Texture Mapping

Reading

Required
- Shirley, 11.1-11.3, 11.5-11.7

Recommended

Optional
- 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

Texture mapping ensures that “all the right things” happen as a textured polygon is transformed and rendered.

Non-parametric texture mapping

With “non-parametric texture mapping”:
- Texture size and orientation are fixed
- They are unrelated to size and orientation of polygon
- Gives cookie-cutter effect
**Parametric texture mapping**

With “parametric texture mapping,” texture size and orientation are tied to the polygon.

**Idea:**
- Separate “texture space” and “screen space”
- Texture the polygon as before, but in texture space
- Deform (render) the textured polygon into screen space

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:

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 coordinates:

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

to texture image coordinates:

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

**Texture resampling**

We need to resample the texture:

A common choice is **bilinear interpolation**: 

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

Q: What do you do when the texture sample you need lands between texture pixels?
**Displacement mapping**

Textures can be used for more than just color.

In **displacement mapping**, a texture is used to perturb the surface geometry itself:

\[
\begin{align*}
\tilde{Q}(u) &= Q(u) + d(u) N(u) \\
N(u) &= \text{normal}[Q(u)]
\end{align*}
\]

- These displacements "animate" with the surface

Q: Do you have to do hidden surface calculations on \( \tilde{Q} \)?

**Bump 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:

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

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

**Displacement vs. bump mapping**

Input texture

Rendered as displacement map over a rectangular surface

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)

\[
\begin{align*}
\text{in}(x,y,z) &= \text{stripes}(x) \\
\text{shift}(x,y,z) &= K \cdot \text{noise}(x) \\
\text{out}(x,y,z) &= \text{stripes}(x + \text{shift}(x))
\end{align*}
\]

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
- Really, a simplified form of ray tracing
- Environment mapping works well when there is just a single object – or in conjunction with ray tracing

Under simplifying assumptions, environment mapping can be implemented in hardware.

With a ray tracer, the concept is easily extended to handle refraction as well as reflection.