Texture Mapping

- Important topic: nearly all objects textured
  - Wood grain, faces, bricks and so on
  - Adds visual detail to scenes
- Meant as a fun and practically useful lecture

Adding Visual Detail

- Basic idea: use images instead of more polygons to represent fine scale color variation

Parameterization

- $\text{geometry} + \text{image} = \text{texture map}$
- Q: How do we decide where on the geometry each color from the image should go?

Option: Varieties of projections

- [Paul Bourke]
Option: unfold the surface

Option: make an atlas

Option: it's the artist's problem

CAPE Evaluations
- Fill out now, can be done on phone
- Enthusiasm important to future offerings (new to offer in winter this year, many enrollments in 167)
- Comments useful to future years
- Some key innovations: modern OpenGL, GLSL; feedback servers (including code), UCSD Online, …
- Separately, please also evaluate the TAs

Outline
- Types of projections
- Interpolating texture coordinates
- Broader use of textures

How to map object to texture?
- To each vertex (x,y,z in object coordinates), must associate 2D texture coordinates (s,t)
- So texture fits “nicely” over object
**Idea: Use Map Shape**
- Map shapes correspond to various projections
  - Planar, Cylindrical, Spherical
- First, map (square) texture to basic map shape
- Then, map basic map shape to object
  - Or vice versa: Object to map shape, map shape to square
- Usually, this is straightforward
  - Maps from square to cylinder, plane, sphere well defined
  - Maps from object to these are simply spherical, cylindrical, cartesian coordinate systems

**Planar mapping**
- Like projections, drop z coord $(s,t) = (x,y)$
- Problems: what happens near $z = 0$?

**Cylindrical Mapping**
- Cylinder: $r, \theta, z$ with $(s,t) = (\theta/(2\pi), z)$
  - Note seams when wrapping around ($\theta = 0$ or $2\pi$)

**Spherical Mapping**
- Convert to spherical coordinates: use latitude/long.
  - Singularities at north and south poles

**Cube Mapping**

**Cube Mapping**
Outline

- Types of projections
- Interpolating texture coordinates
- Broader use of textures

Artifacts

- What artifacts do you see?
- Why?
- Why not in standard Gouraud shading?
- Hint: problem is in interpolating parameters

Interpolating Parameters

- The problem turns out to be fundamental to interpolating parameters in screen-space
  - Uniform steps in screen space ≠ uniform steps in world space

Texture Mapping

- Linear interpolation of texture coordinates
- Correct interpolation with perspective divide

Interpolating Parameters

- Perspective foreshortening is not getting applied to our interpolated parameters
  - Parameters should be compressed with distance
  - Linearly interpolating them in screen-space doesn’t do this
Perspective-Correct Interpolation

- Skipping a bit of math to make a long story short...
  - Rather than interpolating \( u \) and \( v \) directly, interpolate \( u/z \) and \( v/z \)
    - These do interpolate correctly in screen space
    - Also need to interpolate \( z \) and multiply per-pixel
  - Problem: we don’t know \( z \) anymore
  - Solution: we do know \( w \approx 1/z \)
  - So, interpolate \( uw \) and \( vw \) and \( w \), and compute
    - \( u = uw/w \) and \( v = vw/w \) for each pixel
    - This unfortunately involves a divide per-pixel
- Wikipedia page

Texture Map Filtering

- Naive texture mapping aliases badly
- Look familiar?
  - \( u_{val} = \text{int}(u \cdot \text{denom} + 0.5) \)
  - \( v_{val} = \text{int}(v \cdot \text{denom} + 0.5) \)
  - \( \text{pix} = \text{texture.getPixel}(u_{val}, v_{val}) \)
- Actually, each pixel maps to a region in texture
  - \( |\text{PIX}| < |\text{TEX}| \)
    - Easy: interpolate (bi linear) between texel values
  - \( |\text{PIX}| > |\text{TEX}| \)
    - Hard: average the contribution from multiple texels
  - \( |\text{PIX}| \approx |\text{TEX}| \)
    - Still need interpolation!

Mip Maps

- Keep textures prefiltered at multiple resolutions
  - For each pixel, linearly interpolate between two closest levels (e.g., trilinear filtering)
  - Fast, easy for hardware
- Why “Mip” maps?

MIP-map Example

- No filtering:
  - AAAAAAGH
  - MY EYES ARE BURNING
  - Where are my glasses?
- MIP-map texturing:

Outline

- Types of projections
- Interpolating texture coordinates
- Broader use of textures

Texture Mapping Applications

- Modulation, light maps
- Bump mapping
- Displacement mapping
- Illumination or Environment Mapping
- Procedural texturing
- And many more
**Modulation textures**
Map texture values to scale factor

\[
I = T(N \cdot L) + K_a \cdot I_a + \sum (K_n (N \cdot L) + K_d (V \cdot R)^n) S_n + K_r R_n + K_f F_n
\]

---

**Bump Mapping**

- Texture = change in surface normal!

![Images of bump mapping](image1)

---

**Displacement Mapping**

---

**Illumination Maps**
- Quake introduced *illumination maps or light maps* to capture lighting effects in video games

Texture map:

![Texture map](image2)

Texture map + light map:

---

**Environment Maps**

Images from Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments
Gene Miller and C. Robert Hoffman
SIGGRAPH 1984 "Advanced Computer Graphics Animation" Course Notes

---

**Solid textures**
Texture values indexed by 3D location (x,y,z)
- Expensive storage, or
- Compute on the fly, e.g. Perlin noise
Procedural Texture Gallery