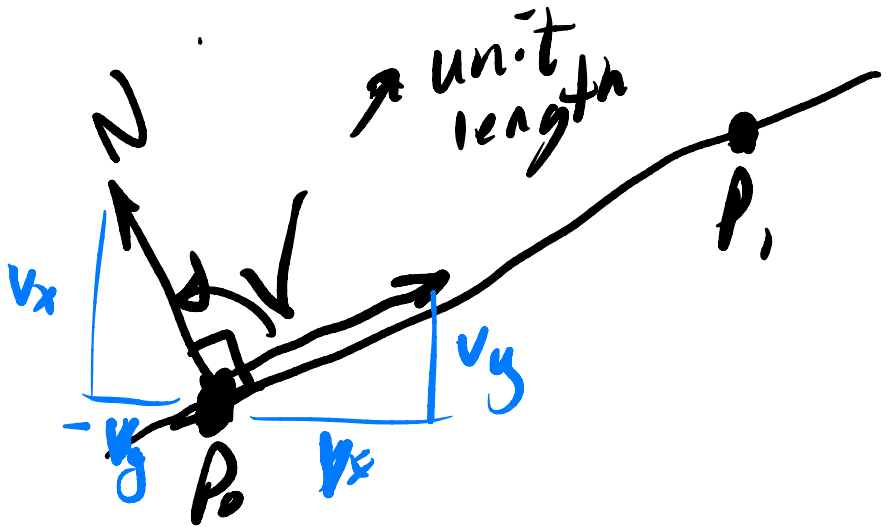
14 – raytracing (2)

Computing Plane Intersection: Implicit Line and Plane Equations

(Ray → Polygon; Ray → Plane & then see if intersection is in polygon)



$$R = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$V = \frac{P_1 - P_0}{|P_1 - P_0|}$$

$$V = (v_x, y_x)$$

$$N = (-v_y, v_x)$$

substitute P_0 into eq., we can solve for c

$$f(x,y) = ax + by + c = 0$$

$$= -v_y x + v_x y + c = 0$$

Implicit Plane Equation

$$f(x,y,z) = ax + by + cz + d = 0$$

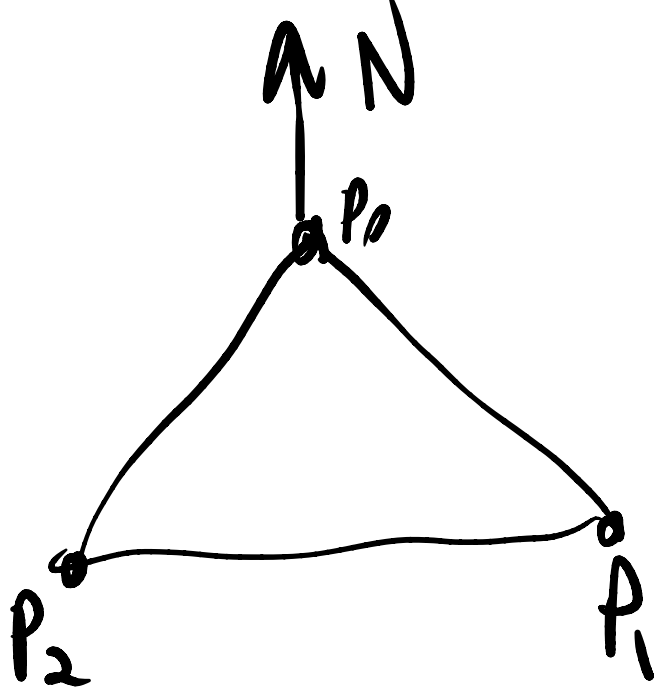
$E_1 = P_1 - P_0$

$E_2 = P_2 - P_0$

$$N = \frac{E_1 \times E_2}{|E_1 \times E_2|}$$ ← unit length

$N = (a, b, c)$

Substitute $P_0 = (x_0, y_0, z_0)$ into $ax + by + cz + d = 0$ & solve for d



line eq:

$x(t) = x_0 + t\,dx$

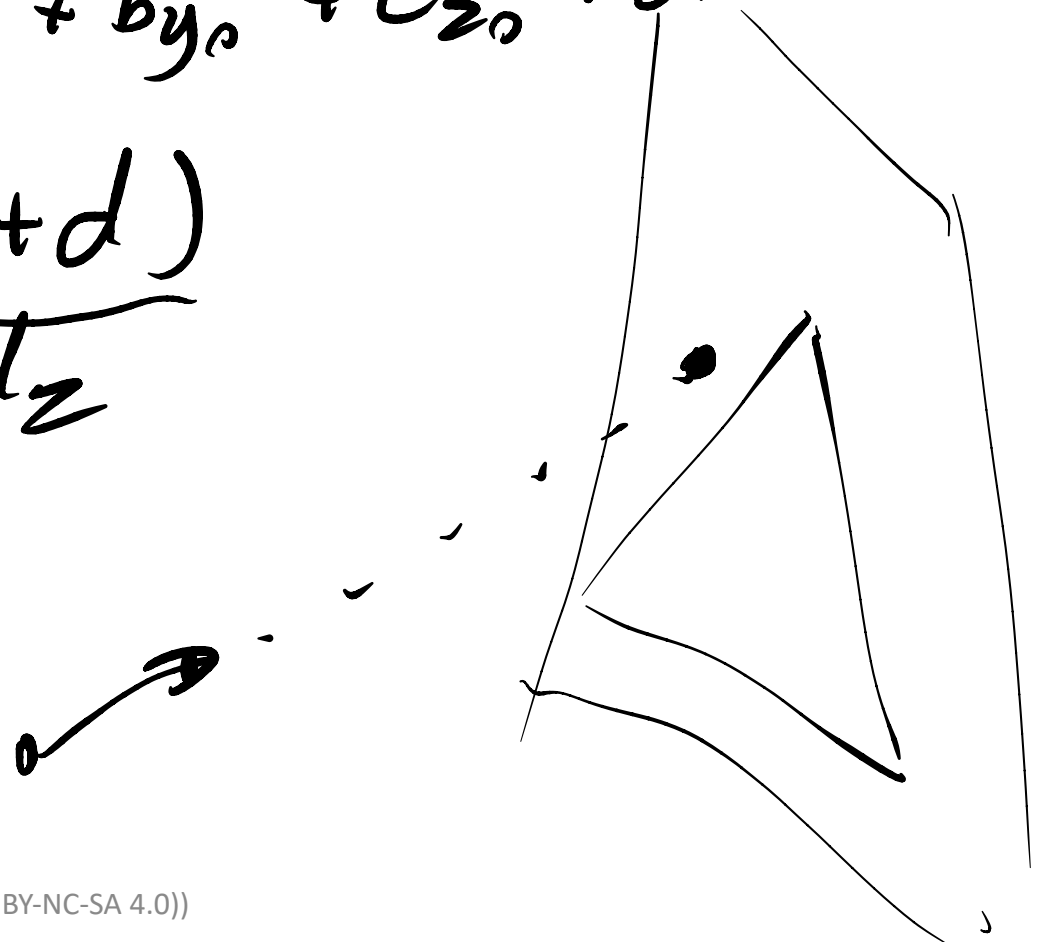
⋮

$$a(x_0 + t d_x) + b(y_0 + t d_y) + c(z_0 + t d_z) + d = 0$$

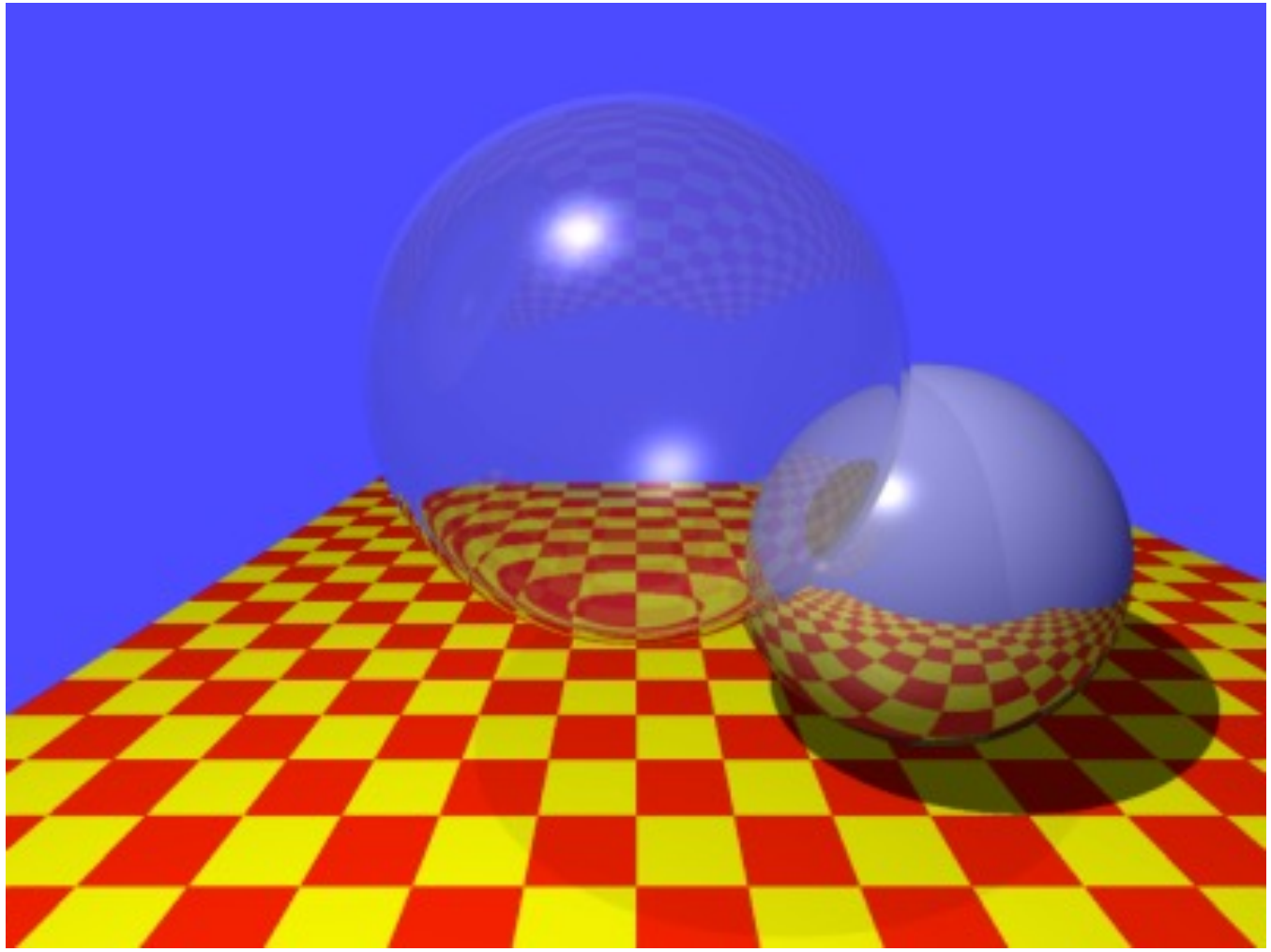
$$t(a d_x + b d_y + c d_z) + a x_0 + b y_0 + c z_0 + d = 0$$

$$t = \frac{-(a x_0 + b y_0 + c z_0 + d)}{a \cdot d_x + b d_y + c \cdot d_z}$$

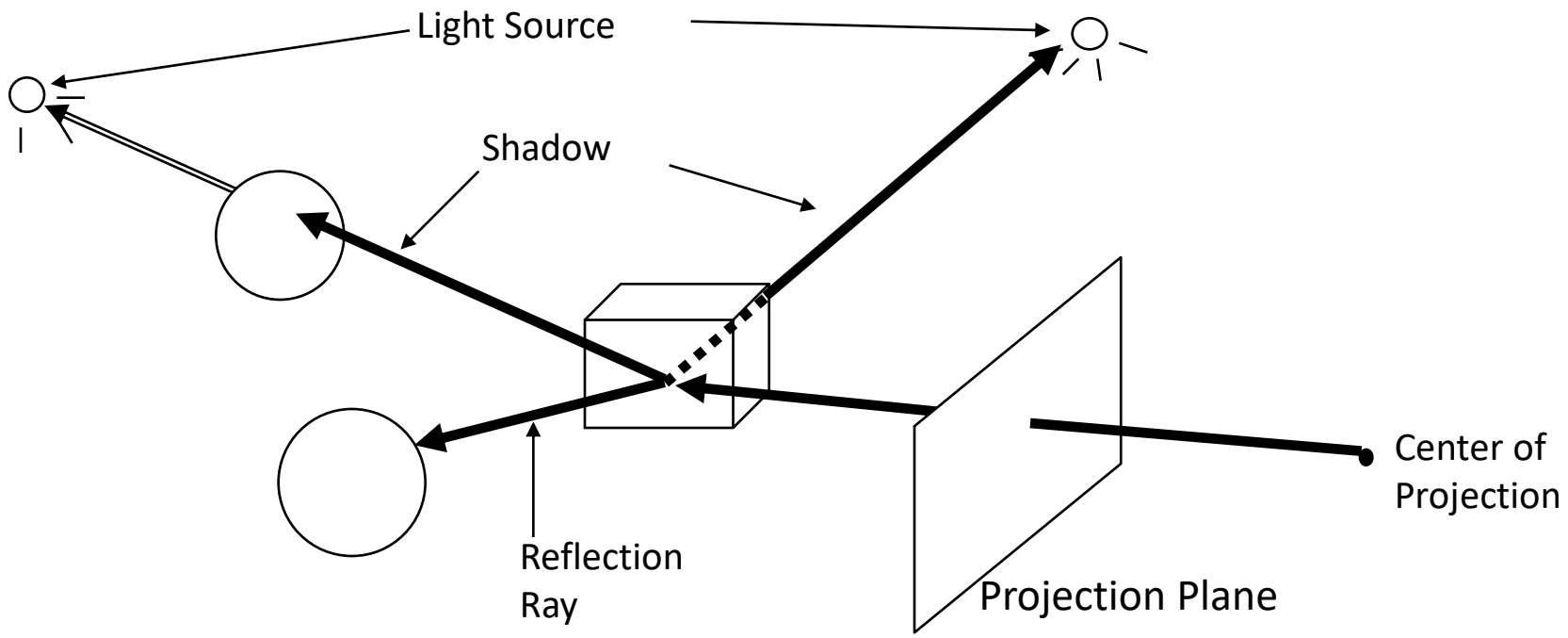
N · d



What about these other rays?



Basic Idea



Illumination of a point

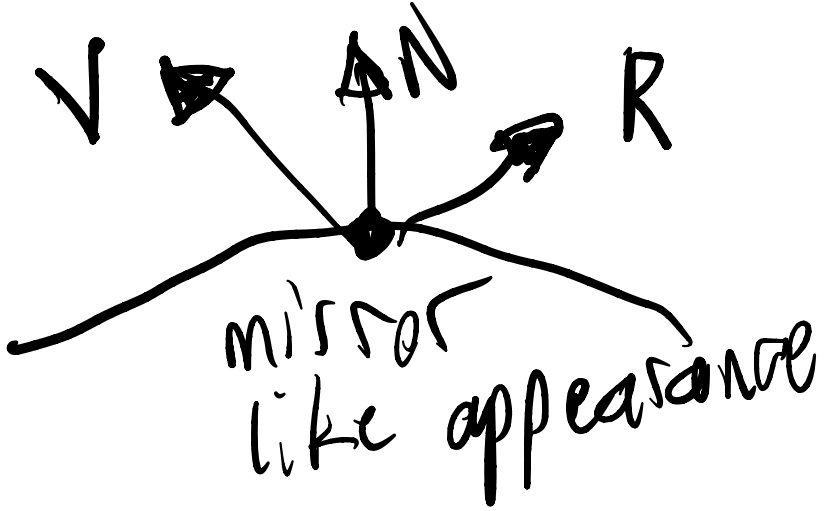
ambient

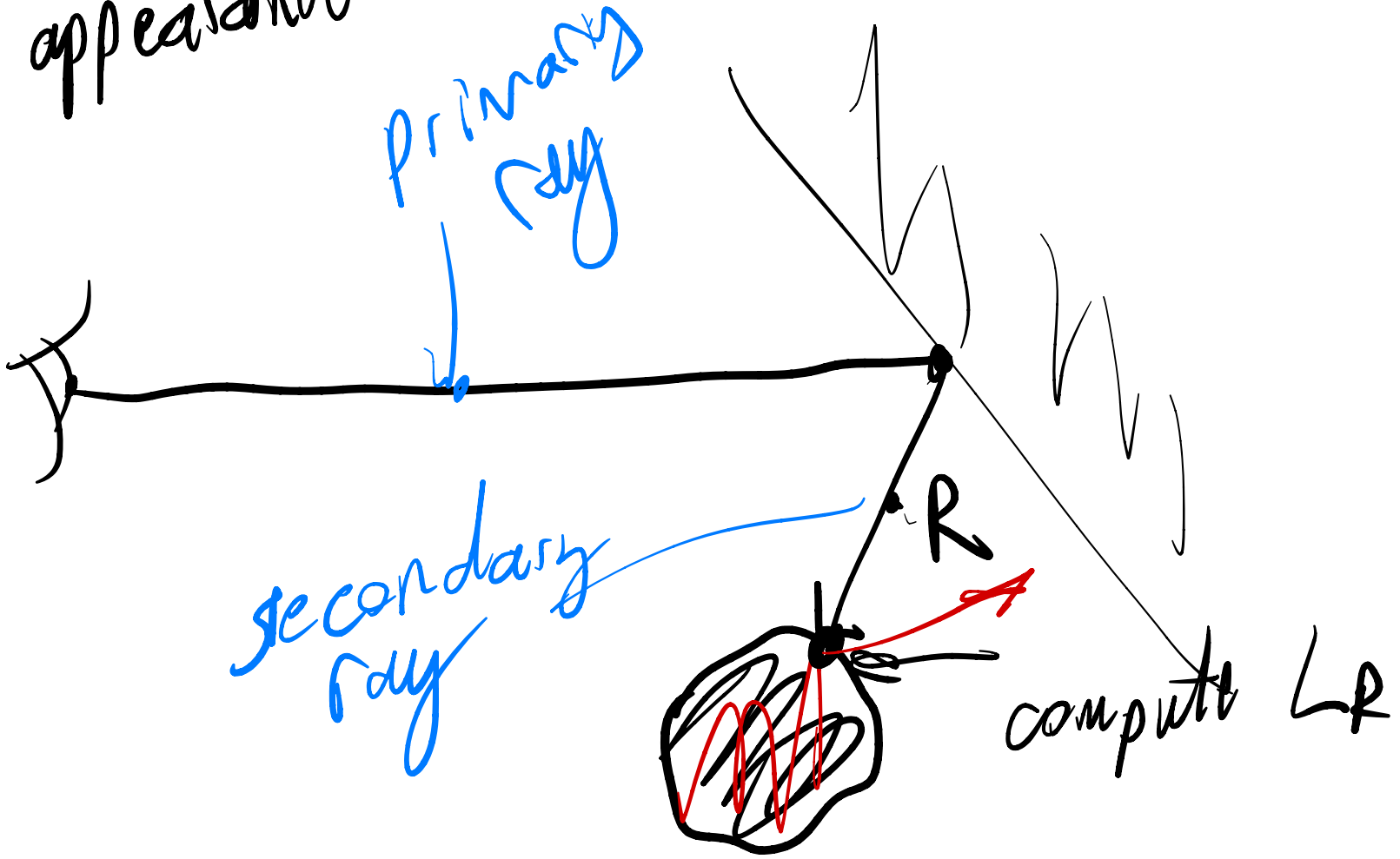
previous eq

$$L = k_a I_a + k_s L_r + k_t L_t + \sum_{1 \le i \le N} S_i I_i \left[k_d (N \cdot L_i) + k_s (R_i \cdot V)^{p_i} \right]$$

reflected refracted

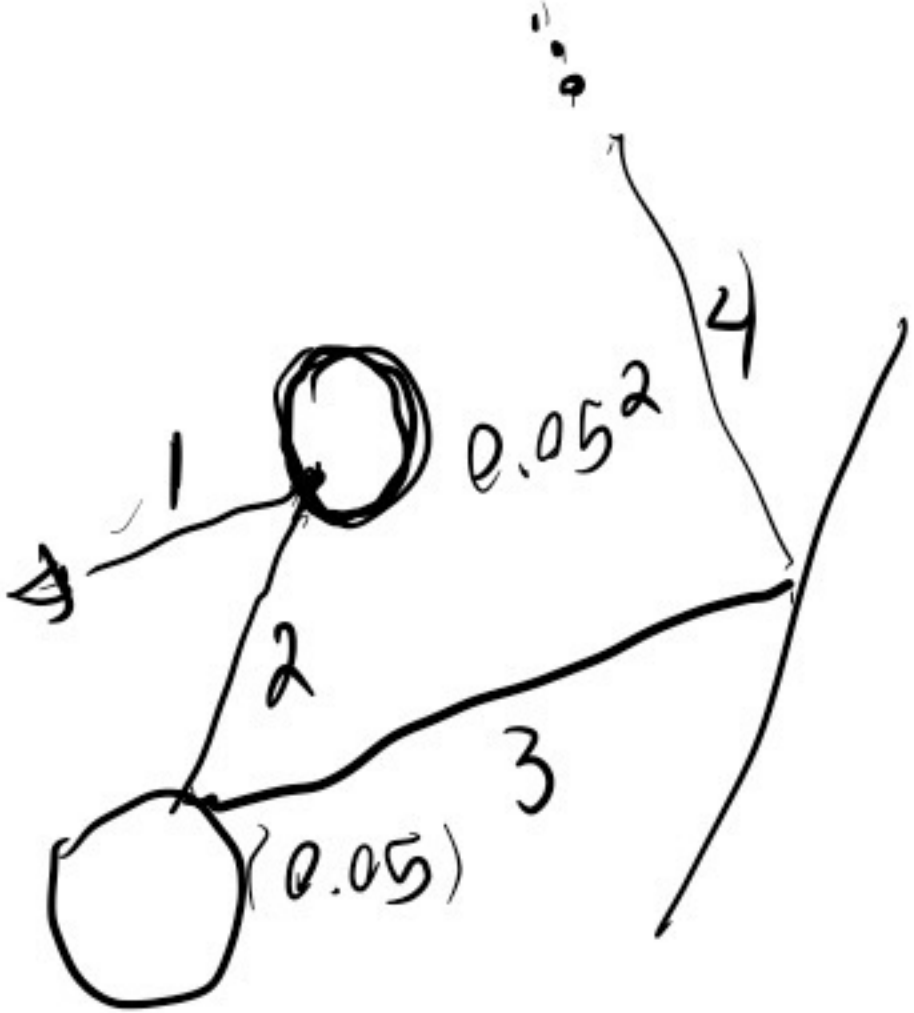
$$S_i = \begin{cases} 0 & \text{– if shadow ray (light ray) is blocked} \\ 1 & \text{– if not blocked (reached light)} \end{cases}$$





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

how reflective? how trans

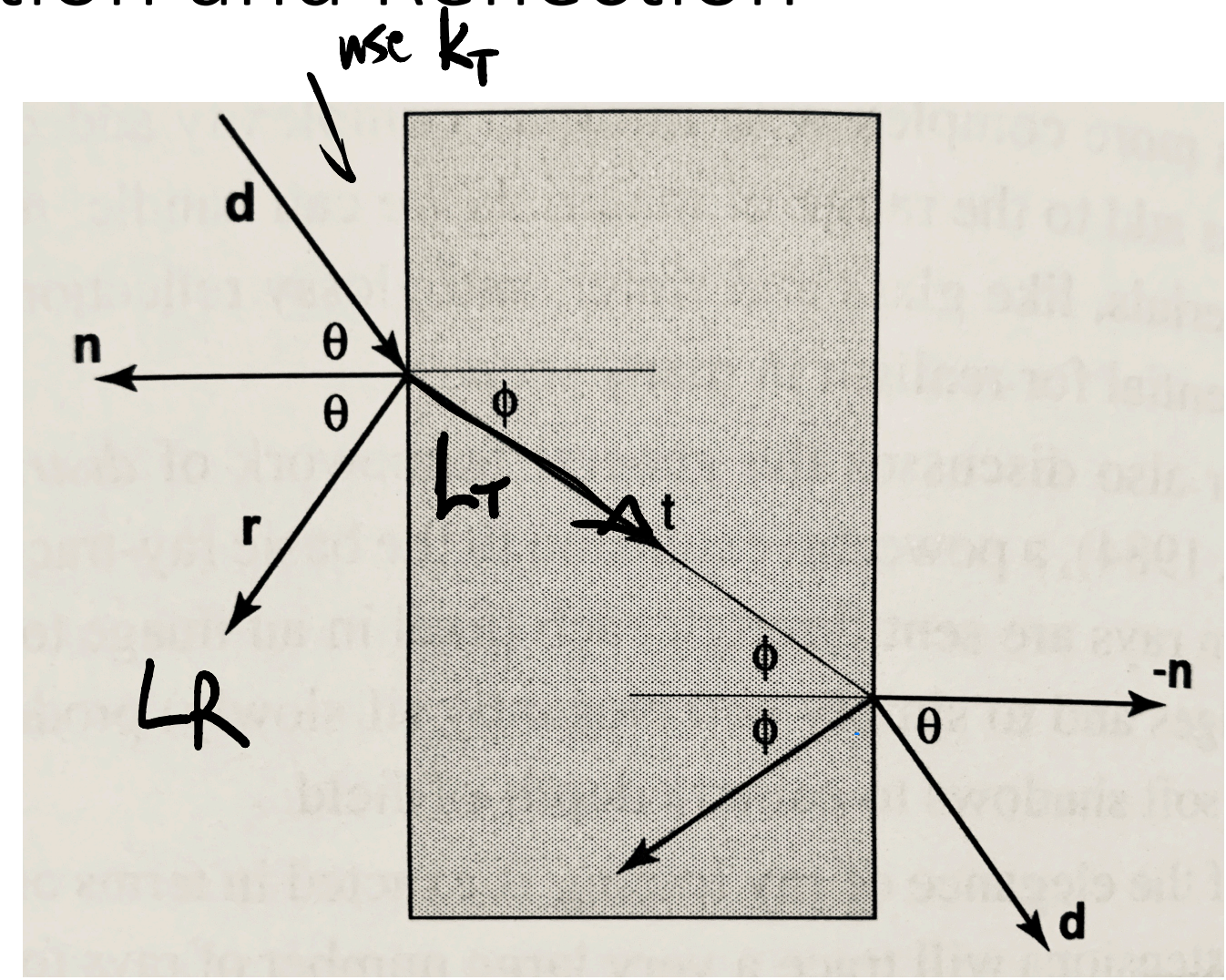


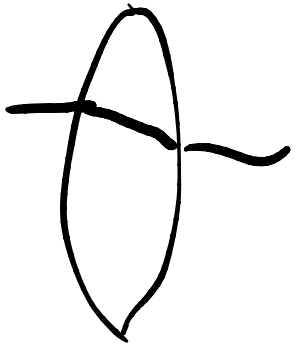
shootRay(R) → L

When to stop?

1) set max recursive step
2) contribution of ray is small

Diffraction and Reflection





Transparent Surfaces - (Refraction)

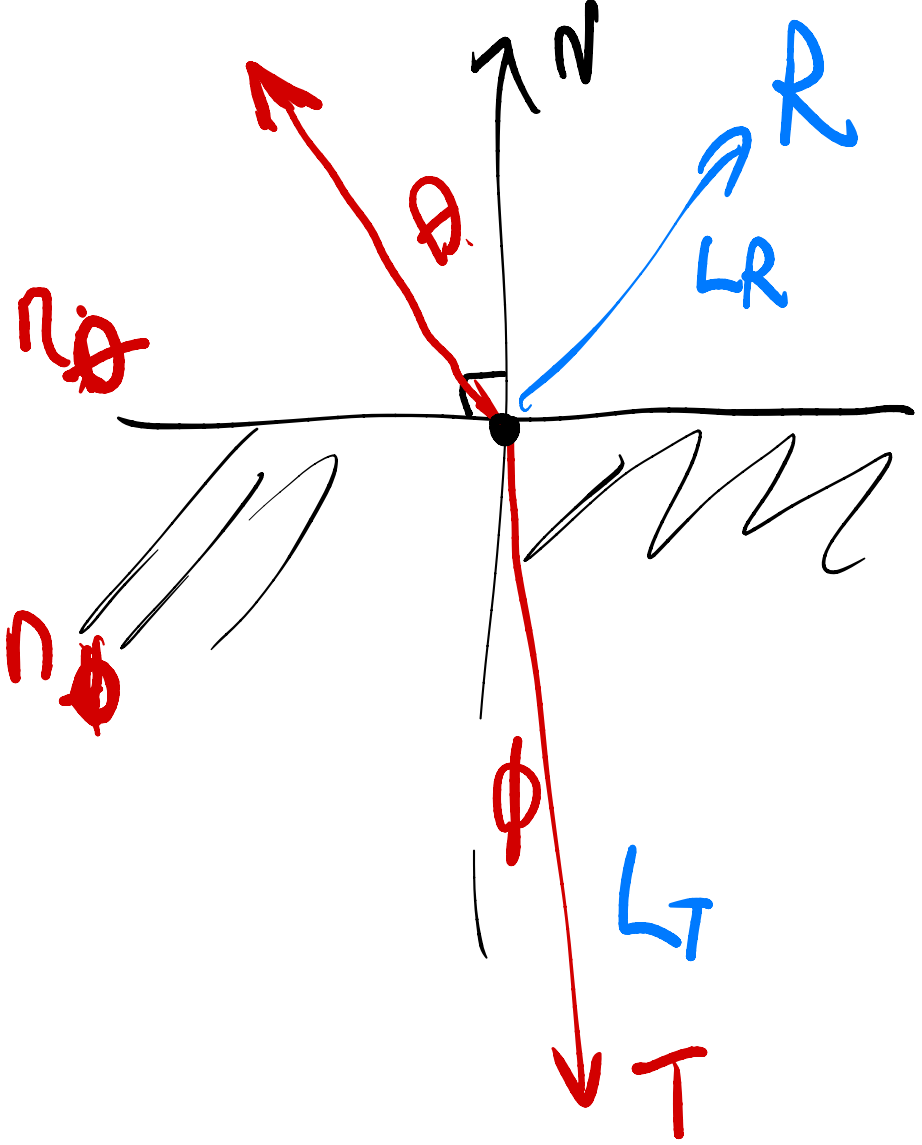
Light travels at different speeds through diff materials

Refraction: glasses, mag glasses
glass of liquid

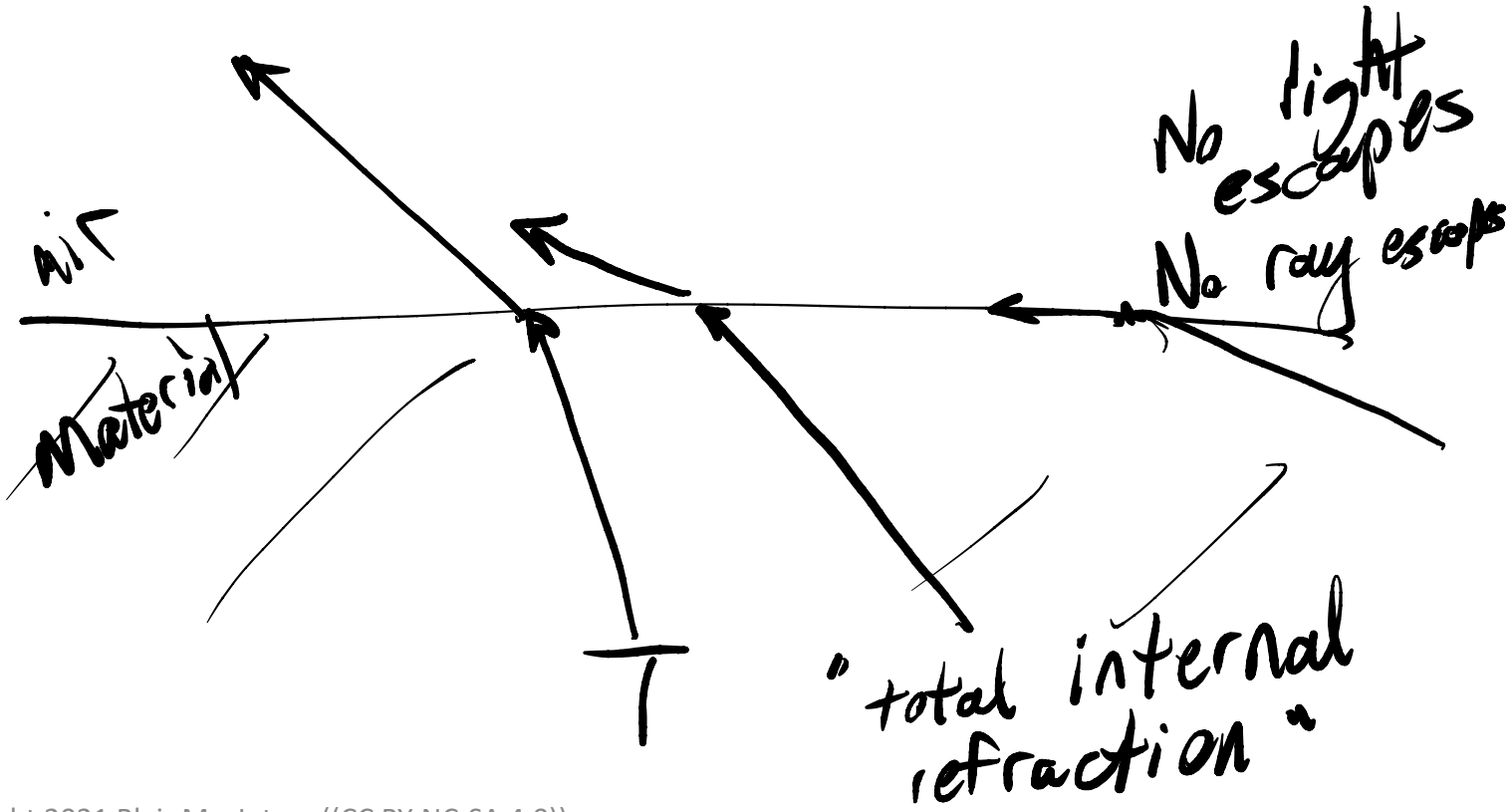
$$\text{Index of refraction} = \frac{\text{speed of light thru vacuum}}{\text{speed of light thru material}}$$

$$\geq 1$$

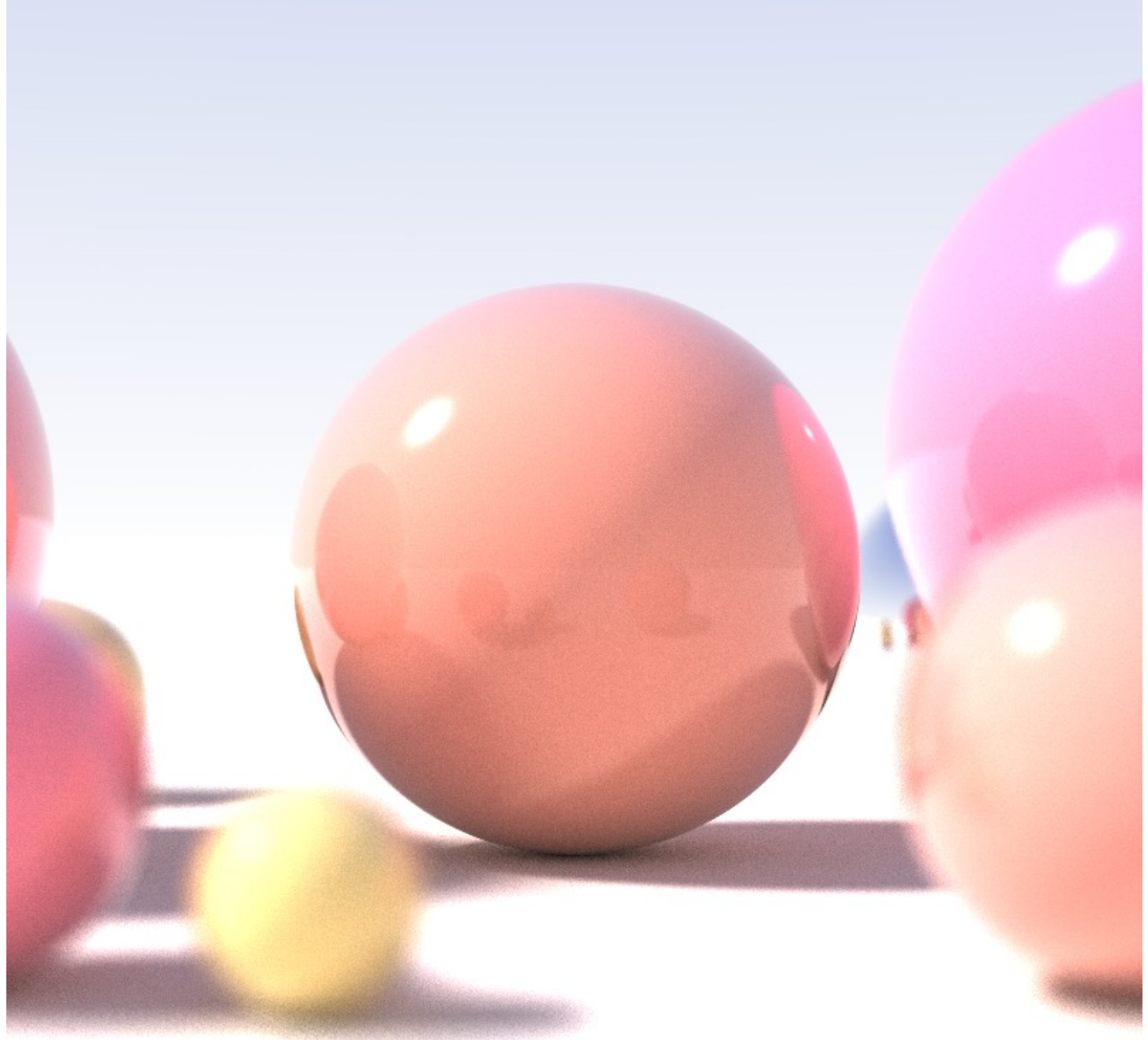
material	η
vacuum	1
air	1.0003
water	1.33
glass	~1.9
ice	1.309
quartz	1.544
diamond	2.417

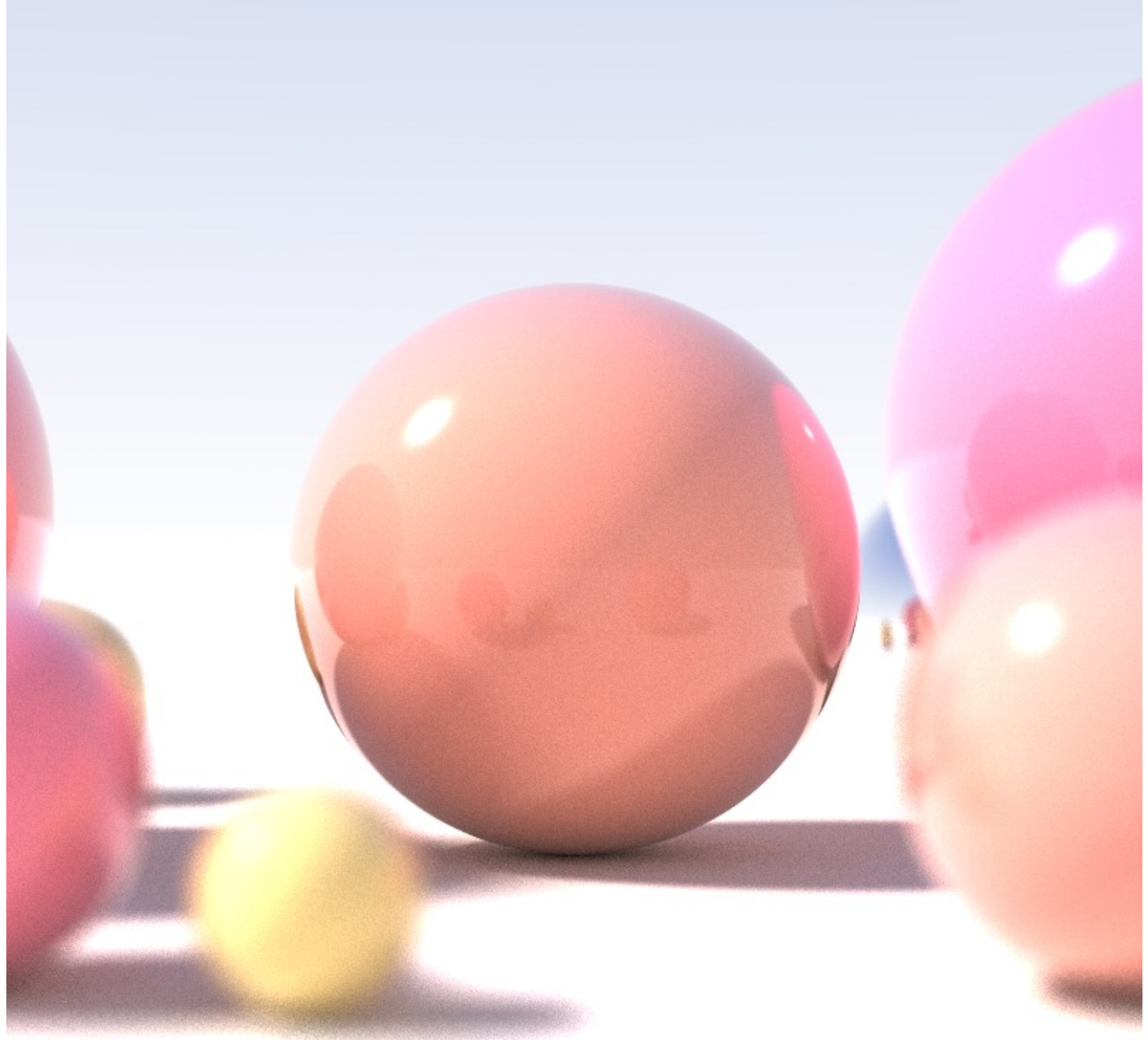


$$\frac{\sin\theta}{\sin\phi} = \frac{n_\phi}{n_\theta}$$



https://www.theverge.com/2013/6/21/4446606/how-pixar-changed-the-way-light-works-for-monsters-university

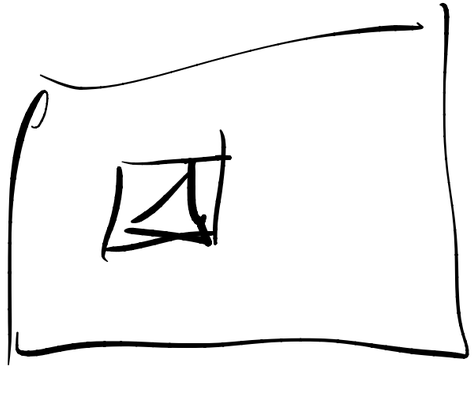




2 Rendering Methods

Rasterization & Z-buffering

```
for each object
    for each pixel in object projection
        test z-value
        shade
```



Ray Tracing

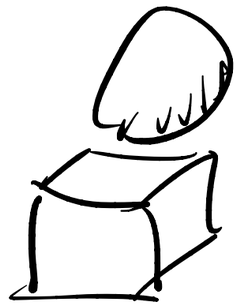
```
For each pixel on screen
    shoot R ray
        for each object
            does R hit it?
            (keep track of closest
            shade it
```

Ray Tracing Speed Ups

Fast to Intersect

spheres

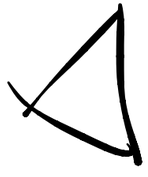
box



cylinder

ellipsoids

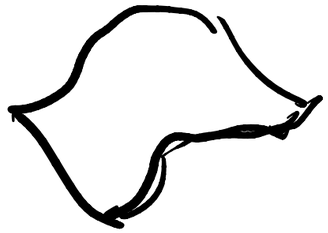
polygons



slow

torus

surfaces (cubic)



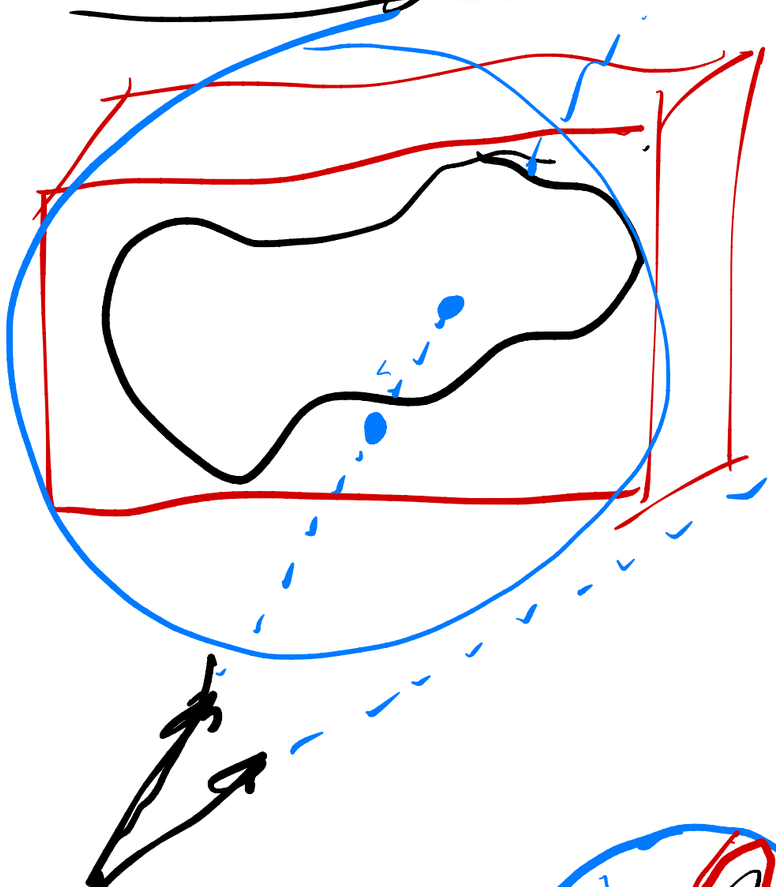
blobby

collections of stuff

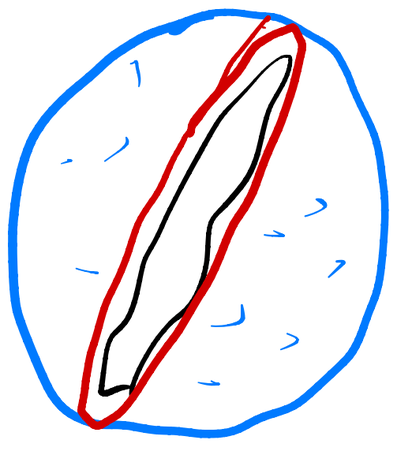
spheres,

polygons → meshes

Bounding Volumes



sphere
of
cube

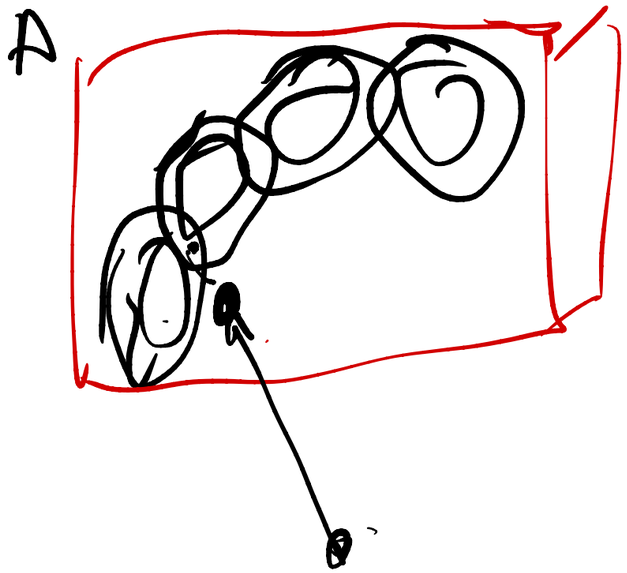


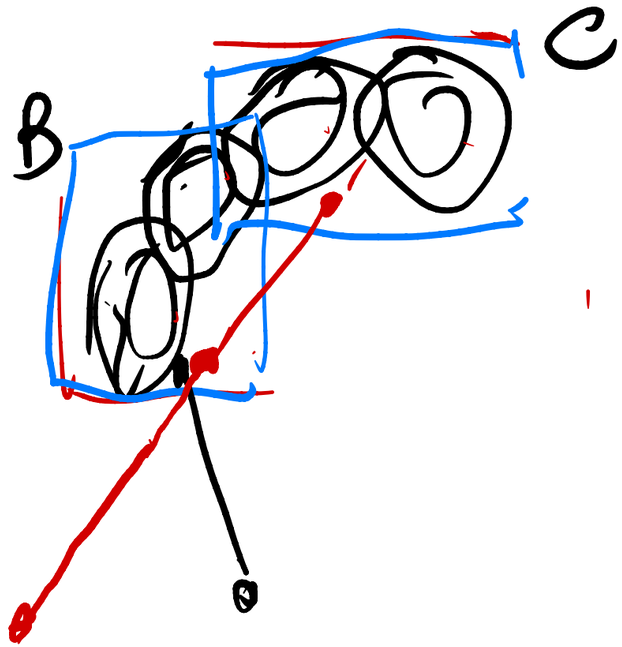
outer volume (sphere cube or ellipse or cylinder) is efficient to test

VS

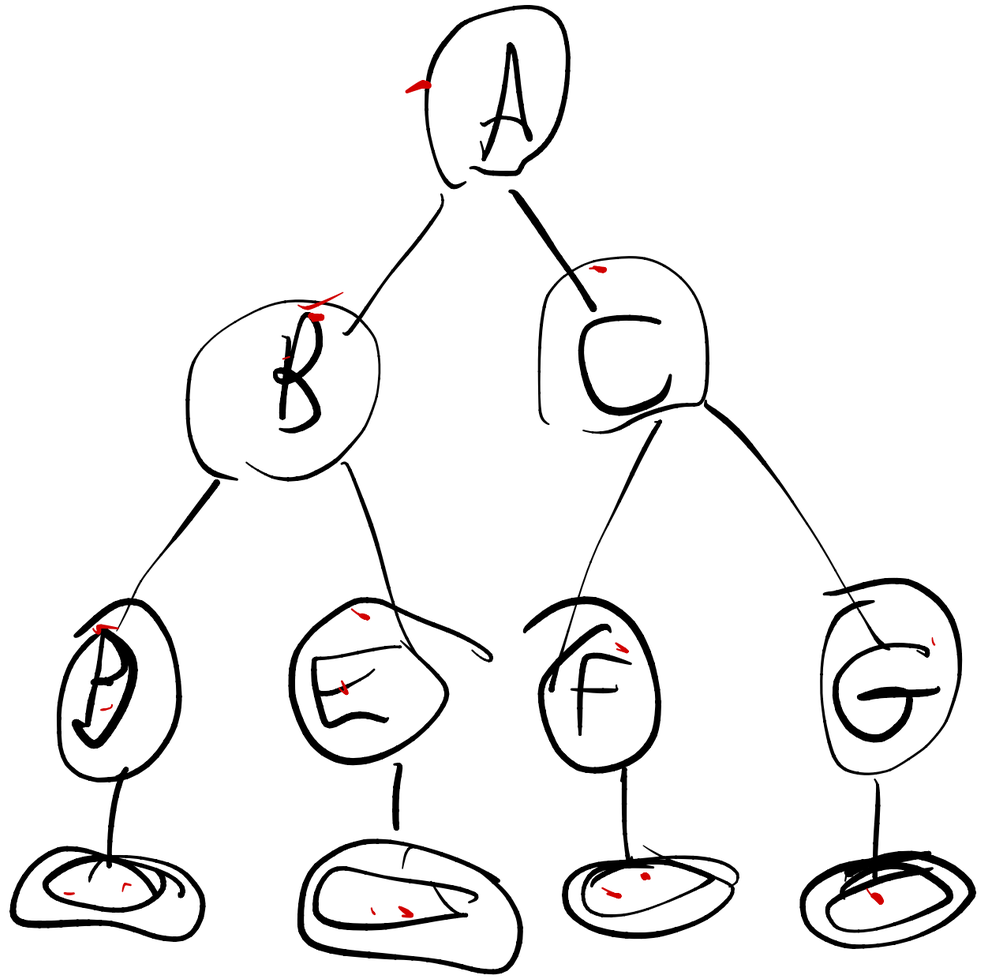
cost of what's inside

Want outer volume to be as tight as possible





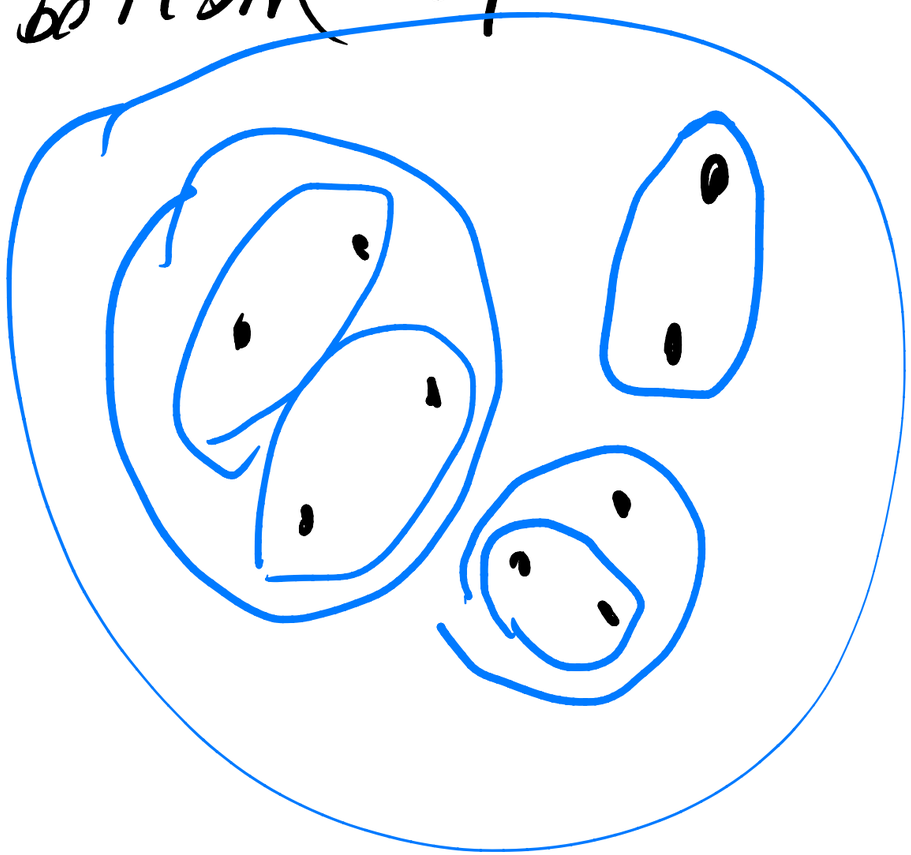
Bounding Volume Hierchies



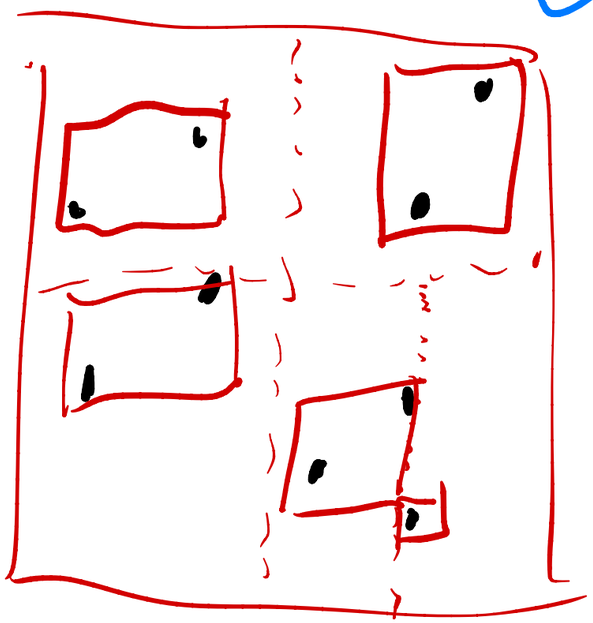
Build Bounding Hierchies

- small / tight fit between volume & stuff
- balanced tree

bottom-up

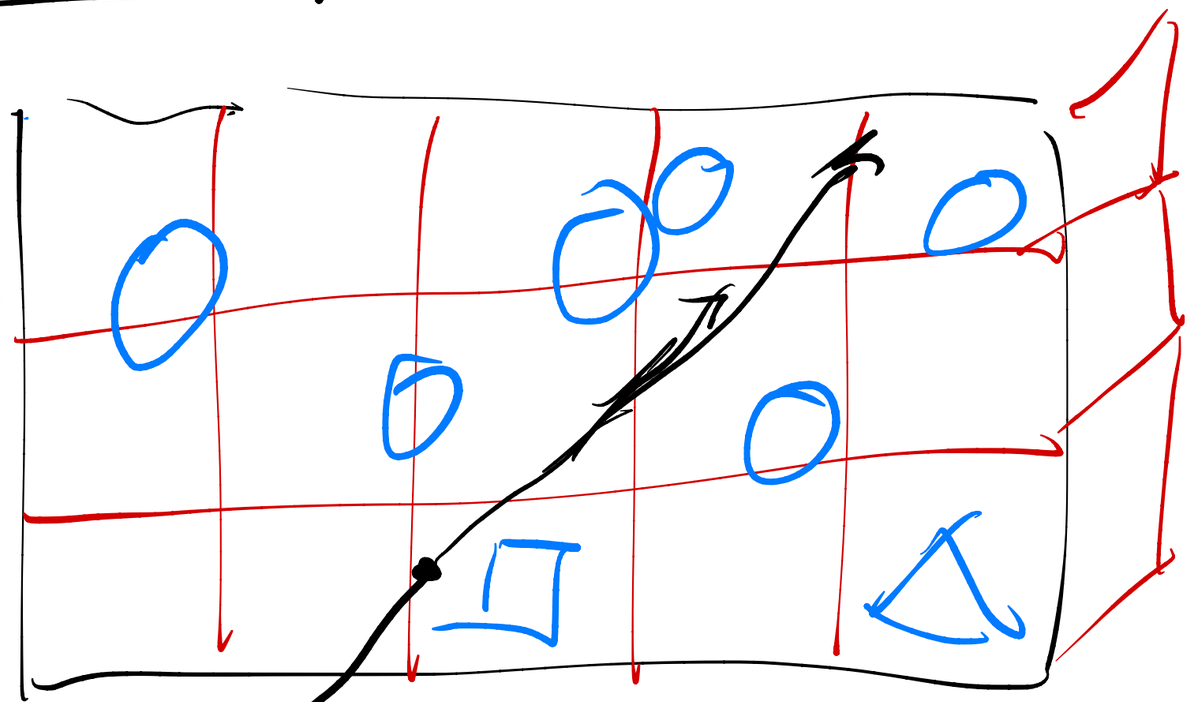


top-down



Grids - Uniform Spatial Partition

each cell has list of objects in it



traverse cells (all same size) using alg similar to rasterization

- good for lots of similarly sized small objects

kD-tree → book calls this axis-aligned BSP-trees

Idea: use planes to split space into pieces