Texture Mapping

Instructor – Stephen J. Guy
Review - Shading

- What type of Shading are these?

- How do you compute the normal in Phong Shading?
Review - Shading

- Which is Which?
Review - Shading

- Why use Phong?
- Why use Flat Shading?
- Why use Gouraud Shading?
Review Fractals

- What Defines a Fractal

- Where do we find them?
Overview

- Texture Mapping Method
  - Parameterization
  - Mapping
  - Filtering

- Texture Mapping Applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
  - Volume texture
  - Non-photorealistic Rendering
Textures

- Describe color variation in interior of 3D polygon
  - When scan converting a polygon, vary pixel colors according to values fetched from a texture
3D Rendering Pipeline

- 3D Primitives
- Modeling Transformation
- Lighting
- Viewing Transformation
- Projection Transformation
- Clipping
- Viewport Transformation
- Scan Conversion
- 3D Rendering

Texture Mapping
Surface Texture

- Add visual detail to surface of 3D objects
Surface Texture - Comparison

- Add visual detail to surface of 3D objects

Polygonal Model

With Surface Textures
Surface Texture

- Add visual detail to surface of 3D objects

[Daren Horley]
Q: How do we decide *where* the color from the image should go on the geometry?
Options – Variety of Projections

Option – Unfold the Surface

[Images of 3D models and diagrams]

[Piponi 2000]
Option – Make an Atlas

charts  atlas  surface

[Sander 2001]
Overview

- Texture Mapping Method
  - Parameterization
  - Mapping
  - Filtering

- Texture Mapping Applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
  - Volume texture
  - Non-photorealistic Rendering
Texture Mapping

- Define texture
- Specify mapping from texture to surface
- Lookup texture values during scan conversion

![Diagram showing texture mapping process]
Texture Mapping – Scan Convert

- Scan Conversion maps from:
  - Image Coordinate System \((x,y)\) to...
  - Modeling Coordinate System \((u,v)\) to...
  - Texture Coordinate System \((s,t)\)
Texture Mapping

- Texture Mapping is a 2D projective transformation
  - Texture coordinate system: \((t, s)\) to
  - Image coordinate system: \((x, y)\)
Naïve Texture Mapping

- A first cut at a texture-mapping rasterizer:
  - For each pixel:
    - Interpolate \( u \) & \( v \) down edges and across spans
    - Look up nearest texel in texture map
    - Color pixel according to texel color (possibly modulated by lighting calculations)

- McMillan’s demo of this is at:
  [http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture 21/Slide05.html](http://graphics.lcs.mit.edu/classes/6.837/F98/Lecture 21/Slide05.html)

- What artifacts to do you see?
Naïve Texturing Artifacts

- Warping at edges of triangles

- A more obvious example:
  http://groups.csail.mit.edu/graphics/classes/6.837/F98/Lecture21/Slide06.html

- What’s going on here?
  - Consider the geometry of interpolating parameters more carefully
Interpolating Parameters

- There is a fundamental problem with interpolating parameters in screen space
  - *Uniform steps in screen space ≠ Uniform steps in world space*
Texture Mapping - Comparison

[Linear interpolation of texture coordinates]

[Correct interpolation of texture coordinates]

[Hill Figure 8.42]
Interpolating Parameters

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

- WAIT! Haven’t we seen this before?
  - Isn’t this is a problem with Gouraud Shading?

- A: Yes! But we usually don’t notice.

http://groups.csail.mit.edu/graphics/classes/6.837/F98/Lecture21/Slide17.html
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 \( w \propto 1/z \)
- So…interpolate \( uw \) and \( vw \) and \( w \), and compute
  - \( u = uw/w \) and \( v = vw/w \) for each pixel
    - Unfortunately this involves an extra divide per pixel

Overview

- Texture Mapping Method
  - Parameterization
  - Mapping
  - Filtering
- Texture Mapping Applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
  - Volume texture
  - Non-photorealistic Rendering
Texture Filtering

- Must sample texture to determine color at each pixel in image

[Angel Figure 9.4]
Texture Map Aliasing

- Naive texture mapping aliases badly

- Look familiar?

  ```
  int uval = (int) (u * denom + 0.5f);
  int vval = (int) (v * denom + 0.5f);
  int pix = texture.getPixel(uval, vval);
  ```

- Actually, each pixel maps to a region in texture
  - |PIX| < |TEX|
    - Easy: interpolate (bilinear) between texel values
  - |PIX| > |TEX|
    - Hard: average the contribution from multiple texels
  - |PIX| ~ |TEX|
    - Still need interpolation!
Texture Filtering

- Size of filter depends on projective warp
  - Can prefilter images
    - Mip maps
    - Summed area tables

Magnification

Minification
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?
Why “Mip” maps?

- Mip maps fit well into square textures
MIP-map example

- No filtering:

- MIP-map texturing:
Summed-area tables

- At each texel keep sum of all values down & right
  - To compute sum of all values within a rectangle, simply subtract two entries
  - Better ability to capture very oblique projections
  - But, cannot store values in a single byte
Summed-area tables

- Mipmaps assume that each pixel projects to a square in the texture (a lie!)
- SAT can integrate texels covered by the pixel more exactly (but still quickly)
- Example:

MIP-map texturing

Summed-Area Table Texturing
Overview

- Texture Mapping Method
  - Parameterization
  - Mapping
  - Filtering

- Texture Mapping Applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
  - Volume texture
  - Non-photorealistic Rendering
Modulation Textures

- Map texture values to scale factor

\[ I = T(s,t)(I_E + K_A I_A + \sum_L (K_D (N \cdot L) + K_S (V \cdot R)^n) S_L I_L + K_T I_T + K_S I_S) \]
Texture Mapping Variations

- A texture can modulate any parameter in the Phong lighting equation

Texture as $R,G,B$:

Texture as diffuse lighting coefficients:
Bump Mapping

- Texture = change in surface normal!

Sphere w/ diffuse texture  Swirly bump map  Sphere w/ diffuse texture and swirly bump map
More Bump Mapping

- How can you tell a bump-mapped object from actual geometry?
Displacement Mapping
Illumination Mapping

- Quake introduced *illumination maps* or *light maps* to capture lighting effects in video games.
Environment Mapping

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 Texturing

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

- Look up value into 1D Texture: