Illumination Models

[electricimage.com]
Local Illumination

Example

Local Illumination in the projective rendering pipeline

Local Illumination
- only models light arriving directly from light source
- interreflections and shadows
  - added through tricks, multiple rendering passes

Types of Models
- Simple, non-physical reflection models (Phong, Blinn)
- physically-based reflection models
  - BRDFs: Bidirectional Reflection Distribution Functions
**Light Sources**

**Types of light sources**
- `glLightfv(GL_LIGHT0, GL_POSITION, light[])`
- Directional/parallel lights
  - *E.g. sun*
  - *Homogeneous vector*
- (Homogeneous) point lights
  - *Same intensity in all directions*
- Spot lights
  - *Limited set of directions:*
    - Point+direction+cutoff angle

**Local Illumination**

**Commonly used model**
*(simple, non-physical)*

**Combine diffuse, specular, ambient**
- *E.g. OpenGL / graphics hardware:*

\[
I_{out}(x) = k_a \cdot I_a + k_d \cdot (l \cdot n) \cdot I_{diff} + k_s \cdot (h \cdot n)^n \cdot I_{spec}
\]

**Materials**

**Ambient Light**
- *Incoming light component that is identical everywhere in the scene*
- *No direction*
- *Hack for replacing true global illumination (light bouncing off from other objects)*

\[
I_{out}(x) = k_a \cdot I_a \quad \text{for} \quad I_a \in [0, 1]
\]
Diffuse Component: Lambert’s Law

Johann Friedrich Lambert (1783):
- Power per unit area arriving at some object point \(x\) also depends on the angle of the surface to the light direction
  - \(dA\): differential surface area surrounding \(x\)
  - \(I\): light direction (unit length!)

\[
I'(x) = \cos(\angle(n, I)) \cdot I_{in}(x) = (n \cdot I) \cdot I_{in}(x)
\]

Materials

Specular/Glossy
- Light is mostly reflected into the directions around the mirror direction \(r_i\) of \(I\)

\[
I_{spec} = k_f \cdot (\cos \alpha)^n
\]

Specular/Glossy
- Computing reflection direction \(r_i\) of \(I\)
  - \(n\) and \(I\) are unit length!

\[
|I| = 1 \\
|N| = 1 \\
|R| = 1 \\
(n \cdot I) \cdot n = 2(n \cdot I) \cdot n - I
\]
**Materials**

**Phong Model (Phong Bui-Tuong, 1975)**
- Use cosine power as heuristic
  \[ I_{\text{spec}}(x) = k_s \cdot (v \cdot r)^n \cdot I_{\text{in}}(x) \]
- For highly specular surfaces, \( n = 100 \)
- For moderately specular, \( n = 10 \)

**Blinn-Phong model (Jim Blinn, 1977)**
- Variation with better physical interpretation
  - \( h \): halfway vector; \( n \): shininess
  \[ I_{\text{out}}(x) = k_s \cdot (h \cdot n)^n \cdot I_{\text{in}}(x); \text{ with } h = (L + V) / 2 \]

**Phong Model**
- Polar plot

\( n=10 \)
\( n=100 \)

**Blinn-Phong Model**
- Polar plot

\( r=0.1 \)
\( r=0.01 \)
Commonly used model (simple, non-physical)

Combine diffuse, specular, ambient
- E.g. OpenGL / graphics hardware:

\[ I_{out}(x) = k_a \cdot I_a + k_d \cdot (l \cdot n) \cdot I_{diff} + k_s \cdot (h \cdot n)^n \cdot I_{spec} \]

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)

```c
void setProperties(void)
{
    /* Default values for material and light properties.*/
    GLfloat mat_ambient[] = { 0.2, 0.2, 0.2, 1.0};
    GLfloat mat_diffuse[] = { 0.6, 0.6, 0.6, 1.0};
    GLfloat mat_specular[] = { 0.5, 0.5, 0.5, 1.0};
    GLfloat lightAmbient[] = { 1.1, 1.1, 1.1};
    GLfloat lightDiffuse[] = { 1.1, 1.1, 1.1};
    GLfloat lightSpecular[] = { 1.1, 1.1, 1.1};
    GLfloat lightZero[] = { 0.0, 0.0, 0.0, 1.0};

    glEnable(GL_LIGHT0);
    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);
    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 );

    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, &nShininess);

    glEnable(GL_LIGHT0);
    glLightfv(GL_LIGHT0, GL_AMBIENT, lightAmbient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, lightDiffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, lightSpecular);
    glEnable(GL_LIGHTING);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_NORMALIZE);
}
```
Flat Shading

Example:

Gouraud Shading

Example:

Materials

Summary
- Very simple reflection models
- Fast (dot products & exponentiation)
- No physical justification
- Not very good for modeling real world

Light Sources

Quadratic falloff
- 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 Sources**

*Non-quadratic falloff:*
- Many systems allow for other falloffs
- Allows for faking of the effect of area light sources
- OpenGL / graphics hardware:
  - \( I_o \): intensity of light source
  - \( x \): object point
  - \( r \): distance of light from \( x \)

\[
I_{in}(x) = \frac{1}{ar^2 + br + c} \cdot I_0
\]

**Materials**

*Bi-directional Reflectance Distribution Function (BRDF):*
- Describes fraction of light reflected for all combinations of incoming (light) and outgoing (viewing) directions
- Color channels (R, G, B) are treated separately
  - Actually: wavelengths (see later in course)

**Bi-directional Reflectance Distribution Function (BRDF):**

\[
I_{out}(x) = f_r(\phi_{in}, \theta_{in}, \phi_{out}, \theta_{out}) \cdot I'_{in}(x)
= f_r(l \rightarrow v) \cdot (n \cdot l) \cdot I_{in}(x)
\]

\( f_r(l \rightarrow v) \) is called BRDF

\( (t,n,b) \) is *local coordinate frame* (normal, tangent, binormal)

**Polar plot of BRDF**
- Fix incoming light direction \( l \)
- Plot \( f_r(l \rightarrow v) \cdot v \) for all viewing directions \( v \)
- Works for 2D and 3D plots
- Example: 2D polar plot for diffuse BRDF
Light Sources

*Area lights:*

- light sources with a finite area
- more realistic model of many light sources
- Not available with projective rendering pipeline, (i.e., not available with OpenGL)

Gouraud Shading

*Mach Bands:*

- Eye enhances discontinuity in first derivative
- Very disturbing, especially for highlights