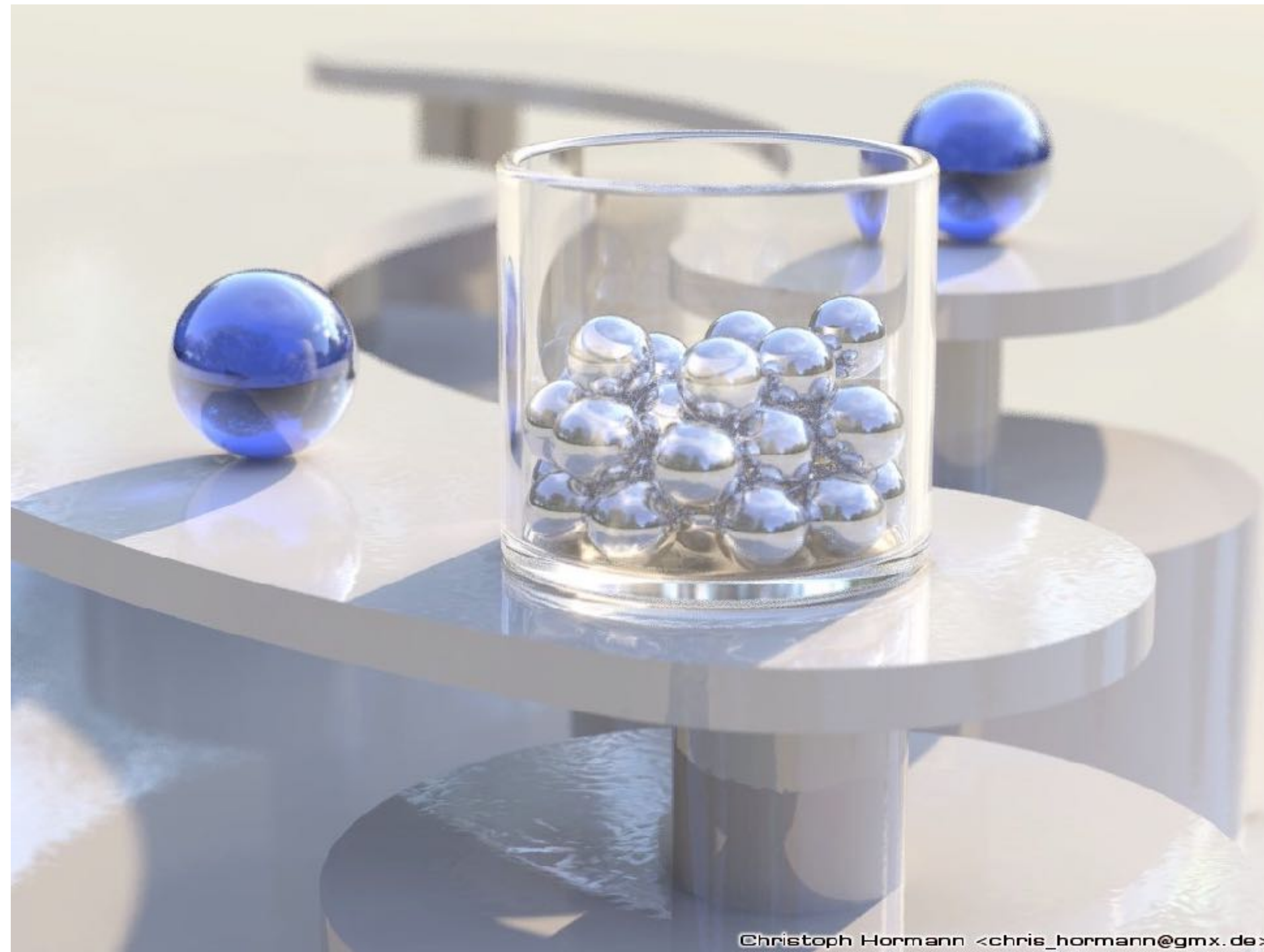


Computer Graphics

Ray Tracing

Subodh Kumar, Dept of Computer Sc. & Engg., IIT Delhi

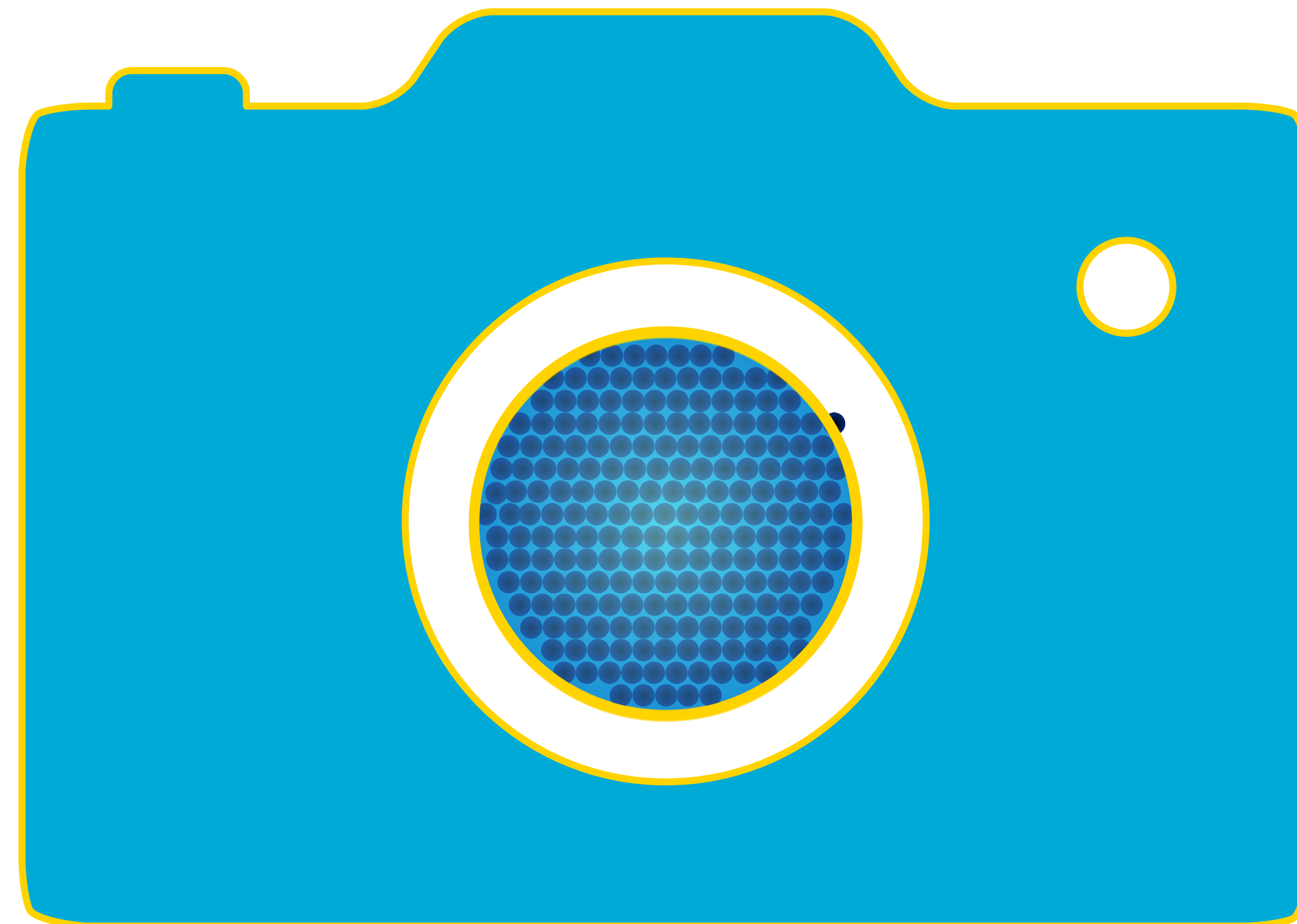
Computer Generated Image



Christoph Hormann <chris_hormann@gmx.de>

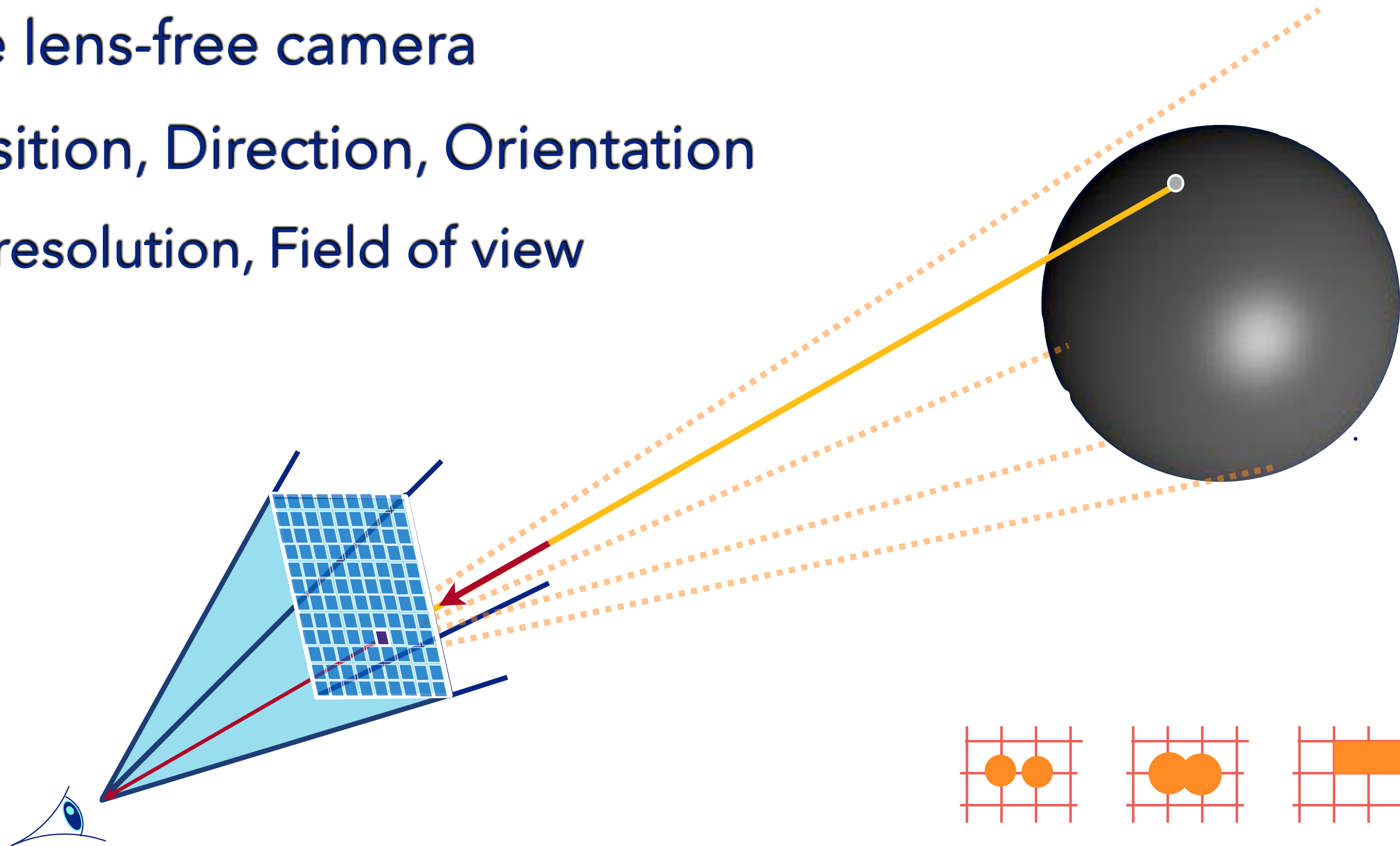
Courtesy POV-Ray gallery

Camera

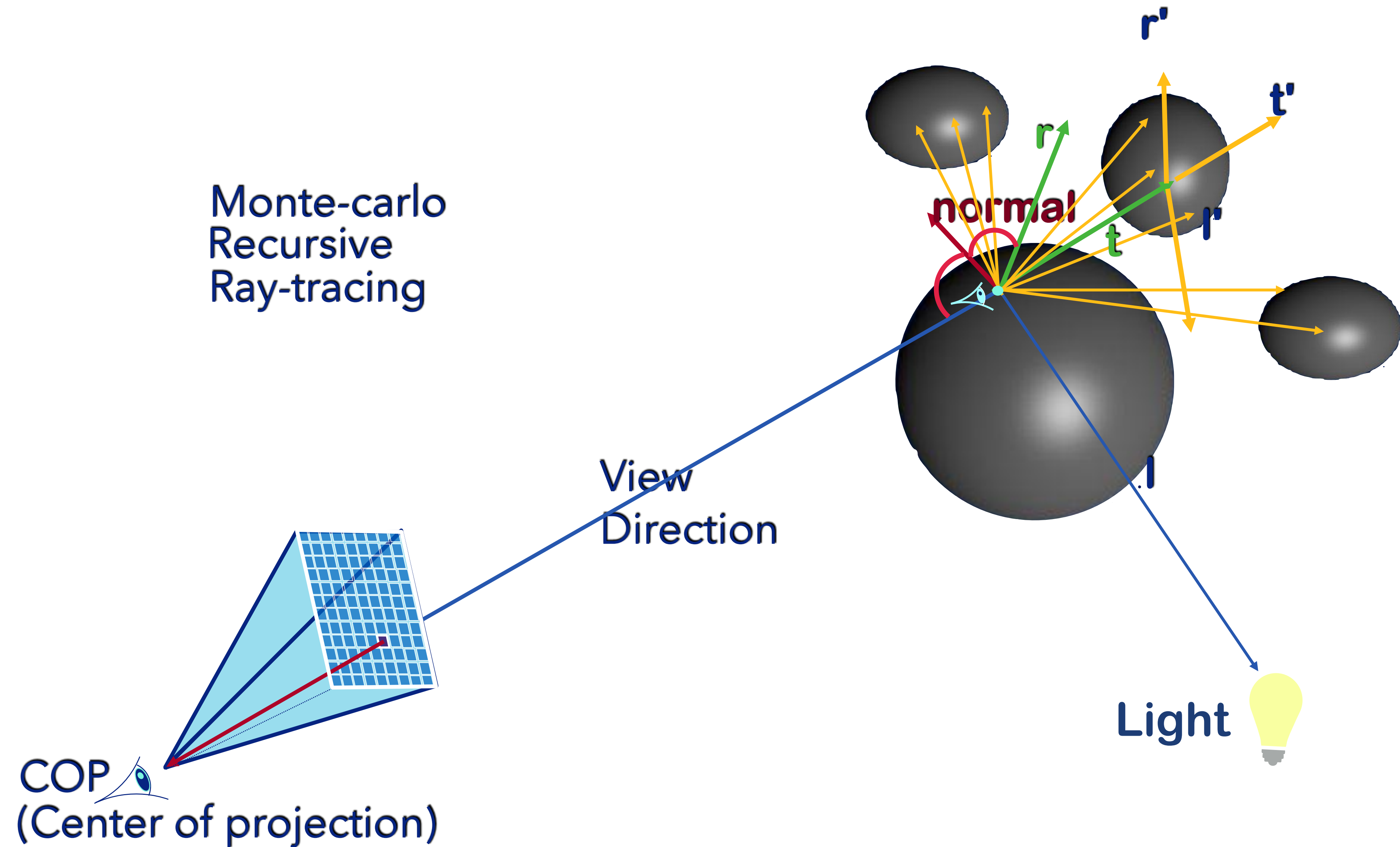


Camera Model

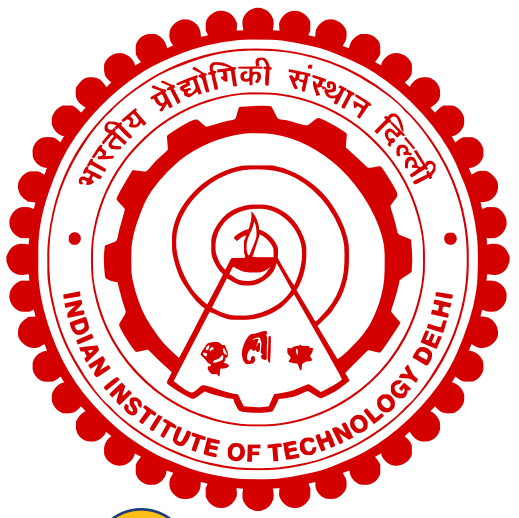
- Pinhole lens-free camera
- Has Position, Direction, Orientation
 - ➔ Pixel resolution, Field of view



Rays of Light



Basic Ray-casting



- For each pixel

- ➡ Form COP → Pixel Ray (compute color: RGB)

- ➡ Find first hit of Ray ⇒ For each object

- Find point of intersection between Ray and object
 - If closest yet, save point, object

- ➡ Compute color (Shade Ray) ⇒ If no saved intersection, return null color

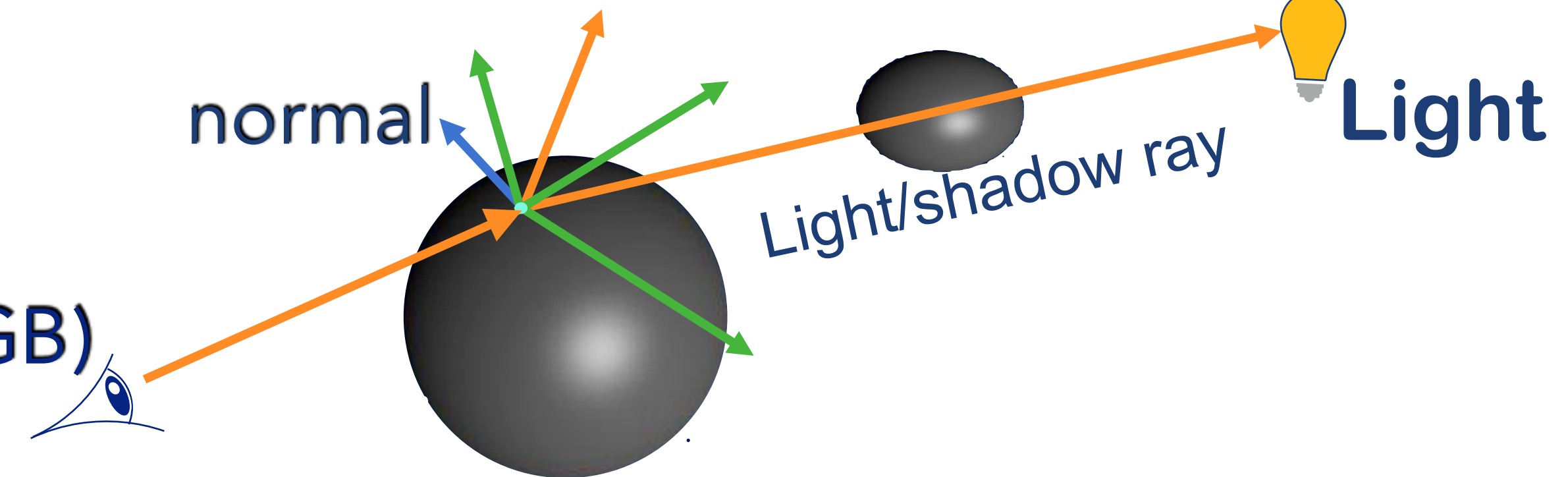
- Otherwise, shade (as per saved object) ⇒

- For each light:

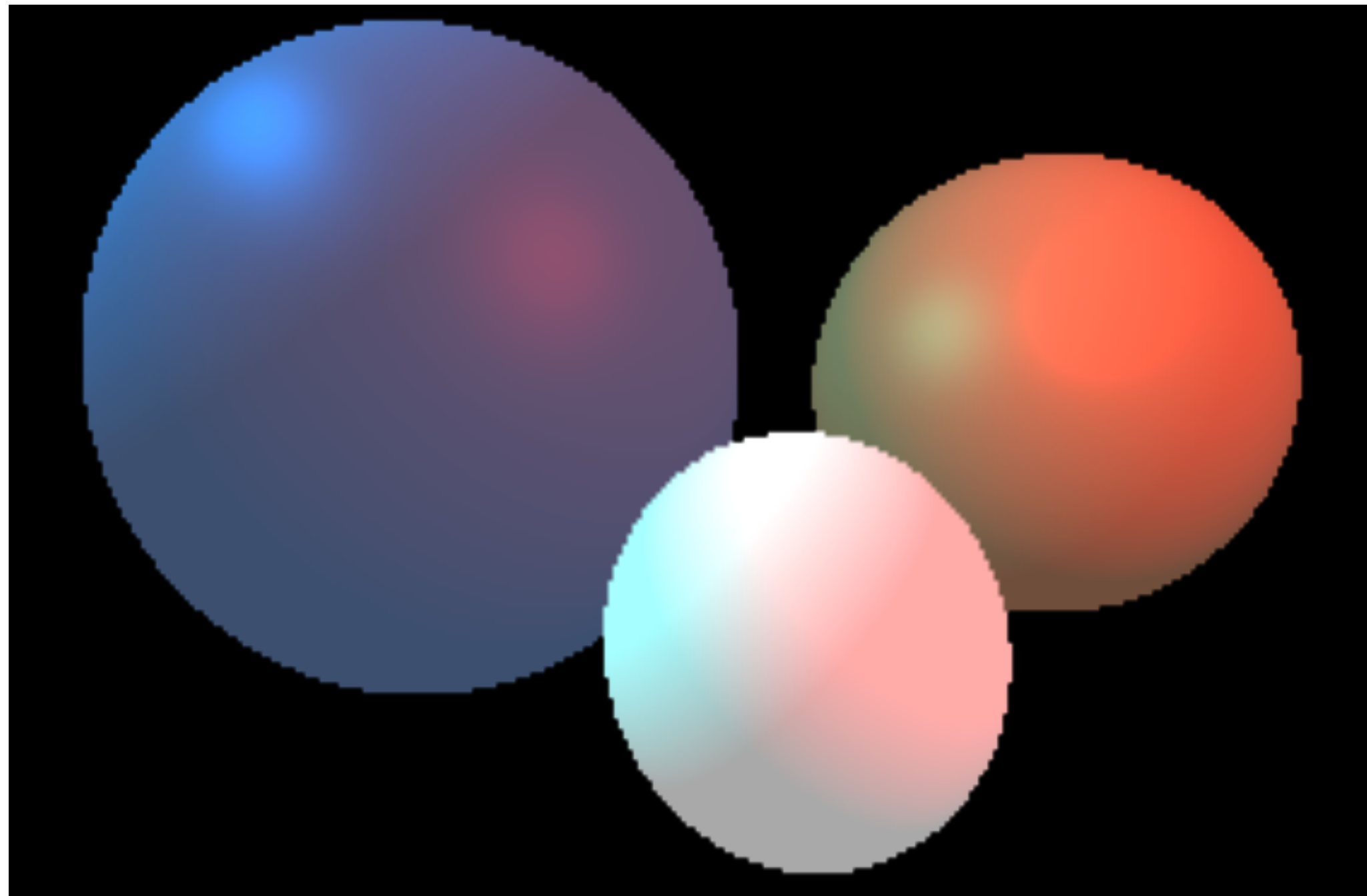
- Form "light/shadow" ray: point → light

- If first intersection == light

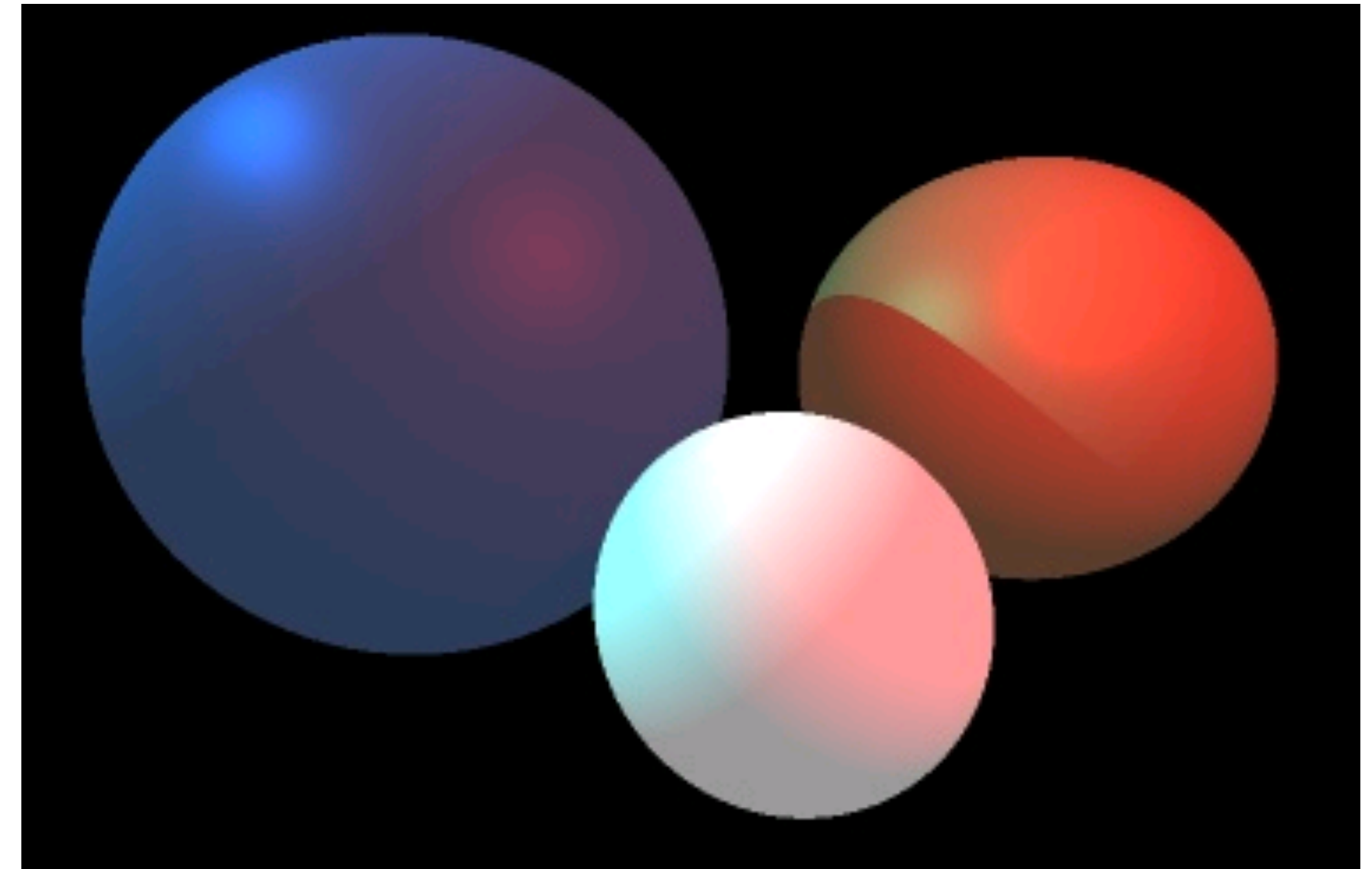
- Shade object and accumulate color



Ray-cast picture

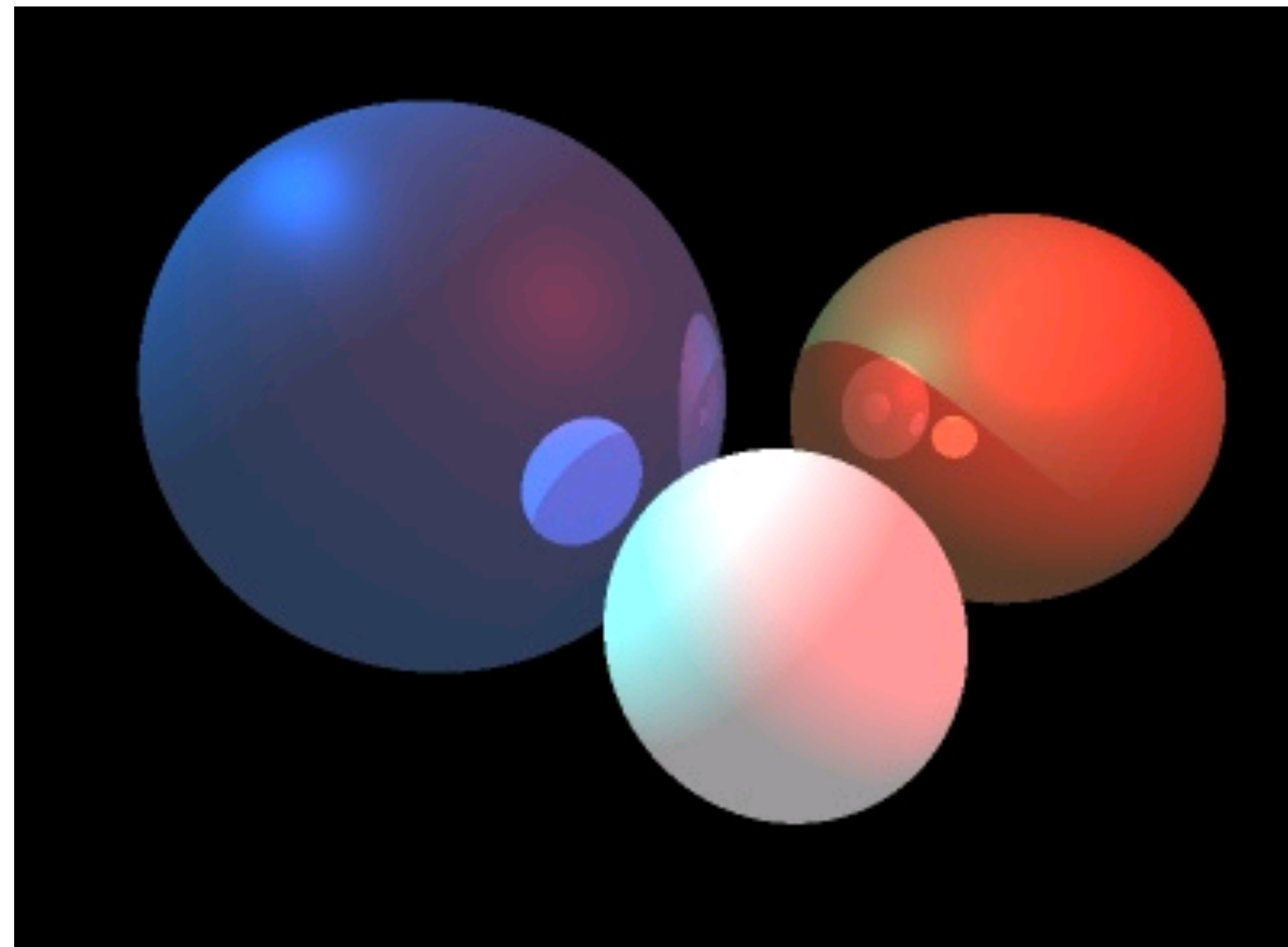


Without Shadow

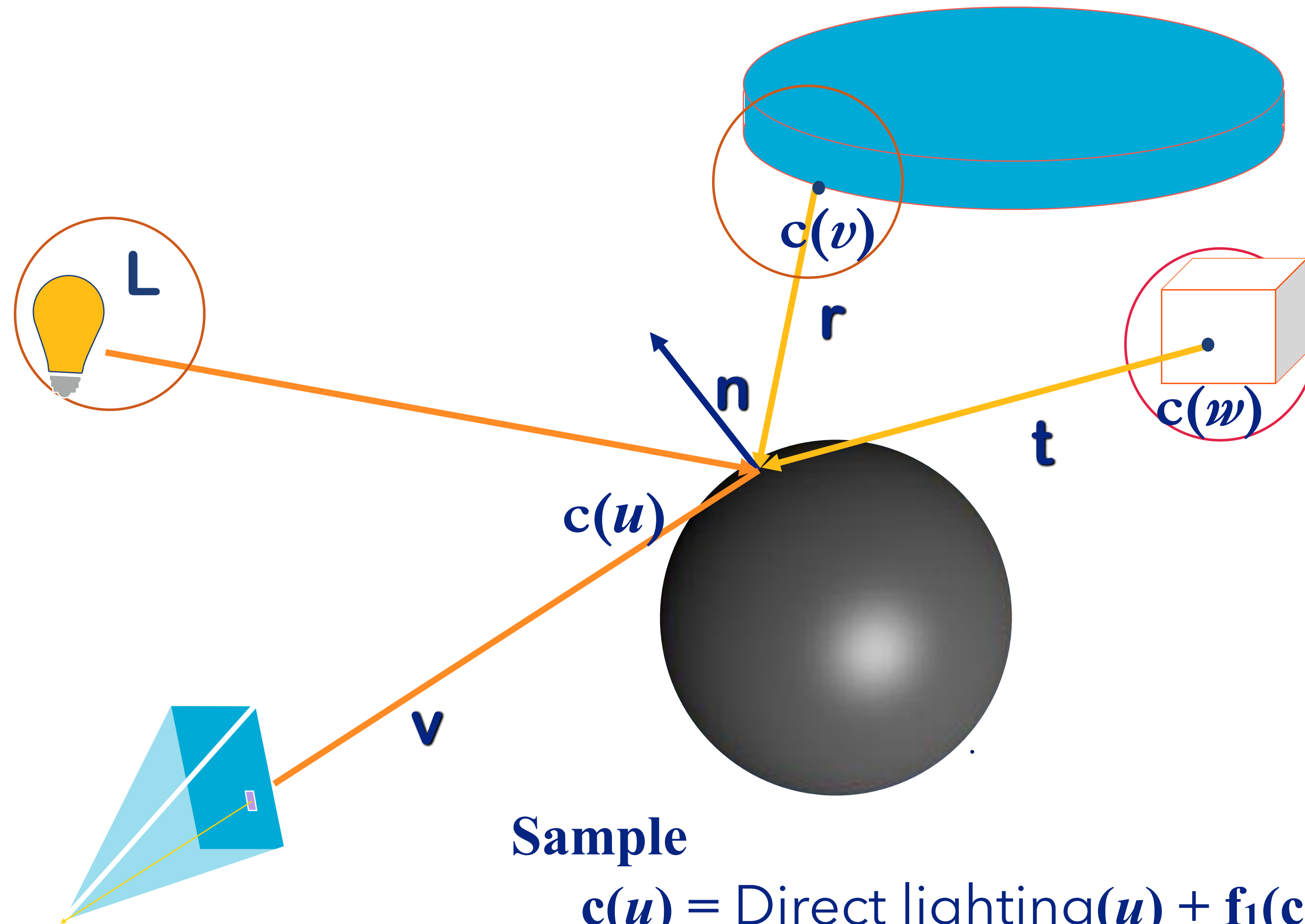


With Shadow

Recursive Ray Tracing



Recursive Ray Tracing





Recursive Ray Tracing

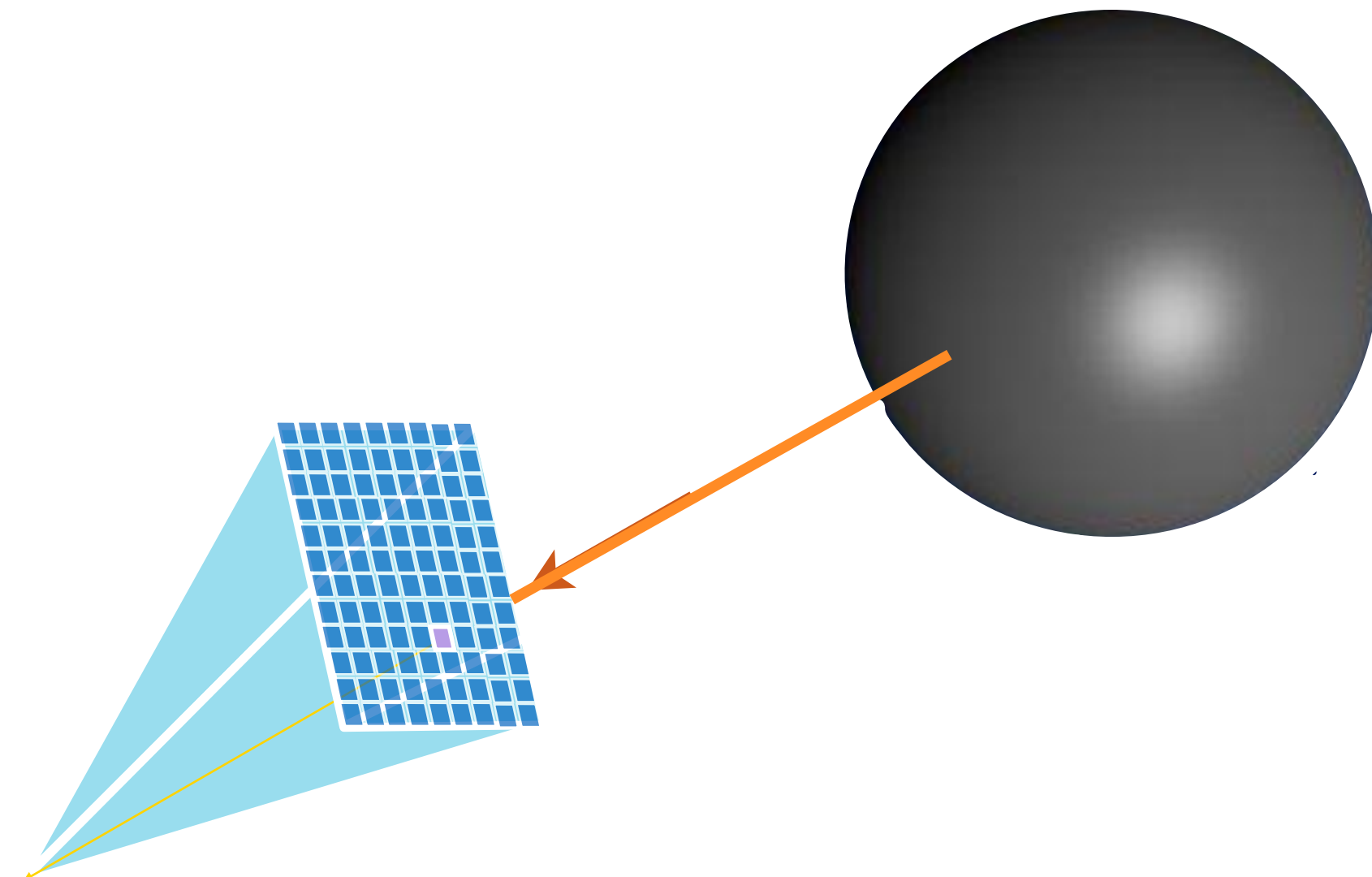
```
Color RayTrace(Ray &ray, int depth)
{
    Point IntersectionPoint, LightPoint;
    if (depth > MAX) return nullColor;
    if(! Intersect(&IntersectionPoint, ray) return Background;
    shadowRay = FormRay(IntersectionPoint, Light);
    if !IntersectAtLight(shadowRay, Light)  $I_{local} = 0$ 
    else  $I_{local} = k_a I_a + I_p(k_d(n \cdot l) + k_s(h \cdot n)^m)$  // Approximate direct lighting at intersection
        point
    return  $I_{local} + k_r * \text{RayTrace}(\text{Reflect}(\text{ray}), \text{depth}+1) + k_t * \text{RayTrace}(\text{Refract}(\text{ray}), \text{depth}+1);$ 
}
```



Per Ray vs Per object Rendering

- For each pixel
 - ➔ Form COP → Pixel **Ray**
 - ➔ Find first intersection of **Ray** with any object
 - ➔ Shade Ray

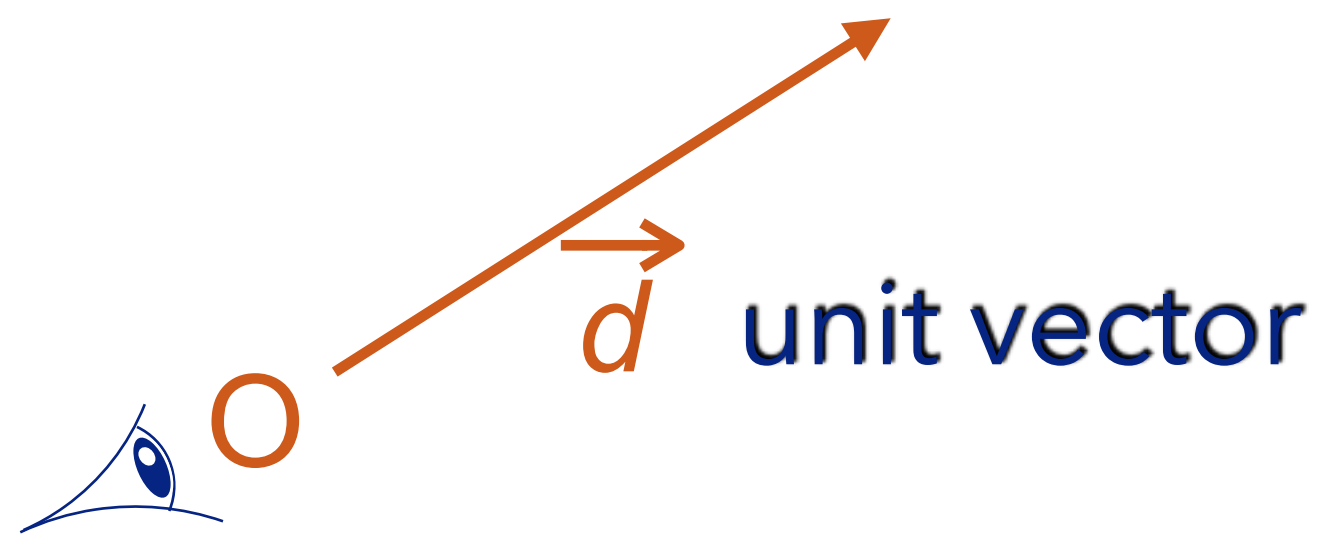
For each point on the object
Find pixel where it projects
Assign point's color to pixel



Ray Tracing Arithmetic



- Represent Ray: $P(t) = O + t \vec{d}$





The direction of reflected ray

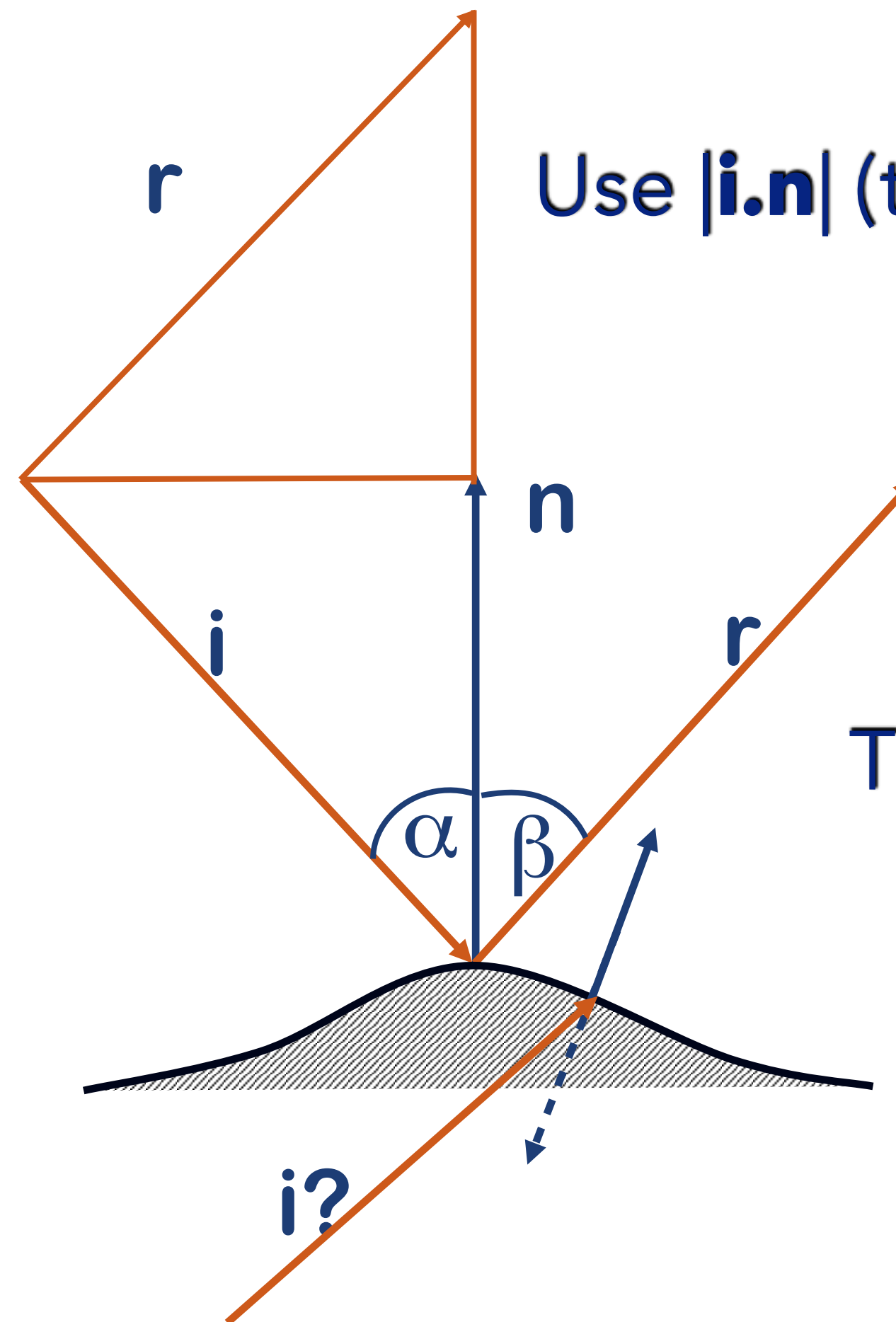
Ideal reflection

$$\alpha = \beta$$

$$-\mathbf{i} + \mathbf{r} = -2(\mathbf{i} \cdot \mathbf{n}) \mathbf{n}$$

$$\mathbf{r} = \mathbf{i} - 2(\mathbf{i} \cdot \mathbf{n}) \mathbf{n}$$

Use $|\mathbf{i} \cdot \mathbf{n}|$ (two-sided lighting) or $\min(0, \mathbf{i} \cdot \mathbf{n})$



There are two surface normals
Usually only one is specified

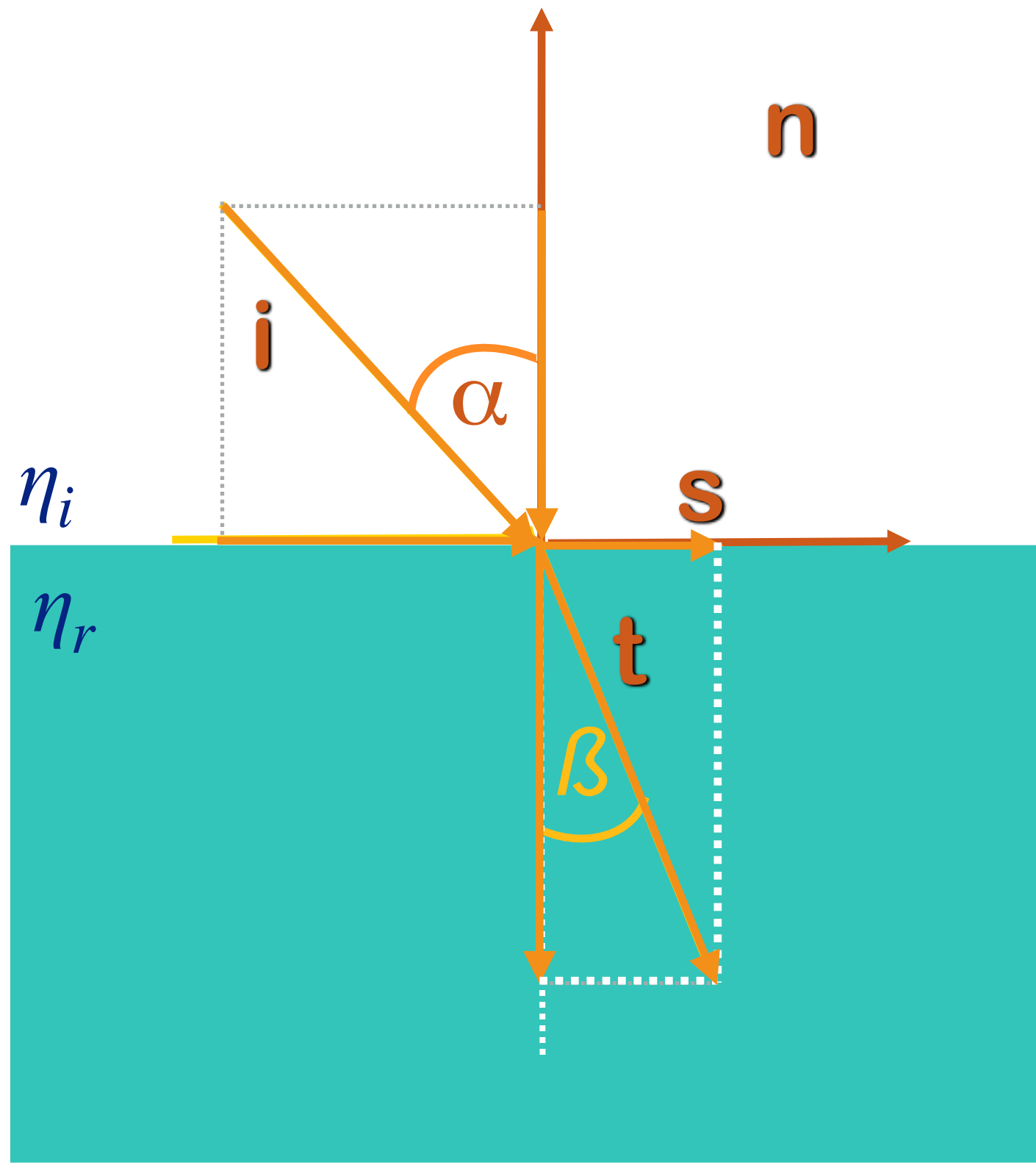


Refraction

Snell's Law:

$$\frac{\sin \alpha}{\sin \beta} = \eta_{i \rightarrow r} = \frac{\eta_r}{\eta_i}$$

*index of
refraction*



$$\mathbf{t} = \sin \beta \mathbf{s} - \cos \beta \mathbf{n}$$

$$\mathbf{i} = \sin \alpha \mathbf{s} - \cos \alpha \mathbf{n}$$

$$\mathbf{s} = \frac{\mathbf{i} + \cos \alpha \mathbf{n}}{\sin \alpha}$$

$$\mathbf{t} = \sin \beta \left(\frac{\mathbf{i} + \cos \alpha \mathbf{n}}{\sin \alpha} \right) - \cos \beta \mathbf{n}$$

$$= \frac{\sin \beta}{\sin \alpha} (\mathbf{i} + \cos \alpha \mathbf{n}) - \sqrt{1 - \sin^2 \beta} \mathbf{n}$$

$$= \eta_{r \rightarrow i} (\mathbf{i} + \cos \alpha \mathbf{n}) - \sqrt{1 - \eta_{r \rightarrow i}^2 \sin^2 \alpha} \mathbf{n}$$

$$= \eta_{r \rightarrow i} \mathbf{i} - \left(\eta_{r \rightarrow i} \mathbf{n} \cdot \mathbf{i} + \sqrt{1 - \eta_{r \rightarrow i}^2 (1 - (\mathbf{n} \cdot \mathbf{i})^2)} \right) \mathbf{n}$$



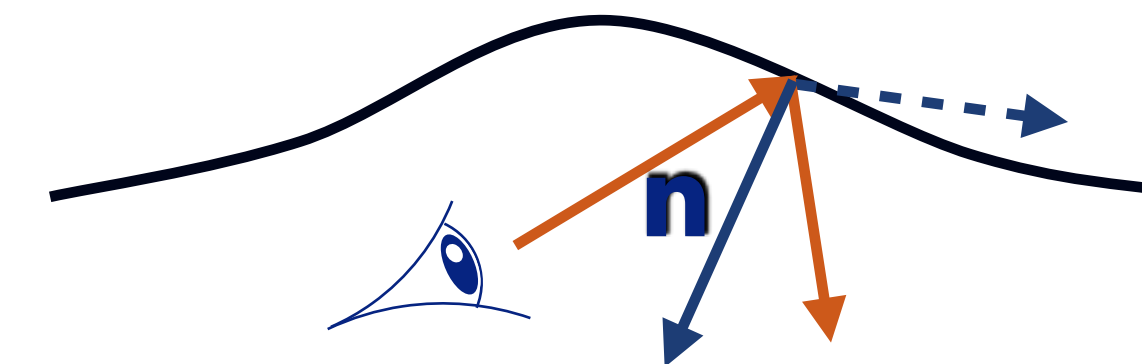
Total Internal Reflection

$$\mathbf{t} = \eta_{r \rightarrow i} \mathbf{i} - \left(\eta_{r \rightarrow i} \mathbf{n} \cdot \mathbf{i} + \sqrt{1 - \eta_{r \rightarrow i}^2 (1 - (\mathbf{n} \cdot \mathbf{i})^2)} \right) \mathbf{n}$$

Negative root => Total internal reflection

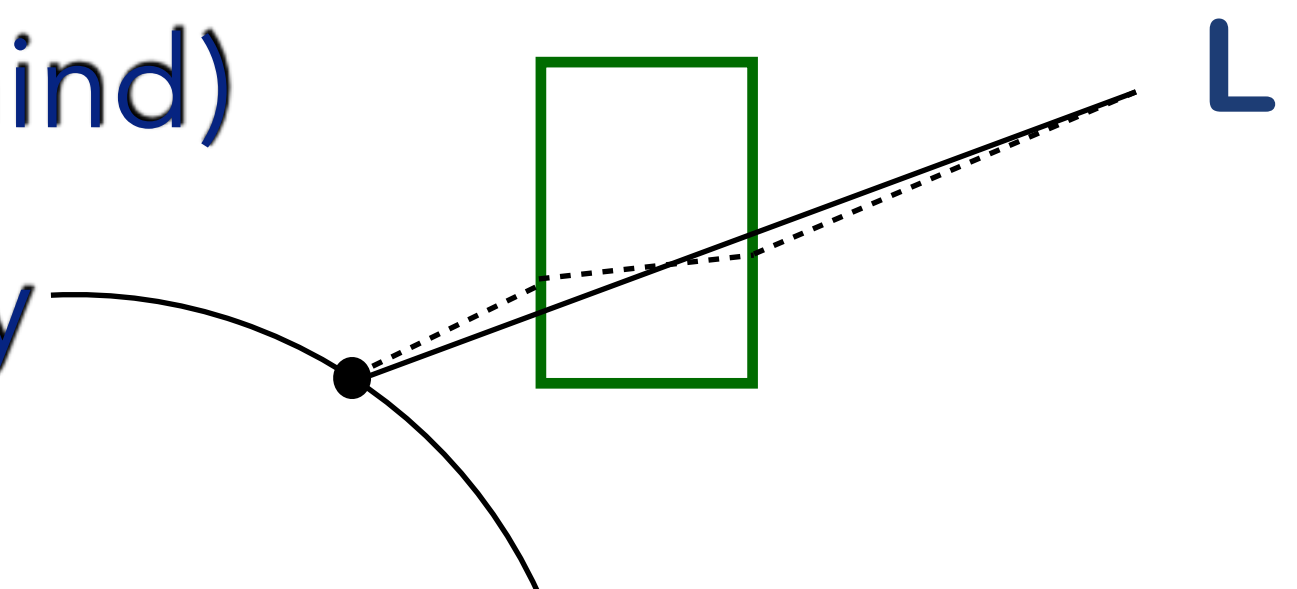
$$[\mathbf{t} = \mathbf{i} - 2(\mathbf{i} \cdot \mathbf{n})\mathbf{n}]$$

Translucent object =>

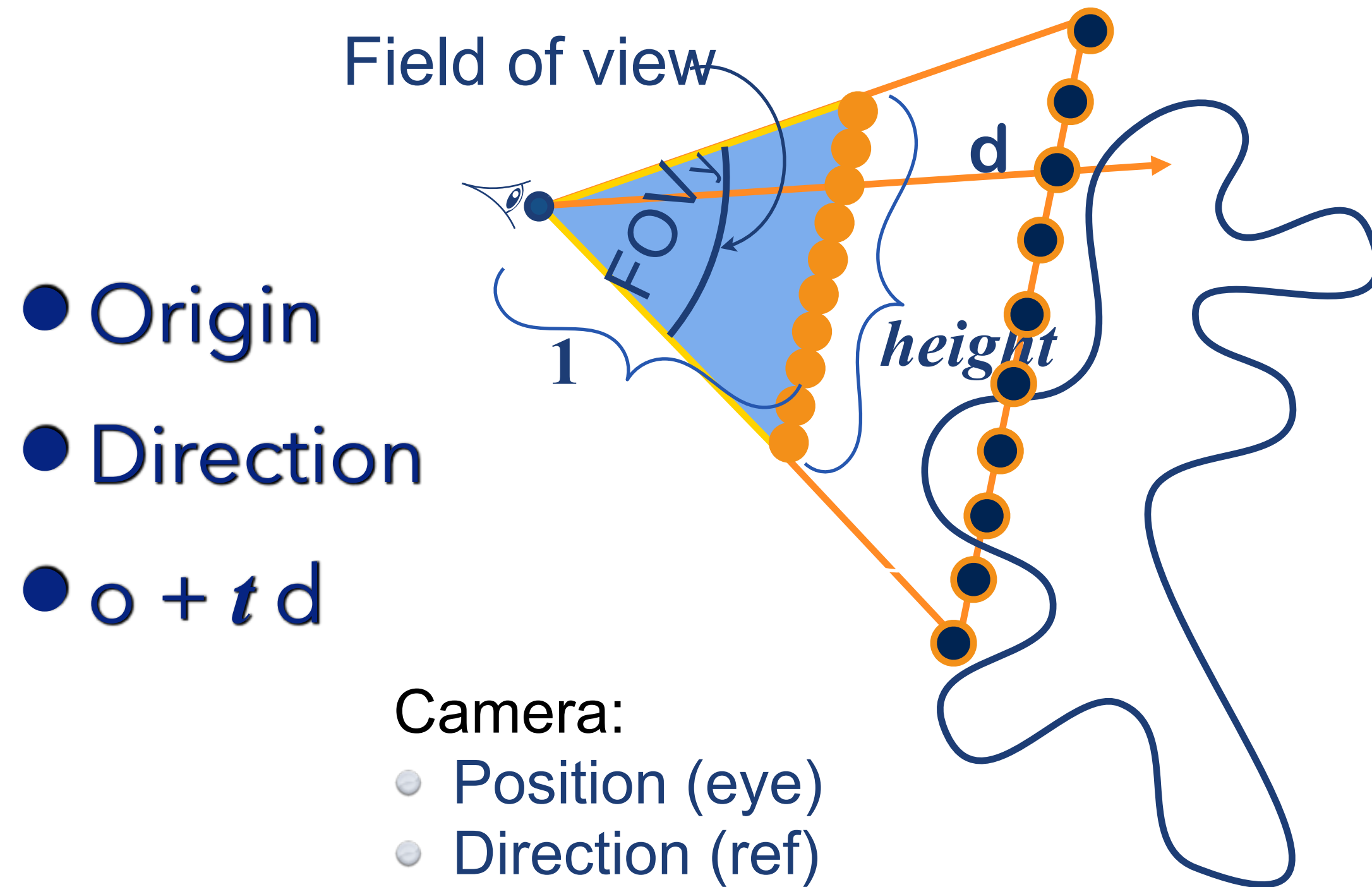


Shoot light ray also to the opposite side of normal (behind)

For translucent "shadow" caster transmits (refracted) ray



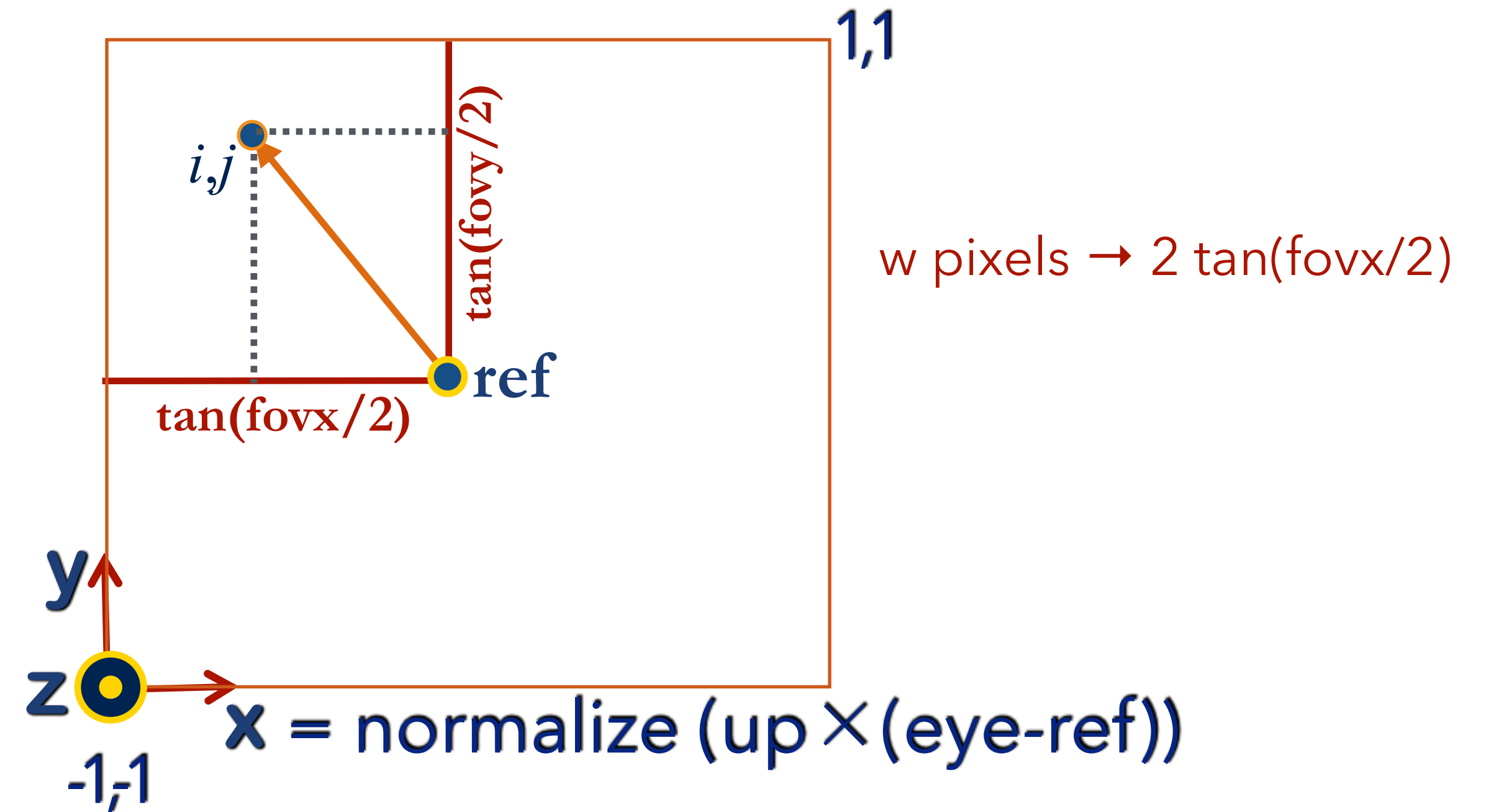
Make Eye Ray



- Origin
- Direction
- $o + t d$

Camera:

- Position (eye)
- Direction (ref)
- Head orientation (up)
- FOVx, FOVy



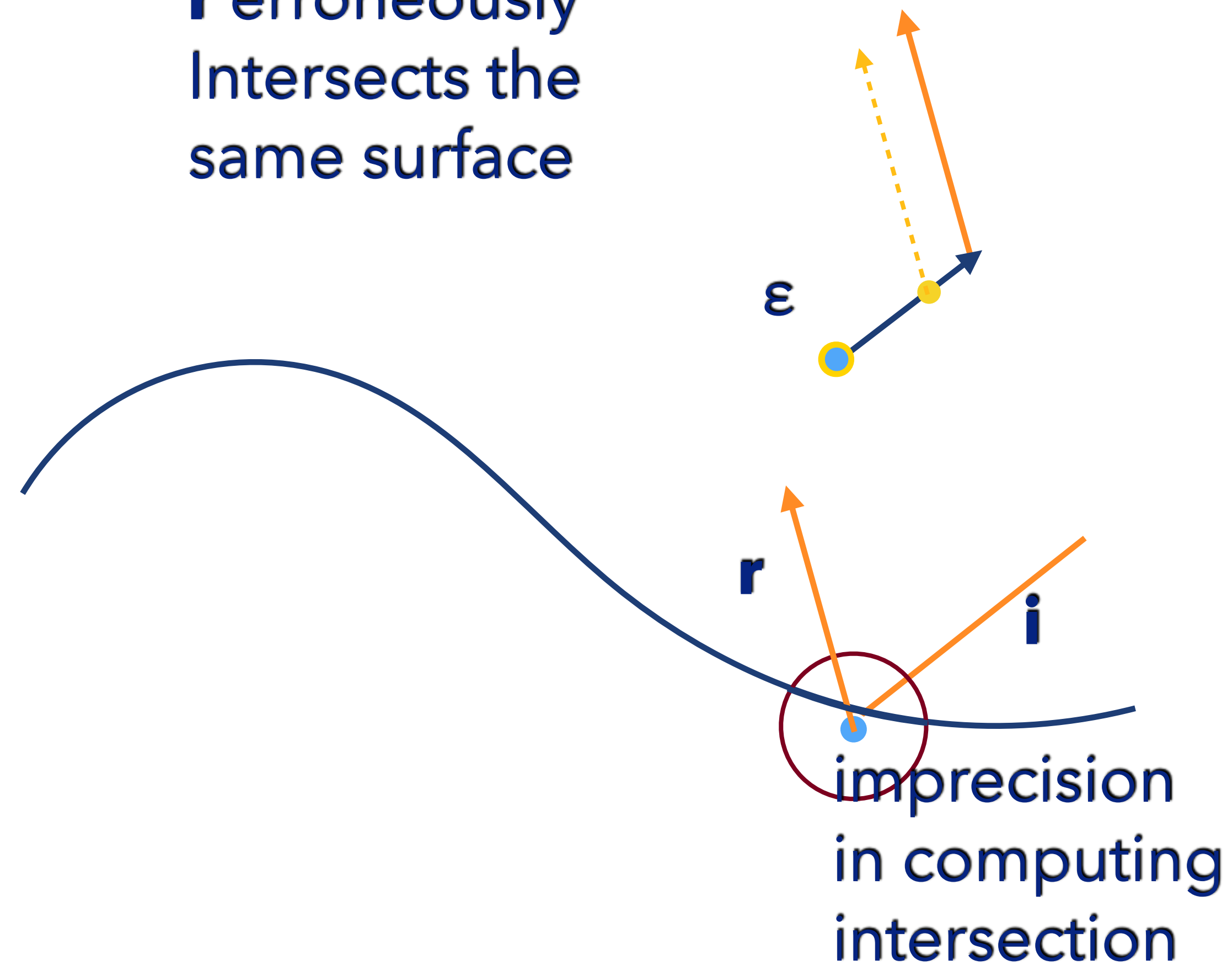
$$i, j \rightarrow \mathbf{ref} + \tan \frac{\text{fov}_x}{2} \left(\frac{2i}{\text{width}} - 1 \right) \mathbf{x} + \tan \frac{\text{fov}_y}{2} \left(\frac{2j}{\text{height}} - 1 \right) \mathbf{y}$$



Make Ray @Intersection

- Offset ray origin
- Or, offset t of intersection

r erroneously
Intersects the
same surface





Algebraic Intersection

- Point satisfies

- ⇒ $\mathbf{o} + t \mathbf{d}$

- ⇒ $f(x, y, z) = 0$

- ⇒ $f(o_x + t d_x, o_y + t d_y, o_z + t d_z) = 0$

- Example

- ⇒ $(0, 0, 5) + t(0, 0, 1)$

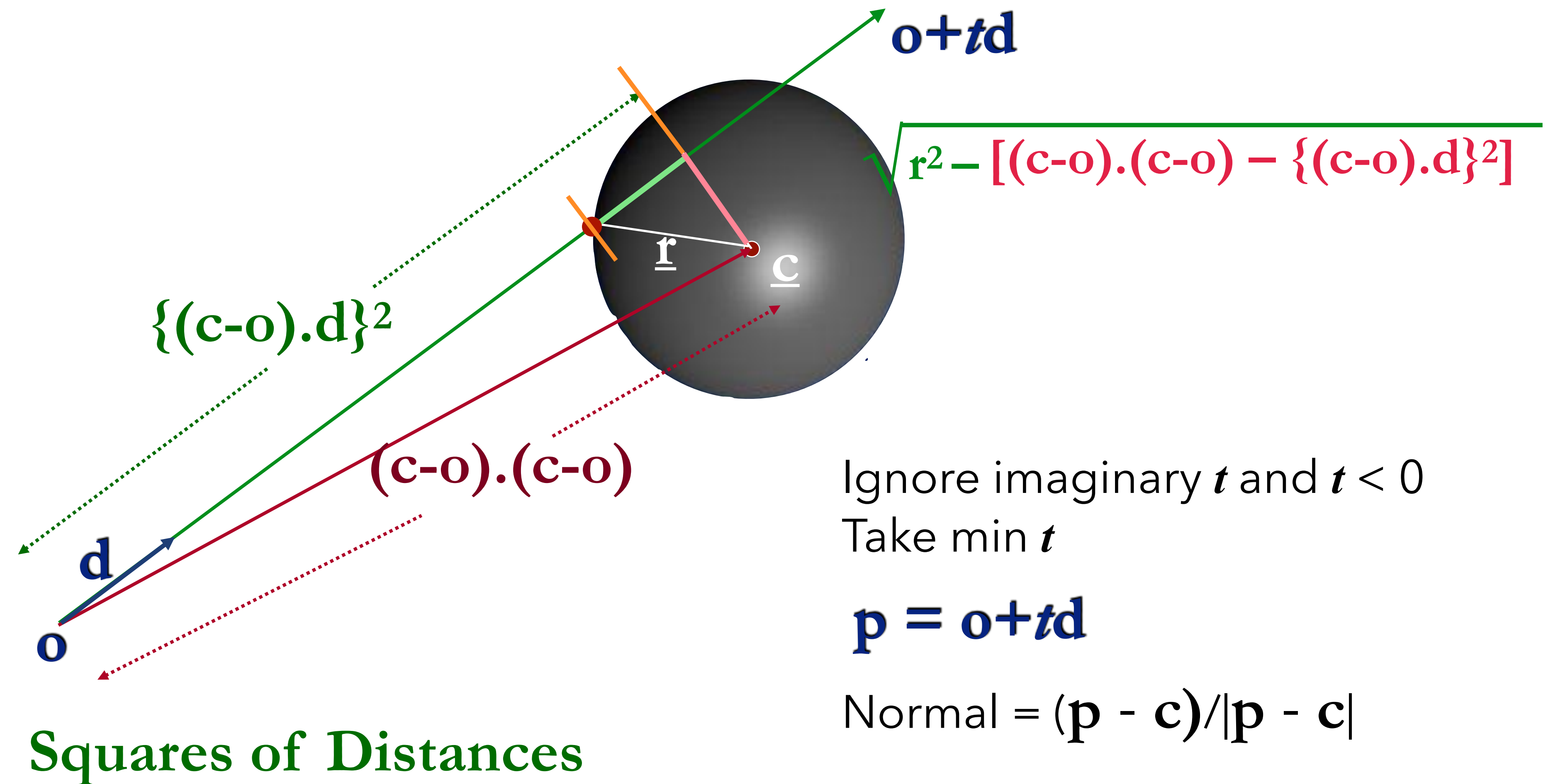
- ⇒ $x^2 + y^2 + z^2 = 1$

- ⇒ $0 + 0 + (5 + t)^2 = 1$, i.e., $t = \pm 1 - 5 = -4$ & -6

- ⇒ $(0, 0, 5) - 4(0, 0, 1) = (0, 0, 1)$



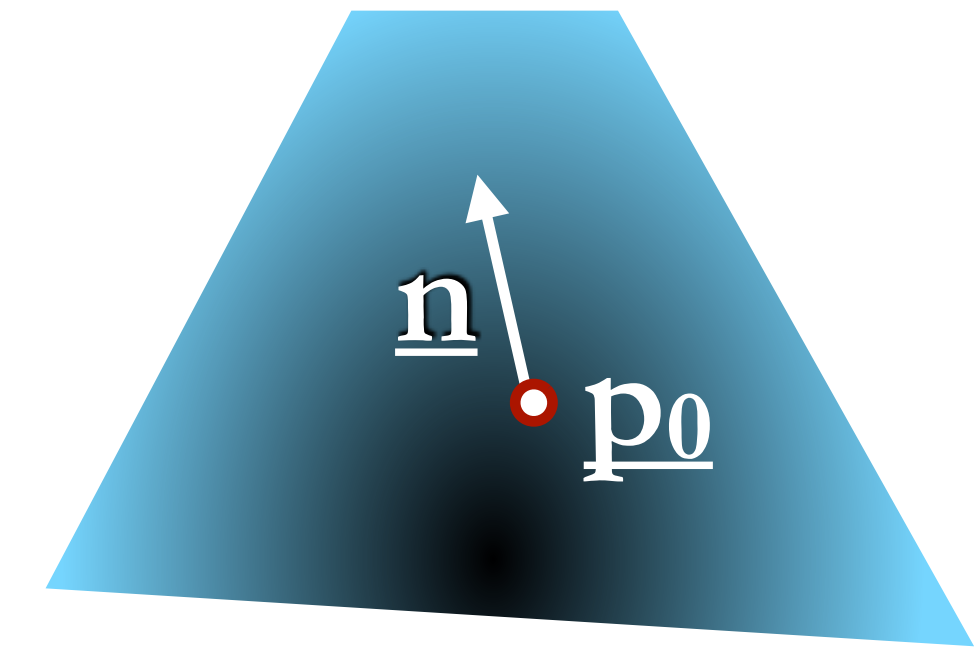
Geometric Intersection





Triangle Intersection?

- Intersect with plane
 - ➔ $Ax + By + Cz = 1$
- Check if point inside triangle (point needed first?)
 - ➔ Barycentric coordinates must be +ve
 - Recall: coordinates are ratio of areas
 - (How can a coordinate be negative then?)
 - ➔ Point must be in the positive half-planes (of edges)



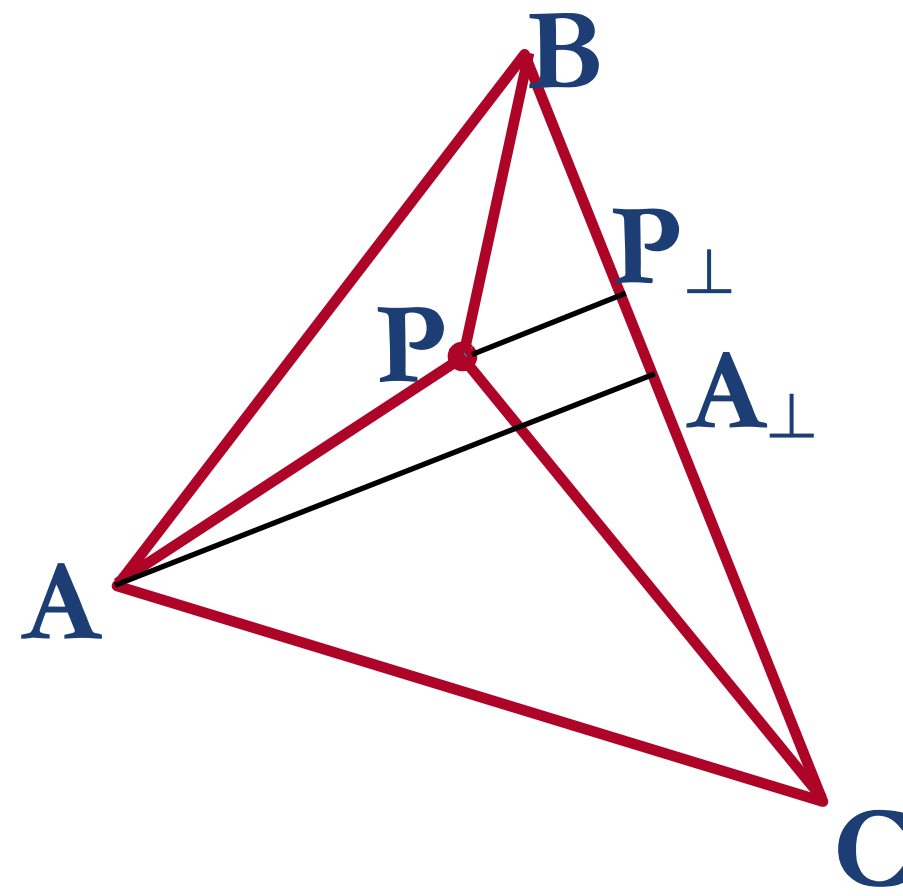
Plane equation:

$$\mathbf{n} \cdot (\mathbf{p} - \mathbf{p}_0) = 0$$



Barycentric Coordinates

$$\mathbf{P} = u\mathbf{A} + v\mathbf{B} + w\mathbf{C}$$



$$u = |\mathbf{PBC}| / |\mathbf{ABC}|$$

etc..

$$u = |\mathbf{PP}_{\perp}| / |\mathbf{AA}_{\perp}|$$

Barycentric Coordinates of $\mathbf{P} = [u, v, w]$

$u + v + w = 1$ inside the triangle



Half-plane Method

$$Ax + By + Cz = 1$$

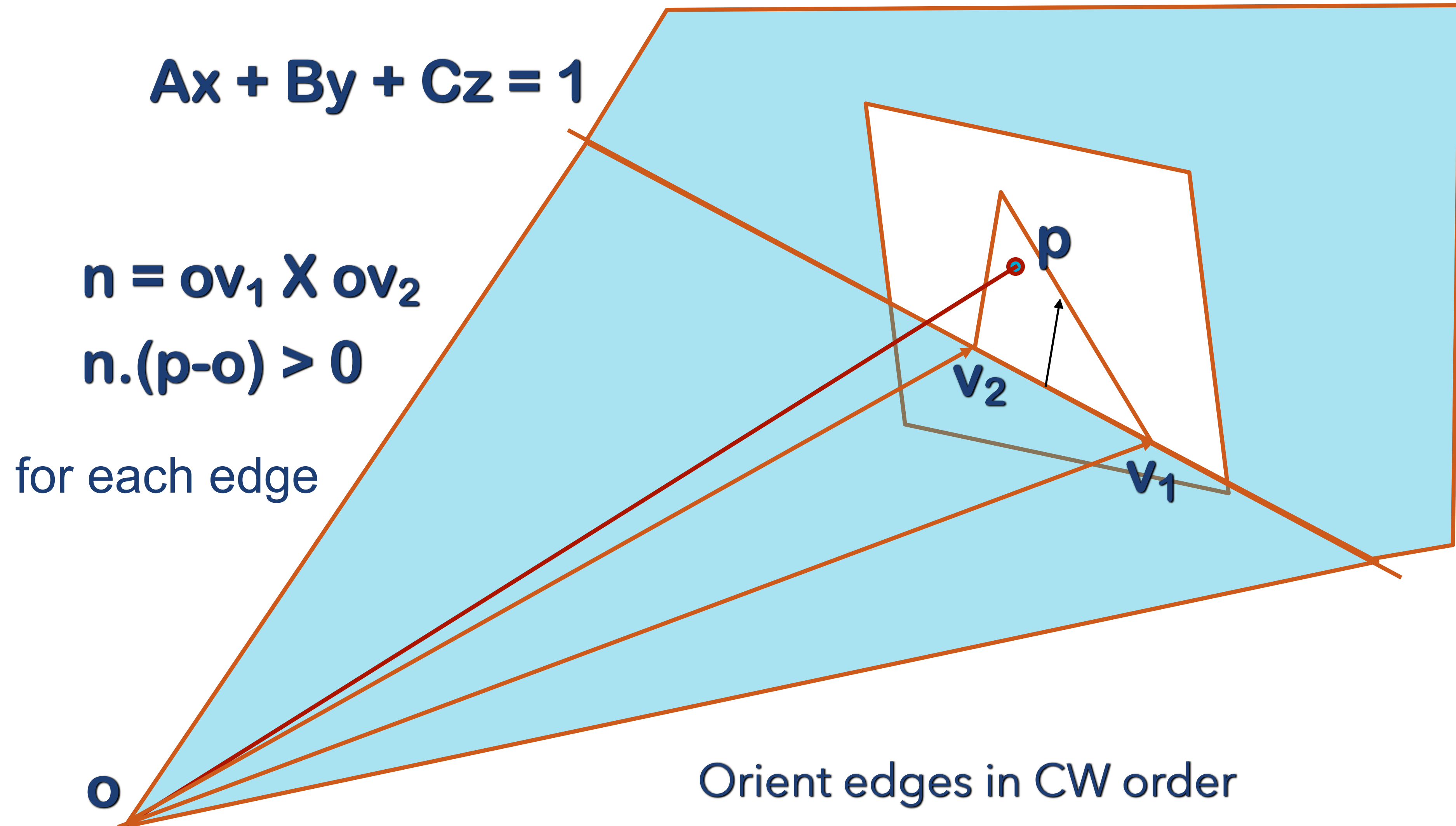
$$\mathbf{n} = \mathbf{ov}_1 \times \mathbf{ov}_2$$

$$\mathbf{n} \cdot (\mathbf{p} - \mathbf{o}) > 0$$

for each edge

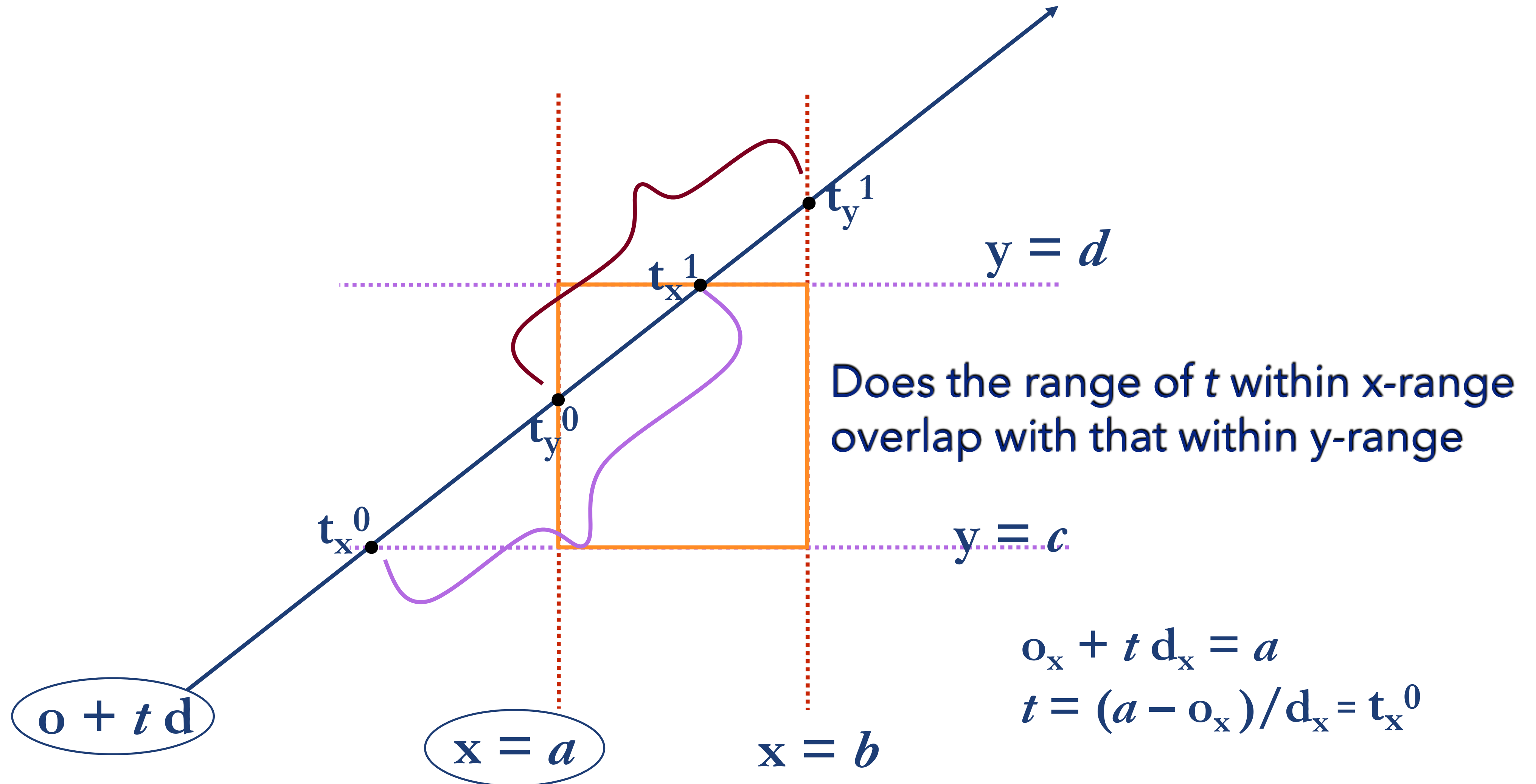
o

Orient edges in CW order





Bounding Box Test



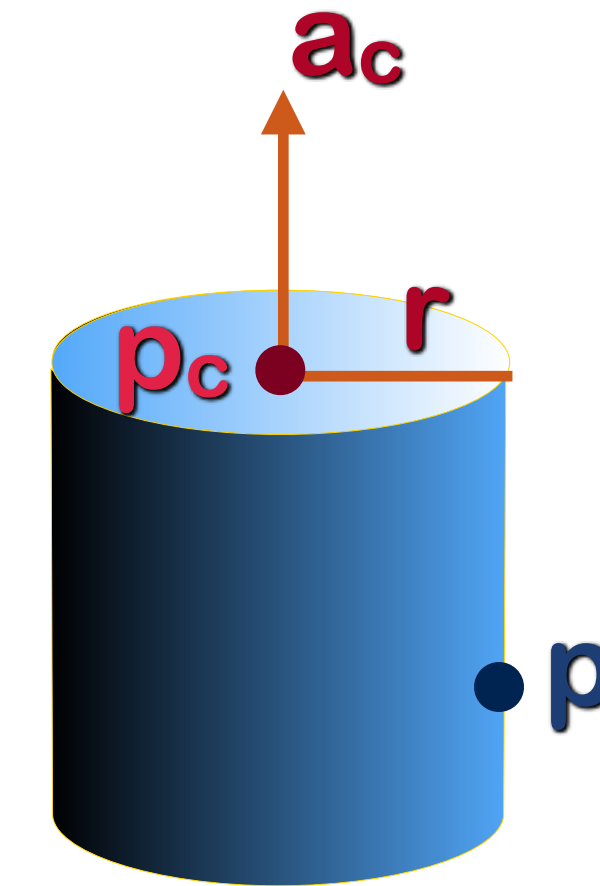


Ray-Cylinder Intersection

- Consider infinite cylinder

$$\Rightarrow x^2 + z^2 = r^2$$

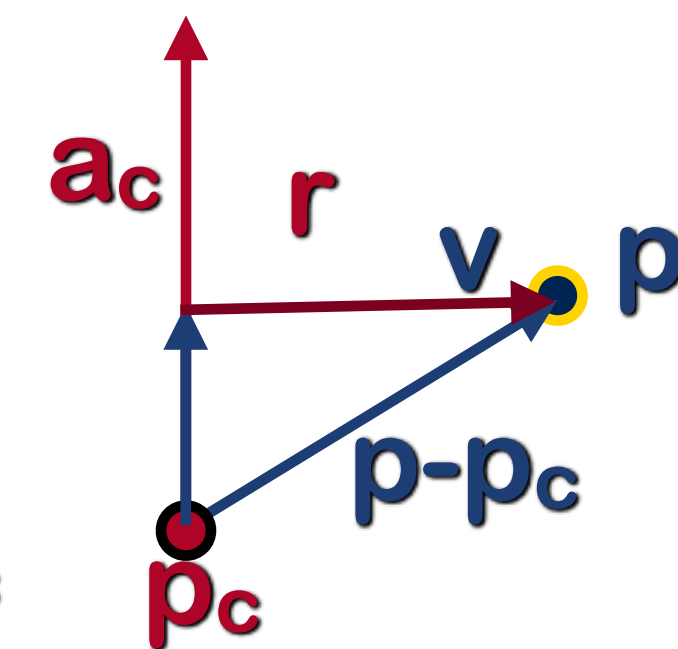
$$\Rightarrow \text{Axis} = p_c + t a_c$$



Replace p with $o + t d$

$$v = (p - p_c) - (p - p_c) \cdot a_c$$

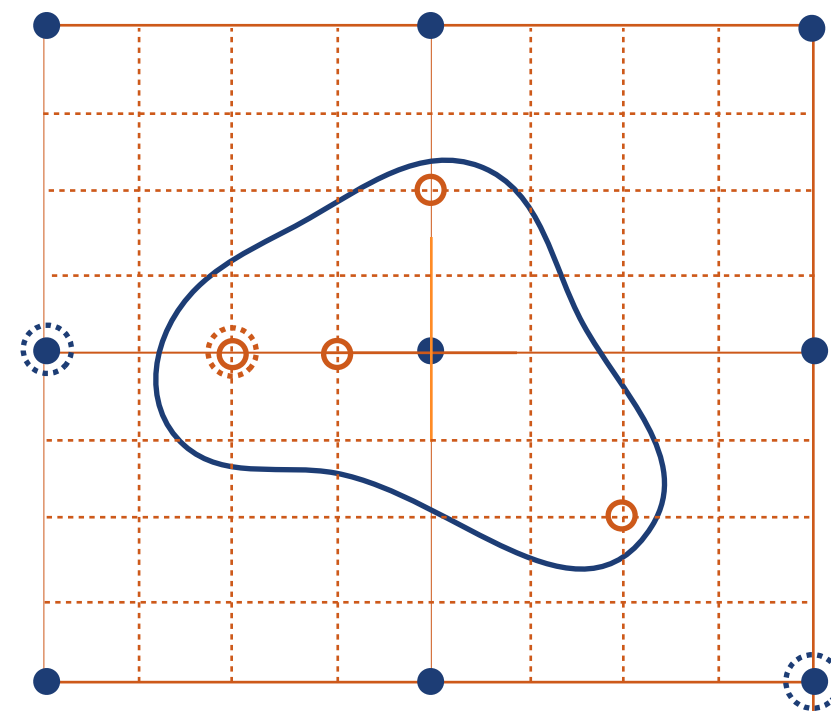
$$v \cdot v = r^2$$



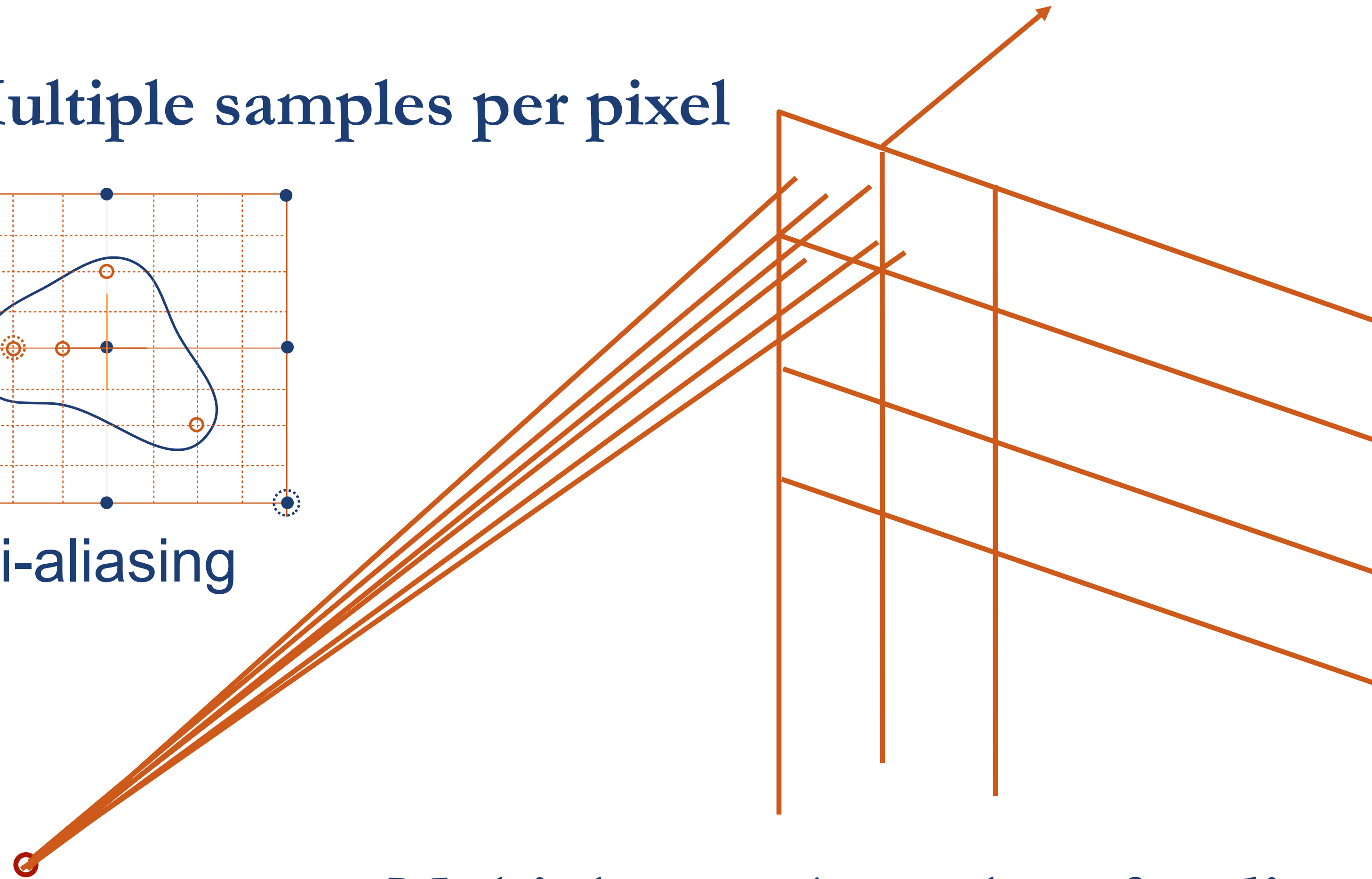


Distributed Ray Tracing

Multiple samples per pixel



Anti-aliasing



Multiple rays (samples of radiance)

Motion Blur



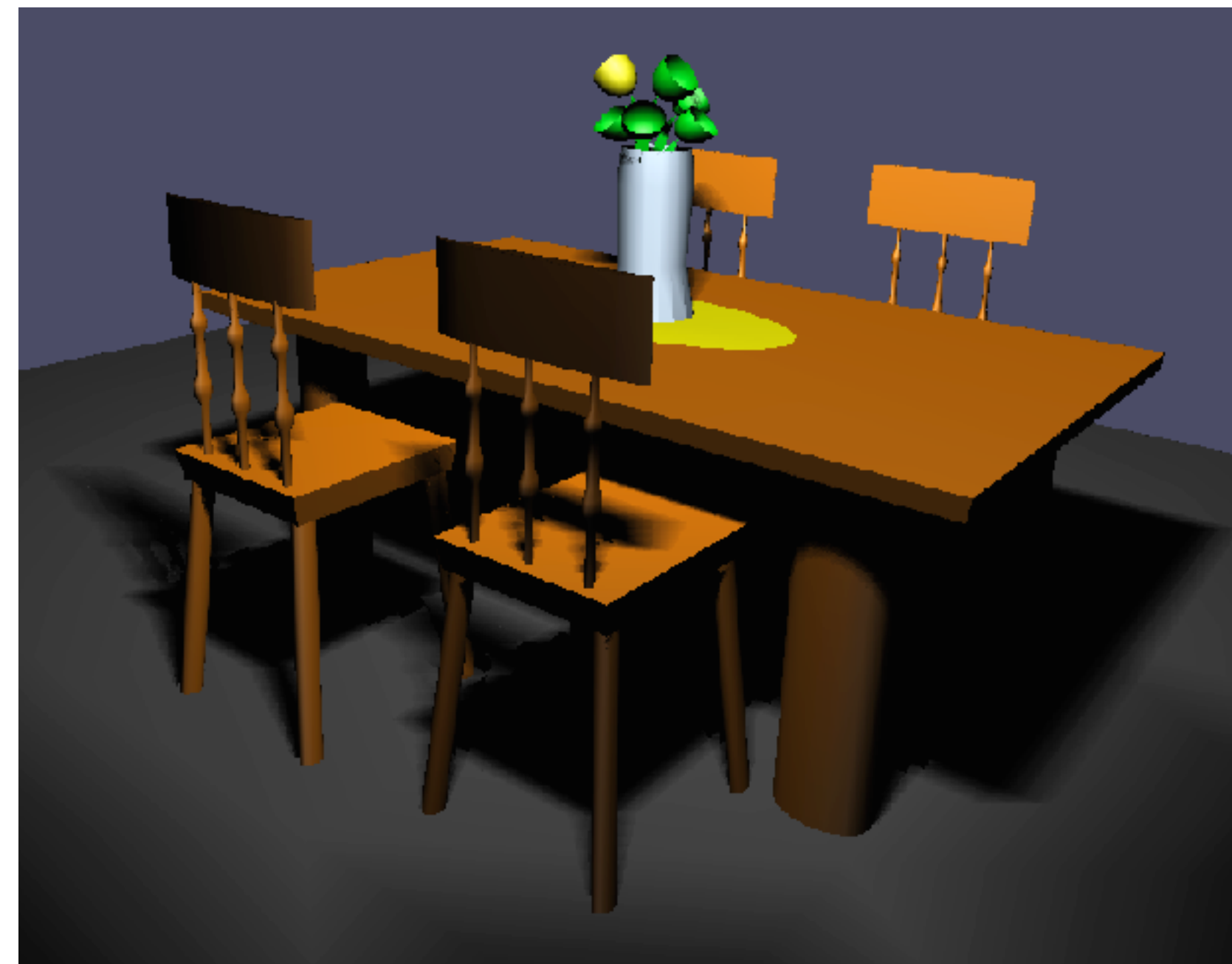
- Jitter rays in time





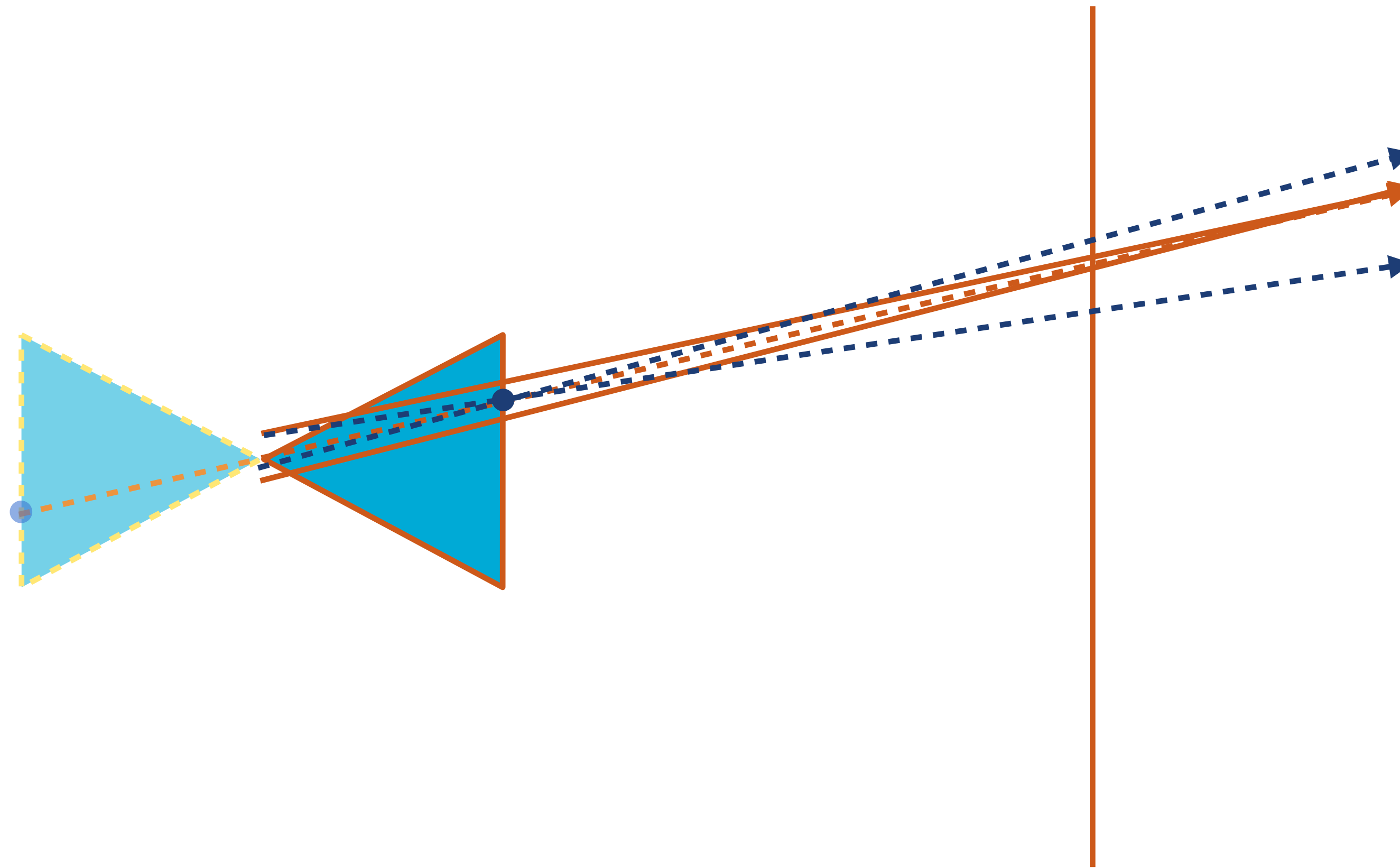
Soft Shadows

- Light has area
 - ➔ Shoot multiple rays to light
- Take average contribution

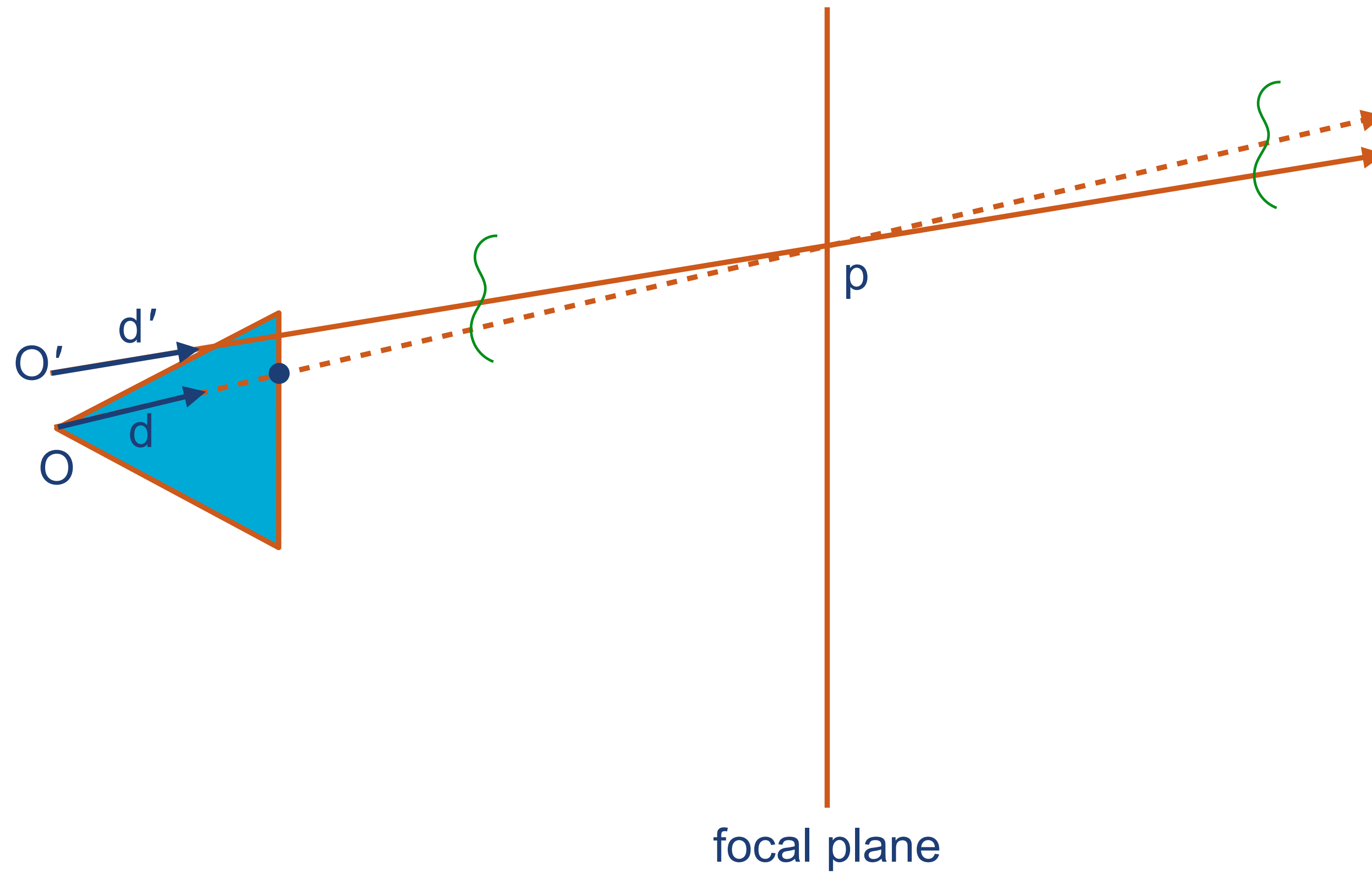


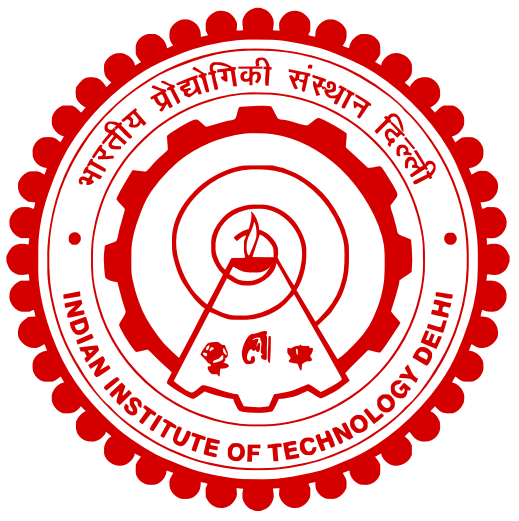
Courtesy H.P. Seidel

Depth of Field



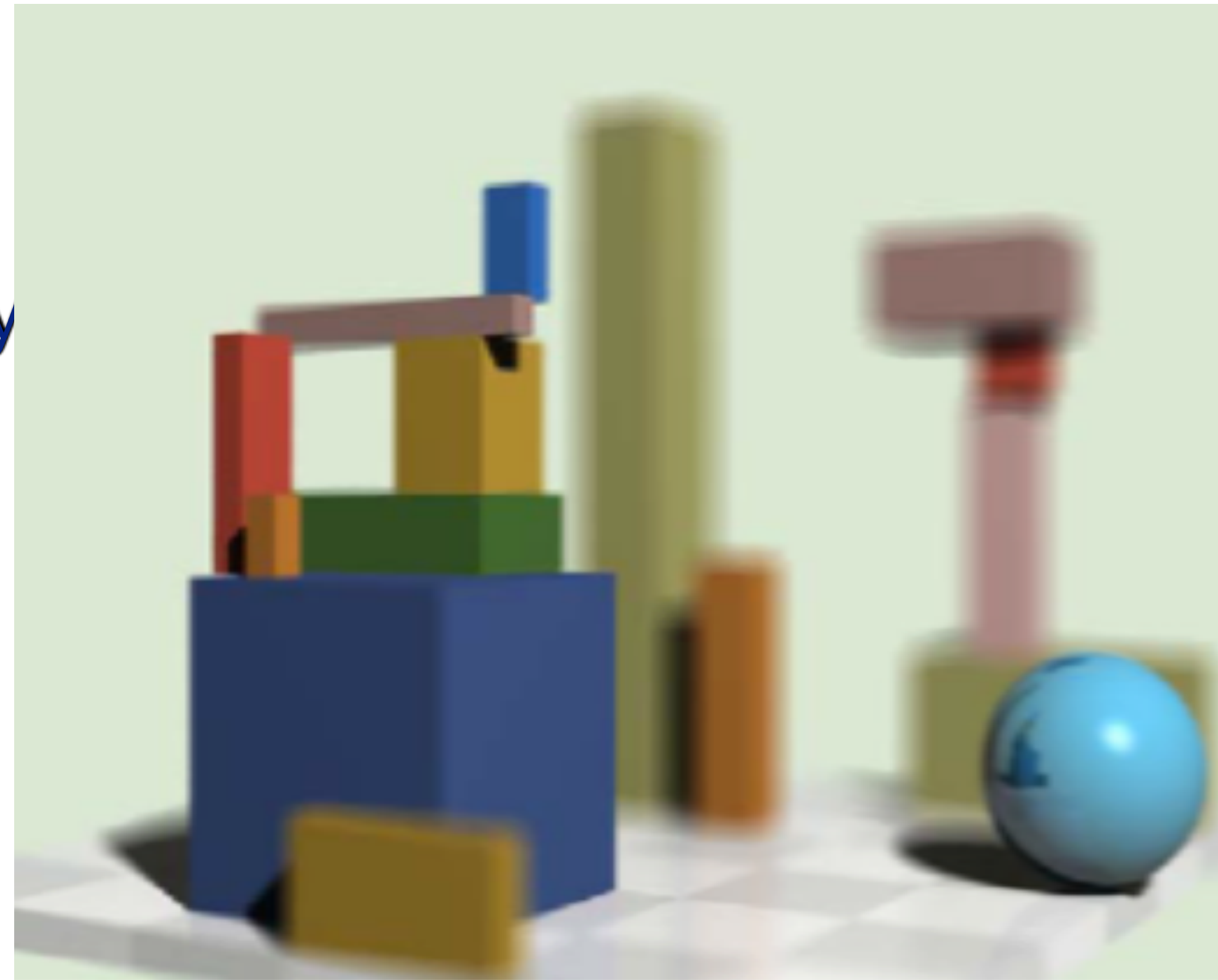
Depth of Field





Depth of Field

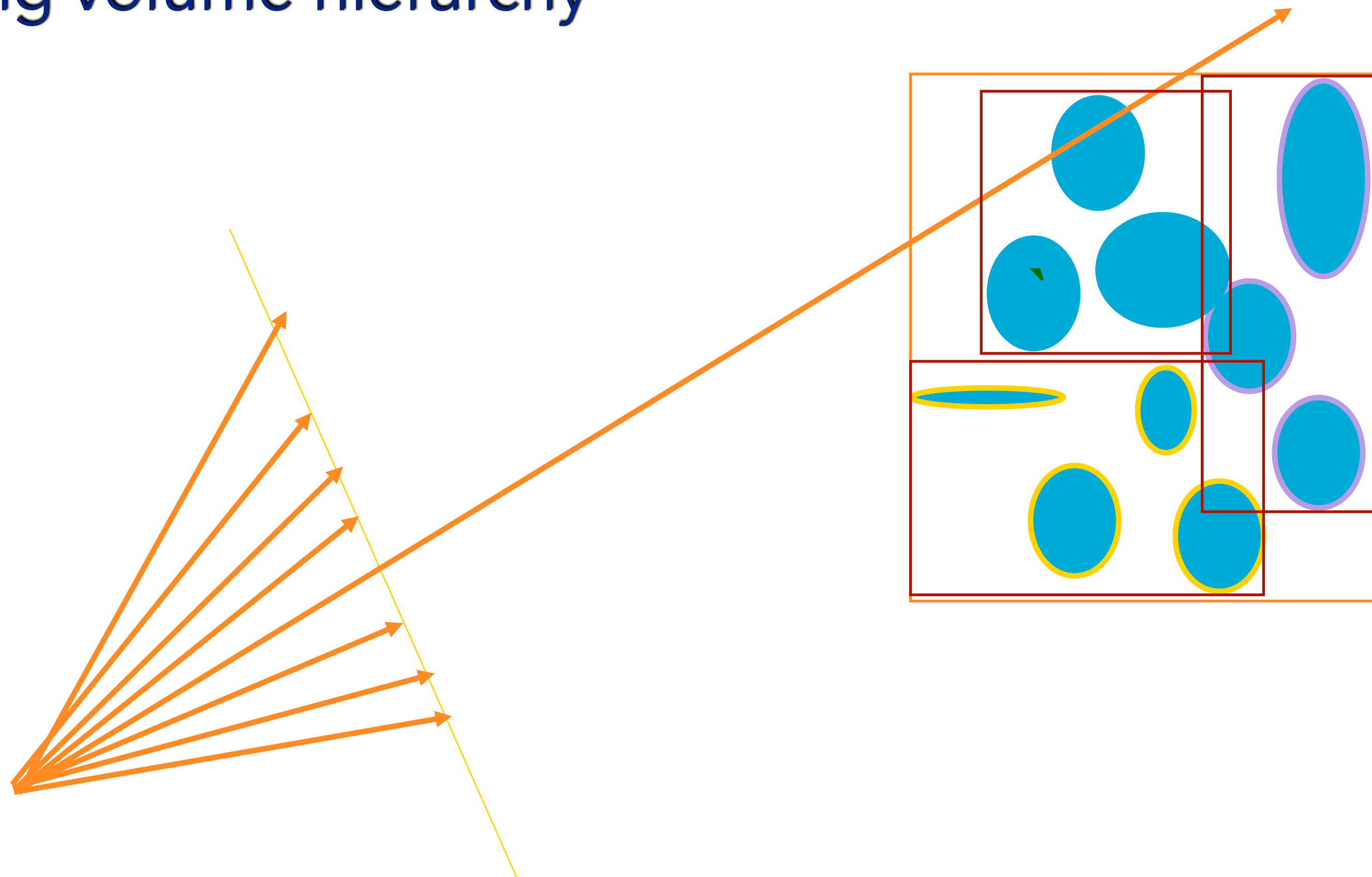
- Use multiple origins
 - ➔ Distributed on a disk
- Perturbed rays intersect original ray



Tracing Efficiency



- “Embarassingly” parallel
- Bounding volume hierarchy



Monte-Carlo Integration



- At intersection
 - ➡ Generate multiple rays
 - ➡ Typically based on material properties
 - ➡ (Weighted) Average of resulting ray colors



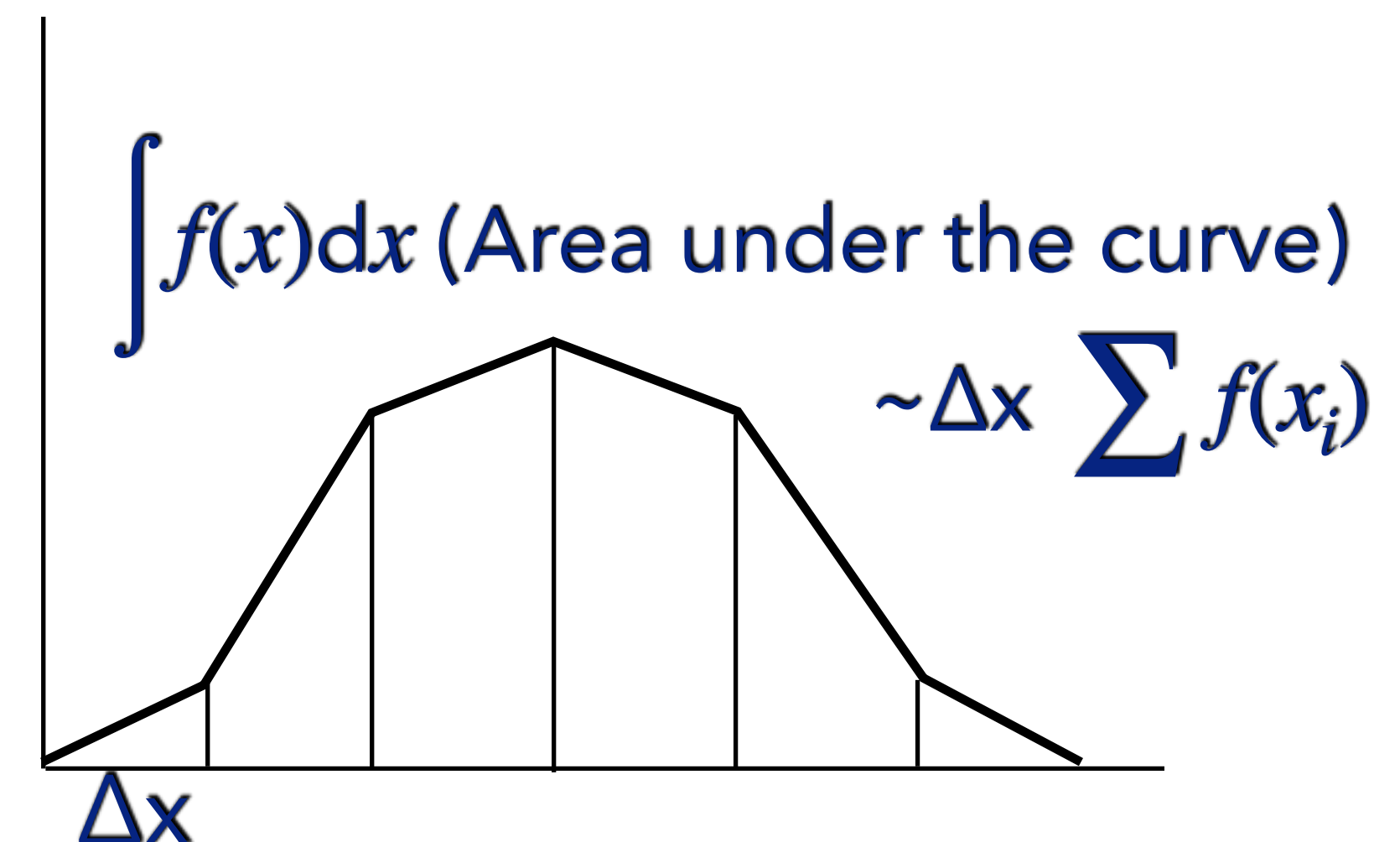
Irradiance H = Power per unit area

Radiance L = Irradiance per unit solid-angle

$$\text{BRDF } \rho = \frac{\text{Outgoing Radiance}}{\text{Incoming Irradiance}}$$

$$\begin{aligned} \text{Reflected Power from } \Delta A \text{ on surface due to } H_i &= \Delta A \int_{\phi=0}^{2\pi} \int_{\theta=0}^{\pi/2} H_i \rho \sin \theta d\theta d\phi \\ &= \Delta A \pi C, \text{ if } \rho = C \\ &\text{for all reflected directions} \end{aligned}$$

If 100% of incident power is reflected, $C = \frac{1}{\pi}$



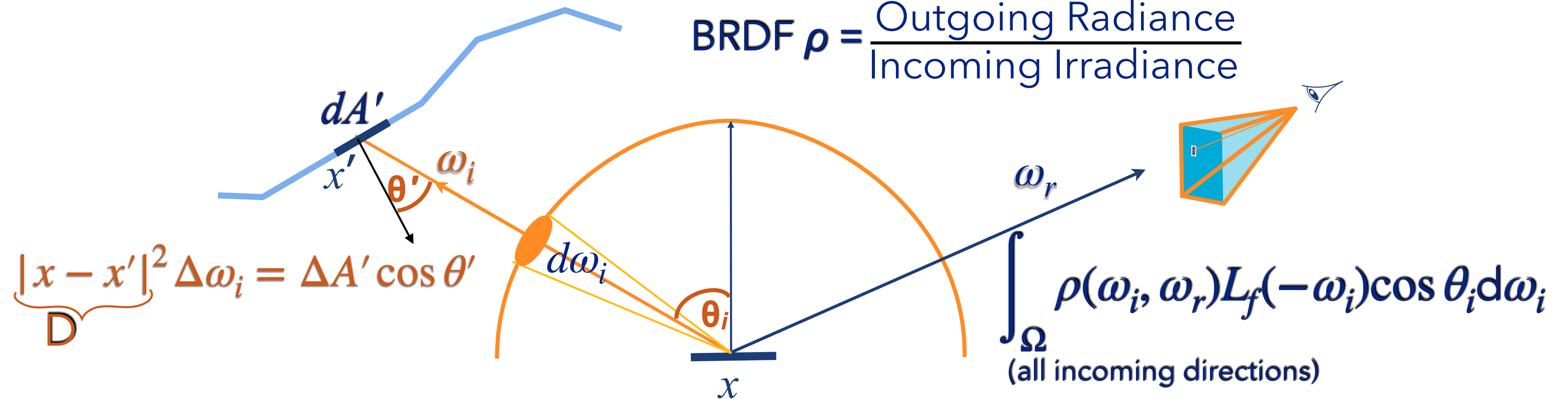


Rendering Equation

Irradiance H = Power per unit area

Radiance L = Irradiance per unit solid-angle

$$\text{BRDF } \rho = \frac{\text{Outgoing Radiance}}{\text{Incoming Irradiance}}$$



Reflected Light
(Output Image)

Emitted
Light

BRDF

Incident
angle

$$L_r(\omega_r)_x = L_e(\omega_r)_x + \int_{\text{all } x' \text{ that are visible}} \rho(\omega_i, \omega_r)_x L_r(-\omega_i)_{x'} \cos \theta_i \frac{\cos \theta' dA'}{D^2}$$

UNKNOWN

UNKNOWN

Source Geometry