



SHADERS, SHADING AND SHADOWS

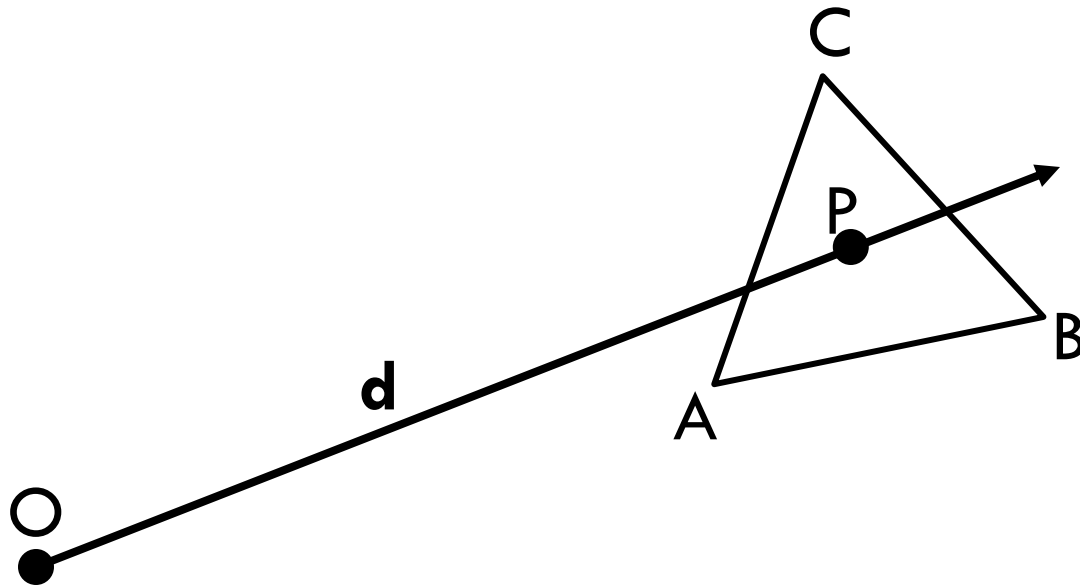
SEMINAR 3

Computer Graphics 2

Ray Triangle Intersection

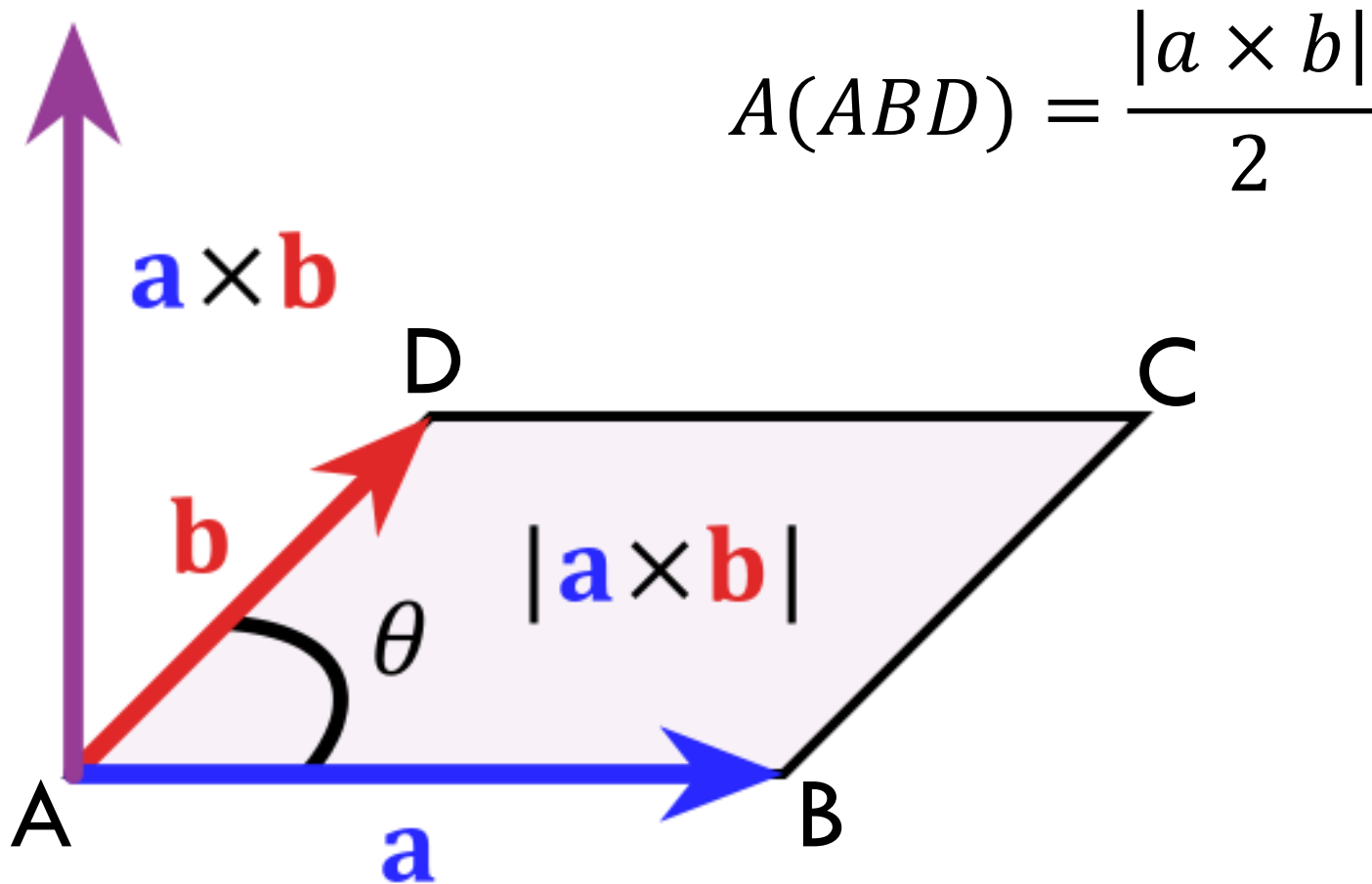
2

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



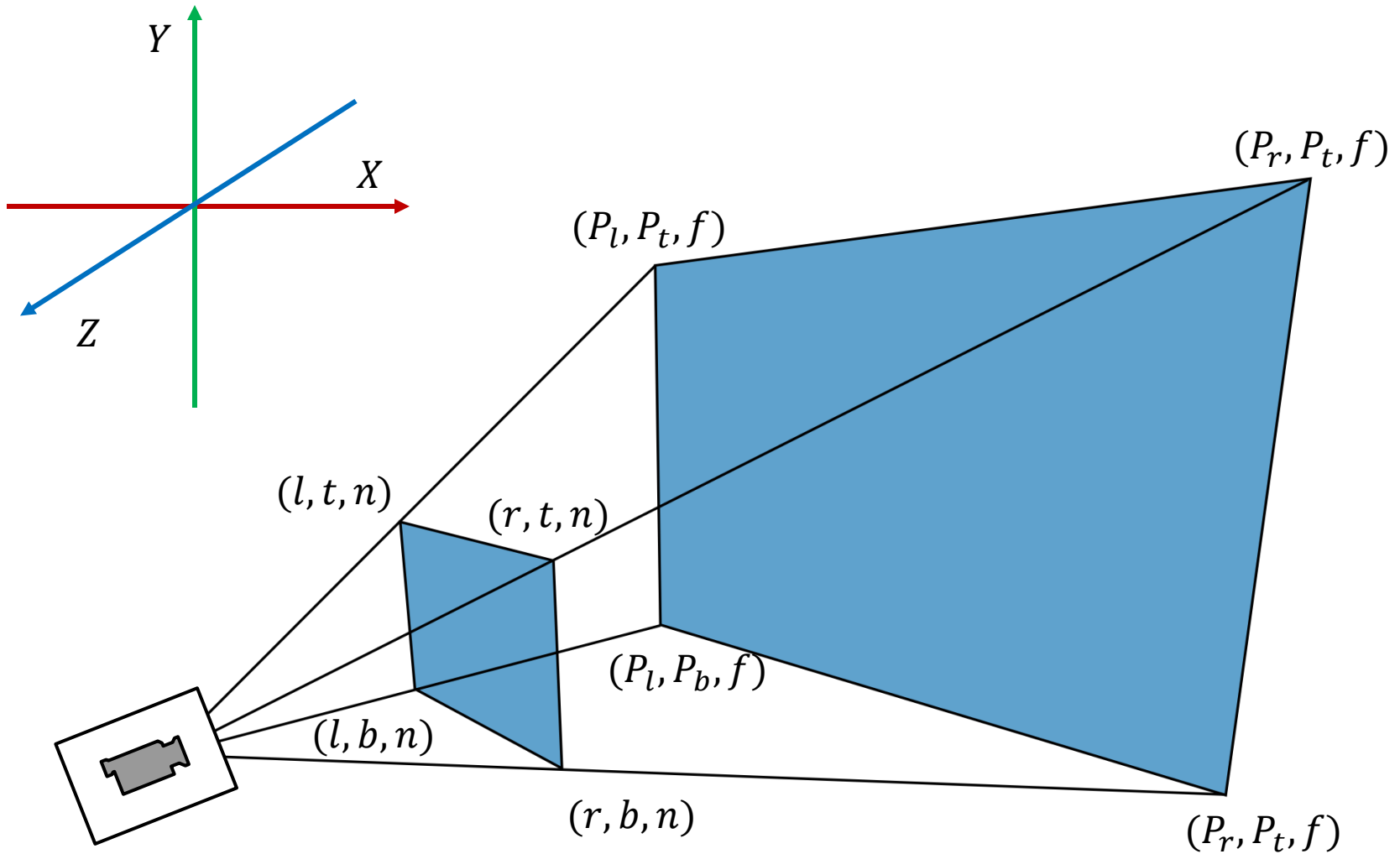
Area Calculation Using Cross Product

3



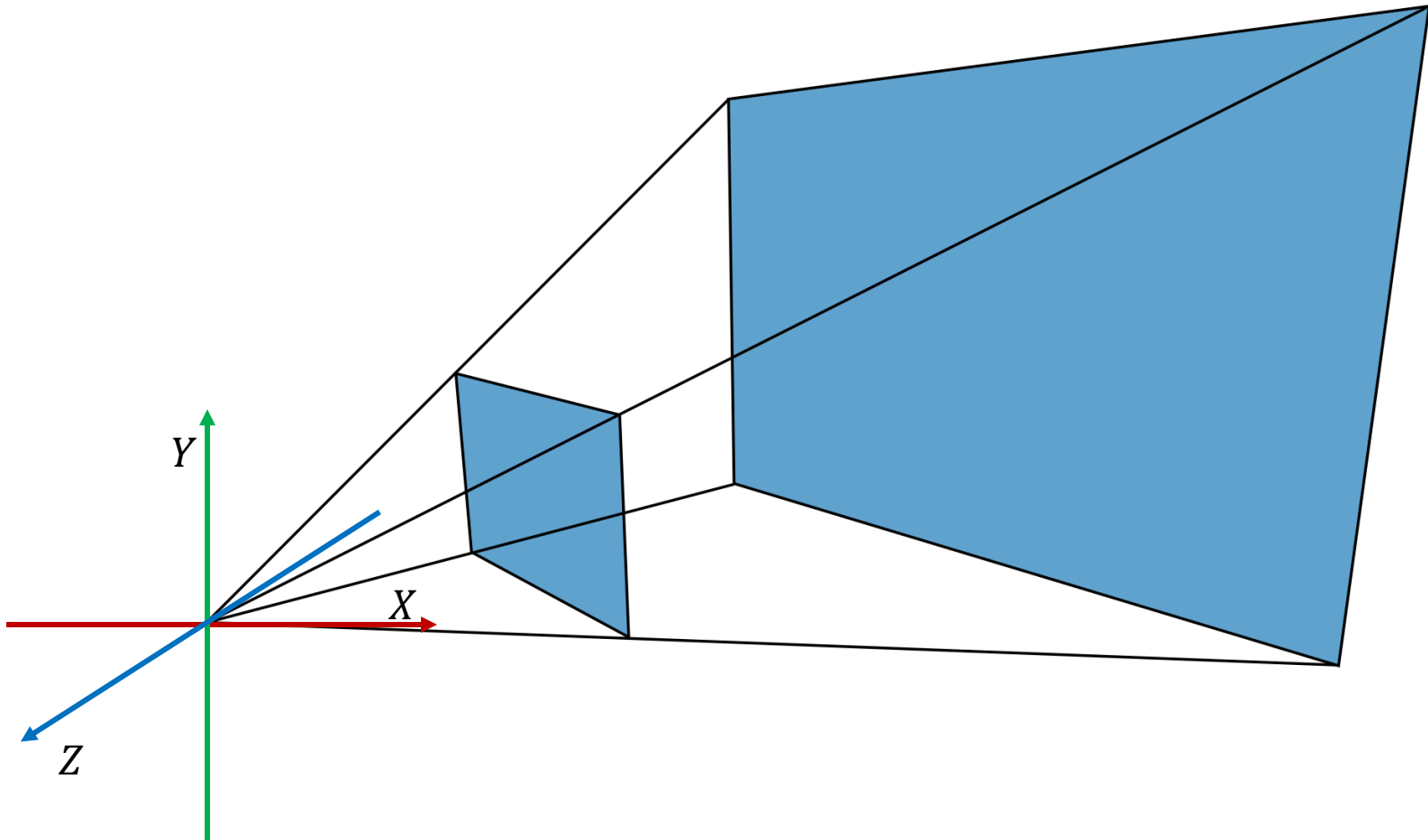
View Frustum

4



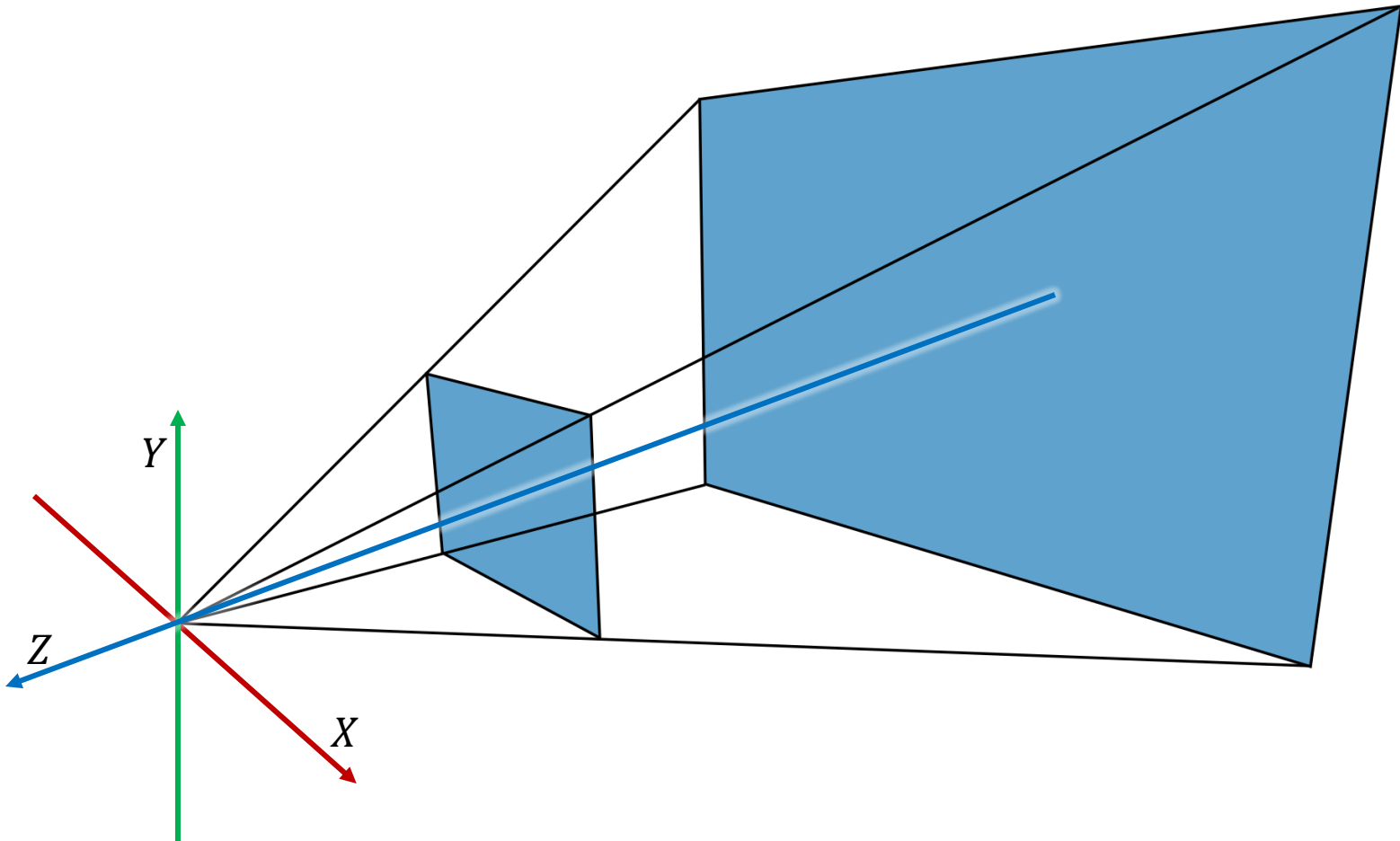
View Frustum Translate

5



View Frustum Rotate

6



What's New?

7

- Ray carries hit normal
- Light
- Shaders

Hit Normal

8

- 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

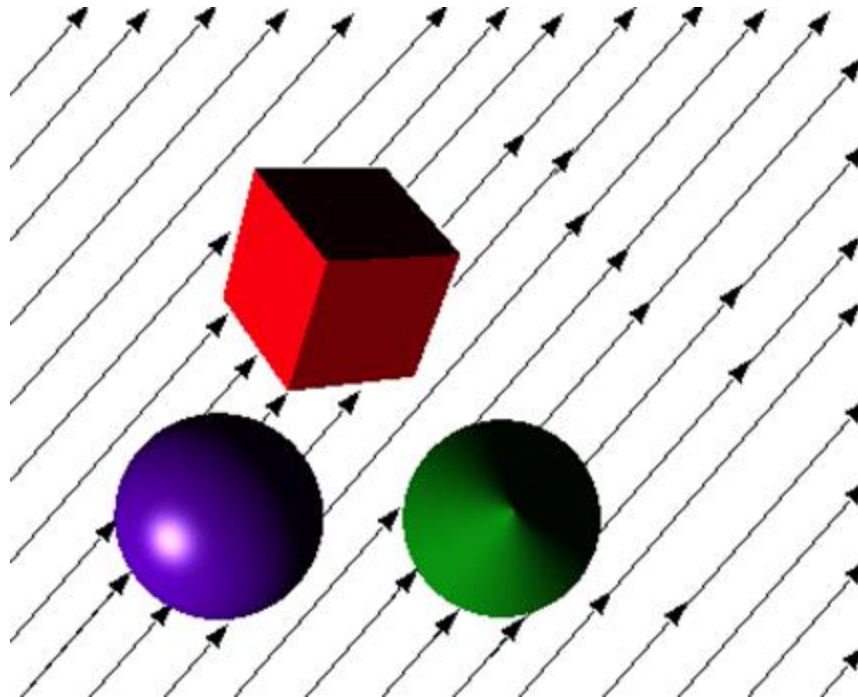
9

- 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

10

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



Shader

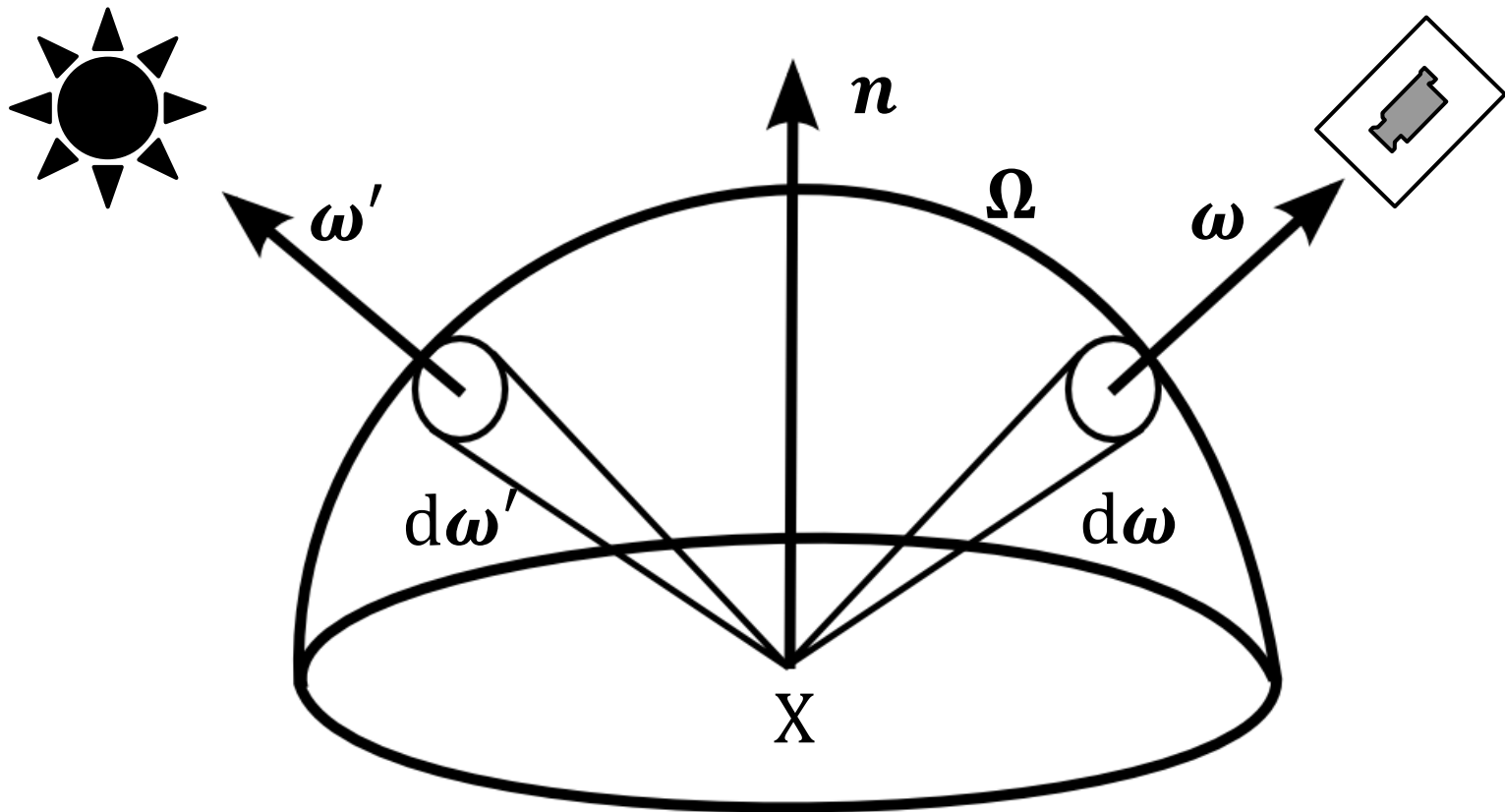
11

- 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

12

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



Bidirectional Reflectance Distribution Function (BRDF)

13

$$f_r(x, \omega', \omega)$$

Positivity:

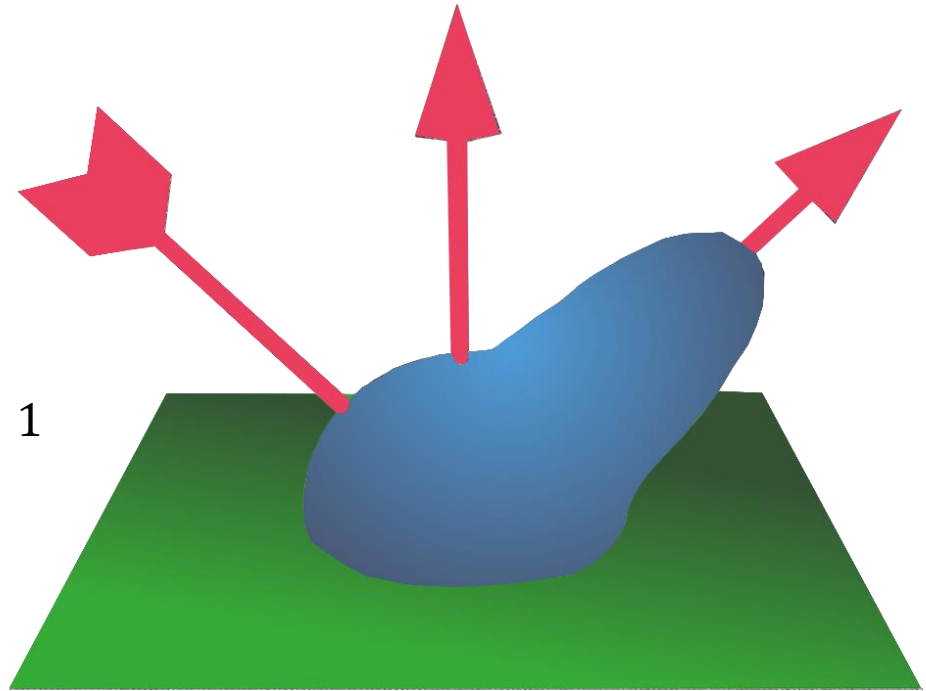
$$f_r(x, \omega', \omega) \geq 0$$

Helmholtz reciprocity:

$$f_r(x, \omega', \omega) = f_r(x, \omega, \omega')$$

Conserving energy:

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



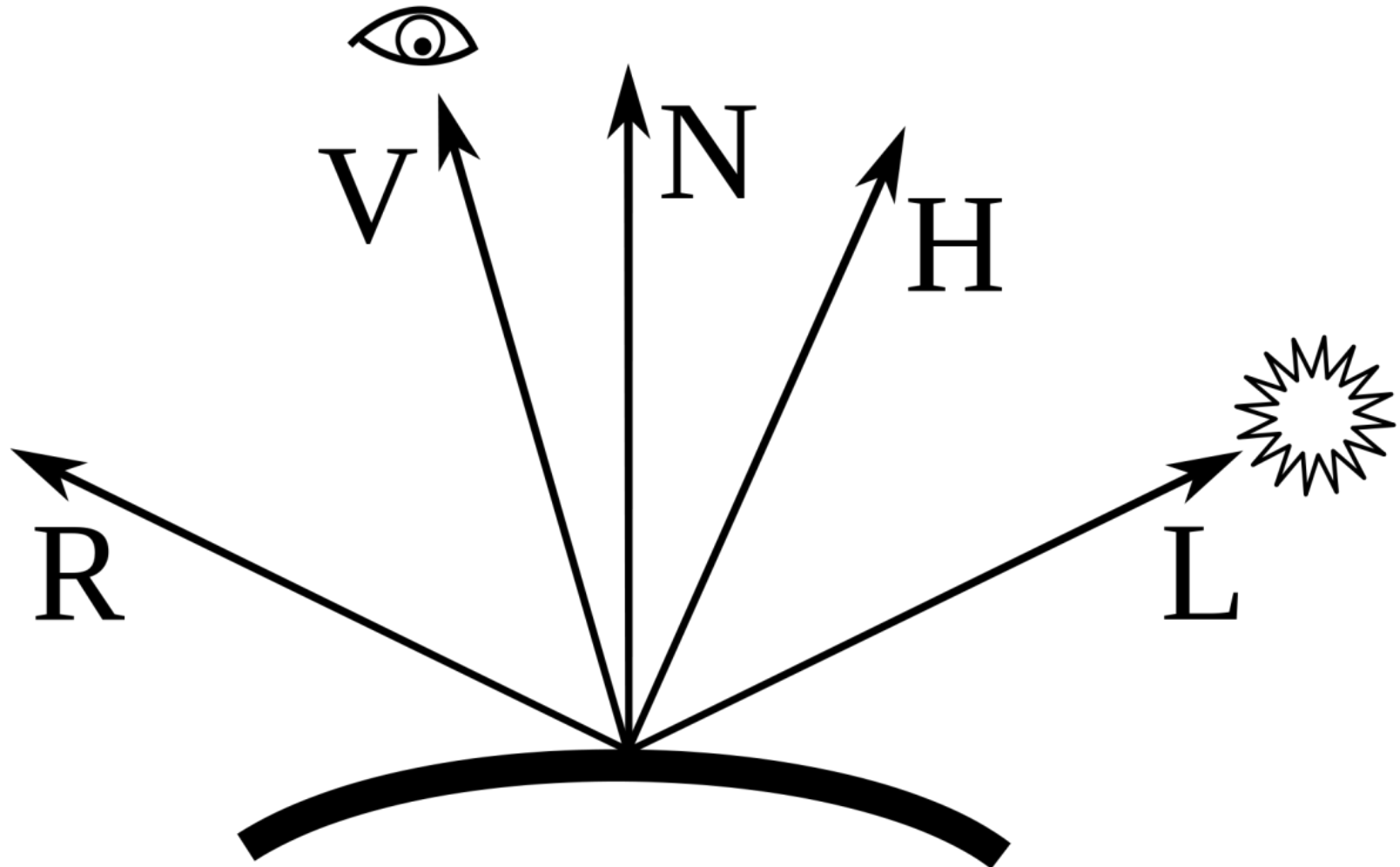
Phong Shader

14

- 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

15



Phong Ambient

16

$$I_{ambient} = k_a I_a$$

- 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

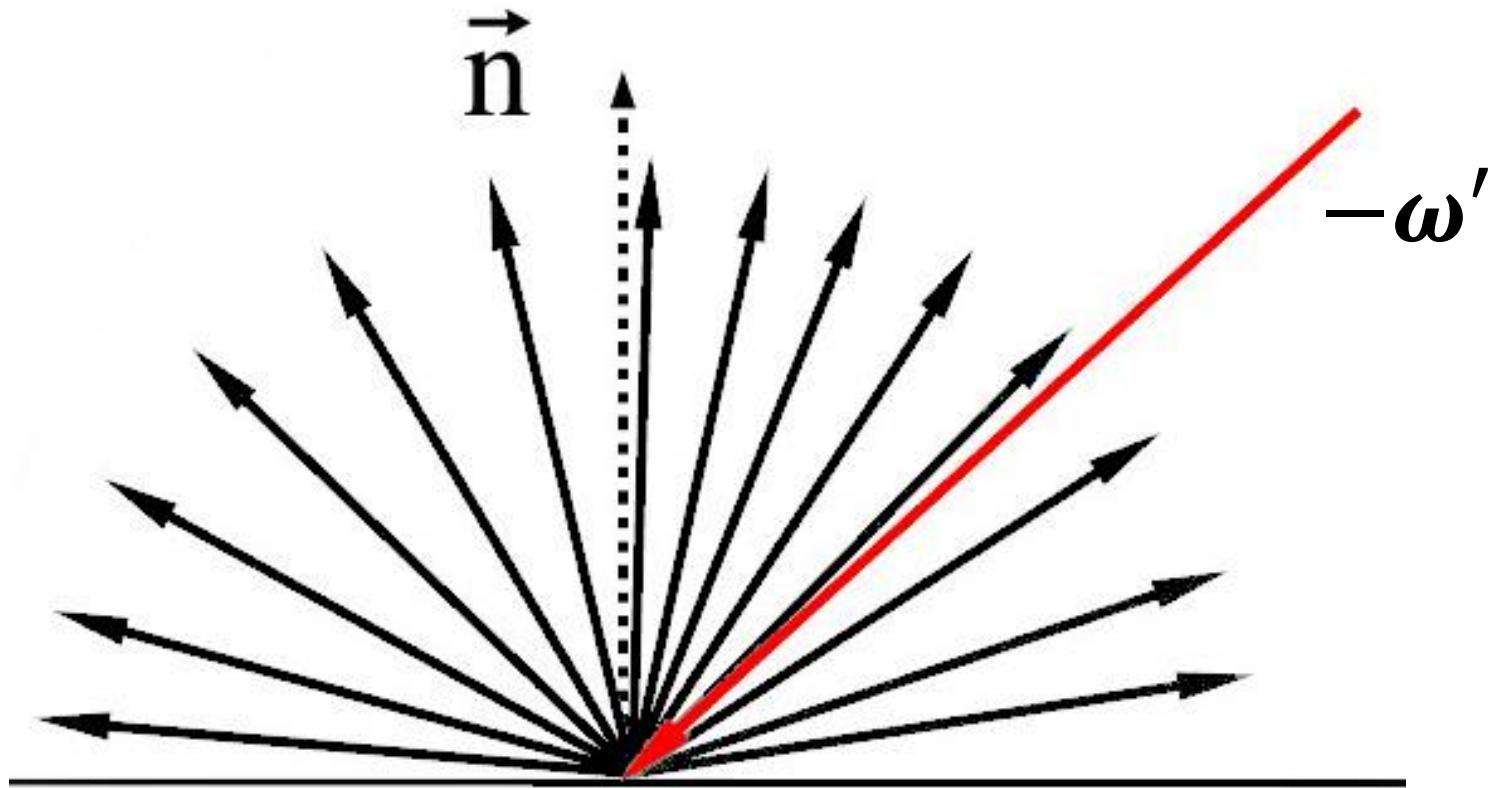
17

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

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

Phong Diffuse BRDF

18



Phong Specular

19

$$I_{spec} = k_s I_l (\mathbf{r} \cdot \mathbf{n})^{n_s}$$

- Specular reflection in direction of perfect glossy reflection
- k_s object specular constant
- I_l incoming light specular color
- \mathbf{r} light vector reflected along point normal
- $(\mathbf{r} \cdot \mathbf{n})$ angle between illuminated point normal and reflected vector
- n_s shininess

Blinn-Phong Specular

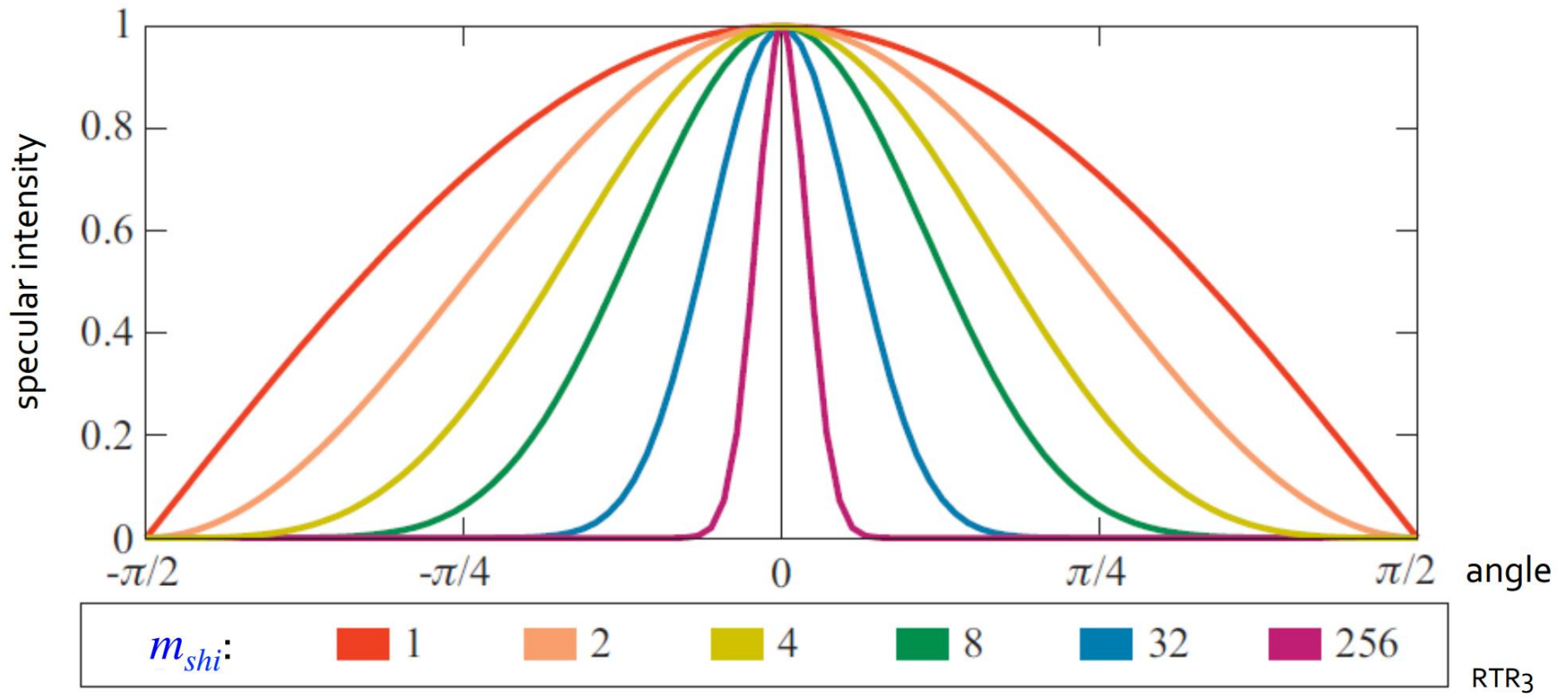
20

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

- Specular reflection in direction of perfect glossy reflection
- k_s object specular constant
- I_l incoming light specular color
- \mathbf{h} vector between point normal and incoming light direction
- $(\mathbf{h} \cdot \mathbf{n})$ angle between illuminated point normal and half vector
- n_s shininess

Phong Specular Component

21

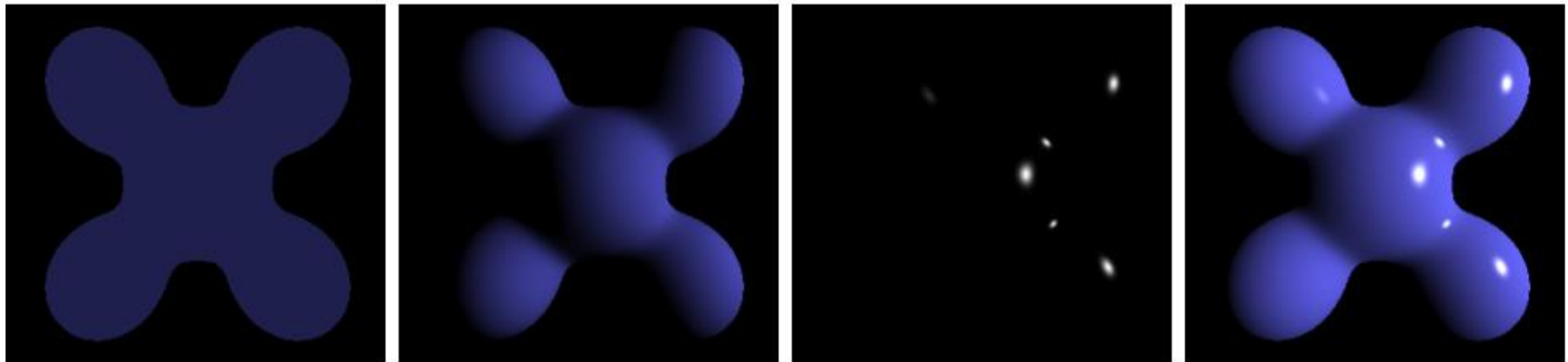


Phong Shader – Putting It All Together

22

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

$$I = \sum_{i=1}^n (k_a I_{i,a} + k_d I_{i,d} (\mathbf{l}_i \cdot \mathbf{n}) + k_s I_{i,s} (\mathbf{h}_i \cdot \mathbf{n})^{n_s})$$



Ambient

+

Diffuse

+

Specular

=

Phong Reflection

Checker Board Shader

23

- Consists of two shaders: Shader0, Shader1
- Defines cube size s
- Partitions space into cubes
 - ▣ Even cubes use Shader0
 - ▣ Odd cubes use Shader1

$$checker(x) = \begin{cases} C_0, & \lfloor x/s \rfloor \bmod 2 = 0 \\ C_1, & \text{otherwise} \end{cases}$$

$$checker(x, y, z) = \begin{cases} C_0, & (\lfloor x/s \rfloor + \lfloor y/s \rfloor + \lfloor z/s \rfloor) \bmod 2 = 0 \\ C_1, & \text{otherwise} \end{cases}$$

24

Questions?