Lecture Notes #11 - Texture Maps

• Reading: Foley et al., Sections 16.3, 16.4, 16.6, 16.12, 17.4.3.
H&B: Sections 14-8, 14-9.

Optional:

• James Blinn and Martin Newell. Texture and reflection in computer generated images. *Communications of the ACM* 19 (10): 542--547, Oct. 1976

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

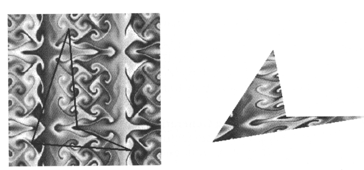
• Paul Haeberli and Mark Segal. Texture mapping as a fundamental drawing primitive. *Proceedings of the Fourth Eurographics Workshop on Rendering* (Paris, France), 259--266, June 1993.

• William T. Reeves, David H. Salesin, and Robert L. Cook. Rendering antialiased shadows with depth maps. Computer Graphics (SIGGRAPH '87) 21(4): 283-291, July 1987.

Outline

- Motivation
- Non-parametric vs. parametric texture mapping
- Rendering
- Aliasing and antialiasing (Mip Maps)
- Solid textures
- Bump mapping
- Displacement mapping
- Environment mapping
- Transparency
- Depth buffer shadow algorithm

Motivation



Texture mapping (Neider et al,., fig 9.1)

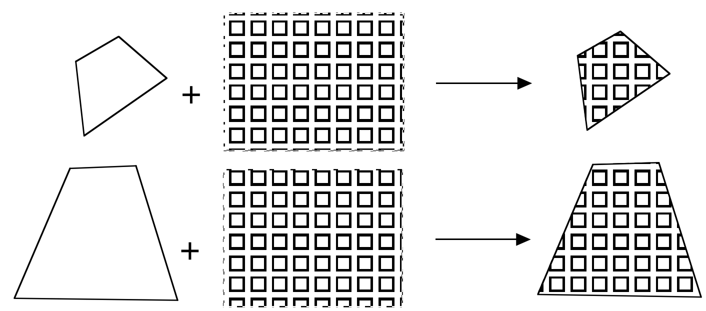
Texture mapping allows you to take a simple polygon and give it the appearance of something much more complex.

- Due to Ed Catmull, PhD thesis, 1974
- Refined by Blinn & Newell, 1976

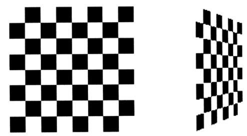
Texture mapping ensures that "all the right things" happen as textured polygon is transformed and rendered.

Non-parametric texture mapping

- Gives cookie-cutter effect
- Texture size and orientation fixed, not tied to the size and orientation of the polygon



Parametric texture mapping



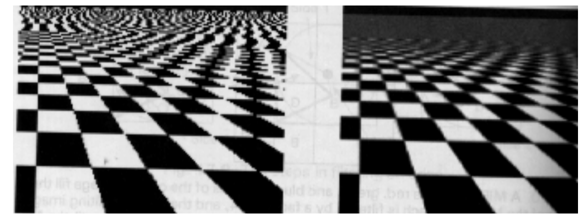
Parametric mapping (Neider et al., fig 9.2)

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 the textured polygon into screen space

Aliasing and antialiasing



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

Proper antialiasing requires area averaging in the texture

Mip Maps

- Lance Williams, '83
- Stands for "multum in parvo" -- many things in a small place.
- Keep textures prefiltered at multiple resolutions
- Figure out closest two levels, and linearly interpolate between the two.
- Example: Neider et al., fig 9-3.
- Other methods: summed area tables, stochastic sampling

Solid textures

Q: What kinds of artifacts might you see from using a wood-grained veneer instead of real wood?

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...

Bump mapping

Textures can be used for more than just color.

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

- The normal is perturbed in each parametric direction according to the partial derivates of the texture:

- These bumps "animate" with the surface

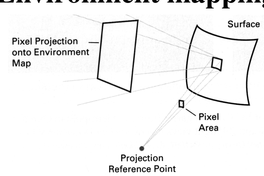
Q: What artifacts in the images would reveal that bump mapping is a fake?

Displacement mapping

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

- Gives rounded edges, even on silhouette
- Requires doing hidden surface after shading

Environment mapping



Hearn and Baker. fig 14.92

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," in which rays bounce around among objects in the scene
- Environment mapping works well when there is just a single object -- or in conjunction with ray tracing

Transparency

- Rays can also be transmitted through a surface, and then refracted into a texture map:

- Kay and Greenberg give a painter's hidden surface algorithm for drawing objects with transparency and approximate refraction.
- Textures can also be used to control the transparency of a surface, itself.
  - Gardner has made nice clouds this way.

Shadow algorithms

Computing shadows is analogous to computing hidden surfaces:

- Surfaces in shadow are exactly those surfaces hidden from the light!
- Thus, for every hidden surface algorithm, there is a corresponding shadow algorithm
- We'll just look at one...

Depth buffer shadow algorithm

Based on the Z-buffer hidden surface algorithm.

Rendering is done in two passes:

First pass:

- Compute scene from point of view of each light
- For each light, save only closest z in a "depth map":

Depth buffer shadow algorithm, cont.

Second pass:

- Render scene from camera
- For each point on surface:
  - Transform point coordinate system of each ligh
  - Compare transformed z against depth map z
  - Surface is in shadow if transformed z is further away
  - Light source contribution is 0 for surfaces in shadow
  - Need to use a slight bias ε so that surface does not shadow itself

Depth buffer shadow algorithm, cont.

Antialiasing:

- Vanilla algorithm has considerable aliasing
- Problem can be alleviated by taking several samples over each pixel
- Transform each sample and compare against depth map
- Can get softer-looking shadows by artificially spreading out the samples
- Idea originally tested in Luxo Jr.
- Recently implemented in hardware on SGI machines