Global Illumination

Using Photon Mapping

Definition

Global illumination is a group of algorithms used in 3D computer graphics that are meant to add more realistic lighting to 3D scenes.

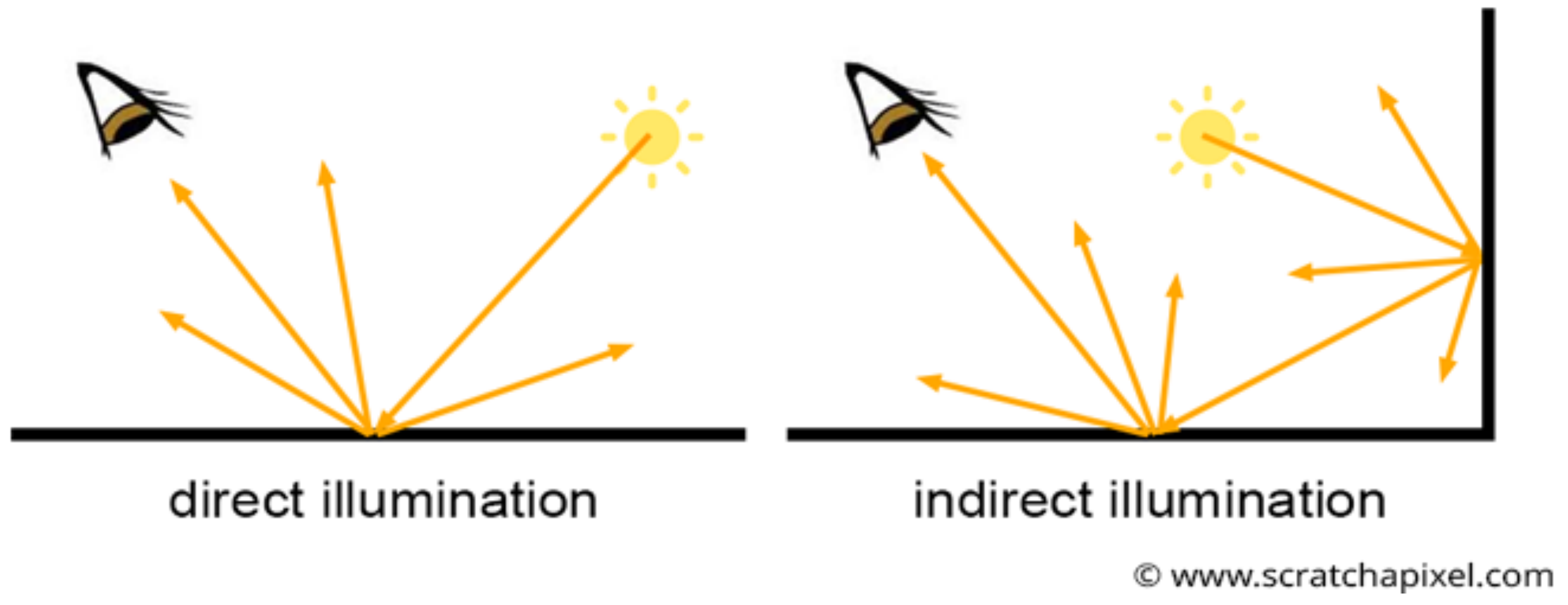
Global illumination involves to simulate both effects: direct plus indirect illumination.

Direct Illumination:

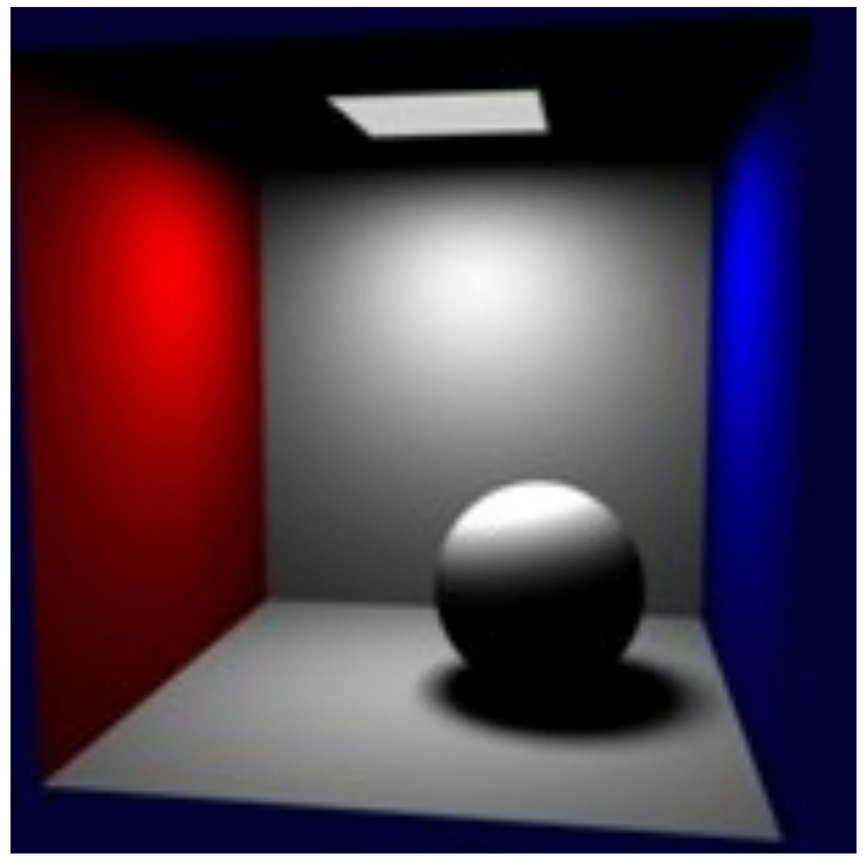
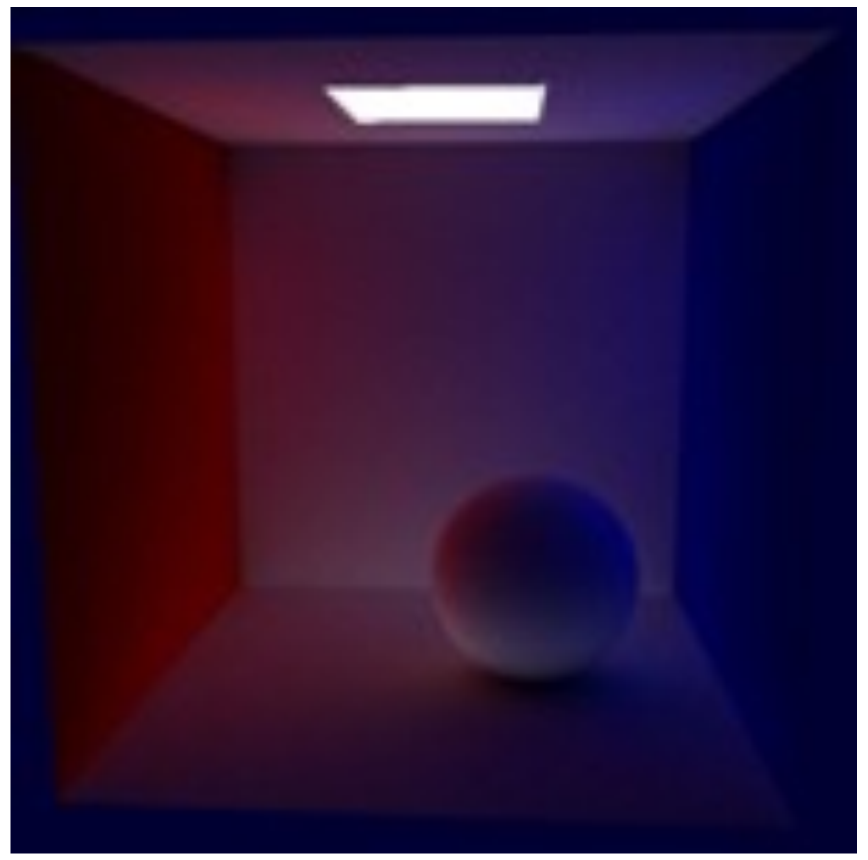
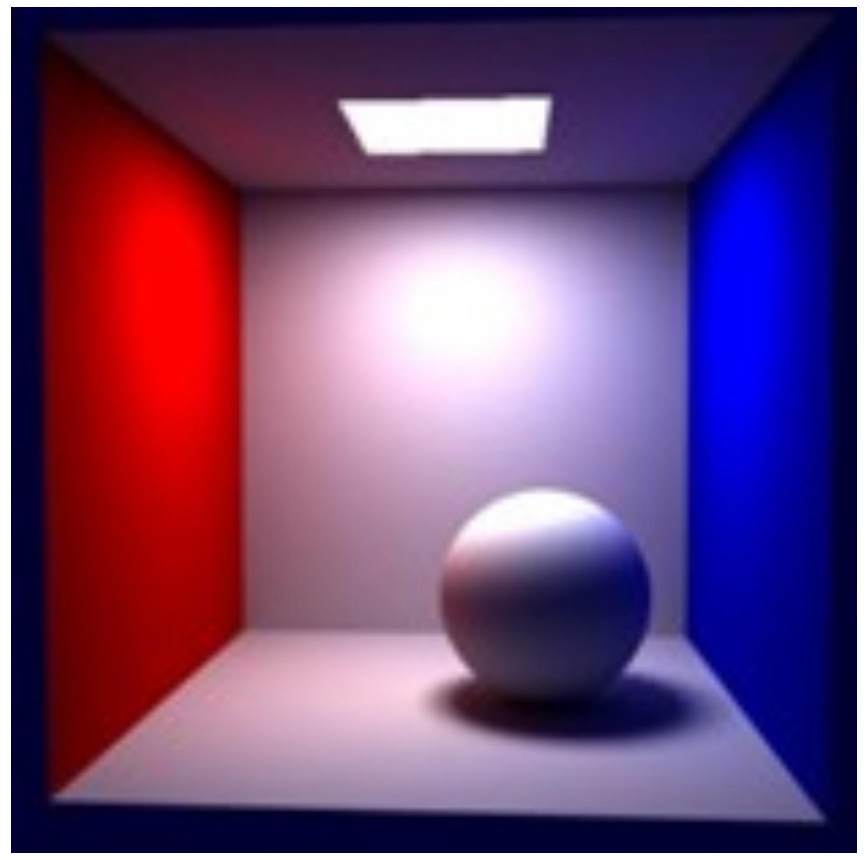
When light rays bounce only once from the surface of an object to reach the eye.

Indirect Illumination:

When light rays are emitted by a light source, they can bounce off of the surface of objects multiple times before reaching the eye.



Direct Illumination + Indirect Illumination = Global Illumination

Rendering Equation:

Total light leaving a point = Emitted light from that point

+

Reflected light from that point

$$L_o(x,\omega) = L_e(x,\omega) + L_r(x,\omega)$$

Various rendering technique have some portion of the rendering equation

If a technique approximates the integral, it is part of the family of "Global Illumination"

Photorealistic Lighting

Images rendered using global illumination algorithms often appear more photo realistic than those using only direct illumination algorithms.

For solving the equation

To compute light leaving x, we need to find how much light reaches it

To find how much light reaches x, we need to compute light leaves every other point

Hard because BRDFs are high dimensional

But some light interaction in the scene is diffused

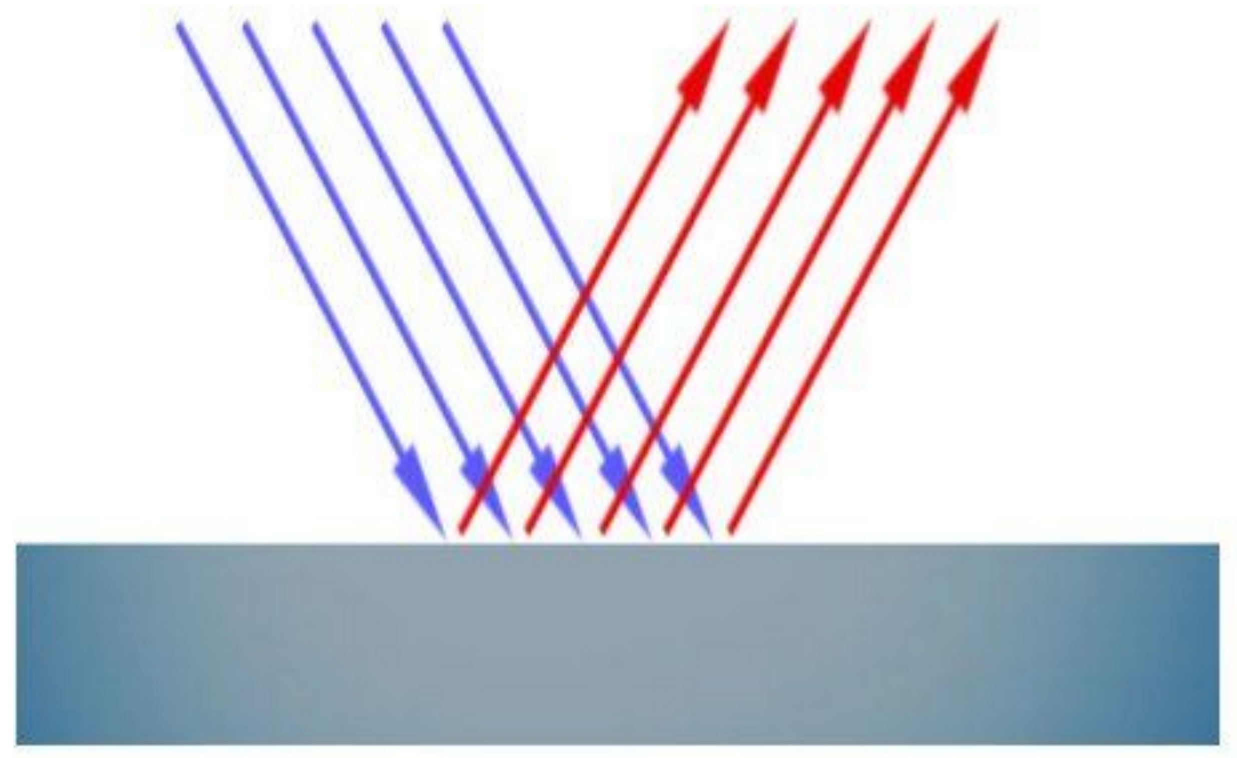
Types of Reflection

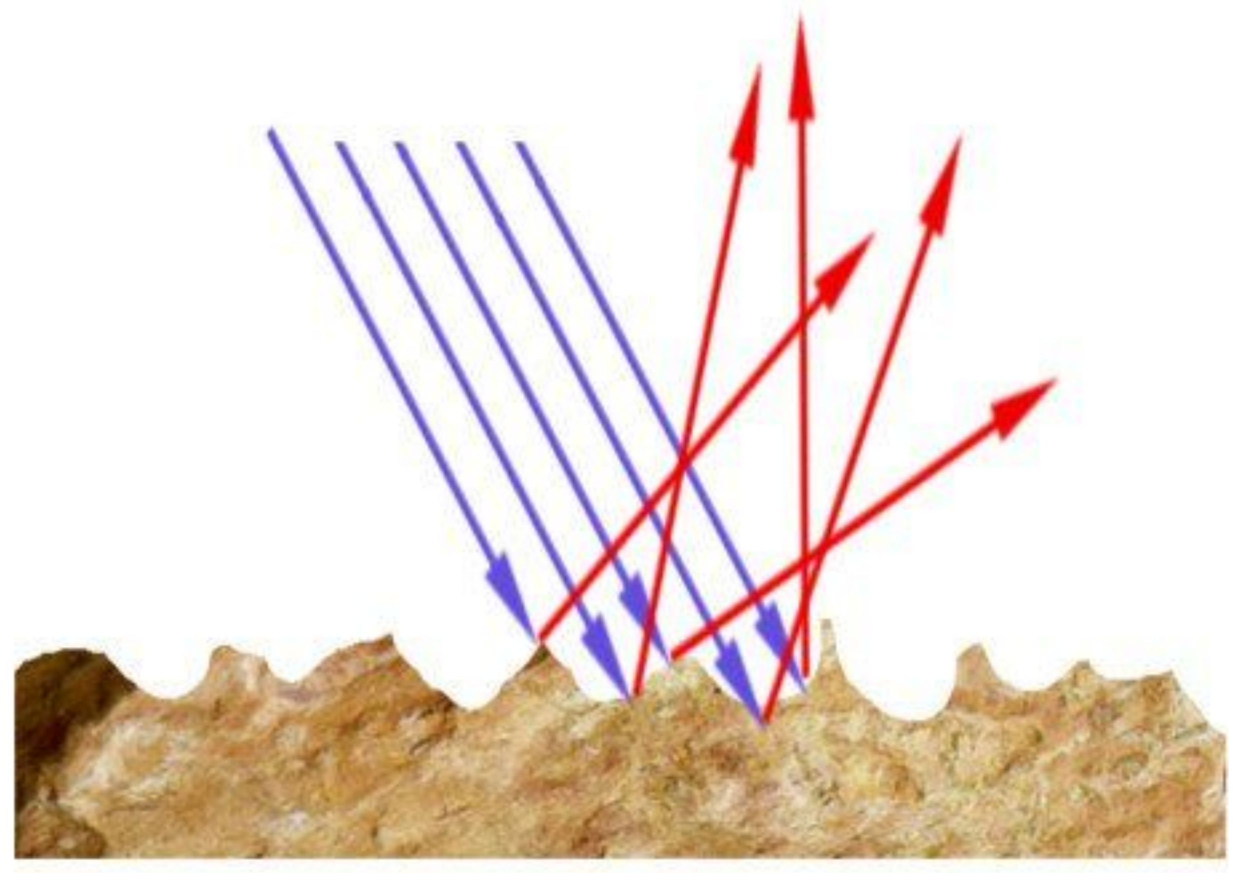
Specular Reflection

Light reflected from a smooth surface at a definite angle

Diffuse Reflection

Light reflected from a rough surface in all directions


Specular Reflection


Diffuse Reflection

Most Popular Global Illumination Techniques

Radiosity

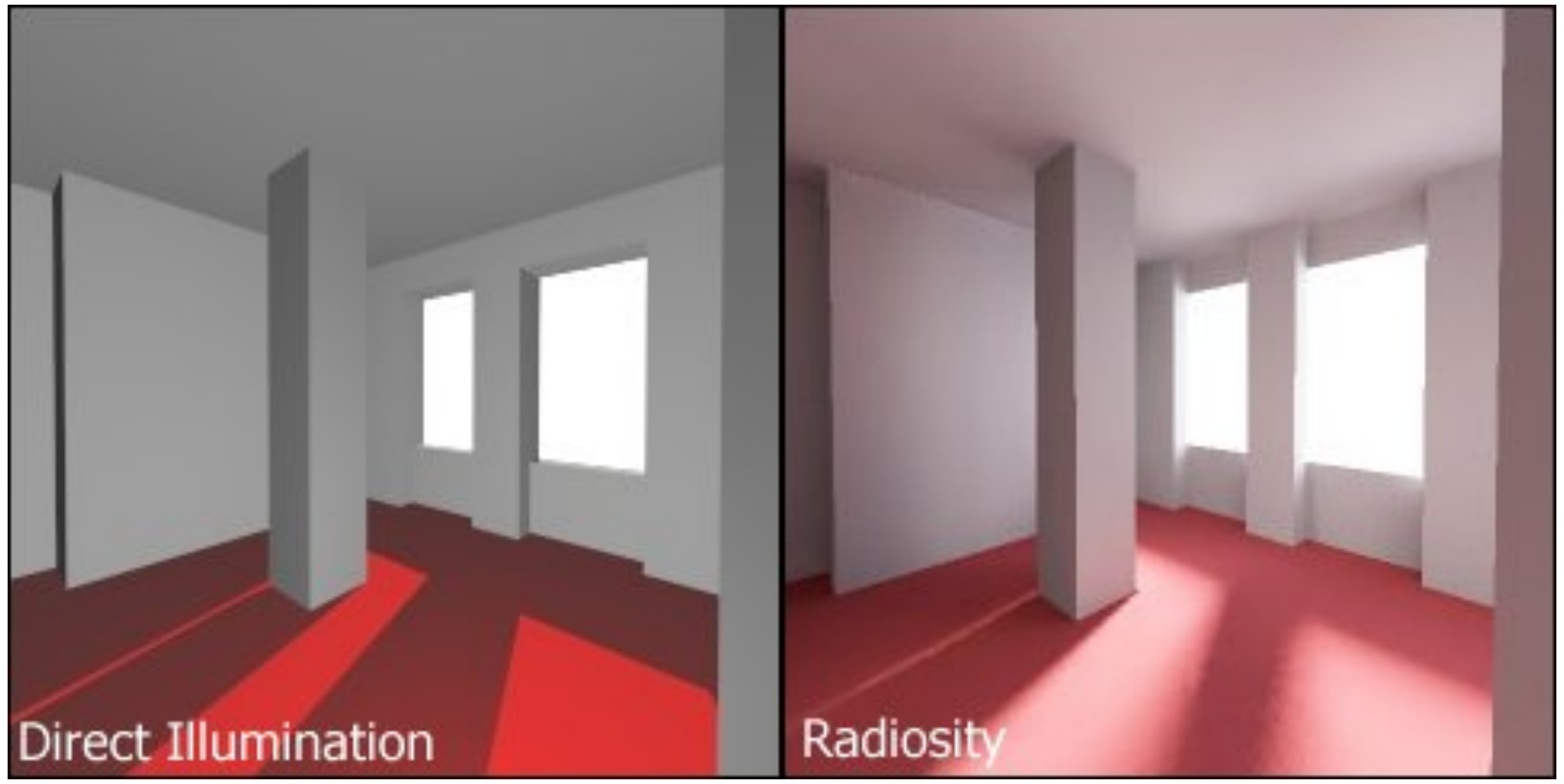
Ray Tracing

Photon Mapping

Other Techniques

beam tracing, cone tracing, path tracing, Metropolis light transport, ambient occlusion, signed distance field and image based lighting .

Radiosity



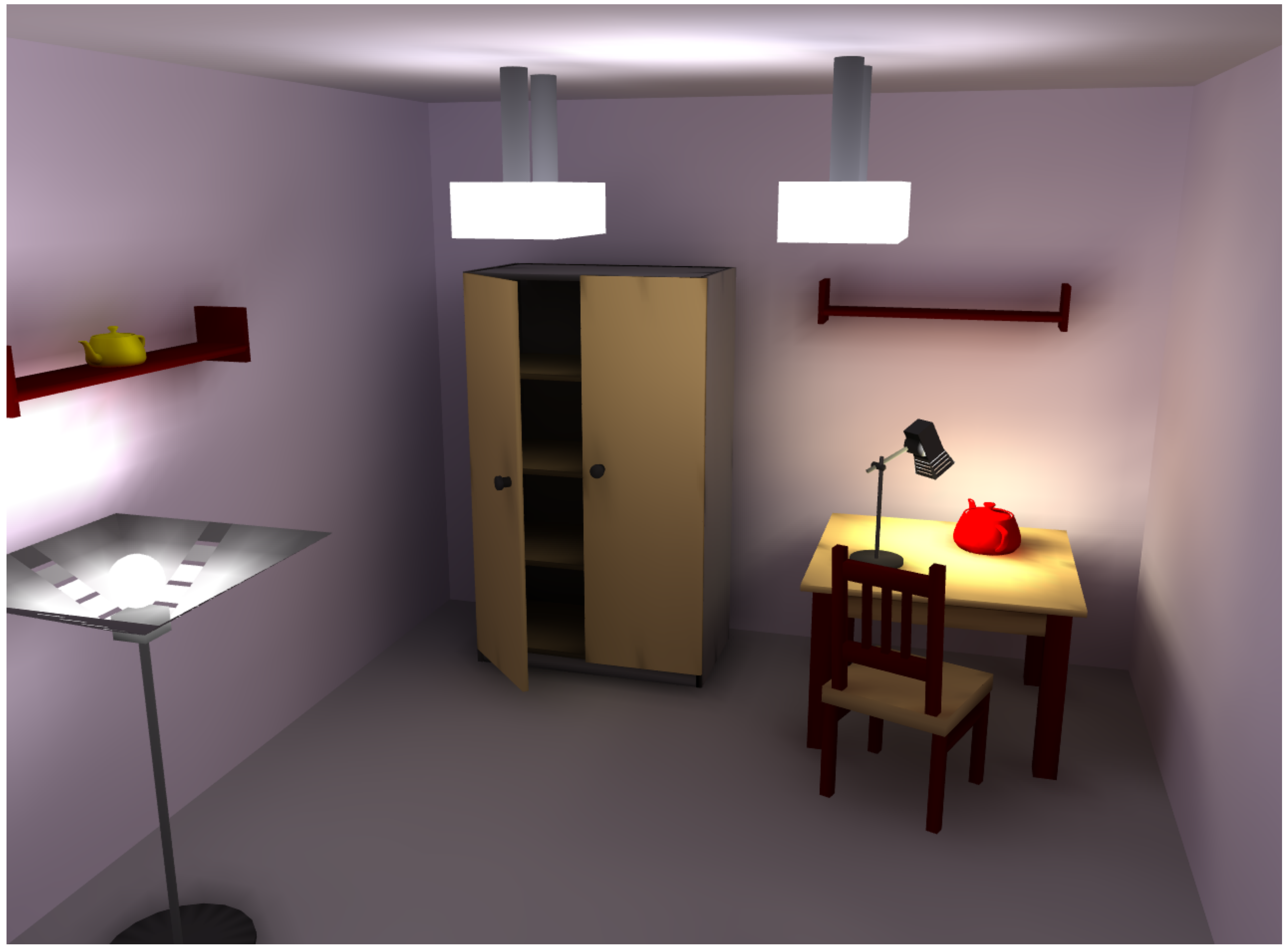
Global illumination algorithm

The main idea of the method is to store illumination values on the surfaces of the objects, as the light is diffuse propagated starting at the light sources.

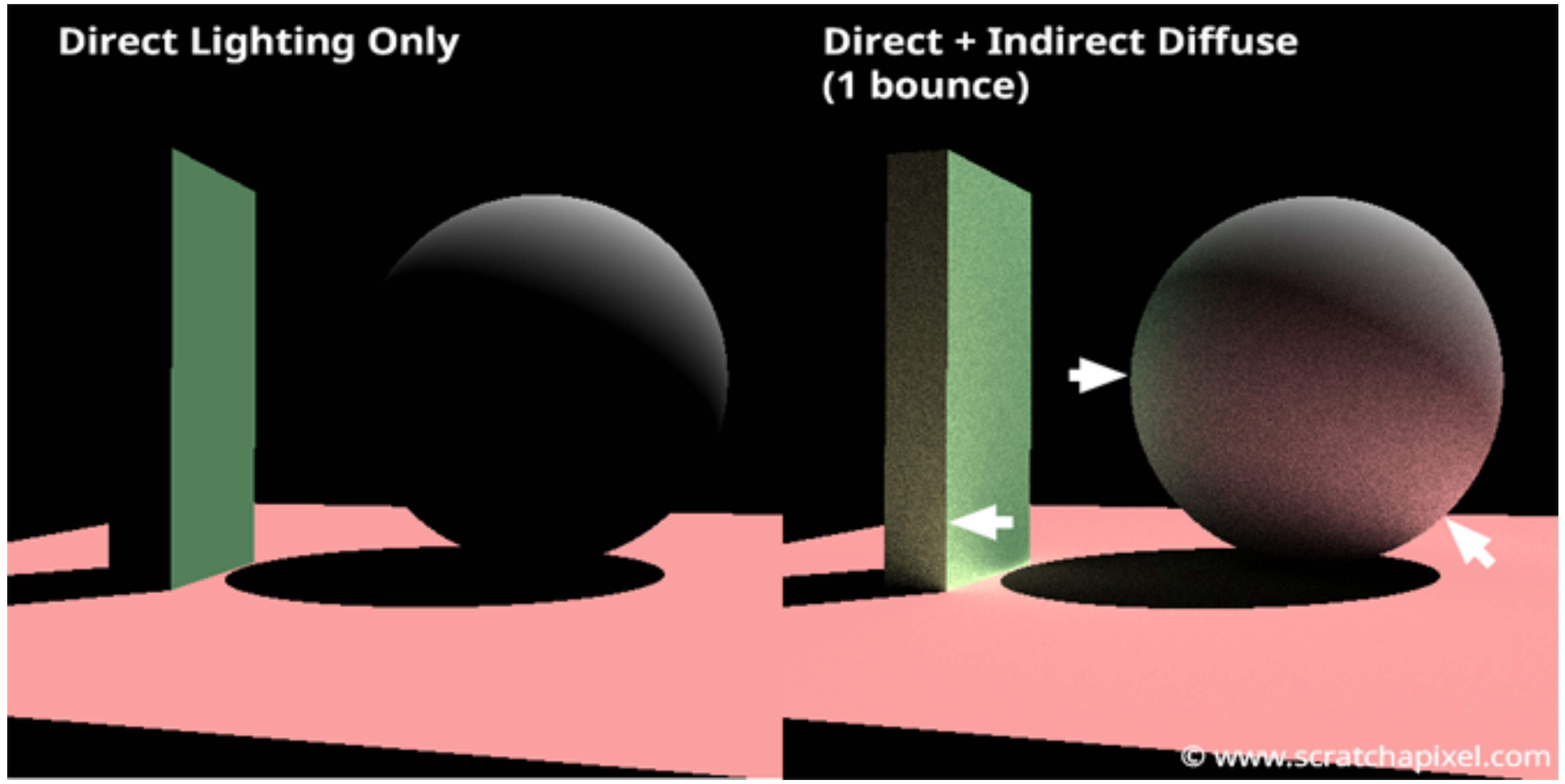
Radiosity does not account for specular reflections , it only accounts for diffuse reflections which resulting in soft scenes

Applications of Radiosity :

Radiosity is typically used to render images of the interior of buildings, and can achieve extremely photo-realistic results for scenes that are comprised of diffuse reflecting surfaces.



Ray Tracing



Global illumination algorithm

Ray tracing follows all rays from the eye of the viewer back to the light sources.

It can produce incredibly realistic lighting effects.

Works well for reflection and refraction

Diffuse inter reflections and caustics are not addressed properly

Applications of Ray Tracing

Ray tracing has been traditionally used with applications for media and entertainment (3D animation, rendering), product development (CAD/CAM/CAE), life sciences (medical, molecular), energy, and other operations.

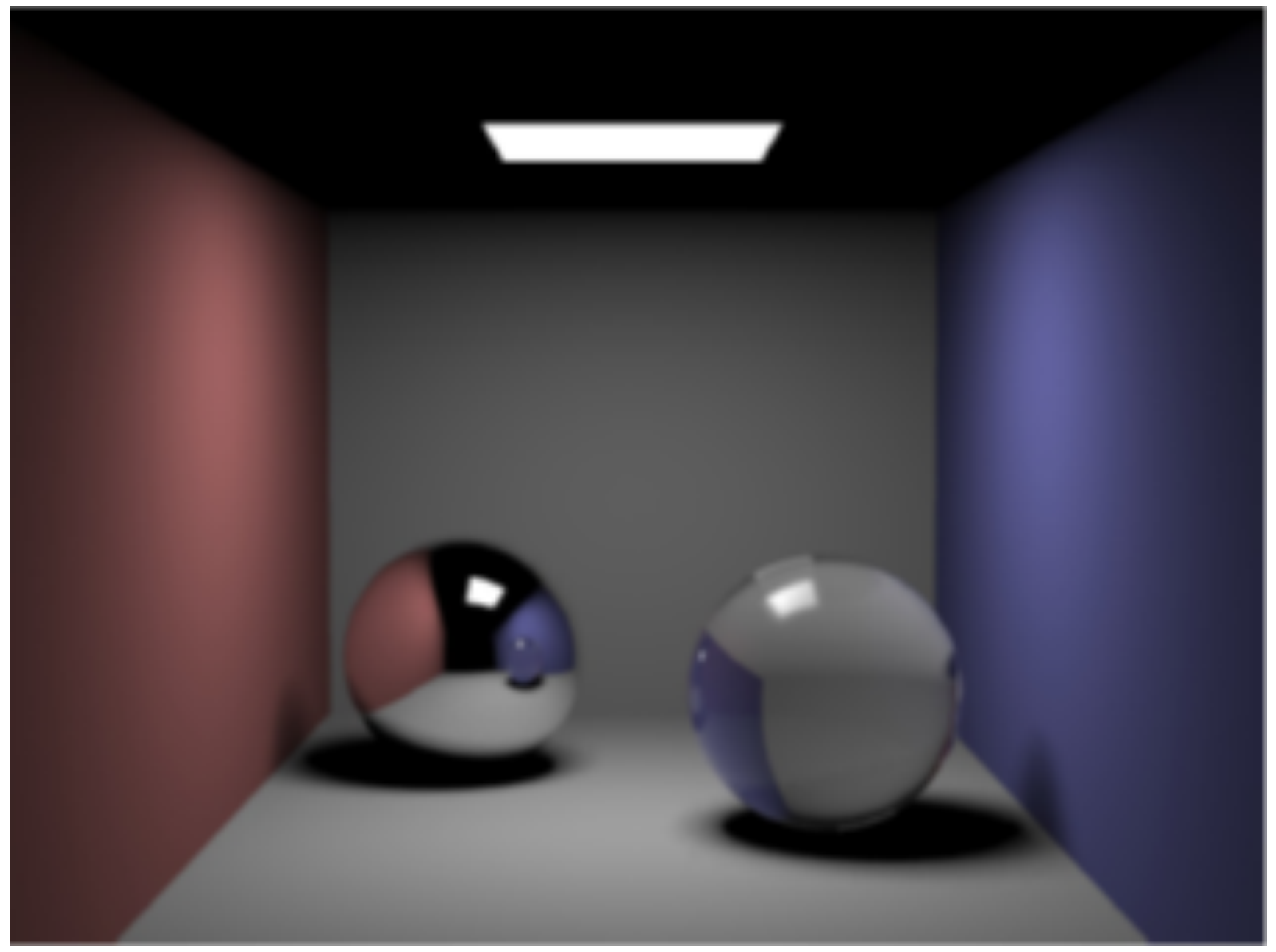
Porsche Product Demo

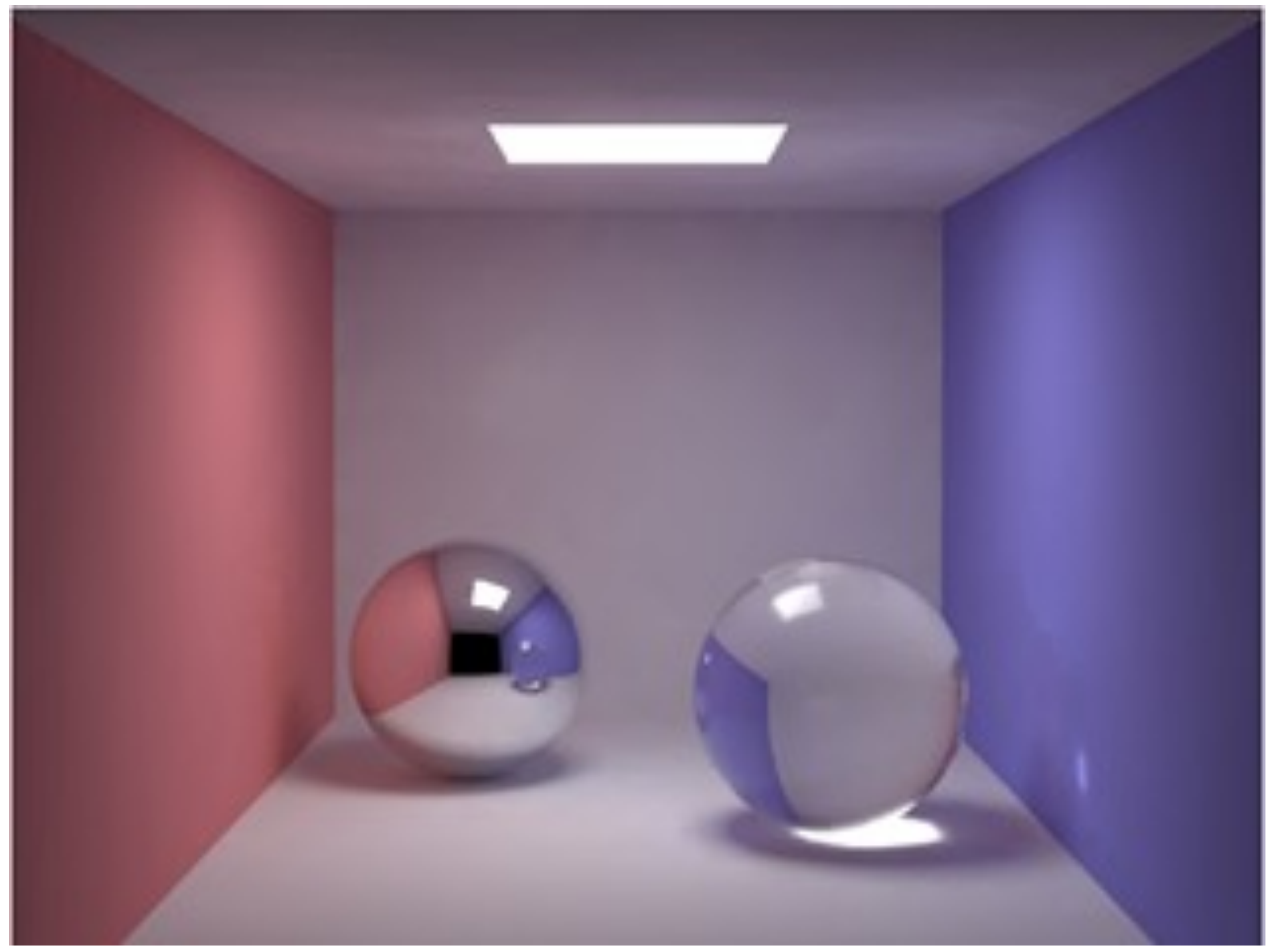
Why Photon Mapping?

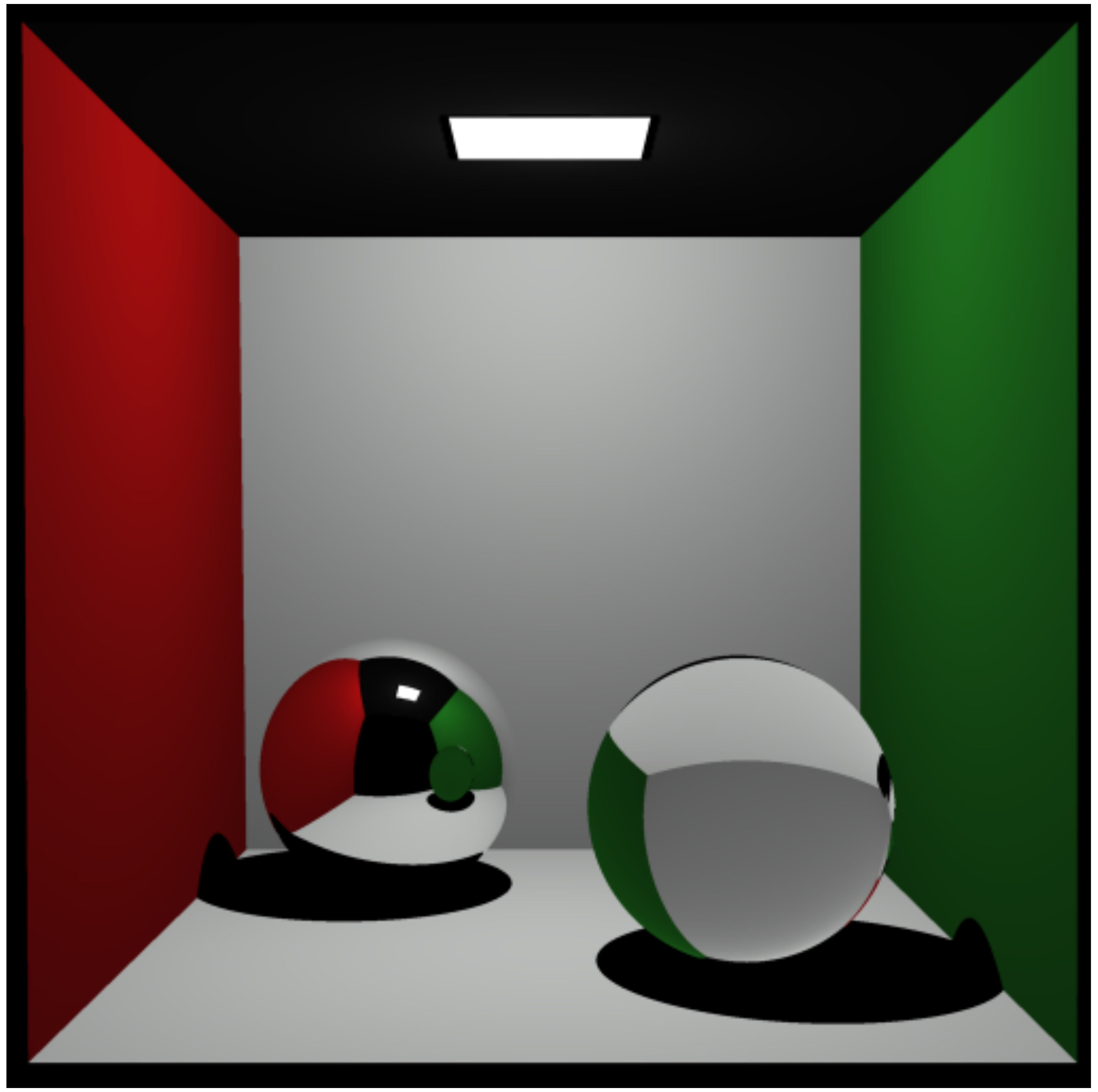
In 1989, Andrew Glassner wrote about ray tracing:

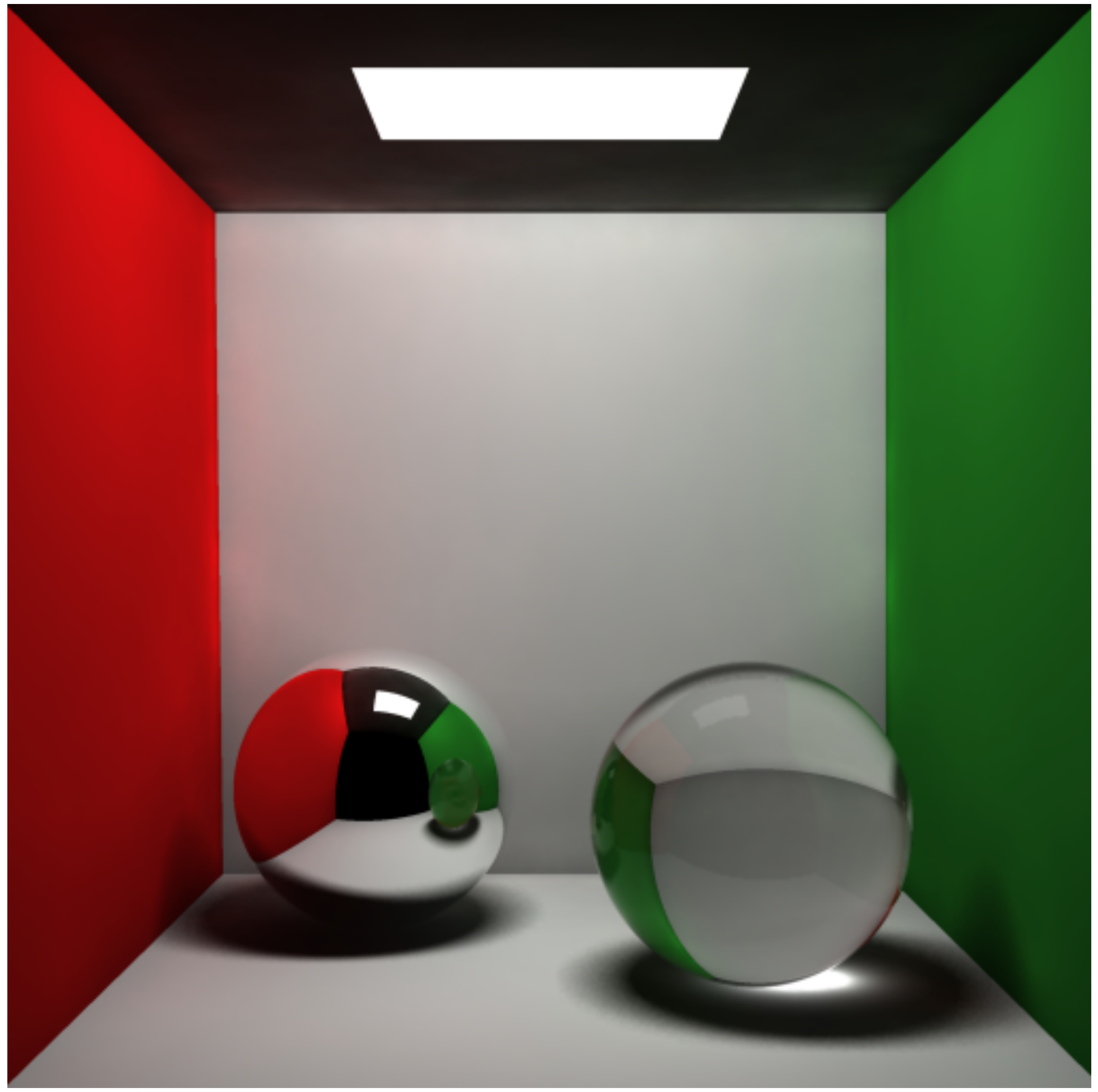
“Today ray tracing is one of the most popular and powerful techniques in the image synthesis repertoire: it is simple, elegant, and easily implemented. [However] there are some aspects of the real world that ray tracing doesn’t handle very well (or at all!) as of this writing. Perhaps the most important omissions are diffuse inter-reflections (e.g. the ‘bleeding’ of coloured light from a dull red file cabinet onto a white carpet, giving the carpet a pink tint), and caustics (focused light, like the shimmering waves at the bottom of a swimming pool).”

The photon mapping method is an extension of ray tracing.


Ray Tracing


Global Photon Map


Ray Tracing

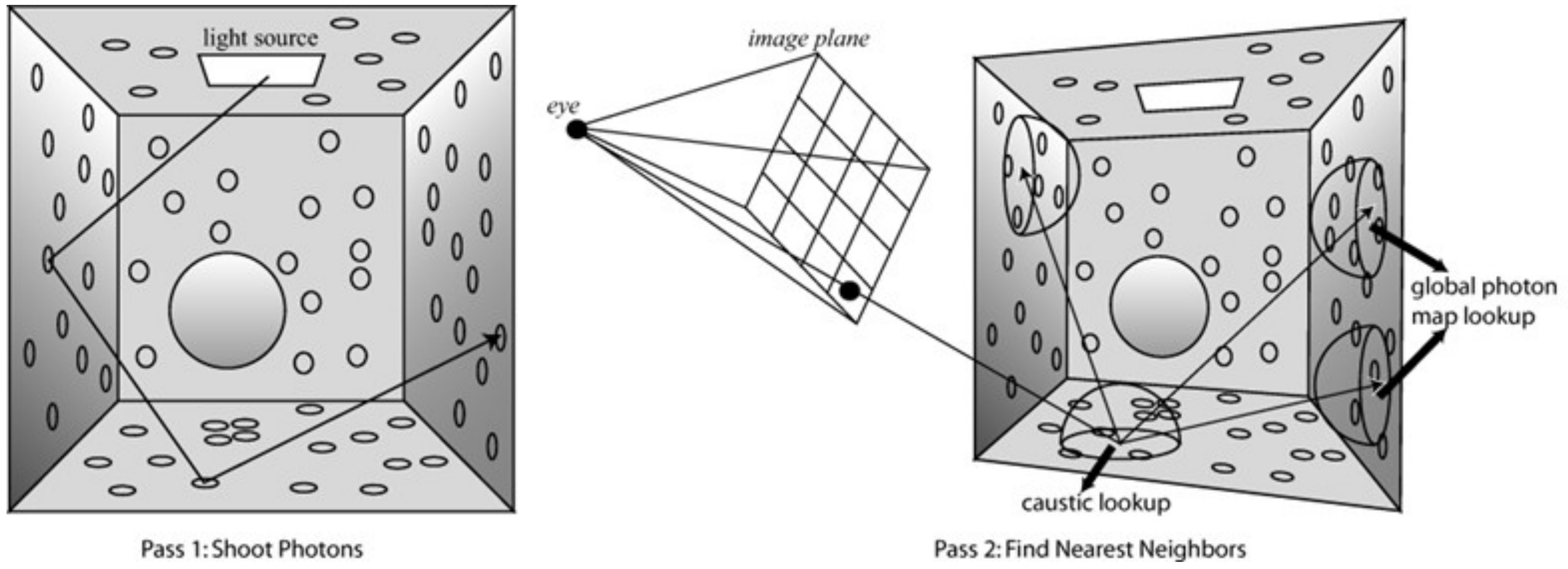

Photon Mapping

Photon Mapping

A two pass global illumination algorithm.

First Pass :- Photon Tracing and building the photon map

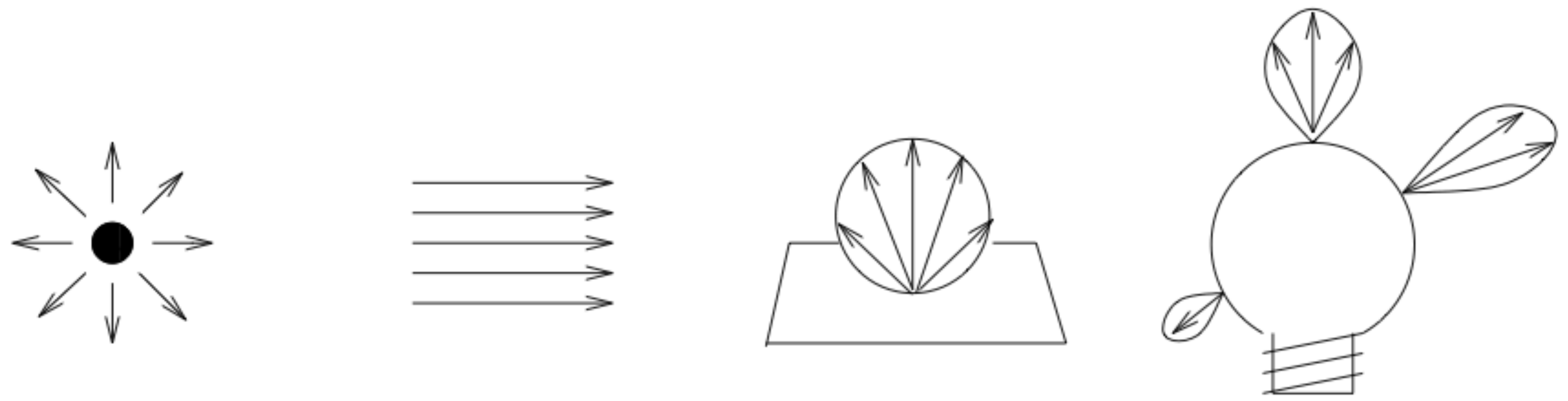
Second Pass :- Rendering



Photon Emission

Emission from a Single Light Source

The photons emitted from a light source should have a distribution corresponding to the distribution of emissive power of the light source. This ensures that the emitted photons carry the same flux — ie. we do not waste computational resources on photons with low power.



Emission from light sources: (a) point light, (b) directional light, (c) square light, (d) general light.

Power of emitted photon

The power (“watt”) of the light source has to be distributed among the pho- tons emitted from it. If the power of the light is Plight and the number of emitted photons is ne, the power of each emitted photon is

Power of photon = power of light / ne

To further reduce variation in the computed indirect illumination (during rendering), it is desirable that the photons are emitted as evenly as possible. This can for example be done with stratification or by using low-discrepancy quasi-random sampling.

Pseudo code for photon emission

```
emit photons from diffuse point light() {
    ne = 0 number of emitted photons
    while (not enough photons) {
    do { use simple rejection sampling to find diffuse photon direction
            x = random number between -1 and 1
            y = random number between -1 and 1
            z = random number between -1 and 1
     } while ( x2+y2+z2 >1 )
     d = < x, y, z >
     p = light source position
      trace photon from p in direction d ne =ne +1 }
scale power of stored photons with 1/ne}
```

Emission from a Multiple Lights

photons should be emitted from each light source. More photons should be emitted from brighter lights than from dim lights, to make the power of all emitted photons approximately even.

In a scene with many light sources, each light contributes less to the overall illumination, and typically fewer photons can be emitted from each light.

Projection Maps

In scenes with sparse geometry, many emitted photons will not hit any objects. Emitting these photons is a waste of time. To optimise the emission, projection maps can be used.

A projection map is simply a map of the geometry as seen from the light source.

First Pass :- Building Photon Map

Once a photon has been emitted, it is traced through the scene using photon tracing (also known as “light ray tracing”, “backward ray tracing”, “forward ray tracing”, and “backward path tracing”). Photon tracing works in exactly the same way as ray tracing except for the fact that photons propagate flux whereas rays gather radiance. This is an important distinction since the interaction of a photon with a material can be different than the interaction of a ray.

For e.g. radiance is changed based on the relative index of refraction but this does not happen to photons.

When a photon hits an object, it can either be reflected, transmitted, or absorbed. Whether it is reflected, transmitted, or absorbed is decided probabilistically based on the material parameters of the surface.

The technique used to decide the type of interaction is known as Russian roulette — basically we roll a dice and decide whether the photon should survive and be allowed to perform an- other photon tracing step.

Why Russian roulette?

Firstly, we prefer photons with similar power in the photon map. This makes the radiance estimate much better using only a few photons.

Secondly it allows to keep the number of traced photons low as well as keep their energy approximately the same.

For e.g. instead of tracing 1000 photons with half energy after hitting the surface, we will trace just 500 photons with the full energy as we assume 500 photons were simply absorbed by material.

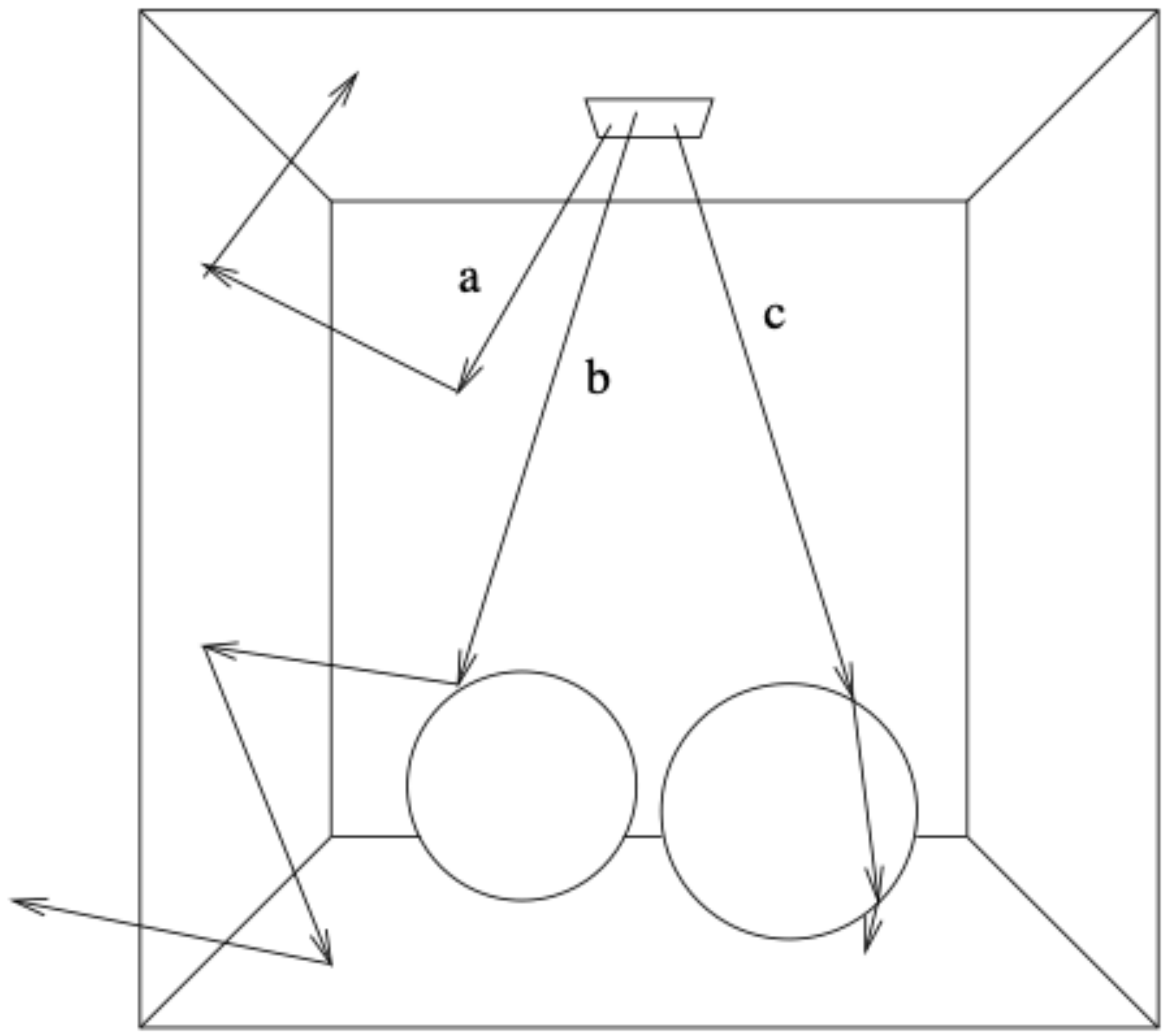
Phase I: Shoot and store photons

Photons are shot from the light into the scene

Photons are allowed to interact with objects in the environment

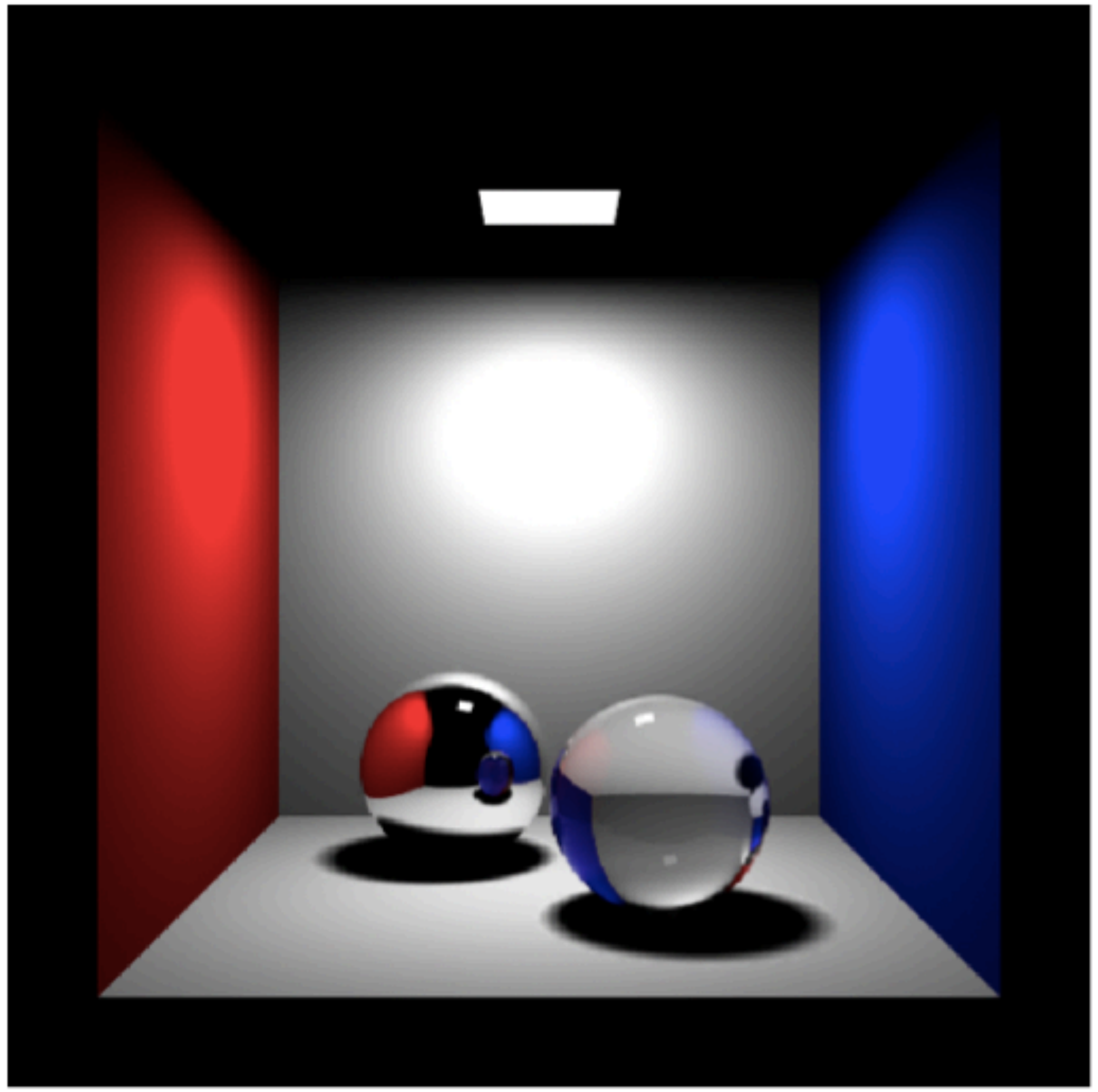
Where photons fall are stored in special data structure called a photon map

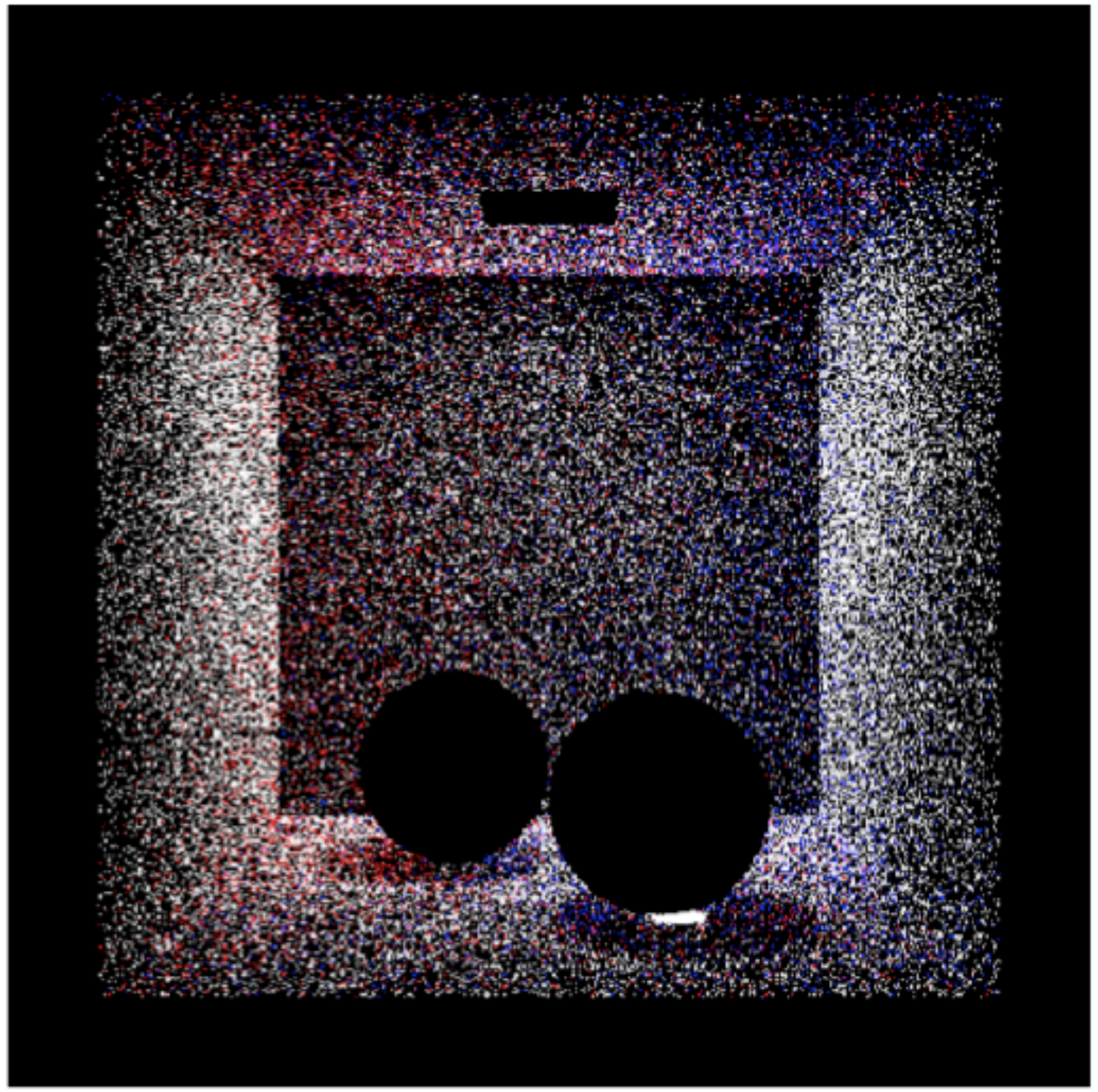
1000s of photos not billions(statistical approximation based on density



Photon Map Data Structure

```
struct photon {
    float x, y, z;       // position
    char p[4];           // power packed as 4 chars
    char phi, theta;  // compressed incident direction
    short flag;          // flag used in kdtree
}
```


Reference Image


Photon Map

Types of Photon Maps

For efficiency reasons, it pays off to divide the stored photons into three photon maps:

Caustic photon map: contains photons that have been through at least one specula
reflection before hitting a diffuse surface: LS+DE.

Global photon map: an approximate representation of the global illumination solution for the scene for all diffuse surfaces: L{S|D|V }∗DE

Volume photon map: indirect illumination of a participating medium: L{S|D|V }+VE.

"L" stands for luminaire, "D" for diffuse reflection, "S" for specular reflection or refraction, "E" for eye

Optimising the photon map

Photons are only generated during the photon tracing pass — in the rendering pass the photon map is a static data structure that is used to compute estimates of the incoming flux and the reflected radiance at many points in the scene.

To do this it is necessary to locate the nearest photons in the photon map. This is an operation that is done extremely often, and it is therefore a good idea to optimise the representation of the photon map before the rendering pass such that finding the nearest photons is as fast as possible.

First, we need to select a good data structure for representing the photon map. The data structure should be compact and at the same time allow for fast nearest neighbour searching. It should also be able to handle highly non-uniform distributions — this is very often the case in the caustics photon map. A natural candidate that handles these requirements is a balanced kd-tree.

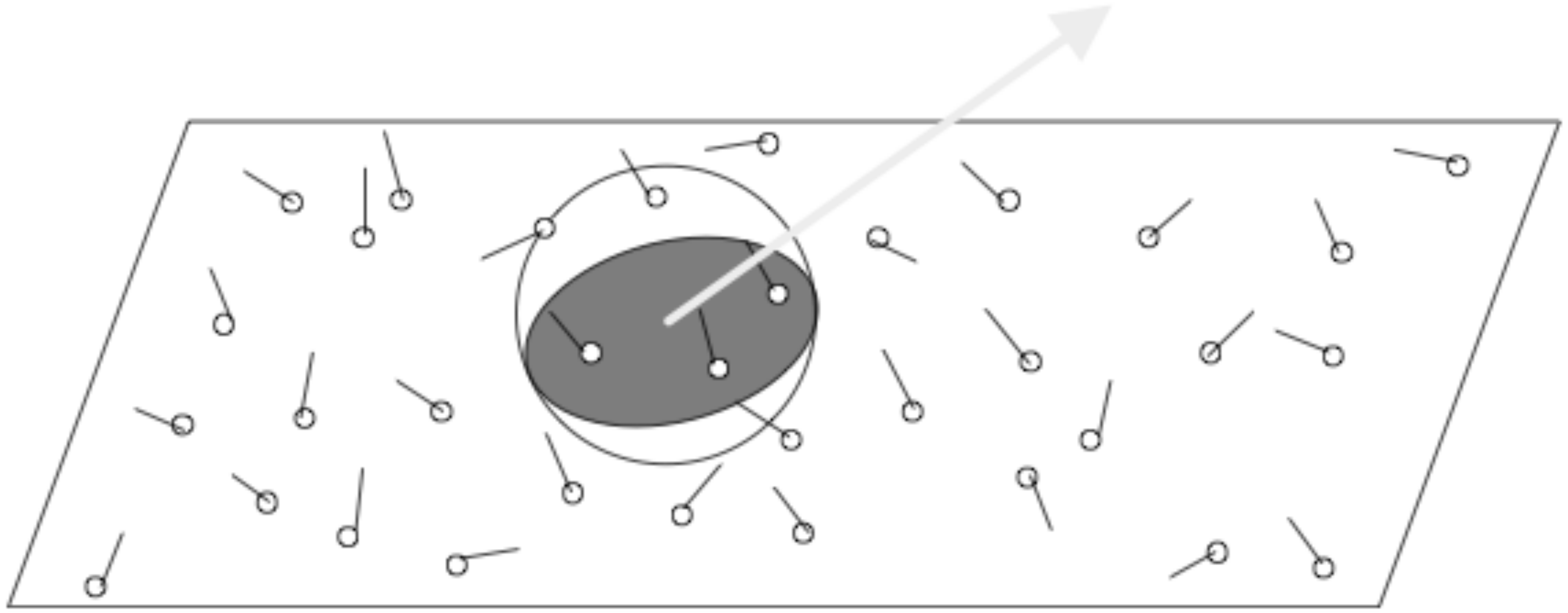
Balanced kd-tree data structure

```
kdtree *balance( points ) {
        Find the cube surrounding the points
        Select dimension dim in which the cube is largest Find median of the
        points in dim
        s1 = all points below median
        s2 = all points above median
        node = median
        node.left = balance( s1 )
        node.right = balance( s2 )
        return node
}
```

Worst time complexity O(log N)

The Radiance Estimate

A fundamental component of the photon map method is the ability to compute radiance estimates at any non-specular surface point in any given direction.



Radiance is estimated using the nearest photons in the photon map.

Radiance estimate at a surface

$$\lim_{N \to \infty} \frac{1}{\pi r^2} \sum_{p=1}^{\lfloor N^\alpha \rfloor} f_r(x, \omega_p, \omega) \Delta\Phi_p(x, \omega_p) = L_r(x, \omega) \ \text{ for } \alpha \in]0, 1[\,.$$

where Lr is the reflected radiance at x in direction ω. Ωx is the hemisphere of incoming directions, fr is the BRDF (bidirectional reflectance distribution function) at x.

Filtering

If the number of photons in the photon map is too low, the radiance estimates becomes blurry at the edges.

To reduce the amount of blur at edges, the radiance estimate is filtered.

Filters used are:-

The cone filter

The gaussian filter

Rendering

Given the photon map and the ability to compute a radiance estimate from it, we can proceed with the rendering pass.

The photon map is view independent, and therefore a single photon map constructed for an environment can be utilised to render the scene from any desired view.

Rendering Equation

The outgoing radiance, Lo is the sum of the emitted, Le and the reflected radiance Lr

$$L_o(x,\omega) = L_e(x,\omega) + L_r(x,\omega)$$

Reflected radiance Lr is computed by

$$L_r(x,\omega) = \int_{\Omega_x} f_r(x,\omega',\omega) L_i(x,\omega') \cos\theta_i \, d\omega_i'$$

where fr is the bidirectional reflectance distribution function (BRDF), and Ωx is the set of incoming directions around x.

The BRDF is separated into a sum of two components: A specular/glossy, fr,s, and a diffuse, fr,d

$$f_r(x,\omega',\omega) = f_{r,s}(x,\omega',\omega) + f_{r,d}(x,\omega',\omega)\,.$$

The incoming radiance is classified using three components:

• $L_{i,l}(x, \omega')$ is direct illumination by light coming from the light sources.
• $L_{i,c}(x,\omega')$ is caustics — indirect illumination from the light sources via

specular reflection or transmission.

• $L_{i,d}(x,\omega')$ is indirect illumination from the light sources which has been reflected diffusely at least once.

The incoming radiance is the sum of these three components:

$$L_i(x,\omega') = L_{i,l}(x,\omega') + L_{i,c}(x,\omega') + L_{i,d}(x,\omega')$$

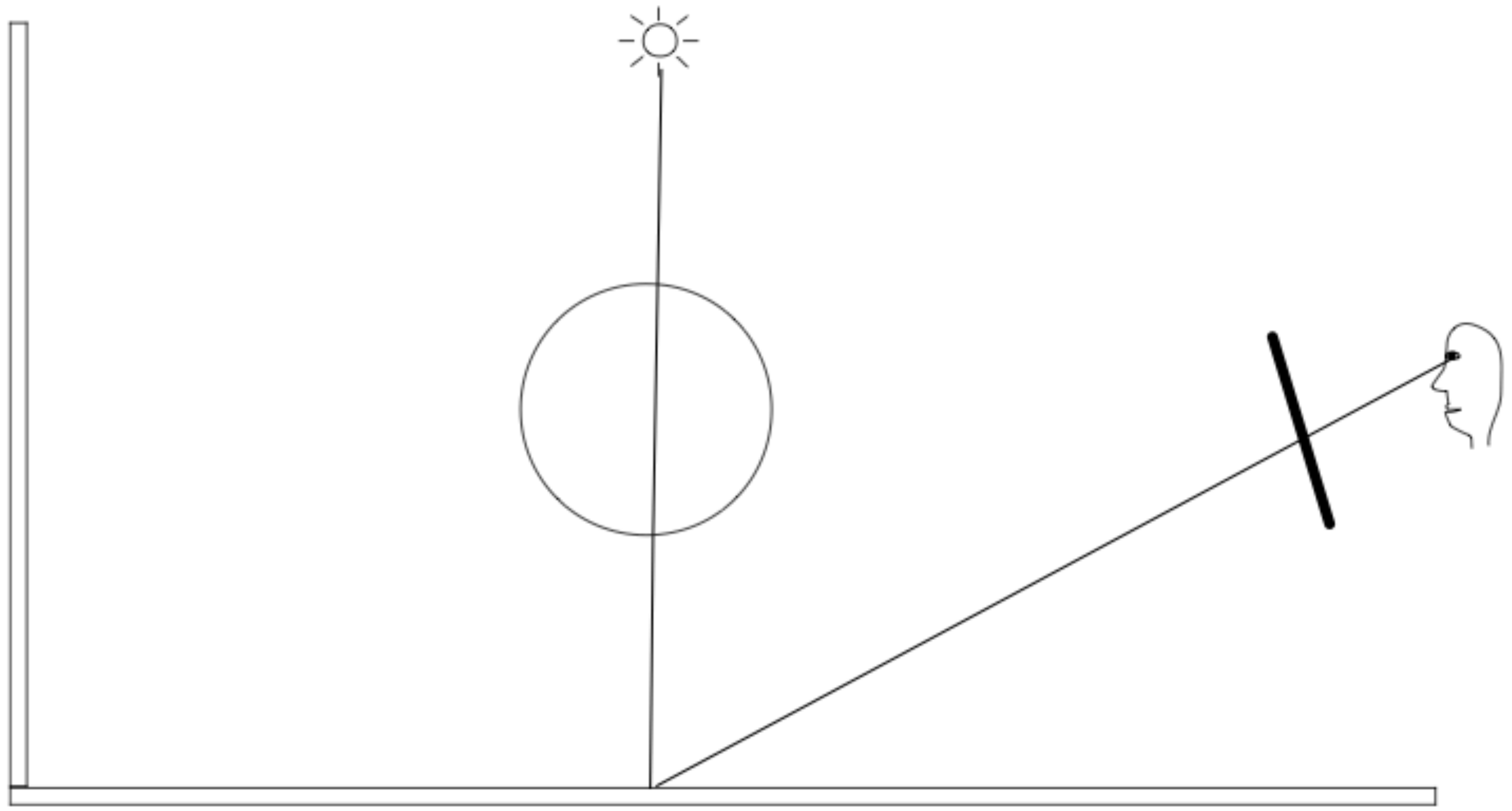
By using the classifications of the BRDF and the incoming radiance we can split the expression for reflected radiance into a sum of four integrals:

$$\begin{aligned} L_r(x,\omega) &= \int_{\Omega_x} f_r(x,\omega',\omega) L_i(x,\omega') \cos\theta_i \, d\omega_i' \\ &= \int_{\Omega_x} f_r(x,\omega',\omega) L_{i,l}(x,\omega') \cos\theta_i \, d\omega_i' + \\ &\quad \int_{\Omega_x} f_{r,s}(x,\omega',\omega)(L_{i,c}(x,\omega') + L_{i,d}(x,\omega')) \cos\theta_i \, d\omega_i' + \\ &\quad \int_{\Omega_x} f_{r,d}(x,\omega',\omega) L_{i,c}(x,\omega') \cos\theta_i \, d\omega_i' + \\ &\quad \int_{\Omega_x} f_{r,d}(x,\omega',\omega) L_{i,d}(x,\omega') \cos\theta_i \, d\omega_i' \, . \end{aligned}$$

Direct illumination

Direct illumination is given by the term

$$\int_{\Omega_x} f_r(x, \omega', \omega) L_{i,l}(x, \omega') \cos\theta_i \, d\omega_i'$$

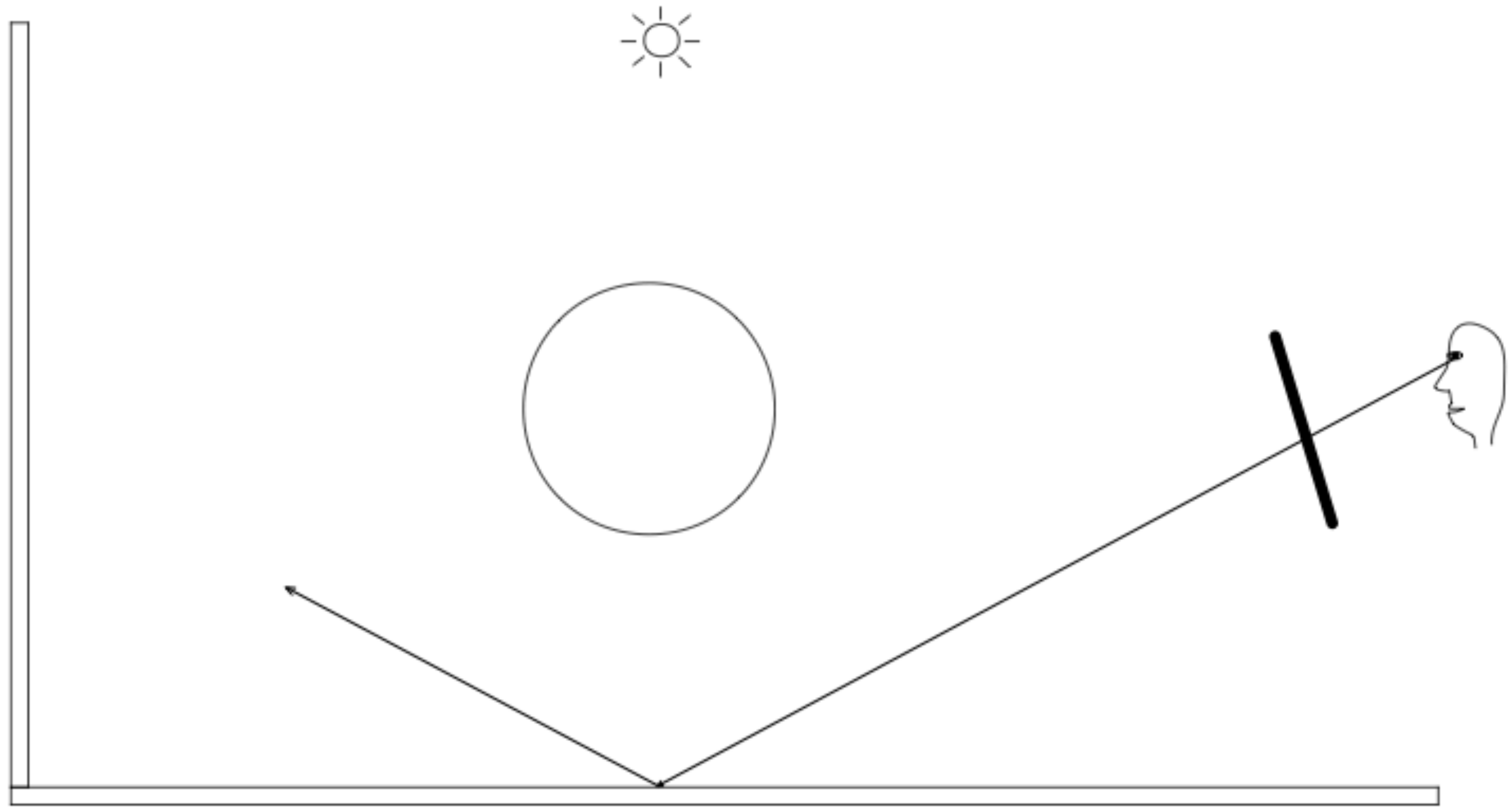


Accurate evaluation of the direct illumination.

Specular and glossy reflection

Specular and glossy reflection is computed by evaluation of the term

$$\int_{\Omega_x} f_{r,s}(x, \omega', \omega)(L_{i,c}(x, \omega') + L_{i,d}(x, \omega')) \cos\theta_i \, d\omega_i'$$

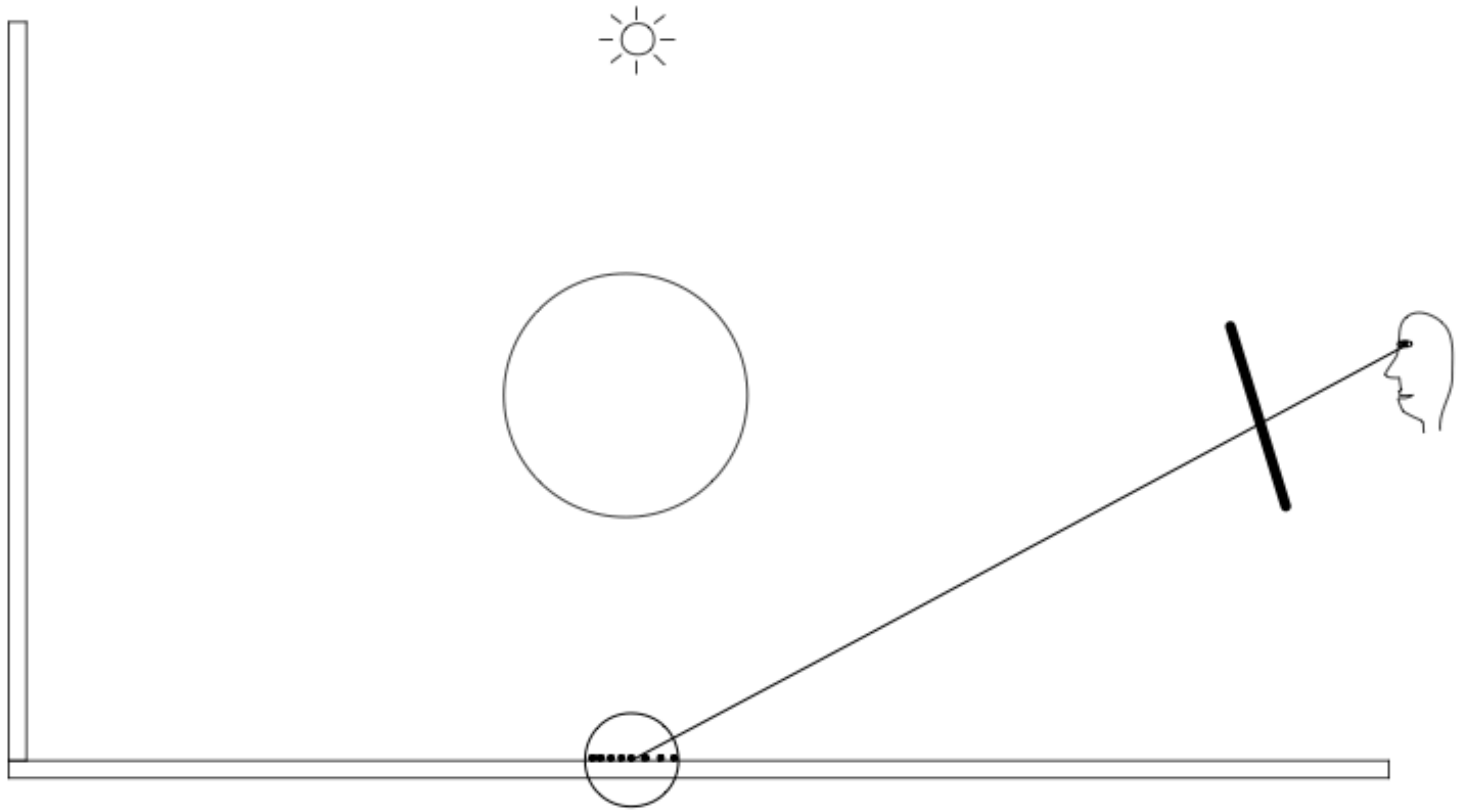


Rendering specular and glossy reflections.

Caustics

Caustics are represented by the integral

$$\int_{\Omega_x} f_{r,d}(x, \omega', \omega) L_{i,c}(x, \omega') \cos\theta_i \, d\omega_i'$$

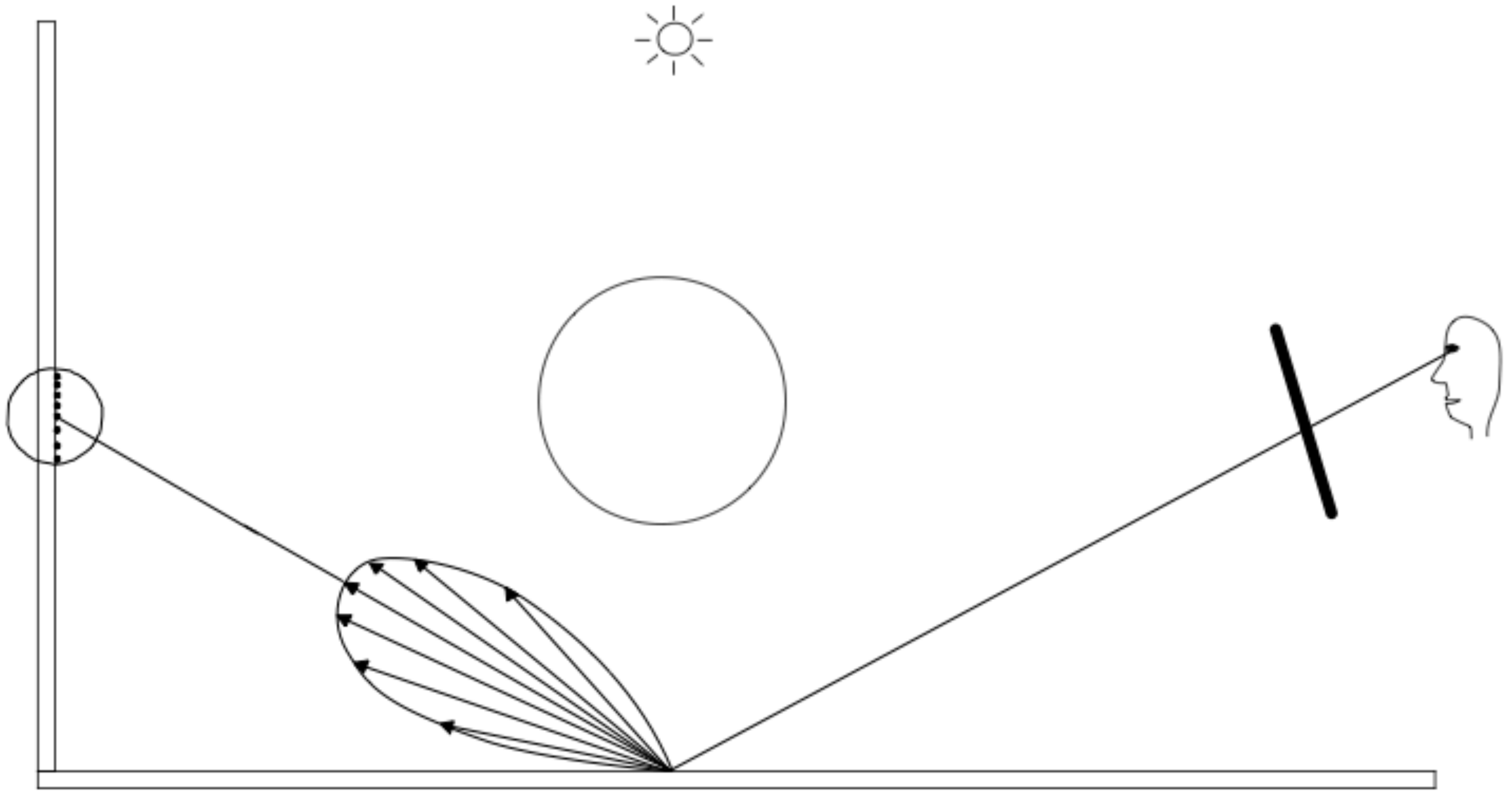


Rendering caustics

Multiple diffuse reflections

Multiple diffuse reflections are given by

$$\int_{\Omega_x} f_{r,d}(x, \omega', \omega) L_{i,d}(x, \omega') \cos\theta_i \, d\omega_i'$$


Computing indirect diffuse illumination with importance sampling.

Advantages of Photon Mapping

using photons to simulate the transport of individual pho- ton energy

being able to calculate global illumination effects

capable of handling arbitrary geometry rather than polygonal scenes

low memory consumption

producing correct rendering results, even though noise could be introduced

Further Reading

Photon Mapping - introduced in 1994 by Henrik Wann.

Progressive PM - Toshiya Hachisuka , H. W. Jensen , 2008

Stochastic PPM - T. Hachisuka, H. W. Jensen, 2009

Both PPM and SPPM slightly improve the previous one.

References

https://courses.cs.washington.edu/courses/cse557/09au/projects/final/artifacts/jca87/Writeup.htm

https://courses.cs.washington.edu/courses/cse557/08wi/projects/final/artifacts/jjohn-isdal/

Thank You