This Lecture: 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
  - But not tested specifically on it

Adding Visual Detail

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

Parameterization

- Q: How do we decide where on the geometry each color from the image should go?

Option: Varieties of projections

To Do

- Work on HW4 milestone
- Prepare for final push on HW 4
- No final exam. HW 4, written ass 1, 2
Option: unfold the surface

Option: make an atlas

Option: it’s the artist’s problem

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 \((0 = 0\ or \ 2\pi)\)

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

Cube Mapping
- Outline
  - Types of projections
  - Interpolating texture coordinates
  - Broader use of textures
**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

**Artifacts**

- McMillan’s demo of this is at [http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture21/Slide05.html](http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture21/Slide05.html)
- Another example [http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture21/Slide06.html](http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture21/Slide06.html)
- What artifacts do you see?
- Why?
- Why not in standard Gouraud shading?
- Hint: problem is in interpolating parameters

**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 ∝ 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

Texture Map Filtering

- Naive texture mapping aliases badly

  \[
  \text{let } u_{\text{val}} = \lfloor u \cdot \text{denom} + 0.5 \rfloor; \\
  \text{let } v_{\text{val}} = \lfloor v \cdot \text{denom} + 0.5 \rfloor; \\
  \text{let } \text{pix} = \text{texture.getPixel}(u_{\text{val}}, v_{\text{val}}); \\
  \]

- Look familiar?

- Actually, each pixel maps to a region in texture
  - \(|\text{PIX}| < |\text{TEX}|\)
    - Easy: interpolate (bilinear) 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

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

MIP-map Example

- No filtering:

  AAAAAAAGH
  MY EYES ARE BURNING

- MIP-map texturing:

  Where are my glasses?

Modulation textures

Map texture values to scale factor

Wood texture

Texture value

\[ l = T(x, y)I_s + K_r I_s + \sum \{K_d(N \cdot L_s) + K_a(V \cdot R_s)S_sJ_s + K_r r_i + K_f f_i \} \]
Bump Mapping

- Texture = change in surface normal!

Displacement Mapping

Illumination Maps

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

Environment Maps

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

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