8. Texture Mapping

8.1 2D texture mapping

8.2 OpenGL texture mapping example

8.3 Notes on OpenGL texturing code

8.4 Texture mapping of surfaces

8.5 Bump mapping

8.6 Displacement mapping

8.7 3D Texturing

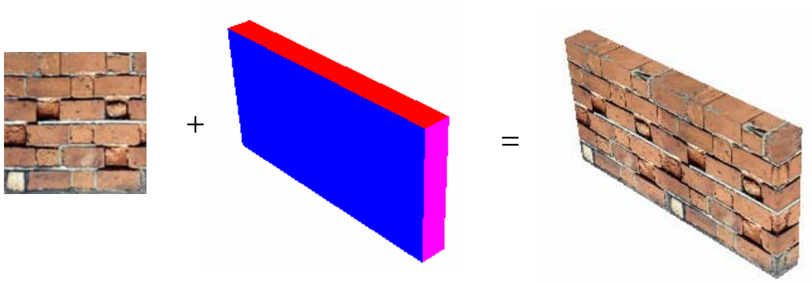
8.8 Procedural textures

8.9 Texture synthesis

8.1 2D Texture Mapping

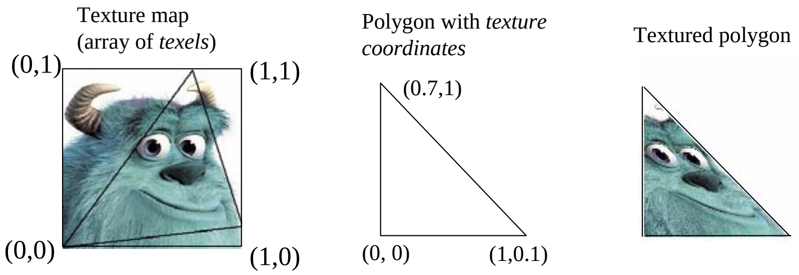
Modelling a large brick with individual polyhedral bricks is very cumbersome.

A much easier solution is to model each side of the wall with a single polygon and to map the image of a brick wall onto it.



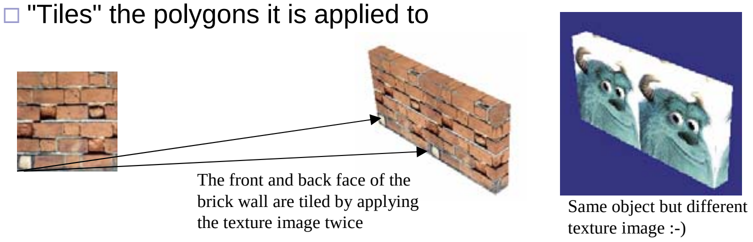
2D Texture Mapping (cont'd)

- Simple idea - Take a (rectangular) image to "stick on" to polygon
  - □ WARNING: Image width and height must be a power of 2 (e.g. 256 pixels)
- Associate with the bottom-left, bottom-right, top-left and top-right corner of the image the *texture coordinates* (0,0), (1,0), (0,1), and (1,1), respectively.
- Specify texture coordinates (*s*,*t*) for each vertex of the polygon.



2D Texture Mapping (cont'd)

- Texture values may be
  - □ copied directly to the output image, or
  - □ used to modulate (i.e. multiply) the colour or alpha value of the shaded polygon, or
  - □ blended with shaded polygon
- Can specify that texture map repeats indefinitely
  - □ "Tiles" the polygons it is applied to

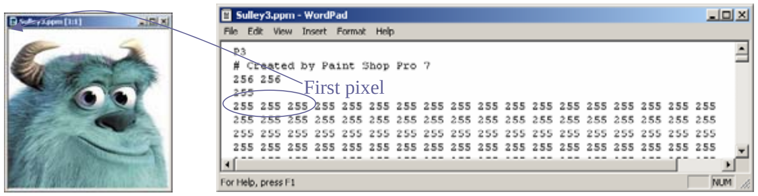

The front and back face of the brick wall are tiled by applying the texture image twice
Same object but different texture image :-)

8.2 OpenGL Texture Mapping Example

- Three things to do
  - □ Construct or read in the texture map, which is a 2D array of RGB or RGBA values (3 or 4 bytes)
  - □ Set up texturing context in OpenGL
    - Pass it the texture map
    - Define parameters like tiling of texture, interpolation method
  - □ Display polygons and set texture coordinate of each vertex using `glTexCoord2*` (before the `glVertex3*` call)

OpenGL Texture Mapping Example (cont’d)

- An easy format for storing images is ‘ppm’ (Portable Pixel Map).
  - □ The first "line" is a magic PPM identifier, it can be "P3" or "P6".
  - □ The next line consists of the width and height of the image.
  - □ The last part of the header gives the maximum value of the colour components for the pixels.
  - □ In addition to the above required lines, a comment can be placed anywhere with a "#" character, the comment extends to the end of the line.
  - □ The following lines specify the pixels of the image as RGB components (left to right, top to bottom).



```
P3
# Created by Paint Shop Pro 7
256 256
255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
```

OpenGL Texture Mapping Example (cont’d)

- Store the texture into a 2D array of RGBA values. In C this corresponds to a 1D array of size `width*height*4`.

```
GLubyte *texture;     // The texture image
int texName;          // ID of texture

// load texture
ifstream textureFile;
textureFile.open("Sulley3.ppm", ios::in);
if (textureFile.fail()){
    cout << "\n Error loading the texture";
    cout.flush(); exit(0);}
skipLine(textureFile); skipLine(textureFile);
textureFile >> textureWidth;
textureFile >> textureHeight;
int maxRGBValue; textureFile >> maxRGBValue;
texture = new GLubyte[textureWidth*textureHeight*4];
```

OpenGL Texture Mapping Example (cont’d)

- Want that the first texel of the texture map corresponds to the texture coordinate (0,0) (which is at the bottom-left of the image)
  - □ Have to reverse columns of the image, i.e. the first rows of pixels in the image becomes the last rows of texels in the texture map.

```
int m,n,c;
for(m=textureHeight-1;m>=0;m--)
   for(n=0;n<textureWidth;n++){
         textureFile >> c;
         texture[(m*textureWidth+n)*4]=(GLubyte) c;
         textureFile >> c;
         texture[(m*textureWidth+n)*4+1]=(GLubyte) c;
         textureFile >> c;
         texture[(m*textureWidth+n)*4+2]=(GLubyte) c;
         texture[(m*textureWidth+n)*4+3]=(GLubyte) 255;
   }
textureFile.close();
```

OpenGL Texture Mapping Example (cont’d)

```
// Ask OpenGL to generate a unique ID for the texture
glGenTextures(1, &texName);

// Do the rest once for each texture map (only 1 in this case)
glBindTexture(GL_TEXTURE_2D, texName);
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_NEAREST);
glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER,
    GL_NEAREST);
glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA, textureWidth, textureHeight,
                0, GL_RGBA, GL_UNSIGNED_BYTE, texture);
delete[] texture;
```

"Wrap" both s and t (i.e. "tile" the texture) if texture parameters are >1.

Take "nearest texel" from texture map

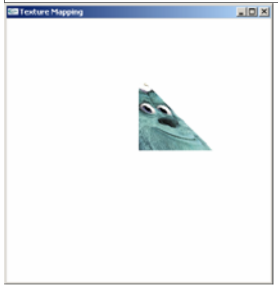
OpenGL Texture Mapping Example (cont’d)

```
// Output of  the textured triangle
// Specify which texture to use (if multiple loaded)
glBindTexture(GL_TEXTURE_2D, texName);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);
glEnable(GL_TEXTURE_2D);

glBegin(GL_TRIANGLES);
glTexCoord2f(0,0);
glVertex2f(0,0);
glTexCoord2f(1,0.1);
glVertex2f(1,0);
glTexCoord2f(0.7,1);
glVertex2f(0,1);
glEnd();

glDisable(GL_TEXTURE_2D);
```

Replace each pixel on the polygon by the texture (default - could omit this line)



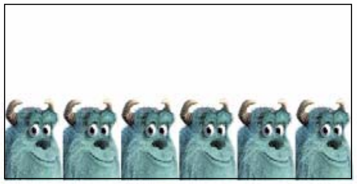
8.3 Notes on OpenGL Texuring Code

- `glGenTextures(GLsizei n, GLuint *textures )` is a request for a given number of “texture handles”
  - □ “Handles” are just integers used to identify texture maps. They are returned in the given `int` array.
  - □ In our example, only one is required.
- `glBindTexture(GLenum target, GLuint texture )` selects one of the texture maps as the current one
  - □ target is `GL_TEXTURE_2D` if using a 2D texture.
- `glTexParameteri` is used to set various parameters of the texture map.

Notes on OpenGL Texuring Code (cont’d)

- The parameters `GL_TEXTURE_WRAP_S` and `GL_TEXTURE_WRAP_T` specify how texture parameters (s,t) outside the unit square (0,0) to (1,1) should be handled
  - □ The `repeat` option tiles texture space with the given map
  - □ Alternatively can `clamp` texture coordinates to the range [0,1].

```
glBegin(GL_QUADS);
glTexCoord2f(0,0);
glVertex3f(0,0,0);
glTexCoord2f(6,0);
glVertex3f(2,0,0);
glTexCoord2f(6,2);
glVertex3f(2,1,0);
glTexCoord2f(0,2);
glVertex3f(0,1,0);
glEnd();
```



```
glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_WRAP_S, GL_REPEAT);

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

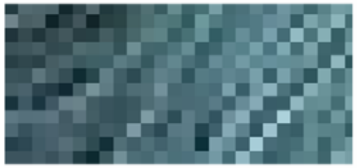
Notes on OpenGL Texuring Code (cont'd)

- The parameter `GL_TEXTURE_MAG_FILTER` determines how to interpolate between texel values when the pixel size is less than the texel size
  - □ `GL_NEAREST` or `GL_LINEAR`

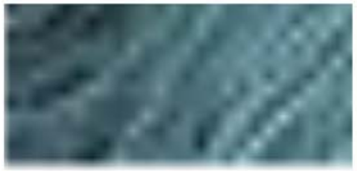
Example: map a small part of the texture image onto a large polygon

```
glBegin(GL_QUADS);
glTexCoord2f(0,0);
glVertex3f(0,0,0.2);
glTexCoord2f(0.1,0);
glVertex3f(2,0,0.2);
glTexCoord2f(0.1,0.05);
glVertex3f(2,1,0.2);
glTexCoord2f(0,0.05);
glVertex3f(0,1,0.2);
glEnd();
```

```
glTexParameteri(
GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER,
GL_NEAREST);
```



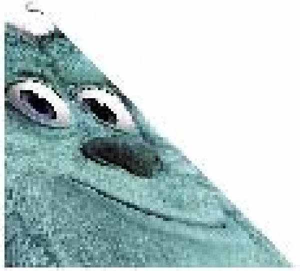
```
glTexParameteri(
GL_TEXTURE_2D,
GL_TEXTURE_MAG_FILTER,
GL_LINEAR);
```



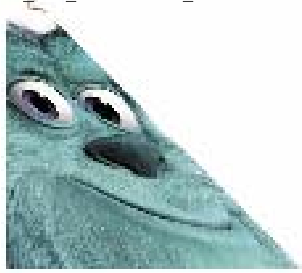
Notes on OpenGL Texuring Code (cont'd)

- The parameter `GL_TEXTURE_MIN_FILTER` determines how to average texel values when the pixel size is greater than the texel size
  - □ Either `GL_NEAREST` or `GL_LINEAR` or one of various `MIP_MAPPING` options (see "The OpenGL Programming Guide")

```
glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER, GL_NEAREST);
```

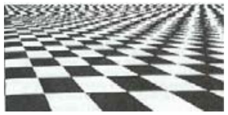


```
glTexParameteri(GL_TEXTURE_2D,
GL_TEXTURE_MIN_FILTER, GL_LINEAR);
```

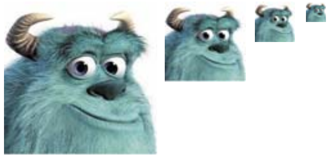


Notes on OpenGL Texuring Code (cont'd)

- Get aliasing problems if sampling texture at low sampling rates
  - □ e.g. chequerboard pattern applied to a ground plane that extends to the horizon



- Aliasing avoided by storing texture at different resolutions (each 1/2 the linear size)
  - □ Called *mipmap* (mip = *multim in parvo* meaning *many things in small place)*
  - □ hardware selects appropriate resolution image
  - □ can also bilinearly interpolate in selected map (so have *trilinearly interpolated textures*)



etc to 1 x 1 pixel image = average colour

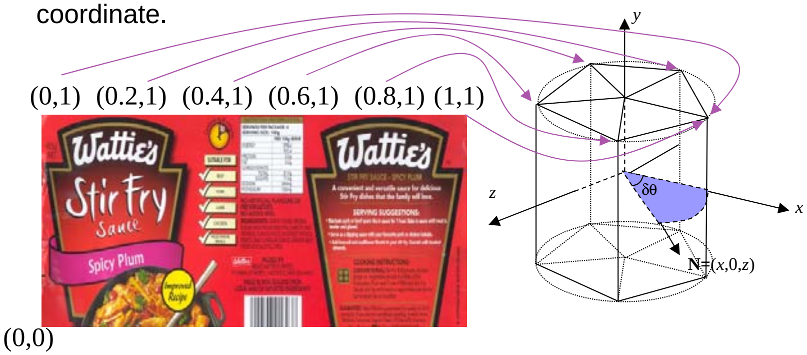
Complete mipmap requires only 1/3 more memory than the original.

Notes on OpenGL Texuring Code (cont'd)

- `glTexImage2D()` sets up the texture image for the currently selected texture
  - □ Lots of options for size, format, MIPMAP level etc
    - (see "The OpenGL Programming Guide")
- `glTexEnv[fi]` sets up the texturing mode to one of:
  - □ GL_REPLACE We only use this
    - Texture overwrites computed pixel colour
  - □ GL_DECAL
    - Texture colour is blended with pixel colour
  - □ GL_MODULATE Useful to texture illuminated surfaces.
    - Multiplies pixel colour by texture colour
  - □ GL_BLEND
    - Uses texture value as an "alpha" to determine how much of a pre-specified "texture environment colour" to blend with the pixel colour

8.4 Texture Mapping of Surfaces

- Up to now we only mapped textures onto individual polygons.
- Mapping a texture onto a surface works analogously: simply associate each vertex of the polygon mesh with a texture coordinate.



Texture Mapping of Surfaces (cont'd)

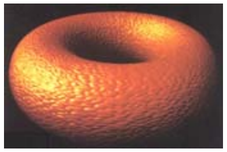
```
glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
float x,z,theta,s, bottomY=-1, topY=1.5;

glBindTexture(GL_TEXTURE_2D, texName);
glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_REPLACE);
glEnable(GL_TEXTURE_2D);
glBegin(GL_QUAD_STRIP);
for (int segment=0; segment <= NUM_SEGMENTS; segment++){
   s=(float) segment/(float) NUM_SEGMENTS;
   theta=2.0f*Pi*s;
   x = (float) cos(theta);
   z = (float) sin(theta);
   glTexCoord2f(1-s,0);
   glVertex3f(x,bottomY,z);
   glTexCoord2f(1-s,1);
   glVertex3f(x,topY,z);
}
glEnd();
glDisable(GL_TEXTURE_2D);
```

8.5 Bump Mapping

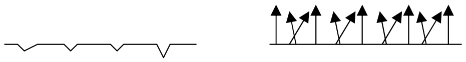
Not Exam Relevant

- Normal 2D texture mapping doesn't look right for uneven surfaces like orange skin
  - □ Texture should vary with illumination



© Jim Blinn, University of Utah

- Can get strong illusion of uneven surface by using a *bump map* to modulate the surface normal before applying illumination calculations.



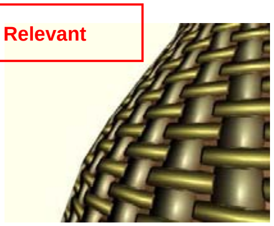
- Part of *Direct3D*
- Not a standard OpenGL capability, but can be done, eg.. http://vcg.isti.cnr.it/activities/geometryegraphics/bumpmapping.html

8.6 Displacement Mapping

Not Exam Relevant

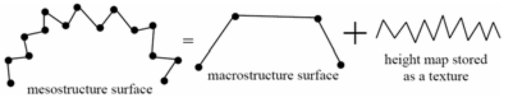
Bump mapping

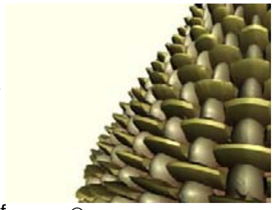
- Modify the normal at each point, rendered geometry is flat.



Displacement mapping

- Modify the position and normal of each surface point **during** rendering.





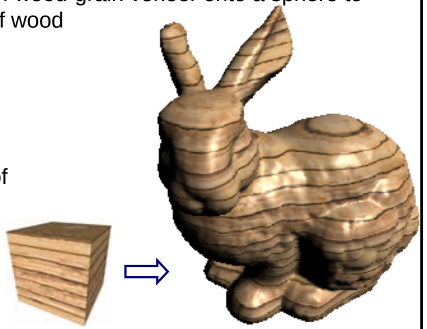
- Usually done using fragment shaders
  - □ e.g. http://www.iit.bme.hu/~szirmay/egdisfinal3.pdf

© http://atec.utdallas.edu/midori/Handouts/texture_mapping.htm

8.7 3D Texturing

Not Exam Relevant

- 2D texture maps very limited for representing objects made from textured 3D material like wood or marble
  - □ e.g., consider how to stick a sheet of wood-grain veneer onto a sphere to make it look like it was carved out of wood
- ◆ 3D Texturing
  - – 3D volume of texels
  - – Each polygon vertex has a 3D texture coordinate
  - – If polygon is rendered the position of a pixel in the 3D texture space is determined and the corresponding RGB/RGBA value is used.
  - – Available as an OpenGL extension (GeForce3)

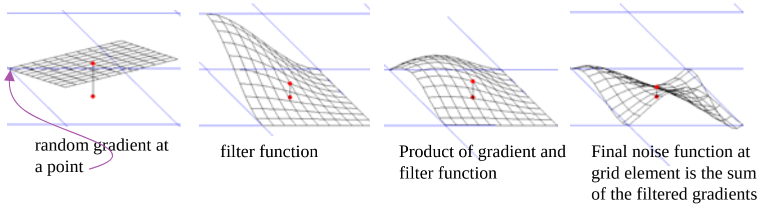


© 2008, Felix Manke & Burkhard Wuensche, University of Auckland

8.8 Procedural Textures

Not Exam Relevant

- Can get arbitrarily high resolution 3D textures for natural materials like wood and marble by use of *procedural textures*
- Done by perturbing a simple geometric model with a band-limited noise function
  - □ e.g. for wood, model may be perfectly cylindrical trunk with perfect rings
  - □ Noise function obtained by summing the filtered random gradients at the grid points.

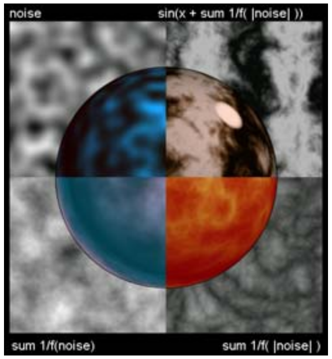

random gradient at a point
filter function
Product of gradient and filter function
Final noise function at grid element is the sum of the filtered gradients at its vertices.

© Images from Ken Perlin

Procedural textures (cont'd)

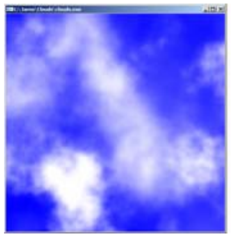
Not Exam Relevant

- Perlin Noise
  - □ http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
  - □ http://mrl.nyu.edu/~perlin/doc/oscar.html



Each quadrant shows a noise texture and the noise texture mapped onto a sphere with additional colour and lighting parameters.

© 2003, Ken Perlin



© 2004, Jarno van der Linden, Graphics Group, University of Auckland

8.9 Texture Synthesis

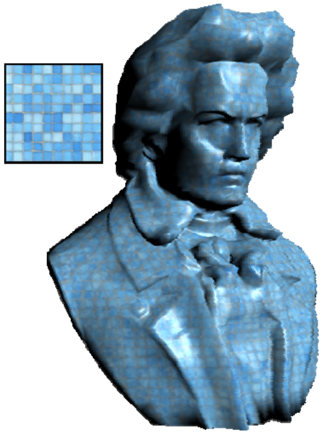
Not Exam Relevant

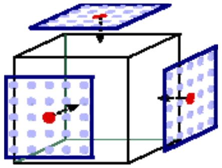
- Create a large texture from a small input exemplar
- Many methods, e.g. for each pixel in the output texture find a pixel in the input exemplar with similar neighbourhood (colour and appearance space attributes).

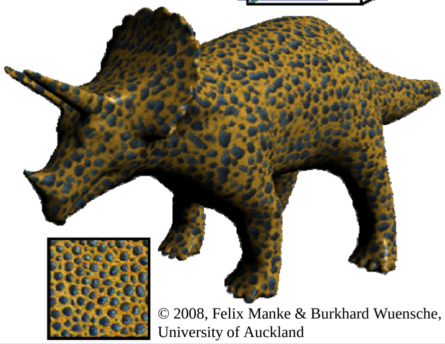
© 2008, Felix Manke & Burkhard Wuensche, University of Auckland

Not Exam Relevant

3D Texture Synthesis



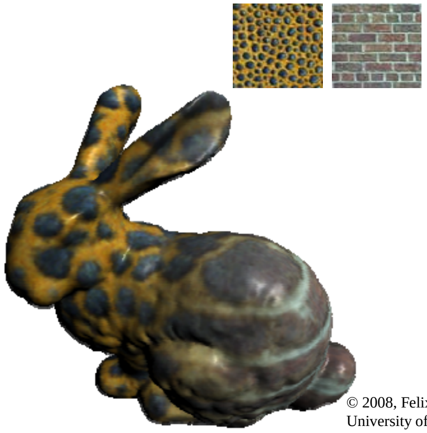


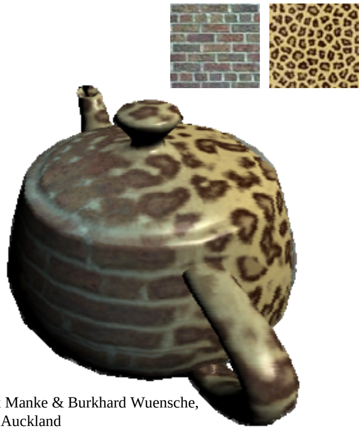


© 2008, Felix Manke & Burkhard Wuensche, University of Auckland

Texture morphing

Not Exam Relevant





© 2008, Felix Manke & Burkhard Wuensche, University of Auckland