Topic 9:

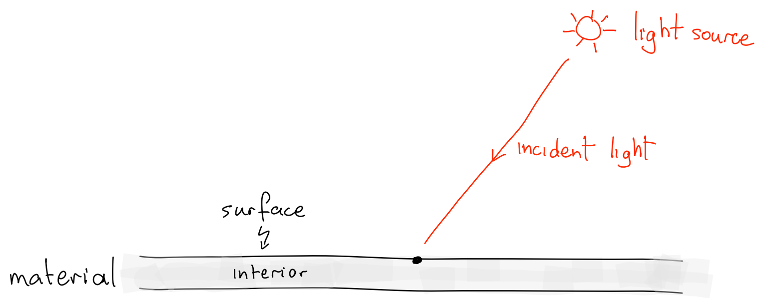
Lighting & Reflection models

- Lighting & reflection
- The Phong reflection model
    - diffuse component
    - ambient component
    - specular component

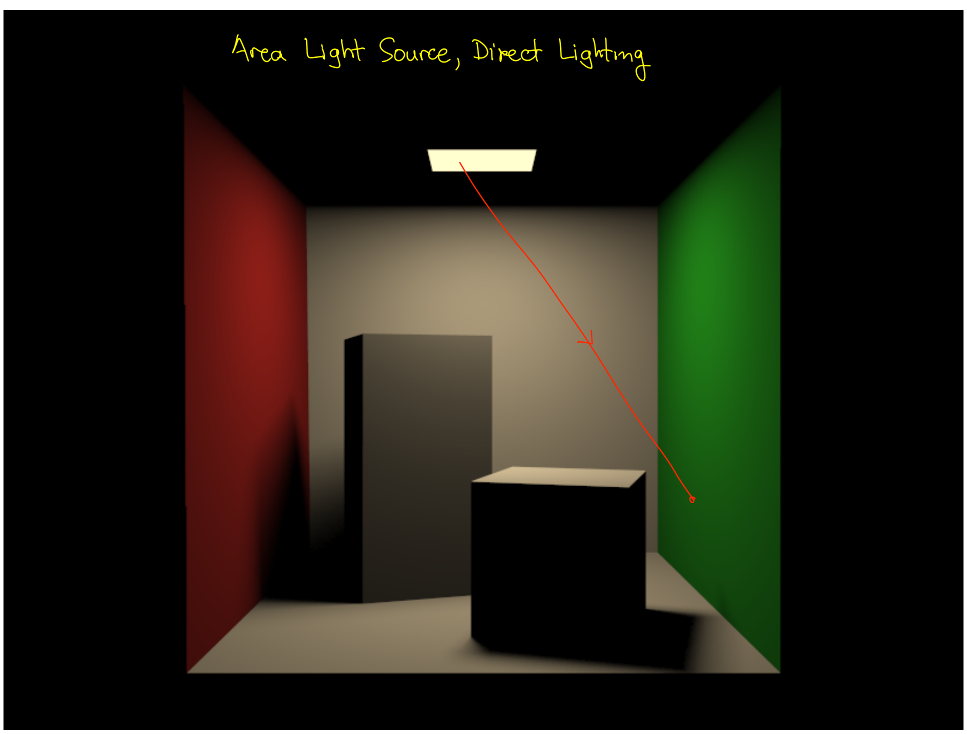

Ng et al, SIGGRAPH'04


Ng et al, SIGGRAPH'04

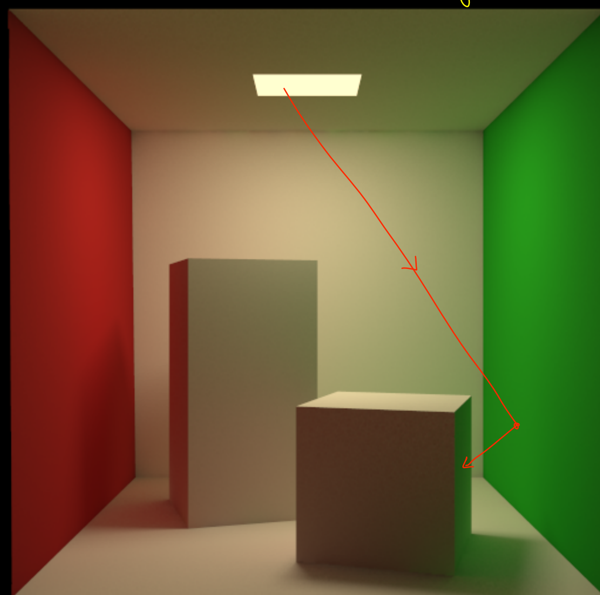
Light Sources



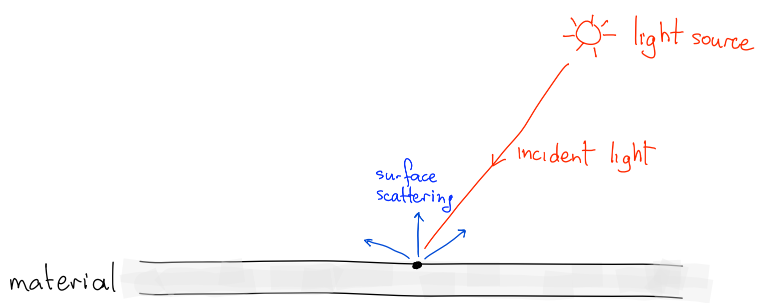
Main sources of light:
- point source
- distant source (spotlight)
- extended source (aka area light source)
- secondary reflection



Area Light Source, Indirect Lighting



Modeling Reflection: Diffuse Reflection

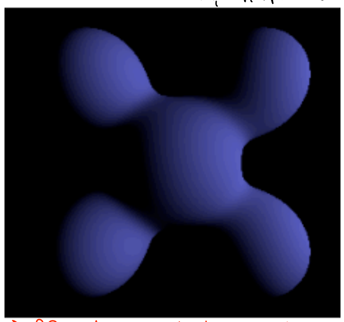


Diffuse reflection

- Represents "matte" component of reflected light
- Usually caused by "rough" surfaces (clay, eggshell, etc)

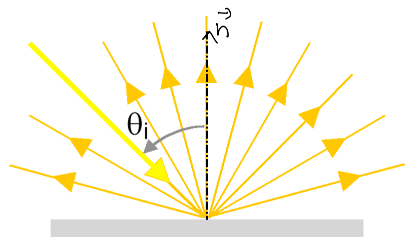
Modeling Reflection: Diffuse Reflection

Brad Smith, Wikipedia



Diffusely-shaded object

θ_i = angle of incidence

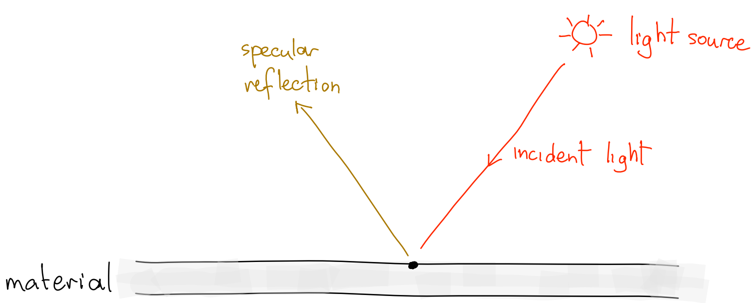


Panjasan, Wikipedia

Diffuse reflection

- Represents "matte" component of reflected light
- Usually caused by "rough" surfaces (clay, eggshell, etc)

Modeling Reflection: Specular Reflection



Specular reflection:

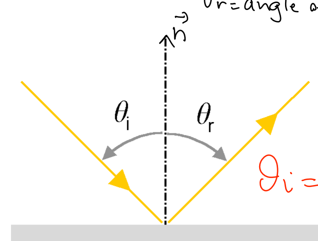
- Represents shiny component of reflected light
- Caused by mirror-like reflection off of smooth or polished surfaces (plastics, polished metal, etc)

Modeling Reflection: Specular Reflection

Romeiro et al, ECCV'08

mirror-like sphere

θ_i = angle of incidence
θ_r = angle of reflection



$\theta_i = \theta_r$

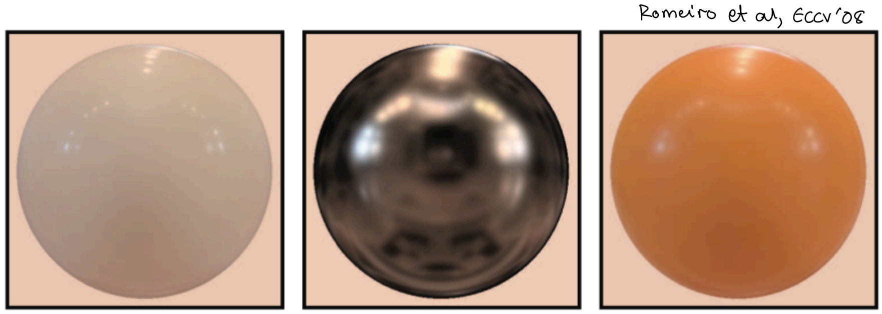
Panjasan, Wikipedia

Specular reflection:

- Represents shiny component of reflected light
- Caused by mirror-like reflection off of smooth or polished surfaces (plastics, polished metal, etc)

Modeling Reflection: Specular Reflection

Romeiro et al, ECCV'08

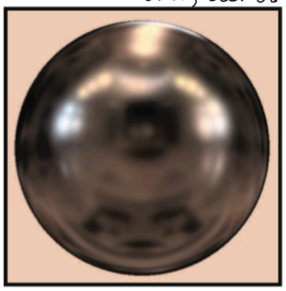


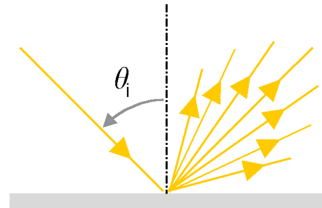
Specular reflection:

- Represents shiny component of reflected light
- Caused by mirror-like reflection off of smooth or polished surfaces (plastics, polished metal, etc)

Modeling Reflection: Specular Reflection

Romeiro et al, ECCV'08





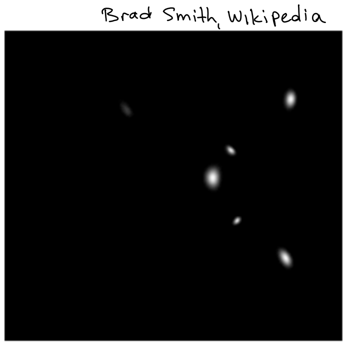
Panjasan, Wikipedia

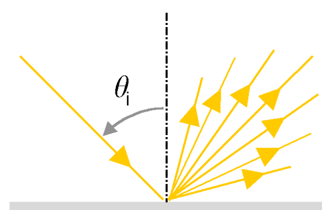
Specular reflection:

- Represents shiny component of reflected light
- Caused by mirror-like reflection off of smooth or polished surfaces (plastics, polished metal, etc)

Modeling Reflection: Specular Reflection

Brad Smith, Wikipedia



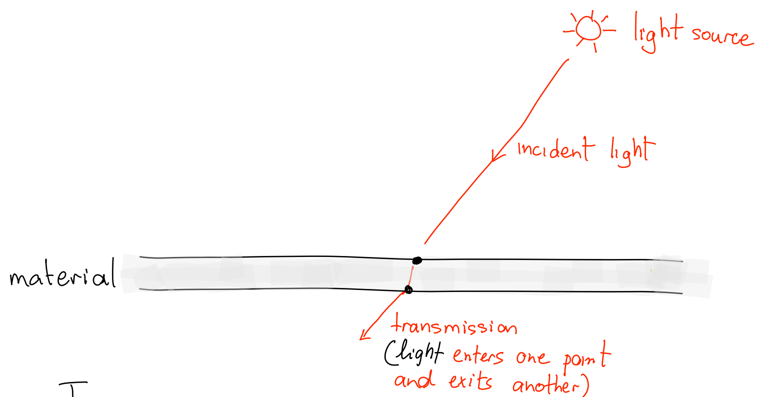


Panjasan, Wikipedia

Specular reflection:

- Represents shiny component of reflected light
- Caused by mirror-like reflection off of smooth or polished surfaces (plastics, polished metal, etc)

Modeling Reflection: Transmission

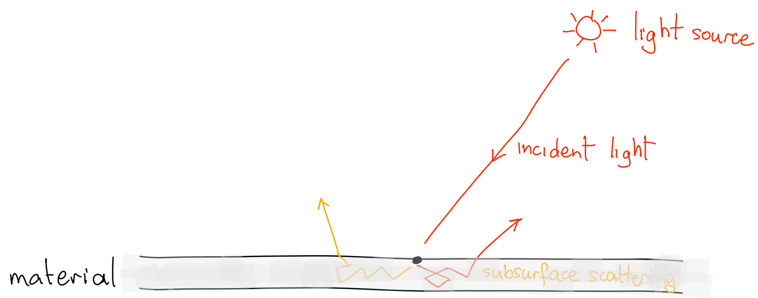


Transmission:

- Caused by materials that are not perfectly opaque
- Examples include glass, water and translucent materials such as skin

Gu et al, EGSR'07

Modeling Reflection: Sub-surface Scattering



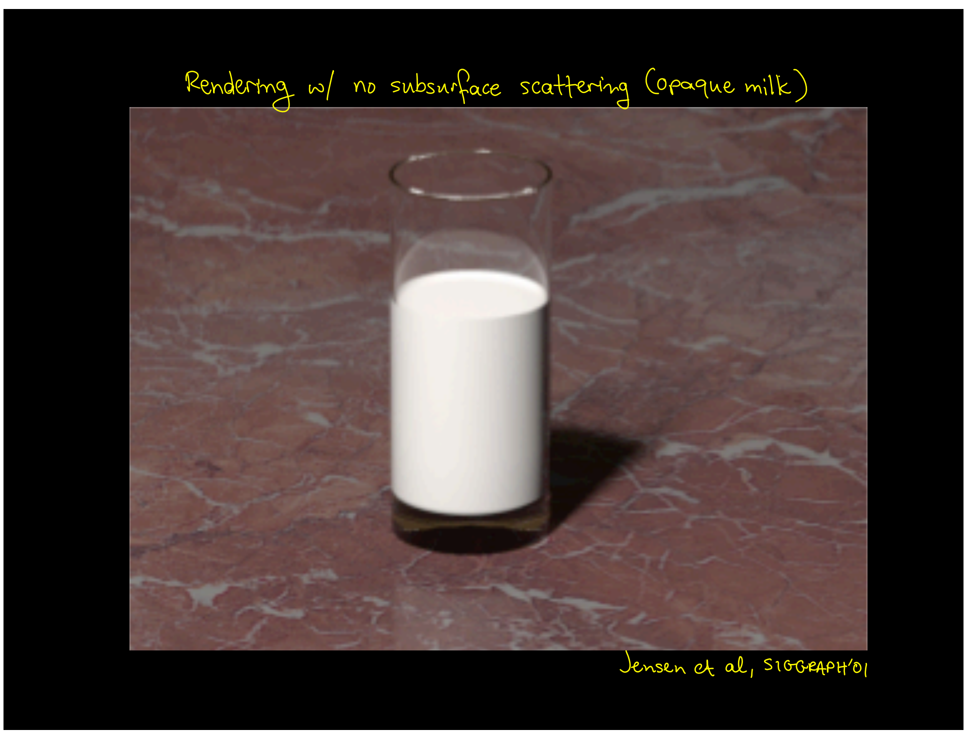
Subsurface scattering:

- Represents the component of reflected light that scatters in the material's interior (after transmission) before exiting again
- Examples include skin, milk, fog, etc.

Rendering w/ no subsurface scattering (opaque skin)

Jensen et al, SIGGRAPH'01


Rendering with subsurface scattering (transluscent skin)
Jensen et al, SIGGRAPH'01


Rendering w/ no subsurface scattering (opaque milk)
Jensen et al, SIGGRAPH'01

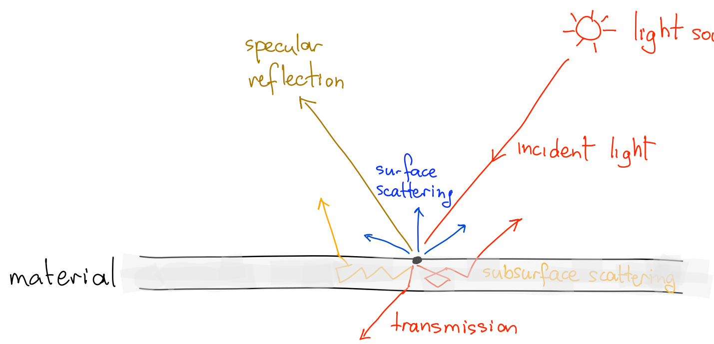
Rendering with subsurface scattering (full milk)

Jensen et al, SIGGRAPH'01

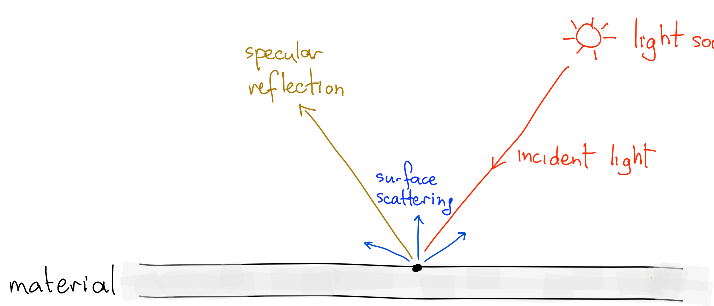
Rendering with subsurface scattering (skim milk)

Jensen et al, SIGGRAPH'01

The Common Modes of "Light Transport"



The Phong Reflectance Model

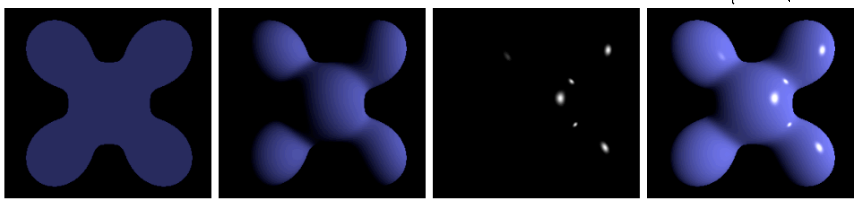


Phong model: A simple, computationally-efficient model that has 3 components:

- Diffuse
- Ambient
- Specular

The Phong Reflectance Model

Brad Smith, Wikipedia



Ambient + Diffuse + Specular = Phong Reflection

Phong model: A simple, computationally-efficient model that has 3 components:

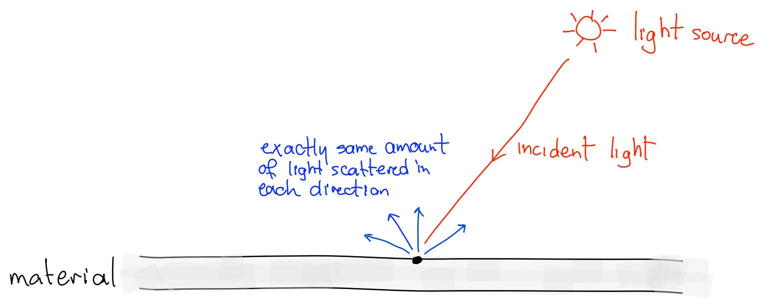
- Diffuse
- Ambient
- Specular

Topic 9:

Lighting & Reflection models

- Lighting & reflection
- The Phong reflection model
  - diffuse component
  - ambient component
  - specular component

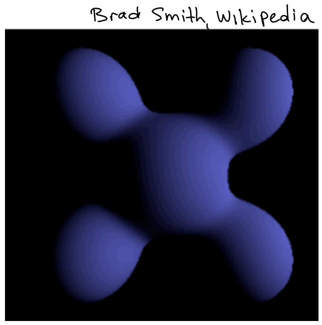
Phong Reflection: The Diffuse Component



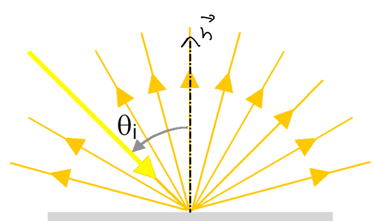
- A diffuse point looks the same from all viewing positions
- Simplest case: a single, point light source

Phong Reflection: The Diffuse Component

Brad Smith, Wikipedia



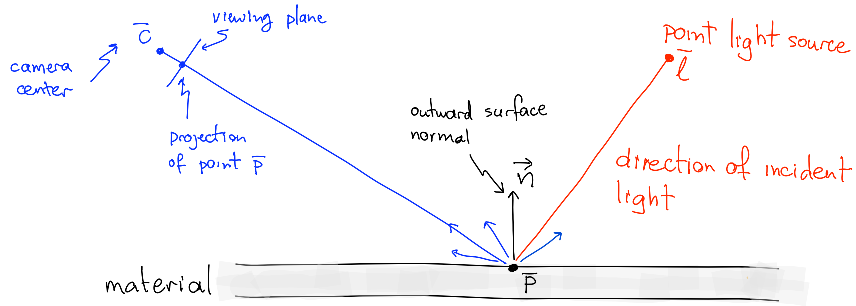
θ_i = angle of incidence



Panjasan, Wikipedia

- A diffuse point looks the same from all viewing positions
- Simplest case: a single, point light source

The Diffuse Component: Basic Equation



- A diffuse point looks the same from all viewing positions
- Simplest case: a single, point light source

fraction of light reflected

outward unit surface normal

$$I_{\bar p} = r_d \cdot I \cdot \max\left(0, \vec s \cdot \vec n\right)$$

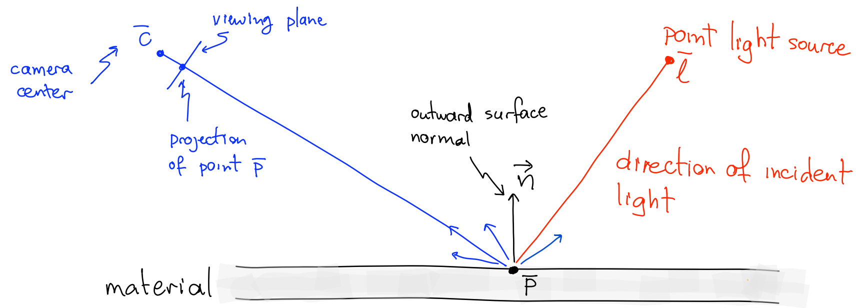
intensity at projection of $\bar p$

intensity of source

direction of light source

$$\vec s = \frac{\bar l - \bar p}{\|\bar l - \bar p\|}$$

The Diffuse Component: Basic Equation



- A diffuse point looks the same from all viewing positions

outward unit surface normal

$$I_{\bar p} = \overbrace{r_d \cdot I \cdot \max\left(0, \vec s \cdot \vec n\right)}^{\text{independent of } \bar c}$$

intensity at projection of $\bar p$

direction of light source

$$\vec s = \frac{\bar l - \bar p}{\|\bar l - \bar p\|}$$

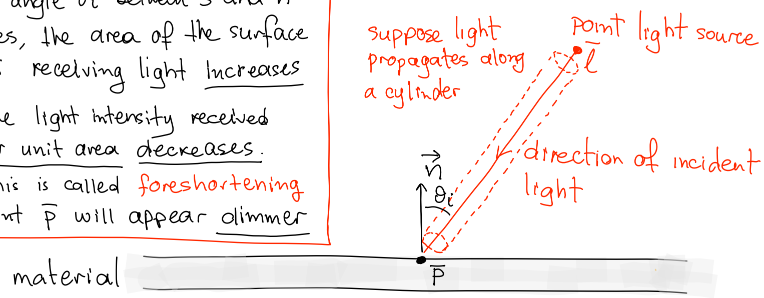
The Diffuse Component: Foreshortening

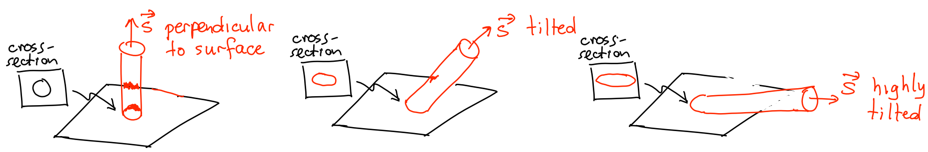
As the angle θ_i between $\vec{s}$ and $\vec{n}$ increases, the area of the surface around $\bar{p}$ receiving light increases

$\Rightarrow$ the light intensity received per unit area decreases.

this is called foreshortening

$\Rightarrow$ point $\bar{p}$ will appear dimmer





$$I_{\bar{p}} = r_d \cdot I \cdot \max\left(0, \underbrace{\vec{s} \cdot \vec{n}}_{\text{accounts for dimming due to foreshortening}}\right)$$

The Diffuse Component: Foreshortening

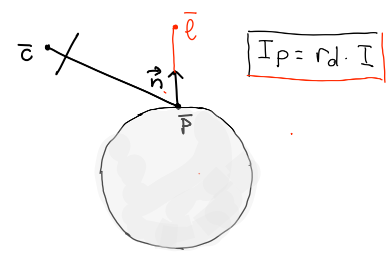
As the angle θ_i between $\vec{s}$ and $\vec{n}$ increases, the area of the surface around $\bar{p}$ receiving light increases

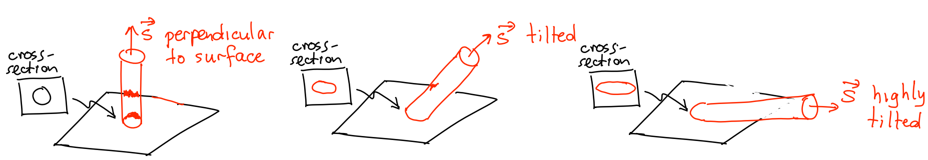
$\Rightarrow$ the light intensity received per unit area decreases.

this is called foreshortening

$\Rightarrow$ point $\bar{p}$ will appear dimmer

Q: What is the intensity at $\bar{p}$'s projection?





$$I_{\bar{p}} = r_d \cdot I \cdot \max\left(0, \underbrace{\vec{s} \cdot \vec{n}}_{\text{accounts for dimming due to foreshortening}}\right)$$

The Diffuse Component: Foreshortening

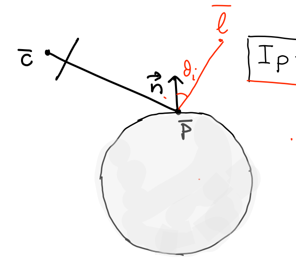
As the angle θ_i between $\vec{s}$ and $\vec{n}$ increases, the area of the surface around $\bar{p}$ receiving light increases

$\Rightarrow$ the light intensity received per unit area decreases.

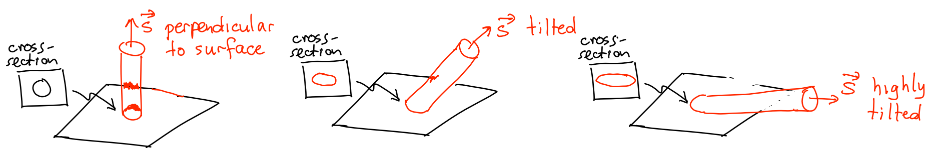
this is called foreshortening

$\Rightarrow$ point $\bar{p}$ will appear dimmer

Q: What is the intensity at $\bar{p}$'s projection?



$\boxed{I_p = r_d\, I \cos\theta_i}$



$$I_{\bar{p}} = r_d \cdot I \cdot \max\big(0, \underbrace{\vec{s}\cdot\vec{n}}_{\text{accounts for dimming due to foreshortening}}\big)$$

The Diffuse Component: Foreshortening

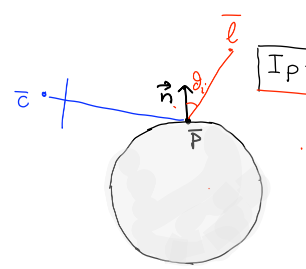
As the angle θ_i between $\vec{s}$ and $\vec{n}$ increases, the area of the surface around $\bar{p}$ receiving light increases

$\Rightarrow$ the light intensity received per unit area decreases.

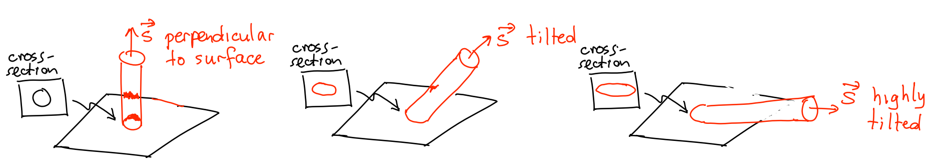
this is called foreshortening

$\Rightarrow$ point $\bar{p}$ will appear dimmer

Q: What is the intensity at $\bar{p}$'s projection?



$\boxed{I_p = r_d\, I \cos\theta_i}$



$$I_{\bar{p}} = r_d \cdot I \cdot \max\big(0, \underbrace{\vec{s}\cdot\vec{n}}_{\text{accounts for dimming due to foreshortening}}\big)$$

The Diffuse Component: Self-Shadowing

As the angle θ_i between $\vec{s}$ and $\vec{n}$ increases, the area of the surface around $\bar{p}$ receiving light increases

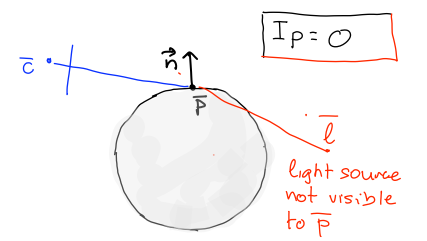
$\Rightarrow$ the light intensity received per unit area decreases.

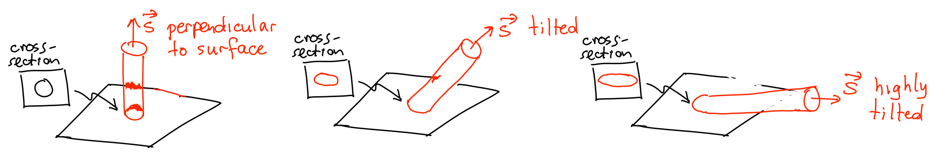
this is called foreshortening

$\Rightarrow$ point $\bar{p}$ will appear dimmer

Q: What is the intensity at $\bar{p}$'s projection?

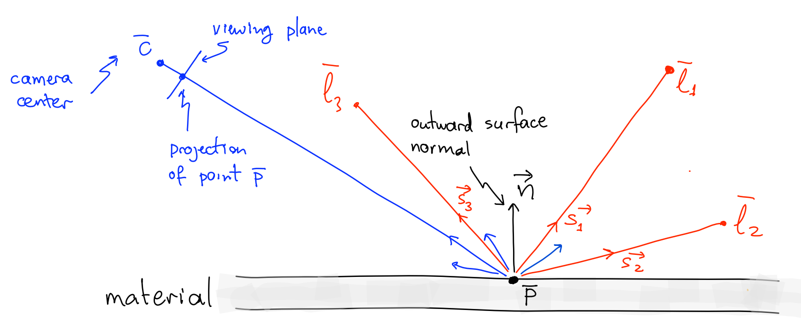
$\boxed{I_p = 0}$





$$I_{\bar{p}} = r_d \cdot I \cdot \underbrace{\max}_{\text{accounts for cases where light source not visible}}\left(0, \vec{s} \cdot \vec{n}\right)$$

The Diffuse Component: Multiple Lights



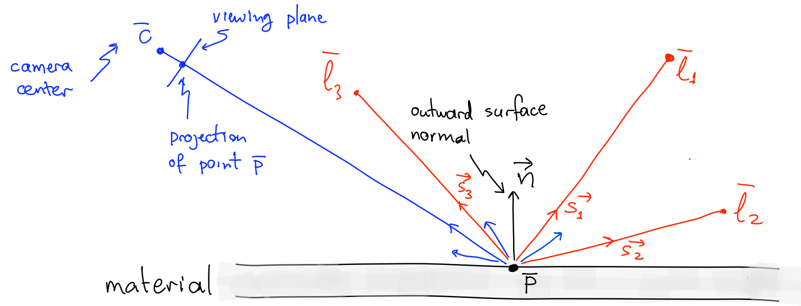
- A diffuse point looks the same from all viewing positions
- When the scene is illuminated by many point sources, we just sum their contributions to the diffuse component

$$I_{\bar{p}} = r_d \sum_i I_i \max\left(0, \vec{s}_i \cdot \vec{n}\right)$$

intensity at projection of $\bar{p}$

intensity of source i

The Diffuse Component: Incorporating Color



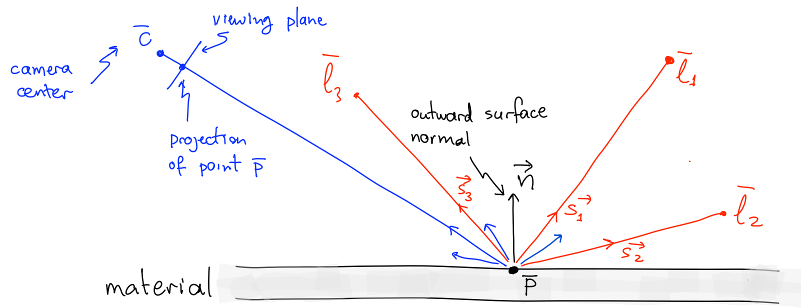
- A diffuse point looks the same from all viewing positions
- Colored sources and colored objects are handled by considering the RGB components of each color seperately

$$I_{\bar{p},q} = r_{d,q} \sum_i I_{i,q} \max(0, \vec{s}_i \cdot \vec{n}) \qquad q = R, G, B$$

intensity of color component q at projection of $\bar{p}$

intensity of color component q for light source i

The Diffuse Component: General Equation



Putting it all together:

$$\boxed{I_{\bar{p},q} = r_{d,q} \sum_i I_{i,q} \max(0, \vec{s}_i \cdot \vec{n})}$$

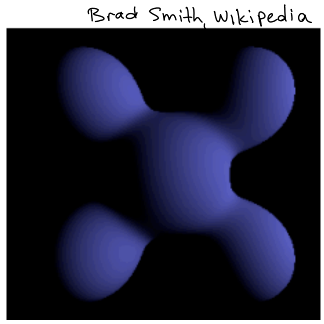
Topic 9:

Lighting & Reflection models

- Lighting & reflection
- The Phong reflection model
  - diffuse component
  - ambient component
  - specular component

Phong Reflection: Ambient Component

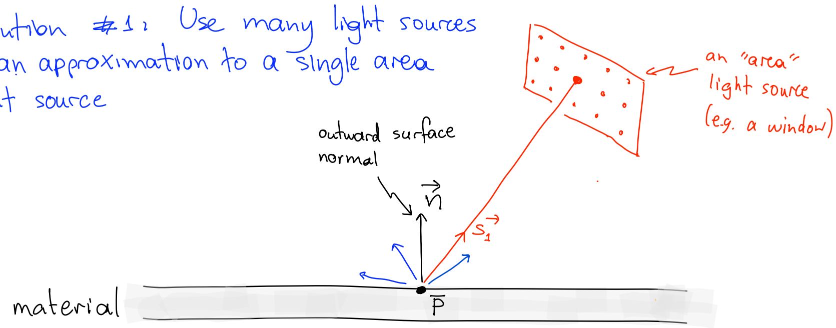
Brad Smith, Wikipedia



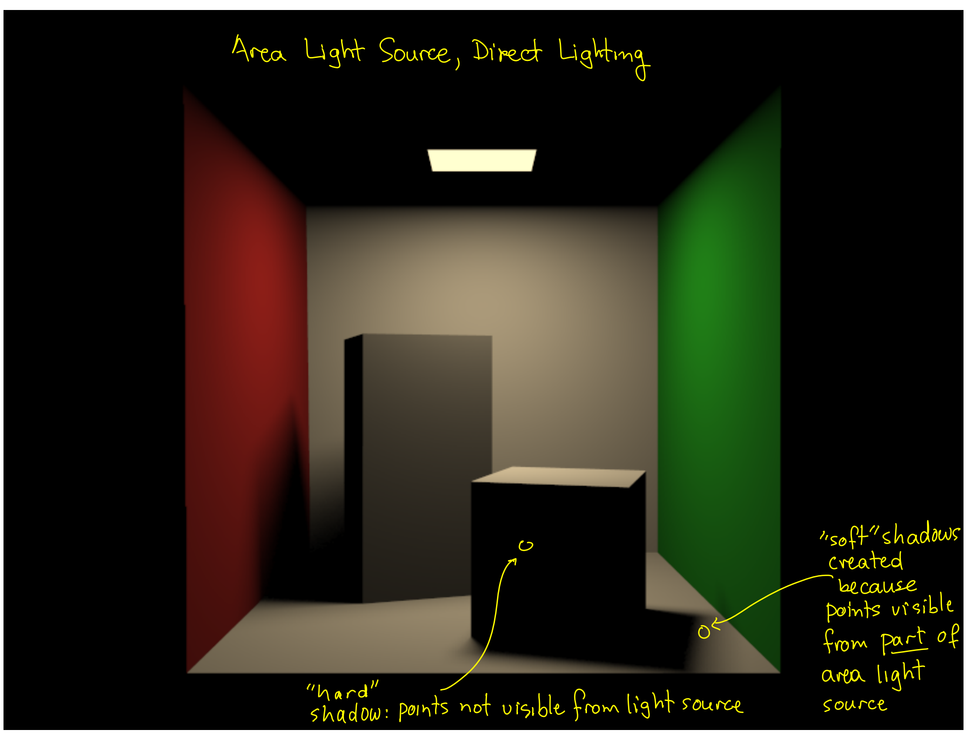
- Diffuse reflectance with a single point light source produces strong shadows
- Surface patches with $\vec{s}\cdot\vec{n}<0$ are perfectly black
  $\Rightarrow$ looks unnatural

Phong Reflection: Ambient Component

Solution #1: Use many light sources as an approximation to a single area light source



- Diffuse reflectance with a single point light source produces strong shadows
- Surface patches with $\vec{s} \cdot \vec{n} < 0$ are perfectly black
  $\Rightarrow$ looks unnatural



Phong Reflection: Ambient Component

Solution #2: (Simpler) Use an "ambient" term that is independent of any light source or surface normal

This term is not meaninfull in terms of physics but improves appearance over pure diffuse reflection

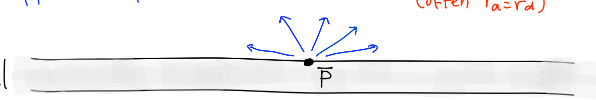
can also have 3 such eqs for R,G,B components

$$I_{\bar{p}} = r_a \cdot I_a$$

ambient reflection coefficient (often $r_a = r_d$)

intensity of ambient illumination

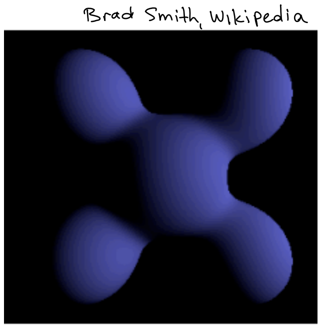
material



- Diffuse reflectance with a single point light source produces strong shadows
- Surface patches with $\vec{s} \cdot \vec{n} < 0$ are perfectly black
  $\Rightarrow$ looks unnatural

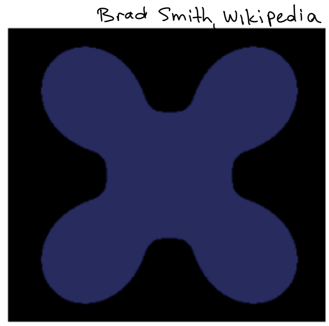
Phong Reflection: Ambient Component

Brad Smith, Wikipedia



Diffuse

Brad Smith, Wikipedia



Ambient

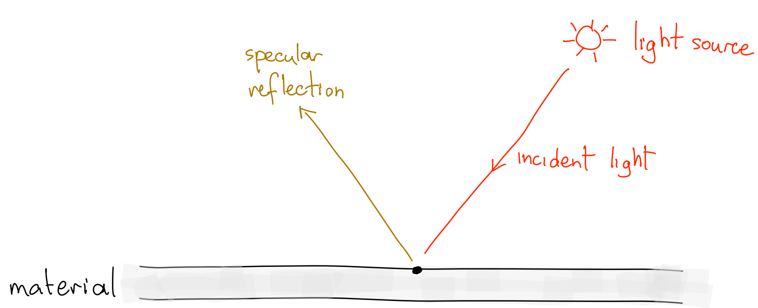
- Diffuse reflectance with a single point light source produces strong shadows
- Surface patches with $\vec{s} \cdot \vec{n} < 0$ are perfectly black
  $\Rightarrow$ looks unnatural

Topic 9:

Lighting & Reflection models

- Lighting & reflection
- The Phong reflection model
  - diffuse component
  - ambient component
  - specular component

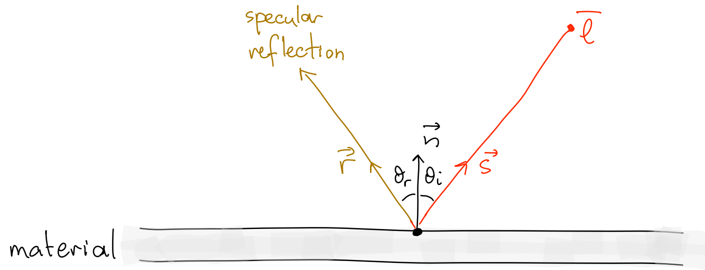
Phong Reflection: The Specular Component



Specular reflection:

- Represents shiny component of reflected light
- Caused by mirror-like reflection off of smooth or polished surfaces (plastics, polished metal, etc)

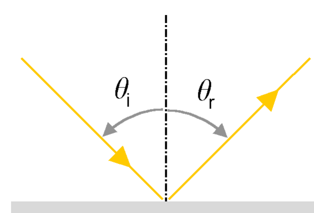
The Ideal Specular Component



- Idea: For each incident direction $\vec{s}$
  there is one emittant direction $\vec{r}$
- It is an idealization of a mirror:
  $$\underbrace{\text{angle}(\vec{n},\vec{s})}_{\theta_i} = \underbrace{\text{angle}(\vec{n},\vec{r})}_{\theta_r}$$

The Ideal Specular Component

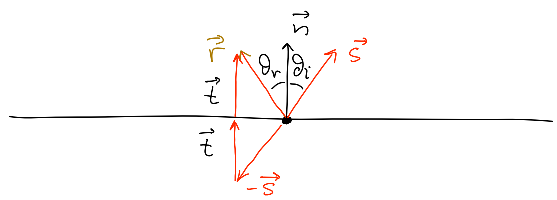
Romeiro et al, ECCV'08



Panjasan, Wikipedia

- Idea: For each incident direction $\vec{s}$
  there is one emittant direction $\vec{r}$
- It is an idealization of a mirror:
  $$\text{angle}(\vec{n},\vec{s}) = \text{angle}(\vec{n},\vec{r})$$
- Q: How can we express $\vec{r}$ in terms of $\vec{n},\vec{s}$?

The Ideal Specular Component



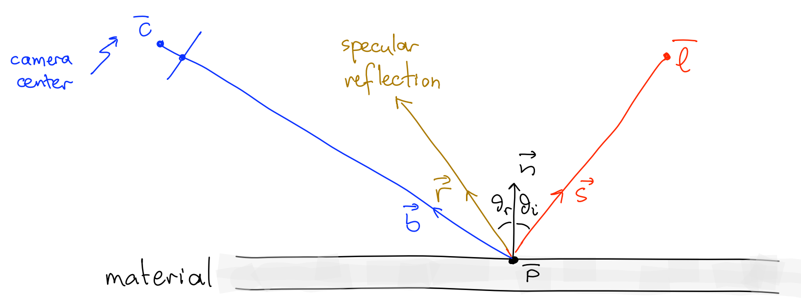
$\vec{r} = -\vec{s} + 2\vec{t}$

$\vec{t}$ = projection of vector $\vec{s}$ onto vector $\vec{n}$
$= (\vec{n}\cdot\vec{s})\,\vec{n}$

$\Rightarrow \boxed{\vec{r} = -\vec{s} + 2(\vec{n}\cdot\vec{s})\,\vec{n}}$

. Q: How can we express $\vec{r}$ in terms of $\vec{n}, \vec{s}$?

The Ideal Specular Component

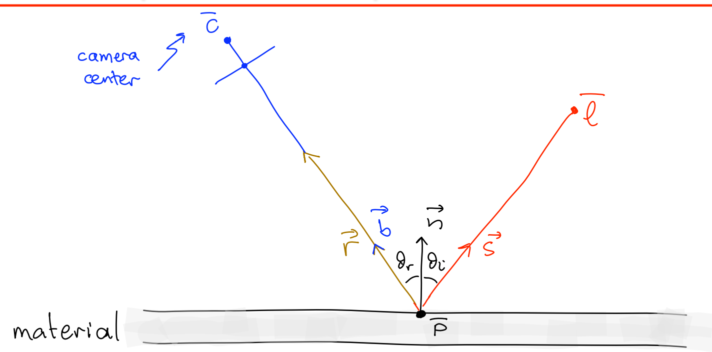


Ideal specular reflection term:

$$I = \underbrace{r_s}_{\text{specular reflection coefficient}} \; \underbrace{I_s}_{\text{intensity of specular light source}} \; \delta\Big(\overbrace{\vec{r}\cdot\underbrace{\vec{b}}_{\text{unit vector in camera direction}} - 1}^{\text{is 1 if and only if camera is along vector } \vec{r}}\Big) \quad \text{where} \quad \delta(x) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

$\vec{b} = \dfrac{\bar{c} - \bar{p}}{\|\bar{c} - \bar{p}\|}$

The Ideal Specular Component



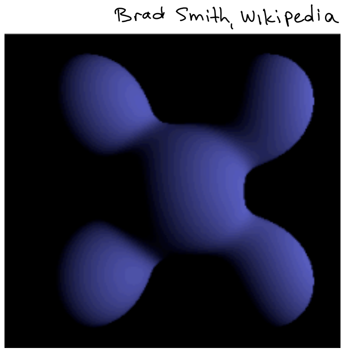
Ideal specular reflection term:

$$I = \underbrace{r_s}_{\text{specular reflection coefficient}} \; \underbrace{I_s}_{\text{intensity of specular light source}} \; \delta\overbrace{\left(\vec{r}\cdot\underbrace{\vec{b}}_{\text{unit vector in camera direction}} - 1\right)}^{\text{is 1 if and only if camera is along vector } \vec{r}} \quad \text{where} \quad \delta(x) = \begin{cases} 1 & \text{if } x=0 \\ 0 & \text{otherwise} \end{cases}$$

$$\vec{b} = \frac{\bar{c}-\bar{p}}{\|\bar{c}-\bar{p}\|}$$

The Ideal Specular Component

Brad Smith, Wikipedia



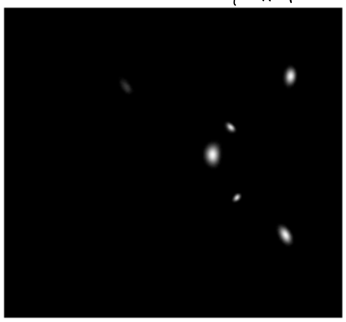
Ideal specular reflection term:

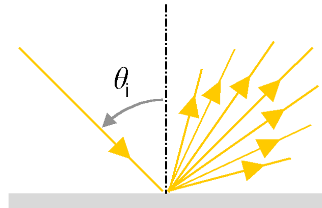
$$I = \underbrace{r_s}_{\text{specular reflection coefficient}} \; \underbrace{I_s}_{\text{intensity of specular light source}} \; \delta\overbrace{\left(\vec{r}\cdot\underbrace{\vec{b}}_{\text{unit vector in camera direction}} - 1\right)}^{\text{is 1 if and only if camera is along vector } \vec{r}} \quad \text{where} \quad \delta(x) = \begin{cases} 1 & \text{if } x=0 \\ 0 & \text{otherwise} \end{cases}$$

$$\vec{b} = \frac{\bar{c}-\bar{p}}{\|\bar{c}-\bar{p}\|}$$

Phong Reflection: Off-Specular Reflection

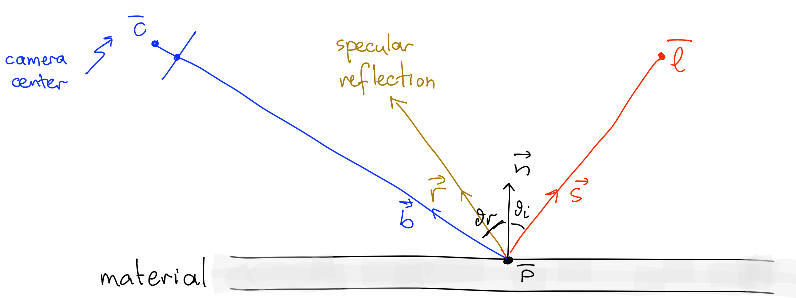
Brad Smith, Wikipedia





Panjasan, Wikipedia

The Specular Component: Basic Equation



In reality, most specular surfaces reflect light into directions near the perfect mirror direction (eg. highlights in plastics, metals)

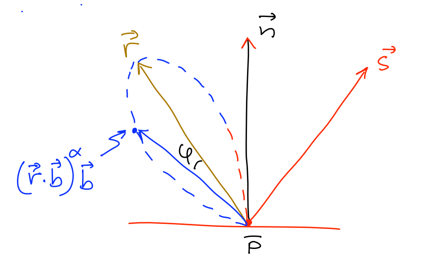
⇒ replace delta function by a cosine power:

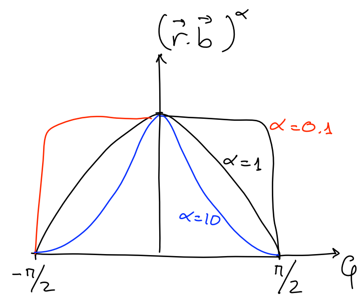
$$I = r_s I_s \max(0, \underbrace{\vec{r}\cdot\vec{b}}_{=1 \text{ when } \vec{r}=\vec{b}})^{\alpha}$$

← when $\alpha \to \infty$ term approaches ideal specular reflection term

The Specular Component: Visualization

φ = deviation from ideal specular reflection angle



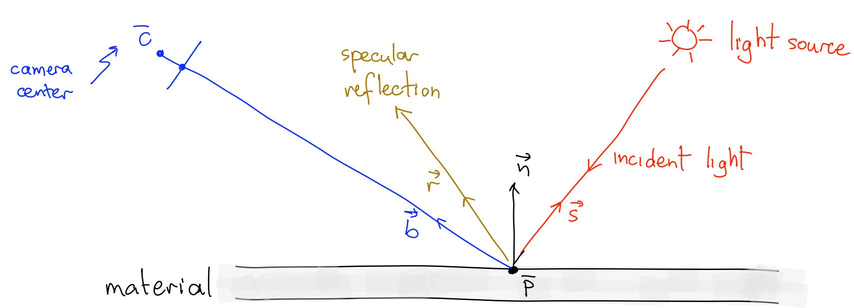


The length of vector $(\vec{r}\cdot\vec{b})^\alpha \vec{b}$ represents the contribution of the specular term when the camera is along $\vec{b}$

$$I = r_s I_s \max(0, \underbrace{\vec{r}\cdot\vec{b}}_{=1 \text{ when } \vec{r}=\vec{b}})^\alpha$$

α ← when $\alpha \to \infty$ term approaches ideal specular reflection term

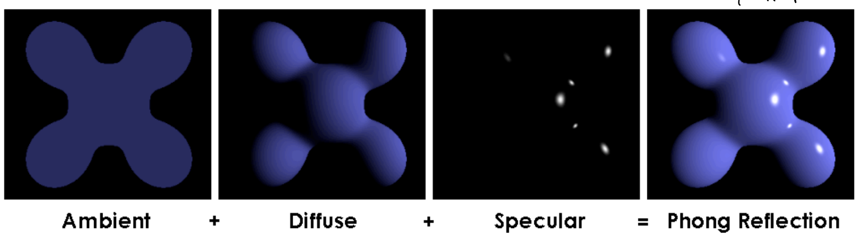
Phong Reflection: The General Equation



$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^\alpha}_{\text{specular}}$$

Phong Reflection: The General Equation

Brad Smith, Wikipedia



$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

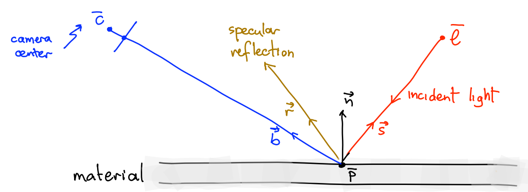
Topic 10:

Shading

- Introduction to Shading
- Flat Shading
- Interpolative Shading
  - Gouraud shading
  - Phong shading
  - Triangle scan-conversion with shading

Shading: Motivation

- Suppose we know how to compute the appearance of a point
- How do we shade a whole polygonal mesh?



Answer

assign intensities to every pixel at the mesh's projection in accordance with Phong reflection model

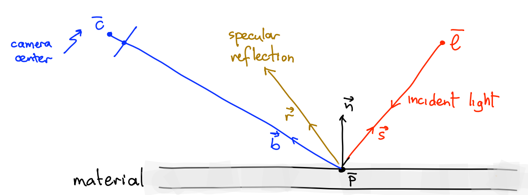
$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

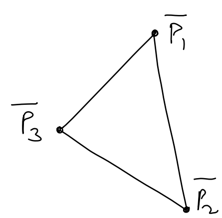
Shading: Motivation

Given

- camera center $(\bar{c})$
- light source position $(\bar{\ell})$
- intensity of ambient, diffuse & specular sources (I_a, I_d, I_s)
- reflection coefficients (r_a, r_d, r_s)
- specular exponent (α)

Shade every pixel in triangle's projection





$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

Shading: Problem Definition

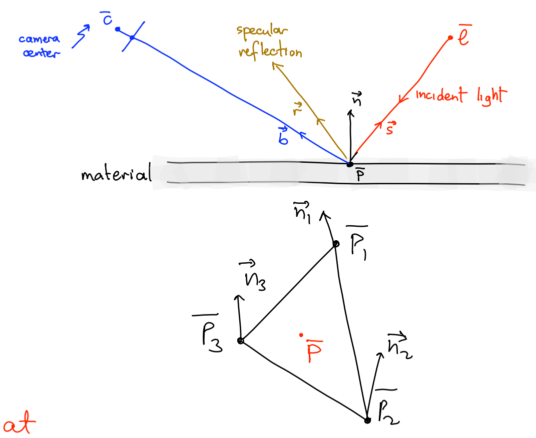
Given

view { • camera center ($\bar{c}$)

scene { • light source position ($\bar{\ell}$)
• intensity of ambient, diffuse & specular sources (I_a, I_d, I_s)

material { • reflection coefficients (r_a, r_d, r_s)
• specular exponent (α)

triangle { • normals at $\bar{p}_1, \bar{p}_2, \bar{p}_3$



Goal

• compute color/intensity at an interior point

$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

Shading: Problem Definition

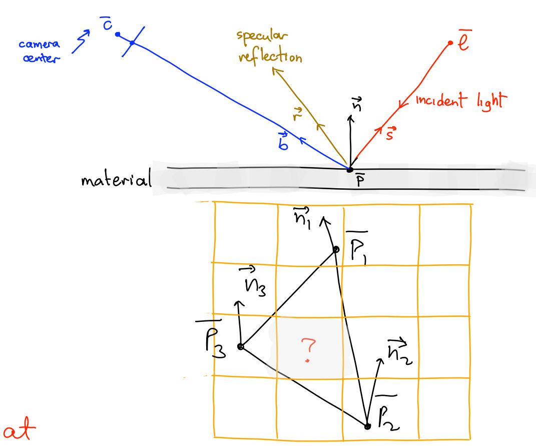
Given

view { • camera center ($\bar{c}$)

scene { • light source position ($\bar{\ell}$)
• intensity of ambient, diffuse & specular sources (I_a, I_d, I_s)

material { • reflection coefficients (r_a, r_d, r_s)
• specular exponent (α)

triangle { • normals at $\bar{p}_1, \bar{p}_2, \bar{p}_3$



Goal

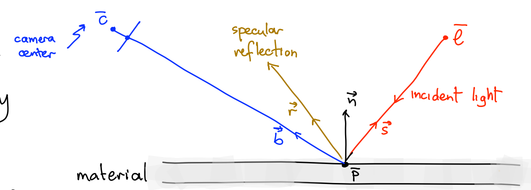
• compute color/intensity at an interior pixel

$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

Basic Approaches to Shading

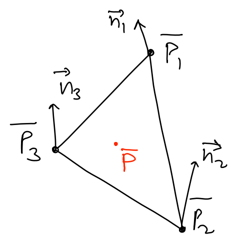
Flat shading

Draw all triangle points $\bar{p}$ with identical color/intensity



Gouraud shading

1. Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex
2. Interpolate the L_i's to get value at $\bar{p}$



Phong shading

1. Interpolate $\vec{b}_i, \vec{n}_i, \vec{s}_i$ to get $\vec{b}, \vec{n}, \vec{s}$ at $\bar{p}$
2. Compute $L(\vec{b}, \vec{n}, \vec{s})$

$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

Topic 10:

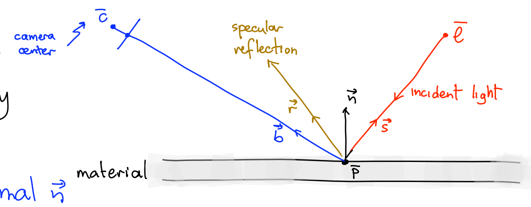
Shading

- Introduction to Shading
- Flat Shading
- Interpolative Shading
  - Gouraud shading
  - Phong shading
  - Triangle scan-conversion with shading

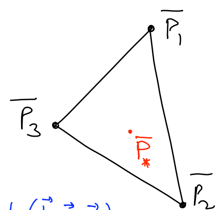
Flat Shading: Main Idea

Flat shading

Draw all triangle points $\bar{p}$ with identical color / intensity



- All points have same normal $\vec{n}$ (i.e. triangle is "flat")
- Phong model applied to center of triangle, $\bar{p}_* = \frac{1}{3}(\bar{p}_1 + \bar{p}_2 + \bar{p}_3)$ (i.e. $\vec{b}, \vec{s}$ computed for $\bar{p}_*$)
- Triangle filled with color / intensity $L(\vec{b}, \vec{n}, \vec{s})$



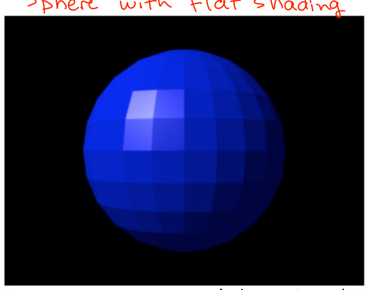
$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

Flat Shading: Main Idea

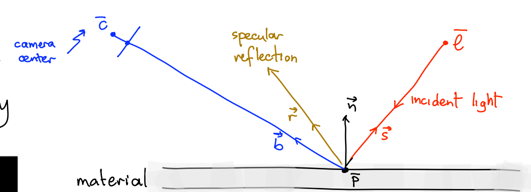
Flat shading

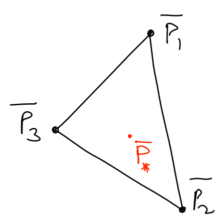
Draw all triangle points $\bar{p}$ with identical color / intensity

Sphere with flat shading



Jalo, wikipedia



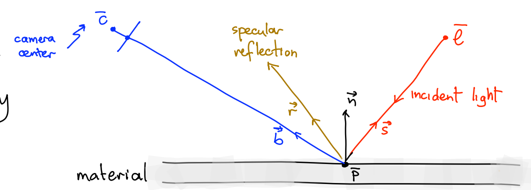


$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

Flat Shading: Key Issues

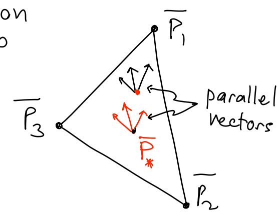
Flat shading

Draw all triangle points $\bar{p}$ with identical color/intensity



Issues

- For large triangles:
  - specular term is poor approximation because highlights should be sharp (often better to drop this term)
  - flat shading essentially assumes a distant light source



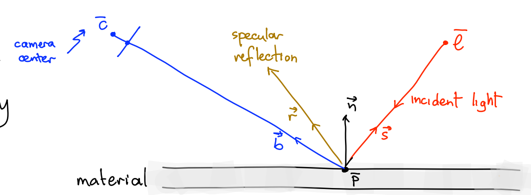
- Triangle boundaries are usually visible (people very sensitive to intensity steps)

$$\underbrace{L(\vec{b},\vec{n},\vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

Flat Shading: Key Issues

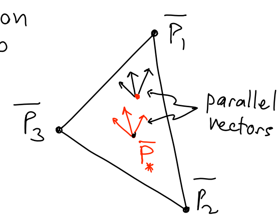
Flat shading

Draw all triangle points $\bar{p}$ with identical color/intensity



Issues

- For large triangles:
  - specular term is poor approximation because highlights should be sharp (often better to drop this term)
  - flat shading essentially assumes a distant light source



- Triangle boundaries are usually visible (people very sensitive to intensity steps)

One solution

Since flat shading treats a triangle as a point, use small triangles!

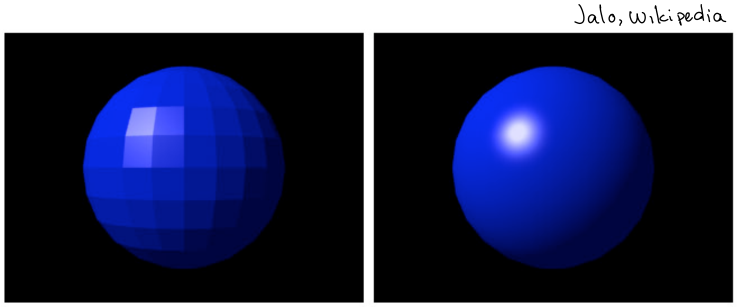
Topic 10:

Shading

- Introduction to Shading
- Flat Shading
- Interpolative Shading
  - Gouraud shading
  - Phong shading
  - Triangle scan-conversion with shading

Interpolated Shading

Jalo, Wikipedia



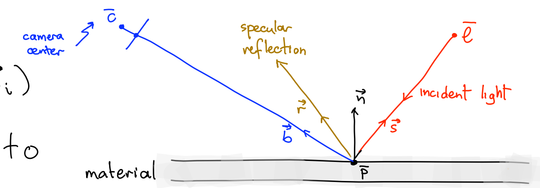
FLAT SHADING PHONG SHADING

Interpolative Shading: Basic Approaches

Gouraud shading –

① Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex

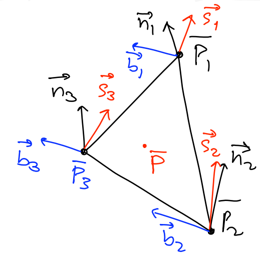
② Interpolate the L_i's to get value at $\bar{p}$



Phong shading –

① Interpolate $\vec{b}_i, \vec{n}_i, \vec{s}_i$ to get $\vec{b}, \vec{n}, \vec{s}$ at $\bar{p}$

② Compute $L(\vec{b}, \vec{n}, \vec{s})$



$$\underbrace{L(\vec{b}_i, \vec{n}_i, \vec{s}_i)}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}_i \cdot \vec{s}_i)}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}_i \cdot \vec{b}_i)^{\alpha}}_{\text{specular}}$$

Topic 10:

Shading

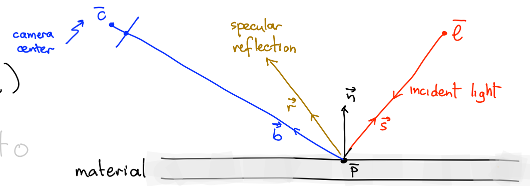
- Introduction to Shading
- Flat Shading
- Interpolative Shading
  - Gouraud shading
  - Phong shading
  - Triangle scan-conversion with shading

Gouraud Shading: Computation at Vertices

Gouraud shading

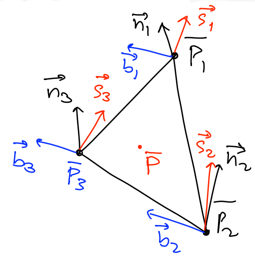
① Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex

② Interpolate the L_i's to get value at $\bar{p}$



Notes

- Vectors $\vec{b}_i, \vec{s}_i$ computed directly from $\bar{p}_i$, $\bar{c}$ and $\bar{\ell}$
- Many possible ways to assign a normal to a vertex



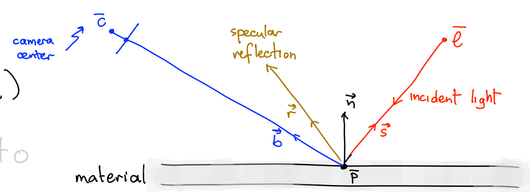
$$\underbrace{L(\vec{b}_i, \vec{n}_i, \vec{s}_i)}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}_i \cdot \vec{s}_i)}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}_i \cdot \vec{b}_i)^{\alpha}}_{\text{specular}}$$

Gouraud Shading: Computation at Vertices

Gouraud shading

① Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex

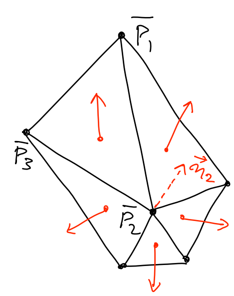
② Interpolate the L_i's to get value at $\bar{p}$



Notes

- Vectors $\vec{b}_i, \vec{s}_i$ computed directly from $\bar{p}_i$, $\bar{c}$ and $\bar{\ell}$
- Many possible ways to assign a normal to a vertex:

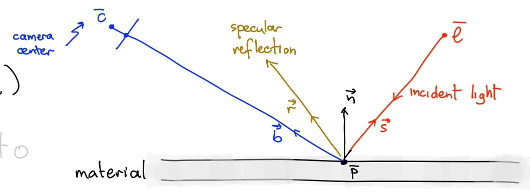
① $\vec{n}_j$ is the average of the normals of all faces that contain vertex $\bar{p}_j$



Gouraud Shading: Computation at Vertices

Gouraud shading

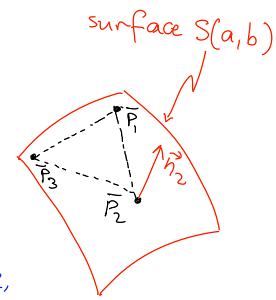
1. Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex
2. Interpolate the L_i's to get value at $\bar{p}$



Notes

- Vectors $\vec{b}_i, \vec{s}_i$ computed directly from $\bar{p}_i$, $\bar{c}$ and $\bar{\ell}$
- Many possible ways to assign a normal to a vertex:

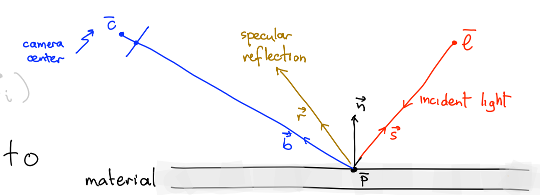
$\vec{n}_j$ is the normal of a point sample on a parametric surface, computed when sampling points to create the original mesh



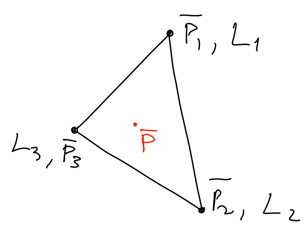
Gouraud Shading: Computation at Pixels

Gouraud shading

1. Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex
2. Interpolate the L_i's to get value at $\bar{p}$



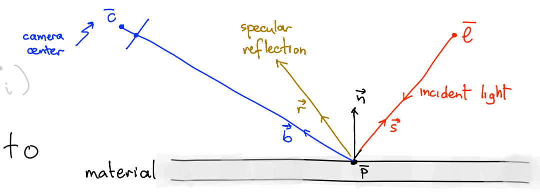
This step is integrated into the standard triangle-filling algorithm discussed in previous lecture



Gouraud Shading: Computation at Pixels

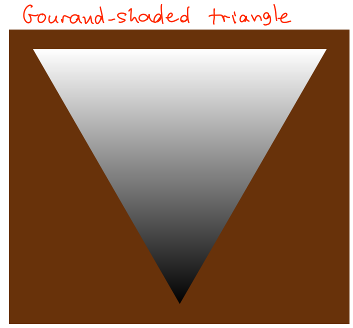
Gouraud shading

1. Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex
2. Interpolate the L_i's to get value at $\bar{p}$



This step is integrated into the standard triangle-filling algorithm discussed in previous lecture

Gourand-shaded triangle

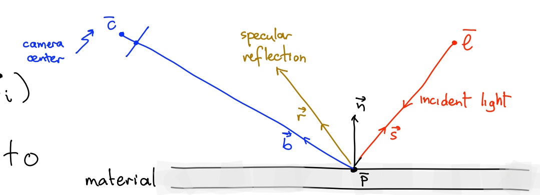


Yzmo, WikiPedia

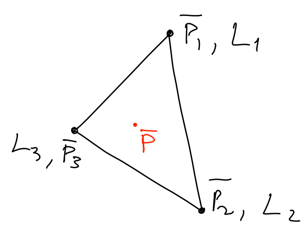
Gouraud Shading: Comparisons

Gouraud shading

1. Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex
2. Interpolate the L_i's to get value at $\bar{p}$



Comparison to flat shading

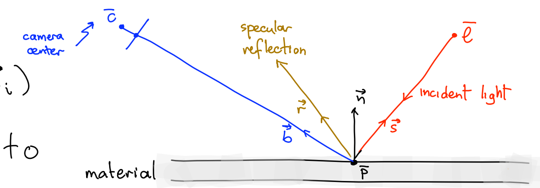


- (+) No visible seams between mesh triangles
- (+) Smooth, visually pleasing intensity variations that "mask" coarse geometry
- (−) Specular highlights still a problem for large triangles (Why?)

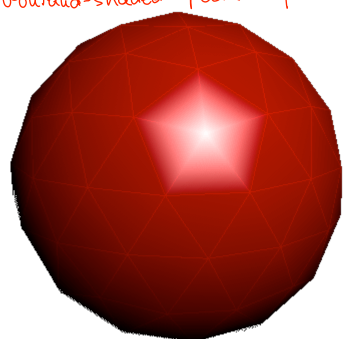
Gouraud Shading: Comparisons

Gouraud shading

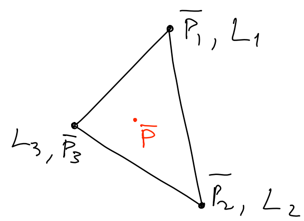
1. Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex
2. Interpolate the L_i's to get value at $\bar{p}$



Gouraud-shaded specular sphere



Jalo, Wikipedia



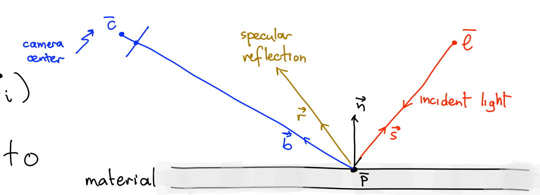
Gouraud Shading: Comparisons

Gouraud shading

1. Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex
2. Interpolate the L_i's to get value at $\bar{p}$:

$$L = \beta L_1 + \gamma L_2 + \epsilon L_3$$

constants determined by interpolation alg.

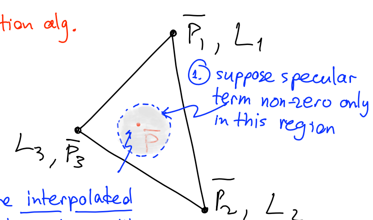


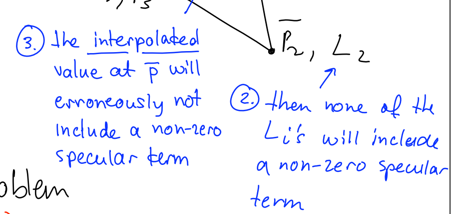
Comparison to flat shading

(+) No visible seams between mesh triangles

(+) Smooth, visually pleasing intensity variations that "mask" coarse geometry

(−) Specular highlights still a problem for large triangles (Why?)





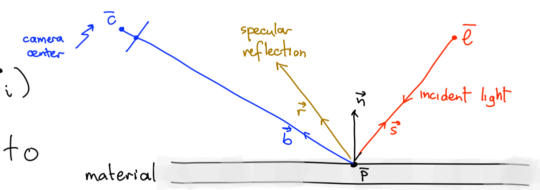
Gouraud Shading: Comparisons

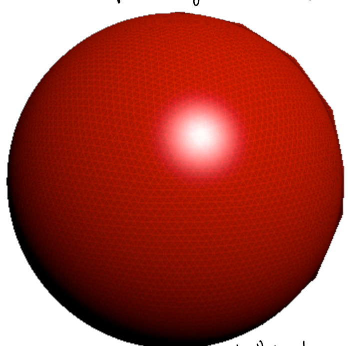
Gouraud shading

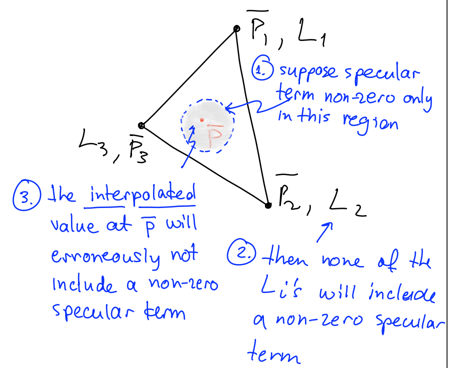
① Compute $L_i = L(\vec{b}_i, \vec{n}_i, \vec{s}_i)$ for each vertex

② Interpolate the L_i's to get value at $\bar{p}$:

$$L = \beta L_1 + \gamma L_2 + \epsilon L_3$$




Jalo, Wikipedia



Topic 10:

Shading

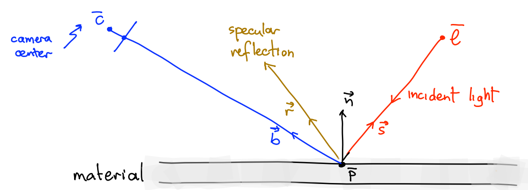
- Introduction to Shading
- Flat Shading
- Interpolative Shading
  - Gouraud shading
  - Phong shading
  - Triangle scan-conversion with shading

Phong Shading: Main Idea

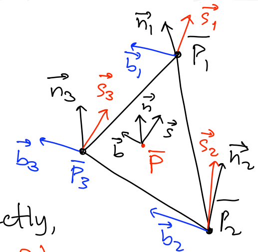
Phong shading –

① Interpolate $\vec{b}_i, \vec{n}_i, \vec{s}_i$
to get $\vec{b}, \vec{n}, \vec{s}$ at $\bar{p}$

② Compute $L(\vec{b}, \vec{n}, \vec{s})$



Comparison to Gouraud shading



(+) Smooth intensity variations as in Gouraud shading

(+) Handles specular highlights correctly, even for large triangles (Why?)

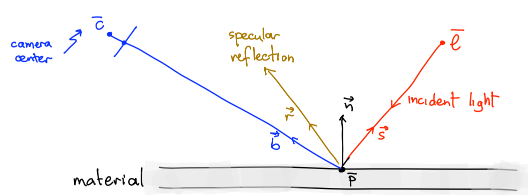
$$\underbrace{L(\vec{b}, \vec{n}, \vec{s})}_{\text{intensity at projection of point } \bar{p}} = \underbrace{r_a I_a}_{\text{ambient}} + \underbrace{r_d I_d \max(0, \vec{n}\cdot\vec{s})}_{\text{diffuse}} + \underbrace{r_s I_s \max(0, \vec{r}\cdot\vec{b})^{\alpha}}_{\text{specular}}$$

Phong Shading: Comparisons

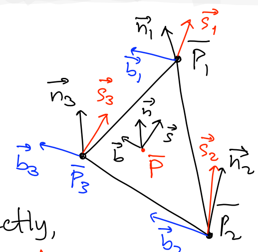
Phong shading –

① Interpolate $\vec{b}_i, \vec{n}_i, \vec{s}_i$
to get $\vec{b}, \vec{n}, \vec{s}$ at $\bar{p}$

② Compute $L(\vec{b}, \vec{n}, \vec{s})$



Comparison to Gouraud shading



(+) Smooth intensity variations as in Gouraud shading

(+) Handles specular highlights correctly, even for large triangles (Why?)

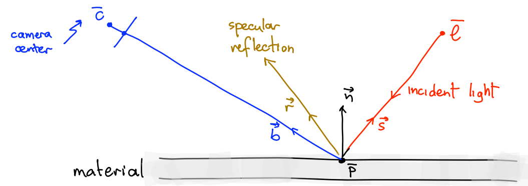
it is possible to have a significant specular component at $\bar{p}$ even when all vertices have a negligible specular component

Phong Shading: Comparisons

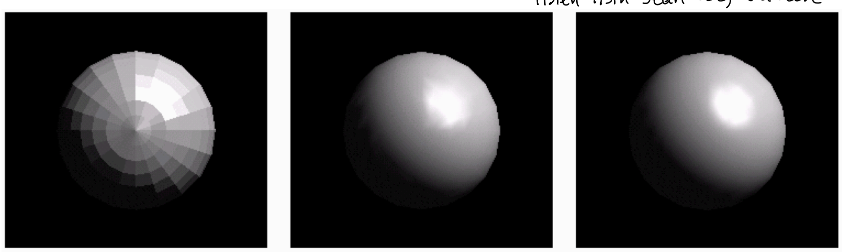
Phong shading –

① Interpolate $\vec{b_i}, \vec{n_i}, \vec{s_i}$
to get $\vec{b}, \vec{n}, \vec{s}$ at $\bar{p}$

② Compute $L(\vec{b}, \vec{n}, \vec{s})$



Hsien-Hsin Sean Lee, GaTech



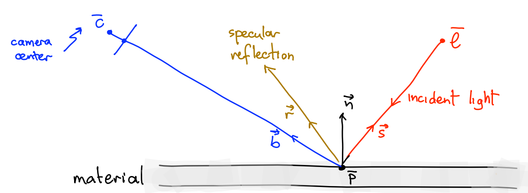
Flat shading | Gouraud shading | Phong shading

Phong Shading: Comparisons

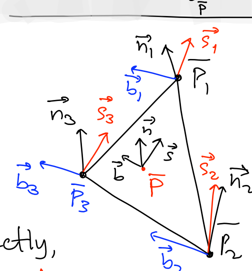
Phong shading –

① Interpolate $\vec{b_i}, \vec{n_i}, \vec{s_i}$
to get $\vec{b}, \vec{n}, \vec{s}$ at $\bar{p}$

② Compute $L(\vec{b}, \vec{n}, \vec{s})$



Comparison to Gouraud shading



(+) Smooth intensity variations as in Gouraud shading

(+) Handles specular highlights correctly, even for large triangles (Why?)

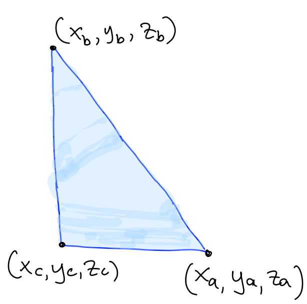
(−) Computationally less efficient (but ok on today's HW!) (must interpolate 3 vectors & evaluate Phong reflection model at each triangle pixel)

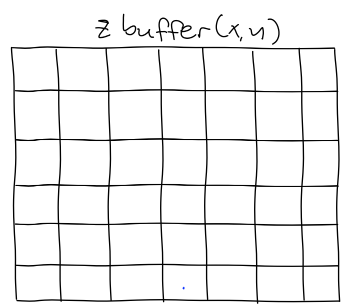
Topic 10:

Shading

- Introduction to Shading
- Flat Shading
- Interpolative Shading
  - Gouraud shading
  - Phong shading
  - Triangle scan-conversion with shading

Scan Conversion with Z-Buffering & Shading





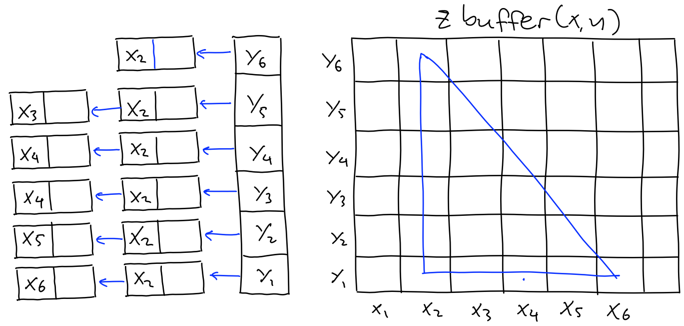
Step a: Build edge list

for each scanline, store x-intersection, pseudodepth and ☐ of each edge pixel

Step b: Fill z-buffer & color at triangle pixels

for each triangle pixel in scanline, interpolate ☐ & pseudodepth & compare to z-buffer

Scan Conversion with Z-Buffering & Shading



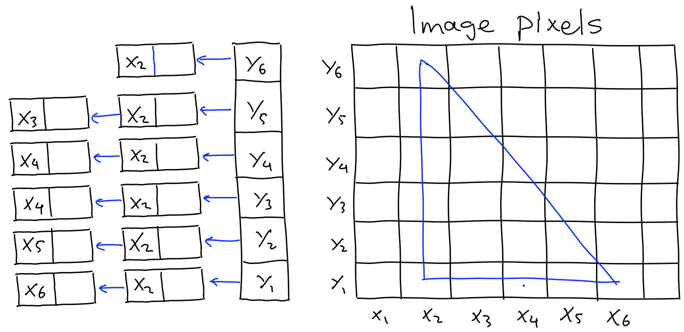
Step a: Build edge list

For each scanline, store x-intersection, pseudodepth and [ ] of each edge pixel

Step b: Fill z-buffer & color at triangle pixels

For each triangle pixel in scanline, interpolate [ ] & pseudodepth & compare to z-buffer

Scan Conversion with Gouraud Shading



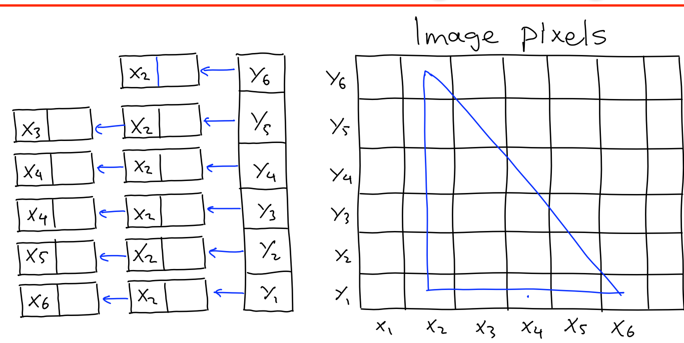
Step a: Build edge list

For each scanline, store x-intersection, pseudodepth and L value of each edge pixel

Step b: Fill z-buffer & color at triangle pixels

For each triangle pixel in scanline, interpolate L value & pseudodepth & compare to z-buffer

Scan Conversion with Phong Shading



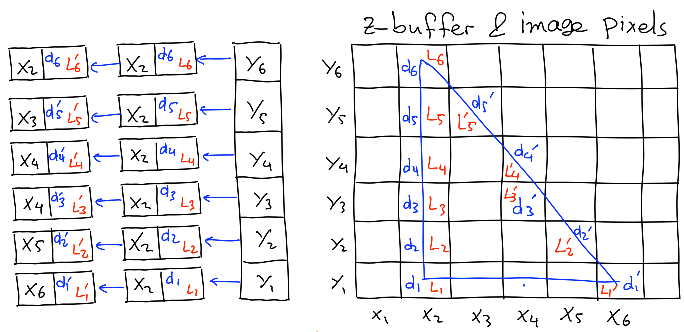
Step a: Build edge list

for each scanline, store x-intersection, pseudodepth and $\boxed{\vec{n}, \vec{s}, \vec{b}}$ of each edge pixel

Step b: Fill z-buffer & color at triangle pixels

for each triangle pixel in scanline, interpolate $\boxed{\vec{n}, \vec{s}, \vec{b}}$ & pseudodepth & compare to z-buffer

Edge List Construction (w/ Gouraud Shading)



Step a: Build edge list

for each scanline, store x-intersection, pseudodepth and $\boxed{L \text{ value}}$ of each edge pixel

for each edge $[(x_u, y_u, d_u, L_u), (x_e, y_e, d_e, L_e)]$ with $y_u > y_e$:

$x = x_e$, $d = d_e$, $\Delta x = \frac{x_u - x_e}{y_u - y_e}$, $\Delta d = \frac{d_u - d_e}{y_u - y_e}$, $\Delta L = \frac{L_u - L_e}{y_u - y_e}$

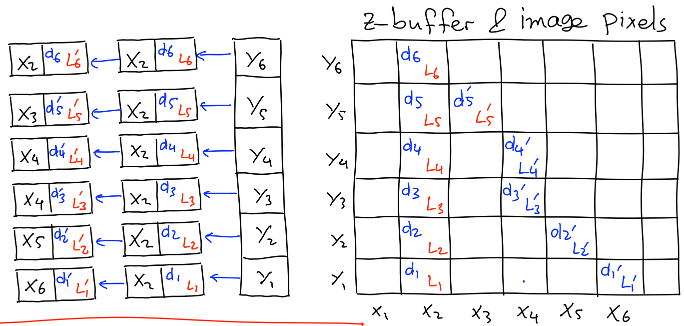
for $(y = y_e;\ y < y_u;\ y{+}{+})$

add (x, d, L) to list of scanline y

sort the list

$x = x + \Delta x$, $d = d + \Delta d$, $L = L + \Delta L$

Scanline Filling with Z-Buffering & Gouraud




```
y⁻ = min(ya, yb, yc)   y⁺ = max(ya, yb, yc)
for (y = y⁻; y ≤ y⁺; y++)
   get (xℓ, dℓ, Lℓ) and (xu, du, Lu) from
      edge list of y, with xℓ < xu
   Δd = (du - dℓ)/(xu - xℓ)      ΔL = (Lu - Lℓ)/(xu - xℓ)
   for (x = xℓ, d = dℓ, L = Lℓ; x ≤ xu; x++)
      if d < zbuffer(x,y)
         putpixel(x, y, L), zbuffer(x,y) = d
      d = d + Δd   L = L + ΔL
```

Step 6: Fill zbuffer & color at triangle pixels

for each triangle pixel in scanline, interpolate $\boxed{\text{L value}}$ & pseudodepth & compare to z-buffer