

csc418/2504 Computer Graphics

Rob Katz

Some Slides/Images adapted from Marschner and Shirley

Today: Ray Tracing



Announcements

MarkUs Access

If you don't have MarkUs access still email the TAs

Subject: CSC418 Access to MarkUs

Assignment 2 is due this Friday

Assignment 3 is available now (due 01/02)

TA Office Hours: Thursday 2:30pm – 3:30pm, BA5287

TA Email Address: *csc418tas@cs.toronto.edu*

Where to Ask Questions

1. GitHub Issues Pages for Assignments

1. “I think there’s a bug in the starter code”
2. “The CMakeFile is missing”
3. “The code didn’t clone correctly”

2. Discussion board

1. “The lecture wasn’t clear”
2. “This assignment task is ambiguous”

3. TA Email List

1. ”I need an extension”
2. “I’d like to request a remark”

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.

Accessibility Services needs dependable volunteer note-takers to assist students living with a disability to achieve academic success!

Volunteers report that by giving to the U of T community their class attendance and note taking skills improve.

All you have to do is attend classes regularly & submit your notes consistently:

1) Register Online as a Volunteer Note-Taker

at: <https://clockwork.studentlife.utoronto.ca/custom/misc/home.aspx>

2) Follow the link that says Volunteer Notetakers

3) Select your course and upload a sample of your notes

4) Once you have been selected as a note-taker you'll get an email notifying you to upload your notes.

If you have any questions, please call us. Our office is open Monday-Friday 9:00am-5:00pm and Tuesdays 9:00 AM-6:00 PM.

Email us at as.notetaking@utoronto.ca or call 416-978-6186 if you have questions or require any assistance.

Volunteers may receive co-curricular credit or a certificate of appreciation.

Your support is much appreciated!

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

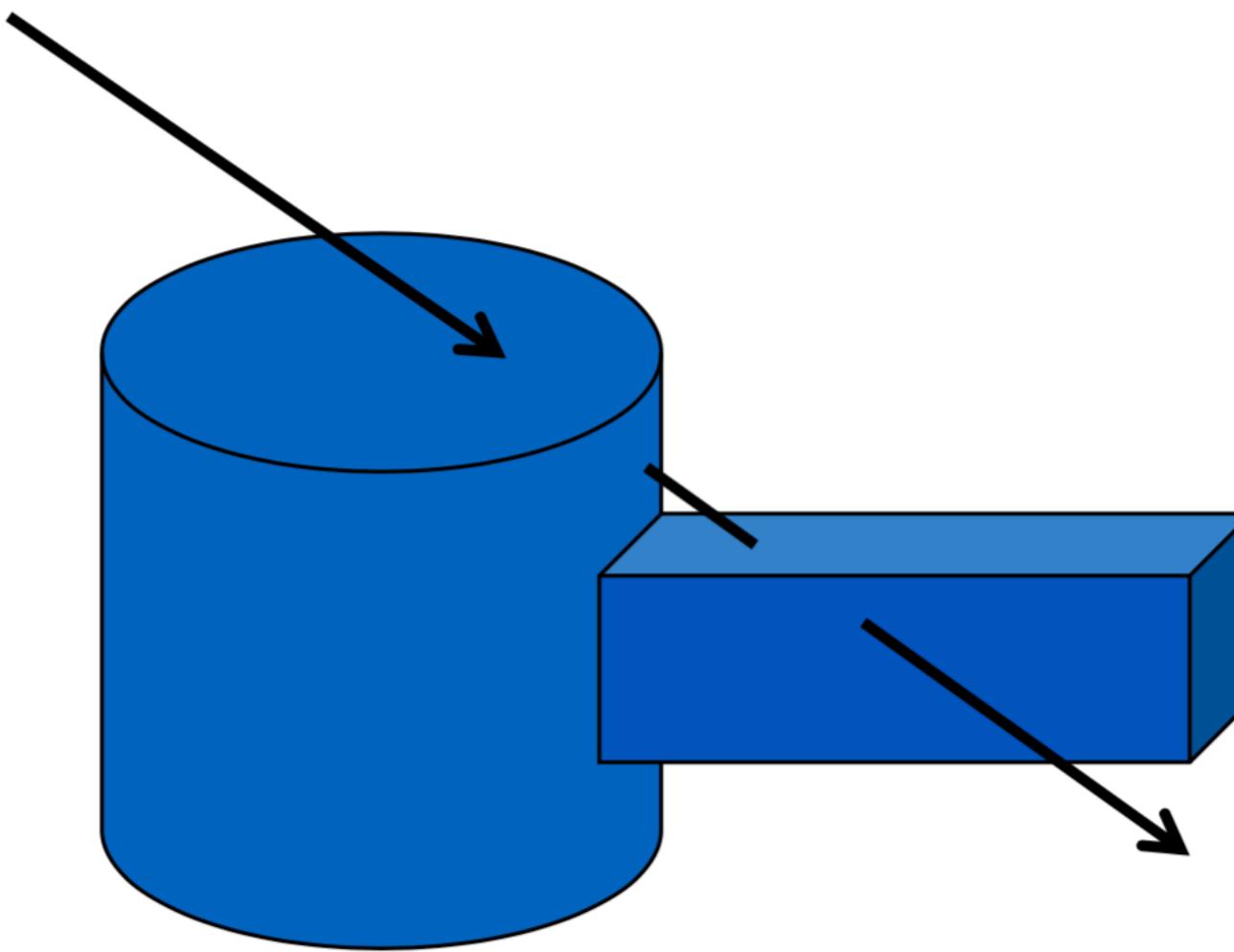
Ray Casting

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

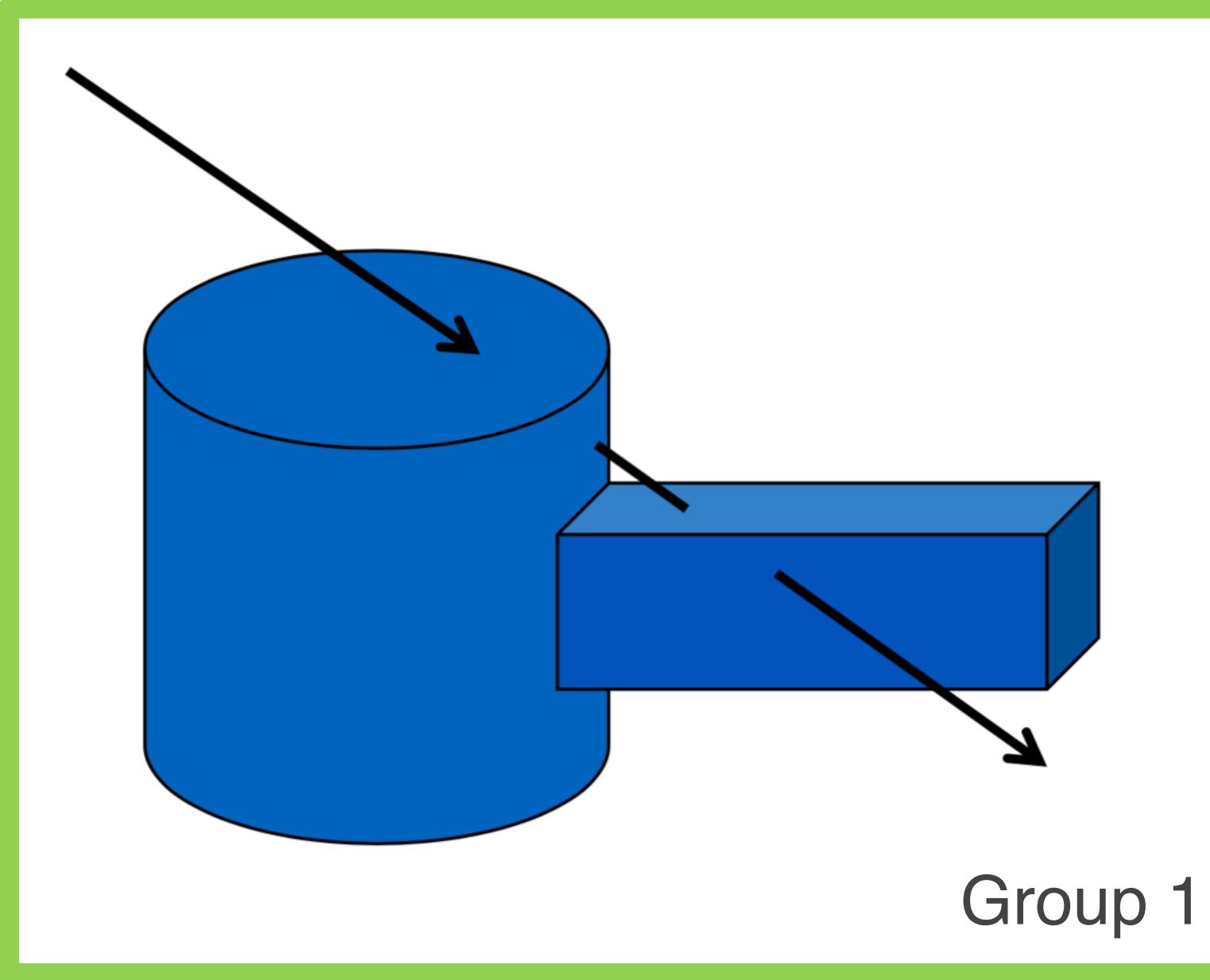
Ray Casting

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

Intersection Tests with Composite Objects



Intersection Tests with Composite Objects



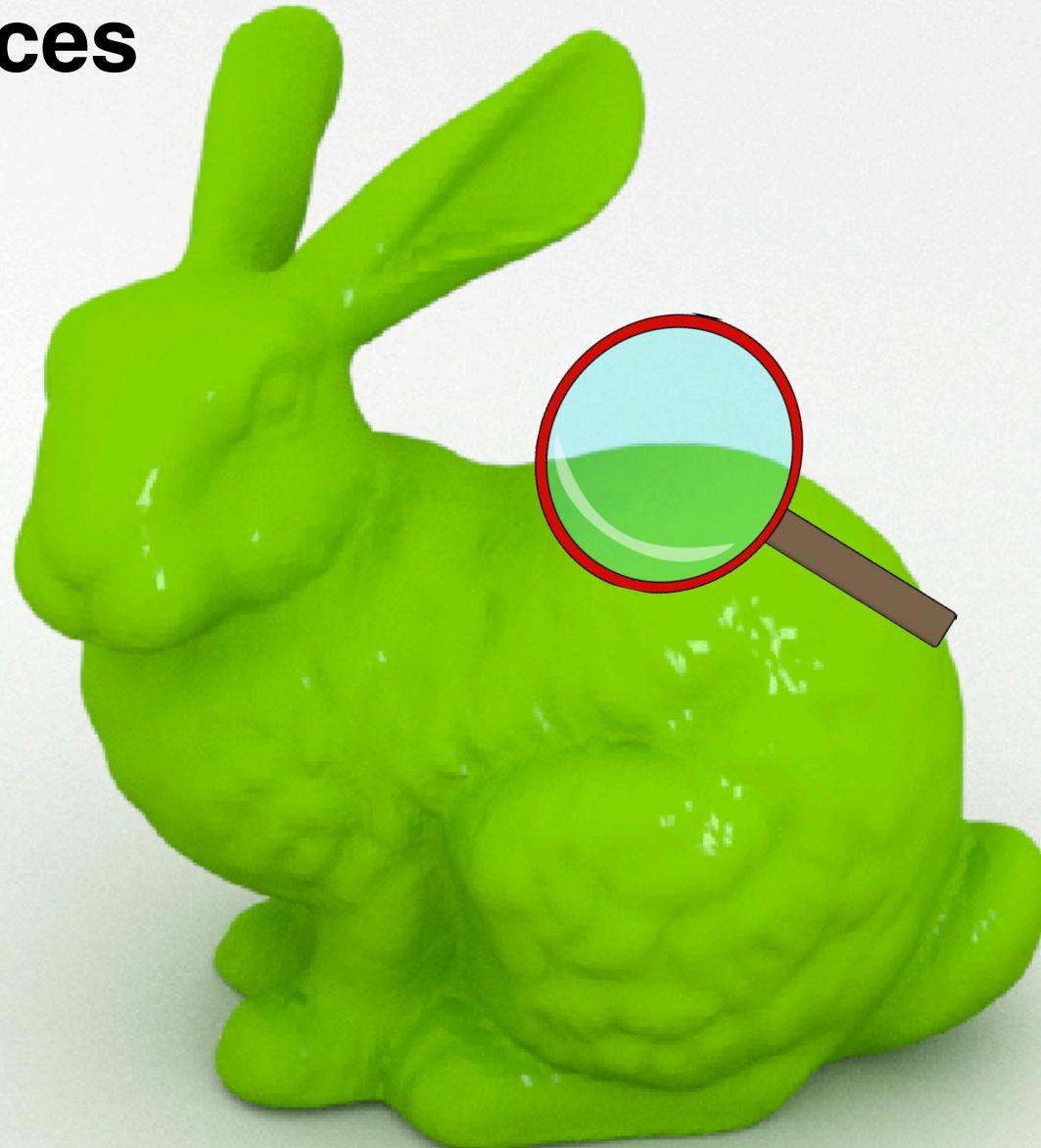
Ray Casting

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

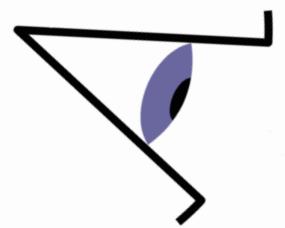
Light and Surfaces



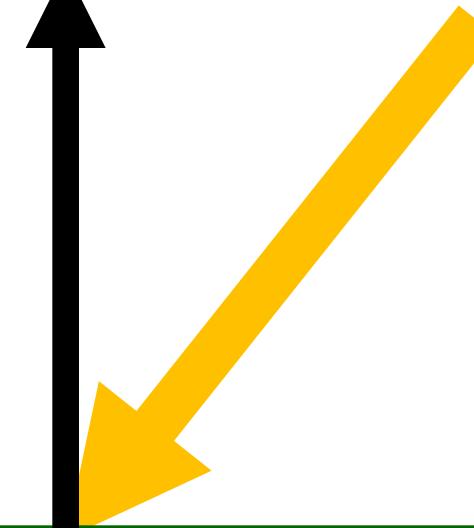
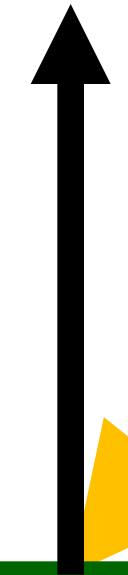
Light and Surfaces



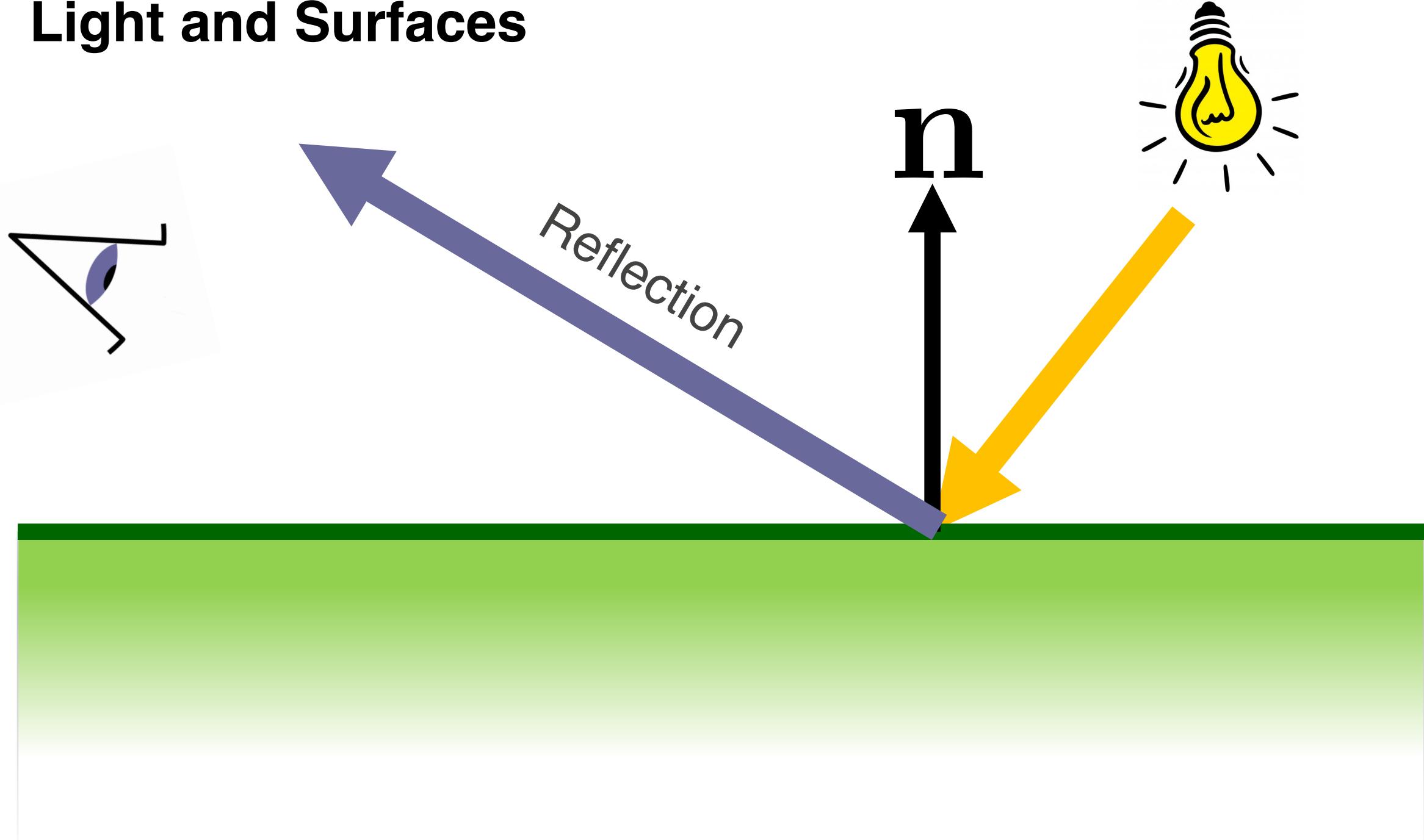
Light and Surfaces



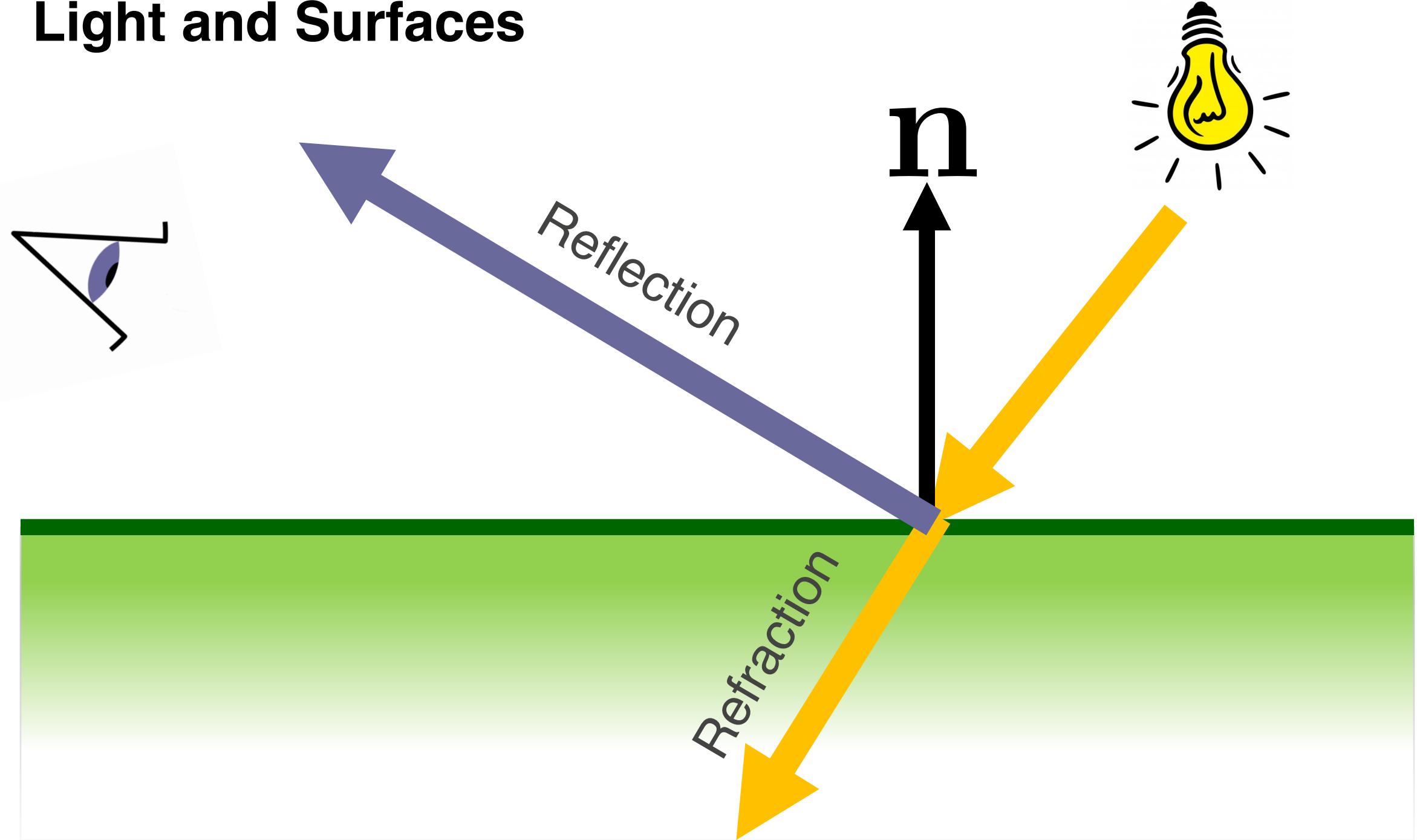
n



Light and Surfaces



Light and Surfaces



Lights

Two types of lights:

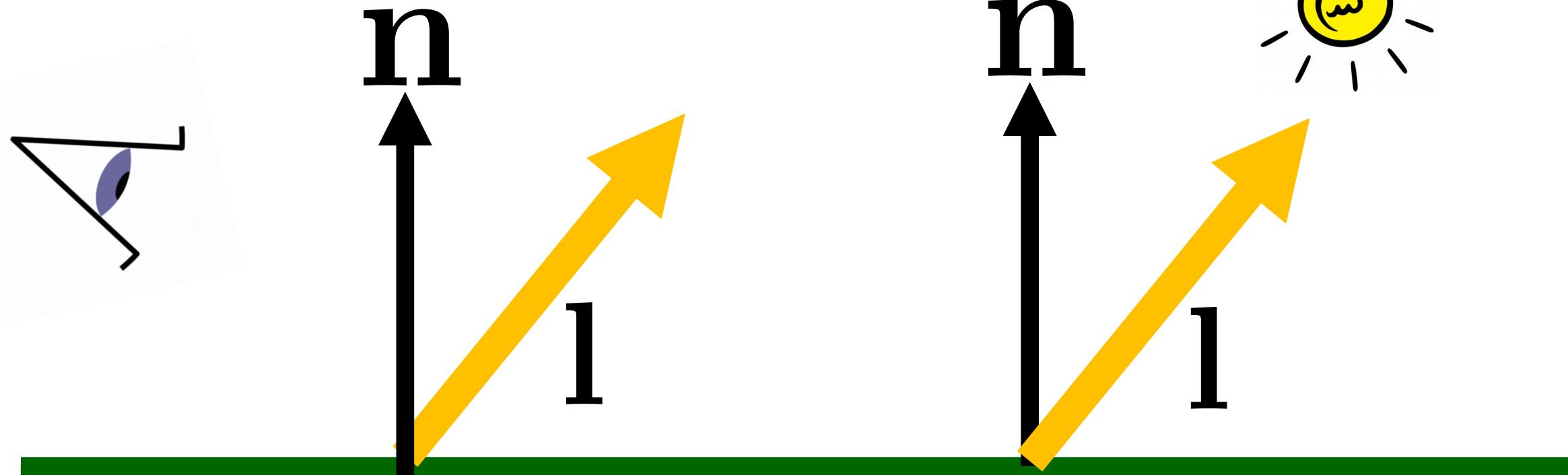
Directional Light:

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

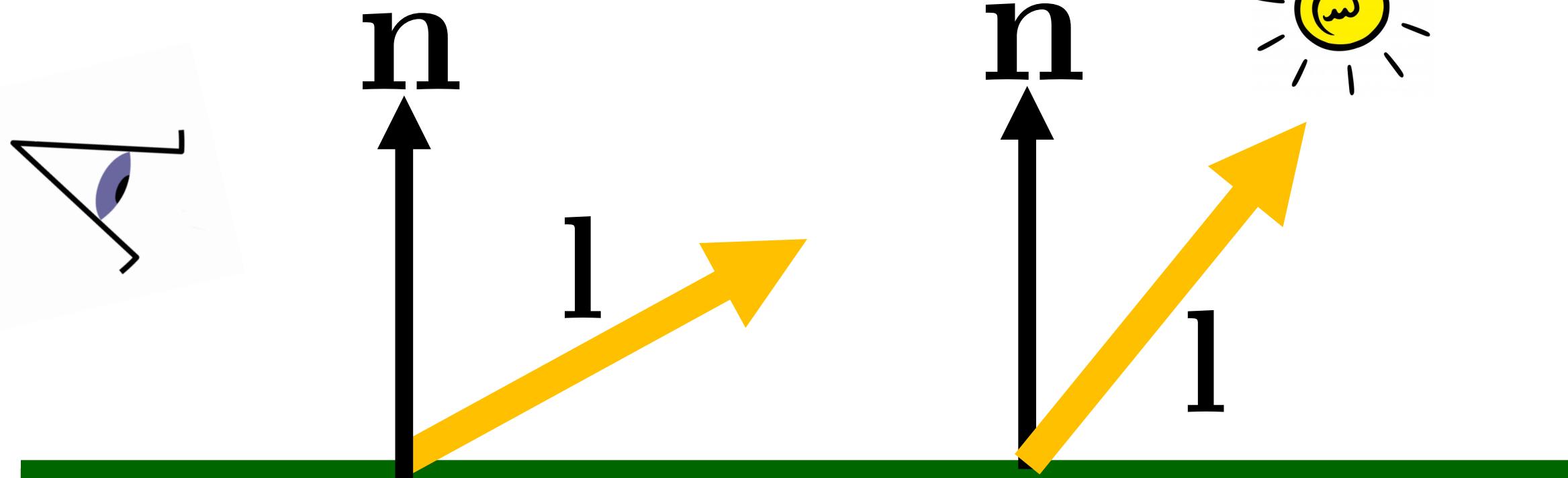
Point Light

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

Directional Light



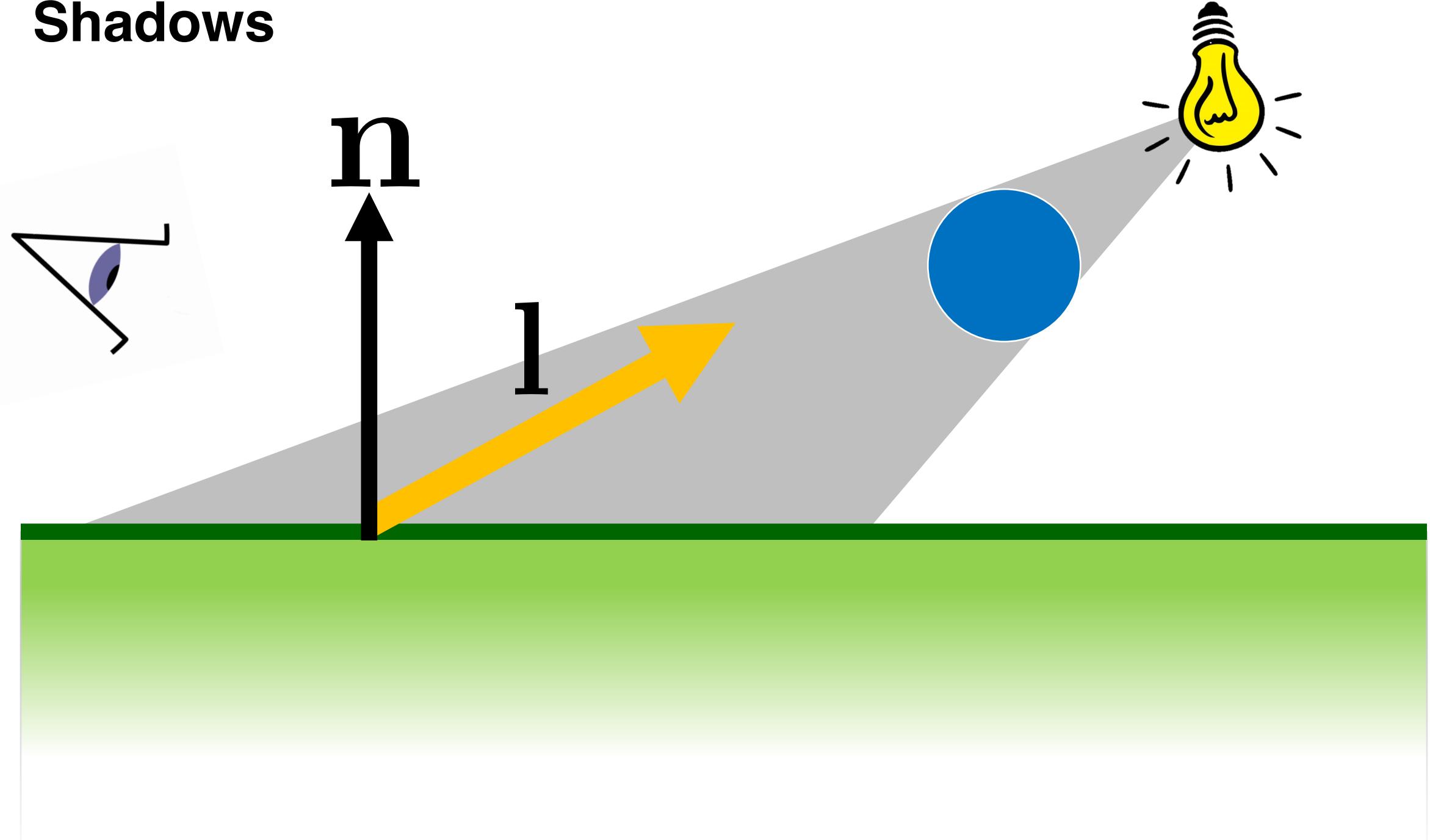
Point Light



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



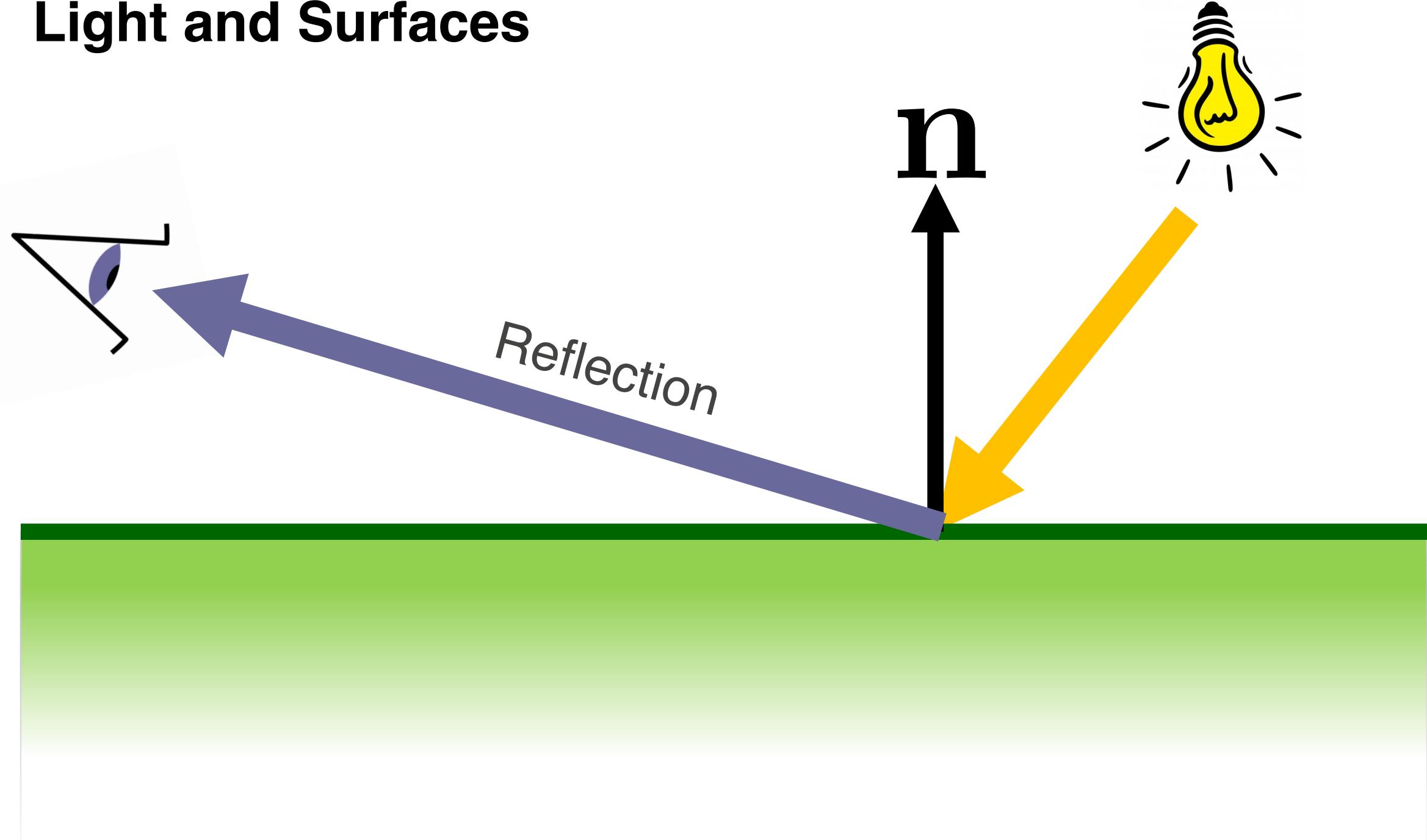
WARNING: Numerical Precision

What are valid values t for the shadow ray?

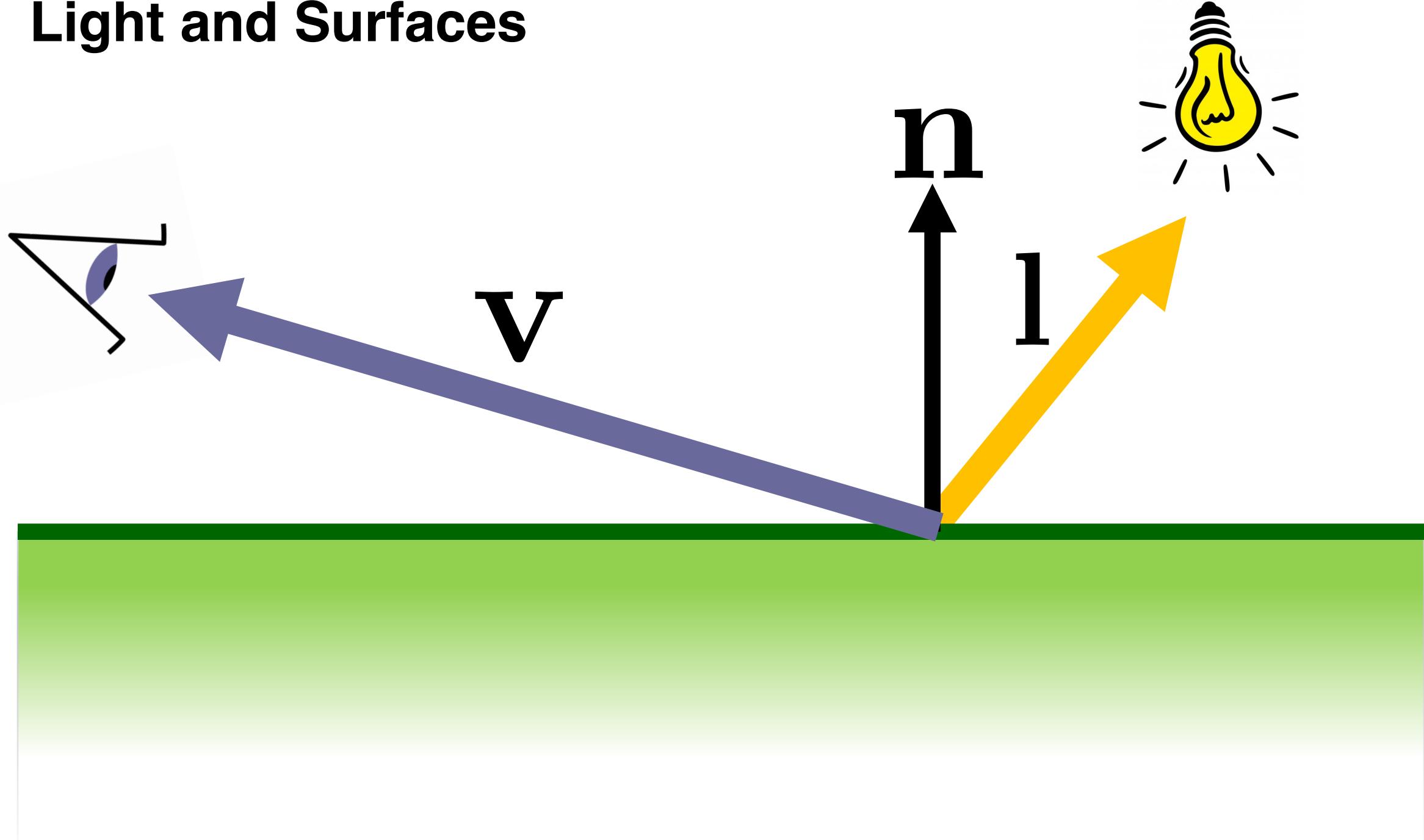
Ray Casting

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

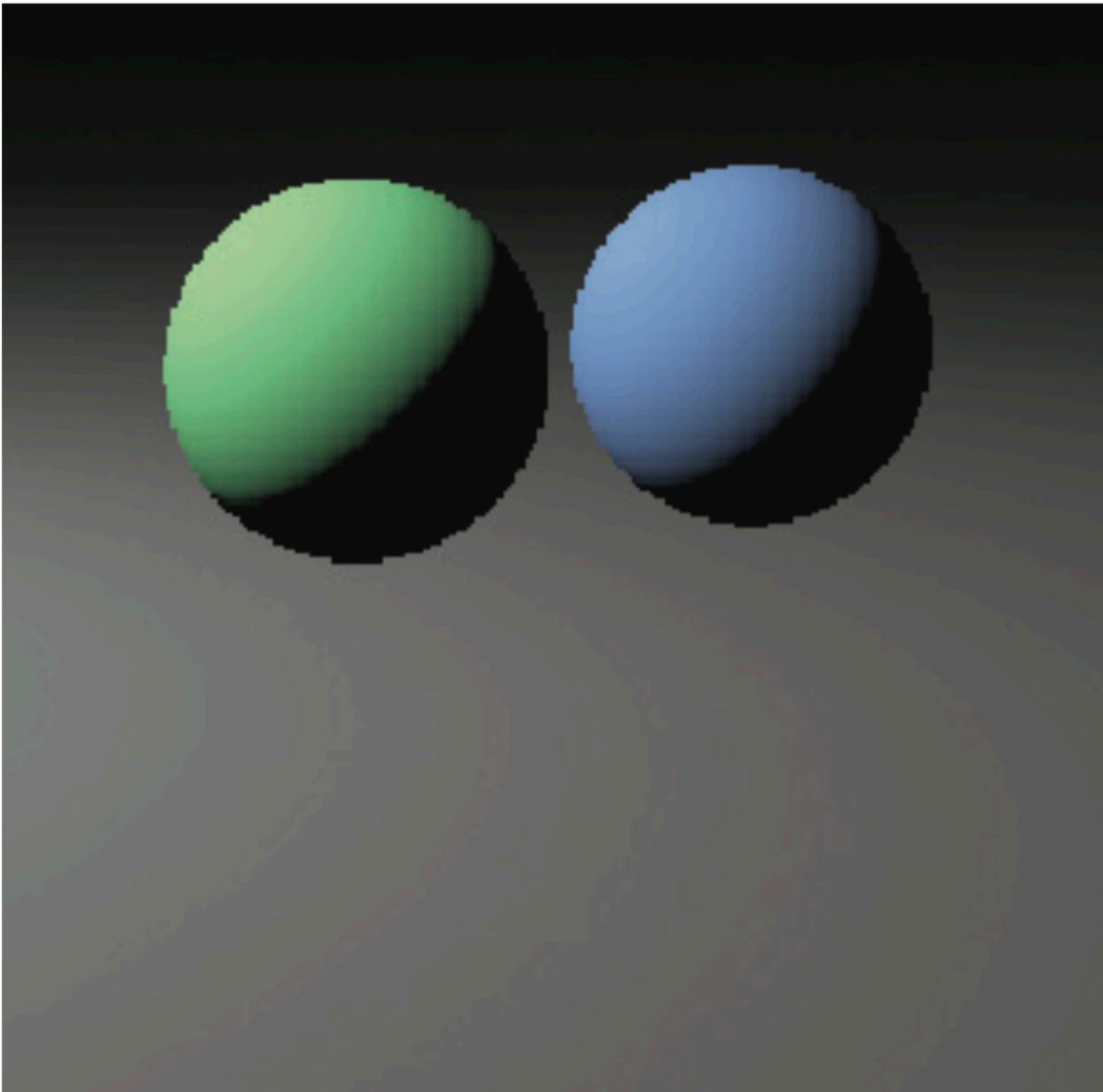
Light and Surfaces



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)

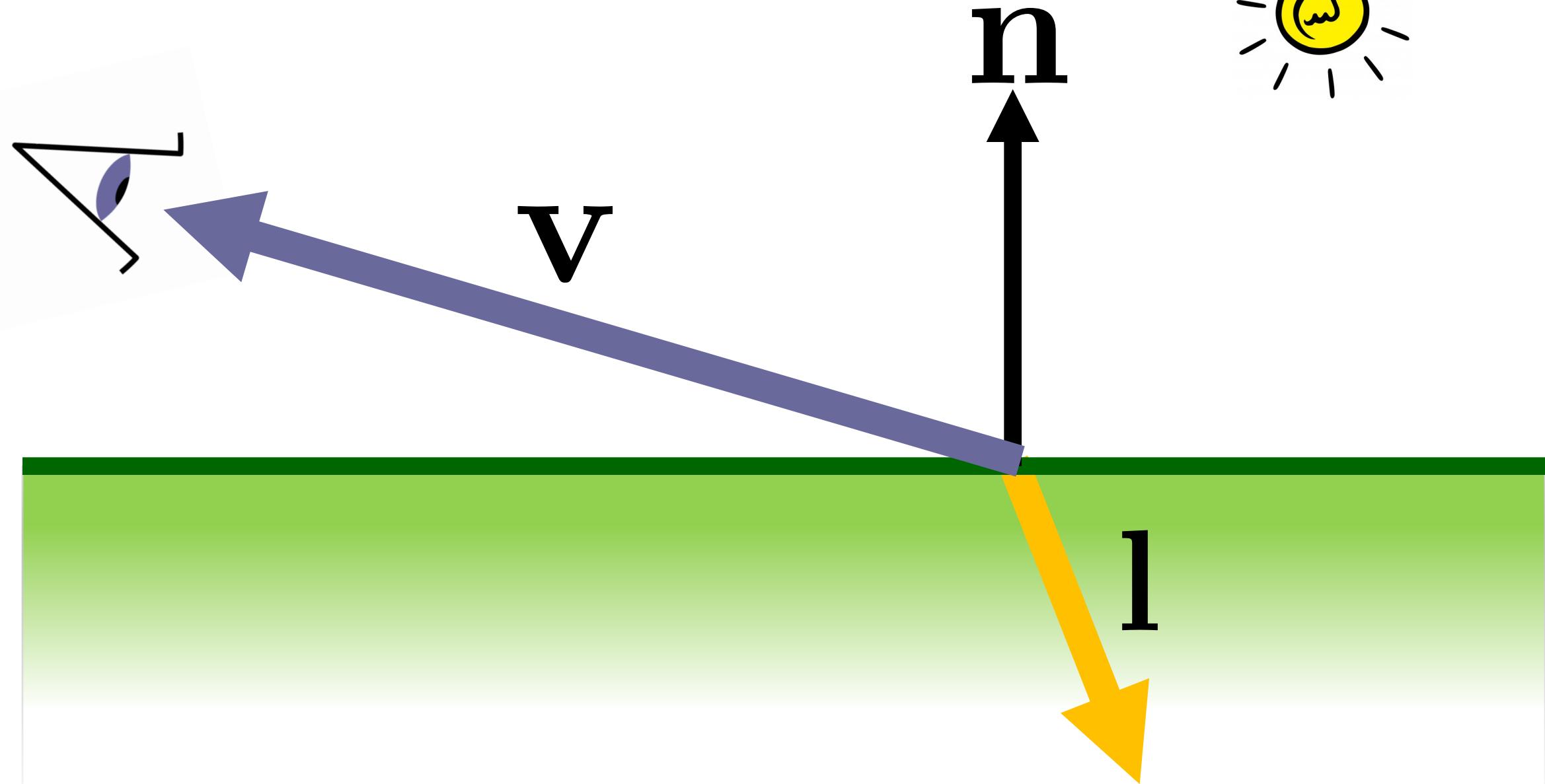
Lambertian (Diffuse) Shading

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.

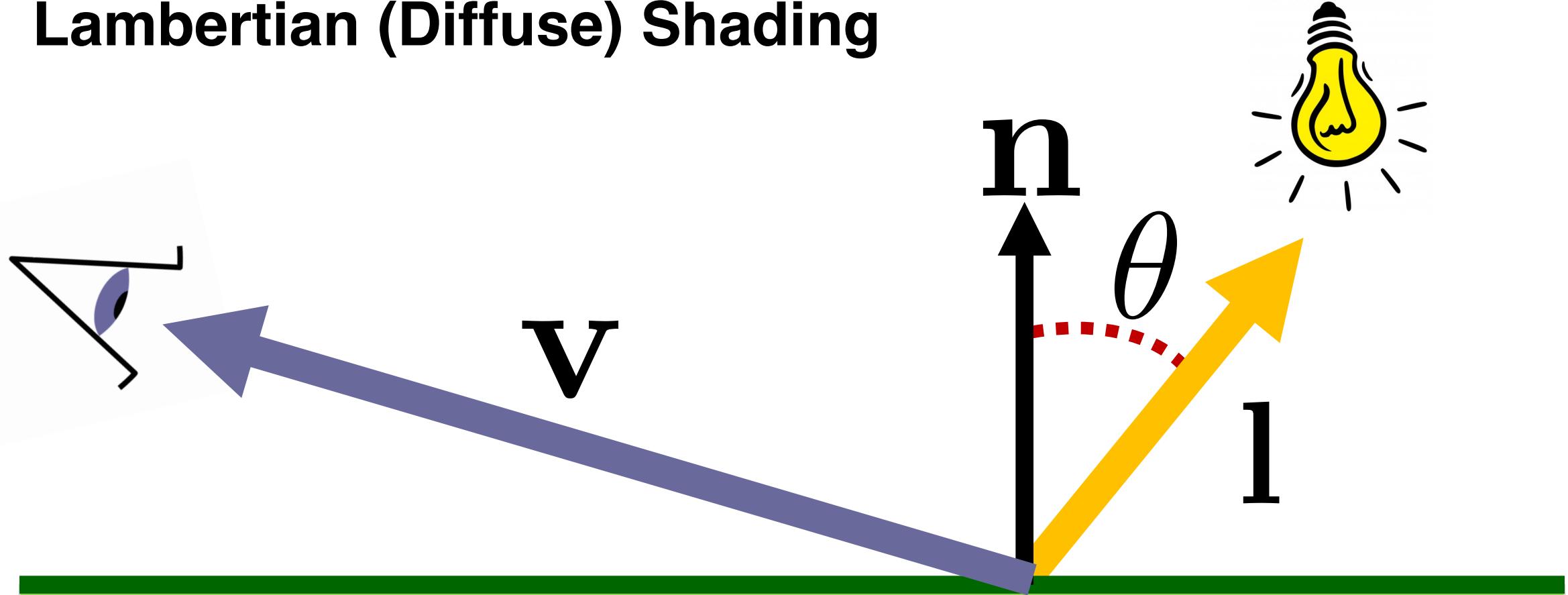
- Lambert (18th century)

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

Self Shadowing

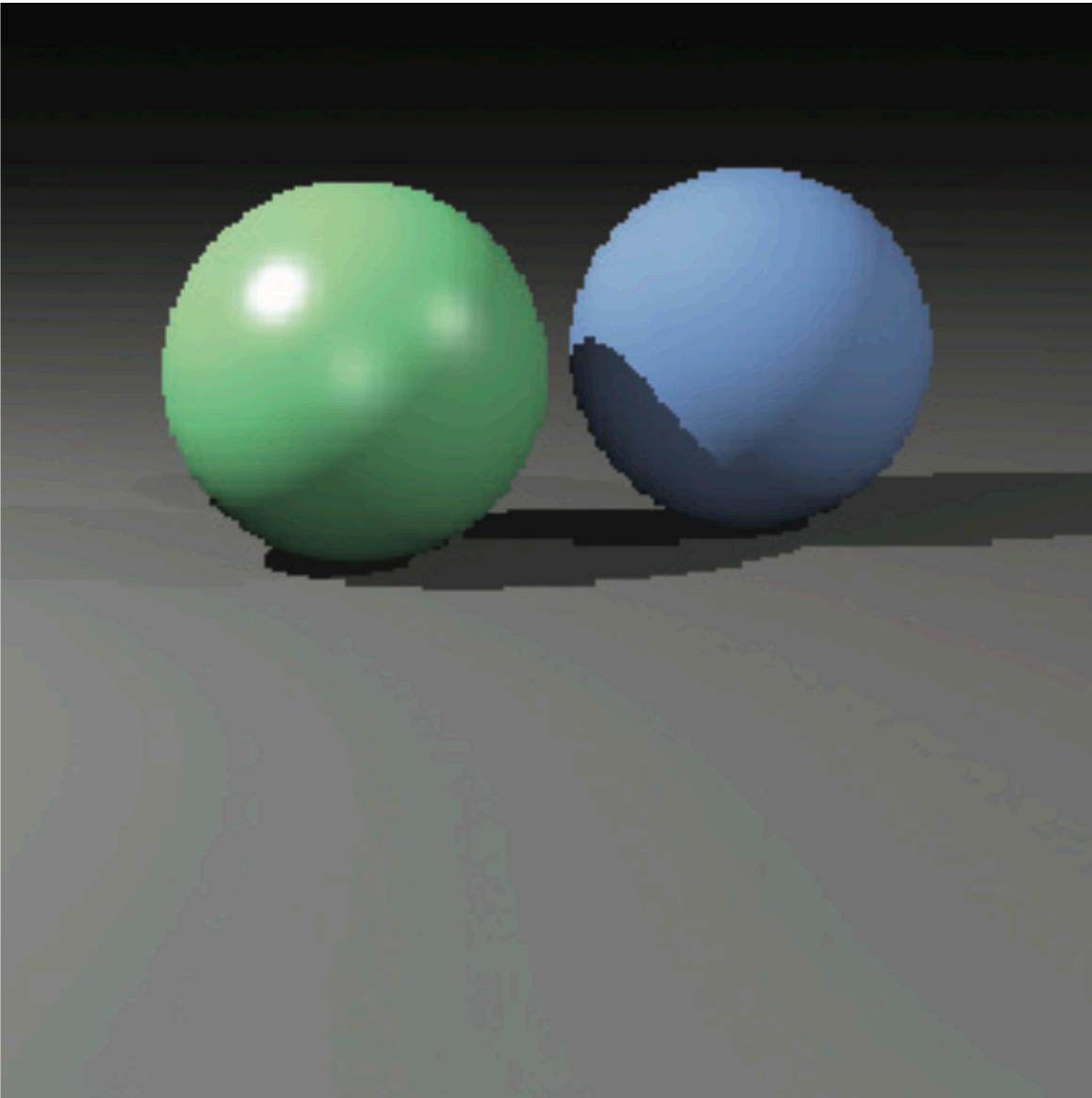


Lambertian (Diffuse) Shading



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

Specular Reflection

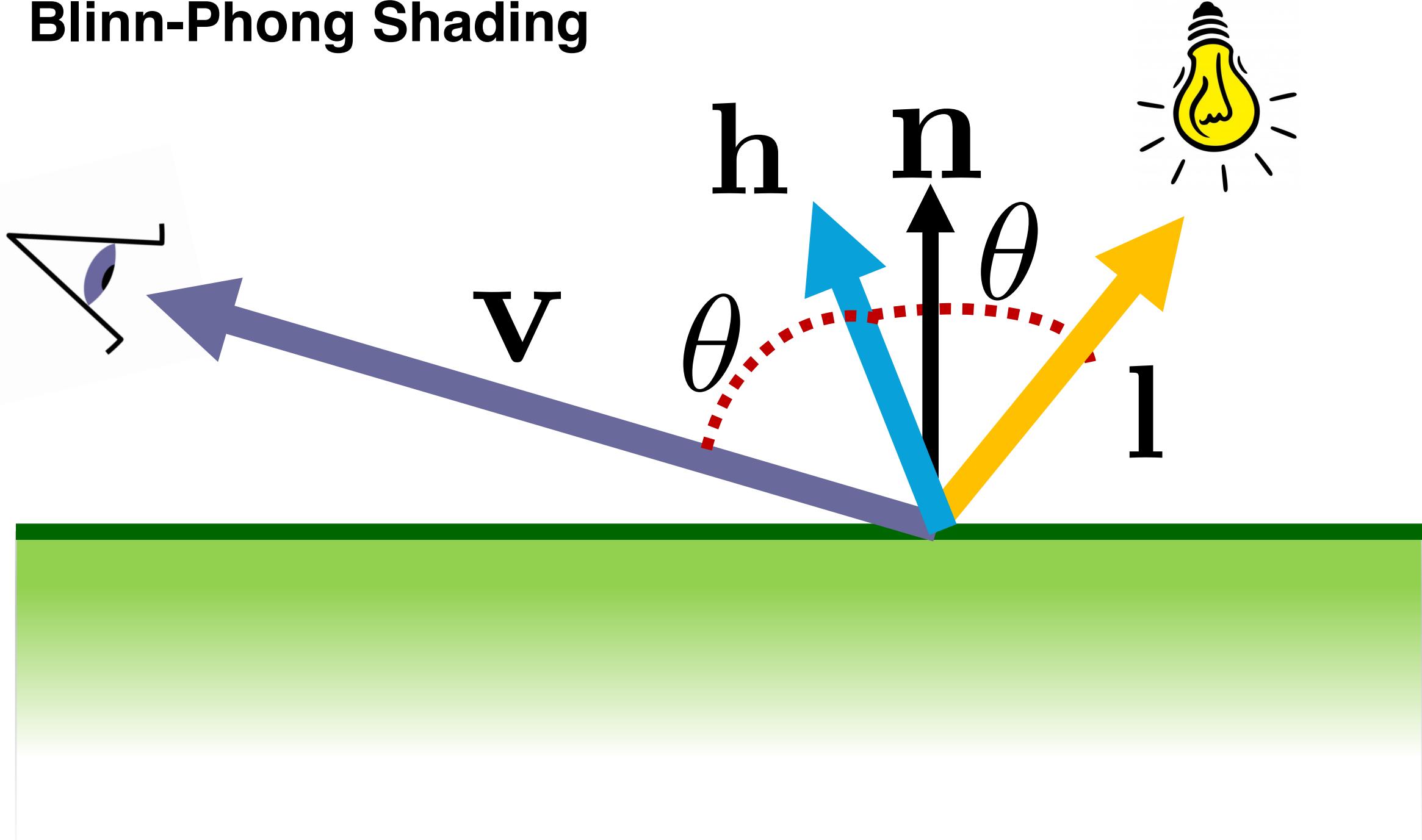


Blinn-Phong Shading Model

“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

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

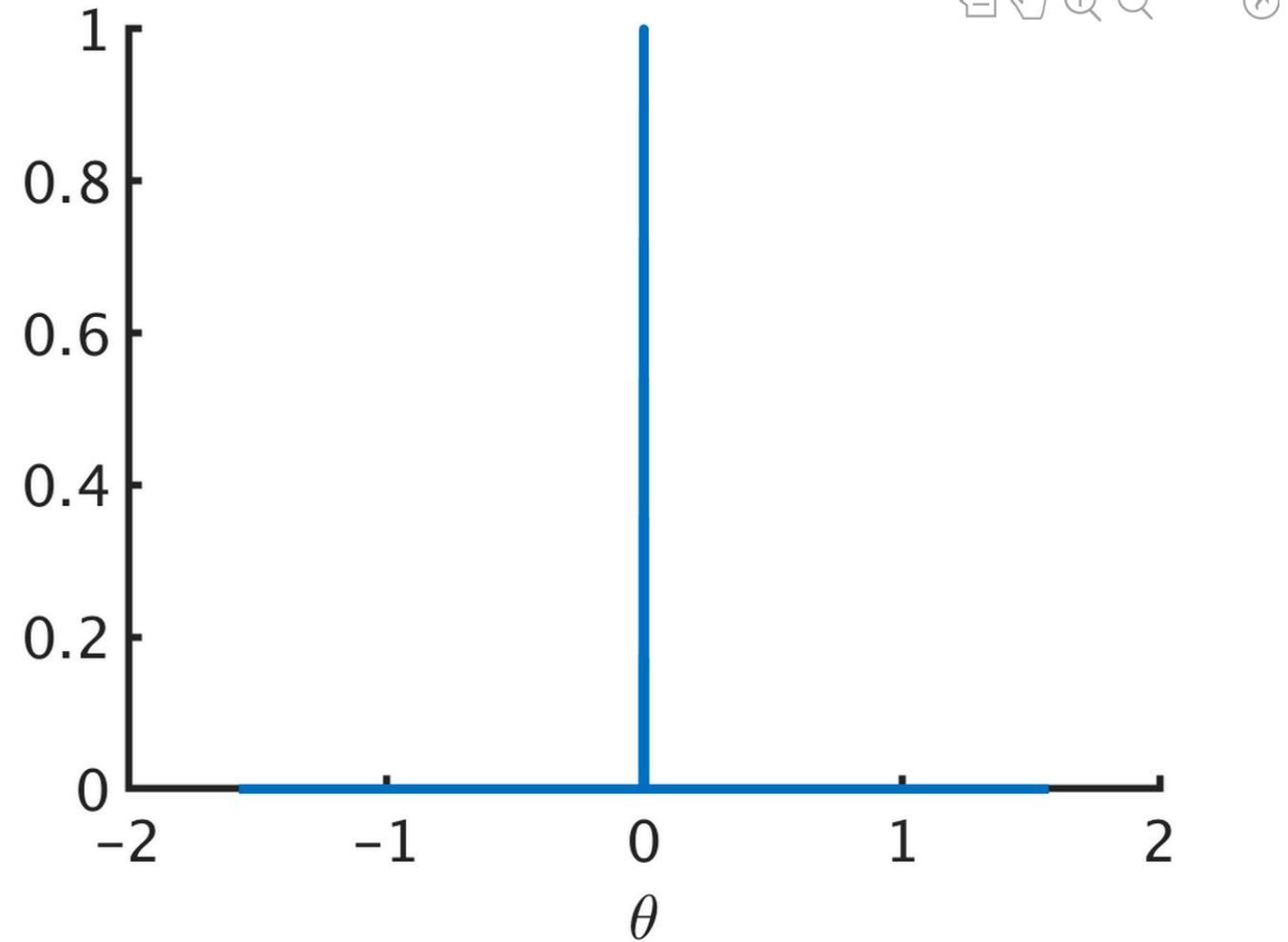
The Half Vector

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

$$L \propto \theta_h^n$$

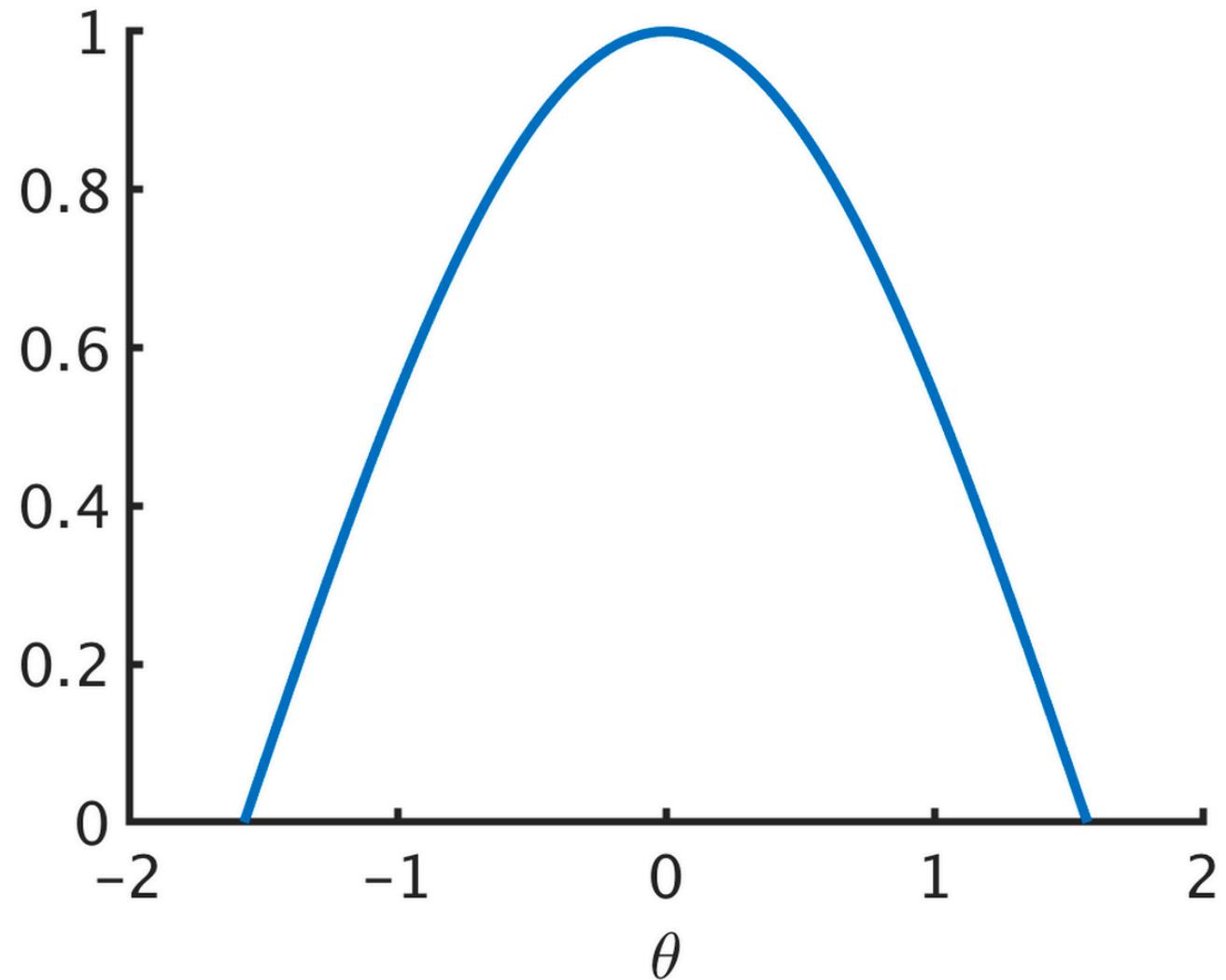
Measuring the Angle

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



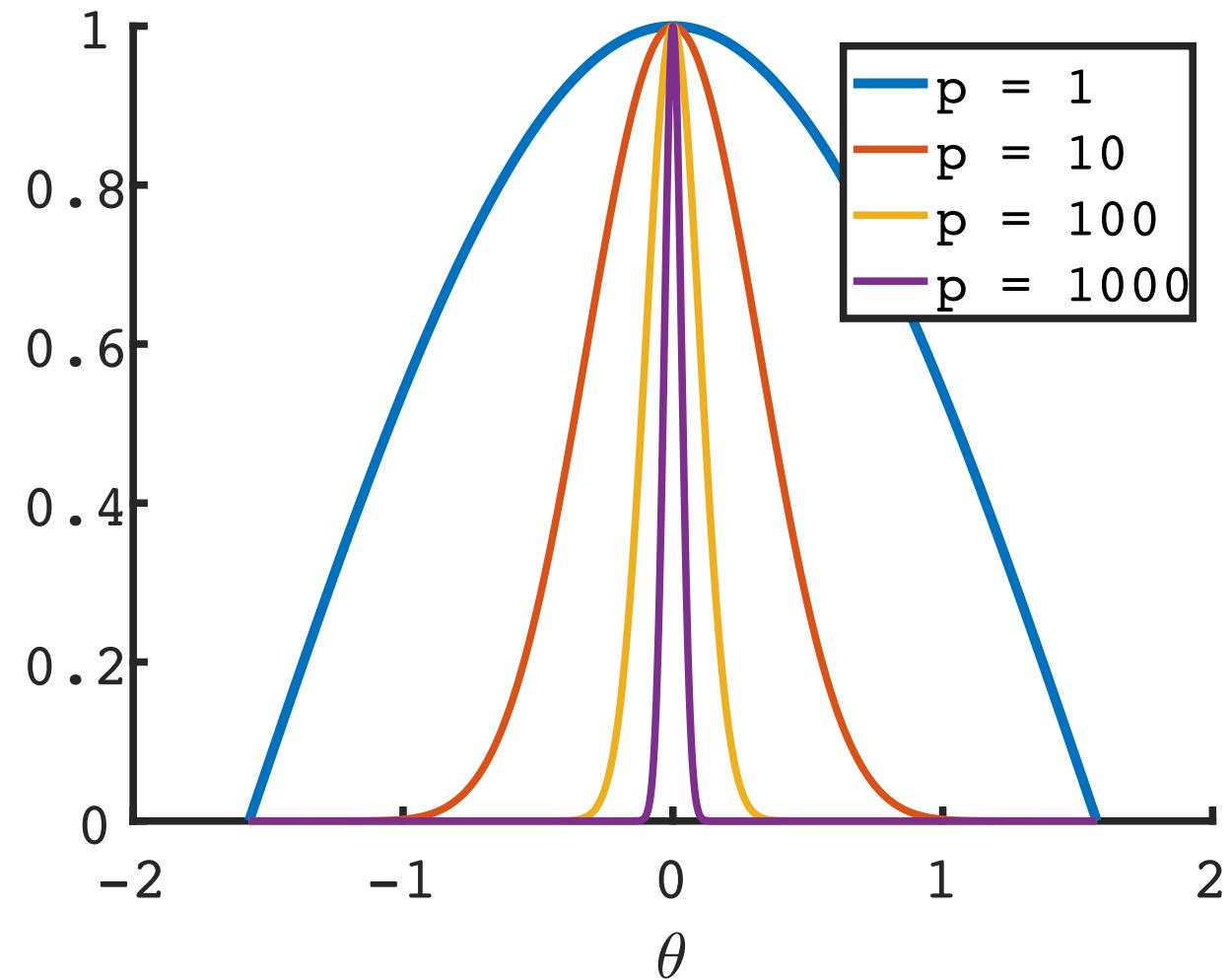
Measuring the Angle

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



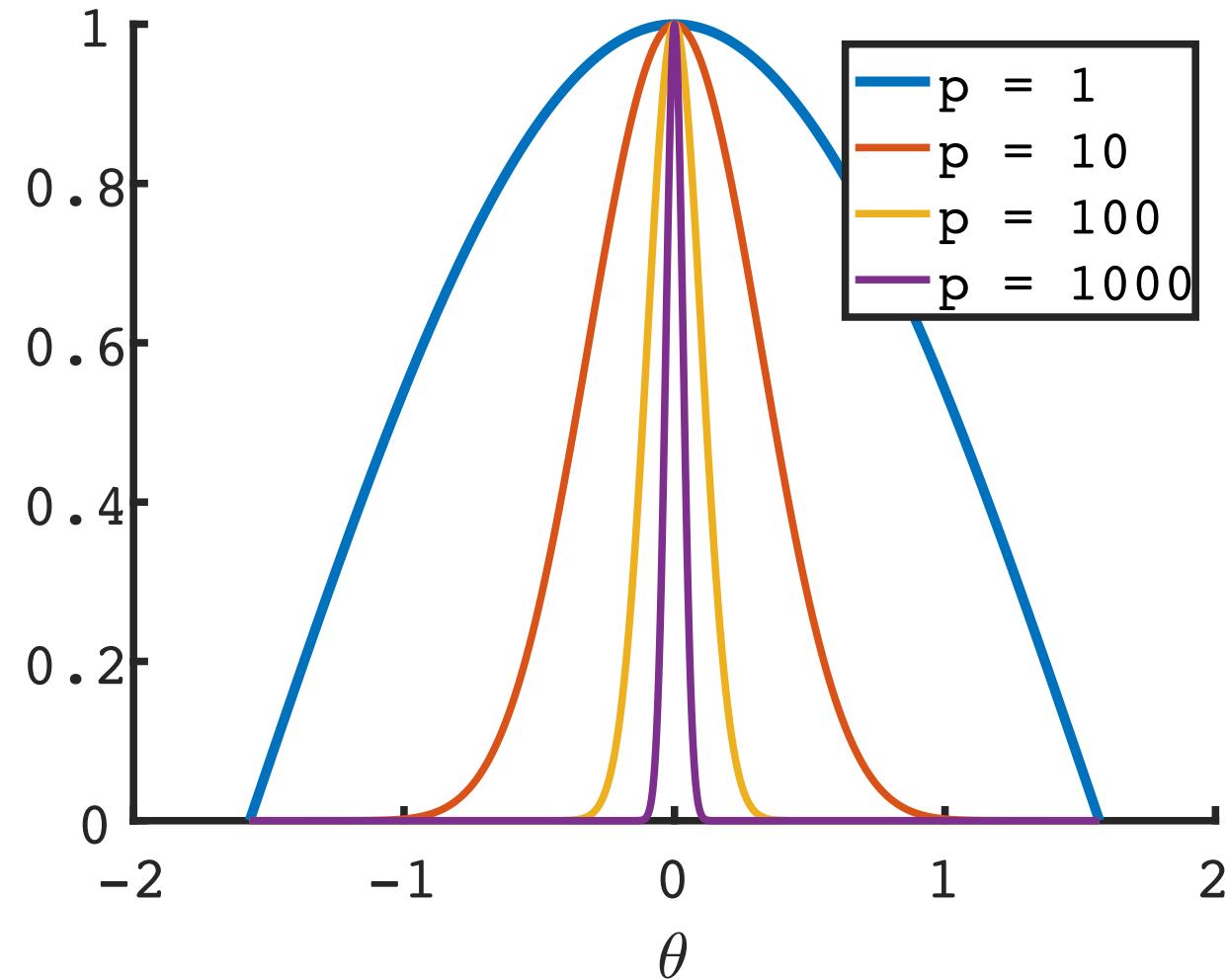
Measuring the Angle

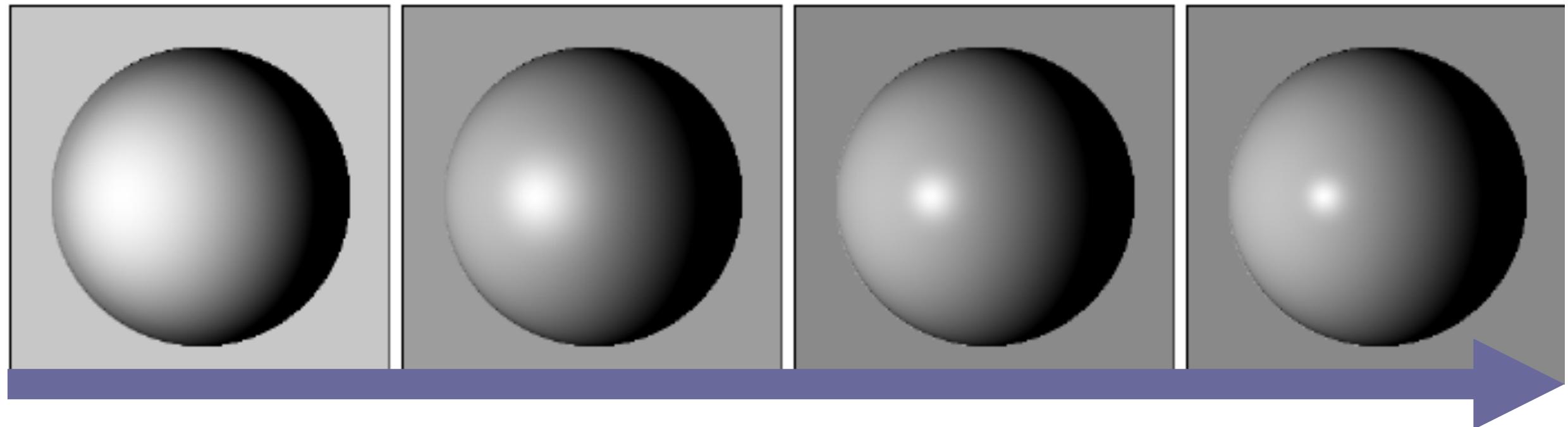
$$L = k_s \max (0, \cos(\theta_{\mathbf{h}}^{\mathbf{n}}))^p$$



Measuring the Angle

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





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.

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

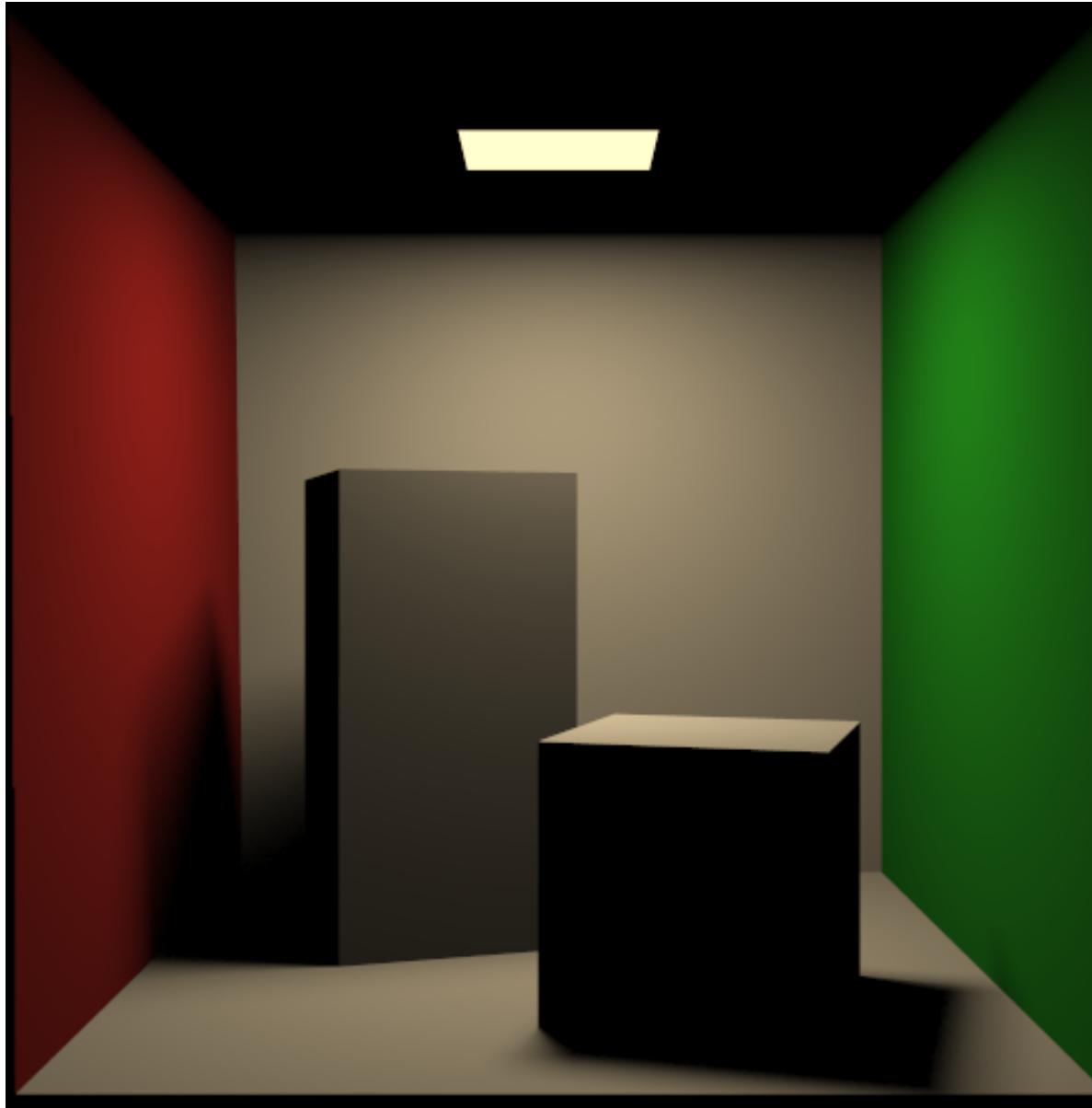
Putting it All Together: The Full Blinn-Phong Model

Light obeys the superposition principle

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

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

No Global Effects



Putting it All Together: The Full Blinn-Phong Model

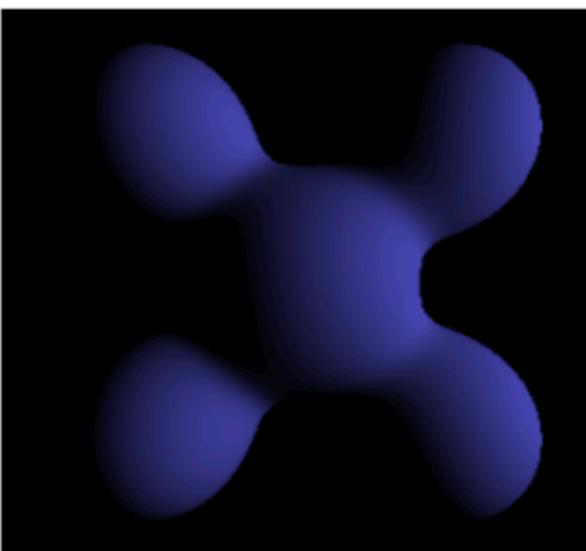
Light obeys the superposition principle

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

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



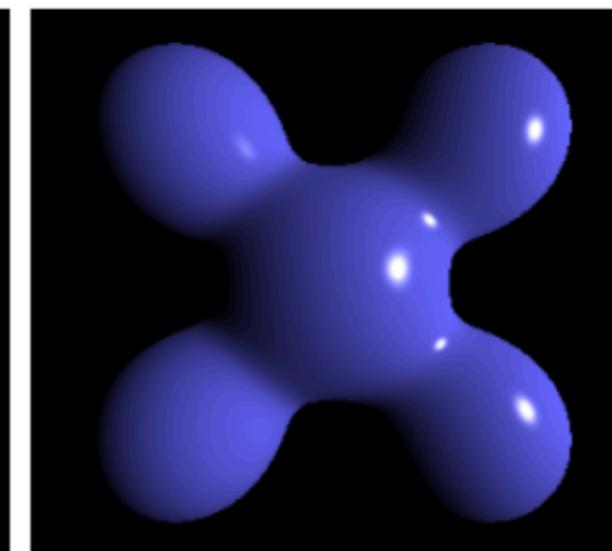
Ambient



Diffuse



Specular

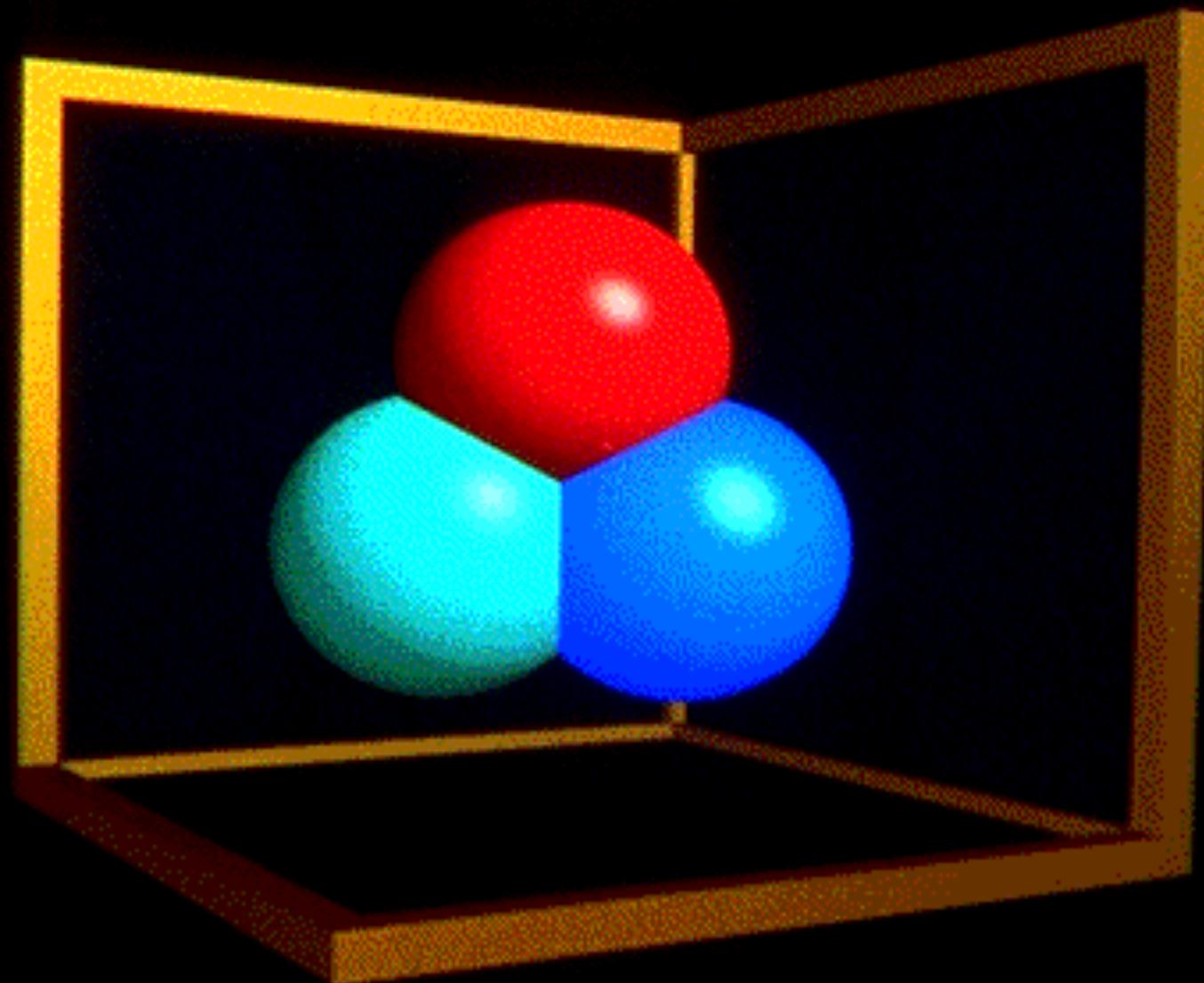


Final

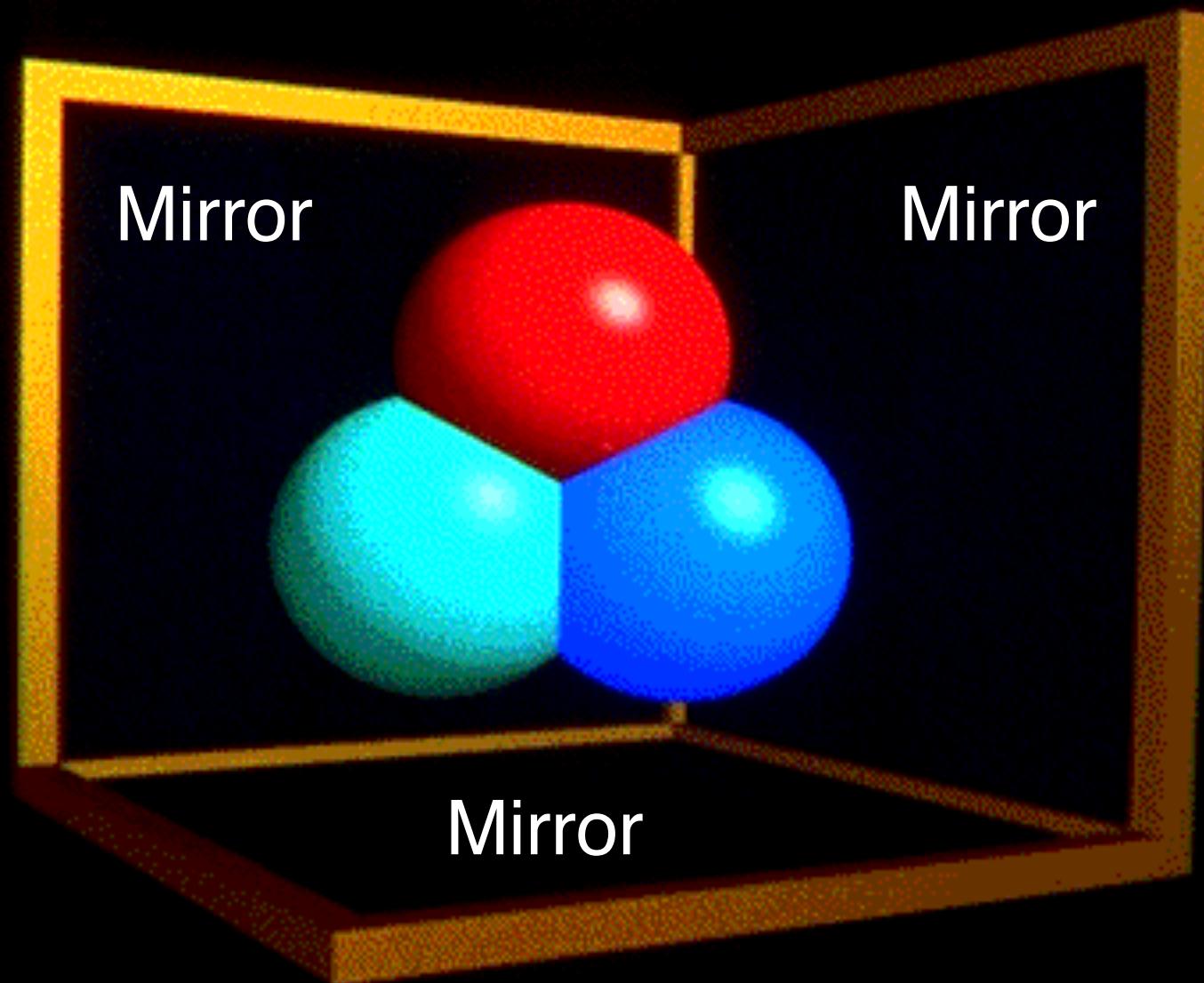
Ray Casting

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

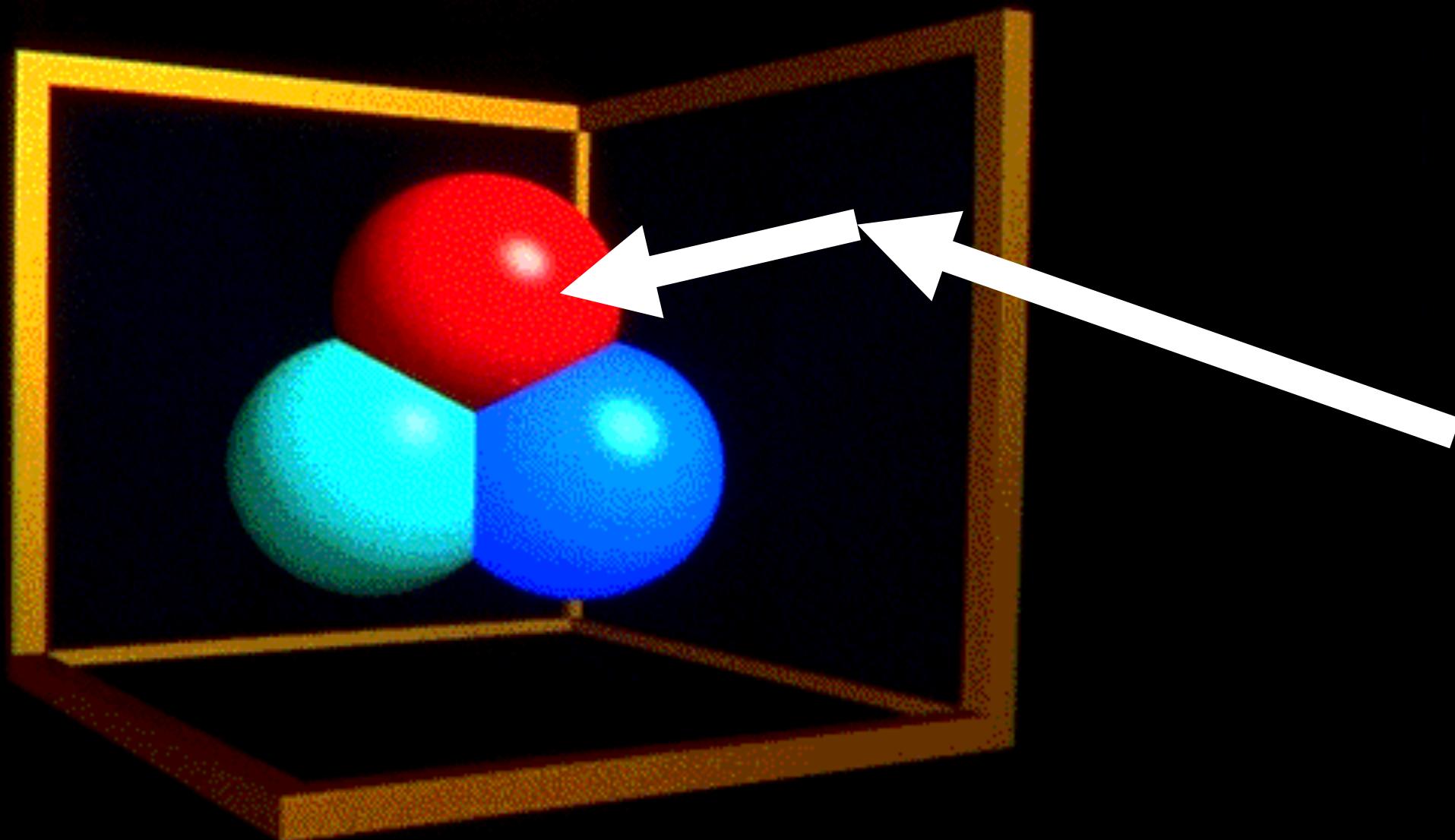
Ray Traced Image



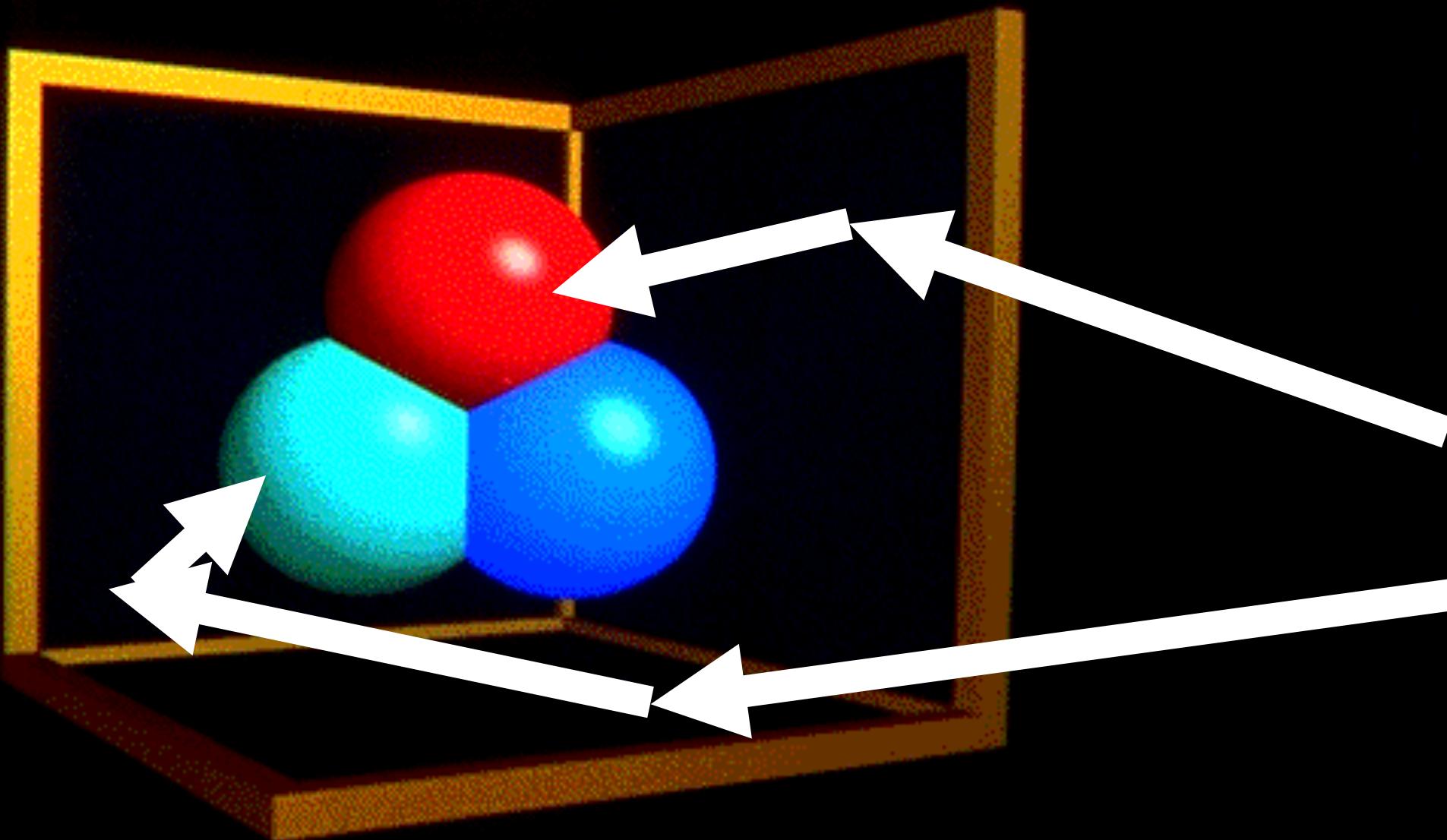
Ray Traced Image

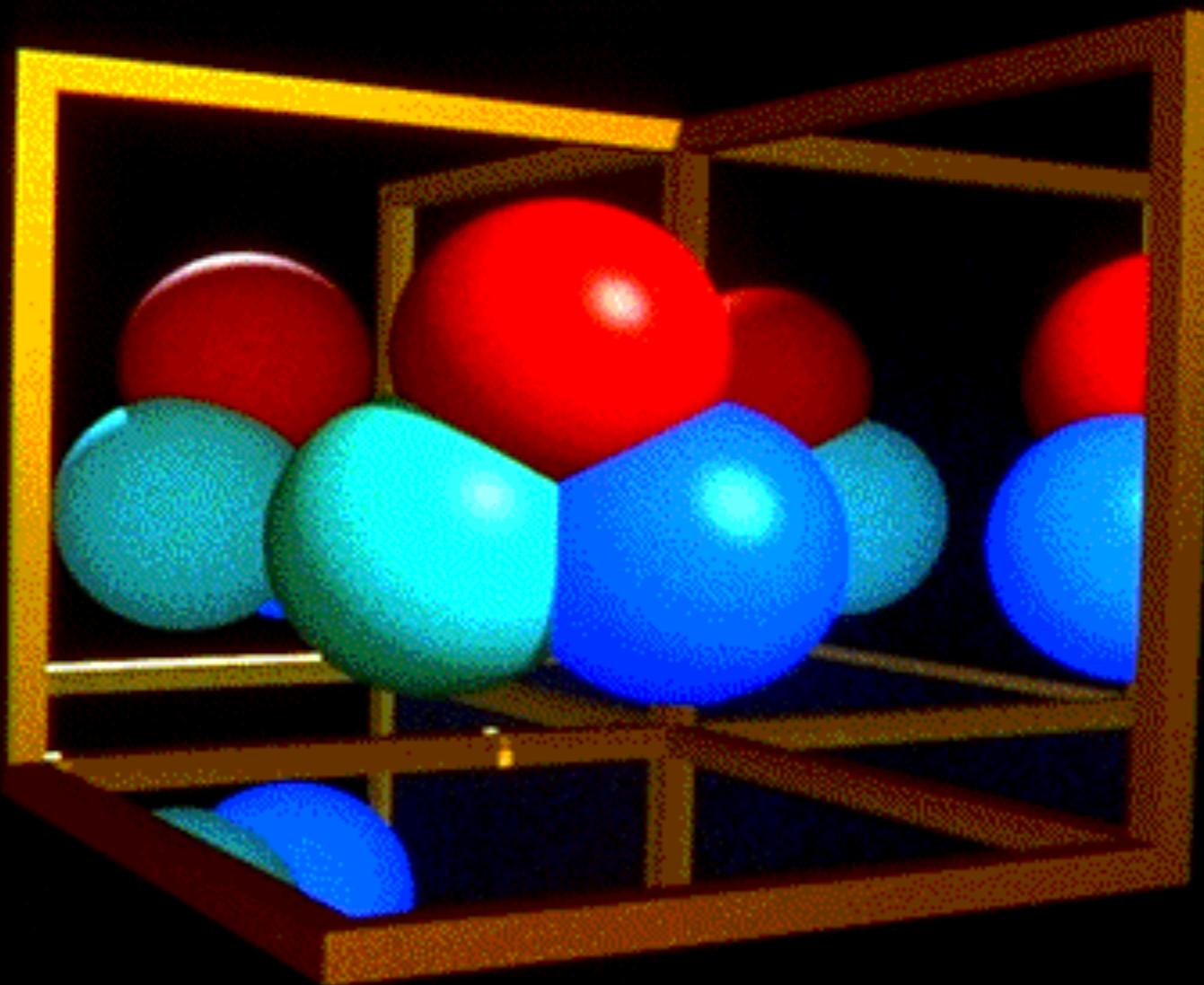


Recursive Ray Tracing

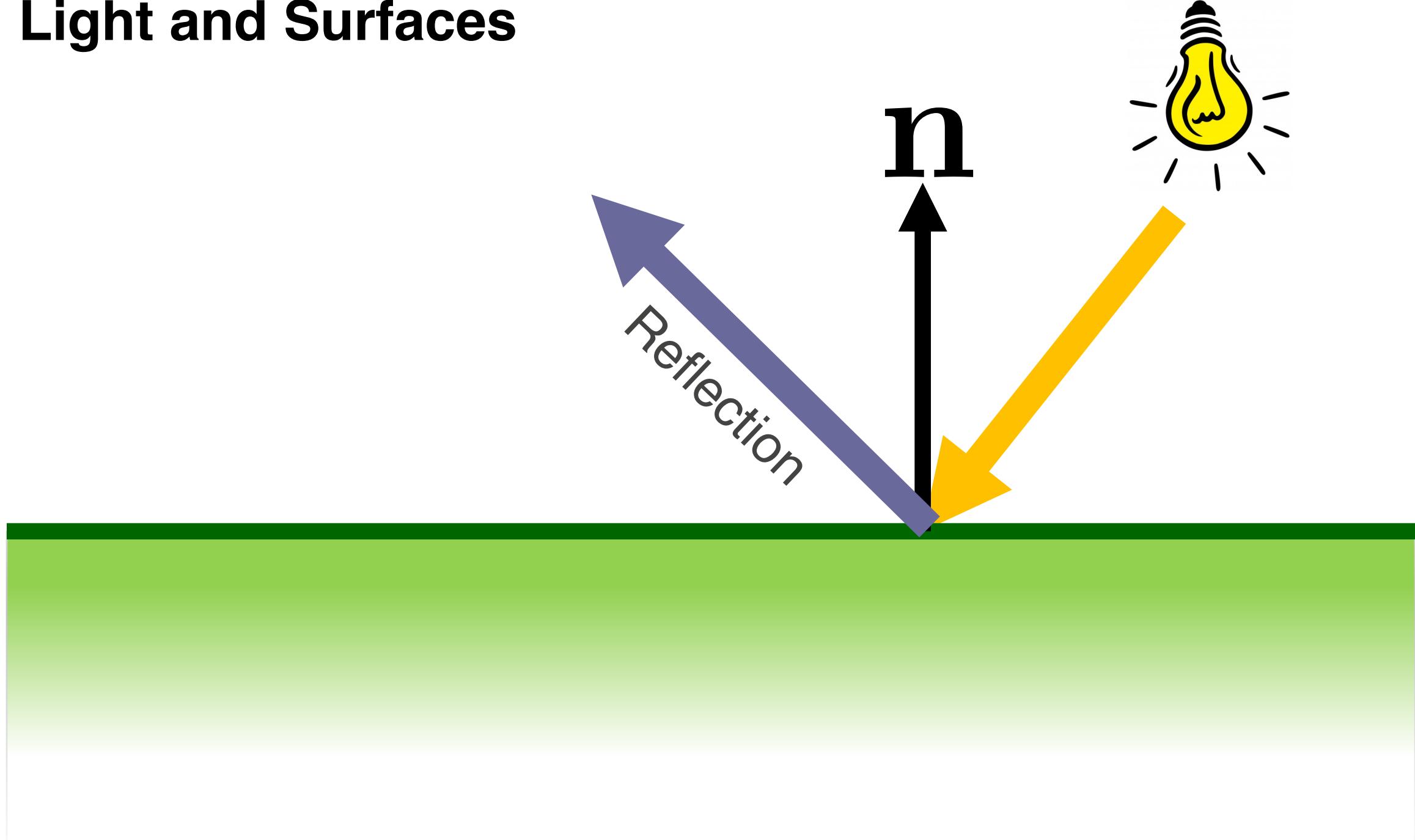


Recursive Ray Tracing

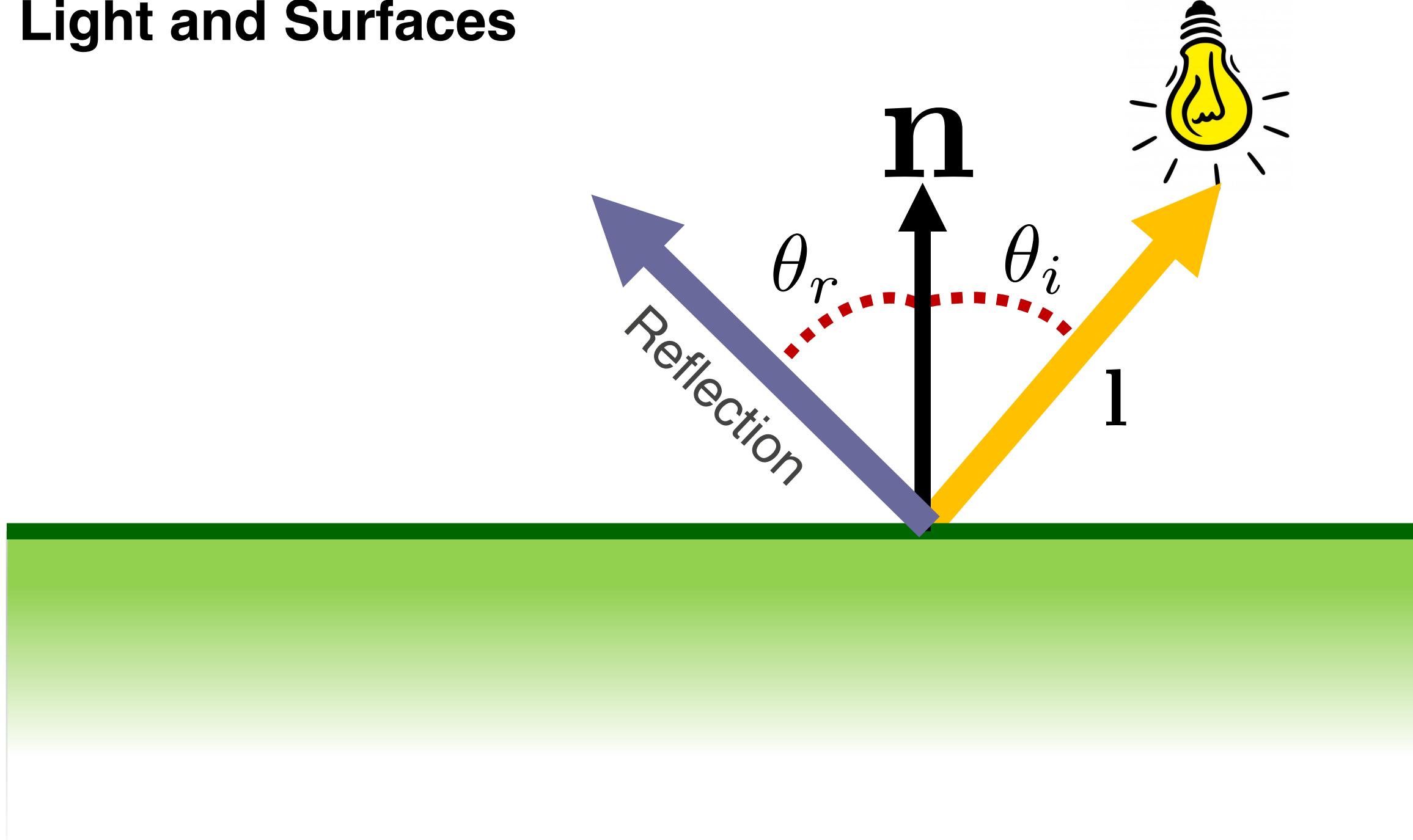




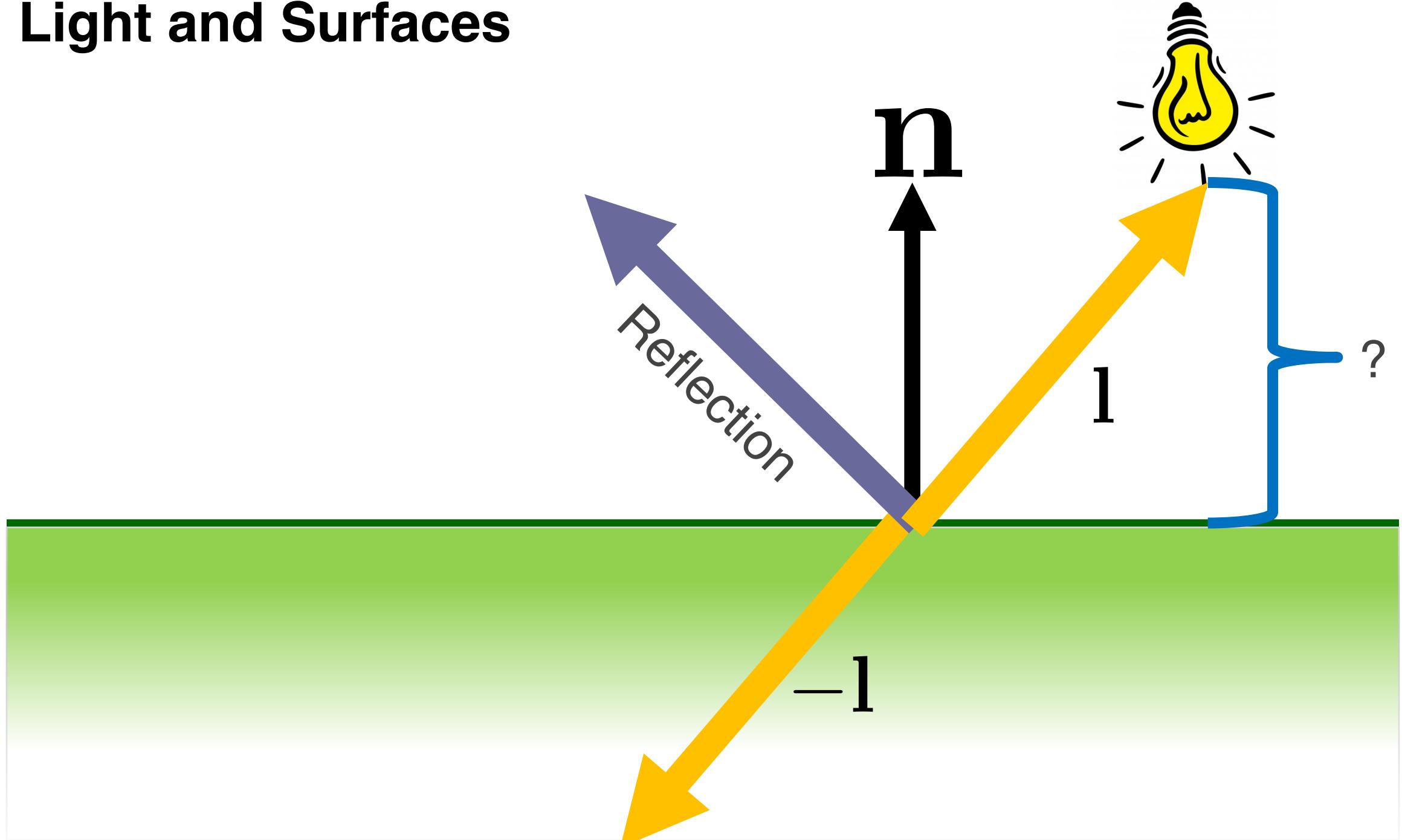
Light and Surfaces



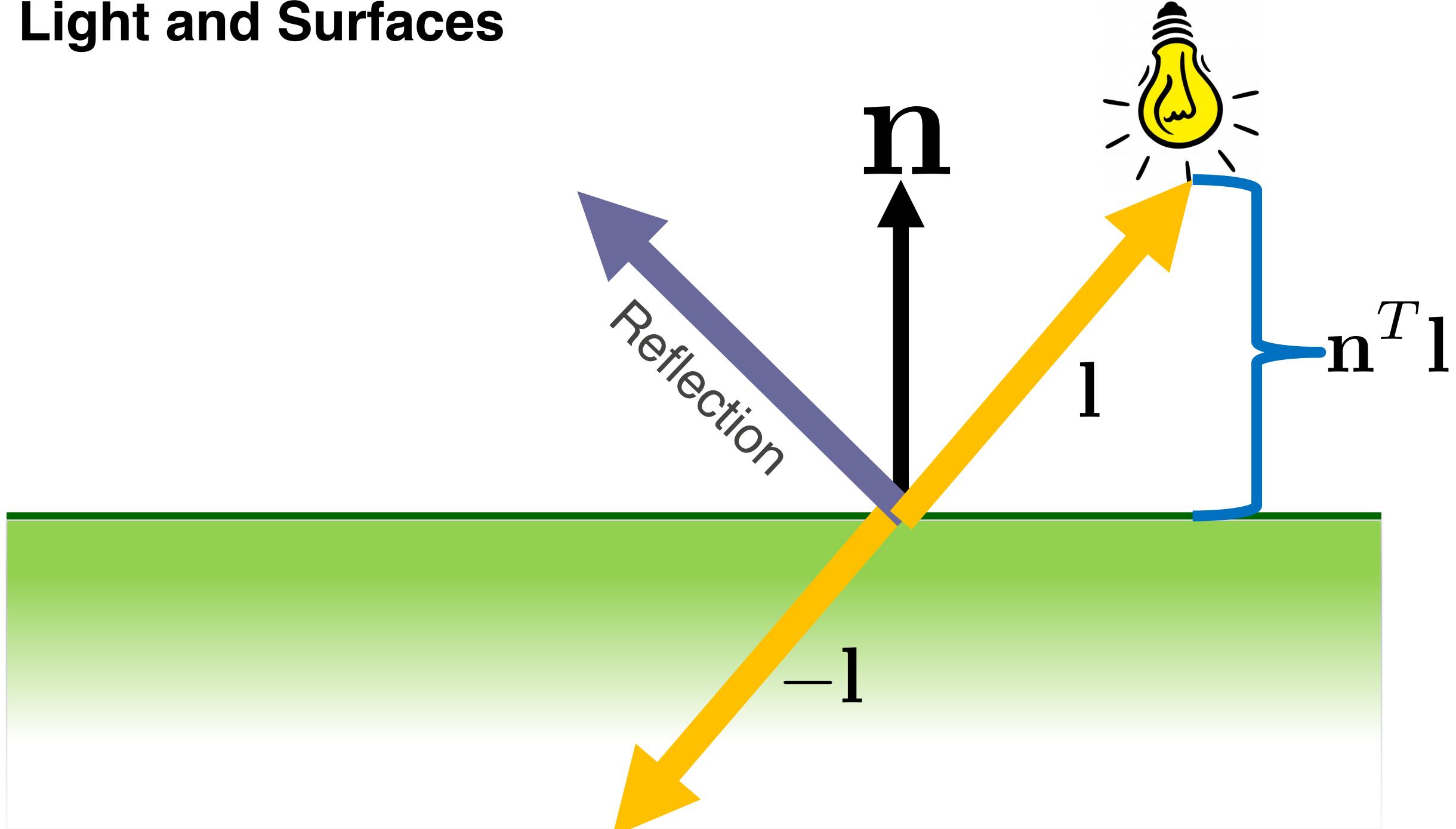
Light and Surfaces



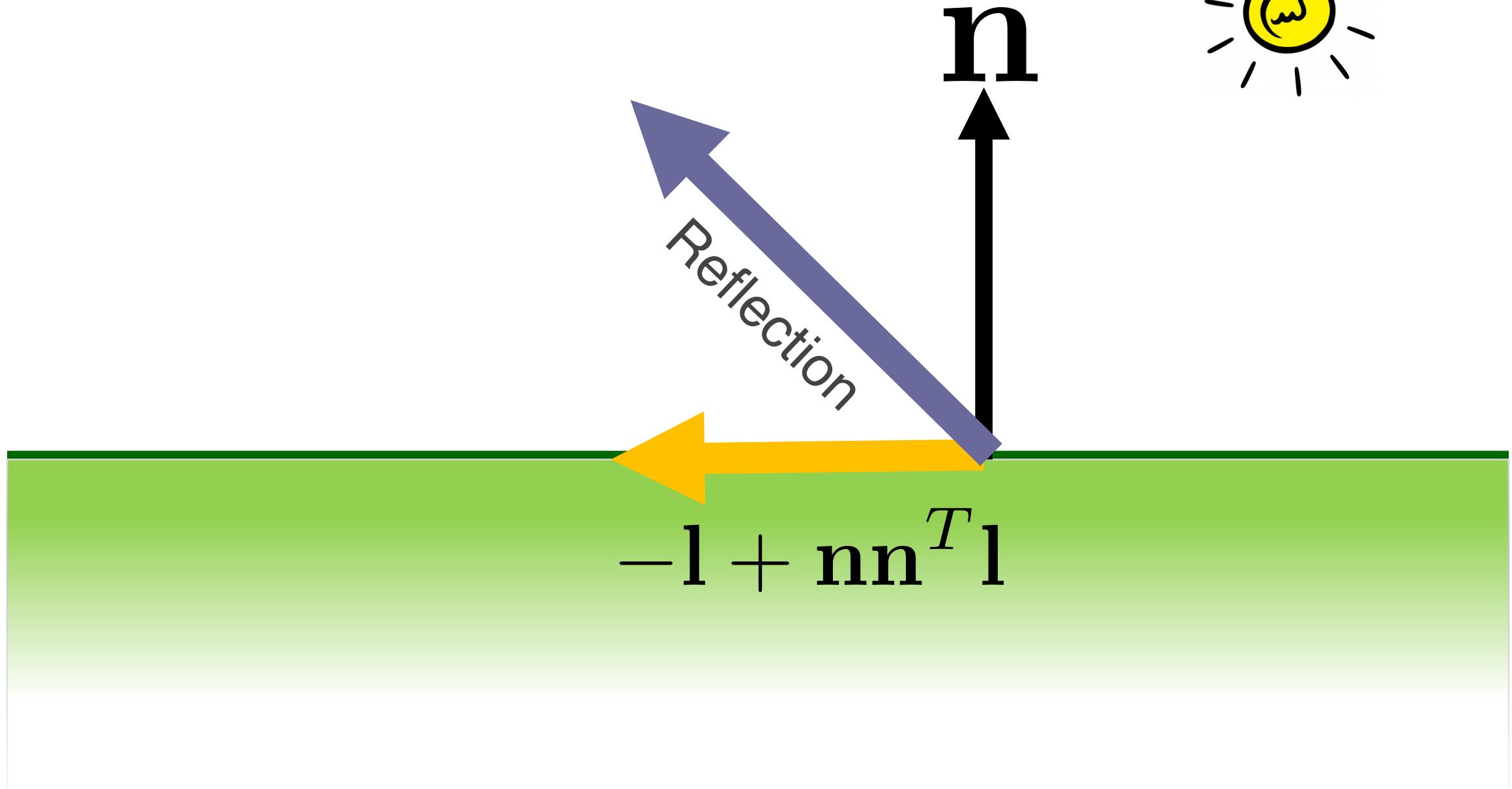
Light and Surfaces



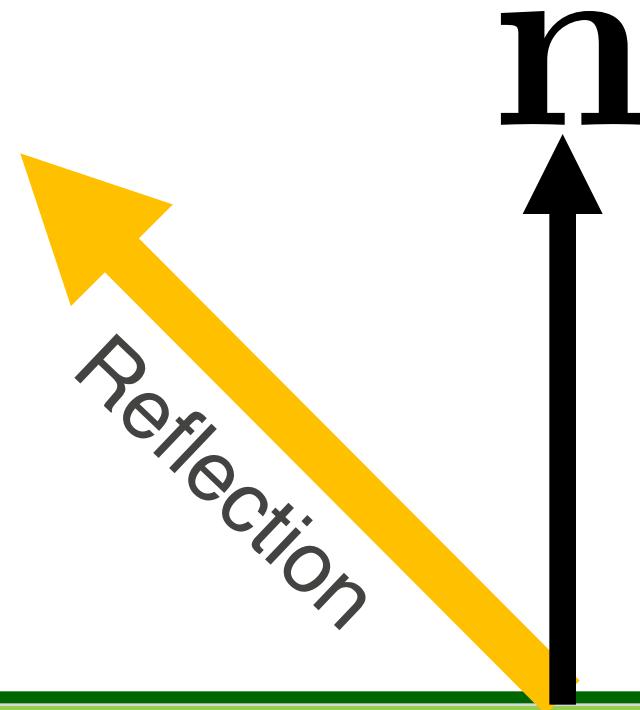
Light and Surfaces



Light and Surfaces

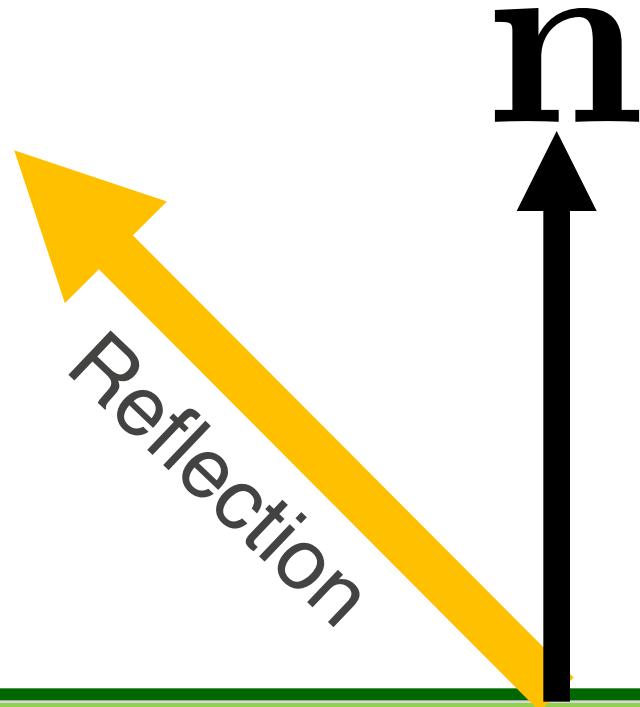
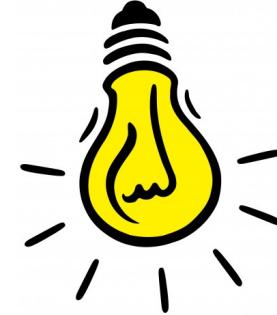


Light and Surfaces



$$-1 + 2\mathbf{n}\mathbf{n}^T\mathbf{l}$$

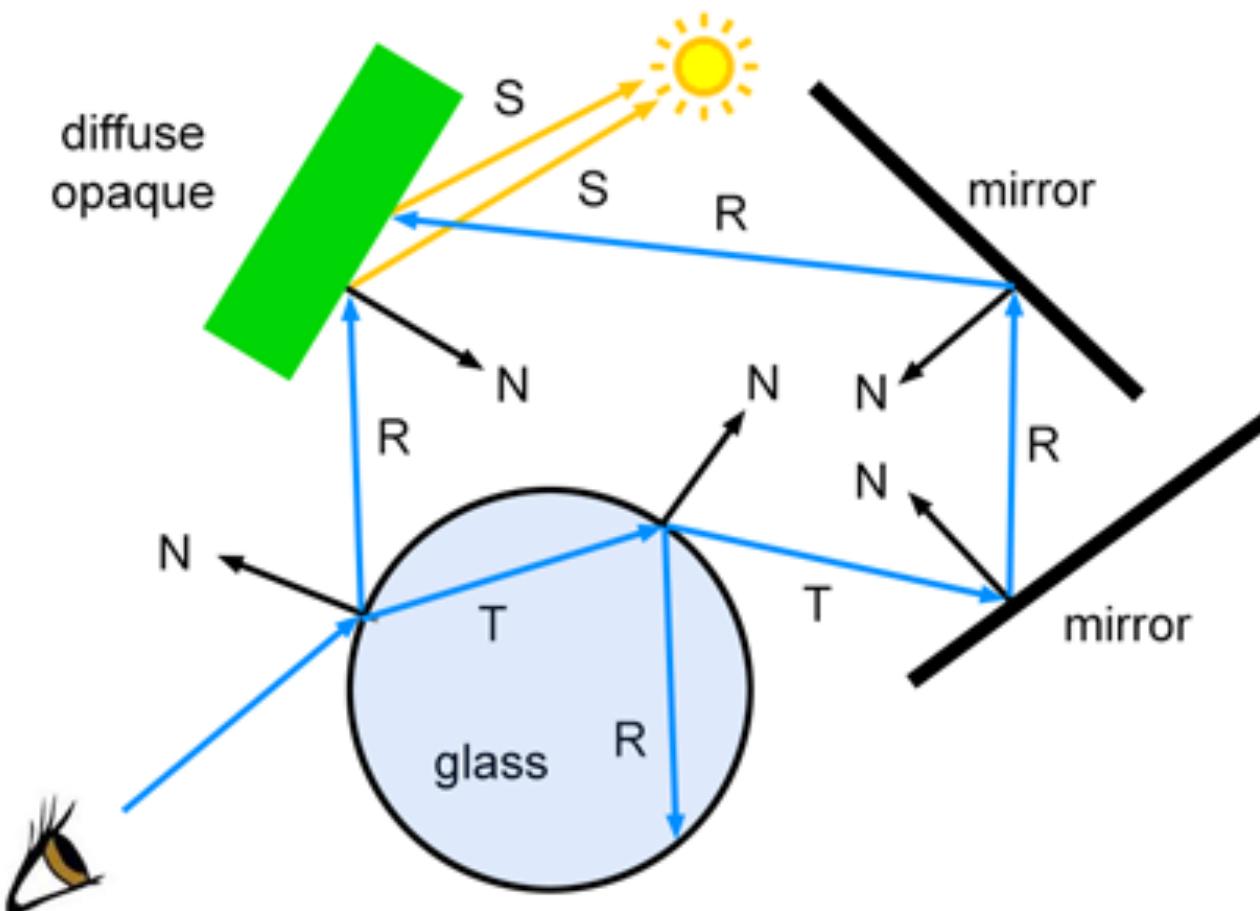
Light and Surfaces



$$(I + 2nn^T)(-l)$$

```
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



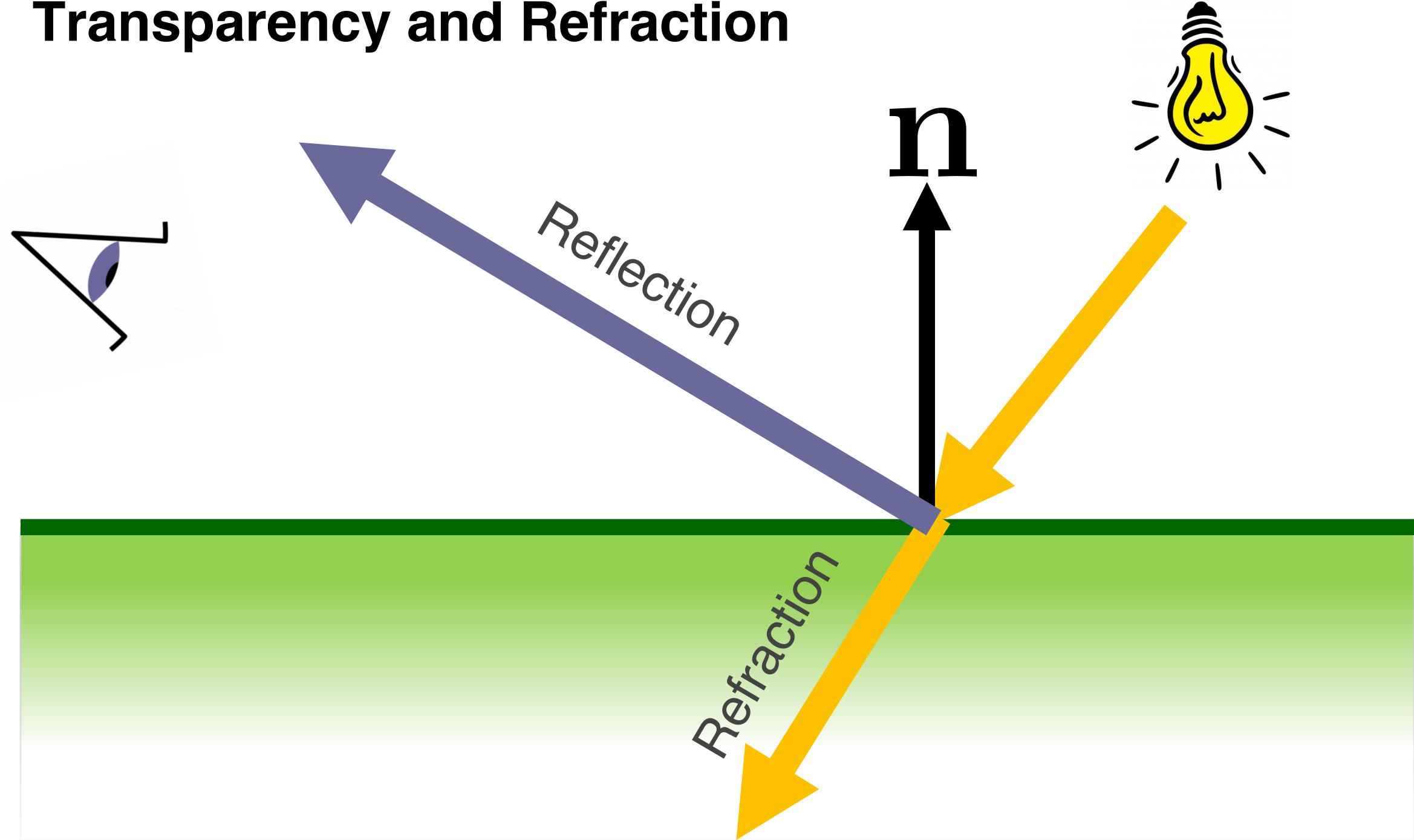
© www.scratchapixel.com

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

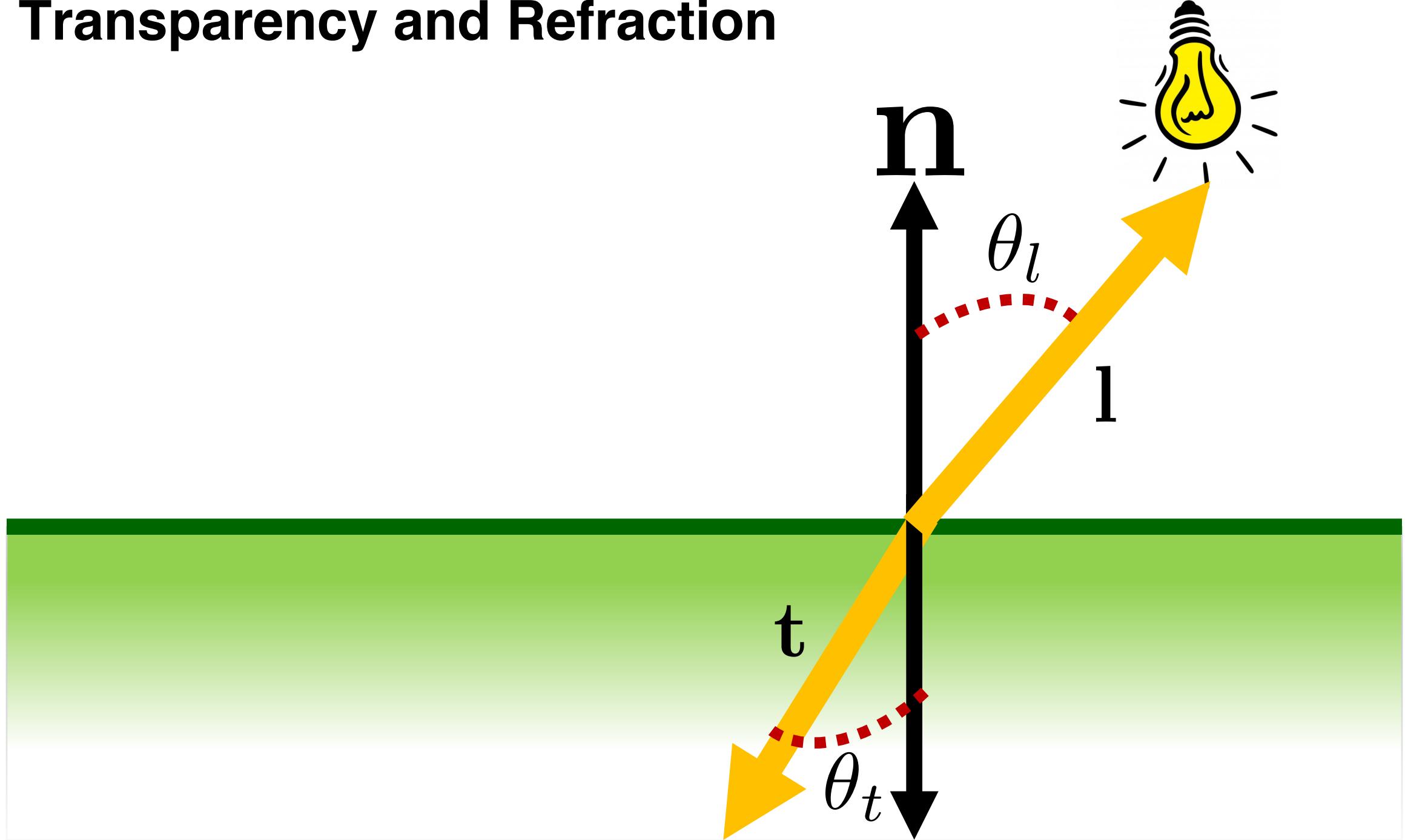
Transparency and Refraction



Transparency and Refraction



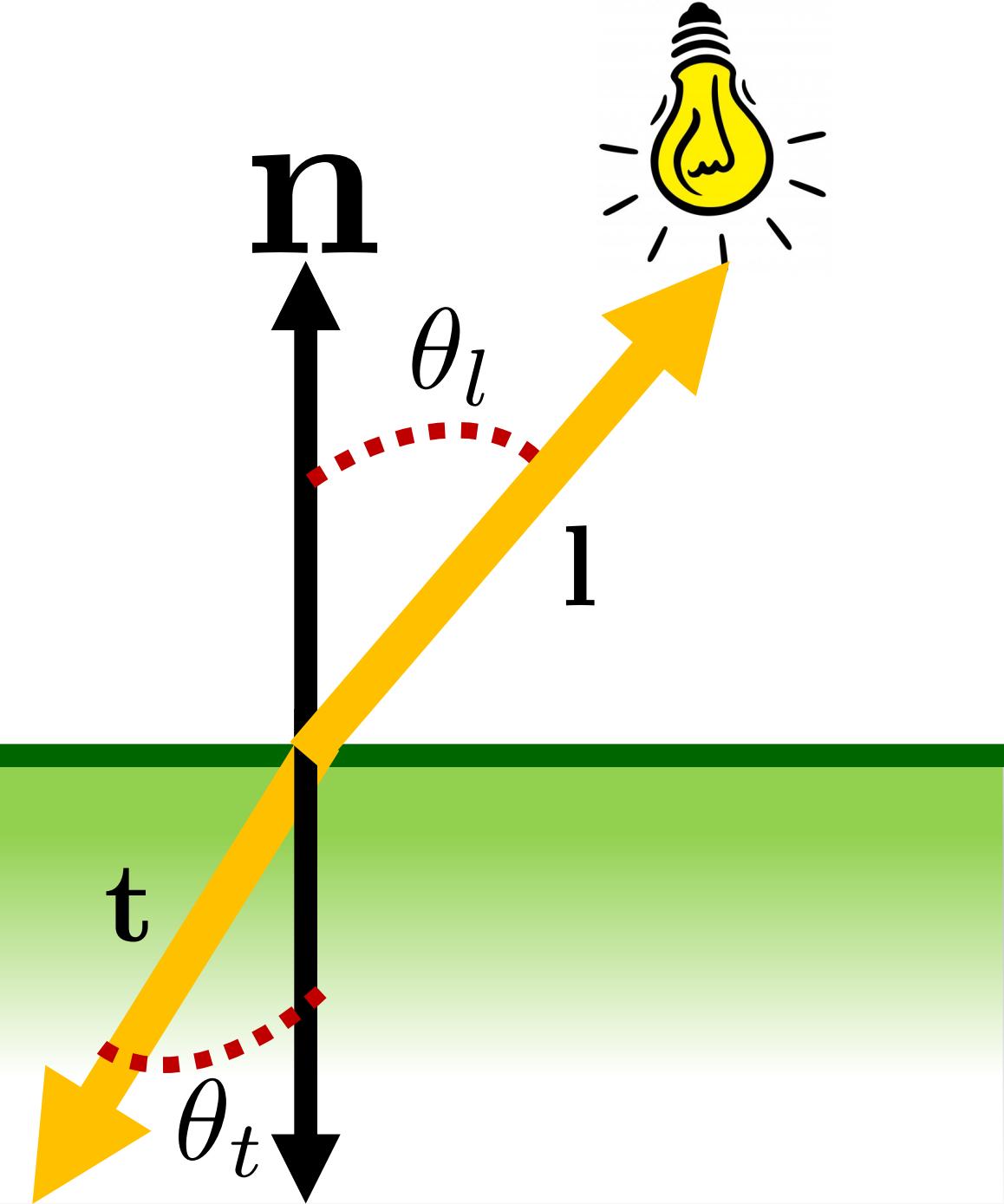
Transparency and Refraction



Snell's Law

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

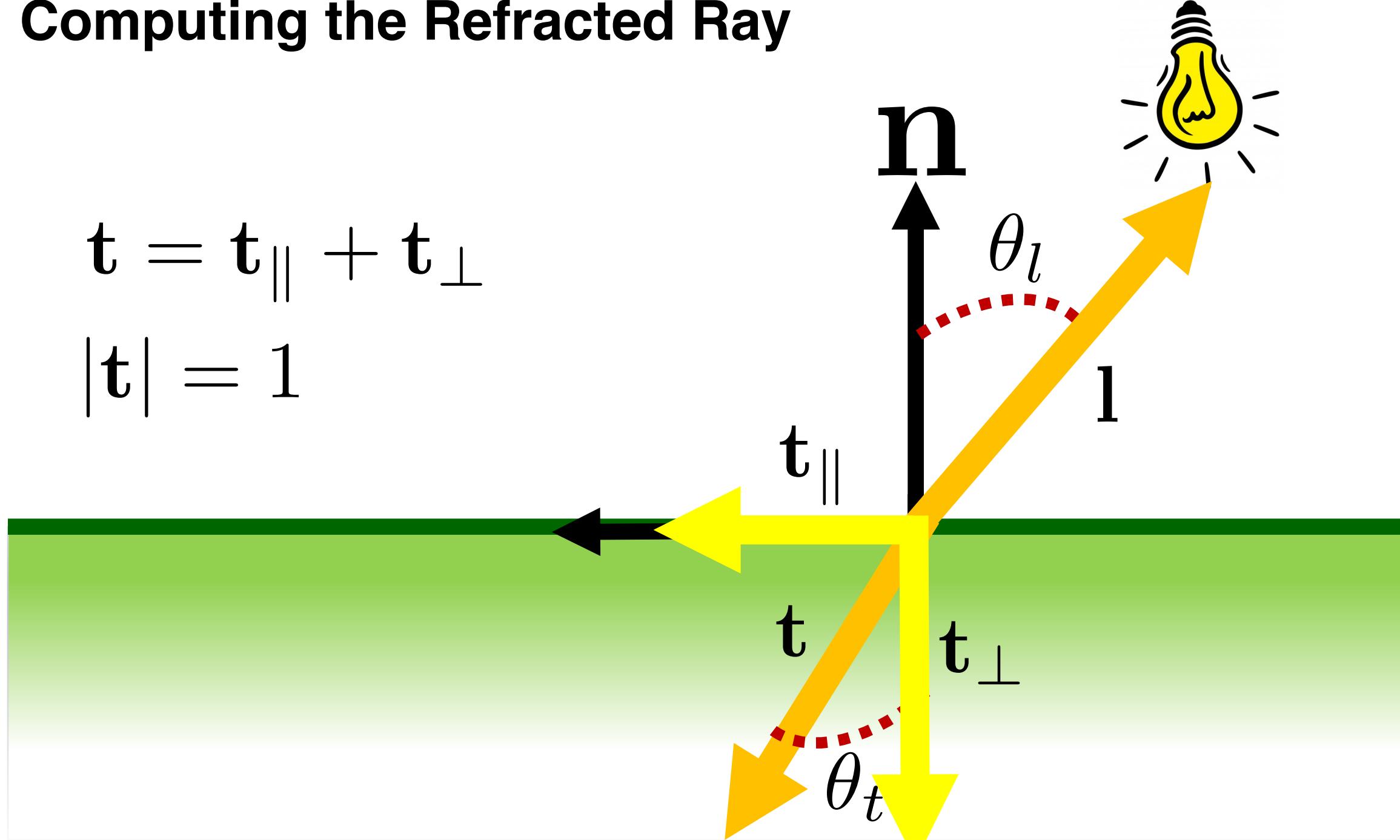
Indices of Refraction



Computing the Refracted Ray

$$t = t_{\parallel} + t_{\perp}$$

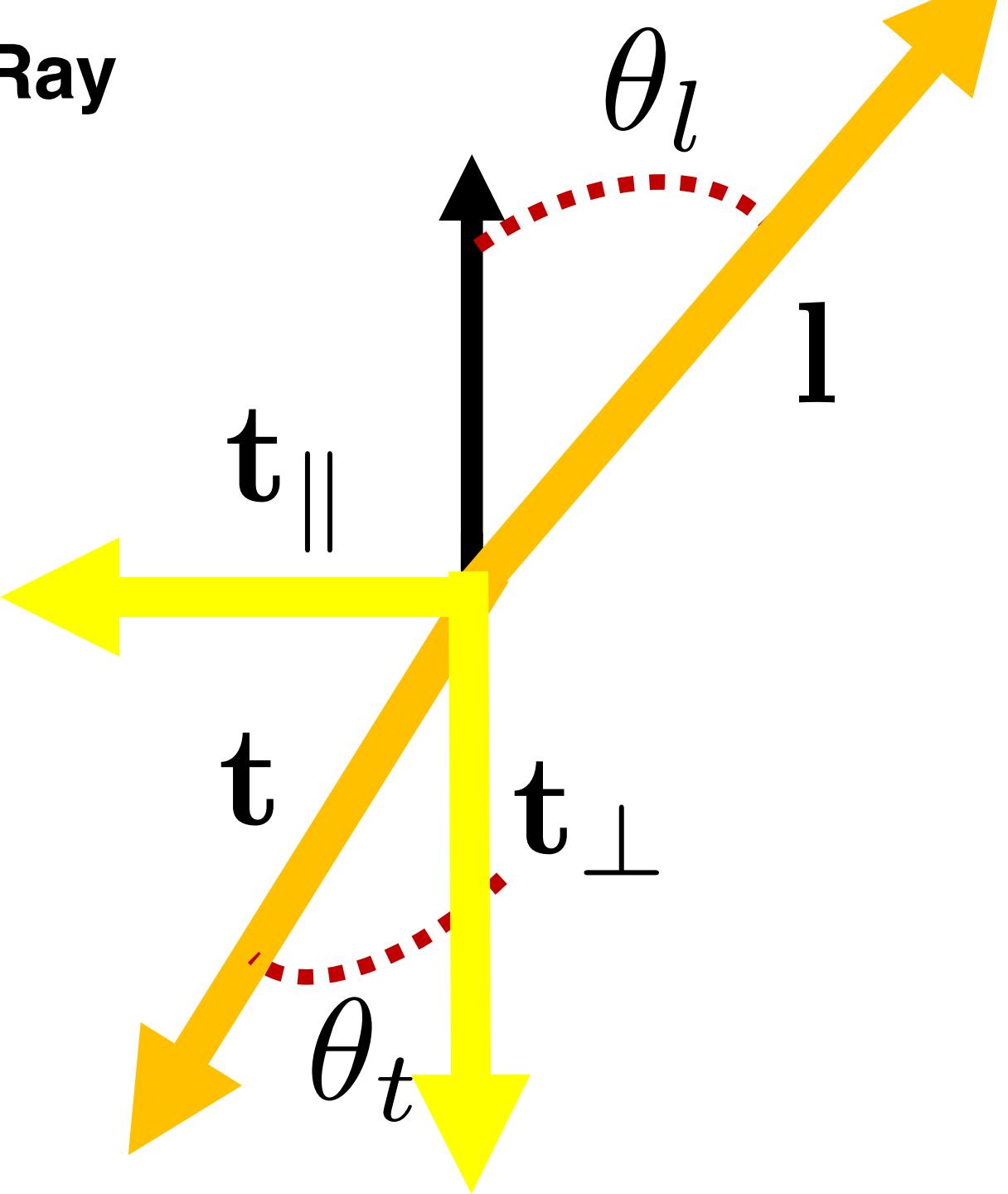
$$|t| = 1$$



Computing the Refracted Ray

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

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



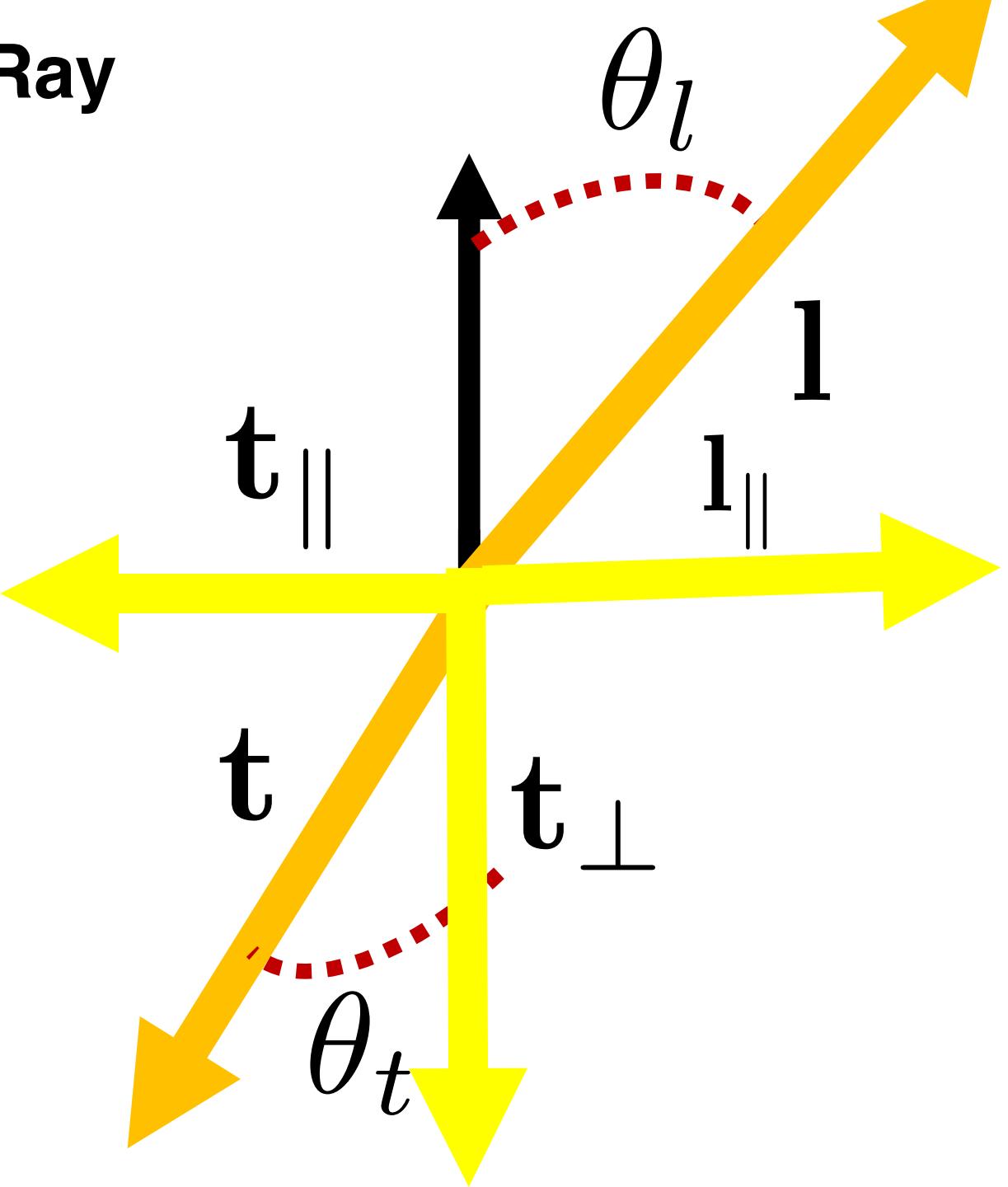
Computing the Refracted Ray

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

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

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

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

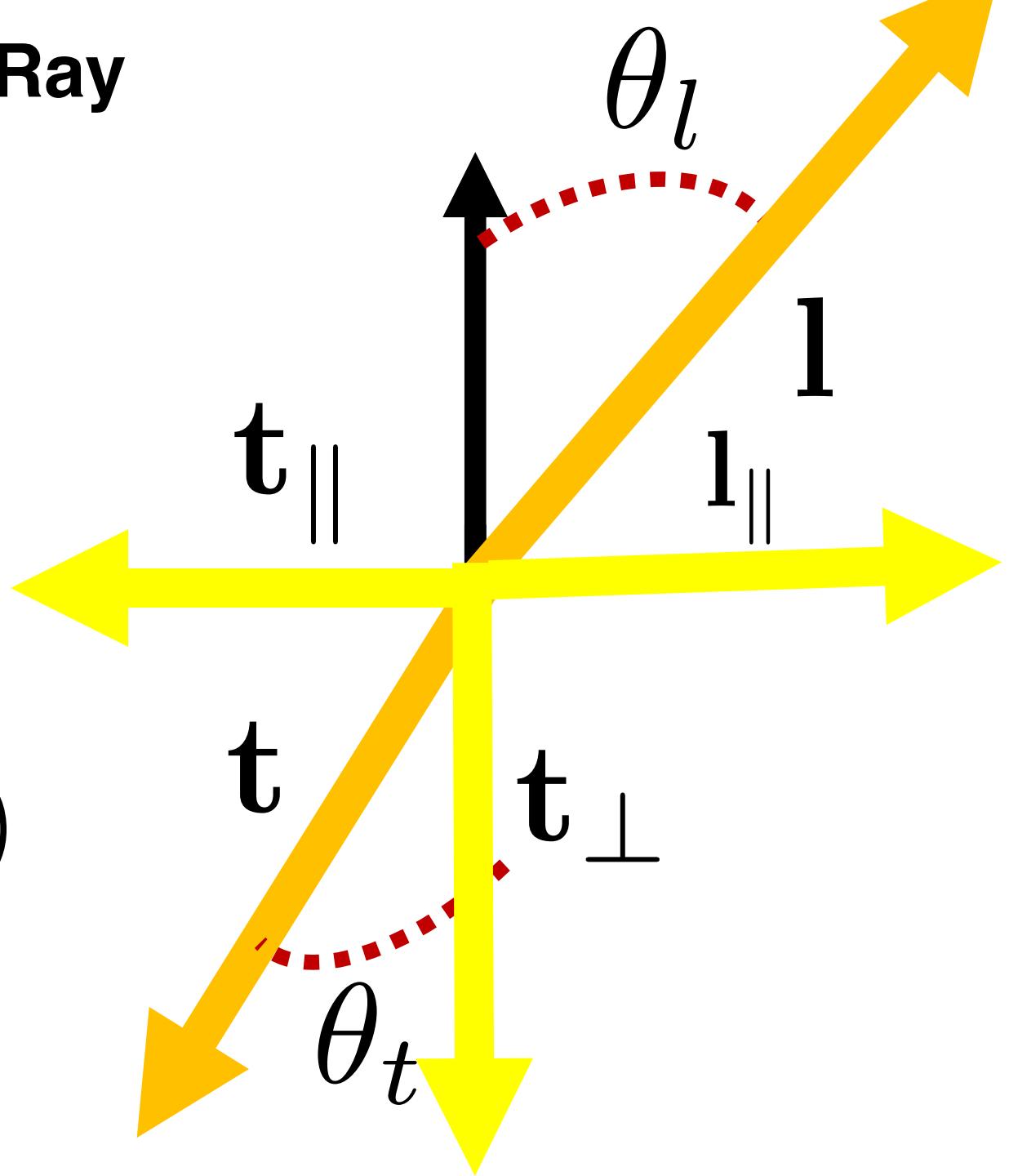


Computing the Refracted Ray

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

$$t_{\parallel} = -\frac{c_l}{c_t} (\mathbf{l} - \mathbf{n}\mathbf{n}^T \mathbf{l})$$

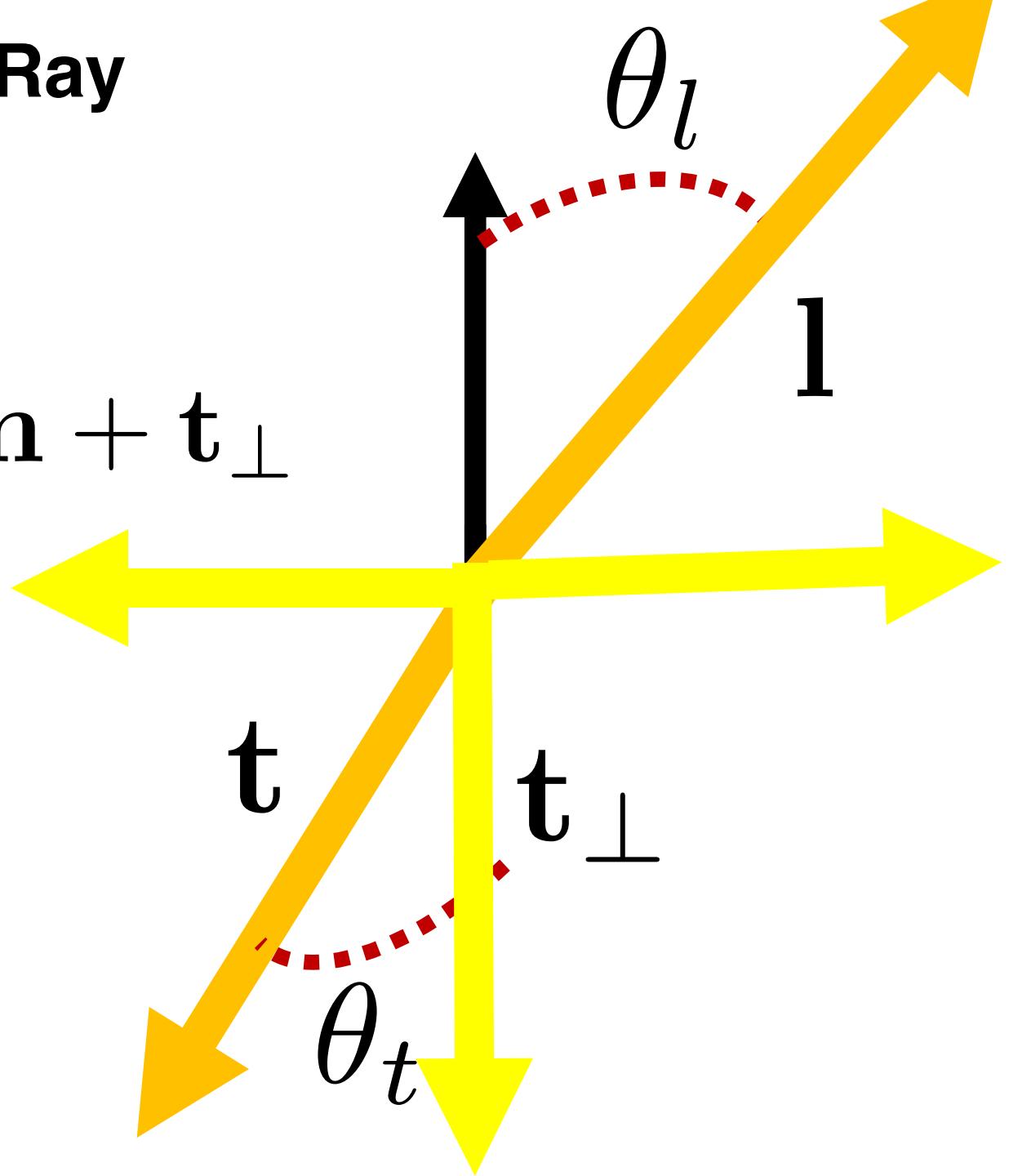
$$t_{\parallel} = -\frac{c_l}{c_t} (\mathbf{l} - \cos \theta_l \mathbf{n})$$



Computing the Refracted Ray

$$t = t_{\parallel} + t_{\perp}$$

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



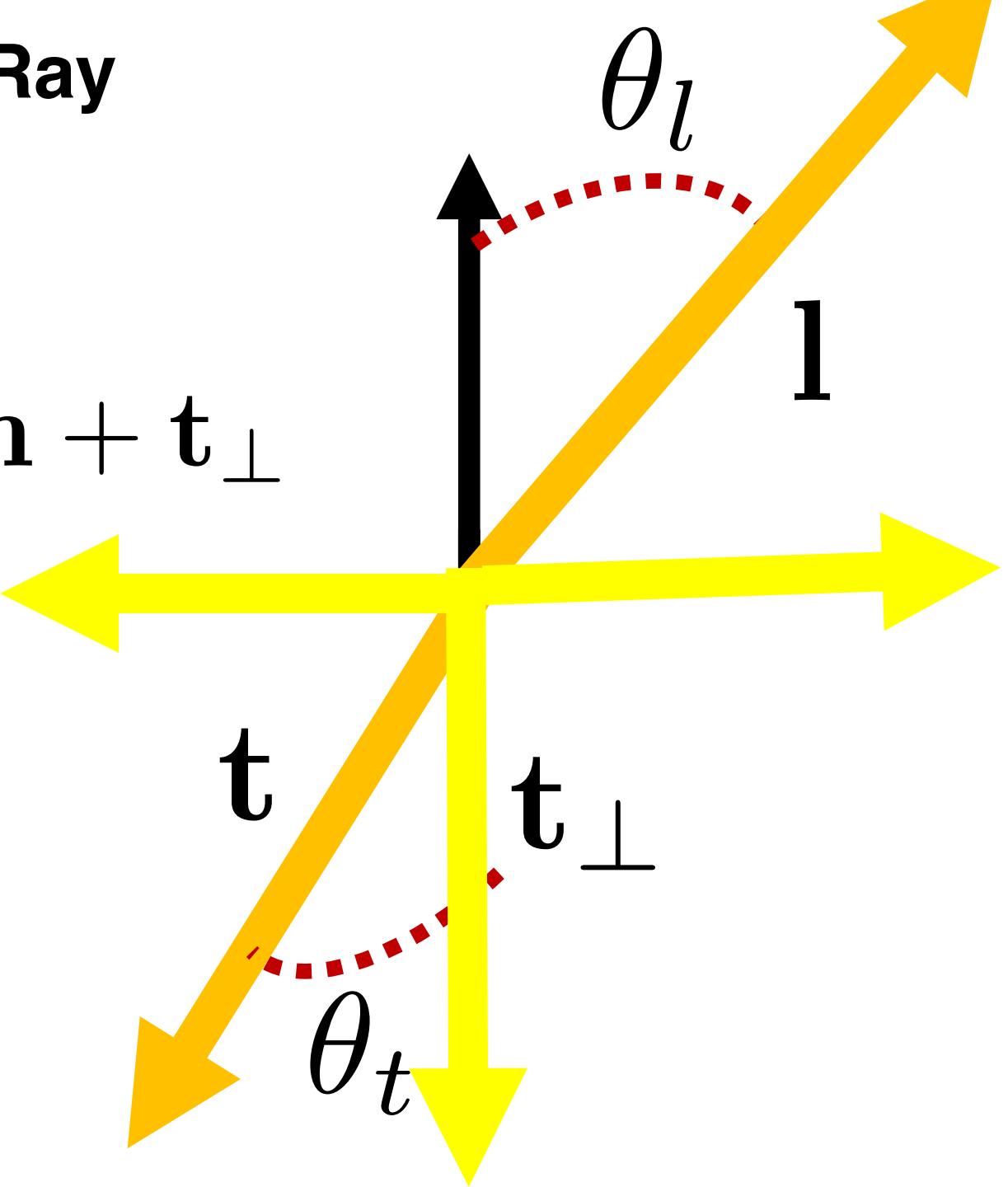
Computing the Refracted Ray

$$t = t_{\parallel} + t_{\perp}$$

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

$$t_{\perp} = -\alpha \mathbf{n}$$

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



Computing the Refracted Ray

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

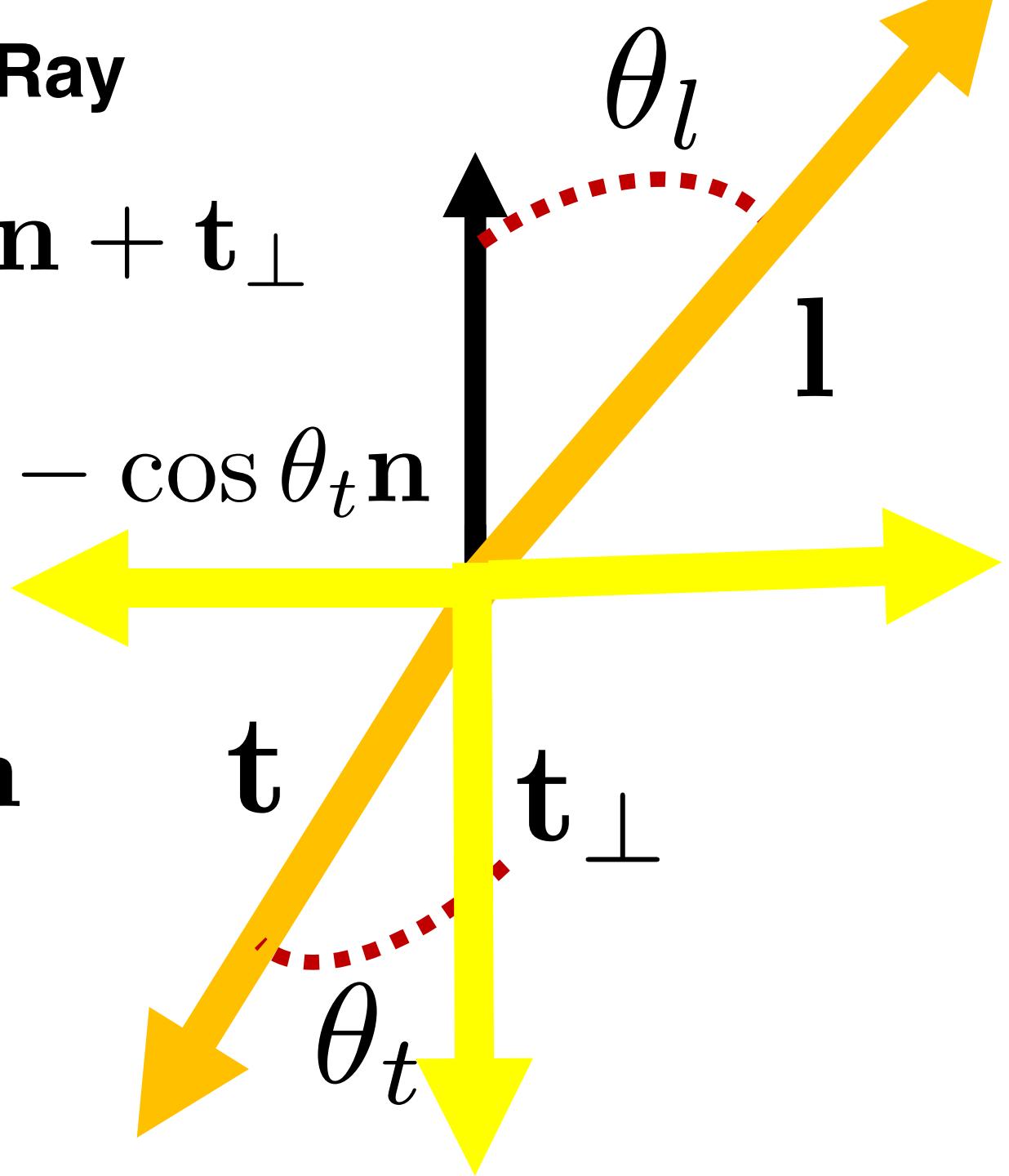
$$\mathbf{t} = -\frac{c_l}{c_t} \mathbf{l} + \frac{c_l}{c_t} \cos \theta_l \mathbf{n} - \cos \theta_t \mathbf{n}$$



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

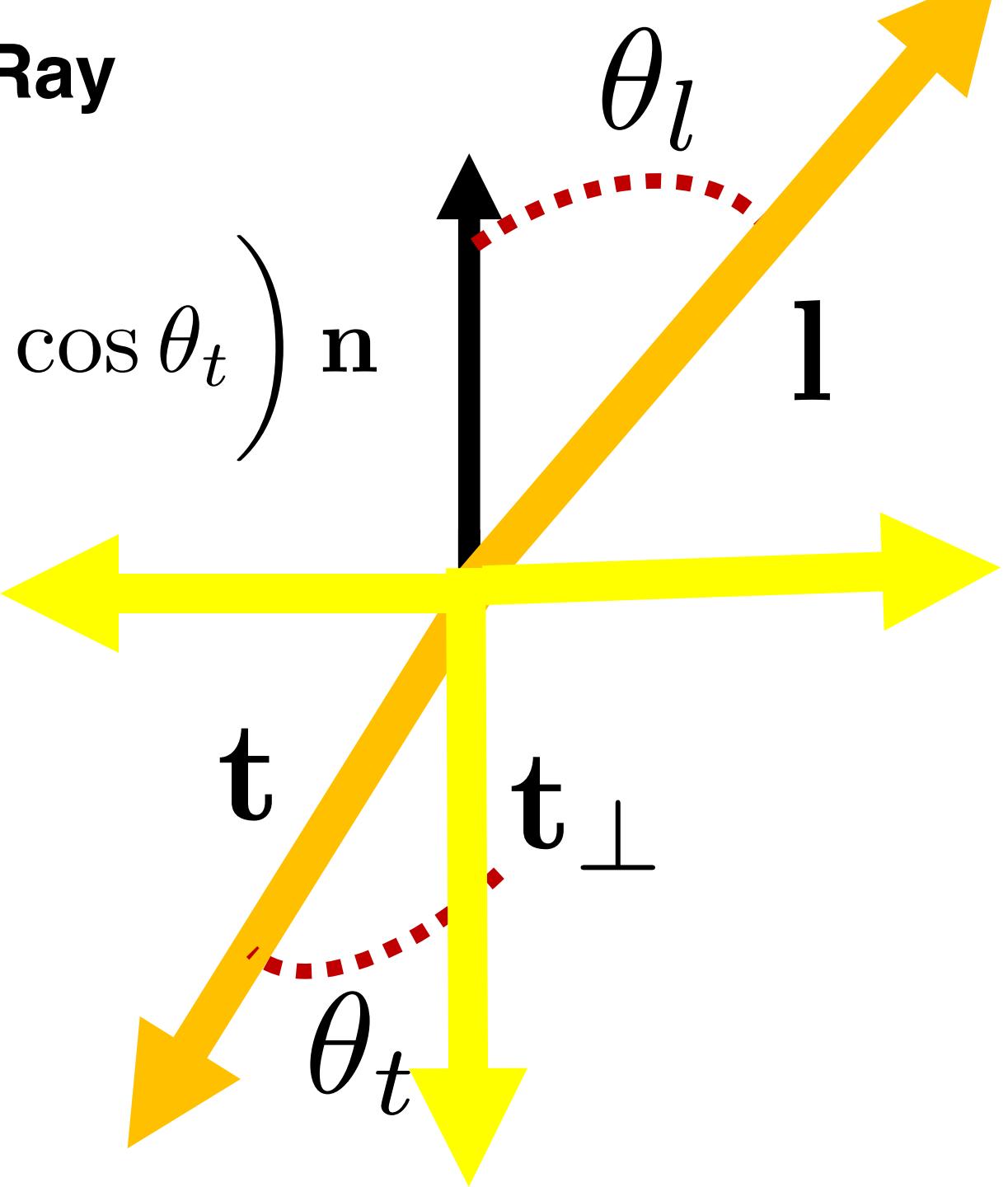
$$\mathbf{t}_\perp = -\sqrt{\cos^2 \theta_t} \mathbf{n}$$

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



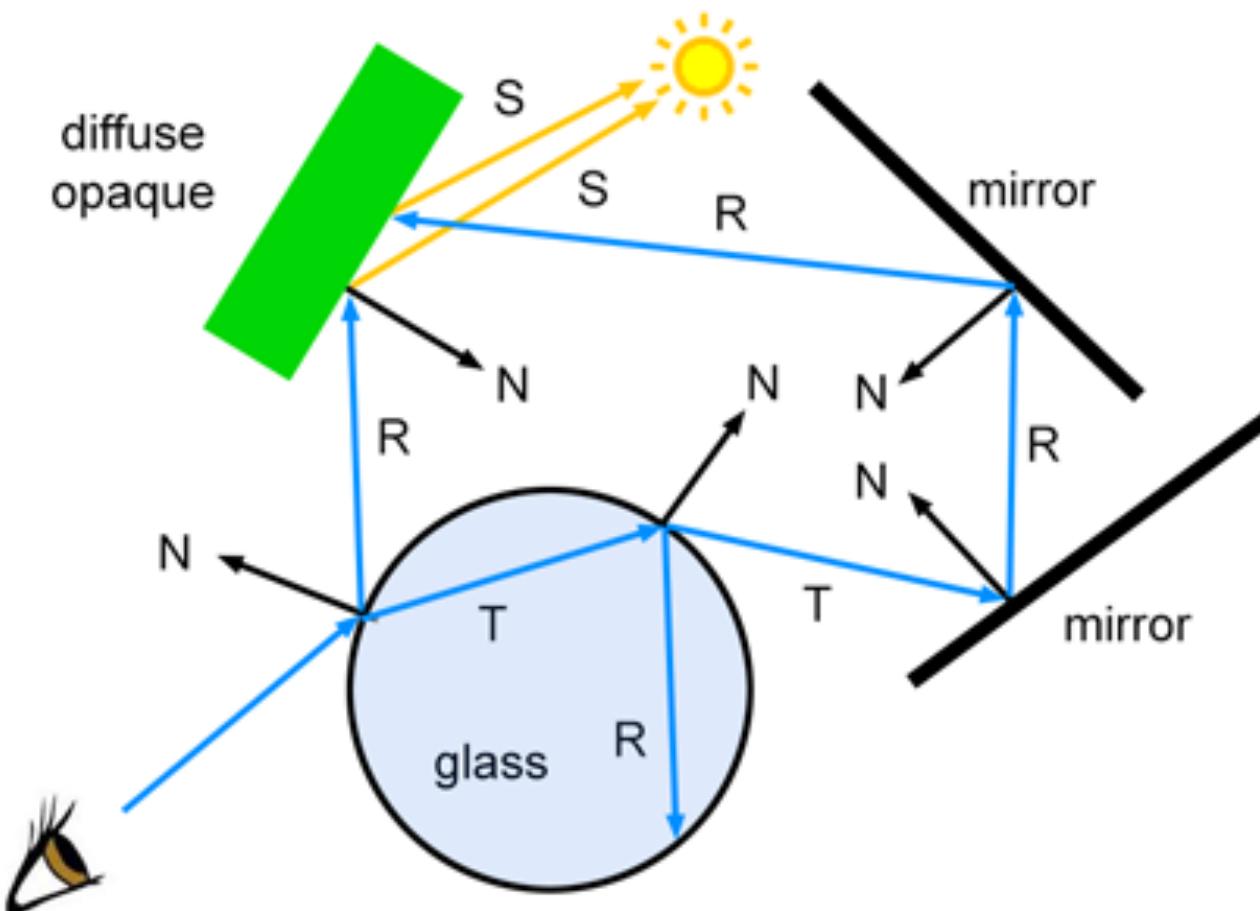
Computing the Refracted Ray

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



```
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



© www.scratchapixel.com

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

Done for Today

Office hours are now BA5268

Assignment 2 due this Friday

Assignment 3 available now