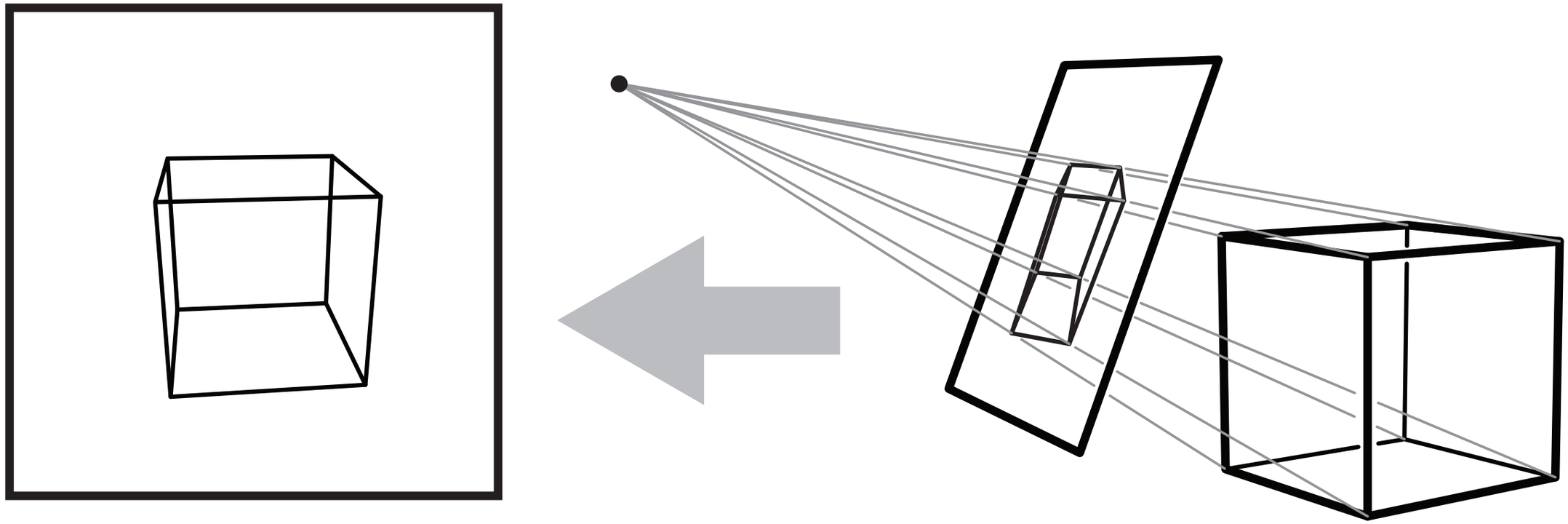
# Ray Tracing Intro

**Steve Marschner**
**CS 4620**
**Cornell University**

# Projection

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



# Two approaches to rendering

# Two approaches to rendering

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

**object order**
or
**rasterization**

# Two approaches to rendering

```
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
      }
   }
}
```

**image order**
or
**ray tracing**

# Two approaches to rendering

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

**object order**
or
**rasterization**

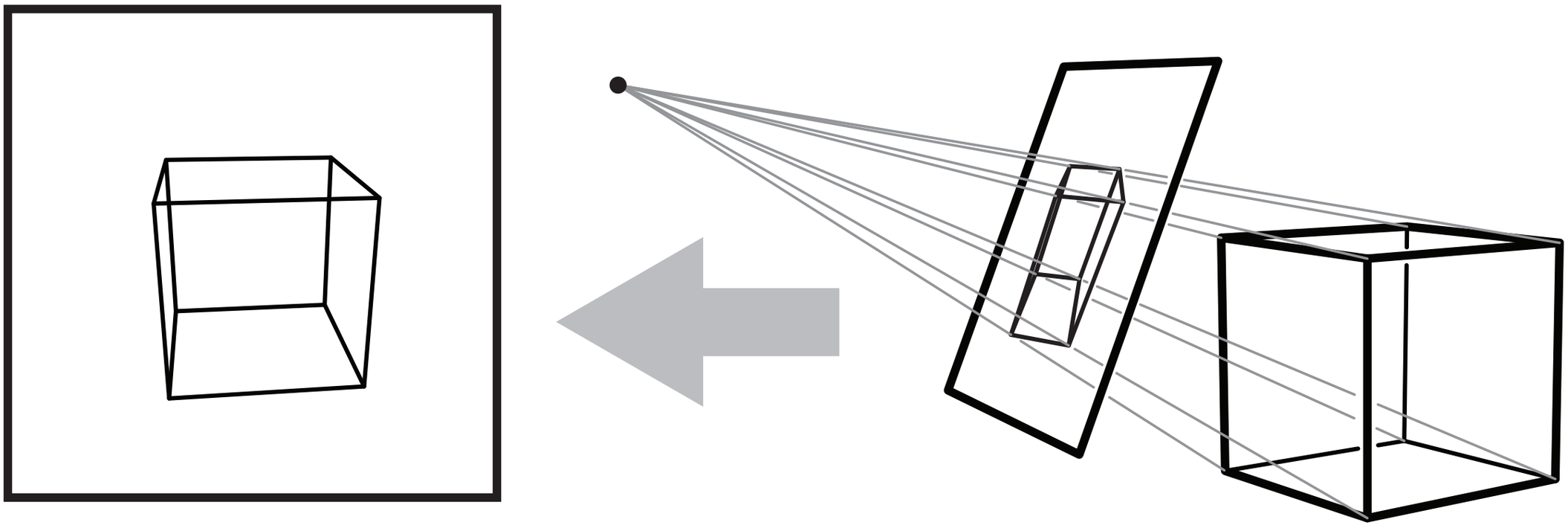
```
for each pixel in the image {
   for each object in the scene {
      if (object affects pixel) {
         do something
      }
   }
}
```

**We will do this first**

**image order**
or
**ray tracing**

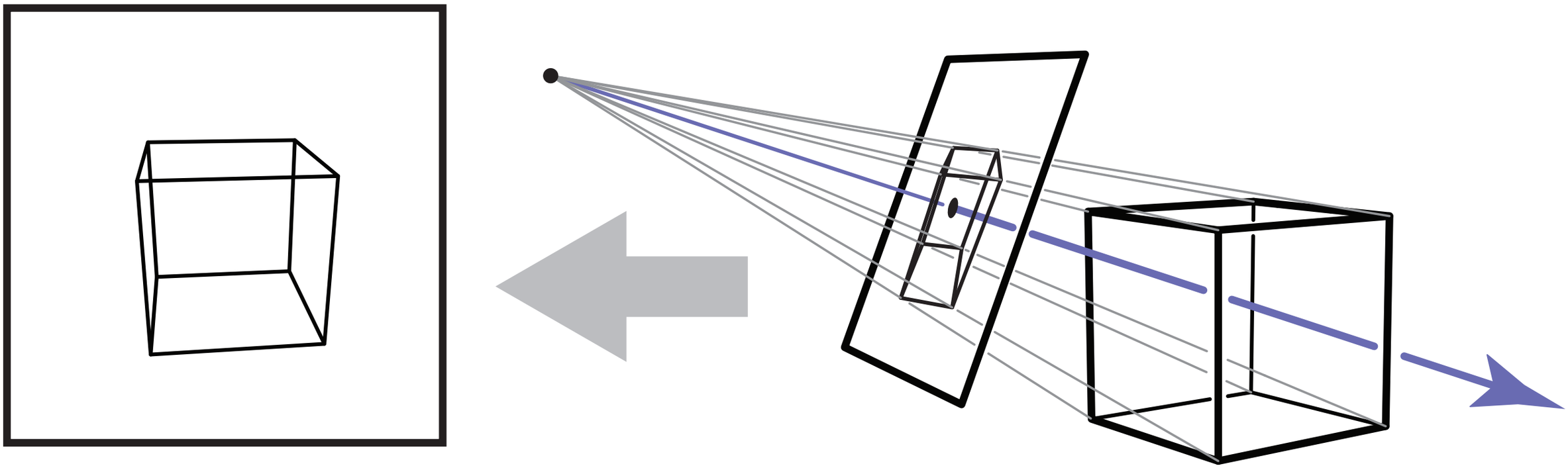
# Ray tracing idea

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

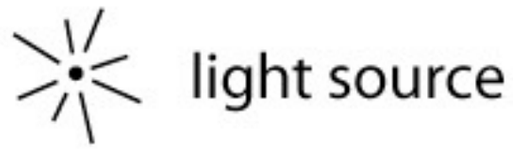


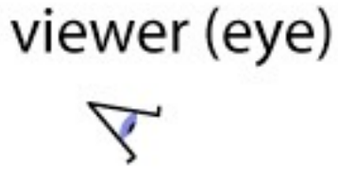
# Ray tracing idea

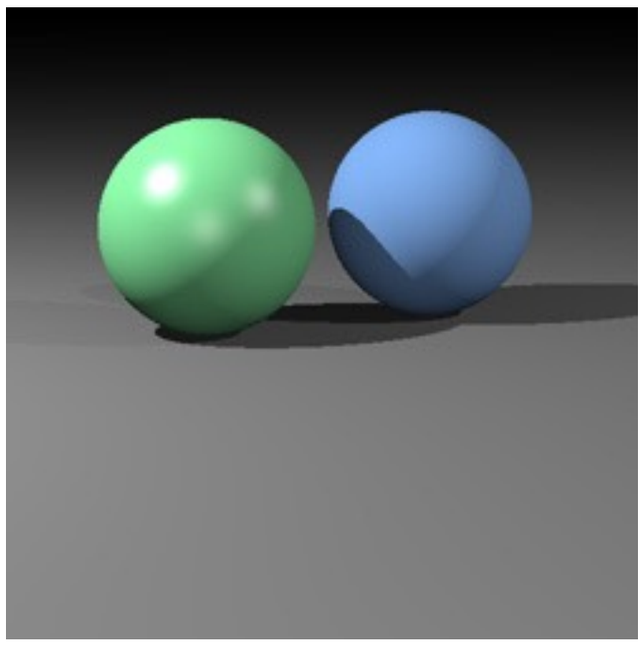
- **Start with a pixel—what belongs at that pixel?**
- **Set of points that project to a point in the image: a ray**

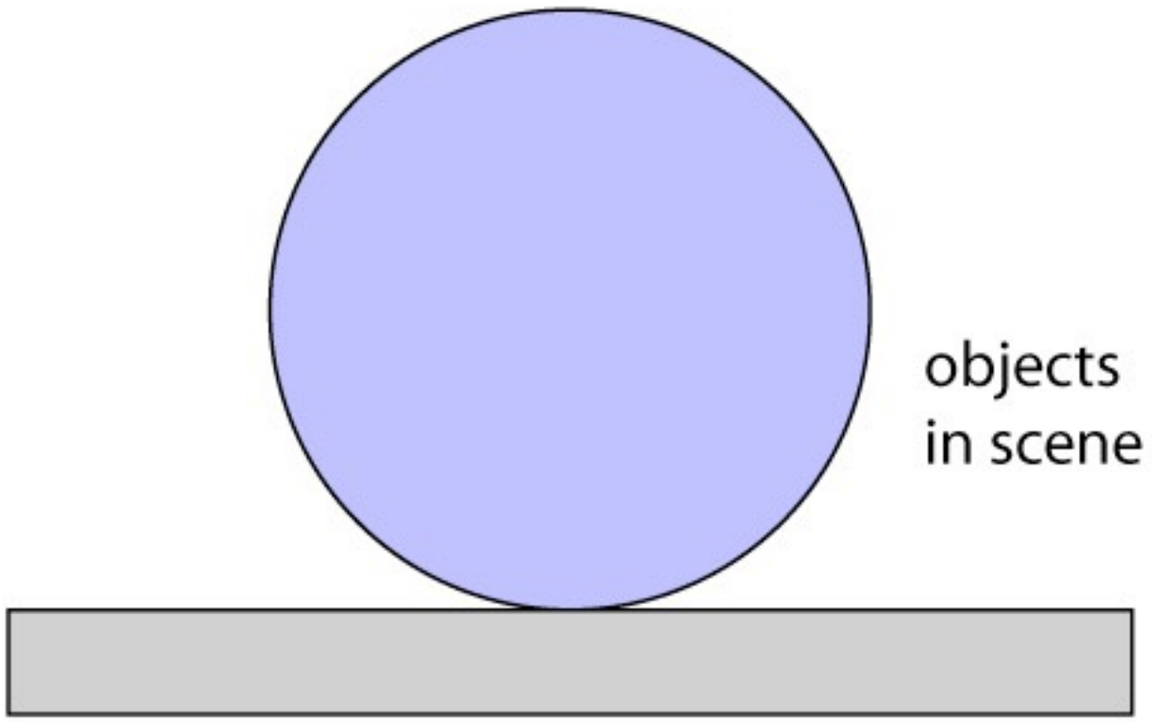


# Ray tracing idea



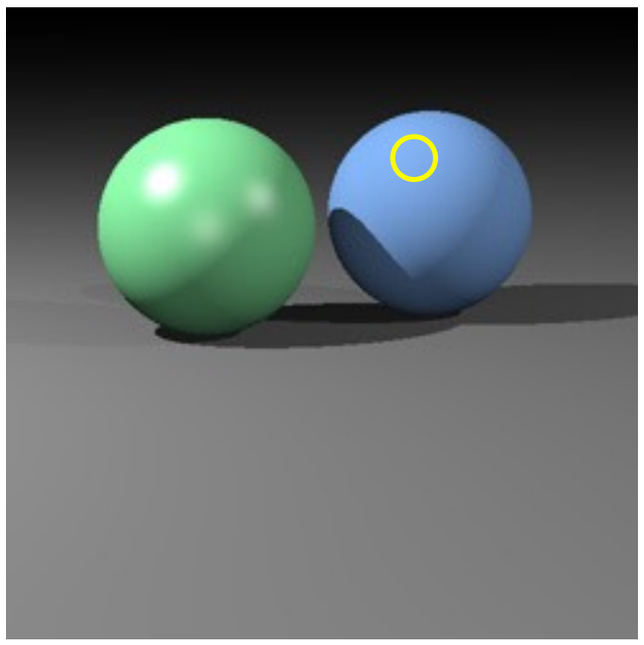



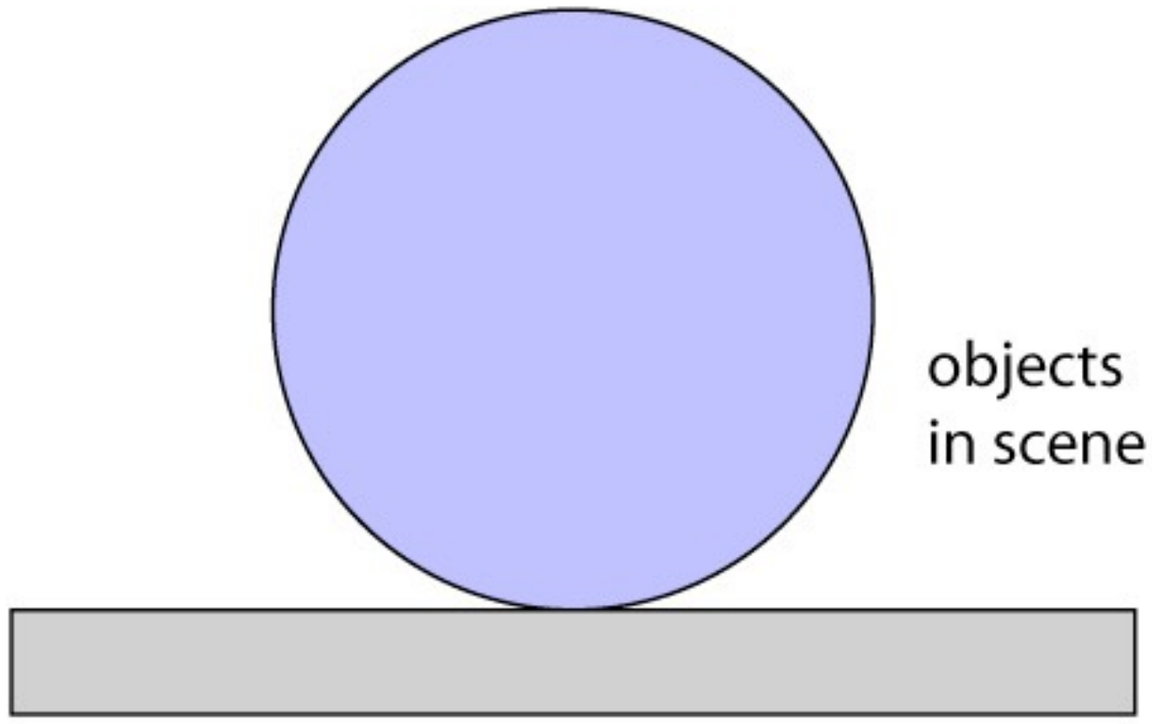


# Ray tracing idea

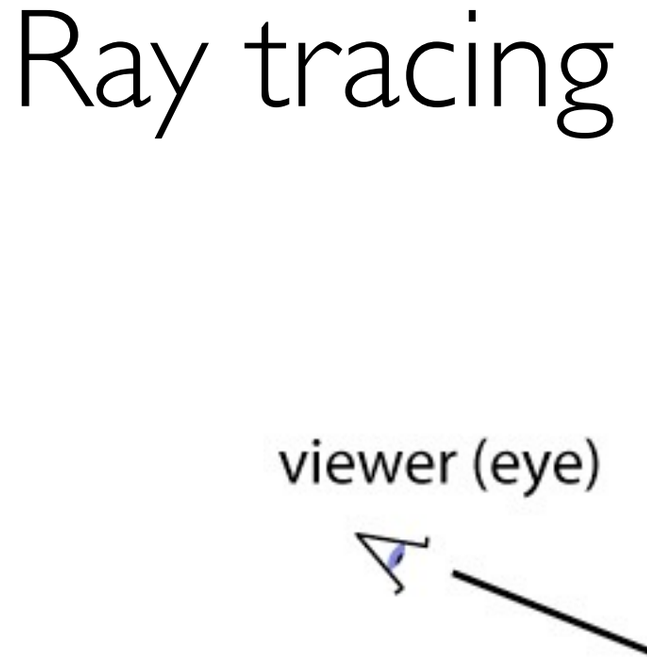
light source

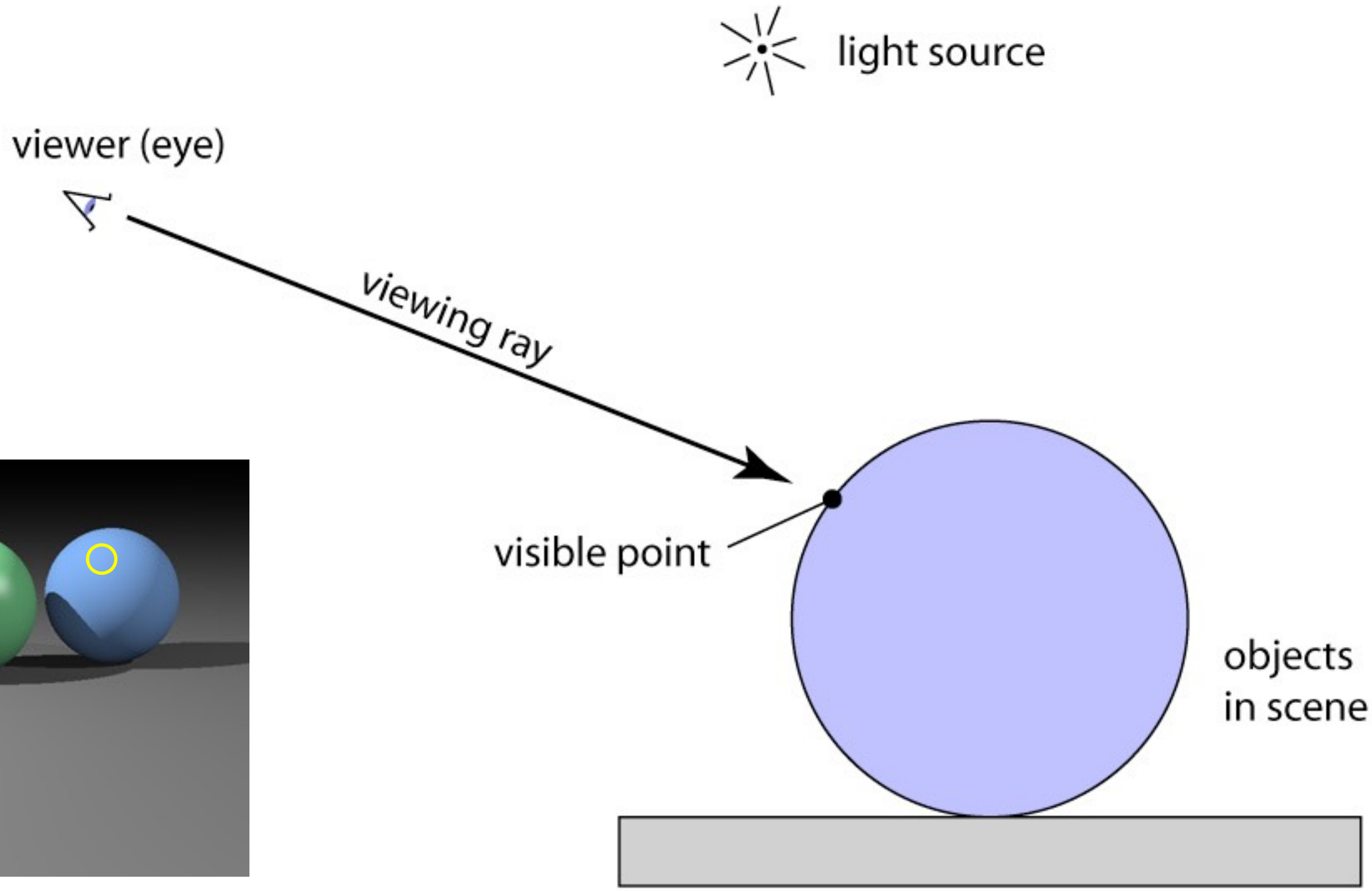
viewer (eye)



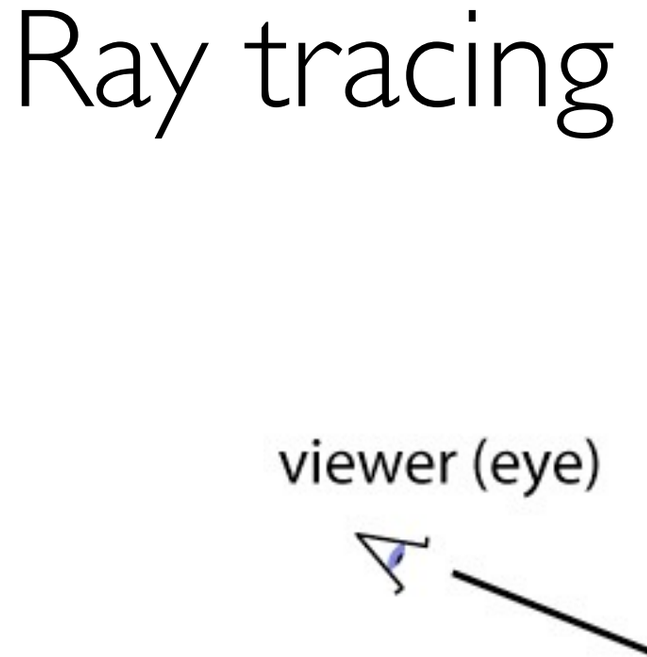


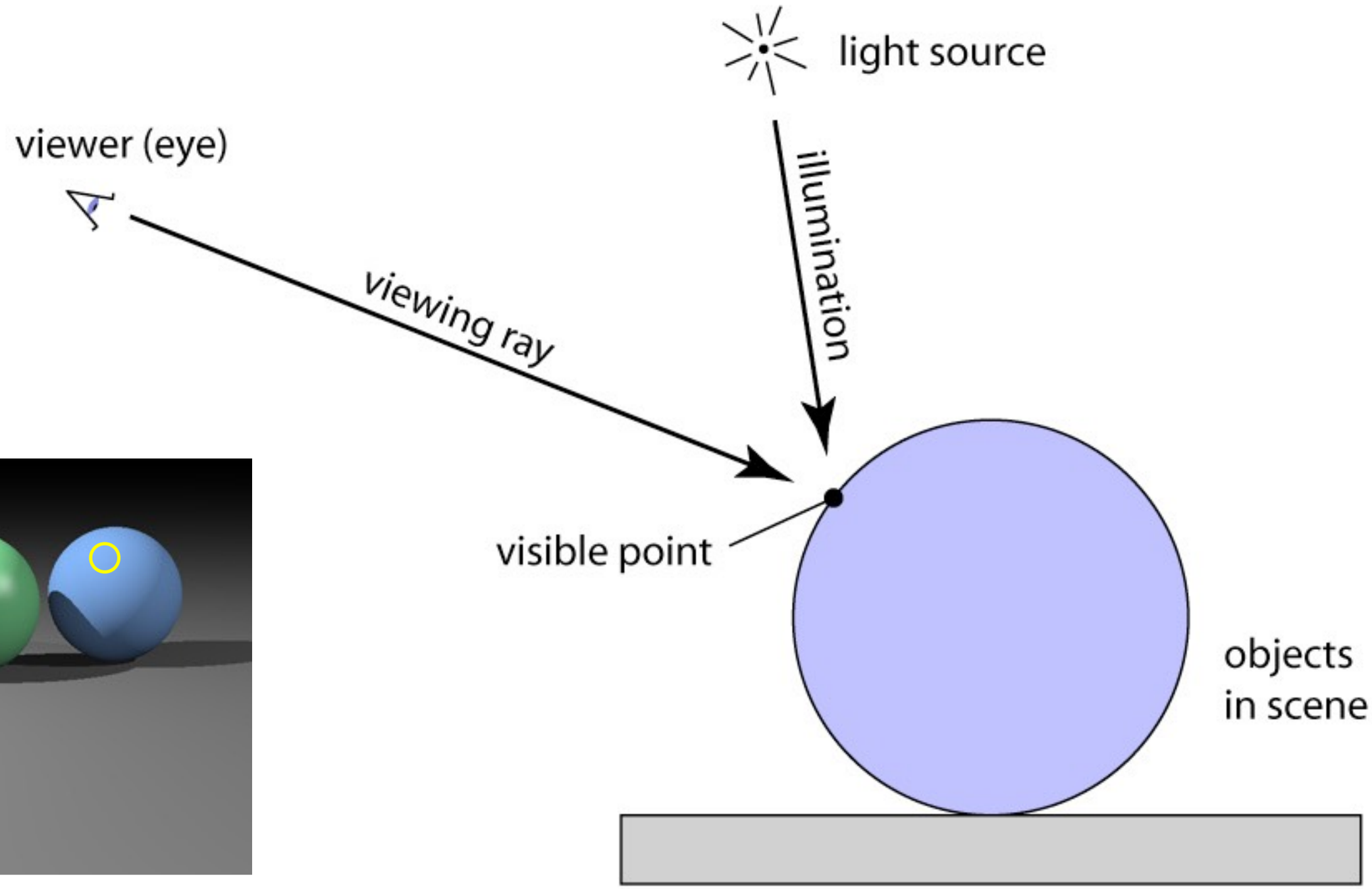
# Ray tracing idea



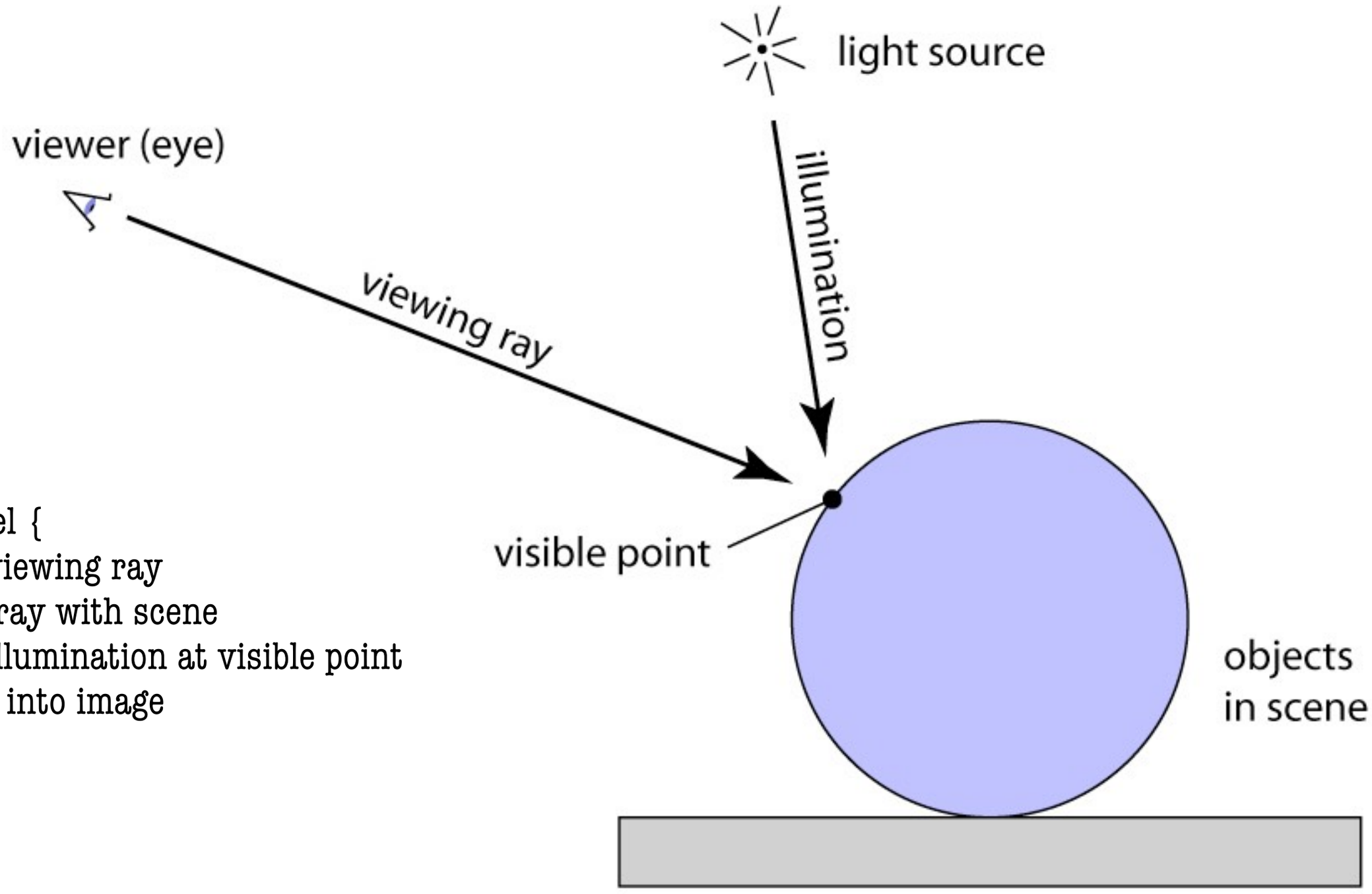


# Ray tracing idea





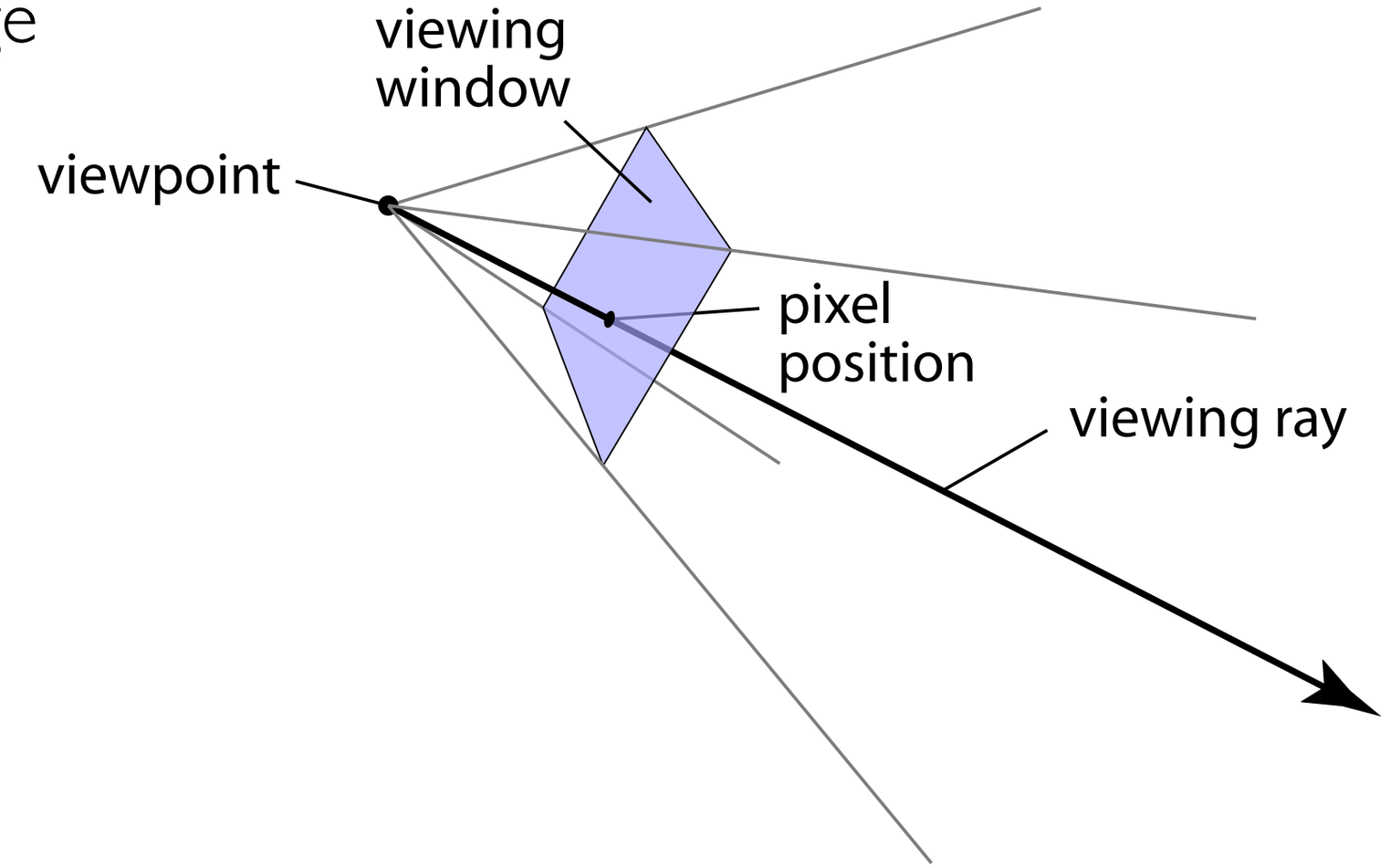
# Ray tracing algorithm




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

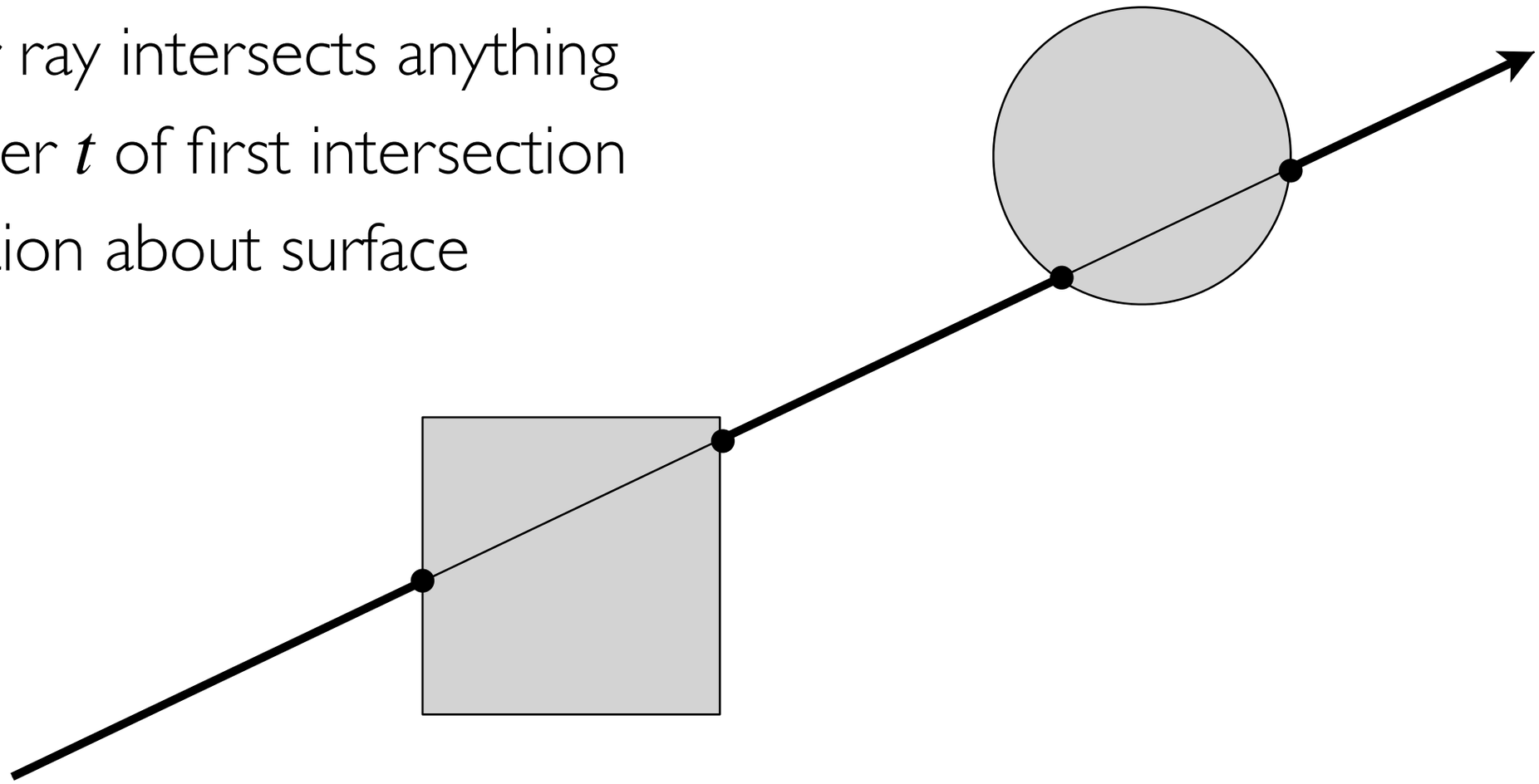
# Eye ray generation

- **inputs:**
  - camera parameters
  - a point in the image
- **outputs:**
  - a viewing ray



# Ray intersection

- **inputs:**
  - list of objects in the scene
  - a ray
- **outputs:**
  - whether ray intersects anything
  - parameter $t$ of first intersection
  - information about surface



# Shading

- **inputs:**
  - eye direction
  - light direction
    (for each of many lights)
  - surface normal
  - surface parameters
    (color, roughness, …)
- **outputs:**
  - light reflected towards eye