GPU Programming:
Environment Mapping
Bump Mapping
Displacement Mapping
Shadow Mapping

Fall 2017

Daniel G. Aliaga
Department of Computer Science
Purdue University
Environment Mapping (or Reflection Mapping)


• The Abyss

• Terminator II
Environment Mapping

• Approximation
  – if the object is small compared to the distance to the environment, the illumination on the surface only depends on the direction of the reflected ray, *not* on the point position on the object

• Algorithm
  – pre-compute the incoming illumination and store it in a texture map
Environment Mapping

Eye → Object

E-Map

N

V

R=?
Environment Mapping

\[ R = V - 2(N \cdot V)N \]
Environment Maps Forms

• Spherical Mapping
• Cubical Mapping (or Cube Map)
• Paraboloidal Mapping
Spherical Mapping
Spherical Mapping

Matt Loper, MERL
Spherical Mapping

Matt Loper, MERL
Spherical Mapping: Renderings
Cubical Mapping
Cubical Mapping: Renderings
Bump Mapping


- Simulates small surface variations
- Key idea: tweak normals used for lighting (geometry stays the same)
- Benefit: much more efficient, geometry-wise, than creating an approximation using very small triangles
Bump Mapping

- Each texel stores two offsets (in u and in v)
Bump Mapping Demo

Normal Map (used for Bump Mapping)

• Use texel values to modify vertex/pixel normals of polygon
• Texel values correspond to normals (or heights) modifying the current normals
• \( \text{RGB} = \frac{(n+1)}{2} \)
• \( n = 2*\text{RGB} – 1 \)
Bump Mapping

• The light source direction $L$ and pixel normal $N$ are represented in the global coord $x, y, z$

• The bump map normal $n$ is in its local coordinates, which is called tangent space or texture space
  
  – $T$: tangent vector
  
  – $N$: surface normal
  
  – $B$: bitangent
  
  – How to compute TNB?
Displacement Mapping

• Bump mapping
  – can be at pixel level
  – has no geometry/shape change

• Displacement Mapping
  – Actually modify the surface geometry (vertices)
  – re-calculate the normals
  – Can include bump mapping
Displacement Mapping

- Bump mapped normals are inconsistent with actual geometry. No shadow.
- Displacement mapping affects the surface geometry
  - Texture stores “offset along the normal”
Light Mapping

• Pre-render special lighting effects
• Multi-texturing idea: arbitrary texel-by-texel shading calc’d from multiple texture maps

Reflectance Texture $\times$ Light Map (Illumination Texture) = Display texture
Shadow Mapping

• Render scene from light’s point of view
  – Store depth of each pixel
Shadow Mapping

- Render scene from light’s point of view
  - Store depth of each pixel
  - From light’s point of view, any pixel blocked is in the shadow.
- When shading a surface:
  - Transform surface pixel into light coordinates
  - Compare current surface depth to stored depth. If depth > stored depth, the pixel is in shadow; otherwise pixel is lit
  - Note: can be very expensive timewise...
Aliasing Problem:

single shadow map pixel

What can be done?
Sampling Problem:
Non-linear Mapping

• Linear interpolation of texture coordinates picks up the wrong texture pixels
  – Solution?
  – One option: use small polygons
Example
Interpolation Problem:
Solution: Perspectively Correct Interpolation

• Normally:

\[ u = (1 - \alpha)u_0 + \alpha u_1 \]

Instead:

\[ u = \left[ \frac{(1 - \alpha)u_0}{z_0} + \frac{\alpha u_1}{z_1} \right] / \left[ \frac{(1 - \alpha)}{z_0} + \frac{\alpha}{z_1} \right] \]