Desenvolvimento de aplicações tridimensionais com OpenGL

Pontifícia Universidade Católica de Minas Gerais, 2004

Alessandro Ribeiro

spdoido@yahoo.com.br

Bruno Evangelista

bpevangelista@yahoo.com.br

Orientador: Marcelo Nery

msnery@terra.com.br

Iluminação

Sumário

- **Visão Geral**
- **Conceitos básicos de iluminação**
- **Iluminação com OpenGL**
- **Fontes de luz**
- **Criando fontes de luz**
- **Materiais**

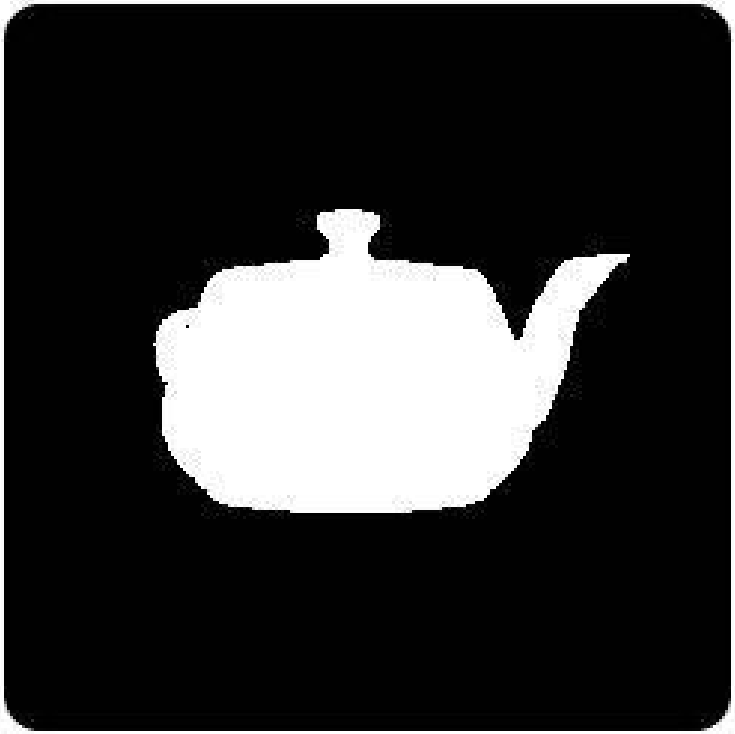
Visão Geral
Motivo de utilização

- **Acrescentar um pouco de realismo a cena**
  - **Realizar simulação gráfica de ambientes**
  - **Visualizar diferenças entre propriedades reflexivas de objetos**

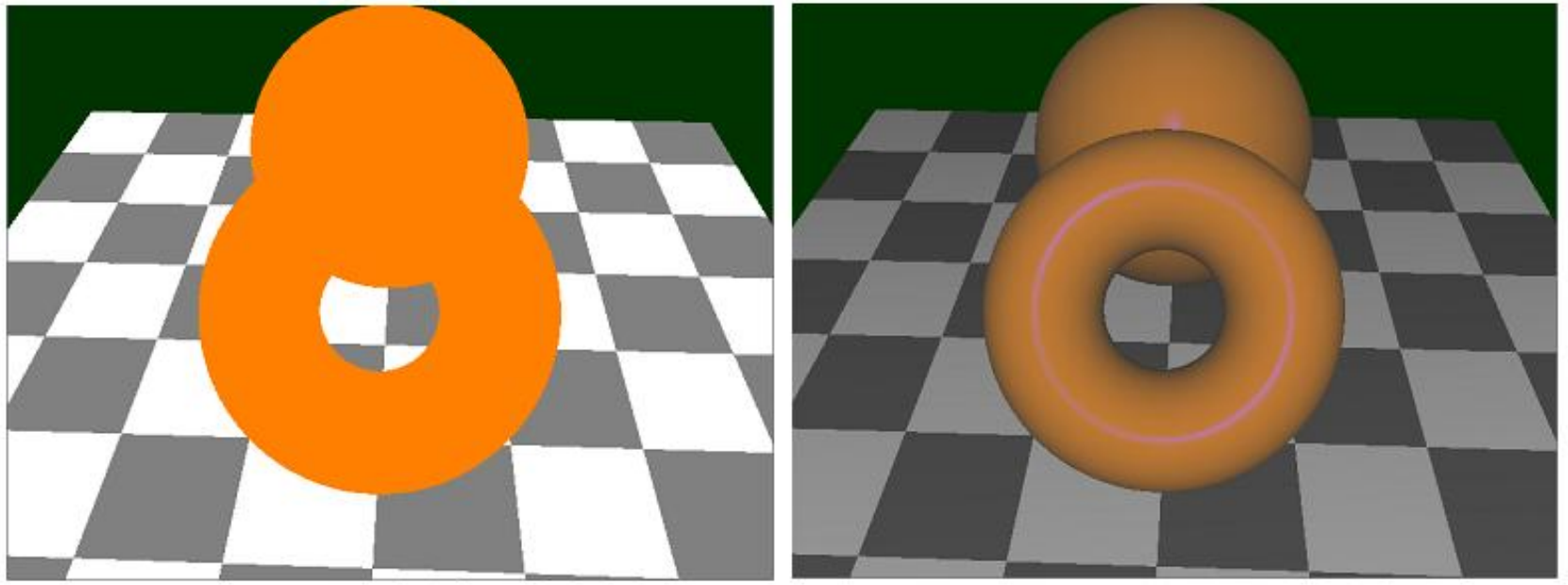
Visão Geral
Conseqüência

- **Normalmente ao se utilizar iluminação tem-se um overhead considerável, pois estes cálculos são feitos para todos os vértices do modelo que é desenhado**

Visão Geral
Exemplo 1



Visão Geral
Exemplo 2



Visão Geral
A luz como fenômeno físico

- **Natureza complexa de ser descrita, pois existem fatores que influenciam o comportamento da luz**
  - **Fenômenos atmosféricos (ex.: gases, pressão,...)**
  - **Fenômenos de mudança de meio (ex.: objetos/meios translúcidos)**
  - **Fenõmenos relacionado ao comportamento de onda (ex.: difração)**

Visão Geral
A luz como fenômeno físico

- **Modelos que tentam descrever a luz:**
  - **Onda - trata a luz fazendo analogia a uma onda na água**
  - **Partícula - trata a luz como uma partícula, onde esta possui determinada quantidade de energia**

Visão Geral
A luz como fenômeno físico

- **Nenhum destes modelos cobrem todas as características da luz**

Conceitos básicos de iluminação
O modelo de partícula

- **A luz viaja em linha reta**
- **A luz transporta energia**
  - **No sistema RGB(Red Green Blue) cada componente pode ser referente à intensidade da luz nesta faixa de cor**

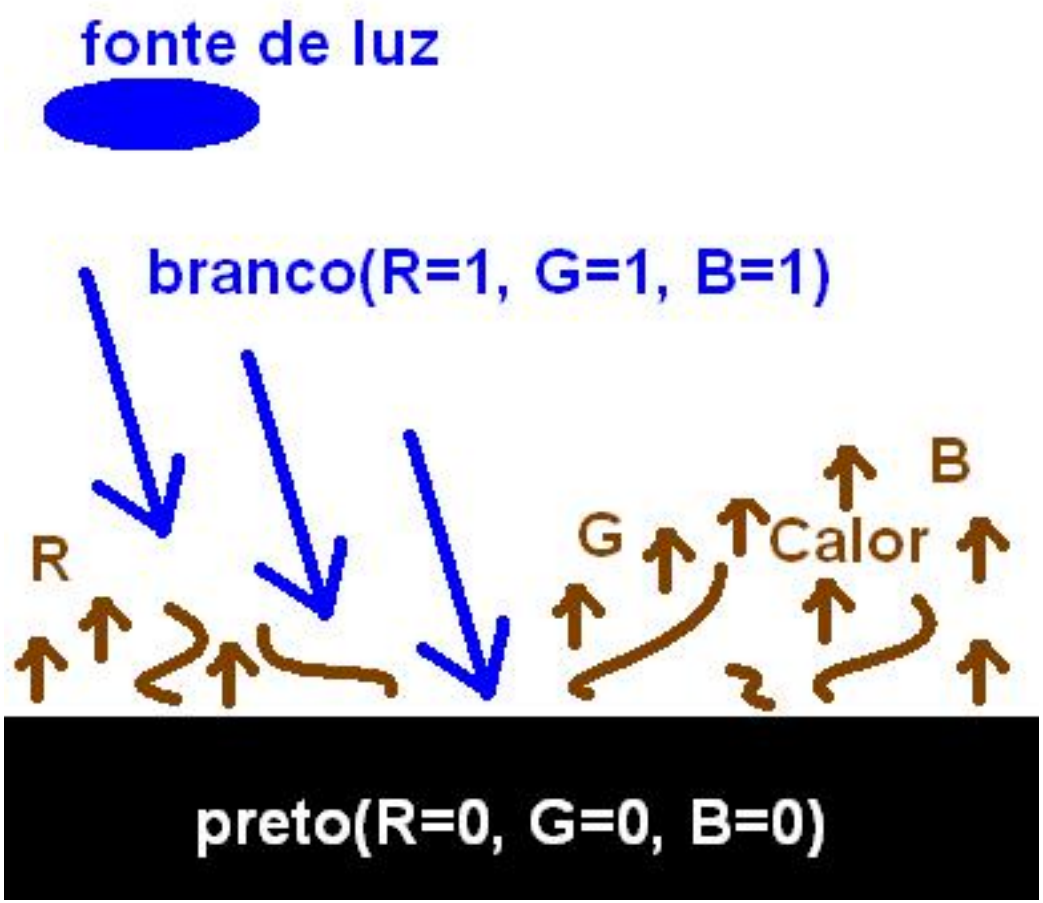
Conceitos básicos de iluminação
O modelo de partícula

- **Esta energia interage com a superfície de objetos**
- **A cor que enxergamos de um determinado objeto se dá em conseqüência destas interações**

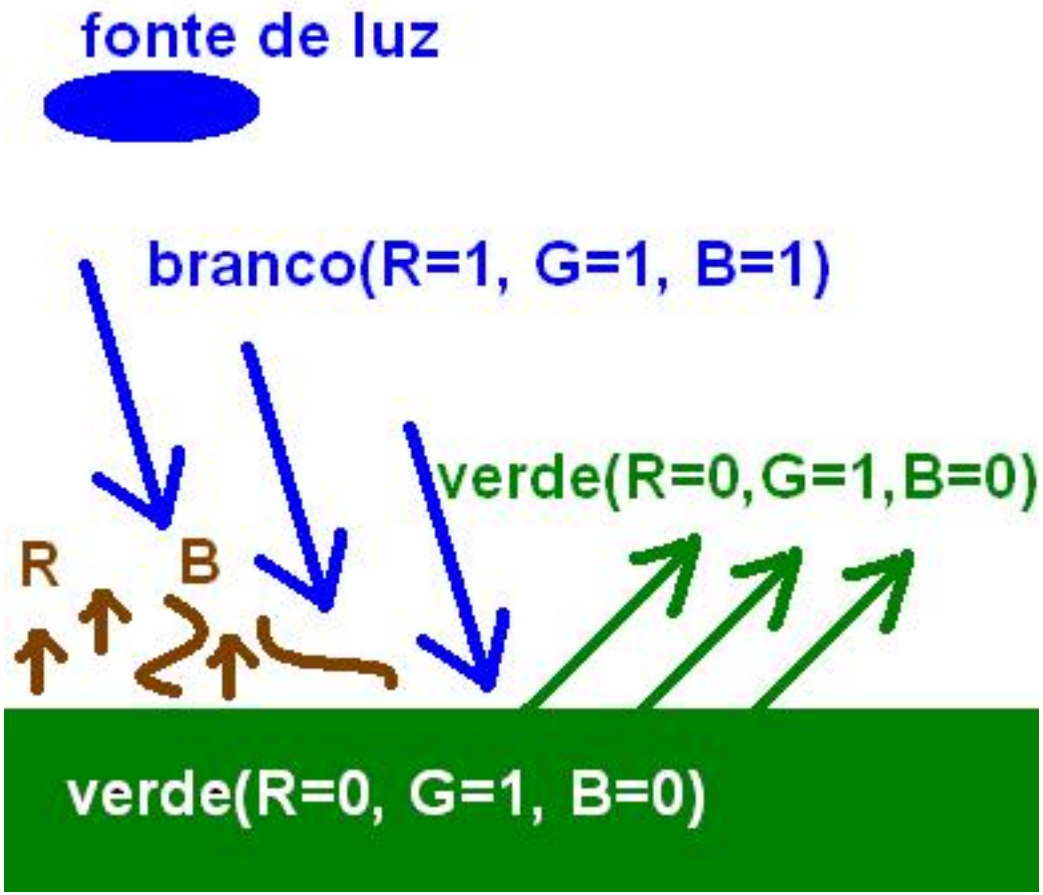
Conceitos básicos de iluminação
O modelo de partícula

- **A interação da luz com objetos pode ocorrer de duas formas:**
  - **O objeto absorver a energia e a liberá-la em forma de calor**
  - **O objeto refletir esta energia**

Conceitos básicos de iluminação
O modelo de partícula



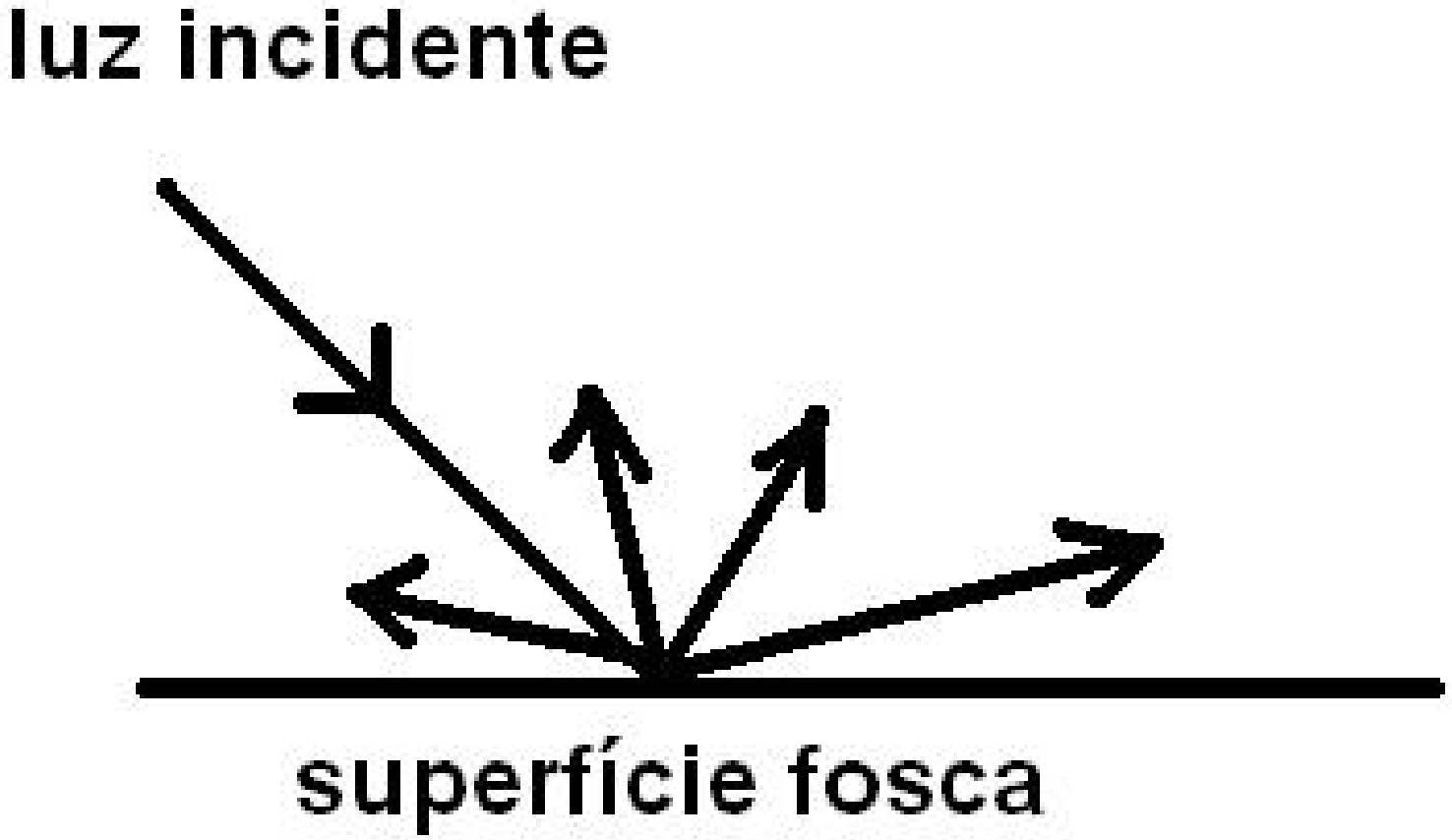
Conceitos básicos de iluminação
O modelo de partícula



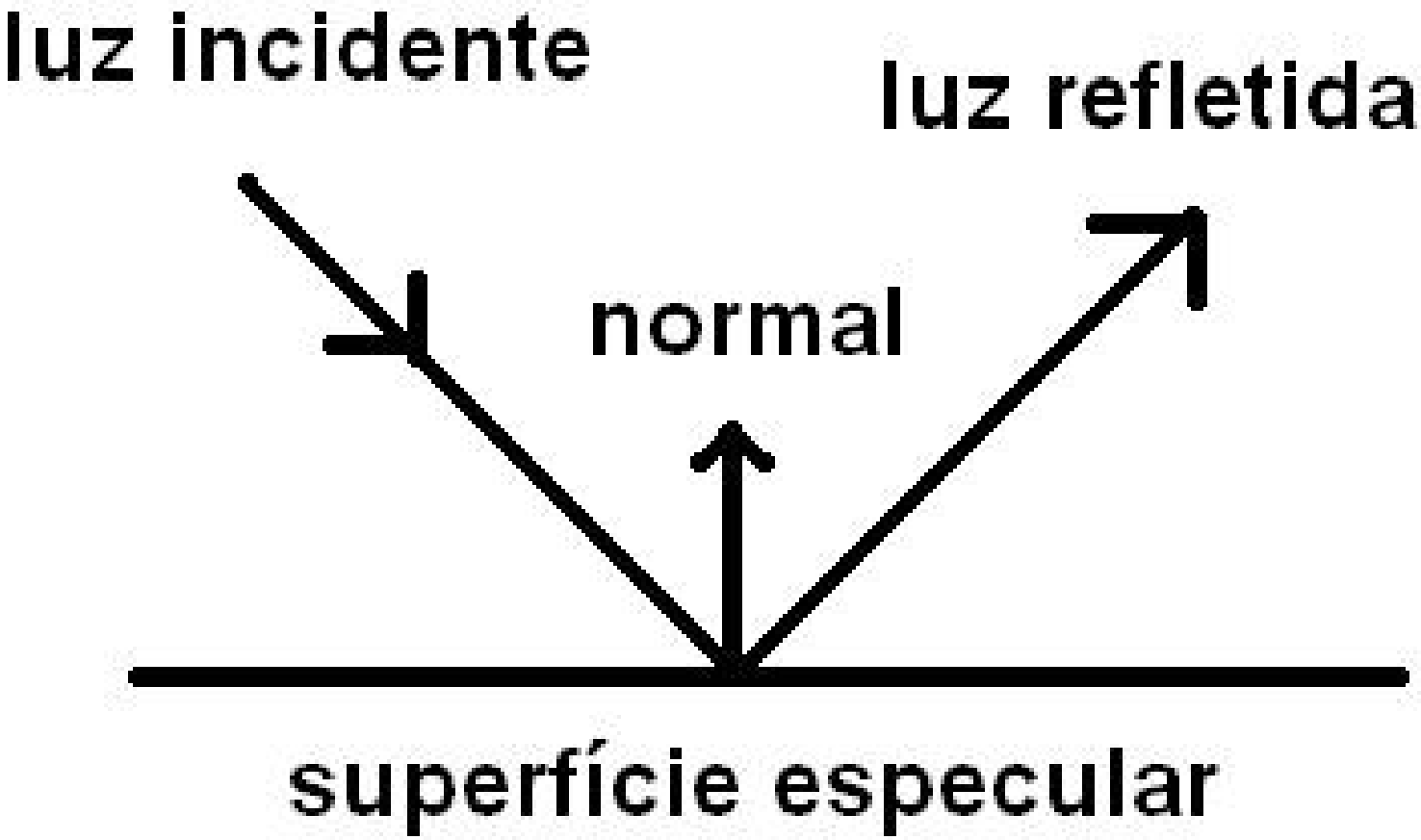
Conceitos básicos de iluminação
O modelo de partícula

- **Existen 4 tipos de comportamentos de reflexão/transmissão da luz:**
  - **reflexão difusa**
  - **reflexão especular**
  - **Transmissão especular**
  - **Reflexão total interna**

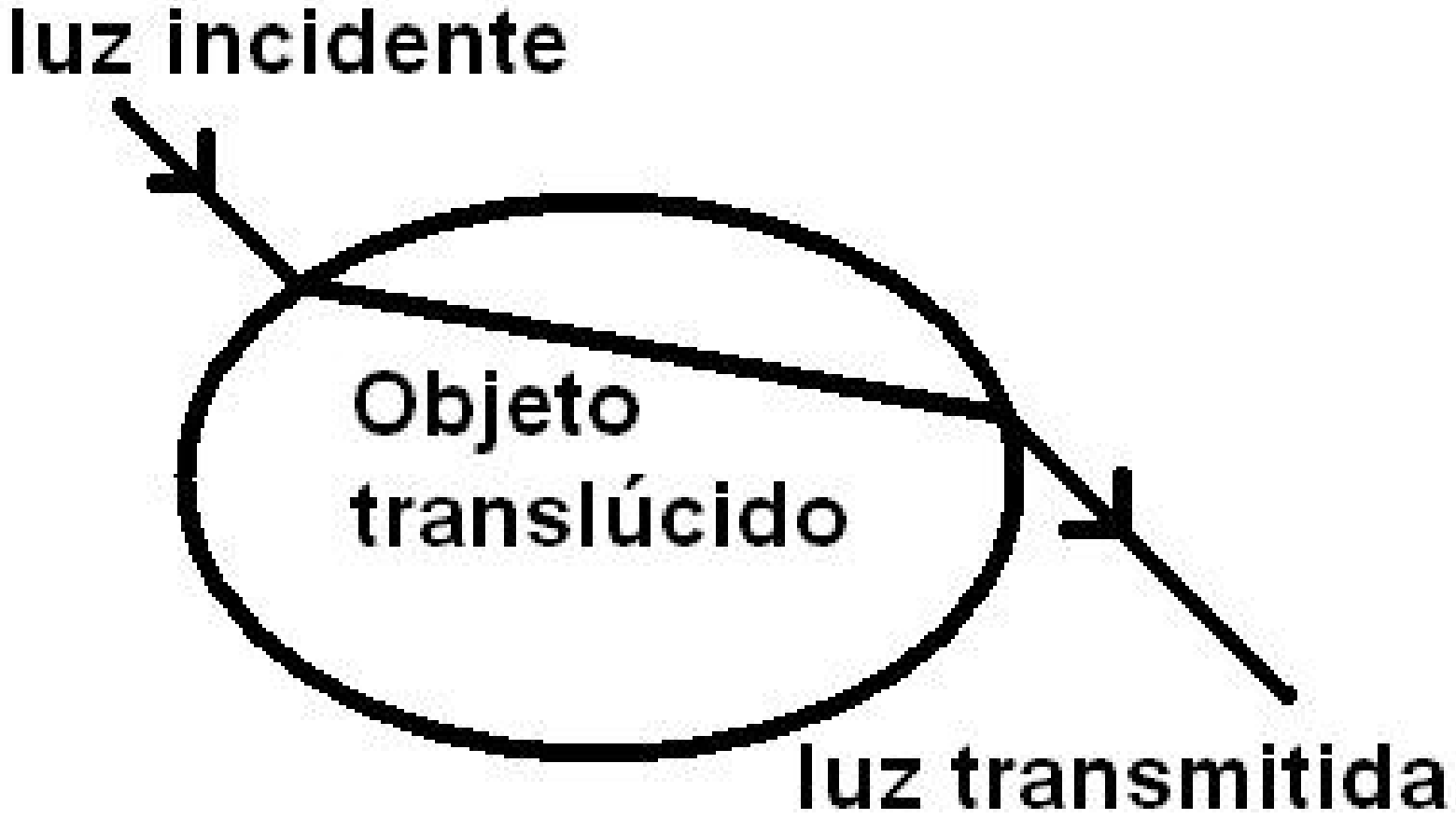
Conceitos básicos de iluminação
O modelo de partícula



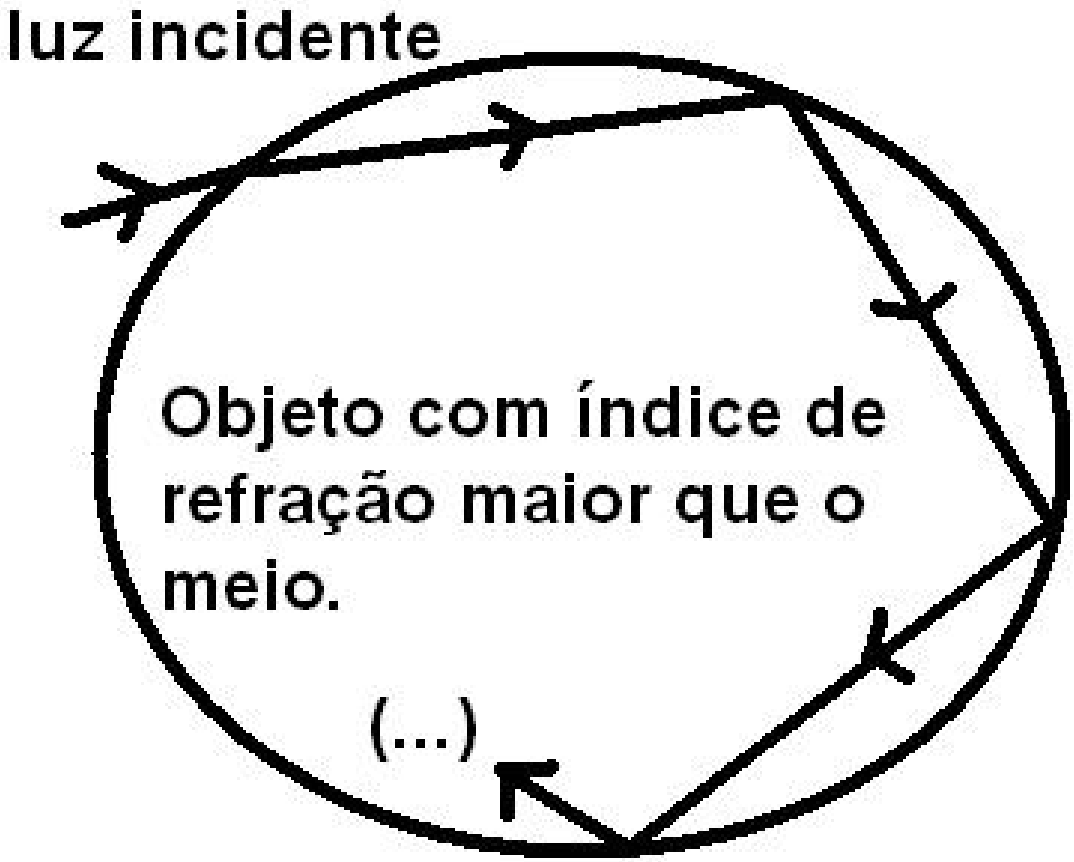
Conceitos básicos de iluminação
O modelo de partícula



Conceitos básicos de iluminação
O modelo de partícula



Conceitos básicos de iluminação
O modelo de partícula



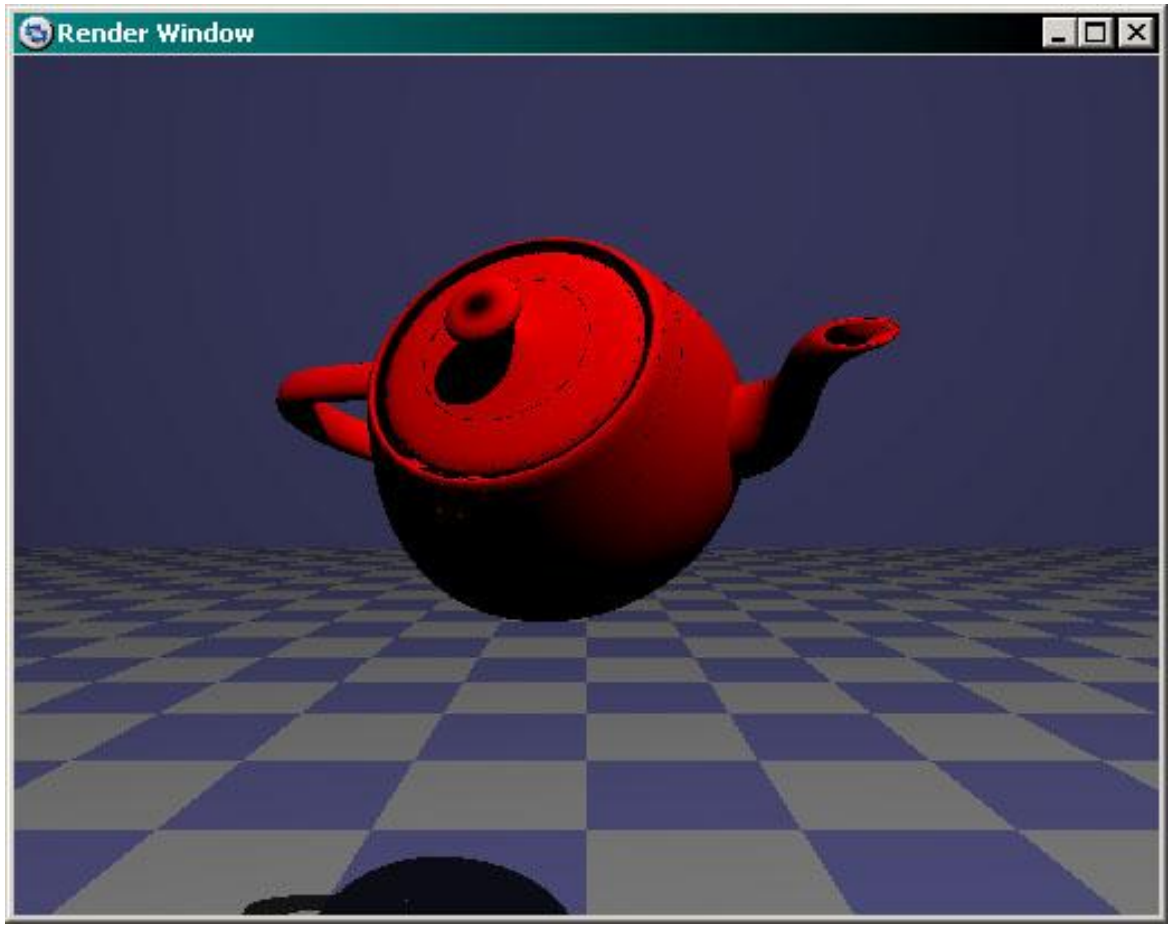
Conceitos básicos de iluminação
O modelo de Phong

- Baseado no modelo de particula
- Considera que a luz em qualquer ponto é composta por 3 componentes:
  - difuso
  - especular
  - ambiente
- Estes componentes agem de forma aditiva

Conceitos básicos de iluminação
O modelo de Phong

- **O coeficiente difuso age sobre um objeto dando intensidade à sua cor**

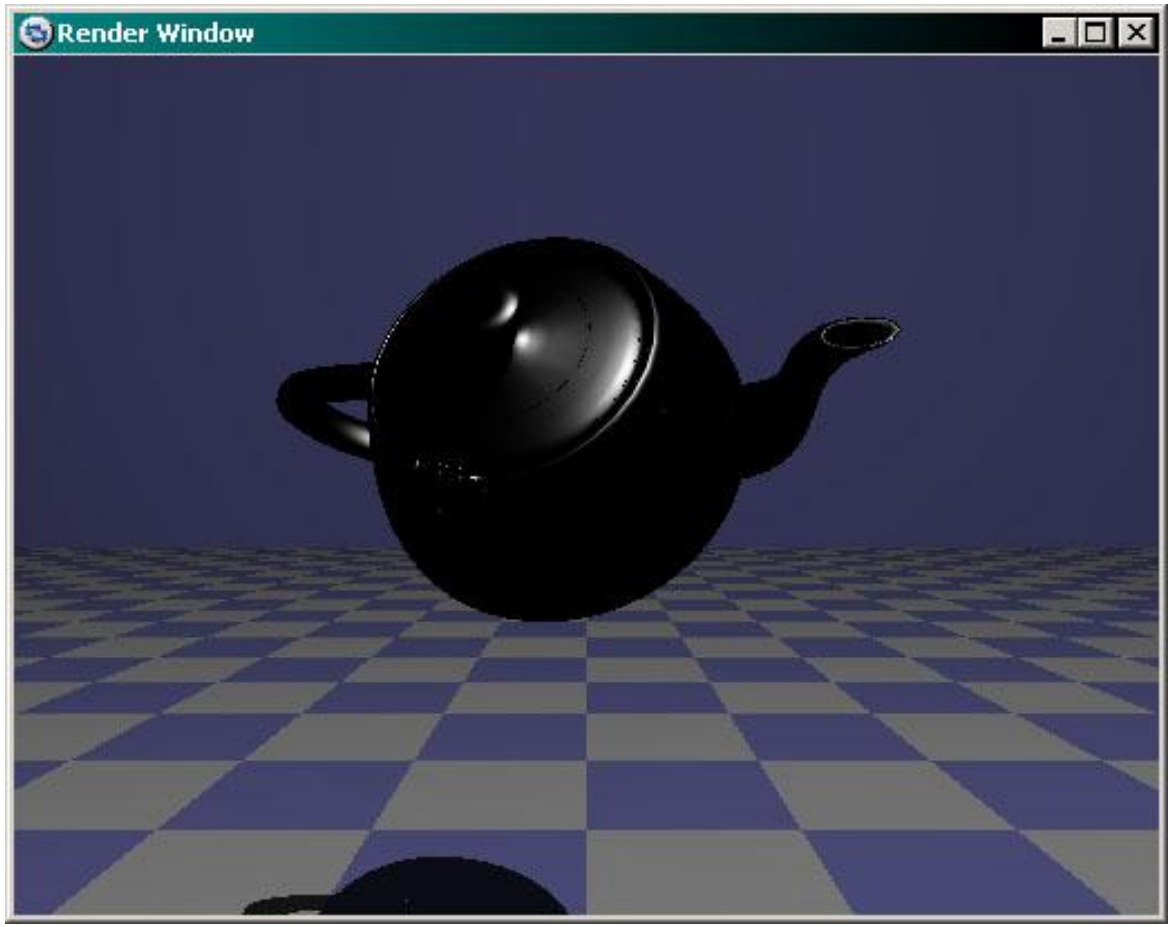
Conceitos básicos de iluminação
O modelo de Phong



Conceitos básicos de iluminação
O modelo de Phong

- O coeficiente especular é referente à reflexao da propria luz no objeto

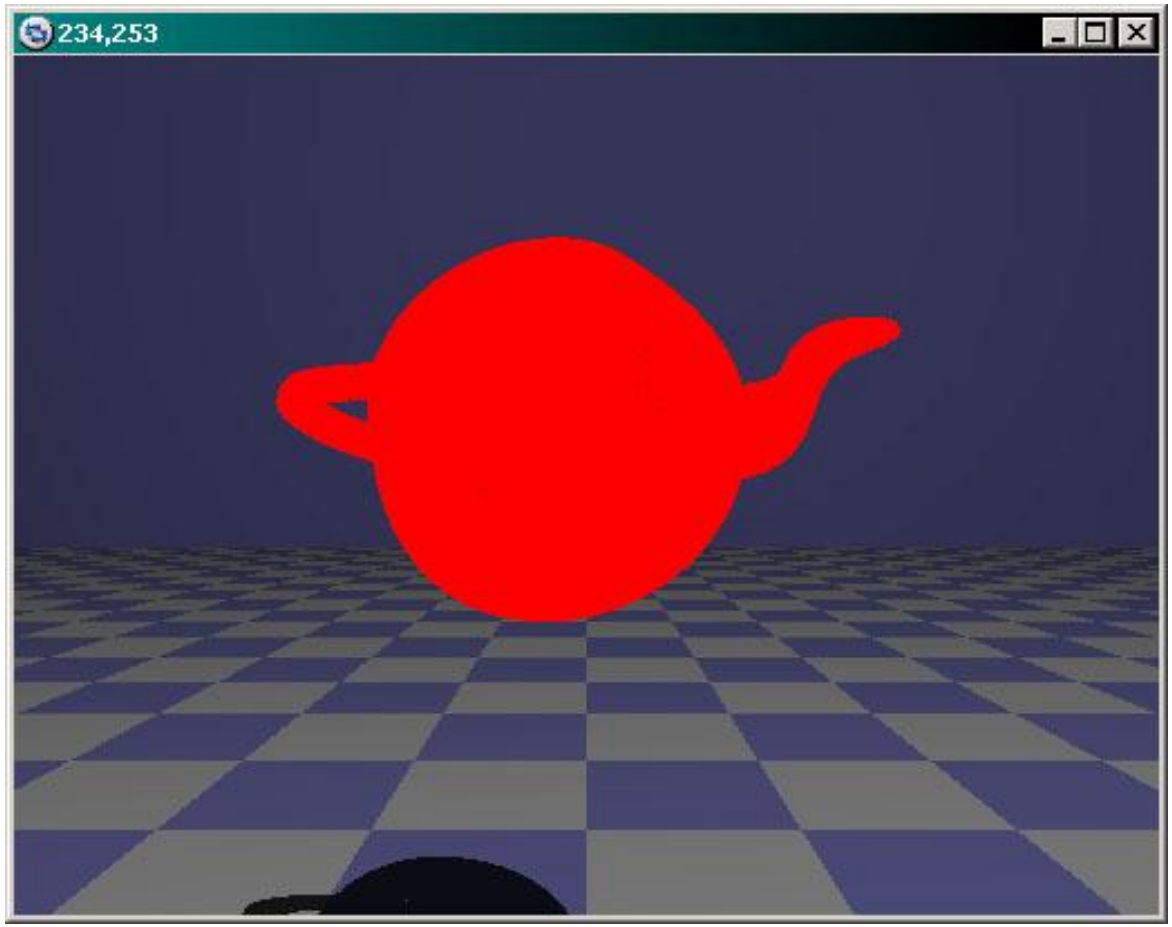
Conceitos básicos de iluminação
O modelo de Phong



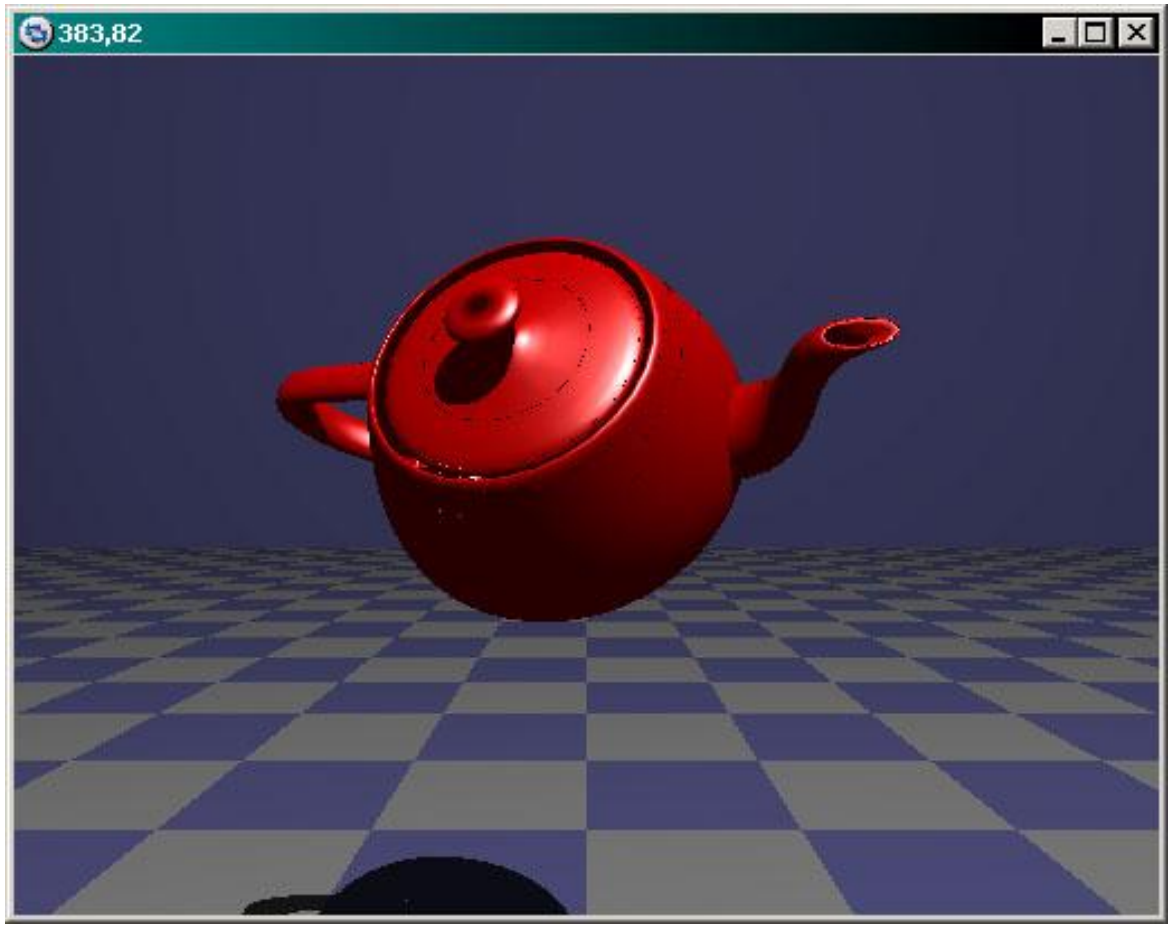
Conceitos básicos de iluminação
O modelo de Phong

- **O coeficiente ambiente equivale às contribuições vindas das diversas reflexoes dos objetos da cena**

Conceitos básicos de iluminação
O modelo de Phong



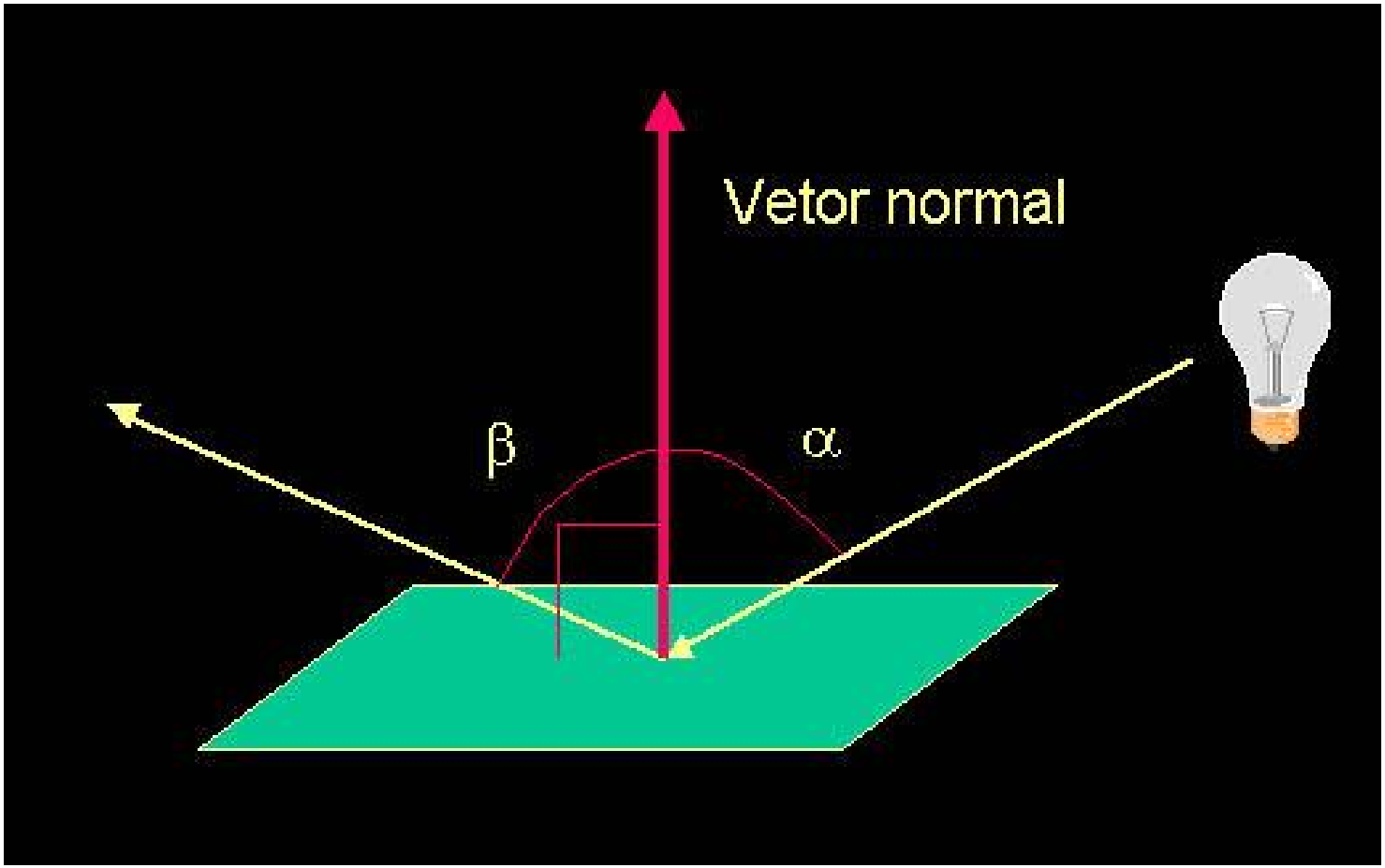
Conceitos básicos de iluminação
O modelo de Phong



Conceitos básicos de iluminação
O modelo de Phong

- **Toda a base para o cálculo da iluminação neste modelo necessita dos valores dos ângulos de incidência e reflexão em relação à superfície de um objeto**

Conceitos básicos de iluminação
O modelo de Phong



Iluminação com OpenGL

- O modelo de iluminação é baseado no modelo de Phong (com pequenas diferênças)
- A luz no OpenGL é quebrada em 3 componentes: Vermelho, Verde e Azul
- Cada componente deve estar entre os limites abaixo (0.0 a 1.0)
  - 0.0 - intensidade de 0% de uma componente
  - 1.0 - intensidade de 100% de uma componente

Iluminação com OpenGL

- Os cálculos referentes a iluminação ocorre em todos os vértices da cena renderizada
- Toda a iluminação é calculada com referência à normal especificada para cada vértice através da função glNormal
  - se para um mesmo triângulo se configurar duas ou três normais diferentes, então será feita uma interpolação destas normais para que se calcule a intensidade de luz sobre a superfície deste triangulo

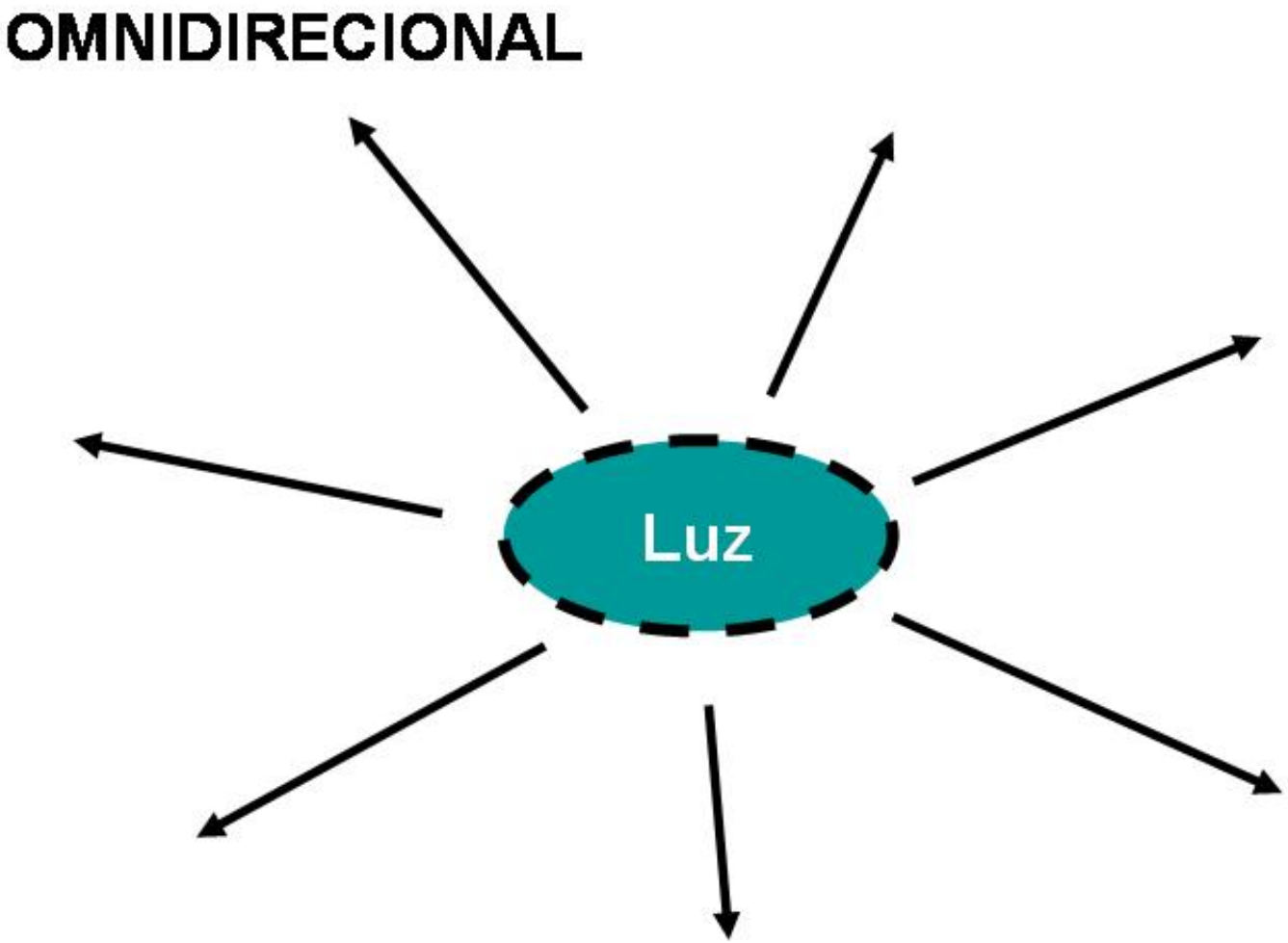
Iluminação com OpenGL

- Como as normais são afetadas pelas matrizes de visão e projeção, então é de extrema importância que se reescale as normais ao realizar uma operação de escala sobre estas matrizes. O OpenGL possui um estado de auto-normalização de normais GL_NORMALIZE que pode ser aplicado quando se utiliza a escala uniforme

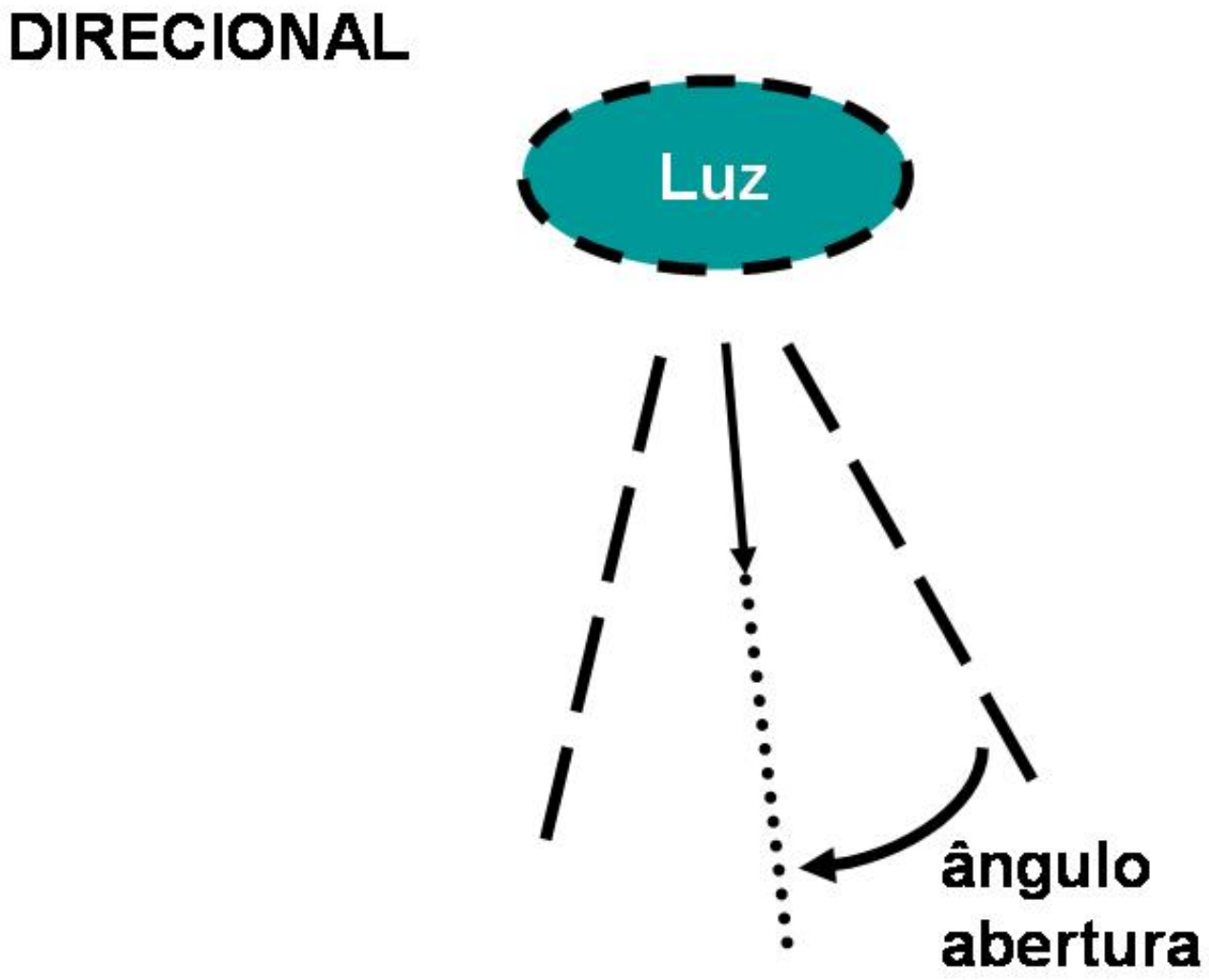
Fontes de luz

- **Como no modelo de Phong, existem três atributos que podem ser configurados:**
  - **ambiente**
  - **difuso**
  - **reflexivo**
- **Existem três tipos de fontes de luzes:**
  - **Omnidirecionais**
  - **direcionais**
  - **paralelas**

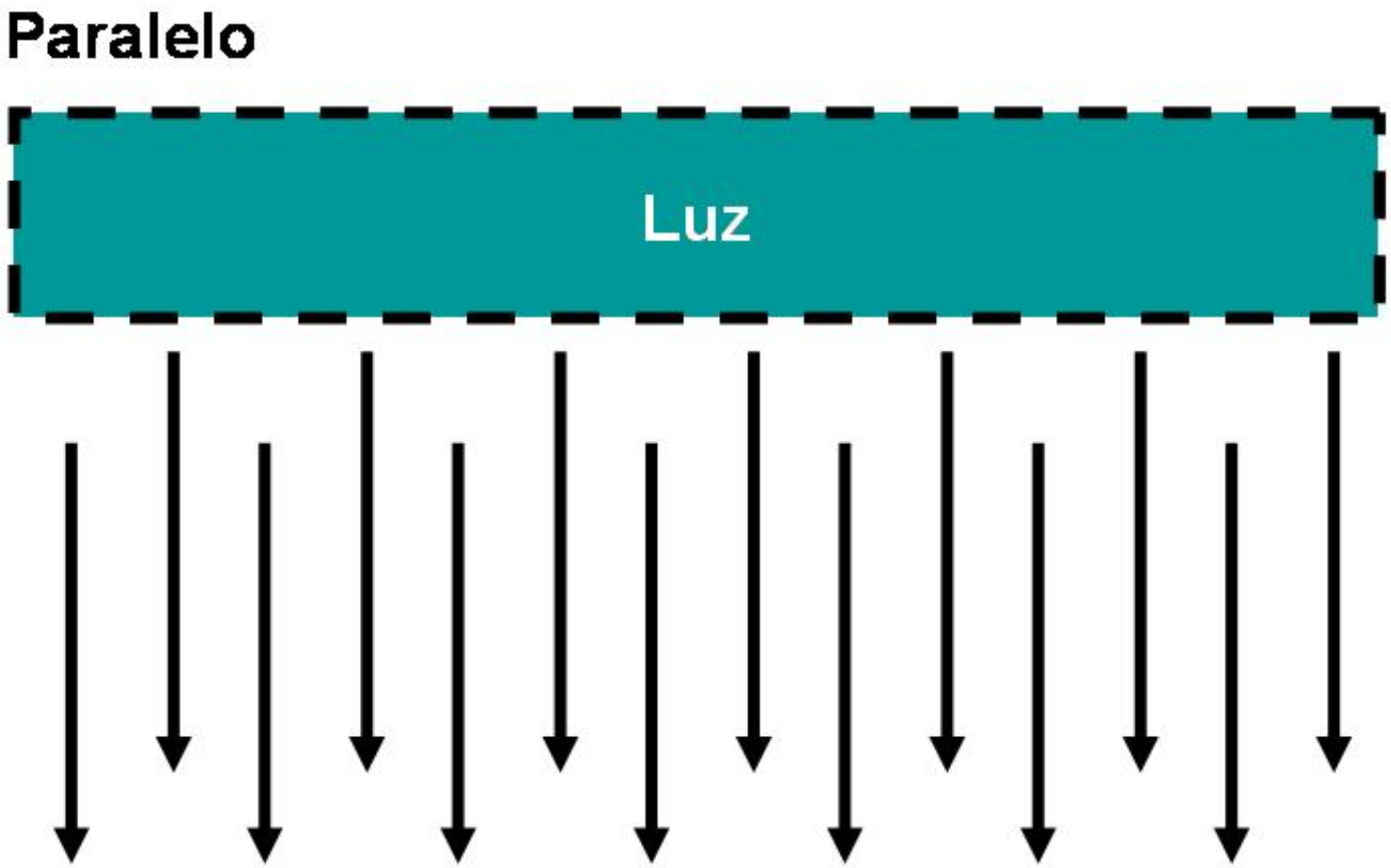
Fontes de luz



Fontes de luz



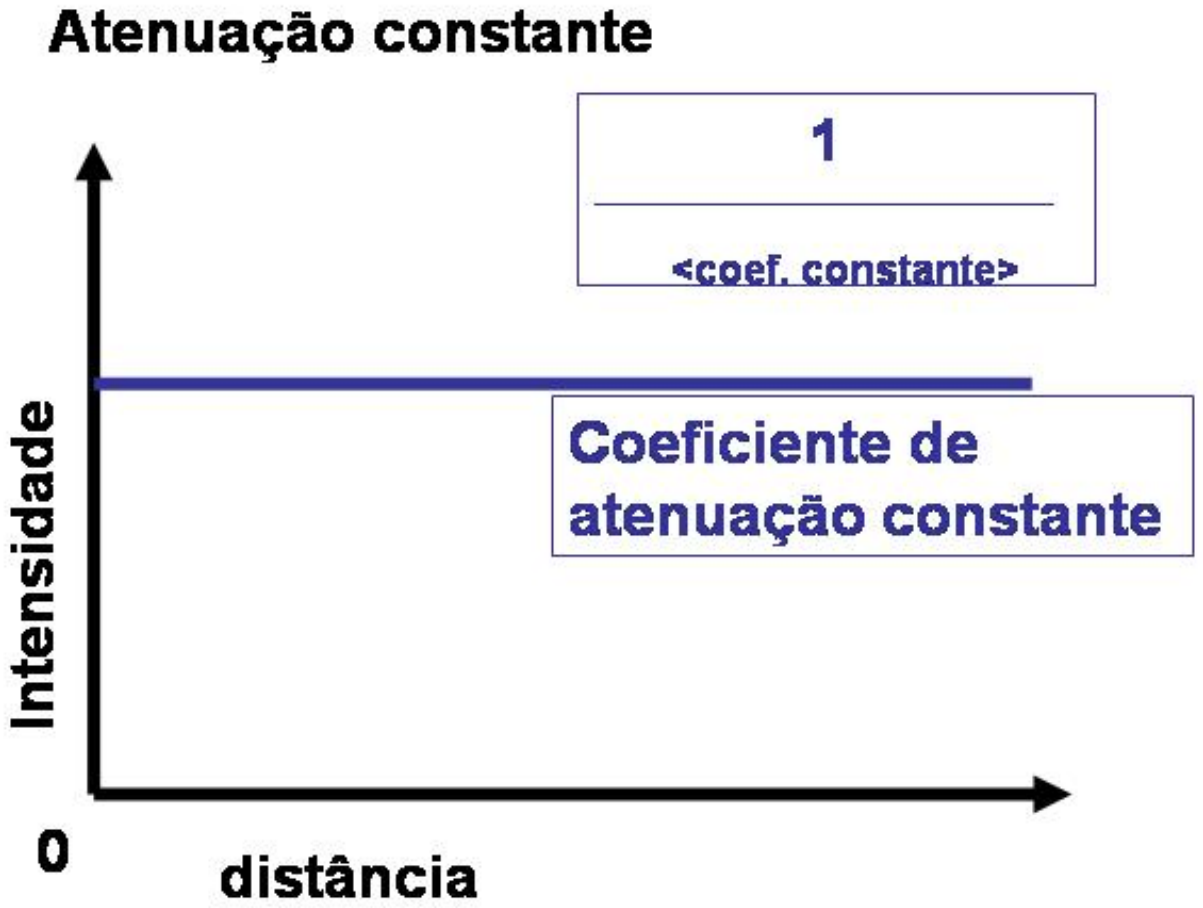
Fontes de luz



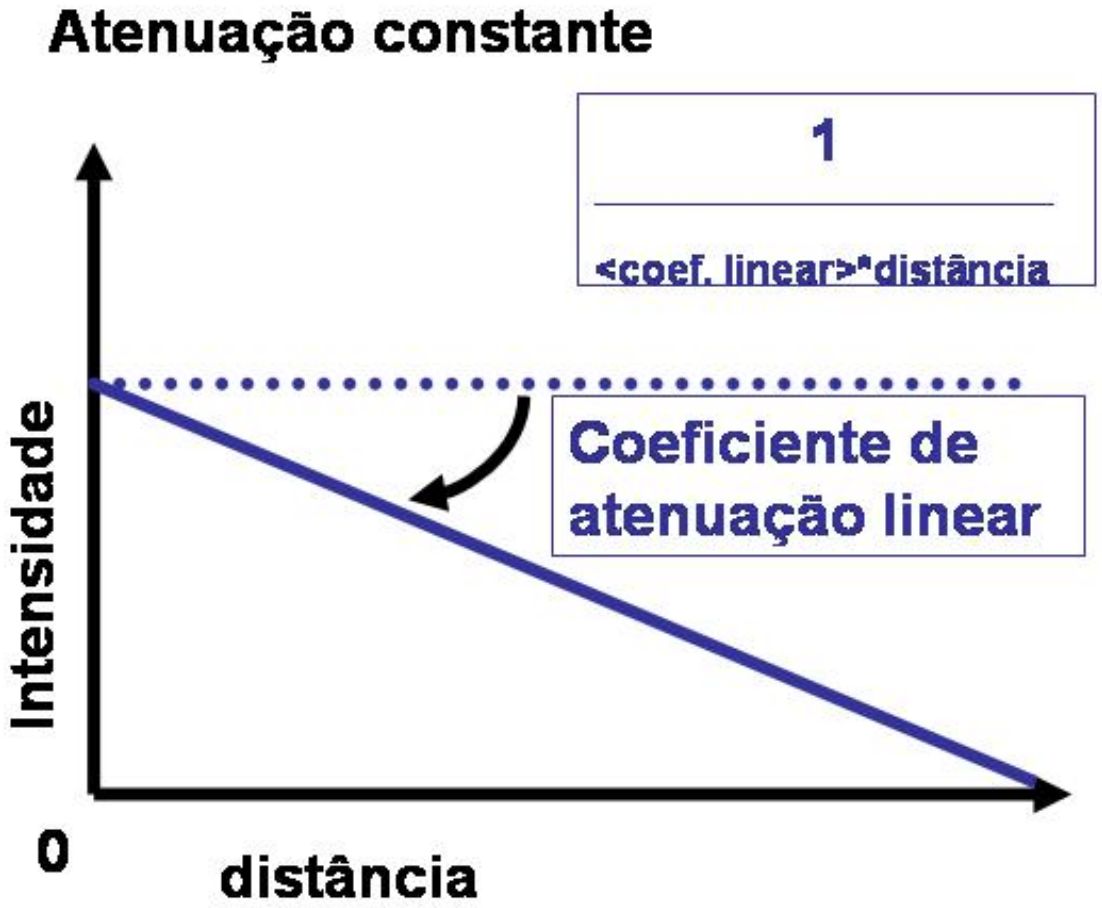
Fontes de luz

- **A luz ao viajar pelo espaço pode perder intensidade, esta perda pode ser configurada através dos atribudos:**
  - **GL_CONSTANT_ATTENUATION**
  - **GL_LINEAR_ATTENUATION**
  - **GL_QUADRATIC_ATTENUATION**

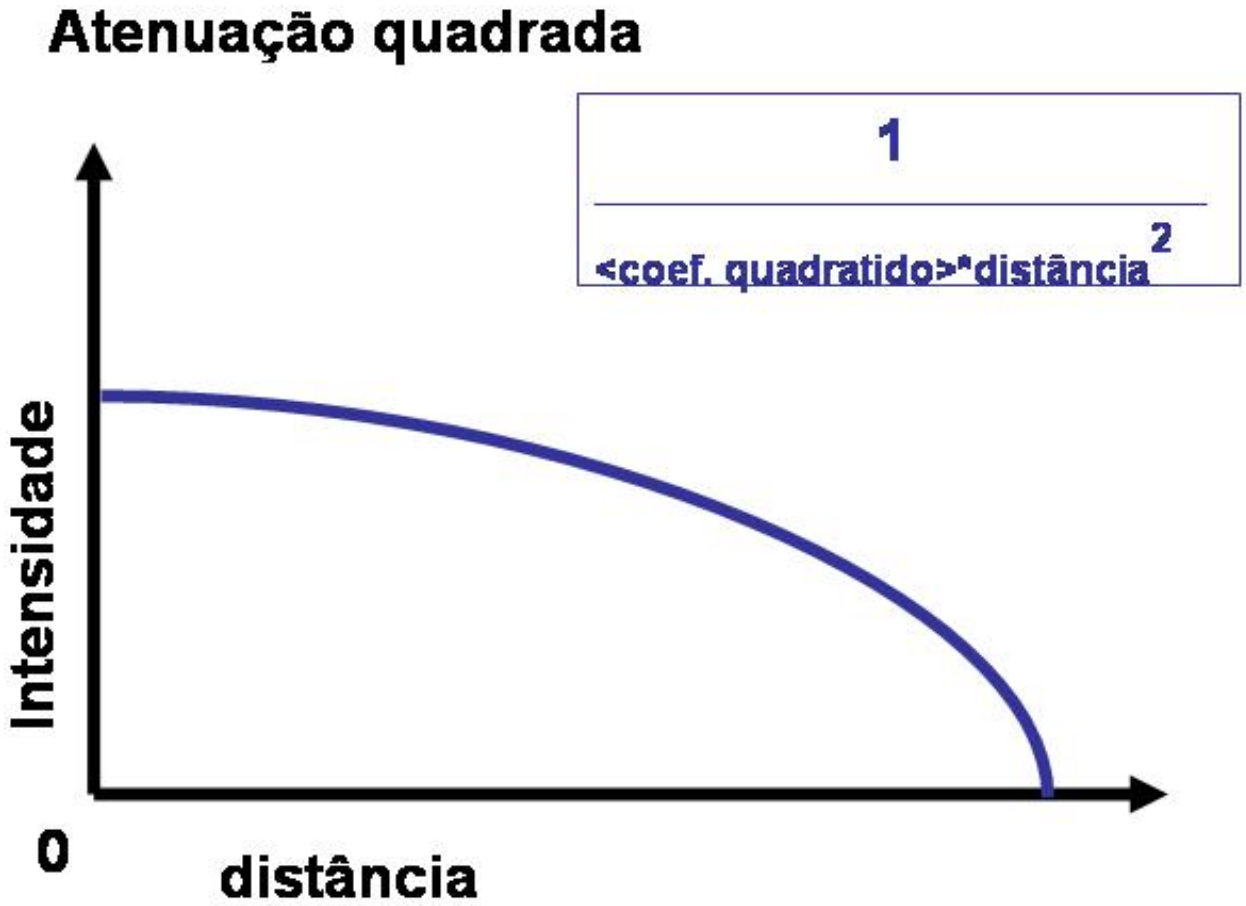
Fontes de luz



Fontes de luz



Fontes de luz



Criando fontes de luz

- Para se utilizar uma luz no OpenGL é necessário habilitá-la
- É possível que se use até 8 luzes simultaneamente (GL_LIGHT0,..., GL_LIGHT7)
- Os cálculos de iluminação, quando feitos pela API do OpenGL são realizados de forma dinâmica

Criando fontes de luz

void glLight{fd}{v}(Luz, Atributo, Valor ou vetor)

Configura algum atributo de uma luz

GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR,

GL_POSITION, GL_SPOT_DIRECTION,

GL_SPOT_EXPONENT, GL_SPOT_CUTOFF,

GL_CONSTANT_ATTENUATION,

GL_LINEAR_ATTENUATION,

GL_QUADRATIC_ATTENUATION

Materiais

- Material é um conjunto de propriedades de um objeto que descrevem seu comportamento frente a interação com a luz
- É o que diz se um objeto reflete mais difusamente que especularmente
- A propriedade Material faz parte da máquina de estados
- Ao habilitar GL_COLOR_MATERIAL as informações de cores dos vértices sobreescrevem algum atributo do material atual

Materiais

- **Para o OpenGL é possível se configurar 4 tipos de atributos de um material:**
  - **Ambiente**
  - **Difuso**
  - **Emissivo (pseudo-luz de um objeto)**
  - **Reflexivo**
- **Estes atributos são combinados com as respectivas propriedades da luz, com exceção do atributo Emissivo**

Materiais

void glMaterial{fd}{v}(Face,Atributo, Valor)

Configura algum atributo de um material

GL_FRONT, GL_BACK, GL_FRONT_AND_BACK

GL_DIFFUSE, GL_SPECULAR, GL_EMISSION, GL_SHININESS

Materiais

void glColorMaterial{fd}{v}(Face,Atributo)

Configura algum atributo de um material

GL_FRONT, GL_BACK, GL_FRONT_AND_BACK

GL_DIFFUSE, GL_SPECULAR, GL_AMBIENT, GL_AMBIENT_AND_DIFFUSE, GL_EMISSION

Exercícios

Texturas

Sumário

- **Visão Geral**
- **Texturas com OpenGL**
- **Habilitando o mapeamento de texturas**
- **Criando objetos de texturas**
- **Especificando filtros**
- **Carregando imagens**
- **Mapeamento de texturas**
- **Combinação e textels a pixels**

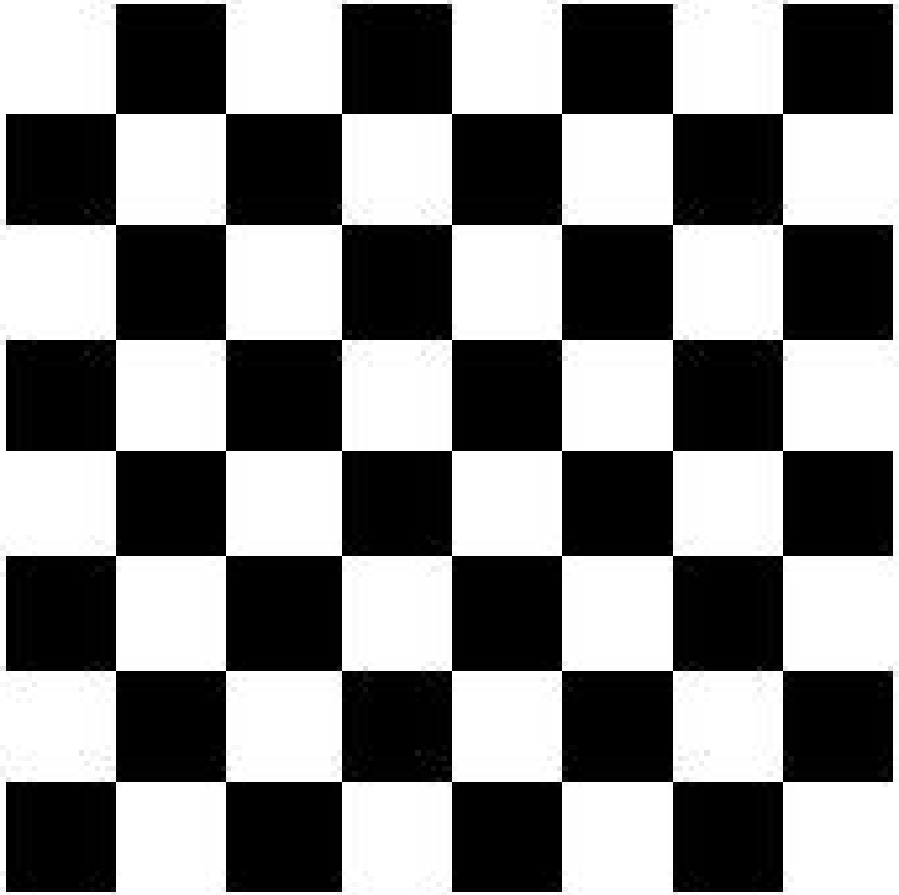
Sumário

- **MIPMAP**
- **Geração automática de coordenadas de textura**
- **Matriz de textura**
- **Conceito de multitexturização**
- **Apresentação do 'The Gimp'**

Visão Geral
Motivos de utilização

- **Nem tudo em uma cena deve ser renderizado na forma de triangulos**
- **Facilita o trabalho de acréscimo de detalhes à cena**

Visão Geral
Um exemplo

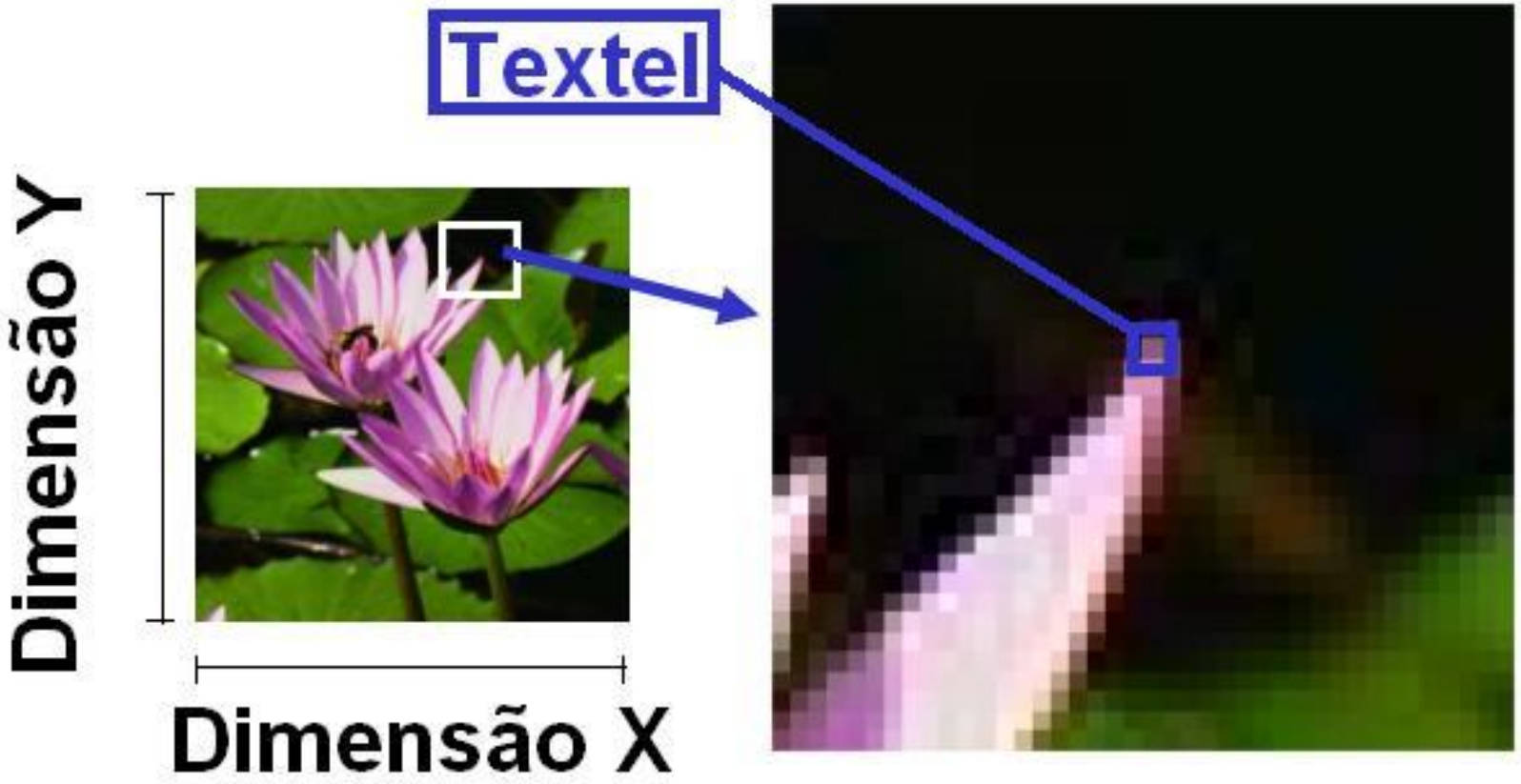


Visão Geral
Um exemplo

Visão Geral
O que é uma textura?

- Uma imagem que é aplicada a uma superfície (a um polígono)
- É composta por textels
- Os textels podem ser pensados como os pixels de uma imagem 2D

Visão Geral
O que é uma textura?



Texturas com OpenGL

- **Uma área de memória, composta por textels, que possui uma determinada dimensão**
- **Cada textel pode possuir os formatos de componentes abaixo:**
  - **GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_RGBA, GL_LUMINANCE, GL_LUMINANCE_ALPHA**
- **O componente básico é inteiro e possui 8Bits (0 a 255), a combinação RBG possui 24 bits**

Texturas com OpenGL

- **A dimensão da textura deve ser potência de 2. ex.: 1x1, 2x2, 4x4, 8x8, 16x16, 32x32, 64x64, 128x128, 4x128**
- **Esta dimensão possui um valor máximo, que é específico em cada hardware**

Texturas com OpenGL

Resolução (256x128)

Dimensão Y:
real (0 a 127)
OGL (0.0 a 1.0)

0

Dimensão X:
real (0 a 255)
OGL (0.0 a 1.0)

Texturas com OpenGL

- O OpenGL não fornece meios de carregar arquivos que contenham imagens (.bmp, .jpg, .tga, ...)
- Fornece sim um mecanismo de "criação" de objetos texturas que se associam a um buffer de uma imagem (que foi carregado de alguma forma)
- Normalmente ao se criar um objeto textura, este é convertido para um formato otimizado para o hardware em utilização

Texturas com OpenGL

- **Ao se trabalhar com uma textura tem-se uma abstração quanto à real dimensão da imagem utilizada**
- **Esta dimensão está entre os limites (0.0 até 1.0)**

Texturas com OpenGL

Resolução (256x128)

Dimensão Y:
real (0 a 127)
OGL (0.0 a 1.0)

0

Dimensão X:
real (0 a 255)
OGL (0.0 a 1.0)

Habilitando o mapeamento de texturas

- Como qualquer outro estado do OpenGL a utilização de textura também deve ser habilitada
- A constante que se refere a ativação do mapeamento de textura é o GL_TEXTURE_2D

Criando objetos de texturas

- **É necessário se reservar um identificador de textura no OpenGL**
- **Através deste identificador é possível manipular as texturas**
- **Como em um programa normalmente se utiliza mais de uma textura, então é necessário ativar um identificador de textura para indicar que se está trabalhando com uma textura específica**

Criando objetos de texturas

glGenTextures(GLsizei n,GLuint*textures)

Reserva uma Quantidade 'n' de Identificadores de Texturas e os armazena no vetor apontado por '*textures'.

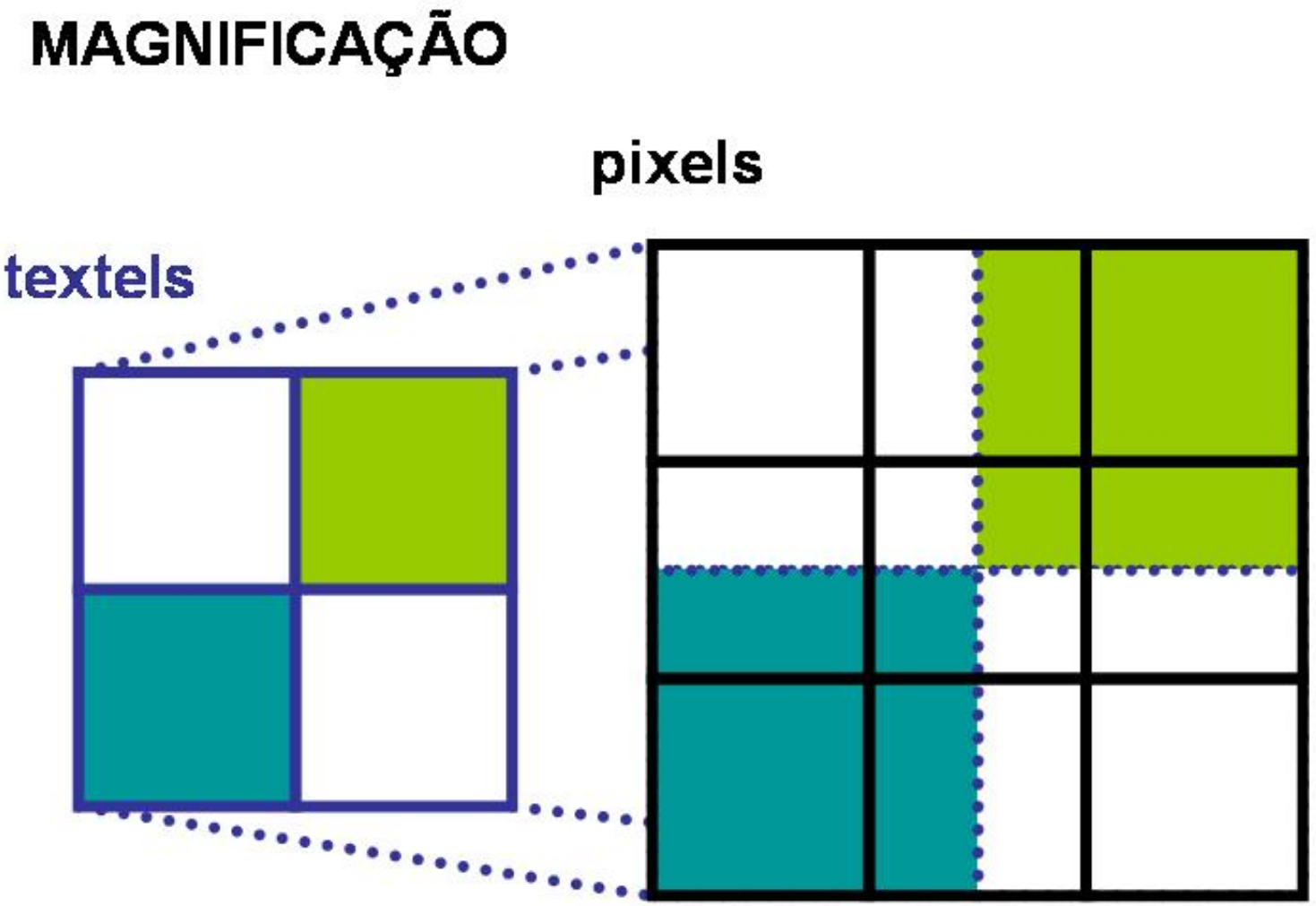
glBindTexture(GLenum target,Gluint texture)

Ativa um tipo de textura indicada por 'target' que pode ser GL_TEXTURE_2D ou GL_TEXTURE_1D que possui o identificador 'texture'

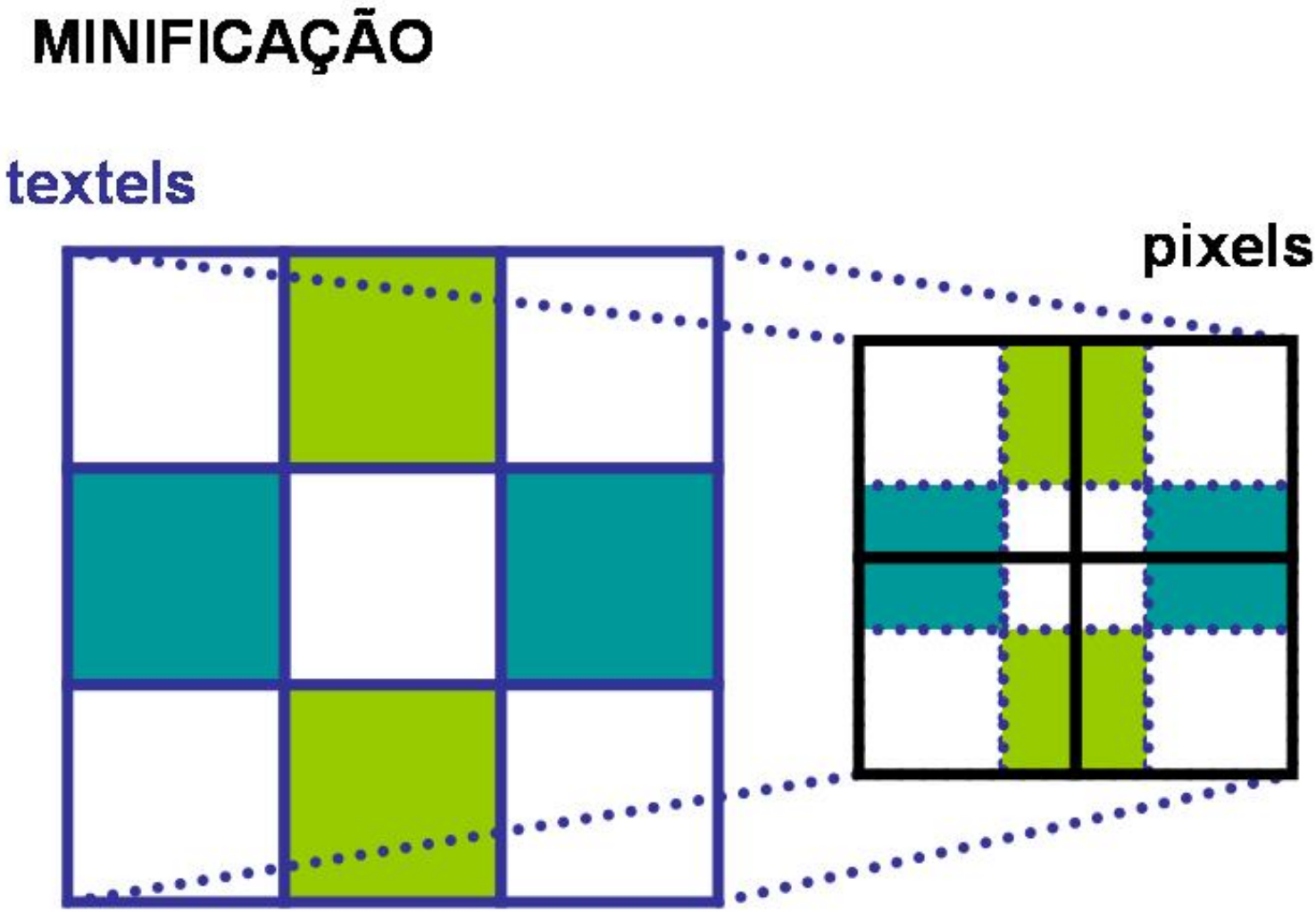
Especificando filtros

- Os filtros que podem ser aplicados ás texturas são equivalentes aos que se aplicam à imagens
- Existem duas situações onde é inevitável que se filtre uma textura:
  - A magnificação. Quando um textel necessita de ser 'desenhado' em mais de um pixel
  - A minificação. Quando um textel necessita de ser 'desenhado' em menos de um pixel
- No OpenGL é possível determinar se é ou não para se filtar a textura em ambos os casos citados acima

Especificando filtros



Especificando filtros



Especificando filtros

glTexParameterf(GLenum target, GLenum pname, GLfloat param)

---

target = GL_TEXTURE_2D ou GL_TEXTURE_1D

pname = GL_TEXTURE_MIN_FILTER ou GL_TEXTURE_MAG_FILTER

param = GL_NEAREST ou GL_LINEAR

Carregando imagens

- O OpenGL não carrega imagens de arquivos, mas sim fornece um mecanismo de associação de buffers
- O carregamento de imagens é feito para a API do OpenGL, onde se tenha um buffer de imagem em um determinado formato e a converta-a para o formato interno do OpenGL

Carregando imagens

glTexImage2D (GLenum target, GLint level, GLint internalformat, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid *pixels)

---

target = GL_TEXTURE_2D ou GL_TEXTURE_1D

level = nível da hierarquia de MIPMAP

internalformat = número de componentes que possuirá a textura (1,2,3 ou 4)

border = (0 ou 1)

Carregando imagens

glTexImage2D (GLenum target, GLint level, GLint internalformat, GLsizei width, GLsizei height, GLint border, GLenum format, GLenum type, const GLvoid *pixels)

---

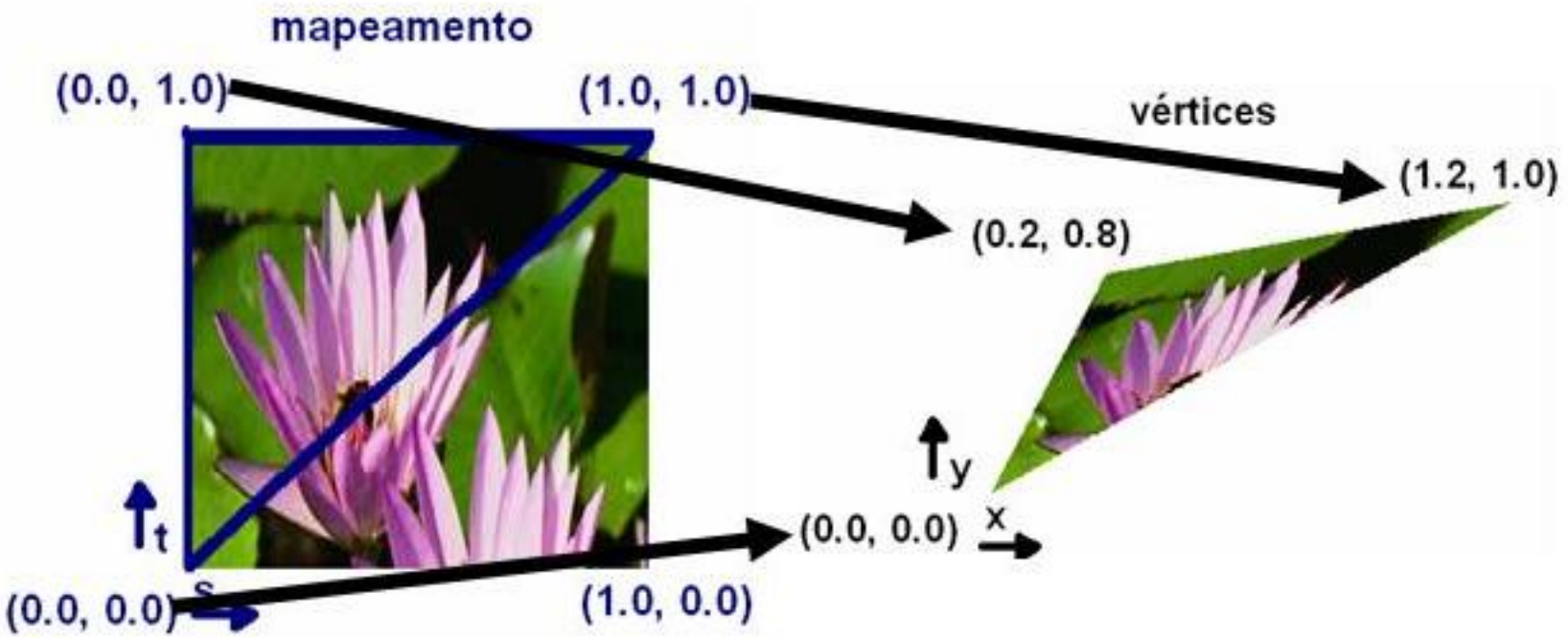
format = GL_RED, GL_GREEN, GL_BLUE, GL_ALPHA, GL_RGB, GL_RGBA, GL_LUMINANCE, GL_LUMINANCE_ALPHA

type = GL_UNSIGNED_BYTE, GL_BYTE, GL_BITMAP, GL_UNSIGNED_SHORT, GL_SHORT, GL_UNSIGNED_INT, GL_INT, GL_FLOAT

Mapeamento de texturas

- O mapeamento é feito associando uma coordenada de textura (s e t) a um vértice através da função glTextCoord
- A proporção desta distância entre textels será mantida

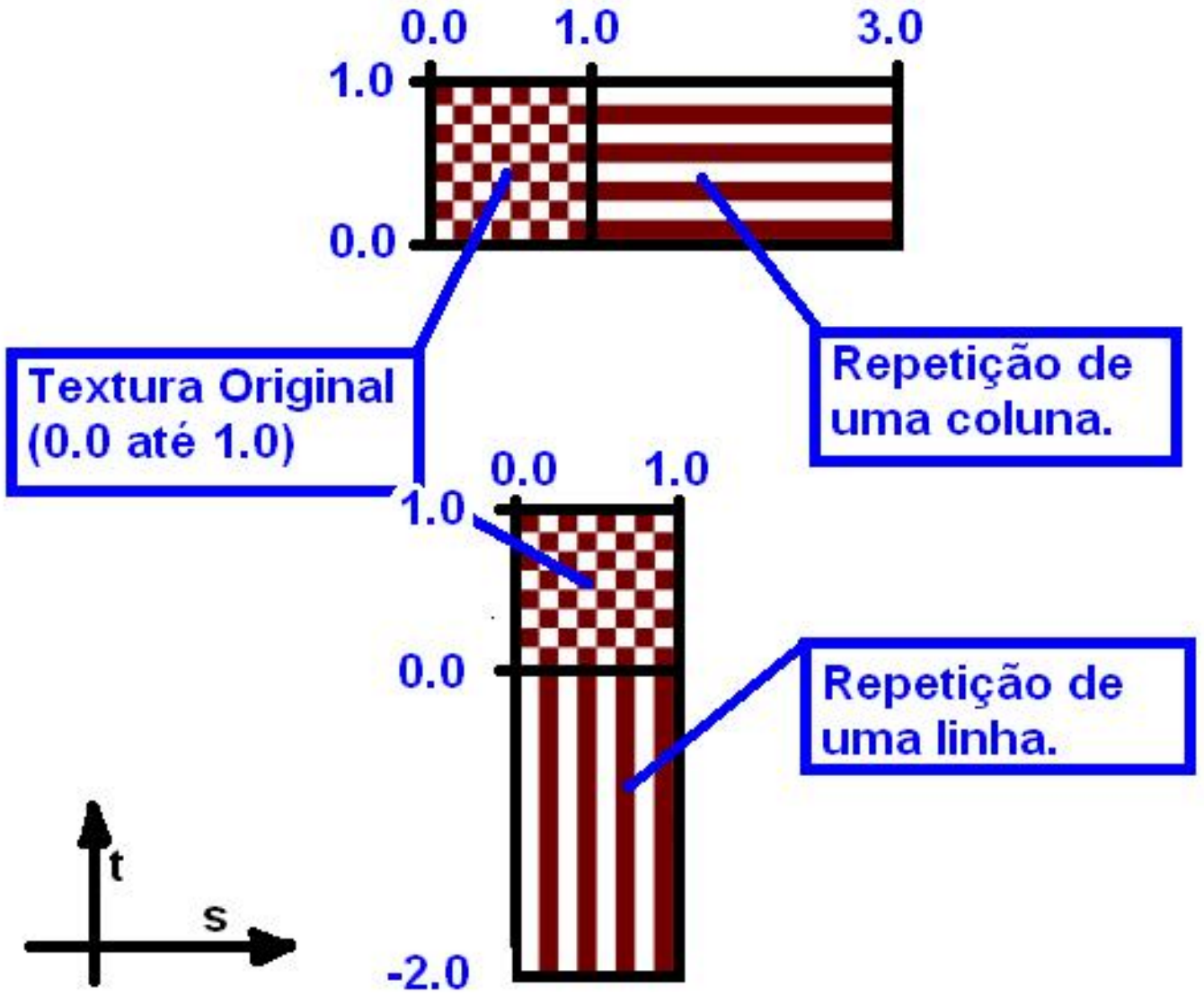
Mapeamento de texturas



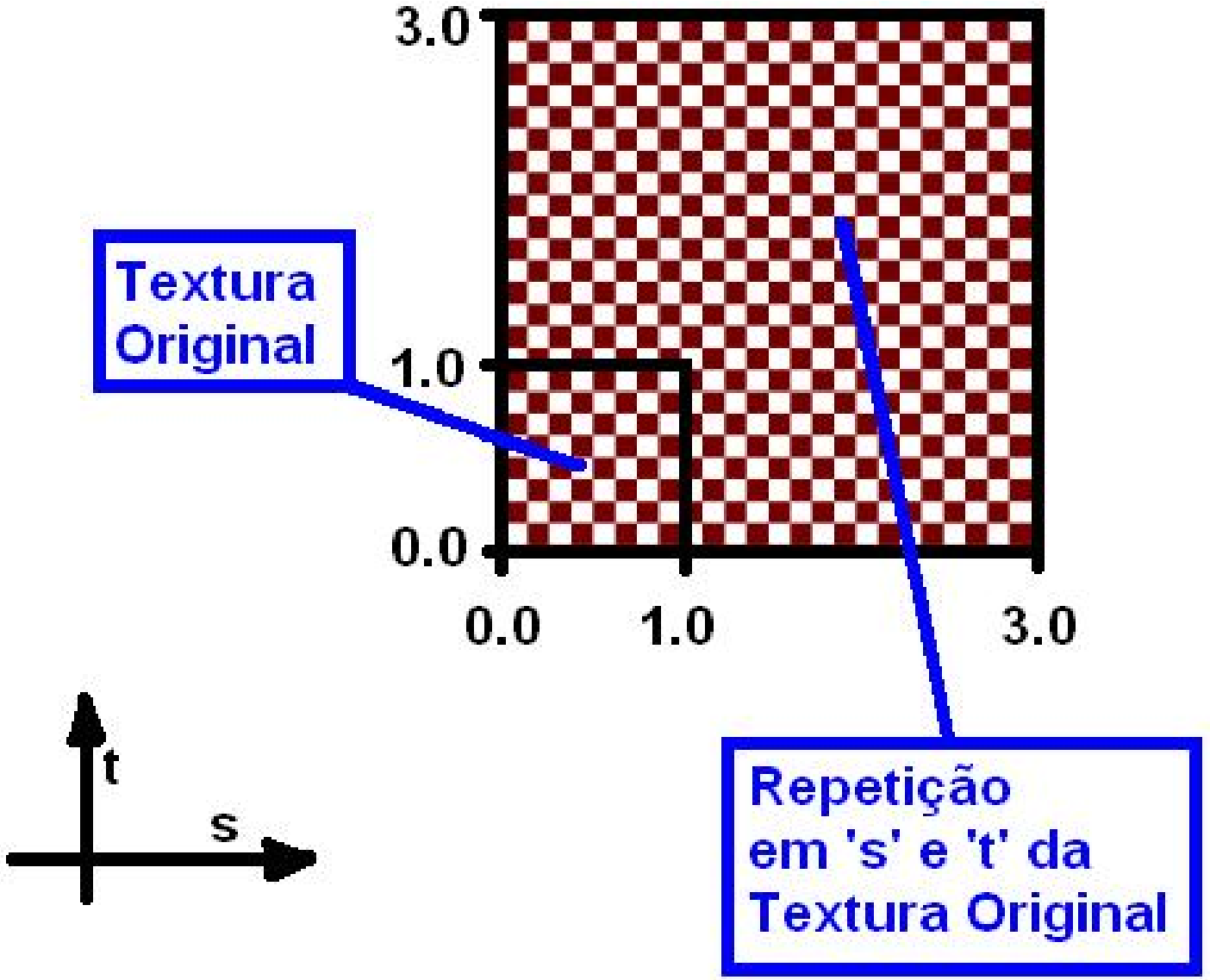
Mapeamento de texturas

- **É possível que se mapeie uma área de textura que não existe, ou seja, está fora do limite de 0.0 a 1.0. Quando isto ocorre podem acontecer duas coisas:**
  - **A textura ser repetida a fim de preencher os espaços mapeados fora do limite**
  - **Se repetir a última linha ou coluna da textura a fim de preencher os espaços mapeados fora do limite**

Mapeamento de texturas



Mapeamento de texturas



Mapeamento de texturas

glTexParameterf(GLenum target, GLenum pname, GLfloat param)

---

target = GL_TEXTURE_2D ou GL_TEXTURE_1D

pname = GL_TEXTURE_WRAP_S, GL_TEXTURE_WRAP_T

param = GL_REPEAT, GL_CLAMP

Combinação e textels a pixels

glTexEnvi(GLenum target, GLenum pname, GLint param)

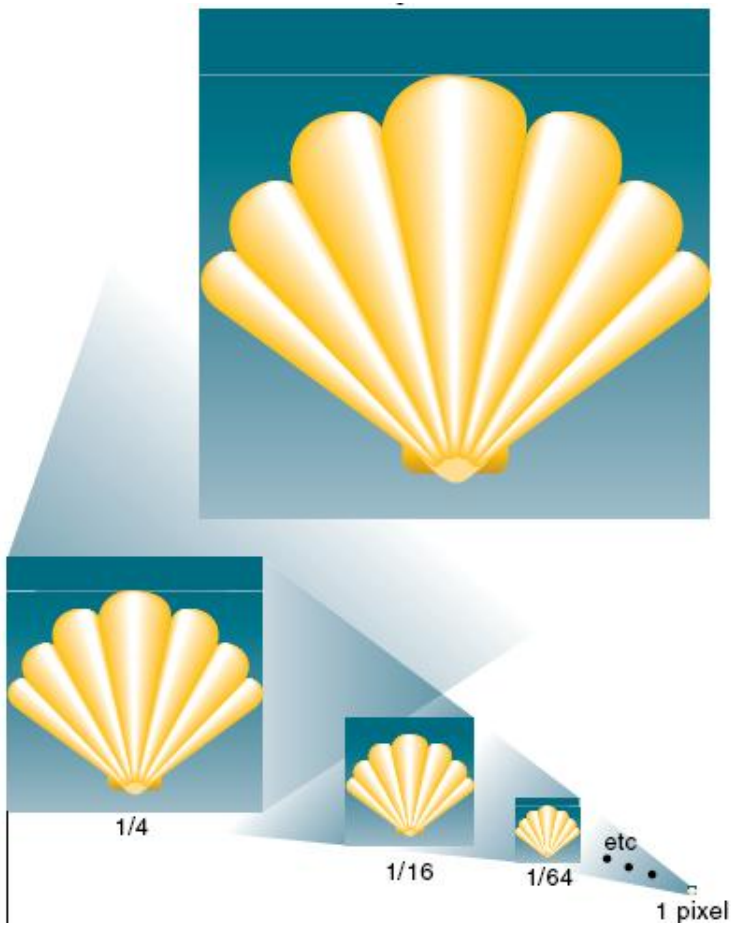
---

target = GL_TEXTURE_ENV

pname = GL_TEXTURE_ENV_MODE ou GL_TEXTURE_ENV_COLOR

param = GL_MODULATE, GL_DECAL, GL_BLEND, ou um vetor RBG

MIPMAP

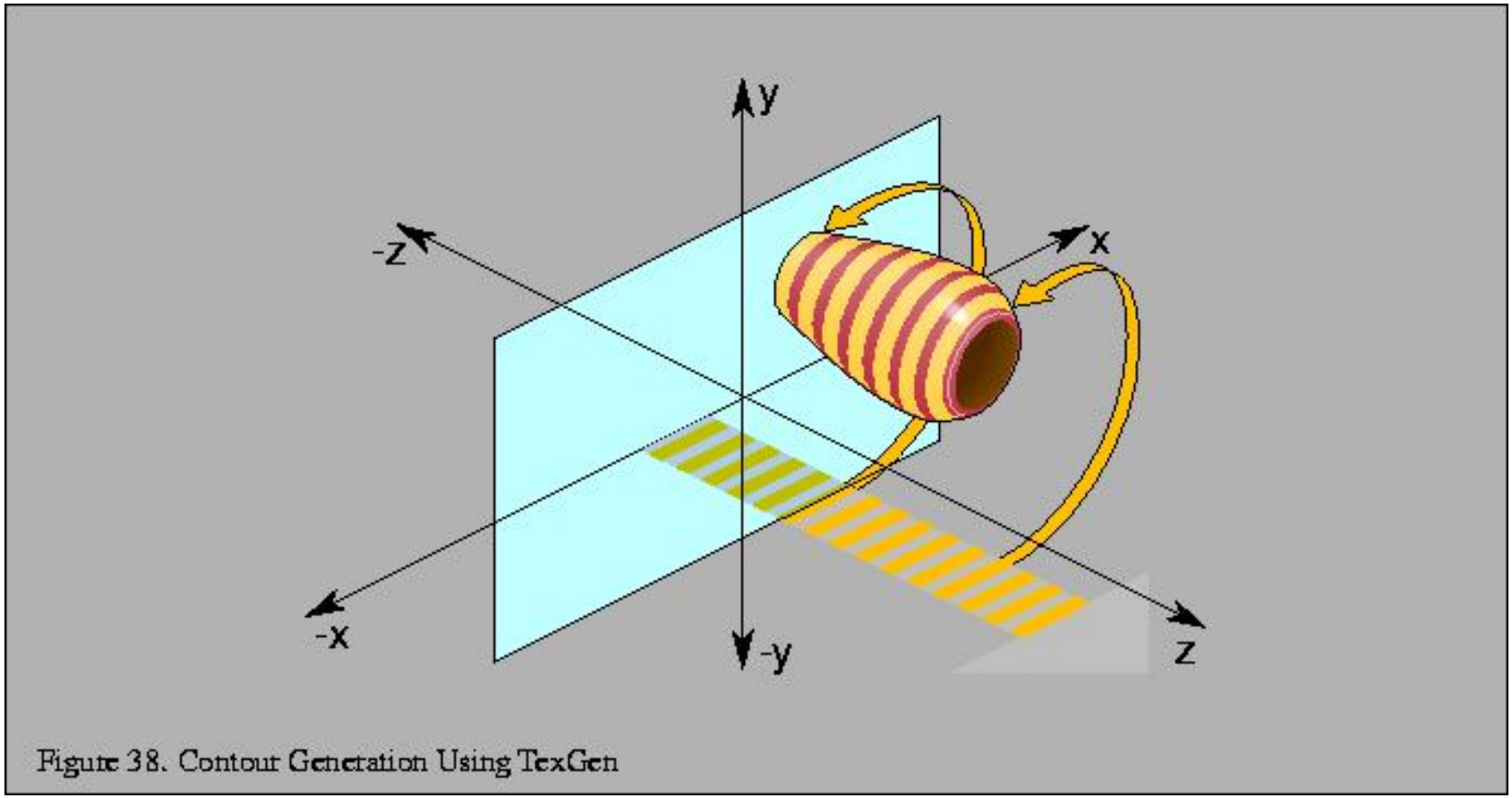


MIPMAP

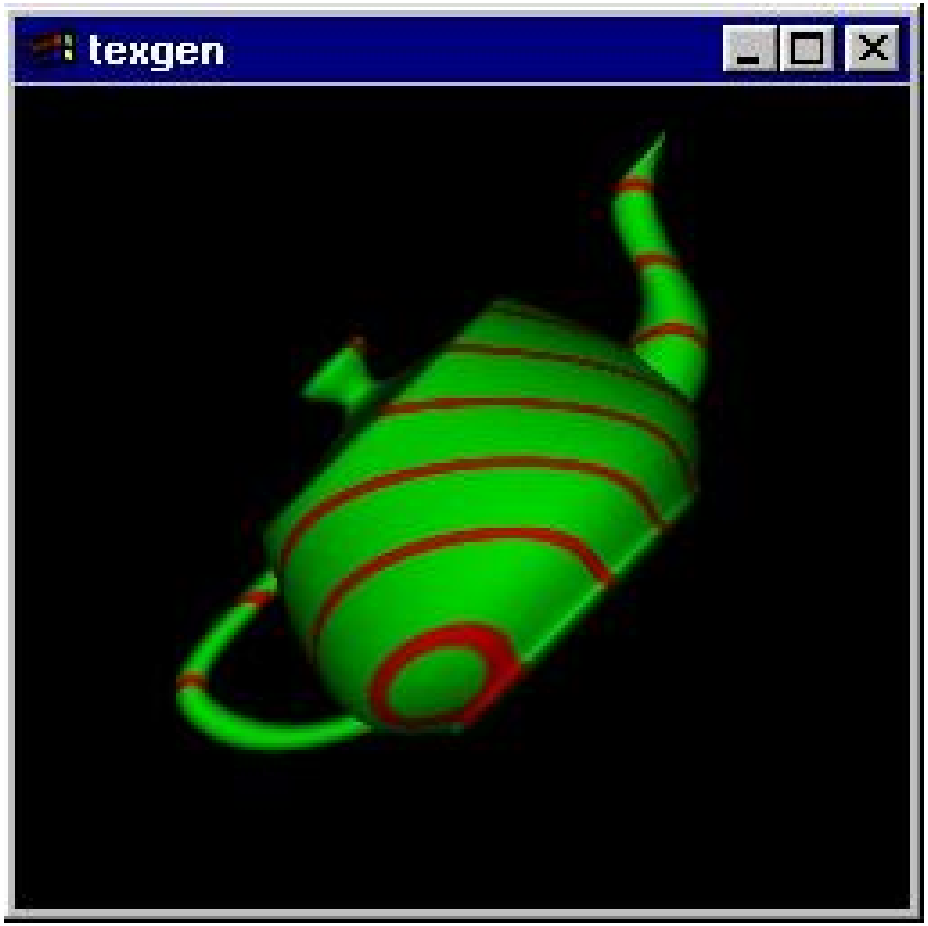
gluBuild2DMipmaps (GLenum target, GLint internalformat, GLsizei width, GLsizei height, GLenum format, GLenum type, const GLvoid *pixels)

Os parâmetros possuem o mesmo significado do glTexImage2D

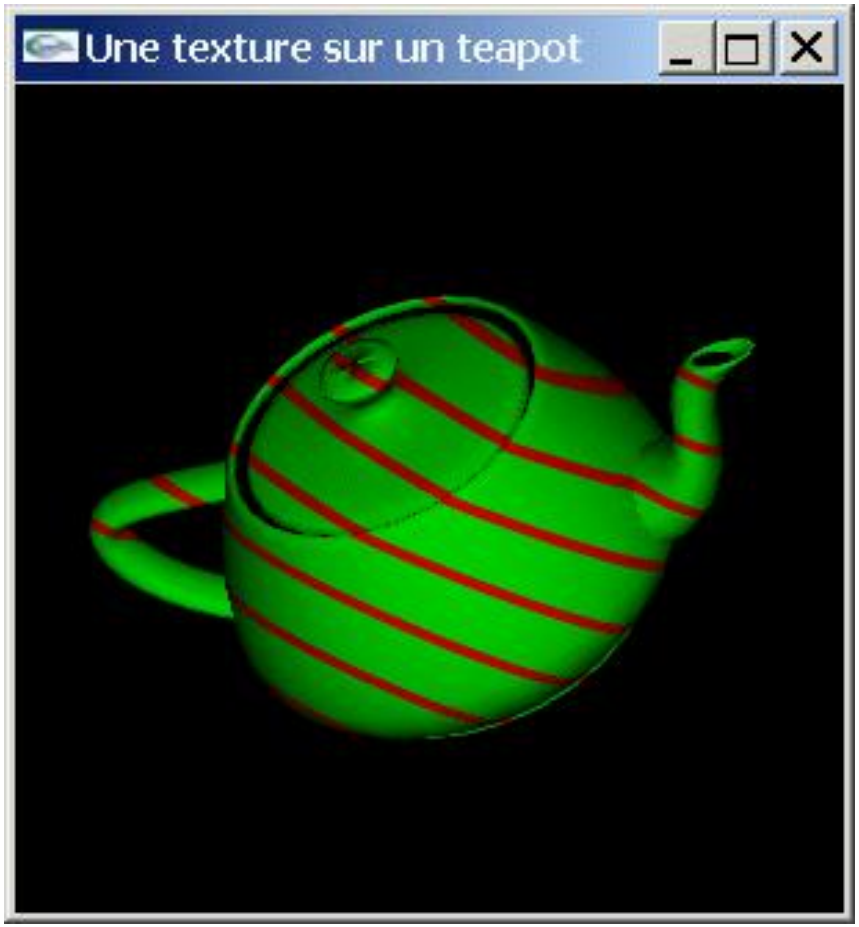
Geração automática de coordenadas de textura



Geração automática de coordenadas de textura



Geração automática de coordenadas de textura



Geração automática de coordenadas de textura

Geração automática de coordenadas de textura

void glTexGen{ifd}{v}(GLenum coord, GLenum pname, TYPE *param)

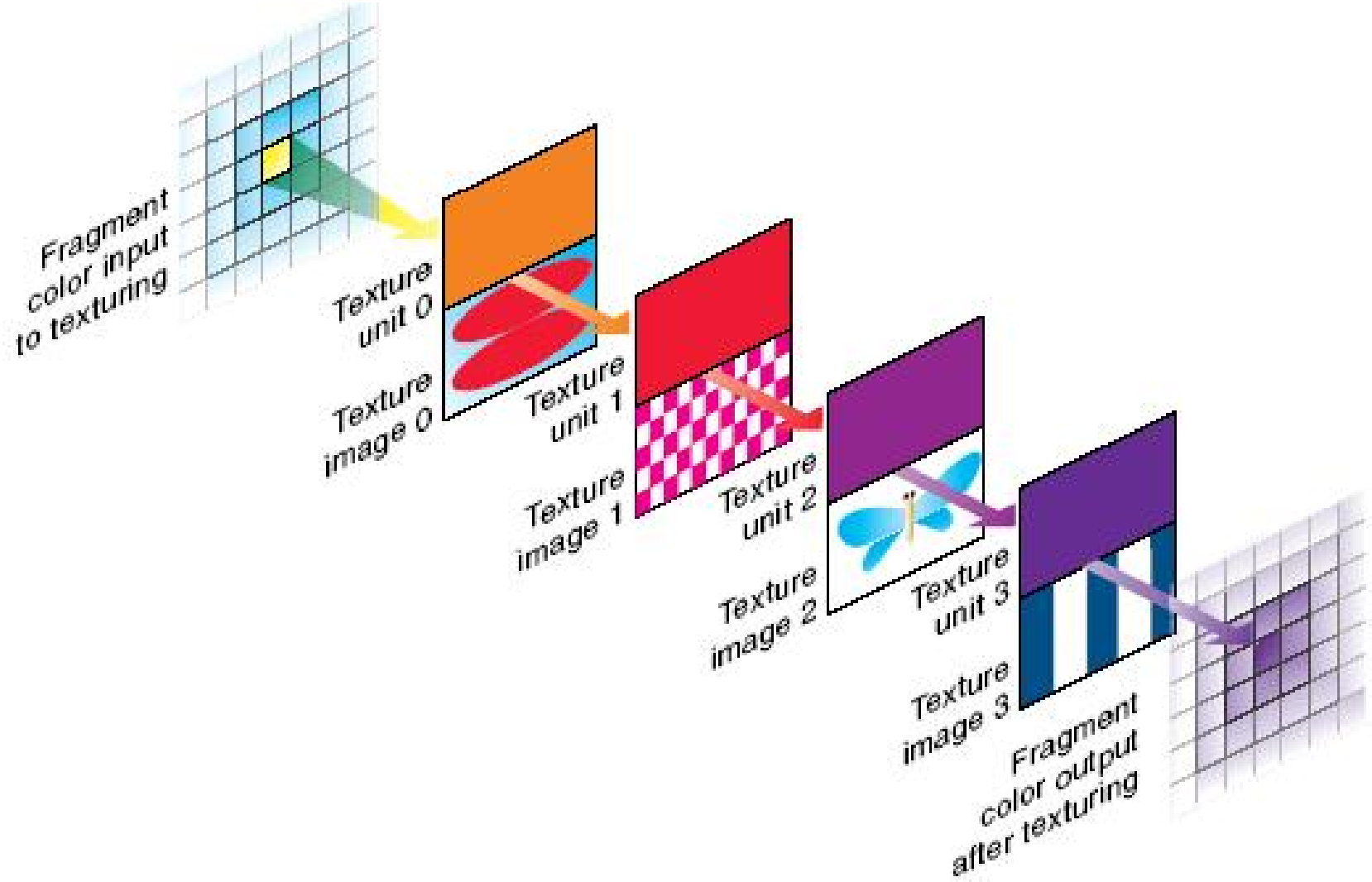
---

coord = GL_S, GL_T, GL_R, or GL_Q

pname = GL_TEXTURE_GEN_MODE, GL_OBJECT_PLANE, or GL_EYE_PLANE

GL_TEXTURE_GEN_MODE -¿ GL_OBJECT_LINEAR, GL_EYE_LINEAR, or GL_SPHERE_MAP

Conceito de multitexturização



Matriz de textura

Apresentação do 'The Gimp'