Ray Intersection

Steve Marschner
CS 4620
Cornell University

Ray Intersection

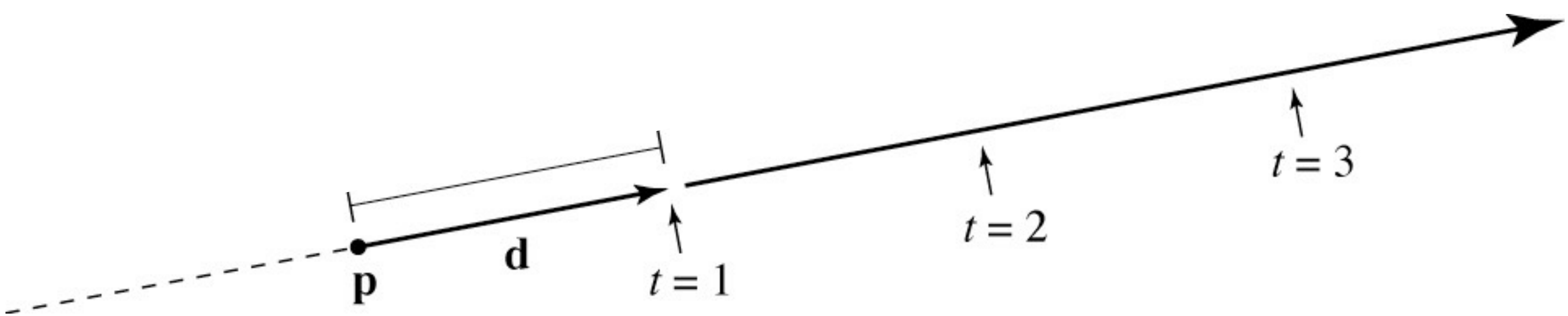
1. Ray-sphere intersection

Ray: a half line

- **Standard representation: origin point $\mathbf{p}$ and direction $\mathbf{d}$**

$$\mathbf{r}(t) = \mathbf{p} + t\mathbf{d}$$

  - this is a ***parametric equation*** for the line
  - lets us directly generate the points on the line
  - if we restrict to $t > 0$ then we have a ray
  - note replacing $\mathbf{d}$ with $\alpha\mathbf{d}$ doesn't change ray (for $\alpha > 0$)



Ray-sphere intersection: algebraic

- **Condition 1: point is on ray**

$$\mathbf{r}(t) = \mathbf{p} + t\mathbf{d}$$

- **Condition 2: point is on sphere**
  – assume unit sphere; see book or notes for general

$$\|\mathbf{x}\| = 1 \Leftrightarrow \|\mathbf{x}\|^2 = 1$$

$$f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{x} - 1 = 0$$

- **Substitute:**

$$(\mathbf{p} + t\mathbf{d}) \cdot (\mathbf{p} + t\mathbf{d}) - 1 = 0$$

  – this is a quadratic equation in ***t***

Ray-sphere intersection: algebraic

- **Solution for t by quadratic formula:**

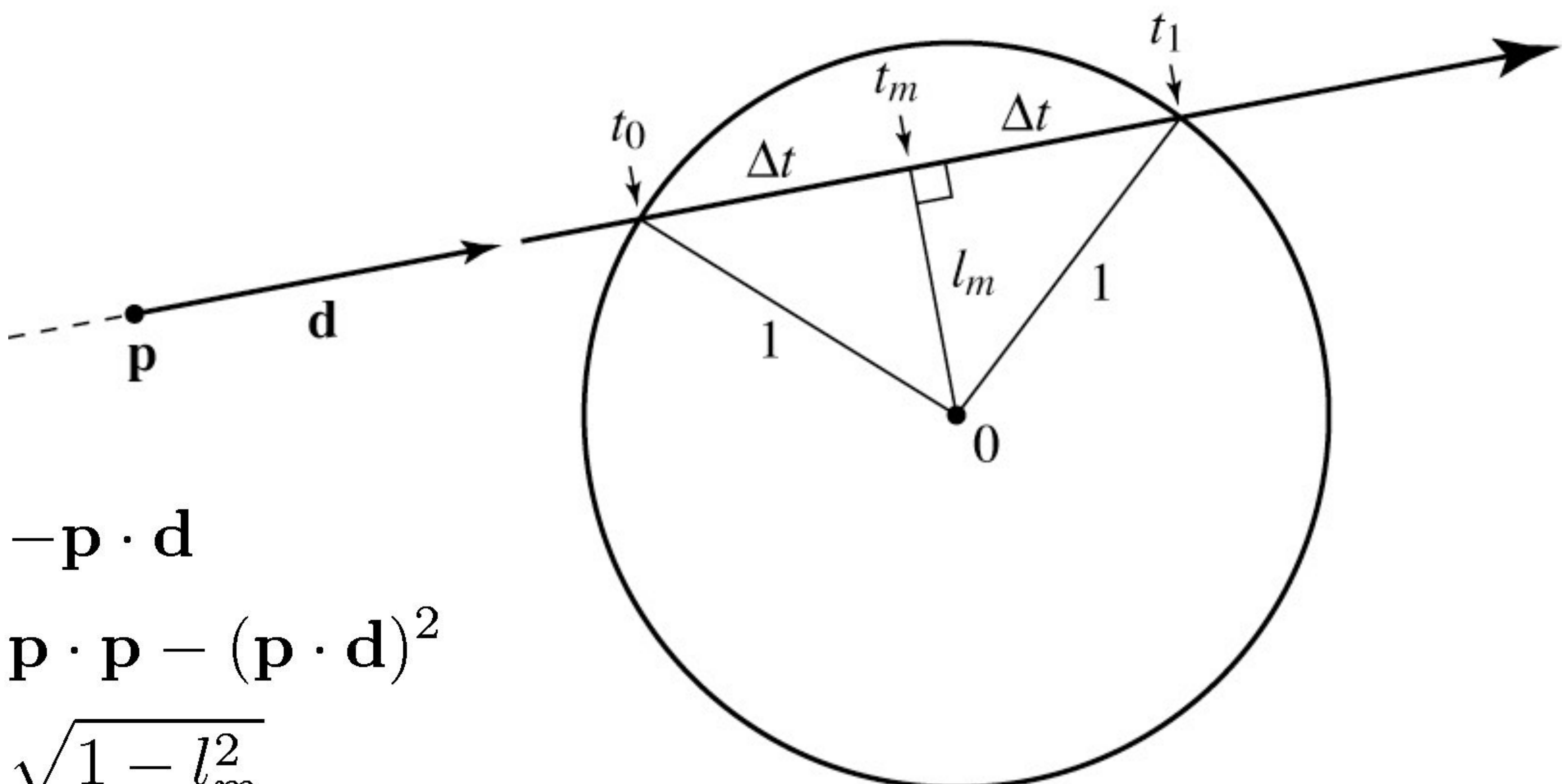
$$t = \frac{-\mathbf{d} \cdot \mathbf{p} \pm \sqrt{(\mathbf{d} \cdot \mathbf{p})^2 - (\mathbf{d} \cdot \mathbf{d})(\mathbf{p} \cdot \mathbf{p} - 1)}}{\mathbf{d} \cdot \mathbf{d}}$$

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

  – simpler form holds when **d** is a unit vector
  but we won't assume this in practice (reason later)

  – I'll use the unit-vector form to make the geometric interpretation

Ray-sphere intersection: geometric



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

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

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

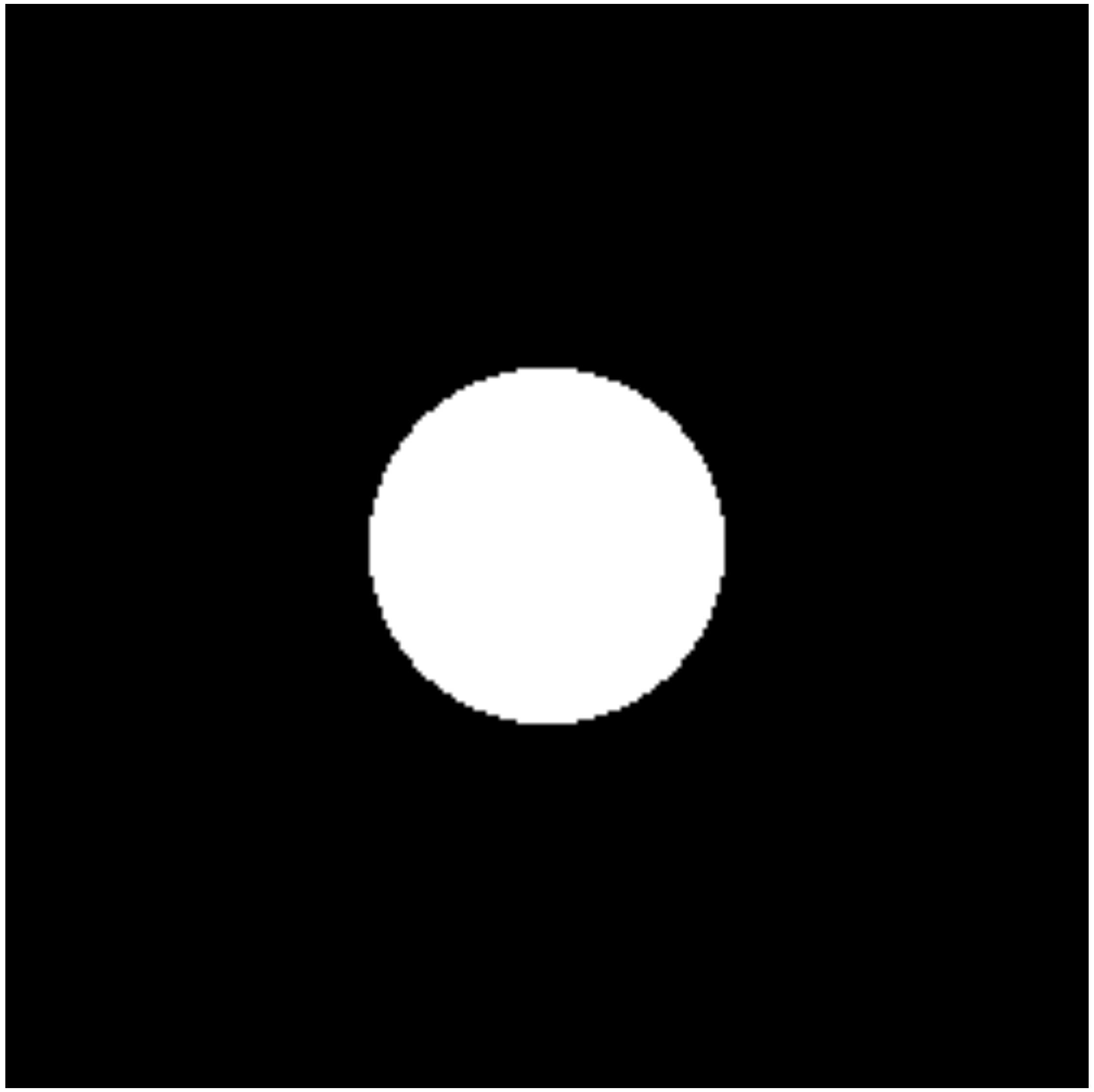
$$= \sqrt{(\mathbf{p} \cdot \mathbf{d})^2 - \mathbf{p} \cdot \mathbf{p} + 1}$$

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

Image so far

- **With eye ray generation and sphere intersection**

```
Surface s = new Sphere((0.0, 0.0, 0.0), 1.0);
for 0 <= iy < ny
    for 0 <= ix < nx {
        ray = camera.getRay(ix, iy);
        hitSurface, t = s.intersect(ray, 0, +inf)
        if hitSurface is not null
            image.set(ix, iy, white);
    }
```



Ray Intersection

2. Ray-triangle intersection

Barycentric coordinates

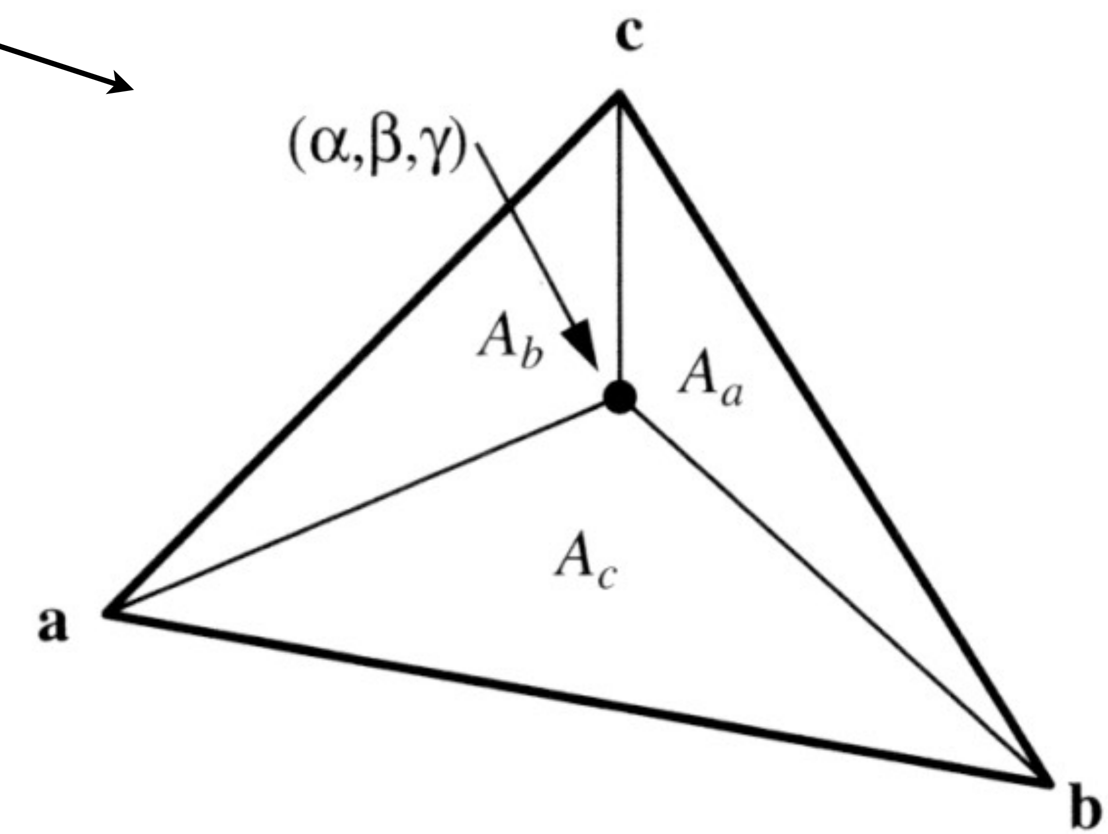
- **A coordinate system for triangles**
  - algebraic viewpoint:

$$\mathbf{p} = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c}$$

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

  - geometric viewpoint (areas):
- **Triangle interior test:**

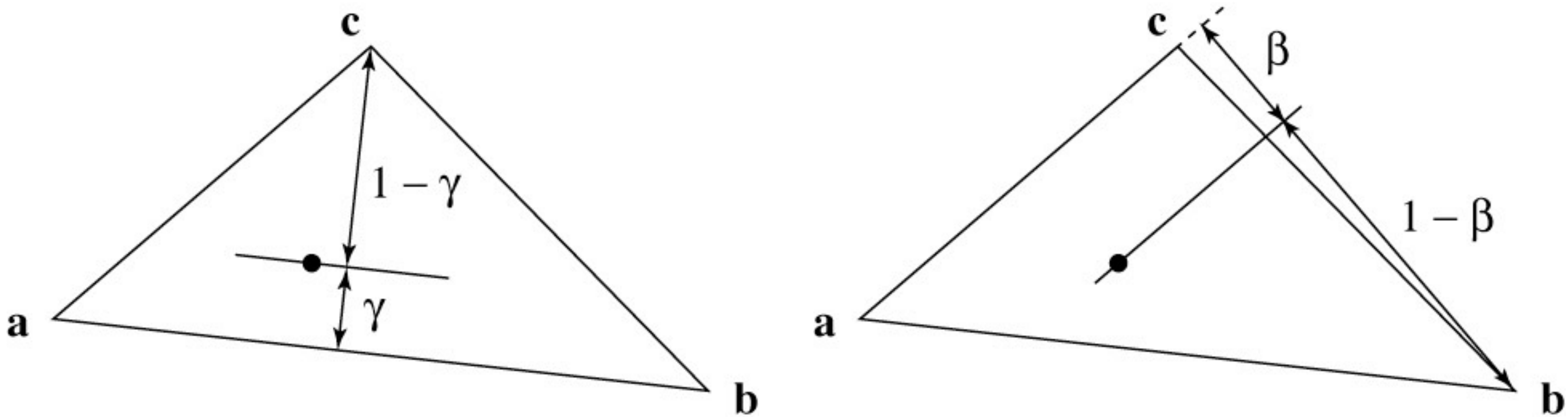
$$\alpha > 0; \quad \beta > 0; \quad \gamma > 0$$



[Shirley 2000]

Barycentric coordinates

- **A coordinate system for triangles**
  - geometric viewpoint: distances



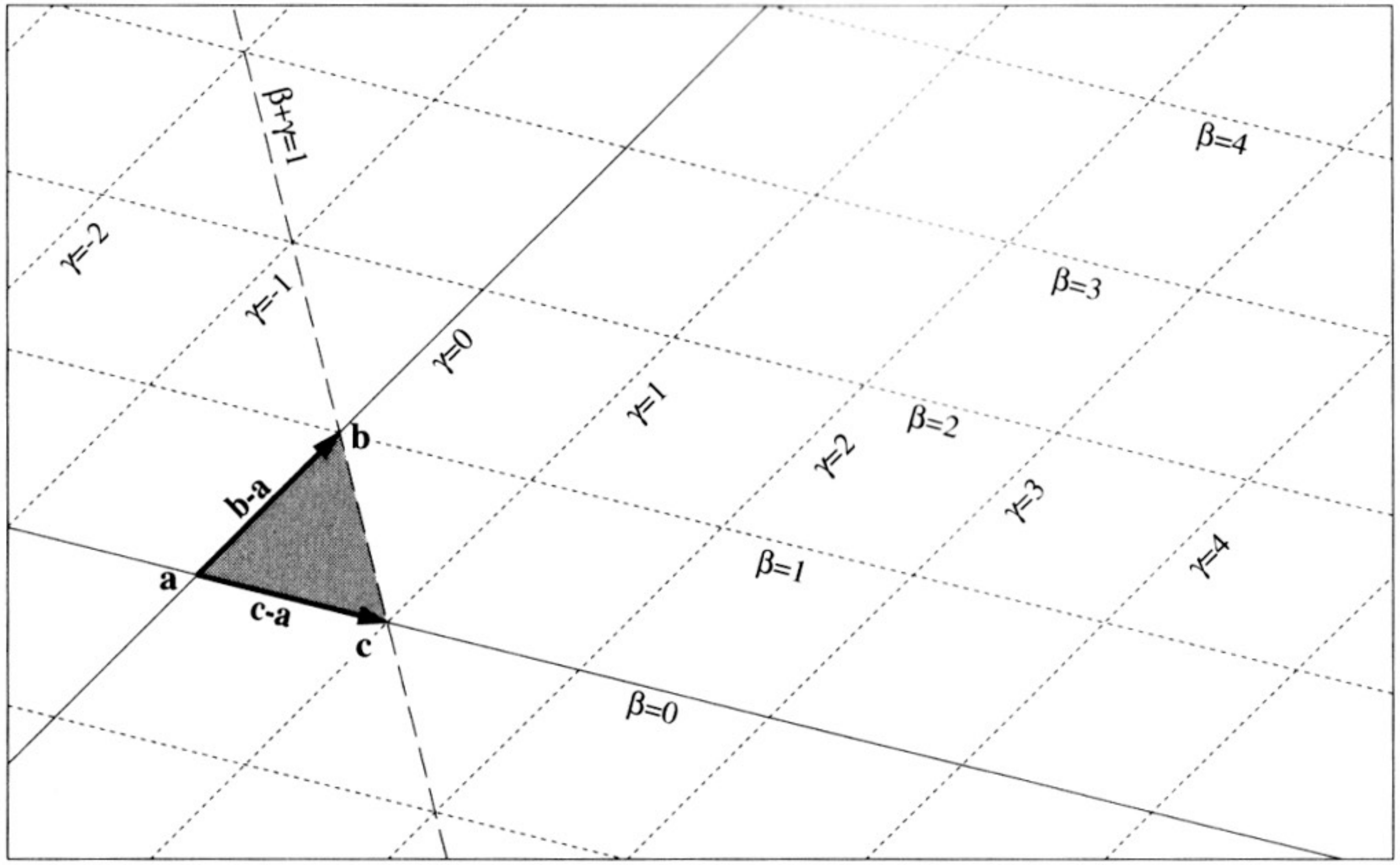
  - linear viewpoint: basis of edges

$$\alpha = 1 - \beta - \gamma$$

$$\mathbf{p} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

Barycentric coordinates

- **Linear viewpoint: basis for the plane**



[Shirley 2000]

  - in this view, the triangle interior test is just

$$\beta > 0; \quad \gamma > 0; \quad \beta + \gamma < 1$$

Barycentric ray-triangle intersection

- **Every point on the plane can be written in the form:**

  $$\mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

  for some numbers *β* and γ.

- **If the point is also on the ray then it is**

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

  for some number *t*.

- **Set them equal: 3 linear equations in 3 variables**

  $$\mathbf{p} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

  …solve them to get *t*, *β*, and γ all at once!

Barycentric ray-triangle intersection

$$\mathbf{p} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a})$$

$$\beta(\mathbf{a} - \mathbf{b}) + \gamma(\mathbf{a} - \mathbf{c}) + t\mathbf{d} = \mathbf{a} - \mathbf{p}$$

$$\begin{bmatrix} \mathbf{a} - \mathbf{b} & \mathbf{a} - \mathbf{c} & \mathbf{d} \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} \mathbf{a} - \mathbf{p} \end{bmatrix}$$

$$\begin{bmatrix} x_a - x_b & x_a - x_c & x_d \\ y_a - y_b & y_a - y_c & y_d \\ z_a - z_b & z_a - z_c & z_d \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} x_a - x_p \\ y_a - y_p \\ z_a - z_p \end{bmatrix}$$

Cramer's rule is a good fast way to solve this system

(see text Ch. 2 and Ch. 4 for details)

Ray Intersection

3. Ray intersection software

Rays

- **Rays are a useful datatype: pair origin with direction**
- **Also very useful to make rays into ray segments**
  - store a start (minimum t) and end (maximum t)
  - ray intersections only count if t is between start and end

```
class Ray {
   Point origin
   Vector direction
   float start
   float end;
}
```

Surfaces

- **All surfaces need to be able to intersect rays with themselves**
  - surfaces with multiple intersections must return the first one
  - convenient to return $t = +\infty$ when there is no intersection

ray to be intersected

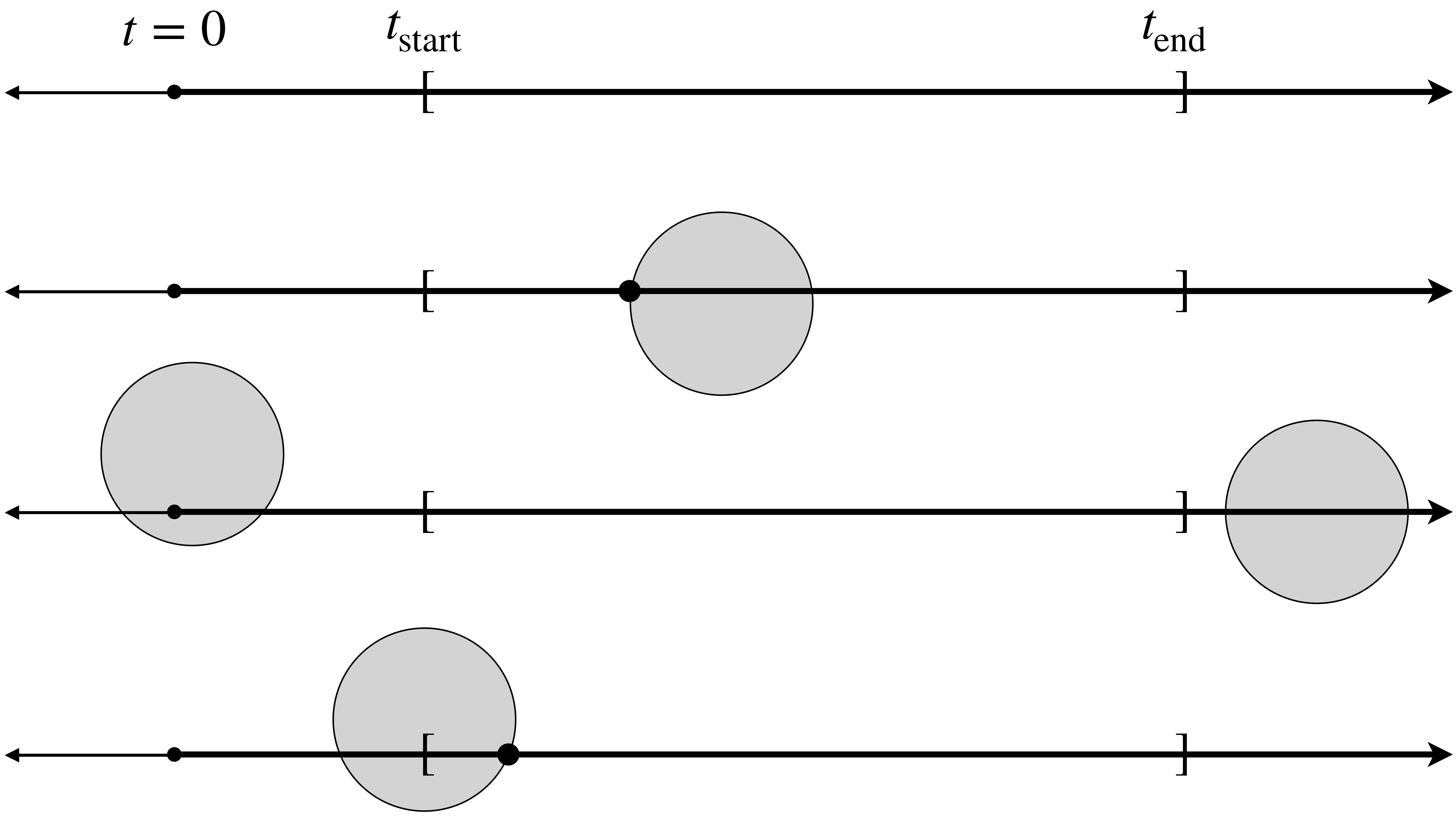
```
class Surface {
  boolean intersect(Hit hit, Ray r)
   ...
}
```

is there any intersection?

information about first intersection

```
class Hit {
   float t
   Surface surface
   Vector position
   Vector normal
   ...
}
```

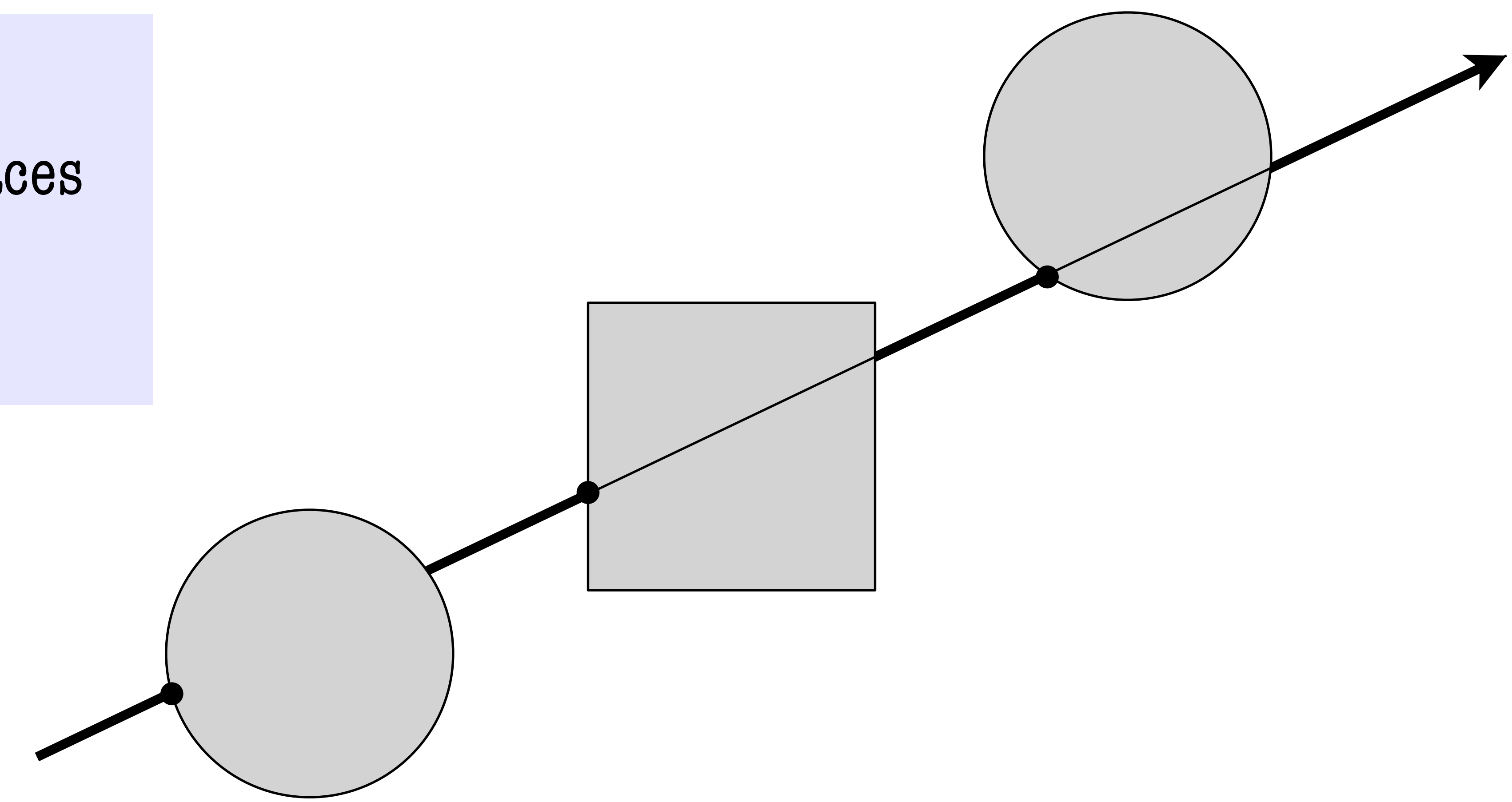
Intersection with ray segments



Scenes

- **Scenes contain many objects**
- **Need to find the first intersection along the ray**
  - that is, the one with the smallest positive t value in [start, end]

```
class Scene {
    List<Surface> surfaces
    ...
}
```



Intersection against many shapes

- **Input is a ray and a collection of surfaces**
- **Simple linear time algorithm:**

```
function intersect(ray, surfaceList)
    for surface in surfaceList
        intersect ray against surface
        if hit, reduce ray.end to t of hit
    return surface corresponding to minimum-t intersection
```

- **This is fine for small scenes, too slow for large ones**
  – real applications use sublinear methods (acceleration structures) which we will see later