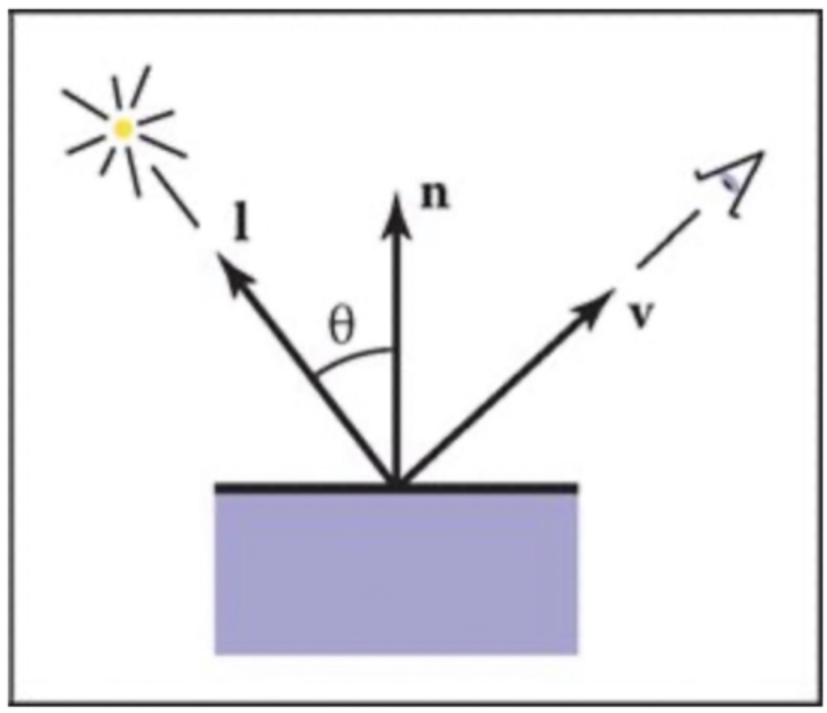
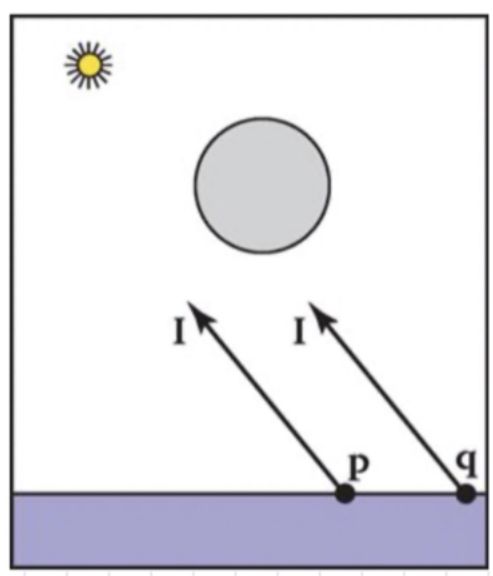
Shadows, Shading, Mirrors (& intro to speeding up raytracing)

4/9

Recall light in a shading model (Lambertian shading)





cast a ray from the point to shade to a light source

check if object in the way

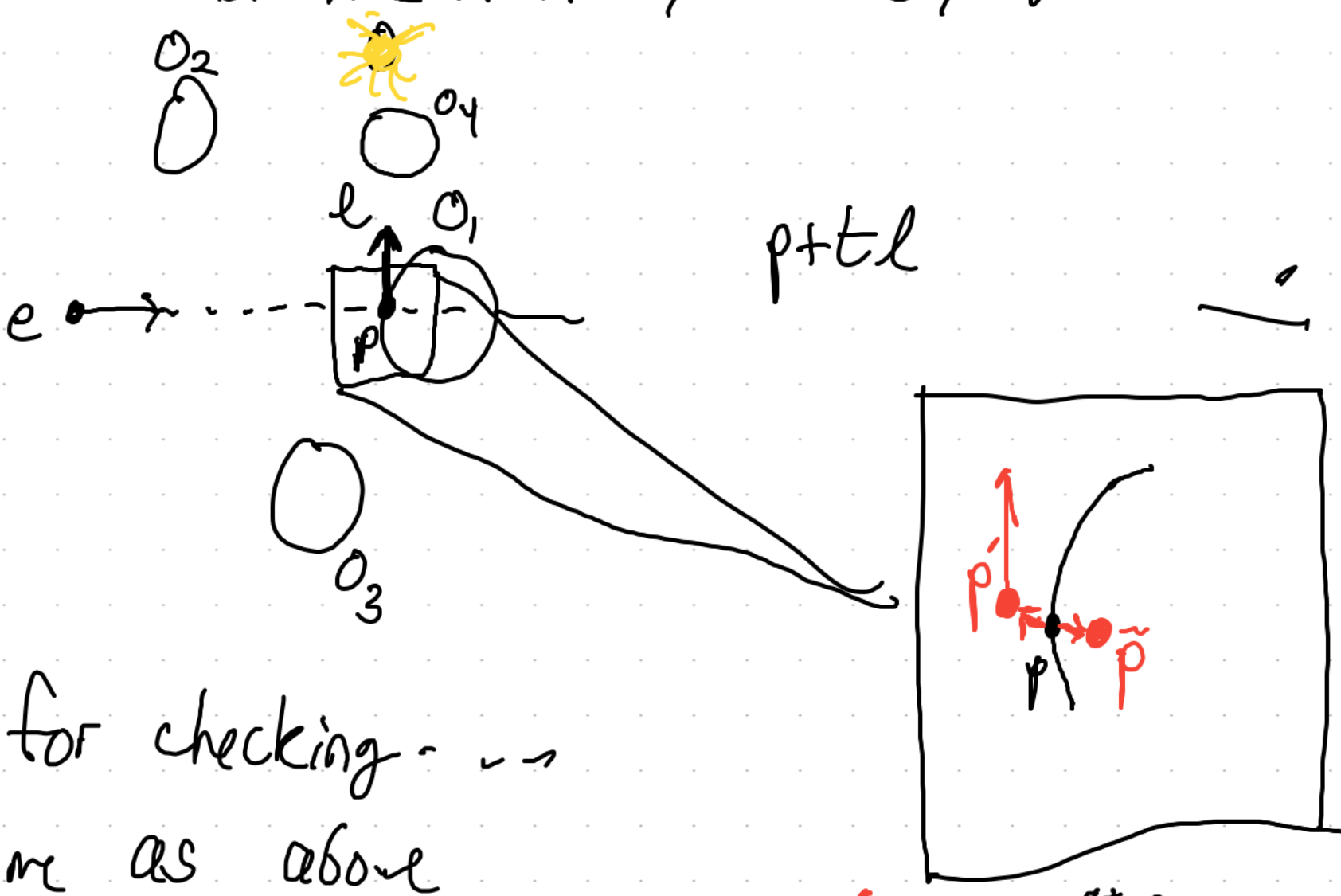
Try 1 for checking if a point is in a shadow

① let ℓ be the direction of the light

② let p be the intersection point of the ray and the object (the point to check)

③ form the ray $p + t\ell$

④ run FINDINTERSECTION w/ $t = [0, \infty)$



Try 2 for checking ---→

① same as above

②

③

④ run FINDINTERSECTION w/ $t = [\varepsilon, \infty)$

$\varepsilon > 0$ (some small const) 10^{-4}

$p' + t\ell$ 1st intersect w/ O_4

$\tilde{p} + t\ell$ 1st intersect w/ O_1

Recall Phong shading

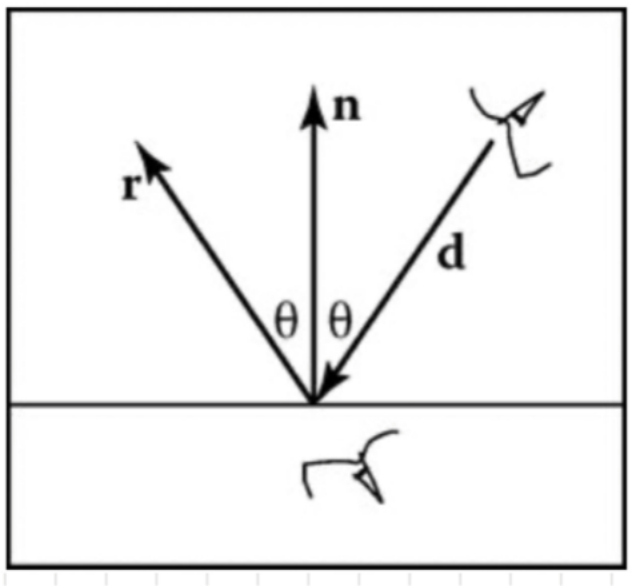
```
function raycolor( ray e + td, real t0, real t1 )
hit-record rec, srec
if (scene→hit(e + td, t0, t1, rec)) then
    p = e + (rec.t) d
    color c = rec.ka Ia
    vector3 h = normalized(normalized(l) + normalized(−d))
    c = c + rec.kd I max (0, rec.n · l) + (rec.ks) I (rec.n · h)^rec.p
    return c
else
    return background-color
```

we can update Phong model w/ our new test

```
function raycolor( ray e + td, real t0, real t1 )
hit-record rec, srec
if (scene→hit(e + td, t0, t1, rec)) then
    p = e + (rec.t) d
    color c = rec.ka Ia
    if (not scene→hit(p + sl, ϵ, ∞, srec)) then
        vector3 h = normalized(normalized(l) + normalized(−d))
        c = c + rec.kd I max (0, rec.n · l) + (rec.ks) I (rec.n · h)^rec.p
    return c
else
    return background-color
```

Idealized Specular Reflection (mirror reflection)

Key observation for specular reflection



recall from Phong Shading lecture

$$r = d - 2(d \cdot n)n$$

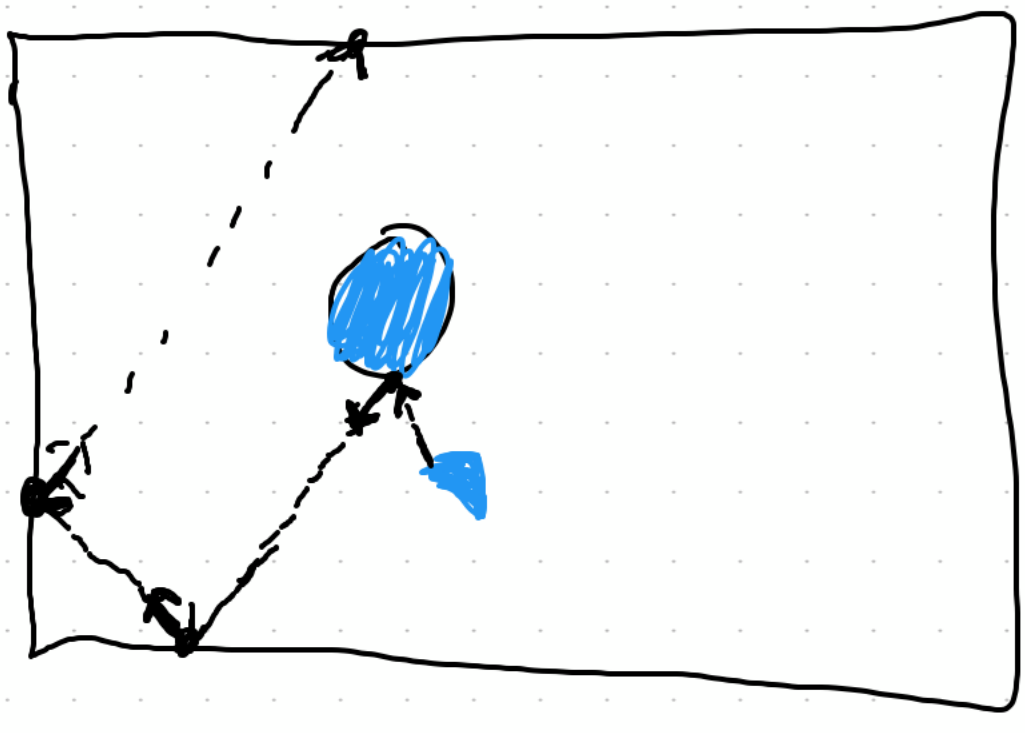
mirror reflection param k_m
"mirror reflectance"

point to shade p

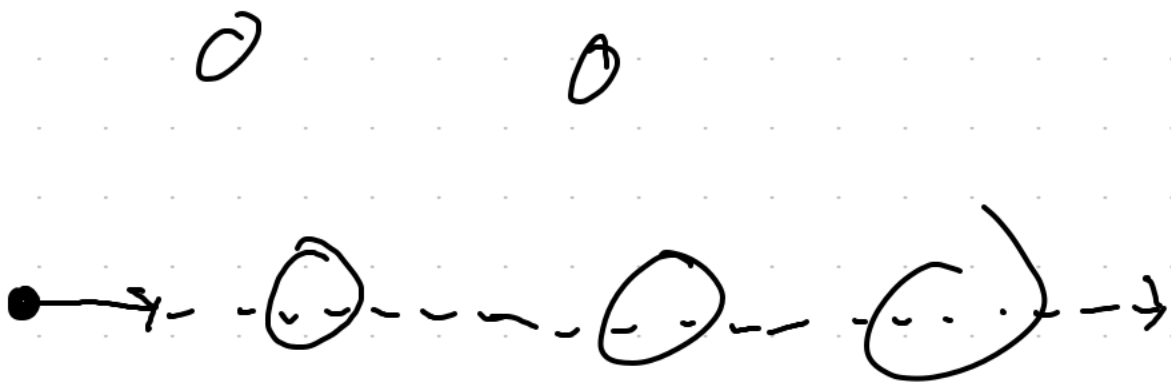
w/ color at point p

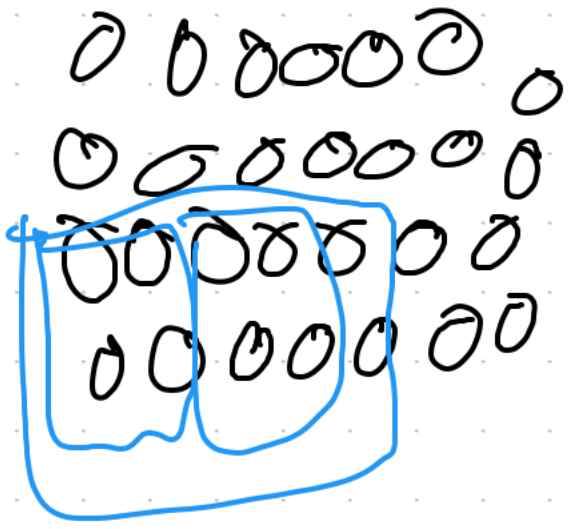
set color at p as

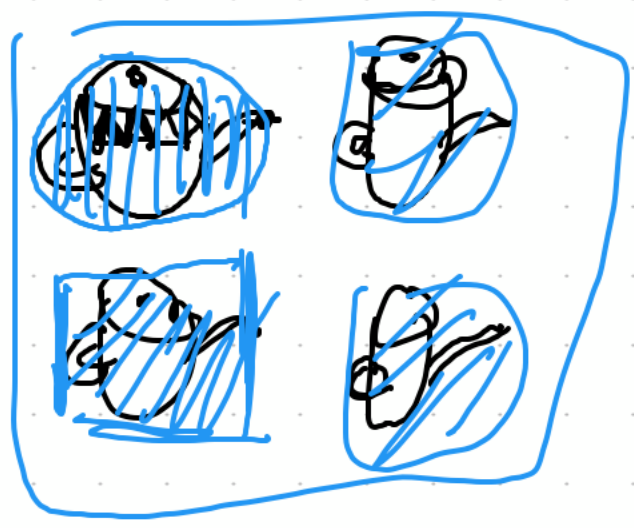
$$C + k_m \, \text{raycolor}(\vec{p}+s\vec{r}, \varepsilon, \infty)$$



usually we stop recursion after a max depth







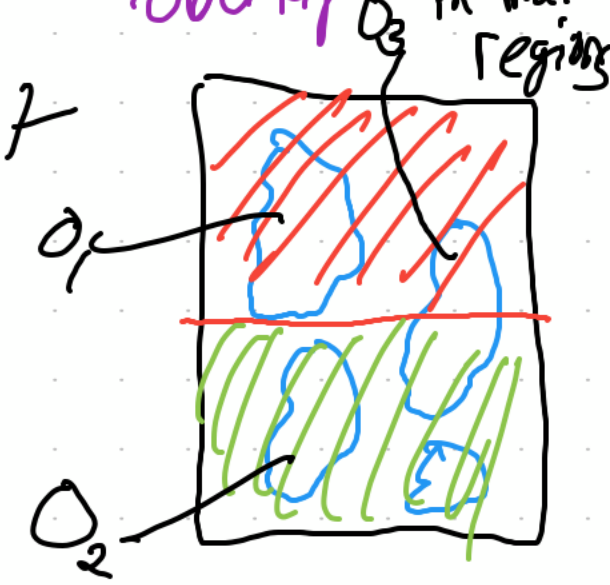
2 major way to think about "Spatial data structures"

Object partitioning scheme

- take objects and divide into disjoint groups
- may have overlap in space

Space partitioning scheme

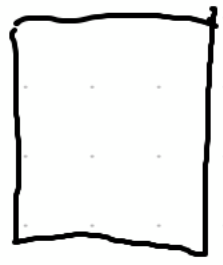
- space partitioned into disjoint regions
- may have an obj in more than one region



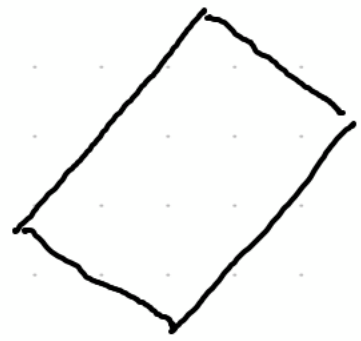
Bounding Boxes (Axis Aligned Bonding Boxes)

AABB

AABB



non-AABB



Why:

1. Less objects to test
2. Only need to test if there is an intersection, not where

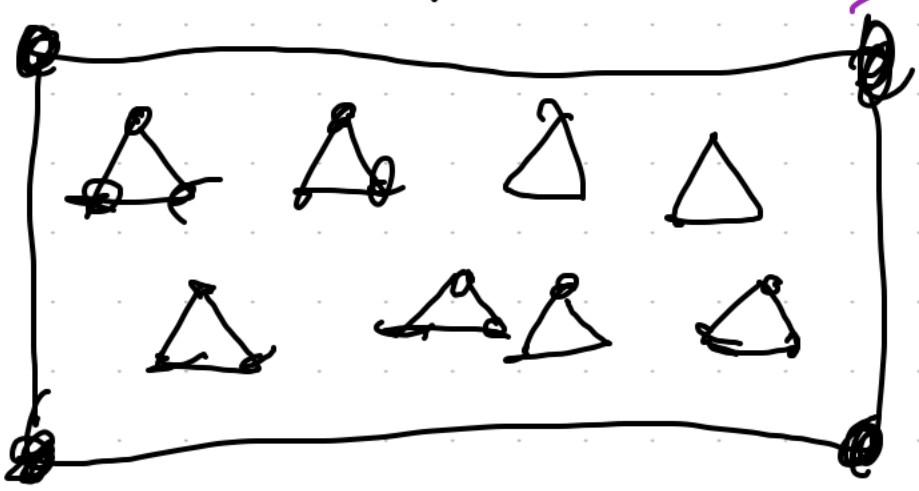
2D warmup

2D AABB rep

$x_{min} \leq x \leq x_{max}$

&

$y_{min} \leq y \leq y_{max}$



$(x,y) \in \mathbb{R}^2 \ni$

$(x,y) \in [x_{min}, x_{max}] \times [y_{min}, y_{max}]$