SPECULAR & PHONG SHADING

adding in shine

BASED ON MIT 6.837

slides adapted & project started code translated to Swift by Dion Larson
adapted course materials available for free here
original course materials available for free here

Specular shading

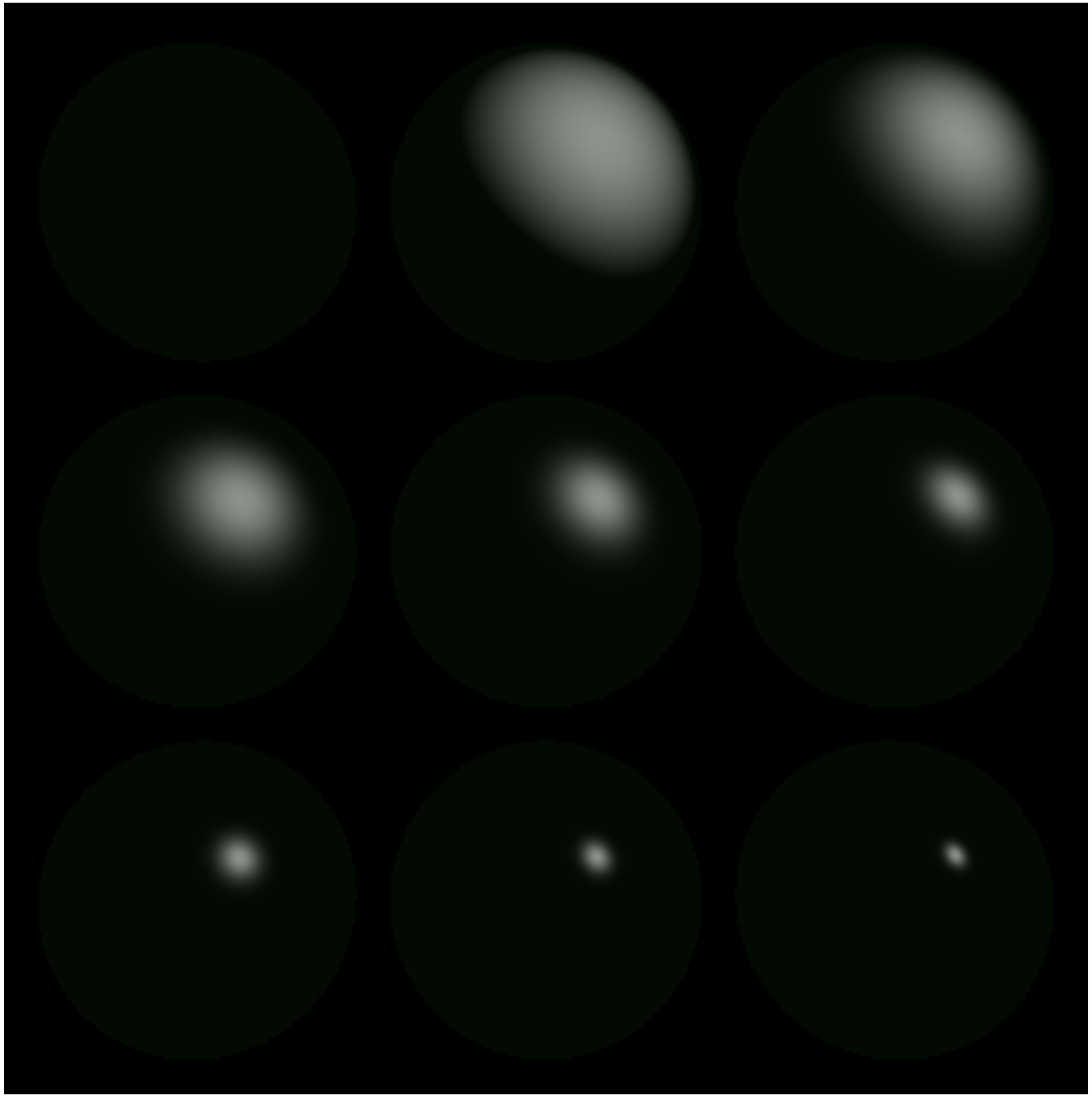
Phong shading model

Next week

Recap of diffuse shading

Mob programming (diffuse shading)

SPECULAR SHADING

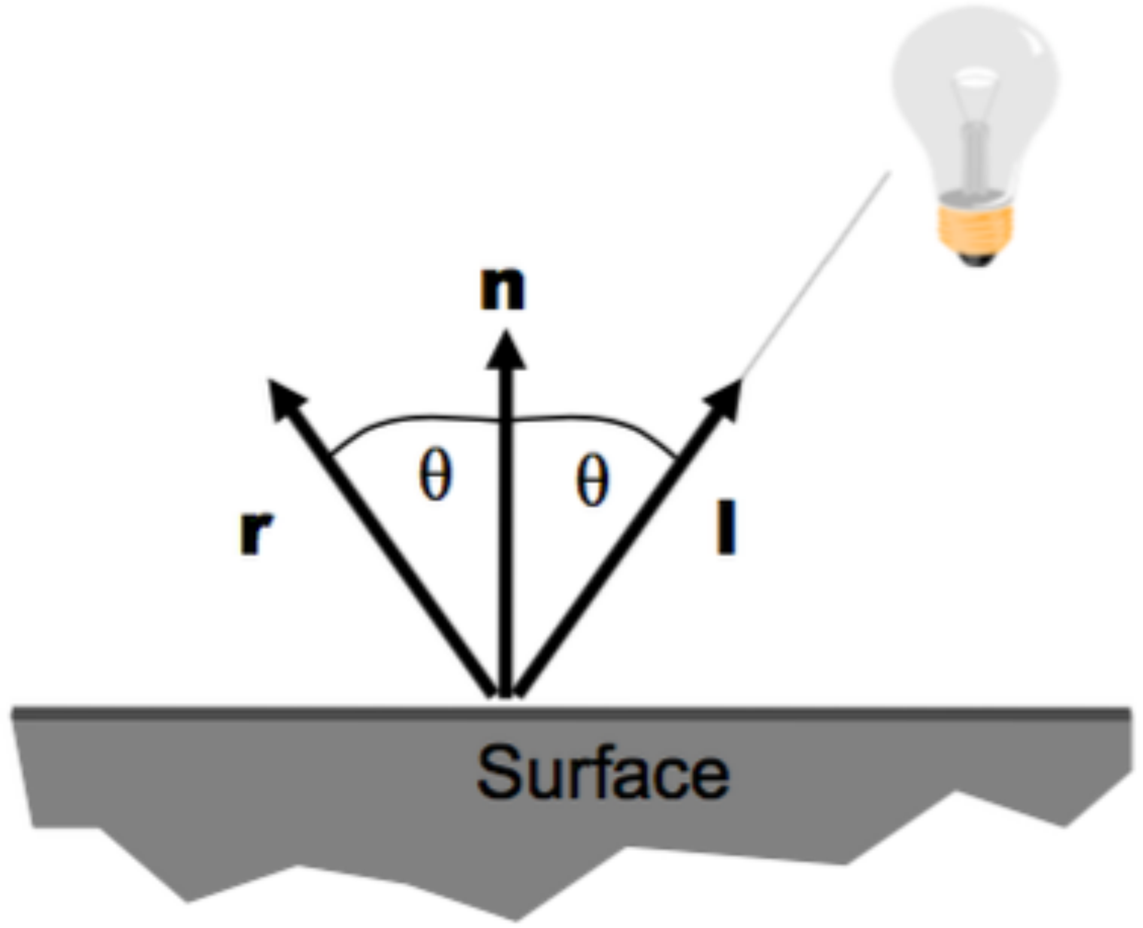


IDEAL SPECULAR REFLECTANCE

Reflection is only at mirror angle

View dependent

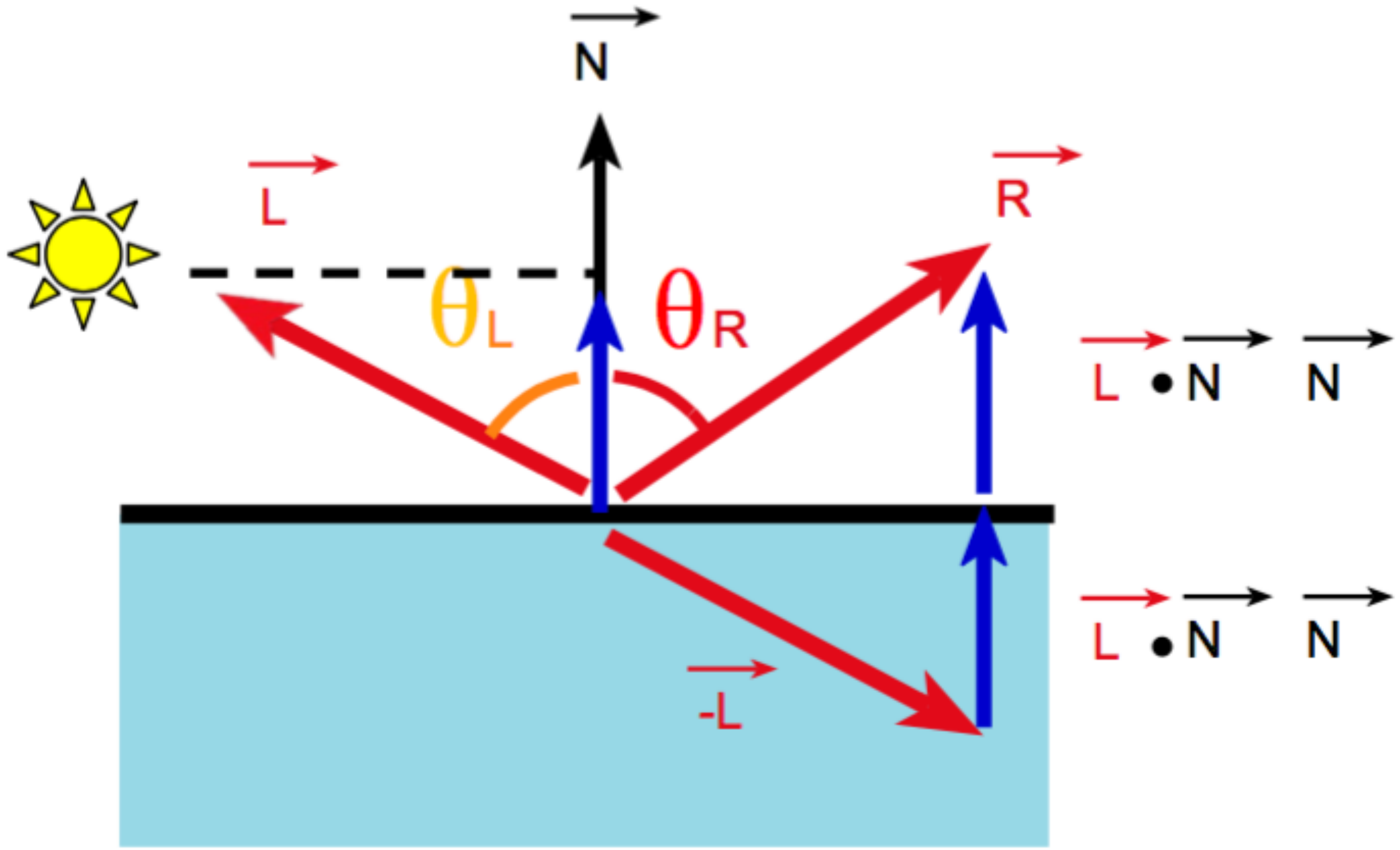
- Microscopic surface elements are usually oriented in the same direction as the surface (mirrors, highly polished metals)



HOW TO GET MIRROR DIRECTION

Reflection angle = light angle

$$R = -L + 2(L \cdot N)N$$



IDEAL REFLECTORS

Light **only** reflects to the mirror direction

Not useful for point lights, only for reflections of other surfaces… Why?

NON-IDEAL REFLECTORS

Really glossy materials are not usually ideal mirror reflectors -- highlight is blurry

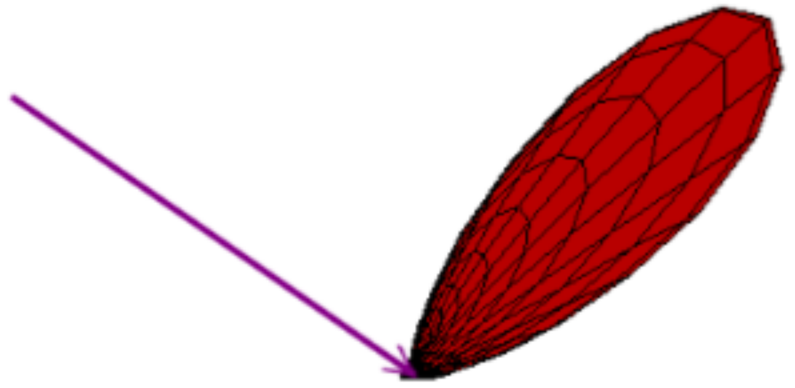
Combination of diffuse and specular

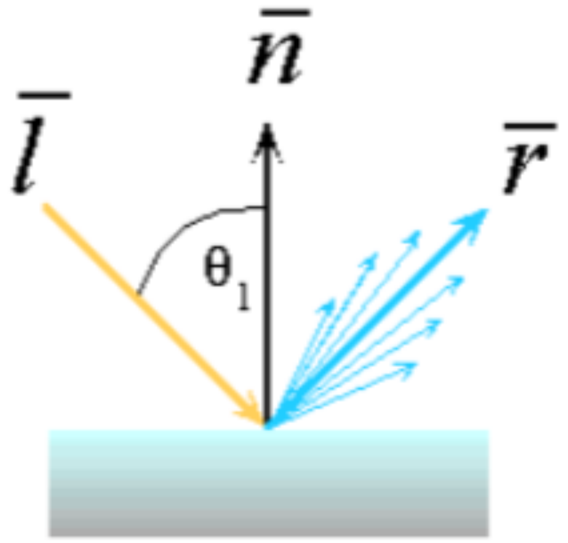
GLOSSY MATERIALS

We expect most of the reflected light to travel in the direction of the ideal mirror ray

Small variations in surface at microscopic level leads to some light being reflected slightly offset from the ideal reflected ray

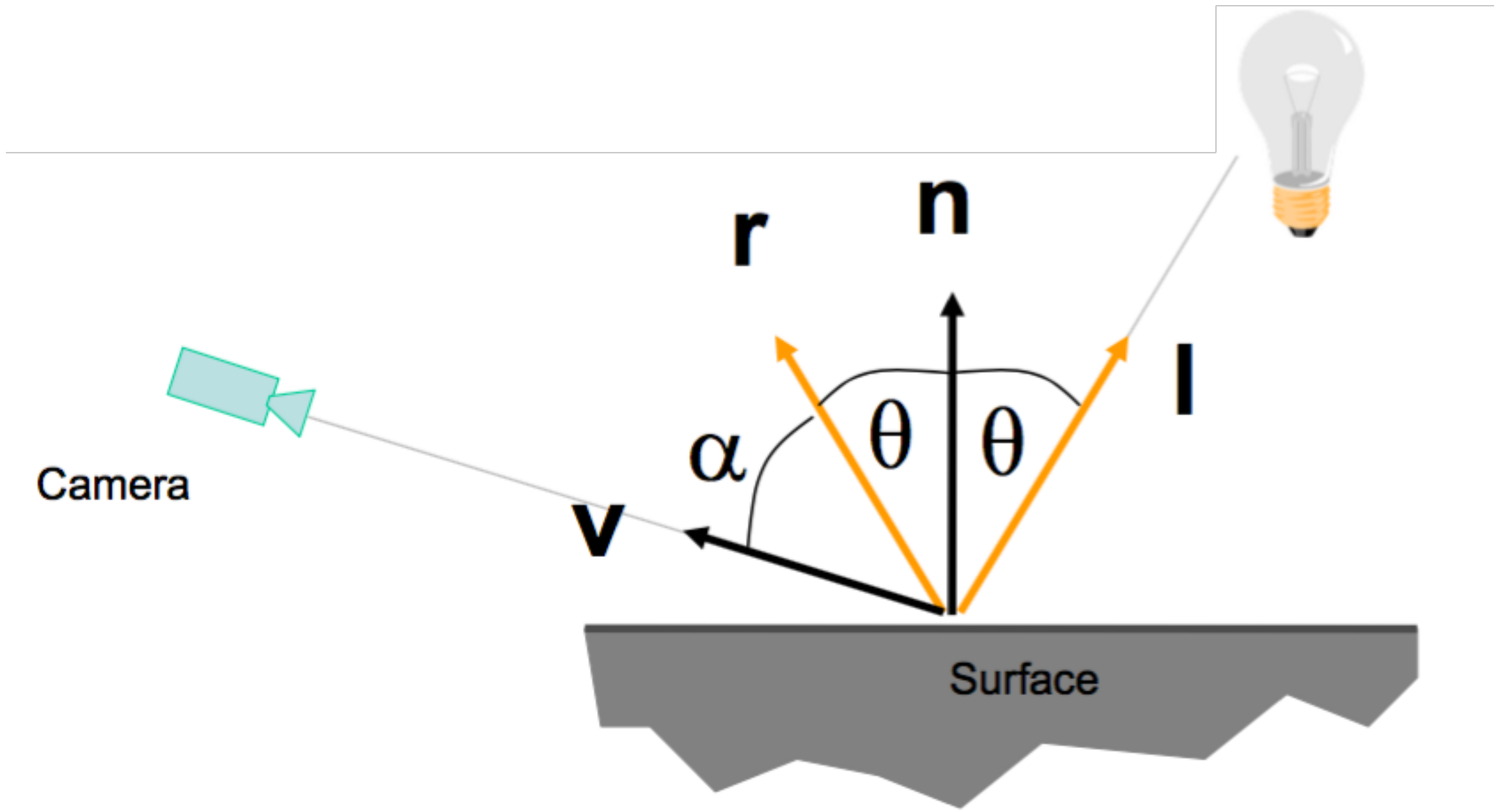
The larger θ is, the less reflected light we expect to see





HOW MUCH LIGHT IS REFLECTED?

Depends on the angle α between the ideal reflection direction r and the viewer direction v

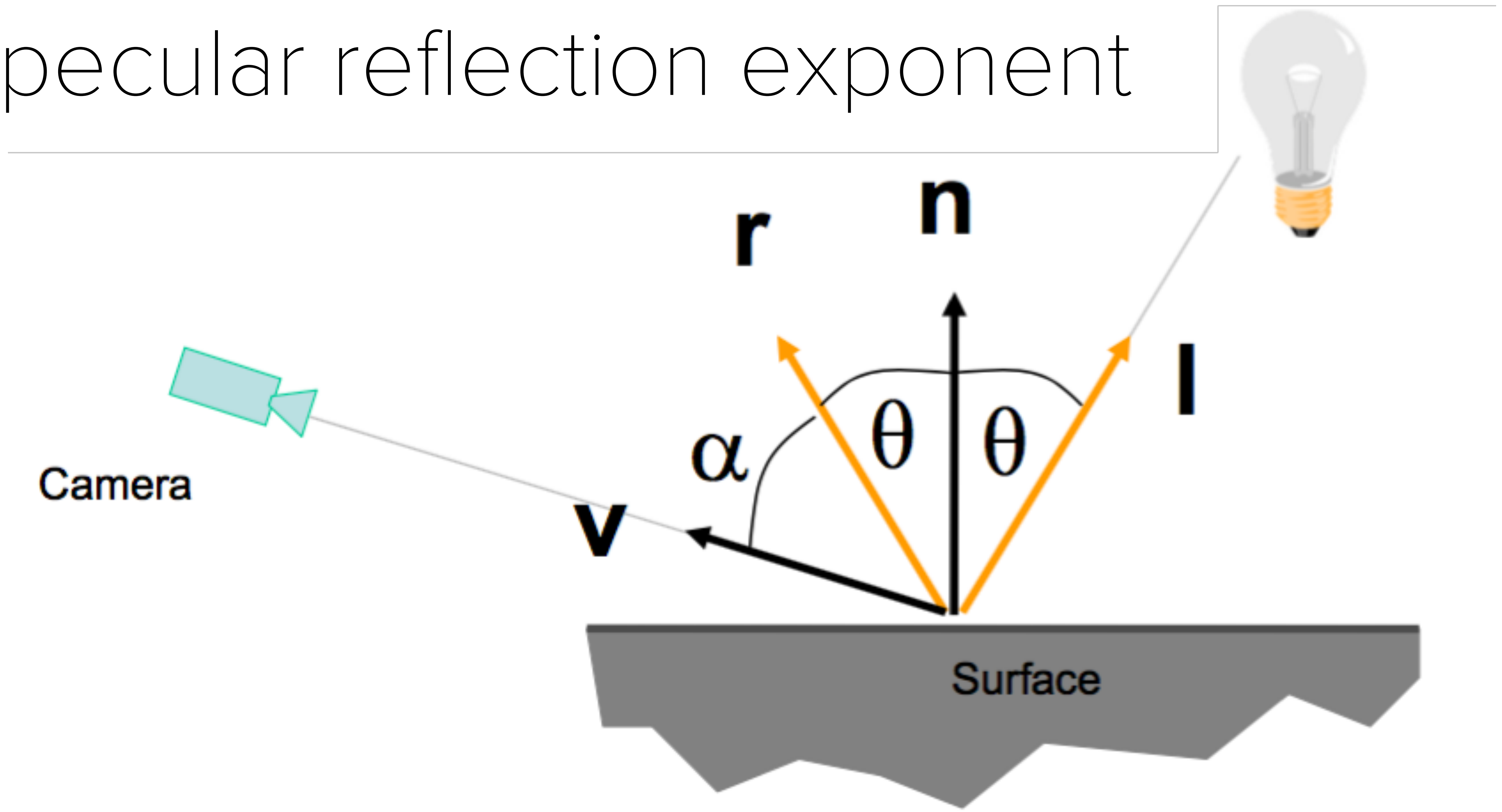


SPECULAR SHADING

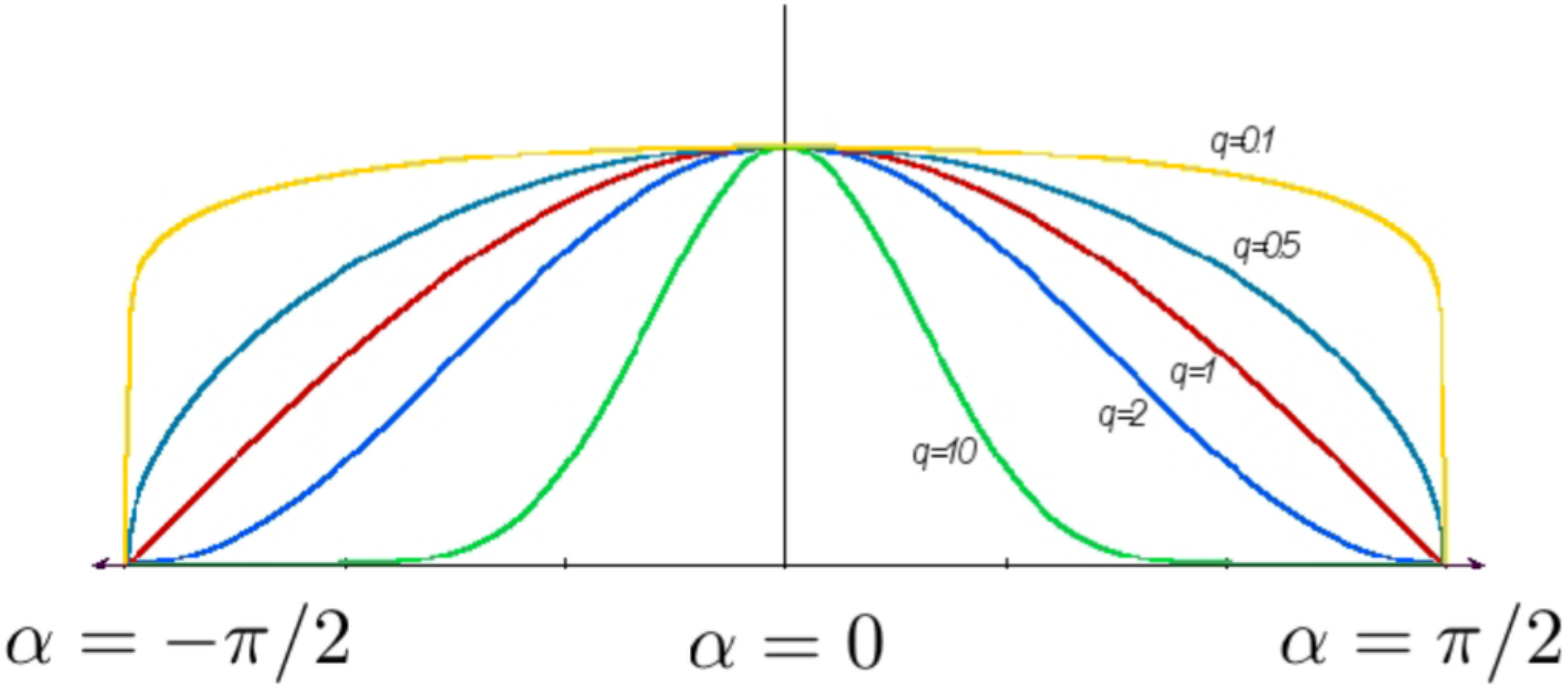
$$L_o = k_s (\cos\alpha)^q L_i = k_s (v \cdot r)^q L_i$$

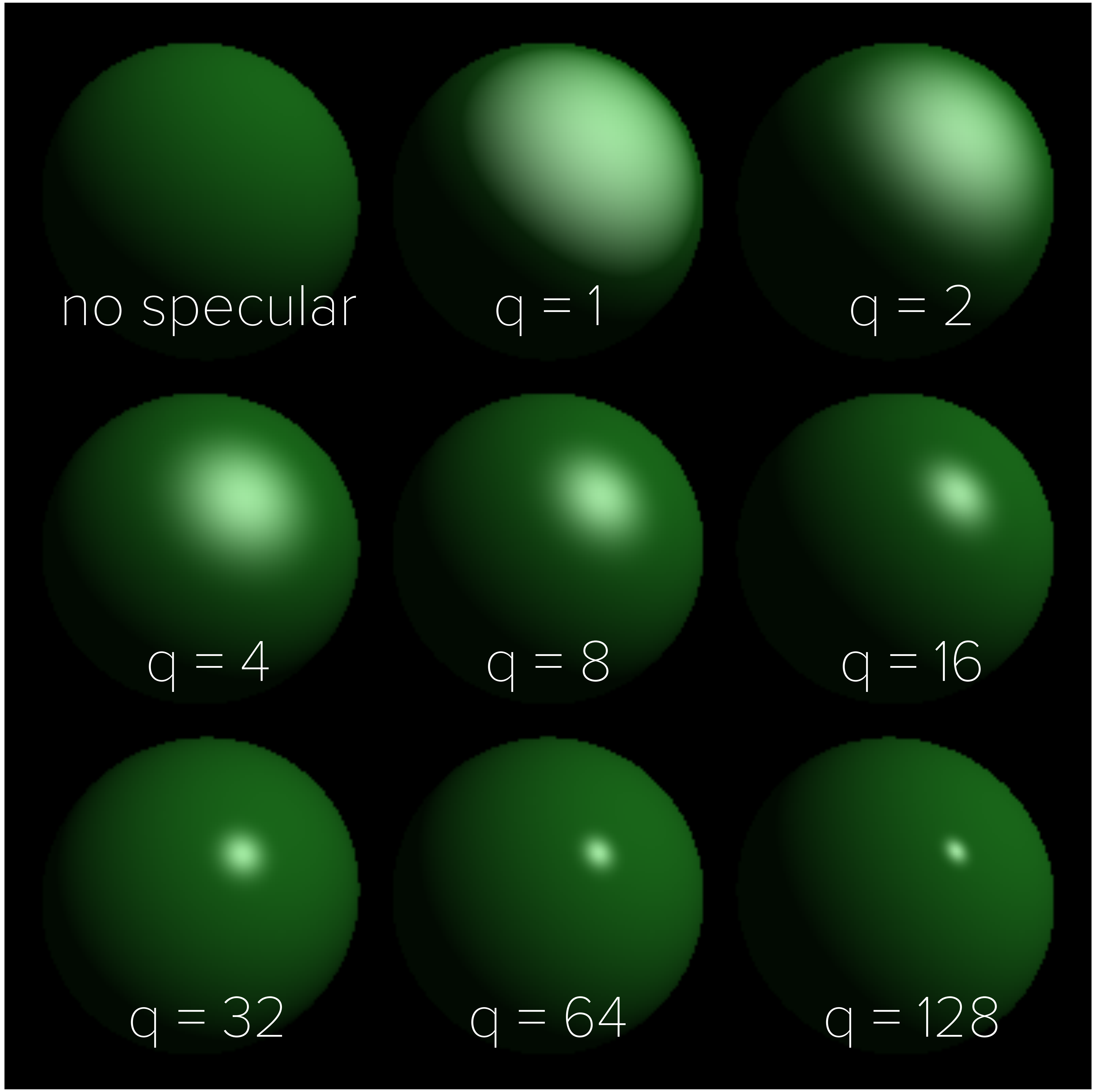
k_s specular reflection coefficient

q specular reflection exponent



EFFECT OF SPECULAR REFLECTION EXPONENT




no specular
q = 1
q = 2
q = 4
q = 8
q = 16
q = 32
q = 64
q = 128

Specular shading

Phong shading model

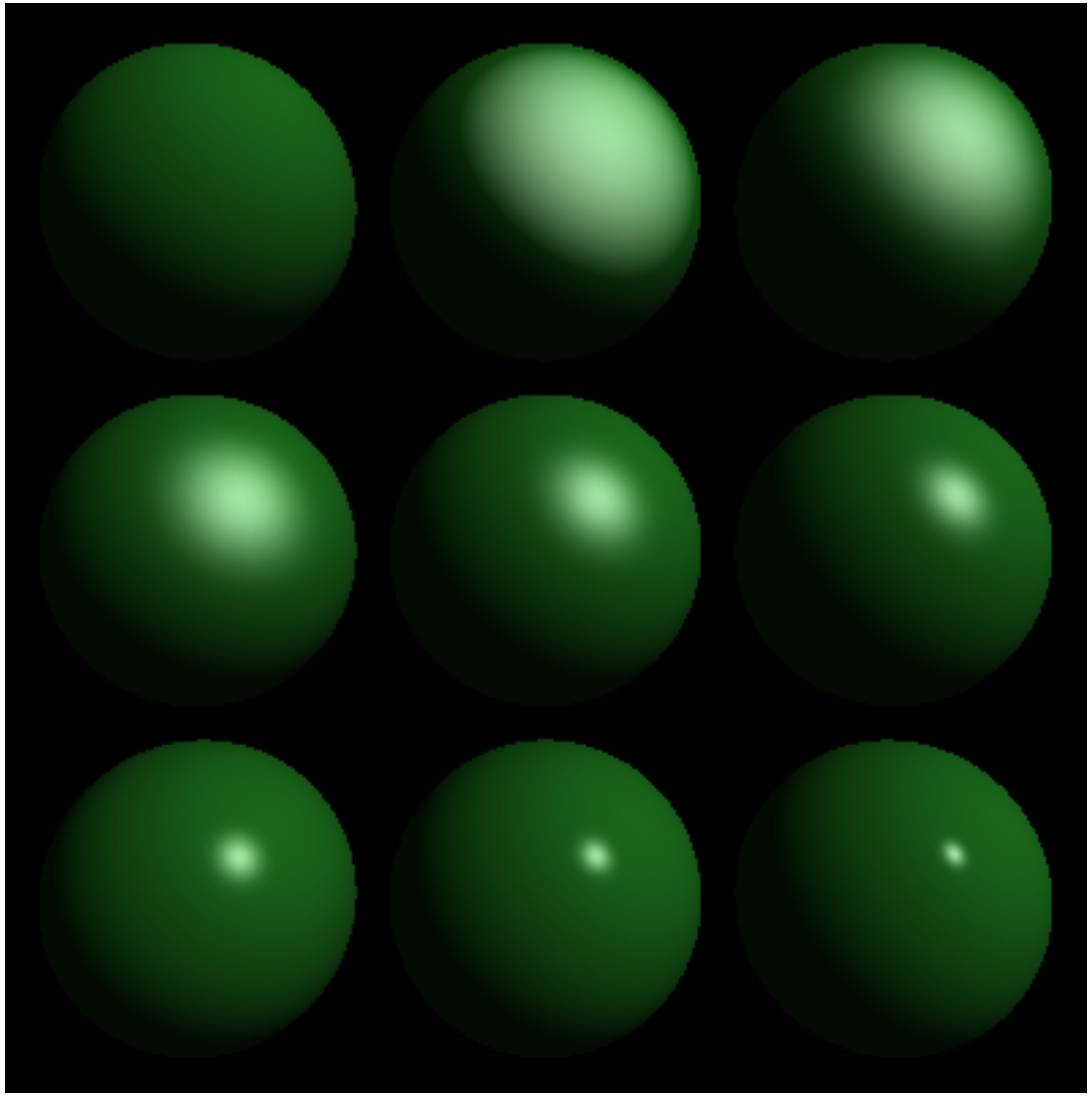
Next week

Recap of diffuse shading

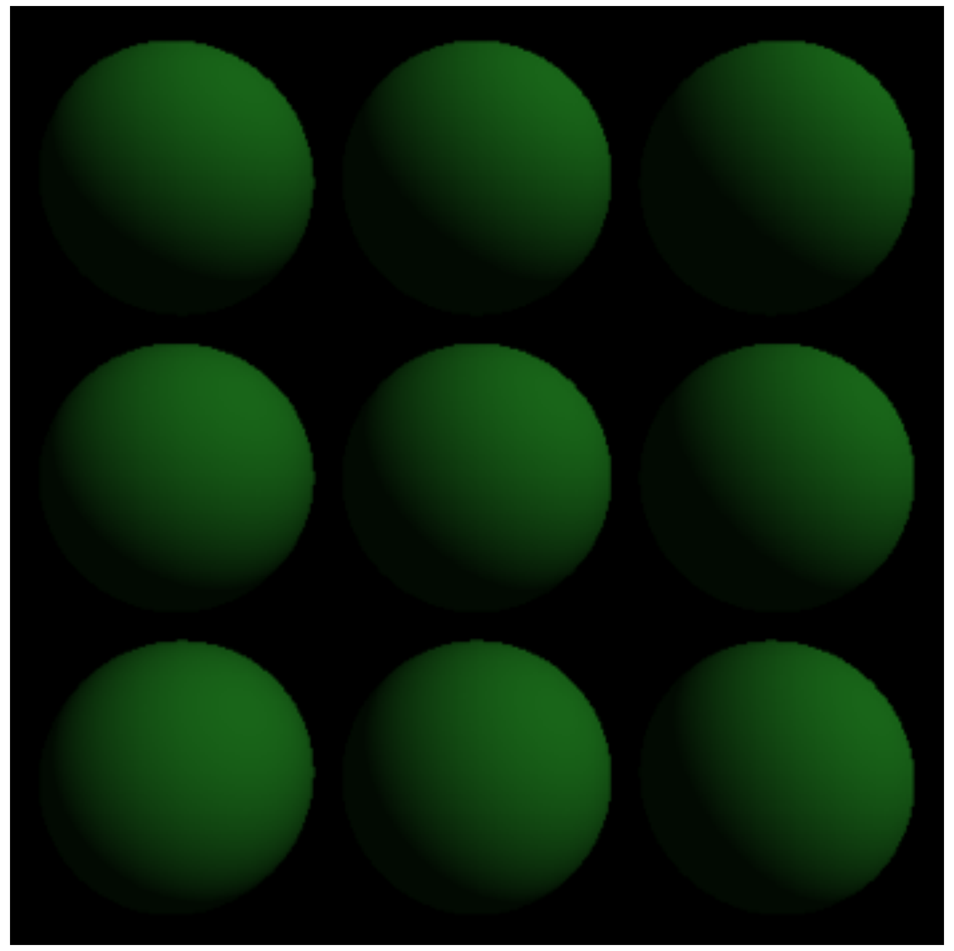
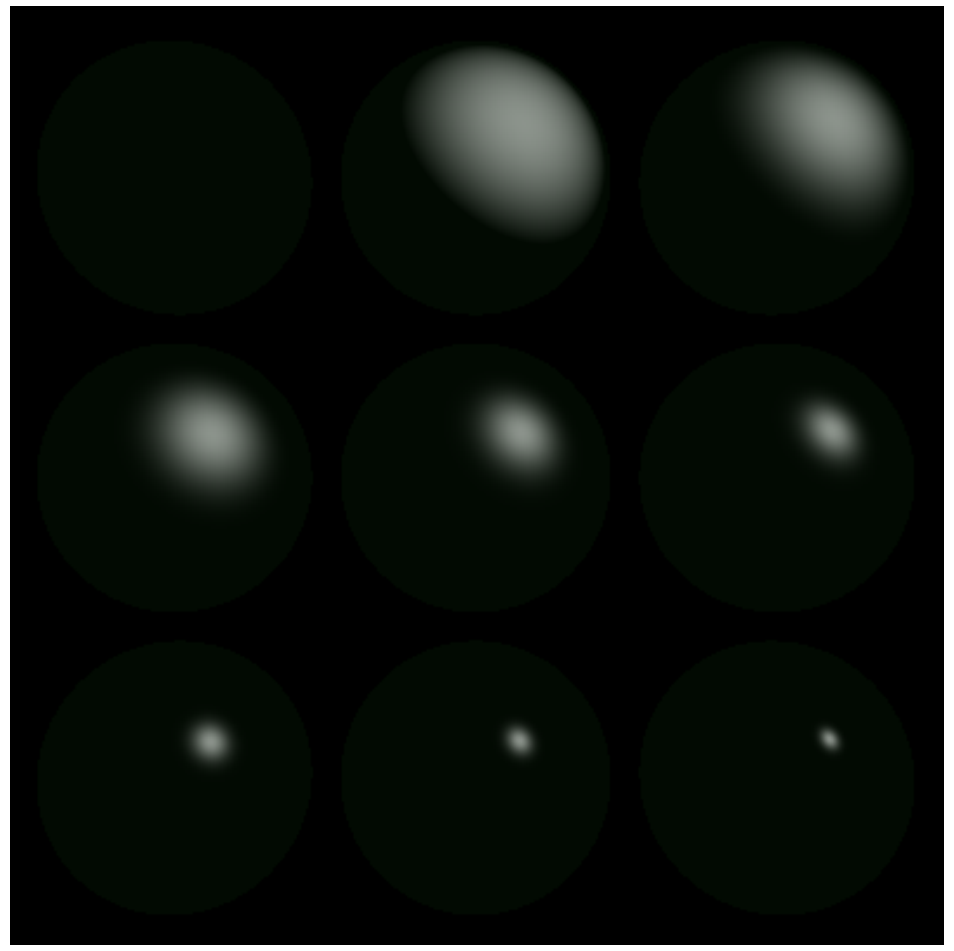
Mob programming (diffuse shading)

PHONG SHADING MODEL

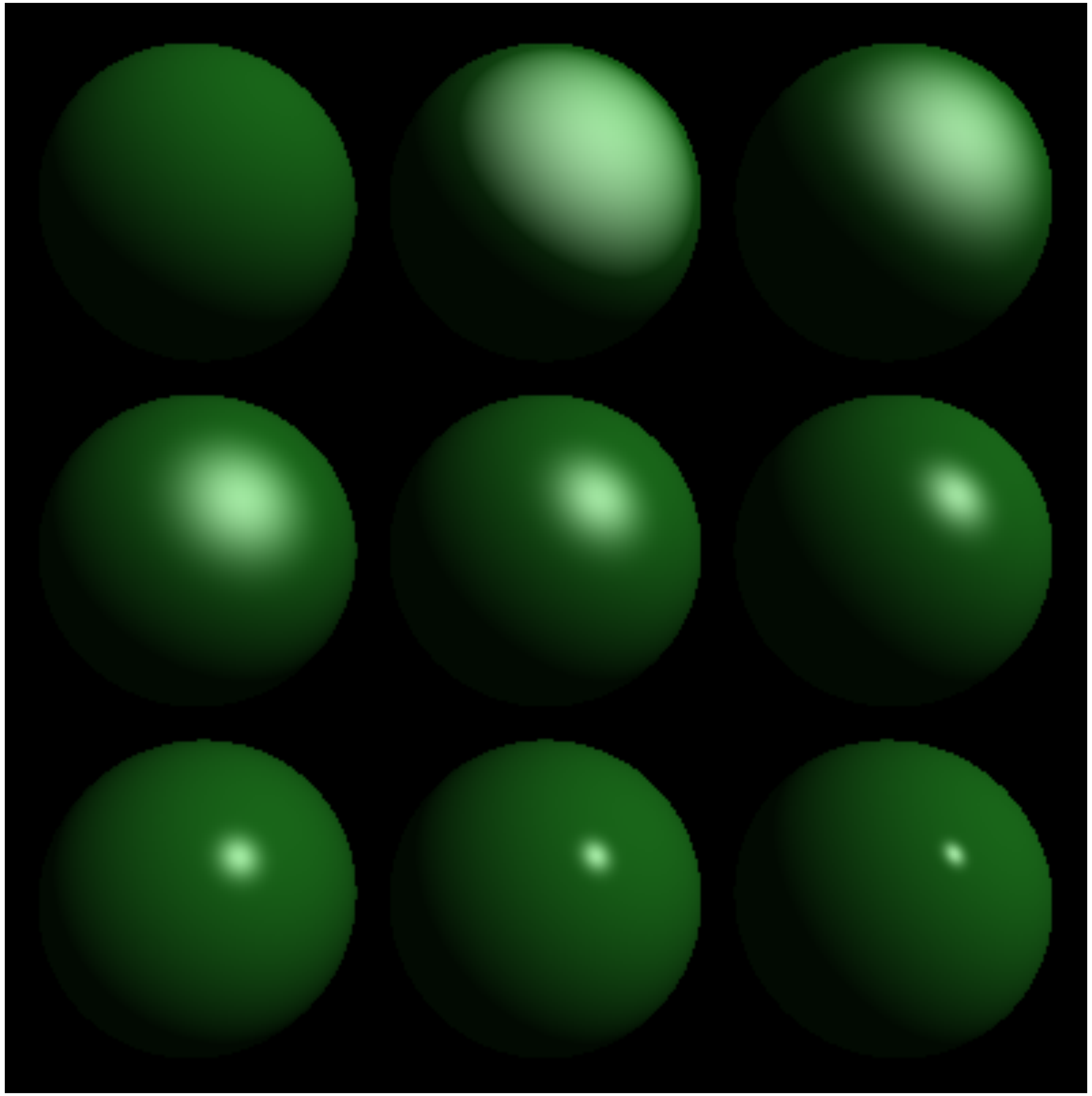
ambient + diffuse + specular



PHONG SHADING MODEL

diffuse + specular = Phong shading



COMPLETE PHONG MODEL

$$L_o = k_a + \sum_{i=1}^{n} L_i [k_d (n \cdot l) + k_s (v \cdot r)^q]$$

COMPLETE PHONG MODEL

$$L_o = k_a + \sum_{i=1}^{n} L_i [k_d (n \cdot l) + k_s (v \cdot r)^q]$$

Not physically based

- Does not conserve energy, may reflect more energy than goes in
- Specular portion does not completely conform to BRDF
- Ambient illumination is a total hack

Specular shading

Phong shading model

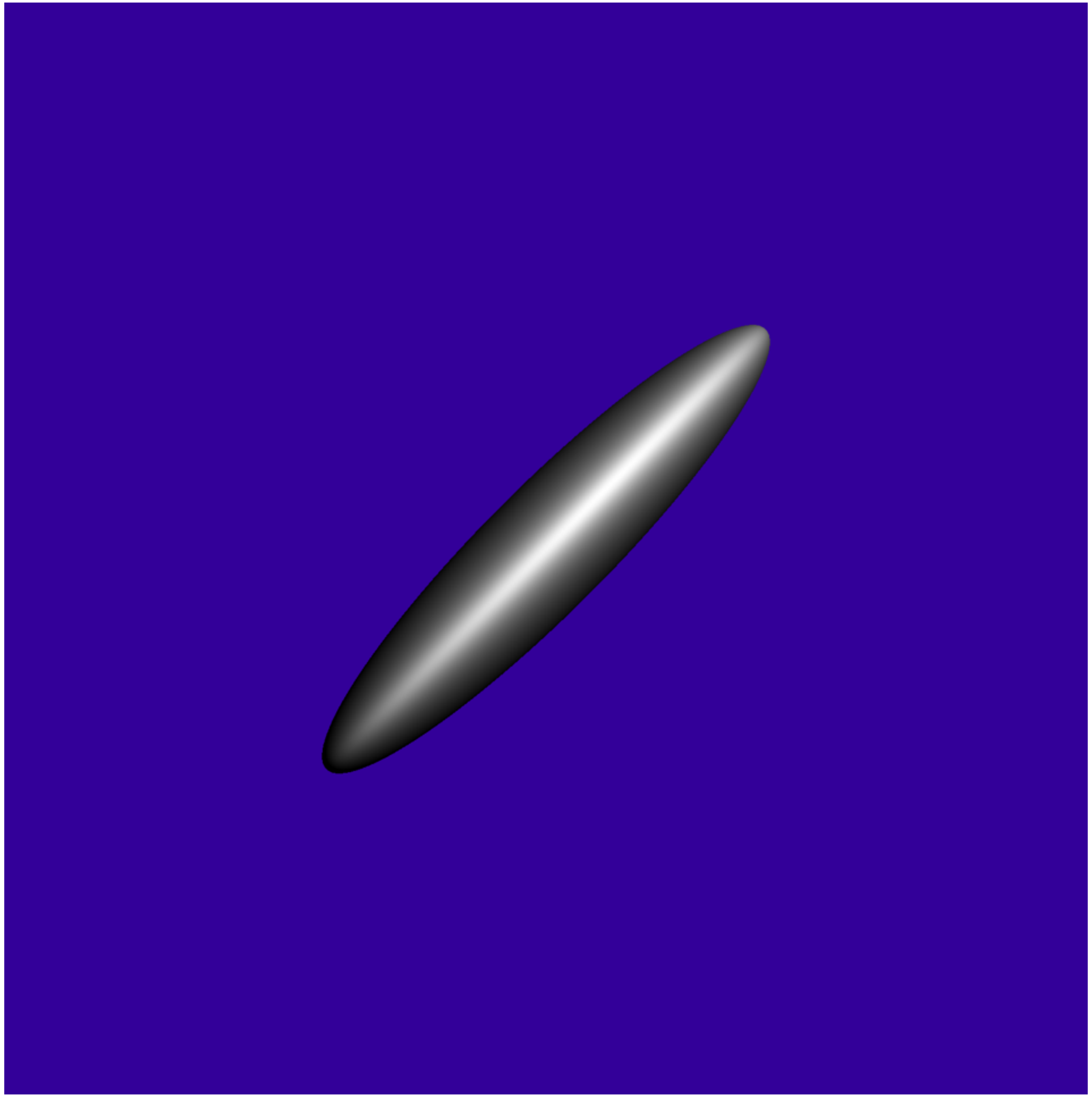
Next week

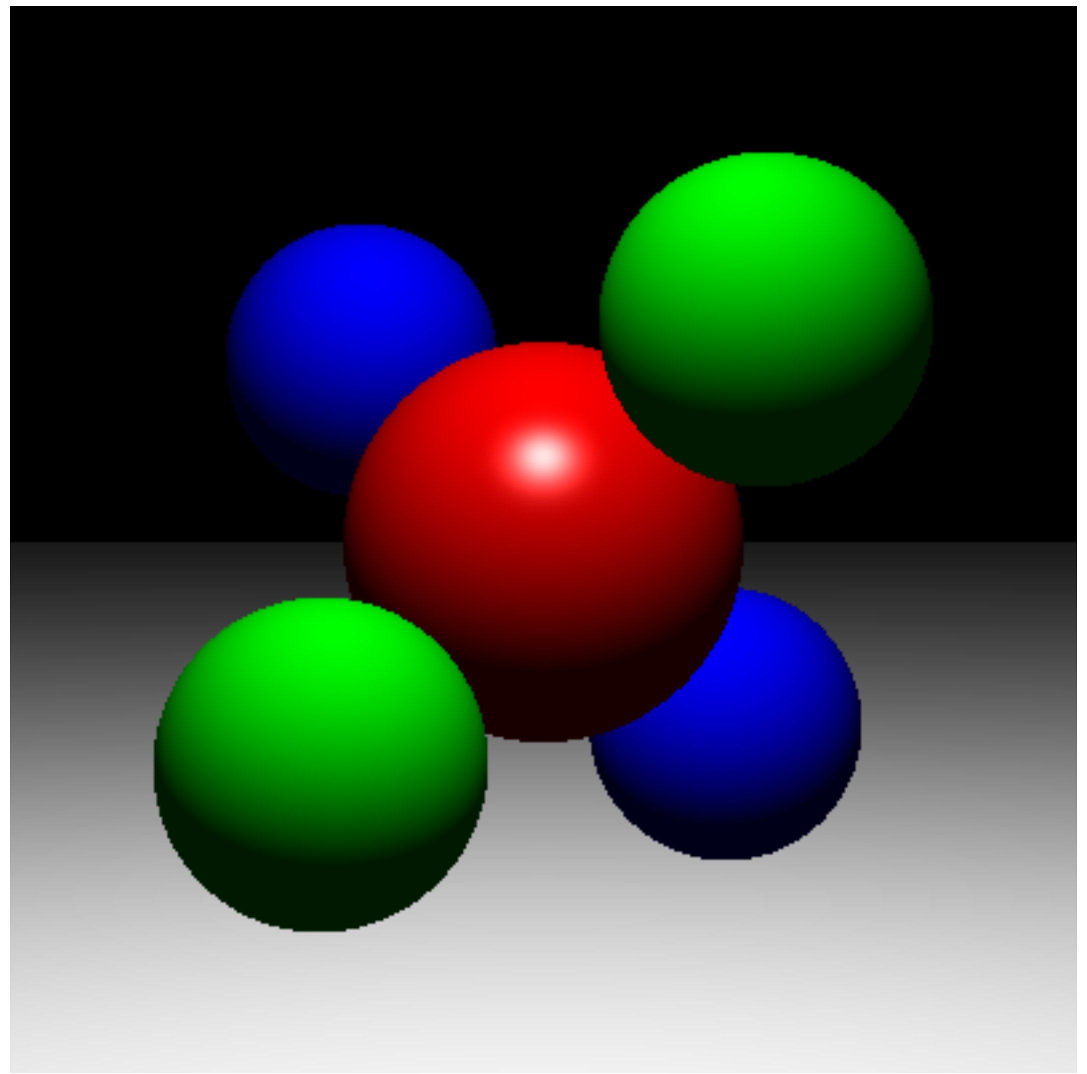
Recap of diffuse shading

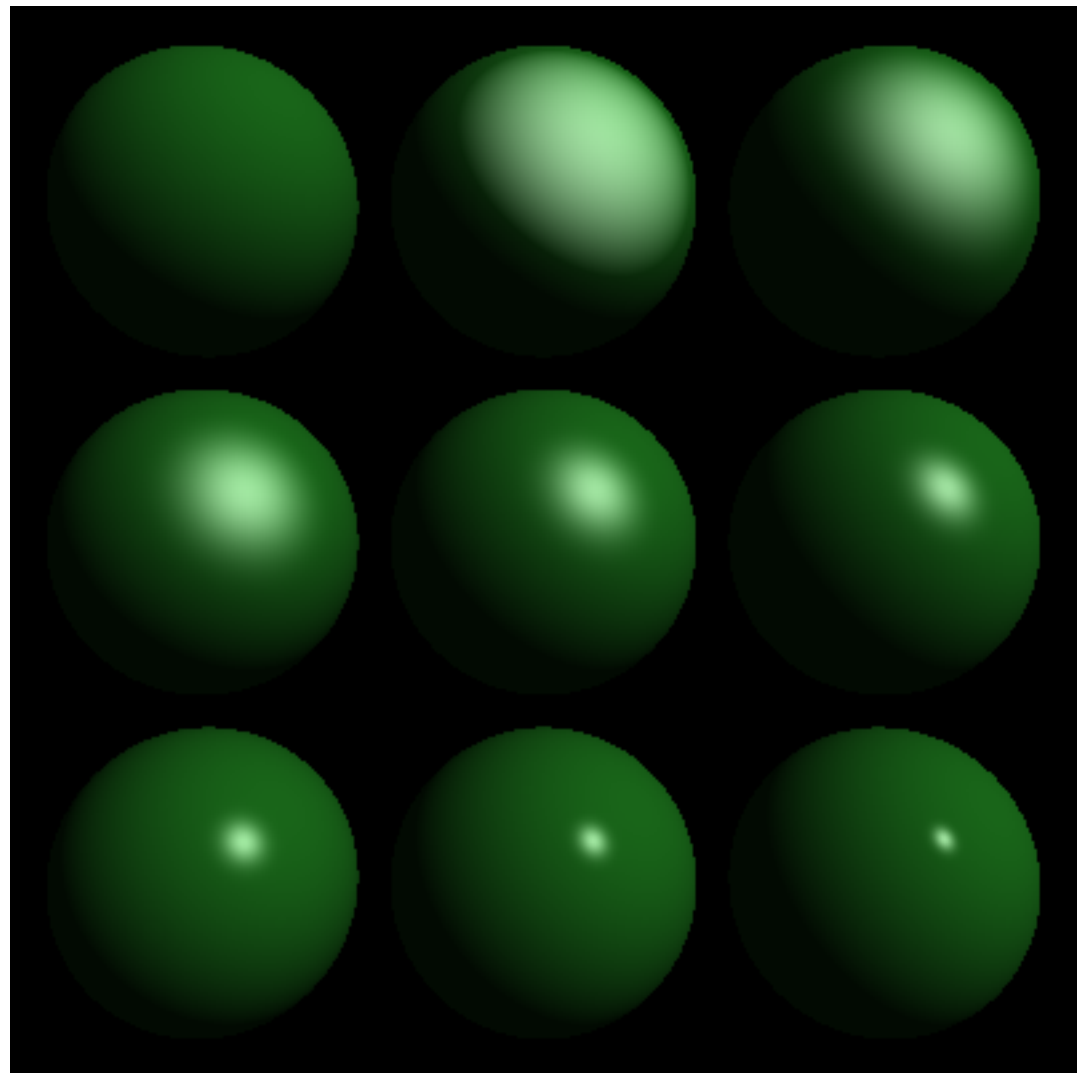
Mob programming (diffuse shading)

TRANSFORMS

Manipulate primitives with transformation matrices







DUE NEXT SESSION

full phong shading
reference the guide

Specular shading

Phong shading model

Next week

Recap of diffuse shading

Mob programming (diffuse shading)

TYPES OF LIGHTS

Ambient light: simplified indirect illumination

Point light: light with position, emits equally in all directions

Directional light: point light infinitely far away, all light comes from same direction

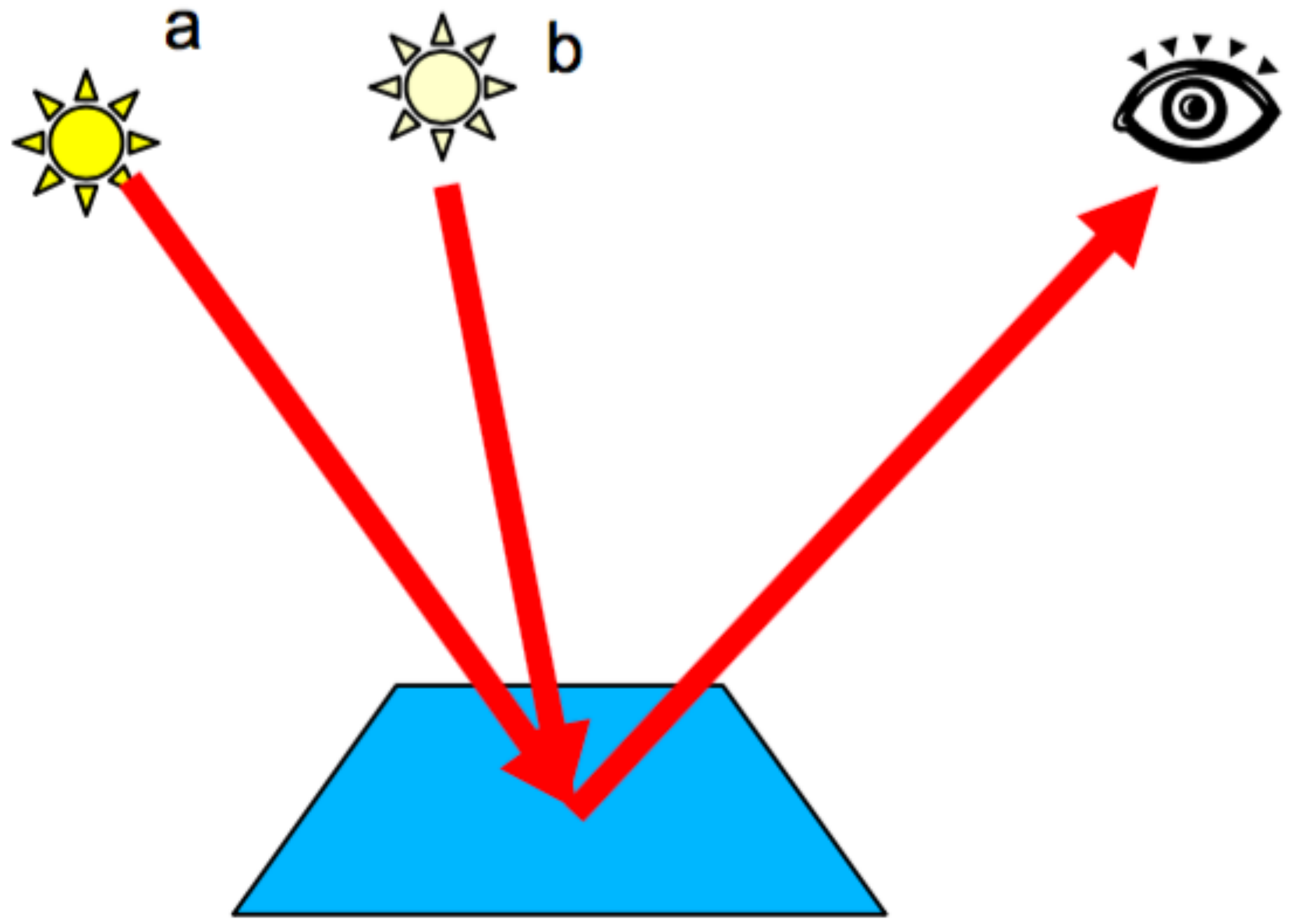
LINEARITY

Light intensity/color is additive

We'll process each light source separately and sum the results

$$I(a+b) = I(a) + I(b)$$

$$I(sa) = sI(a)$$

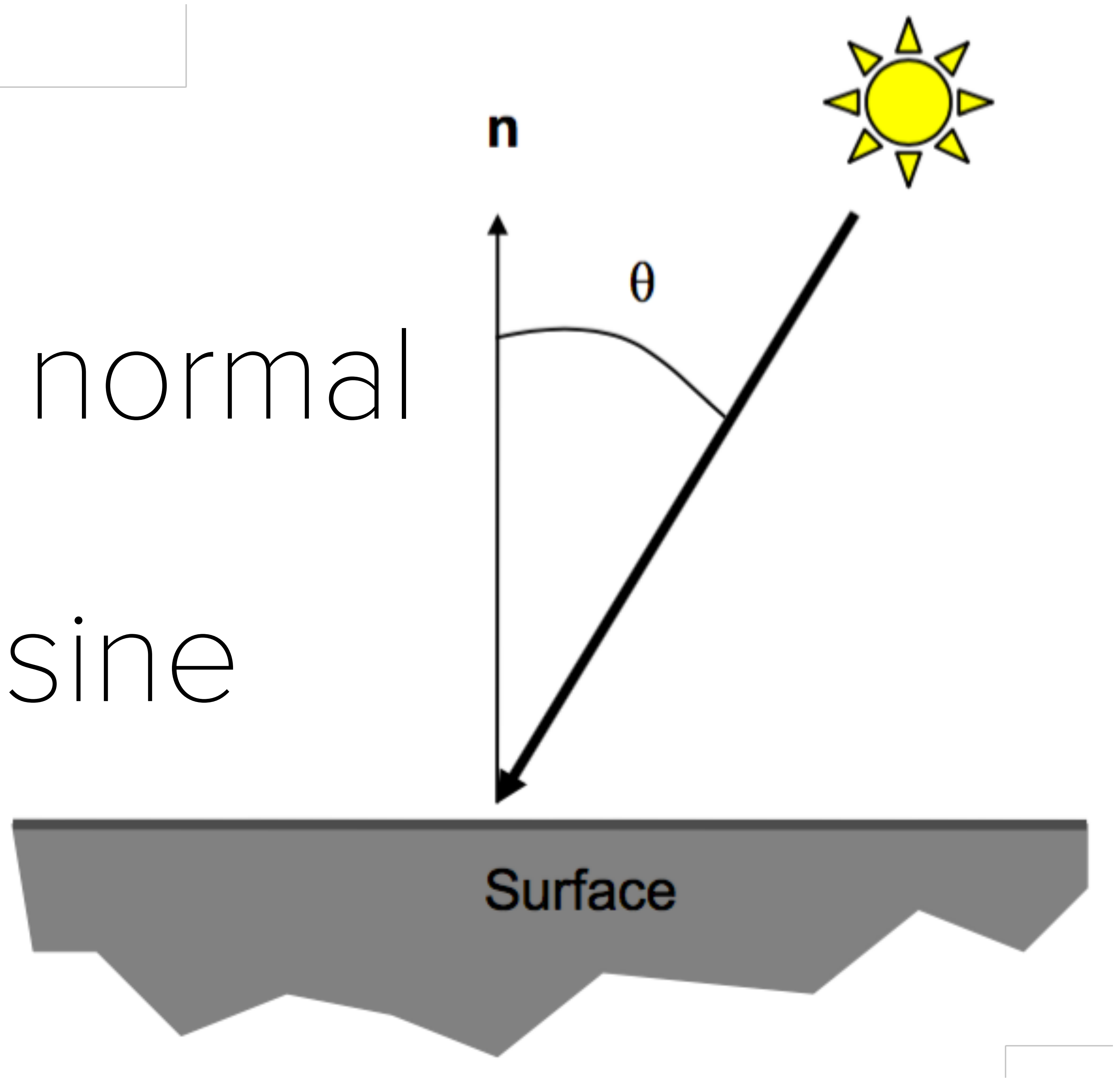


INCOMING IRRADIANCE

Amount of light energy received by surface depends on incoming angle

Largest when light is lined up with normal

Why? Dot product depends on cosine

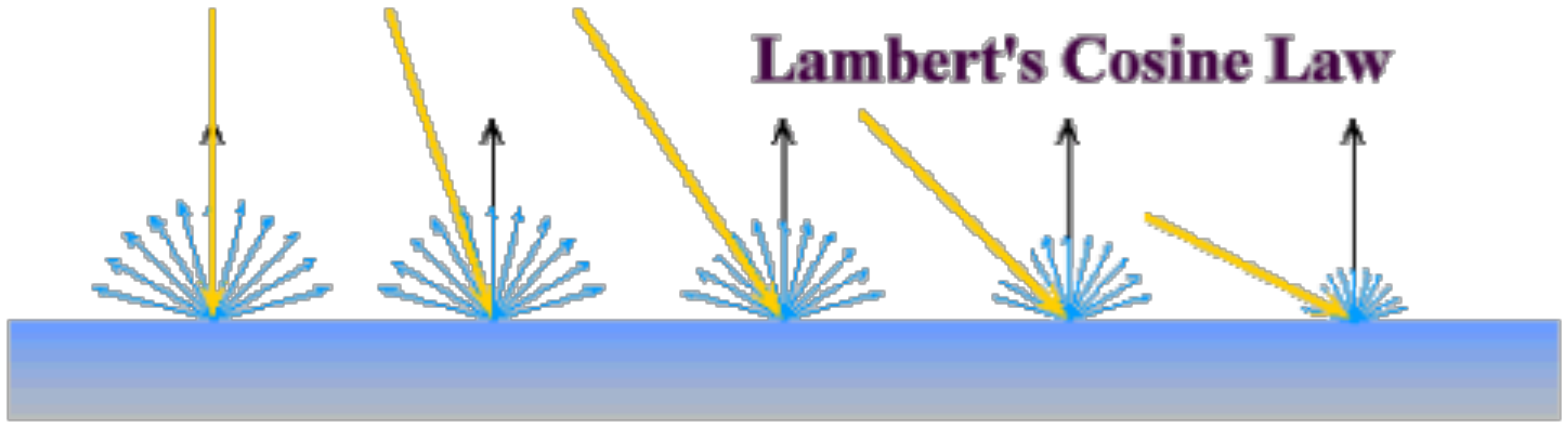


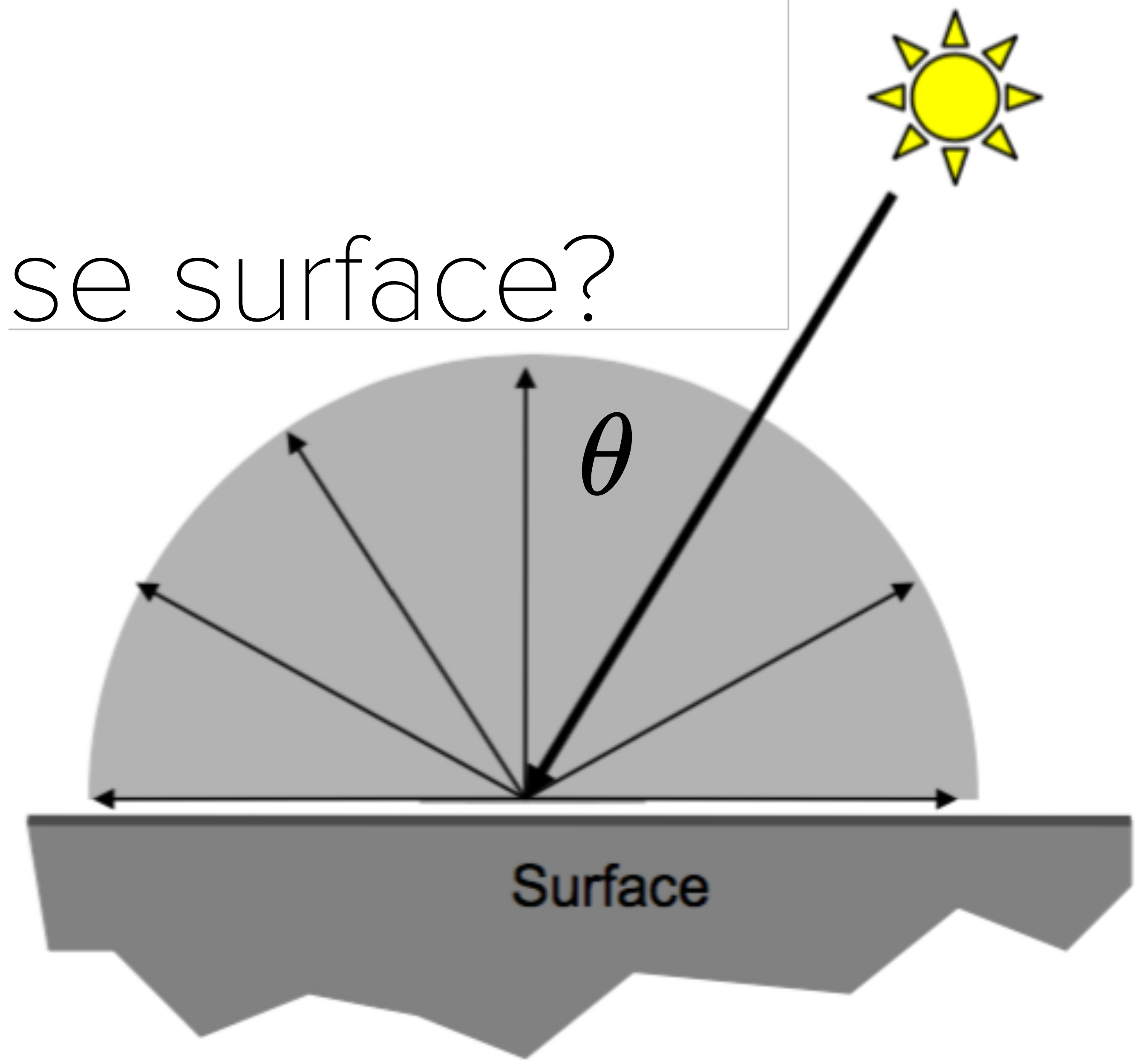
DIFFUSE SURFACES

Perfect/ideal diffuse surface is very rough at the microscopic level (chalk, clay, some paints)

Reflected light varies with cosine (Lambert's cosine law) but not viewing angle

What do we know about BRDF of ideally diffuse surface?





DIFFUSE CALCULATION

$$L_o = k_d \, max(0, n \cdot l) \, L_i$$

For a single light source

k_d diffuse coefficient (color)

n surface normal

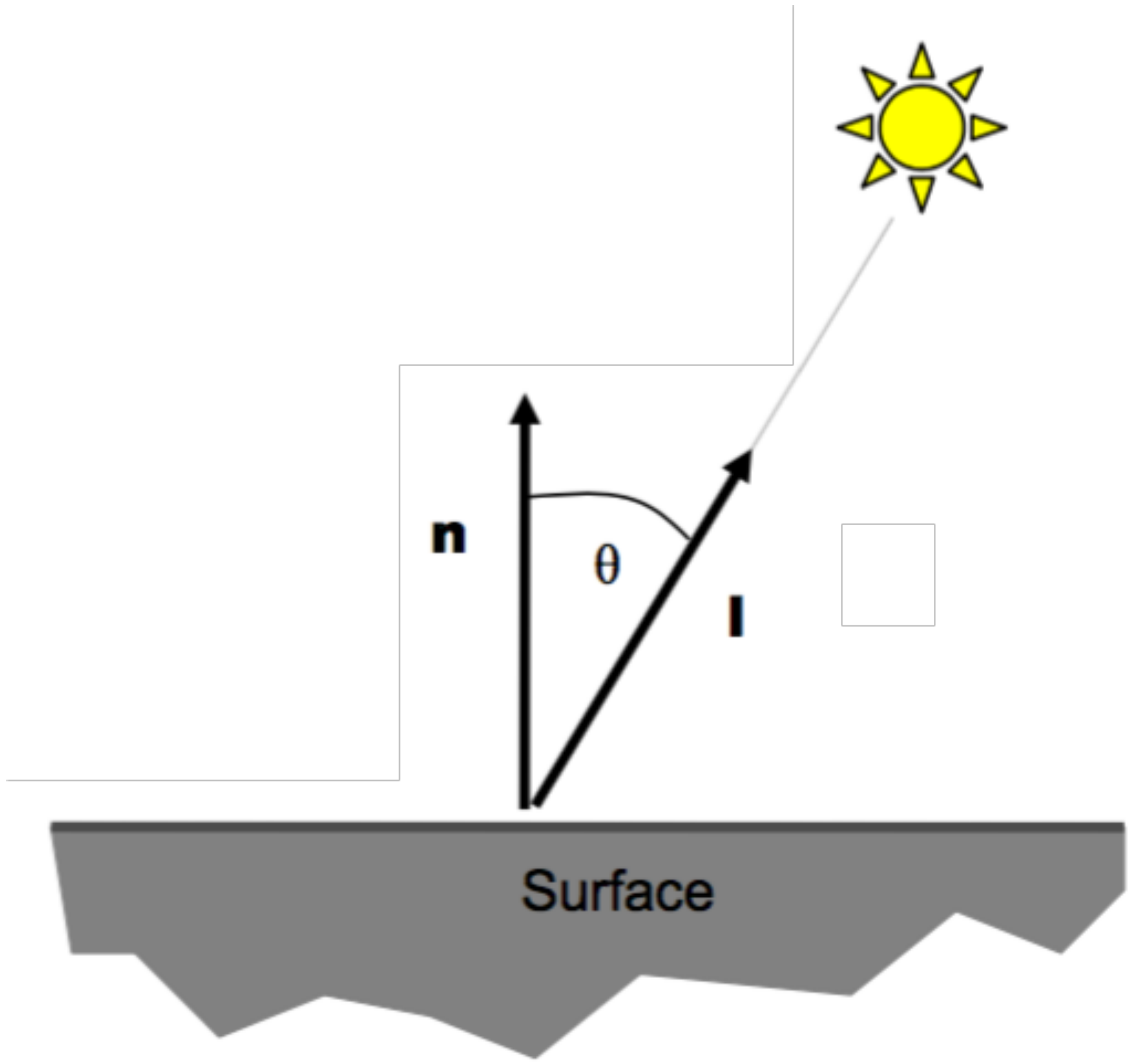
l light direction

L_i light intensity (color)

L_o shaded color

Why clamp the dot product?

Don't forget to normalize $\boldsymbol{n}$ and $\boldsymbol{l}$



Specular shading

Phong shading model

Next week

Recap of diffuse shading

Mob programming (diffuse shading)