LEI / LCC
Departamento de Informática
Universidade do Minho

Texturas

Texturas: Definição e Aplicação
Cube Maps

Texturas

- Aplicar imagens 1D,2D ou 3D a primitivas geométricas

- Utilizações:

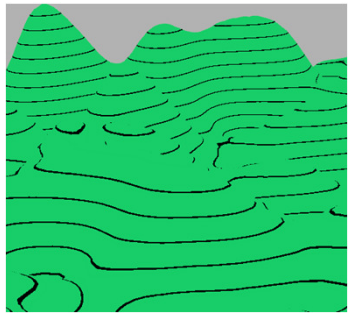
  - Simular materiais: madeira, granito, tijolo
  - Reduzir complexidade geométrica
  - Simulação de fenómenos naturais (reflexões, refracção, luz, lens flare)

Texturas

- Imagem: dimensões são potências de 2. Por exemplo: 256 x 128
  - Nota: actualmente já é possível trabalhar sem esta restrição
- Exemplos de Formatos: RGB, RGBA,...
- OpenGL não tem nenhuma função para carregar texturas de ficheiro

Texturas

- 1D
  - Linha de *pixels*



- 2D
  - Textura "normal": imagem

- 3D
  - Volumes. Permitem aplicar texturas como se tratasse escultura

Texturas - Utilização

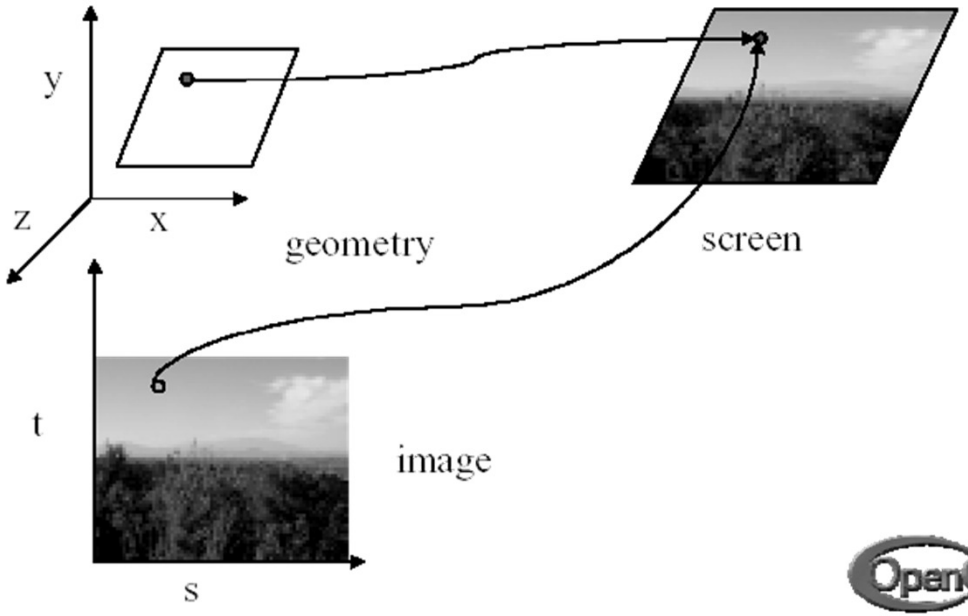
- Definição
  - Carregar a imagem
  - Criar uma textura em OpenGL
  - Definir parâmetros da textura
- Aplicação
  - Definir vértices
  - Definir transformações geométricas a aplicar à textura

Texturas – Aplicação

- Para aplicar uma textura a um polígono é necessário definir um mapeamento entre os *pixels* da textura e os vértices do polígono.

- As texturas 2D têm um sistema de coordenadas nos eixos s (=x),t (=y).

- `glTexCoord2f(s,t)`.



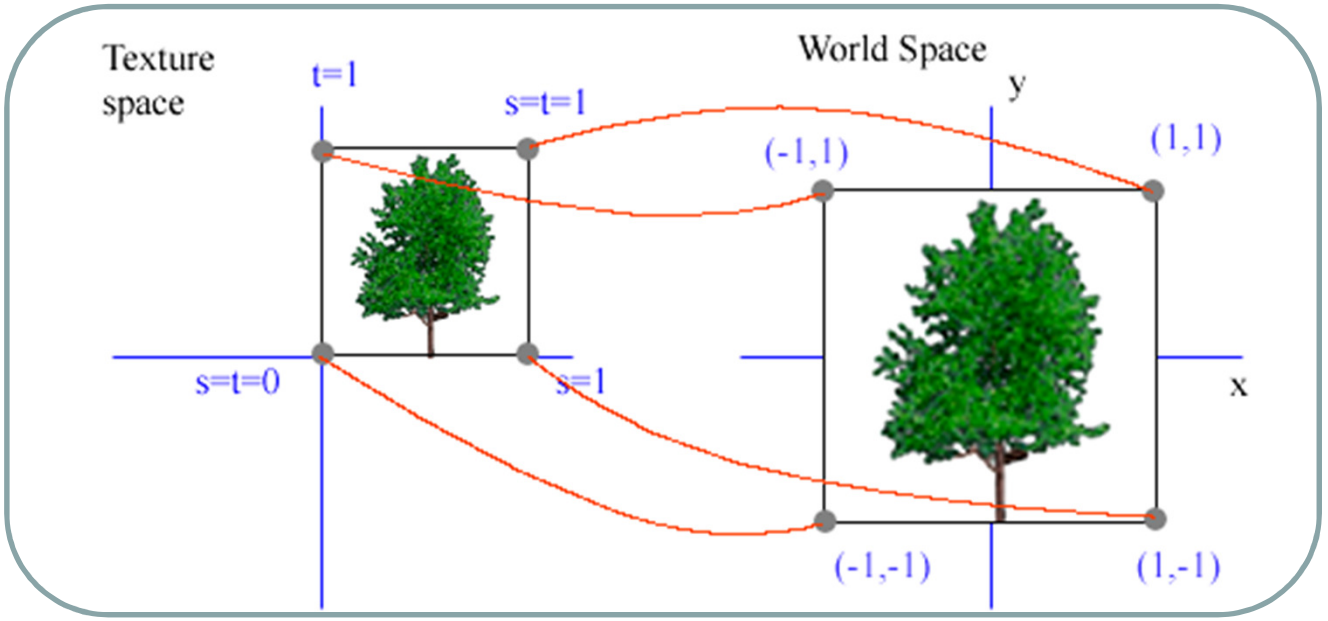
Texturas - Aplicação

- Ao definir os vértices do polígono, definem-se anteriormente as coordenadas a aplicar para a textura.

```
glBindTexture(GL_TEXTURE_2D,texID);
glBegin(GL_QUADS);
     glTexCoord2f(0,0);glVertex3f(-1.0f, -1.0f, 0.0f);
     glTexCoord2f(1,0);glVertex3f( 1.0f, -1.0f, 0.0f);
     glTexCoord2f(1,1);glVertex3f( 1.0f, 1.0f,  0.0f);
     glTexCoord2f(0,1);glVertex3f(-1.0f, 1.0f,  0.0f);
glEnd();
```

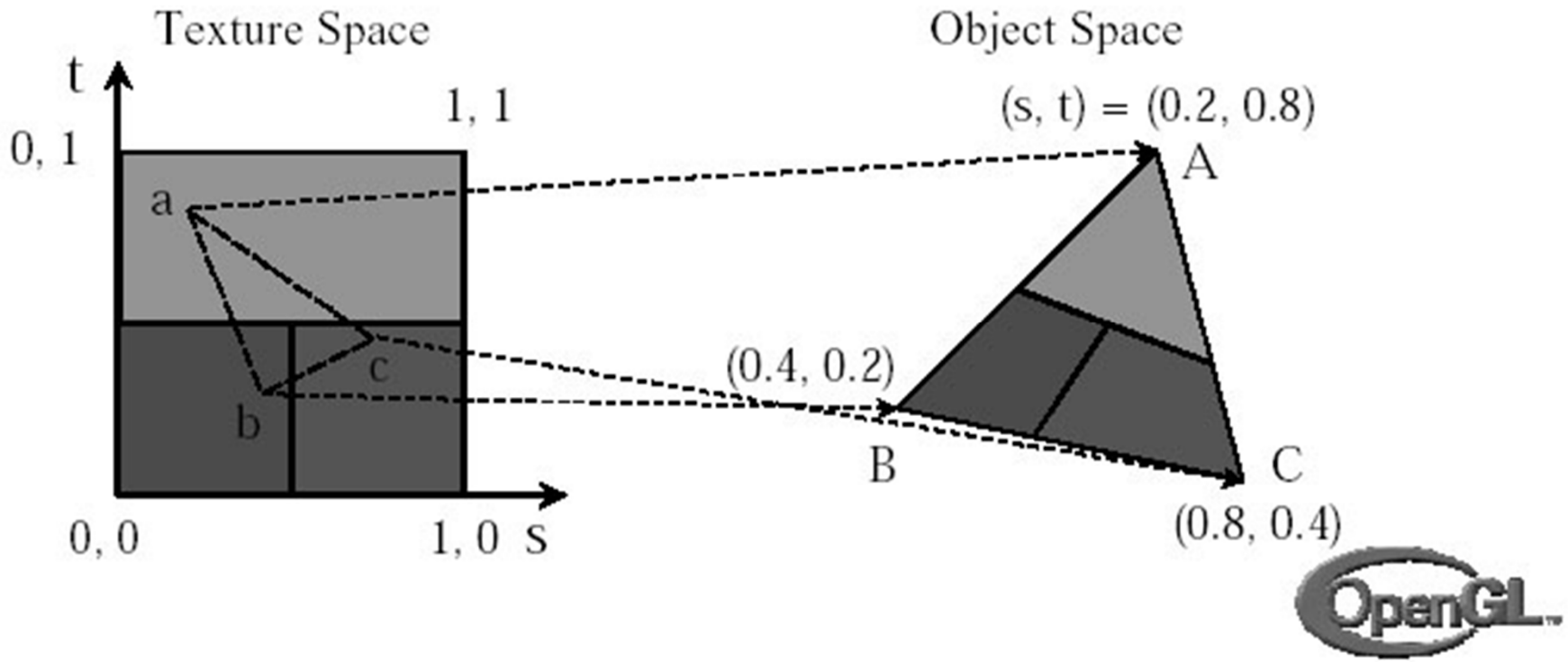
Texturas - Aplicação

```
glBindTexture(GL_TEXTURE_2D,texID);
glBegin(GL_QUADS);
          glTexCoord2f(0,0);glVertex3f(-1.0f, -1.0f, 0.0f);
          glTexCoord2f(1,0);glVertex3f( 1.0f, -1.0f, 0.0f);
          glTexCoord2f(1,1);glVertex3f( 1.0f, 1.0f,  0.0f);
          glTexCoord2f(0,1);glVertex3f(-1.0f, 1.0f,  0.0f);
    glEnd();
```



Texturas - Aplicação

- A escolha de coordenadas no espaço das texturas é "livre".



Texturas - Aplicação

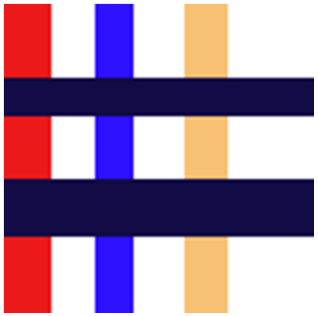
- Matriz para Texturas
  - Permite realizar transformações geométricas sobre a textura.

```
glMatrixMode(GL_TEXTURE);
glTranslatef(0.5,0,0);
glRotatef(45,0,0,1);

glMatrixMode(GL_MODELVIEW);
glBegin(GL_QUADS);
   ...
glEnd();
```

Texturas - Parâmetros

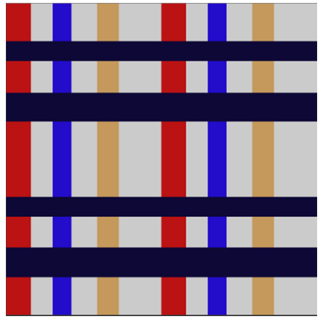
Clamp & Repeat

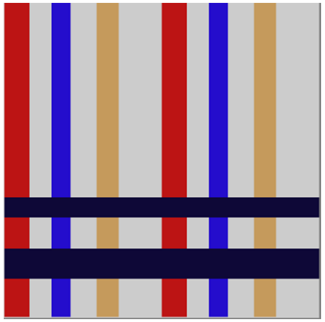
Imagem Original

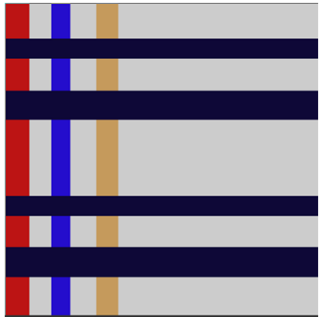


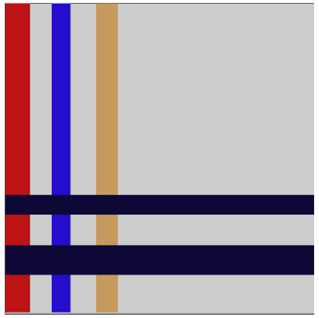
GL_CLAMP
GL_REPEAT

As várias hipóteses para CLAMP e REPEAT (2x2)









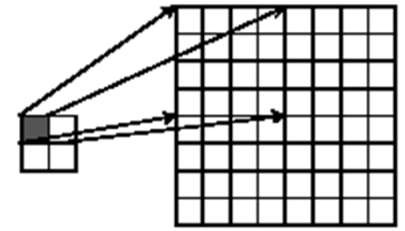
Texturas - Aplicação

Demo Texturas Nate Robbins

Texturas - Filtros: Mag

- Utilizado quando a um pixel da textura corresponde mais que um pixel da imagem final, ou seja quando a textura é ampliada

- `GL_LINEAR` **ou** `GL_NEAREST`

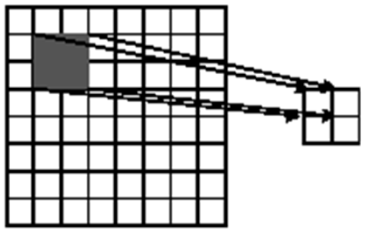


Magnification

Texturas - Filtros: Min

- Utilizado quando a um pixel da textura corresponde <u>menos</u> que um pixel da imagem final, ou seja quando a textura é comprimida visualmente

- `GL_LINEAR` **ou** `GL_NEAREST`



Minification

Texturas - Filtros

Mag:Nearest

Texturas - Filtros

Mag: Linear

parece
desfocado ao perto!

Texturas - Definição

```
// Assumir que as seguintes variáveis imageData, imageWidth e
// imageHeight têm os valores apropriados

int texName[1];

glGenTextures(1, texName);
glBindTexture(GL_TEXTURE_2D, texName[0]);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);

glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, imageWidth, imageHeight,
             border, GL_RGB, GL_UNSIGNED_BYTE, imageData);
```

Texturas: Cor Final

- A cor da textura pode ser combinada com a cor do polígono.

		Modo RGBA
– `GL_REPLACE`	C = Ct	A = At
– `GL_MODULATE`	C = Ct * Cf	A = At * Af
– `GL_BLEND`	C = Cf*(1-Ct) + Cc*Ct	A = Af * At
– `GL_DECAL`	C = Cf*(1-At) + Ct*At	A = Af

f = fragmento, t = textura, c = `GL_TEXTURE_ENV_COLOR`

- ```
  glTexEnvi(GL_TEXTURE_ENV,
      GL_TEXTURE_ENV_MODE,param);
  ```

- ```
  glTexEnvfv(GL_TEXTURE_ENV,
      GL_TEXTURE_ENV_COLOR,param);
  ```

Texturas: Transparência

- Para transparências parciais a ordem de desenho é importante
- Para transparências totais pode-se utilizar o teste do alpha
- Este teste é realizado antes da escrita no Z-buffer, e elimina todos os pixels que não passam no teste...
- ... logo os pixels que são eliminados não alteram o Z-buffer.

Texturas: Transparência

- Transparência total na prática

```
glEnable(GL_ALPHA_TEST);
glAlphaFunc(GL_GREATER, 0);
```

Texturas: Transparências

- Transparências Parciais:
  - A ordem é importante: as transparências devem ser desenhadas no final
  - É necessário especificar como combinar a cor da textura com a cor já presente no frame buffer
    - $C_t * S + C_f * D$ // fórmula geral de *Blend*
      - $S = Alpha_t$; $D = 1 - Alpha_t$

Texturas: Transparência

- Em OpenGL
  - `glEnable(GL_BLEND);`
  - `glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);`
- há quem prefira:
  - `glEnable(GL_BLEND);`
  - `glBlendFunc(GL_SRC_ALPHA,GL_ONE);`

Texturas

- 1D
  - `glTexImage1D(GL_TEXTURE_1D,...)`
- 3D
  - `glTexImage3D(GL_TEXTURE_3D,...)`

Texturas

- Em OpenGL é necessário activar a operação de aplicação de texturas:

```
glEnable(GL_TEXTURE_1D);
glEnable(GL_TEXTURE_2D);
glEnable(GL_TEXTURE_3D);
```

Geração de Coordenadas

- O OpenGL permite a geração de coordenadas para texturas automática.

- Note-se que a geração de coordenadas não pretende substituir a necessidade de definir coordenadas de texturas, mas sim permitir novas aplicações.

Geração de Coordenadas

- Três modos possíveis:
  - `GL_EYE_LINEAR`
    - Permite fixar as texturas no espaço, e o objecto move-se na textura.
  - `GL_OBJECT_LINEAR`
    - Texturas fixas ao objecto, aplicação tradicional.
  - `GL_SPHERE_MAPPING, GL_CUBE_MAPPING`
    - Os objectos reflectem o ambiente, tal como no filme Exterminador, ou no Abismo.

Geração de Coordenadas

Demo Geração Coordenadas
(texGenCoord)

Geração de Coordenadas

- `GL_OBJECT_LINEAR`

$$s = s_0 * x + s_1 * y + s_2 * z + s_3 * w$$

- sendo $v = (x,y,z,w)$ o ponto em coordenadas do mundo, e $S = (s_0, s_1, s_2, s_3)$ um plano

- Os planos S,T,R,Q determinam o referencial a partir do qual são calculadas as coordenadas.

Geração de Coordenadas

Terreno: utiliza textura 2D

Calculo automático: Distância aos planos x=0 (s) e z=0 (t)

Escala da textura de forma a que 1 unidade de textura corresponda ao terreno inteiro

Geração na prática

- **Activar Geração de Coordenadas**

```
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
```

- **Modo: GL_OBJECT_LINEAR**

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
```

- **Os Planos S e T:**

```
GLfloat planeS[] = {1.0, 0.0, 0.0, 0.0};
GLfloat planeT[] = {0.0, 0.0, 1.0, 0.0};

glTexGenfv(GL_S, GL_OBJECT_PLANE, planeS);
glTexGenfv(GL_T, GL_OBJECT_PLANE, planeT);
```

- **Escala da Textura**

```
glMatrixMode(GL_TEXTURE);
glScalef(1.0/(imageWidth),1.0/(imageHeight),1);
glMatrixMode(GL_MODELVIEW);
```

Geração de Texturas

Demo Terreno com Geração de Texturas

Geração de Coordenadas

Curvas de nível: utiliza textura 1D

Calculo automático: Distância ao plano y=0 (s)

Escala da textura de forma a que 1 unidade de textura corresponda a *n* metros.

Textura repetida permite visualizar múltiplas curvas de nível.

Geração na prática

- **Activar Geração de Coordenadas**

```
glEnable(GL_TEXTURE_GEN_S);
```

- **Modo: GL_OBJECT_LINEAR**

```
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_OBJECT_LINEAR);
```

- **O Plano S :**

```
GLfloat planeS[] = {0.0, 1.0, 0.0, 0.0};

glTexGenfv(GL_S, GL_OBJECT_PLANE, planeS);
```

- **Escala da Textura**

```
glMatrixMode(GL_TEXTURE);
glScalef(1.0/n,1,1);
glMatrixMode(GL_MODELVIEW);
```

Geração de Texturas

Demo Curvas de Nível com
Geração de Texturas

Texturas - Mipmapping

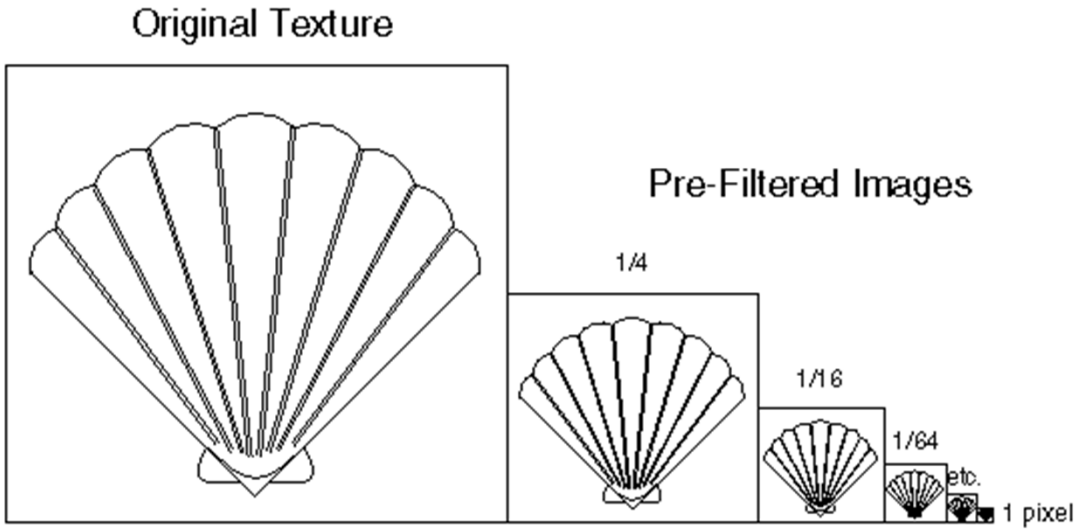
- Do Latim "multum in parvo".
- Problema: alterações inesperadas ao encolher texturas à medida que a camera se afasta.
- Causa: O processo de aplicação de filtros a uma imagem muito encolhida pode implicar alterações abruptas à imagem projectada.

Texturas - Mipmapping

- Solução: Utilizar múltiplas texturas de diferentes resoluções para utilizar a escalas diferentes.
- Por exemplo: textura original 32 x 16
- Fornecer texturas: 32x16, 16x8, 8x4, 4x2, 2x1, 1x1.
- É necessário fornecer uma sequência de níveis consecutivos (potências de 2).

Texturas - Mipmapping

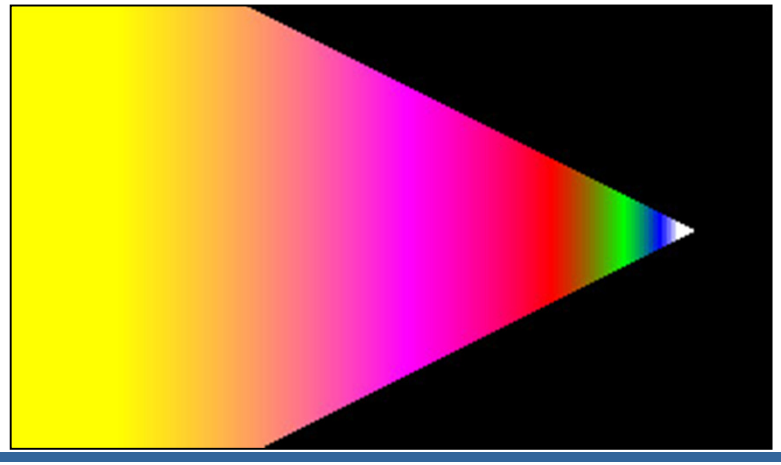
- Figura do Red Book:

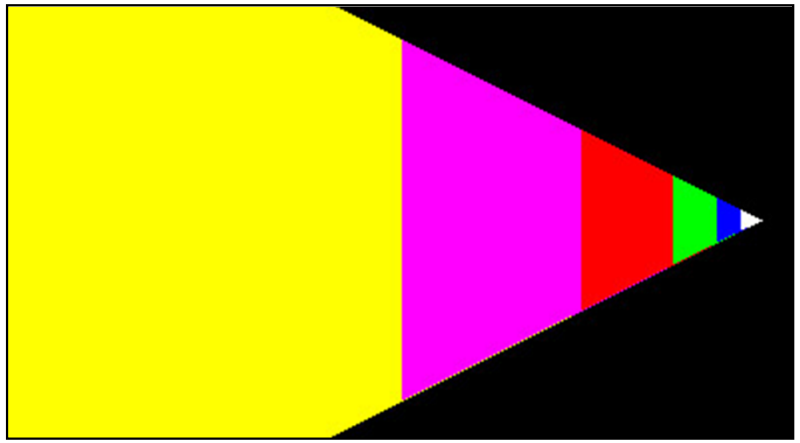


Texturas - Mipmapping

- Que textura escolher para cada pixel?
  - a mais próxima da actual resolução, ou
  - uma combinação linear das duas mais próximas.

  - Imagens do Red Book





Texturas - Mipmapping

4 combinações disponíveis para filtrar uma textura (`GL_MIN_FILTER`):

- `GL_NEAREST_MIPMAP_NEAREST`
- `GL_LINEAR_MIPMAP_NEAREST`
- `GL_NEAREST_MIPMAP_LINEAR`
- `GL_LINEAR_MIPMAP_LINEAR`

O primeiro filtro diz respeito à textura, o segundo ao mipmapping.

Texturas - Mipmapping

- GLU permite a criação automática dos níveis necessários para o mipmapping.

  - Todos:

```
gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGBA, imageWidth, imageHeight,
                  GL_RGBA, GL_UNSIGNED_BYTE, imageData);
```

  - Do nível *base* ao nível *max*:

```
gluBuild2DMipmapLevels(GL_TEXTURE_2D, GL_RGBA, imageWidth, imageHeight,
                  GL_RGBA, GL_UNSIGNED_BYTE, level, base, max,
                  imageData);
```

- Pode-se também especificar quais os níveis a utilizar:

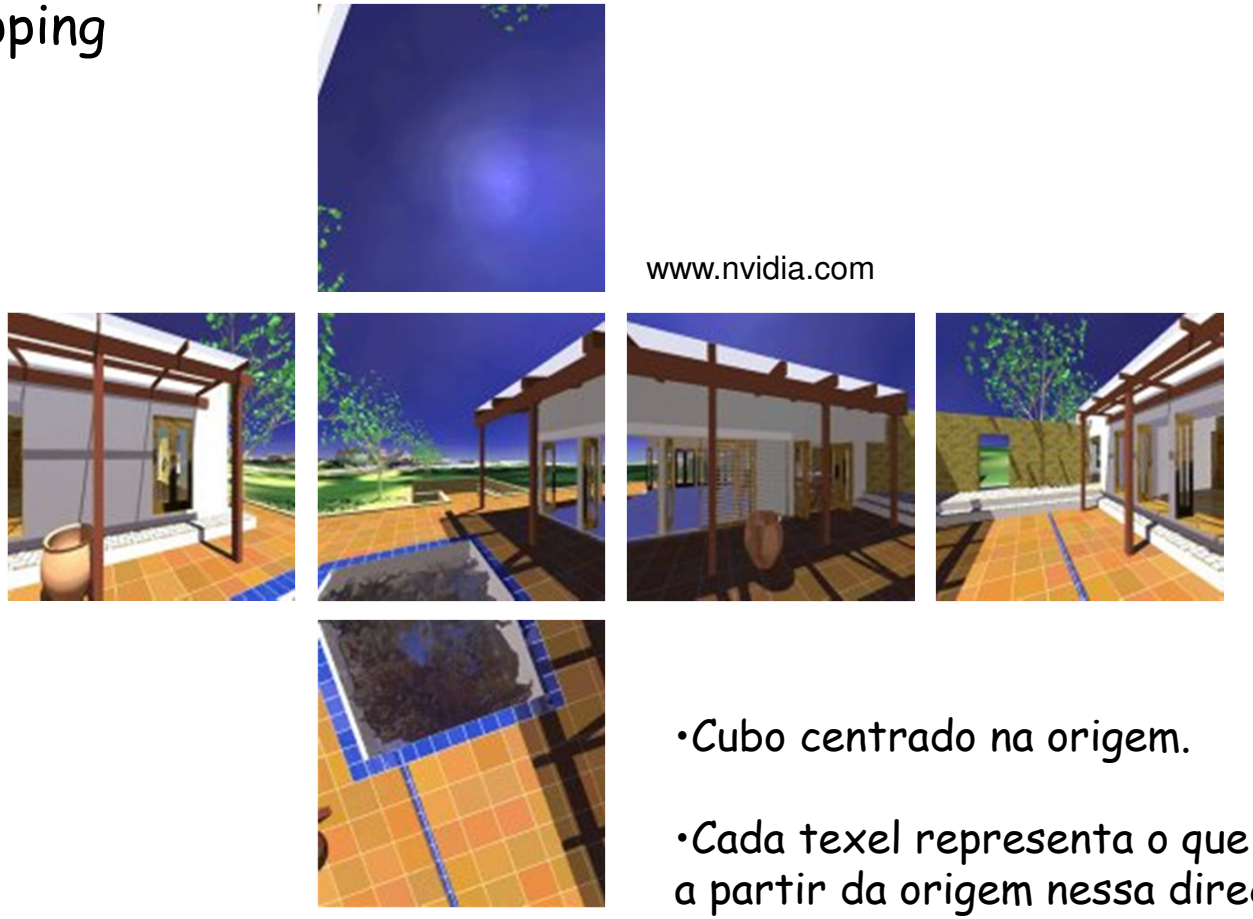
```
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_BASE_LEVEL,2);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAX_LEVEL,5);
```

Texturas - Mipmapping

```
glBindTexture(GL_TEXTURE_2D,texName[1]);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);

glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,
                GL_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER     ,
                GL_NEAREST_MIPMAP_NEAREST);


gluBuild2DMipmaps(GL_TEXTURE_2D, GL_RGB, imageWidth, imageHeight,
                        GL_RGB, GL_UNSIGNED_BYTE, imageData);

// especificar quais os níveis a utilizar
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_BASE_LEVEL,2);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAX_LEVEL,5);
```

OpenGL – Environment Map

Cube Mapping

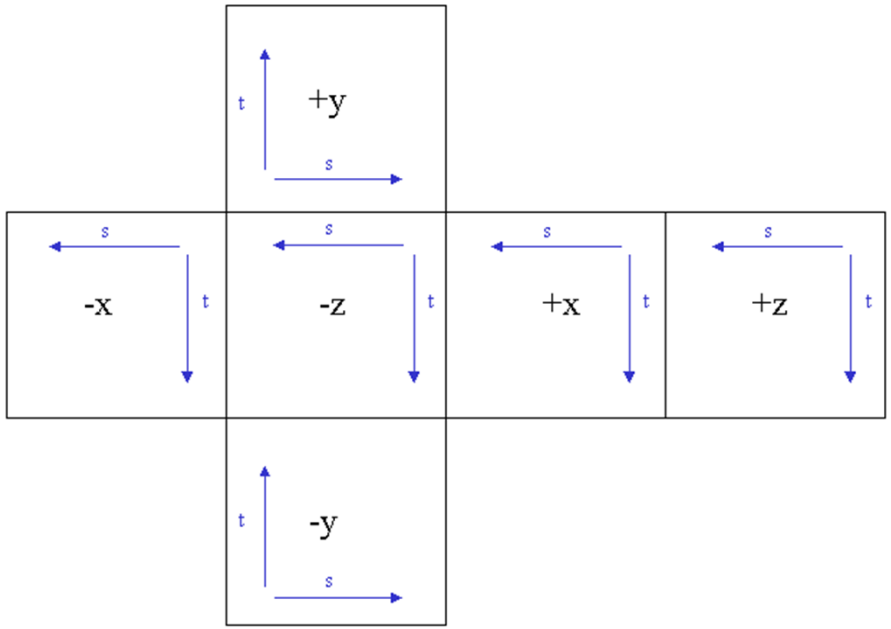


www.nvidia.com

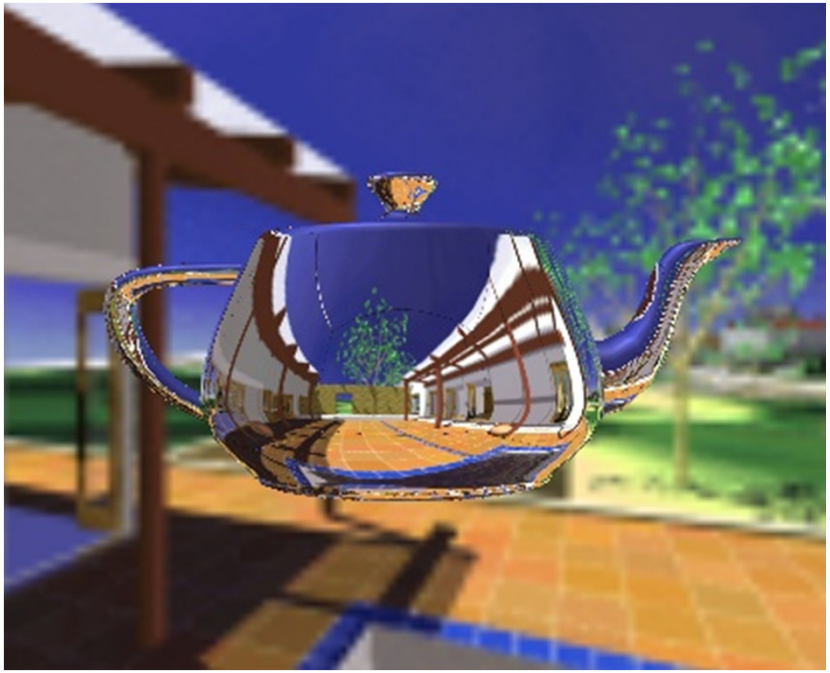
- Cubo centrado na origem.
- Cada texel representa o que seria visto a partir da origem nessa direcção

OpenGL – Environment Map

- Orientação das Imagens



OpenGL – Environment Map



OpenGL - Environment Map

- Código OpenGL para criar Cube Map

```
static GLenum faceTarget[6] = {
  GL_TEXTURE_CUBE_MAP_POSITIVE_X,
  GL_TEXTURE_CUBE_MAP_NEGATIVE_X,
  GL_TEXTURE_CUBE_MAP_POSITIVE_Y,
  GL_TEXTURE_CUBE_MAP_NEGATIVE_Y,
  GL_TEXTURE_CUBE_MAP_POSITIVE_Z,
  GL_TEXTURE_CUBE_MAP_NEGATIVE_Z
};
```

```
glGenTextures(1, texName);
glBindTexture(GL_TEXTURE_CUBE_MAP,texName[0]);

for (i=0; i<6;i++) {

      glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
      glTexParameteri(GL_TEXTURE_CUBE_MAP, GL_TEXTURE_MAG_FILTER, GL_LINEAR);

      glTexImage2D(faceTarget[i], 0, GL_RGB, imageWidth, imageHeight,
                0, GL_RGB, GL_UNSIGNED_BYTE, imageData);

}
```

OpenGL - Environment Map

- Código OpenGL para preparar estado para Cube Map

```
glEnable(GL_TEXTURE_CUBE_MAP);
glEnable(GL_TEXTURE_GEN_S);
glEnable(GL_TEXTURE_GEN_T);
glEnable(GL_TEXTURE_GEN_R);
glTexGeni(GL_S, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glTexGeni(GL_T, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
glTexGeni(GL_R, GL_TEXTURE_GEN_MODE, GL_REFLECTION_MAP);
```

OpenGL – Environment Map

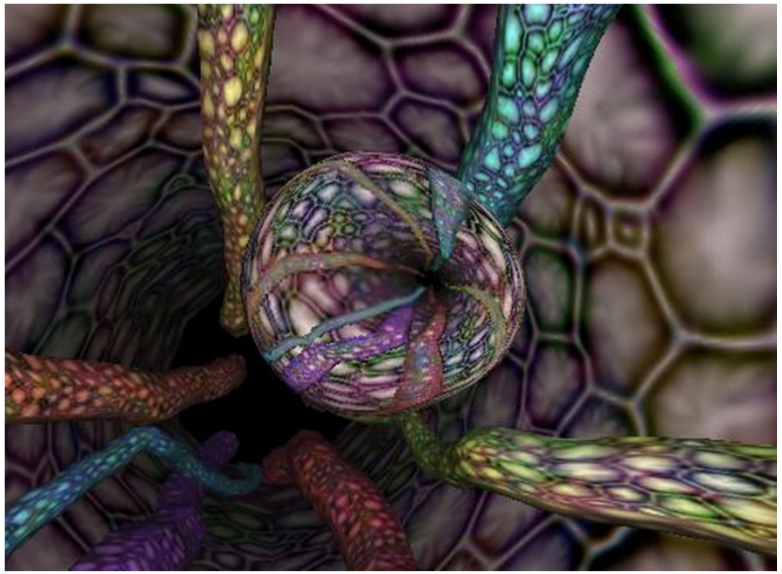
- Vantagens:
  - Rápido em Hardware
  - Fácil de Gerar em Runtime

OpenGL – Environment Map

- Geração para cenas artificiais:
  - Definir uma camera com ângulo de visão de 90º centrada na origem do objecto
  - Apontar a camera no eixo do X+ e capturar o frame buffer para uma textura do cubo
  - Repetir para as restantes 5 direcções.

OpenGL – Environment Map

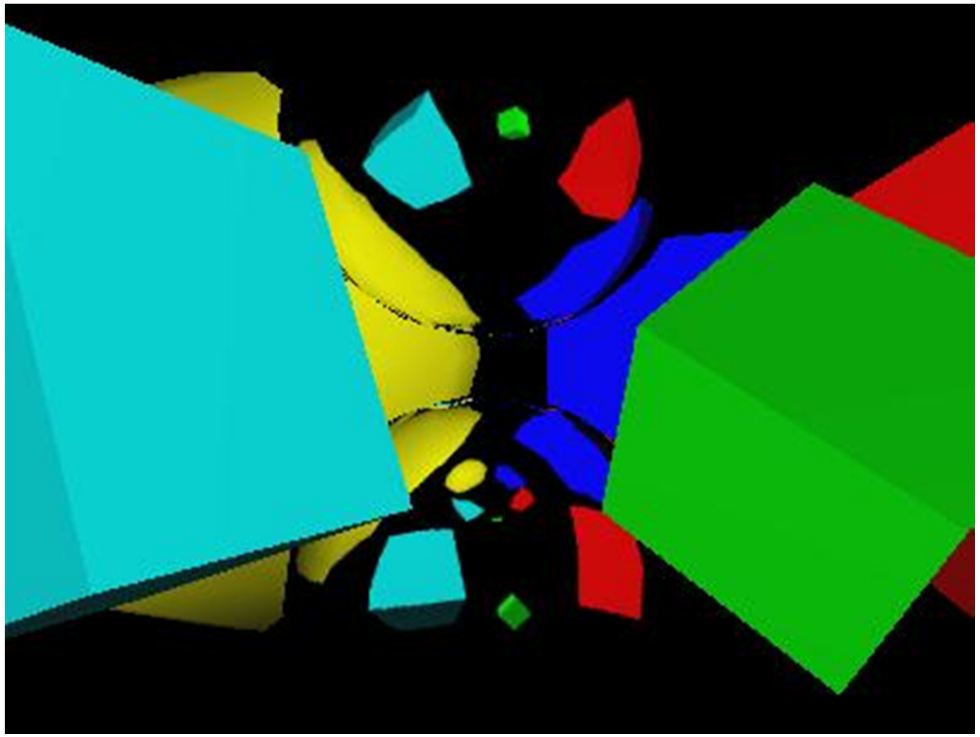
- Geração em tempo real



www.nvidia.com

OpenGL – Environment Map

- Ray Tracing Simulado



Texturas - Desempenho

- Texturas Residentes
  - glAreTexturesResident

  residentes é um array de booleanos que indica se cada uma das texturas alocadas está residente

- Prioridades
  - glPrioritizeTextures

Referências

- **OpenGL Programming Guide**, aka Red Book, OpenGL ARB