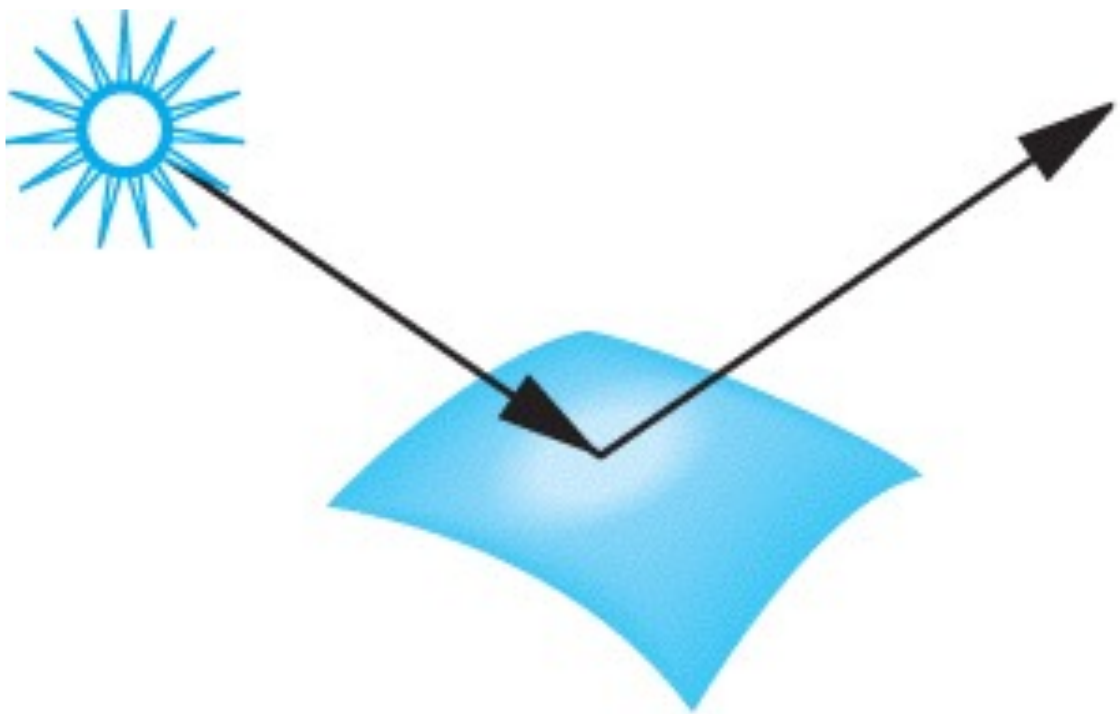
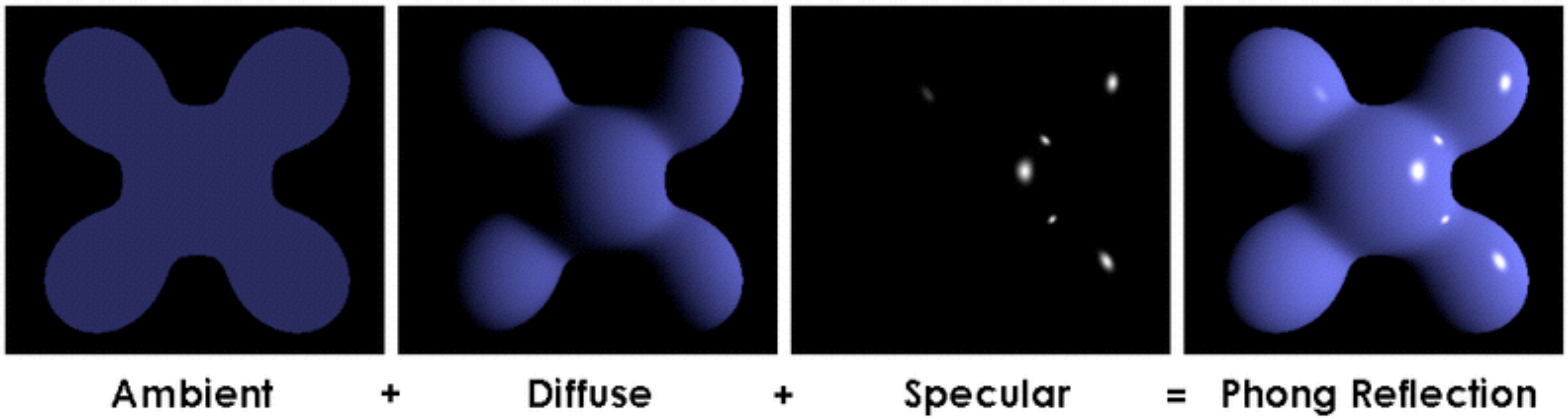
Phong Reflection Model



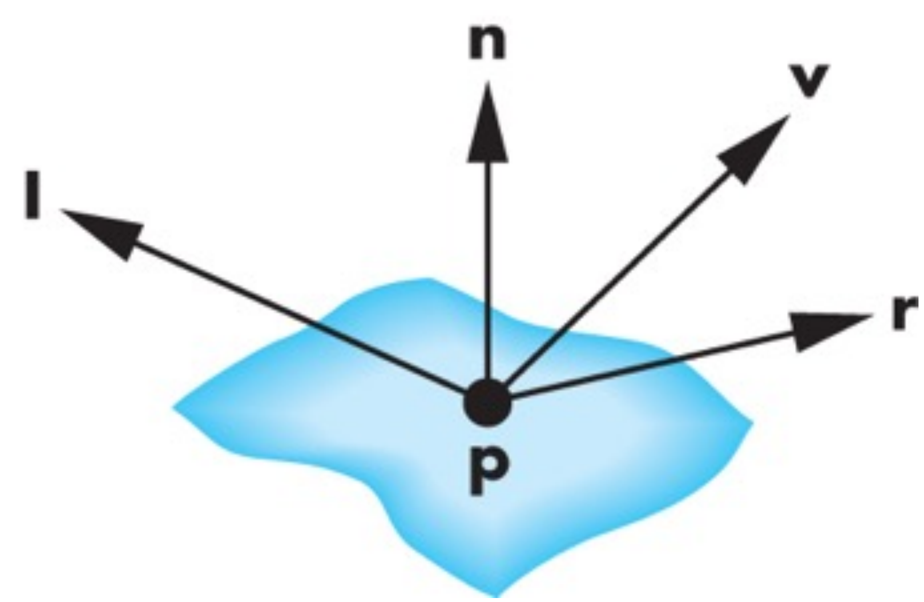
http://www.andreafisherpottery.com/cgi-bin/artistlnk.cgi?Jody_Folwell

Phong Reflection Model

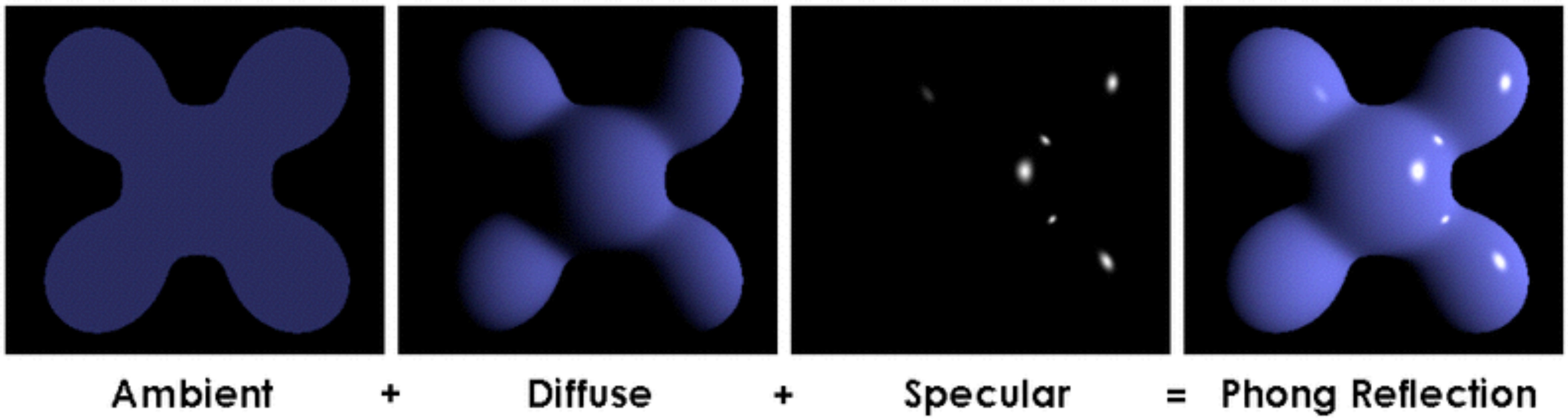


[Brad Smith, Wikimedia Commons]

- efficient, reasonably realistic
- 3 components
- 4 vectors



Phong Reflection Model



[Brad Smith, Wikimedia Commons]

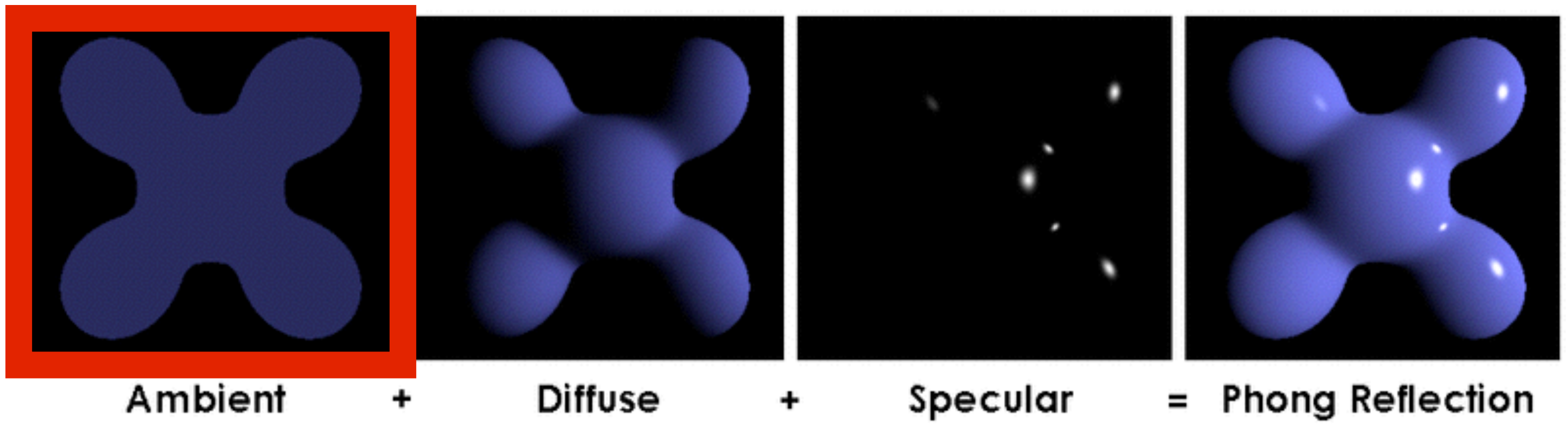
$$
\begin{aligned}
I &= I_a + I_d + I_s \\
&= R_a L_a + R_d L_d \max(0, \mathbf{l} \cdot \mathbf{n}) + R_s L_s \max(0, \cos\phi)^\alpha
\end{aligned}
$$

color intensity

reflectance

illumination

Ambient reflection

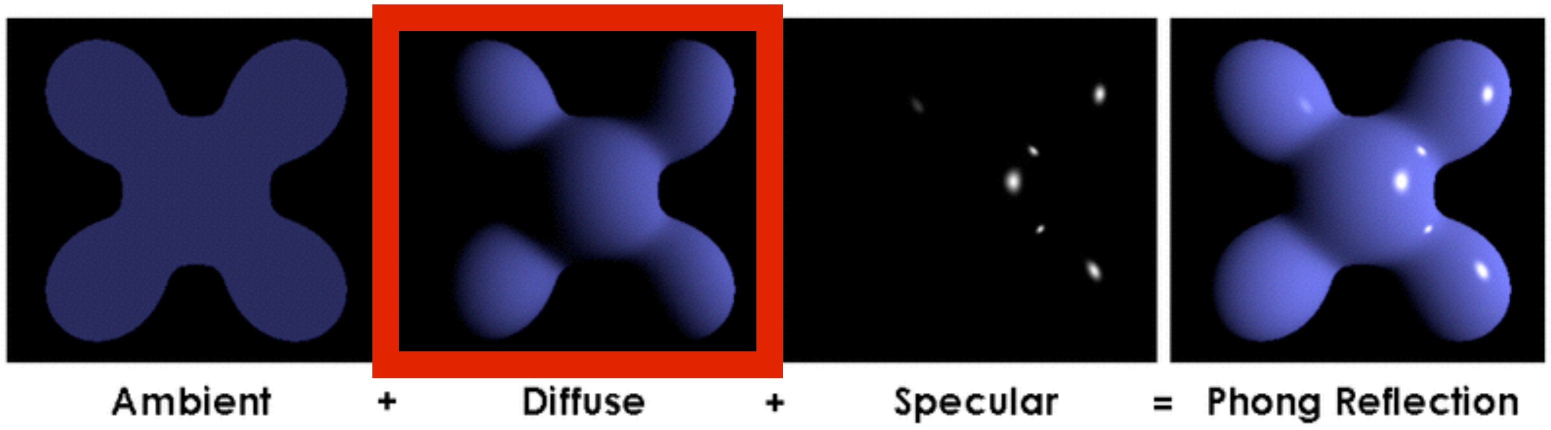


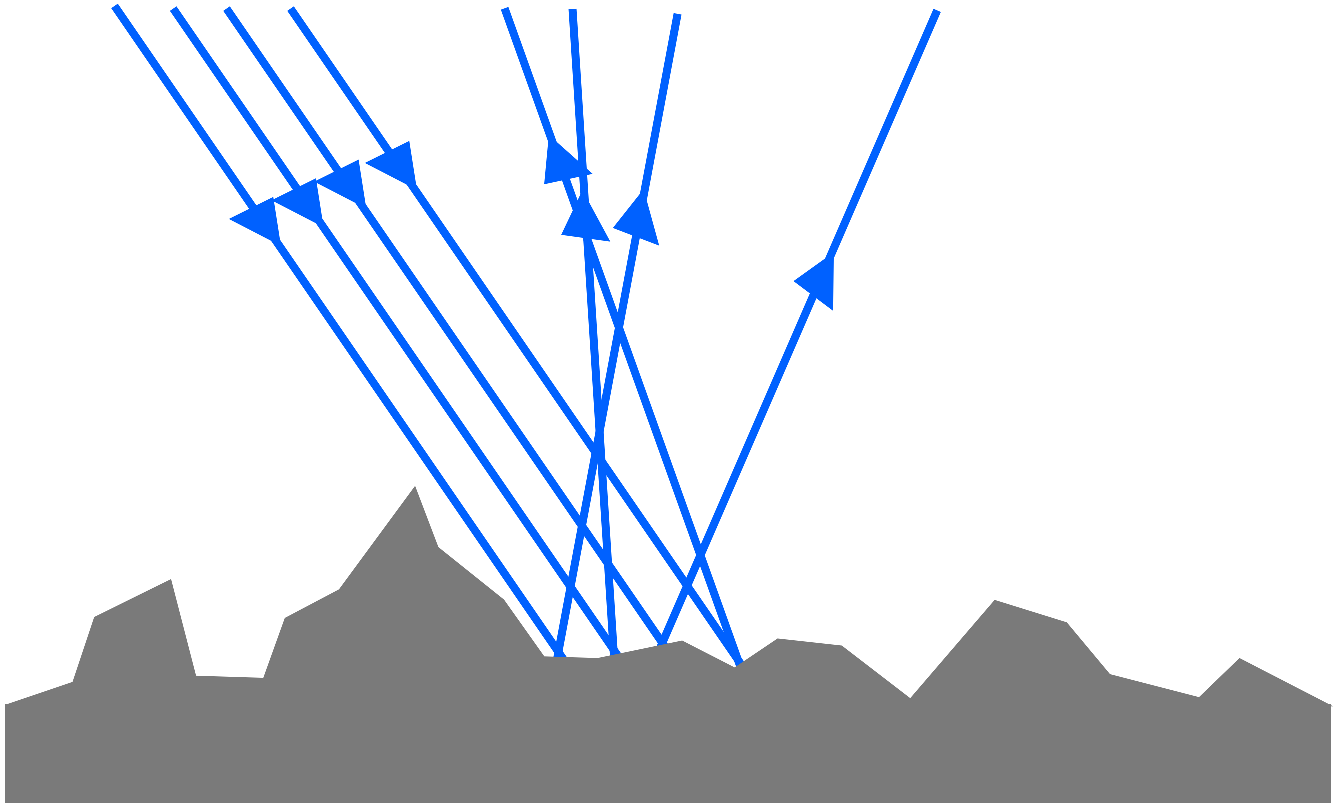
different ambient coefficients for different colors

$$I_a = R_a L_a, \qquad 0 \leq R_a \leq 1$$

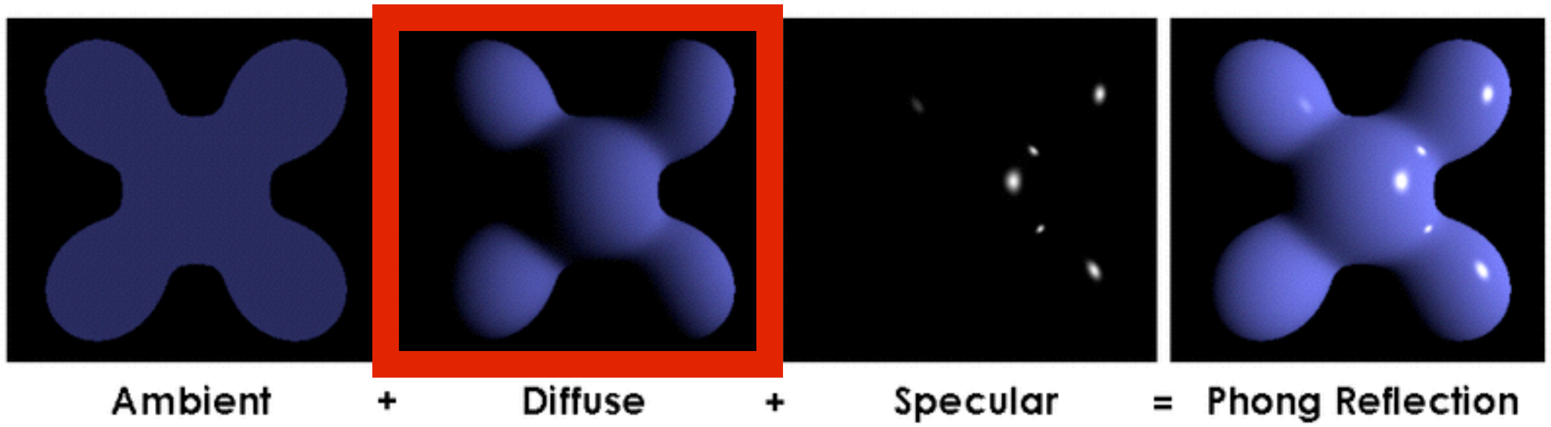
ambient reflection coefficient

Diffuse reflection



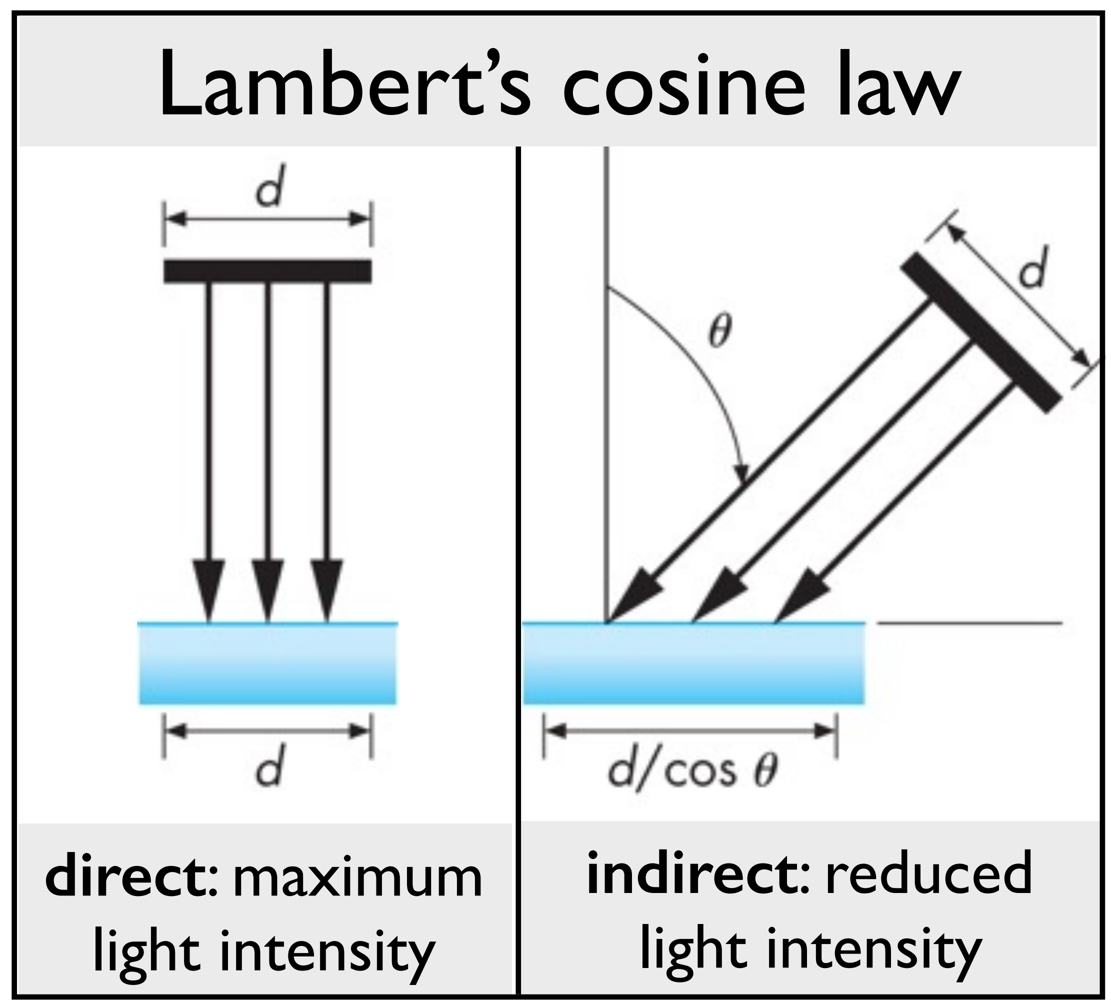


Diffuse reflection

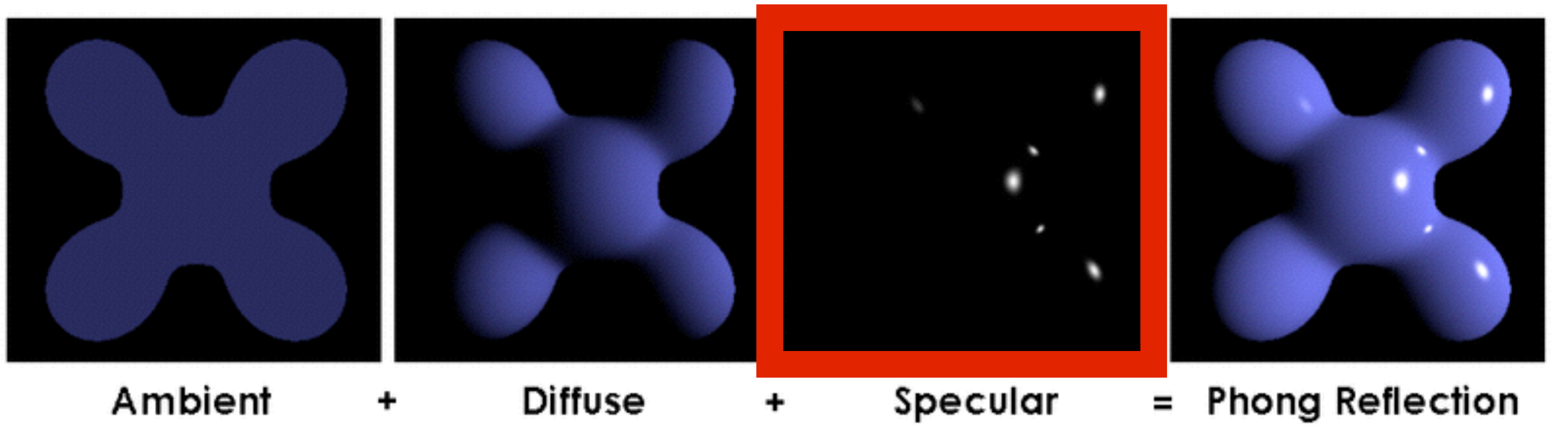


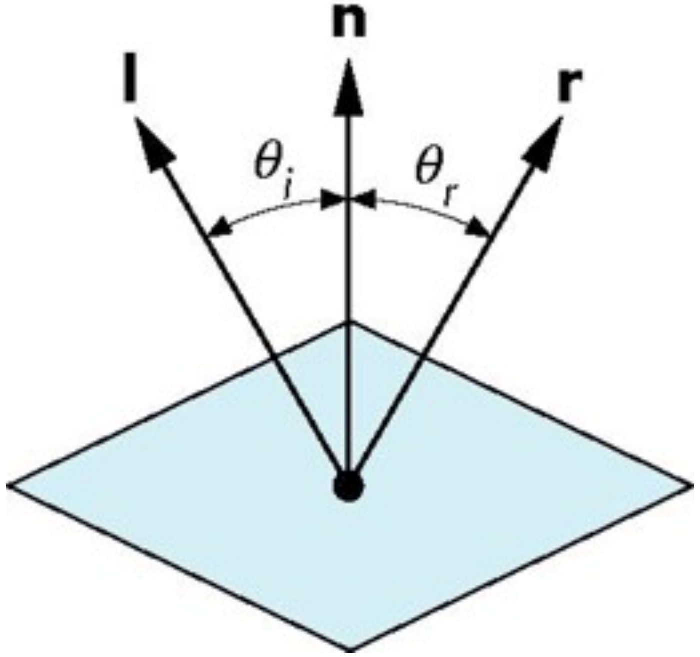
$$I_d = R_d L_d \max(0, \mathbf{l} \cdot \mathbf{n})$$

diffuse reflection coefficient



Specular reflection





Ideal reflector

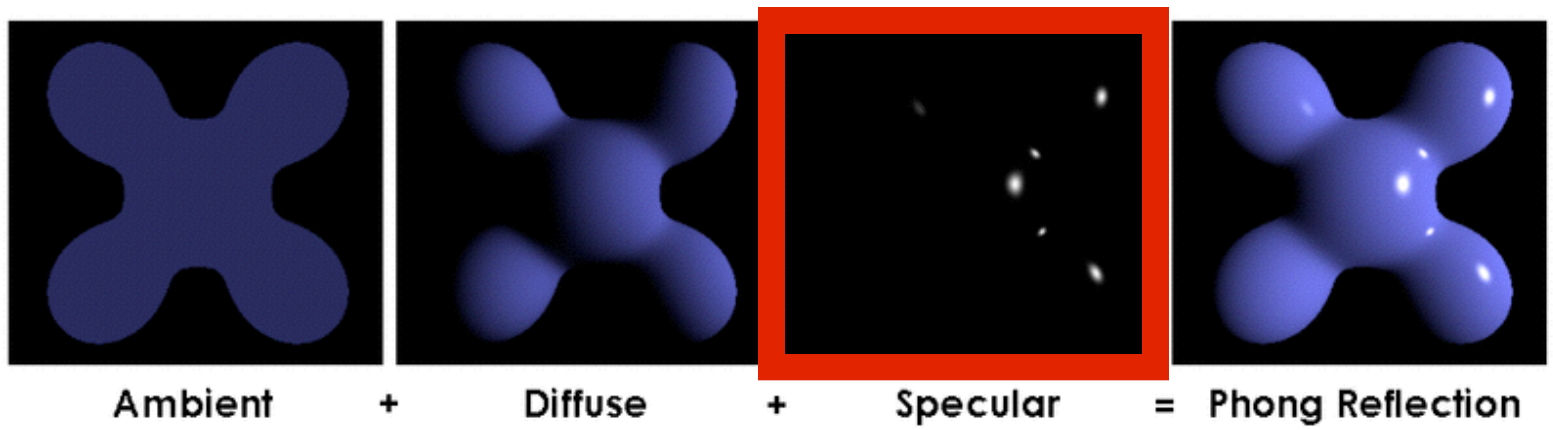
$$\theta_i = \theta_r$$

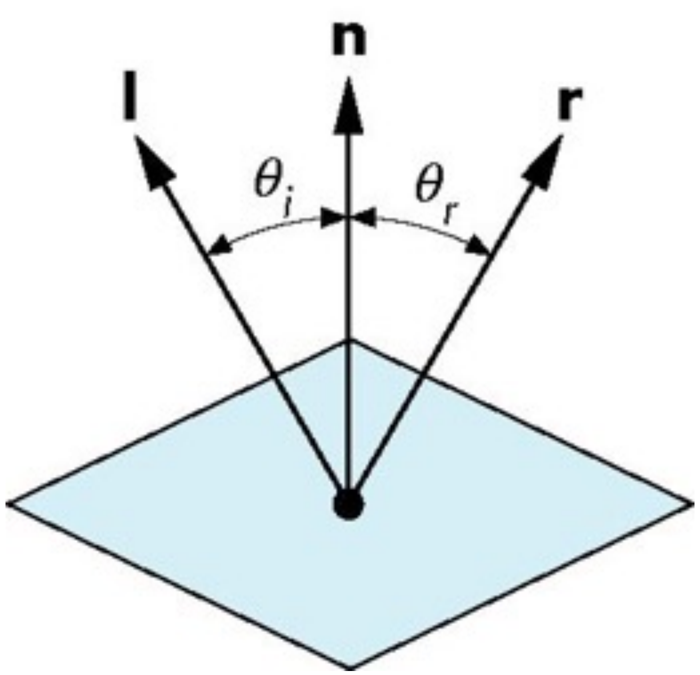
angle of incidence

angle of reflection

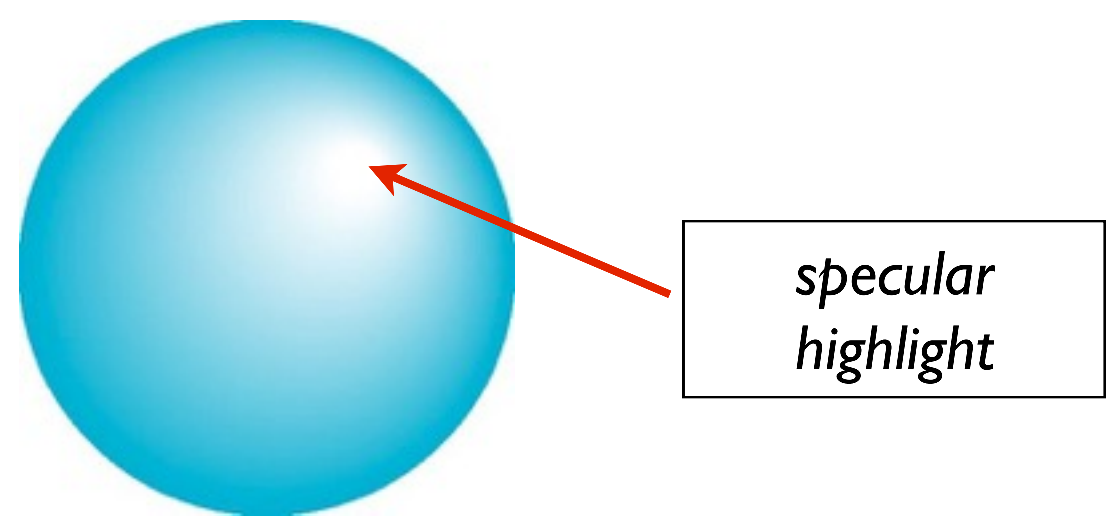
r is the mirror reflection direction

Specular reflection



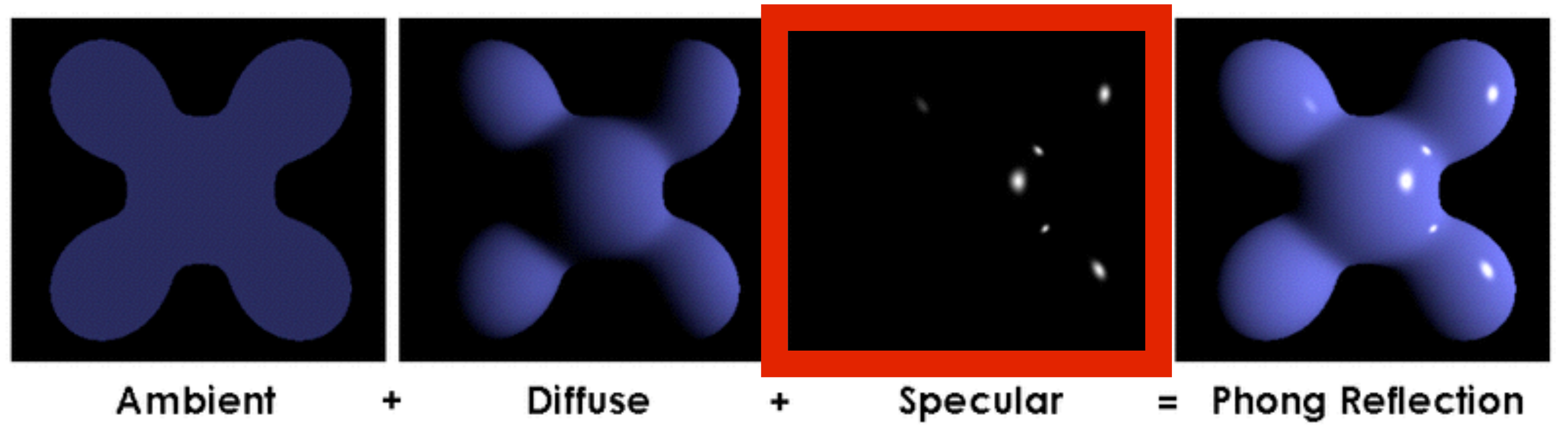


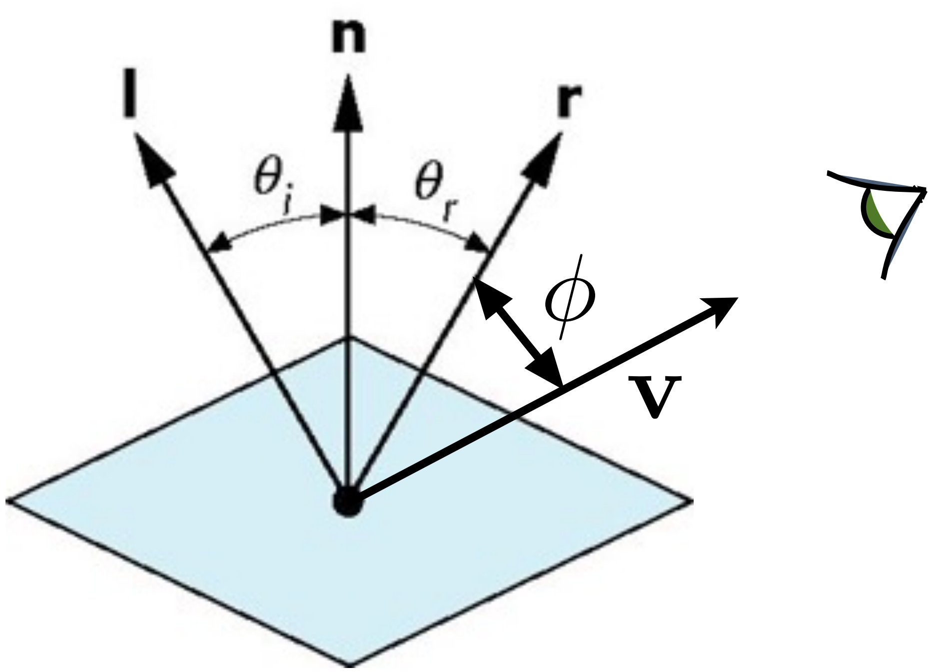
Specular surface



specular reflection is strongest in mirror reflection direction

Specular reflection





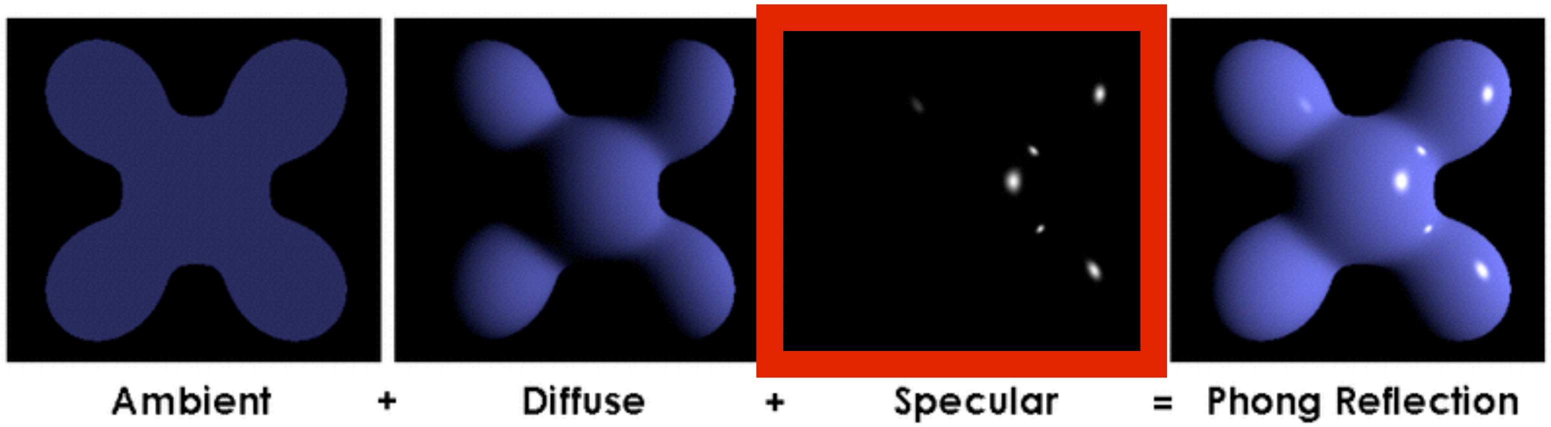
$$I_s = R_s L_s \cos^{\alpha} \phi$$

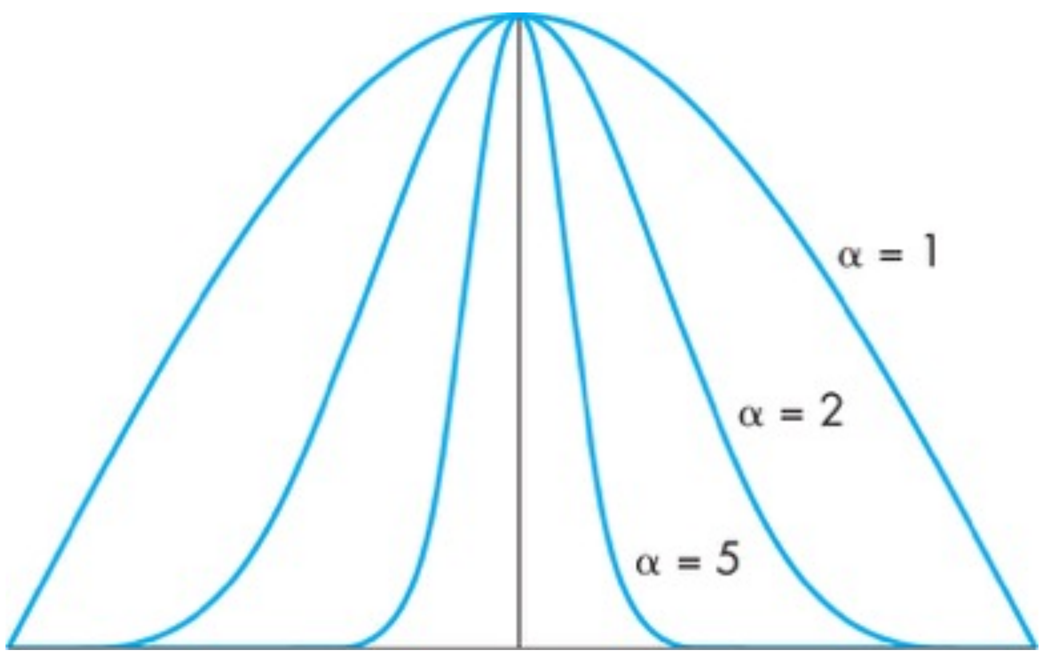
specular reflection coefficient

Phong exponent

specular reflection drops off with increasing angle ϕ

Specular reflection





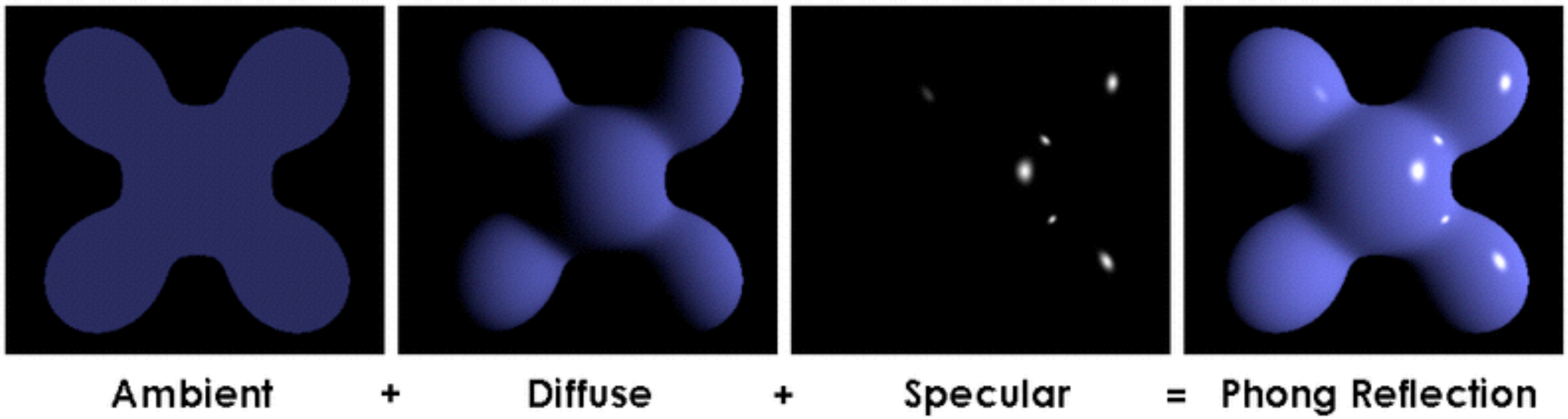
$$I_s = R_s L_s \max(0, \cos\phi)^\alpha$$

Phong exponent

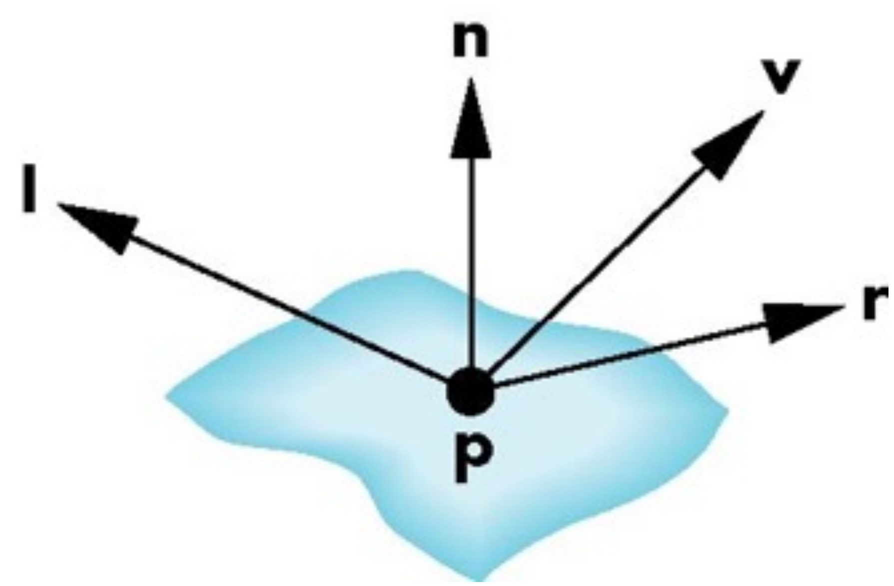
$\alpha = 5..10$ **plastic**

$\alpha = 100..200$ **metal**

Phong Reflection Model



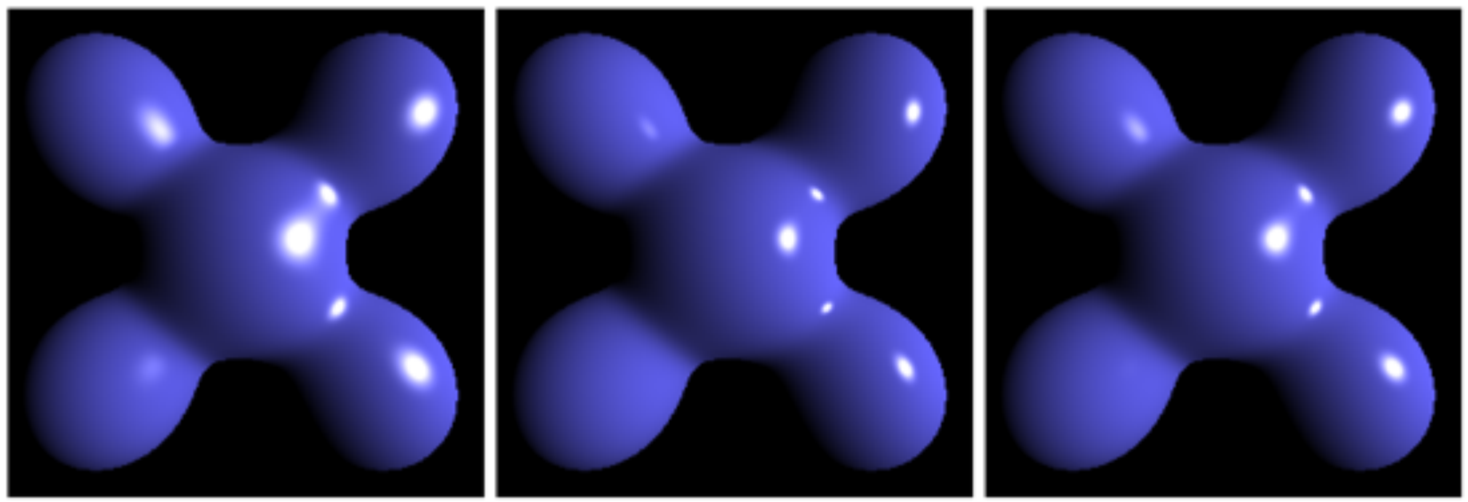
[Brad Smith, Wikimedia Commons]



$$
\begin{aligned}
I &= I_a + I_d + I_s \\
&= R_a L_a + R_d L_d \max(0, \mathbf{l} \cdot \mathbf{n}) + R_s L_s \max(0, \mathbf{v} \cdot \mathbf{r})^\alpha
\end{aligned}
$$

Ambient Diffuse Specular

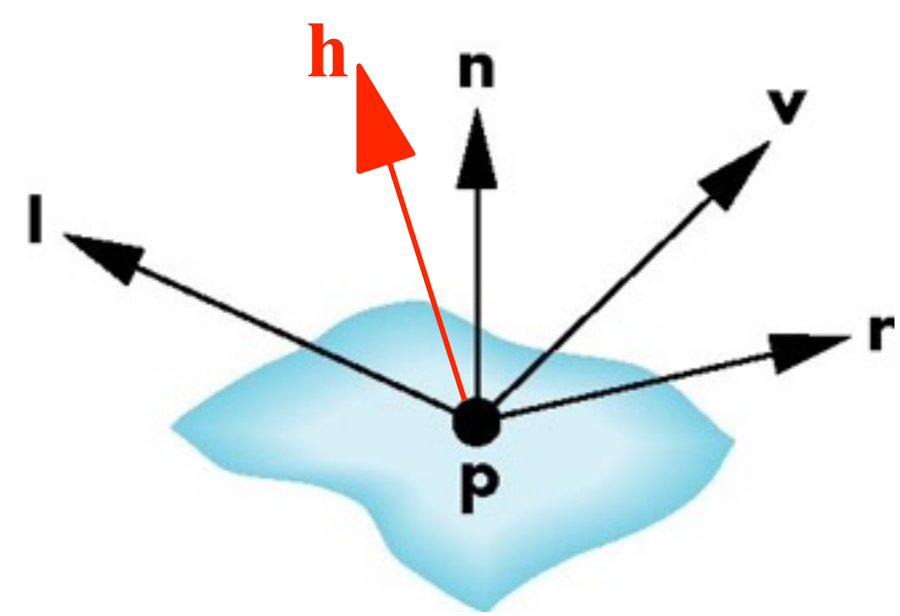
Alternative: Blinn-Phong Model



Blinn–Phong | Phong | Blinn–Phong (Lower Exponent)

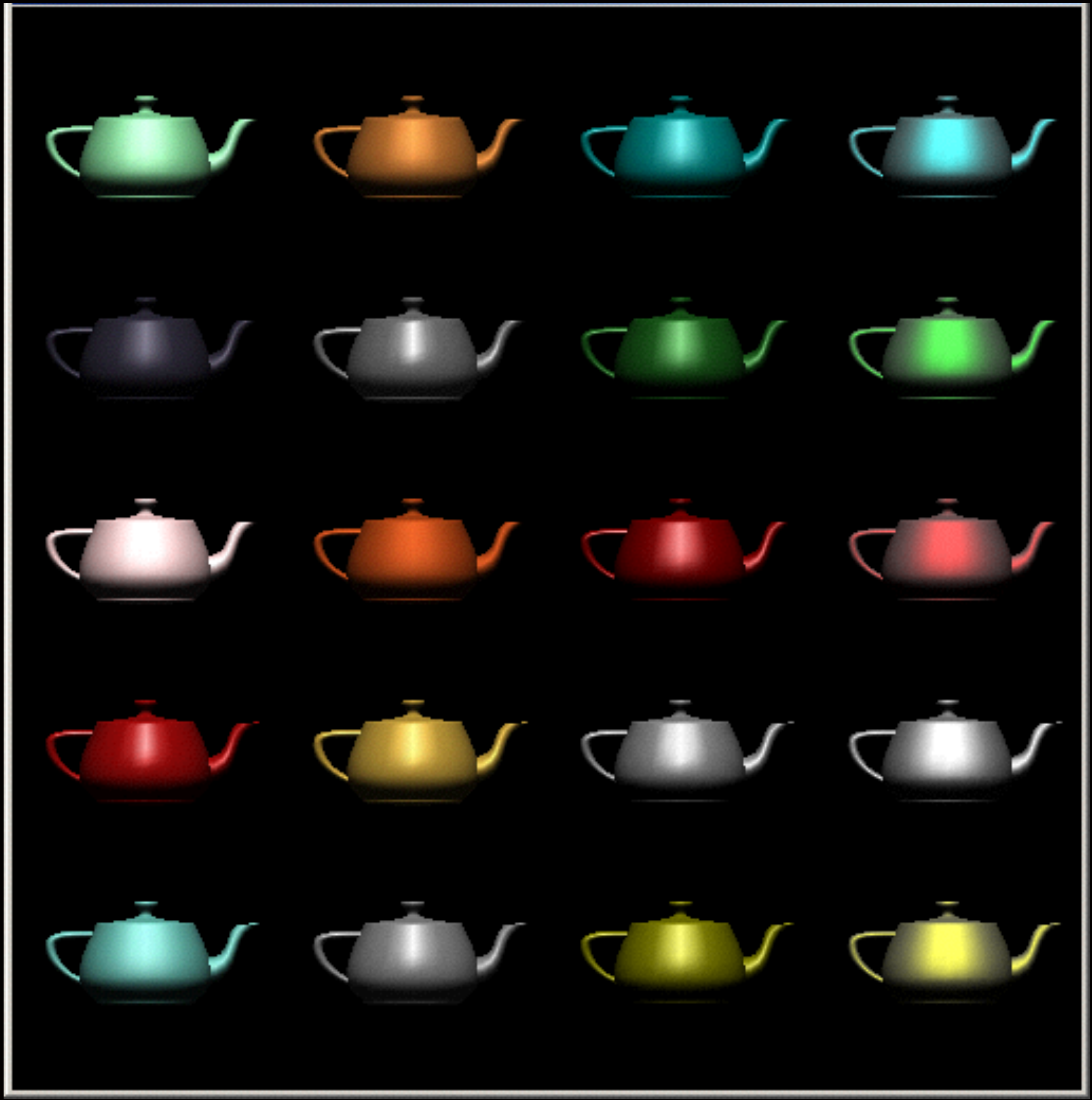
[Brad Smith, Wikimedia Commons]

halfway vector $$\mathbf{h} = \frac{\mathbf{l} + \mathbf{v}}{|\mathbf{l} + \mathbf{v}|}$$



$$I = I_a + I_d + I_s$$
$$= R_a L_a + R_d L_d \max(0, \mathbf{l} \cdot \mathbf{n}) + R_s L_s \max(0, \mathbf{h} \cdot \mathbf{n})^\alpha$$

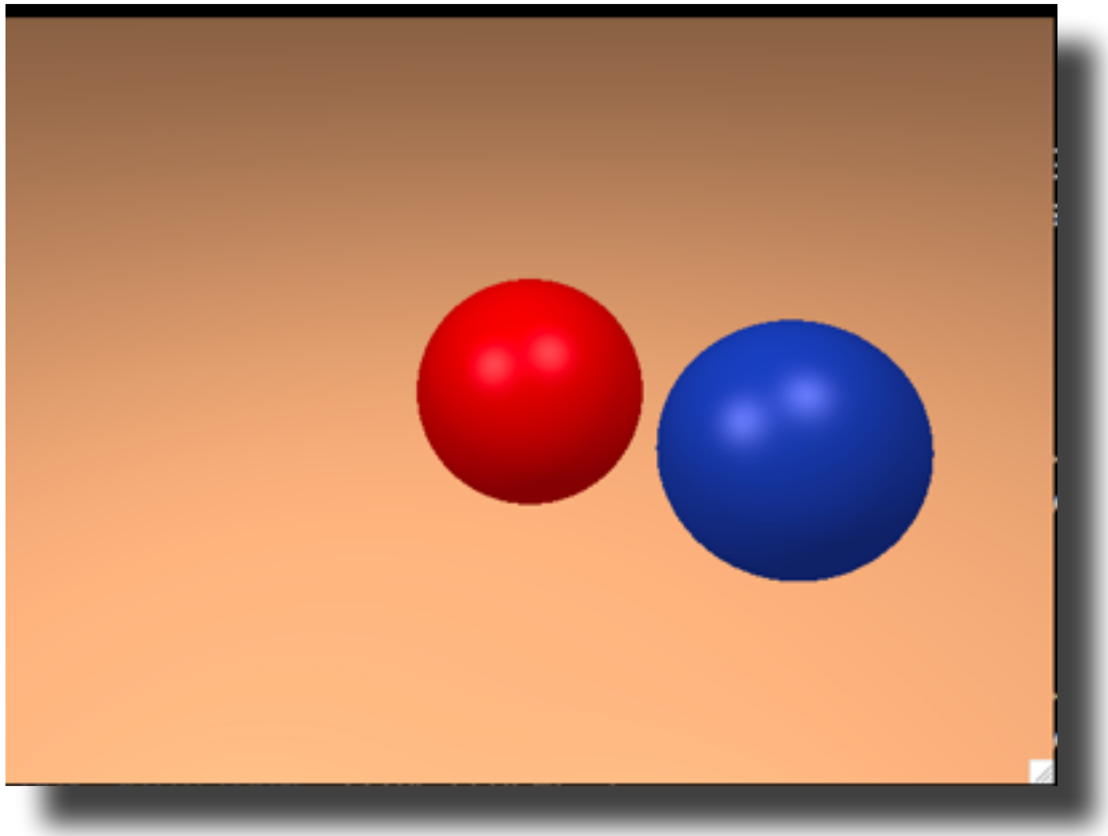
Ambient Diffuse Specular

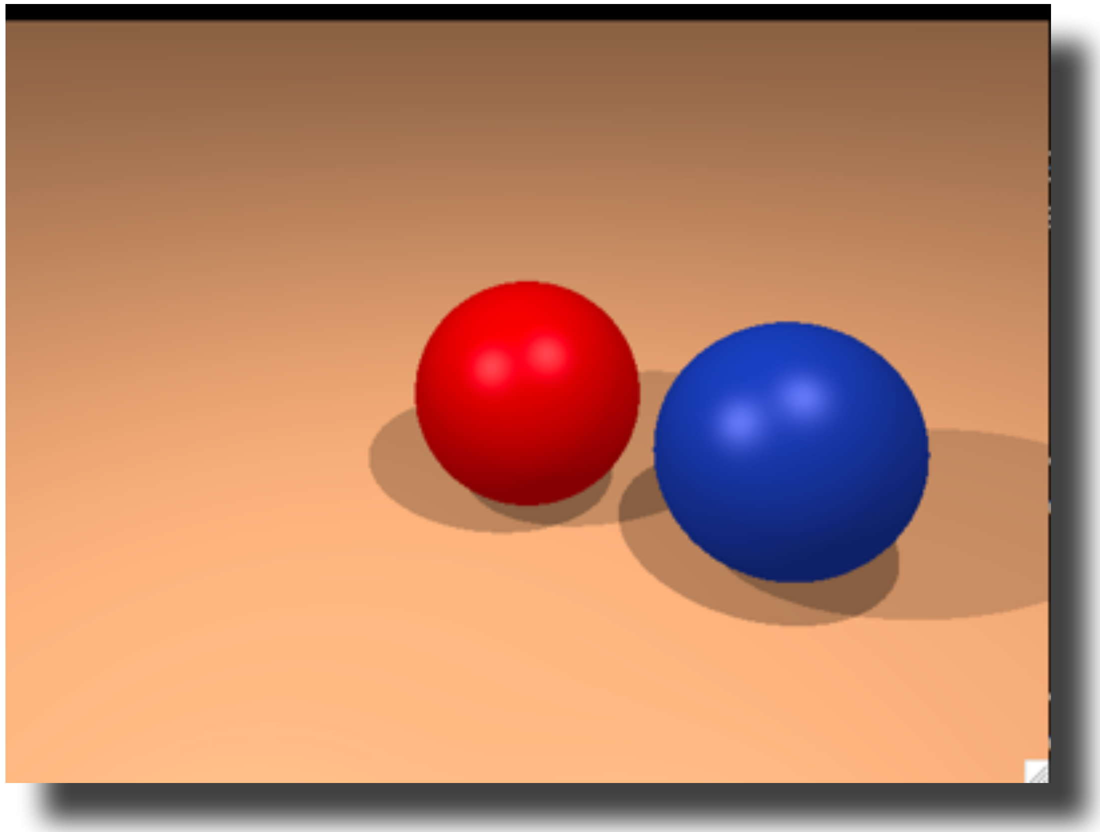


α

10: eggshell
100: shiny
1000: glossy
10000: mirror-like

Shadows





Shadows

```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
```

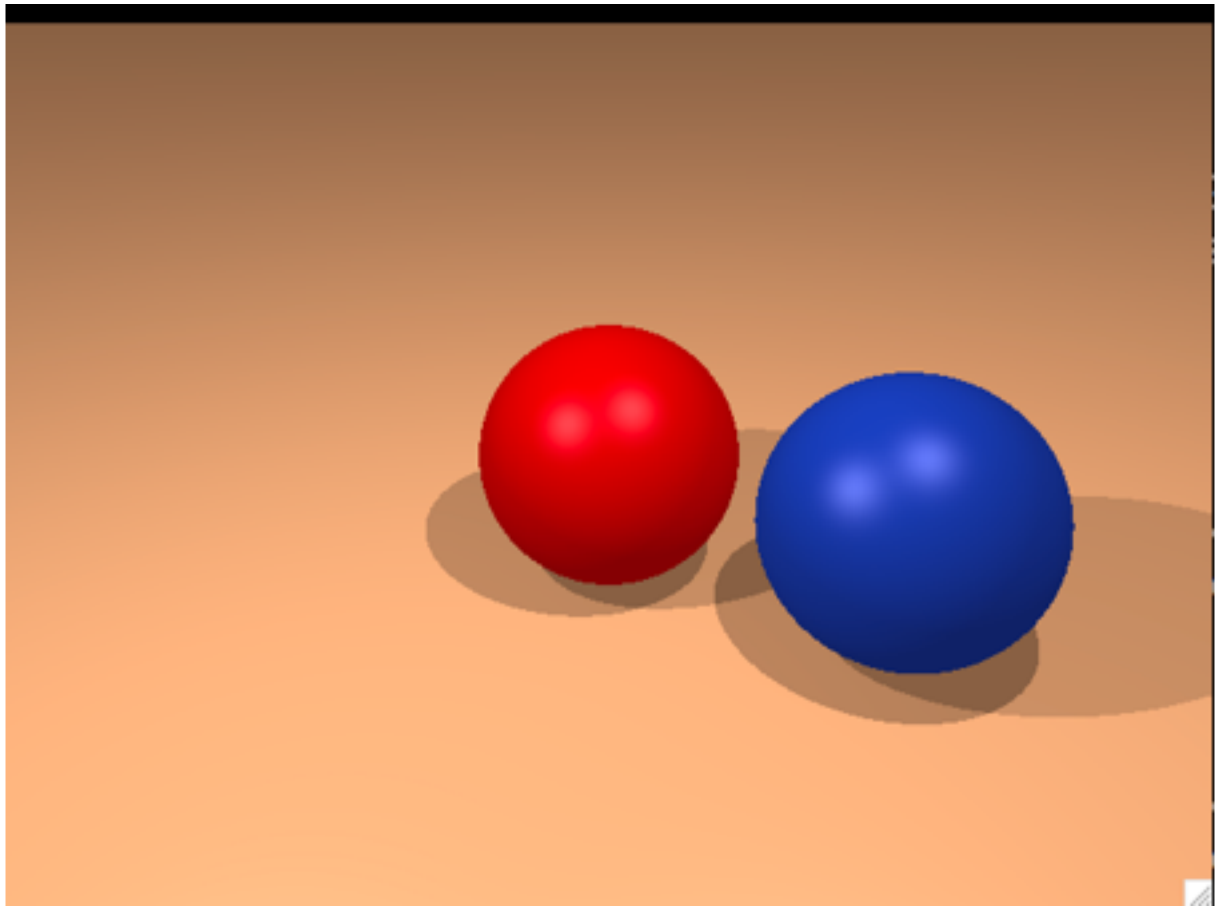
Shadows

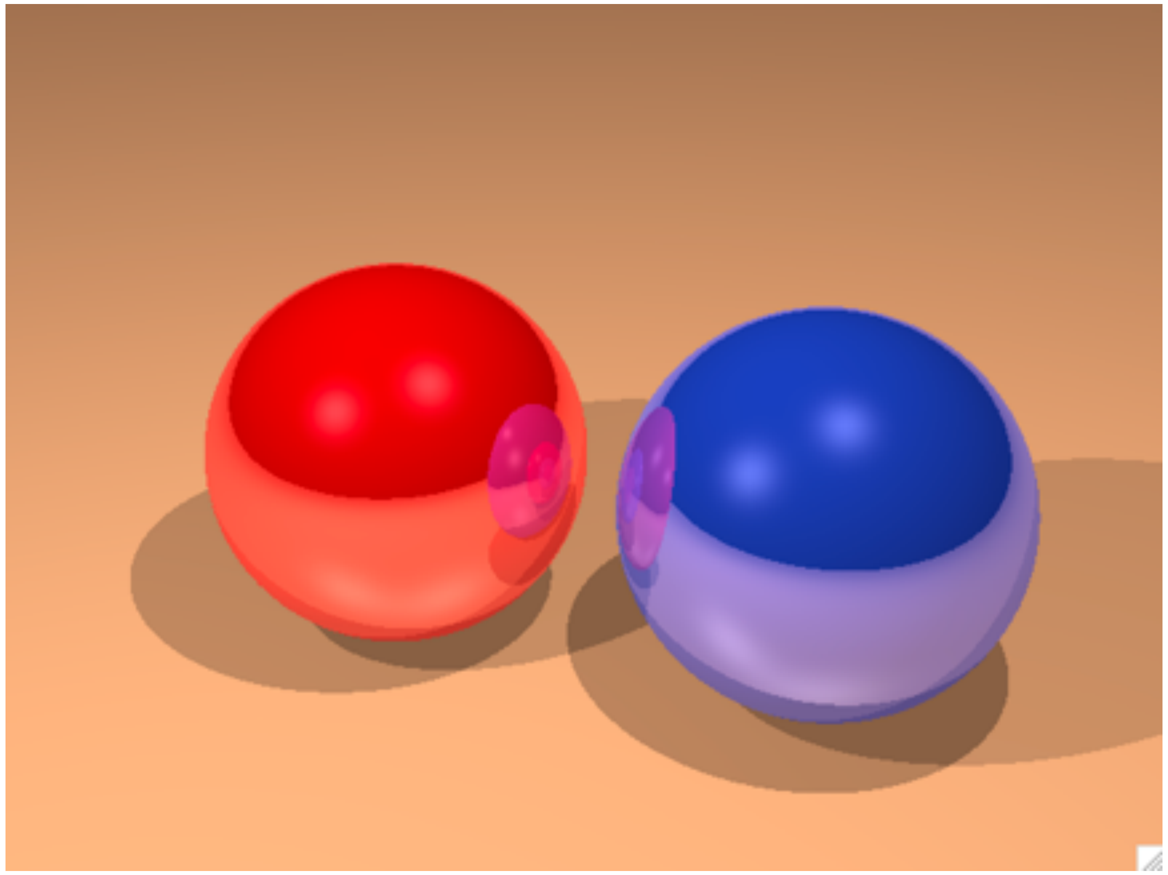
```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
```

Shadows

```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        // e.g., phong shading
        for each light
          add light's ambient component
          compute shadow ray
          if ( ! shadow ray hits an object )
            add light's diffuse and specular components
    else
        set pixel color to the background color
```

Reflections





Reflections

```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
```

Reflections

```
for each pixel do
    compute viewing ray
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        evaluate shading model and set pixel to that color
    else
        set pixel color to the background color
```

Reflections

```
for each pixel do
    compute viewing ray
    pixel color = cast_ray(viewing ray)

cast_ray:
    if ( ray hits an object with t in [0, inf] ) then
        compute n
        return color = shade_surface
    else
        return color = to the background color

shade_surface:
    color = ...
    compute reflected ray
    return color = color + k * cast_ray(reflected ray)
```