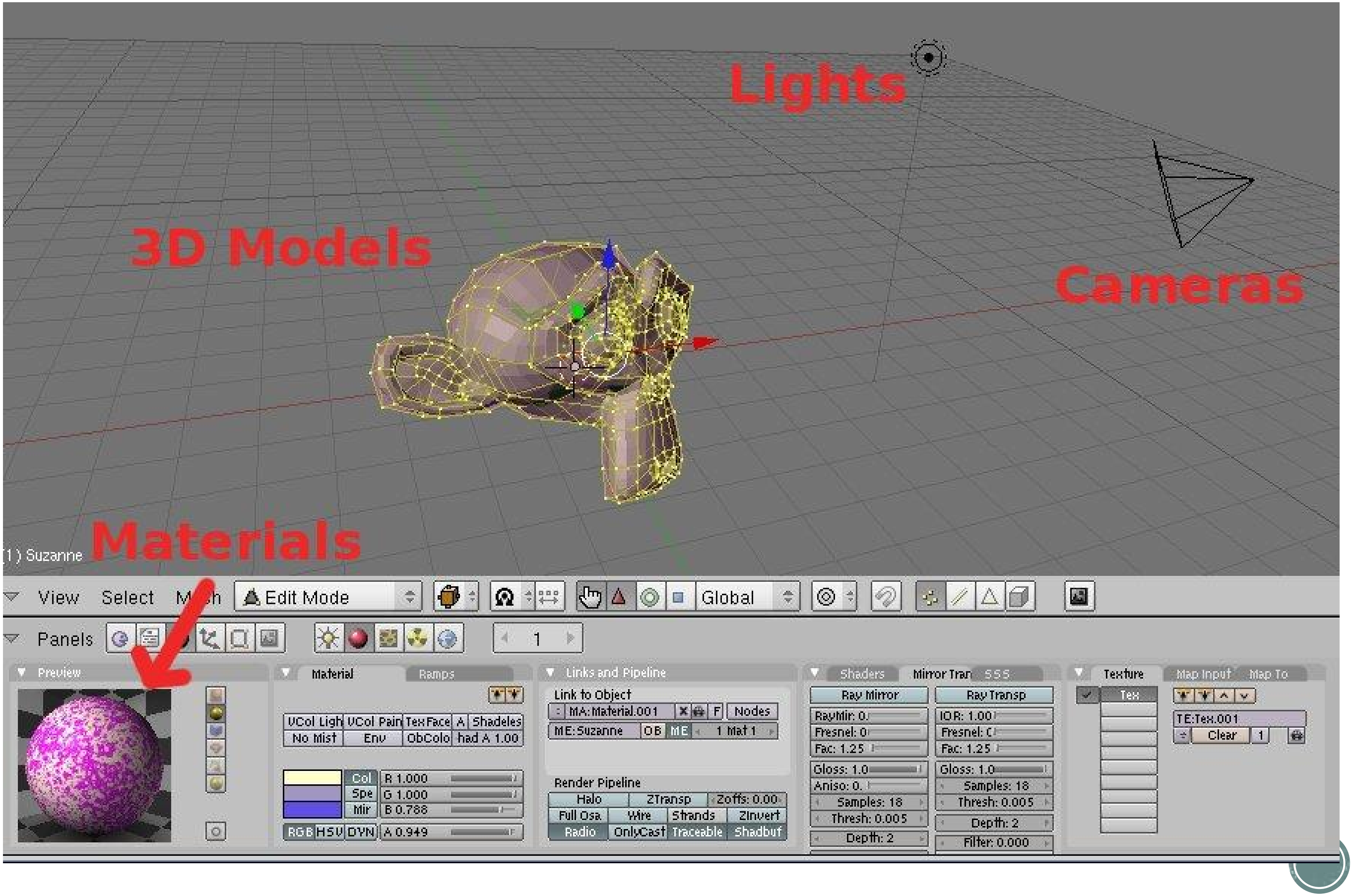
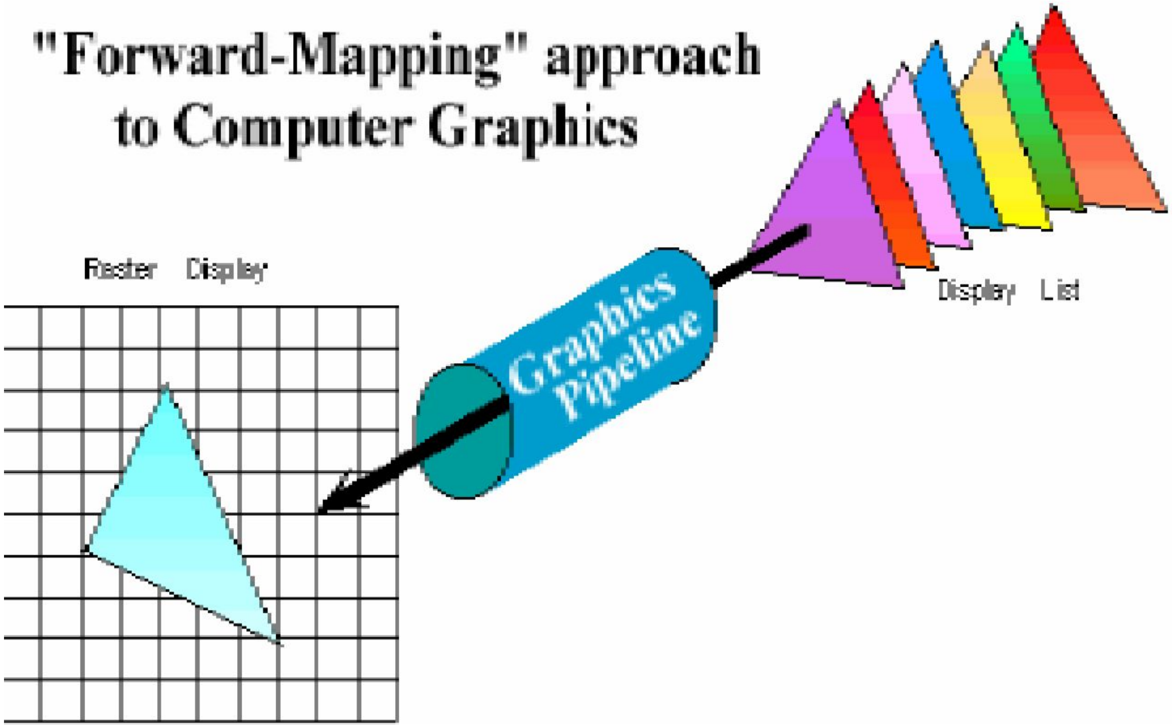
Advanced image synthesis

Which color in each pixel?

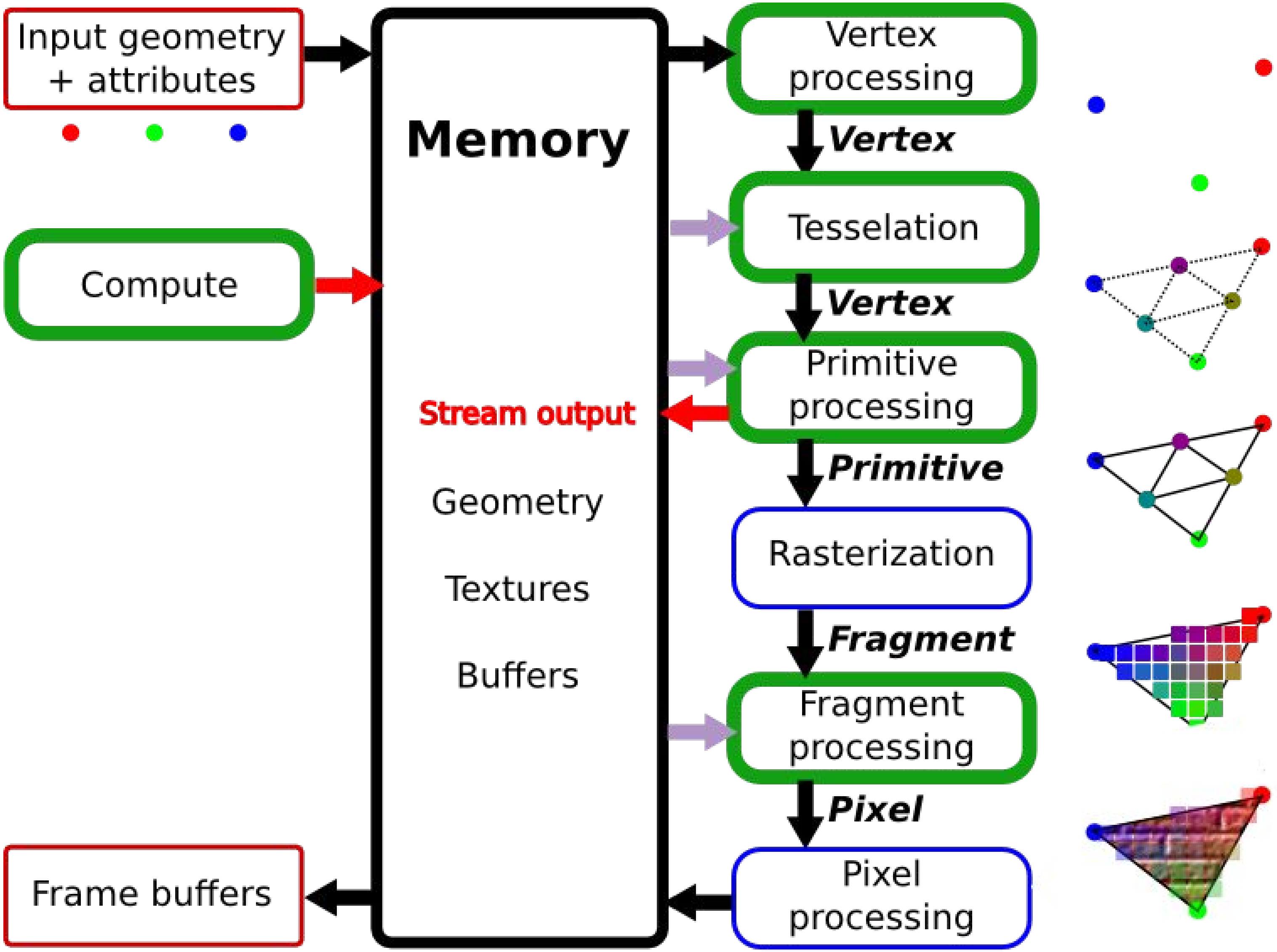


Rasterization pipeline

- For each triangle
  - Project triangle to image plane
  - For each pixel
    - Check pixel in triangle
    - Resolve visibility with z-buffer



Modern graphics pipeline



Input geometry + attributes

Compute

Frame buffers

Memory

Stream output

Geometry

Textures

Buffers

Vertex processing

Vertex

Tesselation

Vertex

Primitive processing

Primitive

Rasterization

Fragment

Fragment processing

Pixel

Pixel processing

Rasterization advantages

- Modern scenes more complicated than images
  - 1920x1080 frame (1080p)
  - 64-bit color and 32-bit depth
  - 24 Mb memory
- Rasterization can stream over triangles
  - One triangle at a time
  - Parallelism
  - Memory optimization

Rasterization limitations

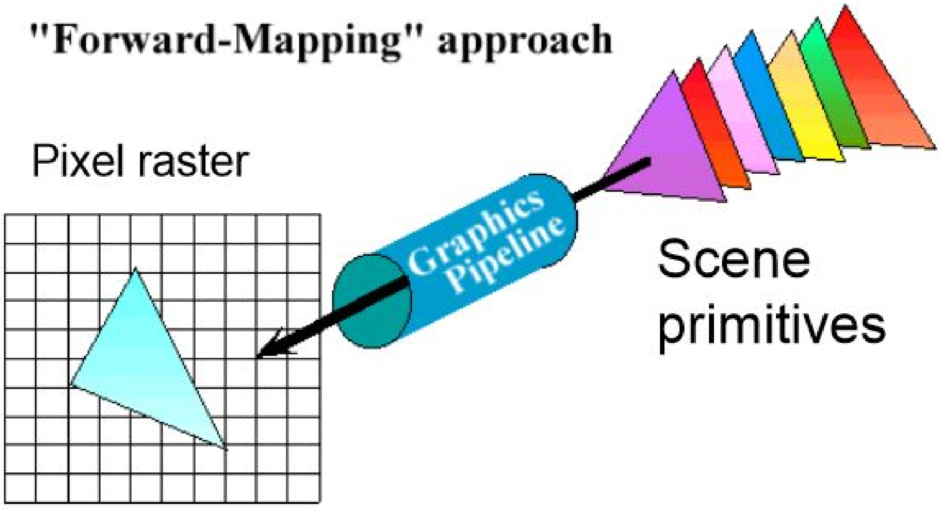
- Restricted to scan-convertible primitives (triangles)

- No unified handling of
  - Shadows
  - Reflection
  - Transparency

- Potential problem of overdraw
  - Depth complexity
  - Each pixel touched many times

Rasterization VS ray-casting

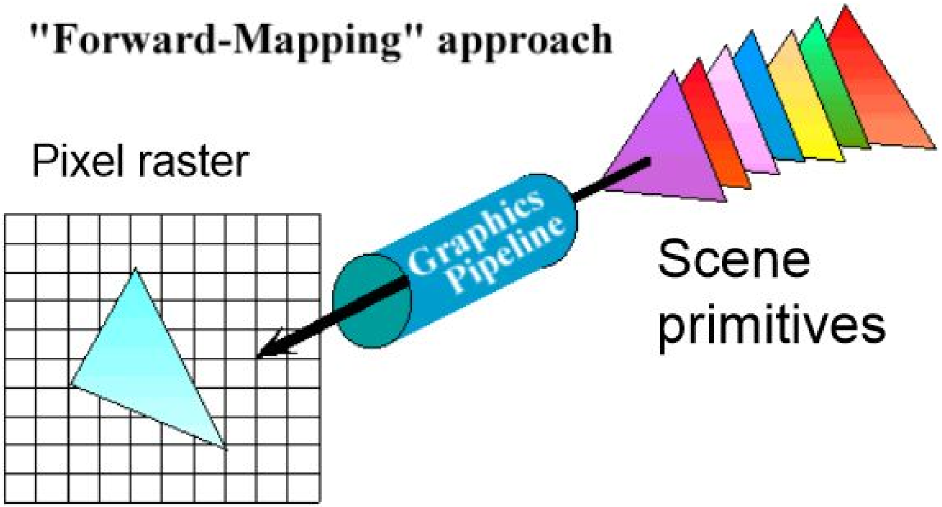
- For each triangle
  - Project triangle to image plane
  - For each pixel
    - Check pixel in triangle
    - Resolve visibility with z-buffer

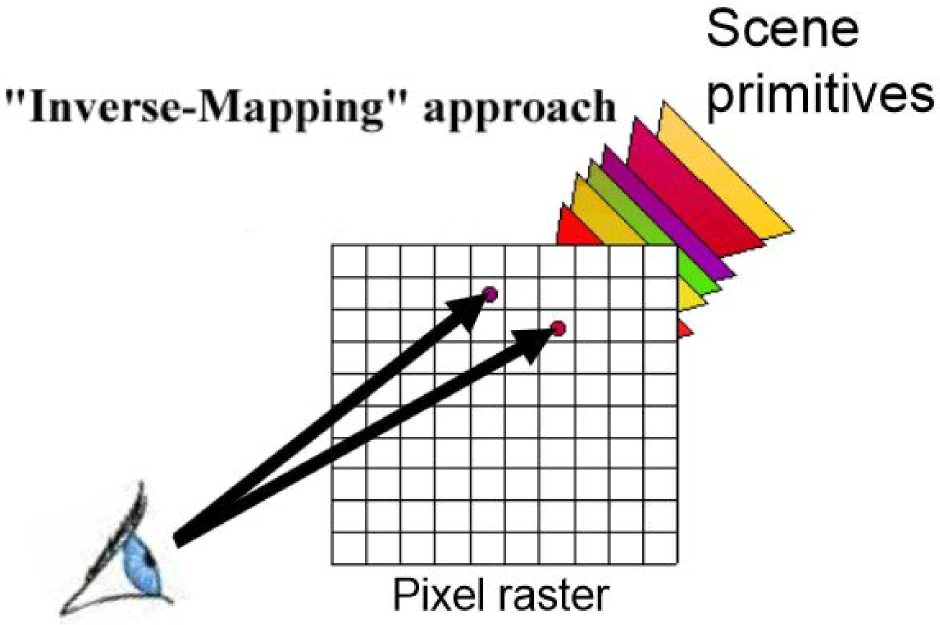


Rasterization VS ray-casting

- For each triangle
  - Project triangle to image plane
  - For each pixel
    - Check pixel in triangle
    - Resolve visibility with z-buffer

- For each pixel
  - Compute pixel ray
  - For each triangle
    - Check ray-triangle intersection
    - Get closest intersection





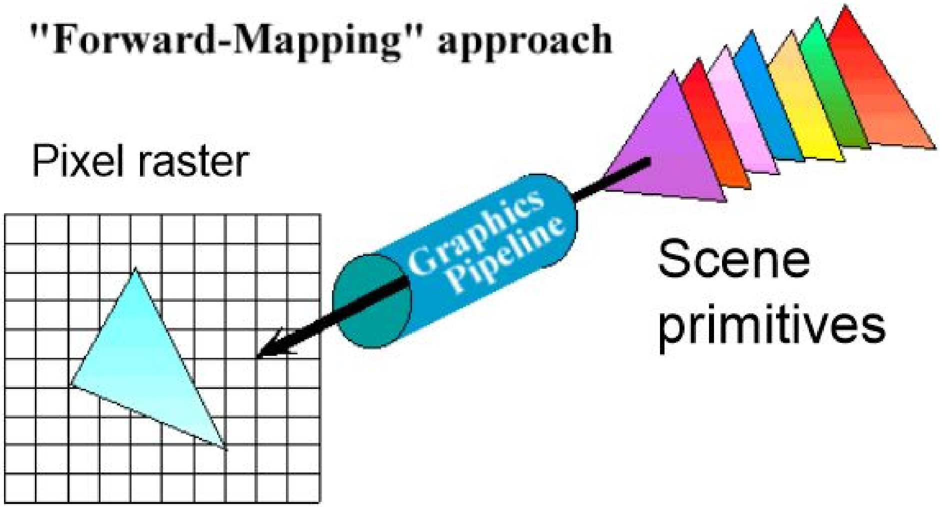
Rasterization VS ray-casting

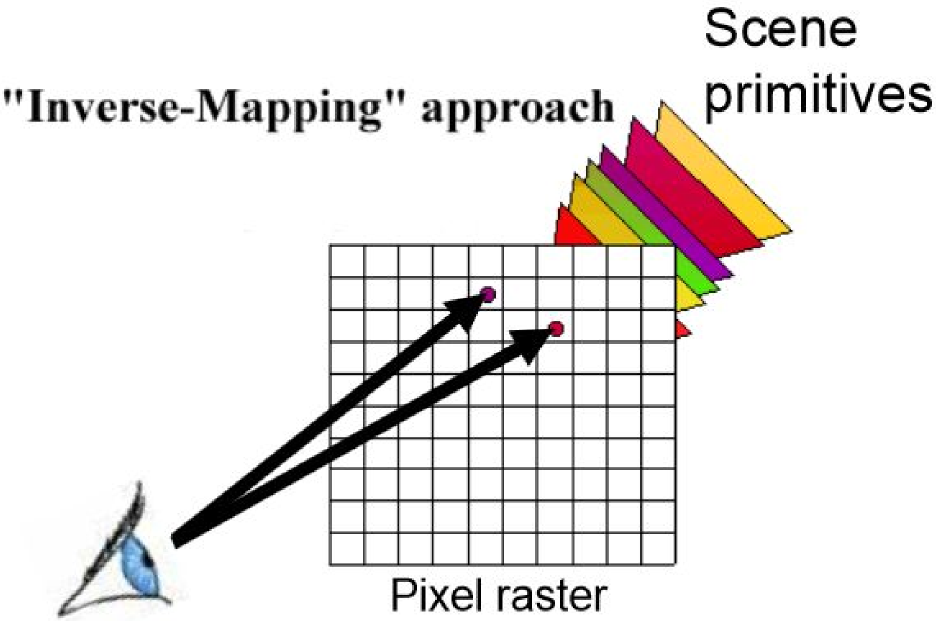
- For each triangle
  - Project triangle to image plane
  - For each pixel
    - Check pixel in triangle
    - Resolve visibility with z-buffer

Triangle-centric

- For each pixel
  - Compute pixel ray
  - For each triangle
    - Check ray-triangle intersection
    - Get closest intersection

Ray-centric





Images from: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/MIT6_837F12_Lec21.pdf

Ray-casting advantages

- Generality
  - Not limited to triangles: can render anything
  - Polygons, implicit, b-rep, etc…
- Shadows, reflection, refraction
  - Uniform handling
  - Directly obtained via recursion
- Base for many advanced algorithms
  - Path tracing, photon mapping, etc…

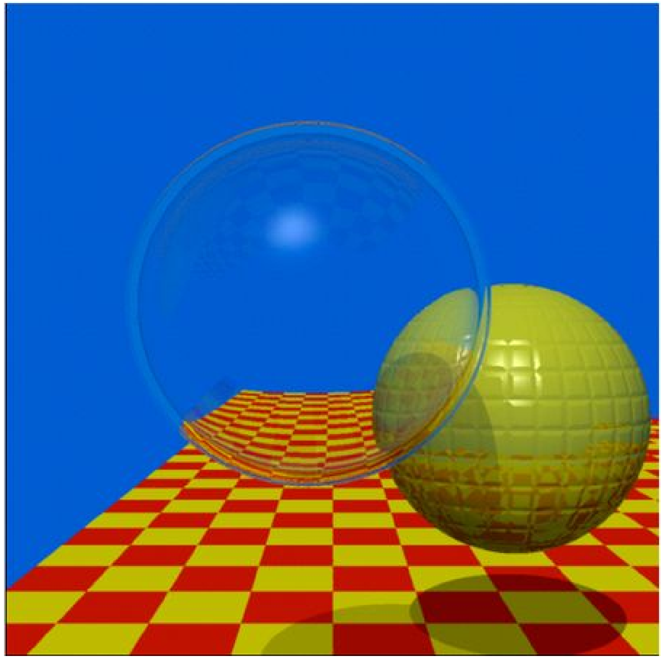
Ray-casting limitations

- Can be hard to implement
  - Entire scene in memory

- Can be slow with large scenes
  - But…

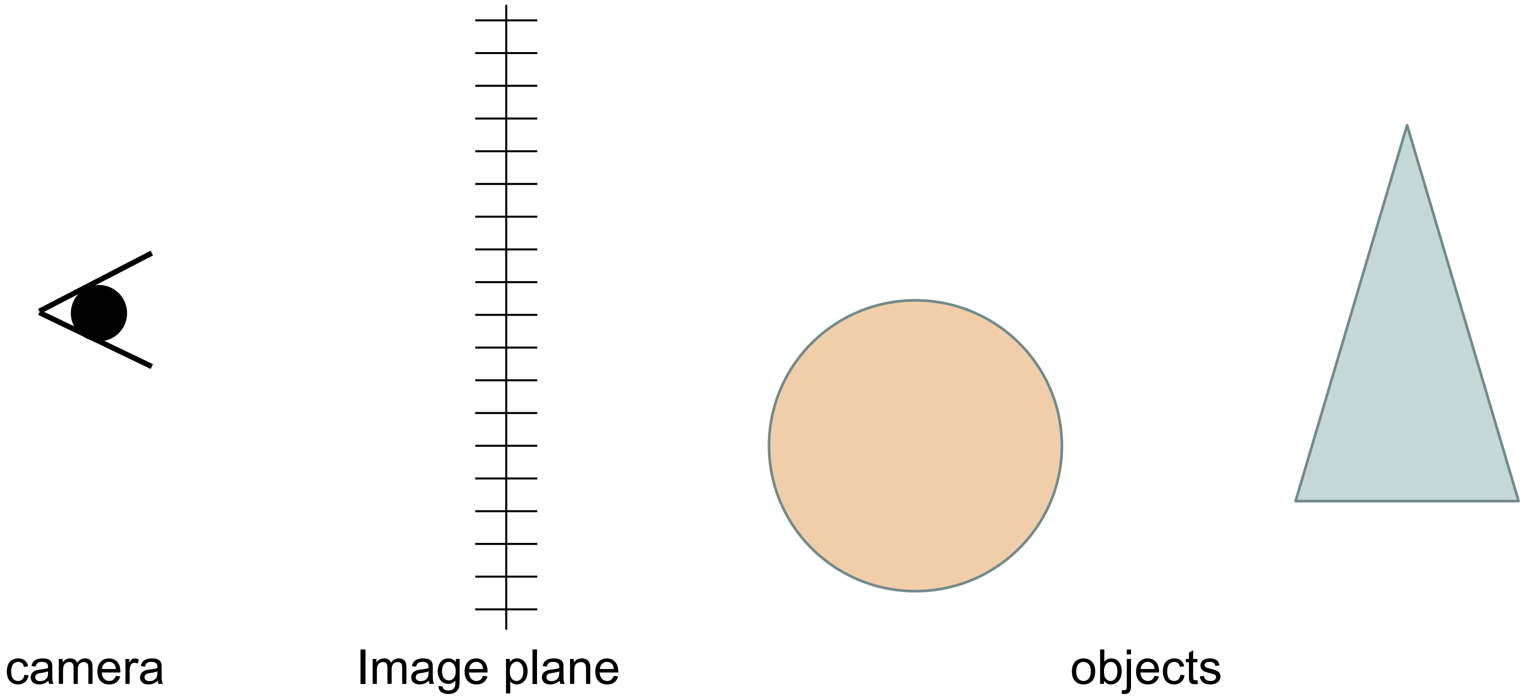
- VAX 11/780 (1979): 74 min
- PC (2006): 6 sec
- GPU (2009): 30 fps
- GPU (2014): > 60 fps

[T. Whitted, 1980]



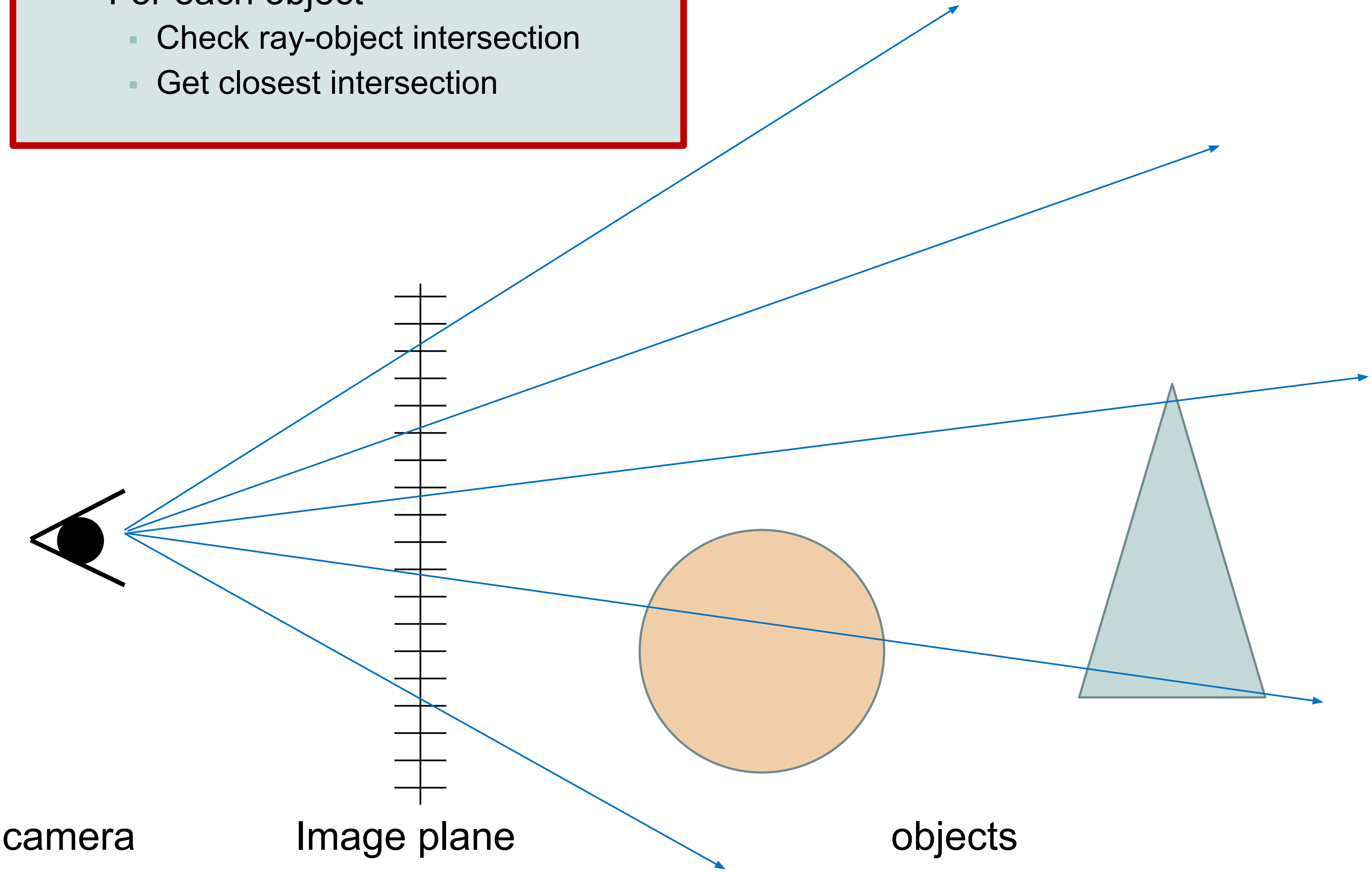
Ray-casting basics

- For each pixel
  - Compute pixel ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection



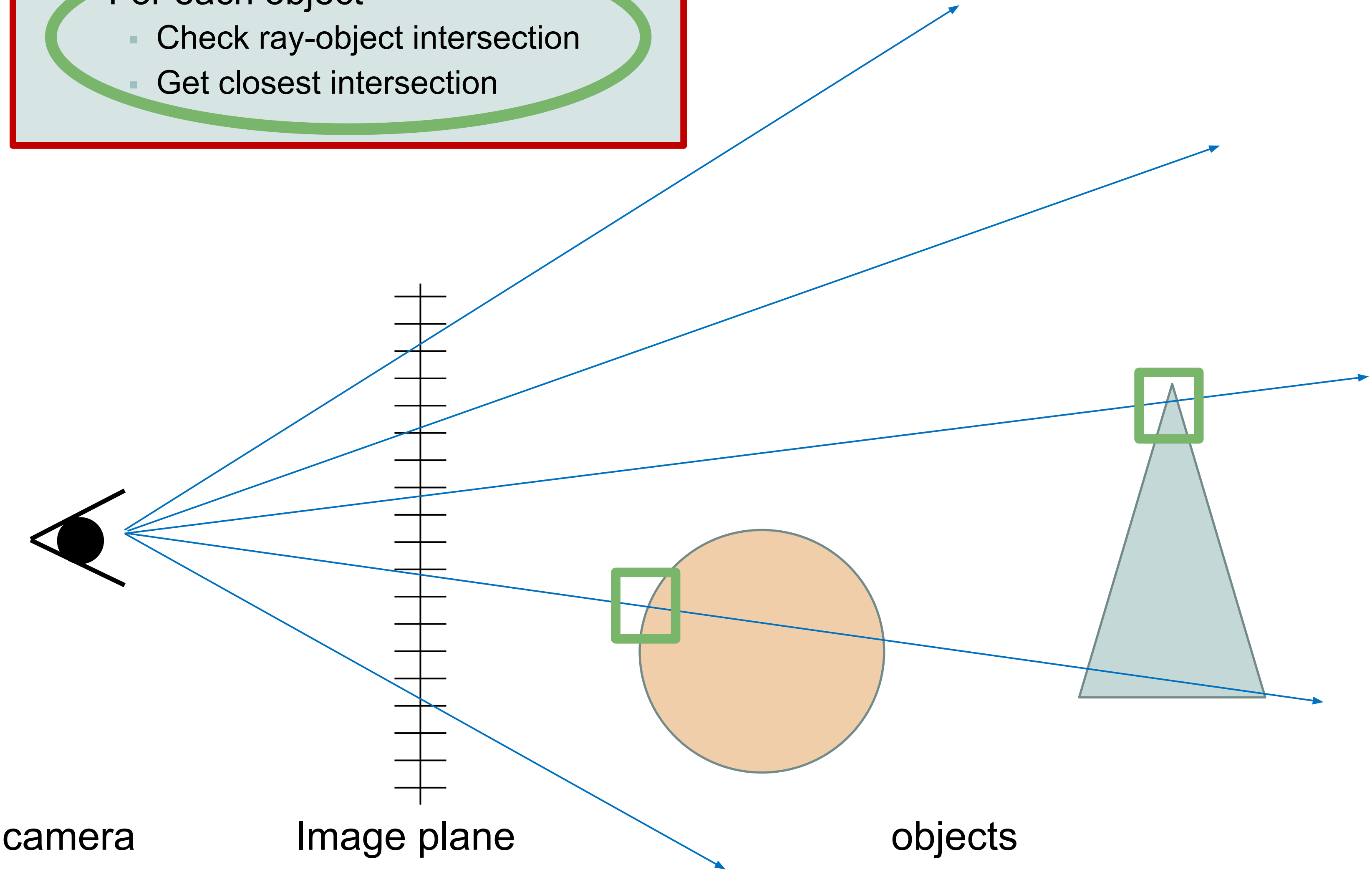
Ray-casting basics

- For each pixel
  - Compute pixel ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection



Ray-casting basics

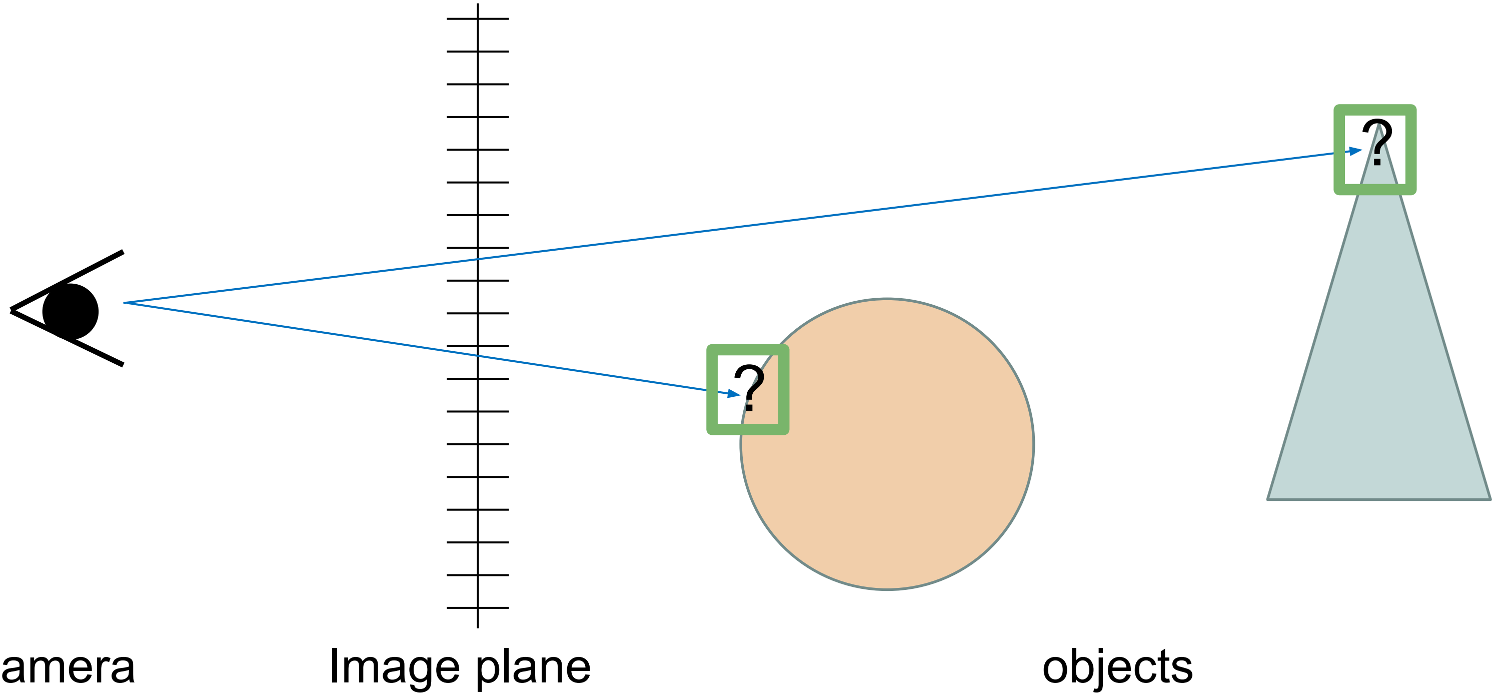
- For each pixel
  - Compute pixel ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection



Ray-casting basics

```
For each pixel
    Compute pixel ray
    For each object
        Check ray-object intersection
        Get closest intersection
```

And then?



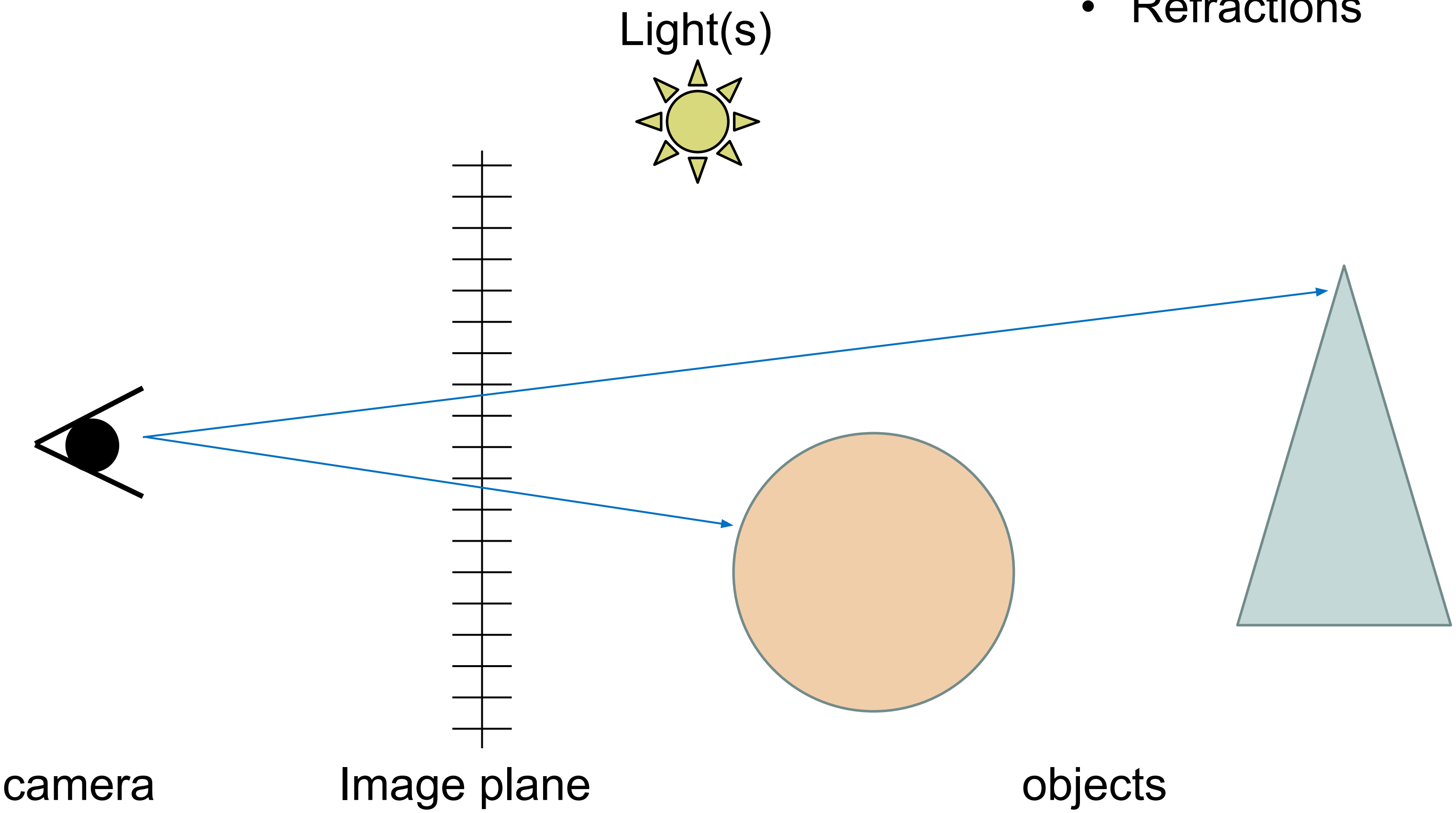
Ray-casting basics

- For each pixel
  - Compute pixel ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection

And then?

Shade!

- Shadow rays
- Reflections
- Refractions



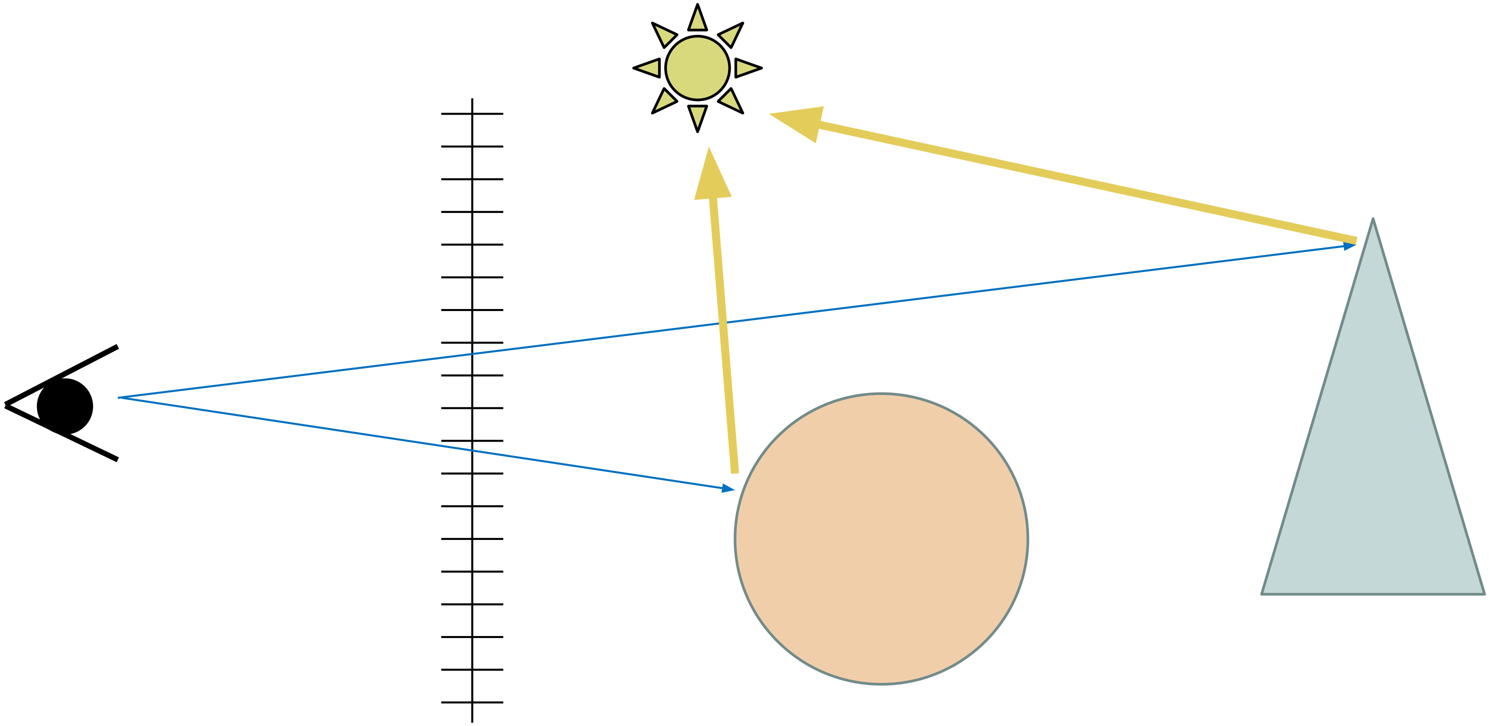
Ray-casting basics

- For each pixel
    - Compute pixel ray
    - For each object
        - Check ray-object intersection
        - Get closest intersection

And then?

Shade!

- Shadow rays
- Reflections
- Refractions



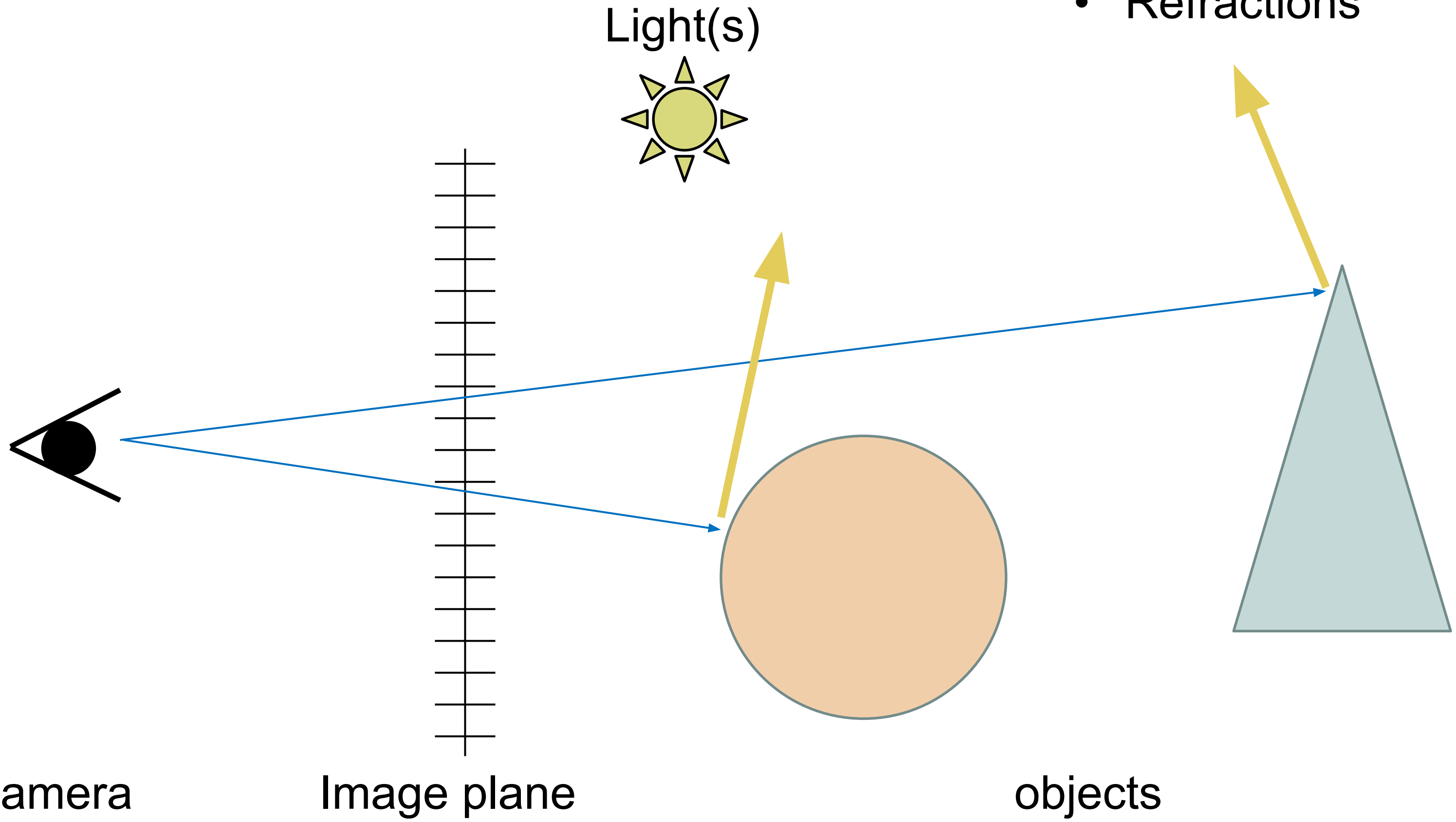
Ray-casting basics

- For each pixel
  - Compute pixel ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection

And then?

Shade!

- Shadow rays
- Reflections
- Refractions



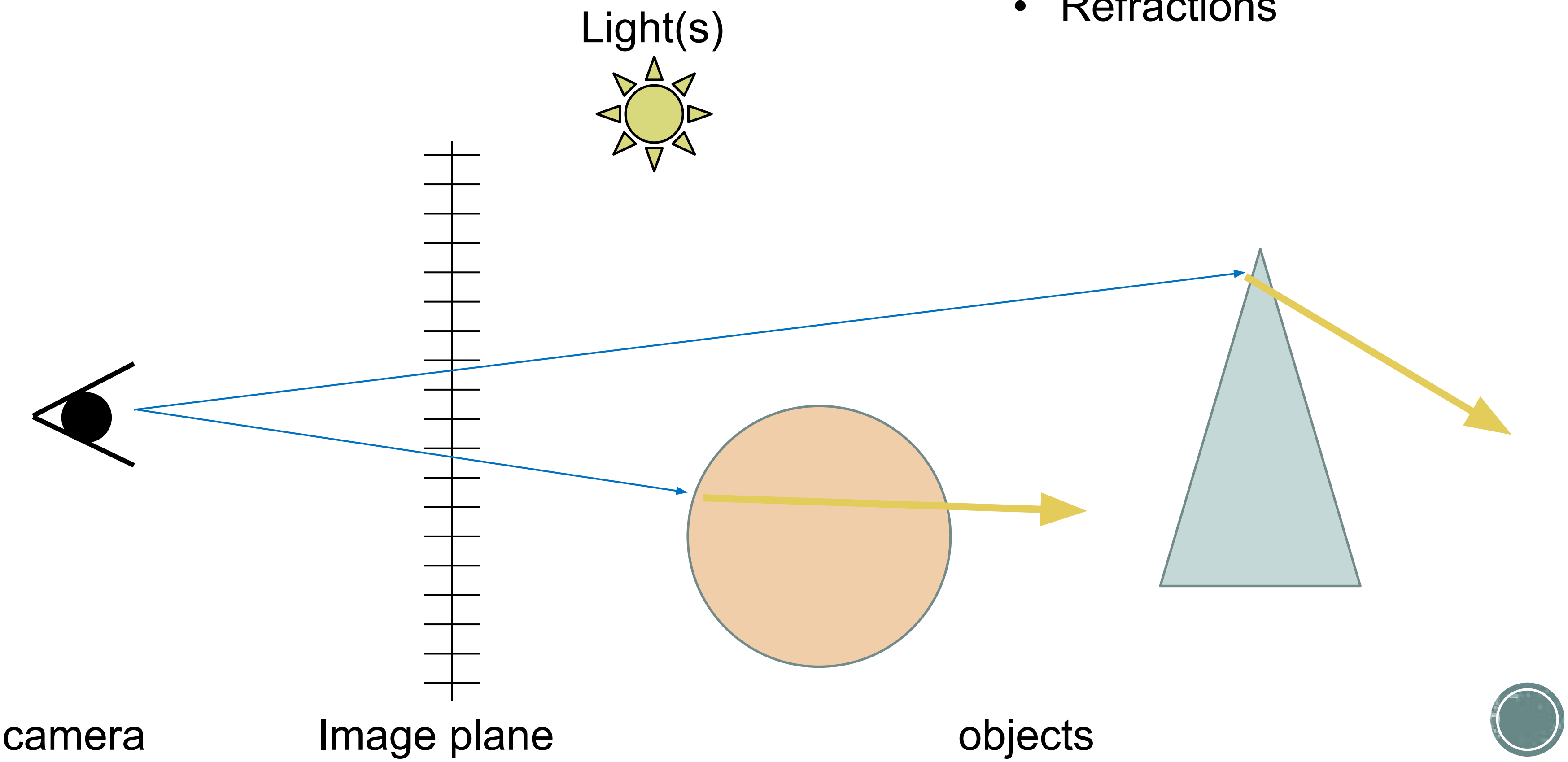
Ray-casting basics

- For each pixel
  - Compute pixel ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection

And then?

Shade!

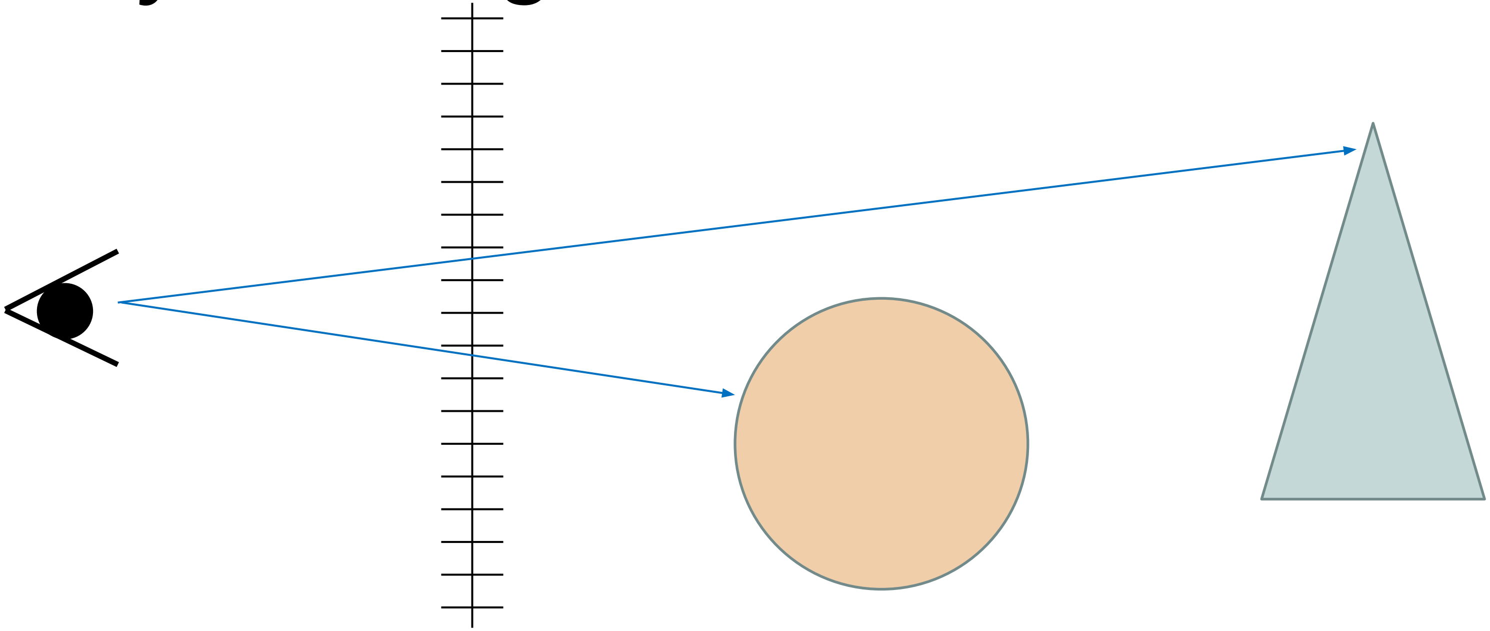
- Shadow rays
- Reflections
- Refractions



Ray-casting vs ray-tracing

Eye rays only
= Ray casting

- Shadow rays
- Reflections
- Refractions



camera

Image plane

objects

Ray-casting vs ray-tracing

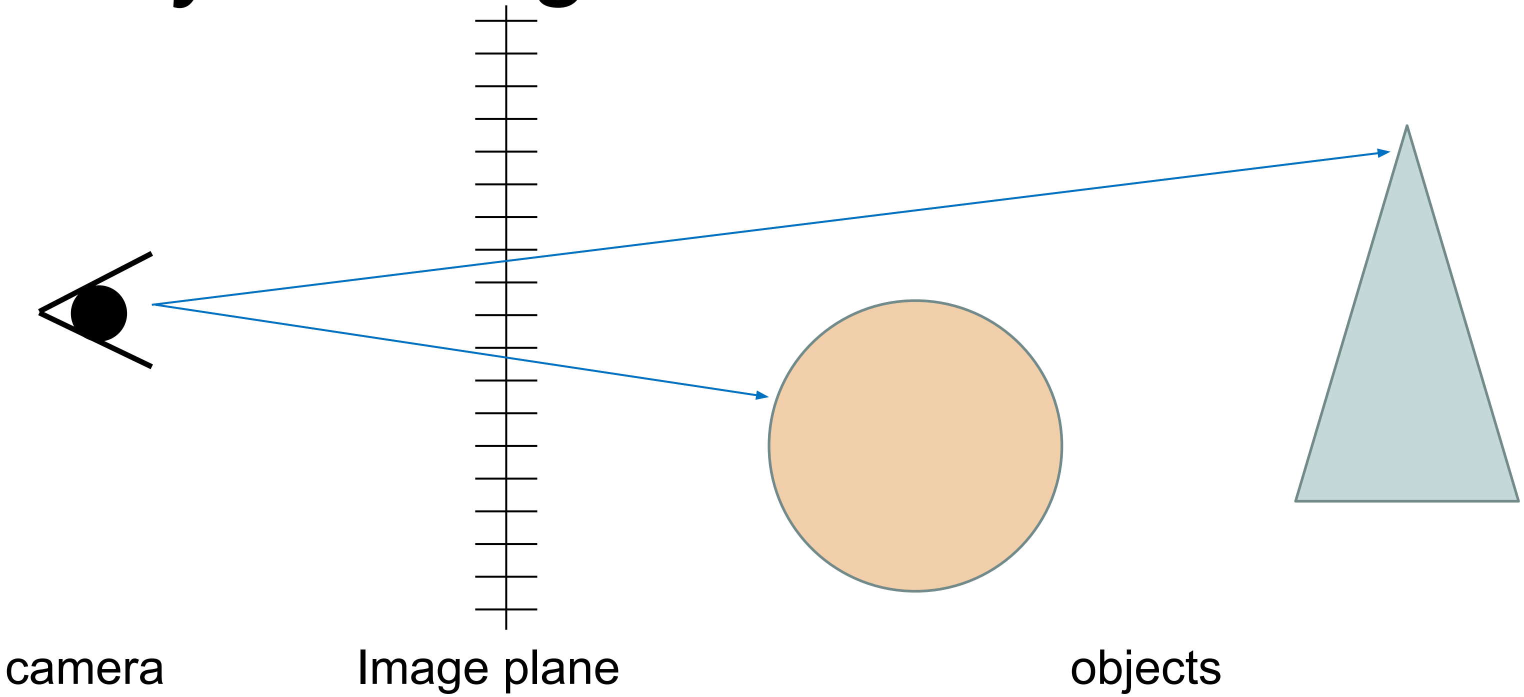
Secondary rays

= Ray tracing

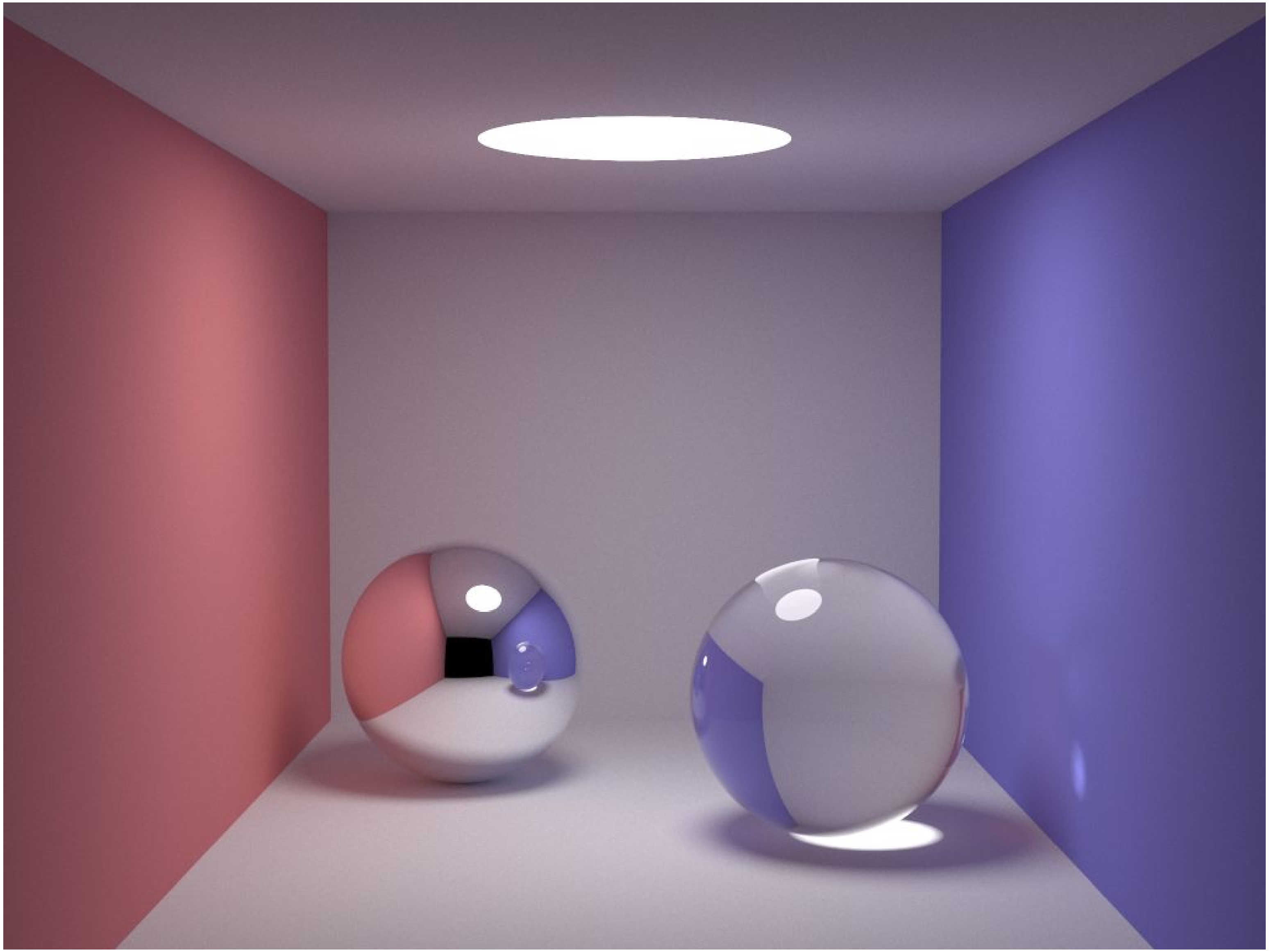
- Shadow rays
- Reflections
- Refractions

Eye rays only

= Ray casting



Ray-casting vs ray-tracing



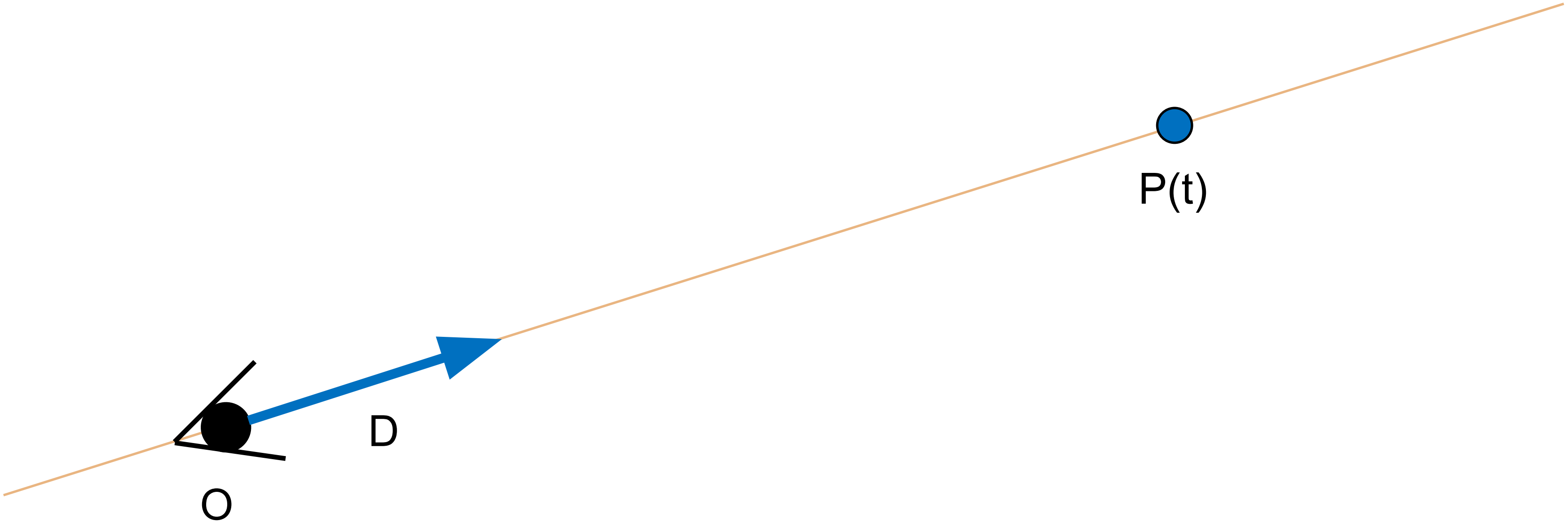
Ray-casting: summary

- For each pixel
  - Compute eye ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection
    - Shade depending on light and normal vector

Finding intersection point and normal is the central part of ray-casting!

Eye ray and camera

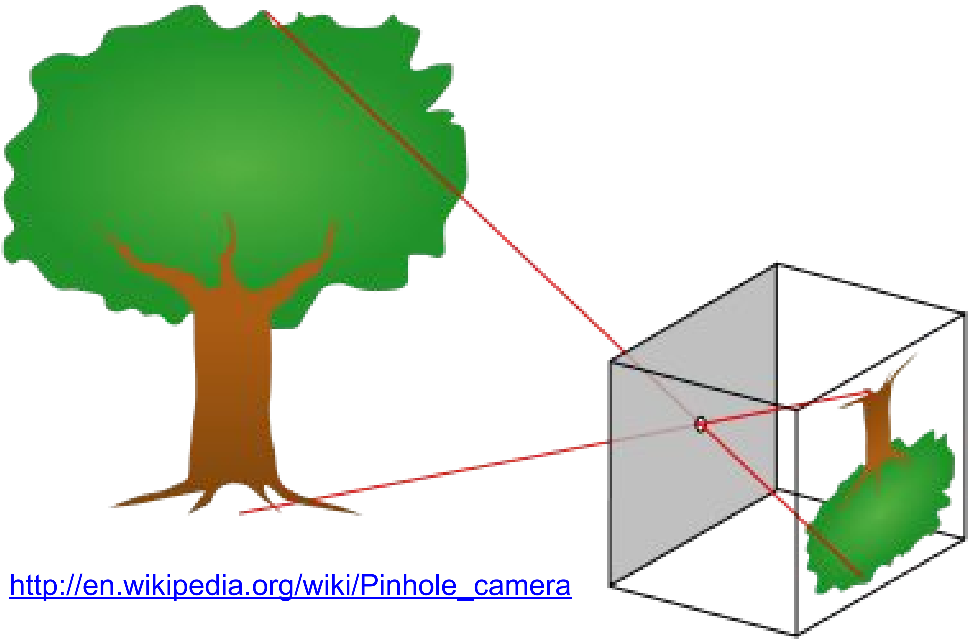
- Ray representation: parametric line
  - Origin O (3D point)
  - Direction D (normalized vector)
  - P(t) = O + t*D



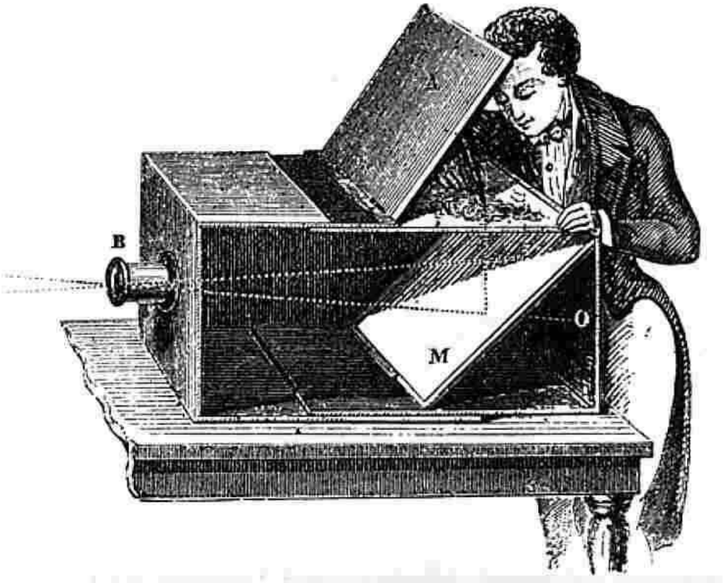
Goal: find smallest t>0 such that P(t) lies on a surface

Eye ray and camera

- Pinhole camera (or camera obscura)
  - Small aperture (perfect image if pinhole infinitely small)
  - Inverted image
  - Pure geometric optics



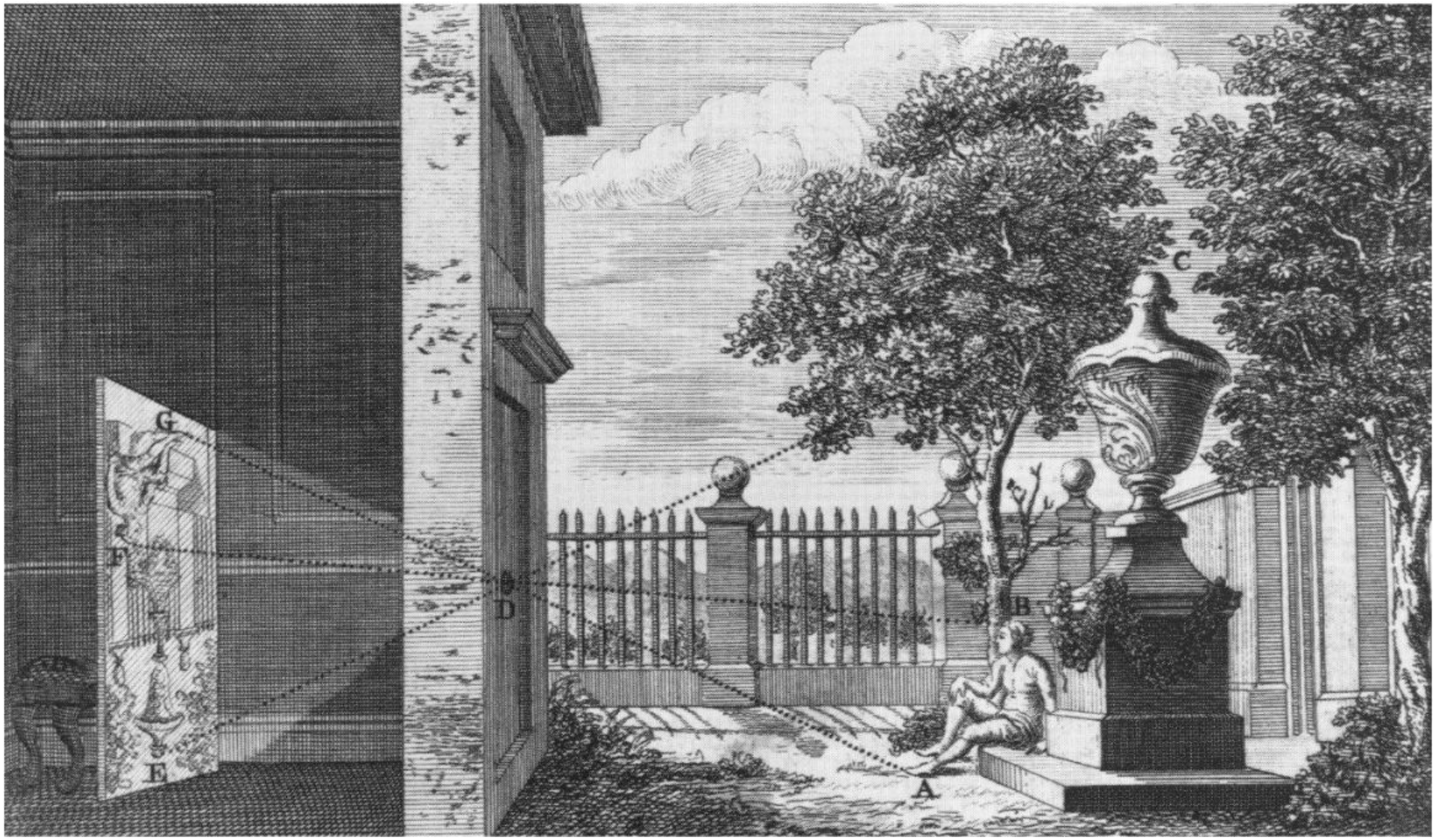
http://en.wikipedia.org/wiki/Pinhole_camera



A 19th-century artist using a camera obscura to outline his subject

Eye ray and camera

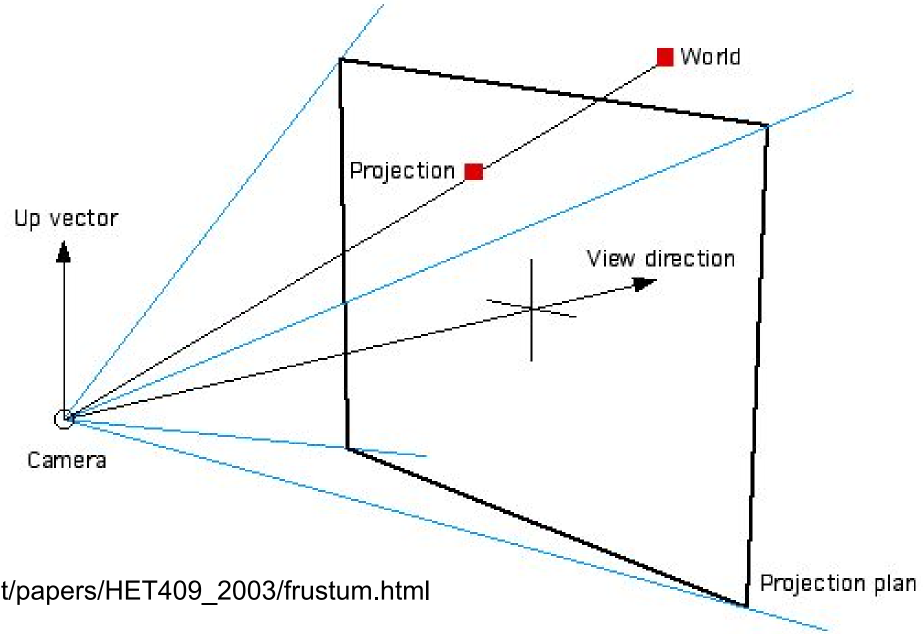
- Pinhole camera (or camera obscura)
  - Small aperture (perfect image if pinhole infinitely small)
  - Inverted image
  - Pure geometric optics



Wellcome Library, London. License: CCBY-NC

Eye ray and camera

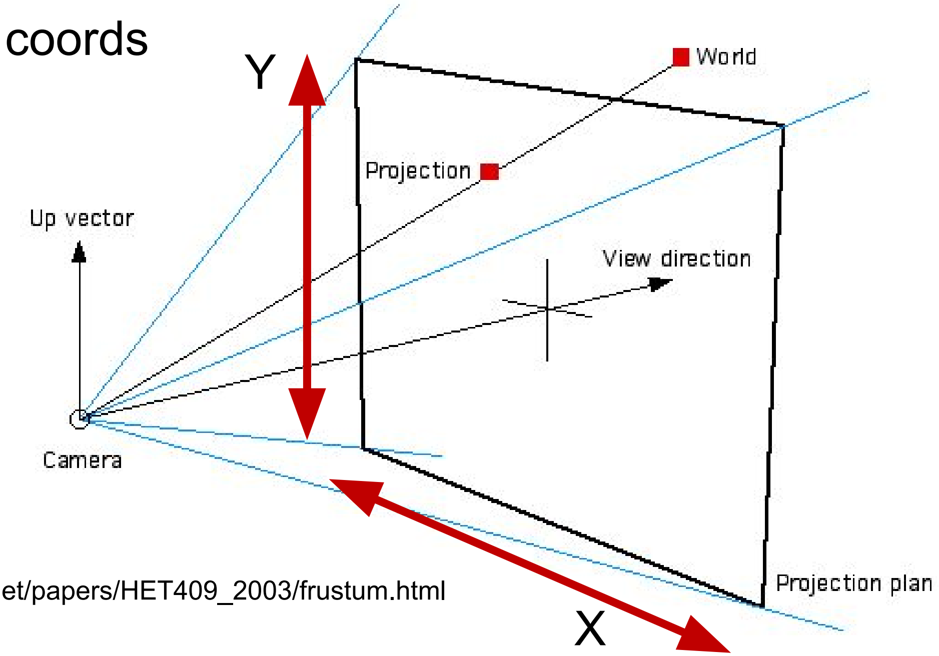
- Simplified Pinhole camera
  - Eye position: e
  - Orthogonal basis: u,v,w (right, up, view) directions
  - Field of view: alpha
  - Aspect ratio: w/h



http://paulbourke.net/papers/HET409_2003/frustum.html

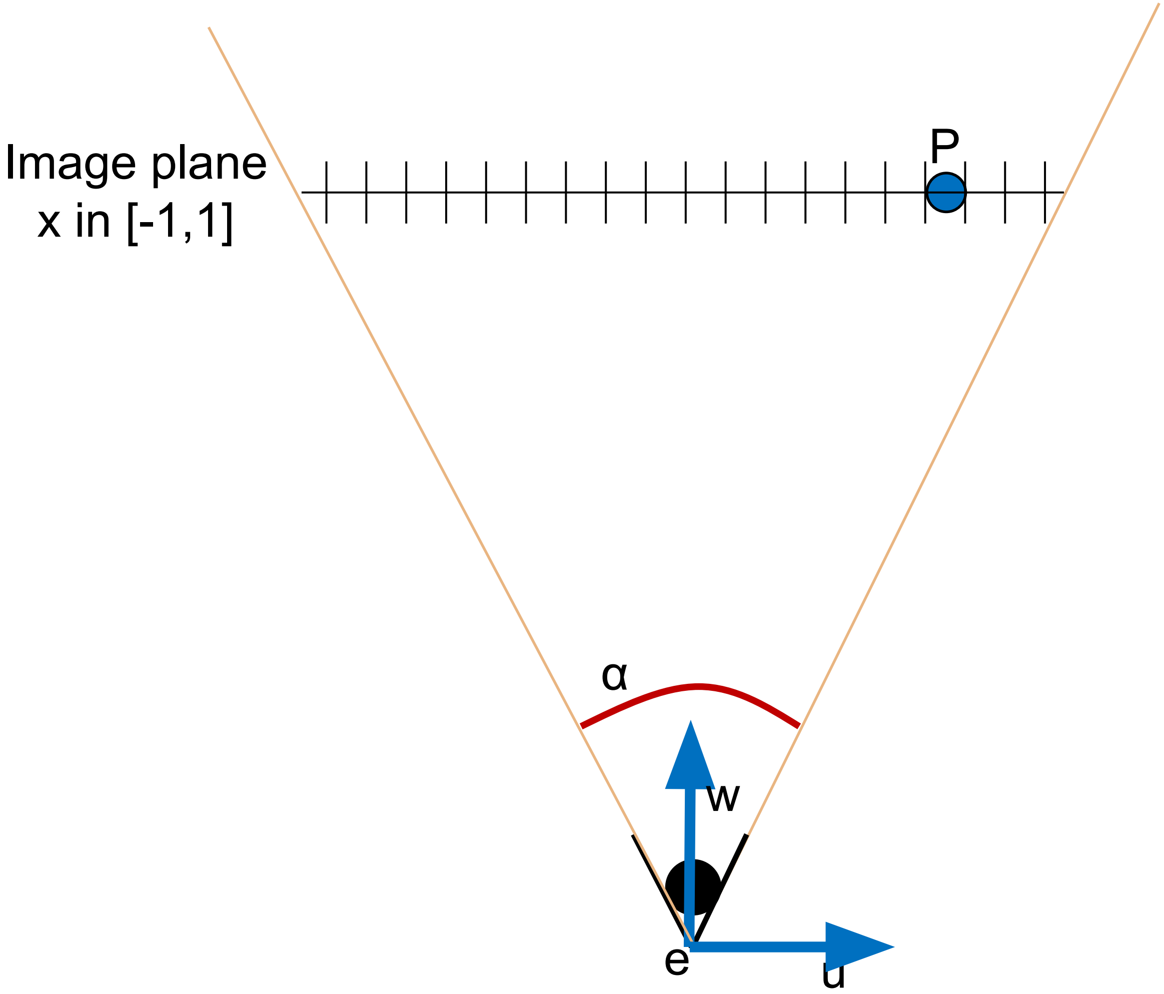
Eye ray and camera

- Simplified Pinhole camera
  - Eye position: e
  - Orthogonal basis: u,v,w (right, up, view) directions
  - Field of view: alpha
  - Aspect ratio: w/h
- Image coordinates
  - Normalized image coords
  - X in [-1,1]
  - Y in [-1,1]

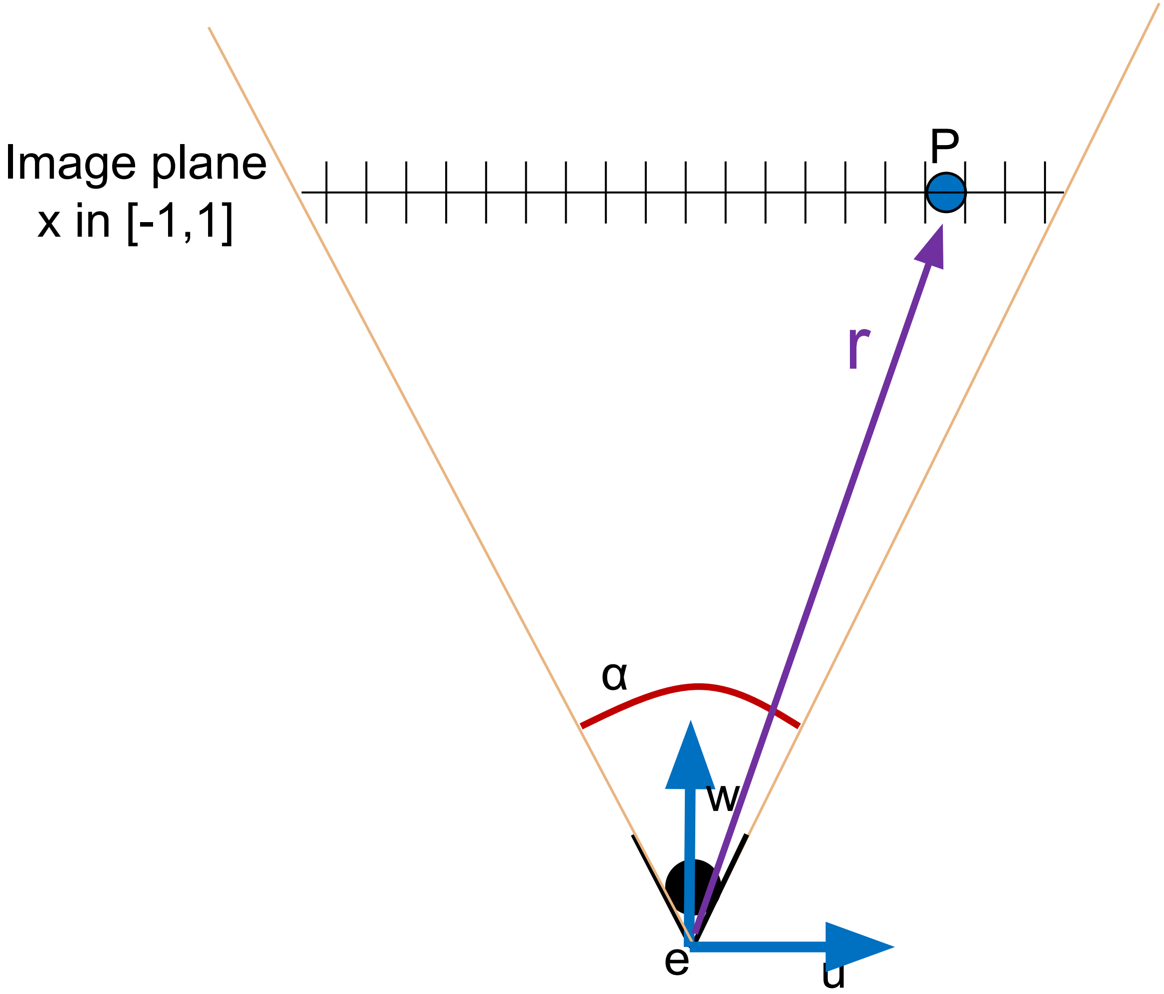


http://paulbourke.net/papers/HET409_2003/frustum.html

Ray generation

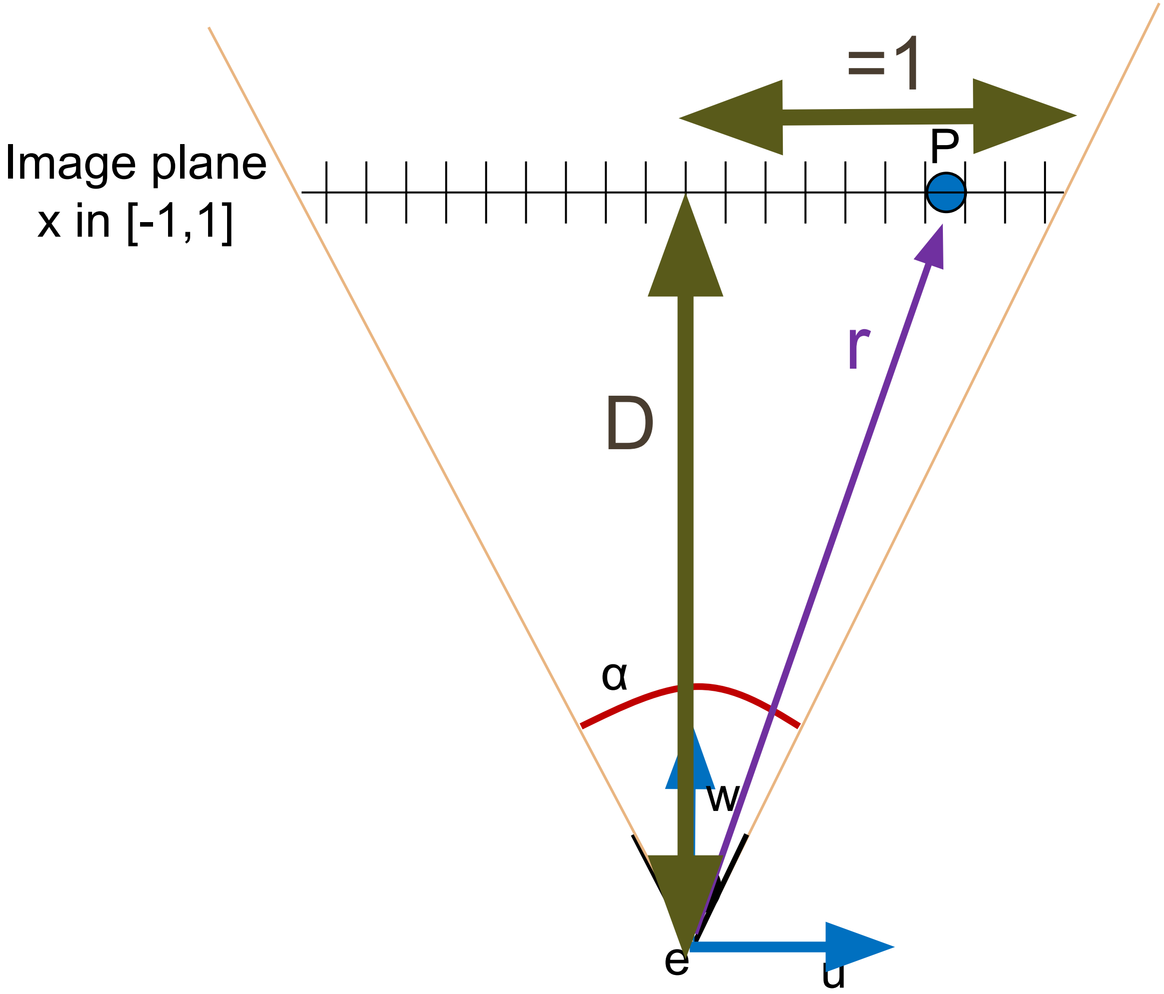


Ray generation



Goal: find r

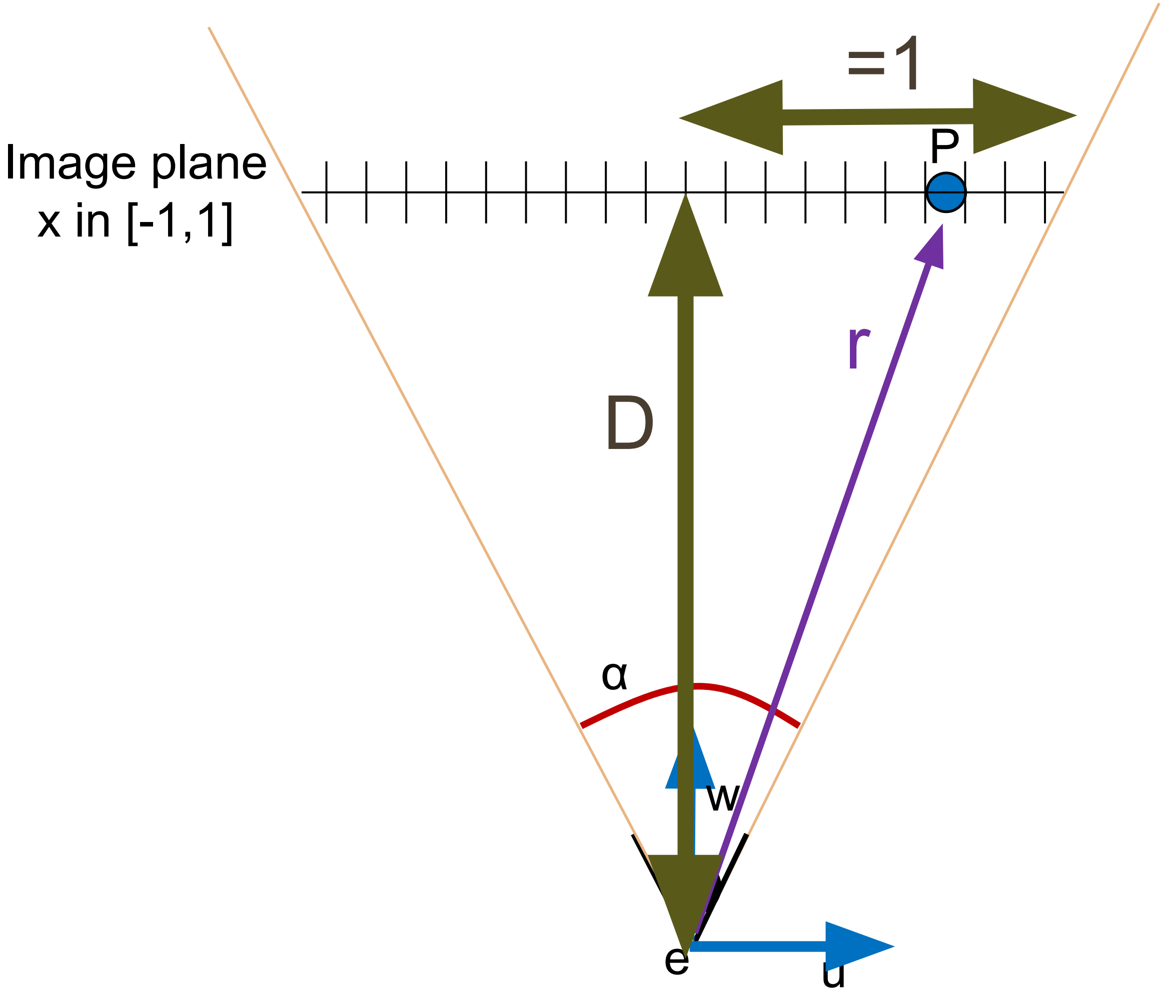
Ray generation



Goal: find r

D = ?

Ray generation

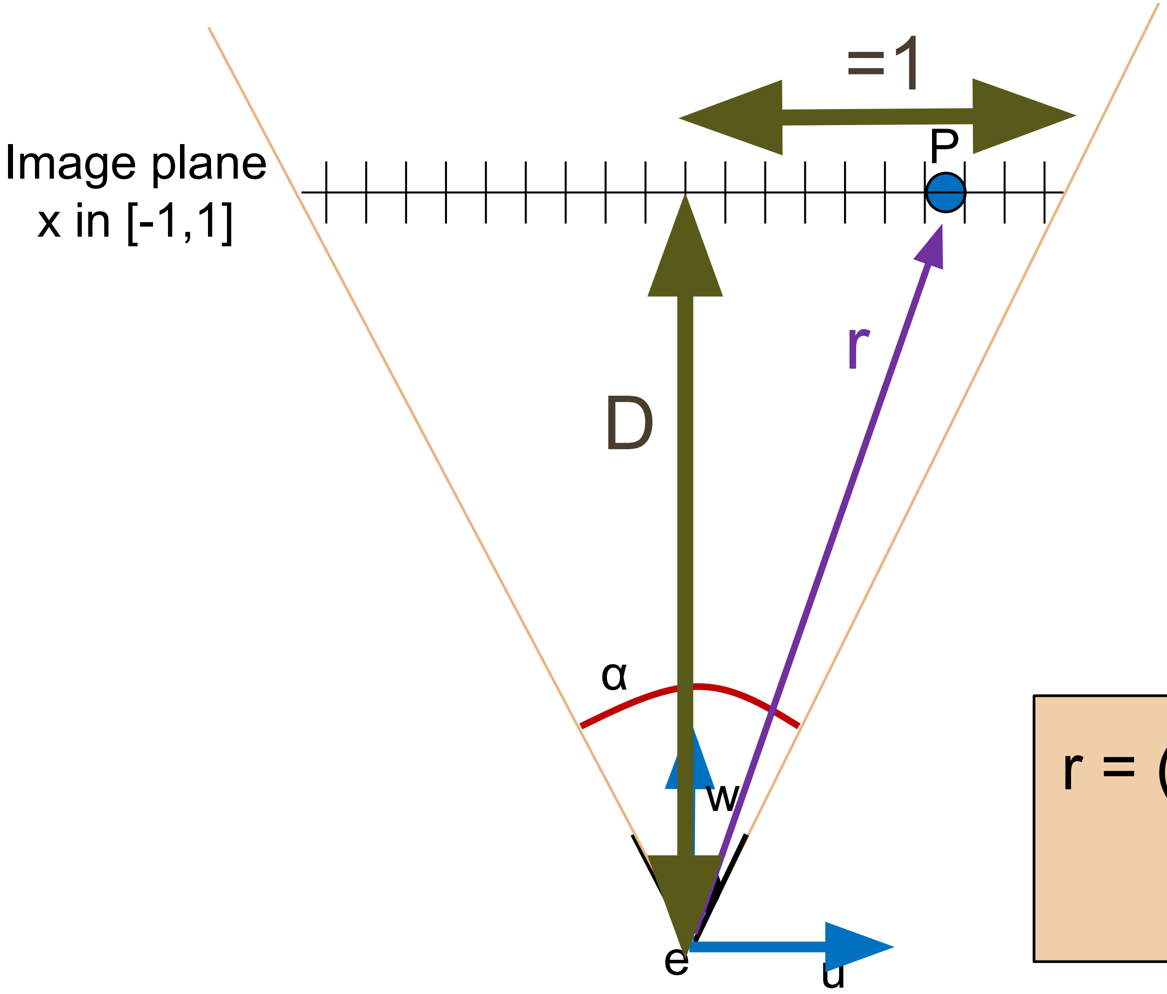


Goal: find r

$$\tan\frac{\alpha}{2} = \frac{1}{D}$$

$$D = \frac{1}{\tan(\alpha/2)}$$

Ray generation



Goal: find r

$$\tan\frac{\alpha}{2} = \frac{1}{D}$$

$$D = \frac{1}{\tan(\alpha/2)}$$

r = (x*u,D*w), normalized

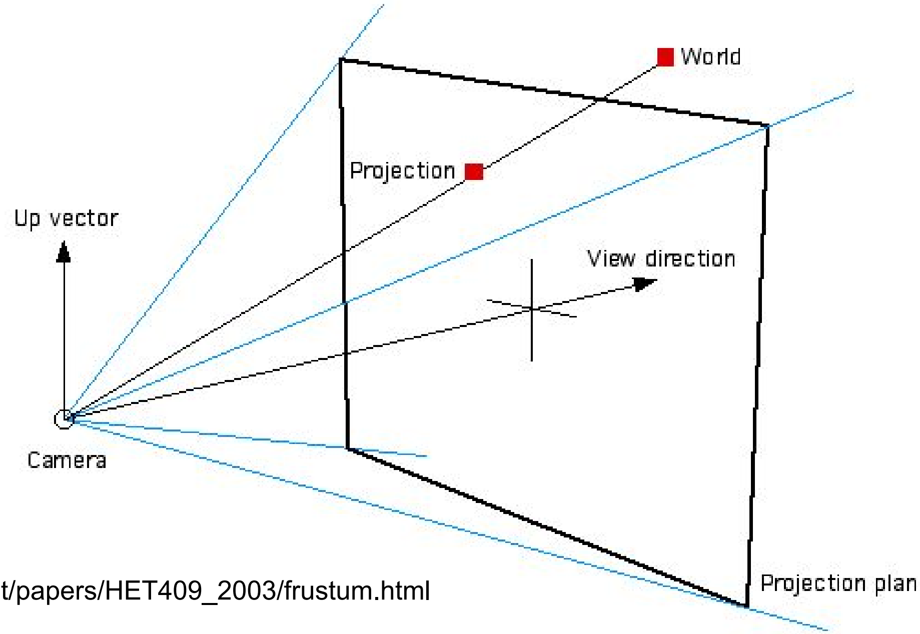
P(t) = e + t*r

Eye ray and camera

- In 3D

r = (x*u,aspect*y*v,D*w), normalized

P(t) = e + t*r



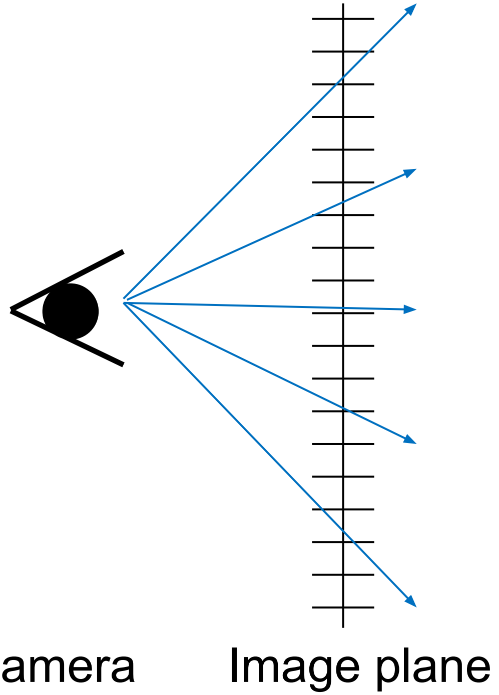
http://paulbourke.net/papers/HET409_2003/frustum.html

Eye ray and camera

- Persective

r = (x*u,aspect*y*v,D*w), normalized

P(t) = e + t*r

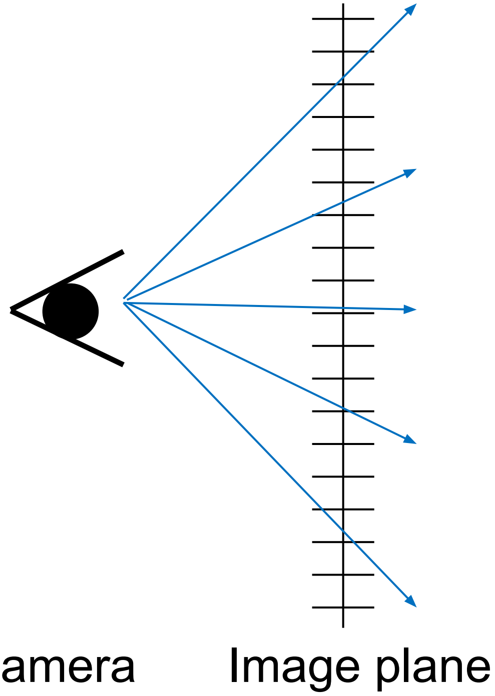


Eye ray and camera

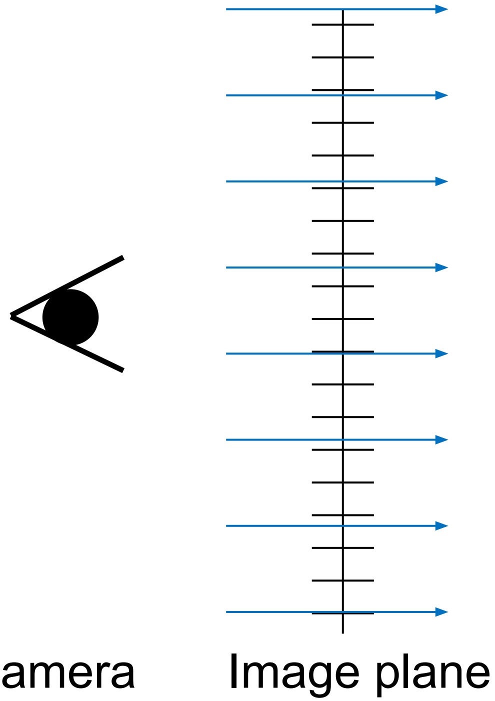
- Persective

r = (x*u,aspect*y*v,D*w), normalized

P(t) = e + t*r



- Orthographic



Eye ray and camera

- Persective

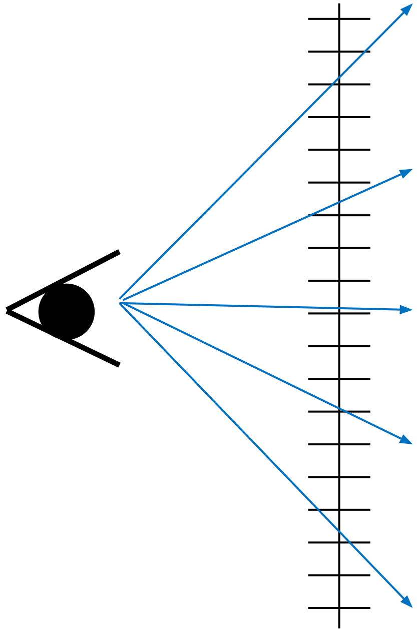
r = (x*u,aspect*y*v,D*w), normalized

P(t) = e + t*r

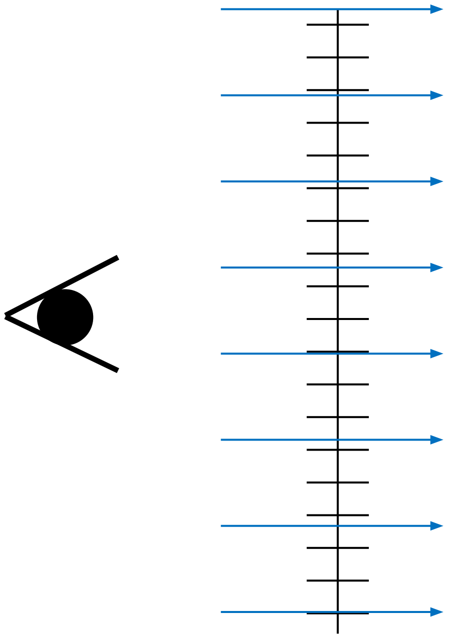
- Orthographic

P(t) = o + t*w

o = e + x*size*u + y*size*v



camera Image plane



camera Image plane

Ray-casting: summary

- For each pixel
  - Compute eye ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection
    - Shade depending on light and normal vector

OK!

Finding intersection point and normal is the central part of ray-casting!

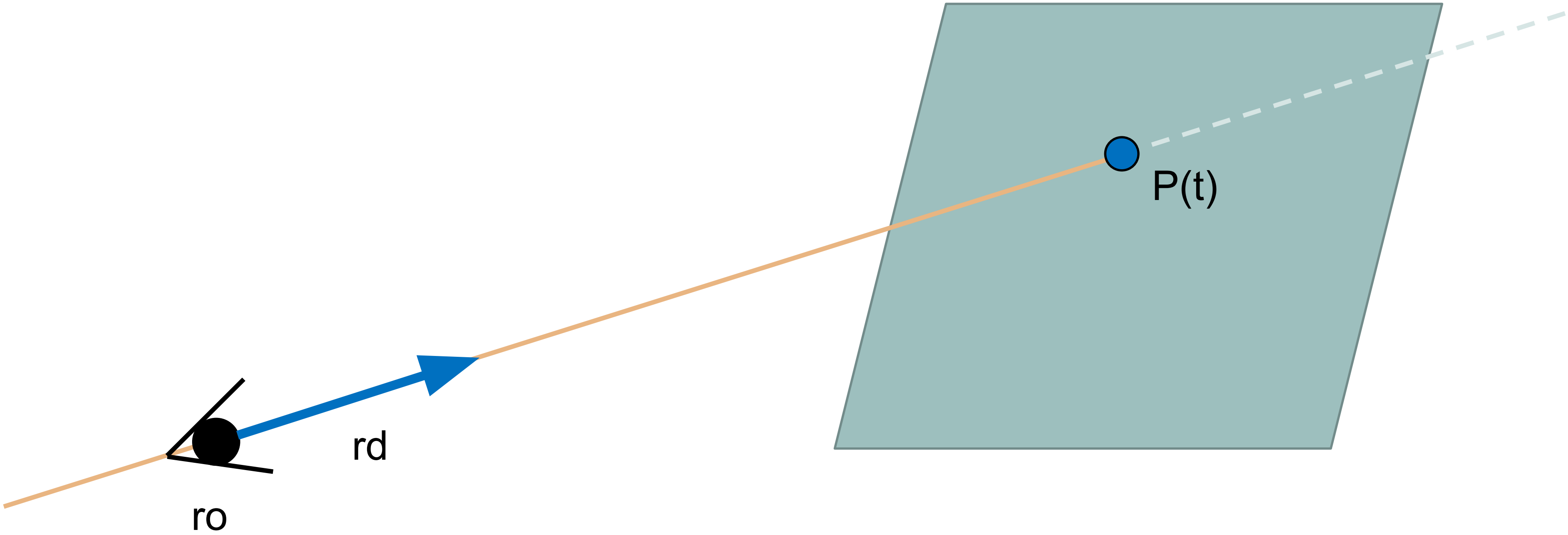
Ray-casting: summary

- For each pixel
  - Compute eye ray
  - For each object
    - Check ray-object intersection ?
    - Get closest intersection
    - Shade depending on light and normal vector

Finding intersection point and normal is the central part of ray-casting!

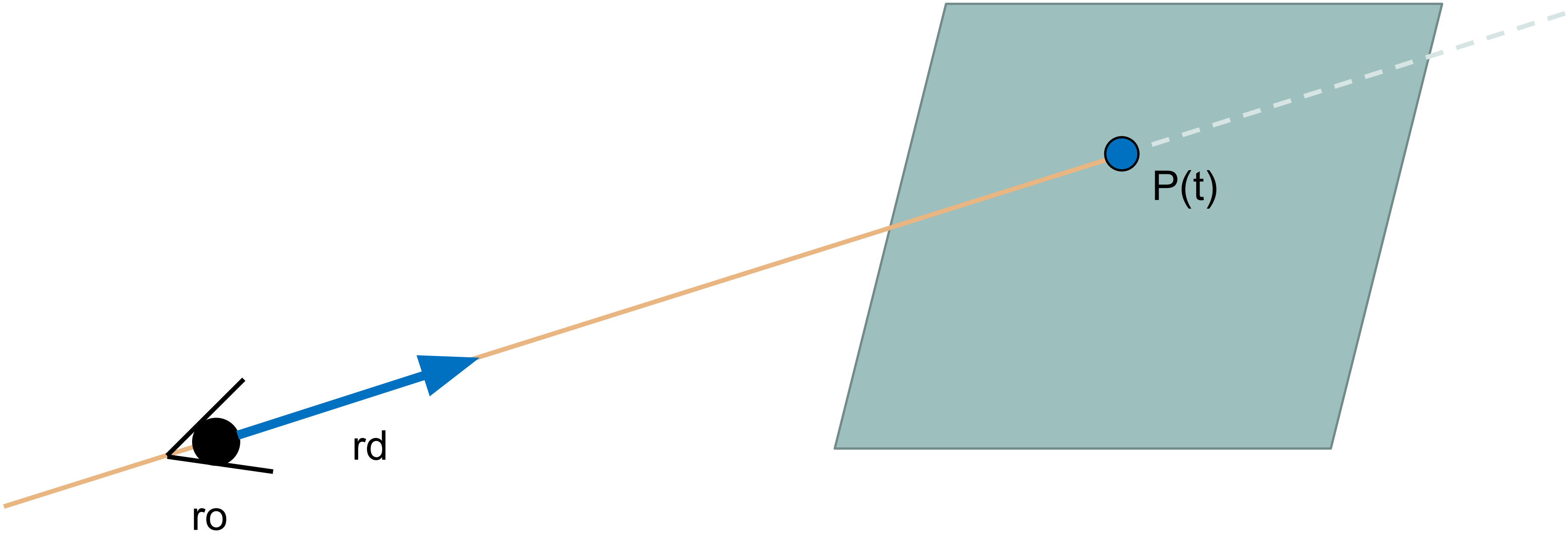
Ray-plane intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$



Ray-plane intersection

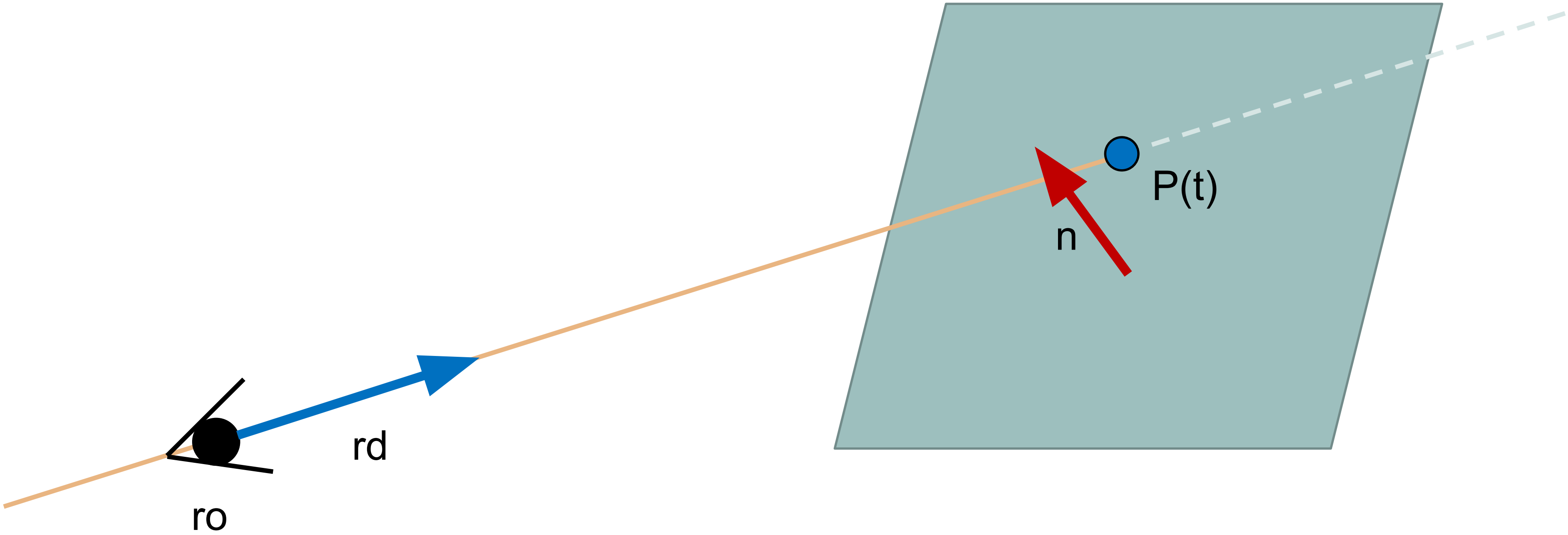
- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit plane equation: $Ax + By + Cz + D = 0$



Ray-plane intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit plane equation: $Ax + By + Cz + D = 0$

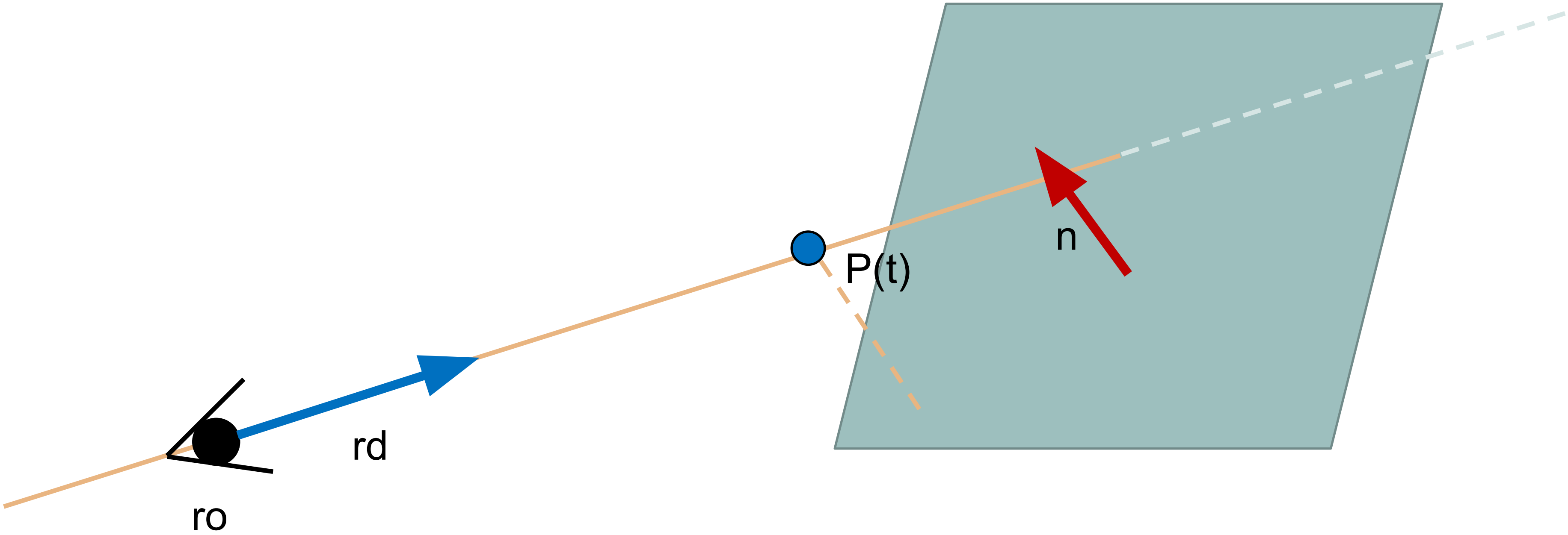
$$n \cdot P + D = 0$$



Ray-plane intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit plane equation: $Ax + By + Cz + D = 0$

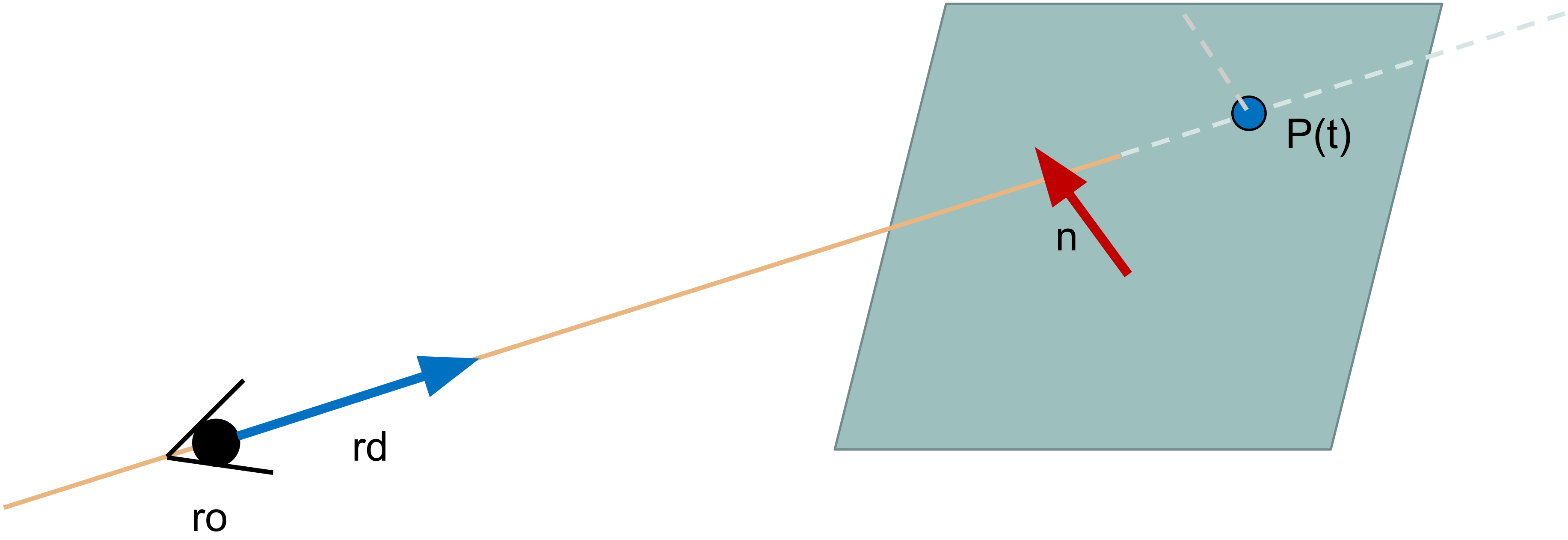
$$n \cdot P + D > 0$$



Ray-plane intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit plane equation: $Ax + By + Cz + D = 0$

$$n \cdot P + D < 0$$



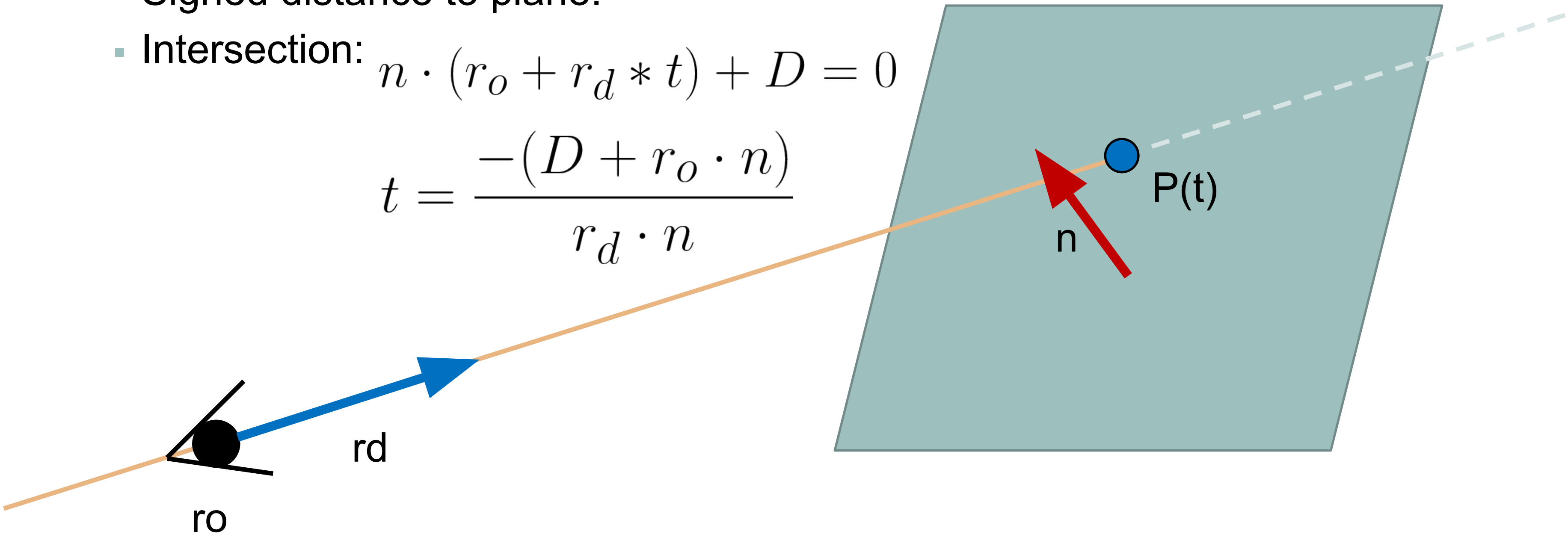
Ray-plane intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit plane equation: $Ax + By + Cz + D = 0$

$$n \cdot P + D = 0$$

  - Signed distance to plane!
  - Intersection: $n \cdot (r_o + r_d * t) + D = 0$

$$t = \frac{-(D + r_o \cdot n)}{r_d \cdot n}$$



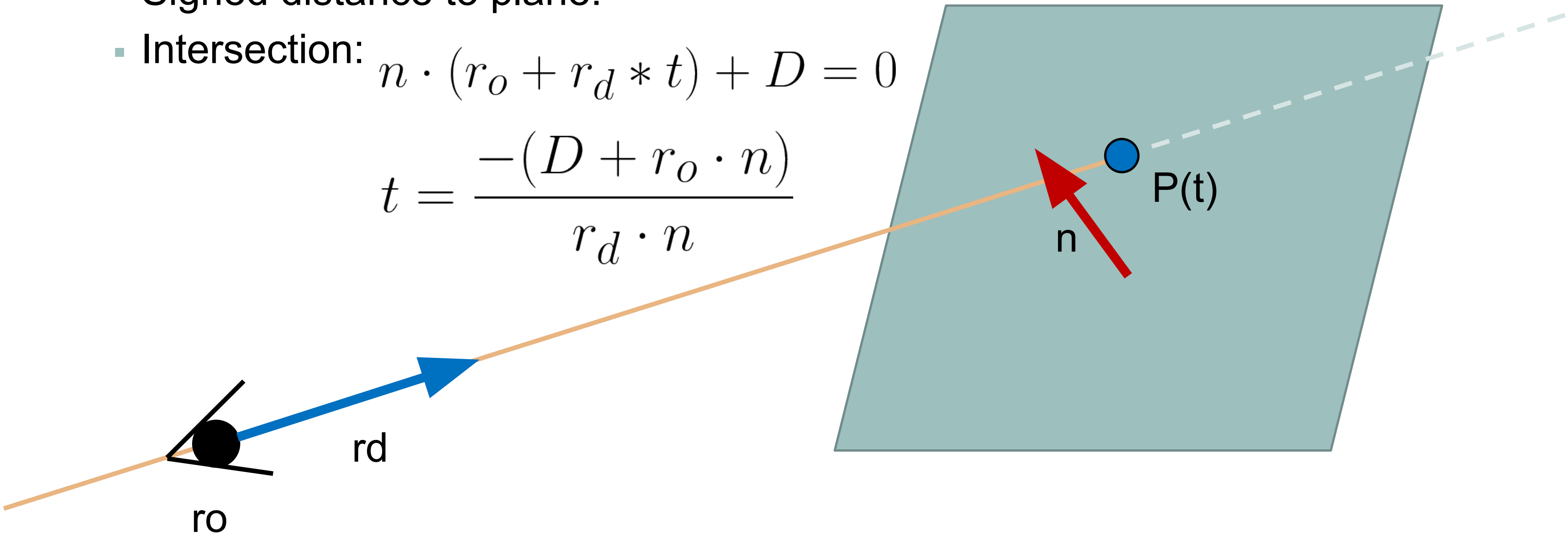
Ray-plane intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit plane equation: $Ax + By + Cz + D = 0$

$$n \cdot P + D = 0$$

  - Signed distance to plane!
  - Intersection: $n \cdot (r_o + r_d * t) + D = 0$

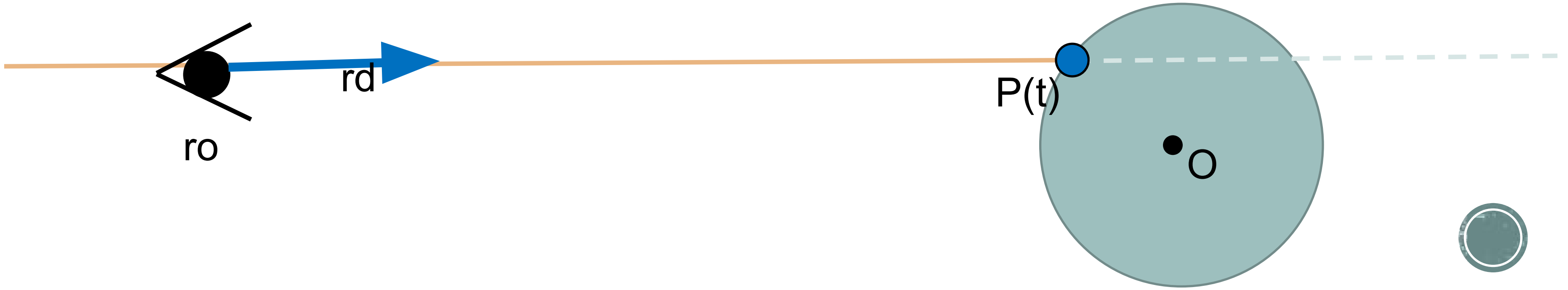
$$t = \frac{-(D + r_o \cdot n)}{r_d \cdot n}$$



  - Normal: constant (n)

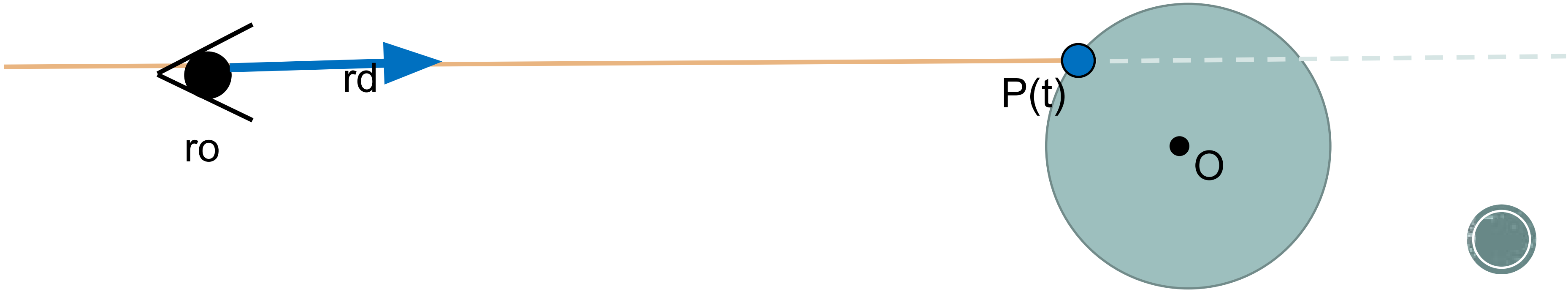
Ray-sphere intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit sphere equation: $||P - O|| - r^2 = 0$



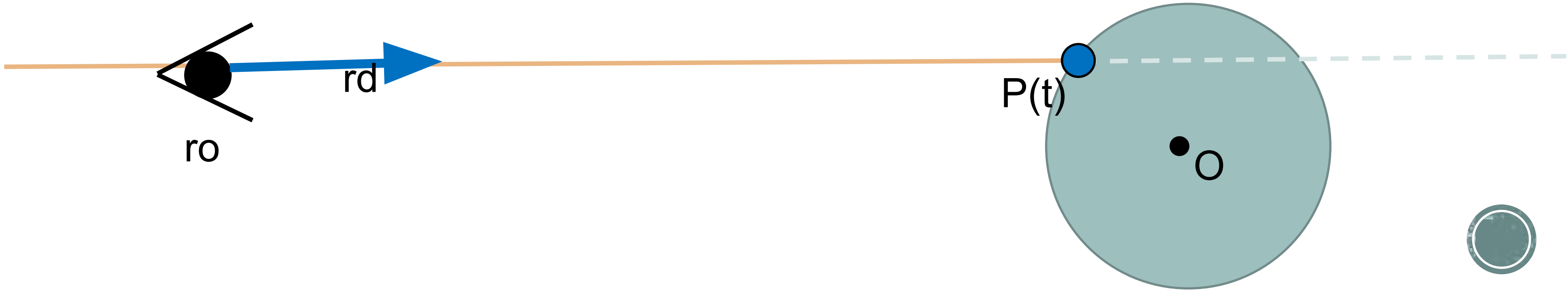
Ray-sphere intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit sphere equation: $||P - O|| - r^2 = 0$
  - $||r_o + r_d * t - O|| - r^2 = 0$



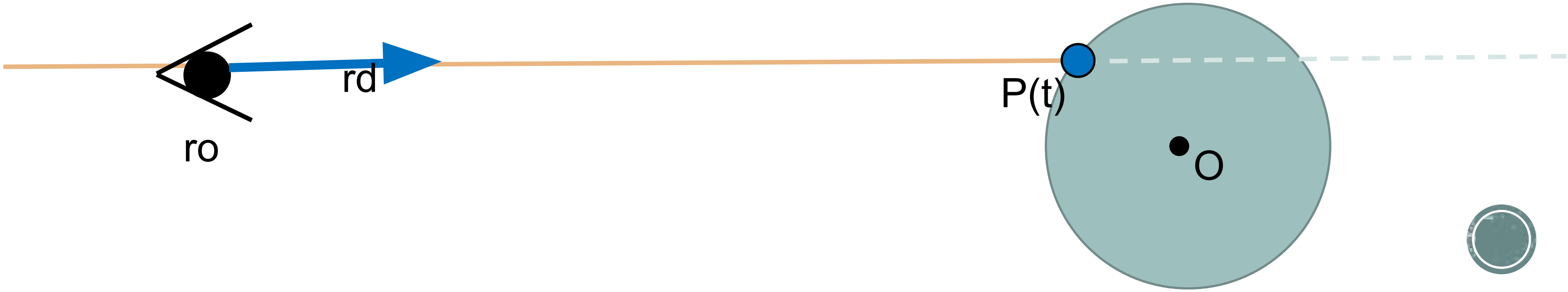
Ray-sphere intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit sphere equation: $||P - O|| - r^2 = 0$
    - $||r_o + r_d * t - O|| - r^2 = 0$
    - $(r_o + r_d * t - O) \cdot (r_o + r_d * t - O) - r^2 = 0$



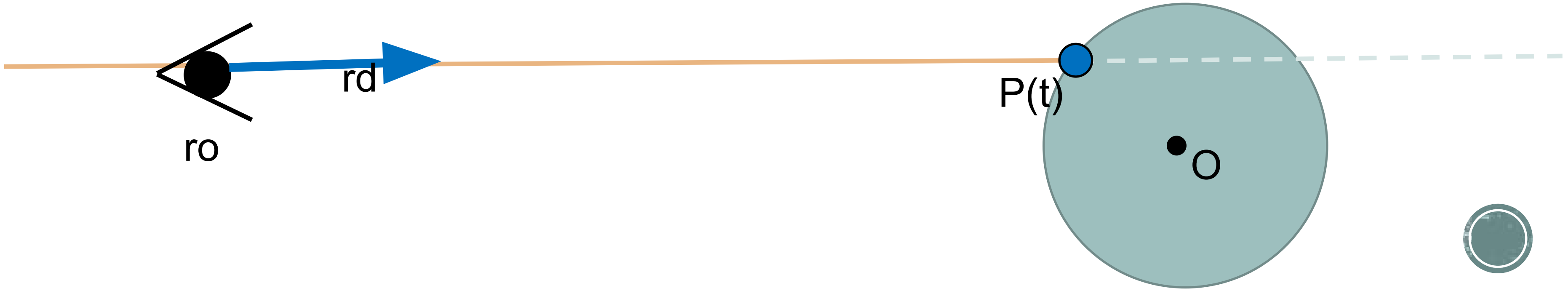
Ray-sphere intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit sphere equation: $||P - O|| - r^2 = 0$
  - $||r_o + r_d * t - O|| - r^2 = 0$
  - $(r_o + r_d * t - O) \cdot (r_o + r_d * t - O) - r^2 = 0$
  - $(r_d \cdot r_d)t^2 + (2r_o \cdot r_d - 2r_d \cdot O)t + (r_o \cdot r_o - 2r_o \cdot O + O \cdot O - r^2) = 0$



Ray-sphere intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit sphere equation: $||P - O|| - r^2 = 0$
  - $||r_o + r_d * t - O|| - r^2 = 0$
  - $(r_o + r_d * t - O) \cdot (r_o + r_d * t - O) - r^2 = 0$
  - $\boxed{(r_d \cdot r_d)} t^2 + \boxed{(2r_o \cdot r_d - 2r_d \cdot O)} t + \boxed{(r_o \cdot r_o - 2r_o \cdot O + O \cdot O - r^2)} = 0$
  - $\rightarrow at^2 + bt + c = 0, \;\; a = 1$

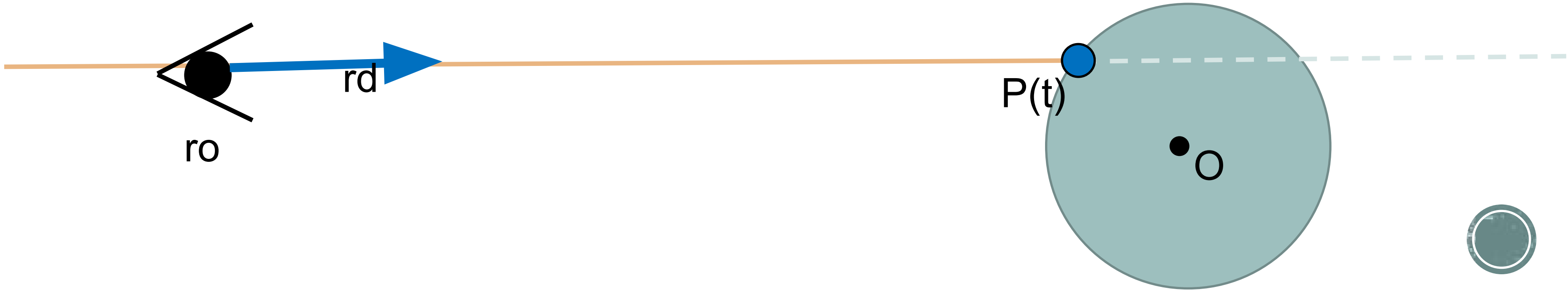


Ray-sphere intersection

- Parametric ray equation: $P(t) = r_o + r_d * t$
- Implicit sphere equation: $||P - O|| - r^2 = 0$
    - $||r_o + r_d * t - O|| - r^2 = 0$
    - $(r_o + r_d * t - O) \cdot (r_o + r_d * t - O) - r^2 = 0$
    - $\boxed{(r_d \cdot r_d)} t^2 + \boxed{(2r_o \cdot r_d - 2r_d \cdot O)} t + \boxed{(r_o \cdot r_o - 2r_o \cdot O + O \cdot O - r^2)} = 0$
    - $\rightarrow at^2 + bt + c = 0, \;\; a = 1$

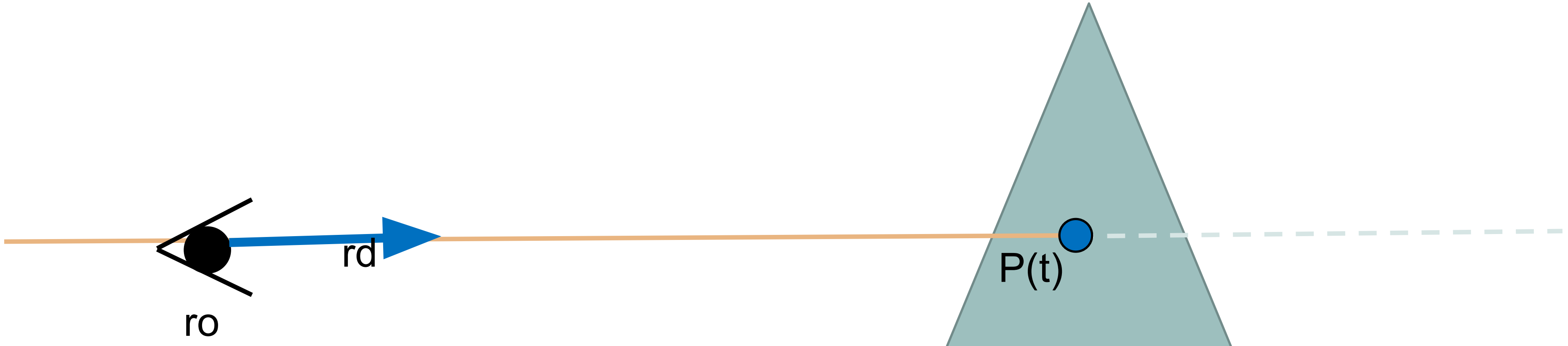
$$d = \sqrt{(}b^2 - 4ac)$$

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



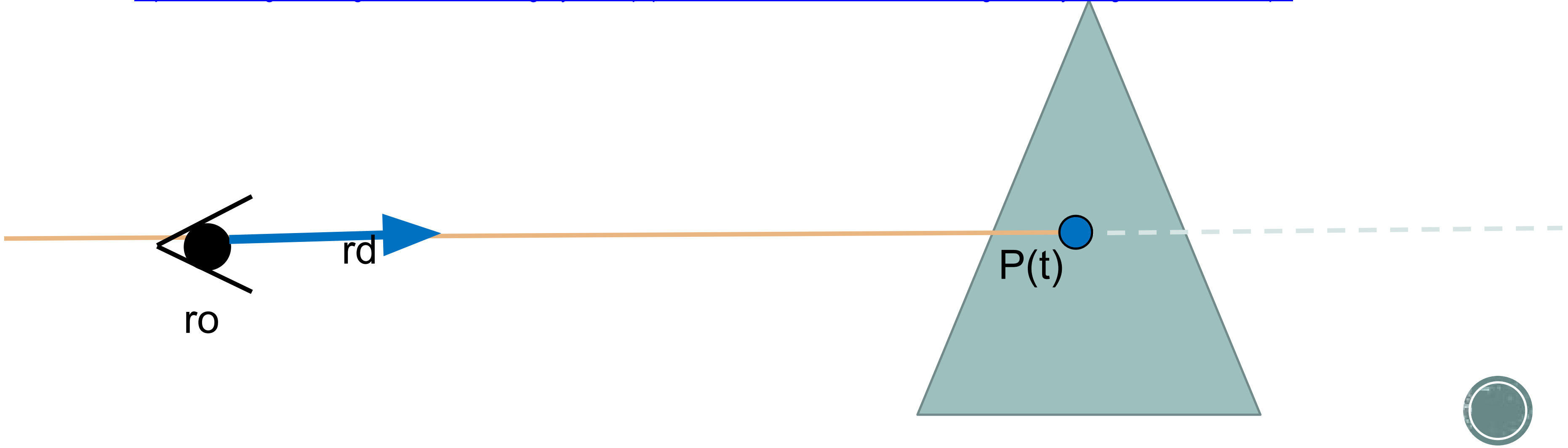
Ray-triangle intersection

- Ray-plane intersection
- Then test each edge…



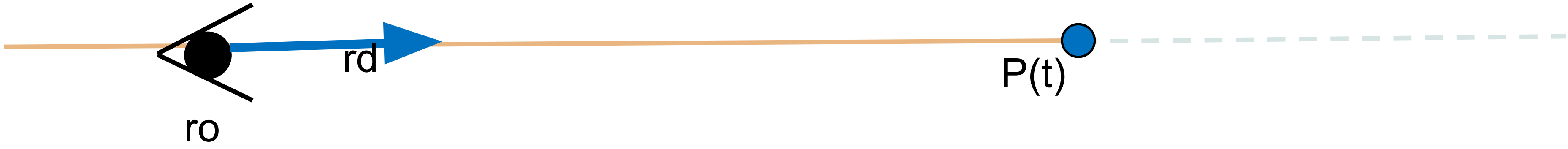
Ray-triangle intersection

- Ray-plane intersection
- Then test each edge…
- Better: parametric solution [Moller & Trumbore 97]
  - $T(u,v) = (1-u-v)V_0 + uV_1 + vV_2, \qquad u \geq 0,\ v \geq 0 \quad \text{and } u+v \leq 1.$
  - $O + tD = (1-u-v)V_0 + uV_1 + vV_2$
  - $\begin{bmatrix} -D, & V_1 - V_0, & V_2 - V_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - V_0$
  - See http://www.cs.virginia.edu/~gfx/Courses/2003/ImageSynthesis/papers/Acceleration/Fast%20MinimumStorage%20RayTriangle%20Intersection.pdf



Other intersections

- Cone, cylinder, elipsoid
  - Similar to sphere
- Box
  - 3 front facing planes
- Convex polygon
  - Similar to triangles
- Concav polygon
  - More complex point-in-polygon test



Ray-casting: summary

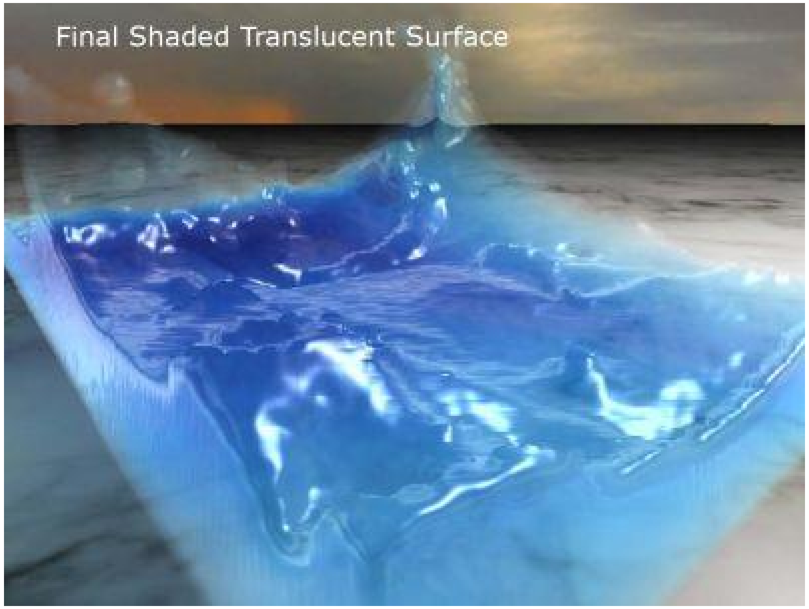
- For each pixel
  - Compute eye ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection
    - Shade depending on light and normal vector

OK

Ray-casting: summary

- For each pixel
  - Compute eye ray
  - For each object
    - Check ray-object intersection
    - Get closest intersection
    - Shade depending on light and normal vector

What if intersection cannot be computed analytically?



References

- MIT:
  - http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-837-computer-graphics-fall-2012/lecture-notes/
- Standford:
  - http://candela.stanford.edu/cs348b-14/doku.php
- Siggraph:
  - http://blog.selfshadow.com/publications/s2014-shading-course/
  - http://blog.selfshadow.com/publications/s2013-shading-course/
- Image synthesis & OpenGL:
  - http://romain.vergne.free.fr/blog/?page_id=97
- Path tracing and global illum:
  - http://www.graphics.stanford.edu/courses/cs348b-01/course29.hanrahan.pdf
  - http://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/realistic_raytracing.html
- GLSL / Shadertoy:
  - https://www.opengl.org/documentation/glsl/
  - https://www.shadertoy.com/
  - http://www.iquilezles.org/
- http://fileadmin.cs.lth.se/cs/Education/EDAN30/lectures/L2-rt.pdf