Ray Tracing

CS 4620 Lecture 5

Announcements

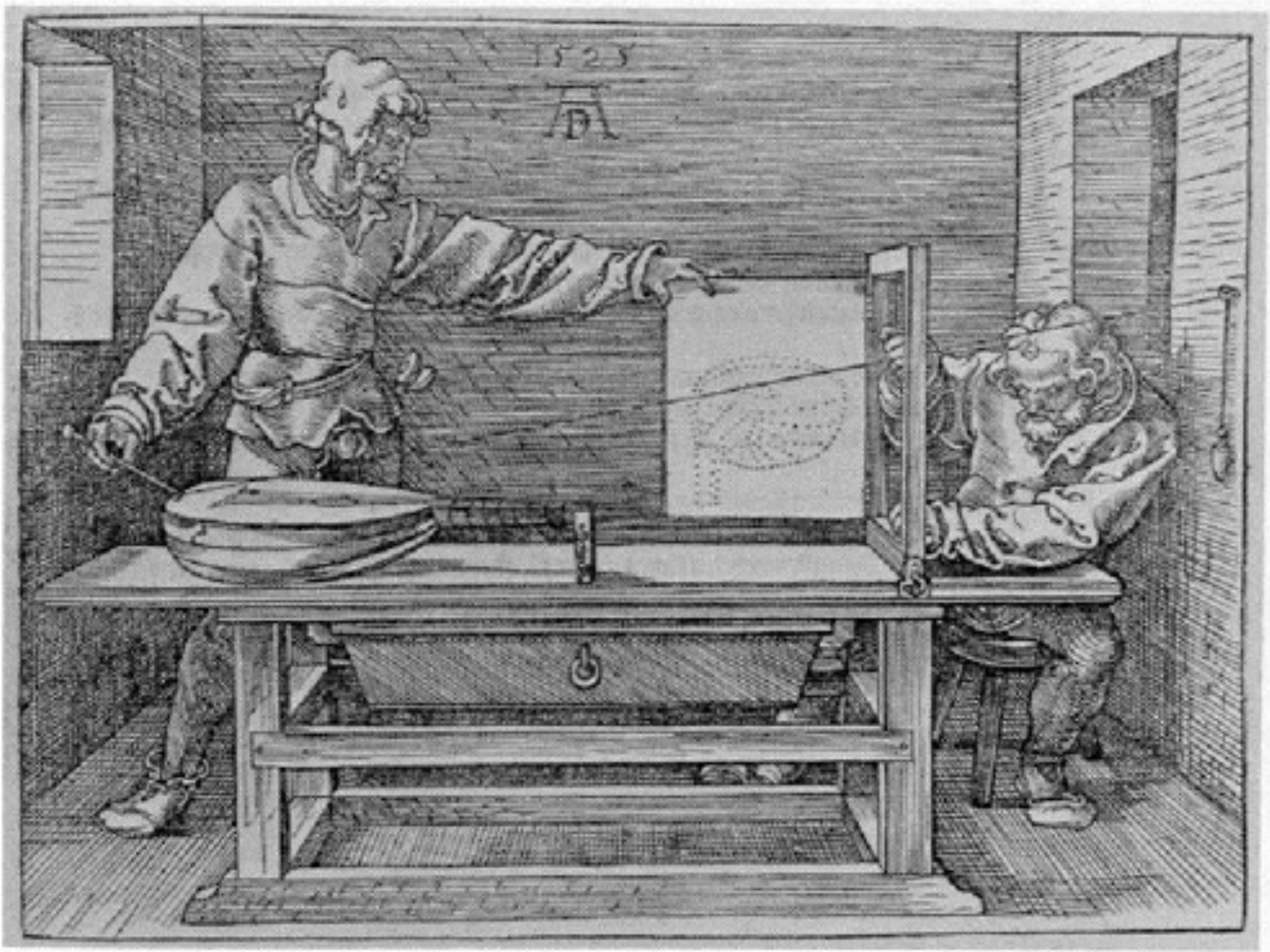
- Hope you had a good break!
- A1 due Thursday
- Will post updated office hours in a calendar to make sure we are all synced up

What is graphics?

- Scenes
  - Triangles
  - Materials
  - Lights

- Images
  - Pixels

Plane projection in drawing



[Carlbom & Paciorek 78, Albrecht Durer]

Two approaches to rendering

- These projection ideas describe the relationship between the world and the image
- But how do we use them to compute an image?

```
for each object in the scene {
    for each pixel in the image {
        if (object affects pixel) {
            do something
        }
    }
}
```

object order
or
rasterization

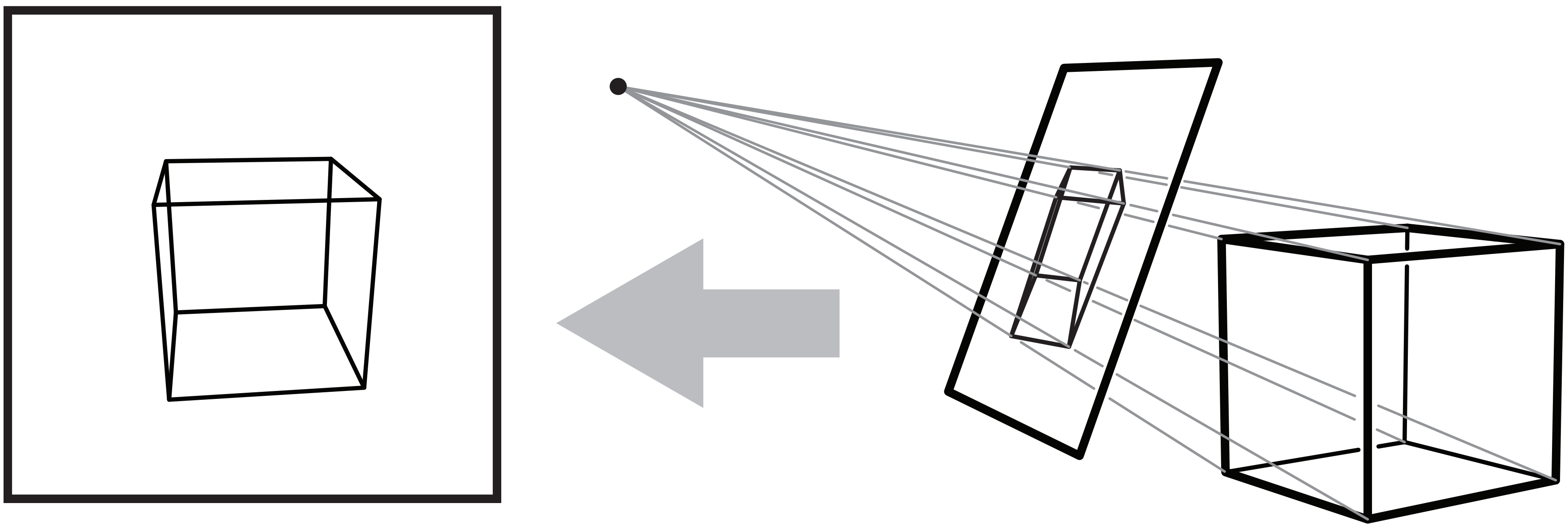
```
for each pixel in the image {
    for each object in the scene {
        if (object affects pixel) {
            do something
        }
    }
}
```

We will do this first

image order
or
ray tracing

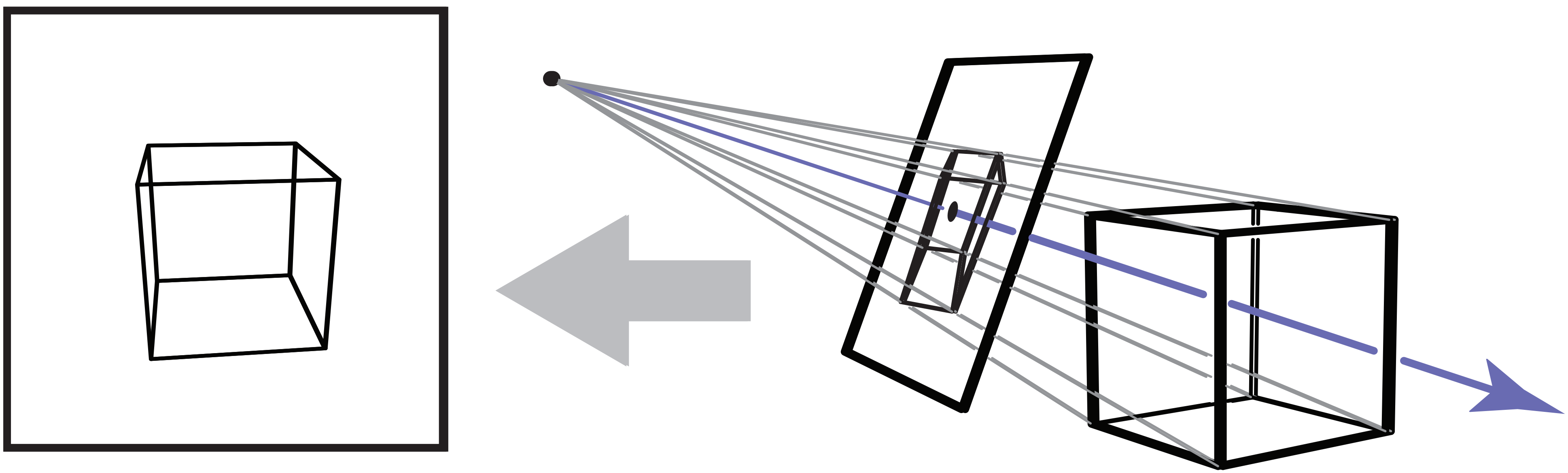
Object Order

- To render an image of a 3D scene, we *project* it onto a plane
- Most common projection type is *perspective projection*

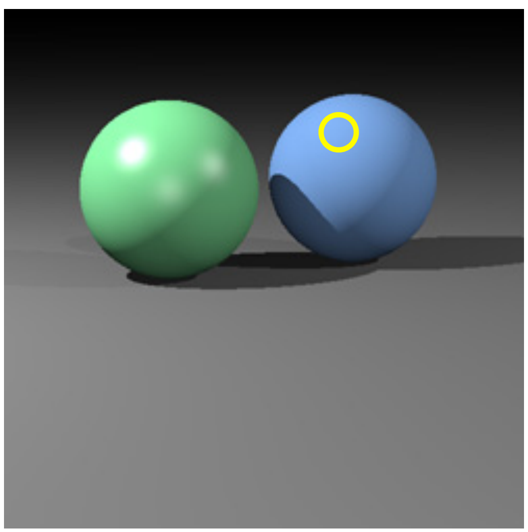


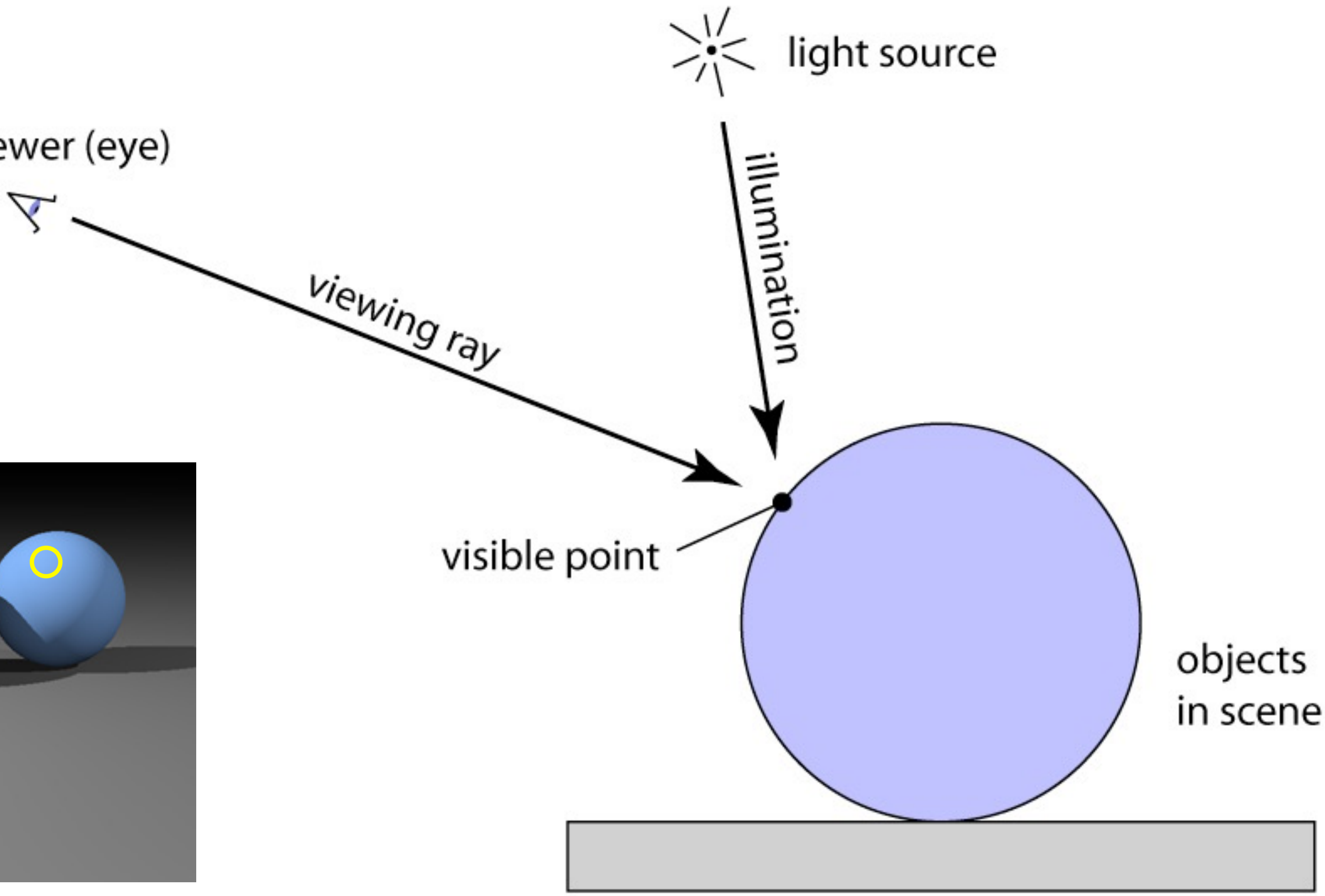
Ray tracing idea

- Start with a pixel—what belongs at that pixel?
- Set of points that project to a point in the image: a **ray**

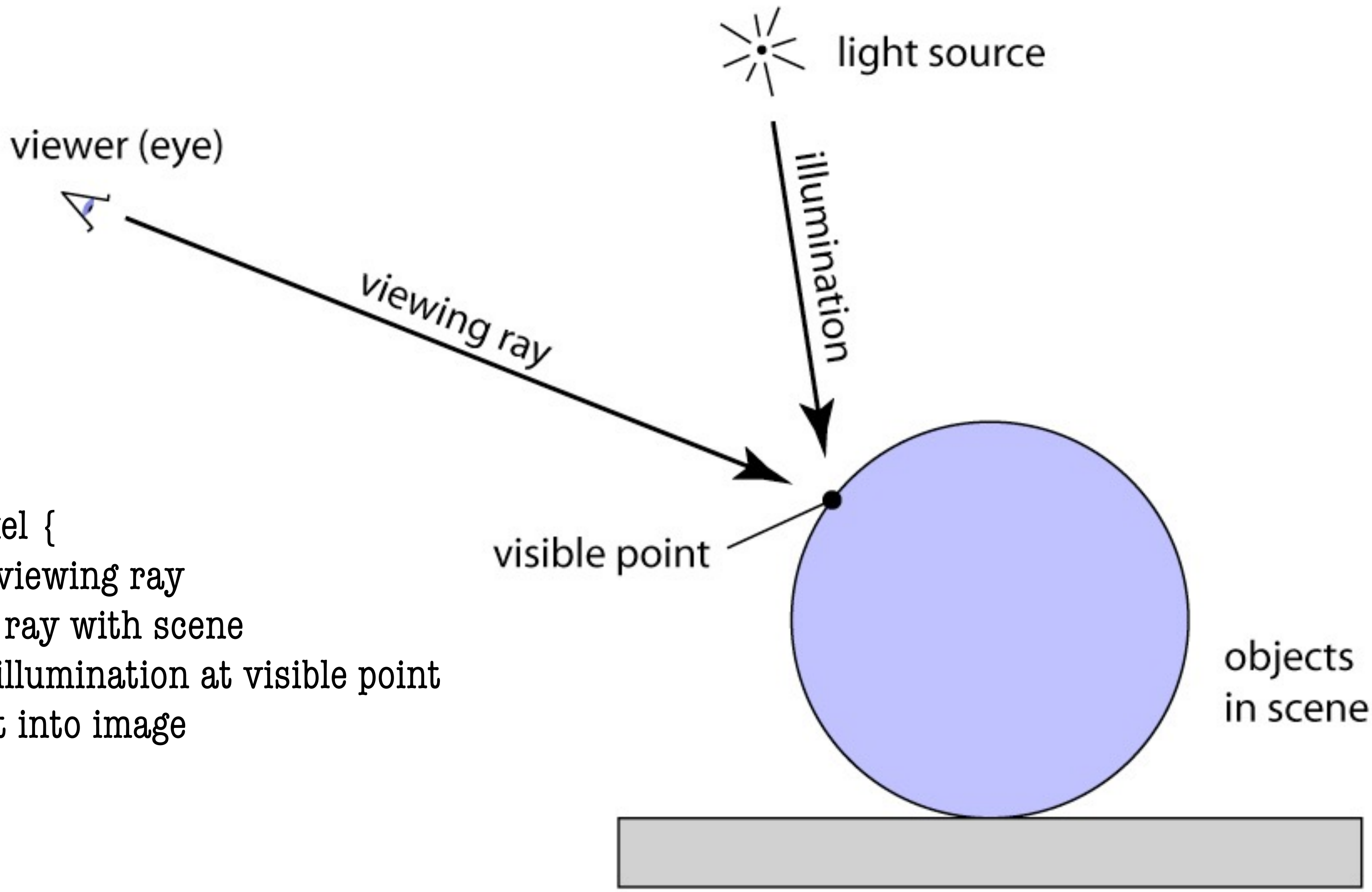


Ray tracing idea





Ray tracing algorithm



for each pixel {
  compute viewing ray
  intersect ray with scene
  compute illumination at visible point
  put result into image
}

Math review

- **Read:**
  - **Tiger, Chapter 2, 5: Misc Math, Linear Algebra**
  - **Gortler, Chapter 1, 2: Linear**
- Vectors and points
- Vector operations
  - addition
  - scalar product
- More products
  - dot product
  - cross product

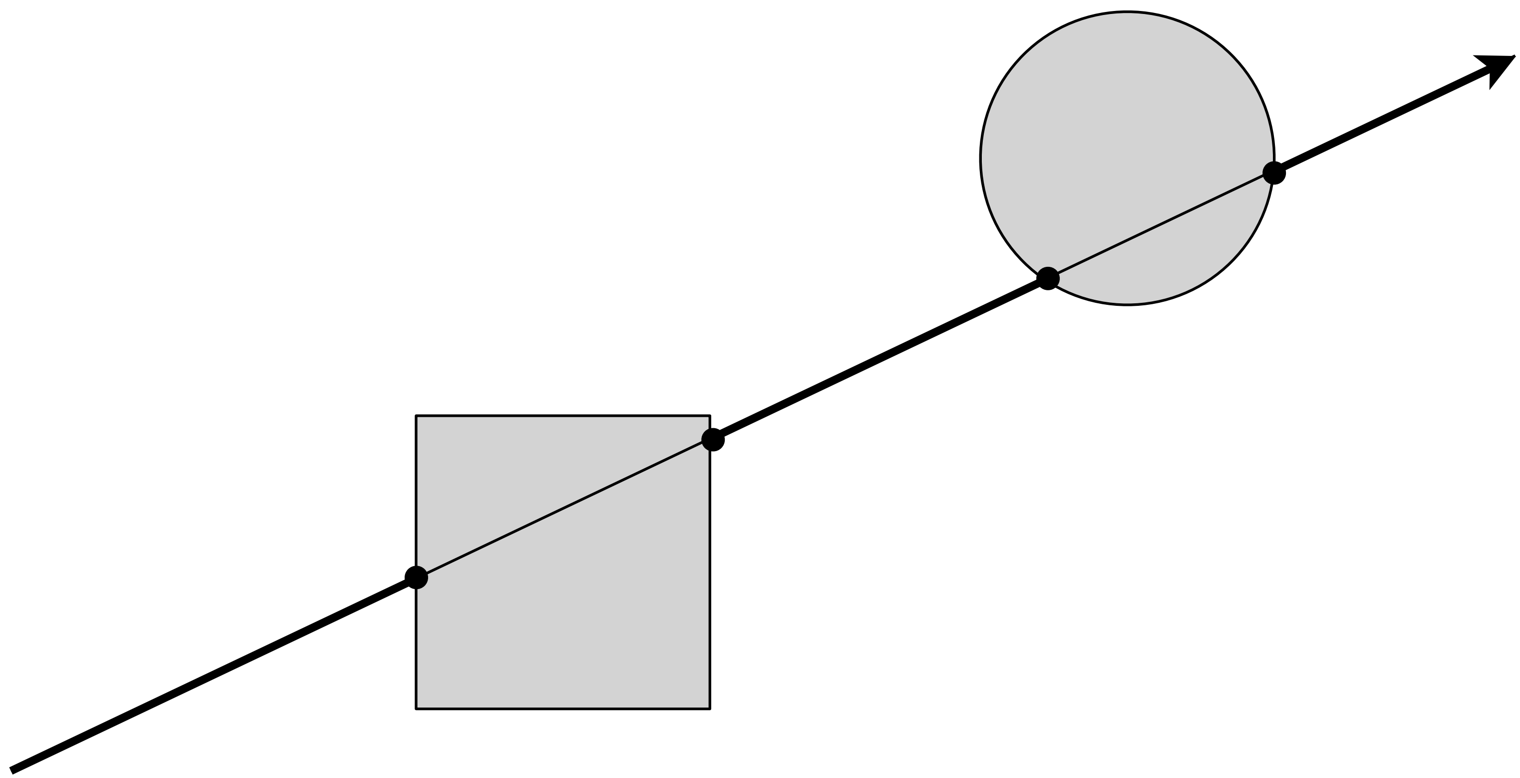
Math review

- Vectors and points
  - P = (x, y, z)
  - V = (a, b, c)
- Vector operations
  - addition
  - scalar product
- Point operations
  - subtraction

Math review

- Vectors and points
  - $P = (x, y, z)$
  - $V = (a, b, c)$
- More products
  - dot product
    - geometric interpretation
  - cross product
    - geometric interpretation

Ray intersection

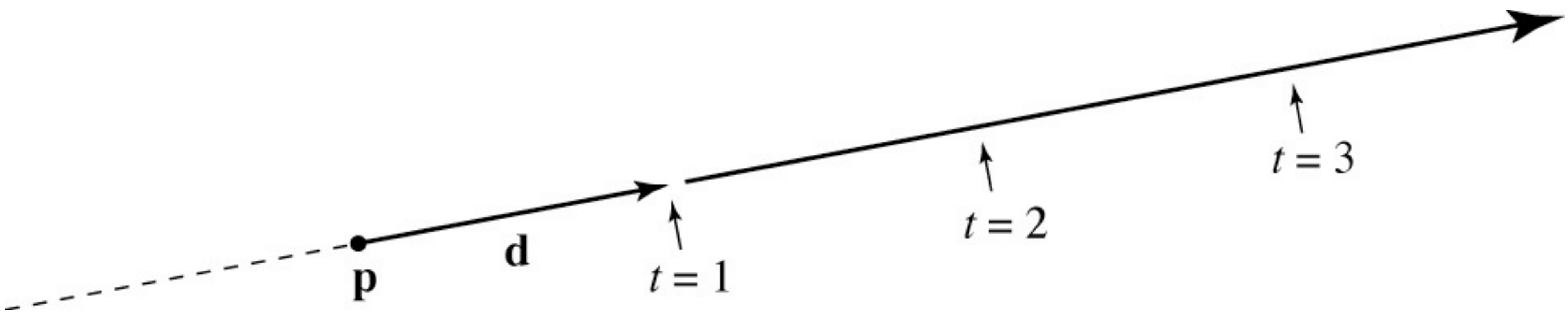


Ray: a half line

- Standard representation: point **p** and direction **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 **d** with α**d** doesn't change ray (α > 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 Shirley 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

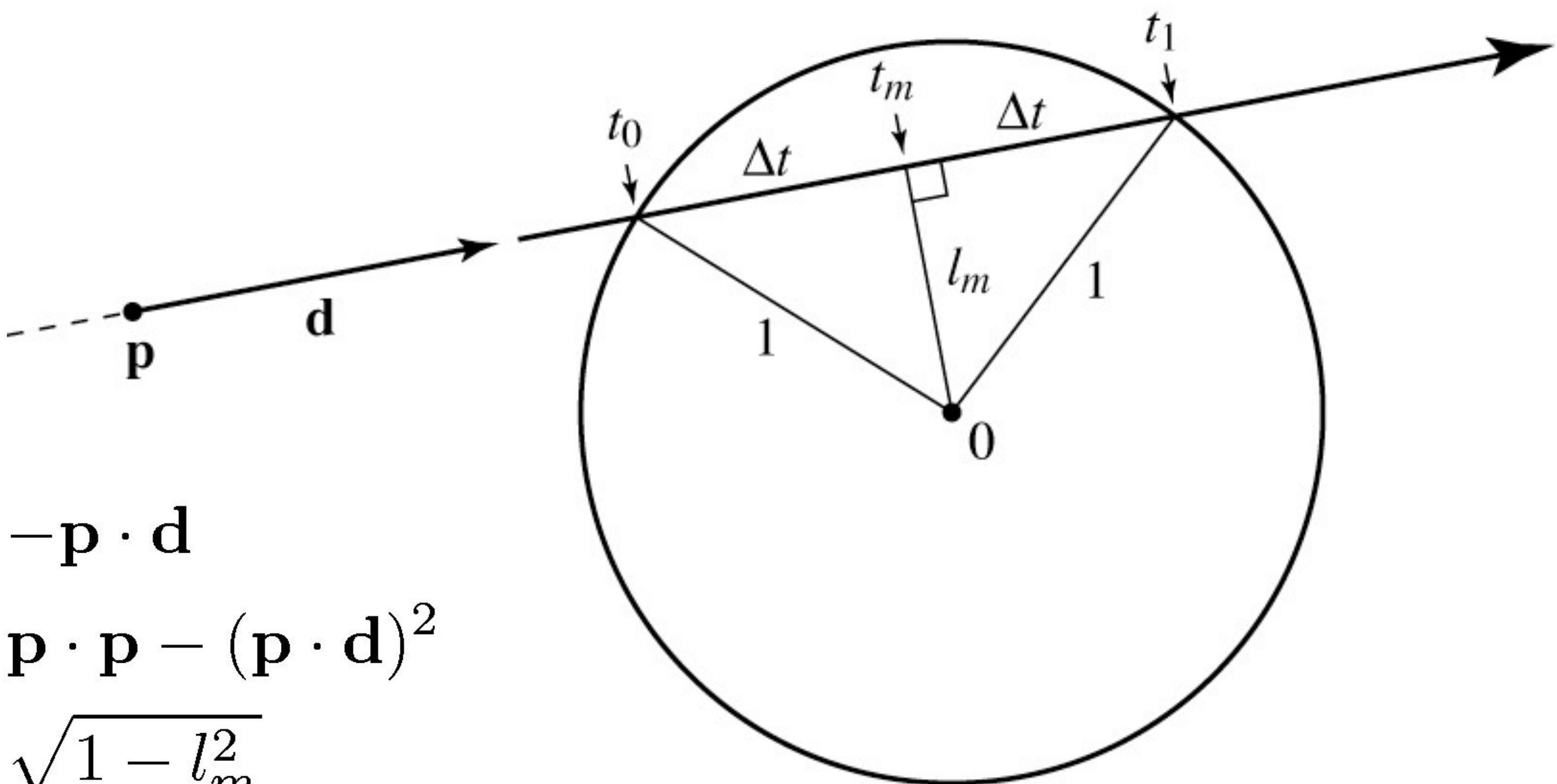
- Solution for *t* by quadratic formula:

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

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

  - 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 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);
        bool didhit = s.intersect(ray, 0, +inf)
        if didhit
            image.set(ix, iy, white);
    }
```