

# Texturas

## Computação Gráfica

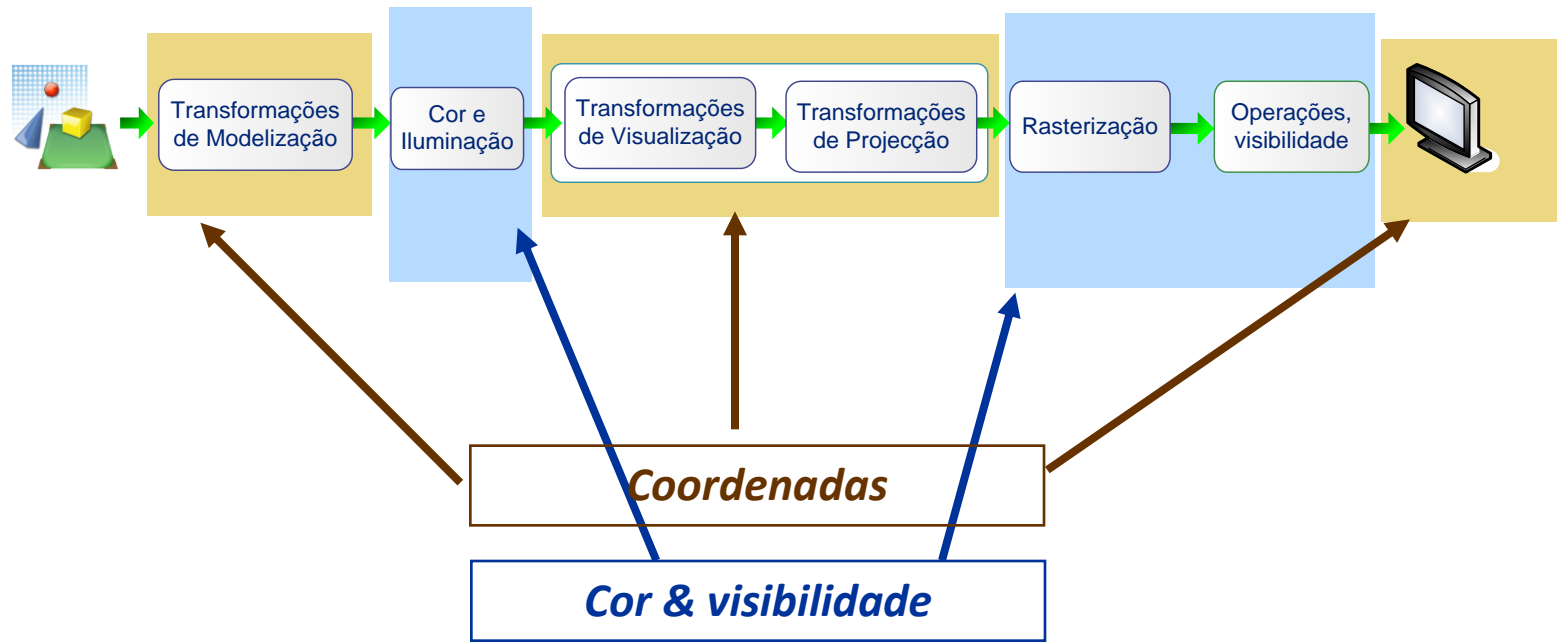
*Jorge Henriques*

*Andre Perrotta*



## OpenGL

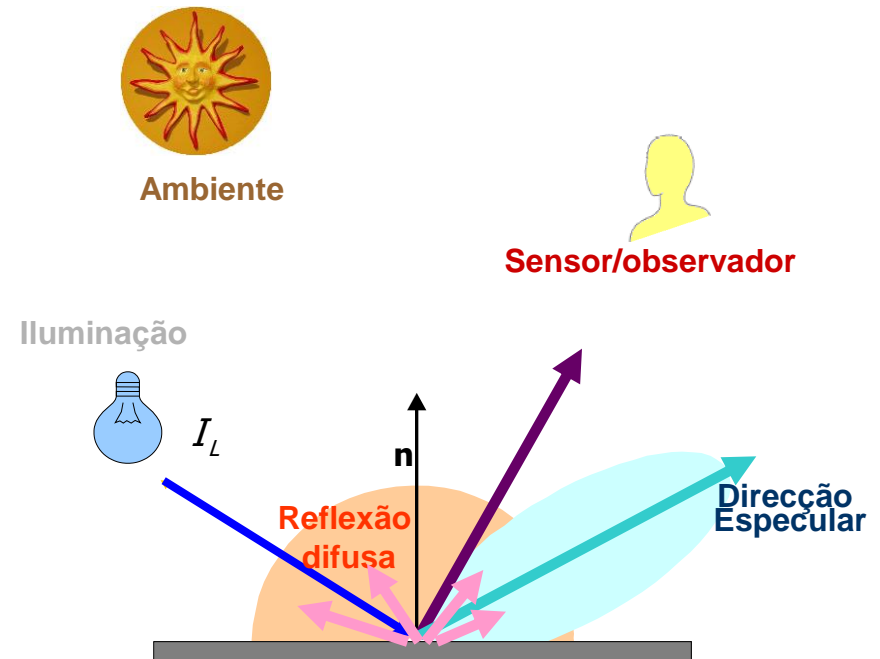
■ 3D → 2D



## OpenGL

### ■ Cor: duas abordagens !

#### ■ 1. Modelos de cor e iluminação



#### ■ 2. Texturas



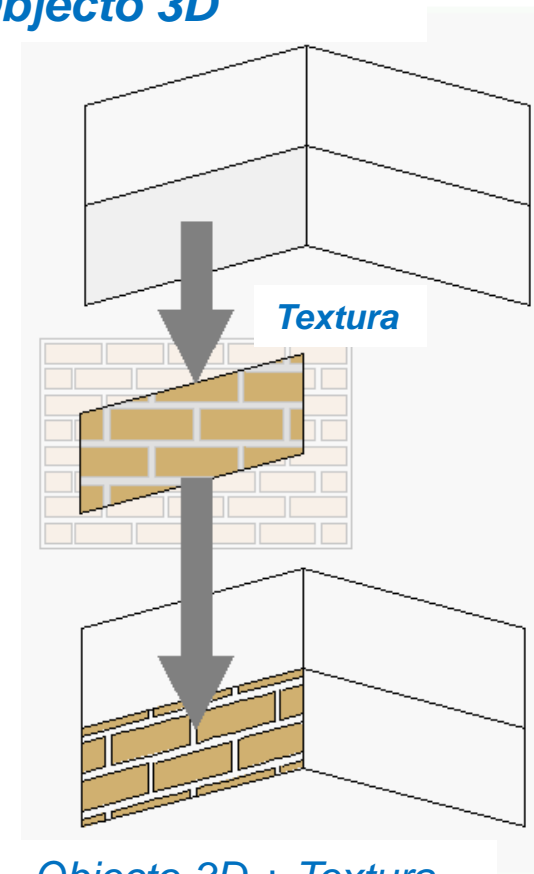
***Aula de hoje***  
**!**

## OpenGL

■ No contexto de Computação Gráfica, o mapeamento de texturas consiste na aplicação de uma imagem sobre as faces de um objecto 3D.

- Uma textura é uma imagem rectangular.
- Um pixel de textura é denominado de texel.

*Objecto 3D*



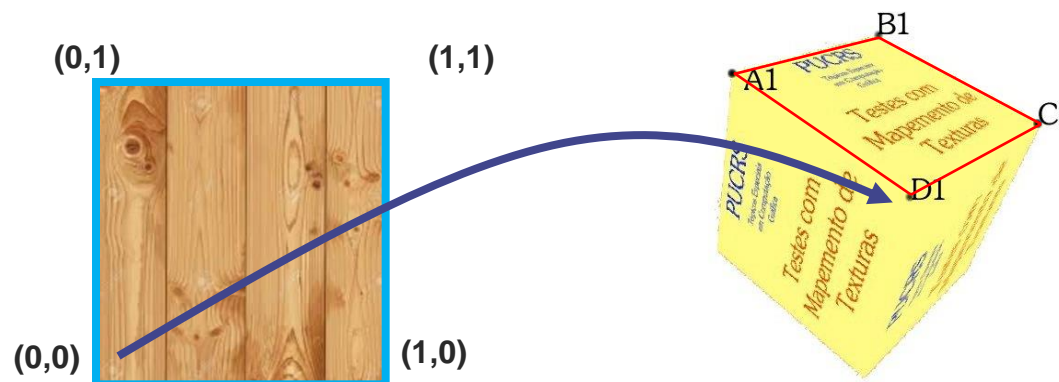
*Objecto 3D + Textura*

## OpenGL

### *Ideia fundamental*

#### Mapeamento /correspondência de pixels 2D / vértices 3D

- 2D - a imagem (textura)
- 3D - o objecto



```
glBegin(GL_QUADS);
```

```
    glTexCoord2f ( 0, 0 );
```

```
    glTexCoord2f ( 1, 0 );
```

```
    glTexCoord2f ( 0, 1 );
```

```
    glTexCoord2f ( 1, 1 );
```

```
glEnd();
```

```
    glVertex3f ( Dx, Dy, Dz );
```

```
    glVertex3f ( Cx, Cy, Cz );
```

```
    glVertex3f ( Bx, By, Bz );
```

```
    glVertex3f ( Ax, Ay, Az );
```



## OpenGL

- Em OpenGL existe o conceito de "***objectos textura***", geridos pelo próprio OpenGL.
  - É da responsabilidade do utilizador carregar as texturas a partir de ficheiros de imagem
  - Proceder à criação de um objecto textura.
  - O OpenGL gera um identificador de textura para uso posterior.
- Em OpenGL podem ser criadas e coexistir várias texturas, existindo no entanto a noção de "***textura activa***"
  - A definição das propriedades é relativa à textura activa.

## Texturas - OpenGL

- 1. Activar o modo de textura
  - `glEnable(GL_TEXTURE_2D) ---- glDisable(GL_TEXTURE_2D)`

*criar*

- 2. Criar o identificador de uma textura
  - `GluInt texture;`
  - `glGenTextures(1, &texture);`

- 3. Especificar a forma de mapeamento (combinação da imagem e textura)
  - `glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);`

*propriedades*

- 4. Especificar parâmetros da textura (filtros e repetições)
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);`
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);`
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);`
  - `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);`

- 5. Construir a textura
  - `glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, imag.GetNumCols(),  
imag.GetNumRows(), 0, GL_RGB, GL_UNSIGNED_BYTE, imag.ImageData() );`

*Imagem*

- 6. *Activação do Objecto Textura*
  - `glBindTexture(GL_TEXTURE_2D, texture);`

- 7. Aplicação da Textura

## OpenGL

- 0. Activar o modo de textura
  - `glEnable(GL_TEXTURE_2D)`
  
- 1. Gerar o **identificador** de uma textura
  - `Gluint texture;`
  - `glGenTextures(1, &texture);` //endereço textura (imagem bmp)
  
- 2. **Criar** um objecto textura (neste **caso 2D**) associando-o ao identificador anteriormente criado
  - `glBindTexture(GL_TEXTURE_2D, texture);` //textura 2D



## OpenGL

- 3. Especificar a **forma de mapeamento** (combinação da imagem e textura)

```
glTexEnvf ( (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, TIPO );
```

**TIPO:**

- GL\_DECAL
- GL\_MODULATE
- GL\_BLEND

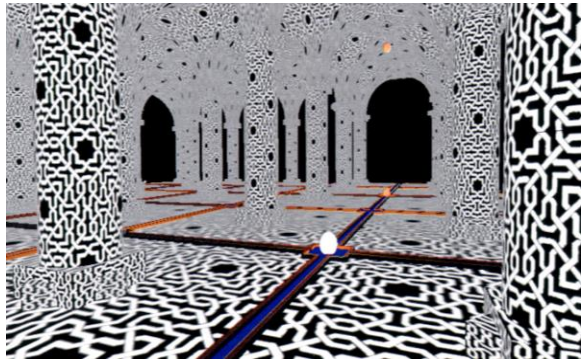


## OpenGL

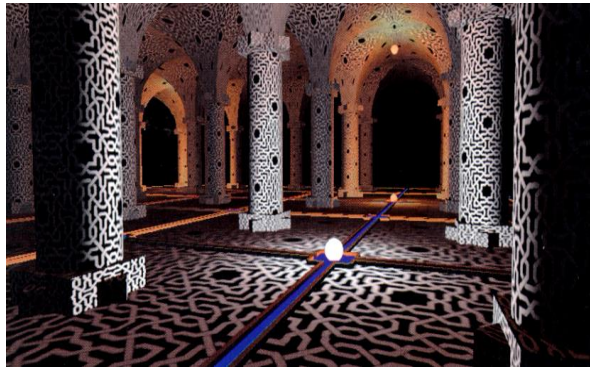
*Original*



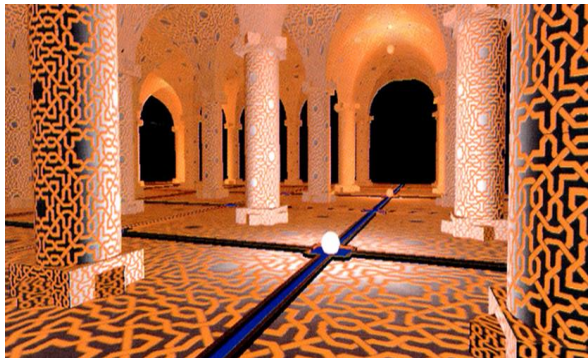
*Decal*



*Modulate*



*Blending*



## OpenGL

### ■ 4. Especificar parâmetros da textura (repetições e filtros)

#### ■ 4.1- Repetições

■ O OpenGL permite distinguir as direcções horizontais e verticais.

- Caso de deseje efectuar a repetição (**REPEAT**)
- Caso não se deseje efectuar a repetição (**CLAMP**)
- Direcção horizontal (**S**)
- Direcção vertical (**T**)

**glTexParameter(GL\_TEXTURE\_2D, direcção, tipoRepetição);**

#### ■ *Direcção*

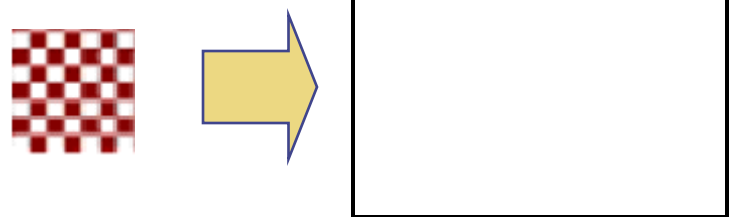
- GL\_TEXTURE\_WRAP\_S
- GL\_TEXTURE\_WRAP\_T

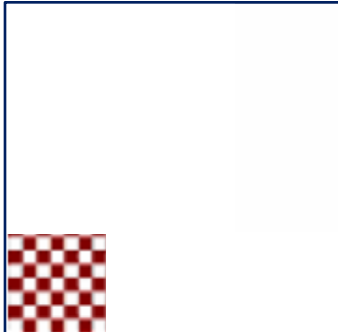
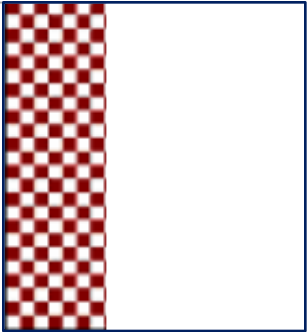


#### ■ *tipoRepetição*

- GL\_CLAMP
- GL\_REPEAT



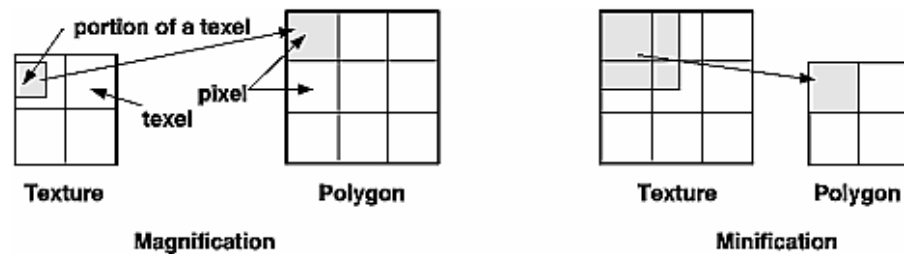
## OpenGL



	
<div>Wrap S : GL_CLAMP Wrap T : GL_CLAMP</div>	<div>Wrap S : GL_CLAMP Wrap T : GL_REPEAT</div>
	
<div>Wrap S : GL_REPEAT Wrap T : GL_CLAMP</div>	<div>Wrap S : GL_REPEAT Wrap T : GL_REPEAT</div>

## OpenGL

### ■ 4.2 Filtros



`glTexParameteri(GL_TEXTURE_2D, Operação, tipoFiltro);`

#### ■ Operação

- `GL_TEXTURE_MAG_FILTER`
- `GL_TEXTURE_MIN_FILTER`

#### ■ tipoFiltro

- `GL_NEAREST`
- `GL_LINEAR`
- ...



# Interpolação



Nearest Neighbor



Linear





## OpenGL

### ■ 5. Finalmente *construir a textura*

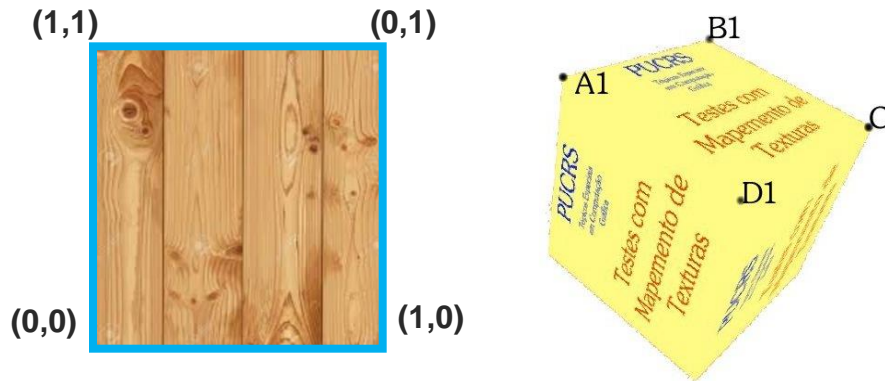
- `glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, imag.GetNumCols(), imag.GetNumRows(), 0, GL_RGB, GL_UNSIGNED_BYTE, imag.ImageData());`
- **Tipo=GL\_TEXTURE\_2D** , pretende-se criar uma textura 2D,
- **Level=0**, significa uma única imagem de textura. Note-se que podem existir várias e, neste contexto, usa-se a técnica de *mip-mapping* para obter níveis de detalhe diferente nas texturas
- **Internalformat= GL\_RGBA**, descreve a representação interna da textura (em OpenGL).
- **Width**, Tamanho (largura) da imagem em pixels
- **Height**, Tamanho (altura) da imagem em pixels
- **Border=0**, indica se desejamos uma borda de 0 (sem borda) ou mais pixels.
- **Format=GL\_RGB**, correspondem ao formato original da imagem
- **Type= GL\_UNSIGNED\_BYTE**, tipo de dados usados no formato da imagem
- **Pixels=image**, corresponde ao array (bidimensional, neste caso) de imagem propriamente dito.

## OpenGL

### 6. Activar a textura (Bind)

Para a aplicação da textura é necessário

- Activar a textura desejada
- Especificar a relação entre os vértices da textura e os vértices dos polígonos sobre os quais se desenha mapear a textura escolhida.



```
glBindTexture(GL_TEXTURE_2D, texture); //Activar textura desejada
glBegin(GL_QUADS);
    glTexCoord2f ( 0, 0 );   glVertex3f ( Dx, Dy, Dz );
    glTexCoord2f ( 1, 0 );   glVertex3f ( Cx, Cy, Cz );
    glTexCoord2f ( 0, 1 );   glVertex3f ( Bx, By, Bz );
    glTexCoord2f ( 1, 1 );   glVertex3f ( Ax, Ay, Az );
glEnd();
```



## Imagem -> textura

### RgbImage

- Atribuir uma imagem **bmp** a uma textura
- **Uso da biblioteca RgbImage** (disponibilizada na infordocente)
  - RgbImage.cpp
  - RgbImage.h
- **Atenção**
  - Imagens bmp 24 bits
  - Dimensão: 128x128, 256x256, 512x512,
  - O projecto tem ser c++
  - **fopen** versus **fopen\_s**

## *fopen vs fopen\_s*

```
FILE* infile = fopen (filename, "rb");// Open for reading binary data
if (!infile) {
    fprintf(stderr, "Unable to open file: %s\n", filename);
    ErrorCode = OpenError;
    return false;
}
```

```
FILE* infile;
if ( fopen_s (&infile, filename, "rb")) {
    fprintf(stderr, "Unable to open file: %s\n", filename);
    ErrorCode = OpenError;
    return false;
}
```



RgbImage imag;

■ GLuint texture[2]; // 0-imagem1 , 1-imagem2, 2-imagem3 // lista de texturas

//----- Criar cada uma das texturas 0, 1, 2-----

```
glGenTextures(1, &texture[0]);  
glBindTexture(GL_TEXTURE_2D, texture[0]);  
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_DECAL);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);  
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);  
imag.LoadBmpFile("imagem1.bmp");  
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA,  
             imag.GetNumCols(),  
             imag.GetNumRows(), 0, GL_RGB, GL_UNSIGNED_BYTE,  
             imag.ImageData());
```

## Usar a textura [0]

# Polígonos

//.....Activar textura

```
glBindTexture (GL_TEXTURE_2D, texture[0] );
```

//.....Desenhar poligono + Textura

```
glPushMatrix();
```

```
    glTranslatef( x, y, z);
```

```
    glBegin(GL_QUADS);
```

```
        glTexCoord2f (0.0f, 0.0f);
```

```
        glTexCoord2f (1.0f, 0.0f);
```

```
        glTexCoord2f (1.0f, 1.0f);
```

```
        glTexCoord2f (0.0f, 1.0f);
```

```
    glEnd();
```

```
glPopMatrix();
```

```
        glVertex3f ( quad, 0, 0);
```

```
        glVertex3f ( 0, 0, 0);
```

```
        glVertex3f ( 0, quad, 0);
```

```
        glVertex3f ( quad, quad, 0);
```



## Usar a textura [1]

# GLUT

**CHALEIRA** - mapeamento implementado internamente na GLUT  
**Note-se que é o único objecto da GLUT que permite textura !!**

```
//..... Activar textura
glBindTexture (GL_TEXTURE_2D, texture[1] );

//..... Desenha chaleira+Textura
glPushMatrix();
    glTranslatef( a, b, c);
    glutSolidTeapot(bule);
glPopMatrix();
```

## Usar a textura [2]

### Esfera - quádrlica

```
// ..... Activar textura
glBindTexture (GL_TEXTURE_2D, texture[2] );
```

```
// ..... Desenhar esfera + Textura
glPushMatrix();
    GLUquadricObj* q = gluNewQuadric ( );
    gluQuadricDrawStyle ( q, GLU_FILL );
    gluQuadricNormals ( q, GLU_SMOOTH );
    gluQuadricTexture ( q, GL_TRUE );
    gluSphere ( q, raioEsfera, 100, 100);
    gluDeleteQuadric ( q );
glPopMatrix();
```



# Vertex Array

## 1. Definir vertices de textura

```
GLfloat arrayTexture[ ]={ 0,0, 1,0, 1,1, 0,1, // mapeamento da textura em cada vertice (A,B,C,D)
                          0,0, 1,0, 1,1, 0,1,
                          .... }
```

## 2. Activar vertices de textura

```
glVertexPointer(3, GL_FLOAT, 0, vertices);
glEnableClientState(GL_VERTEX_ARRAY);
glNormalPointer(GL_FLOAT, 0, normais);
glEnableClientState(GL_NORMAL_ARRAY);
glColorPointer(3, GL_FLOAT, 0, cores);
glEnableClientState(GL_COLOR_ARRAY);
glTexCoordPointer(2, GL_FLOAT, 0, arrayTexture);
glEnableClientState(GL_TEXTURE_COORD_ARRAY);
```

## 3. Desenhar poligono normalmente

```
glBindTexture(GL_TEXTURE_2D, texture[2] ); // mesma imagem da esfera
glDrawElements(GL_POLYGON, 4, GL_UNSIGNED_INT, poligono);
```



## Trabalho

- **1. Código disponibilizado**
- **2. Adaptar ao projecto**
  - Claro que podem ser introduzidos novos elementos/técnicas de texturas





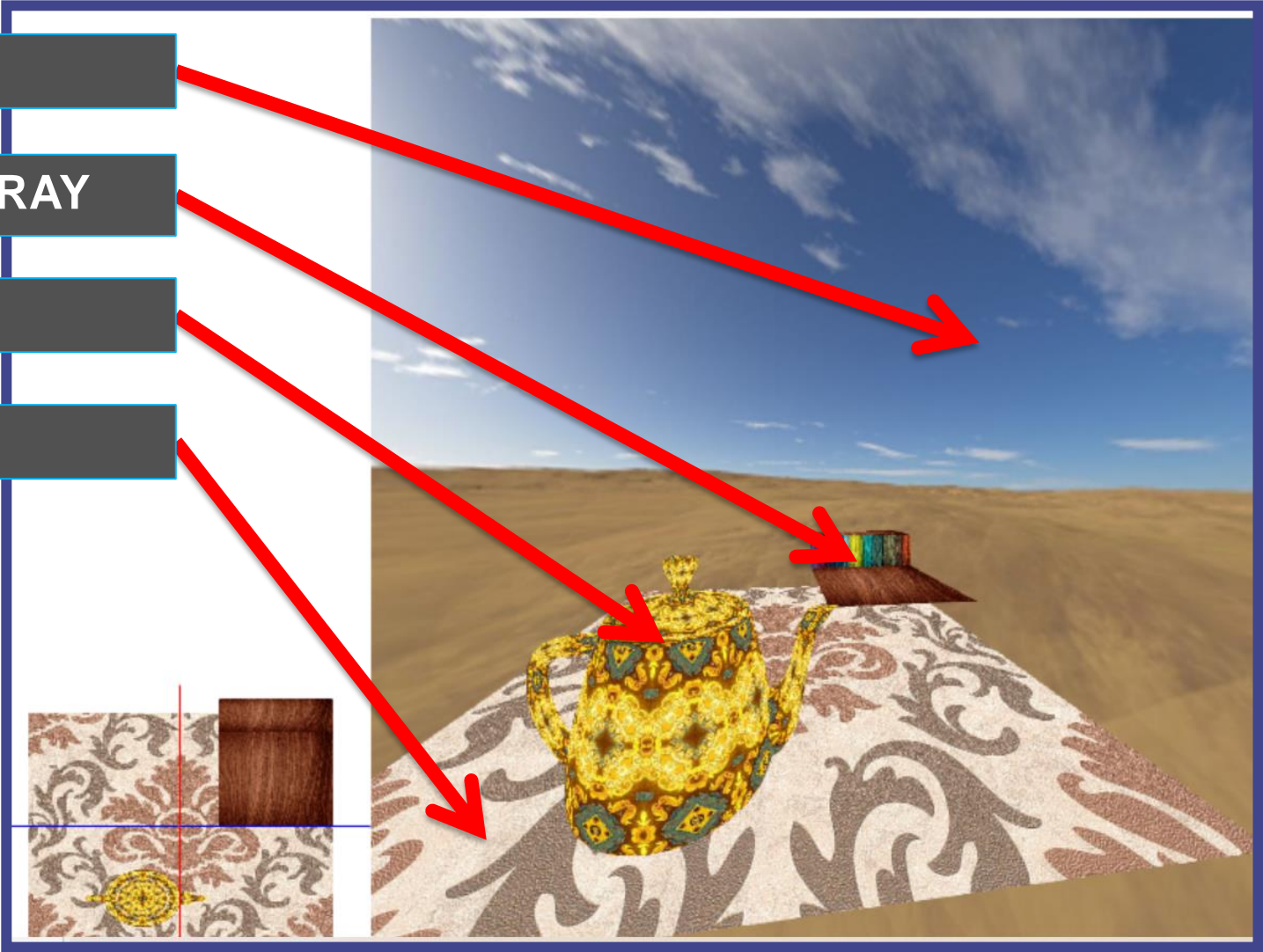
## Trabalho – código disponibilizado

Sky box

VERTEX ARRAY

GLUT

Poligono





## Trabalho – Adaptar código fornecido

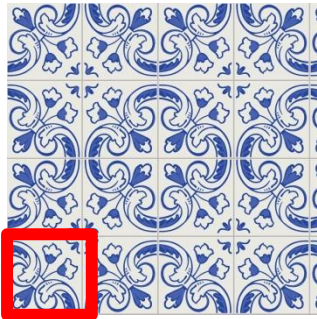
### ■ 1. Comando/botões

- Poligonos ?
- Vertex Array ?
- Sky box ?
- É necessário ?
- Qual o mais adequado?

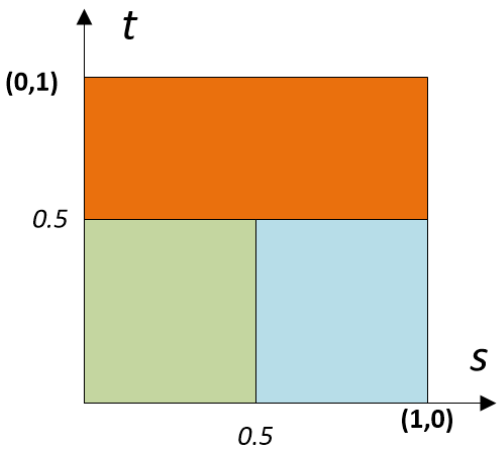
## Trabalho - Adaptar

### ■ 2. Propriedades

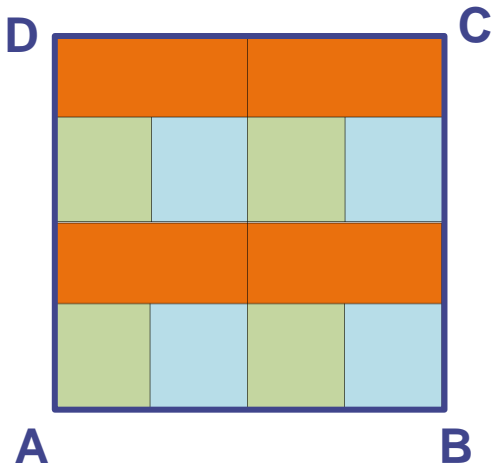
■ Repetir uma textura ?



■ É necessário ?



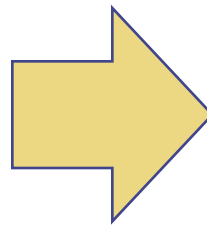
A  $\rightarrow$  (0.0, 0.0)  
B  $\rightarrow$  (2.0, 0.0)  
C  $\rightarrow$  (2.0, 2.0)  
D  $\rightarrow$  (0.0, 2.0)

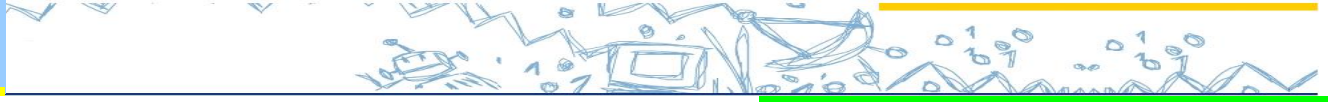


## Trabalho - Adaptar

### ■ 3. Mapeamento correto ?

- Nem sempre se deve aplicar toda a textura





## Trabalho - Adaptar

### ■ 4. Outras técnicas ?

- Light mapping?
- Bump mapping ?

## Observador

Posição não é fixa.

Observador movimenta-se na cena (*first person*)

`gllookat(obsP[0], obsP[1], obsP[2], dest[0],dest[1],dest[2], 0,1,0)`

