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

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

Option: Varieties of projections
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), edX edge, …
- 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
Outline

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

1st idea: Gouraud interp. of texcoords

\[ I_a = I_1 \frac{(y_s - y_2)}{y_1 - y_2} + I_2 \frac{(y_1 - y_s)}{y_1 - y_2} \]

\[ I_a = I_1 \frac{(y_s - y_3)}{y_1 - y_3} + I_3 \frac{(y_1 - y_s)}{y_1 - y_3} \]

\[ I_a = I_a \frac{(x_b - x_p)}{x_b - x_a} + I_b \frac{(x_p - x_a)}{x_b - x_a} \]

Actual implementation efficient: difference equations while scan converting

Artifacts

- Wikipedia page
- 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

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
    \[ u = uw/w \quad \text{and} \quad v = vw/w \] for each pixel
  - This unfortunately involves a divide per pixel
- Wikipedia page

Texture Map Filtering

- Naive texture mapping aliases badly
- Look familiar?
  \[
  \begin{align*}
  u_{val} &= \lfloor u \times \text{denom} + 0.5 \rfloor \\
  v_{val} &= \lfloor v \times \text{denom} + 0.5 \rfloor \\
  \text{pix} &= \text{texture.getPixel}(u_{val}, v_{val})
  \end{align*}
  \]
- 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
- MIP-map texturing:
  - Where are my glasses?

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

\[ I = T(c_0 + K_n L_n + K_a F_a + \sum K_i (N \cdot L_i) + K_s (F \cdot R)^s S_1 f_s + K_r R_f + K_f F_f) \]

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

Texture map:

Texture map + light map:

Solid textures

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

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
Procedural Texture Gallery