CSE 167 (FA22)
Computer Graphics: Lighting
Albert Chern
Shading color on geometry, Illumination

- Introduction
- Reflection model
- Phong reflection model
- Getting $v$, $n$, $l$ correctly
- Gouraud vs Phong shadings
Normal shading

• Coloring based on surface normal
  ▶ X component maps to Red.
  ▶ Y component maps to Green.
  ▶ Z component maps to Blue.

• Need to map the range $[-1,1]$ of the components of the normal vector to the range $[0,1]$ for color.

\[
\text{Color} = 0.5 \times \mathbf{n} + (0.5,0.5,0.5)
\]
Realistic shading

- **Appearance**
  = Material definition + light sources.
- **Compute interaction of light with materials**
- **Requires simulation of optical physics**
- **“Global illumination”**
  - Multiple bounces of light
  - Computationally expensive, minutes per image
  - Used in movies, architectural designs, etc.
Realistic shading

- Computes interaction of light with materials
- Requires simulation of physics
- Global illumination
- Multiple bounces of light
  - Computationally expensive, minutes per image
- Used in movies, architectural designs, etc.

Appearance = Material definition + light sources.
Interactive applications

- Simplified models instead of fully physics-based light simulation
- Reproduce perceptually most important effects
- “Local illumination”
  - Only one bounce of light between light source and viewer
Local illumination

- Describe reflection of light only at surfaces
  - Assumption: no subsurface scattering

- Light can be reflected by
  - Mirror
  - Rough surface
  - Glossy material
  - etc

- Gives material its color
Local Reflection Model

- Introduction
- Reflection model
- Phong reflection model
- Getting $v$, $n$, $I$ correctly
- Gouraud vs Phong shadings
Local illumination

• Bidirectional reflectance distribution function (BRDF)
  ▶ Given light direction, viewing direction, how much light is reflected towards the viewer (per unit range of direction)
  ▶ For any pair of light/viewing direction!

• Usually use a reflection model to approximate a real physical BRDF
Reflection model

- A lighting model or reflection model is a \( \text{vec3} \)-valued function

\[
C(v, n, l; \text{param}_1, \ldots, \text{param}_k)
\]

\( v = \) unit vector pointing to the viewer \( \in \mathbb{R}^3 \)

\( n = \) unit outward normal vector of the surface \( \in \mathbb{R}^3 \)

\( l = \) unit vector pointing to the light \( \in \mathbb{R}^3 \)

Parameters are typically intuitive material properties

in the world coordinate or in the camera coordinate (NOT the model coordinate or the normalized device coordinate)
A lighting model or reflection model is a \( \text{vec3} \)-valued function

\[
C(v, n, l; \text{param}_1, \ldots, \text{param}_k)
\]

- \( v \) = unit vector pointing to the viewer \( \in \mathbb{R}^3 \)
- \( n \) = unit outward normal vector of the surface \( \in \mathbb{R}^3 \)
- \( l \) = unit vector pointing to the light \( \in \mathbb{R}^3 \)

in the world coordinate or in the camera coordinate (NOT the model coordinate or the normalized device coordinate).

The \( \text{vec3} \) output values of \( C \) represent “The portion of light is allowed to pass after the reflection in each of the RGB channels”

- Suppose the light source emits light color \( L = (L_r, L_g, L_b) \in \mathbb{R}^3 \)
- Suppose the value of reflection model is \( C = (C_r, C_g, C_b) \in \mathbb{R}^3 \)
- Then the viewer sees light color \( R = CL = (C_r L_r, C_g L_g, C_b L_b) \in \mathbb{R}^3 \)
Reflection model

- **A lighting model or reflection model** is a \texttt{vec3}-valued function

\[ C(v, n, l; \text{param}_1, \ldots, \text{param}_k) \]

- \( v \) = unit vector pointing to the viewer \( \in \mathbb{R}^3 \)
- \( n \) = unit outward normal vector of the surface \( \in \mathbb{R}^3 \)
- \( l \) = unit vector pointing to the light \( \in \mathbb{R}^3 \)

\textit{parameters are typically intuitive material properties}

in the \textit{world coordinate} or in the \textit{camera coordinate}

- The reflection model should satisfy frame invariance:
  - For any \textit{rotation matrix} \( Q \in \mathbb{R}^{3 \times 3} \)
    \[ C(v, n, l) = C(Qv, Qn, Ql) \]
    \( \text{not true for arbitrary matrices!} \)
  - So using either world coordinate or camera coordinate will give the same result
Side note on multiplication of colors

- There are two types of color vectors
  - **Light** $\mathbf{L}$ (values are positive)
  - **Transmittance/reflectance** $\mathbf{C}$ (values usually range $[0,1]$)
- Both have RGB channels
- Product of a light and a transmittance is a light

\[ \mathbf{R} = \mathbf{C} \mathbf{L} \]

with the multiplication carried out *componentwise*

\[ R_i = C_i L_i, \quad i = r, g, b \]

- Product of transmittances is still a transmittance. There is no product between lights.

\[ \mathbf{R} = \mathbf{C}_1 \mathbf{C}_2 \mathbf{C}_3 \mathbf{L} \]
Side note on multiplication of colors

• The color finally shown on the pixel is of the type of light (rather than transmittance)
  ▶ The value of light can be greater than 1.
  ▶ Over exposure can happen.
  ▶ 8-bit integer \{0, \ldots, 255\} can represent \([0,1]\) interval, but not values greater than 1.
  ▶ To keep the real value of light, one can use float to store the color. This is called high dynamic range image (HDRI)
  ▶ Or, one can map between \([0,1]\) and \([0,a]\) intervals using \(y = ax^\gamma\). This is called tone mapping. The use of the power law is called Gamma color correction.
Basic Diffuse, Specular, Ambient (Phong reflection model)

- Introduction
- Reflection model
- Phong reflection model
- Getting $v$, $n$, $I$ correctly
- Gouraud vs Phong shadings
Phong reflection model

- Phong reflection model
  - Sum of 3 components
  - Covers a large class of real surfaces
Phong reflection model

- Phong reflection model
  - Sum of 3 components
  - Covers a large class of real surfaces
Diffuse reflection

- Ideal diffuse material reflects light equally in all directions
- View-independent
- Matte, non-shiny materials
  - Paper
  - Unpolished wood, stone
- Provides visual cues
  - Surface curvature
  - Depth variation
Diffuse reflection

- Beam of parallel rays shining on a surface
  - Amount of rays intercepted by the surface per unit area varies with the angle between the beam and the normal.

The oblique sunbeam distributes its light energy over twice as much area.
Diffuse reflection

• Beam of parallel rays shining on a surface
  ▶ Amount of rays intercepted by the surface per unit area varies with the angle between the beam and the normal.

• Lambert’s cosine law (1760)
  ▶ Incident light per unit area is proportional to the cosine of the angle between the light direction and the normal.
  ▶ Object darkens as normal turns away from light.
  ▶ Diffuse surfaces are also called Lambertian surfaces.
Derivation of Lambert Law

portion of the light that contributes to a unit area of the surface
Diffuse reflection

• Given
  ▶ Unit (normalized) surface normal \( n \)
  ▶ Unit (normalized) light direction \( l \)
  ▶ Material diffuse reflectance (material color) \( C_{\text{diffuse}} \)
  ▶ Light color (intensity) \( L \)

• The reflected diffuse color (intensity) is

\[
R_{\text{diffuse}} = C_{\text{diffuse}} \cdot L \cdot \max(n \cdot l, 0)
\]
Local illumination

- Simplified model
  - Sum of 3 components
  - Covers a large class of real surfaces
Specular reflection

- Shiny surfaces
  - Polished metal
  - Glossy car finish
  - plastics

- Specular highlight
  - Blurred reflection of the light source
  - Position of highlight depends on viewing direction
Specular reflection

• Ideal specular reflection
  ▶ Perfectly smooth surface
  ▶ Incoming light is bounced in single direction
  ▶ Angle of incidence equals angle of reflection
Specular reflection

- Reflection direction
  - Given unit surface normal $\mathbf{n}$ and light direction $\mathbf{l}$
  - The projection of $\mathbf{l}$ on $\mathbf{n}$ is $(\mathbf{n} \cdot \mathbf{l})\mathbf{n}$
  - The unit reflection direction $\mathbf{r}$ satisfies \[
  \frac{\mathbf{r} + \mathbf{l}}{2} = (\mathbf{n} \cdot \mathbf{l})\mathbf{n}
  \]
  - Therefore, $\mathbf{r} = 2(\mathbf{n} \cdot \mathbf{l})\mathbf{n} - \mathbf{l}$
Specular reflection (glossy)

- Many materials are not perfect mirrors
  - Glossy material
- Microscopic variation/noise of normals
- Smooth surface has sharp highlight
- Rough surface has blurred highlight
Specular reflection (glossy)

- Most light still follow the mirror reflection direction
- Due to microscopic variation/noise of normals, some light is reflected off the ideal reflection direction.
  - Brightest when view vector $\mathbf{v}$ aligns with reflection $\mathbf{r}$
  - Decreases as the angle between $\mathbf{v}$ and $\mathbf{r}$ increases.
Phong’s specular reflection model

- Developed by Bui Tuong Phong (1973)
- Let $C_{\text{specular}}$ be the specular reflectance coefficient.
- Let $p$ be the Phong exponent (bigger $p$ gives sharper the highlight)

$$R_{\text{Phong specular}} = C_{\text{specular}} L \left[ \max(v \cdot r, 0) \right]^p$$
Blinn–Phong specular reflection model

- Modified by Jim Blinn (1977)
- Compute the half-way vector
  \[ h = \frac{v + l}{|v + l|} \]
- Replace \((v \cdot r)\) by \((n \cdot h)\)

\[ R_{\text{BlinnPhong, specular}} = C_{\text{specular}} L \left[ \max(n \cdot h, 0) \right]^{\sigma} \]

- For distant light and camera, \(h\) is constant. This can speed up the rendering.
Blinn–Phong reflection model

\[ \sigma = 4p \]

Phong  Blinn–Phong
Local illumination

- Simplified model
  - Sum of 3 components
  - Covers a large class of real surfaces
Ambient light

• In real world, light is bounced all around the scene
• Areas with no direct illumination are not completely dark
• Could use global illumination techniques to simulate ambient light
• Simple approximation

\[ R_{\text{ambient}} = C_{\text{ambient}} L \]
Complete Phong shading model

- Phong model supports multiple light sources

\[
R = E + \sum_{j} L_j \left( C_{\text{ambient}} + C_{\text{diffuse}} \max(n \cdot l_j, 0) + C_{\text{specular}} \left[ \max(n \cdot h_j, 0) \right]^\sigma \right)
\]

self-emission

ambient + diffuse + specular = full shading model
Complete Phong shading model

- Phong model supports multiple light sources

\[
R = E + \sum_{j} \frac{L_j}{f_j(d_j)} \left( C_{\text{ambient}} + C_{\text{diffuse}} \max(n \cdot l_j, 0) + C_{\text{specular}} \left[ \max(n \cdot h_j, 0) \right]^{\sigma} \right)
\]

\[
\begin{align*}
\sigma & : distance to the j-th light \\
f(x) & = a_0 + a_1 x + a_2 x^2
\end{align*}
\]

A more accurate model is to include distance-drop-off.
Getting view, normal, light directions correctly

- Introduction
- Reflection model
- Phong reflection model
- Getting $v$, $n$, $l$ correctly
- Gouraud vs Phong shadings
Getting $v$, $n$, $l$ correctly

- Suppose we are rendering an object with raw vertex buffer data
  - Position $p_{\text{model}} \in \mathbb{R}^3_{\text{model}}$
  - Normal $n_{\text{model}} \in \mathbb{R}^3_{\text{model}}$

using 4x4 model matrix $M$, 4x4 view matrix $V$, and thus modelview matrix $VM$ (all affine)

- Suppose there is a light at location $q_{\text{world}} \in \mathbb{R}^3_{\text{world}}$ in the world coordinate

- Construct $v$, $n$, $l$ either all in world coordinate or all in camera coordinate
• Construct $v, n, l$ either all in world coordinate or all in camera coordinate
Getting $\mathbf{v}$, $\mathbf{n}$, $\mathbf{l}$ correctly

Construct $\mathbf{v}, \mathbf{n}, \mathbf{l}$ either all in world coordinate or all in camera coordinate.
Getting $\mathbf{v}$, $\mathbf{n}$, $\mathbf{l}$ correctly

- In the camera coordinate

\[
\begin{bmatrix}
q_{\text{cam}}^1 \\
0 \\
0 \\
0
\end{bmatrix} = \begin{bmatrix}
V \\
0 \\
0 \\
0
\end{bmatrix} \begin{bmatrix}
q_{\text{world}}^1 \\
0 \\
0 \\
0
\end{bmatrix}
\]

\[
\begin{bmatrix}
p_{\text{cam}}^1 \\
0 \\
0 \\
0
\end{bmatrix} = \begin{bmatrix}
VM \\
0 \\
0 \\
0
\end{bmatrix} \begin{bmatrix}
p_{\text{model}}^1 \\
0 \\
0 \\
0
\end{bmatrix}
\]
Getting $\mathbf{v}$, $\mathbf{n}$, $\mathbf{l}$ correctly

- In the world coordinate

\[
\begin{bmatrix}
\mathbf{p}_{\text{world}}^1 \\
\mathbf{i}_{\text{world}}^1
\end{bmatrix} = \begin{bmatrix}
\mathbf{M}
\end{bmatrix} \begin{bmatrix}
\mathbf{p}_{\text{model}}^1 \\
\mathbf{0}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\mathbf{i}_{\text{world}}^1
\end{bmatrix} = \begin{bmatrix}
\mathbf{V}^{-1}
\end{bmatrix} \begin{bmatrix}
0 \\
0 \\
0 \\
1
\end{bmatrix}
\]
Getting $v$, $n$, $l$ correctly

- What about normal?
Getting $\mathbf{v}$, $\mathbf{n}$, $\mathbf{l}$ correctly

- Suppose we have an affine transformation on positions
  \[
  \begin{bmatrix}
  p \\
  1
  \end{bmatrix} \mapsto
  \begin{bmatrix}
  A & b \\
  0 & 1
  \end{bmatrix}
  \begin{bmatrix}
  p \\
  1
  \end{bmatrix}
  \]

- and normal vectors transform according to
  \[
  \begin{bmatrix}
  \mathbf{n}
  \end{bmatrix} \mapsto
  \text{What is this 3x3 matrix?}
  \begin{bmatrix}
  \mathbf{n}
  \end{bmatrix}
  \] (followed by a normalization)
Getting $\mathbf{v}$, $\mathbf{n}$, $\mathbf{l}$ correctly

- Suppose we have an affine transformation on positions
  \[
  \begin{bmatrix}
  p \\ 1
  \end{bmatrix}
  \rightarrow
  \begin{bmatrix}
  A & b \\ 0 & 1
  \end{bmatrix}
  \begin{bmatrix}
  p \\ 1
  \end{bmatrix}
  \]

- and normal vectors transform according to
  \[
  \begin{bmatrix}
  \mathbf{n}
  \end{bmatrix}
  \rightarrow
  \begin{bmatrix}
  A^{-T}
  \end{bmatrix}
  \begin{bmatrix}
  \mathbf{n}
  \end{bmatrix}
  \] (followed by a normalization)
Getting $\mathbf{v}$, $\mathbf{n}$, $\mathbf{l}$ correctly

- In the world coordinate

\[
\begin{bmatrix}
    p_{world}^T \\
    1
\end{bmatrix}
= \begin{bmatrix}
    M \\
    1
\end{bmatrix}
\begin{bmatrix}
    p_{model}^T \\
    1
\end{bmatrix}
\]

\[
\begin{bmatrix}
    \mathbf{v}_{world}^T \\
    0 \\
    0 \\
    0 \\
    1
\end{bmatrix}
= \begin{bmatrix}
    \mathbf{V}^{-1}
\end{bmatrix}
\begin{bmatrix}
    0 \\
    0 \\
    0 \\
    1
\end{bmatrix}
\]

\[
\mathbf{n}_{world} = \text{NORMALIZE} \left( \begin{bmatrix}
    M_{\text{top-left}} \\
    3 \times 3 \text{ block}
\end{bmatrix}^{-T} \mathbf{n}_{model} \right)
\]
Why do normals transform like that?

- Equation for plane in 3D: \( ax + by + cz + d = 0 \)
- Normal vector of the plane: \( \mathbf{n} = \text{NORMALIZE} \left( \begin{bmatrix} a \\ b \\ c \end{bmatrix} \right) \)

Conversely, given a point \( \mathbf{p} \in \mathbb{R}^3 \) and a normal \( \mathbf{n} \in \mathbb{R}^3 \), the plane passing through the point with the given normal has

\[(a, b, c, d) = (n_x, n_y, n_z, -(\mathbf{p} \cdot \mathbf{n}))\]
Why do normals transform like that?

- Equation for plane in 3D: $ax + by + cz + d = 0$
- Normal vector of the plane: $\mathbf{n} = \text{NORMALIZE}\left(\begin{bmatrix} a \\ b \\ c \end{bmatrix}\right)$
- The plane can be described in homogeneous coordinate as that a point $\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$ (or any other representative $\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda z \\ \lambda \end{bmatrix}$) is on the plane if and only if $\begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda z \\ \lambda \end{bmatrix} = 0$
- Under transformations (linear, affine, projective) we need to transform the plane coefficients so that the incidence is preserved (transformed point lies on the transformed plane)
Why do normals transform like that?

\[
\begin{bmatrix}
a & b & c & d
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix} = 0
\quad \text{if and only if} \quad
\begin{bmatrix}
a' & b' & c' & d'
\end{bmatrix}
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = 0
\]

where

\[
\begin{bmatrix}
x' \\
y' \\
z' \\
w'
\end{bmatrix} = \mathbf{L}_{4\times4}
\begin{bmatrix}
x \\
y \\
z \\
w
\end{bmatrix}
\]

and

\[
\begin{bmatrix}
a' \\
b' \\
c' \\
d'
\end{bmatrix} = \begin{bmatrix} ?? \end{bmatrix}
\]

TBD
Why do normals transform like that?

\[
\begin{bmatrix}
  a & b & c & d
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix} = 0 \quad \text{if and only if} \quad
\begin{bmatrix}
  a' & b' & c' & d'
\end{bmatrix}
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix} = 0
\]

where

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  w'
\end{bmatrix} = \begin{bmatrix} L_{4\times4} \end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  w
\end{bmatrix}
\]

\[
\begin{bmatrix}
  a' & b' & c' & d'
\end{bmatrix} = \begin{bmatrix} ?? \end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c \\
  d
\end{bmatrix}
\]
Why do normals transform like that?

\[
\begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 0 \text{ if and only if } \begin{bmatrix} a' & b' & c' & d' \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = 0
\]

must be (scalar multiple of) \( L^{-1} \)

\[
\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} L_{4 \times 4} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}
\]

must be (scalar multiple of) \( L^{-T} \)
Why do normals transform like that?

- Therefore, when points are transformed by

\[
\begin{bmatrix}
    x' \\
    y' \\
    z' \\
    w'
\end{bmatrix} = \begin{bmatrix}
    L_{4\times4}
\end{bmatrix} \begin{bmatrix}
    x \\
    y \\
    z \\
    w
\end{bmatrix}
\]

coefficients of plane should be transformed by

\[
\begin{bmatrix}
    a' \\
    b' \\
    c' \\
    d'
\end{bmatrix} = \begin{bmatrix}
    L_{4\times4}
\end{bmatrix}^{-1} \begin{bmatrix}
    a \\
    b \\
    c \\
    d
\end{bmatrix}
\]

\(\begin{bmatrix}
    a' \\
    b' \\
    c'
\end{bmatrix} = \begin{bmatrix}
    \cdot \\
    \cdot \\
    \cdot
\end{bmatrix}
\]

\( (a, b, c, d) = (n_x, n_y, n_z, -(p \cdot n)) \)

- Now, read off the new normal vectors:

\[
n' = \text{NORMALIZE} \left( \begin{bmatrix}
    a' \\
    b'
\end{bmatrix} \right)
\]
Why do normals transform like that?

- Therefore, when points are transformed by

\[
\begin{bmatrix}
    x' \\
    y' \\
    z' \\
    w'
\end{bmatrix} = \begin{bmatrix}
    L_{4\times4}
\end{bmatrix}\begin{bmatrix}
    x \\
    y \\
    z \\
    w
\end{bmatrix}
\]

coefficients of plane should be transformed by

\[
\begin{bmatrix}
    a' \\
    b' \\
    c' \\
    d'
\end{bmatrix} = \begin{bmatrix}
    L_{4\times4}
\end{bmatrix}^{-\top} \begin{bmatrix}
    a \\
    b \\
    c \\
    d
\end{bmatrix}
\]

- Now, read off the new normal vectors:

\[
n' = \text{NORMALIZE}\left(\begin{bmatrix}
    a' \\
    b' \\
    c'
\end{bmatrix}\right)
\]

For **affine transforms**

\[
L = \begin{bmatrix}
    A_{3\times3} & b_{3\times1} \\
    0_{1\times3} & 1
\end{bmatrix}
\]

\[
L^{-1} = \begin{bmatrix}
    A^{-1} & -A^{-1}b \\
    0 & 1
\end{bmatrix}
\]

\[
L^{-\top} = \begin{bmatrix}
    A^{-\top} & 0_{3\times1} \\
    (-A^{-1}b)^\top & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
    a' \\
    b' \\
    c' \\
    d'
\end{bmatrix} = \begin{bmatrix}
    A^{-\top} & 0_{3\times1} \\
    (-A^{-1}b)^\top & 1
\end{bmatrix} \begin{bmatrix}
    a \\
    b \\
    c \\
    d
\end{bmatrix}
\]

new normal

\[
n' = \frac{A^{-\top}n}{|A^{-\top}n|}
\]

old normal
Why do normals transform like that?

For *affine transforms*

\[
L = \begin{bmatrix}
A_{3\times3} & b_{3\times1} \\
0_{1\times3} & 1
\end{bmatrix}
\]

\[
L^{-1} = \begin{bmatrix}
A^{-1} & -A^{-1}b \\
0 & 1
\end{bmatrix}
\]

\[
L^{-T} = \begin{bmatrix}
A^{-T} & 0_{3\times1} \\
(-A^{-1}b)^T & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
a' \\
b' \\
c' \\
d'
\end{bmatrix} = \begin{bmatrix}
a \\
b \\
c \\
d
\end{bmatrix}
\]

That’s why we omitted “inverse transpose” on normal in an earlier slide around page 12.

For \( A \) being a rotation (such as in the case of in *view matrix* transforming between world and camera)

\[
A^T = A^{-1}
\]

\[
A^{-T} = A
\]

That’s why we omitted “inverse transpose” on normal in an earlier slide around page 12.

*The reflection model should satisfy frame invariance:*

- For any *rotation matrix* \( Q \in \mathbb{R}^{3\times3} \)

\[
C(v, n, l) = C(Qv, Qn, Ql)
\]

- So using either world coordinate or camera coordinate will give the same result.
Flat, Gouraud, and Phong shadings

- Introduction
- Reflection model
- Phong reflection model
- Getting $v$, $n$, $I$ correctly
- Gouraud vs Phong shadings
Types of shading

- Per triangle
- Per vertex
- Per pixel
Per-triangle shading

- Also known as *flat shading*
- Evaluate shading once per triangle, based on normal vector

  **Advantage**
  
  ▶ Fast

  **Disadvantage**
  
  ▶ Faceted appearance
Per-vertex shading

• Also known as **Gouraud shading** (Henri Gouraud 1971)

• Compute color per vertex, then interpolate the result across triangles

• Advantage
  ▶ Fast
  ▶ Smoother surface appearance than flat shading

• Disadvantage
  ▶ Problem with highlights
Per-pixel shading

- Also known as **Phong interpolation** (not to be confused with Phong’s illumination model)
  - Let the rasterizer interpolate normals (instead of colors) across triangle
  - Illumination evaluated at each fragment
  - Simulates shading with normals of a curved surface
- Advantage
  - High rendering quality
- Disadvantage
  - Slightly slower (not really)
- We always use per-pixel shading in CSE167
Per-pixel shading

FLAT SHADING
GOURAUD SHADING
PHONG SHADING
#version 330 core

in vec4 position; // raw position in the model coord
in vec3 normal;   // raw normal in the model coord
uniform mat4 modelview; // from model coord to eye coord
uniform mat4 view;      // from world coord to eye coord

// Material parameters
uniform vec4 ambient;
uniform vec4 diffuse;
uniform vec4 specular;
uniform vec4 emission;
uniform float shininess;

// Light source parameters
const int maximal_allowed_lights = 10;
uniform bool enablelighting;
uniform int nlights;
uniform vec4 lightpositions[maximal_allowed_lights];
uniform vec4 lightcolors[maximal_allowed_lights];

// Output the frag color
out vec4 fragColor;

void main (void){
  // HW3: You will compute the lighting here.
}

\[
R = E + \sum_j L_j (C_{ambient} + C_{diffuse} \max(n \cdot l_j, 0) + C_{specular} [\max(n \cdot h_j, 0)]^\gamma)
\]

- Make sure all positions & vectors are respecting a common coordinate system (world or camera frame)
- Are normal vectors transformed correctly?
- Light positions might be at infinity (w coord = 0)
- Useful way to debug is to set pixel color by variables you want to visualize.