BRDFs II & Texture Mapping
Last Time

• Point light models
  – Isotropic lights (1/r² fall-off)
  – Directional lights ("point lights that are infinitely far")
  – Spotlights
Last Time

• Point light models
  – Isotropic lights ($1/r^2$ fall-off)
  – Directional lights (“point lights that are infinitely far”)
  – Spotlights

• The BRDF
  – Bidirectional Reflectance Distribution function $f_r(\mathbf{v}, \mathbf{l})$
    • Which fraction of light incident from direction $\mathbf{l}$ is reflected towards direction $\mathbf{v}$?
  – Incident intensity depends on cosine with surface normal
  – Diffuse BRDF $= \text{constant}$ (“diffuse color”)
    • Surface looks the same from every direction
Today

• The Phong Model
  – A hacky but common specular reflection model

• Introduction to microfacet theory
  – How BRDFs are modeled

• Texture mapping
  – How to make surfaces look more interesting?
  – Informally: Let’s paste an image on the object
  – Formally: Spatial variation of reflectance (BRDF) parameters
Ideal Specular Reflectance

- Reflection is only at mirror angle
- View dependent
  - Microscopic surface elements are usually oriented in the same direction as the surface itself.
  - Examples: mirrors, highly polished metals.
Ideal Specular BRDF

• Light *only* reflects to the mirror direction
• The BRDF is a “Dirac delta function”
  – To do this formally we’d need solid angles and integrals

  \[ I_{in} f_r(\nu, l) = \begin{cases} 
  I_{in} & \text{when } \nu = \text{reflect}(l) \\
  0 & \text{otherwise}
  \end{cases} \]

  light reflected towards \( \nu \)

• **Not useful for point lights**, only for reflections of other surfaces
  – Why? You can’t really see an ideal mirror reflection of an infinitely small light!
Non-ideal Reflectors

• Real glossy materials usually deviate significantly from ideal mirror reflectors
  – Highlight is blurry
• They are not ideal diffuse surfaces either …
Non-ideal Reflectors

• Simple Empirical Reasoning for Glossy Materials
  – We expect most of the reflected light to travel in the direction of the ideal mirror ray.
  – However, because of microscopic surface variations we might expect some of the light to be reflected just slightly offset from the ideal reflected ray.
  – As we move farther and farther, in the angular sense, from the reflected ray, we expect to see less light reflected.
The Phong Specular Model

• How much light is reflected?
  – Depends on the angle $\alpha$ between the ideal reflection direction $\mathbf{r}$ and the viewer direction $\mathbf{v}$. 
The Phong Specular Model

\[ L_o = k_s (\cos \alpha)^q \frac{L_i}{r^2} = k_s (\mathbf{v} \cdot \mathbf{r})^q \frac{L_i}{r^2} \]

- **Parameters**
  - \( k_s \): specular reflection coefficient
  - \( q \): specular reflection exponent
The Phong Model

- Effect of the $q$ coefficient

\[ \alpha = -\pi/2 \quad \alpha = 0 \quad \alpha = \pi/2 \]
Terminology: Specular Lobe

- The specular reflection distribution is usually called a “lobe”
  - For Phong, its shape is $(r \cdot v)^q$
  - The larger $q$, the narrower the lobe
Recap: How to Get Mirror Direction

• Reflection angle = view angle
  – Normal component is negated
  – Remember particle collisions?

• \( \mathbf{R} = \mathbf{V} - 2 (\mathbf{V} \cdot \mathbf{N}) \mathbf{N} \)
The Complete Phong Model

- Sum of three components:
  - ideal diffuse reflection +
  - specular reflection +
  - “ambient”.

Surface
Ambient Illumination

• Represents all indirect illumination
  – This is a total hack!

• Avoids the complexity of really computing indirect (“global”) illumination

• A much better ambient term: make it depend on surface normal
  – Surfaces pointing “up” receive more ambient illumination
• Phong Illumination Model (for 1/r^2 point light)

\[ L_o = \left[ k_a + k_d (n \cdot l) + k_s (v \cdot r)^q \right] \frac{L_i}{r^2} \]

ambient  diffuse  specular

<table>
<thead>
<tr>
<th>Phong</th>
<th>( \phi )</th>
<th>( \rho_{\text{ambient}} )</th>
<th>( \rho_{\text{diffuse}} )</th>
<th>( \rho_{\text{specular}} )</th>
<th>( \rho_{\text{total}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi = 60^\circ )</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
<td></td>
</tr>
<tr>
<td>( \phi = 25^\circ )</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td></td>
</tr>
<tr>
<td>( \phi = 6^\circ )</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
<td></td>
</tr>
</tbody>
</table>
Putting It All Together

• Phong Illumination Model

\[ L_o = \left[ k_a + k_d (n \cdot l) + k_s (v \cdot r)^q \right] \frac{L_i}{r^2} \]

- ambient
diffuse
specular

• Is it physically based?
  – No, does not even conserve energy, may well reflect more energy than what goes in
  – Furthermore, it doesn’t even conform to the BRDF model directly (we are taking the proper cosine for diffuse, but not for specular)
  – And ambient was a total hack
Phong Examples

- The spheres illustrate specular reflections as the direction of the light source and the exponent $q$ (amount of shininess) is varied.

$$L_o = \left[ k_a + k_d (n \cdot l) + k_s (v \cdot r)^q \right] \frac{L_i}{r^2}$$
Phong Normal Interpolation

- Interpolate the average vertex normals across the face and use this in shading computations
  - Again, use barycentric interpolation!

Must be renormalized
Questions?
Blinn-Torrance Variation of Phong

- Uses the “halfway vector” \( h \) between \( l \) and \( v \).

\[
L_o = k_s \cos(\beta) q \frac{L_i}{r^2} = k_s (n \cdot h)^q \frac{L_i}{r^2}
\]

Motivation: When normal coincides with halfway vector, \( v \) and \( l \) are at mirror angles.
Lobe Comparison

- **Half vector lobe**
  - Gradually narrower when approaching grazing angle
- **Mirror lobe**
  - Always circular
Half Vector Lobe is Better

- More consistent with what is observed in measurements (*Ngan, Matusik, Durand 2005*)
Microfacet Theory

• What determines how the surface reflects light?

• It’s the microstructure
  – Diffuse objects are very rough at some fine scale
  – Mirrors are very, very smooth
  – Most materials are somewhere in between

• We can model the microstructure and come up with parametric BRDF models that way
Microfacet Theory

• Example
  – Think of water surface as lots of tiny mirrors (microfacets)
  – “Bright” pixels are...
    • Microfacets aligned with the vector between sun and eye
    • But not the ones in shadow
    • And not the ones that are occluded
Microfacet Theory

- Model surface by tiny mirrors
  [Torrance & Sparrow 1967]
Microfacet Theory

- Value of BRDF at (L, V) is a product of
  - number of mirrors oriented halfway between L and V
Microfacet Theory

- Value of BRDF at \((L,V)\) is a product of
  - number of mirrors oriented halfway between \(L\) and \(V\)
Microfacet Theory

- Value of BRDF at (L,V) is a product of
  - number of mirrors oriented halfway between L and V
Microfacet Theory

- Value of BRDF at \((L, V)\) is a product of
  - number of mirrors oriented halfway between \(L\) and \(V\)
  - ratio of the un(shadowed/masked) mirrors
Microfacet Theory

- Value of BRDF at (L,V) is a product of
  - number of mirrors oriented halfway between L and V
  - ratio of the un(shadowed/masked) mirrors
  - Fresnel coefficient
Microfacet-based Models

- Develop BRDF models by imposing simplifications [Torrance-Sparrow 67], [Blinn 77], [Cook-Torrance 81], [Ashikhmin et al. 2000]

- Model the distribution $p(H)$ of microfacet normals
  - Also, statistical models for shadows and masking
Questions?

T. Weyrich et al., Fabricating Microgeometry for Custom Surface Reflectance, SIGGRAPH 2009
BRDF Examples from Ngan et al.

Material – Dark blue paint
Dark blue paint

Finding the BRDF model parameters that best reproduce the real material

Material – Dark blue paint
Finding the BRDF model parameters that best reproduce the real material

**Material – Dark blue paint**
Observations

• Some materials impossible to represent with a single lobe

Material – Red Christmas Ball

Acquired data

Cook-Torrance
Adding a second lobe

- Some materials impossible to represent with a single lobe

![Acquired data](image1)

![Cook-Torrance 2 lobes](image2)

Material – Red Christmas Ball
Image-Based Acquisition

• See W. Matusik et al. for how the original data was captured
  – A Data-Driven Reflectance Model, SIGGRAPH 2003
  – The data is available from MERL if you ask nicely
Questions?
Spatial Variation

• All materials seen so far are the same everywhere
  – In other words, we’re assuming the BRDF is independent of the surface point \( x \)
  – No real reason to make that assumption
Spatial Variation

• “Texture mapping”
  – We’ll allow the BRDF parameters to vary over space
    • This’ll give us much more complex surface appearance
  – Or, in terms of BRDFs, we are making the diffuse color $k_d$
    vary with $x$ (can also vary other parameters)
Effect of Textures

Model

Model + Shading

Model + Shading + Textures

For more info on the computer artwork of Jeremy Birn see http://www.3drender.com/jbirn/productions.html
Texture Mapping
Texture Mapping

3D model → Texture mapped model

Image: Praun et al.
Texture Mapping

Texture mapped model

We need a function that associates each surface point with a 2D coordinate in the texture map

Texture map (2D image)
Texture Mapping

Texture mapped model

For each point rendered, look up color in texture map

Texture map (2D image)
UV Coordinates

• Each vertex $P$ has associated 2D $(u, v)$ “texture coordinates”
  – UVs determine the 2D location in the texture for the vertex
• Then we interpolate using barycentrics

\[
\begin{align*}
(u_0, v_0) & \quad (\alpha u_0 + \beta u_1 + \gamma u_2, \\
(u_1, v_1) & \quad (\alpha v_0 + \beta v_1 + \gamma v_2) \quad (u_2, v_2)
\end{align*}
\]
UV Coordinates

• Each vertex P has associated 2D (u, v) “texture coordinates”
  – UVs determine the 2D location in the texture for the vertex
• Then we interpolate using barycentrics
Creating Torso Portion in Max

3D Model

UV Mapping
Pseudocode – Ray Casting

• Ray cast pixel \((x, y)\), get visible point and \(\alpha, \beta, \gamma\)
• Get texture coordinates \((u, v)\) at that point
  – Interpolate from vertices using barycentrics
• Look up texture color using UV coordinates
Pseudocode – Rasterization

For every triangle
    ComputeProjection
    Compute interpolation matrix (last time)
    Compute bbox, clip bbox to screen limits
    For all pixels x,y in bbox
        Test edge functions
        If all \( E_i > 0 \)
            compute barycentrics (last time)
            interpolate z from vertices
            if \( z < zbuffer[x,y] \)
                interpolate UV coordinates from vertices
                look up texture color \( k_d \)
                Framebuffer[x,y] = \( k_d \)
Questions?
Texture Tiling

• Specify a texture coordinate \((u,v)\) at each vertex
• Canonical texture coordinates \((0,0) \rightarrow (1,1)\)
  – Wrap around when coordinates are outside \((0,1)\)

Note the range \((0,1)\) unlike normalized screen coordinates!
Problem

• What of non-triangular geometry?
  – Spheres, etc.

• No vertices, can’t specify UVs that way!

• Solution: Parametric Texturing
  – Deduce \((u, v)\) from \((x, y, z)\)
  – Various mappings are possible....
Common Texture Coordinate Mappings

- Planar
  - Vertex UVs and linear interpolation is a special case!
- Cylindrical
- Spherical
- Perspective Projection
Projective Mappings

• A slide projector
  – Analogous to a camera!
  – Usually perspective projection tells us where points project to in our image plane
  – This time we’ll use these coordinates as UVs

• No need to specify texture coordinates explicitly
Projective Texture Example

- Modeling from photographs
- Using input photos as textures

Figure from Debevec, Taylor & Malik
http://www.debevec.org/Research
Questions?
Texture Mapping & Illumination

- Texture mapping can be used to alter some or all of the constants in the illumination equation
  - Diffuse color $k_d$, specular exponent $q$, specular color $k_s$...
  - Any parameter in any BRDF model!

\[
L_o = \left[ k_a + k_d (\mathbf{n} \cdot \mathbf{l}) + k_s (\mathbf{v} \cdot \mathbf{r})^q \right] \frac{L_i}{r^2}
\]
Gloss Mapping Example

Spatially varying $k_d$ and $k_s$
We Can Go Even Further...

- The normal vector is really important in conveying the small-scale surface detail
  - Remember cosine dependence
  - The human eye is really good at picking up shape cues from lighting!

- We can exploit this and look up also the normal vector from a texture map
  - This is called “normal mapping” or “bump mapping”
  - A coarse mesh combined with detailed normal maps can convey the shape very well!
Normal Map Example

Original Mesh
4M triangles
Normal Map Example

Simplified mesh, 500 triangles

Simplified mesh + normal mapping

Paolo Cignoni
Normal Map Example

Diffuse texture $k_d$

Normal Map

Final render

Looks familiar from your ray caster, doesn’t it!
Generating Normal Maps

• Model a detailed mesh
• Generate a UV parametrization for the mesh
  – A UV mapping such that each 3D point has a unique image in the 2D texture map
  – This is a difficult problem, but tools are available
    • E.g., the DirectX SDK has functionality to do this
• Simplify the mesh (again, see DirectX SDK)
• Overlay simplified and original model
• For each point \( P \) on the simplified mesh, find closest point \( P' \) on original model (ray casting)
• Store the normal at \( P' \) in the normal map. Done!
Normal Map Details

• You can store an object-space normal
  – Convenient if you have a unique parameterization
• ....but if you want to use a tiling normal map, this won’t do
  – Must account for the curvature of the object!
  – Think of mapping this diffuse+normal map combination on a cylindrical tower
• Solution: Tangent space normal map
  – Encode a “difference” from the geometric normal in a local coord. system
Questions?
Displacement Mapping

- Encode a displacement distance in the texture map
  - Measured e.g. along interpolated normal
Displacement Mapping

- Encode a displacement distance in the texture map
  - Measured e.g. along interpolated normal
- Input triangles are tessellated within the pipeline and vertices displaced
Displacement Mapping

• Pixar’s RenderMan architecture is heavily based on displacement mapping
  – In pretty much every movie you see today
  – See the original paper by Cook et al., SIGGRAPH 87

• Direct3D 11 has hardware (GPU) functionality to perform tessellation and displacement on the fly!
Questions?
Aliasing

• What happens when you look at a textured object in a steep angle?
• What happens when you look at a textured object in a steep angle?
  – Even though every ray corresponds to a distance of just one pixel, the world-space intervals get larger
  – At some point, we start missing the squares altogether
  – If we just take one point sample at the hit location, we get almost random results!
Aliasing – What Does it Look Like?

Image: Xiaohu Zhang
Aliasing – What Does it Look Like?

Nearest neighbor

“Tri-linear Mip-mapping”
Sampling Texture Maps

- How to map the texture area seen through the pixel window to a single pixel value?
• Seems clear we should be taking some sort of average of the texture under the “pixel footprint”
But That’s A Whole Another Story

- Thursday: Graphics Hardware
  - Guest lecture by Jonathan Ragan-Kelley
- Tuesday: Color
  - Guest lecture by professor William Freeman
- Next Thursday: Thanksgiving