Texture Mapping

Brian Curless  
CSE 457  
Autumn 2010

Reading

Required
- Angel, 8.6, 8.7, 8.9, 8.10, 9.13-9.13.2

Recommended

Optional
- Woo, Neider, & Davis, Chapter 9

Implementing texture mapping

A texture lives in its own abstract image coordinates parameterized by \((s, t)\) 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 z-buffers, everything gets converted to a triangle mesh with associated \((s, t)\) coordinates.

Note: if the surface moves/deforms, the texture goes with it.
Mapping to texture image coords

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

\[(s,t)\] in the range \([0..1], [0..1]\)

to texture image coordinates:

\[(s_{tex}, t_{tex})\] in the range \([0..w_{tex}], [0..h_{tex}]\)

**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**:

\[T(i+\Delta_x, j) = \frac{1}{2} T(i, j) + \frac{1}{2} T(i+1, j)\]
\[T(i+\Delta_x, j+1) = \frac{1}{2} T(i, j+1) + \frac{1}{2} T(i+1, j+1)\]
\[T(i+\Delta_x, j+\Delta_y) = \frac{1}{4} T(i, j) + \frac{1}{4} T(i+1, j) + \frac{1}{4} T(i, j+1) + \frac{1}{4} T(i+1, j+1)\]

Texture mapping and the z-buffer

Texture-mapping can also be handled in z-buffer 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:

- These displacements "animate" with the surface
- In 3D, you would of course have \((s,t)\) parameters instead of just \(s\).

Suppose \(Q\) is a simple surface, like a sphere. Will it take more work to render the modified surface \(Q\)?
**Bump mapping**

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

- Use the original, simpler geometry, $\mathbf{Q}(s)$, for hidden surfaces.
- Use the normal from the displacement map for shading:

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

What artifacts in the images would reveal that bump mapping is a fake?

**Displacement vs. bump mapping**

**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)*

**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.

**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) 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).