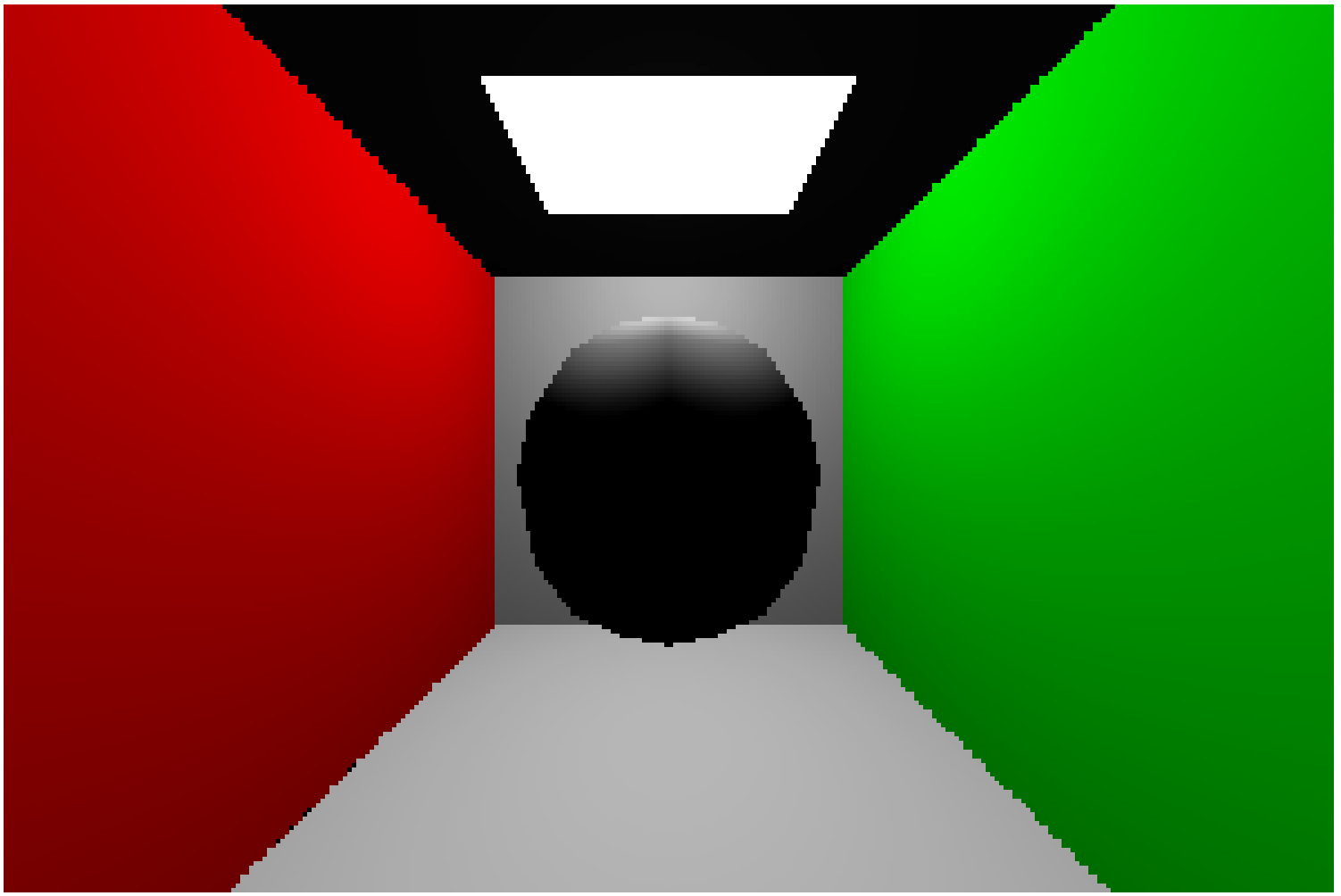
Computer Graphics

MTAT.03.015

Raimond Tunnel

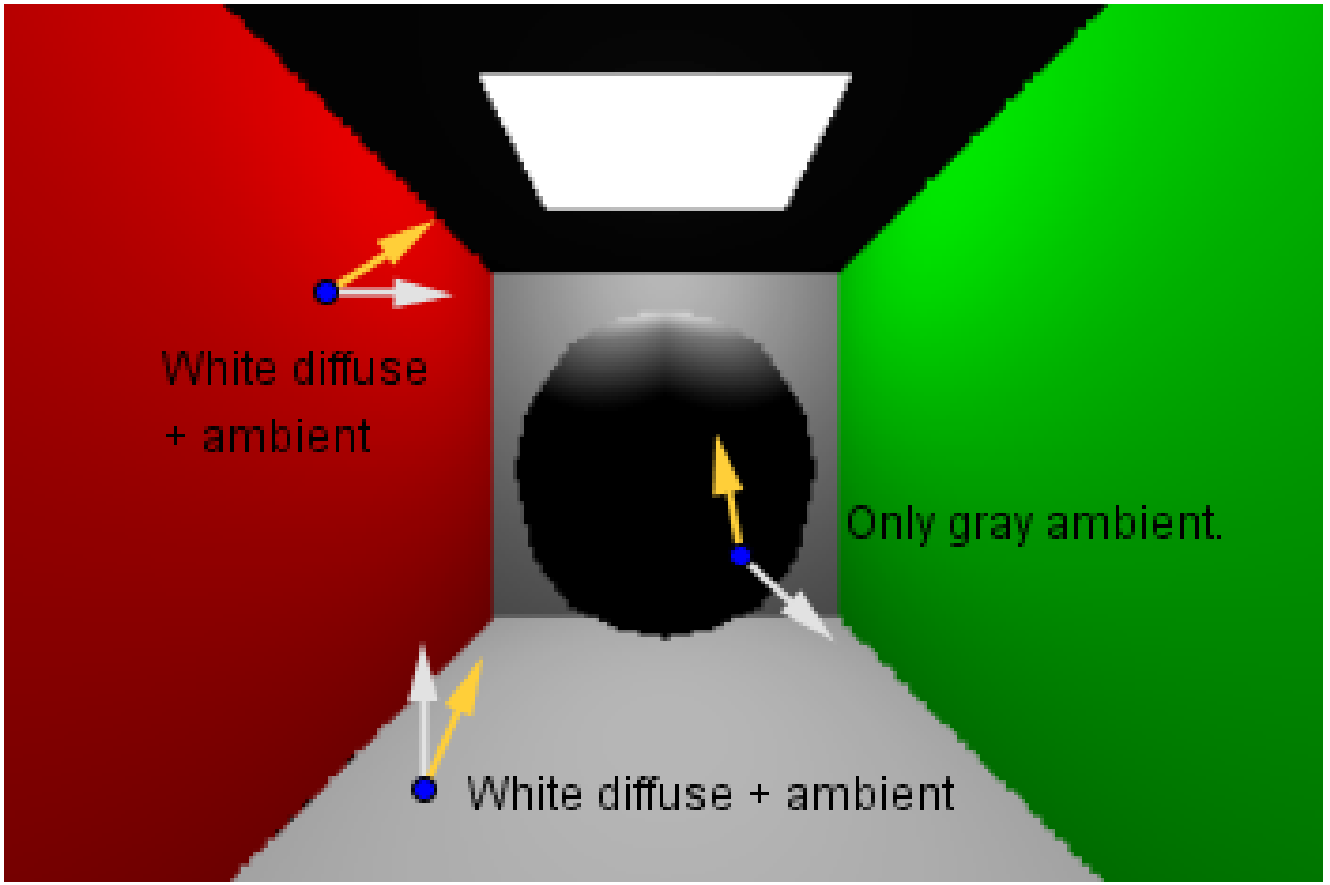
Study IT in .ee

The Road So Far...



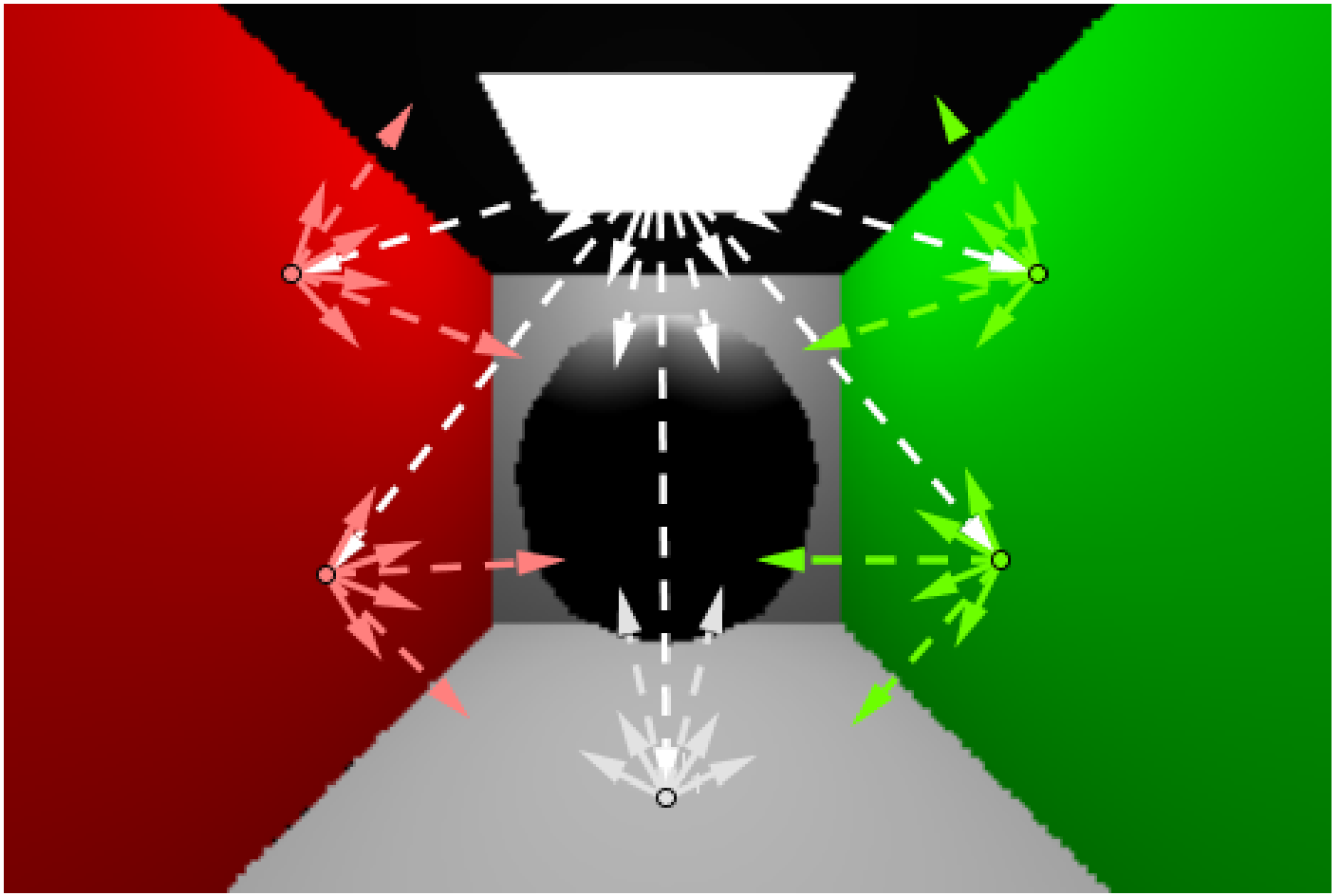
Direct / Local Illumination

- Apply a lighting model for a fragment. We consider only the light source.



Indirect Lighting

- Not only does the light source illuminate the scene. The scene itself illuminates the scene.



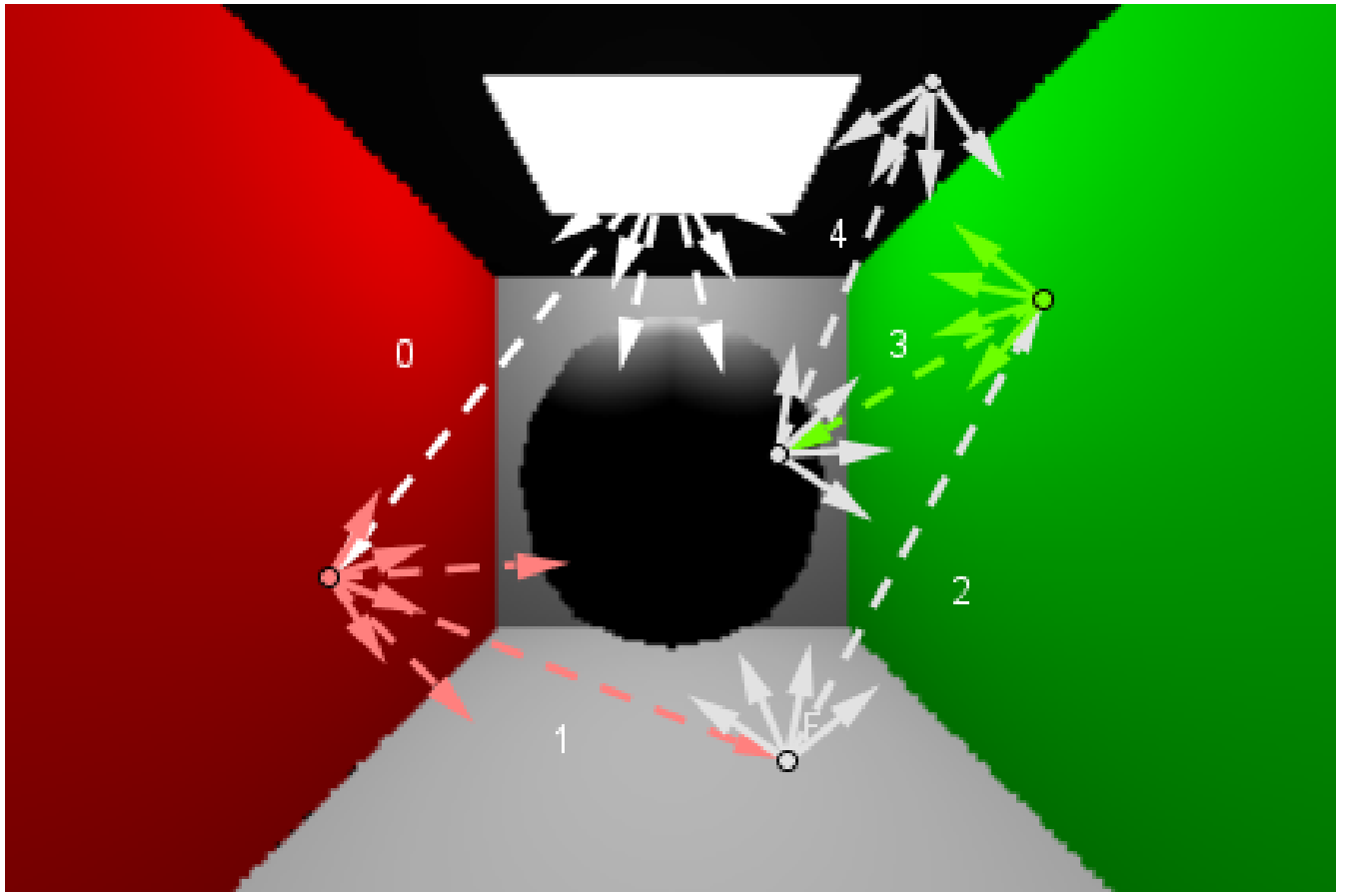
Cornell Box

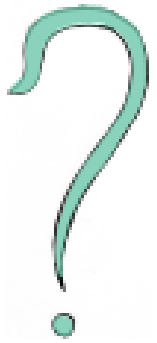
- This scene is called the Cornell Box scene.
- It is often used to test global illumination.
- Left wall red, right wall green.
- Other walls are white / gray.
- Light source in the ceiling.
- Objects inside the room.

By LockRickard:
http://lockrikard.deviantart.com/art/Minecraft-Cornell-Box-356384680

Indirect Lighting

- But the light bounces around more than once...

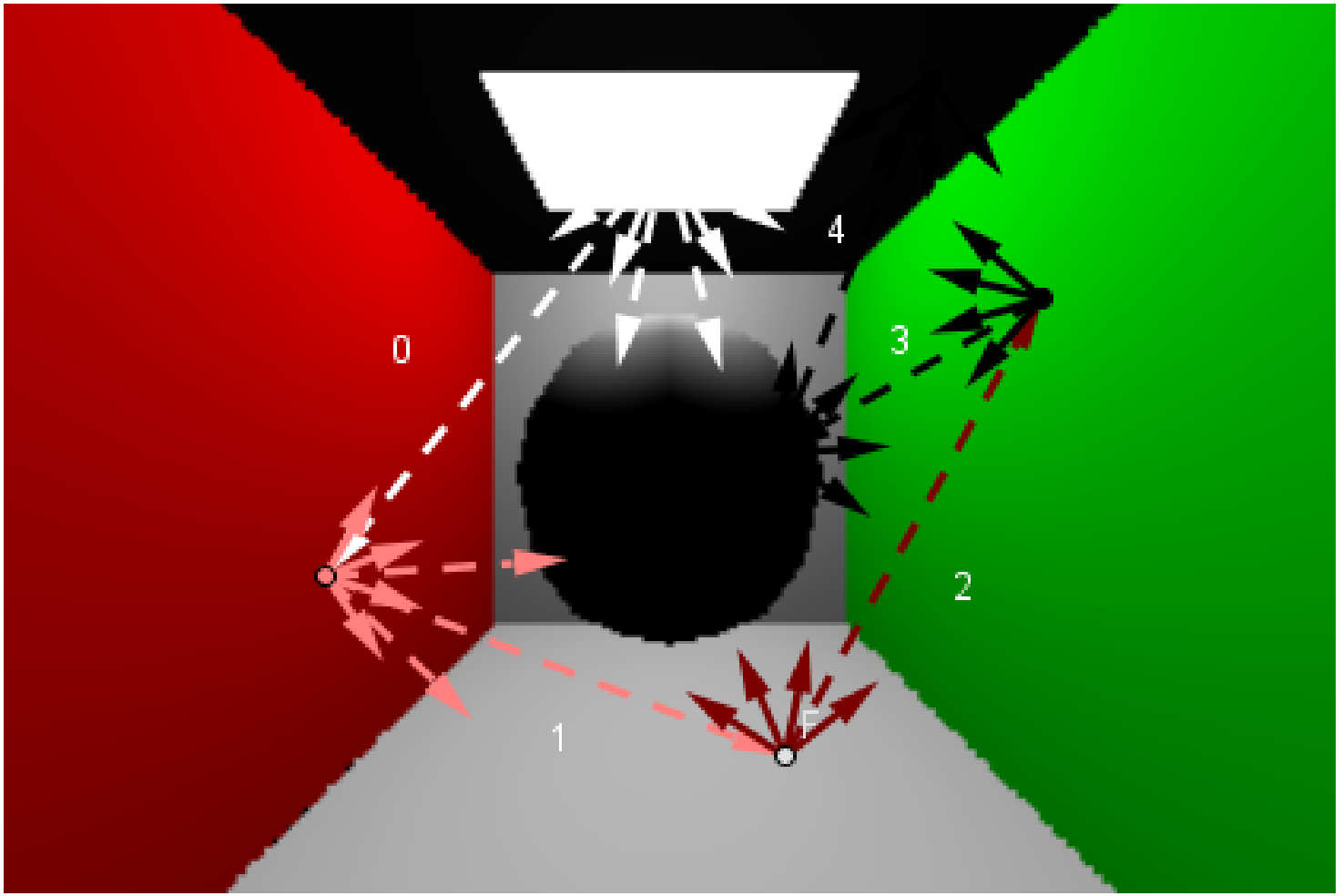




- What is wrong with this picture?

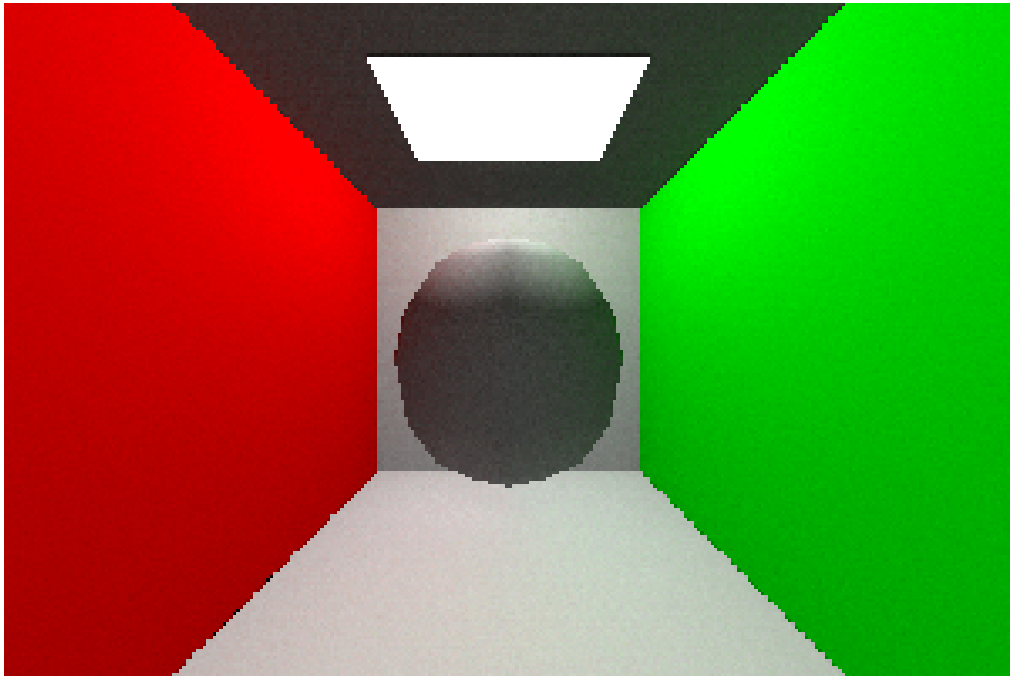
Indirect Lighting

- It is more like...



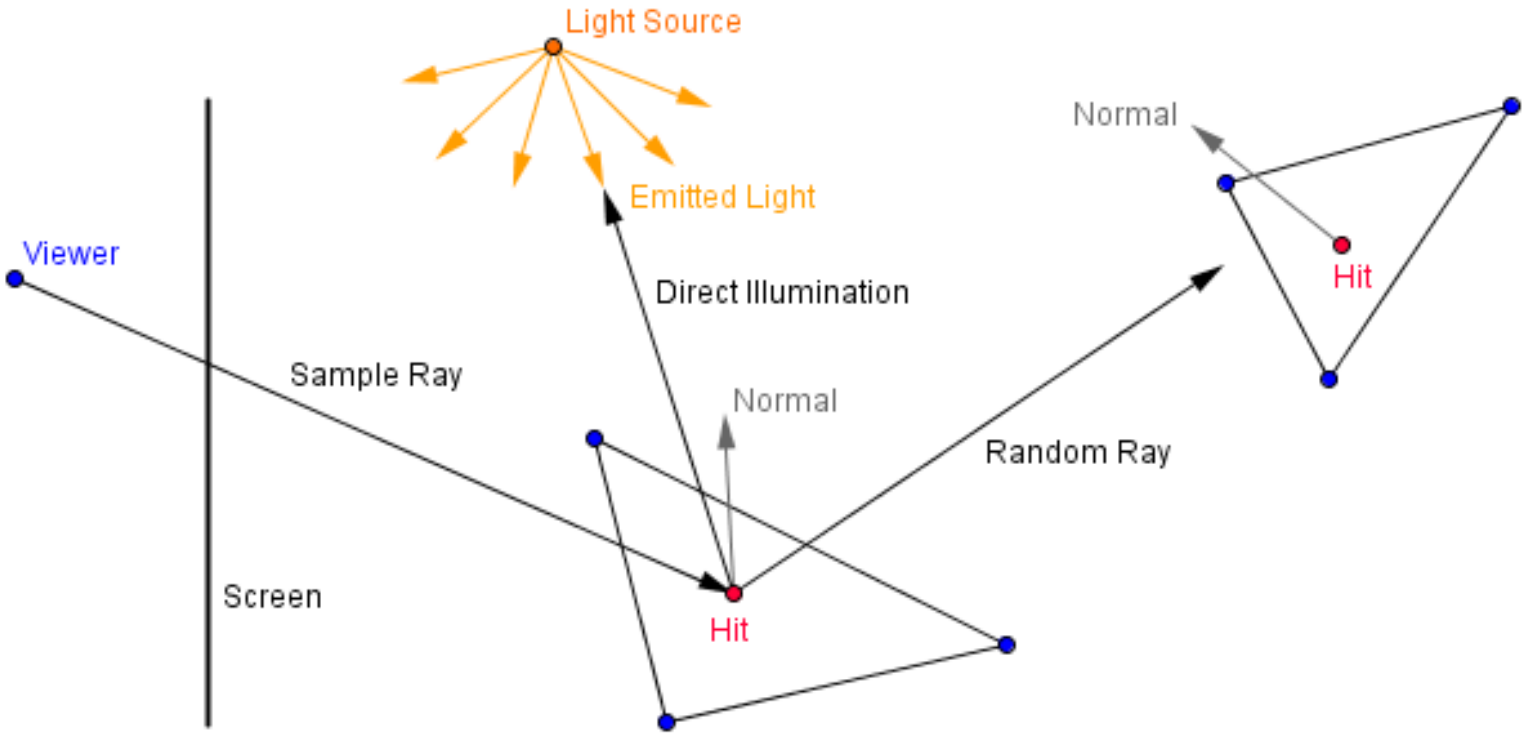
Global Illumination

- Take into account both direct (from the light source) and indirect lighting.
- Alternatively, consider all surfaces as light sources.



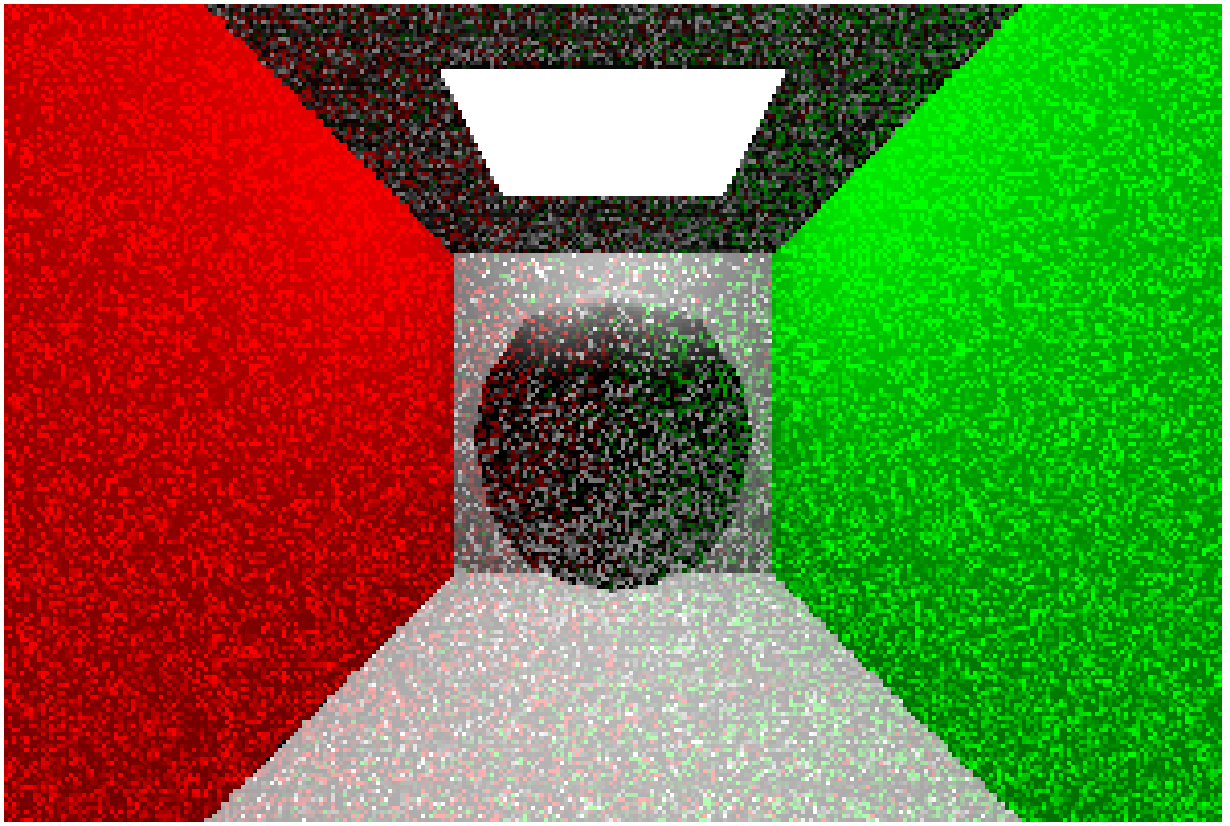
Path Tracing (with Direct Illum.)

- Based on ray trace rendering.
- For each hit, send random ray(s) to sample indirect illumination from the scene.



Path Tracing (with Direct Illum.)

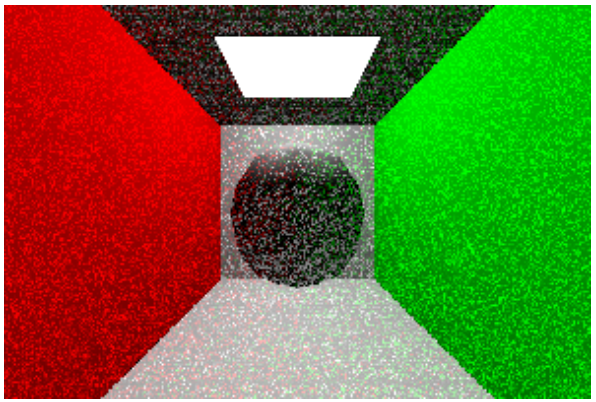
- Did just that, got this result:



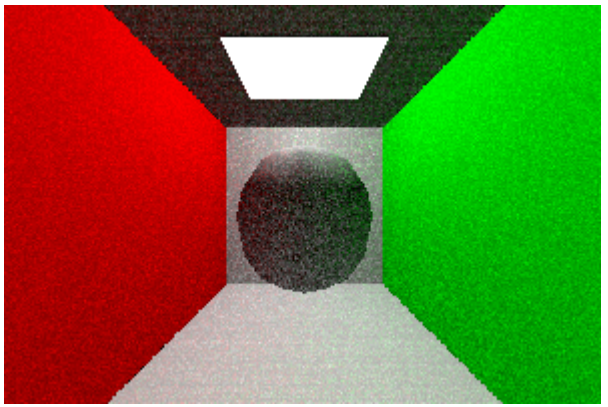
- That is correct, it just one sample.

Path Tracing (with Direct Illum.)

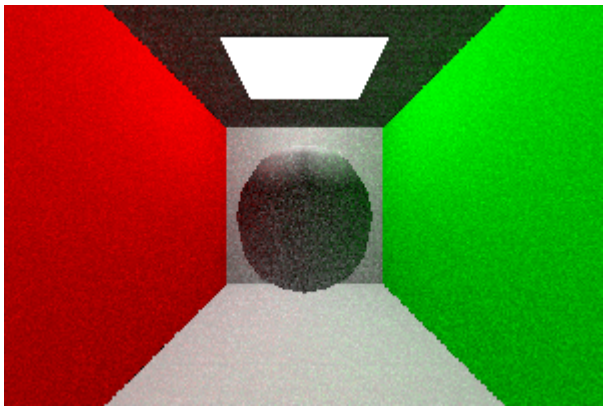
- Because the rays are random, we need to sample a lot of them, and average all results.



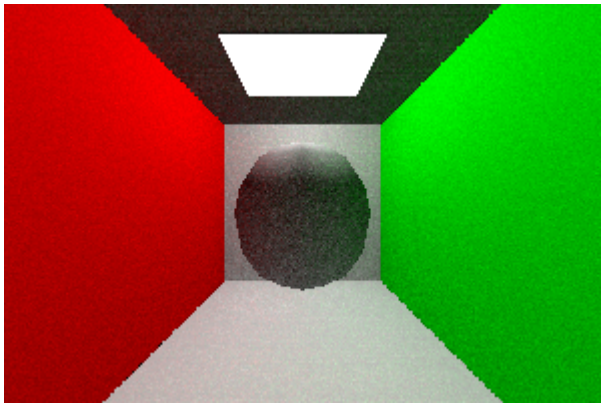
1 sample



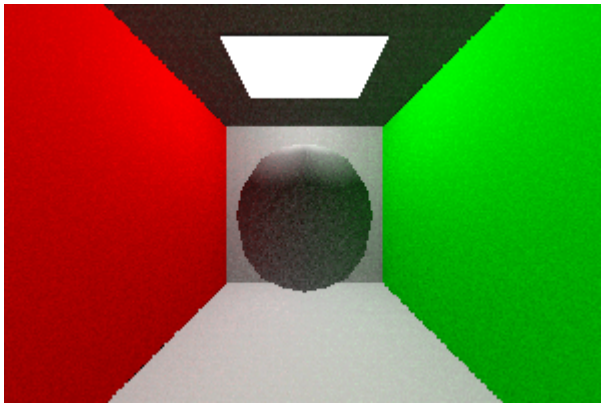
5 samples



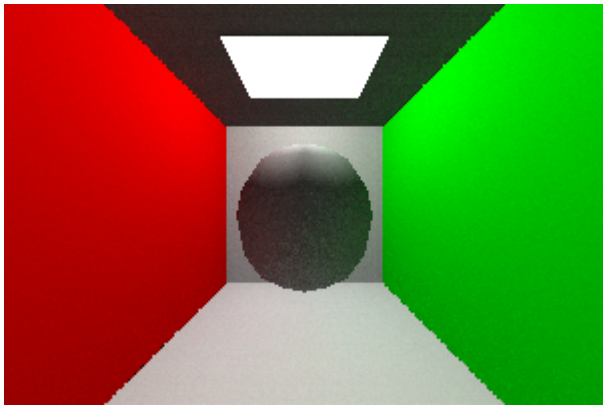
10 samples



20 samples



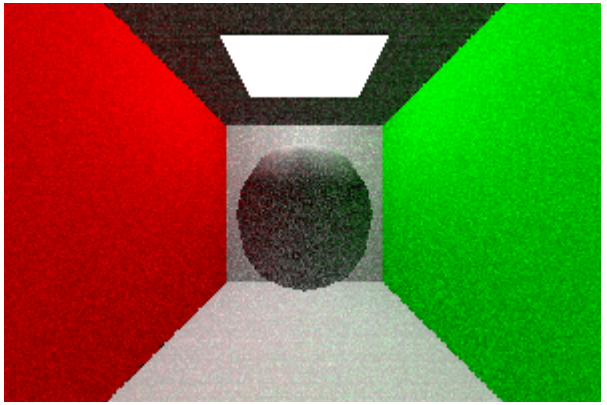
40 samples



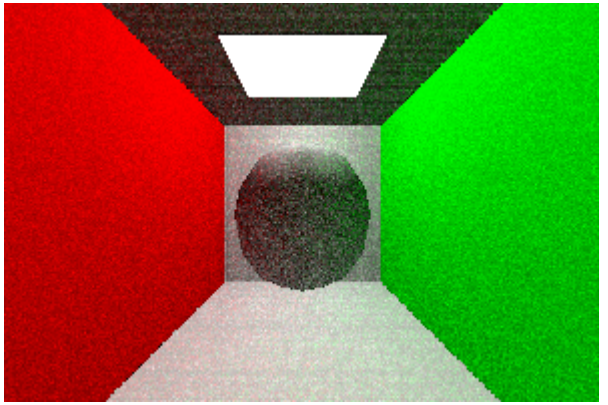
80 samples

Path Tracing (with Direct Illum.)

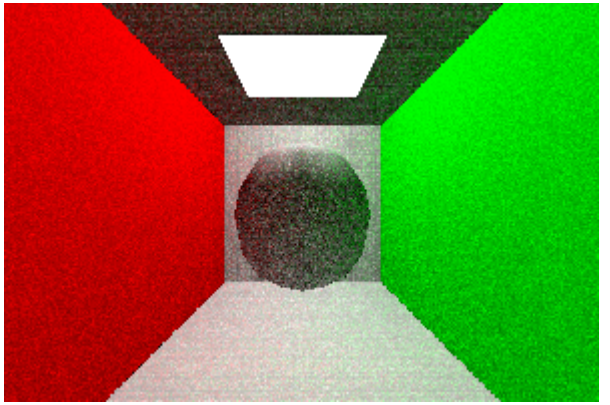
- That was just one bounce. We can do more.



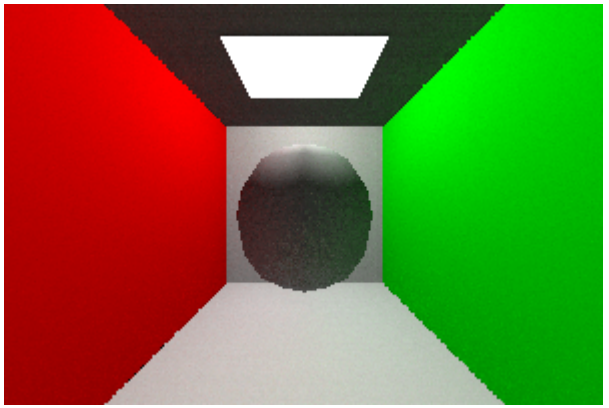
5 samples, 1 bounce



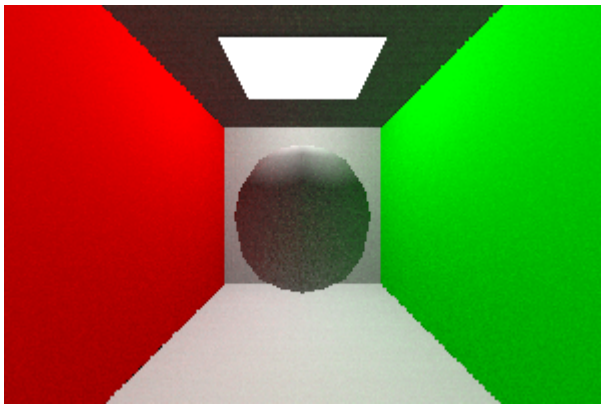
5 samples, 2 bounces



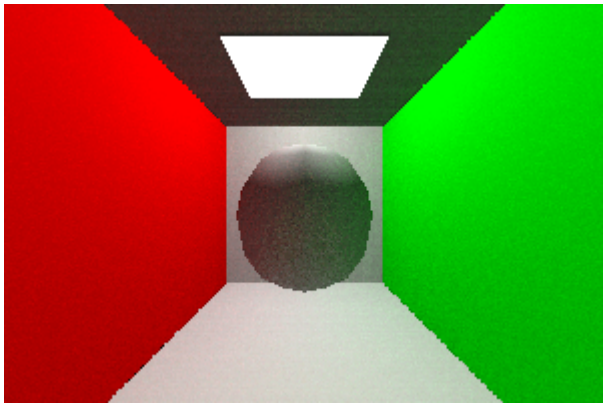
5 samples, 3 bounces



80 samples, 1 bounce



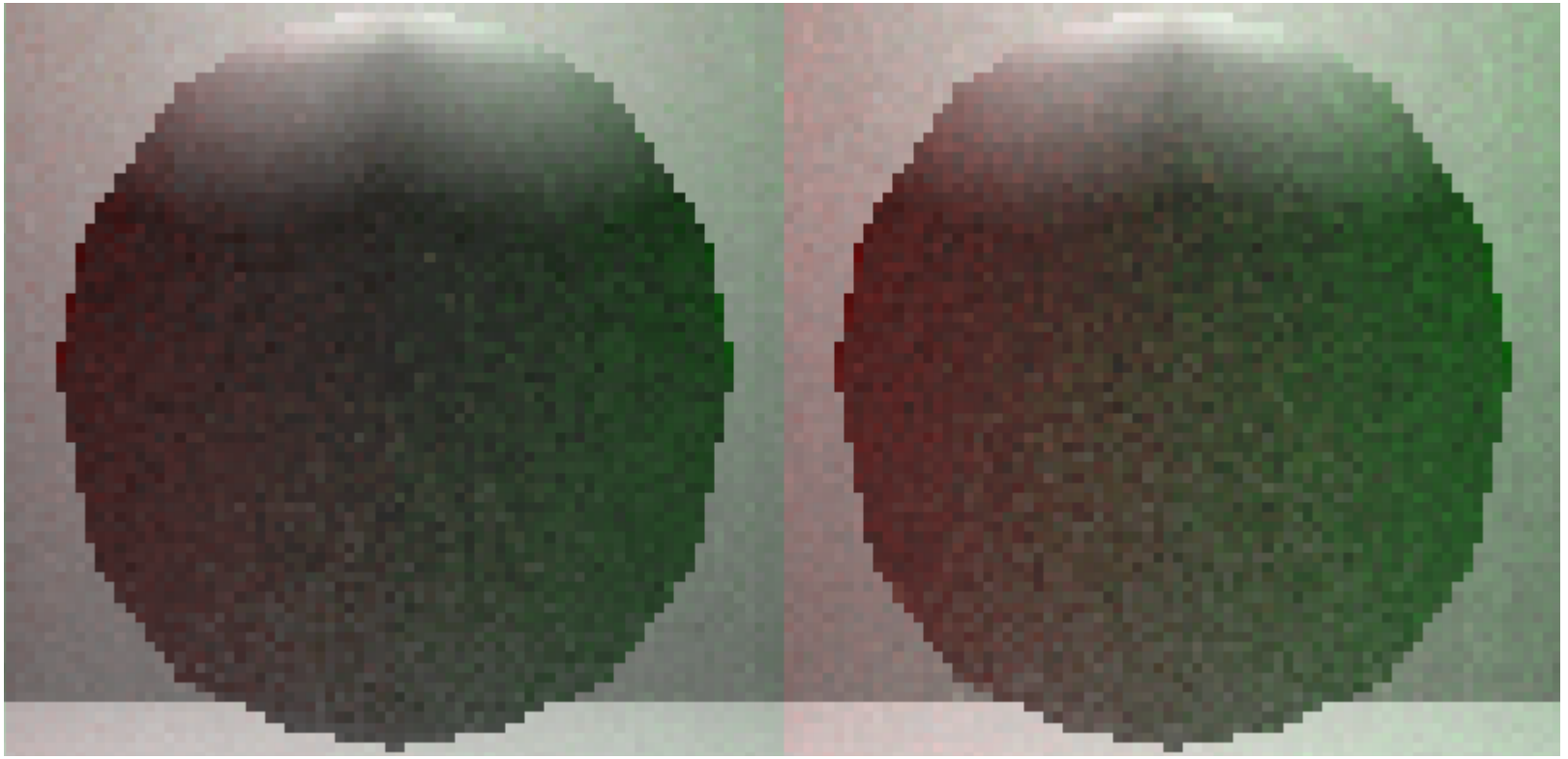
80 samples, 2 bounces



80 samples, 3 bounces

Path Tracing (with Direct Illum.)

- There are differences.



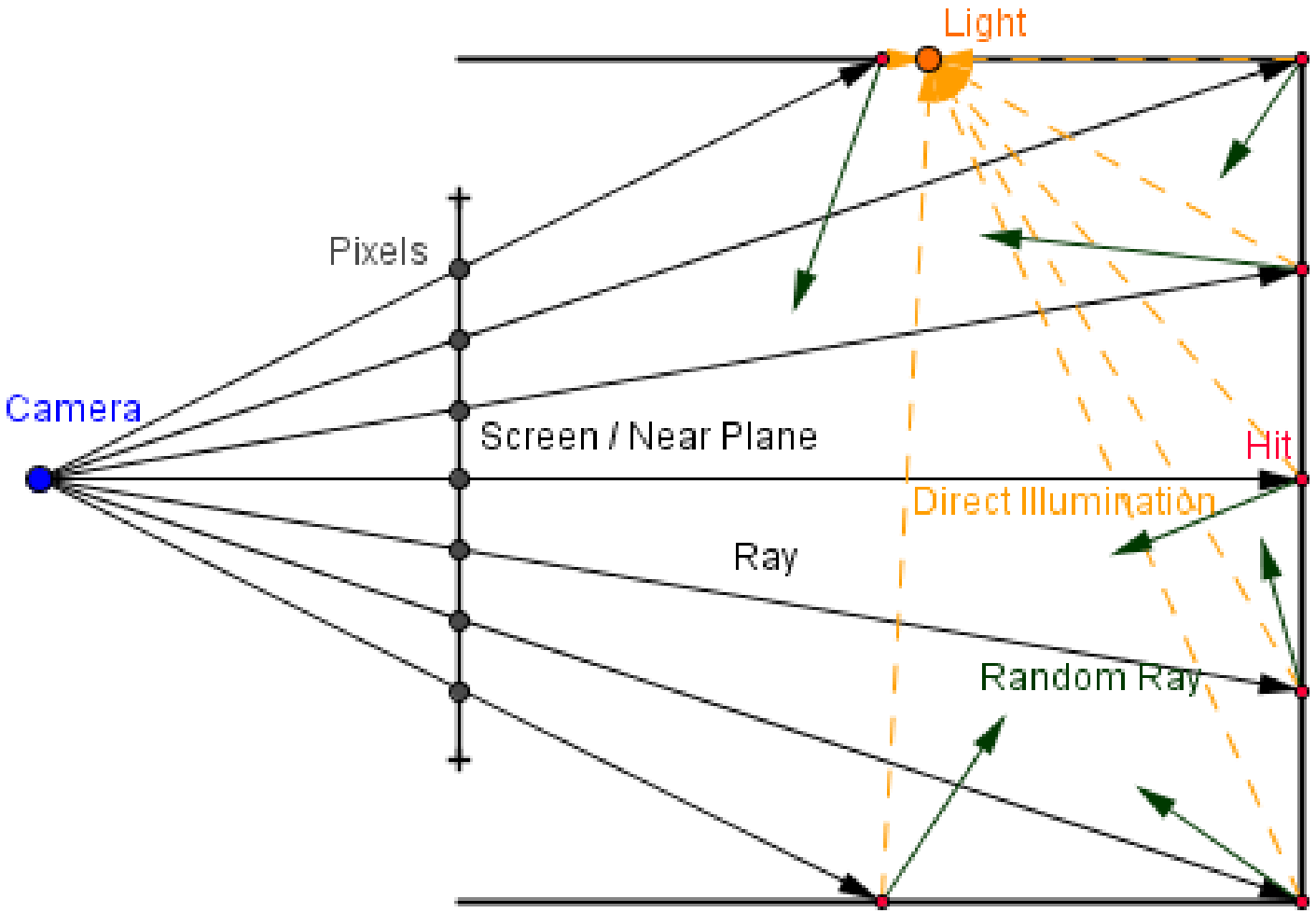
80 samples, 1 bounce versus 3 bounces

Path Tracing

- There are other path tracing techniques also.
- Without direct illumination:
  - Only bounce, until we reach the light source.
  - Light source intensity should be more than 1.
  - Direct into the light source at a random bounce.
- Create a number of random rays at the first bounce.

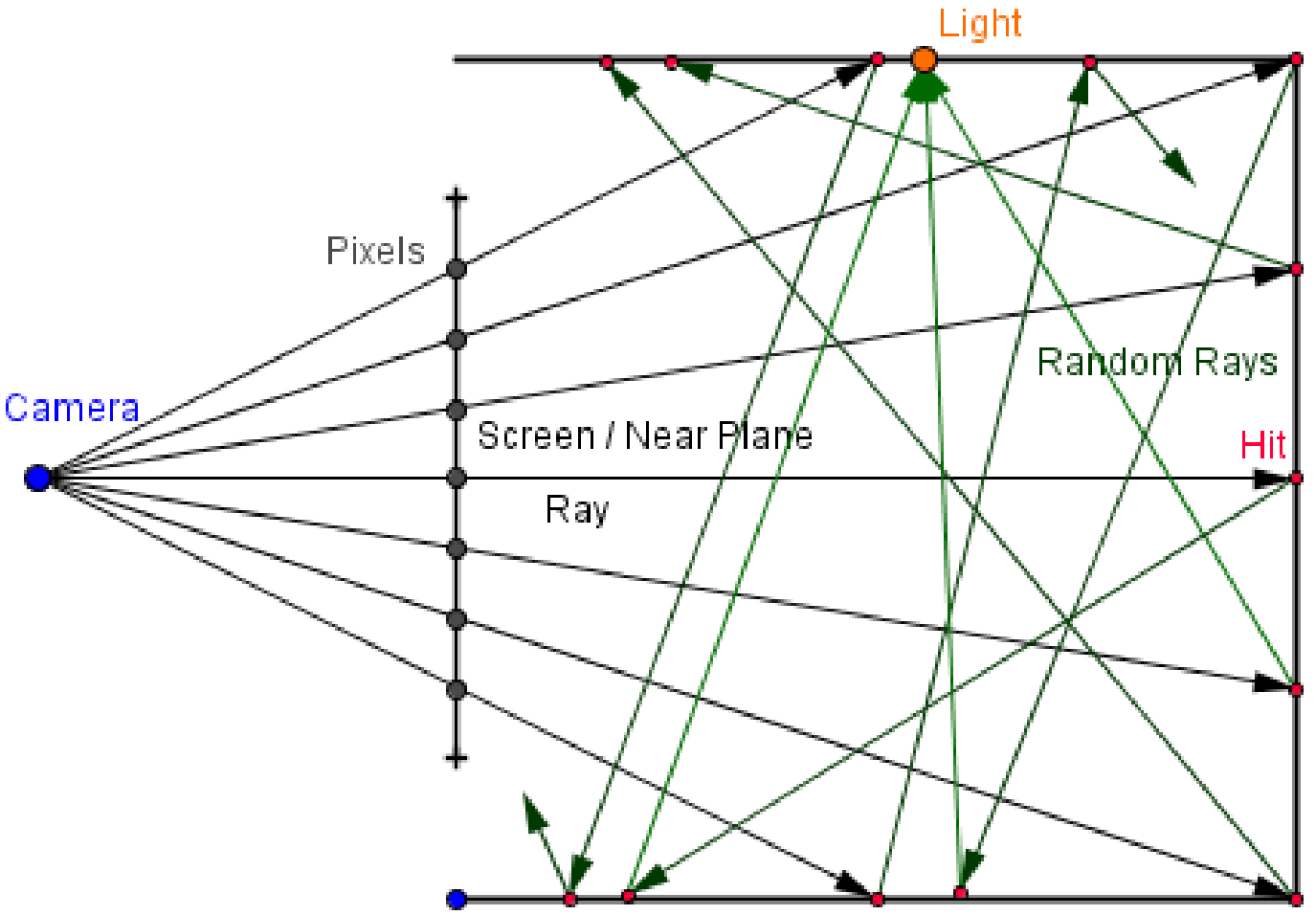
Path Tracing

- With direct illumination



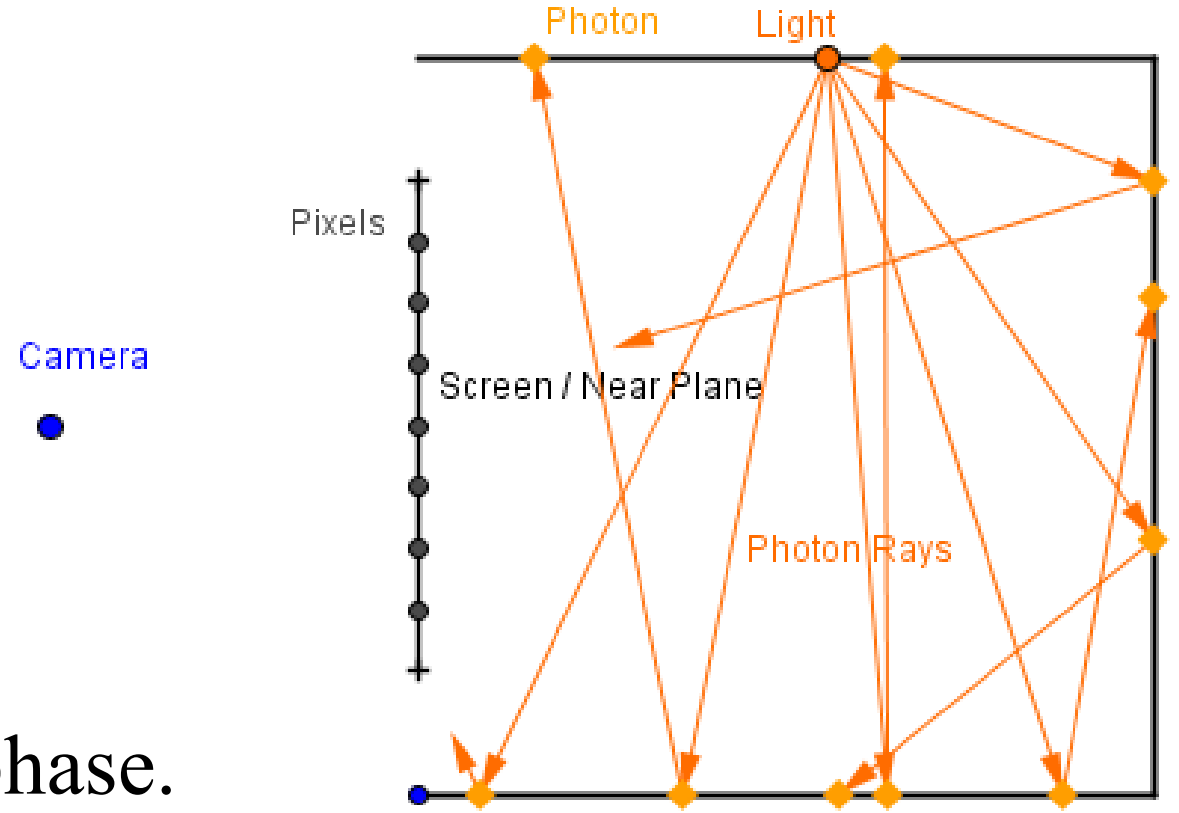
Path Tracing

- Without direct illumination (direct rays to light, after a random number of bounces).



Photon Mapping

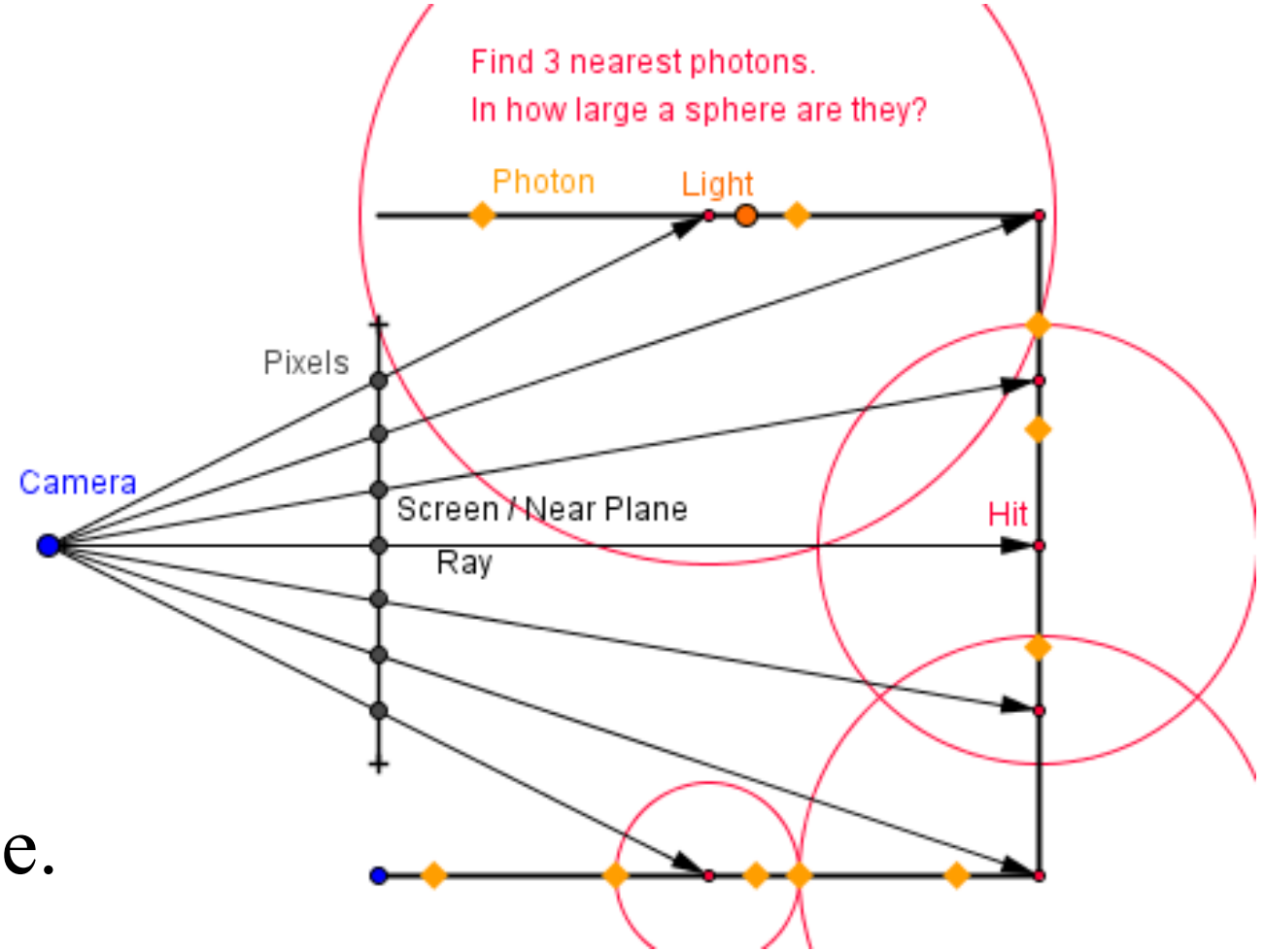
- First shoot rays from the light source.
- Create a map of photons, that those rays distribute.



Construction phase.

Photon Mapping

- Then shoot rays from the camera, find nearest photons for the hit points.



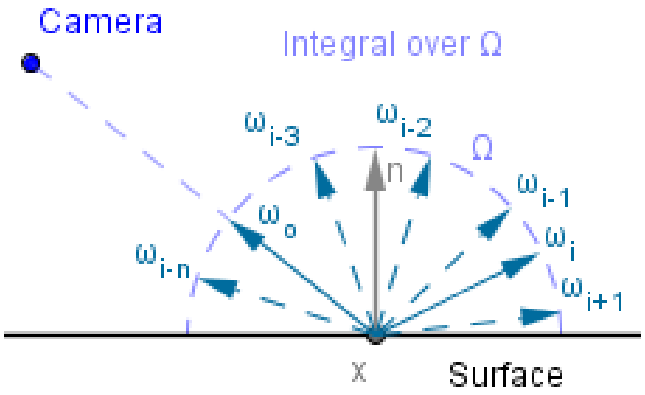
Gathering phase.

Photon Mapping

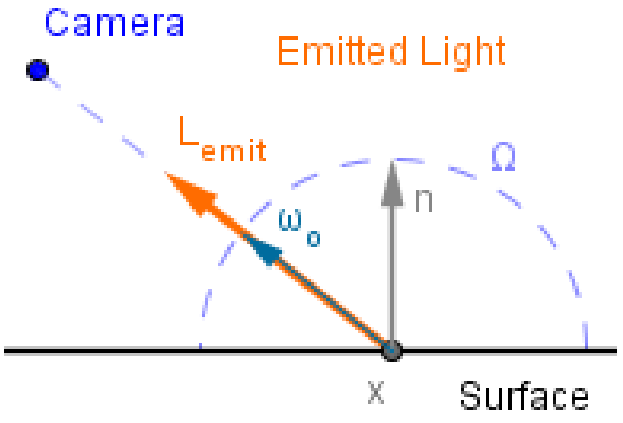
- Illumination of the surface can be estimated, by accounting the nearest photons, divided by the area of the sphere.
- Some illumination (direct, specular) can be calculated without the photon map.
- Gathering phase is much slower, then the construction.
- How to store the photon map for fast nearest neighbour search?

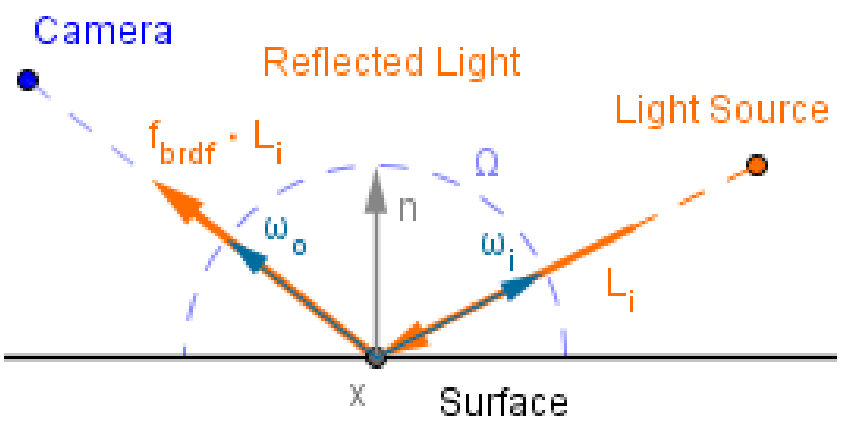
The Rendering Equation

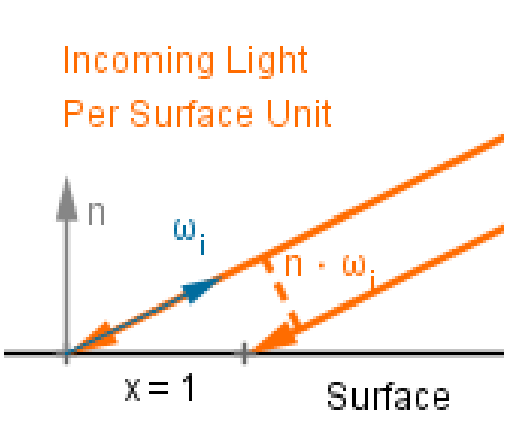
- Mathematical formulation for the visible color (outgoing light) of the surface.



$$L_{out}(x, \omega_o) = L_{emit}(x, \omega_o) + \int_{\Omega} f_{brdf}(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i) \cdot (\omega_i \cdot n) \, \mathrm{d}\omega_i$$







The Rendering Equation

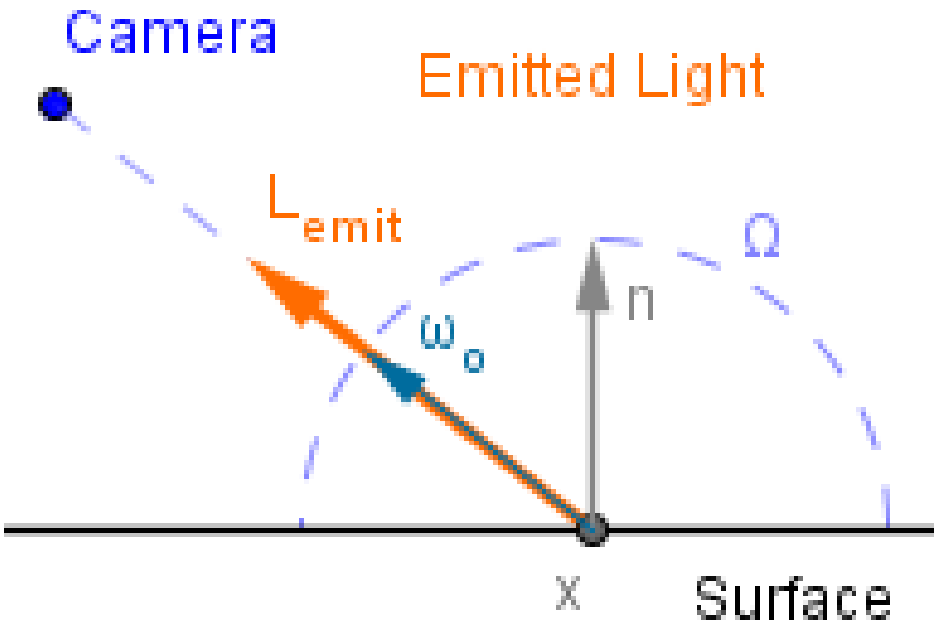
$$L_{out}(x, \omega_o) = L_{emit}(x, \omega_o) + \int_{\Omega} f_{brdf}(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i) \cdot (\omega_i \cdot n) \, \mathrm{d}\omega_i$$

- x – Surface point
- ω_o – Viewing (outgoing light) angle
- ω_i – Incoming light angle
- L_{out} – Amount of outgoing light
- L_{emit} – Amount of emitted light
- f_{brdf} – Amount of reflected light (BRDF)
- $L_i \cdot (\omega_i \cdot n)$ – Amount of incoming light per surface

The Rendering Equation

- Some surfaces can emit light on their own (light sources *eg* car headlights, fluorescent materials)

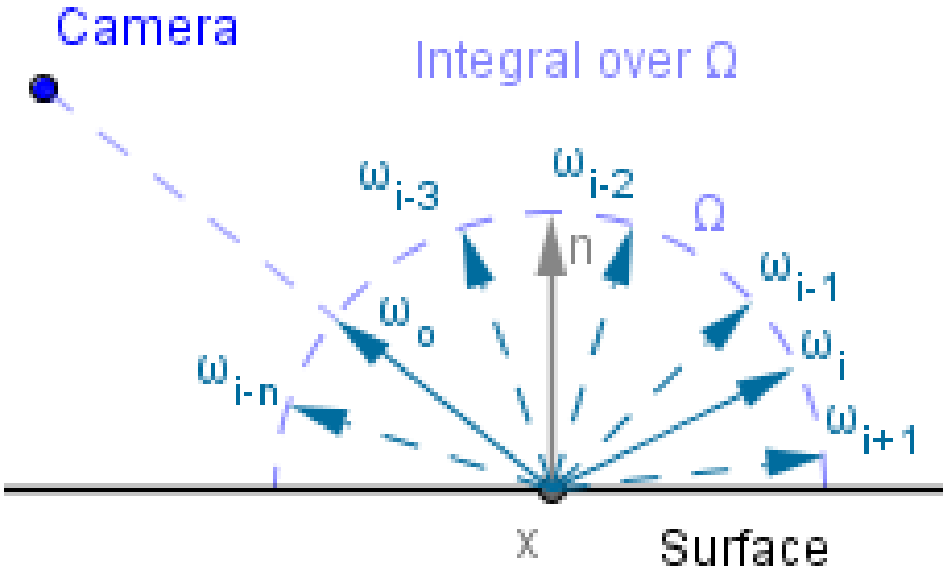
$$L_{out}(x, \omega_o) = \boldsymbol{L_{emit}(x, \omega_o)} + \int_{\Omega} f_{brdf}(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i) \cdot (\omega_i \cdot n) \, \mathrm{d}\omega_i$$



The Rendering Equation

- Outgoing light is dependant of the light coming from all directions. So we integrate over the hemisphere.

$$L_{out}(x, \omega_o) = = L_{emit}(x, \omega_o) + \int_{\Omega} f_{brdf}(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i) \cdot (\omega_i \cdot n) \, d\omega_i$$



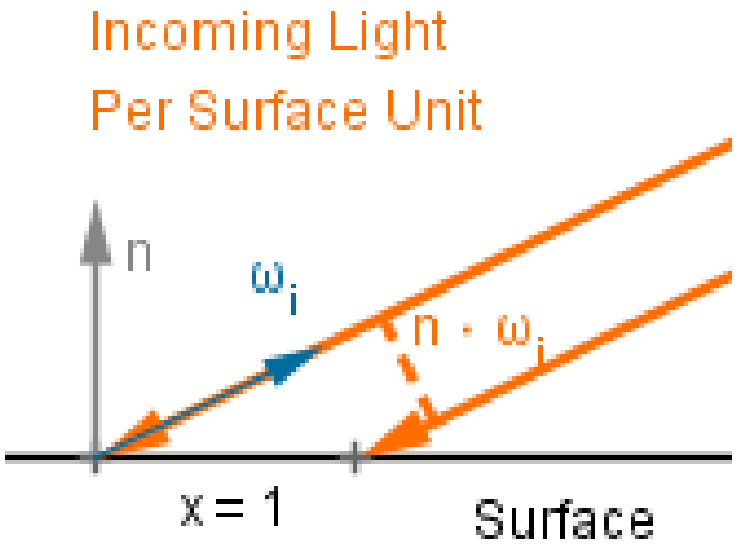
The Rendering Equation

- Light reaching the surface unit is proportional to

$$\cos\left(\angle\left(\omega_i, n\right)\right) = \left(\omega_i \cdot n\right)$$

$$L_{out}(x, \omega_o) =$$

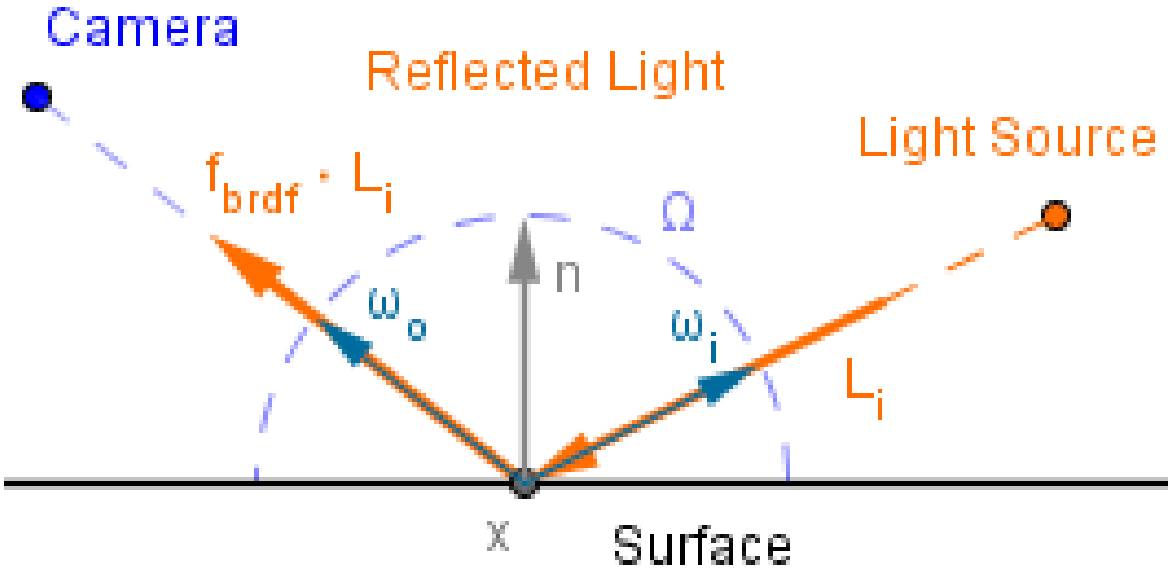
$$= L_{emit}(x, \omega_o) + \int_{\Omega} f_{brdf}(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i) \cdot \left(\boldsymbol{\omega_i} \cdot \boldsymbol{n}\right) \mathrm{d}\,\omega_i$$



The Rendering Equation

- Reflected light depends on the incoming light L_i and the material properties represented by f_{brdf}
- BRDF – Bidirectional Reflectance Distribution Function

$$L_{out}(x, \omega_o) = = L_{emit}(x, \omega_o) + \int_{\Omega} \boldsymbol{f}_{\boldsymbol{brdf}}(\boldsymbol{x}, \boldsymbol{\omega_i}, \boldsymbol{\omega_o}) \cdot \boldsymbol{L_i}(\boldsymbol{x}, \boldsymbol{\omega_i}) \cdot (\omega_i \cdot n) \, \mathrm{d}\omega_i$$



The Rendering Equation

- BRDF for Phong would be:

$$f_{brdf} = M_{diffuse} + M_{specular} \cdot (v \cdot r)^{shininess}$$

- Notice, that we are also multiplying the specular term with $\omega_i \cdot n$.
- This means that the intensity of the specular highlight also depends on the angle light is reaching the surface.

Radiosity

- Divides our geometry into (small) patches.
- Tries to approximate the rendering equation by replacing the integral with a finite sum.
- Assumes that each patch radiates light equally in all directions (diffuse reflection).

$$L_{emit}(x,\omega_o)+\int_{\Omega} f_{brdf}(x,\omega_i,\omega_o)\cdot L_i(x,\omega_i)\cdot(\omega_i\cdot n)\,\mathrm{d}\,\omega_i$$

$$L_{emit}(x,\omega_o)+\sum_j f_{brdf}(x,\omega_j,\omega_o)\cdot L_i(x_j)\cdot(\omega_j\cdot n)\,\Delta\omega_j$$

- For each patch *k*: $E_k+\rho_k\sum_j F_{k,j}\cdot L_j$

Radiosity

- For each patch k:

$$L_k = E_k + \rho_k \sum_j F_{k,j} \cdot L_j$$

How much patch j emits light

How much this patch emits light

How much this patch reflects light (its color)

View factor
How much this patch k receives light from patch j

Radiosity

- Vector equation for the entire scene

$$L = E + \rho F \cdot L$$
$$-E = \rho F \cdot L - L$$
$$-E = (\rho F - I) L$$
$$E = (I - \rho F) L$$
$$L = (I - \rho F)^{-1} E$$

$$F = \begin{Bmatrix} 0 & F_{0,1} & F_{0,2} & \cdots & F_{0,k} \\ F_{1,0} & 0 & F_{1,2} & \cdots & F_{1,k} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ F_{k,0} & F_{k,1} & F_{k,2} & \cdots & 0 \end{Bmatrix}$$

View factors in each row / column need to sum up to 1. We can not have more than 100% energy entering or exiting a patch!

- Unfortunately, finding the solution this way is $O(k^3)$

Radiosity

- Jacobi iteration method solves $Ax = b$, when A is strictly diagonally dominant.

$$(I - \rho F) L = E$$

Diagonal element has greater absolute value, then the sum of absolute values of other elements in a row (or column).

$$|a_{i,i}| > \sum_{i \neq j} |a_{i,j}|$$

- Is $(I - \rho F)$ strictly diagonally dominant? When?

- Jacobi iteration method finds approximation of the solution:

$$L^{(k+1)} = D^{-1}(E - R \cdot L^k)$$

$$L^{(k+1)} = (E + \rho F \cdot L^k)$$

This is exactly, what we started with...

D – diagonal of $(I - \rho F)$

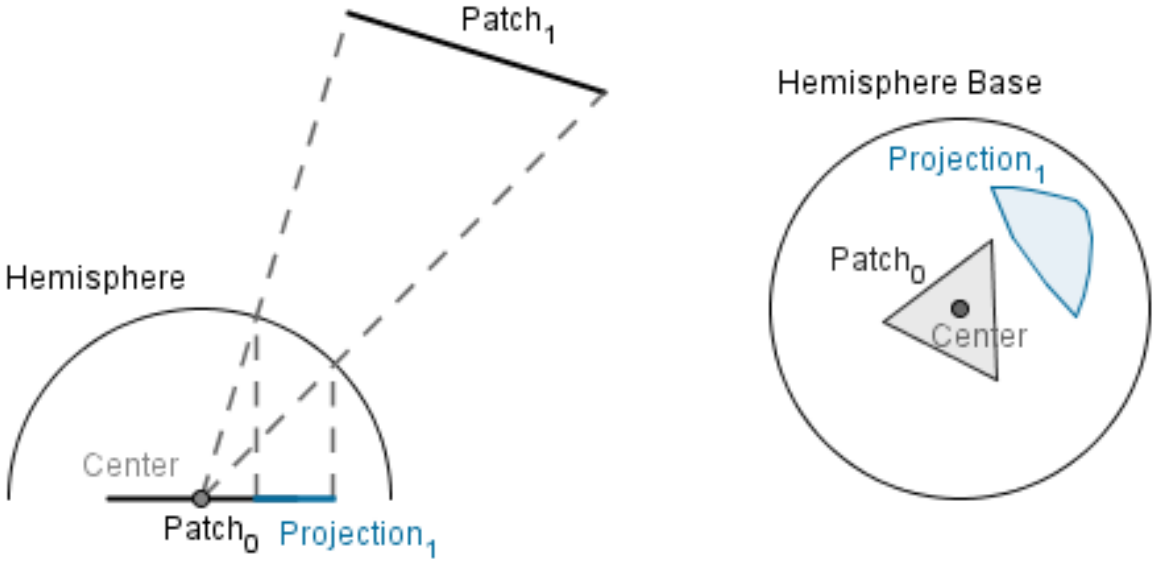
$R = (I - \rho F) - D = -\rho F$

L^k – k-th approximation

L^0 – initial random guess

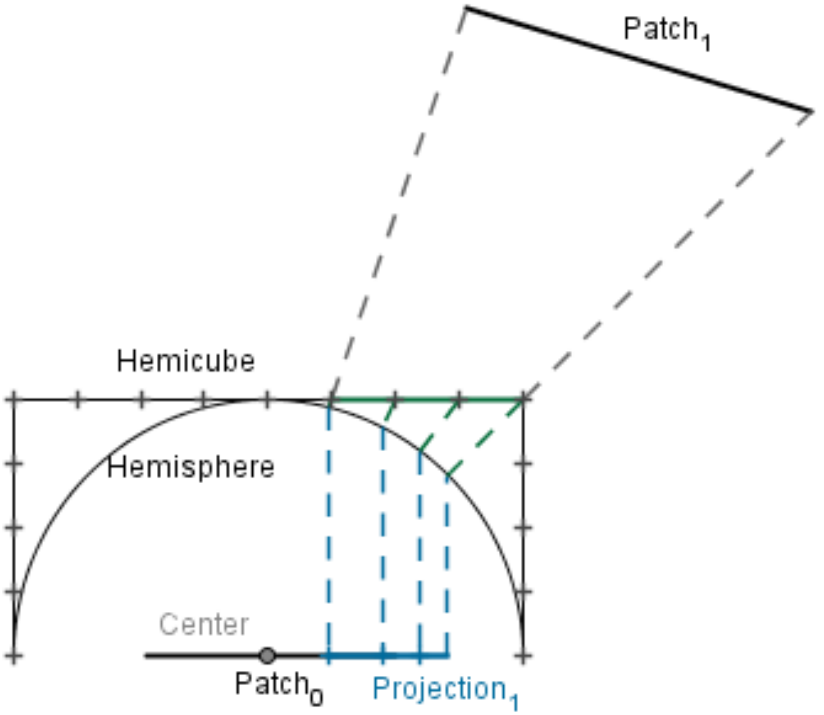
Radiosity

- Difficult part is to find the view factors for all the patches. One way to do it, is to:
  - Create an unit hemisphere around the patch
  - Project another patch onto the hemisphere
  - Then project that onto the base of the hemisphere
  - Ratio between the area of the last projection and the area of the base, is the view factor.



Radiosity

- Because finding the actual hemisphere projection for each patch is again complicated, we can create a hemicube.
- We create a rectangular grid on the hemicube and pre-calculate the hemisphere projections of each cell.
- Then we project other patches onto the cube, and see what cells they cover.



Global Illumination

- Different combinations of the algorithms, also many algorithms for real time approximations.
- Global Illumination in a Nutshell: http://www.thepolygoners.com/tutorials/GIIntro/GIIntro.htm
- Voxel Octree Cone Tracing – Real time, by Nvidia, 2011: https://research.nvidia.com/publication/interactive-indirect-illumination-using-voxel-cone-tracing
- Lightmass – Unreal Engine 4: https://docs.unrealengine.com/latest/INT/Engine/Rendering/LightingAndShadows/Lightmass/index.html

BRDF

- Disney BRDF explorer, to see different BRDF-s (incl. Lambert and Phong) in action: http://www.disneyanimation.com/technology/brdf.html
- Simon's Tech Blog, post about BRDF-s, includes interactive demos: http://simonstechblog.blogspot.com/2011/12/microfacet-brdf.html

What did you learn today?

What more would you like to know?

Next time: Holiday (1st of Dec)

Read about Environment Mapping

After that: Shadows, Conclusion!