Texture Mapping

Reading

Required

- Angel, 7.6-7.8.

Recommended

- Paul S. Heckbert. Survey of texture mapping. **IEEE Computer Graphics and Applications** 6(11): 56--67, November 1986.

Optional

- OpenGL Programming Guide, Woo, Neider, & Davis, Chapter 9
- James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. **Communications of the ACM** 19(10): 542--547, October 1976.

Texture mapping

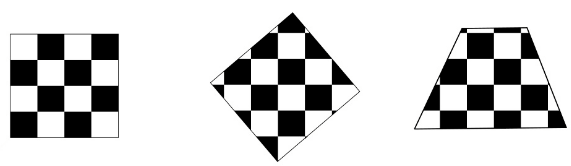
Texture mapping (Woo et al., fig. 9-1)

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.

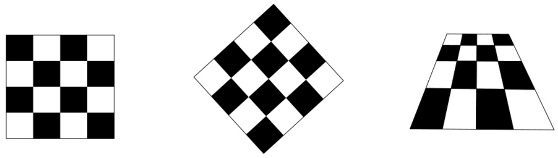
Non-parametric texture mapping



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.

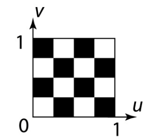
<u>Idea</u>:

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

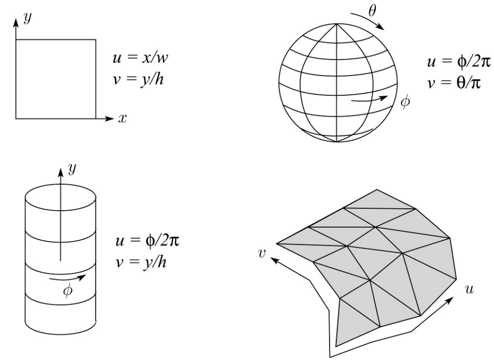
A texture can modulate just about any parameter – diffuse color, specular color, specular exponent, …

Implementing texture mapping

A texture lives in it own 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: as the surface moves or deforms, the texture goes with it.

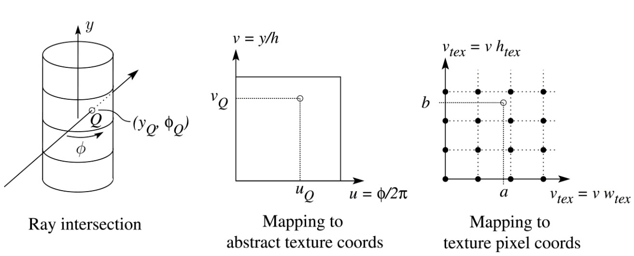
Mapping to texture image coords

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:

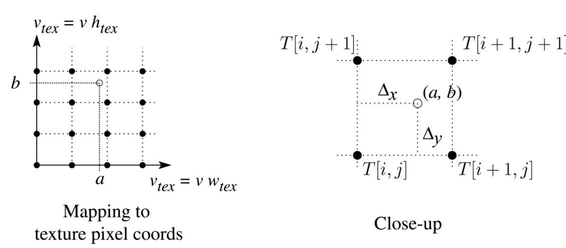
(u_{tex},v_{tex}) in the range ([0.. w_{tex}], [0.. h_{tex}])



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**:

$$T(a,b) = T\left(i+\Delta_x, j+\Delta_y\right)$$

$$= \_\_\_\_\_\_\_\_\_\_ T[i,j] +$$

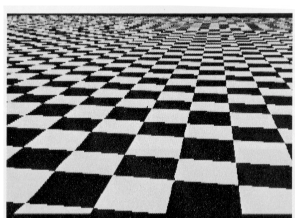
$$\_\_\_\_\_\_\_\_\_\_ T[i+1,j] +$$

$$\_\_\_\_\_\_\_\_\_\_ T[i,j+1] +$$

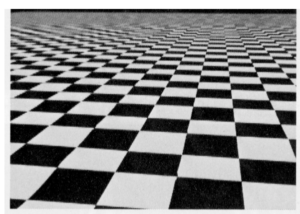
$$\_\_\_\_\_\_\_\_\_\_ T[i+1,j+1]$$

Antialiasing

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



Proper antialiasing requires area averaging in the texture:



From Crow, SIGGRAPH '84

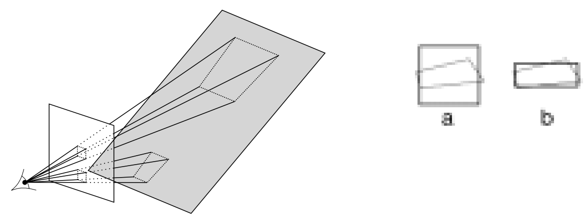
Computing the average color

The computationally difficult part is summing over the covered pixels.

Several methods have been used:

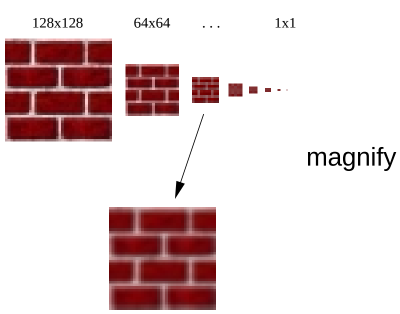
The simplest is **brute force**:

Figure out which texels are covered and add up their colors to compute the average.



Approximating a quadrilateral texture area with (a) a square, (b) a rectangle. Too small an area causes aliasing; too large an area causes blurring. After Heckbert 86.

Mip maps



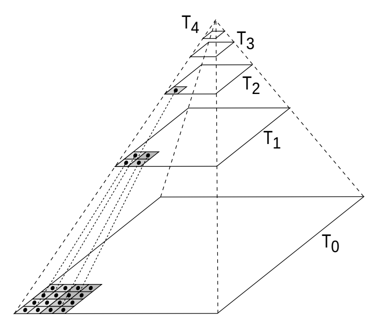
A faster method is **mip maps** developed by Lance Williams

- Stands for "multum in parvo" – many things in a small place
- Keep textures prefiltered at multiple resolutions
- Has become the graphics hardware standard
- figure out the closest two levels and then interpolate

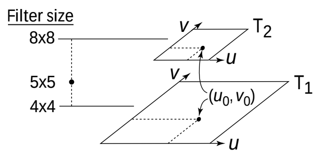
Mip map pyramid

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



Mip map resampling



What would the mip-map return for an average over a 5x5 neighborhood at location (u_0,v_0)?

Summed area tables

A more accurate method than mip maps is **summed area tables** invented by Frank Crow.

Rectangles vs squares.

Recall from calculus:

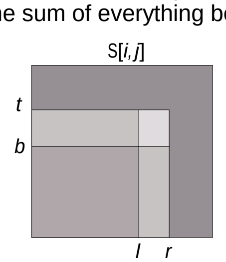
$$\int_a^b f(x)dx = \int_{-\infty}^{b} f(x)dx - \int_{-\infty}^{a} f(x)dx$$

In discrete form:

$$\sum_{i=k}^{m} f[i] = \sum_{i=0}^{m} f[i] - \sum_{i=0}^{k} f[i]$$

Summed area tables (cont'd)

We can extend this idea to 2D by creating a table, S[*i*,*j*], that contains the sum of everything below and to the left.



Q: How do we compute the average over a region from (*l*, *b*) to (*r*, *t*)?

Characteristics:

Requires more memory
Gives less blurry textures

Comparison of techniques

Point sampled

MIP-mapped

Summed area table

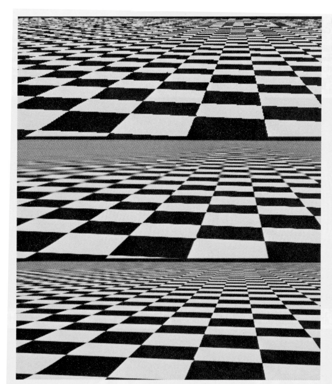


Figure 5: CheckerBoards mapped onto a square showing vertically compressed texture.

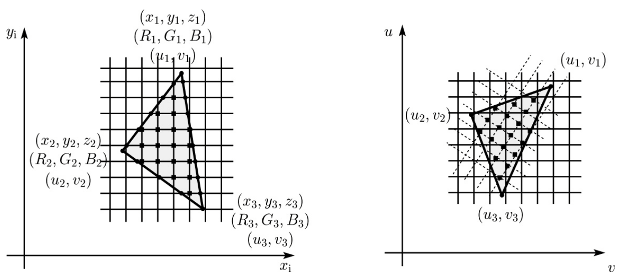
From Crow, SIGGRAPH '84

Texture mapping and the z-buffer

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

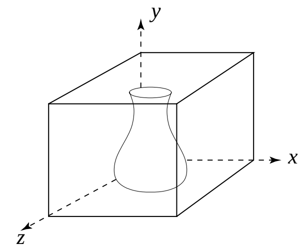
Method:

- 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



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)

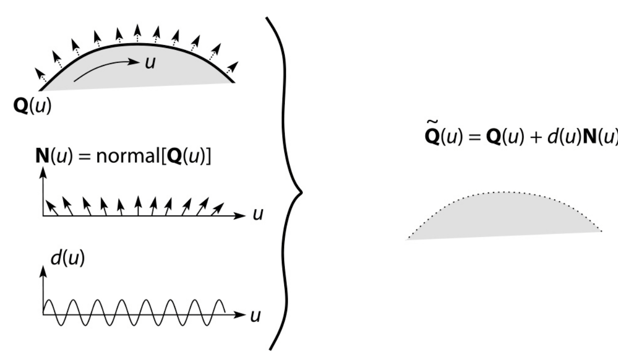
Here's an example for a vase cut from a solid marble texture:

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

Displacement mapping

Textures can be used for more than just color.

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



- These displacements move with the surface

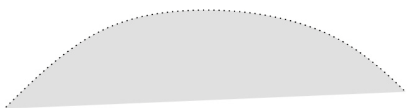
Q: Do you have to do hidden surface calculations on $\tilde{\mathbf{Q}}$?

Bump mapping

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

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

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

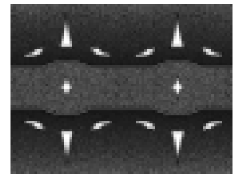


$\mathbf{Q}(u)$

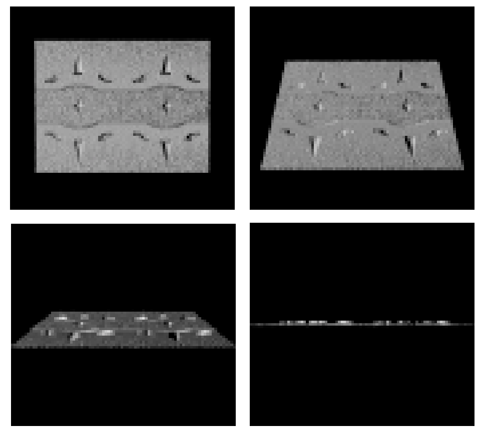
Q: What artifacts in the images would reveal that bump mapping is a fake?

Displacement vs. bump mapping

Input texture



Rendered as displacement map over a rectangular surface



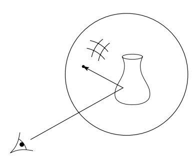
Displacement vs. bump mapping (cont'd)

Original rendering

Rendering with bump map wrapped around a cylinder

Bump map and rendering by Wyvern Aldinger

Environment mapping



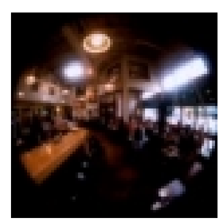
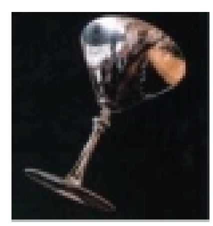


Image of gazing sphere (taken in a pub)



Environment-mapped rendering

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

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)

Phong lighting with diffuse texture

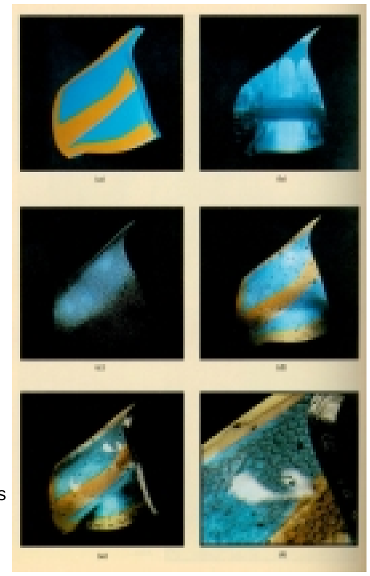
Environment-mapped mirror reflection

Bump mapping + textures Glossy reflection

Combine 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:

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