Computer Graphics

Lecture 8

Ray-Sphere Intersection

Computer Graphics

Lecture 8

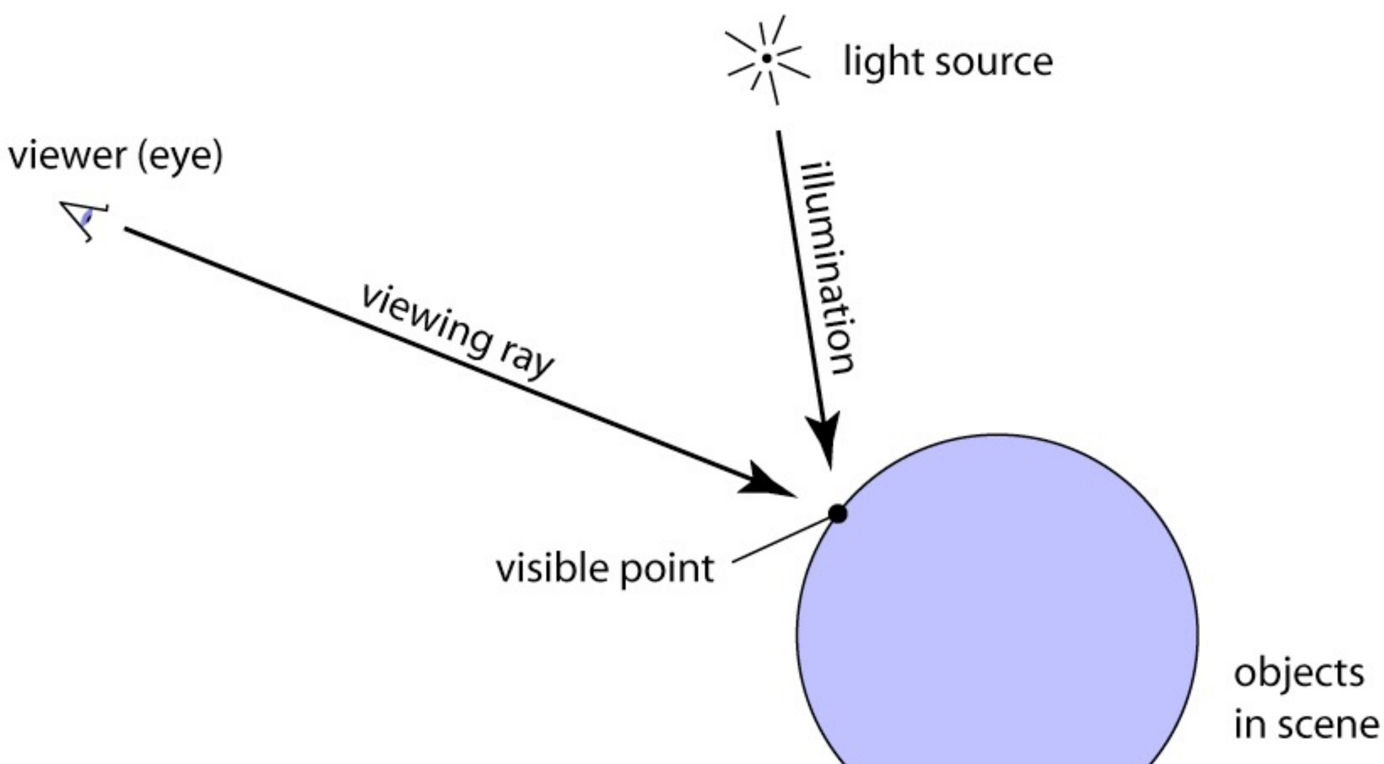
Ray-Sphere Intersection

Announcements

- A1 is done in pairs - if you don't have a partner yet, let's do some pairing at the end of class.
  - Partners must be in the same section (480 or 580)

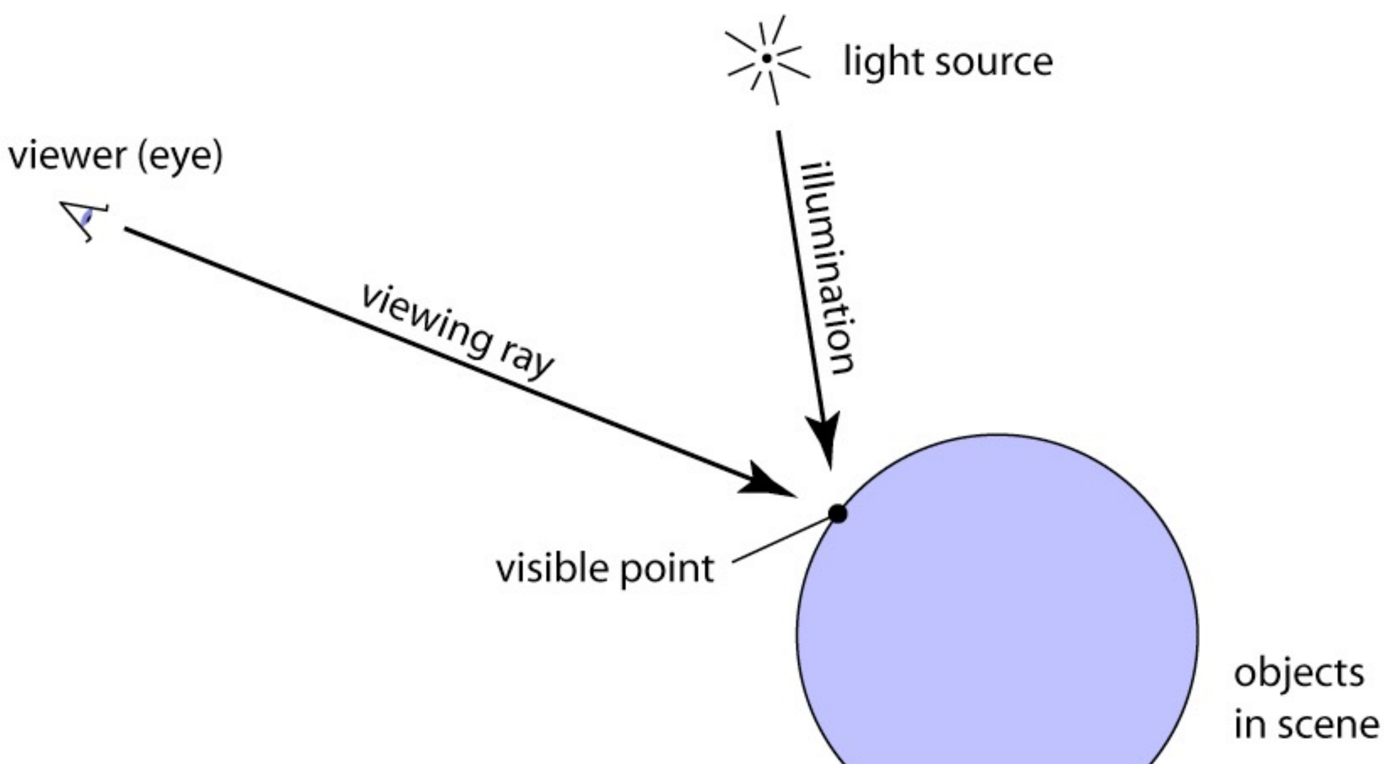
Ray Tracing: Pseudocode

```
for each pixel:
    generate a viewing ray for the pixel
    find the closest object it intersects
    determine the color of the object
```



Ray Tracing: Pseudocode

```
for each pixel:
    generate a viewing ray for the pixel
    find the closest object it intersects
    determine the color of the object
```



Reminder: Implicit vs Parametric

- Implicit equations: a property true at all points
  - e.g., $ax + by + c = 0$ for a line
- Parametric equations: use a free parameter variable to *generate* all points:
  - e.g., $\mathbf{r}(t) = \mathbf{p} + t\mathbf{d}$, for a line

Ray-Sphere Intersection

Ray (parametric): $\mathbf{p} + t\mathbf{d} = \begin{bmatrix} p_x + td_x \\ p_y + td_y \\ p_z + td_z \end{bmatrix}$

Sphere (parametric): $\begin{bmatrix} \cos\theta \sin\phi \\ \sin\theta \\ \cos\theta \cos\phi \end{bmatrix}$

In principle: set these equal and solve for t, θ, ϕ

In practice: math is cleanest when intersecting **implicit** with **parametric**.

Ray-Sphere Intuition: Geometric

Ponder:

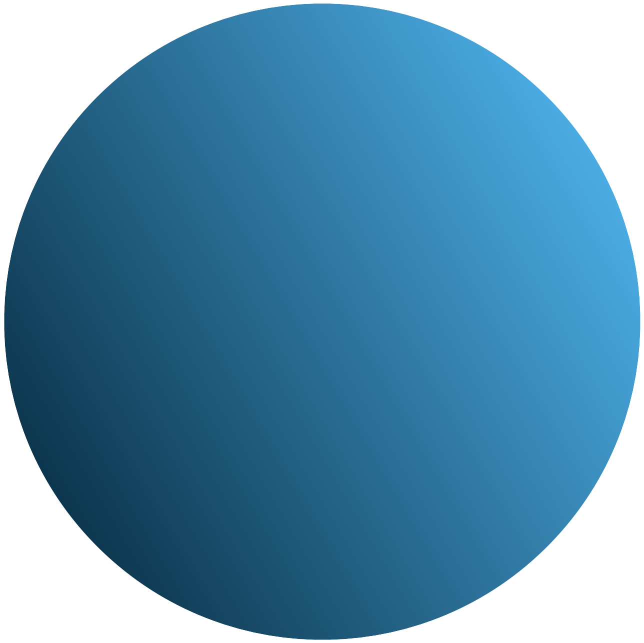
1. How many times might a ray intersect a sphere? What are the possibilities?

2. What's an implicit equation for a sphere? or: What's true of all points on a sphere?

- For now, assume a unit sphere at the origin.

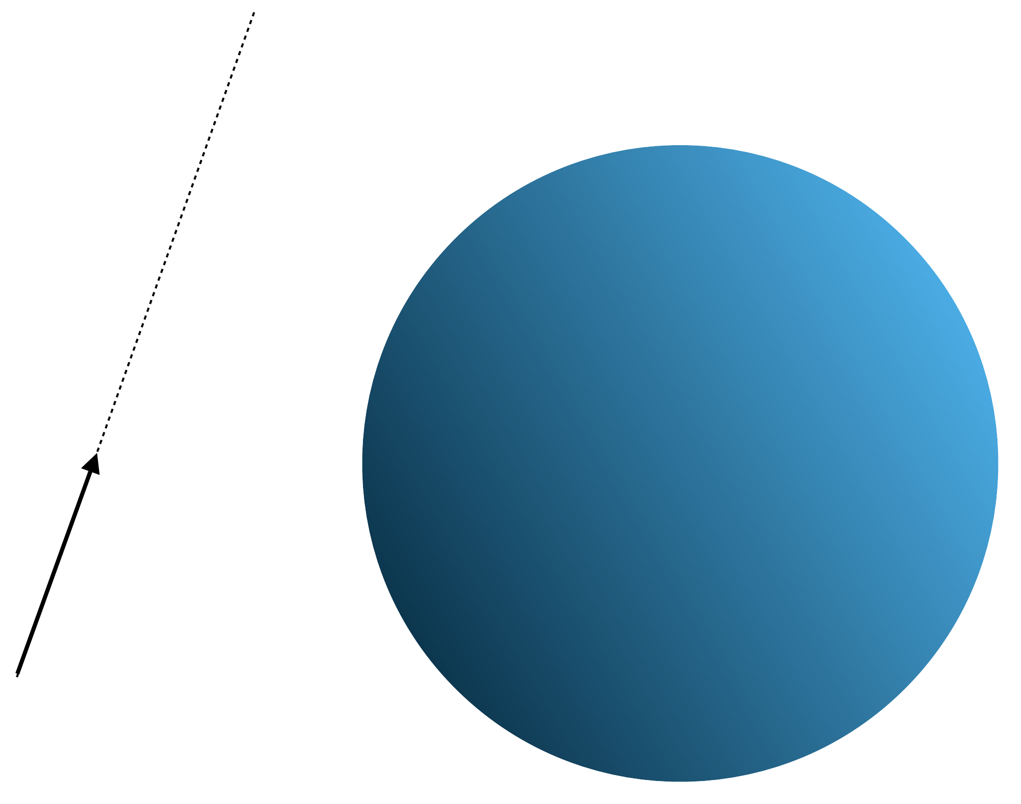
Ray-Sphere Intuition: Geometric

How many times can ray intersect a sphere?



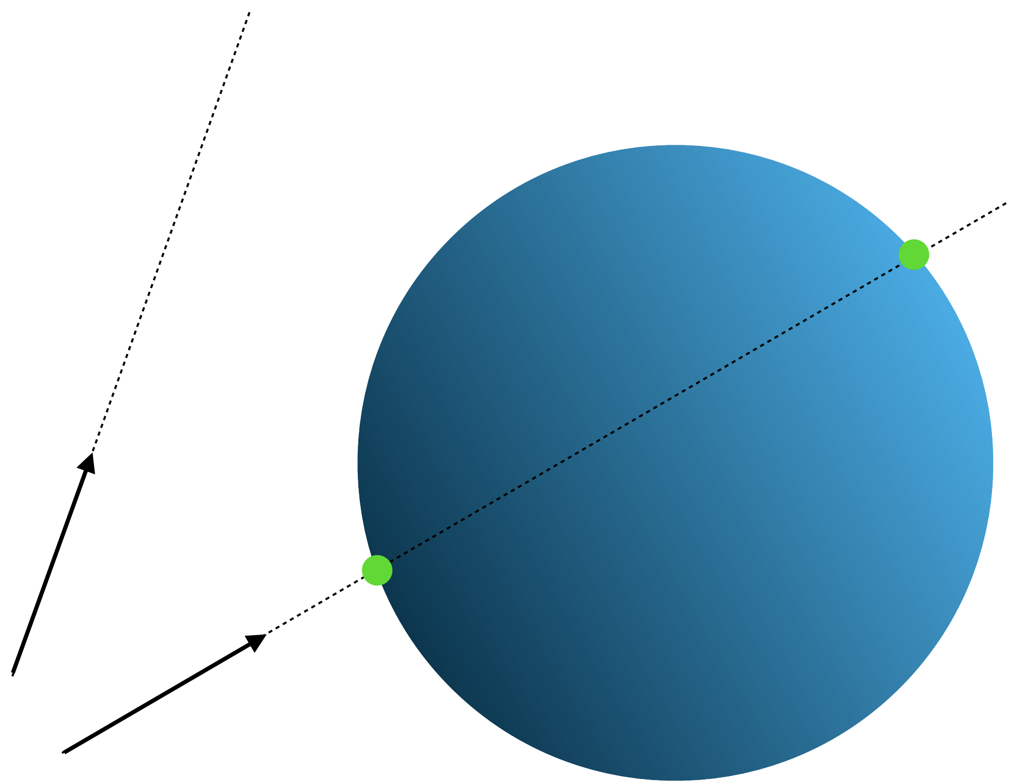
Ray-Sphere Intuition: Geometric

How many times can ray intersect a sphere?



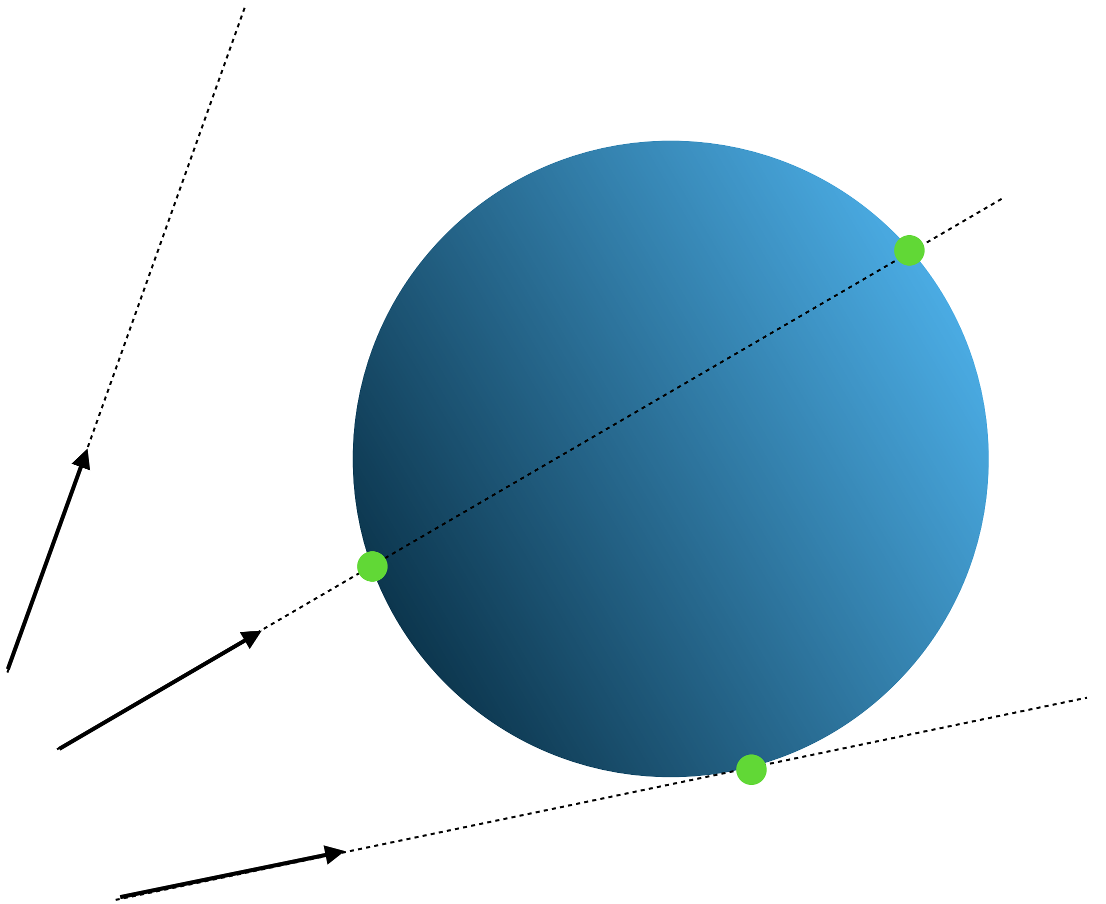
Ray-Sphere Intuition: Geometric

How many times can ray intersect a sphere?



Ray-Sphere Intuition: Geometric

How many times can ray intersect a sphere?



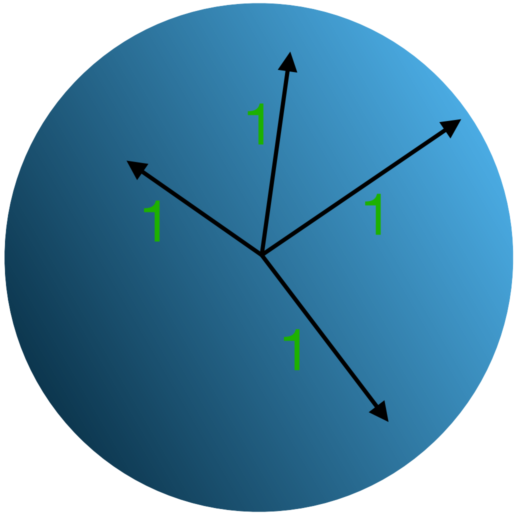
Ray-Sphere Intuition: Geometric

1. How many times can ray intersect a sphere? 0, 1, or 2.

2. What's an implicit equation for a sphere?
or: What's true of all points on a sphere?

They're all equidistant from the center.

For a unit sphere at the origin,
they're all distance 1 from (0, 0, 0)



Ray-Sphere Intuition: Geometric

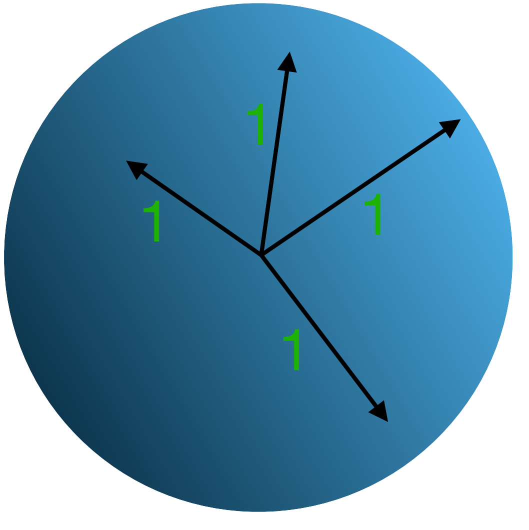
1. How many times can ray intersect a sphere? 0, 1, or 2.

2. What's an implicit equation for a sphere? or: What's true of all points on a sphere?

They're all equidistant from the center.

For a unit sphere at the origin,
they're all distance 1 from (0, 0, 0)

$$\sqrt{x^2 + y^2 + z^2} = 1$$



Ray-Sphere Intuition: Geometric

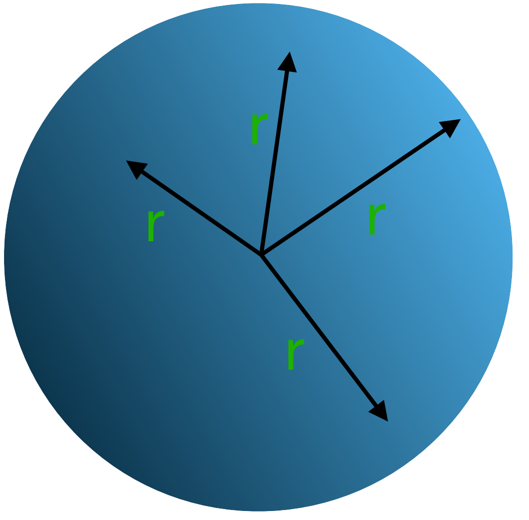
1. How many times can ray intersect a sphere? 0, 1, or 2.

2. What's an implicit equation for a sphere? or: What's true of all points on a sphere?

They're all equidistant from the center.

For **any** sphere at the origin, they're all distance r from (0, 0, 0)

$$\sqrt{x^2 + y^2 + z^2} = r$$



Ray-Sphere Intersection: Algebraic

$\vec{a} \cdot \vec{a} = \vec{a}^T \vec{a}$

Ray: $\vec{p} + t\vec{d}$

Sphere: $\sqrt{x^2 + y^2 + z^2} = 1$

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

$x^2 + y^2 + z^2 - 1 = 0$

Let $\vec{a} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$. Then:

$xx + yy + zz - 1 = 0$

$\vec{a} \cdot \vec{a} - 1 = 0$

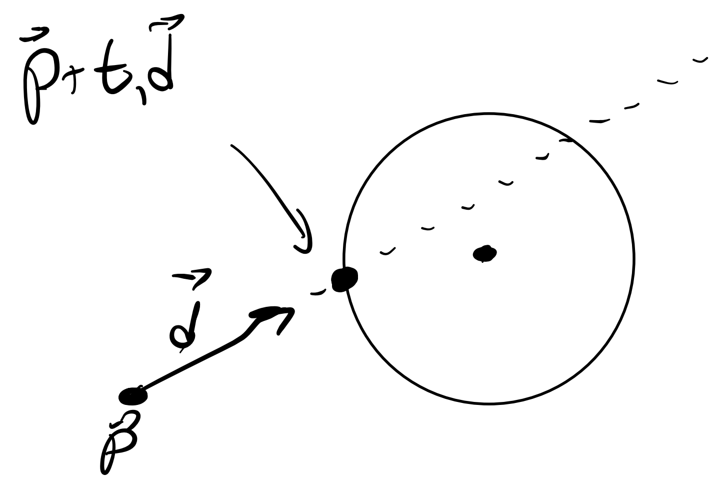
Substitute:

$(p + td)^T (p + td) - 1 = 0$

LOTF:

$d \cdot d\, t^2 + (2p \cdot d)t + p \cdot p - 1 = 0$

$A t^2 + Bt + C = 0$



$$t = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$t = \frac{-d \cdot p \pm \sqrt{(d \cdot p)^2 - (d \cdot d)(p \cdot p - 1)}}{d \cdot d}$$

$$p + t\vec{d}$$

Number of Intersections

$$t = \frac{-\mathbf{d} \cdot \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \mathbf{p})^2 - (\mathbf{d} \cdot \mathbf{d})(\mathbf{p} \cdot \mathbf{p} - 1)}}{\mathbf{d} \cdot \mathbf{d}}$$

Given only **d** and **p**, how can you tell how many intersections the ray has with the sphere?

Ray-Sphere intersection

For now, assume unit sphere centered at the origin. See 4.4.1 for general derivation.

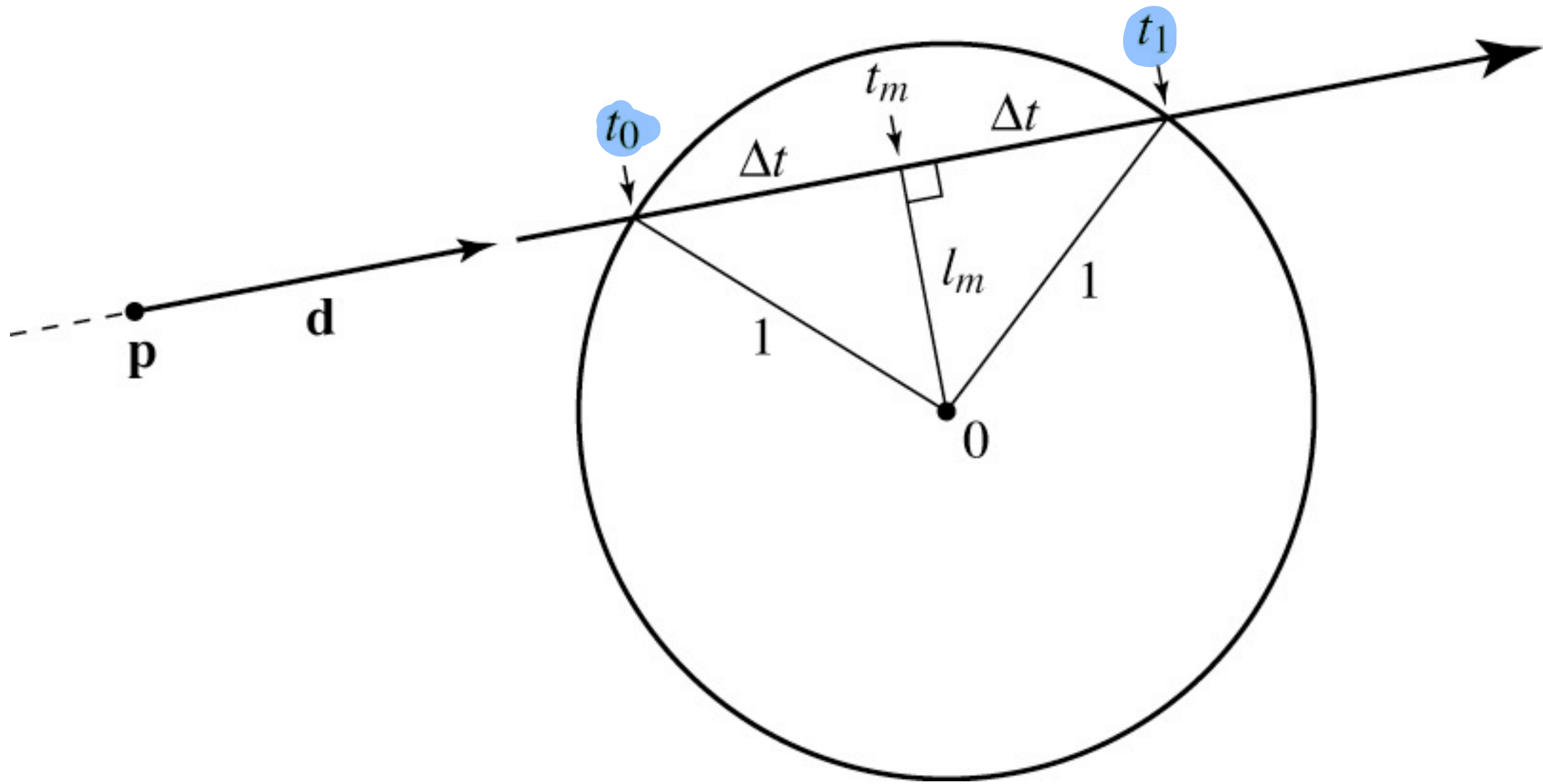
$$t = \frac{-\mathbf{d} \cdot \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \mathbf{p})^2 - (\mathbf{d} \cdot \mathbf{d})(\mathbf{p} \cdot \mathbf{p} - 1)}}{\mathbf{d} \cdot \mathbf{d}}$$

If **d** were normalized to unit-length, this reduces to:

$$t = -\mathbf{d} \cdot \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \mathbf{p})^2 - \mathbf{p} \cdot \mathbf{p} + 1}$$

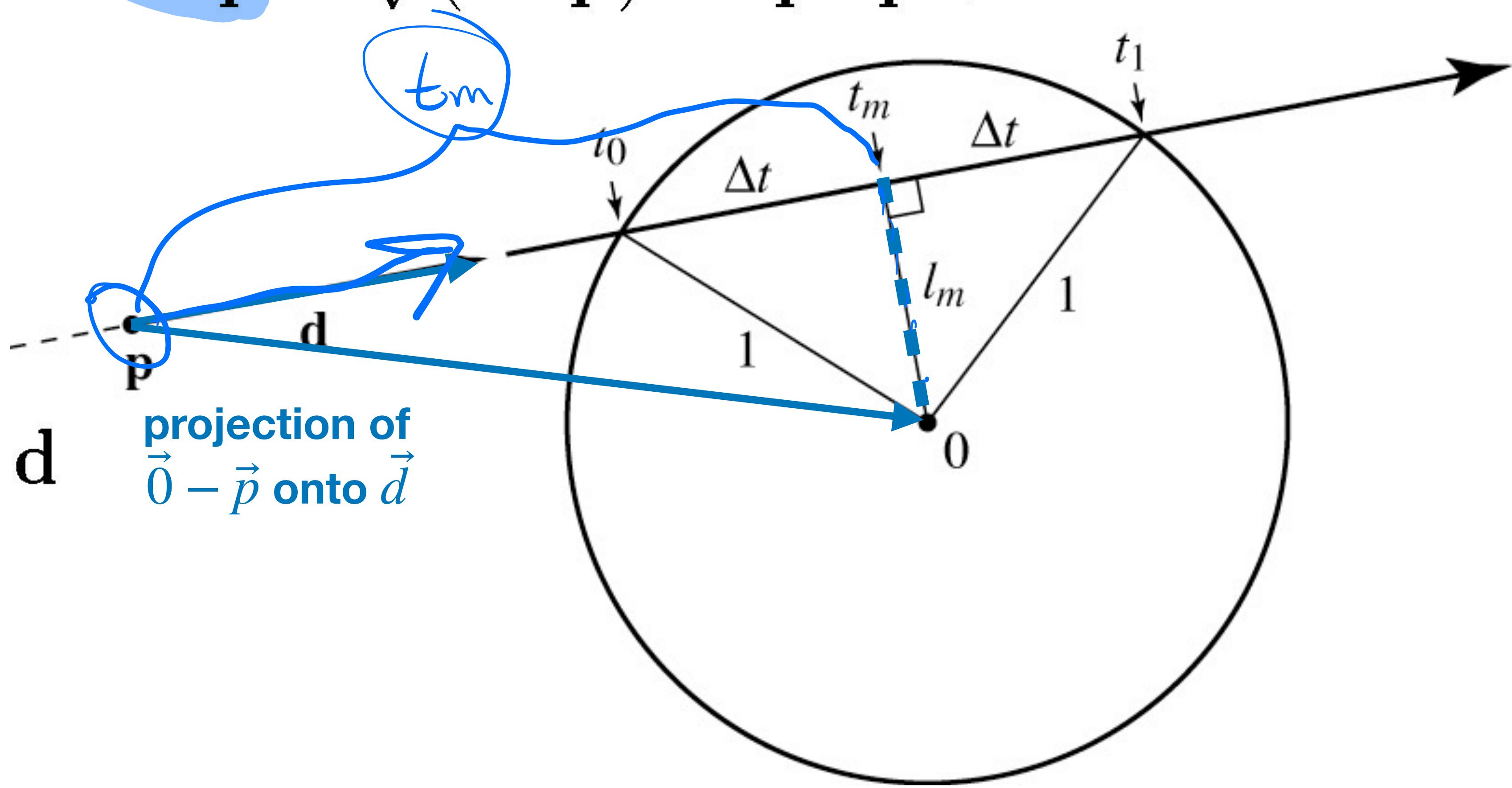
Geometric Intuition

$$t = -\mathbf{d} \cdot \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \mathbf{p})^2 - \mathbf{p} \cdot \mathbf{p} + 1}$$



Geometric Intuition

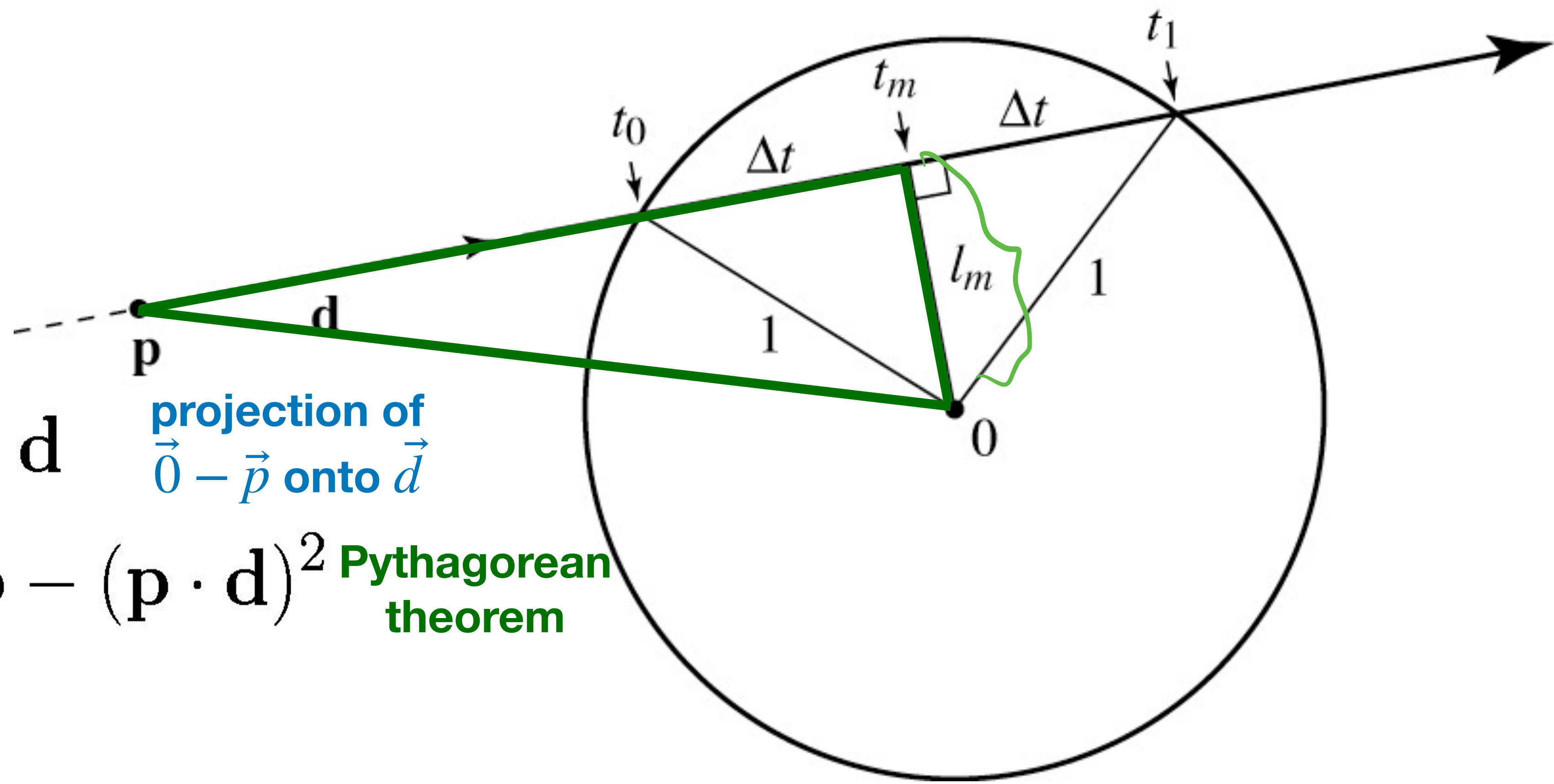
$$t = -\mathbf{d} \cdot \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \mathbf{p})^2 - \mathbf{p} \cdot \mathbf{p} + 1}$$



$$t_m = -\mathbf{p} \cdot \mathbf{d}$$

Geometric Intuition

$$t = -\mathbf{d}\cdot\mathbf{p} \pm \sqrt{(\mathbf{d}\cdot\mathbf{p})^2 - \mathbf{p}\cdot\mathbf{p} + 1}$$



$$t_m = -\mathbf{p}\cdot\mathbf{d}$$

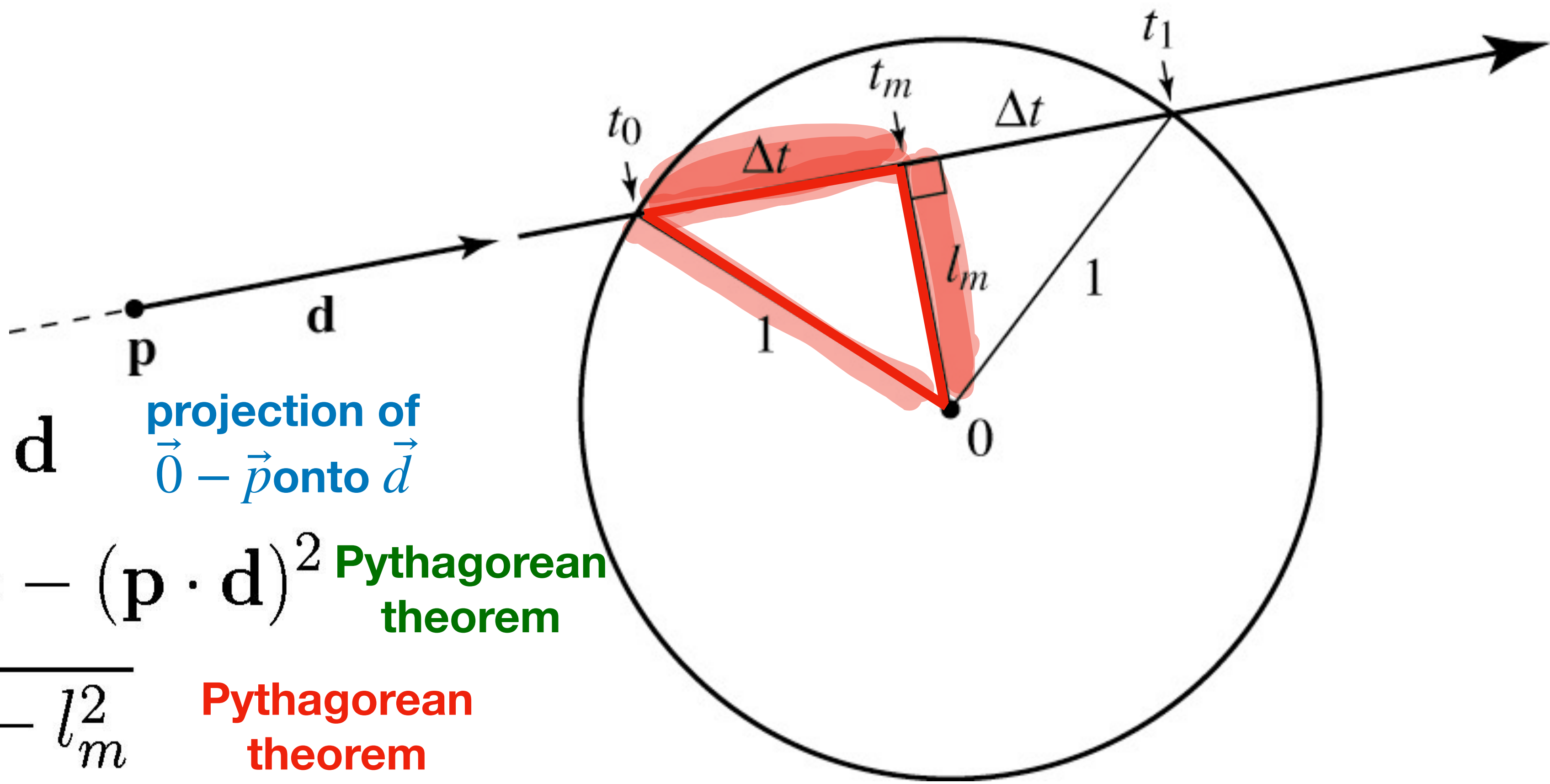
projection of $\vec{0} - \vec{p}$ onto $\vec{d}$

$$l_m^2 = \mathbf{p}\cdot\mathbf{p} - (\mathbf{p}\cdot\mathbf{d})^2$$

Pythagorean theorem

Geometric Intuition

$$t = -\mathbf{d} \cdot \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \mathbf{p})^2 - \mathbf{p} \cdot \mathbf{p} + 1}$$



$$t_m = -\mathbf{p} \cdot \mathbf{d}$$ projection of $\vec{0} - \vec{p}$ onto $\vec{d}$

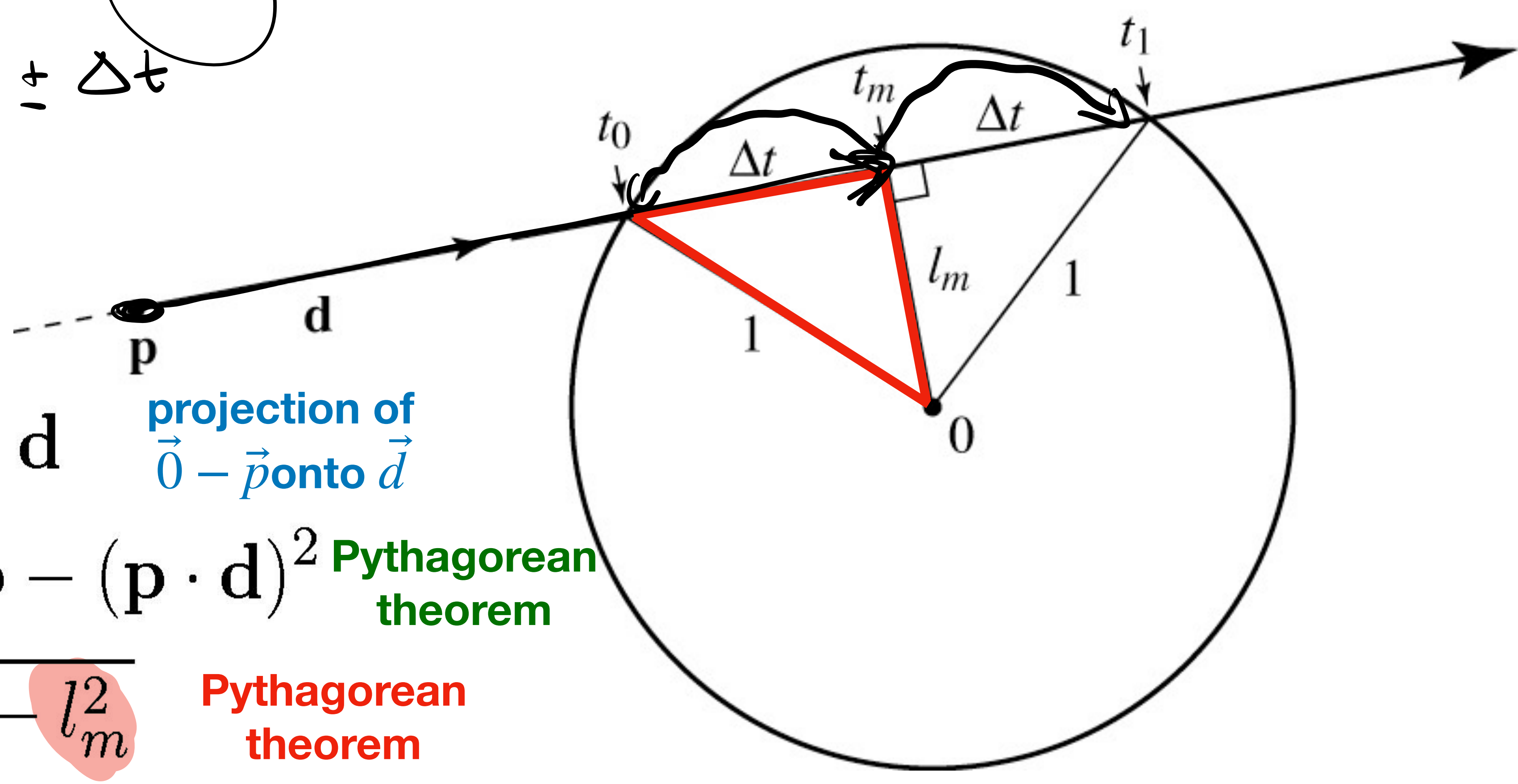
$$l_m^2 = \mathbf{p} \cdot \mathbf{p} - (\mathbf{p} \cdot \mathbf{d})^2$$ Pythagorean theorem

$$\Delta t = \sqrt{1 - l_m^2}$$ Pythagorean theorem

Geometric Intuition

$$t = -\mathbf{d}\cdot\mathbf{p} \pm \sqrt{(\mathbf{d}\cdot\mathbf{p})^2 - \mathbf{p}\cdot\mathbf{p} + 1}$$

$t = t_m \pm \Delta t$



$t_m = -\mathbf{p}\cdot\mathbf{d}$ projection of $\vec{0} - \vec{p}$ onto $\vec{d}$

$l_m^2 = \mathbf{p}\cdot\mathbf{p} - (\mathbf{p}\cdot\mathbf{d})^2$ Pythagorean theorem

$\Delta t = \sqrt{1 - l_m^2}$ Pythagorean theorem

$= \sqrt{(\mathbf{p}\cdot\mathbf{d})^2 - \mathbf{p}\cdot\mathbf{p} + 1}$ plug and chug

$$t_{0,1} = t_m \pm \Delta t = -\mathbf{p}\cdot\mathbf{d} \pm \sqrt{(\mathbf{p}\cdot\mathbf{d})^2 - \mathbf{p}\cdot\mathbf{p} + 1}$$

Ray-Sphere: Code Sketch

```
function ray_intersect(ray, sphere, tmin, tmax):
```

0 ∞

ε

- Use above math to find +/- *t*
- If none, return `nothing`
- Otherwise, return closest *t* that lies between `tmin` and `tmax`

Ray-Scene: Code Sketch

Brute force: check all objects.
There are better ways - more on this later.

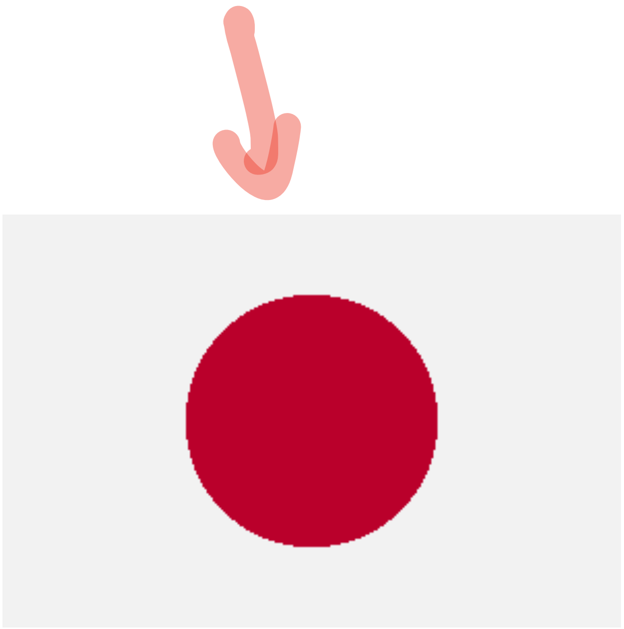
```
find_intersection(ray, scene):
  closest_t = Inf
  closest_obj = nothing
  for obj in scene:
    t = ray_intersect(ray, obj, 1, closest_t)
    if obj != nothing:
      closest_t = t
      closest_obj = surf
  return closest_t, closest_obj
```

Ray Tracing: Code Sketch

```
scene = model_scene()
for each pixel (i,j):
    ray = get_view_ray(i, j)
    t, obj = find_intersection(ray, scene)
    if obj != nothing:
      canvas[i,j] = obj.color
    else:
      canvas[i,j] = scene.bgcolor
```

Ray Tracing: Code Sketch

```
scene = model_scene()
for each pixel (i,j):
    ray = get_view_ray(i, j)
    t, obj = find_intersection(ray, scene)
    if obj != nothing:
      canvas[i,j] = obj.color
    else:
      canvas[i,j] = scene.bgcolor
```

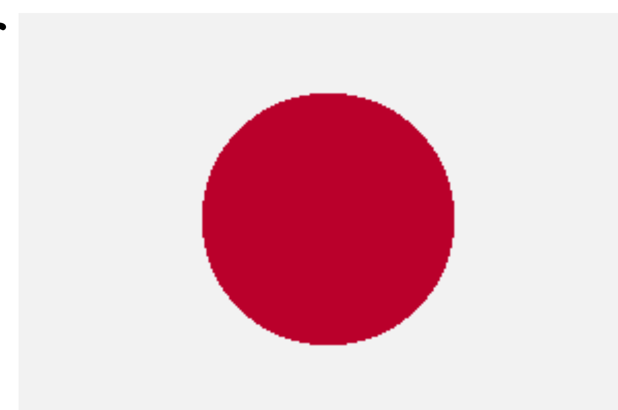


Next time...

```
scene = model_scene()
for each pixel (i,j):
    ray = get_view_ray(i, j)
    t, obj = find_intersection(ray, scene)
    if obj != nothing:
      canvas[i,j] = obj.color
    else:
      canvas[i,j] = scene.bgcolor
```

Next time...

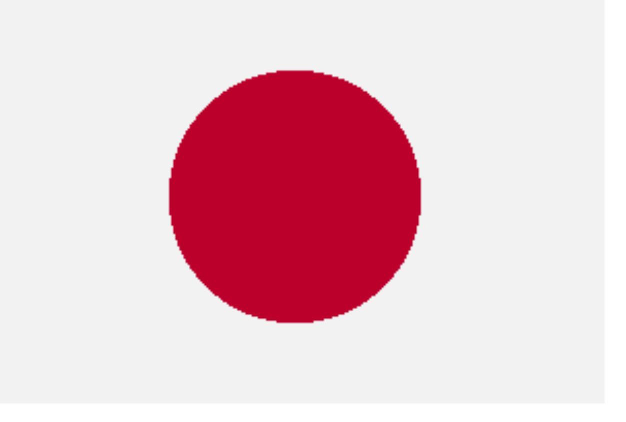
```
scene = model_scene()
for each pixel (i,j):
    ray = get_view_ray(i, j)
    t, obj = find_intersection(ray, scene)
    if obj != nothing:
      canvas[i,j] = obj.color
    else:
      canvas[i,j] = scene.bgcolor
```



Next time...

```
scene = model_scene()
for each pixel (i,j):
    ray = get_view_ray(i, j)
    t, obj = find_intersection(ray, scene)
    if obj != nothing:
        canvas[i,j] = obj.color
    else:
        canvas[i,j] = scene.bgcolor
```

Let's work on this.



Problems

- Write ray intersection code for axis-aligned rectangles.
- Model an empty Cornell box.