POLITECNICO
MILANO 1863

DIPARTIMENTO DI ELETTRONICA
INFORMAZIONE E BIOINGEGNERIA

DEIB

2024

Dipartimento di Elettronica, Informazione e Bioingegneria

Computer Graphics

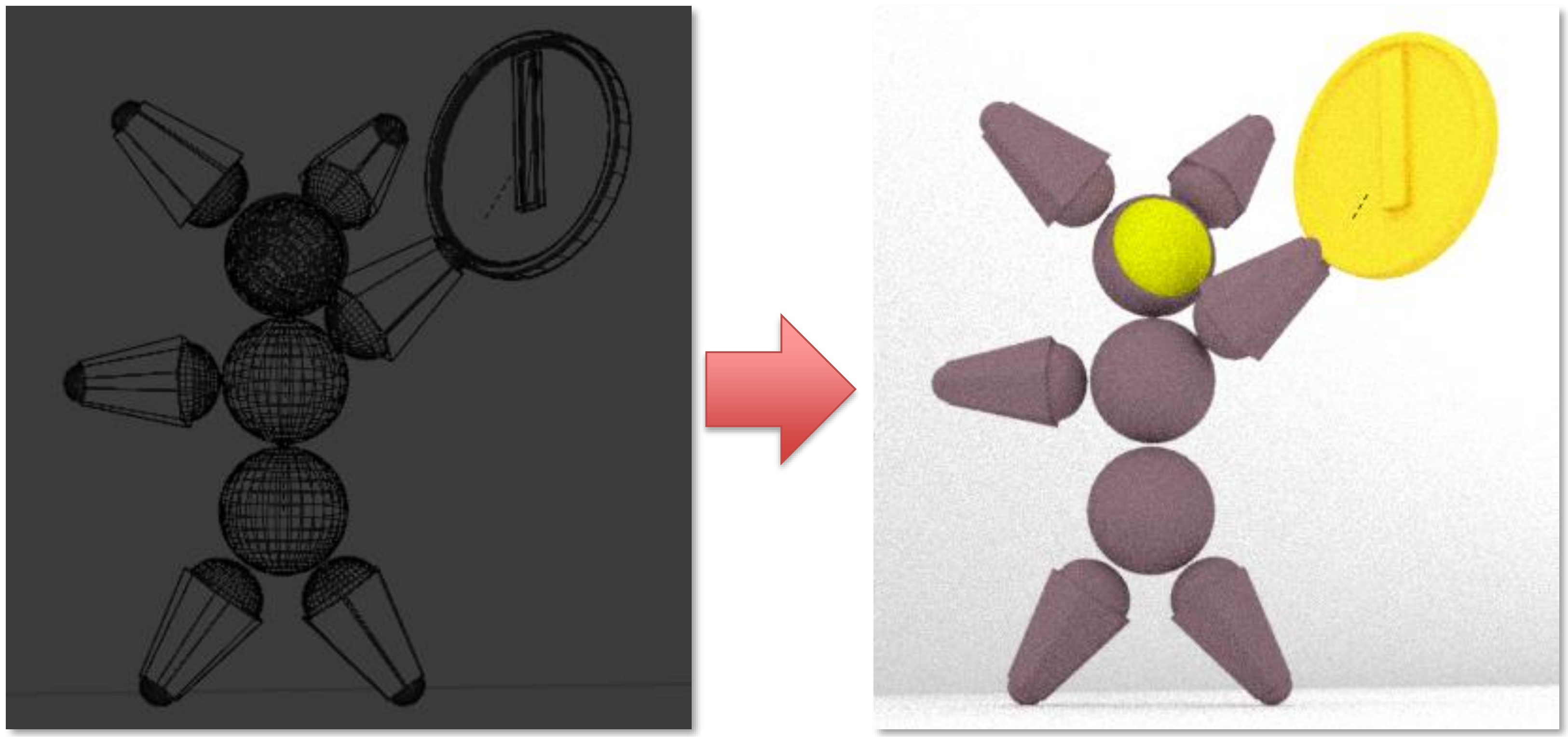
Milano, 2024

Computer Graphics

- Rendering

Rendering

The process of creating realistic 3D images with accurate colors is called *Rendering*.



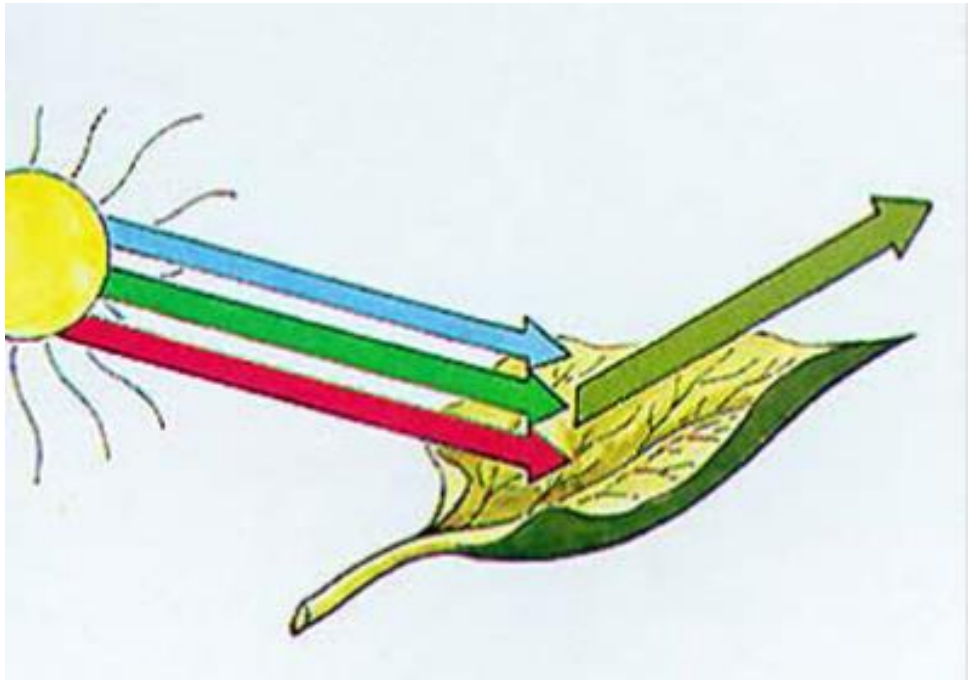
Rendering

To obtain realistic images with filled surfaces (i.e. not just wireframe), the reflection the light should be correctly emulated.

Rendering reproduces the effects of the illumination by defining the light sources of the virtual environment, and the surface properties of the objects populating the 3D world.

Rendering

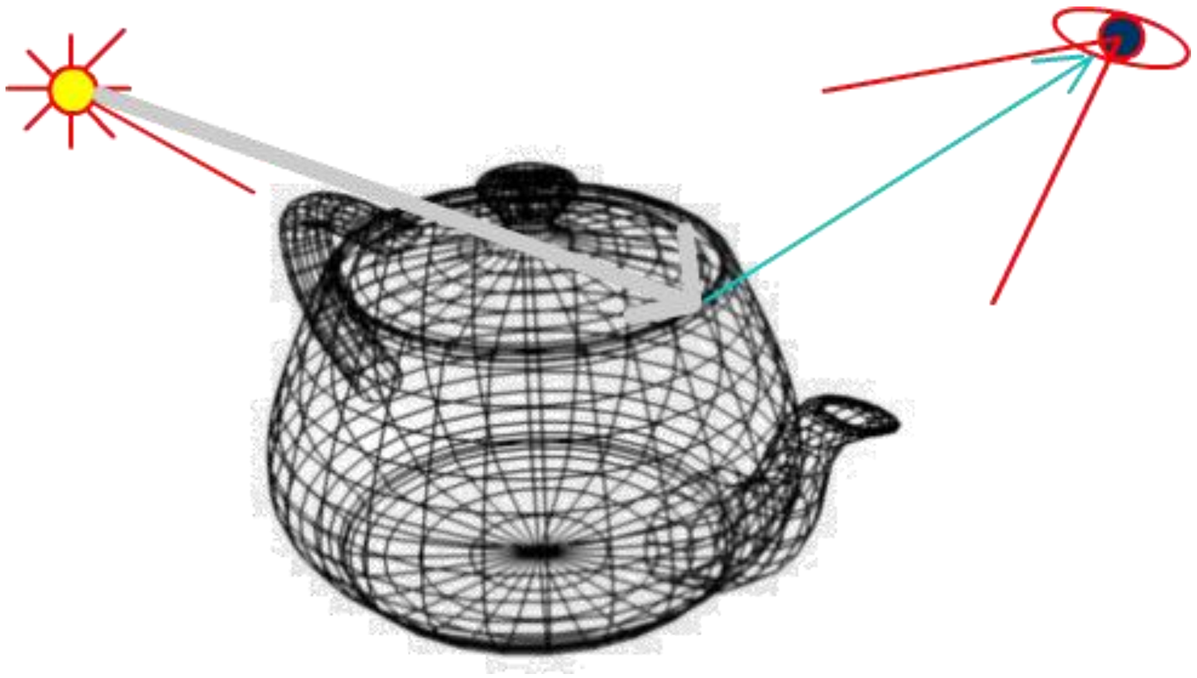
The Lights sources are elements of the scene from which illumination starts: as introduced, the light sources emit different frequencies of the spectrum, and objects reflect part of them.



Rendering

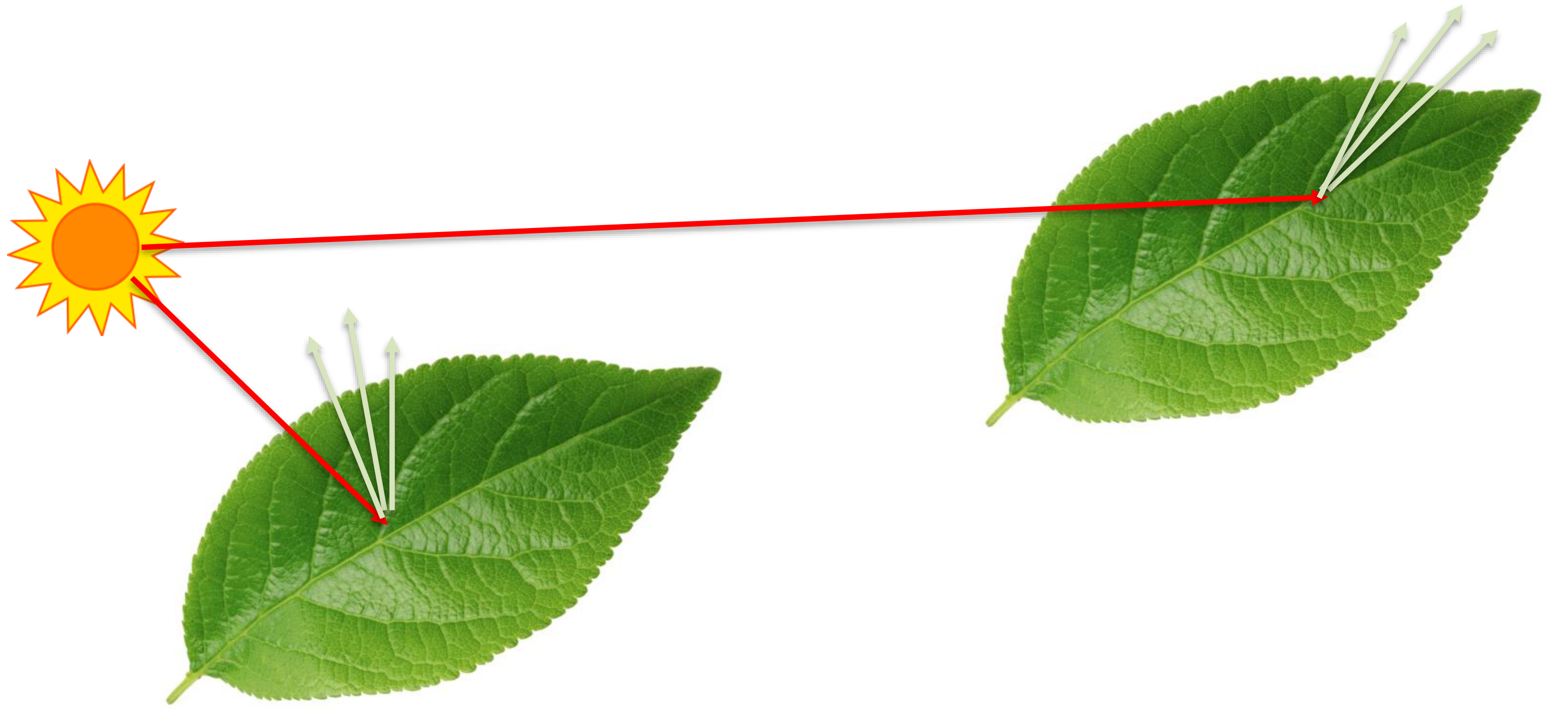
The photons emitted by the light sources bounce on the objects, and some of them reach the viewpoint (the camera).

Rendering computes the intensity and the color of such photons.



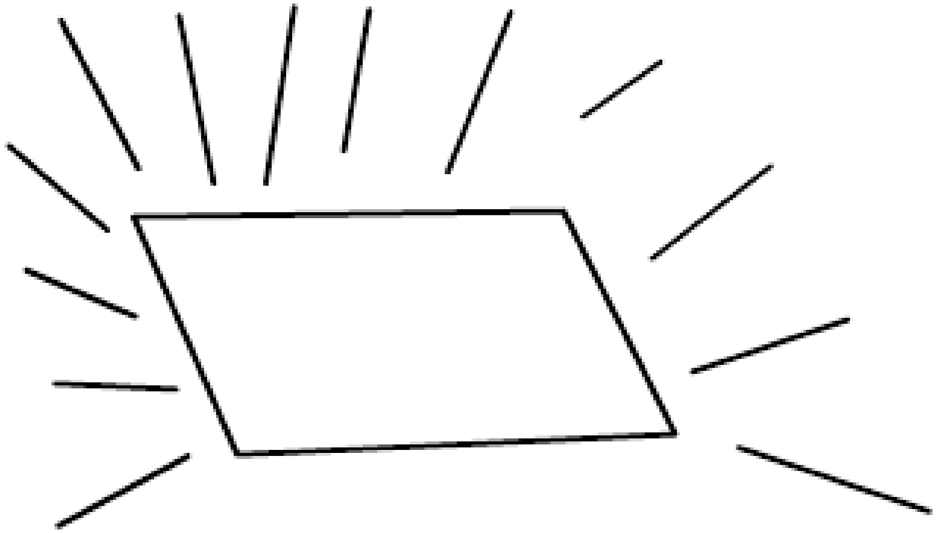
Rendering

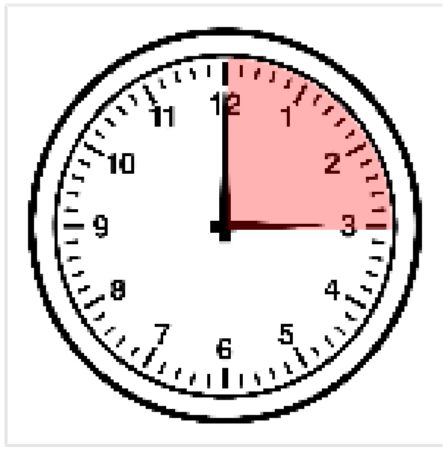
The quantity of light reflected depends on the input and output directions, since incoming photons can bounce in many different ways.



Radiance

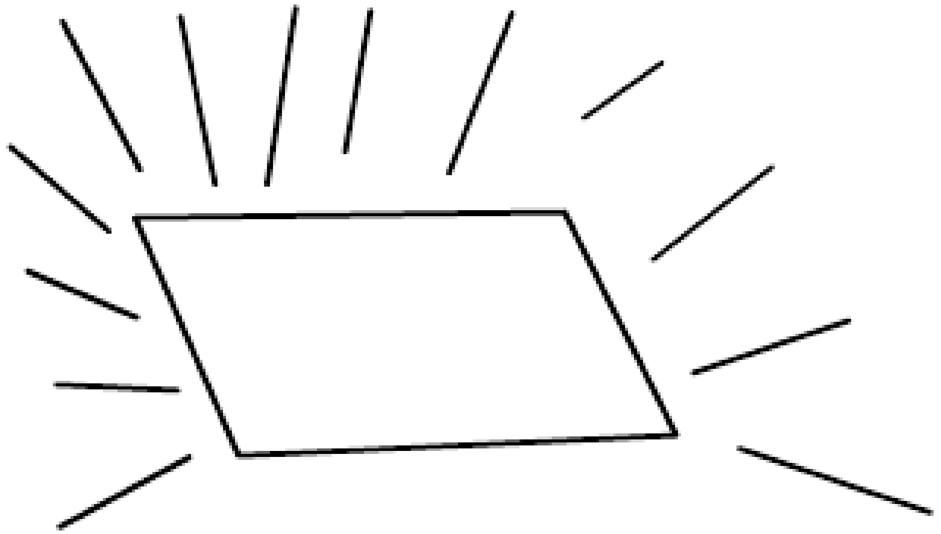
The *energy* (expressed in *Joules - J*) measures the total light emitted by a surface in all the directions during a time interval.

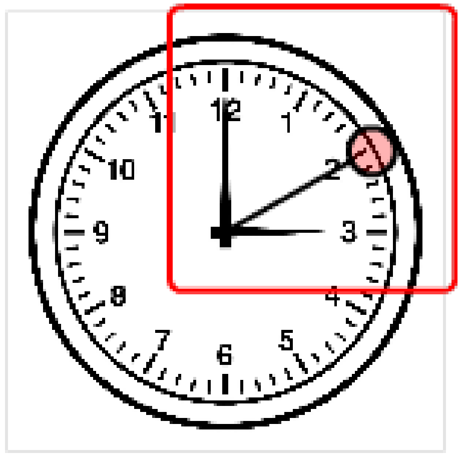




Radiance

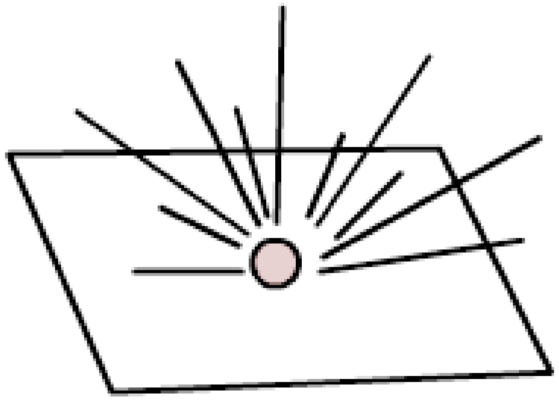
The *power* (measured in *Watts* - $W = J \cdot s^{-1}$) is the instantaneous light energy (emitted by a surface in all the directions in a given time instant).

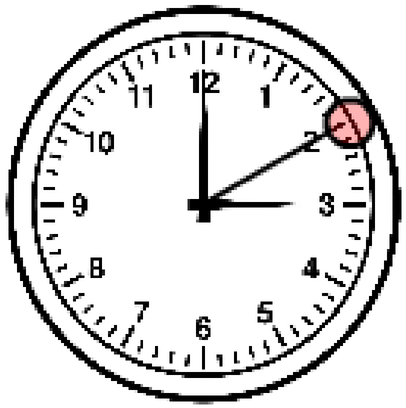




Radiance

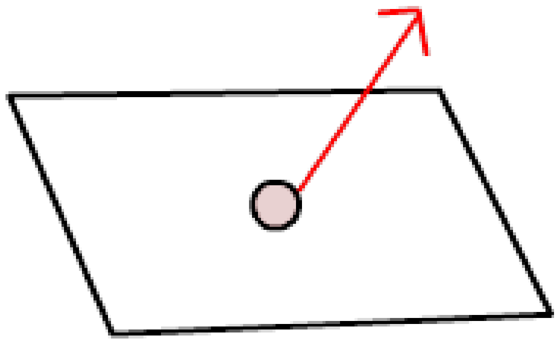
The *irradiance* is the fraction of power emitted by a point of a surface (in a given time instant). It is measured in W / m^2. In the following, it will be denoted by letter E.

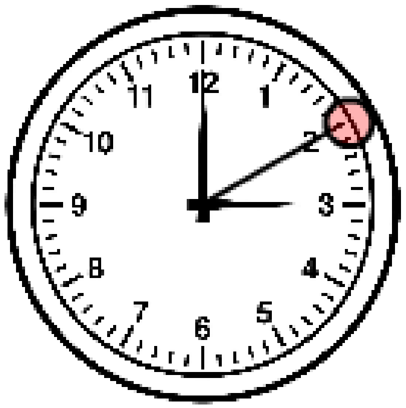




Radiance

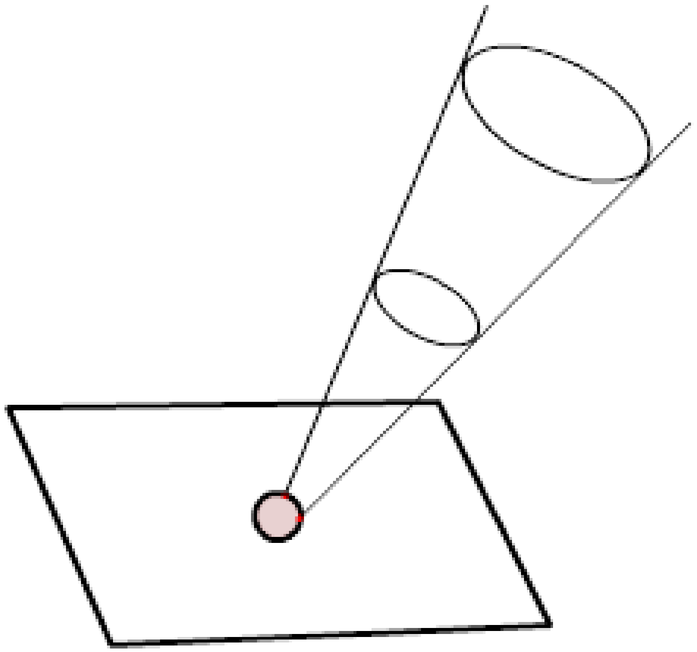
Radiance measures the energy emitted in a given time instant from a point of a surface in a given direction. It is measured in $W / (m^2 \cdot sr)$ (where *sr* are *steradians*, the unit of measure for the solid-angle).
In the following, it will be denoted by letter *L*.

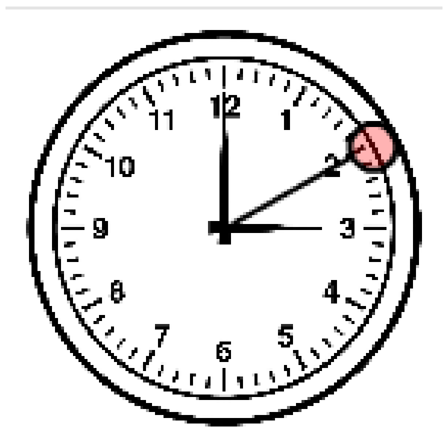




Radiance

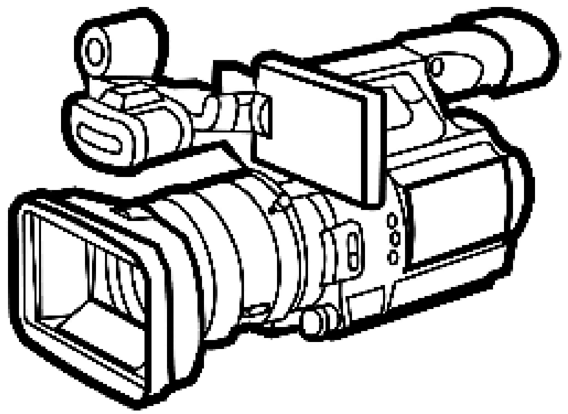
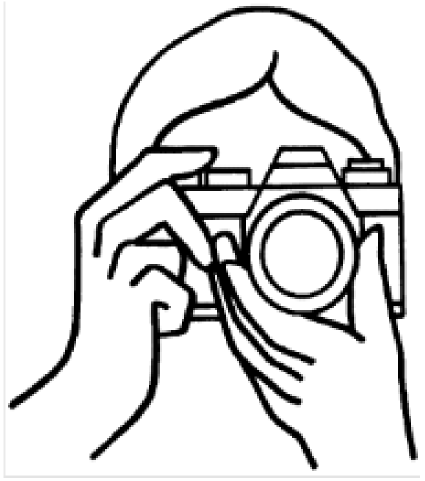
The use of solid angles, allows the *radiance* to be independent from the distance at which the object is observed.





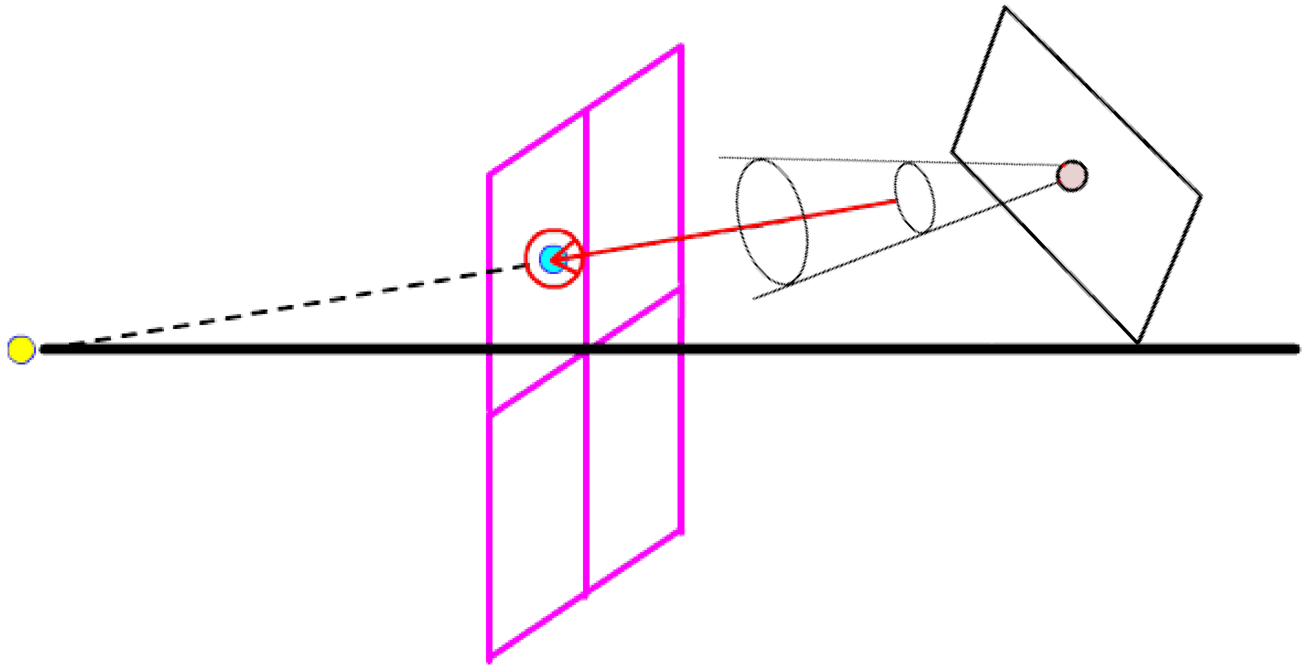
Radiance

Readings of most light sensors (including the human eyes and the cameras) are proportional to the *radiance*.

Radiance

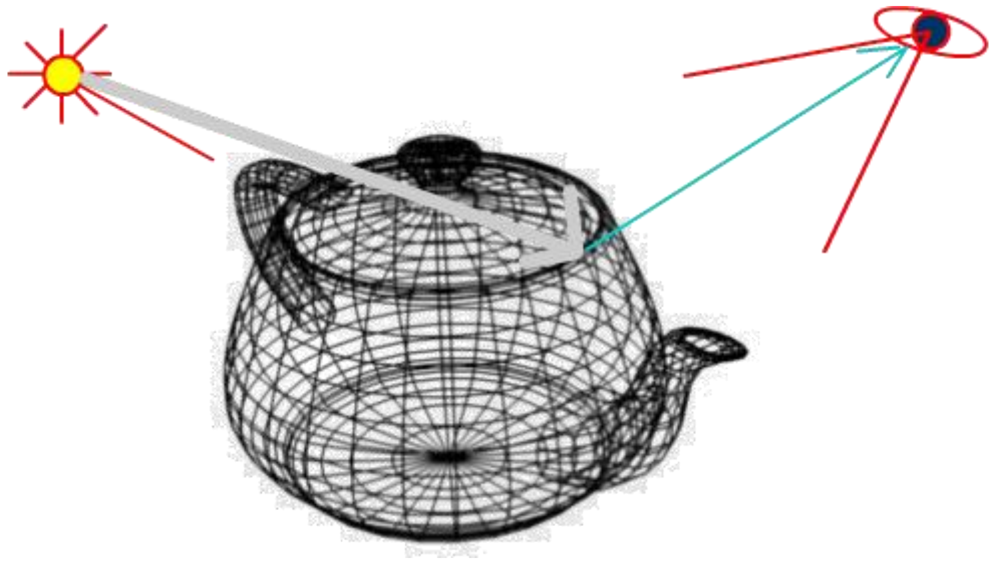
Rendering determines the *radiance* received in *each point* of the *projection plane* (i.e each pixel of the screen) according to the direction of the corresponding *projection ray*.



The Bidirectional Reflectance Distribution Function

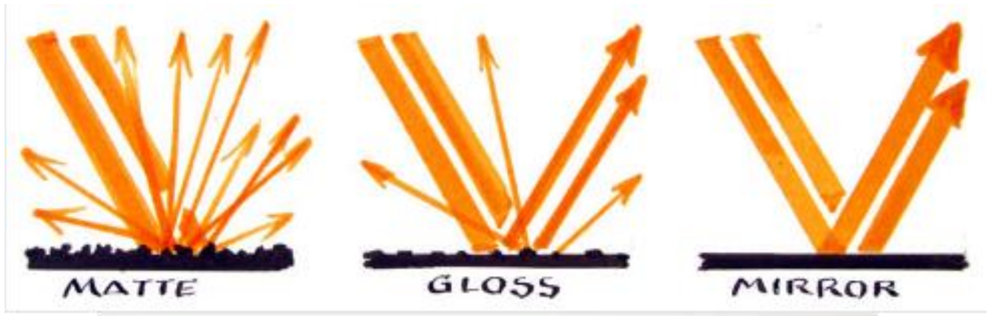
Light emitted by the sources is characterized by its radiance.

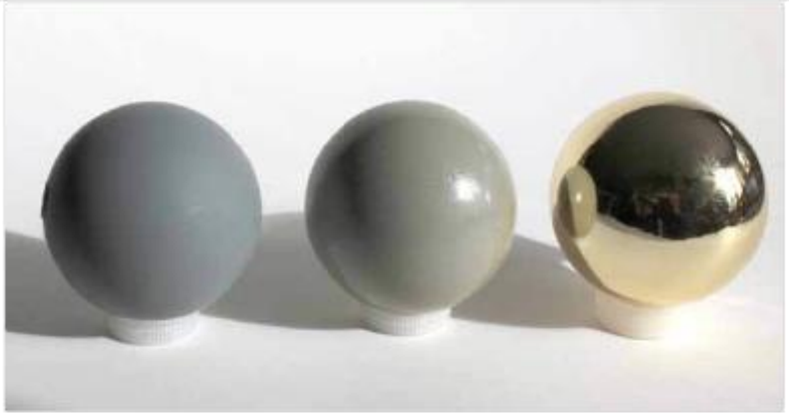
The material that composes the surface of an object determines the intensity of the reflected light in a given direction.



The Bidirectional Reflectance Distribution Function

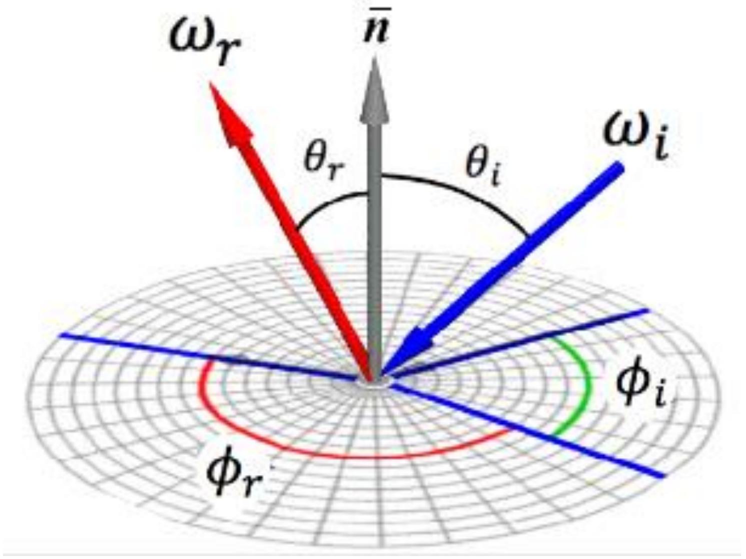
In practice, the direction of the bounces depends on the microscopic structure of the surface: it can be very different from material to material.





The Bidirectional Reflectance Distribution Function

The surface properties of one object can be described by a function called the *Bidirectional Reflectance Distribution Function* (*BRDF*).

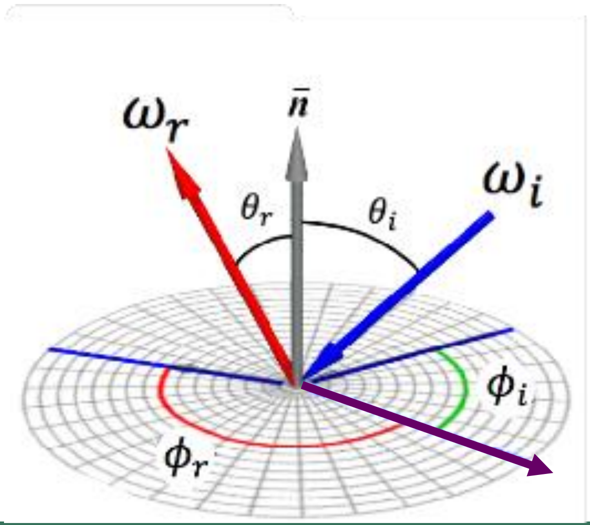


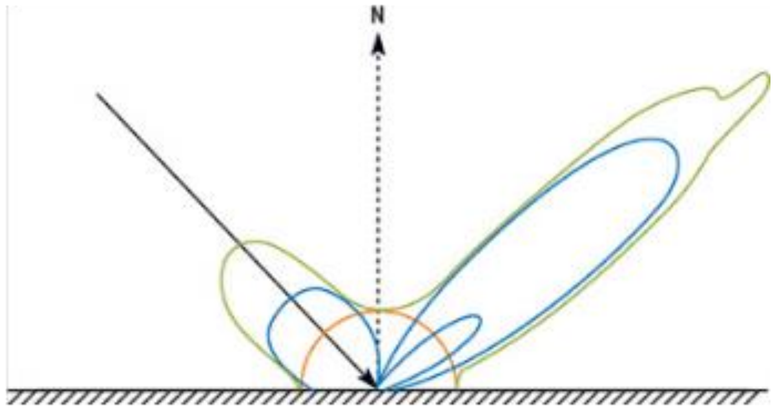
The Bidirectional Reflectance Distribution Function

The function inputs are the incoming ω_i and outgoing ω_r light directions; they are both unit vectors.

The function tells how much *irradiance* from the incoming angle, is reflected to an outgoing angle. It is measured in *sr* $^{-1}$.

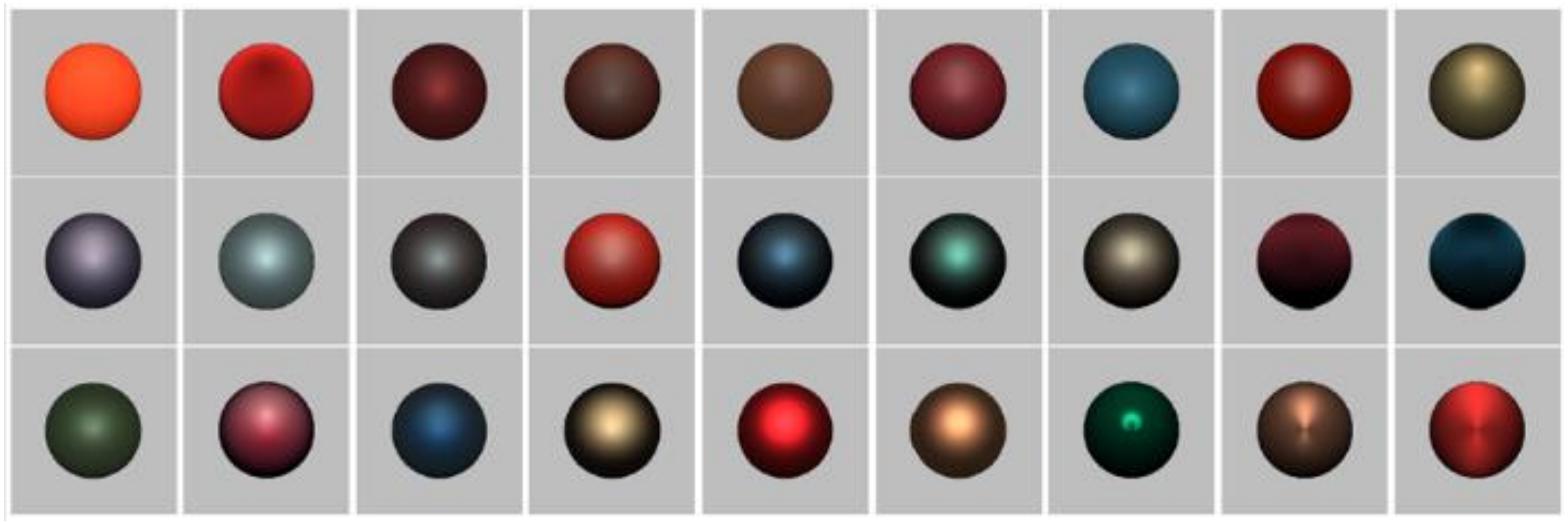
$$f_r(\theta_i, \phi_i, \theta_r, \phi_r) = f_r(\omega_i, \omega_r)$$





The Bidirectional Reflectance Distribution Function

This allows considering different ways in which the incoming radiance can be reflected at the different angles, depending on the surface material.



The Bidirectional Reflectance Distribution Function

A good BRDF should satisfy two main properties: *Reciprocity* and *Energy conservation*.

Reciprocity means that if the ingoing and outgoing directions are swapped, the value of the function does not change (for this reason this is known as *bidirectional*).

$$f_r(\omega_i, \omega_r) = f_r(\omega_r, \omega_i)$$

Energy conservation means that the BRDF cannot increase the total irradiance that leaves a point on a surface.

$$\int f_r(\omega_i, \omega_r) \cos\theta_r \, d\omega_r \leq 1$$

This is the reason why the BRDF is measured in *sr*$^{-1}$: when integrating over all the angles, the result is a dimensionless number.

The integral can be less than one if the point *absorbs* energy.

The Bidirectional Reflectance Distribution Function

Several approximations to the most common BRDF functions have been proposed in the literature: some of them can provide good results during rendering, even if they do not satisfy the two previous properties.

In the following lessons, we will present several common BRDF functions and how can they be implemented real time.

Databases that provides measured BRDF exist: see for example the *MERL database at*

http://www.merl.com/brdf/

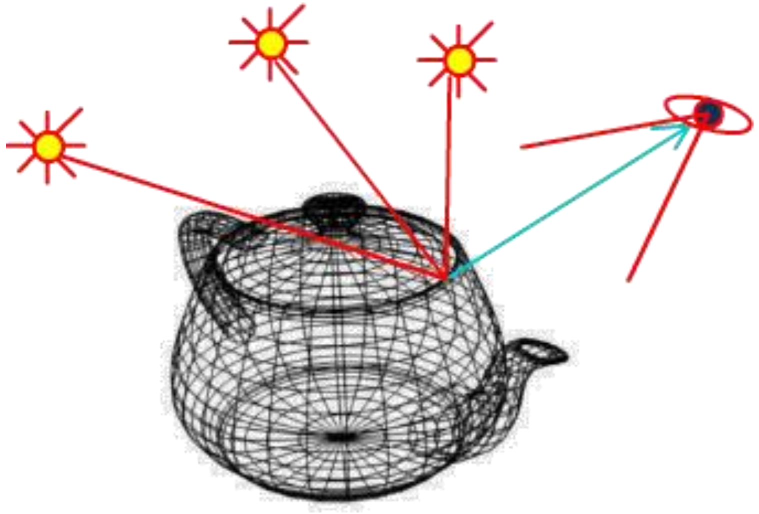
The rendering equation

The BRDF allows relating together the irradiance in all the directions for all the points of the objects composing a scene. This relation is called the *Rendering Equation*:

$$L(x,\omega_r) = L_e(x,\omega_r) + \int L(y,\overrightarrow{yx}) f_r(x,\overrightarrow{yx},\omega_r) G(x,y) V(x,y) dy$$

The rendering equation

The equation determines the radiance of each point *x* of any object in the 3D world, for any output direction *ω*.



$$\boxed{L(x,\omega_r)} = L_e(x,\omega_r) + \int L(y,\overrightarrow{yx}) f_r(x,\overrightarrow{yx},\omega_r) G(x,y) V(x,y)\,dy$$

The rendering equation

Objects can emit light: term $L_e()$ accounts for the light that the object at x emits in direction ω.

$$L(x,\omega_r) = \boxed{L_e(x,\omega_r)} + \int L(y,\overrightarrow{yx}) f_r(x,\overrightarrow{yx},\omega_r) G(x,y) V(x,y)\, dy$$

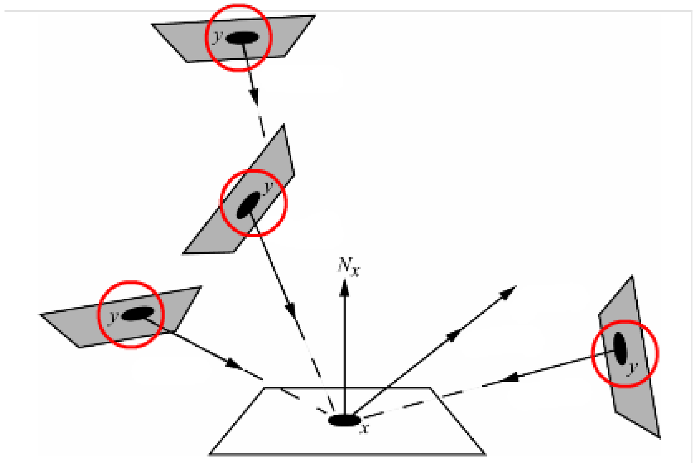
The rendering equation

This parameter mainly characterizes light sources (i.e. a bulb or a neon), and it is the way to “start” the process, by injecting photons in the scene.

$$L(x,\omega_r) = \boxed{L_e(x,\omega_r)} + \int L(y,\overrightarrow{yx})f_r(x,\overrightarrow{yx},\omega_r)G(x,y)V(x,y)dy$$

The rendering equation

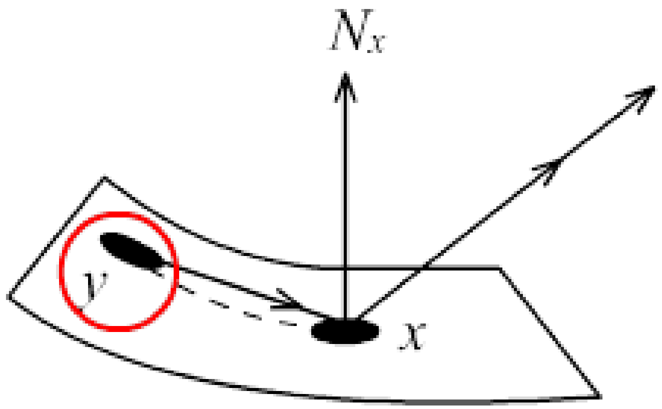
The integral accounts for the light that hits the considered point *x* from all the points *y* of the surfaces (including the light sources) of all the objects and lights in the scene.



$$L(x,\omega_r) = L_e(x,\omega_r) + \boxed{\int} L(y,\overrightarrow{yx}) f_r(x,\overrightarrow{yx},\omega_r) G(x,y) V(x,y) \boxed{dy}$$

The rendering equation

The integral also includes other points of the same object to allow the computation of effects such as *self-shadowing* or *self-reflection*.

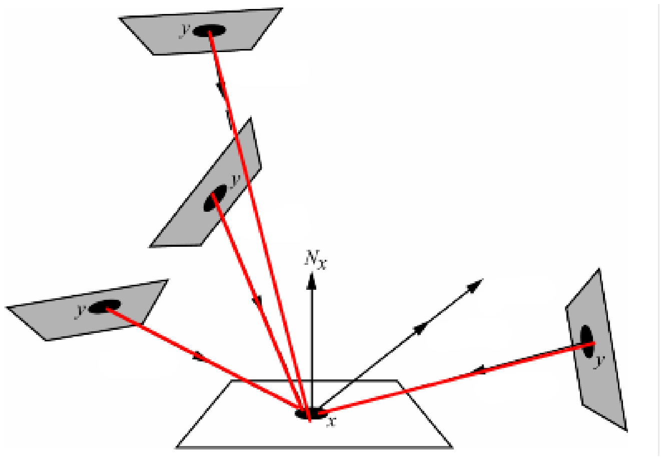


$$L(x,\omega_r) = L_e(x,\omega_r) + \boxed{\int} L(y,\vec{yx}) f_r(x,\vec{yx},\omega_r) G(x,y) V(x,y) \boxed{dy}$$

The rendering equation

For each object, the equation considers the radiance emitted toward point *x*.

Here $\overrightarrow{yx}$ represents the direction of the line that connects point *y* to point *x*.

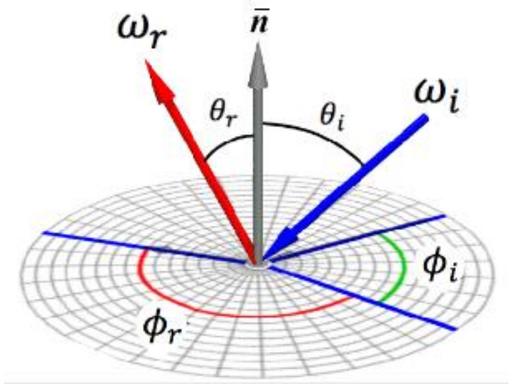


$$L(x,\omega_r) = L_e(x,\omega_r) + \int \boxed{L(y,\overrightarrow{yx})} f_r(x,\overrightarrow{yx},\omega_r) G(x,y) V(x,y) dy$$

The rendering equation

f_r() is the *BRDF* of the material of the object at point *x*.

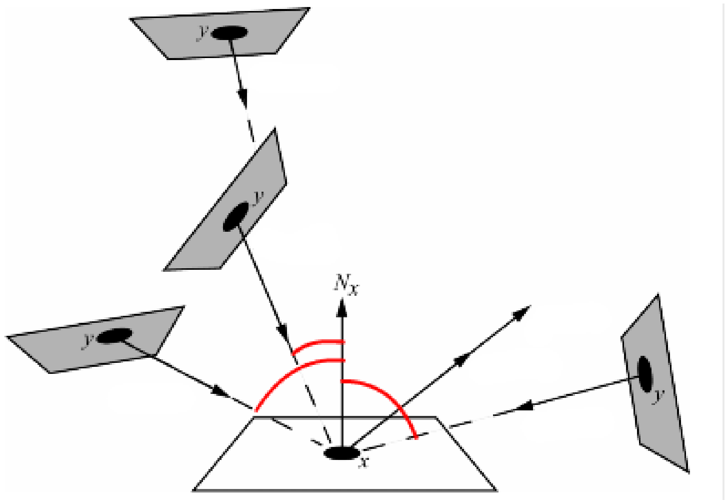
Since the materials of objects might change over the surface, the BRDF depends also on position of the considered point *x*.



$$L(x,\omega_r) = L_e(x,\omega_r) + \int L(y,\overrightarrow{yx})\boxed{f_r(x,\overrightarrow{yx},\omega_r)}G(x,y)V(x,y)dy$$

The rendering equation

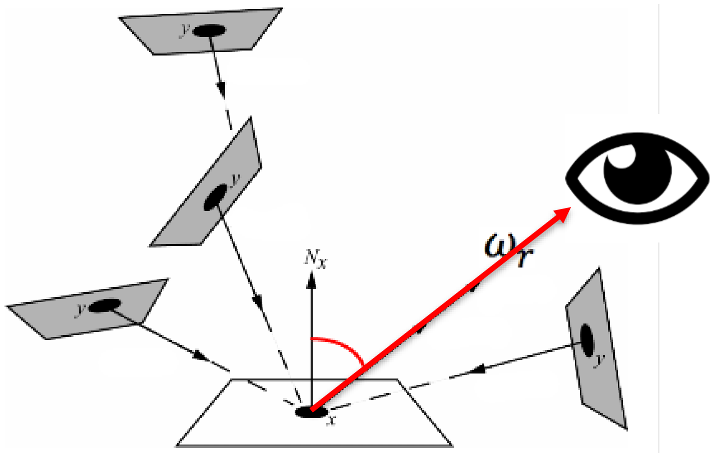
The input angle corresponds to the direction of the incoming light, oriented along the segment that connects *y* to *x*.



$$L(x,\omega_r) = L_e(x,\omega_r) + \int L(y,\overrightarrow{yx}) f_r(x,\boxed{\overrightarrow{yx}},\omega_r) G(x,y) V(x,y)\, dy$$

The rendering equation

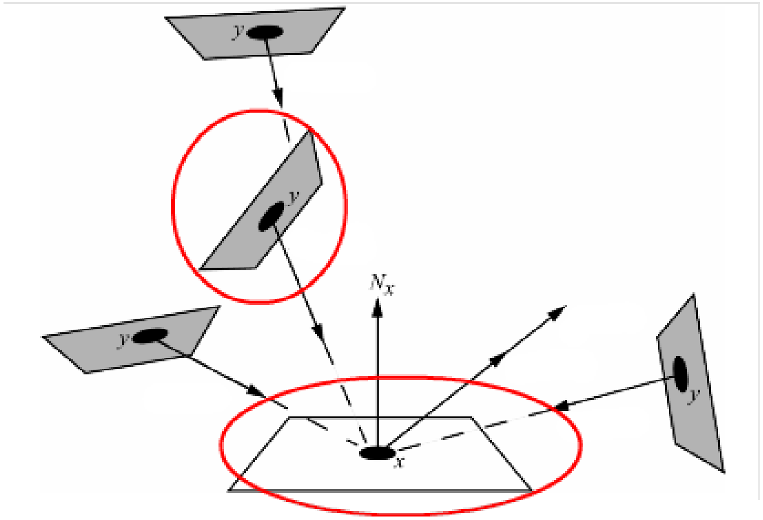
The output angle corresponds to the direction from which the output radiance is being computed.
It is parameter ω_r on the left hand side of the equation.



$$L(x,\boxed{\omega_r}) = L_e(x,\omega_r) + \int L(y,\vec{yx})f_r(x,\vec{yx},\boxed{\omega_r})G(x,y)V(x,y)dy$$

The rendering equation

Factor *G(x,y)* encodes the geometric relation between points *x* and *y*. This is necessary, because the angle between the surfaces has an impact on both the light emitted and reflected.



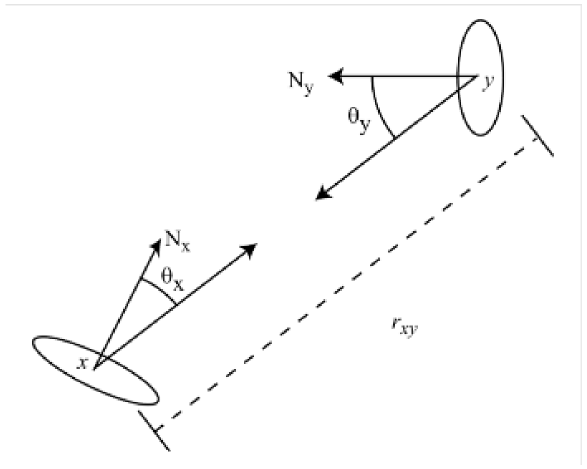
$$L(x,\omega_r) = L_e(x,\omega_r) + \int L(y,\vec{yx}) f_r(x,\vec{yx},\omega_r) \boxed{G(x,y)} V(x,y) dy$$

The rendering equation

It considers both the relative orientation and the distance of the two points, and it is defined as shown below.

The two *cos()* terms accounts for the angle relative to the respective normal vectors, and r_{xy}^2 represents the squared distance of the two points.

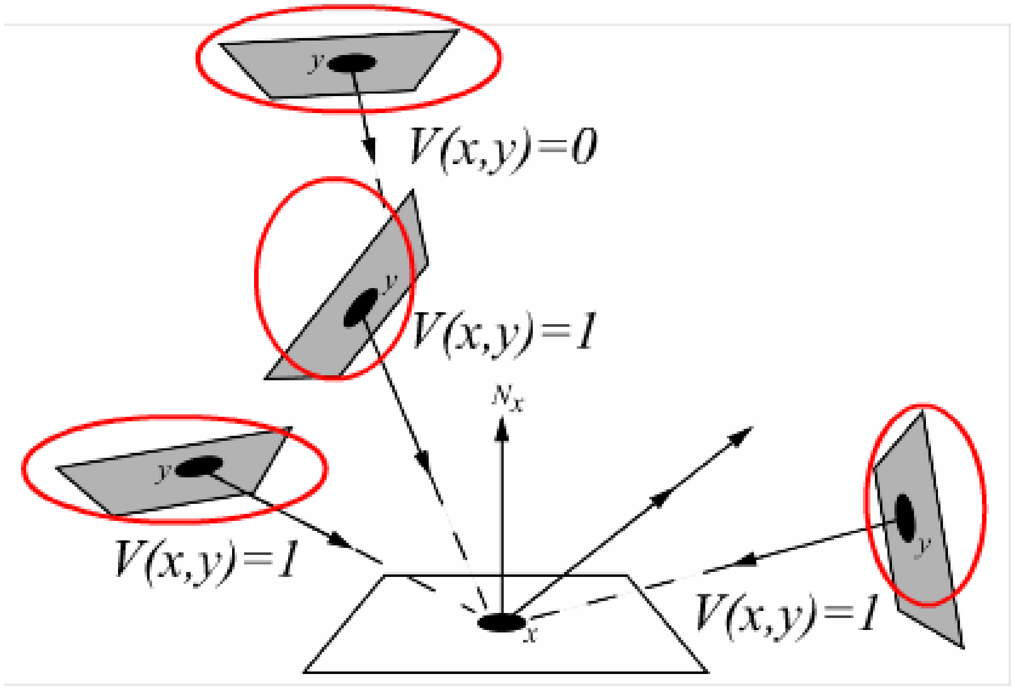
$$G(x,y) = \frac{\cos\theta_x \cos\theta_y}{r_{xy}^2}$$



$$L(x,\omega_r) = L_e(x,\omega_r) + \int L(y,\overrightarrow{yx}) f_r(x,\overrightarrow{yx},\omega_r) \boxed{G(x,y)} V(x,y)\,dy$$

The rendering equation

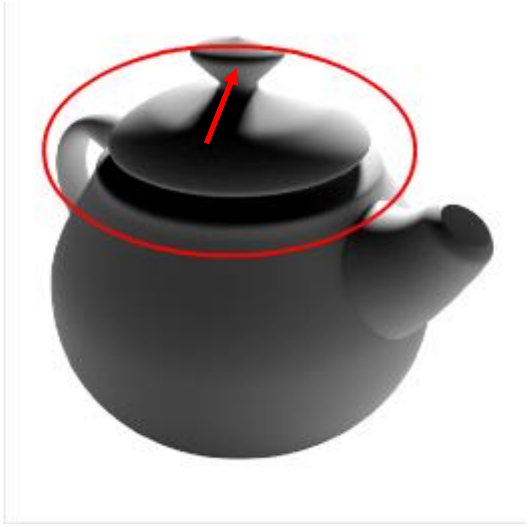
Finally, term *V(x,y)* considers the *visibility* between two points x and y: *V(x,y) = 1* if the two points can see each other, and *V(x,y) = 0* if point y is hidden by some other objects in between.



$$L(x,\omega_r) = L_e(x,\omega_r) + \int L(y,\vec{yx}) f_r(x,\vec{yx},\omega_r) G(x,y) \boxed{V(x,y)} dy$$

The rendering equation

Term V(x,y) allows for the computation of shadows, and makes sure that in each input direction, *at most* a single object is considered.



$$L(x,\omega_r) = L_e(x,\omega_r) + \int L(y,\overrightarrow{yx}) f_r(x,\overrightarrow{yx},\omega_r) G(x,y) \boxed{V(x,y)} dy$$

The rendering equation

Term *L(x, ω)* is the unknown of the equation.

Since it appears on both sides of the expression, the rendering equation is called, using a mathematical notation, an *integral equation of the second kind*.

$$\varphi(x) = f(x) + \lambda \int_a^b K(x,t)\,\varphi(t)\,dt.$$

$$\boxed{L(x,\omega_r)} = L_e(x,\omega_r) + \int \boxed{L(y,\overrightarrow{yx})} f_r(x,\overrightarrow{yx},\omega_r) G(x,y) V(x,y) dy$$

Considering colors

The rendering equation is repeated for every wavelength λ of the light: usually this means that the equation is repeated for the three different RGB channels.

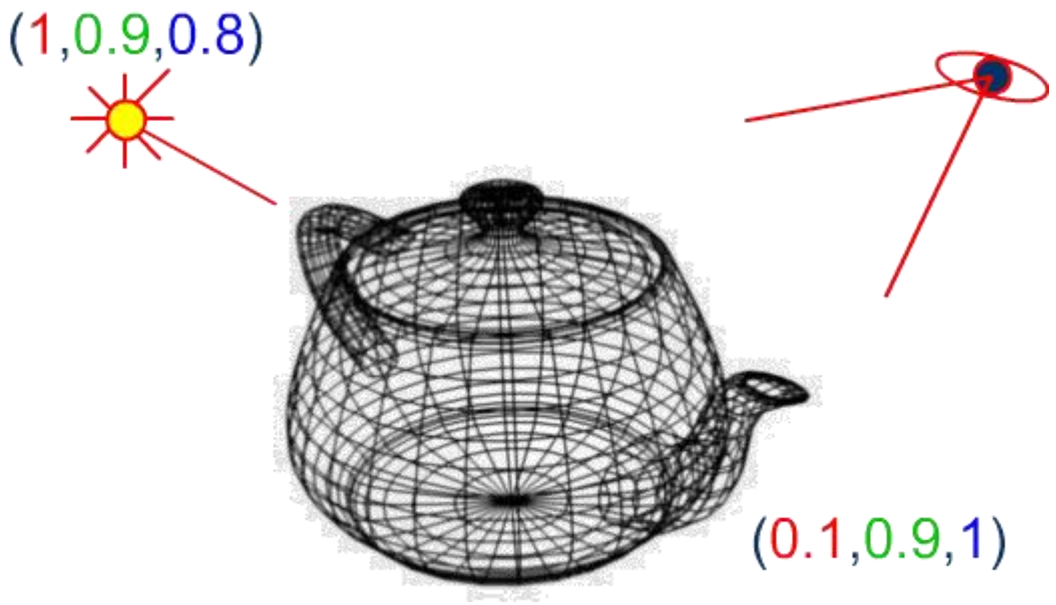
The geometric and visibility terms do not depend on the wavelength, and are unique for every couple of points.

$$L(x,\omega_r,\lambda) = L_e(x,\omega_r,\lambda) + \int L(y,\vec{yx},\lambda) f_r(x,\vec{yx},\omega_r,\lambda) G(x,y) V(x,y) dy$$

Considering colors

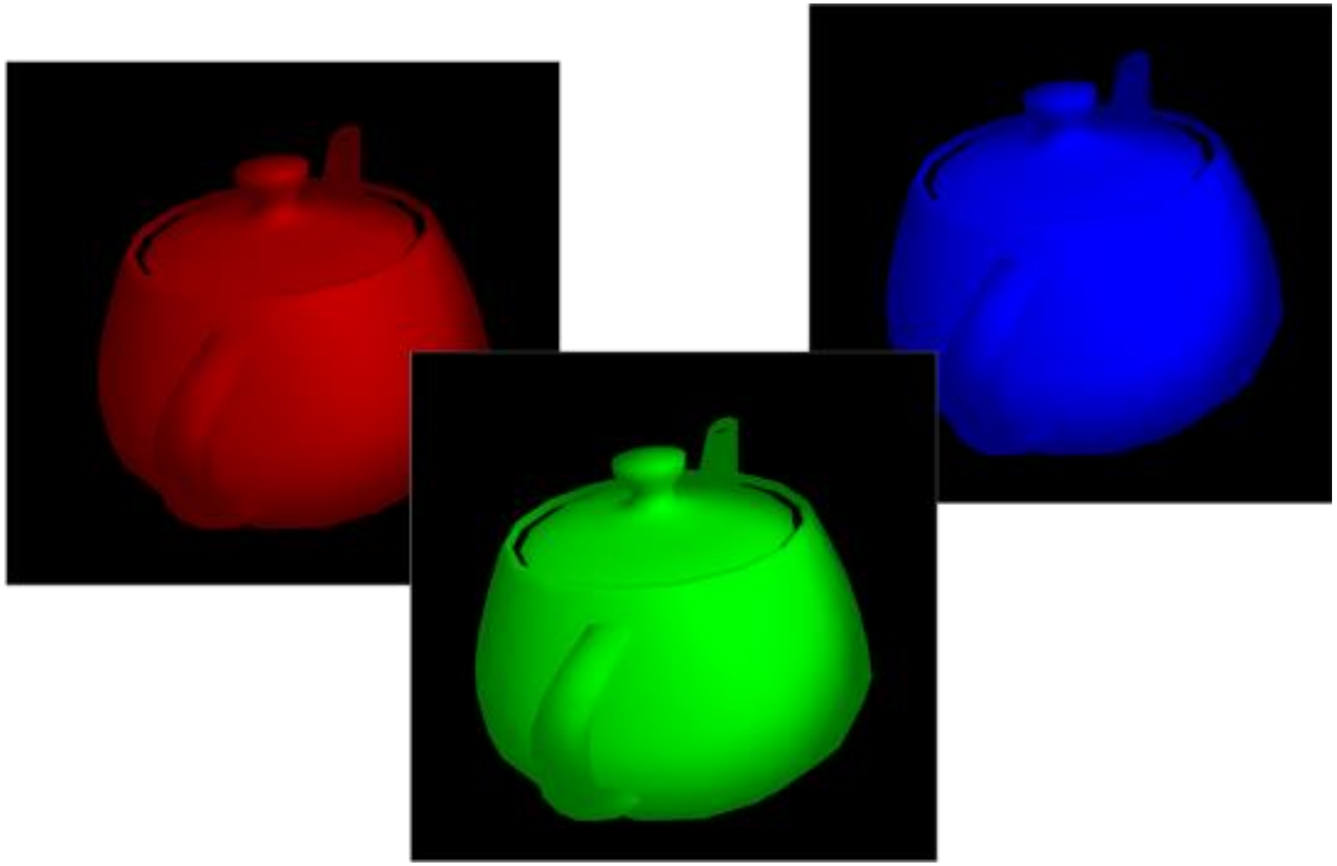
In particular, the lights have associated the RGB values that accounts for the photons emitted for each of the three main frequencies.

The objects are characterized by a BRDF with different parameters for each of the three primary colors.



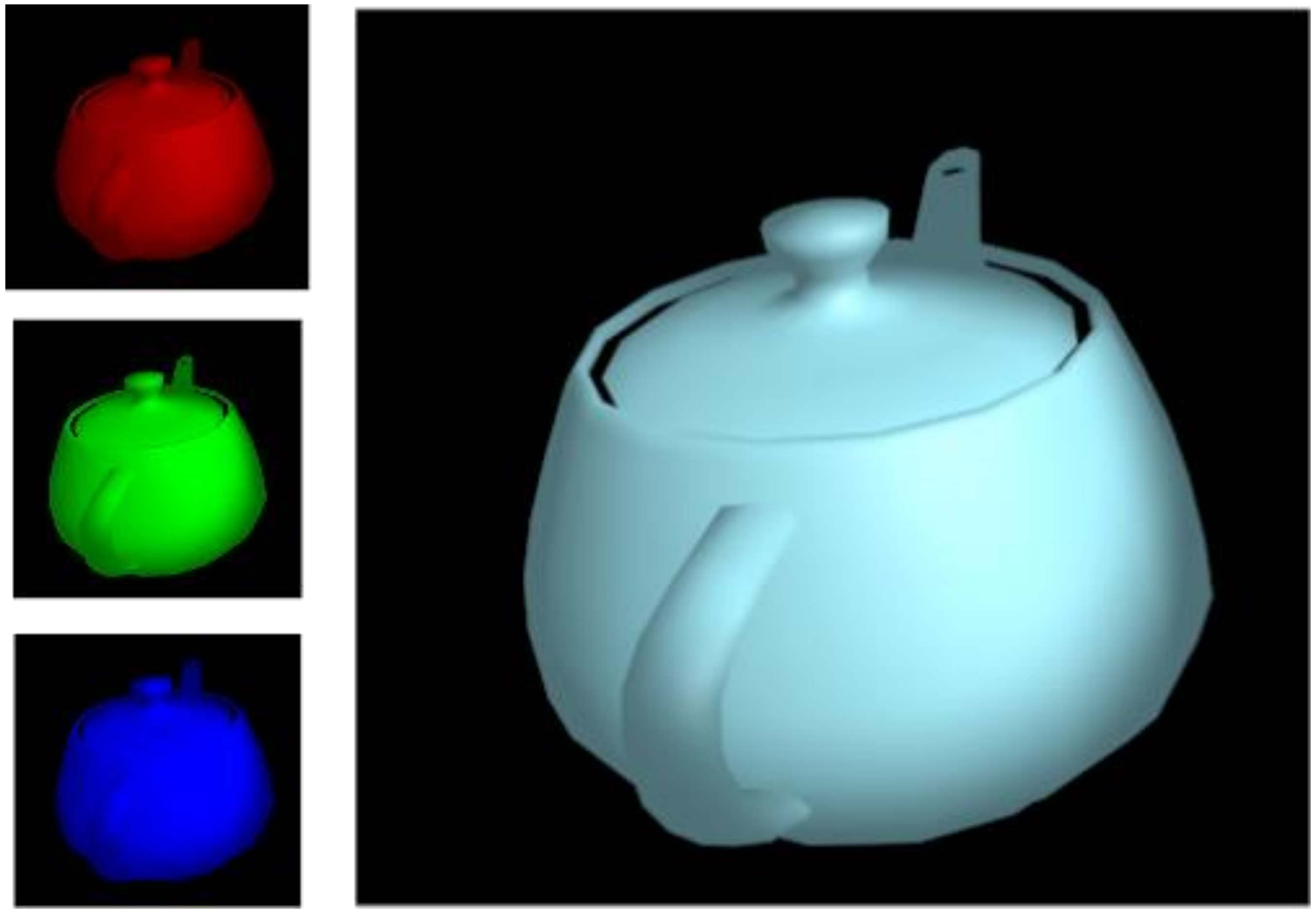
Considering colors

Three images are produced independently, each considering either the red, green or blue components alone.



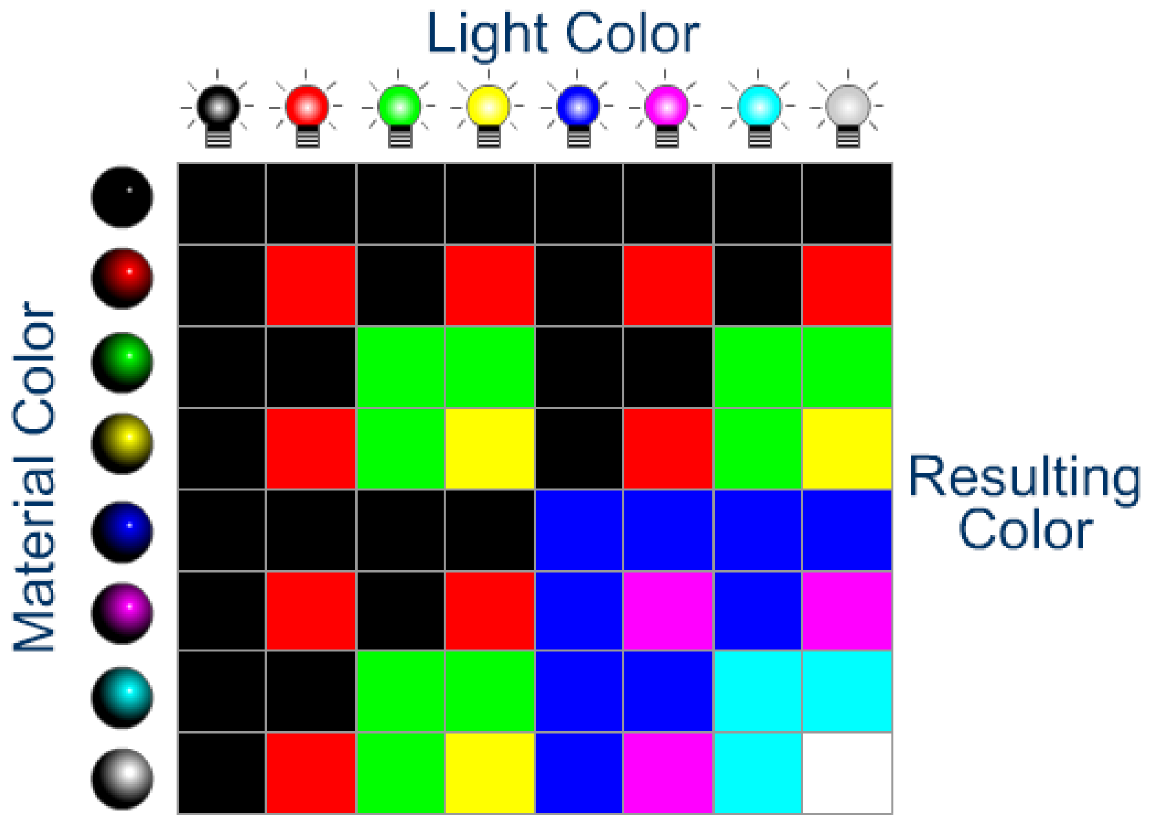
Considering colors

Finally, the three colors are assembled to create the output image.



Considering colors

Due to the separation of the color components, the combinations of the light and the material colors lead to results that are not straightforward.



Extensions to the rendering equation

The proposed rendering equation is capable of accurately computing several effects like reflections, shadows, matte and glossy objects.

However, it cannot simulate other effects such as *participating media* (i.e. rendering of gases and fumes) or even transparent objects like glass or water.

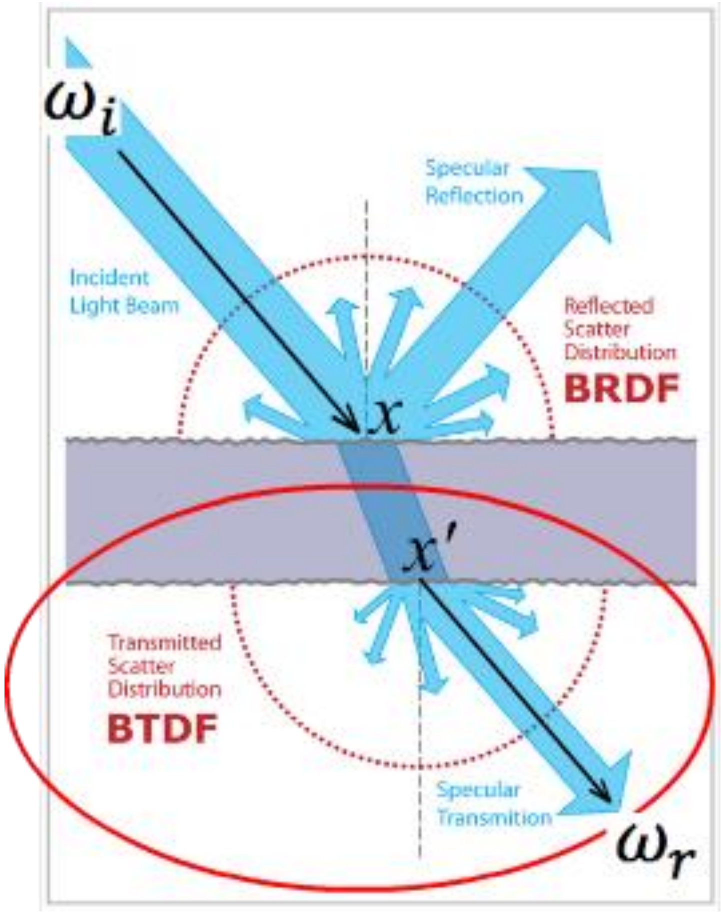
The BDRF and the rendering equations have been extended to account for these other important materials.

Extensions to the rendering equation

The first extension is to consider *transmitted lights* (i.e *transparency*): this is done by defining the *BTDF*: *Bidirectional Transmittance Distribution Function*.

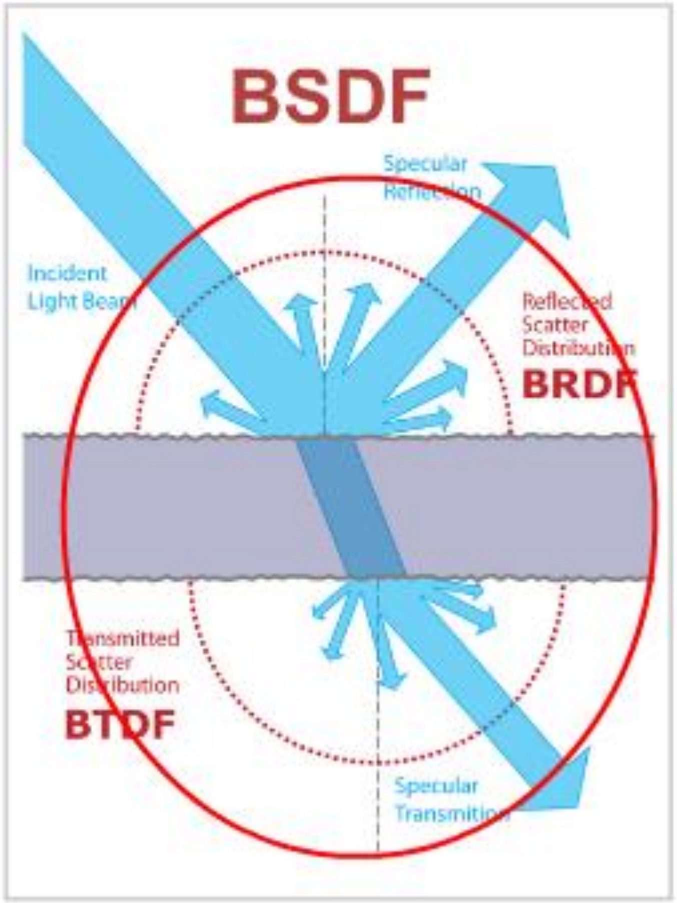
It has a similar definition to the *BRDF*, but it is used in the opposite direction.

$$f_t(\theta_i, \phi_i, \theta_r, \phi_r) = f_t(\omega_i, \omega_r)$$



Extensions to the rendering equation

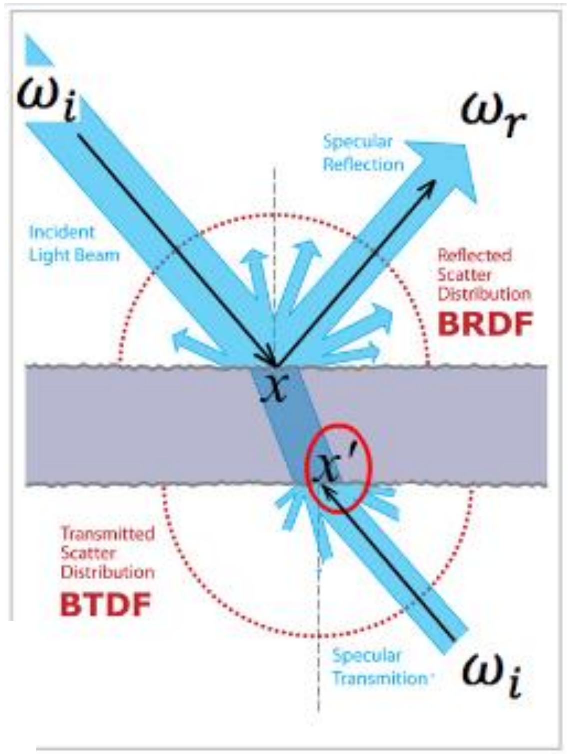
Since usually the angles for the *BRDF* and *BTDF* do not overlap, they are normally included in a single function called *BSDF*: *Bidirectional Scattering Distribution Function*.



Extensions to the rendering equation

The rendering equations should be updated to consider both the BRDF and the BTDF.

Here x' denotes the point on the other side of the surface (supposing it is unique) from which light is directed toward point x.



$$L(x,\omega_r) = L_e(x,\omega_r) + \int L(y,\overrightarrow{yx}) f_r(x,\overrightarrow{yx},\omega_r) G(x,y) V(x,y)\, dy + \int L\left(y,\overrightarrow{yx'}\right) f_t\left(x',\overrightarrow{yx'},\omega_r\right) G(x',y) V(x',y)\, dy$$

Extensions to the rendering equation

In several important materials, light can bounce inside the object and exit from another point. This phenomenon is called *sub-surface scattering*.

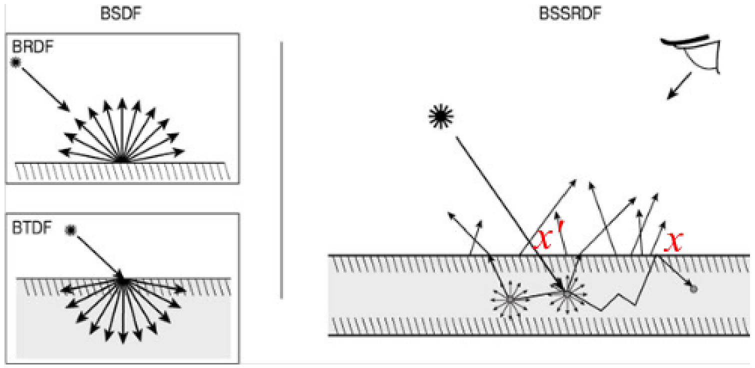
Examples of such materials are the human skin and the marble.

Extensions to the rendering equation

A more complex function, called *BSSRDF* (*Bidirectional surface reflectance distribution function*) must be used.

The function has an extra parameter x' that considers the point on the surface from which lights enters at angle ω_i.

The rendering equation now integrates over all the points of an object, to consider the quantity of lights that can enter from there, and exit from the given position and direction.



$$L(x,\omega_r) = L_e(x,\omega_r) + \iint L\left(y,\overrightarrow{yx'}\right) f_{ss}\left(x,\overrightarrow{yx'},x',\omega_r\right) G(x',y)V(x',y)\,dx'dy$$

Solution of the rendering equation

The rendering equations are very hard to solve, and generally require complex discretization techniques.

In the following, we will see very simple approximations to the rendering equation that are capable of providing good results with a reasonable complexity.

Some of them, are supported in Vulkan with some of its specific features.

Lights basics

The simplest approximations to the rendering equation, divide the light sources into *direct* and *indirect*.

Lights basics

Direct sources represent lights coming from specific positions and directions (e.g. a lamp, a studio spotlight, or the sun in an outdoor scene).

Lights basics

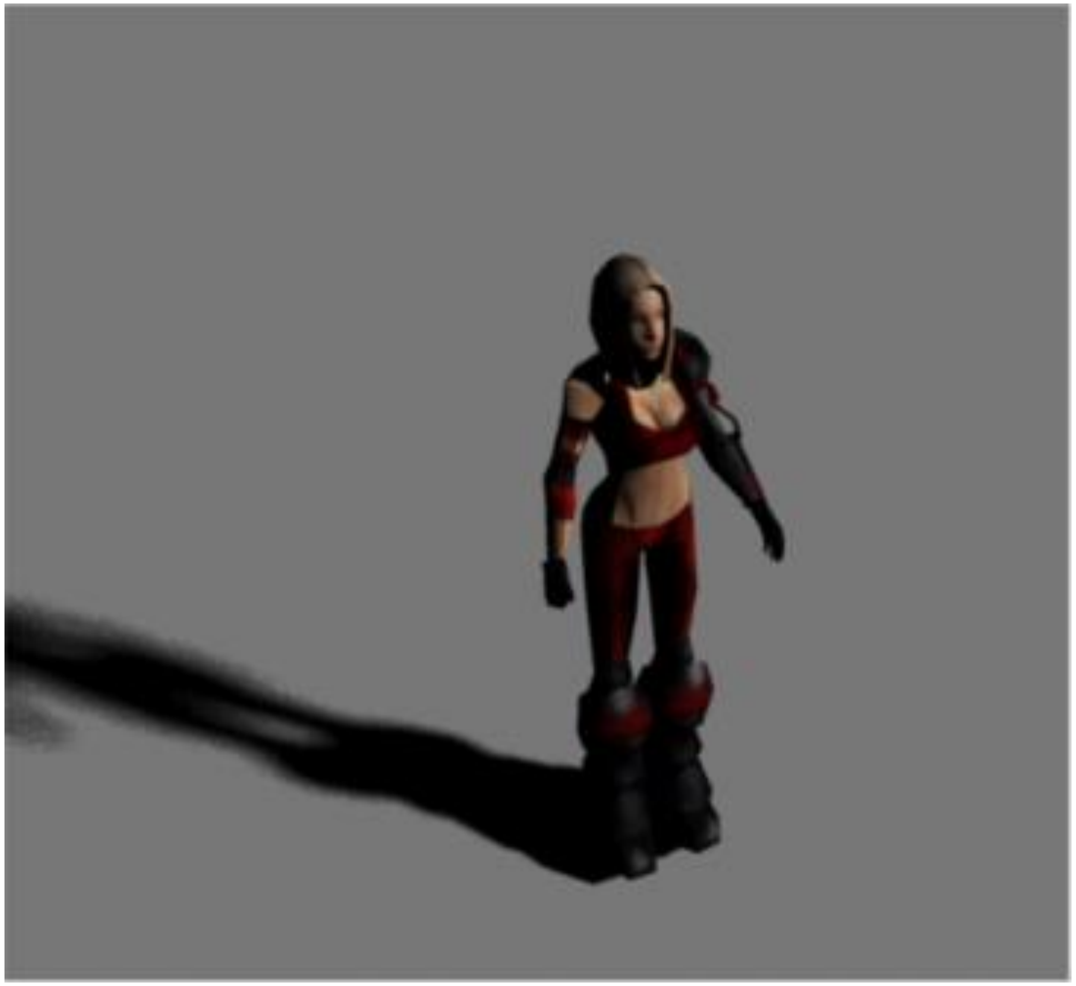
Indirect sources consider all the other types of illumination, mainly caused by light bounces and reflections among the surfaces.

Sometimes photographers, exploit these sources to add soft lights in their pictures.

Lights basics

With only *direct* sources, the images become very dark and do not seem very realistic: if a point is not hit by any light, it appears pitch black.

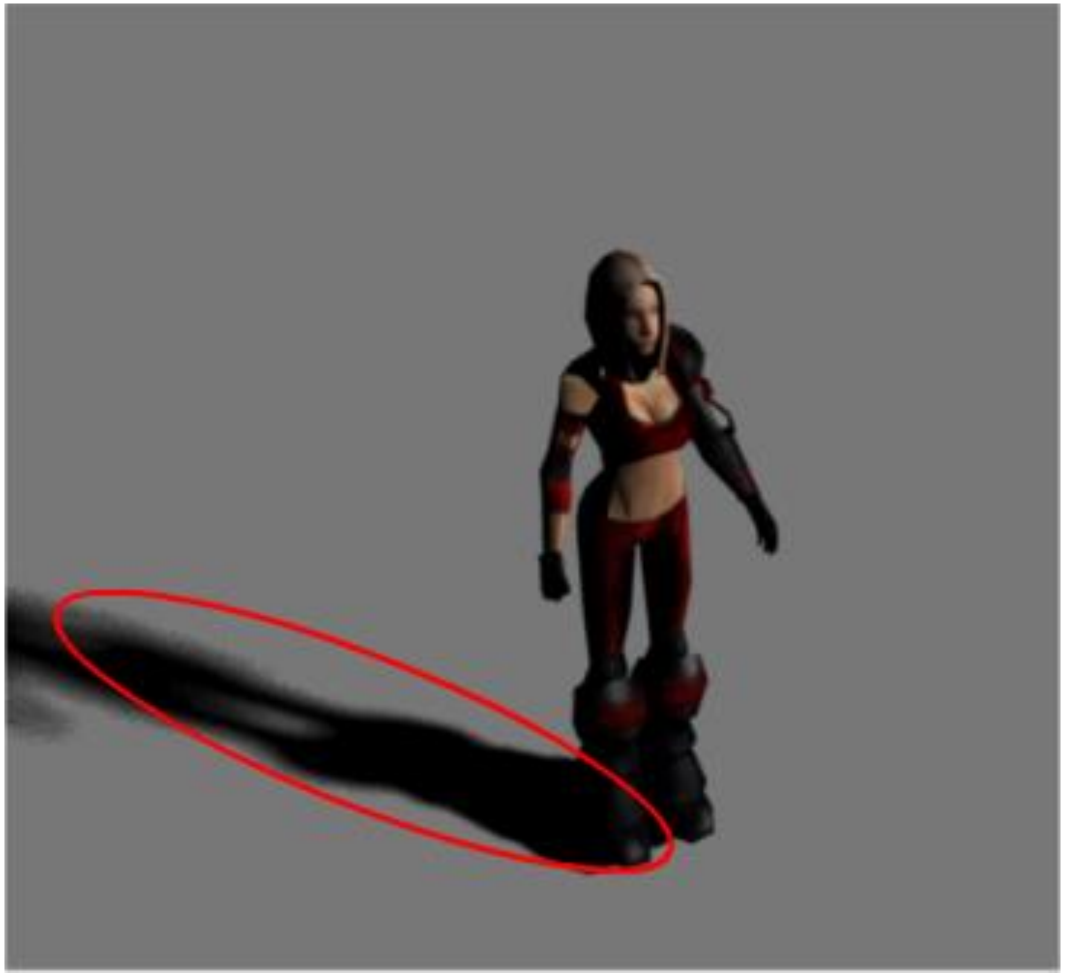
This setup is not realistic, and it must be avoided: unfortunately, it is also the simplest assumption that can be done to simplify the solution of the rendering equation in real-time CG!



Lights basics

Projected shadows are created by the occlusion of direct light sources.

As we will see, they can be approximated with specific techniques.



Lights basics

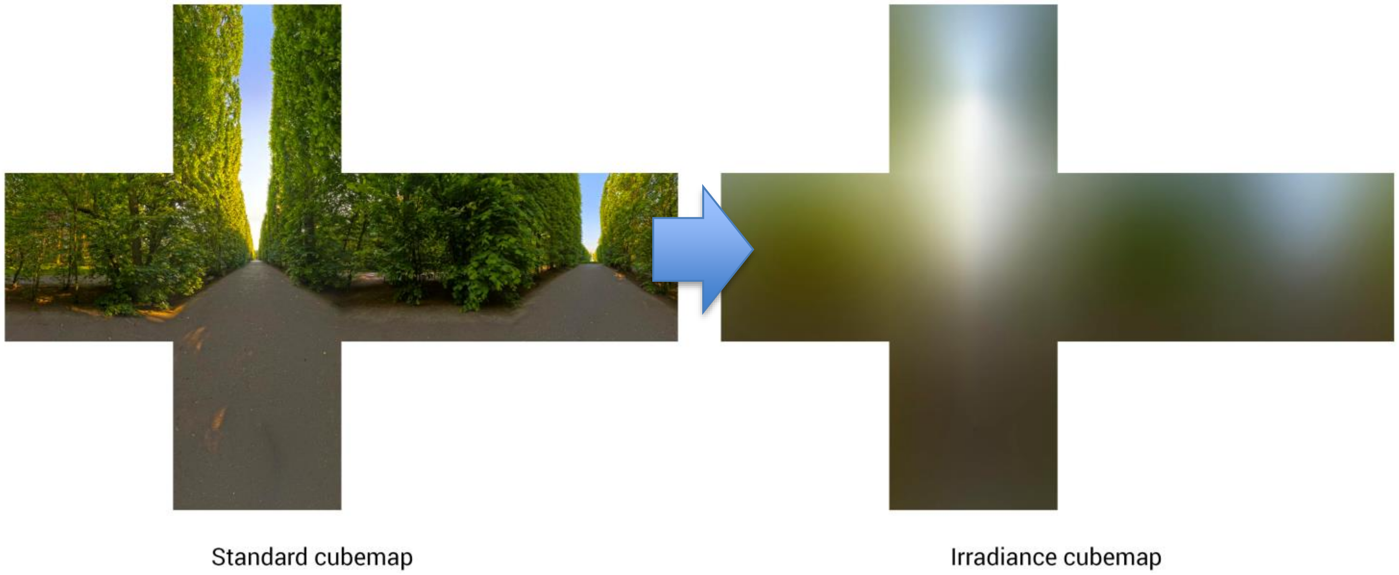
Indirect lighting adds realism, by considering also the light that bounces on other surfaces.

Accurate computation requires a lot of effort, and it is not easy to implement in real time.

However, several approximation to recreate indirect lighting effects exists.

Light basics

In most of the cases, the light contribution for a single point and direction is computed off-line and stored in some image-based structure, which is later used to perform these approximations.



We will briefly introduce some of these techniques later in the course.

Marco Gribaudo

Associate Professor

CONTACTS

Tel. +39 02 2399 3568
marco.gribaudo@polimi.it
https://www.deib.polimi.it/eng/home-page

(Remember to use the phone, since mails might require a lot of time to be answered. Microsoft Teams messages might also be faster than regular mails)