Virtual Reality for Human Computer Interaction

Appearance: Phong Lighting Model

Ambient light

- Ambient reflection depends on ambient light intensity $I_{a}$ and object's ambient material properties:
  - Objects diffuse color $O_{d}$ ($O_{dr}$, $O_{dg}$, $O_{db}$)
  - Overall fraction reflected is the ambient-reflection coefficient $k_{a}$ (0 to 1)
  - The overall fraction of primary reflected is: $k_{a} O_{d}$
  - Specification allows independent control of the overall intensity of reflection and of its color.
- **The illumination model at an object point thus far:**
  $$I_{a} = I_{a} k_{a} O_{d}$$
- Ambient light is not viewer location dependent
- Resulting images:

Ambient light in OpenGL

• Enable a global ambient light:
  float globalAmbient[] = {r,g,b,1};
  glLightModelfv (GL_LIGHT_MODEL_AMBIENT, globalAmbient);
• OpenGL allows an ambient term in individual lights (e.g., GL_LIGHT0)

• Specify ambient material property:
  • float ambient[] = {r,g,b,1};
  • glMaterialfv (GL_FRONT_AND_BACK, GL_AMBIENT, ambient);
• Note that $ka$ and $Odl$ are combined.


Diffuse Reflection

(i) Scattering of Reflective Light.
• Consider how a dull, matte surface (e.g., chalk) scatters light:

  • When its orientation is fixed relative to a light, its illumination looks the same from all viewing angles.
  • When its orientation changes relative to a light, its illumination changes.
    • It is brightest when the light shines directly on it.
    • It is dimmer when it makes an angle to the light.
  • This reflection is diffuse (Lambertian) reflection.
Lambertian Reflection

• Diffuse (Lambertian) reflection: what we see is according to Lambert’s law the vertical component of the incoming light

  This vertical component at \( p \) is:

  \[
  I_{p\lambda} = I_p \lambda \cos(\theta) \quad \text{or} \quad I_{p\lambda} = I_p \lambda (\mathbf{N} \cdot \mathbf{L}),
  \]

  where:

  • The unit surface normal at a point, \( p \), is \( \mathbf{N} \).
  • \( \mathbf{L} \) is a unit vector pointing to the light source
  • \( \theta \) is the angle between \( \mathbf{N} \) and \( \mathbf{L} \).
  • The reflected light is 0 for \( \theta > 90 \) degrees.

Lambertian Reflection

• The diffusely reflected light depends on the surface’s material properties:

  • Objects diffuse color \( O_d \) (\( O_{dr} \), \( O_{dg} \), \( O_{db} \))
  • The overall fraction reflected is the diffuse-reflection coefficient \( k_d \), range(0 to 1)
  • The overall fraction of primary reflected is: \( k_d O_{d\lambda} \)

  • Given point light source, the diffuse intensity at it is:

  \[
  I_{p\lambda} k_d O_{d\lambda} (\mathbf{N} \cdot \mathbf{L})
  \]
Lighting model continued

- The illumination model at a point is thus far:
  \[ I_\lambda = I_a \kappa_a O_{d\lambda} + I_p \kappa_a O_{d\lambda} (N \cdot L) \]
- Diffuse lighting is not viewer location dependent
- The dot product is calculated at every point
- The L vector is calculated at every point except:
  - The light’s position is infinitely far away.
  - All rays are parallel by the time they reach the scene.
- The resulting images look like:

Diffuse Reflection in OpenGL

- Specify the light’s color:
  ```
  float diffuse0[] = {r,g,b,1};
  glLightfv(GL_LIGHT0, GL_DIFFUSE, diffuse0);
  ```
- Specify the light’s direction:
  ```
  float direction0[] = {dx,dy,dz,0};
  glLightfv(GL_LIGHT0, GL_POSITION, direction0);
  ```
  - The parameter being set is GL_POSITION
  - The 0 in the last element of direction0 indicates that this light is a directional light.
- Specify diffuse material property:
  ```
  float diffuse[] = {r,g,b,1};
  glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, diffuse);
  ```
Specular Reflection

- Consider a glossy, shiny surface (e.g., plastic, metal).
  - The surface reflects a bright highlight.
  - The highlight changes with viewing angle.
- This reflection is specular reflection.
- Reflection is highest in a certain direction.
- Reflection intensity depends on angle between reflection distribution and viewer position.

More precisely:
- $N$ is the unit normal at point $p$.
- $L$ is the unit vector pointing to the light source.
- $\theta$ is the angle between $N$ and $L$.
- $R$ is the vector of mirror reflection.
  - $R$ also makes angle with $N$.
  - $R$ is on the "other side" of $L$.
- $V$ is a unit vector pointing to the camera.
  - $\alpha$ is the angle between $R$ and $V$.

- The highlight’s visible intensity depends on:
  1. $\alpha$
     - The highlight is most intense when $\alpha = 0$
     - The highlight becomes dimmer as $\alpha$ grows.
  2. material properties
     - Example: Mirror reflects only with $\alpha = 0$
     - A Mirror is a Perfect Reflector
Phong Model for Non-Perfect Reflectors

- A light of intensity $I_{p,l}$ produces a highlight intensity proportional to $I_{p,l} \left( \cos (\alpha) \right)^n$.
- The exponent, $n$ is a material property
  - (specular-reflection coefficient)
  - Varies from 1 to several hundred
    (from broad gentle falloff to sharp focused falloff):

Specular Reflection

- Other material properties affect the intensity specularly reflected.
  - The overall fraction of light reflected is $W(\theta)$, often taken to be the constant $k_s$ (ranges 0 to 1)
  - The fraction of primary $\lambda$ reflected is $O_{\lambda,i}$
- The specular intensity is thus:
  \[ I_{p,l} k_s O_{\lambda,i} \left( \cos (\alpha) \right)^n \quad \text{or} \quad I_{p,l} k_s O_{\lambda,i} \left( R \cdot V \right)^n \]
- The illumination model at a point is thus so far:
  \[ I_\lambda = I_{d,l} k_a O_{d,l} + I_{p,l} k_d O_{d,l} (N \cdot L) + I_{p,l} k_s O_{\lambda,i} (R \cdot V)^n \]
Approximating Phong Lighting

- Calculating Phong Lighting equation requires calculation of perfect light reflection vector.
- Term is maximum when \( R \) and \( V \) are equal.
- A popular variation by (Blinn, 1977): \( I_{spec} = (N \cdot H)^n \)
- Where \( H \) is the normalized halfway vector between \( L \) and \( V \): \( H = \frac{L + V}{|L + V|} \)
- This approximation is expressed as: \((R \cdot V)^n = (N \cdot H)^{4n}\)

Applying the lighting model

\[
I_{\lambda} = I_{a\lambda} k_a O_{a\lambda} + I_{p\lambda} k_d O_{p\lambda} (N \cdot L) + I_{p\lambda} k_s O_{p\lambda} (R \cdot V)^n
\]
Specular Reflection in OpenGL

• Specify the light that can be specularly reflected:
  float specular0[] = {r,g,b,1};
  glLightfv(GL_LIGHT0, GL_SPECULAR, specular0);

• Specify specular material properties:
  glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, n);
  float specular[] = {r,g,b,1};
  glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, specular);
  Note that $k_s$ and $O_{s,l}$ are combined.

Light-Source Attenuation

• To deal with light source distance, we introduce: $f_{att}$
• One option – light energy falls off at inverse square:
  \[ f_{att} = \frac{1}{d^2} \]
  • In reality – this does not work well
• Alternative:
  \[ f_{att} = \min \left( \frac{1}{c_1 + c_2 d_L + c_3 d_L^2}, 1 \right) \]
  • Where $c_1$, $c_2$, and $c_3$ are user defined constants for a light source
  • OpenGL: Attenuation can be set for each light source
Atmospheric Attenuation

- Handles distance from observer to object
- More distant objects rendered with lower intensity than closer ones
- Define front and back depth-cue planes, and a (low intensity) color $l_{\text{dc}}$ at the back depth-cue plane
- Set: $I'_\lambda = f_{\text{fog}} I'_\lambda + (1 - f_{\text{fog}}) l_{\text{dc}}$
  - Where $f_{\text{fog}} = 0$ for objects in front of front plane,
  - $f_{\text{fog}} = 1$, for objects behind back plane,
  - $f_{\text{fog}}$, the fog factor, increasing between front and back planes
  - If $f_{\text{fog}}$ increases, fog effect decreases...

- FOG is OpenGL’s implementation of atmospheric attenuation

- The illumination model at a point is finally:
  $$I'_\lambda = I_a \times K_a \times O_{\text{cl}} + f_{\text{att}} [I_p \times K_p \times O_{\text{cl}} \times (N \cdot L) + I_s \times K_s \times O_{\text{cl}} \times (R \cdot V)^n]$$
  $$I'_\lambda = f_{\text{fog}} I'_\lambda + (1 - f_{\text{fog}}) l_{\text{dc}}$$

Fog example

- Often just a matter of
  - Choosing fog color
  - Choosing fog model
  - Turning it on
- How to compute $f_{\text{fog}}$?
- 3 ways
  - Linear: $f_{\text{fog}} = \frac{z_{\text{end}} - z_p}{z_{\text{end}} - z_{\text{start}}}$
  - Exponential: $f_{\text{fog}} = e^{-d z_p}$
  - Exponential-squared: $f_{\text{fog}} = e^{-(d z_p)^2}$
Appearance

- Appearance can be greatly enhanced by using textures.
- Textures are 2D or 3D arrays of values which are fed as an additional parameter source into the render pipe’s calculations.
- Textures are used to describe appearance details.
- Simple textures are images which define a pixel’s color or a color modification of the color produced by the lighting calculation.
- Normal maps or bump maps describe fine-grained surface structures.
- Shadow maps describe shadows cast onto a geometry.
- Environment maps reflect light from the environment (e.g., for simulating mirrors).
- ...

What now?

Shading

- Ideally, the renderer should apply the illumination model at every visible point on each surface.
- This approach requires too much computation.
- As an alternative:
  - Apply the illumination model at a subset of points.
  - Interpolate the intensity of the other points.
  - Apply the illumination model only visible surfaces