CMP205: Computer Graphics

Lecture 11: Ray Tracing II

Ahmed S. Kaseb
Fall 2018

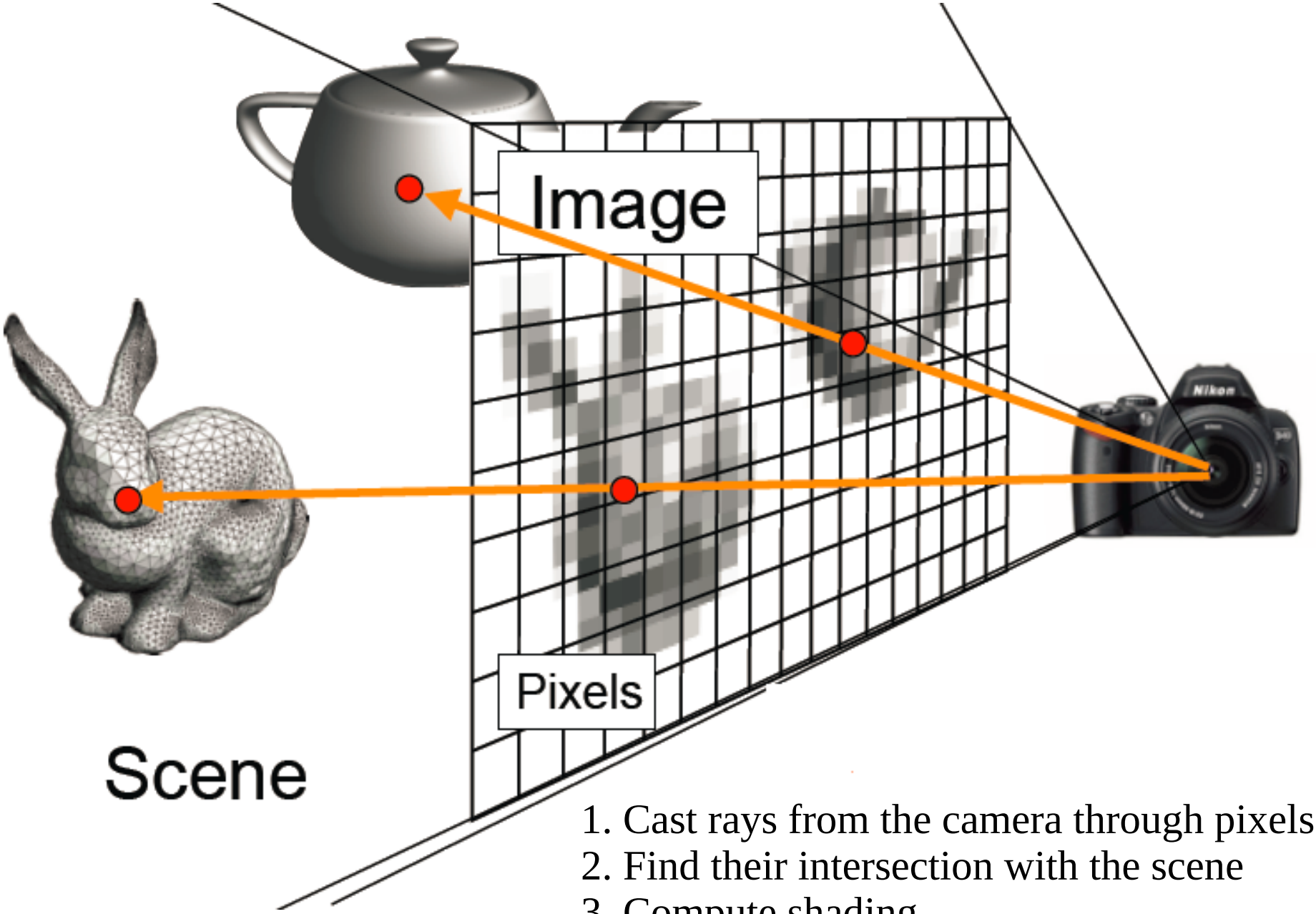
Slides by: Dr. Mohamed Alaa El-Dien Aly

Agenda

- Shadows
- Reflection
- Refraction
- Distribution Ray Tracing
- Instancing

Acknowledgment: Some slides adapted from Steve Marschner, Maneesh Agrawala, and Fredo Durand

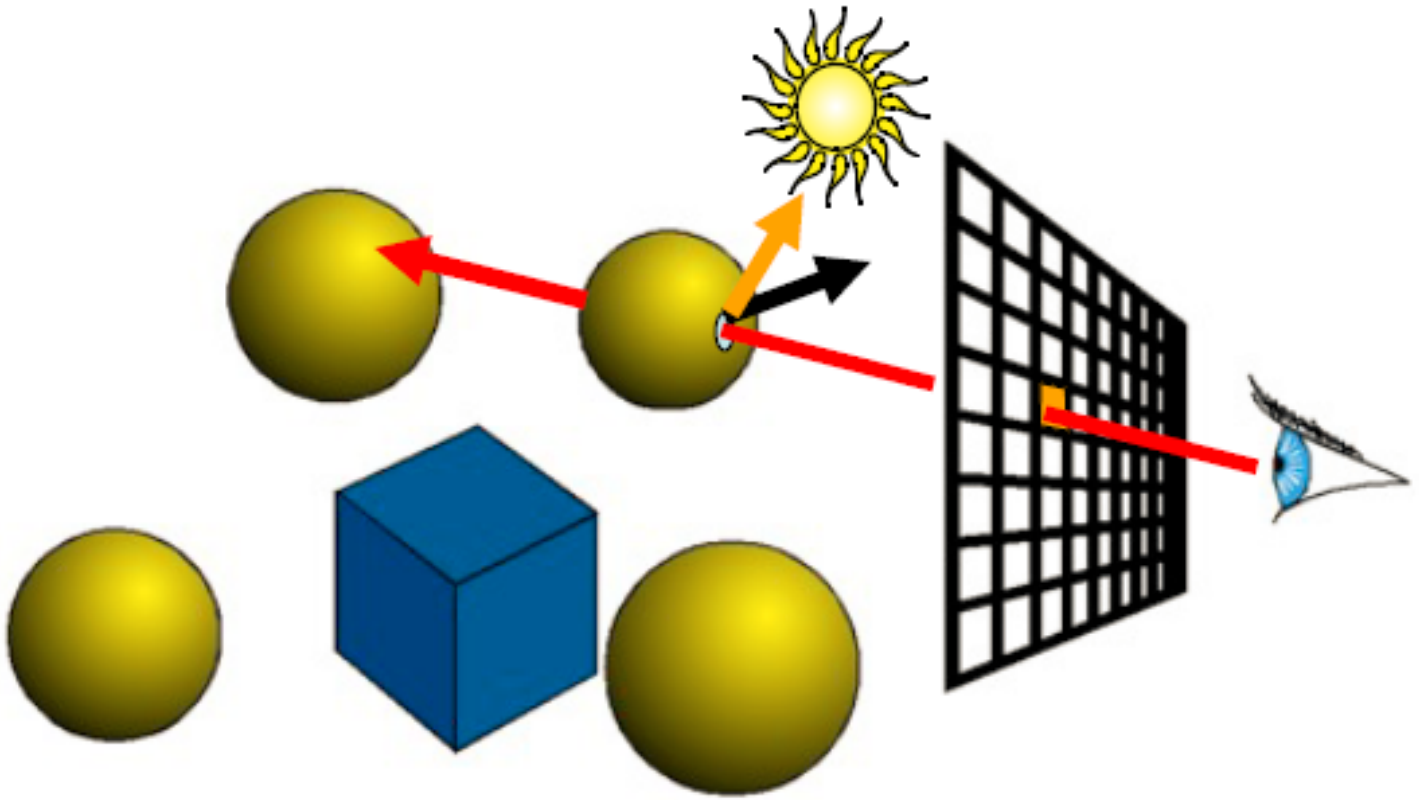
Review: Ray Tracing



1. Cast rays from the camera through pixels
2. Find their intersection with the scene
3. Compute shading

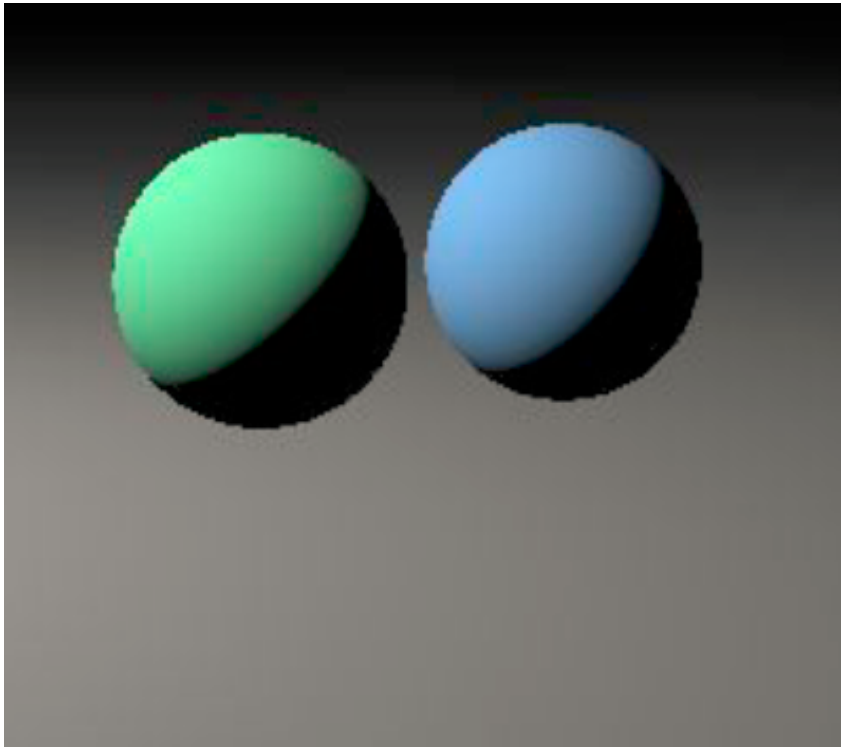
Review: Ray Tracing

```
For each pixel
  Construct a ray from the eye
  For each object in the scene
    Find intersection point (and surface normal)
    Keep if closest
    Compute Shading
```

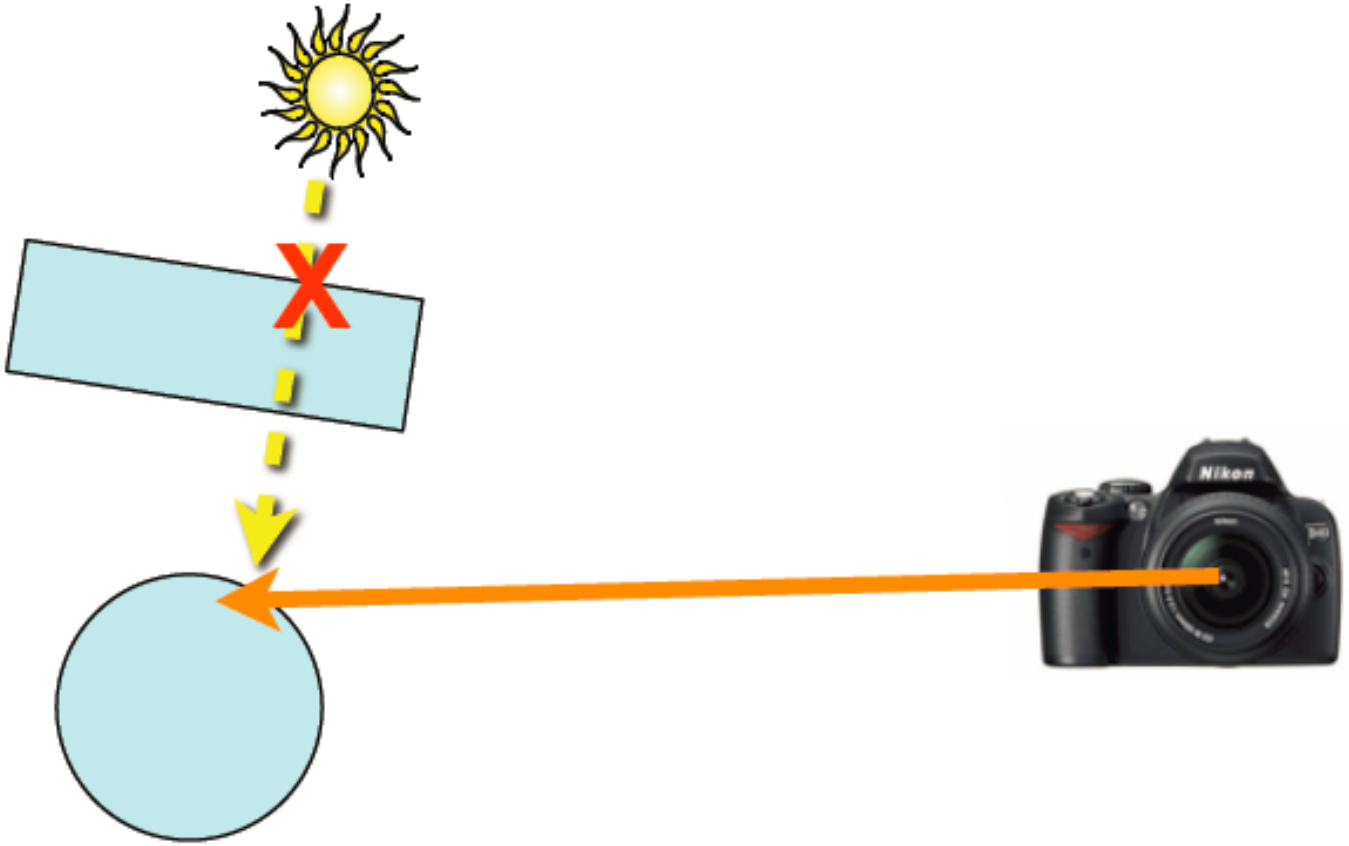


Review: Ray Tracing Program

```
function ComputeShading(ray, t0, t1)
  Get intersection of ray with scene
  If intersection != null
    Color = ambient
    Get n, h, l
    Color += kd * max(0, <n,l>) + ks * <h, n>^p
  Else
    Color = background
```

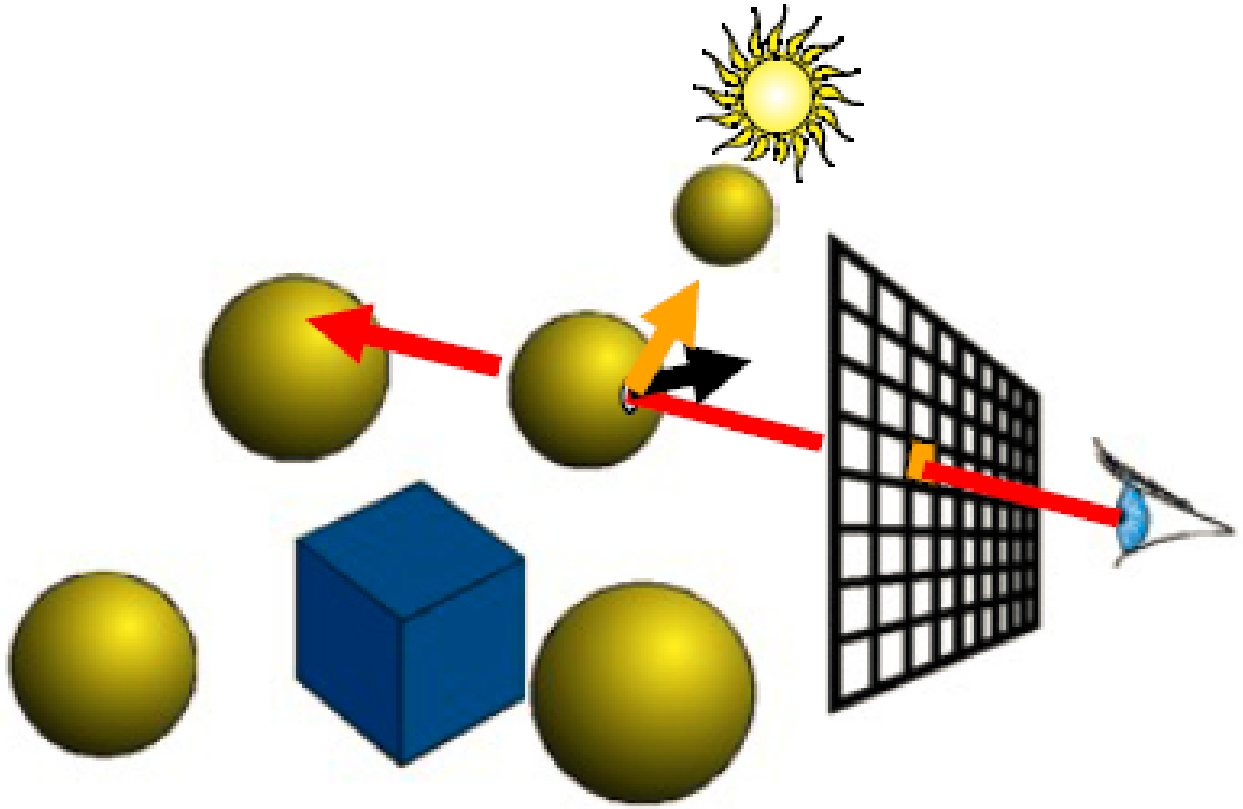


Shadows



Point in shadow if ray from light is blocked !

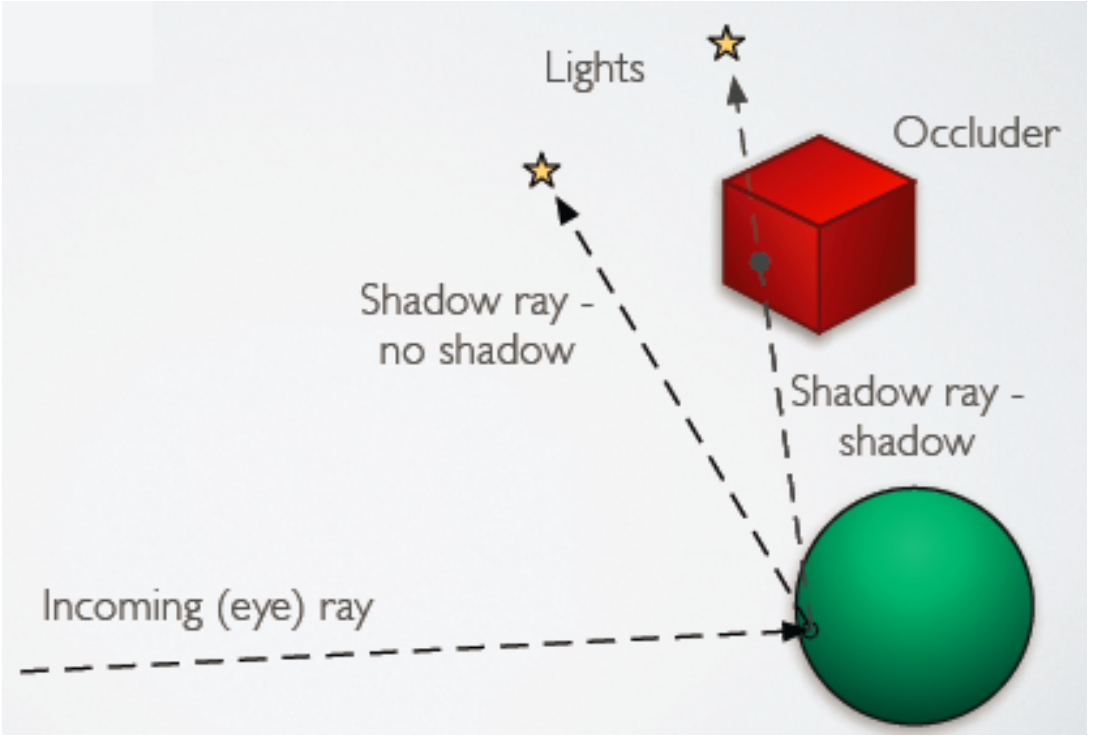
Shadows



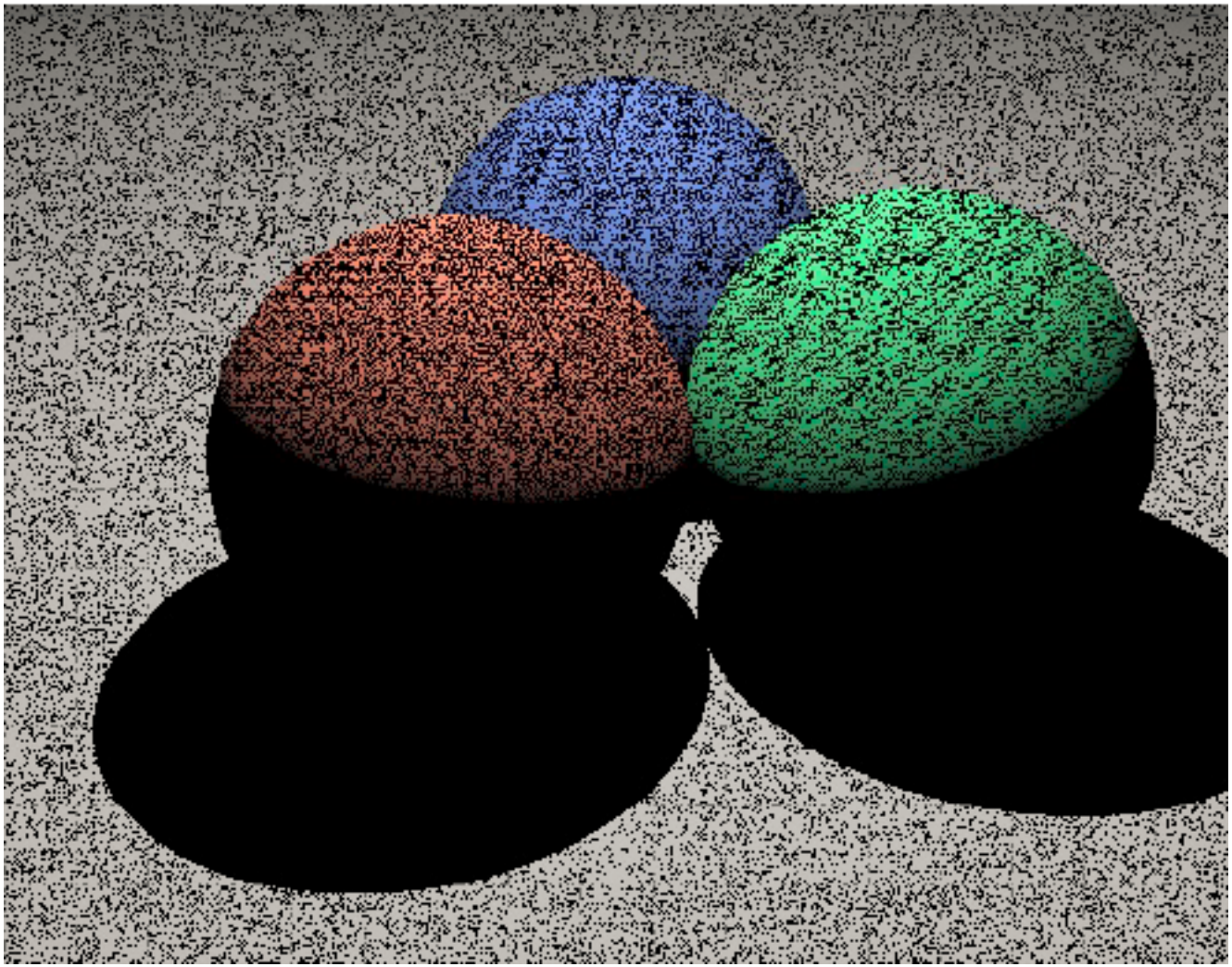
Idea: Send out a “shadow” ray from every surface point hit to check if it *sees* the light source or no

Shadows

```
function ComputeShading(ray, t0, t1)
  Get intersection of ray with scene
  If intersection != null
    Color = ambient
    If NOT intersection(shadowray, 0, ∞)
      Get n, h, l
      Color += kd * max(0, <n,l>) + ks * <h, n>^p
  Else
    Color = background
```



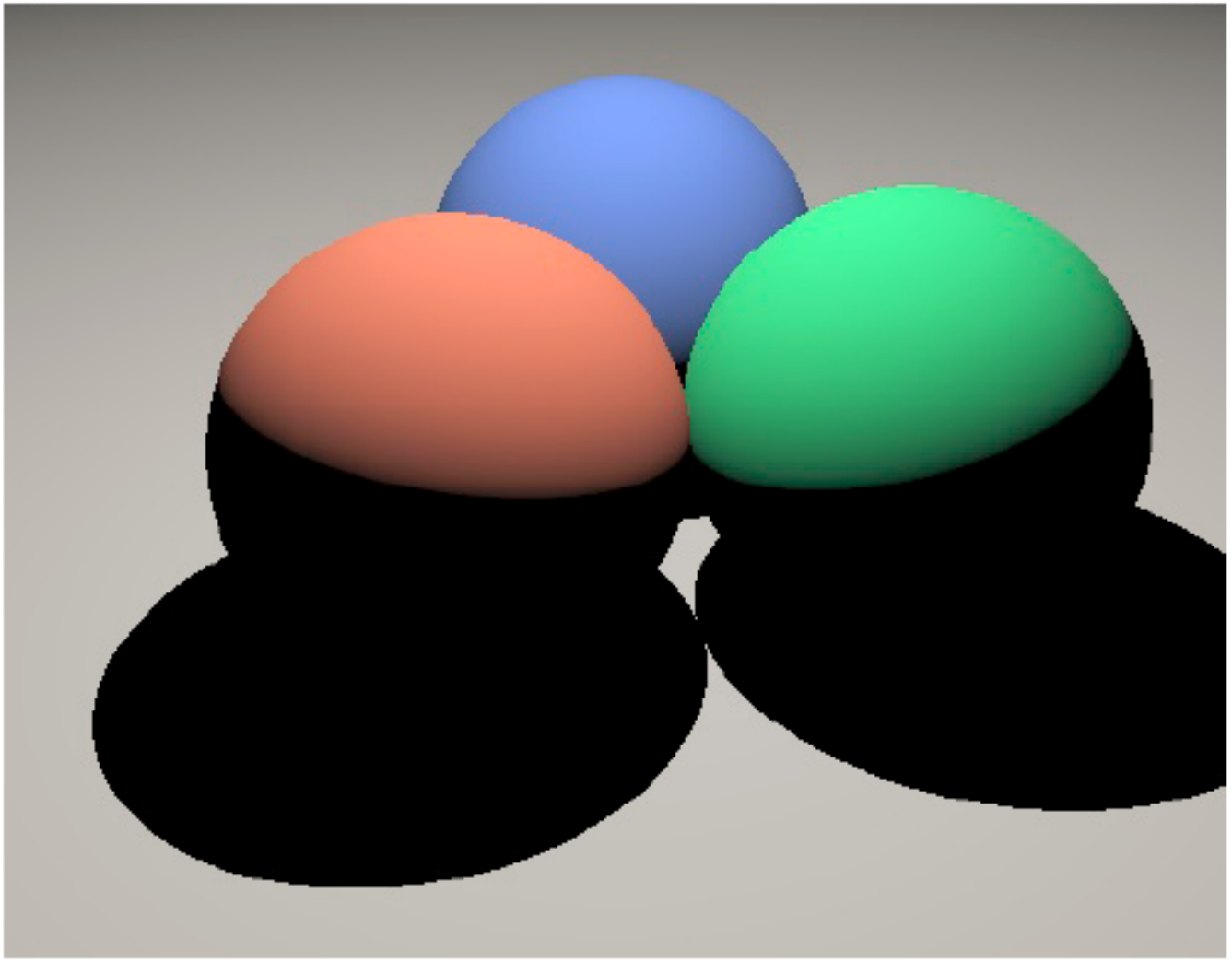
Shadows



Problem !!

Shadow ray intersects object at $t = 0$!

Shadows

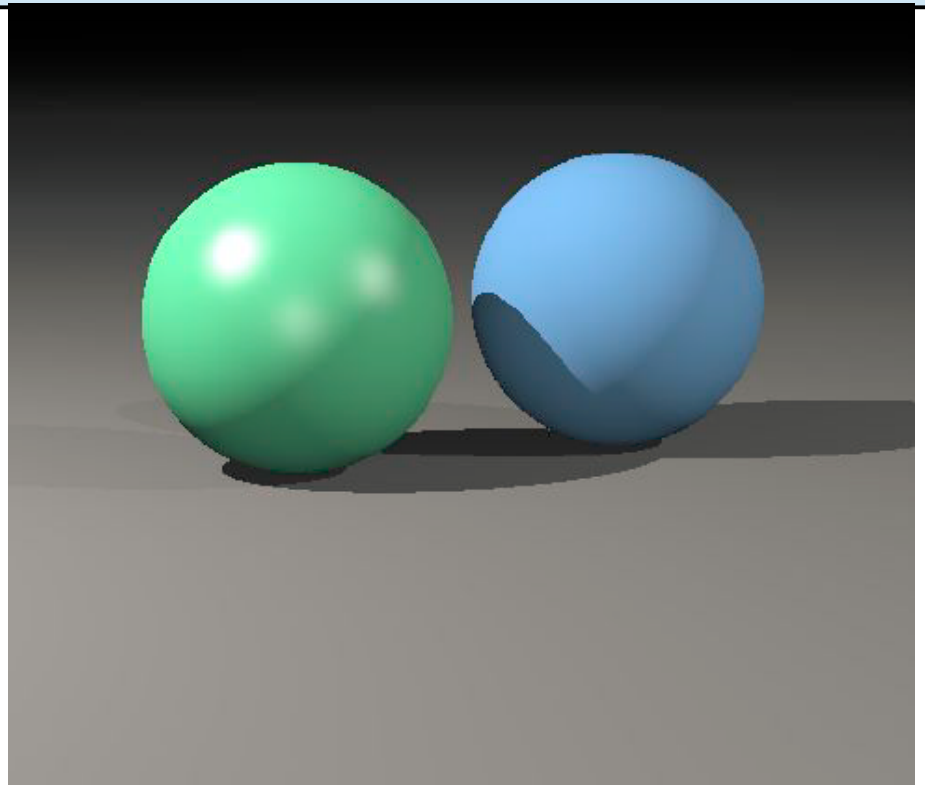


Solution?

Shadow ray starts a bit away from the surface i.e. intersection: $t \geq \varepsilon$

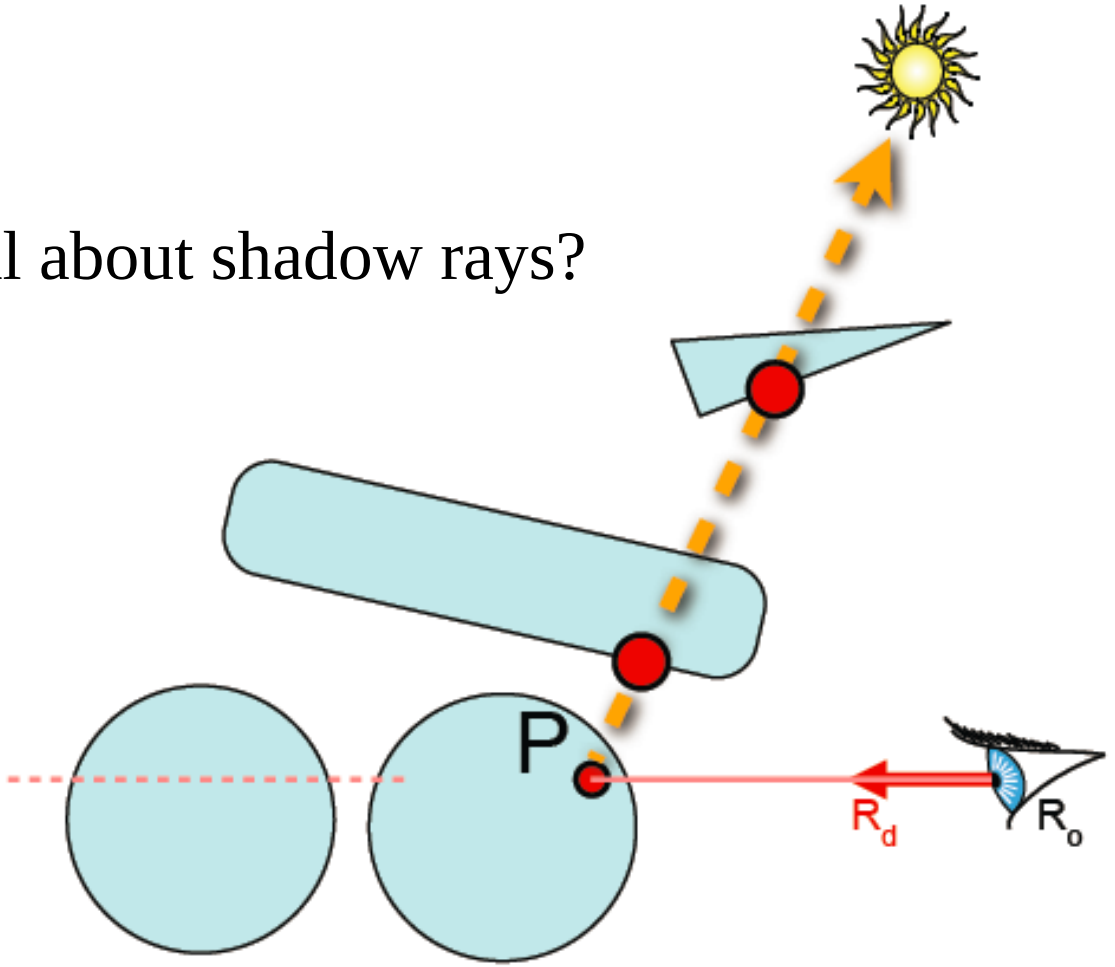
Shadows

```
function ComputeShading(ray, t0, t1)
  Get intersection of ray with scene
  If intersection != null
    Color = ambient
    Get n, h, l
    If !blocked(shadowray, ε, ∞)
      Color += kd * max(0, <n,l>) + ks * <h, n>^p
  Else
    Color = background
```



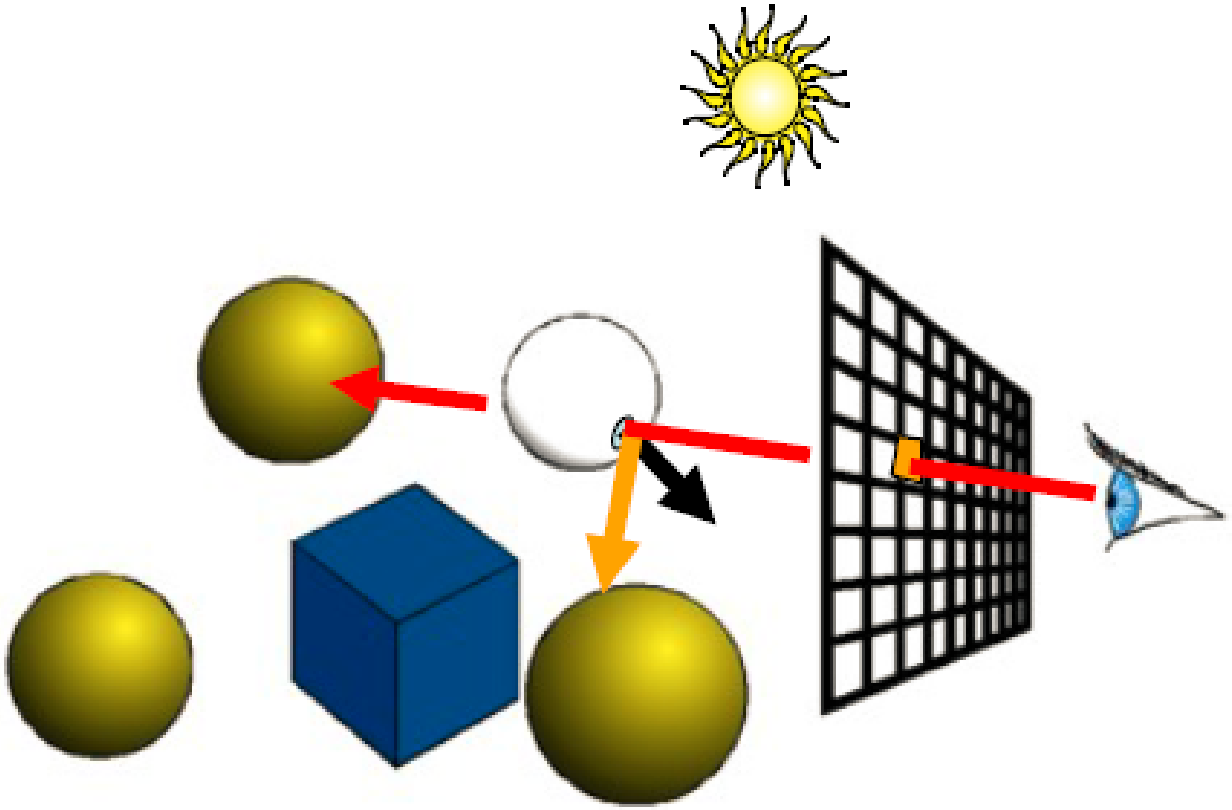
Shadows

What's special about shadow rays?



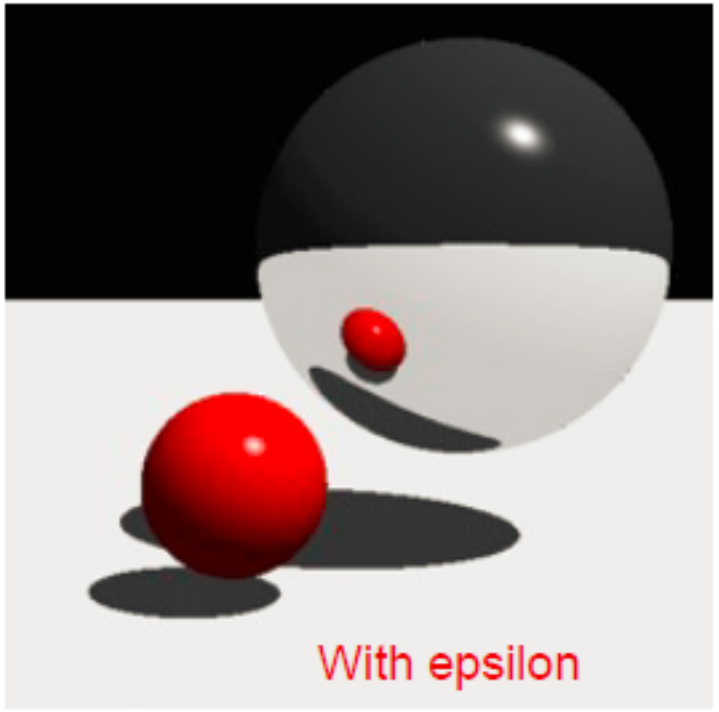
Only need to find *any* intersection NOT the closest one!
Stop once we found the first intersection

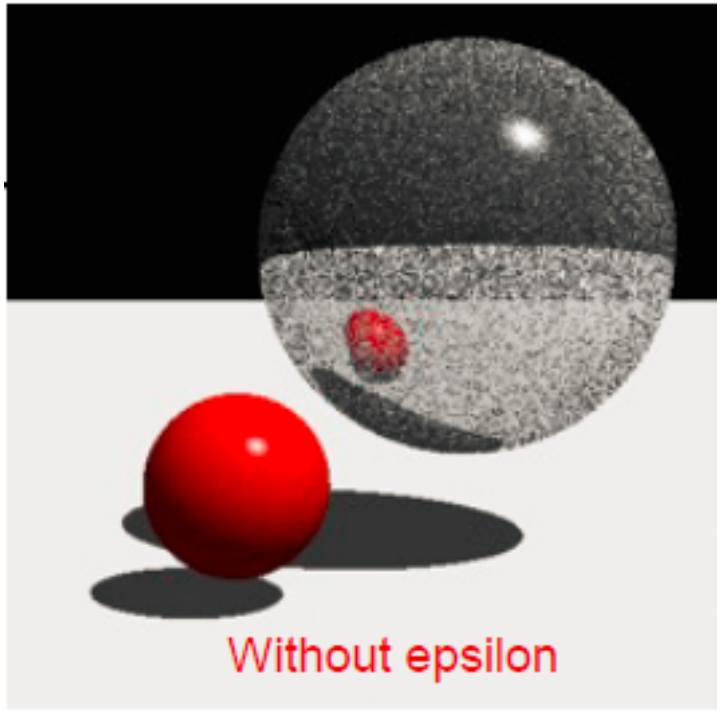
Reflection



Send out a “reflection” ray from the surface point and multiply by specular (or reflection) coefficient k_m

Reflection

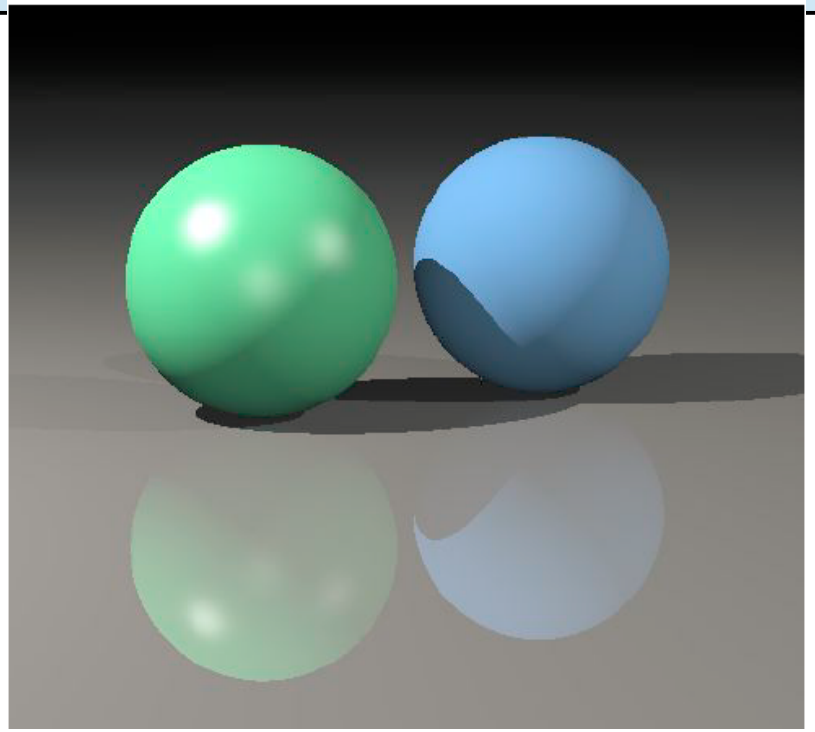




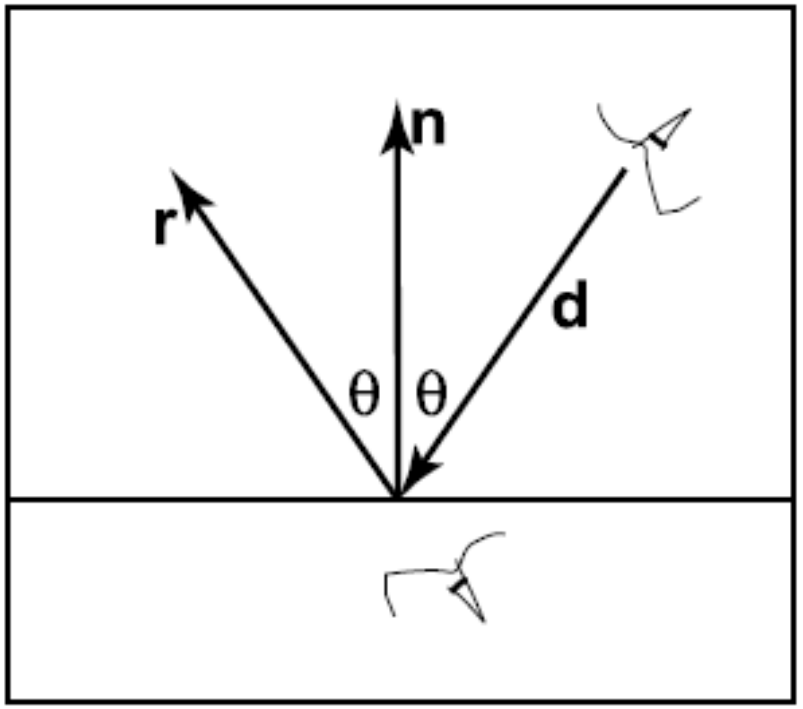
Don't forget ε

Reflection

```
function ComputeShading(ray, t0, t1)
  Get intersection of ray with scene
  If intersection != null
    Color = ambient
    Get n, h, l, r
    If !blocked(shadowray, ε, ∞)
      Color += kd * max(0, <n,l>) + ks * <h, n>^p
         + km * ComputeShading(reflected ray, ε, ∞)
  Else
    Color = background
```

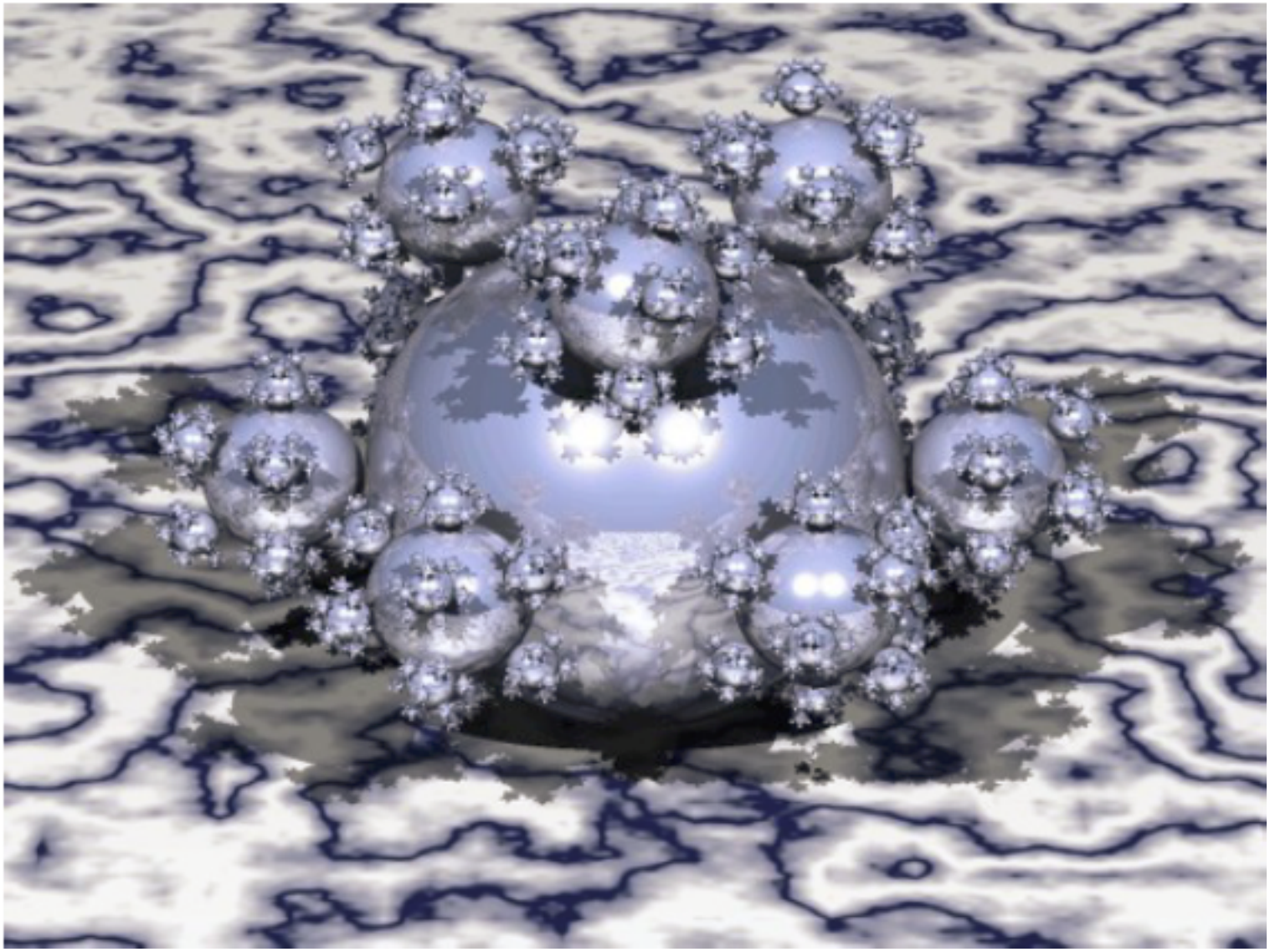


Reflection

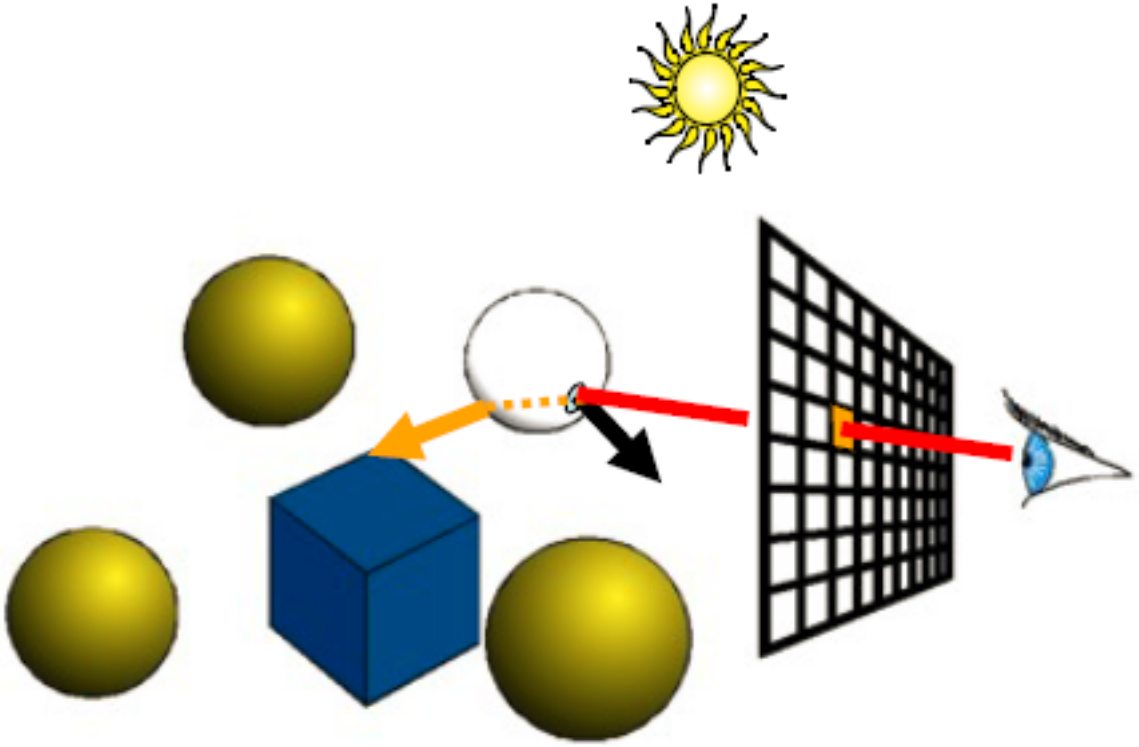


$$\boldsymbol{r} = \boldsymbol{d} - 2\left(\boldsymbol{d}\cdot\boldsymbol{n}\right)\boldsymbol{n}$$

Reflection

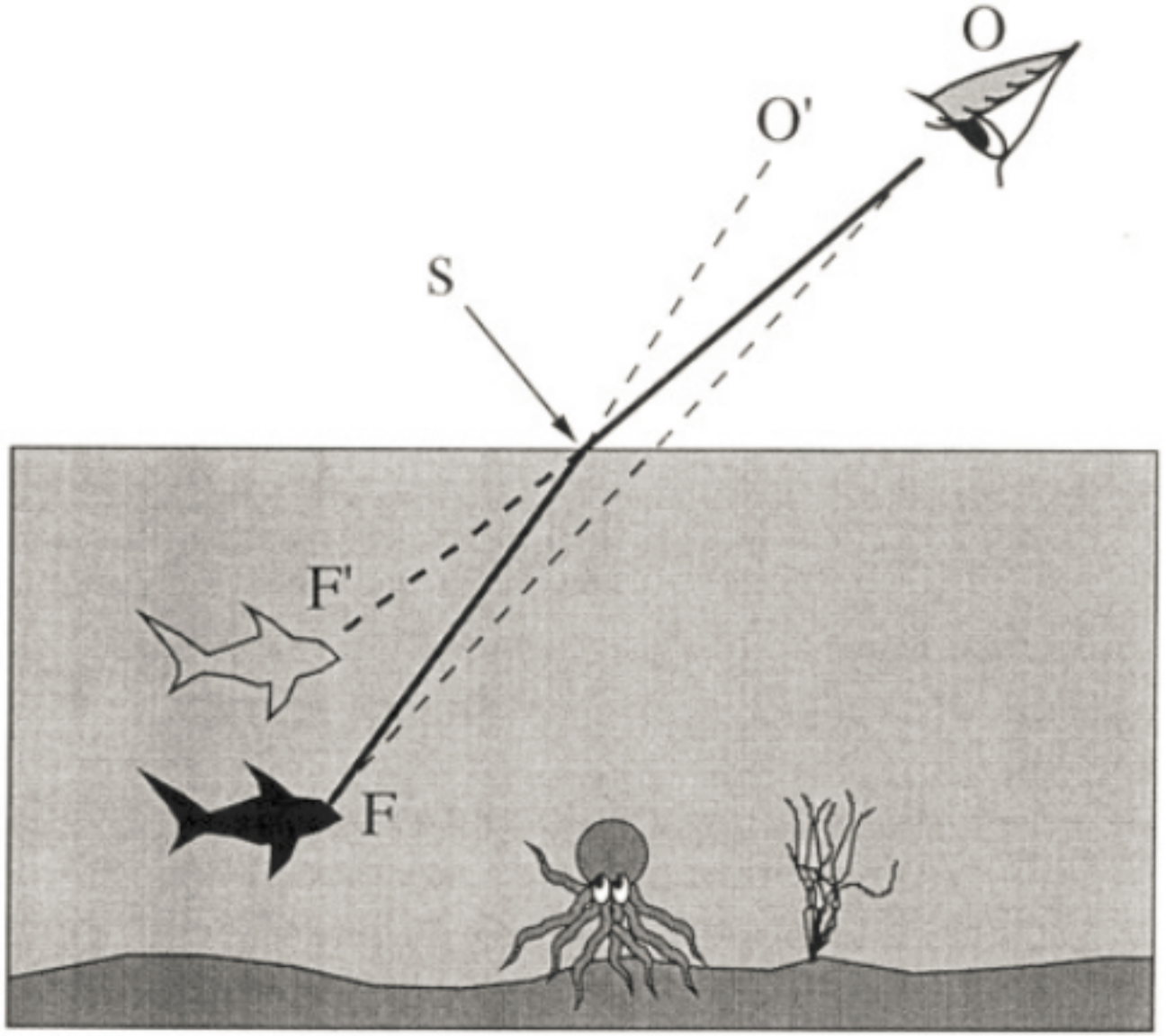


Refraction



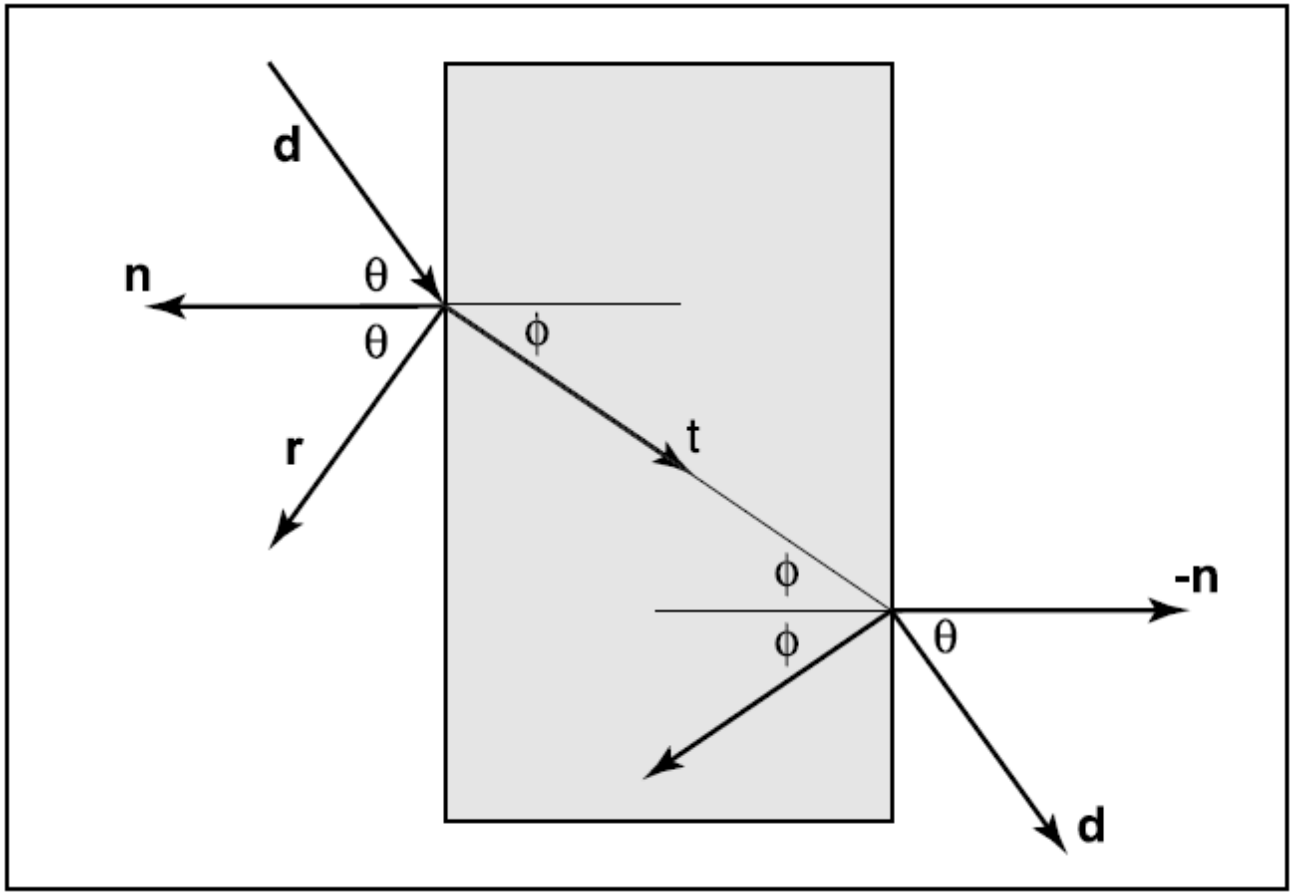
Send out a “refraction” ray from the surface point and multiply by the transparency coefficient k_t

Refraction



Light *bends* when crossing interface

Refraction



η_i & η_t indices of refraction

Snell's Law $\eta_i \sin\theta = \eta_t \sin\phi$

$$\sin\phi = \frac{\eta_i \sin\theta}{\eta_t} \qquad \sin\phi = \frac{\eta_i \sqrt{1-\cos^2\theta}}{\eta_t}$$

Refraction

What's $\boldsymbol{t}$ in terms of $\boldsymbol{b}$ and $\boldsymbol{n}$?

$$\boldsymbol{t} = \sin\phi\,\boldsymbol{b} - \cos\phi\,\boldsymbol{n}$$

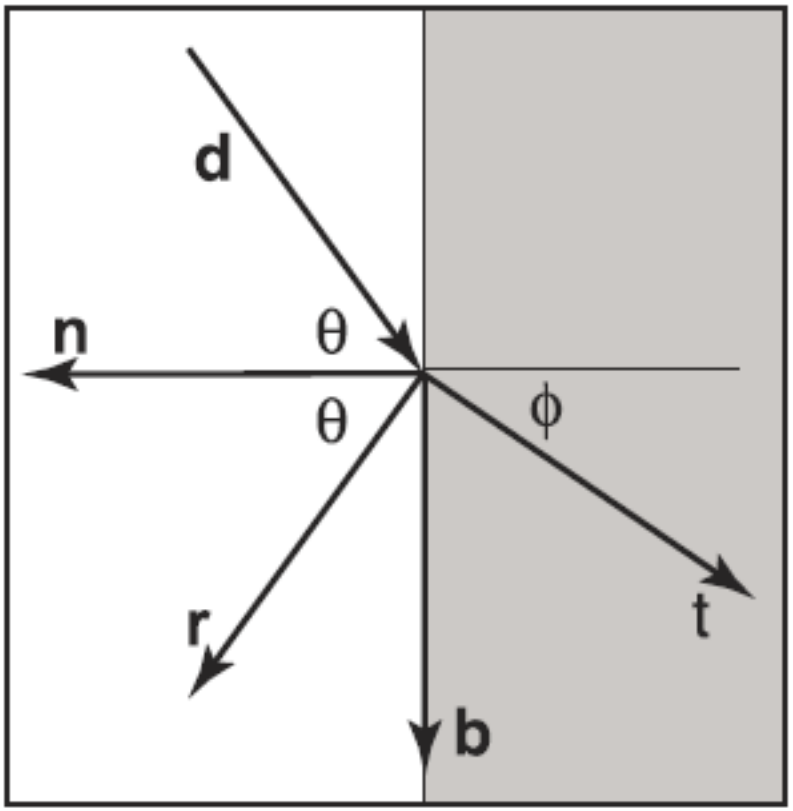
We know $\boldsymbol{n}$ and ϕ, but what's $\boldsymbol{b}$?

$$\boldsymbol{d} = \sin\theta\,\boldsymbol{b} - \cos\theta\,\boldsymbol{n}$$

$$\boldsymbol{b} = \frac{\boldsymbol{d} + \cos\theta\,\boldsymbol{n}}{\sin\theta}$$

$$\boldsymbol{t} = \sin\phi\,\frac{\boldsymbol{d} + \cos\theta\,\boldsymbol{n}}{\sin\theta} - \cos\phi\,\boldsymbol{n}$$

$$\boldsymbol{t} = \frac{\eta_i\left(\boldsymbol{d} - (\boldsymbol{n}\cdot\boldsymbol{d})\,\boldsymbol{n}\right)}{\eta_t} - \sqrt{1 - \frac{\eta_i^2\left(1 - (\boldsymbol{d}\cdot\boldsymbol{n})^2\right)}{\eta_t^2}}\,\boldsymbol{n}$$

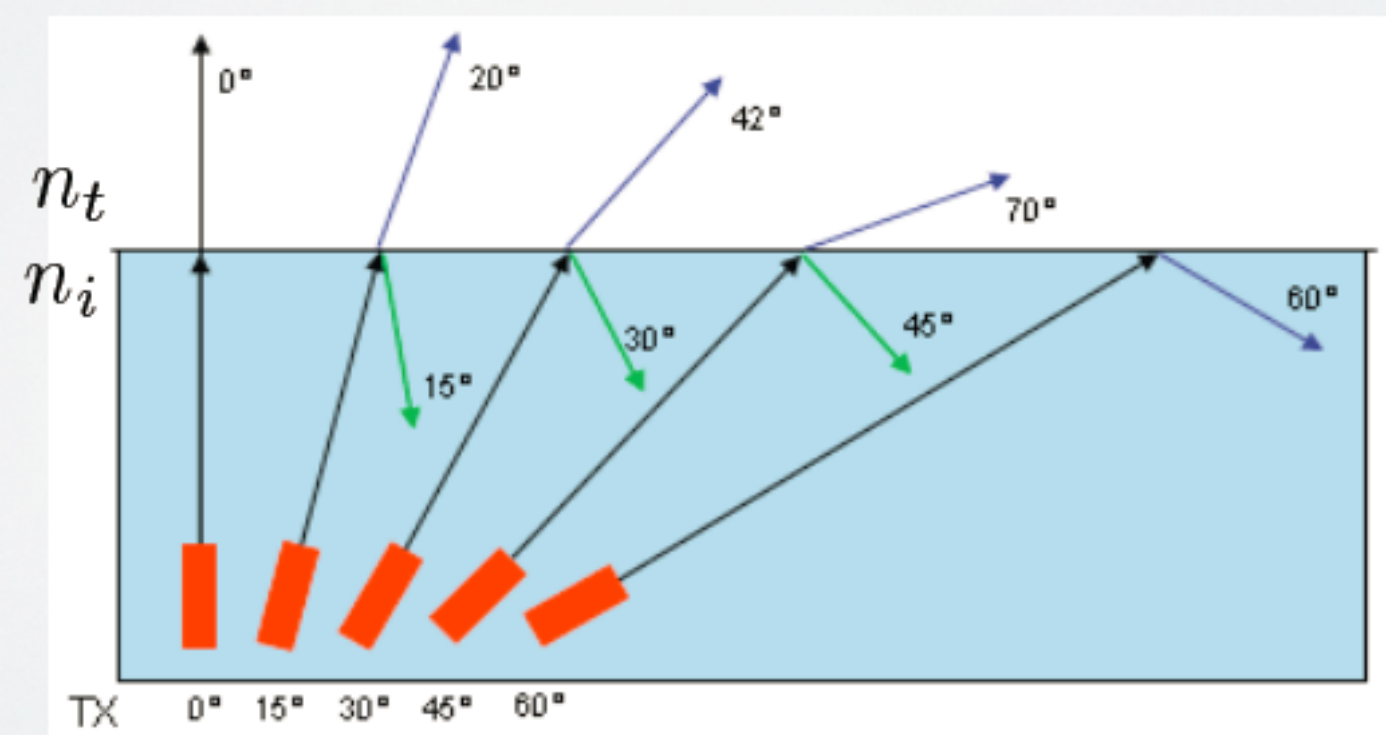


Refraction

What if this is imaginary?

$$\boldsymbol{t} = \frac{\eta_i\left(\boldsymbol{d} - (\boldsymbol{n}\cdot\boldsymbol{d})\boldsymbol{n}\right)}{\eta_t} - \sqrt{1 - \frac{\eta_i^2\left(1-(\boldsymbol{d}\cdot\boldsymbol{n})^2\right)}{\eta_t^2}}\,\boldsymbol{n}$$

Total Internal Reflection: Light trapped in the material



Refraction

Total Internal Reflection

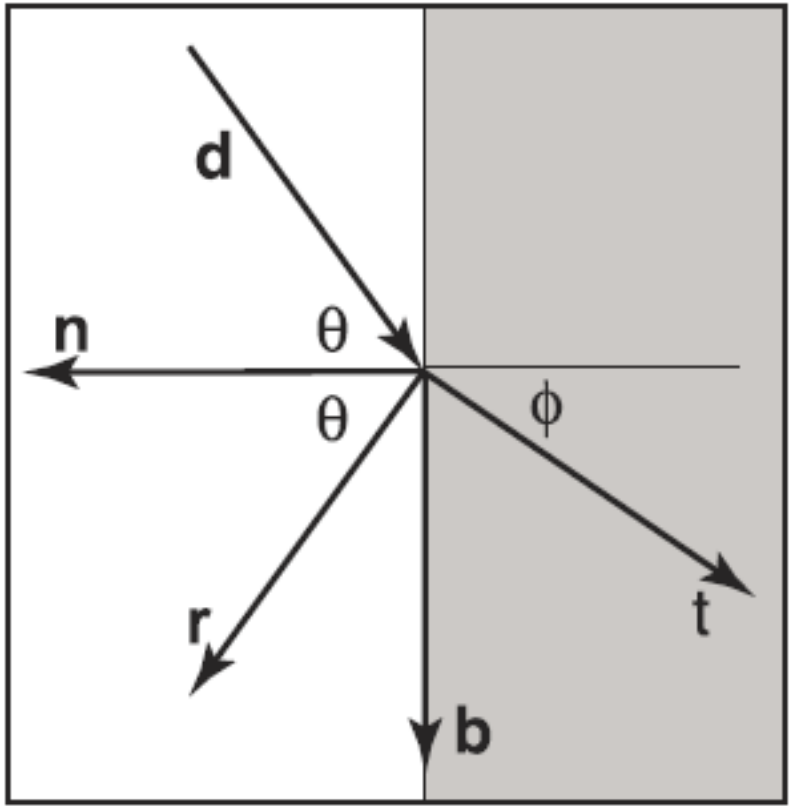
Refraction

Reflectivity varies with incidence angle

Schlick's approximation to Fresnel Equations

$$R(\theta_i) = R_0 + (1 - R_0)(1 - \cos\theta_i)^5$$

$$R_0 = \left(\frac{\eta_t - 1}{\eta_t + 1} \right)^2$$

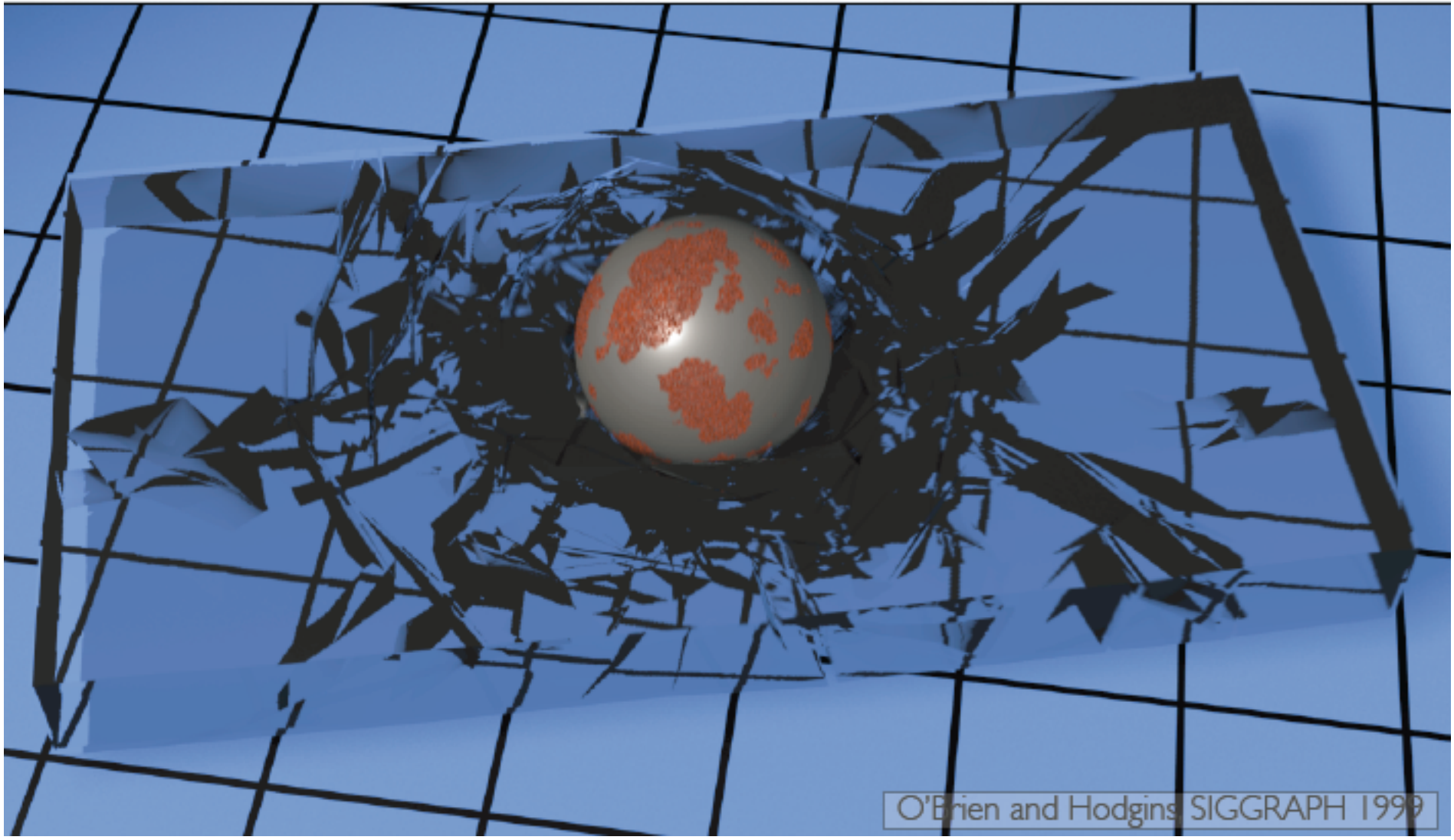


A fraction *R* is reflected, and the rest is refracted (transmitted)

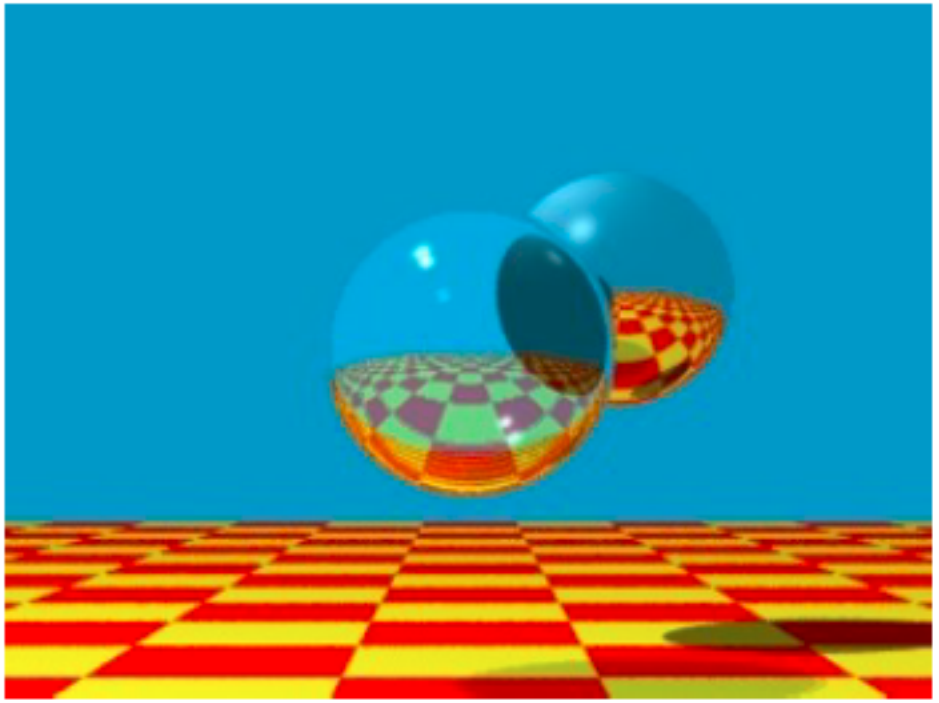
Refraction

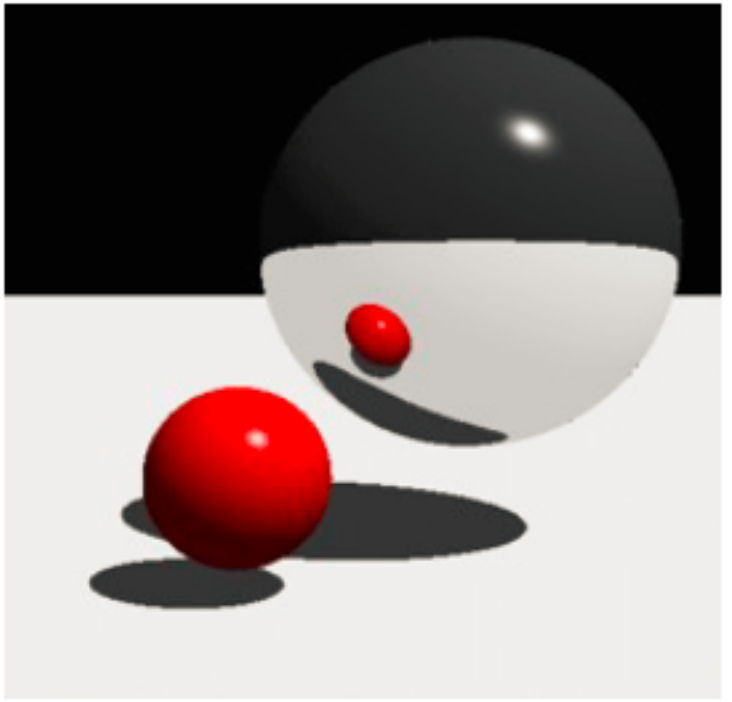
```
function ComputeShading(ray, t0, t1)
  Get intersection of ray with scene
  If intersection != null
    Color = ambient
    Get n, h, l, r, t
    If !blocked(shadowray, ε, ∞)
      ReflectedColor = ComputeShading(reflected ray, ε, ∞)
      If Not Total Internal Reflection
        ReflectedColor = R * ReflectedColor
          + (1 – R) * ComputeShading(refracted ray, ε, ∞)
      Color += kd * max(0, <n,l>) + ks * <h, n>^p
       + kt * ReflectedColor
  Else
    Color = background
```

Refraction



Distribution Ray Tracing



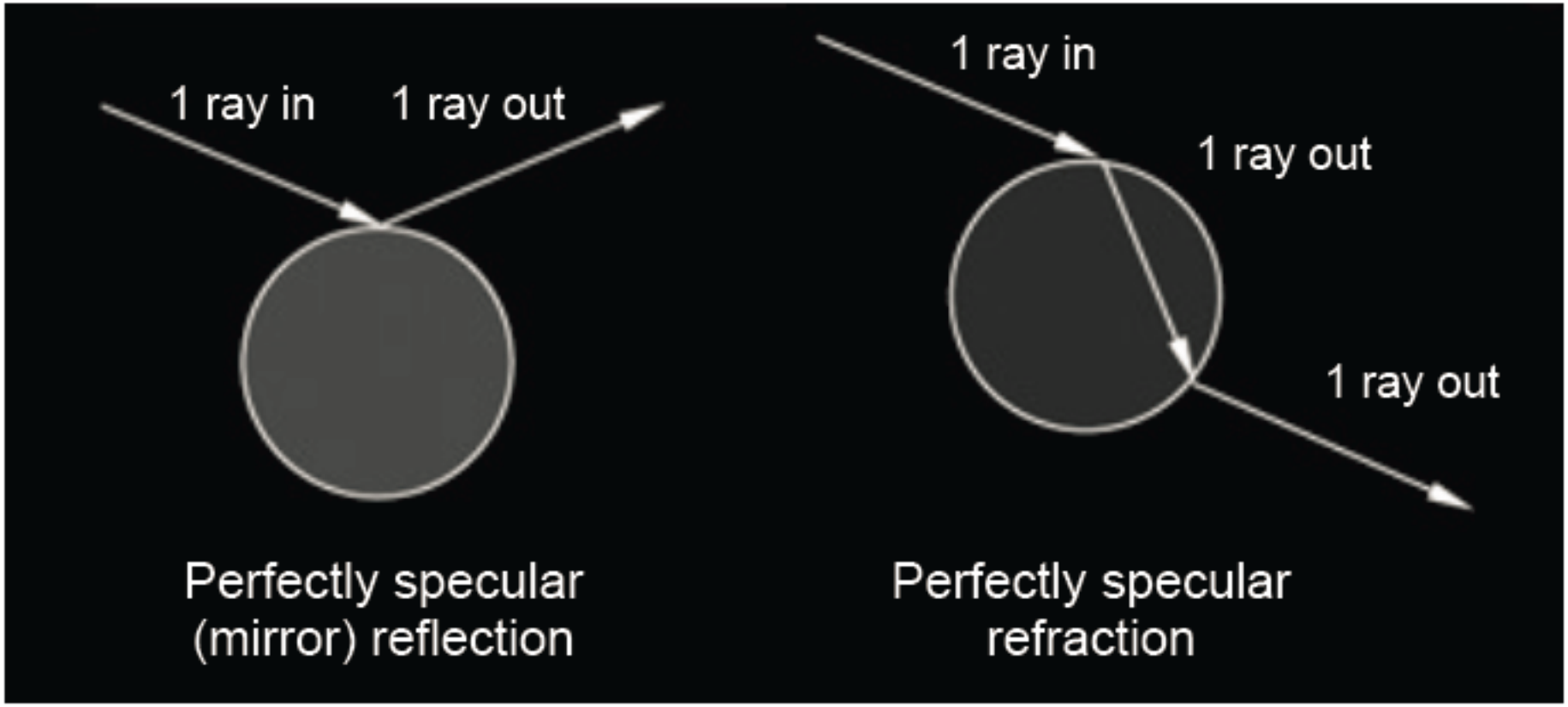


Anything wrong with these images?

They look too “clean” and “crisp” !

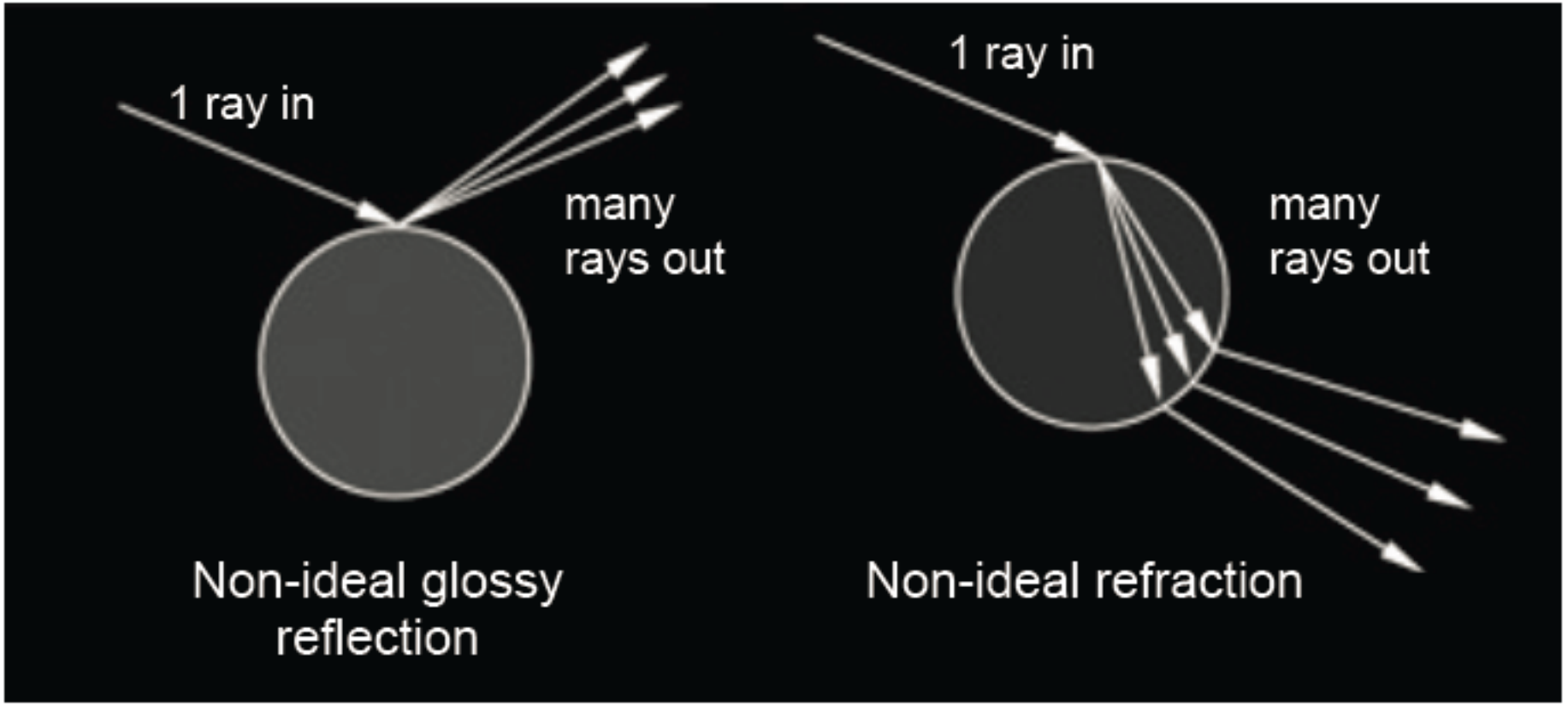
- Sharp reflections
- Sharp refractions
- Sharp shadows

Distribution Ray Tracing



Perfect/Ideal Surfaces

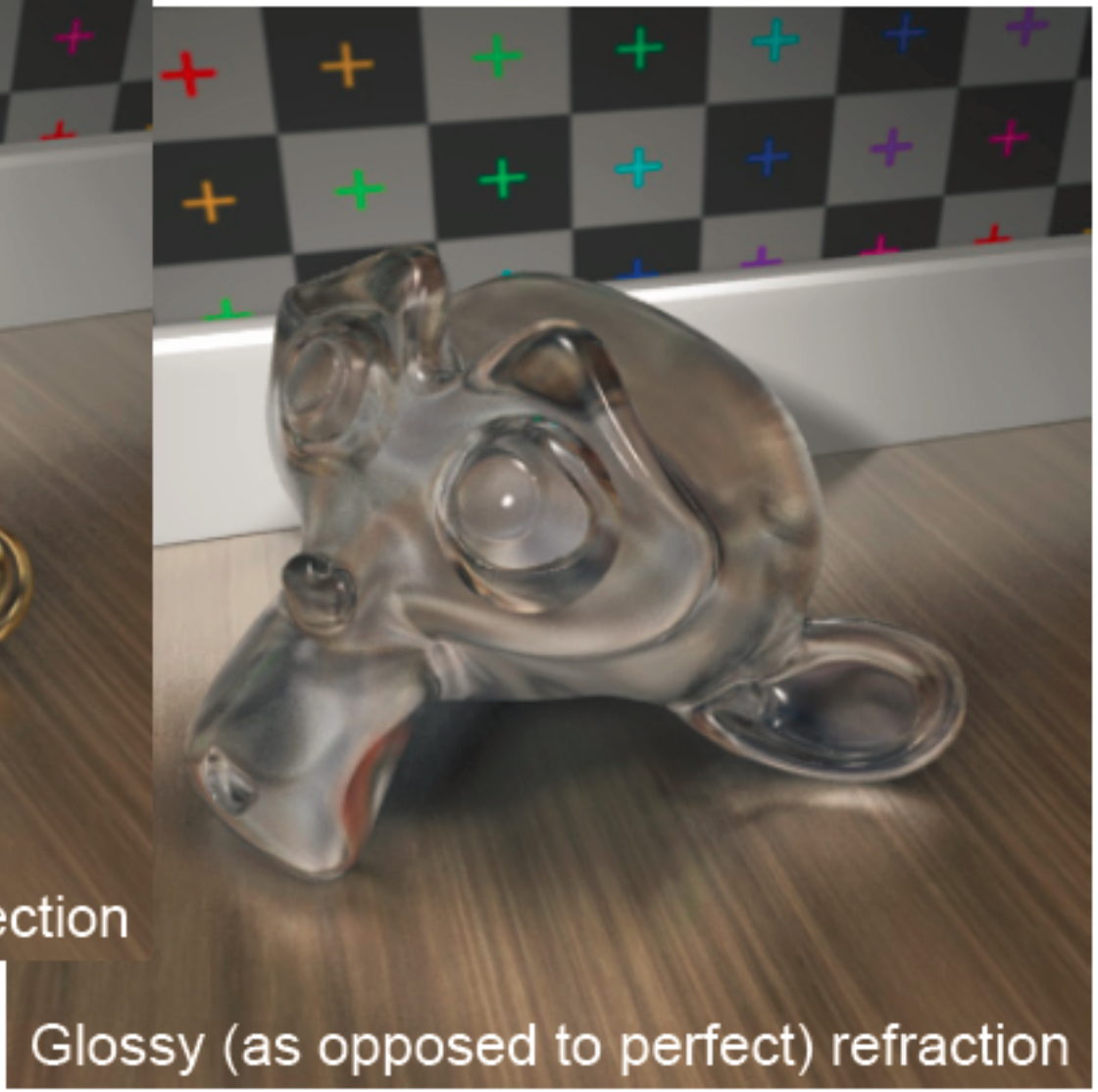
Distribution Ray Tracing



Realistic/Non-ideal Surfaces

Distribution Ray Tracing

Glossy (as opposed to mirror) reflection

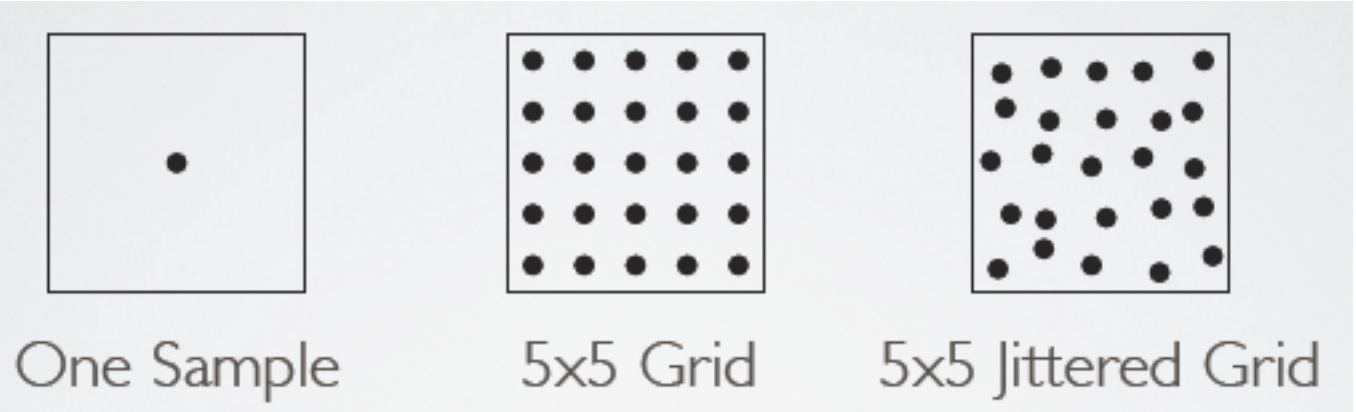


Glossy (as opposed to perfect) refraction

Distribution Ray Tracing

- Sending out multiple rays instead of one ray
- Used for many things:
  - Anti-aliasing (super-sampling)
  - Glossy reflections
  - Glossy refractions
  - Soft shadows

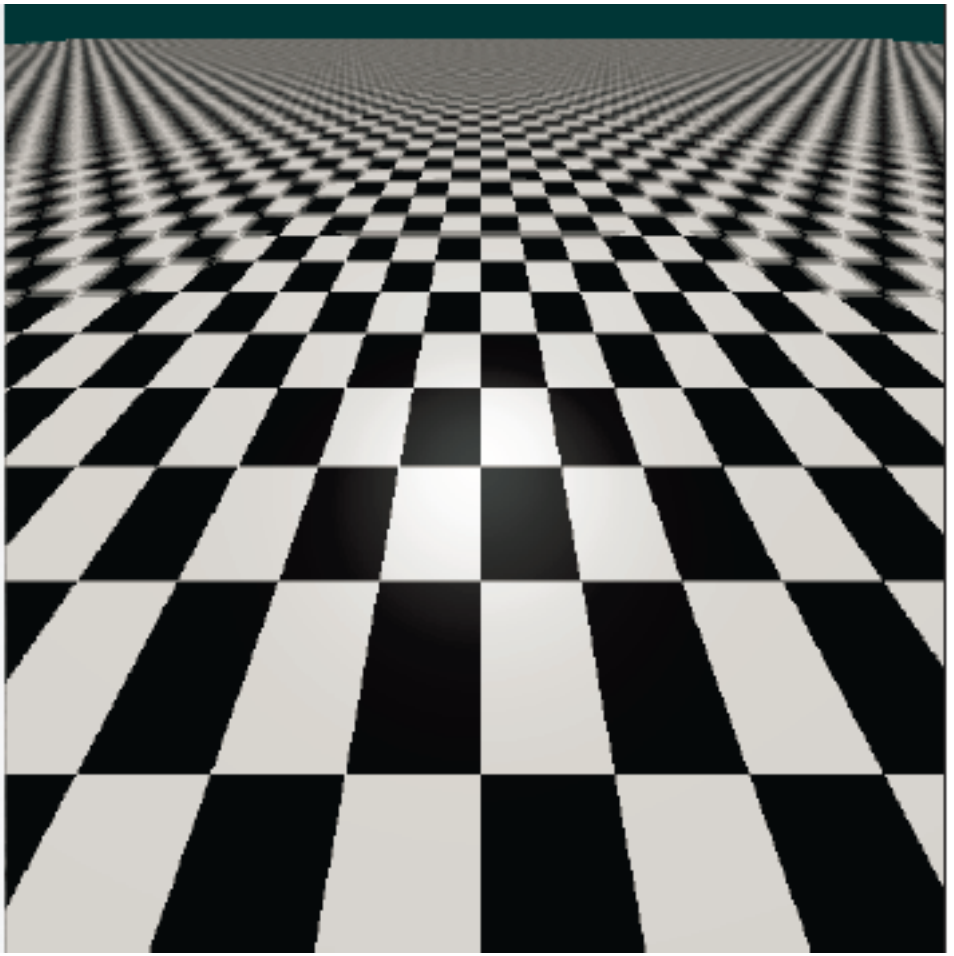
Anti-Aliasing



Send out multiple rays through each pixel and average results

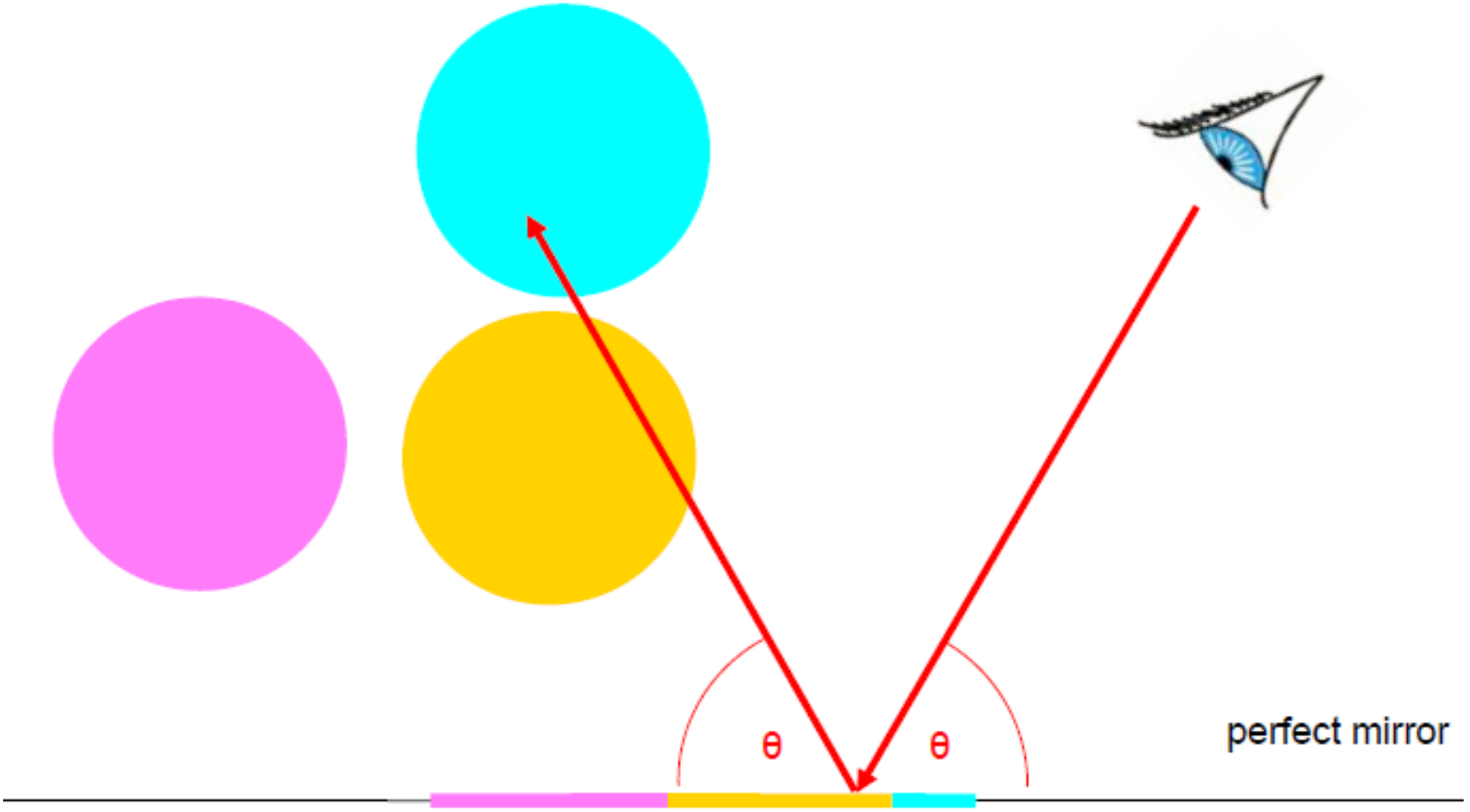
Anti-Aliasing

Without Anti-aliasing



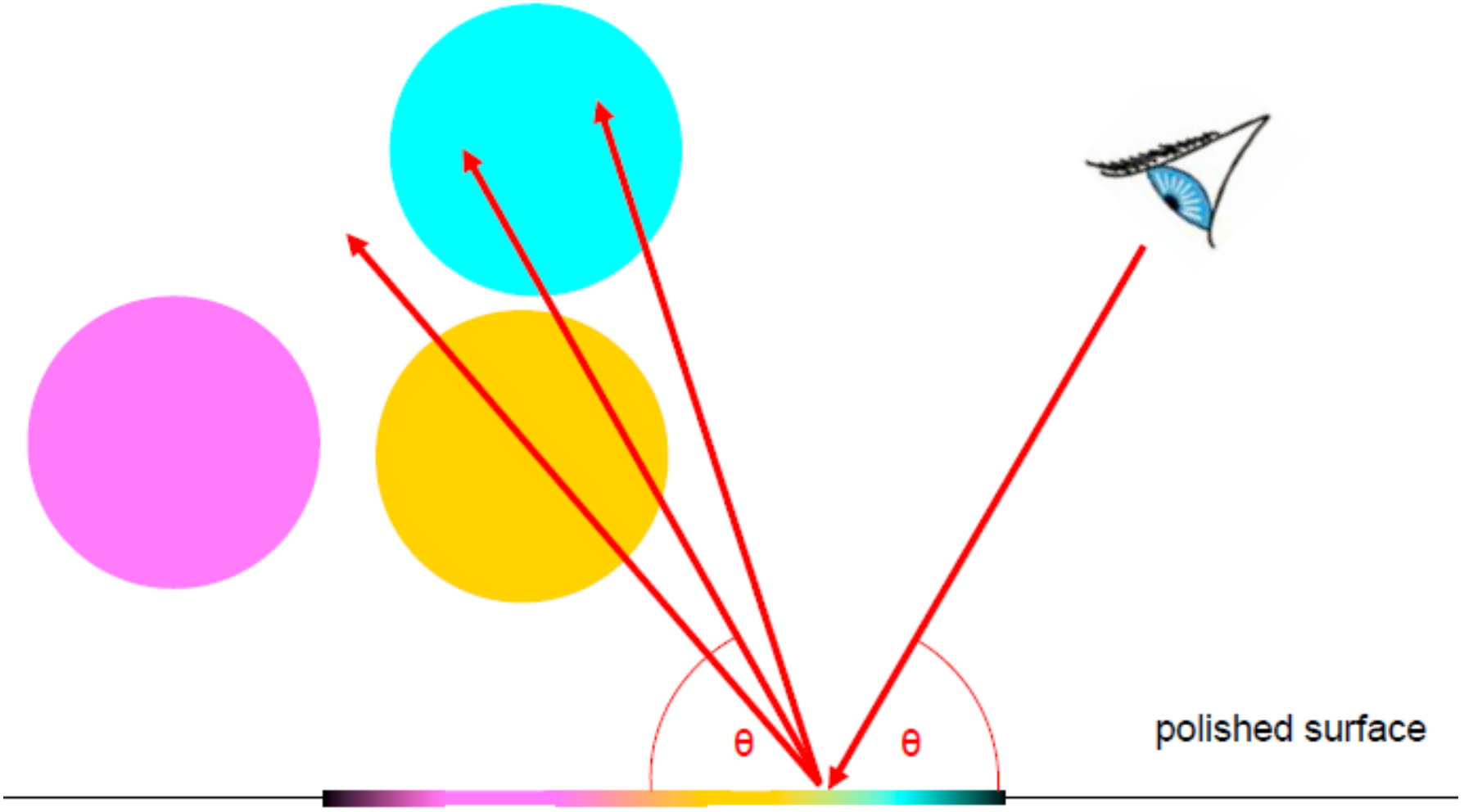
With Anti-aliasing

Glossy Reflections



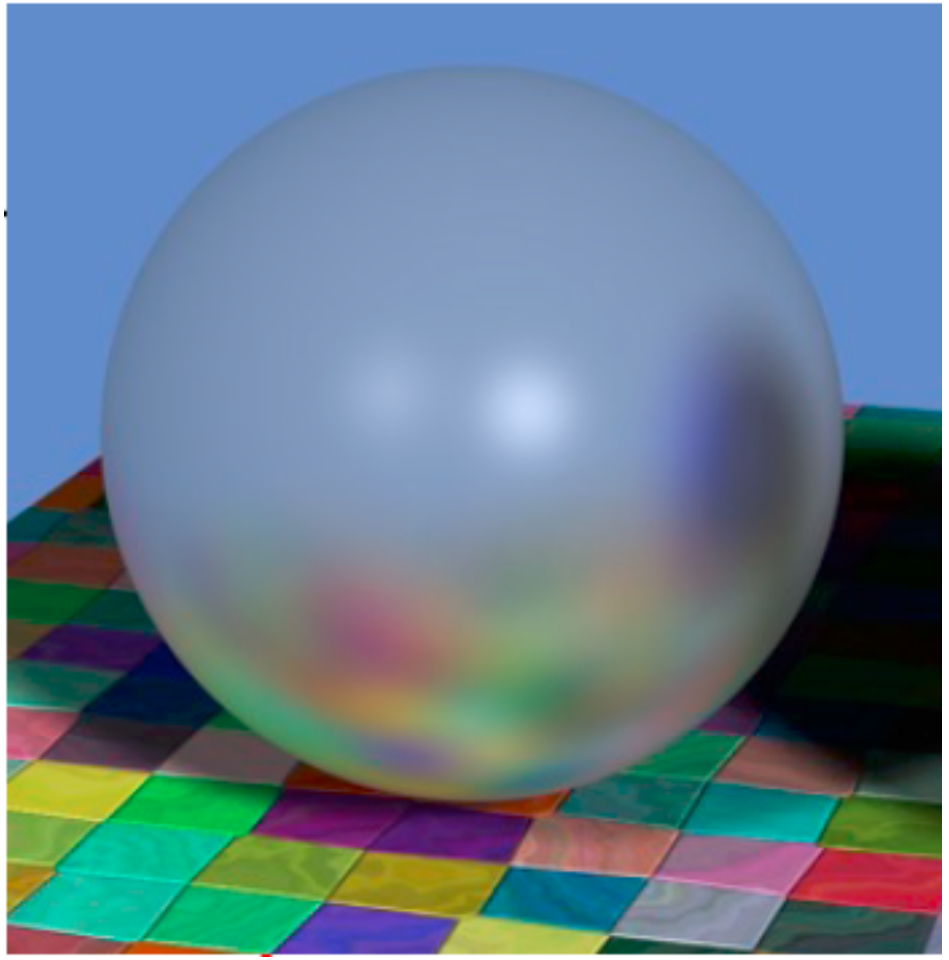
One ray per intersection

Glossy Reflections

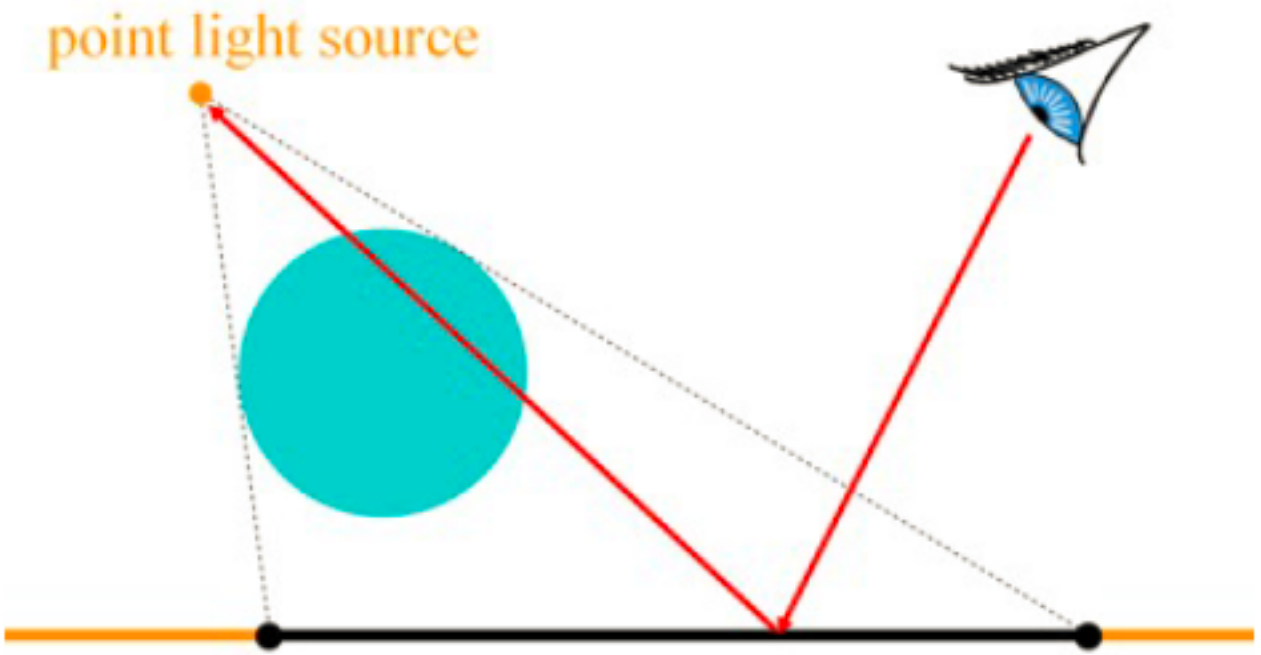


Multiple rays per intersection

Glossy Reflections

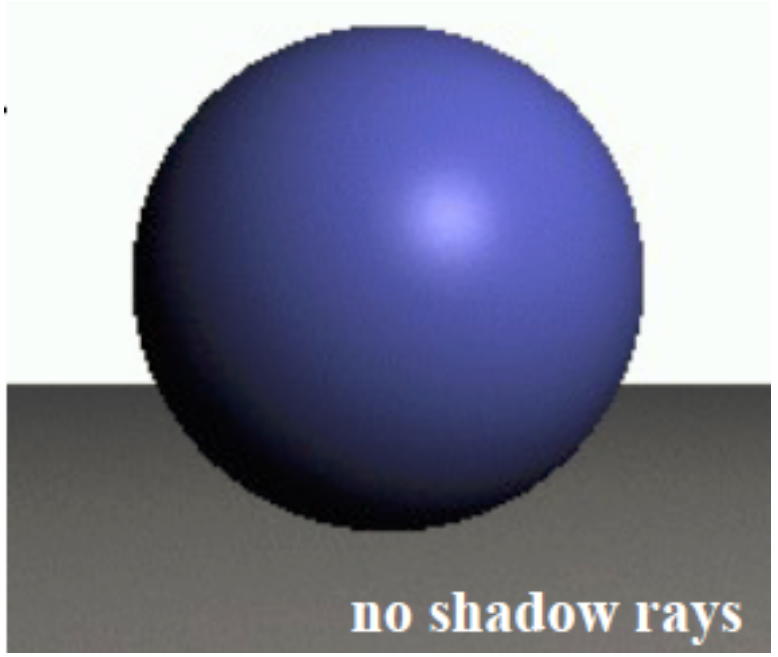


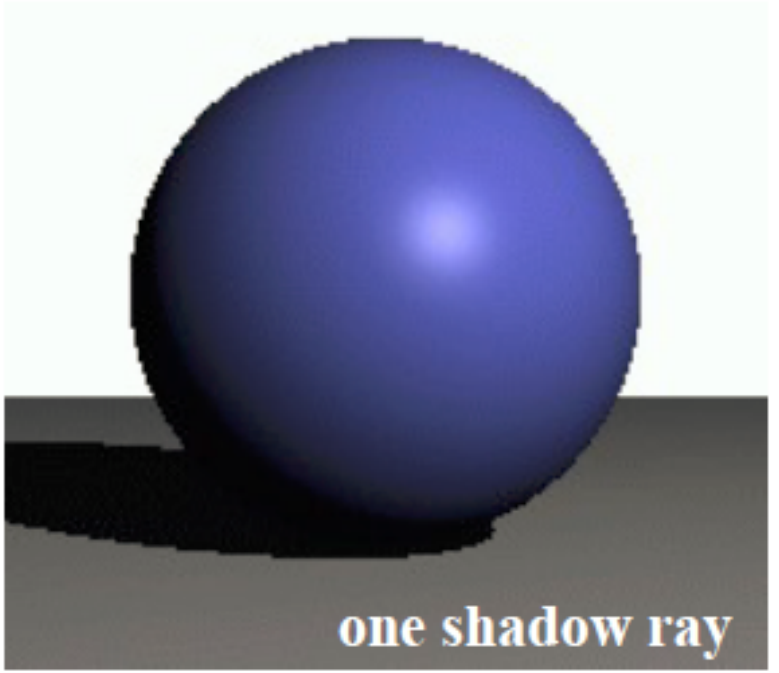
Soft Shadows



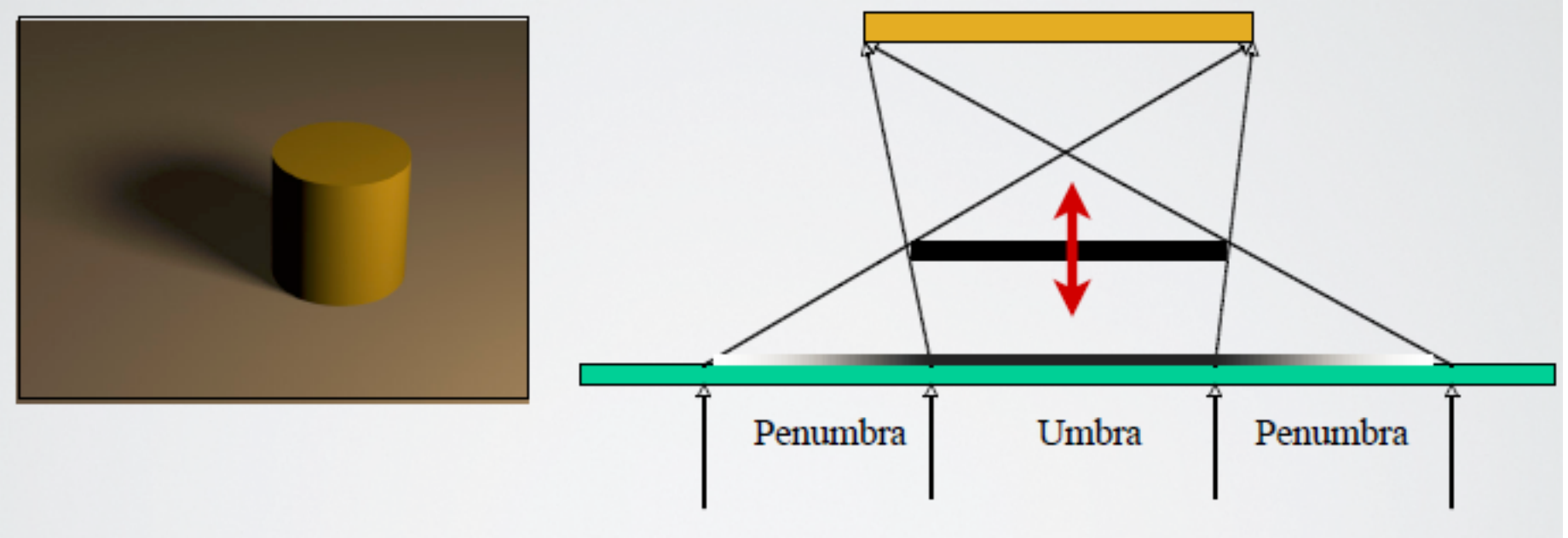
One shadow ray per intersection

Soft Shadows



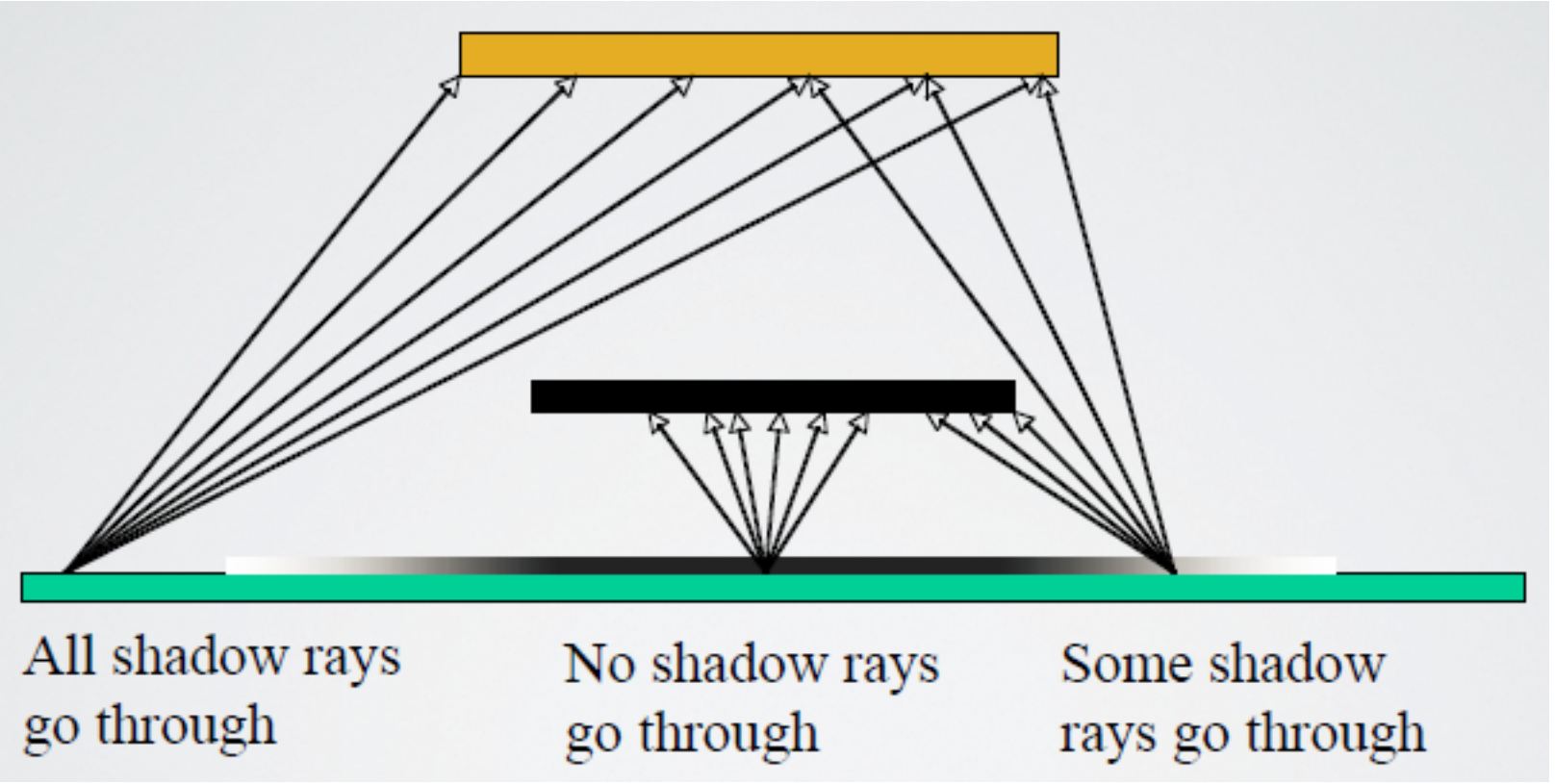


Soft Shadows



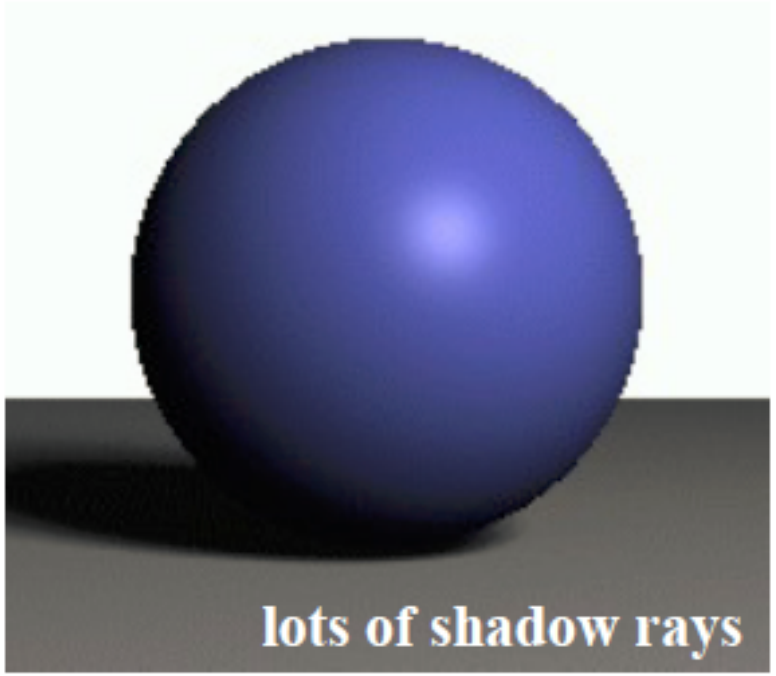
Soft shadows result from “area” light sources

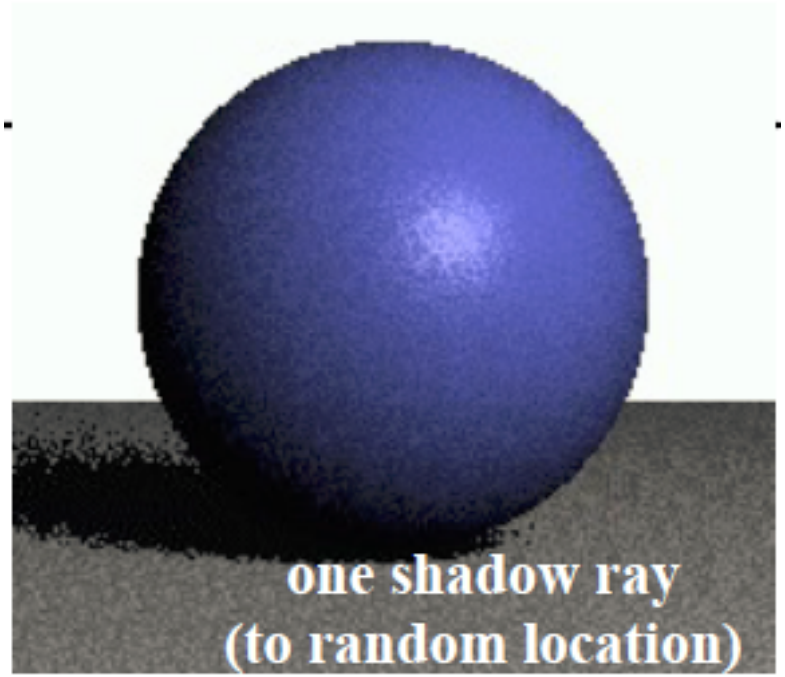
Soft Shadows



Multiple “random” shadow rays per intersection

Soft Shadows





Recap

- Shadows
- Reflection
- Refraction
- Distribution Ray Tracing