

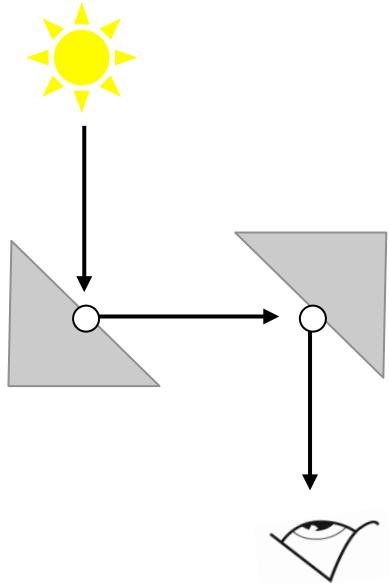
Ray-tracing

Tobias Ritschel

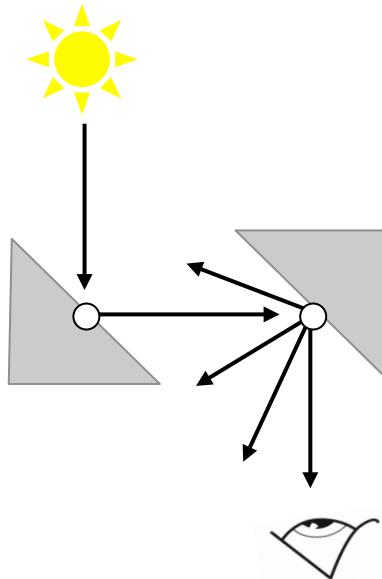
Overview

- Recursive ray tracing
- Shadow tests
- Snell's law for refraction
- When to stop!

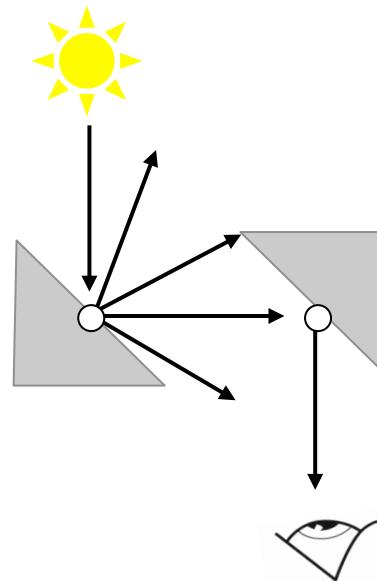
Recap: Light Transport



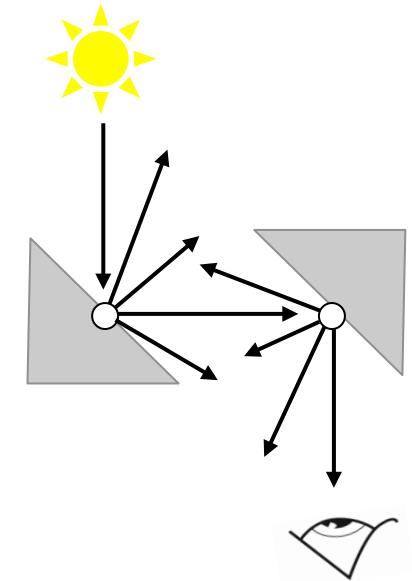
Specular-specular



Diffuse-specular



Specular-diffuse



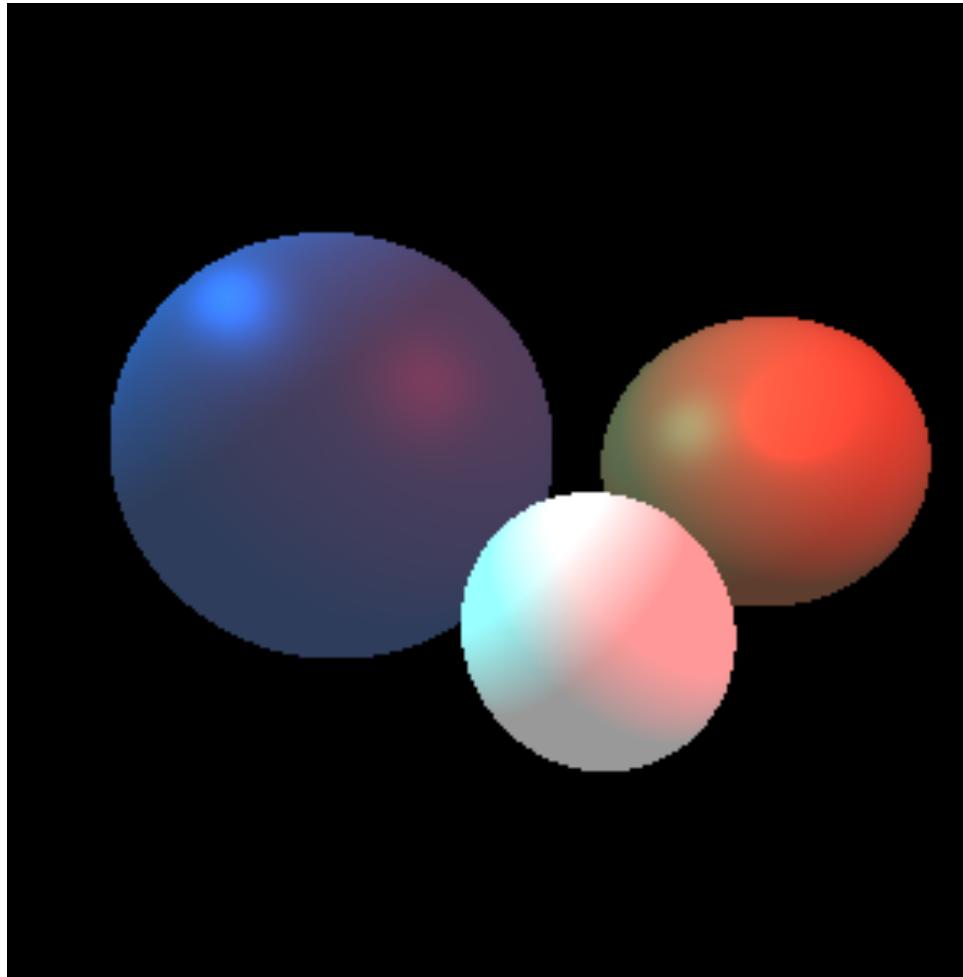
Diffuse-diffuse

Recap: Local Illumination

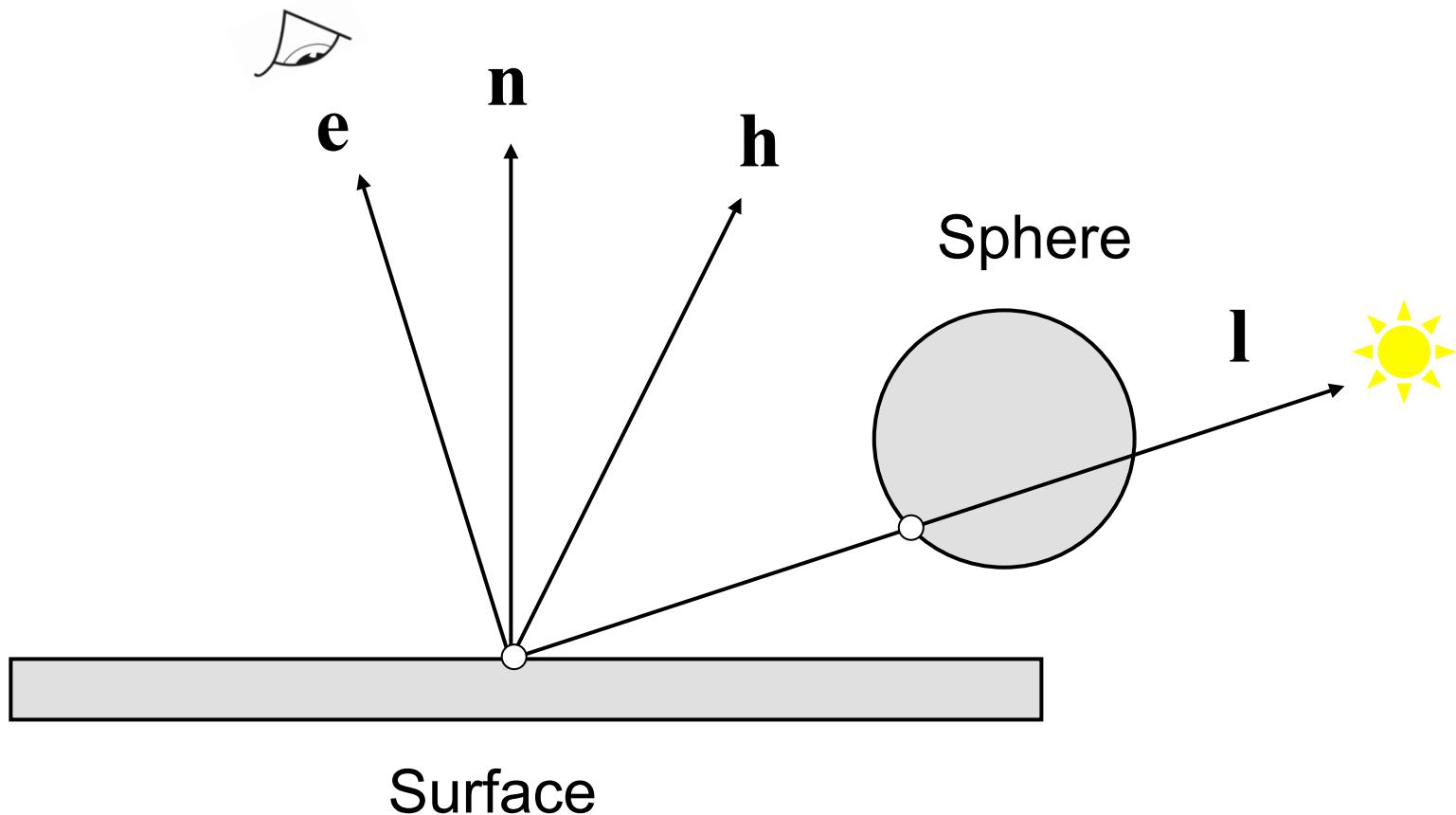
$$I_r = k_a I_a + I_i (k_d \langle \mathbf{n}, \mathbf{l} \rangle^+ + k_s (\langle \mathbf{h}, \mathbf{n} \rangle^+)^m)$$

- Ambient, diffuse & specular components
- Sum over multiple lights

Recap: Result of Ray Casting



Shadows

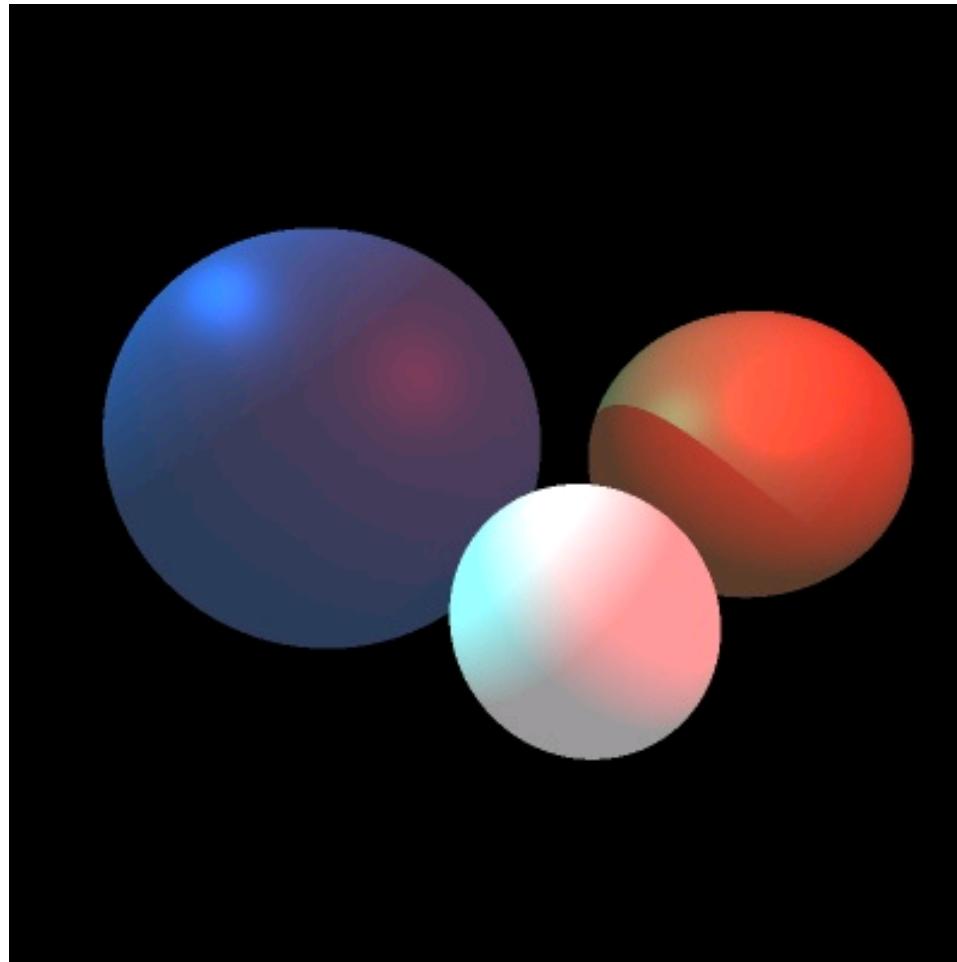


Illumination with shadows

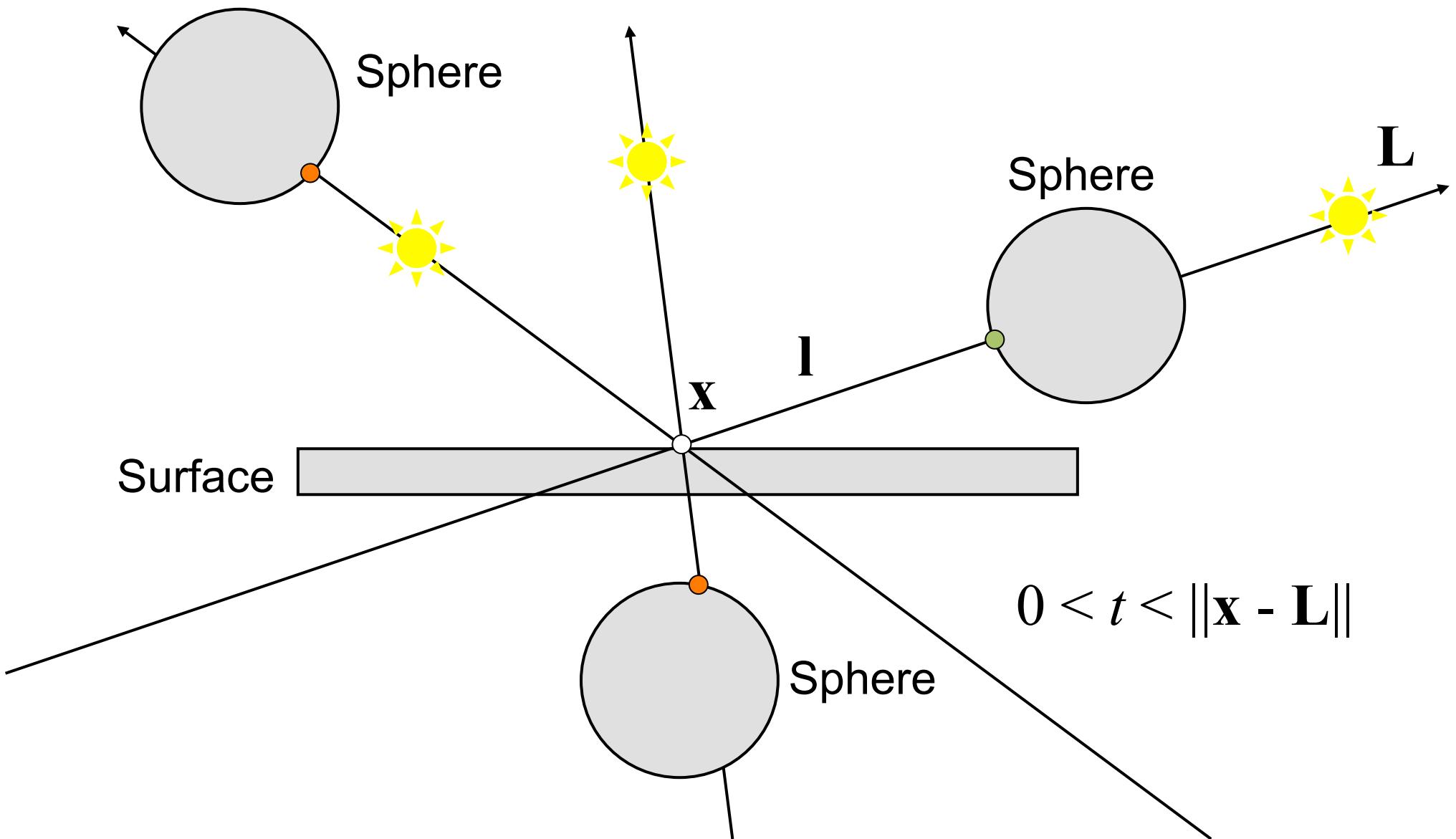
$$I_r = k_a I_a + v_i I_i (k_d \langle \mathbf{n}, \mathbf{l} \rangle^+ + k_s (\langle \mathbf{h}, \mathbf{n} \rangle^+)^m)$$

- Where v_i is the result of intersecting the ray \mathbf{x}, \mathbf{l} with the scene
- v_i is the **visibility** of light i

Result of visibility tests



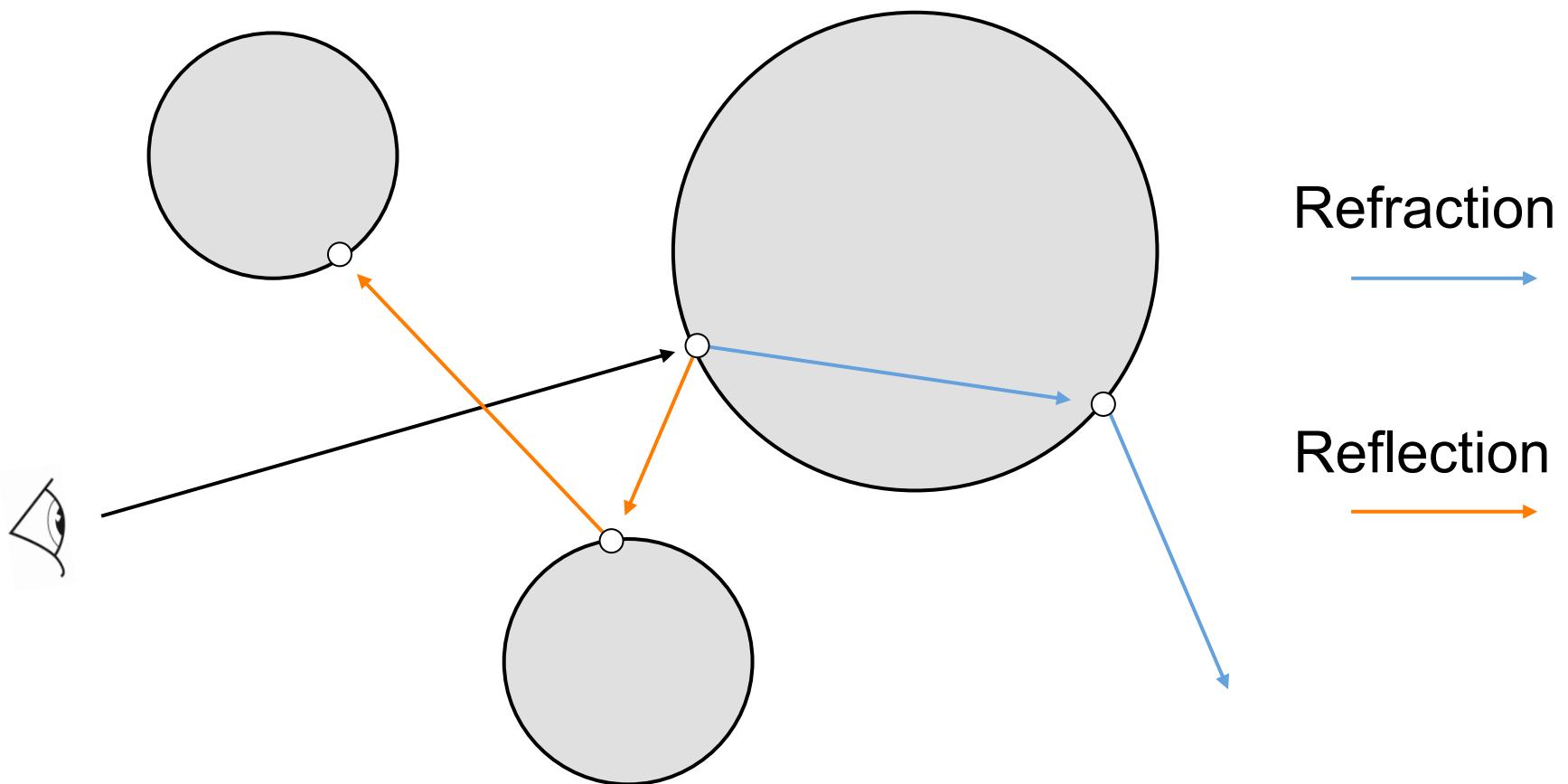
Shadow test correctness



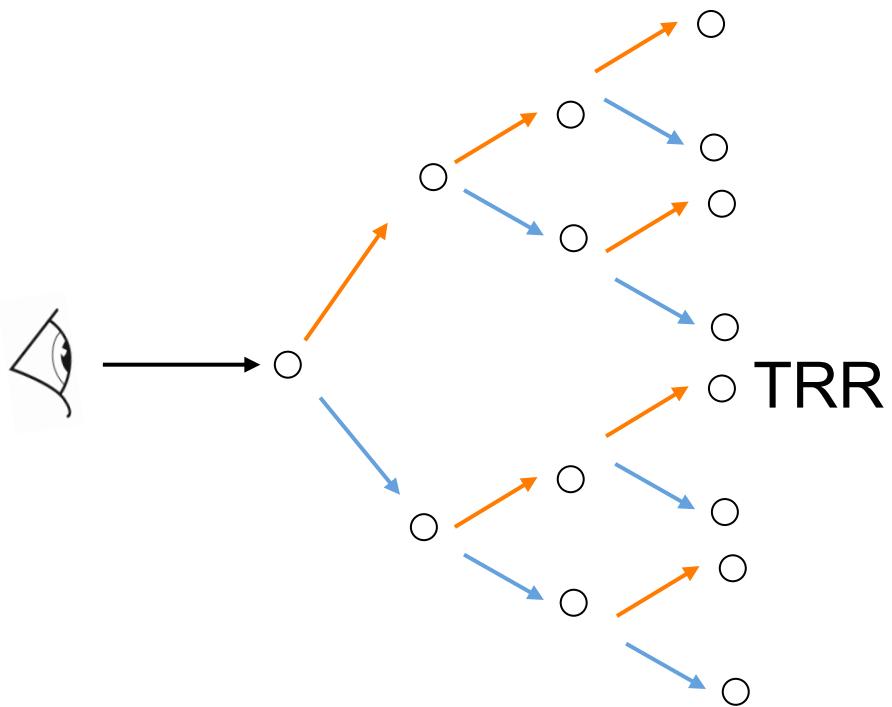
Recursive ray-tracing

- We can simulate specular-specular transmission elegantly by recursing and casting secondary rays from the intersection points
- We must obviously choose a termination depth to cope with multiple reflections

Recursive ray-tracing



Ray tree



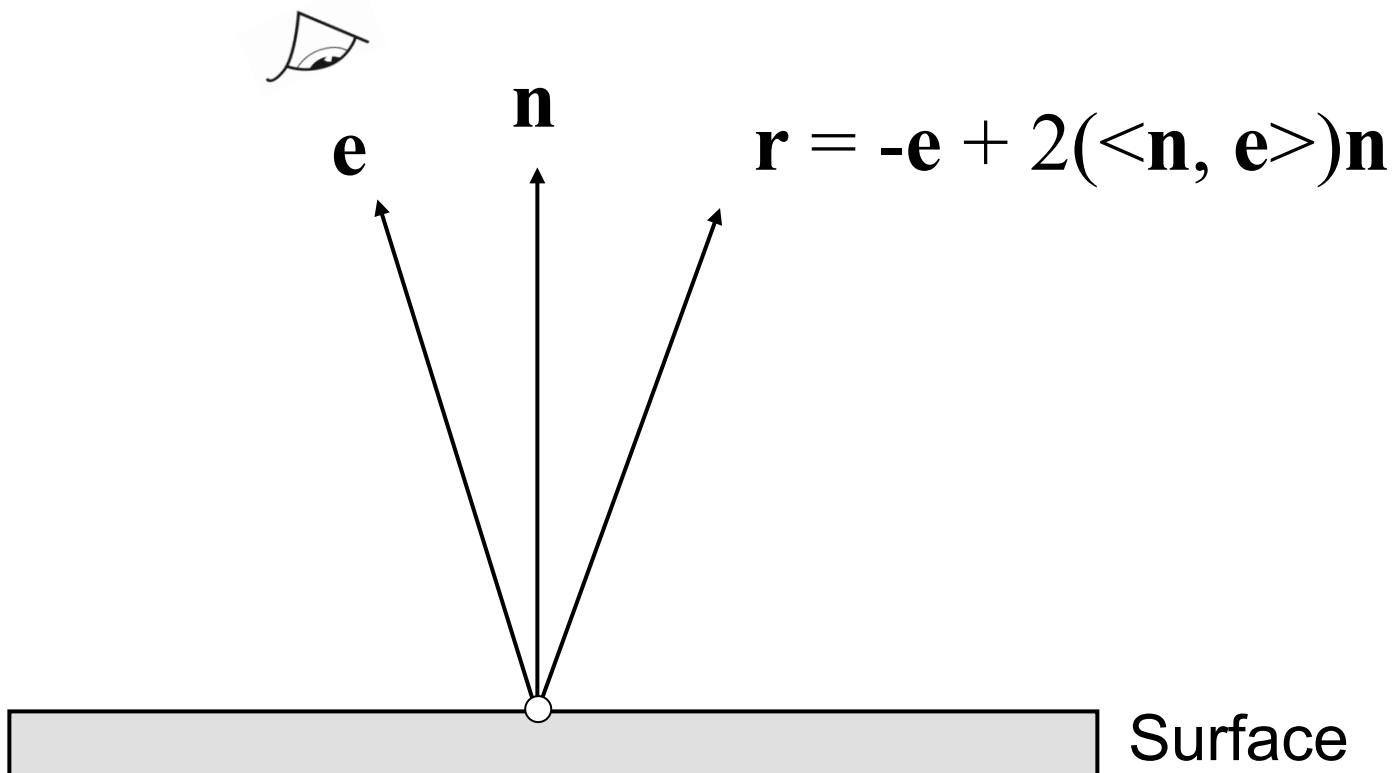
Refraction (T)



Reflection (R)



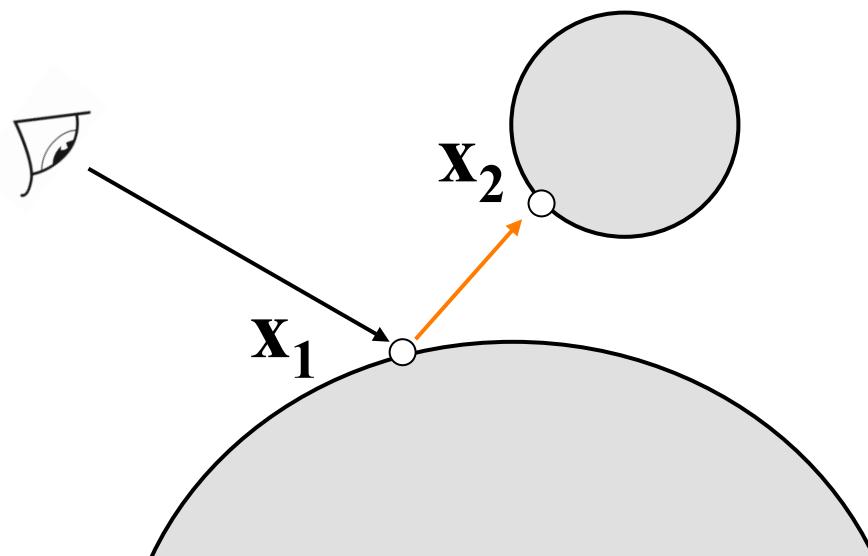
Introducing reflection



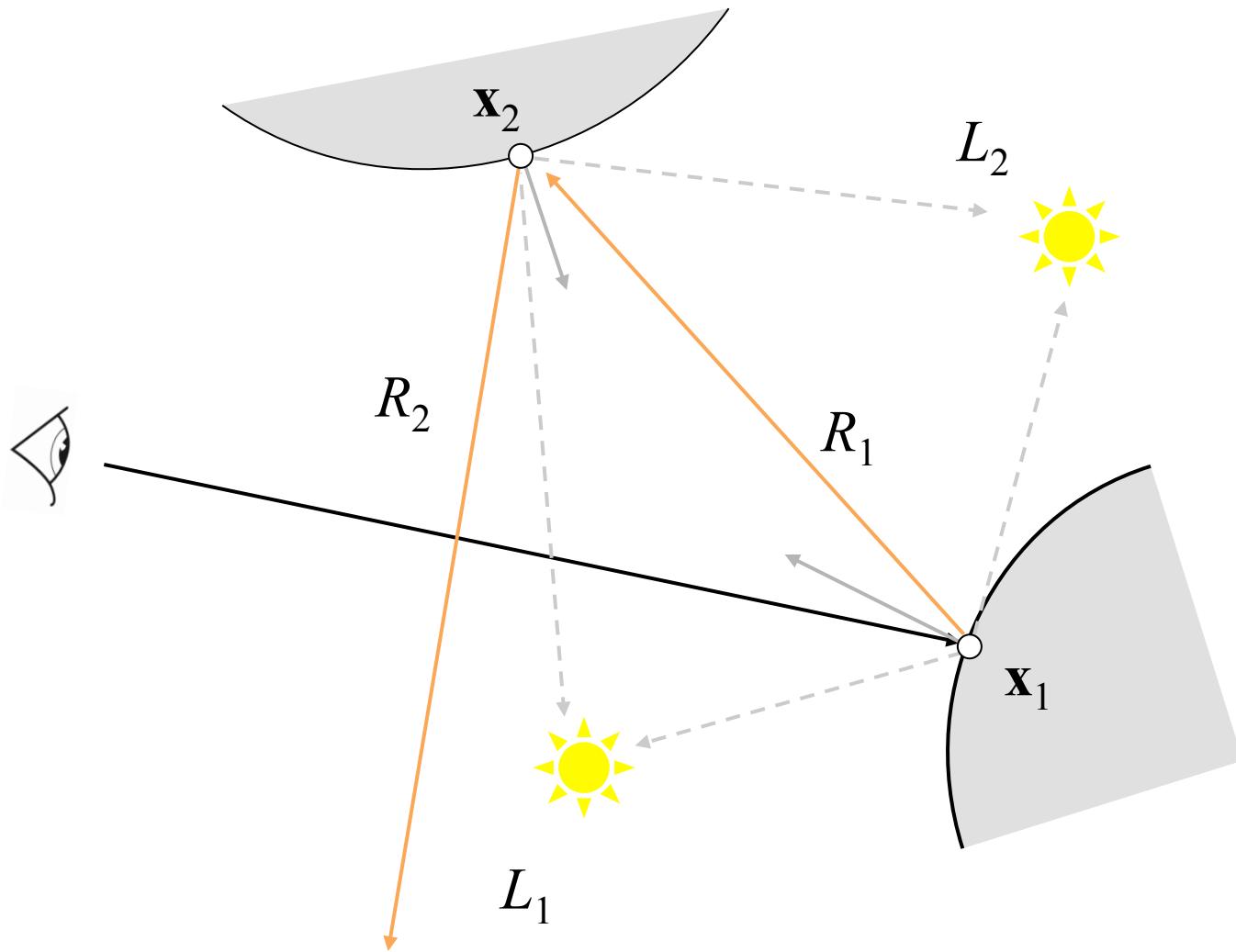
Computing mirror reflections

$$I_r = I_{\text{Local}} + k_r I_{\text{Mirror}}$$

- I_{Local} is computed at \mathbf{x}_1 as before
- I_{Mirror} is computed at \mathbf{x}_2 , the location visible in the mirror



Recursive ray-tracing

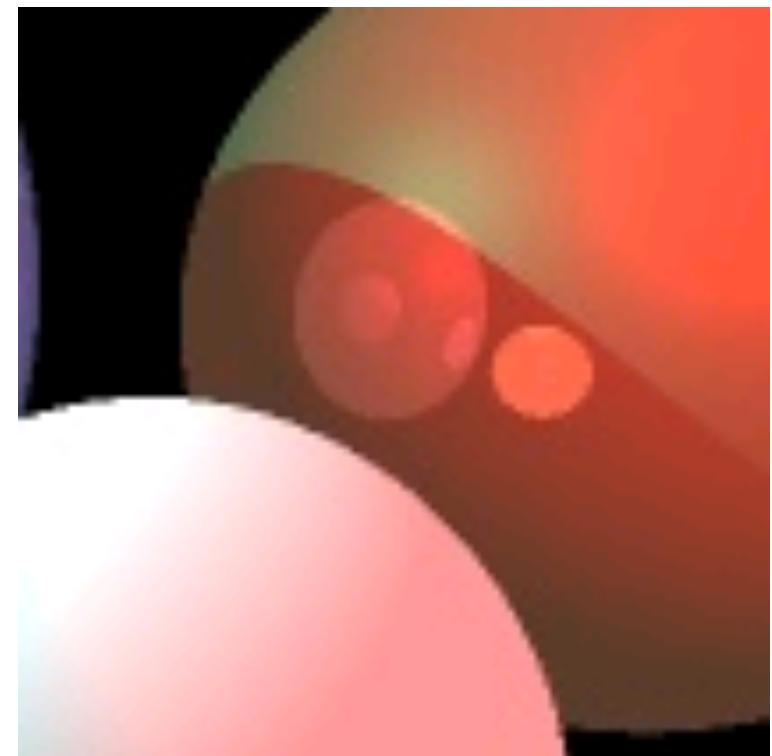
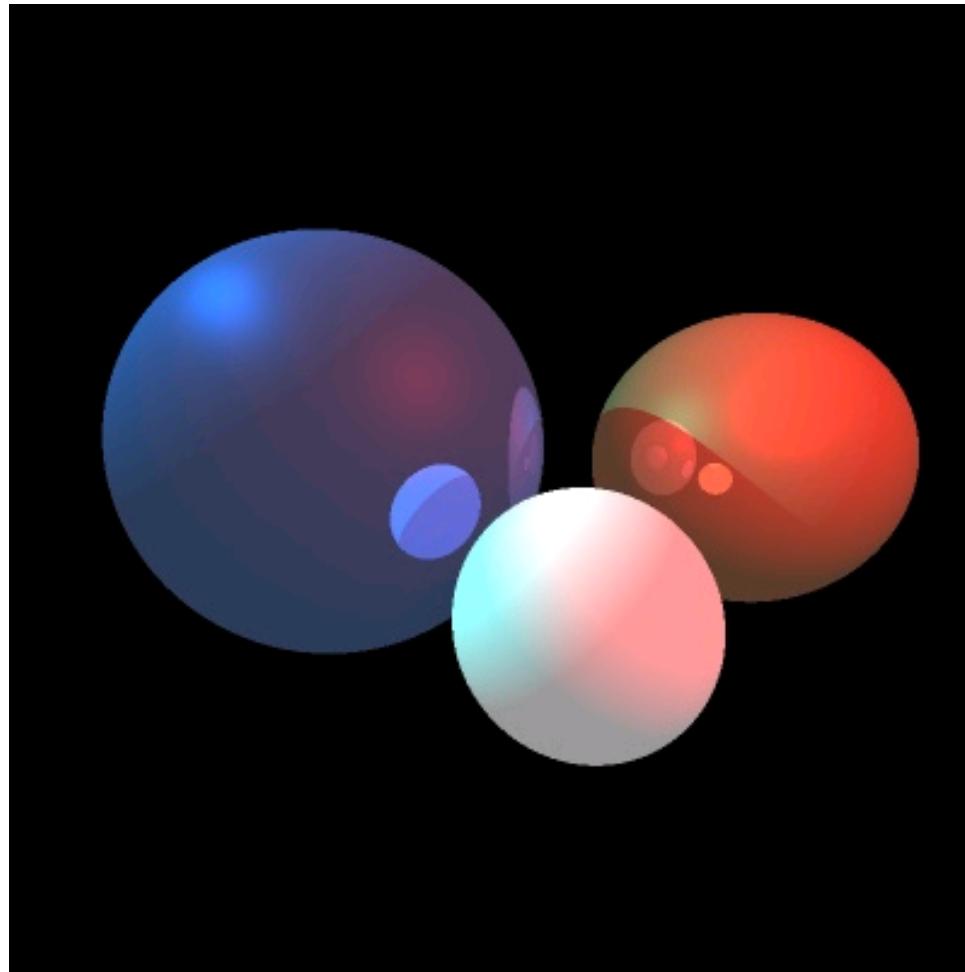


Pseudo code

```
vec3 trace(Ray ray, Scene scene, int depth) {
    if (depth > MAX) return Black;
    HitInfo hitInfo = intersect(ray, scene);
    if (!hitInfo.isHit) return Background

    return
        shade(ray, hitInfo, scene) +
        hitInfo.k_r * trace(reflect(ray, hitInfo), scene, depth + 1);
}
```

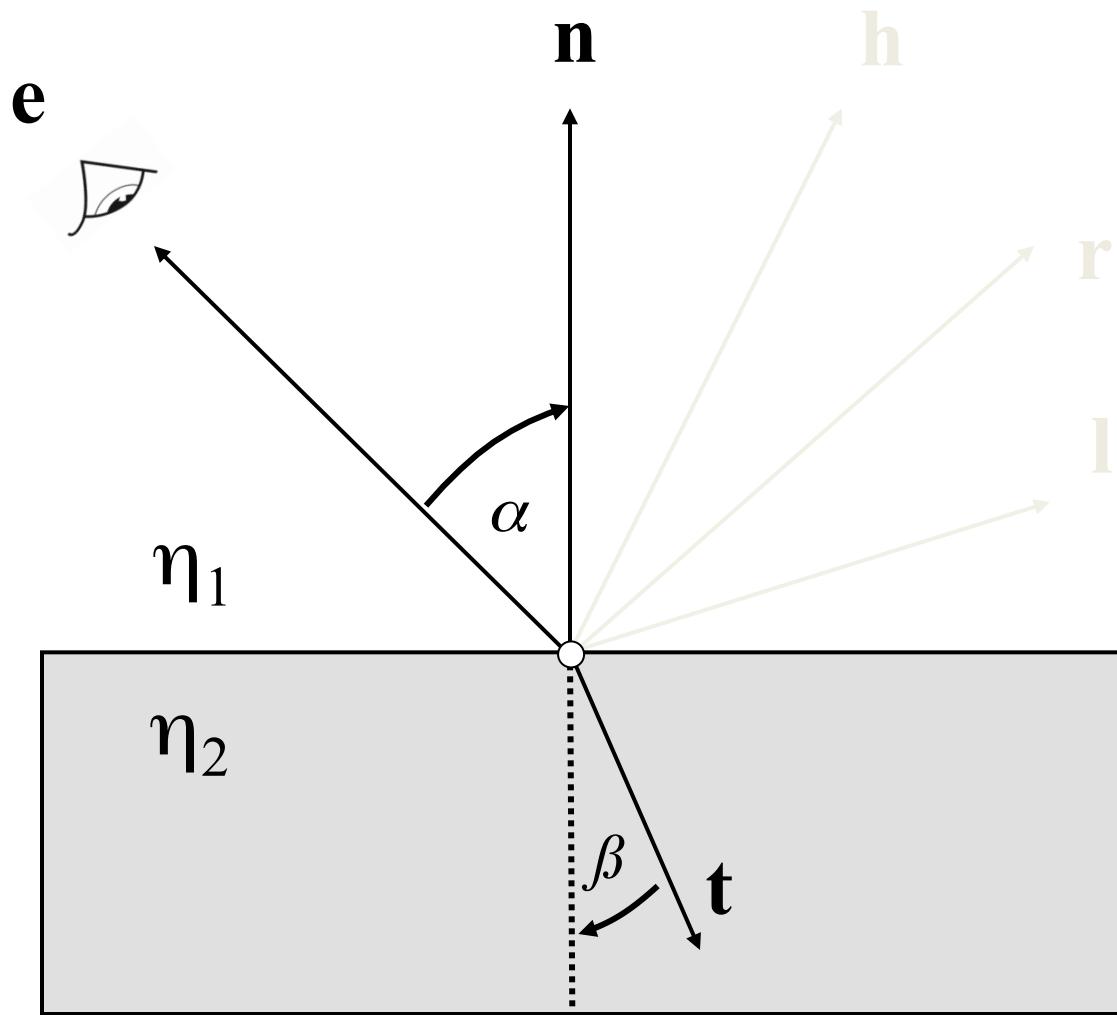
Result of recursion



Refraction



Refraction

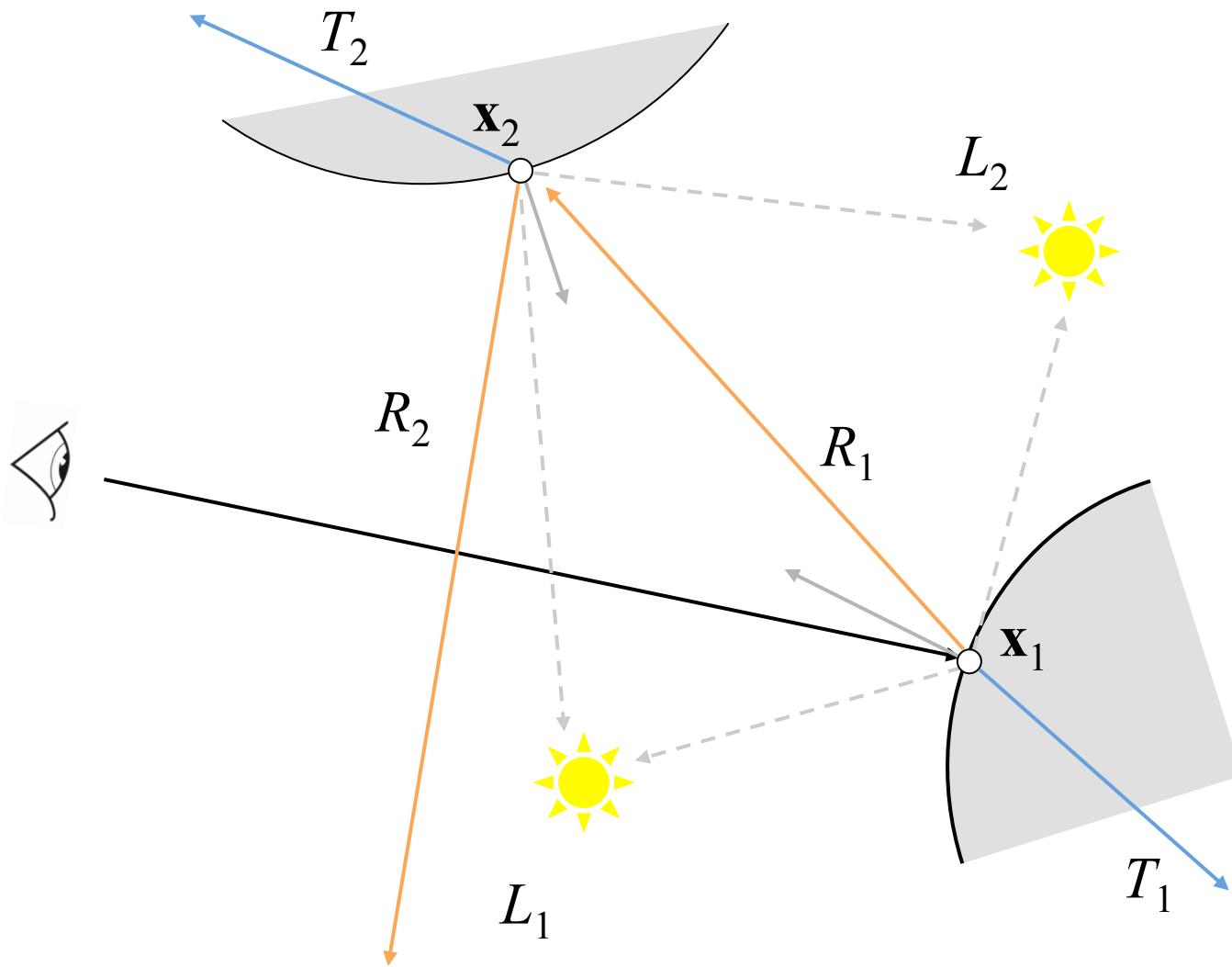


Snell's law

$$\frac{\sin \alpha}{\sin \beta} = \frac{\eta_2}{\eta_1}$$

η is index of
refraction

Recursive ray-tracing



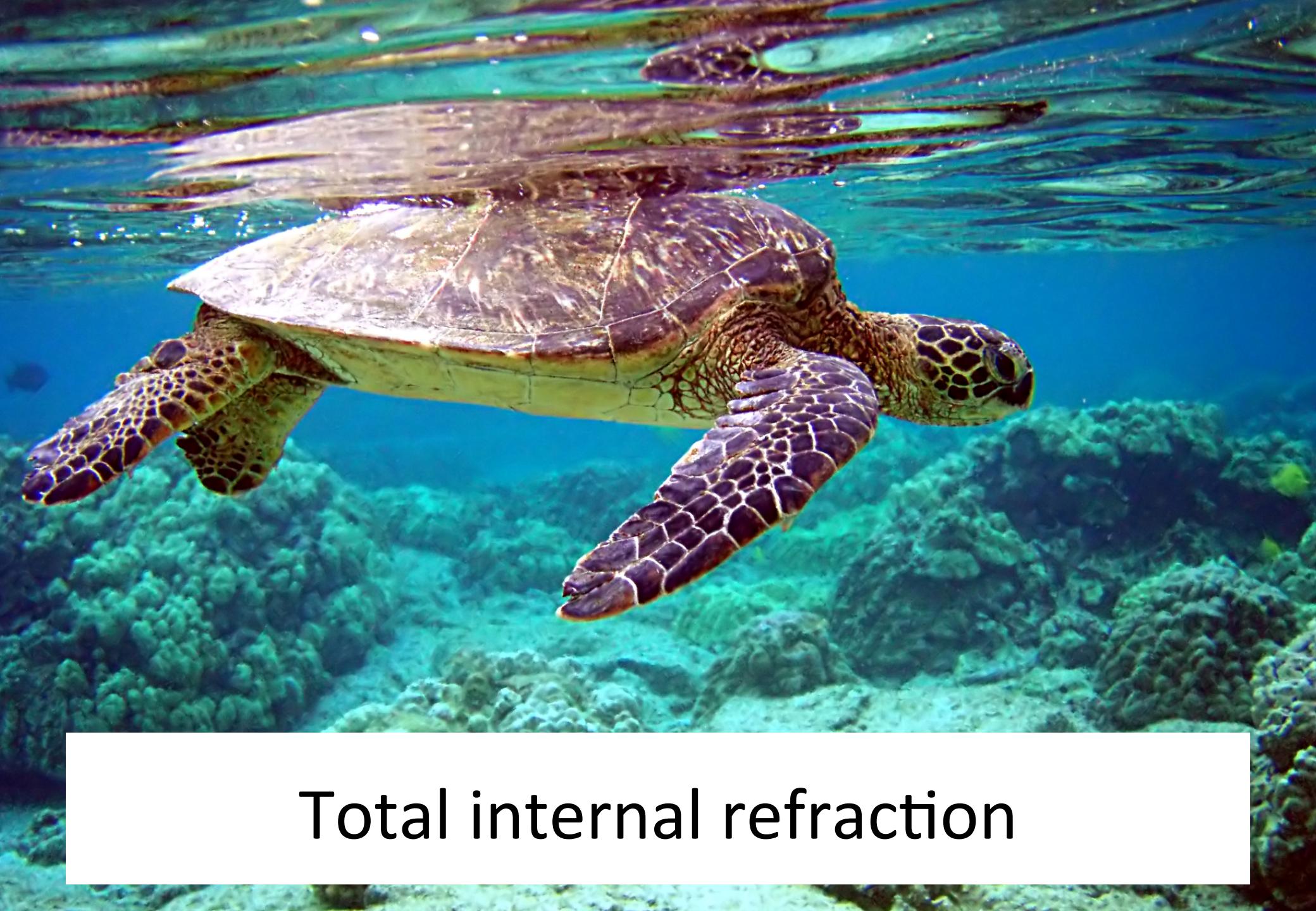
Using Snell's law

$$\frac{\sin \alpha}{\sin \beta} = \frac{\eta_2}{\eta_1} = \eta_{21}$$

Refraction direction:

$$\mathbf{t} = -\eta_{12}\mathbf{e} + \mathbf{n} \left(\eta_{12} \cdot \cos \alpha - \sqrt{1 + \eta_{12}^2 \cdot (\cos^2 \alpha - 1)} \right)$$

Note that if the root is negative then total internal reflection has occurred and you just reflect the vector as normal



Total internal refraction

New pseudo code

```
vec3 trace(Ray ray, Scene scene, int depth) {
    if (depth > MAX) return Black;
    HitInfo hitInfo = intersect(ray, scene);
    if (!hitInfo.isHit) return Background

    return
        shade(ray, hitInfo, scene) +
        hitInfo.kr * trace(reflect(ray, hitInfo), scene, depth + 1) +
        hitInfo.kt * trace(refract(ray, hitInfo), scene, depth + 1);
}
```

Stackless raytracing

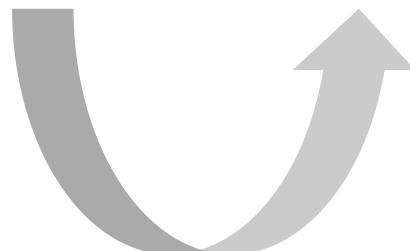
Implementation with stack:

```
void a(int depth) {  
    echo depth;  
    if (depth > 10) return;  
    a(depth + 1);  
}
```

```
a(0);
```

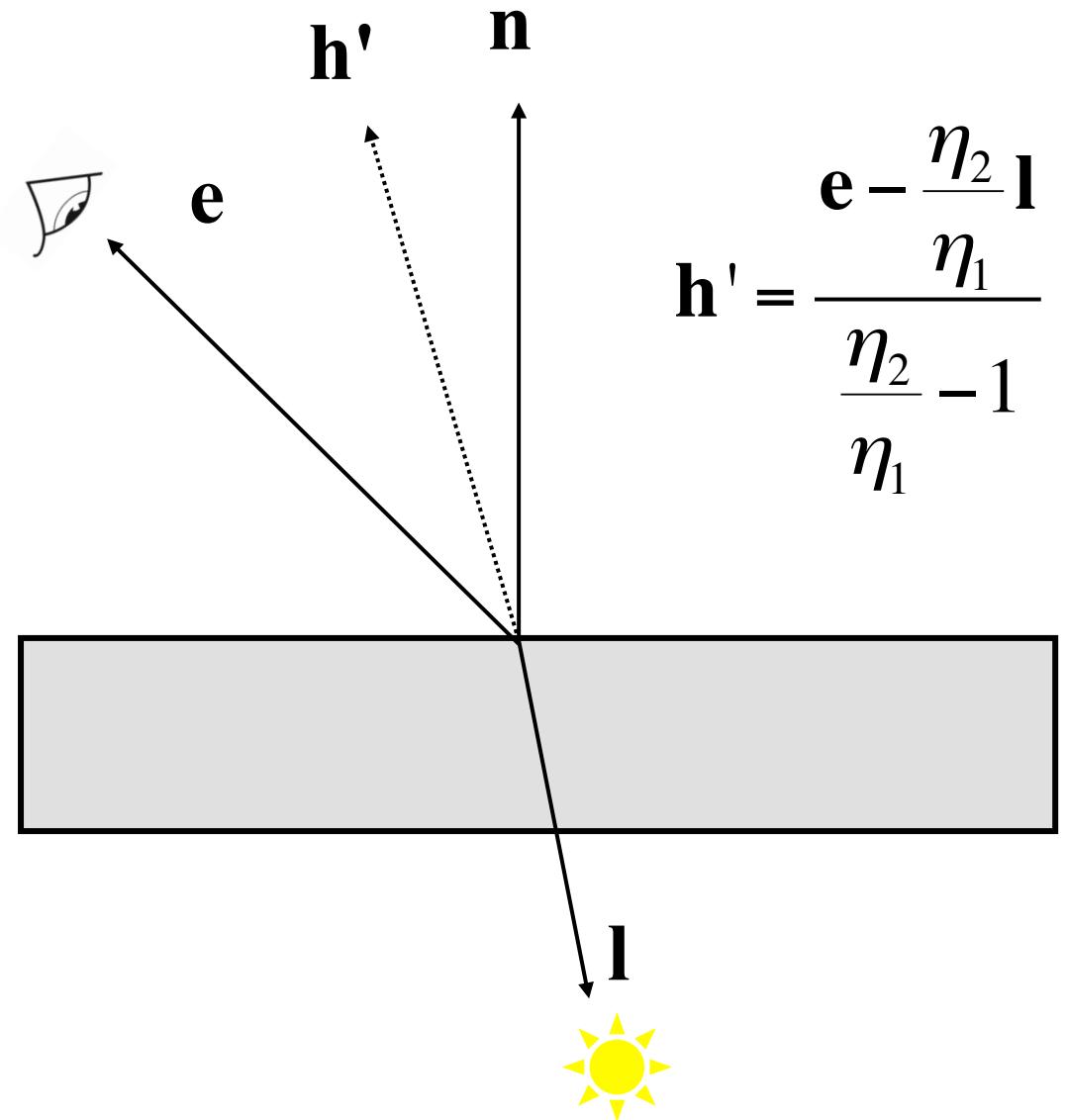
Implementation w/o stack:

```
depth = 0  
for(int i from 0 to 10) {  
    echo depth;  
    depth = depth + 1;  
}
```

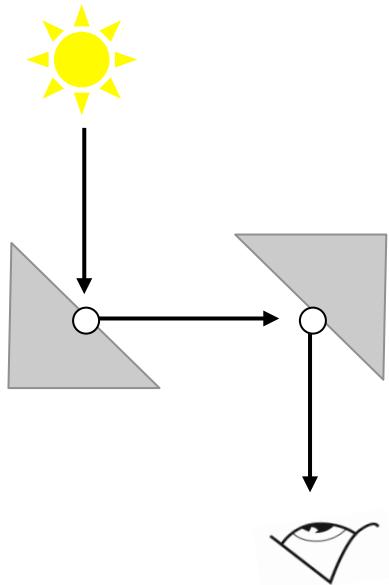




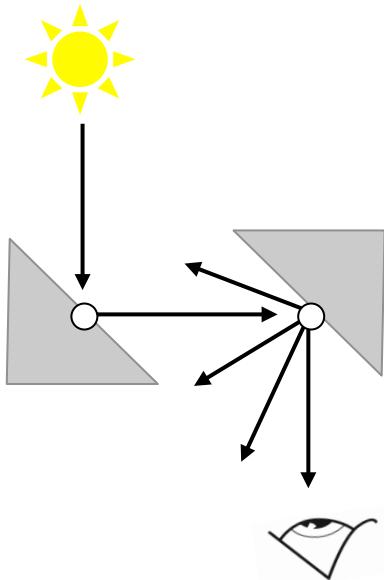
A transparent surface
can be illuminated from
behind and this should
be calculated in I_{Local}



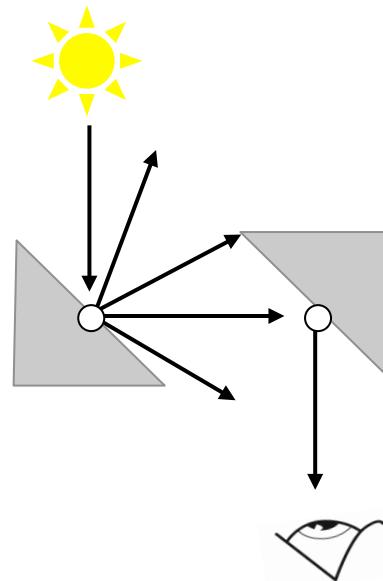
Discussion – What Can't We Simulate?



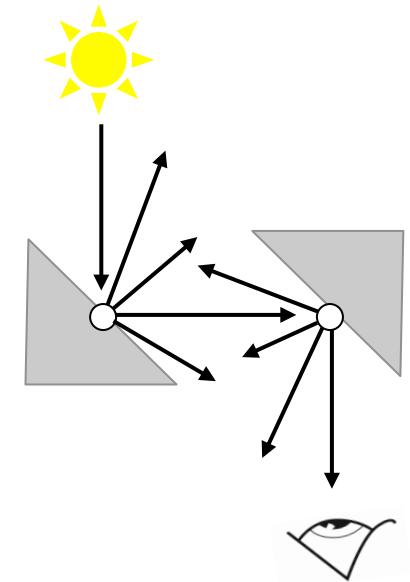
Specular-specular



Diffuse-specular



Specular-diffuse



Diffuse-diffuse

Light?



Photography by © Kevin Best

Remark

- Reflection and refraction-only
 - What should be added to consider diffuse reflection?
- Why it's expensive
 - Intersection of rays with polygons (90%)
- How to reduce the cost?
 - Reduce the number of rays
 - Reduce the cost on each ray
 - First check with bounding box of the object
 - Methods to sort the scene and make it faster

Summary

- Recursive ray tracing is a good simulation of specular reflections
- We've seen how the ray-casting can be extended to include shadows, reflections and transparent surfaces
- However this is a very slow process and still misses some types of effect!