Indirect Illumination

Michael Kazhdan

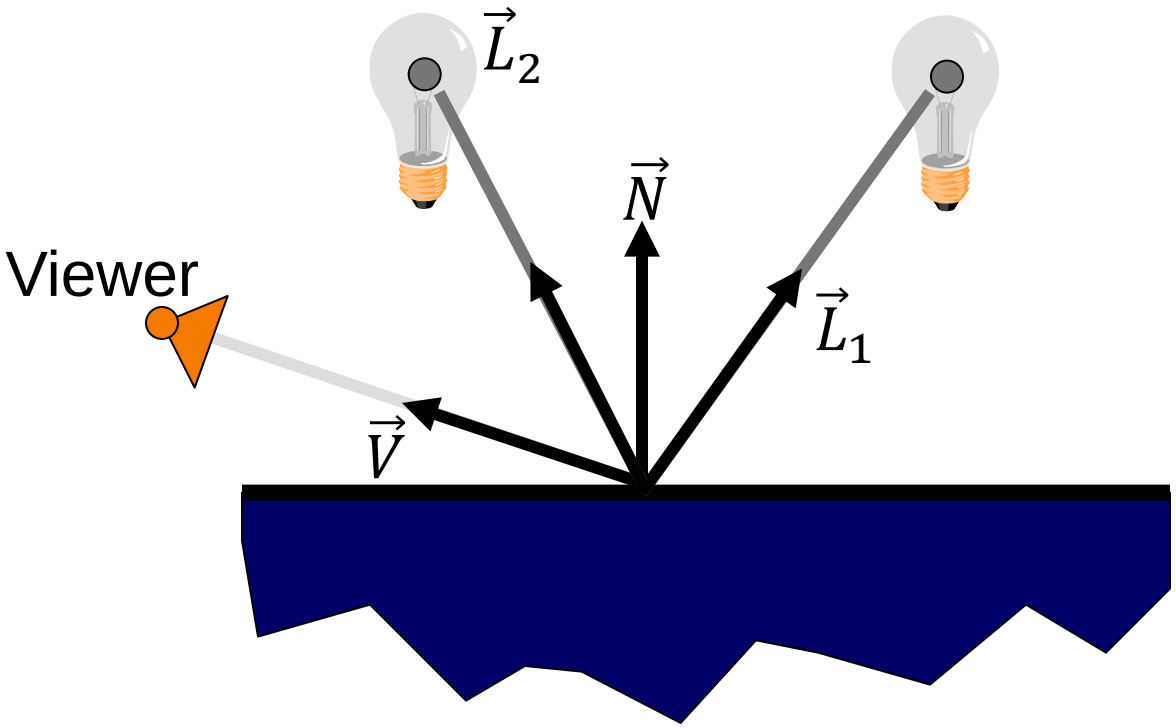
(601.457/657)

HB Ch. 14.1, 14.2

FvDFH 16.1, 16.2

Surface Illumination Calculation

- Multiple light source:



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

Overview

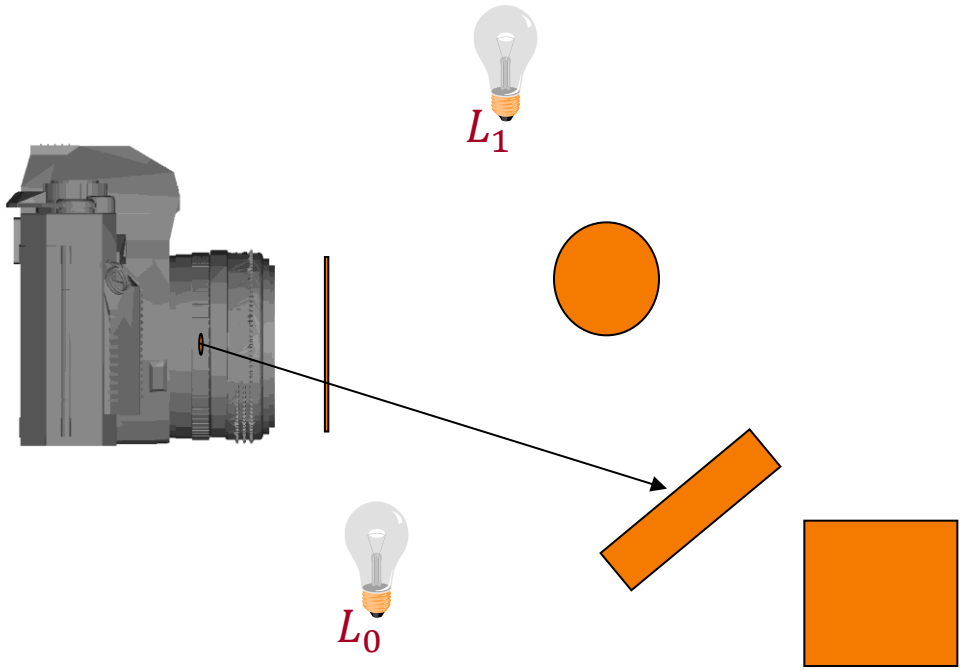
- Direct Illumination
  - Emission at light sources
  - Direct light at surface points
- Global illumination
  - Shadows
  - Inter-object reflections
  - Transmissions

Shadows

- How do we tell if a point where the ray intersects the surface is in shadow?
  - Cast ray towards each light source L
  - If the ray is blocked, do not consider the contribution of the light.

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

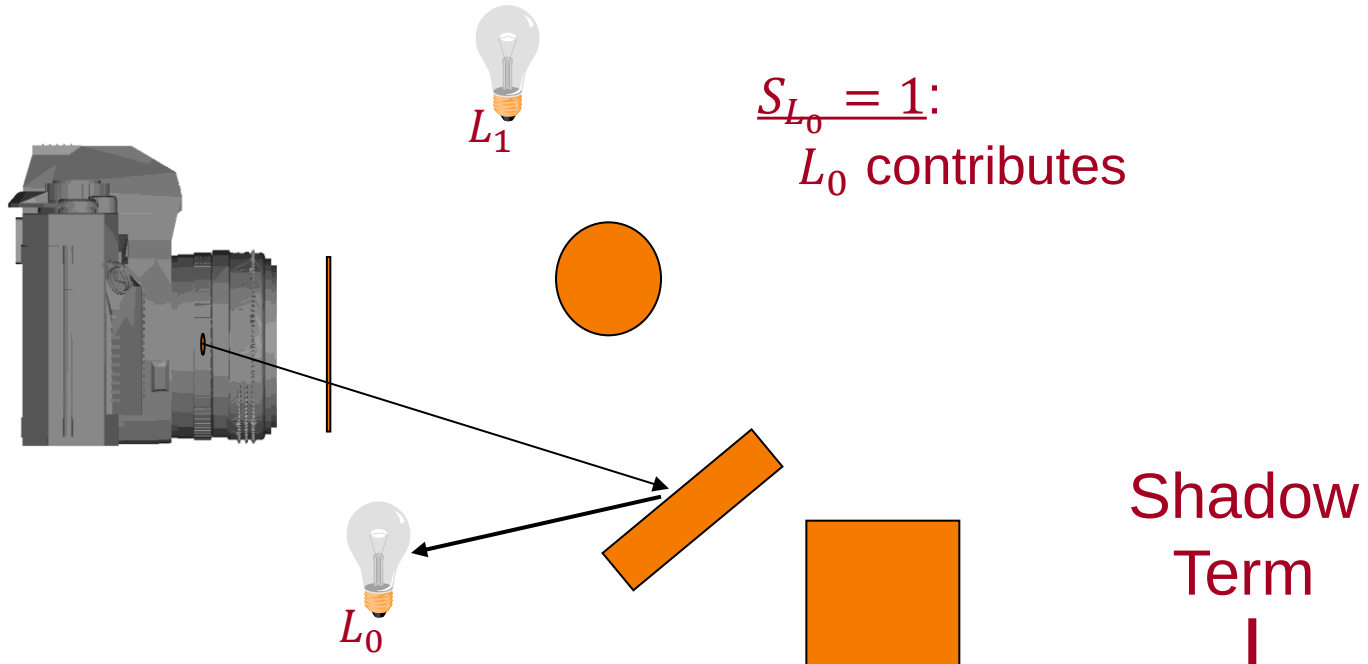


Shadow Term

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

Shadows

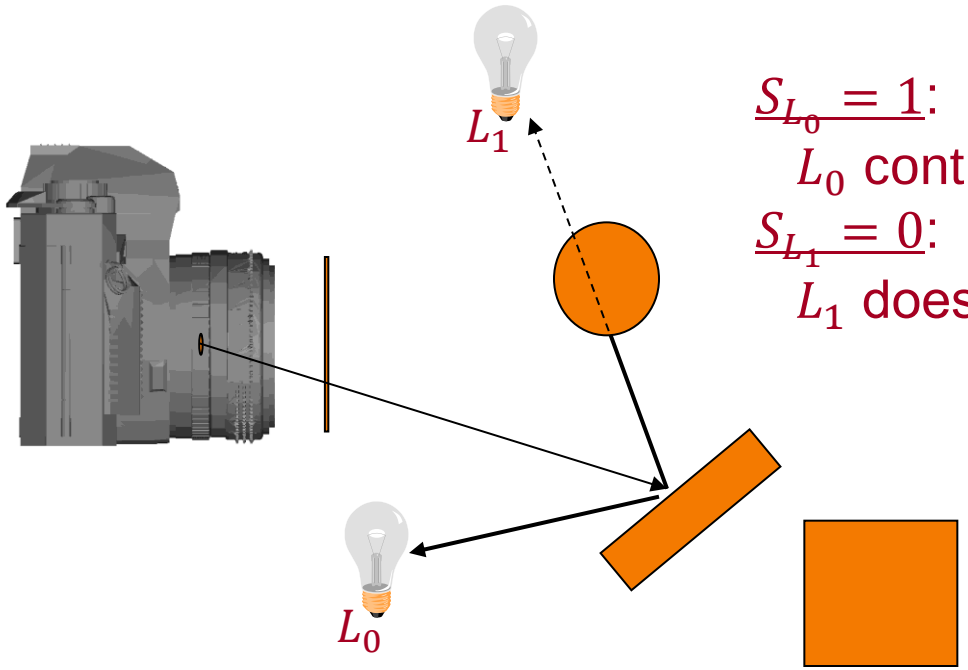
- Shadow term tells if light sources are blocked
  - Cast ray towards each light source L
  - $S_L = 0$ if ray is blocked, $S_L = 1$ otherwise



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

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



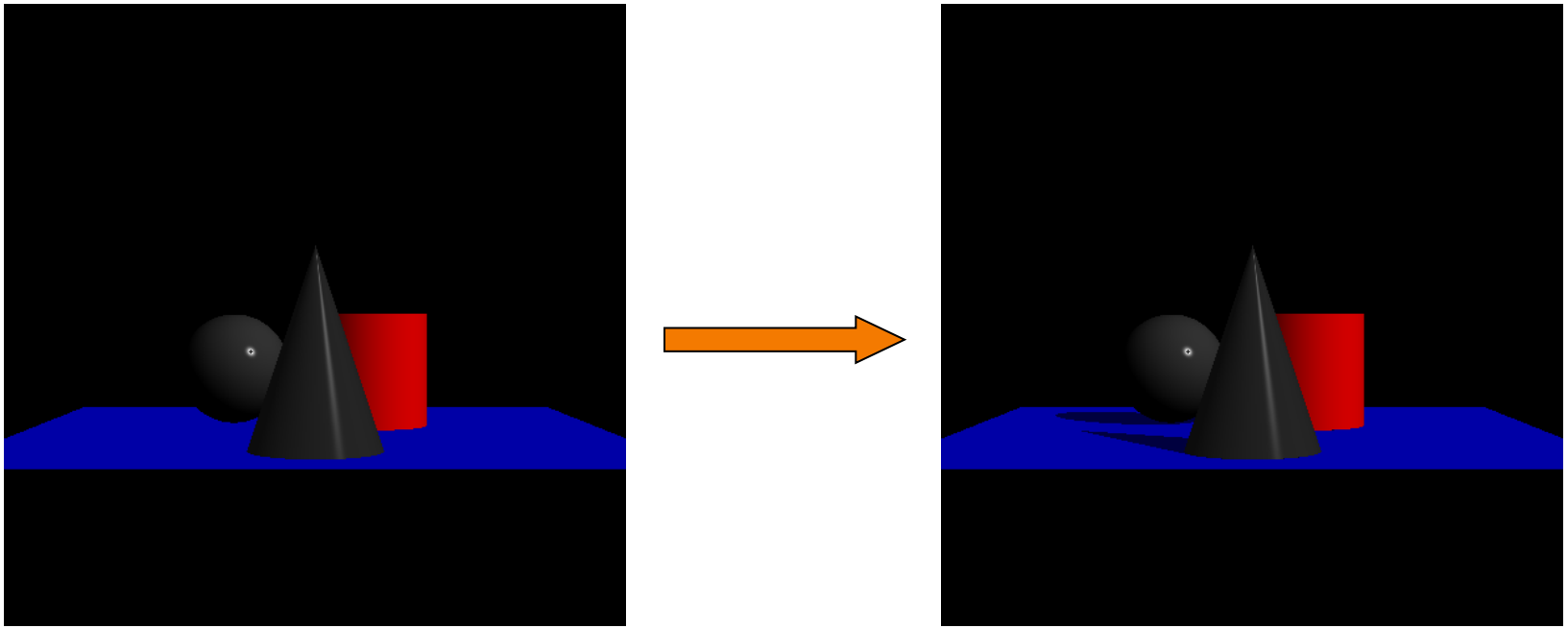
$\underline{S_{L_0} = 1}$:
L_0 contributes
$\underline{S_{L_1} = 0}$:
L_1 doesn't contribute

Shadow Term

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

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

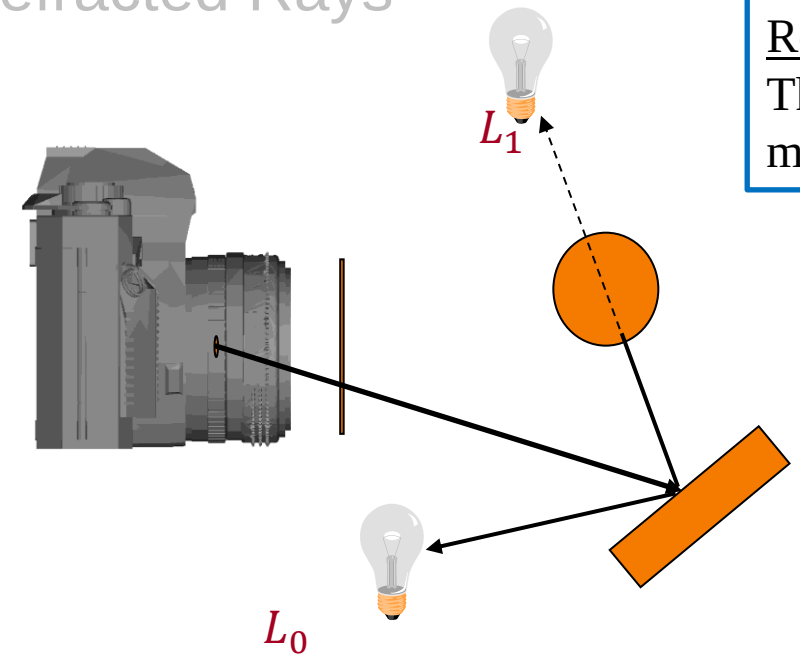


Recursive Ray Tracing

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

Mirror Reflections

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays



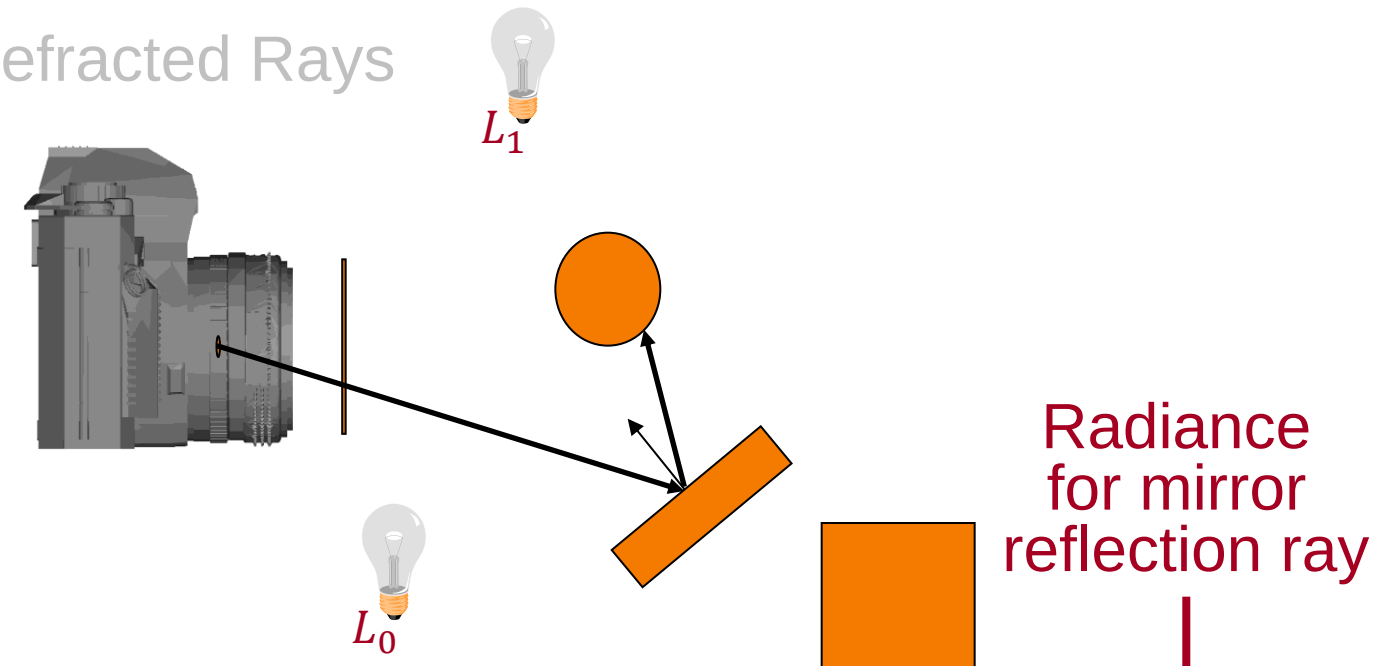
Recall:
The specularity of a surface, K_S, measures how mirror-like it is.

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

Mirror Reflections

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays

       What is the contribution from the reflected direction?
    2. Refracted Rays



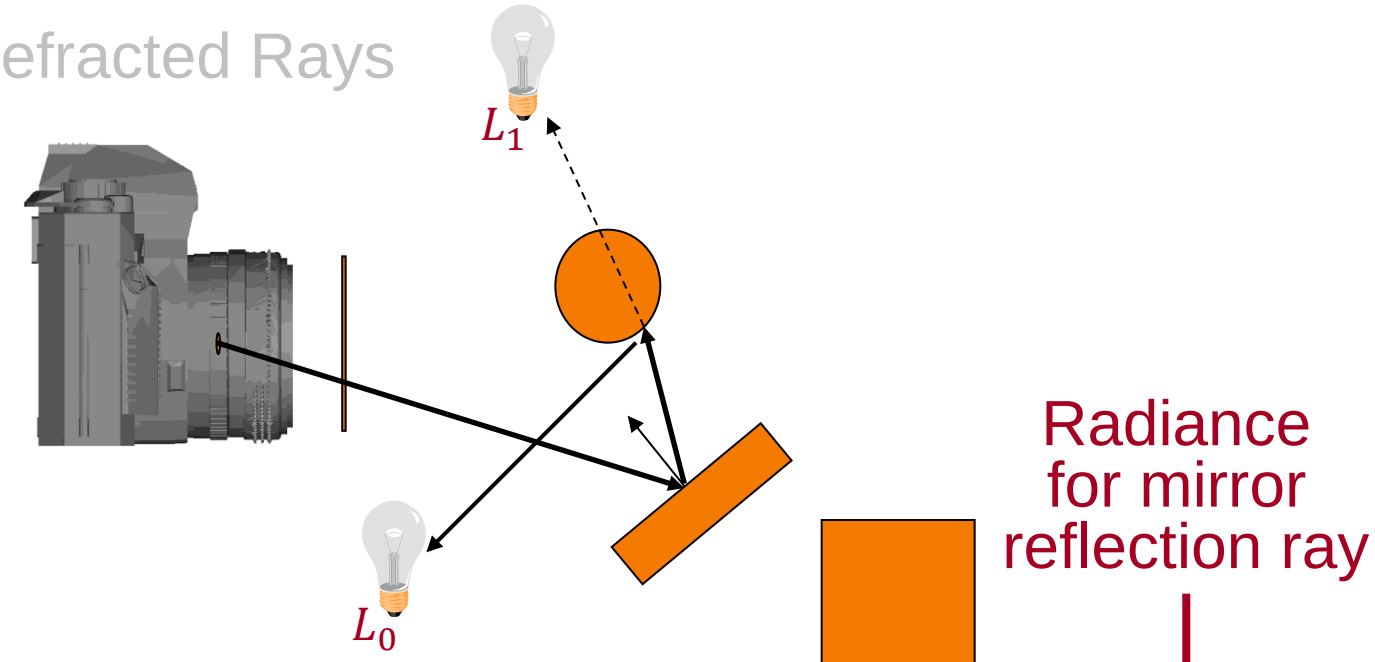
Radiance for mirror reflection ray

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

Mirror Reflections

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays

       What is the contribution from the reflected direction?
    2. Refracted Rays



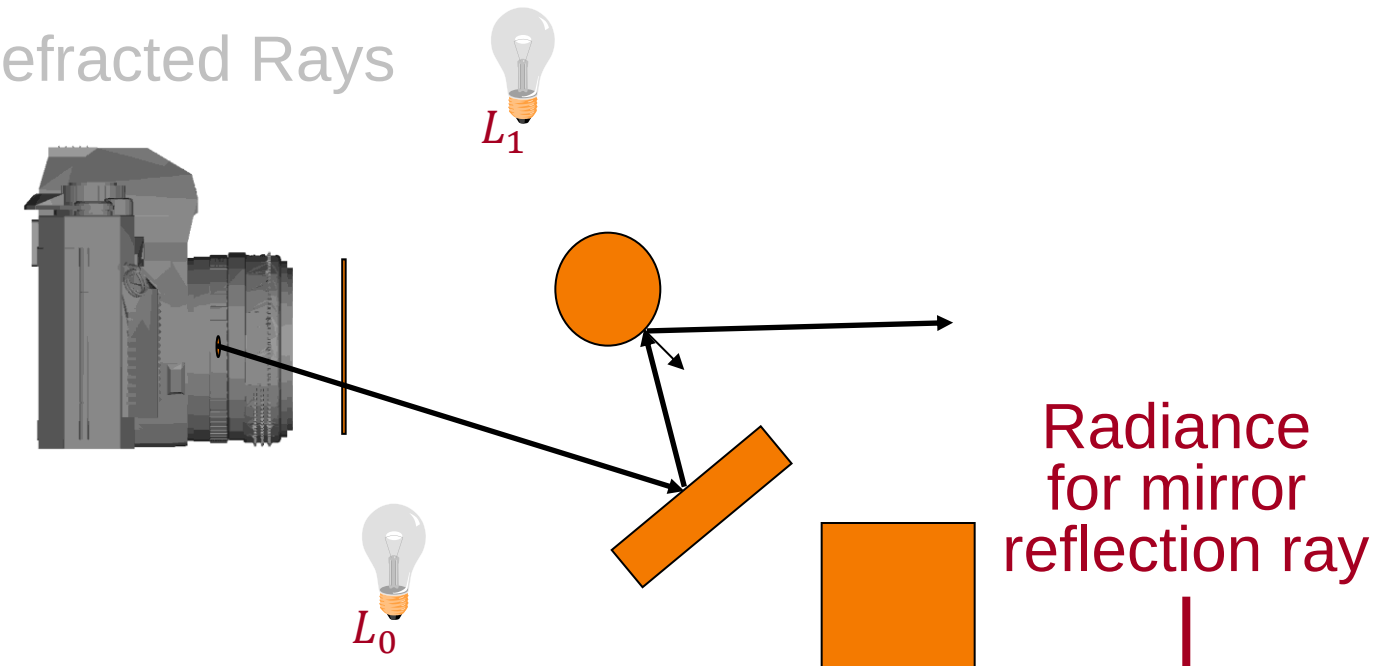
Radiance for mirror reflection ray

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

Mirror Reflections

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays

       What is the contribution from the reflected direction?
    2. Refracted Rays

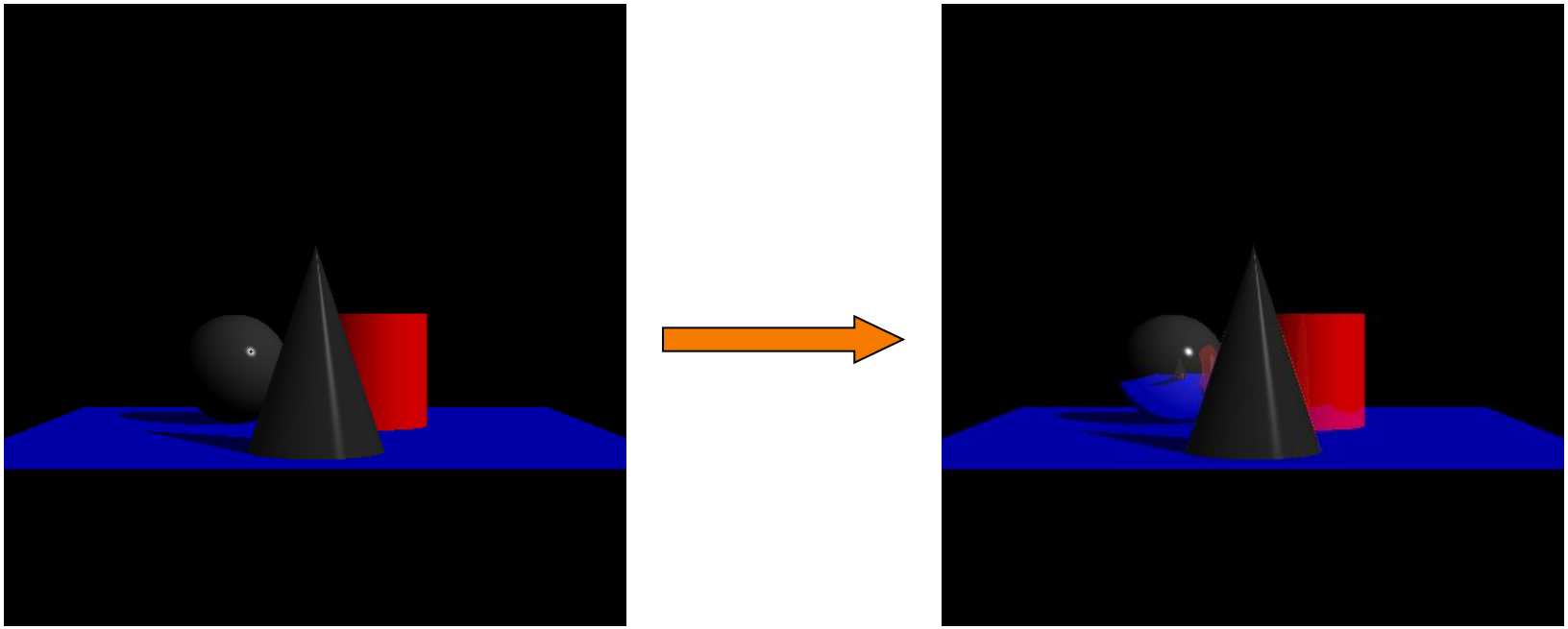


Radiance for mirror reflection ray

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

Mirror Reflections

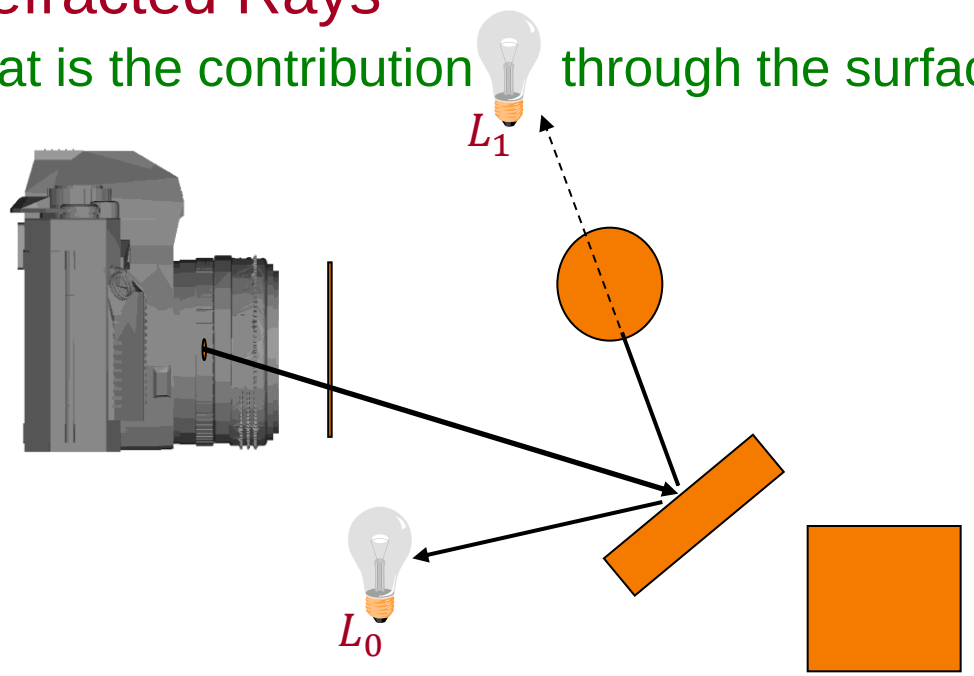
- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays



Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

What is the contribution through the surface?



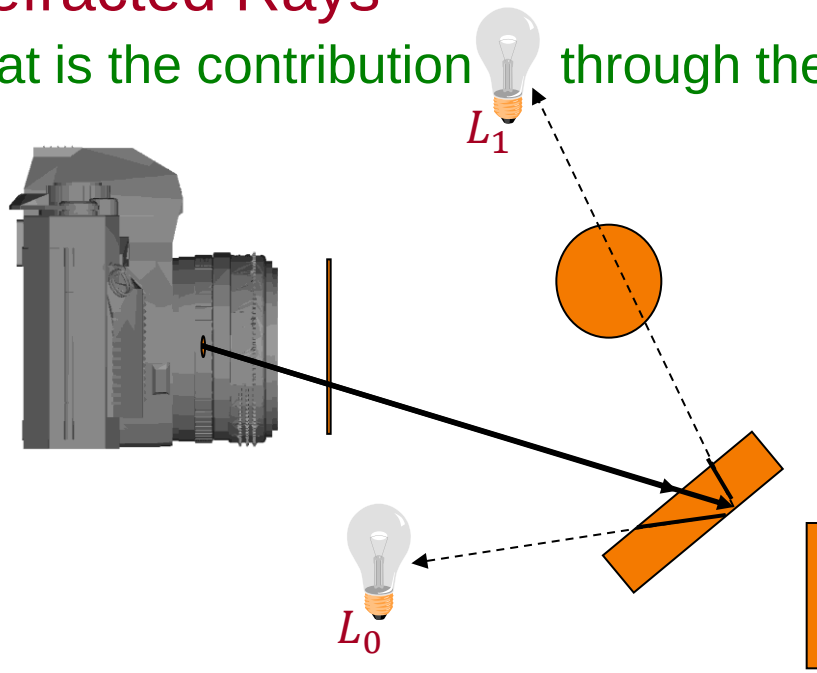
Radiance for refraction ray

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

Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

What is the contribution through the surface?



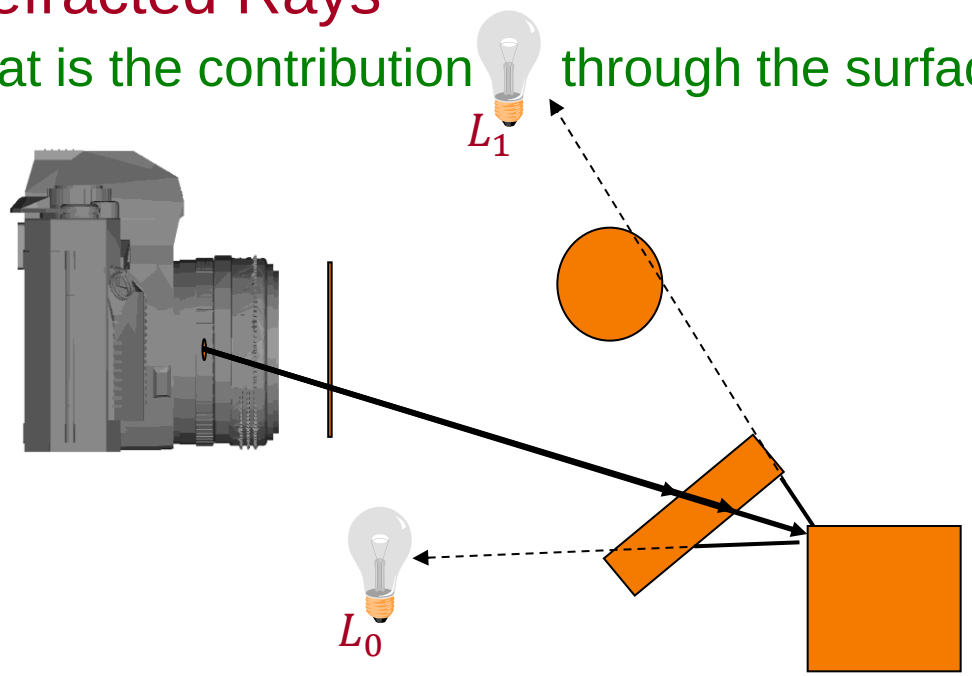
Radiance for refraction ray

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

Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

What is the contribution through the surface?



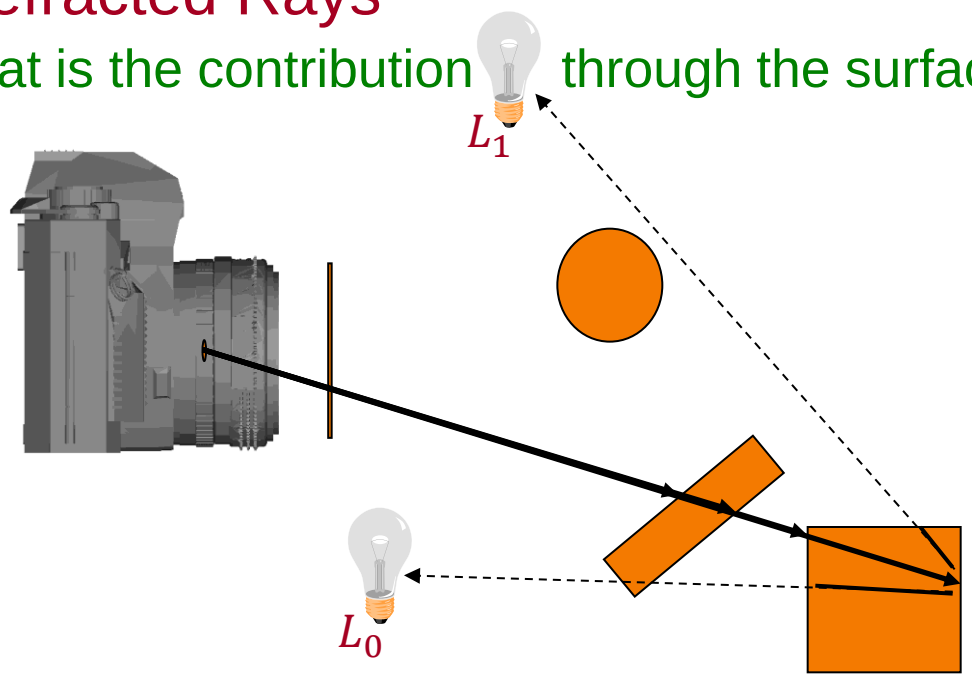
Radiance for refraction ray

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

Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

What is the contribution L_1 through the surface?



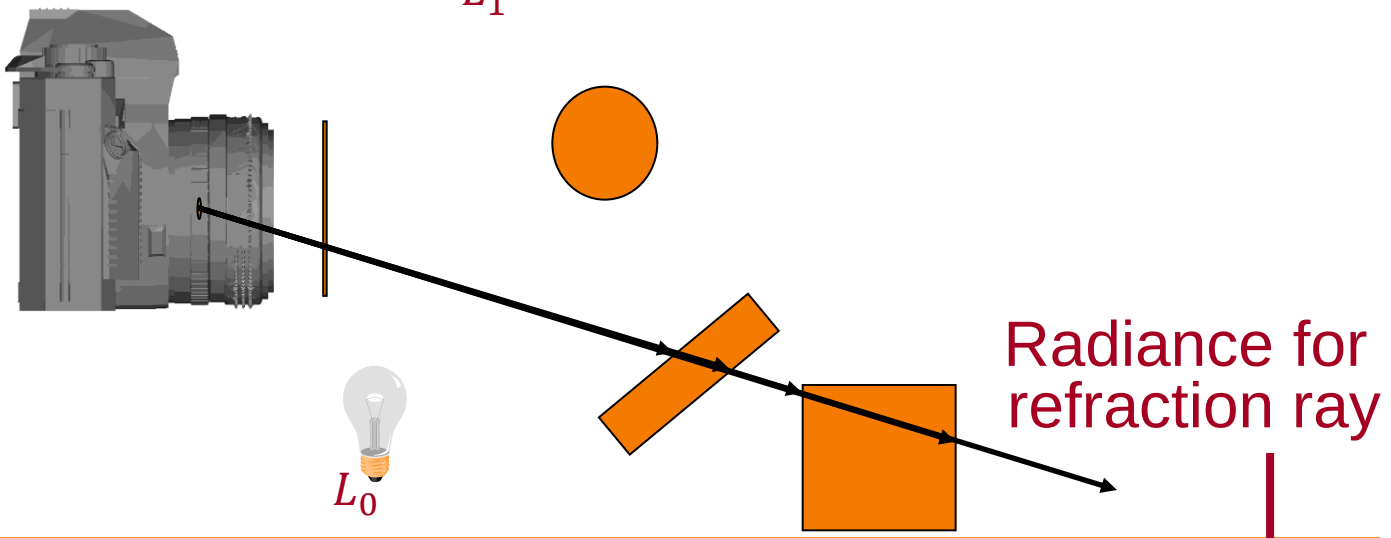
Radiance for refraction ray

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

Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays

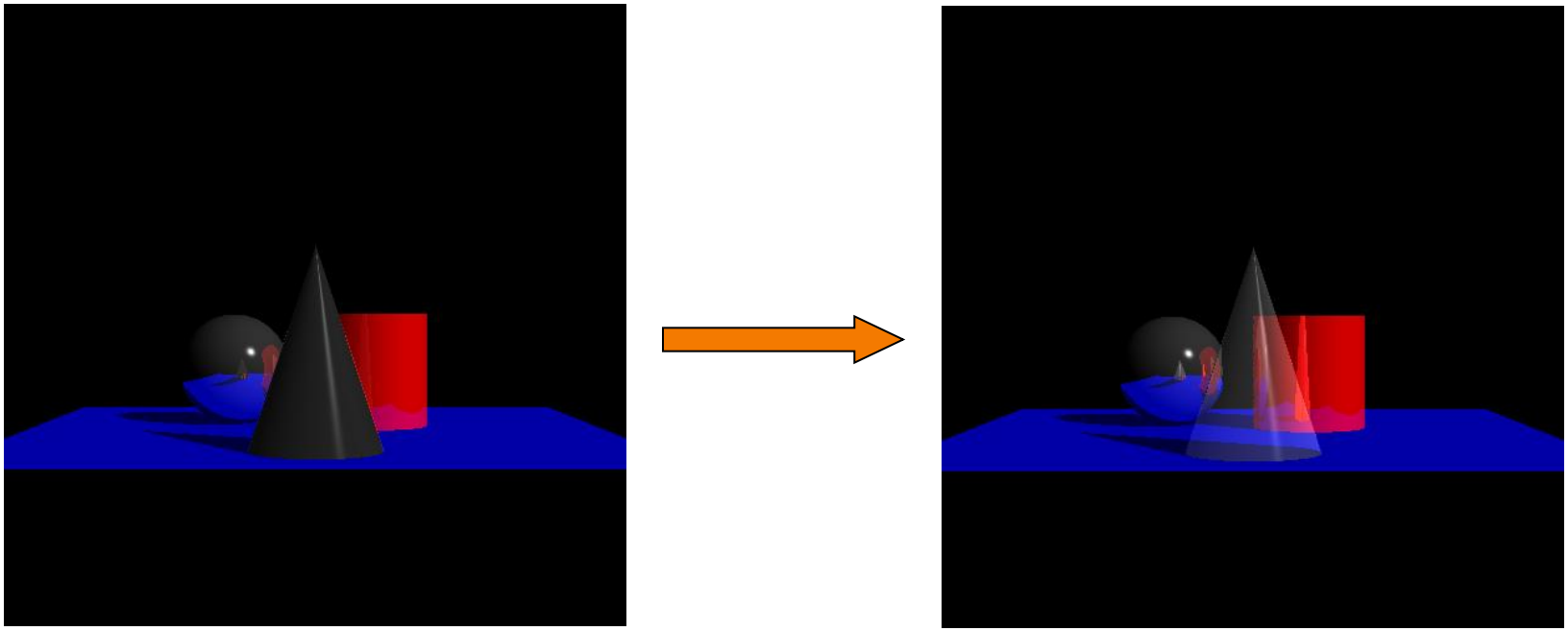
What is the contribution L_1 through the surface?



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

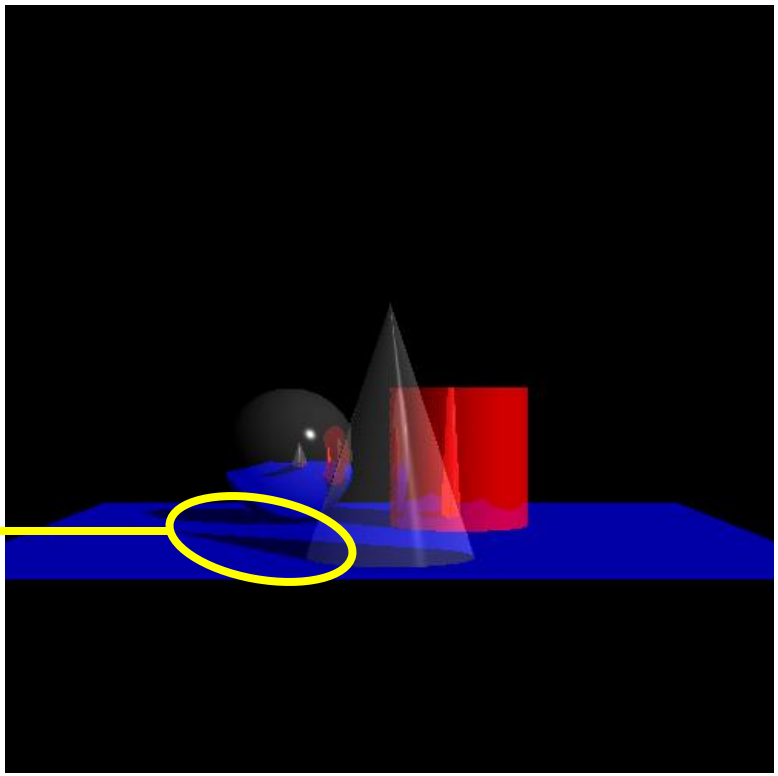
Transparent Refraction

- Also trace secondary rays from hit surfaces
  - Consider contributions from:
    1. Reflected Rays
    2. Refracted Rays



Transparency and Shadow

- Problem:
  - If a surface is transparent, then rays to the light source may pass through the object



Transparency and Shadow

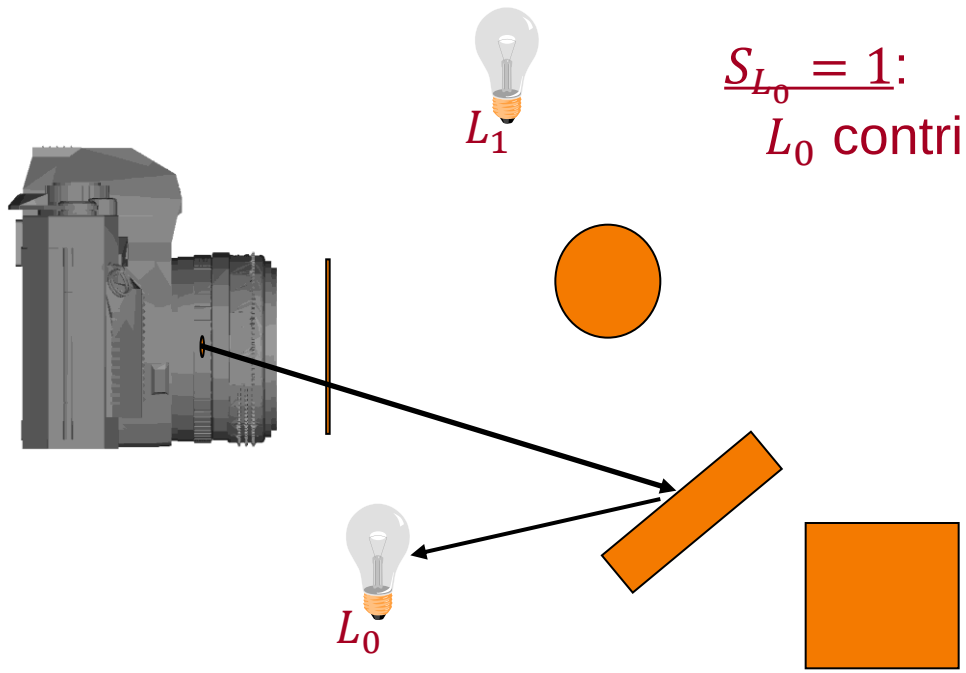
- Problem:
  - If a surface is transparent, then rays to the light source may pass through the object
  - Need to modify the shadow term so that instead of representing a binary (0/1) value, it gives the **proportion** of light passing through.

  ⇒ Accumulate transparency values as the ray travels to the light source.

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

Transparency and Shadow

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

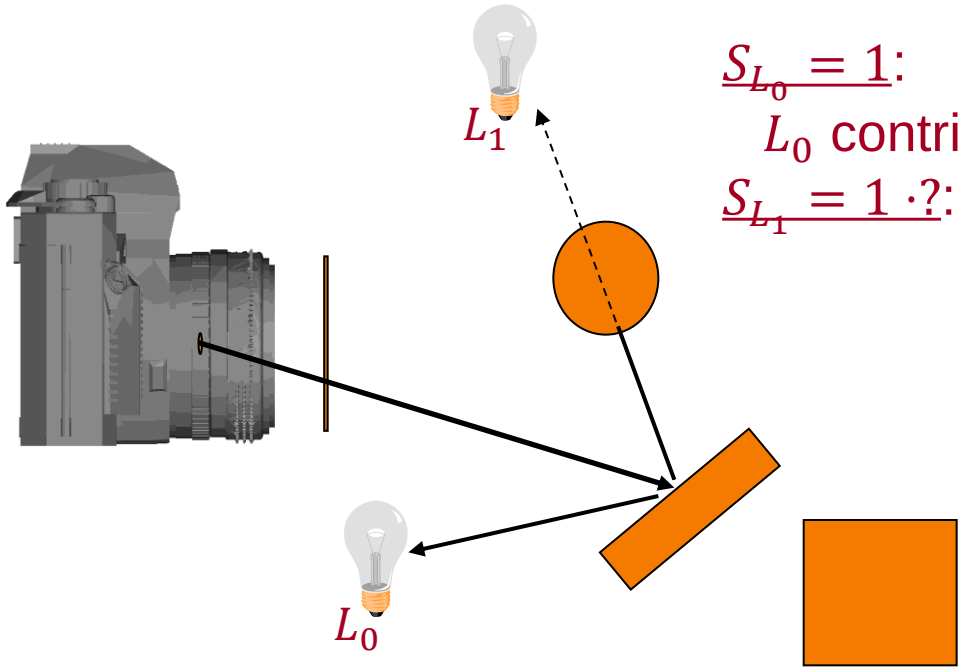


$S_{L_0} = 1$:
L_0 contributes fully

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

Transparency and Shadow

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

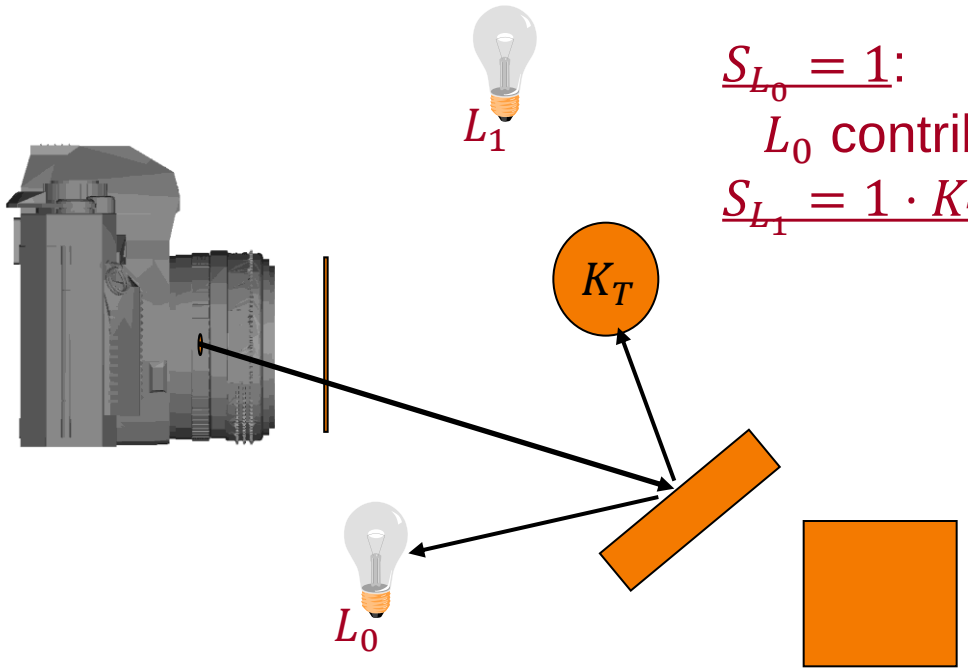


$S_{L_0} = 1$:
L_0 contributes fully
$S_{L_1} = 1 \cdot ?$:

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

Transparency and Shadow

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

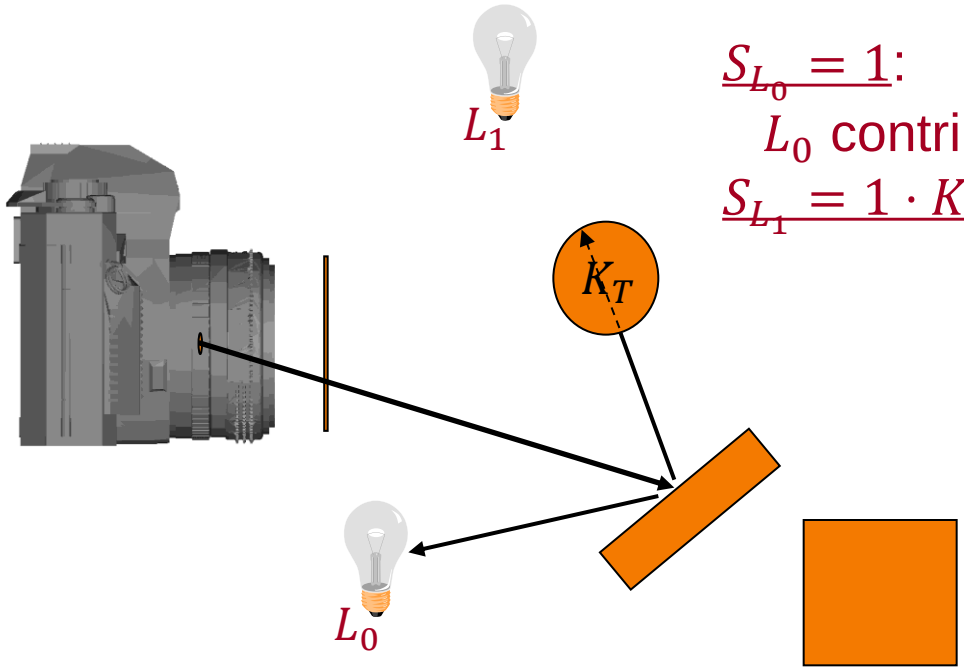


$S_{L_0} = 1$:
L_0 contributes fully
$S_{L_1} = 1 \cdot K_T \cdot ?$:

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

Transparency and Shadow

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

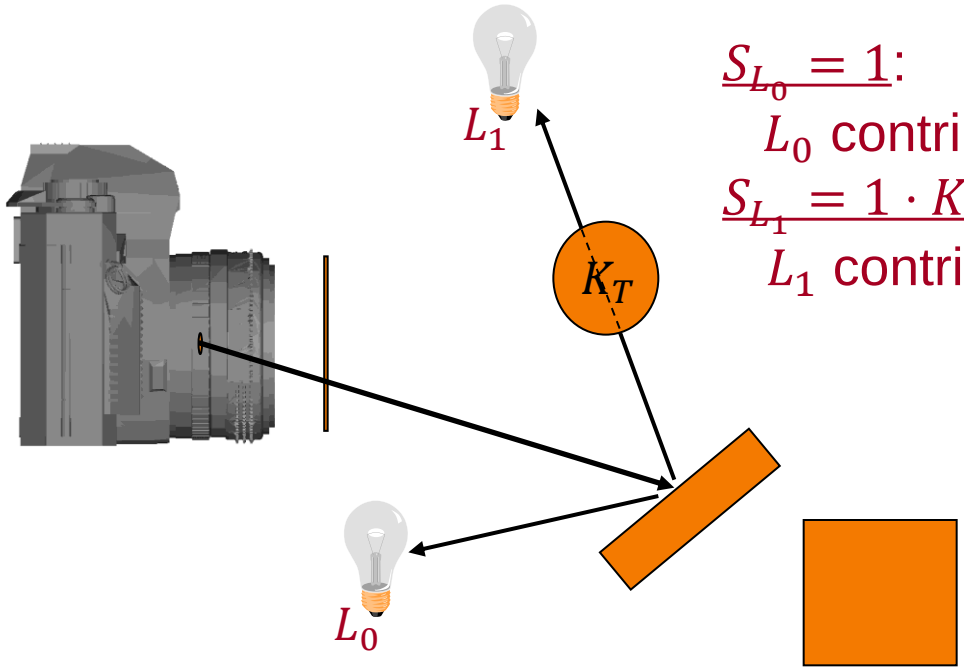


$S_{L_0} = 1$:
L_0 contributes fully
$S_{L_1} = 1 \cdot K_T \cdot K_T \cdot ?$:

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

Transparency and Shadow

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

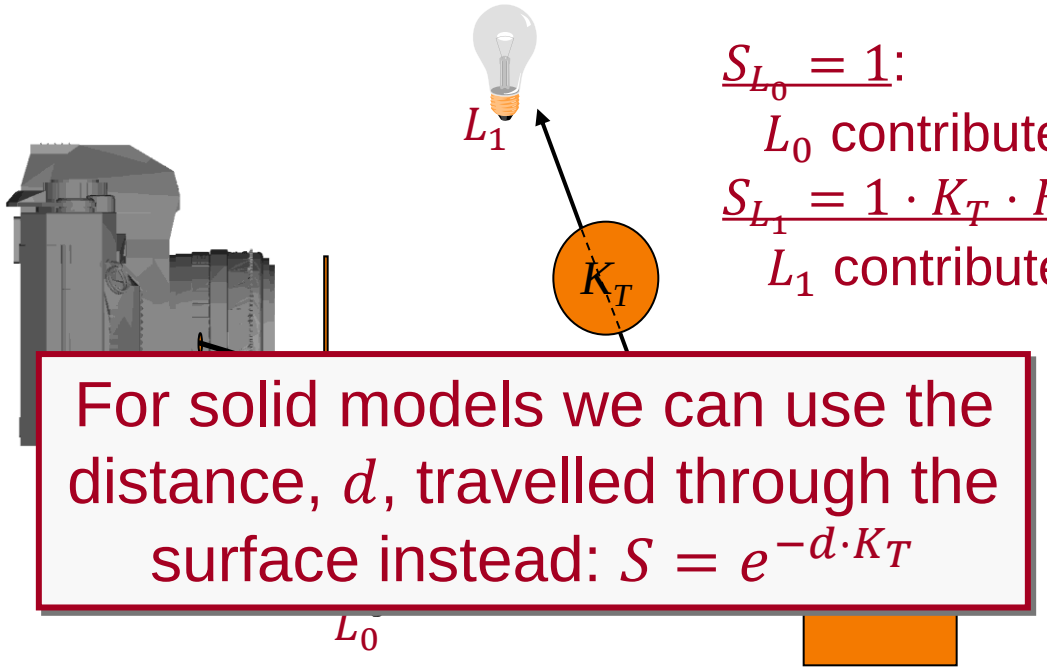


$\underline{S_{L_0} = 1}$:
L_0 contributes fully
$\underline{S_{L_1} = 1 \cdot K_T \cdot K_T}$:
L_1 contributes partially

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

Transparency and Shadow

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



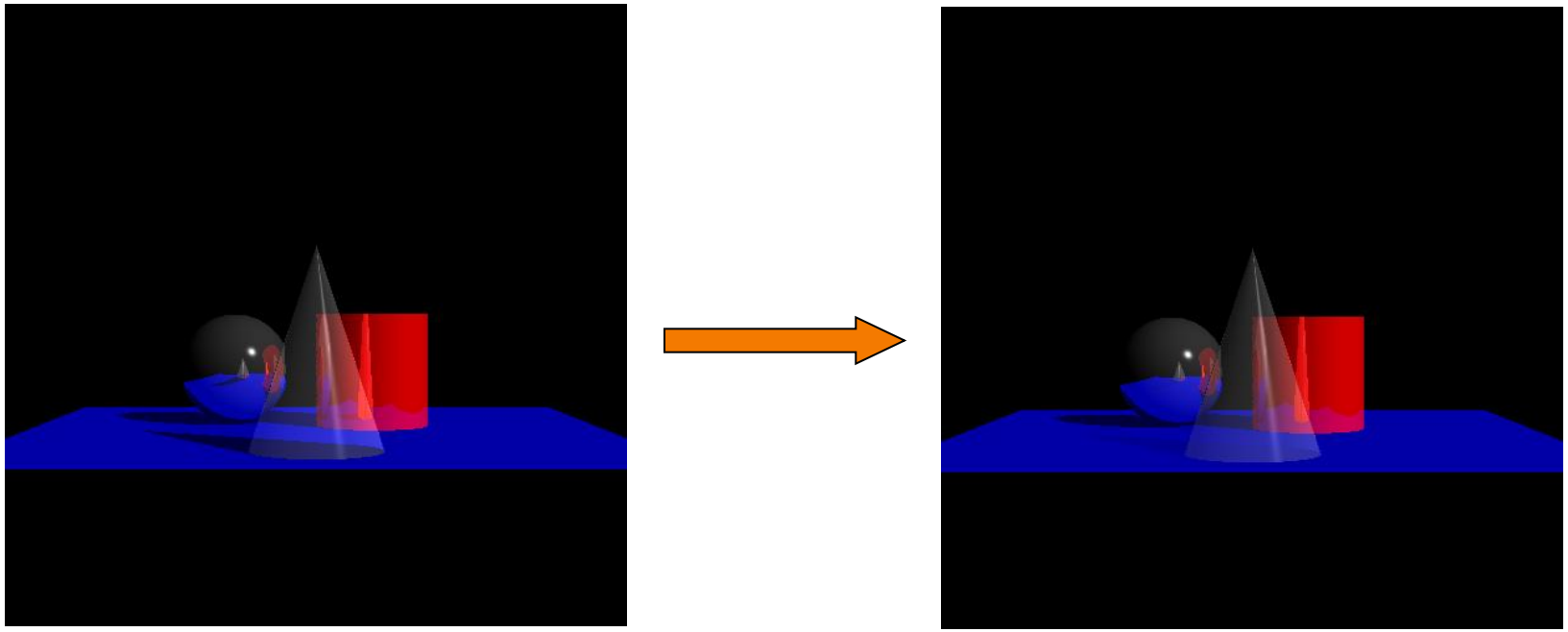
$S_{L_0} = 1$:
L_0 contributes fully
$S_{L_1} = 1 \cdot K_T \cdot K_T$:
L_1 contributes partially

For solid models we can use the distance, d, travelled through the surface instead: $S = e^{-d \cdot K_T}$

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

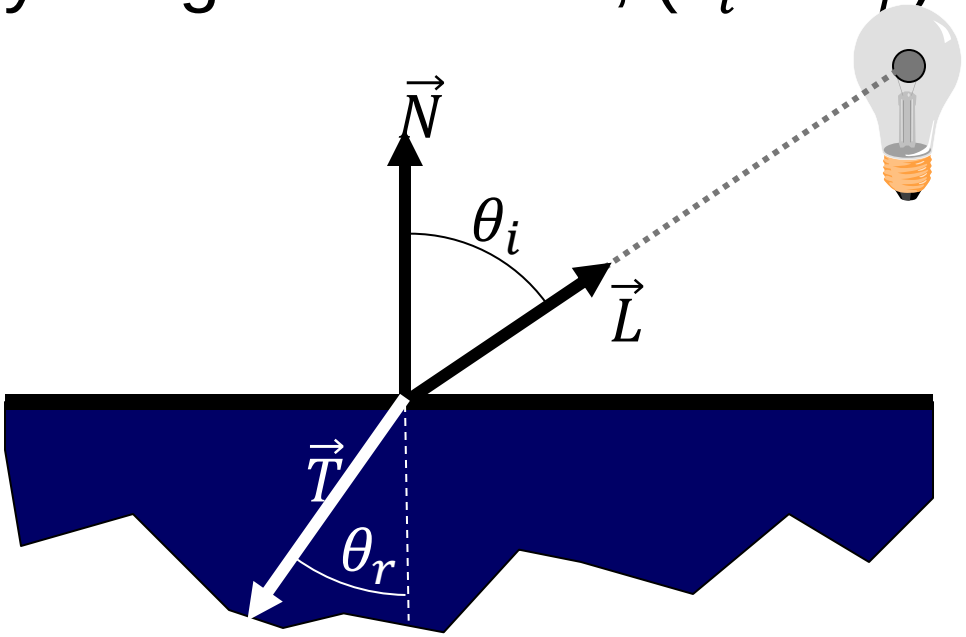
Transparency and Shadow

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



Transparent Refraction

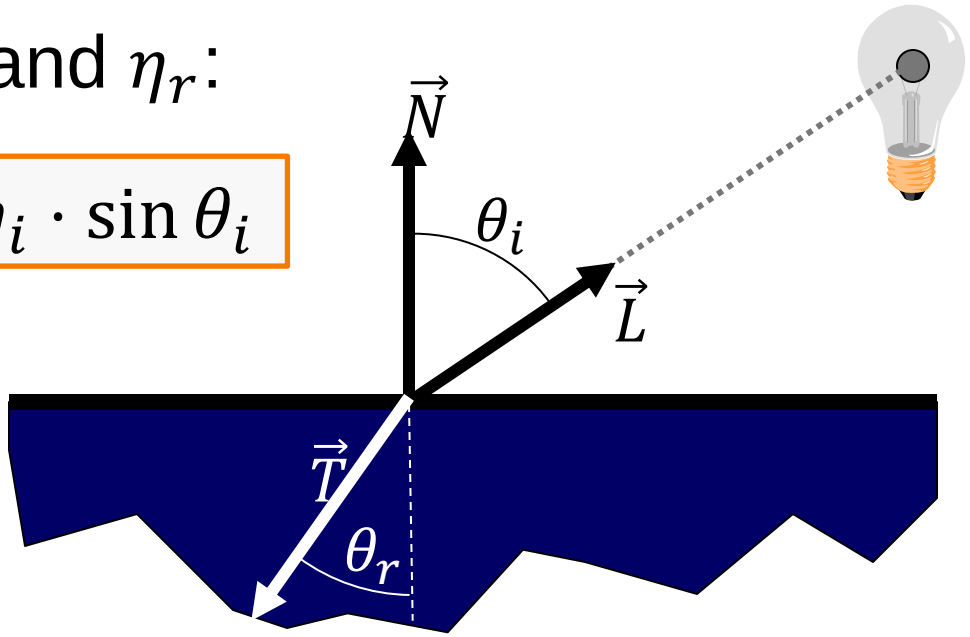
- When a light of light passes through a transparent object, the ray of light can bend, ($\theta_i \neq \theta_r$).



Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i$$

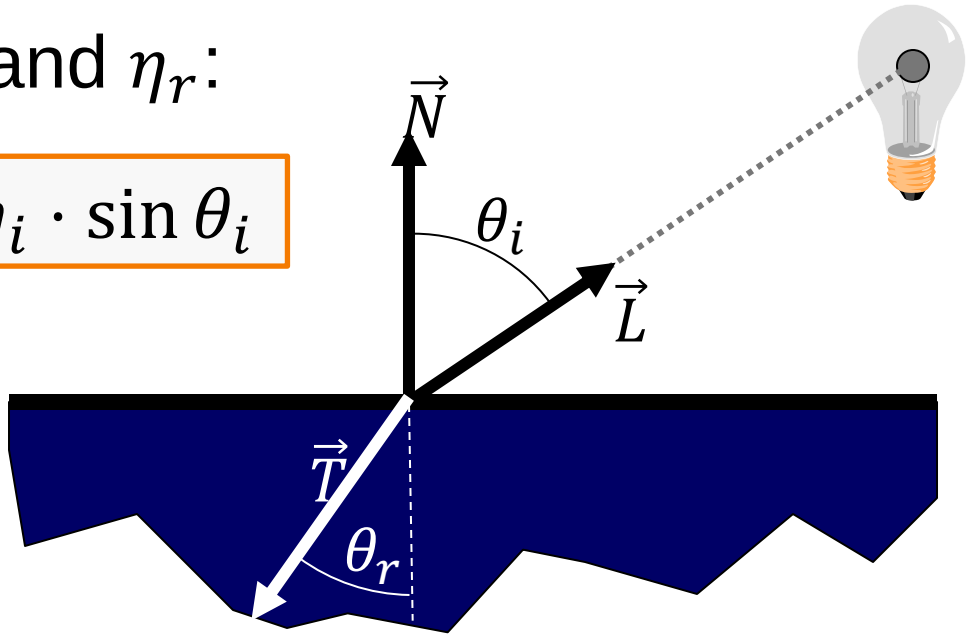


The index of refraction of air is $\eta = 1$.

Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i$$



Note (Critical Angle):

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

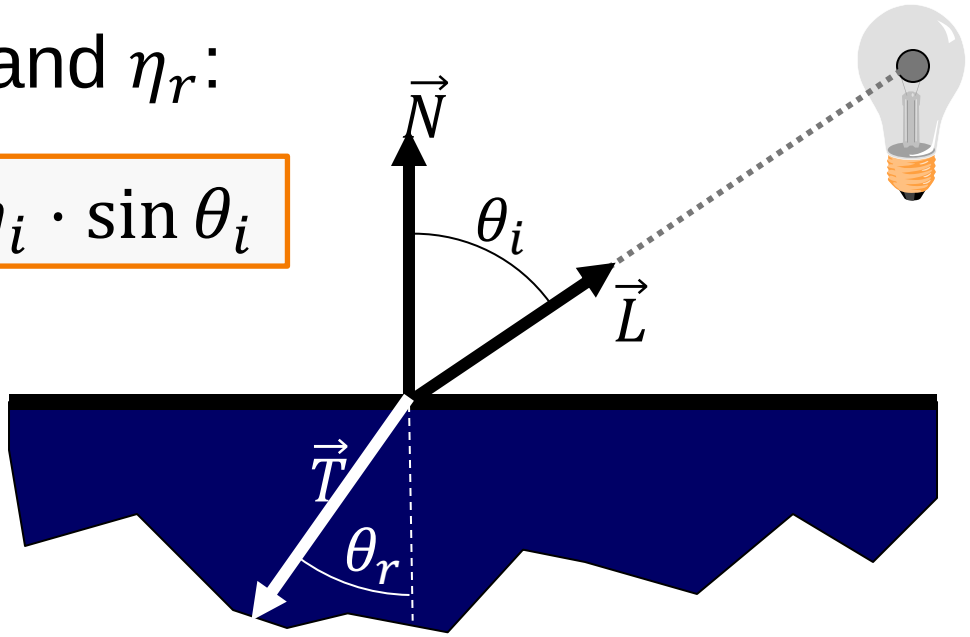
In this case:

- There is no value of θ_r such that $\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i$.
- The light reflects off the surface and does not pass through.

Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i$$

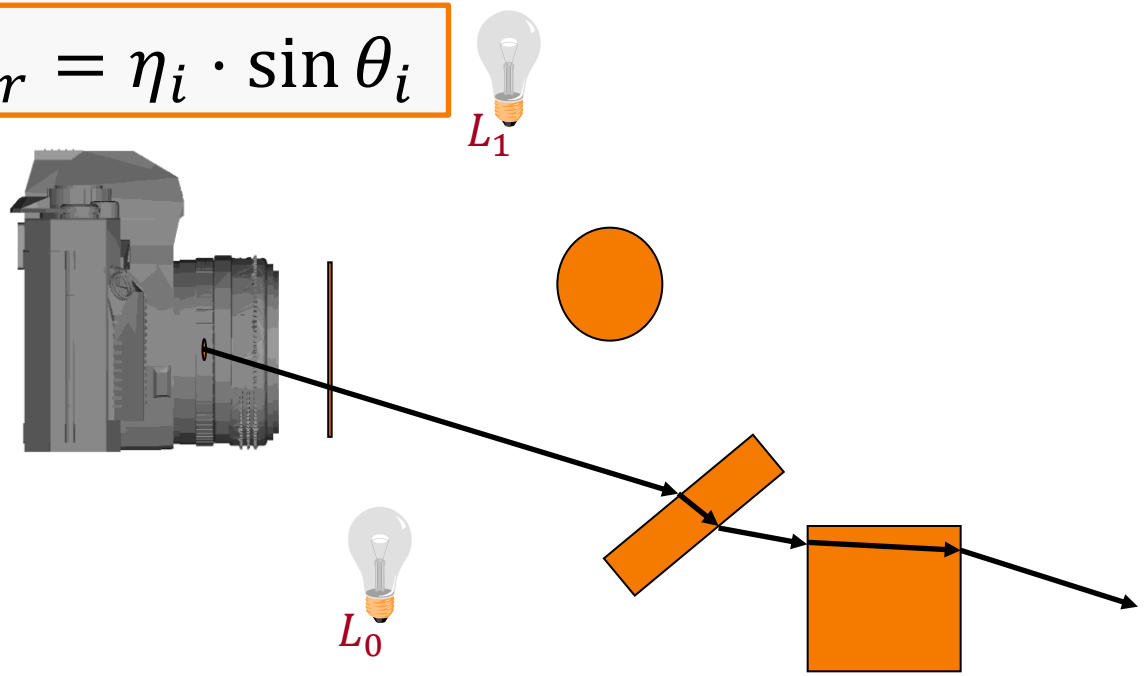


$$\vec{T} = \left(\frac{\eta_i}{\eta_r} \cdot \cos\theta_i - \cos\theta_r\right) \cdot \vec{N} - \frac{\eta_i}{\eta_r} \cdot \vec{L}$$

Snell's Law

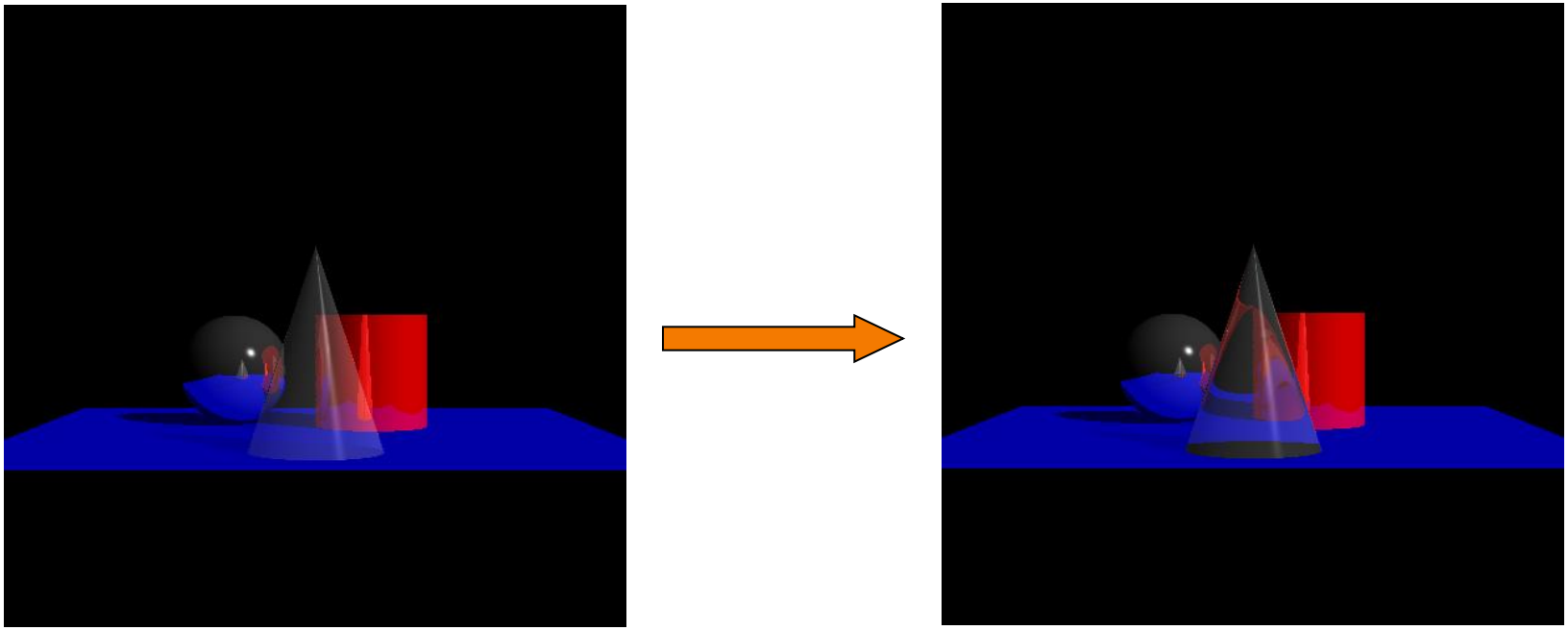
- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

$$\eta_r \cdot \sin\theta_r = \eta_i \cdot \sin\theta_i$$



Snell's Law

- The way that light bends is determined by the indices of refraction of the internal and external materials η_i and η_r:

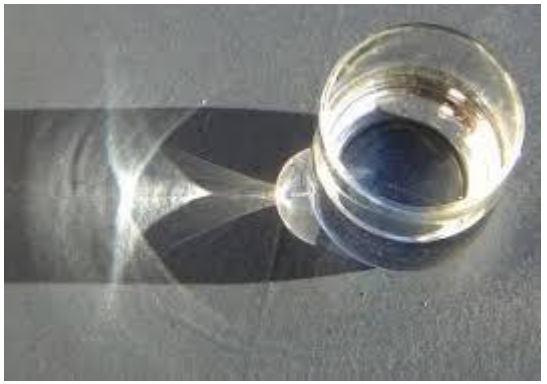


Snell's Law and Caustics

- Challenge:
  - If a surface is transparent, then rays to the light source may 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

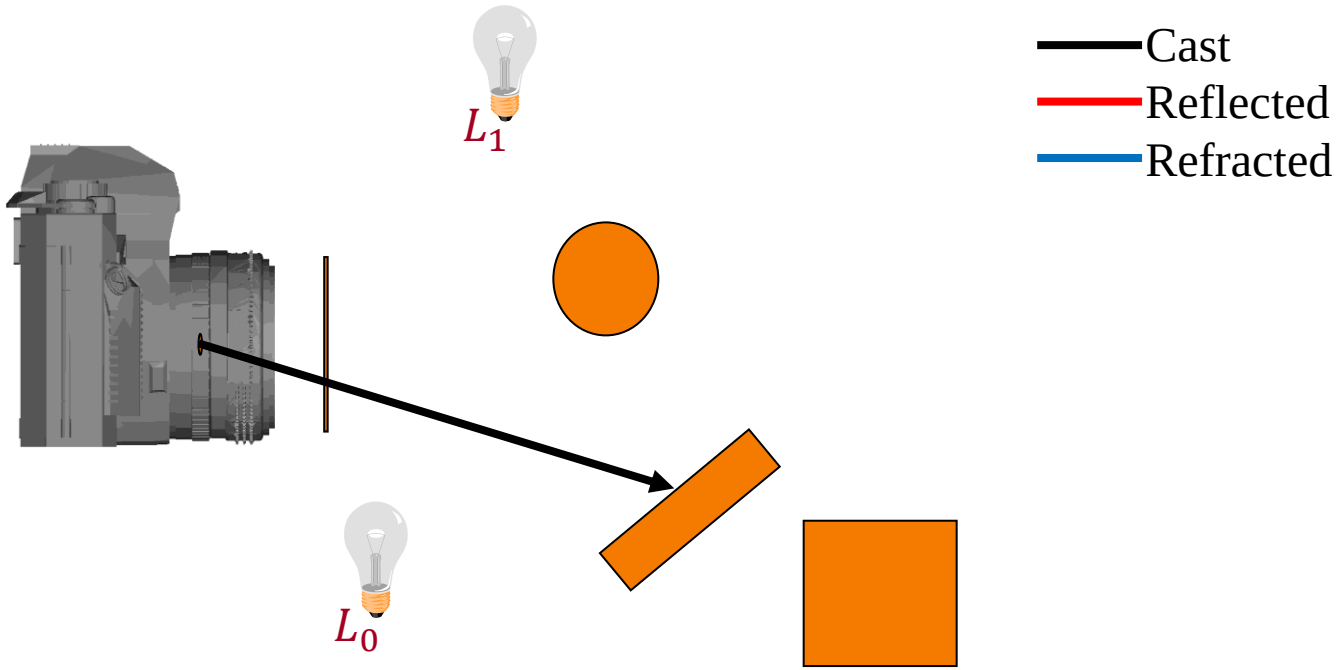
  ⇒ Summing the intensity contribution from all of these directions/paths we get bright caustics
  - This is difficult to address with ray-tracing



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

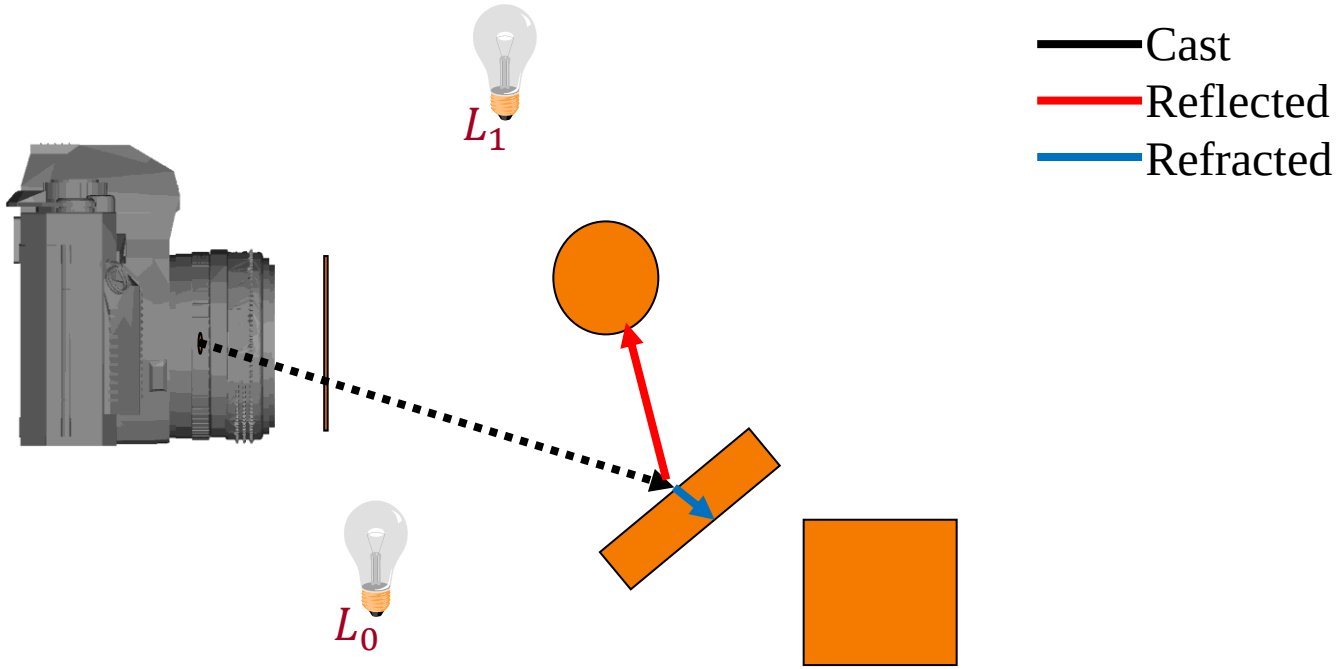
(Rough) Complexity Analysis

- Simulating reflection and refraction in the ray tracer, a ray splits into two at each bounce



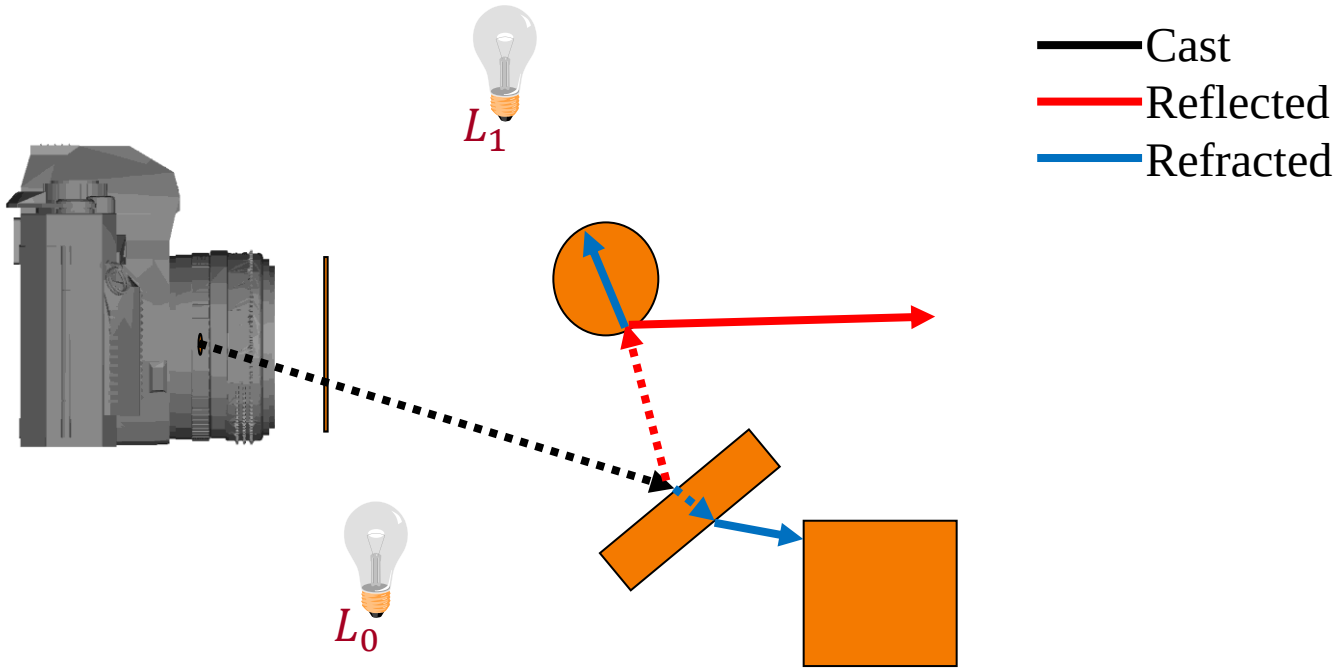
(Rough) Complexity Analysis

- Simulating reflection and refraction in the ray tracer, a ray splits into two at each bounce



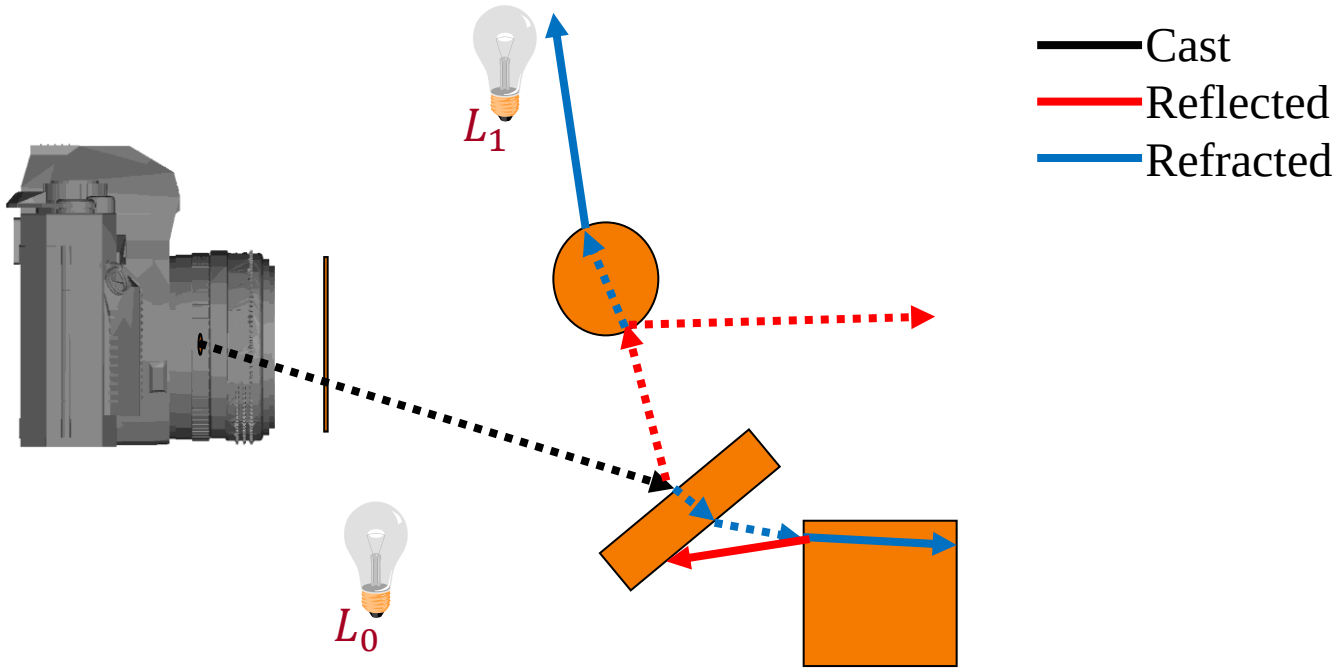
(Rough) Complexity Analysis

- Simulating reflection and refraction in the ray tracer, a ray splits into two at each bounce



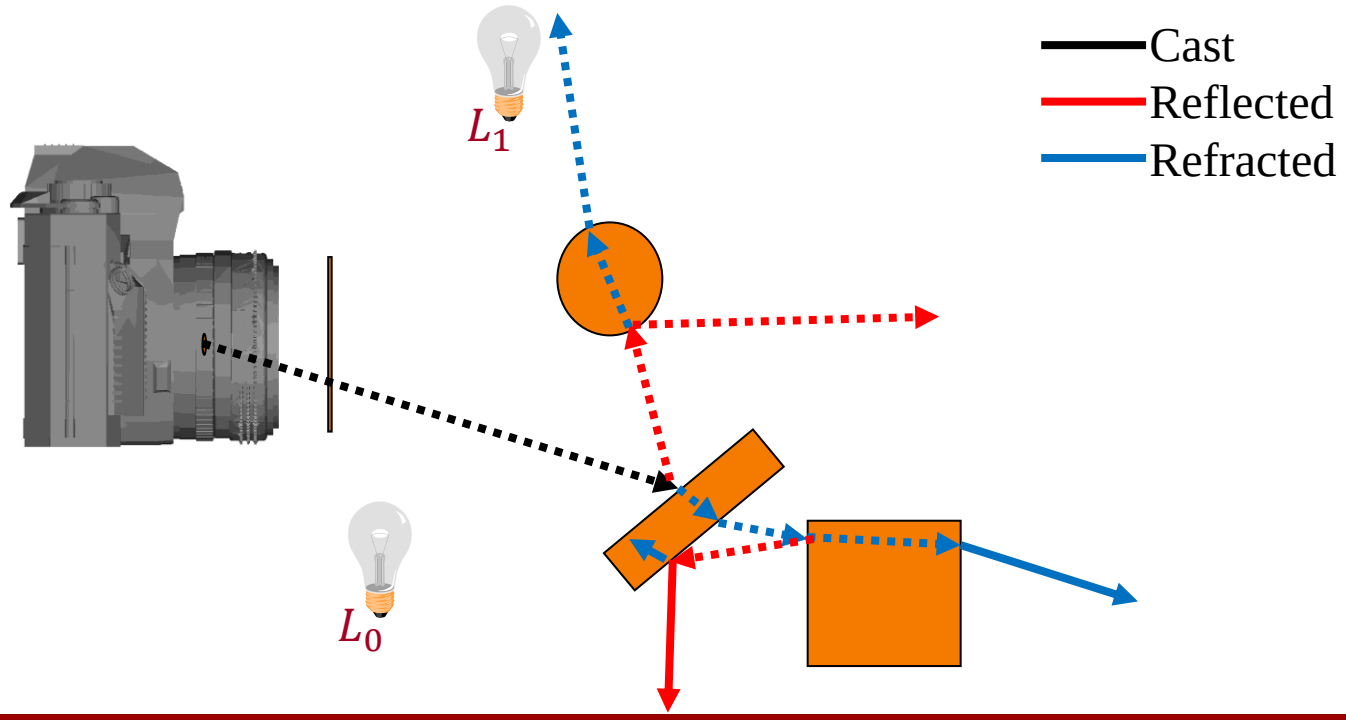
(Rough) Complexity Analysis

- Simulating reflection and refraction in the ray tracer, a ray splits into two at each bounce



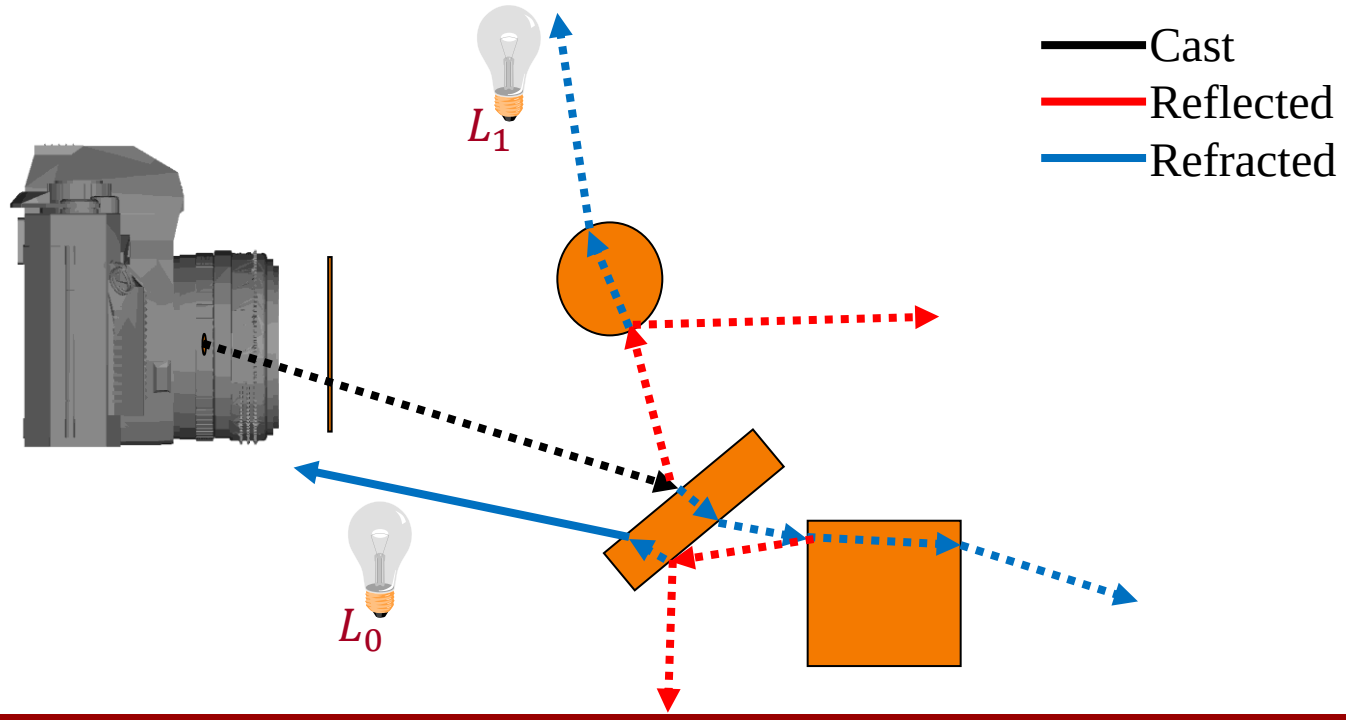
(Rough) Complexity Analysis

- Simulating reflection and refraction in the ray tracer, a ray splits into two at each bounce



(Rough) Complexity Analysis

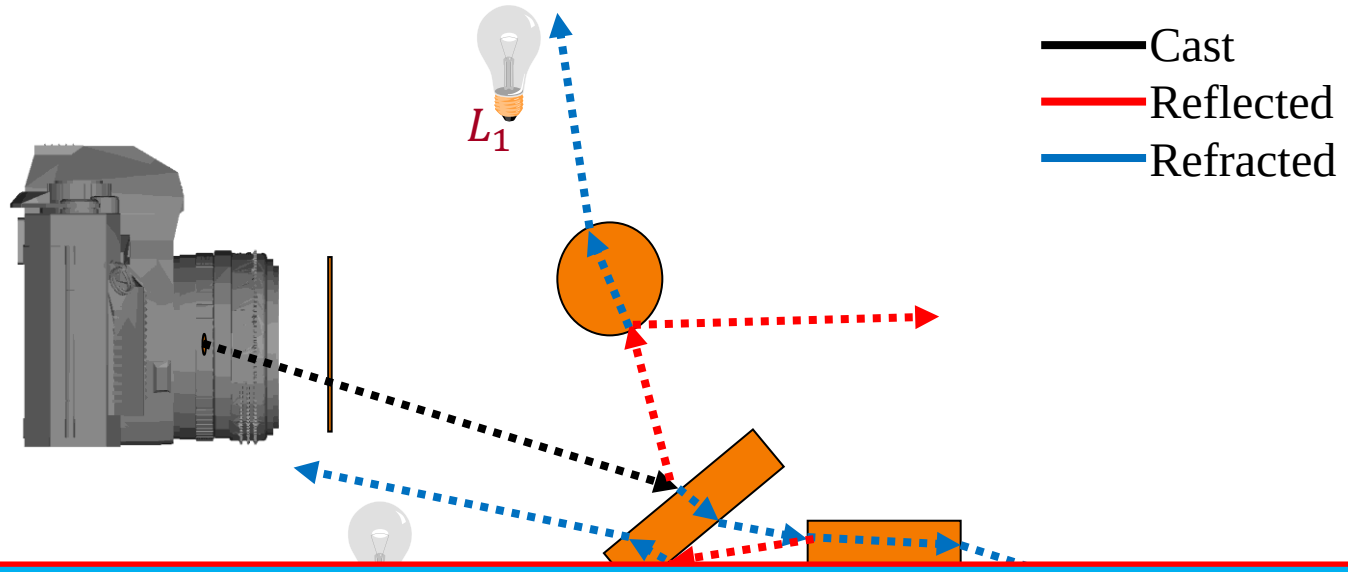
- Simulating reflection and refraction in the ray tracer, a ray splits into two at each bounce



(Rough) Complexity Analysis

- Simulating reflection and refraction in the ray tracer, a ray splits into two at each bounce

⇒ <u>Exponential</u> growth in the number of rays as a function of the number of bounces



Note that we do not cast reflected rays from the interior of a shape.
(i.e. if the dot-product of the normal with the ray direction is positive)

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 )
		{
			reflect.direction = Reflect( ray.direction , hit.normal );
			reflect.position = hit.position + reflect.direction*ε;
			c += GetColor( scene , reflect , ir )*hit.kSpec;
		}

		refract.direction = Refract( ray.direction , hit.normal , ir , hit.ir );
		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

```
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 )
		{
			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;
		}

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

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;

}
```

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;
	Intersection hit;

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

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

```
}
```

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;
	Intersection hit;

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

		if( Dot( ray.direction , hit.normal )<0 )
		{
			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;
		}
	}
}
```

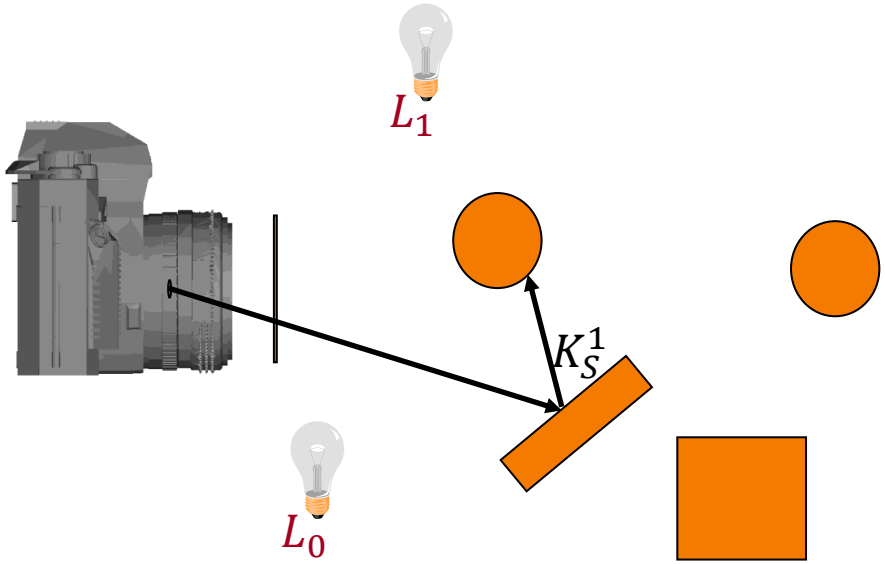
$$I = I_E + \sum_L \left[K_A \cdot I_L^A + \left(K_D \cdot \langle \vec{N}, \vec{L} \rangle + K_S \cdot \langle \vec{V}, \vec{R} \rangle^n \right) \cdot I_L \cdot S_L \right] + K_S \cdot I_R + K_T \cdot I_T$$

Termination Criteria

```
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.

$\Rightarrow K_S^1 >$ cut-off



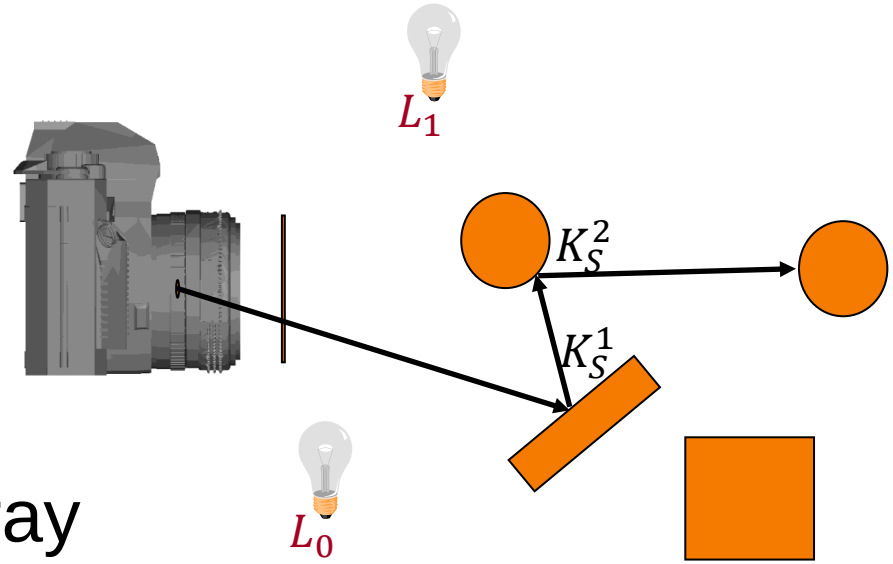
Termination Criteria

```
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

```
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 )
		{
			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;
		}

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

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

Termination Criteria

```
Color GetColor( Scene scene , Ray< 3 > ray , float ir , int rDepth , Color cutOff )
{
	Color c(0,0,0);
	Ray< 3 > reflect ,
	if( !rDepth || ( c                                                    urn c;
	Intersection hit;

	if( FindIntersec
	{
		c += GetSu

		if( Dot(
		{
			ref
			reflect.position = hit.position + reflect.direction*ε;
			c += GetColor( scene , reflect , ir , rDepth-1 , cutOff/hit.kSpec )*hit.kSpec;
		}

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

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

To ensure that the new ray does not hit its starting location!

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

Termination Criteria

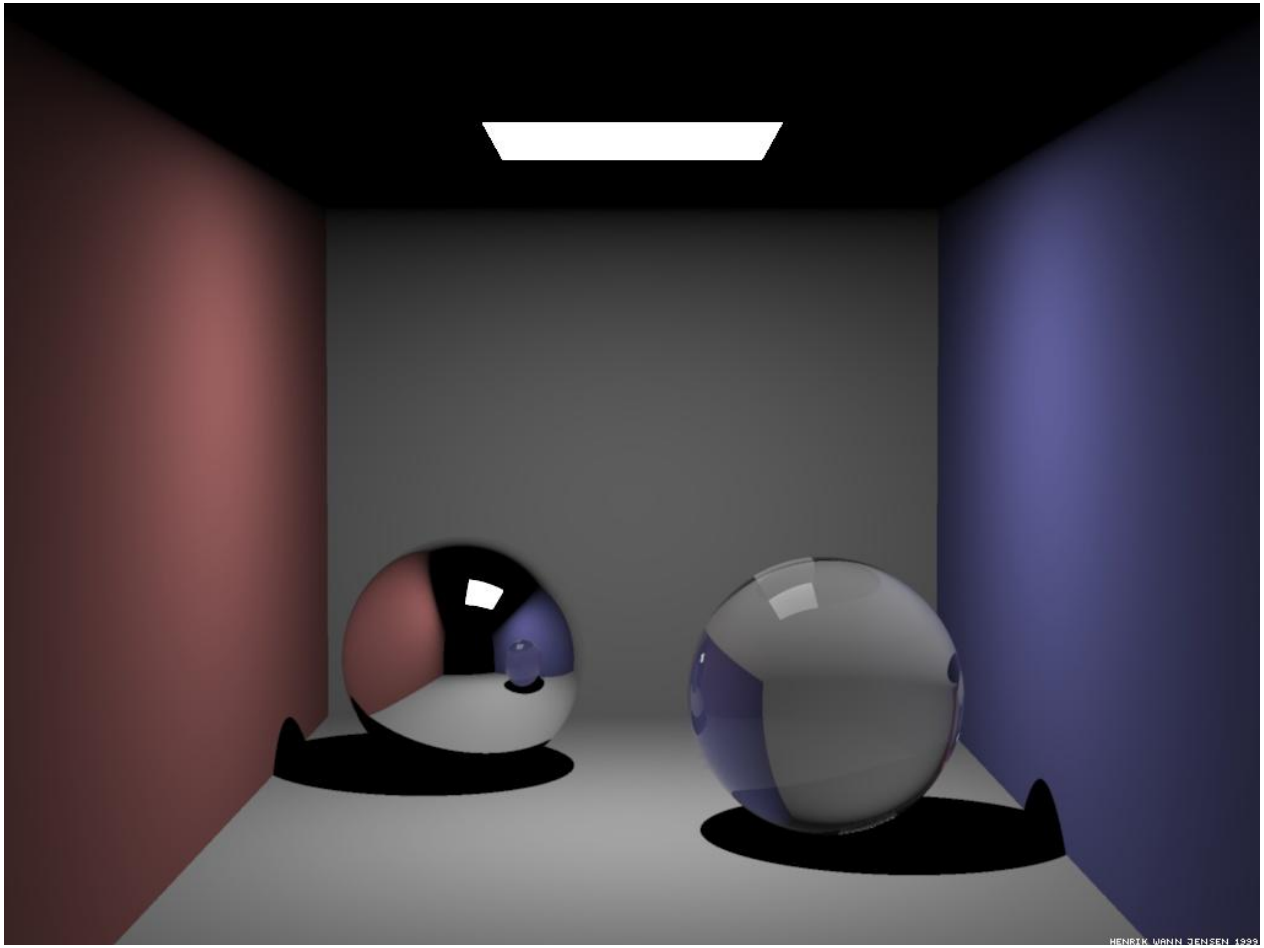
```
Color GetColor( Scene scene , Ray< 3 > ray , float ir , int rDepth , Color cutOff )
{
    Color c(0,0,0);
    Ray< 3 > reflect , refract;
    if
    I
    if
                c += GetColor( scene , reflect , ir , rDepth-1 , cutOff/hit.kSpec )*hit.kSpec;
            }

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

Note:
When we refract, we enter a new material and the associated index of refraction changes.

For simplicity, we assume that the interface will always be between a solid and air:
⇒ One of the indices of refraction is always equal to 1.
Which one depends on the alignment of the ray with the normal.

Illumination Examples

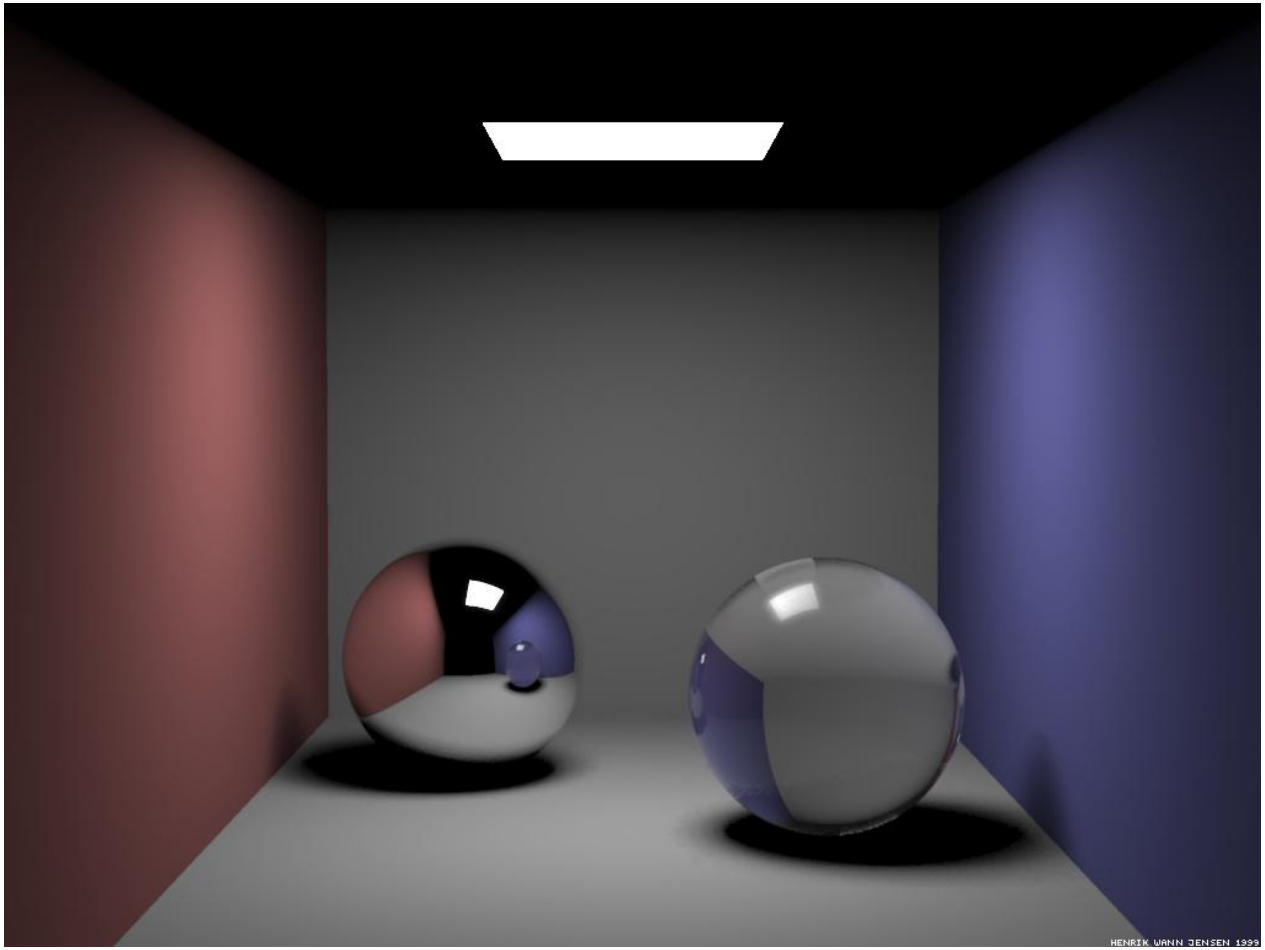
- Ray tracing (rays to point light source)



Courtesy Henrik Wann Jensen

Illumination Examples

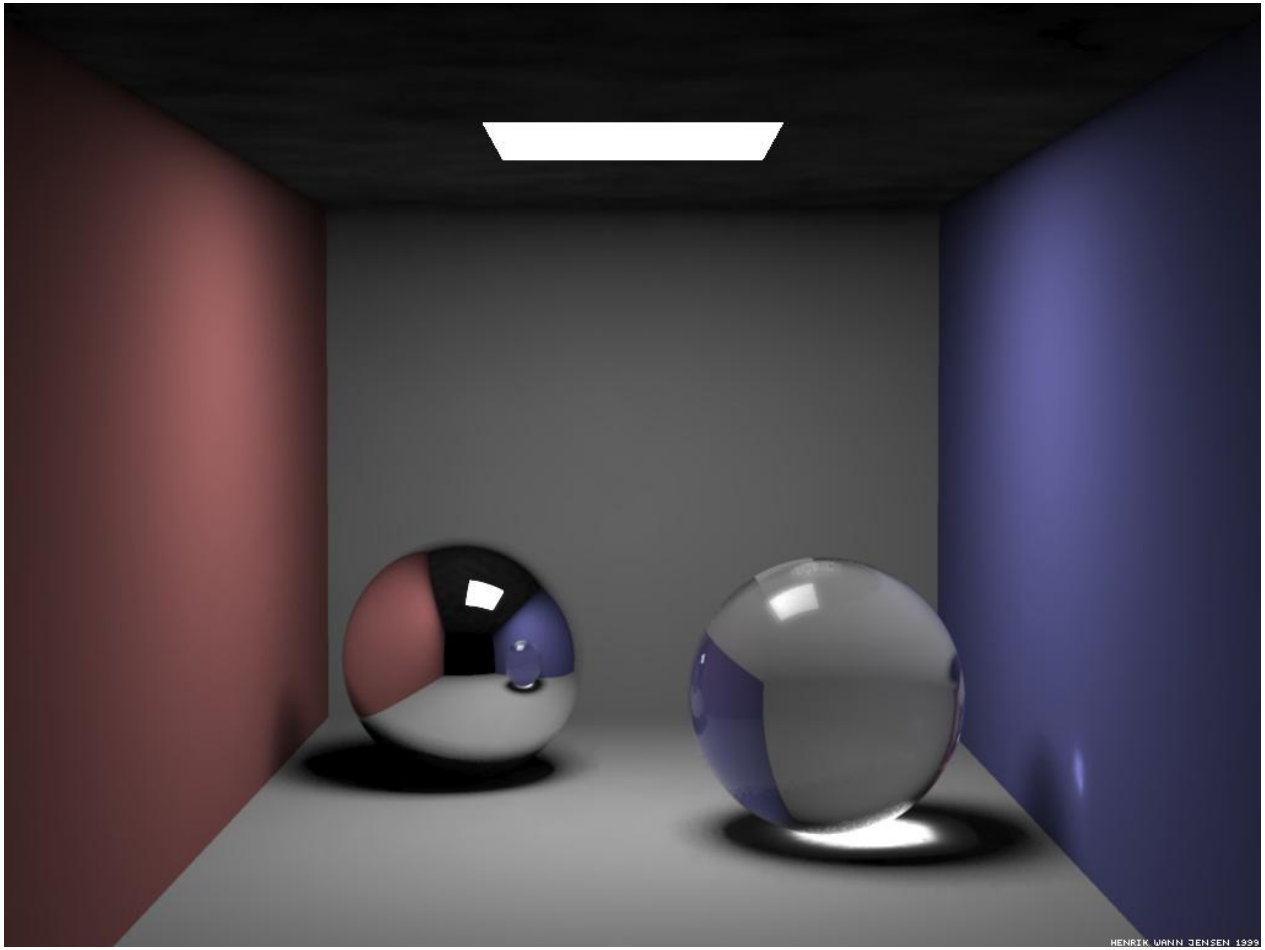
- Soft shadows (rays to area light source)



Courtesy Henrik Wann Jensen

Illumination Examples

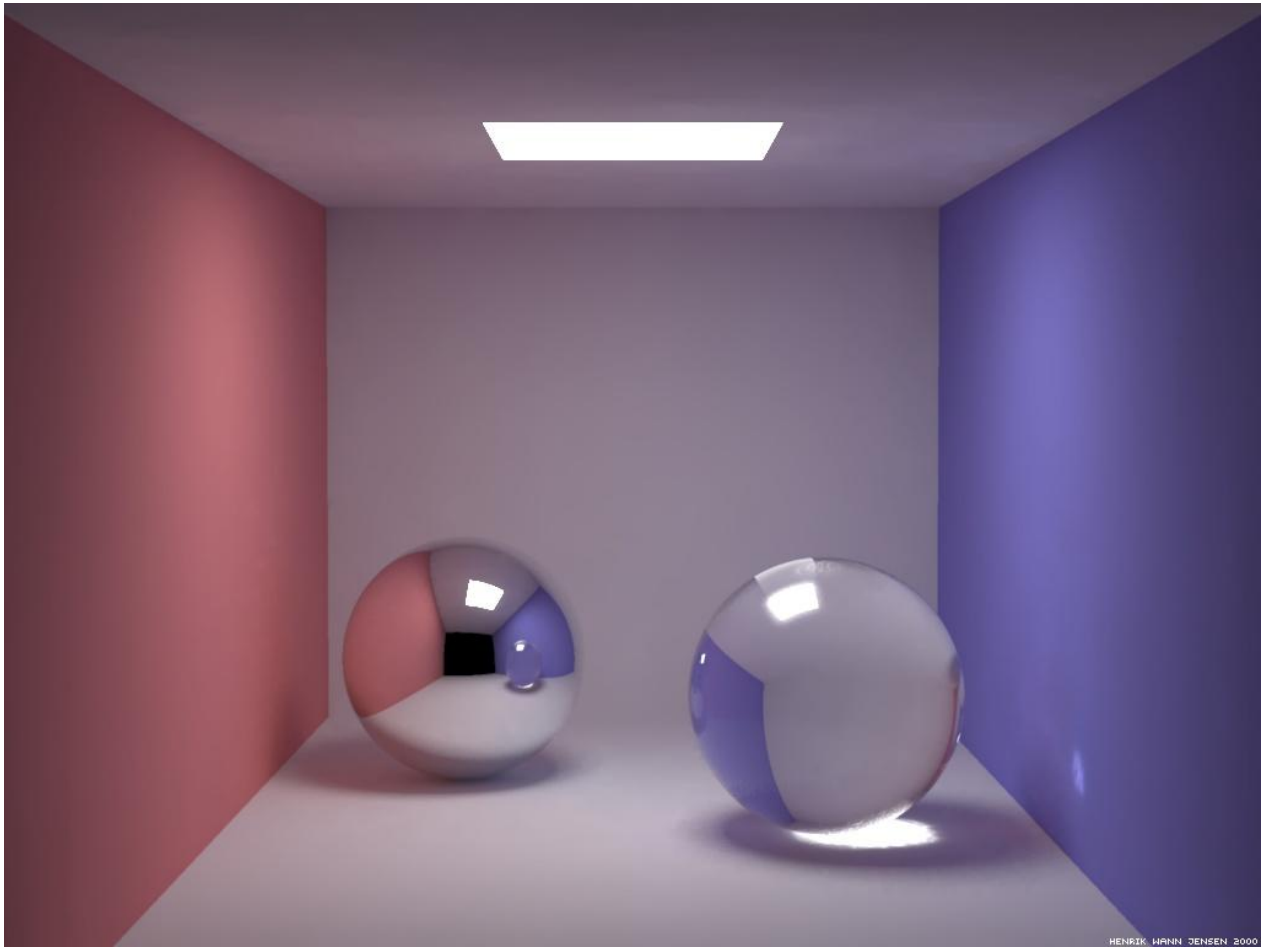
- Caustics (rays from light source)



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 pure refractions

All of this is an approximation so that it is practical to compute

More on global illumination later!