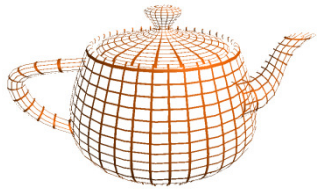
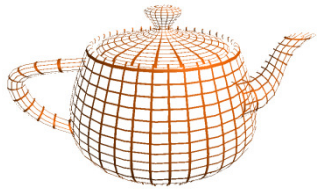
Computação Gráfica

Iluminação



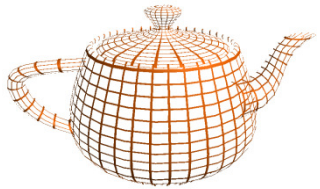
Iluminação

- Tópicos:
  - Fundamentos de iluminação
    - Aplicação em OpenGL
  - Modelos de Shading
  - Materiais em OpenGL
  - Iluminação em OpenGL



Iluminação

- Em CG a iluminação simula como os objectos reflectem a luz em tempo real.
- É uma aproximação empírica da iluminação real, por vezes sem profundas bases teóricas que as sustentem, no entanto com resultados práticos bastante aceitáveis.
- Esta aproximação deve-se a dois factores:
  - 1. As equações da iluminação real não são totalmente conhecidas
  - 2. Mesmo os modelos simplificados da realidade são extremamente complexos



Iluminação - Fundamentos

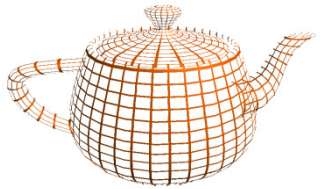
- A versão mais simples consiste em utilizar iluminação ambiente
- Todos os objectos têm uma

  intensidade intrínseca.

- Para cada objecto é definida uma constante, que indica a intensidade intrínseca.

  $I = K_a$

- Objectos mais claros têm uma intensidade intrínseca elevada, enquanto que objectos escuros têm um intensidade intrínseca reduzida.



Iluminação - Fundamentos

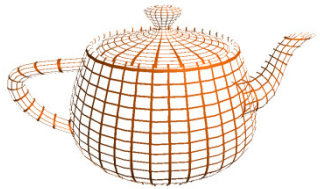
- Iluminação ambiente.

- Esta forma iluminação simula de uma forma básica as reflexões que a luz realiza, iluminando todos os objectos por igual.

$$I = K_a * I_a$$

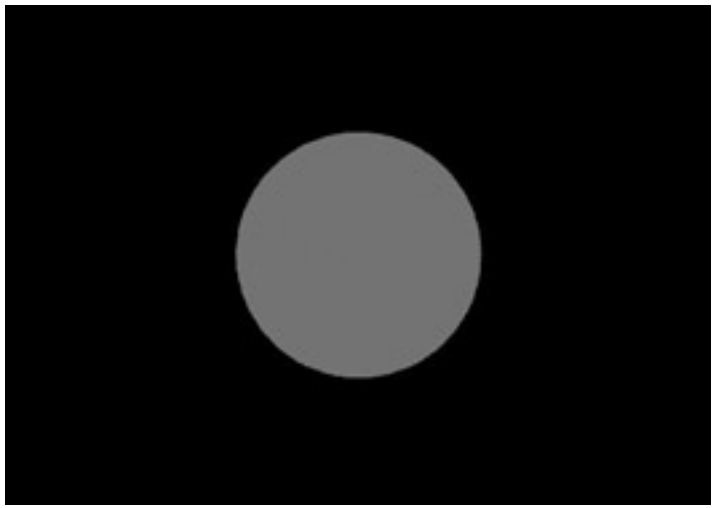
Propriedade do material

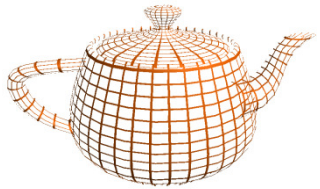
Propriedade da luz



Iluminação - Fundamentos

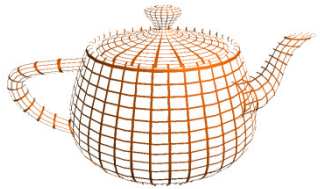
- As equações até agora apresentadas não têm em consideração a posição e orientação da fonte de luz.
- Todos os objectos são iluminados de forma uniforme independentemente da sua posição ou orientação.
- A própria luz não tem posição e/ou orientação definidas.





Iluminação - Fundamentos

- Consideremos agora uma luz com uma posição determinada que emite luz em todas as direcções, por ex: lâmpada.
- A luz emite raios de luz uniformemente distribuídos.
- Podemos ter em conta dois factores:
  - orientação do objecto em relação à luz
  - distância à luz



Iluminação - Fundamentos

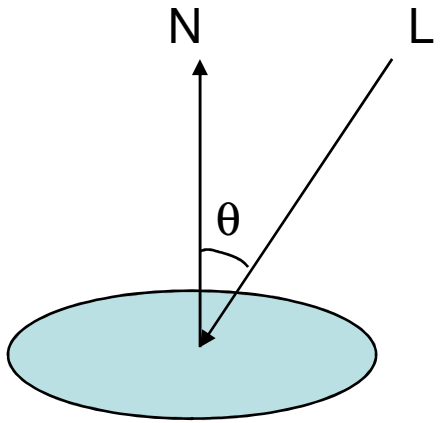
- Reflexão difusa (Lambert)

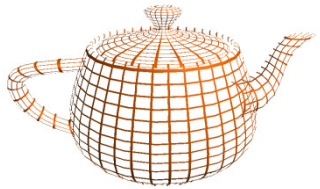
  - A intensidade de um objecto é proporcional ao ângulo entre a direcção da luz e a normal do objecto.

Intensidade da luz

$$I = I_p * K_d * \cos(\theta)$$

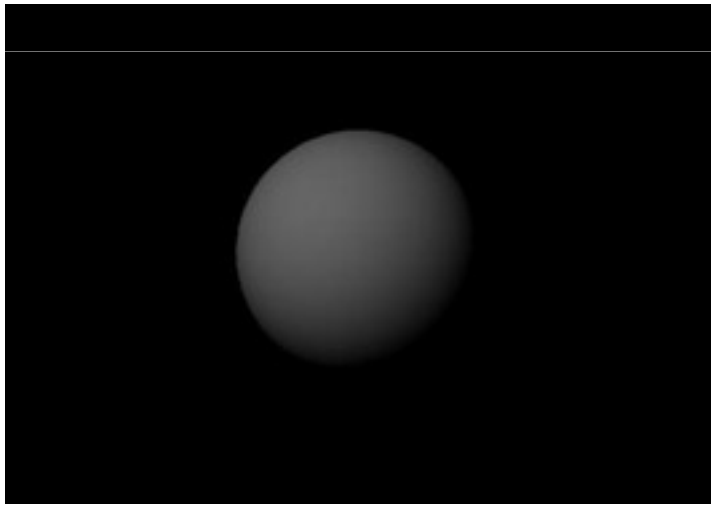
coeficiente de reflexão difusa



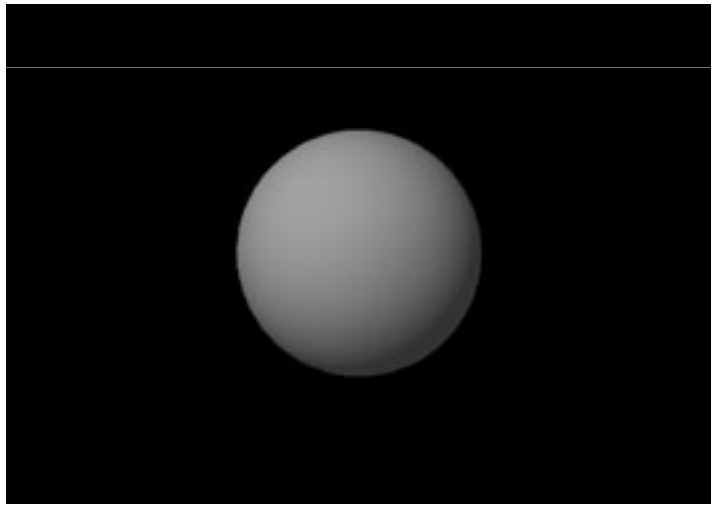


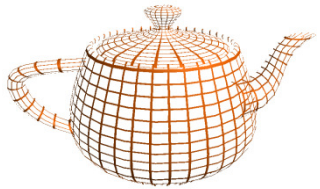
Iluminação - Fundamentos

iluminação difusa



difusa + ambiente





Iluminação - Fundamentos

- Ficamos portanto com:

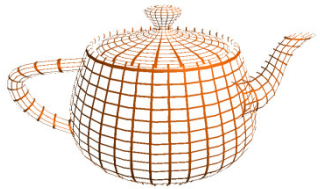
$$I = I_a * K_a + I_p * K_d * \cos(\theta)$$

- Caso ambos os vectores estejam normalizados pode-se substituir o coseno pelo produto interno,

$$I = I_a * K_a + I_p * K_d * (N \cdot L)$$

Produto Interno N.L $= n_x * l_x + n_y * l_y + n_z * l_z$

Nota: só se consideram valores positivos do produto interno ou coseno



Iluminação - Fundamentos

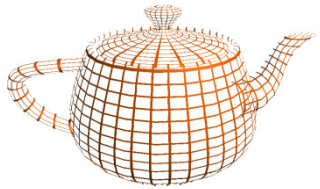
- Atenuação baseada na distância

$$I = I_a * K_a + \boxed{f_{att}} * I_p * K_d * (N \cdot L)$$

- A escolha *correcta* seria:

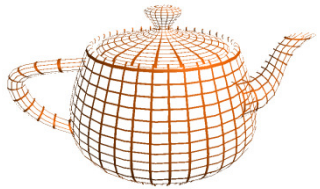
$$f_{att} = 1/d^2 \longleftarrow \text{distância à luz}$$

- Ou seja, a intensidade é inversamente proporcional ao quadrado da distância entre o objecto e a fonte de luz



Iluminação - Fundamentos

- A escolha *correcta* no entanto não produz os efeitos desejados.
  - Luz próxima => grandes variações
  - Luz distante => variações demasiado pequenas
- A realidade não se resume a uma fonte de luz sem interacção entre os objectos.
- Numa situação real a luz reflectida por um objecto tem influência nos outros objectos.



Iluminação - Fundamentos

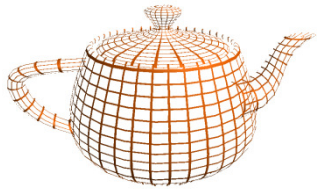
- Uma solução de compromisso é:

$$f_{att} = \min(1/(c_1 + c_2*d + c_3*d^2) , 1)$$

c1,c2,c3: constantes associadas à fonte de luz.

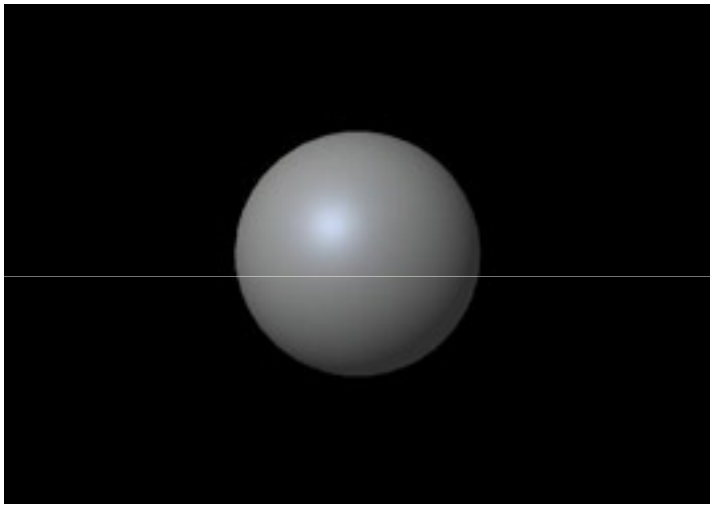
c1: constante que evita que o denominador fique muito pequeno quando a luz está muito perto.

O cálculo do mínimo obriga a diminuir ou manter a intensidade, i.e. f_{att} , <= 1.

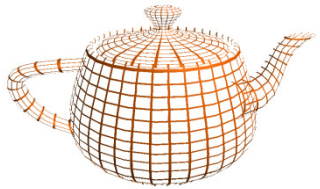


Iluminação - Fundamentos

- Reflexão Especular - Phong



- Ao iluminar materiais brilhantes verifica-se uma mancha brilhante cuja posição depende da posição da câmara.



Iluminação - Fundamentos

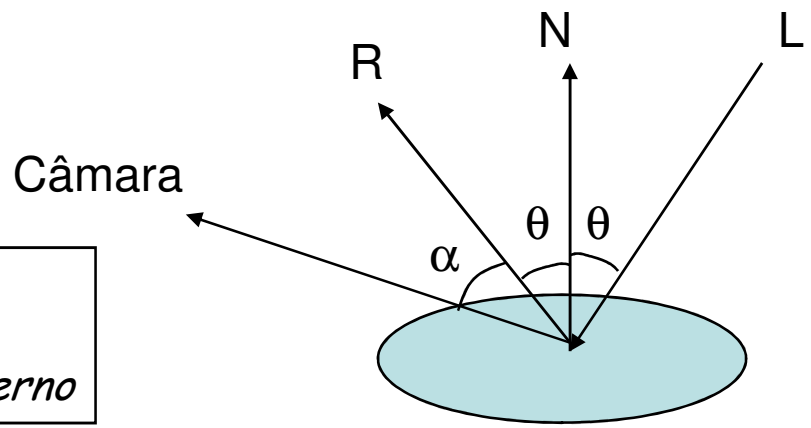
- $I = I_a * K_a + f_{att} * [I_d * K_d * (N \cdot L) + I_s * K_s * (R.\text{Câmara})^s]$

ambiente

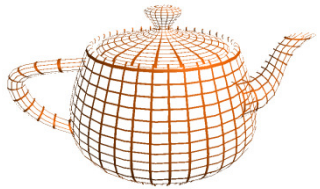
difusa

Coeficiente de especularidade

especular

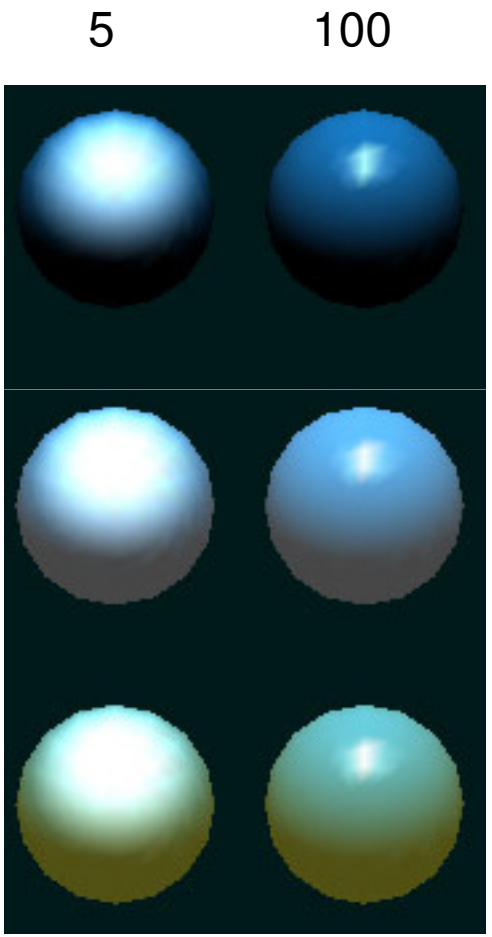


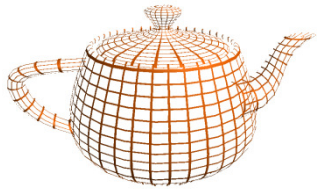
Nota: *só se consideram os valores positivos do produto interno*



Iluminação - Fundamentos

- O coeficiente de especularidade [0,128] determina a dimensão da mancha brilhante.
- Quanto maior o valor, mais pequena é a mancha.
- Materiais metálicos tendem a ter valores altos, enquanto que materiais baços definem-se com valores baixos.

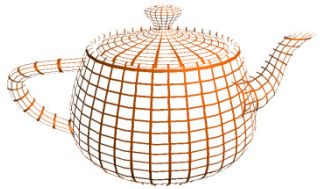




Iluminação - Fundamentos

- Cor: R G B

- $I_R = I_{aR} * K_{aR} +$
  $f_{att} * I_p * [$
  $K_{dR} * (N \,.\, L) +$
  $I_s * K_{sR} * (R.\text{Câmara})^s$
  $]$
- $I_G = \ldots$
- $I_B = \ldots$

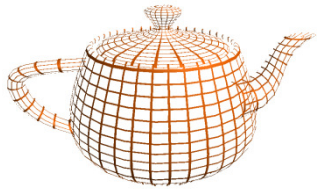


Iluminação - OpenGL

- $V_c = V_e +$
  $V_a * L_{ga} +$
  $sum(fatt * (V_a * L_a +$
  $max\{N.L, 0\} * V_d * L_d +$
  $max\{R.Câmara, 0\}^s * V_s * L_s)$

Vc = cor final do vértice
Ve = cor emissiva do vértice
Va = cor ambiente do vértice
Vd = cor difusa do vértice
Vs = cor especular do vértice
s = coeficiente de especularidade

Lga = cor global ambiente
La = cor ambiente da luz
Ld = cor diffusa da luz
Ls = cor especular da luz



Iluminação - OpenGL

- Para cada vértice é necessário definir a sua normal.

- Podemos considerar a normal como sendo um vector unitário perpendicular à superfície do polígono.

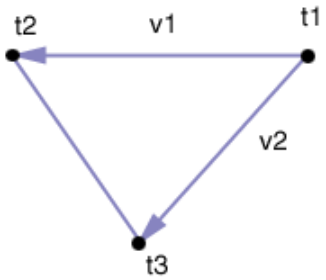
- Considerando um triângulo, podemos definir a sua normal como sendo o produto externo entre duas arestas:
  - n = v1 x v2 = [nx,ny,nz]

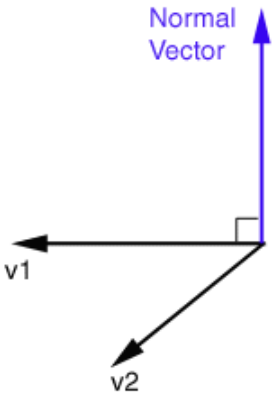
    nx = v1y * v2z - v1z * v2y

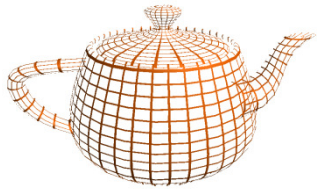
    ny = v1z * v2x - v1x * v2z

    nz = v1x * v2y - v1y * v2x

- Nota: O vector obtido através do produto externo deve ser normalizado.







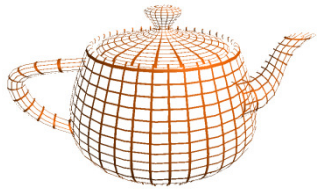
Iluminação - OpenGL

- Código OpenGL:

```
glBegin(GL_TRIANGLE);
        glNormal3f(0,1,0);
        glVertex3f(0,0,0);
        glNormal3f(0,1,0);
        glVertex3f(0,0,1);
        glNormal3f(0,1,0);
        glVertex3f(1,0,0);
glEnd();
```

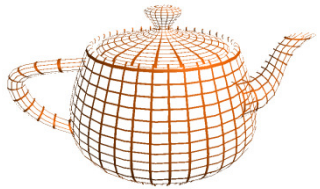
No caso de se utilizar a mesma normal para todos os vértices pode-se escrever:

```
glBegin(GL_TRIANGLE);
        glNormal3f(0,1,0);
        glVertex3f(0,0,0);

        glVertex3f(0,0,1);

        glVertex3f(1,0,0);
glEnd();
```



Iluminação

- Tópicos:
  - Fundamentos de iluminação
    - Aplicação em OpenGL
  - Modelos de Shading
  - Materiais em OpenGL
  - Iluminação em OpenGL

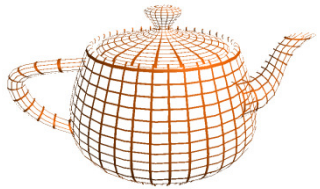


Modelos de *Shading*

- Processo para colorir um polígono (ou superfície) utilizando um determinado modelo de iluminação.

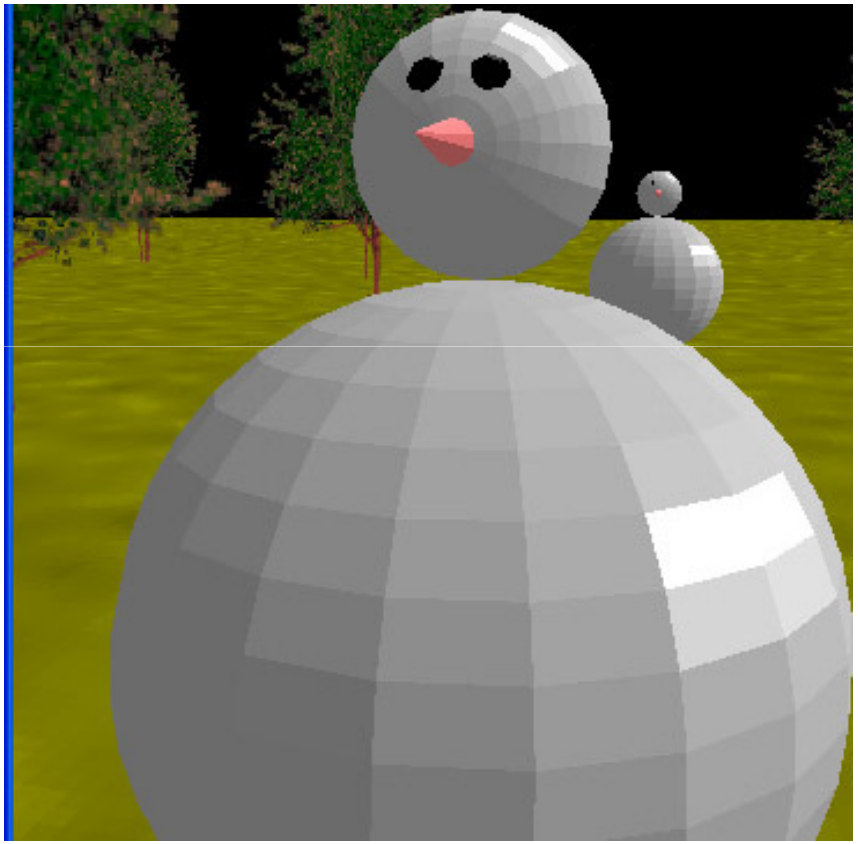
- Alguns modelos de iluminação:

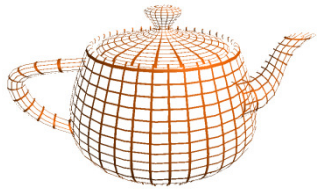
  - Flat (constante)
  - Interpolação
  - Gouraud
  - Phong (não disponível em OpenGL)



Shading - Flat

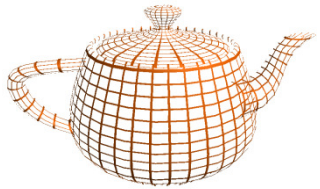
- Neste modelo o polígono tem iluminação constante em toda a sua superfície.
- Uma normal para cada polígono
- Produz um resultado facetado





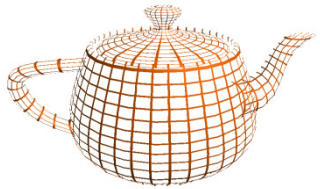
Shading - Flat

- Este modelo só faz sentido se:
  - A distância entre a fonte de luz e o polígono for infinita, de forma a que N.L é constante ao longo do polígono.
  - O utilizador encontra-se também a uma distância infinita, para que não haja variação da componente especular ao longo do polígono.
  - A modelação é uma representação fiel da superfície a modelar, i.e. não é uma aproximação.



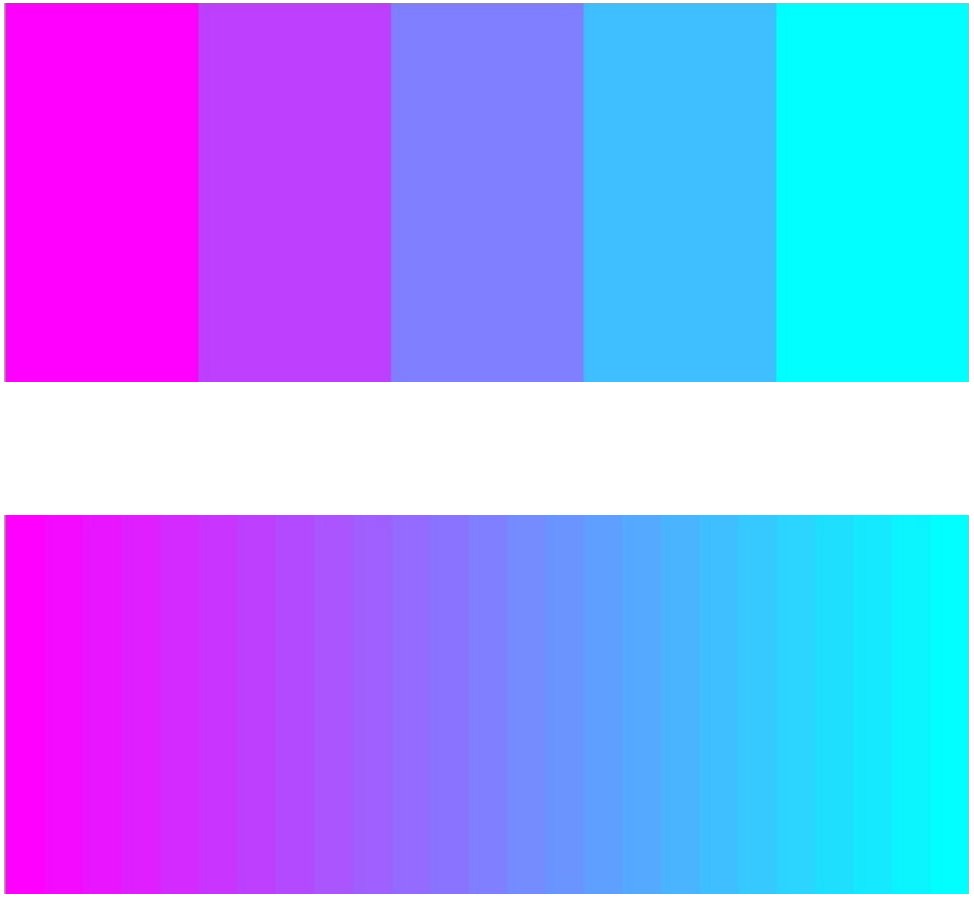
Shading - Flat

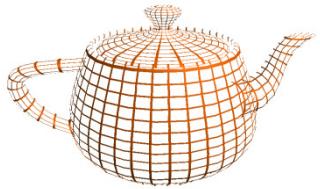
- Problema: Aspecto facetado!
- Solução: Definir uma malha poligonal mais fina.
- Esta solução tem desvantagens óbvias:
  - Primeiro implica um número mais elevado de polígonos o que pode diminuir o desempenho.
  - Em segundo lugar, o aspecto facetado é de facto intensificado devido ao efeito das bandas de Mach.



Shading Flat

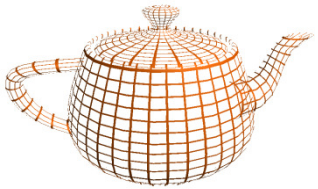
- Mach Band – Fenómeno provocado pela disparidade entre a diferença real de intensidade e a intensidade percepcionada.





Shading - Interpolação

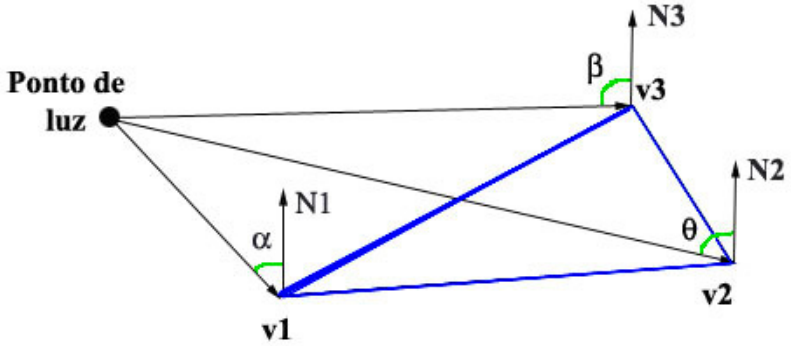
- Neste modelo, proposto por Gouraud, determina-se a intensidade da luz para cada vértice utilizando a normal respectiva.
- A intensidade dos restantes pontos do polígono é calculada por interpolação
- Desta forma elimina-se a primeira restrição do modelo FLAT: a distância do polígono à luz não necessita de ser infinita.

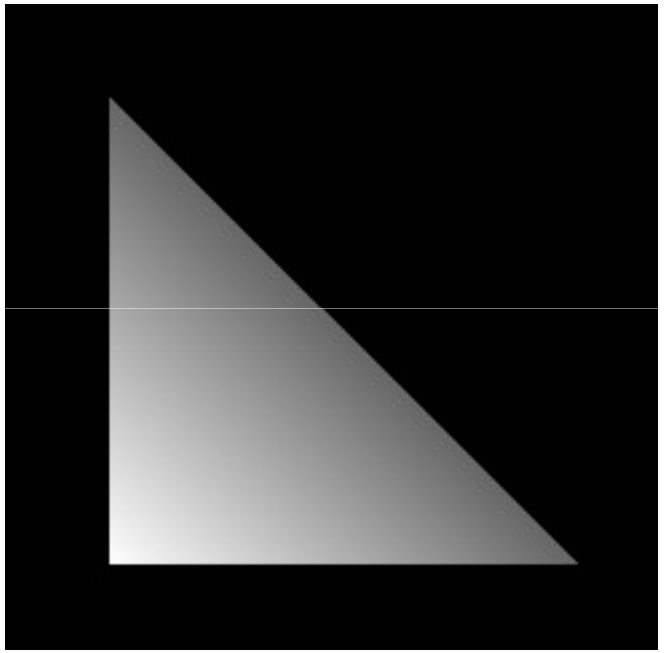


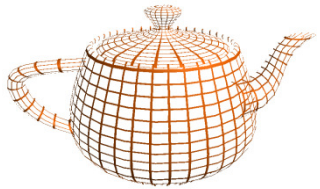
Shading - Interpolação

A intensidade varia ao longo do polígono.

A intensidade do vértice é proporcional ao ângulo entra a sua normal e a direcção da luz.

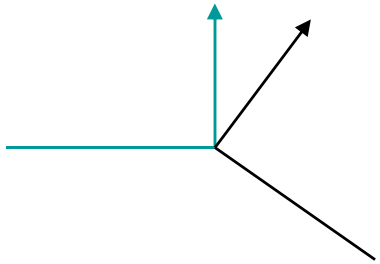






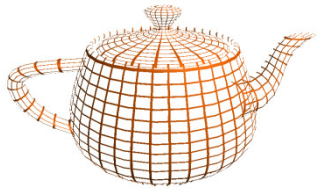
Shading - Interpolação

- Problema: Superfície continua facetada.



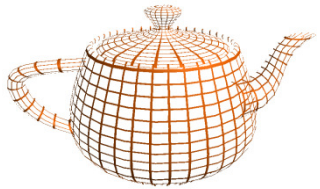
- As normais nos pontos de descontinuidade são diferentes!

- Polígonos com orientações diferentes têm intensidades diferentes nas suas arestas.



Shading - Gouraud

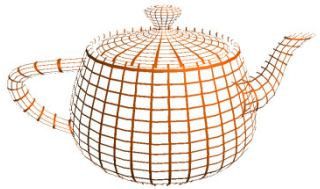
- Muitos dos objectos a modelar são constituídos por superfícies curvas, e a modelação poligonal é apenas uma aproximação.
- Objectivo: Aproximar uma superfície curva por uma malha poligonal
- Mas, se cada polígono for iluminado individualmente ...
- ... continua-se a ter uma aparência facetada, e por consequência torna-se fácil distinguir um polígono dos seus vizinhos, cuja orientação é diferente.



Shading - Gouraud

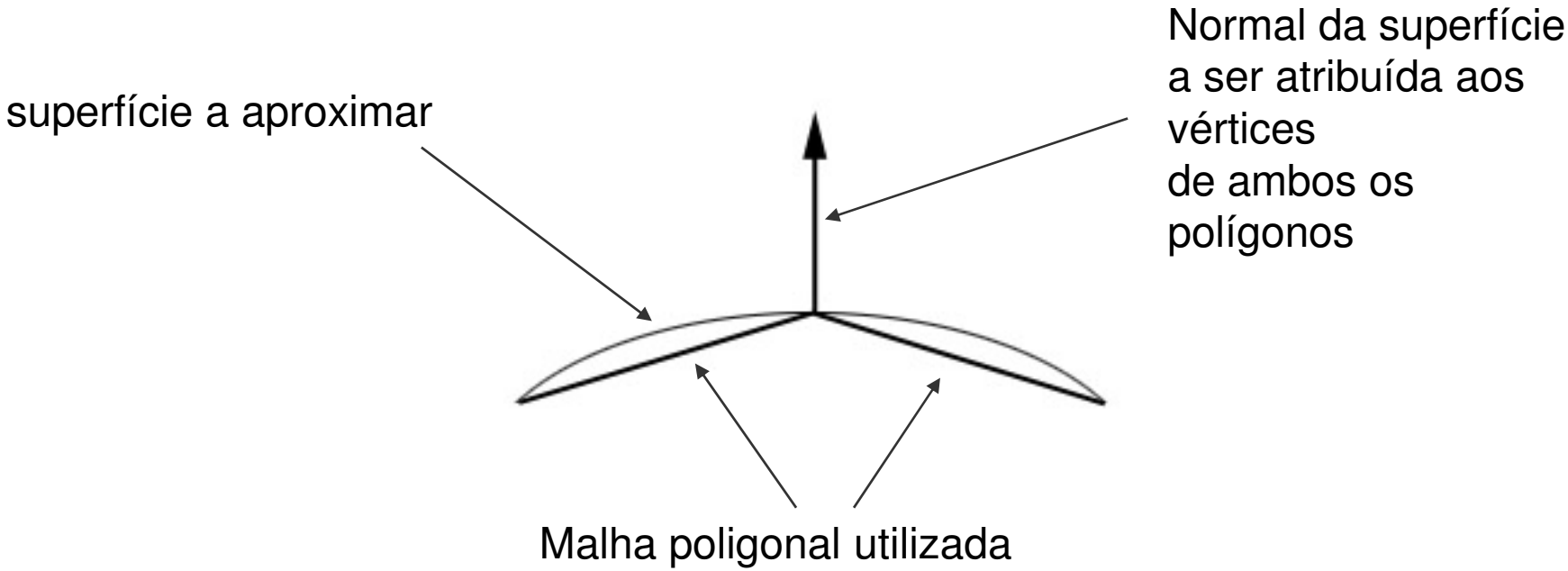
Para resolver este problema, Gouraud posteriormente sugeriu que ...

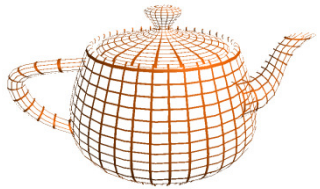
- ... cada vértice tivesse uma normal que representasse, não a orientação do polígono, ...
- ... mas sim a normal da superfície que a malha poligonal pretende aproximar.



Shading - Gouraud

- Isto implica que as normais da superfície a aproximar sejam conhecidas para cada vértice.

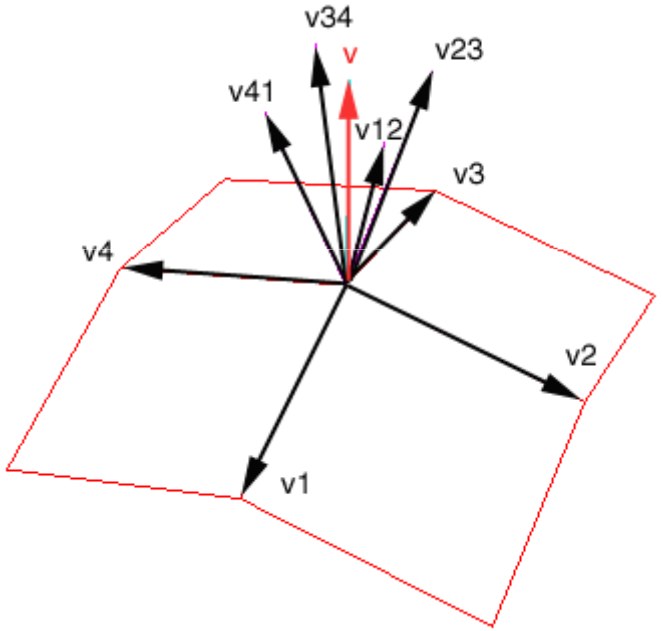


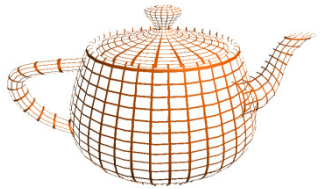


Shading - Gouraud

- No caso das normais da superfície não serem conhecidas, e não for possível o seu cálculo, ...

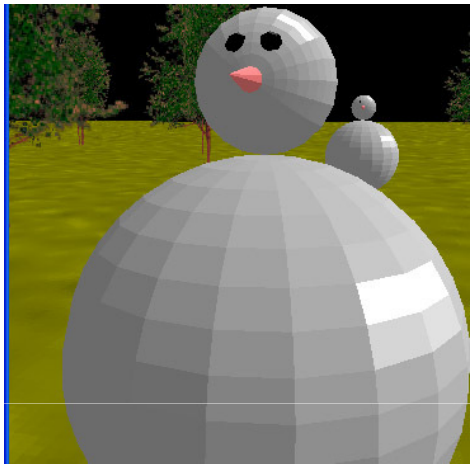
- ... é possível obter uma aproximação através da média (normalizada) das normais de cada polígono individual que partilhe o vértice.



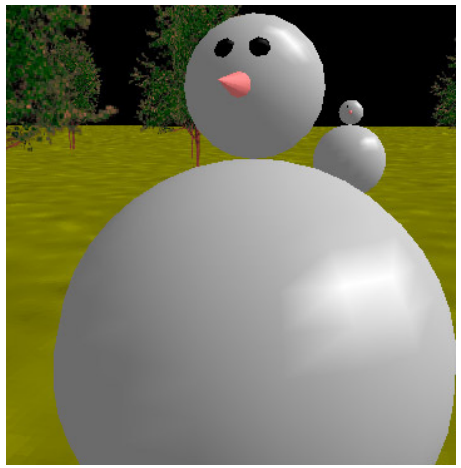


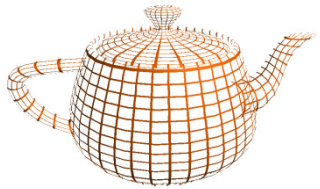
Shading - Gouraud

A imagem de cima apresenta o modelo com flat shading



A de baixo utiliza Gouraud shading.

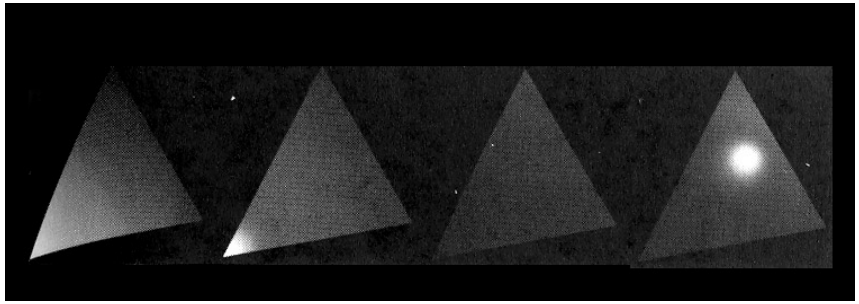


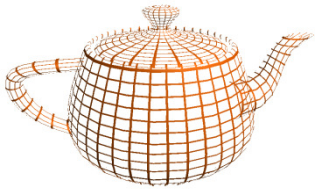


Shading - Gouraud

- O modelo de Gouraud não elimina completamente o problema das bandas de Mach, embora as reduza consideravelmente.

- As manchas especulares não são reproduzidas fielmente



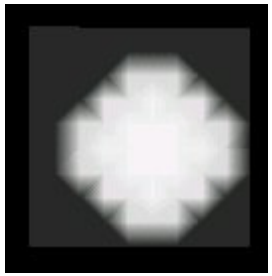


Shading - Gouraud

- Problema: Dependência da iluminação nos vértices do polígono.
- Um polígono parcialmente iluminado, em que nenhum dos vértices é iluminado é representado como se a totalidade do polígono não fosse iluminado.
- Solução: Malha mais fina

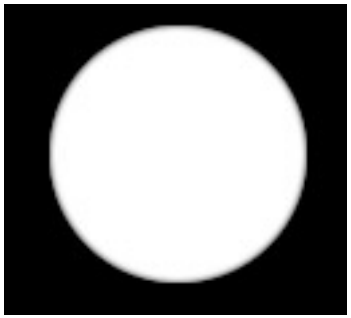
luz circular inclui só um canto do polígono

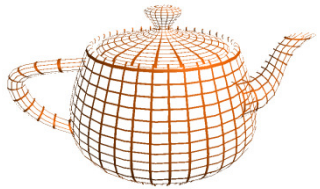
luz circular inteiramente dentro do polígono



malha mais fina

situação desejada



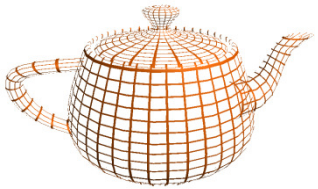


Shading - Phong

- Phong propõe:

  Interpolar Normais em vez de intensidades

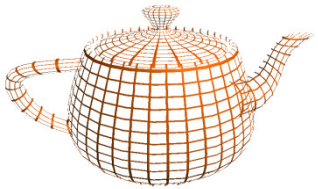
- Problema: Tempo Real? (já ultrapassado)



Shading

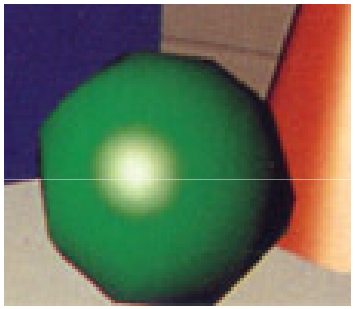
- Gouraud

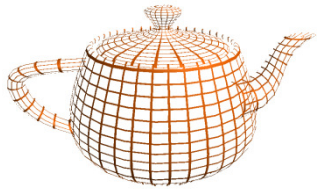
- Phong



Shading

- Problema: Silhueta
- Independente da qualidade do modelo de shading



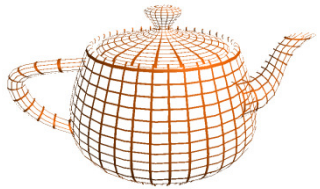


Shading

- Problema: Nos modelos apresentados os objectos não interagem em termos de iluminação. São modelos locais.

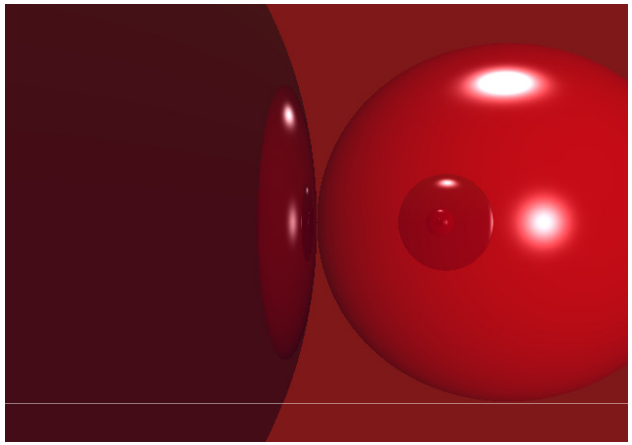
- Por exemplo: Objectos não causam sombras noutros objectos

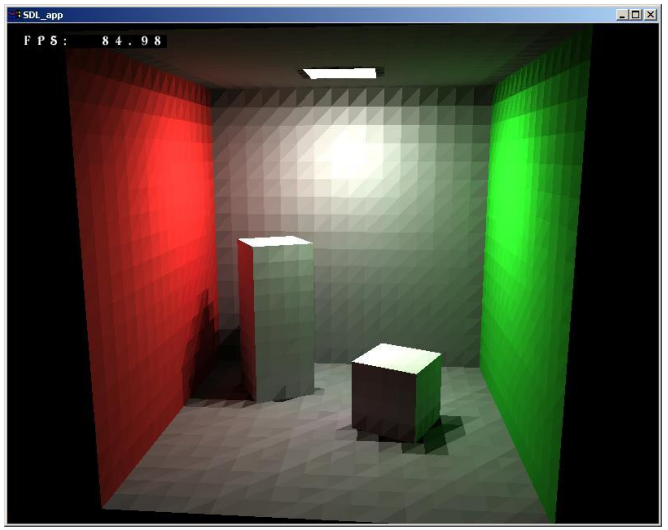
A luz vem da esquerda

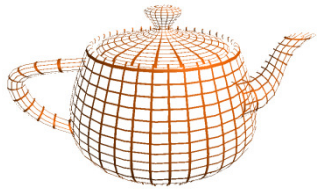


Shading

- Modelos mais complexos, que contemplem iluminação global, não são utilizáveis em tempo real:
  - Ray tracing
    - cálculo de interacções especulares
    - cálculo de sombras
  - Radiosidade
    - cálculo de interacções difusas
  - Photon Mapping

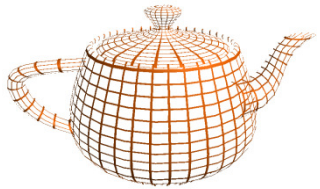






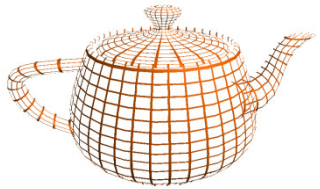
Iluminação

- Tópicos:
  - Fundamentos de iluminação
    - Aplicação em OpenGL
  - Modelos de Shading
  - Materiais em OpenGL
  - Iluminação em OpenGL



OpenGL - Materiais

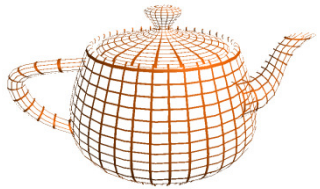
- Componentes da cor:
  - Difusa
  - Especular
  - Ambiente
  - Emissiva



OpenGL - Materiais

- Se para um objecto não for definido um material a sua representação é a da direita

- Esfera azul iluminada por luz vermelha



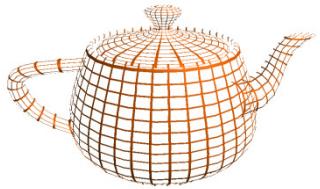
OpenGL - Materiais

- Atribuir materiais:
  - glMaterialfv(GL_FRONT, componente, array);
  - glMaterialf(GL_FRONT,GL_SHININESS,valor);

    0..128

  Componente:

  GL_DIFFUSE
  GL_AMBIENT
  GL_SPECULAR
  GL_EMISSION
  GL_AMBIENT_AND_DIFFUSE

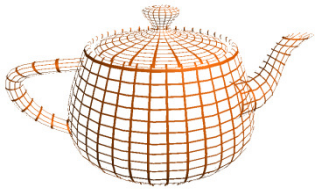


OpenGL - Materiais

```
glMaterialfv(GL_FRONT,GL_DIFFUSE, vermelho);
desenha1();
glMaterialfv(GL_FRONT,GL_DIFFUSE, azul);
desenha2();
```

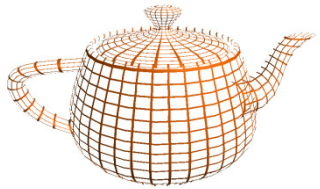
- OU

```
glColorMaterial(GL_FRONT, GL_DIFFUSE);
glEnable(GL_COLOR_MATERIAL);
glColor3fv(vermelho);
desenha1();
glColor3fv(azul);
desenha2();
```



Iluminação

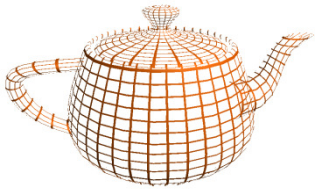
- Tópicos:
  - Fundamentos de iluminação
    - Aplicação em OpenGL
  - Modelos de Shading
  - Materiais em OpenGL
  - Iluminação em OpenGL



OpenGL - Iluminação

- Definir propriedades da luz

- glLight{if}(GL_LIGHTi, param, valor1,valor2, ...);
- glLight{if}v(GL_LIGHTi, param, array_valores)

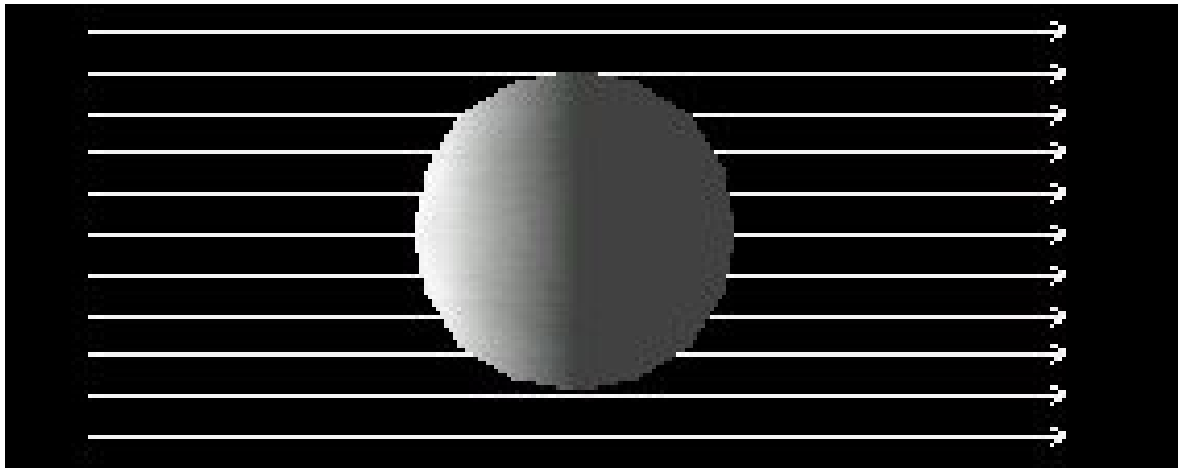


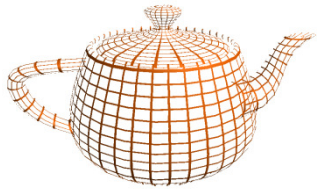
OpenGL - Iluminação

Direccional

Atributos: direcção, cor

GL_POSITION





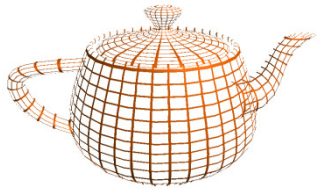
OpenGL - Iluminação

- Definir uma luz direccional

```
GLfloat amb[3] = {0.2, 0.2, 0.2};
GLfloat diff[3] = {1.0, 1.0, 1.0};
GLfloat pos[4] = {0.0, 0.0 ,1.0, 0.0};

glLightfv(GL_LIGHT0, GL_POSITION, pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diff);
```

0.0 indica que a luz é direccional

A posição indica a direcção da luz



OpenGL - Iluminação

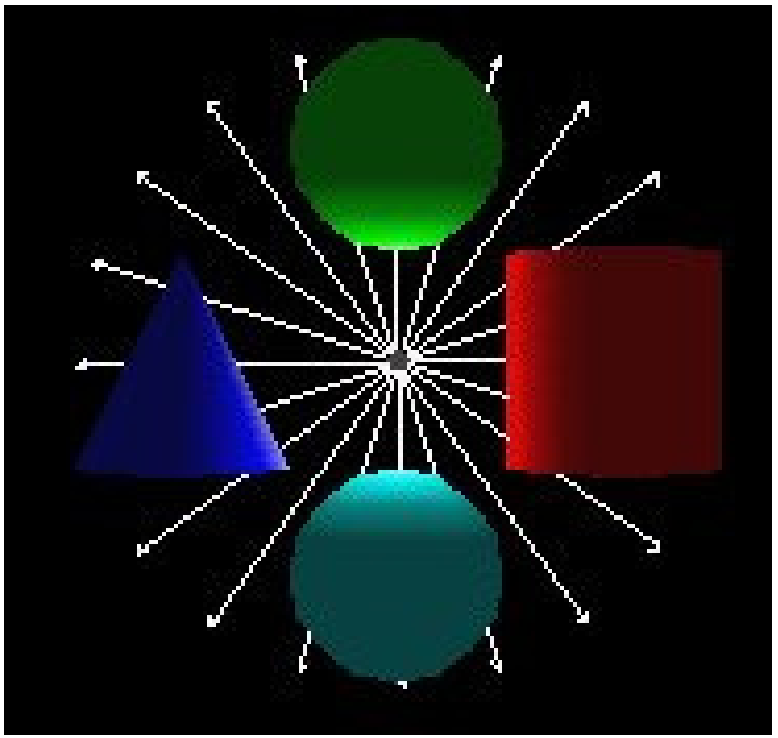
- Ponto de Luz

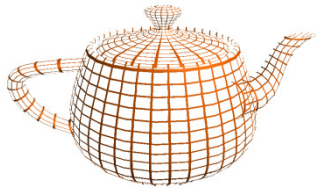
  Atributos: posição,

  atenuação, cor

GL_POSITION

GL_..._ATTENUATION



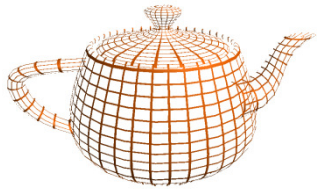


OpenGL - Iluminação

- Definir um ponto de luz

```
GLfloat amb[3] = {0.2, 0.2, 0.2};
GLfloat diff[3] = {1.0, 1.0, 1.0};
GLfloat pos[4] = {0.0, 0.0 ,10.0, 1.0};

glLightfv(GL_LIGHT0, GL_POSITION, pos);
glLightfv(GL_LIGHT0, GL_AMBIENT, amb);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diff);
```

1.0 indica que se trata de um ponto de luz

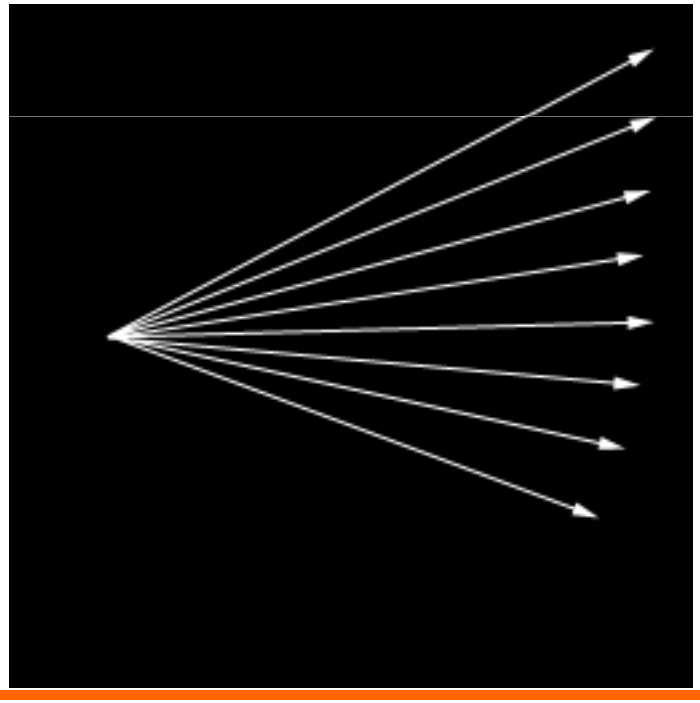


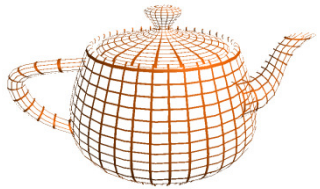
OpenGL - Iluminação

Foco de Luz (Spotlight)

Atributos: posição, ângulo, atenuação, direcção, cor

GL_POSITION
GL_SPOT_DIRECTION
GL_SPOT_CUTOFF
GL_..._ATTENUATION

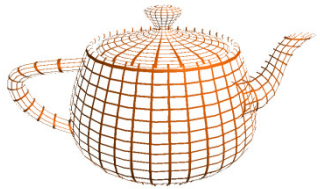




OpenGL - Iluminação

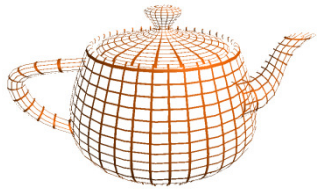
- Definir um foco de luz

```
GLfloat diff[3] = {1.0, 1.0, 1.0};
GLfloat pos[4] = {0.0, 0.0 ,10.0, 1.0};
GLfloat spotDir[3] = {0.0, 0.0, -1.0};

glLightfv(GL_LIGHT0, GL_POSITION, pos);
glLightfv(GL_LIGHT0, GL_DIFFUSE, diff);
glLightfv(GL_LIGHT0, GL_SPOT_DIRECTION, spotDir);
glLightf(GL_LIGHT0, GL_SPOT_CUTOFF, 45.0); // [0,90] ou 180
glLightf(GL_LIGHT0,GL_SPOT_EXPONENT, 0.0); // [0,128]
```



OpenGL - Iluminação

- Ligar, desligar luzes individuais (por omissão estão desligadas)
  - glEnable(GL_LIGHTi);
  - glDisable(GL_LIGHTi);
- Ligar,desligar o quadro
  - glEnable(GL_LIGHTING);
  - glDisable(GL_LIGHTING);



OpenGL - Iluminação

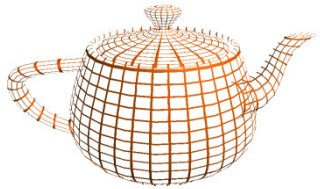
Algumas Propriedades Globais

- Luz Ambiente Global

```
GLfloat mod_amb[3] = {0.2, 0.2, 0.2, 1.0};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT,
               mod_amb);
```

- Cálculo da reflexão especular

```
glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER,
              GL_FALSE);
```



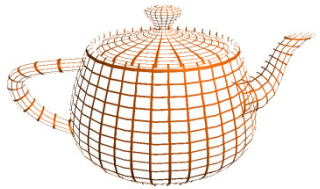
OpenGL - Iluminação

Algumas Propriedades Globais

- Cálculo separado da cor especular

```
glLightModeli(GL_LIGHT_MODEL_COLOR_CONTROL,
                    GL_SINGLE_COLOR);
glLightModeli(GL_LIGHT_MODEL_COLOR_CONTROL,
                    GL_SEPARATE_SPECULAR_COLOR);
```

O cálculo da cor pode ser realizado em 1 ou dois passos. Neste último caso a cor especular não é influenciada pela aplicação de texturas.



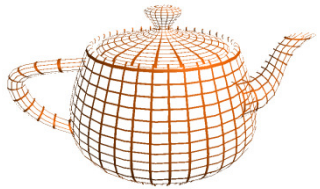
OpenGL - Iluminação

Algumas Propriedades Globais

- Cálculo da iluminação em ambos os lados

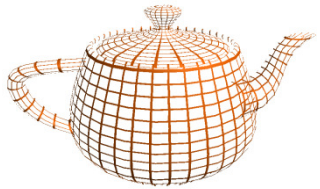
```
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE,
                                GL_TRUE);
```

Quando se especifica TWO_SIDE, estamos a pedir ao OpenGL que inverta as normais para o lado de trás do polígono



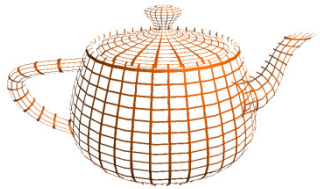
OpenGL - Iluminação

Demo sobre luzes e materiais



OpenGL - Iluminação

Demo sobre posicionamento da luz



Referências

- **Computer Graphics – Principles and Practice**, Foley, van Dam, Feiner and Hughes
- **OpenGL Reference Manual**, OpenGL ARB
- **OpenGL Programming Guide**, OpenGL ARB