1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

dei25 departamento de engenharia informática 1995 – 2020

Computação Gráfica

André Perrotta (avperrotta@dei.uc.pt)

Hugo Amaro (hamaro@dei.uc.pt)

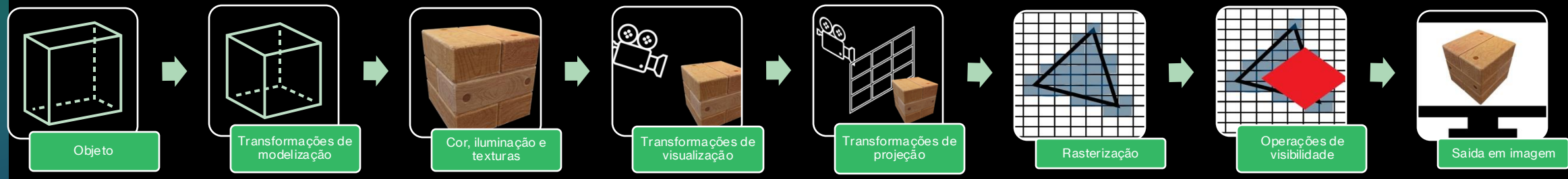
T_09:

Texturas - intro

Objetivos da aula

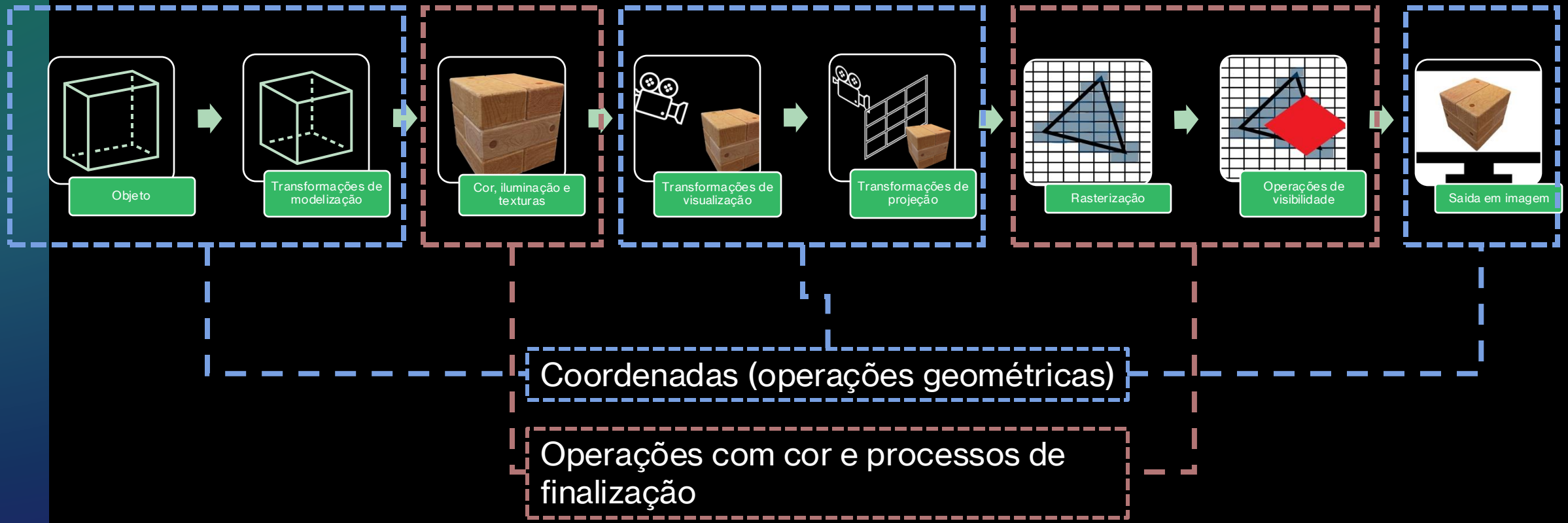
- Introduzir o conceito de Texturas em OpenGl
- Entender como o processo é realizado
- Entender configurações básicas
- Implementar texturas em OpenGl/OF

Render pipeline



Pipeline de renderização POLIGONAL

Render pipeline



Cor e Iluminação

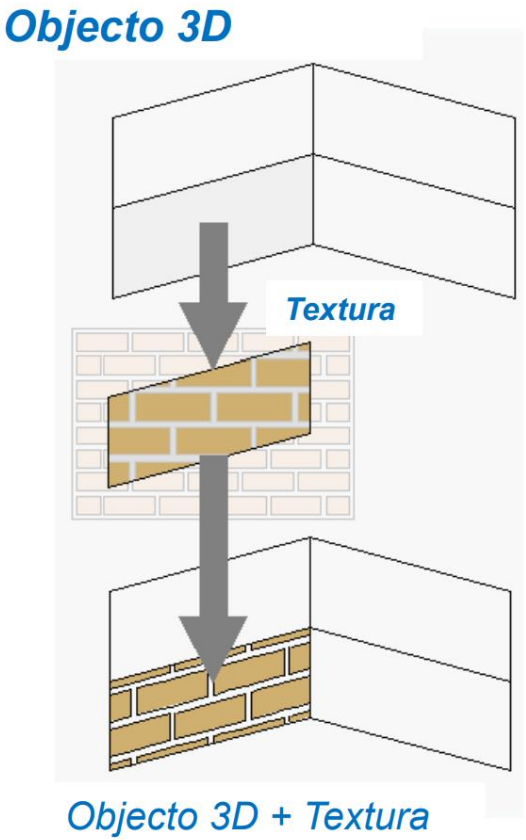
Cor, iluminação e texturas

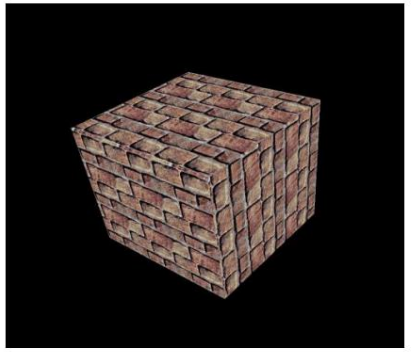
Ponto de partida:

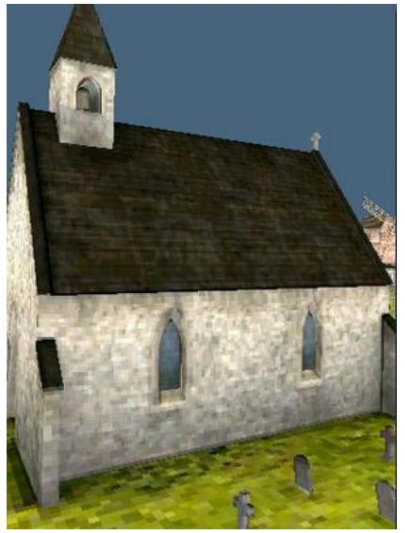
- Obter uma estética realista, com detalhes e efeitos de luz complexos, através de modelagem 3D e cálculo de iluminação com o modelo de Phong, nem sempre é possível
  - Uma estratégia para contornar este problema é a utilização de imagens 2D mapeadas aos objetos 3D.

Textura!

- Forma mais simples de conseguir images com aspecto real !

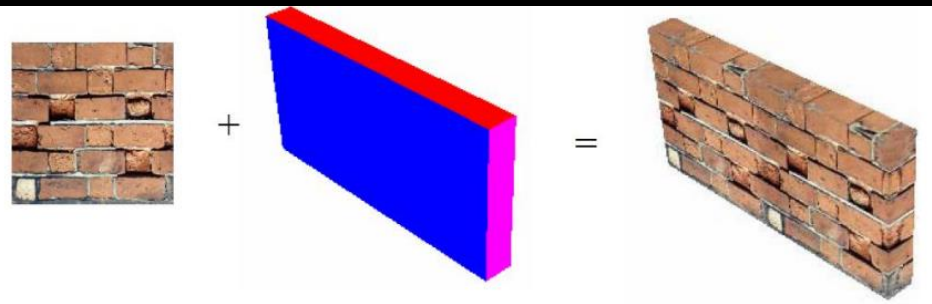




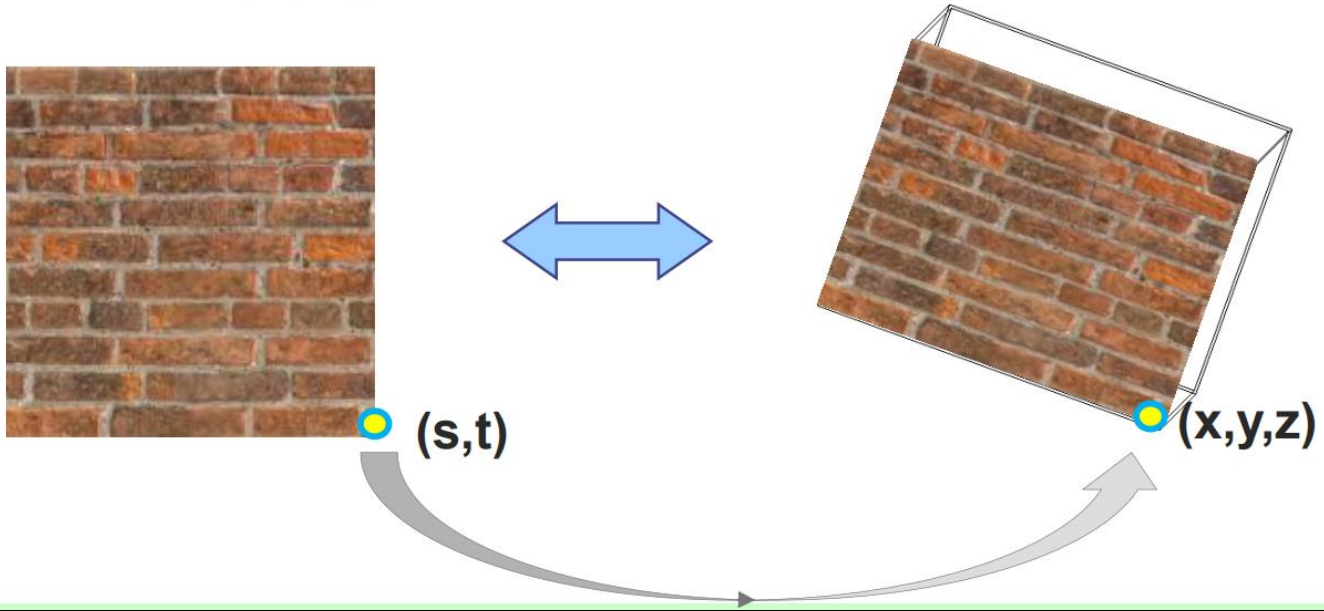


Problema: Mapeamento de textura

- Mapeamento

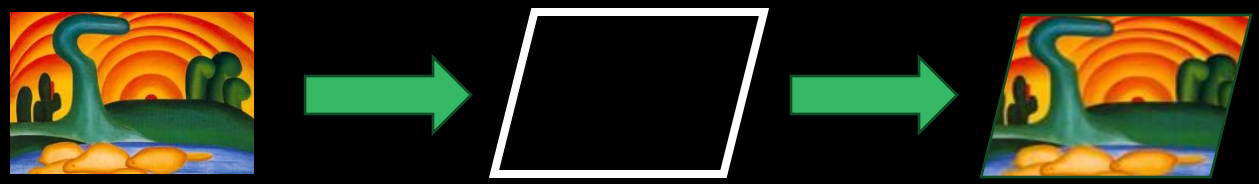


- Admitindo uma textura uma **imagem bidimensional**, como associar um ponto de um mapa bidimensional **(s,t)** com um vértice de um objecto tridimensional **(x,y,z)** de modo unívoco ?

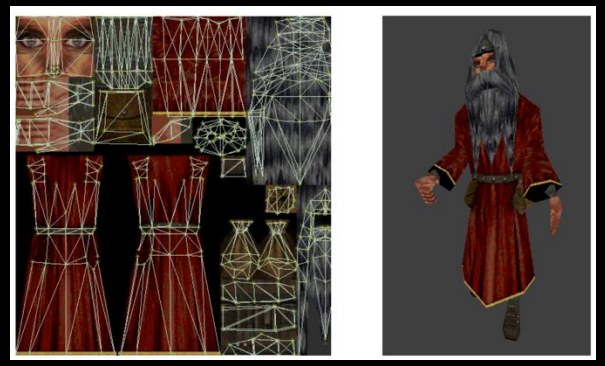


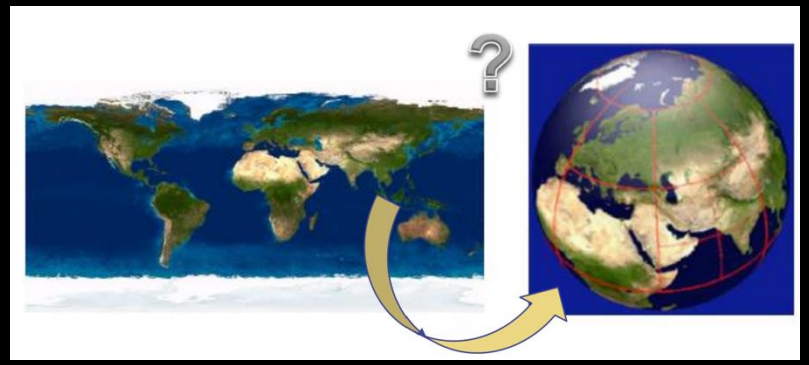
Mapeamento de textura

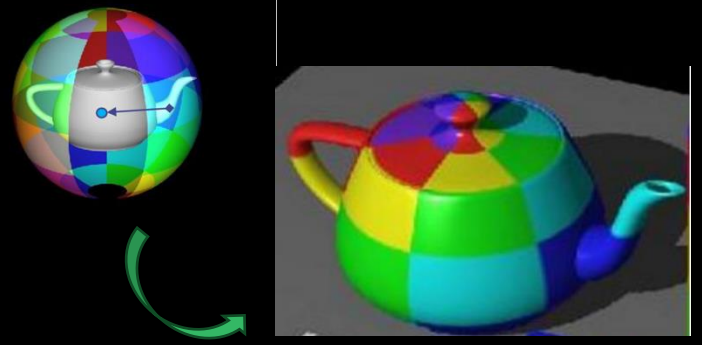
- Pode ser simples e direto



- Pode ser complexo

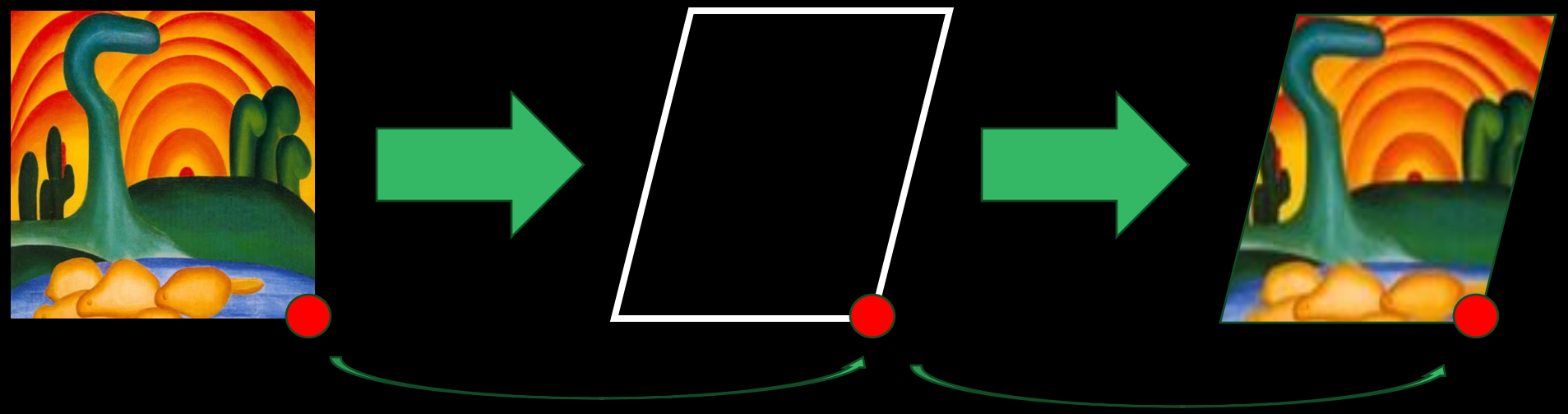






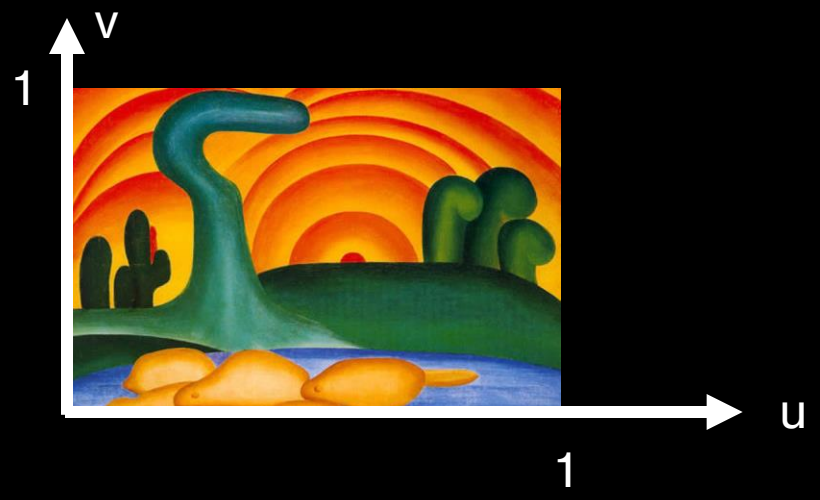
Mapeamento simples: poligonal

- O mapeamento poligonal ou direto vai atribuir uma coordenada da imagem para cada vértice de um polígono.
- O caso mais simples é o de uma imagem à um retângulo (ou paralelogramo)



Espaço de coordenadas: textura

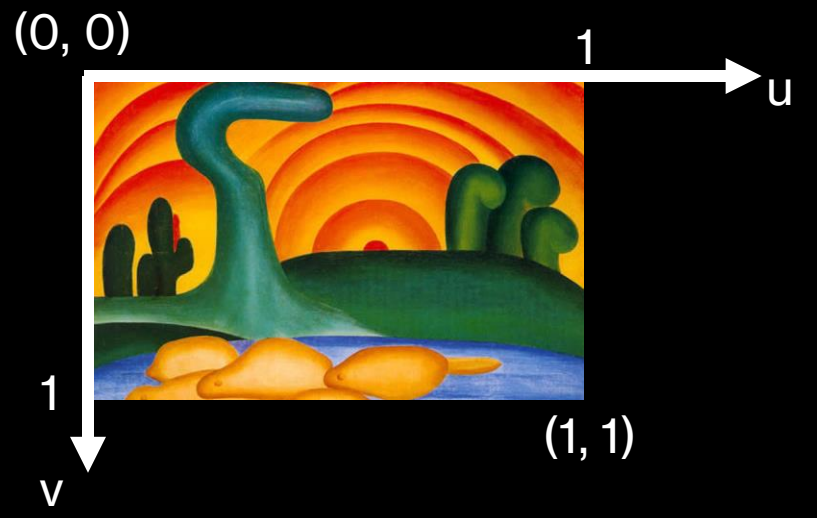
- A textura (imagem) é uma função: T(u, v)
  - Domínio: espaço de coordenadas cartesiano, normalizado
    - Horizontal: u [0, 1] -> corresponde [0, largura em pixels]
    - Vertical: v [0, 1] -> corresponde [0, altura em pixels]
  - Contradomínio: cor(u, v) (pixel RGB*)



- *ou outro formato de cor, depende do tipo de imagem

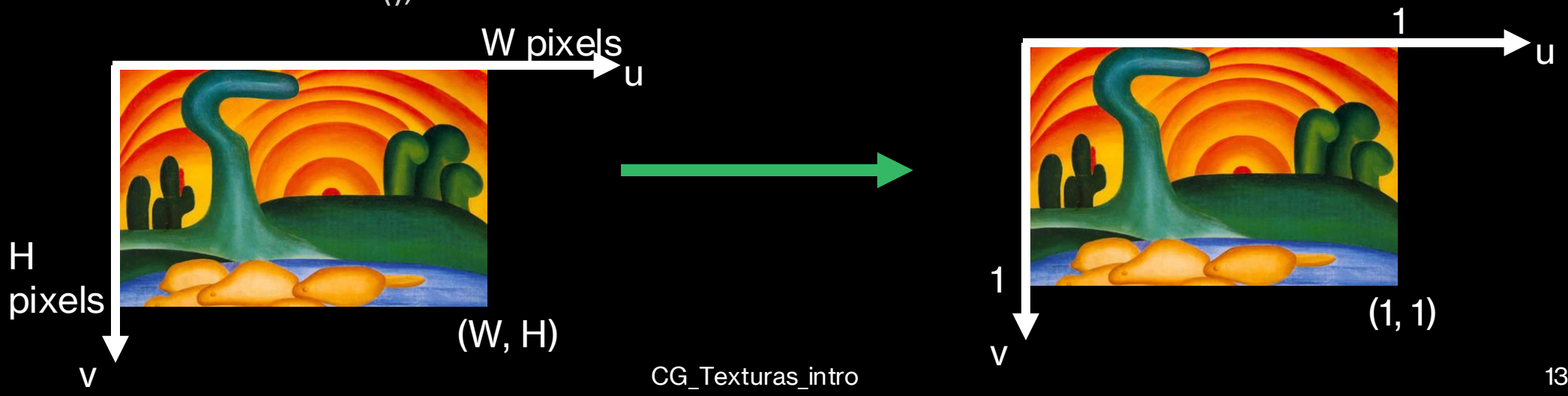
Espaço de coordenadas: textura

- A origem do sistema de coordenadas da textura pode variar.
- É dependente da API/framework utilizada para carregar as imagens.
- Em OpenFrameworks a origem é o top-left da imagem



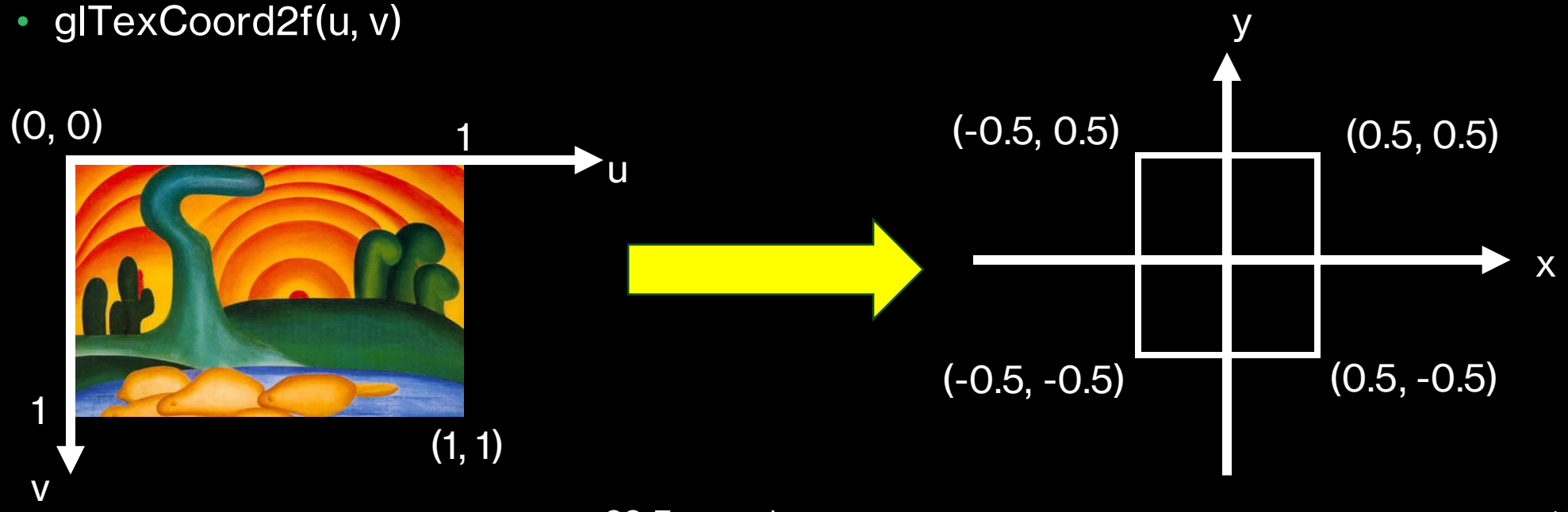
Definição importante em OF

- No OpenFrameworks, por defeito, as coordenadas de texturas são absolutas (usam o tamanho em pixels da imagem).
- Para utilizar coordenadas normalizadas [0, 1] devemos utilizar a chamada:
  - ofDisableArbTex();

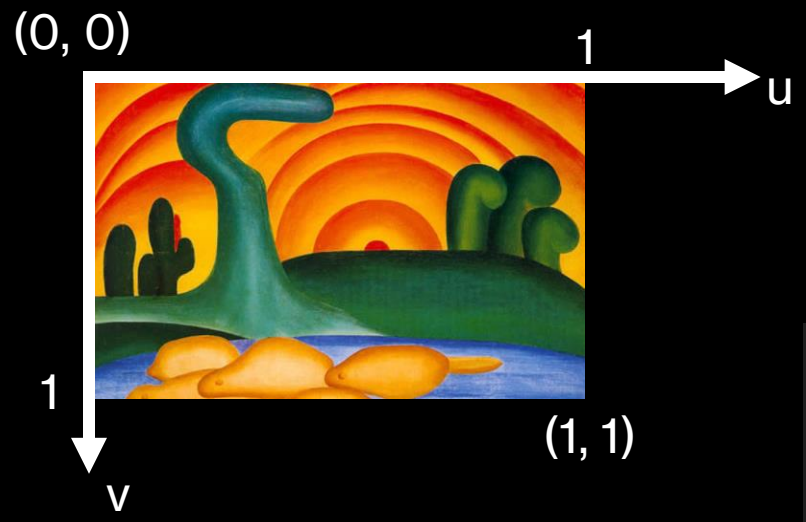


Mapeamento: (u, v) -> (x, y, z)

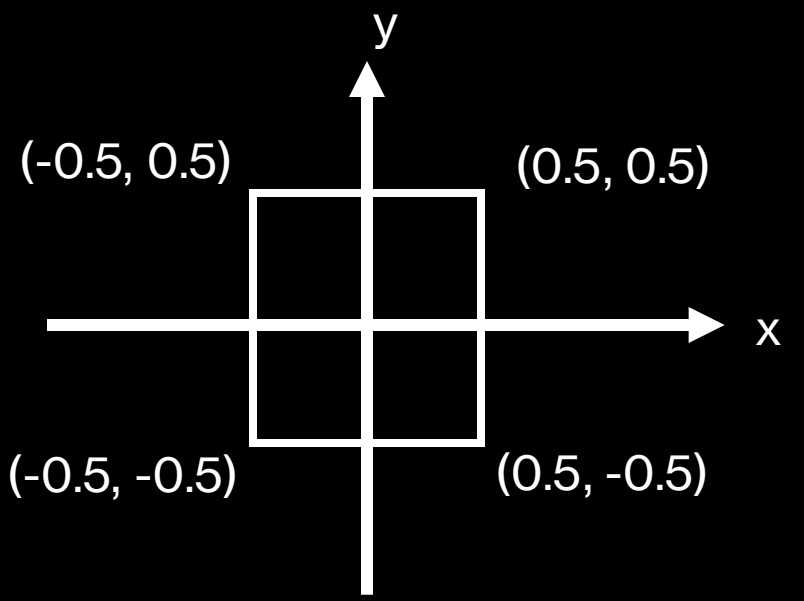
- O mapeamento direto em openGl é bastante simples. Basta assignar uma coordenada de textura (u, v) a cada vértice (x, y, z) da geometria onde quero “pintar” a textura.
  - glTexCoord2f(u, v)



Mapeamento: (u, v) -> (x, y, z)



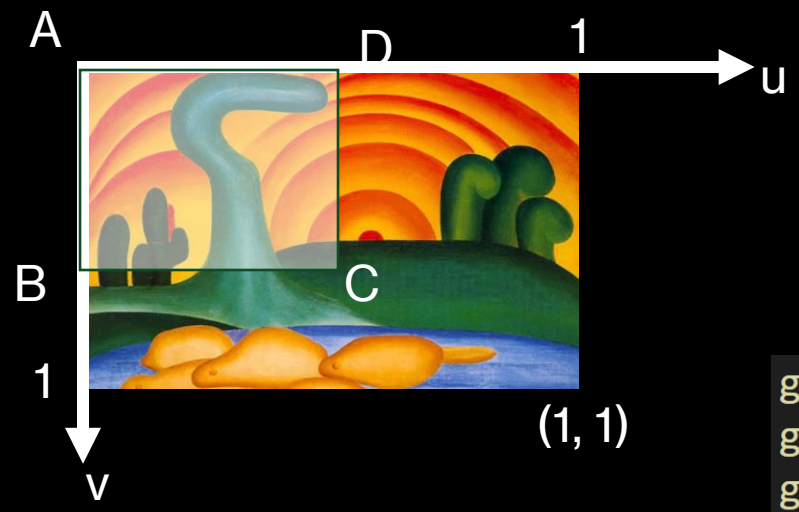

```
glBegin(GL_QUADS);
glTexCoord2f(0, 0);
glVertex2f(-0.5, 0.5);
glTexCoord2f(0, 1);
glVertex2f(-0.5, -0.5);
glTexCoord2f(1, 1);
glVertex2f(0.5, -0.5);
glTexCoord2f(1, 0);
glVertex2f(0.5, 0.5);
glEnd();
```



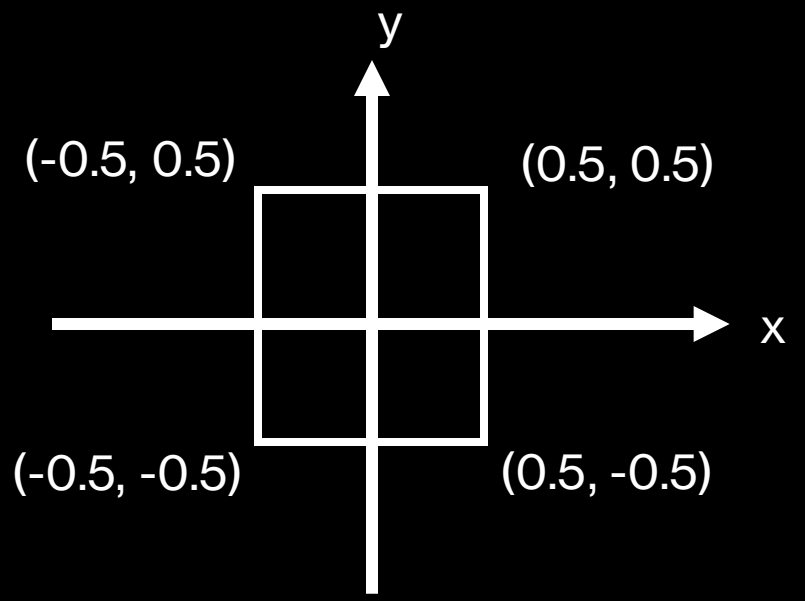
Mapeamento: (u, v) -> (x, y, z)

- Apesar de as coordenadas (u, v) da textura serem normalizadas entre [0, 1], não há proibição de utilizarmos quaisquer valores. Por exemplo: (2, 2) ou (0.5, 0.5) para mapear à um vértice (x, y, z).
- Isso gera possibilidades de configuração da textura.
  - Recorte
  - tiling (repetição da textura dentro do polígono)

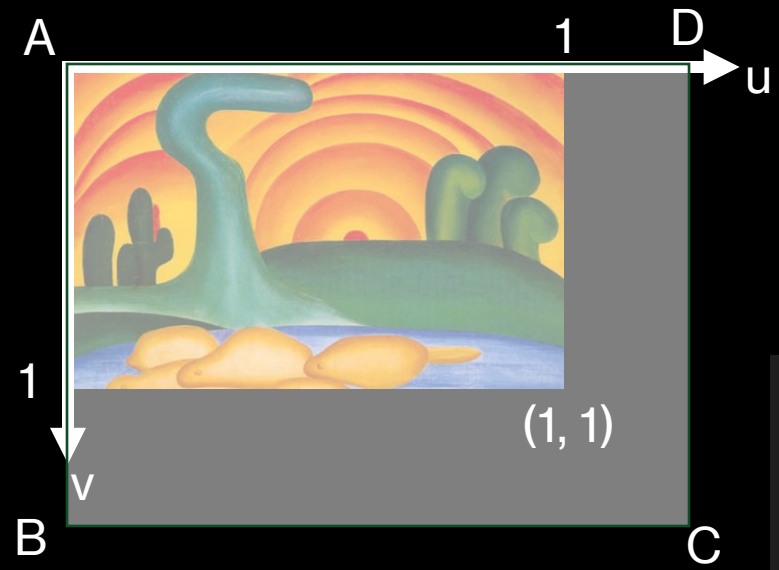
Recorte (zoom in)



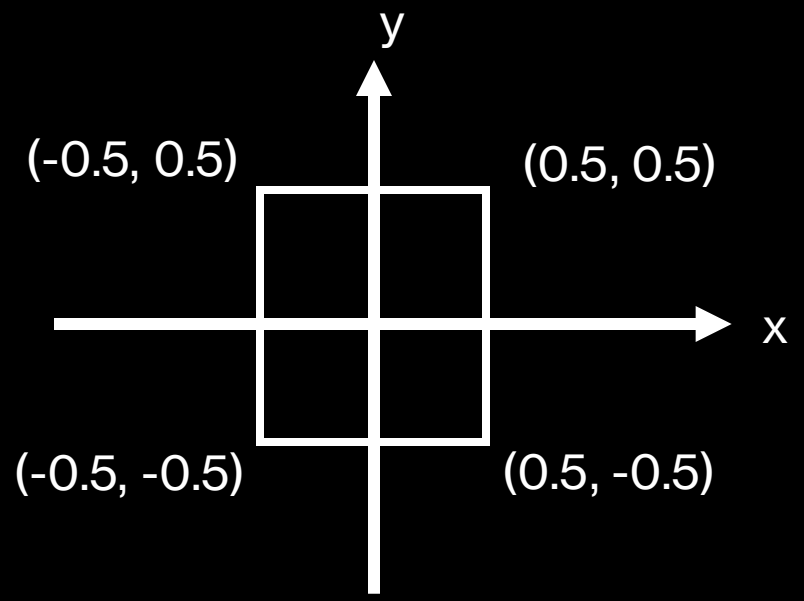

```
glBegin(GL_QUADS);
glTexCoord2f(0, 0);
glVertex2f(-0.5, 0.5);
glTexCoord2f(0, 0.5);
glVertex2f(-0.5, -0.5);
glTexCoord2f(0.5, 0.5);
glVertex2f(0.5, -0.5);
glTexCoord2f(0.5, 0);
glVertex2f(0.5, 0.5);
glEnd();
```

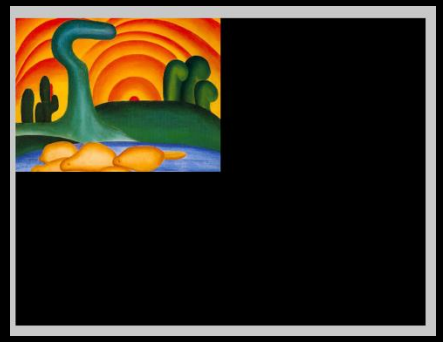


Recorte (zoom out)

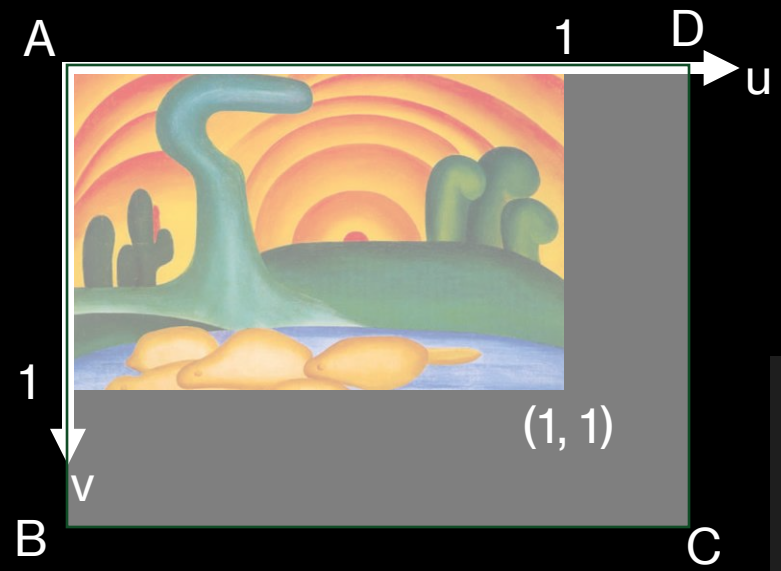



```
glBegin(GL_QUADS);
glTexCoord2f(0, 0);
glVertex2f(-0.5, 0.5);
glTexCoord2f(0, 2);
glVertex2f(-0.5, -0.5);
glTexCoord2f(2, 2);
glVertex2f(0.5, -0.5);
glTexCoord2f(2, 0);
glVertex2f(0.5, 0.5);
glEnd();
```

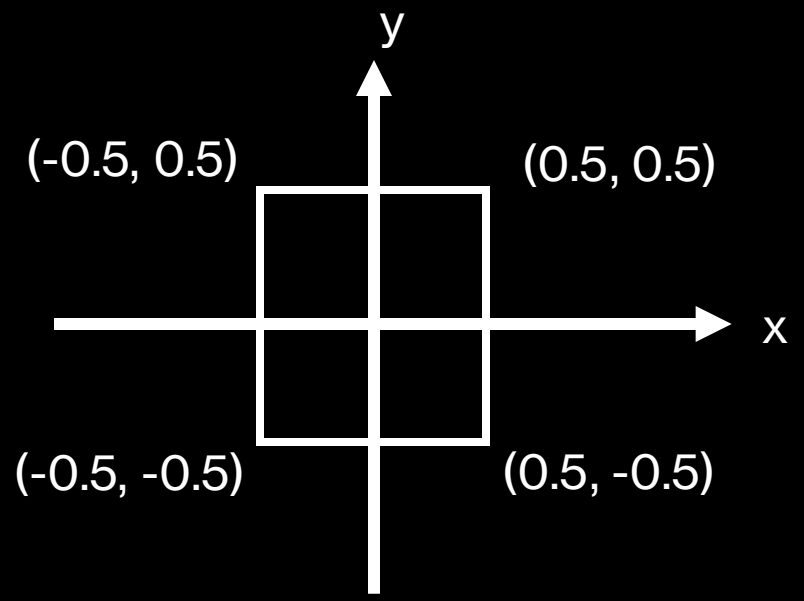




Tiling




```
glBegin(GL_QUADS);
glTexCoord2f(0, 0);
glVertex2f(-0.5, 0.5);
glTexCoord2f(0, 2);
glVertex2f(-0.5, -0.5);
glTexCoord2f(2, 2);
glVertex2f(0.5, -0.5);
glTexCoord2f(2, 0);
glVertex2f(0.5, 0.5);
glEnd();
```



Implementação

- Em OpenGL/OpenFrameworks é (razoavelmente) simples operacionalizar o uso de texturas:

1. Carregar imagem
   - ofImage img; img.load(“imagem.formato”)
     - formato pode ser bmp, jpg, png, etc.
     - Imagens devem estar na pasta bin\data do projeto
2. Ativar o modo de textura do OpenGl
   - glEnable(GL_TEXTURE);
3. Indicar ao OpenGl que queremos usar a imagem como textura
   - img.bind();
4. Escolher as configurações da textura
   - são muitas possibilidades
   - (https://registry.khronos.org/OpenGL-Refpages/gl4/html/glTexParameter.xhtml)

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

5. Desenhar a geometria indicando as coordenadas de glTexCoord2f(u, v) para cada vértice.
   - glBegin() .... glEnd()
6. Indicar que queremos “desconectar” da textura
   - img.unbind();
7. Desabilitar o modo textura para não influenciar os próximos desenhos
   - glDisable(GL_TEXTURE);

Implementação: configurações

- Modo e operação:
  - glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, mode);
    - Mode = GL_DECAL -> não é influenciada pela iluminação/glColor
    - Mode = GL_MODULATE -> cor final é corTextura*cor_Iluminação/glColor
  - Método de interpolação:
    - Para maior:
      - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, tipo);
    - Para menor:
      - glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, tipo);
    - tipo = muitas possibilidades! Ver na referência online (link no slide anterior)
  - Método de preenchimento do espaçõ vazio
    - Horizontal: glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, mode);
    - Vertical: glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, mode);
      - Mode = muitas possibilidades!
      - Vai definer se fazemos tiling ou não, se complete com a cor do último pixel ou da borda, etc.

Créditos

- Muitos slides e imagens foram “emprestados” do ótimo material do prof. Jorge Henriques ☺