

TEXTURAS

O uso de texturas pode eliminar a necessidade de uma geometria mais pormenorizada



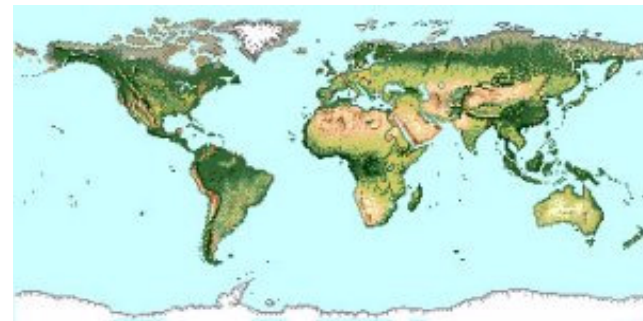
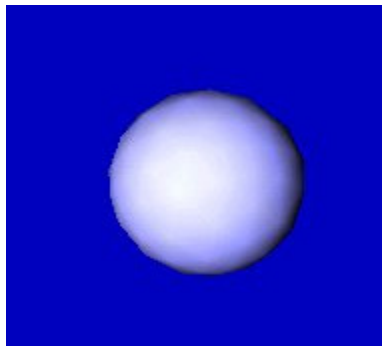
S/ texturas



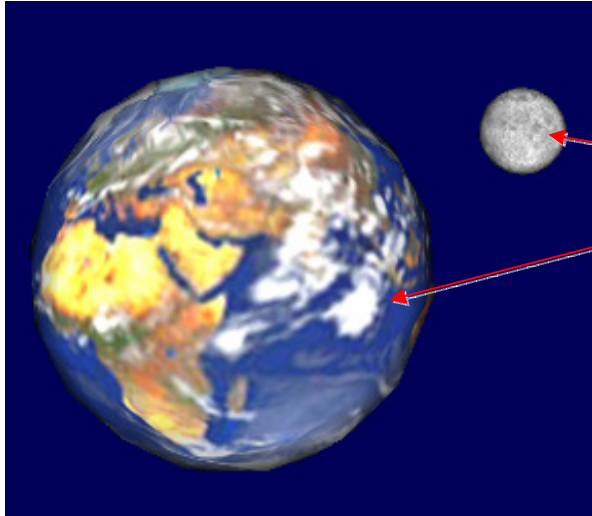
C/ texturas

Mapeamento de Texturas

(TEXTURE MAPPING)



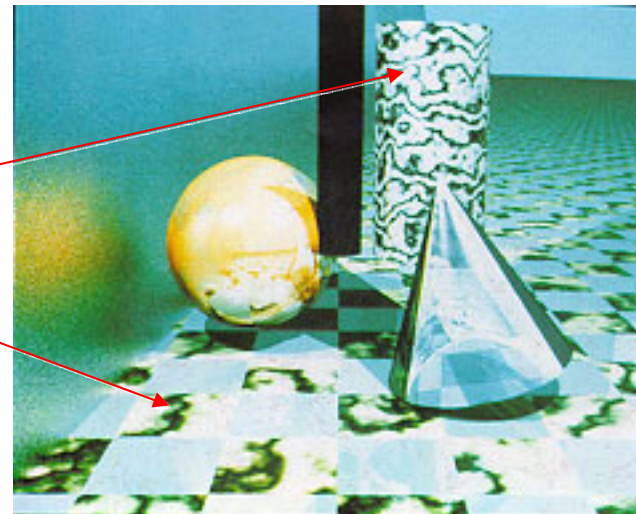
OBTENÇÃO DAS TEXTURAS



Uma textura pode ser armazenada num quadro, quer proveniente de imagem digital...

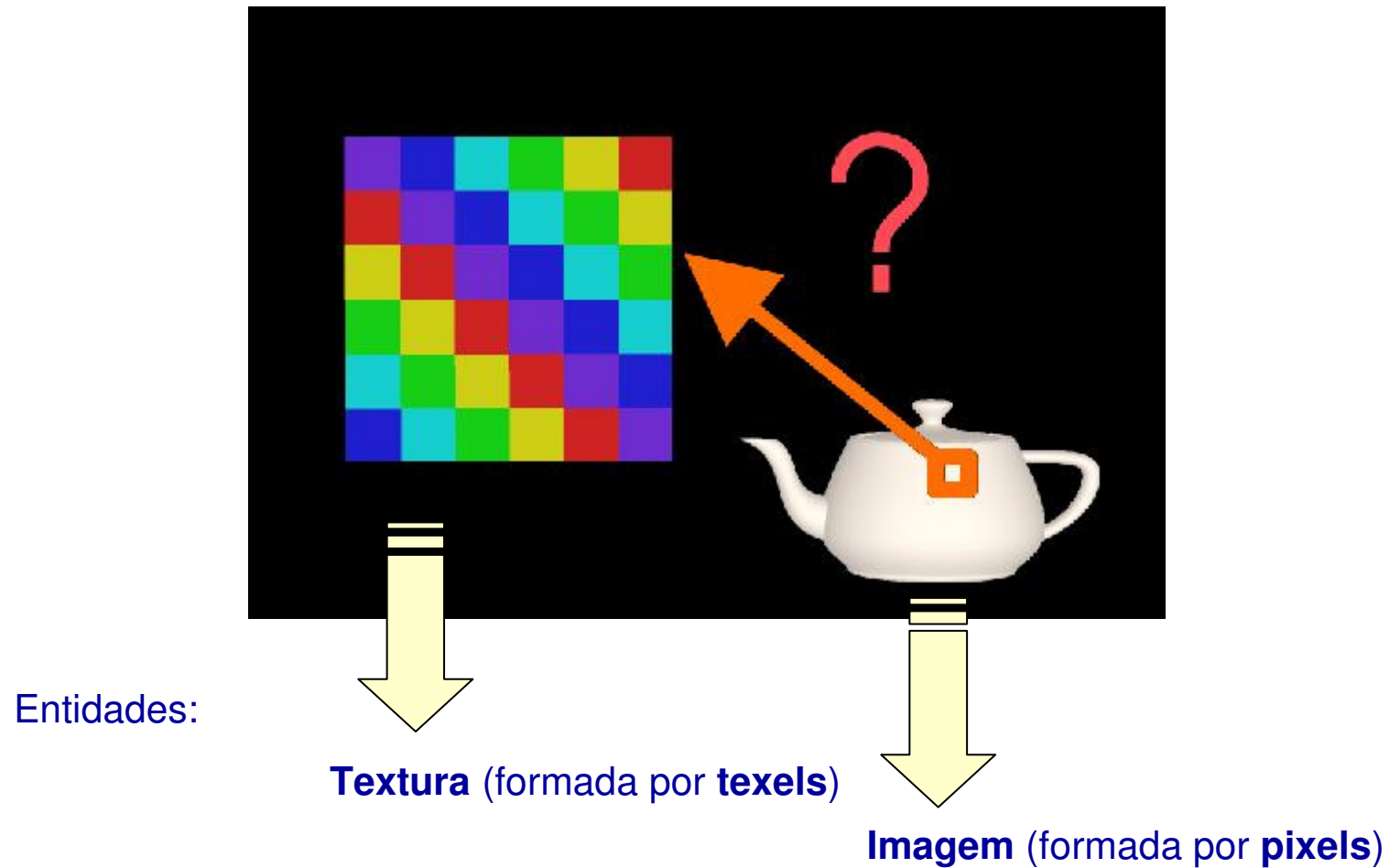
Para se utilizar um quadro como textura, em OpenGL:
`glTexImage2D()`

... quer calculada por um procedimento em tempo de execução



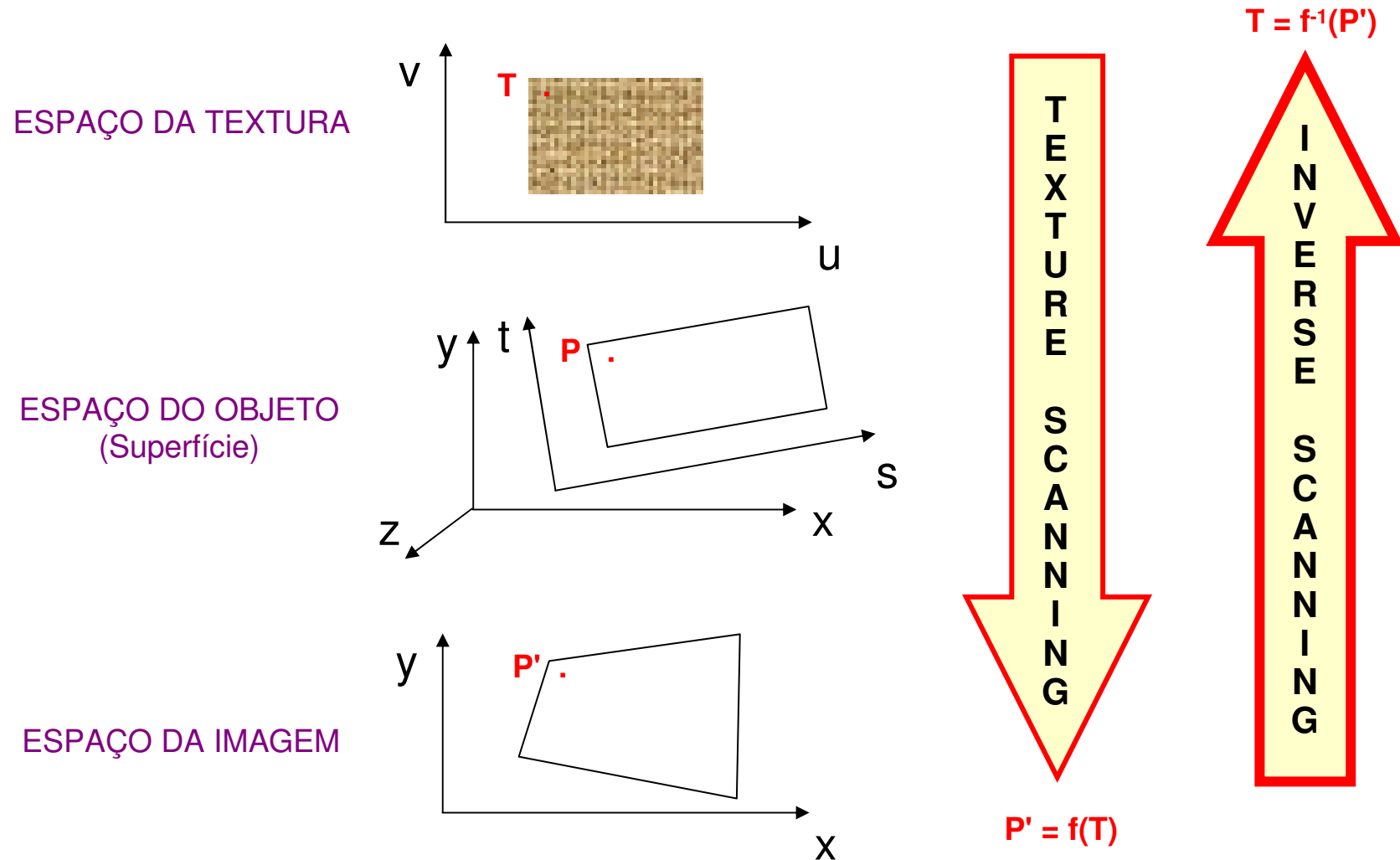
TEXTURAS

O problema da correspondência no mapeamento



MAPEAMENTO DE UMA TEXTURA

Tratamento Matemático



scanning $\equiv$ varrimento

COMPARAÇÃO DOS MÉTODOS RELATIVOS À ORDEM DE EXECUÇÃO

TEXTURE SCANNING (varrimento da textura)

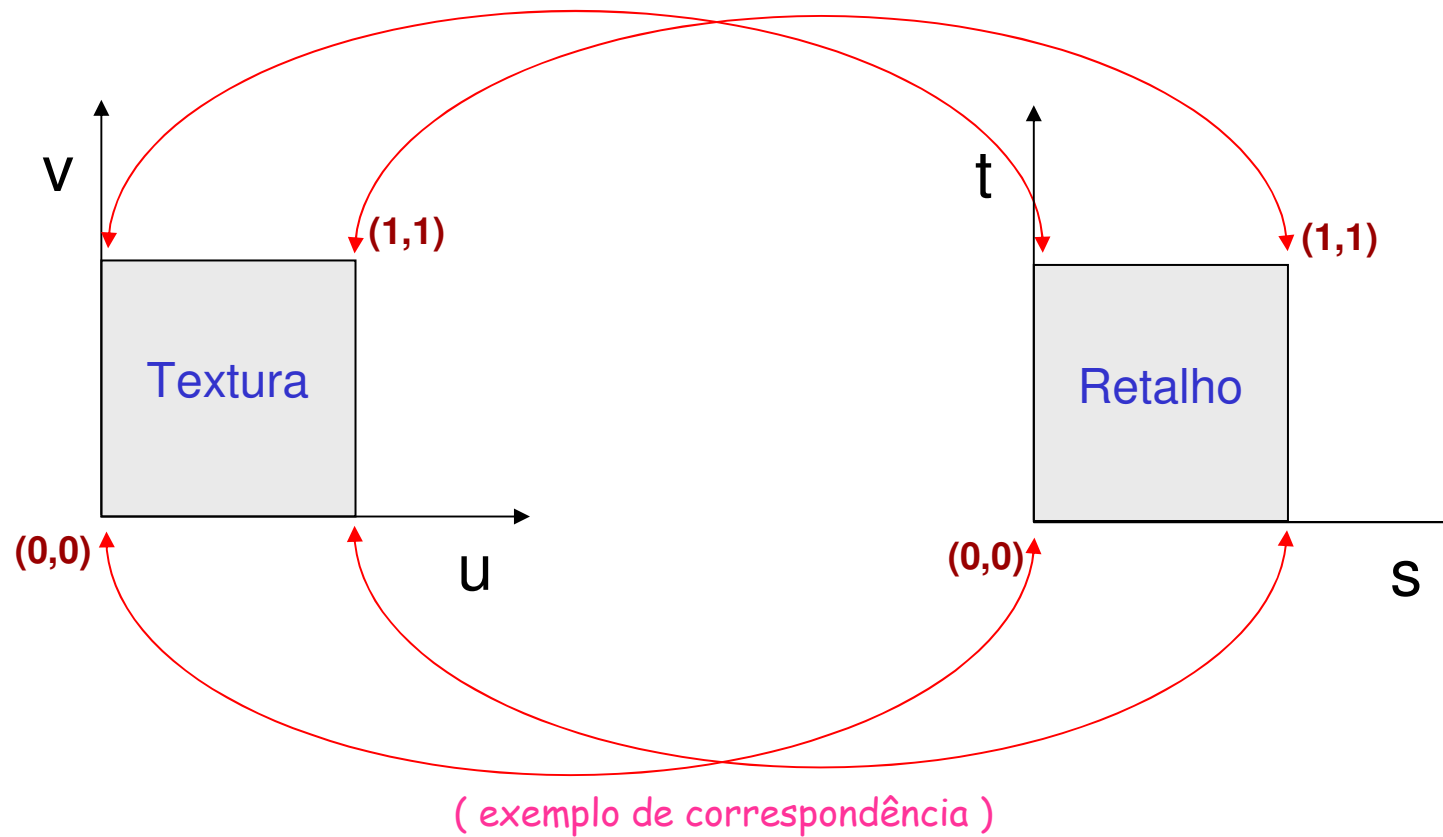
- ☹️ Em geral a textura não coincide com um número inteiro de pixels, implicando cálculos para a subdivisão da área de um pixel.
- 😊 As transformações de visualização seguem-se no sentido mais natural.

INVERSE SCANNING (varrimento inverso)

- ☹️ Exige o cálculo das transformações de visualização inversas.
- 😊 Evita a subdivisão de pixels e permite, além disso, o uso de filtros para *antialiasing* (matéria posterior do curso).

Conclusão: o método usualmente preferido é o de INVERSE SCANNING.

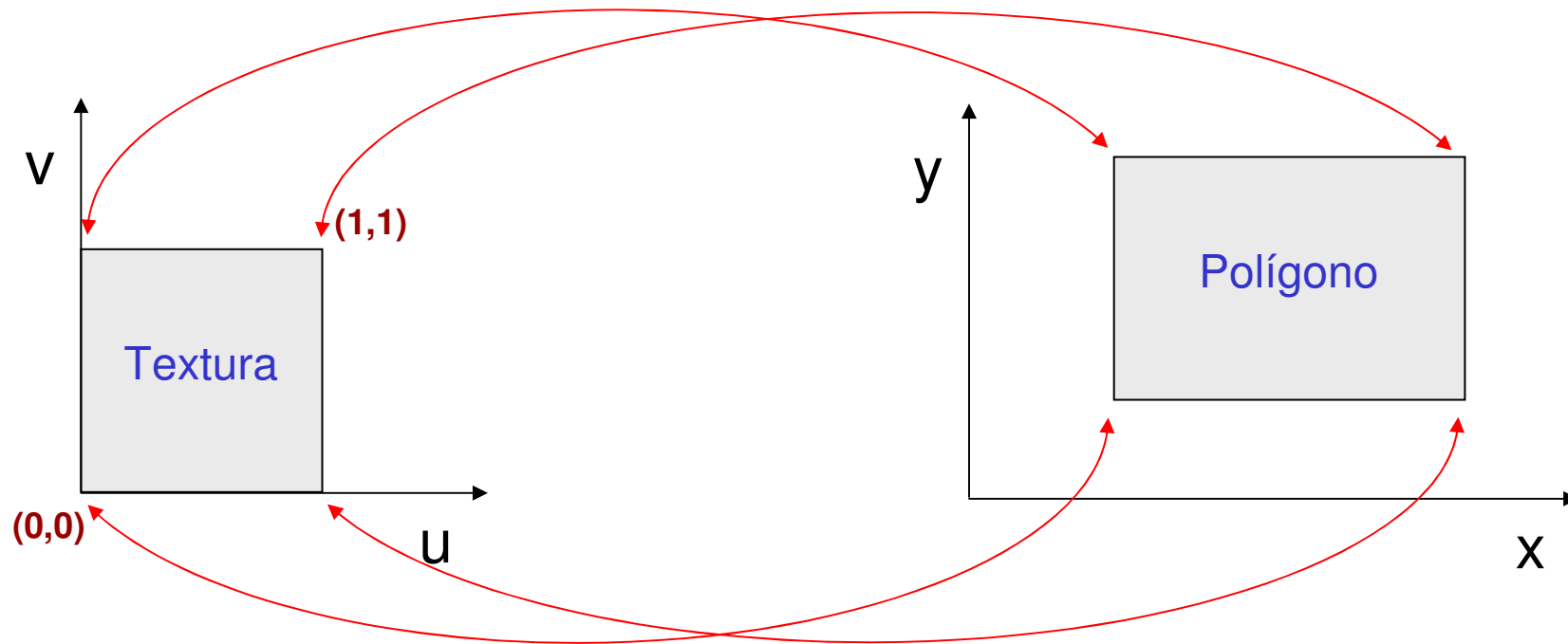
SUPERFÍCIE DEFINIDA PARAMETRICAMENTE



É habitual fazer-se com que este tipo de correspondência seja uma interpolação linear:

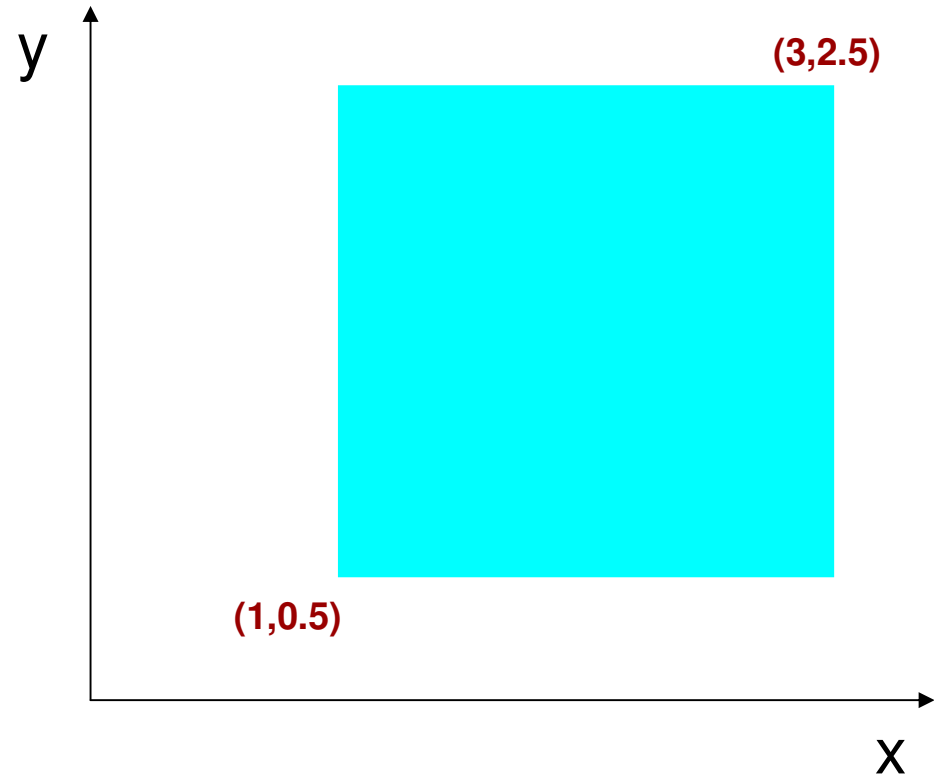
$$\begin{bmatrix} s \\ t \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

SUPERFÍCIE DEFINIDA POR UM POLÍGONO



Também é habitual que a correspondência seja uma interpolação linear
(estar sempre ciente das possíveis distorções!)

Exemplos com OpenGL

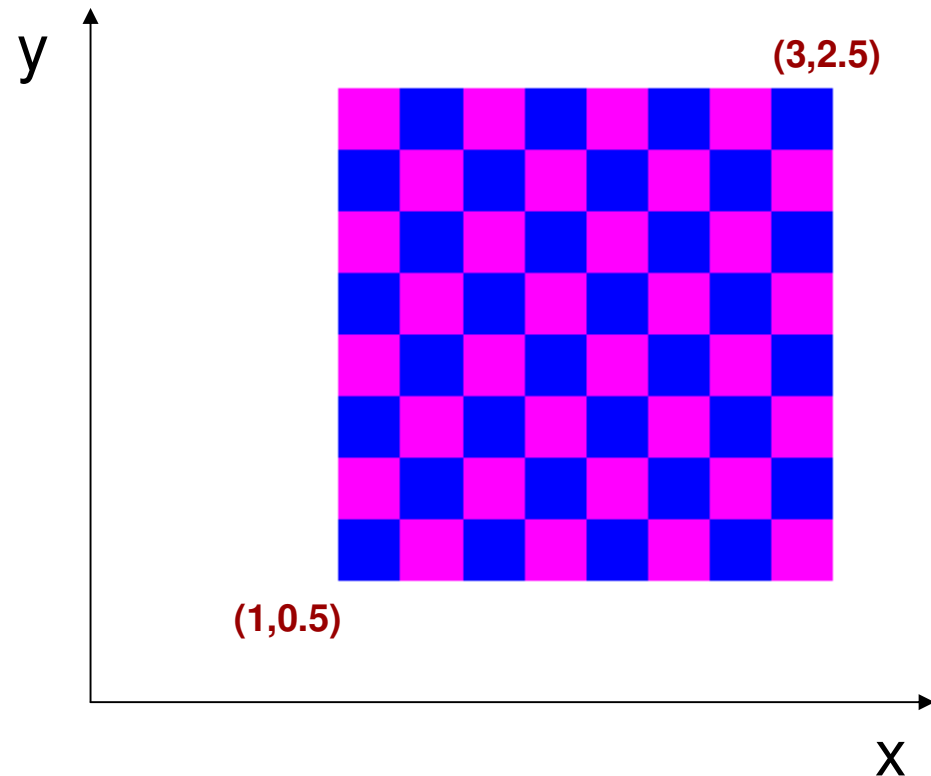
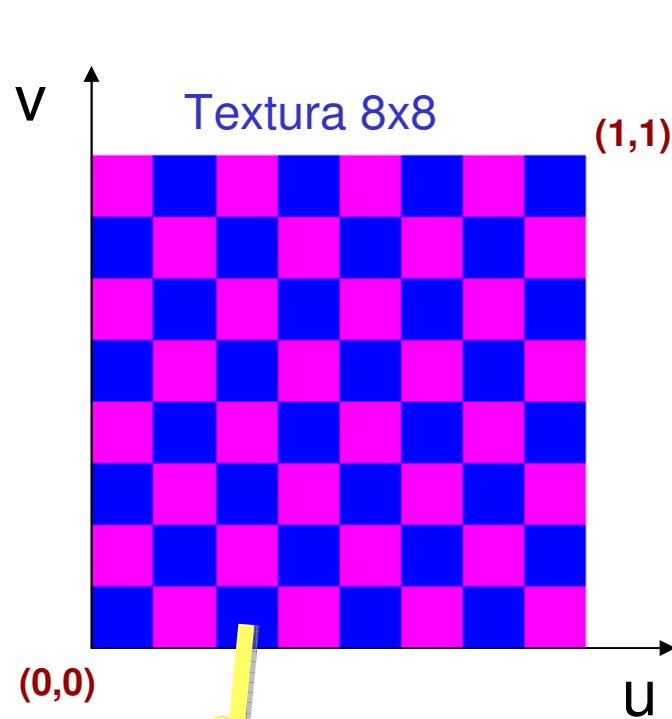


```
glColor3d(0.0, 1.0, 1.0);  
glBegin(GL_QUADS);  
    glVertex3d(1.0, 0.5, 0.0);  
    glVertex3d(1.0, 2.5, 0.0);  
    glVertex3d(3.0, 2.5, 0.0);  
    glVertex3d(3.0, 0.5, 0.0);  
glEnd();
```

Nos exemplos seguintes, com texturas,
presume-se já existir antes a instrução

```
glTexEnvf(GL_TEXTURE_ENV,  
          GL_TEXTURE_ENV_MODE,  
          GL_DECAL)
```

para que cada pixel seja pintado apenas
com a cor da textura.

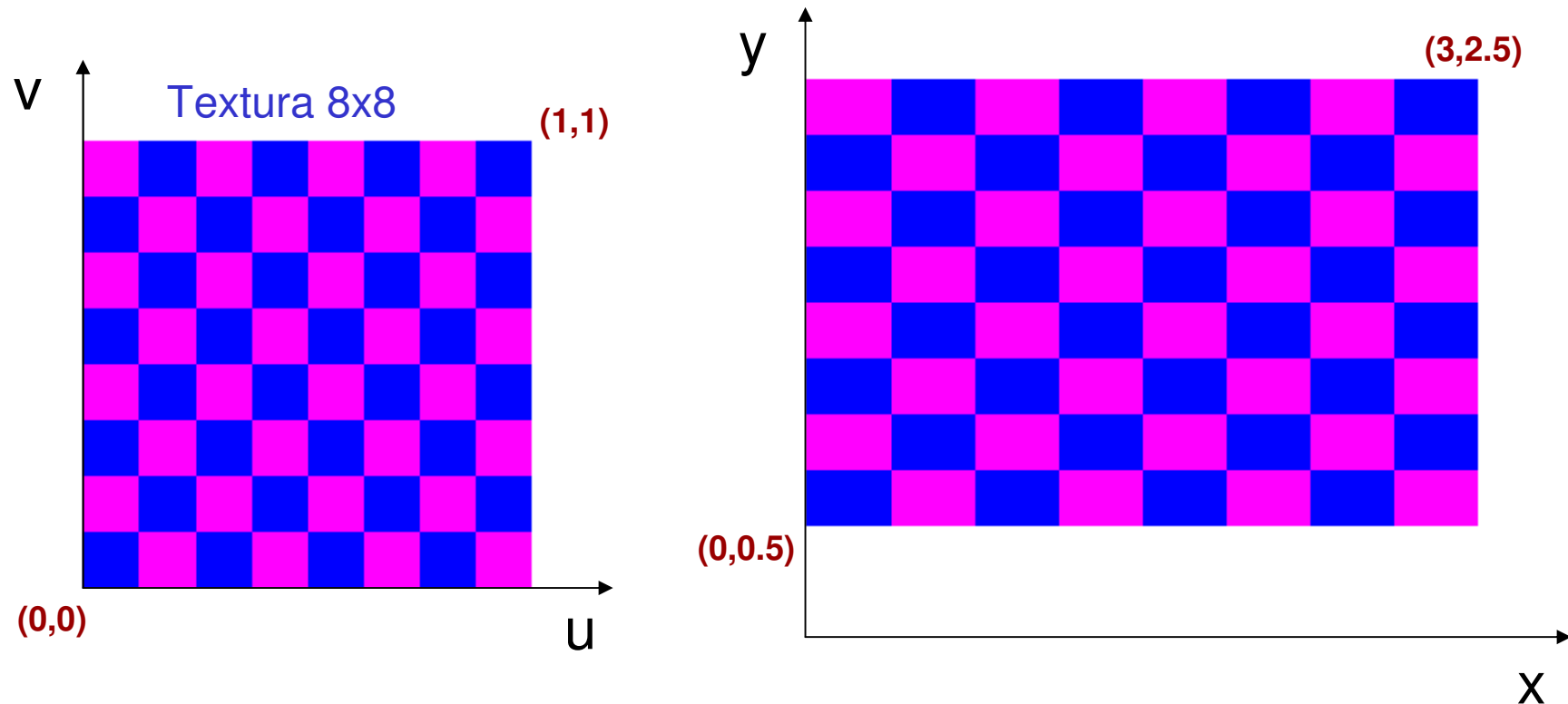


Definiu-se cada quadrícula como um único texel:

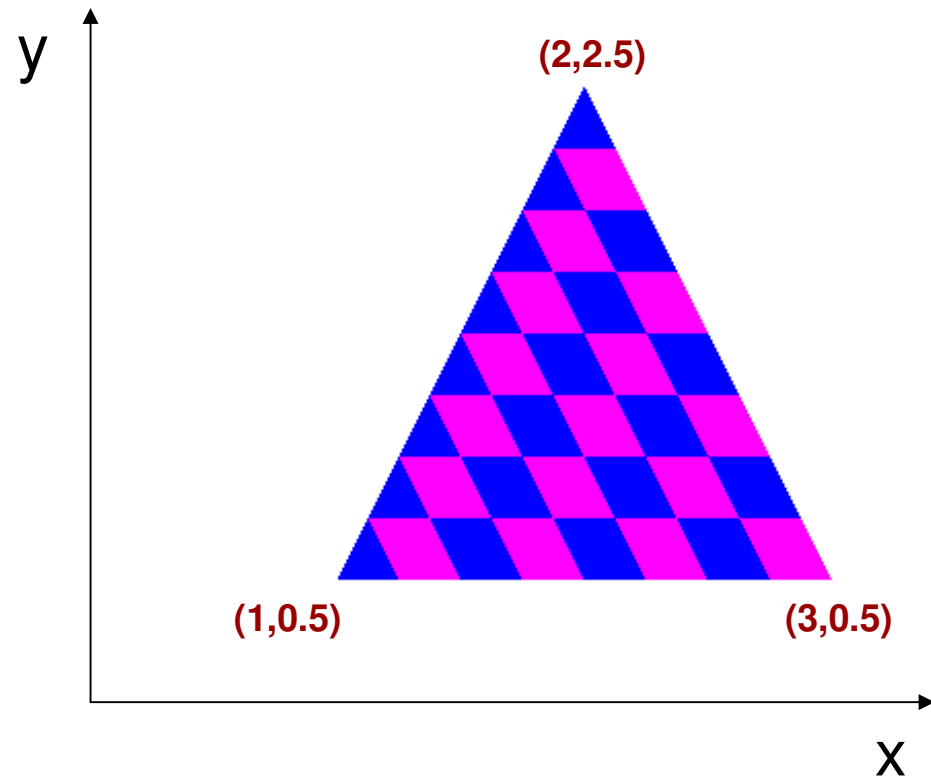
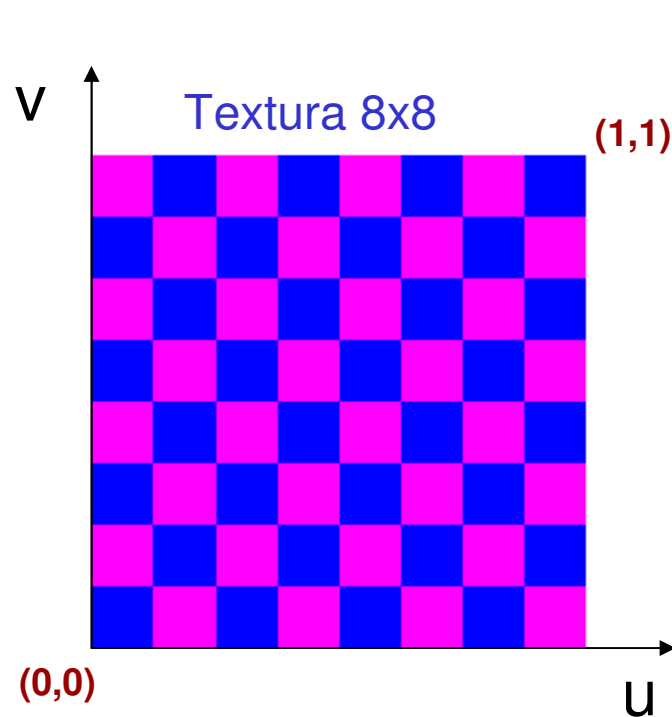
```
GLubyte textura[8][8][3];
```

Cor RGB 24 bits

```
glBegin(GL_QUADS);
    glTexCoord2d(0.0, 0.0); glVertex3d(1.0, 0.5, 0.0);
    glTexCoord2d(0.0, 1.0); glVertex3d(1.0, 2.5, 0.0);
    glTexCoord2d(1.0, 1.0); glVertex3d(3.0, 2.5, 0.0);
    glTexCoord2d(1.0, 0.0); glVertex3d(3.0, 0.5, 0.0);
glEnd();
```

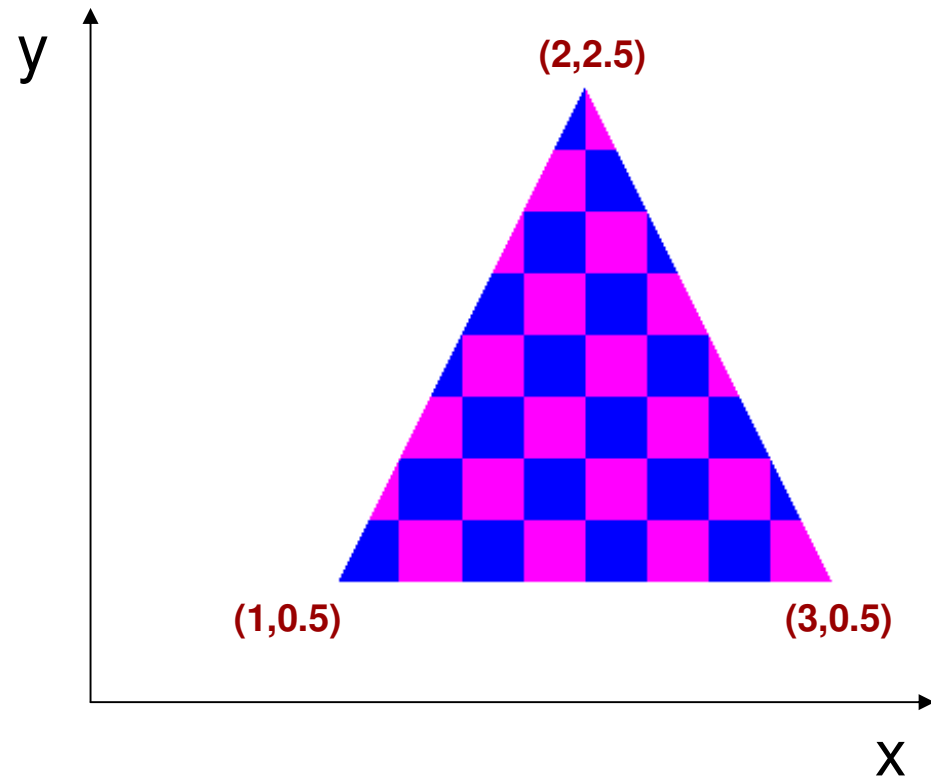
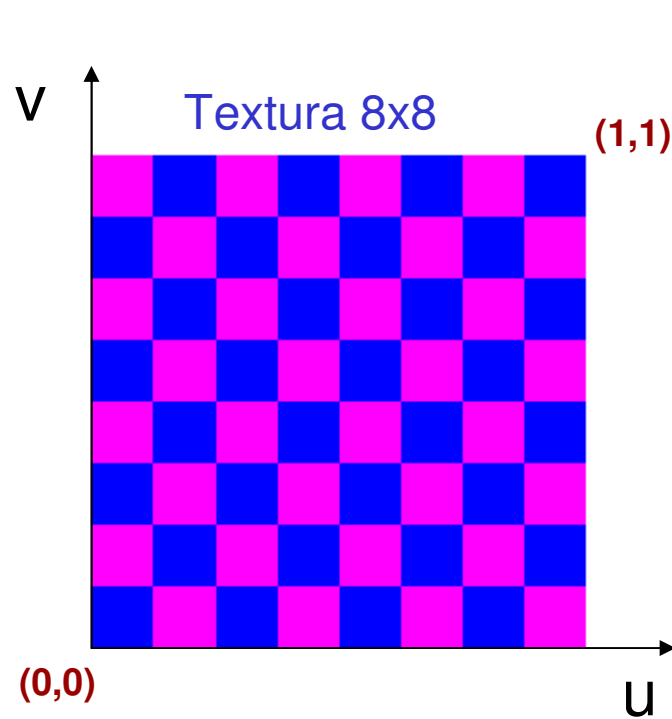


```
glBegin(GL_QUADS);  
    glTexCoord2d(0.0, 0.0); glVertex3d(0.0, 0.5, 0.0);  
    glTexCoord2d(0.0, 1.0); glVertex3d(0.0, 2.5, 0.0);  
    glTexCoord2d(1.0, 1.0); glVertex3d(3.0, 2.5, 0.0);  
    glTexCoord2d(1.0, 0.0); glVertex3d(3.0, 0.5, 0.0);  
glEnd();
```

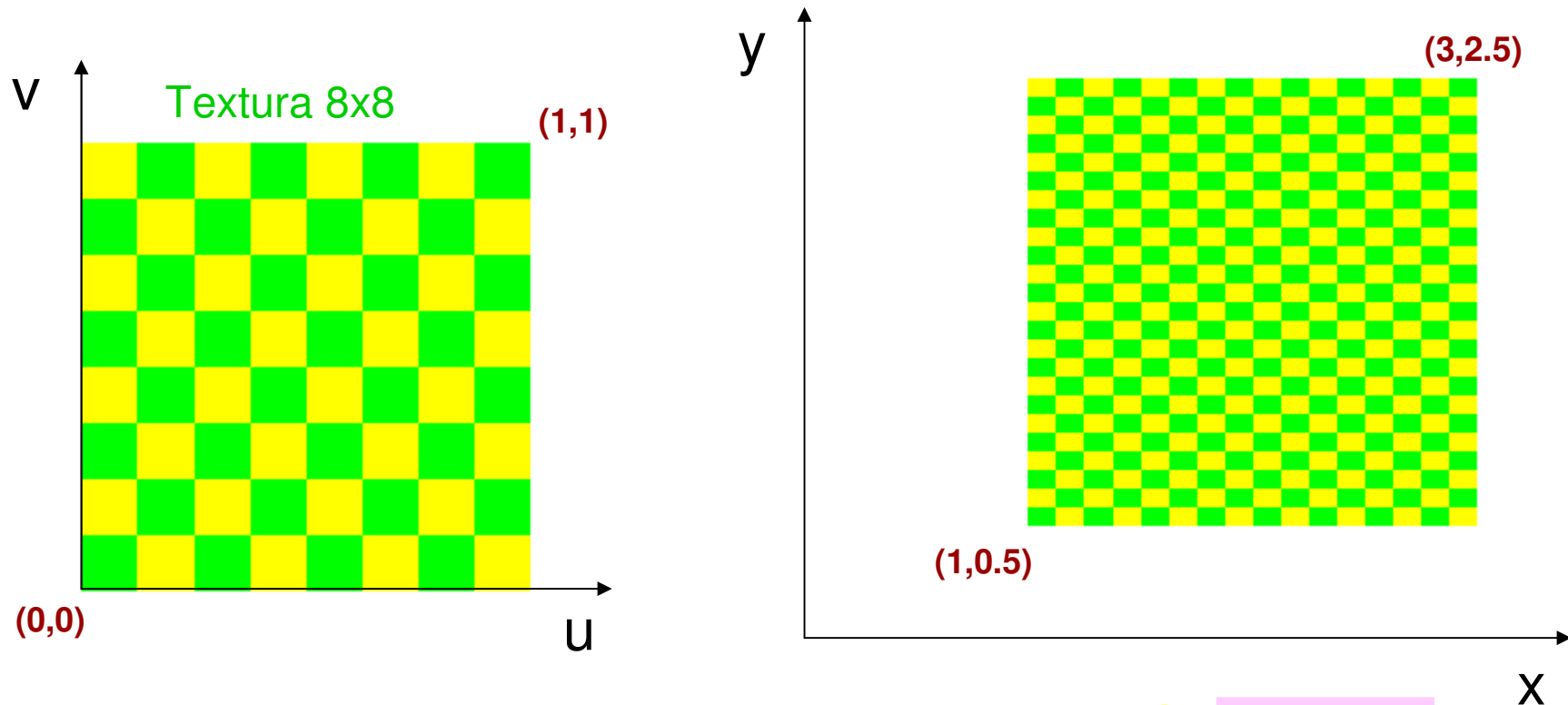


ATENÇÃO!

```
glBegin(GL_TRIANGLES);  
    glTexCoord2d(0.0, 0.0); glVertex3d(1.0, 0.5, 0.0);  
    glTexCoord2d(1.0, 1.0); glVertex3d(2.0, 2.5, 0.0);  
    glTexCoord2d(1.0, 0.0); glVertex3d(3.0, 0.5, 0.0);  
glEnd();
```



```
glBegin(GL_TRIANGLES);
    glTexCoord2d(0.0, 0.0); glVertex3d(1.0, 0.5, 0.0);
    glTexCoord2d(0.5, 1.0); glVertex3d(2.0, 2.5, 0.0);
    glTexCoord2d(1.0, 0.0); glVertex3d(3.0, 0.5, 0.0);
glEnd();
```

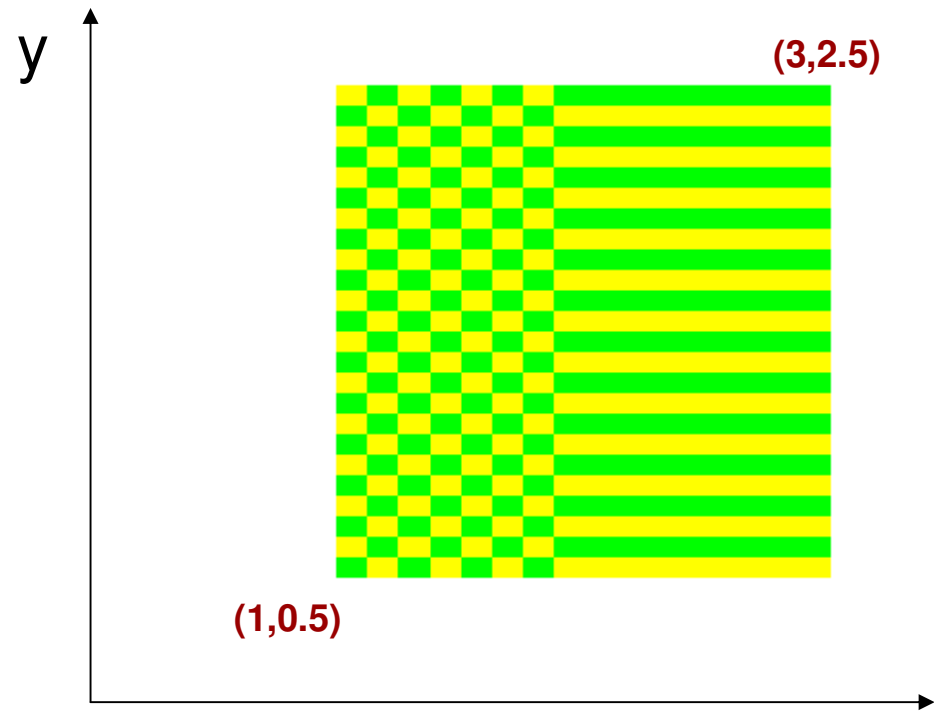
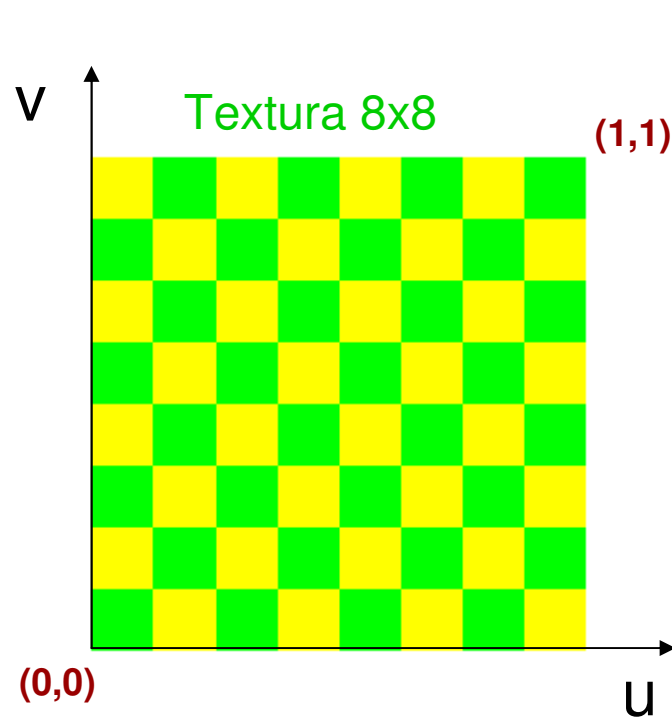


```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
```

```
glBegin(GL_QUADS);
    glTexCoord2d(0.0, 0.0); glVertex3d(1.0, 0.5, 0.0);
    glTexCoord2d(0.0, 3.0); glVertex3d(1.0, 2.5, 0.0);
    glTexCoord2d(2.0, 3.0); glVertex3d(3.0, 2.5, 0.0);
    glTexCoord2d(2.0, 0.0); glVertex3d(3.0, 0.5, 0.0);
glEnd();
```

Notação
das coordenadas
de textura em
OpenGL !

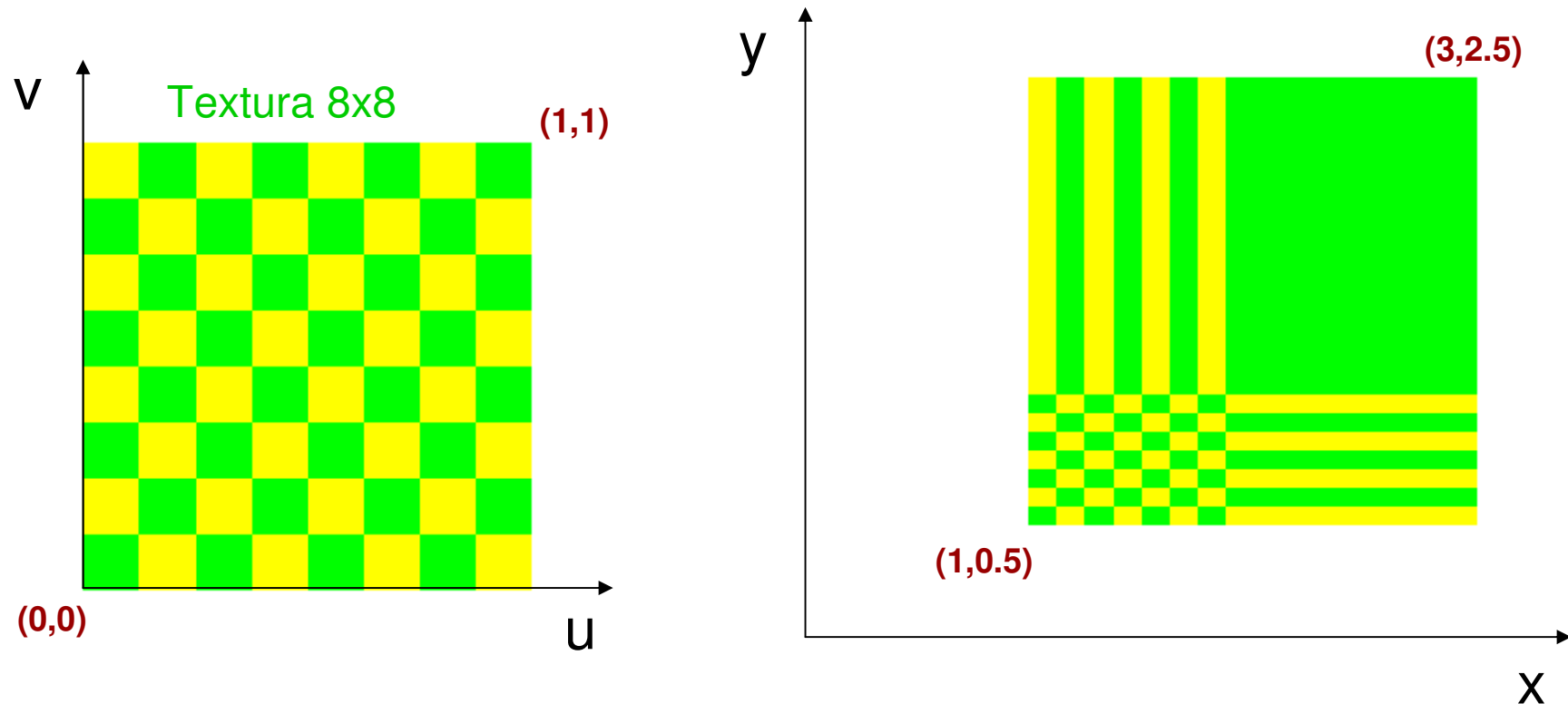
Repetição da textura se o parâmetro cair fora do intervalo



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

```
glBegin(GL_QUADS);
    glTexCoord2d(0.0, 0.0); glVertex3d(1.0, 0.5, 0.0);
    glTexCoord2d(0.0, 3.0); glVertex3d(1.0, 2.5, 0.0);
    glTexCoord2d(2.0, 3.0); glVertex3d(3.0, 2.5, 0.0);
    glTexCoord2d(2.0, 0.0); glVertex3d(3.0, 0.5, 0.0);
glEnd();
```

$u > 1 \rightarrow u = 1$
 $u < 0 \rightarrow u = 0$

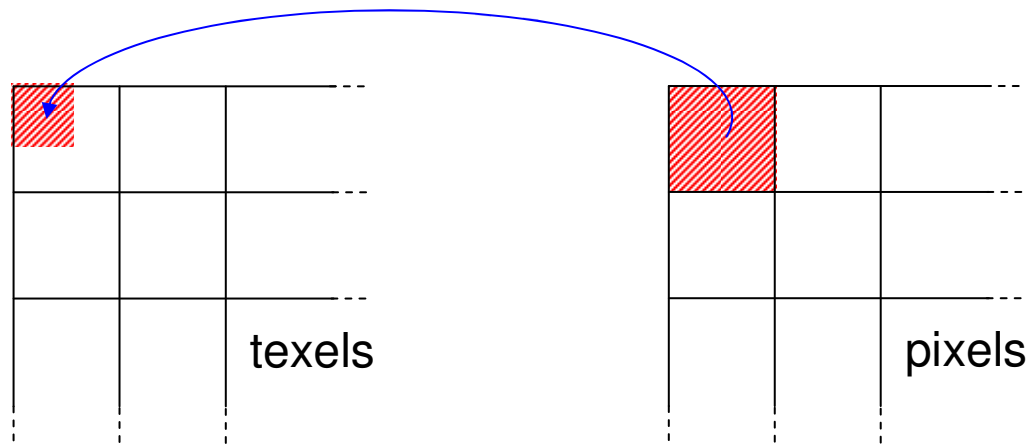


```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP);
```

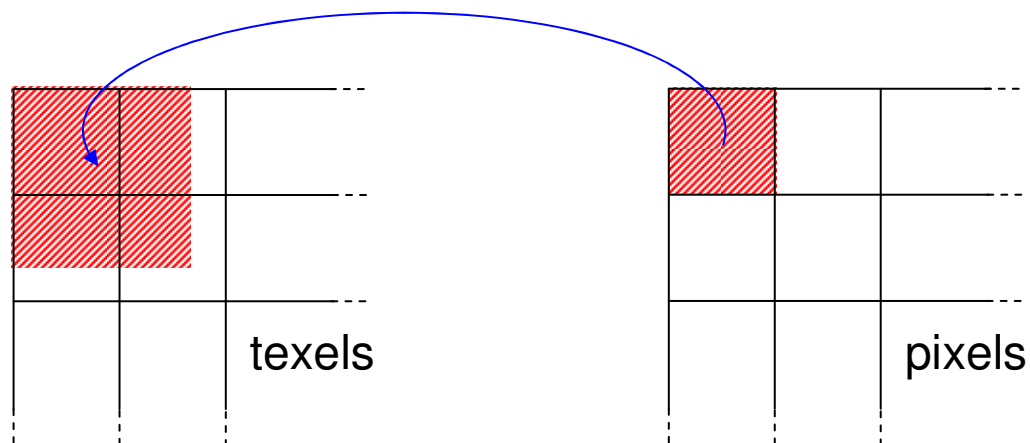
```
glBegin(GL_QUADS);
    glTexCoord2d(0.0, 0.0); glVertex3d(1.0, 0.5, 0.0);
    glTexCoord2d(0.0, 3.0); glVertex3d(1.0, 2.5, 0.0);
    glTexCoord2d(2.0, 3.0); glVertex3d(3.0, 2.5, 0.0);
    glTexCoord2d(2.0, 0.0); glVertex3d(3.0, 0.5, 0.0);
glEnd();
```


CORRESPONDÊNCIA ENTRE PIXELS E TEXELS (1)

Um pixel tem uma determinada área. Quando se faz a correspondência com a textura, a área aí obtida poderá ser diferente da área de um texel.



Pixel pintado com **ampliação**
(MAGNIFICATION) da textura



Pixel pintado com **redução**
(MINIFICATION) da textura

Nestes casos, qual o valor com que se deve pintar o pixel ?

Obs.: Havendo distorção, a área na textura não será um quadrado.

CORRESPONDÊNCIA ENTRE PIXELS E TEXELS (2)

Soluções em alternativa (que se podem misturar), tanto para **ampliação** como para **redução**:

Em OpenGL:

- Usar-se o texel mais próximo do ponto calculado pela correspondência com o (centro do) pixel.

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

- Fazer-se uma média ponderada com os 2x2 texels mais próximos do ponto calculado pela correspondência com o pixel (**filtragem**).

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

CORRESPONDÊNCIA ENTRE PIXELS E TEXELS (3)

EXEMPLOS DE RESULTADOS DE APLICAÇÃO

64x64 texels



360x360 pixels

GL_NEAREST

GL_LINEAR

mais lento,
melhor resultado

Texture Mapping in OpenGL

Example in C to be used together with a vertex evaluator

Brief summary:

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

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, filter_value);
```

```
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, filter_value);
```

```
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGB, image_w, image_h,  
             0, GL_RGB, GL_UNSIGNED_BYTE, image);
```

A byte buffer,
ByteBuffer in Java

```
glMap2d(GL_MAP2_TEXTURE_COORD_2, 0.0, 1.0, 2, 2,  
        0.0, 1.0, 4, 2, texture_points);
```

```
glEnable(GL_MAP2_TEXTURE_COORD_2);
```

```
glEnable(GL_TEXTURE_2D);
```

```
/* evaluators here */
```

u stride

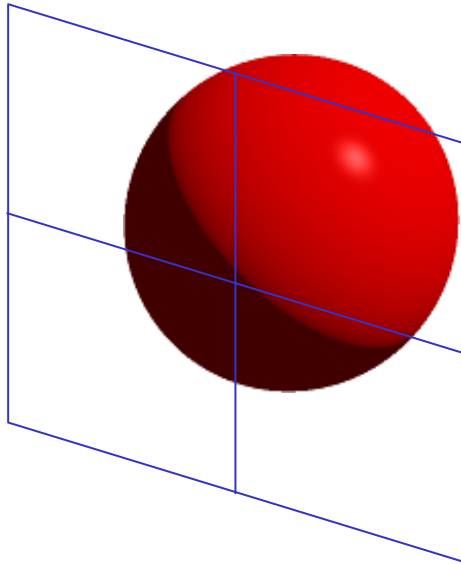
v stride

4 corners
(texture coordinates)

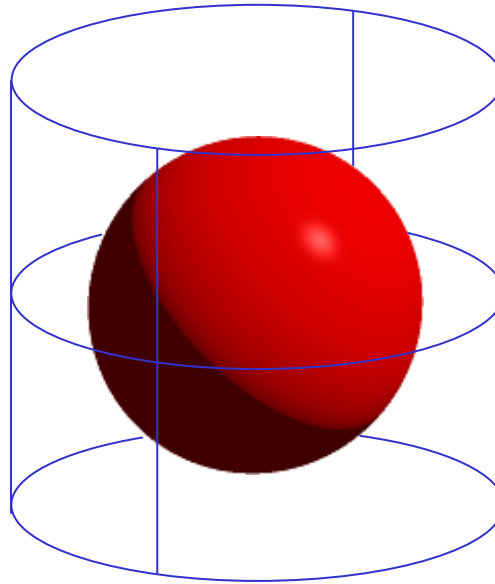
Part of the texture to
be visualized, e.g.:
{ 0,1, 0.8,1,
 0,0, 0.8,0 }

TEXTURAS MAPEADAS

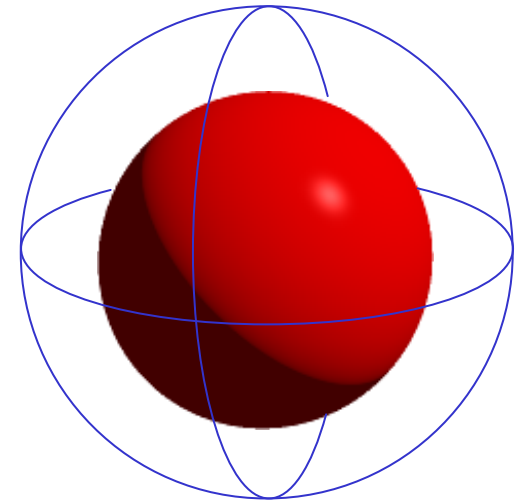
TIPOS CLÁSSICOS DE MAPEAMENTO:



Ortogonal



Cilíndrico



Esférico

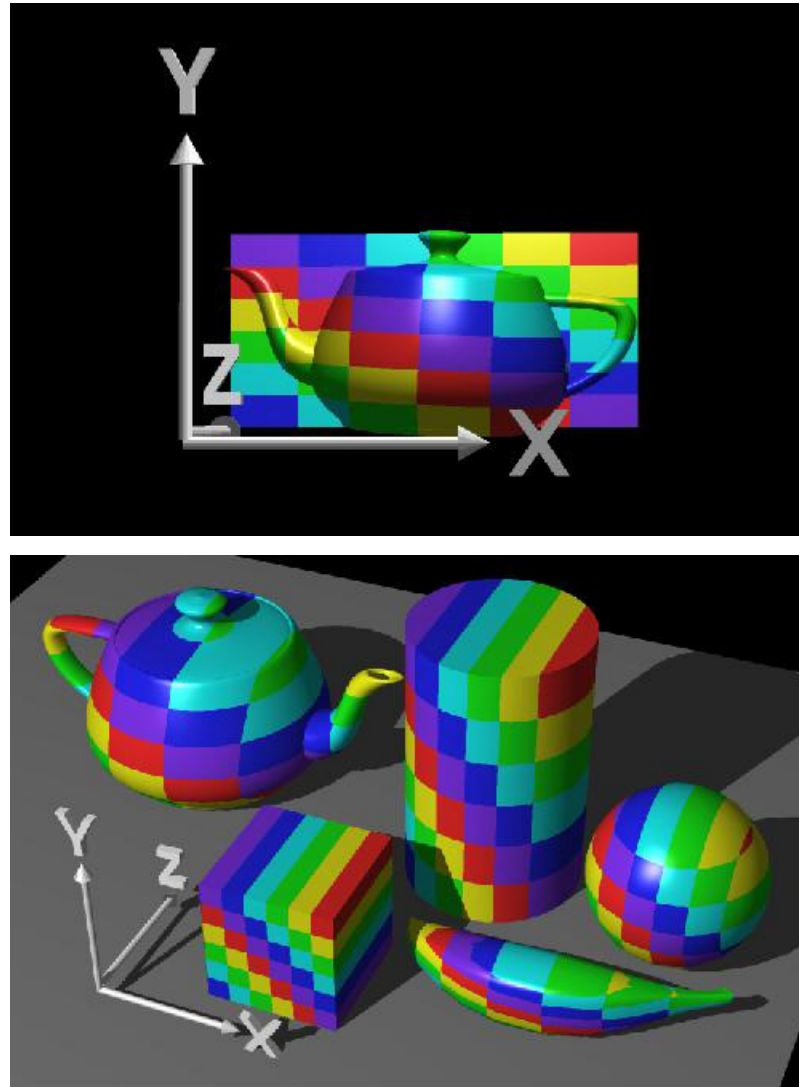
Os slides seguintes, num referencial segundo a regra da mão esquerda, são da autoria de



Rosalee Wolfe
SIGGRAPH 97 Education Slide set

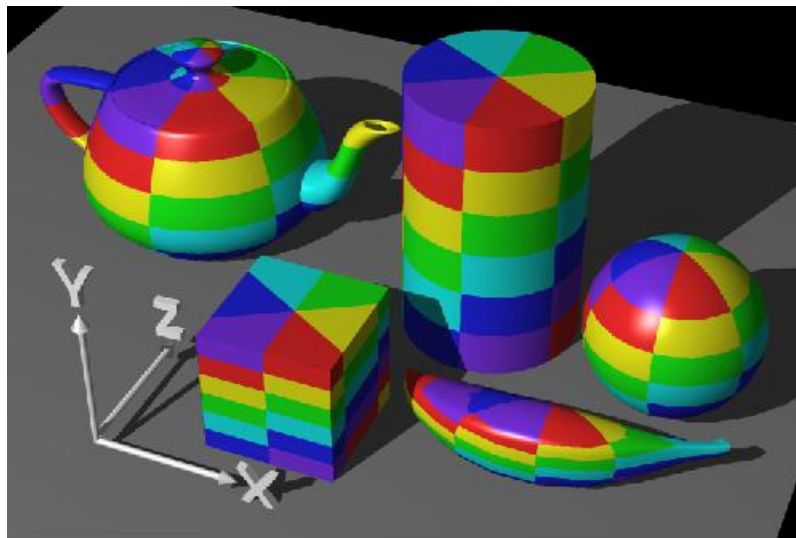
M. Próspero

Mapeamento Ortogonal



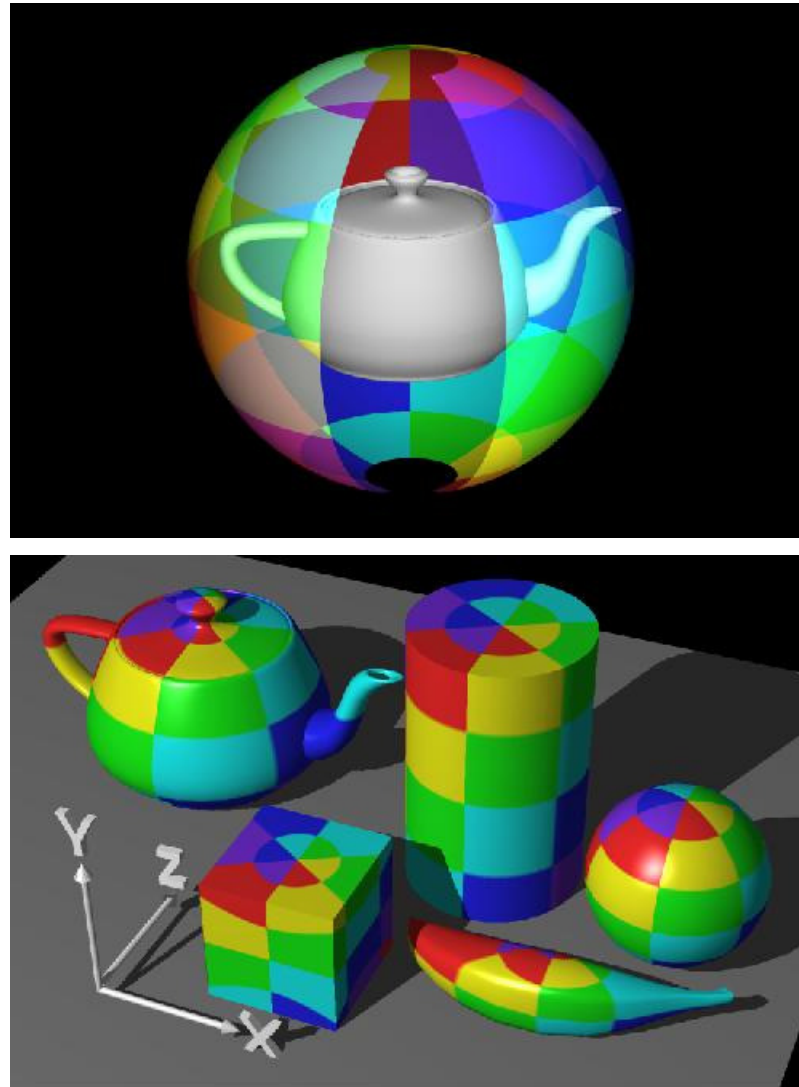
Qual a correspondência entre o espaço da textura e o espaço do objeto ?

Mapeamento Cilíndrico



Qual a correspondência entre o espaço da textura e o espaço do objeto ?

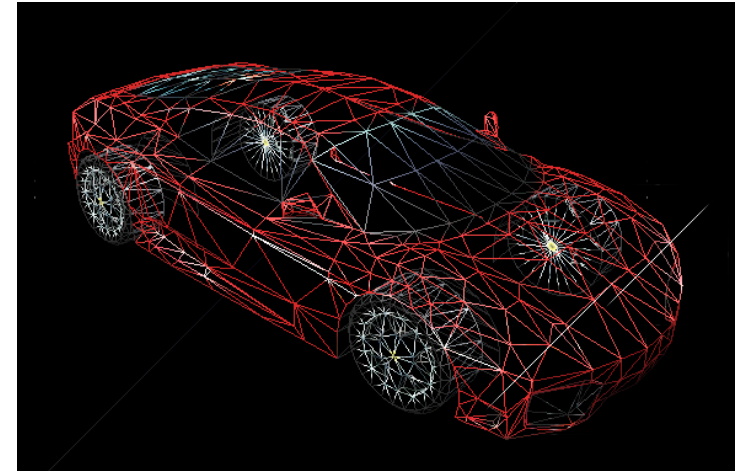
Mapeamento Esférico



Qual a correspondência entre o espaço da textura e o espaço do objeto ?



+



512x512

From Roland's VRML97 Site

M. Próspero