13 – raytracing (1)

3 approaches to graphics

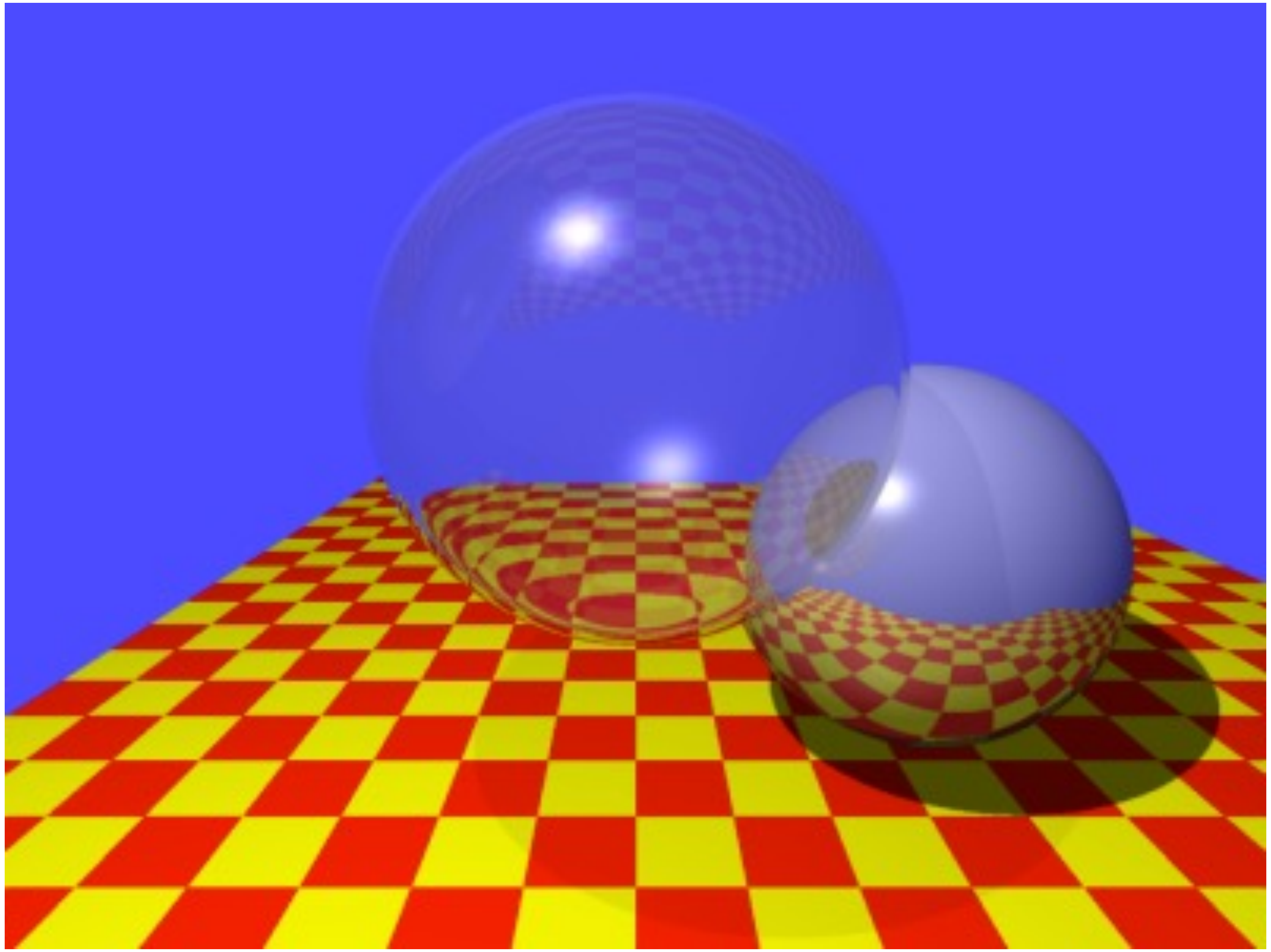
- On-line / "real-time" WebGL in general
  - Immediate mode

    Raw WebGL calls
  - Retained mode / Scene Graphs/etc
- Off-line / batch / "slow"

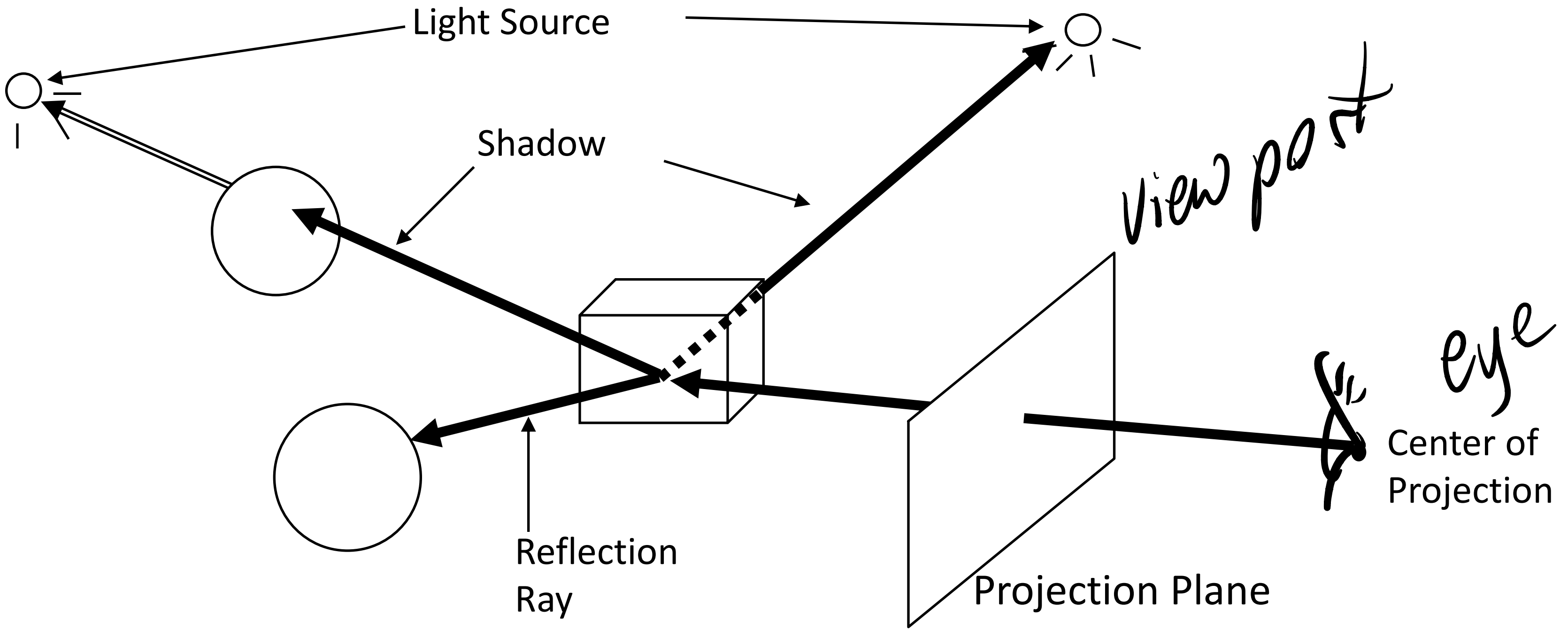
Ray Tracing

1980s a way to deal with shadows
& hidden
surfaces

Whitted



Basic Idea

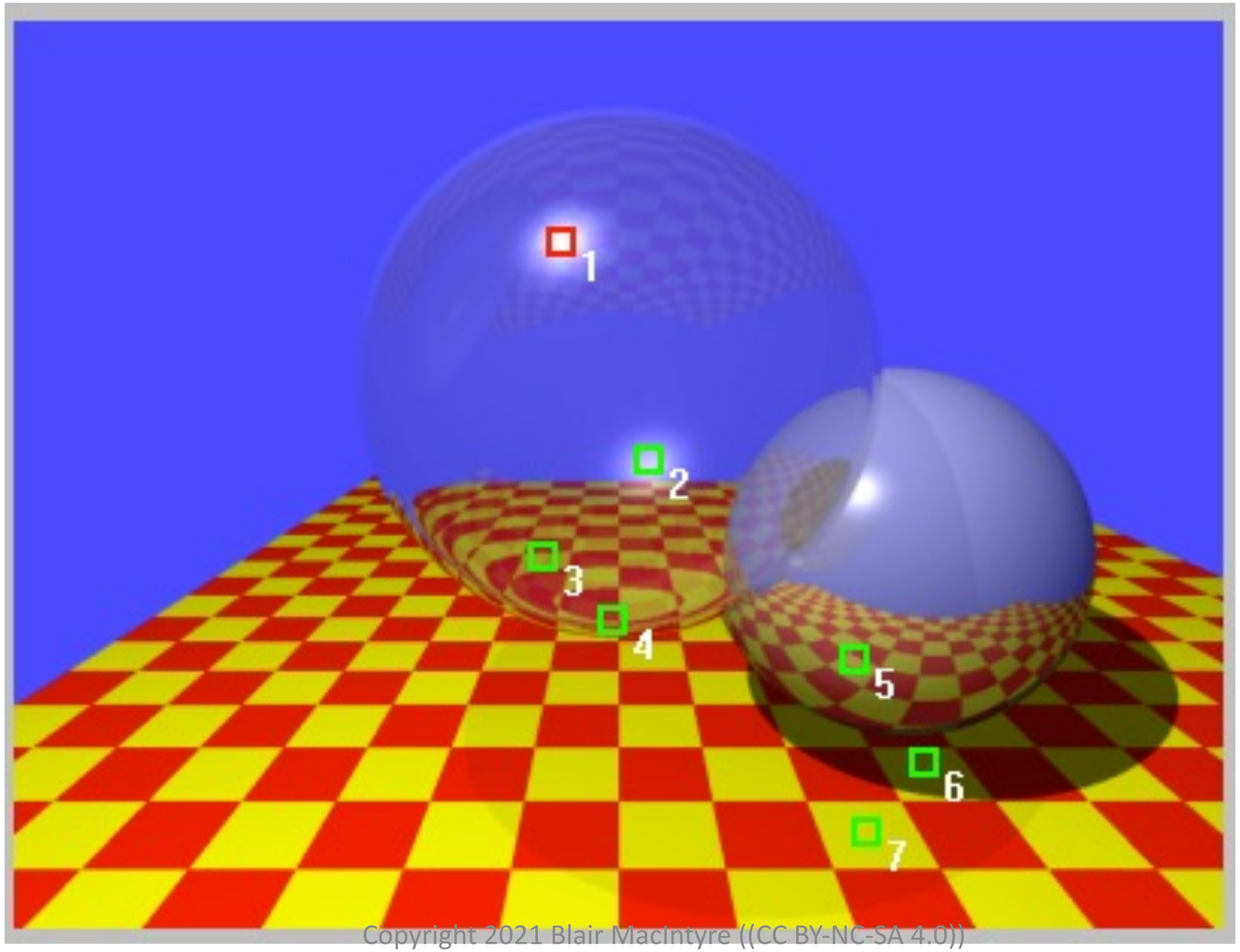

Light Source
Shadow
view port
eye
Center of
Projection
Reflection
Ray
Projection Plane

Basic Algorithm

for each pixel (x_s, y_s)
create a Ray R from eye through (x_s, y_s)
for each object O_i in scene
if R intersects O_i & it's the closest so far
record this intersection
shade pixel based on nearest intersection
(recursively for ref & transmition)

from *Alan Watt, 3D Computer Graphics, 3rd Edition, 1999*

The Adventures of 7 Rays

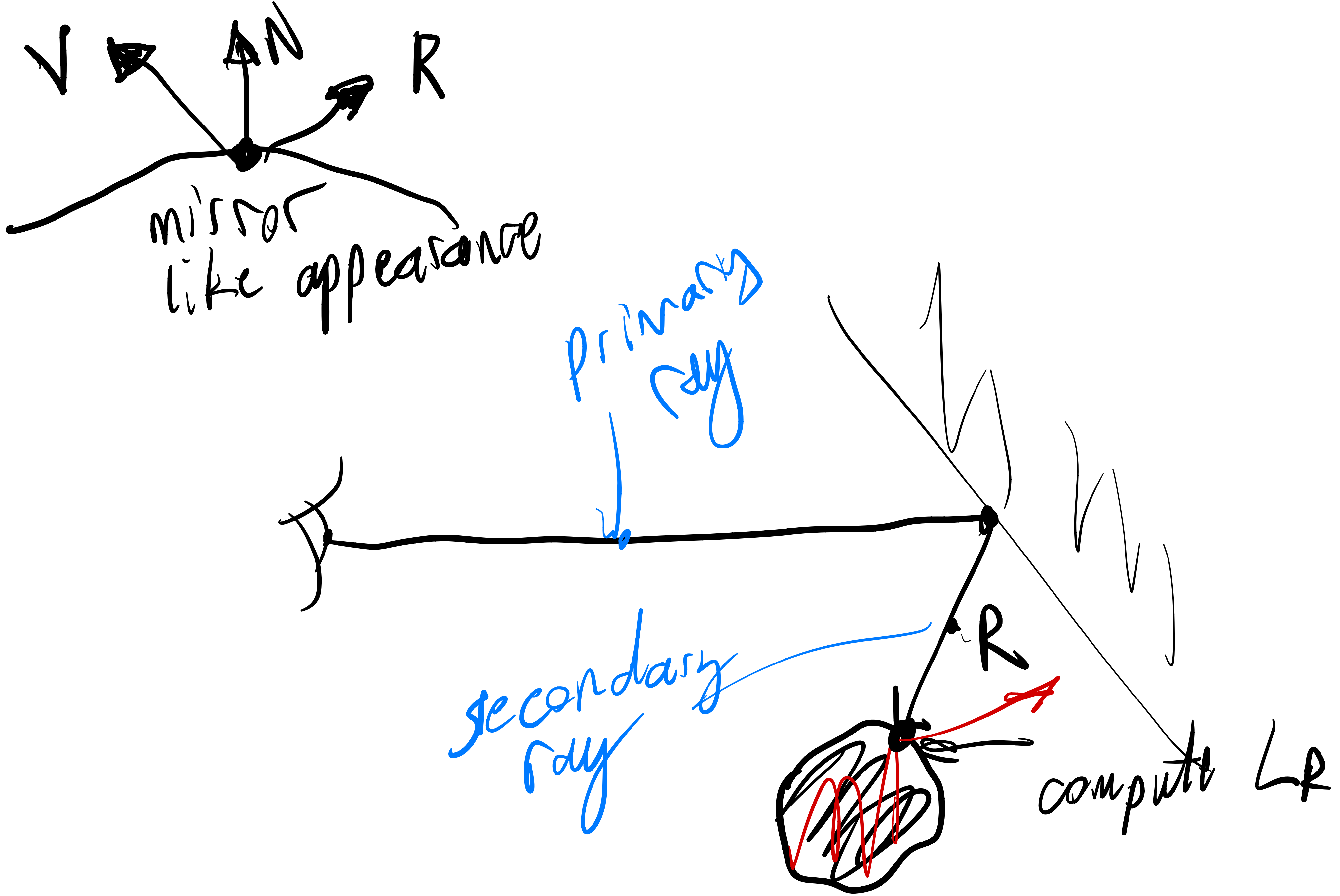


Basic Algorithm

Illumination of a point

$$L = \overbrace{k_a I_a}^{\text{ambient}} + \underbrace{k_s L_r}_{\text{reflected}} + \underbrace{k_t L_t}_{\text{refracted}} + \sum_{1 \le i \le N} S_i I_i \overbrace{\left[\, k_d (N \cdot L_i) + k_s (R_i \cdot V)^{p_i} \,\right]}^{\text{previous eq}}$$

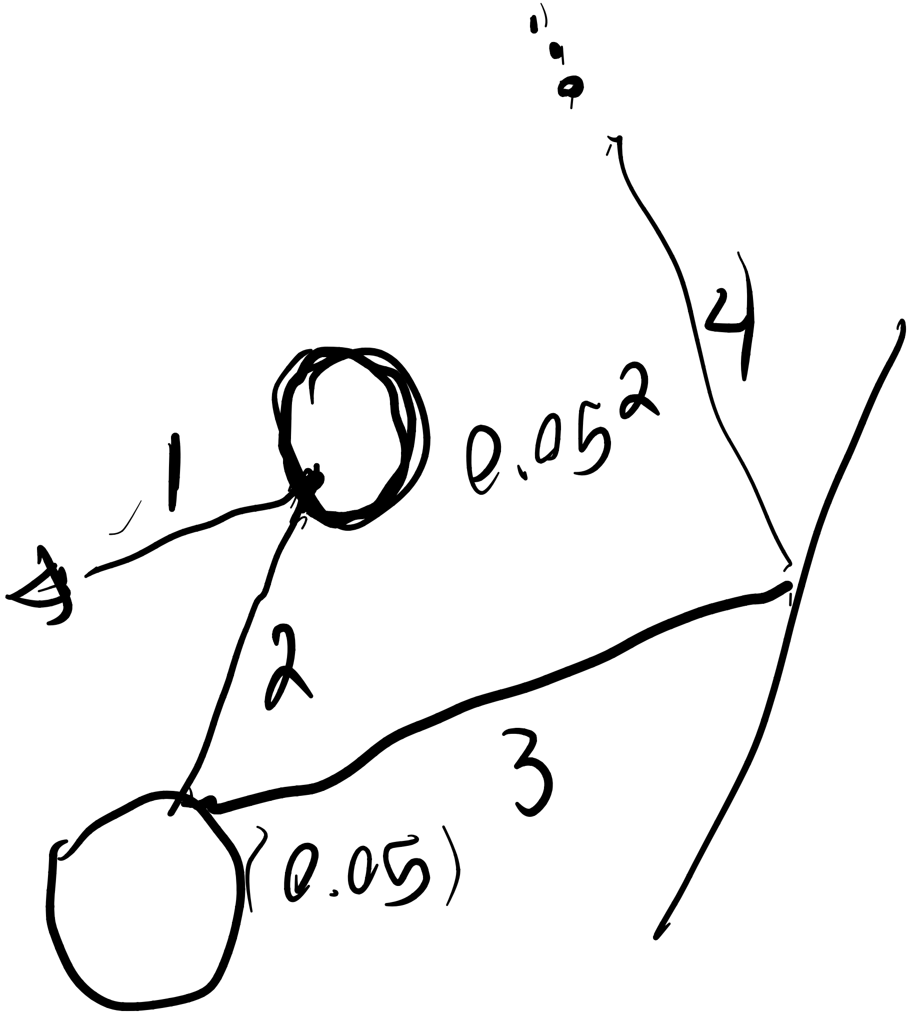
$$S_i = \begin{cases} 0 & \text{– if shadow ray (light ray) is blocked} \\ 1 & \text{– if not blocked (reached light)} \end{cases}$$


V
N
R
mirror
like appearance
primary
ray
R
secondary
ray
compute LR

ambient + diffuse + specular + $k_r L_r$ + $k_t L_t$

how reflective?

how trans



shootRay(R) → L

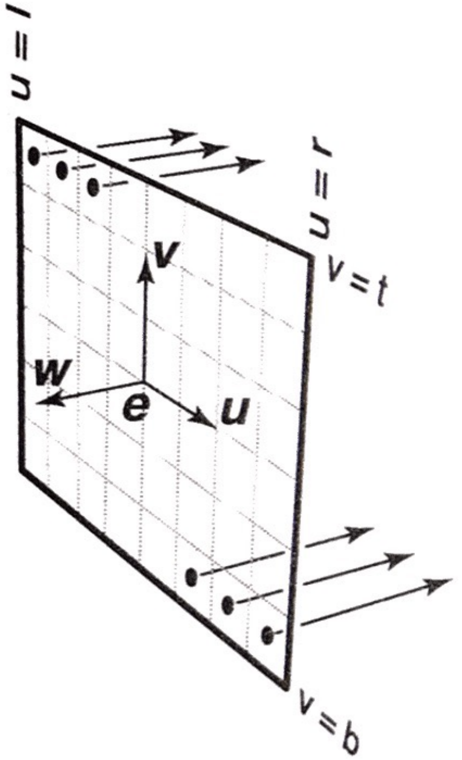
when to stop?

1) set max recursive step

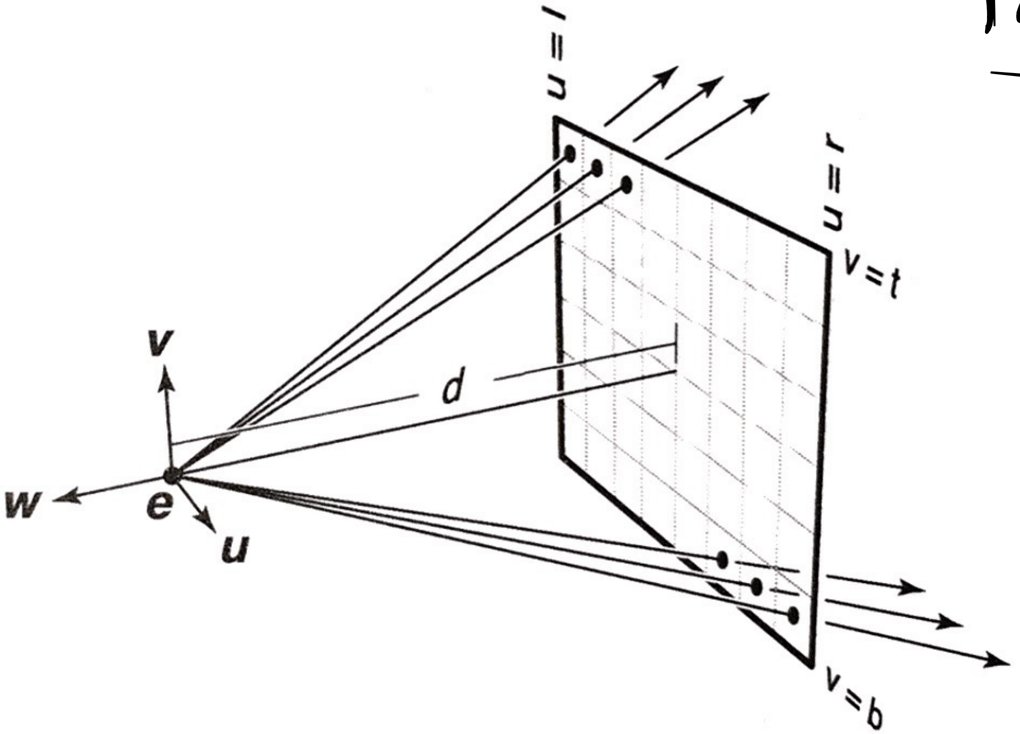
2) contribution of ray is small

Eye Rays: Depends on Projection (Orthographic, Perspective, Oblique)

fov



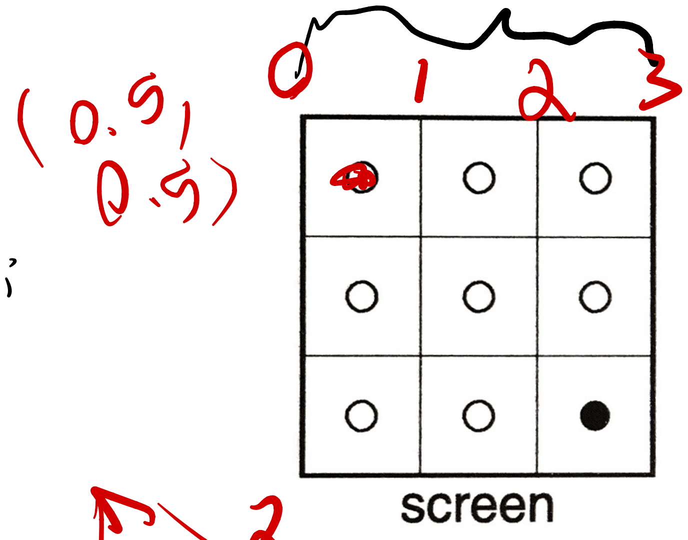
Parallel projection
same direction, different origins

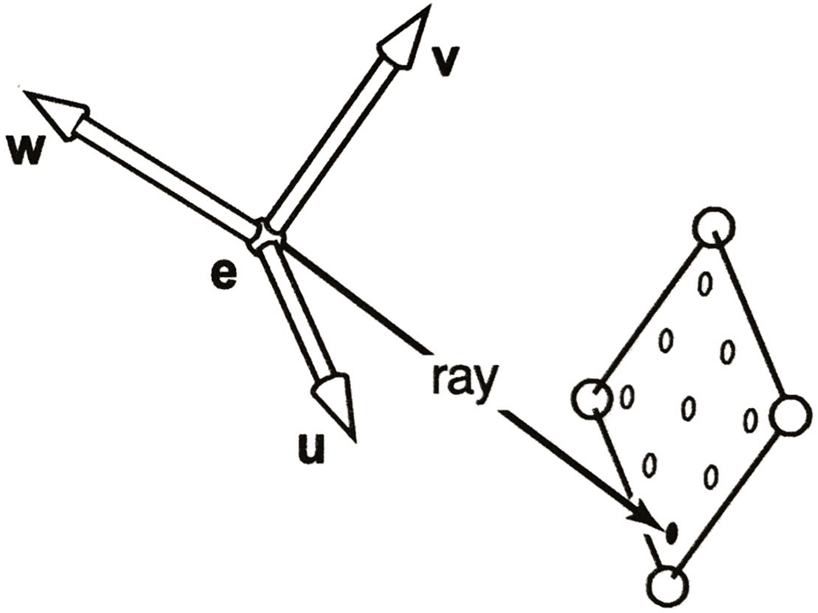


Perspective projection
same origin, different directions

Parametric eq'n:

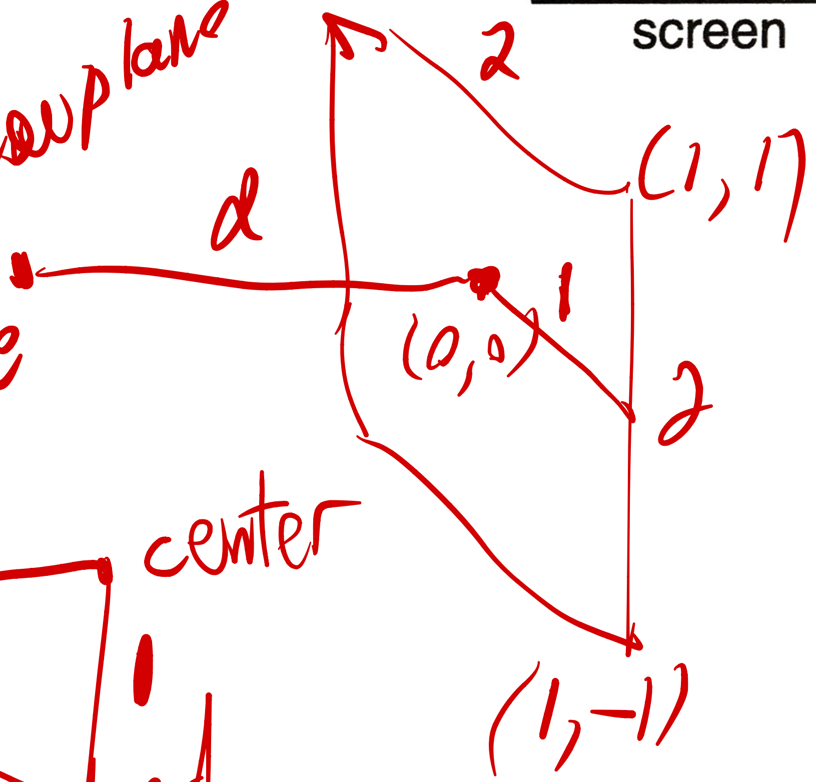
$$p(t) = e + t(s - e)$$



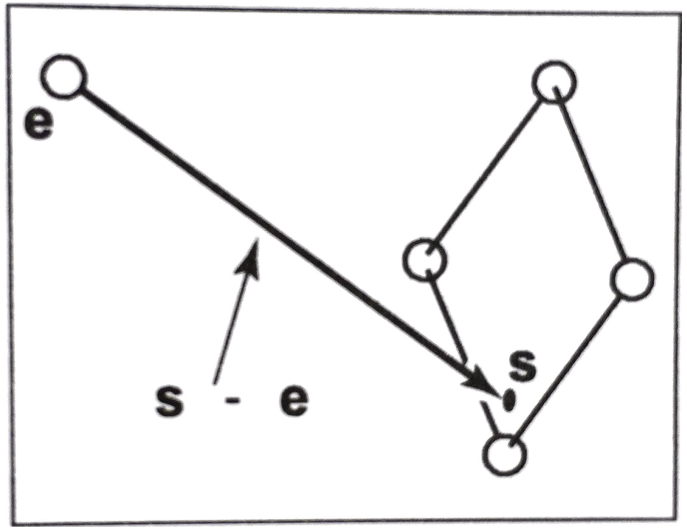


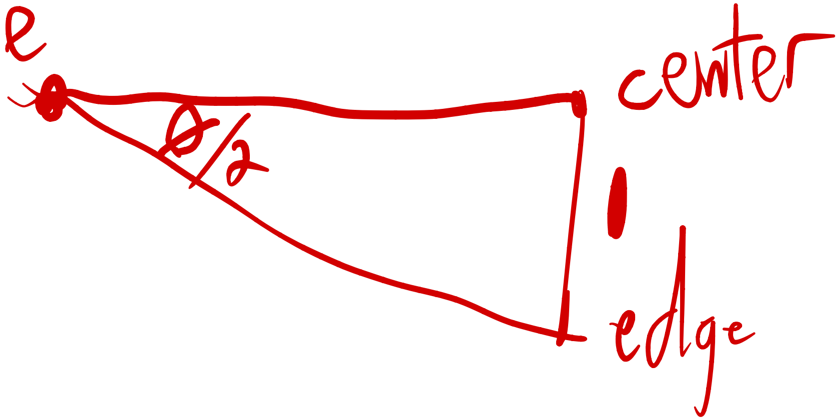
1) fov θ

2) size of viewplane



$$d = \frac{1}{\tan\left(\frac{\theta}{2}\right)}$$



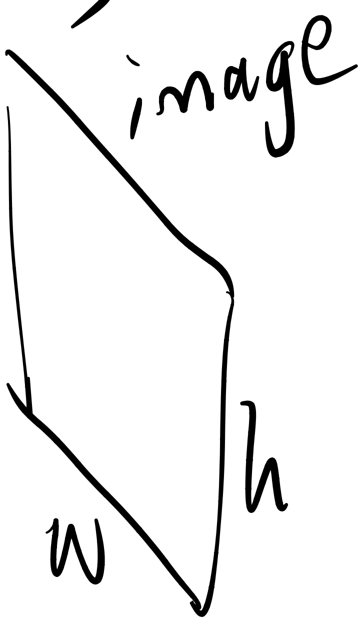


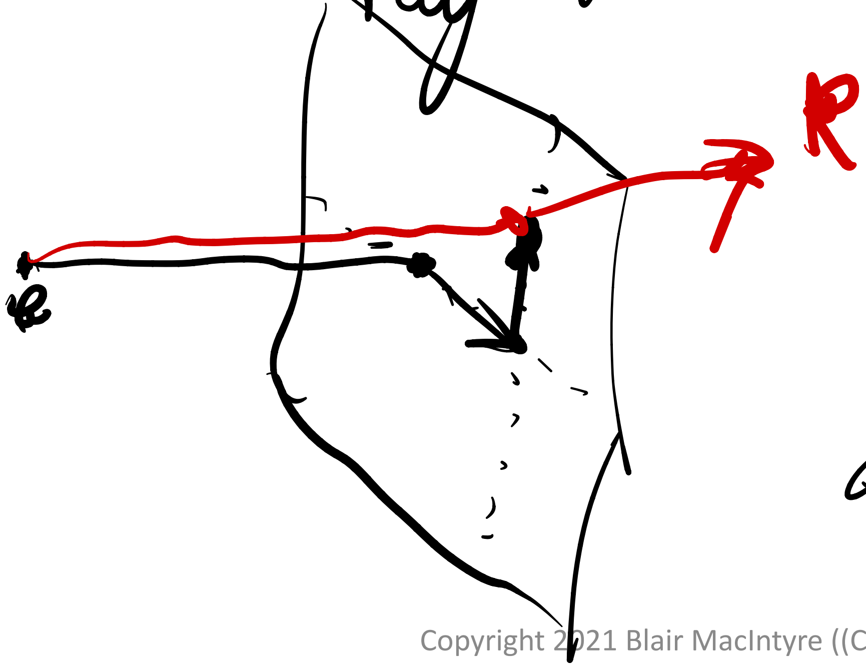
$$u_s = -1 + \frac{2i}{w}$$
$$v_s = -1 + \frac{2i}{h}$$

u & v

$-1 \ldots 1$ range

$\mathbf{u}$
$\mathbf{v}$
$\mathbf{w}$





$$\text{ray direction} = \frac{s - e}{|s - e|}$$

$$s = -d\mathbf{w} + u_s \mathbf{u} + v_s \mathbf{v}$$

$$\text{origin} = e$$

for

for i = 0 to $w-1$

for j = 0 to $h-1$

computeR(u_s, v_s) using (i, j)

j ~~not~~ i

shootRay(R)

Computing Intersections

different for each object

- sphere is easy
- polygons are easy
- objects mesh (compute per triangle)
- implicit surfaces

Sphere/Ray Intersections

$p(t) = e + td \quad d = s - e$

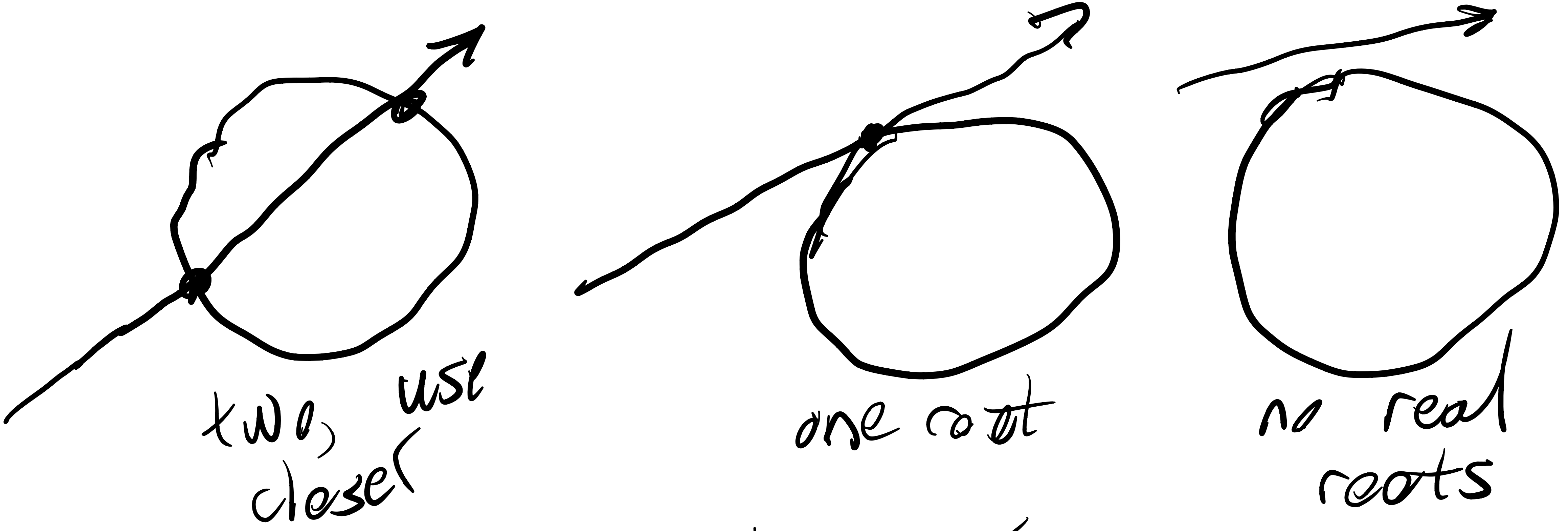
canonical sphere
center at $(0,0,0)$
radius of 1

$$x^2 + y^2 + z^2 = 1^2$$

$$(x_e + t d_x)^2 + (y_e + t d_y)^2 + (z_e + t d_z)^2 = 1$$

$$t^2 \underbrace{(d_x^2 + d_y^2 + d_z^2)}_{a} + t \underbrace{2(x_e d_x + y_e d_y + z_e d_z)}_{b} + \underbrace{x_e^2 + y_e^2 + z_e^2 - 1}_{c} = 0$$

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

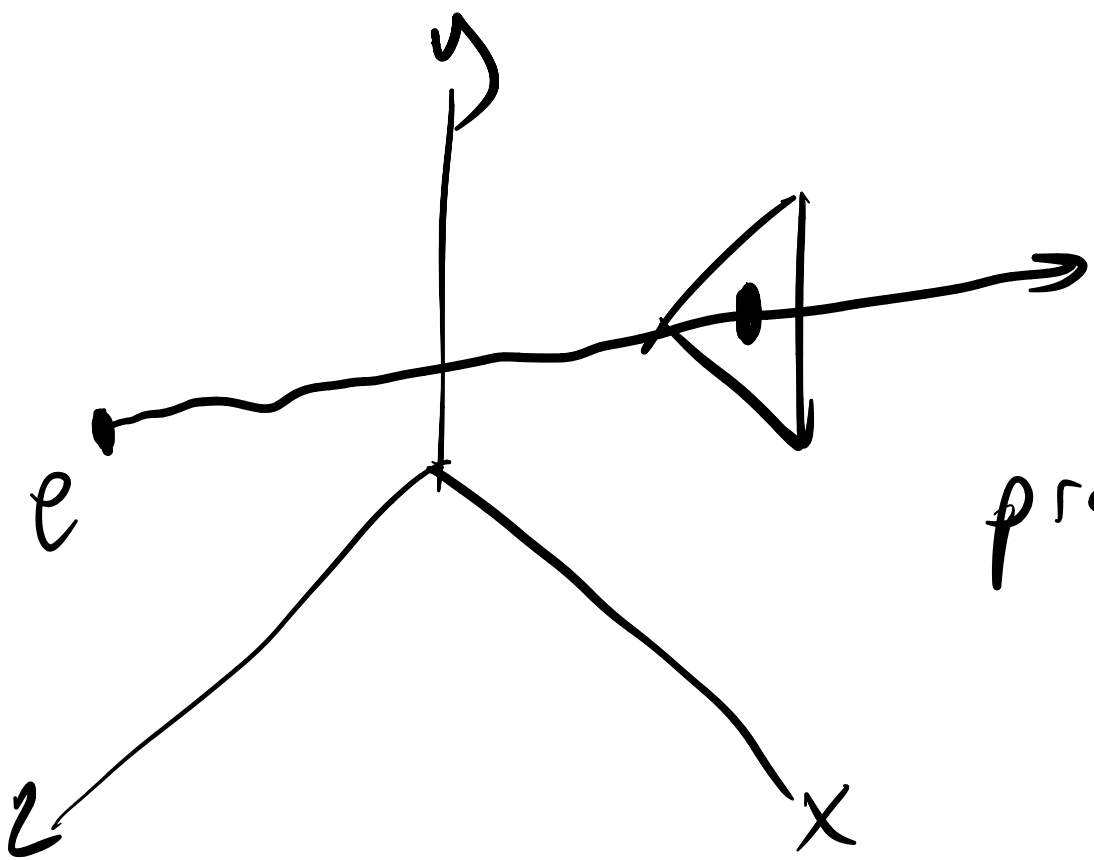


general case

center $= (x_c, y_c, z_c)$
radius $= R$

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = R^2$$

Ray/Triangle Intersection



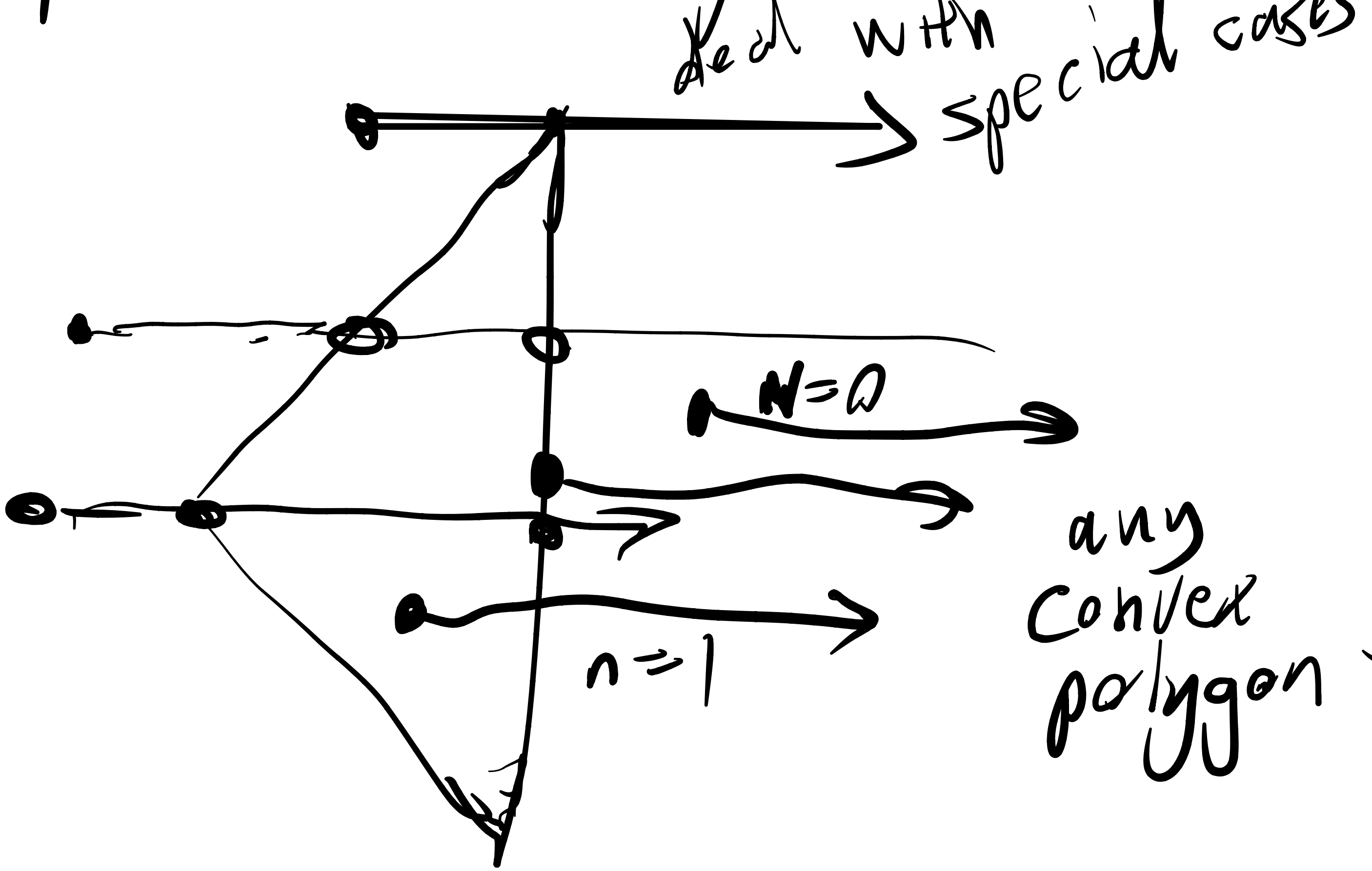
compute intersection of $\hat{R}$ with the plane of the triangle

project to 2D

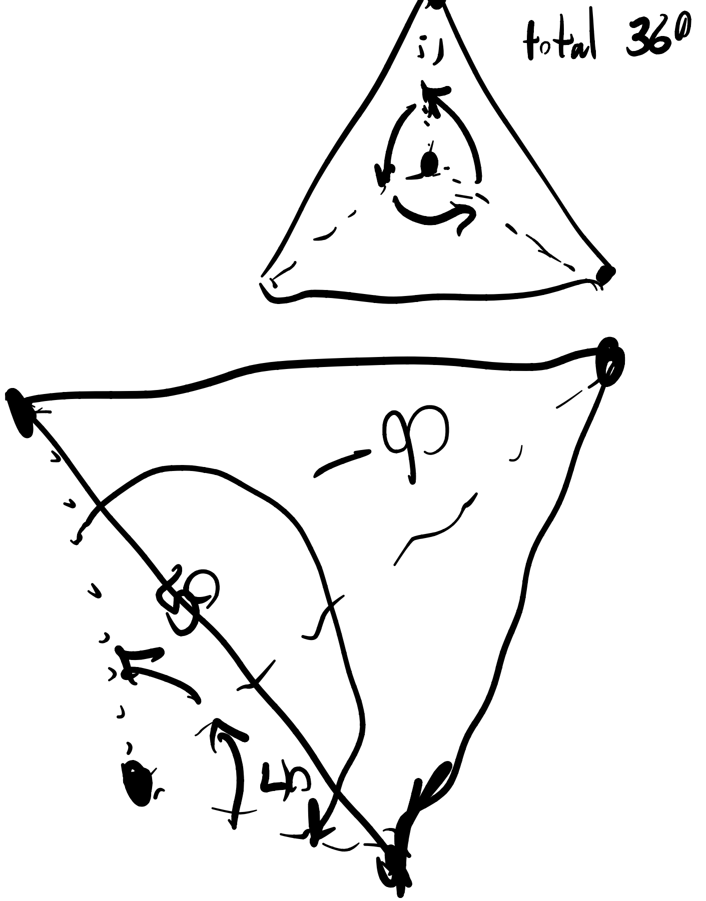
→ going to xy, xz, or yz planes

based of Normal of plane (n_x, n_y, n_z) the largest

Point in Triangle in 2D



any convex polygon too!

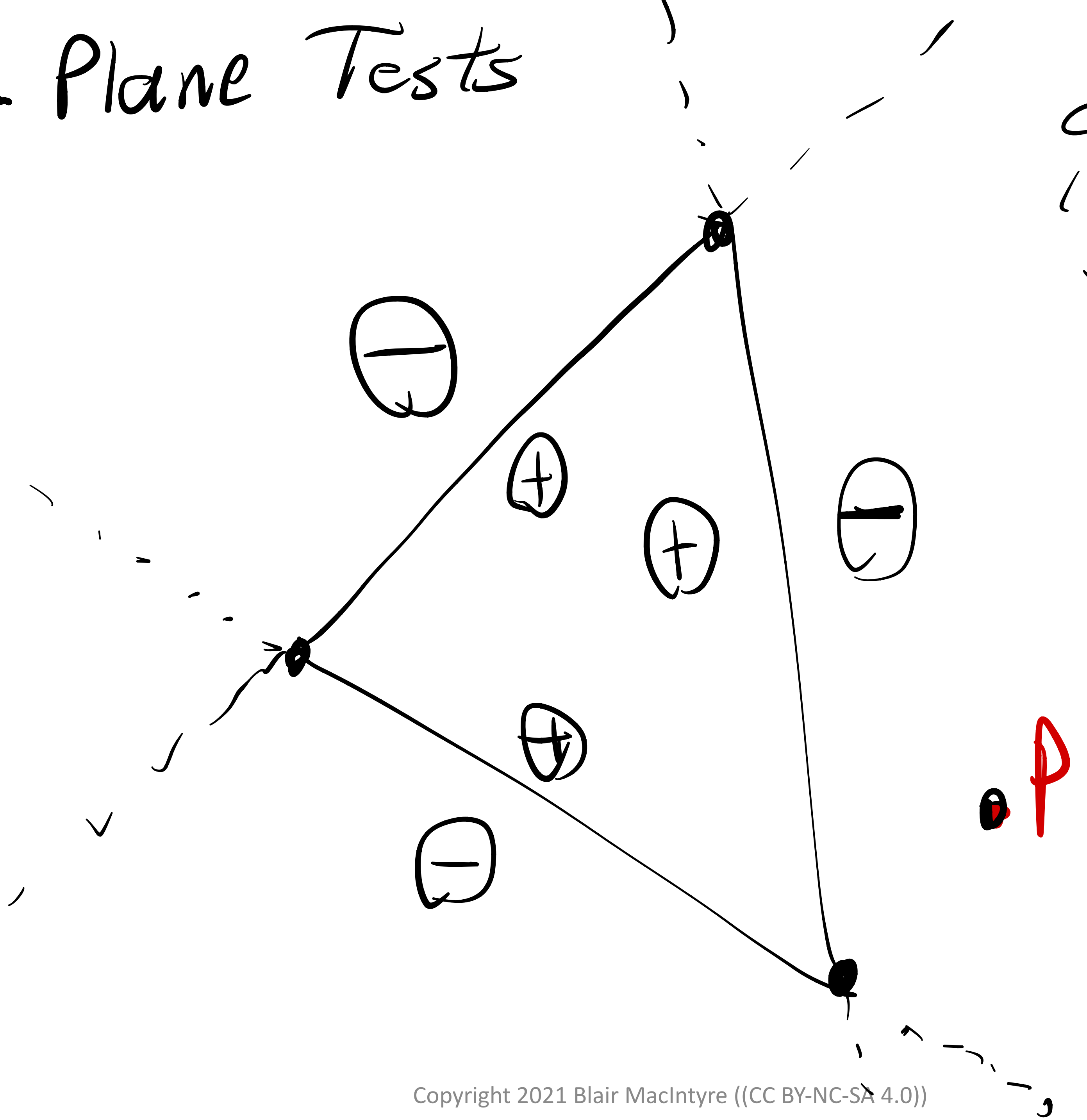


$40 + 50 - 90 = 0$

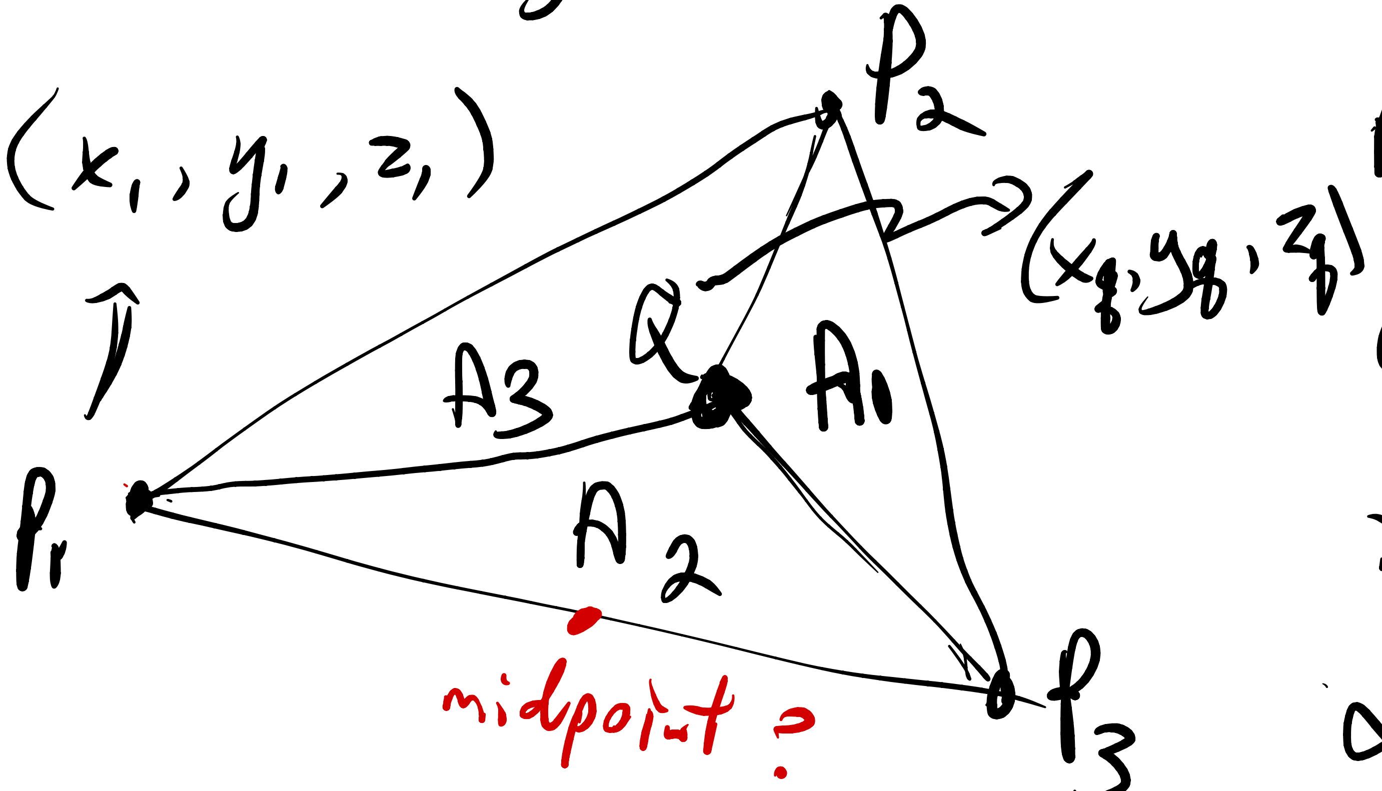
even number, outside
odd, inside

half - Plane Tests

compute implicit line equation for each line



Barycentric Coordinates



A_1 = area of sub-triangle opposite P_1

$(A_2 ; A_3)$ ...

$$A = A_1 + A_2 + A_3$$

$$\alpha = A_1 / A$$

$$\beta = A_2 / A$$

$$\gamma = A_3 / A$$

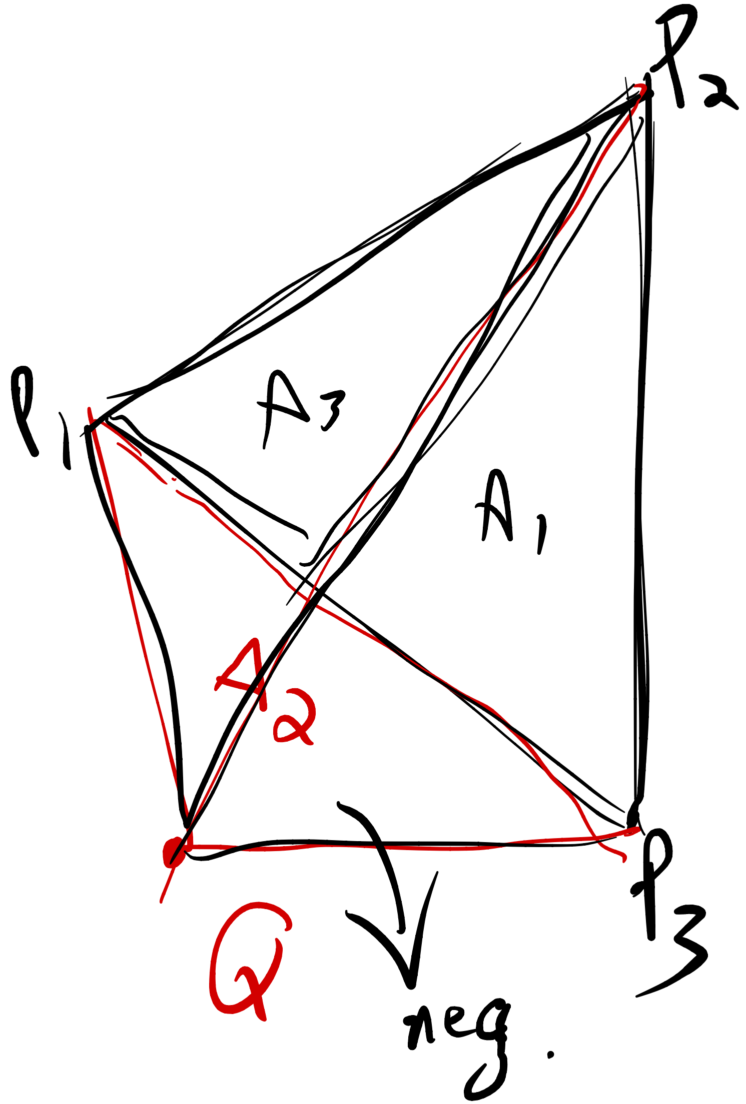
$$\alpha + \beta + \gamma = 1$$

$$Q = \alpha P_1 + \beta P_2 + \gamma P_3$$

$$Q = \alpha P_1 + \beta B_2 + \gamma P_3$$

α, β, γ are positive inside tri

one or more negative if point is outside



Computing Plane Intersection: Implicit Line Equation