Computer Graphics

Lecture 9
Light Sources
Diffuse Shading

Announcements

- Monday and Tuesday flipped (videos are posted):

  - 3 videos on mirrors, gloss, and shadows for Monday

  - 2 videos on ray-triangle intersection for Tuesday

Artifact voting opens at after class

Cast your vote by Sunday night!

Announcements

- Some A1 stats:

  - Time since release: 4 days

  - Time to deadline: 5 days

  - Office hours remaining before deadline: 3

  - A1 questions in office hours so far: 0

  - Pairs that have created a repository: 8

  - Pairs that have made any commits to their repo: 0

Goals

- Understand the definition of point and directional light sources.

- Know how to calculate diffuse shading for Lambertian surfaces.
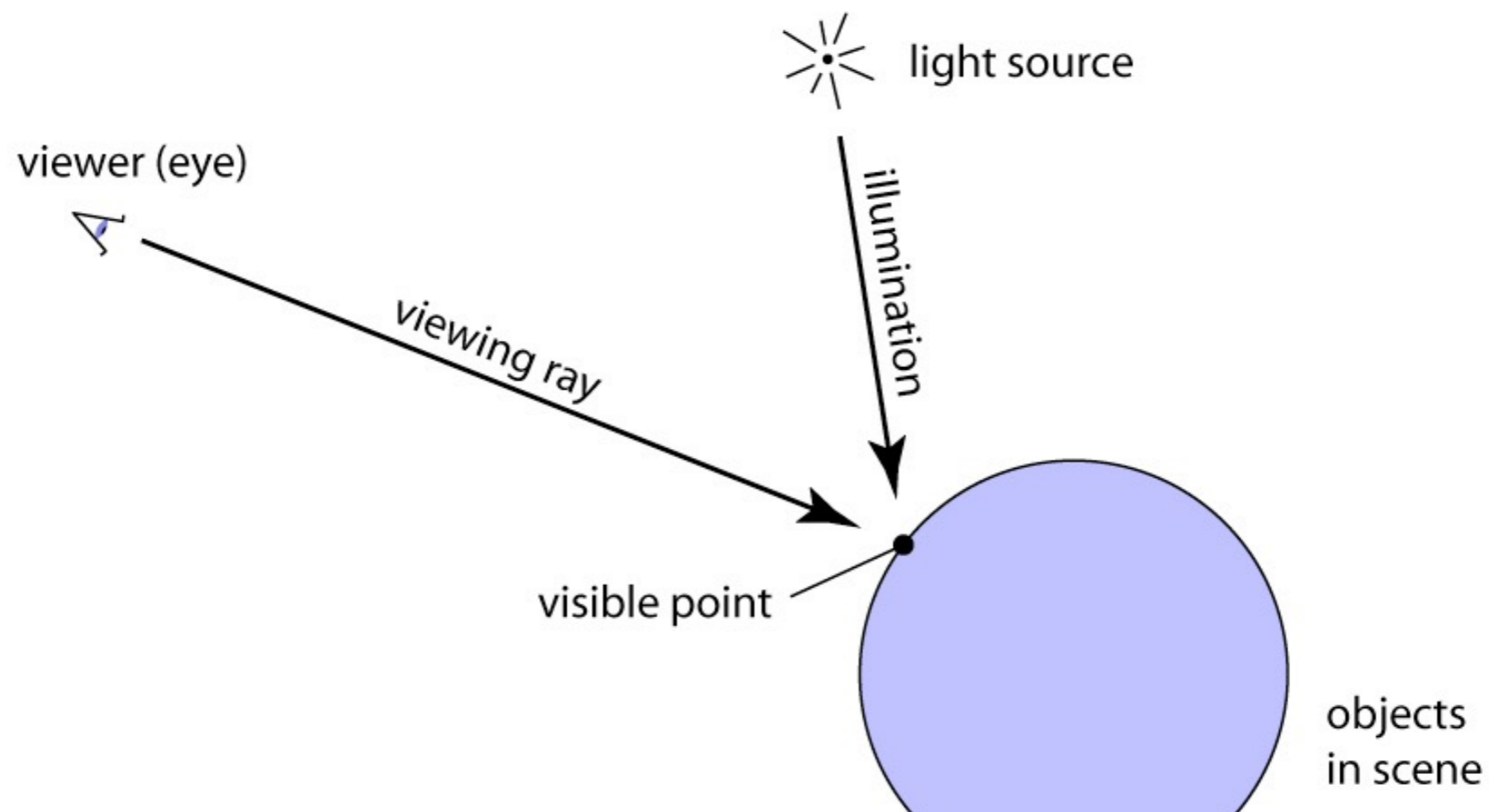
Ray Tracing: Pseudocode

```
for each pixel:
      generate a viewing ray for the pixel
      find the closest object it intersects
      determine the color of the object
```

Ray-Sphere: Code Sketch

```
function ray_intersect(ray, sphere, tmin, tmax):
```

- Use last lecture's math to find $\pm t$

- If no real solutions, return `nothing`

- Otherwise, return closest t that lies between `tmin` and `tmax`

- Also return info needed for shading - store in a `HitRecord` struct.

In A2: t, intersection point, normal, texture coordinate, object

Ray-Sphere: Code Sketch

```
function ray_intersect(ray, sphere, tmin, tmax):
```
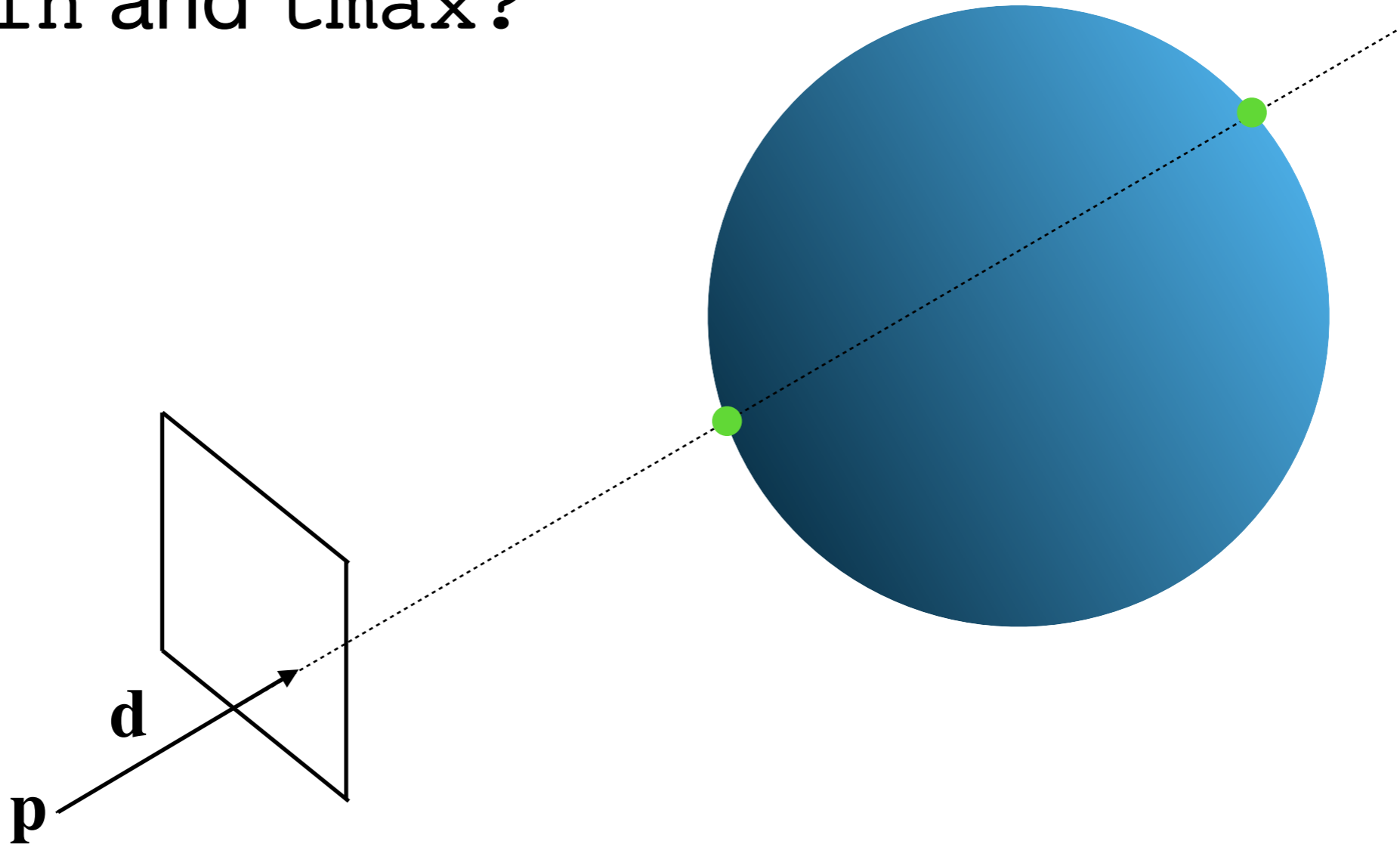
Why `tmin` and `tmax`?

Ray-Sphere: Code Sketch

```
function ray_intersect(ray, sphere, tmin, tmax):
```
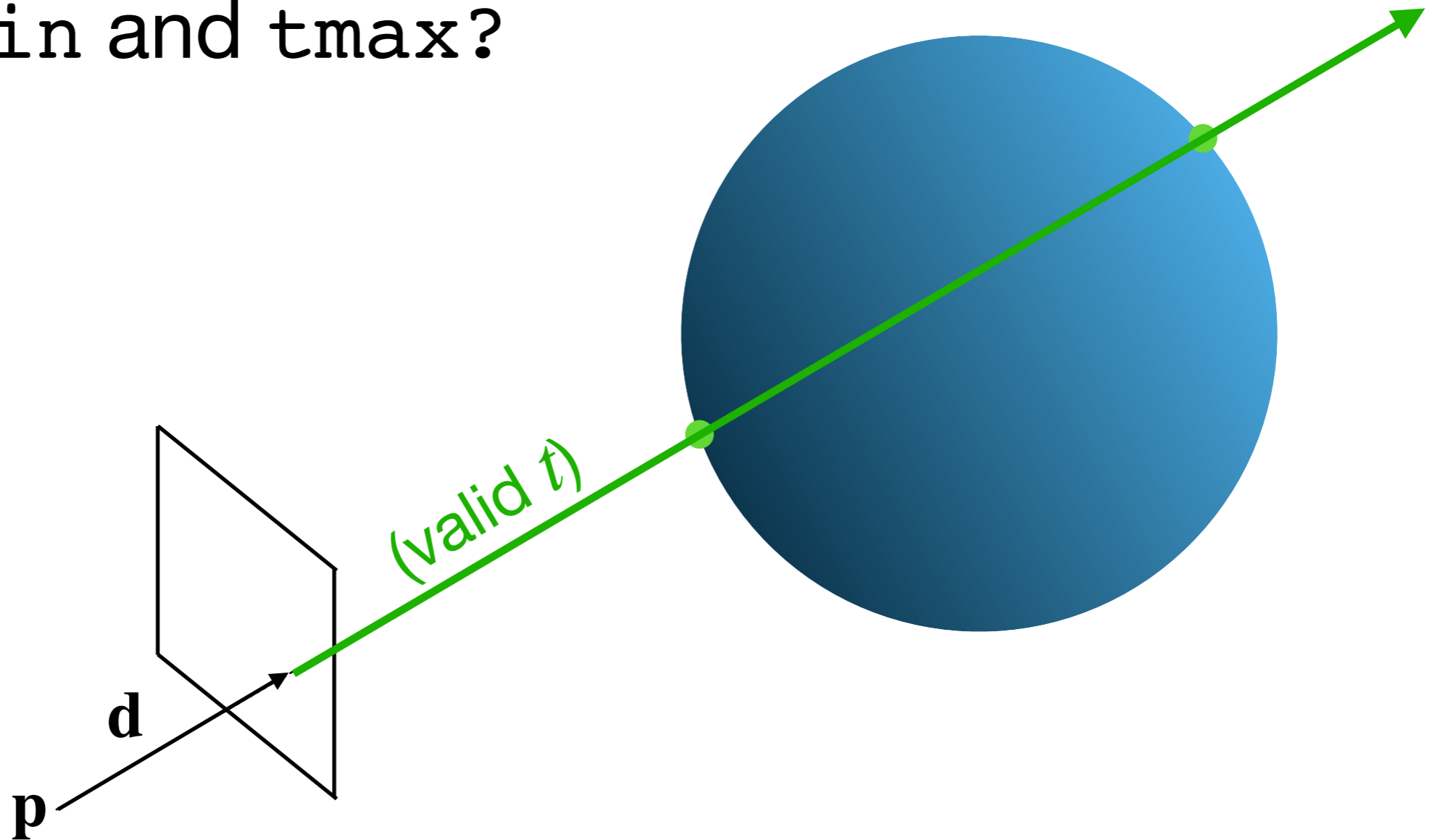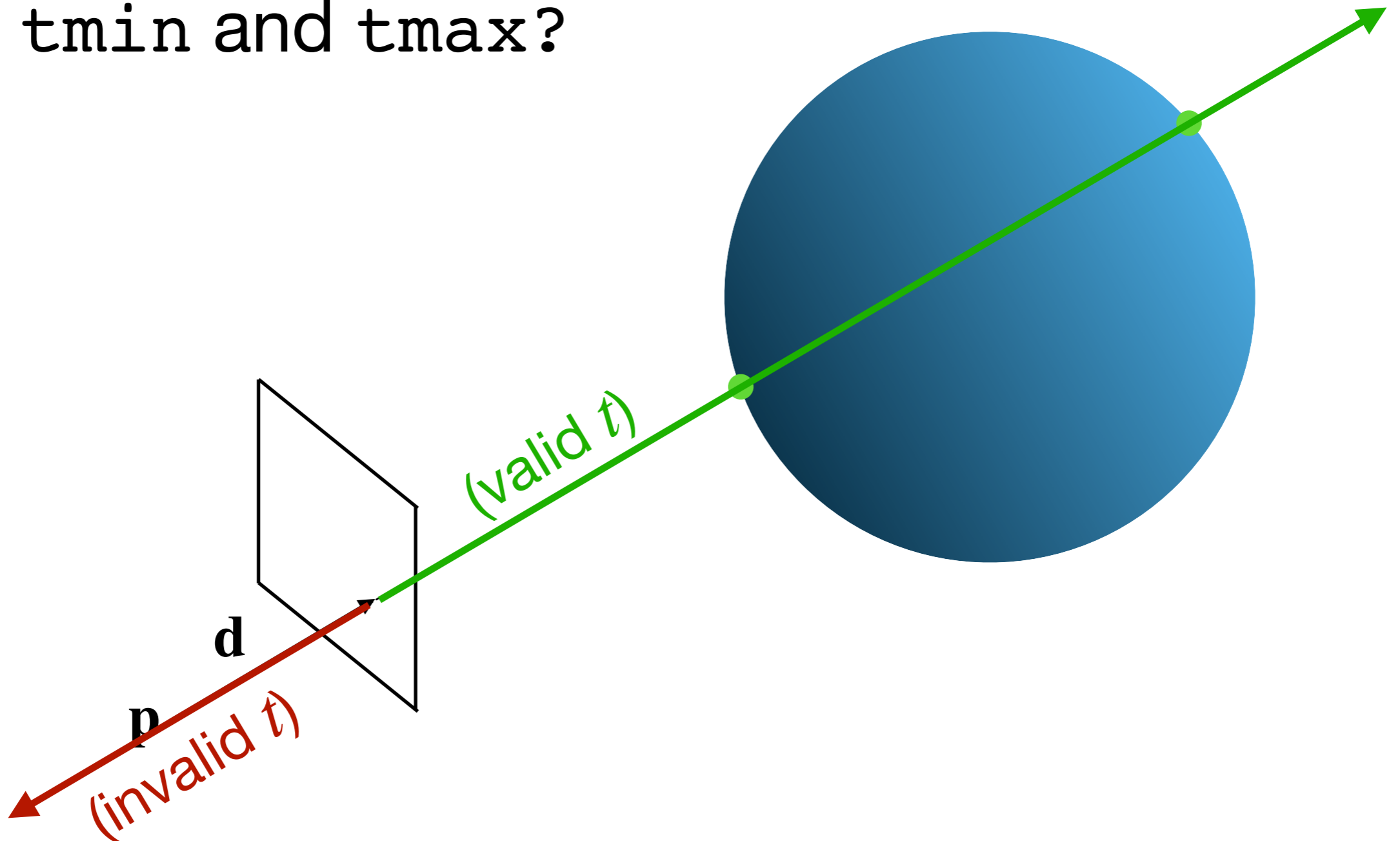
Why `tmin` and `tmax`?

Ray-Sphere: Code Sketch

```
function ray_intersect(ray, sphere, tmin, tmax):
```

Why `tmin` and `tmax`?

(valid t)

d

p

Ray-Sphere: Code Sketch

```
function ray_intersect(ray, sphere, tmin, tmax):
```
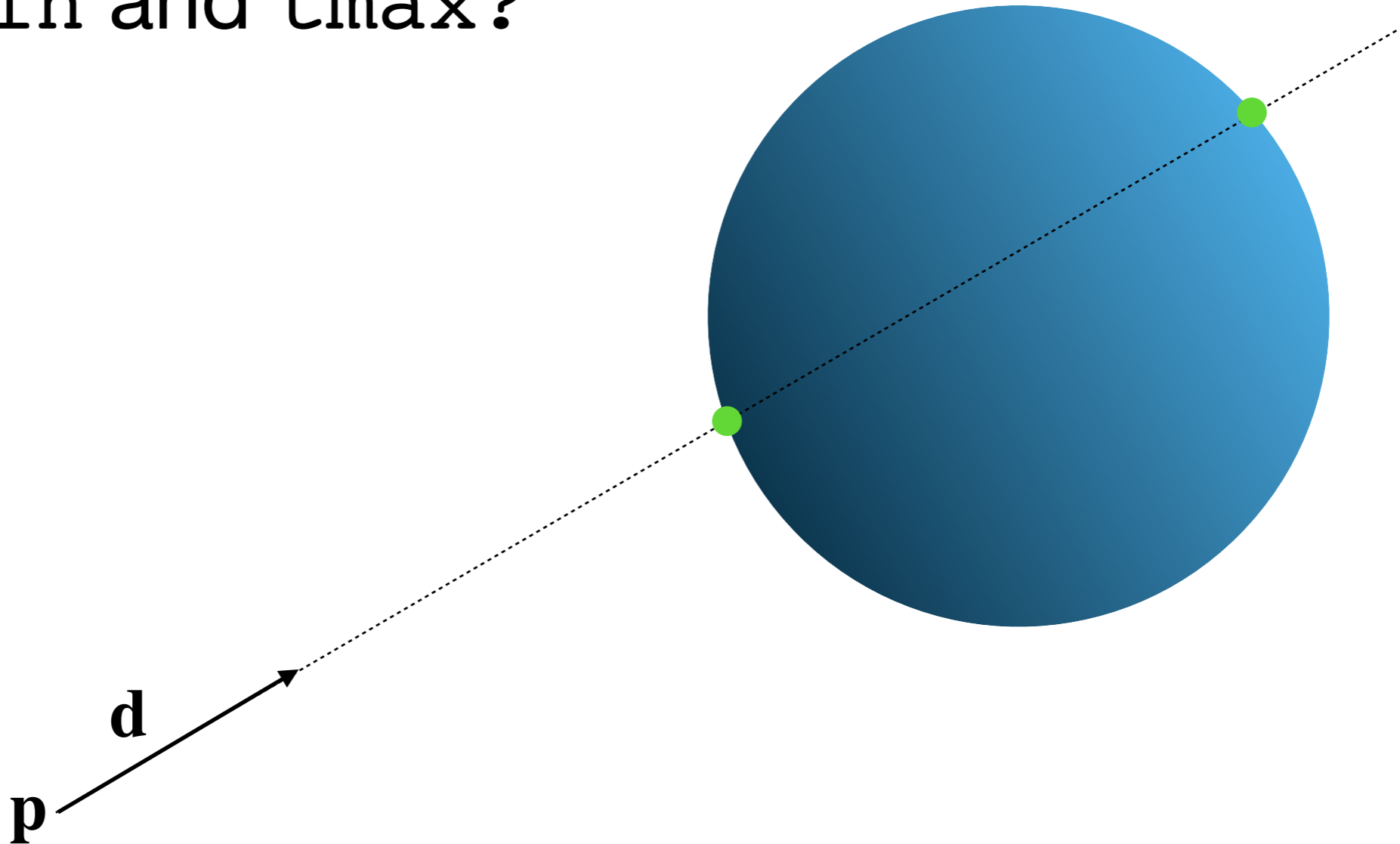
Why `tmin` and `tmax`?

Ray Tracing: Code Sketch

```
scene = model_scene()
for each pixel (i,j):
    ray = get_view_ray(i, j)
    canvas[i,j] = traceray(scene, ray, tmin, tmax)
```

Ray Tracing: Code Sketch

```
scene = model_scene()
for each pixel (i,j):
    ray = get_view_ray(i, j)
    canvas[i,j] = traceray(scene, ray, tmin, tmax)


function traceray(scene, ray, tmin, tmax):
    t, rec = ray_intersect(ray, scene, tmin, tmax)
    if rec != nothing:
        canvas[i,j] = rec.obj.color
    else:
        canvas[i,j] = scene.bgcolor
```
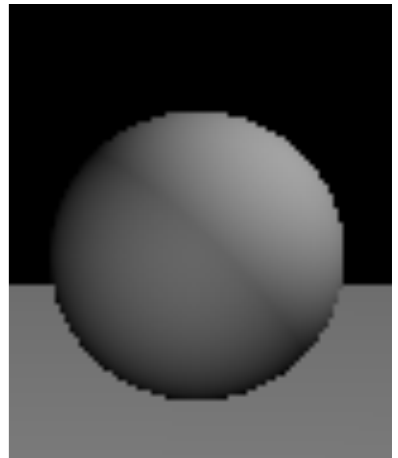
Ray Tracing: Code Sketch

```
scene = model_scene()

for each pixel (i,j):

    ray = get_view_ray(i, j)

    canvas[i,j] = traceray(scene, ray, tmin, tmax)


function traceray(scene, ray, tmin, tmax):

    t, rec = ray_intersect(ray, scene, tmin, tmax)

    if rec != nothing:

      canvas[i,j] = rec.obj.color

    else:

      canvas[i,j] = scene.bgcolor
```

Ray Tracing: Code Sketch

```
scene = model_scene()
for each pixel (i,j):
    ray = get_view_ray(i, j)
    canvas[i,j] = traceray(scene, ray, tmin, tmax)


function traceray(scene, ray, tmin, tmax):
    t, rec = ray_intersect(ray, scene, tmin, tmax)
    if rec != nothing:
        canvas[i,j] = rec.obj.color       Let's work on this.
    else:
        canvas[i,j] = scene.bgcolor
```
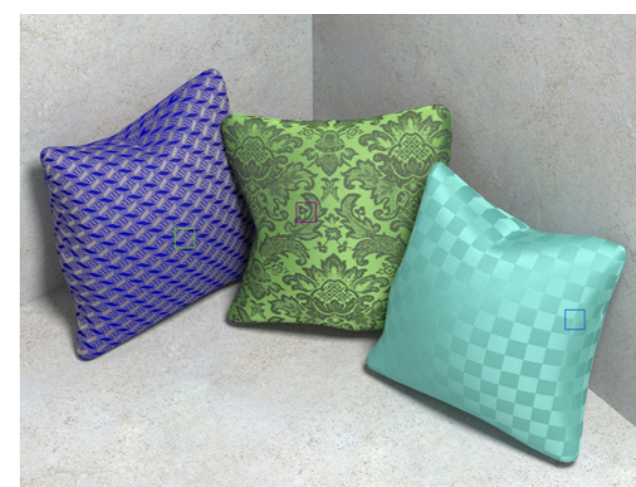
Shading

What does the color of a pixel depend on?

Shading

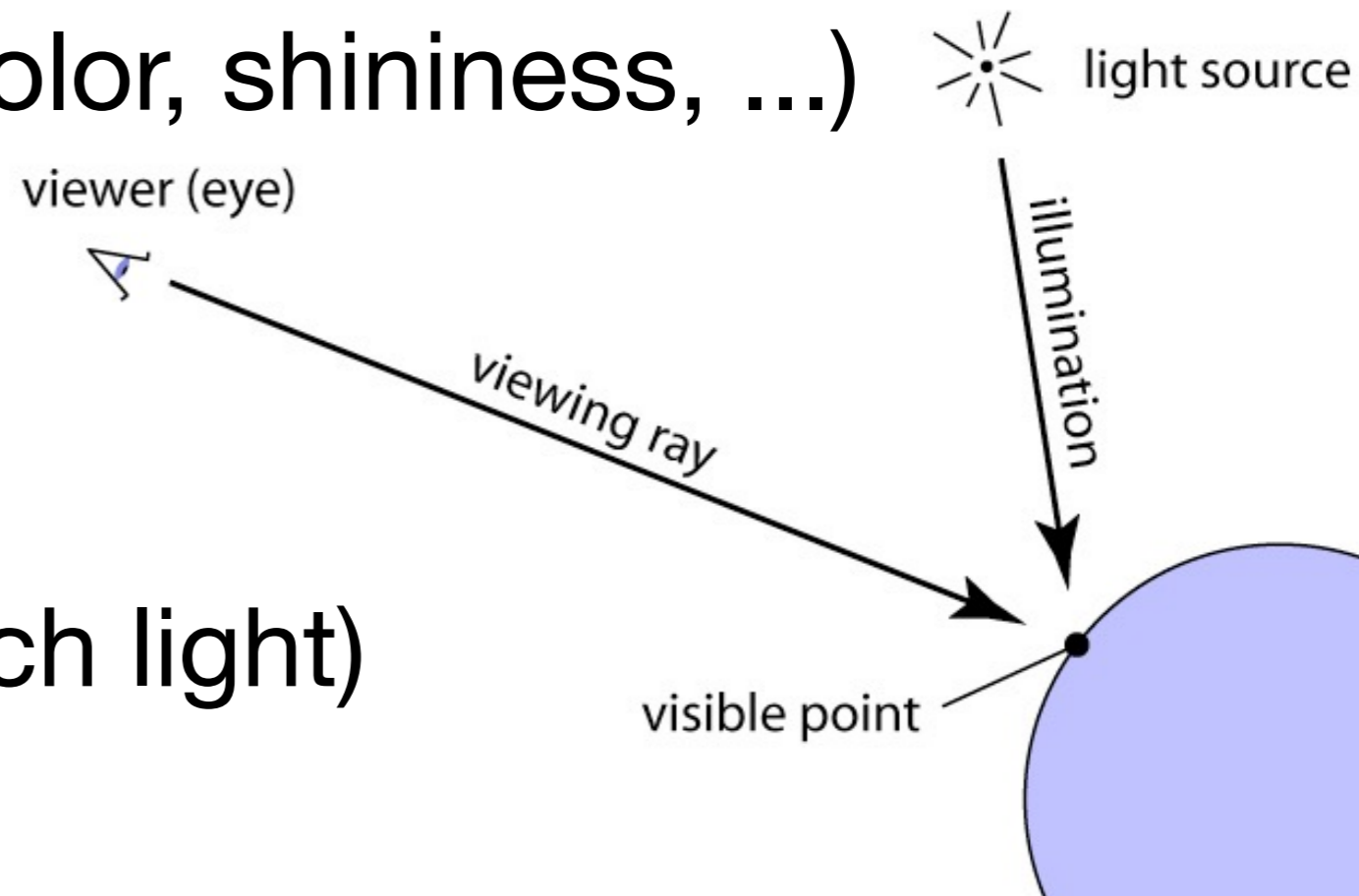What does the color of a pixel depend on?

Try to think beyond matte gray spheres.

Wood? Velvet? Hair? Brushed stainless steel? Glass? Wax?

Shading

What does the color of a pixel depend on?

- surface normal

- surface properties (color, shininess, ...)

- eye direction

- light direction (for each light)

light source

viewer (eye)
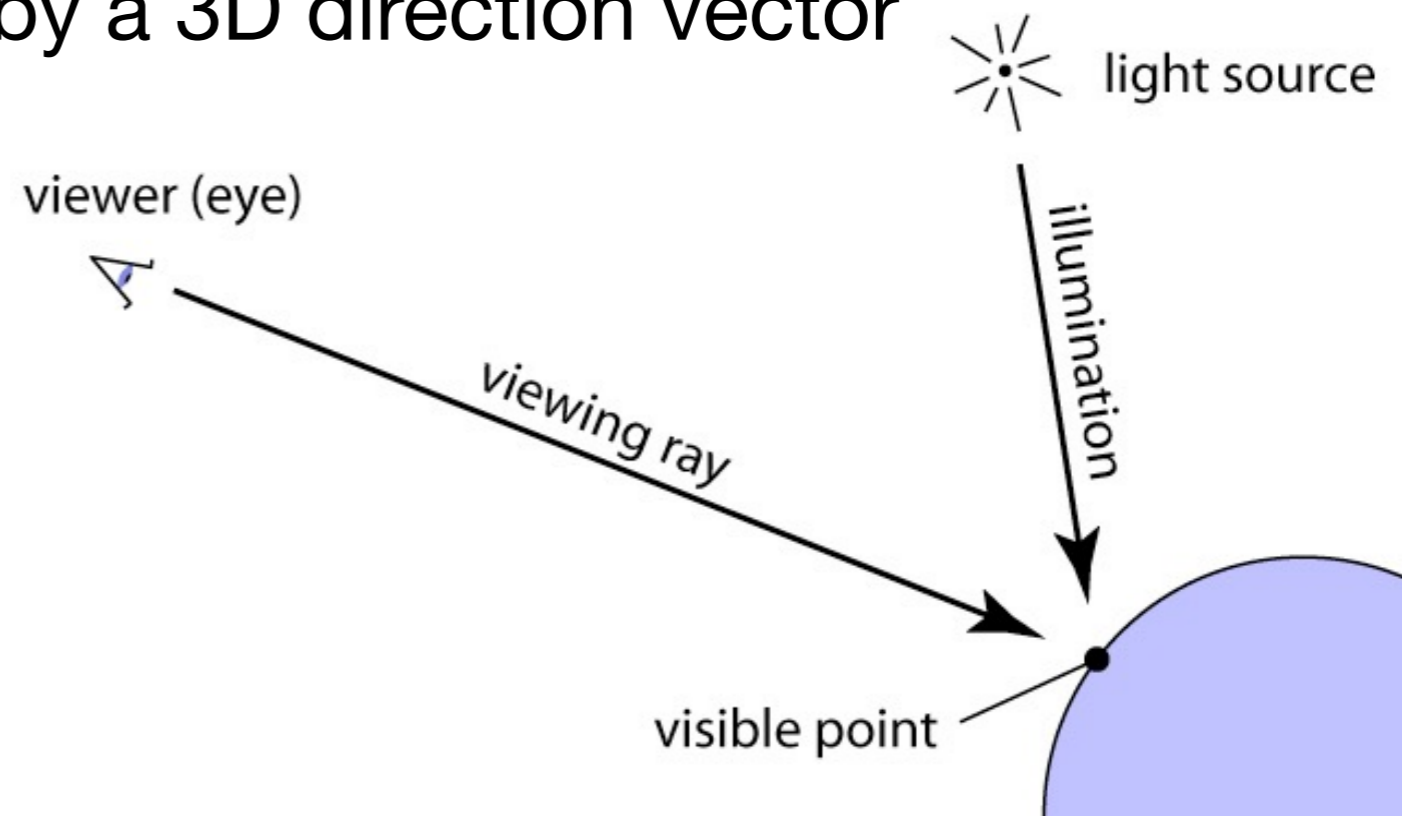
viewing ray

illumination

visible point

Shading

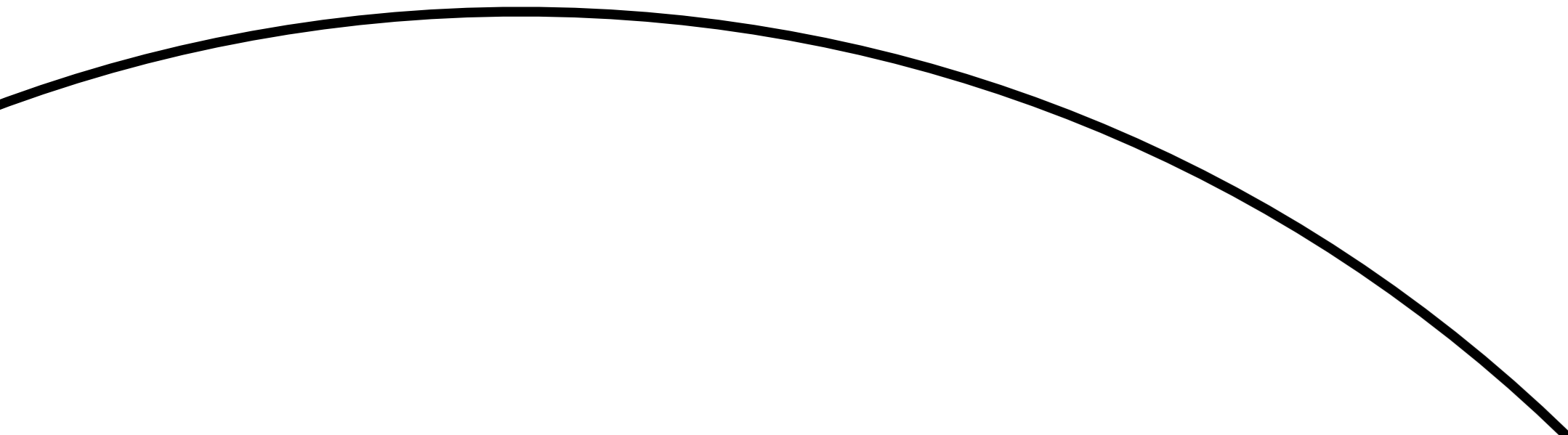What does the color of a pixel depend on?

- surface normal   *stored in or calculated from object*

- surface properties (color, shininess, ...)   *stored in object*

- eye direction   *calculated from viewing ray and intersection point*

- light direction (for each light)   *calculated from light and intersection point*

Light Sources

- Where does light come from?

- Two simple kinds of sources:

  - point source: defined by a 3D position

  - directional source: defined by a 3D direction vector

  - ...many other possibilities!

Point and Directional Lights

Shading

What does the color of a pixel depend on?

- surface normal *stored in or calculated from object*

- surface properties (color, shininess, ...) *stored in object*

- eye direction ***calculated*** *from viewing ray and intersection point*

- light direction (for each light) ***calculated*** *from light and intersection point*

Problems 1-2: calculated how?

Problems 1-2:
Eye Direction and Light Direction

Given a viewing ray ($\mathbf{p} + t\mathbf{d}$) and the t at which it intersects a surface,

1. Find a unit vector giving the direction from the surface towards the viewer.

2. Find a unit vector giving the direction from the surface towards:

   - a point light source at position $\vec{s}$

   - a directional light source with direction $\vec{\ell}$

Diffuse (Lambertian) Reflection

- On a *diffuse* surface, light scatters uniformly in all directions.

- No dependence on view direction.

- Many surfaces are approximately diffuse:

    - matte painted surfaces, projector screens,

    - anything that doesn't look "shiny"
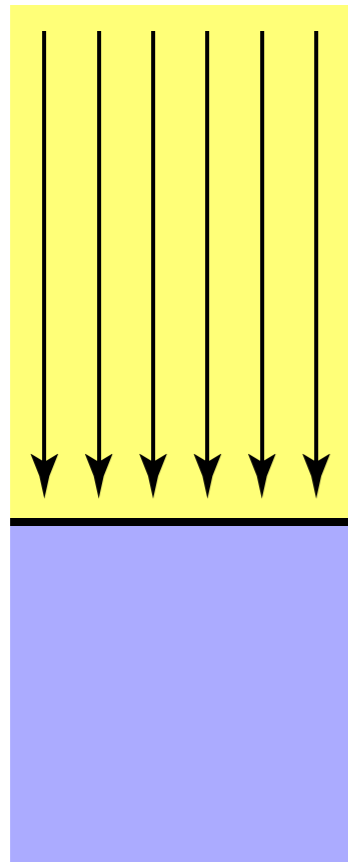
Diffuse (Lambertian) Reflection

Problem 3
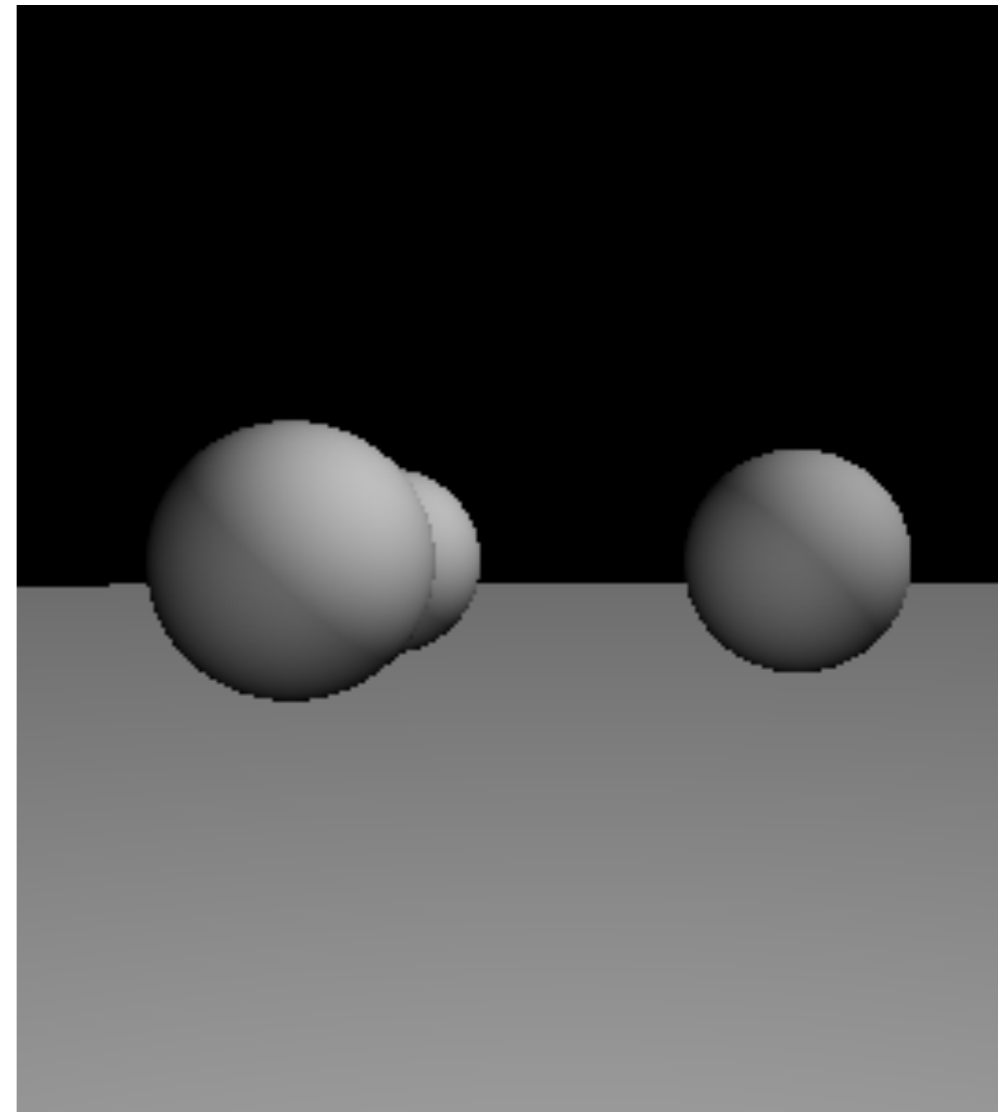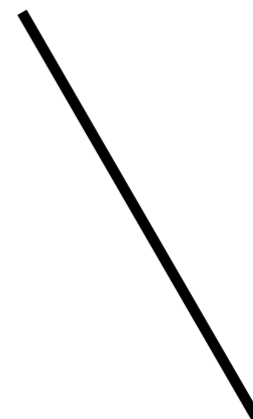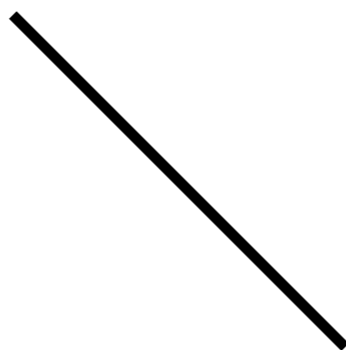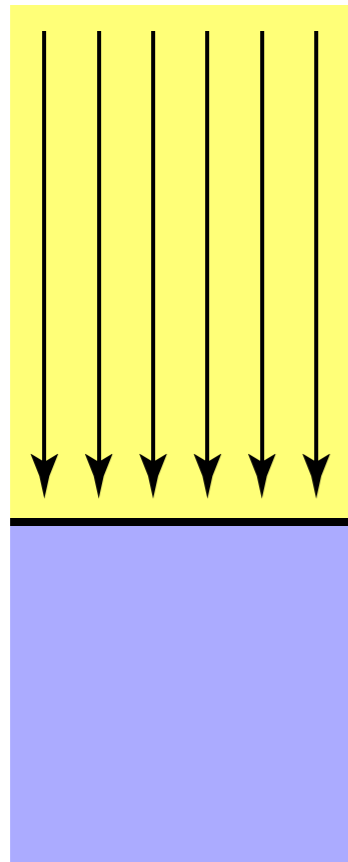
Diffuse (Lambertian) Reflection

Highly recommended reading:
https://ciechanow.ski/lights-and-shadows/

Diffuse (Lambertian) Reflection

The top face of a cube receives some amount of light.

Highly recommended reading:
https://ciechanow.ski/lights-and-shadows/

Diffuse (Lambertian) Reflection

The top face of a cube receives some amount of light.
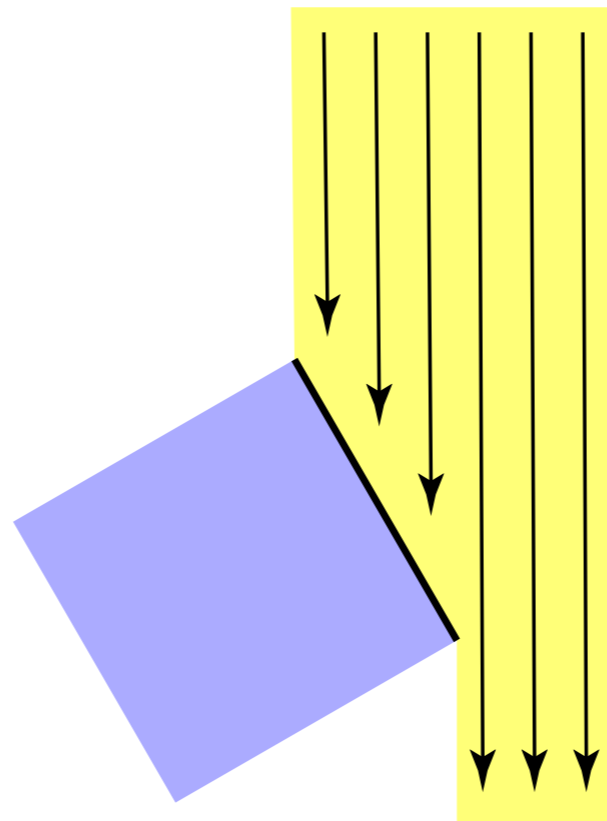
Rotated 60º, the same face receives half the light.

Highly recommended reading:
https://ciechanow.ski/lights-and-shadows/
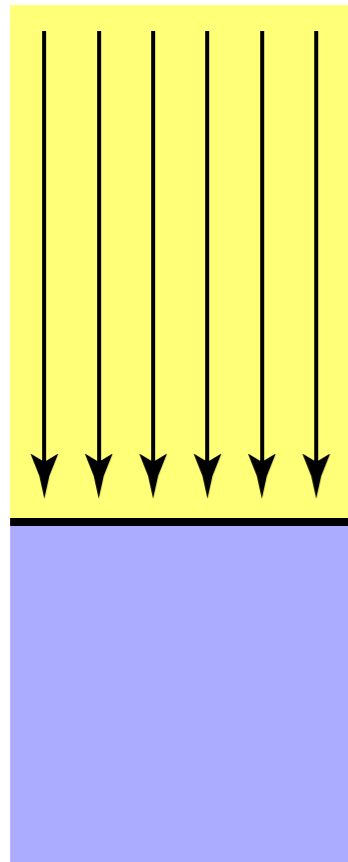
Diffuse (Lambertian) Reflection

The top face of a cube receives some amount of light.
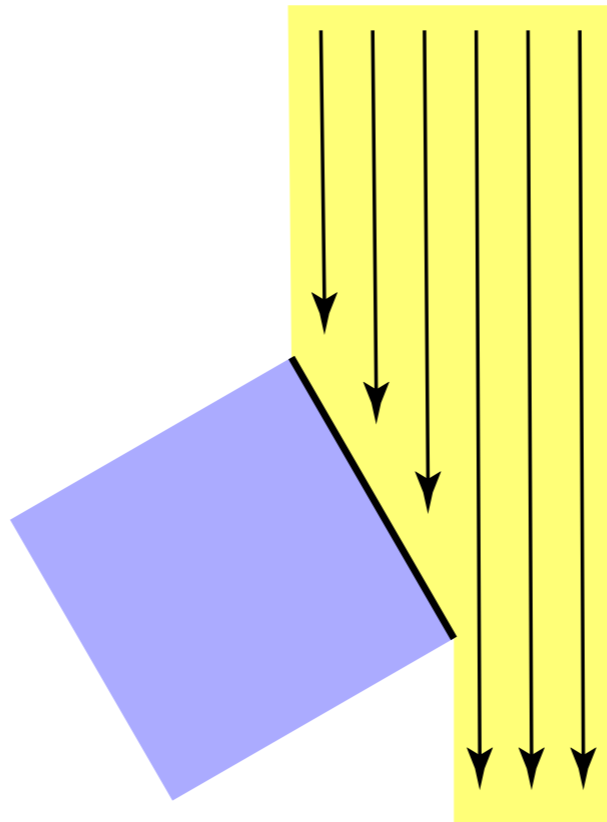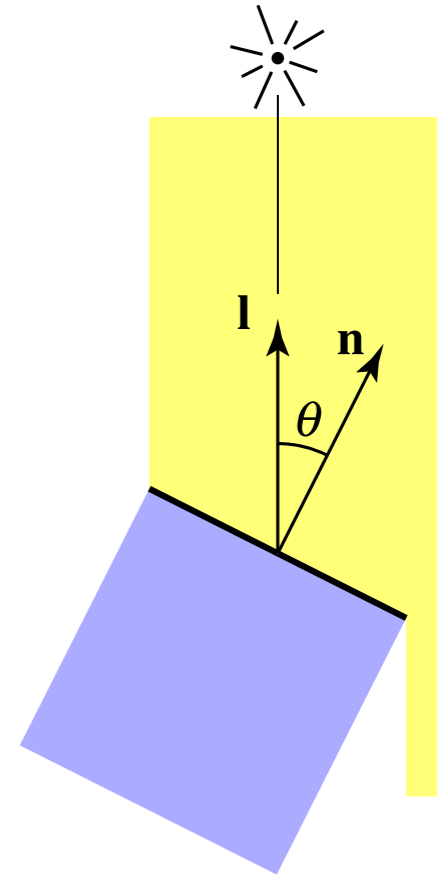
Rotated 60º, the same face receives half the light.

Light per unit area is proportional to $\cos\theta = \vec{n} \cdot \vec{\ell}$

Highly recommended reading:
https://ciechanow.ski/lights-and-shadows/
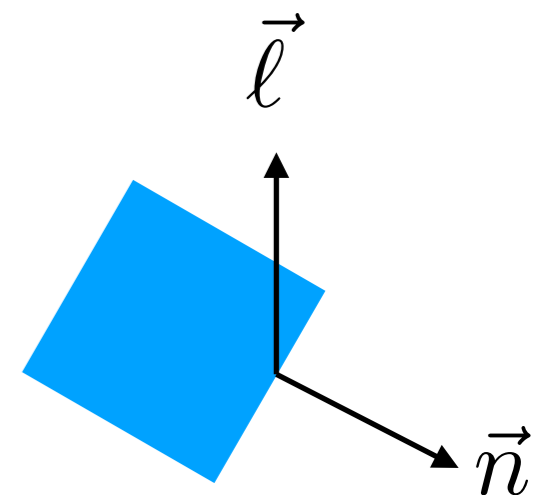
Diffuse (Lambertian) Shading

- The full model:

$$L_d = k_d I \max(0, \vec{n} \cdot \vec{\ell})$$

diffuse
coefficient

diffusely
reflected light

light intensity

Diffuse (Lambertian) Shading

- The full model:

$$L_d = k_d I \max(0, \vec{n} \cdot \vec{\ell})$$

why max with 0?

diffuse
coefficient

diffusely
reflected light

light intensity

Diffuse (Lambertian) Shading

- The full model:

$$L_d = k_d I \max(0, \vec{n} \cdot \vec{\ell})$$

why max with 0?

diffuse
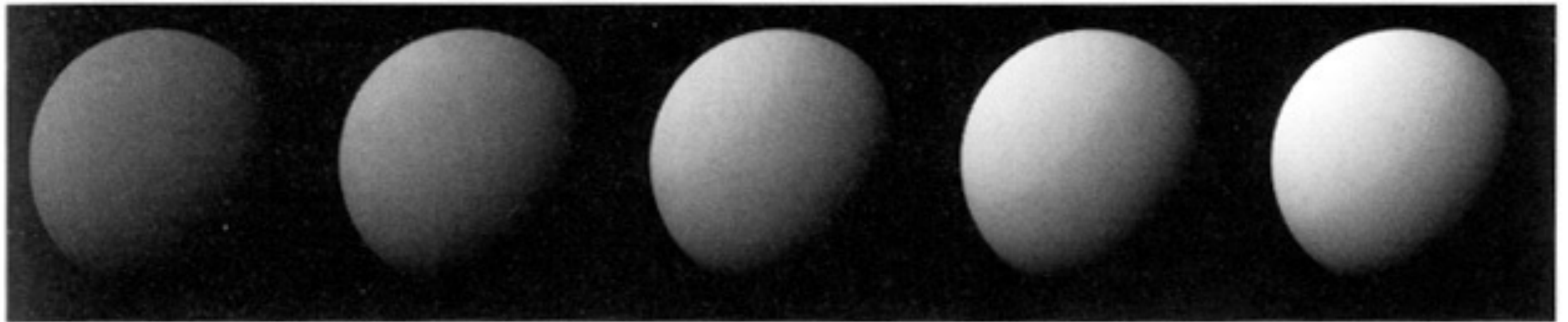coefficient

diffusely
reflected light

light intensity

$\vec{\ell}$

$\vec{n}$

Diffuse (Lambertian) Shading

$$L_d = k_d I \max(\vec{n} \cdot \vec{\ell})$$

$k_d \longrightarrow$

[Foley et al.]

For colored objects, k_d is a 3-vector of R, G, and B reflectances.

Problem 4:
Diffuse Reflection