CS130 : Computer Graphics

Lecture 9: Texture Mapping (cont.)

Tamar Shinar
Computer Science & Engineering
UC Riverside
The major issues in texture mapping...

- What should the actual mapping be?

   easy: rectangular surface  harder: parametric surface

Teapot: Which image looks better? The image on the left uses **object coordinates** in the texture mapping – this makes more sense. The image on the right uses **world coordinates** – texture ends up changing relative to the object. **Want a nice map that doesn’t look distorted**
Given a point on the object \((x, y, z)\), what point \((u, v)\) in the texture we use?
Example: planar mapping
Intermediate surfaces

First map the texture to a simpler, intermediate surface
Cylindrical mapping

\[(x, y, z) \rightarrow (\theta, h) \rightarrow (u, v)\]

- note “pie slice” phenomena
- which coordinate axis is parallel to the cylinder axis?
Spherical Mapping

$(x, y, z) \rightarrow (\text{latitude}, \text{longitude}) \rightarrow (u, v)$

spherical map stretches squares at equator and squeezes squares at poles
Box Mapping

- similar to planar mapping
- planar projection -- choose which plane to project onto
How do we map between intermediate and actual objects?

We associated (x,y,z) on the intermediate object with the texture (u,v). But which point on the actual object is this?

We choose both the **intermediate shape** and the **mapping from the actual shape to the intermediate shape**

1. a point on the object relative to its bounding box
2. see where surface normal intersects intermediate surface
3. shoot ray from centroid through surface point to intermediate surface
4. use the reflection vector (depends on the viewer position and normal)
How do we map between intermediate and actual objects?

Can you tell what intermediate shape was used?  
Planar map - in xy plane
Parametric Surfaces

32 parametric patches
3D solid textures can map object \((x, y, z)\) directly to texture \((u, v, w)\).
Procedural textures

e.g., Perlin noise
Triangles
Texturing triangles

```
glTexCoord*();
```

Texture Space

Object Space

$$\text{(u,v)} = (0.2, 0.8)$$

A

B

C

$$\text{(0.8, 0.4)}$$

$$\text{(0.4, 0.2)}$$

Angel and Shreiner
Multitexturing
Texture Sampling
Texture Mapping

- Texture coordinates: Used to identify points in the image to be mapped
- Object Coordinates: Conceptually, where the mapping takes place
- Window Coordinates: Where the final image is really produced
Point Sampling

Map back to texture image and use the **nearest texel**
Point sampling of the texture can lead to aliasing artifacts. Point samples in (or x,y,z) space miss blue stripes. Point samples in texture space.
Magnification and Minification
Magnification and Minification

More than one texel can cover a pixel (minification) or more than one pixel can cover a texel (magnification).

Can use point sampling (nearest texel) or linear filtering (2 x 2 filter) to obtain texture values.
Aliasing artifacts

We apply **filtering** to reduce aliasing artifacts
A better but slower option is to use **area averaging**
Use bilinear filtering

```
<table>
<thead>
<tr>
<th></th>
<th>p3</th>
<th></th>
<th>p2</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td></td>
<td>p</td>
<td></td>
</tr>
<tr>
<td>p0</td>
<td></td>
<td>p1</td>
<td></td>
</tr>
</tbody>
</table>
```

- nearest neighbor
- bilinear
- bicubic

mitigate magnification artifacts

sMOOTHs out the texture – no sharp boundaries
Mipmapping

Reduce minification artifacts
Prefilter the texture to obtain reduced resolutions
Requires 1/3 more space
Get a texture hierarchy indexed by level

Togikun, Wikimedia Commons

128×128, 64×64, 32×32, 16×16, 8×8, 4×4, 2×2, 1×1
point sampling

mipmapped point sampling

linear filtering

mipmapped linear filtering

[Angel and Shreiner]