9. Texture Mapping

Reading

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

Recommended

Optional
- Watt, the rest of Chapter 8
- Woo, Neider, & Davis, Chapter 9
What adds visual realism?

- Geometry only
- Phong shading
- Phong shading + Texture maps

Texture mapping

Texture mapping (Woo et al., fig. 9-1)

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:

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

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:

\((u,v)\) 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:

A common choice is **bilinear interpolation**:

\[
T(a,b) = T(i + \Delta_x, j + \Delta_y)
\]

\[
= \left(1 - \alpha x\right) \left(1 - \alpha y\right) T[i,j] + \\
\alpha x (1 - \alpha y) T[i+1,j] + \\
\alpha x \alpha y T[i,j+1] + \\
(1 - \alpha x) \alpha y T[i+1,j+1]
\]

### 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)](image)

Displacement mapping

Textures can be used for more than just color.

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

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

- These displacements “animate” with the surface

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

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.

Diffuse color Specular coefficient Material properties (coefficients in shading equation)

Environment map (not necessary in ray tracer)
Can define material by program

A ‘surface shader’ computes the color of each ray that hits the surface.

Example: Renderman surface shader

```c
/*
  * Checkerboard
  */
surface checker(float Kd=.5, Ka=.1) {
  float smod = mod(10*s, 1);
  float tmod = mod(10*t, 1);
  if (smod < 0.5) {
    if (tmod < 0.5) Ci=Cs; else Ci=color(0,0,0);
  } else {
    if (tmod < 0.5) Ci=color(0,0,0); else Ci=Cs;
  }
  Oi = Os;
  Ci = Oi*Ci*(
    Ka*ambient() +
    Kd*diffuse(faceforward(normalize(N),I)));
}
```

How do we anti-alias textures?

We could just super-sample.

But textures (and shader programs) are a special case; we can use true area integration!

- Approximate footprint as parallelogram
- Determine this approximate footprint using discrete differences
Cost of filtering can be reduced

Store a pyramid of pre-filtered images:

During texture lookup, read from appropriate level of the pyramid.

Next time: Hierarchical modeling

How do we represent translation and rotation of complex objects using hierarchies of transformations?

(Easy in principle, tough to get right in practice)

Read:
• Angel, sections 9.1 - 9.6 [reader pp. 169-185]
• OpenGL Programming Guide, chapter 3 [available online]