Texture Mapping

Brian Curless
CSE 557
Autumn 2017

Reading

Optional

- Angel and Shreiner: 7.4-7.10
- Marschner and Shirley: 11.1-11.2.3, 11.2.5, 11.4-11.5

Further reading

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

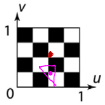
Texture mapping allows you to take a simple polygon and give it the appearance of something much more complex.

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

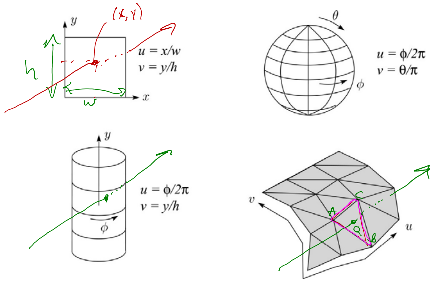
A texture can modulate just about any parameter – diffuse color, specular color, specular exponent, …

Implementing texture mapping

A texture lives in it own abstract image coordinates paramaterized by (u, v) in the range $([0..1], [0..1])$:



It can be wrapped around many different surfaces:



$$Q = \alpha \overset{(u_A, v_A)}{A} + \beta \overset{(u_B, v_B)}{B} + \gamma \overset{(u_C, v_C)}{C}$$

$$(u_Q, v_Q) =$$

With a ray caster, we can do the sphere and cylinder mappings directly (as we will see later). For graphics hardware, everything gets converted to a triangle mesh with associated (u, v) coordinates.

Note: if the surface moves/deforms, the texture goes with it.

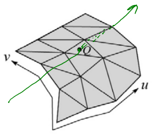
Mapping to texture image coords

The texture is usually stored as an image. Thus, we need to convert from abstract texture coordinate:

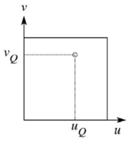
(u, v) in the range $([0..1], [0..1])$

to texture image coordinates:

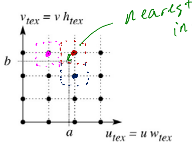
(u_{tex}, v_{tex}) in the range $([0..\ w_{tex}\ ], [0..\ h_{tex}\ ])$



Point on triangle mesh



Mapping to abstract texture coords



Mapping to texture pixel coords

nearest neighbor interpolation

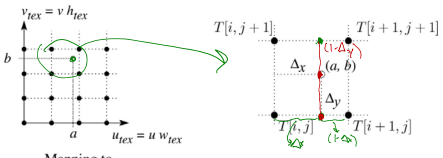
ick!

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

Texture resampling

We need to resample the texture:

$\Delta x, \Delta y \in [0,1]$



Mapping to texture pixel coords

Close-up

Thus, we seek to solve for: $\mathrm{T}(a,b) = \mathrm{T}\left(i+\Delta_x, j+\Delta_y\right)$

A common choice is **bilinear interpolation**:

$$\mathrm{T}\left(i+\Delta_x, j\right) = \underline{(1-\Delta_x)}\ \mathrm{T}[i,j] \quad + \quad \underline{\Delta_x}\ \mathrm{T}[i+1,j]$$

$$\mathrm{T}\left(i+\Delta_x, j+1\right) = \underline{(1-\Delta_x)}\ \mathrm{T}[i,j+1] \quad + \quad \underline{\Delta_x}\ \mathrm{T}[i+1,j+1]$$

$$\mathrm{T}\left(i+\Delta_x, j+\Delta_y\right) = \underline{(1-\Delta_y)}\ \mathrm{T}\left(i+\Delta_x, j\right) \quad + \quad \underline{\Delta_y}\ \mathrm{T}\left(i+\Delta_x, j+1\right)$$

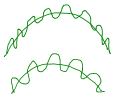
$$= \underline{(1-\Delta_x)(1-\Delta_y)}\ \mathrm{T}[i,j] \quad + \quad \underline{\Delta_x(1-\Delta_y)}\ \mathrm{T}[i+1,j] \quad +$$

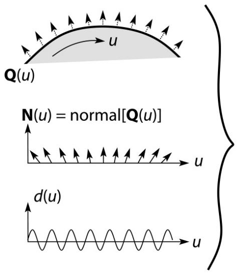
$$\underline{(1-\Delta_x)\Delta_y}\ \mathrm{T}[i,j+1] \quad + \quad \underline{\Delta_x \Delta_y}\ \mathrm{T}[i+1,j+1]$$

Displacement mapping

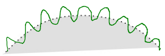
Textures can be used for more than just color.

In **displacement mapping**, a texture is used to perturb the surface geometry itself. Here's the idea in 2D:





$\tilde{\mathbf{Q}}(u) = \mathbf{Q}(u) + d(u)\mathbf{N}(u)$



- These displacements "animate" with the surface
- In 3D, you would of course have (u, v) parameters instead of just u.

Suppose $\mathbf{Q}$ is a simple surface, like a cube. Will it take more work to render the modified surface $\tilde{\mathbf{Q}}$?

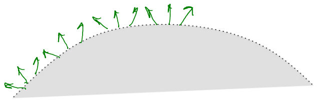
Yes

Bump and normal 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:

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



$\mathbf{Q}(u)$

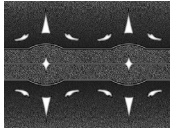
An alternative to compute the normals from the original bump map height field and map them over the smooth surface. This is called **normal mapping**.

What artifacts in the images would reveal that bump (or normal) mapping is fake?

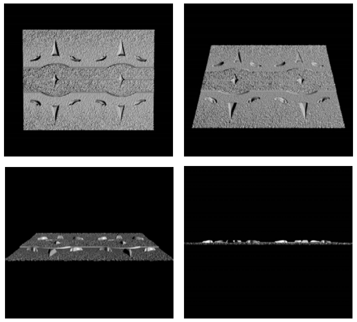
shadows
reflections
silhouettes

Displacement vs. bump mapping

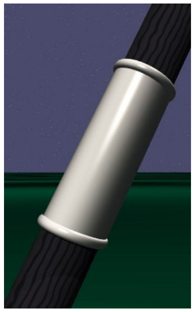
Input texture



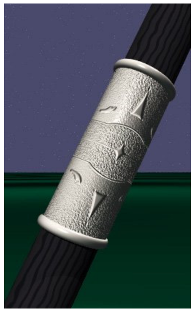
Rendered as displacement map over a rectangular surface



Displacement vs. bump mapping (cont'd)



Original rendering

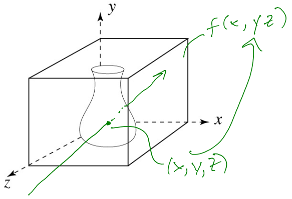


Rendering with bump map wrapped around a cylinder

Bump map and rendering by Wyvern Aldinger

Solid textures

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



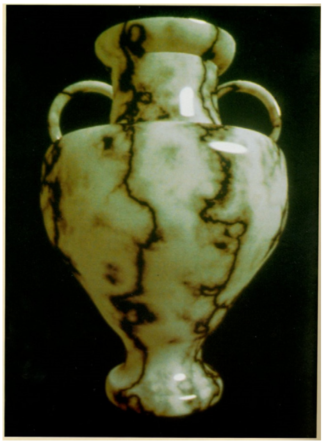
One solution is to use **solid textures**:

- 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:



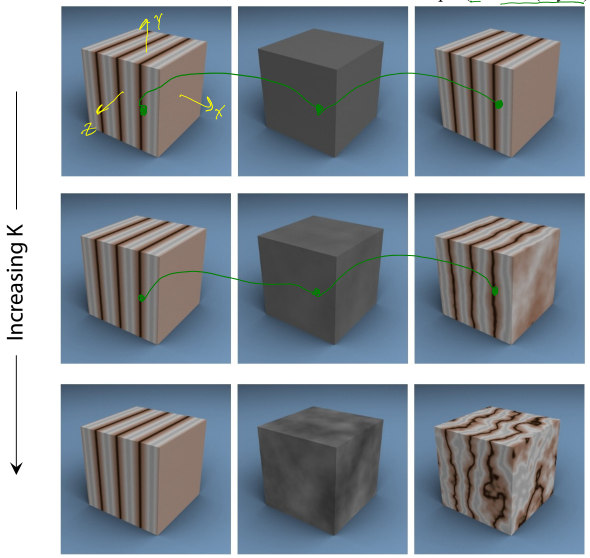
Solid marble texture by Ken Perlin, (Foley, IV-21)

Solid textures (cont'd)

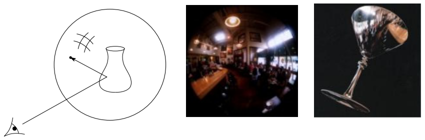
$\text{in}(x, y, z) = \text{stripes}(x)$

$\text{shift}(x, y, z) = K \bullet \text{noise}(x, y, z)$

$\underline{\text{out}}(x ,y, z) = \text{stripes}(\underline{x} + \underline{\text{shift}(x, y, z)})$



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
- Environment mapping works well when there is just a single object – or in conjunction with ray tracing

This can be readily implemented (without interreflection) in graphics hardware using a fragment shader, where the texture is stored in a "cube map" instead of a sphere.

With a ray tracer, the concept is easily extended to handle refraction as well as reflection (and interreflection).

Summary

What to take home from this lecture:

1. The meaning of the boldfaced terms.
2. Familiarity with the various kinds of texture mapping, including their strengths and limitations.