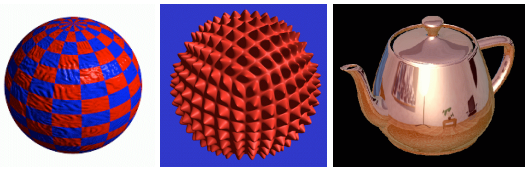
Procedural Modeling

Last Time?

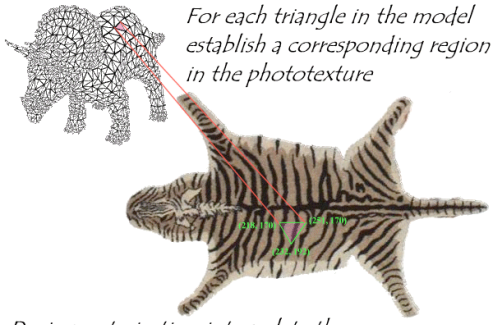
- Modern Graphics Hardware
- Cg Programming Language
- Gouraud Shading vs. Phong Normal Interpolation
- Bump, Displacement, & Environment Mapping



Today

- Texture Mapping
- Common Texture Coordinate Mappings
- Solid Texture
- Procedural Textures
- Perlin Noise
- Procedural Modeling
- L-Systems

Texture Mapping

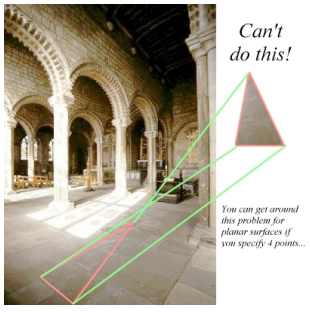


For each triangle in the model establish a corresponding region in the phototexture

During rasterization interpolate the coordinate indices into the texture map

Texture Mapping Difficulties

- Tedious to specify texture coordinates
- Acquiring textures is surprisingly difficult
  - Photographs have projective distortions
  - Variations in reflectance and illumination
  - Tiling problems

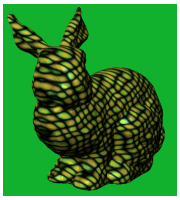


Can't do this!

You can get around this problem for planar surfaces if you specify 4 points...

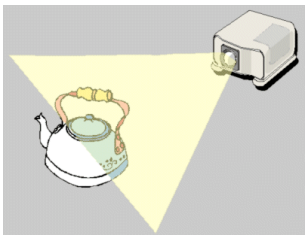
Common Texture Coordinate Mappings

- Orthogonal
- Cylindrical
- Spherical
- Perspective Projection
- Texture Chart



Projective Textures

- Use the texture like a slide projector
- No need to specify texture coordinates explicitly



Projective Texture Example

- Modeling from photographs
- Using input photos as textures

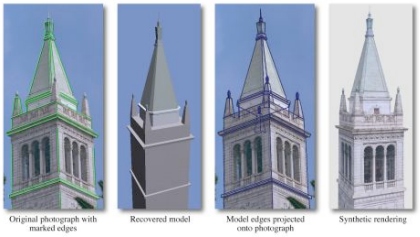
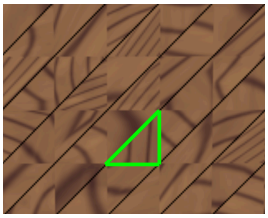
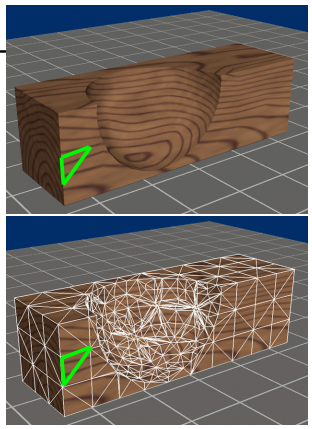


Figure from Debevec, Taylor & Malik
http://www.debevec.org/Research

Texture Chart

- Pack triangles into a single image



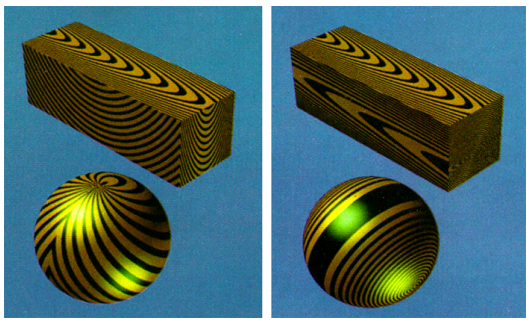


Questions?

Today

- Texture Mapping
- Common Texture Coordinate Mappings
- Solid Texture
- Procedural Textures
- Perlin Noise
- Procedural Modeling
- L-Systems

Texture Map vs. Solid Texture



"Solid Texturing of Complex Surfaces",
Peachey, SIGGRAPH 1985

Procedural Textures

f (x,y,z) → color

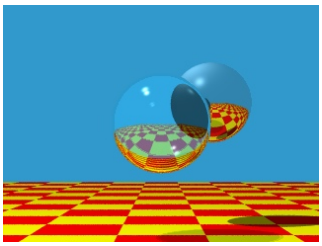
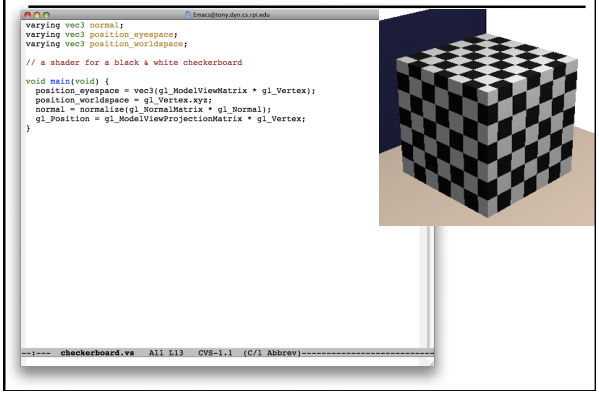


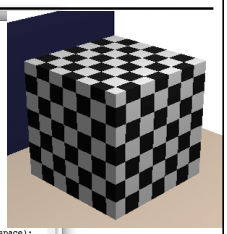
Image by Turner Whitted

GLSL example: checkerboard.vs



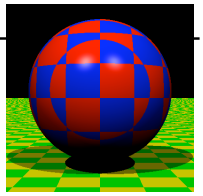
```
varying vec3 normal;
varying vec3 position_eyespace;
varying vec3 position_worldspace;

// a shader for a black & white checkerboard

void main(void) {
  position_eyespace = vec3(gl_ModelViewMatrix * gl_Vertex);
  position_worldspace = gl_Vertex.xyz;
  normal = normalize(gl_NormalMatrix * gl_Normal);
  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

GLSL example: checkerboard.fs

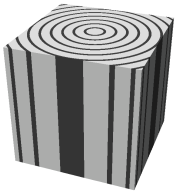


```
varying vec3 normal;
varying vec3 position_eyespace;
varying vec3 position_worldspace;

// a shader for a black & white checkerboard

void main (void) {

  vec3 color;

  // determine the parity of this point in the 3D checkerboard
  int count = 0;
  if (mod(position_worldspace.x,0.3)> 0.15) count++;
  if (mod(position_worldspace.y,0.3)> 0.15) count++;
  if (mod(position_worldspace.z,0.3)> 0.15) count++;
  if (count == 1 || count == 3) {
    color = vec3(0.1,0.1,0.1);
  } else {
    color = vec3(1,1,1);
  }

  // direction to the light
  vec3 light = normalize(gl_LightSource[1].position.xyz - position_eyespace);

  // basic diffuse
  float ambient = 0.3;
  float diffuse = 0.7*max(dot(normal,light),0.0);
  color = ambient*color + diffuse*color;
  gl_FragColor = vec4 (color, 1.0);
}
```

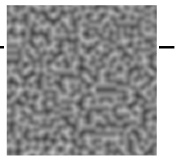
Procedural Textures

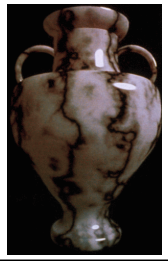


- Advantages:
  - easy to implement in ray tracer
  - more compact than texture maps (especially for solid textures)
  - infinite resolution
- Disadvantages
  - non-intuitive
  - difficult to match existing texture



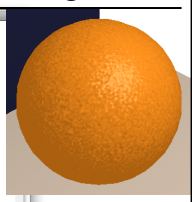
Readings for Today:





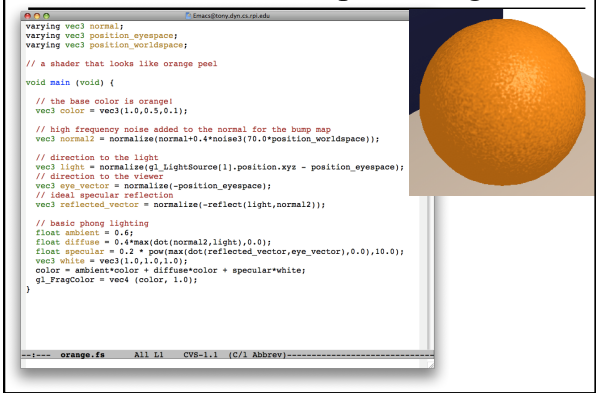
Ken Perlin,
"An Image Synthesizer", SIGGRAPH 1985
"Improving Noise", SIGGRAPH 2002

Another GLSL example: orange.vs



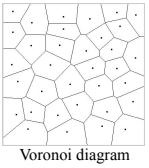
```
varying vec3 normal;
varying vec3 position_eyespace;
varying vec3 position_worldspace;

// a shader that looks like orange peel

void main(void) {

  // the fragment shader requires both the world space position (for
  // consistent bump mapping) & eyespace position (for the phong
  // specular highlight)
  position_eyespace = vec3(gl_ModelViewMatrix * gl_Vertex);
  position_worldspace = gl_Vertex.xyz;

  // pass along the normal
  normal = normalize(gl_NormalMatrix * gl_Normal);

  gl_Position = gl_ModelViewProjectionMatrix * gl_Vertex;
}
```

Another GLSL example: orange.fs

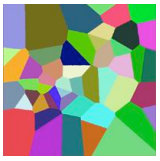


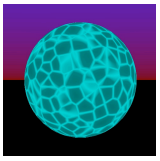
```
varying vec3 normal;
varying vec3 position_eyespace;
varying vec3 position_worldspace;

// a shader that looks like orange peel

void main (void) {

  // the base color is orange!
  vec3 color = vec3(1.0,0.5,0.1);

  // high frequency noise added to the normal for the bump map
  vec3 normal2 = normalize(normal+0.4*noise3(70.0*position_worldspace));

  // direction to the light
  vec3 light = normalize(gl_LightSource[1].position.xyz - position_eyespace);
  // direction to the viewer
  vec3 eye_vector = normalize(-position_eyespace);
  // ideal specular reflection
  vec3 reflected_vector = normalize(-reflect(light,normal2));

  // basic phong lighting
  float ambient = 0.6;
  float diffuse = 0.4*max(dot(normal2,light),0.0);
  float specular = 0.2 * pow(max(dot(reflected_vector,eye_vector),0.0),10.0);
  vec3 white = vec3(1.0,1.0,1.0);
  color = ambient*color + diffuse*color + specular*white;
  gl_FragColor = vec4 (color, 1.0);
}
```

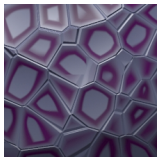
Cellular Textures



Voronoi diagram

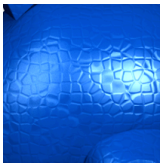






"A Cellular Texture Basis Function", Worley, SIGGRAPH 1996

www.worley.com



Questions?

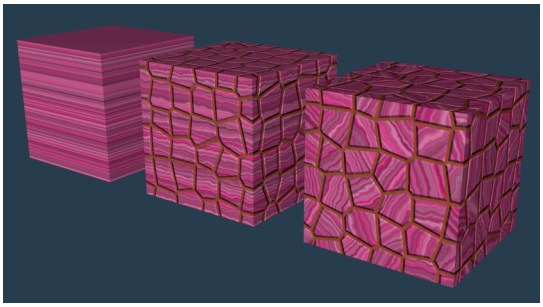
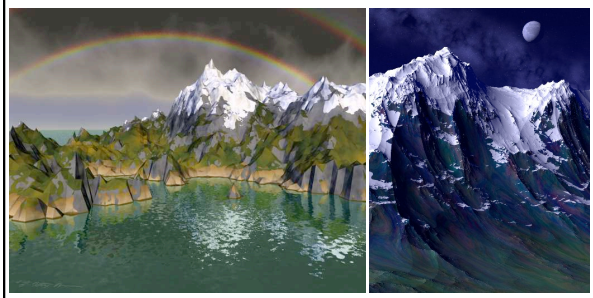


Image by Justin Legakis

Today

- Texture Mapping
- Common Texture Coordinate Mappings
- Solid Texture
- Procedural Textures
- Perlin Noise
- Procedural Modeling
- L-Systems

Procedural Displacement Mapping



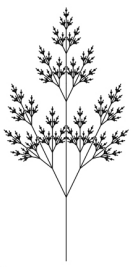
Ken Musgrave
www.kenmusgrave.com

L-Systems

alphabet: {a,b}
initiator: a
production rules:
  a -> b
  b -> ba

generations:
  a
  b
  ba
  bab
  babba
  babbabab
  babbababbabba
  babbababbabbababbabab

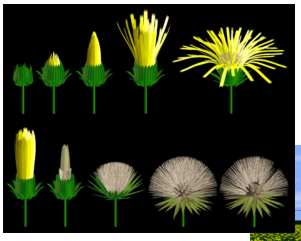
d
n=7, δ=20°
X
X → F[+X]F[-X]+X
F → FF



e
n=7, δ=25.7°
X
X → F[+X][-X]FX
F → FF

Prusinkiewicz & Lindenmayer,
The Algorithmic Beauty of Plants, 1990
http://algorithmicbotany.org/

L-Systems



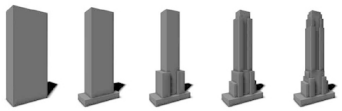
Animation of Plant Development
Prusinkiewicz et al., SIGGRAPH 1993

Prusinkiewicz & Lindenmayer, *The Algorithmic Beauty of Plants,* 1990
http://algorithmicbotany.org/

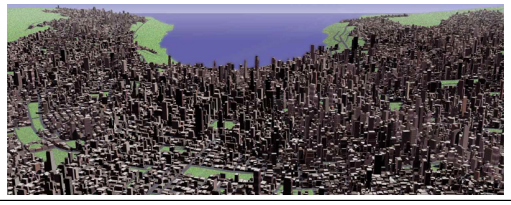
Reading for Today:

- "Procedural Modeling of Buildings", Mueller, Wonka, Haegler, Ulmer & Van Gool, SIGGRAPH 2006

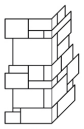
L-Systems for Cities

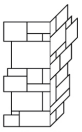


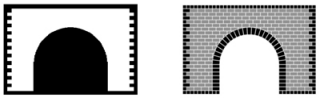
"Procedural Modeling of Cities", Parish & Müller, SIGGRAPH 2001



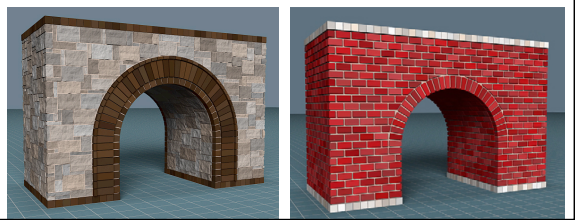
Cellular Texturing for Architecture







"Feature-Based Cellular Texturing for Architectural Models", Legakis, Dorsey, & Gortler, SIGGRAPH 2001



Questions?

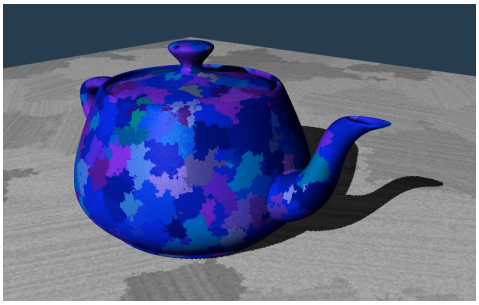


Image by Justin Legakis

Reading for Tuesday:

- "Artistic Thresholding" Xu & Kaplan, NPAR 2008