Introduction

This lecture and the next we discuss illumination and shading models.

Illumination models express how light affects a surface's color at a given point.

Shading models use illumination models to determine the color across a surface.

• Apply the illumination model at some points.
• Interpolate the illumination at other points.
This process is sometimes called “lighting the object”.

The models are only loosely physical, and emphasize:
• empirical success
• efficiency

For example, these models focus on direct illumination.

Later lectures will cover:
• indirect illumination
• shadows

Outline

1. Light sources
2. Basic Illumination Models account for
   • The light source
     - ambient light
     - point light source
     - distributed light source
     - spot lights
   • The surface properties
     - diffuse/rough/dull (diffuse reflection)
     - specular/smooth/shiny (specular reflection, Phong)
   • The geometry of source/surface/viewer, distances.
   • For more realistic appearance, the reflection from a surface is attenuated sum of ambient, diffuse and specular component s (in all primaries, and also over all light sources)
3. Lighting and shading in OpenGL.
   Light sources can be turned on/off:
   glEnable(GL_LIGHTING);
   glEnable(GL_LIGHT0);
OpenGL supports multiple lights, but performance suffers. Lighting model adds independent components (emissive, ambient, diffuse, specular color “light”, also similar component description of “materials reflectance”).

Illumination models

Illumination, I, at a point is modeled as the sum of several terms.
• More terms give more plausible results.
• Fewer terms give more efficient computations.

Each additive term of I is expressed in primary colors, Iₐ, I₉, and I₈, with each such Iₙ computed independently.
• Components used to express how much light a source emits and a surface reflects.

Each Iₙ is the sum of a terms, one for each light source, computed independently.
• Overflow is possible
• There are various solutions for dealing with it.
  - One solution is to clamp Iₙ to max allowable
  - Normalize individual terms

Ambient light

An ambient term approximates uniform, indirect illumination present everywhere.

The ambient light intensity is Iₐₗ for primary color λ.

How an object reflects ambient light depends on its material properties.
• The overall fraction reflected is kₐ.
• The fraction of primary λ reflected is Oₐₗ.
• The overall fraction of primary λ reflected is thus kₐ Oₐₗ.
• This specification allows independent control of the overall intensity of reflection and of its color.

The illumination model at an object point is thus so far:

Iₙ = Iₐₗkₐ Oₐₗ.

A resulting image:
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).
• What does this mean?

Specify ambient material property:
float ambient[] = {r, g, b, 1};
glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, ambient);
• Note that \( r, g, b \) are combined.
• Note different meaning of \( r, g, b, 1 \)

Nonambient light

Most light sources do not illuminate all surfaces uniformly.
Point light sources:
• radiate evenly in all directions from a position
• provide different illumination on surfaces with different orientations.
• when point light source is far, the light rays are well approximated by parallel rays

Diffuse reflection

Consider a dull, matte surface (e.g., chalk), it 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: what we see is according to Lambert’s law is the vertical component of the incoming light:
This vertical component at \( p \) is \( I_p \cos(\theta) \) or \( I_p (\mathbf{N} \cdot \mathbf{L}) \),
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 \( \theta > 90^\circ \).
Intuition: consider a far point source shining on a plane. As the source is lowered the same amount of light is spread over larger area (vertical component decreases, surface appears dimmer)

The diffusely reflected light depends on the surface’s material properties.
• The overall fraction reflected is \( k_d \).
• The fraction of primary \( \lambda \) reflected is \( O_{\lambda} \).
Given point light source, the diffuse intensity at it is:
\[
I_p = k_d O_{\lambda} (\mathbf{N} \cdot \mathbf{L})
\]
The illumination model at a point is thus so far:
\[
I_k = I_o k_o O_{\lambda} + I_p k_d O_{\lambda} (\mathbf{N} \cdot \mathbf{L})
\]
The dot product calculated at every point. Evaluation is more efficient assuming directional point light sources.
• The light’s position is infinitely far away
• All rays are parallel by the time they reach the scene.
• \( \mathbf{L} \) is a constant throughout the scene.

A resulting image:
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.

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

- The highlight’s visible intensity depends on \( \alpha \).
  - The highlight is most intense when \( \alpha = 0 \).
  - The highlight becomes dimmer as \( \alpha \) grows.

Specular reflection

An equivalent expression for specular reflection is:

\[
I_{pl} = I_{pe} k_s O_{sh} (\mathbf{R} \cdot \mathbf{V})^n
\]

Deriving \( \mathbf{R} \):

1. Perfect specular reflection (mirror):
2. In 1975, Phong developed and approximation to the dependency of specular reflection on \( \alpha \).
   - A light of intensity \( I_{pl} \) produces a highlight intensity proportional to \( I_{pe} (\cos(\alpha))^n \).
   - The exponent, \( n \) is a material property.
   - Other material properties affect the intensity specularly reflected.
     - The overall fraction reflected is \( W(\theta) \), often taken to be the constant \( k_r \).
     - The fraction of primary \( \lambda \) reflected is \( O_{sh} \).

The specular intensity is thus:

\[
I_{pl} = I_{pe} k_s O_{sh} (\cos(\alpha))^n
\]

Specular reflection

The illumination model at a point is thus so far:

\[
I_{pl} = I_{pe} k_s O_{sh} + I_{pe} k_o O_{sh} (\mathbf{N} \cdot \mathbf{L}) + I_{pe} k_s O_{sh} (\mathbf{R} \cdot \mathbf{V})^n
\]

Some resulting images:

- Use the halfway vector, \( \mathbf{H} = (\mathbf{L} + \mathbf{V})/\|\mathbf{L} + \mathbf{V}\| \), to approximate \( \mathbf{R} \cdot \mathbf{V} \) by \( \mathbf{N} \cdot \mathbf{H} \).
Specular reflection in OpenGL

Specify the light that can be specularly reflected:
```c
float specular0[] = {r, g, b, 1};
glLightfv(GL_LIGHT0, GL_SPECULAR, specular0);
```
Specify specular material properties:
```c
float specular[] = {r, g, b, 1};
glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, specular);
```
- Note that $k_s$ and $O_A$ are combined.

Other terms in illumination models

Light can attenuate with distance from the source.
Fog can affect the reflection visible at the camera.
Spotlights can produce light limited to a cone around $L$.

Shading models

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

Flat (constant) shading

Sample illumination at one point per polygon.
Use constant interpolation: all other points on the polygon get that point’s intensity.
This approach would be valid if:
- The true surface really is faceted, so $N$ is constant.
- The light source is at infinity so $L$ is constant.
- The viewer is at infinity so $V$ is constant.
A resulting image:

Flat shading in OpenGL

Just enable it:
```c
glShadeModel(GL_FLAT);
```

Mach banding

Flat shading of more facets does not necessarily look smoother.
Imaginary dark and light lines appear at facet boundaries.
These lines appear at any discontinuity or drastic change in the rate of shading.
These lines are perceptual artifacts called *Mach bands*.
Mach bands are caused by the eye’s *lateral inhibition*.
- When one receptor responds to a high intensity, it inhibits its neighboring receptors’ responses.
- Receptors on the bright side of a discontinuity receive less inhibition from the dark side.
- Receptors on the dark side of a discontinuity receive more inhibition from the light side.
Gouraud shading

Apply the illumination model at each polygonal vertex.
- \( N \) is the vertex normal.
- The calculation must use \( N \) in world coordinates.

Interpolate intensities as part of scan conversion.
- Interpolate span endpoints from edge vertices.
- Interpolate points within a span from span endpoints.
- Perform interpolation through incremental updates.

Gouraud shading reduces Mach bands (but not entirely).
- Vertices shared between polygons will have the same vertex normal, and hence the same intensity.

Some resulting images:

Gouraud shading in OpenGL

Just enable it:
```c
glShadeModel(GL_SMOOTH);
```

Phong shading

Apply the illumination model at each pixel.
Interpolate a pixel’s normal, \( N \), as part of scan conversion.
- To get \( N \) at span endpoints, interpolate from edge vertices’ normals.
- To get \( N \) within a span, interpolate from span endpoints.
- Perform interpolation through incremental updates.
- The vertex \( N \)s must be in world coordinates so the interpolated \( N \)s will be, too, for the illumination calculations.

This approach avoids some errors in specular illumination that appear with Gouraud shading.
- Gouraud misses specular highlights within polygons.
- Gouraud spreads specular highlights along edges.

Problems common to Gouraud and Phong shading

Silhouette edges are not smoothed.

Interpolation ignores perspective distortion.
- Due to foreshortening, a change in scanlines does not correspond to a constant change in \( z \) in OpenGL eye coordinates.

- So the scanline halfway between two vertices does \textit{not} correspond to \( z \) halfway between the vertices’ \( z \)s.
- But pixels on that scanline get an interpolated quantity (intensity or \( N \)) that \textit{does} correspond to the halfway \( z \).