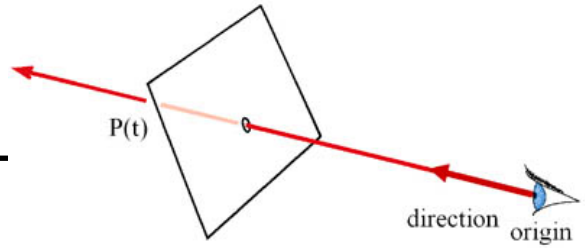
50.017 Graphics and Visualization

Ray Casting II

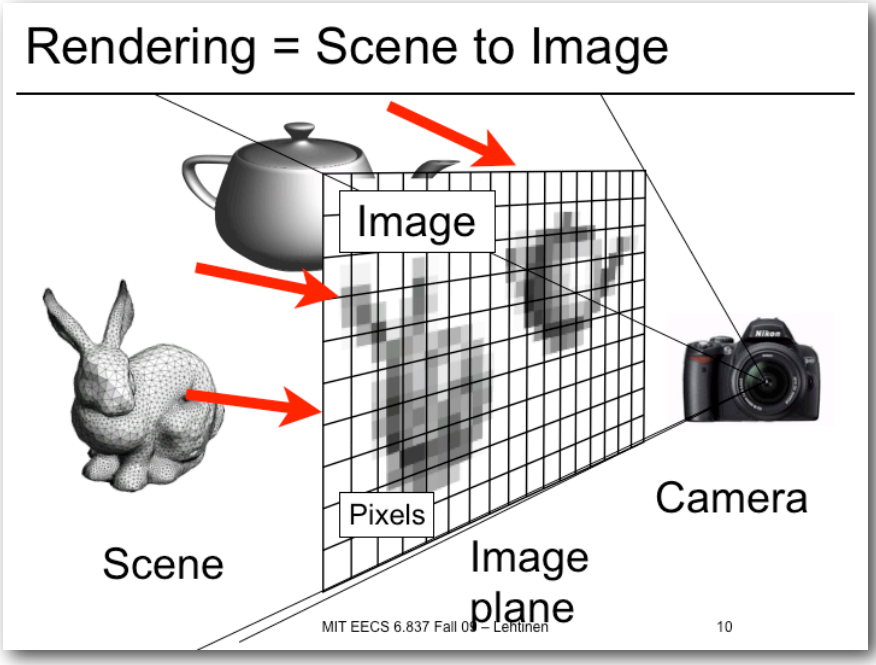
Sai-Kit Yeung SUTD ISTD

Notes courtesy by Wojciech Matusik

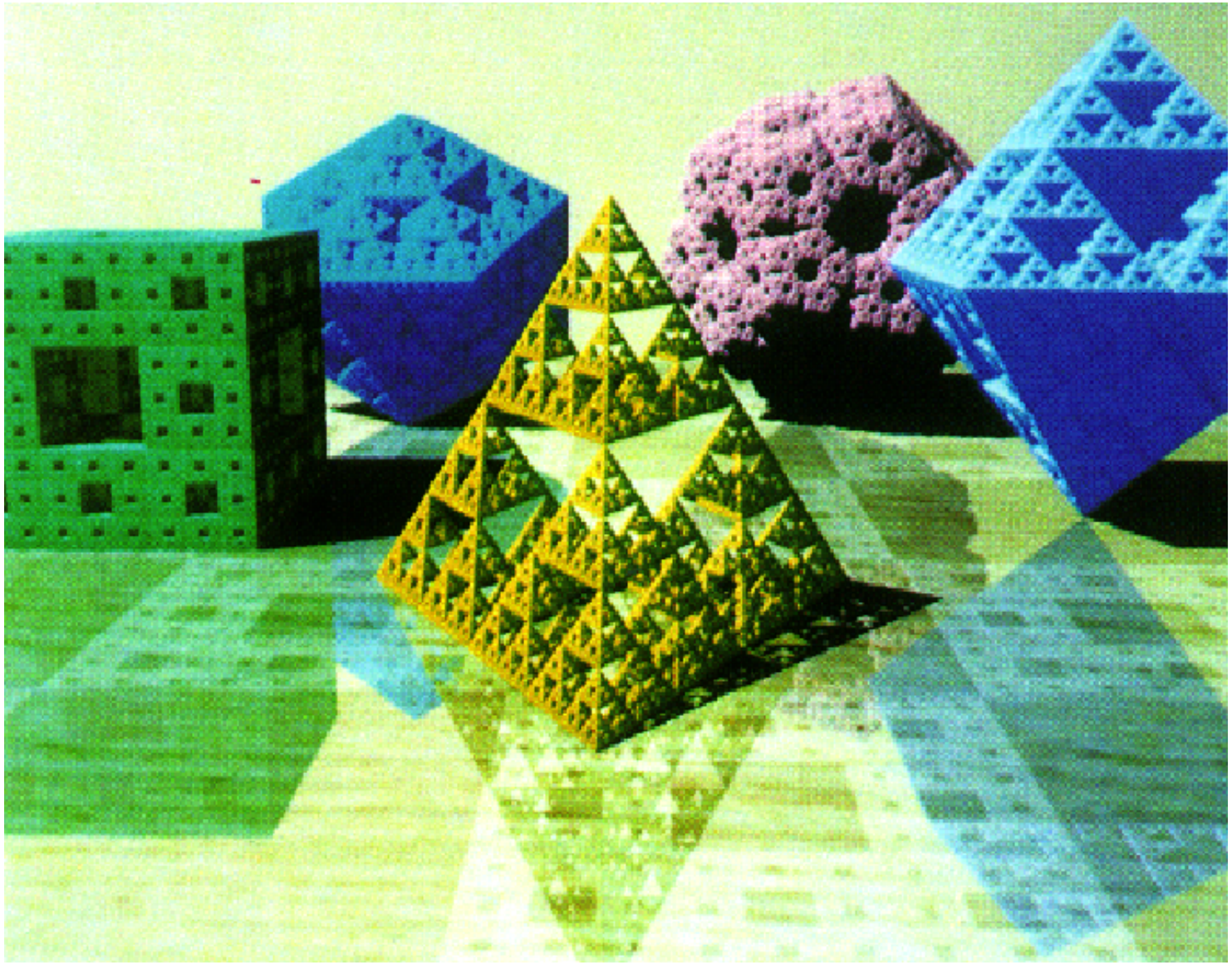
Thursday Recap



- Intro to rendering
  - Producing a picture based on scene description
  - Main variants: Ray casting/tracing vs. rasterization
  - Ray casting vs. ray tracing (secondary rays)
- Ray Casting basics
  - Camera definitions
    - Orthographic, perspective
  - Ray representation
    - P(t) = origin + t * direction
  - Ray generation
  - Ray/plane intersection
  - Ray-sphere intersection

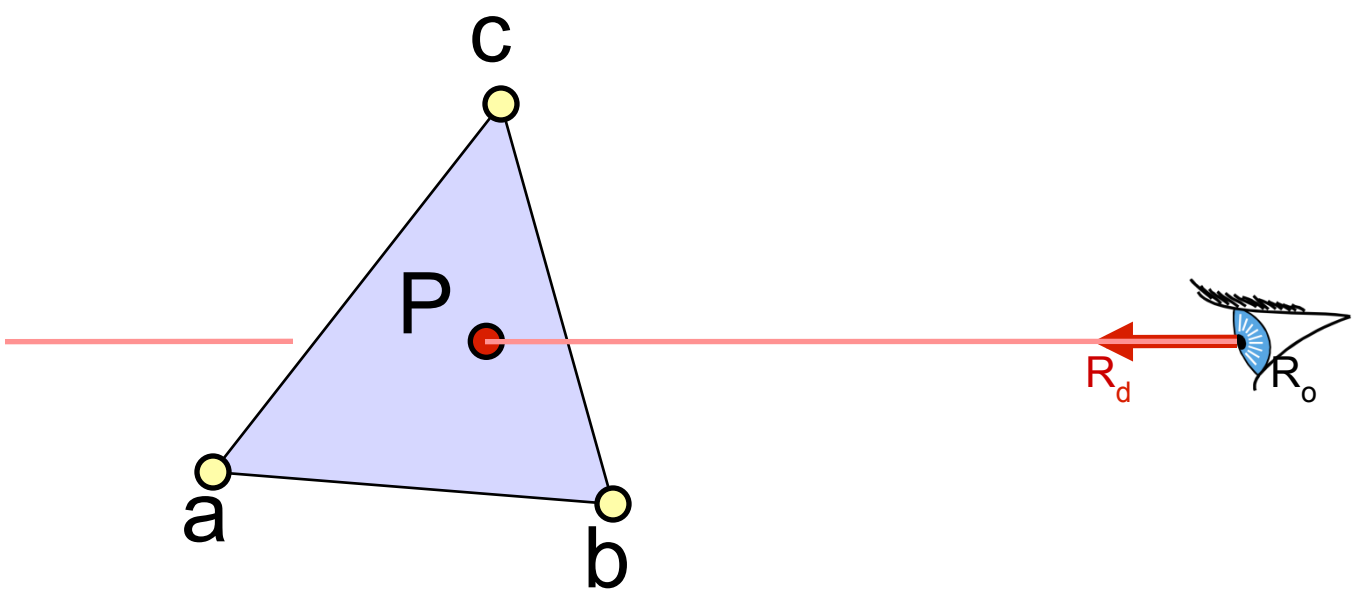


Questions?



Ray-Triangle Intersection

- Use ray-plane intersection followed by in-triangle test
- Or try to be smarter
  - Use barycentric coordinates

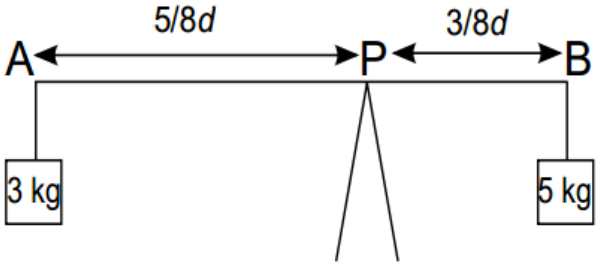


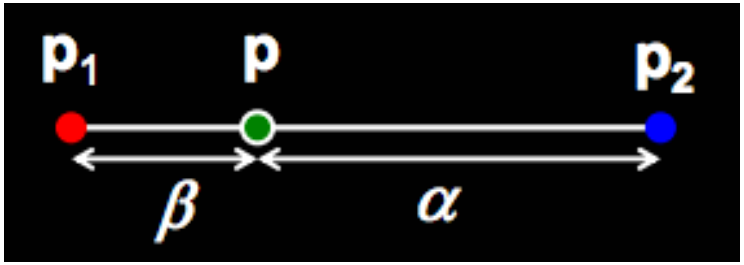
Barycentric coordinate

- Introduced by August Ferdinand Möbius [1827]
  - weightless rod with weights attached at two distinct points,
  - Locate the rod's centre of gravity

Barycentric coordinate in 1D

- Distance between A and B is *d*
- Geometric intuition
  - Weight each vertex by ratio of distances from ends
- Linear interpolation
  - Coordinate, color, normal, …
- $p(t) = \alpha\, p_1 + \beta p_2$ where $\alpha+\beta = 1$
- α, β are called barycentric coordinates



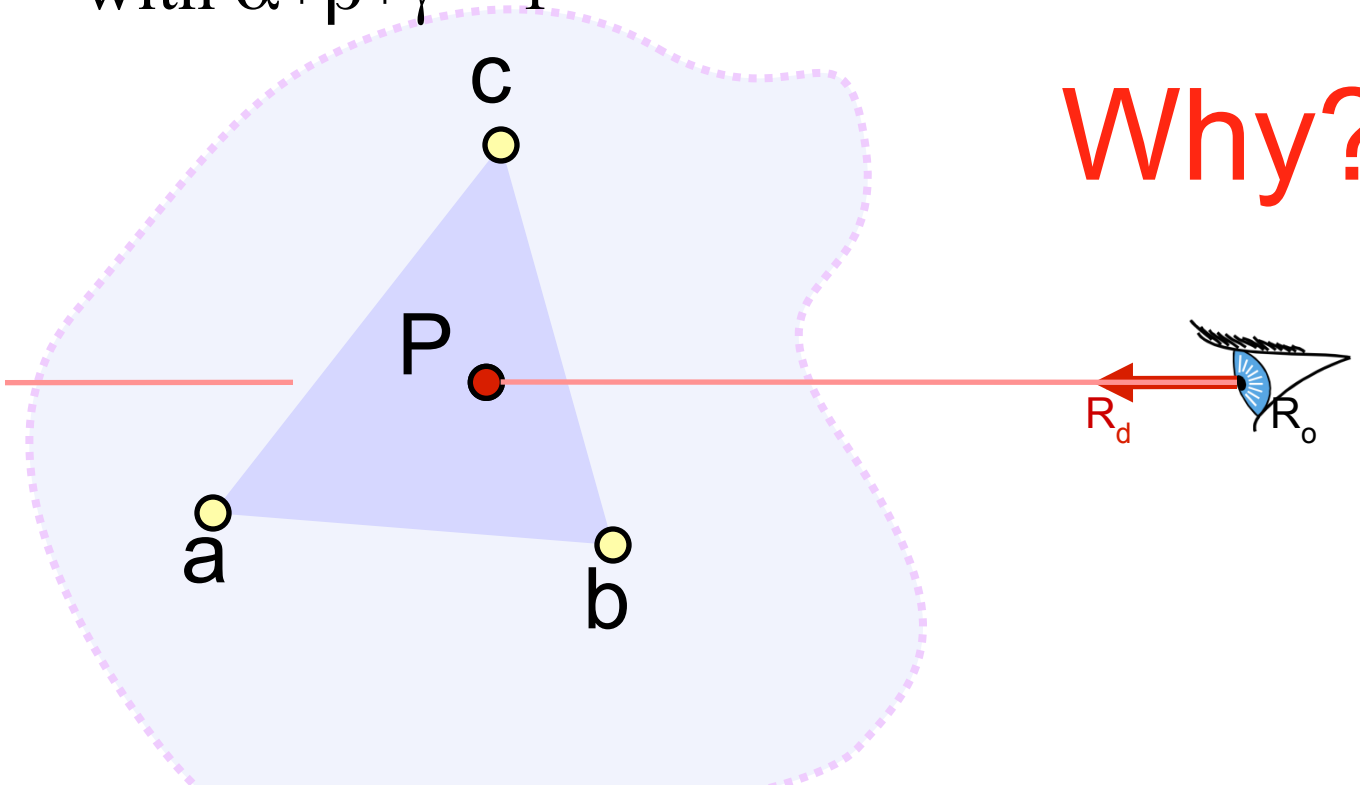


Barycentric Definition of a Plane

- A (non-degenerate) triangle (**a**,**b**,**c**) defines a plane
- Any point **P** on this plane can be written as

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

  with $\alpha+\beta+\gamma = 1$



Why? How?

[Möbius, 1827]

Barycentric Coordinates

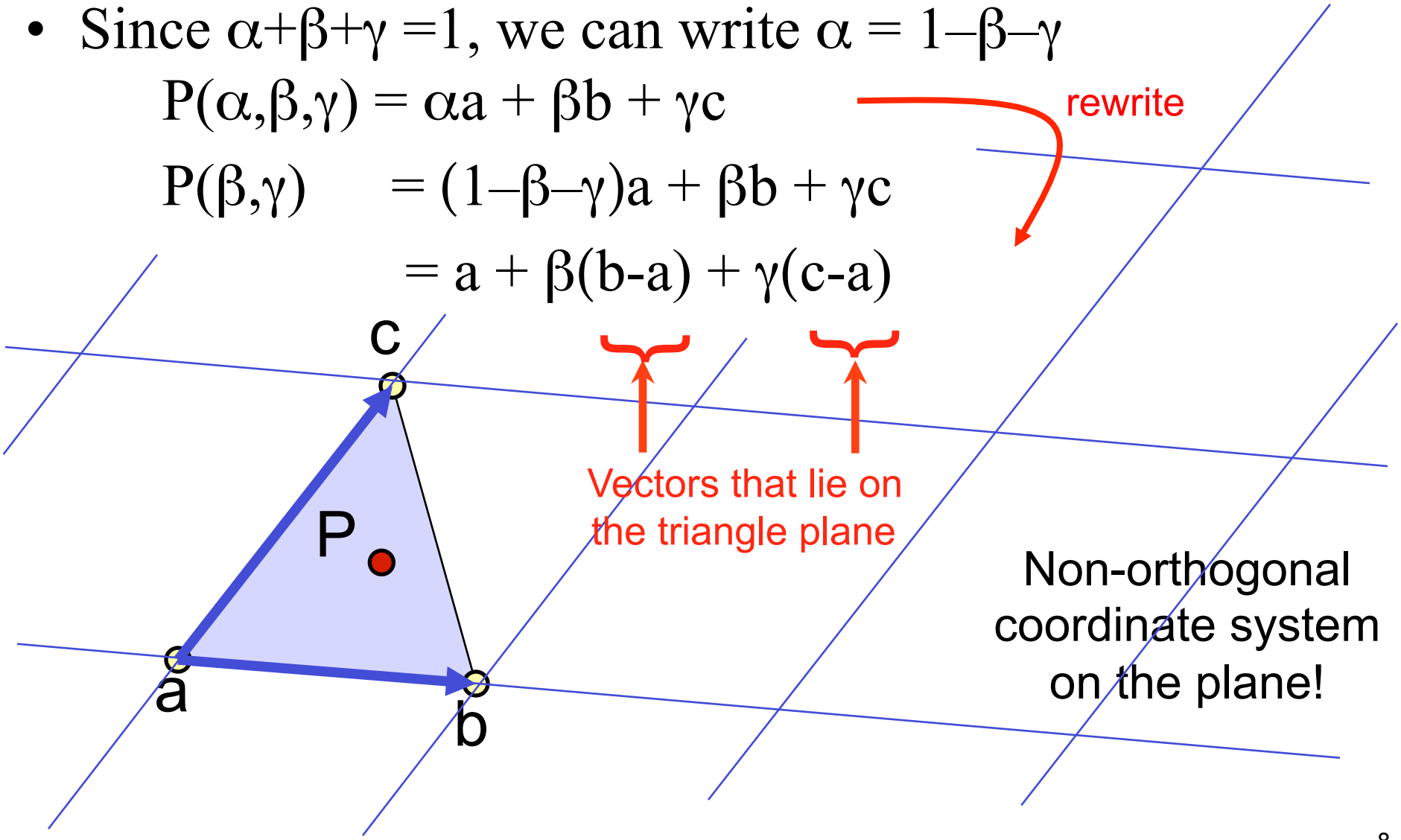
- Since $\alpha+\beta+\gamma=1$, we can write $\alpha = 1-\beta-\gamma$

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

rewrite

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

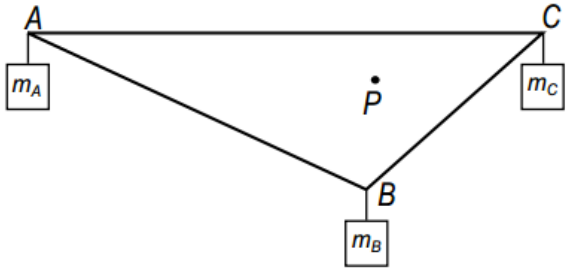
$$= a + \beta(b-a) + \gamma(c-a)$$

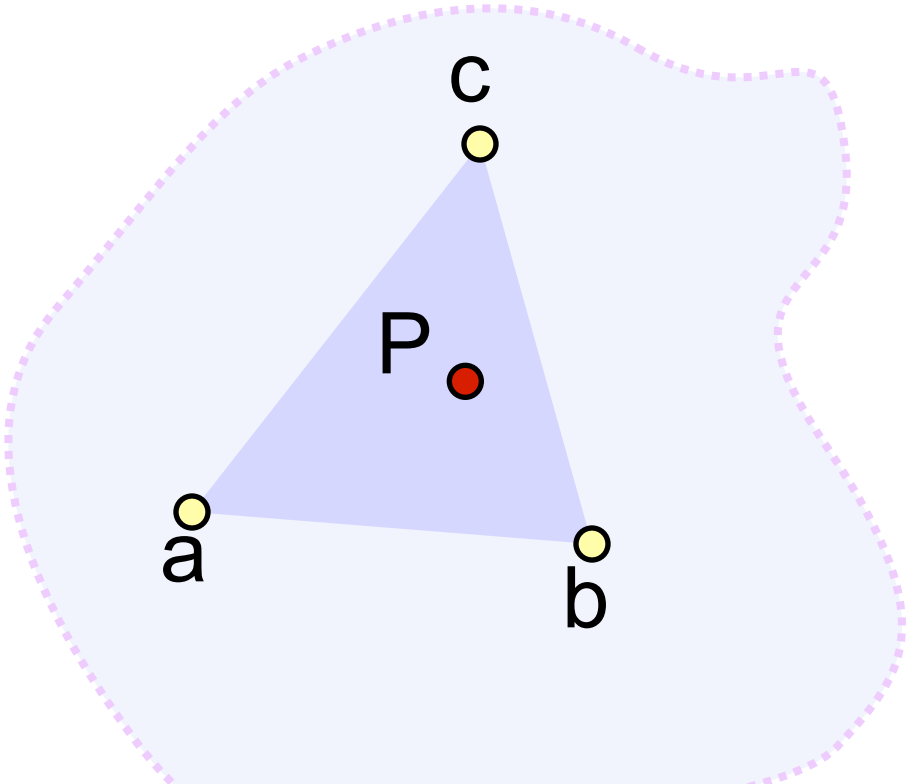


Barycentric Definition of a Plane

[Möbius, 1827]

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

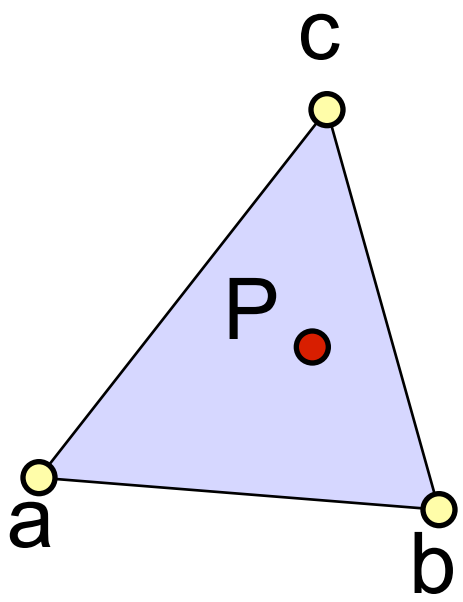




Fun to know:
P is the barycenter,
the single point upon which the triangle would balance if weights of size α, β, & γ are placed on points a, b & c.

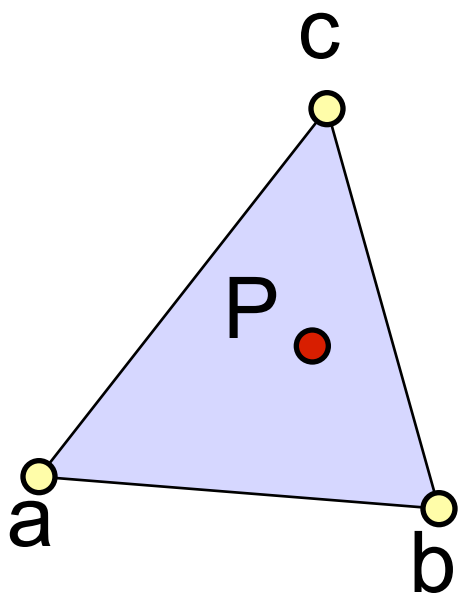
Barycentric Definition of a Triangle

- $P(\alpha,\beta,\gamma) = \alpha a + \beta b + \gamma c$
  with $\alpha+\beta+\gamma = 1$ parameterizes the entire plane



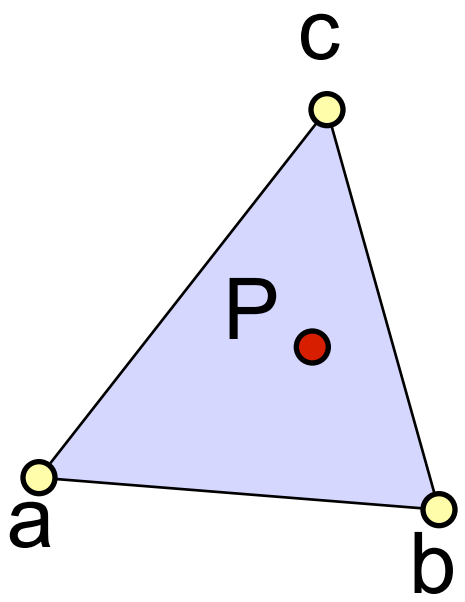
Barycentric Definition of a Triangle

- $P(\alpha,\beta,\gamma) = \alpha a + \beta b + \gamma c$
  with $\alpha+\beta+\gamma = 1$ parameterizes the entire plane
- If we require in addition that
  $\alpha, \beta, \gamma >= 0$, we get just the triangle!
  - Note that with $\alpha+\beta+\gamma = 1$ this implies
    $0 \le \alpha \le 1 \quad \& \quad 0 \le \beta \le 1 \quad \& \quad 0 \le \gamma \le 1$
  - Verify:
    - $\alpha = 0 \Rightarrow$ P lies on line b-c
    - $\alpha, \beta = 0 \Rightarrow P = c$
    - etc.



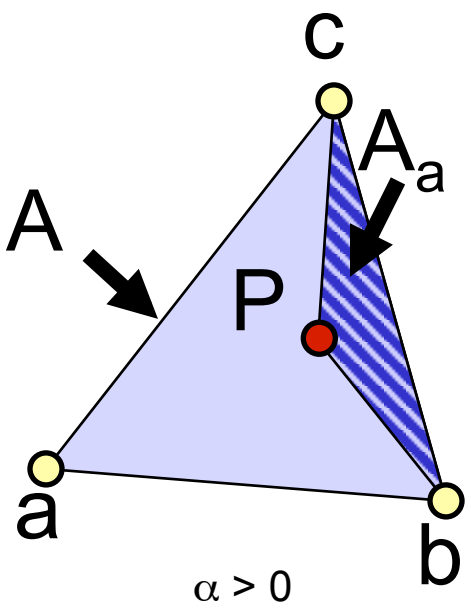
Barycentric Definition of a Triangle

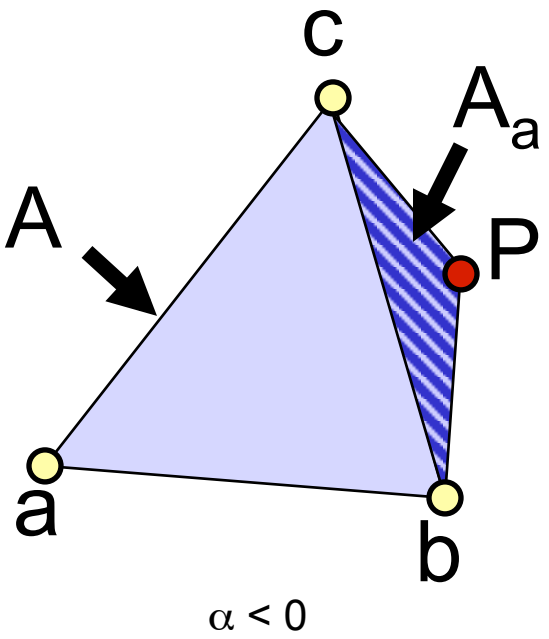
- $P(\alpha,\beta,\gamma) = \alpha a + \beta b + \gamma c$
- Condition to be barycentric coordinates:
  $\alpha+\beta+\gamma = 1$
- Condition to be inside the triangle:
  $\alpha, \beta, \gamma \geq 0$



How Do We Compute α, β, γ ?

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





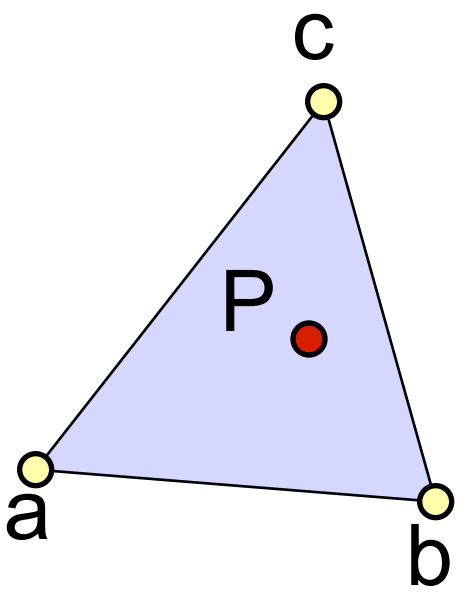
How Do We Compute α, β, γ ?

- Or write it as a 2×2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$

  $e_1 = (b-a)$, $e_2 = (c-a)$

$$\boldsymbol{a} + \beta \boldsymbol{e}_1 + \gamma \boldsymbol{e}_2 - \boldsymbol{P} = 0$$

This should be zero



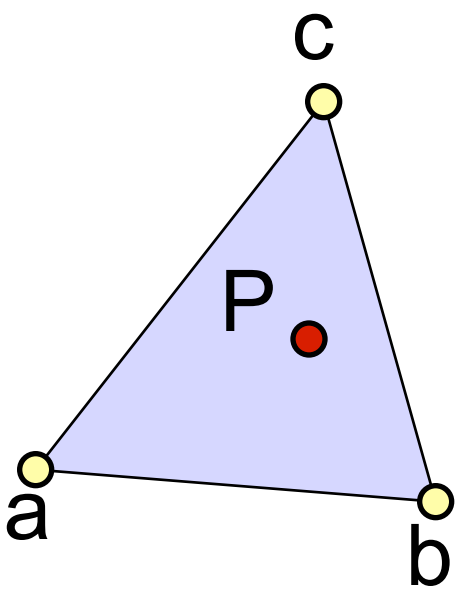
How Do We Compute α, β, γ ?

- Or write it as a 2×2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$

  $e_1 = (b-a)$, $e_2 = (c-a)$

$$\boldsymbol{a} + \beta \boldsymbol{e}_1 + \gamma \boldsymbol{e}_2 - \boldsymbol{P} = 0$$

This should be zero



Something's wrong... This is a linear system of 3 equations and 2 unknowns!

Why?

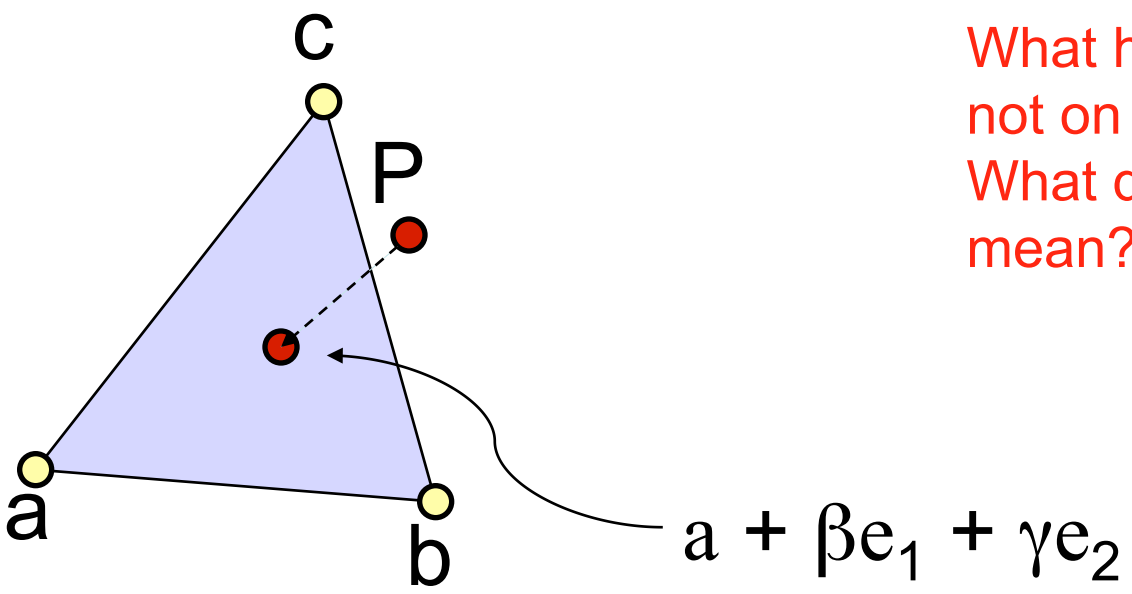
How Do We Compute α, β, γ ?

- Or write it as a 2×2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$

  $e_1 = (b-a)$, $e_2 = (c-a)$

$$\boldsymbol{a} + \beta \boldsymbol{e}_1 + \gamma \boldsymbol{e}_2 - \boldsymbol{P}$$

What happen if P is not on the plane? What does the above mean?



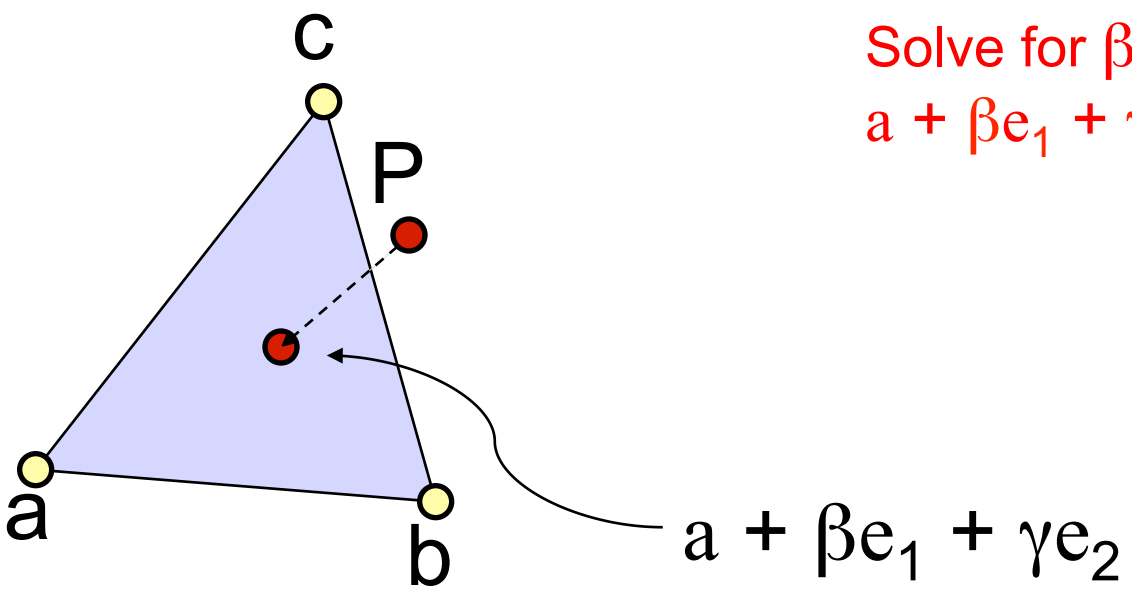
How Do We Compute α, β, γ ?

- Or write it as a 2×2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$

  $e_1 = (b-a)$, $e_2 = (c-a)$

$$\boldsymbol{a} + \beta \boldsymbol{e}_1 + \gamma \boldsymbol{e}_2 - \boldsymbol{P}$$

Solve for β, γ such that
$a + \beta e_1 + \gamma e_2$ is closet to P



How Do We Compute α, β, γ ?

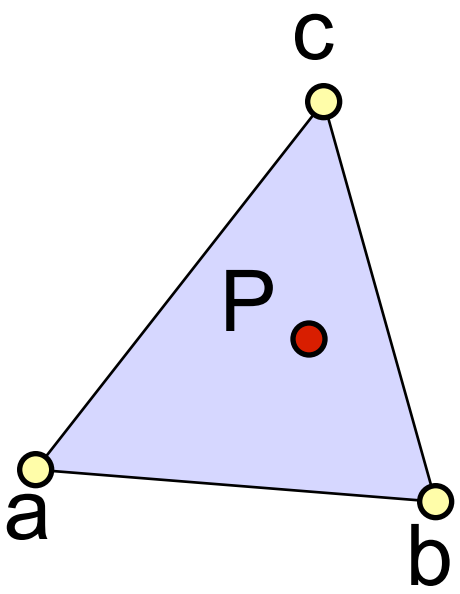
- Or write it as a 2×2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$

  $e_1 = (b-a)$, $e_2 = (c-a)$

$$\langle e_1,\ a + \beta e_1 + \gamma e_2 - P \rangle = 0$$

$$\langle e_2,\ a + \beta e_1 + \gamma e_2 - P \rangle = 0$$

These should be zero



Ha! We'll take inner products of this equation with e_1 & e_2

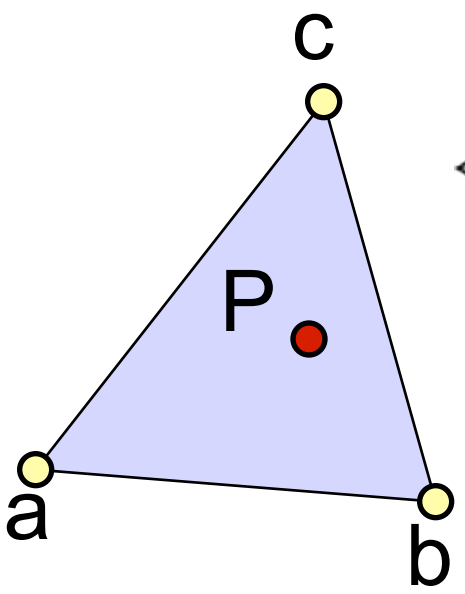
How Do We Compute α, β, γ ?

- Or write it as a 2×2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$

  $e_1 = (b-a)$, $e_2 = (c-a)$

$$\langle e_1,\ a + \beta e_1 + \gamma e_2 - P \rangle = 0$$

$$\langle e_2,\ a + \beta e_1 + \gamma e_2 - P \rangle = 0$$



$\Leftrightarrow$

$$\begin{pmatrix} \langle e_1, e_1 \rangle & \langle e_1, e_2 \rangle \\ \langle e_2, e_1 \rangle & \langle e_2, e_2 \rangle \end{pmatrix} \begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

where $\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \langle (P - a), e_1 \rangle \\ \langle (P - a), e_2 \rangle \end{pmatrix}$

and <a,b> is the dot product.

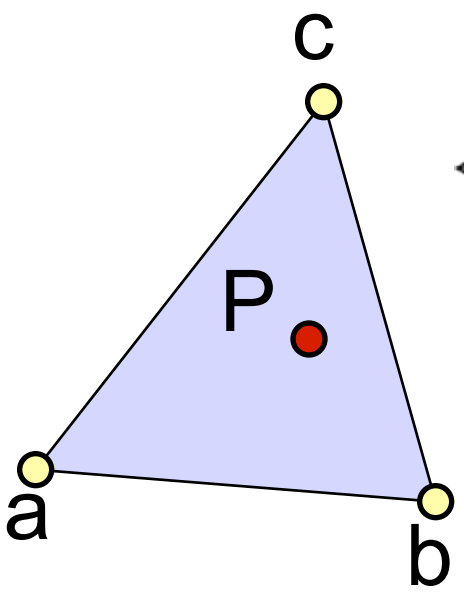
How Do We Compute α, β, γ ?

- Or write it as a 2×2 linear system
- $P(\beta, \gamma) = a + \beta e_1 + \gamma e_2$

  $e_1 = (b-a)$, $e_2 = (c-a)$

Questions?

$$\langle e_1,\ a + \beta e_1 + \gamma e_2 - P \rangle = 0$$
$$\langle e_2,\ a + \beta e_1 + \gamma e_2 - P \rangle = 0$$



$\Leftrightarrow$

$$\begin{pmatrix} \langle e_1, e_1 \rangle & \langle e_1, e_2 \rangle \\ \langle e_2, e_1 \rangle & \langle e_2, e_2 \rangle \end{pmatrix} \begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

where $\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \langle (P - a), e_1 \rangle \\ \langle (P - a), e_2 \rangle \end{pmatrix}$

and <a,b> is the dot product.

Intersection with Barycentric Triangle

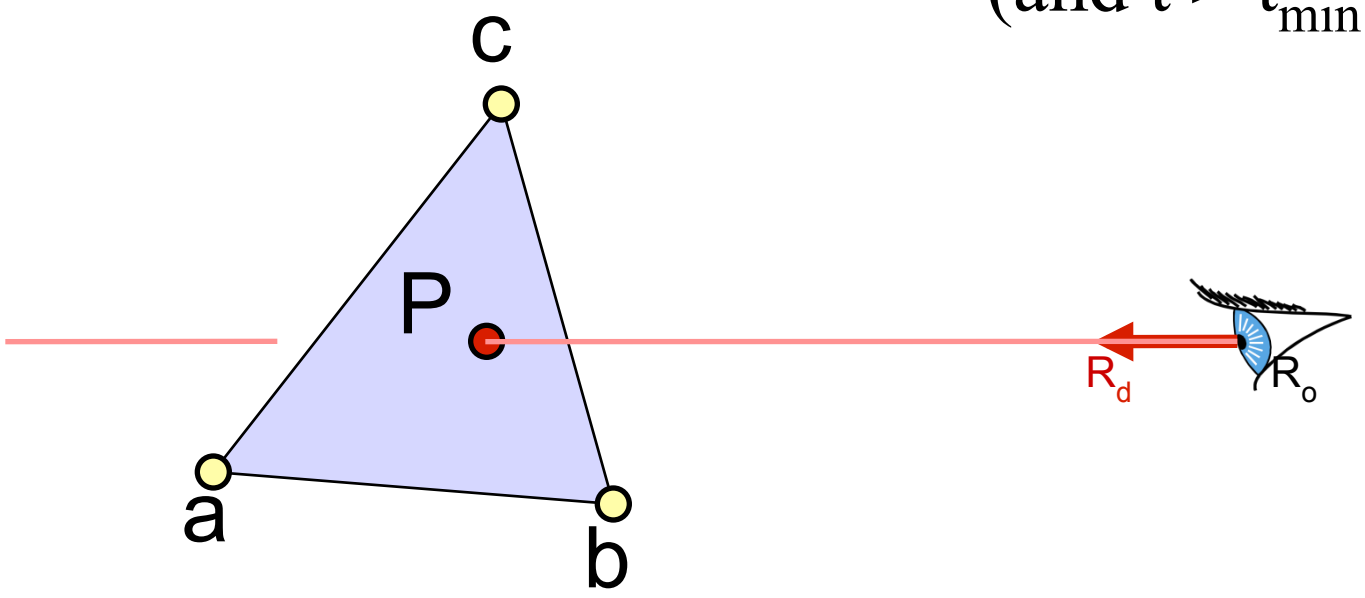
- Again, set ray equation equal to barycentric equation

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

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

- Intersection if $\beta + \gamma \leq 1$ & $\beta \geq 0$ & $\gamma \geq 0$

(and $t > t_{min} \ldots$)



Intersection with Barycentric Triangle

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

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

3 equations,
3 unknowns

- Regroup & write in matrix form Ax=b

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

Cramer's Rule

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

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

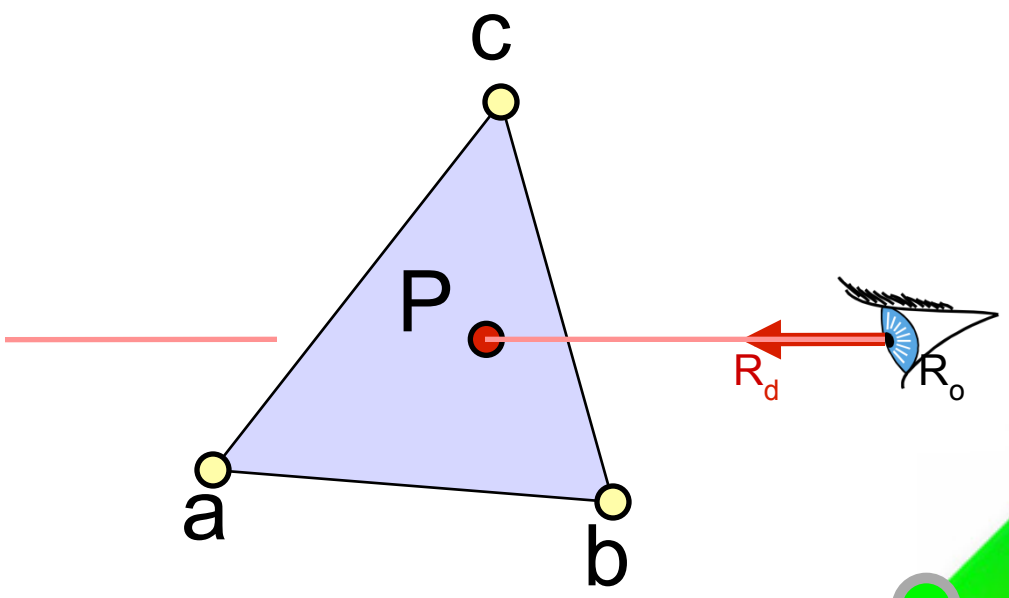
$$t = \frac{\begin{vmatrix} a_x - b_x & a_x - c_x & a_x - R_{ox} \\ a_y - b_y & a_y - c_y & a_y - R_{oy} \\ a_z - b_z & a_z - c_z & a_z - R_{oz} \end{vmatrix}}{|A|}$$

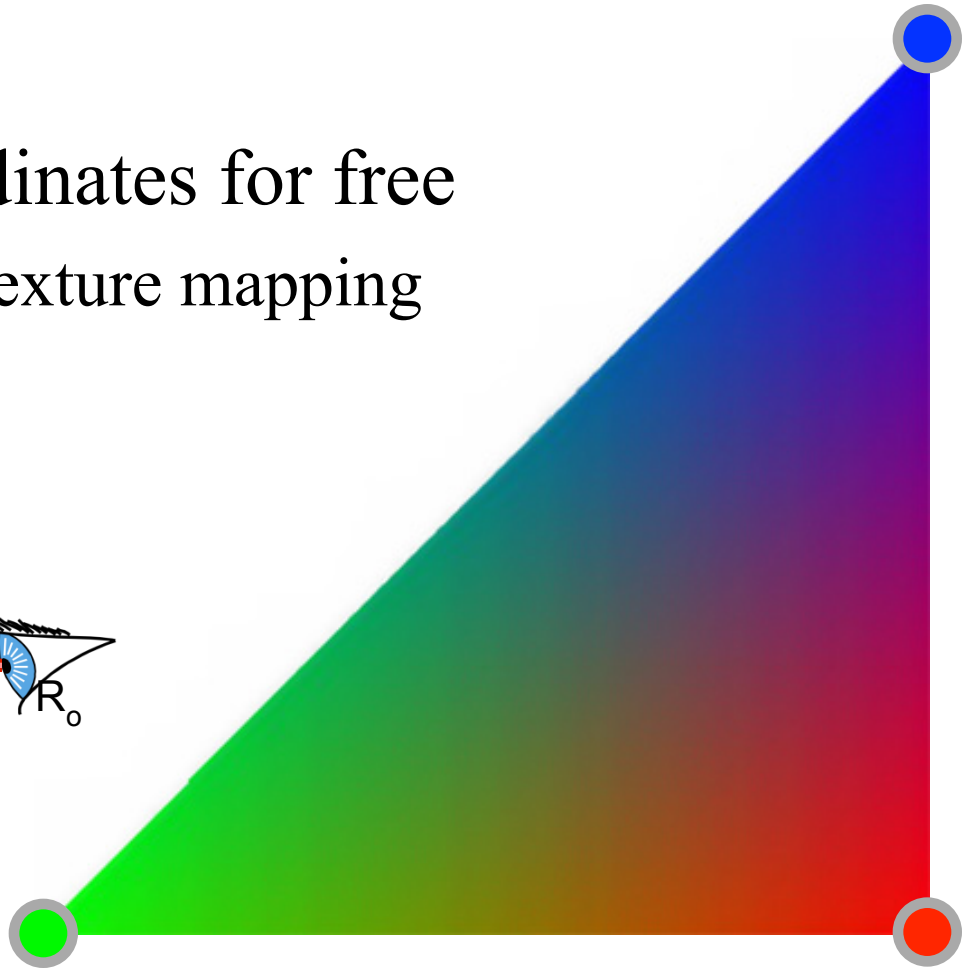
| | denotes the determinant

Can be copied mechanically into code

Barycentric Intersection Pros

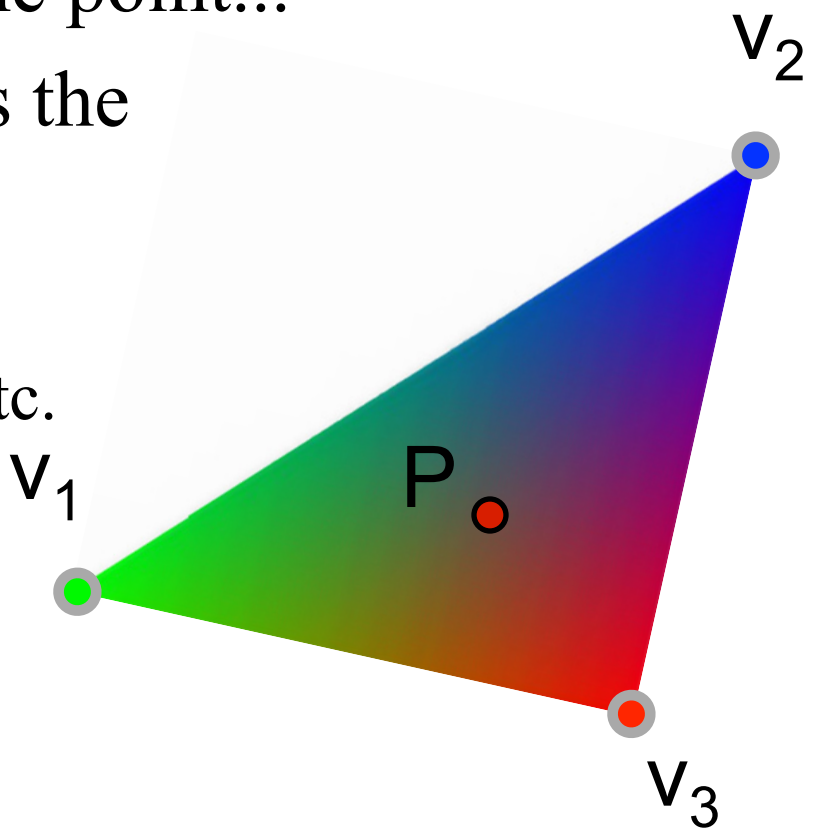
- Efficient
- Stores no plane equation
- Get the barycentric coordinates for free
  - Useful for interpolation, texture mapping





Barycentric Interpolation

- Values v_1, v_2, v_3 defined at a, b, c
  - Colors, normal, texture coordinates, etc.
- $P(\alpha,\beta,\gamma) = \alpha a + \beta b + \gamma c$ is the point...
- $v(\alpha,\beta,\gamma) = \alpha v_1 + \beta v_2 + \gamma v_3$ is the barycentric interpolation of v_1,v_2,v_3 at point P
  - Sanity check: $v(1,0,0) = v_1$, etc.
- I.e, once you know α,β,γ you can interpolate values using the same weights.
  - Convenient!

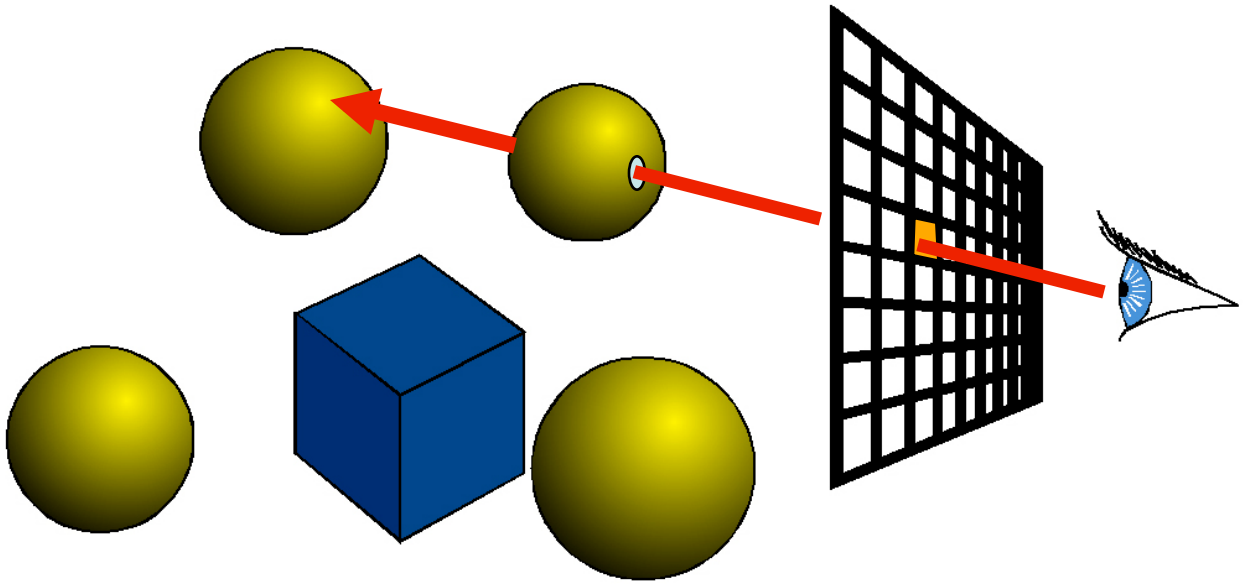


Questions?

- Image computed using the RADIANCE system by Greg Ward

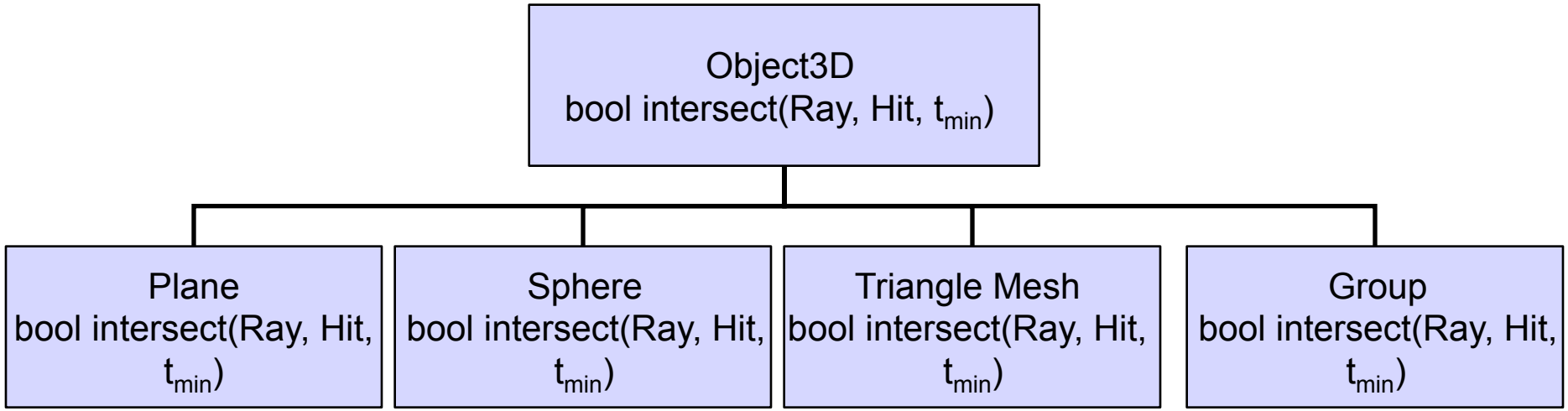
Ray Casting: Object Oriented Design

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



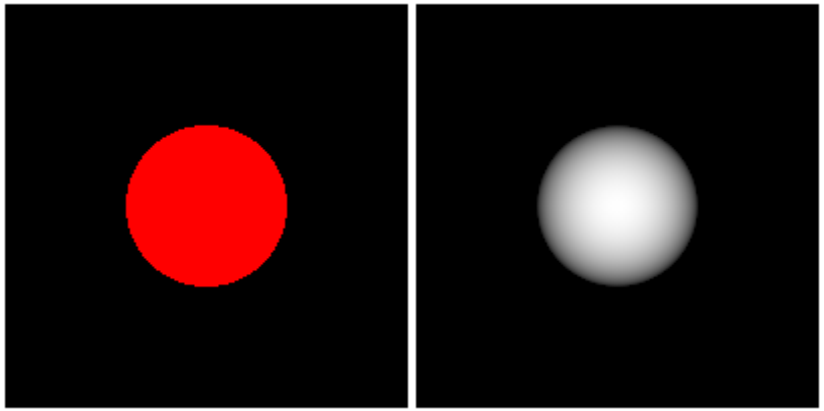
Object-Oriented Design

- We want to be able to add primitives easily
  - Inheritance and virtual methods
- Even the scene is derived from Object3D!

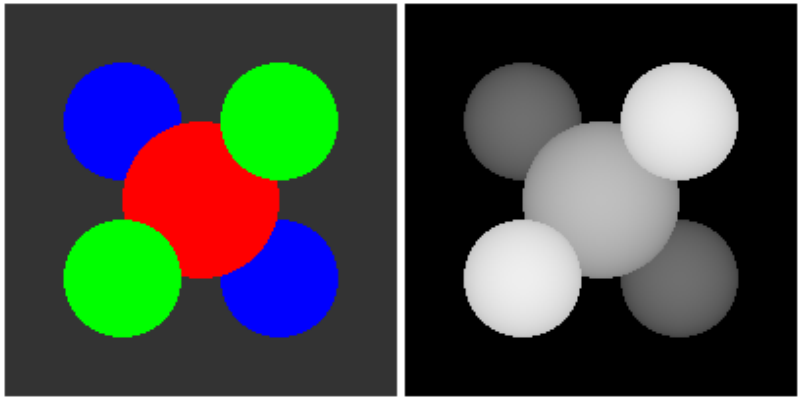


- Also cameras are abstracted (perspective/ortho)
  - Methods for generating rays for given image coordinates

Assignment 4 & 5: Ray Casting/Tracing



- Write a basic ray caster
  - Orthographic and perspective cameras
  - Spheres and triangles
  - 2 Display modes: color and distance

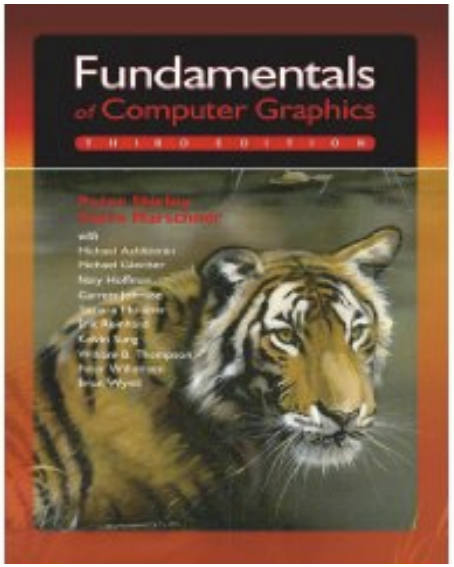


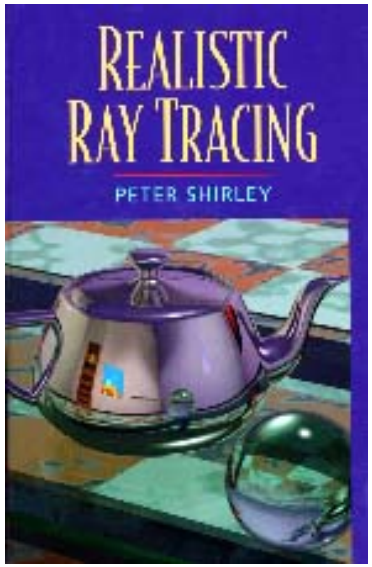
- We provide classes for
  - Ray: origin, direction
  - Hit: t, Material, (normal)
  - Scene Parsing
- You write ray generation, hit testing, simple shading

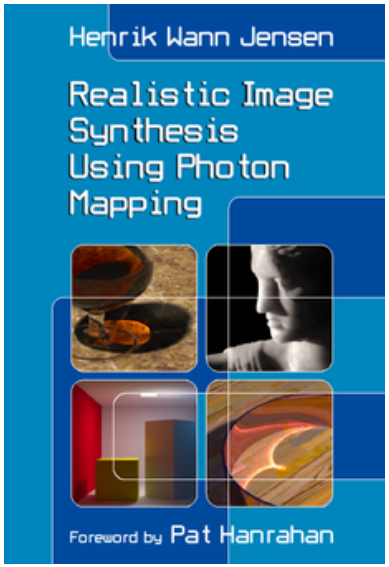
Books

- Peter Shirley et al.:
  Fundamentals of Computer Graphics
  AK Peters

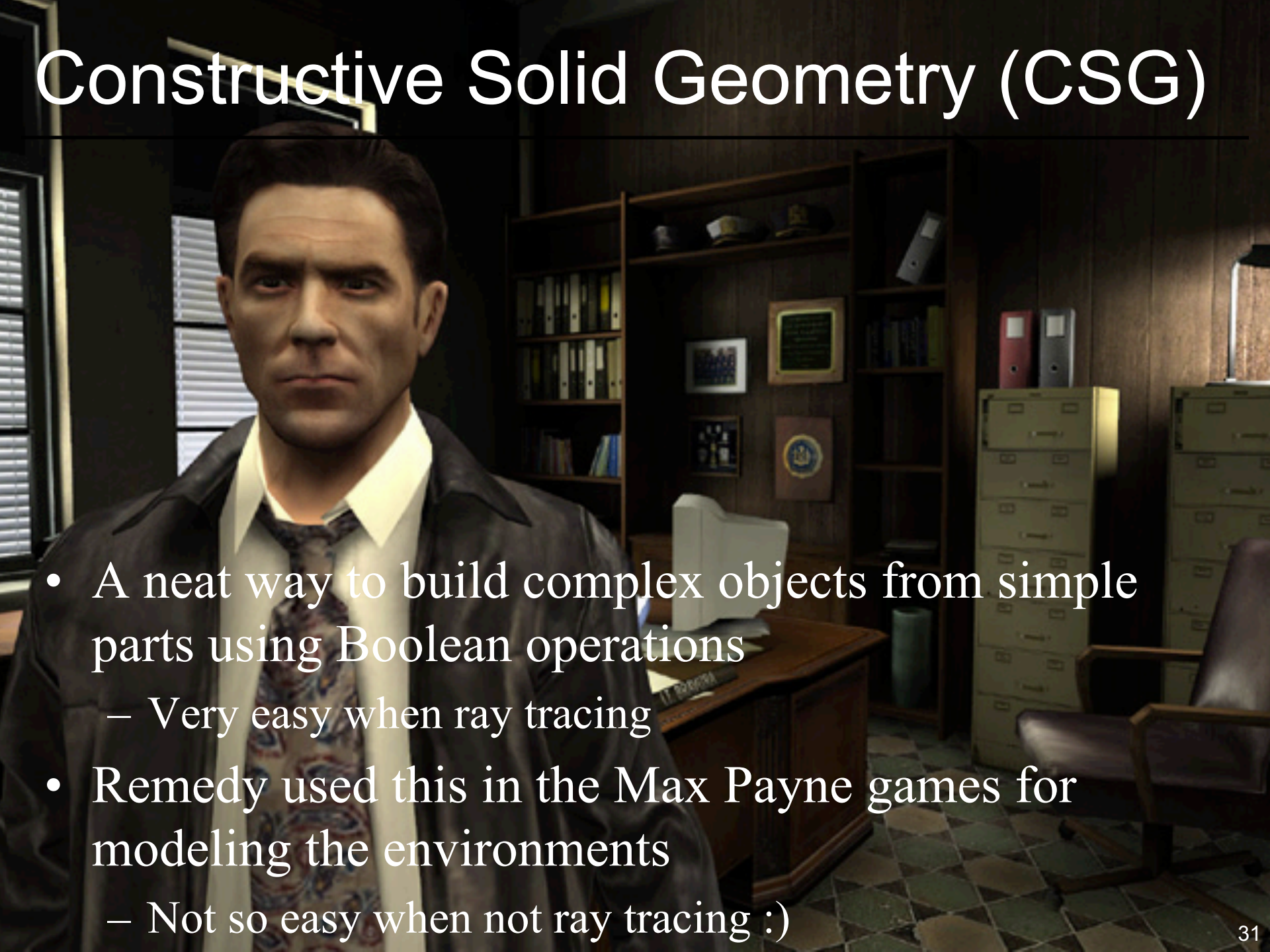
- Ray Tracing
  - Jensen
  - Shirley
  - Glassner





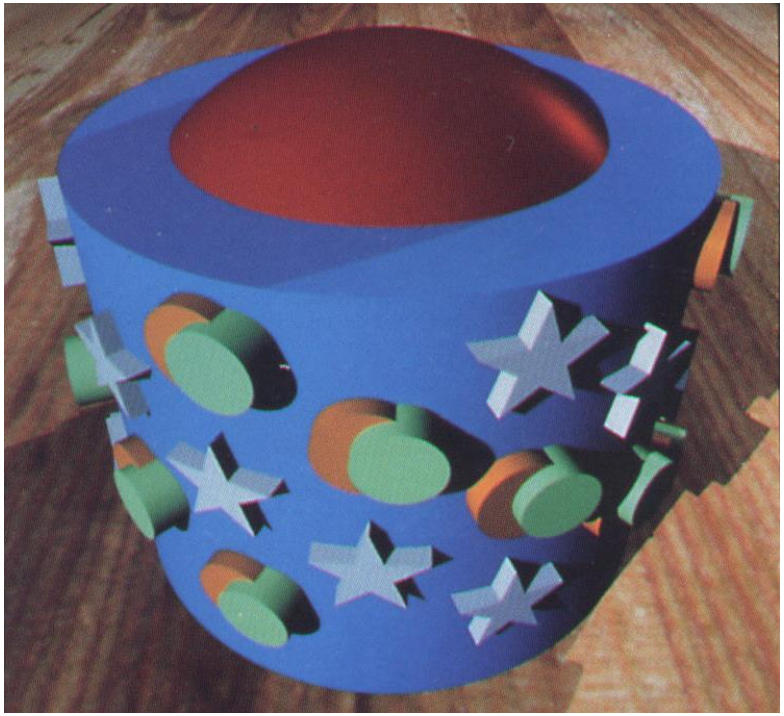


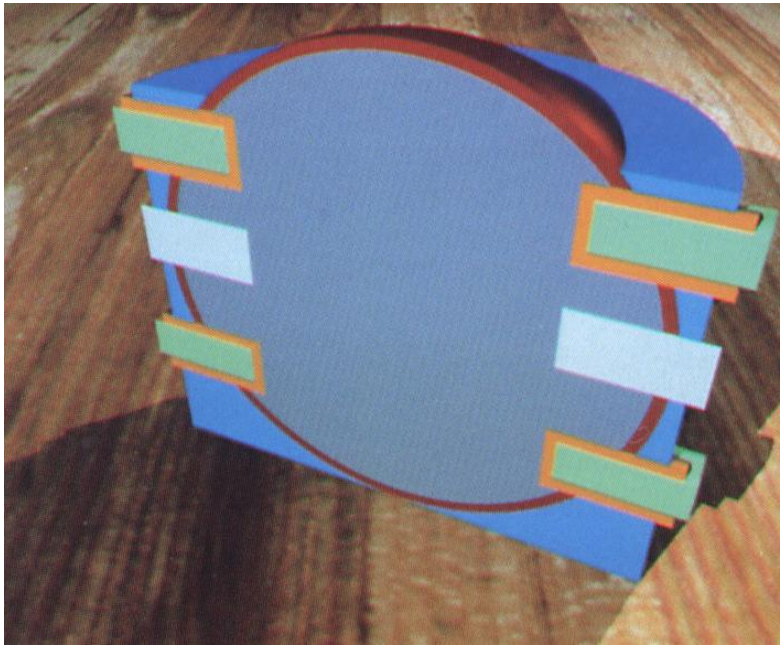
Constructive Solid Geometry (CSG)



- A neat way to build complex objects from simple parts using Boolean operations
  - Very easy when ray tracing
- Remedy used this in the Max Payne games for modeling the environments
  - Not so easy when not ray tracing :)

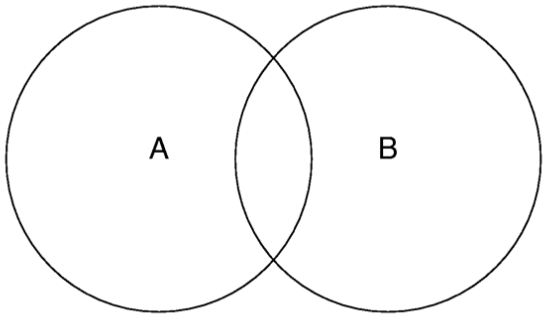
CSG Examples



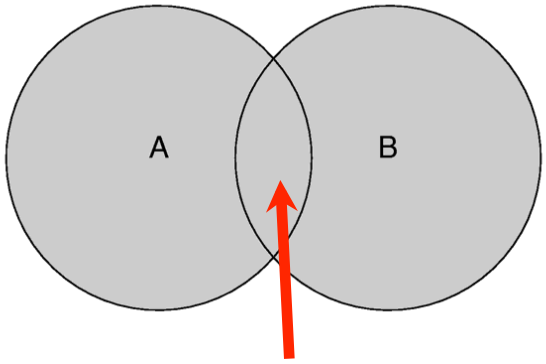


Constructive Solid Geometry (CSG)

Given overlapping shapes A and B:

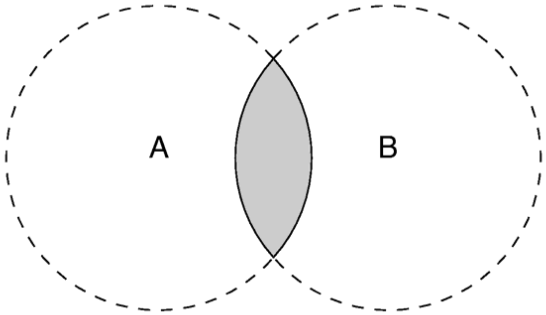


Union

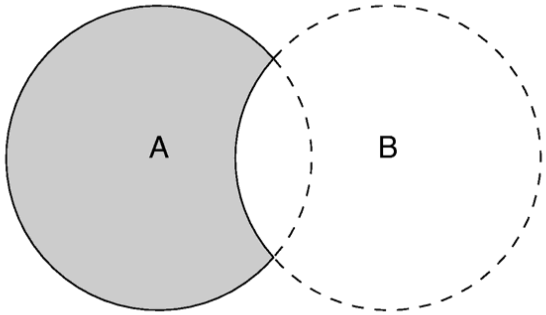


Should only “count” overlap region once!

Intersection

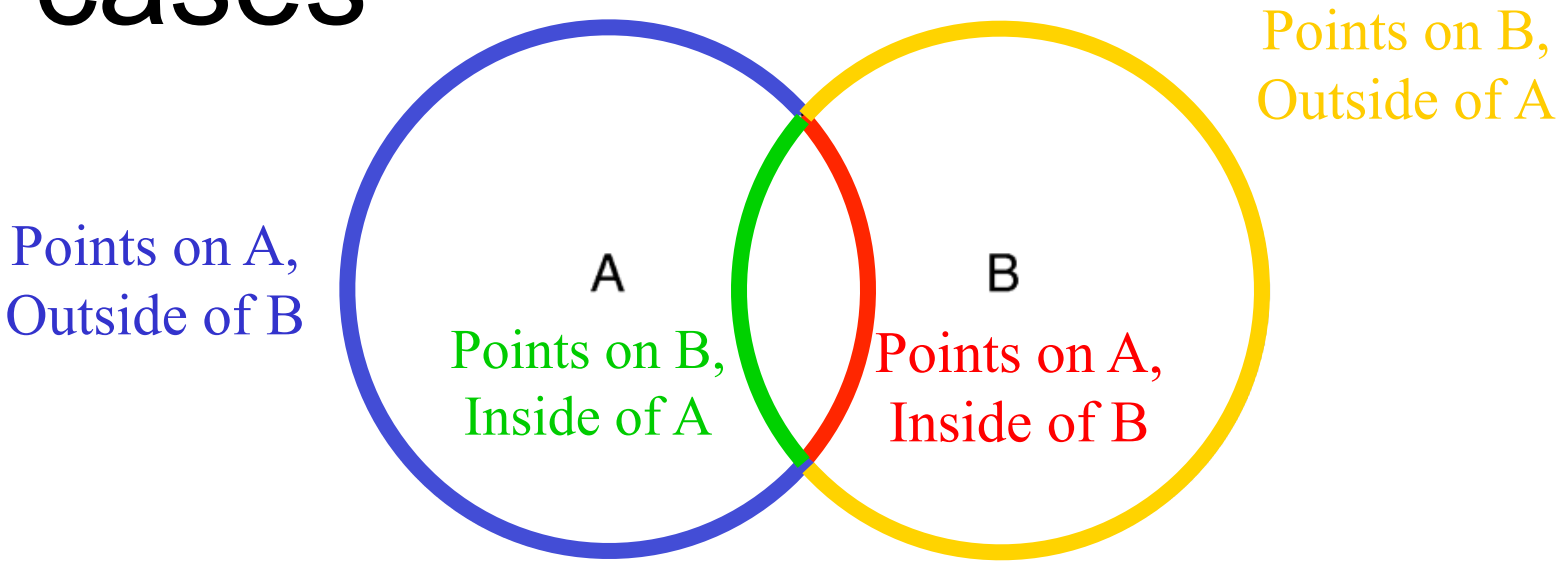


Subtraction

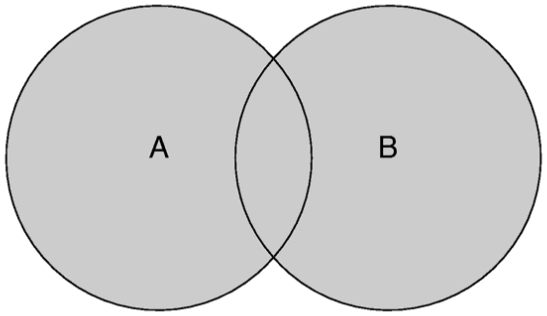


How Can We Implement CSG?

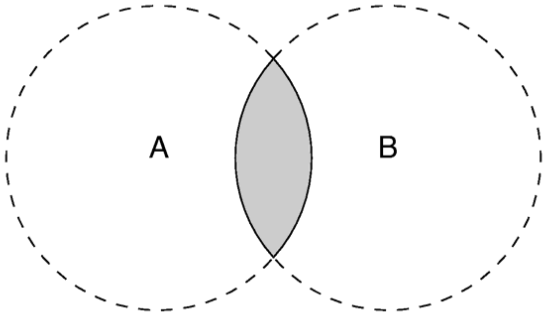
4 cases



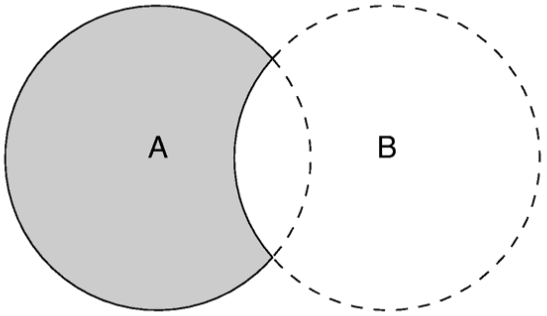
Union



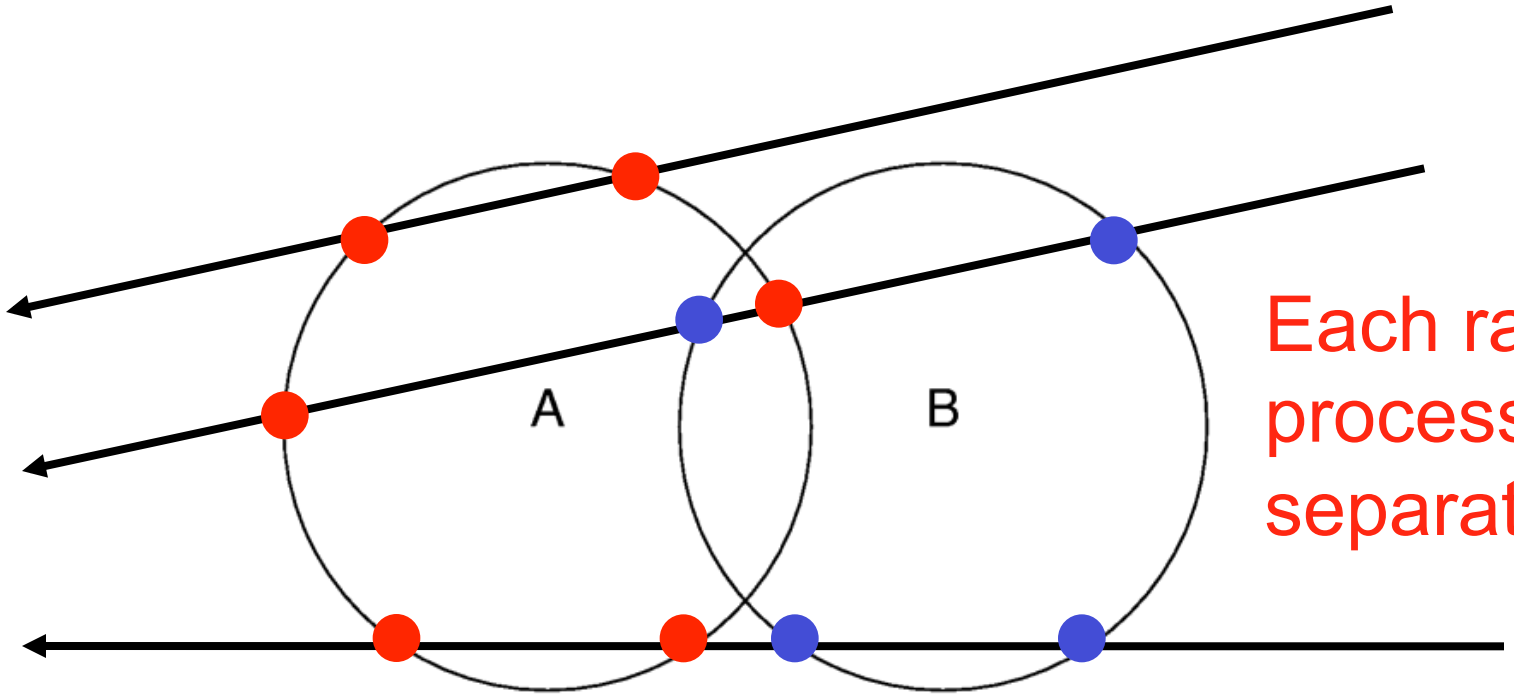
Intersection



Subtraction

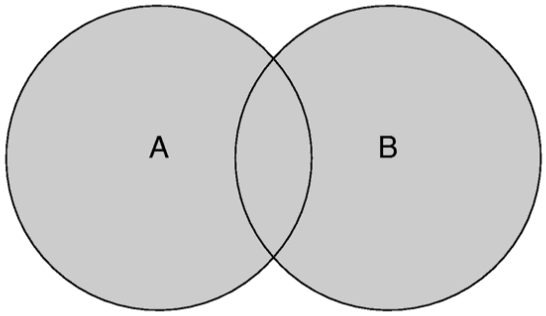


Collect Intersections

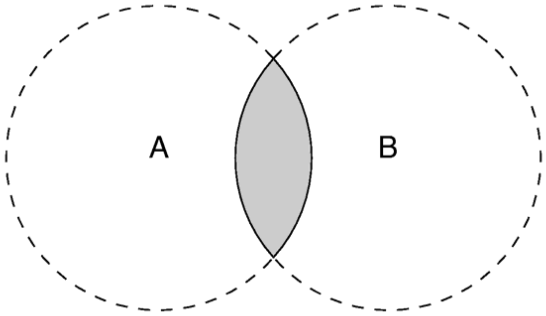


Each ray processed separately!

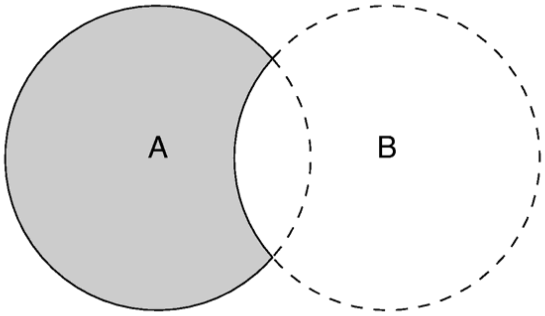
Union



Intersection

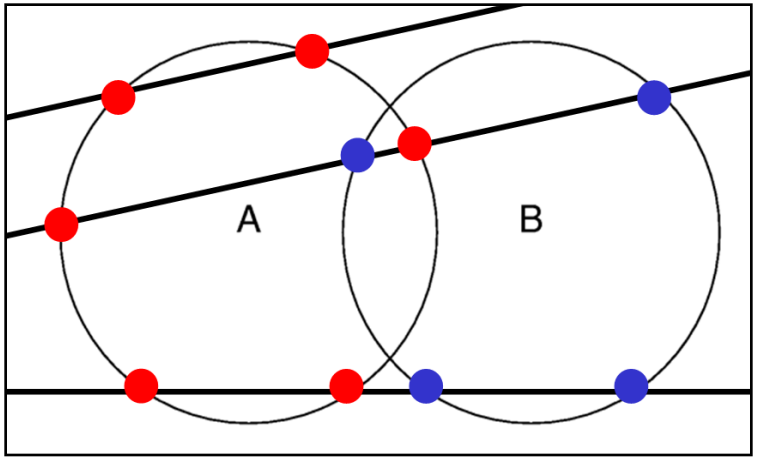


Subtraction



Implementing CSG

1. Test "inside" intersections:
   - Find intersections with A, test if they are inside/outside B
   - Find intersections with B, test if they are inside/outside A

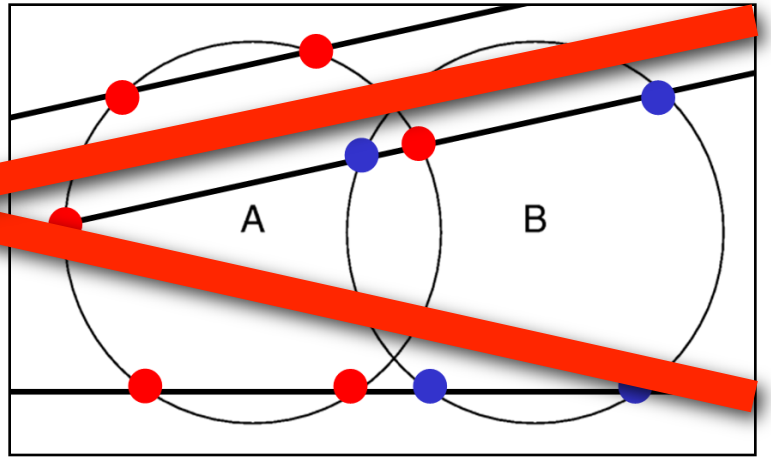


This would certainly work, but would need to determine if points are inside solids...

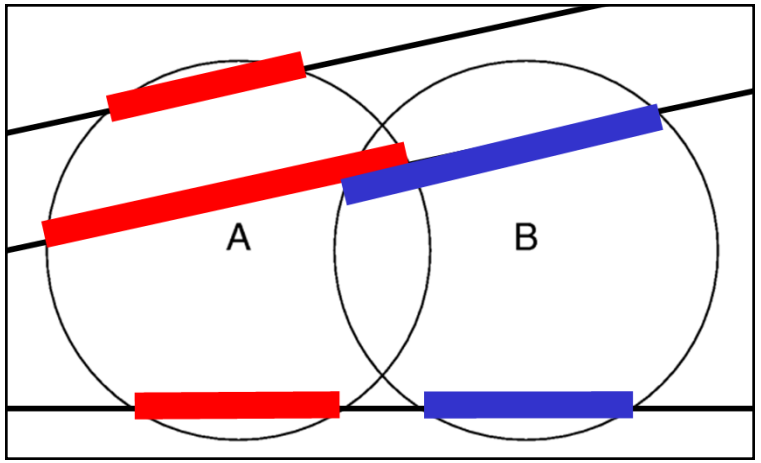
:-(

Implementing CSG

1. Test "inside" intersections:
   - Find intersections with A, test if they are inside/outside B
   - Find intersections with B, test if they are inside/outside A



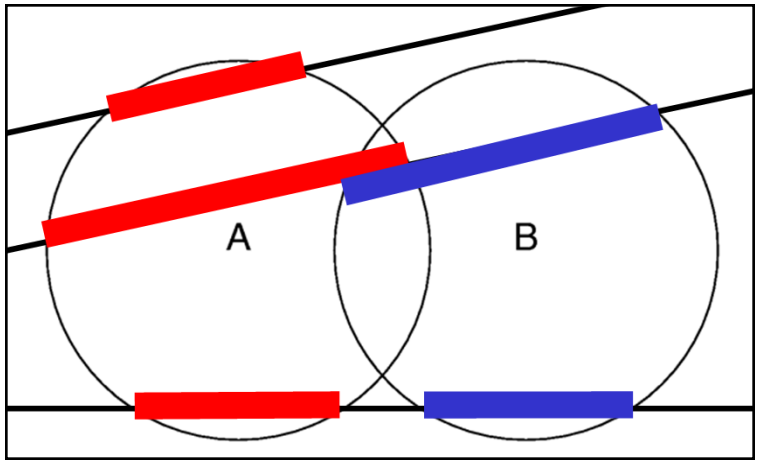
2. Overlapping intervals:
   - Find the intervals of "inside" along the ray for A and B
   - How? Just keep an “entry” / “exit” bit for each intersection
     - Easy to determine from intersection normal and ray direction
   - Compute union/intersection/ subtraction of the intervals



Implementing CSG

Problem reduces to 1D for each ray

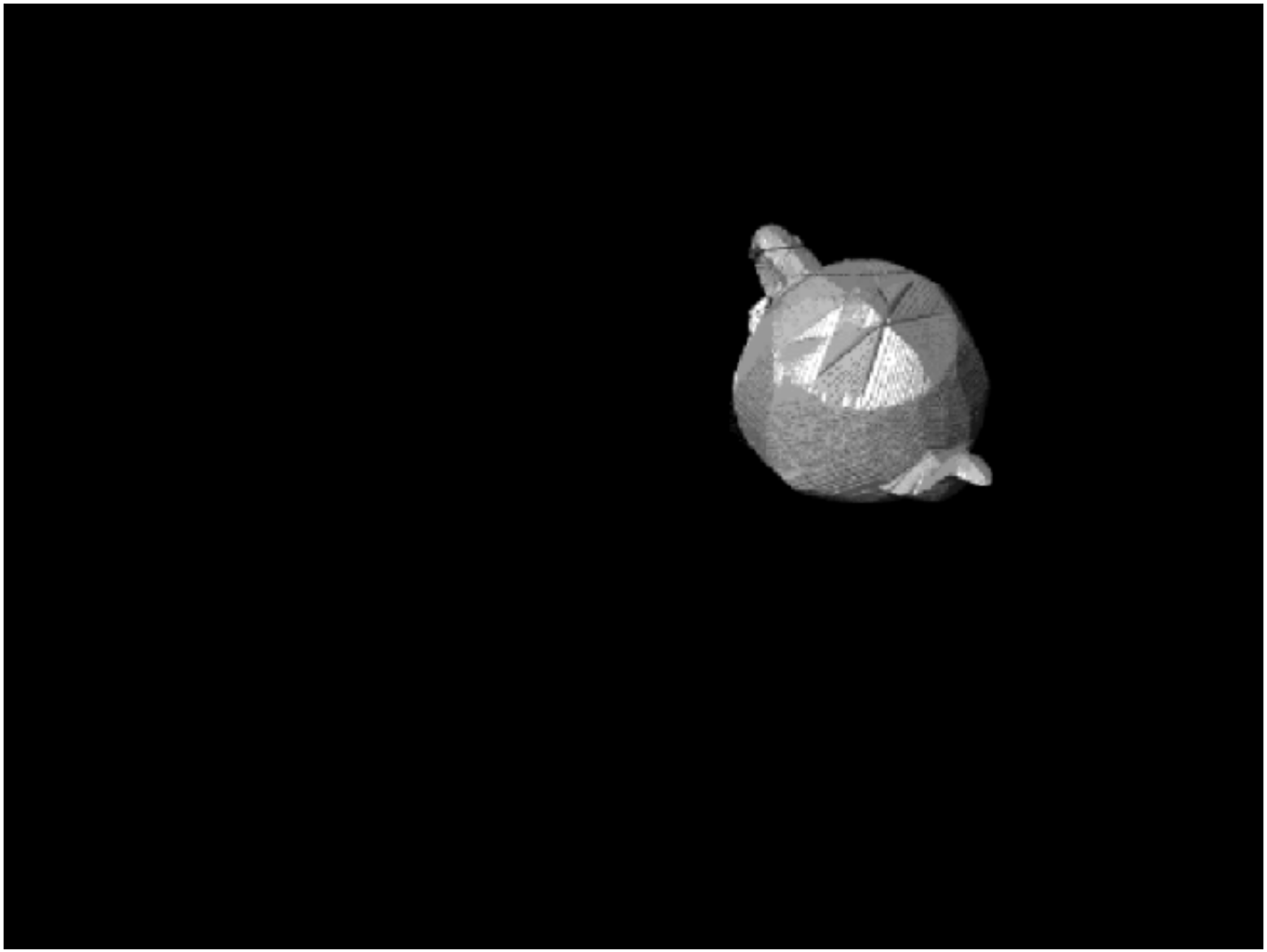
2. Overlapping intervals:
   - Find the intervals of "inside" along the ray for A and B
   - How? Just keep an “entry” / “exit” bit for each intersection
     - Easy to determine from intersection normal and ray direction
   - Compute union/intersection/ subtraction of the intervals



CSG is Easy with Ray Casting...

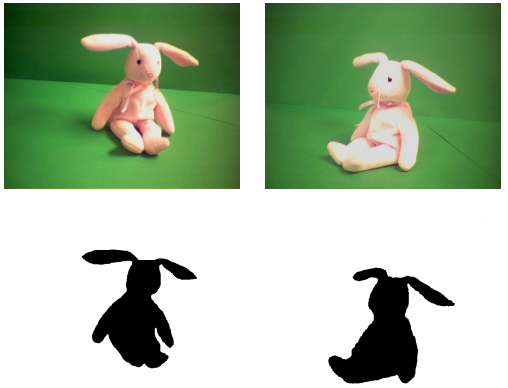
- ...but very hard if you actually try to compute an explicit 3D representation of the resulting surface as a triangle mesh
- In principle very simple,
  but floating point numbers are not exact
  - E.g., points do not lie exactly on planes...
  - Computing the intersection A vs B is not necessarily the same as B vs A...
  - The line that results from intersecting two planes does not necessarily lie on either plane...
  - etc., etc.

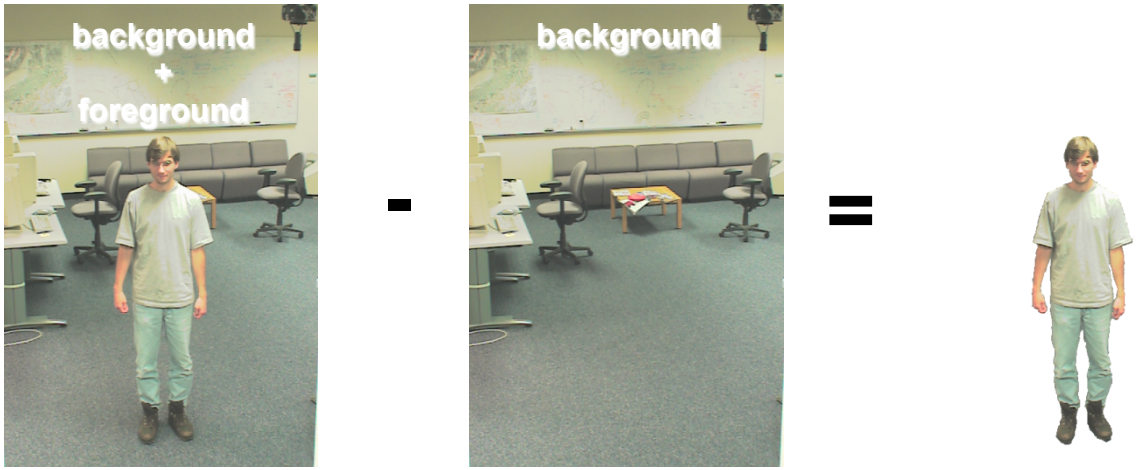
What is a Visual Hull?



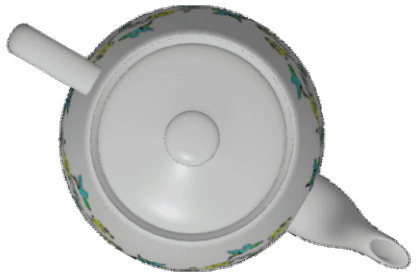
Why Use a Visual Hull?

- Can be computed robustly
- Can be computed efficiently

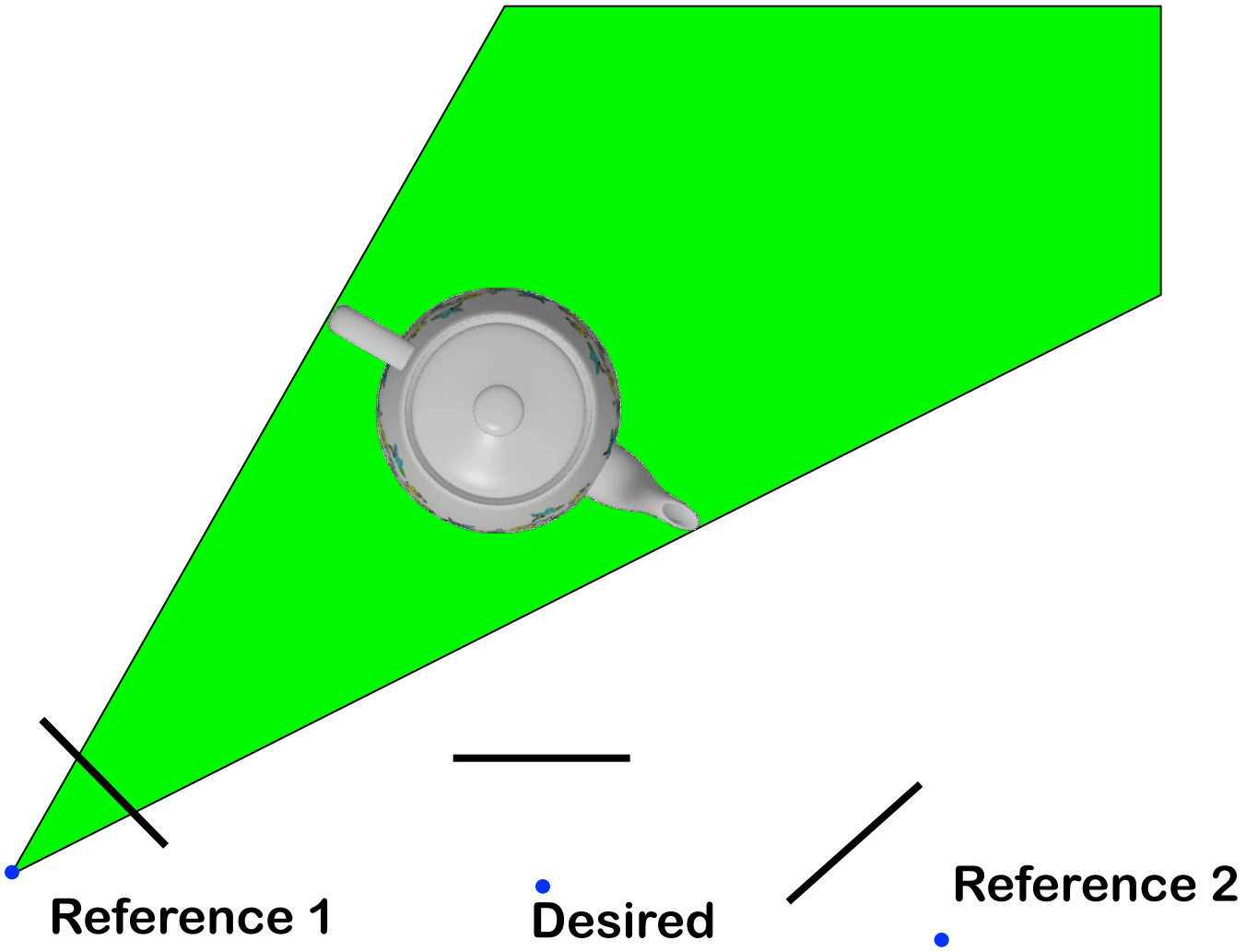




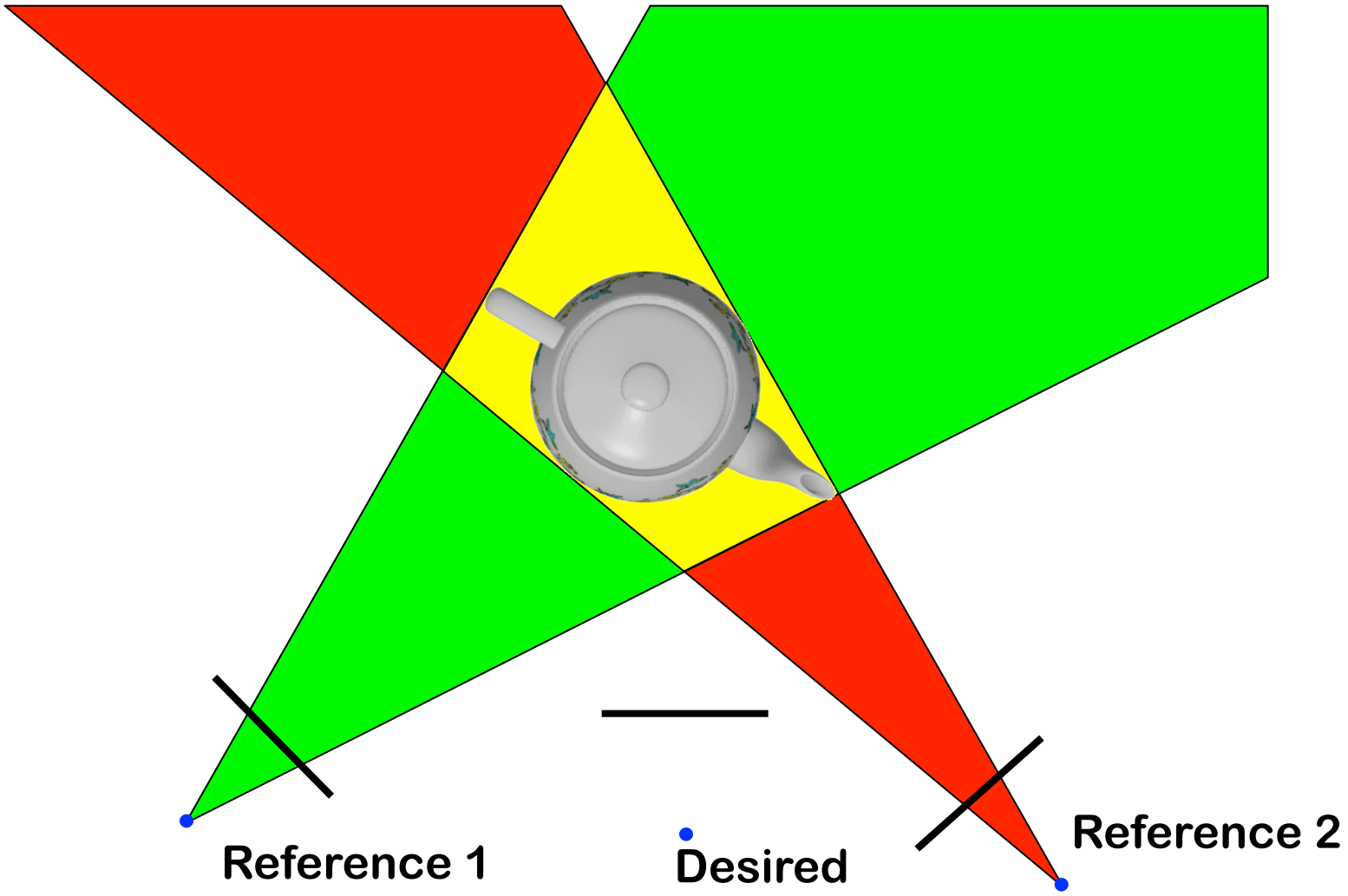
Rendering Visual Hulls



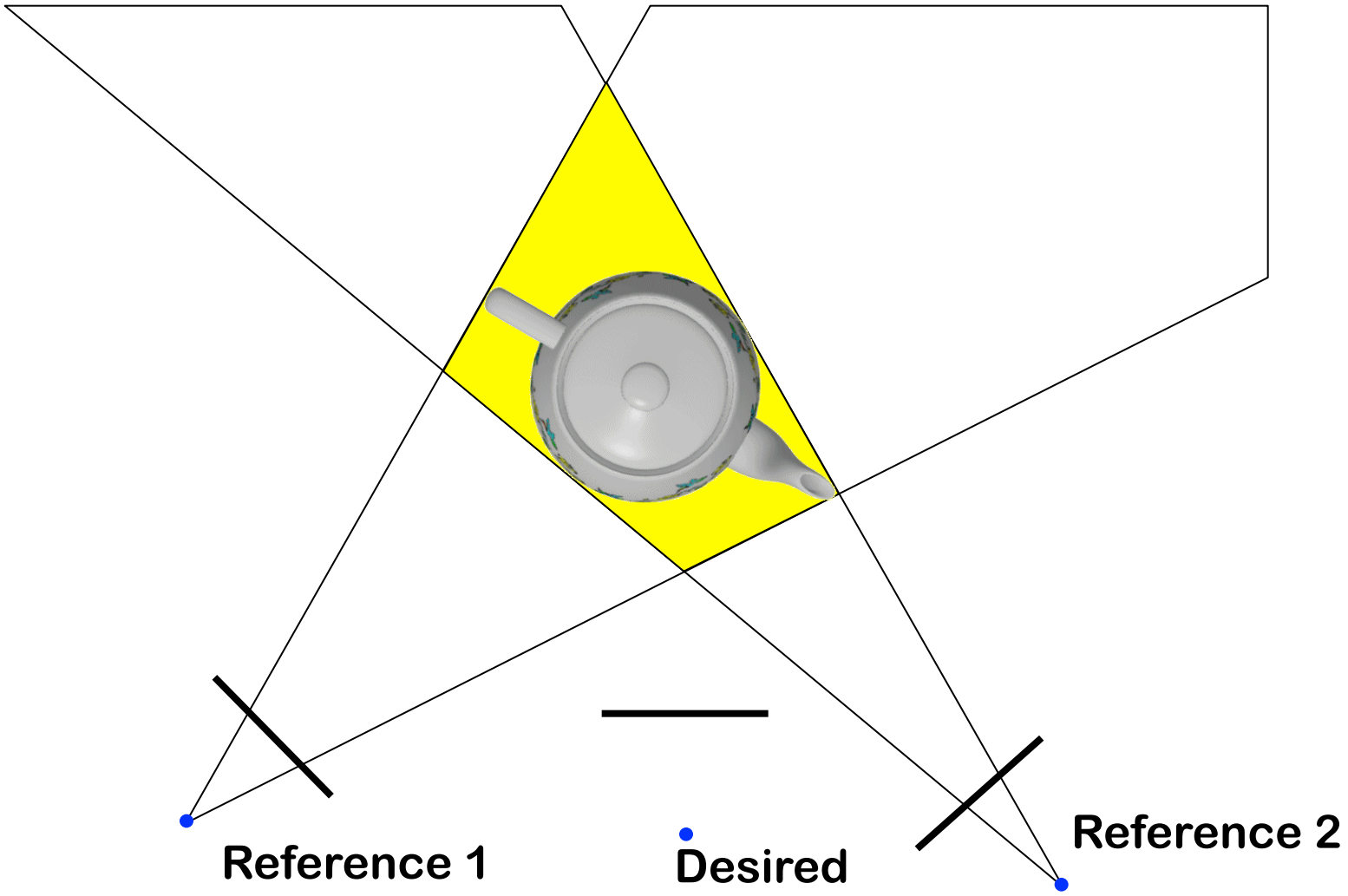
CSG then Ray Casting



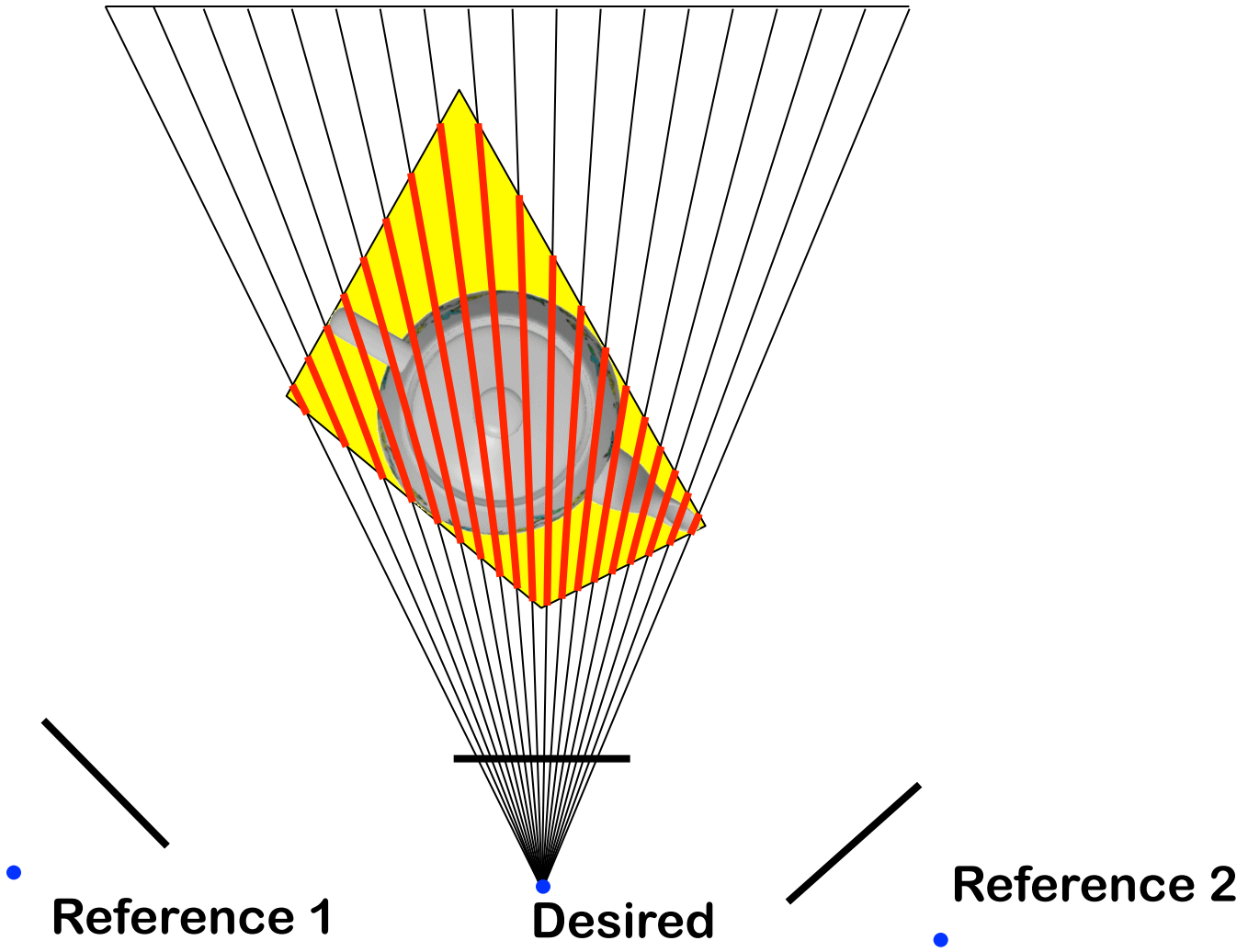
CSG then Ray Casting



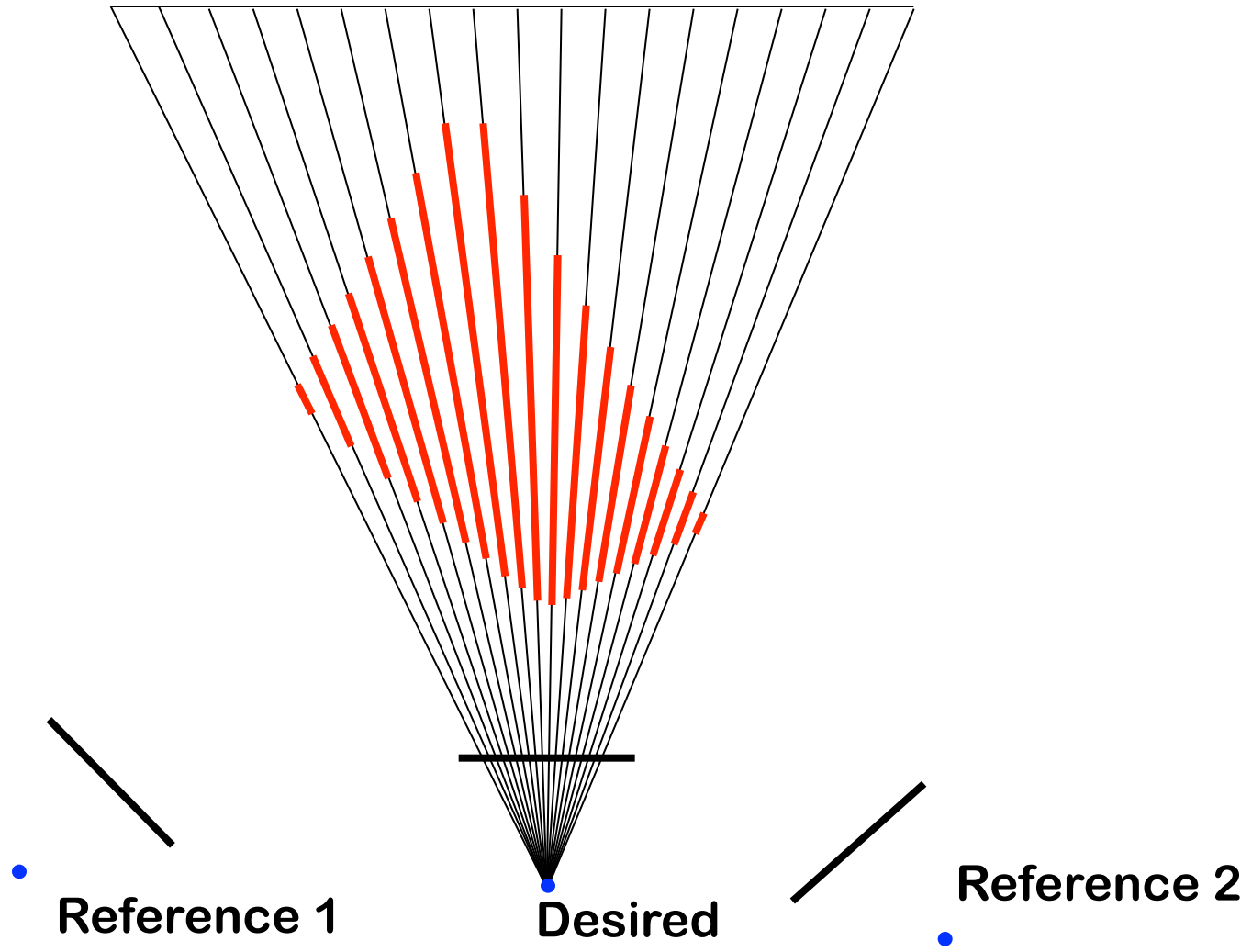
CSG then Ray Casting



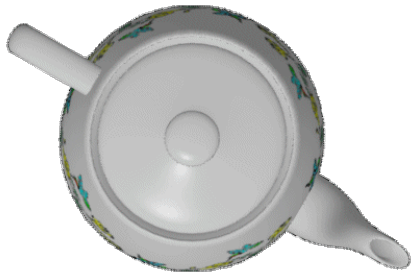
CSG then Ray Casting



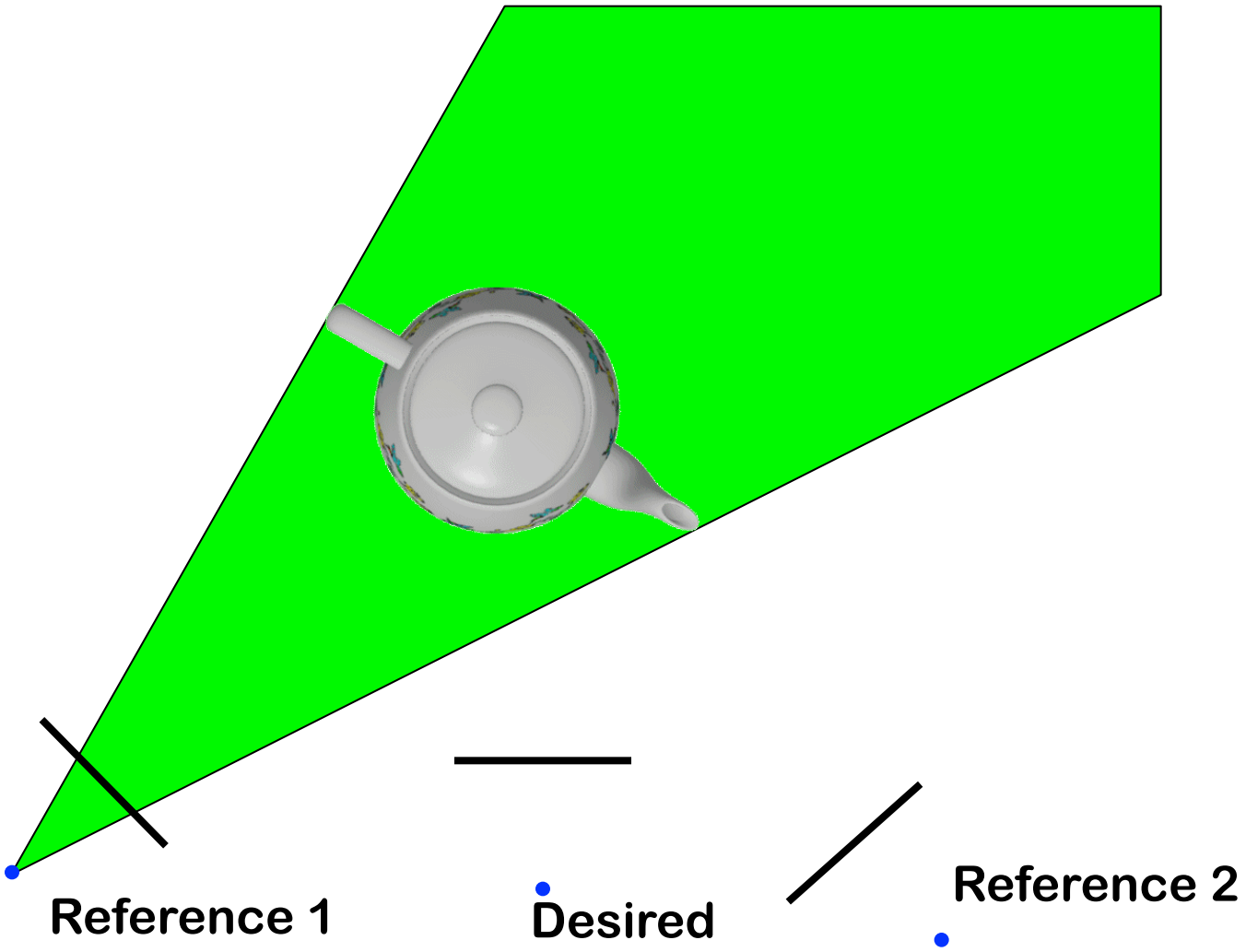
CSG then Ray Casting



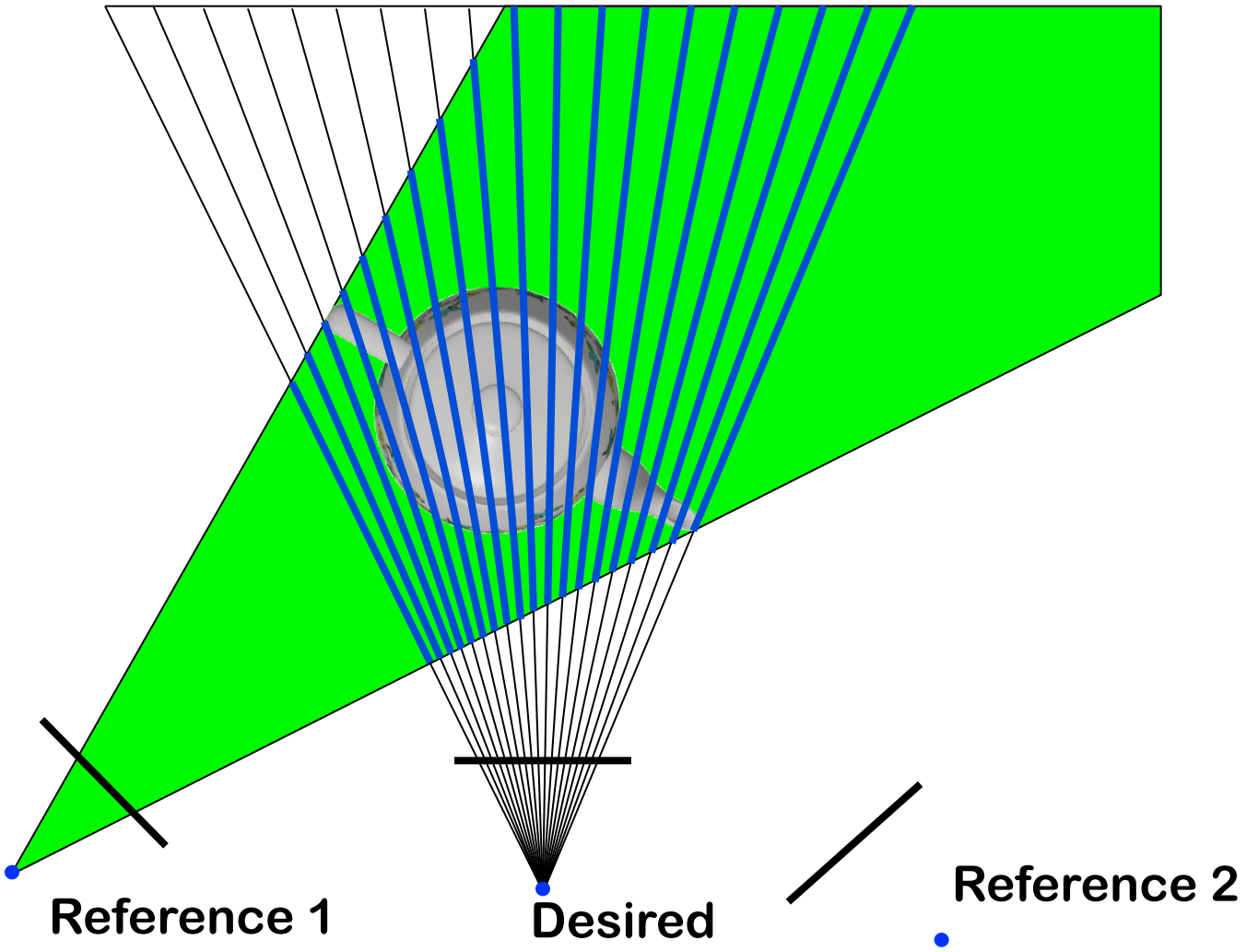
Ray Casting then Intersection



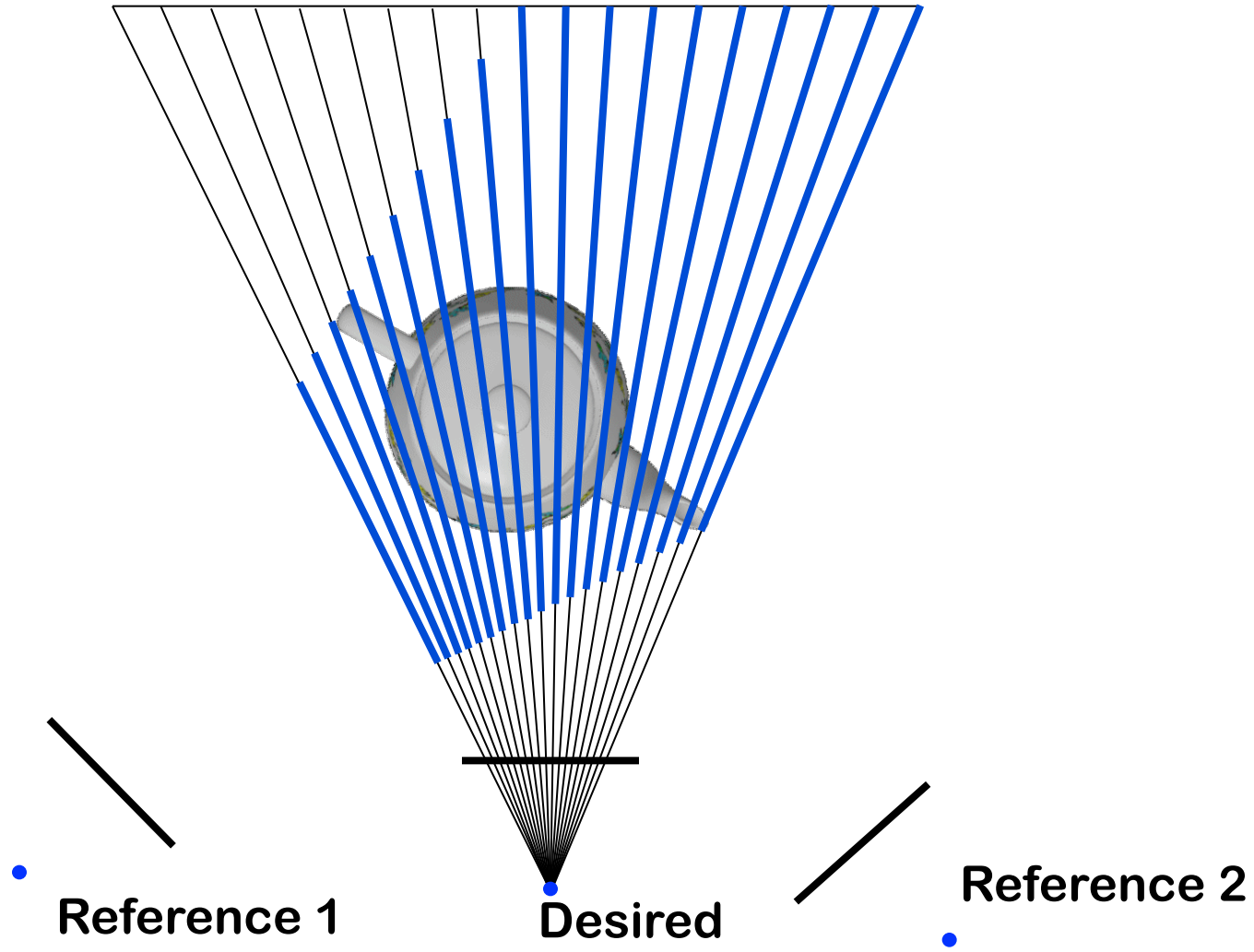
Ray Casting then Intersection



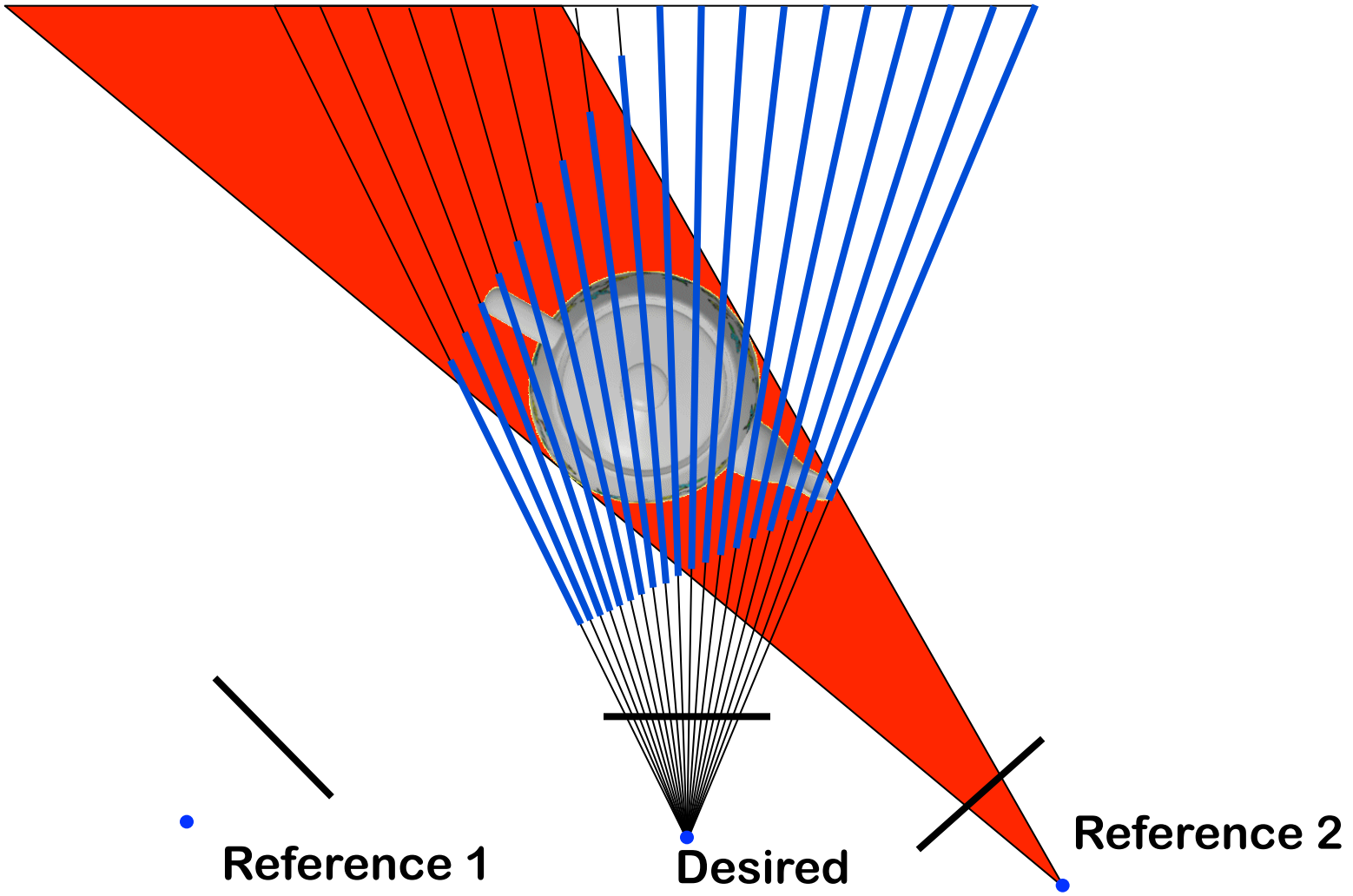
Ray Casting then Intersection



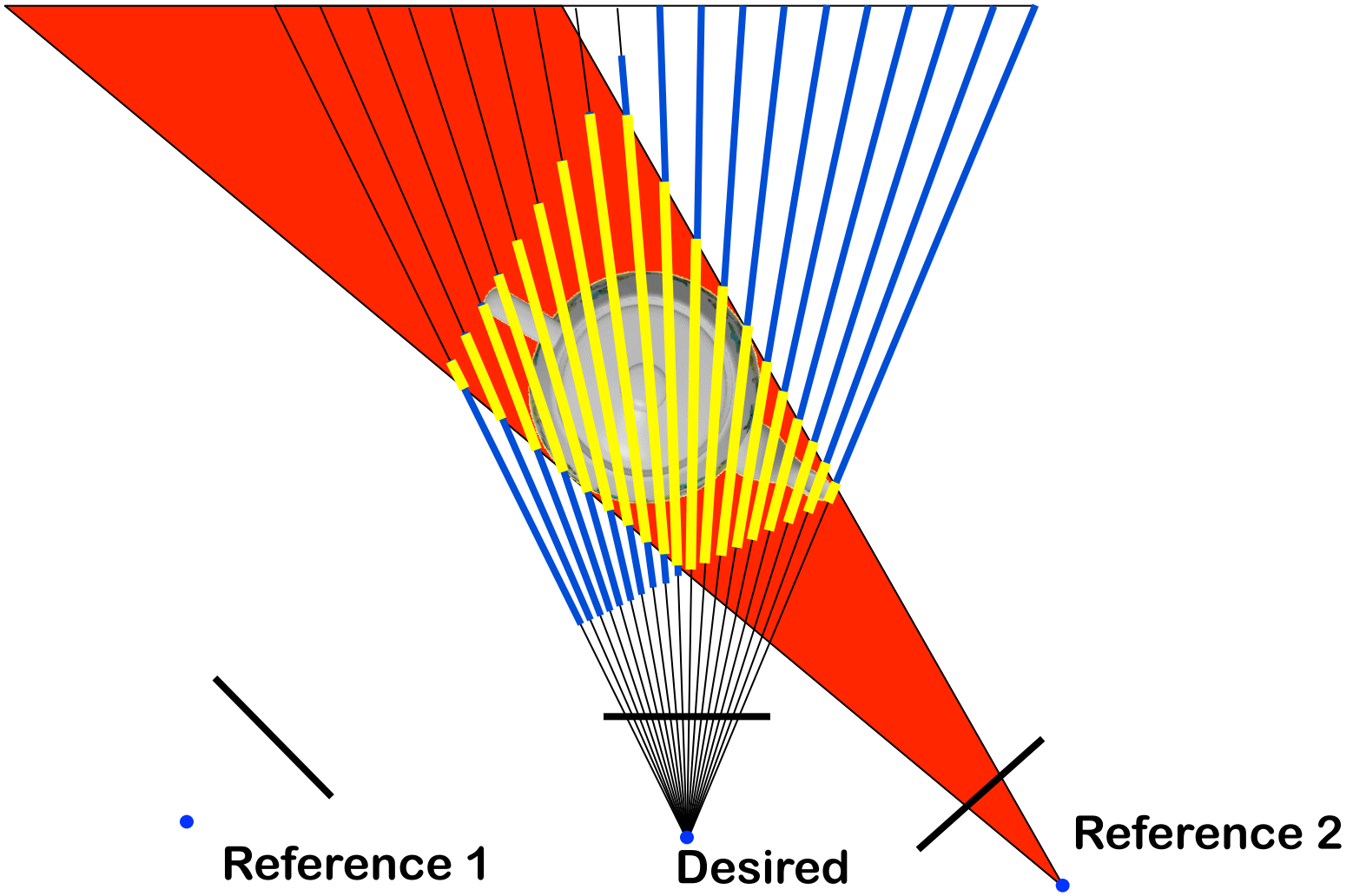
Ray Casting then Intersection



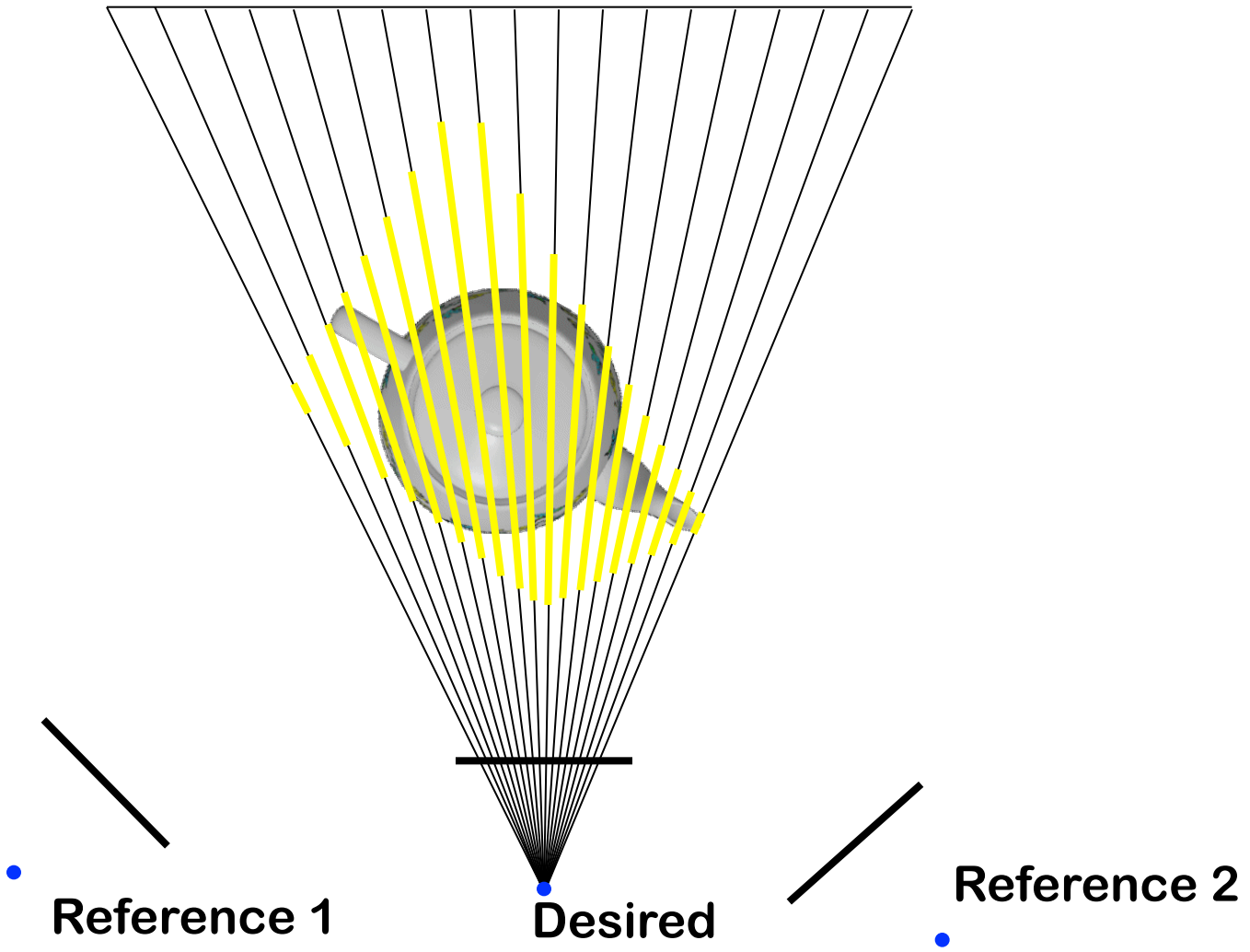
Ray Casting then Intersection



Ray Casting then Intersection



Ray Casting then Intersection



Ray Casting then Intersection

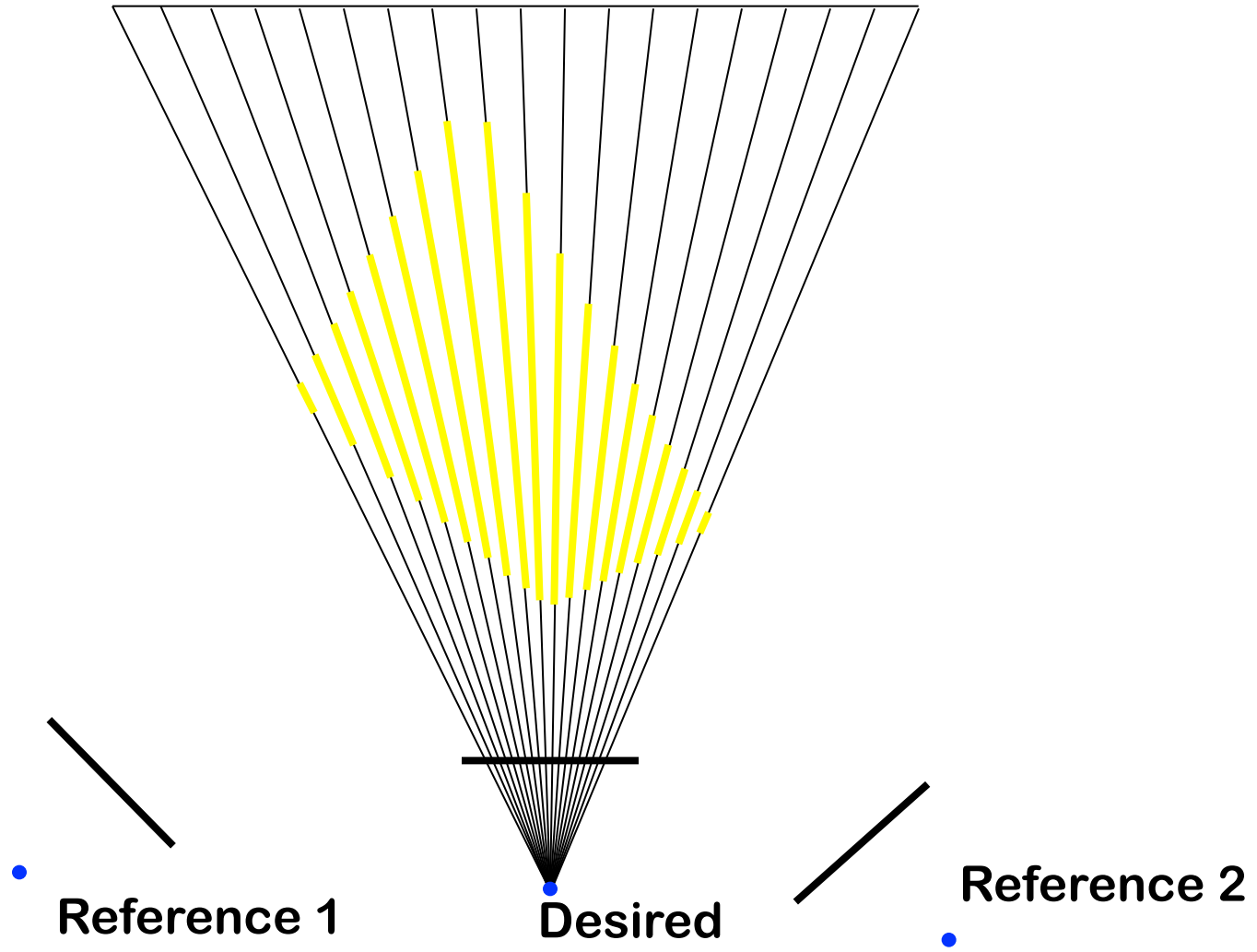
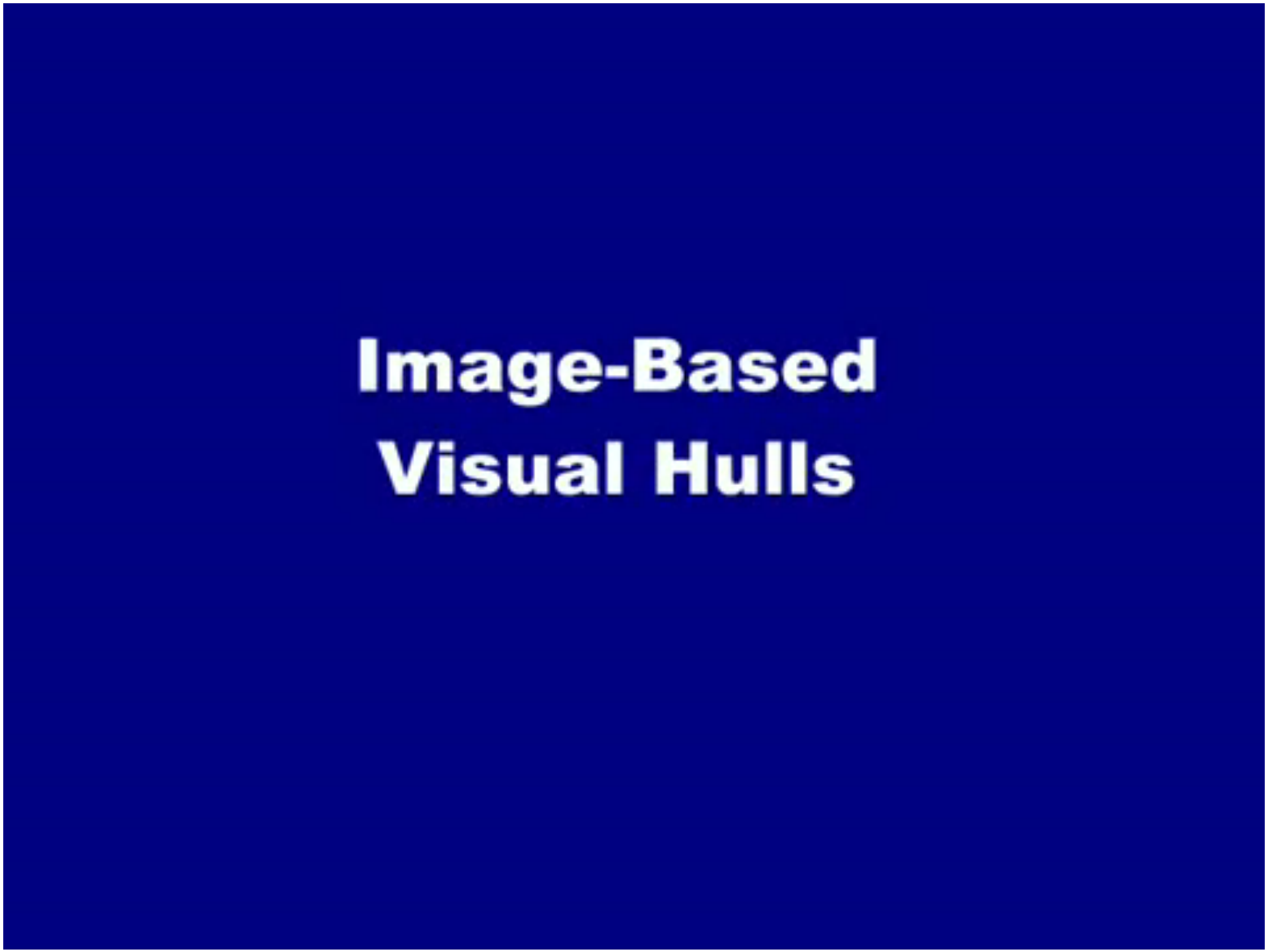
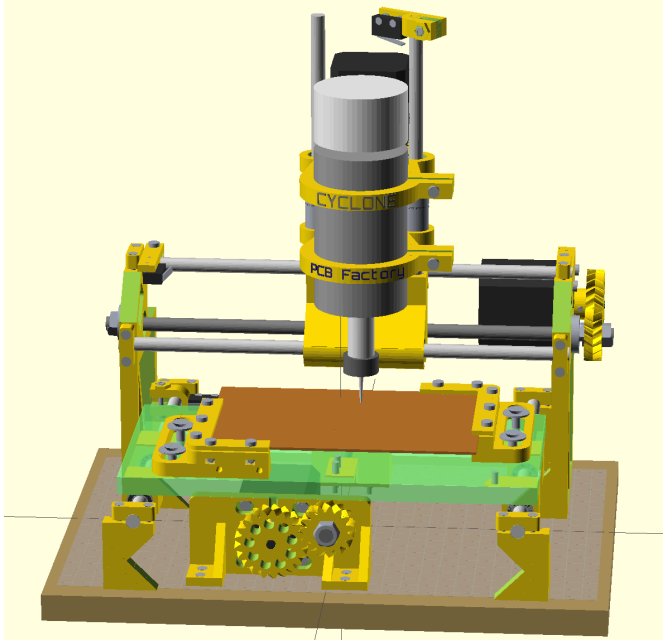


Image Based Visual Hulls



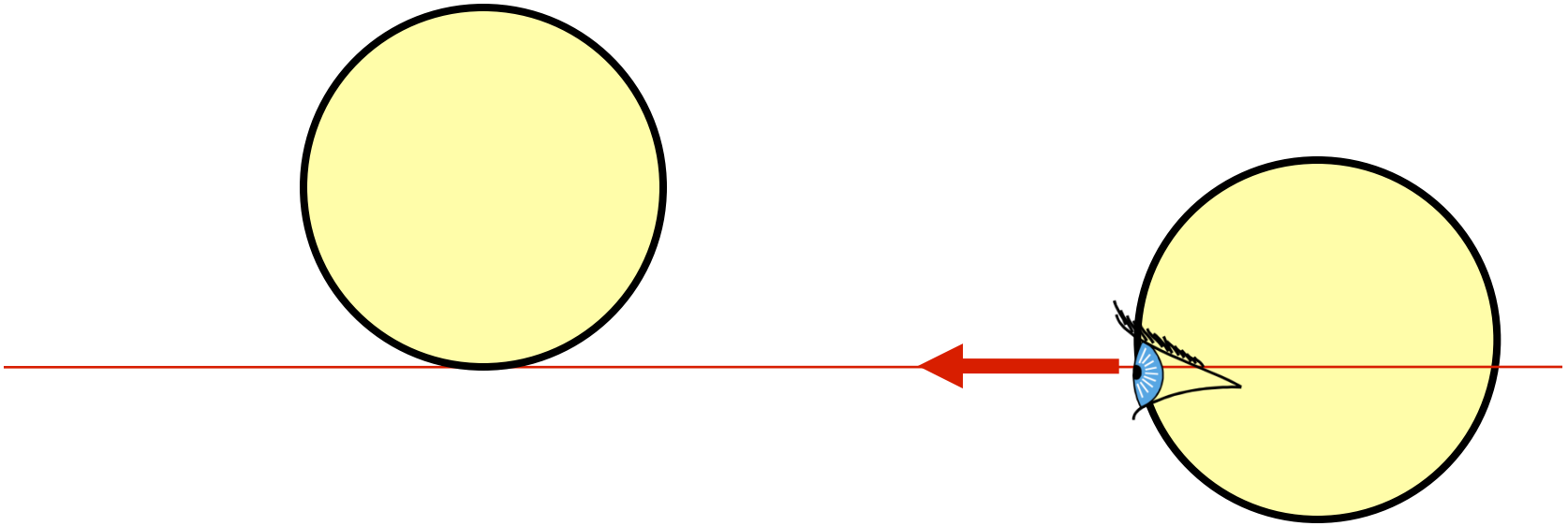
Questions?

- In Computational Fabrication, we will learn the exact operation
- Very simple when we use volumetric representation
- CSG is one of the most popular way in CAD for mechanical part



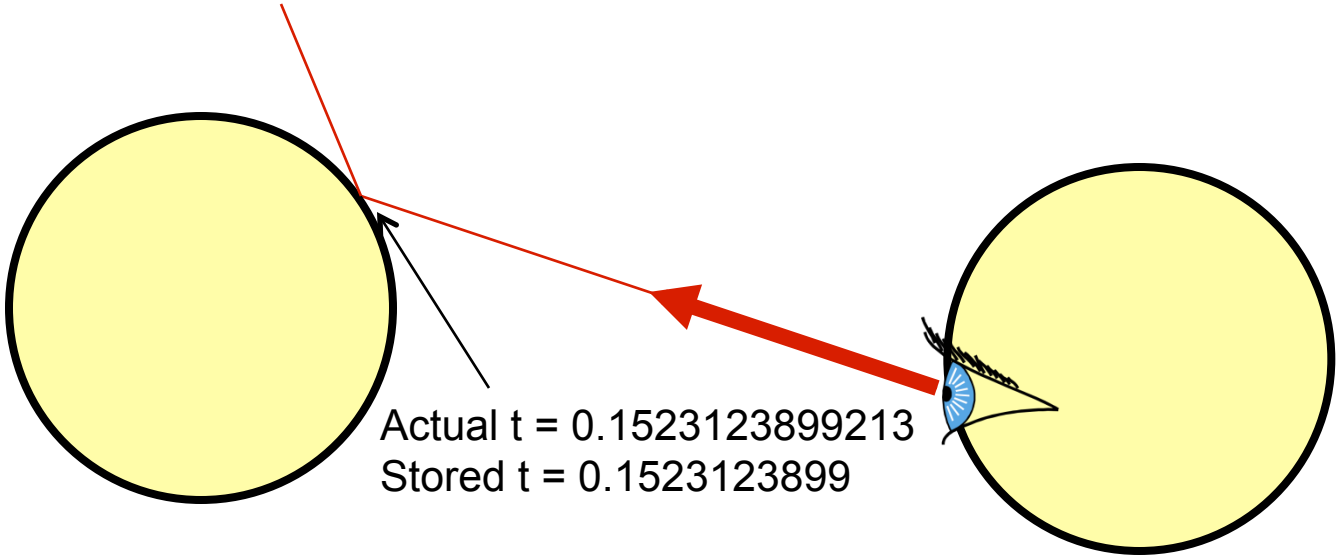
Precision

- What happens when
  - Ray Origin lies on an object?
  - Grazing rays?
- Problem with floating-point approximation



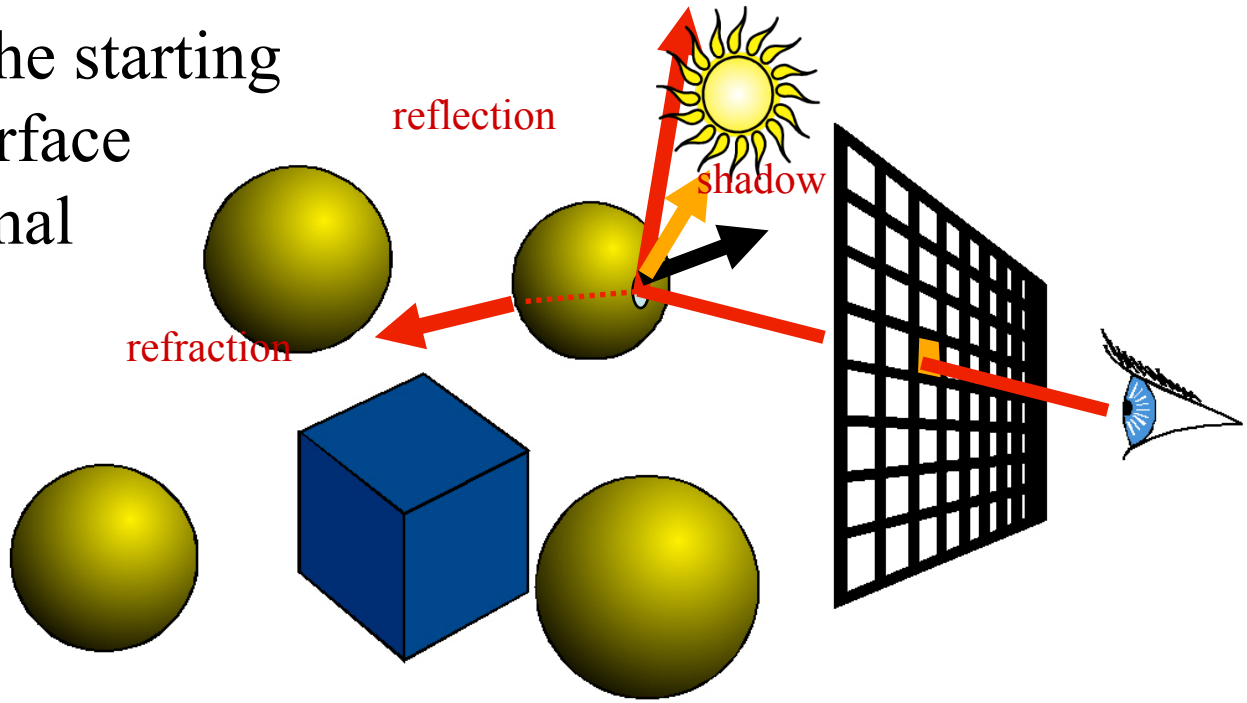
Precision

- What happens when
  - Ray Origin lies on an object?
  - Grazing rays?
- Problem with floating-point approximation



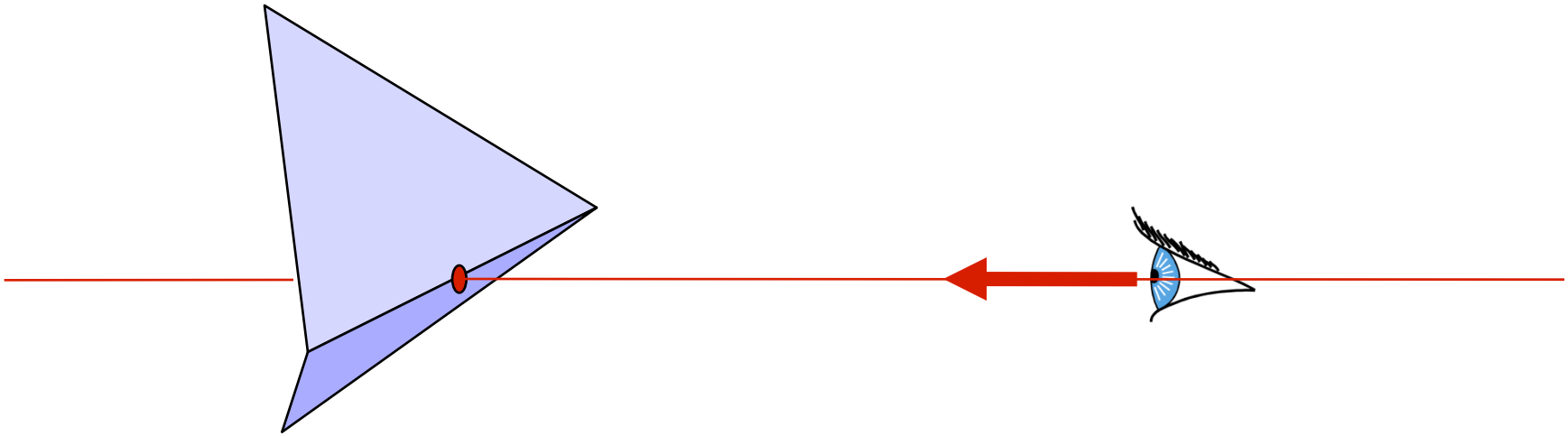
The Evil ε

- Floating-point roundoff can add up in a ray tracer, and create unwanted artifacts
  - Secondary rays start on surfaces: self-intersection!
  - Requires epsilons
  - Best to nudge the starting point off the surface e.g., along normal



The Evil ε

- Edges in triangle meshes
  - Must report intersection (otherwise not watertight)
  - Hard to get right



Questions?

Image by Henrik Wann Jensen

Transformations and Ray Casting

- We have seen that transformations such as affine transforms are useful for modeling & animation
- How do we incorporate them into ray casting?

Incorporating Transforms

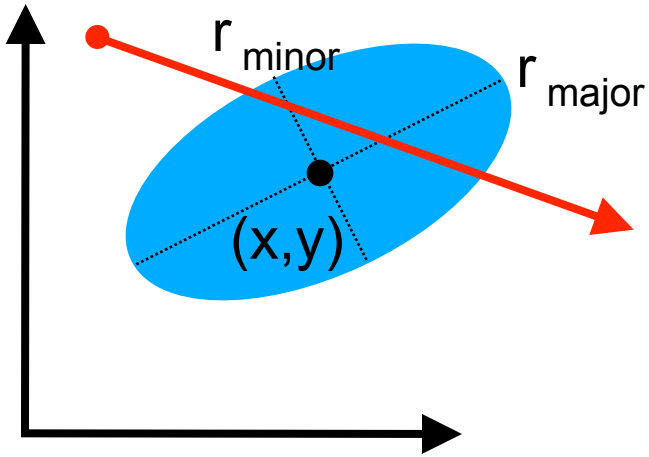
1. Make each primitive handle any applied transformations and produce a camera space description of its geometry

```
Transform {
    Translate { 1 0.5 0 }
    Scale { 2 2 2 }
    Sphere {
        center 0 0 0
        radius 1
    }
}
```

2. ...Or Transform the Rays

Primitives Handle Transforms

```
Sphere {
     center 3 2 0
     z_rotation 30
     r_major 2
     r_minor 1
}
```

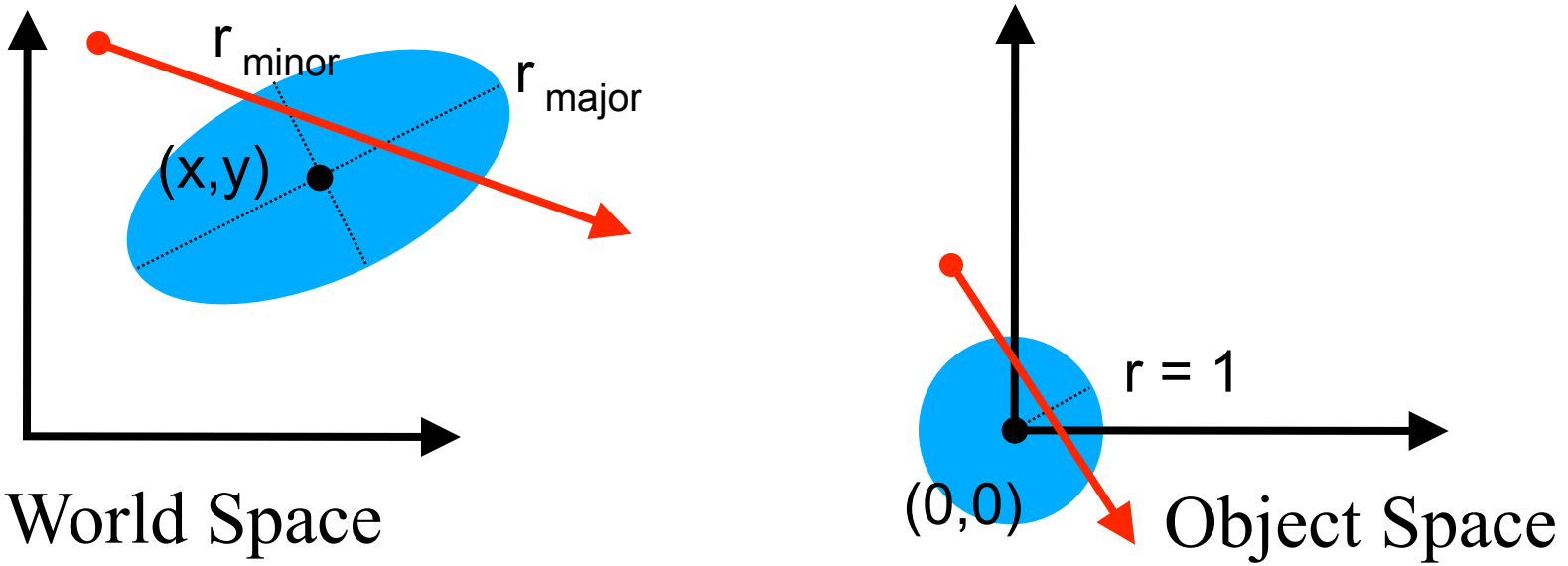


Form a Matrix M to transform primitive Sphere to world space

- Complicated for many primitives

Transform Ray

- Move the ray from World Space to Object Space



$$p_{WS} = M \; p_{OS}$$

$$p_{OS} = M^{-1} \; p_{WS}$$

Transform Ray

- New origin:

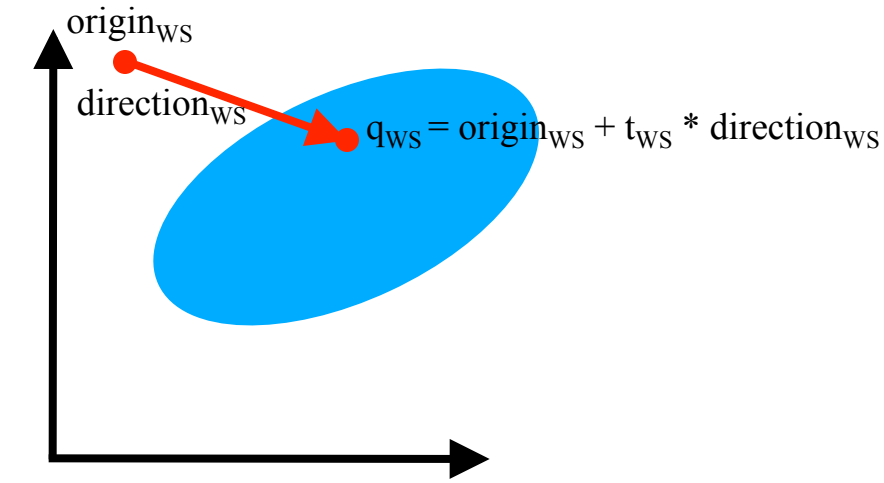
  $origin_{OS} = M^{-1} origin_{WS}$

- New direction:

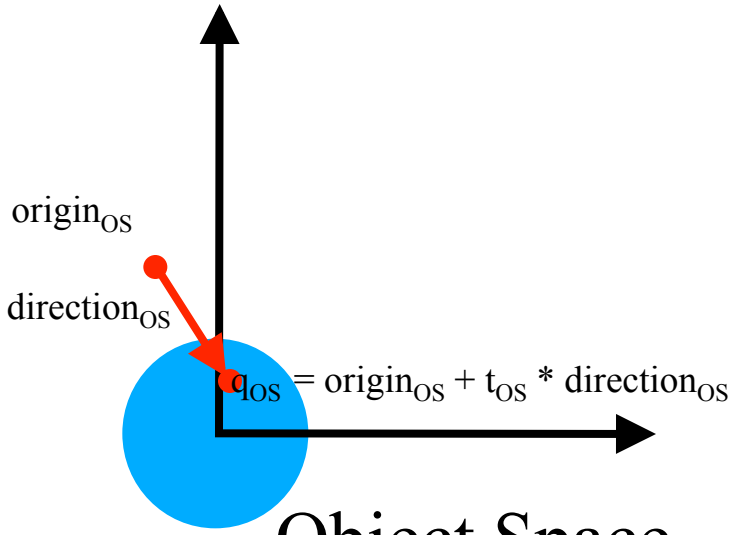
  $direction_{OS} = M^{-1} (origin_{WS} + 1 * direction_{WS}) - M^{-1} origin_{WS}$

  $direction_{OS} = M^{-1} direction_{WS}$

Note that the w component of direction is 0



World Space



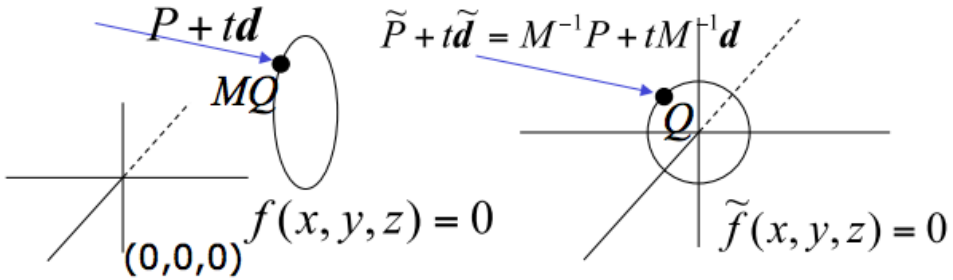
Object Space

What About t ?

- If M includes scaling, direction_{OS} ends up NOT be normalized after transformation

Object Space Intersection

Transform ray into object space



- Express world-space point of intersection as *MQ*, where Q is some point in object-space:

$$P + t\boldsymbol{d} = MQ$$

$$M^{-1} \cdot (P + t\boldsymbol{d}) = Q$$

$$M^{-1}P + tM^{-1}\boldsymbol{d} = Q$$

$$\text{Let} \quad \tilde{P} = M^{-1}P, \quad \tilde{\boldsymbol{d}} = M^{-1}\boldsymbol{d}$$

- If $\tilde{f}(x,y,z)$ is the equation of the untransformed object, we just have to solve

$$\tilde{f}(\tilde{P} + t\tilde{\boldsymbol{d}}) = 0$$

  - note $\tilde{d}$ is probably not a unit vector
  - the parameter *t* along this vector and its world space counterpart always have the same value. Normalizing $\tilde{d}$ would alter this relationship.

Do NOT normalize d .

What About t ?

- If M includes scaling, direction_{OS} ends up NOT be normalized after transformation

- Just directly get the t!

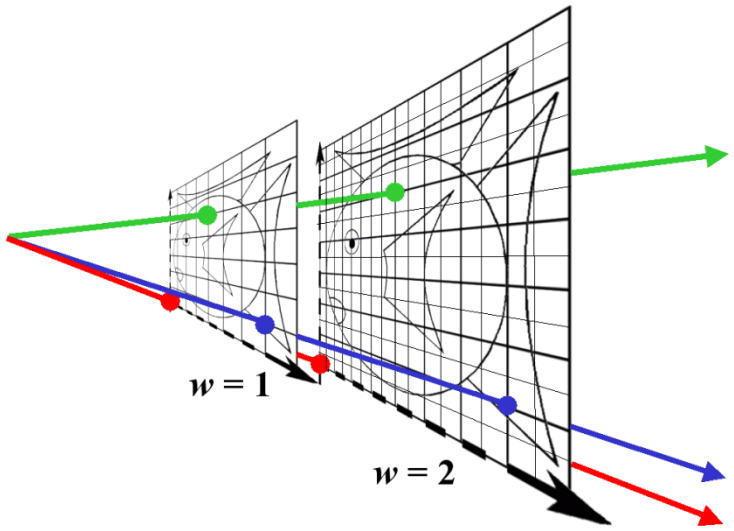
Transforming Points & Directions

- Transform point

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} ax+by+cz+d \\ ex+fy+gz+h \\ ix+jy+kz+l \\ 1 \end{pmatrix}$$

- Transform direction

$$\begin{pmatrix} x' \\ y' \\ z' \\ 0 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix} = \begin{pmatrix} ax+by+cz \\ ex+fy+gz \\ ix+jy+kz \\ 0 \end{pmatrix}$$

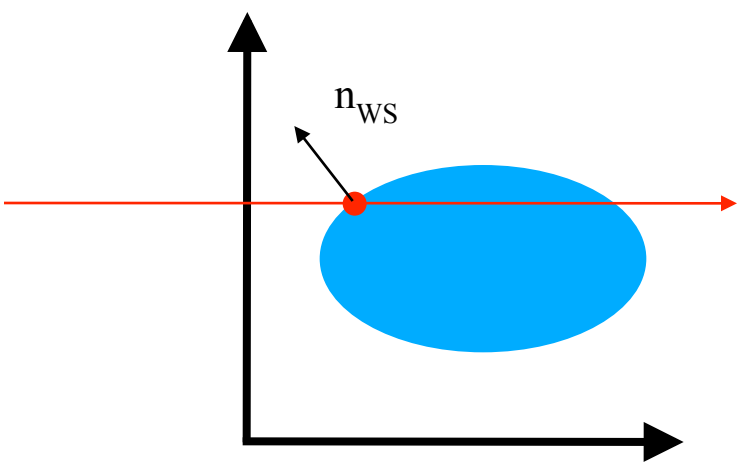


Homogeneous Coordinates: (x,y,z,w)

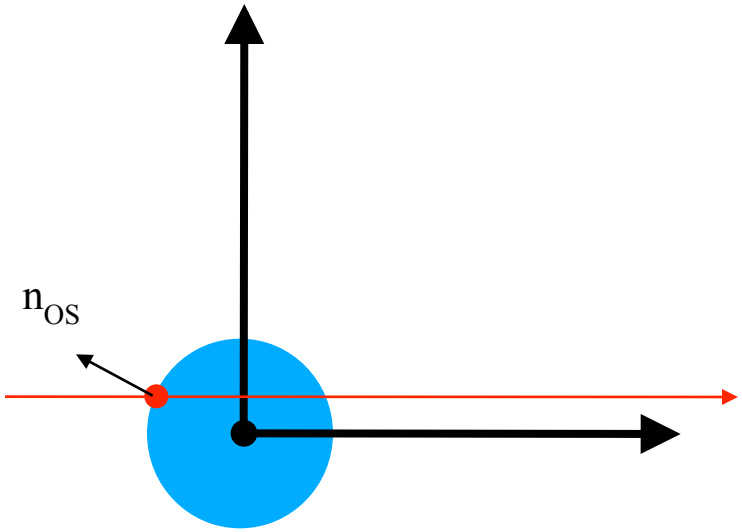
$w = 0$ is a point at infinity (direction)

- If you do not store w you need different routines to apply M to a point and to a direction ==> Store everything in 4D!

Recap: How to Transform Normals?



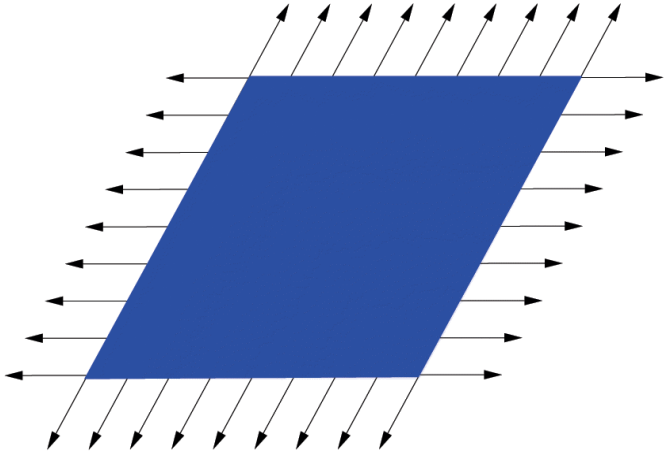
World Space

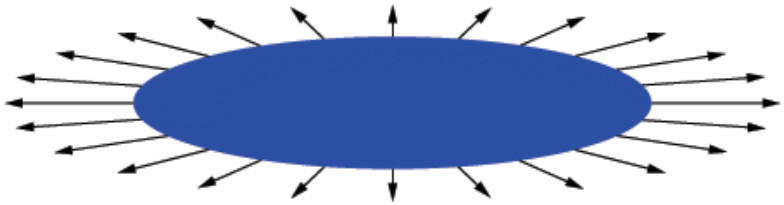


Object Space

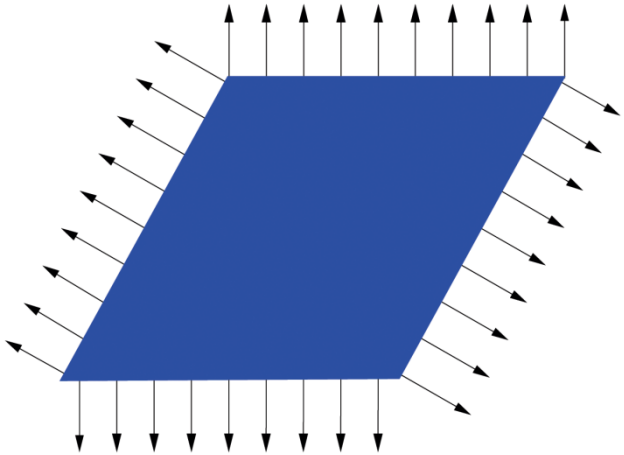
Transformation for Shear and Scale

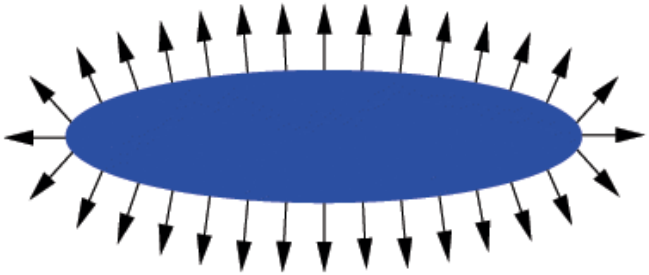
Incorrect
Normal
Transformation





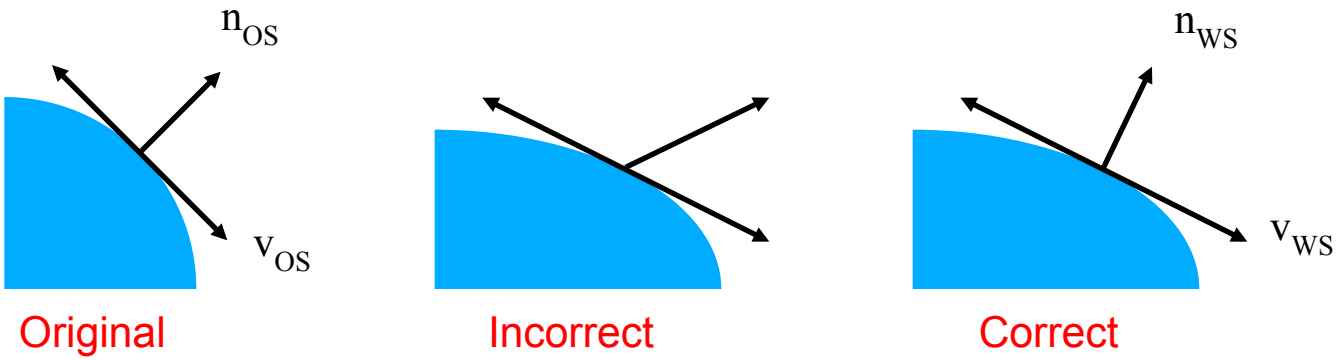
Correct
Normal
Transformation





So How Do We Do It Right?

- Think about transforming the tangent plane to the normal, not the normal vector



Pick any vector v_{OS} in the tangent plane, how is it transformed by matrix M?

$$v_{WS} = M\ v_{OS}$$

Transform Tangent Vector v

v is perpendicular to normal n:

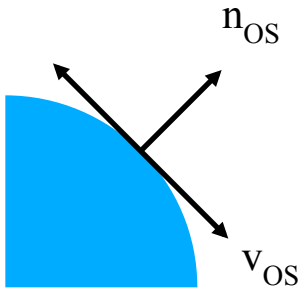
Dot product

$$n_{OS}^T v_{OS} = 0$$

$$n_{OS}^T (M^{-1} M) v_{OS} = 0$$

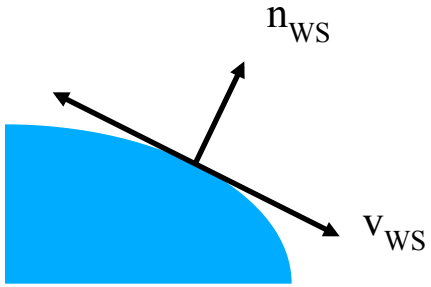
$$(n_{OS}^T M^{-1}) (M v_{OS}) = 0$$

$$(n_{OS}^T M^{-1}) v_{WS} = 0$$



v_{WS} is perpendicular to normal n_{WS}:

$$n_{WS}^T v_{WS} = 0$$



$$n_{WS}^T = n_{OS}^T (M^{-1})$$

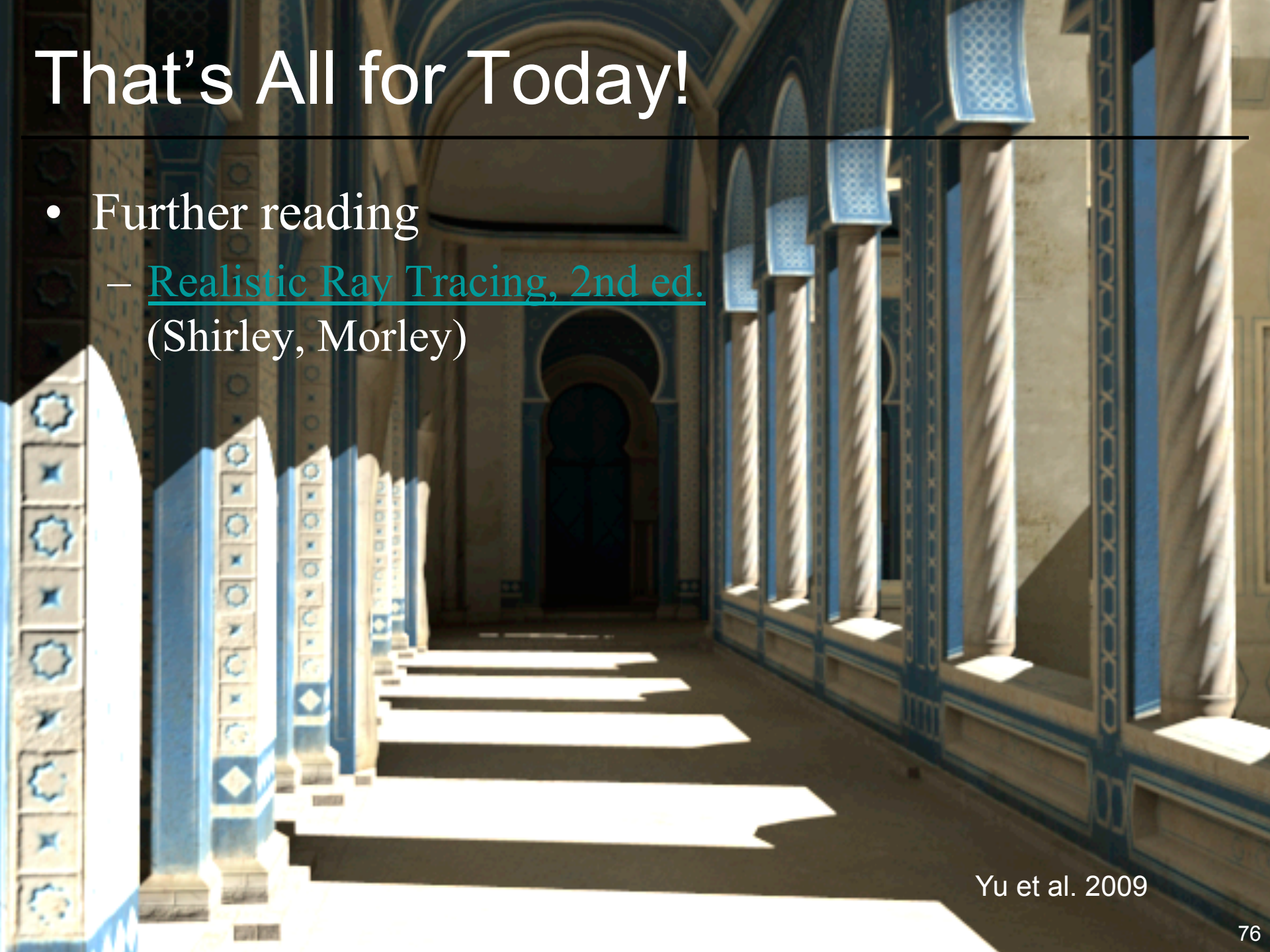
$$\boxed{n_{WS} = (M^{-1})^T n_{OS}}$$

Position, Direction, Normal

- Position
  - transformed by the full homogeneous matrix M
- Direction
  - transformed by M except the translation component
- Normal
  - transformed by M^{-T}, no translation component

That's All for Today!

- Further reading
  - Realistic Ray Tracing, 2nd ed. (Shirley, Morley)



Yu et al. 2009