12. Texture Mapping

Reading

Recommended:

- Watt, sections 7.5 – 7.10
- Woo, Neider, & Davis, Chapter 9

Optional:

- James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Communications of the ACM* 19(10): 542–547, October 1976.
- Paul S. Heckbert. Survey of texture mapping. *IEEE Computer Graphics and Applications* 6(11): 56–67, November 1986.

Texture mapping

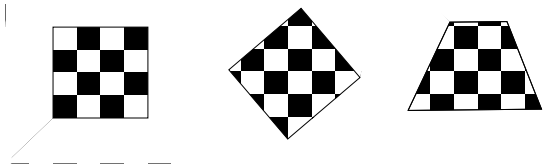
Texture mapping (Woo et al., fig 9-1)

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.

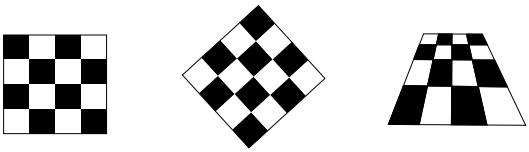
Non-parametric texture mapping



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.

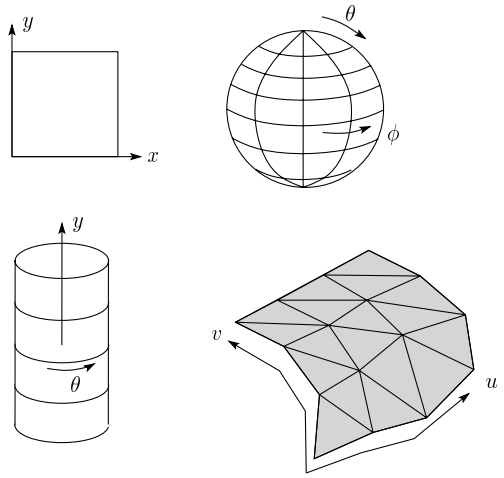
Idea:

- 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 lives in it own image coordinates paramaterized by (u, v).

Implementing texture mapping

Textures can be wrapped around many different surfaces:

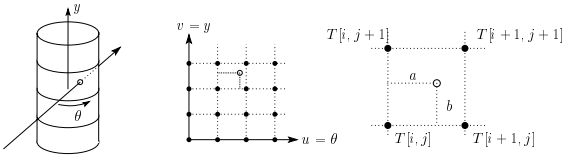


Computing (u, v) texture coordinates in a ray tracer is fairly straightforward.

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?



Ray intersection | Mapping to texture pixels | Close-up

We need to **resample** the texture.

A common choice is **bilinear** resampling:

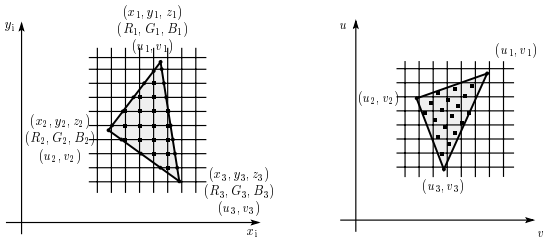
$$T(i+a, j+b) = \_\_\_\_\_T[i,j] + \_\_\_\_\_T[i+1,j] + \_\_\_\_\_T[i,j+1] + \_\_\_\_\_T[i+1,j+1]$$

Texture mapping and the z-buffer

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

Method:

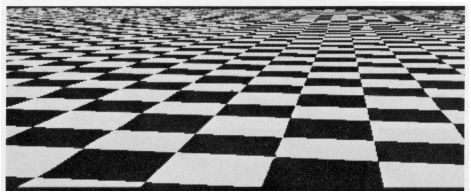
- Scan conversion is done in screen space, as usual
- Each pixel is colored according to the texture
- Texture coordinates are found by Gouraud-style interpolation



Note: Mapping is more complicated if you want to do perspective right!

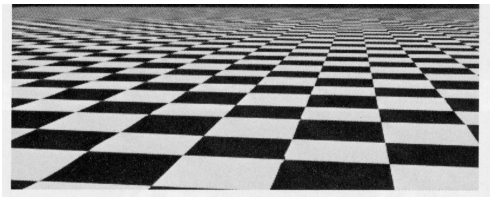
Antialiasing

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



From Crow, SIGGRAPH '84

Proper antialiasing requires area averaging in the texture:



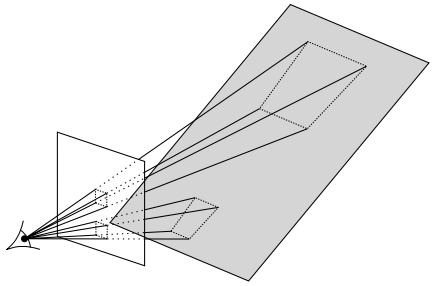
From Crow, SIGGRAPH '84

Computing the average color

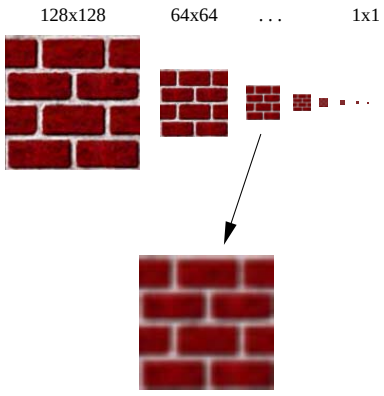
The computationally difficult part is summing over the covered pixels.

Several methods have been used:

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



Computing the average color, cont'd



2. **Mip maps**:
   - Lance Williams, 1983
   - Stands for "multum in parvo" — many things in a small place
   - Keep textures prefiltered at multiple resolutions.
   - Figure out closest two levels
   - Linear interpolate between the two

Computing the average color, cont'd

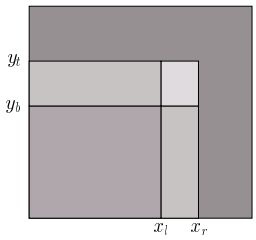
The mip map hierarchy can be thought of as an image pyramid:

- Level 0 is the original image.
- Level 1 averages over 2x2 neighborhoods of original
- Level 2 averages over 4x4 neighborhoods of original
- Level 3 averages over 8x8 neighborhoods of original

What would the mip-map return for an average over a 5x5 neighborhood at location (u_1, v_1)?

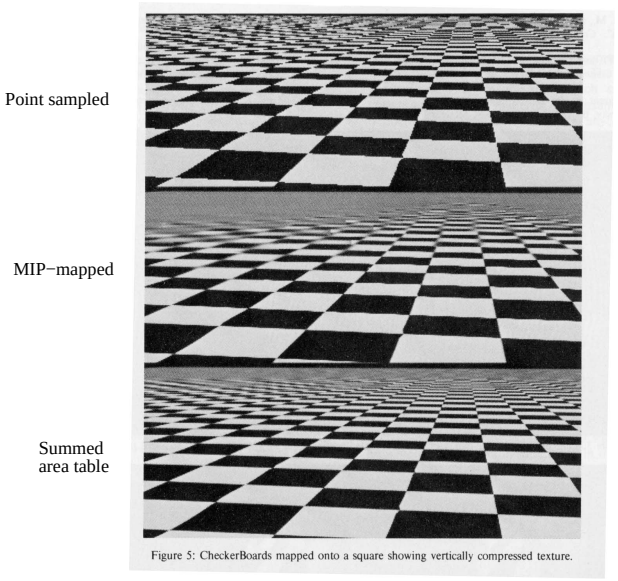
Computing the average color, cont'd

3. **Summed area tables**:
   - Frank Crow, 1984
   - Keep sum of everything below and to the left
   - Use four table lookups:



   - Requires more memory
   - Gives less blurry textures

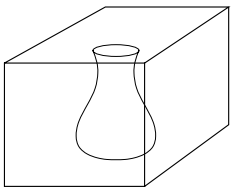
Comparison of techniques



From Crow, SIGGRAPH '84

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. . . .

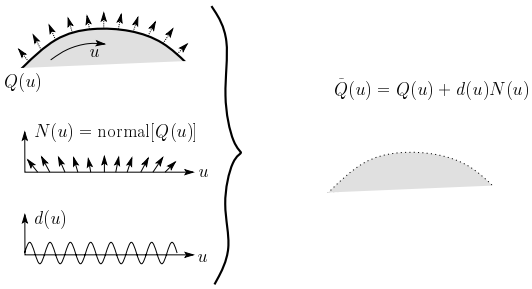
Solid textures, cont'd

Here's an example for a vase cut from a solid marble texture:

Solid marble texture by Ken Perlin, (Foley, IV-21)

Displacement mapping

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



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

Bump mapping

Textures can be used for more than just color.

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

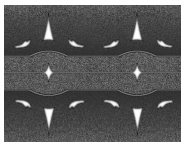
- Use the original geometry, $Q(u)$, for hidden surfaces
- Use the normal from the displacement map for shading:

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

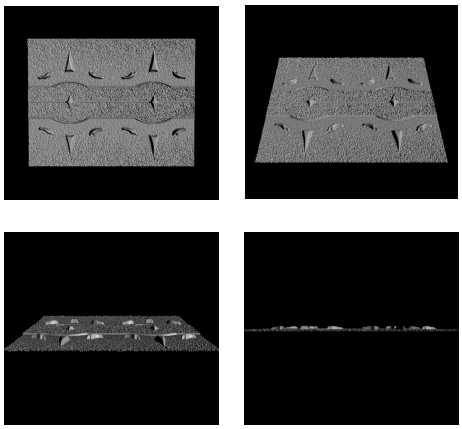
Q: What artifacts in the images would reveal that bump mapping is a fake?

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:

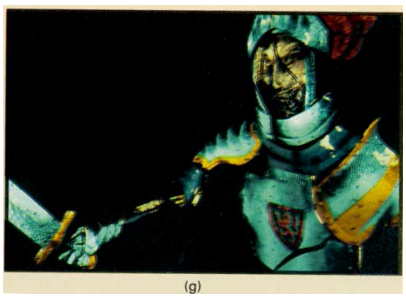
- Rays are bounced off objects into environment
- Color of the environment used to determine color of the illumination
- Really, a simplified form of ray tracing
- Environment mapping works well when there is just a single object — or in conjunction with ray tracing

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.

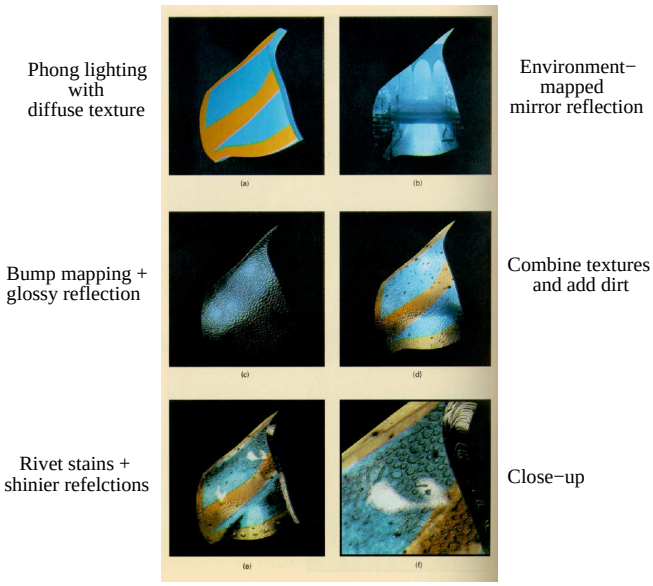
Combining texture maps

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



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

Combining texture maps, cont'd



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.