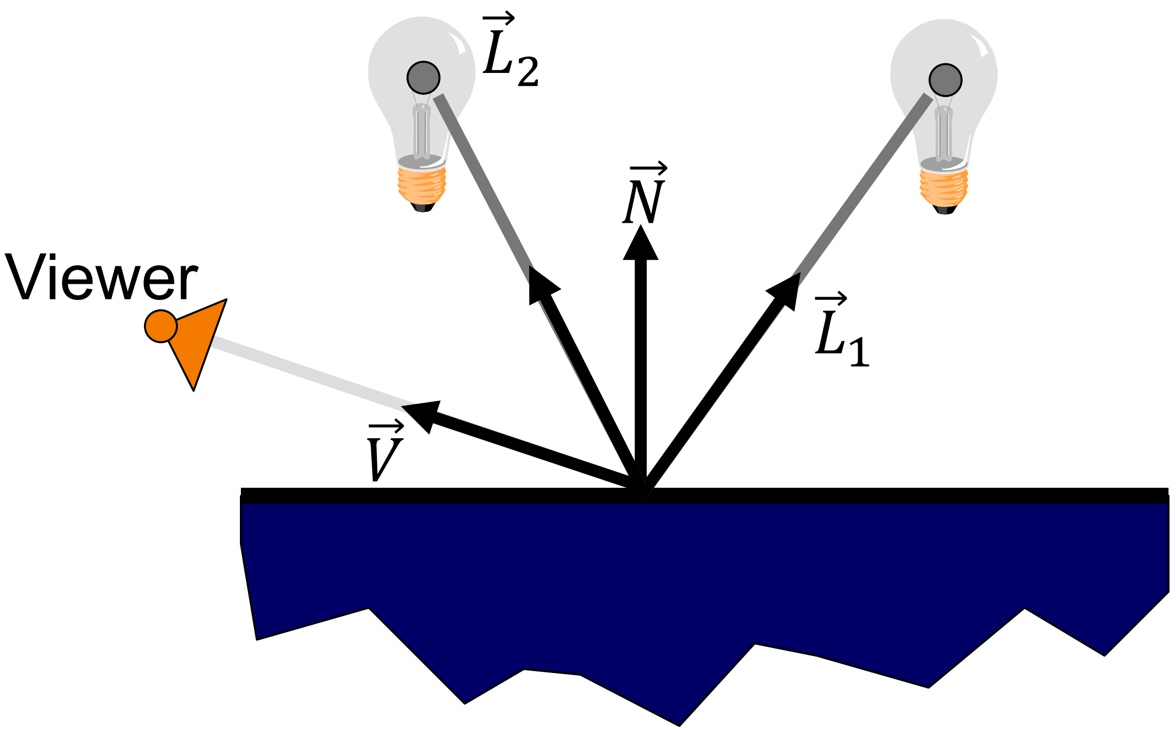
Indirect Illumination

Misha Kazhdan

(601.457/657)

Surface Illumination Calculation

Multiple light source:



$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \right]$$

Overview

Direct Illumination

- Emission at light sources
- Direct reflection

Global illumination

- Shadows
- Inter-object reflections
- Transmissions

Shadows

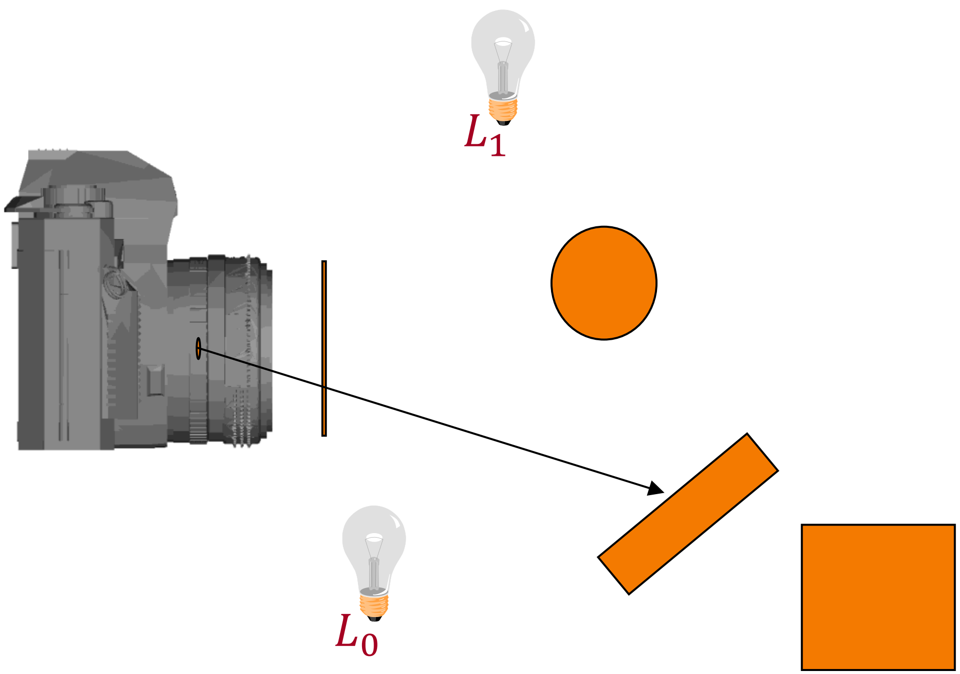
How do we tell if the point of ray/surface intersection is in shadow?

- Cast ray towards each light source L
- If blocked, do not consider the light's contribution.

Shadows

Shadow term tells if light sources are blocked

- Cast ray towards each light source L
- $S_L = 0$ if ray is blocked, $S_L = 1$ otherwise



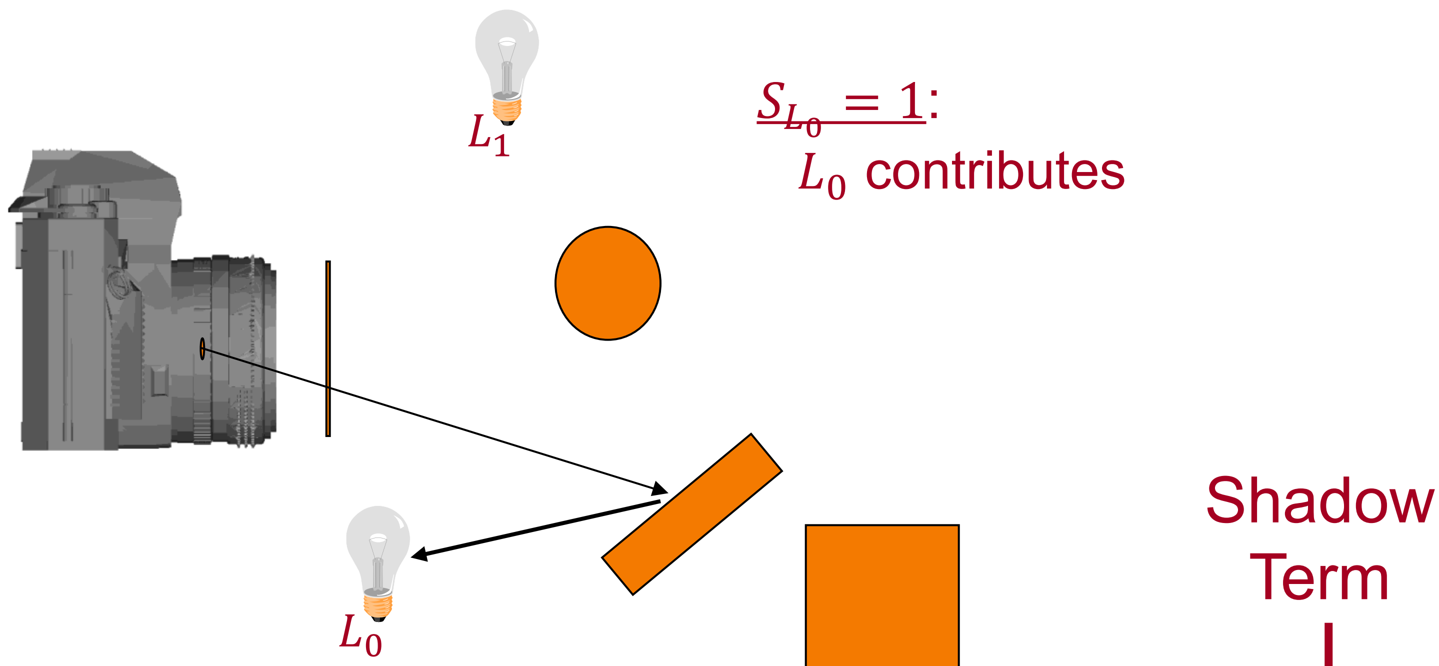
Shadow Term

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right]$$

Shadows

Shadow term tells if light sources are blocked

- Cast ray towards each light source L
- $S_L = 0$ if ray is blocked, $S_L = 1$ otherwise

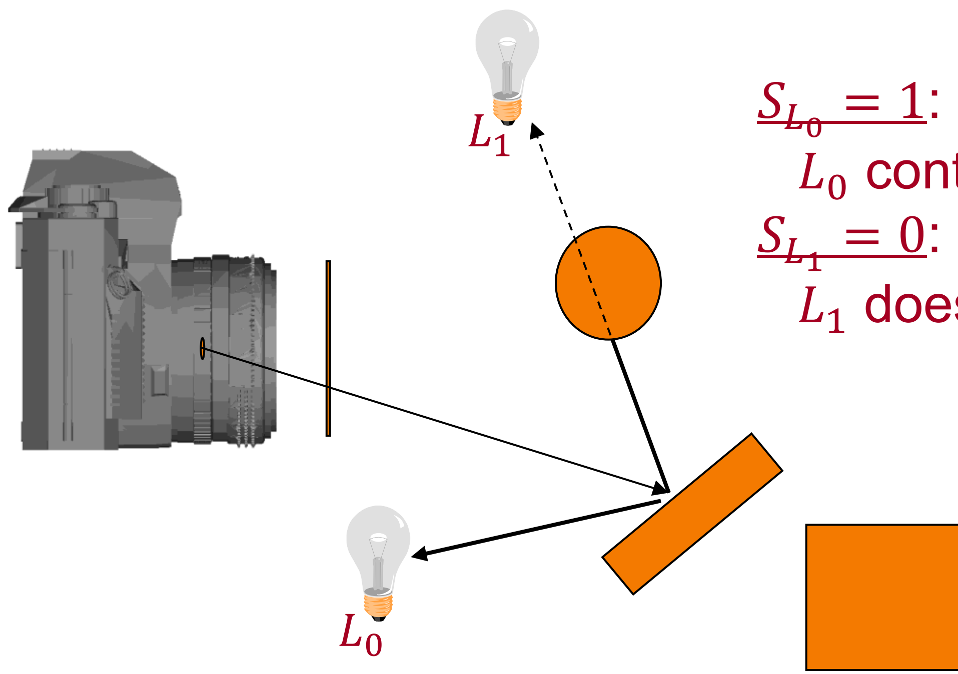


$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right]$$

Shadows

Shadow term tells if light sources are blocked

- Cast ray towards each light source L
- $S_L = 0$ if ray is blocked, $S_L = 1$ otherwise



$S_{L_0} = 1$:
L_0 contributes

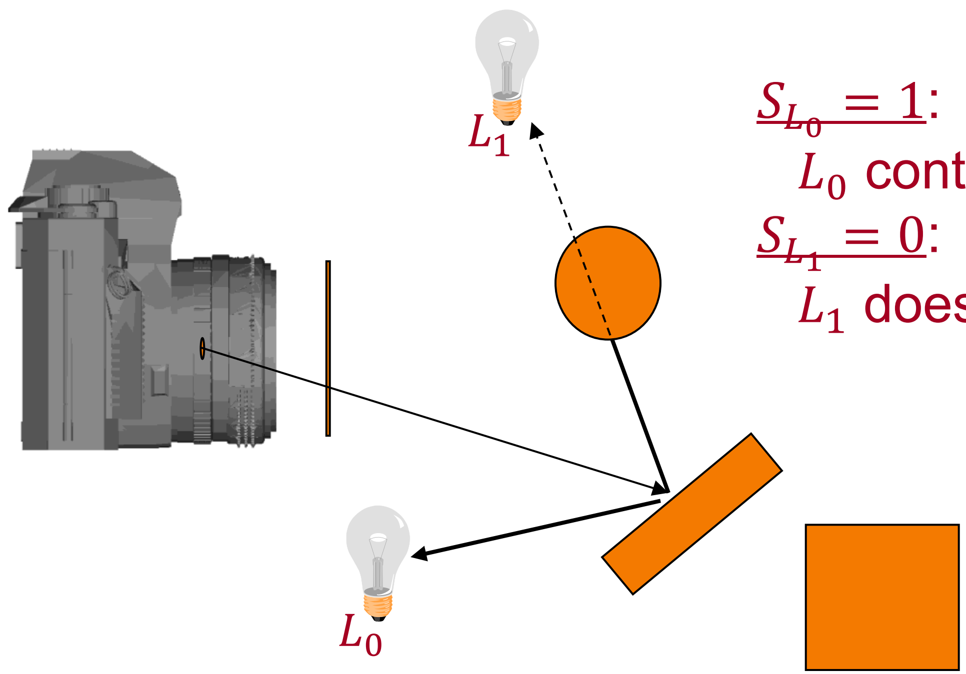
$S_{L_1} = 0$:
L_1 doesn't contribute

Shadow Term

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right]$$

Note:
For directional lights, check for intersections anywhere along the ray.
For point/spot lights, only check up to the source of the light.

The shadow term only affects the direct (i.e. diffuse and specular) contributions from the light source.



$S_{L_0} = 1$:
L_0 contributes
$S_{L_1} = 0$:
L_1 doesn't contribute

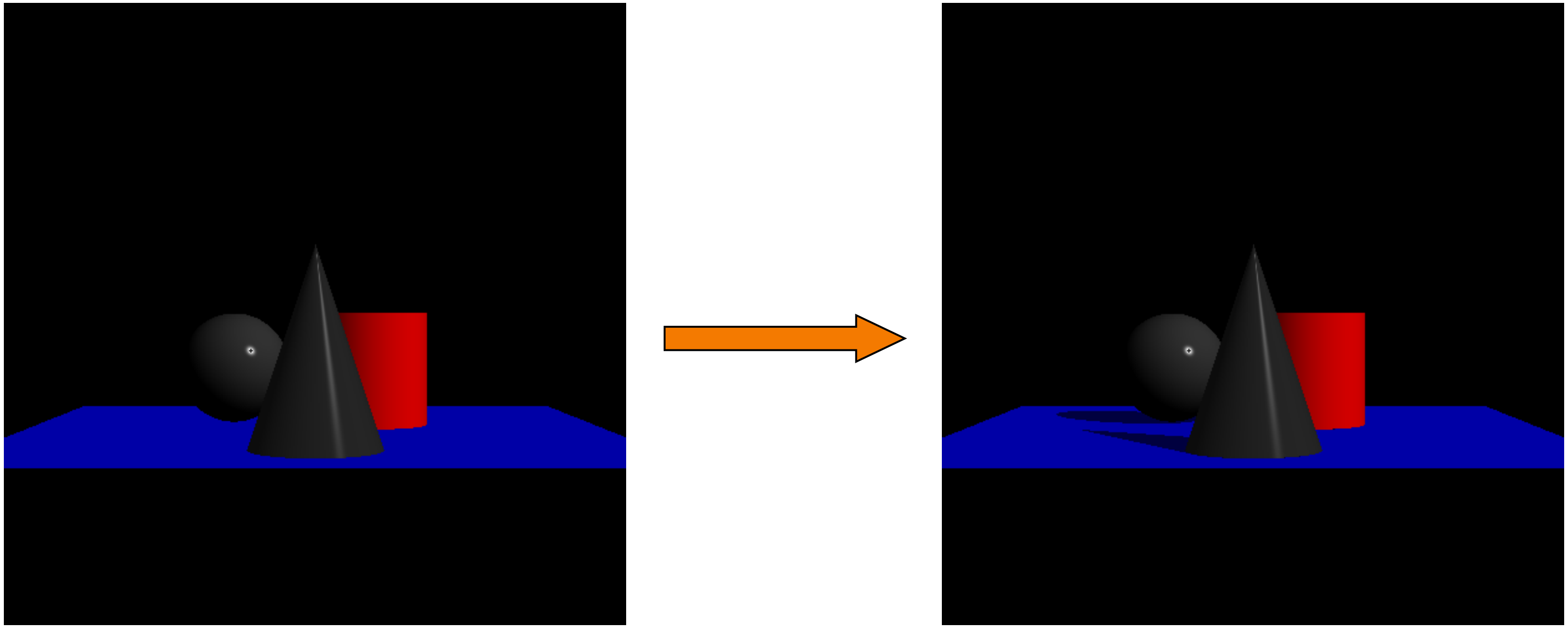
Shadow Term

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right]$$

Ray Casting

Trace rays from camera to first point of contact with the geometry, and from the first point of contact to the light source(s)

- Direct illumination from unblocked lights only



Recursive Ray Tracing

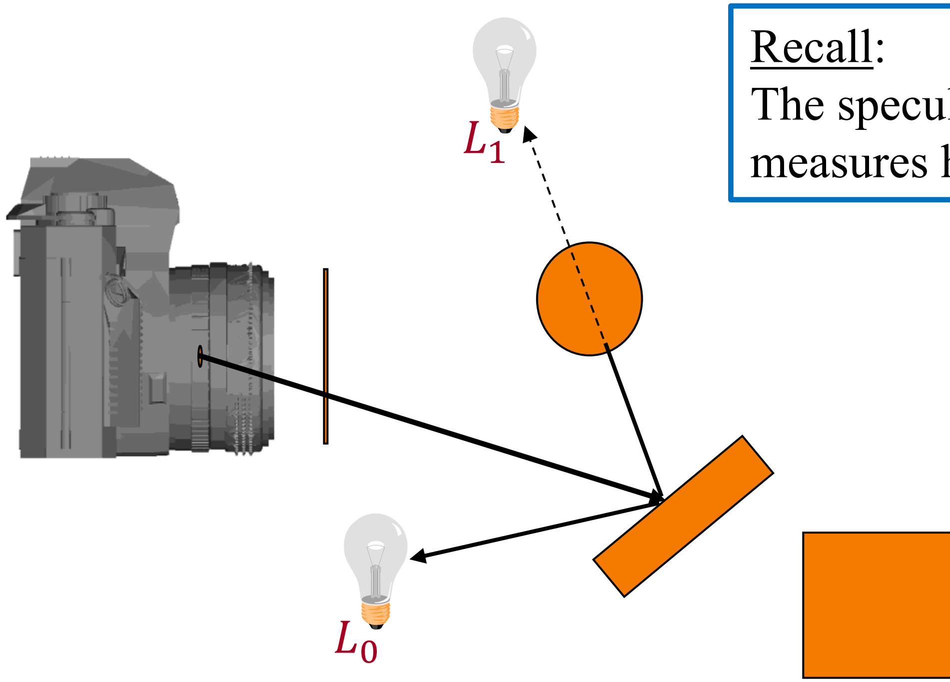
Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays

Mirror Reflections

Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays



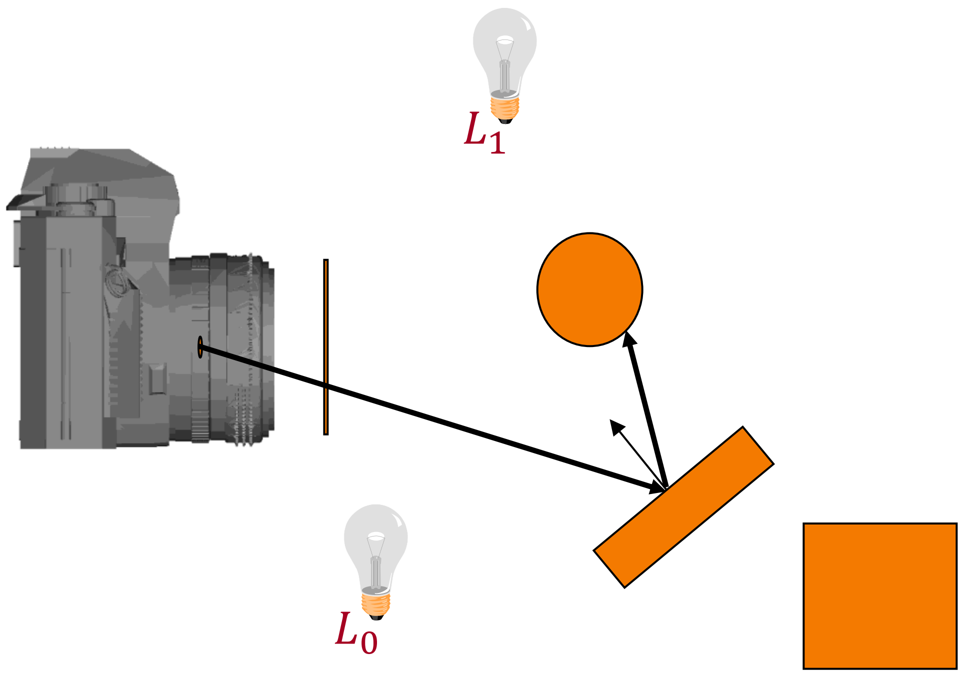
Recall:
The specularity of a surface, K_S, measures how mirror-like it is.

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right]$$

Mirror Reflections

Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays



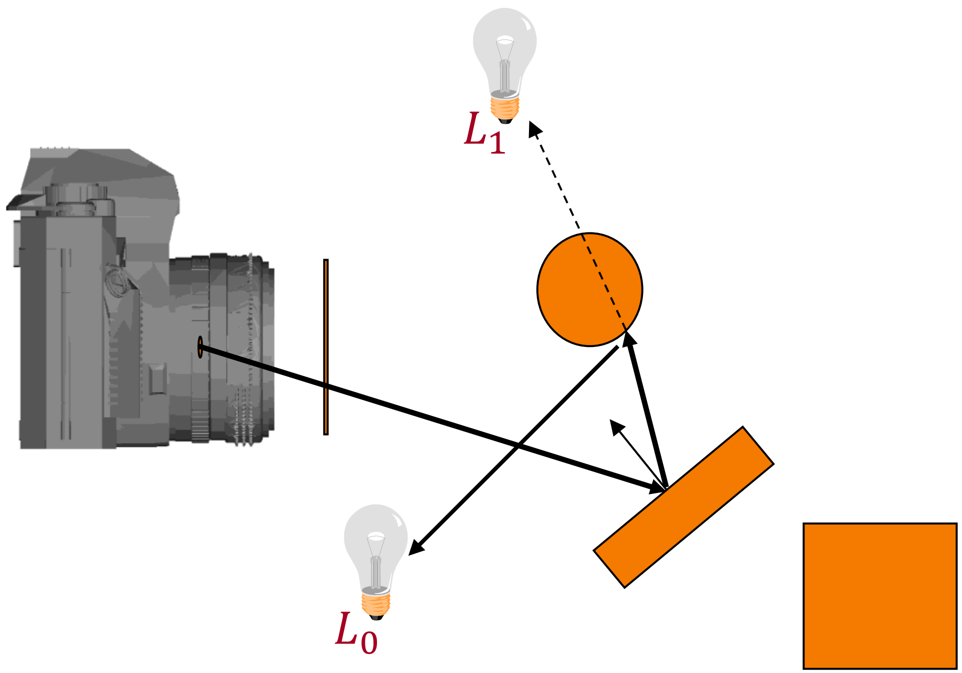
Specular contribution

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right] + K_s \cdot I_R$$

Mirror Reflections

Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays



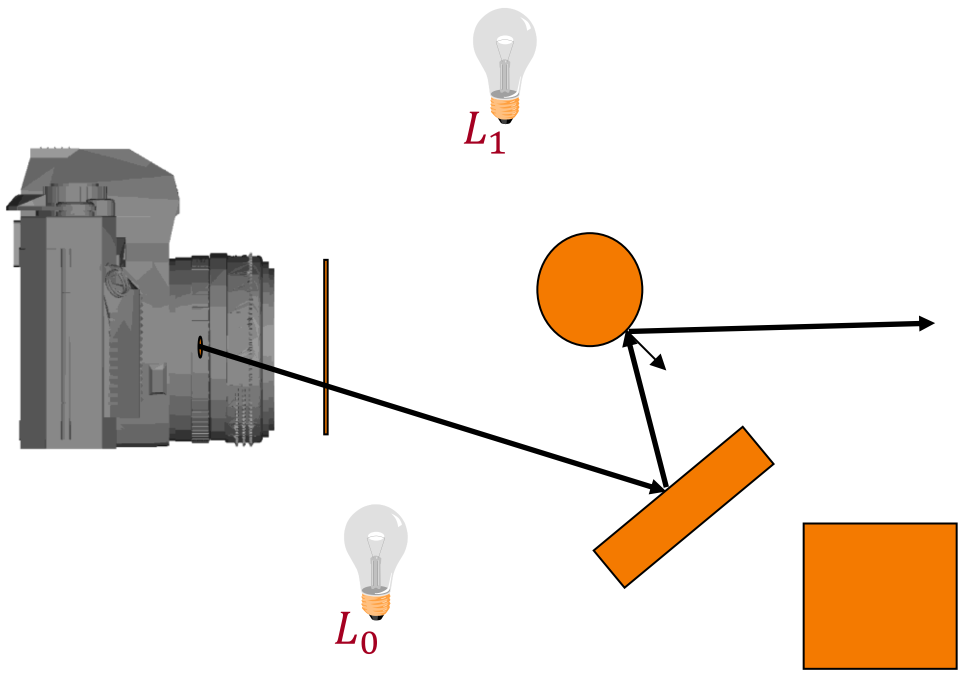
Specular contribution

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right] + K_s \cdot I_R$$

Mirror Reflections

Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays



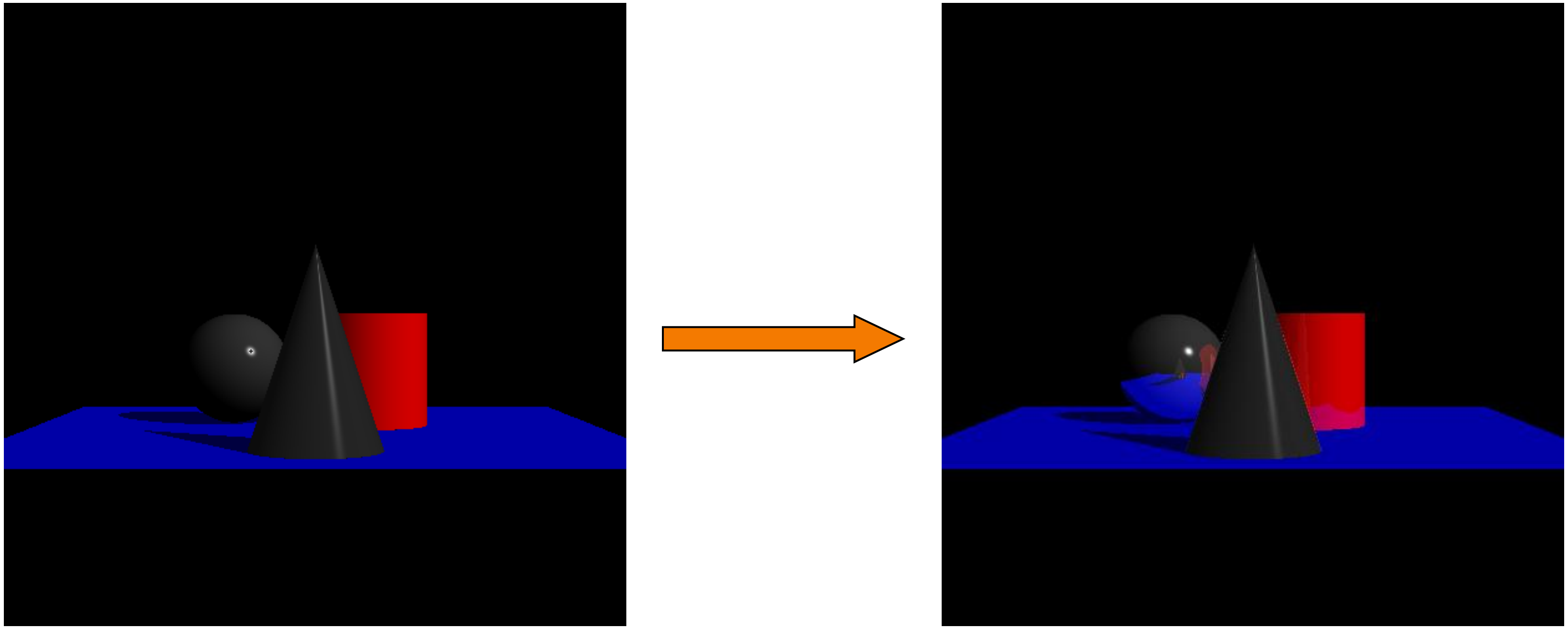
Specular contribution

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_s \cdot I_R$$

Mirror Reflections

Also trace secondary rays from hit surfaces

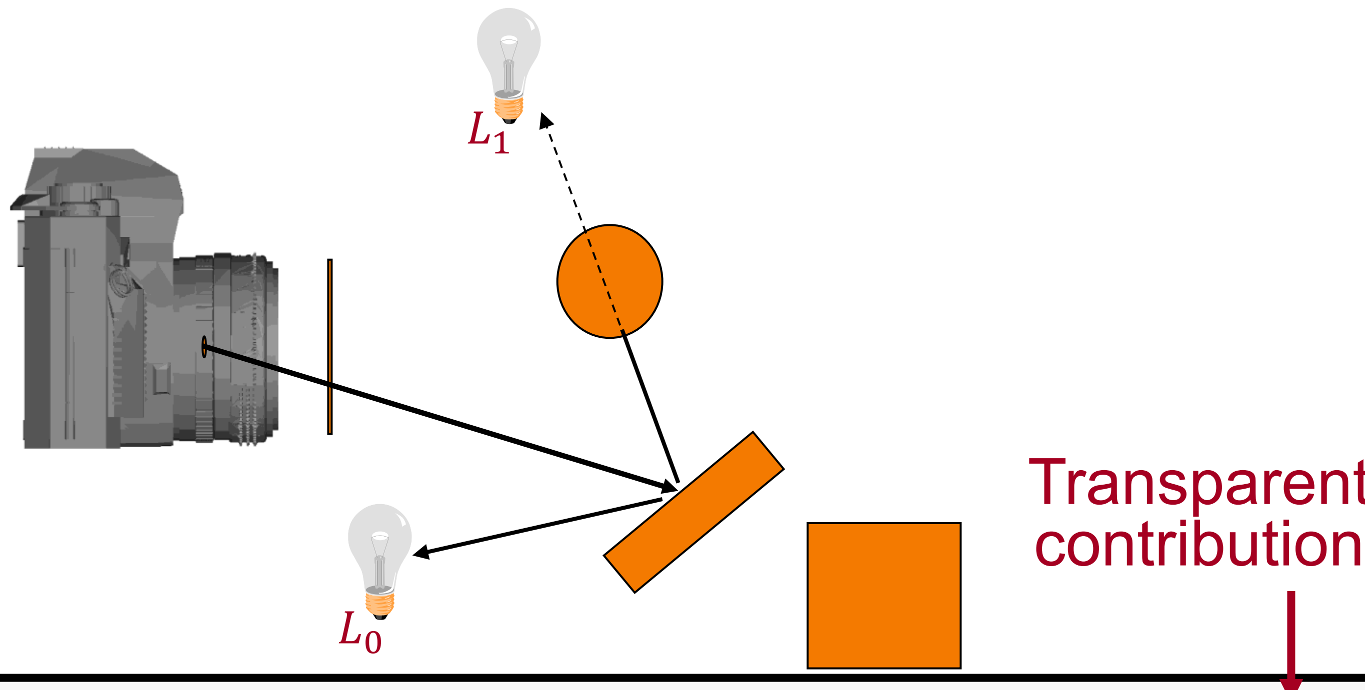
1. Reflected Rays
2. Refracted Rays



Transparent Refraction

Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays

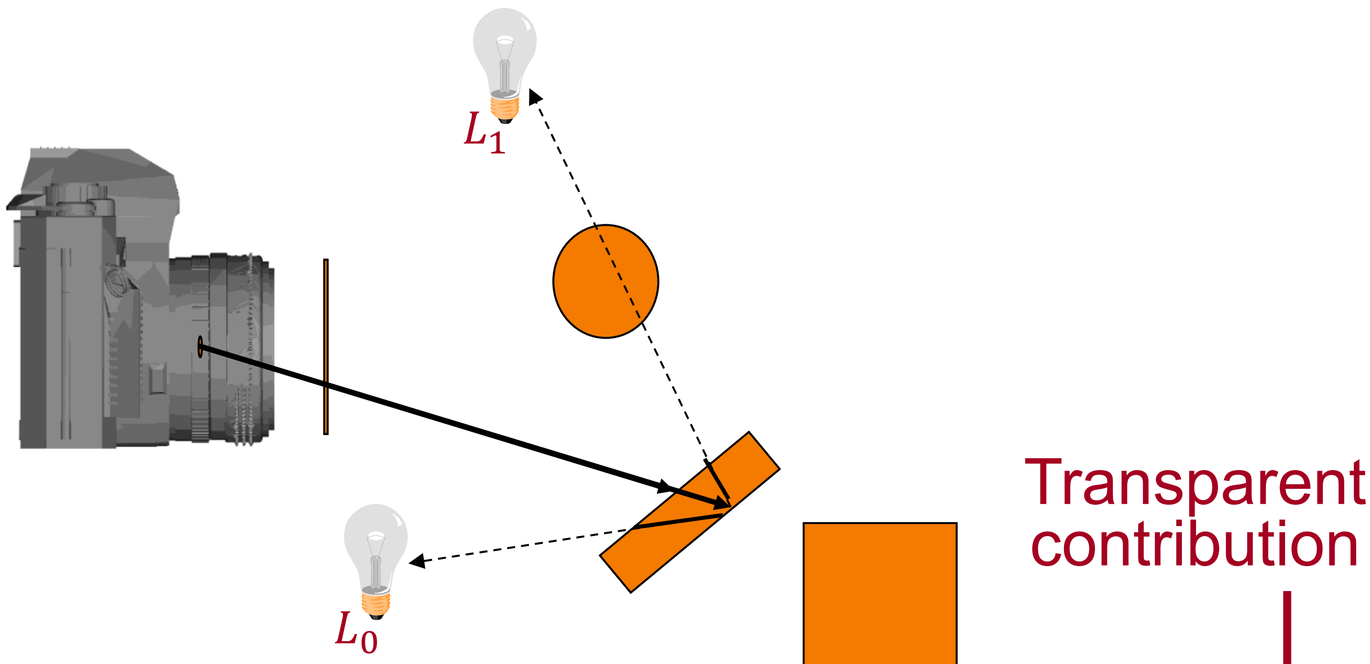


$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right] + K_S \cdot I_R + K_T \cdot I_T$$

Transparent Refraction

Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays

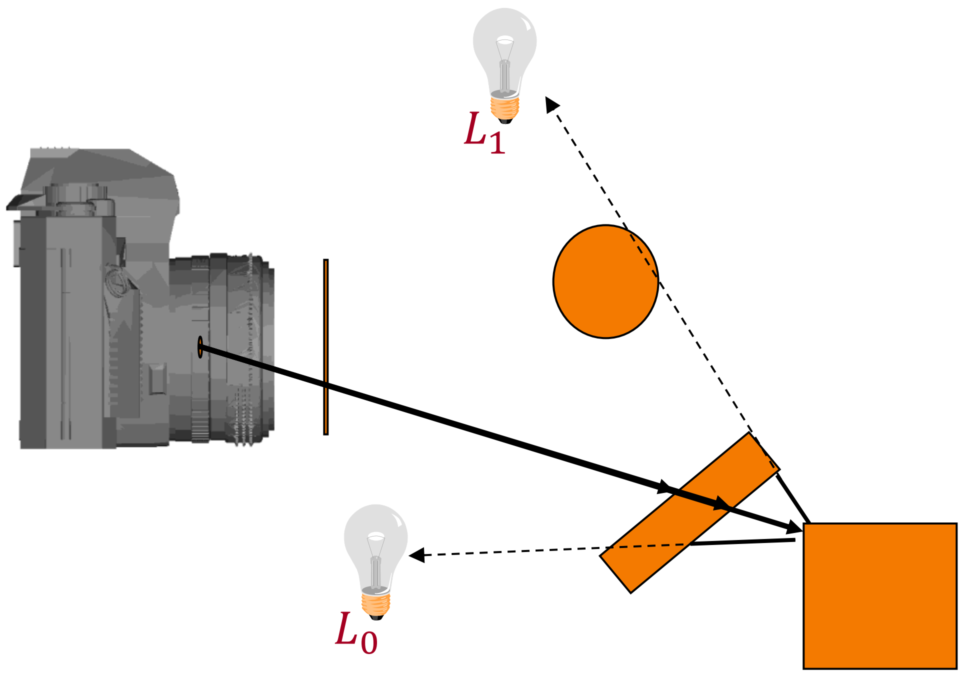


$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_S \cdot I_R + K_T \cdot I_T$$

Transparent Refraction

Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays



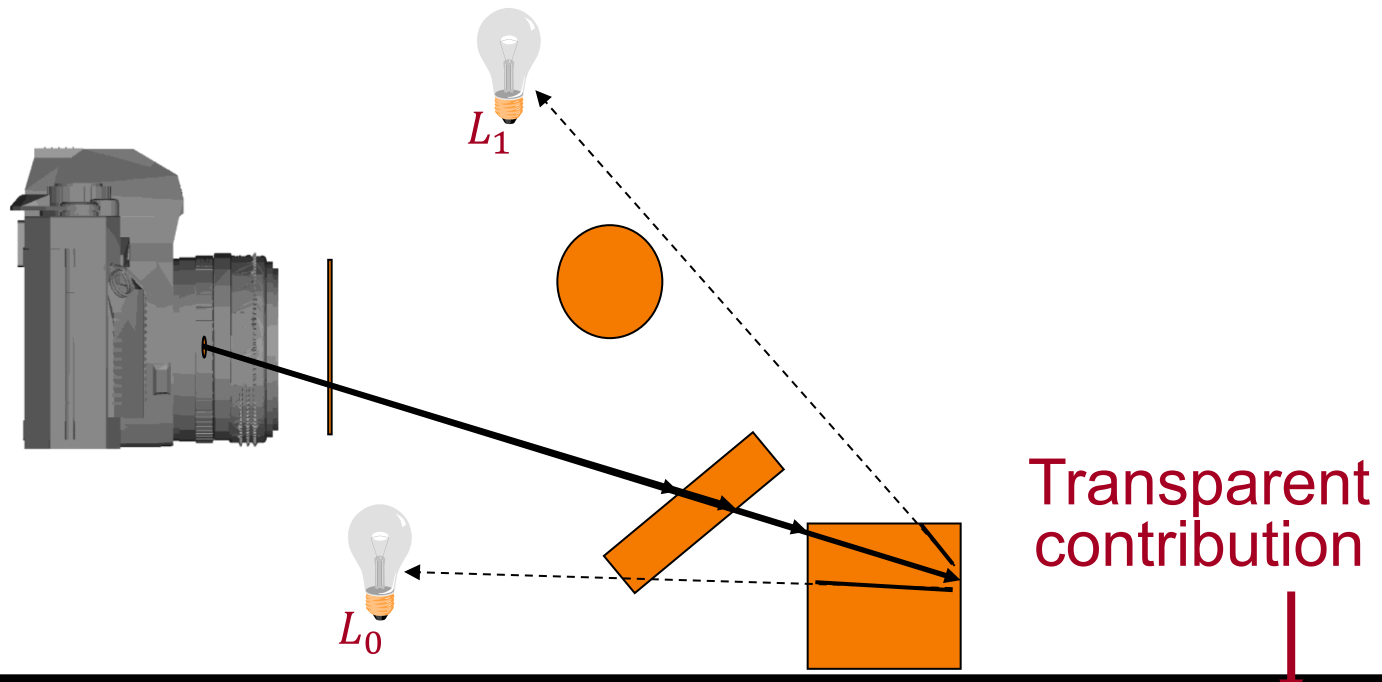
Transparent contribution

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_S \cdot I_R + K_T \cdot I_T$$

Transparent Refraction

Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays

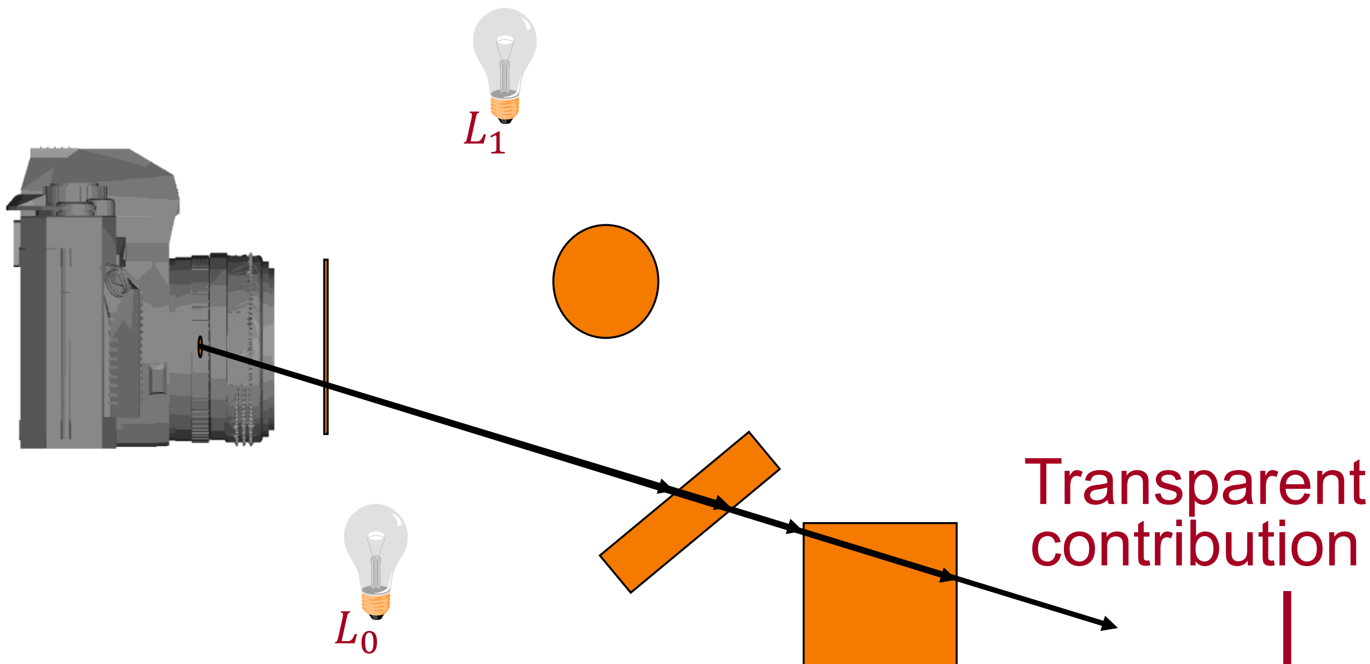


$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_S \cdot I_R + K_T \cdot I_T$$

Transparent Refraction

Also trace secondary rays from hit surfaces

1. Reflected Rays
2. Refracted Rays

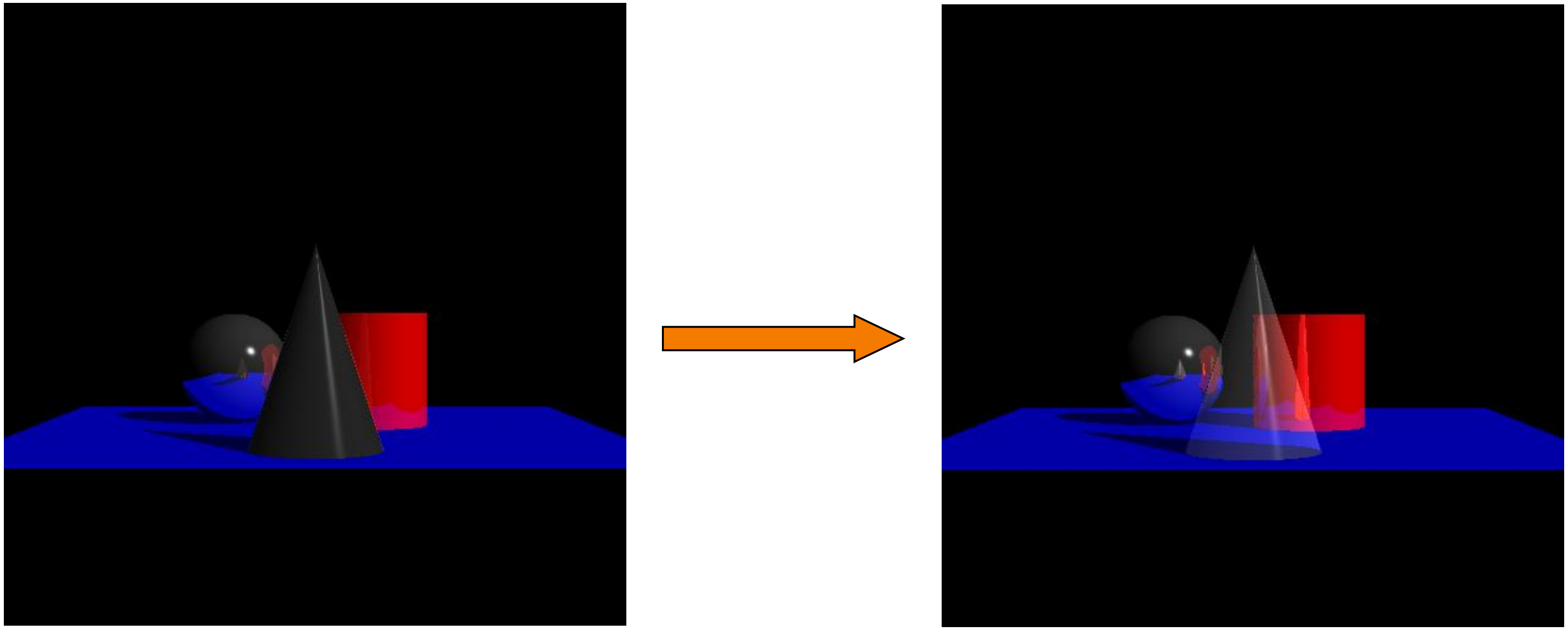


$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_S \cdot I_R + K_T \cdot I_T$$

Transparent Refraction

Also trace secondary rays from hit surfaces

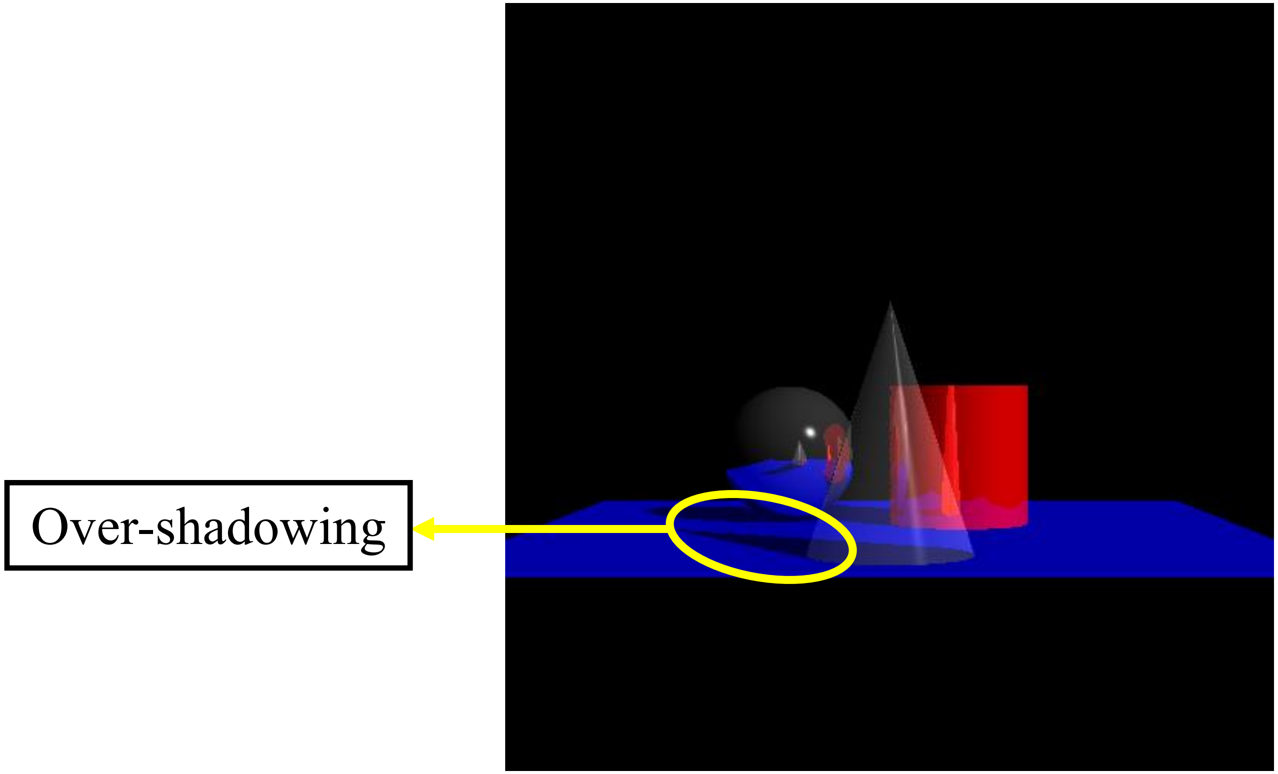
1. Reflected Rays
2. Refracted Rays



Transparency and Shadow

Problem:

- If a surface is partially transparent, then rays to the light source will partially pass through the object



Transparency and Shadow

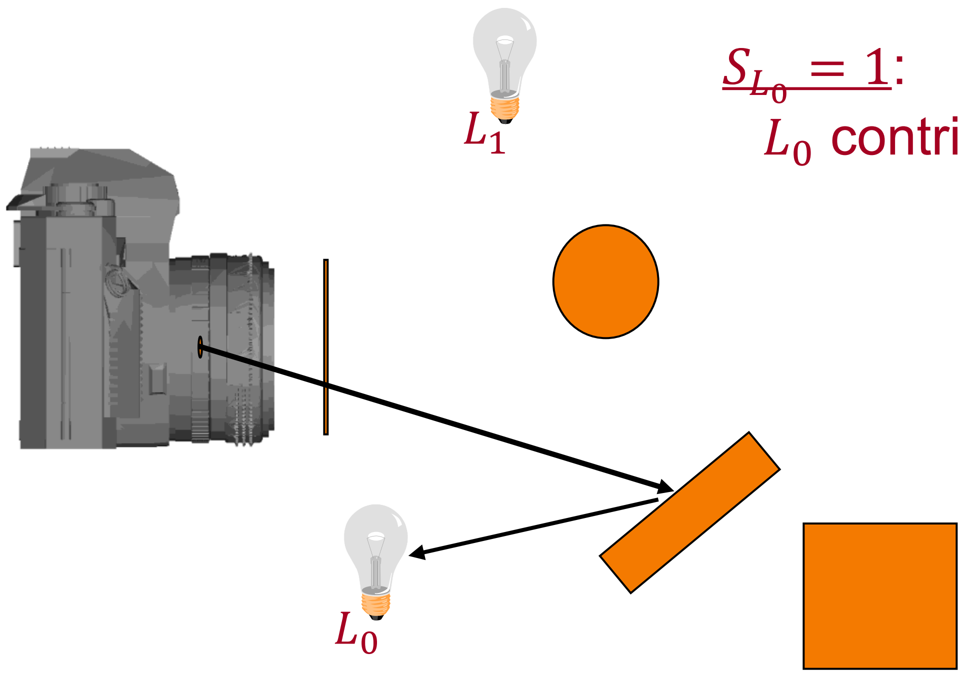
Problem:

- ◦ If a surface is partially transparent, then rays to the light source will partially pass through the object
- ⇒ Modify the shadow term to give the **fraction** of light passing through…
- … by "accumulating" transparency values as the ray travels to the light source.

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right] + K_s \cdot I_R + K_T \cdot I_T$$

Transparency and Shadow

Start with $S_L = 1$ and "accumulate" transparency values as the ray travels to the light source.

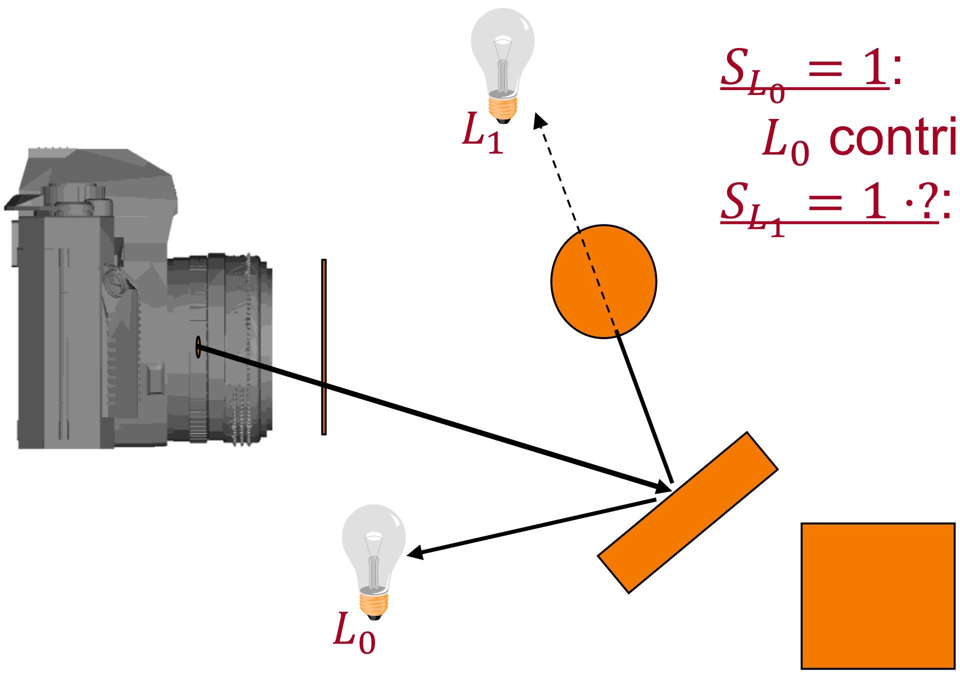


$S_{L_0} = 1$:
L_0 contributes fully

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_s \cdot I_R + K_T \cdot I_T$$

Transparency and Shadow

Start with $S_L = 1$ and "accumulate" transparency values as the ray travels to the light source.

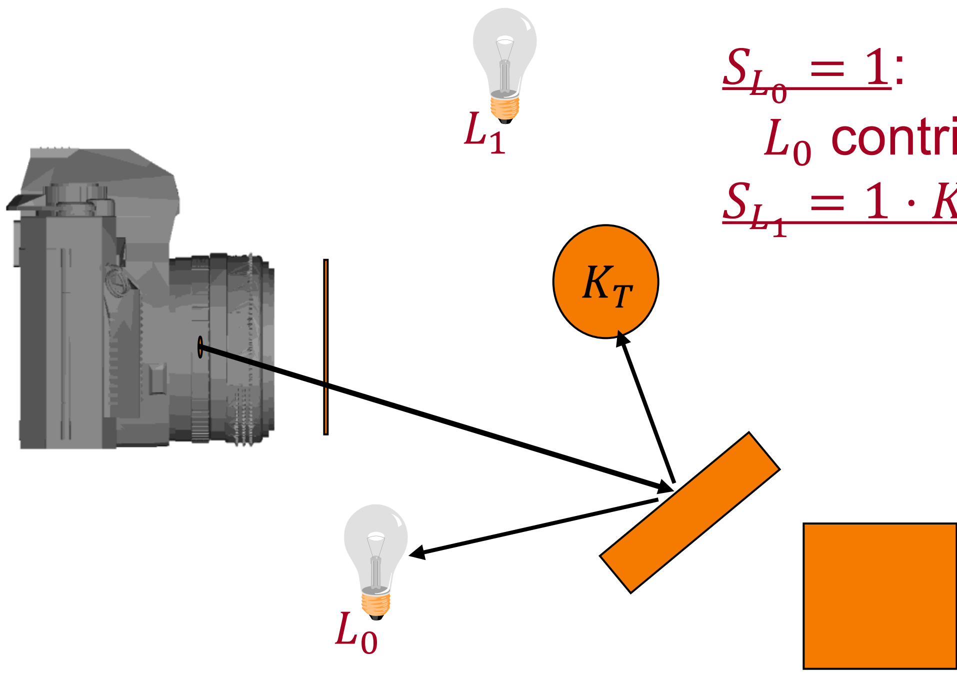


$\underline{S_{L_0} = 1}$:
L_0 contributes fully
$\underline{S_{L_1} = 1 \cdot ?}$:

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_s \cdot I_R + K_T \cdot I_T$$

Transparency and Shadow

Start with $S_L = 1$ and “accumulate” transparency values as the ray travels to the light source.

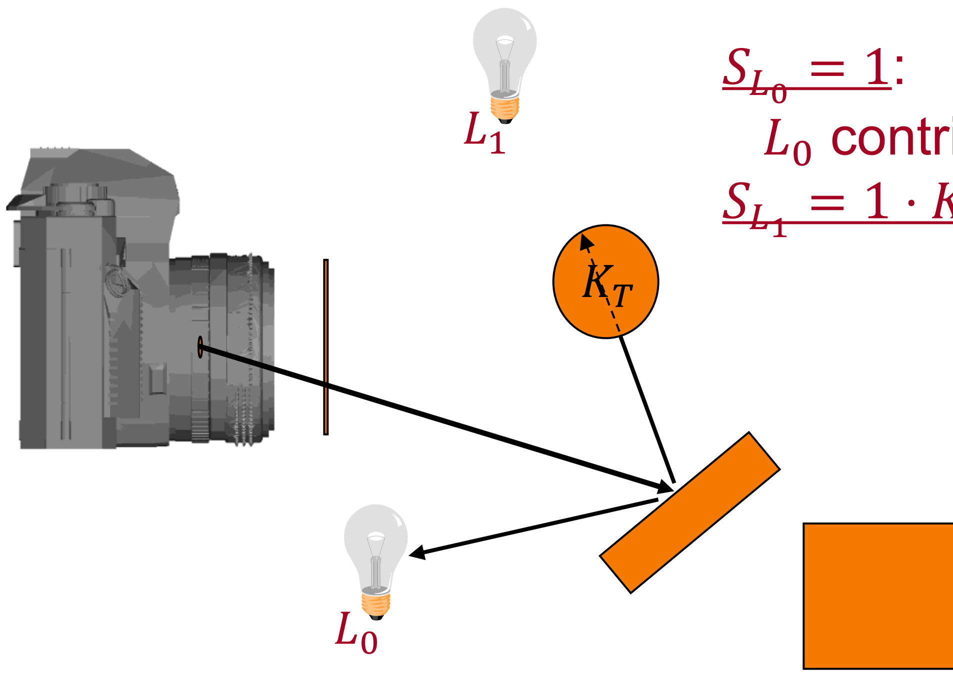


$S_{L_0} = 1$:
L_0 contributes fully
$S_{L_1} = 1 \cdot K_T \cdot ?$:

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right] + K_s \cdot I_R + K_T \cdot I_T$$

Transparency and Shadow

Start with $S_L = 1$ and "accumulate" transparency values as the ray travels to the light source.

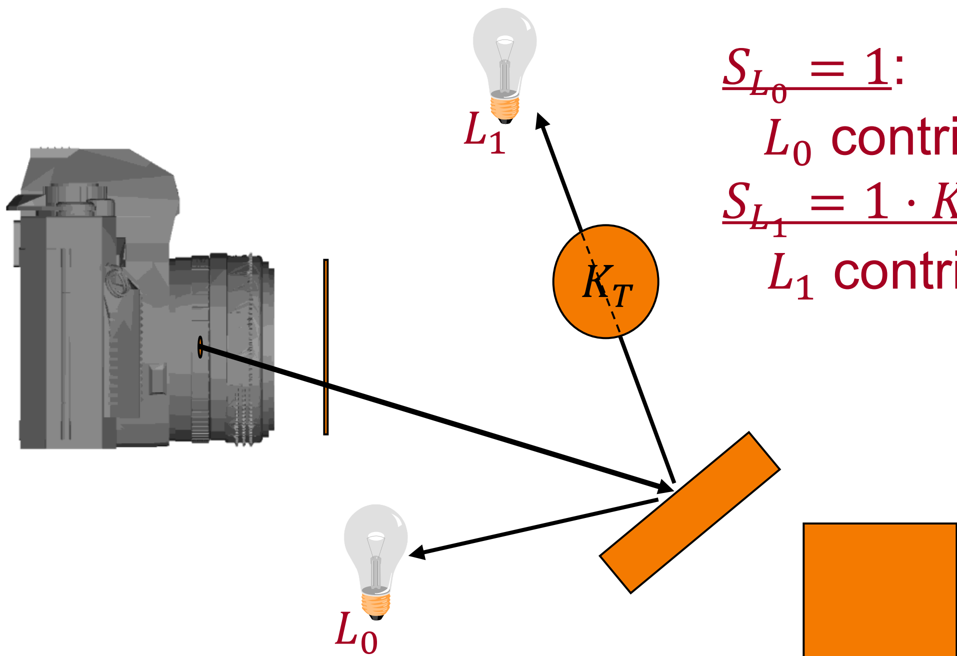


$S_{L_0} = 1$:
L_0 contributes fully
$S_{L_1} = 1 \cdot K_T \cdot K_T \cdot ?$:

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n} \right) \cdot I_L \cdot S_L \right] + K_S \cdot I_R + K_T \cdot I_T$$

Transparency and Shadow

Start with $S_L = 1$ and "accumulate" transparency values as the ray travels to the light source.

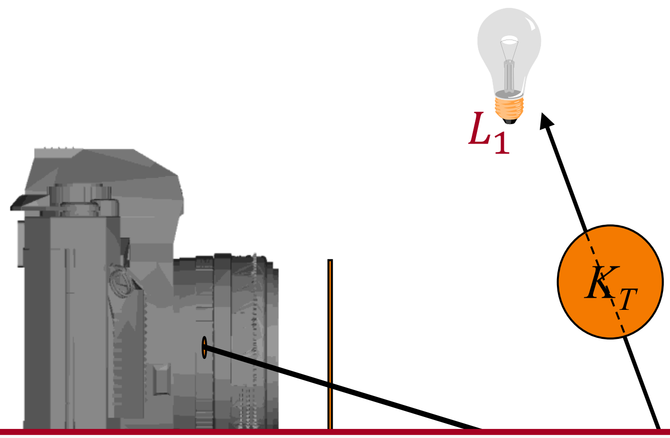


$S_{L_0} = 1$:
L_0 contributes fully
$S_{L_1} = 1 \cdot K_T \cdot K_T$:
L_1 contributes partially

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_S \cdot I_R + K_T \cdot I_T$$

Transparency and Shadow

Start with $S_L = 1$ and "accumulate" transparency values as the ray travels to the light source.



$S_{L_0} = 1$:
L_0 contributes fully
$S_{L_1} = 1 \cdot K_T \cdot K_T$:
L_1 contributes partially

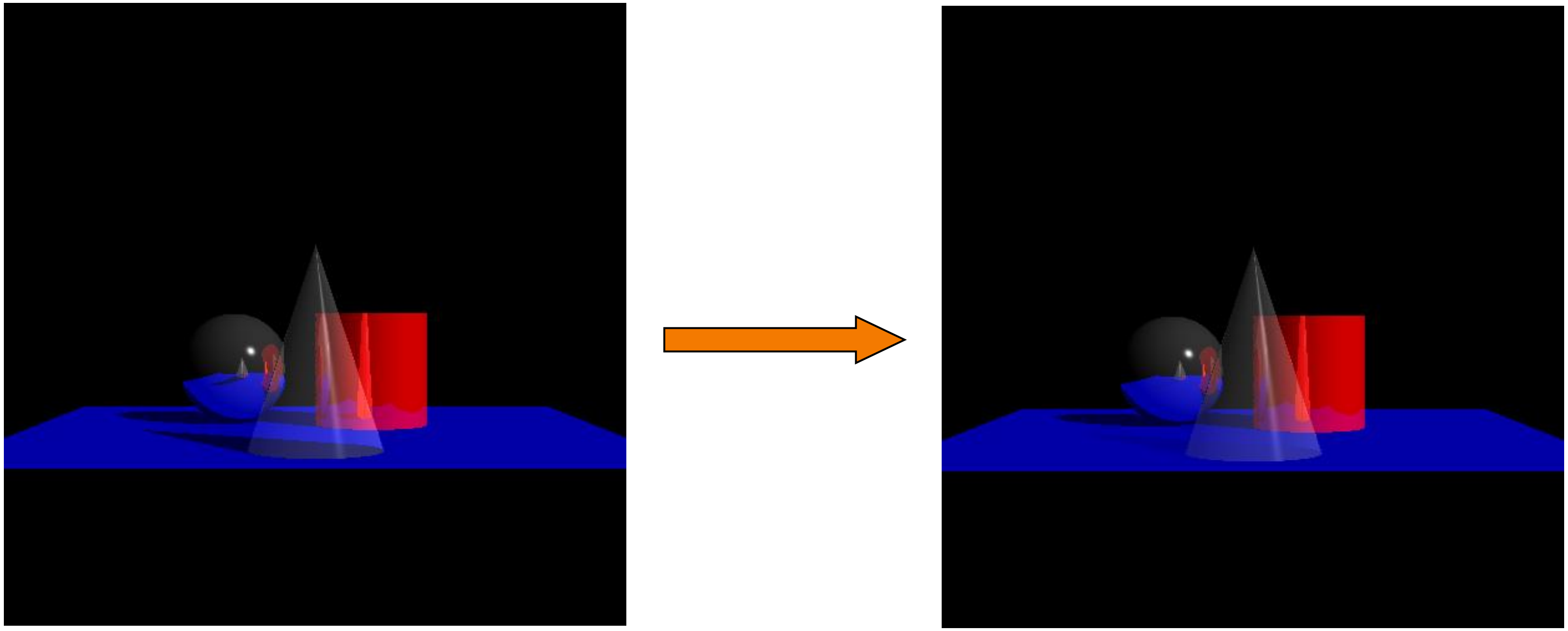
For solid models, opacity should be a function of the distance, d, travelled through the surface:

$$S = e^{-d \cdot K_T}$$

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_s \cdot I_R + K_T \cdot I_T$$

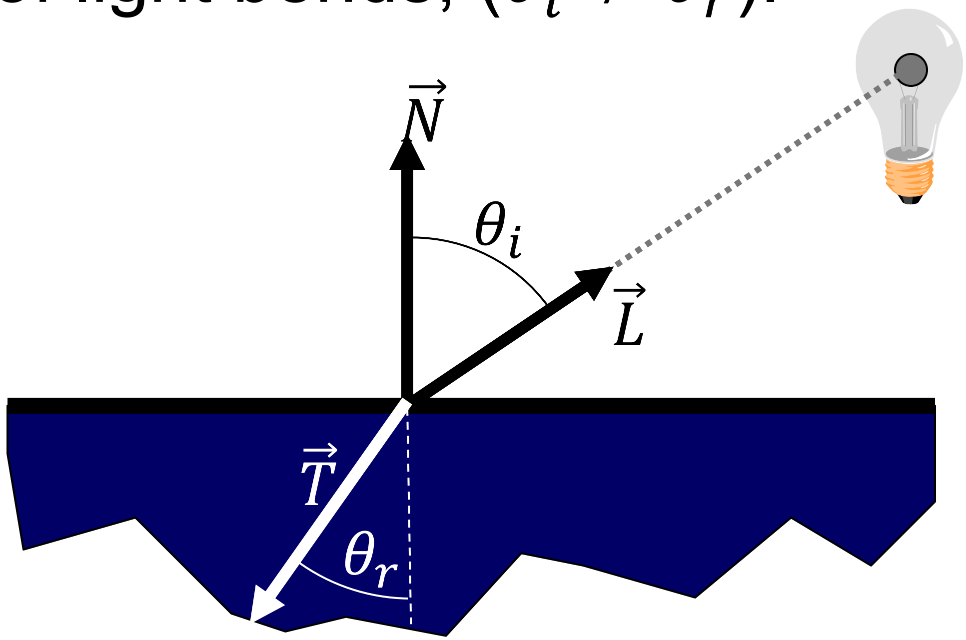
Transparency and Shadow

Start with $S_L = 1$ and "accumulate" transparency values as the ray travels to the light source.



Transparent Refraction

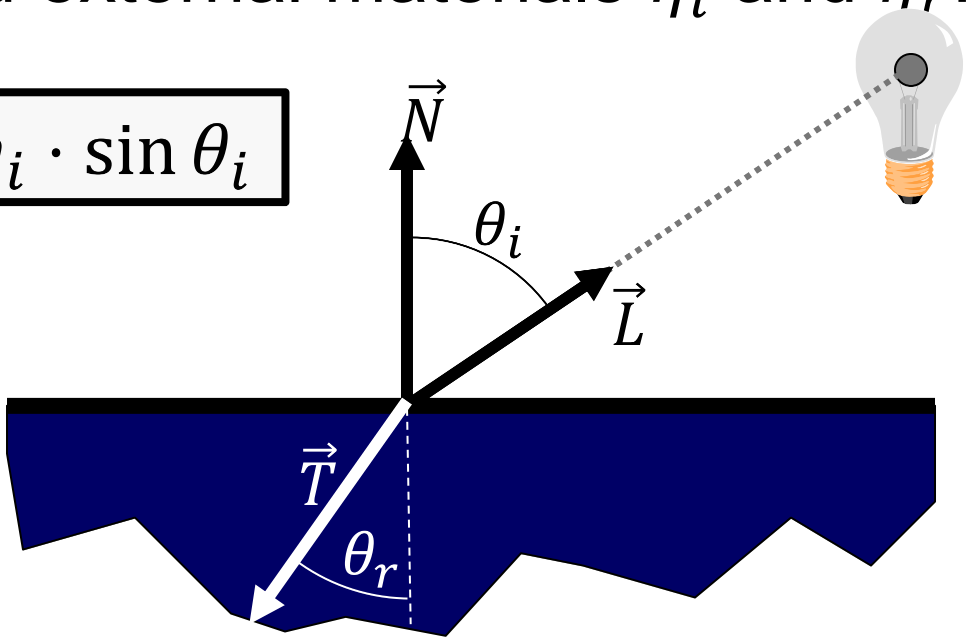
When a light of light passes through a transparent object, the ray of light bends, ($\theta_i \neq \theta_r$).



Snell's Law

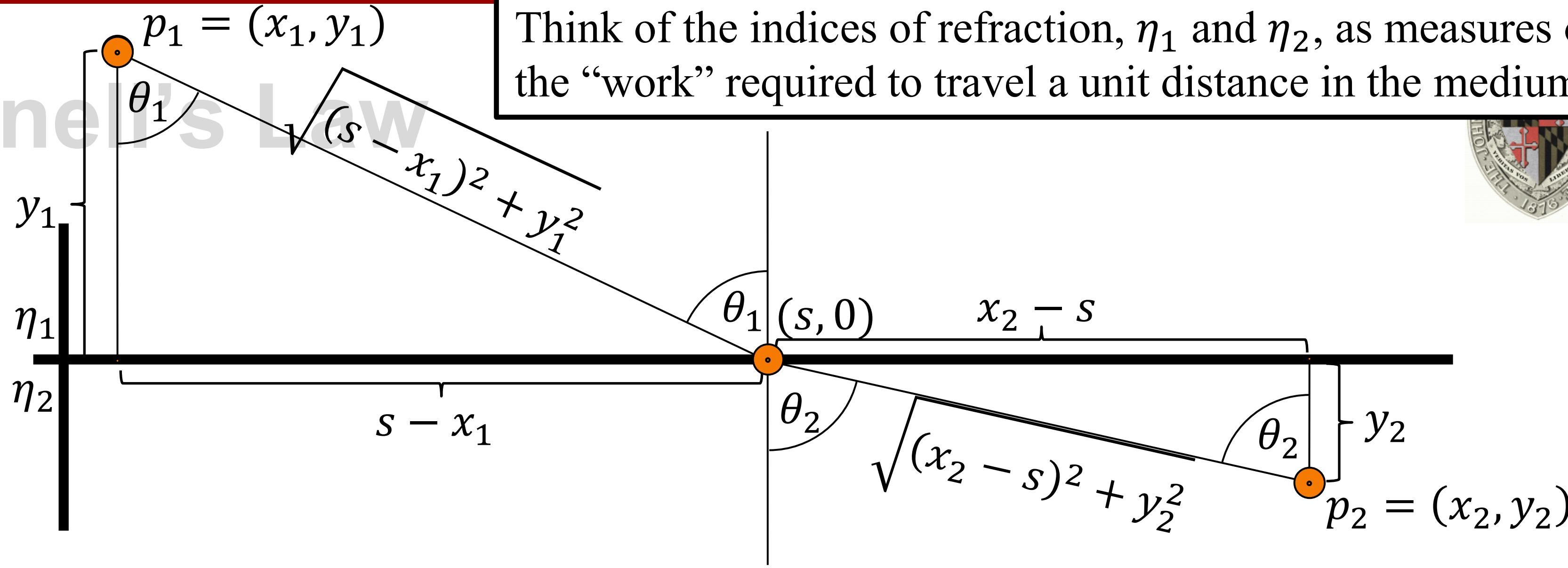
Bending is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\boxed{\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i}$$



Snell's Law

Think of the indices of refraction, η_1 and η_2, as measures of the "work" required to travel a unit distance in the medium.



Goal:

Find the value of s minimizing the "work" of travel from p_1 to p_2

$$W(s) = \sqrt{(s - x_1)^2 + y_1^2} \cdot \eta_1 + \sqrt{(x_2 - s)^2 + y_2^2} \cdot \eta_2$$

Taking the derivative and setting to zero:

$$\Rightarrow 0 = W'(s) = \frac{1}{2}\frac{2(s - x_1)}{\sqrt{(s - x_1)^2 + y_1^2}} \cdot \eta_1 + \frac{1}{2}\frac{-2(x_2 - s)}{\sqrt{(x_2 - s)^2 + y_2^2}} \cdot \eta_2$$

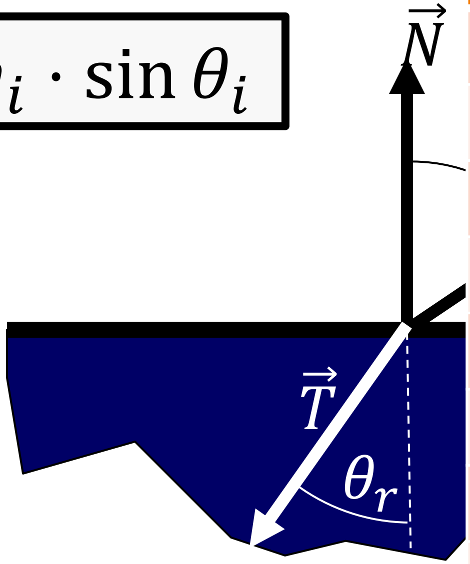
$$\Leftrightarrow \frac{(s - x_1)}{\sqrt{(s - x_1)^2 + y_1^2}} \cdot \eta_1 = \frac{(x_2 - s)}{\sqrt{(x_2 - s)^2 + y_2^2}} \cdot \eta_2$$

$$\Leftrightarrow \sin(\theta_1) \cdot \eta_1 = \sin(\theta_2) \cdot \eta_2$$

Snell's Law

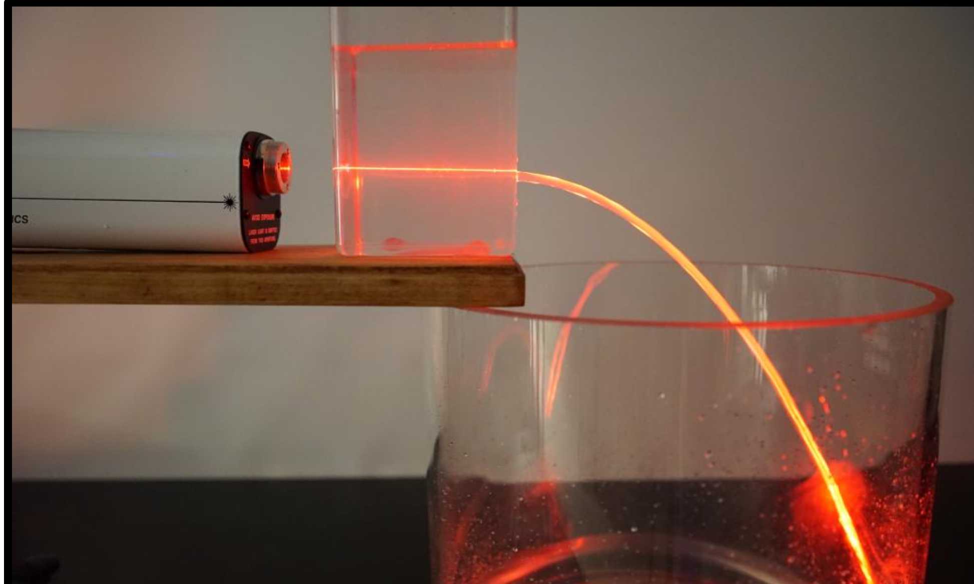
Bending is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\boxed{\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i}$$

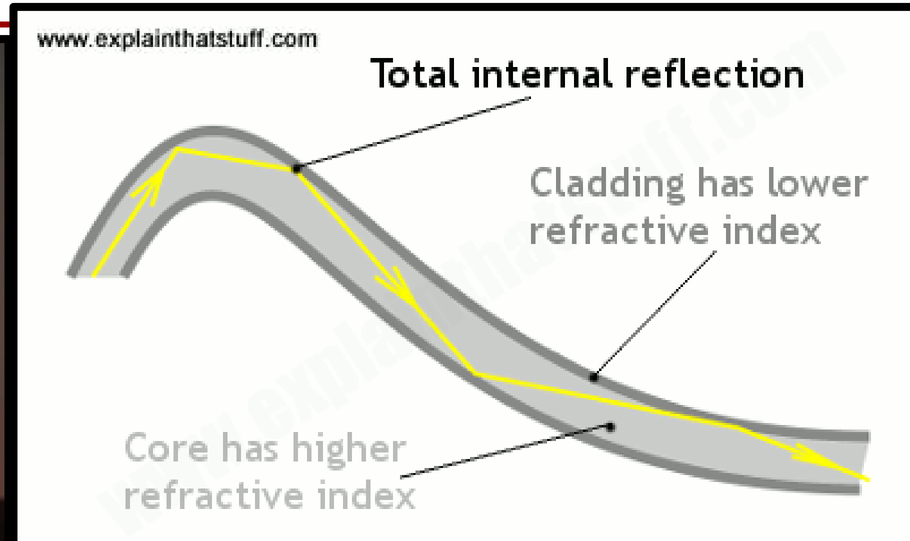


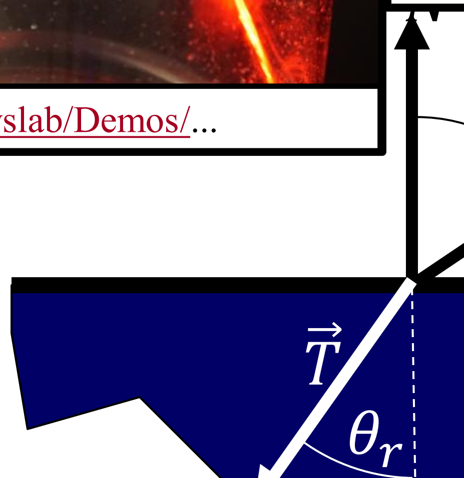
Medium	η
Vacuum	1.00 exactly
Air (actual)	1.0003
Air (accepted)	**1.00**
Water	1.33
Ethyl alcohol	1.36
Oil	1.46
Glass (typical)	1.50
Polystyrene plastic	1.59
Cubic zirconia	2.18
Diamond	2.41
Silicon (infrared)	3.50

https://physics.umd.edu/courses/Phys132



[https://physics.nyu.edu/~physlab/Demos/...](https://physics.nyu.edu/~physlab/Demos/)





Vacuum	1.00 exactly
Air (actual)	1.0003
Air (accepted)	**1.00**
Water	1.33
Ethyl alcohol	1.36
Oil	1.46
Glass (typical)	1.50

Note (Critical Angle):

If $\eta_i > \eta_r$ it is possible that $\left|\frac{\eta_i}{\eta_r} \cdot \sin\theta_i\right| > 1$.

$\Rightarrow$ There is no value of θ_r such that $\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i$.

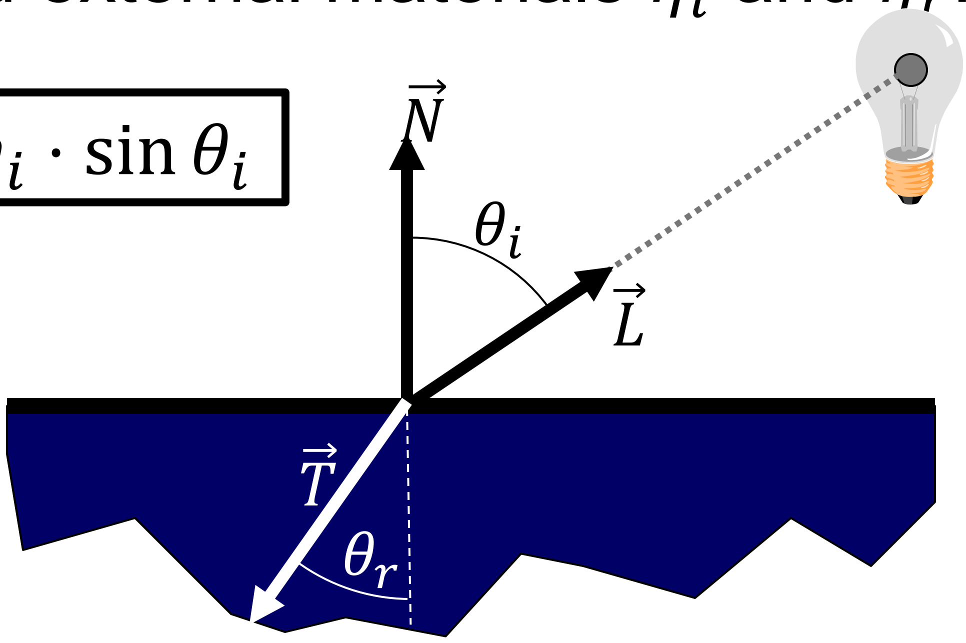
$\Rightarrow$ The light reflects off the surface, a.k.a. **internal reflection**, and does not pass through.

This describes the physics behind fiber-optic cables.

Snell's Law

Bending is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\boxed{\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i}$$

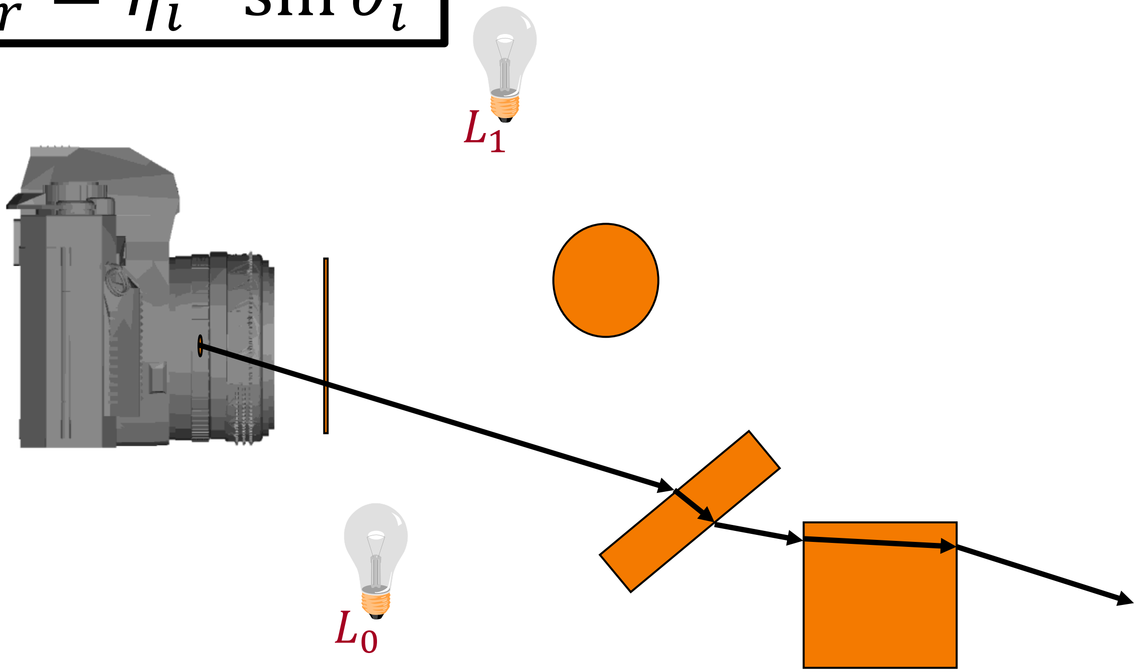


$$\boxed{\vec{T} = \left(\frac{\eta_i}{\eta_r} \cdot \cos\theta_i - \cos\theta_r\right) \cdot \vec{N} - \frac{\eta_i}{\eta_r} \cdot \vec{L}}$$

Snell's Law

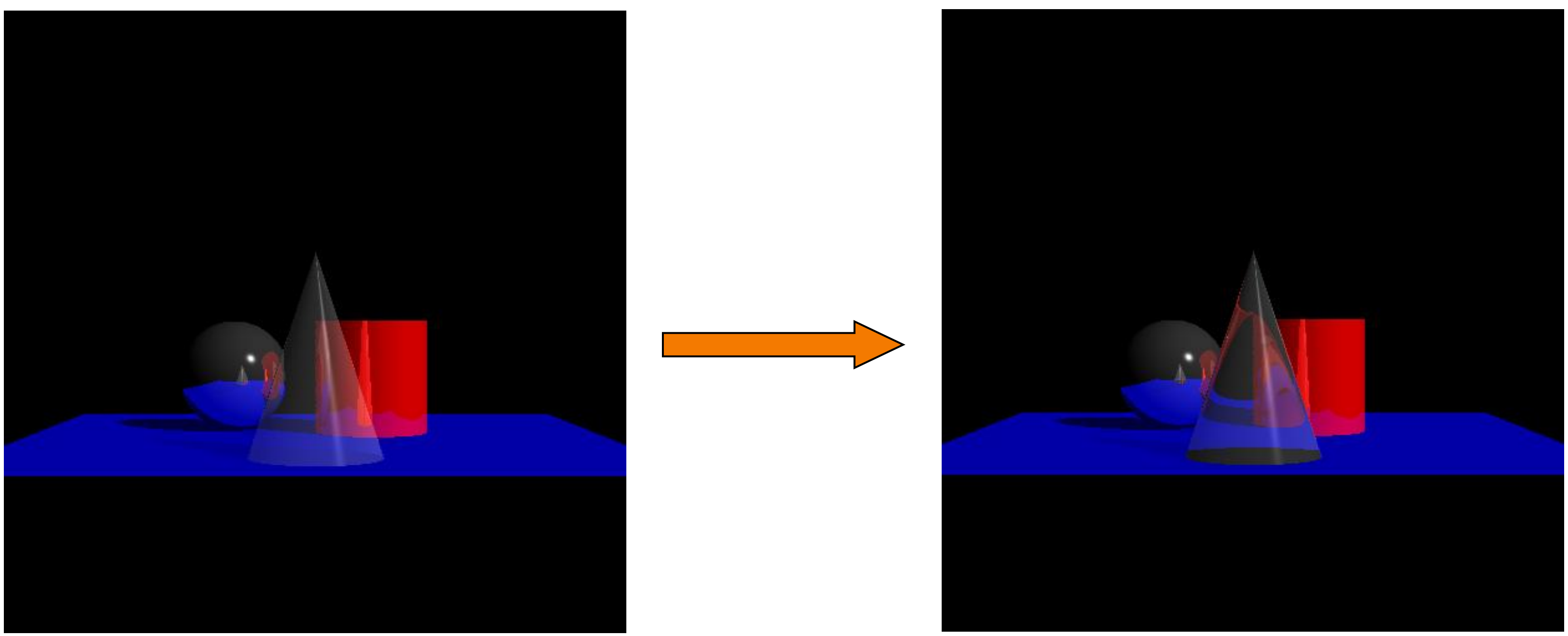
Bending is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\boxed{\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i}$$



Snell’s Law

Bending is determined by the indices of refraction of the internal and external materials η_i and η_r:



Snell's Law and Caustics

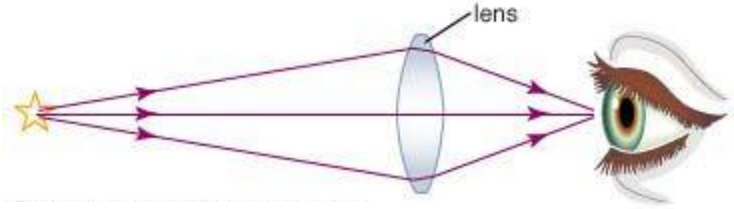
Challenge:

If a surface is transparent, then rays to the light source will not travel along straight paths

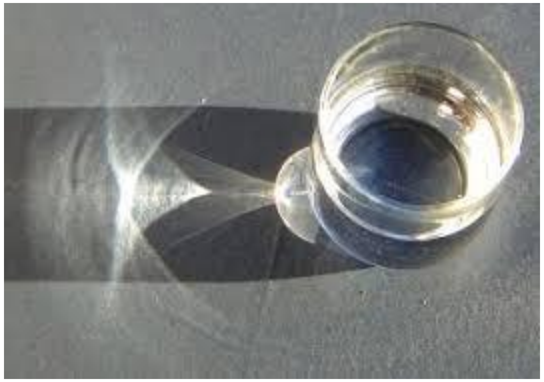
⇒ For a given ray/surface intersection, there may be multiple paths a ray can travel to get to the light

⇒ Summing the contribution of all paths from the eye to a light source gives **caustics**

✘ This is difficult to address with ray-tracing



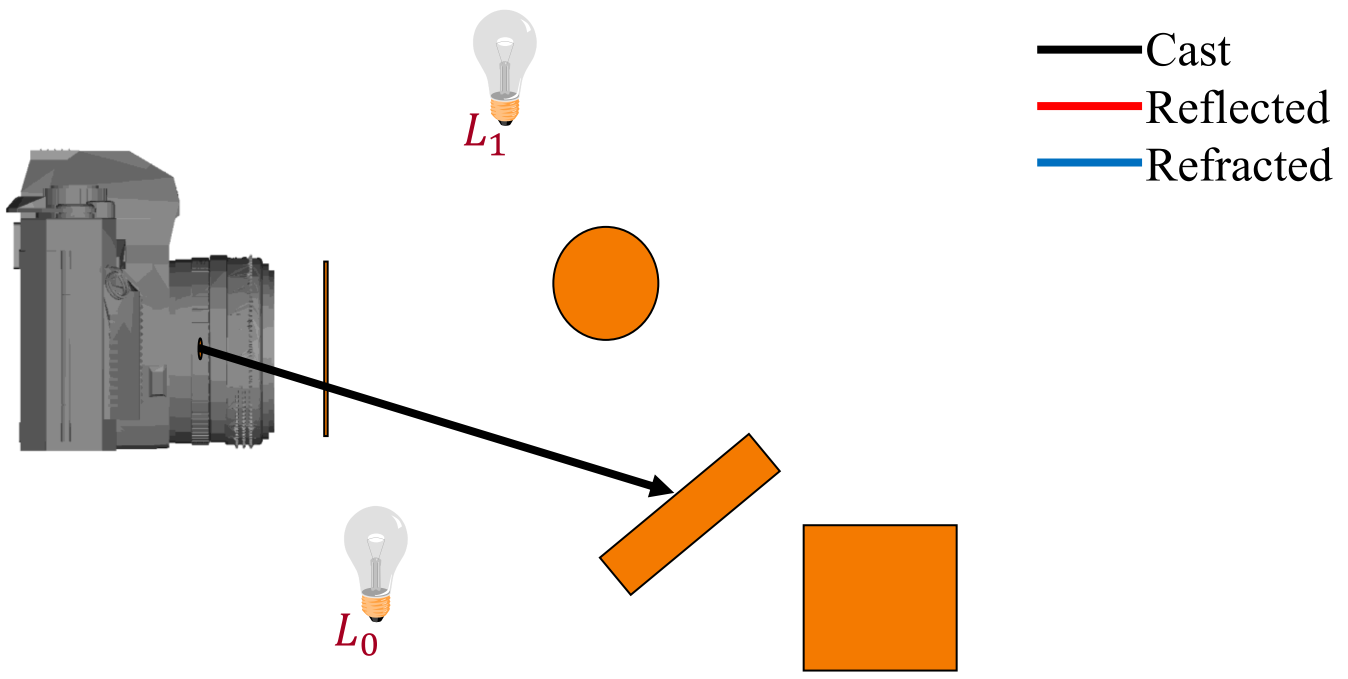
https://www.britannica.com/



https://en.wikipedia.org/wiki/Caustic_(optics)

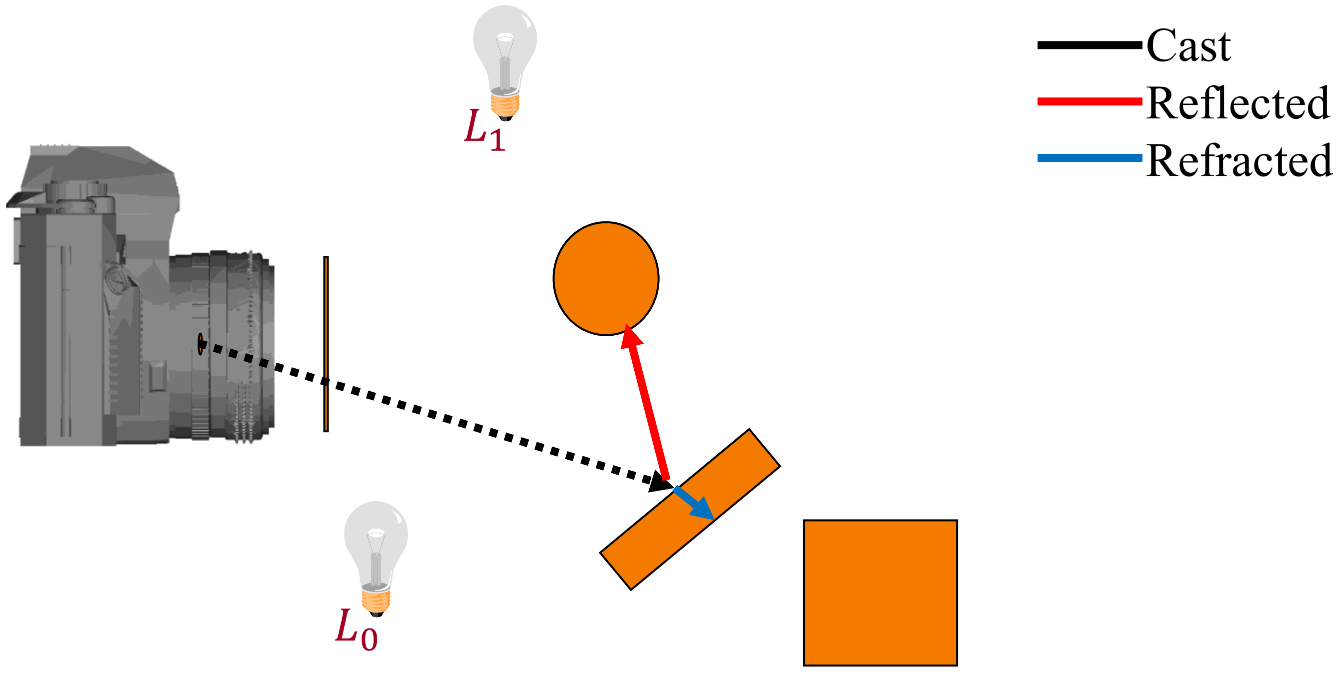
(Rough) Complexity Analysis

Simulating reflection and refraction in the ray tracer, a ray splits into two at half the bounces



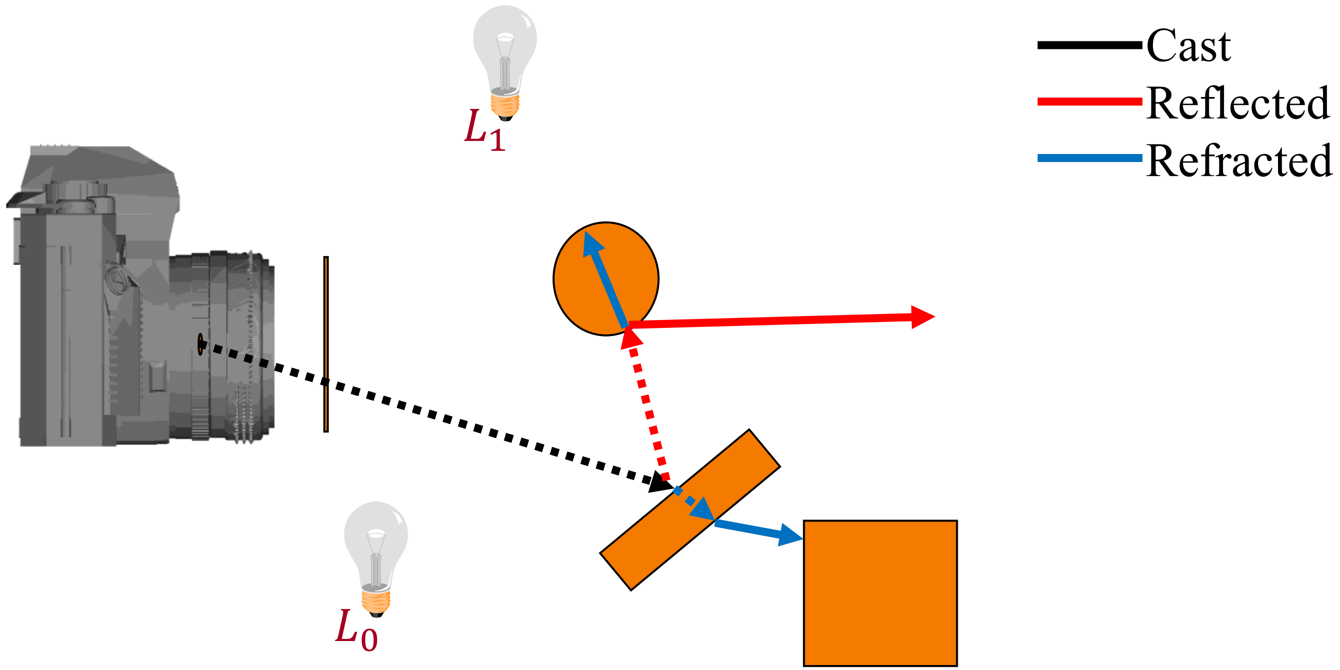
(Rough) Complexity Analysis

Simulating reflection and refraction in the ray tracer, a ray splits into two at half the bounces



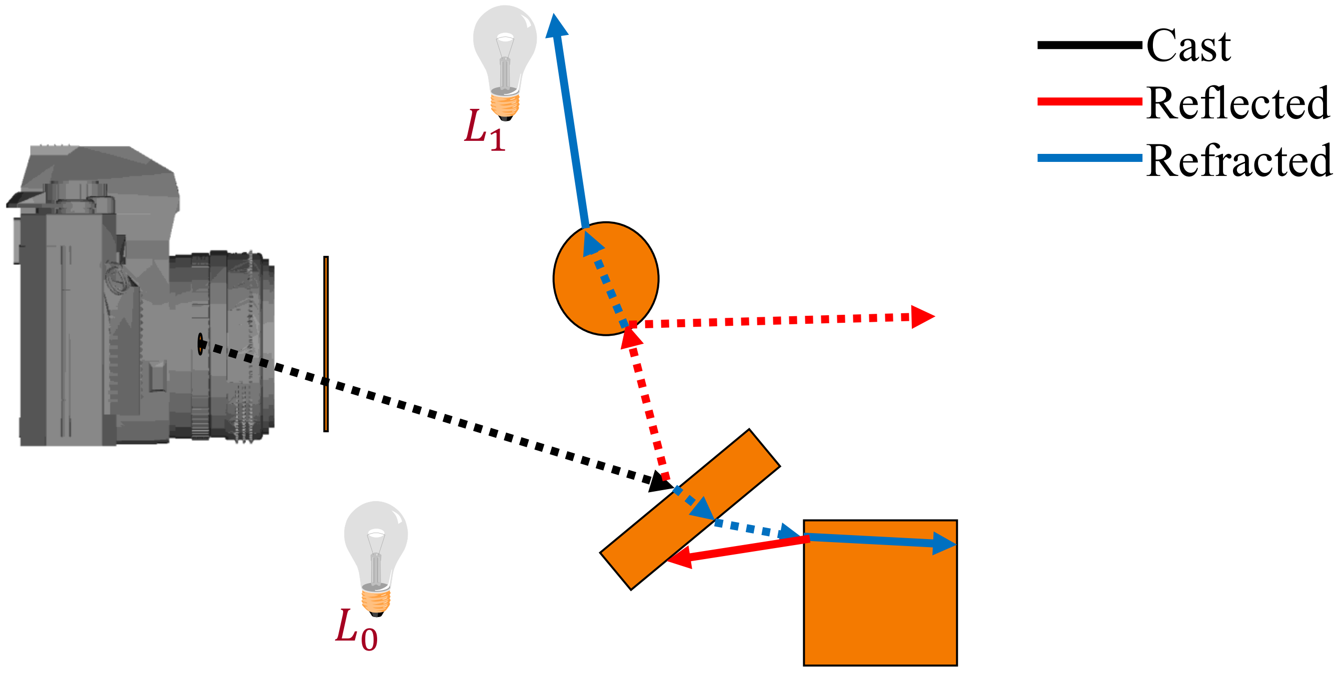
(Rough) Complexity Analysis

Simulating reflection and refraction in the ray tracer, a ray splits into two at half the bounces



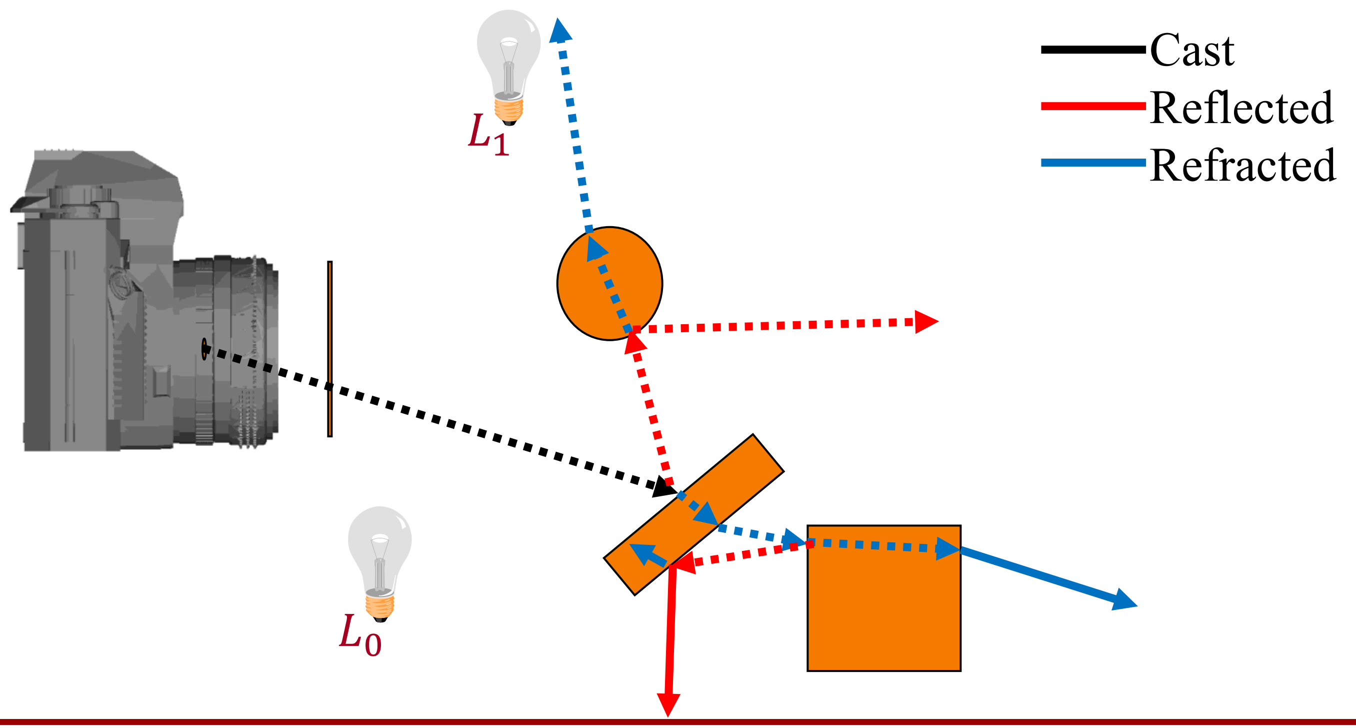
(Rough) Complexity Analysis

Simulating reflection and refraction in the ray tracer, a ray splits into two at half the bounces



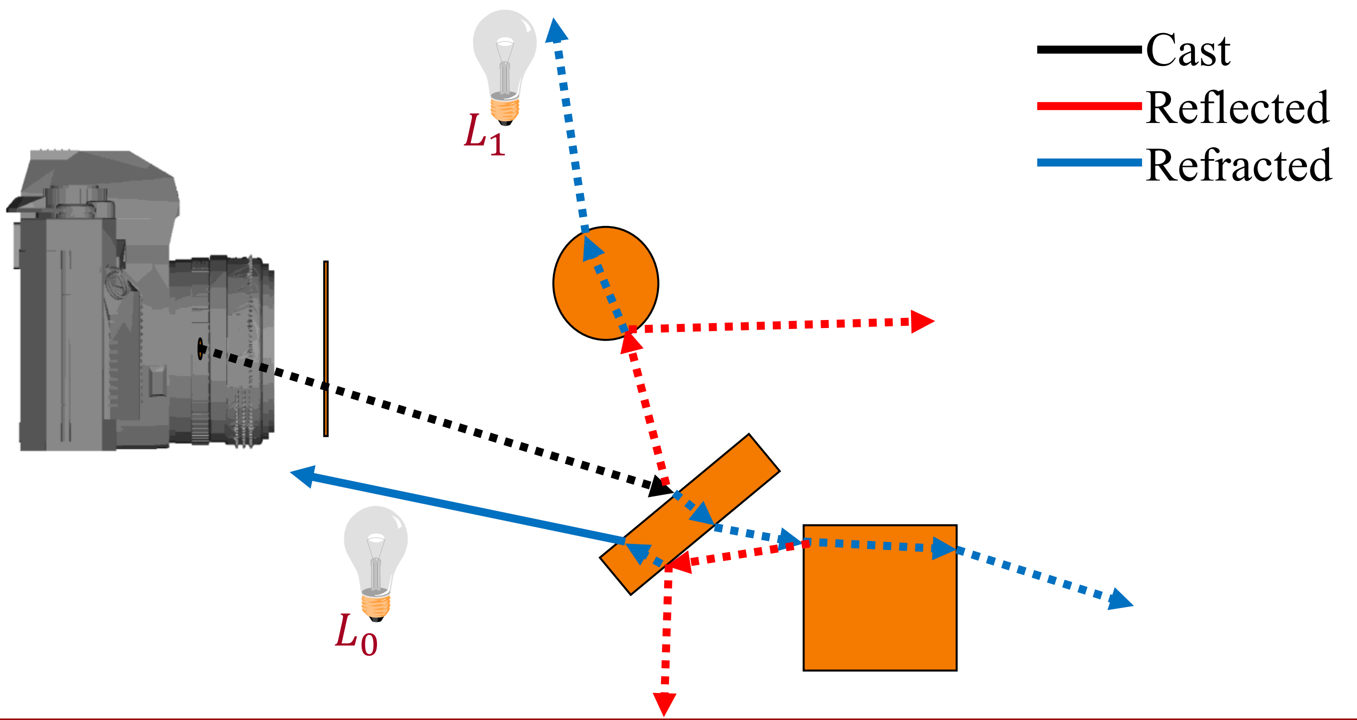
(Rough) Complexity Analysis

Simulating reflection and refraction in the ray tracer, a ray splits into two at half the bounces



(Rough) Complexity Analysis

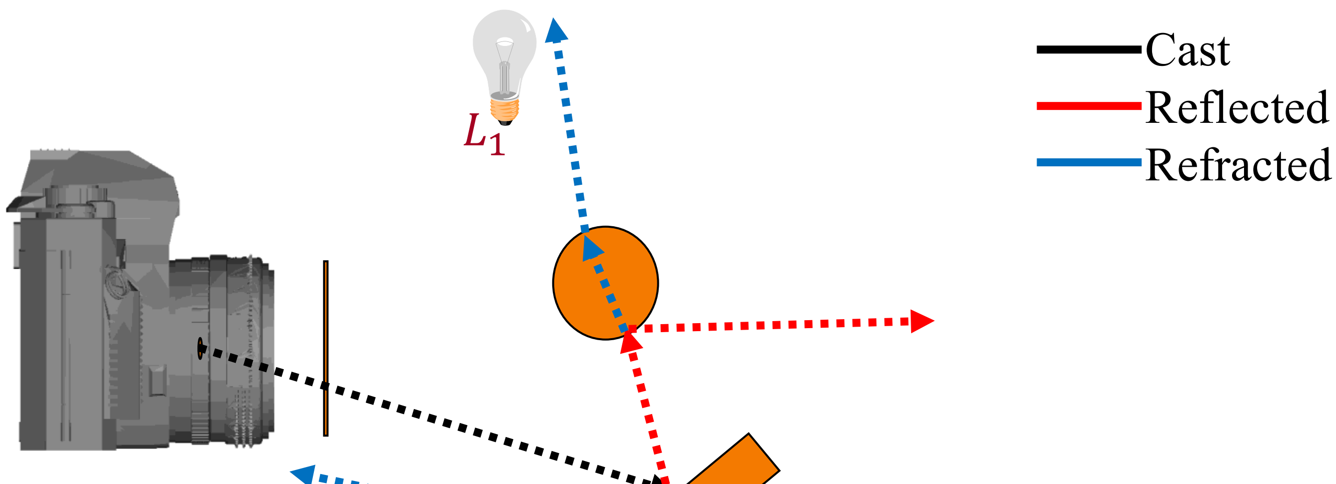
Simulating reflection and refraction in the ray tracer, a ray splits into two at half the bounces



(Rough) Complexity Analysis

Simulating reflection and refraction in the ray tracer, a ray splits into two at half the bounces

⇒ Exponential growth in the number of rays as a function of the number of bounces



Note:
We don't cast reflected rays from within the interior of a shape (unless we are modeling internal reflection).

Termination Criteria

```
Color GetColor( Scene scene , Ray< 3 > ray , float ir )
{
	Color c(0,0,0);
	Ray< 3 > reflect , refract;

	Intersection hit;

	if( FindIntersection( ray , scene , hit ) )
	{
		c += GetSurfaceColor( hit.position );

		if( Dot( ray.direction , hit.normal )<0 ) // Ray coming in from the outside
		{
			reflect.direction = Reflect( ray.direction , hit.normal );
			reflect.position = hit.position + reflect.direction*ε;
			c += GetColor( scene , reflect , ir )*hit.kSpec;
		}

		if(  Refract( ray.direction , hit.normal , ir , hit.ir , refract.direction )
		{
			refract.position = hit.position + refract.direction*ε;
			c += GetColor( scene , refract , hit.ir )*hit.kTran;
		}
	}
	return c;
}
```

Termination Criteria

How do we determine when to stop recursing?

- If the ray bounces around too much
- If the contribution will be too small

Termination Criteria (Recursion)

```
Color GetColor( Scene scene , Ray< 3 > ray , float ir , int rDepth , Color cutOff )
{
	Color c(0,0,0);
	Ray< 3 > reflect , refract;
	if( !rDepth || ( cutOff[0]>1 && cutOff[1]>1 && cutOff[2]>1 ) ) return c;
	Intersection hit;

	if( FindIntersection( ray , scene , hit ) )
	{
		c += GetSurfaceColor( hit.position );

		if( Dot( ray.direction , hit.normal )<0 ) // Ray coming in from the outside
		{
			reflect.direction = Reflect( ray.direction , hit.normal );
			reflect.position = hit.position + reflect.direction*ε;
			c += GetColor( scene , reflect , ir , rDepth-1 , cutOff/hit.kSpec )*hit.kSpec;
		}

		if(  Refract( ray.direction , hit.normal , ir , hit.ir , refract.direction )
		{
			refract.position = hit.position + refract.direction*ε;
			c += GetColor( scene , refract , hit.ir , rDepth-1 , cutOff/hit.kTran )*hit.kTran;
		}
	}
	return c;
}
```

Termination Criteria (Contribution)

```
Color GetColor( Scene scene , Ray< 3 > ray , float ir , int rDepth  , Color cutOff  )
{
        Color c(0,0,0);
        Ray< 3 > reflect , refract;
        if( !rDepth || ( cutOff[0]>1 && cutOff[1]>1 && cutOff[2]>1 ) ) return c;
}
```

Termination Criteria (Contribution)

```
Color GetColor( Scene scene , Ray< 3 > ray , float ir , int rDepth , Color cutOff )
{
        Color c(0,0,0);
        Ray< 3 > reflect , refract;
        if( !rDepth || ( cutOff[0]>1 && cutOff[1]>1 && cutOff[2]>1 ) ) return c;
        Intersection hit;

        if( FindIntersection( ray , scene , hit ) )
        {
                c += GetSurfaceColor( hit.position );
```

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_S \cdot I_R + K_T \cdot I_T$$

Termination Criteria (Contribution)

```
Color GetColor( Scene scene , Ray< 3 > ray , float ir , int rDepth , Color cutOff )
{
	Color c(0,0,0);
	Ray< 3 > reflect , refract;
	if( !rDepth || ( cutOff[0]>1 && cutOff[1]>1 && cutOff[2]>1 ) ) return c;
	Intersection hit;

	if( FindIntersection( ray , scene , hit ) )
	{
		c += GetSurfaceColor( hit.position );
		if( Dot( ray.direction , hit.normal )<0 ) // Ray coming in from the outside
		{
			reflect.direction = Reflect( ray.direction , hit.normal );
			reflect.position = hit.position + reflect.direction*ε;
			c += GetColor( scene , reflect , ir , rDepth-1 , cutOff/hit.kSpec )*hit.kSpec;
		}
```

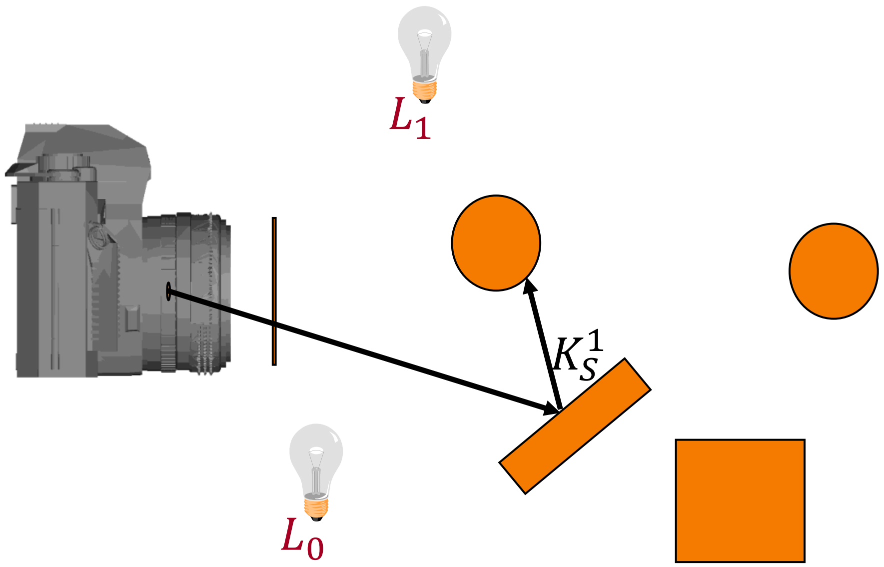
$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + \boxed{K_S \cdot I_R} + K_T \cdot I_T$$

Termination Criteria (Contribution)

```
c += GetColor( scene , reflect , ir , rDepth-1 , cutOff/hit.kSpec )*hit.kSpec;
```

The first reflected ray should be cast if it *can* contribute an intensity greater than the cut-off.

$\Rightarrow K_S^1 > \text{cut-off}$



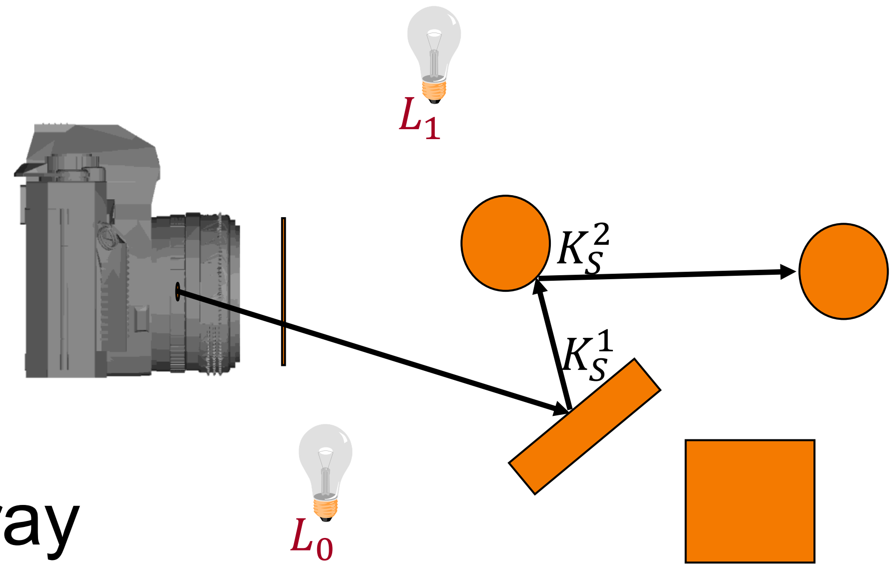
Termination Criteria (Contribution)

```
c += GetColor( scene , reflect , ir , rDepth-1 , cutOff/hit.kSpec )*hit.kSpec;
```

The second reflected ray should be cast if it *can* contribute an intensity greater than the cut-off.

$$\Rightarrow K_S^2 \cdot K_S^1 > \text{cut-off}$$

$$\Leftrightarrow K_S^2 > \frac{\text{cut-off}}{K_S^1}$$



$\Rightarrow$ The second reflected ray should be cast if it can reflect at least $\frac{\text{cut-off}}{K_S^1}$ of the local intensity

Termination Criteria (Contribution)

```
Color GetColor( Scene scene , Ray< 3 > ray , float ir , int rDepth  , Color cutOff  )
{
	Color c(0,0,0);
	Ray< 3 > reflect , refract;
	if( !rDepth || ( cutOff[0]>1 && cutOff[1]>1 && cutOff[2]>1 ) ) return c;
	Intersection hit;

	if( FindIntersection( ray , scene , hit ) )
	{
		c += GetSurfaceColor( hit.position );

		if( Dot( ray.direction , hit.normal )<0 ) // Ray coming in from the outside
		{
			reflect.direction = Reflect( ray.direction , hit.normal );
			reflect.position = hit.position + reflect.direction*ε;
			c += GetColor( scene , reflect , ir , rDepth-1 , cutOff/hit.kSpec )*hit.kSpec;
		}

		if(  Refract( ray.direction , hit.normal , ir , hit.ir , refract.direction )
		{
			refract.position = hit.position + refract.direction*ε;
			c += GetColor( scene , refract , hit.ir , rDepth-1 , cutOff/hit.kTran )*hit.kTran;
		}
```

$$I = K_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R}(\vec{L}) \rangle^{K_n}\right) \cdot I_L \cdot S_L\right] + K_S \cdot I_R + \boxed{K_T \cdot I_T}$$

Termination Criteria

```
Color GetColor( Scene scene , Ray< 3 > ray , float ir , int rDepth , Color cutOff )
{
	Color c(0,0,0);
	Ray< 3 > reflect , refract;
	if( !rDepth || ( cutOff[0]>1 && cutOff[1]>1 && cutOff[2]>1 ) ) return c;
	In
	if(
	{
```

Why add a small amount of the direction to the position?

So that the new ray does not hit its starting location!

Note that the new position is on the same ray, just offset forward a little.

For the same reason, you will want to offset rays when shadow testing.

```
			reflect.direction = Reflect( ray.direction , hit.normal );
			reflect.position = hit.position + reflect.direction*ε;
			c += GetColor( scene , reflect , ir , rDepth-1 , cutOff/hit.kSpec )*hit.kSpec;
		}

		if(  Refract( ray.direction , hit.normal , ir , hit.ir , refract.direction )
		{
			refract.position = hit.position + refract.direction*ε
			c += GetColor( scene , refract , hit.ir , rDepth-1 , cutOff/hit.kTran )*hit.kTran;
		}
	}
	return c;
}
```

Index of Refraction

```
Color GetColor( Scene scene , Ray< 3 > ray , float ir , int rDepth , Color cutOff )
{
	Color c(0,0,0);
```

<u>For simplicity</u>:

Assume interface is always between some medium and air (or vice-versa).

⇒ If we enter the medium: The index of refraction is that of the medium

Otherwise: The index of refraction is that of air ($\eta = 1$)

(Alignment of the normal and ray direction tells us which case we're in.)

<u>Reflection</u>: We stay in the same material ⇒ index of refraction is unchanged

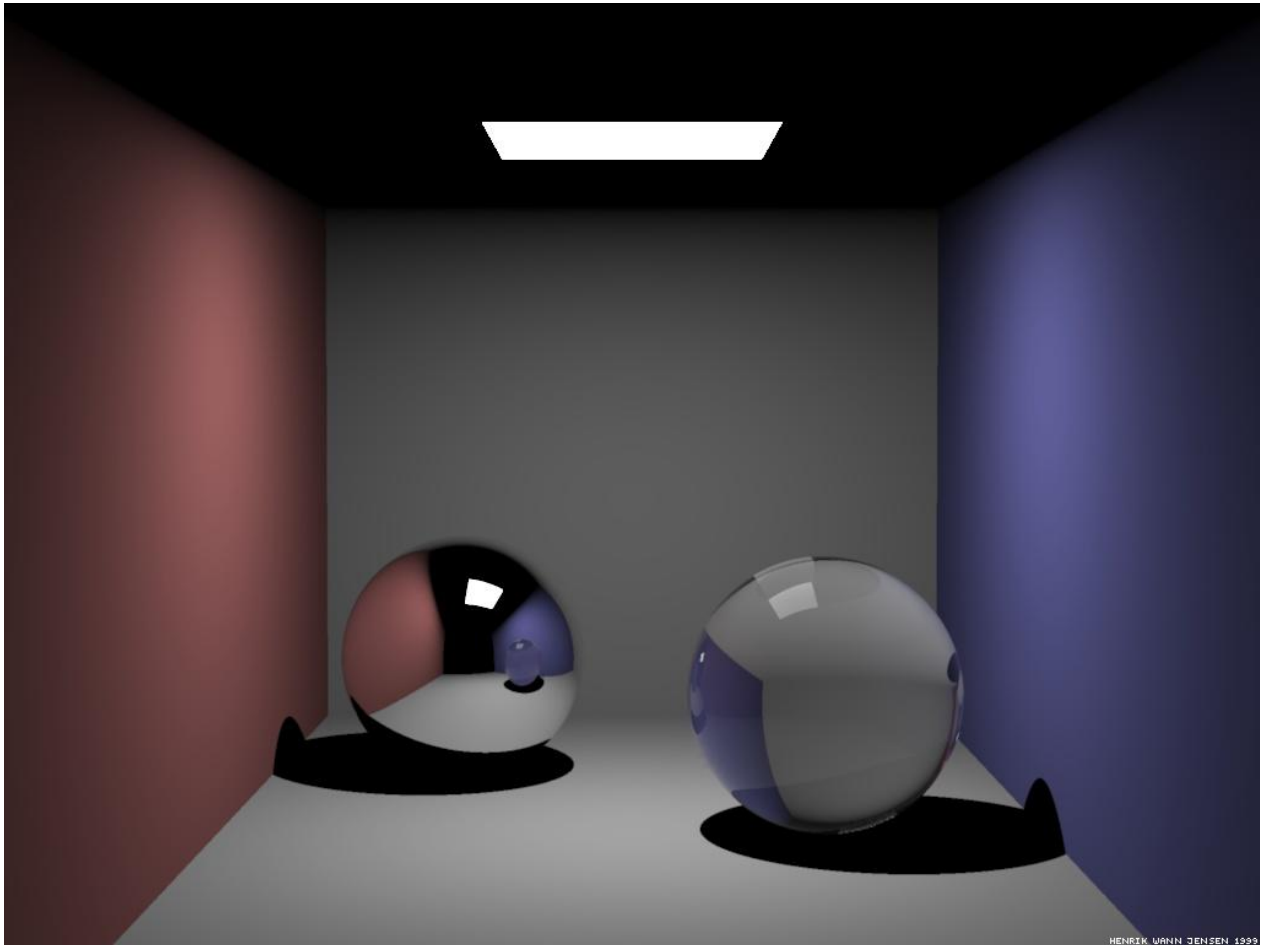
<u>Refraction</u>: We enter a new material ⇒ index of refraction changes

```
			c += GetColor( scene , reflect , ir , rDepth-1 , cutOff/hit.kSpec )*hit.kSpec;
		}

		if(  Refract( ray.direction , hit.normal , ir , hit.ir , refract.direction )
		{
			refract.position = hit.position + refract.direction*ε;
			c += GetColor( scene , refract , hit.ir , rDepth-1 , cutOff/hit.kTran )*hit.kTran;
		}
	}
	return c;
}
```

Illumination Examples

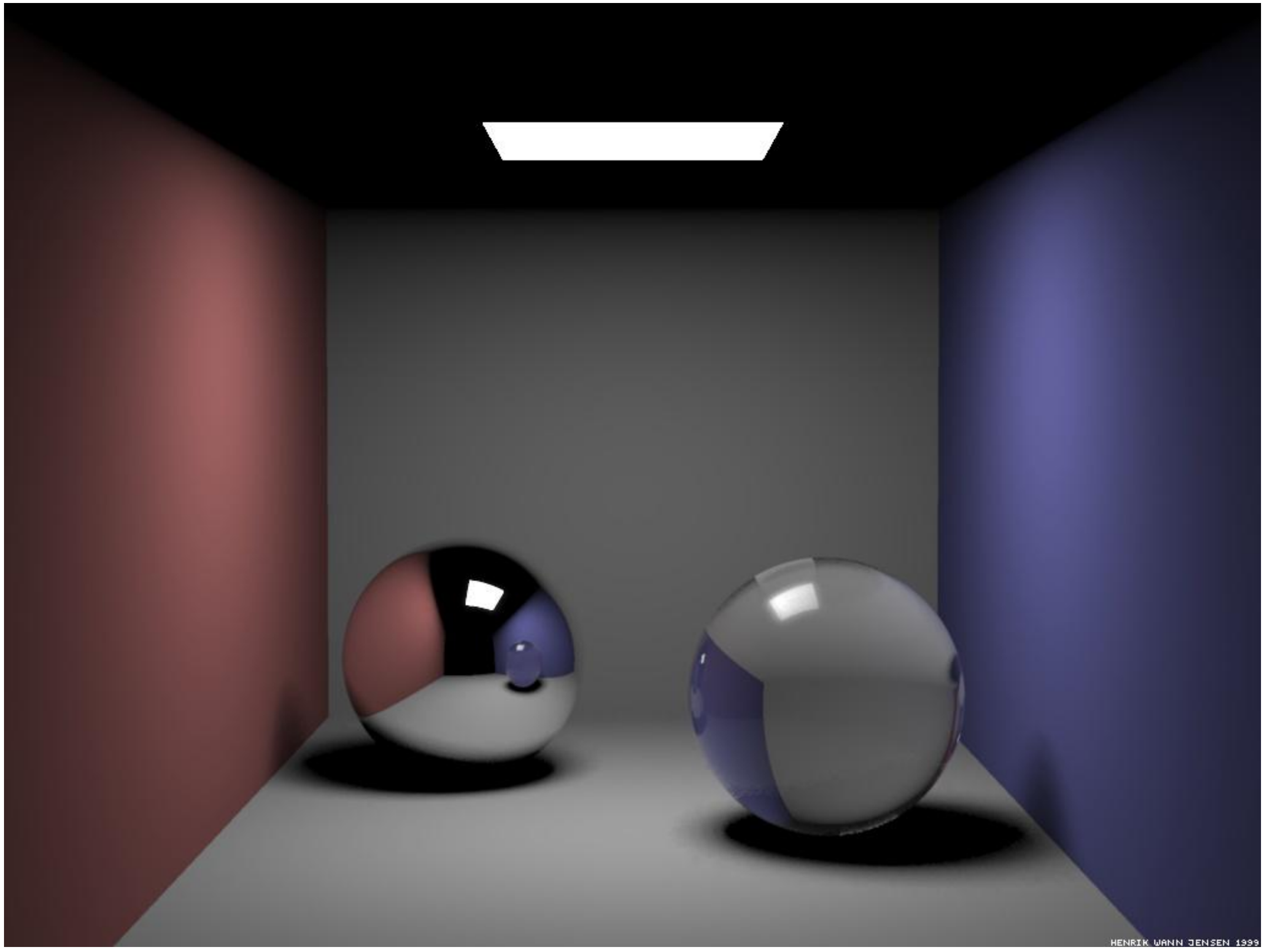
Ray tracing (rays to point light source)



Courtesy Henrik Wann Jensen

Illumination Examples

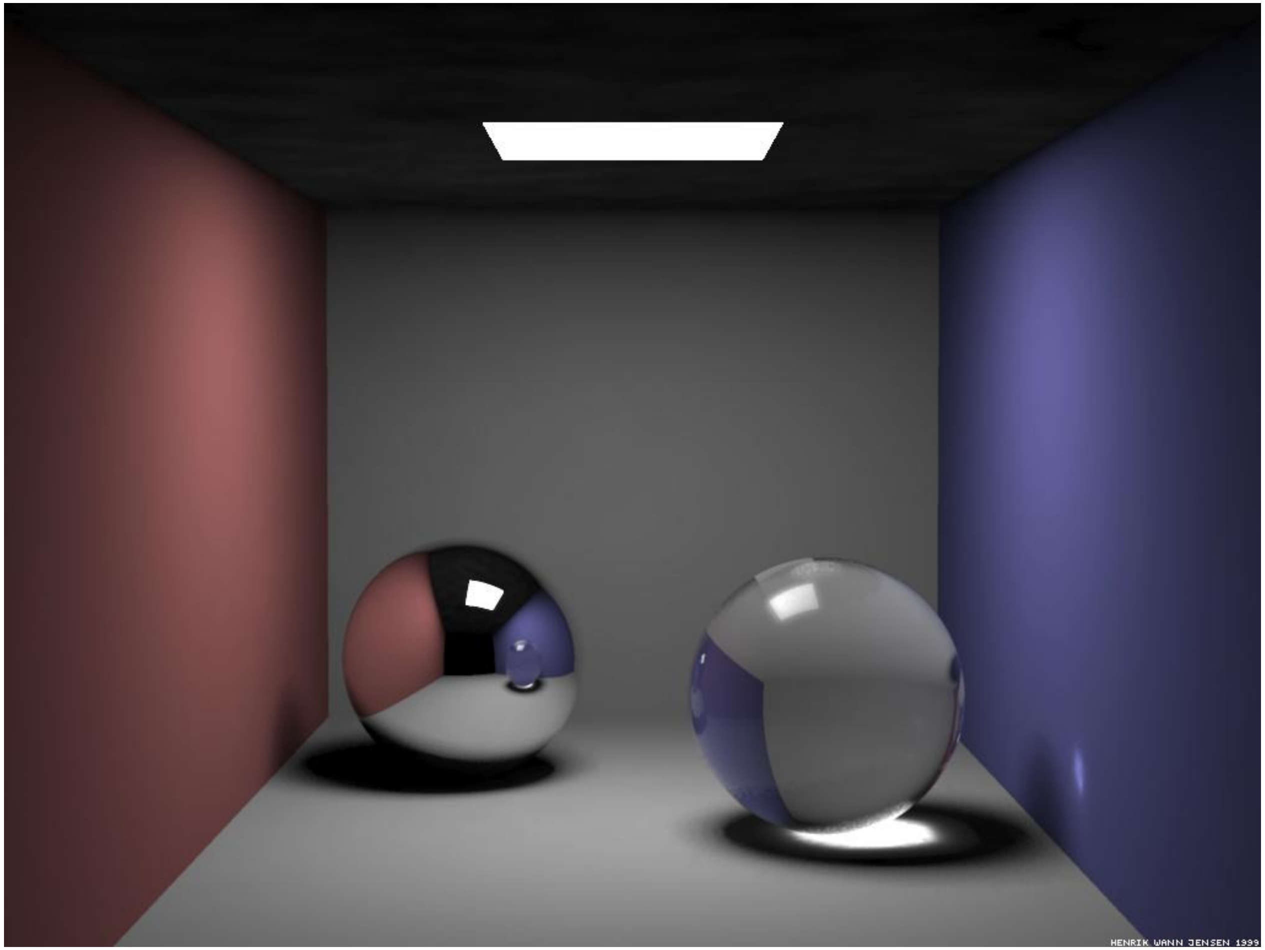
Soft shadows (rays to area light source)



Courtesy Henrik Wann Jensen

Illumination Examples

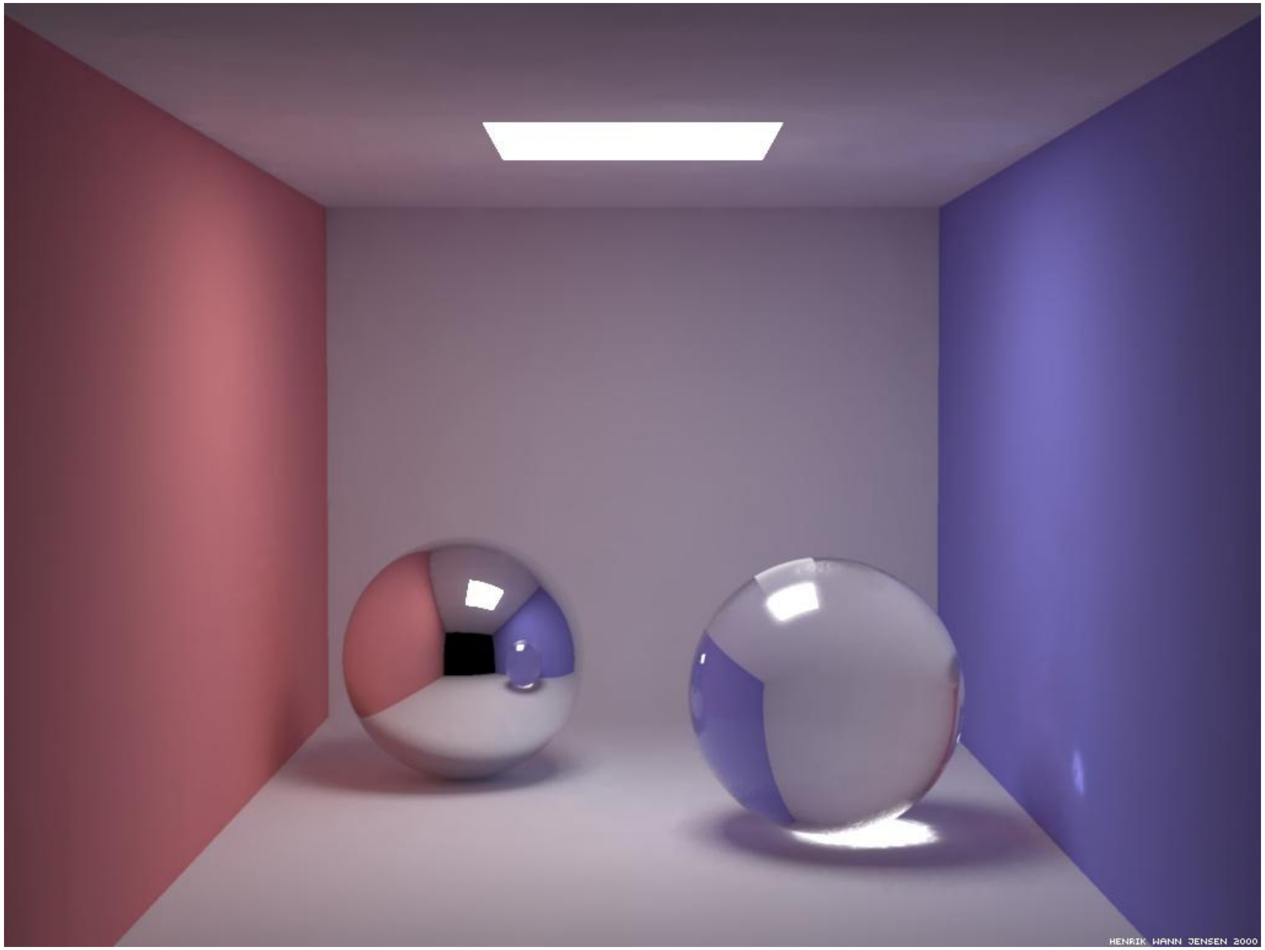
Caustics (rays from light source)



Courtesy Henrik Wann Jensen

Illumination Examples

Full Global Illumination



Courtesy Henrik Wann Jensen

Summary

Ray casting (direct Illumination)

- Use simple analytic approximations for light source emission and surface reflectance

Recursive ray tracing (global illumination)

- Incorporate shadows, mirror reflections, and refractions

Much of this is an approximation making computation practical