



Introdução à Computação Gráfica

Marcel P. Jackowski

mjack@ime.usp.br

Aula #11



Objetivos

- Consolidar conhecimentos
 - Mapeamentos de textura
 - Mapeamentos de rugosidade
 - Mapeamentos de deslocamento
- Alguns detalhes de mapeamentos em OpenGL

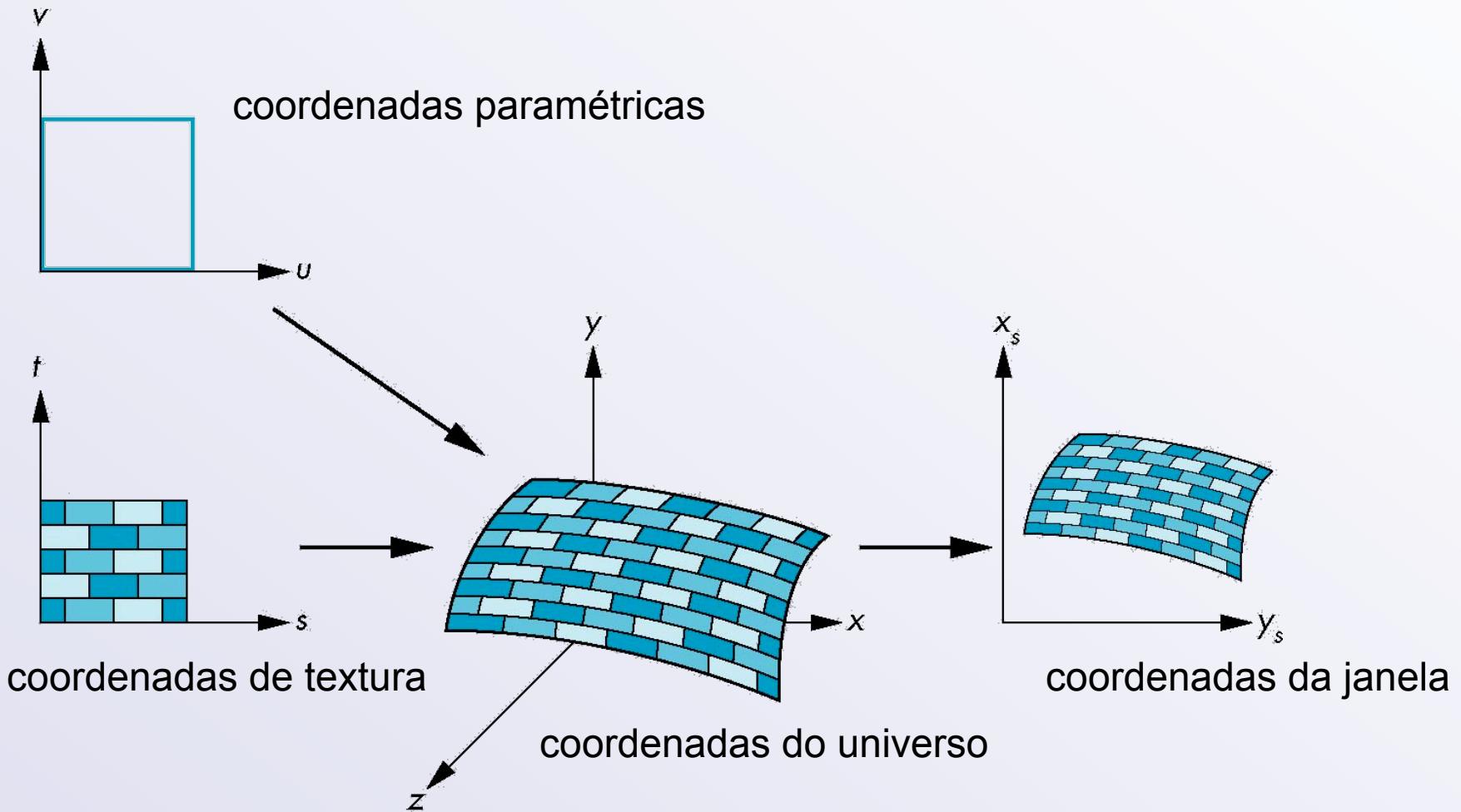
Mapeamento de textura

- Modelos de iluminação não são apropriados para descrever todas as diferenças de cor observáveis em uma superfície
 - Superfícies pintadas com padrões ou imagens
 - Superfícies com padrões de rugosidade
- 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

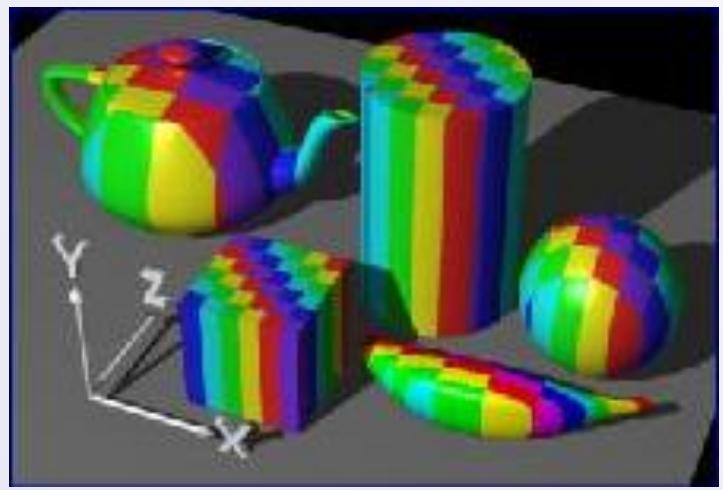
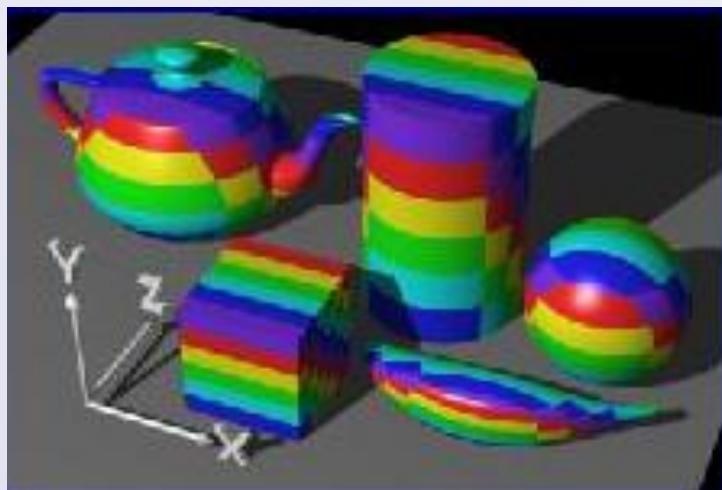
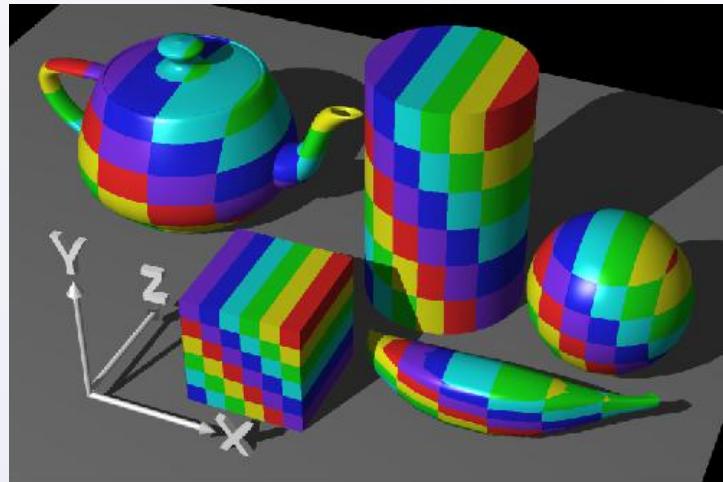
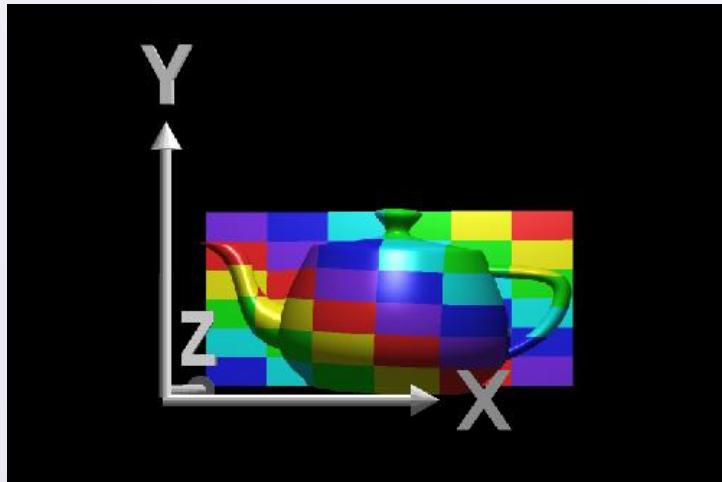
Tipos de textura

- Texturas podem ter componentes RGB ou RGBA ou simplesmente luminosidade (L).
- Podem ter diferentes dimensões:
 - 1D: Linhas
 - 2D: Imagens
 - 3D: Volumes (imagens volumétricas)
- Em versões < 2.0 de OpenGL, os tamanhos de texturas deveriam ser potência de 2.

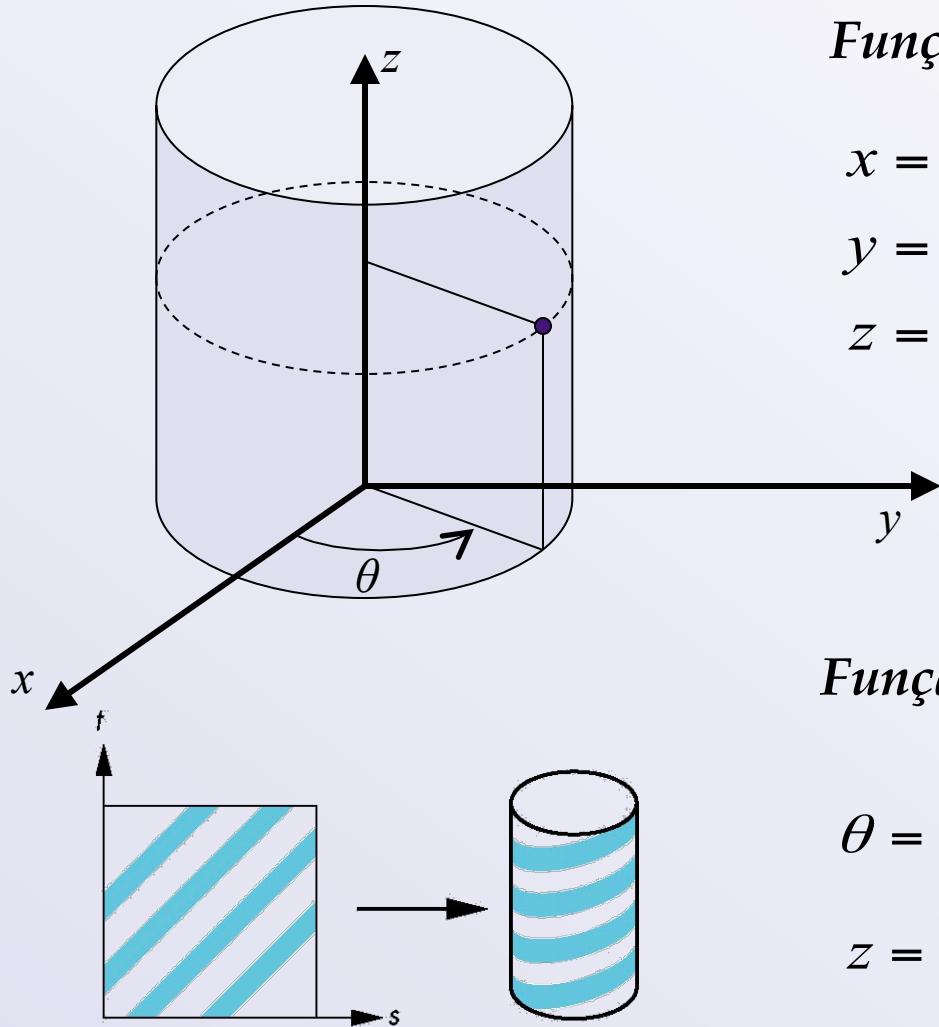
Mapeamento de textura



Mapeamento planar



Parametrização Cilíndrica



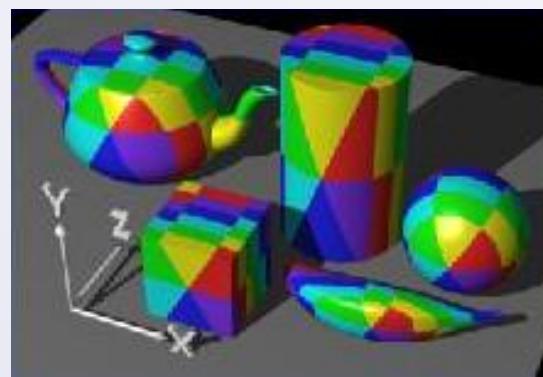
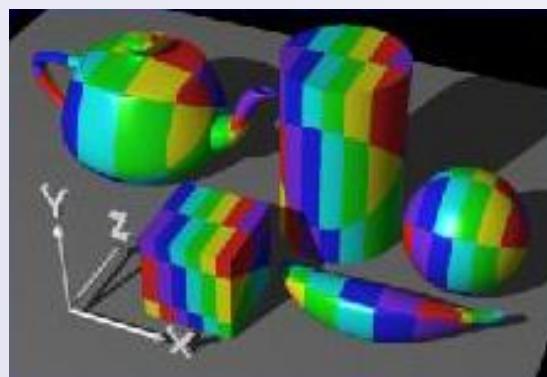
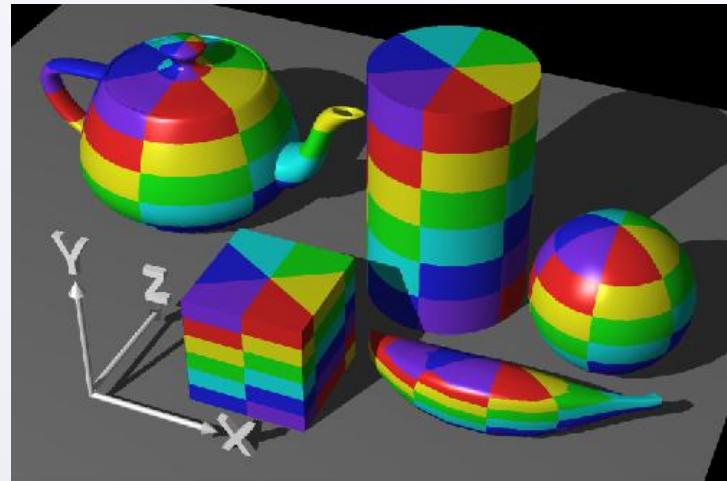
Função de mapeamento

$$\begin{aligned}x &= \cos \theta & \theta &= 2\pi \cdot s \\y &= \sin \theta & z &= t \\z &= z\end{aligned}$$

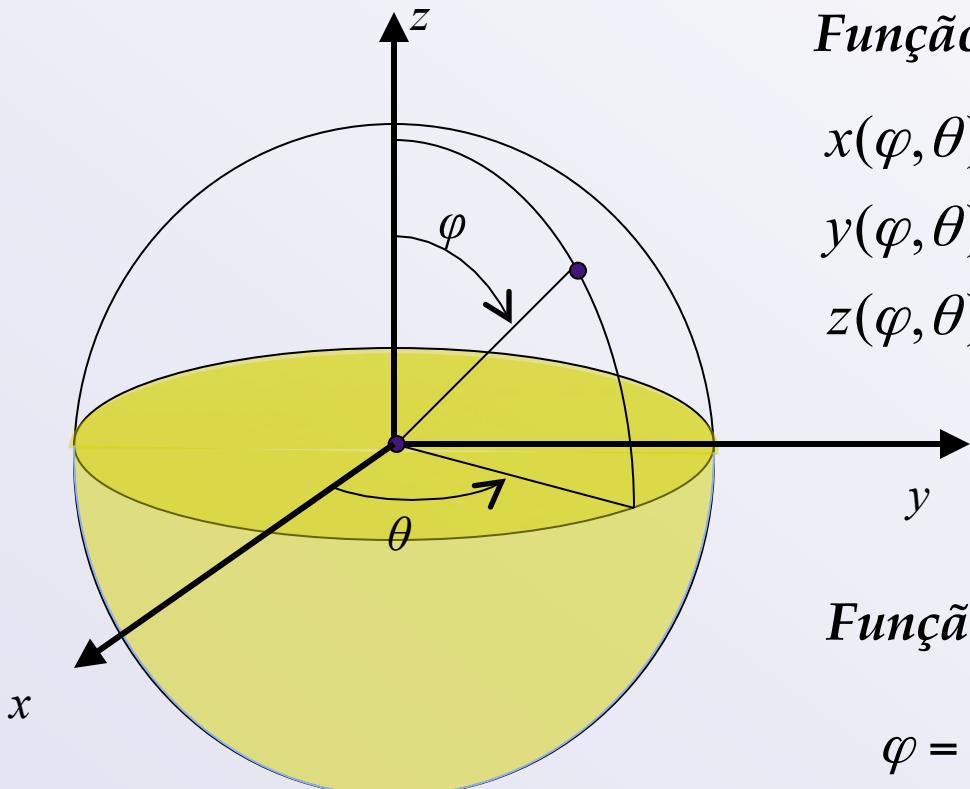
Função de mapeamento inversa

$$\begin{aligned}\theta &= \arctan \frac{y}{x} & s &= \frac{\theta}{2\pi} \\z &= z & t &= z\end{aligned}$$

Mapeamento cilíndrico



Parametrização Esférica



Função de mapeamento

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

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

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

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

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

Função de mapeamento inversa

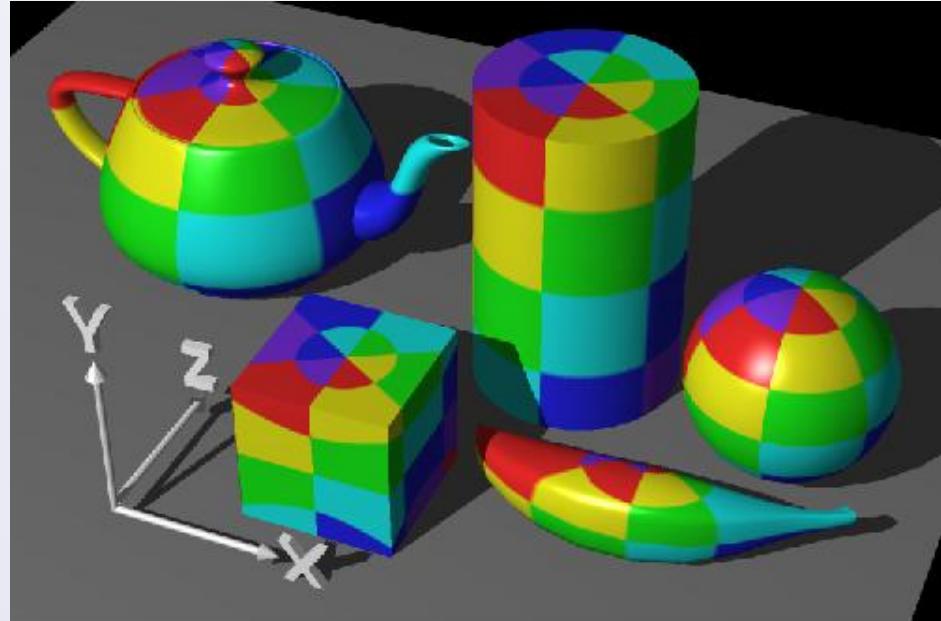
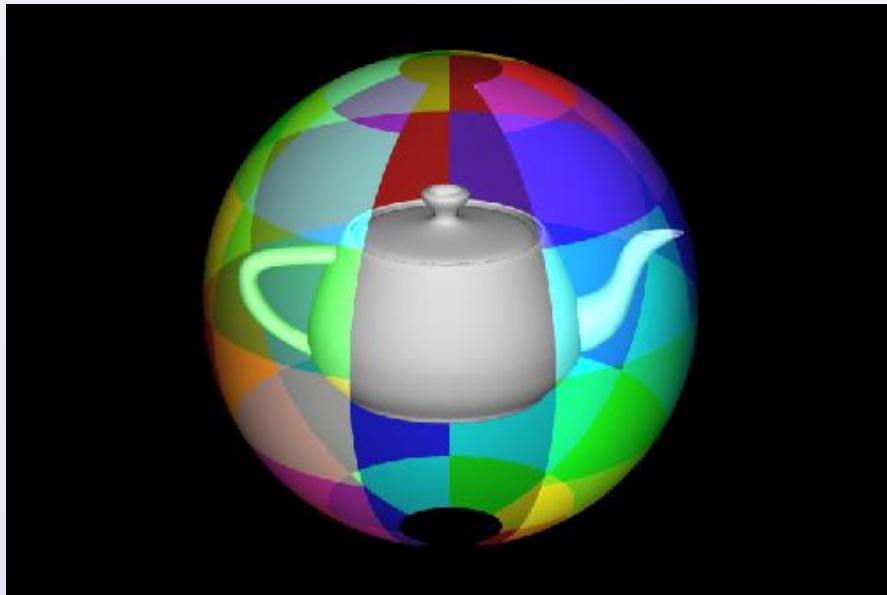
$$\varphi = \arccos z$$

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

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

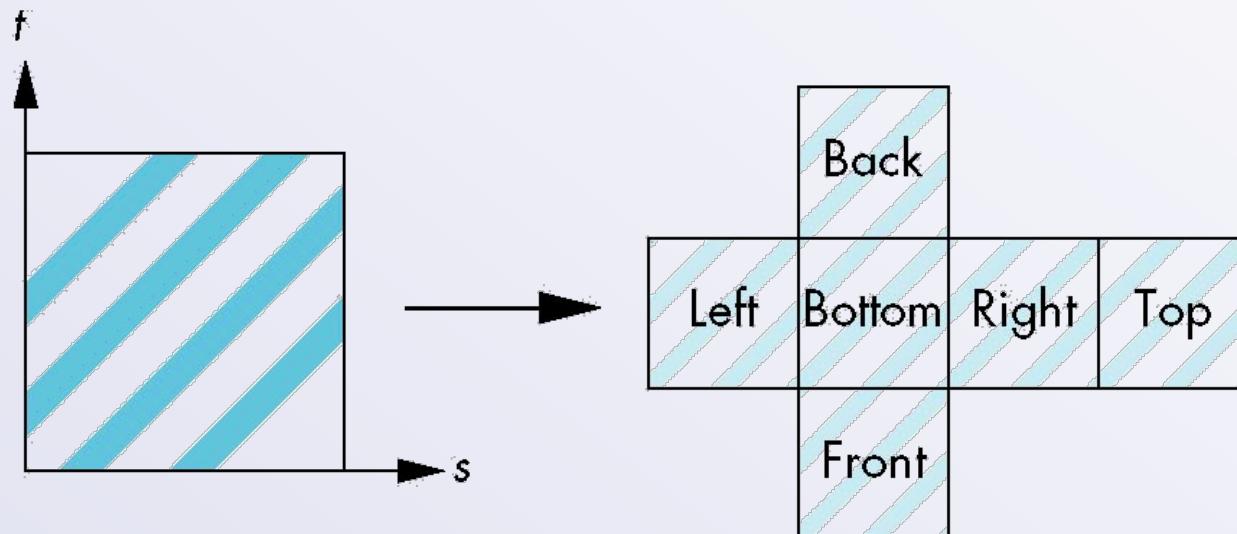
$$s = \frac{\arctan \frac{y}{x}}{2\pi}$$

Mapeamento esférico

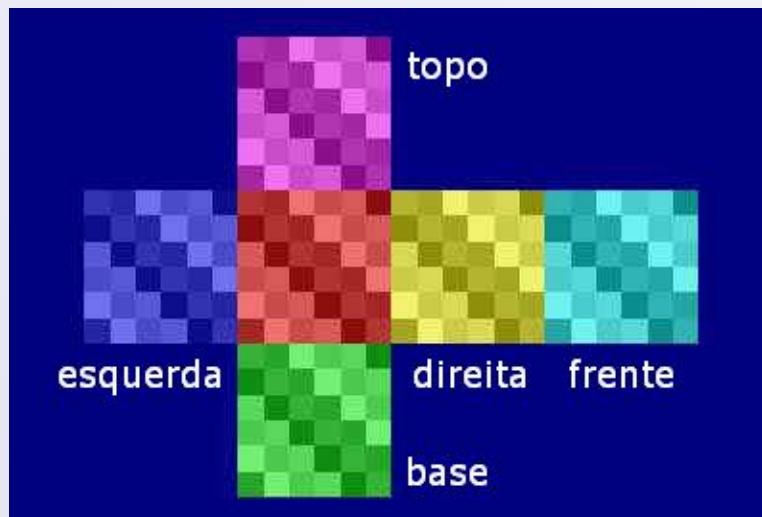
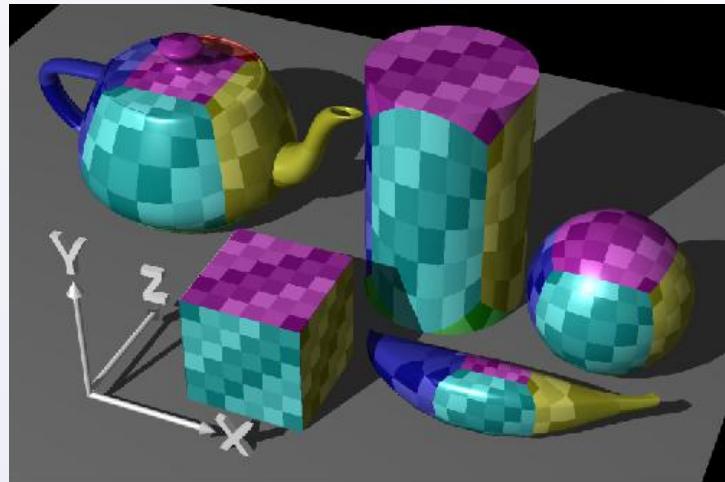
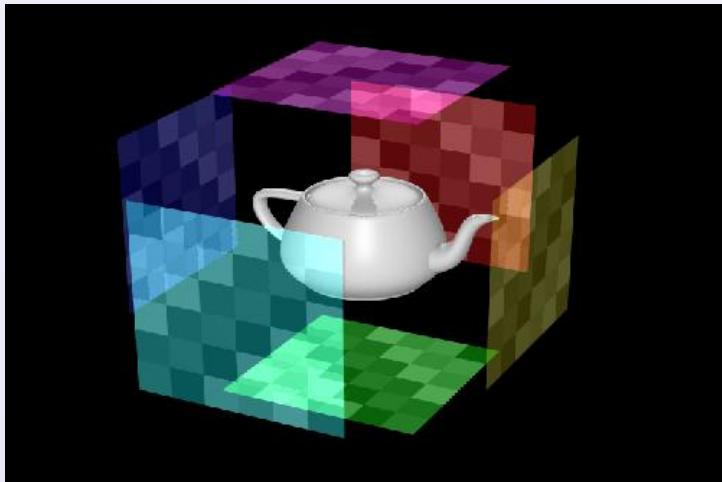


Parametrização Cúbica

- Fácil de usar com projeções ortográficas simples
- Usado em mapeamentos de ambiente

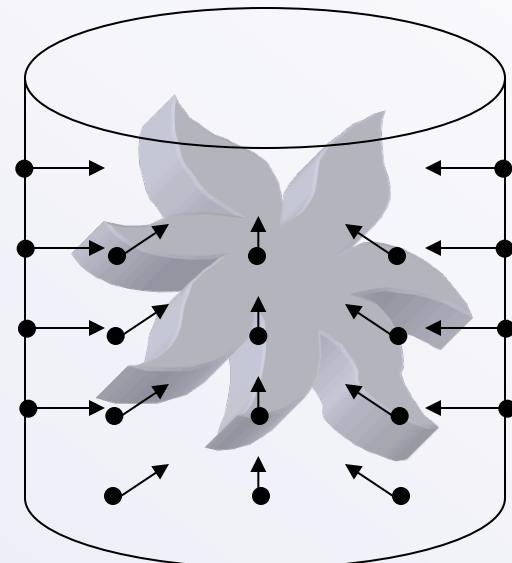


Mapeamento cúbico



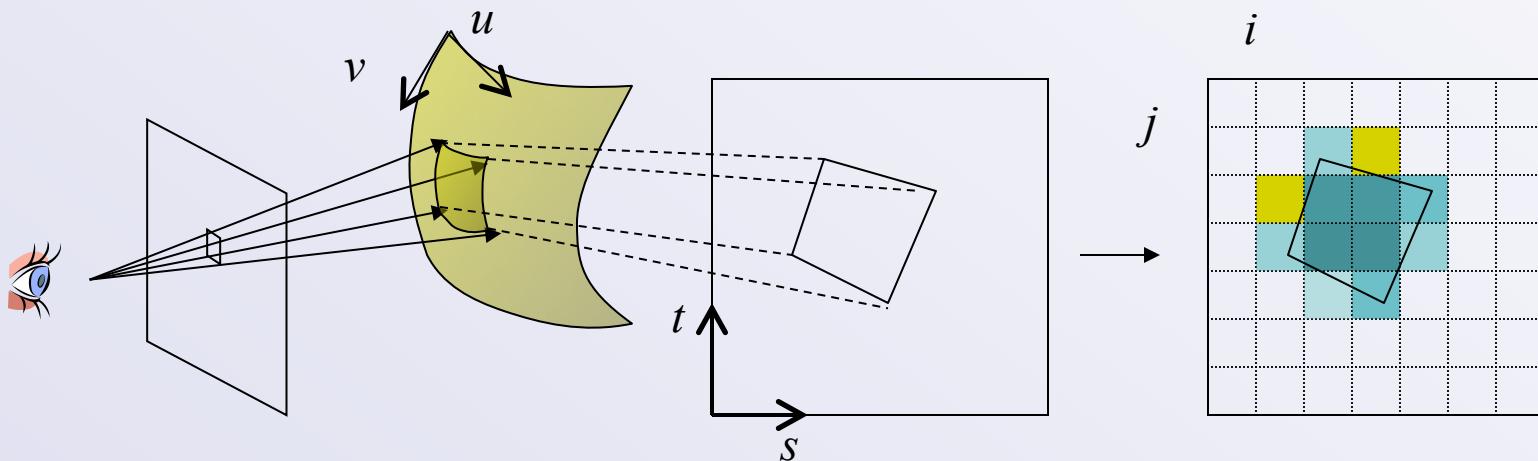
Objetos genéricos

- 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*)



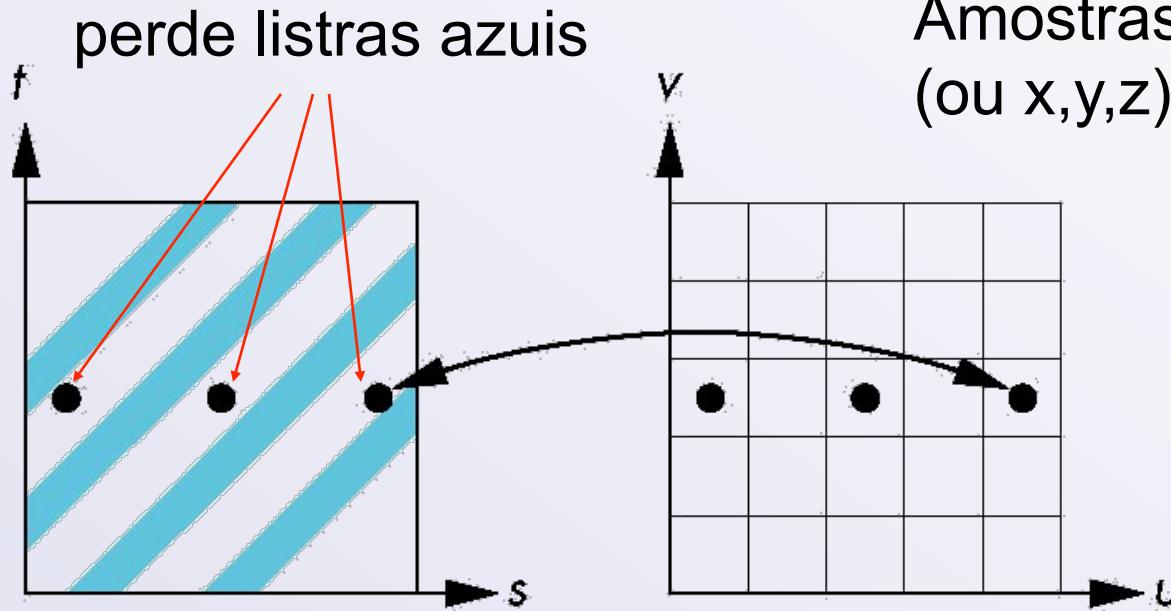
Mapeamento de texturas

- 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



Subamostragem

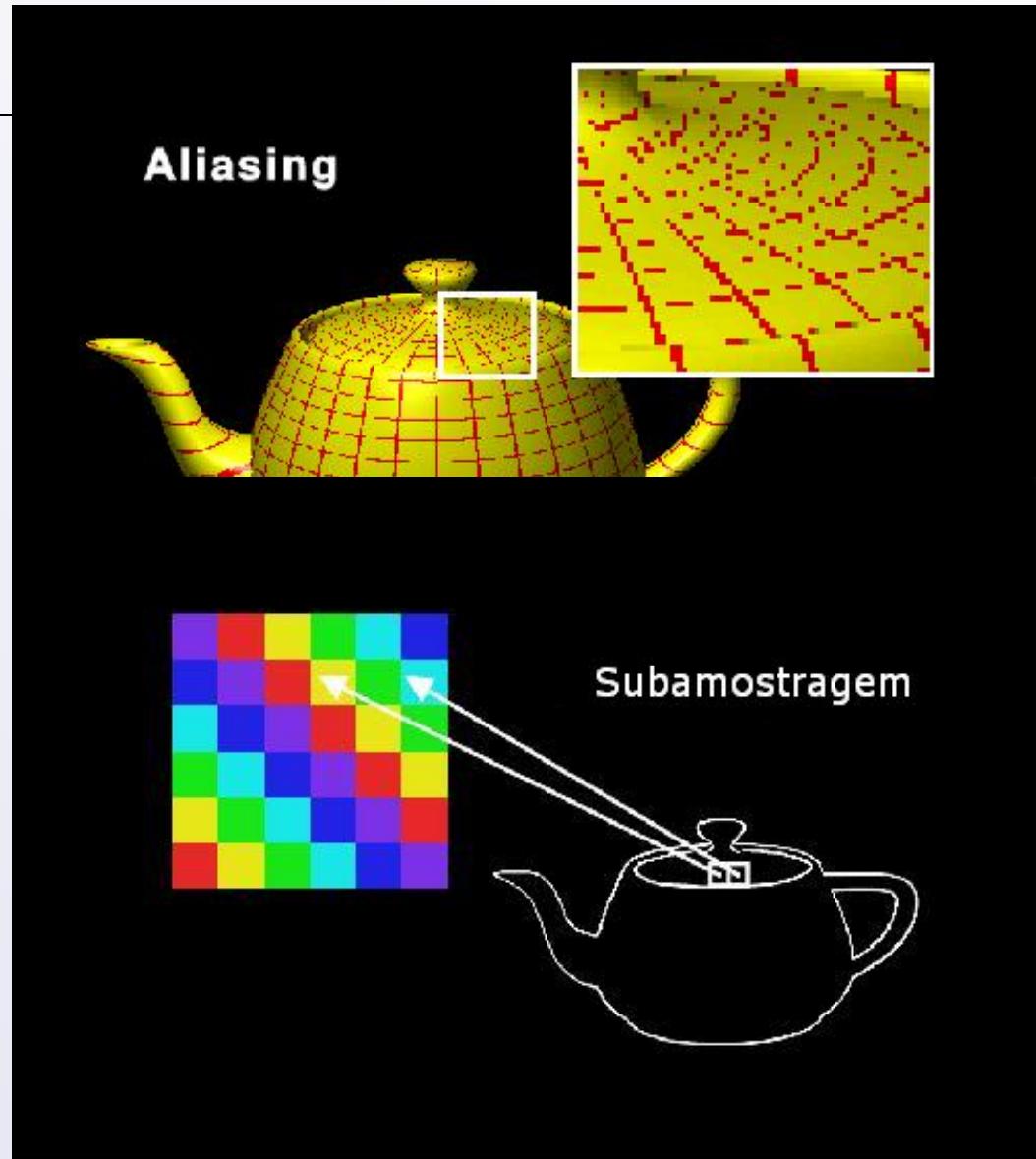
- A amostragem de textura por pontos pode acarretar em erros de **aliasing**



amostras no espaço da textura

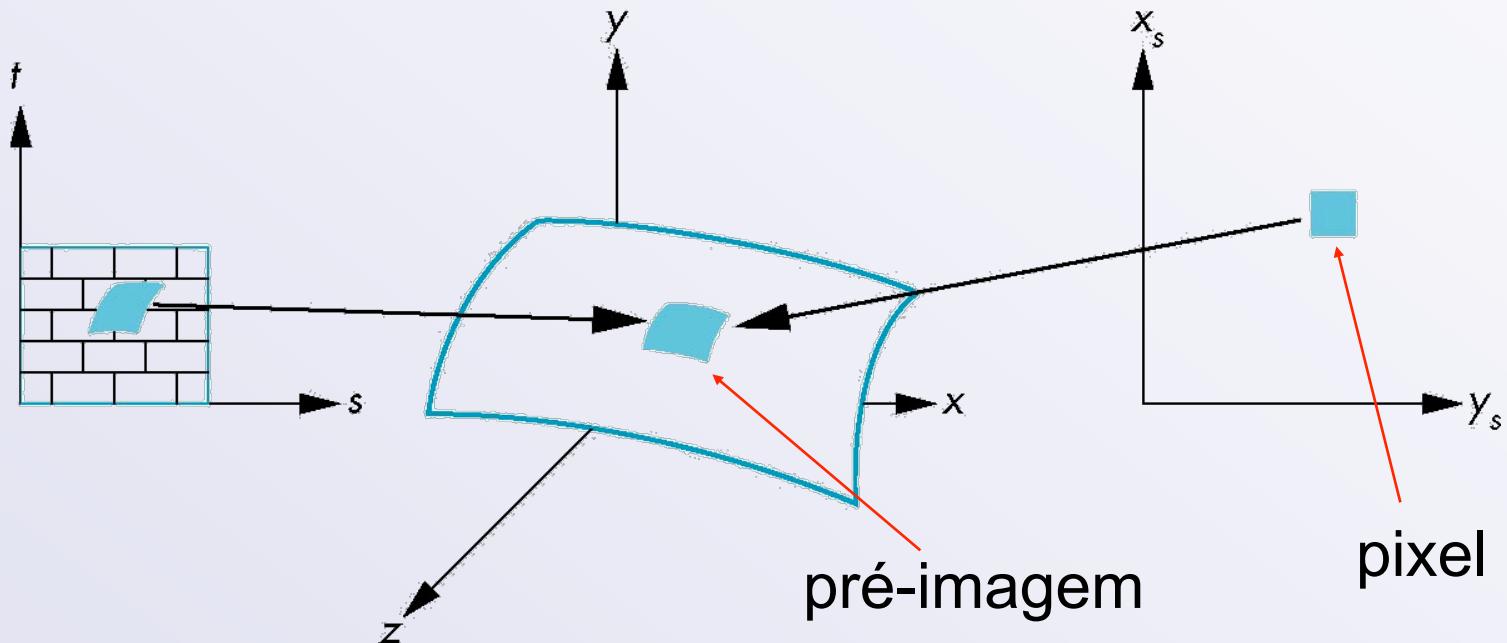
Antialiasing

- Resultado com perda de continuidade (serrilhado)



Área

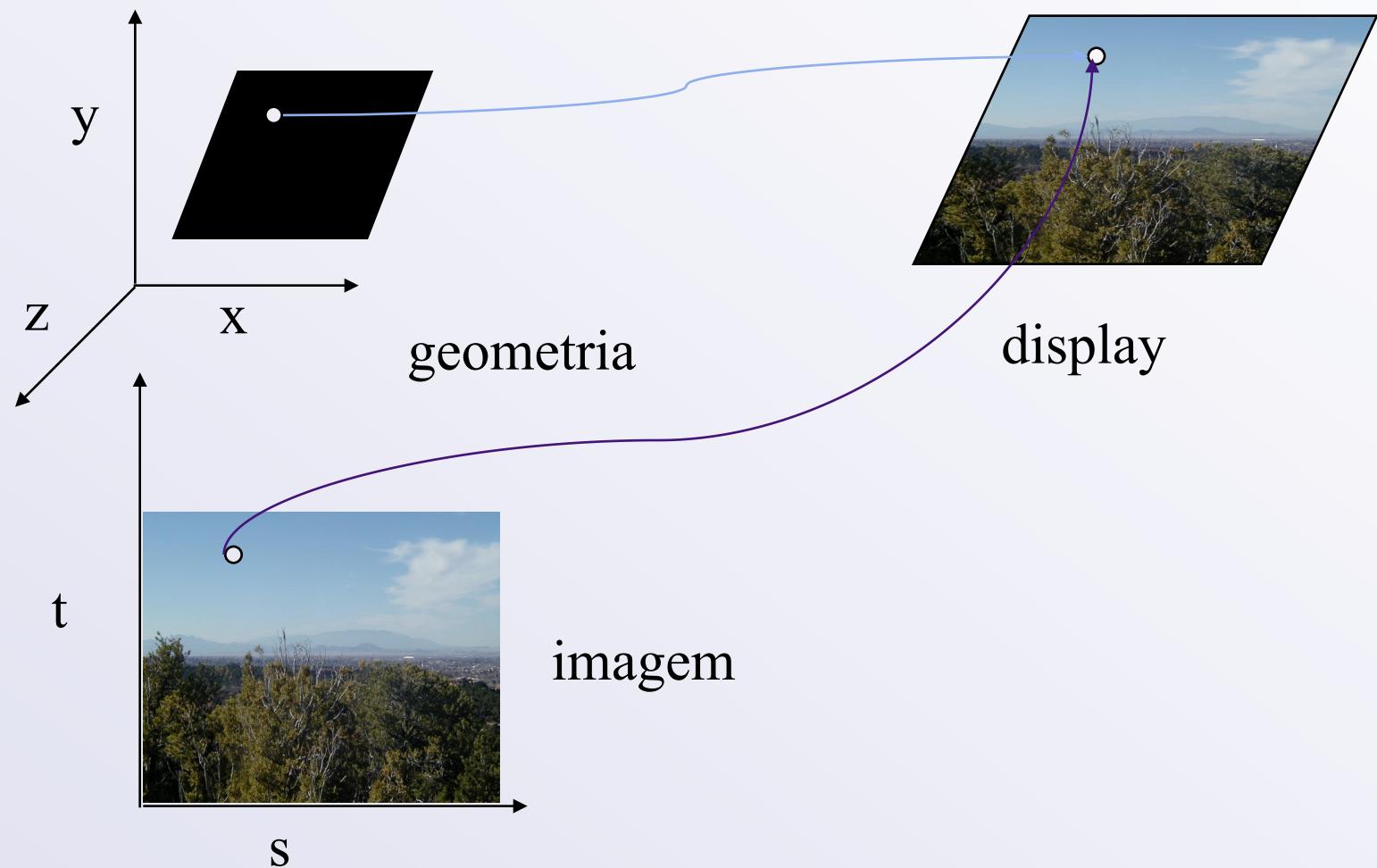
Uma melhor solução, porém menos eficiente, é utilizar uma área de amostragem



Mapeamento em OpenGL

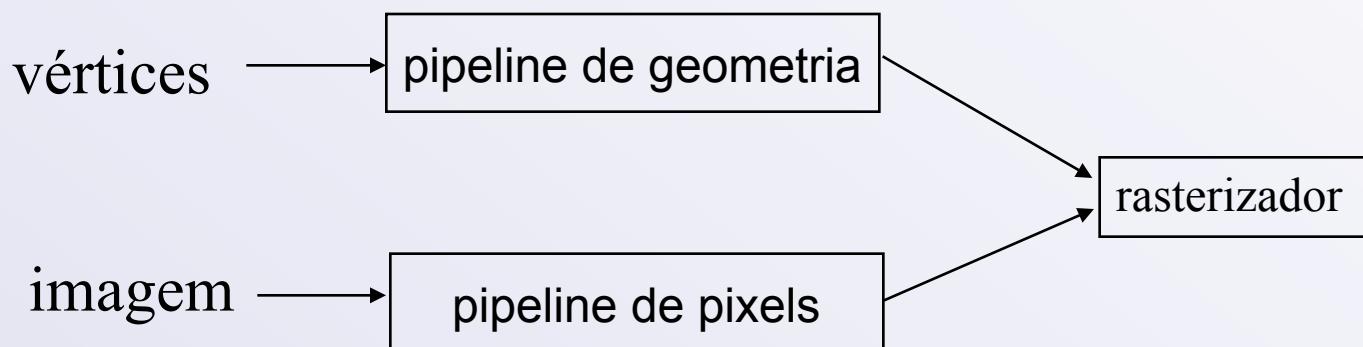
- Três passos principais
 - Especificação da textura
 - Leitura ou geração da imagem
 - Atribuir a uma textura
 - Habilitar textura
- Atribuir coordenadas da textura aos vértices
 - Responsável pela aplicação
- Especificar parâmetros da textura
 - Modo de repetição, filtragem

Mapeamento de textura



Texturas no pipeline OpenGL

- Imagens e geometrias navegam em pipelines separados até chegarem no rasterizador



Especificando uma textura

- Criar uma imagem a ser usada como textura a partir de um array de *texels* (elementos de textura) em memória:

```
Glubyte texels[512][512];
```
- É definida como qualquer outro pixmap
 - Imagem escaneada
 - Geração automática (procedural)
- Habilitar mapeamento de textura
 - `glEnable(GL_TEXTURE_2D)`
 - OpenGL suporta mapas de texturas de 1D-3D

Texturas em OpenGL

1. Habilitar o mapeamento de texturas
 - `glEnable(GL_TEXTURE_2D);`
2. Especificar a textura:
 - `glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA,
128, 128, 0, GL_RGBA, GL_UNSIGNED_BYTE,
img);`

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*

Definindo imagem como textura

```
glTexImage2D( target, level, components,  
              w, h, border, format, type, texels );
```

target: tipo da textura, e.g. `GL_TEXTURE_2D`

level: usado em mipmapping

components: elementos por texel

w, h: largura e comprimento dos `texels` em pixels

border: usado para suavização

format and type: formato e tipo dos dados

texels: ponteiro para o buffer de texels

```
glTexImage2D(GL_TEXTURE_2D, 0, 3, 512, 512, 0,  
             GL_RGB, GL_UNSIGNED_BYTE, texels);
```

Operação de escala

- Exemplo: se o tamanho da imagem não é uma potência de 2
 - `gluScaleImage(format, w_in, h_in,
type_in, *data_in, w_out, h_out,
type_out, *data_out);`
 - $*_{\text{in}} = \text{imagem original}$
 - $*_{\text{out}} = \text{imagem destino}$

Outros métodos

- Usar o frame buffer como fonte da imagem de textura

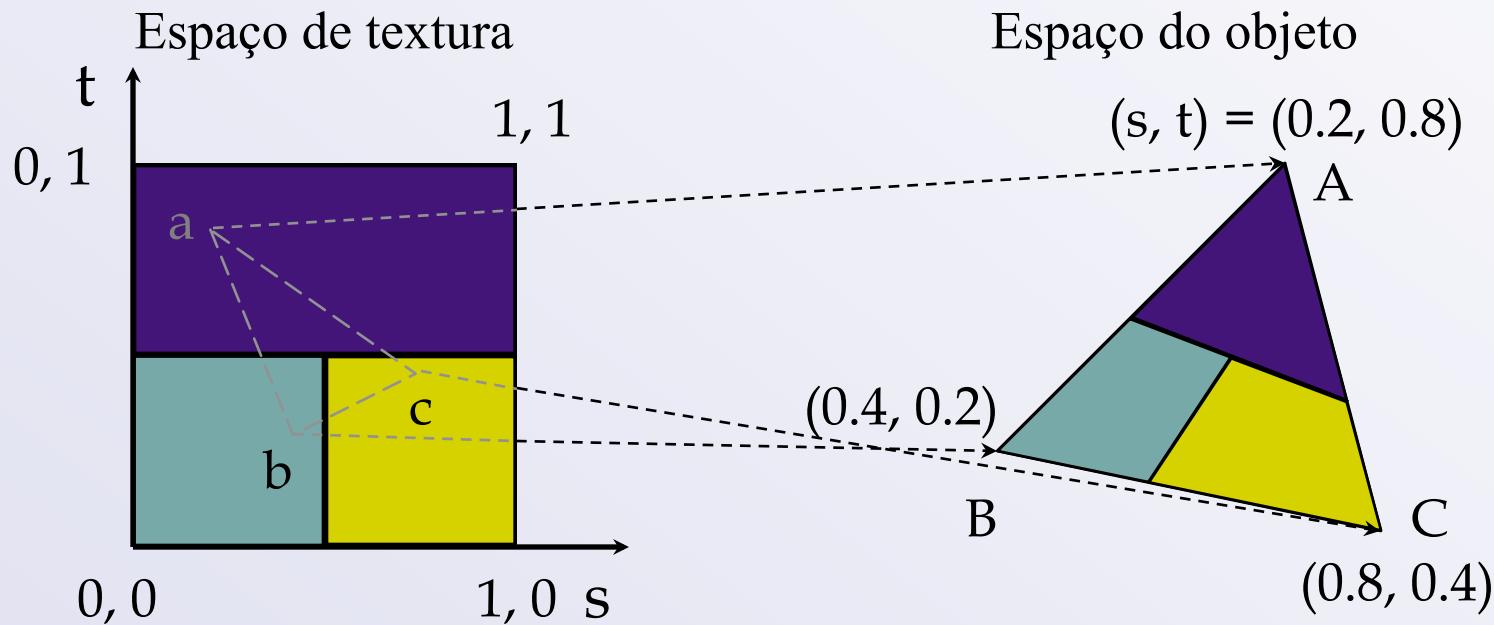
`glCopyTexImage1D(. . .)`
`glCopyTexImage2D(. . .)`

- Modificar parte de uma textura pré-definida

`glTexSubImage1D(. . .)`
`glTexSubImage2D(. . .)`
`glTexSubImage3D(. . .)`

Mapeamento de textura

- Baseada em coordenadas paramétricas
- `glTexCoord*` () especificada em cada vértice



Código em OpenGL

```
glBegin(GL_POLYGON) ;  
    glColor3f(r0, g0, b0) ;  
    glNormal3f(u0, v0, w0) ;  
    glTexCoord2f(s0, t0) ;  
    glVertex3f(x0, y0, z0) ;  
  
    glColor3f(r1, g1, b1) ;  
    glNormal3f(u1, v1, w1) ;  
    glTexCoord2f(s1, t1) ;  
    glVertex3f(x1, y1, z1) ;  
    .  
    .  
glEnd() ;
```

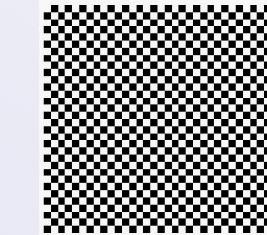
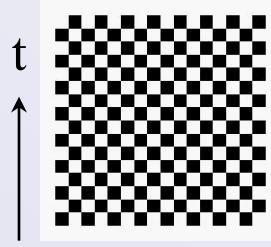
Podemos usar vertex arrays para aumentar a eficiência

Parâmetros de textura

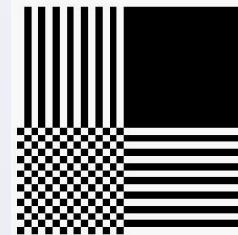
- O OpenGL tem uma série de parâmetros que controlam a aplicação de texturas:
 - Repetição: determina o que acontece quando s e t estão fora da faixa $(0,1)$
 - Filtragem: amostragem por pontos ou área
 - Mipmapping: utilização de texturas em múltiplas resoluções
 - Parâmetros de ambiente: determinam a interação do mapeamento de textura com a tonalização.

Modos de repetição

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



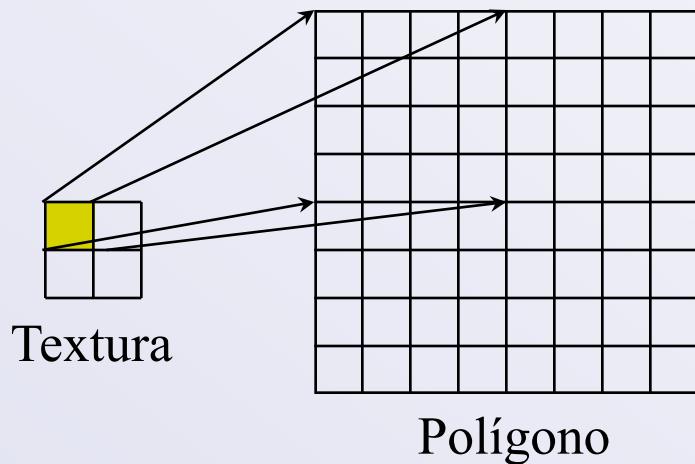
GL_REPEAT



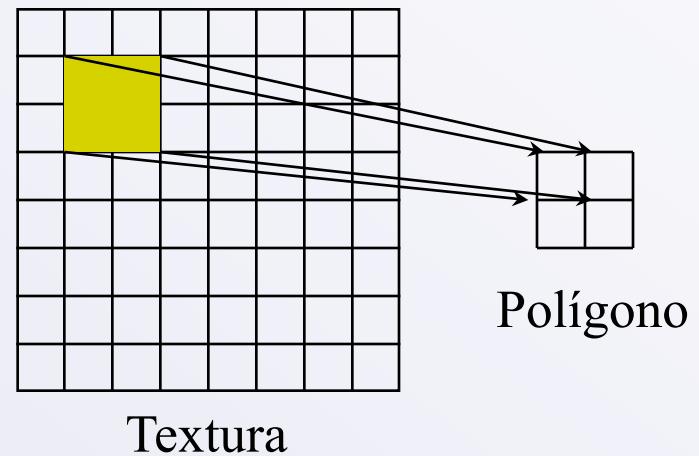
GL_CLAMP

Magnificação ou minificação

Magnificação



Minificação



Filtragem

Modos determinados através de:

- `glTexParameter(target, type, mode)`

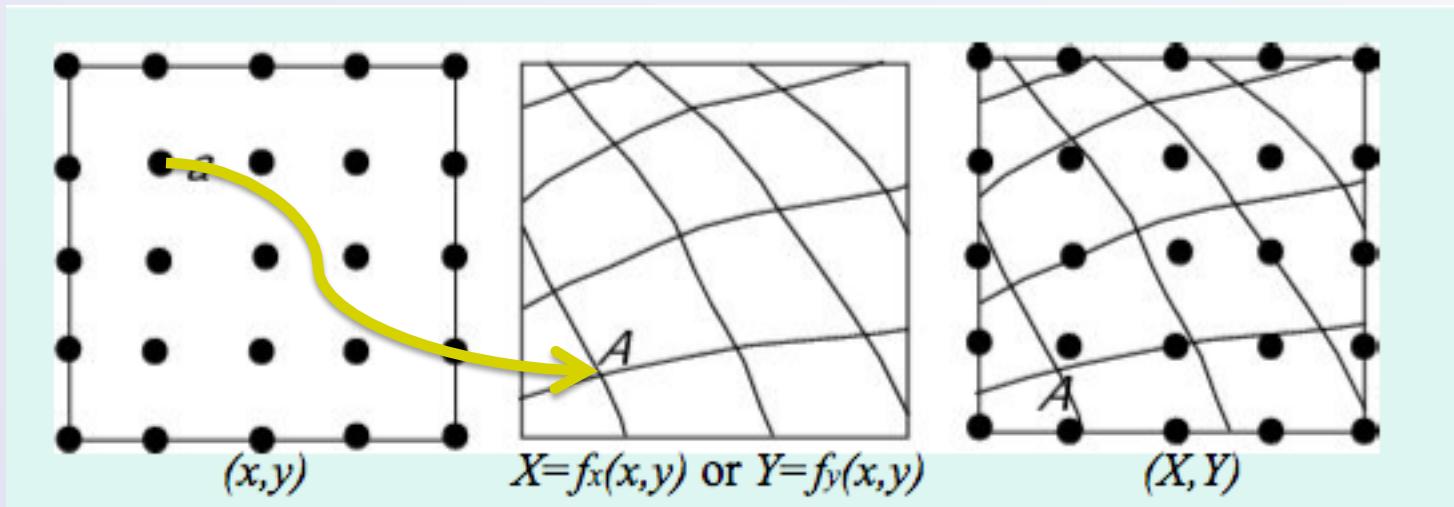
```
glTexParameter(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,  
               GL_NEAREST);
```

```
glTexParameter(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,  
               GL_LINEAR);
```

O uso da filtragem linear requer uma borda de um texel extra para filtragem nas arestas

Reamostragem

- **Problema:** Conhecendo-se as intensidades da imagem em coordenadas discretas, determinar intensidades em coordenadas arbitrárias.
- **Abordagem:** Aproxima-se uma superfície utilizando as intensidades das coordenadas discretas e estima-se o valor da superfície nas coordenadas desejadas.

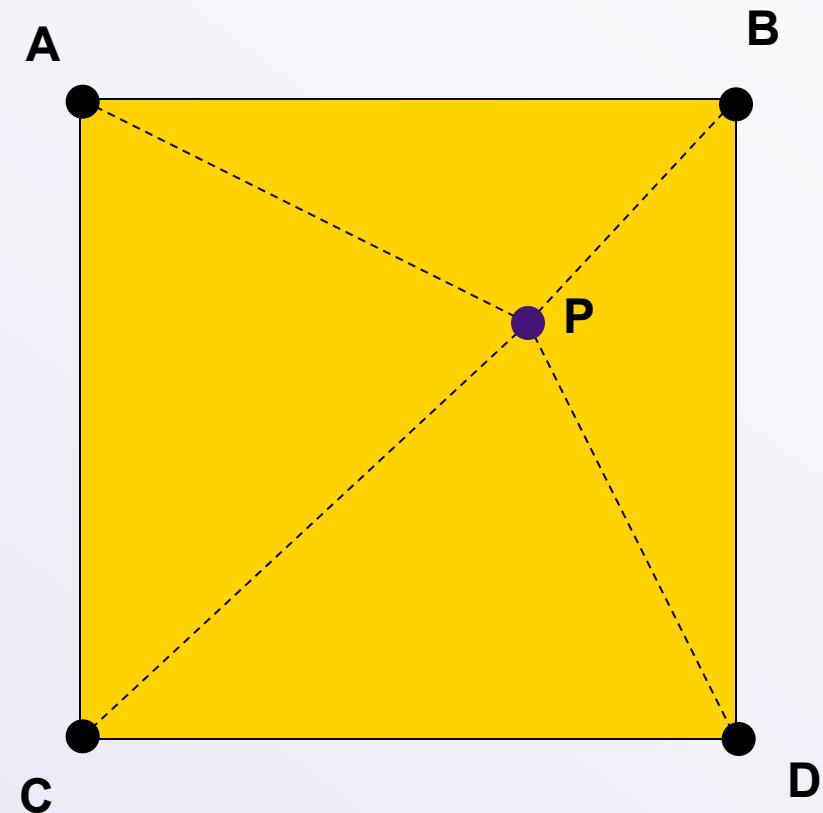


Métodos de reamostragem

- Nearest Neighbor
- Interpolação Bilinear e Trilinear
- Convolução Cúbica
- Interpolação por Spline Cúbica
- Funções Radiais Simétricas

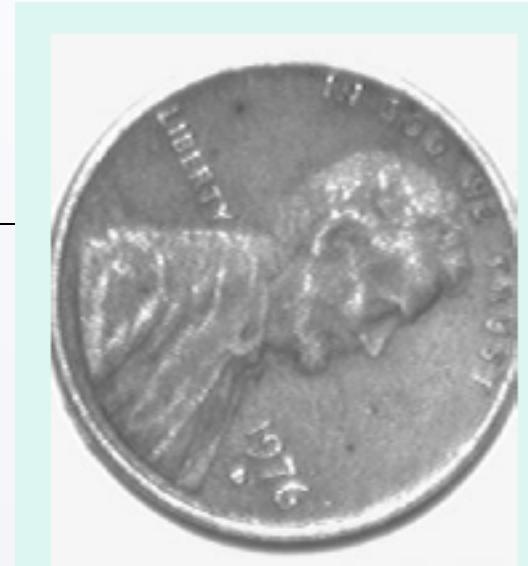
Interpolação “Nearest Neighbor”

- O valor de P é determinado pelo ponto mais próximo;
- Medidas:
 - Distância Euclideana
 - Distância de Manhattan



Nearest Neighbor

- Atribua a intensidade no ponto (X, Y) como a intensidade do ponto mais próximo: $[\text{round}(X), \text{round}(Y)]$.
- Rápido, mantém histograma original, porém produz *aliasing* nas bordas.



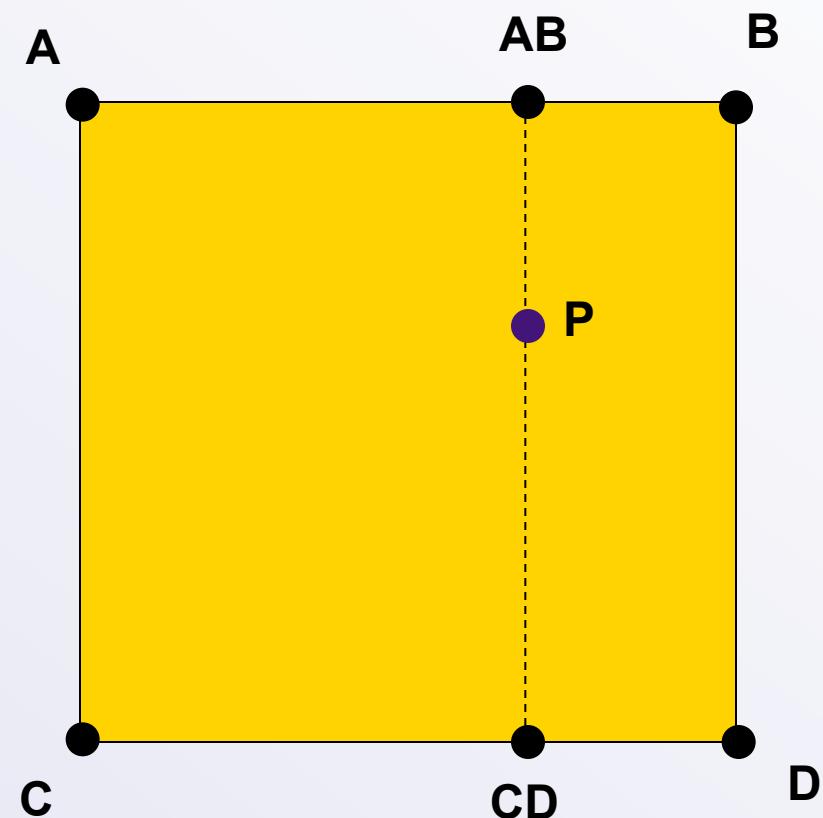
Original



Resampled

Interpolação Bilinear

- Utilizando interpolação linear, determina-se:
 - AB
 - CD
- Interpola-se o valor de P considerando AB e CD;



Interpolação Bilinear: Exemplo



Original



Resampled

Convolução Cúbica

- Na convolução cúbica, 16 pixels são usados para determinar a intensidade de (X,Y).
- Calculada linha a linha, coluna a coluna.

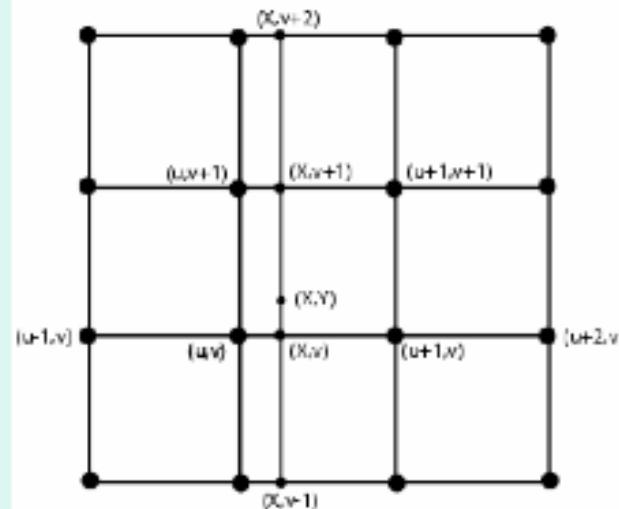
$$f(X) = I(u-1)f_{-1} + I(u)f_0 + I(u+1)f_1 + I(u+2)f_2$$

$$f_{-1} = -\frac{1}{2}t^3 + t^2 - \frac{1}{2}t,$$

$$f_0 = \frac{3}{2}t^3 - \frac{5}{2}t^2 + 1,$$

$$f_1 = -\frac{3}{2}t^3 + 2t^2 + \frac{1}{2}t,$$

$$f_2 = \frac{1}{2}t^3 - \frac{1}{2}t^2,$$



Convolução Cúbica: Exemplo



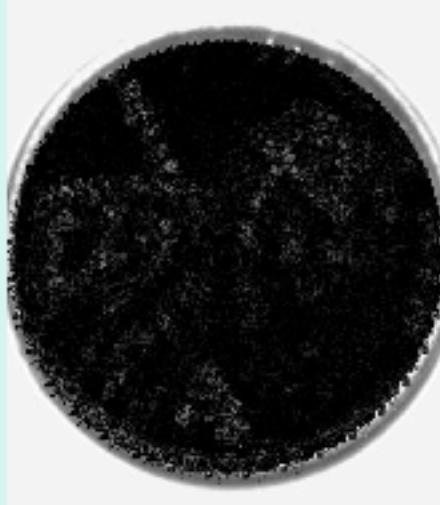
Original



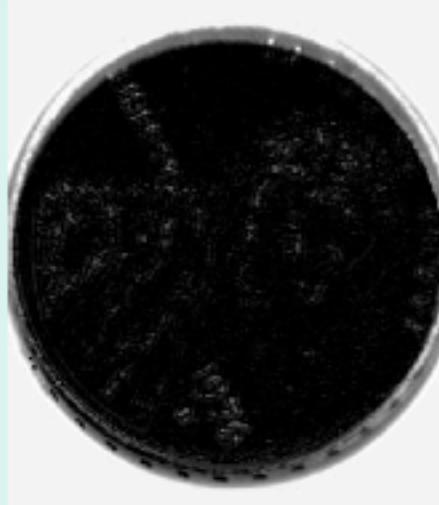
Resampled

Comparação

- Vamos supor que uma imagem seja rotacionada 36 vezes com um incremento angular de 10° e então subtraída da imagem original.



Nearest-neighbor



Bilinear



Cubic convolution

Filtragem

Modos determinados através de:

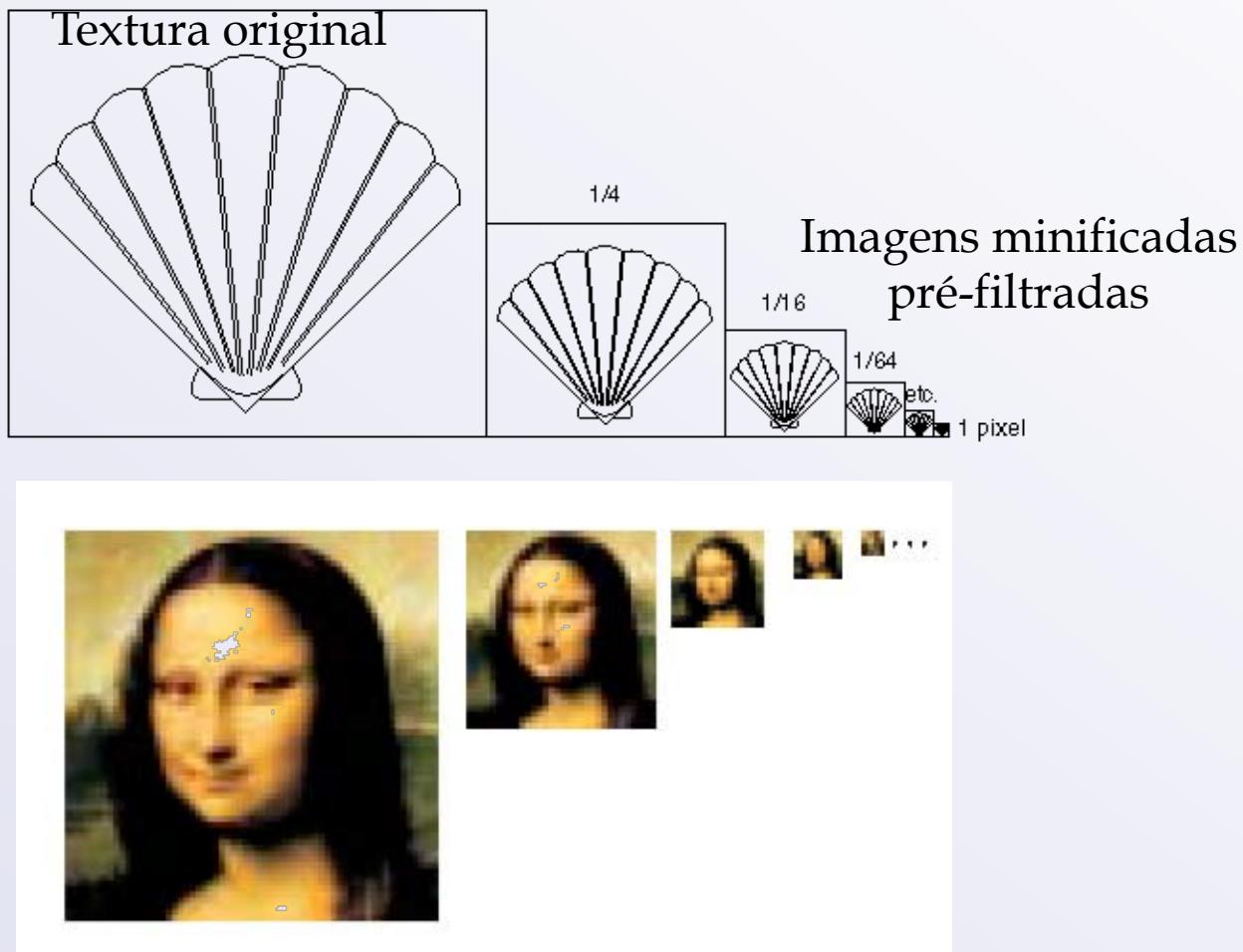
- `glTexParameter(target, type, mode)`

```
glTexParameter(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,  
               GL_NEAREST);
```

```
glTexParameter(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,  
               GL_LINEAR);
```

O uso da filtragem linear requer uma borda de um texel extra para filtragem nas arestas (border = 1)

Texturas Mipmap



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

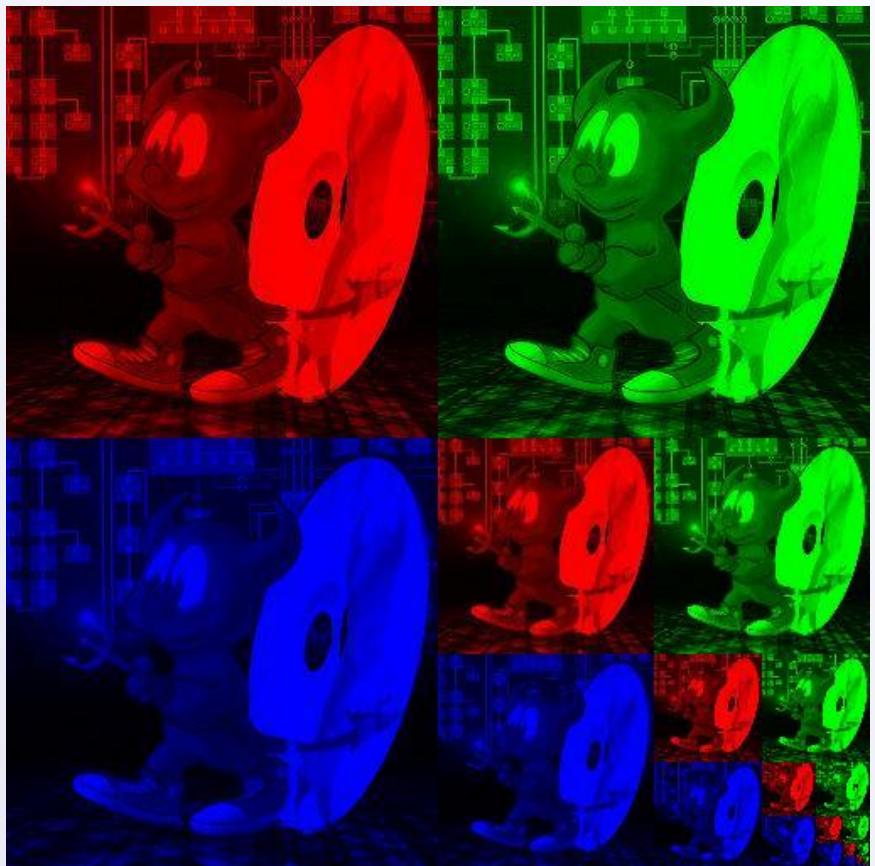
```
glTexImage*D( GL_TEXTURE_D, level, ... )
```

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

```
gluBuild*DMsMipmaps( ... )
```

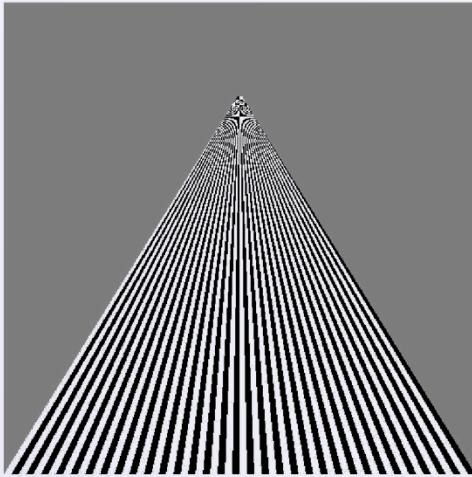
- OpenGL a partir da versão 1.2 suporta facilidades mais sofisticadas para níveis de detalhe (LOD)

Mapeamento MIP

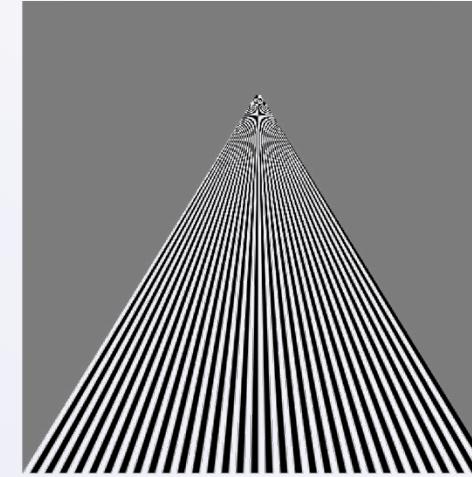


Exemplo

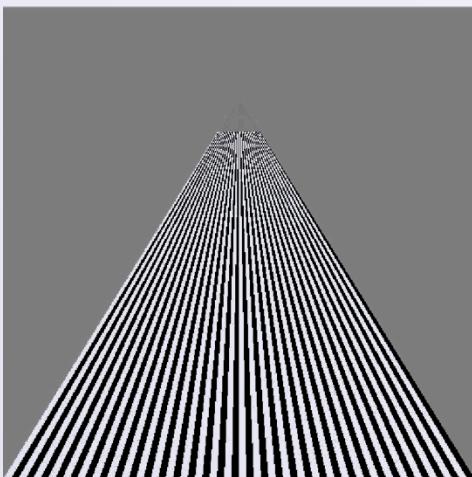
amostragem
por
pontos



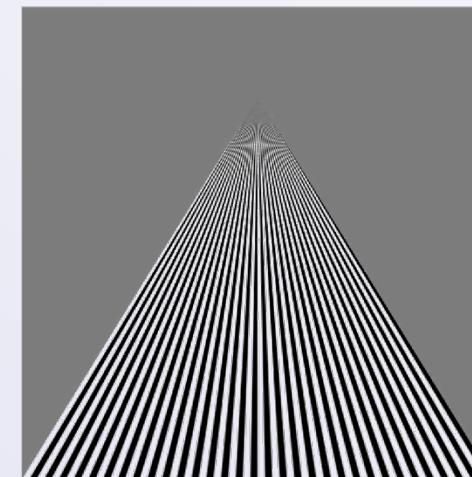
filtragem linear



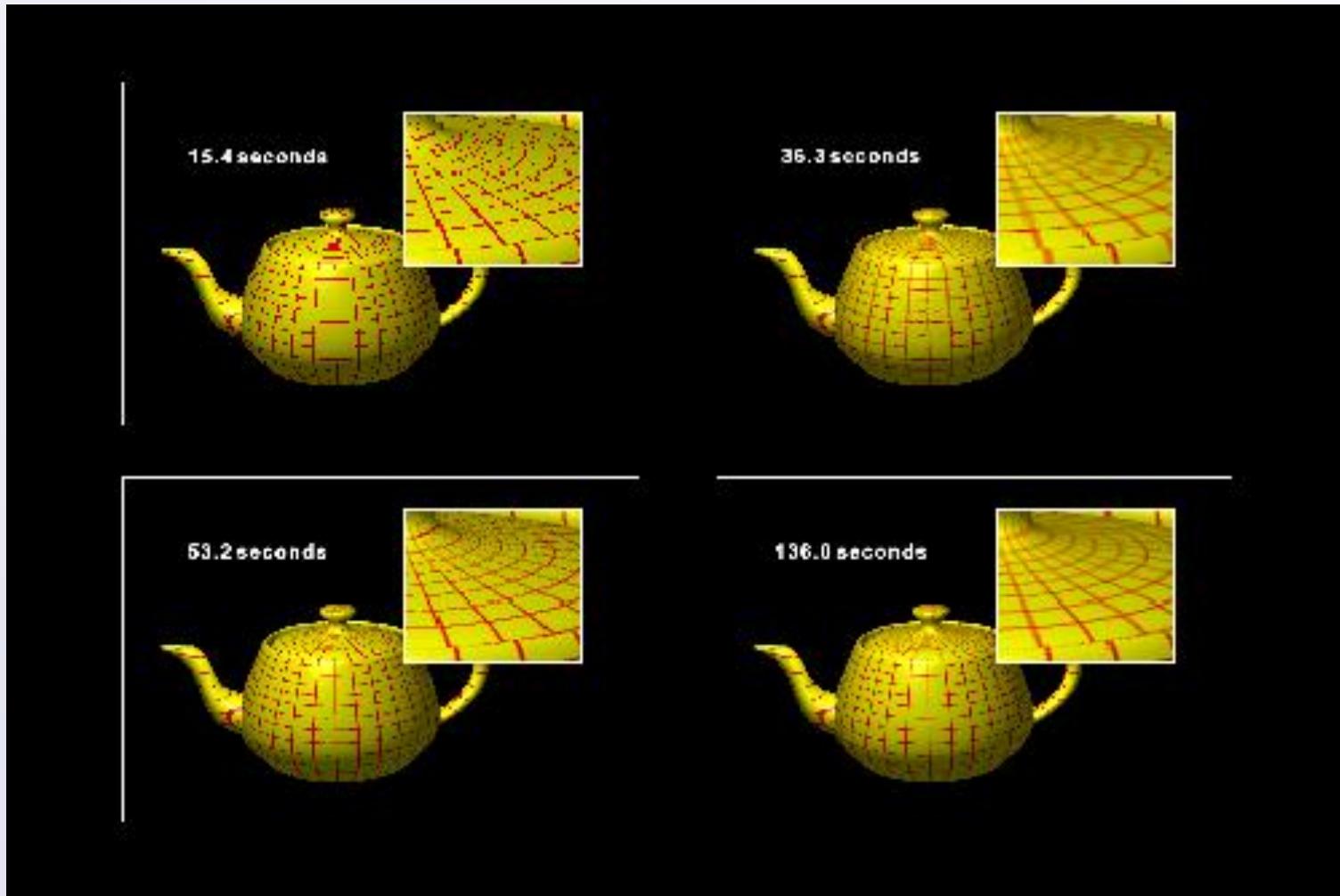
amostragem
por pontos
mipmap



filtragem linear
mipmap



Exemplo

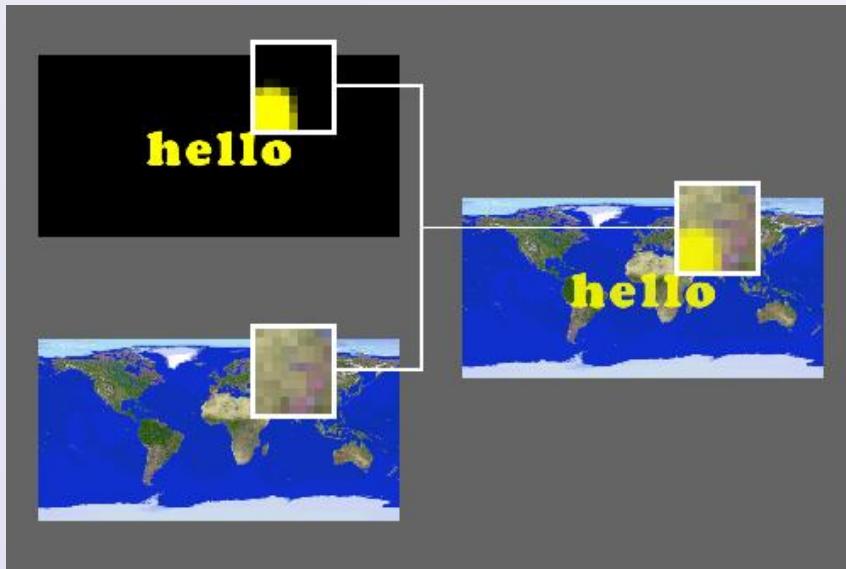


Outras Facilidades

- 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

Composição de texturas

- Ocasionalmente, é necessário combinar-se duas ou mais texturas em um objeto. Pode ser realizado através do canal alfa, emulando transparência em texturas.



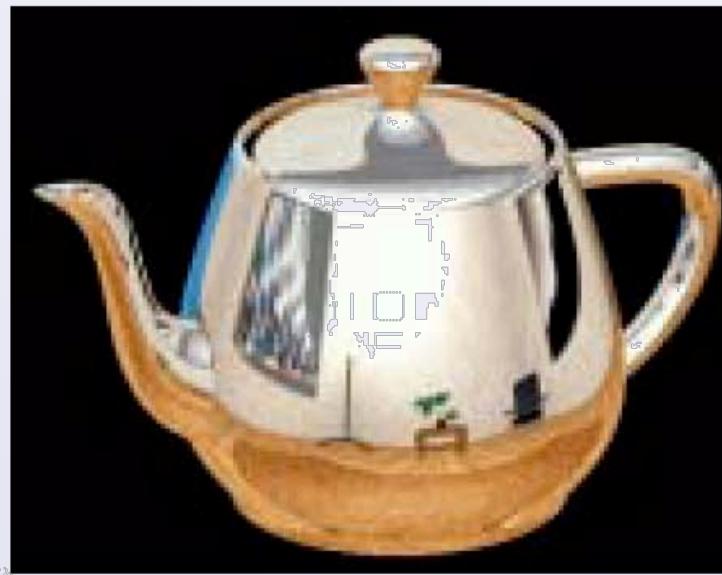
Coordenadas de textura

- O OpenGL pode gerar coordenadas de textura automaticamente

`glTexGen{ifd} [v] ()`

- Especificação de um plano
 - Geração de coordenadas de textura a partir da distância do plano
- Modos de geração
 - `GL_OBJECT_LINEAR`
 - `GL_EYE_LINEAR`
 - `GL_SPHERE_MAP`
 - `GL_REFLECTION_MAP`

Mapeamentos de reflexão

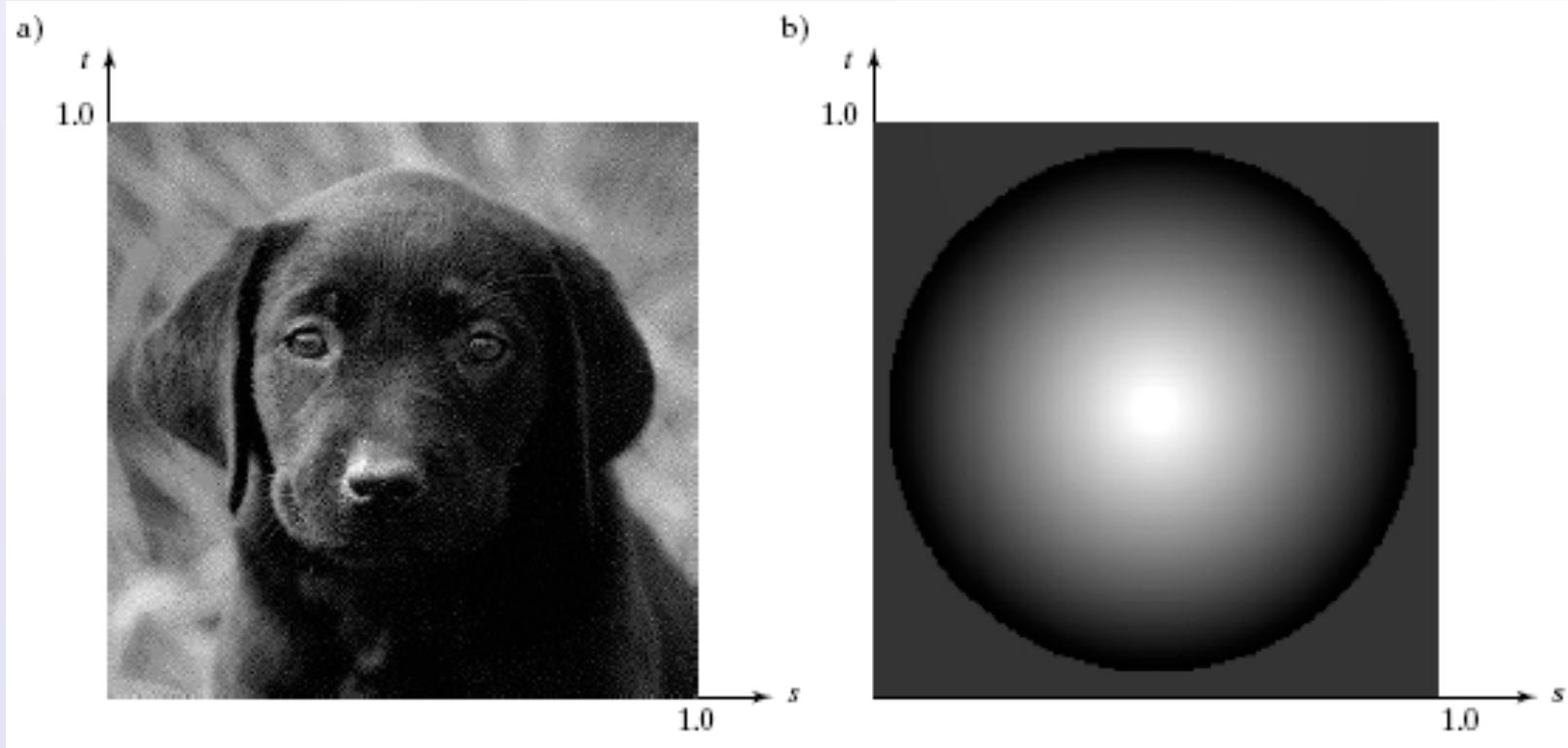


(a)



(b)

Criação de texturas



Texturas procedurais

- Função para desenho de uma esfera:

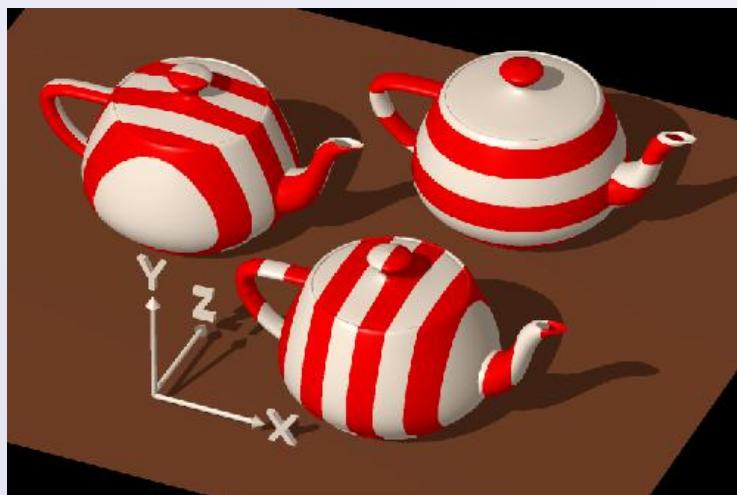
```
float Esfera (float s, float t) {  
    float r = sqrt((s-0.5)*(s-0.5) + (t-0.5)*(t-0.5));  
    if (r <= 0.3) return 1 - r / 0.3;  
    else return 0.2;  
} // 0.2 é a cor de fundo
```

- A função varia da cor branca (centro) para a cor preta (bordas)
- Podemos criar qualquer função para que, dados os valores de s e t , produza um valor de textura.

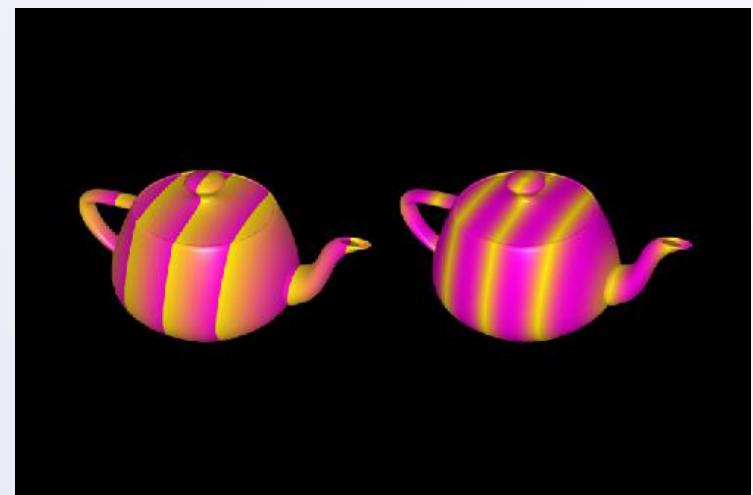
Texturas procedurais

A textura não é gerada por mapas. Em vez disso, um procedimento é usado para calcular a cor a ser aplicada ao pixel

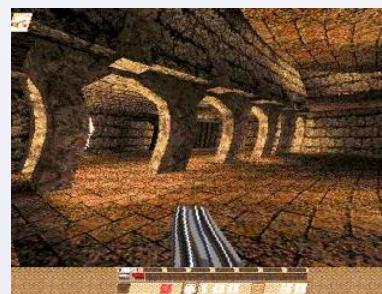
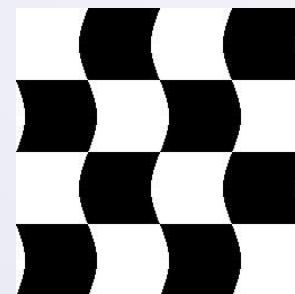
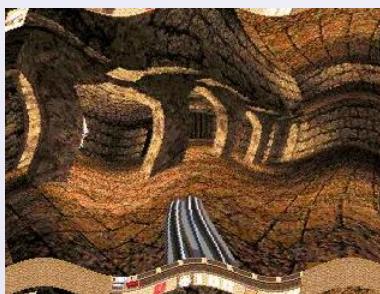
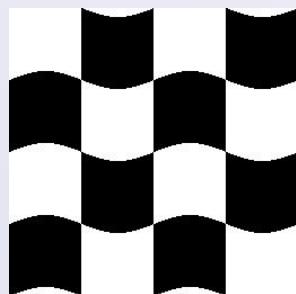
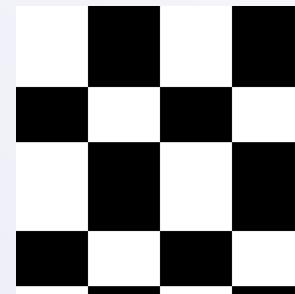
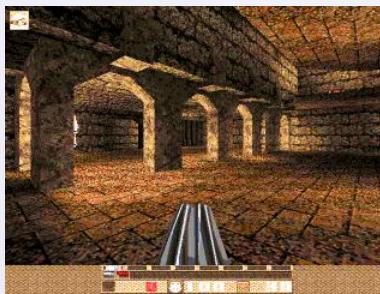
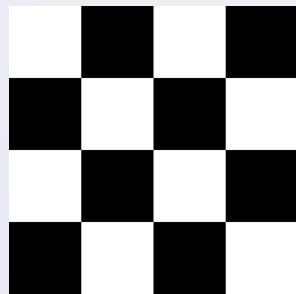
$\text{floor}(z)\%2$



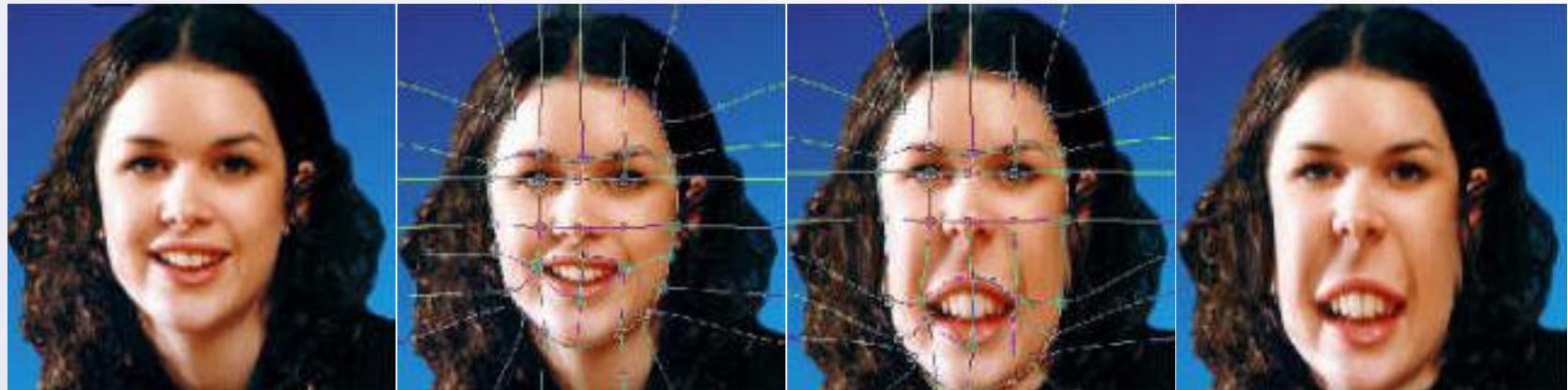
$\text{ramp}(x) \quad \sin(x)$



Distorções



Exemplo de distorções



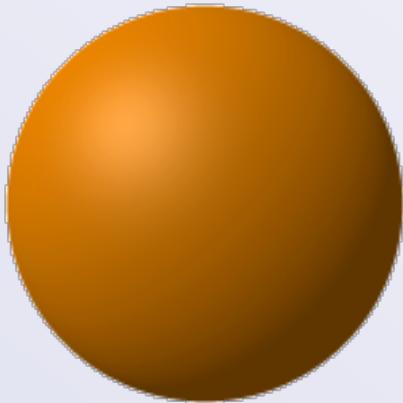
Action: Warp, shrink ce eyes
Warp, vertical doubleB on mouth

Action: Warp, horizontal doubleG on head

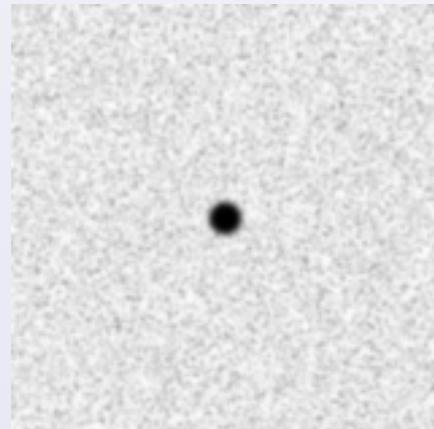
Action: Warp, blob ce eyes, blobx2 on mouth

Mapeamento de rugosidade

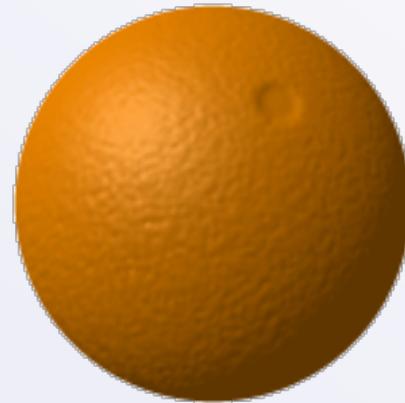
- A imagem a ser mapeada é utilizada para fazer uma perturbação do vetor normal à superfície.



geometria



imagem



mapeamento

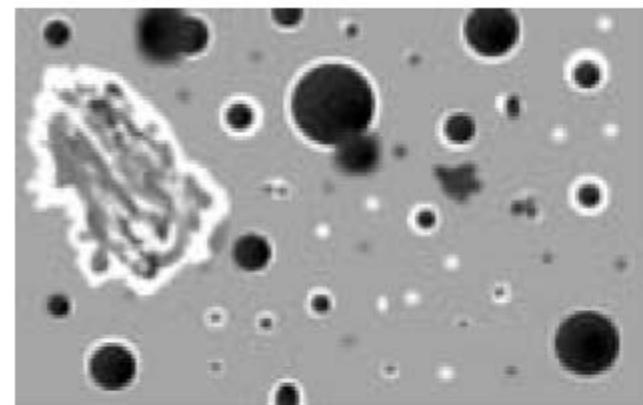
Mapeamento de rugosidade

- A perturbação do vetor é definido como:

$$Q(u, v) = P(u, v) + I(u, v) \cdot \mathbf{n}(u, v)$$

$$\mathbf{n}_1 = \frac{\partial Q}{\partial u} \times \frac{\partial Q}{\partial v}$$

$$\mathbf{n}' = \mathbf{n}_1 / |\mathbf{n}_1|$$

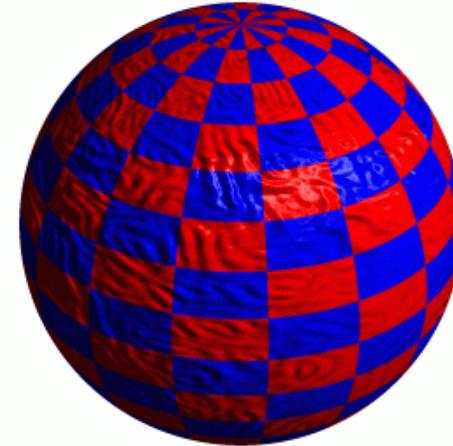
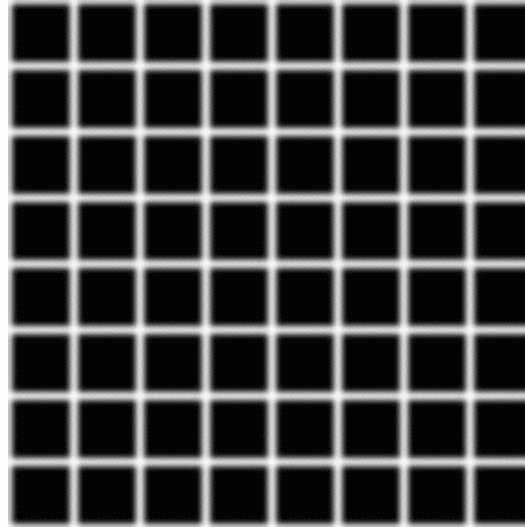
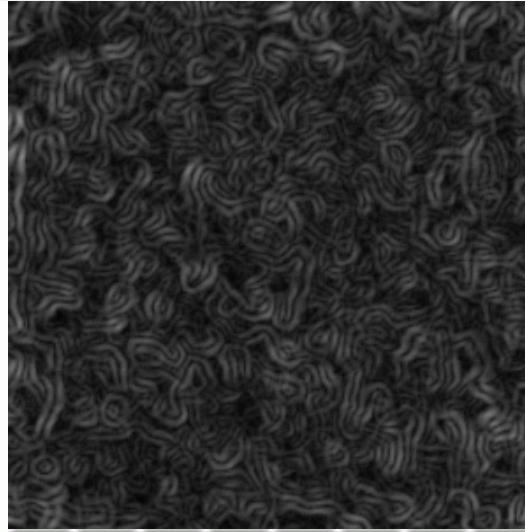
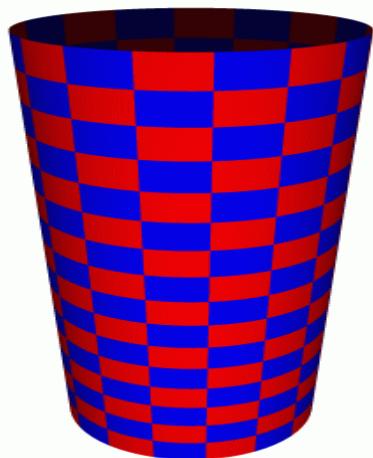
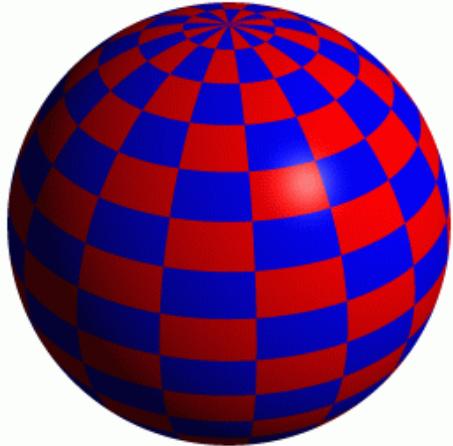


(a) Textura fonte.



(b) Mapeamento.

Rugosidade



27.3



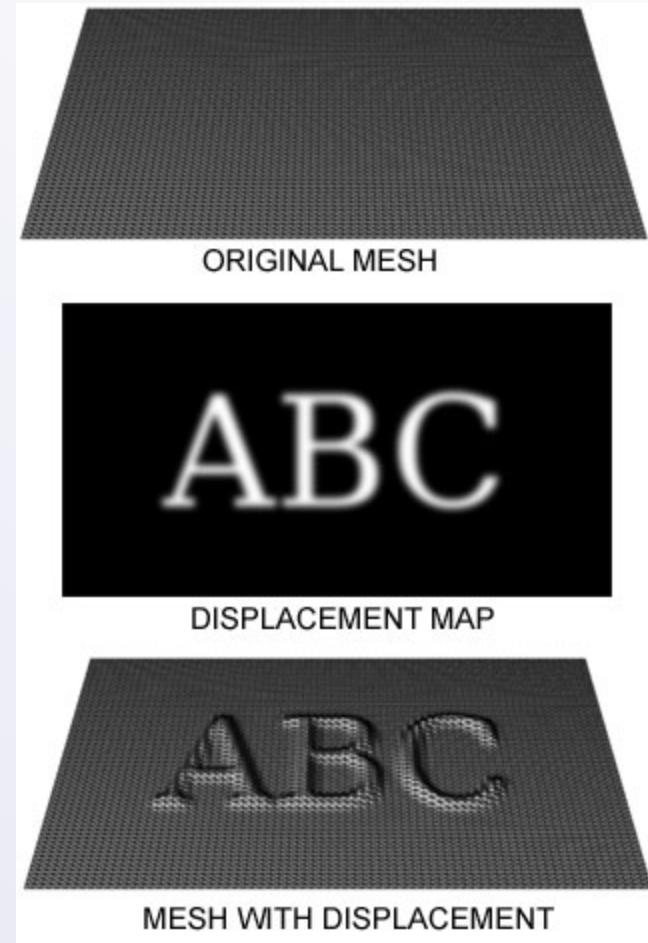
Mapeamento de deslocamento

- A imagem a ser mapeada é utilizada para fazer uma perturbação na geometria da superfície.

$$P'(u, v) = P(u, v) + I(u, v) \cdot \mathbf{n}(u, v)$$

$$\mathbf{n}_1 = \frac{\partial P}{\partial u} \times \frac{\partial P}{\partial v}$$

$$\mathbf{n}' = \mathbf{n}_1 / |\mathbf{n}_1|$$



Comparação

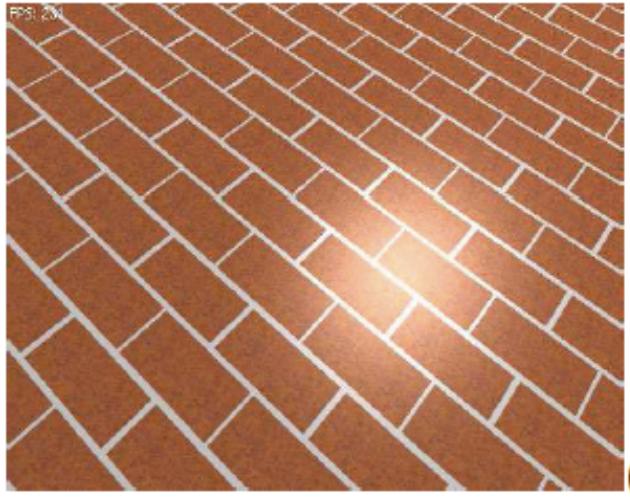


Figura 1.4: Ilustração da técnica de mapeamento de deslocamentos. (a) Textura mapeada do modo convencional. (b) Textura mapeada com deslocamento. (*Cortesia de Fabio Policarpo.*)

Comparação



(a) De rugosidade.



(b) De deslocamento.

Figura 39. Mapeamentos.

Tarefa de casa

- Além dos mapeamentos que vimos, existe ainda um outro:
 - “Shadow mapping”
- Como ele funciona?
- Como implementá-lo em OpenGL ?

