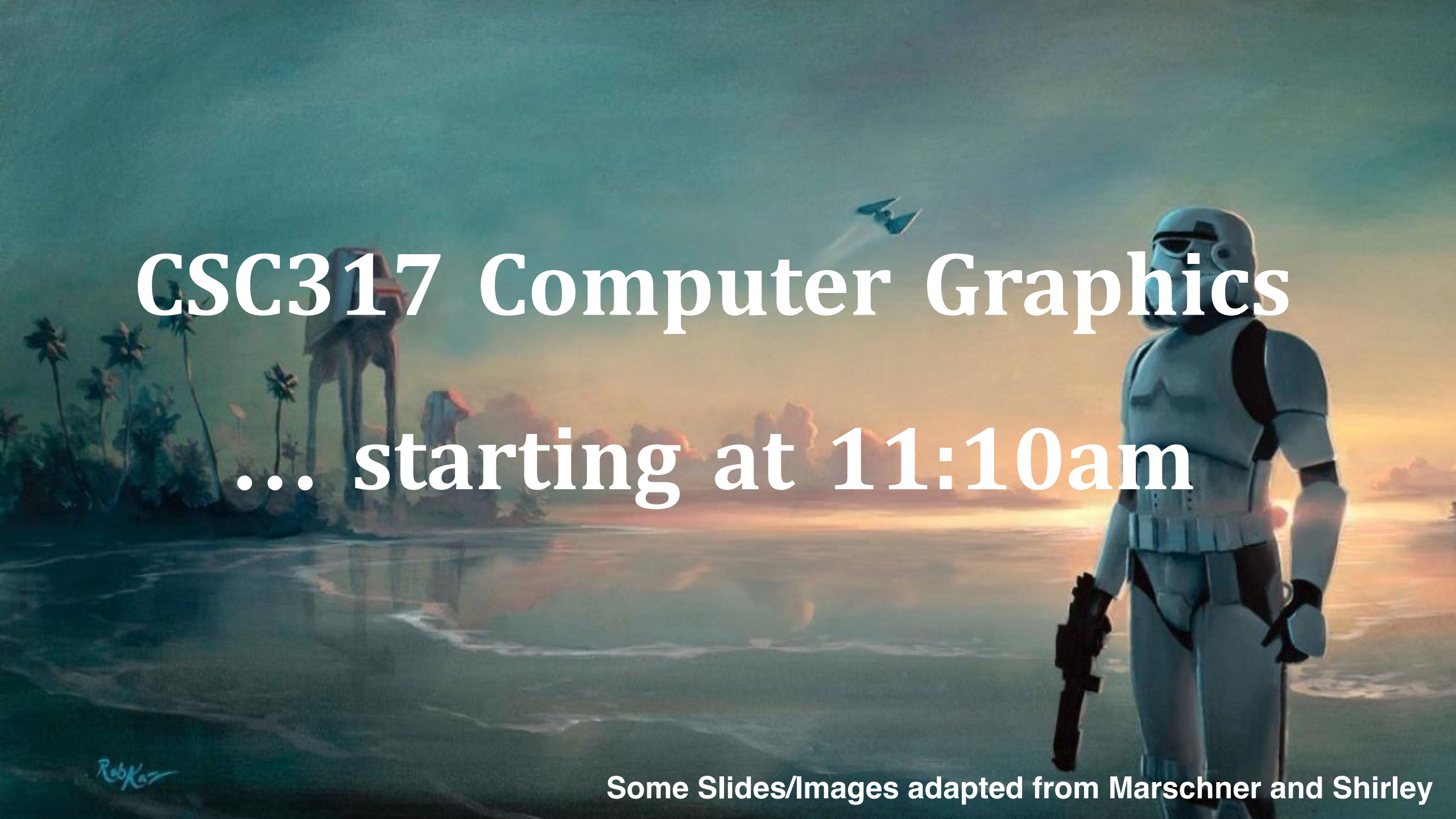

CSC317 Computer Graphics
… starting at 11:10am
Some Slides/Images adapted from Marschner and Shirley

Today: Ray Tracing

Announcements

Assignment 2 is due Wednesday

Assignment 3 available today, (due October 4th)

Marking Scheme Updated

Things you should do before asking for help

1. Google any errors. 99% of the time the solution to your problem is on the first page of google hits.
2. Look at the file output. Image file doesn't open ? Write a small example in ASCII format to disk and open it with a text editor.
3. Build simple test cases and isolate the function or line of code that is not working
4. Be aware that we cannot tell you how to implement something. Only clarify explanations/theory behind the assignments.

Any Questions ?

Today: Ray Tracing

Review Ray Casting

Intersection Testing for Groups of Objects

Point and Directional Lights

Shadows

Lambertian Shading Model

Blinn-Phong Shading Model

Reflection

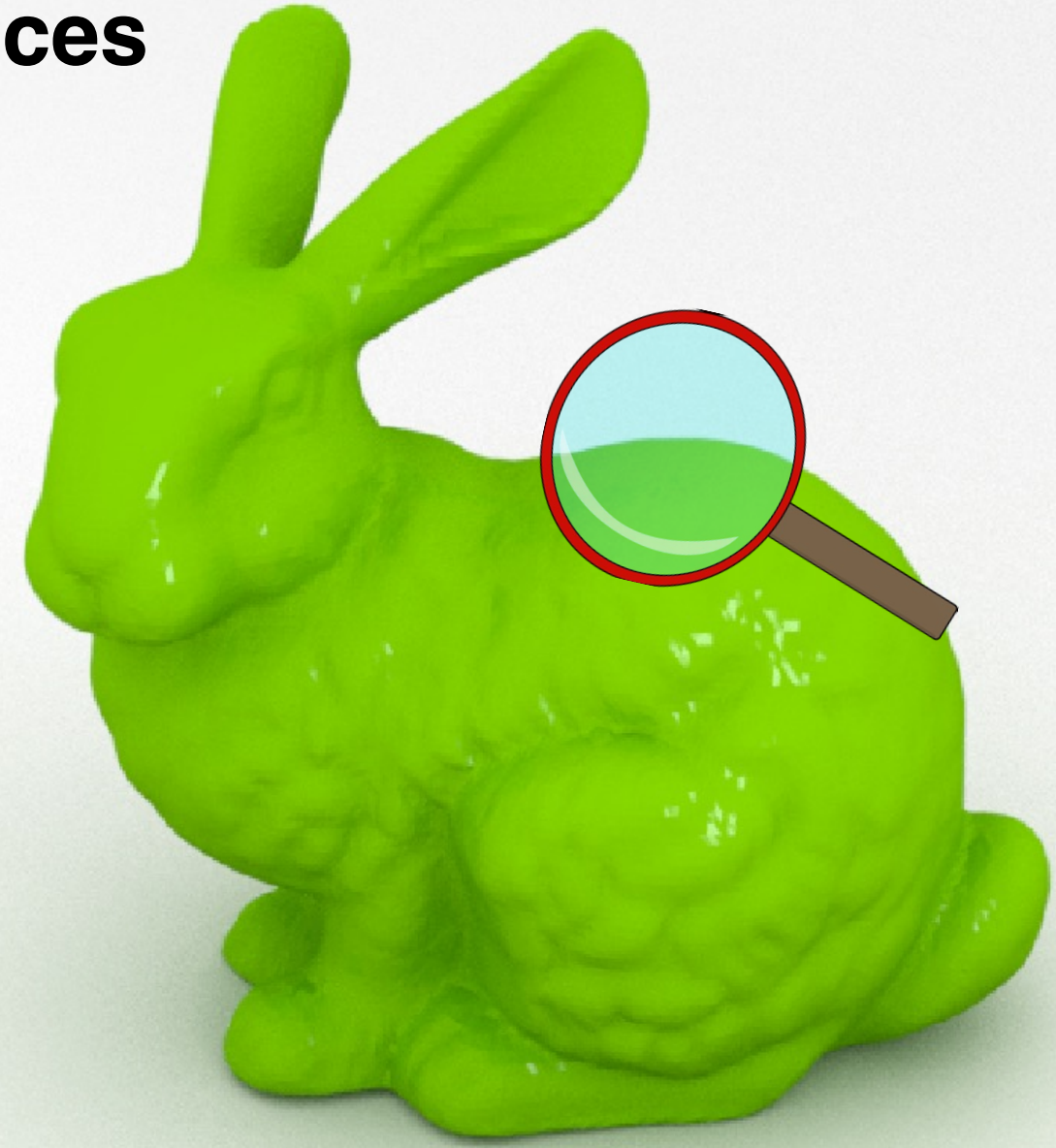
Transparency and Refraction

Ray Casting

```
for each pixel in the image {
    Generate a ray
    for each object in the scene {
        if (Intersect ray with object) {
            Set pixel colour
        }
    }
}
```

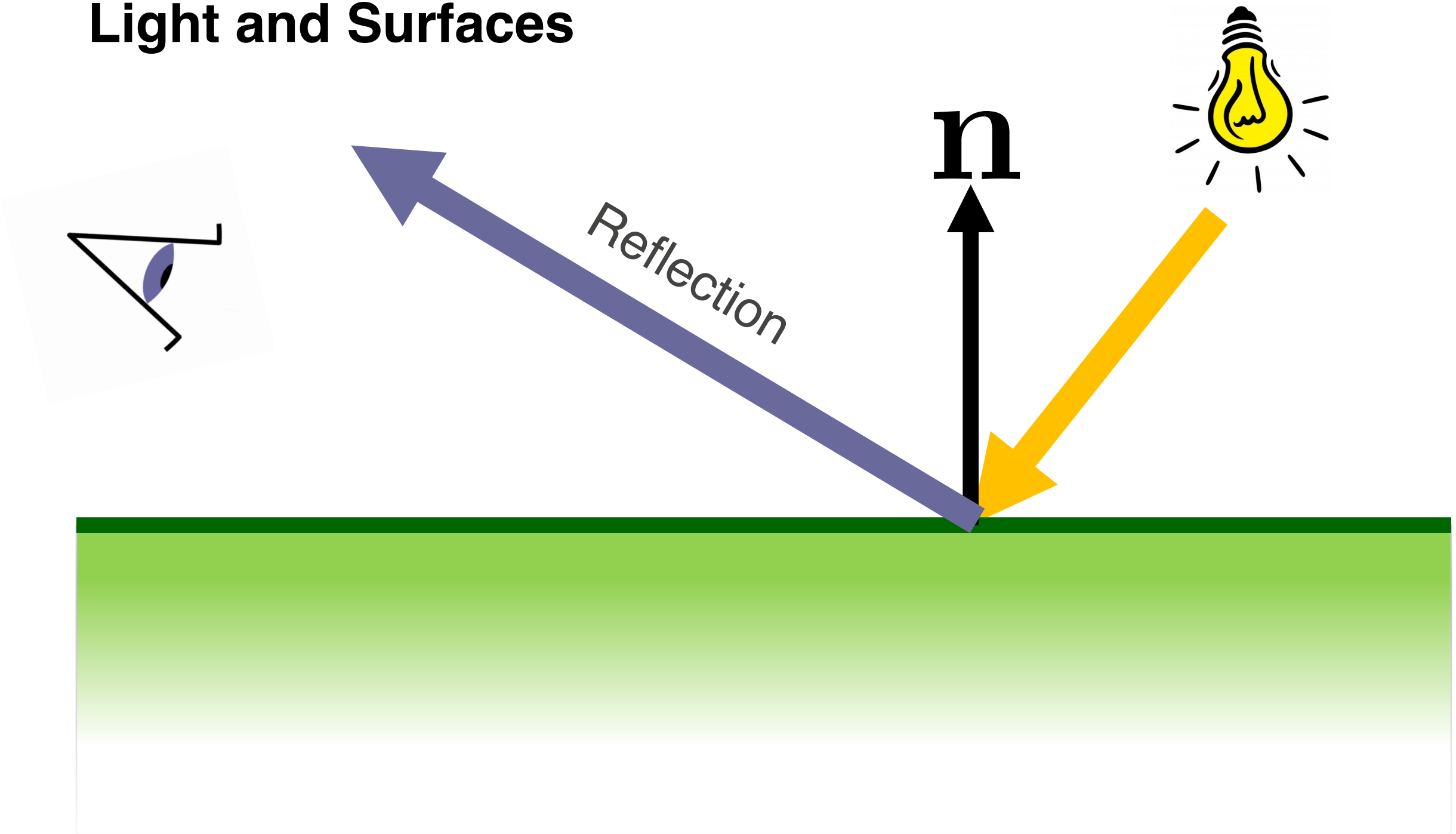
Light and Surfaces

Light and Surfaces

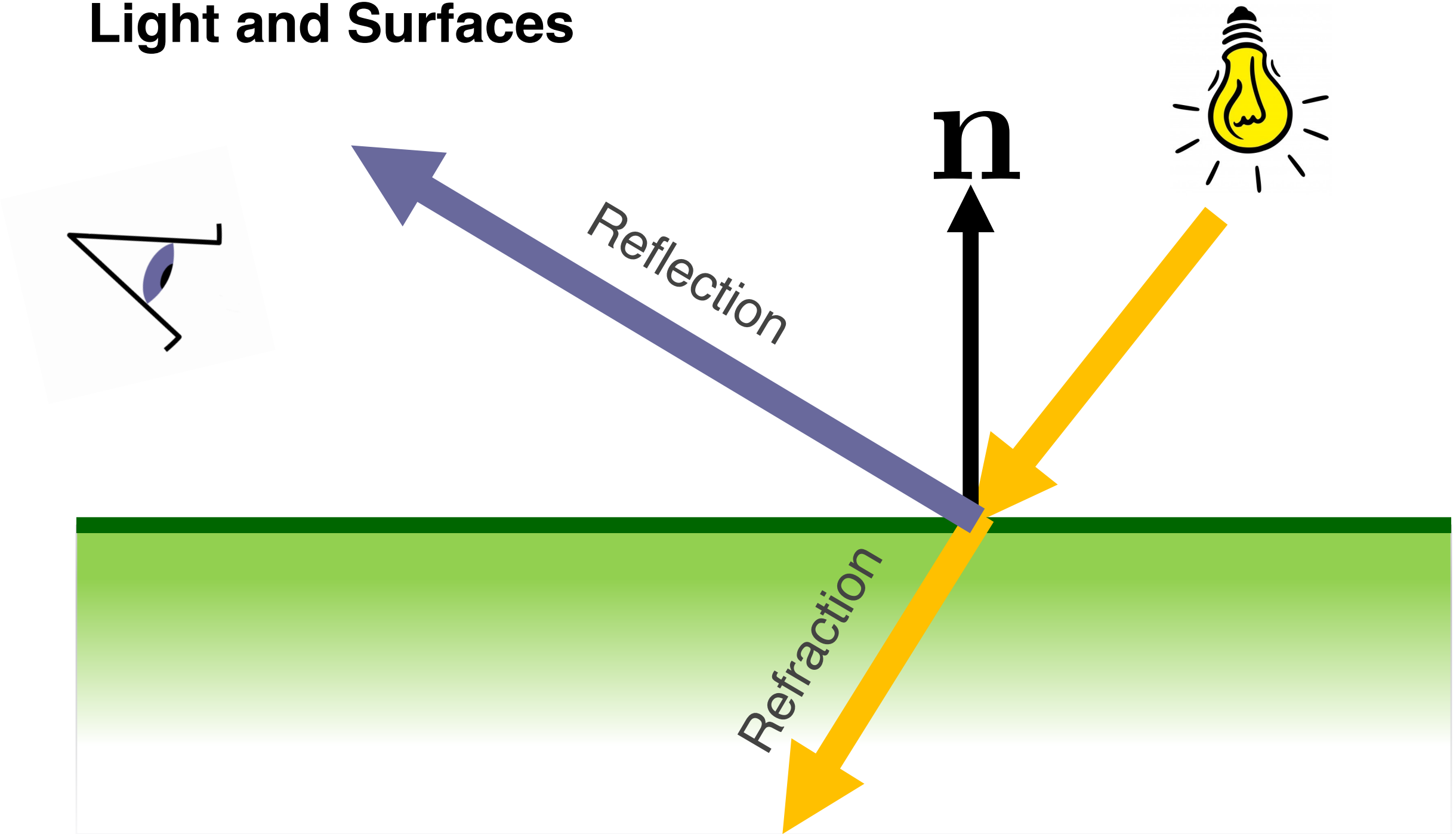


Light and Surfaces

Light and Surfaces



Light and Surfaces



Lights

Two types of lights:

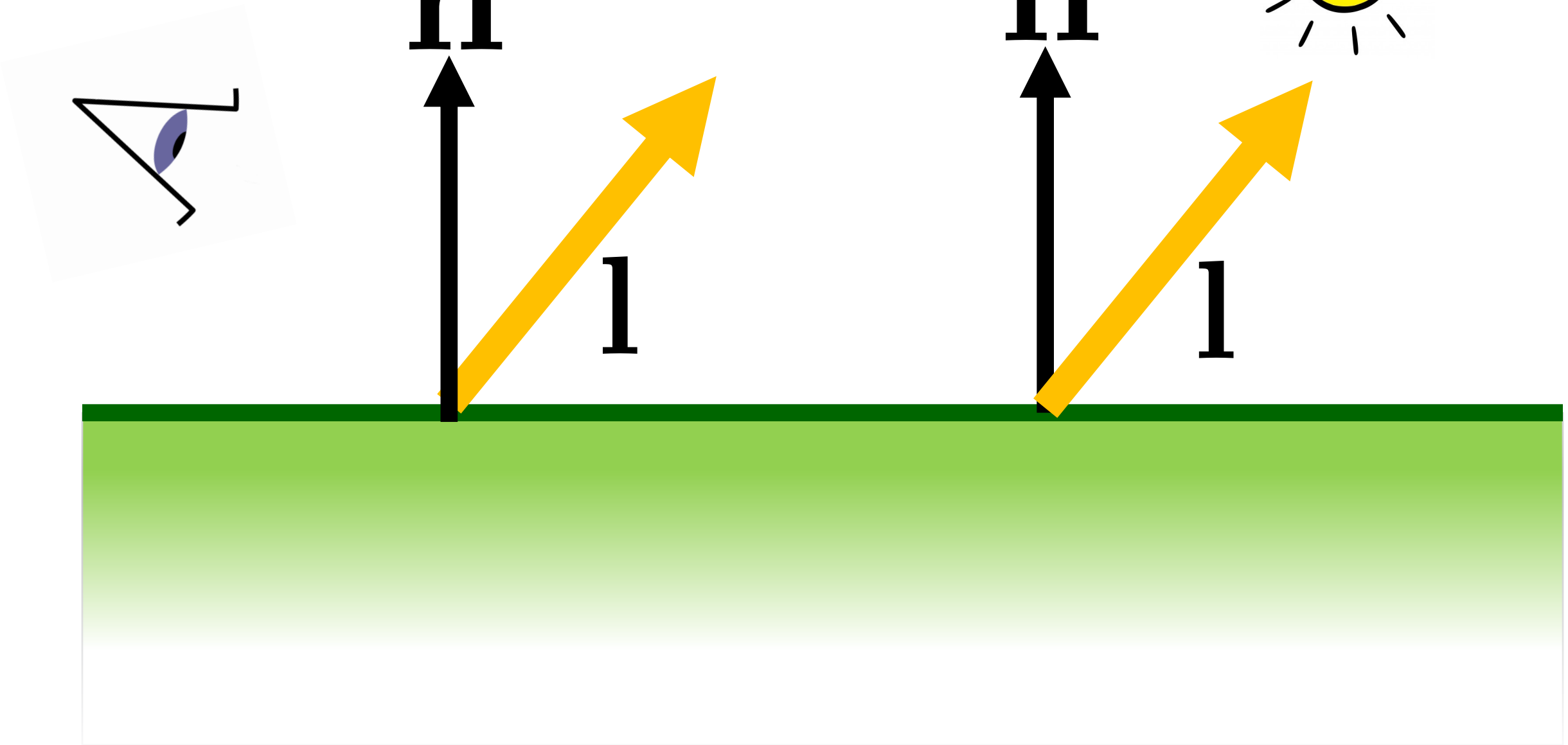
Directional Light:

Direction of light does not depend on the position of the object. **Light is very far away**

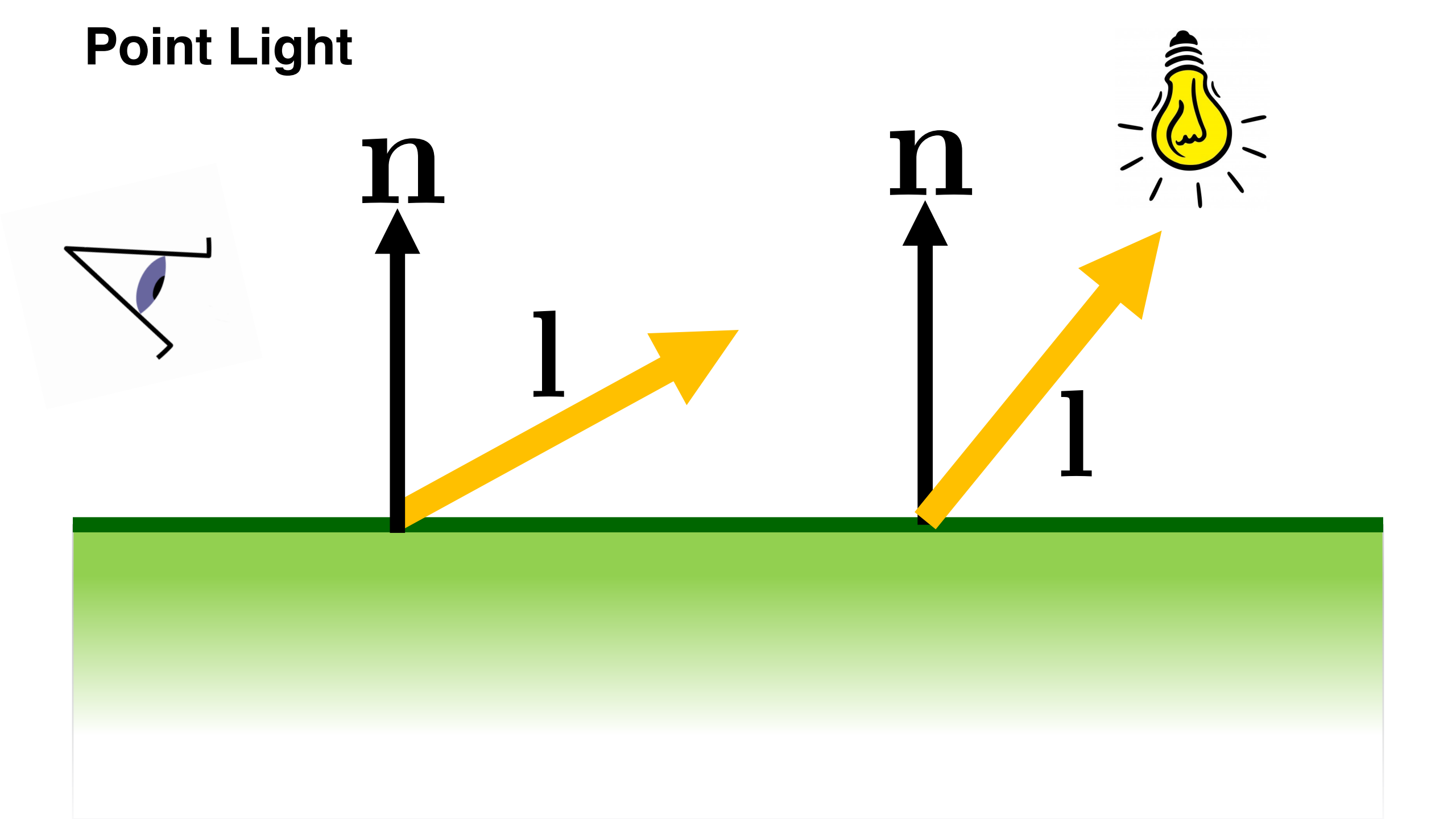
Point Light

Direction of light depends on position of object relative to light.

Directional Light



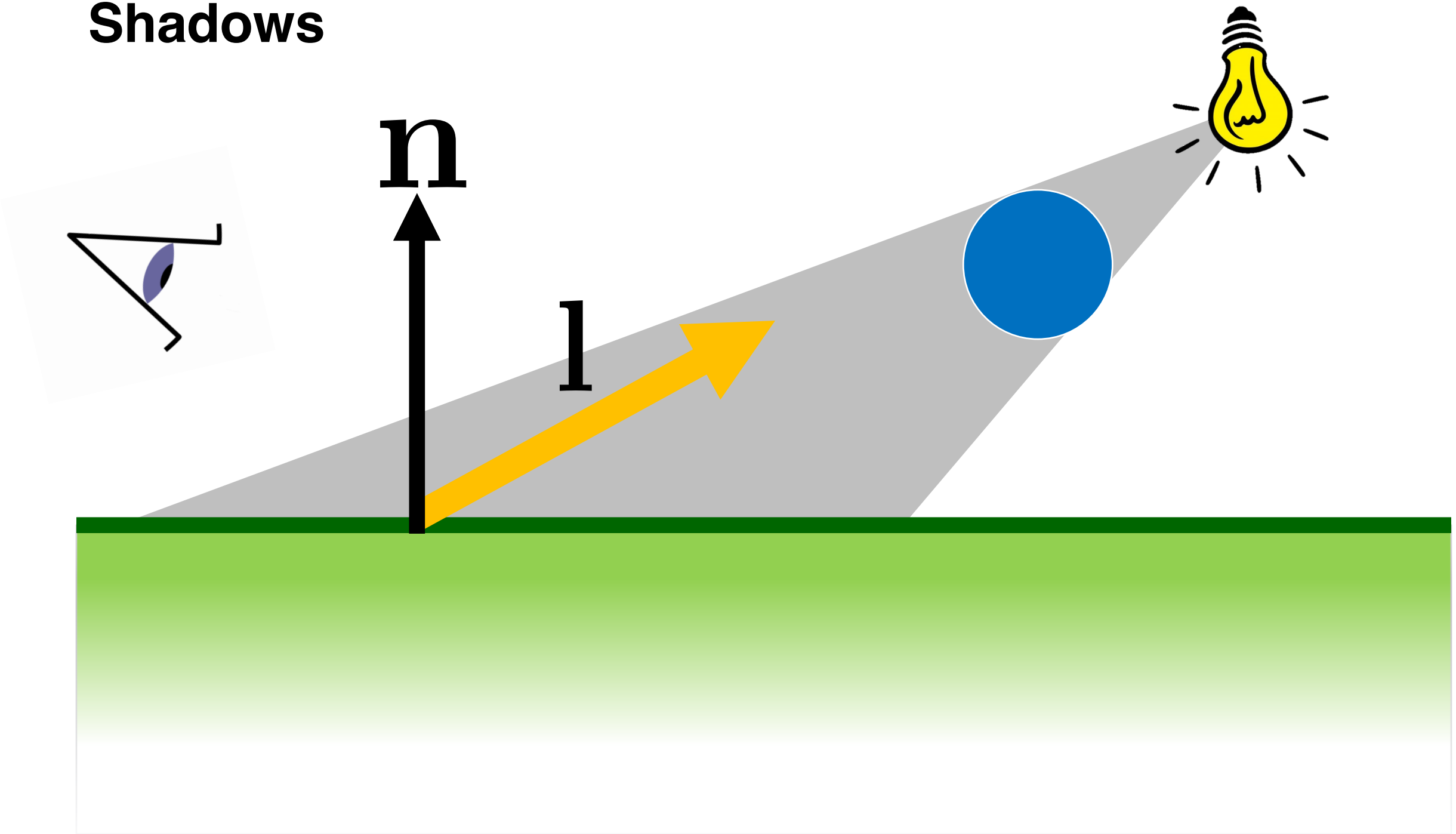
Point Light



Ray Casting

```
for each pixel in the image {
    Generate a ray
    for each object in the scene {
        if (Intersect ray with object) {
            Set pixel colour
        }
    }
}
```

Shadows



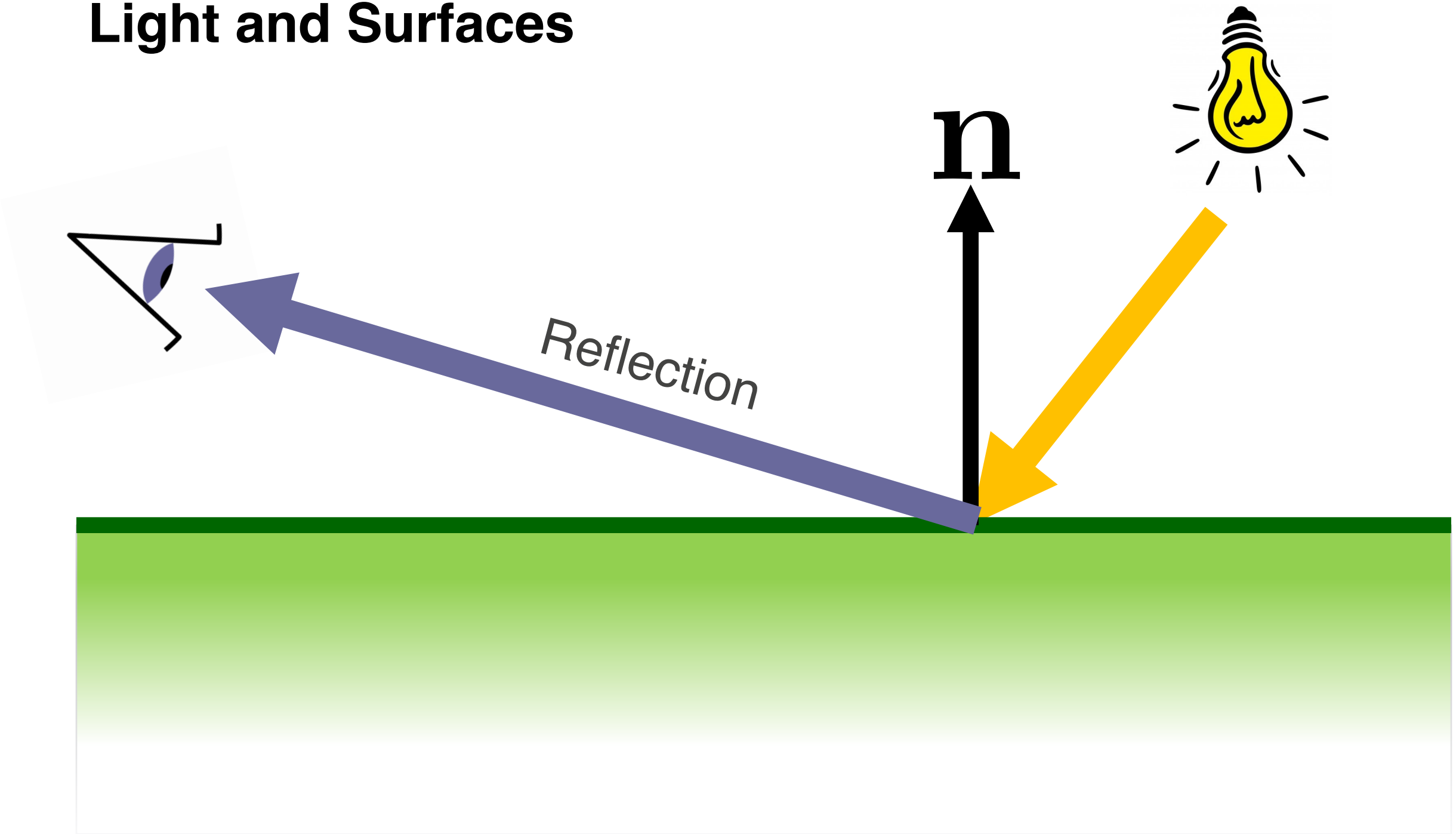
WARNING: Numerical Precision

What are valid values t for the shadow ray?

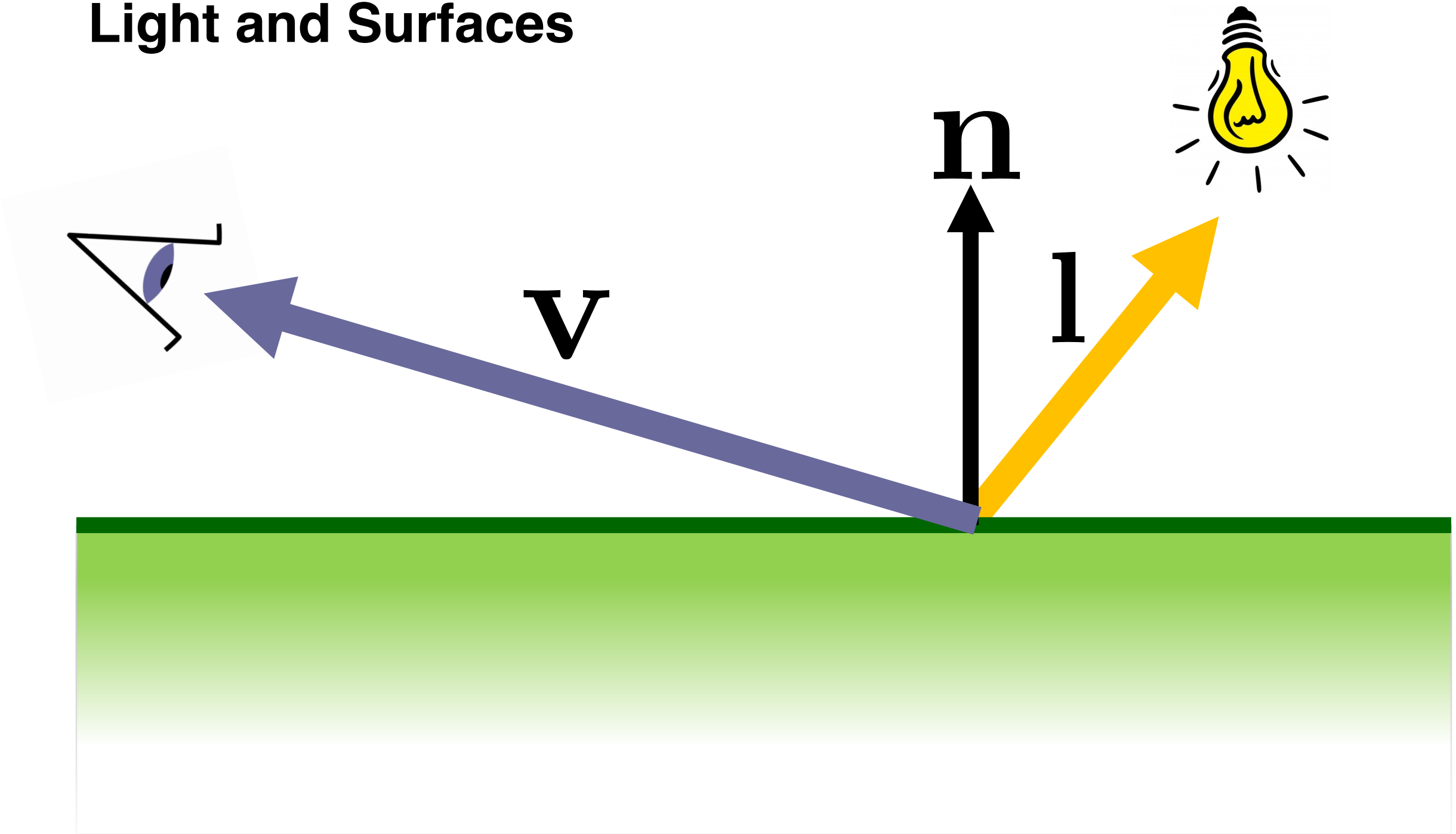
Ray Casting

```
for each pixel in the image {
    Generate a ray
    for each object in the scene {
        if (Intersect ray with object) {
            Set pixel colour
        }
    }
}
```

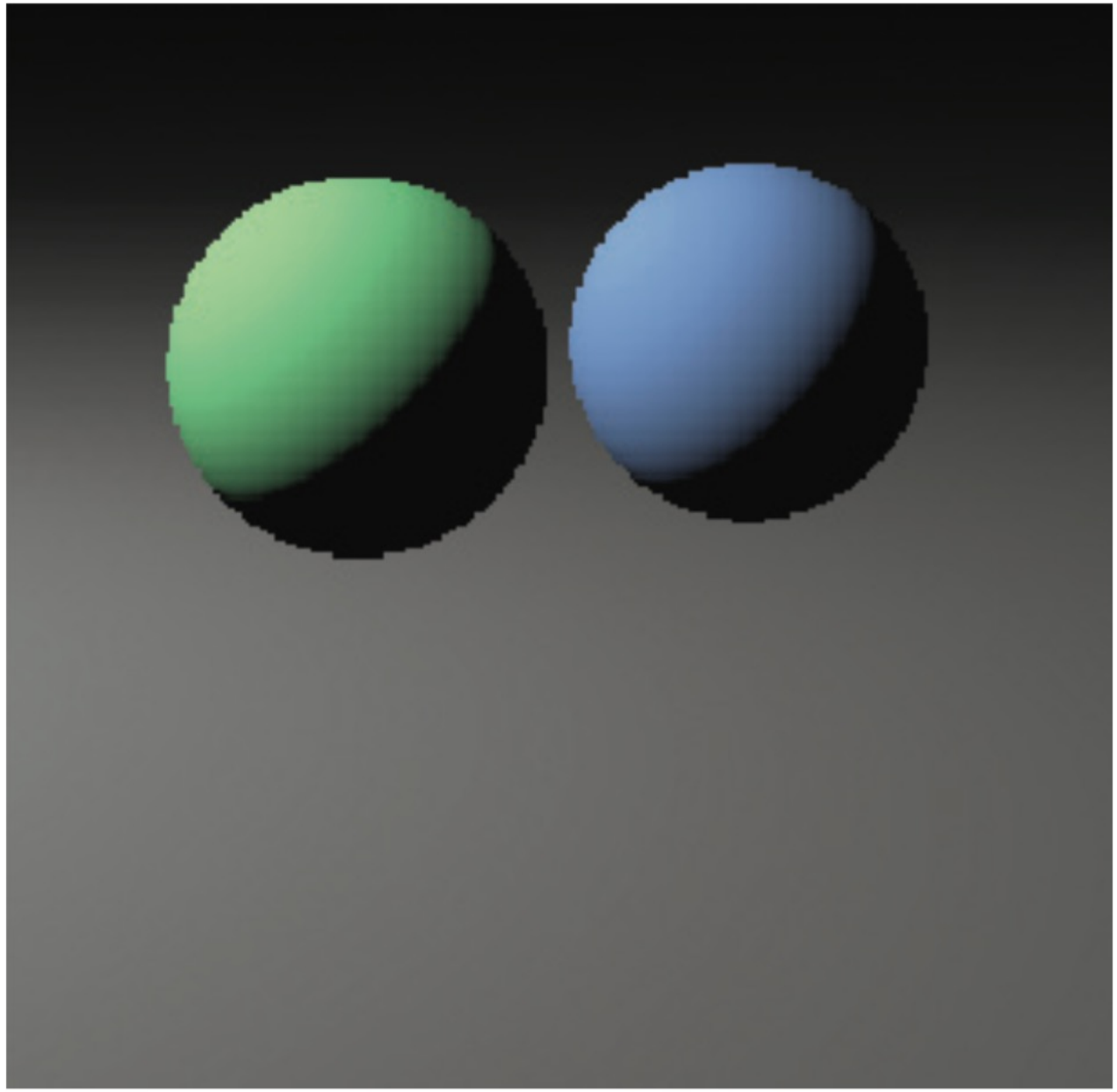
Light and Surfaces



Light and Surfaces



Lambertian (Diffuse) Shading



Lambertian (Diffuse) Shading

the amount of energy from a light source that falls on an area of surface depends on the angle of the surface to the light.

- Lambert (18th century)

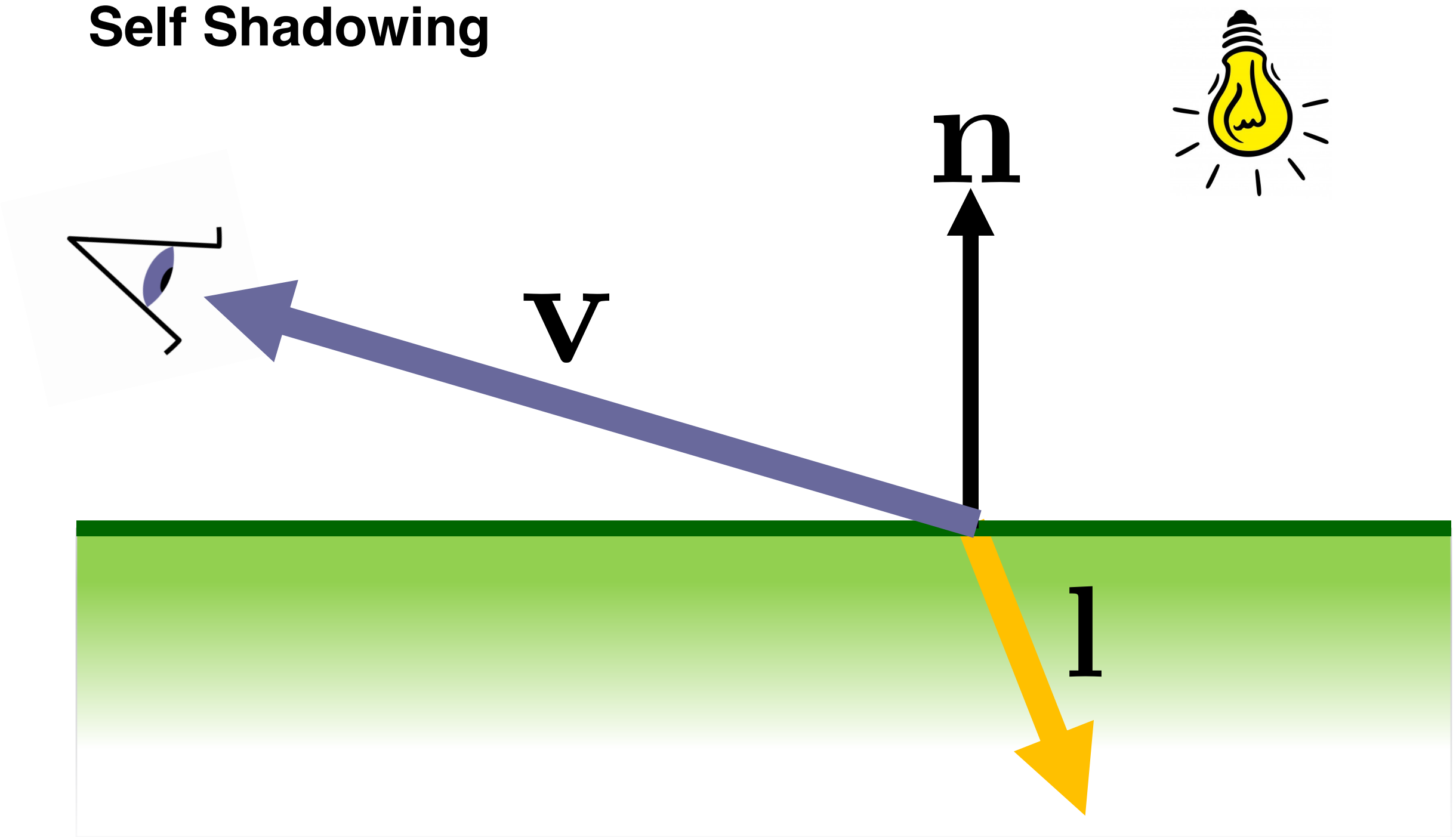
Lambertian (Diffuse) Shading

the amount of energy from a light source that falls on an area of surface depends on the angle of the surface to the light.

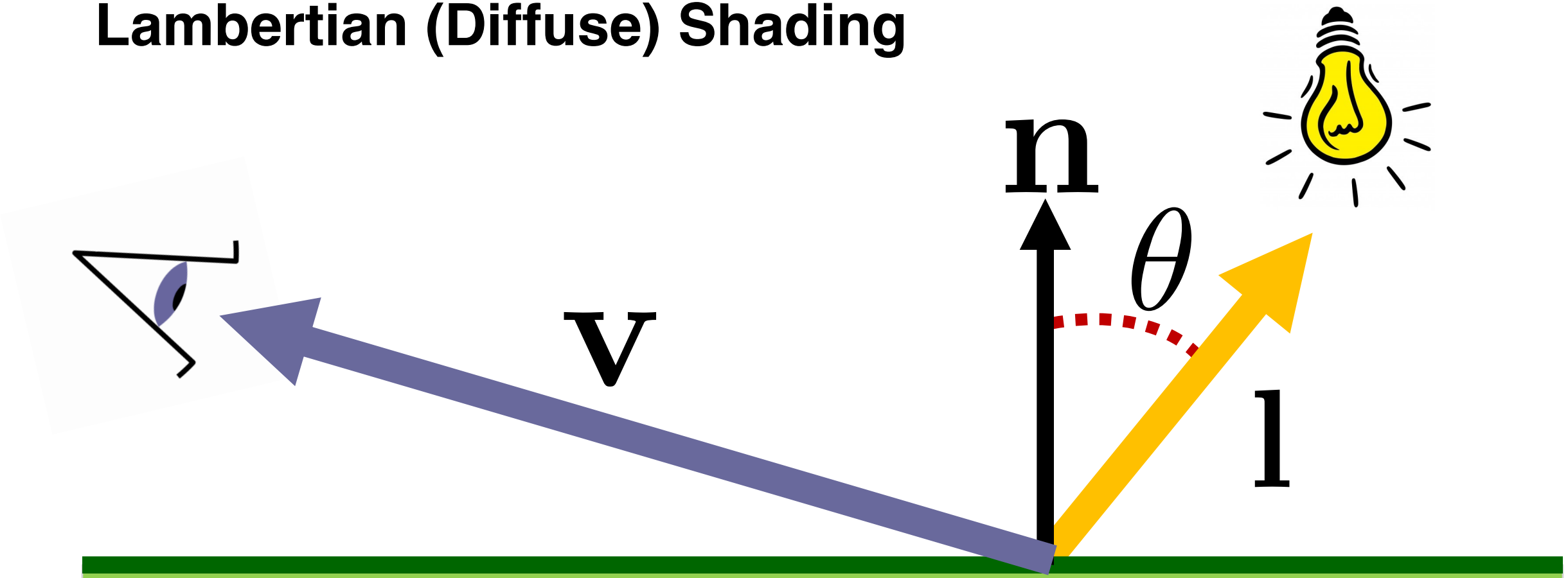
- Lambert (18th century)

$$L = k_d \max\left(0, \mathbf{n}^T \mathbf{l}\right)$$

Self Shadowing

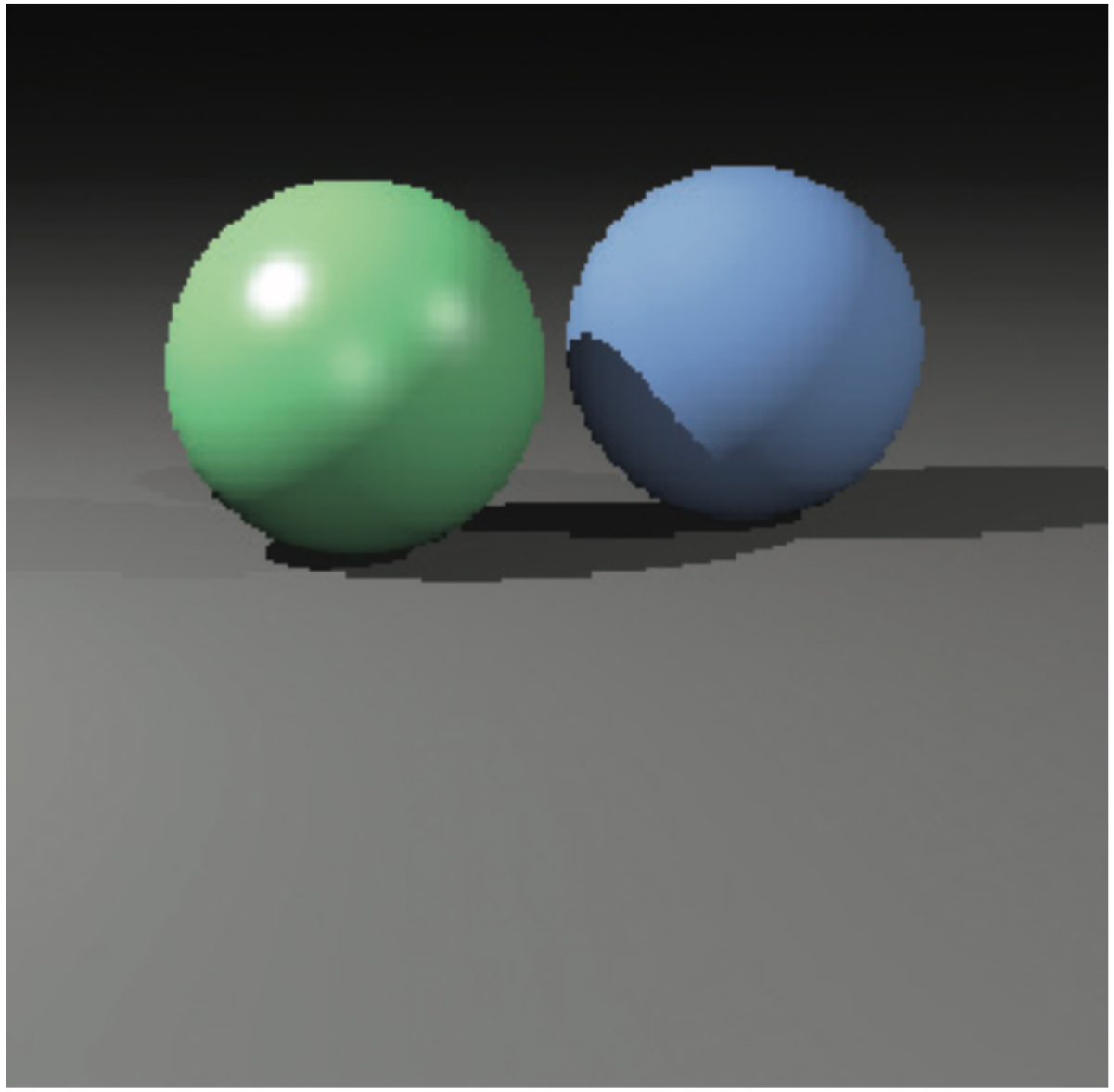


Lambertian (Diffuse) Shading



$$L = k_d \max\left(0, \mathbf{n}^T \mathbf{l}\right)$$

Specular Reflection

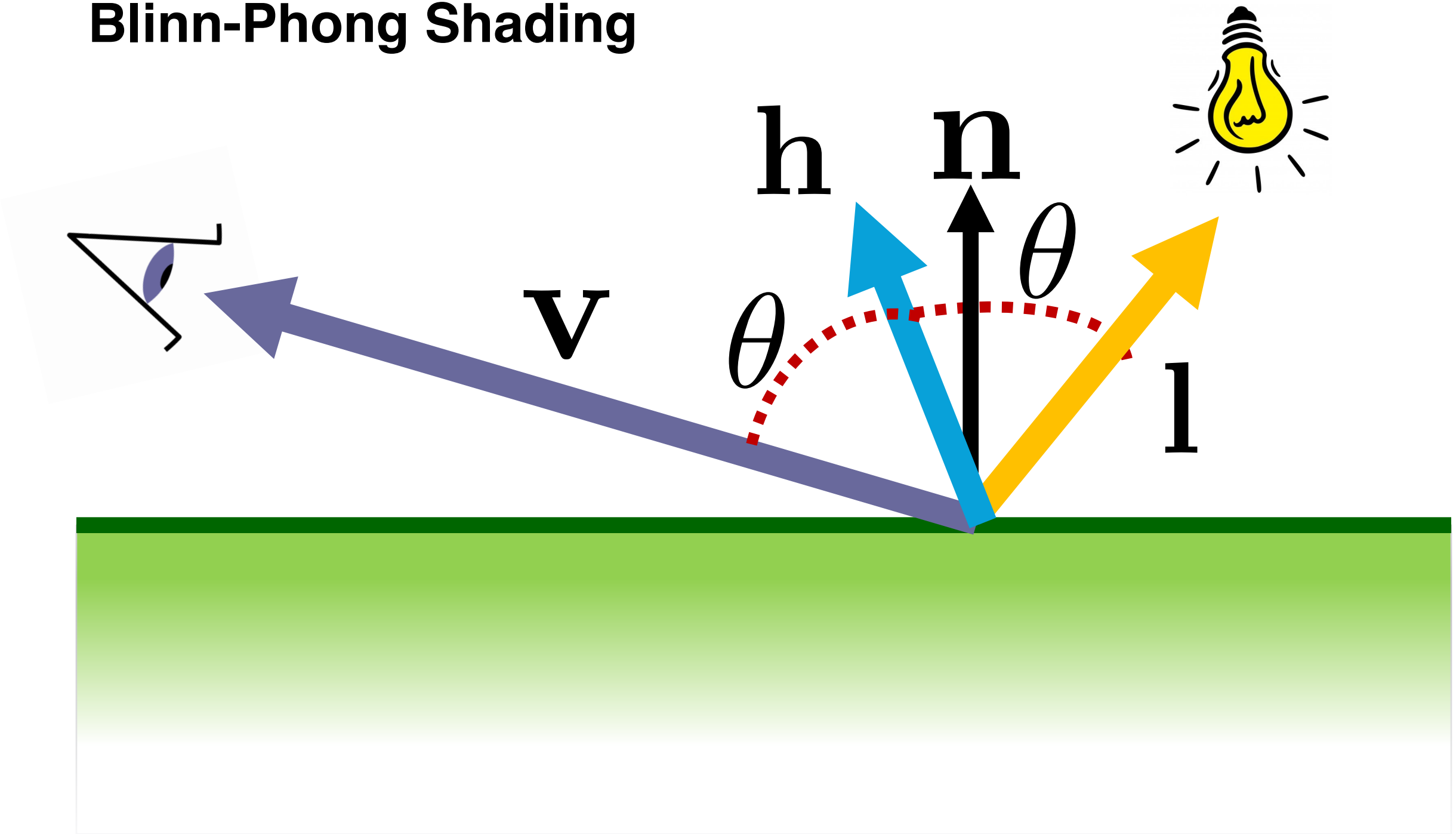


Blinn-Phong Shading Model

“The idea is to produce reflection that is at its brightest when v and l are symmetrically positioned across the surface normal, which is when mirror reflection would occur; the reflection then decreases smoothly as the vectors move away from a mirror configuration. ”

– Marschner and Shirley

Blinn-Phong Shading



The Half Vector

$$\mathbf{h} = \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|}$$

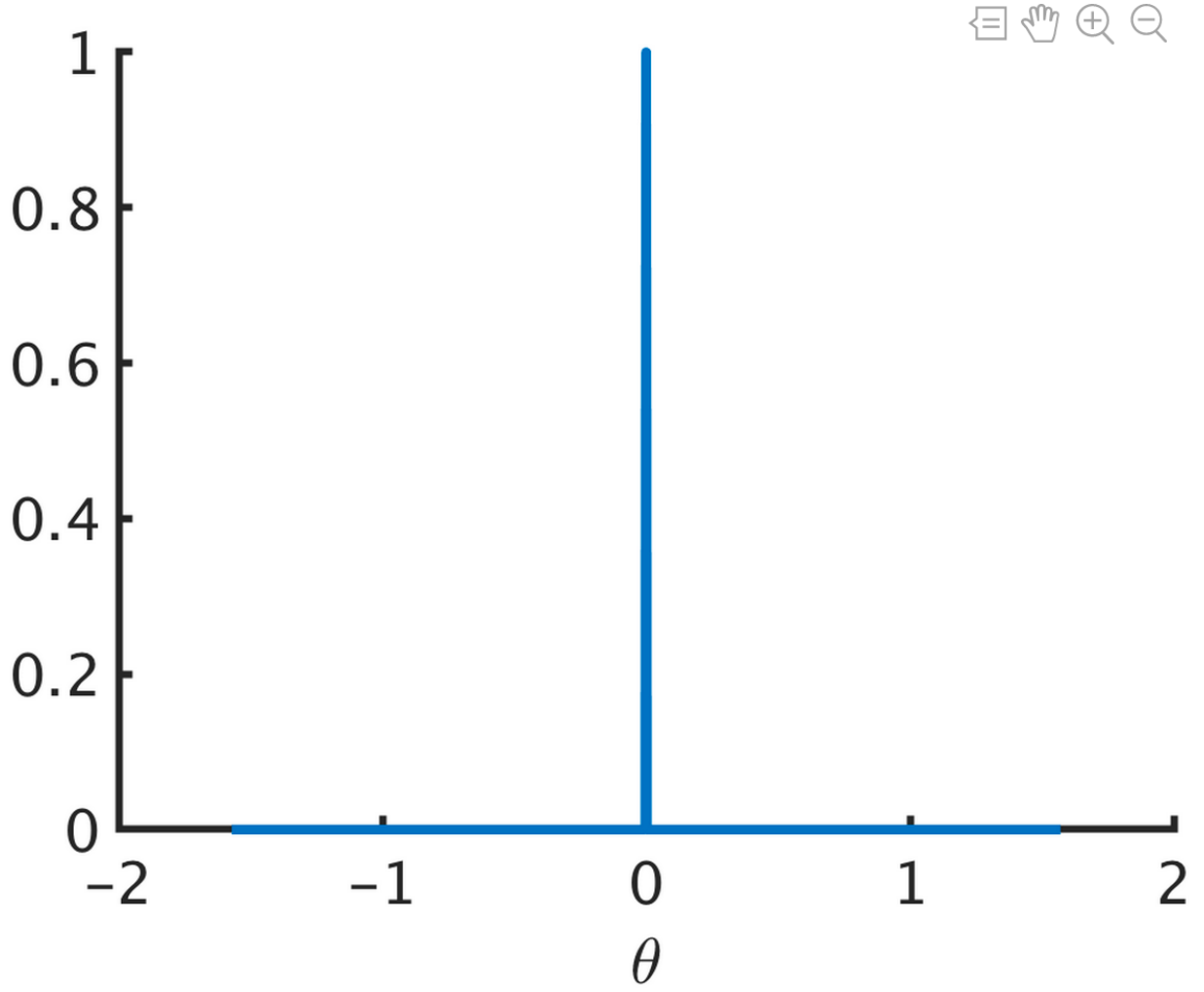
The Half Vector

$$\mathbf{h} = \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|}$$

$$L \propto \theta_{\mathbf{h}}^{\mathbf{n}}$$

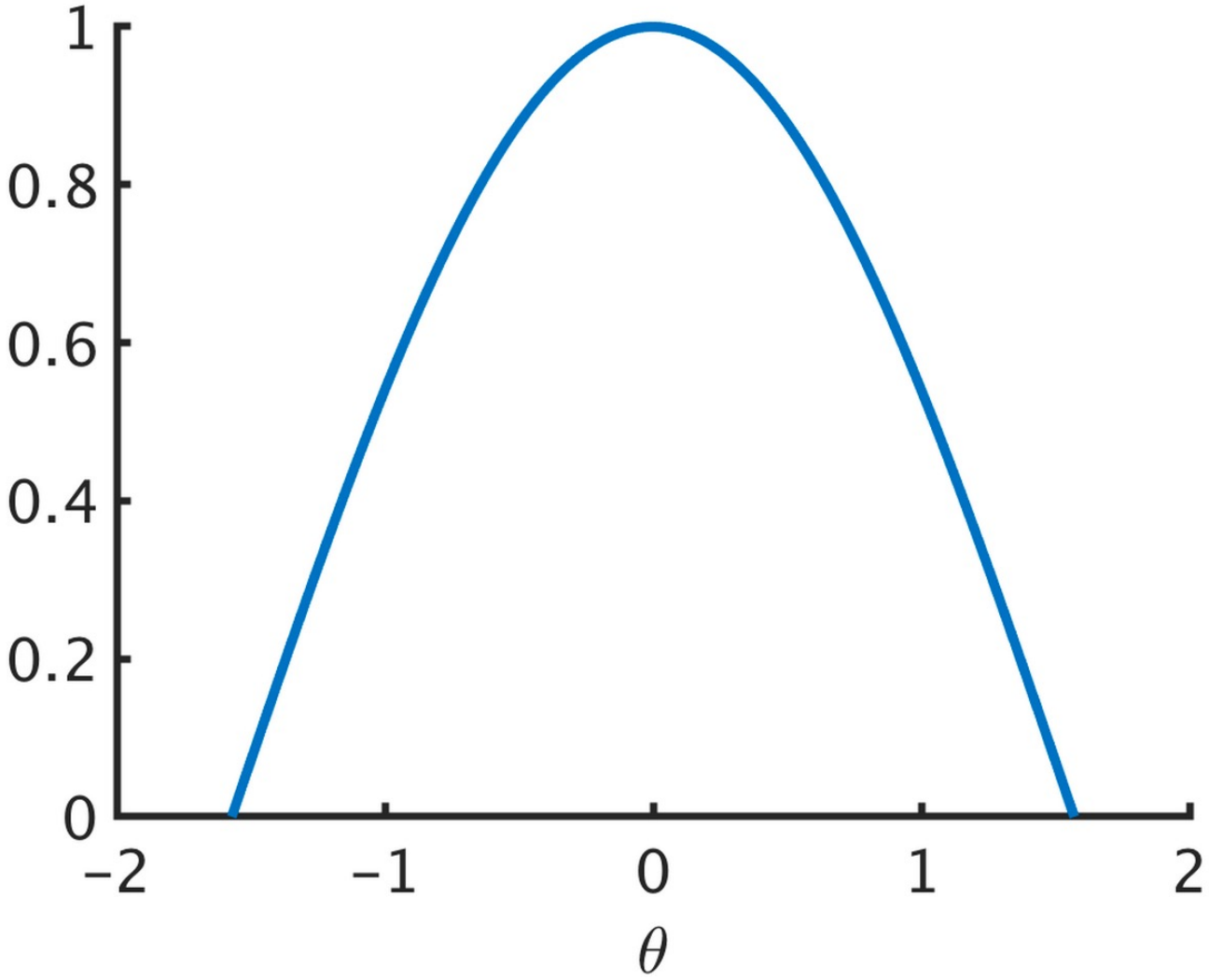
Measuring the Angle

$$L = k_s \max\left(0, \delta(\theta_{\mathbf{h}}^{\mathbf{n}})\right)$$



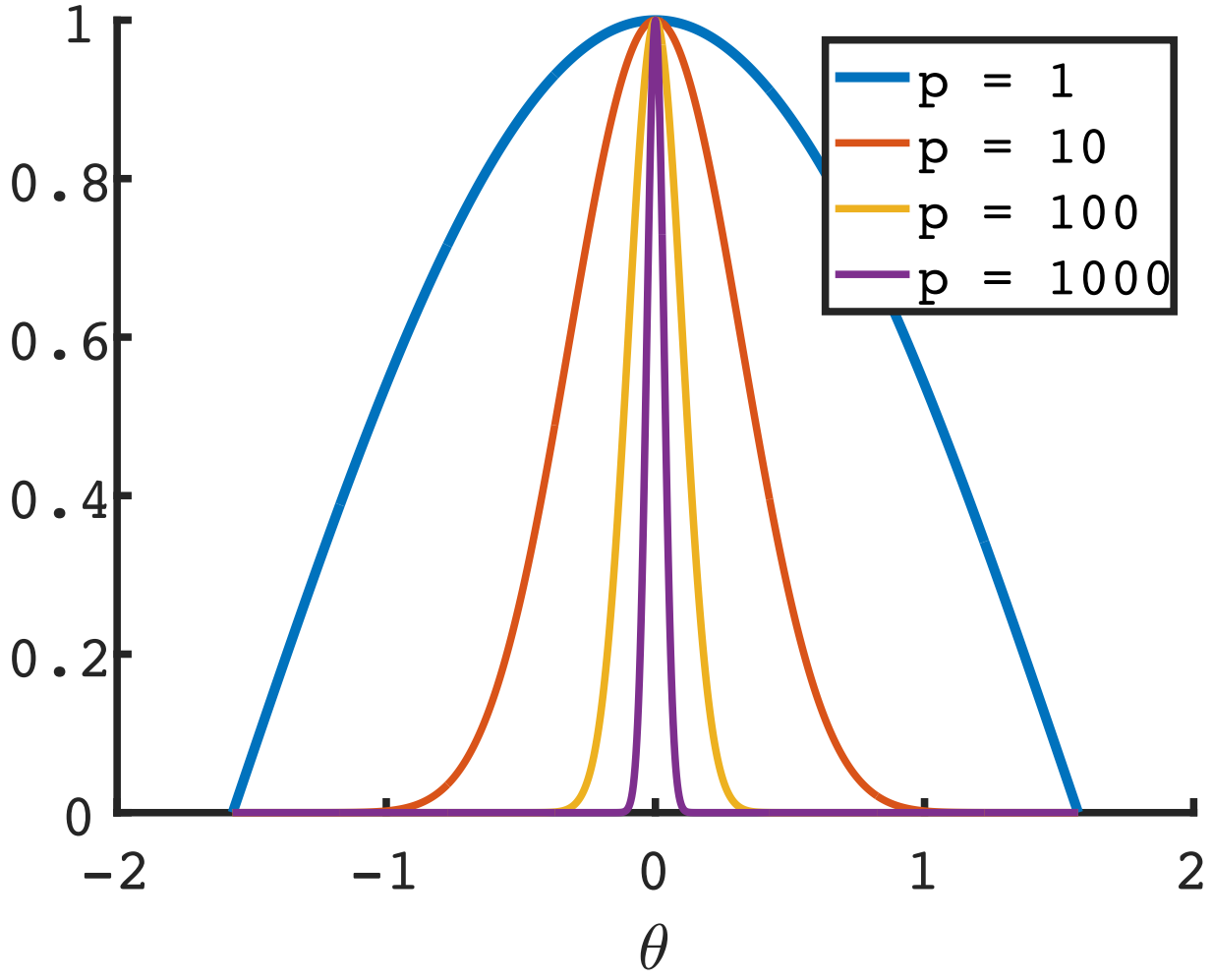
Measuring the Angle

$$L = k_s \max\left(0, \cos(\theta_{\mathbf{h}}^{\mathbf{n}})\right)$$



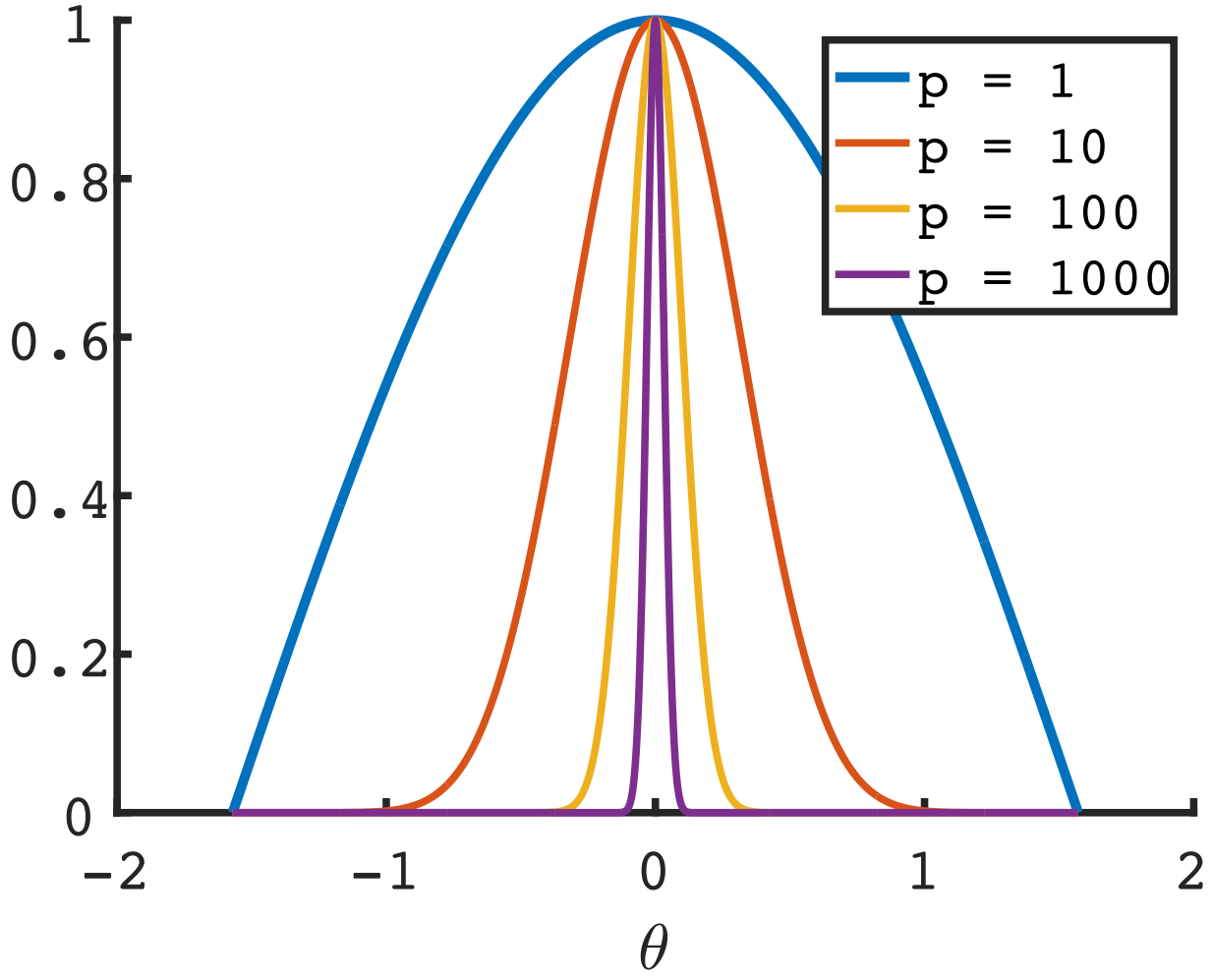
Measuring the Angle

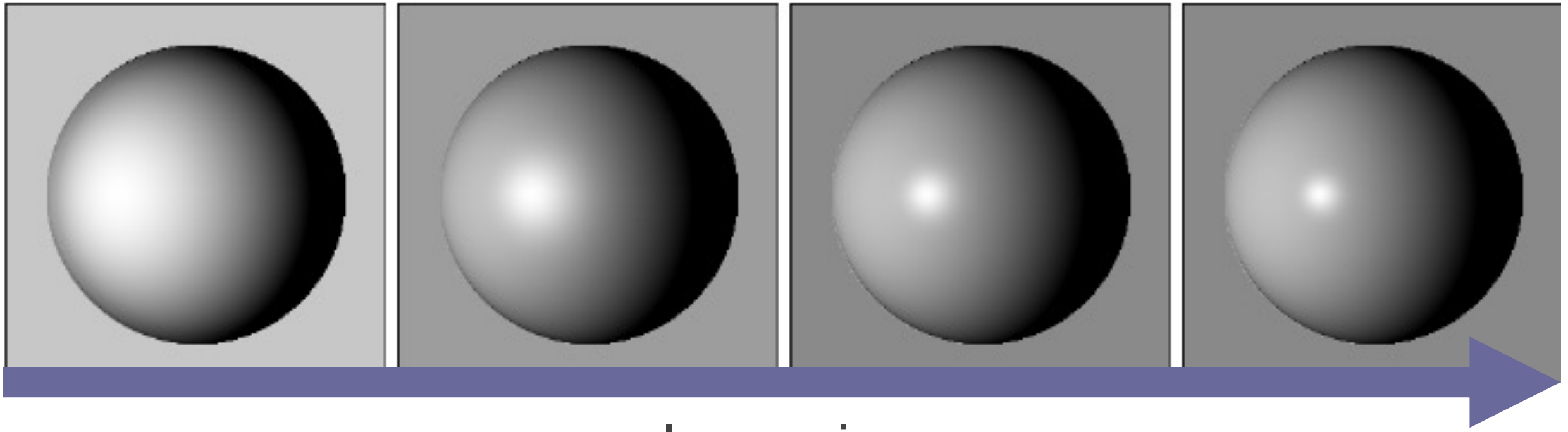
$$L = k_s \max\left(0, \cos(\theta_{\mathbf{h}}^{\mathbf{n}})\right)^p$$



Measuring the Angle

$$L = k_s \max\left(0, \mathbf{n}^T \mathbf{h}\right))^p$$





Increasing p

Putting it All Together: The Full Blinn-Phong Model

Light obeys the superposition principle

Total amount of received light is sum of light from all incoming sources.

$$L = \text{Lambertian} + \text{Specular}$$

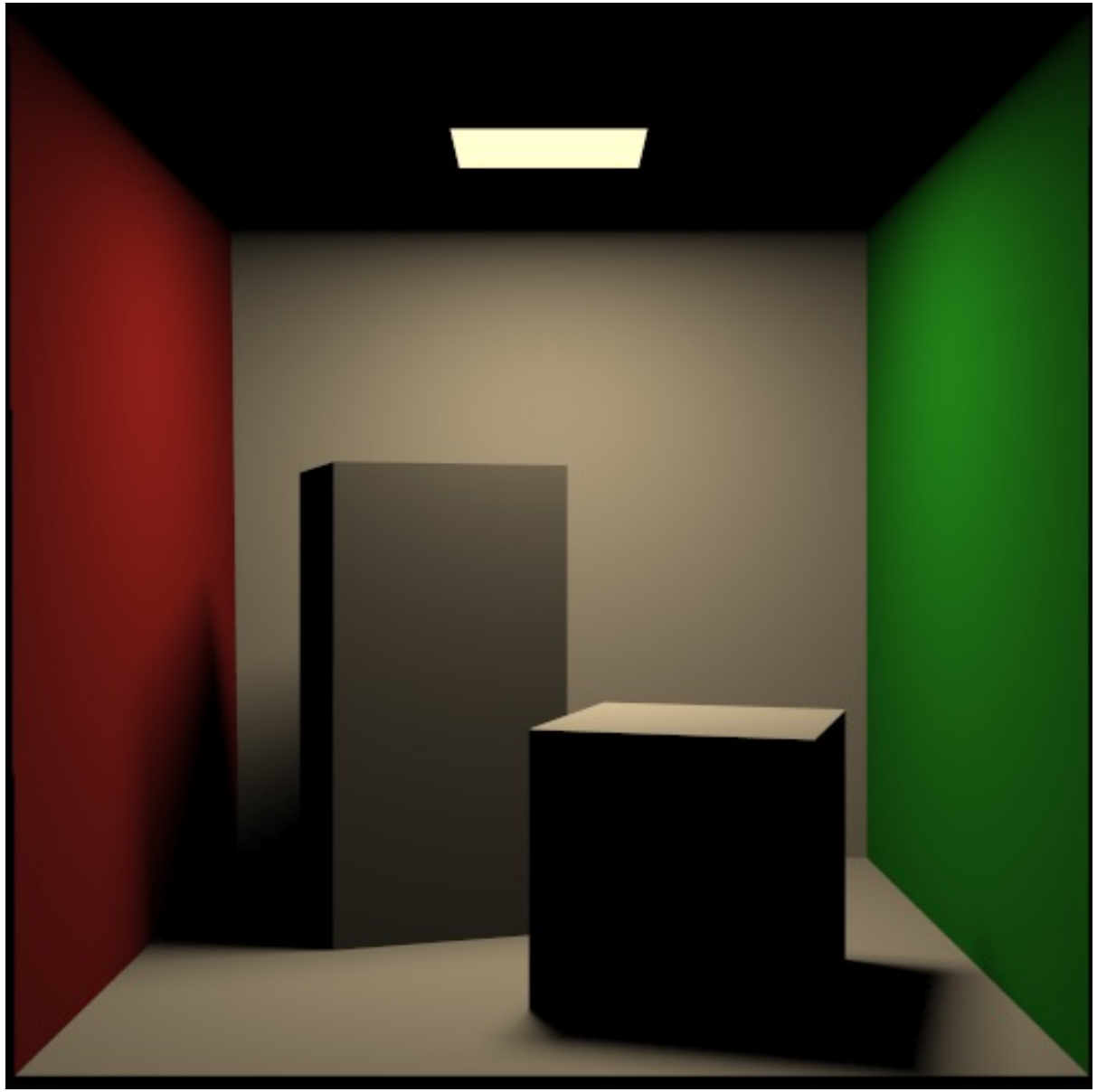
Putting it All Together: The Full Blinn-Phong Model

Light obeys the superposition principle

Total amount of received light is sum of light from all incoming sources.

$$L = k_d \max\left(0, \mathbf{n}^T \mathbf{l}\right) + k_s \max\left(0, \cos(\theta_{\mathbf{h}}^{\mathbf{n}})\right)^p$$

No Global Effects

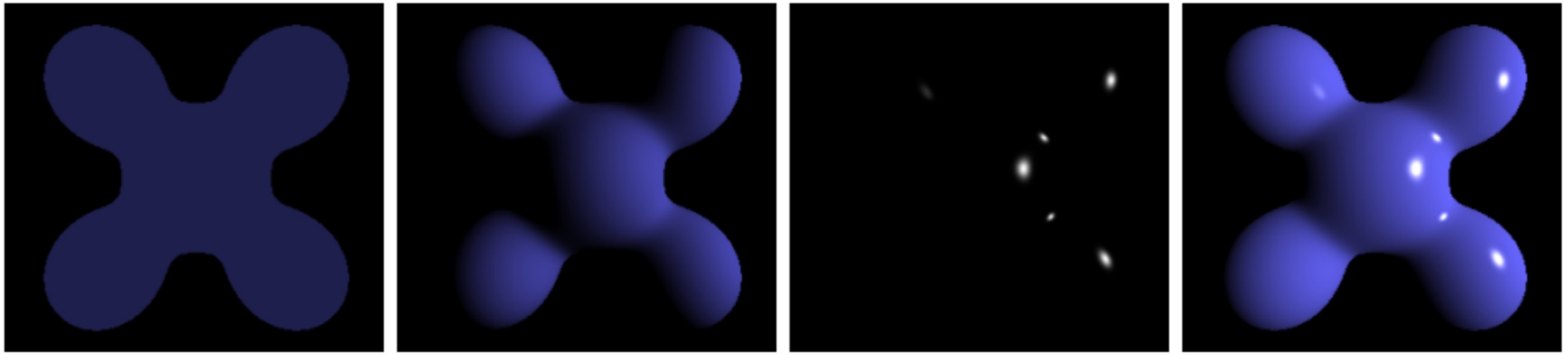


Putting it All Together: The Full Blinn-Phong Model

Light obeys the superposition principle

Total amount of received light is sum of light from all incoming sources.

$$L = k_a + k_d \max\left(0, \mathbf{n}^T \mathbf{l}\right) + k_s \max\left(0, \cos(\theta_{\mathbf{h}}^{\mathbf{n}})\right)^p$$



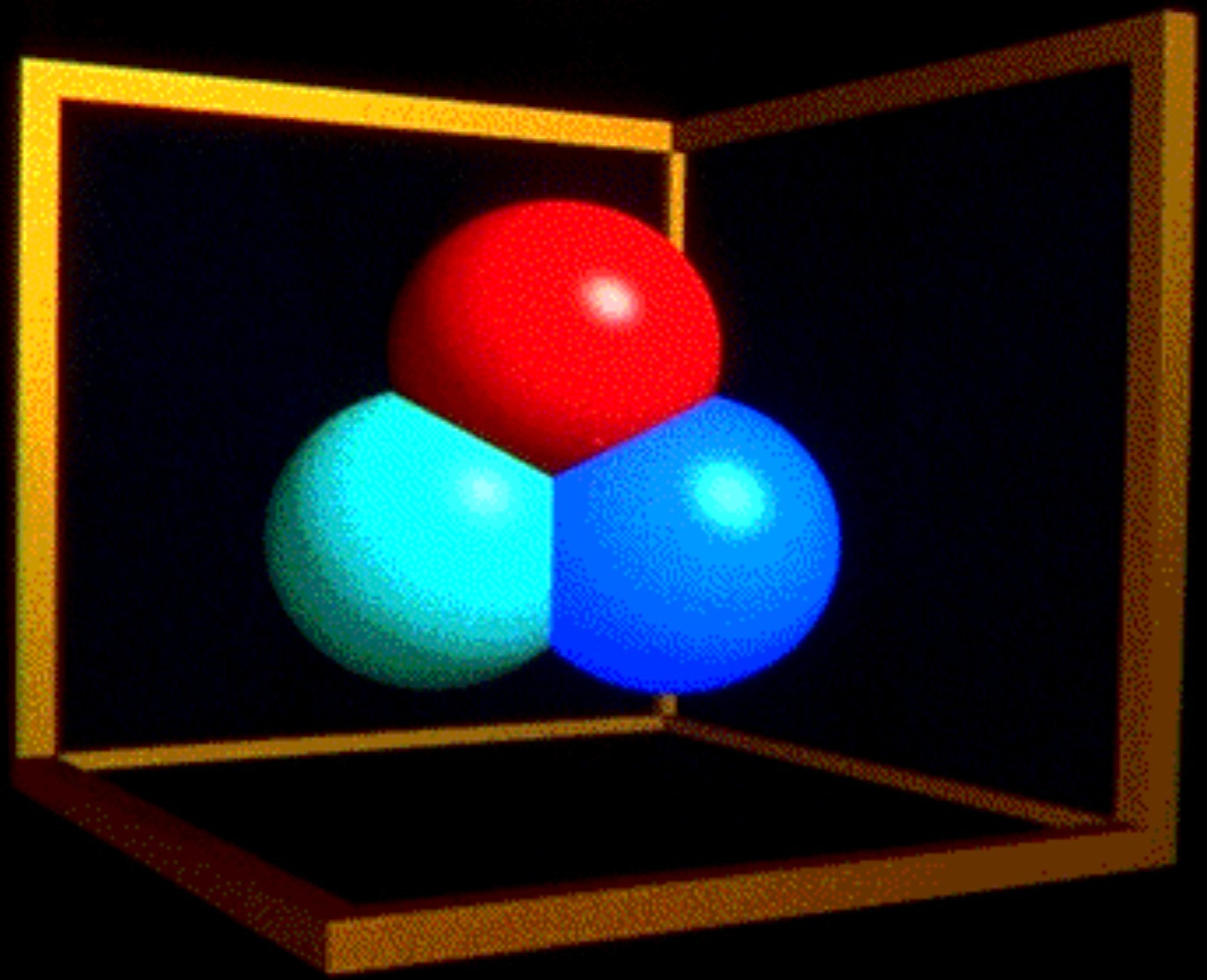
Ambient Diffuse Specular Final

Wikipedia | Brad Smith

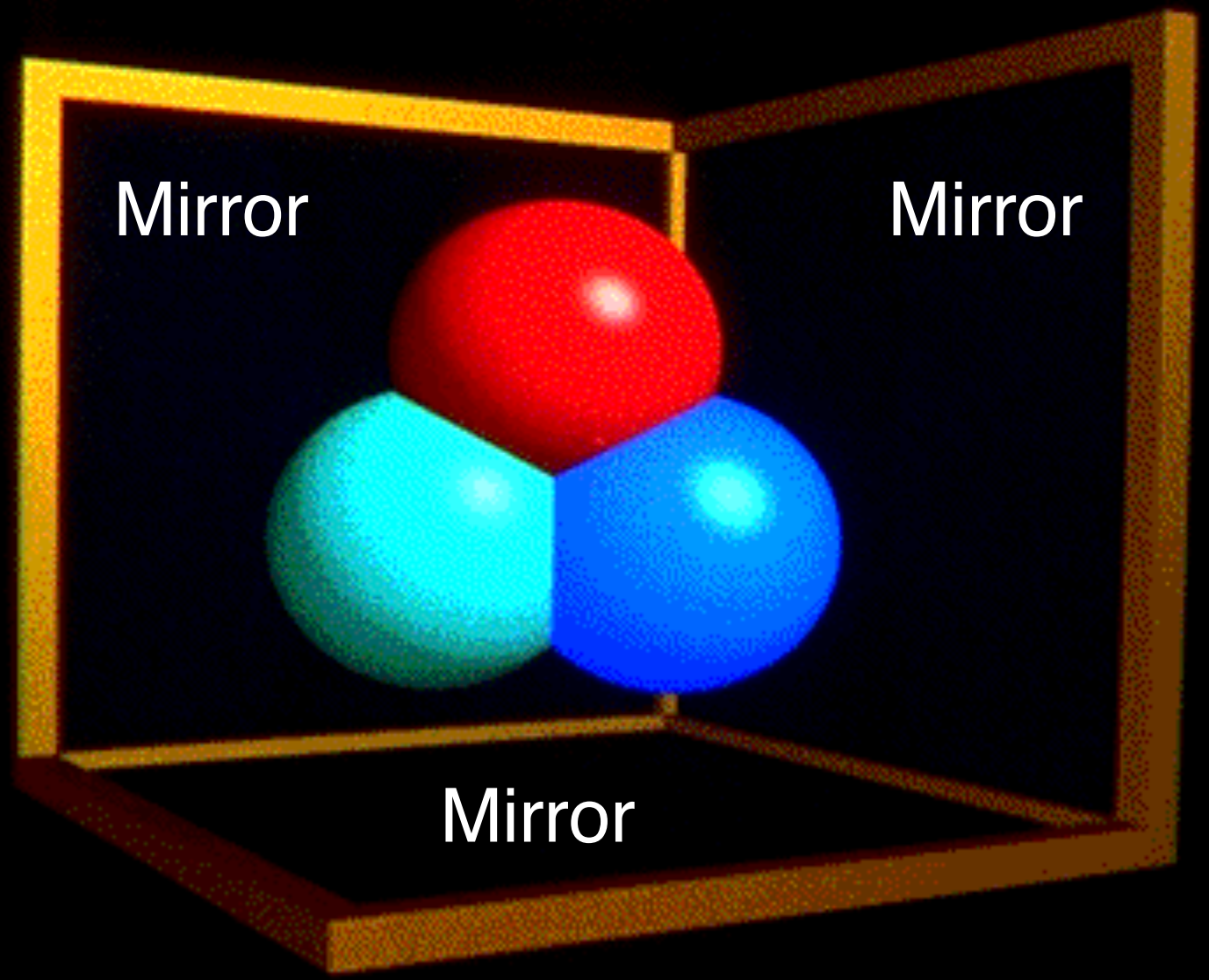
Ray Casting

```
for each pixel in the image {
    Generate a ray
    for each object in the scene {
        if (Intersect ray with object) {
            Set pixel colour
        }
    }
}
```

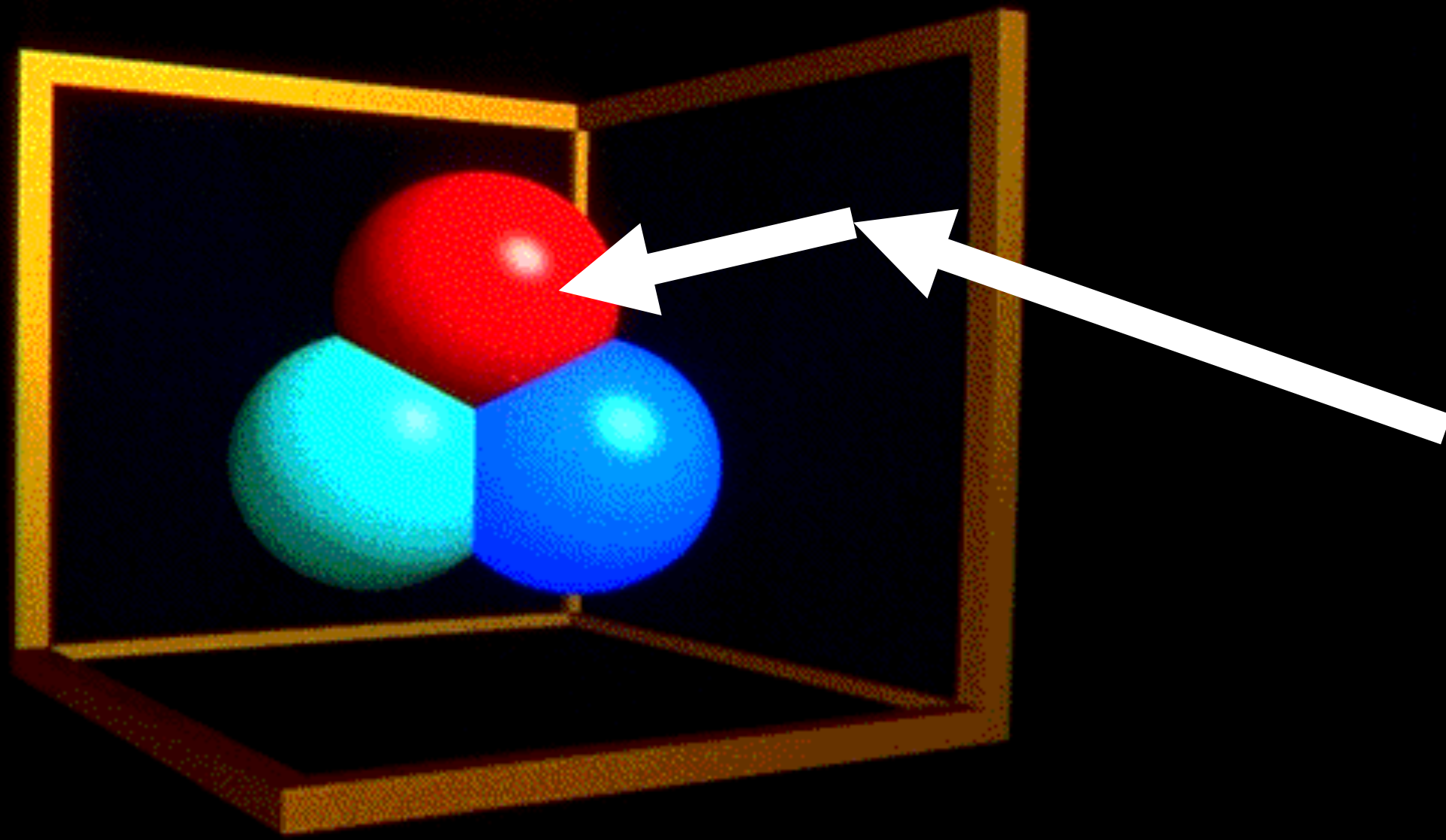
Ray Traced Image



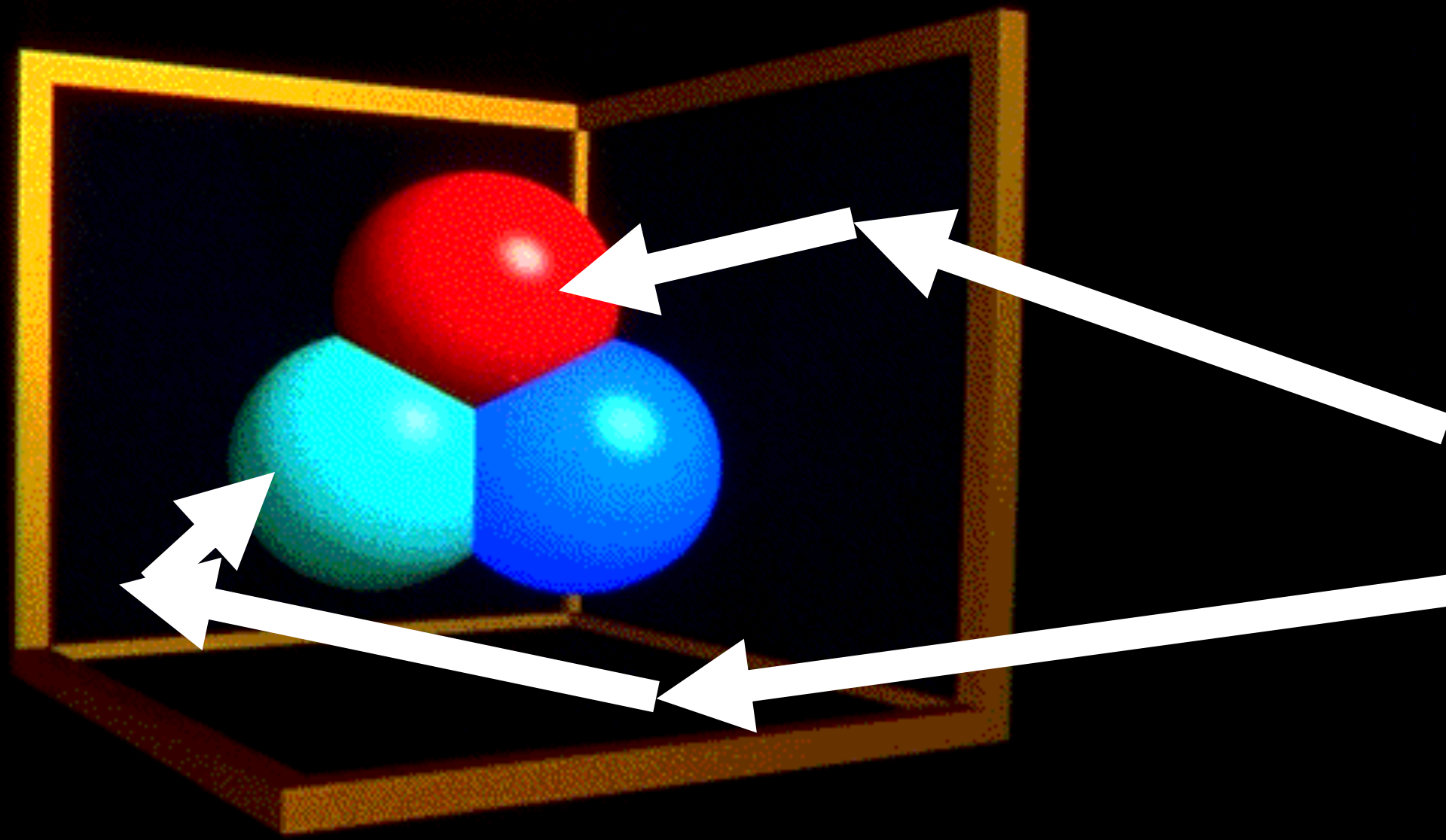
Ray Traced Image

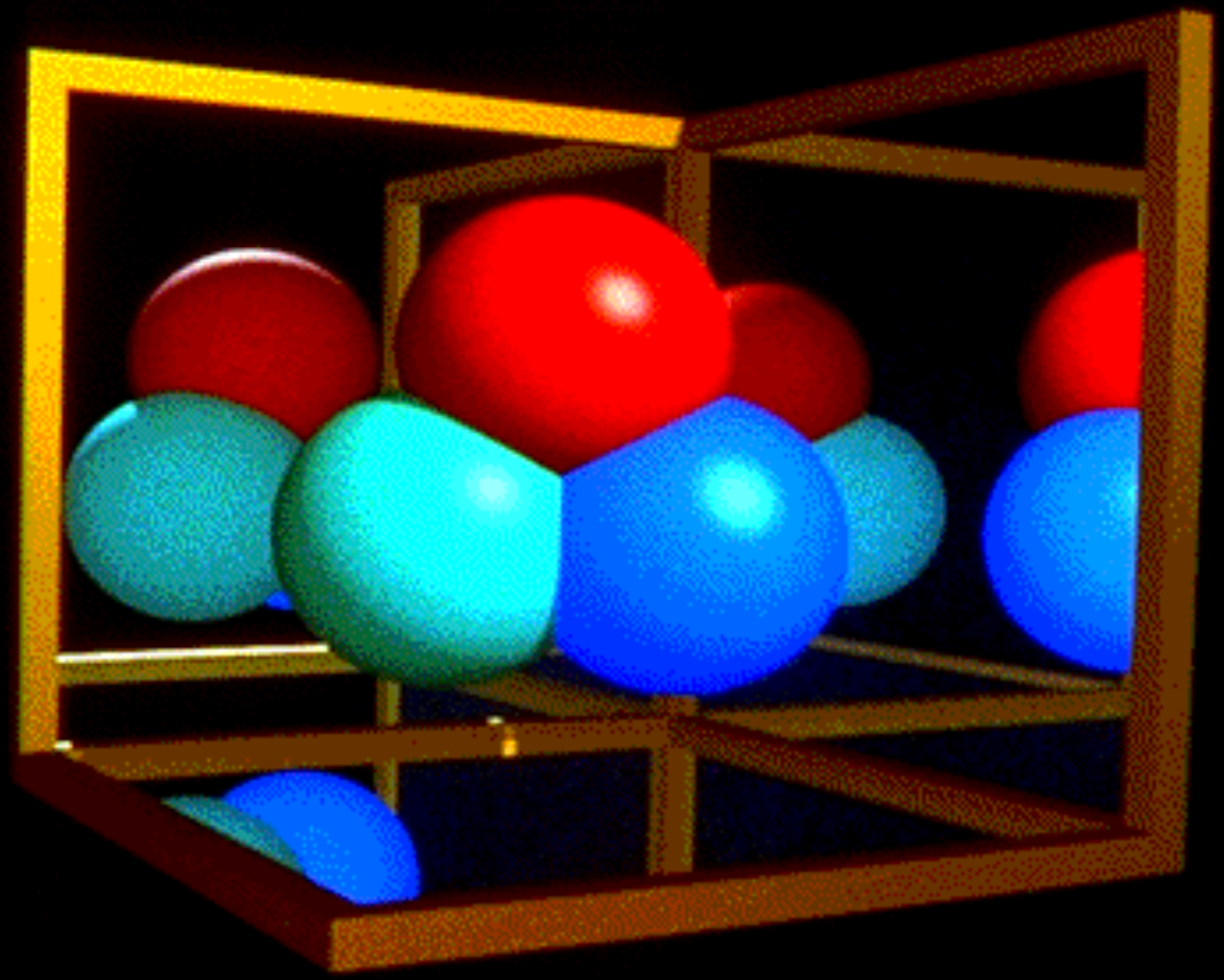


Recursive Ray Tracing

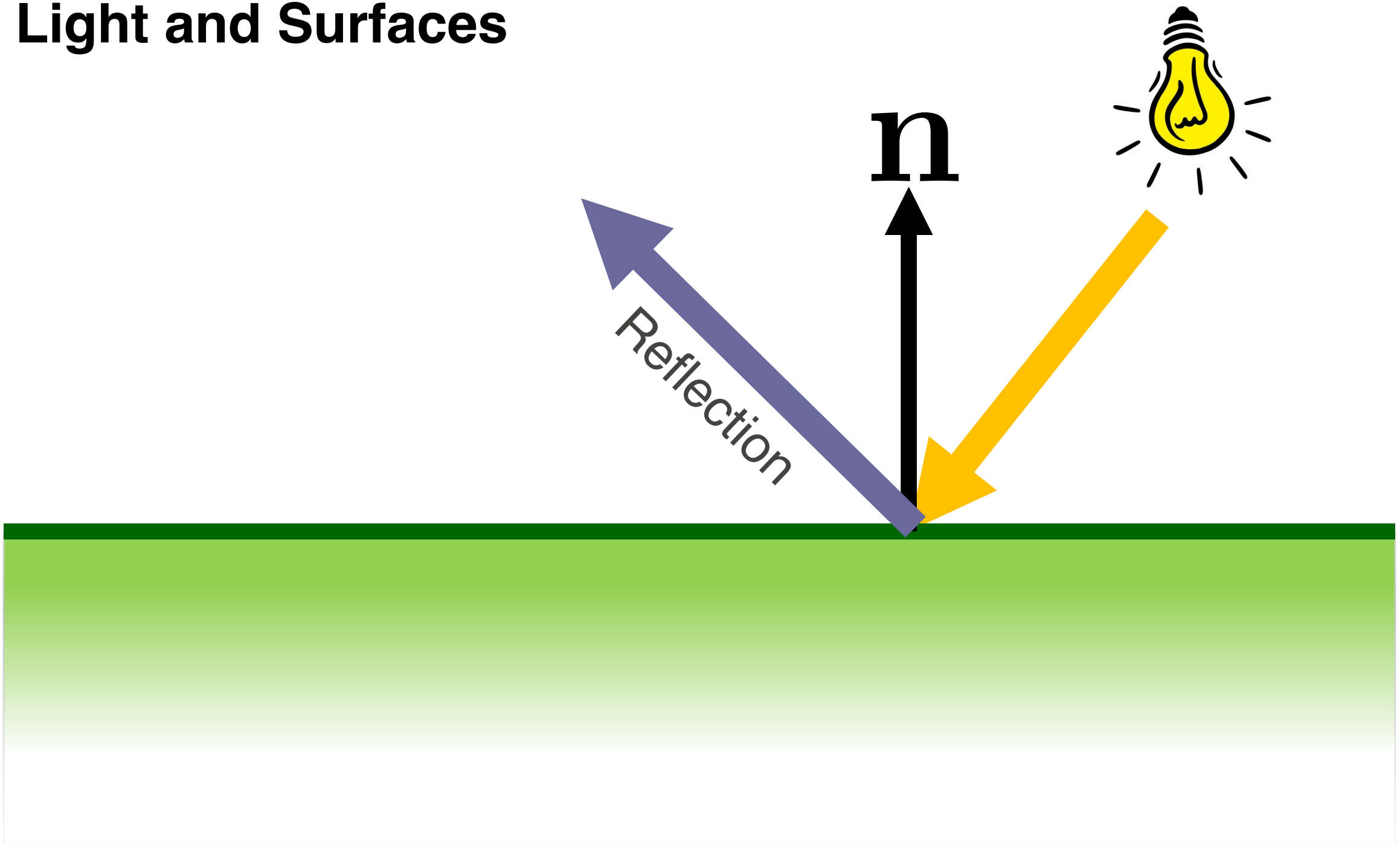


Recursive Ray Tracing

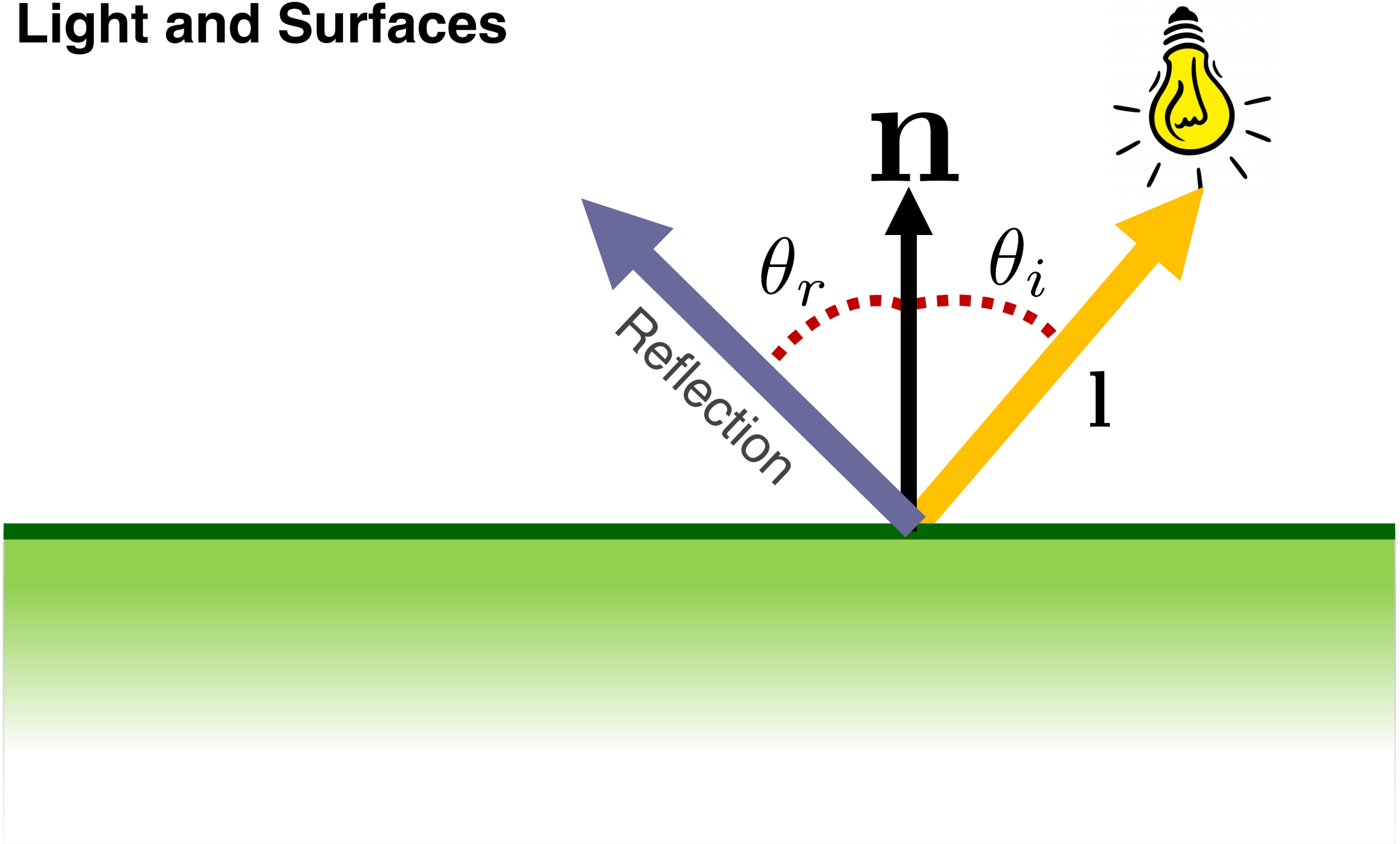




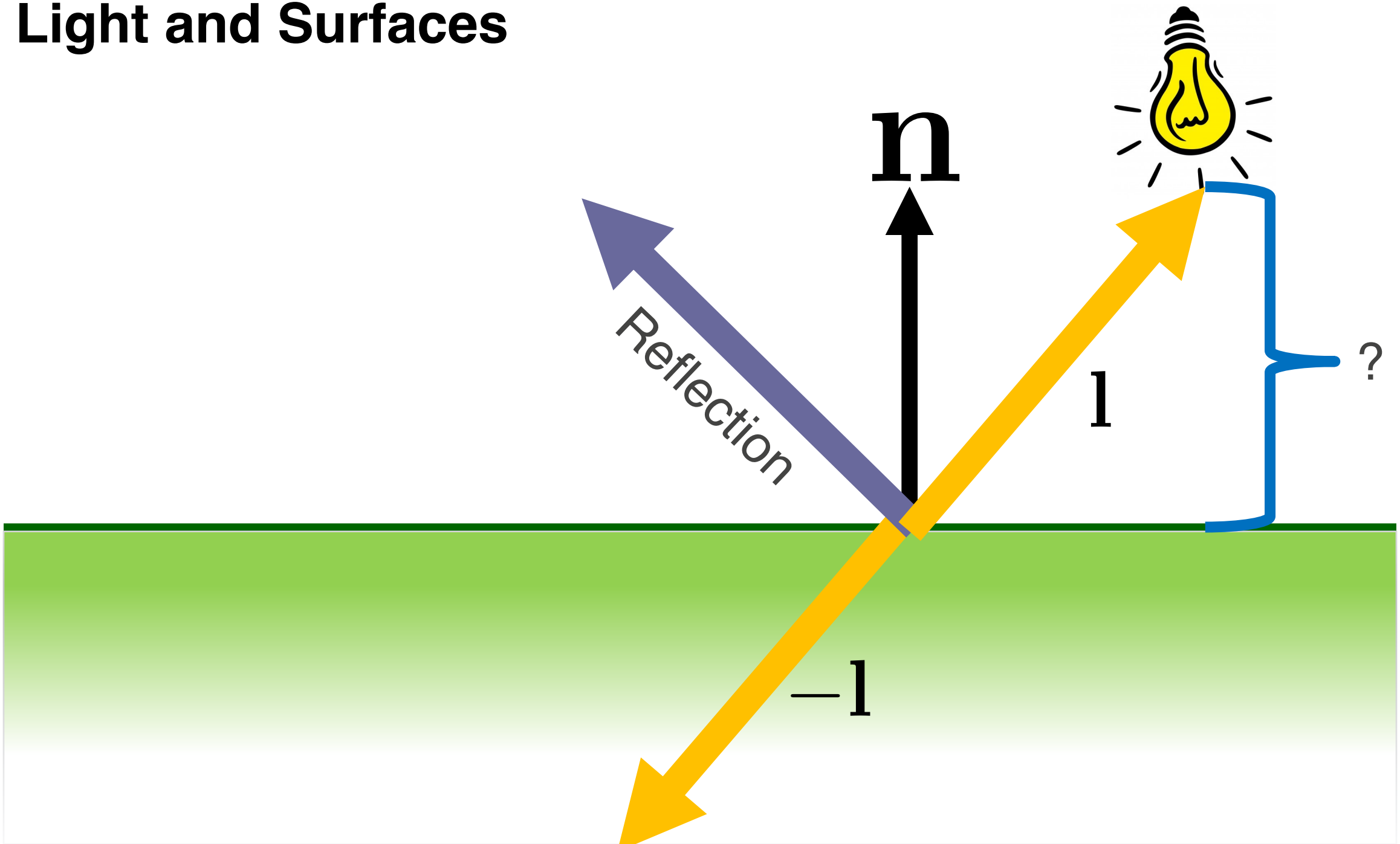
Light and Surfaces



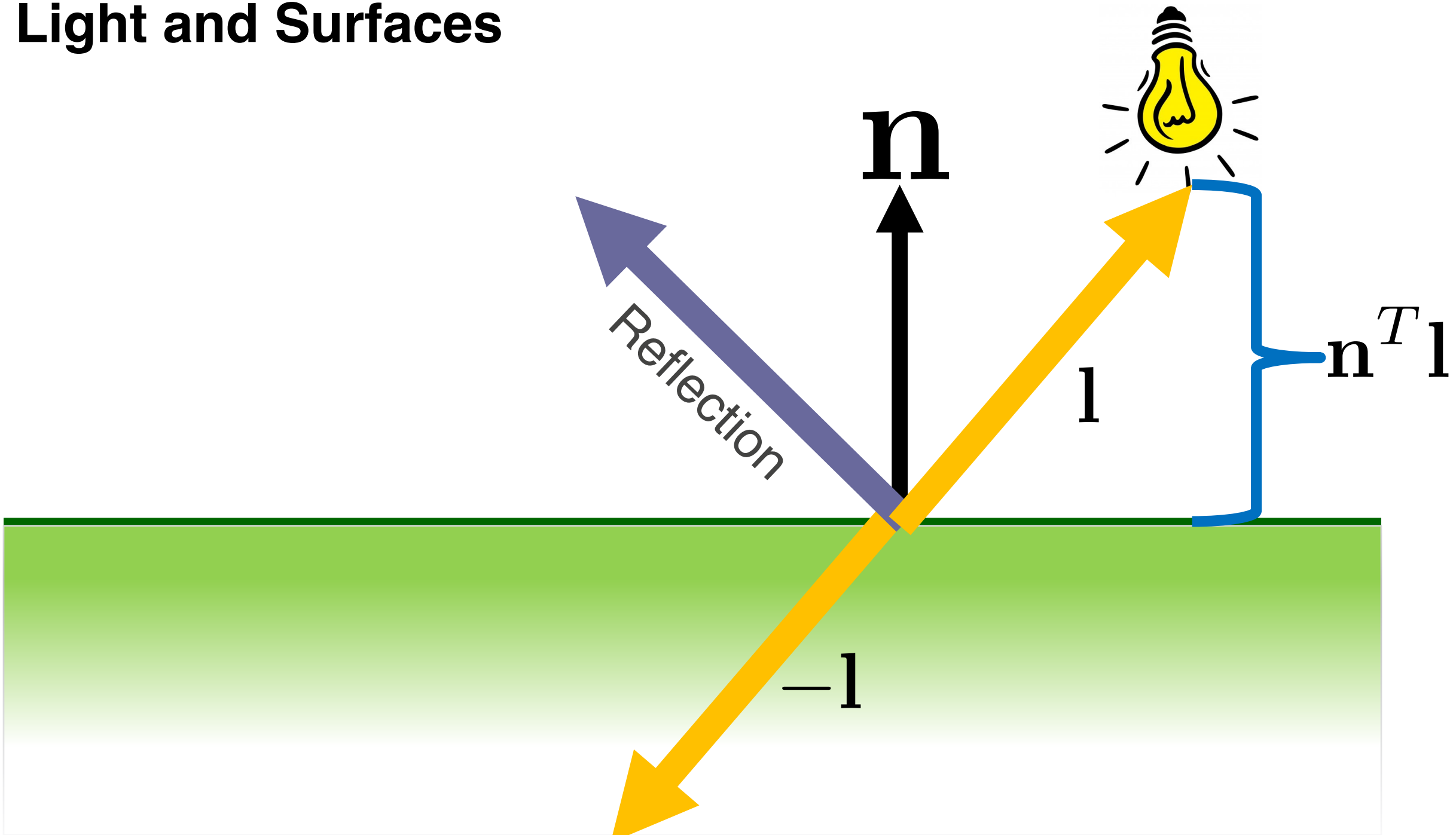
Light and Surfaces



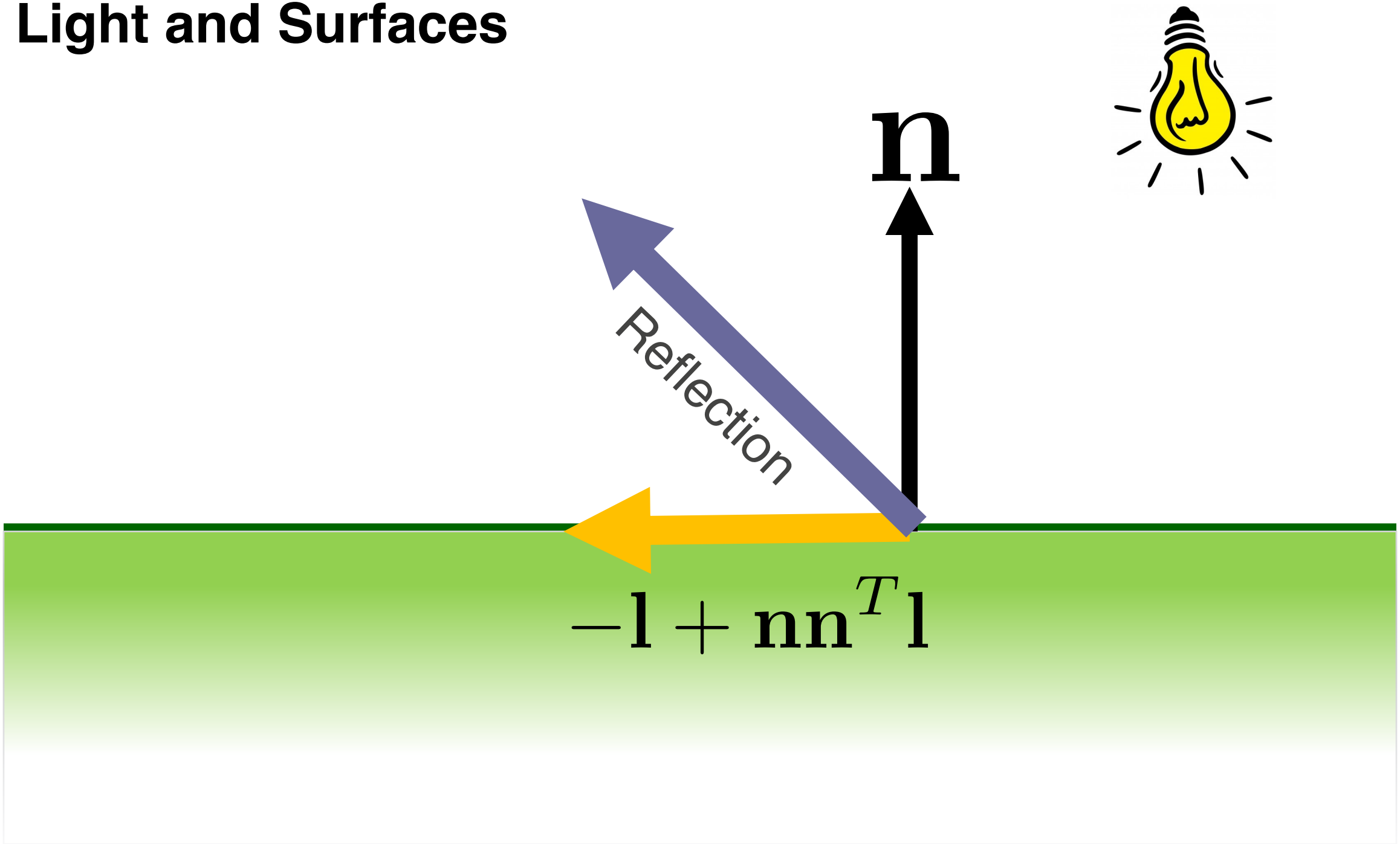
Light and Surfaces



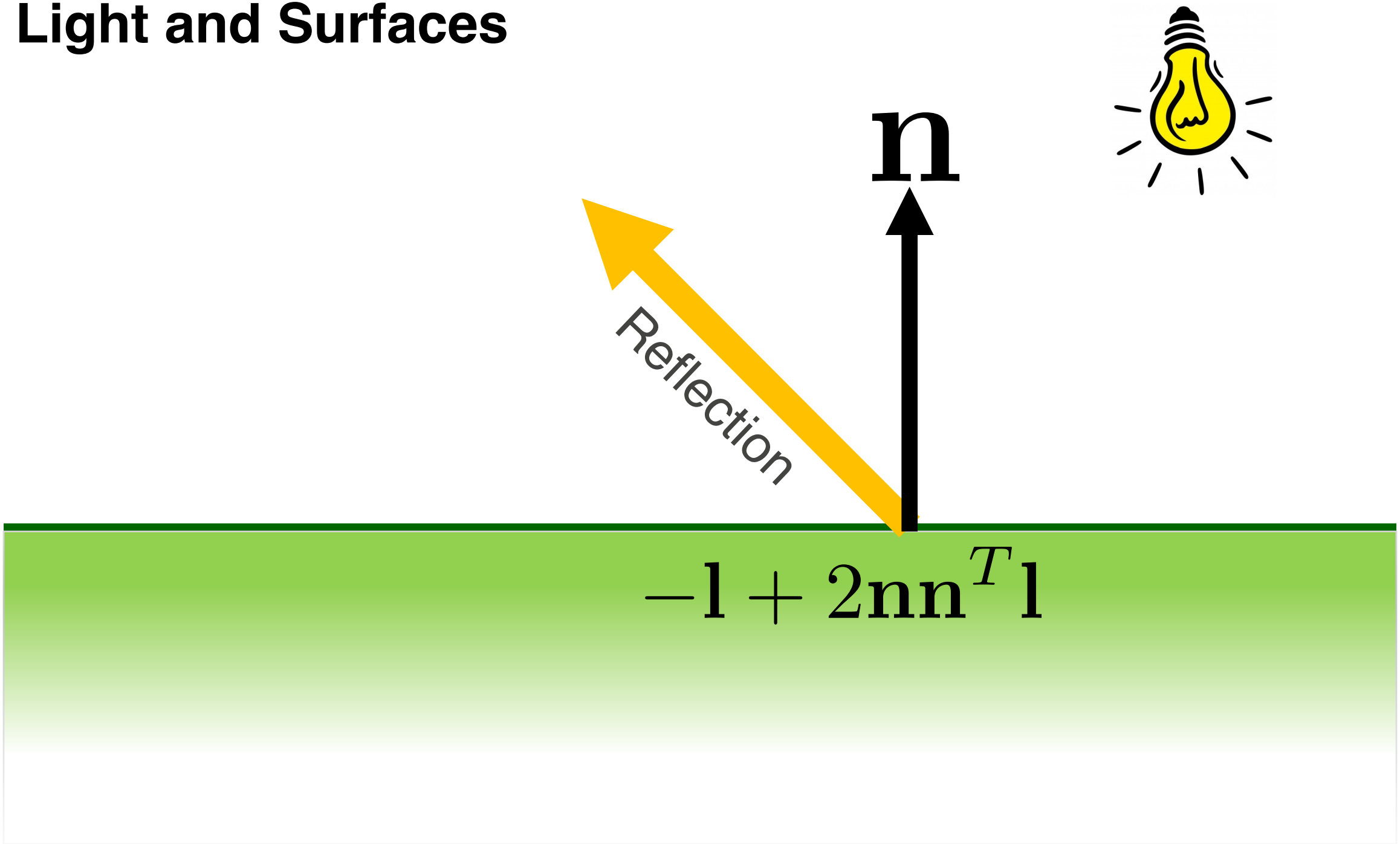
Light and Surfaces



Light and Surfaces



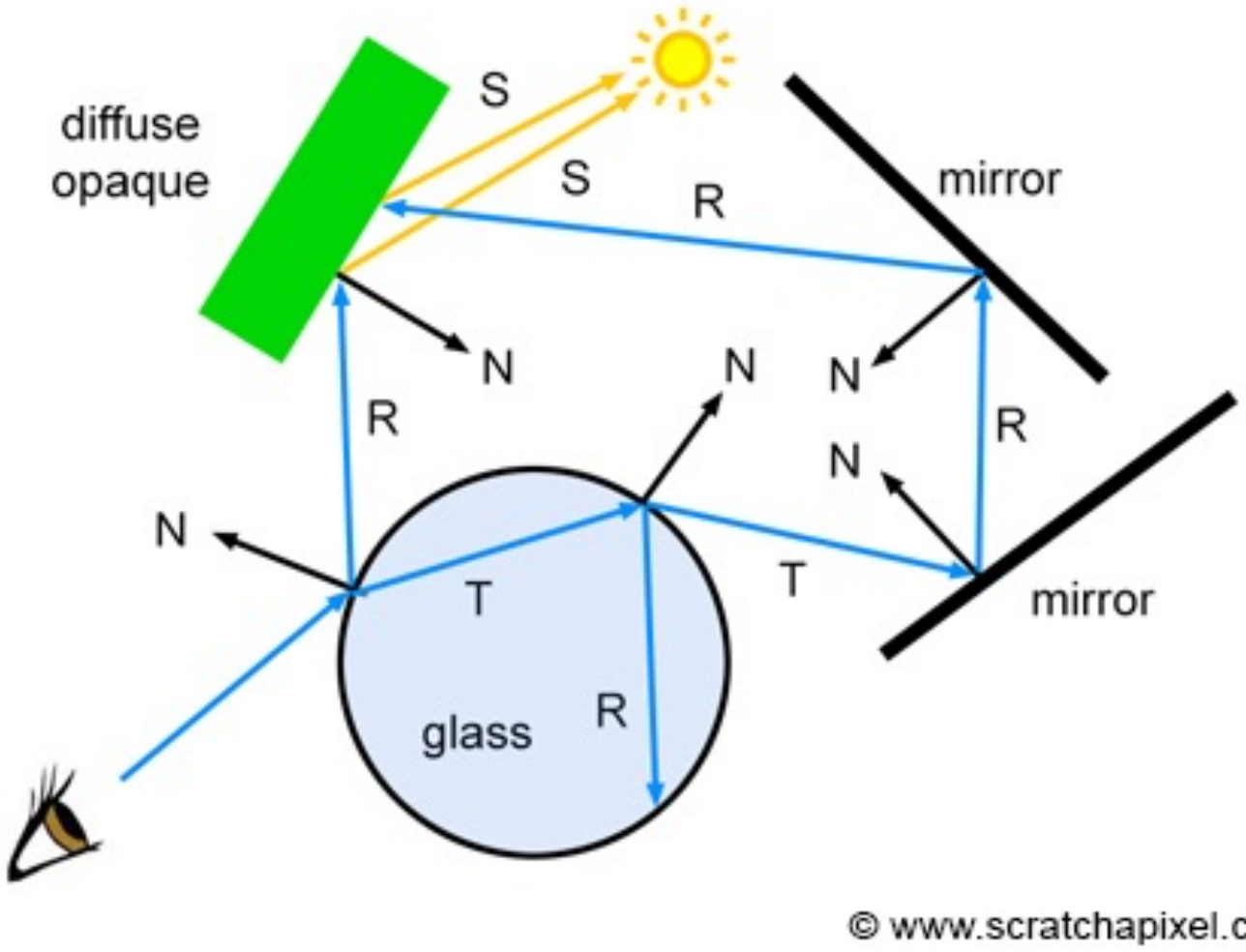
Light and Surfaces



Light and Surfaces

```
for each pixel in the image {
        pixel colour = rayTrace(viewRay, 0)
}
  colour rayTrace(Ray, depth) {
        for each object in the scene {
                if(Intersect ray with object) {
                        colour  = shading model
                        if(depth < maxDepth)
                                colour +=  rayTrace(reflectedRay,depth+1)
                }
        }
        return colour
  }
```

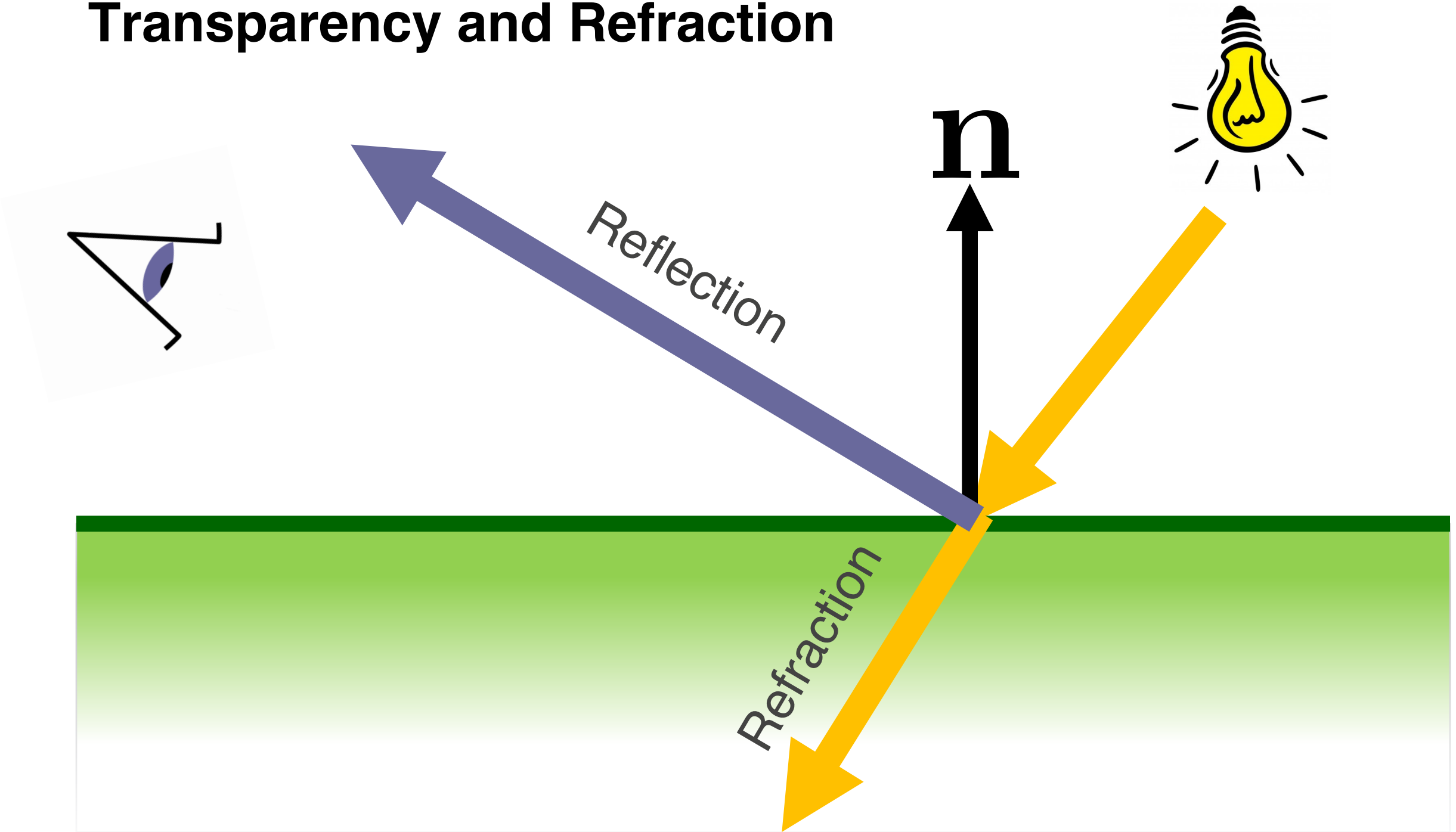
Ray Spawning



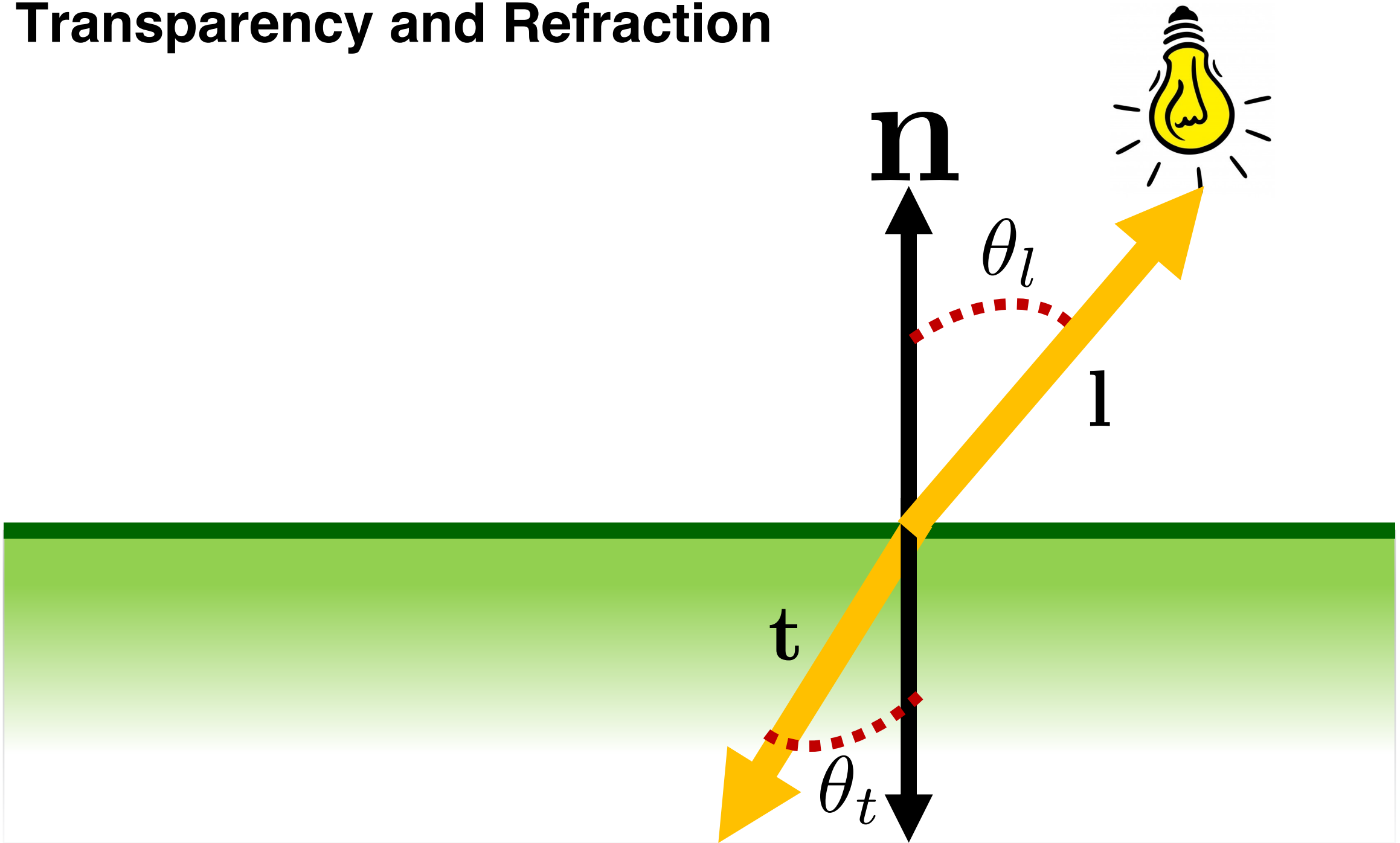
https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-overview/light-transport-ray-tracing-whitted

Transparency and Refraction

Transparency and Refraction



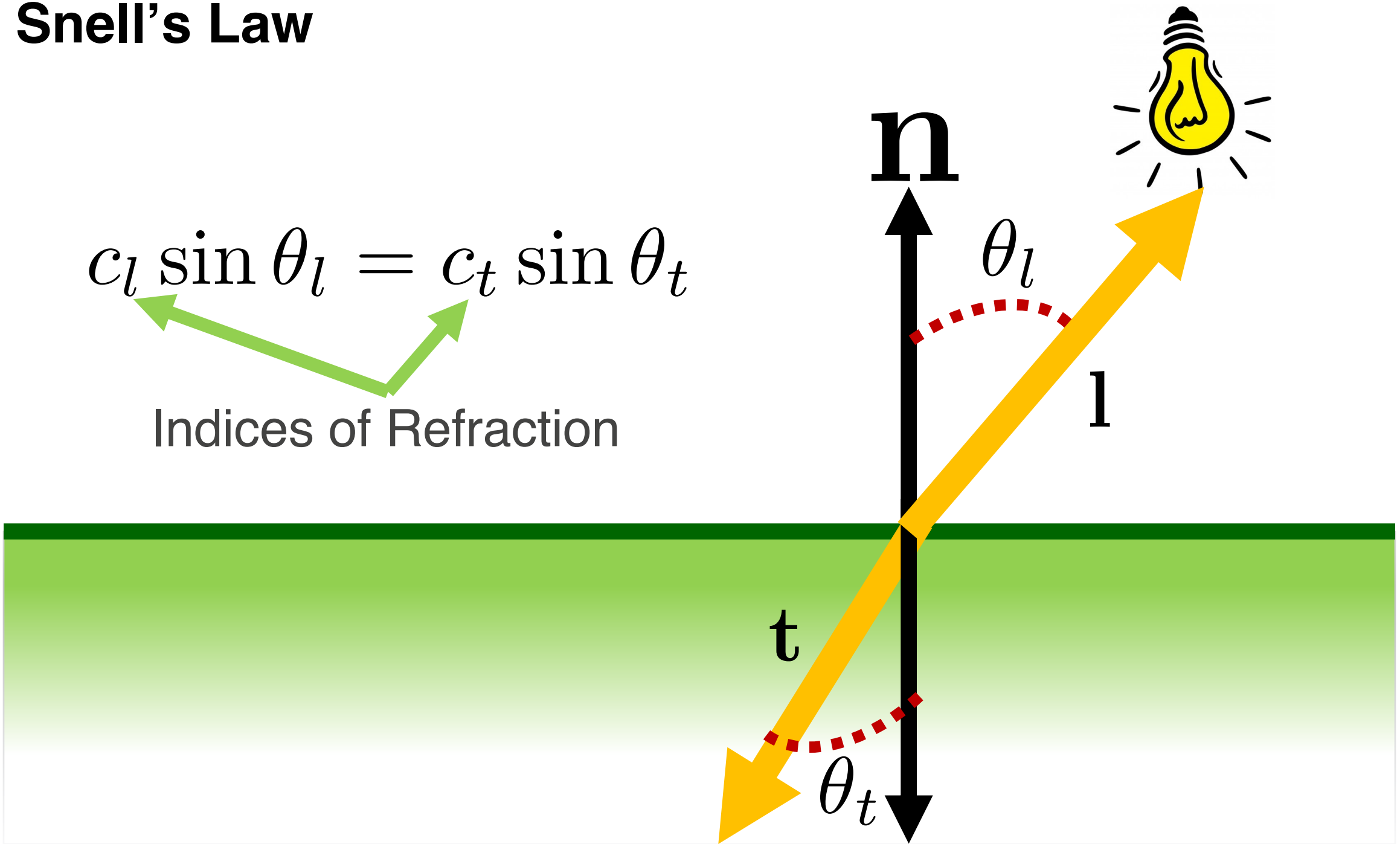
Transparency and Refraction



Snell's Law

$$c_l \sin\theta_l = c_t \sin\theta_t$$

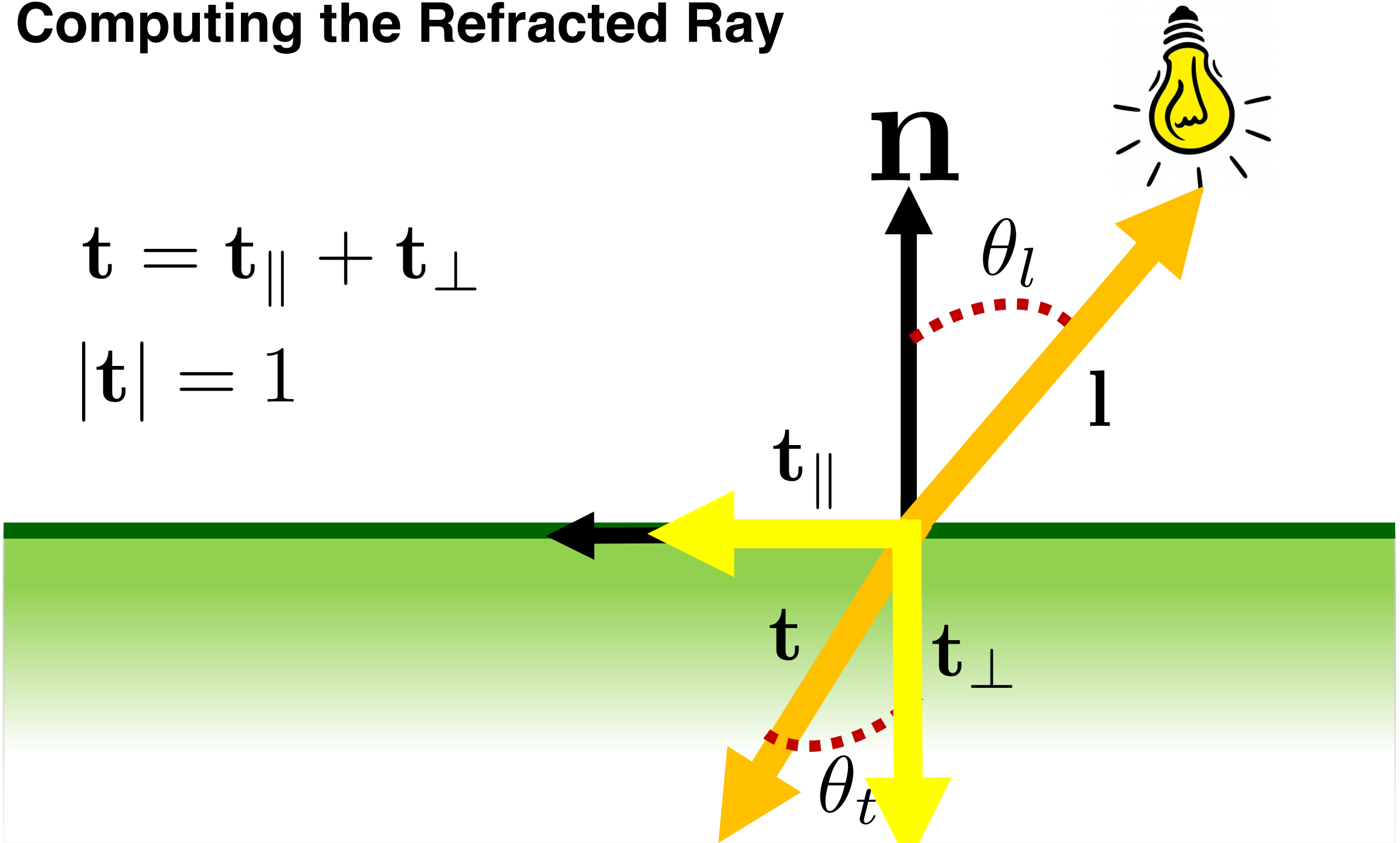
Indices of Refraction



Computing the Refracted Ray

$$\mathbf{t} = \mathbf{t}_{\parallel} + \mathbf{t}_{\perp}$$

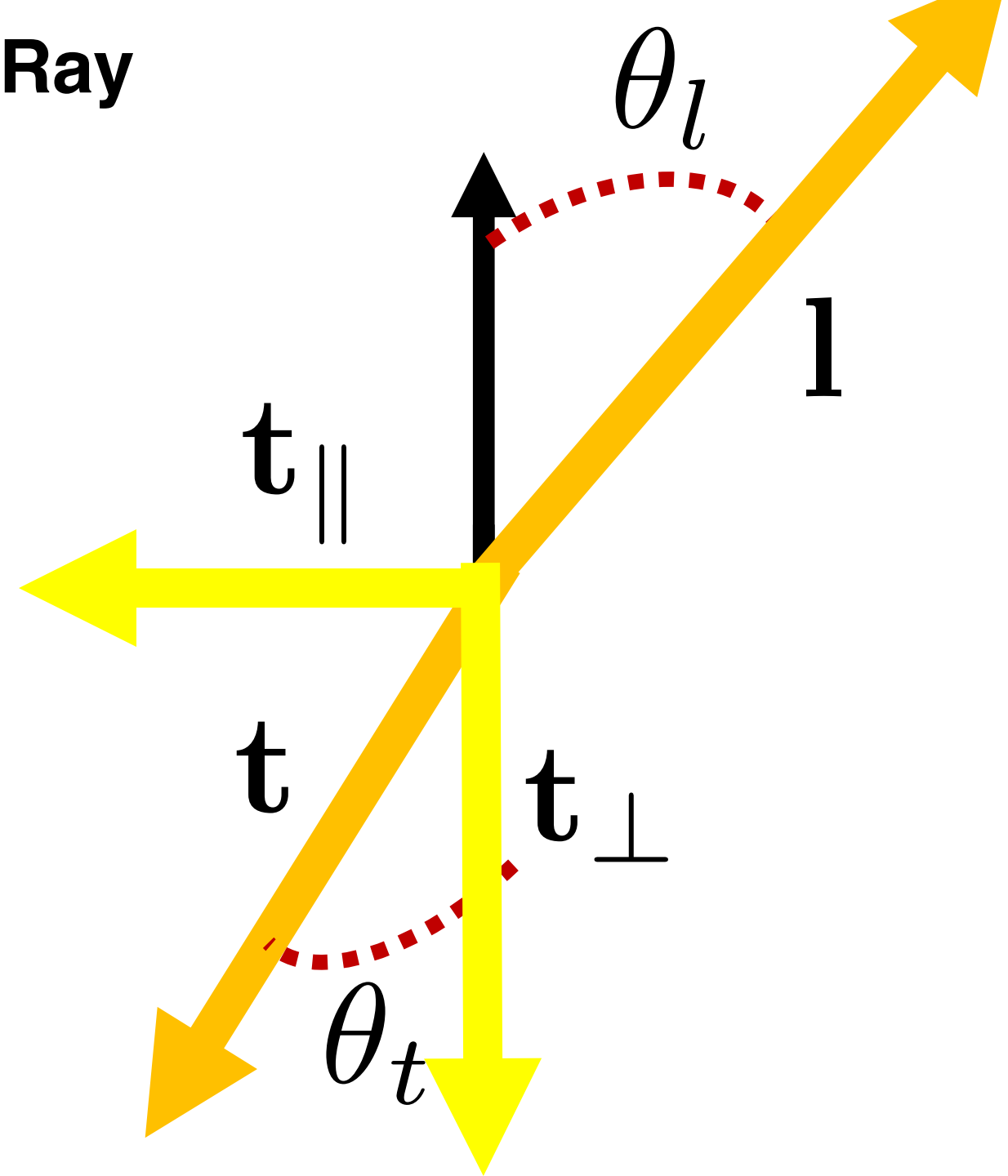
$$|\mathbf{t}| = 1$$



Computing the Refracted Ray

$$|\mathbf{t}_{\parallel}| = \sin\theta_t$$

$$|\mathbf{t}_{\parallel}| = \frac{c_l}{c_t}\sin\theta_l$$



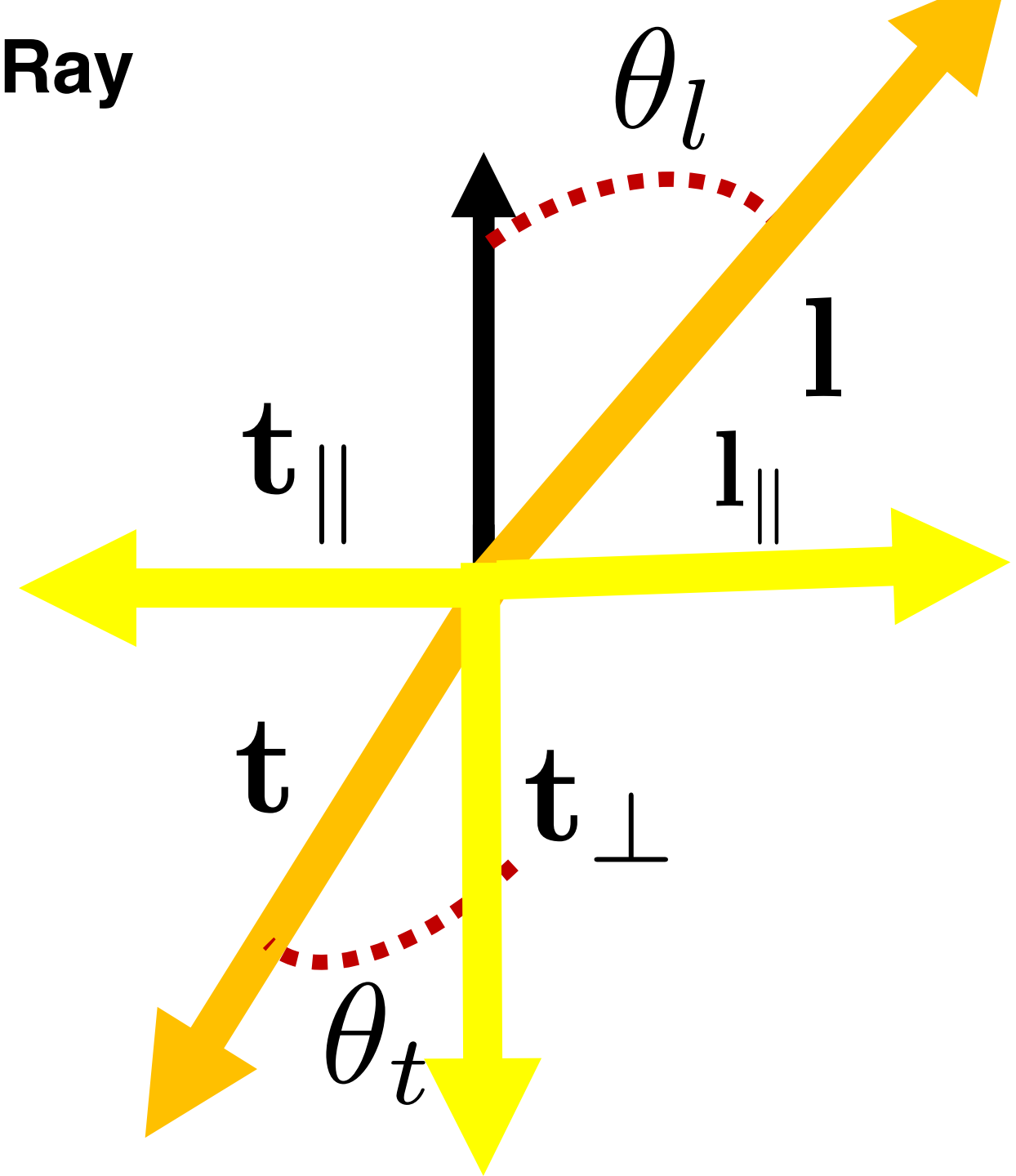
Computing the Refracted Ray

$$|\mathbf{t}_{\parallel}| = \sin\theta_t$$

$$|\mathbf{t}_{\parallel}| = \frac{c_l}{c_t}\sin\theta_l$$

$$|\mathbf{t}_{\parallel}| = \frac{c_l}{c_t}|\mathbf{l}_{\parallel}|$$

$$\mathbf{t}_{\parallel} = -\frac{c_l}{c_t}\mathbf{l}_{\parallel}$$

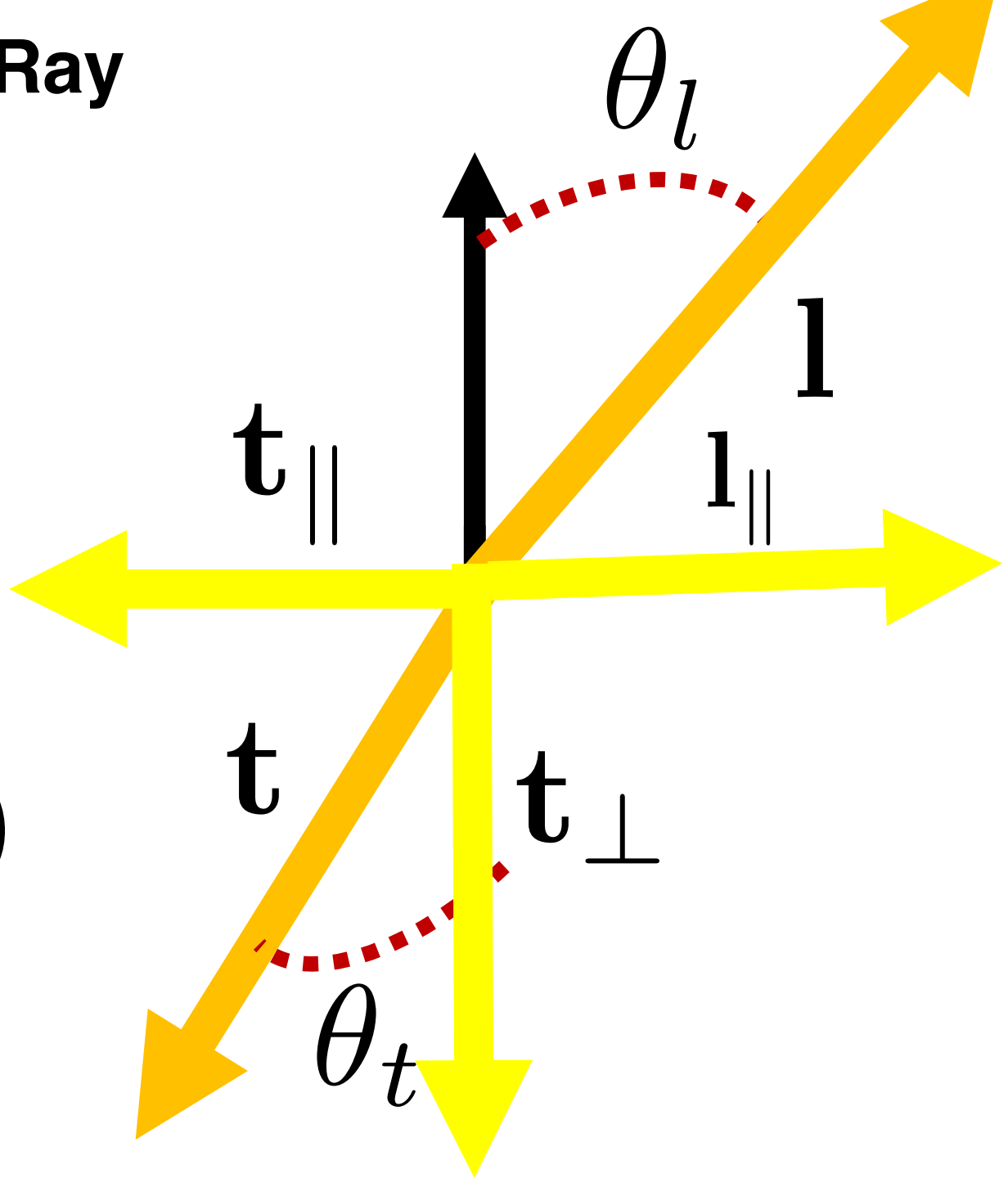


Computing the Refracted Ray

$$\mathbf{t}_{\parallel} = -\frac{c_l}{c_t}\mathbf{l}_{\parallel}$$

$$\mathbf{t}_{\parallel} = -\frac{c_l}{c_t}\left(\mathbf{l} - \mathbf{n}\mathbf{n}^T\mathbf{l}\right)$$

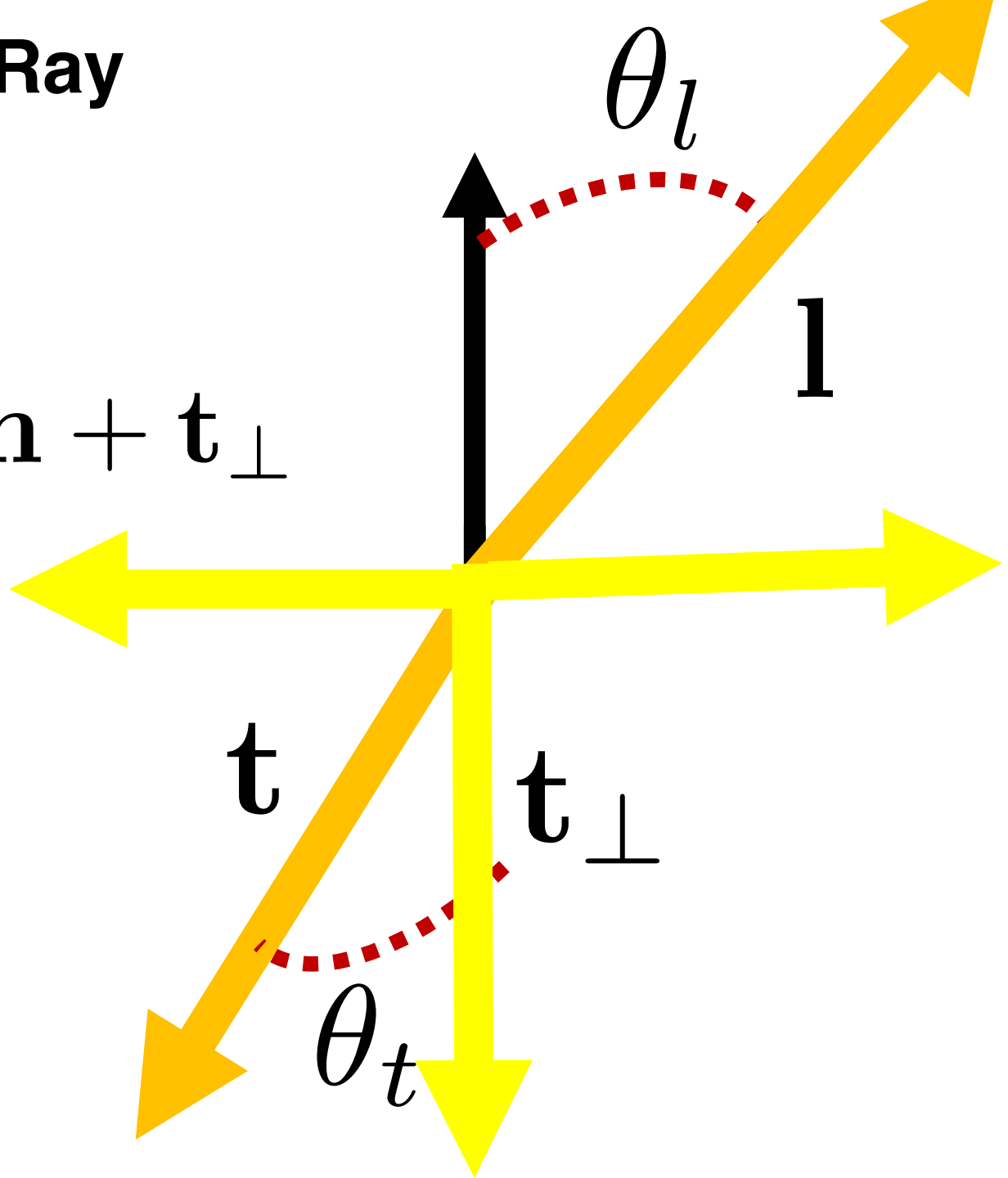
$$\mathbf{t}_{\parallel} = -\frac{c_l}{c_t}\left(\mathbf{l} - \cos\theta_l\mathbf{n}\right)$$



Computing the Refracted Ray

$$\mathbf{t} = \mathbf{t}_{\parallel} + \mathbf{t}_{\perp}$$

$$\mathbf{t} = -\frac{c_l}{c_t}\mathbf{l} + \frac{c_l}{c_t}\cos\theta_l \mathbf{n} + \mathbf{t}_{\perp}$$



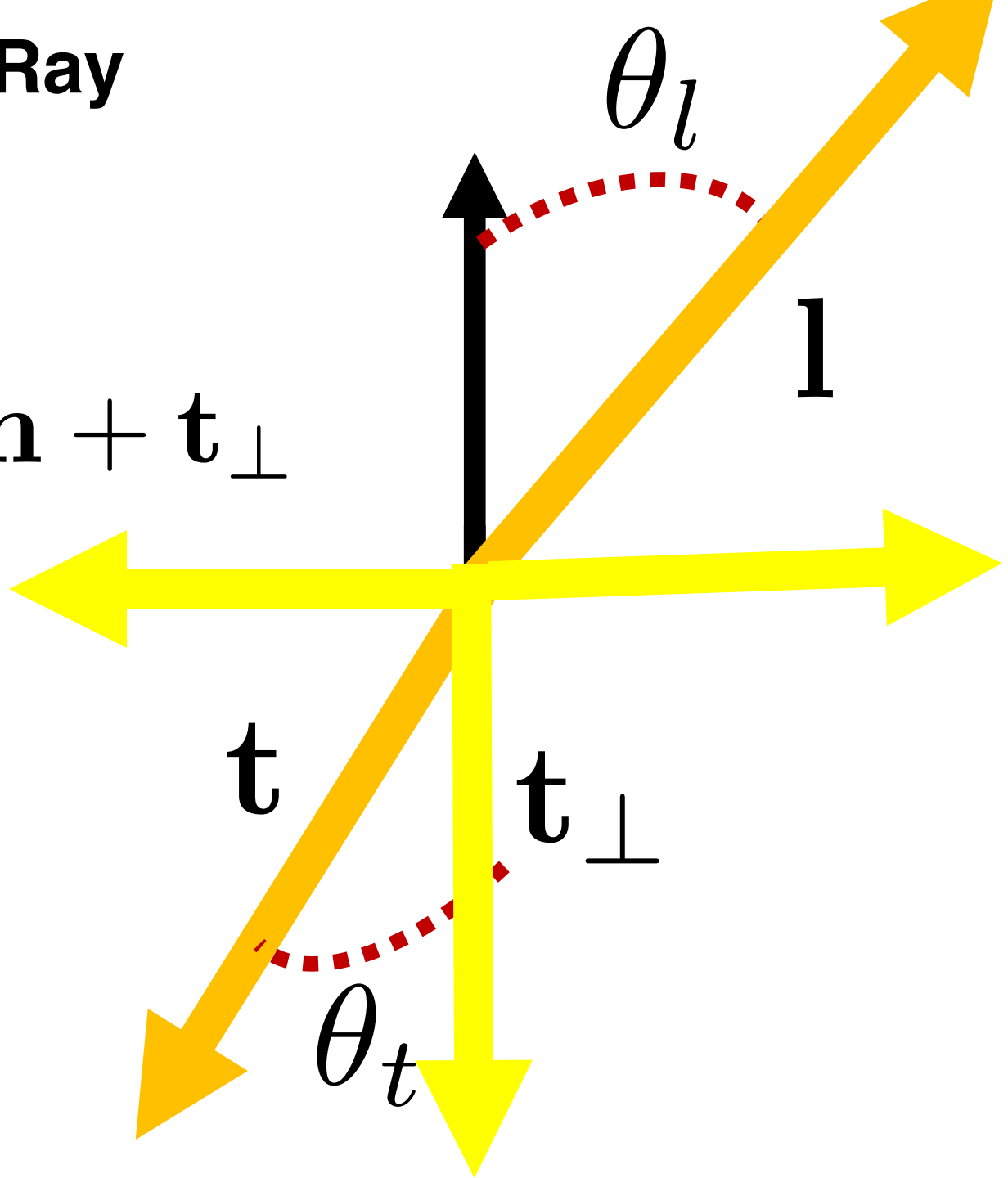
Computing the Refracted Ray

$$\mathbf{t} = \mathbf{t}_{\parallel} + \mathbf{t}_{\perp}$$

$$\mathbf{t} = -\frac{c_l}{c_t}\mathbf{l} + \frac{c_l}{c_t}\cos\theta_l \mathbf{n} + \mathbf{t}_{\perp}$$

$$\mathbf{t}_{\perp} = -\alpha \mathbf{n}$$

$$\mathbf{t}_{\perp} = -\sqrt{1 - \sin^2\theta_t}\,\mathbf{n}$$



Computing the Refracted Ray

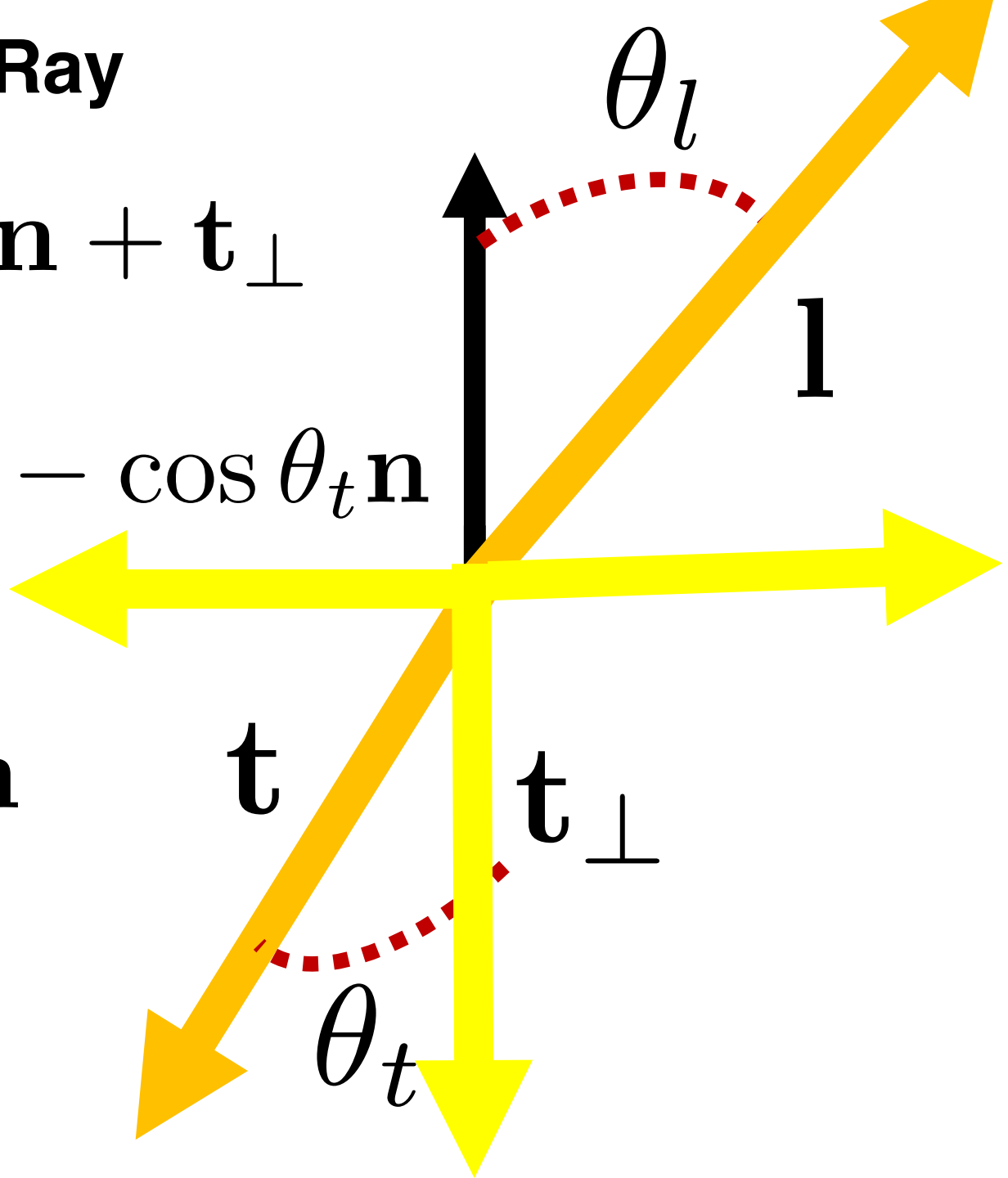
$$\mathbf{t} = -\frac{c_l}{c_t}\mathbf{l} + \frac{c_l}{c_t}\cos\theta_l\mathbf{n} + \mathbf{t}_\perp$$

$$\mathbf{t} = -\frac{c_l}{c_t}\mathbf{l} + \frac{c_l}{c_t}\cos\theta_l\mathbf{n} - \cos\theta_t\mathbf{n}$$

$$\mathbf{t}_\perp = -\sqrt{1-\sin^2\theta_t}\mathbf{n}$$

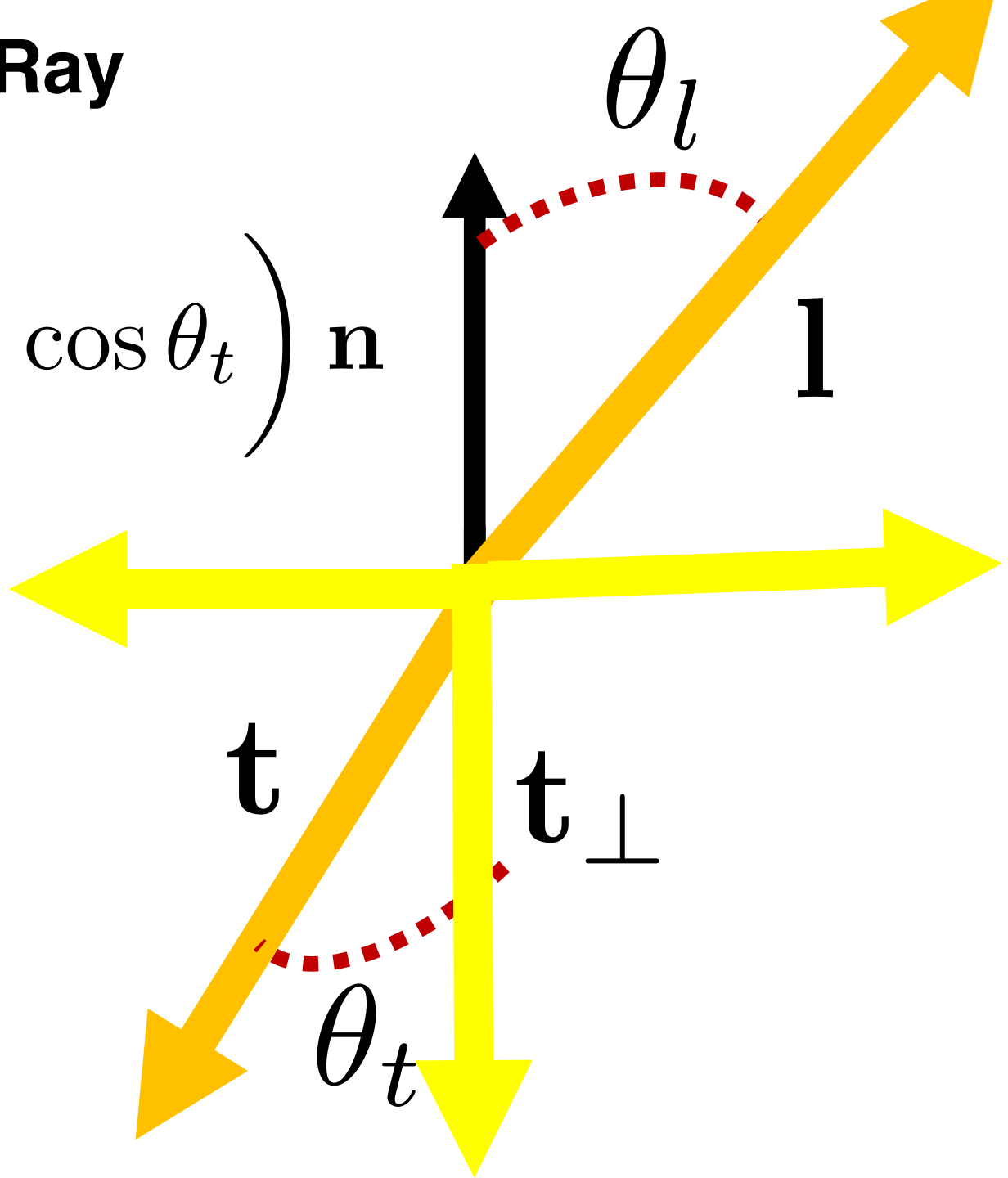
$$\mathbf{t}_\perp = -\sqrt{\cos^2\theta_t}\mathbf{n}$$

$$\mathbf{t}_\perp = -\cos\theta_t\mathbf{n}$$



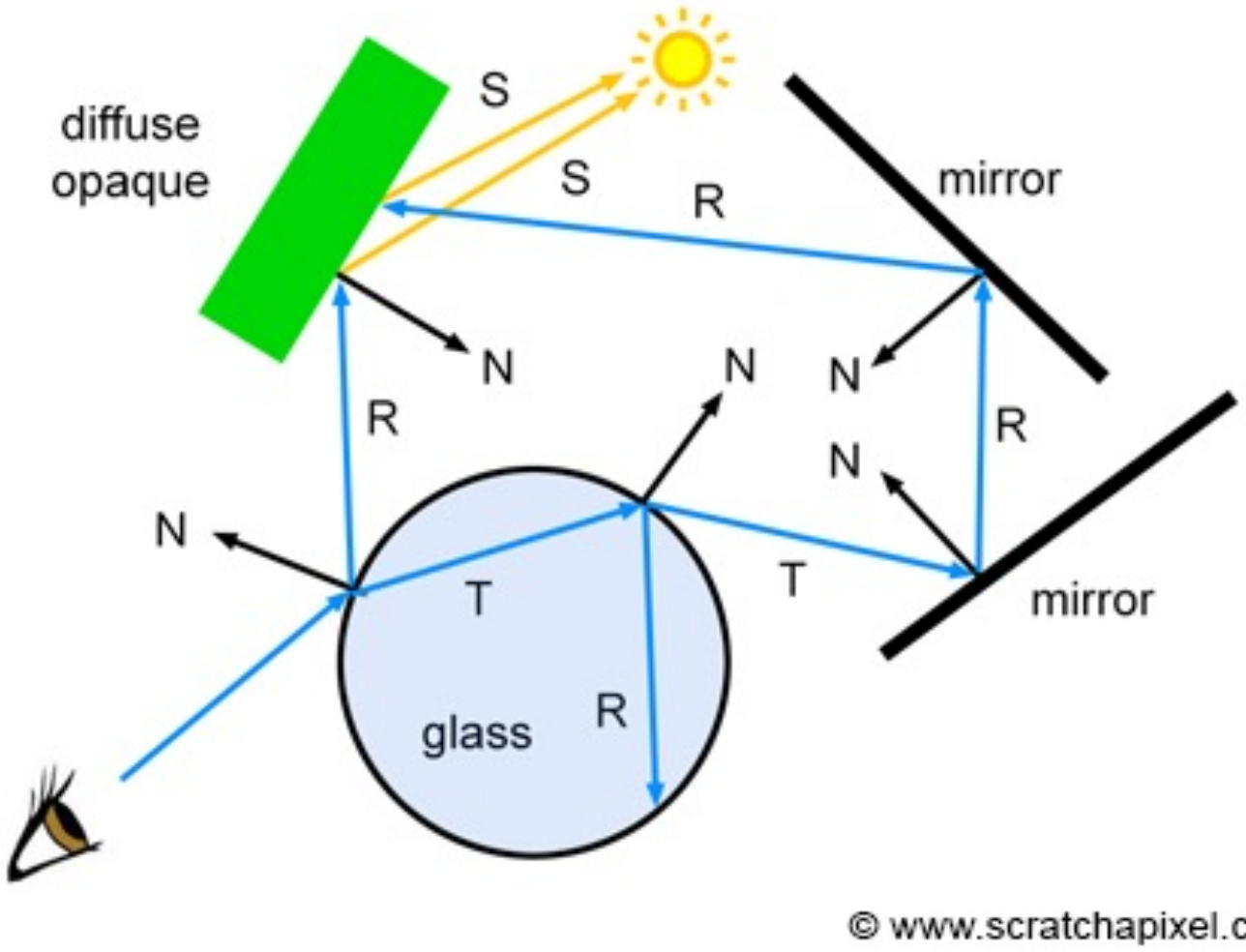
Computing the Refracted Ray

$$\mathbf{t} = -\frac{c_l}{c_t}\mathbf{l} + \left(\frac{c_l}{c_t}\cos\theta_1 - \cos\theta_t\right)\mathbf{n}$$



```
colour rayTrace(Ray, depth) {
    for each object in the scene {
        if(Intersect ray with object) {
            colour  = shading model
            if(depth < maxDepth)  {
                colour +=  rayTrace(reflectedRay,depth+1)
                colour +=  rayTrace(refractedRay,depth+1)
            }
        }
    }
    return colour
}
```

Ray Spawning



https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-overview/light-transport-ray-tracing-whitted

Done for Today

Assignment 2 due this Wednesday

Assignment 3 available today