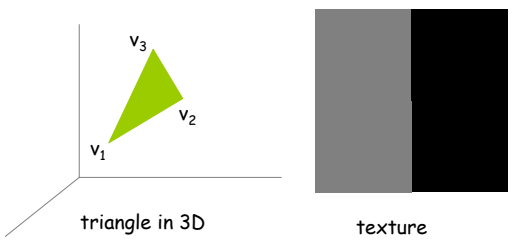
TEXTURE MAPPING

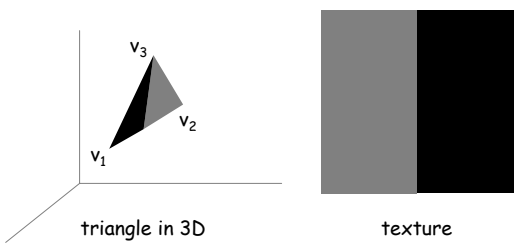
Overview

- Basic texture mapping
- Variations
- Advanced techniques

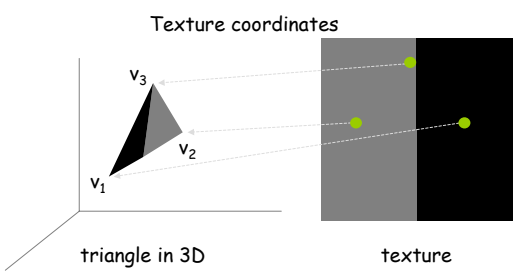
Let start with a simple problem …



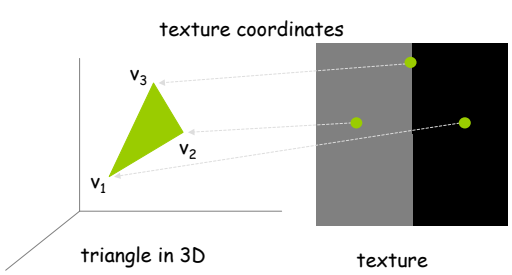
We want to "glue" the image onto the triangle.



How should the texture be placed?

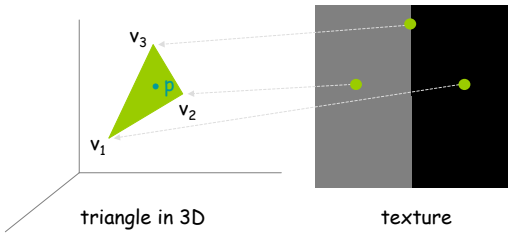


this is the input to our algorithm



Our algorithm answers this question:

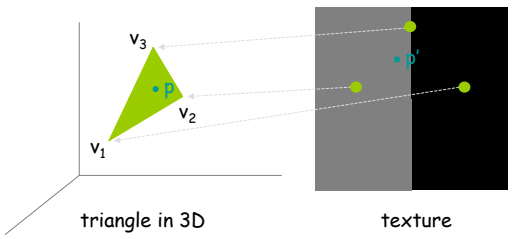
"What color is point p on the texture mapped triangle?"



Algorithm

1. Find the texture coordinates for p.
2. Find the color of the texture at the texture coordinates for p.

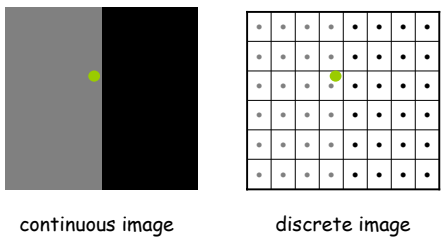
Use barycentric coordinates to find p'



Algorithm

1. Find the texture coordinates for p.
   Use barycentric coordinates.
2. Find the color of the texture at the texture coordinates for p.

Resample to find texture color



Algorithm

1. Find the texture coordinates for p.
   Use barycentric coordinates.
2. Find the color of the texture at the texture coordinates for p.

   Resample.

Basic Texturing in OpenGL

- Create the texture object
- Describe how texture should be applied
- Enable texturing
- Render scene supplying texture coordinates

Basic Texturing in OpenGL (cont.)

Create the texture object

```
void glTexImage2D(    GLenum target,
                      GLint level,
                      GLint internalFormat,
                      GLsizei width, GLsizei height,
                      GLint border,
                      GLenum format,
                      GLenum type,
                      const GLvoid *pixels );

Example:
    glTexImage2D( GL_TEXTURE_2D, 0, GL_RGBA, 128, 128, 0, GL_RGBA,
                      GL_UNSIGNED_BYTE, myTexureData );
```

Basic Texturing in OpenGL (cont.)

Specify how texture should be applied

```
void glTexEnvi( GLenum target, GLenum pname, GLint param );
void glTexEnvf( GLenum target, GLenum pname, GLfloat param );
void glTexEnviv( GLenum target, GLenum pname, Glint* param );
void glTexEnvfv( GLenum target, GLenum pname, GLfloat* param );
```

```
Example:
void glTexEnvf( GL_TEXTURE_ENV , GL_TEXTURE_ENV_MODE, GL_REPLACE );
```

Basic Texturing in OpenGL (cont.)

Enable and render

No Mapping

```
glBegin( GL_TRIANGLES);
glVertex2f( 1, 1 );
glVertex2f( 1, 0 );
glVertex2f( 0, 0 );
glEnd();
```

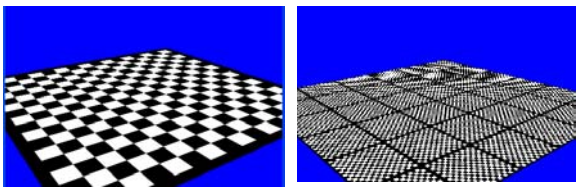
Mapping

```
glEnable(GL_TEXTURE_2D);
glBindTexture(myTexture);
 glBegin( GL_TRIANGLES);
 glTexCoord2f( 1, 1 );
 glVertex2f( 1, 1 );
 glTexCoord2f( 1, 0 );
 glVertex2f( 1, 0 );
 glTexCoord2f( 0, 0 );
 glVertex2f( 0, 0 );
 glEnd();
glDisable(GL_TEXTURE_2D);
```

Variations

- Repeat, Mirror, Clamp, Border
- Decal, Replace, Modulate

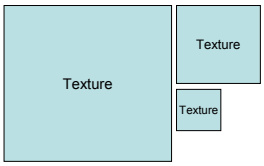
Minification



Minification (cont.)

Mipmapping

- Downsample the image ahead of time
- Sample from different versions as appropriate



Transparency & Translucency

Alpha Testing

Simple boolean test:

- Transparent
- Not Transparent

(Mega Man)

Transparency & Translucency (cont.)

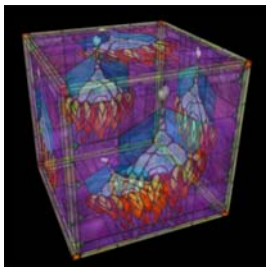
Blending

Normally:

```
glTexImage2D( GL_SRC_ALPHA,
   GL_ONE_MINUS_SRC_ALPHA )
```

Different options:

```
GL_ONE_MINUS_SRC_ALPHA
GL_ONE_MINUS_DST_ALPHA
GL_SRC_ALPHA_SATURATE
GL_SRC_ALPHA
GL_DST_ALPHA
GL_ZERO
GL_ONE
GL_SRC_COLOR
GL_DST_COLOR
GL_ONE_MINUS_SRC_COLOR
GL_ONE_MINUS_DST_COLOR
```



(http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=08)

Bump & Normal Mapping

Goal: More accurately simulate additional model detail. (Replace polygons)

Regular Model

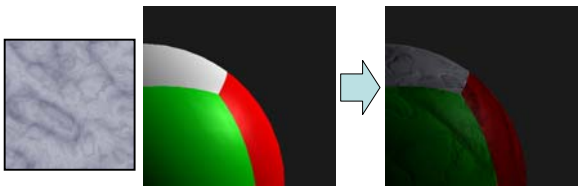
Normal-Mapped Model

(http://www.ionization.net/tutsnorm4.htm)

Bump & Normal Mapping (cont.)

Bump-Mapping

Use a heightmap to represent deviations from the planar polygon surface

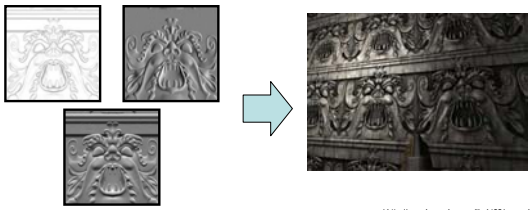


(http://www.ati.com/developer/sdk/rage128sdk/OpenGL/Samples/Rage128Bump.html)

Bump & Normal Mapping (cont.)

Normal-Mapping

Use a 3-channel texture to store the actual per-texel normal vectors



(http://members.shaw.ca/jimht03/normal.html)

Gloss & Light Mapping

Goal: More accurately simulate complex lighting effects

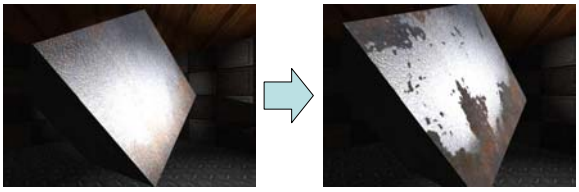
Light-Mapping
(Quake)

Gloss-Mapping
(Half-Life 2 Video)

Gloss & Light Mapping (cont.)

Gloss-Mapping

Use a greyscale image to store specular properties (“shininess”)



(http://www.ronfrazier.net/apparition/research/advanced_per_pixel_lighting.html)

Gloss & Light Mapping (cont.)

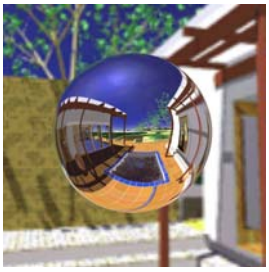
Light-Mapping

Pre-calculate complex lighting models and “bake” them into textures

(http://www.flipcode.com/articles/article_lightmapping.shtml)

Environment Mapping

Goal: Making things even more shiny



(http://www.nvidia.co.uk/page/demo_archive.html)

Misc. techniques

Billboarding

(Microsoft Billboard Demo – DirectX SDK)

Particle Systems

(http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=19)

Procedural Techniques

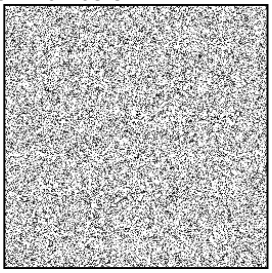
Goal: Get the computer to do the hard work of making the textures.

(http://www.csee.umbc.edu/~olano/s2001c24/ch02.html)

Procedural Techniques (cont.)

Simple Noise

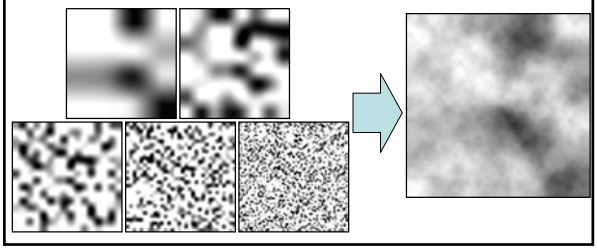
Just random numbers.



Procedural Techniques (cont.)

Perlin Noise

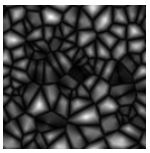
Still random numbers,
but more clever.



Procedural Techniques (cont.)

Cellular Texturing

Voronoi Diagrams
etc.



(http://www.blackpawn.com/texts/cellular/)

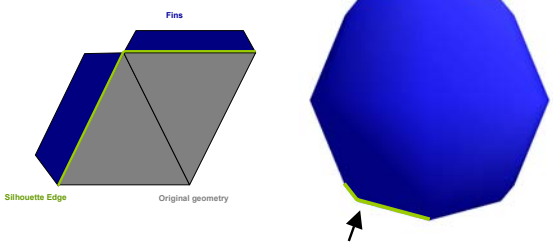
Reaction-Diffusion

Partial differential
equations

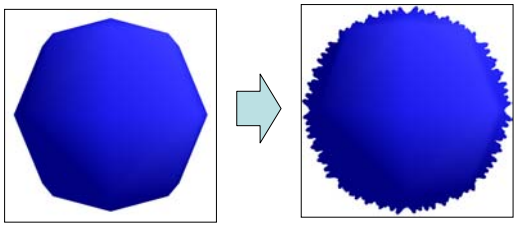
(http://www.gvu.gatech.edu/people/faculty/greg.turk/reaction_diffusion/reaction_diffusion.html)

Finning

When texturing just isn't enough.



Finning (cont.)



And now for some billboarding