Lecture 14: Shading

Rui Wang
Rendering: 70’s (Local Illumination)

Gouraud

Phong

Shadow

Planar Reflection
Rendering: 80’s – (Global Illumination)

Soft shadows, complex reflection, environment lighting ...
Assumptions

☐ Local illumination
   - Every point is shaded independently, as if no other objects exist in the scene. Hence:
     ☐ No interreflections
     ☐ No shadows

☐ One Single Point Light
☐ Simple material (diffuse + specular)
Shading Geometry

P: Shading Point
L: Light Vector (from P to the light)
V: View Vector (from P to the eye)
N: Surface Normal

All vectors are assumed to be normalized
Surface Reflection Types

- Mirror (Perfectly Specular)

Photograph by Senthil Singaravelu 2004
Surface Reflection Types

- Diffuse (Lambertian): matte paint, paper, fabrics...

http://www.astrofilms.net
Surface Reflection Types

- Glossy / Shiny

Frank Gehry Architecture, Ragheb ed. 2001
Diffuse / Lambertian Shading

- Rough/Dull matte surfaces
  - matte paint, unfinished wood, paper, ...
- Obeys Lambert’s cosine law
- Think of the sunlight during a day.
Diffuse / Lambertian Shading

- Rough/Dull matte surfaces
  - matte paint, unfinished wood, paper, ...
- Obeys Lambert’s cosine law
- Independent of the view vector: color remains the same from all angles.

\[ B \sim \cos(\theta) = \vec{N} \cdot \vec{L} \]
Diffuse / Lambertian Shading

- Rough/Dull matte surfaces
  - matte paint, unfinished wood, paper, ...
- Obeys Lambert’s cosine law
- Independent of the view vector: color remains the same from all angles.

\[
B \sim \cos(\theta) = \vec{N} \cdot \vec{L}
\]

\[
B = I_d \cdot k_d \cdot (\vec{N} \cdot \vec{L})
\]
Diffuse Shading – Complete Formula

To avoid potentially negative values (against physics)
- Take the max of the dot product and 0.

\[ B = I_d \cdot k_d \cdot \max(\vec{N} \cdot \vec{L}, 0) \]
Perfectly Specular Surfaces

- Mirror Reflection
- How to compute mirror reflection vector?
Perfectly Specular Surfaces

Mirror Reflection

How to compute mirror reflection vector?

\[ \vec{R} = 2(\vec{L} \cdot \vec{N})\vec{N} - \vec{L} \]
Specular Shading

- Find a simple way to create highlights that happen at about the right place
  - Gloss paint, plastics, whiteboard...
- View dependent: color changes depending on the viewing angle
Specular Shading

- Imagine that the surface is quite glossy, so the strongest reflection is along the mirror reflection direction \( \mathbf{R} \).
  - If the viewer is looking along \( \mathbf{R} \), he sees the strongest reflection.

\( \mathbf{R} \) is the mirror reflection vector.
Specular Shading

- Imagine that the surface is quite glossy, so the strongest reflection is along the mirror reflection direction $\mathbf{R}$.
- If the viewer is looking along $\mathbf{R}$, he sees the strongest reflection.

$\mathbf{R}$ is the mirror reflection vector.

$$\mathbf{R} = 2(\mathbf{L} \cdot \mathbf{N}) \mathbf{N} - \mathbf{L}$$
Specular Shading

- But what if the viewer is not looking along R?
  - He observes reduced reflection.
Specular Shading

- But what if the viewer is not looking along R?
  - He observes reduced reflection.
- Let’s take the cosine between R and V as a attenuation factor.

\[ B \sim \cos(\alpha) = (\vec{R} \cdot \vec{V}) \]
Specular Shading

To control specularity (shininess), let’s take a power of the cosine term.

\[ B \sim \cos^p(\alpha) = (\vec{R} \cdot \vec{V})^p \]
Specular Shading

\[ p=1 \]

\[ p=5 \]

\[ p=20 \]
Specular Shading

- Changing shininess (p):
Specular Shading

☐ Changing shininess (p)
Specular Shading – Complete Formula

Known as the Phong specular formula:

\[ B = I_s \cdot k_s \cdot (\max(0, \vec{R} \cdot \vec{V}))^p \]
Specular Shading Facts:

- It’s a simple way to create highlights that happen at about the right place
- A physically motivated hack!
  - Not physically based
  - Does not even conserve energy
Emission

\[ B = I_e \]
Ambient Shading Color

☐ Our shading formula so far can produce complete darkness.

☐ In the real world, we don’t typically see objects that are completely dark.
Ambient Shading Color

- Consider there is some global lighting that comes from all directions in space.
  - Simulates indirectly reflected lights
  - Constant color per object
  - Again, this is a hack.

\[ B^+ = I_a \cdot k_a \]
Putting Everything Together

- Surface shading can be a mixture of all the mentioned shading formulas.
- To shade, simply sum them up together
- One Light

\[ B = I_e + I_a \cdot k_a + I_d \cdot k_d \cdot \max(\vec{N} \cdot \vec{L}, 0) + I_s \cdot k_s \cdot (\max(0, \vec{R}_L \cdot \vec{V}))^p \]

emission  ambient  diffuse  specular
Illumination is Linear

- What about many light sources?
Putting Everything Together

- When you have multiple lights, simply sum the contribution from each light (except the emission and ambient terms).

\[
B = I_e + I_a \cdot k_a + \sum_j \left[ I_d^j \cdot k_d \cdot (\vec{N} \cdot \vec{L}_j) + I_s^j \cdot k_s \cdot (\max(0, \vec{R}_L \cdot \vec{V}))^e \right]
\]
More on Shading

- **Gouraud Shading**
  - Compute shading color per vertex, then interpolate color for pixels.
  - Standard graphics pipeline.

- **Phong Shading**
  - Compute shading at per-pixel level.
  - Involves normal interpolation.
  - Need programmable shading to achieve this.

- **Gouraud Shading and Phong Shading**
  [comparison](#)