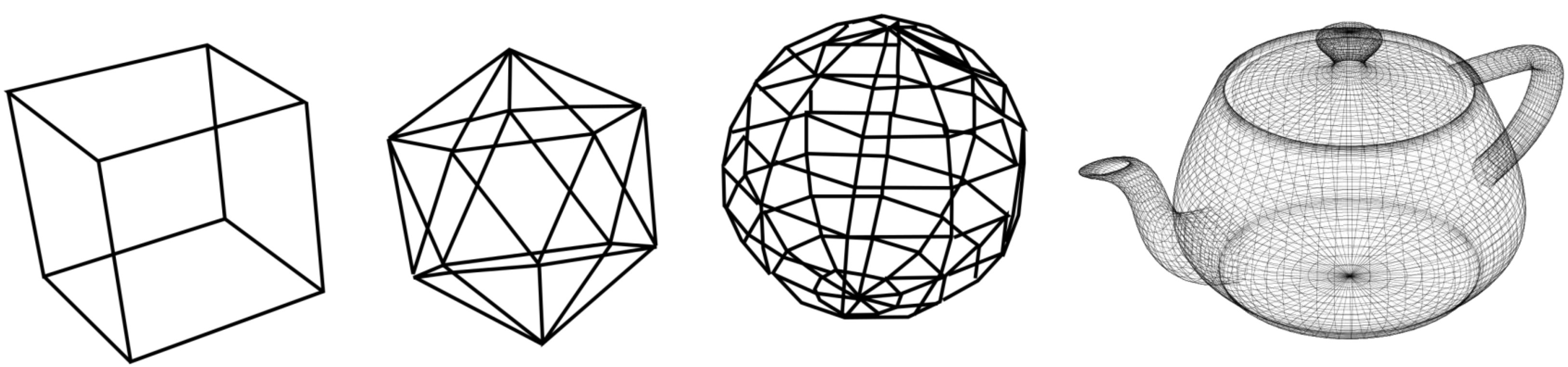
VIETNAM ACADEMY OF SCIENCE AND TECHNOLOGY
UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI

COMPUTER GRAPHICS

Lecture 6: Lighting

Lecturer: Dr. NGUYEN Hoang Ha



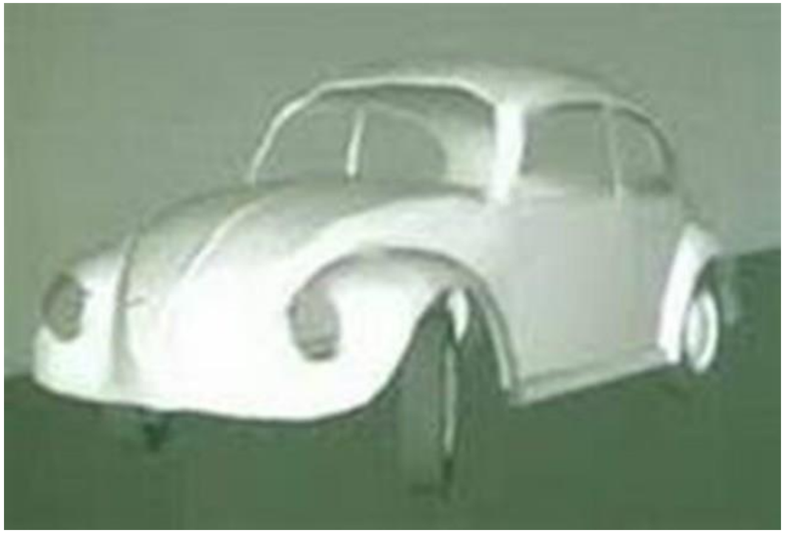
Reference: JungHyun Han. 2011. 3D Graphics for Game Programming (1st ed.), chapter 5

What is Illumination?

- Illumination (or lighting): techniques handling the interaction between light sources and objects.
- 2 categories:
  - Local illumination considers only direct lighting in the sense that the illumination of a surface depends solely on the properties of the light sources and the surface materials. This has been dominant in real-time graphics.
  - In the real world, however, every surface receives light indirectly. (Even though a light source is invisible from a particular point of the scene, light can still be transferred to the point through reflections or refractions from other surfaces of the scene.) For indirect lighting, the global illumination (GI) model considers the scene objects as potential lighting sources.
- Problems of interactive GI
  - The cost is often too high to permit interactivity.
  - The rasterization-based architecture of GPU is more suitable for local illumination.
- Current status of GI
  - Approximate GI instead of pursuing precise GI.
  - Pre-compute GI, store the result in a texture, and use it at run time.

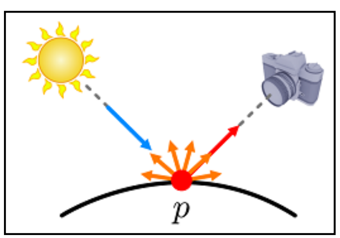
Bui Tuong Phong

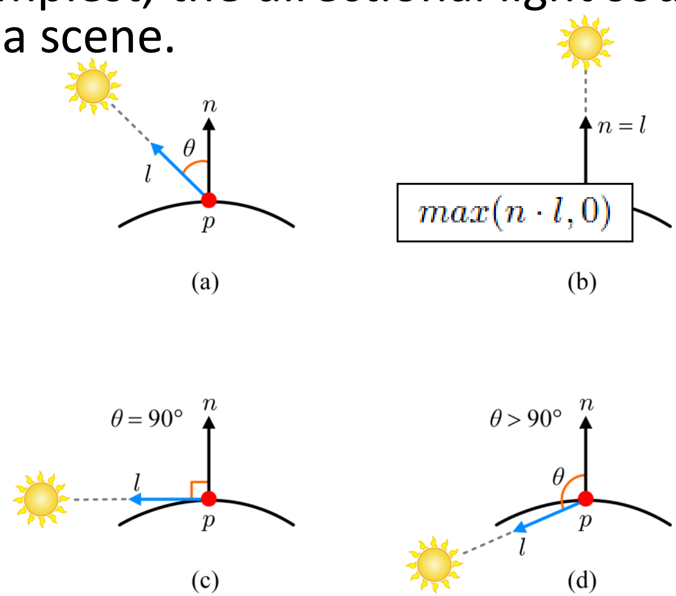
- 1942: born in Hanoi
- 1964-1966: bachelor in Grenoble
- 1968: engineer in Toulouse
- 1971-1973: PhD at Utah University, supervised by Sutherland
- 1973: Illumination model



Phong Lighting Model - Diffuse Term

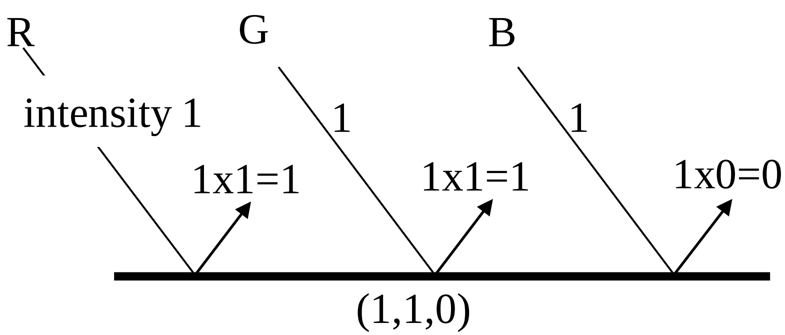
- The most popular local illumination method is based on the *Phong model*. It is composed of diffuse, specular, ambient, and emissive terms.
- The diffuse term is based on Lambert's law. Reflections from ideally diffuse surfaces (Lambertian surfaces) are scattered with equal intensity in all directions.
- So, the amount of perceived reflection is independent of the view direction, and is just proportional to the amount of incoming light.
- Among various light types such as point, area, spot, and directional light sources, let's take the simplest, the directional light source, where the *light vector* (*l*) is constant for a scene.

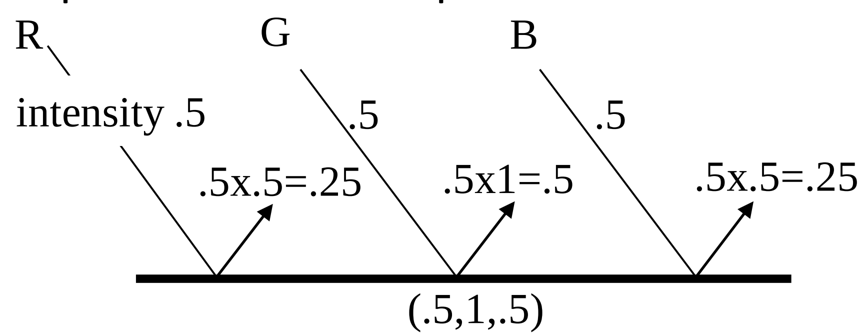




Phong Lighting Model - Diffuse Term (cont')

- Suppose a white light (1,1,1). If an object lit by the light appears yellow, it means that the object reflects R and G and absorbs B. We can easily implement this kind of filtering through material parameter, i.e., if it is (1,1,0), then (1,1,1)⊗(1,1,0)=(1,1,0) where ⊗ is component-wise multiplication.

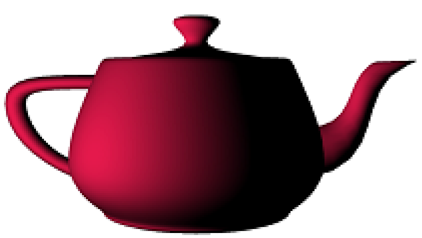




$$\boxed{s_d \otimes m_d}$$

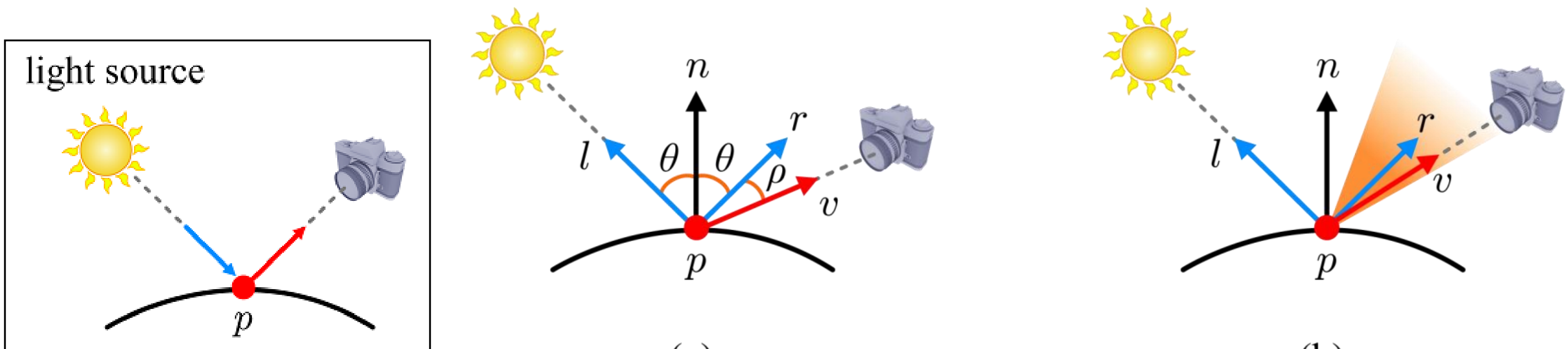
$$max(n \cdot l, 0) s_d \otimes m_d$$

- The diffuse term:



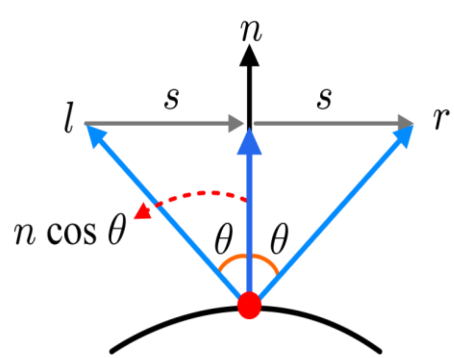
Phong Lighting Model - Specular Term

- The specular term is used to make a surface look shiny via *highlights*, and it requires *view vector* (*v*) and *reflection vector* (*r*) in addition to *light vector* (*l*).



(a) (b)

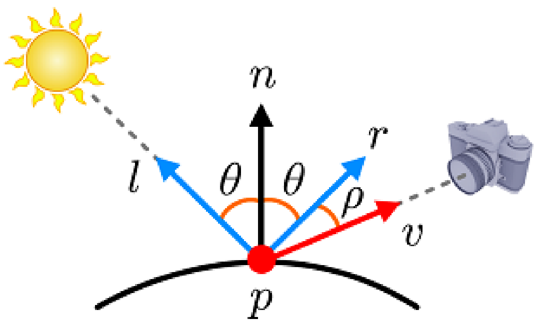
- Computing the reflection vector

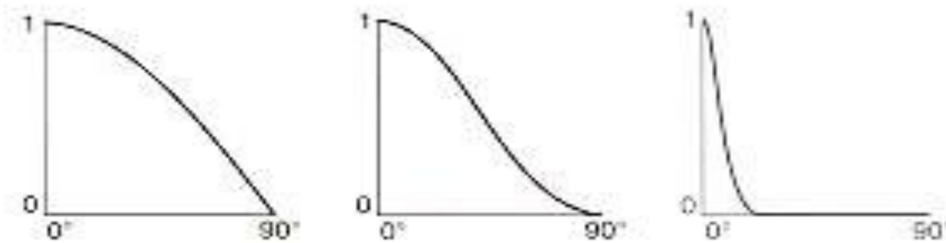


$$s = n\cos\theta - l$$
$$s = r - n\cos\theta$$
$$r = 2n\cos\theta - l = 2n\,(n\cdot l) - l$$

Phong Lighting Model - Specular Term (cont')

- Whereas the diffuse term is view-independent, the specular term is highly view-dependent.
  - For a perfectly shiny surface, the highlight at p is visible only when ρ equals 0.
  - For a surface that is not perfectly shiny, the maximum highlight occurs when ρ equals 0, but falls off sharply as ρ increases.
  - The rapid fall-off of highlights is often approximated by $(r\cdot v)^{sh}$, where sh denotes shininess.

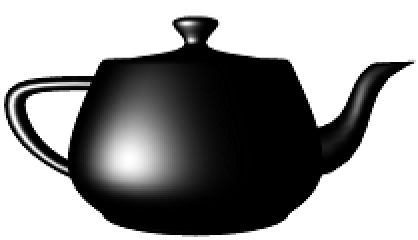




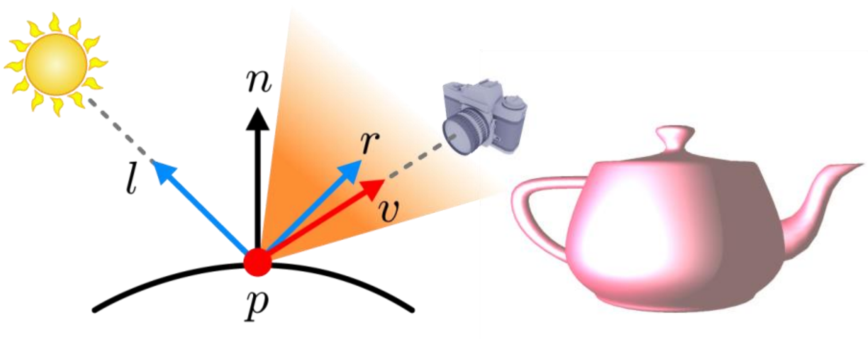
$\cos\rho$ $\cos^2\rho$ $\cos^{64}\rho$

$\boxed{(r\cdot v)^{sh}}$

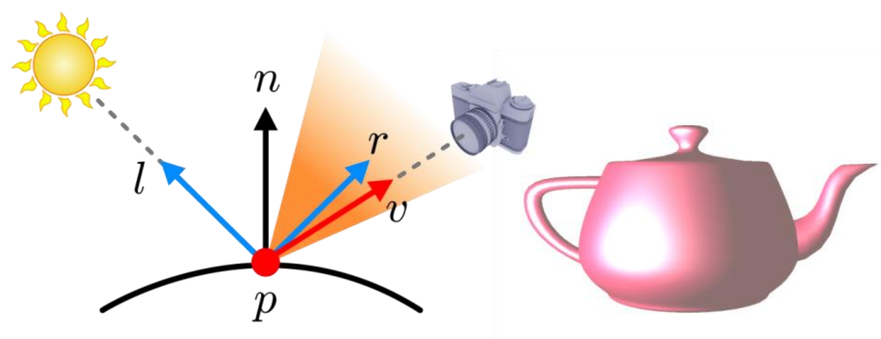
- The specular term: $(max(r\cdot v,0))^{sh} s_s \otimes m_s$
- Unlike m_d, m_s is usually a gray-scale value rather than an RGB color. It enables the highlight on the surface to end up being the color of the light source.



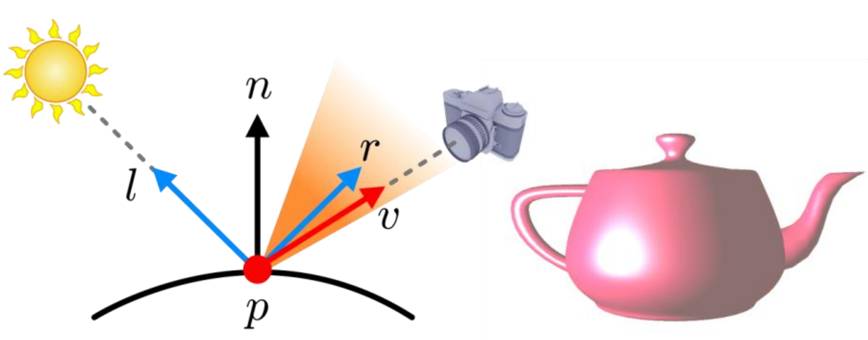
Phong Lighting Model - Specular Term (cont')



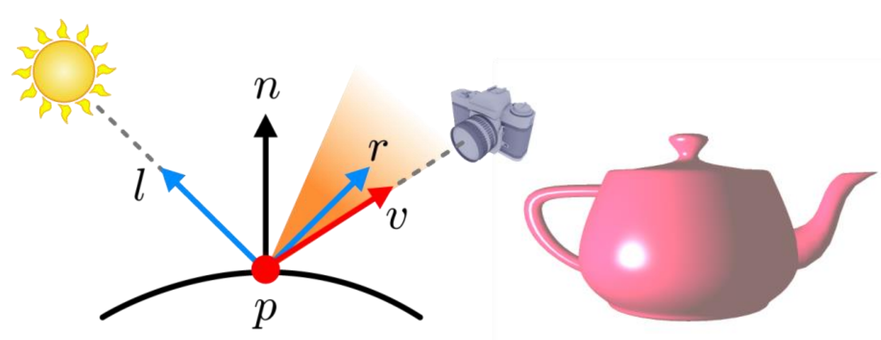
(a) $sh = 5$



(b) $sh = 10$



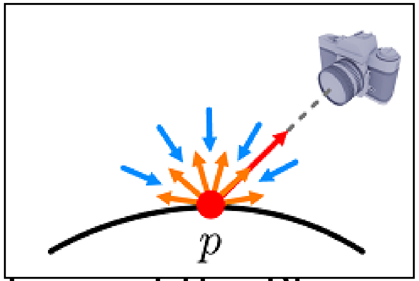
(c) $sh = 20$



(d) $sh = 80$

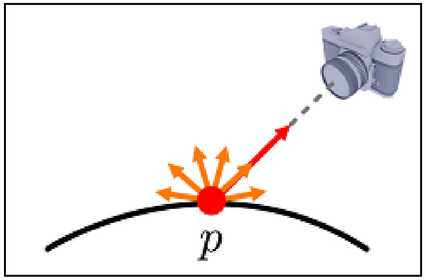
Phong Lighting Model – Ambient and Emissive Terms

- The ambient light describes the light reflected from the various objects in the scene, i.e., it accounts for *indirect lighting*.
- As the ambient light has bounced around so much in the scene, it arrives at a surface point from all directions, and reflections from the surface point are also scattered with equal intensity in all directions.



$s_a \otimes m_a$

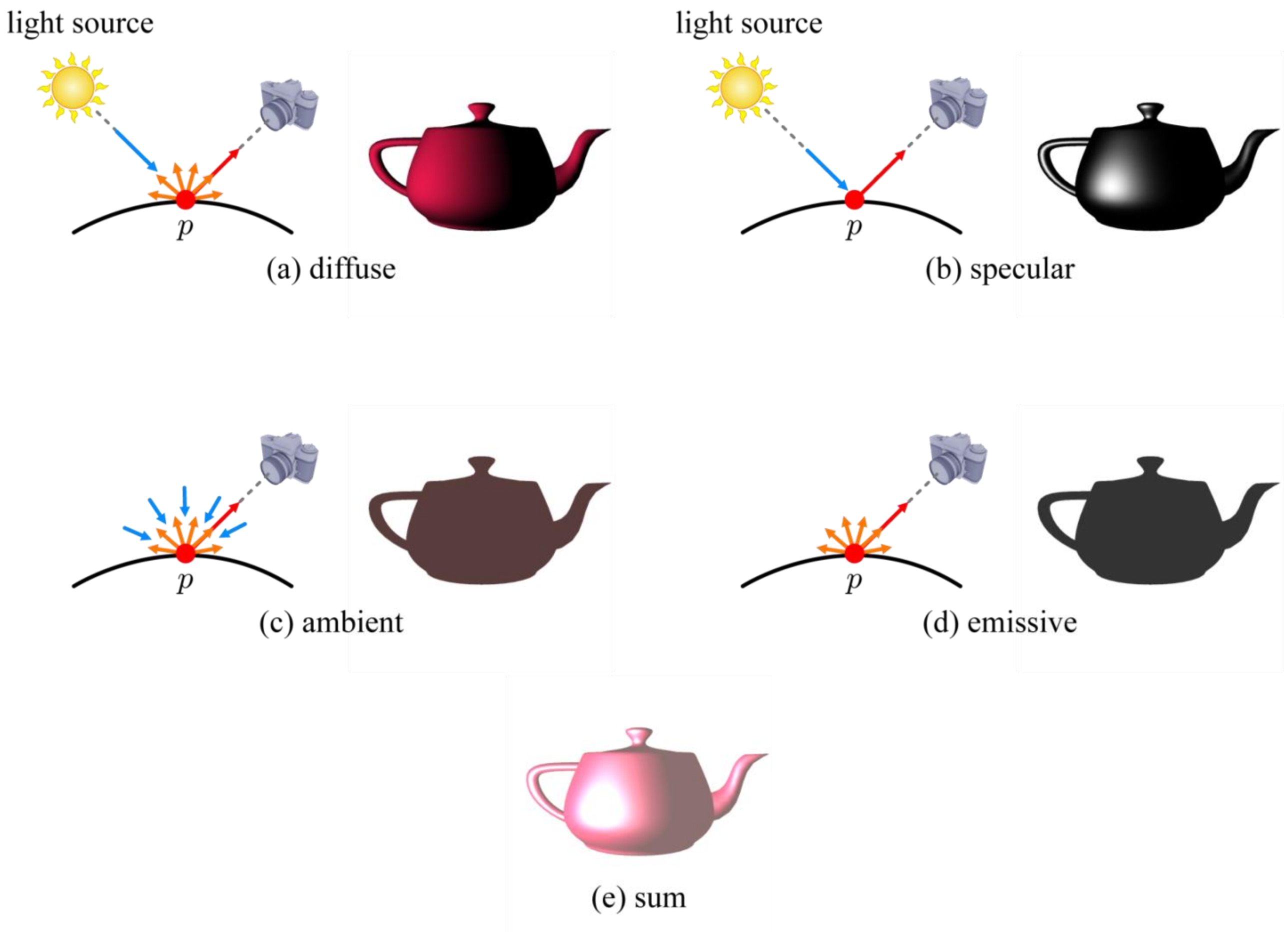
- The last term of the Phong model is the emissive term m_e that describes the amount of light emitted by a surface itself.



Phong Lighting Model

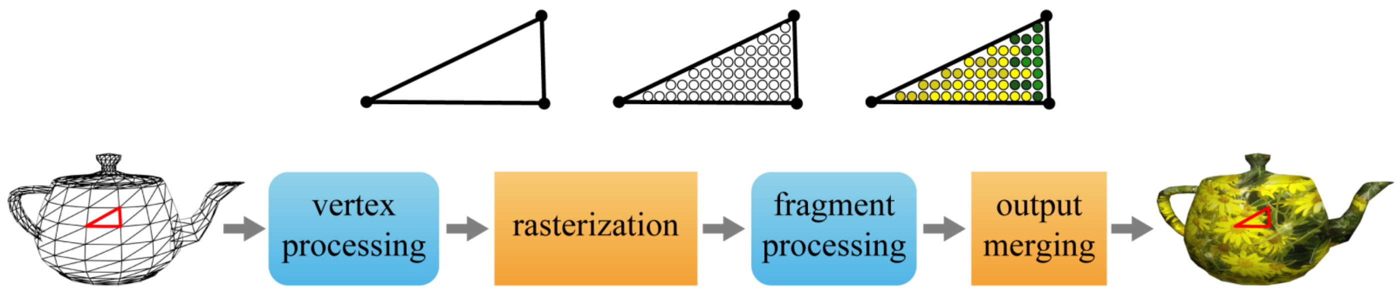
- The Phong model sums the four terms!!

$$\boxed{max(n \cdot l, 0) s_d \otimes m_d + (max(r \cdot v, 0))^{sh} s_s \otimes m_s + s_a \otimes m_a + m_e}$$



Shader

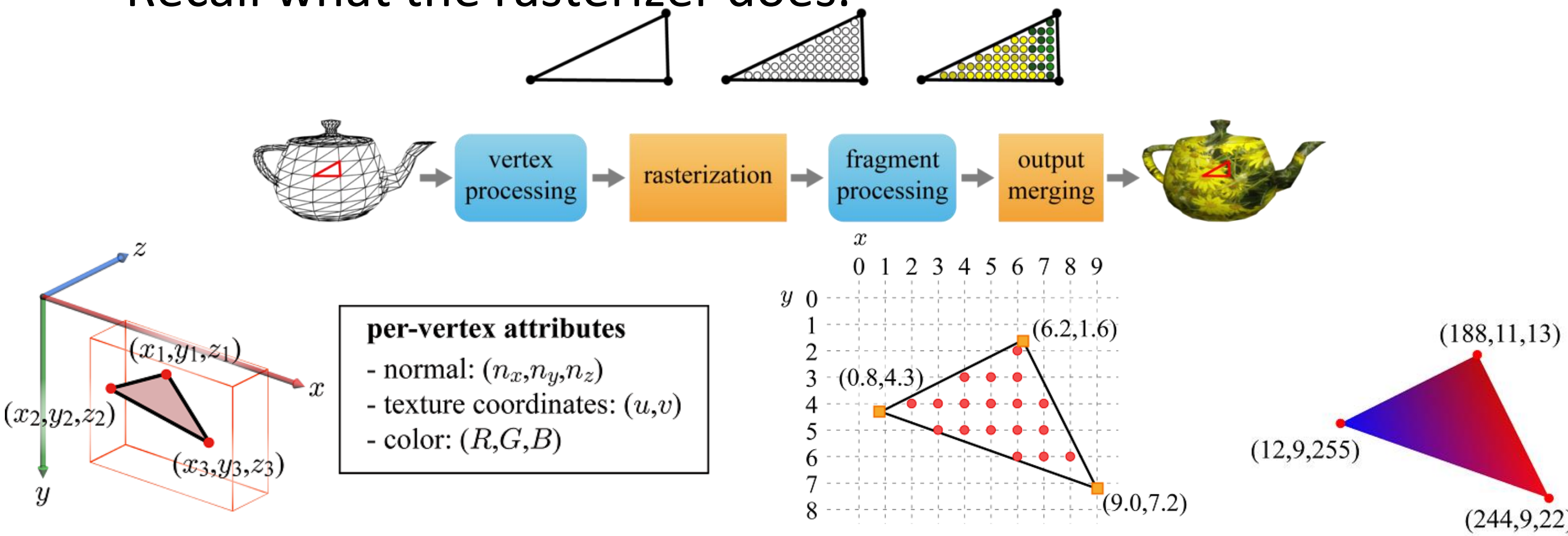
- A shader is an executable program running on the GPU, and consists of a set of software instructions.



- Shading languages (They are all C-like.)
  - High Level Shading Language (HLSL) by Microsoft
  - Cg (C for graphics) by NVIDIA
  - GLSL (OpenGL Shading Language) by OpenGL

Per-vertex Lighting (cont')

- Recall what the rasterizer does.

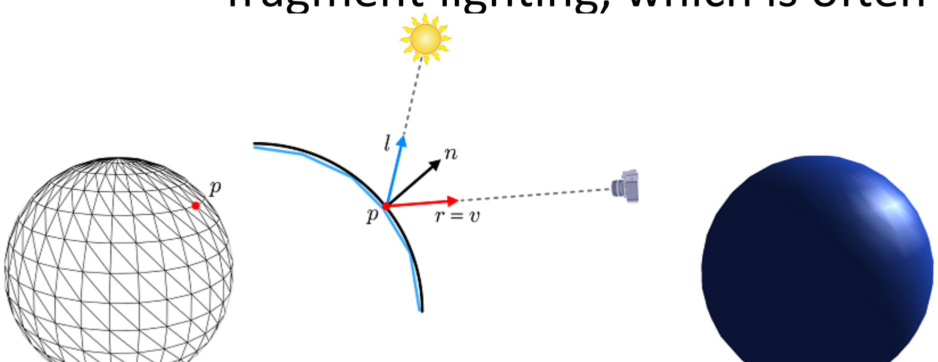


- The simplest fragment shader for per-vertex lighting

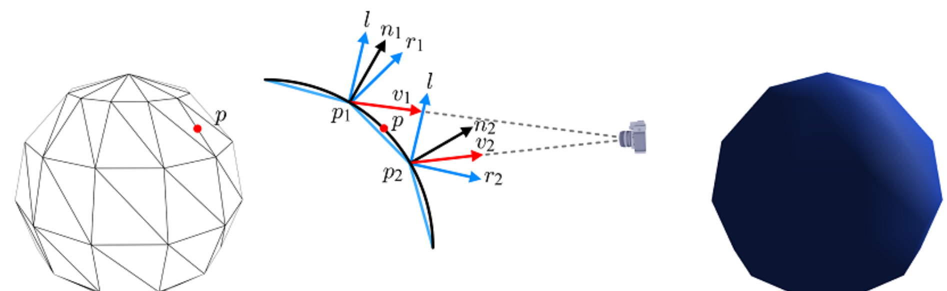
```
float4 FS_PhongPerVertex(float4 Color : COLOR) : COLOR {
  return Color;
}
```

Problems in Per-vertex Lighting

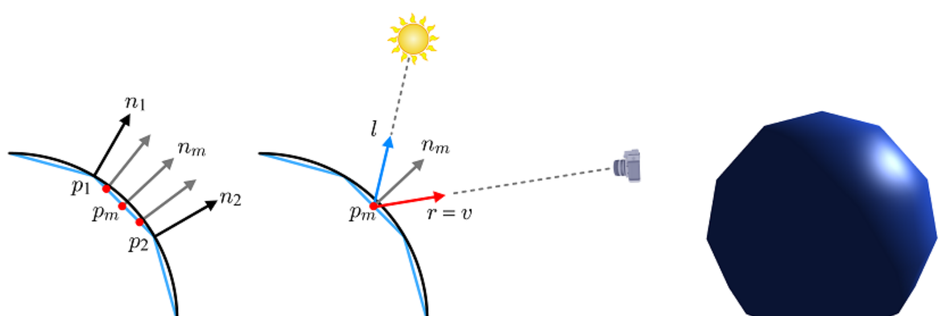
- Per-vertex lighting followed by "color interpolation" makes the colors at vertices *evenly* interpolated along the edges and then along the scan lines. So, highlights inside a triangle cannot be caught.
- Such a problem of per-vertex lighting can be overcome by employing per-fragment lighting, which is often called a normal interpolation shading.



(a)



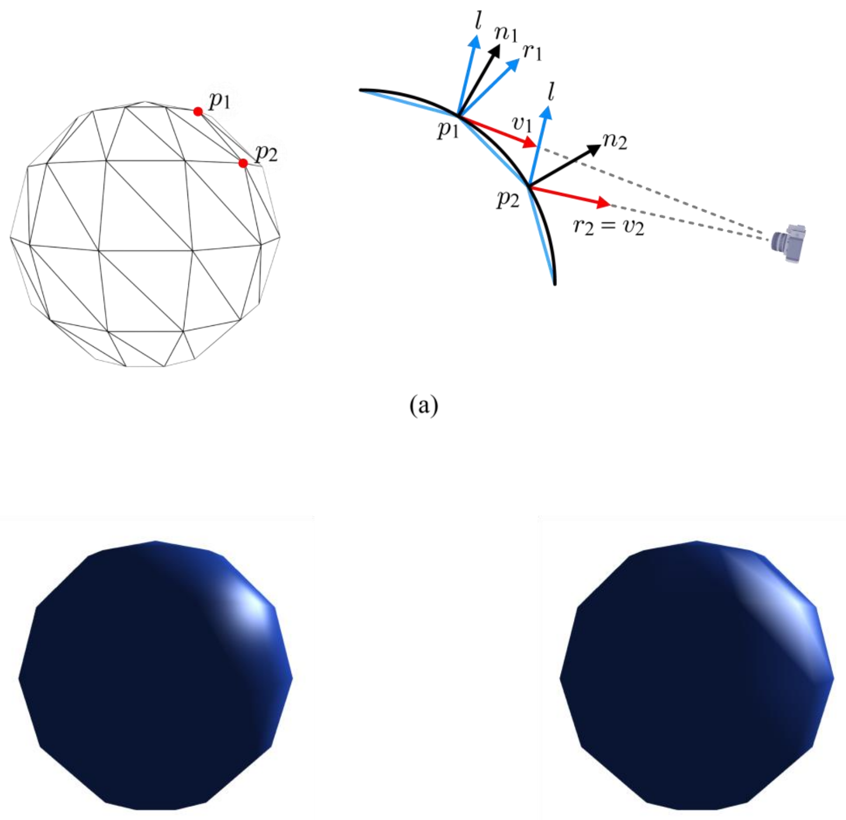
(b)



(c)

Problems in Per-vertex Lighting (cont')

- When the camera moves, the scene rendered by per-vertex lighting can have a popping effect.



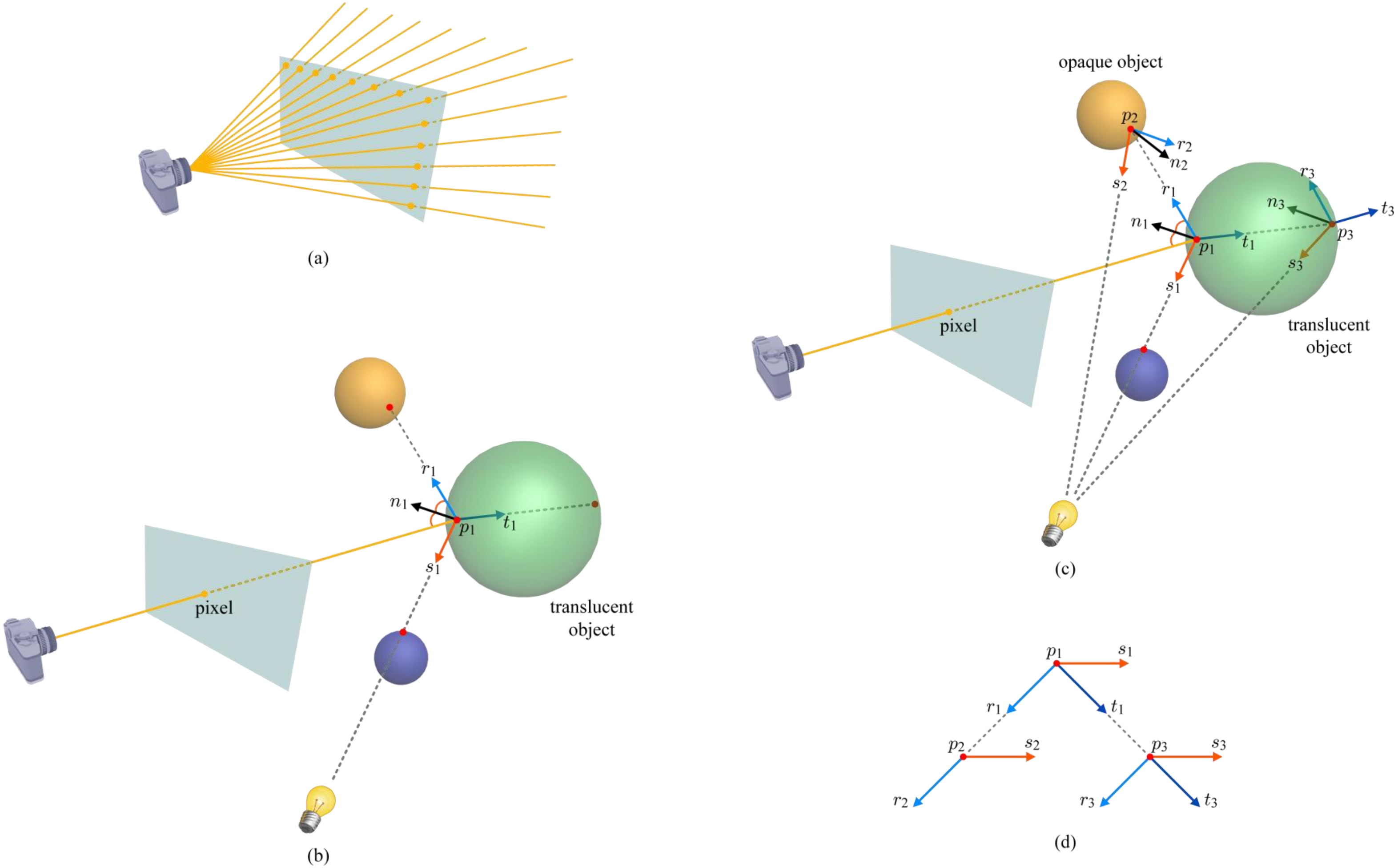
(a)

(b)

(c)

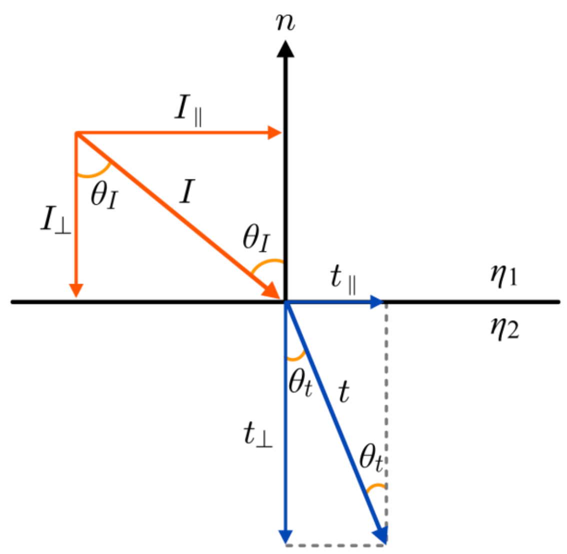
Global Illumination – Ray Tracing

- When a primary ray is shot and intersects an object, three secondary rays would be spawned: a shadow ray, a reflection ray, and a refraction ray.



Global Illumination – Ray Tracing

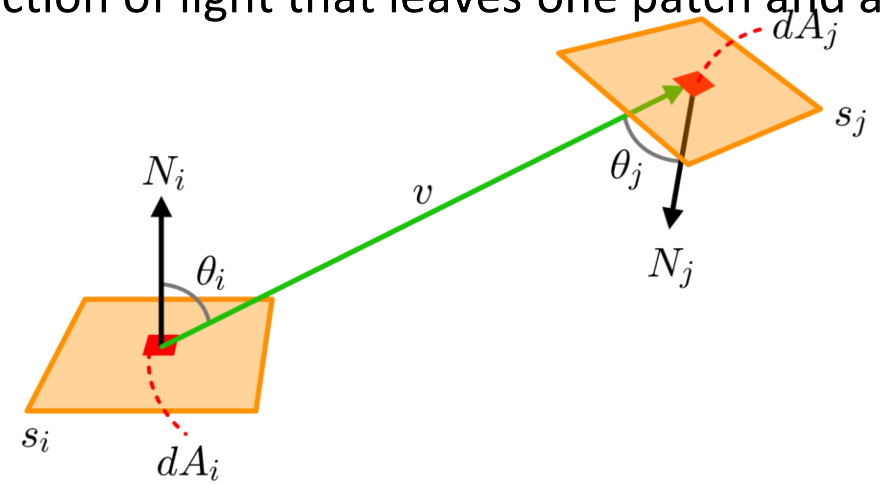
- The refraction ray is computed using the Snell's law.



Global Illumination – Ray Tracing (cont')

Global Illumination – Radiosity

- Radiosity algorithm simulates bounced light between diffuse surfaces. Light hitting a surface is reflected back to the environment. As light bounces around an environment, each surface of the environment works itself as a light source. The radiosity algorithm does not distinguish light sources from the objects to be lit by the light sources.
- Radiosity of a surface describes the brightness of the surface, and is defined to be the rate at which light leaves the surface.
- All surfaces of the scene are subdivided into *patches*, and then the form factors among all the patches are computed. The form factor describes the fraction of light that leaves one patch and arrives at another.

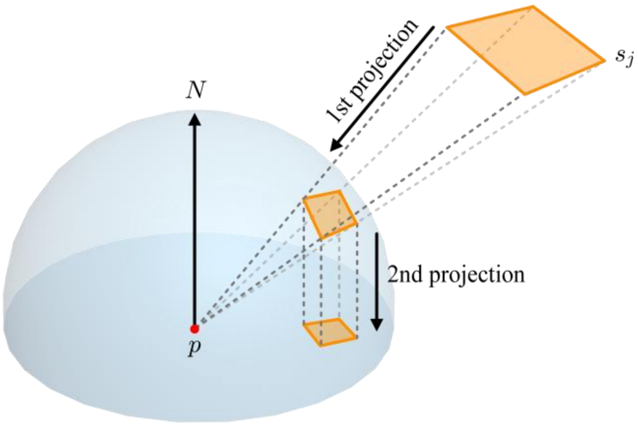


$$\alpha_{ij}\frac{cos\theta_i cos\theta_j}{\pi||v||^2}$$

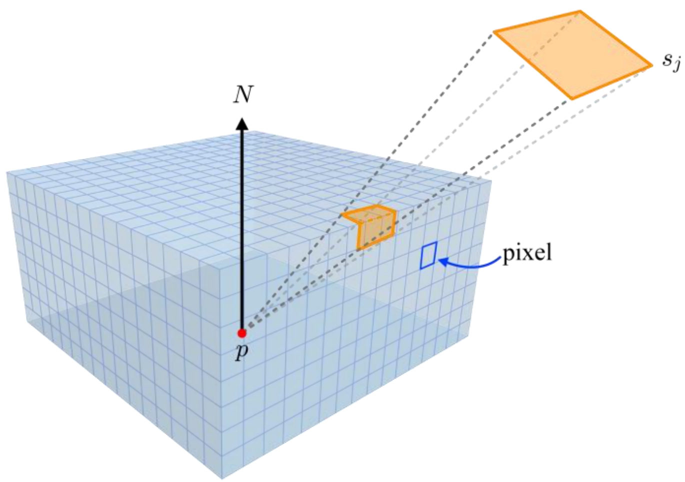
Global Illumination – Radiosity (cont')

Global Illumination – Radiosity (cont')

- In principle, the point-to-point form factor needs to be *integrated* to define a patch-to-patch form factor. Assuming the radiosities are constant over the extent of a patch, computing the patch-to-patch form factor is reduced to computing a point-to-patch form factor.
- The form factor between *p* and s_j is conceptually calculated using the hemisphere placed at *p*. The form factor is defined to be the area projected on the hemisphere's base divided by the base area.
- The hemisphere is often replaced by the hemicube. The per-pixel form factors can be pre-computed. Then, patch s_j is projected onto the hemicube. The form factor is the sum of the form factors of the pixels covered by the projection.



(a)



(b)