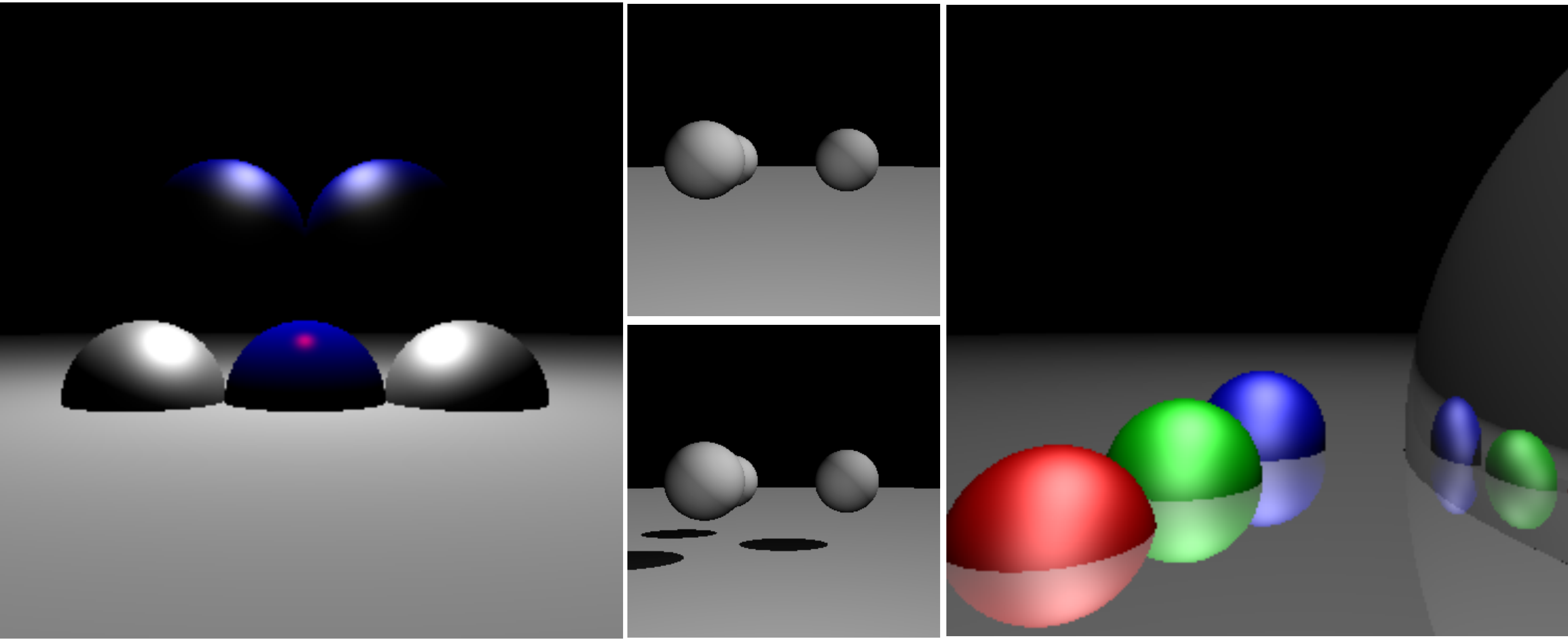


Computer Graphics

Lecture 9

Specular Shading

Shadows

Announcements

- Watch 2 videos before class on Monday:

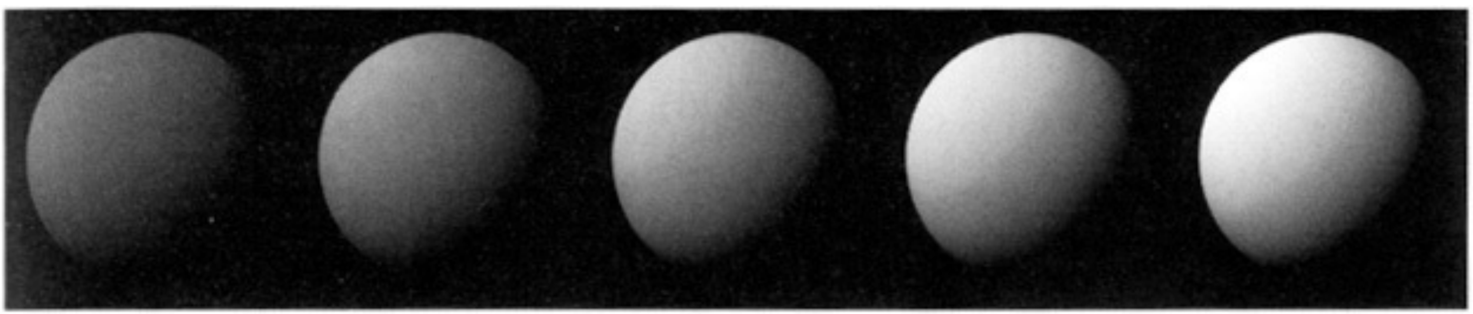
  L10A, L10B

- A1, HW1 due Monday
- A2, HW2 out Monday
  - must work with a <u>different</u> partner than A1

    start pairing now! must have partner by Tuesday

Diffuse (Lambertian) Shading

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

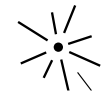


[Foley et al.]

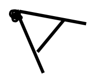
$$k_d \longrightarrow$$

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

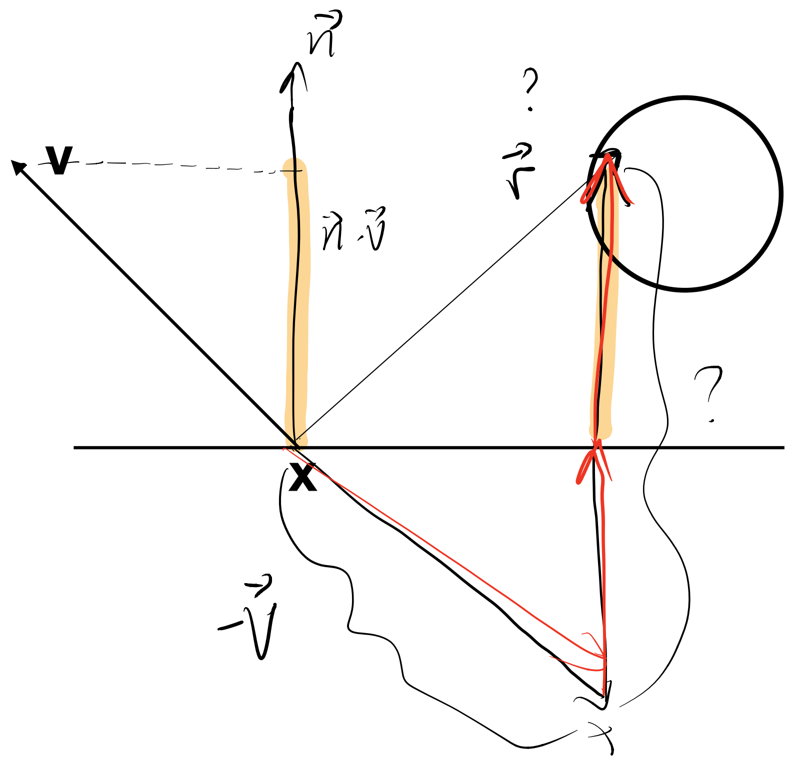
Mirror Reflection



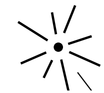
What does a camera see when it looks at a mirror?



$$\vec{r} = -\vec{v} + 2 \cdot (\vec{n} \cdot \vec{v})\,\vec{n}$$



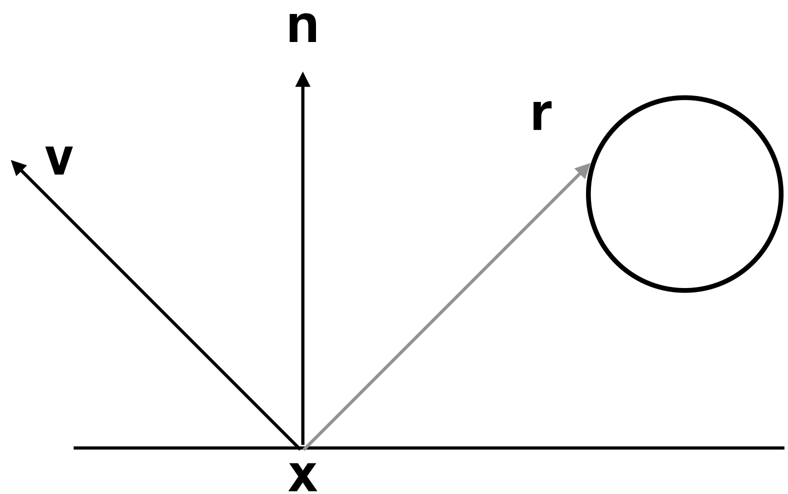
Mirror Reflection



What does a camera see when it looks at a mirror?

From last time:

$$\vec{r} = -\vec{v} + 2(\vec{v} \cdot \vec{n})\vec{n}$$

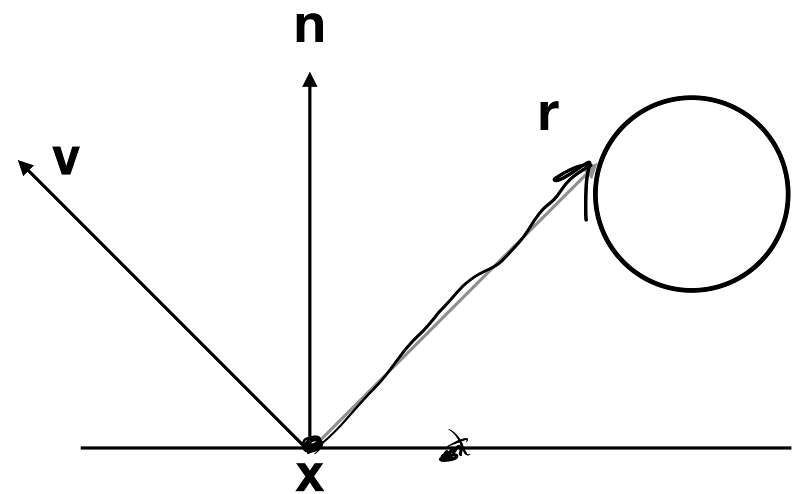


Hint: 

How do we do this using the tools we already have?

Mirror Reflection

What does a camera see when it looks at a mirror?



From last time:

$$\vec{r} = -\vec{v} + 2(\vec{v} \cdot \vec{n})\vec{n}$$

```
mirr_ray.origin = x
mirr_ray.direction = r
color = traceray(scene, mirr_ray, epsilon, Inf):
```

Hint:

small value to avoid hitting the surface x lies on

tmin

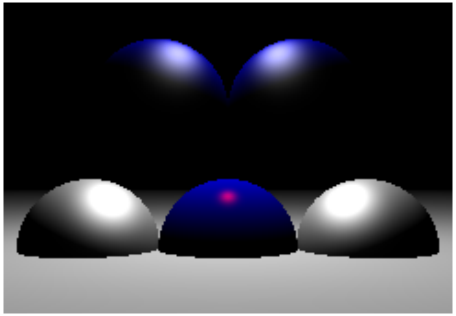
tmax

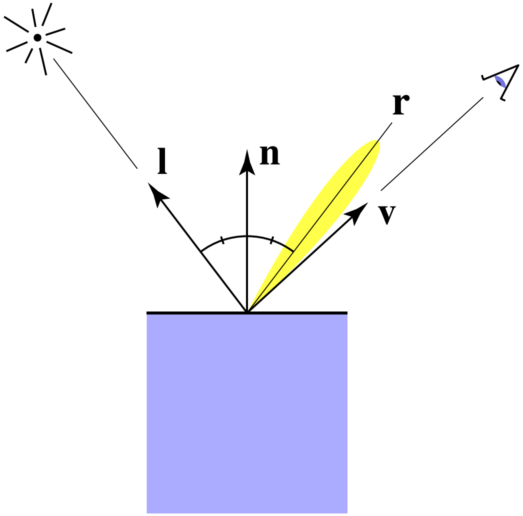
Recursion!?

```
traceray(ray, scene):
    t, rec = find_intersection(ray, scene)
    if rec.obj is a mirror:
      compute r, the reflection direction
      mirror_ray = Ray(rec.x, r)
      return traceray(mirror_ray, scene)
    # other cases, ...
```

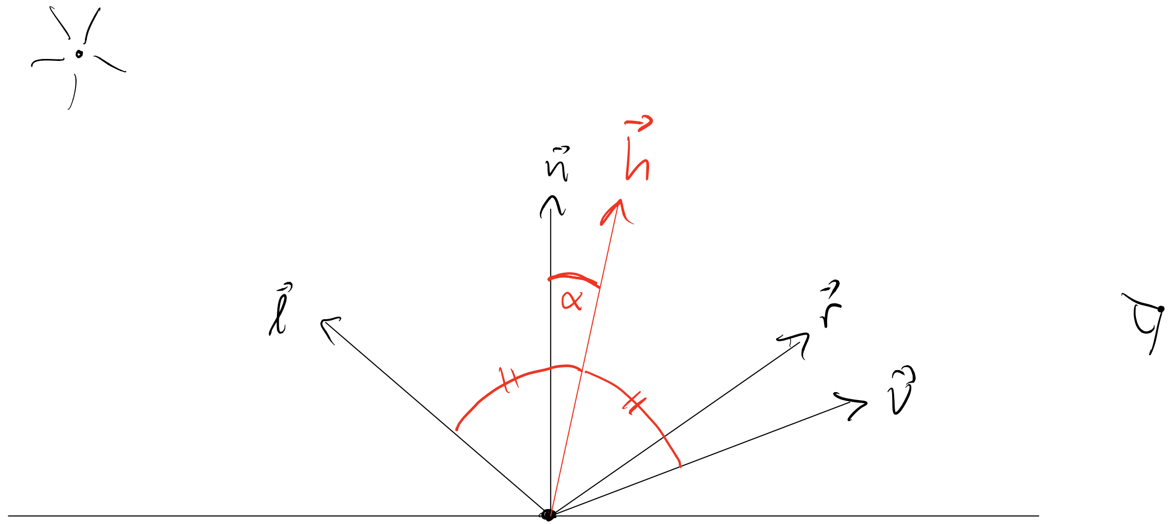
Specular Reflection

- What about non-mirror shiny surfaces?
- They appear brighter *near* "mirror" configuration
- Phong reflection: specular reflection is a function of angle between **r** and **v**.





Phong Reflection



Phong Reflection — Intuitive: $f(\vec{r}, \vec{v}) = f(\cos\alpha) = (\vec{r} \cdot \vec{v})^p$

Blinn-Phong Reflection — Alternative: $f(\vec{h}, \vec{n}) = \ldots (\vec{h} \cdot \vec{n})^p$

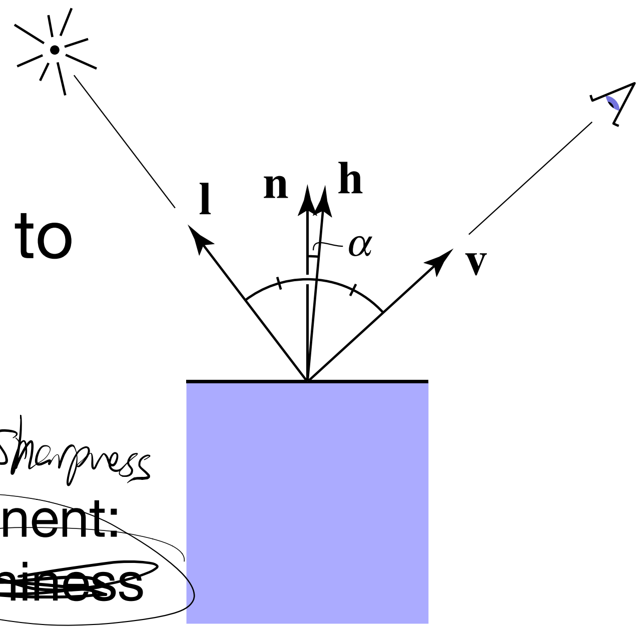
Specular Reflection

- Blinn-Phong: specular reflection is a function of angle between (**half-way vector** between view and light) and (the **normal**).
- h = bisector(**v**, **l**)
- Reflected light proportional to

$$k_s \max(0, \vec{n} \cdot \vec{h})^p$$

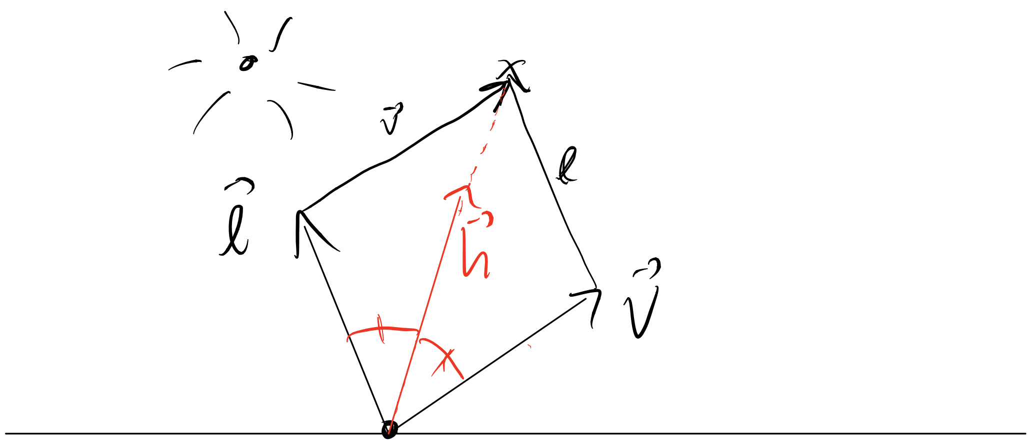
specular coefficient: determines strength of specularity term

specular exponent: determines shininess

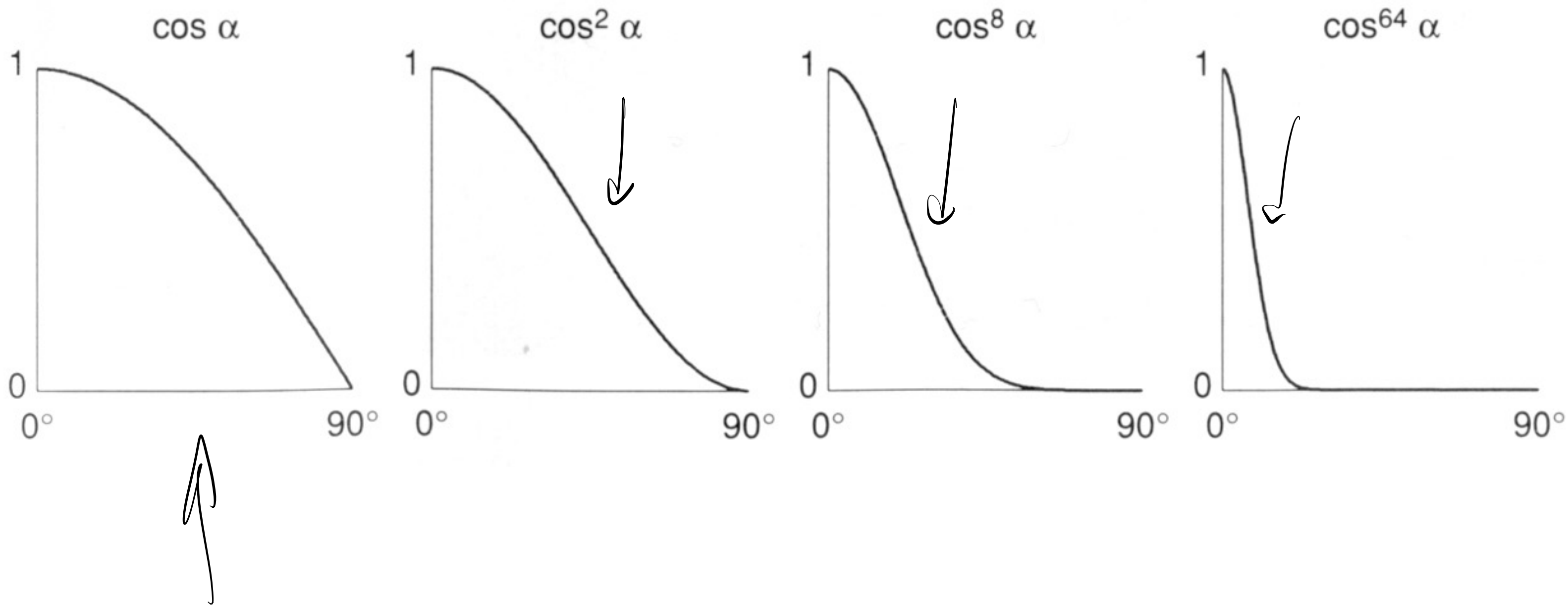


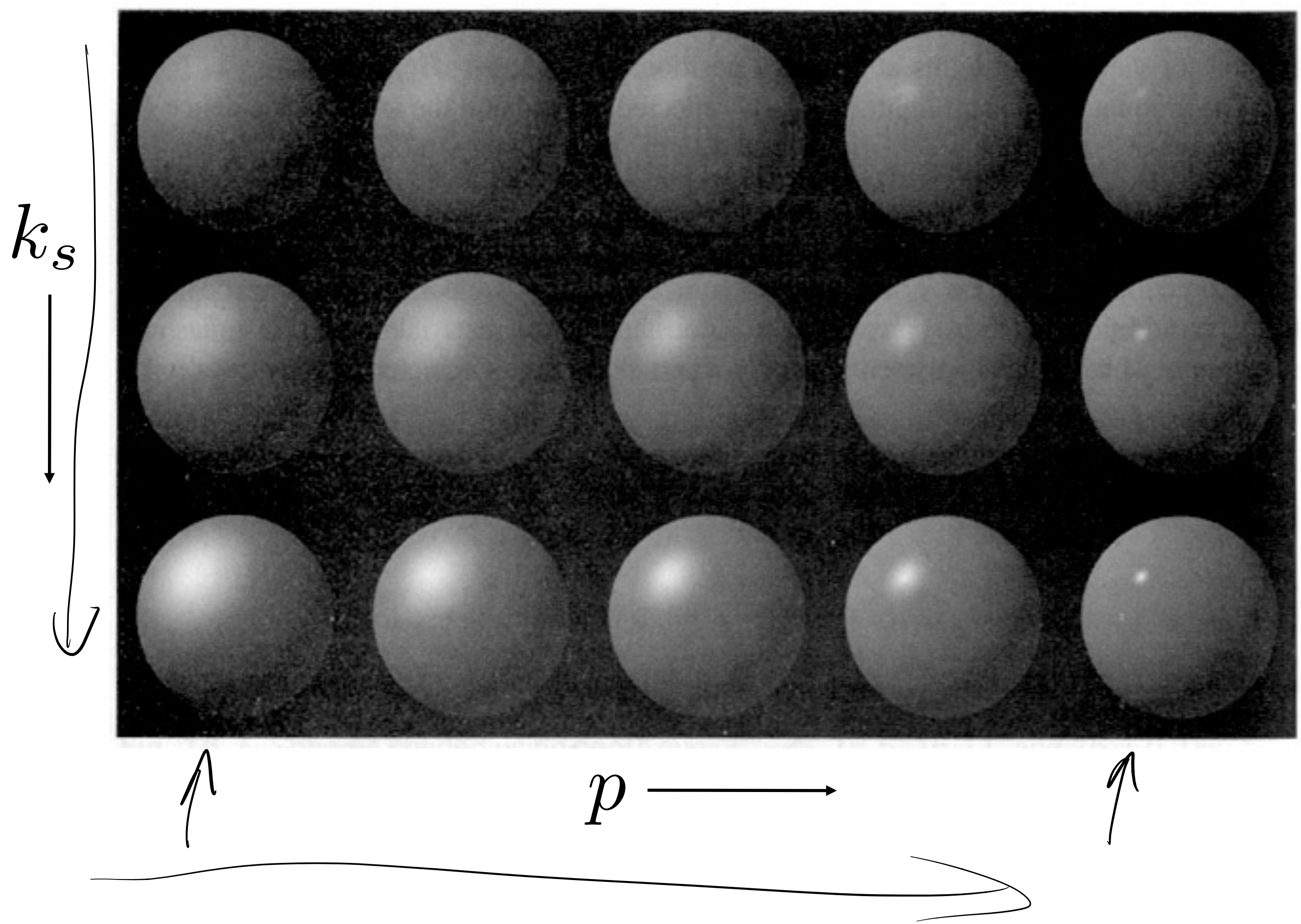
Computing h

$$\text{bisector}(\vec{v}, \vec{\ell}) = \frac{\vec{\ell} + \vec{v}}{\|\vec{\ell} + \vec{v}\|}$$



Effect of p





k_s
p

Putting it all Together: Blinn-Phong Reflection Model

Usually surfaces have both diffuse *and* specular components, so we'll combine the two:

diffuse reflection

specular reflection

light reflected

specular exponent (*sharpness* of specularity)

$$L = L_d + L_s$$

$$= k_d I \max(0, \vec{n} \cdot \vec{l}) + k_s I \max(0, \vec{n} \cdot \vec{h})^p$$

diffuse coefficient (surface brightness and color)

light intensity

normal

light direction

specular coefficient (*strength* [and color] of specularity)

half-vector between **l** and **v**

In code: `function shade_light(light, hitrec,...)`

What if there are multiple lights?

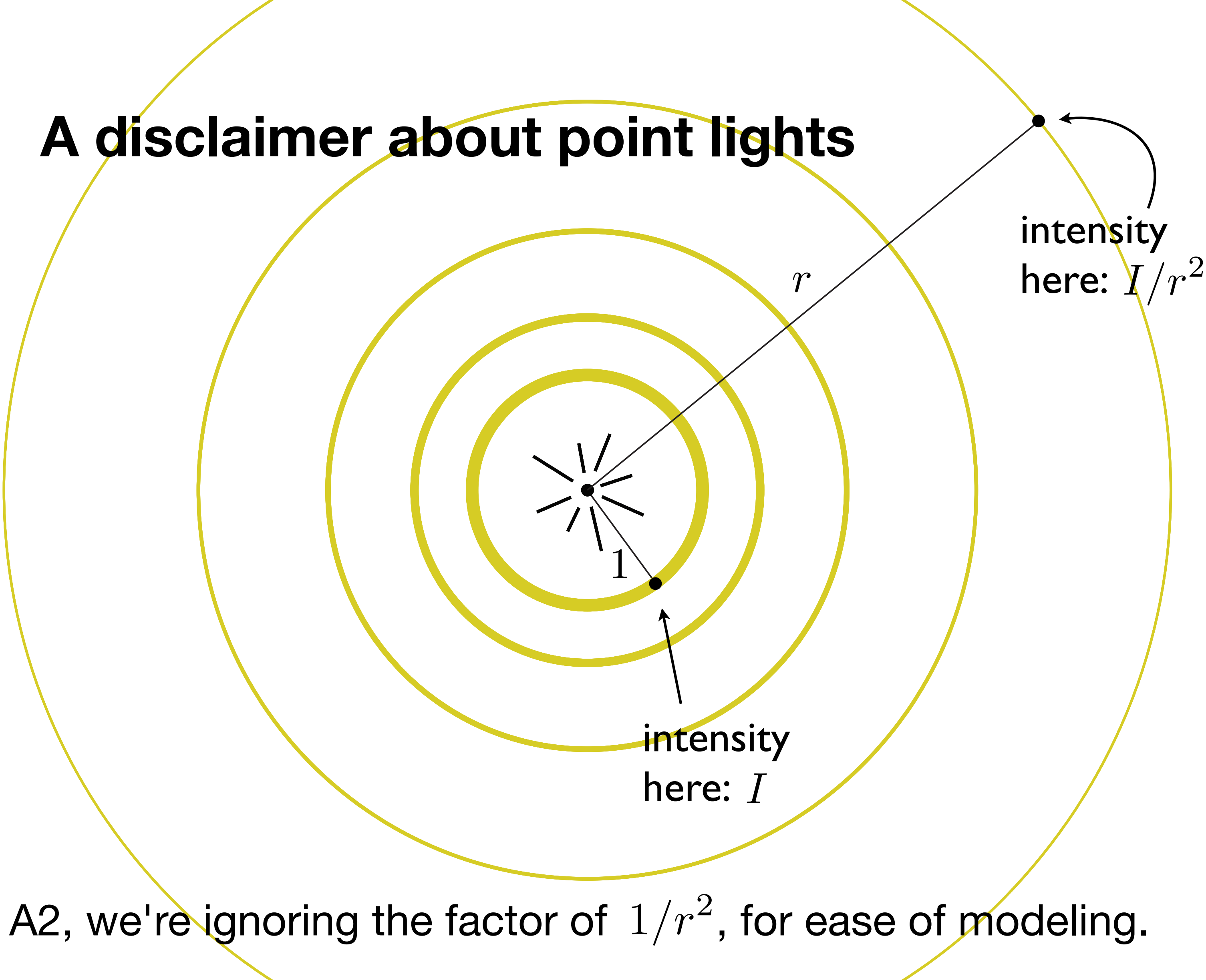
Light is additive - add them together:

$$L = \sum_{i=1}^{\#\text{ lights}} k_d I \max(0, \vec{n} \cdot \vec{l_i}) + k_s I \max(0, \vec{n} \cdot \vec{h_i})^p$$

In code:

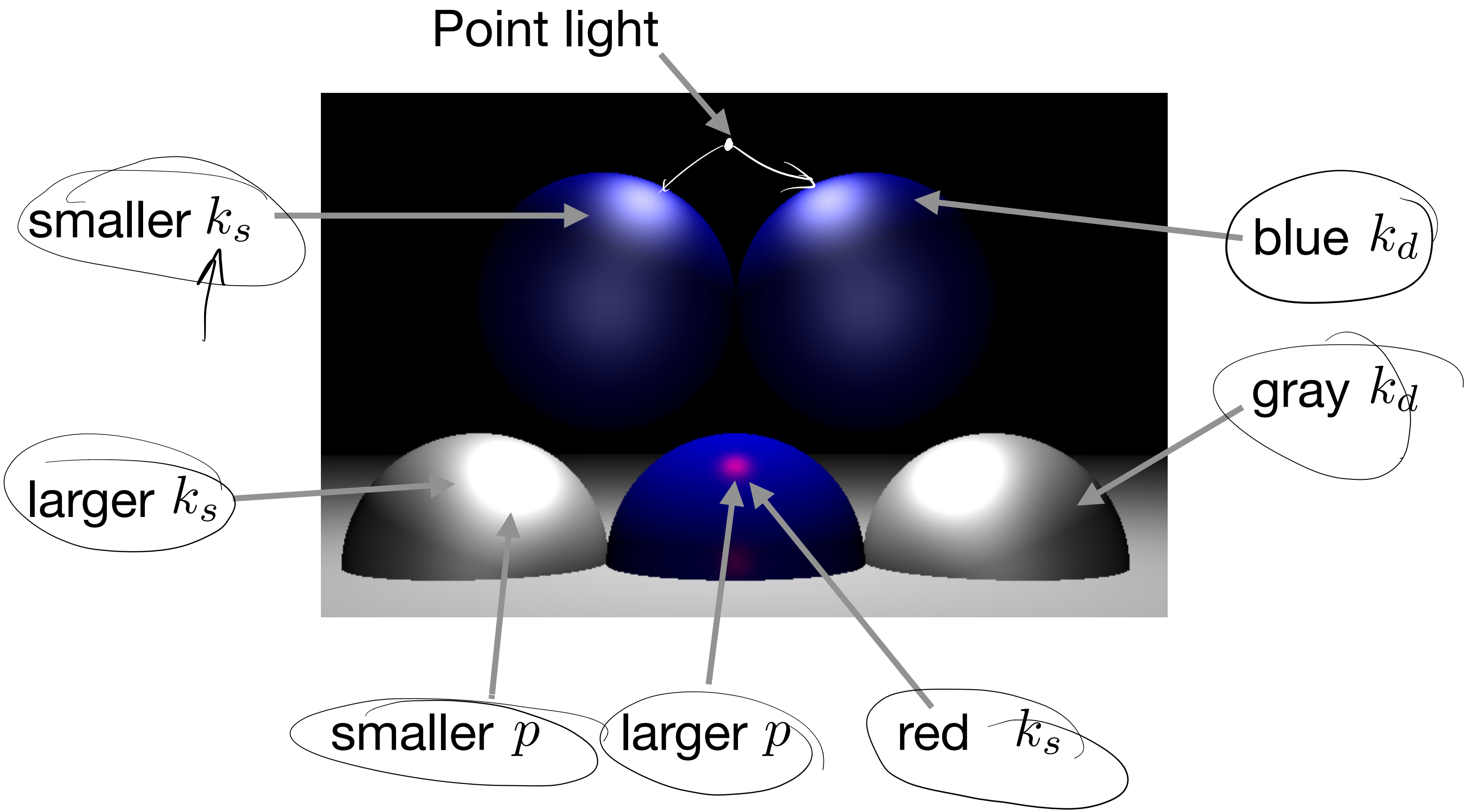
```
function determine_color(hitrec, ray, scene, ...):
    color = black
    for light in scene.lights:
        color += shade_light(light, hitrec, ...)
```

A disclaimer about point lights



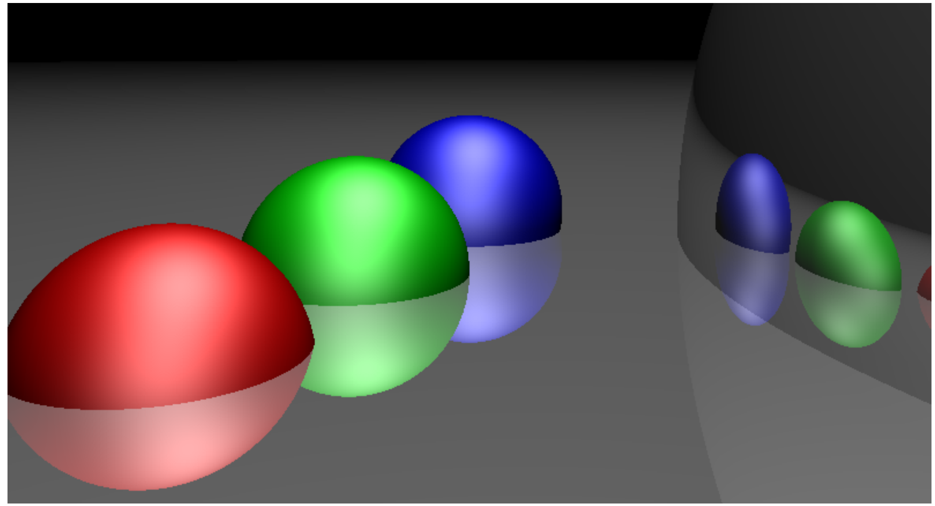
In A2, we're ignoring the factor of $1/r^2$, for ease of modeling.

Our images so far:

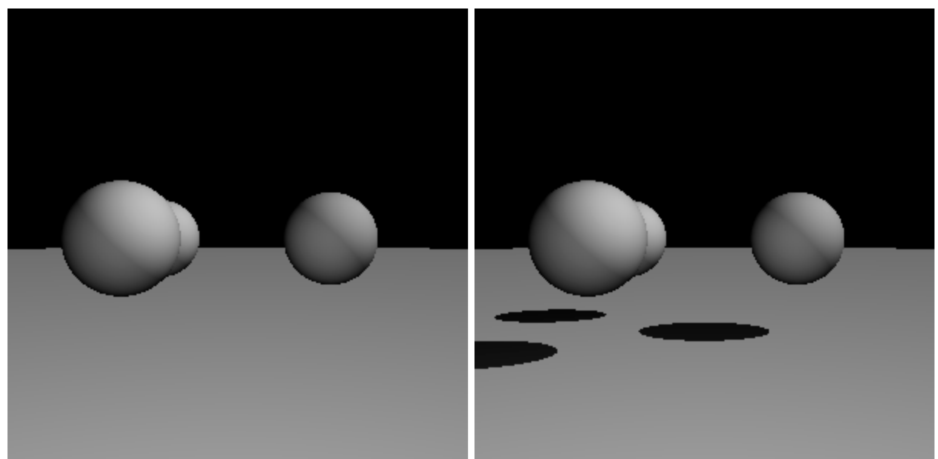


What's next?

Mirror-reflective surfaces

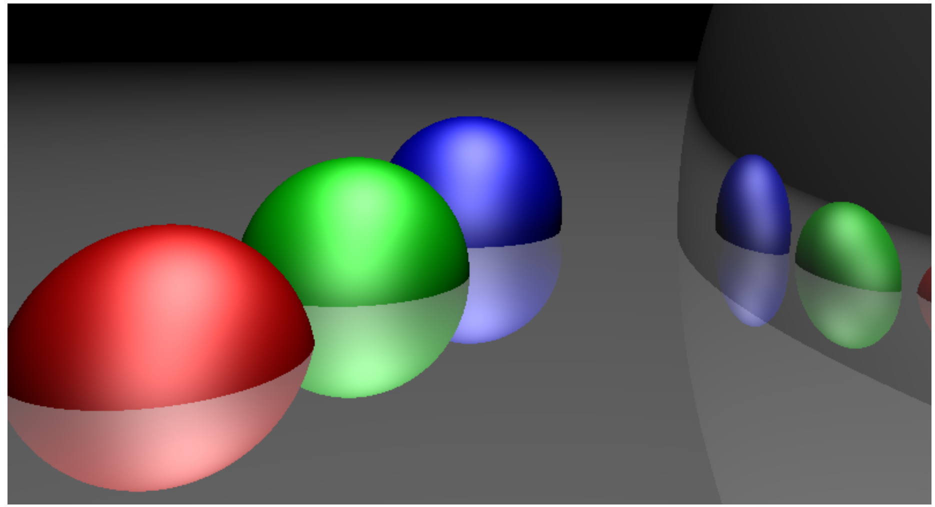


Shadows



Partially-Mirrored Surfaces

Notice the floor is gray but also mirror-reflective.



Materials store a **mirror coefficient**: fraction of light that is reflected in a mirror-like fashion

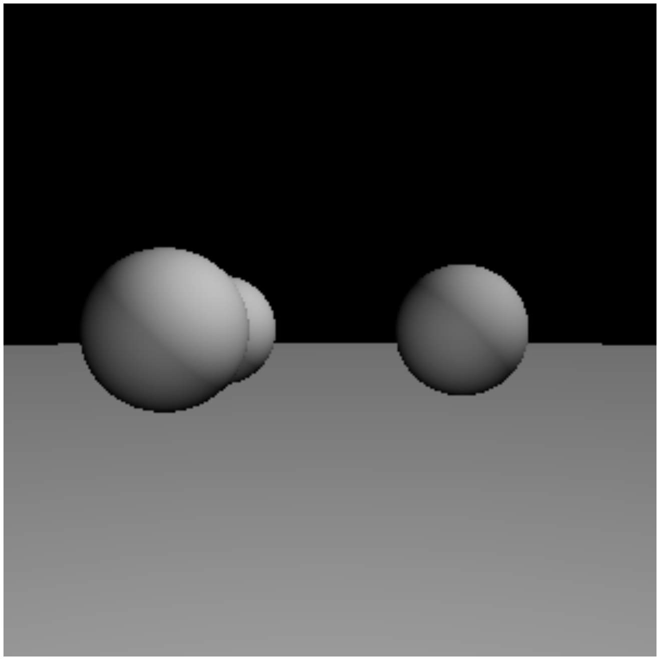
$$L = k_m L_r + (1 - k_m)(L_d + L_s)$$

mirror coefficient

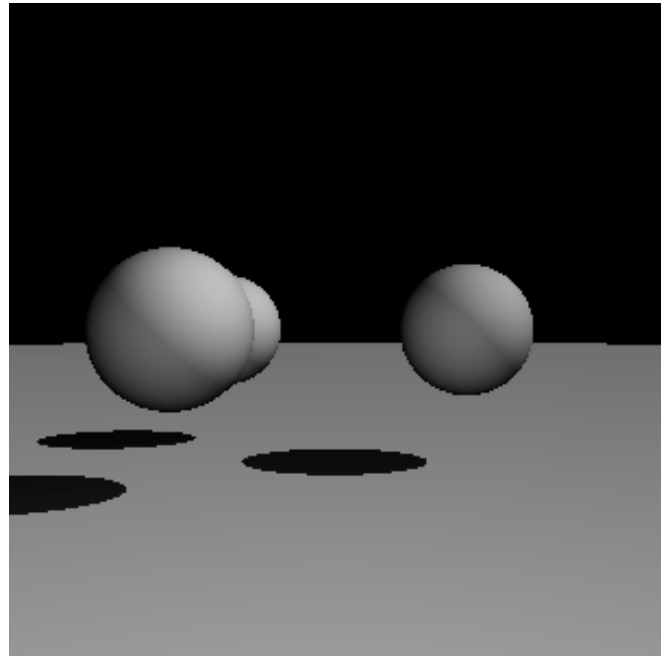
mirror-reflected light

"local" color (Blinn-Phong)

Shadows



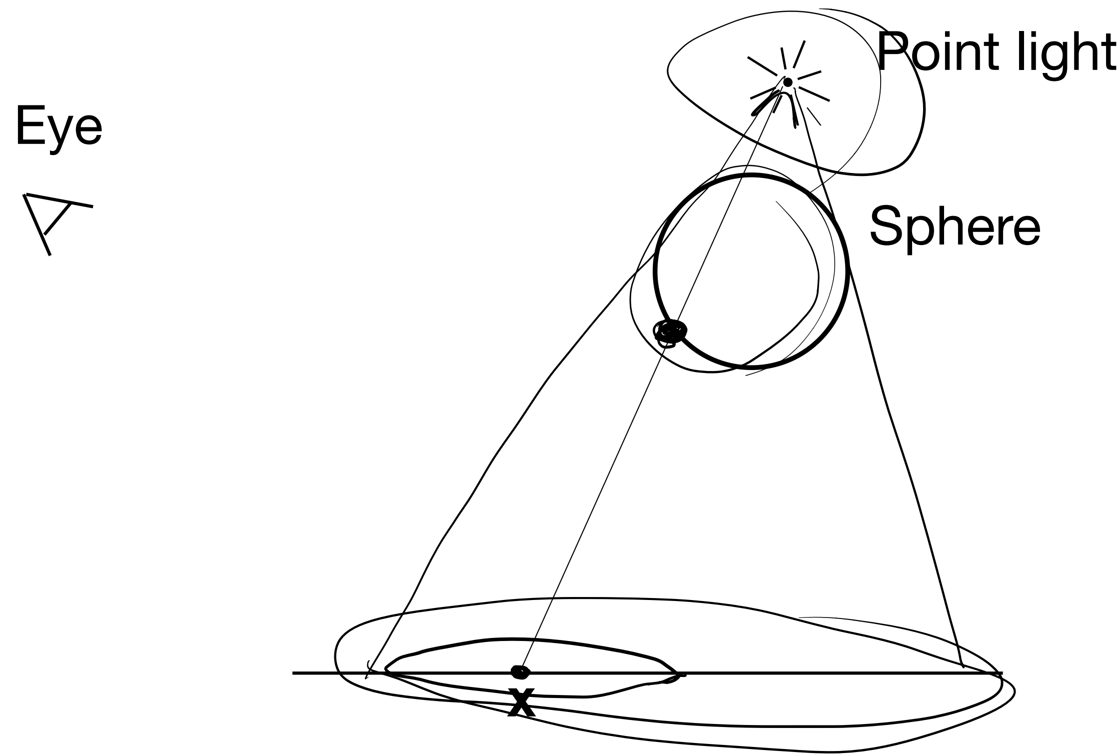
Wrong



Less Wrong

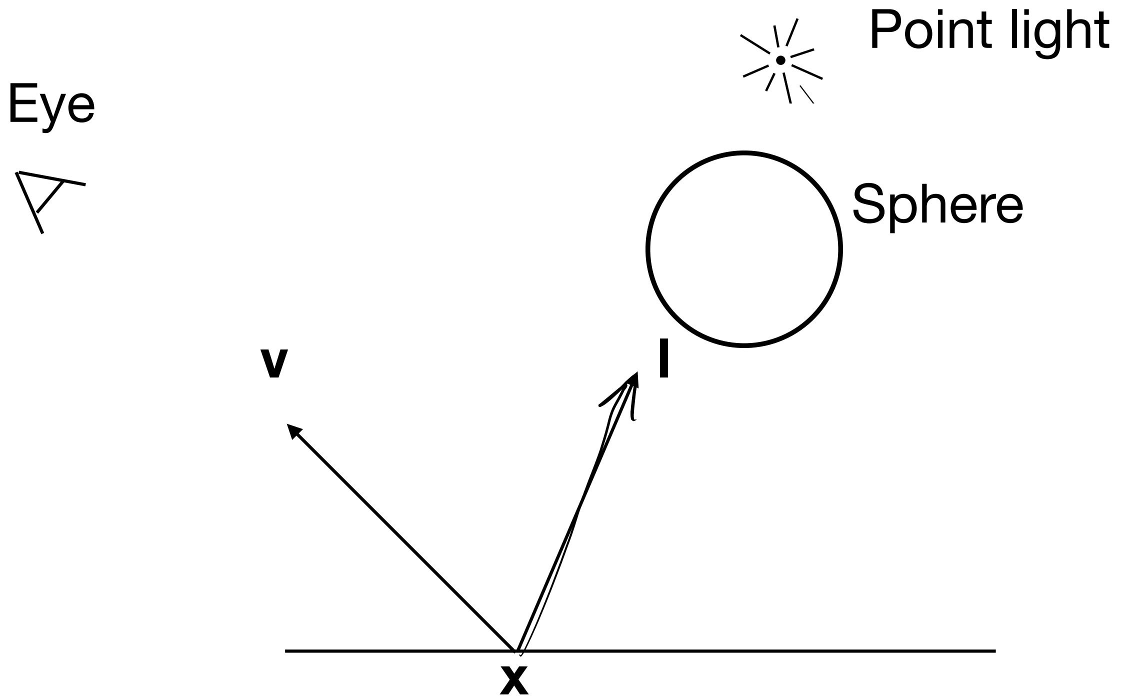
Shadows

How can we tell if a point is in shadow?



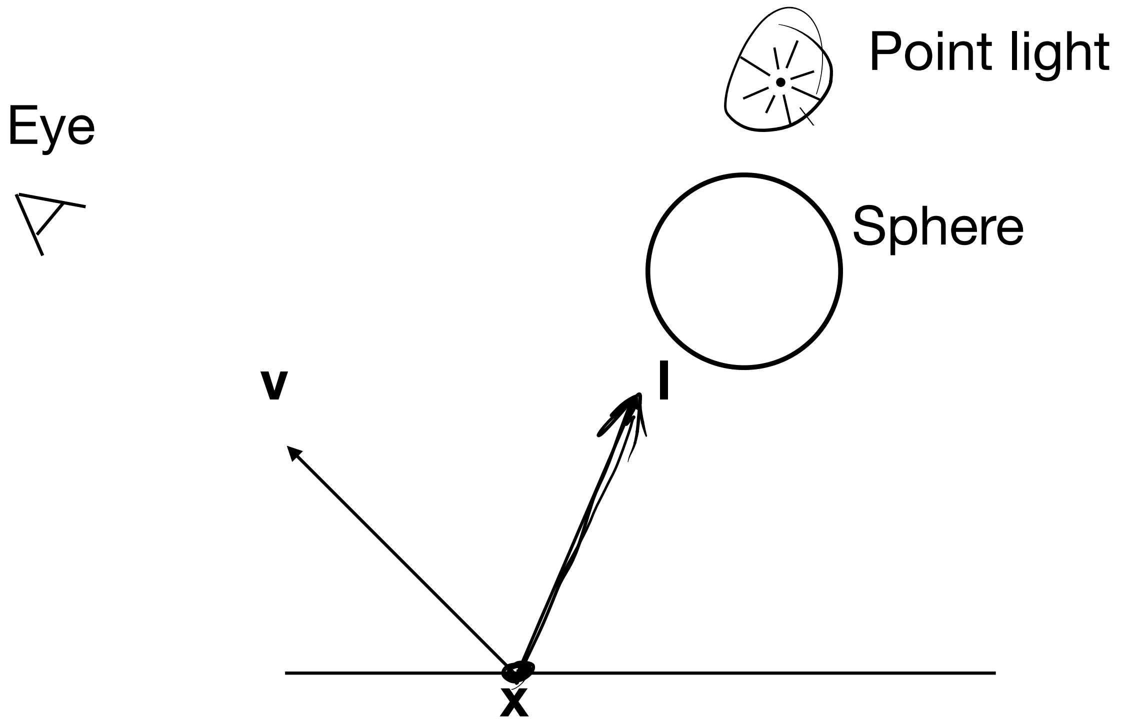
Shadows

How can we tell if a point is in shadow?



Shadows

How can we tell if a point is in shadow?



Point is shadowed iff:

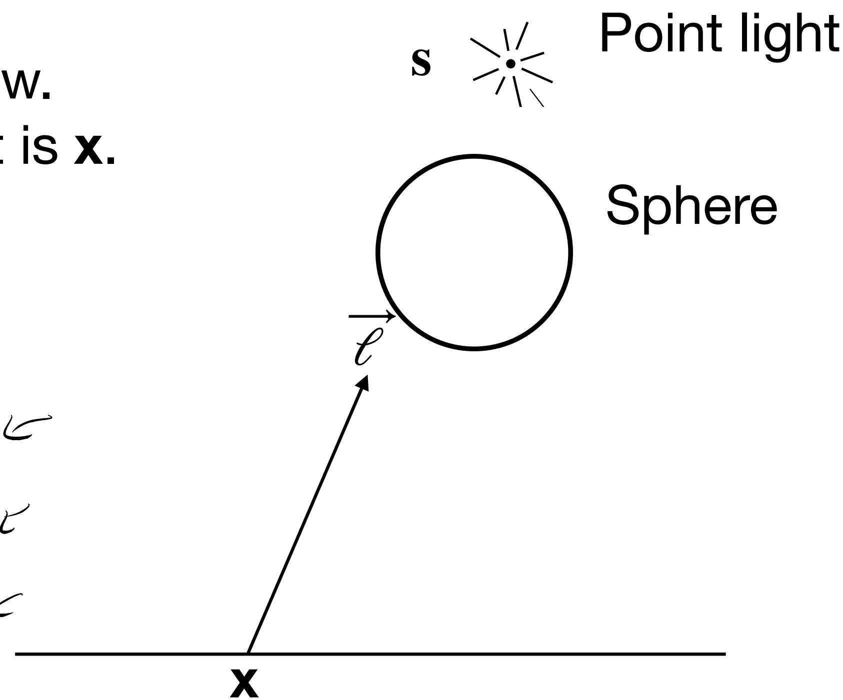
```
closest_intersect(objs, Ray(x, l), tmin, tmax) != nothing
```

Shadows

How can we tell if a point is in shadow?

Problem: Fill in the table below.
Assume the intersection point is **x**.

	Directional light $\vec{\ell}$	Point light **S**
r.orig	x	x
r.dir	$\vec{\ell}$	$\vec{S} - \vec{x}$
tmin	eps	eps
tmax	infinity	1



Point is shadowed iff:

```
closest_intersect(objs, Ray(orig, dir), tmin, tmax) != nothing
```

```
function determine_color(hitrec, ray, scene, ...):
    color = black
    for light in scene.lights:
        if !is_shadowed(scene, light, hitrec)
            color += shade_light(light, hitrec, ...)
```

Next time...
Let's talk about bunnies.

If we want bunnies, we still need to implement

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

Then, we can treat a triangle mesh as simply a list of triangles.