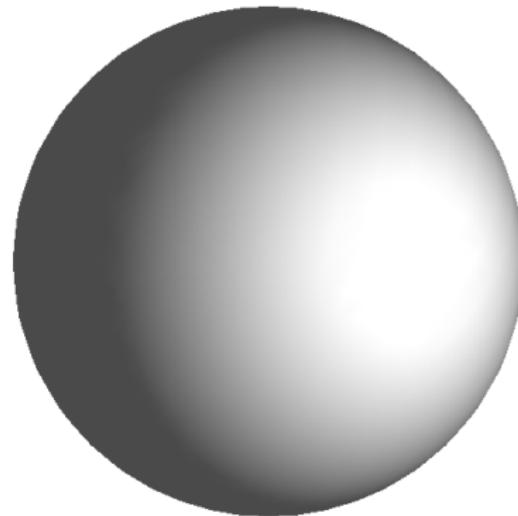
```
1 GLfloat surface_emission[] = {0.25,0.25,0.25,1.0};  
2 glMaterialfv(GL_FRONT, GL_EMISSION, surface_emission);
```

- 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 void lighting(void)
2 {
3     //define a posição e parâmetros da luz 0
4     GLfloat position[] = {2.0,2.0,2.0,1.0};
5     GLfloat white[] = {1.0,1.0,1.0,1.0};
6     GLfloat black[] = {0.0,0.0,0.0,1.0};
7
8     glLightfv(GL_LIGHT0,GL_POSITION,position);
9     glLightfv(GL_LIGHT0,GL_AMBIENT,black);
10    glLightfv(GL_LIGHT0,GL_DIFFUSE,white);
11    glLightfv(GL_LIGHT0,GL_SPECULAR,white);
12
13    //superficie vai emitir luz
14    GLfloat surface_emission[] = {0.25,0.25,0.25,1.0};
15    glMaterialfv(GL_FRONT, GL_EMISSION, surface_emission);
16
17    //ativa a iluminação
18    glEnable(GL_LIGHTING);
19    glEnable(GL_LIGHT0);
20 }
```

# Propriedades da Superfície



# Propriedades da Superfície

- Os flags  $GL\_AMBIENT$ ,  $GL\_DIFFUSE$  e  $GL\_SPECULAR$  são usados para definir os coeficientes de reflexão da superfície
  - Normalmente os valores dos coeficientes difuso e ambiente são os mesmos, para isso usamos  $GL\_AMBIENT\_AND\_DIFFUSE$
- 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  $GL\_SHININESS$  entre 0 e 128
  - O valor padrão é 0

# Propriedades da Superfície

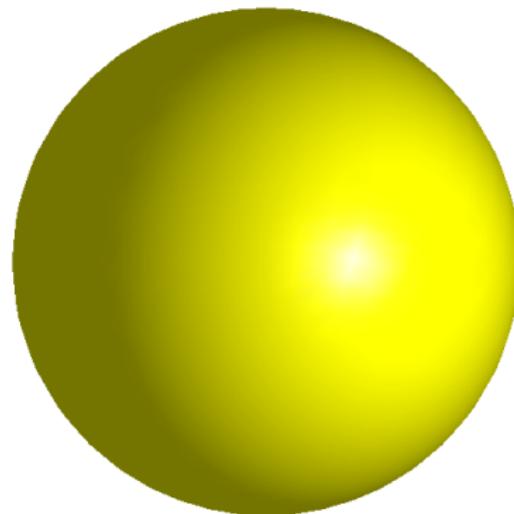
- Por exemplo, para definir os coeficientes de reflexão ambiente, difusa e especular podemos fazer

```
1 GLfloat diffuse[] = {0.65,0.65,0.0,1.0};  
2 GLfloat specular[] = {0.9,0.9,0.9,1.0};  
3 GLfloat shininess = 25.0;  
4  
5 glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,diffuse);  
6 glMaterialfv(GL_FRONT,GL_SPECULAR,specular);  
7 glMaterialf(GL_FRONT,GL_SHININESS,shininess);
```

# Parâmetros de Iluminação Global

```
1 void lighting(void)
2 {
3     //define a posição e parâmetros da luz 0
4     GLfloat position[] = {2.0,2.0,2.0,1.0};
5     GLfloat white[] = {1.0,1.0,1.0,1.0};
6     GLfloat black[] = {0.0,0.0,0.0,1.0};
7
8     glLightfv(GL_LIGHT0,GL_POSITION,position);
9     glLightfv(GL_LIGHT0,GL_AMBIENT,black);
10    glLightfv(GL_LIGHT0,GL_DIFFUSE,white);
11    glLightfv(GL_LIGHT0,GL_SPECULAR,white);
12
13    //ativando luz ambiente global
14    GLfloat global_ambient[]={0.7,0.7,0.7,1.0};
15    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);
16
17    //define as propriedades de reflexão da superfície
18    GLfloat diffuse[] = {0.65,0.65,0.0,1.0};
19    GLfloat specular[] = {0.9,0.9,0.9,1.0};
20    GLfloat shininess = 65.0;
21
22    glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,diffuse);
23    glMaterialfv(GL_FRONT,GL_SPECULAR,specular);
24    glMaterialf(GL_FRONT,GL_SHININESS,shininess);
25
26    //ativa a iluminação
27    glEnable(GL_LIGHTING);
28    glEnable(GL_LIGHT0);
29 }
```

# Parâmetros de Iluminação Global



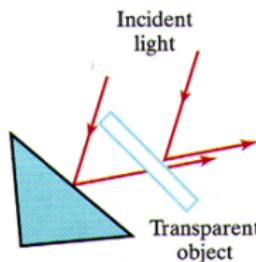
# Sumário

- 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)
- 6 Métodos de Rendering de Superfície
  - Programação OpenGL (Rendering)

# Luminância

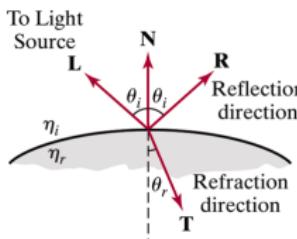
## Superfícies Transparentes

- Objetos **transparentes** (como vidro) devem permitir que seja possível ver outros objetos através desses, o contrário são os objetos **opacos**
- Objetos intermediários, onde a luz é transmitida e difundida em todas as direções são conhecidos como **translúcidos** (p.ex. plástico)



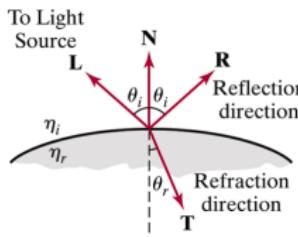
# Refração da Luz

- **Imagens realísticas** de materiais transparentes são 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 velocidade da luz depende do material, o caminho da luz refratada é diferente da luz incidente



# Refração da Luz

- A direção da refração, dada pelo **ângulo de refração** ( $\theta_r$ ), é função do **índice de refração** do material e da direção da luz incidente
  - O índice de refração é a razão da velocidade da luz no vácuo sobre a velocidade da luz no material

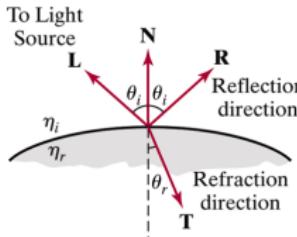


# Refração da Luz

- O ângulo de refração  $\theta_r$  é calculado usando a **lei de Snell**

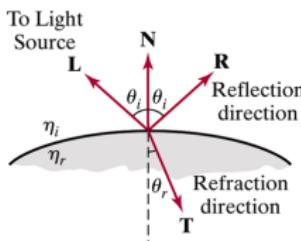
$$\sin \theta_r = \frac{\eta_i}{\eta_r} \sin \theta_i$$

- Onde  $\theta_i$  é o ângulo de incidência,  $\eta_i$  é o índice de refração do material incidente e  $\eta_r$  é o índice de refração do material refratado



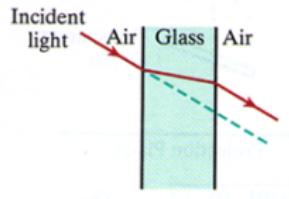
# Refração da Luz

- 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  $\theta_r = 18^0$



## Refração da Luz

- O efeito da refração é mudar o caminho da luz incidente para um caminho um pouco diferente mas paralelo ao original



- A equação de refração é computacionalmente cara de ser avaliada, então normalmente aproximações são empregadas

## Refração da Luz

- O vetor de transmissão **T** na direção de refração  $\theta_r$  é calculado como

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

- Onde **N** é a normal a superfície, **L** é a direção da fonte de luz
- T** pode ser usado para localizar intersecções do caminho de refração com objetos atrás da superfície transparente

## Modelo Básico de Transparência

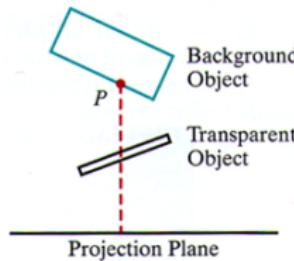
- Um modelo simples de refração pode ignorar a mudança do caminho da luz
  - Não existe mudança de índice de refração entre materiais diferentes
  - O ângulo de incidência será igual ao ângulo de refração
- Acelera o processamento e produz bons resultados para superfícies finas

## 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**  $k_t$

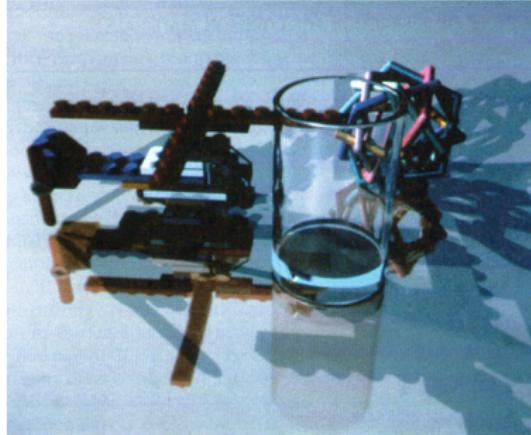
$$I = (1 - k_t) I_{refl} + k_t I_{trans}$$

- Onde o temos  $(1 - k_t)$  é o fator de opacidade



## Modelo Básico de Transparência

- Esse procedimento pode ser usado com qualquer quantidade 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)
- 6 Métodos de Rendering de Superfície
  - Programação OpenGL (Rendering)

# 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*
  - Efeitos de refração serão ignorados
  - 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 *glMaterial(...)* e *glColor(...)*
- Por exemplo, usando a função

```
1 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 glEnable(GL_BLEND);
2 glBlendFunc(GL_ONE_MINUS_SRC_ALPHA, 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  $A = k_t$

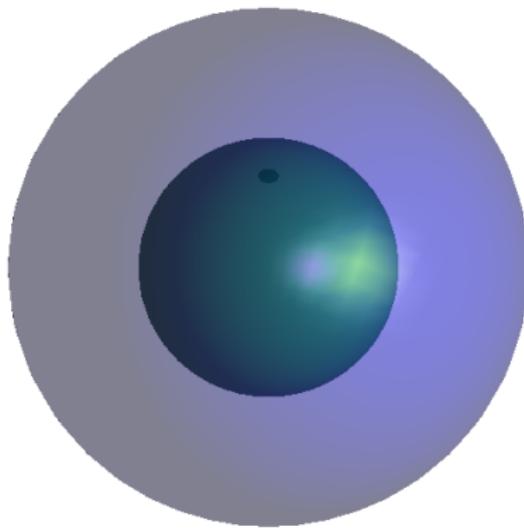
# Funções de Transparências

```
1 void lighting(void)
2 {
3     //define a posição e parâmetros da luz 0
4     GLfloat position[] = {2.0,2.0,2.0,1.0};
5     GLfloat white[] = {1.0,1.0,1.0,1.0};
6     GLfloat black[] = {0.0,0.0,0.0,1.0};
7
8     glLightfv(GL_LIGHT0,GL_POSITION,position);
9     glLightfv(GL_LIGHT0,GL_AMBIENT,black);
10    glLightfv(GL_LIGHT0,GL_DIFFUSE,white);
11    glLightfv(GL_LIGHT0,GL_SPECULAR,white);
12
13    //ativa a iluminação
14    glEnable(GL_LIGHTING);
15    glEnable(GL_LIGHT0);
16
17    //ativa transparência
18    glEnable(GL_BLEND);
19    glBlendFunc(GL_ONE_MINUS_SRC_ALPHA, GL_SRC_ALPHA);
20 }
```

# Funções de Transparências

```
1 void display(void)
2 {
3     //limpa o buffer
4     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
5
6     //define que a matrix é a de modelo
7     glMatrixMode(GL_MODELVIEW);
8
9     //desenho a esfera de dentro
10    GLfloat diffuse1[] = {0.0,0.65,0.0,0.25};
11    GLfloat specular1[] = {0.9,0.9,0.9,0.25};
12    GLfloat shininess1 = 65.0;
13    glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,diffuse1);
14    glMaterialfv(GL_FRONT,GL_SPECULAR,specular1);
15    glMaterialf(GL_FRONT,GL_SHININESS,shininess1);
16    glutSolidSphere(0.5,40,40);
17
18     //desenho a esfera de fora
19    GLfloat diffuse2[] = {0.0,0.0,0.65,0.5};
20    GLfloat specular2[] = {0.9,0.9,0.9,0.5};
21    GLfloat shininess2 = 65.0;
22    glMaterialfv(GL_FRONT,GL_AMBIENT_AND_DIFFUSE,diffuse2);
23    glMaterialfv(GL_FRONT,GL_SPECULAR,specular2);
24    glMaterialf(GL_FRONT,GL_SHININESS,shininess2);
25    glutSolidSphere(1.0,40,40);
26
27     //força o desenho das primitivas
28     glutSwapBuffers();
29 }
```

# Funções de Transparências



# Sumário

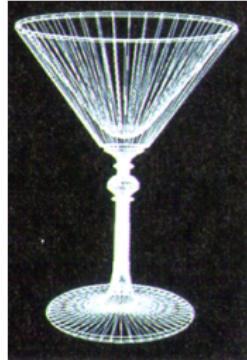
- 1 Introdução
- 2 Fontes de Luz
- 3 Efeitos de Luz em Superfícies
- 4 Modelos Básicos de Iluminação
  - Programação OpenGL (Iluminação)
- 5 Superfícies Transparentes
  - Programação OpenGL (Transparência)
- 6 Métodos de Rendering de Superfície
  - Programação OpenGL (Rendering)

# Métodos de Rendering de Superfície

- O modelo de iluminação pode ser usado de formas diferentes para definir a cor de uma superfície
  - Executado em cada pixel projetado (realismo)
  - Executado em alguns pixels e interpolado no restante (tempo real)
  
- Maioria das APIs gráficas reduz o processamento usando algoritmos de *scan-line*
  - As intensidades são calculadas nos vértices e interpoladas nas posições restantes dos polígonos

# Rendering de Superfície de Intensidade Constante

- O método mais simples (e rápido) para renderizar uma superfície poligonal é usar a mesma cor para todos pontos da superfície (**flat surface rendering**)
- Emprega-se o modelo de iluminação para determinar a intensidade das 3 componentes RGB em uma única posição da superfície
  - Vértice ou centróide do polígono



(a)



(b)

# Rendering de Superfície de Intensidade Constante

- O **flat surface rendering** normalmente define resultados preciso se
  - O polígono é uma face de um poliedro e **não** uma seção de uma **superfície curva**
  - Todas as **fontes de luz** estão **distantes** o suficiente da superfície de forma que  $\mathbf{N} \cdot \mathbf{L}$  e a função de atenuação são **constantes**
  - A **posição visão** é **distante** o suficiente do polígono de forma que  $\mathbf{V} \cdot \mathbf{R}$  é constante
- Mesmo se alguma dessas condições for falsa, uma boa aproximação pode ser conseguida se os polígonos empregados forem pequenos

# Rendering de Superfície de Gouraud

- O esquema de **Gouraud surface rendering** interpola linearmente as intensidades nos vértices por toda face do polígono de um objeto iluminado
- Desenvolvido para aproximar superfícies curvas, amenizando as transições de intensidades entre polígonos adjacentes
  - Elimina as descontinuidades de intensidades do **flat surface rendering**

# Rendering de Superfície de Gouraud

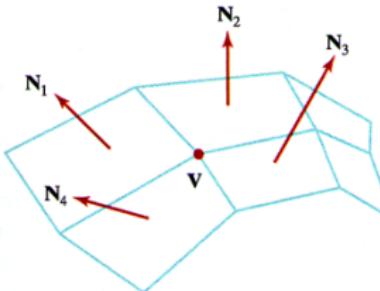
- Cada polígono de uma superfície curva é processado usando o seguinte procedimento
  - ① Determina o vetor unitário normal médio em cada vértice do polígono
  - ② Aplica o modelo de iluminação em cada vértice para obter as intensidades
  - ③ Interpola linearmente as intensidades dos vértices sobre a área projetada do polígono

# Rendering de Superfície de Gouraud

- O vetor unitário normal médio em um vértice é obtido fazendo a média das normais de todos os polígonos que compartilham esse vértice

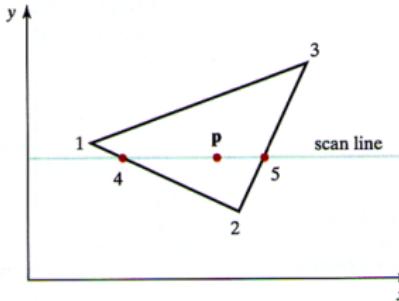
$$\mathbf{N}_V = \frac{\sum_{k=1}^n \mathbf{N}_k}{|\sum_{k=1}^n \mathbf{N}_k|}$$

- Usando essas normais o modelo de iluminação é então executado para calcular as intensidades em cada vértice



# Rendering de Superfície de Gouraud

- Esses valores de intensidade são então interpolados para se obter as intensidades ao longo de *scan-lines* que intersectam a área projetada do polígono

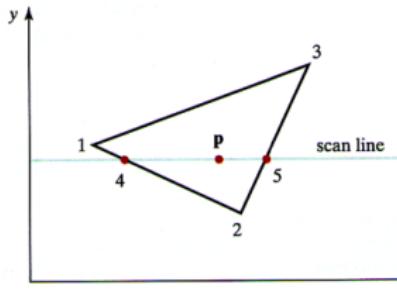


- As intensidades das intersecções das *scan-lines* com as arestas dos polígonos são calculadas interpolando linearmente as intensidades dos pontos finais das arestas

# Rendering de Superfície de Gouraud

- Por exemplo, a intensidade em 4 pode ser calculada considerando somente o deslocamento vertical da *scan-line*

$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2$$

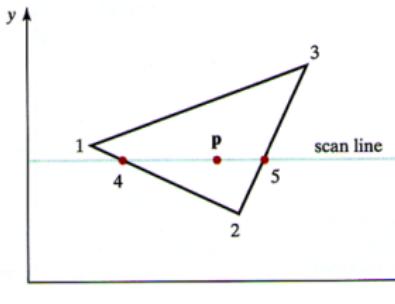


- A intensidade em 5 pode ser obtida da mesma forma interpolando as intensidades em 2 e 3

# Rendering de Superfície de Gouraud

- Considerando as intensidades obtidas em 4 e 5, as intensidades em qualquer ponto  $p$  da *scan-line* são obtidas fazendo

$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5$$



- Esse método é conhecido como interpolação bilinear e é executado para os 3 componentes RGB separadamente

# Rendering de Superfície de Gouraud

- Essa interpolação de intensidades elimina descontinuidades mas tem alguns **problemas**
  - Brilhos na superfície podem apresentar formatos estranhos
  - Intensidades claras ou escuras podem parecer “riscadas” (**mach bands**)



(a)



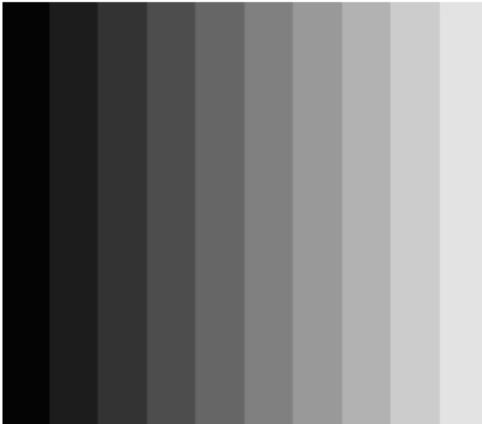
(b)



(c)

# Rendering de Superfície de Gouraud

- Efeito de **mach bands** consiste em faixas claras ou escuras que são percebidas próximo das fronteiras entre duas regiões de que diferentes gradientes de luz



# Rendering de Superfície de Phong

- Um método mais preciso de interpolação é conhecido como **Phong surface rendering**
- Ao invés de interpolar valores de intensidades, normais são interpoladas
  - Cálculos mais precisos de intensidades
  - Brilhos mais realísticos nas superfícies
  - Redução do efeito *mach-band*
- Computacionalmente mais caro que o método de Gouraud

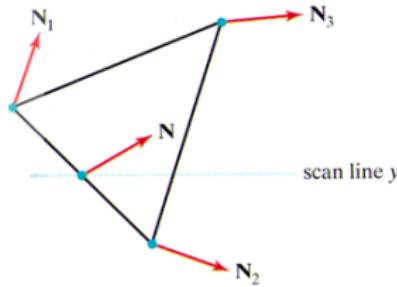
# Rendering de Superfície de Phong

- Cada polígono é processado usando o seguinte procedimento
  - ➊ Determina o vetor unitário normal médio em cada vértice do polígono
  - ➋ Interpola linearmente as normais dos vértices sobre a área projetada do polígono
  - ➌ Aplica o modelo de iluminação nas posições ao longo da *scan-line* para calcular as intensidades dos pixels usando as normais interpoladas

# Rendering de Superfície de Phong

- O procedimento de interpolação das normais é o mesmo da interpolação das intensidades do método de Gouraud
- Por exemplo, o vetor normal  $\mathbf{N}$  é verticalmente interpolado a partir das normais nos vértices 1 e 2 fazendo

$$\mathbf{N} = \frac{y - y_2}{y_1 - y_2} \mathbf{N}_1 + \frac{y_1 - y}{y_1 - y_2} \mathbf{N}_2$$



# 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)
- 6 Métodos de Rendering de Superfície
  - Programação OpenGL (Rendering)

# Funções de Rendering de Superfície

- Podemos usar dois métodos de rendering da superfície: (1) de intensidade constante; (2) e o modelo de Gouraud
  - Não existe suporte para o modelo de Phong
- Para definir o método de rendering usamos

1    `glShadeModel(rendering_method);`

- Onde *rendering\_method* pode ser *GL\_FLAT* e *GL\_SMOOTH*

# Funções de Rendering de Superfície

- Para se definir a normal à superfície usamos

```
1 glNormal3*(Nx, Ny, Nz);
```

- Com o sufixo dependendo do tipo de parâmetro *b*, *s*, *i*, *f* e *d*, ou com a adição de *v* caso o parâmetro seja um vetor
  - Valores *byte*, *short* e *integer* são convertidos para valores de ponto flutuante na faixa de -1.0 a 1.0
- A normal a superfície é um valor de estado da OpenGL e tem valor padrão igual a (0.0, 0.0, 1.0)

# Funções de Rendering de Superfície

- Para rendering de intensidade constante definimos apenas uma normal para cada polígono

```
1  glNormal3fv(normal_vector);
2
3  glBegin(GL_TRIANGLES);
4      glVertex3fv(vertex1);
5      glVertex3fv(vertex2);
6      glVertex3fv(vertex3);
7  glEnd();
```

# Funções de Rendering de Superfície

- Para o rendering de Gouraud uma normal deve ser definida para cada vértice

```
1 glBegin(GL_TRIANGLES);
2     glNormal3fv(normal_vector1);
3     glVertex3fv(vertex1);
4
5     glNormal3fv(normal_vector2);
6     glVertex3fv(vertex2);
7
8     glNormal3fv(normal_vector3);
9     glVertex3fv(vertex3);
10    glEnd();
```

# Funções de Rendering de Superfície

- Apesar dos vetores normais não precisarem ser especificados com tamanho unitário, fazendo isso reduzimos o custo computacional
- É possível solicitar a OpenGL normalizar qualquer vetor normal que não seja unitário chamando

```
1 glEnable(GL_NORMALIZE);
```

- Esse comando renormaliza todas as normais às superfícies incluindo as que foram modificadas por transformações geométricas de escala e cisalhamento

# Funções de Rendering de Superfície

```
1 void lighting(void)
2 {
3     //define a posição e parâmetros da luz 0
4     GLfloat position[] = {2.0,2.0,2.0,1.0};
5     GLfloat white[] = {1.0,1.0,1.0,1.0};
6     GLfloat black[] = {0.0,0.0,0.0,1.0};
7
8     glLightfv(GL_LIGHT0,GL_POSITION,position);
9     glLightfv(GL_LIGHT0,GL_AMBIENT,black);
10    glLightfv(GL_LIGHT0,GL_DIFFUSE,white);
11    glLightfv(GL_LIGHT0,GL_SPECULAR,white);
12
13    //superfície vai emitir luz
14    GLfloat surface_emission[] = {0.25,0.25,0.25,1.0};
15    glMaterialfv(GL_FRONT, GL_EMISSION, surface_emission);
16
17    //ativa a iluminação
18    glEnable(GL_LIGHTING);
19    glEnable(GL_LIGHT0);
20
21    //ativa o rendering de intensidade constante
22    glShadeModel(GL_FLAT);
23 }
```

# Funções de Rendering de Superfície

