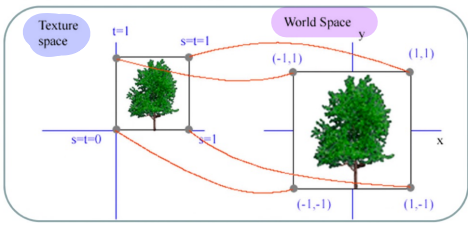
Texturing

```
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();
```

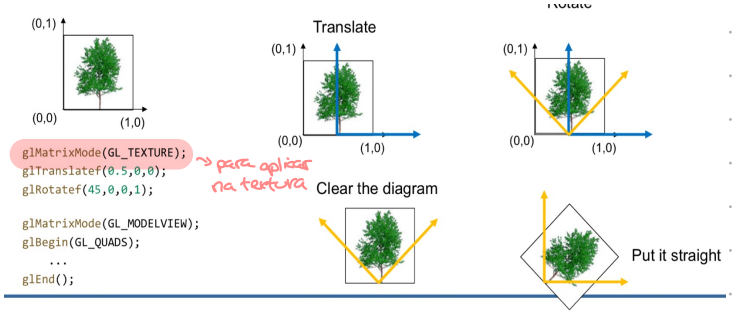


Textures:

- Usage:
  - → load image
  - → create texture in openGL
  - → define texture parameters
- Applications: they have their own coordinate systems (s,t and r axis) that define a mapping between the vertices and the coordinates in texture

Coordinates:

- With VBOs:
  - → create an array with texture coordinates
  - → create a buffer and copy the array data to the buffer



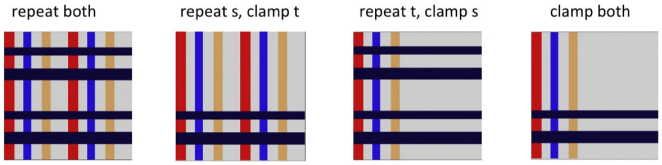
Geometric Operations:

```
// Assume an image has been loaded and that w and h contain the width
// and height of the image respectively.
// Furthermore, assume that each pixel contains 4 unsigned bytes (RGBA)

int texName[1];

glGenTextures(1, texName);
glBindTexture(GL_TEXTURE_2D, texName[0]);
// wrapping parameters
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT);
// filtering
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, w, h,
             0, GL_RGBA, GL_UNSIGNED_BYTE, imageData);
```

Clamp & Repeat

Original image

GL_CLAMP
GL_REPEAT



Parameters:

- Mag: when the texture needs to be expanded to fit the triangles on screen
- Min: when the texture is shrunk

Flickering and Aliasing:

- vertices have texture coordinates specified by the application
- pixels have texture coordinates interpolated based on distance to vertices

when the camera moves, triangle shifts in screen and pixel coordinates are updated.
when a large image is used to cover a small portion of the screen, pixels may get totally different colors causing flickering.

MIPmapping: when the texture is severely shrunk it glitters when the camera or objects move - aliasing.
MIPmapping creates multiple textures at different scales, as in a pyramid.

- filtering: choose more suitable level or a linear combination between 2 or more suitable levels (LINEAR or NEAREST)

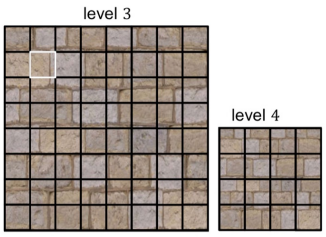
Advantages: better quality
potentially faster

Disadvantages: memory required (+- 33%.)
initial setup

Assume that:
- mipmap level is 3,25
- Texture coordinate = (0.2,0.8)

GL_NEAREST_MIPMAP_NEAREST

Pixel color =



Assume that:
- mipmap level is 3,25
- Texture coordinate = (0.2,0.8)

GL_NEAREST_MIPMAP_LINEAR

Pixel color = 0.75 * + 0.25 *

Assume that:
- mipmap level is 3,25
- Texture coordinate = (0.2,0.8)

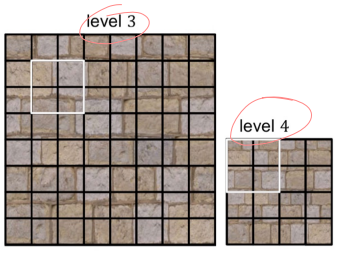
GL_LINEAR_MIPMAP_NEAREST

Pixel color = weighted average

Assume that:
- mipmap level is 3,25
- Texture coordinate = (0.2,0.8)

GL_LINEAR_MIPMAP_LINEAR

Pixel color = 0.75 * weighted average + 0.25 * weighted average



Final color:

- GL_REPLACE	C = Ct	A = At	C = RGBA
- GL_MODULATE	C = Ct * Cg	A = At * Ag	
- GL_BLEND	C = Cg*(1-Ct) + Ce*Ct	A = Ag * At	
- GL_DECAL	C = Cg*(1-At) + Ct*At	A = Af	

g = geometry, t = texture, e = GL_TEXTURE_ENV_COLOR

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

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

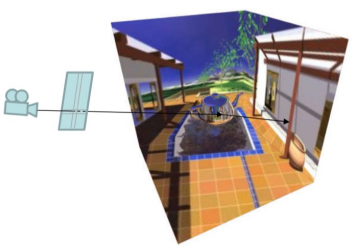
Transparency: drawing order is relevant for partial transparence. For total transparency, the alpha channel test is an appropriate solution. The test is performed before the z-buffer is written and eliminates every pixel which fails the test. Hence, the pixels do not affect the z buffer.

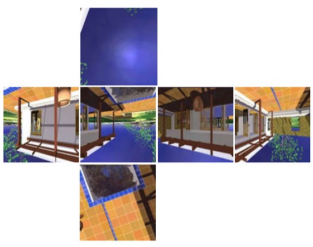
- Partial: allows to combine a color with what was previously written in the frame buffer. Ordering is crucial: opaque elements must be drawn first and transparent elements must be ordered based on distance do camera or using BSP. Furthest elements are drawn first.
  To compute the final color mix the two using weights for the fragment and new colors.

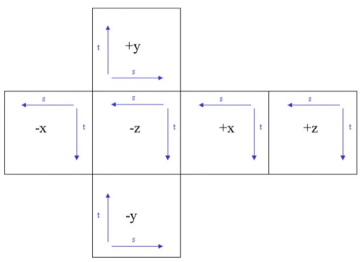
Final color = Cn * S + C * D

S = Alpha n
D = 1 - Alpha n

Environment Map: each texel represents a direction and its color is the color we would see if we were at the center of the cube looking in that direction.







- based on the normal at the pixel of the object a reflection vector and its intersection with the box are computed. The texel at the point of intersection is used to shade the object.

Cube Mapping:

- define a camera with a field view of 90 degrees
- aim the camera along the positive x axis and capture the frame for the respective cube side
- repeate for the remaining 5 directions.

```
glTexCoord2f( 0  , 1,5 );
glTexCoord2f( 1  , 1,5 );
glTexCoord2f( 1  , 0,5 );
glTexCoord2f( 0  , 0,5 );
```