Texture Mapping

Brian Curless
CSEP 557
Spring 2019
Reading

Optional

- Angel and Shreiner: 7.4-7.10
- Marschner and Shirley: 11.1-11.2.3, 11.2.5, 11.4-11.5

Further reading

- Woo, Neider, & Davis, Chapter 9
Texture mapping

Texture mapping allows you to take a simple polygon and give it the appearance of something much more complex.

- Due to Ed Catmull, PhD thesis, 1974
- Refined by Blinn & Newell, 1976

A texture can modulate just about any parameter – diffuse color, specular color, specular exponent, …
Implementing texture mapping

A texture lives in its own abstract image coordinates parameterized by \((u, v)\) in the range \([0..1, 0..1]\):

It can be wrapped around many different surfaces:

With a ray caster, we can do the sphere and cylinder mappings directly (as we will see later). For graphics hardware, everything gets converted to a triangle mesh with associated \((u, v)\) coordinates.

Note: if the surface moves/deforms, the texture goes with it.
Recall that for a surface of revolution, we have:

- **Profile curve:** $C[j]$ where $j \in [0..M-1]$

- **Rotation angles:** $\theta[i] = \frac{2\pi i}{N}$ where $i \in [0..N]$

The simplest assignment of texture coordinates would be:

- $U = \frac{i}{N}$
- $V = \frac{j}{M-1}$

Note that you should include the rotation angles for $i = 0$ and $i = N$, even though they produce the same points (after rotating by $0$ and $2\pi$). Why do this??
Texture coordinates on a surface of revolution

If we wrap an image around this surface of revolution, what artifacts would we expect to see?

We can reduce distortion in $v$. Define:

$$d[j] = \begin{cases} \|C[j] - C[j-1]\|, & \text{if } j \neq 0 \\ 0, & \text{if } j = 0 \end{cases}$$

and set $v$ to fractional distance along the curve:

$$v = \frac{\sum_{k=0}^{j} d[k]}{\sum_{k=0}^{M-1} d[k]}$$

You must do this for $v$ for the programming assignment!
Mapping to texture image coords

The texture is usually stored as an image. Thus, we need to convert from abstract texture coordinate:

\((u, v)\) in the range \(([0..1], [0..1])\)

to texture image coordinates:

\((u_{\text{tex}}, v_{\text{tex}})\) in the range \(([0.. w_{\text{tex}}], [0.. h_{\text{tex}}])\)

Q: What do you do when the texture sample you need lands between texture pixels?
Texture resampling

We need to resample the texture:

Thus, we seek to solve for: \( T(a, b) = T(i + \Delta_x, j + \Delta_y) \)

A common choice is **bilinear interpolation:**

\[
T(i + \Delta_x, j) = \frac{(1-\Delta_x)}{\Delta_x} T[i, j] + \frac{\Delta_x}{\Delta_x} T[i+1, j]
\]

\[
T(i + \Delta_x, j + 1) = \frac{(1-\Delta_y)}{\Delta_y} T[i, j+1] + \frac{\Delta_y}{\Delta_y} T[i+1, j+1]
\]

\[
T(i + \Delta_x, j + \Delta_y) = \frac{(1-\Delta_y)}{\Delta_y} T(i + \Delta_x, j) + \frac{\Delta_y}{\Delta_y} T(i + \Delta_x, j + 1)
\]

\[
= \frac{(1-\Delta_y)(1-\Delta_x)}{\Delta_x \Delta_y} T[i, j] + \frac{\Delta_x}{\Delta_x} \left(1 - \frac{1}{\Delta_x} \right) T[i+1, j] + \]

\[
\frac{(1-\Delta_y)}{\Delta_y} T[i, j+1] + \frac{\Delta_x}{\Delta_x} \Delta_y T[i+1, j+1]
\]
Texture mapping and rasterization

Texture-mapping can also be handled in rasterization algorithms.

**Method:**
- Scan conversion is done in screen space, as usual
- Each pixel is colored according to the texture
- Texture coordinates are found by Gouraud-style interpolation

**Note:** Mapping is more complicated to handle perspective correctly.
Displacement mapping

Textures can be used for more than just color.

In **displacement mapping**, a texture is used to perturb the surface geometry itself. Here’s the idea in 2D:

\[
\tilde{Q}(u) = Q(u) + d(u)N(u)
\]

- These displacements “animate” with the surface
- In 3D, you would of course have \((u, v)\) parameters instead of just \(u\).

Suppose \(Q\) is a simple surface, like a cube. Will it take more work to render the modified surface \(\tilde{Q}\)?
Bump and normal mapping

In **bump mapping**, a texture is used to perturb the normal:

- Use the original, simpler geometry, $Q(u)$, for hidden surfaces
- Use the normal from the displacement map for shading:

$$\tilde{N} = \text{normal}[\tilde{Q}(u)]$$

An alternative is to compute the normals from the original bump map height field and map them over the smooth surface. This is called **normal mapping**.

What artifacts in the images would reveal that bump (or normal) mapping is fake?
Displacement vs. bump mapping

Input texture

Rendered as displacement map over a rectangular surface
Displacement vs. bump mapping (cont'd)

Original rendering  Rendering with bump map wrapped around a cylinder

*Bump map and rendering by Wyvern Aldinger*
Solid textures

**Q:** What kinds of artifacts might you see from using a marble veneer instead of real marble?

One solution is to use **solid textures**: 
- Use model-space coordinates to index into a 3D texture
- Like “carving” the object from the material

One difficulty of solid texturing is coming up with the textures.
Solid textures (cont'd)

Here's an example for a vase cut from a solid marble texture:

*Solid marble texture by Ken Perlin, (Foley, IV-21)*
Solid textures (cont'd)

\[ \text{in}(x, y, z) = \text{stripes}(x) \]
\[ \text{shift}(x, y, z) = K \cdot \text{noise}(x, y, z) \]
\[ \text{out}(x, y, z) = \text{stripes}(x + \text{shift}(x, y, z)) \]
Environment mapping

In **environment mapping** (also known as **reflection mapping**), a texture is used to model an object's environment:

- Rays are bounced off objects into environment
- Color of the environment used to determine color of the illumination
- Environment mapping works well when there is just a single object – or in conjunction with ray tracing

This can be readily implemented (without interreflection) in graphics hardware using a fragment shader, where the texture is stored in a “cube map” instead of a sphere.

With a ray tracer, the concept is easily extended to handle refraction as well as reflection (and interreflection).
Summary

What to take home from this lecture:

1. The meaning of the boldfaced terms.

2. Familiarity with the various kinds of texture mapping, including their strengths and limitations.