CSE528 Computer Graphics: Theory, Algorithms, and Applications

Hong Qin
Department of Computer Science
State University of New York at Stony Brook (Stony Brook University)
Stony Brook, New York 11794--4400
Tel: (631)632-8450; Fax: (631)632-8334
qin@cs.sunysb.edu
http://www.cs.sunysb.edu/~qin
Texture Mapping for Computer Graphics
The Limitations of Geometric Modeling

- Although graphics cards can render over 10 million polygons per second, that number is insufficient for many phenomena
  - Clouds
  - Grass
  - Terrain
  - Skin
The Problem with Geometric Models

• We do not want to represent all of these details with geometry ONLY!!!
Texture Mapping: Basic Concept

- Increase the apparent complexity of simple geometry
- Like wallpapering or gift-wrapping with stretchy paper
- Curved surfaces require extra stretching or even cutting
Objectives and Topics

• **Introduction of mapping methods**
  – Texture mapping
  – Environment mapping
  – Bump mapping

• **Consider basic strategies**
  – Forward vs. backward mapping
  – Point sampling vs. area averaging
Modeling an Orange (A Classical Example)

- Consider the problem of modeling an orange (the fruit)
- Start with an orange-colored sphere
  - Too simple
- Replace sphere with a more complex shape
  - Does not capture surface characteristics (small dimples)
  - Takes too many polygons to model all the dimples
Modeling an Orange

• Take a picture of a real orange, scan it, and “paste” onto simple geometric model
  – This process is known as texture mapping

• Still might not be sufficient because resulting surface will be smooth
  – Need to change local shape
  – Bump mapping
Texture Mapping

- A clever way of adding surface details
- Two ways can achieve the goal:
  - Surface detail polygons: create more and more polygons to model object details
    - Add scene complexity and thus slow down the graphics rendering performance
    - Some fine features are hard to model!
  - Map a texture to the surface (a more popular approach)

Complexity of images does not affect the complexity of geometry processing (transformation, clipping...)

Department of Computer Science
Center for Visual Computing

CSE528 Lectures

ST·NY BR·K
STATE UNIVERSITY OF NEW YORK
Three Types of Mapping

- **Texture mapping**
  - Uses images to fill inside of polygons

- **Environment (reflection mapping)**
  - Uses a picture of the environment for texture maps
  - Allows simulation of highly specular surfaces

- **Bump mapping**
  - Emulates altering normal vectors during the rendering process
Texture Mapping

geometric model

Texture-mapped model
Environment Mapping
Environment Mapping Example
Bump Mapping
Where Does Mapping Take Place?

- Mapping techniques are implemented at the end of the rendering pipeline
  - Very efficient because few polygons make it past the clipper

![Diagram of rendering pipeline]

- Vertices → Geometric processing → Rasterization → Display
- Pixels → Pixel operations → Geometric processing
Is It Really Simple?

- Although the idea is simple - map an image to a surface - there are 3 or 4 coordinate systems involved.
Coordinate Systems

- **Parametric coordinates**
  - May be used to model curves and surfaces
- **Texture coordinates**
  - Used to identify points in the image to be mapped
- **Object or world coordinates**
  - Conceptually, where the mapping takes place
- **Window coordinates**
  - Where the final image is really produced
Texture Mapping

- Parametric coordinates
- Texture coordinates
- World coordinates
- Window coordinates
Mapping Functions

- Basic problem is how to find the maps
- Consider mapping from texture coordinates to a point of a surface
- Appear to need three functions
  \[ x = x(s,t) \]
  \[ y = y(s,t) \]
  \[ z = z(s,t) \]
- But we really want to go the other way

\[(x, y, z)\]
Backward Mapping

- We really want to go backwards
  - Given a pixel, we want to know to which point on an object it corresponds
  - Given a point on an object, we want to know to which point in the texture it corresponds

- Need a map of the form
  \[ s = s(x, y, z) \]
  \[ t = t(x, y, z) \]

- Such functions are difficult to find in general
Map Textures to Surfaces

• **Texture mapping is performed in rasterization (backward mapping)**

  ❑ For each pixel that is to be painted, its texture coordinates \((s, t)\) are determined (interpolated) based on the corners’ texture coordinates (why not just interpolate the color?)

  ❑ The interpolated texture coordinates are then used to perform texture lookup
Texture Mapping Pipeline

1. projection
2. texture lookup
3. patch texel

3D geometry → 2D projection of 3D geometry → 2D image

- 3D geometry
- 2D projection of 3D geometry
- 2D image
- t
- S
- 3D geometry
- 2D image
Texture Value Lookup

- For the given texture coordinates \((s, t)\), we can find a unique image value from the texture map.

How about coordinates that are not exactly at the intersection (pixel) positions?

A) Nearest neighbor
B) Linear Interpolation
C) Other filters
Texture Rasterization

- Texture coordinates are interpolated from polygon vertices just like ... remember line drawing ....
  - Color: Gouraud shading
  - Depth: Z-buffer
    - First along polygon edges between vertices
    - Then along scanlines between left and right sides

![Diagram of texture rasterization](image)
Texture Interpolation

• Specify a texture coordinate \((u,v)\) at each vertex
• Can we just linearly interpolate the values in screen space?
Interpolation - What Goes Wrong?

- **Linear interpolation in screen space:**

  - Texture source image
  - What we get
  - What we want
Linear Texture Coordinate Interpolation

- This doesn’t work in perspective projection!
- The textures look warped along the diagonal
- Noticeable during an animation
Why?

- Equal spacing in screen (pixel) space is **not** the same as in texture space in perspective projection
  - **Perspective foreshortening**
Visualizing the Problem

- Notice that uniform steps on the image plane do not correspond to uniform steps along the edge.
Perspective-Aware Texture Coordinate Interpolation

• Interpolate \((\text{tex\_coord}/w)\) over the polygon, then do perspective division after interpolation

• Compute at each vertex after perspective transformation
  – “Numerators” \(s/w, t/w\)
  – “Denominator” \(1/w\)

• Linearly interpolate \(1/w, s/w,\) and \(t/w\) across the polygon

• At each pixel
  - Perform perspective division of interpolated texture coordinates: \((s/w, t/w)\) by interpolated \(1/w\) (i.e., numerator over denominator) to get \((s, t)\)
Perspective-Correct Interpolation

- That fixed it!
Common Texture Coordinate Mappings

- Orthogonal
- Cylindrical
- Spherical
- Perspective Projection
- Texture Chart
Map Textures to Surfaces

- The key question: Establish mapping from texture to surfaces (polygons):
  - Application program needs to specify texture coordinates for each corner of the polygon

The polygon can be in an arbitrary size

(0,0)  (1,0)  (1,1)
Texture Mapping Difficulties

- Tedious to specify texture coordinates
- Acquiring textures is surprisingly difficult
  - Photographs have projective distortions
  - Variations in reflectance and illumination
  - Tiling problems
Projector Functions

- How do we map the texture onto an arbitrary (complex) object?
  - Construct a mapping between the 3-D point and an intermediate surface
  - Idea: Project each object point to the intermediate surface with a parallel or perspective projection
  - The focal point is usually placed inside the object

- Plane
- Cylinder
- Sphere
- Cube

![Planar projector]
Planar Projector

Orthographic projection onto \( XY \) plane:
\[ u = x, \quad v = y \]

...onto \( YZ \) plane

...onto \( XZ \) plane

courtesy of R. Wolfe
Two-part Mapping

- One solution to the mapping problem is to first map the texture to a simple intermediate surface
- Example: map to cylinder
Cylindrical Projector

- Convert rectangular coordinates \((x, y, z)\) to cylindrical \((r, \mu, h)\), use only \((h, \mu)\) to index texture image
Cylindrical Mapping

Parametric cylinder

\[ x = r \cos 2\pi u \]
\[ y = r \sin 2\pi u \]
\[ z = v/h \]

maps rectangle in \( u,v \) space to cylinder
of radius \( r \) and height \( h \) in world coordinates

\[ s = u \]
\[ t = v \]

maps from texture space
Spherical Projector

- Convert rectangular coordinates \((x, y, z)\) to spherical \((\theta, \phi)\)
Spherical Map

We can use a parametric sphere

\[ x = r \cos 2\pi u \]
\[ y = r \sin 2\pi u \cos 2\pi v \]
\[ z = r \sin 2\pi u \sin 2\pi v \]

in a similar manner to the cylinder but have to decide where to put the distortion

Spheres are used in environmental maps
Parametric Surfaces

- **A parameterized surface patch**
  - $x = f(u, v)$, $y = g(u, v)$, $z = h(u, v)$
  - You will get the mapping via parameterization

![Diagram of parametric surfaces](image)
Box Mapping

- Easy to use with simple orthographic projection
- Also used in environment maps
What's the Best Chart?

Box Map

Latitude Map

GL Map
Second Mapping

- Map from intermediate object to actual object
  - Normals from intermediate to actual
  - Normals from actual to intermediate
  - Vectors from center of intermediate
Projective Textures

- Use the texture like a slide projector
- No need to specify texture coordinates explicitly
- A good model for shading variations due to illumination
- A fair model for reflectance (can use pictures)
Projective Texture Example

- Modeling from photographs
- Using input photos as textures
Texture Tiling

- Specify a texture coordinate \((u,v)\) at each vertex
- Canonical texture coordinates \((0,0) \rightarrow (1,1)\)

Tiles with visible seams

Seamless tiling (repeating)
Specify More Coordinates?

- We can reduce the perceived artifacts by subdividing the model into smaller triangles.

- However, sometimes the errors become obvious
  - At "T" joints
  - Between levels-of-details
Subdivision
Subdivision

texture source  what we get  what we want
Texture Mapping & Illumination

- Texture mapping can be used to alter some or all of the constants in the illumination equation:
  - pixel color, diffuse color, alter the normal, ....

\[ I_{total} = k_a I_{ambient} + \sum_{i=1}^{lights} I_i \left( k_d \left( \hat{N} \cdot \hat{L} \right) + k_s \left( \hat{V} \cdot \hat{R} \right)^{n_{shiny}} \right) \]

Phong's Illumination Model

- Constant Diffuse Color
- Diffuse Texture Color
- Texture used as Label
- Texture used as Diffuse Color
Texture Chart

- Pack triangles into a single image
Procedural and Solid Textures
Procedural Textures

\[ f(x,y,z) \rightarrow \text{color} \]
Procedural Textures

\[ f(x,y,z) \rightarrow \text{color} \]
Procedural Textures

• **Advantages:**
  – easy to implement in ray tracer
  – more compact than texture maps
  – especially for solid textures
  – infinite resolution

• **Disadvantages**
  – non-intuitive
  – difficult to match existing texture
Solid Texture Examples
Procedural Solid Textures

- Noise
- Turbulence
What's Missing?

• What's the difference between a real brick wall and a photograph of the wall texture-mapped onto a plane?

• What happens if we change the lighting or the camera position?
Bump Mapping

• **Other Mapping Techniques:**
  – Bump Mapping
  – Displacement Mapping
Remember Gouraud Shading?

- Instead of shading with the normal of the triangle, shade the vertices with the average normal and interpolate the color across each face.
Phong Normal Interpolation

- Interpolate the average vertex normals across the face and compute *per-pixel shading*.

Must be renormalized.
Bump Mapping

- **Use textures to alter the surface normal**
  - Does not change the actual shape of the surface
  - Just shade as if it were a different shape
Bump Mapping

- Treat the texture as a single-valued height function
- Compute the normal from the partial derivatives in the texture
Another Bump Map Example

Cylinder w/Diffuse Texture Map

Bump Map

Cylinder w/Texture Map & Bump Map
What's Missing?

- There are no bumps on the silhouette of a bump-mapped object

- Bump maps don’t allow self-occlusion or self-shadowing
Displacement Mapping

- Use the texture map to actually move the surface point
- The geometry must be displaced before visibility is determined
Displacement Mapping

Image from:

Geometry Caching for Ray-Tracing Displacement Maps
by Matt Pharr and Pat Hanrahan.

note the detailed shadows cast by the stones
Displacement Mapping
Environment Maps

• We can simulate reflections by using the direction of the reflected ray to index a spherical texture map at "infinity".

• Assume that all reflected rays begin from the same point.
Illumination + Texture Mapping
Texture Maps for Illumination

• Also called "Light Maps"
Questions?