

Fall 2017

CSCI 420: **Computer Graphics**

7.2 Ray Tracing



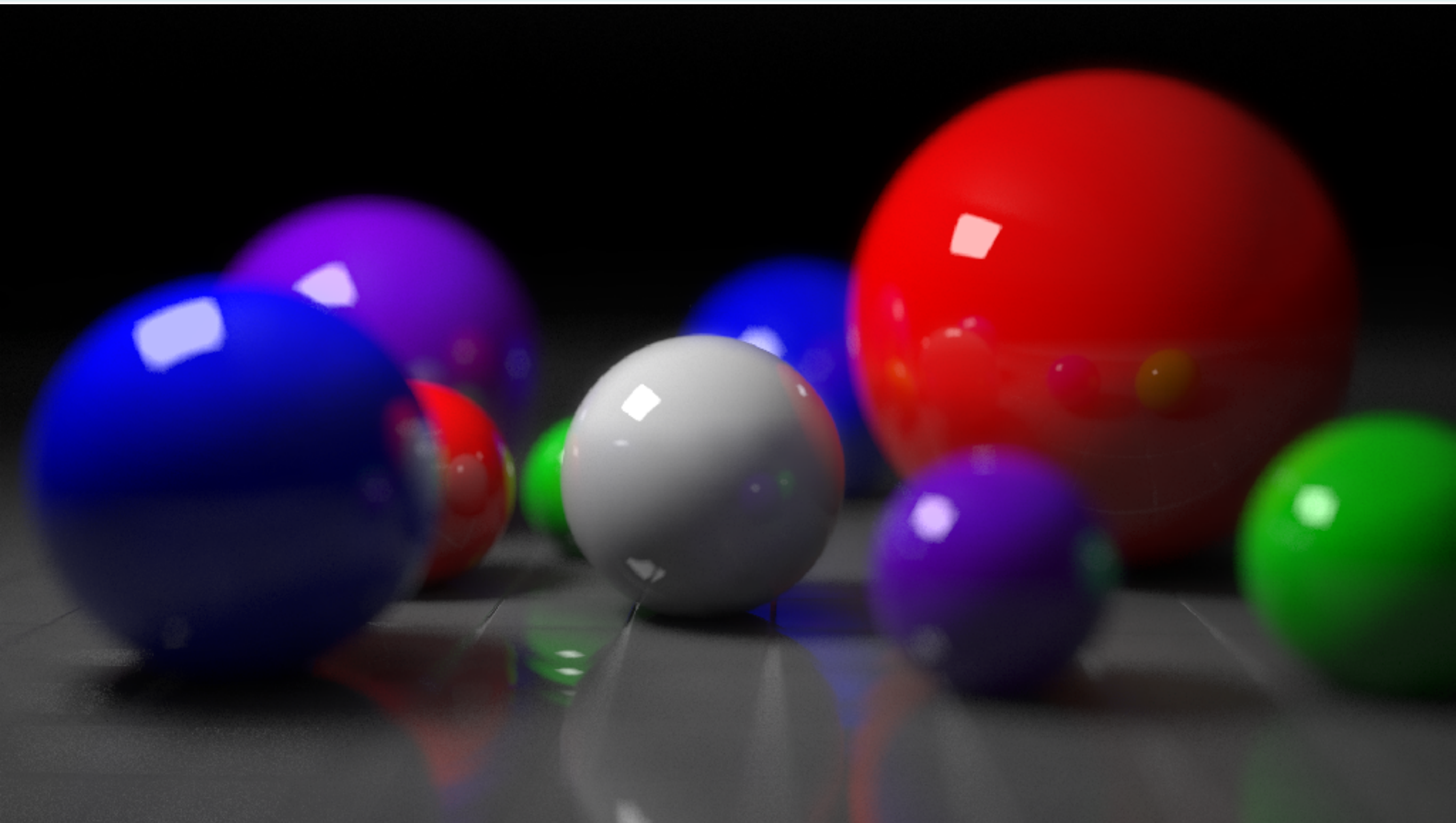
Hao Li

<http://cs420.hao-li.com>

Motivation: Reflections

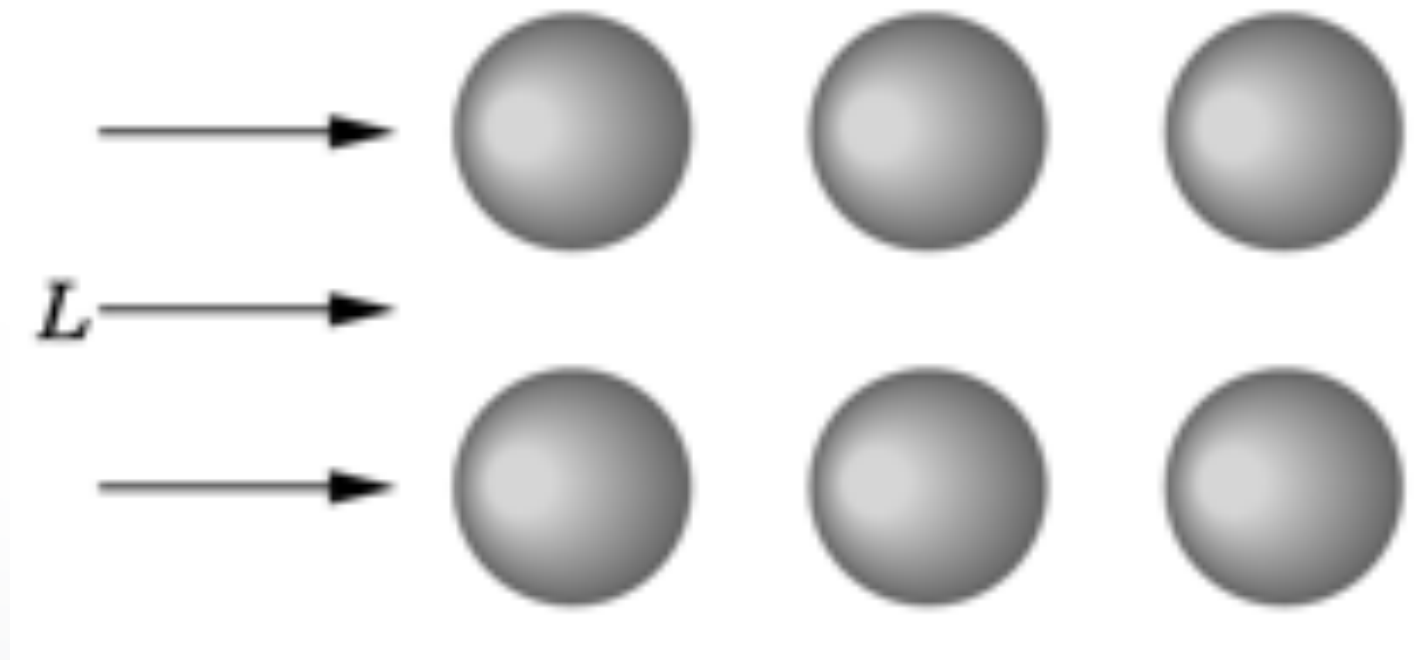


Motivation: Depth of Field



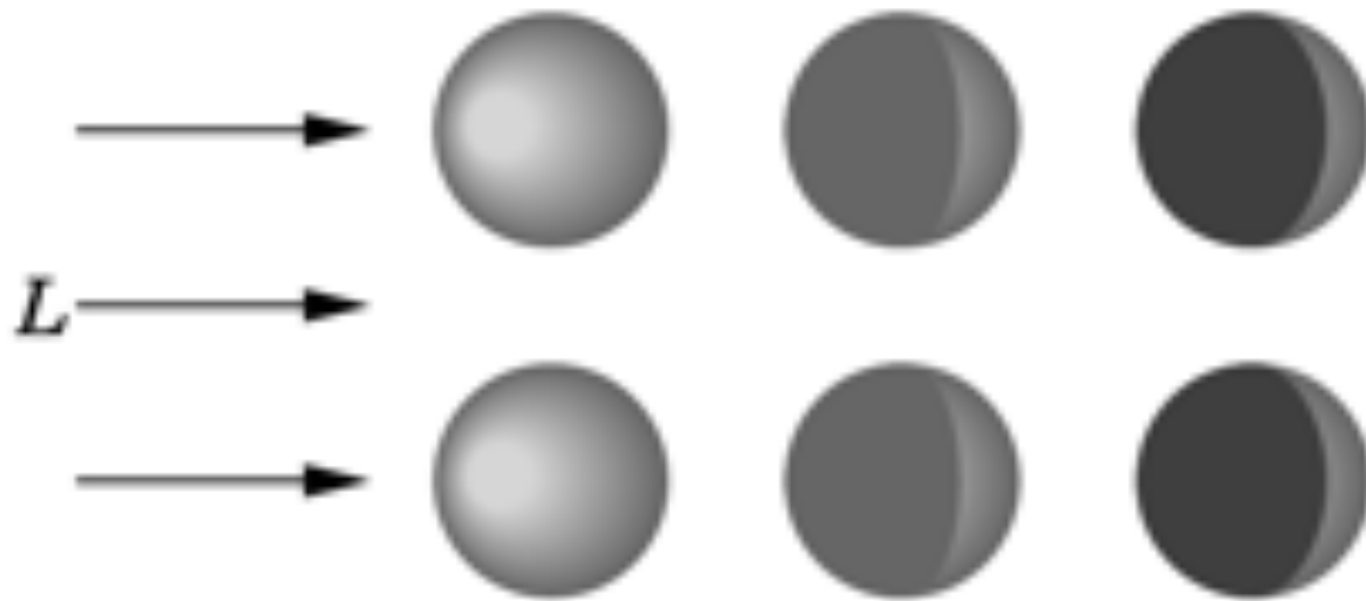
Local Illumination

- Object illuminations are independent
- No light scattering between objects
- No real shadows, reflection, transmission
- OpenGL pipeline uses this



Global Illumination

- Ray tracing (highlights, reflection, transmission)
- Radiosity (surface inter reflections)
- Photon mapping
- Precomputed Radiance Transfer (PRT)



Object Space

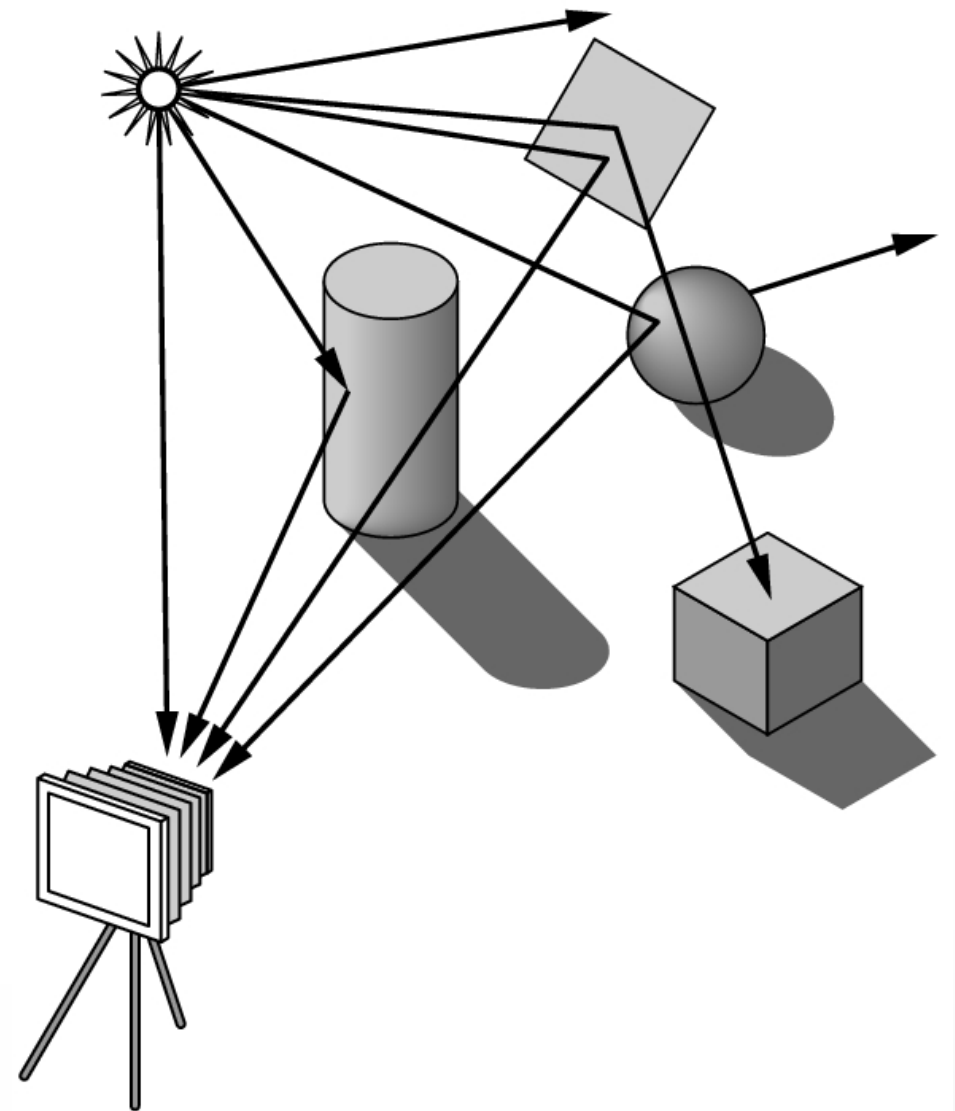
- Graphics pipeline: **for each object**, render
 - Efficient pipeline architecture, real-time
 - Difficulty: object interactions (shadows, reflections, etc.)

Image Space

- Ray tracing: for each pixel, determine color
 - Pixel-level parallelism
 - Difficulty: very intensive computation, usually off-line

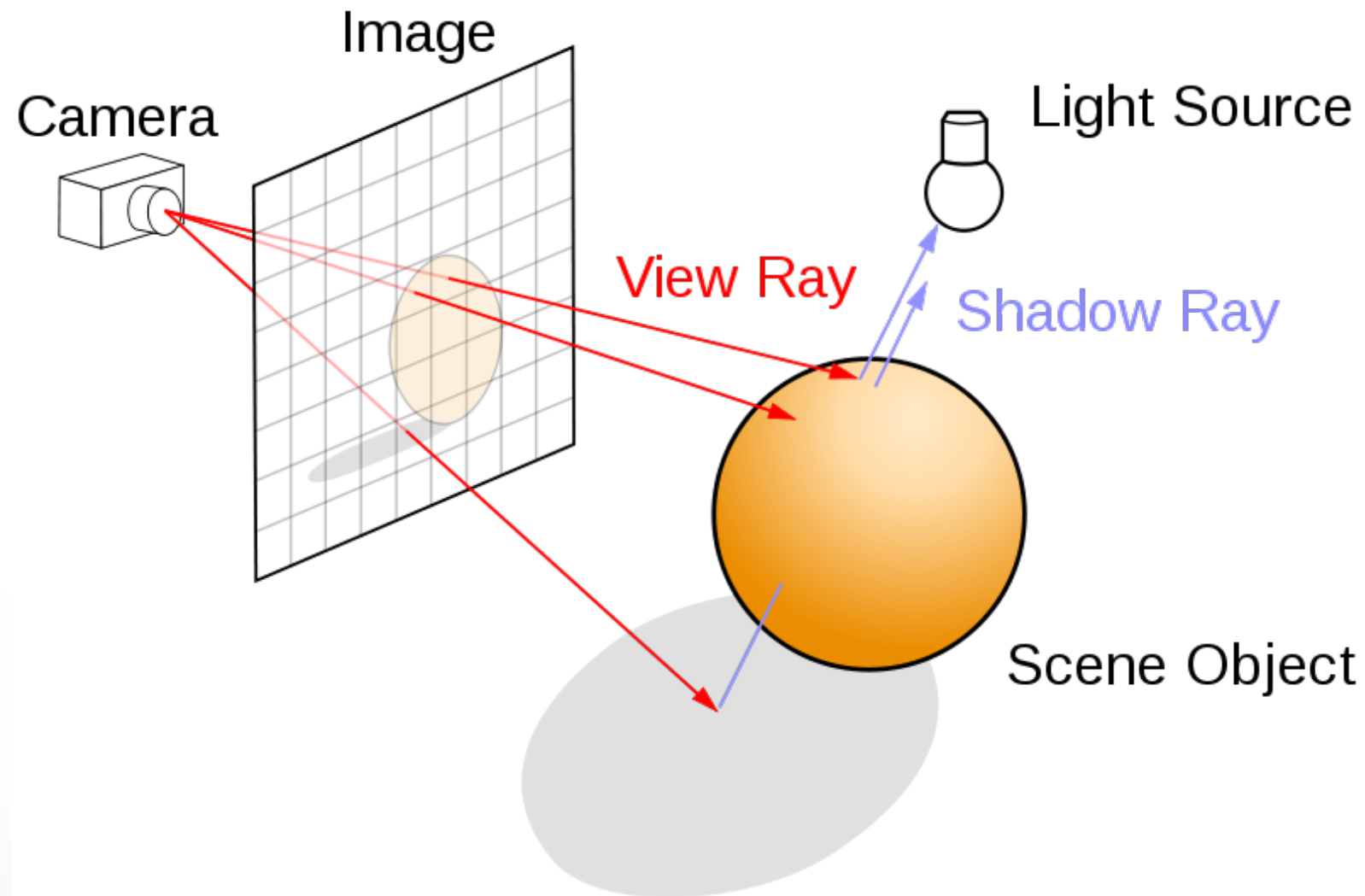
First idea: Forward Ray Tracing

- Shoot (many) light rays from each light source
- Rays bounce off the objects
- Simulates paths of photons
- Problem: many rays will
- miss camera and not contribute to image!
- This algorithm is not practical



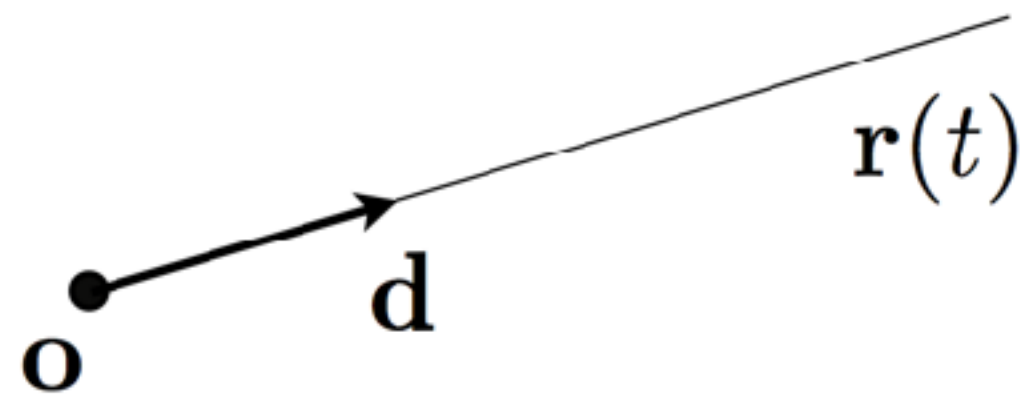
Backward Ray Tracing

- Shoot one ray from camera through each pixel in image plane



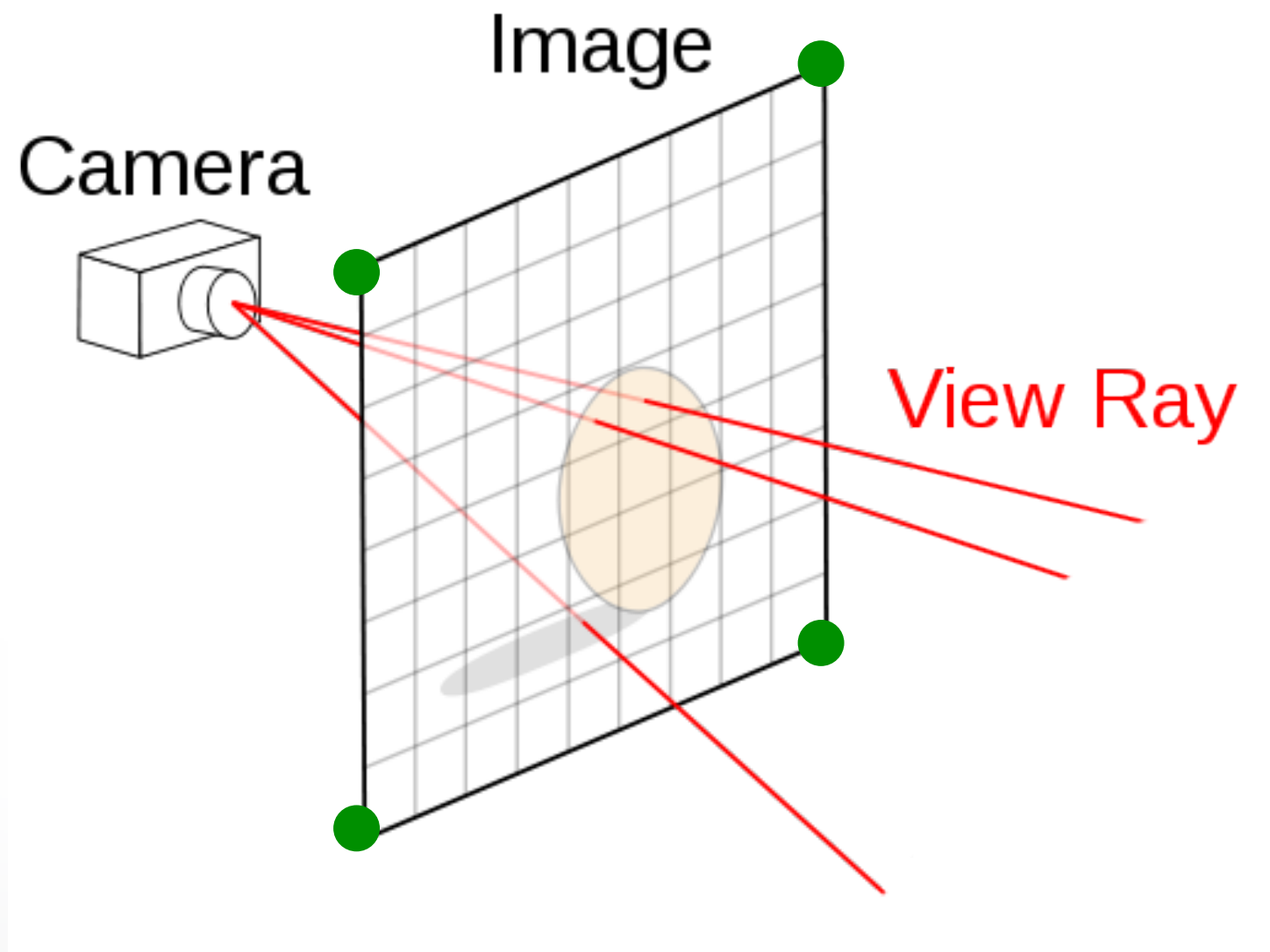
Generating Rays

$$\underset{\substack{\uparrow \\ \text{ray}}}{\mathbf{r}(t)} = \underset{\substack{\uparrow \\ \text{origin}}}{\mathbf{o}} + t \underset{\substack{\swarrow \\ \text{direction}}}{\mathbf{d}}$$



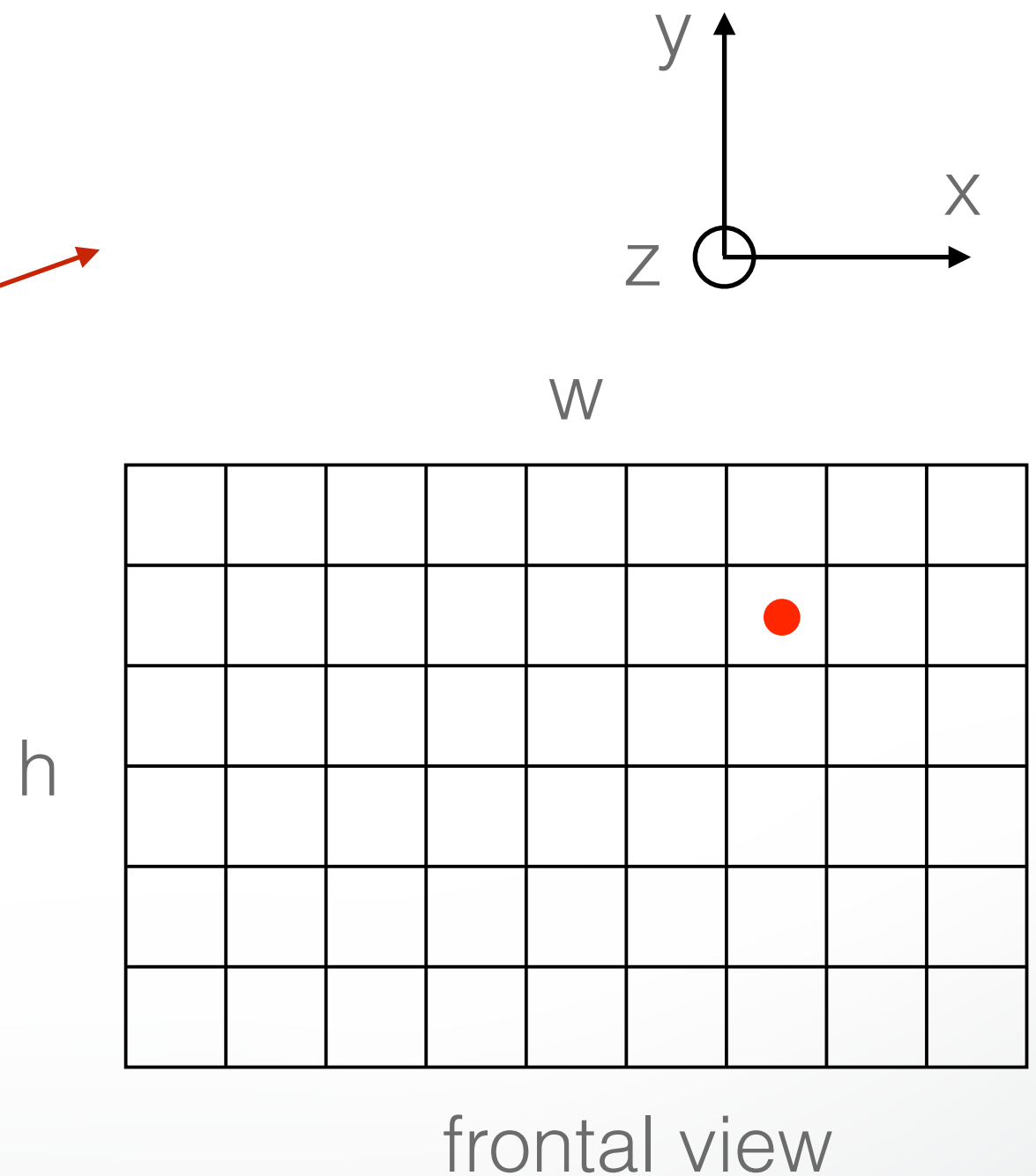
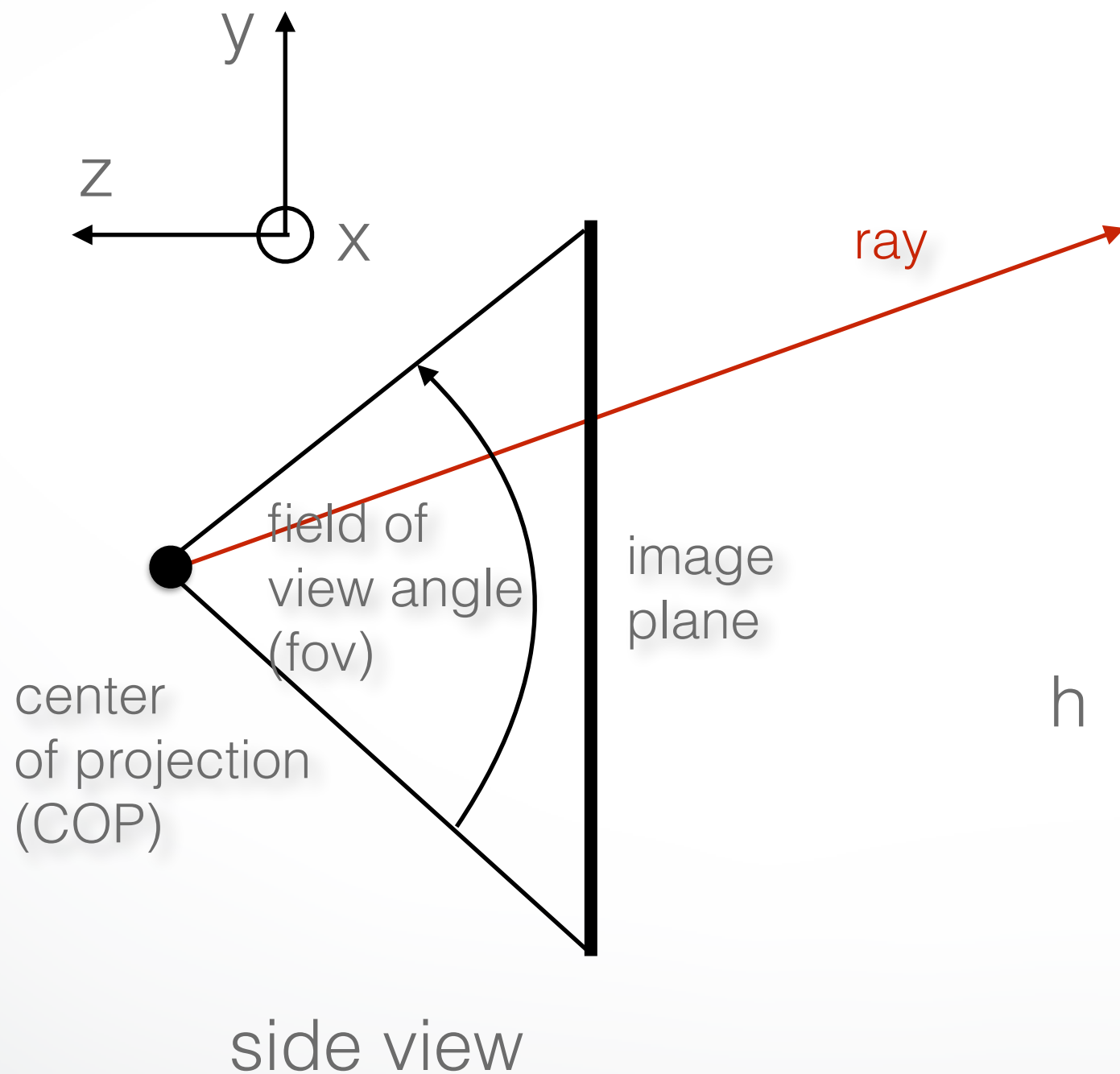
Generating Rays

- Camera is at (0,0,0) and points in the negative z-direction
- Must determine coordinates of image corners in 3D

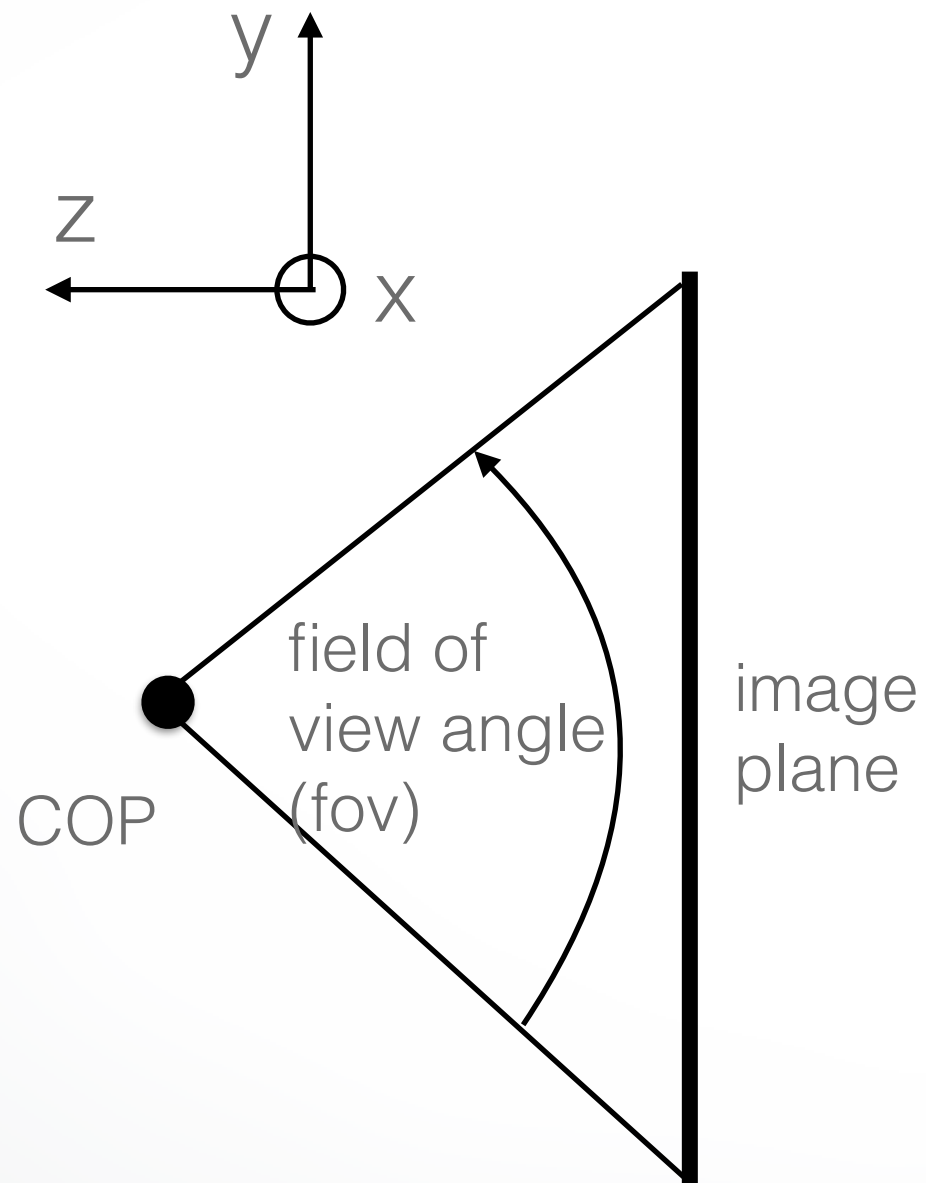


Generating Rays

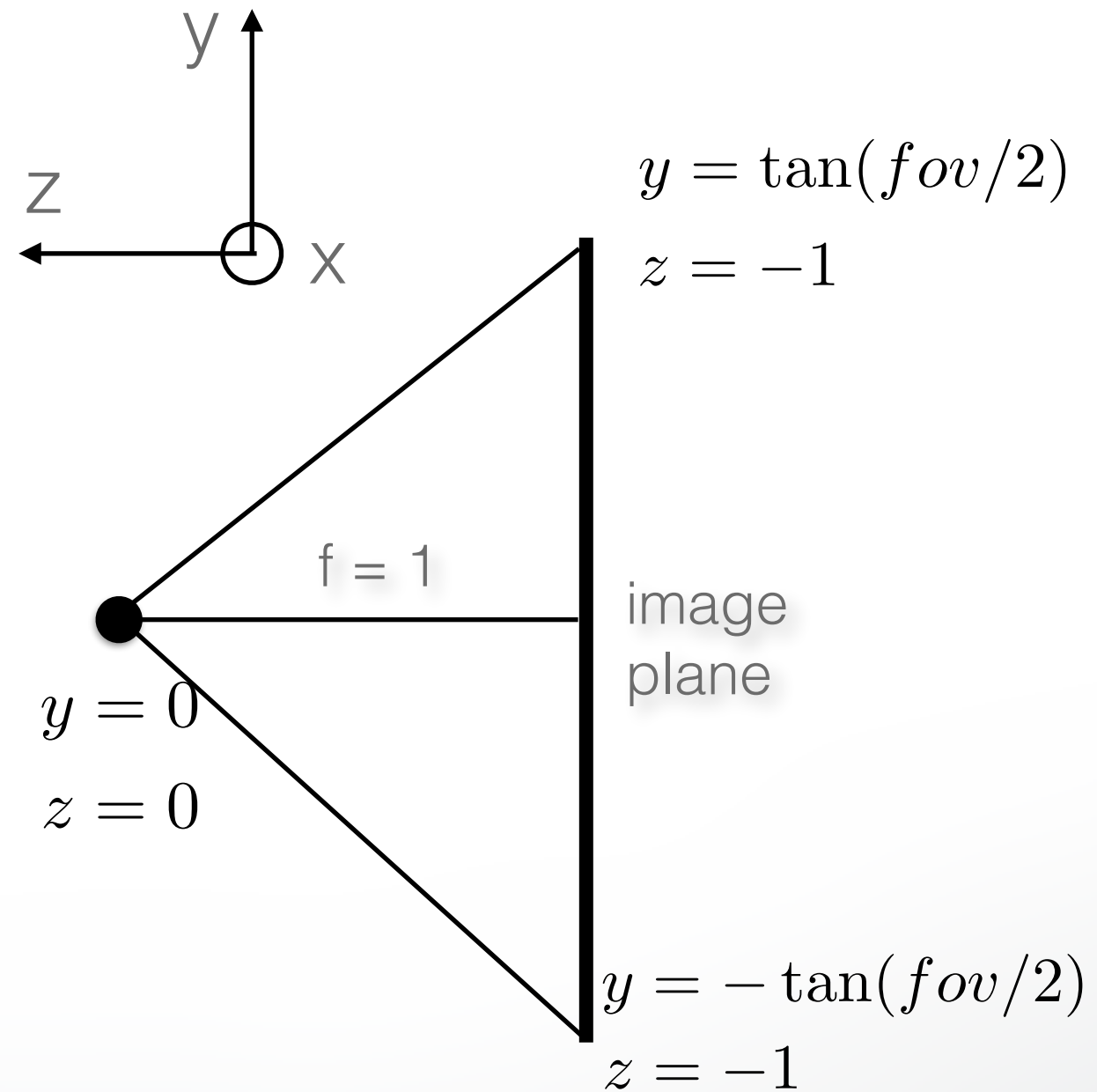
$$\text{aspect ratio} = w / h$$



Generating Rays

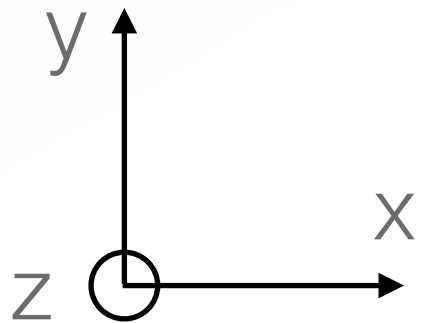


side view

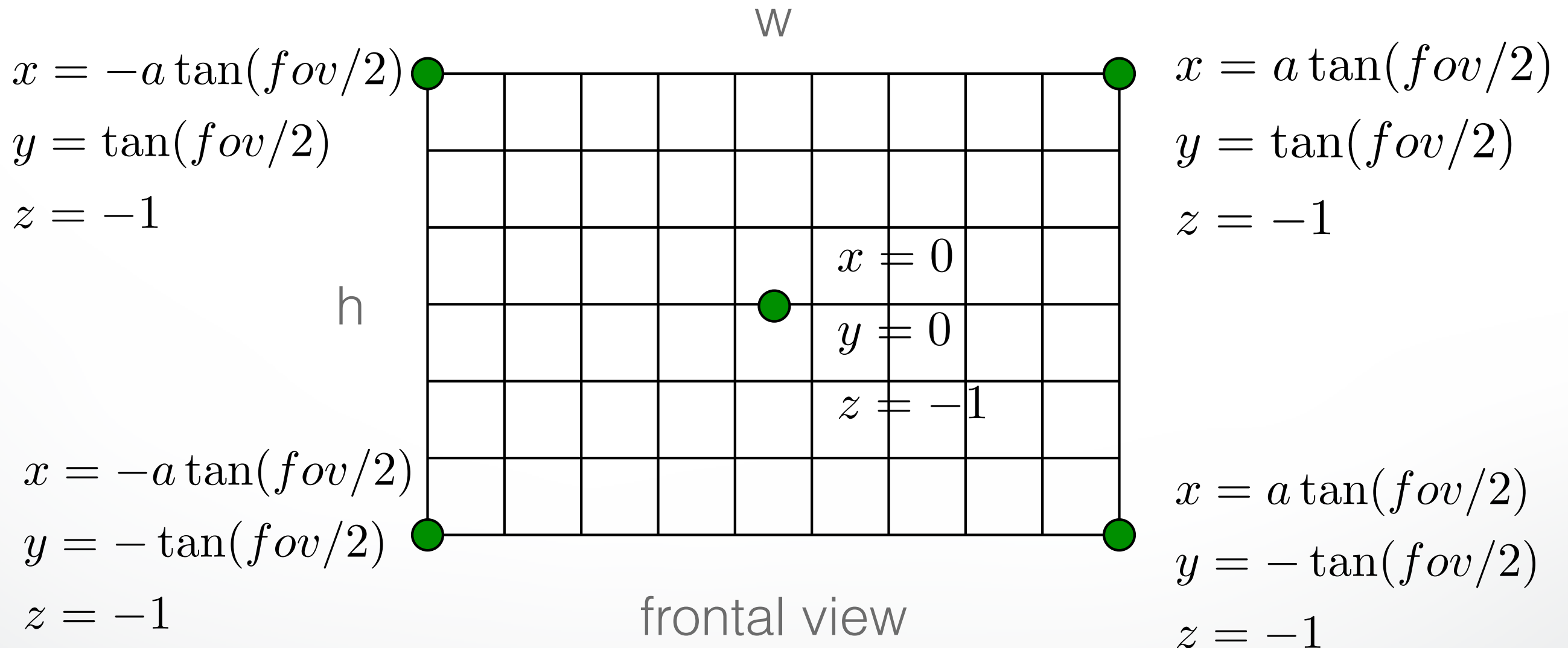


side view

Generating Rays



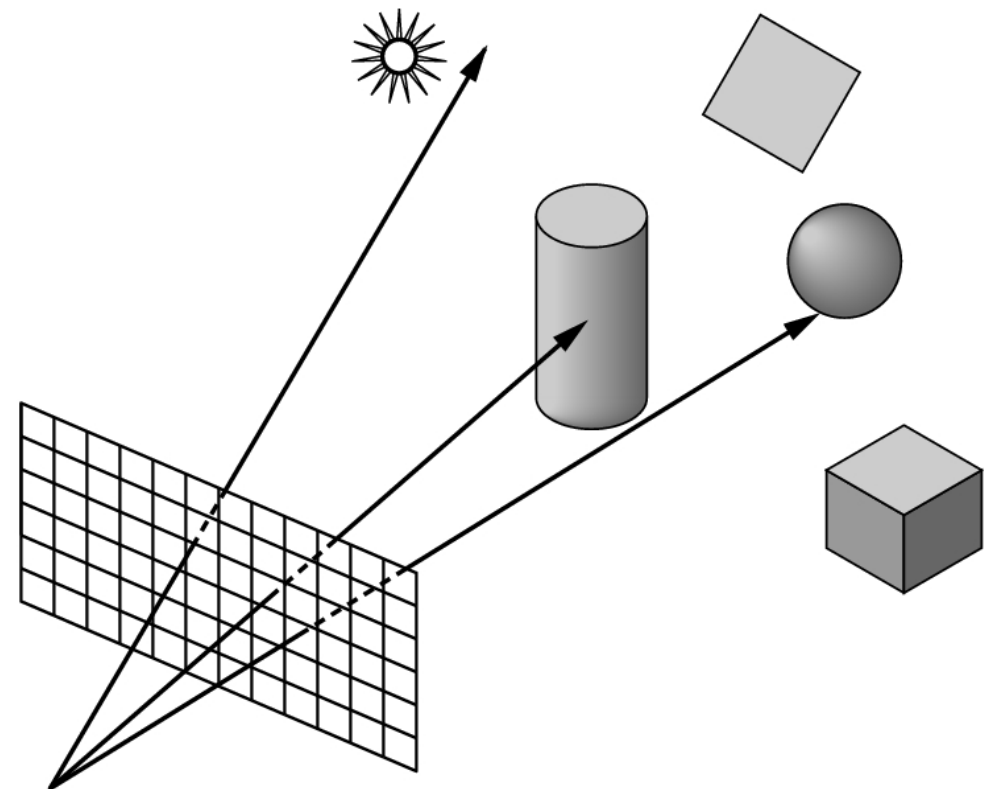
$a = \text{aspect ratio} = w / h$



Determining Pixel Color

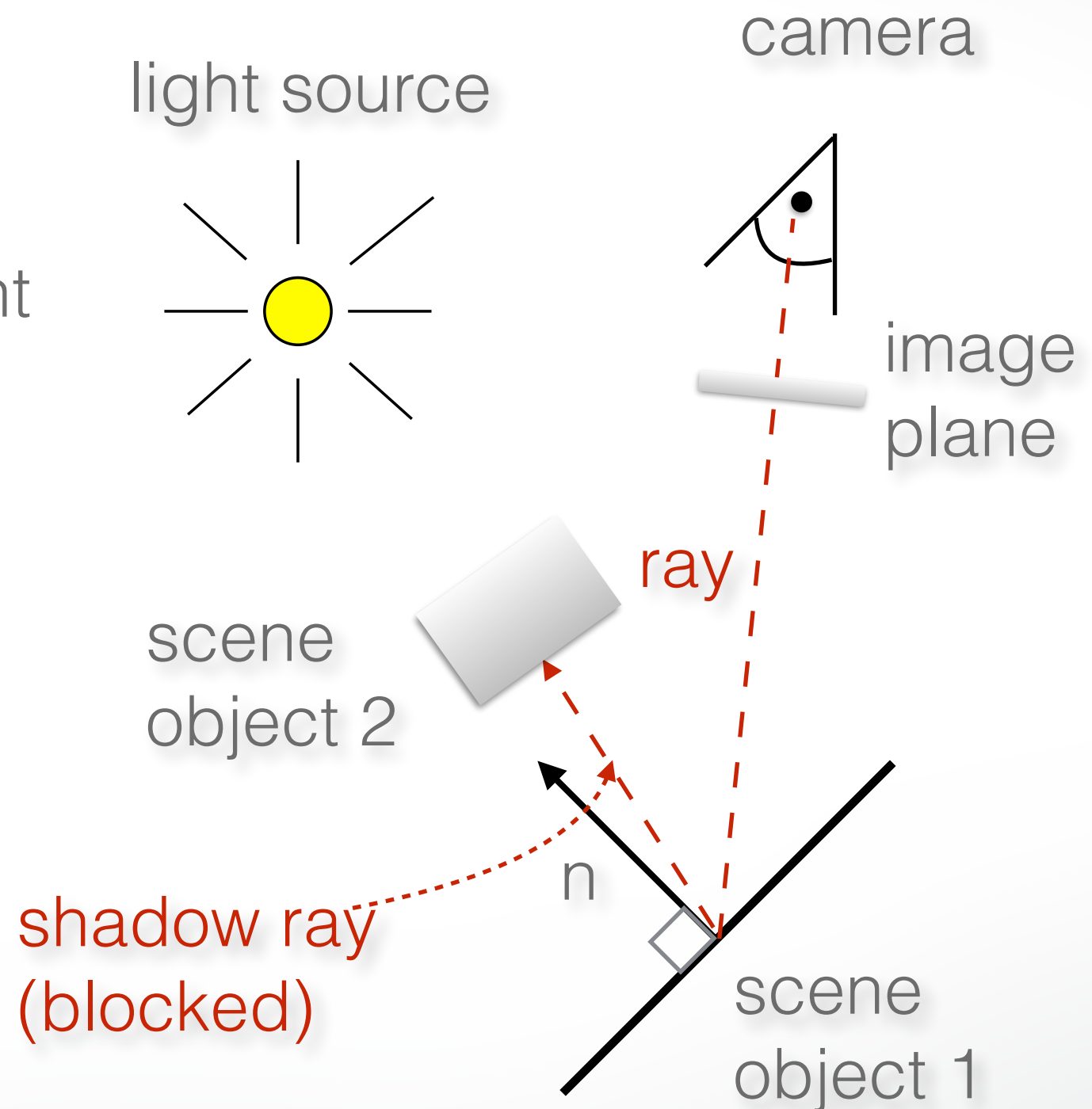
1. Phong model (local as before)
2. Shadow rays
3. Specular reflection
4. Specular transmission

Steps (3) and (4) require recursion.



Shadow Rays

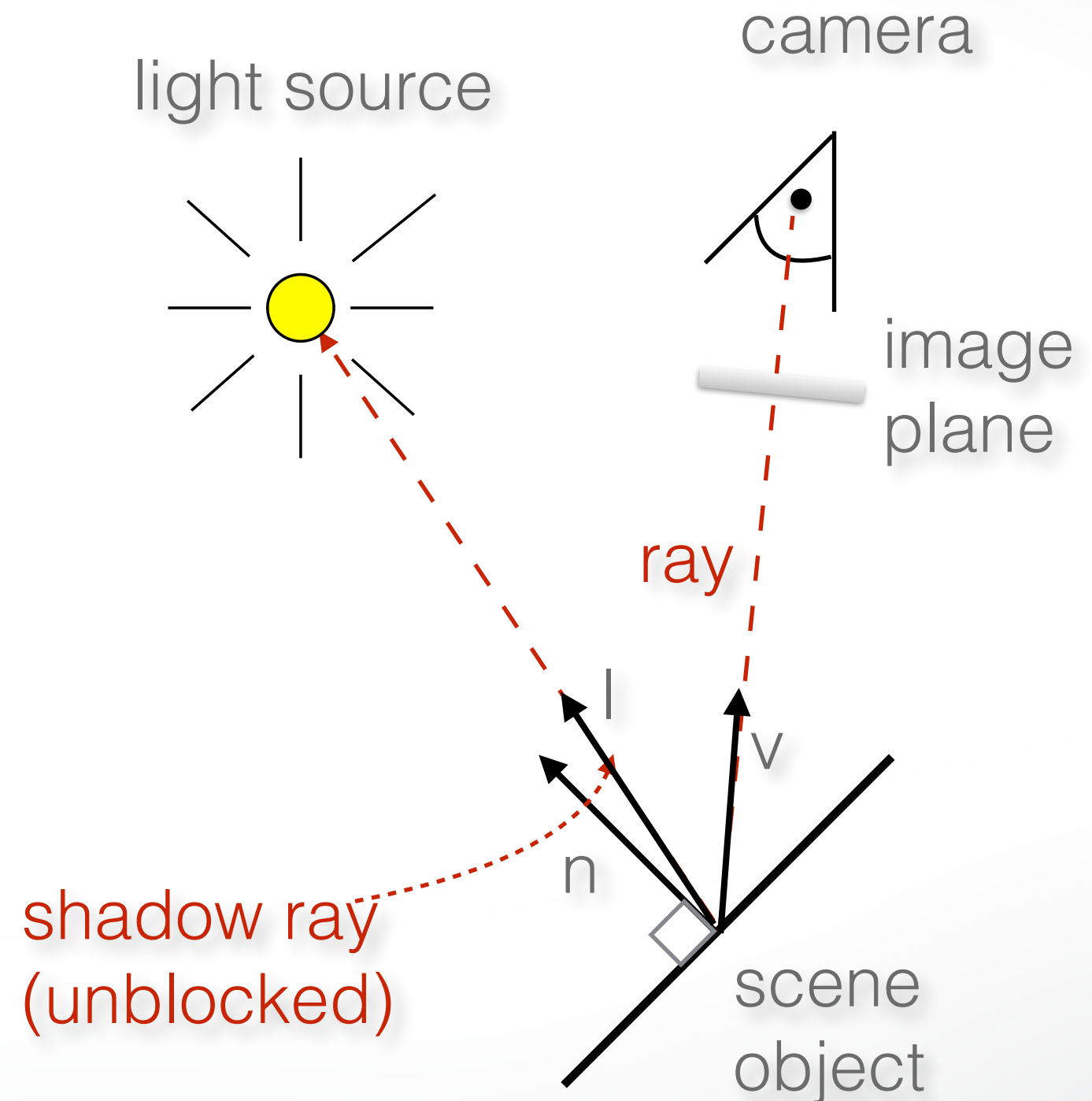
- Determine if light “really” hits surface point
- Cast **shadow ray** from surface point to each light
- If shadow ray hits opaque object, no contribution from that light
- This is essentially improved diffuse reflection



Phong Model

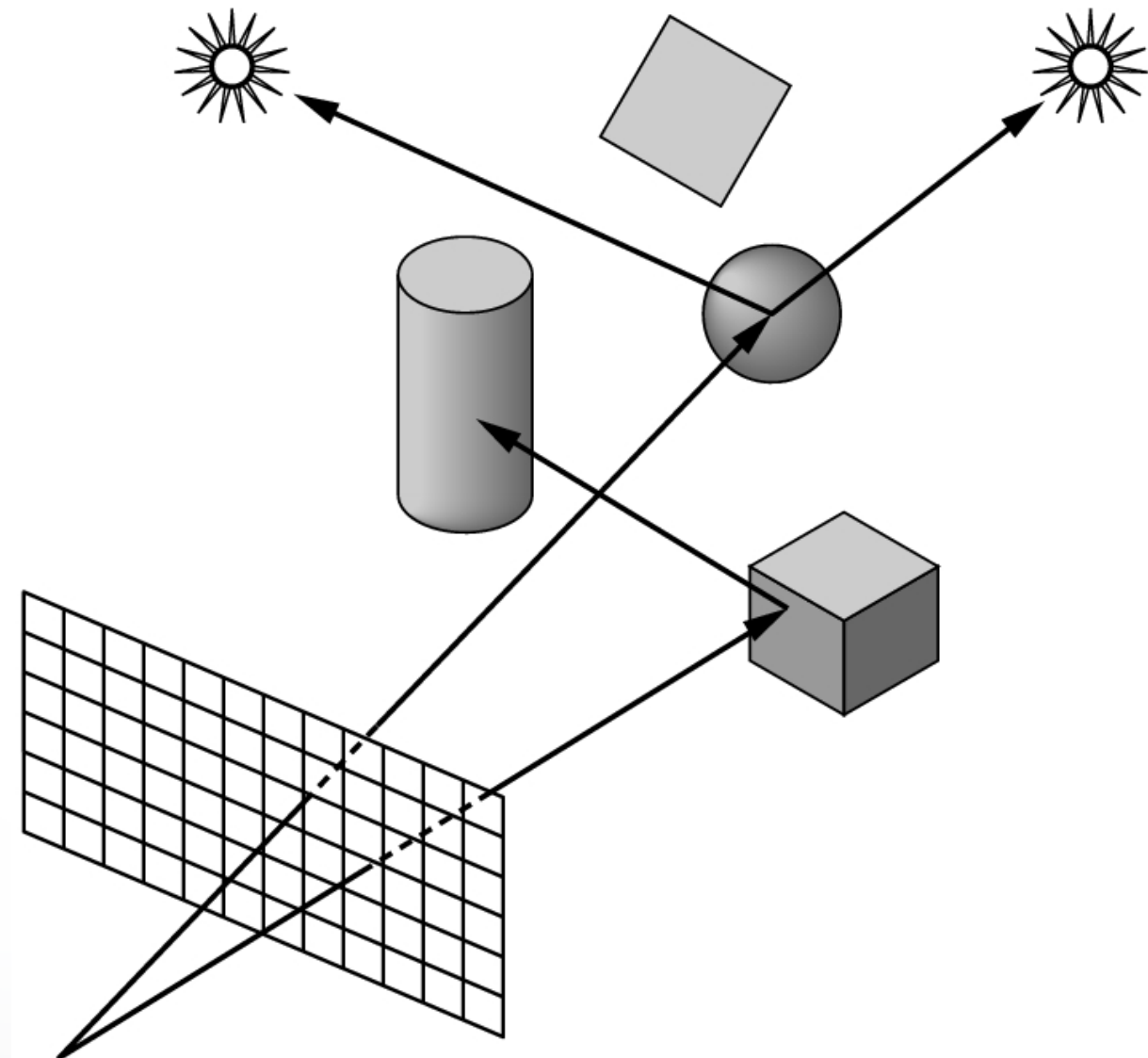
- If shadow ray can reach to the light, apply a standard Phong model

$$I = L \left(k_d (l \cdot n) + k_s (r \cdot v)^\alpha \right)$$



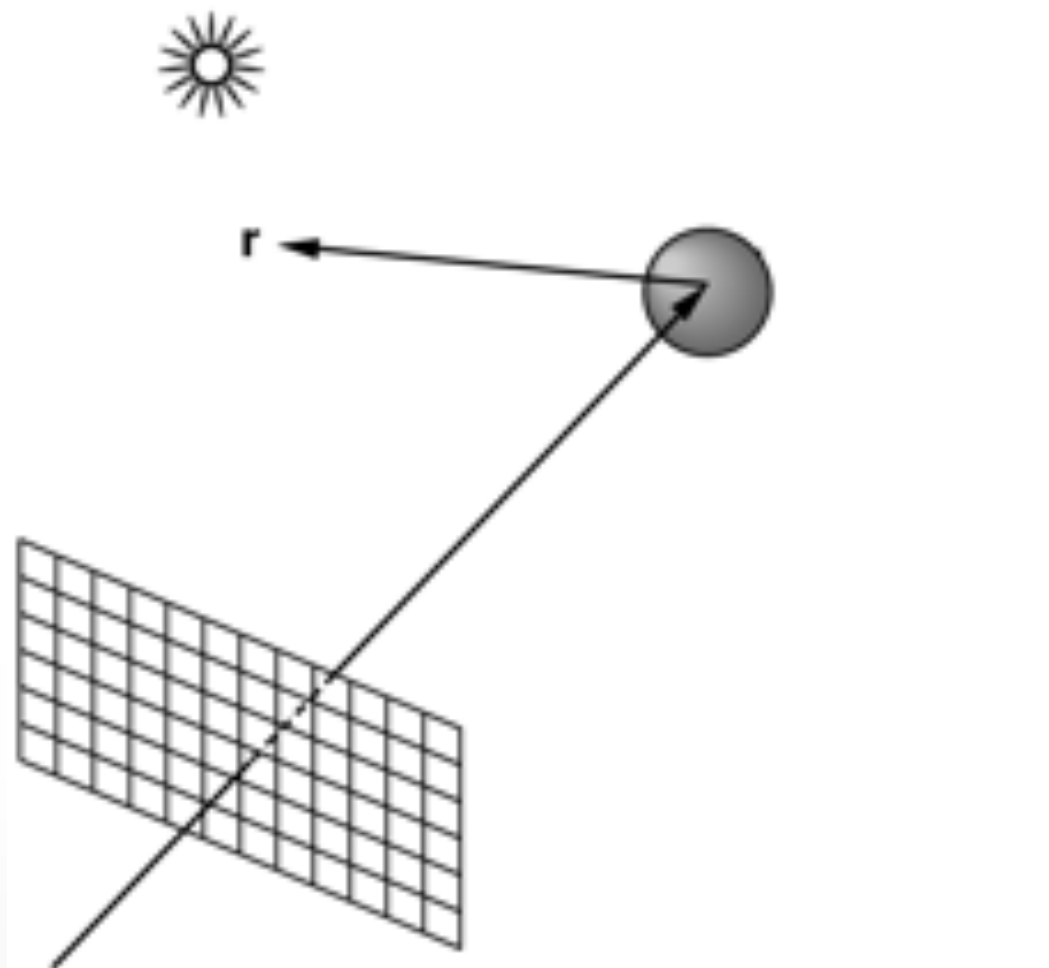
**Where is Phong model applied
in this example?**

Which shadow rays are blocked?



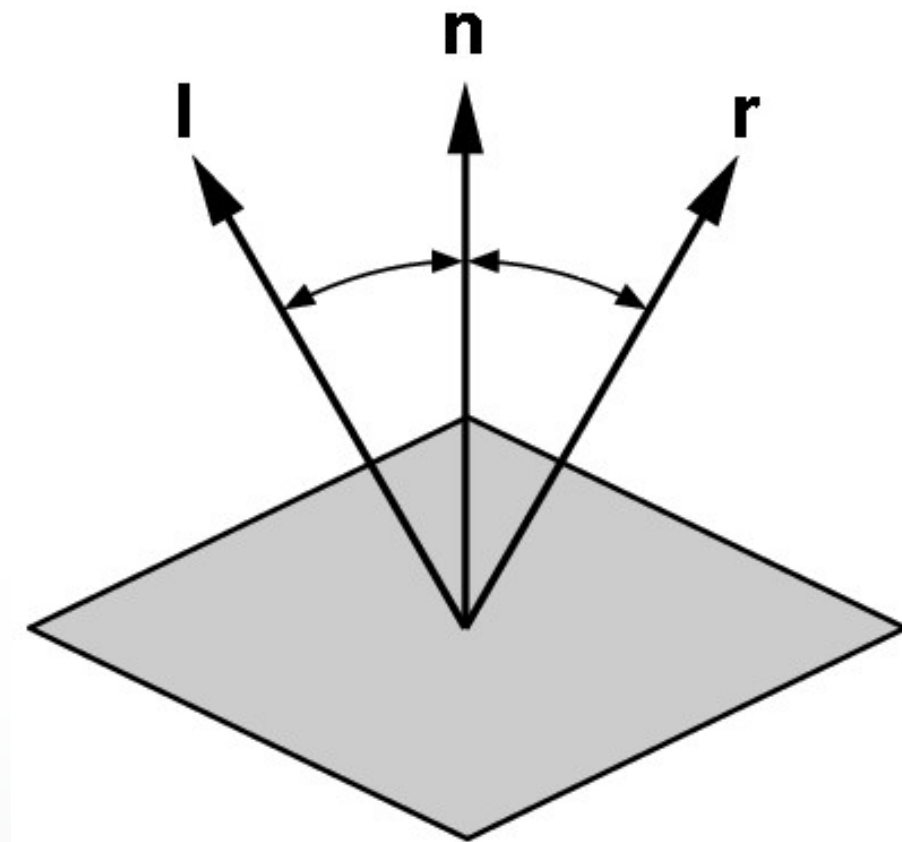
Reflection Rays

- For specular component of illumination
- Compute **reflection ray** (recall: backward!)
- Call ray tracer recursively to determine color

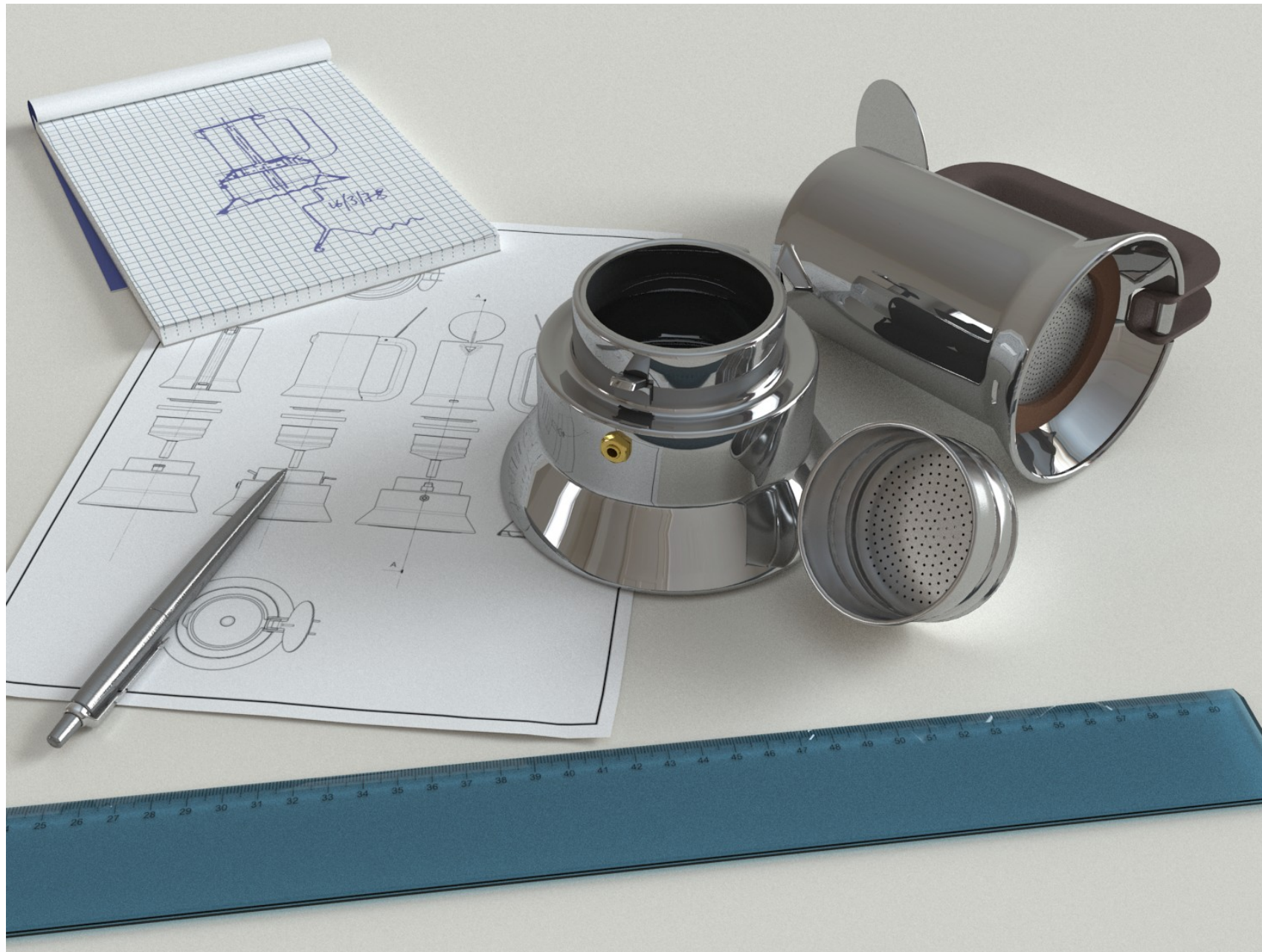


Angle of Reflection

- Recall: incoming angle = outgoing angle
- $r = 2(l \cdot n)n - l$
- Compute only for surfaces that are reflective



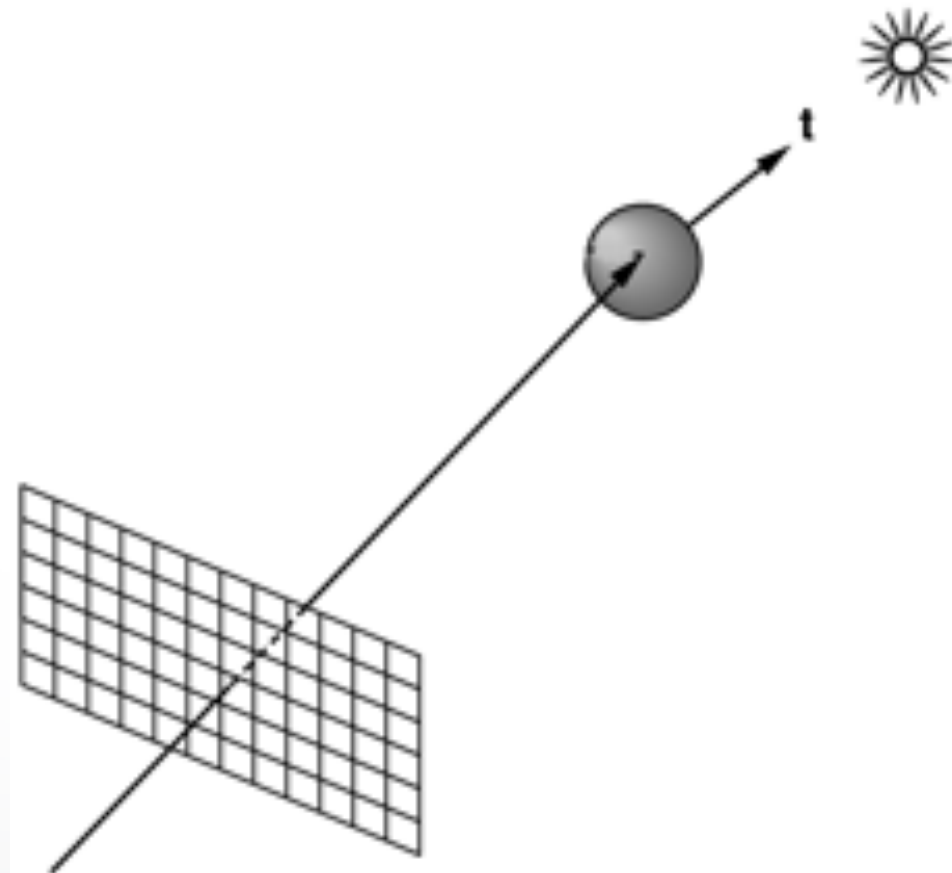
Reflections Example



www.yafaray.org

Transmission Rays

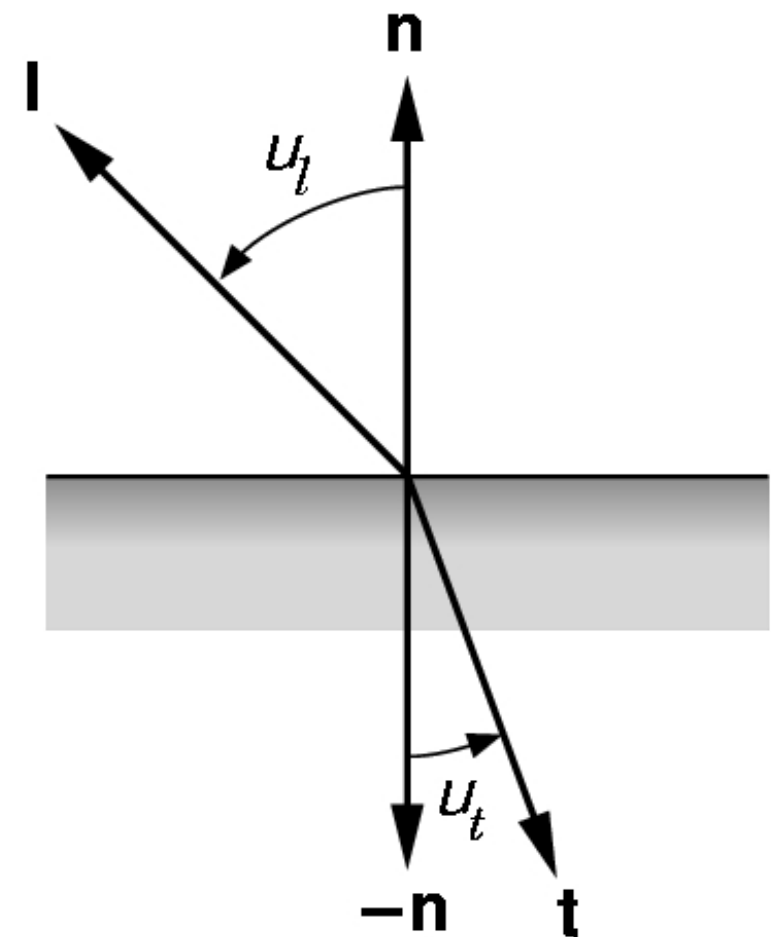
- Calculate light transmitted through surfaces
- Example: water, glass
- Compute **transmission ray**
- Call ray tracer recursively to determine color



Transmitted Light

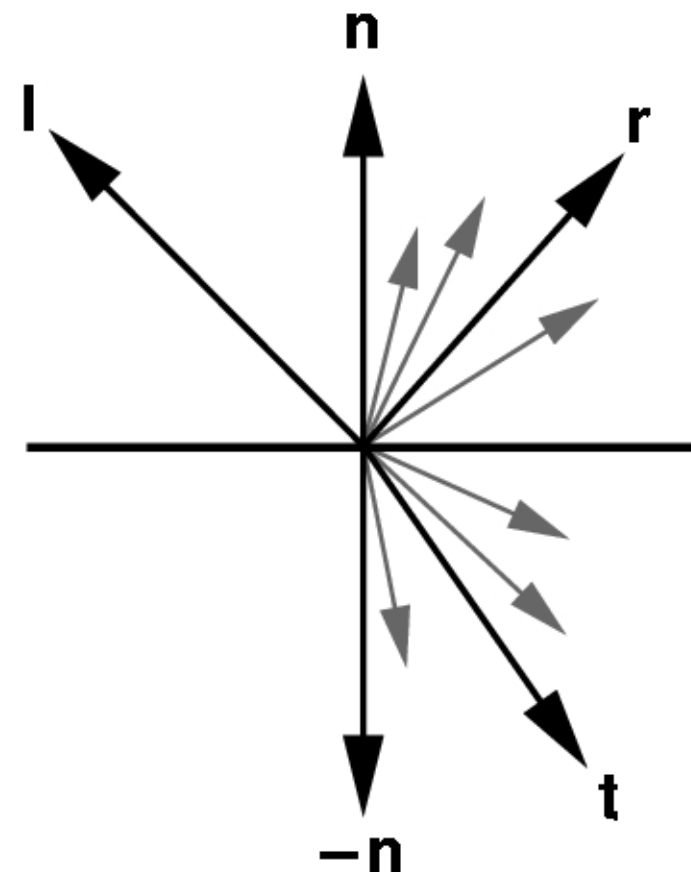
- Index of refraction is speed of light, relative to speed of light in vacuum
 - Vacuum: 1.0 (per definition)
 - Air: 1.000277 (approximate to 1.0)
 - Water: 1.33
 - Glass: 1.49
- Compute t using Snell's law
 - η_l = index for upper material
 - η_t = index for lower material

$$\frac{\sin(u_l)}{\sin(u_t)} = \frac{\eta_t}{\eta_l} = \eta$$



Translucency

- Most real objects are not transparent, but blur the background image
- Scatter light on other side of surface
- Use stochastic sampling (called distributed ray tracing)



Transmission + Translucency Example



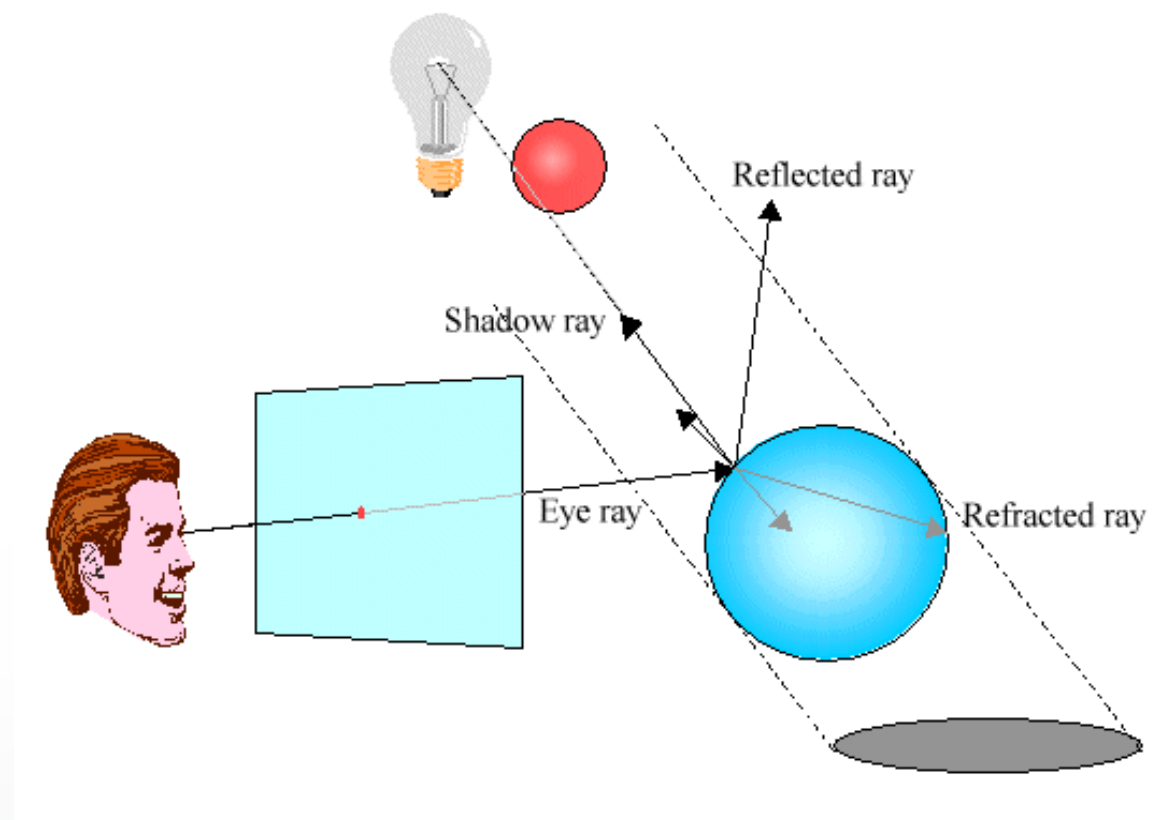
www.povray.org

The Ray Casting Algorithm

- Simplest case of ray tracing
 1. For each pixel (x,y) , fire a ray from COP through (x,y)
 2. For each ray & object, calculate closest intersection
 3. For closest intersection point **p**
 - Calculate surface normal
 - For each light source, fire shadow ray
 - For each unblocked shadow ray, evaluate local Phong model for that light, and add the result to pixel color
- Critical operations
 - Ray-surface intersections
 - Illumination calculation

Recursive Ray Tracing

- Also calculate specular component
 - Reflect ray from eye on specular surface
 - Transmit ray from eye through transparent surface
- Determine color of incoming ray by recursion
- Trace to fixed depth
- Cut off if contribution below threshold



Ray Tracing Assessment

- Global illumination method
- Image-based
- Pluses
 - Relatively accurate shadows, reflections, refractions
- Minuses
 - Slow (intersection computations)
 - Aliasing
 - Inter-object diffuse reflections require many bounces

Raytracing Example I



www.yafaray.org

Raytracing Example II



www.povray.org

Raytracing Example III



www.yafaray.org

Raytracing Example IV



www.povray.org

Summary

- Ray Casting
- Shadow Rays and Local Phong Model
- Reflection
- Transmission
- Next lecture: Geometric queries

<http://cs420.hao-li.com>

Thanks!

