Texture Mapping
There are limits to geometric modeling

Although modern GPUs can render millions of triangles/sec, that’s not enough sometimes...

http://www.beinteriordecorator.com

National Geographic
Use texture mapping to increase realism through detail

This image is just 8 polygons!
[Angel and Shreiner]

No texture

With texture
Store 2D images in buffers and lookup pixel reflectances.

![Diagram of a computer graphics pipeline](image)

$f(s, t)$

**procedural**

**photo**
3D solid textures
Other uses of textures...

Light maps
Shadow maps
Environment maps
Bump maps
Opacity maps
Animation

[Angel and Shreiner]
[Stam 99]
Texture mapping in the OpenGL pipeline

- Geometry and pixels have separate paths through pipeline
- meet in **fragment processing** - where textures are applied
- texture mapping applied at end of pipeline - efficient since relatively few polygons get past clipper
uv Mapping

- 2D texture is parameterized by \((u, v)\)
- Assign polygon vertices texture coordinates
- Interpolate within polygon
Texture Calibration
The major issues in texture mapping...

- What should the actual mapping be?

  easy: rectangular surface
  harder: parametric surface
Given a point on the object \((x,y,z)\), what point \((u,v)\) in the texture we use?
Example: planar mapping
Intermediate surfaces

First map the texture to a simpler, intermediate surface
Cylindrical mapping

\[(x,y,z) \rightarrow (\theta, h) \rightarrow (u,v)\]
Spherical Mapping

\((x, y, z) \rightarrow (\text{latitude}, \text{longitude})\)

\(\rightarrow (u, v)\)
Box Mapping
How do we map between intermediate and actual objects?
How do we map between intermediate and actual objects?

What intermediate shape was used here?
Cylindrical

Spherical
Parametric Surfaces

32 parametric patches
3D solid textures
can map object \((x,y,z)\) directly to texture \((u,v,w)\)
Procedural textures

e.g., Perlin noise

Rosalee Wolfe
Triangles
Texturing triangles

- Store \((u,v)\) at each vertex
- Interpolate inside triangles using barycentric coordinates
Texturing triangles

- Store (u,v) at each vertex
- Interpolate inside triangles using barycentric coordinates

\[
p(\beta, \gamma) = a + \beta(b - a) + \gamma(c - a),
\]

\[
u(\beta, \gamma) = u_a + \beta(u_b - u_a) + \gamma(u_c - u_a),
\]

\[
v(\beta, \gamma) = v_a + \beta(v_b - v_a) + \gamma(v_c - v_a).
\]
Texturing triangles

Choice of \((u,v)\) makes big difference
Texturing triangles

Choice of (u,v) makes big difference
Texturing triangles

Choice of \((u,v)\) makes big difference
Textures in OpenGL

• Assign \((u,v)\) to vertices

• OpenGL then uses interpolation for triangle interior

good selection of tex coordinates

poor selection of tex coordinates

texture stretched over trapezoid showing effects of bilinear interpolation
Multitexturing

Fragment → Texture unit 0 → Texture unit 1 → Texture unit 2 → Frame buffer
Texture Sampling
Texture Mapping

Texture coordinates

Object coordinates

Window coordinates

Texels

Pixels

[Angel and Shreiner]
Point Sampling

Map back to texture image and use the nearest texel
Aliasing

**Point sampling** of the texture can lead to aliasing artifacts

Point samples in (or x,y,z) space

miss blue stripes

point samples in texture space

[Angel and Shreiner]
Magnification and Minification
Magnification and Minification

More than one texel can cover a pixel (minification) or more than one pixel can cover a texel (magnification)

Can use point sampling (nearest texel) or linear filtering (2 x 2 filter) to obtain texture values.
We apply filtering to reduce aliasing artifacts
A better but slower option is to use area averaging.
Use bilinear filtering

\[ p = ? \]

- nearest neighbor
- bilinear
- bicubic

mitigate magnification artifacts
Mipmapming

Reduce minification artifacts
Prefilter the texture to obtain reduced resolutions
Requires 1/3 more space
Get a texture hierarchy indexed by level

128×128, 64×64, 32×32, 16×16, 8×8, 4×4, 2×2, 1×1

Togikun, Wikimedia Commons
point sampling

mipmapped point sampling

linear filtering

mipmapped linear filtering

[Angel and Shreiner]