Texture Mapping

Daniel Leventhal
Adapted from Brian Curless
CSE 457
Autumn 2011

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, …

Reading

Recommended

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

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:

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

\[ Q: \text{What do you do when the texture sample you need lands between texture pixels?} \]

Texture resampling

We need to resample the texture:

\[ \theta \leq \alpha, \theta, \alpha \leq 1 \]

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

A common choice is \textit{bilinear interpolation}:

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

Texture mapping and the z-buffer

Texture-mapping can also be handled in z-buffer algorithms.

\textbf{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

\[ Q:\text{Mapping is more complicated to handle perspective correctly!} \]

Displacement mapping

Textures can be used for more than just color.

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

\[ \Delta \mathbf{Q}(s) = \mathbf{Q}(s) + \mathbf{d}(s) \mathbf{N}(s) \]

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

Suppose \( \mathbf{Q} \) is a simple surface, like a cube. \textbf{Will it take more work to render the modified surface} \( \mathbf{Q} + \Delta \mathbf{Q} \)?
Bump mapping

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

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

\[
\vec{N} = \text{normal}[Q(s)]
\]

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

*Shows silhouettes, lighting, perspective*

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)

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).

HDR Capture
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.