



COMPUTAÇÃO GRÁFICA



LEI / LCC
DEPARTAMENTO DE INFORMÁTICA
UNIVERSIDADE DO MINHO

Texturing

Cylinder



Textured Cylinder



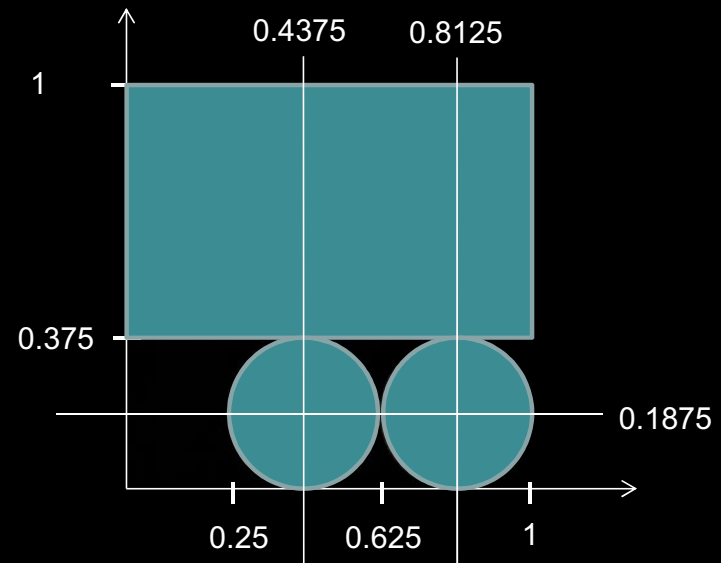


Texture Atlas

Image space

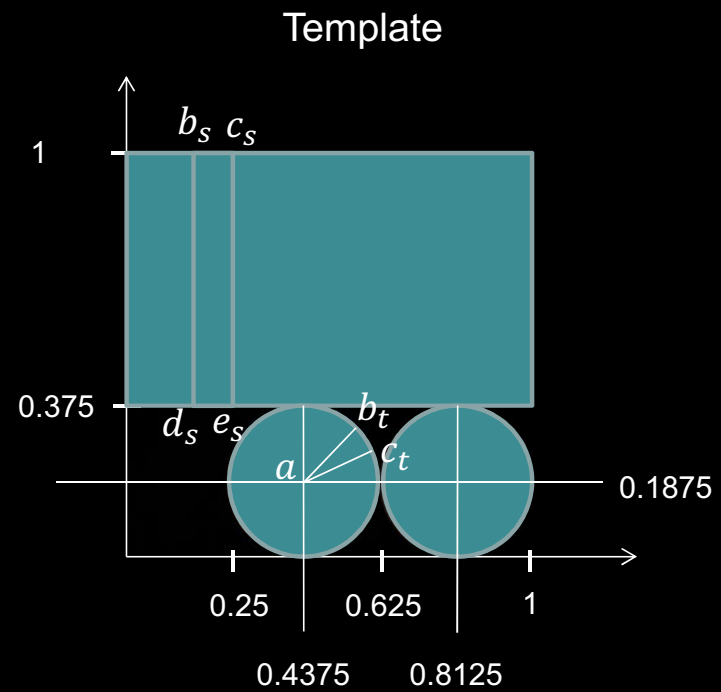
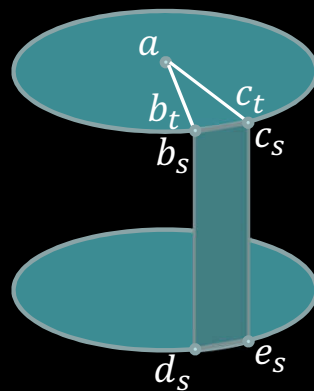


Texture space





Texture Coordinates





DevIL

- Open an image file and define origin of image space in DevIL

```
// setup - done once
ilInit();
ilEnable(IL_ORIGIN_SET);
ilOriginFunc(IL_ORIGIN_LOWER_LEFT);

// for each image ...
ilGenImages(1,ima); // unsigned int ima[...]
ilBindImage(ima[0]);
ilLoadImage((ILstring)filename); // char *filename
```



DevIL

- Convert to RGBA

```
ilConvertImage(IL_RGBA, IL_UNSIGNED_BYTE);
```



DevIL

- Get the required info

```
int width = ilGetInteger(IL_IMAGE_WIDTH);  
int height = ilGetInteger(IL_IMAGE_HEIGHT);  
  
unsigned char *imageData = ilGetData();
```



Texture Creation in OpenGL

```
// create a texture slot
glGenTextures(1,&texID);
// bind the slot
glBindTexture(GL_TEXTURE_2D,texID);

// define texture parameters
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);

// send texture data to OpenGL
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, tw, th, 0,
             GL_RGBA, GL_UNSIGNED_BYTE, texData);
```




Texture Setup

- Code to create a texture from a file (DevIL + OpenGL):

```
unsigned int t, tw, th;
unsigned char *texData;
ilGenImages(1,&t);
ilBindImage(t);
ilLoadImage((ILstring)"Oil_Drum001h.jpg");
tw = ilGetInteger(IL_IMAGE_WIDTH);
th = ilGetInteger(IL_IMAGE_HEIGHT);
ilConvertImage(IL_RGBA, IL_UNSIGNED_BYTE);
texData = ilGetData();

glGenTextures(1,&texID); // unsigned int texID - variavel global;

glBindTexture(GL_TEXTURE_2D,texID);
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_RGBA, tw, th, 0, GL_RGBA, GL_UNSIGNED_BYTE, texData);
```



Textures

- Activate 2D texturing (init):

```
glEnable(GL_TEXTURE_2D);
```

} Setup

- Before drawing:

```
glBindTexture(GL_TEXTURE_2D, texID);
```

- Drawing (VBOs or immediate mode)

- After drawing:

```
glBindTexture(GL_TEXTURE_2D, 0);
```

} Render



Tex Coords in immediate mode

- For each vertex define its texture coordinate:

```
glBegin(GL_TRIANGLES);  
    glTexCoord2f(s1, t1);  
    glVertex3f(x1,y1,z1);  
    glTexCoord2f(s2, t2);  
    glVertex3f( x2,y2,z2);  
    ...  
glEnd();
```



Textures w/ VBOs - Setup

- Activate arrays:

```
glEnableClientState(GL_VERTEX_ARRAY);  
glEnableClientState(GL_NORMAL_ARRAY);  
glEnableClientState(GL_TEXTURE_COORD_ARRAY);
```



Textures w/ VBOs - Setup

- Add an array with texture coordinates:
 - Two components per vertex
 - Create an extra VBO;

```
unsigned int vertices, normals, texCoords;
float *v,*n,*t;
int count; // store the number of vertices
...
glGenBuffers(1, &vertices);
glBindBuffer(GL_ARRAY_BUFFER, vertices);
glBufferData(GL_ARRAY_BUFFER, sizeof(float) * 3 * count, v, GL_STATIC_DRAW);

glGenBuffers(1, &normals);
glBindBuffer(GL_ARRAY_BUFFER, normals);
glBufferData(GL_ARRAY_BUFFER, sizeof(float) * 3 * count, n, GL_STATIC_DRAW);

glGenBuffers(1, &texCoords);
glBindBuffer(GL_ARRAY_BUFFER, texCoords);
glBufferData(GL_ARRAY_BUFFER, sizeof(float) * 2 * count, t, GL_STATIC_DRAW);
```



Textures w/ VBOs - Render

- Semantics:

```
glBindBuffer(GL_ARRAY_BUFFER, vertices);  
glVertexPointer(3, GL_FLOAT, 0, 0);
```

```
glBindBuffer(GL_ARRAY_BUFFER, normals);  
glNormalPointer(GL_FLOAT, 0, 0);
```

```
glBindBuffer(GL_ARRAY_BUFFER, texCoords);  
glTexCoordPointer(2, GL_FLOAT, 0, 0);
```

- Texture bind and call glDraw*



Assignment

- Compute the texture coordinates for the cylinder and floor and apply the given textures:
 - `drawFloor`: add texture coordinates in immediate mode
 - `prepareCylinder`: add texture coordinates using VBOs
 - `drawCylinder`: add texture coordinates buffer and define its semantic
 - `renderScene`: bind textures



Going Further

- Create another texture compatible with the provided template
- Create a template and an example texture for a box
- Are there any issues when applying automatic mipmapping to atlas textures?