

Textures

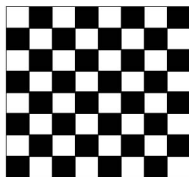
Textures

Texture = 1/2/3D data table.

Textures

Texture = 1/2/3D data table.

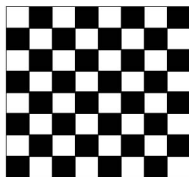
Often: (color values of) 2D picture.



Textures

Texture = 1/2/3D data table.

Often: (color values of) 2D picture.

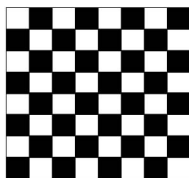


External file, generated online inside program (animated textures), or generated (offline or online) via rendering in OpenGL.

Textures

Texture = 1/2/3D data table.

Often: (color values of) 2D picture.



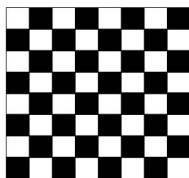
External file, generated online inside program (animated textures), or generated (offline or online) via rendering in OpenGL.

But texture data can be interpreted as anything, e.g. normal vectors, light maps/shadow maps, heightfields, ... (when using programmed shaders).

Textures

Texture = 1/2/3D data table.

Often: (color values of) 2D picture.



External file, generated online inside program (animated textures), or generated (offline or online) via rendering in OpenGL.

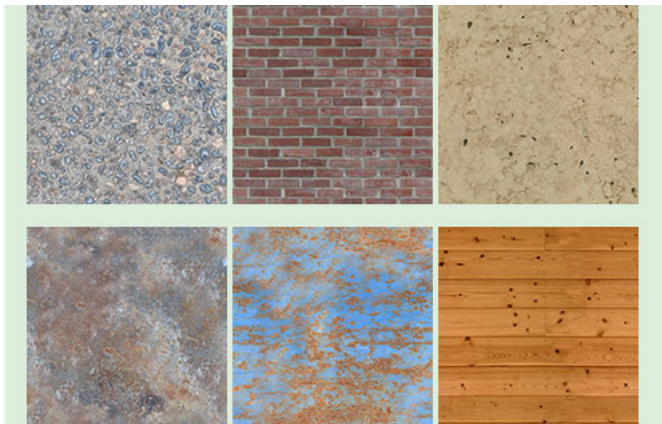
But texture data can be interpreted as anything, e.g. normal vectors, light maps/shadow maps, heightfields, ... (when using programmed shaders).

A data entry in a texture table is called a **texel**.

Use of Textures

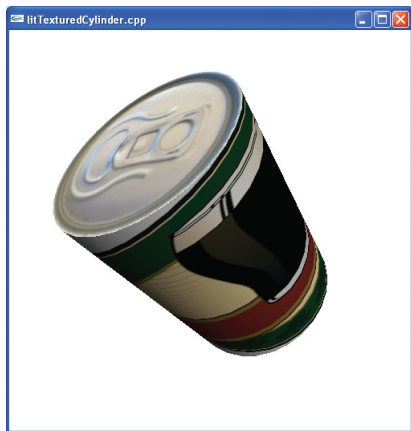
- ▶ Generate detailed graphical content hardly possible with triangles (such clouds, skyboxes, plain pictures (posters, decals) on surfaces).
- ▶ Create illusion of structure, saving lots of triangles. Can be (low level) part of a level-of-detail scheme.
- ▶ Most of a game's graphical expression is via artwork using textures.
- ▶ Hold special-purpose data for use in rendering process.

Examples



(From All Things Designed)

Examples



Examples

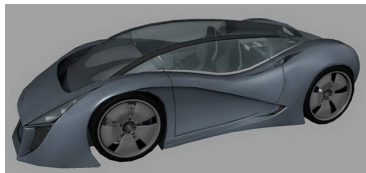
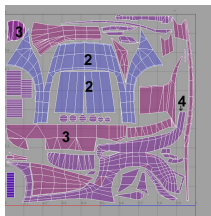
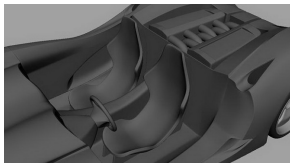
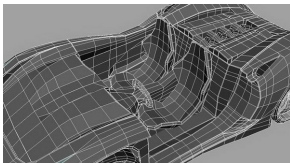


Examples



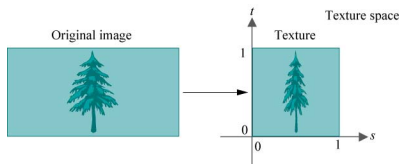
(From Sly Cooper)

Examples



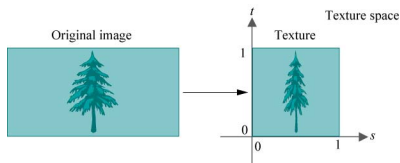
(By Valentin Nadolu)

Texture Coordinates



Texture data get mapped to $[0; 1]^{1,2,3}$ in **texture space**.

Texture Coordinates



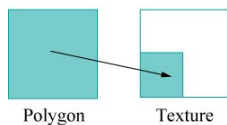
Texture data get mapped to $[0; 1]^{1,2,3}$ in **texture space**.

Vertices can be associated with texture coordinates

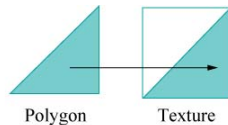
```
glBegin(GL_POLYGON);  
    glTexCoord2f(0.0, 0.0); glVertex3f(-10.0, -10.0, 0.0);  
    glTexCoord2f(1.0, 0.0); glVertex3f(10.0, -10.0, 0.0);  
    glTexCoord2f(1.0, 1.0); glVertex3f(10.0, 10.0, 0.0);  
    glTexCoord2f(0.0, 1.0); glVertex3f(-10.0, 10.0, 0.0);  
glEnd();
```

Texture Coordinates

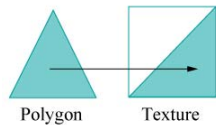
Texture space points can be arbitrary:



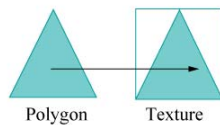
(a)



(b)



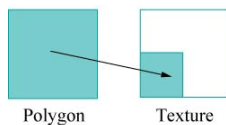
(c)



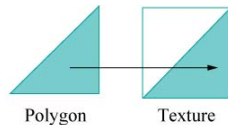
(d)

Texture Coordinates

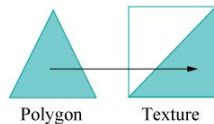
Texture space points can be arbitrary:



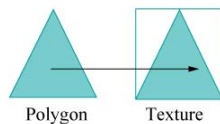
(a)



(b)



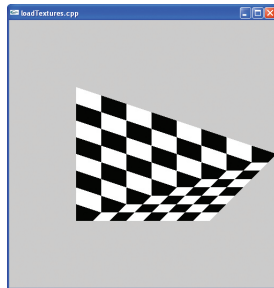
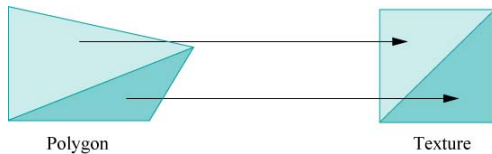
(c)



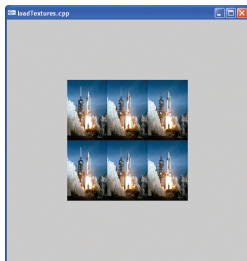
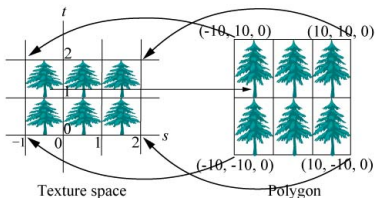
(d)

Points internally in triangle are associated with points in texture space using [interpolation](#) (via barycentric coordinates, as usual).

Interpolation Example



Coordinates Outside the Texture



```
glBegin(GL_POLYGON);  
    glTexCoord2f(-1.0, 0.0); glVertex3f(-10.0, -10.0, 0.0);  
    glTexCoord2f(2.0, 0.0); glVertex3f(10.0, -10.0, 0.0);  
    glTexCoord2f(2.0, 2.0); glVertex3f(10.0, 10.0, 0.0);  
    glTexCoord2f(-1.0, 2.0); glVertex3f(-10.0, 10.0, 0.0);  
glEnd();
```

Above is repetition

Above is **clamping** (on s -axis in texture space).

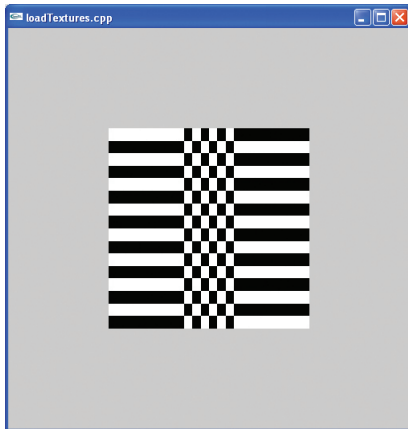
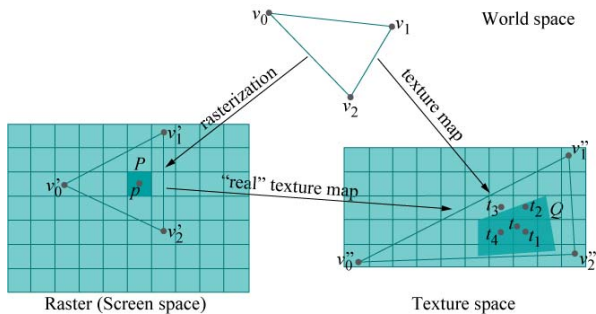


Table Lookup in Texture

Points on triangle do not correspond exactly to (middle of) texture cells.



- ▶ Nearest neighbor
- ▶ Linear interpolation (bilinear: linear interpolation on both s - and t -axes).

Minification/Magnification

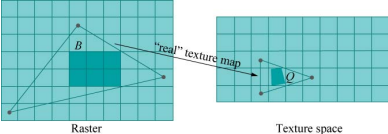
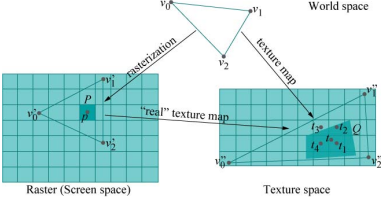
Associated pixels and texels may not be similar in size.

- ▶ Minification: one pixel corresponds to many texels.
- ▶ Magnification: one pixel corresponds to a fraction of a texel.

Minification/Magnification

Associated pixels and texels may not be similar in size.

- ▶ Minification: one pixel corresponds to many texels.
- ▶ Magnification: one pixel corresponds to a fraction of a texel.



Problems:

- ▶ Magnification: Nearest neighbor lookup gives block-like result.
- ▶ Minification: Results of lookup for neighboring pixels are unrelated, and appear random. Even worse if viewing angle changes (flicker from the changing random lookups).

Problems:

- ▶ Magnification: Nearest neighbor lookup gives block-like result.
- ▶ Minification: Results of lookup for neighboring pixels are unrelated, and appear random. Even worse if viewing angle changes (flicker from the changing random lookups).



Remedies::

- ▶ Magnification: Use interpolation.
- ▶ Minification: use [mipmaps](#).

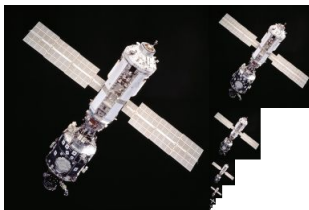
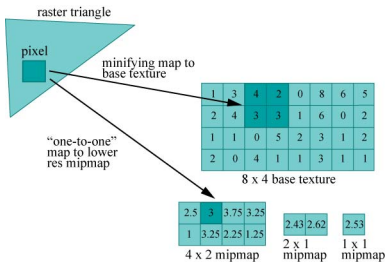
Remedies::

- ▶ Magnification: Use interpolation.
- ▶ Minification: use [mipmaps](#).

Mipmap: Pre-made series of resolutions of a texture.

OpenGL has method for finding (during lookup) which one is closest to matching pixels and texels in size, and makes lookup in that. Or can use linear interpolation on lookups from two neighboring candidate resolutions. (Trilinear interpolation: also do bilinear interpolation in lookups).

Start with high-resolution texture. Create lower-resolution versions by averaging over sets of neighboring texels. (See GLU library.)



(By Mike Hick)

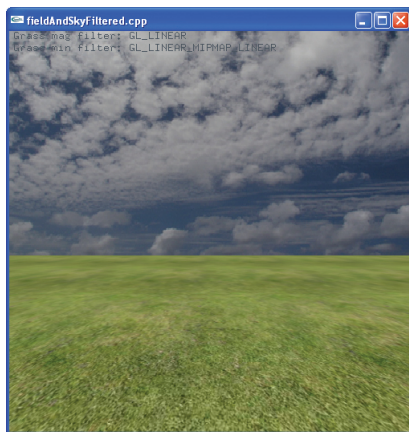
Example

No mipmap, nearest neighbor:



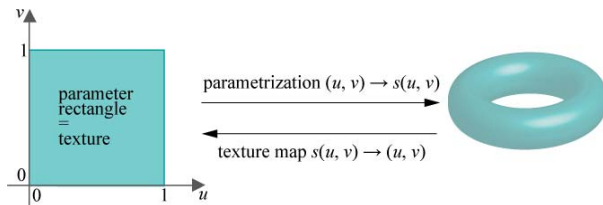
Example

Using mipmap:



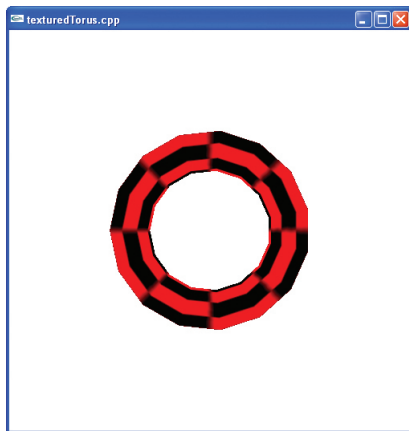
Parametrised Surfaces and Textures

Usually parameters can map simply to texture space



Example

Texture: red checkerboard.



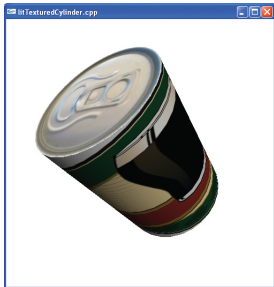
Light and Textures

Light calculation for pixel is independent of texture (not counting uses of texture lookups in programmed shaders).

Texture value may e.g.:

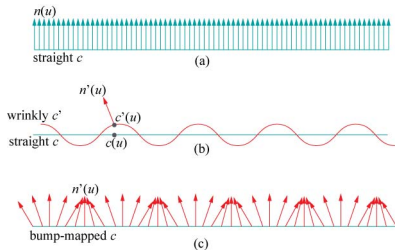
- ▶ Overwrite value from light calculation.
- ▶ Modulate (channel-wise multiplication) value from light calculation.

In the latter case, highlights will often look wrong (should be “outside” of texture). It is possible to have OpenGL save the specular term from light calculation, and **add** it after modulation.



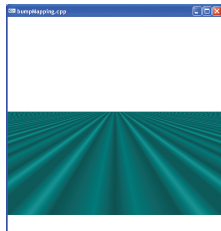
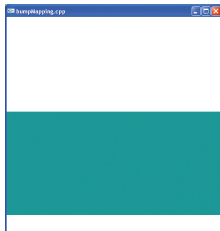
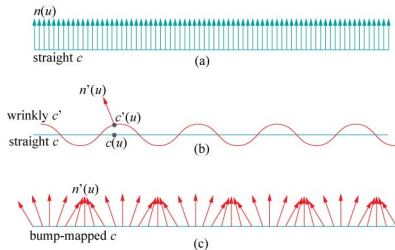
Texture Use: Bumpmapping

Store surface normals (or perturbation of normals) in texture.



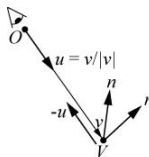
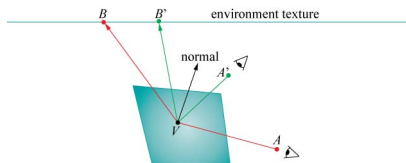
Texture Use: Bumpmapping

Store surface normals (or perturbation of normals) in texture.



Texture Use: Environment Mapping

Reflections can see environment. Make part of shading calculation.



$$r = u - 2(n \cdot u)n$$

Environment Mapping

Easiest with Cube mapping:



Six single textures. Can each be generated [online](#) by rendering from center (even if moving) and saving framebuffer as texture.

Note: environment mapping assumes distance to environment large compared to extent of model (or else, correct lookup would not depend on only vector r).