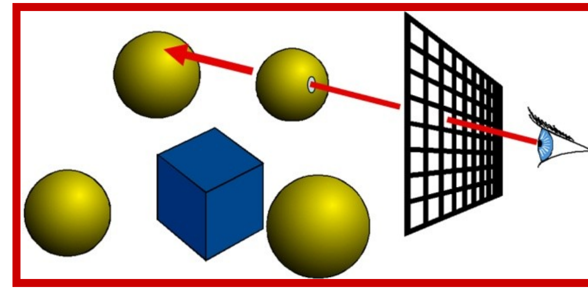
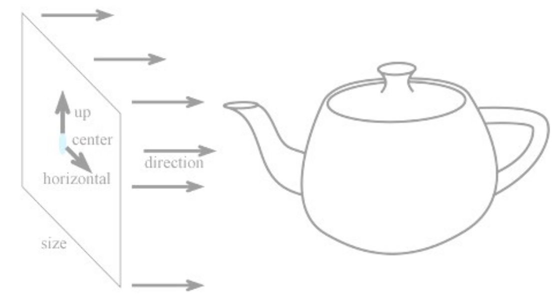
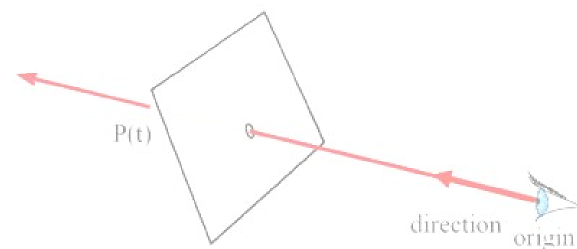
Ray Casting and Ray Tracing

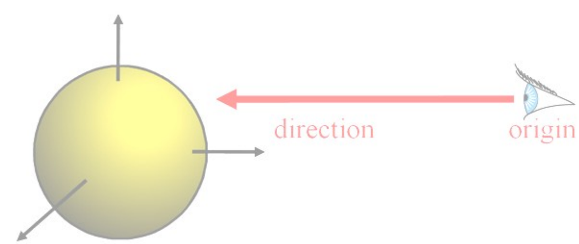
Topics

- Ray Casting Basics
- Camera and Ray Generation
- Ray Object Intersection
  - Plane
  - Sphere
  - Triangle
  - General Quadric Surface
- Recursive Ray Tracing
  - Mirror Reflection
  - Refraction



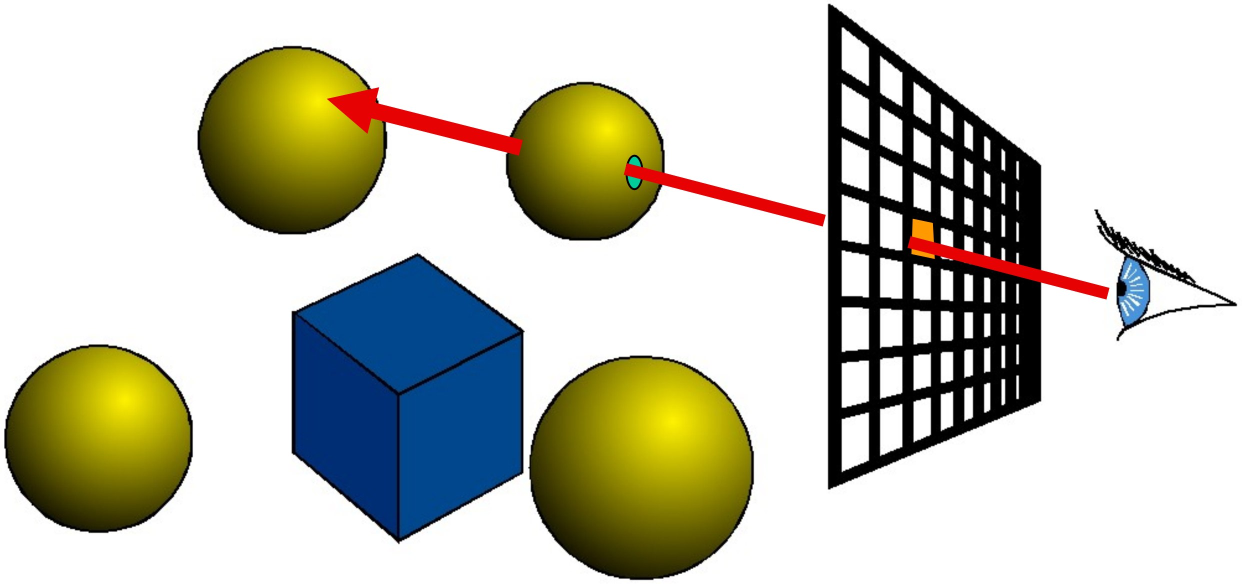






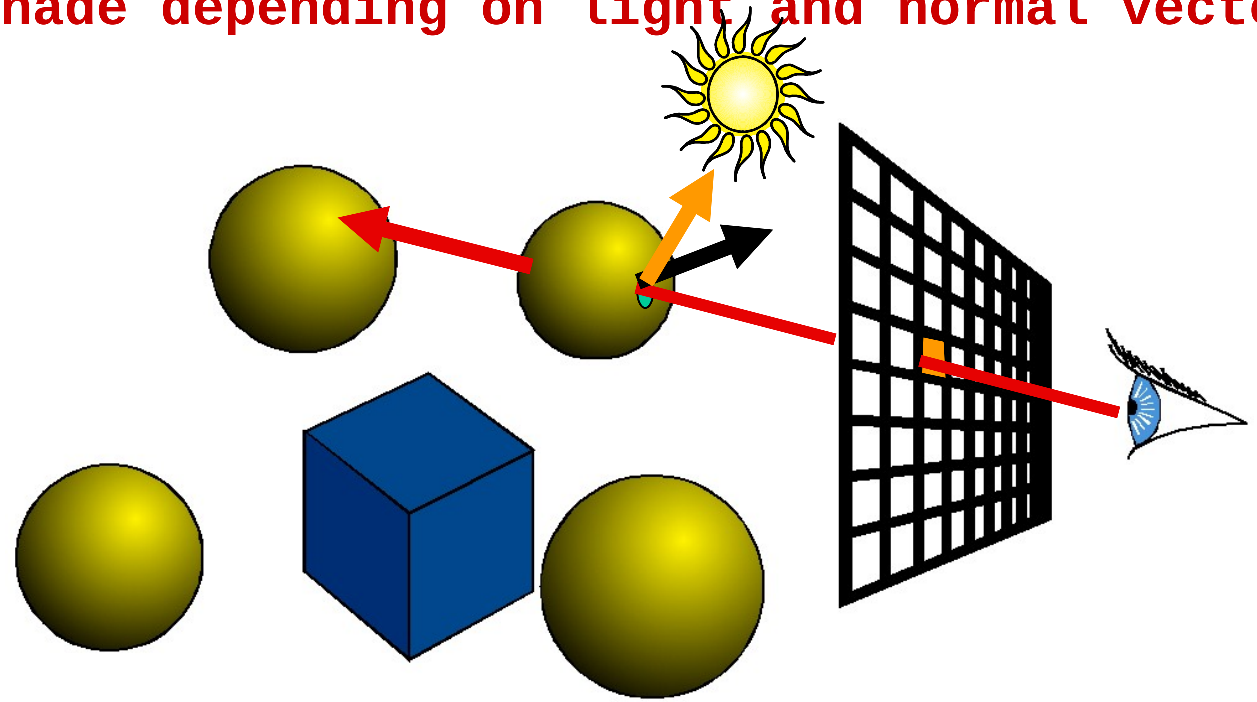
What is Ray Casting?

```
For every pixel
  Construct a ray from the eye
  For every object in the scene
    Find intersection with the ray
    Keep if closest
```



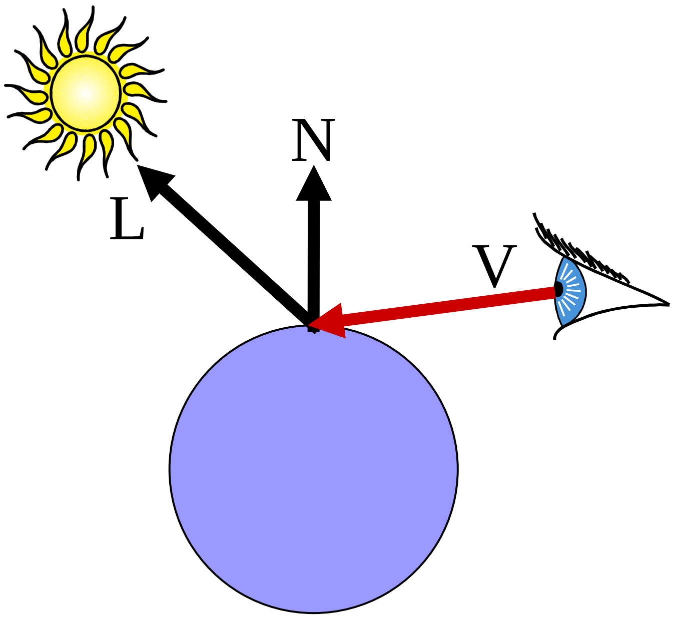
Shading

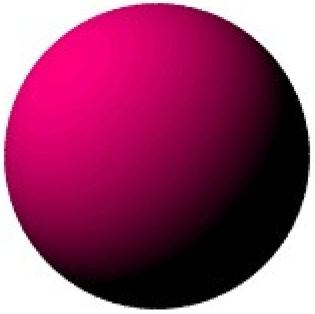
```
For every pixel
  Construct a ray from the eye
  For every object in the scene
    Find intersection with the ray
    Keep if closest
  Shade depending on light and normal vector
```



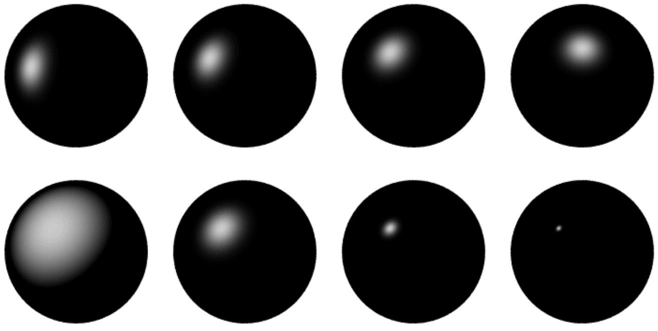
Shading - Recap

- Surface/Scene Characteristics:
  - Surface normal
  - Direction to light
  - Viewpoint
- Material Properties
  - Diffuse (matte)
  - Specular (shiny)
  - …
- Lighting Model etc.





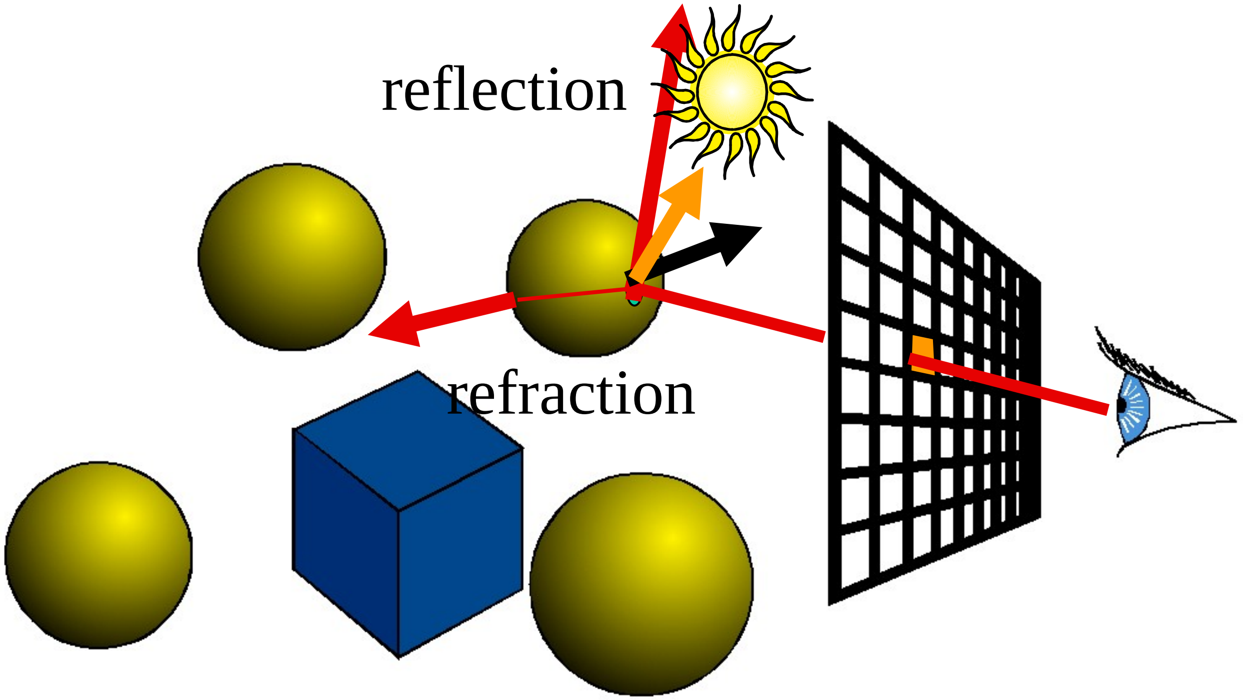
Diffuse sphere



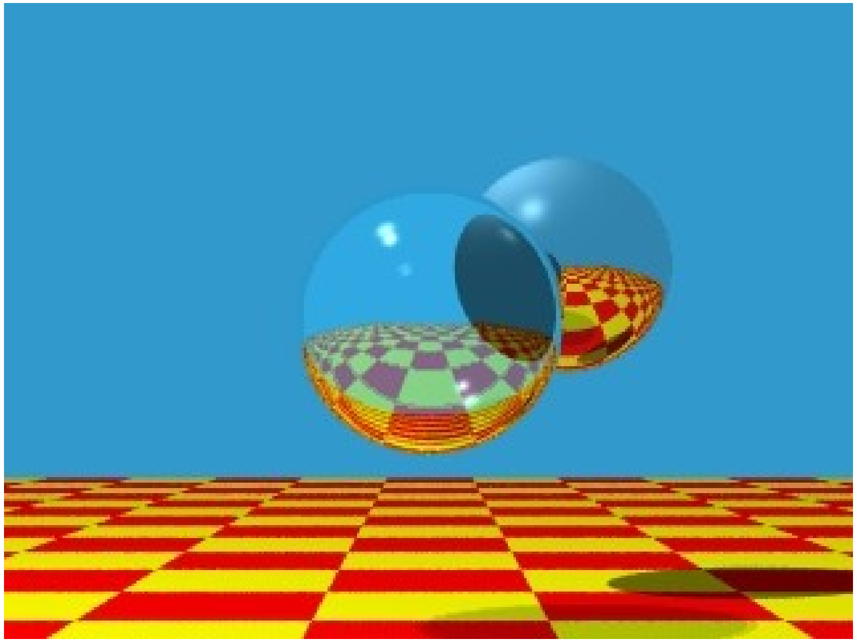
Specular spheres

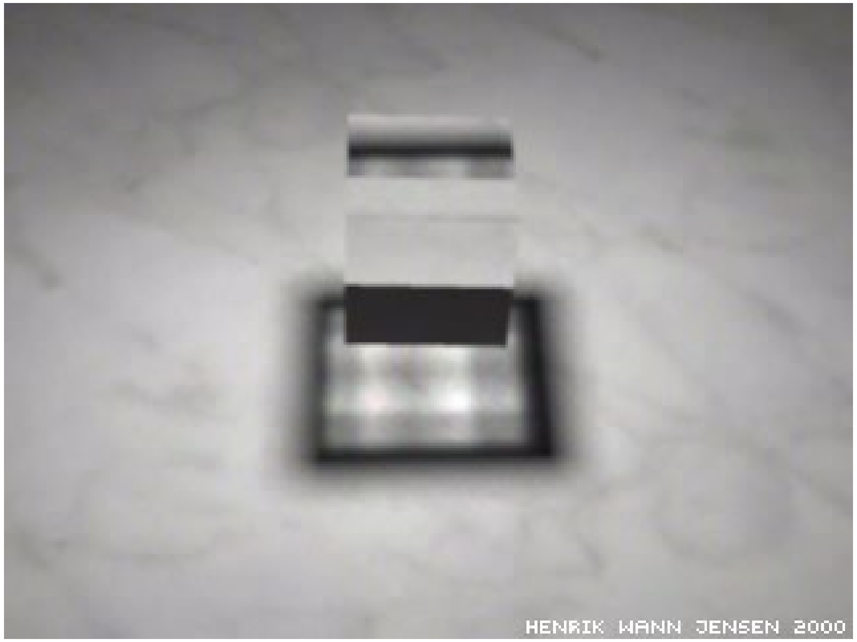
What is Ray Tracing?

- Secondary rays (shadows, reflection, refraction)



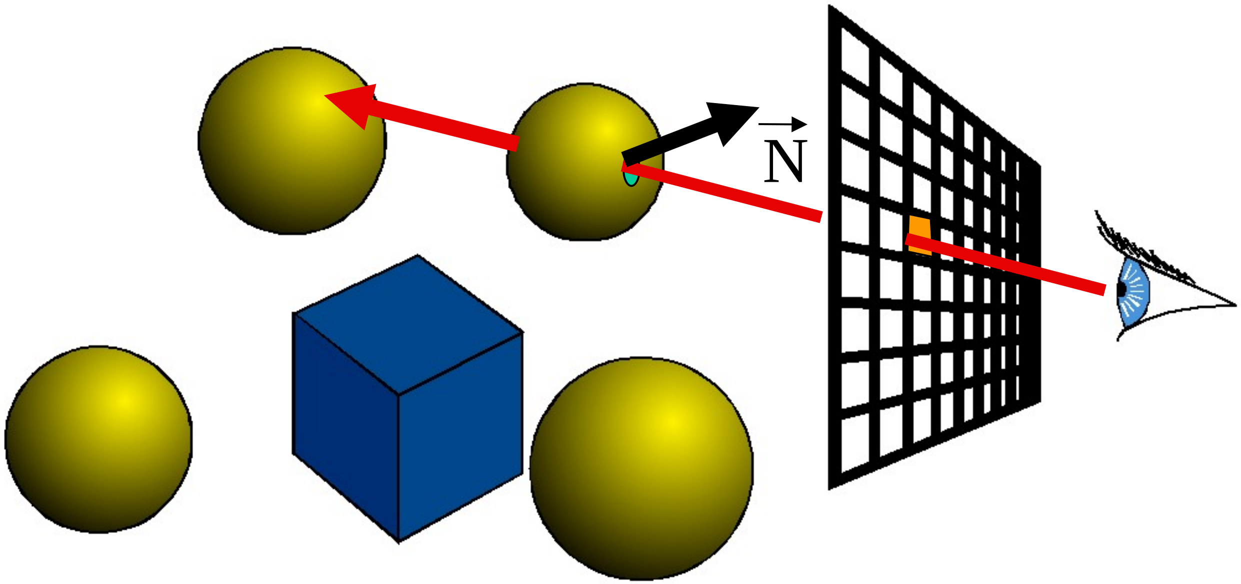
Ray Tracing Scenes





Ray Casting - Basic Computations

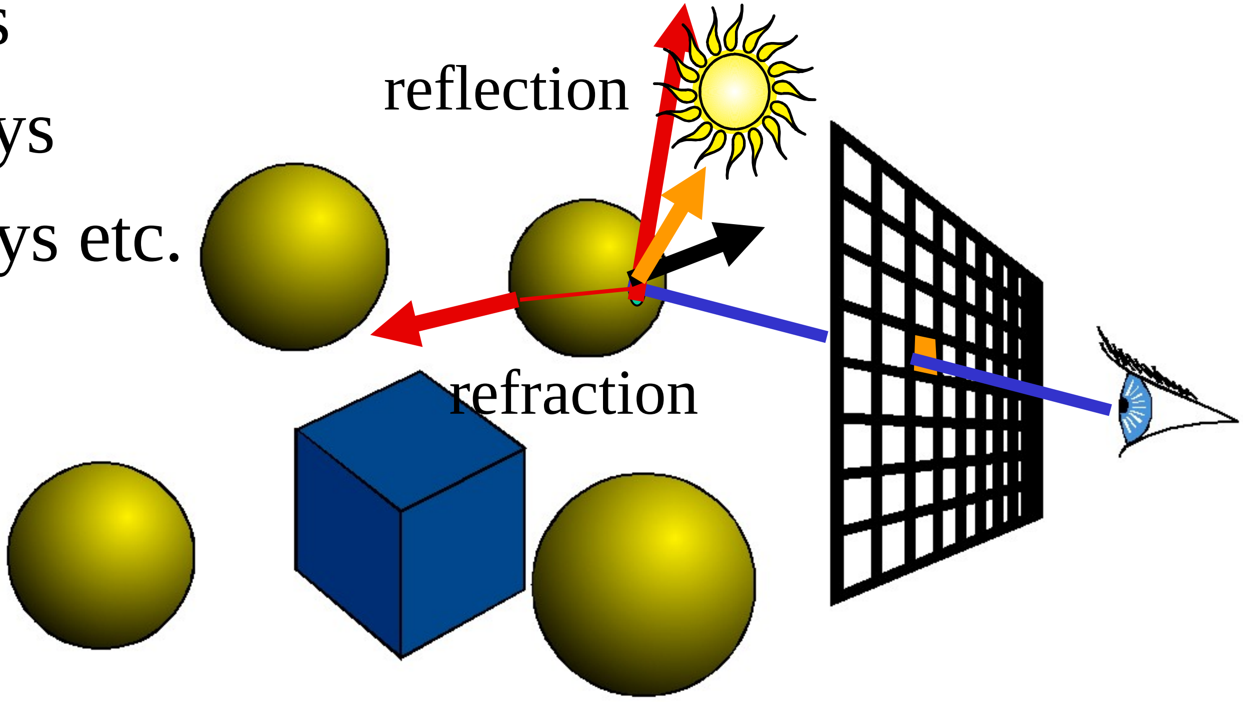
```
For every pixel
  Construct a ray from the eye
  For every object in the scene
      Find intersection with the ray
      Keep if closest
   Shade depending on light and normal vector
```



Finding the intersection and normal is the central part of ray casting

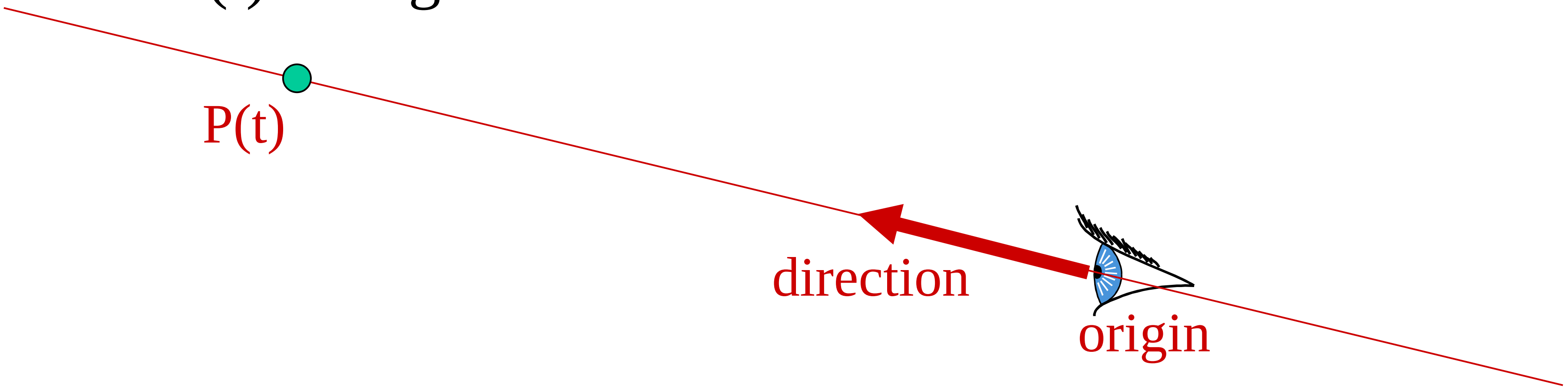
Ray Casting vs. Ray Tracing

- Ray Casting : Consider eye or camera rays only
- Ray Tracing : Consider secondary rays too
  - – Shadow rays
  - – Reflected rays
  - – Refracted rays etc.



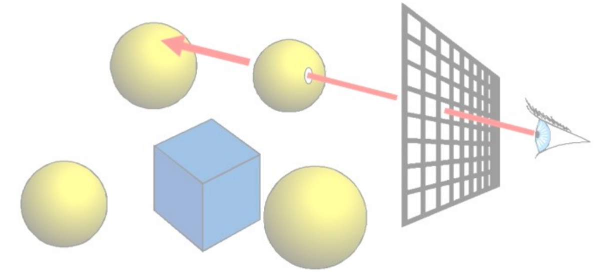
Ray Representation

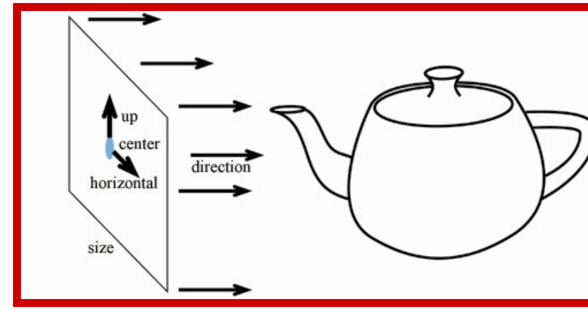
- Two vectors:
  - Origin
  - Direction (normalized is better)
- Parametric line
  - P(t) = origin + t * direction

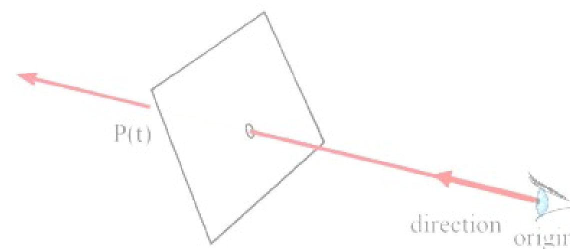


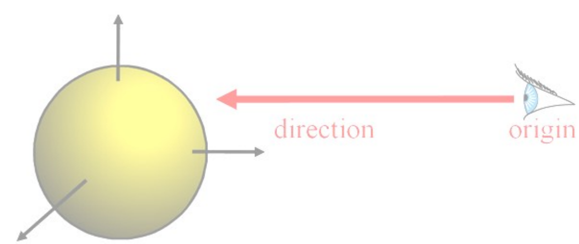
Topics

- Ray Casting Basics
- Camera and Ray Generation
- Ray Object Intersection
  - Plane
  - Sphere
  - Triangle
  - General Quadric Surface
- Recursive Ray Tracing
  - Mirror Reflection
  - Refraction



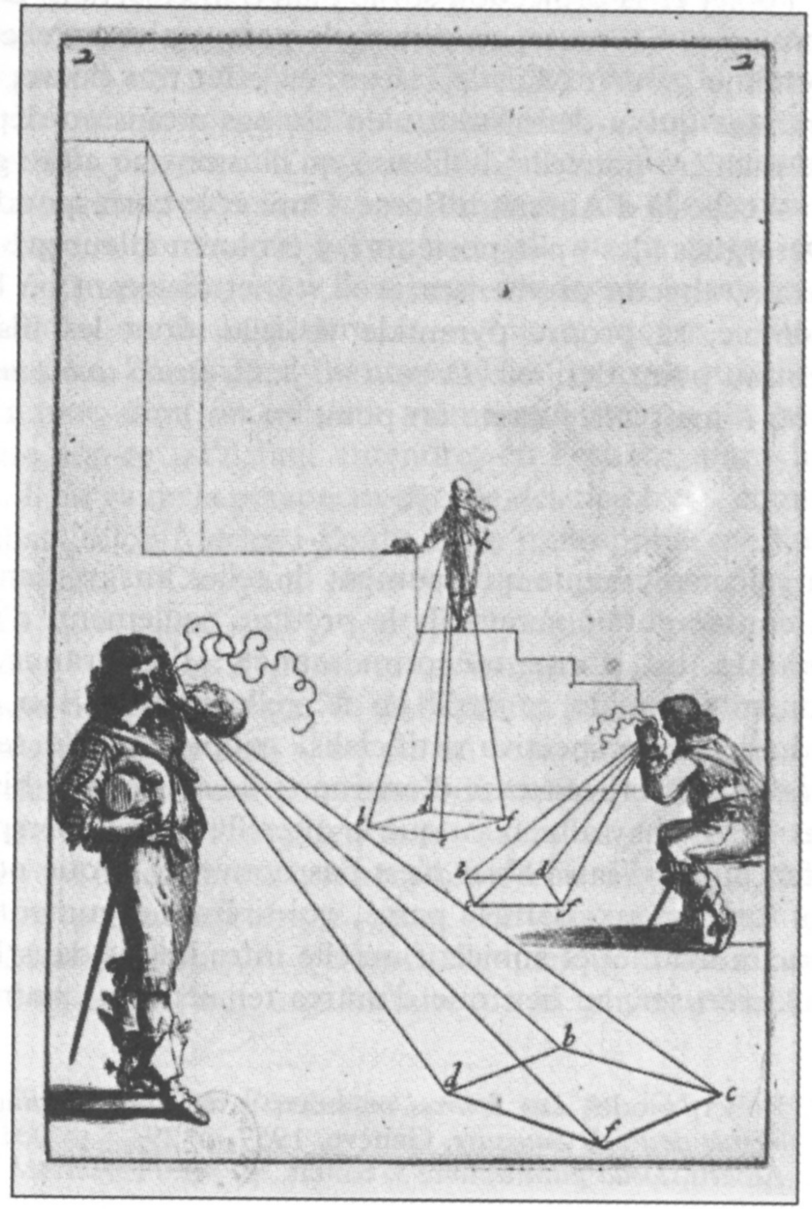






Cameras

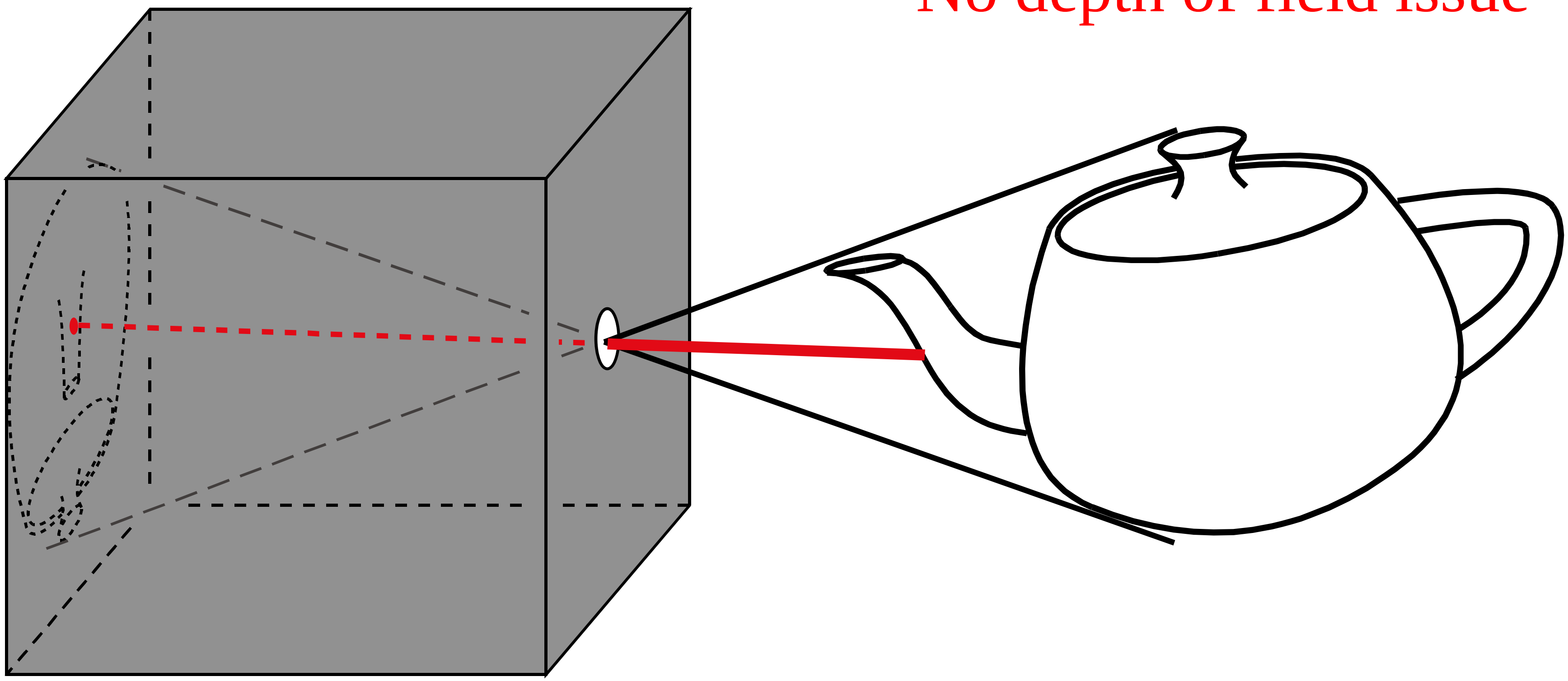
```
For every pixel
  Construct a ray from the eye
  For every object in the scene
    Find intersection with ray
      Keep if closest
```



Abraham Bosse, *Les Perspecteurs*. Gravure extraite de la *Manière*

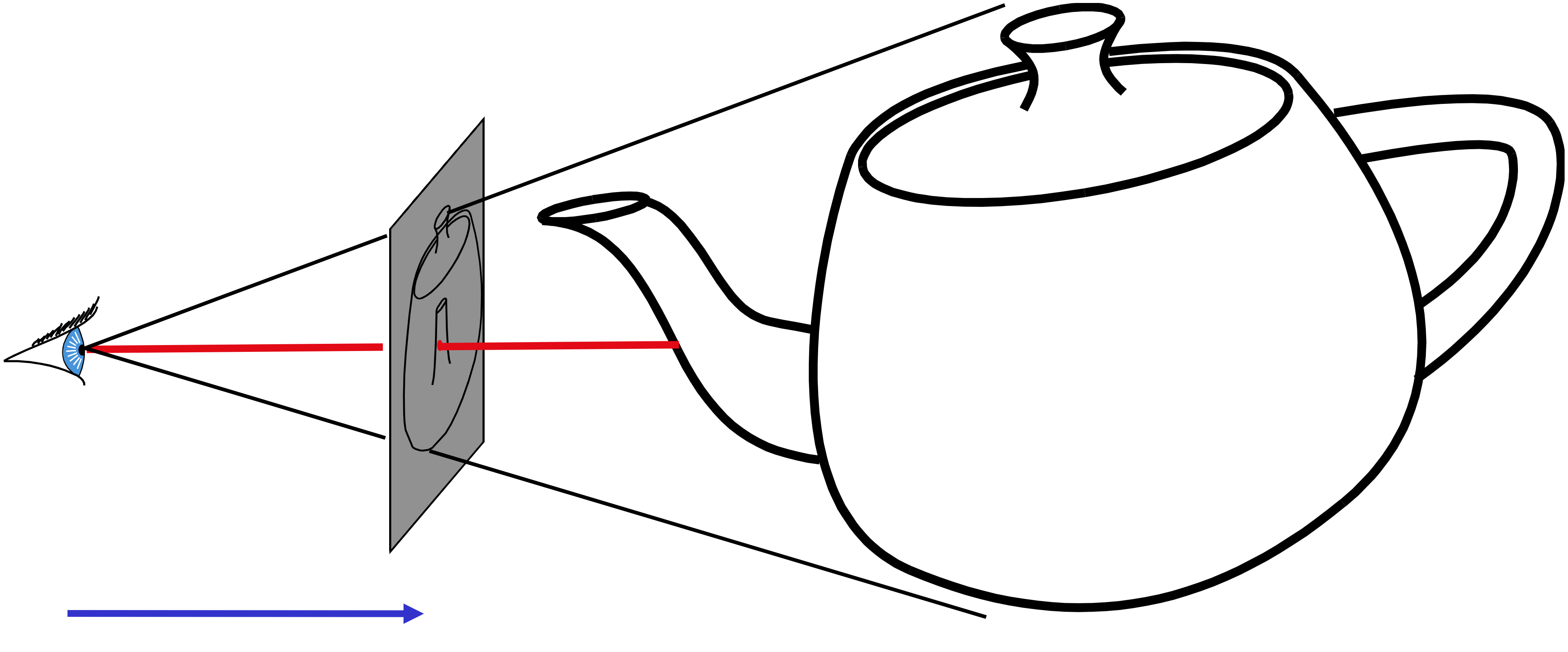
Pinhole Camera

- Box with a tiny hole
- Inverted image
- Similar triangles
- Perfect image if hole infinitely small
- Pure geometric optics
- No depth of field issue



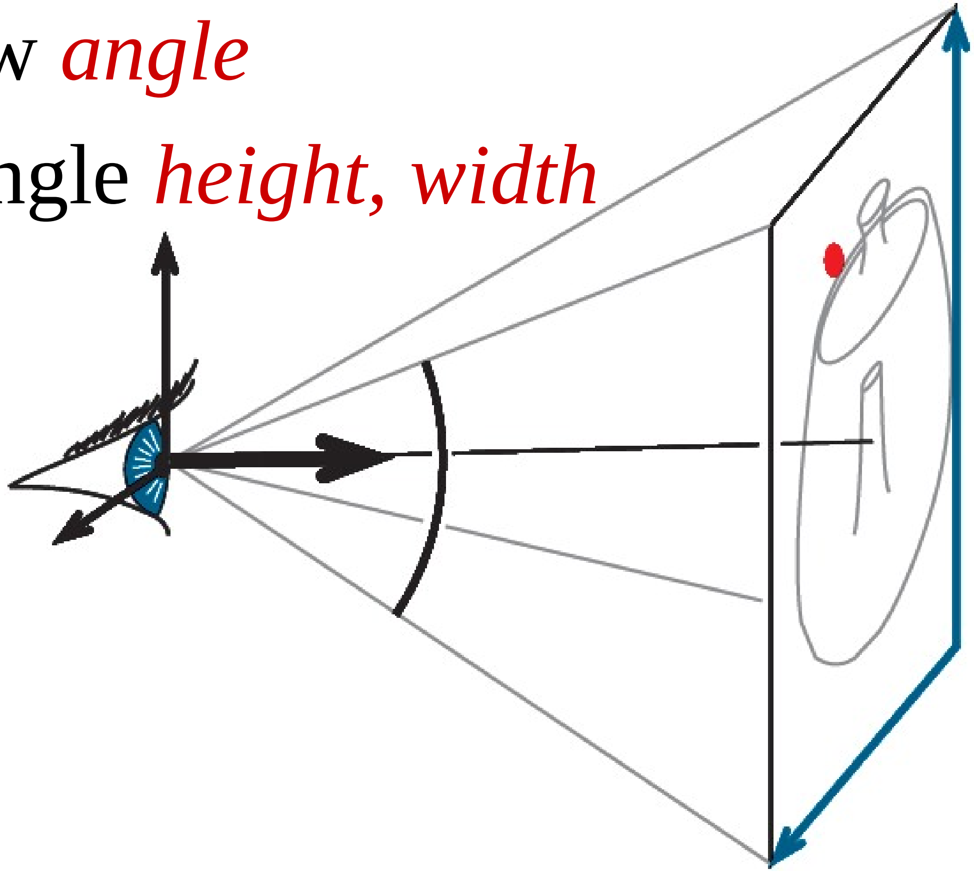
Simplified Pinhole Camera

- Eye-image pyramid (frustum)
- Note that the distance/size of image are arbitrary

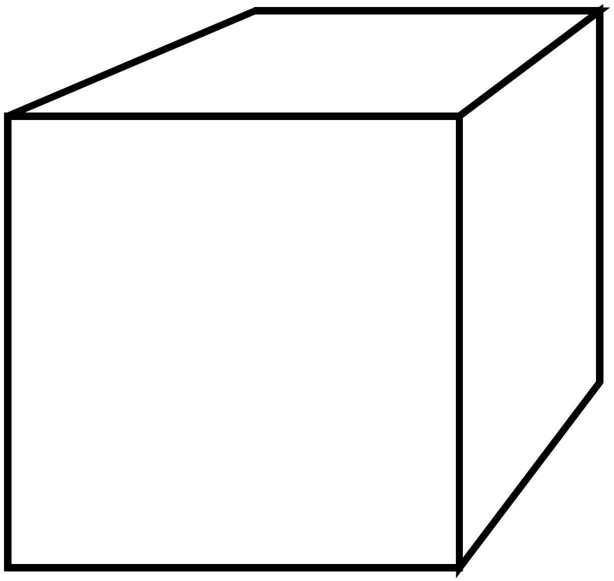


Camera Description

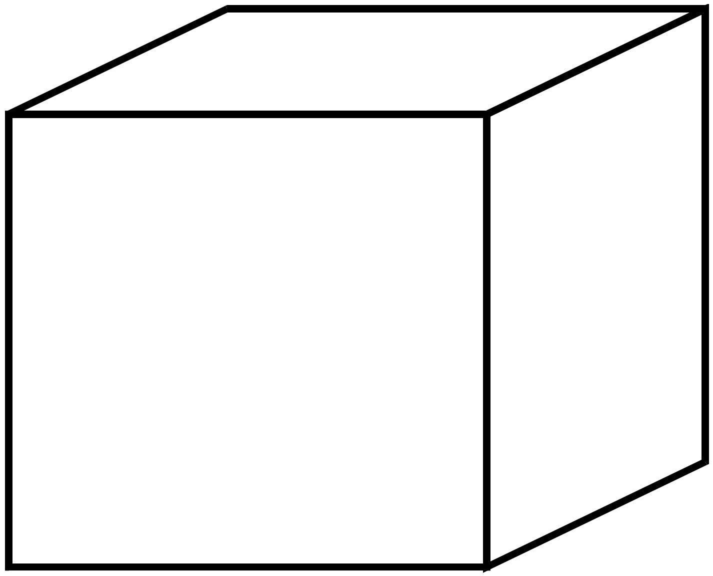
- Eye point *e (center)*
- Orthobasis *u, v, w (horizontal, up, -direction)*
- Field of view *angle*
- Image rectangle *height, width*



Perspective vs. Orthographic



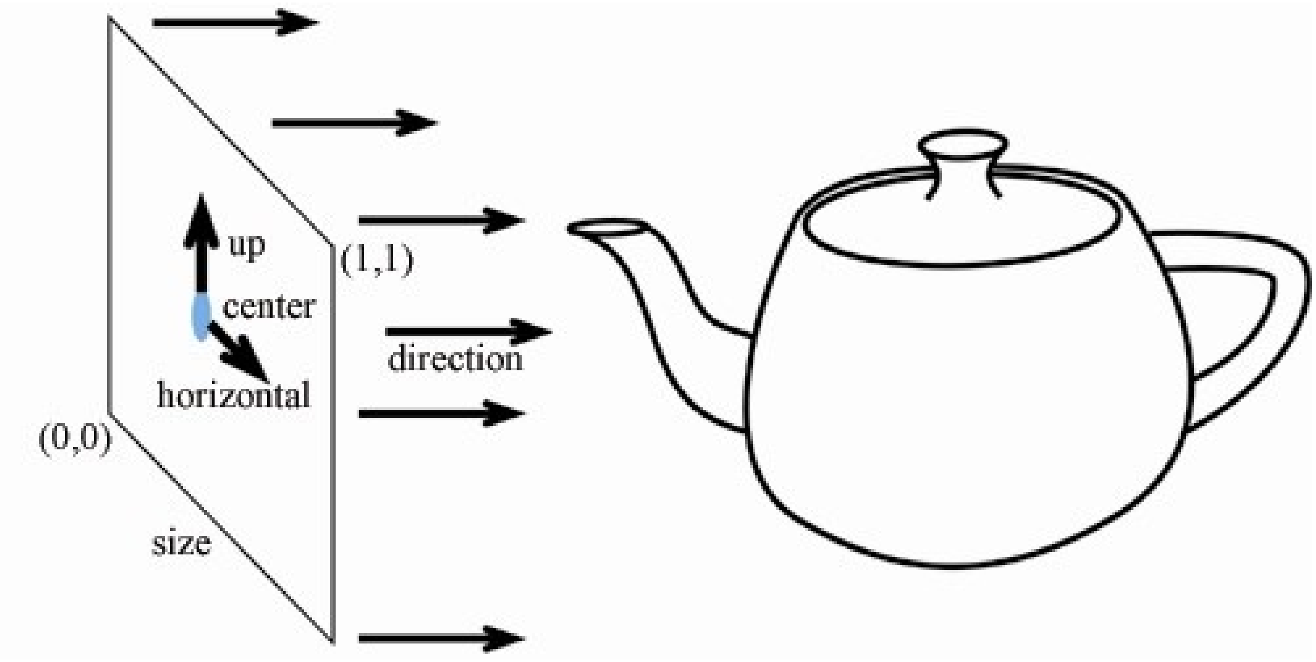
perspective



orthographic

- Parallel projection
- No foreshortening
- No vanishing point

Orthographic Camera



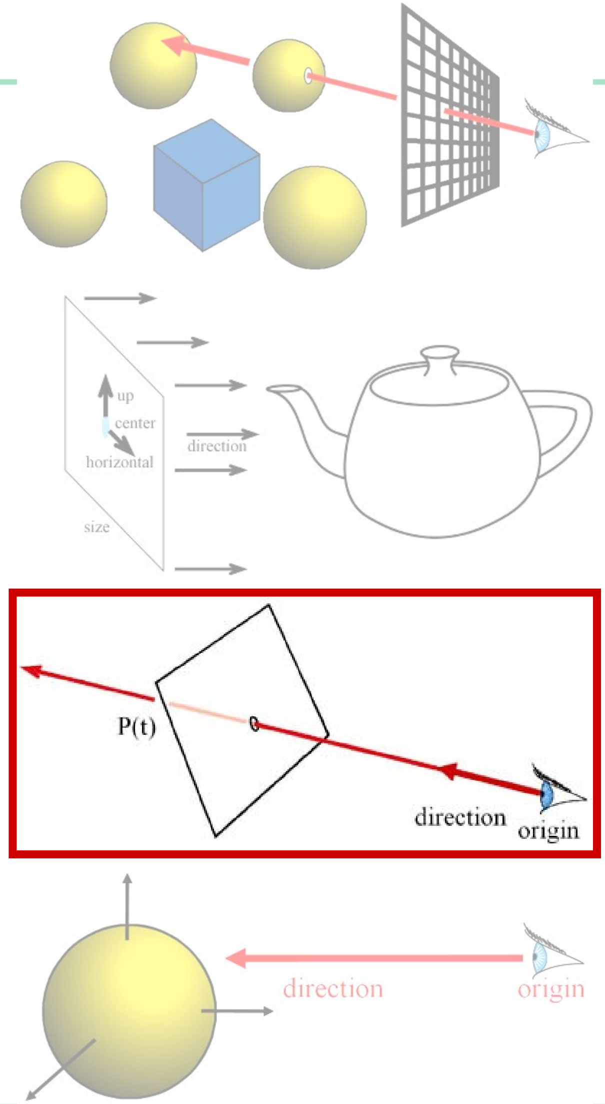
- Ray Generation?
  - Origin = center + (x-0.5)*size*horizontal + (y-0.5)*size*up ??
  - Direction is constant

Other Weird Cameras

- E.g. fish eye, omnimax, panorama

Topics

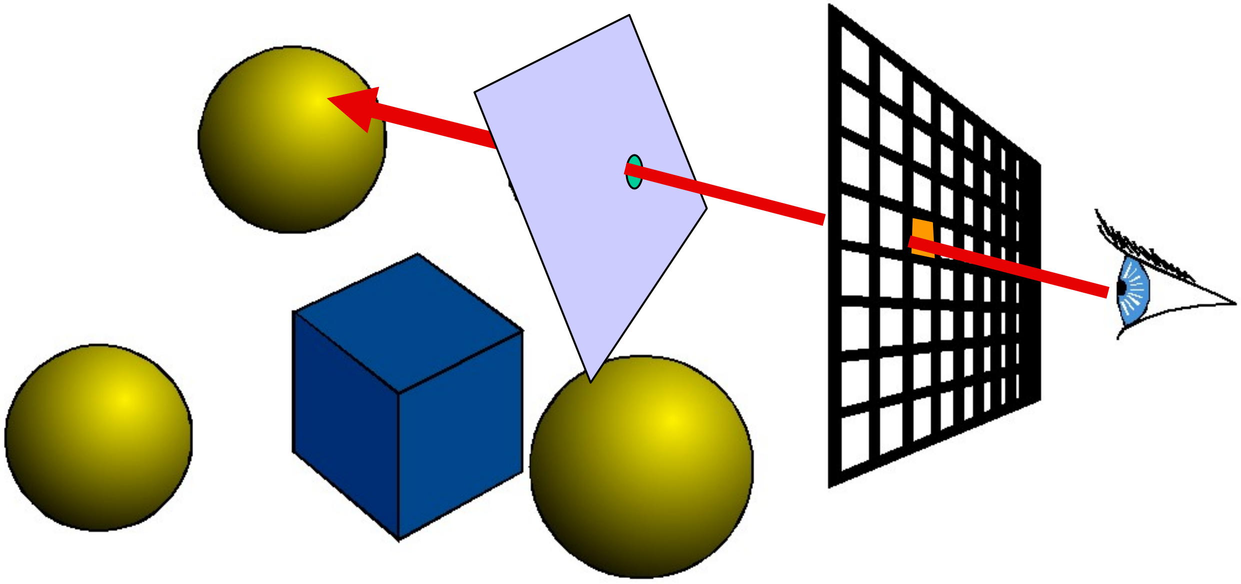
- Ray Casting Basics
- Camera and Ray Generation
- Ray Object Intersection
  - Plane
  - Sphere
  - Triangle
  - General Quadric Surface
- Recursive Ray Tracing
  - Mirror Reflection
  - Refraction



Ray Casting – Finding Intersection

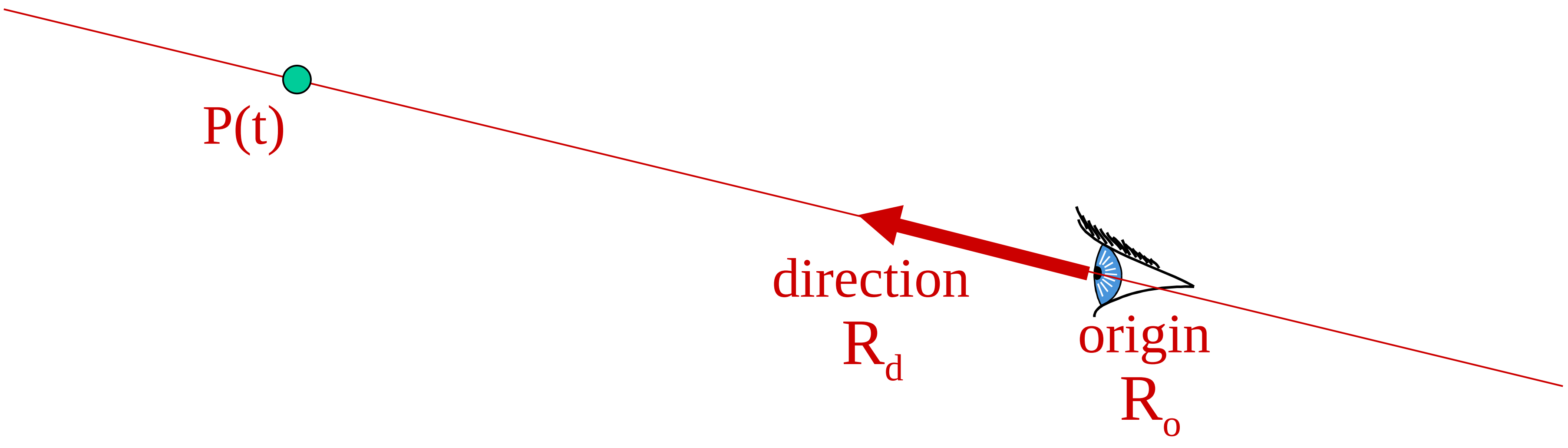
```
For every pixel
  Construct a ray from the eye
  For every object in the scene
      Find intersection with the ray
      Keep if closest
```

First we will study ray-plane intersection



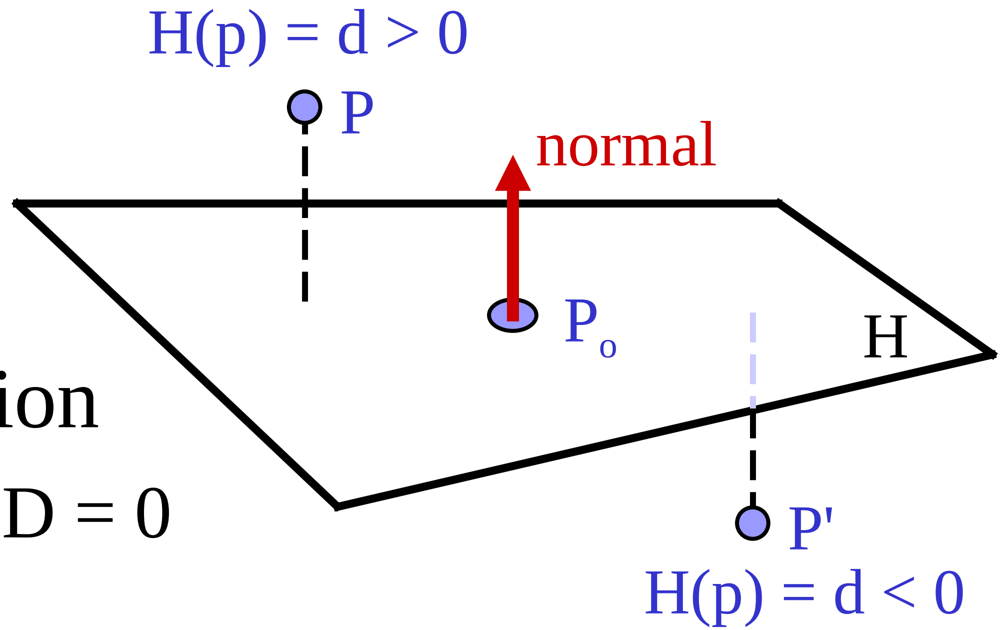
Recall: Ray Representation

- Parametric line
- $P(t) = R_o + t * R_d$
- Explicit representation



3D Plane Representation?

- Plane defined by
  - $P_o = (x,y,z)$
  - $n = (A,B,C)$
- Implicit plane equation
  - $H(P) = Ax+By+Cz+D = 0$
    $= n \cdot P + D = 0$
- Point-Plane distance?
  - If n is normalized,
    distance to plane, $d = H(P)$
  - d is the *signed distance*!

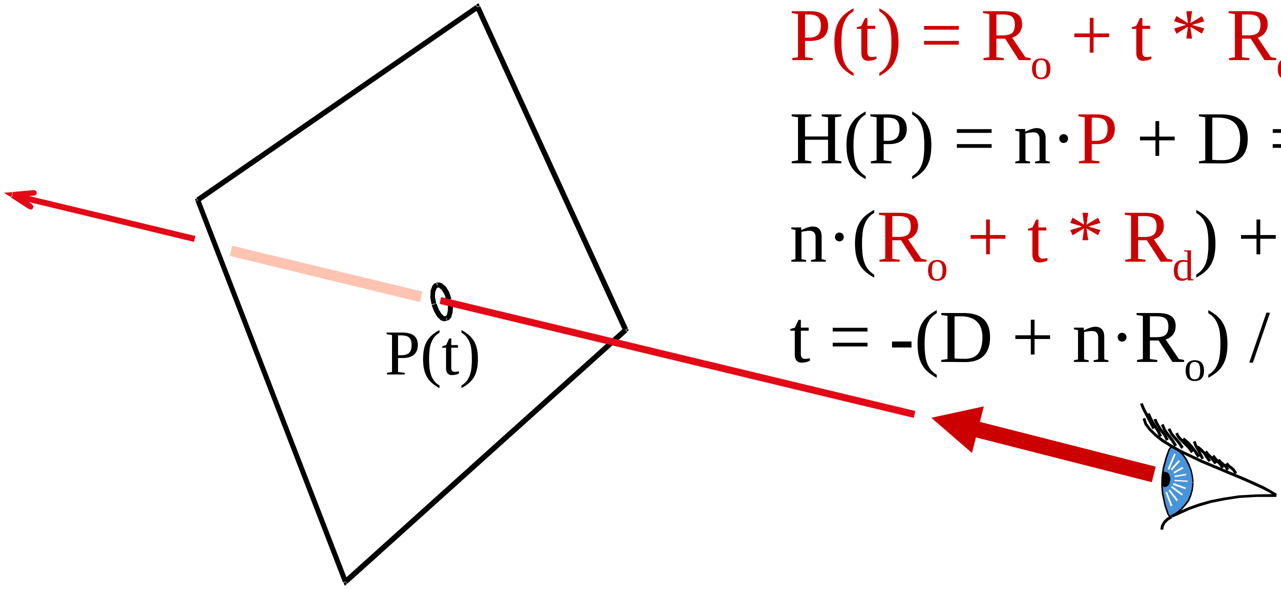


Explicit vs. Implicit?

- Ray equation is explicit $P(t) = R_o + t * R_d$
  - Parametric
  - Generates points
  - Hard to verify that a point is on the ray
- Plane equation is implicit $H(P) = n \cdot P + D = 0$
  - Solution of an equation
  - Does not generate points
  - Verifies that a point is on the plane

Ray-Plane Intersection

- Intersection means both are satisfied
- So, insert explicit equation of ray into implicit equation of plane & solve for t



$P(t) = R_o + t * R_d$

$H(P) = n \cdot P + D = 0$

$n \cdot (R_o + t * R_d) + D = 0$

$t = -(D + n \cdot R_o) / n \cdot R_d$

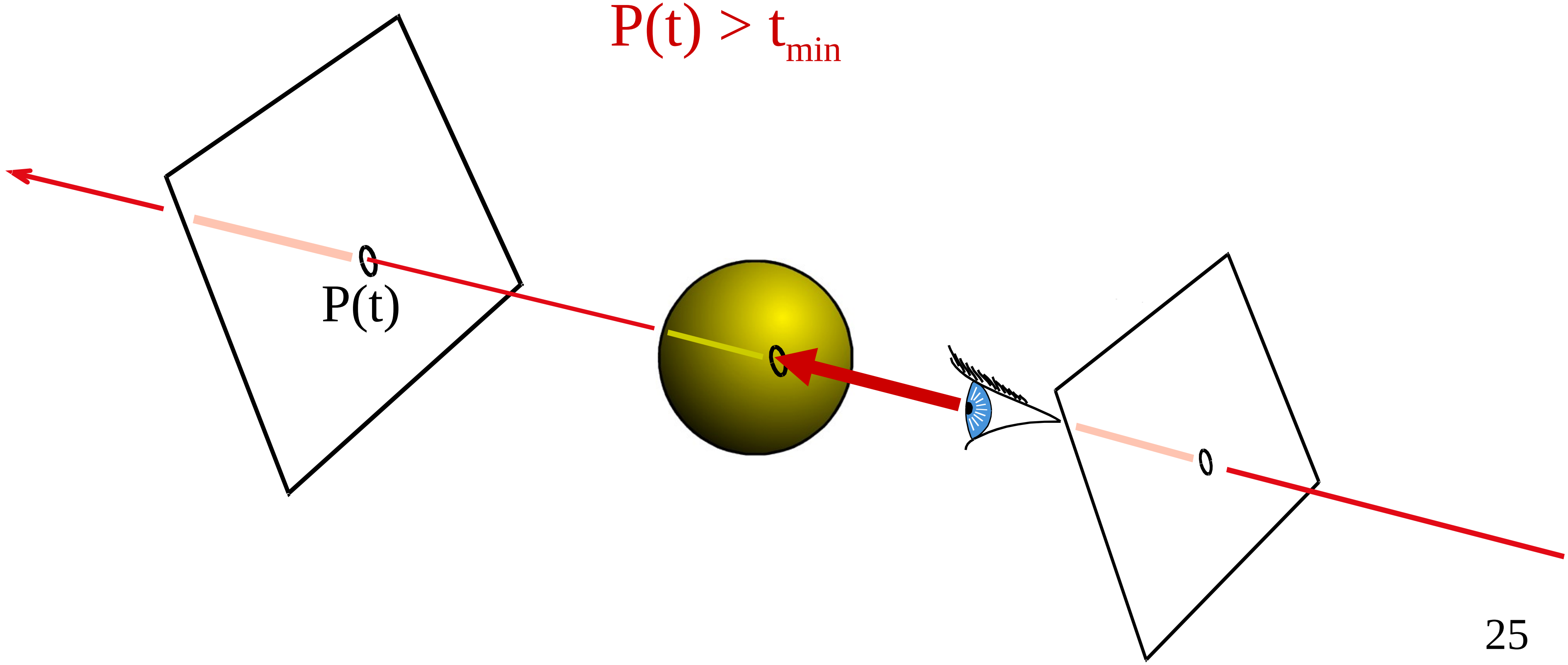
Additional Checks

- Verify that intersection is closer than previous

$$P(t) < t_{current}$$

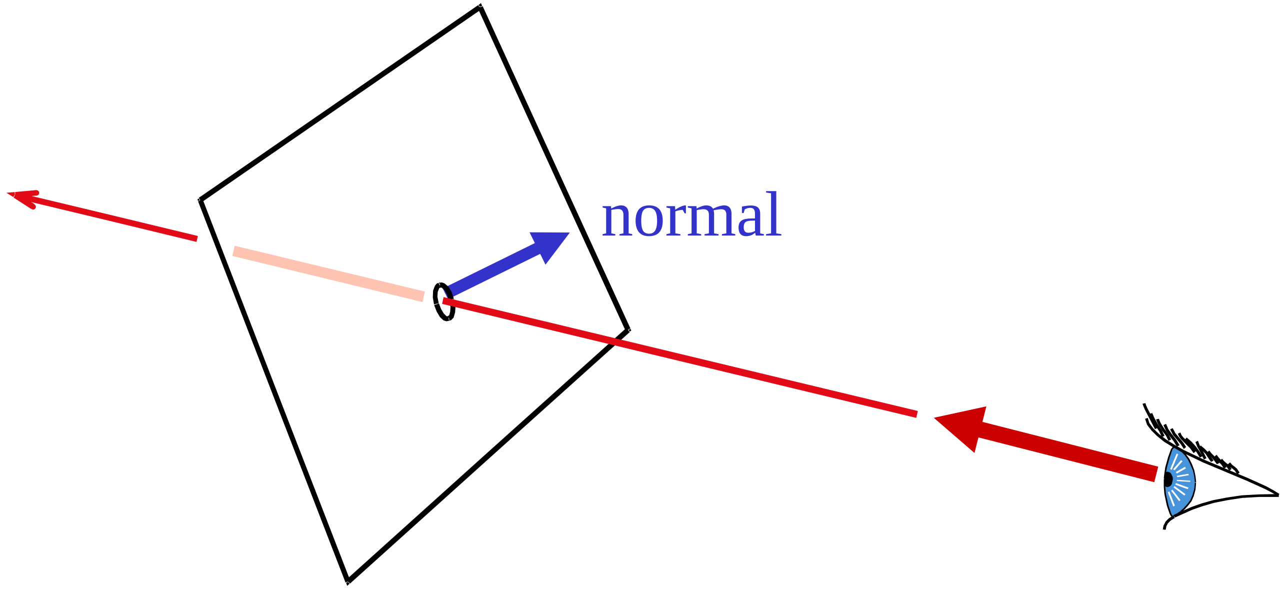
- Verify that it is not out of range (behind eye)

$$P(t) > t_{min}$$



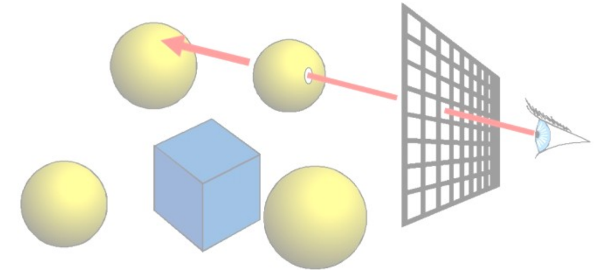
Normal

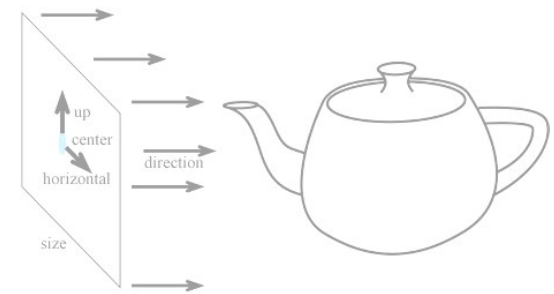
- For shading
  - diffuse: dot product between light and normal
- Normal is constant

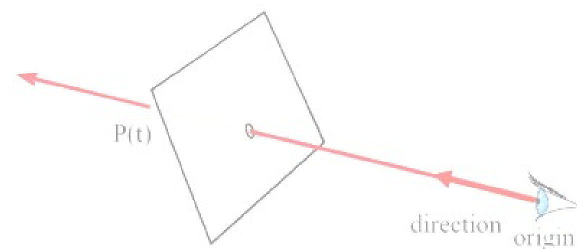


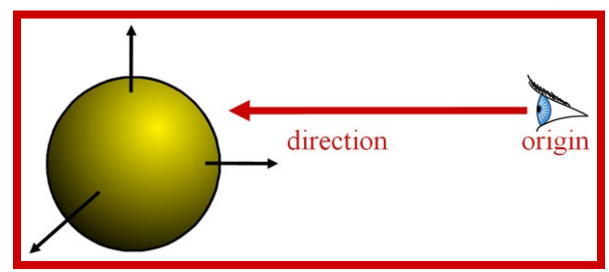
Topics

- Ray Casting Basics
- Camera and Ray Generation
- Ray Object Intersection
  - Plane
  - Sphere
  - Triangle
  - General Quadric Surface
- Recursive Ray Tracing
  - Mirror Reflection
  - Refraction



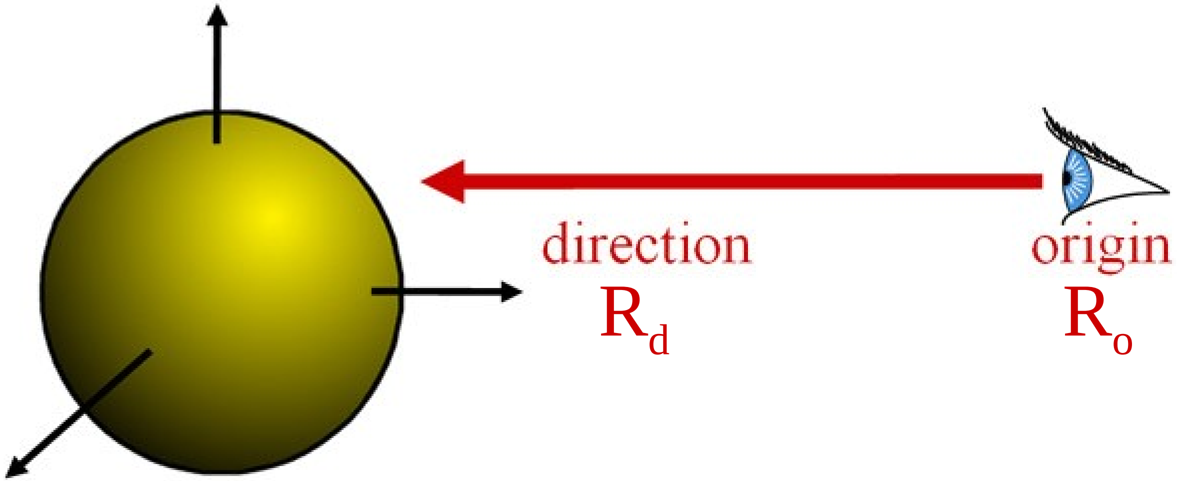






Ray-Sphere Intersection

- Sphere Representation : Implicit sphere equation
  - Assume centered at origin (easy to translate)
  - $H(P) = P \cdot P - r^2 = 0$



R_d R_o

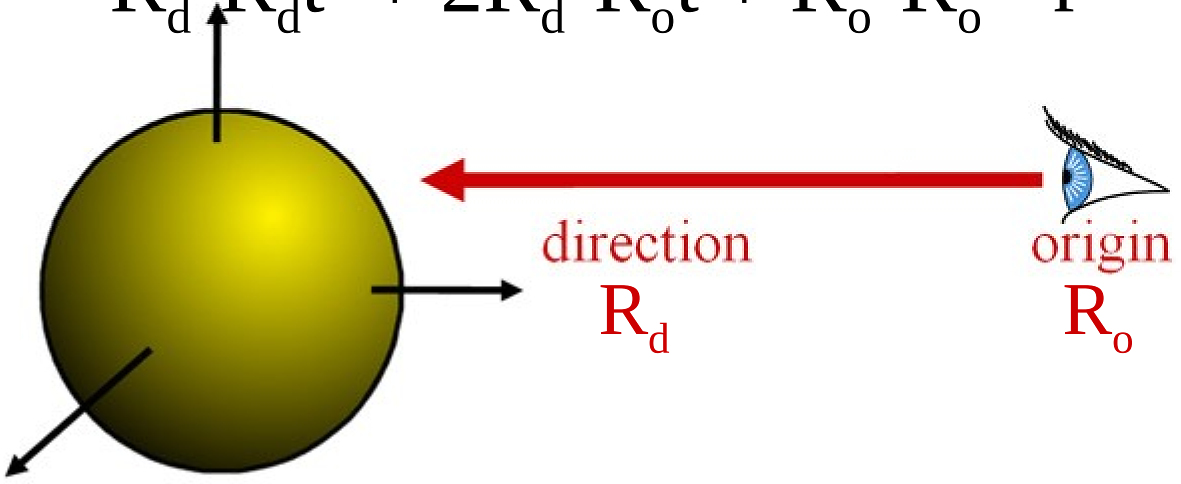
Ray-Sphere Intersection

- Insert explicit equation of ray into implicit equation of sphere & solve for t

$$P(t) = R_o + t^*R_d \qquad H(P) = P \cdot P - r^2 = 0$$

$$(R_o + tR_d) \cdot (R_o + tR_d) - r^2 = 0$$

$$R_d \cdot R_d t^2 + 2R_d \cdot R_o t + R_o \cdot R_o - r^2 = 0$$



Ray-Sphere Intersection

- Quadratic: $at^2 + bt + c = 0$
  - $a = 1$ (remember, $\|R_d\| = 1$)
  - $b = 2R_d \cdot R_o$
  - $c = R_o \cdot R_o - r^2$

$$d = \sqrt{b^2 - 4ac}$$

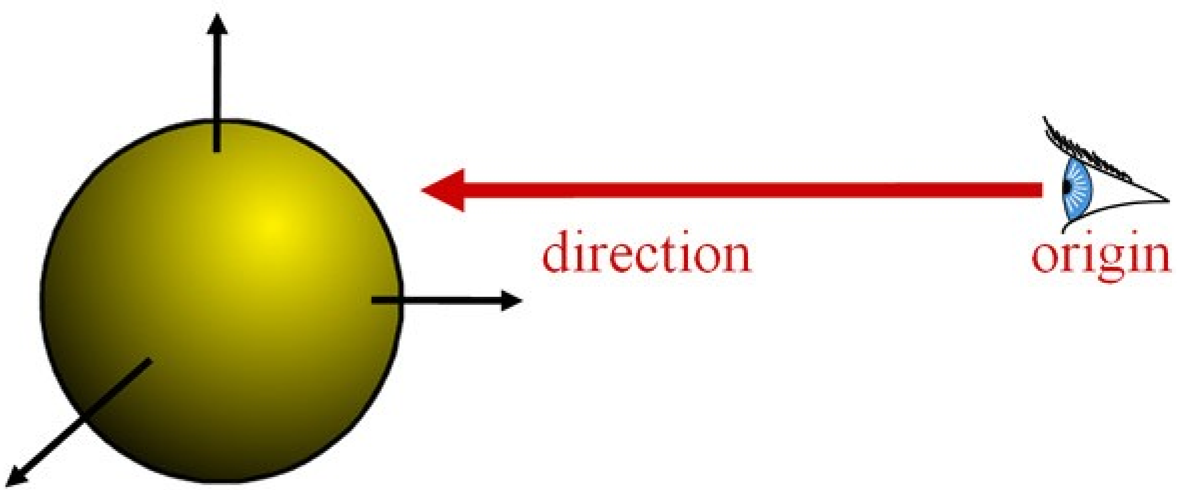
- with discriminant

$$t_{\pm} = \frac{-b \pm d}{2a}$$

- and solutions

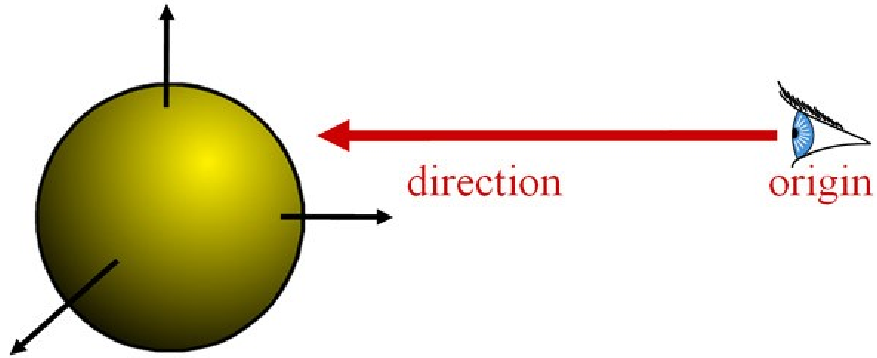
Ray-Sphere Intersection

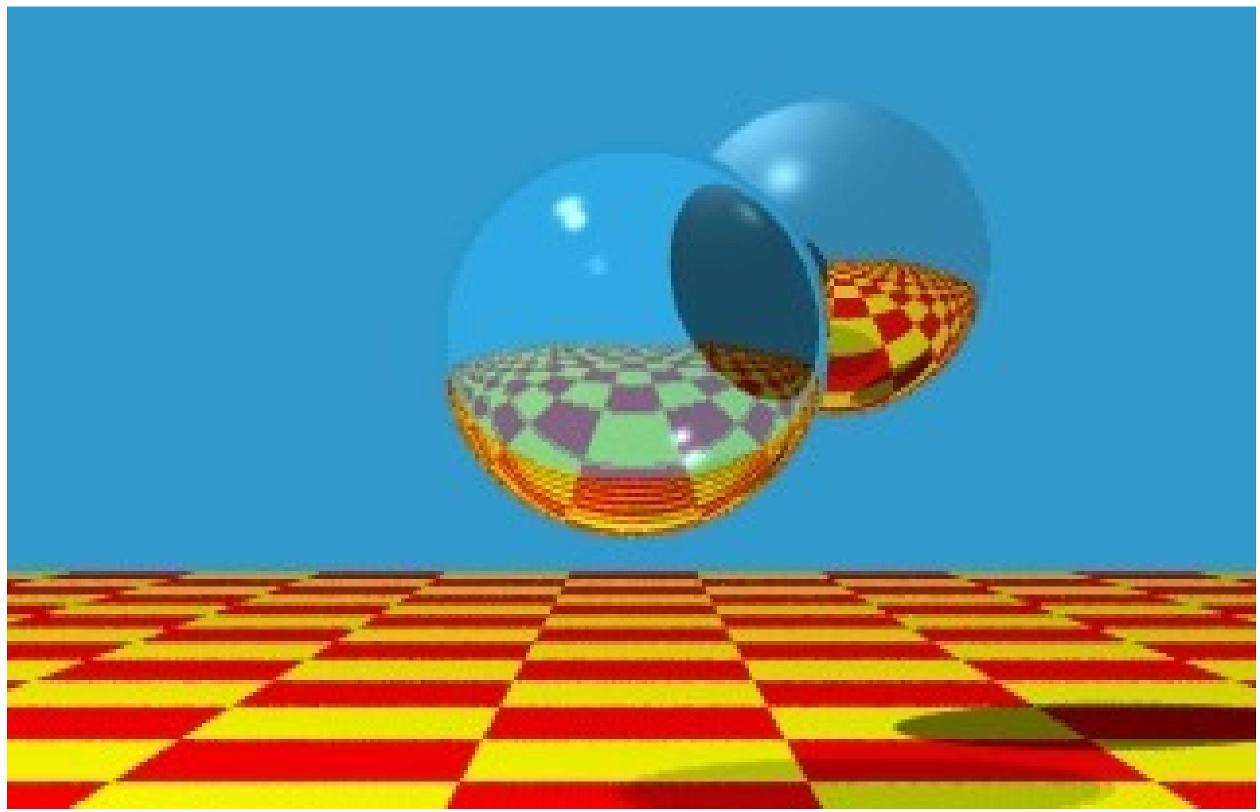
- 3 cases, depending on the sign of $b^2 - 4ac$
- What do these cases correspond to?
- Which root (t+ or t-) should you choose?
  - Closest positive! (usually t-)



Ray-Sphere Intersection

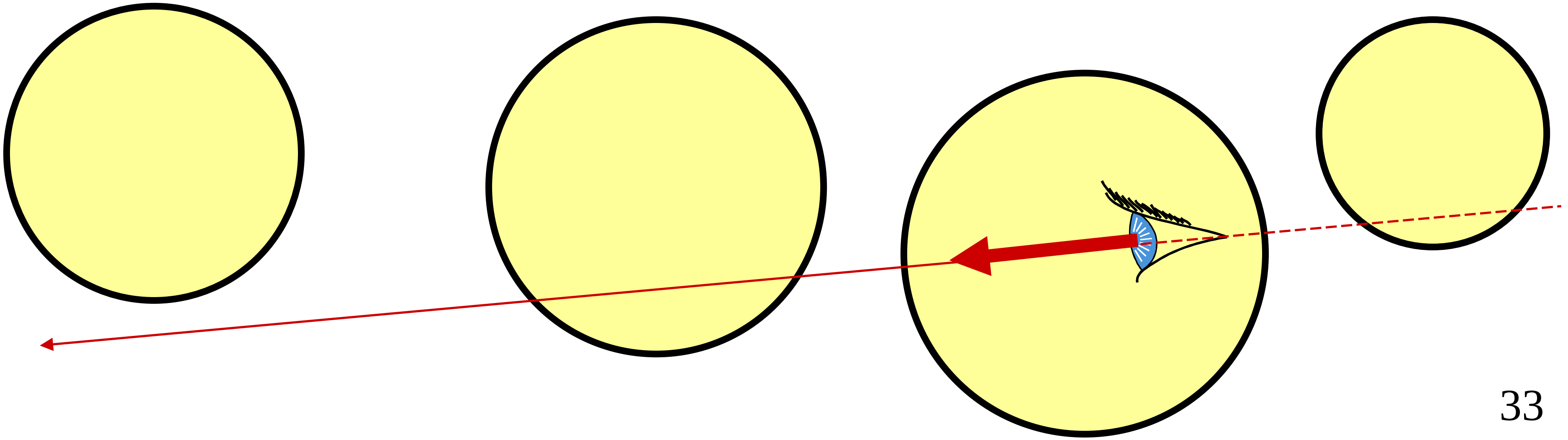
- It's so easy that all ray-tracing images have spheres!





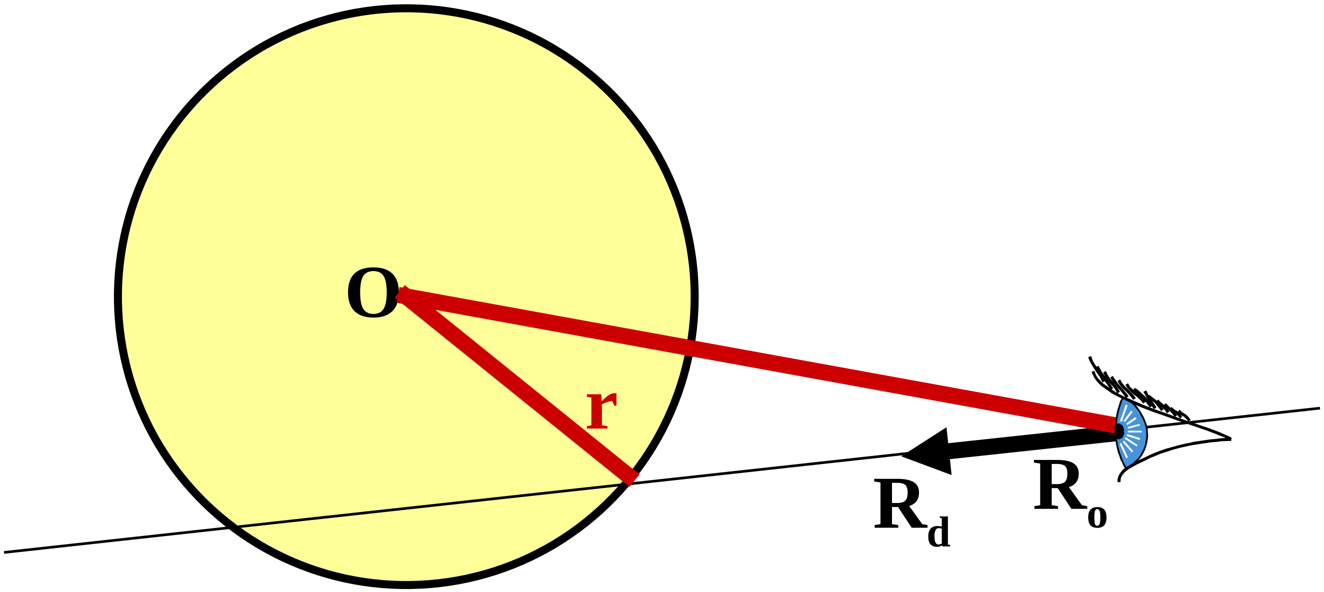
Geometric Ray-Sphere Intersection

- Shortcut / easy reject
- What geometric information is important?
  - Ray origin inside/outside sphere?
  - Closest point to ray from sphere origin?
  - Ray direction: pointing away from sphere?



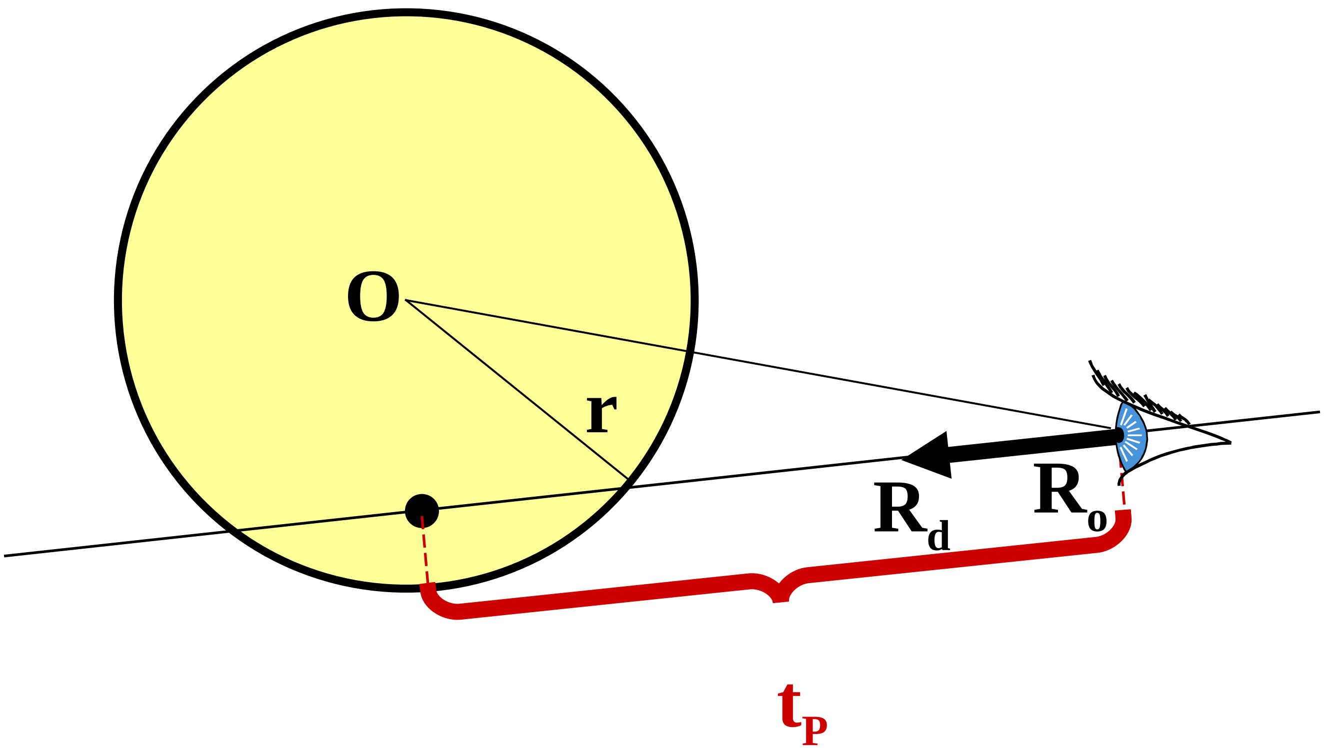
Geometric Ray-Sphere Intersection

- Is ray origin **inside/outside/on** sphere?
  - ($R_o \cdot R_o < r^2$ / $R_o \cdot R_o > r^2$ / $R_o \cdot R_o = r^2$)
  - If origin on sphere, be careful about degeneracies…



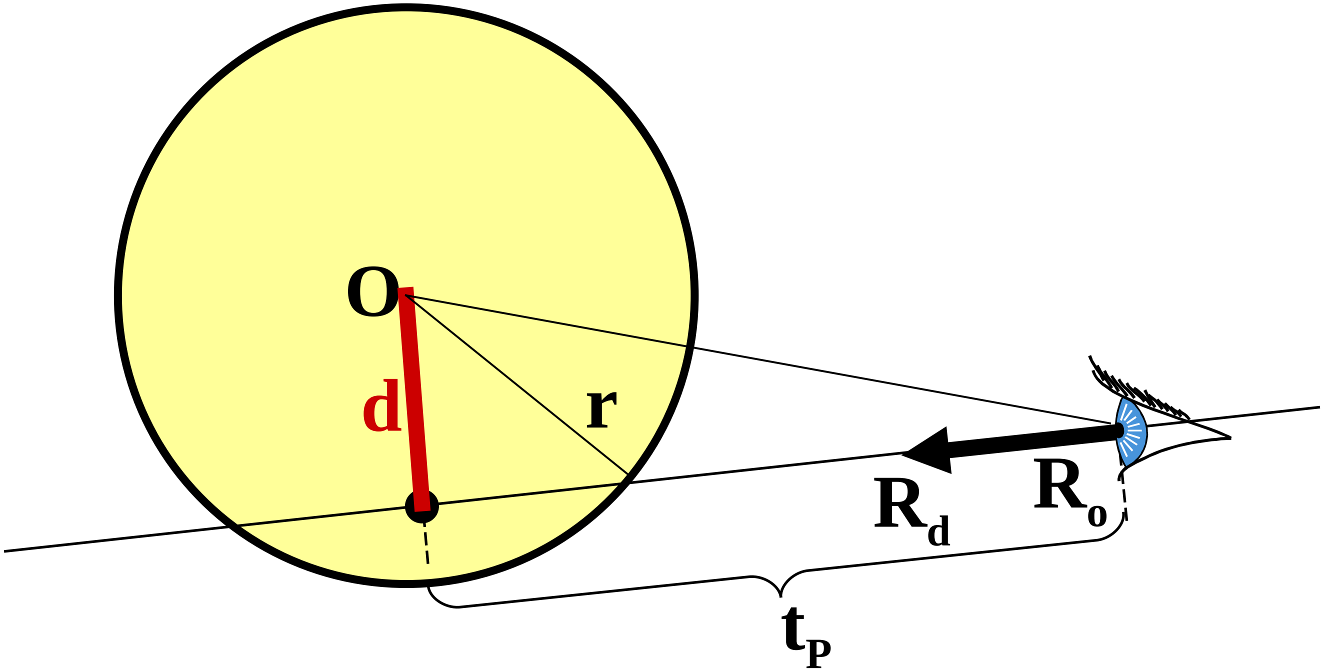
Geometric Ray-Sphere Intersection

- Is ray origin **inside/outside/on** sphere?
- Find closest point to sphere center, $\mathbf{t_P = - R_o \cdot R_d}$
  - If origin outside & $t_P < 0$ → **no hit**



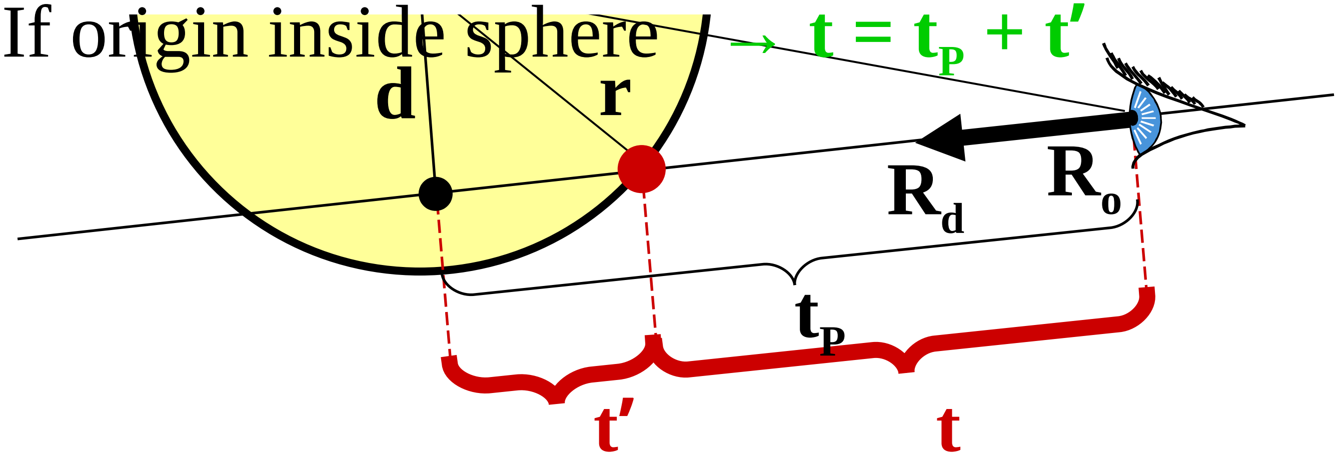
Geometric Ray-Sphere Intersection

- Is ray origin **inside/outside/on** sphere?
- Find closest point to sphere center, $\mathbf{t_P = - R_o \cdot R_d}$.
- Find squared distance, $\mathbf{d^2 = R_o \cdot R_o - t_P^2}$
  - If $d^2 > r^2$ → **no hit**



Geometric Ray-Sphere Intersection

- Is ray origin **inside/outside/on** sphere?
- Find closest point to sphere center, $\mathbf{t_P = - R_o \cdot R_d}$.
- Find squared distance: $\mathbf{d^2 = R_o \cdot R_o - t_P{}^2}$
- Find distance (t′) from closest point (t_P) to correct intersection: $\mathbf{t'^2 = r^2 - d^2}$
  - If origin outside sphere → $\mathbf{t = t_P - t'}$
  - If origin inside sphere → $\mathbf{t = t_P + t'}$

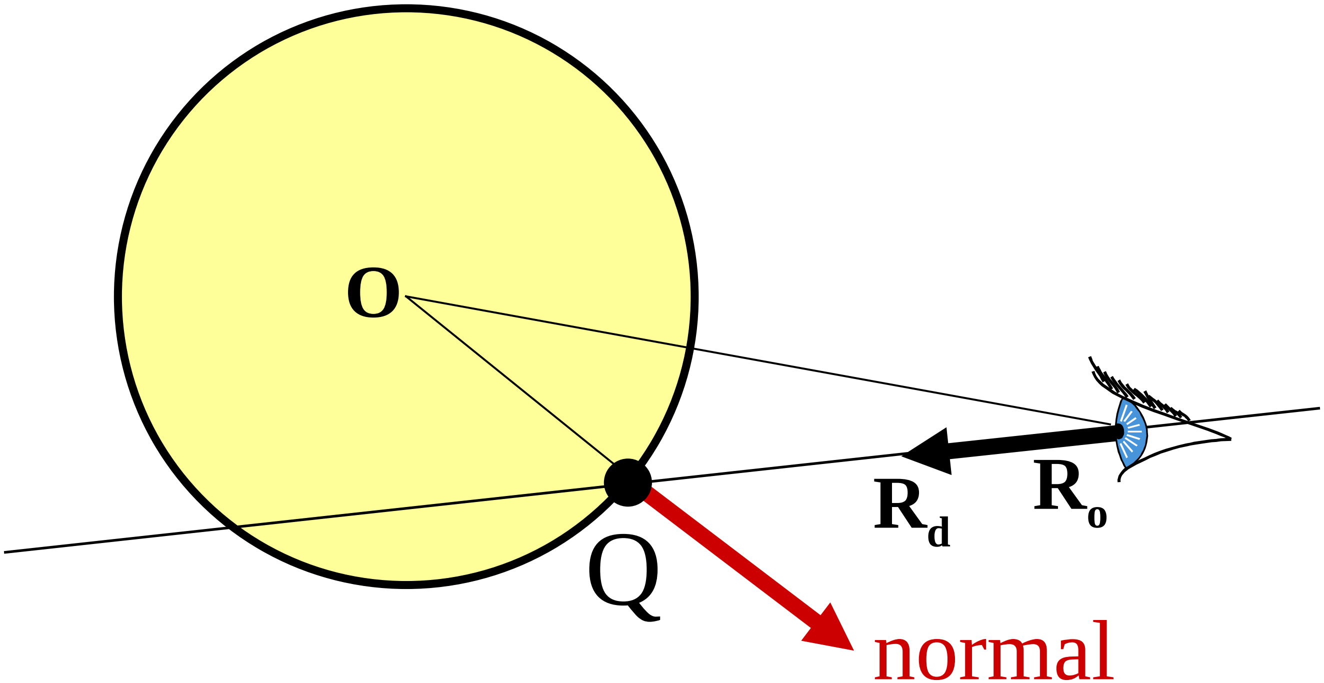


Geometric vs. Algebraic

- Algebraic is simple & generic
- Geometric is more efficient
  - Timely tests
  - In particular for rays outside and pointing away

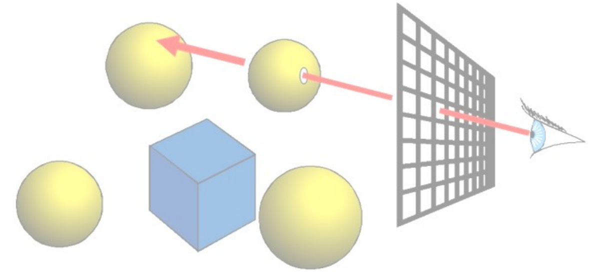
Sphere Normal

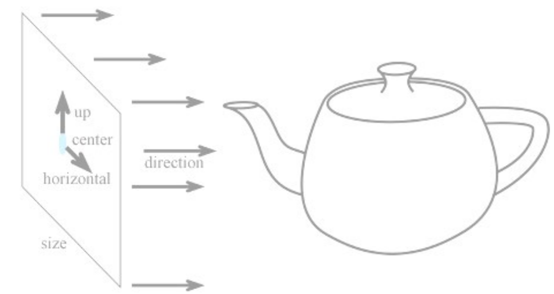
- Simply Q/||Q||
  - Q = P(t), intersection point
  - (for spheres centered at origin)

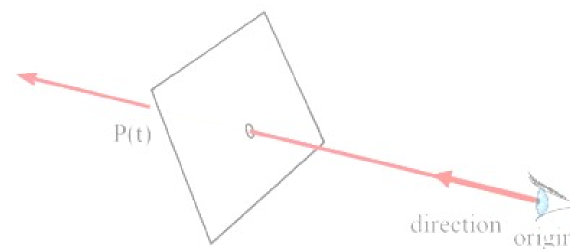


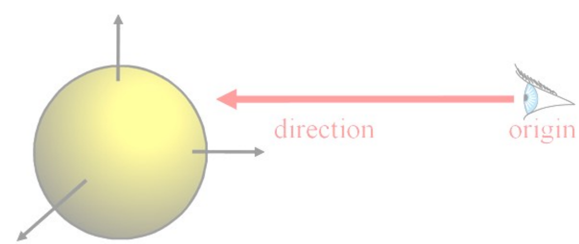
Topics

- Ray Casting Basics
- Camera and Ray Generation
- Ray Object Intersection
  - Plane
  - Sphere
  - Triangle
  - General Quadric Surface
- Recursive Ray Tracing
  - Mirror Reflection
  - Refraction



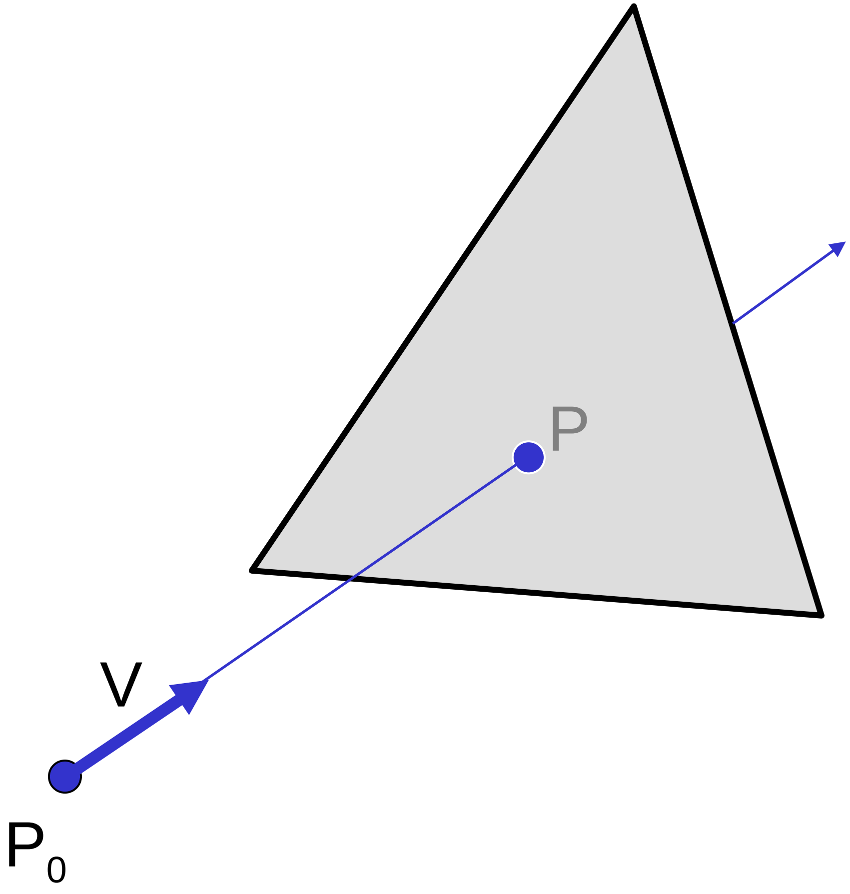






Ray-Triangle Intersection

- First, intersect ray with plane
- Then, check if point is inside triangle



Ray-Triangle Intersection

- Check if point is inside triangle parametrically

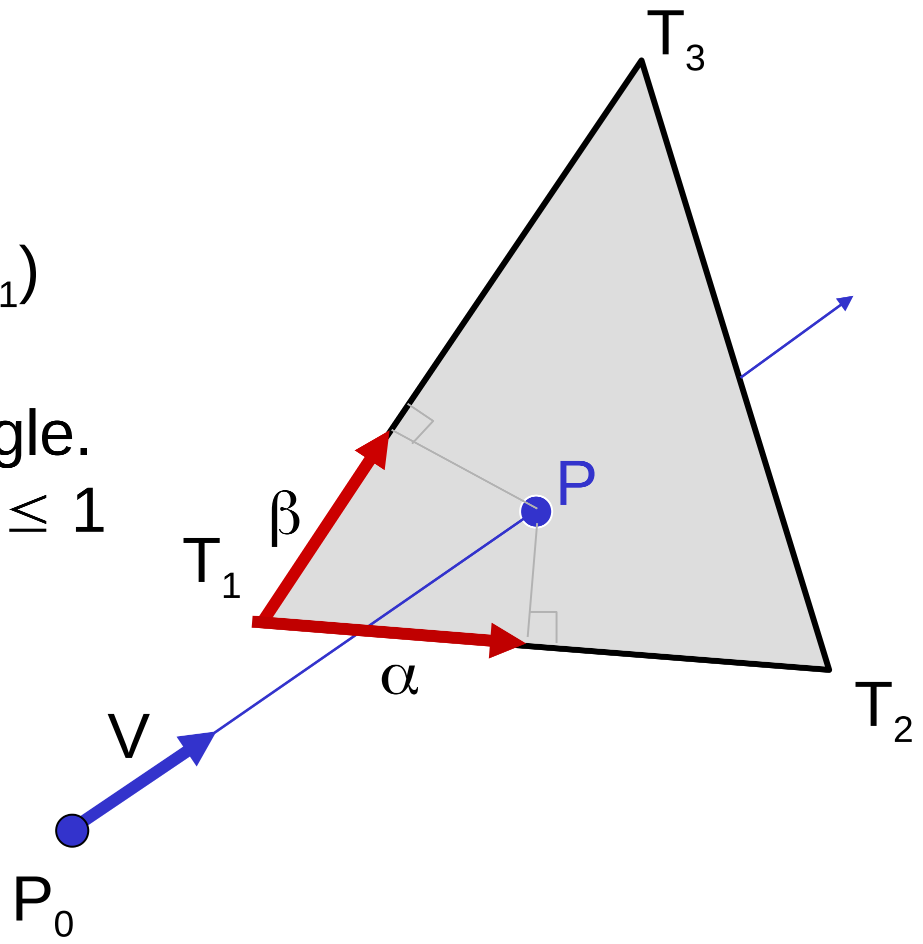
Compute α, β:

$$P = \alpha\,(T_2 - T_1) + \beta\,(T_3 - T_1)$$

Check if point inside triangle.

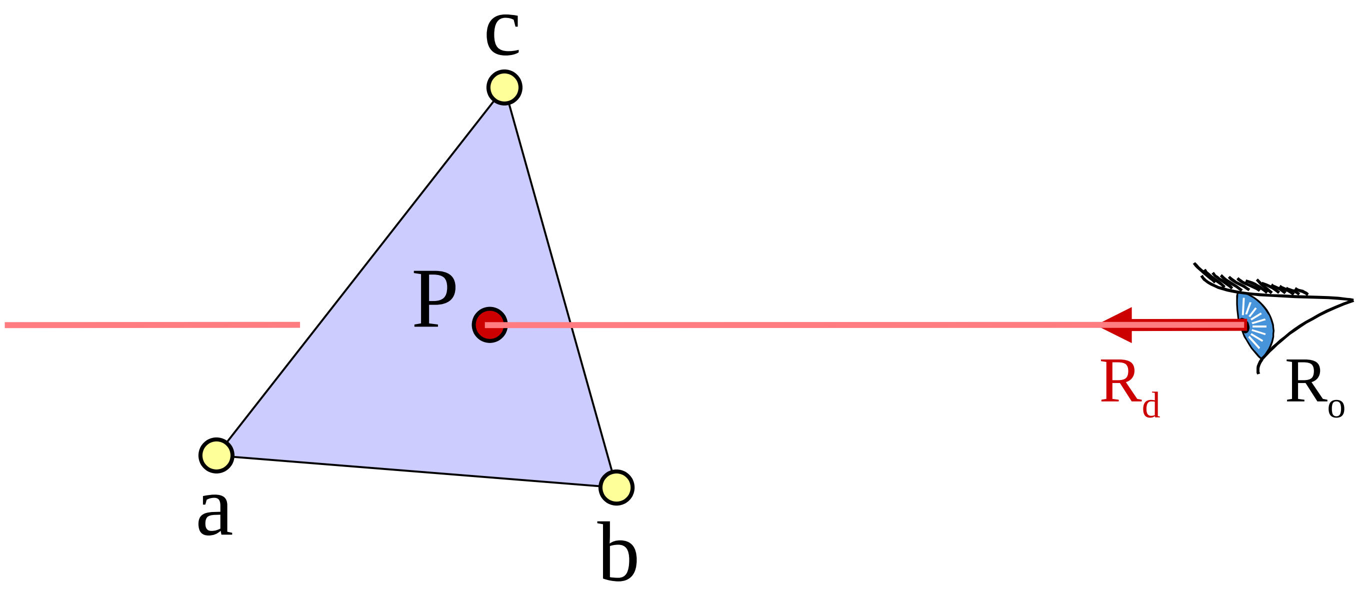
$$0 \le \alpha \le 1 \text{ and } 0 \le \beta \le 1$$

$$\alpha + \beta \le 1$$



Ray-Triangle Intersection

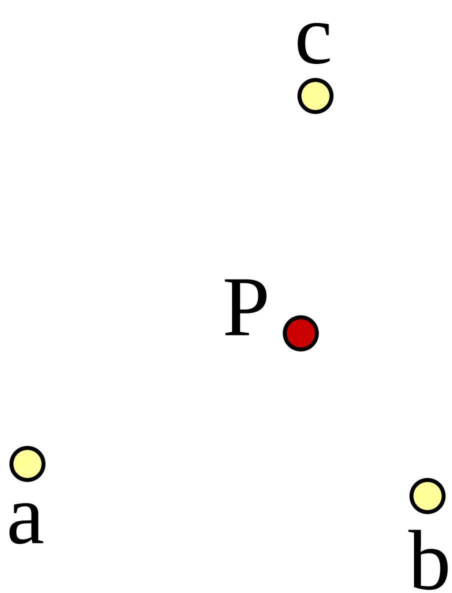
- Use general ray-polygon
- Or try to be smarter
  - Use barycentric coordinates (XM)



Barycentric Definition of a Plane

[Möbius, 1827]

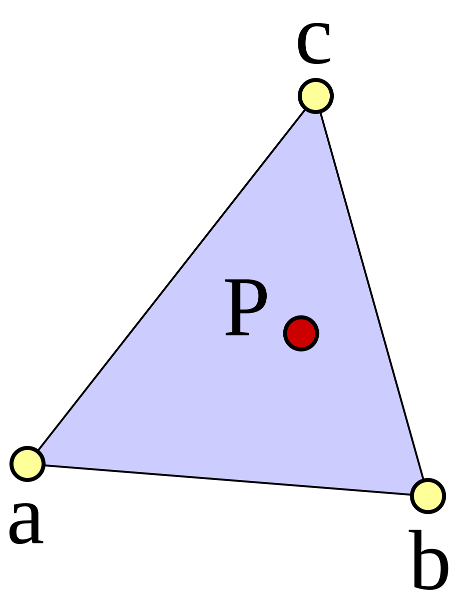
- $P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$
  with $\alpha + \beta + \gamma = 1$
- Is it explicit or implicit?



P is the *barycenter*:
the single point upon which
the plane would balance if
weights of size α, β, & γ
are
placed on points a, b, & c.

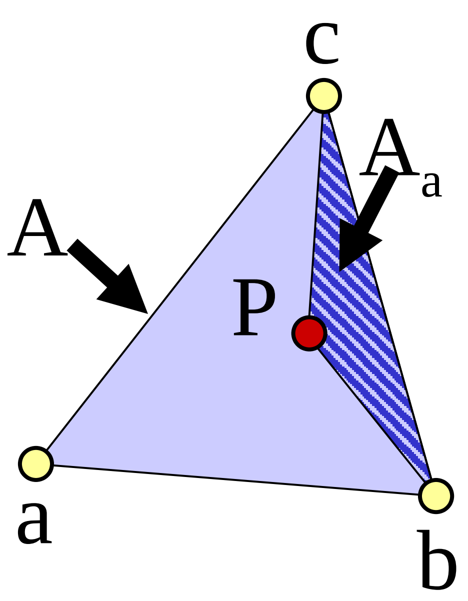
Barycentric Definition of a Triangle

- $P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$
  with $\alpha + \beta + \gamma = 1$
- AND $0 < \alpha < 1$ & $0 < \beta < 1$ & $0 < \gamma < 1$



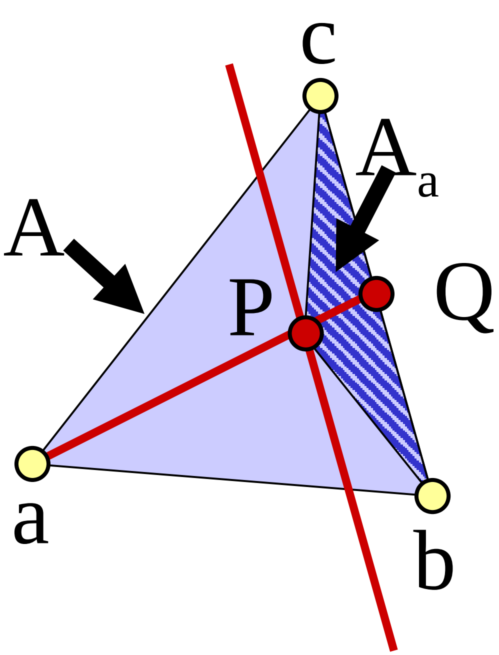
How Do We Compute α, β, γ ?

- Ratio of opposite sub-triangle area to total area
  - $\alpha = A_a/A \qquad \beta = A_b/A \qquad \gamma = A_c/A$
- Use signed areas for points outside the triangle



Intuition Behind Area Formula

- P is barycenter of a and Q
- A_a is the interpolation coefficient on $\overline{aQ}$
- All points on lines parallel to $\overline{bc}$ have the same α (All such triangles have same height/area)



Simplify

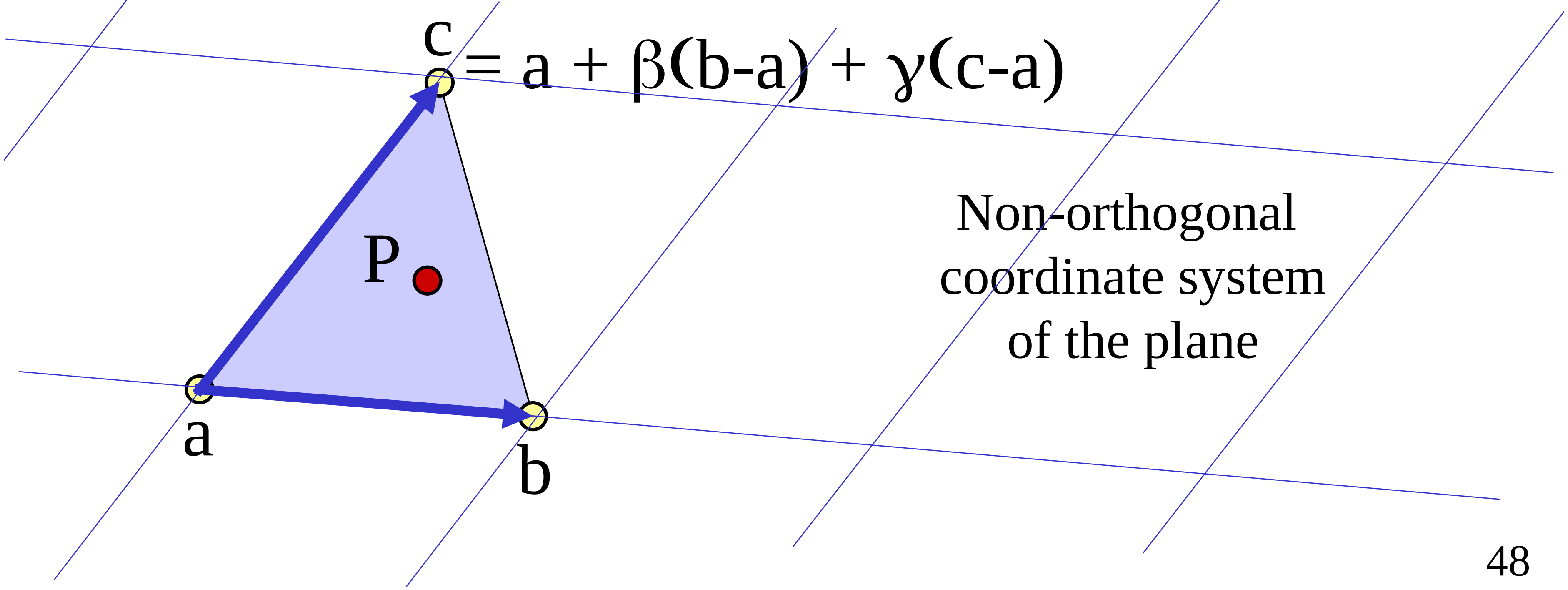
- Since $\alpha + \beta + \gamma = 1$, we can write $\alpha = 1 - \beta - \gamma$

rewrite

$$P(\alpha, \beta, \gamma) = \alpha a + \beta b + \gamma c$$

$$P(\beta, \gamma) \quad = (1-\beta-\gamma)a + \beta b + \gamma c$$

$$= a + \beta(b-a) + \gamma(c-a)$$



Intersection with Barycentric Triangle

- Set ray equation equal to barycentric equation

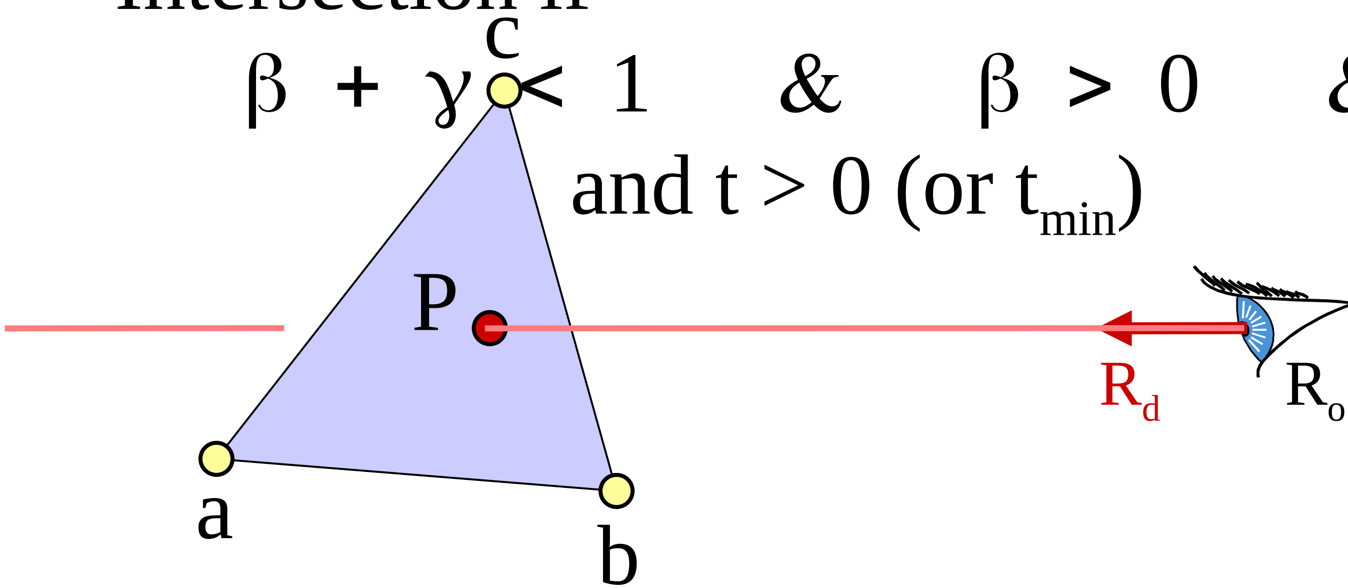
$$P(t) = P(\beta, \gamma)$$

$$R_o + t * R_d = a + \beta(b-a) + \gamma(c-a)$$

- Intersection if

$$\beta + \gamma < 1 \quad \& \quad \beta > 0 \quad \& \quad \gamma > 0$$

and $t > 0$ (or t_{min})



Intersection with Barycentric Triangle

- $R_o + t * R_d = a + \beta(b\text{-}a) + \gamma(c\text{-}a)$

$$\left.\begin{aligned} R_{ox} + tR_{dx} &= a_x + \beta(b_x - a_x) + \gamma(c_x - a_x) \\ R_{oy} + tR_{dy} &= a_y + \beta(b_y - a_y) + \gamma(c_y - a_y) \\ R_{oz} + tR_{dz} &= a_z + \beta(b_z - a_z) + \gamma(c_z - a_z) \end{aligned}\right\}$$

3 equations,
3 unknowns

- Regroup & write in matrix form:

$$\begin{bmatrix} a_x - b_x & a_x - c_x & R_{dx} \\ a_y - b_y & a_y - c_y & R_{dy} \\ a_z - b_z & a_z - c_z & R_{dz} \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} a_x - R_{ox} \\ a_y - R_{oy} \\ a_z - R_{oz} \end{bmatrix}$$

Cramer's Rule

- Used to solve for one variable at a time in system of equations

$$\beta = \frac{\begin{vmatrix} a_x - R_{ox} & a_x - c_x & R_{dx} \\ a_y - R_{oy} & a_y - c_y & R_{dy} \\ a_z - R_{oz} & a_z - c_z & R_{dz} \end{vmatrix}}{|A|} \qquad \gamma = \frac{\begin{vmatrix} a_x - b_x & a_x - R_{ox} & R_{dx} \\ a_y - b_y & a_y - R_{oy} & R_{dy} \\ a_z - b_z & a_z - R_{oz} & R_{dz} \end{vmatrix}}{|A|}$$

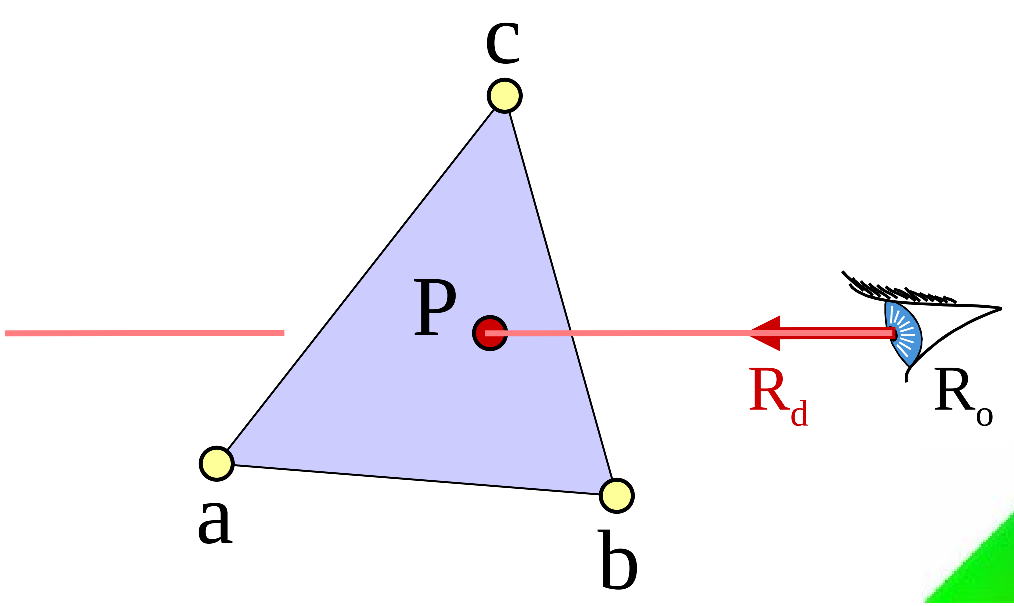
$$t = \frac{\begin{vmatrix} a_x - b_x & a_x - c_x & a_x - R_{ox} \\ a_y - b_y & a_y - c_y & a_y - R_{oy} \\ a_z - b_z & a_z - c_z & a_z - R_{oz} \end{vmatrix}}{|A|}$$

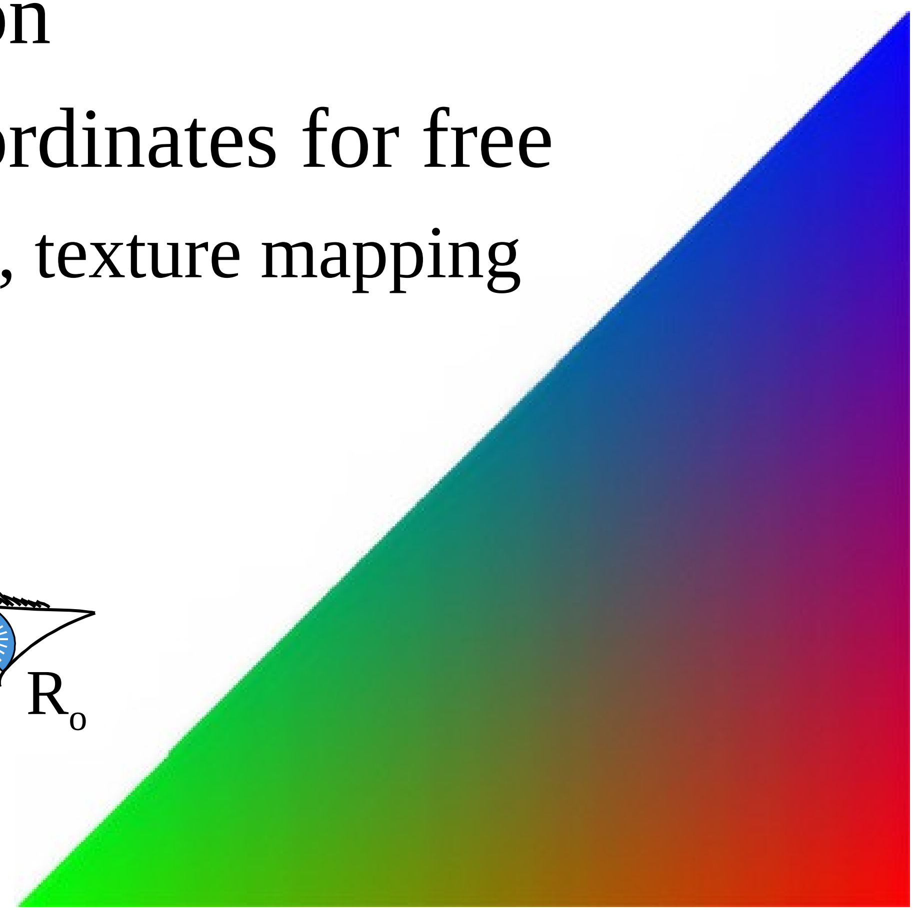
| | denotes the determinant

Can be copied mechanically into code

Advantages of Barycentric Intersection

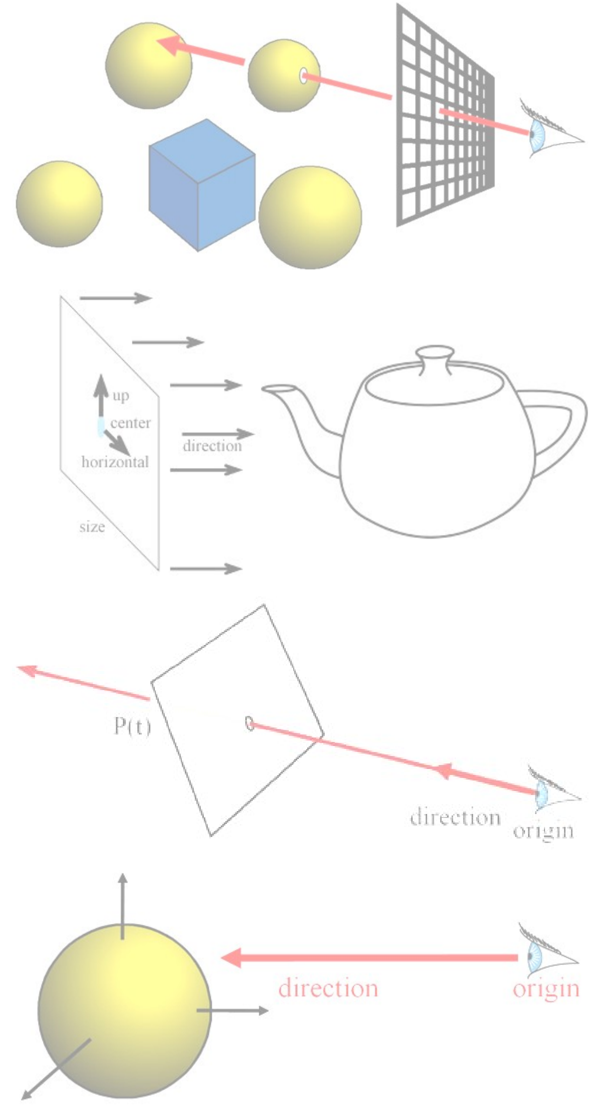
- Efficient
- Stores no plane equation
- Get the barycentric coordinates for free
  - Useful for interpolation, texture mapping





Topics

- Ray Casting Basics
- Camera and Ray Generation
- Ray Object Intersection
  - Plane
  - Sphere
  - Triangle
  - General Quadric Surface
- Recursive Ray Tracing
  - Mirror Reflection
  - Refraction



General Quadric Surfaces

- Some Common Quadric Surfaces
  - Ellipsoid $(x^2/a^2 + y^2/b^2 + z^2/c^2 + 1 = 0)$
  - Cone $(x^2/a^2 - y^2/b^2 + z^2/c^2 = 0)$
  - Cylinder
  - Hyperboloid
  - Paraboloid – Eliptic, Hyperbolic etc.

* Check out the following links for the figures & equations:

https://tutorial.math.lamar.edu/Classes/CalcIII/QuadricSurfaces.aspx

https://mrl.cs.nyu.edu/~dzorin/rend05/lecture2.pdf (page12-14)

Ray - Quadric Surface Intersection

- General Quadric Surface Equation

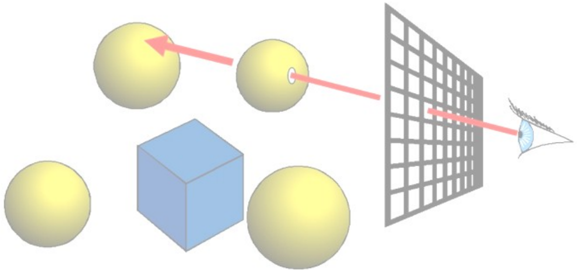
$Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fxz + Gx + Hy + Iz + J = 0 \ldots \ldots (1)$

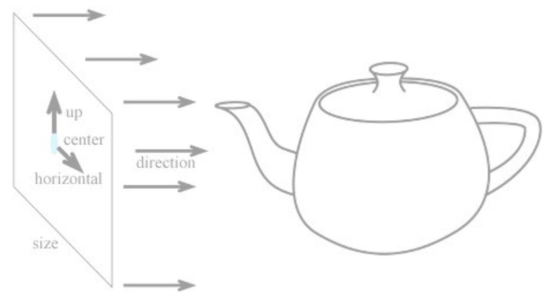
- Ray Equation : $P(t) = \mathbf{R_o} + t * \mathbf{R_d}$
- So, $P_x = R_{0x} + t*R_{dx}$, Similar for P_y, P_z
- Put P_x, P_y, P_z as x, y, z in eq.(1) and solve for t
- Accept the smaller non –ve real value of t

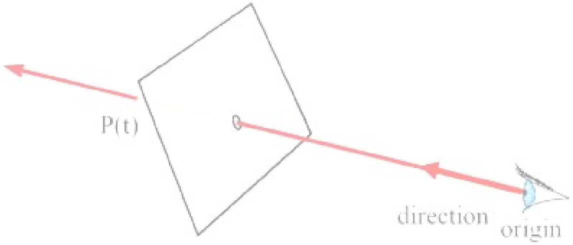
- General Quadric Surface Normal
  - Use partial derivatives!

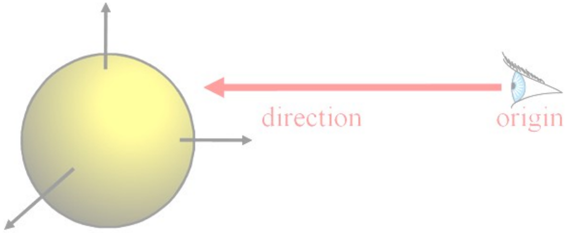
Topics

- Ray Casting Basics
- Camera and Ray Generation
- Ray Object Intersection
  - Plane
  - Sphere
  - Triangle
  - General Quadric Surface
- Recursive Ray Tracing
  - Mirror Reflection
  - Refraction



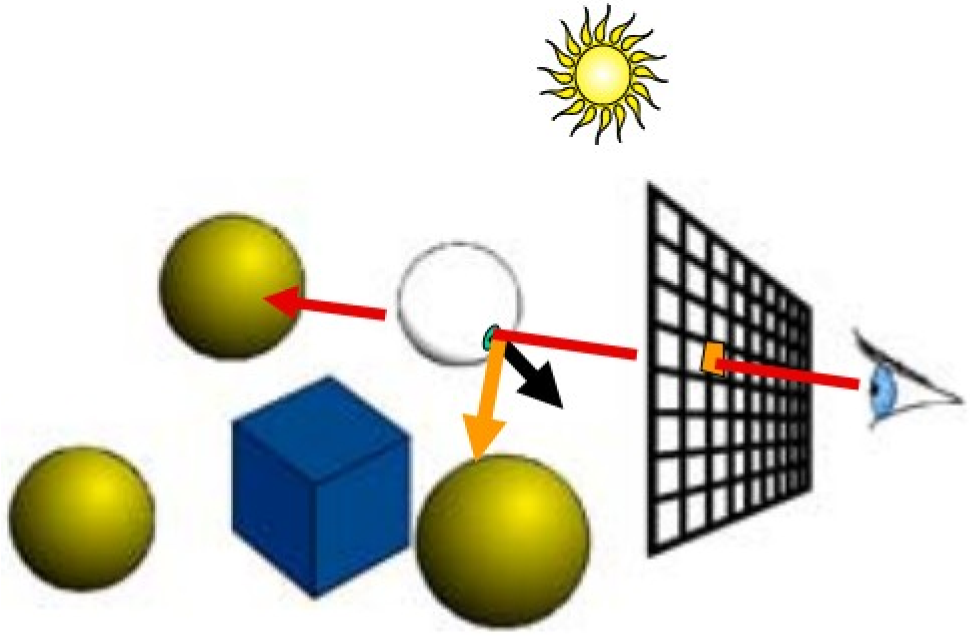






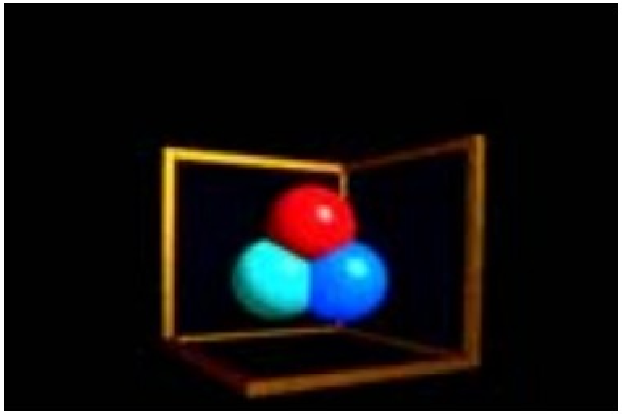
Mirror Reflection

- Compute reflected ray – according to law of reflection on an ideal mirror
- Include contribution of this reflection in color

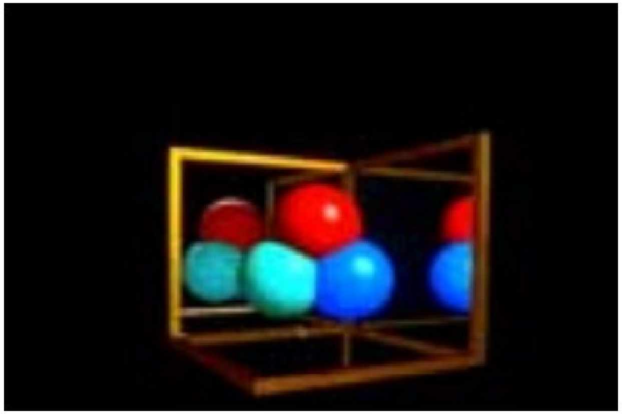


Mirror Reflection

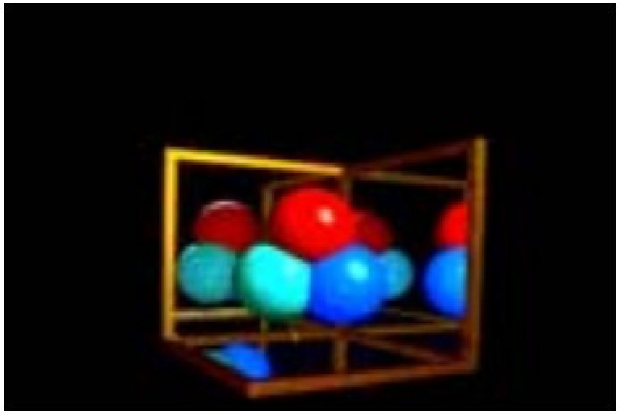
- Depth of recursion in reflection



Depth = 0



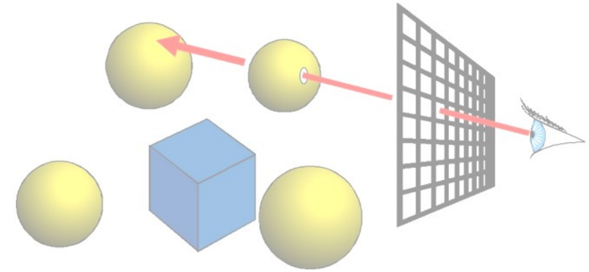
Depth = 1

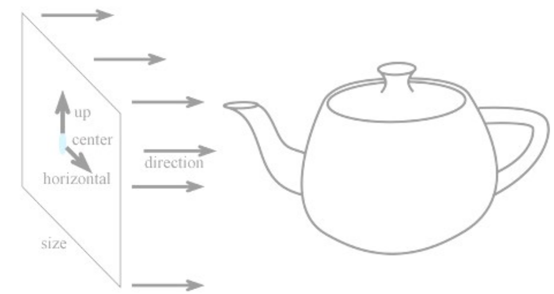


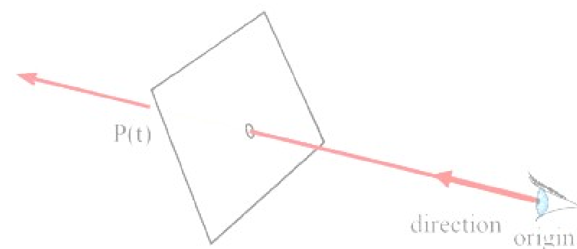
Depth = 2

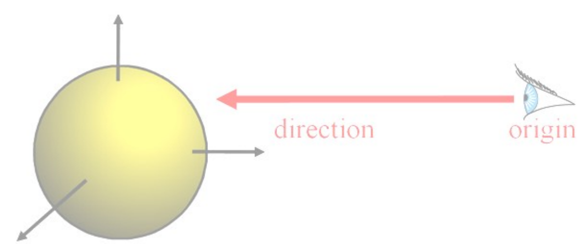
Topics

- Ray Casting Basics
- Camera and Ray Generation
- Ray Object Intersection
  - Plane
  - Sphere
  - Triangle
  - General Quadric Surface
- Recursive Ray Tracing
  - Mirror Reflection
  - Refraction



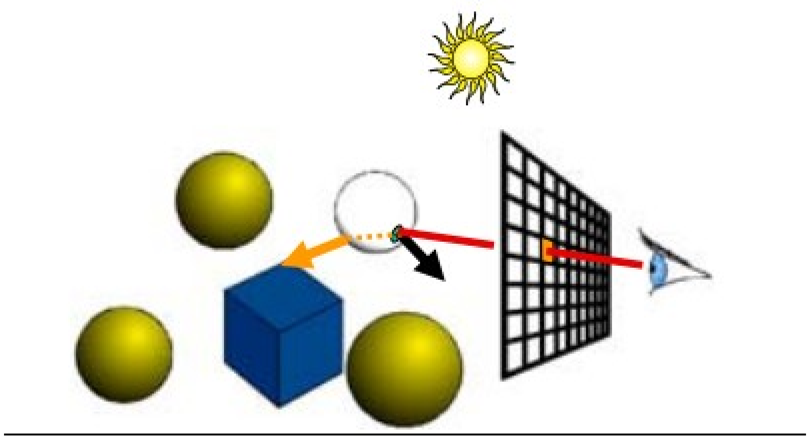


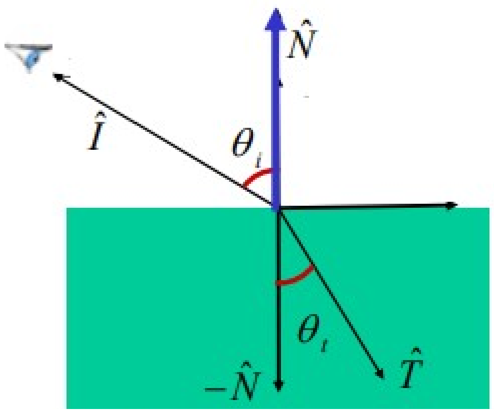




Refraction

- Compute refracted ray – according to law of refraction





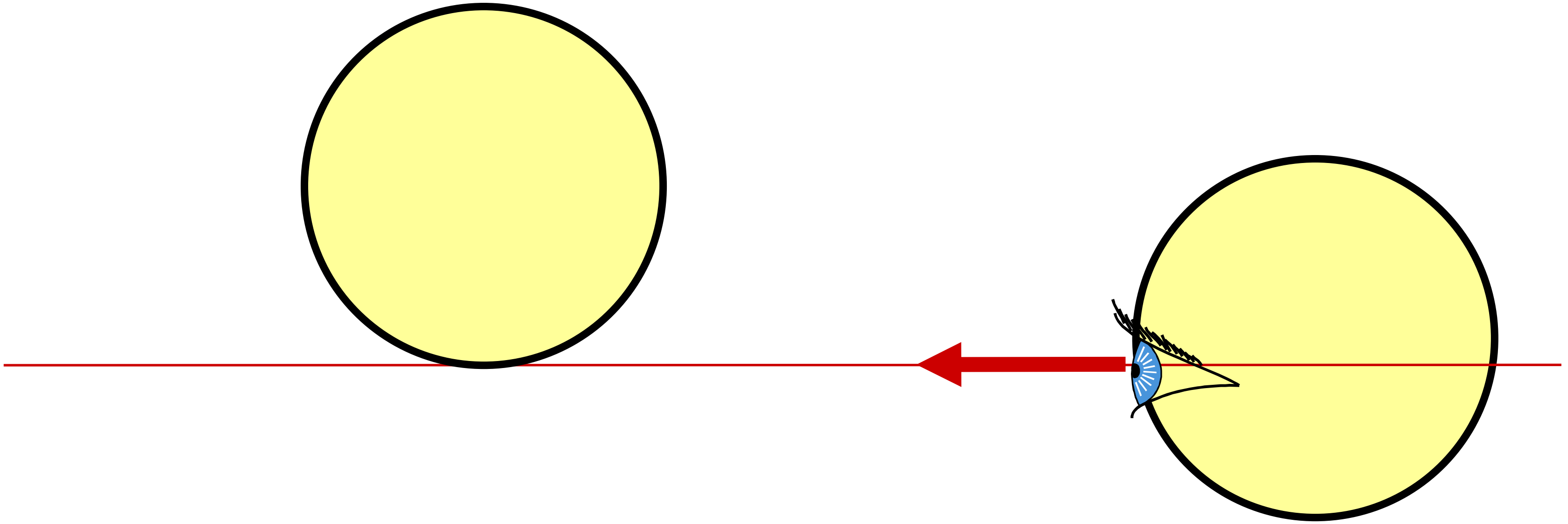
$$\boxed{\frac{\sin\theta_i}{\sin\theta_t} = \frac{\eta_t}{\eta_i} = \eta_r}$$

Base Case of Recursive Ray Tracing?

- A fixed no. of times (controlling recursion depth)
- When ray contribution becomes insignificant
- Both

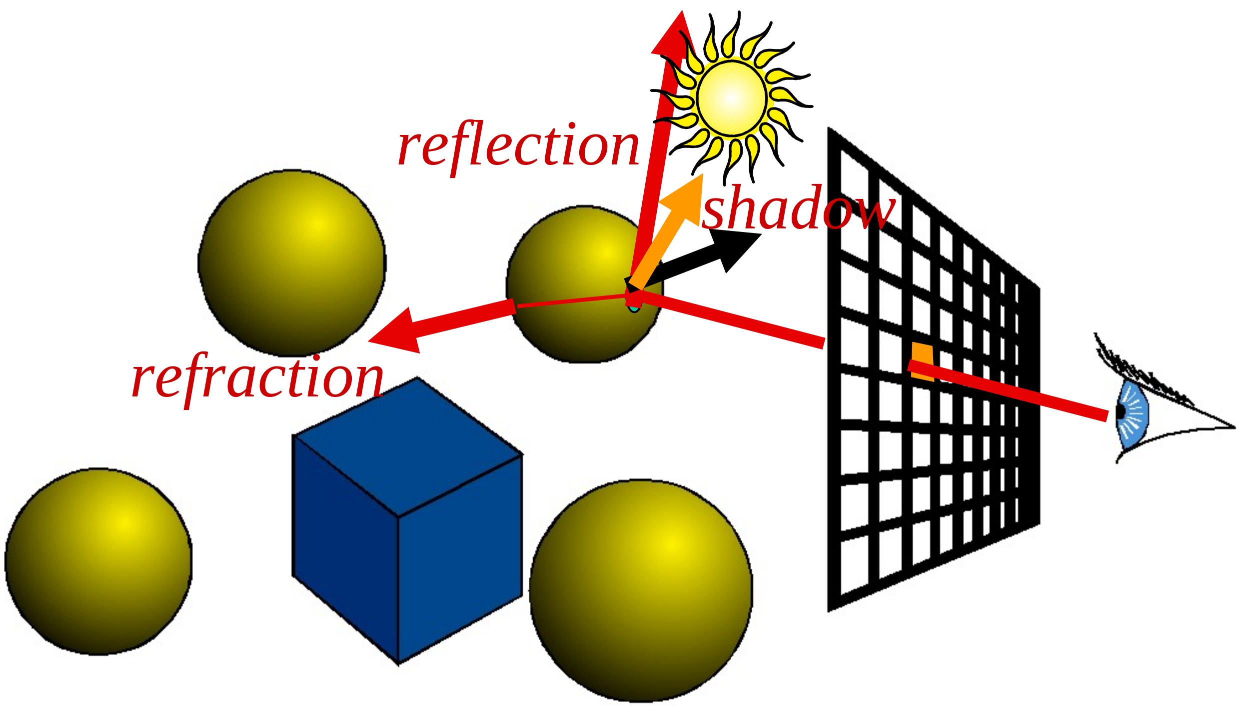
Precision Issues

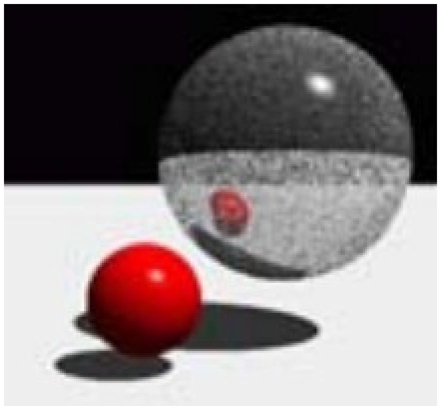
- What happens when
  - Origin is on an object?
    - Self shadowing!!
    - Self reflection!!
  - Grazing rays?
- Problem with floating-point approximation



The Evil ε

- In ray tracing, do NOT report intersection for rays starting at the surface (no false positive)
  - Because secondary rays requires epsilons

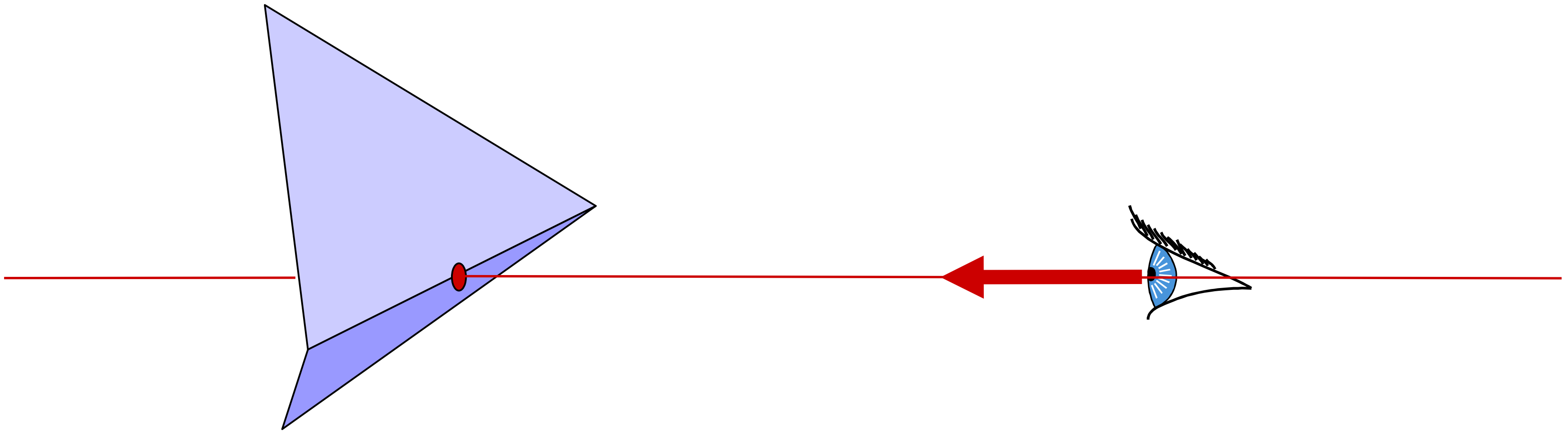




Without epsilon

The Evil ε : Challenges

- Edges in triangle meshes
  - Must report intersection (otherwise not watertight)
  - No false negative



References

- Textbook : Fundamentals of Computer Graphics (3rd edition) by Peter Shirley, Steve Marschner et. al – Chapter 4 (Extra – Chapter 13) *[May be insufficient]*
- Computer Graphics – MIT OpenCourseWare : Lecture 11, 12, 13 (Extra resource – Lecture 14) https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/
- https://tutorial.math.lamar.edu/Classes/CalcIII/QuadricSurfaces.aspx
- https://mrl.cs.nyu.edu/~dzorin/rend05/lecture2.pdf
- http://www.cs.tau.ac.il/~dcor/Graphics/adv-slides/ray-tracing06.pdf
- Shirley P., M. Ashikhmin and S. Marschner, *Fundamentals of Computer Graphics*
- Shirley P. and R.K. Morley, Realistic Ray Tracing
- Jensen H.W., Realistic Image Synthesis Using Photon Mapping

Thank you ☺