Texture Mapping & Hidden Surface Removal

Thomas Funkhouser
Princeton University
COS 426, Spring 2004

Texture Mapping Overview

- Texture mapping methods
  - Parameterization
  - Mapping
  - Filtering

- Texture mapping applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
  - Non-photorealistic rendering

Surface Textures

- Add visual detail to surfaces of 3D objects

3D Rendering Pipeline (for direct illumination)

Texture Mapping

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

Surface Textures

- Add visual detail to surfaces of 3D objects

With surface texture

Polygonal model

[Daren Horley]
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

Texture Mapping Overview

- Texture mapping methods
  - Parameterization
  - Mapping
  - Filtering

- Texture mapping applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
  - Volume textures
  - Non-photorealistic rendering

Option: unfold the surface

Option: make an atlas

Texture Mapping

- Steps:
  - Define texture
  - Specify mapping from texture to surface
  - Lookup texture values during scan conversion
Texture Mapping

- When scan convert, map from ... 
  - image coordinate system \((x,y)\) to 
  - modeling coordinate system \((u,v)\) to 
  - texture image \((t,s)\)

Texture Mapping

- Texture mapping is a 2D projective transformation 
  - texture coordinate system \((t,s)\) to 
  - image coordinate system \((x,y)\)

Texture Mapping

- Scan conversion 
  - Interpolate texture coordinates down/across scan lines 
  - Distortion due to bilinear interpolation approximation 
    - Cut polygons into smaller ones, or 
    - Perspective divide at each pixel

Overview

- Texture mapping methods 
  - Parameterization 
  - Mapping 
  - Filtering

- Texture mapping applications 
  - Modulation textures 
  - Illumination mapping 
  - Bump mapping 
  - Environment mapping 
  - Image-based rendering 
  - Non-photorealistic rendering

Texture Filtering

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

- Aliasing is a problem

Point sampling

Area filtering

Angel Figure 9.5

**Texture Filtering**

- Ideally, use elliptically shaped convolution filters

In practice, use rectangles

**Texture Filtering**

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

Magnification

Minification

Angel Figure 9.14

**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

**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

**Overview**

- Texture mapping methods
  - Parameterization
  - Mapping
  - Filtering

- Texture mapping applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
Modulation textures
Map texture values to scale factor

\[ I = T(s,t)I_L + K_sI_L + \sum (K_d(N \cdot L) + K_a(V \cdot N)^s)S_i I_L + K_tI_L + K_rI_L + K_nI_L \]

Illumination Mapping
Map texture values to surface material parameter
- \( K_s \)
- \( K_d \)
- \( K_a \)
- \( K_t \)
- \( K_r \)
- \( n \)

\[ I = T(s,t)I_L + K_sI_L + \sum (K_d(N \cdot L) + K_a(V \cdot N)^s)S_i I_L + K_tI_L + K_rI_L + K_nI_L \]

Bump Mapping
Texture values perturb surface normals

Environment Mapping
Texture values are reflected off surface patch

Image-Based Rendering
Map photographic textures to provide details for coarsely detailed polygonal model
Solid textures

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

Hidden Surface Removal (HSR)

- Surfaces may be back-facing.
- Surfaces may be occluded.
- Surfaces may overlap in the image plane.
- Surfaces may intersect.

Texture Mapping Summary

- Texture mapping methods
  - Parameterization
  - Mapping
  - Filtering
- Texture mapping applications
  - Modulation textures
  - Illumination mapping
  - Bump mapping
  - Environment mapping
  - Image-based rendering
  - Volume textures

3D Rendering Pipeline

Somewhere in here we have to decide which objects are visible, and which objects are hidden.

HSR Algorithms

[Hidden Surface Removal Algorithm diagram]

Hidden Surface Removal Algorithms

- Object space
  - Back-face detection
  - Depth sort
- Screen space
  - Ray casting
  - Scan-line
  - Z-buffer
  - Area subdivision
**Back-face detection**

Q: How do we test for back-facing polygons?  
A: Dot product of the normal and view directions.

**3D Rendering Pipeline**

A polygon is back-facing if \( \mathbf{v} \cdot \mathbf{n} > 0 \)

**Depth sort**

“Painter’s algorithm”
- Sort surfaces in order of decreasing maximum depth
- Scan convert surfaces in back-to-front order

**BSP Tree**

- Binary space partition with solid cells labeled
  - Constructed from polygonal representations
  - Provides linear-time depth sort for arbitrary view

**Hidden Surface Removal Algorithm**

- Object space
  - Back-face detection
  - Depth sort
- Screen space
  - Ray casting
  - Scan-line
  - Z-buffer
  - Area subdivision

(We’ll come back to this…)

Naylor
Ray Casting

- Fire a ray for every pixel
  - If ray intersects multiple objects, take the closest

Ray Casting Pipeline

- Ray casting comments
  - O(p log n) for p pixels
  - May (or not) use pixel coherence
  - Simple, but generally not used

Z-Buffer

- Color & depth of closest object for every pixel
  - Update only pixels whose depth is closer than in buffer
  - Depths are interpolated from vertices, just like colors

Z-buffer comments
- Polygons rasterized in any order
- Requires lots of memory
  - 1K x 1K x 24bits
- Was expensive, cheap now
- Subject to aliasing (A-buffer)
- Commonly in hardware

Scan-Line Algorithm

- For each scan line, construct spans
  - Sort by depth

Scan-line comments
- Fully compute only visible pixels
- Coherence among along scans
- Commonly in software
Area Subdivision

Warnock’s algorithm
- Fill area if:
  - All surfaces are outside area, or
  - Only one surface intersects area, or
  - One surface occludes other surfaces in area
- Otherwise, subdivide

Summary
- Texture Mapping
  - Add detail during scan conversion
- Hidden surface removal
  - Find visible surfaces

3D Rendering Pipeline

Area subdivision comments
- Augments scan conversion
- Polygon coherence
- Commonly in software