Shading 4:
Texture tricks

Christian Miller
CS 354 - Fall 2011
Regular Texturing

- Texture replaces the diffuse component in the lighting equation
- Allows for more color detail on the surface
More complex effects

• A texture can be used to represent anything that changes over a surface

• Clever use of texcoords can get an even wider range of effects, including view-dependent tricks

• Textures can even be lookup tables
Specular mapping

- Use a greyscale texture as a multiplier for the specular component
Specular mapping
Alpha mapping

- Represent the alpha channel with a texture
- Can give complex outlines, used for plants
**Environment mapping**

- Simulate specular reflections by storing an object’s surroundings in a texture
- When rendering the object, compute texcoords to look up the reflection in the texture
- Assumes the object is a point and that reflected objects are far away
- No self-reflections
Sphere mapping

- Use a single warped image as the texture
- Use warped reflection vector as texcoords
Sphere mapping

- Compute reflection vector in eye coords
- Apply a distortion to generate texcoords

\[ r = 2(u \cdot n)n - u \]

\[ m = 2\sqrt{r_x^2 + r_y^2 + (r_z + 1)^2} \]

\[ s = r_x/m + 0.5 \]
\[ t = r_y/m + 0.5 \]
Sphere mapping

- Sphere map is rather distorted, loses detail at edges
- Reflects an unchanging hemisphere, difficult to update at runtime
- Uses just one texture
- Fine for cheap shiny effects
Cube mapping

- Use six textures, one for each face of a cube that surrounds the object
- Graphics hardware allows you to use reflection vector directly as texture coordinates
Cube mapping

- Requires storing 6 textures
- No strange warping effects
- View independent (covers all view directions)
- Relatively easy to generate dynamically
- Used all the time in games
Cube mapping
Light mapping

- Good shadows are complicated and expensive
- If the lighting and objects aren’t going to change, neither are the shadows
- Can “bake” the shadows into a texture map as a preprocess step
- During shading, lightmap values are multiplied into resulting pixel
Light mapping

DIFFUSE \times LIGHTMAP = DIFFUSE \times LIGHTMAP
**Light map size**

- Diffuse textures usually repeat over big surfaces
- Light maps never repeat, which means there’s much more texture to store
- Usually use a single channel and much lower resolution than diffuse textures
**Irradiance mapping**

- You can reuse environment maps for diffuse reflections.
- Integrate the map over a hemisphere at each pixel (basically blurs the whole thing out).
IRRADIANCE MAPPING
• Bump mapping increases surface detail by modifying normals, which then interact with the lighting model

• Does not alter geometry!
We can set up a coordinate system at every pixel using the normal, tangent, and binormal.

The last two are derived from texture coordinates.
**Heightfield**

- A greyscale image that represents the height at a particular point on the surface
- Can then be used to derive the surface normals
The usual way to do this effect is to bypass the heightmap and store normals directly in the texture.

Coordinates of normal (relative to tangent space) are encoded in color channels.
NORMAL MAPPING
Normal mapping

- Very useful for making low-resolution geometry look like it’s much more detailed
Normal mapping

- Doesn’t help you with edge detail
Parallax mapping

- Normal maps increase lighting detail, but they lack a sense of depth when you get up close
- Parallax mapping uses a heightmap to offset the texture value / normal lookup
- Pretty cheap, looks good, but fails on steep heights
• Implement a heightfield raytracer in a shader
• Pretty expensive, but looks amazing
Normal + environment mapping combined
Intermediate buffers

• You can render part of the scene into an image, then use that as a texture later

• For example, generating cube map faces on the fly

• Also very useful for other effects...
Water effects

• Render everything below water into one buffer
• Render everything above the water (flipped) into another buffer
• Use those buffers as textures when rendering
• Use per-vertex reflection / refraction to perturb texture coordinates and create ripple effects
• Textures can be used as precomputation tables for fancy lighting models, like factorized BRDFs
It’s a big iceberg

• Realtime graphics is all about finding fast, clever ways to utilize textures and shaders that get good looking results

• Most modern games use all of these techniques and much more