Chapter 6

Lighting

Rendering Pipeline

Illumination Models/Algorithms

Lighting/Shading

Materials
Most surfaces exhibit complex reflectances:
- Vary with incident and reflected directions.
- Model with combination – known as BRDF

**BRDF:** Bidirectional Reflectance Distribution Function

**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
### 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

Copyright 2013, Alla Sheffer, UBC
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

Empirical Approximation

Angular falloff

How to model this falloff?

Phong Examples

I_{specular} = k_sI_{light}(\cos \phi)^{n_s}

Calculating Phong Lighting

I_{specular} = k_sI_{light} (v \cdot r)^{n_s}

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_s (r \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

**Light Position**
- Not Important

**Viewer Position**
- Not Important

**Surface Angle**
- Not Important

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

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

**Ambient Light Sources**
- Scene lit only with an ambient light source

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

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

**Ambient Light Sources**
- Scene lit only with an ambient light source

**Directional Light Sources**
- Scene lit with directional and ambient light
**Computer Graphics**

**Illumination Models**

**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](image)

**Light**
- Light has color
- Interacts with object color \((r,g,b)\)

\[
I = I_o \cdot k_a \\
I_s = (I_o \cdot I_{ad} \cdot I_{sh}) \\
k_o = (k_{ao}, k_{ag}, k_{ab}) \\
I = (I_o \cdot I_{sd} \cdot I_{sh}) = (I_o \cdot k_{ao} \cdot I_{ad} \cdot k_{ag} \cdot I_{sh} \cdot k_{ab})
\]
- Blue light on white surface?
- Blue light on red surface?

**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 \((1.0,0.0,0)\) x green surface \((0.1,0,0)\) = black \((0,0,0)\)

**Illumination Equation**
- For multiple light sources:

\[
I = I_o \cdot k_s + \sum_{p} \frac{I_p}{A(d_p)} (k_s(n \cdot l_p) + k_s(r_p \cdot v)^p)
\]

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

**In OpenGL**
- \(k_s, k_d, k_r\) - surface color (RGB)
- Modify by \(\text{glMaterialfv(GL_FRONT_AND_BACK, pname, RGB[])}\)
  - \(pname\) - GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR
- Light source properties (also RGB)
  - \(\text{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);

glEnable(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 – 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