12. Texture Mapping

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

Reading

Required

- Watt, intro to Chapter 8 and intros to 8.1, 8.4, 8.6, 8.8.

Optional

- Watt, the rest of Chapter 8
- Woo, Neider, & Davis, Chapter 9
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 it own image coordinates paramaterized by \( (u,v) \):

It can be wrapped around many different surfaces:

Computing \( (u,v) \) texture coordinates in a ray tracer is fairly straightforward.

Texture resampling

The texture is usually stored as an image.

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

We need to **resample** the texture.

A common choice is **bilinear** resampling:

\[
T(i+a,j+b) = \frac{a}{\alpha} T[i,j] + \frac{b}{\beta} T[i+1,j] + \frac{a}{\gamma} T[i,j+1] + \frac{b}{\delta} T[i+1,j+1]
\]

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 if you want to do perspective right!
Antialiasing

If you point-sample the texture map, you get aliasing:

Proper antialiasing requires area averaging in the texture:

Computing the average color

The computationally difficult part is summing over the covered pixels.

Several methods have been used:

1. Brute force:
   - Just sum
   - (Original method)

Mip Maps

The mip map hierarchy can be thought of as an image pyramid:

- Level 0 ($T_0[i,j]$) is the original image.
- Level 1 ($T_1[i,j]$) averages over 2x2 neighborhoods of original.
- Level 2 ($T_2[i,j]$) averages over 4x4 neighborhoods of original
- Level 3 ($T_3[i,j]$) averages over 8x8 neighborhoods of original

What’s a fast way to pre-compute the texture map for each level?

What would the mip-map return for an average over a 5x5 neighborhood at location $(u,v)$?

What if you need to average over a non-square region?
3. Summed area tables:
   - Frank Crow, 1984
   - Keep sum of everything below and to the left
   - Use four table lookups:

   ![Summed area table diagram](image)

   Recall from calculus:
   \[
   \int_{c}^{d} f(x) dx = \int_{-\infty}^{d} f(x) dx - \int_{-\infty}^{c} f(x) dx
   \]

   - Requires more memory
   - Gives less blurry textures

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](image)
Displacement mapping

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

\[
\begin{align*}
Q(u) & \quad \text{Original geometry} \\
N(u) &= \text{normal}[Q(u)] \\
d(u) & \quad \text{Displacement vector}
\end{align*}
\]

- These displacements “animate” with the surface
- Requires doing additional hidden surface calculations

\[
Q(u) = Q(u) + d(u)N(u)
\]

Bump mapping

Textures can be used for more than just color.

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

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

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

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

Displacement vs. bump mapping (cont’d)

Original rendering Rendering with bump map wrapped around a cylinder

*Bump map and rendering by Wyvern Aldinger*
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.

Combining texture maps

Using texture maps in combination gives even better effects, as *Young Sherlock Holmes* demonstrated …

**Combining texture maps (cont’d)**

- Phong lighting with diffuse texture
- Environment-mapped mirror reflection
- Bump mapping + Glossy reflection
- Combine textures and add dirt
- Rivet stains + Shinier reflections
- Close-up

**Construction of the glass knight, (Foley, IV-24)**

**Summary**

What to take home from this lecture:

1. The meaning of the boldfaced terms.
2. Understanding of the various approaches to antialiased texture mapping:
   - Brute force
   - Mip maps
   - Summed area tables
3. Familiarity with the various kinds of texture mapping, including their strengths and limitations.