Chapter 6

Lighting

Rendering Pipeline

Geometric Content

Model/View Transform

Lighting

Perspective Transform

Clipping

Scan Conversion

Texturing

Depth Test

Blending

Rasterization

Fragment Processing

Frame-buffer

Lighting/Shading

Goal

- Model the interaction of light with surfaces to render realistic images
- Generate per (pixel/vertex) color

Illumination Models/Algorithms

Local illumination - Fast

- Ignore real physics, approximate the look
- Interaction of each object with light
- Compute on surface (light to viewer)

Global illumination – Slow

- Physically based
- 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?

Copyright 2012, Alla Sheffer, UBC
Specular Material

Diffuse Material

Glossy Material

Basic Types

Reflectance Distribution Model

BRDF measurements/plots

Copyright 2012, Alla Sheffer, UBC
**Computer Graphics**

### 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
  - $I_{diffuse} = k_d I_{light} \cos \theta$
- In practice use vector arithmetic
  - $I_{diffuse} = k_d I_{light} (\mathbf{n} \cdot \mathbf{l})$
- 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
  - $I_{light}$: incoming light intensity
  - $I_{diffuse}$: 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°
    - Why?

### 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

---

**Illumination Models**
**Computer Graphics**

**Illumination Models**

---

**Empirical Approximation**
- Snell’s law = perfect mirror-like surfaces
  - But ..
    - few surfaces exhibit perfect specularity
    - Gaze and reflection directions never EXACTLY coincide
  - Expect most reflected light to travel in direction predicted by Snell’s Law
  - But 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

**Phong Examples**

\[ I_{\text{specular}} = k_s I_{\text{light}} \left( \cos \phi \right)^n \]

**Calculating Phong Lighting**
- How to model this falloff?

**Phong Lightin**
- Most common lighting model in computer graphics
  - (Phong Bui-Tuong, 1975)

\[ I_{\text{specular}} = k_s I_{\text{light}} \left( \cos \phi \right)^n \]

**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

\[ \sum I_p (k_s (n \cdot l_p) + k_r (r_p \cdot v)^n) \]
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
  - Can model as point source at infinity
  - Directional light

- 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

- Non-directional light – environment light
- Object illuminated with same light everywhere
  - Looks like silhouette
- Illumination equation $I = I_a k_r$
  - $I_a$ - ambient light intensity
  - $k_r$ - fraction of this light reflected from surface

Directional Light Sources

- Scene lit only with an ambient light source

Point Light Sources

- Scene lit with ambient and point light source

Copyright 2012, Alla Sheffer, UBC
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

Non-quadratic falloff
- Many systems allow for other falloffs
- Allows for faking effect of area light sources
- OpenGL / graphics hardware
  - \( I \): intensity of light source
  - \( x \): object point
  - \( r \): distance of light from \( x \)
  - \( I_a(x) = \frac{1}{ar^2 + br + c} \cdot I_0 \)

Illumination Equation

For multiple light sources:

\[
I = I_0 k_0 + \sum \frac{I_k}{d_k^2} (k_s (n \cdot l_s) + k_d (r_p \cdot v)^\alpha + k_r (r_p \cdot v)^\beta)
\]

\( d_s \) - distance between surface and light source
\( + \) distance between surface and viewer, \( A \) = attenuation function

Lighting in OpenGL

- Light source: amount of RGB light emitted
  - value = 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)

In OpenGL

- \( k_s, k_d, k_r \) - surface color (RGB)

Modify by `glMaterialfv(GL_FRONT_AND_BACK, pname, RGB[])`

`pname` - GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR

Light source properties (also RGB)
`glLightfv(GL_LIGHTi, pname, light[])`
Lighting in OpenGL

```c
glLightfv(GL_LIGHT0, GL_AMBIENT, amb_light_rgba);
glLightfv(GL_LIGHT0, GL_DIFFUSE, dif_light_rgba);
glLightfv(GL_LIGHT0, GL_SPECULAR, spec_light_rgba);
glLightfv(GL_LIGHT0, GL_POSITION, position);
egEnable(GL_LIGHT0);

glMaterialfv( GL_FRONT, GL_AMBIENT, ambient_rgba );
glMaterialfv( GL_FRONT, GL_DIFFUSE, diffuse_rgba );
glMaterialfv( GL_FRONT, GL_SPECULAR, specular_rgba );
glMaterialfv( GL_FRONT, GL_SHININESS, n );
```

Light Sources - OpenGL

- Specify parameters
  ```c
glLightfv(GL_LIGHTi, GL_POSITION, light[i])
i – 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)

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