Chapter 6

Lighting

Goal
- Model the interaction of light with surfaces to render realistic images

Contributing Factors
- Light sources
  - Shape and color
- Surface materials
  - How surfaces reflect light
- Transport of light
  - How light moves in a scene

Appearance depends on
- Light sources, locations, properties
- Material (surface) properties
- Viewer position

Algorithms:
- Local illumination - Fast
  - "Fake" - Ignore real physics, approximate the look
  - Compute at material, from light to viewer
- Global illumination – Slow
  - Physically based

Rendering Pipeline

Local vs. Global Illumination Model

Local model - interaction of each object with light

Global model: interactions between objects

Materials

Surface reflectance:
- Illuminate surface point with a ray of light from different directions
- How much light is reflected in each direction?
**Computer Graphics**

**Illumination Models**

- **Basic Types**
  - Diffuse Material
  - Glossy Material
  - Specular Material

- **Reflectance Distribution Model**
  - Most surfaces exhibit complex reflectances
  - Vary with incident and reflected directions.
  - Model with combination – known as BRDF
  - BRDF: Bidirectional Reflectance Distribution Function

- **BRDF measurements/plots**
  - 2D slice
  - 3D surface

---

Copyright 2011, Alla Sheffer, UBC
**Materials**

- Practical Considerations
  - In practice, often simplify (computational efficiency)
  - Derive specific formulas that describe basic reflectance behaviors
    - diffuse, glossy, specular
    - OpenGL choice

**Computing Diffuse Reflection**

- Depends on angle of incidence: angle between surface normal and incoming light
  - \( I_{\text{diffuse}} = k_d \, I_{\text{light}} \cos \theta \)
  - In practice use vector arithmetic
  - Always normalize vectors used in lighting
    - \( \mathbf{n}, \mathbf{l} \) should be unit vectors
  - Scalar (B/W intensity) or 3-tuple or 4-tuple (color)
    - \( k_d \): diffuse coefficient, surface color
    - \( I_{\text{light}} \): incoming light intensity
    - \( I_{\text{diffuse}} \): outgoing light intensity (for diffuse reflection)

**Physics of Diffuse Reflection**

- Ideal diffuse reflection
  - Very rough surface at the microscopic level
    - Real-world example: chalk
    - Microscopic variations mean incoming ray of light equally likely to be reflected in any direction over the hemisphere
  - Reflected intensity only depends on light direction!

**Diffuse Lighting Examples**

- Lambertian sphere from several lighting angles:
  - need only consider angles from 0° to 90°

**Lambert’s “Law”**

Intuitively: cross-sectional area of the “beam” intersecting an element of surface area is smaller for greater angles with the normal.

**Physics of Specular Reflection**

- Geometry of specular (perfect mirror) reflection
  - Snell’s law

Copyright 2011, Alla Sheffer, UBC
Glossy Reflectance
- Snell’s law applies to perfect mirror-like surfaces, but aside from mirrors (and chrome) few surfaces exhibit perfect specularity
- How can we capture the “softer” reflections of surface that are glossy, not mirror-like?
- One option: model exact physics
  - Multiple reflections on microgeometry level
  - or...

Empirical Approximation
- Expect most reflected light to travel in direction predicted by Snell’s Law
- But (because of microscopic surface variations) some light may be reflected in a direction slightly off the ideal reflected ray
- As angle from ideal reflected ray increases, we expect less light to be reflected

Empirical Approximation
- Angular falloff
  - How might we model this falloff?

Phong Lighting
- Most common lighting model in computer graphics
  - (Phong Bui-Tuong, 1975)
  - \( I_{\text{specular}} = k_s I_{\text{light}} (\cos \phi)^n_s \)
  - \( \phi \): angle between \( \tau \) and view direction \( \nu \)
  - \( n_s \): purely empirical constant, varies rate of falloff
  - \( k_s \): specular coefficient, highlight color
    - no physical basis, works ok in practice

Phong Lighting: The \( n_s \) Term
- Phong reflectance term drops off with divergence of viewing angle from ideal reflected ray
  - Viewing angle – reflected angle

Phong Examples
- varying \( I \)
- varying \( n_{shiny} \)
Calculating Phong Lighting

- Compute cosine term of Phong lighting with vectors

\[ I_{\text{specular}} = k_s I_{\text{light}} (v \cdot r)^n \]

- \( v \): unit vector towards viewer/eye
- \( r \): ideal reflectance direction (unit vector)
- \( k_s \): specular component
- \( I_{\text{light}} \): incoming light intensity

Alternative Model

- Blinn-Phong model (Jim Blinn, 1977)
  - Variation with better physical interpretation
    - \( h \): halfway vector; \( r \): roughness

\[ I_{\text{specular}} = k_s \cdot (h \cdot n)^{1/r} \cdot I_{\text{light}} ; \text{with } h = (1 + v) / 2 \]

Materials (last bit)

- Light is linear
  - If multiple rays illuminate the surface point the result is just the sum of the individual reflections for each ray

Light Sources

- Point source
  - Light originates at a point
  - Rays hit planar surface at different angles
- Parallel source
  - Light rays are parallel
  - Rays hit a planar surface at identical angles
  - May be modeled as point source at infinity
  - Directional light

Light Sources

- Area source
  - Light originates at finite area in space.
  - In-between point and parallel sources
- Spotlights
  - Position, direction, angle
- Ambient light (environment light)
  - Hack for replacing true global illumination
  - (light bouncing off from other objects)

Ambient Light Sources

- Scene lit only with an ambient light source
**Computer Graphics**

### Directional Light Sources
- Scene lit with directional and ambient light

![Directional Light Source Diagram]

**Light Position Important**  
**Surface Angle Important**  
**Viewer Position Not Important**

### Point Light Sources
- Scene lit with ambient and point light source

![Point Light Source Diagram]

**Light Position Important**  
**Viewer Position Important**  
**Surface Angle Important**

### Light Source Falloff
- Quadratic falloff (point- and spot lights)
  - Brightness of objects depends on power per unit area that hits the object
  - The power per unit area for a point or spot light decreases quadratically with distance

![Light Source Falloff Diagram]

**Non-quadratic falloff**
  - Many systems allow for other falloffs
  - Allows for faking effect of area light sources
  - OpenGL / graphics hardware
    - \( I_o \): intensity of light source
    - \( x \): object point
    - \( r \): distance of light from \( x \)
    - \( I_{in}(x) = \frac{1}{ar^2 + br + c} \cdot I_o \)

### Ambient Light
- non-directional light – environment light
- Object illuminated with same light everywhere
  - Looks like silhouette
  - Illumination equation \( I = I_{amb} \)
    - \( I_{amb} \): ambient light intensity
    - \( k_a \): fraction of this light reflected from surface
    - Defines object color

### Illumination Equation
- For multiple light sources:
  \[
  I = I_a k_a + \sum_p \frac{I_p}{A(d_p)}(k_d(a \cdot I_p) + k_v(v \cdot I_p)) + k_a \]
  - \( d_p \): distance between surface and light source
  - \( A \): distance between surface and viewer, \( A \) – attenuation function
**Computer Graphics**

**Illumination Models**

**Light**
- Light has color
- Interacts with object color \((r,g,b)\)
  \[
  I = I_s k_s \\
  I_s = (I_w, I_d, I_s) \quad k_s = (k_w, k_d, k_s) \\
  I = (I_w, I_d, I_s) = (I_w, I_d, I_s) k_w k_d k_s
  \]
- Blue light on white surface?
- Blue light on red surface?

**Lighting in OpenGL**
- Light source: amount of RGB light emitted
  - value represents percentage of full intensity, e.g., \((1.0,0.5,0.5)\)
  - every light source emits ambient, diffuse, and specular light
- Materials: amount of RGB light reflected
  - value represents percentage reflected, e.g., \((0.0,1.0,0.5)\)
- Interaction: multiply components
  - Red light \((1,0,0)\) x green surface \((0,1,0)\) = black \((0,0,0)\)

**Light Sources - OpenGL**
- Specify parameters
  \[
  \text{glLightfv}(GL\_LIGHTi, GL\_POSITION, light[]) \\
  i \text{ -- between 0 & 8 (or more)}
  \]
- Directional \([x \ y \ z \ 0]\)
- Point source \([x \ y \ z \ 1]\)
- Spotlight has extra parameters:
  - GL\_SPOT\_DIRECTION, GL\_SPOT\_EXponent, GL\_SPOT\_CUTOFF
- Area source – too complex for projective pipeline (e.g. OpenGL)

**In OpenGL**
- \(k_w, k_d, k_s\) - surface color (RGB)
- Modify by \(\text{glMaterialfv}(GL\_FRONT\_AND\_BACK, \) \(pname, \text{RGB}[])\)
- \(pname\) - GL\_AMBIENT, GL\_DIFFUSE, GL\_SPECULAR
- Light source properties (also RGB)
  \(\text{glLightfv}(GL\_LIGHTi, pname, light[])\)

**Lighting in Rendering Pipeline**
- Notes:
  - Lighting is applied to every vertex
    - i.e. the three vertices in a triangle
  - Per-vertex lighting
  - Will later see how the interior points of the triangle obtain their color
    - This process is called shading
  - Will discuss in the context of scan conversion

Copyright 2011, Alla Sheffer, UBC