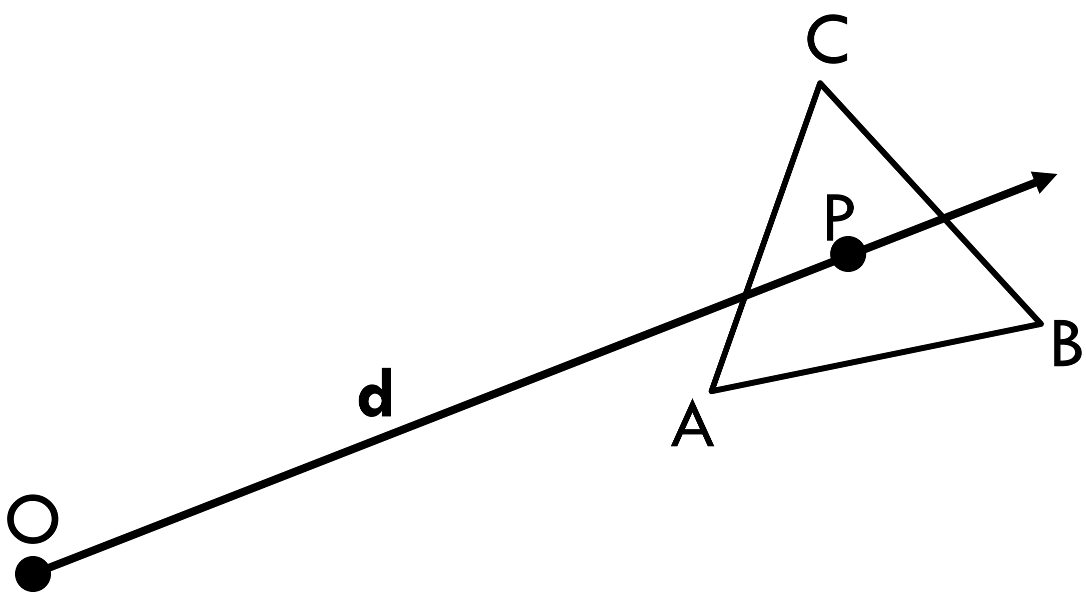
SHADERS, SHADING AND SHADOWS

SEMINAR 3

Computer Graphics 2

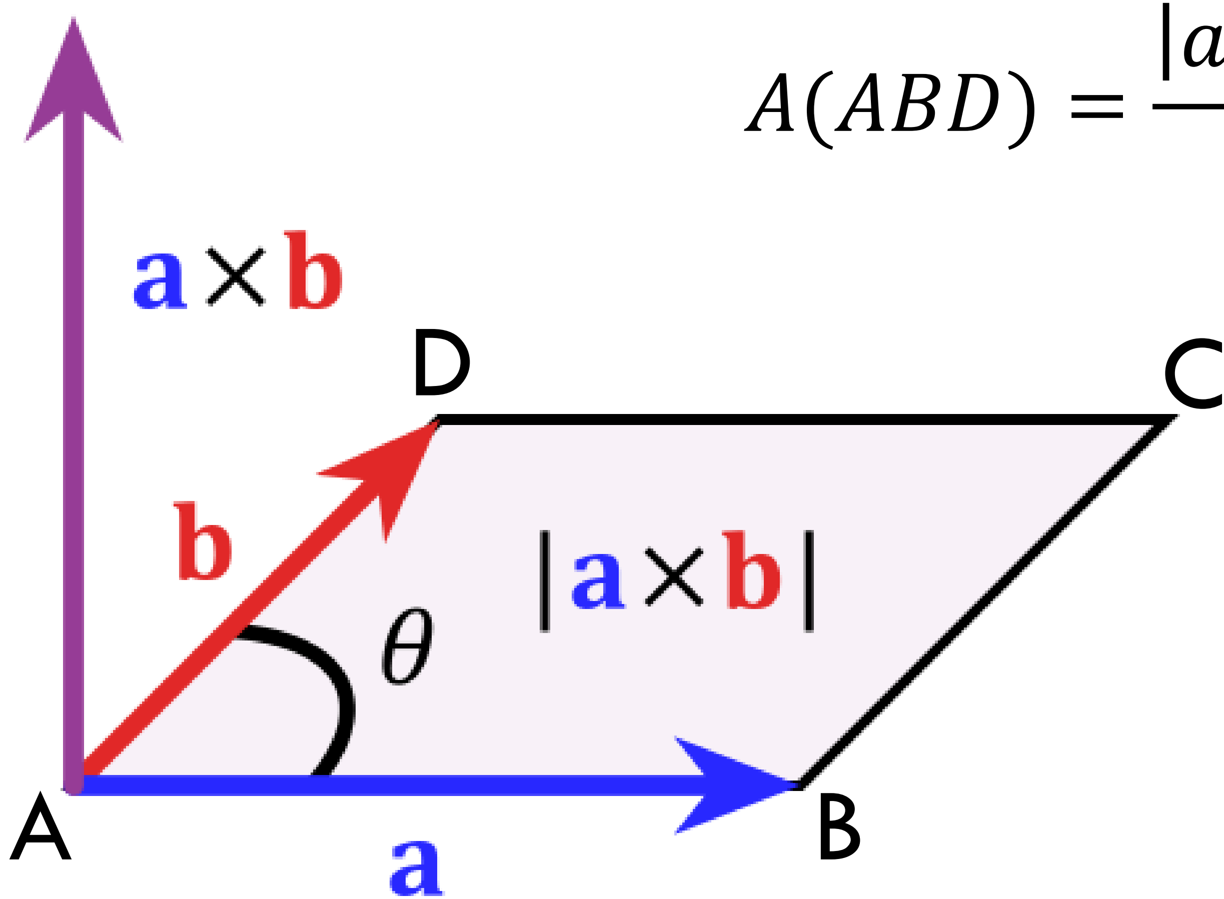
Ray Triangle Intersection

- First calculate u, v – check barycentric coordinates
- With valid barycentric coordinates calculate t
- 0.68s vs 1s in sample scene

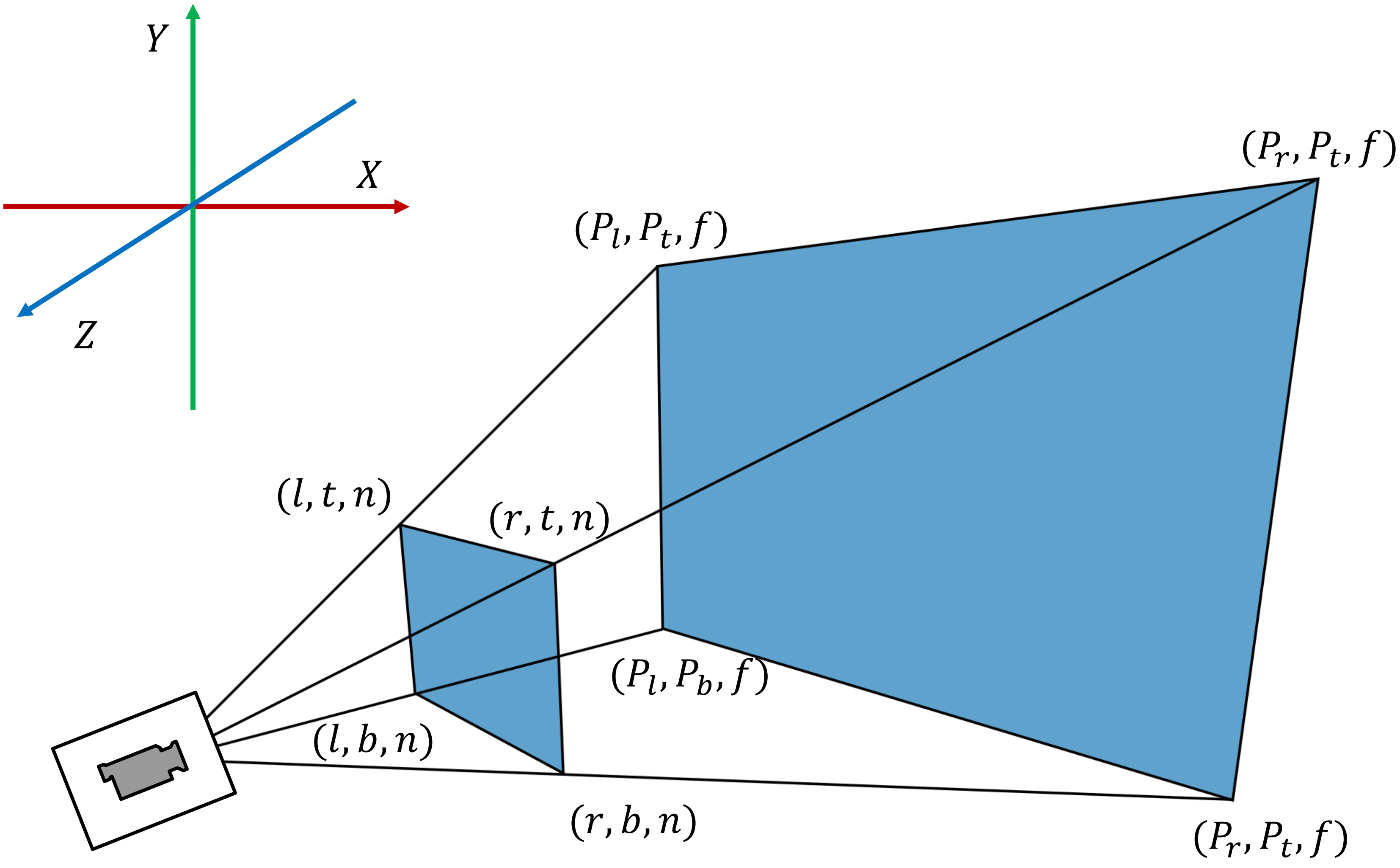


Area Calculation Using Cross Product

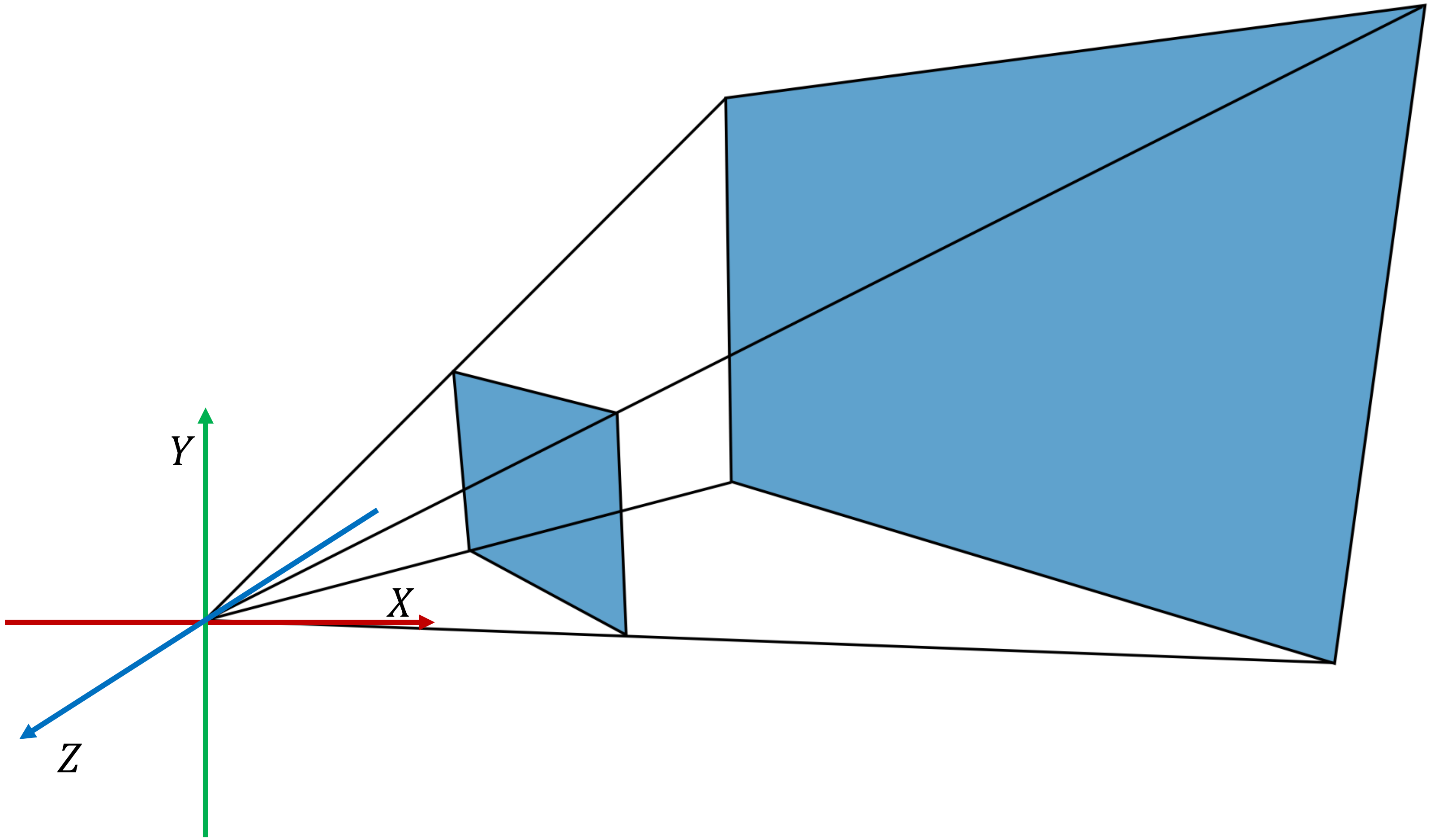
$$A(ABD) = \frac{|a \times b|}{2}$$



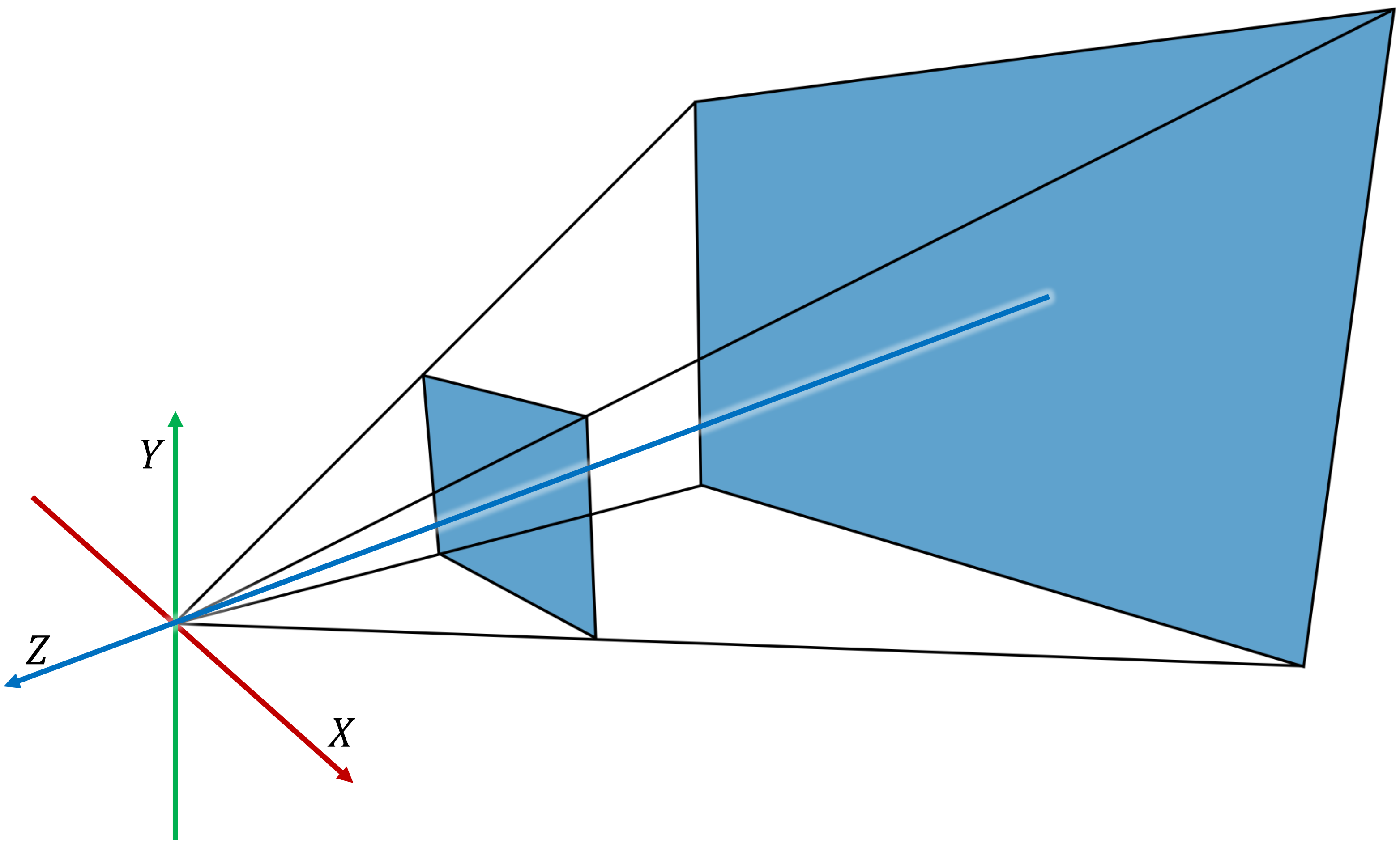
View Frustum



View Frustum Translate



View Frustum Rotate



What's New?

- Ray carries hit normal
- Light
- Shaders

Hit Normal

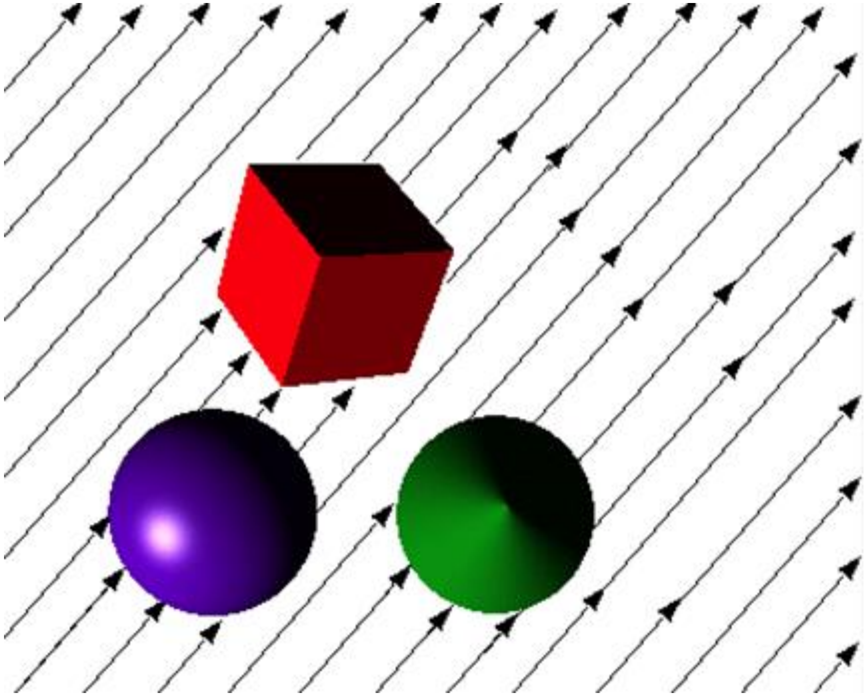
- Normal of objects' surface at intersection point of a ray with an object
  - How to calculate it for plane and sphere?
- Used in calculation of illumination

Light

- Various types of light sources
  - Directional light, spot light, point light, area light
- Each light has
  - Intensity – defines strength with which light illuminates the scene
  - Color – defines the color of the light
    - Diffuse color
    - Specular color
    - Ambient color

Directional Light - Sun

- Infinite distance from the scene
- Light rays emanate in single parallel direction
- Equal intensity in the whole scene

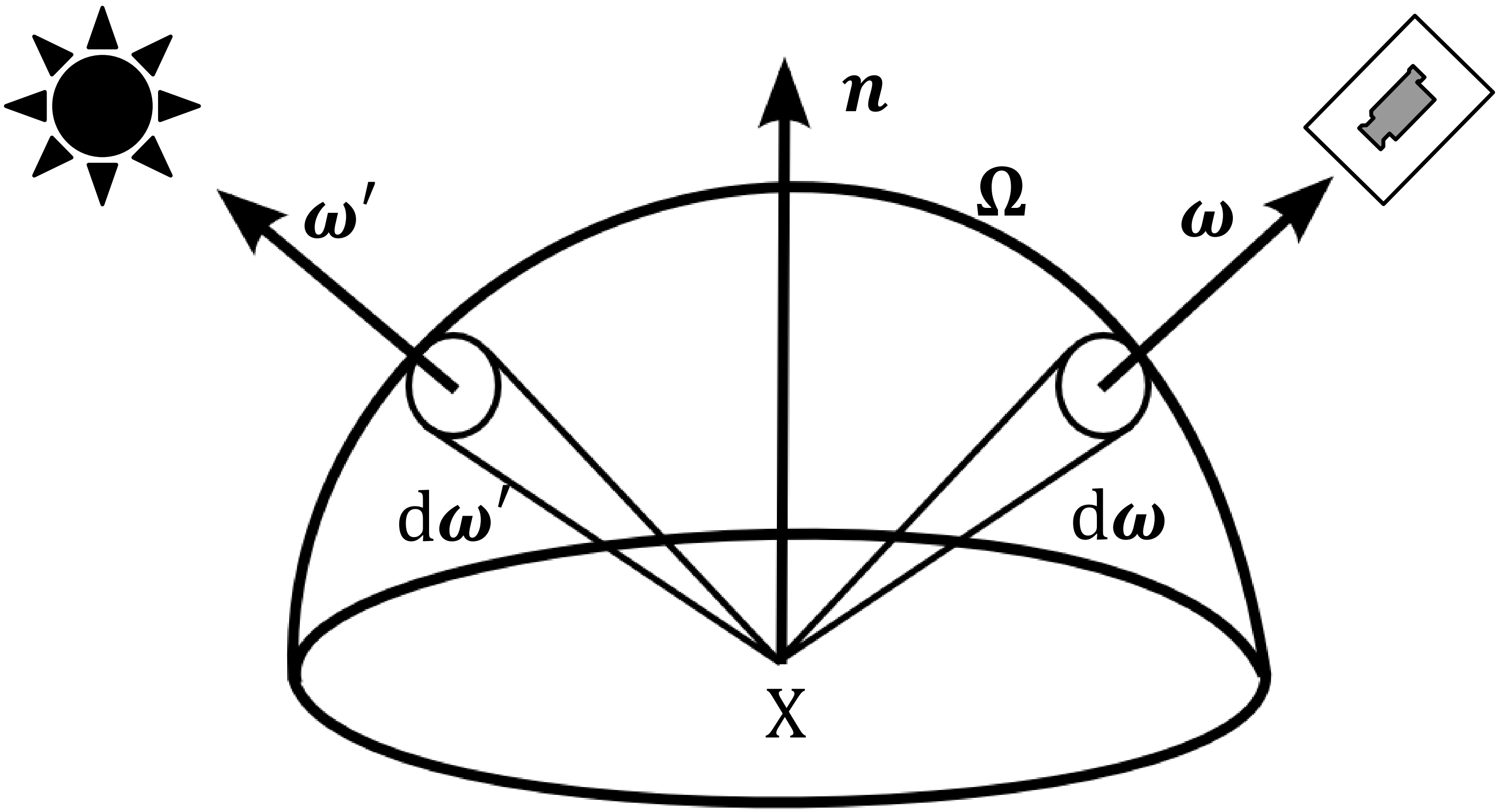


Shader

- Used to define color at a point
- Color is usually calculated using:
  - Point in the scene
  - Normal of points' surface
  - Direction from point to eye
  - Direction from point to light source
  - Light intensity and color at point

Rendering Equation

$$L_0(x,\boldsymbol{\omega}) = L_e(x,\boldsymbol{\omega}) + \int_{\Omega} f_r(x,\boldsymbol{\omega}',\boldsymbol{\omega})L_i(x,\boldsymbol{\omega}')(\boldsymbol{\omega}'\cdot\boldsymbol{n})\mathrm{d}\boldsymbol{\omega}'$$



Bidirectional Reflectance Distribution Function (BRDF)

$$f_r(x, \boldsymbol{\omega}', \boldsymbol{\omega})$$

Positivity:

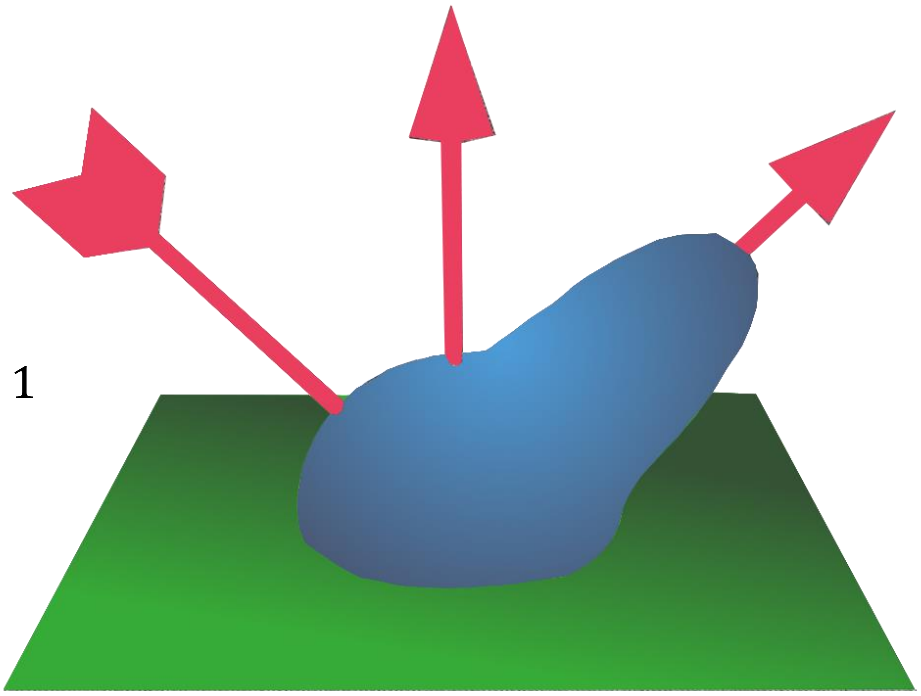
$$f_r(x, \boldsymbol{\omega}', \boldsymbol{\omega}) \geq 0$$

Helmholtz reciprocity:

$$f_r(x, \boldsymbol{\omega}', \boldsymbol{\omega}) = f_r(x, \boldsymbol{\omega}, \boldsymbol{\omega}')$$

Conserving energy:

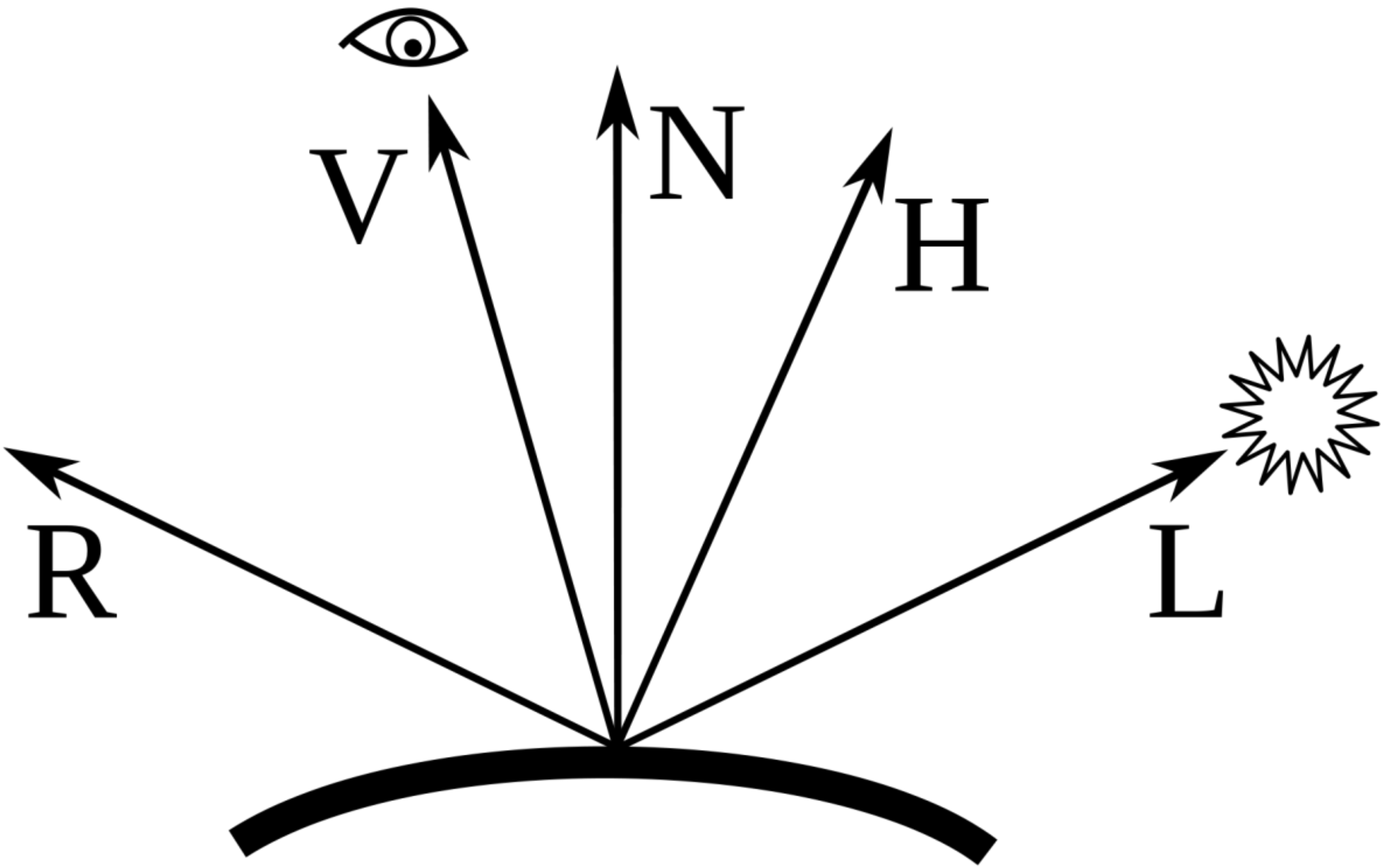
$$\forall \boldsymbol{\omega}', \int_{\Omega} f_r(x, \boldsymbol{\omega}', \boldsymbol{\omega}) L_i(x, \boldsymbol{\omega}')(\boldsymbol{\omega}' \cdot \boldsymbol{n}) \mathrm{d}\boldsymbol{\omega}' \leq 1$$



Phong Shader

- Local illumination model
- Not physically based, does not support:
  - Helmholtz reciprocity
  - Conserving energy
- Split light into components:
  - Ambient – constant for the material
  - Diffuse – depends on position of the light
  - Specular – depends on light and eye position

Phong Shader - Illustration



Phong Ambient

$$I_{ambient} = k_a I_a$$

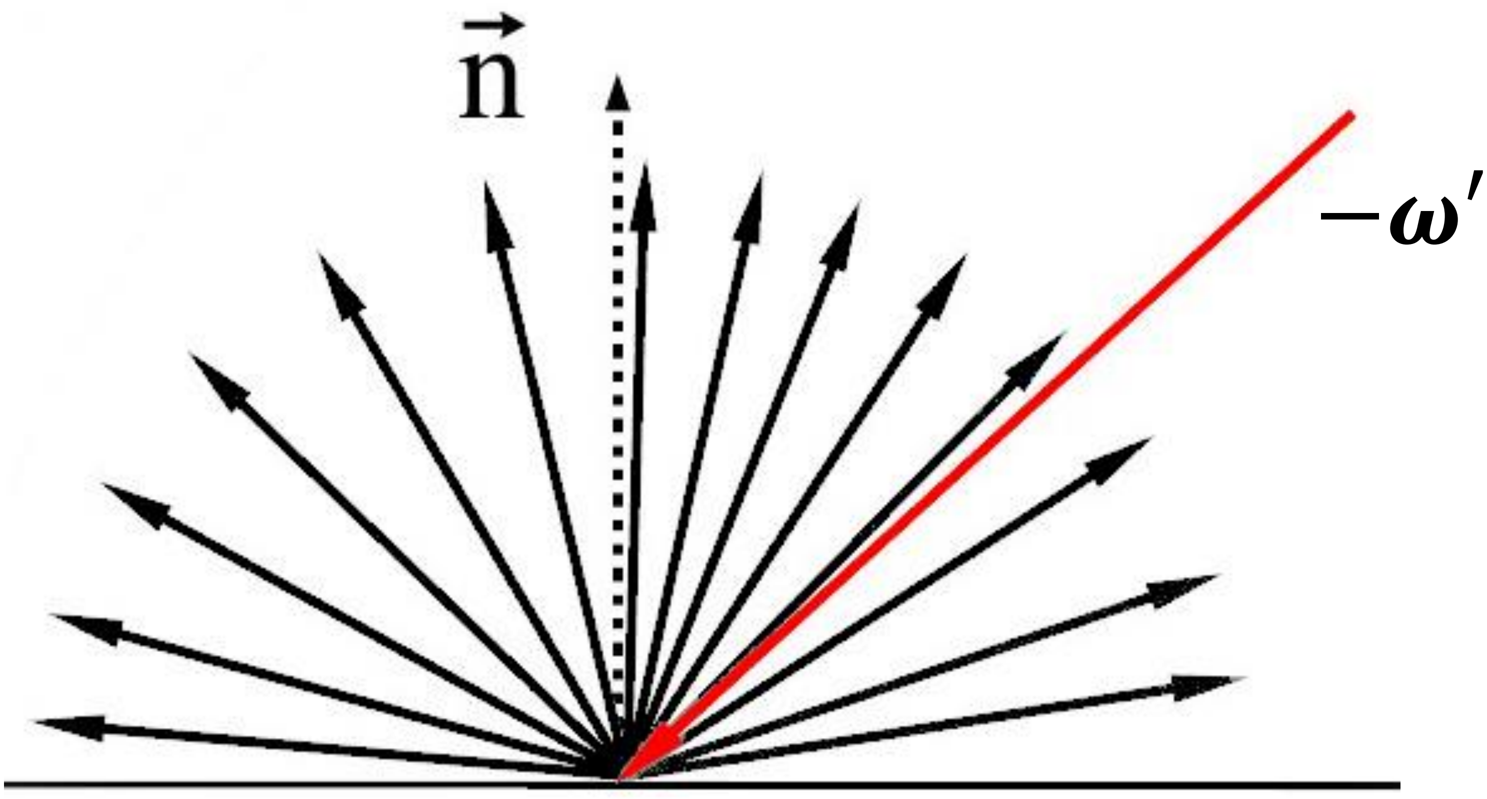
- Simulates light incoming from objects in the scene
- No physical basis – just a constant
- k_a object ambient constant
- I_a ambient light color of a light source

Phong Diffuse

$$I_{diff} = k_d I_d(\boldsymbol{l} \cdot \boldsymbol{n})$$

- Lambertian diffuse reflection
- k_d object diffuse constant
- I_d incoming light diffuse color
  - Scaled by light intensity
- $(\boldsymbol{l} \cdot \boldsymbol{n})$ angle between illuminated point normal and incoming light direction

Phong Diffuse BRDF



Phong Specular

$$I_{spec} = k_s I_l (\boldsymbol{r} \cdot \boldsymbol{v})^{n_s}$$

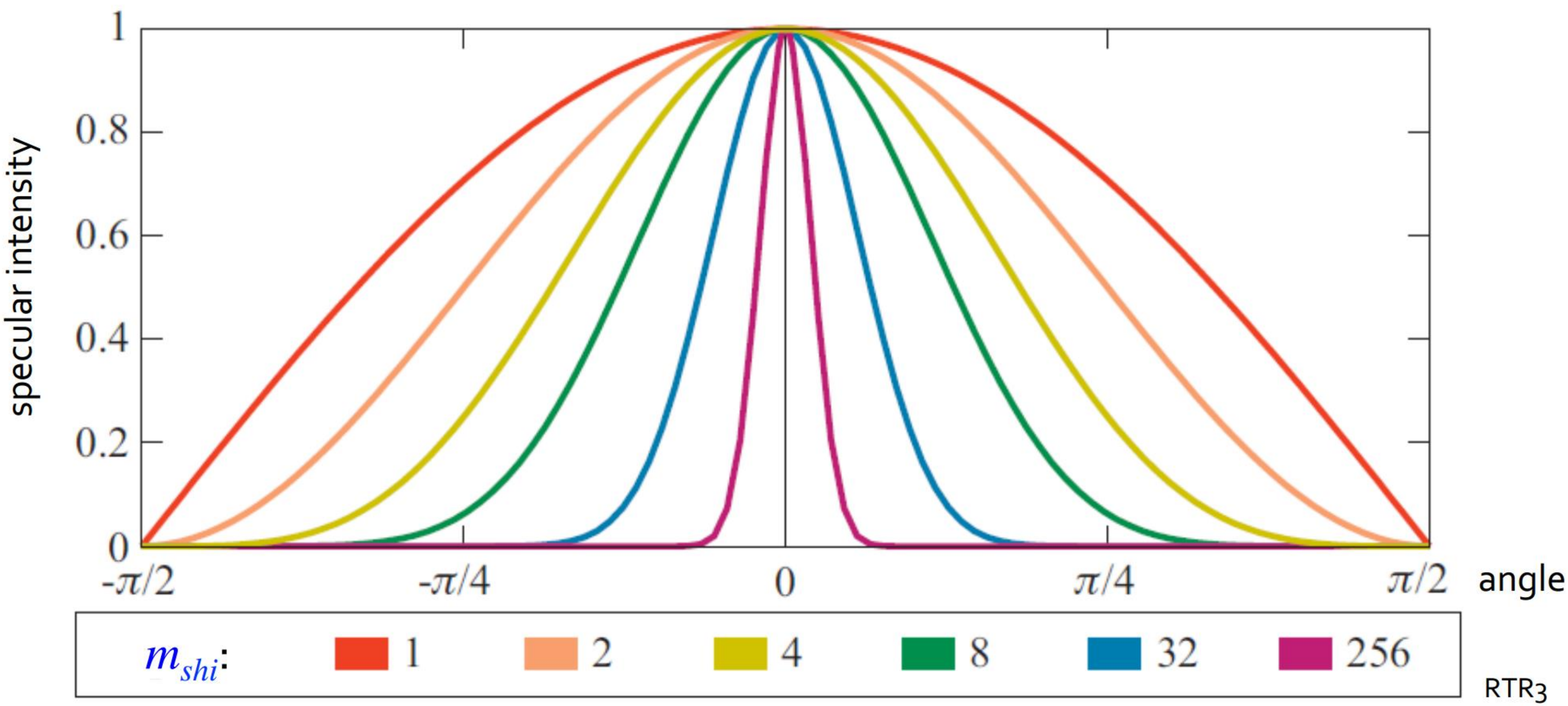
- Specular reflection in direction of perfect glossy reflection
- k_s object specular constant
- I_l incoming light specular color
  - Scaled by light intensity
- $\boldsymbol{r}$ light vector reflected along point normal
- $\boldsymbol{v}$ view direction
- $(\boldsymbol{r} \cdot \boldsymbol{v})$ angle between view direction and reflected vector
- n_s shinines

Blinn-Phong Specular

$$I_{spec} = k_s I_l (\boldsymbol{h} \cdot \boldsymbol{n})^{n_s}$$

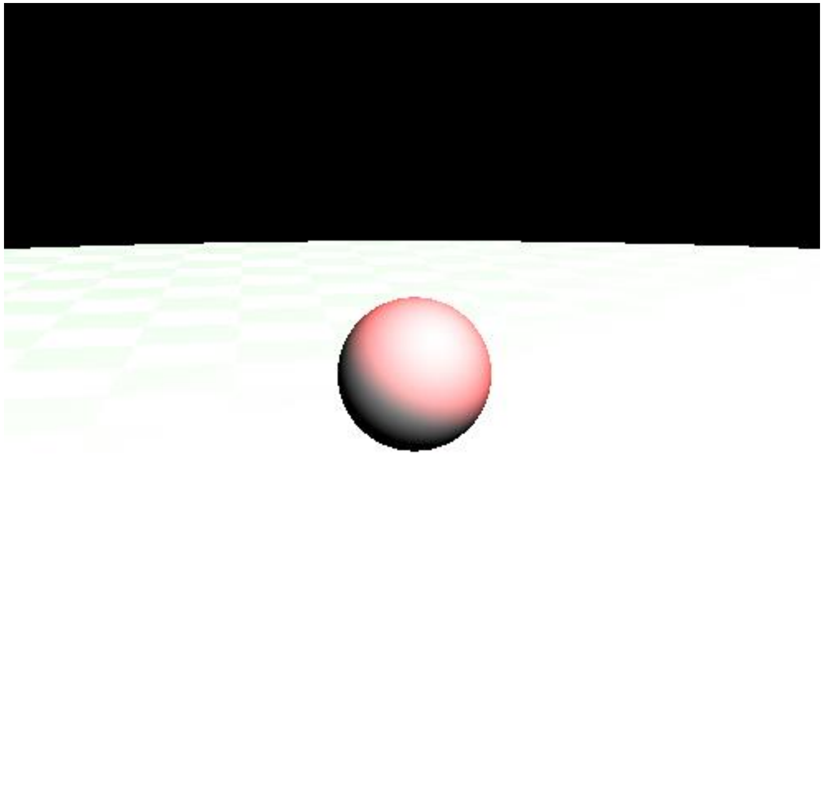
- Specular reflection in direction of perfect glossy reflection
- k_s object specular constant
- I_l incoming light specular color
  - Scaled by light intensity
- $\boldsymbol{h} = \frac{\boldsymbol{l}+\boldsymbol{v}}{|\boldsymbol{l}+\boldsymbol{v}|}$ vector between point normal and incoming light direction
- $(\boldsymbol{h} \cdot \boldsymbol{n})$ angle between illuminated point normal and half vector
- n_s shinines

Phong Specular Component

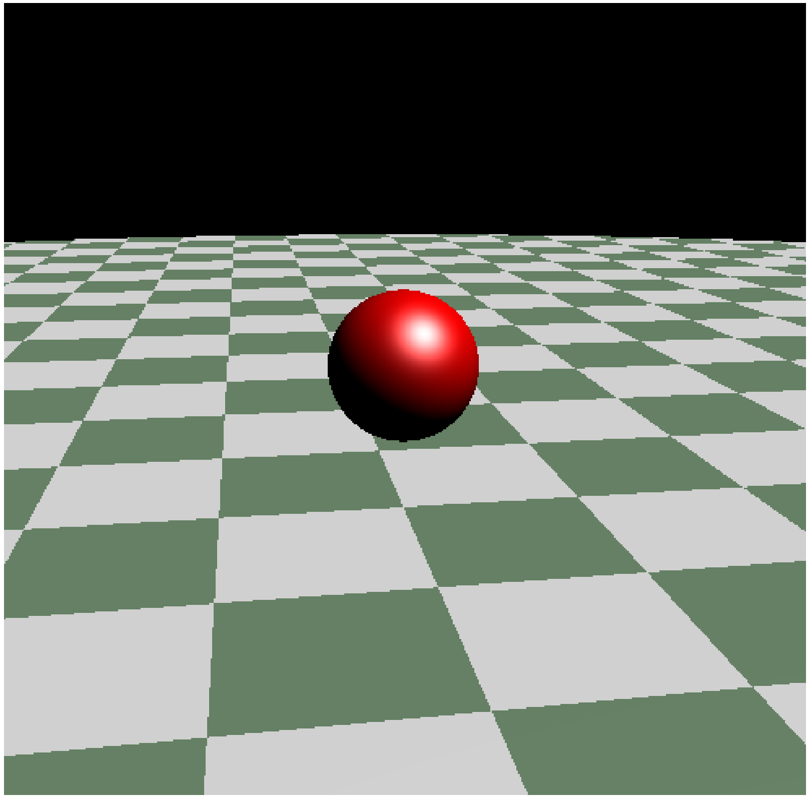


Specular Component Visualization 1

Shininess = 1

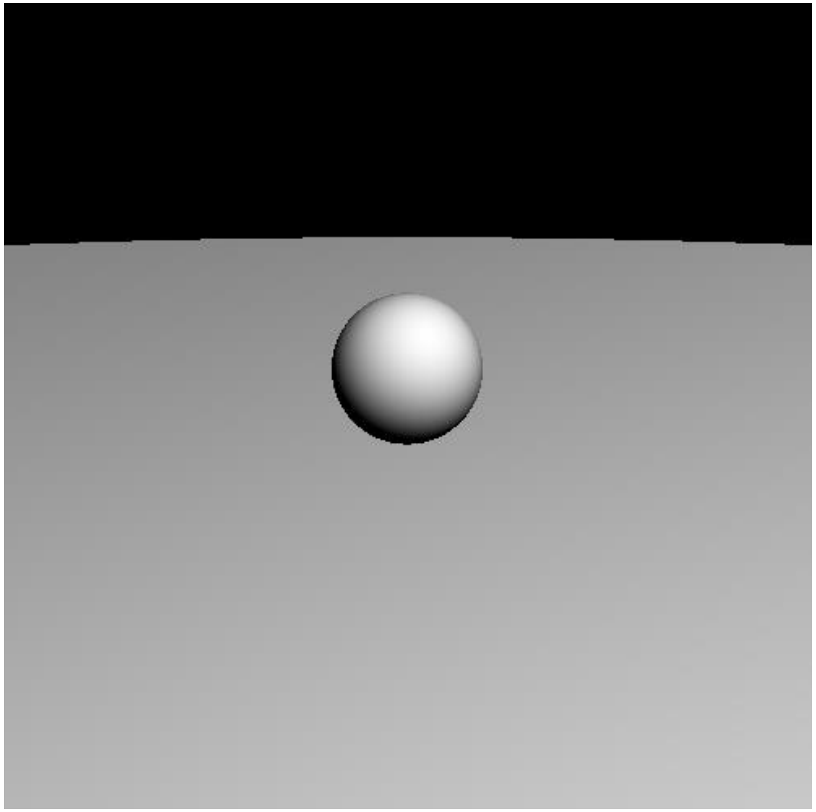


Shininess = 20

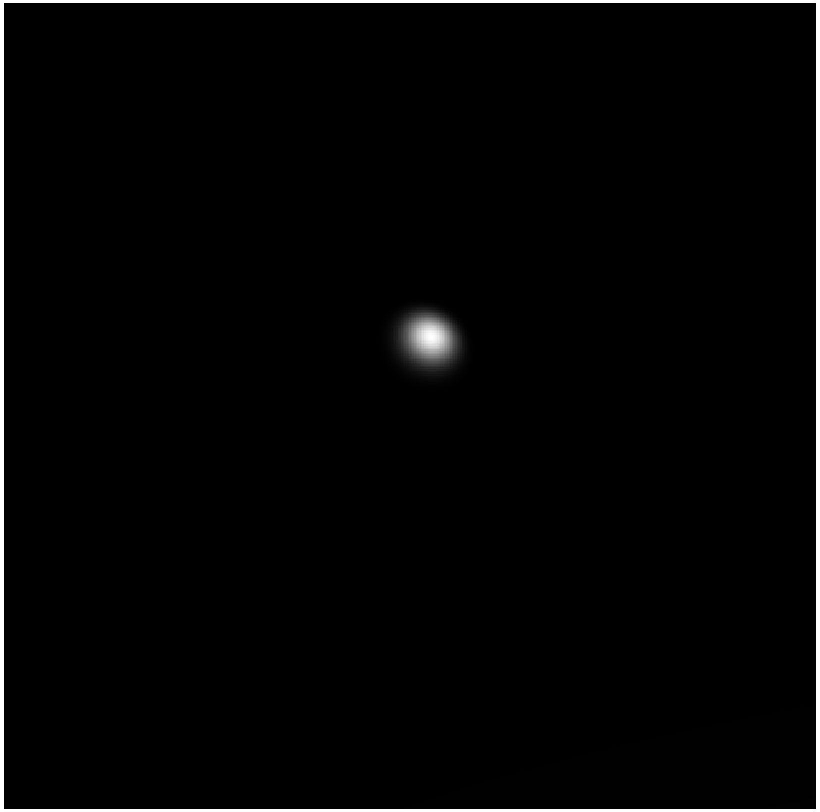


Specular Component Visualization 2

Shininess = 1



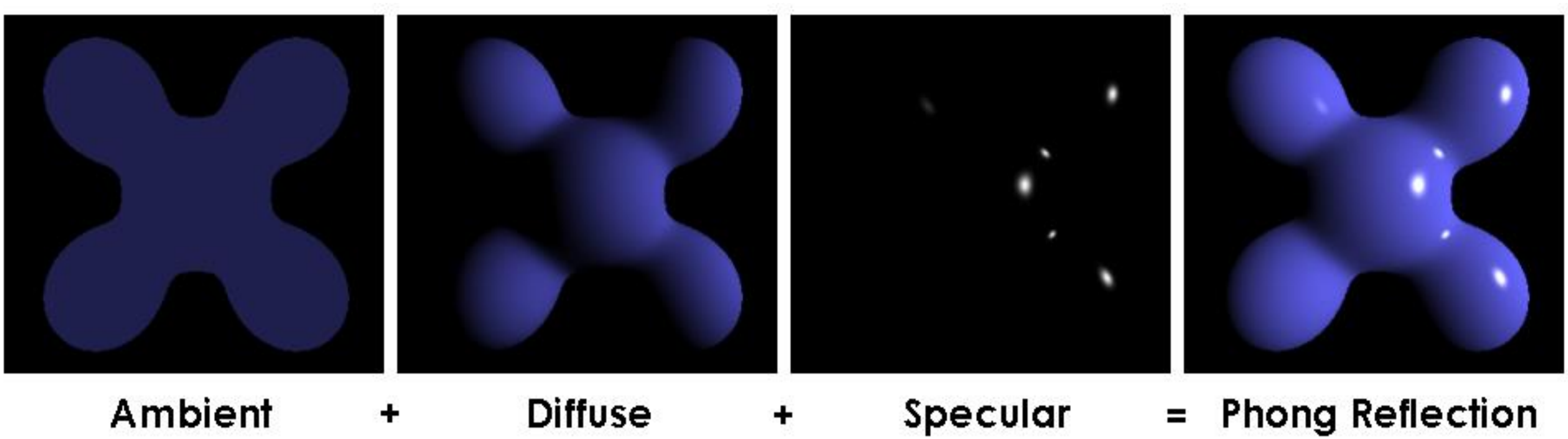
Shininess = 20



Phong Shader – Putting It All Together

$$I = I_{ambient} + I_{diff} + I_{spec} = k_a I_a + k_d I_d (\boldsymbol{l} \cdot \boldsymbol{n}) + k_s I_s (\boldsymbol{h} \cdot \boldsymbol{n})^{n_s}$$

$$I = \sum_{i=1}^{n} (k_a I_{i,a} + k_d I_{i,d} (\boldsymbol{l_i} \cdot \boldsymbol{n}) + k_s I_{i,s} (\boldsymbol{h_i} \cdot \boldsymbol{n})^{n_s})$$



Checker Board Shader

- Consists of two shaders: S0, S1
- Defines cube size s
- Partitions space into cubes
  - Even cubes use S0
  - Odd cubes use S1

$$checker(x) = \begin{cases} S_0, & \lfloor x/s \rfloor \ mod\ 2 = 0 \\ S_1, & otherwise \end{cases}$$

$$checker(x, y, z) = \begin{cases} S_0, & (\lfloor x/s \rfloor + \lfloor y/s \rfloor + \lfloor z/s \rfloor)\ \ mod\ 2 = 0 \\ S_1, & otherwise \end{cases}$$

Questions?