Texture Mapping

Jian Huang

This set of slides references the ones used at Ohio State for instruction.
Can you do this …
What Dreams May Come
Texture Mapping

- Of course, one can model the exact micro-geometry + material property to control the look and feel of a surface.
- But, it may get extremely costly.
- So, graphics use a more practical approach – texture mapping.
Texture Mapping

- Particles and fractals
  - gave us lots of detail information
  - not easy to model
  - mathematically and computationally challenging
Texture Mapping

- (Sophisticated) Illumination models
  - gave us “photo”-realistic looking surfaces
  - not easy to model
  - mathematically and computationally challenging

- Phong illumination/shading
  - easy to model
  - relatively quick to compute
  - only gives us dull surfaces
Texture Mapping

- Surfaces “in the wild” are very complex
- Cannot model all the fine variations
- We need to find ways to add **surface detail**
- How?
Texture Mapping

- Solution - (it's really a cheat!!)

MAP surface detail from a predefined multi-dimensional table ("texture") to a simple polygon

- How?
Textures Make A Difference

- Good textures, when applied correctly, make a world of difference!
Do you wonder what they may look like with textures on?
Do you wonder what they may look like with textures on?
A Texture can be?

- \( F(u,v) \) ==> a continuous or discrete function of:
  - \{ R(u,v), G(u,v), B(u,v) \}
  - \{ I(u,v) \}
  - \{ index(u,v) \}
  - \{ alpha(u,v) \} (transparency)
  - \{ normals(u,v) \} (bump map)
  - \{ surface\_height(u,v) \} (displacement map)
  - Specular color (environment map)
  - ...

The Generalized Pipeline

The generalized pipeline of texture mapping

- Compute obj space location
- Use proj func to find \((u,v)\)
- Use corre func to find texel
- Apply value transform func
- Modify equation or fragment color

- Fragment: after rasterization, the data are not pixels yet, but are fragments. Each fragment has coordinate, color, depth, and undergo a series of tests and ops before showing up in the framebuffer
Texture Mapping

Problem #1

- Fitting a square peg in a round hole
- We deal with non-linear transformations
- Which parts map where?
Inverse Mapping

- Need to transform back to obj/world space to do the interpolation
- Orientation in 3D image space
  - (.5, 1) (.8, 1)
  - (.1, .6) (.6, .2)

- Foreshortening
Texture Mapping

Problem #2

- Mapping from a pixel to a “texel”
- Aliasing is a huge problem!
Mapping to A Texel?

- Basically map to an image
- Need to interpolate
- Same as ....
  - How can I find an appropriate value for an arbitrary (not necessarily integer) index?
    - How would I rotate an image 45 degrees?
    - How would I translate it 0.5 pixels?
Interpolation

Nearest neighbor

Linear Interpolation
How do we get $F(u,v)$?

- We are given a discrete set of values:
  - $F[i,j]$ for $i=0,...,N$, $j=0,...,M$

- Nearest neighbor:
  - $F(u,v) = F[\text{round}(N\times u), \text{round}(M\times v)]$

- Linear Interpolation:
  - $i = \text{floor}(N\times u)$, $j = \text{floor}(M\times v)$
  - interpolate from $F[i,j], F[i+1,j], F[i,j+1], F[i+1,j+1]$

- Filtering in general!
How do we get $F(u,v)$?

- Higher-order interpolation
  - $F(u,v) = \sum_i \sum_j F[i,j] \cdot h(u,v)$
  - $h(u,v)$ is called the reconstruction kernel
    - Gaussian
    - Sinc function
    - splines
- Like linear interpolation, need to find neighbors.
  - Usually four to sixteen
Texture and Texel

- Each pixel in a texture map is called a Texel.
- Each Texel is associated with a \((u,v)\) 2D texture coordinate.
- The range of \(u, v\) is \([0.0, 1.0]\).
For any \((u,v)\) in the range of \((0-1, 0-1)\), we can find the corresponding value in the texture using some interpolation.
The Projector Function

1. Model the mapping: \((x,y,z) \rightarrow (u,v)\)
2. Do the mapping
Image space scan

For each y /* scan-line */
  For each x /* pixel on scan-line */
    compute u(x,y) and v(x,y)
    copy texture(u,v) to image(x,y)

- Samples the warped texture at the appropriate image pixels.
- inverse mapping
Image space scan

Problems:

- Finding the inverse mapping
  - Use one of the analytical mappings
  - Bi-linear or triangle inverse mapping
- May miss parts of the texture map
Texture Parameterization

Definition:

- The process of assigning texture coordinates or a texture mapping to an object.

The mapping can be applied:

- Per-pixel
- Per-vertex
Interpolation Concepts

T is texture
Find textures at vertices first!
Bilinear Interpolation of Depth Values
Texture space scan

For each \( v \)
  For each \( u \)
    compute \( x(u,v) \) and \( y(u,v) \)
    copy texture\((u,v)\) to image\((x,y)\)

- Places each texture sample to the mapped image pixel.
- Forward mapping
Texture space scan

Problems:
- May not fill image
- Forward mapping needed
Simple Projector Functions

- Spherical
- Cylindrical
- Planar

For some model, a single projector function suffices. But very often, an artist may choose to subdivide each object into parts that use different projector
Planar

- **Mapping to a 3D Plane**
  - Simple Affine transformation
    - rotate
    - scale
    - translate
Mapping to a Cylinder

- Rotate, translate and scale in the uv-plane
- \( u \rightarrow \theta \)
- \( v \rightarrow z \)
- \( x = r \cos(\theta), \ y = r \sin(\theta) \)
Spherical

- Mapping to Sphere
  - Impossible!!!!
  - Severe distortion at the poles
  - $u \rightarrow \theta$
  - $v \rightarrow \phi$
  - $x = r \sin(\theta) \cos(\phi)$
  - $y = r \sin(\theta) \sin(\phi)$
  - $z = r \cos(\theta)$
Two-pass Mapping

- Idea by Bier and Sloan
- $S$: map from texture space to intermediate space
- $O$: map from intermediate space to object space
Two-pass Mapping

- Map texture to intermediate:
  - Plane
  - Cylinder
  - Sphere
  - Box
- Map object to same.
Texture Mapping

- O mapping:
  - reflected ray (environment map)
  - object normal
  - object centroid
  - intermediate surface normal (ISN)

- that makes 16 combinations
- only 5 were found useful
Texture Mapping

- **Cylinder/ISN (shrinkwrap)**
  - Works well for solids of revolution

- **Plane/ISN (projector)**
  - Works well for planar objects

- **Box/ISN**
- **Sphere/Centroid**
- **Box/Centroid**

Works well for roughly spherical shapes
Texture Parameterization

- What is this ISN?
  - Intermediate surface normal.
  - Needed to handle concave objects properly.
  - Sudden flip in texture coordinates when the object crosses the axis.
Texture Parameterization

- Flip direction of vector such that it points in the same half-space as the outward surface normal.
Texture Parameterization

- Plane/ISN
Texture Parameterization

- **Plane/ISN**
  - Draw vector from point (vertex or object space pixel point) in the direction of the texture plane.

  - The vector will intersect the plane at some point depending on the coordinate system.
Texture Parameterization

- Plane/ISN
  - Resembles a slide projector
  - Distortions on surfaces perpendicular to the plane.
Texture Parameterization

- Cylinder/ISN
  - Distortions on horizontal planes
  - Draw vector from point to cylinder
  - Vector connects point to cylinder axis
Texture Parameterization

- Sphere/ISN
  - Small distortion everywhere.
  - Draw vector from sphere center through point on the surface and intersect it with the sphere.
Interpolating Without Explicit Inverse Transform

- Scan-conversion and color/z/normal interpolation take place in screen space, but really, what space should it be in?
- What about texture coordinates?
  - Do it in clip space, or homogenous coordinates
In Clip space

- Two end points of a line segment (scan line)
  \[ Q_1 = (x_1, y_1, z_1, w_1) \quad Q_2 = (x_2, y_2, z_2, w_2) \]

- Interpolate for a point \( Q \) in-between
  \[ Q = (1 - t)Q_1 + tQ_2 \]
In Screen Space

- From the two end points of a line segment (scan line), interpolate for a point Q in-between:
  
  \[ Q^g = (1 - t^g)Q_1^g + t^gQ_2^g \]

- Where: \( Q_1^g = Q_1/w_1 \) and \( Q_2^g = Q_2/w_2 \).

- Easy to show: in most occasions, \( t \) and \( t^s \) are different.
From $t^s$ to $t$

- Change of variable: choose
  - $a$ and $b$ such that $1 - t^s = a/(a + b)$, $t^s = b/(a + b)$
  - $A$ and $B$ such that $(1 - t) = A/(A + B)$, $t = B/(A + B)$.

- Easy to get
  \[ Q^s = \frac{aQ_1/w_1 + bQ_2/w_2}{(a + b)} = \frac{AQ_1 + BQ_2}{Aw_1 + BW_2} \]

- Easy to verify: $A = aw_2$ and $B = bw_1$ is a solution
Texture Coordinates

- All such interpolation happens in homogeneous space.
- Use A and B to linearly interpolate texture coordinates
- The homogeneous texture coordinate is: \((u,v,1)\)
Homogeneous Texture Coordinates

- \( u^l = \frac{A}{A+B} u_1^l + \frac{B}{A+B} u_2^l \)
- \( w^l = \frac{A}{A+B} w_1^l + \frac{B}{A+B} w_2^l = 1 \)
- \( u = u^l/w^l = u^l = (Au_1^l + Bu_2^l)/(A + B) \)
- \( u = (au_1^l + Bu_2^l)/(A + B) \)
- \( u = (au_1^l/w_1^l + bu_2^l/w_2^l )/(a^l/w_1^l + b^l/w_2^l) \)
Homogeneous Texture Coordinates

- The homogeneous texture coordinates suitable for linear interpolation in screen space is computed simply by
  - Dividing the texture coordinates by screen \( w \)
  - Linearly interpolating \((u/w, v/w, 1/w)\)
  - Dividing the quantities \(u/w\) and \(v/w\) by \(1/w\) at each pixel to recover the texture coordinates