

# Texturas

André Tavares da Silva

[andre.silva@udesc.br](mailto:andre.silva@udesc.br)

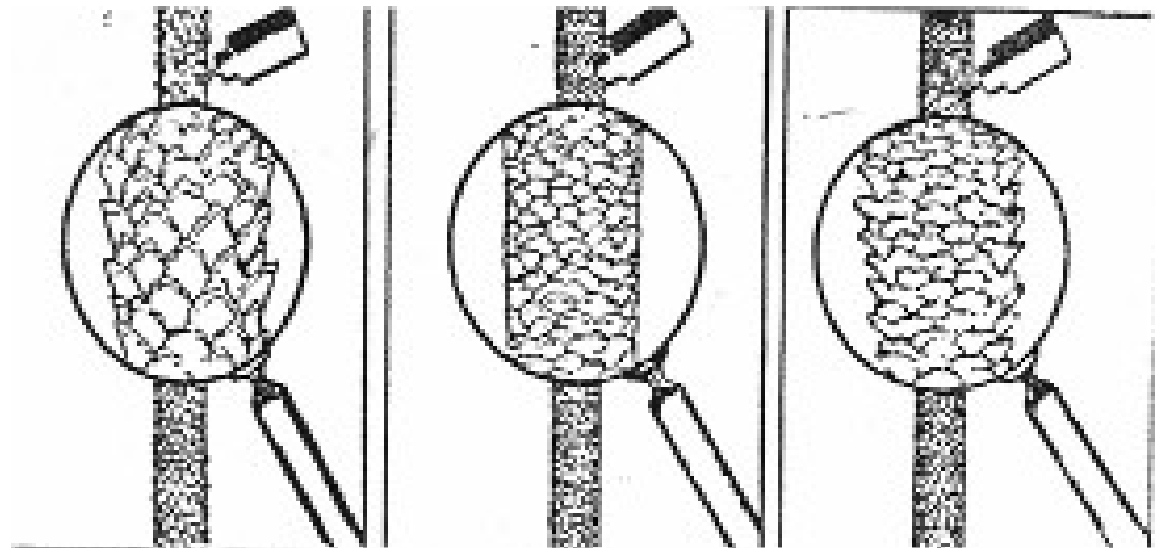
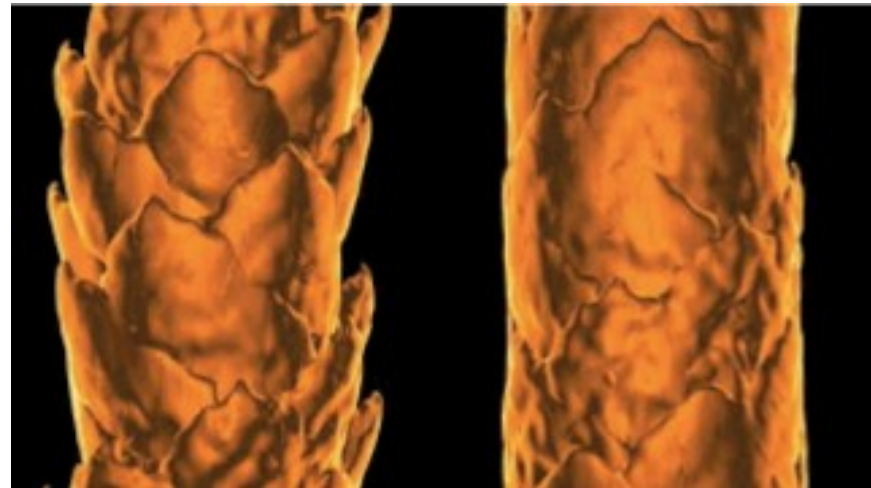
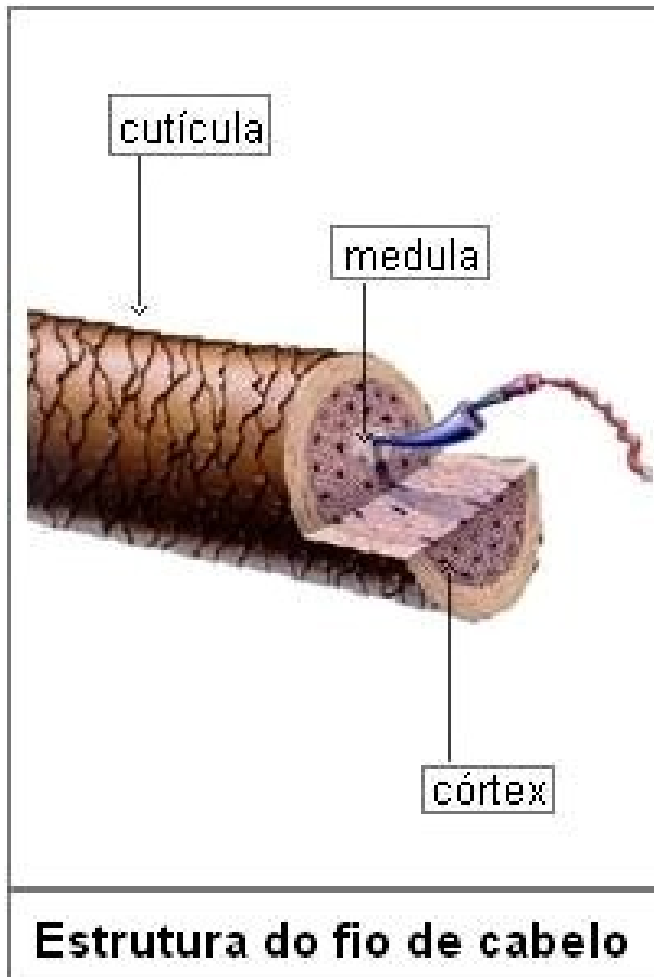
# Detalhes de Superfícies

- Modelos de iluminação não são apropriados para descrever todas as diferenças de cor observáveis em uma superfície
  - Superfícies pintadas com padrões ou imagens
    - A capa ou uma página de um livro
  - Superfícies com padrões de rugosidade
    - Tecidos ou uma parede de tijolos
- Em princípio é possível modelar esses detalhes com geometria e usando materiais de propriedades óticas distintas





# Modelar cada detalhe?

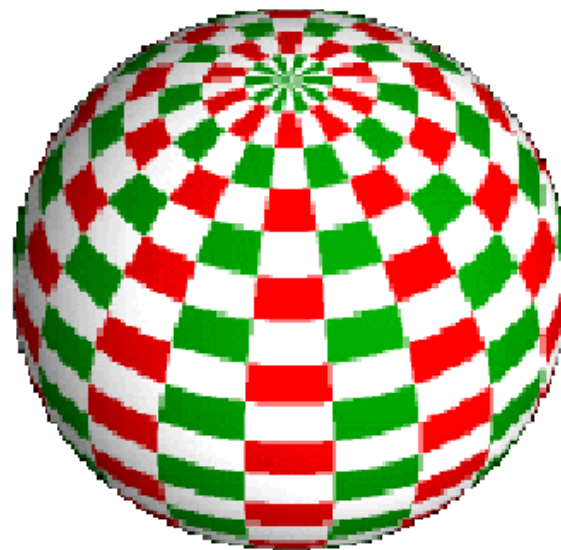
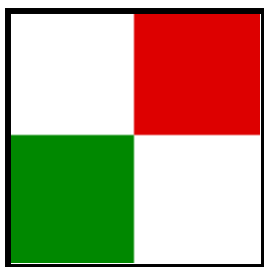


# Detalhes de Superfícies

- Modelos de iluminação não são apropriados para descrever todas as diferenças de cor observáveis em uma superfície
  - Superfícies pintadas com padrões ou imagens
    - A capa ou uma página de um livro
  - Superfícies com padrões de rugosidade
    - Tecidos ou uma parede de tijolos
- Em princípio é possível modelar esses detalhes com geometria e usando materiais de propriedades óticas distintas
- Na prática, esses efeitos são modelados usando uma técnica chamada *mapeamento de textura*

# Mapeamento de Textura

- A ideia é reproduzir sobre a superfície de algum objeto da cena as propriedades de alguma função – ou mapa - bidimensional (cor, por exemplo)



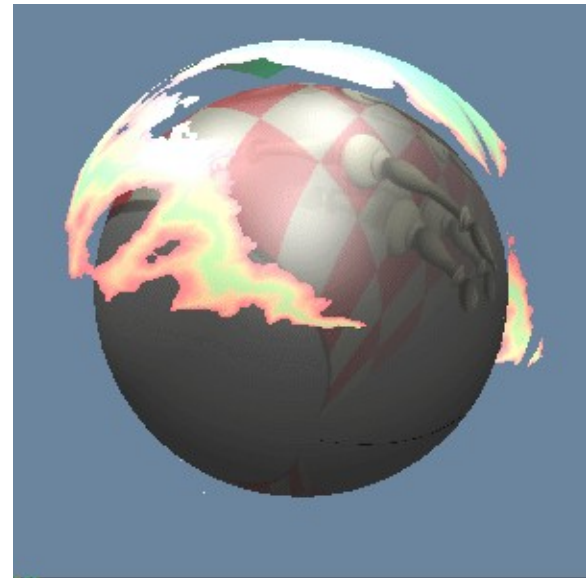
# Mapeamento de Textura





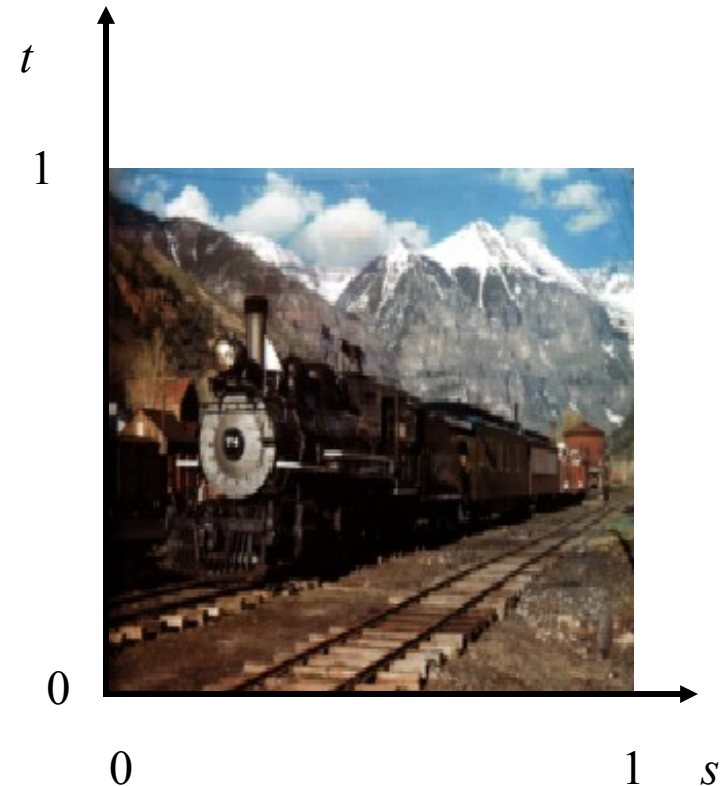
# Propriedades Mapeáveis

- Quais parâmetros ou propriedades pode-se reproduzir a partir de mapas:
  - Cor (coeficientes de reflexão difusa)
  - Coeficientes de reflexão especular e difusa
    - Mapeamento de ambiente
  - Perturbação do vetor normal
    - “Bump Mapping”
  - Perturbação da superfície na direção da normal
    - “Displacement Mapping”
  - Transparência / opacidade



# Espaço de Textura

- Texturas 2D são funções  $T(s, t)$  cujo domínio é um espaço bidimensional e o contradomínio pode ser cor, opacidade, etc
- É comum ajustar a escala da imagem de tal forma que a imagem toda se enquadre no intervalo  $0 \leq s, t \leq 1$
- Normalmente a função em si é derivada de alguma imagem capturada
  - Se a imagem está armazenada numa matriz  
 $Im [0..N-1, 0..M-1]$
  - Então  
 $T(s, t) = Im [\lfloor (1 - t) N \rfloor, \lfloor s M \rfloor]$



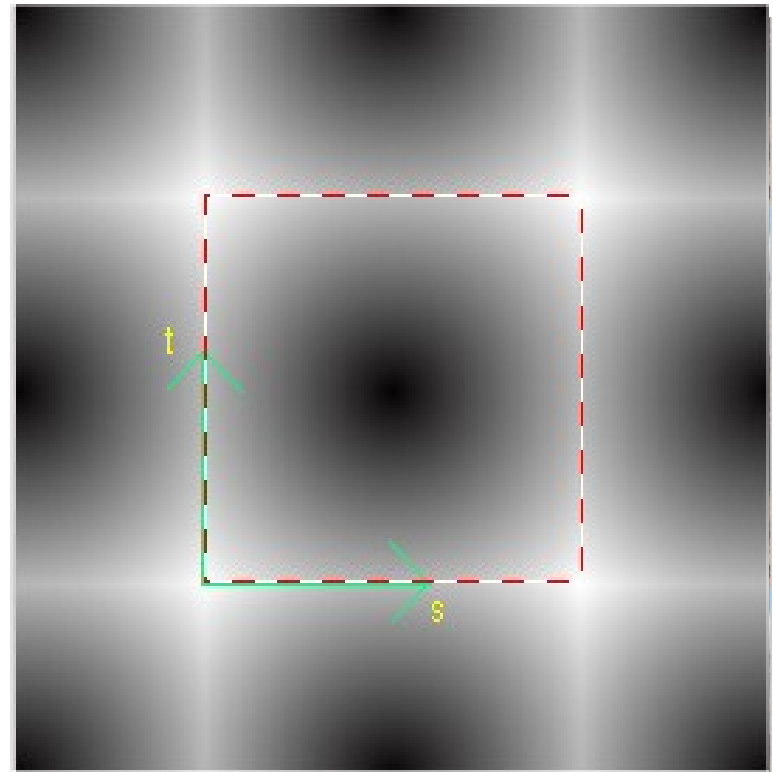
# Espaço de Textura

- Pode ser vantajoso assumir que o padrão da imagem se repete fora desse intervalo

$$T(s, t) = \begin{matrix} \text{Im} [ \lfloor (1 - t) N \rfloor \bmod N, \\ \lfloor s M \rfloor \bmod M ] \end{matrix}$$

- A função de textura pode ser também definida algebricamente:

$$T(s, t) = \sqrt{(s - 0.5)^2 + (t - 0.5)^2}$$



# Função de Mapeamento

- Retorna o ponto do objeto correspondente a cada ponto do espaço de textura
$$(x, y, z) = F(s, t)$$
- Corresponde à forma com que a textura é usada para “embrulhar” (*wrap*) o objeto
  - Na verdade, na maioria dos casos, precisamos de uma função que nos permita “desembrulhar” (*unwrap*) a textura do objeto, isto é, a inversa da função de mapeamento
- Se a superfície do objeto pode ser descrita em forma paramétrica esta pode servir como base para a função de mapeamento

# Parametrização da Esfera

*Função de mapeamento*

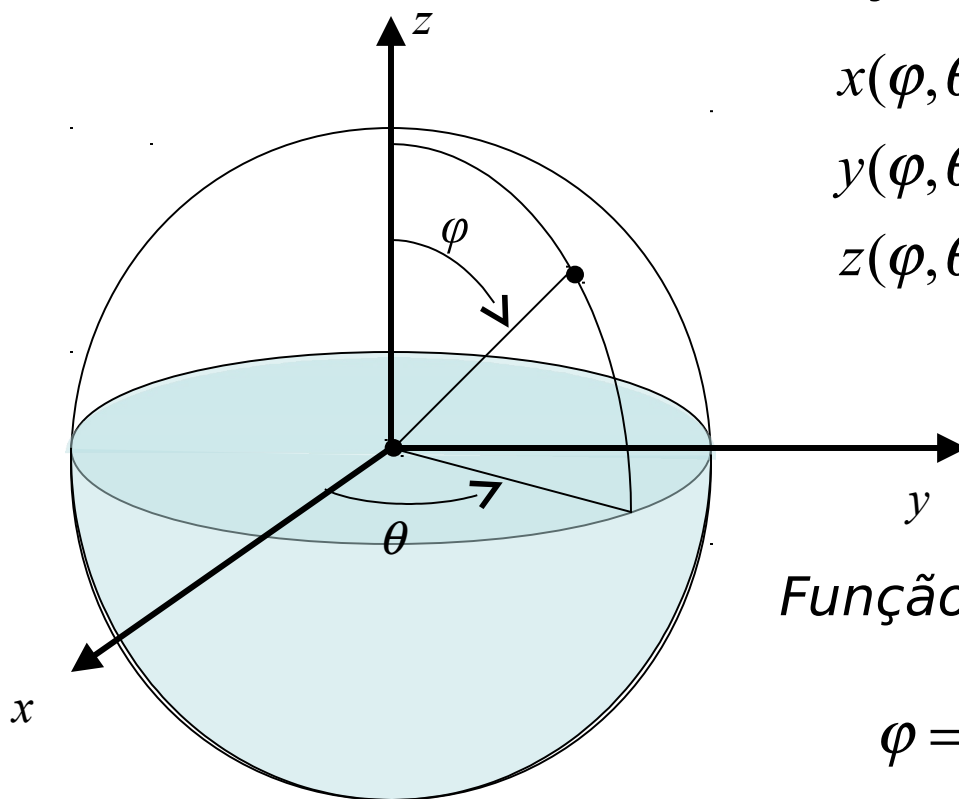
$$x(\varphi, \theta) = \sin \varphi \cos \theta$$

$$\varphi = \pi \cdot t$$

$$y(\varphi, \theta) = \sin \varphi \sin \theta$$

$$\theta = 2\pi \cdot s$$

$$z(\varphi, \theta) = \cos \varphi$$



*Função de mapeamento inversa*

$$\varphi = \arccos z$$

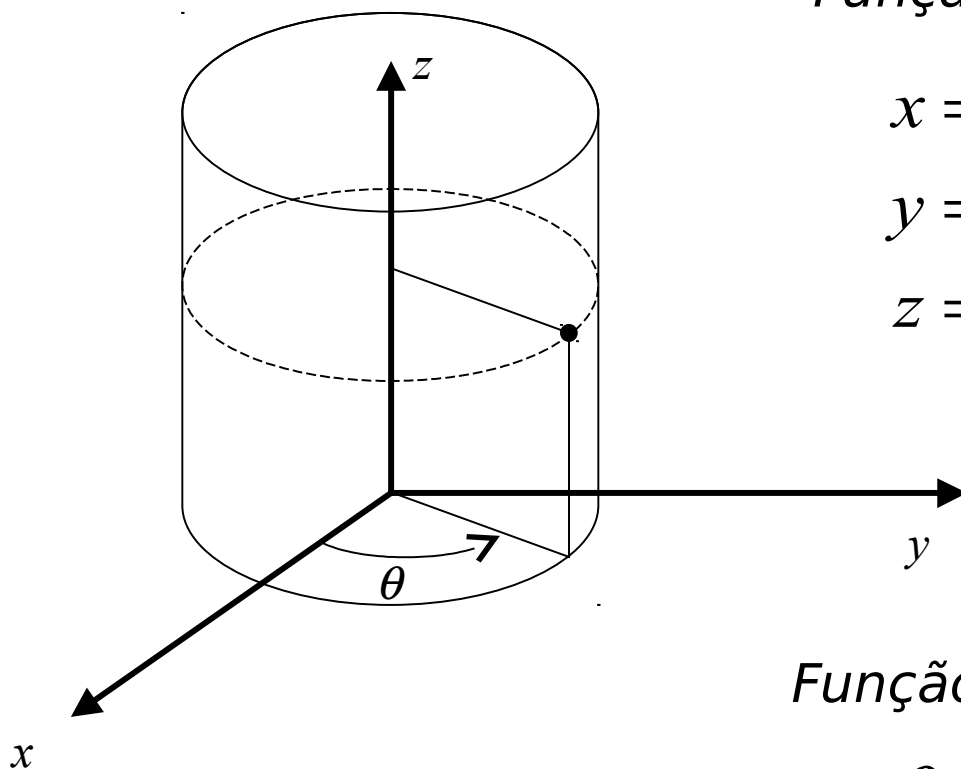
$$t = \frac{\arccos z}{\pi}$$

$$\theta = \arctan \frac{y}{x}$$

$$s = \frac{\arctan \frac{y}{x}}{2\pi}$$

# Parametrização do Cilindro

*Função de mapeamento*



$$x = \cos \theta$$

$$\theta = 2\pi \cdot s$$

$$y = \sin \theta$$

$$z = t$$

$$z = z$$

*Função de mapeamento inversa*

$$\theta = \arctan \frac{y}{x}$$

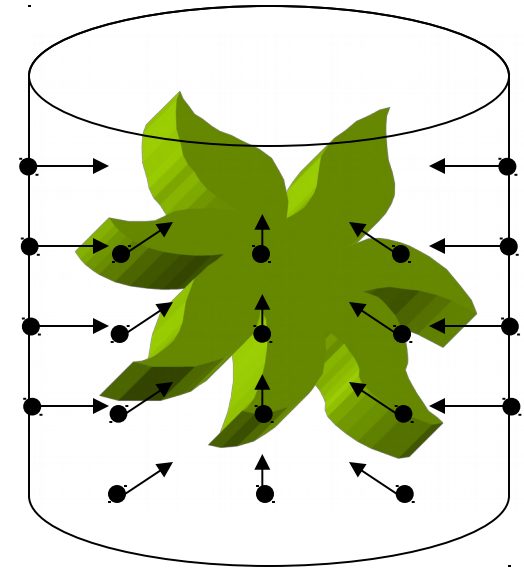
$$s = \frac{\theta}{2\pi}$$

$$z = z$$

$$t = z$$

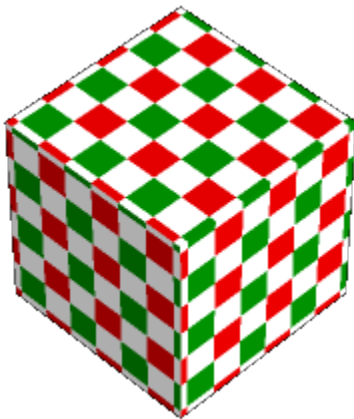
# Parametrizando Objetos Genéricos

- O que fazer quando o objeto não comporta uma parametrização natural?
- Uma sugestão é usar um mapeamento em 2 estágios [Bier e Sloan]:
  - Mapear textura sobre uma superfície simples como cilindro, esfera, etc aproximadamente englobando o objeto
  - Mapear superfície simples sobre a superfície do objeto. Pode ser feito de diversas maneiras
    - Raios passando pelo centróide do objeto
    - Raios normais à superfície do objeto
    - Raios normais à superfície simples
    - Raios refletidos (*environment mapping*)

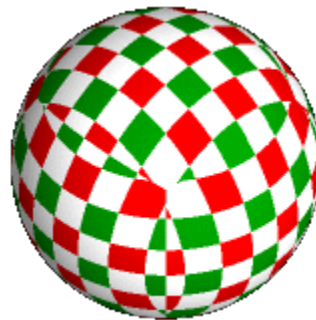


# Exemplos

Parametrização  
cúbica



Projetada em  
uma esfera



Projetada em  
um cilindro





# Exemplos

Parametrização  
cilíndrica



Projetada em  
uma esfera

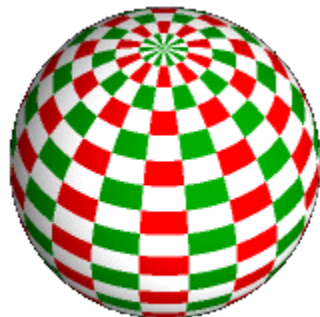


Projetada em  
um cubo

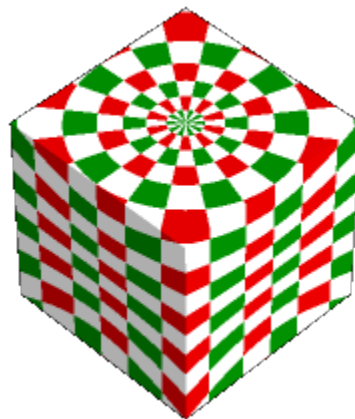


# Exemplos

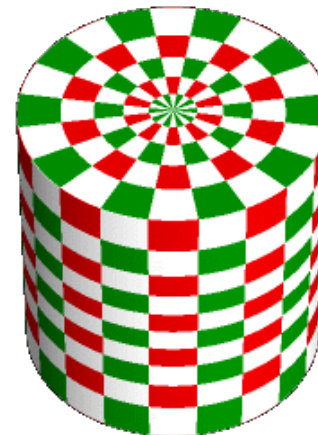
Parametrização  
esférica



Projetada em  
um cubo

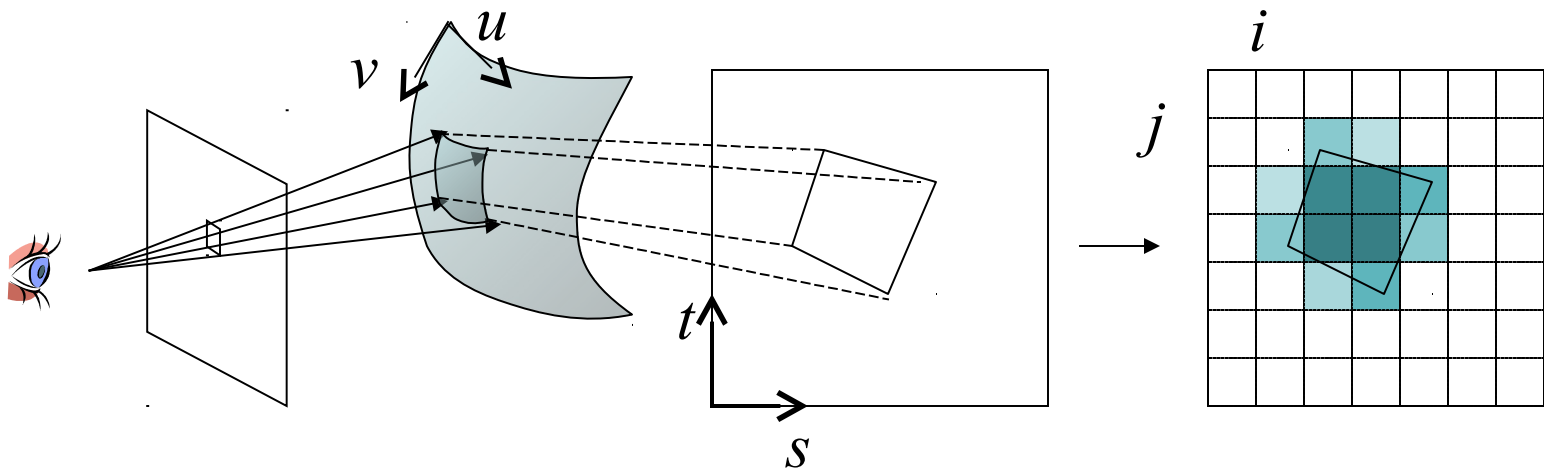


Projetada em  
um cilindro



# Processo de Mapeamento de Texturas

- Projeção do pixel sobre a superfície
  - Pontos da superfície correspondentes aos vértices do pixel
- Parametrização
  - Coordenadas paramétricas dos vértices do pixel projetados
- Mapeamento inverso
  - Coordenadas dos vértices no espaço de textura
- Média
  - Cor média dos ‘Texels’ proporcional à área coberta pelo quadrilátero



# Mapeamento de Texturas em Polígonos

- Polígonos são freqüentemente usados para representar fronteiras de objetos
- Em OpenGL, além das coordenadas dos vértices e do vetor normal, é possível também especificar coordenadas de textura:

```
glBegin (GL_POLYGON) ;  
    glNormal3fv (N) ;  
    glTexCoord2fv (T) ;  
    glVertex3fv (V) ;  
    ...  
glEnd () ;
```

# Mapeamento de Texturas em Polígonos

- A maneira mais simples e rápida:
  - Projetar os vértices do polígono na imagem
  - A cada vértice projetado  $P_i$  corresponde um ponto  $Q_i$  no espaço de textura
  - Um pixel  $P$  do polígono na imagem é dado por uma combinação afim. Ex.:

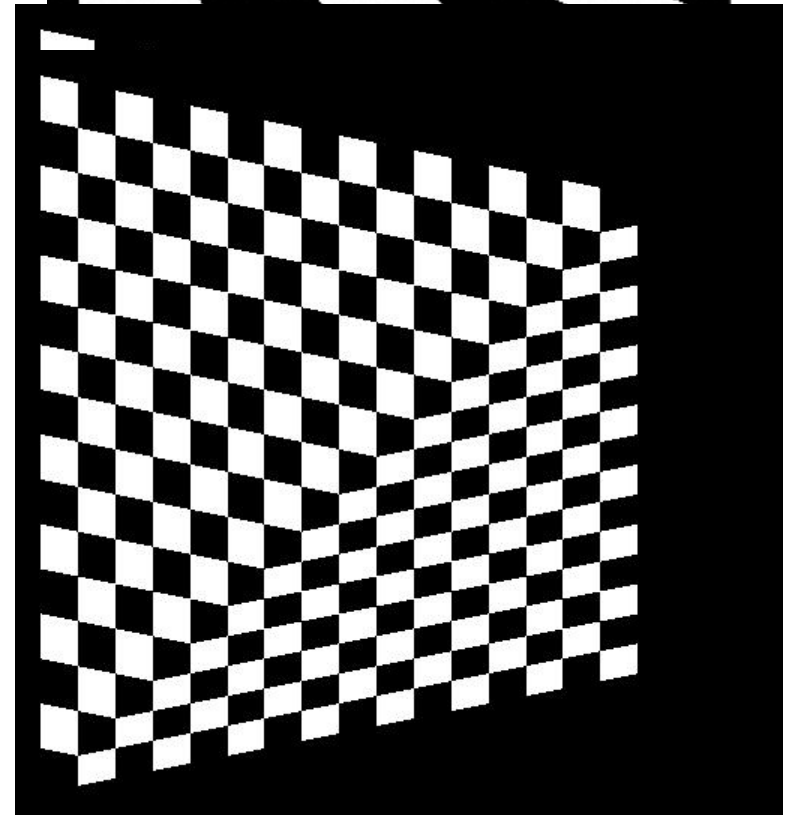
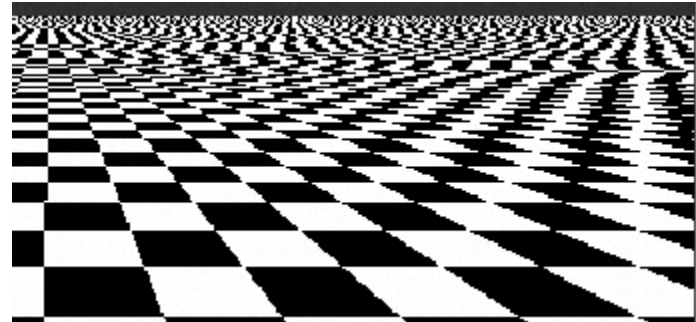
$$P = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3$$

- Pixel  $P$  é pintado com a cor do texel obtido com a mesma combinação afim. Ex.:

$$Q = \alpha_1 Q_1 + \alpha_2 Q_2 + \alpha_3 Q_3$$

# Mapeamento de Texturas em Polígonos

- Problemas da abordagem simples:
  - Aliasing
    - Pixel  $\neq$  Texel
    - Soluções:
      - Interpolação
      - Mip-mapping
  - Deformação
    - Combinações afim não são preservadas em projeções perspectivas
    - Soluções:
      - Mais vértices
      - Coordenadas homogêneas



# Mapeamento de Texturas em OpenGL

## 1. Ligar o mapeamento de texturas

- ♦ **`glEnable(GL_TEXTURE_2D);`**

## 2. Especificar a textura

- ♦ Usar **`glTexImage2D`** que tem o formato

`void glTexImage2D (GLenum target, GLint level, GLint  
internalFormat, GLsizei width, GLsizei height, GLint border,  
GLenum format, GLenum type, const GLvoid *pixels);`

- ♦ **Exemplo:**

`glTexImage2D (GL_TEXTURE_2D, 0, GL_RGBA, 128, 128, 0,  
GL_RGBA, GL_UNSIGNED_BYTE, img);`

# Mapeamento de Texturas em OpenGL

## 3. Configurar diversos parâmetros

- ◆ Modos de filtragem
  - Magnificação ou minificação
  - Filtros mipmap de minificação
- ◆ Modos de repetição de padrões
  - Cortar ou repetir
- ◆ Funções de aplicação de textura
  - Como misturar a cor do objeto com a da textura
    - Misturar, modular ou substituir texels

## 4. Especificar coordenadas de textura

- ◆ Por vértice
  - `glTexCoord*`
- ◆ Coordenadas computadas automaticamente
  - `glTexGen*`



# Especificando imagem de textura

- Imagem de textura normalmente carregada a partir de um array de texels na memória principal
  - `glTexImage2D( target, level, components, w, h, border, format, type, *texels );`
  - *antigamente: tamanho da imagem em potência de 2*
- Cores dos texels são processadas pela parte do pipeline que processa pixels
  - Boa parte do repertório de operações sobre bitmaps pode ser usada

# Outros métodos para especificar texturas

- Usar o frame buffer como fonte da imagem de textura

**glCopyTexImage1D(...)**

**glCopyTexImage2D(...)**

- Modificar parte de uma textura pré-definida

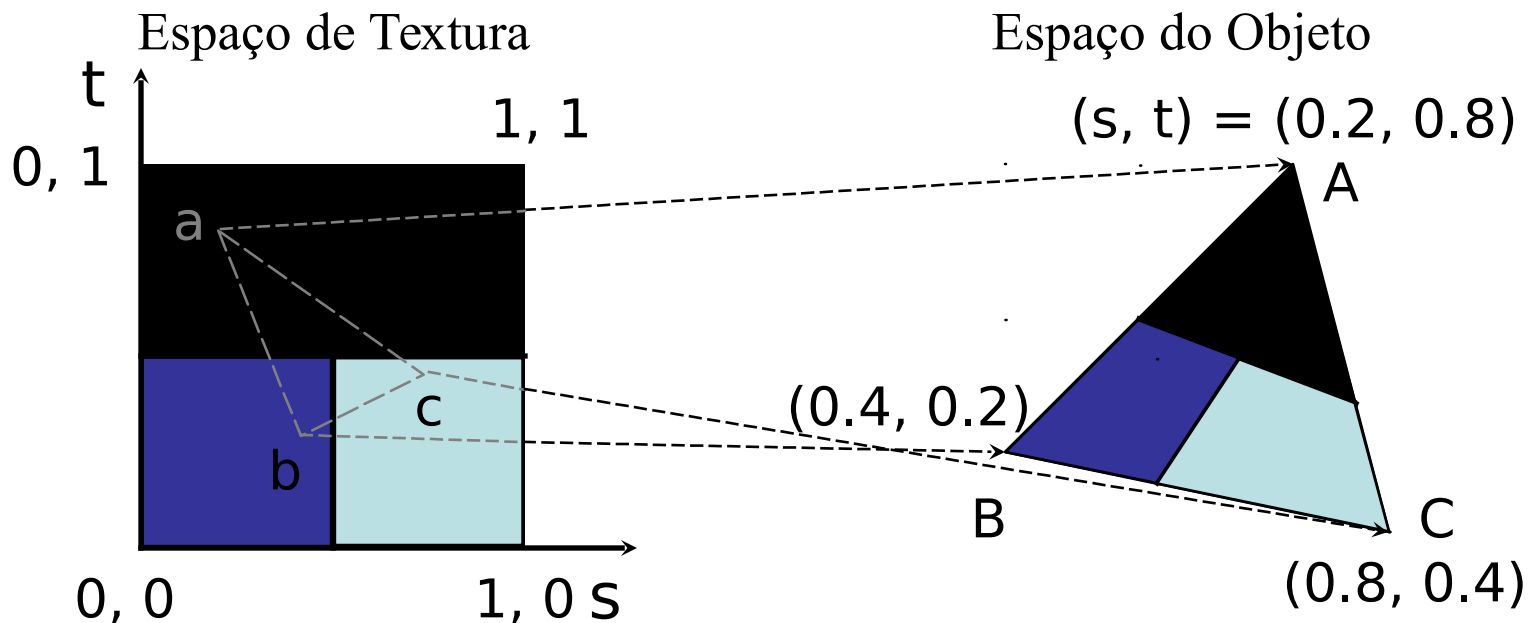
**glTexSubImage1D(...)**

**glTexSubImage2D(...)**

**glTexSubImage3D(...)**

# Mapeando a Textura

- Baseado em coordenadas paramétricas de textura
- Chamar `glTexCoord*()` para cada vértice



# Gerando Coordenadas de Texturas Automaticamente

- Habilitar a geração automática de coordenadas de textura

```
glEnable (GL_TEXTURE_GEN_{STRQ}) ;
```

- Especificar parâmetros

```
void glTexGen{ifd} (GLenum coord, GLenum pname, TYPE param);
```

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

- ♦ Qual coordenada de textura?

- *Coord* = GL\_S / GL\_T / GL\_R / GL\_Q

- ♦ Plano de referência

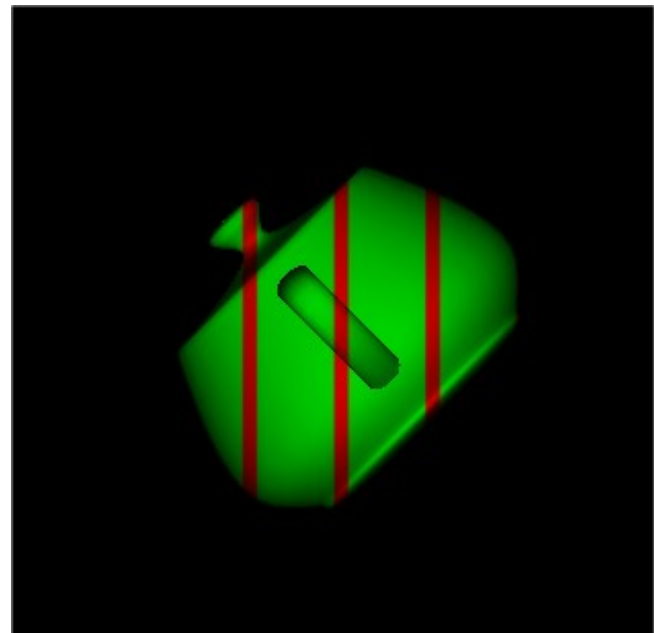
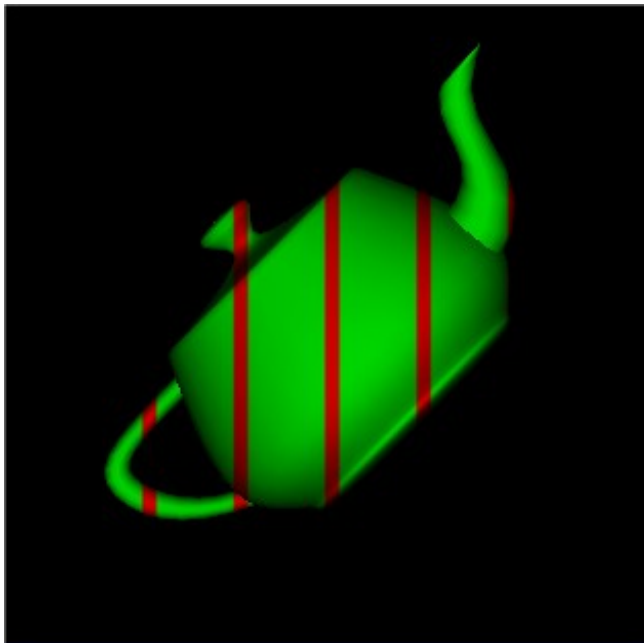
- *Pname* = GL\_OBJECT\_PLANE / GL\_EYE\_PLANE
- *Param* = coeficientes A/B/C/D do plano

- ♦ Modos de geração de coordenadas

- *Pname* = GL\_TEXTURE\_GEN\_MODE
- *Param* = GL\_OBJECT\_LINEAR / GL\_EYE\_LINEAR / GL\_SPHERE\_MAP

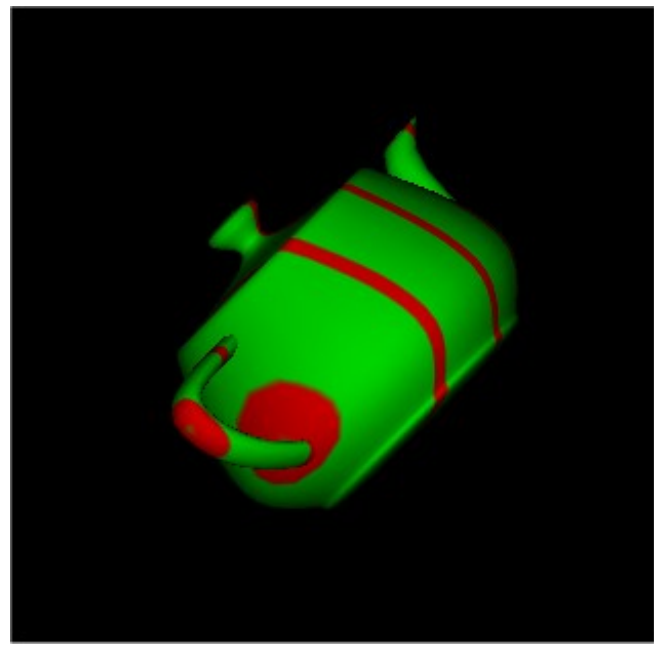
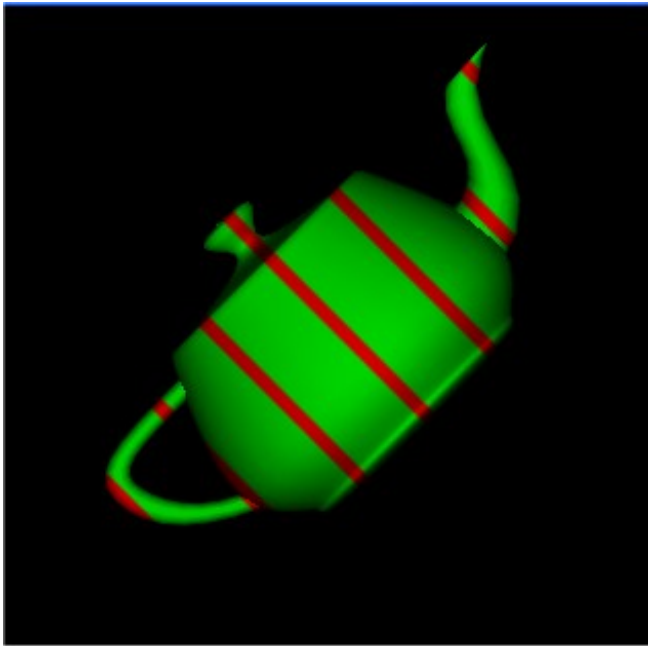
# Geração Automática de Coordenadas de Textura

GL\_EYE\_LINEAR



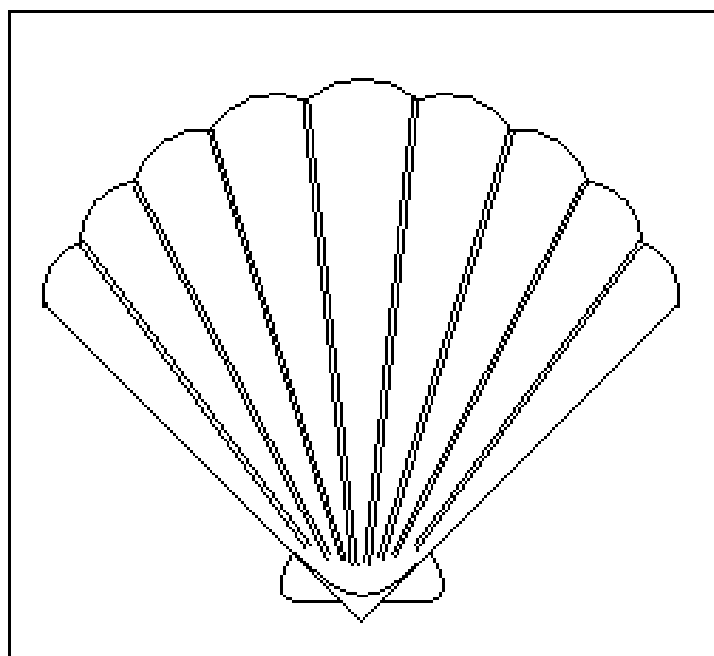
# Geração Automática de Coordenadas de Textura

GL\_OBJECT\_LINEAR



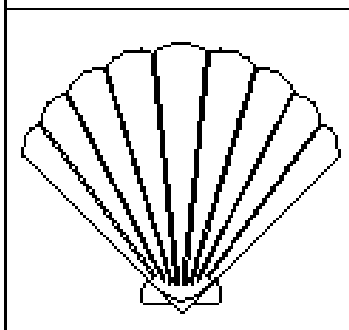
# Texturas Mipmap

Textura original

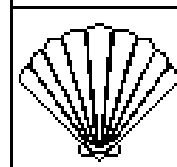


Imagens minificadas  
pré-filtradas

1/4



1/16



1/64



etc.



1 pixel

# Texturas Mipmap

- Permite que texturas de diferentes níveis de resolução sejam aplicadas de forma adaptativa
- Reduz aliasing devido a problemas de interpolação
- O nível da textura na hierarquia mipmap é especificada durante a definição da textura

**`glTexImage*D( GL_TEXTURE_*D, level, ... )`**

- GLU possui rotinas auxiliares para construir texturas mipmap com filtragem adequada

**`gluBuildDMipmaps( ... )`**

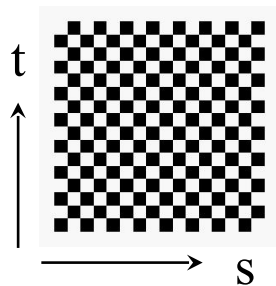
- OpenGL 1.2 suporta facilidades mais sofisticadas para níveis de detalhe (LOD)



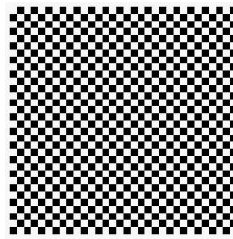
# Modos de Repetição

- Exemplo:

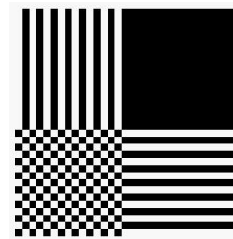
```
glTexParameteri( GL_TEXTURE_2D,  
                 GL_TEXTURE_WRAP_S, GL_CLAMP )  
glTexParameteri( GL_TEXTURE_2D,  
                 GL_TEXTURE_WRAP_T, GL_REPEAT )
```



textura



GL\_REPEAT



GL\_CLAMP

# Modos de Aplicação de Textura

- Controla como a cor da textura afeta a cor do pixel

```
glTexEnv{fi}[v](GL_TEXTURE_ENV, prop, param )
```

- Modos (*prop* = *TEXTURE\_ENV\_MODE*)
  - GL\_MODULATE
  - GL\_BLEND
  - GL\_REPLACE
- Cor a ser misturada (GL\_BLEND)
  - Especificada com *prop* = *GL\_TEXTURE\_ENV\_COLOR*

# Correção Perspectiva

- Mapeamento de texturas em polígonos pode ser feito:
  - Da forma simples e rápida (interpolação linear)
  - Usando interpolação em coordenadas homogêneas
- Comportamento do OpenGL é influenciado por “dicas” (“*hints*”)

```
glHint( GL_PERSPECTIVE_CORRECTION_HINT, hint )
```

onde *hint* pode ser

- GL\_DONT\_CARE
  - GL\_NICEST
  - GL\_FASTEST
- O OpenGL não necessariamente obedece!

# Outras Facilidades

- Objetos de Textura (Texture Objects)
  - Permite mudar rapidamente de texturas durante a renderização de diversos objetos
- Controle de espaço na memória de texturas
  - Texturas residentes na placa são mais rápidas
- Multitexturas (Extensões OpenGL)
  - Placas + modernas (NVidia GeForce /ATI Radeon)
  - Mais de uma textura mapeada no mesmo objeto
  - Permite uma série de efeitos interessantes
    - Shadow mapping
    - Bump mapping