

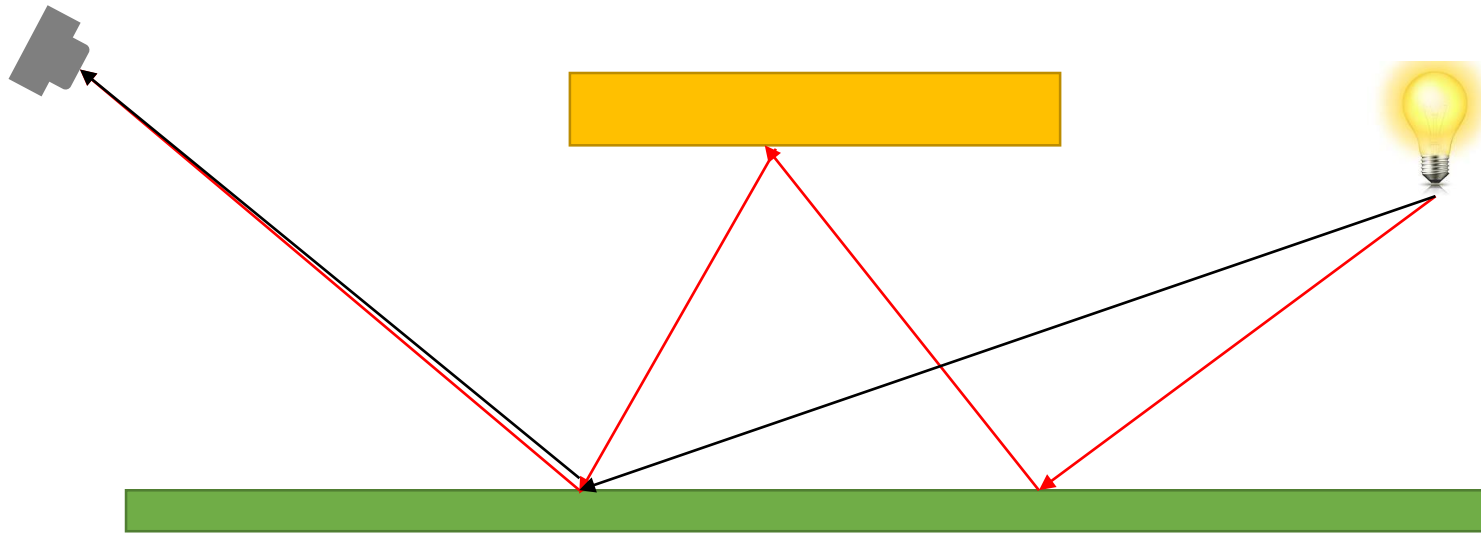
Lecture slides (AI3501/CT4201/EC4215 – Computer Graphics)

Illumination and Shading

Lecturer: Bochang Moon

Photorealism

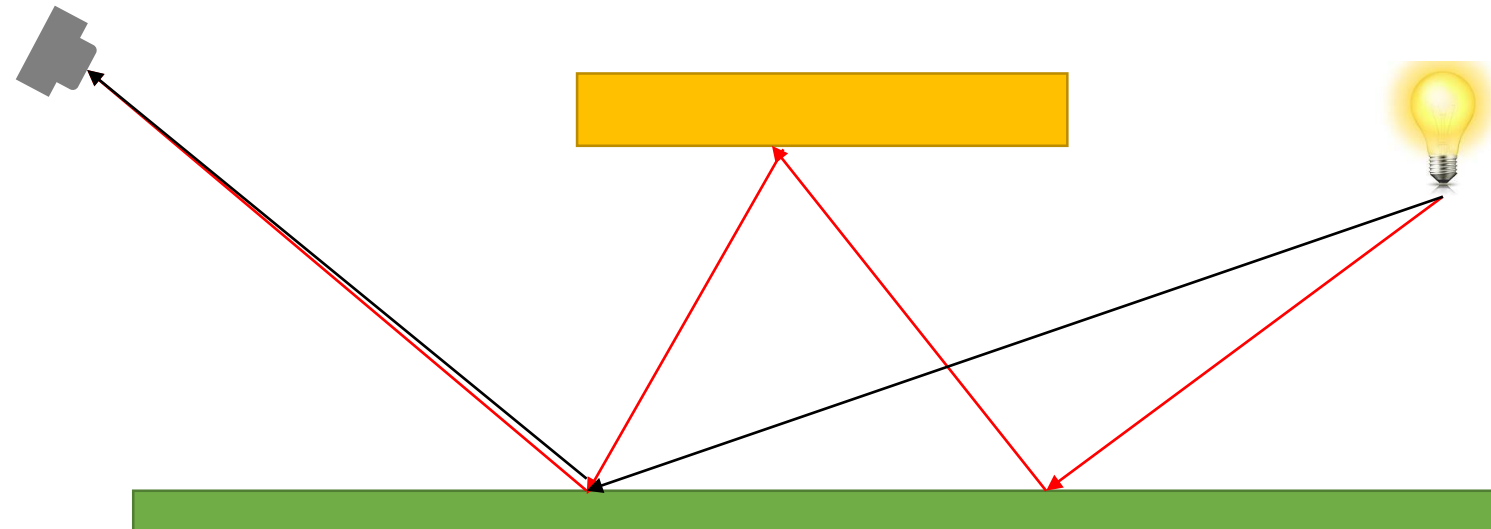
- The ultimate goal of rendering is to produce photo realistic images.
 - i.e., rendered images should be indistinguishable from photographs



Illumination (Lighting) Model

- A technique (or a model) that computes the color of a surface, by considering the following:
 - Lights
 - color, geometry (position and direction), ...
 - Surfaces
 - reflectance (color), geometry (position and normal), ...

— direct path
— indirect path



Phong Illumination Model

- A simple model that considers the three terms below:
 - Ambient:
 - Assume that each surface receives a constant amount of light
 - Ignore the following:
 - Position and orientation of a surface
 - Position of a viewer
 - Direction of a light



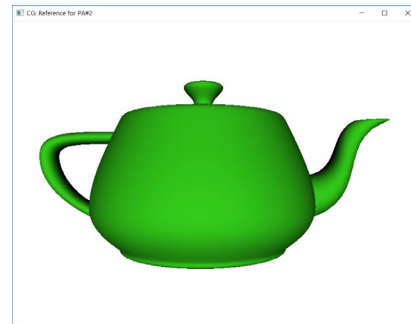
ambient

$$I = L_a k_a$$

L_a : intensity of an ambient light
 k_a : coefficient of ambient reflection

Phong Illumination Model

- A simple model that considers the three terms below:
 - Diffuse:
 - Surfaces can have different colors based on distance and orientation with respect to the light source



diffuse

Diffuse Reflection

- A simple model that considers the three terms below:
 - Diffuse:
 - When light intersects with an “ideal” diffuse surface, the surface reflects light equally in all directions.
 - Lambertian reflection
 - Outgoing light intensity is independent from the position of the viewer.

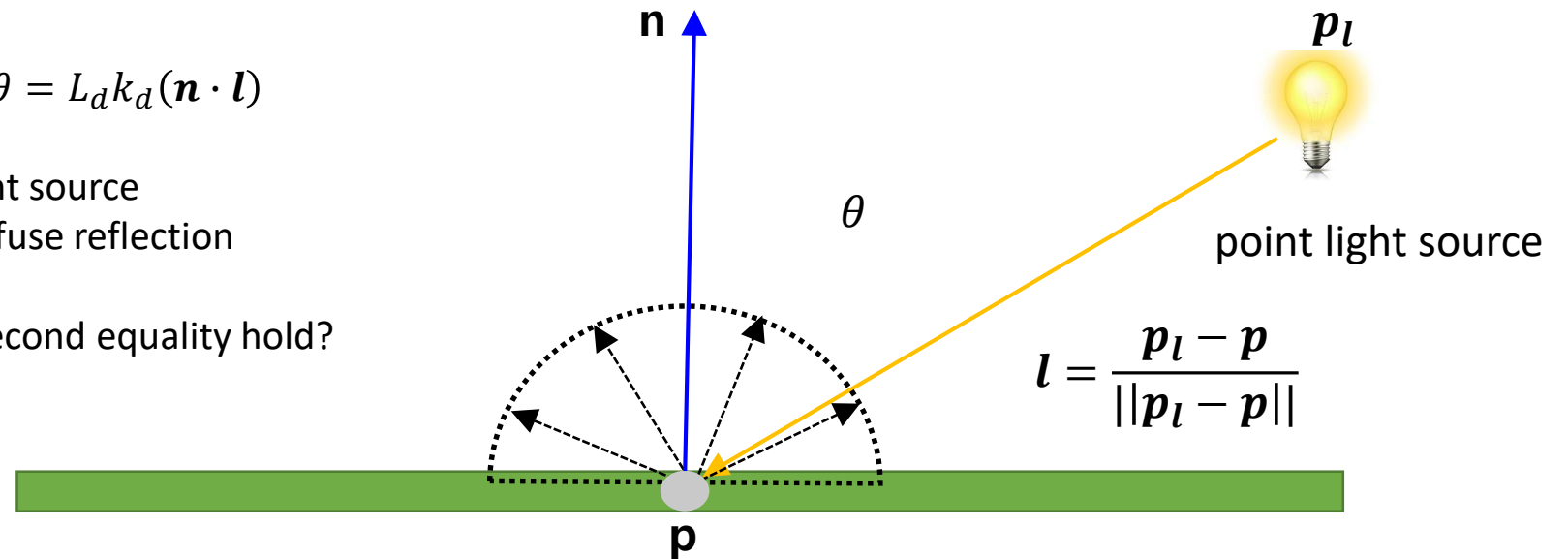
$$I = L_d k_d \cos\theta = L_d k_d (\mathbf{n} \cdot \mathbf{l})$$

L_d : intensity of a light source

k_d : coefficient of diffuse reflection

Q. When does the second equality hold?

Q. $k_d > 1$?



Diffuse Reflection

- A simple model that considers the three terms below:
 - Diffuse:
 - When light intersects with an “ideal” diffuse surface, the surface reflects light equally in all directions.
 - Lambertian reflection
 - Outgoing light intensity is independent from the position of the viewer.

$$I = L_d k_d \cos\theta = L_d k_d (\mathbf{n} \cdot \mathbf{l})$$

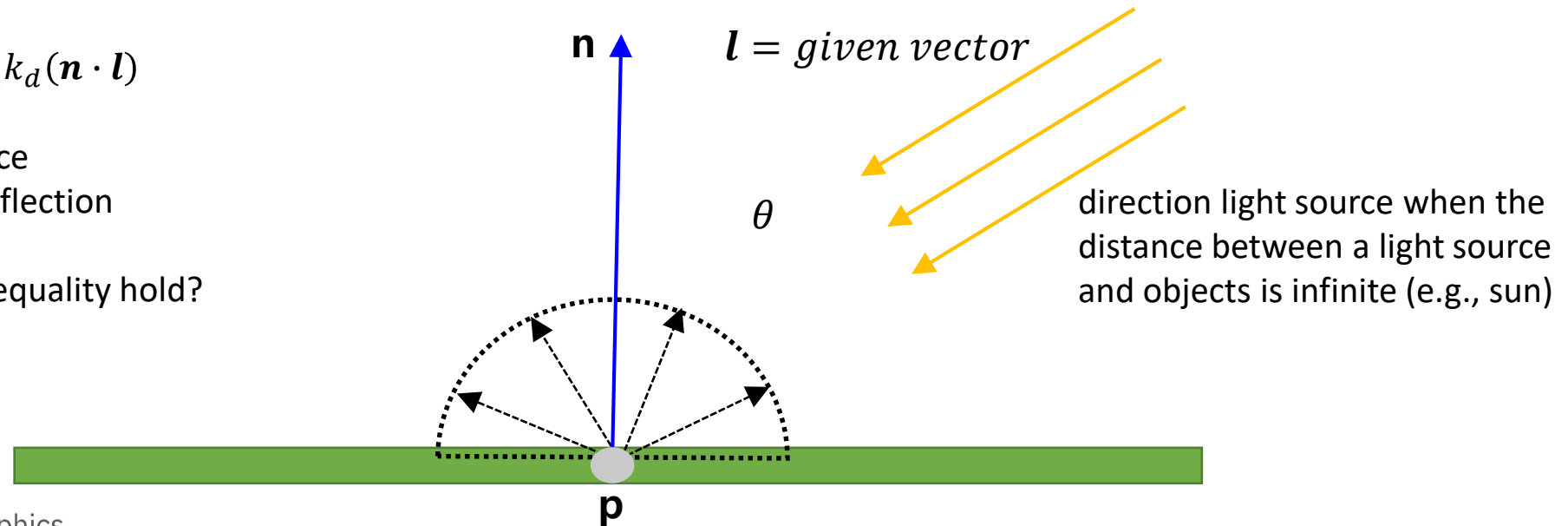
L_d : intensity of a light source

k_d : coefficient of diffuse reflection

Q. When does the second equality hold?

Q. $k_d > 1$?

Q. $\mathbf{n} \cdot \mathbf{l} < 0$?



Light Source Attenuation (optional)

- A simple model that considers the three terms below:
 - Diffuse:
 - When light intersects with an “ideal” diffuse surface, the surface reflects light equally in all directions.
 - Lambertian reflection
 - Outgoing light intensity is independent from the position of the viewer.

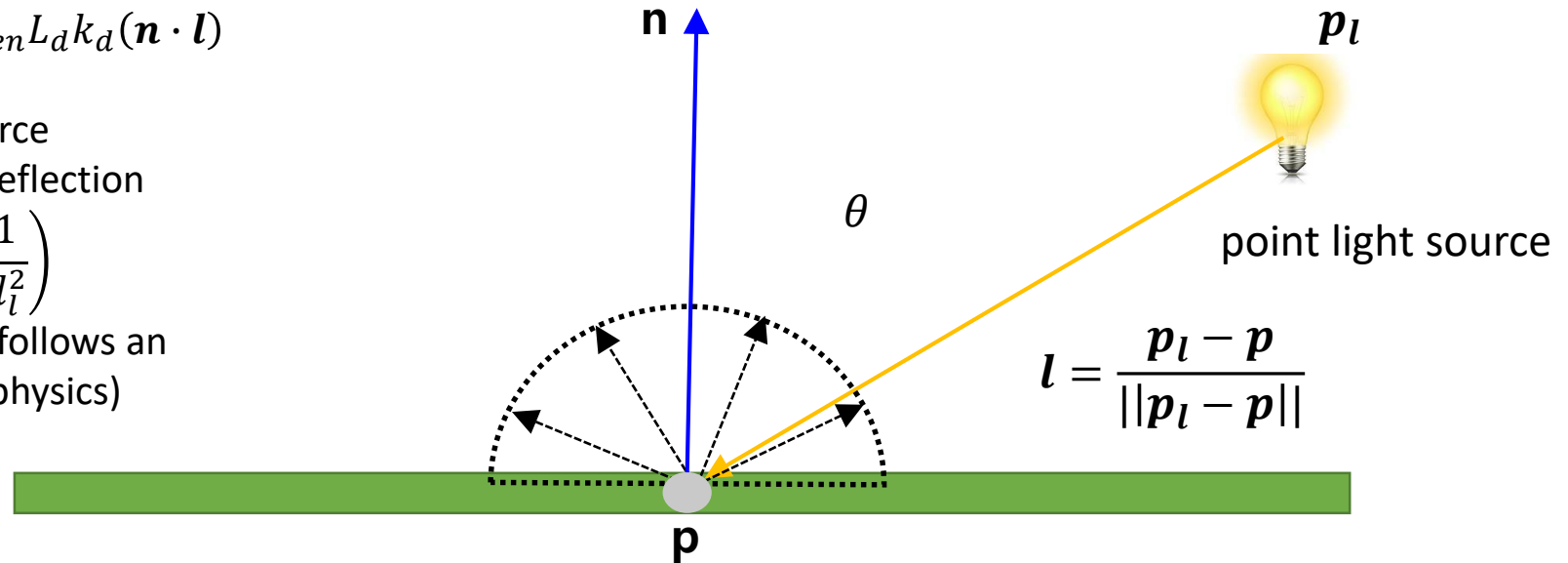
$$I = L_d k_d \cos\theta = f_{atten} L_d k_d (\mathbf{n} \cdot \mathbf{l})$$

L_d : intensity of a light source

k_d : coefficient of diffuse reflection

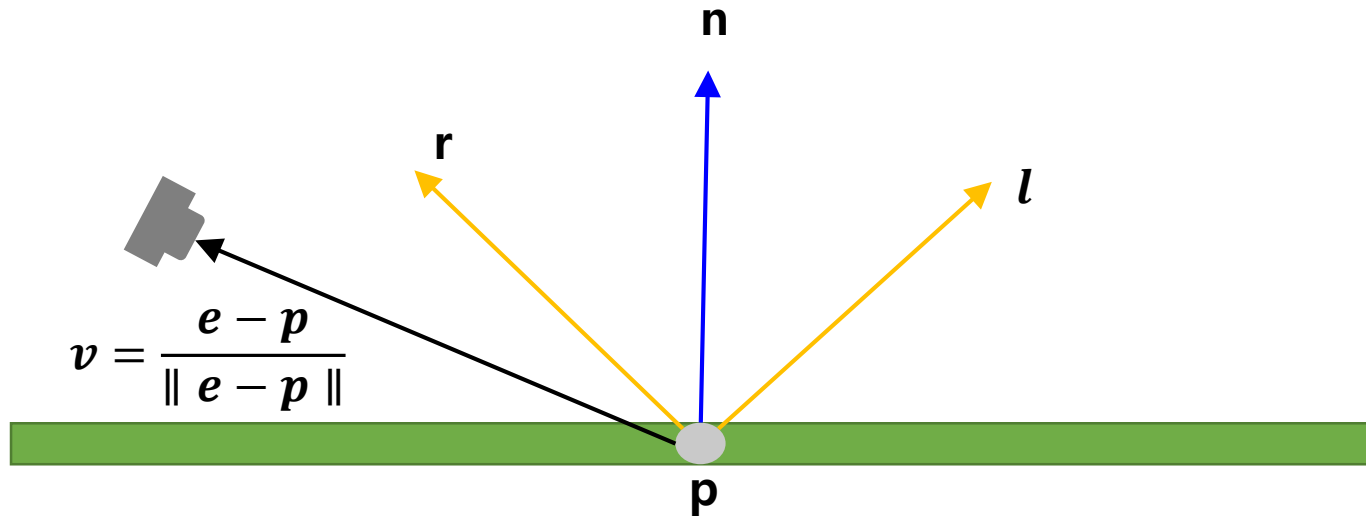
$$f_{atten} = \frac{1}{d_l^2} \text{ or } \min\left(c_1, \frac{1}{d_l^2}\right)$$

Intensity of a light source follows an inverse square law (from physics)



Specular Reflection

- A simple model that considers the three terms below:
 - specular:
 - When light intersects with a shiny surface, it can be seen as highlights (directional)
 - It also depends on how much the surface is shiny (shininess).
 - The viewing direction is also important.



Specular Reflection

- The reflection vector r is determined by Snell's law.
- Snell's law
 - $n_i \sin \theta_i = n_o \sin \theta_o$
 - n_i, n_o : indices of refraction
 - Determined by the relative speeds of light
 - Reflection is a special case
 - $\theta_l = \theta_r$
 - The medium of the incoming light and reflected ray is the same.

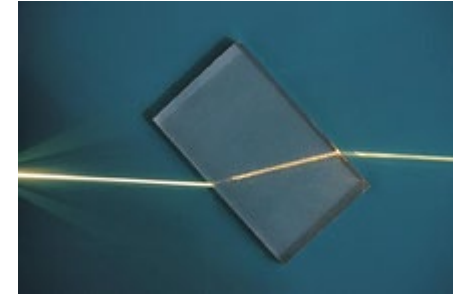
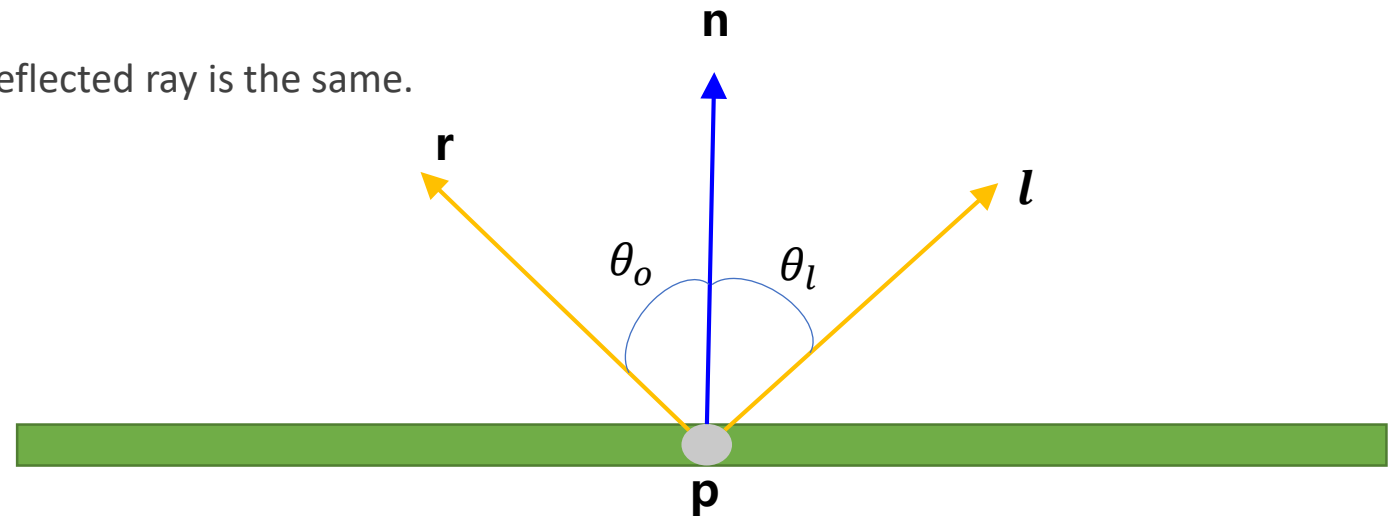


Image from wikipedia



Specular Reflection

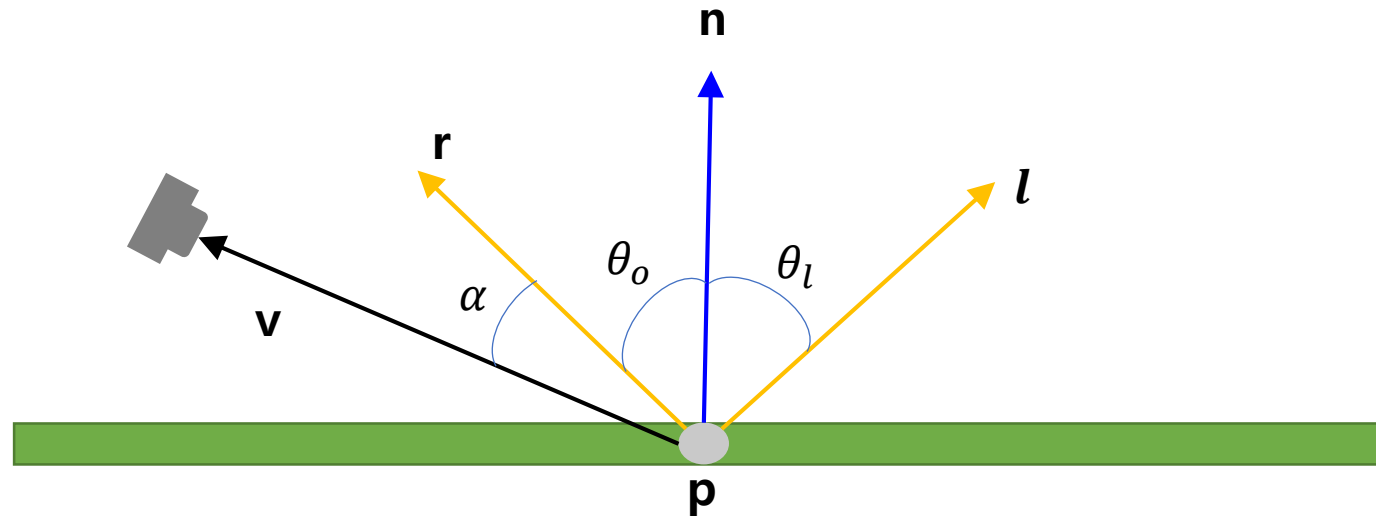
- A simple model that considers the three terms below:
 - specular:

$$I = L_S k_S \cos^s \alpha = L_S k_S (\mathbf{r} \cdot \mathbf{v})^s$$

L_S : intensity of a light source

k_S : coefficient of specular reflection

s : shininess of a surface



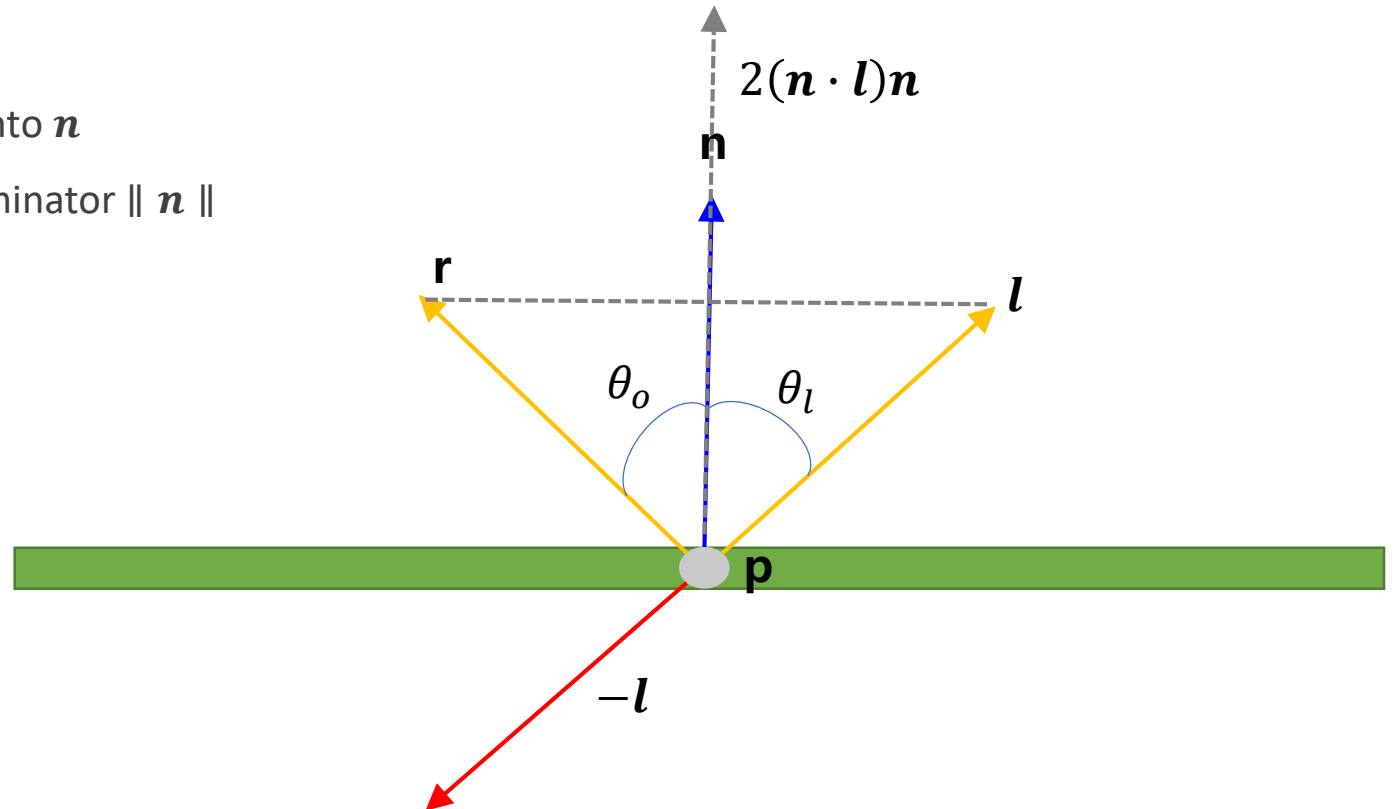
Specular Reflection

- The reflection vector, r , can be computed as the following:

- $r = 2(\mathbf{n} \cdot \mathbf{l})\mathbf{n} - \mathbf{l}$

- $\mathbf{n} \cdot \mathbf{l}$: length of the projected vector onto \mathbf{n}

- Note: we don't need to use the denominator $\|\mathbf{n}\|$



Non-ideal Specular Reflection

- A simple model that considers the three terms below:

- specular:

$$I = L_s k_s \cos^s \alpha = L_s k_s (\mathbf{r} \cdot \mathbf{v})^s$$

L_s : intensity of a light source

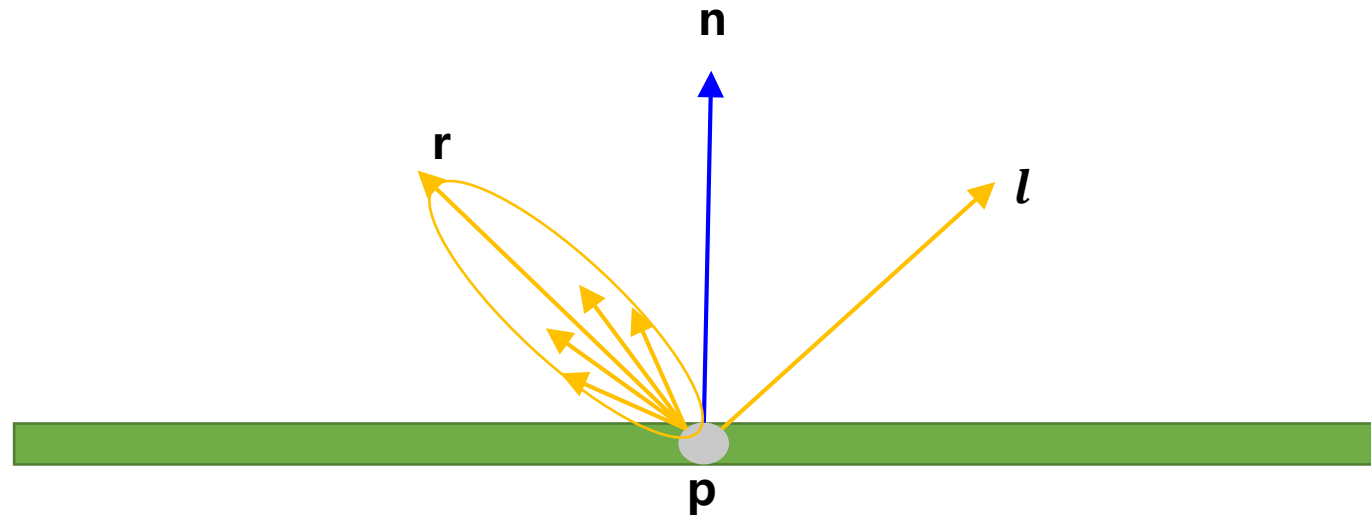
k_s : coefficient of specular reflection

s : shininess of a surface

In practice, we should use a clamping function, $\max(\mathbf{r} \cdot \mathbf{v}, 0)$

Note:

- Snell's law does not explain this case. i.e., it is only for ideal reflection (e.g., reflection on mirror)
- Approaches
 - Simulate all the reflected rays
 - Empirically capture this effect (highlights) with simple parameters (i.e., s)



Non-ideal Specular Reflection

- A simple model that considers the three terms below:
 - specular:

$$I = L_S k_S \cos^s \alpha = L_S k_S (\mathbf{r} \cdot \mathbf{v})^s$$

L_S : intensity of a light source

k_S : coefficient of specular reflection

s : shininess of a surface – control the shape of the highlights



$s = 3$



$s = 10$



$s = 30$

Blinn-Phong Model

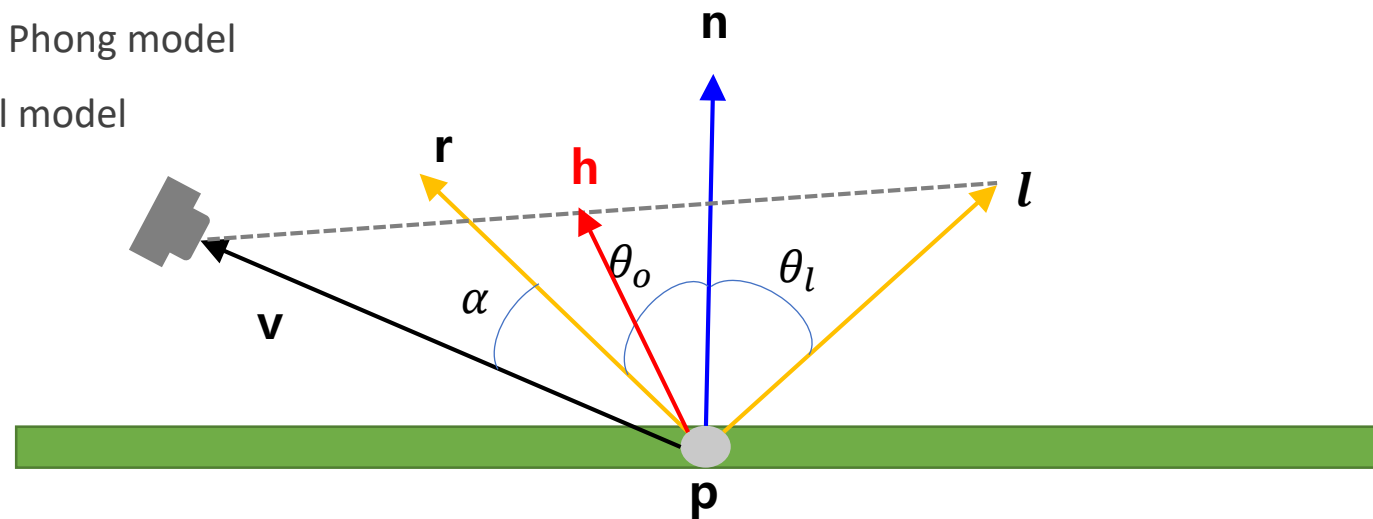
- A modification to the Phong specular model is to replace the reflection vector with the halfway vector.

- $h = \frac{l+v}{\|l+v\|}$

- $I = L_S k_S \cos^s \alpha = L_S k_S (\mathbf{n} \cdot \mathbf{h})^s$

- Note

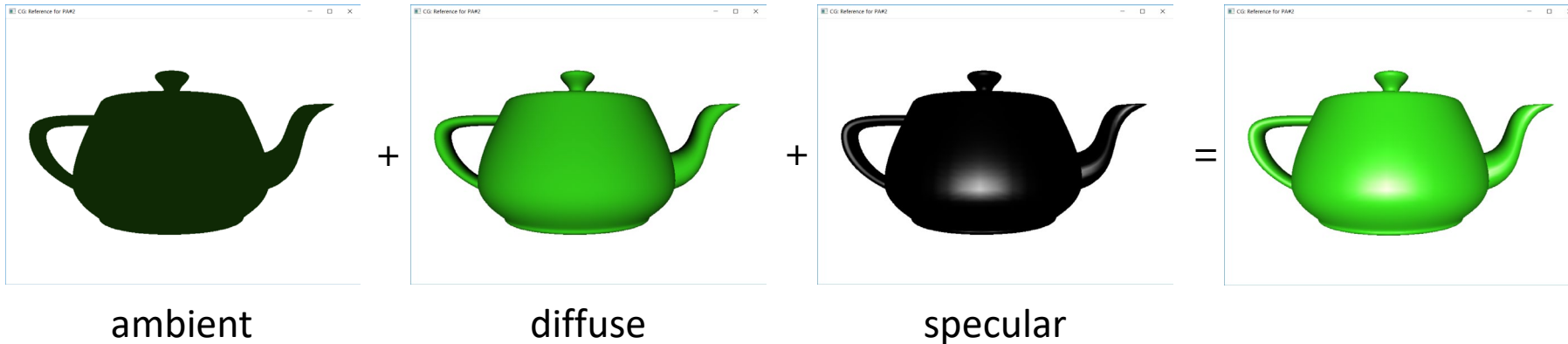
- Cheaper than the Phong model
- Another empirical model



Phong Illumination Model

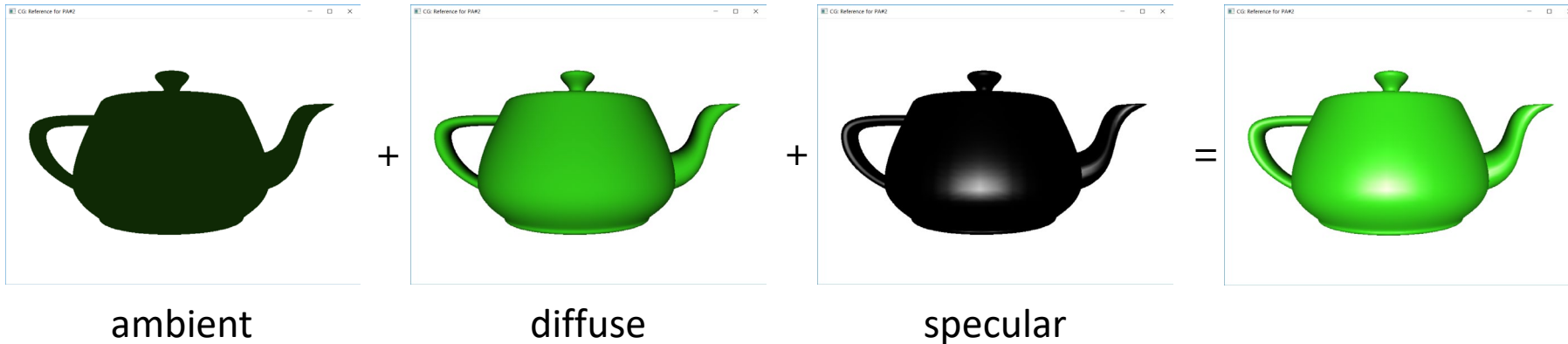
- A simple model that considers the three terms below:

$$\blacksquare I = I_a + I_d + I_s = L_a k_a + L_d k_d \max(0, \mathbf{n} \cdot \mathbf{l}) + L_s k_s \max(0, \mathbf{r} \cdot \mathbf{v})^s$$



Multiple Light Sources

- $I = \sum_{i=1}^{\# \text{ of lights}} L_a^i k_a + L_d^i k_d \max(0, \mathbf{n} \cdot \mathbf{l}^i) + L_s^i k_s \max(0, \mathbf{r}^i \cdot \mathbf{v})^s$
- OpenGL uses this empirical model.
 - Need to specify L_a, L_d, L_s for each light
 - Need to assign material properties k_a, k_d, k_s, s to each object (or each triangle)

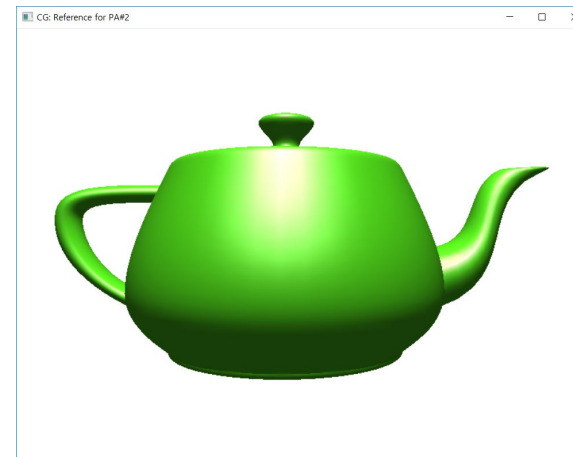


Example Code in OpenGL

- `// build a point light on the position & set parameters for the light`
- `float pointLight [] = { 0.0, 10.0, 0.0, 1.0 };`
- `float La[] = { 0.1, 0.1, 0.1, 1.0 };`
- `float Ld[] = { 1.0, 1.0, 1.0, 1.0 };`
- `float Ls[] = { 1.0, 1.0, 1.0, 1.0 };`
- `glLightfv(GL_LIGHT0, GL_POSITION, pointLight);`
- `glLightfv(GL_LIGHT0, GL_AMBIENT, La);`
- `glLightfv(GL_LIGHT0, GL_DIFFUSE, Ld);`
- `glLightfv(GL_LIGHT0, GL_SPECULAR, Ls);`
- `glEnable(GL_LIGHT0);`

Example Code in OpenGL

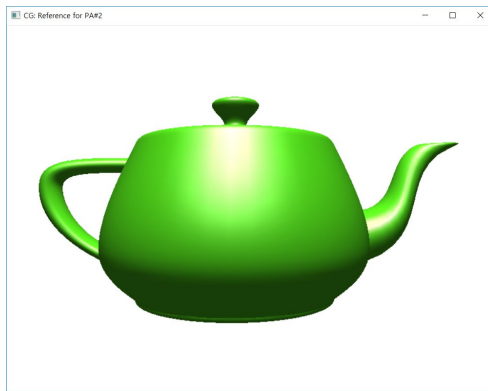
- `// assign a material to an object`
- `float ka[] = { 0.3, 0.8, 0.1, 1.0 };`
- `float kd[] = { 0.3, 0.8, 0.1, 1.0 };`
- `float ks[] = { 0.9, 0.9, 0.9, 1.0 };`
- `float shininess[] = { 30.0 };`
- `glMaterialfv(GL_FRONT, GL_AMBIENT, ka);`
- `glMaterialfv(GL_FRONT, GL_DIFFUSE, kd);`
- `glMaterialfv(GL_FRONT, GL_SPECULAR, ks);`
- `glMaterialfv(GL_FRONT, GL_SHININESS, shininess);`



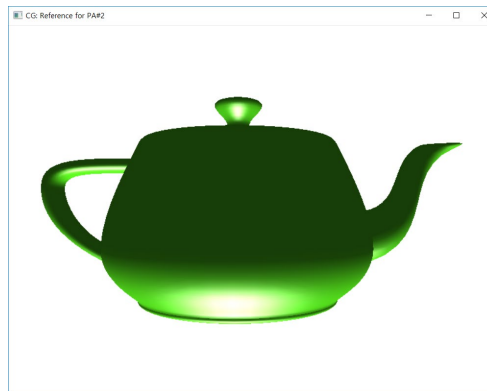
Illuminated by light0

Example Code in OpenGL

- `// build a directional light (w = 0.0)`
- `float directionalLight[] = { 0.0, -10.0, 0.0, 0.0 };`
- `glLightfv(GL_LIGHT1, GL_POSITION, directionalLight);`
- `glLightfv(GL_LIGHT1, GL_AMBIENT, La);`
- `glLightfv(GL_LIGHT1, GL_DIFFUSE, Ld);`
- `glLightfv(GL_LIGHT1, GL_SPECULAR, Ls);`
- `glEnable(GL_LIGHT1);`



Illuminated by light0



Illuminated by light1



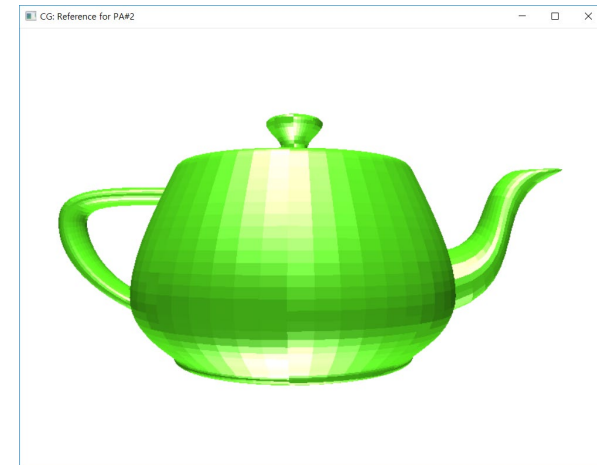
Illuminated by both lights

Shading

- An illumination model defines how to compute the color of a point on surfaces.
- A shading model defines where we should use the illumination model.
 - Flat (constant)
 - Gouraud
 - Phong

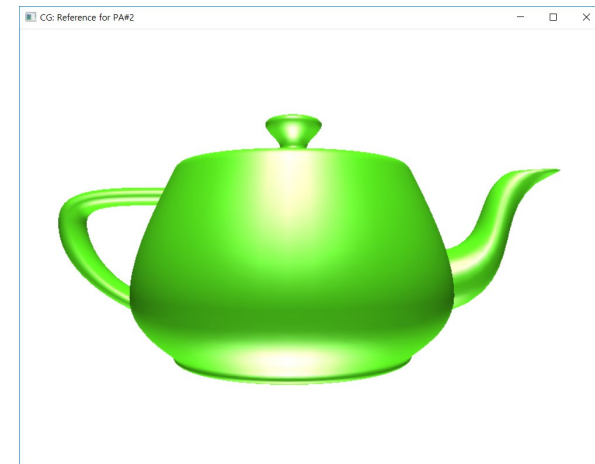
Flat (Constant) Shading

- The illumination model is applied only once to a polygon
 - All points within the polygon have the same color
- Properties
 - Very simple & cheap
 - Very low rendering quality (not smooth)
- OpenGL code
 - `glShadeModel(GL_FLAT);`



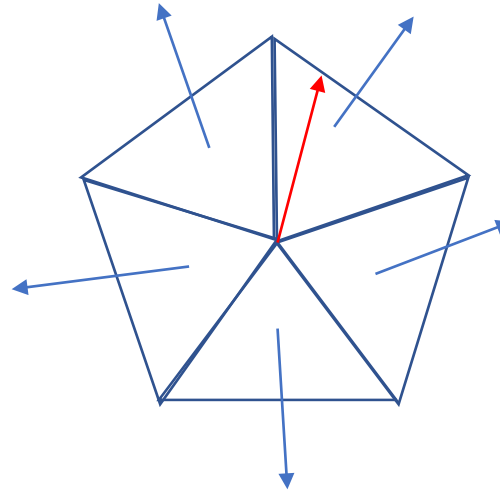
Gouraud Shading

- The illumination model is applied to each vertex of a polygon
 - The inside points' colors of the polygon are interpolated ones of the vertex colors
- Properties
 - More expensive than the flat shading
 - Higher quality than the constant
 - Relatively good but still not very smooth
- OpenGL code
 - `glShadeModel(GL_SMOOTH);`



Vertex Normal

- The normal at a vertex is generally an averaged one of the normals of neighboring polygons

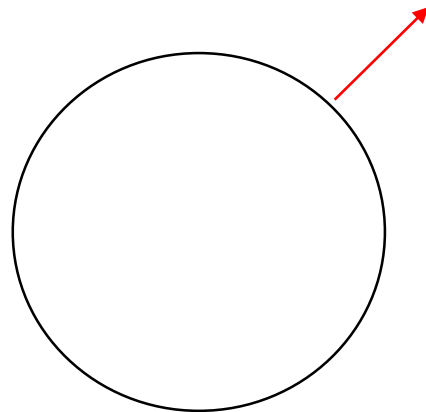


Phong Shading

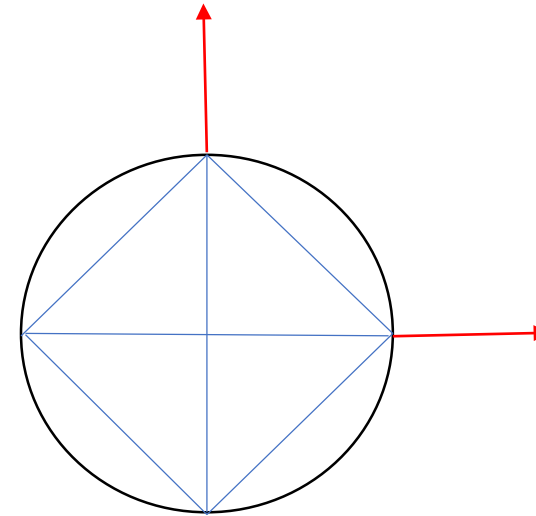
- The illumination model is applied for each point of a polygon
 - Should compute the normal at a point by interpolating the vertex normals
- Properties
 - Much more expensive than the flat and Gouraud shading methods
 - Produce very smooth appearance
- OpenGL code
 - Not supported as a simple function call
 - You should implement it (e.g., per-fragment shading)
 - Further reading: search fragment shader (c/c++ like language)
 - You can make your own shading

Gouraud vs. Phong Shading

- Gouraud shading may miss some highlights in the middle of a polygon
 - It sacrifices the quality to improve rendering speed
- Q. Is there a way to improve the quality without using Phong shading?



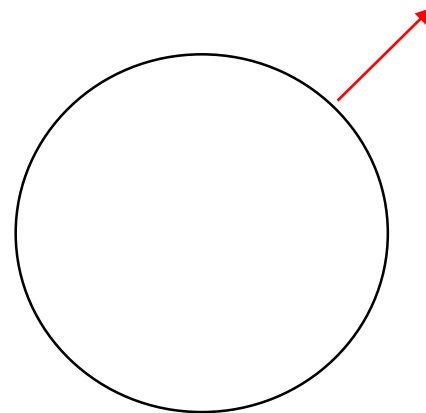
Original model



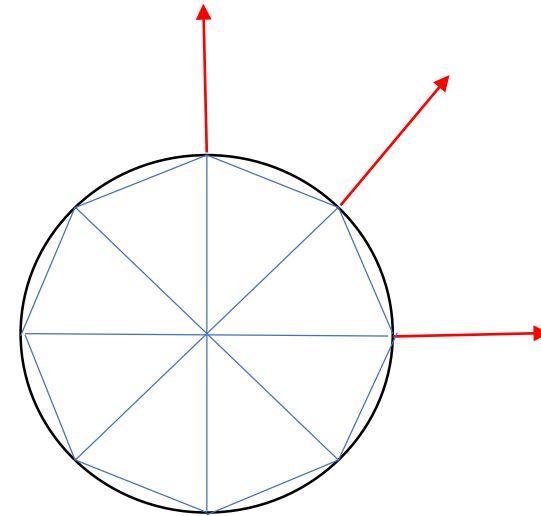
Approximate model

Gouraud vs. Phong Shading

- Gouraud shading may miss some highlights in the middle of a polygon
 - It sacrifices the quality to improve rendering speed
- Q. Is there a way to improve the quality without using Phong shading?
 - We can subdivide a polygon into multiple polygons



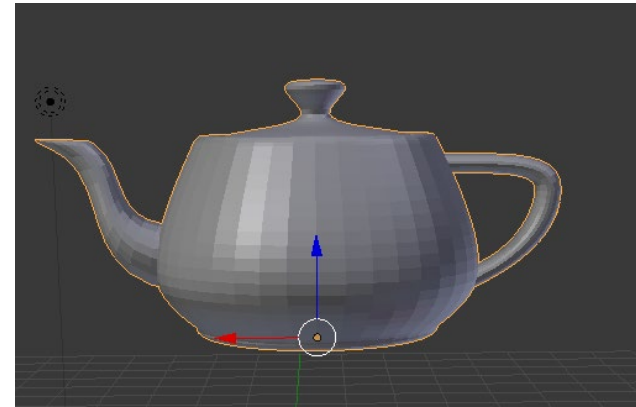
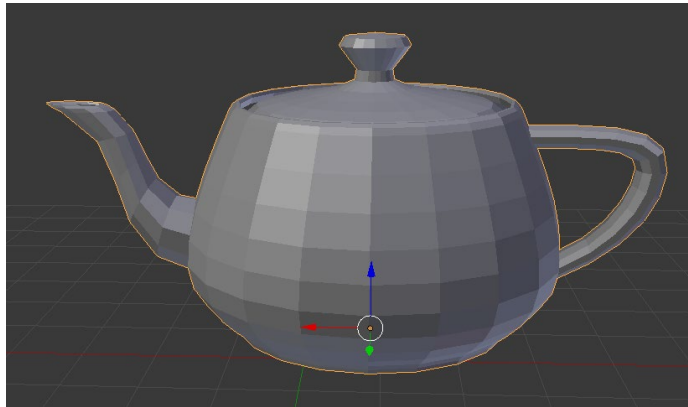
Original model



Approximate model

Gouraud vs. Phong Shading

- Gouraud shading may miss some highlights in the middle of a polygon
 - It sacrifices the quality to improve rendering speed
- Q. Is there a way to improve the quality without using Phong shading?
 - We can subdivide a polygon into multiple polygons



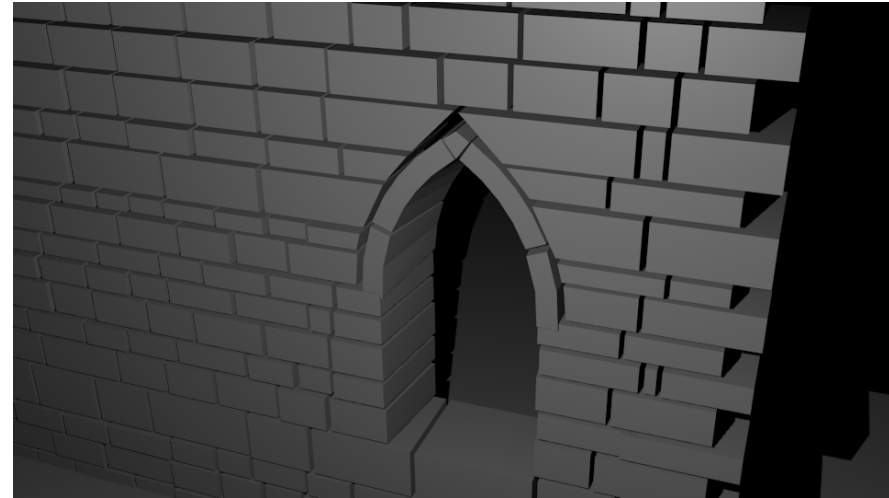
Illumination Models

- Local illumination
- Global illumination



Local Illumination

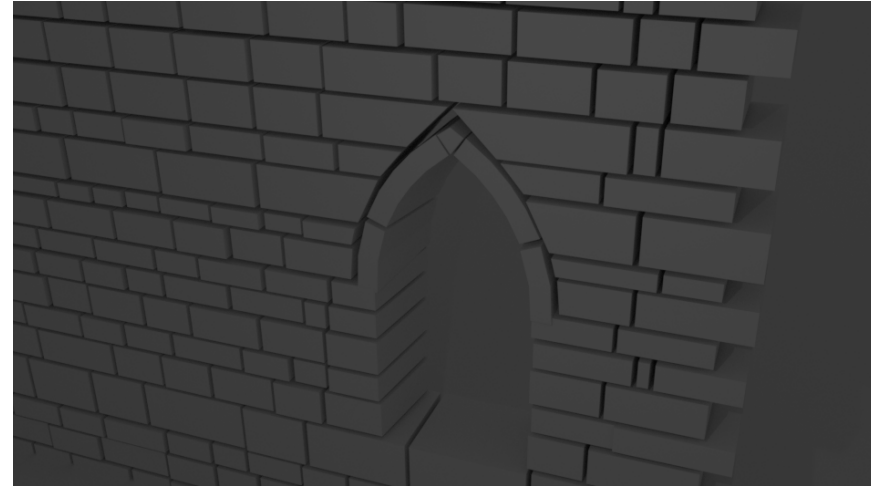
- Determine the colors of surface points based on only local information:
 - Materials of the surface
 - Geometry (normal and position) of the point
 - Lights
- Limitations:
 - One-bounce reflection of light
 - Hack (ambient term) for approximation indirect illumination
- Phong illumination model
 - Empirical model for local illumination



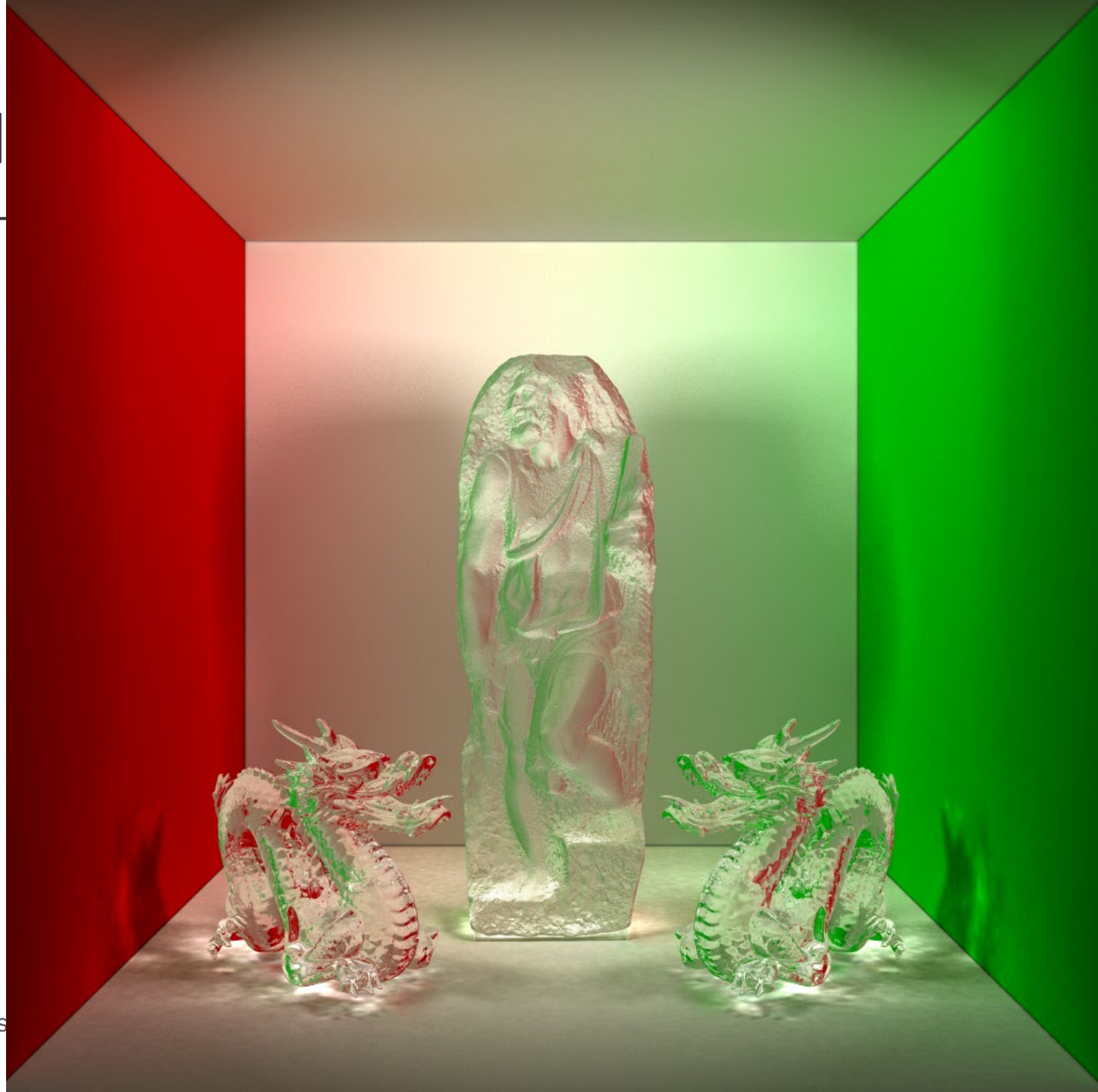
Global Illumination

- Light can traverse in the virtual world:
 - Colors of a surface can be affected by lights and other surfaces

- e.g., ray tracing



Global Illu





Some
videos?

Further Readings

- Chapter 4.5
- Chapter 10