Textures

Estelle Duveau - estelle.duveau@inria.fr

MoSIG1, Introduction to Computer Graphics, 25/03/2009

Planning

- **Lecture :** introduction to OpenGL
  Lab : first steps in OpenGL and modeling - 25/02/2009
- **Lecture/Lab :** transformations and hierarchical modeling - 04/03/2009
- **Lecture :** lights and materials in OpenGL - 11/03/2009
- **Lab :** lights and materials in OpenGL - 18/03/2009
- **Lecture :** textures in OpenGL
  Lab : textures in OpenGL - 25/03/2009
- **Lab :** procedural animation - 01/04/2009
- **Lab :** physical animation : particle systems - 08/04/2009
- **Lab :** physical animation : collisions - 22/04/2009

Plan

Plan

Reminder

- planar image $I(u, v)$ + mapping $P(x, y, z) \rightarrow (u, v)$
- For each vertex : coordinates + normal + texture coordinates
- $\Rightarrow$ Interpolation of texture coordinates in a face
- Aliasing problems

Texture mapping

Very general, many possibilities :

- Texture = 2D/3D array of data called texels
- Several texture formats available (RGBA, depth, luminance, intensity)
- Several mapping modes available (replace, blend, modulate)
- Rotations/translations before mapping
- Sub or over-sampling
- A texture can be repeated or not on a face
- ...

2D RGBA textures

Texture mapping : process

1. Create or load a array of texels
2. Create a texture object and assign a texture to it
3. Define the mapping parameters
4. Enable texture mapping
5. Draw the scene, specifying how the textures are 'attached' to the objects

Plan

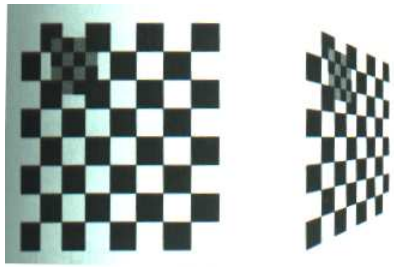
Load a texture

```
glTexImage2D(target,level,internalFormat,
width,height,border,format,type,ptr_texels)
```

- target : generally, `GL_TEXTURE_2D`
- level : if we are using mip-mapping, level-of-detail
- internalFormat : number of color components
- width : width of the texture (must be a multiple of 2)
- height : height of the texture (must be a multiple of 2)
- border : 1 ou 0 : whether or not the texture has a border
- format : format of the pixel data (RGBA, depth, ...)
- type : data type of the pixel data
- ptr_texels : pointer to the image data
- 1D and 3D textures : `glTexImage1D(...)`, `glTexImage3D(...)`

Replace one piece of the texture

A piece of the texture can be replaced by another texture :
`glTexSubImage2D(target,level,xOffset,yOffset,`
`width,height,format,type,ptr_texels)`
→ no need to create a new texture from scratch



Plan

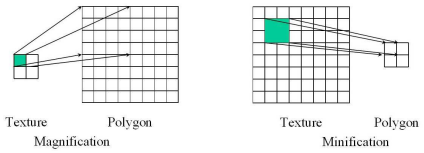
Manage texture objects

- A texture object avoids having to reload a texture at every time step : the data of the texture is stored in memory
- Give a name (an integer) to each of the *n* texture objects :
  `glGenTextures(n,ptr_texNames)`
- Bind and use a texture object :
  `glBindTexture(GL_TEXTURE_2D,texName)`
- Delete *n texture objects* :
  `glDeleteTextures(n,ptr_texNames)`

Plan

Texture filtering

- Repeat/Clamp : see in *Map texture to object*
- If a texel is mapped on several pixels,
  magnification (over-sampling)
  ⇒ `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER,GL_NEAREST or GL_LINEAR)`
- If several texels are mapped on one pixel,
  minification (sub-sampling)
  ⇒ `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,GL_NEAREST or GL_LINEAR)`



Mip-mapping

- Pre-calculated set of the same image at different levels of details
- The renderer switches to the suitable mipmap image depending on the distance from viewpoint to object
- ⇒ Artefacts reduced since the mipmap images are already anti-aliased
- In OpenGL :
  - build mipmaps with `GLU` :
    `gluBuild2DMipmaps(target, level-of-details, width, height, format, type, ptr_texels)`
  - Filters : set magnification and minification filters to
    `GL_(TextureFilter)_MIPMAP_(MipmapFilter)`

Texture functions - 1/2

- The texel can replace, be modulated by, be blended with the color obtained by local illumination
- Texture function : `glTexEnv(target,pname,param)`
- Very general and complex : *see documentation*
- A simple use :
  - `glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,param)`
  - `param` = `GL_DECAL`, `GL_REPLACE`, `GL_MODULATE`, `GL_BLEND`, `GL_ADD` ou `GL_COMBINE`

Texture functions - 2/2

`GL_REPLACE`	(RGB_{tex}, A_{tex})
`GL_MODULATE`	$(RGB_{tex}RGB_{loc}, A_{tex}A_{loc})$
`GL_DECAL`	$((1-A_{tex})RGB_{tex} + A_{tex}RGB_{loc}, A_{loc})$
`GL_BLEND`	$(RGB_{loc}(1-RGB_{tex}) + RGB_{const}RGB_{tex}, A_{tex}A_{loc})$
`GL_ADD`	$(RGB_{tex} + RGBloc, A_{tex}A_{loc})$
...	...

Plan

Enable texture mapping

`glEnable(GL_TEXTURE_2D)`

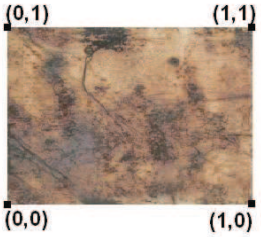
Example :

```
GLubyte Texture[16] = {0,0,0,0, 0xFF,0xFF,0xFF,0xFF,
0xFF,0xFF,0xFF,0xFF, 0, 0, 0, 0 };
GLuint Name;
void InitGL() {
          ⋮
    glEnable(GL_TEXTURE_2D);
    glGenTexture(1, &Name);
    glBindTexture(GL_TEXTURE_2D, Name);
    glTexImage2D(GL_TEXTURE_2D, 0, 4, 2, 2, 0, GL_RGBA,
GL_UNSIGNED_BYTE, Texture);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);
          ⋮
}
```


Plan

Texture coordinates

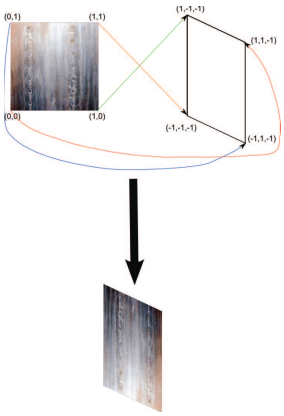
- For each vertex, specify the texture coordinates (s, t)
- → **2D texture considered a square of side-length** 1



- `glTexCoord2f(coords)`

Example

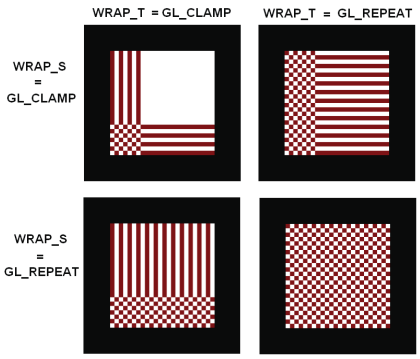
```
glBegin(GL_QUADS);
   glTexCoord2i(1,1);
   glVertex3i(-1,-1,-1);

   glTexCoord2i(1,0);
   glVertex3i(+1,-1,-1);

   glTexCoord2i(0,0);
   glVertex3i(+1,+1,-1);

   glTexCoord2i(0,1);
   glVertex3i(-1,+1,-1);
glEnd();
```



Repeat/Clamp texture

What if texture coordinates greater than 1?

⇒ `glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S,GL_REPEAT)`



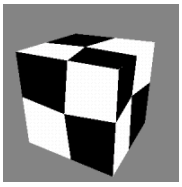
Plan

glHint()

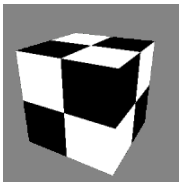
- `glHint(target,hint)` controls some behaviours of OpenGL
- hint = `GL_FASTEST`, `GL_NICEST` or `GL_DONT_CARE`
- target = `GL_LINE_SMOOTH_HINT` (anti-aliasing), `GL_FOG_HINT`, `GL_TEXTURE_COMPRESSION_HINT`, ...

Example :

```
glHint(GL_PERSPECTIVE_CORRECTION_HINT,GL_NICEST)
```



Without



With

Plan

Conclusion

- Done :
  - Texture mapping process
- Highlights :
  - Many possibilities/parameters
  - Filters
- To do :
  - Lab session : experiment textures
  - Next subject : animation