UCL

Texture generation and Texture mapping

V1.1.1

Anthony Steed

Based on slides from Celine Loscos (v1.0)

UCL

Textures and Computer Graphics

- Polygons can't do everything to a model
- Textures add realism without increasing the number of polygons
- But there's more to it. Textures can:
  - Be used to render things impossible with polygons
  - Provide a power tool to fundamental graphics primitives
  - Be creative!

UCL

Overview

1. Texture mapping
   - Parameterisation
   - Texture projection, offline and online
   - Blending
   - Texture coordinate generation
   - Multi-texture
   - Other texture modes
2. Texture generation
   - Texture capture
   - Texture synthesis
     - Statistical textures
     - Procedural textures

UCL

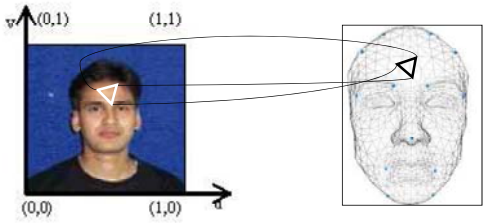
Texture mapping

- What we've seen in 3071/GV04
  - Forward and inverse mapping
  - Mipmapping (to reduce aliasing)

UCL

Parameterisation

- Problem: map (u,v) coordinates to (x,y,z) coordinates
- Can be resumed at finding the right parameterisation



UCL

Example

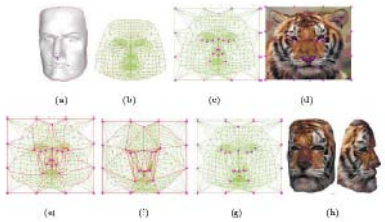


Figure 2.4: Match maker algorithm Example: Mapping a tiger face onto a human. (a)3D model with feature vertices(b) plan input mesh (c) Parameterization with virtual boundary (d) Texture with feature points (e) Matching triangulation (f) Triangle embedding. (g) Mesh after constrained smoothing. (h) Resulting textured model.Images courtesy of Kraevoy et al.[10]

UCL

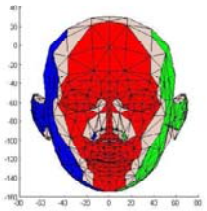
Coordinate Generation

- GPUs can generate texture coordinates, based on distance in eye, object or local coordinates
- Useful for "projection" of textures on to objects
- Useful for non-photorealistic rendering

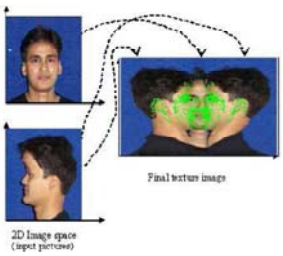
UCL

Blending

- Textures can come from different viewpoints
  - Example of view point selection



- Blending in the overlapping area



UCL

Texture mapping with OpenGL

- Fixed-function pipeline
  - Configuring texture-related state
  - Loading the texture(s) to the card
  - Setting the texture-indices to render
  - Per vertex supplying (multi)-texture coordinates
- Shader-enabled pipeline
  - ditto plus ..
  - Dependent texture reads (calculate new texture coordinates)
  - Much more general colour calculation per fragment

UCL

Loading a texture with OpenGL

- Assign texture ID
  `glBindTexture(GL_TEXTURE_2D, id); //id is an integer`
- Method to pack pixels
  `glPixelStorei(GL_UNPACK_ALIGNMENT, 1); //1 byte per component, aligned as the texture data is loaded`
- Type of wrapping and filtering

```
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);
```

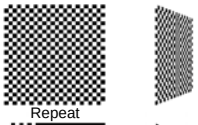
- Interaction of the texture with other information on the polygon
  `glTexEnvf(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, GL_MODULATE);`
- Load the image data

```
glTexImage2D (GL_TEXTURE_2D, 0, GL_RGB, imageWidth, imageHeight, 0,
GL_RGB, GL_UNSIGNED_BYTE, imageData);
```

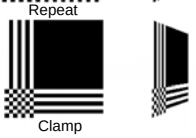
UCL

Different loading methods

- Types of wrapping
  - GL_REPEAT or GL_CLAMP
- Types of filtering
  - GL_TEXTURE_MAG_FILTER:
    - GL_NEAREST or GL_LINEAR
  - GL_TEXTURE_MIN_FILTER:
    - GL_NEAREST, GL_LINEAR, GL_NEAREST_MIPMAP_NEAREST, GL_NEAREST_MIPMAP_LINEAR, GL_LINEAR_MIPMAP_NEAREST, GL_LINEAR_MIPMAP_LINEAR
- Types of interaction
  - GL_DECAL (replacing the stored colours), GL_MODULATE (multiply texture by stored colour), GL_BLEND (use luminence to blend texture and colour)
- Types of data
  - GL_RGB, GL_RGBA, GL_RED, GL_BLUE, GL_GREEN, GL_UNSIGNED_BYTE, GL_UNSIGNED_INT, GL_FLOAT
  - (other available at http://www.opengl.org/documentation/specs/man_pages/hardcopy/GL/html/gl/teximage2d.html)



Repeat



Clamp

Repeat + Clamp

UCL

Rendering a textured polygon

```
glEnable(GL_TEXTURE_2D);
…
glBindTexture (GL_TEXTURE_2D, id);
glBegin (GL_QUADS);
glTexCoord2f (0.0, 0.0);
glVertex3f (0.0, 0.0, 0.0);
glTexCoord2f (1.0, 0.0);
glVertex3f (10.0, 0.0, 0.0);
glTexCoord2f (1.0, 1.0);
glVertex3f (10.0, 10.0, 0.0);
glTexCoord2f (0.0, 1.0);
glVertex3f (0.0, 10.0, 0.0);
glEnd ();
```

UCL

Multitexturing

- Similar but allows to apply multiple textures on an object
  - Useful to integrate multiple effects together: lighting, shadows, bumps, texture image, etc.
- Depending on your graphics card, up to 16 or more
- Use `EXT_texture_env_combine` or `ARB_texture_env_add`
- Activating

```
glActiveTextureARB(GL_TEXTURE0_ARB);
glBindTexture(GL_TEXTURE_2D, tex0);
glEnable(GL_TEXTURE_2D);
glActiveTextureARB(GL_TEXTURE1_ARB);
glBindTexture(GL_TEXTURE_2D, tex1);
glEnable(GL_TEXTURE_2D);
```

- Rendering

```
glBegin(GL_TRIANGLES);
glMultiTexCoord2fvARB(GL_TEXTURE0_ARB, &t0[0]);
glMultiTexCoord2fvARB(GL_TEXTURE1_ARB, &t1[0]);
glVertex3fv(&v[0]);
glMultiTexCoord2fvARB(GL_TEXTURE0_ARB, &t0[1]);
glMultiTexCoord2fvARB(GL_TEXTURE1_ARB, &t1[1]);
glVertex3fv(&v[1]);
glMultiTexCoord2fvARB(GL_TEXTURE0_ARB, &t0[2]);
glMultiTexCoord2fvARB(GL_TEXTURE1_ARB, &t1[2]);
glVertex3fv(&v[2]);
glEnd();
```

UCL

Example

- From L3D

Texture Unit 0

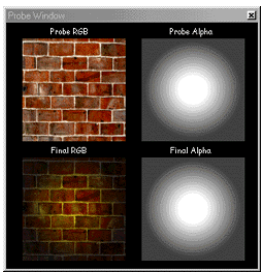
Texture Unit 1

Textured Cube

UCL

Example

- From ATI



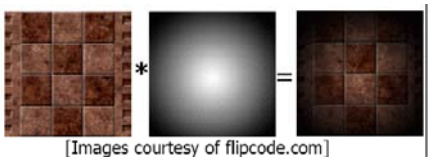
UCL

Applications

- Light mapping
- Environment mapping
- Per-pixel lighting
- Bump mapping/displacement maps
- And others
  - Volumetric textures, Texture shading animated textures, projective textures, …

UCL

Light mapping



[Images courtesy of flipcode.com]

[Images courtesy of www.gamasutra.com]

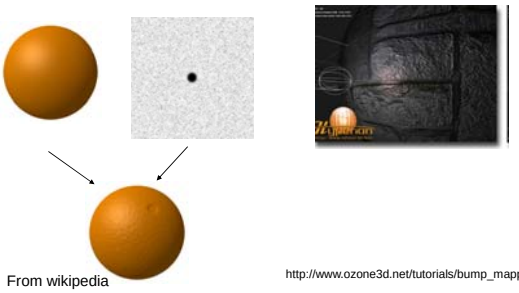
UCL

Bump mapping/Displacement maps

- To deal with depth effects
- Bump mapping
  - Perturbe surface normals in order to simulate light effects on a bumpy surface
- Displacement maps
  - Displace the actual position of points on a surface

UCL

Bump mapping



From wikipedia

http://www.ozone3d.net/tutorials/bump_mapping.php

UCL

Bump Mapping Code

- "Bump Map" contains
  - Normal offset in u,v space
- Created from a height map
  - Take gradient
- Not self-shadowing
- Good for shallow features
- Very common, as very cheap

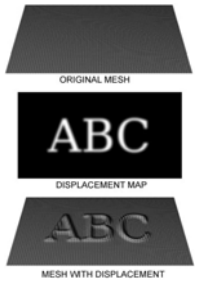
UCL

Relief Textures

- Actually simulate displacement of the surface
  - Motion parallax
  - Self-occlusion
  - Self-shadowing
- Involves a highly optimised ray-trace in local texture coordinates
- Slow, but starting to be used
- Good for deep features, even whole 3D objects as image imposters
  - See Week 6 notes on "Image Imposters"

UCL

Displacement (reflief) maps



From wikipedia

Oliveira, Manuel M., Gary Bishop, David McAllister.
Relief Texture Mapping. Proceedings of SIGGRAPH 2000 (New Orleans, La), July 23-28, 2000, pp.359-368.

UCL

Environment Maps

- Simulate a reflection off an object
- Index a texture by the *normal* not a mapped u,v coordinate
- Very cheap

- Vertex Shader

```
void ftexgen(in vec3 normal, in vec4 ecPosition)
{
    gl_TexCoord[0] = vec4( normal, 1.0 );
}
```

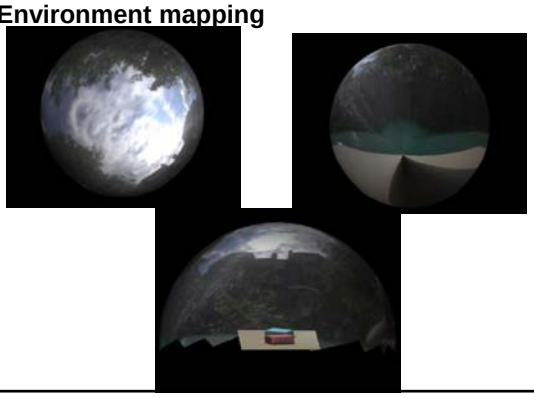
- Fragment Shader

```
uniform sampler2D texUnit0;
void main (void)
{
   vec4 color;
   color = gl_Color;
   color *= texture2D(texUnit0, gl_TexCoord[0].xy);
   gl_FragColor = color;
}
```

Code snippets from ShaderGen

UCL

Environment mapping



UCL

Overview

1. Texture mapping
   - Parameterisation
   - Texture projection, offline and online
   - Blending
   - Texture coordinate generation
   - Multi-texture
   - Other texture modes
2. Texture generation
   - Texture capture
   - Texture synthesis
     - Statistical textures
     - Procedural textures

UCL

Texture generation

- To map a texture, you need to have a texture!
- How?
  - Texture capture
  - Texture synthesis
    - Statistical textures
    - Procedural textures

UCL

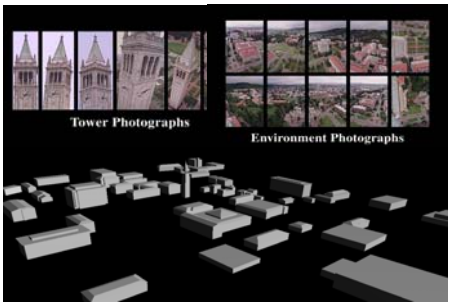
Texture Capture

- Simply take a photograph
- Better if you're approximating an orthographic projection
  - You can capture from reality or a rendered environment
- Used to
  - Model realistic environments
  - Render precalculated information (lighting, shadows)

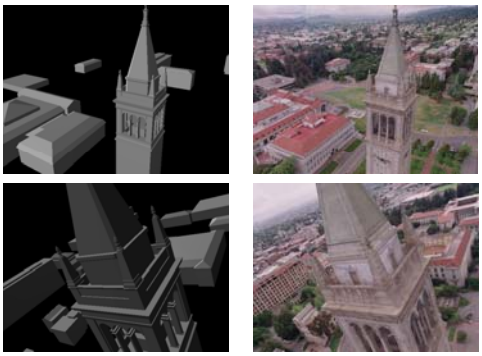
UCL

Modelling from Texture

- Extract images from calibrated photography



UCL



UCL

Problems with Captured Textures

- Tend to repeat in a scene
  - Problems can occur when two polygons with the same texture connect (discontinuities)
- Resolution doesn't always adapt to the polygon size
- May contain undesirable effects (lighting highlights, shadows) present in the original photograph
- Difficult to get a good picture of a surface such as a building
  - Cars, pedestrians, trees, other buildings in the way
- Good textures are a skill
  - Hence the price of texture compendia

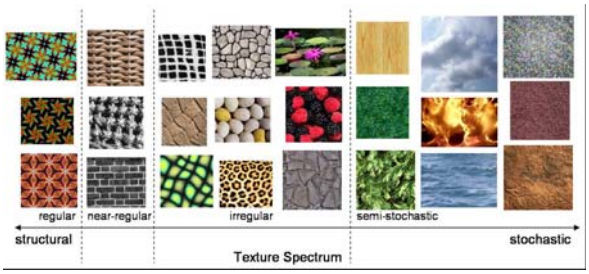
UCL

Texture Synthesis

- Instead of capturing textures, create them automatically!
- Textures are treated differently whether they are structural or regular
- Idea:
  - Generate a sample or start with a sample
  - Create a new texture from this sample
- Useful
  - To avoid large textures
  - To reduce repetition: each generated texture can be different while being similar
  - To create new textures
  - To fill gaps

UCL

Different texture types



From Yanxi Liu, http://www.cs.cmu.edu/~wclin/nrt.htm

UCL

Stochastic

- Perlin noise

[Vase with Perlin noise, Image courtesy of graphics.lcs.mit.edu]

- Heeger and Bergen

UCL

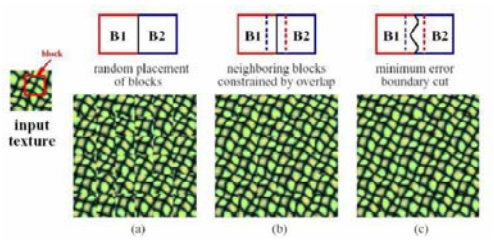
Texture Expansion

- Creating a larger texture from a smaller one
- Saves time, makes sure the texture has certain statistical properties
  - E.G. No luminance gradient that you might get if you took a larger image
- Most of the techniques work by “local expansion”
  - Do some sort of local neighbourhood statistics
  - Create a new texture, by moving and swapping similar neighbourhoods

UCL

Image Quilting

- Image Quilting for Texture Synthesis and Transfer, Alexei A. Efros and William T. Freeman,Proceedings of SIGGRAPH '01, Los Angeles, California, August, 2001.



UCL

Results

UCL

Summary and conclusion

- The graphics hardware is well-equipped to manage textures
- Research has been done in
  - Texture mapping techniques
  - Texture generation
- We've seen a few examples but many more exist in the literature!