

Rendering: The Rendering Equation

Adam Celarek



Research Division of Computer Graphics

Institute of Visual Computing & Human-Centered Technology

TU Wien, Austria



The Rendering Equation

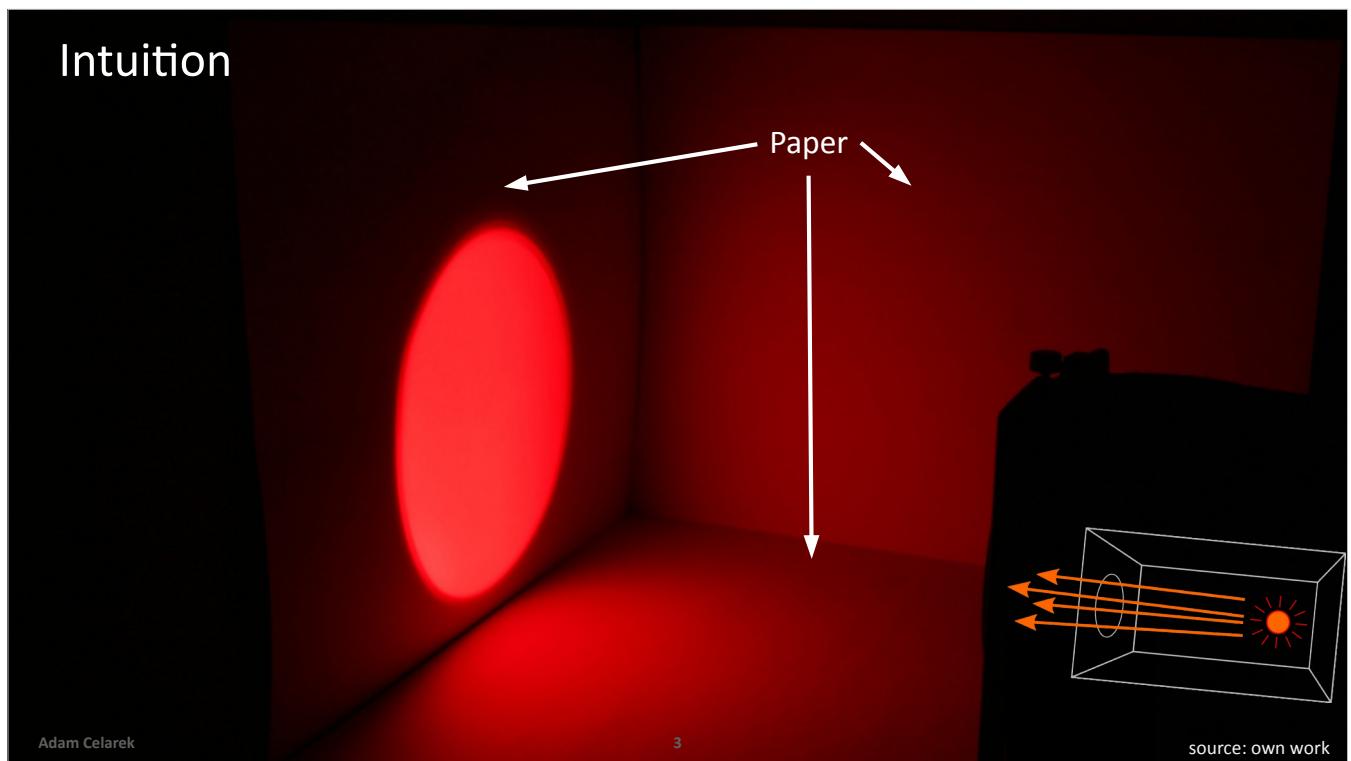
- Intuition
- Recursive Formulation
- Operator Formulation
- Path Integral Formulation

Adam Celarek

2

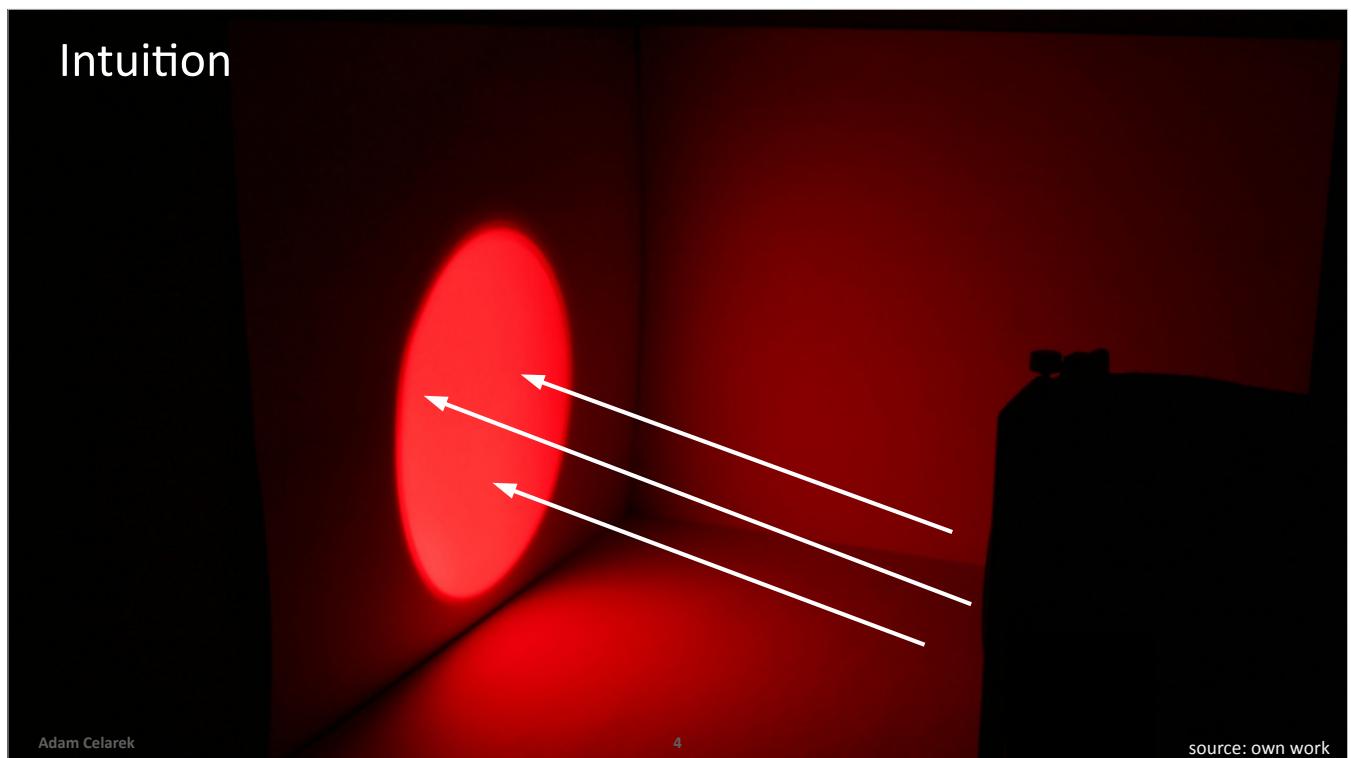
source: own work

Intuition



let's look at this scene

Intuition



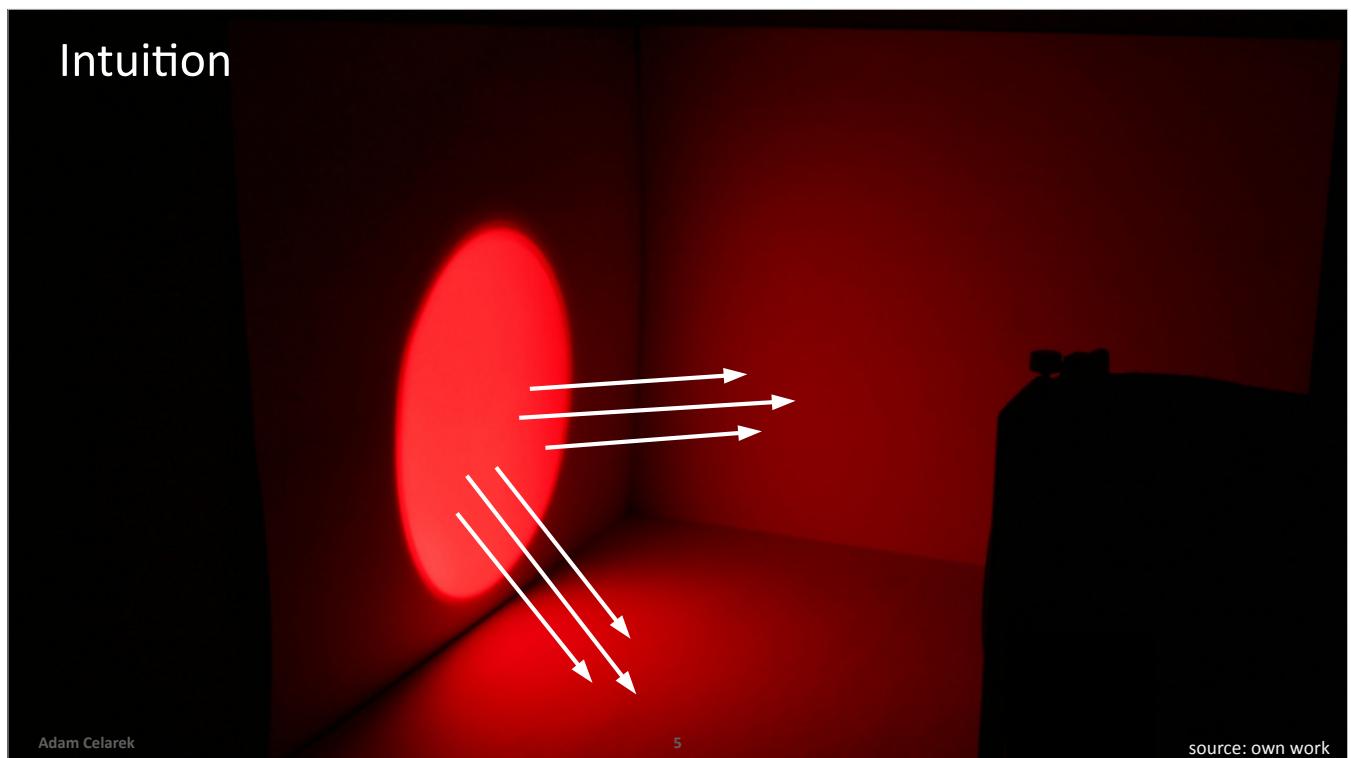
Adam Celarek

4

source: own work

how to compute. we can simulate what happens to photons

Intuition

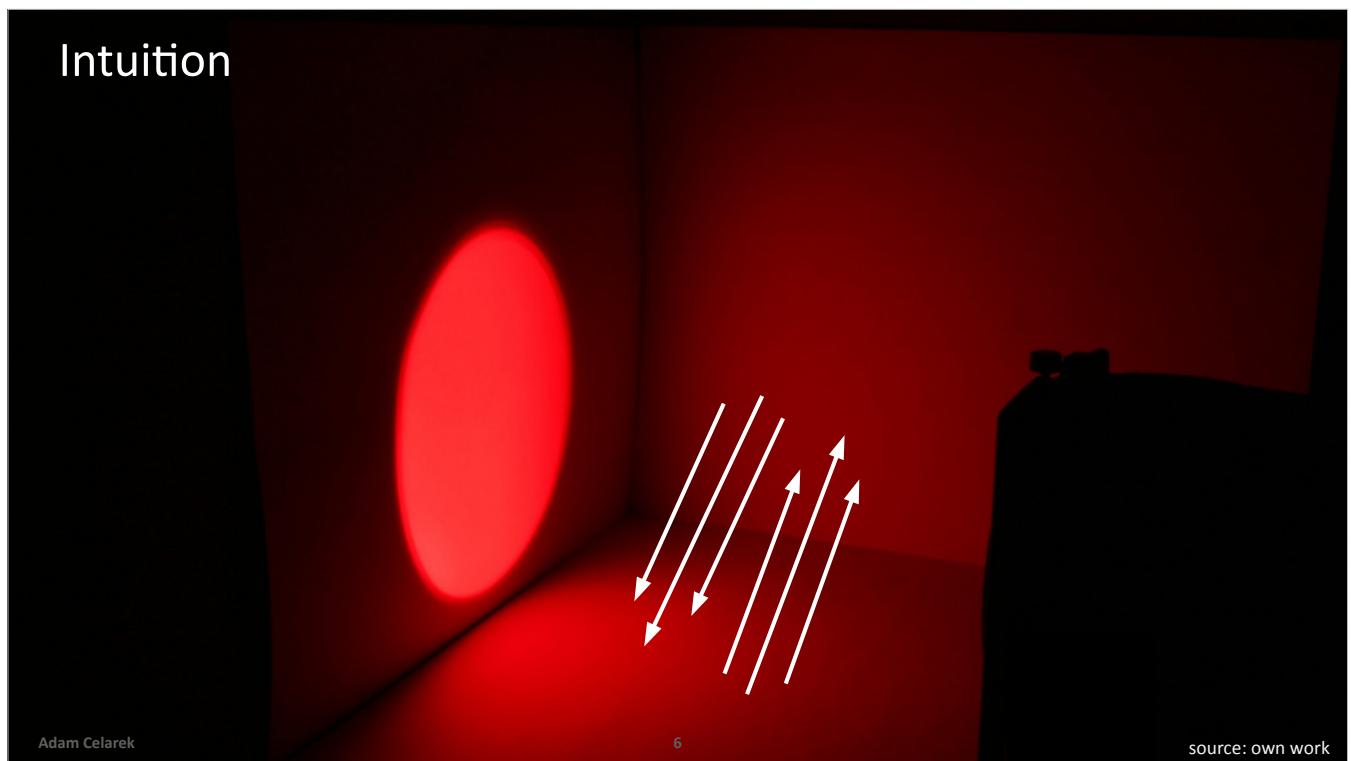


Adam Celarek

5

source: own work

Intuition

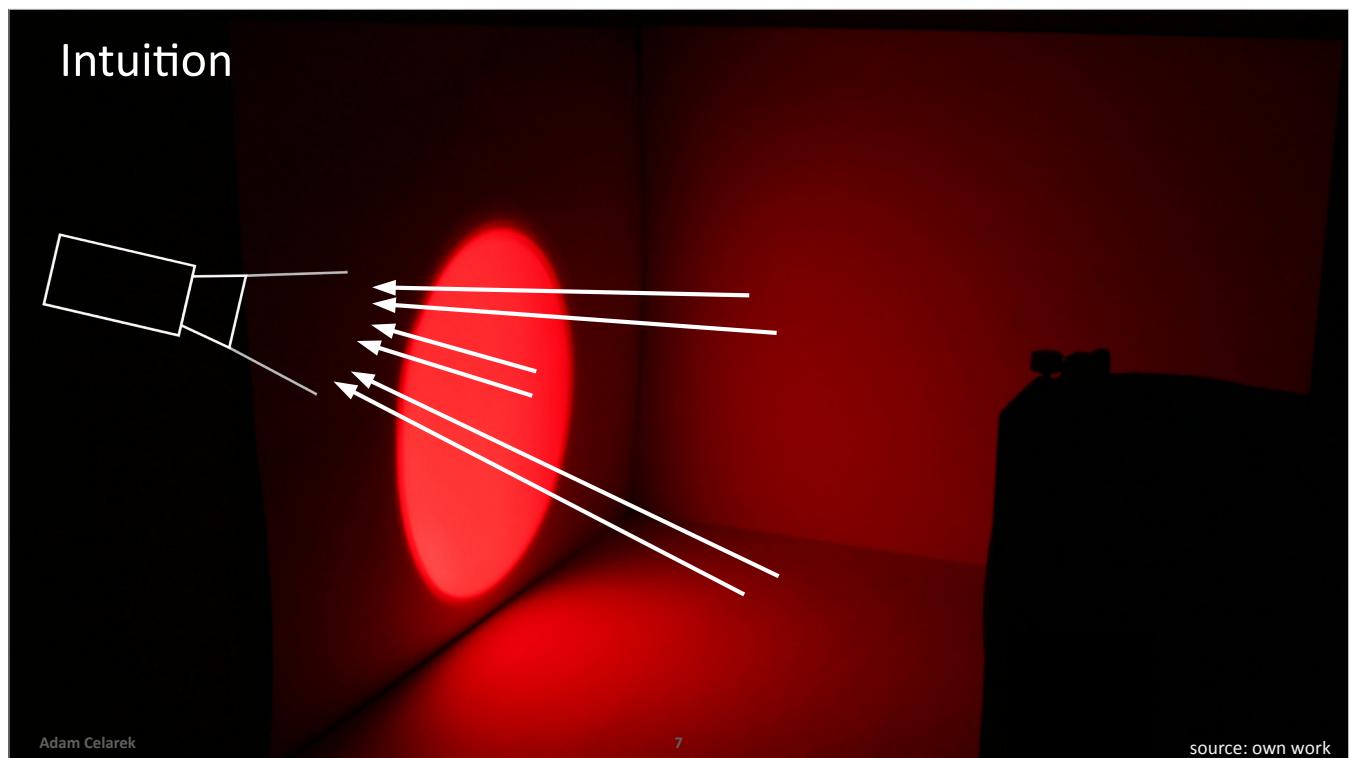


Adam Celarek

6

source: own work

Intuition



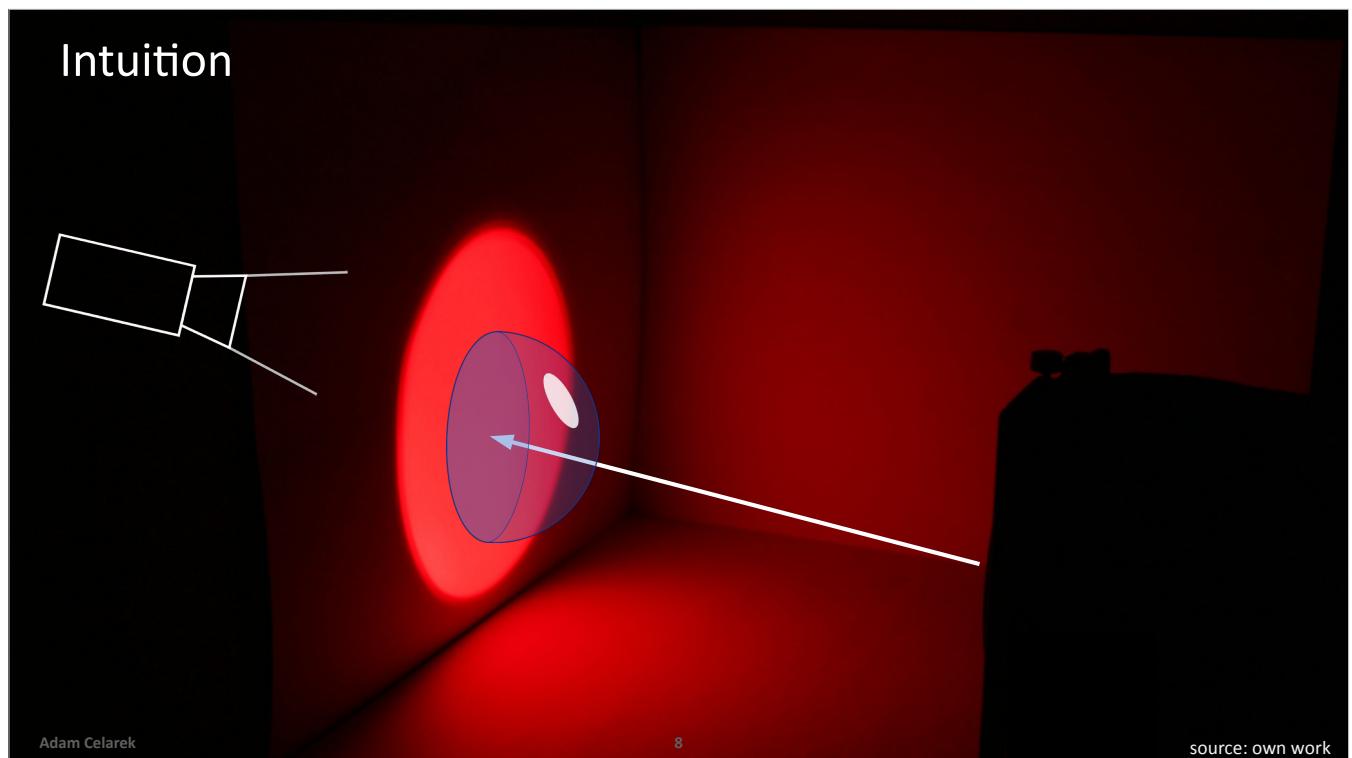
Adam Celarek

7

source: own work

This is a method. lots of variants, store photons in the surfaces (photon tracing, radiosity). can also do: trace paths of photons

Intuition



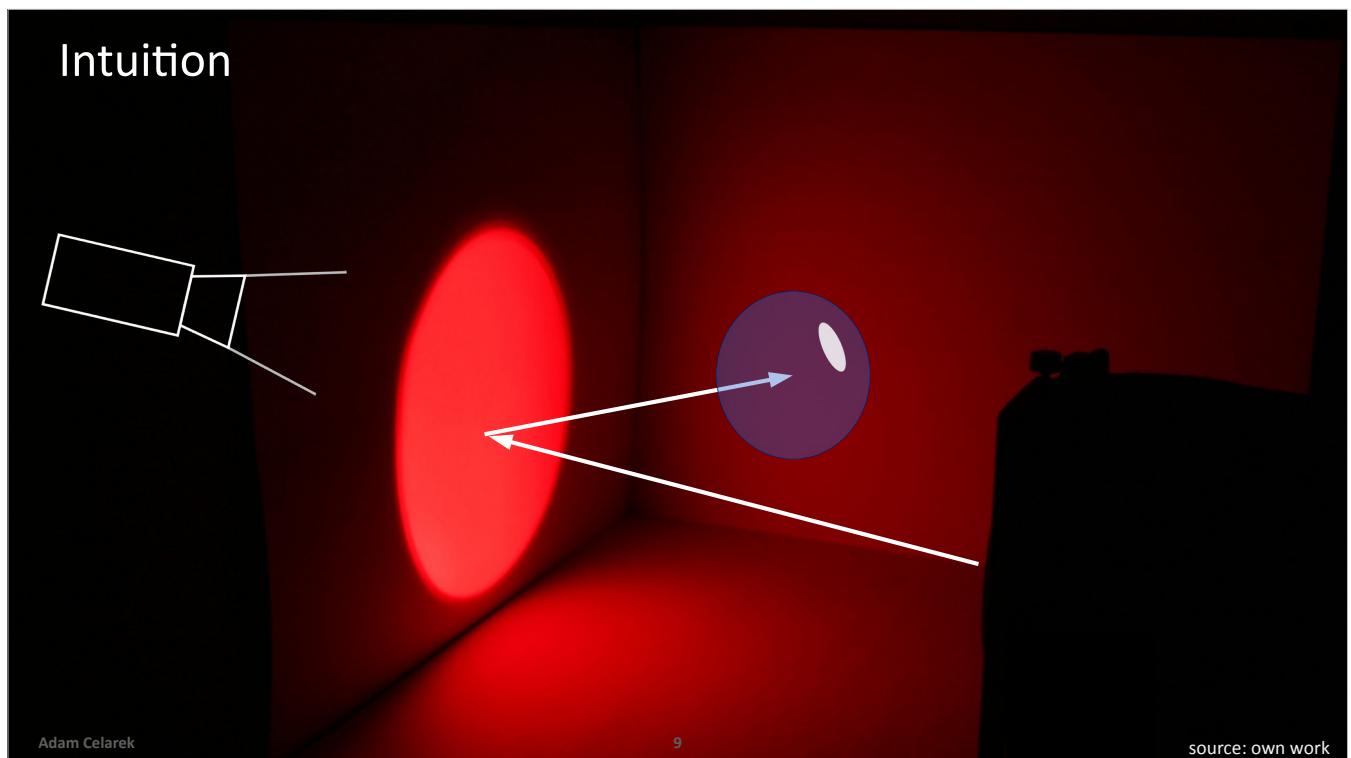
Adam Celarek

8

source: own work

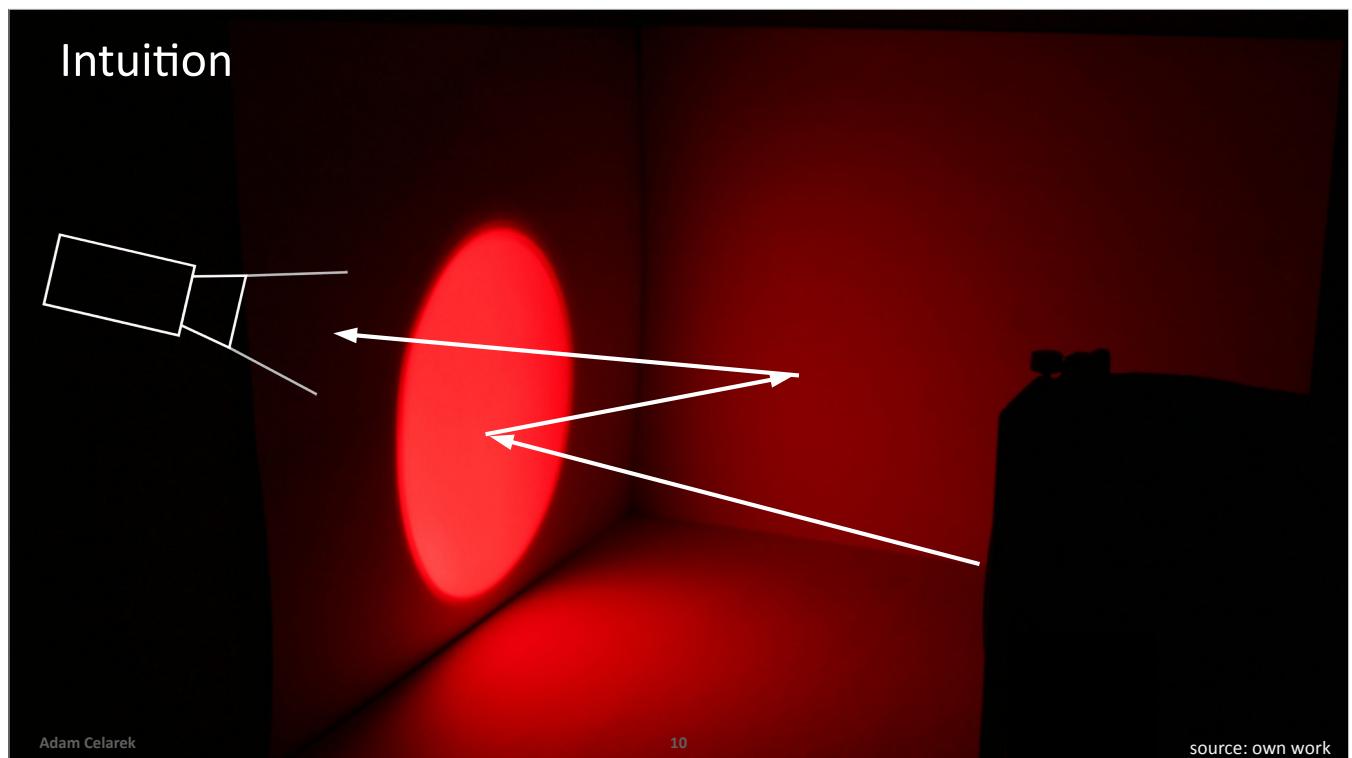
sample direction on hemisphere

Intuition



importance sampling is also possible (i.e. cast a ray directly to the camera)

Intuition



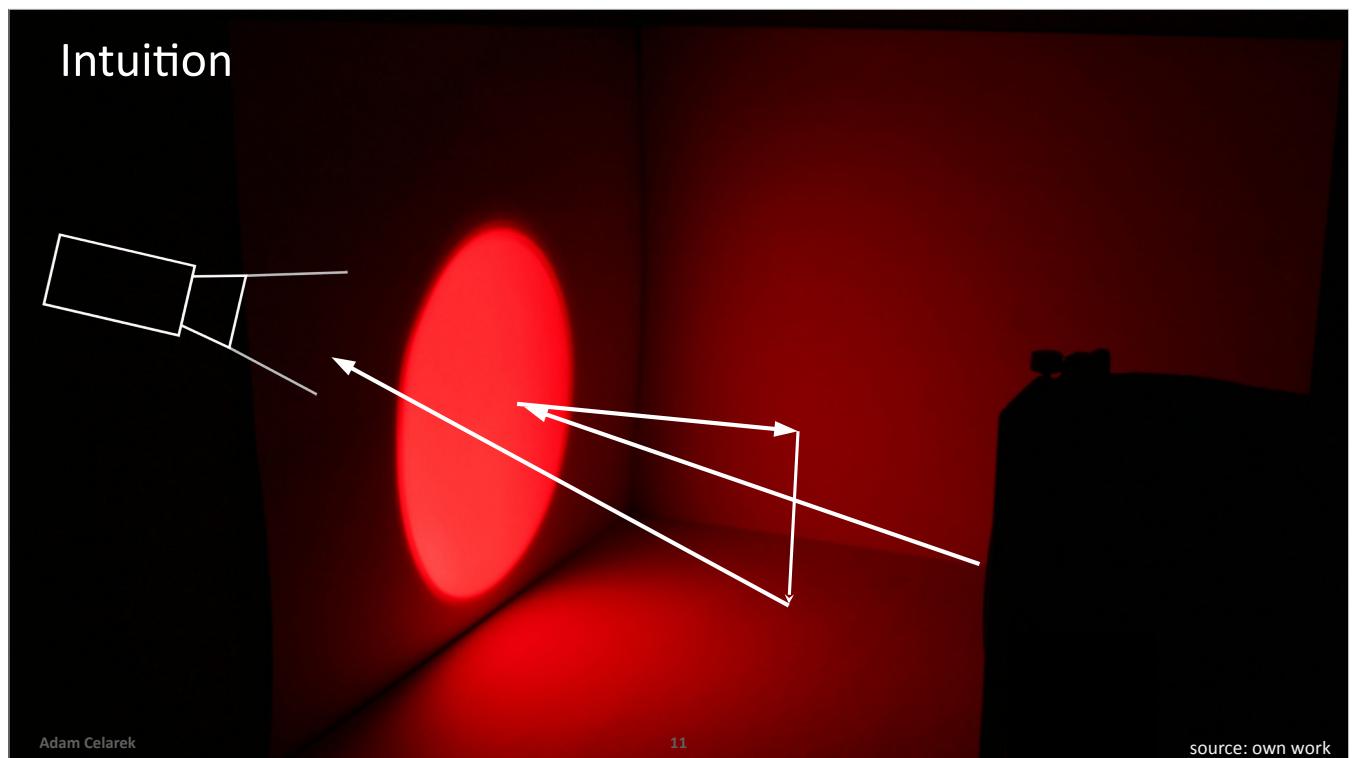
Adam Celarek

10

source: own work

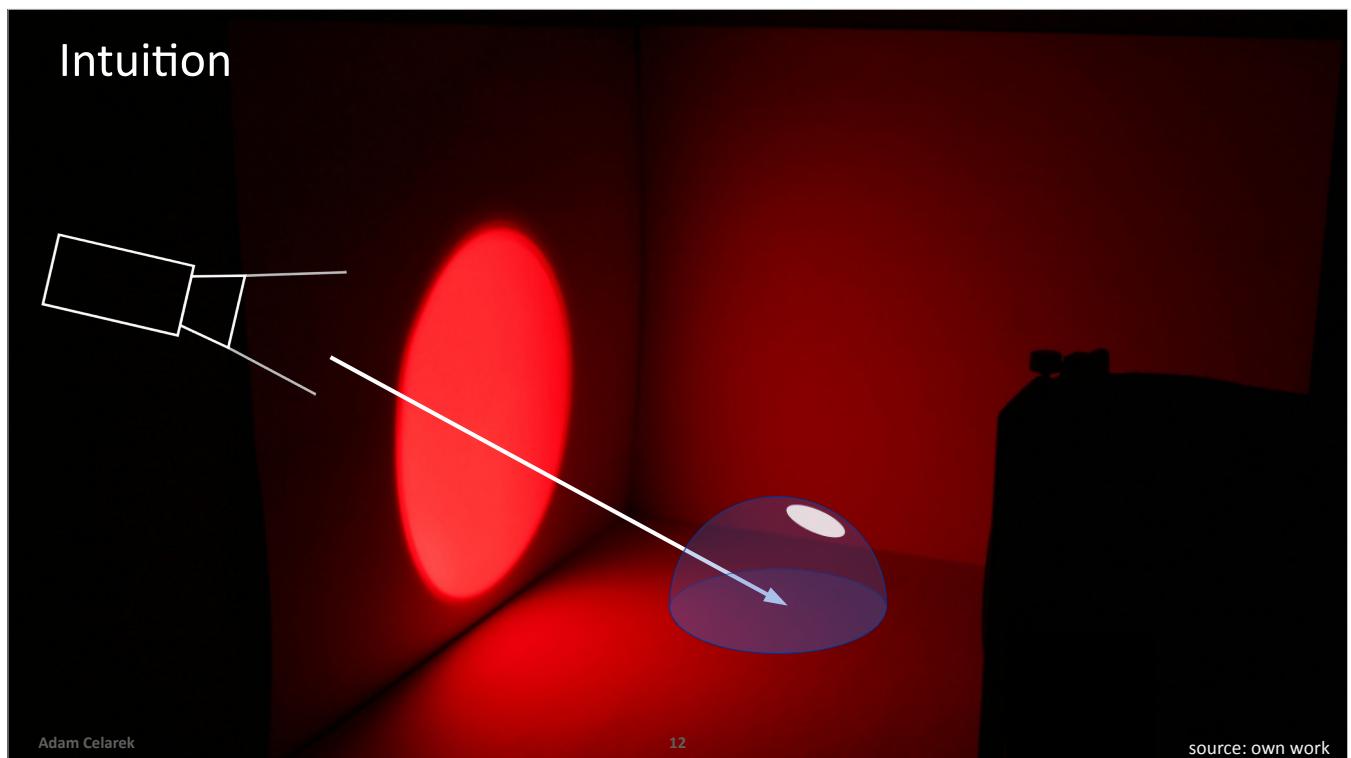
we have a full path. next slide: other paths can be sampled the same way

Intuition



finally add up per pixel in the camera (see, adding up -> integration)

Intuition



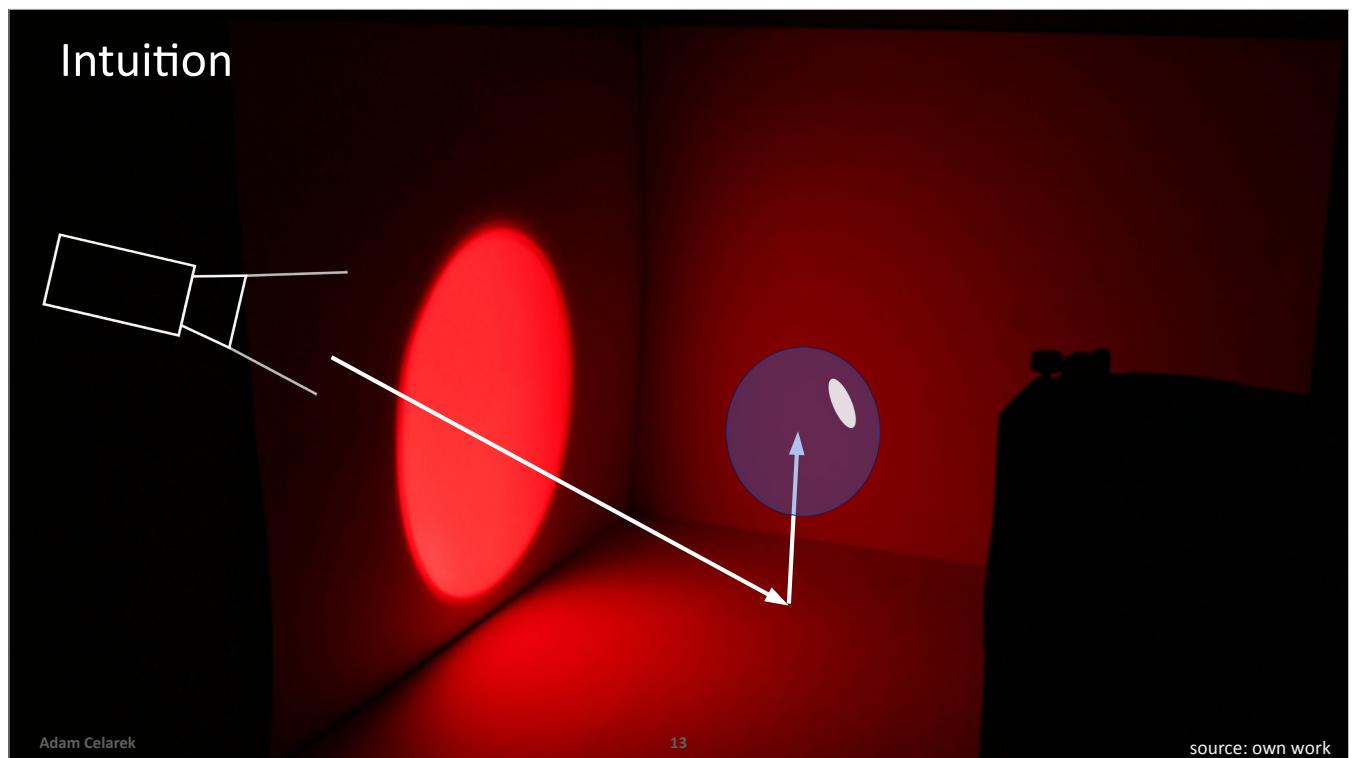
Adam Celarek

12

source: own work

can also start at camera. integrate over hemisphere.

Intuition

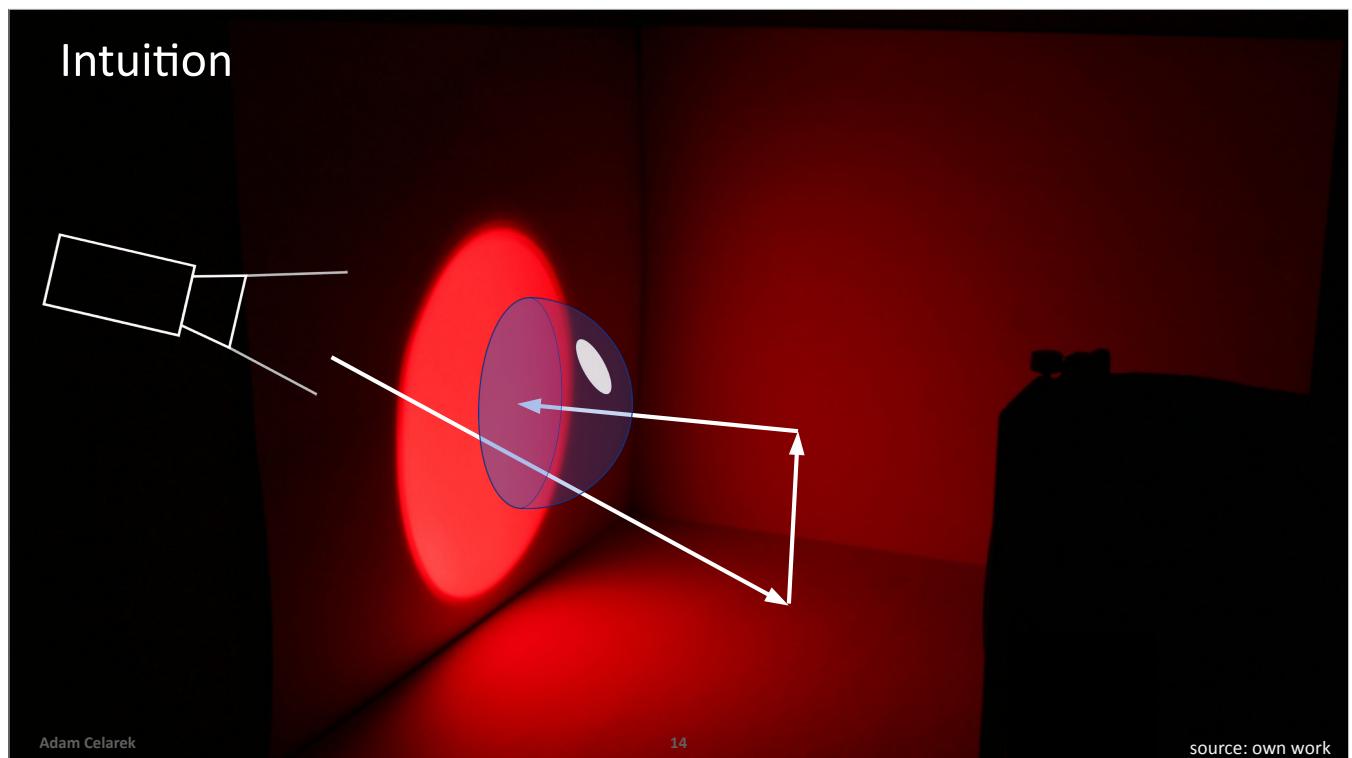


Adam Celarek

13

source: own work

Intuition

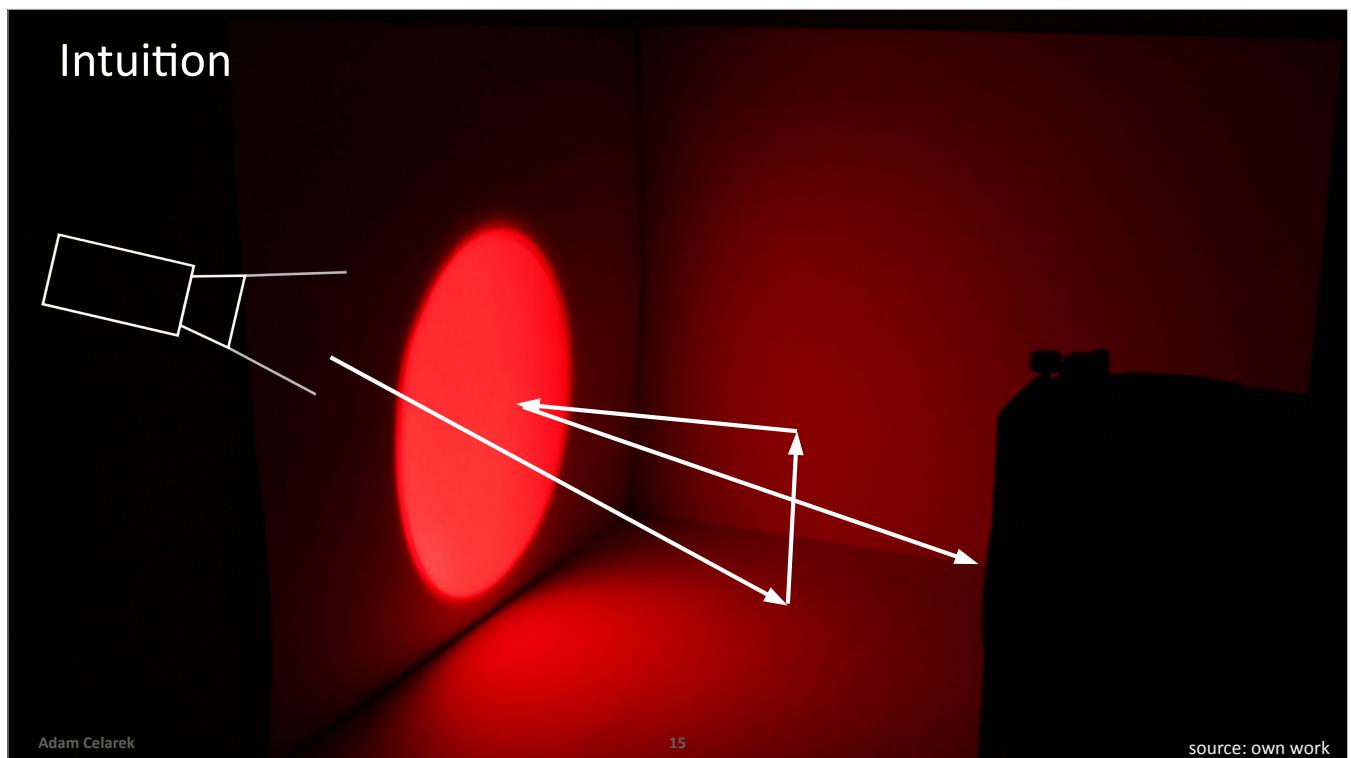


Adam Celarek

14

source: own work

Intuition



Adam Celarek

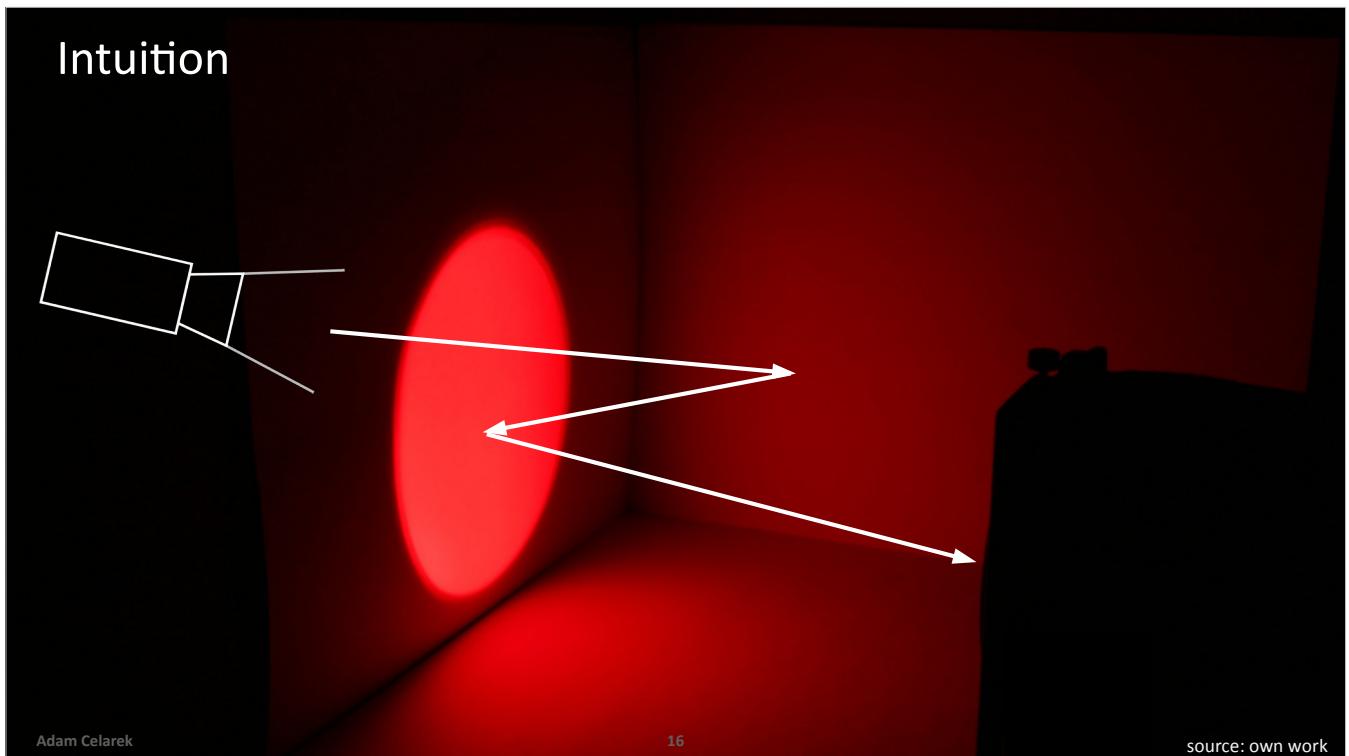
15

source: own work

we have a full path. only with a full path we have a light measurement (contribution to the pixel).

next slide: other paths can be sampled the same way

Intuition



Adam Celarek

16

source: own work

collect factors on its way to the light, that are multiplied with the radiance of the light to compute the contribution.

tracing importons, adjoint operation.

tracing ‘bundles of photons’, which become fewer every reflection.

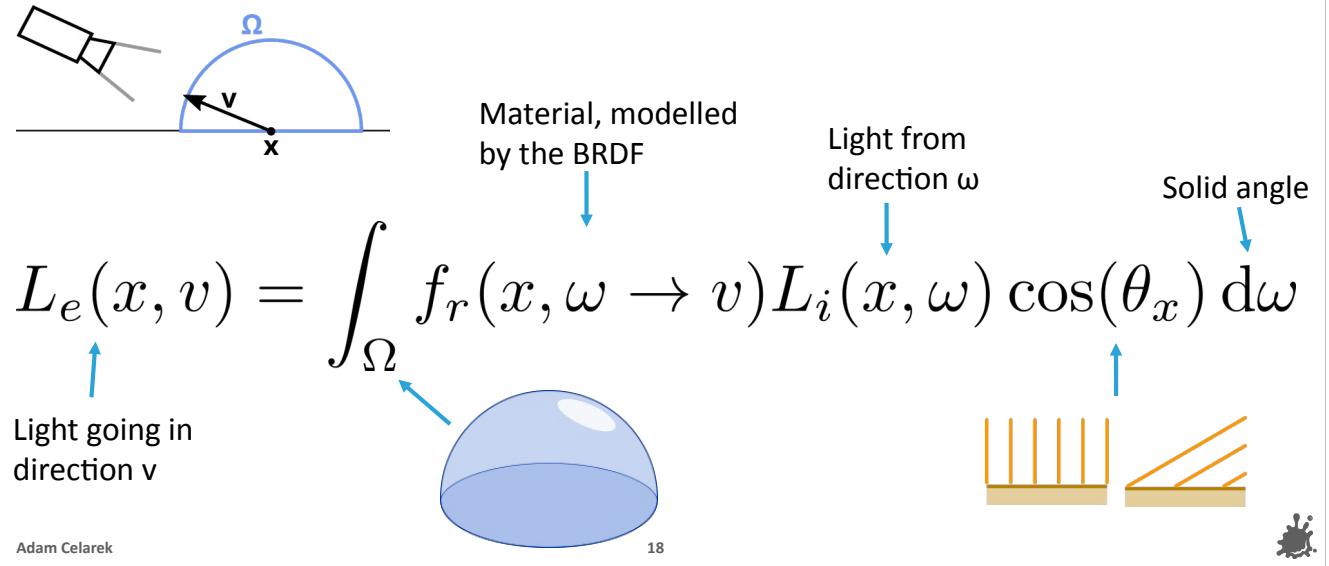
trace ‘bundles of importons’

Intuition

Photons are emitted from light sources, reflected by surfaces in the scene until they reach the sensor.

In rendering, we (can) go the opposite way. We trace *importons* until they reach a light source.

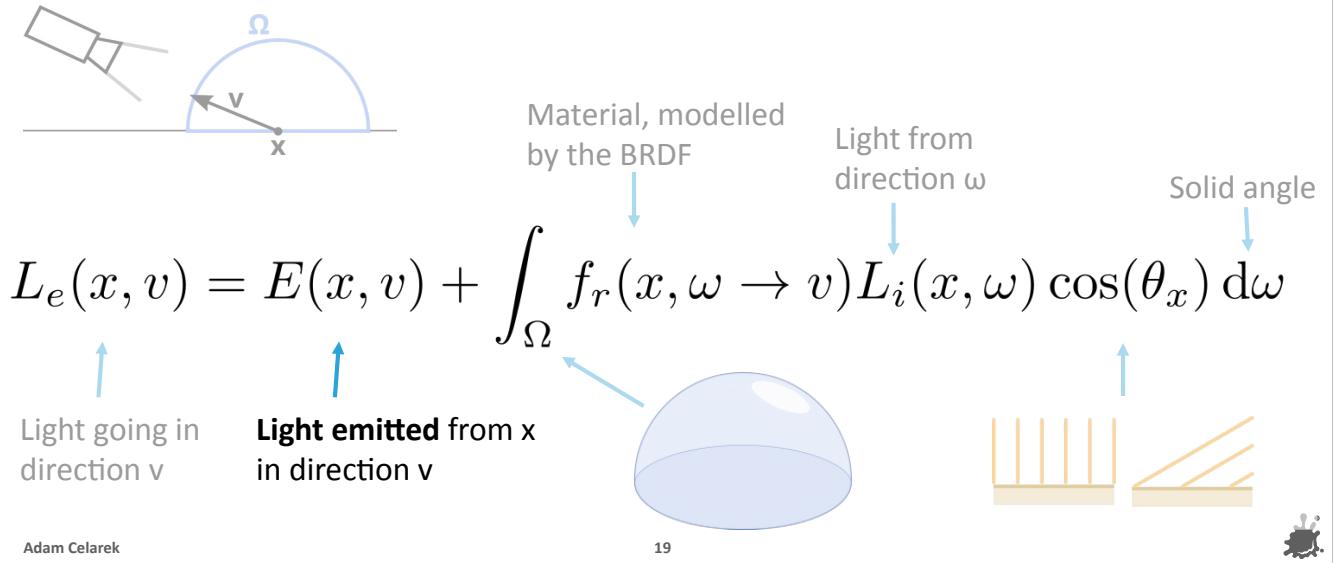
Next: Recap and recursive formulation



Recap light integral:

Compute the light which is going into direction v , integrate over hemisphere, check all directions for incoming light, cosine weighting and material.

next slide: The first think we have to add is light emittance.



The first think we have to add is light emittance.

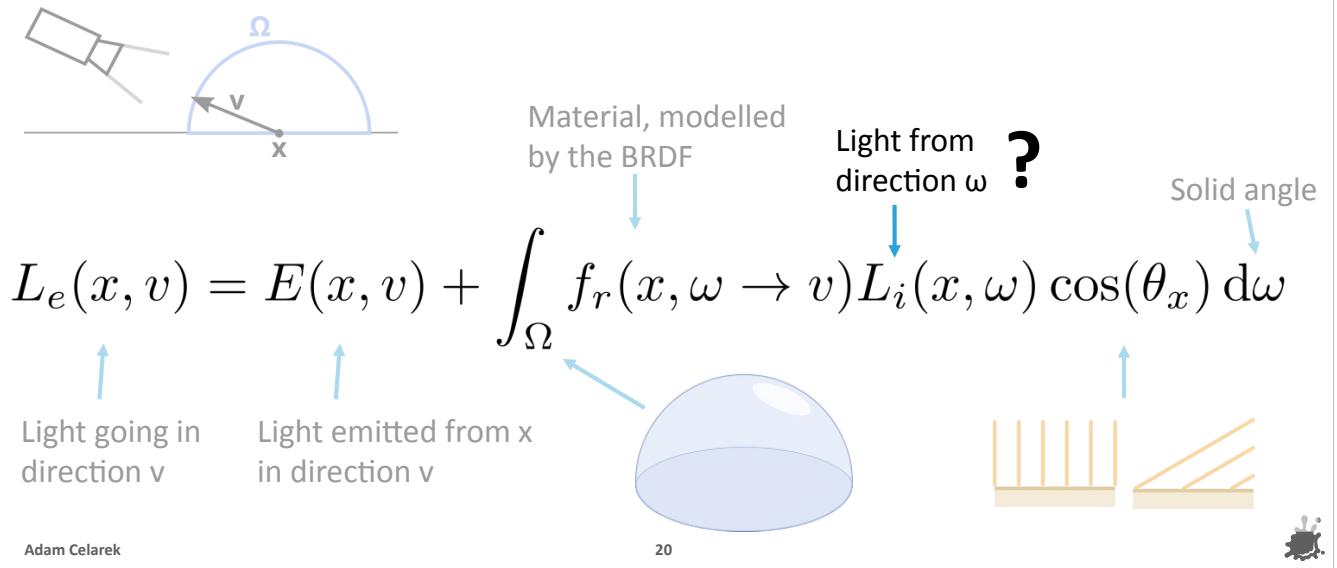
Imagine the camera is directed right at a light source, then the emitted light will be the dominating factor.

Some light sources have a larger radiance at certain positions or in certain directions (think of a head lamp in a car), therefore the Emittance E depends on the position and the direction.

The right part of the sum is the same as before: integral over the hemisphere of light from direction ω , weighted by the cosine and the brdf.

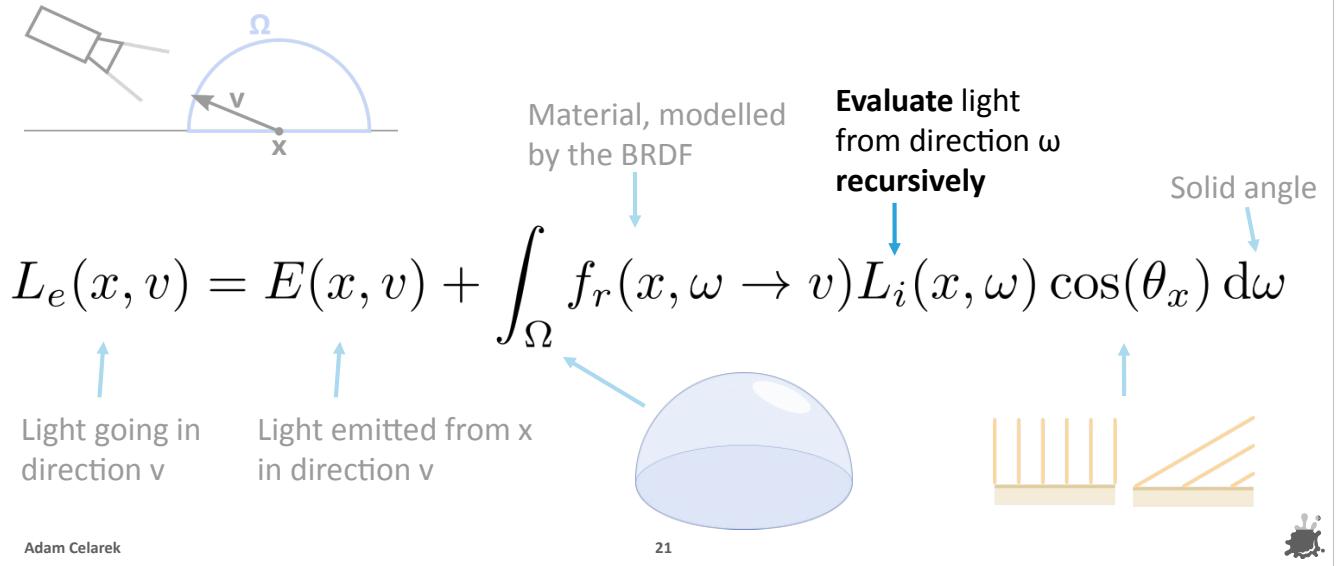
Next: But how to get the radiance coming from direction ω ?

Recursive Formulation



But how to get the radiance coming from direction ω ? What can we do?

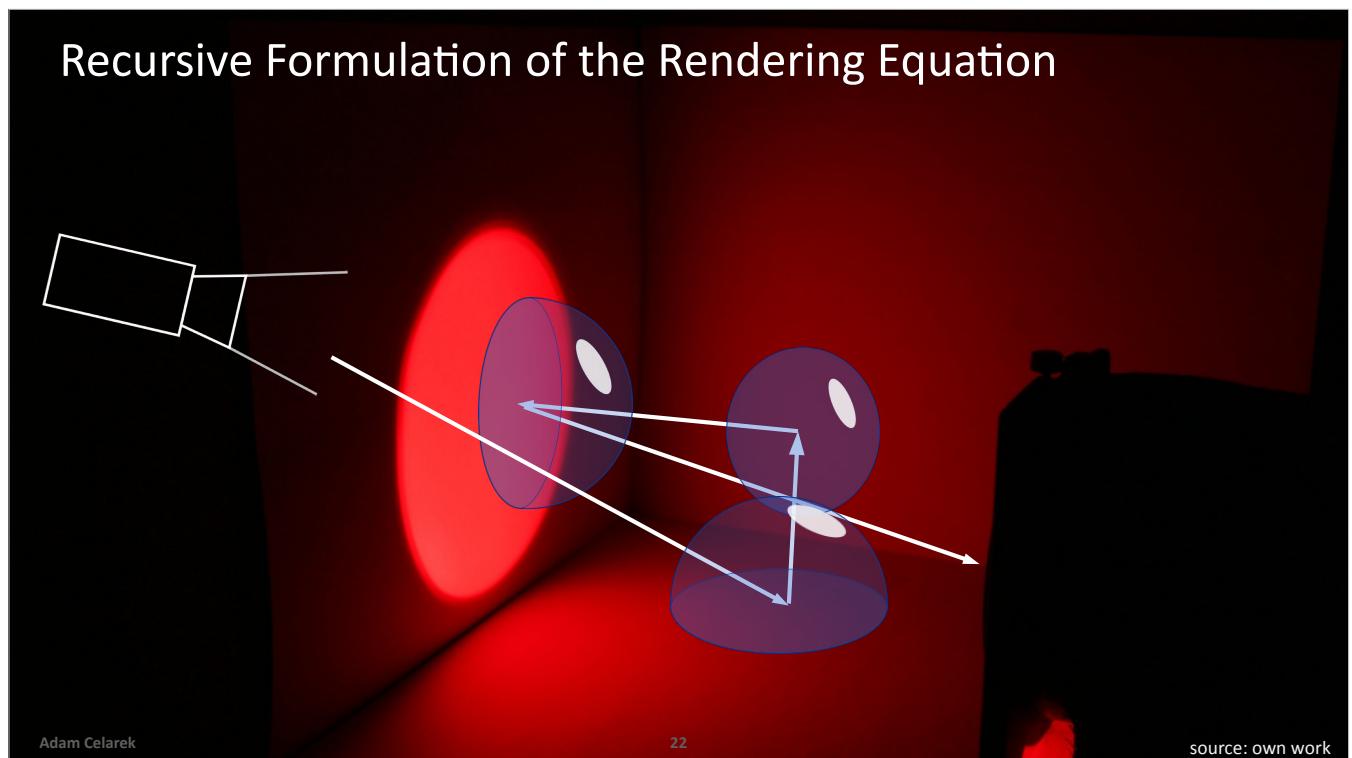
Recursive Formulation of the Rendering Equation



Well, this is named recursive formulation. So probably we will get it recursively :)

We can sample a ray on the hemisphere..

Recursive Formulation of the Rendering Equation



.. continue recursively until it reaches the light source

Questions?



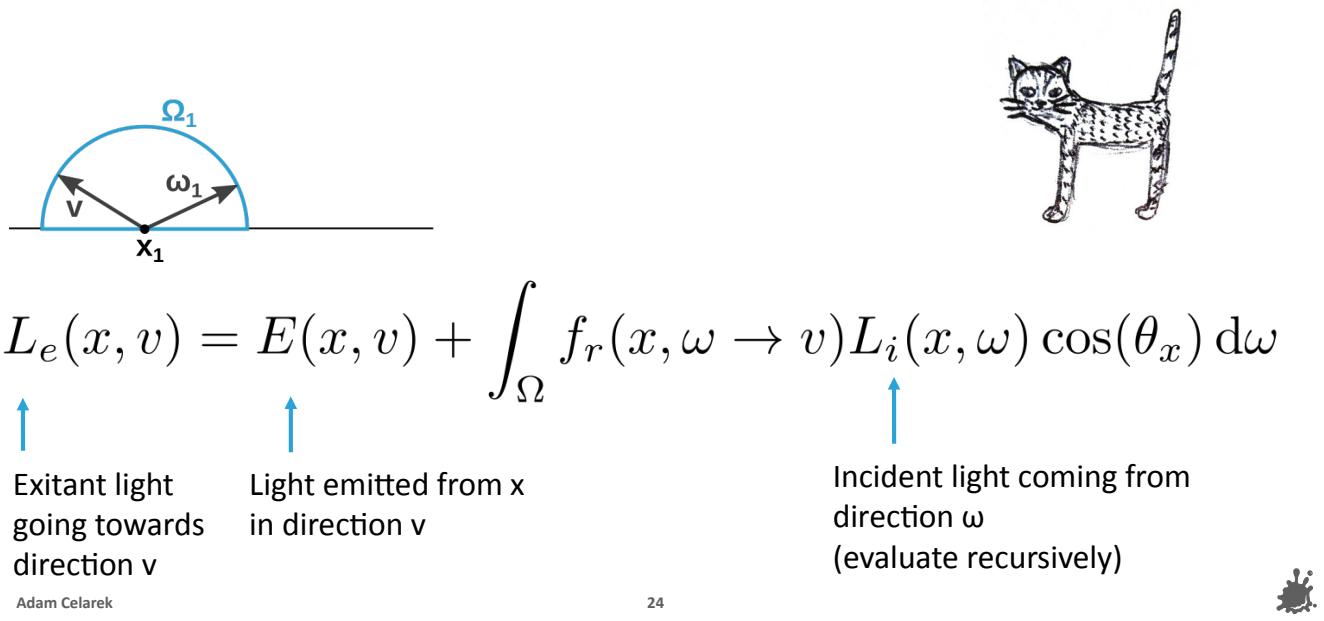
Adam Celarek

23

source: own work

yes, the cat has a question

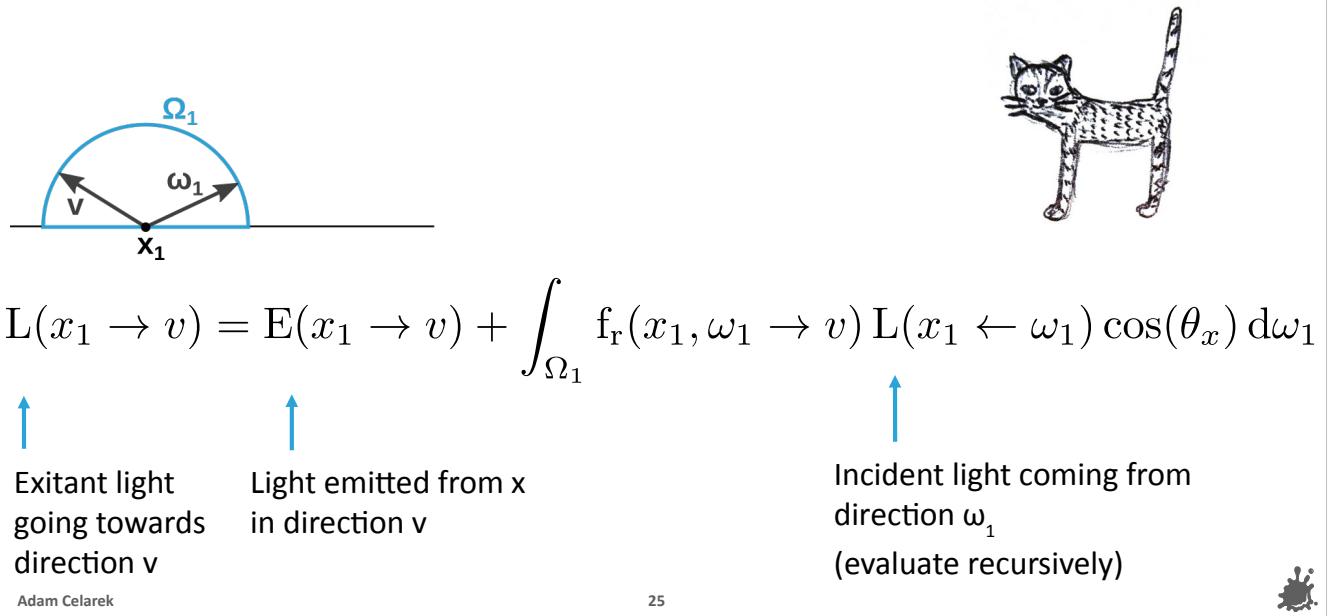
Recursive Formulation of the Rendering Equation



Yes, the cat has a question, but first we make a change in notation.

Look at exitant, emitted and incident light.

Recursive Formulation of the Rendering Equation

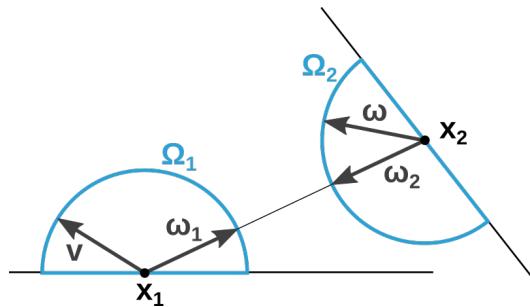


We now use arrows to show the direction of photons.
(however, ω still points away of the point x).

We also changed the name of the differential (added a 1), but that is just a variable name.

Next: We said recursion, ..

Recursive Formulation of the Rendering Equation



$$L(x_1 \rightarrow v) = E(x_1 \rightarrow v) + \int_{\Omega_1} f_r(x_1, \omega_1 \rightarrow v) L(x_1 \leftarrow \omega_1) \cos(\theta_x) d\omega_1$$

$$L(x_2 \rightarrow \omega_2) = E(x_2 \rightarrow \omega_2) + \int_{\Omega_2} f_r(x_2, \omega \rightarrow \omega_2) L(x_2 \leftarrow \omega) \cos(\theta_x) d\omega$$

Adam Celarek

26



This is one expansion of such a recursion.

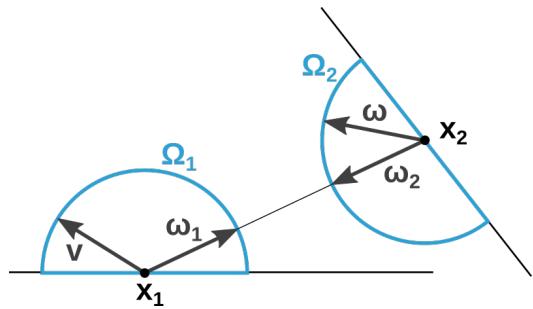
We are standing on position x_1 and want to know how much light is coming from directions $d\omega_1$ (the whole hemisphere!)

From a mathematical standpoint we are not sending rays, at least not a finite number of rays. We integrate over the hemisphere.

However, in the spirit of Monte Carlo and as a mental picture, we can trace a ray into direction ω_1 to look what there is. We hit a point x_2 , and we can compute the exitant radiance for ω_2 ($\omega_2 = -\omega_1$).

But, (next slide)

Recursive Formulation of the Rendering Equation



$$L(x_1 \rightarrow v) = E(x_1 \rightarrow v) + \int_{\Omega_1} f_r(x_1, \omega_1 \rightarrow v) L(x_1 \leftarrow \omega_1) \cos(\theta_x) d\omega_1$$

$$L(x_1 \leftarrow \omega_1) = L(x_2 \rightarrow \omega_2) ?$$

$$L(x_2 \rightarrow \omega_2) = E(x_2 \rightarrow \omega_2) + \int_{\Omega_2} f_r(x_2, \omega \rightarrow \omega_2) L(x_2 \leftarrow \omega) \cos(\theta_x) d\omega$$



Inside box: On the left/top we have incoming radiance, on the right/bottom we have exitant radiance.

Cat: Is that the same?

Radiance $L =$

flux per unit projected area per unit solid angle

$$L = \frac{d\Phi}{dA^\perp d\omega}$$

dA , $d\omega$ and $d\Phi$ are differentials. check out [3blue1brown](#), if you want a really good explanation

$$[L] = \left[\frac{W}{m^2 sr} \right]$$



we had that already in the lecture about light. Back then, we were looking at radiance.

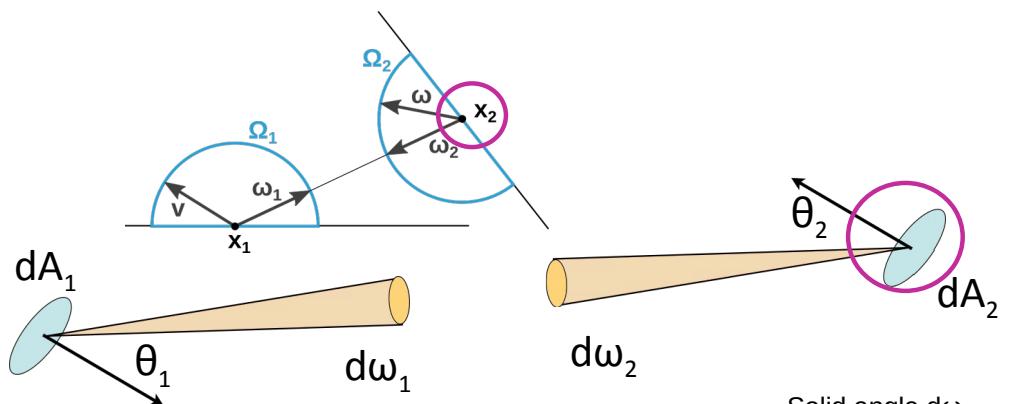
Radiance is the differential flux (measured in Watts, think of number of photons) per unit projected area per unit solid angle.

“ dA projected” accounts for tilting dA , that is the cosine rule.

And $d\omega$ means that we are looking at a infinitesimal angle

Therefore we are looking at the amount of energy (number of photons) that are flying into directions $d\omega$ in a beam of width dA projected.

Recap About Physics



$$L = \frac{d\Phi}{dA^\perp d\omega}$$

$$d\Phi = L(x_2 \rightarrow \omega_2) \overbrace{\cos \theta_2 dA_2}^{dA_2^\perp} \overbrace{\frac{dA_1 \cos \theta_1}{r^2}}^{d\omega_1}$$



We calculate the differential flux ($d\Phi$) that is sent from area differential A_2 towards area differential A_1 . This answers the question about how much energy leaves.

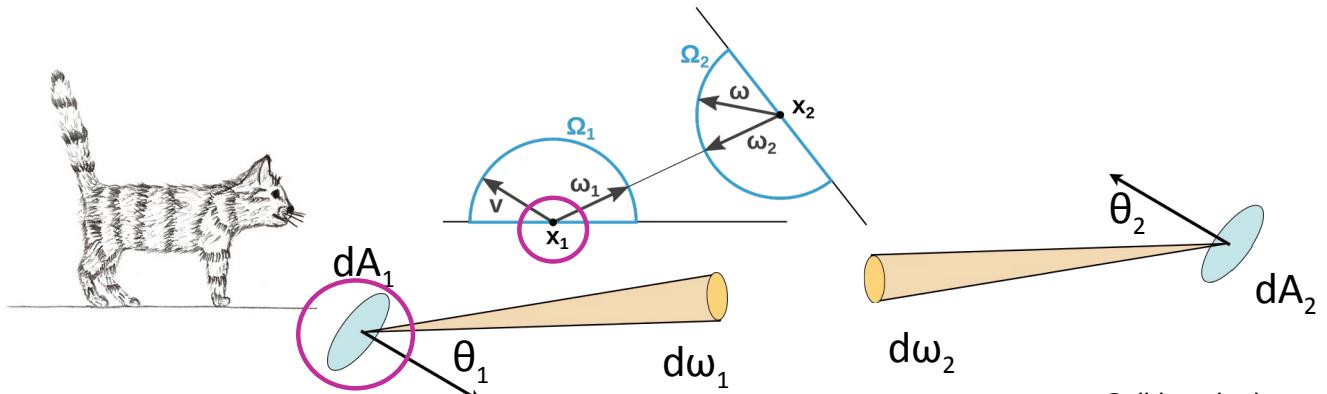
(You can see the calculation at the bottom.)

$d\omega_2$ is the solid angle subtended by dA_1 as seen from dA_2 . Photons don't make turns, so all energy that is sent towards $d\omega_2$ will reach dA_1 .

$L(x_2 \rightarrow \omega_2)$ is the radiance sent by dA_2 into direction ω_2 , $\cos \theta_2 dA_2$ is the projected area (beam width at the start), and the fraction is just the solid angle $d\omega_2$.

Ok, let's now turn to our receiver.

Recap About Physics



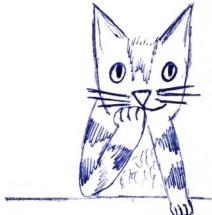
$$L = \frac{d\Phi}{dA^\perp d\omega}$$

$$d\Phi = L(x_1 \leftarrow \omega_1) \overbrace{\cos \theta_1 dA_1}^{dA_1^\perp} \overbrace{\frac{dA_2 \cos \theta_2}{r^2}}^{d\omega_2}$$



$L(x_1 \leftarrow \omega_1)$ is the radiance received by dA_1 from directions $d\omega_1$.

In order to compute the differential flux (energy), we again have to compute the projected area for dA_1 , and the angles $d\omega_1$ (which is the solid angle subtended by dA_2 as seen from dA_1).



$$d\Phi = L(x_2 \rightarrow \omega_2) \cos \theta_2 dA_2 \frac{dA_1 \cos \theta_1}{r^2}$$

$$d\Phi = L(x_1 \leftarrow \omega_1) \cos \theta_1 dA_1 \frac{dA_2 \cos \theta_2}{r^2}$$

$$\Rightarrow L(x_2 \rightarrow \omega_2) = L(x_1 \leftarrow \omega_1)$$

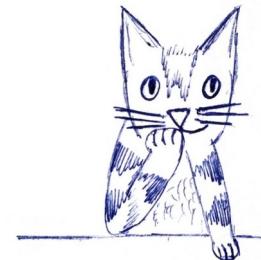
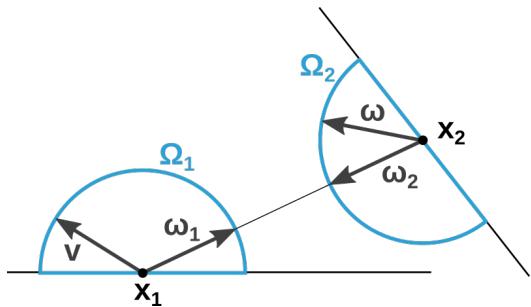


Let's put those two equations right next to each other.

As said, we know that photons don't make turns (not in vacuum), therefore both of the $d\Phi$ (energy) are the same and we can equate the top equation with the bottom one.

We quickly see, that all factors but the $L(..)$ are the same. Hence the amount of radiance going from x_1 towards direction ω_1 is the same as reaching x_2 from direction ω_2 .

Recursive Formulation of the Rendering Equation



$$L(x_1 \rightarrow v) = E(x_1 \rightarrow v) + \int_{\Omega_1} f_r(x_1, \omega_1 \rightarrow v) L(x_1 \leftarrow \omega_1) \cos(\theta_x) d\omega_1$$

$$L(x_1 \leftarrow \omega_1) = L(x_2 \rightarrow \omega_2) !$$

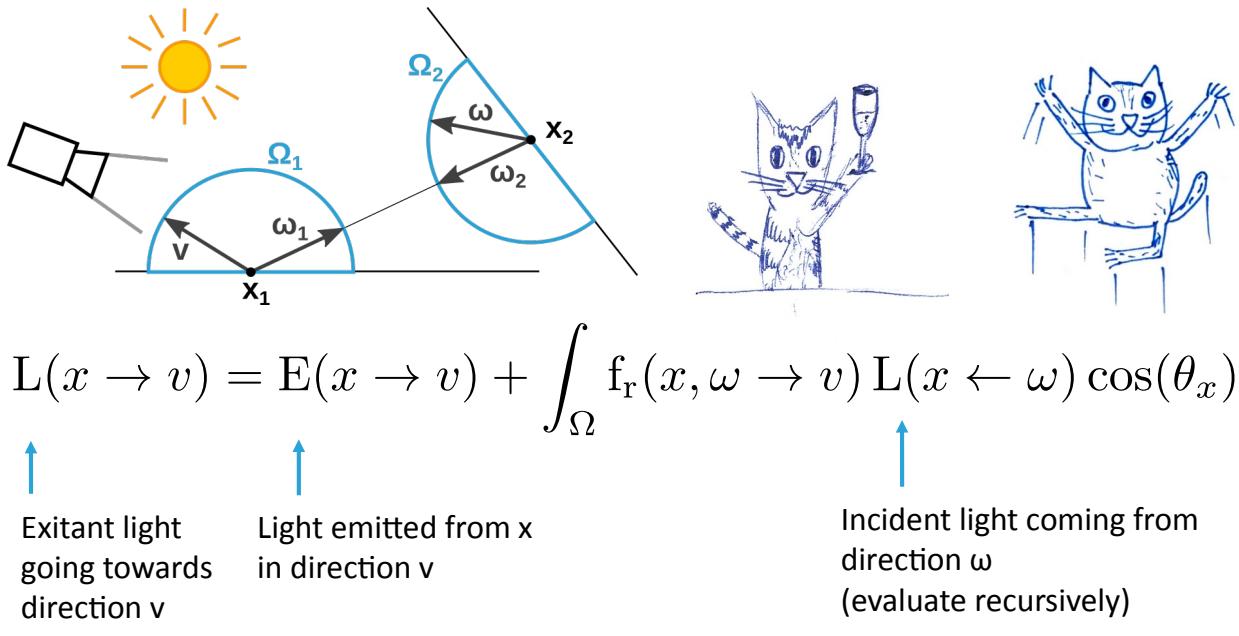
$$L(x_2 \rightarrow \omega_2) = E(x_2 \rightarrow \omega_2) + \int_{\Omega_2} f_r(x_2, \omega \rightarrow \omega_2) L(x_2 \leftarrow \omega) \cos(\theta_x) d\omega$$



Let's look at the recursive formulation again.

Ok, cool. We can do this. The cat is happy.

Recursive Formulation of the Rendering Equation



Adam Celarek

33



We start from camera \rightarrow we get a hit point \rightarrow get the emitted light + the reflected light. When computing the reflected light, we have to trace a ray again \rightarrow we get a hit point $\rightarrow \dots$

Realise that the problem is infinitely dimensional. not possible to write down analytical solution for any practical scene. have to solve numerically.

monte carlo can deal with many dimensions. But still, in practice we have to stop at some point, and we will learn soon how to do that in an unbiased way (unbiased means, that we will have the correct result on average).

As said this is the adjoint method, we are tracing importants. And yes, the very same integral also works for photons. In that case ' $E(x, v)$ ' is the camera sensor emitting importance (for each sensor element = pixel separately). We would then measure, how much importance reaches the light surface. Multiplied with the amount of emitted light this would give us the same value, and we would be able to update the corresponding pixel that sent the emission. This might sound extremely inefficient, but that isn't the case. Just like we can sample a light source directly, we could also sample the camera directly, and the method becomes feasible.

Recursive Formulation of the Rendering Equation

- First published: The rendering equation, James Kajiya, Siggraph 1986
- This is the most important formulation
- It is used for path tracing, the most common algorithm for physically based rendering
- But path tracing or even MC is not the only method to solve the rendering equation, see later.



$$L(x \rightarrow v) = E(x \rightarrow v) + \int_{\Omega} f_r(x, \omega \rightarrow v) L(x \leftarrow \omega) \cos(\theta_x) d\omega$$



- Next: Operator formulation

Adam Celarek

35

source: own work

$$L = L_e + TL$$



neat, isn't it? let's have a look at what the symbols mean

$$L = L_e + TL$$

Emitted light in the scene

Light transport operator

All light in the scene



written here in terms of radiance / photos / light propagation. but very similar for importance / importons (adjoint operator)

think of radiance stored on surfaces. then iterate to solve

Do you see what that is? L on the left and on the right is the same L . we're looking for the solution, where light propagation is in equilibrium. similar to $x = a + bx$ or such matrix problems. in fact this is also a linear system but with functions instead of simple vectors.

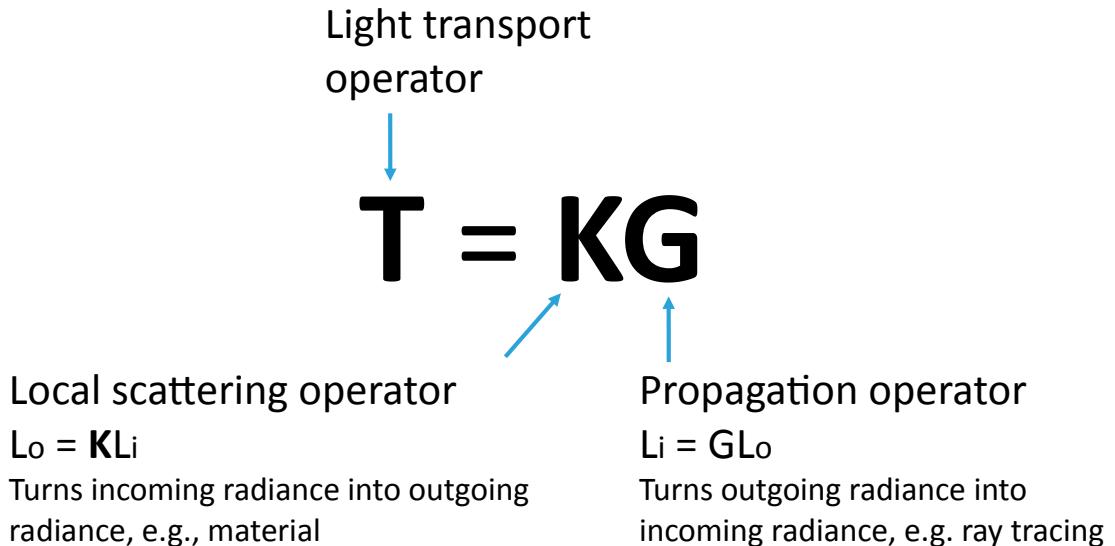
```
>>> a = 1.5  
>>> b = 0.7  
>>> x = 1  
>>> x = a + b * x; print(x) # 2.2  
>>> x = a + b * x; print(x) # 3.04  
>>> x = a + b * x; print(x) # 3.628  
>>> x = a + b * x; print(x) # 4.0396  
>>> x = a + b * x; print(x) # 4.32772  
>>> x = a + b * x; print(x) # 4.529404  
>>> x = a + b * x; print(x) # 4.6705828
```

```
>>> x = a + b * x; print(x) # 4.76940796  
>>> x = a + b * x; print(x) # 4.838585572  
>>> x = a + b * x; print(x) # 4.8870099004  
>>> x = a + b * x; print(x) # 4.92090693028  
>>> x = a + b * x; print(x) # 4.9446348512  
>>> x = a + b * x; print(x) # 4.96124439584  
>>> x = a + b * x; print(x) # 4.97287107709  
>>> x = a + b * x; print(x) # 4.98100975396  
>>> x = a + b * x; print(x) # 4.98670682777
```

$$L = L_e + TL$$



Do you see what that is? L on the left and on the right is the same L. we're looking for the solution, where light propagation is in equilibrium. there are similar iterative methods for solving certain matrix problems.



scattering operator..

This is the propagation operator. It turns outgoing radiance into incoming radiance, which means that this operator is responsible for all the ray tracing.

compared to the recursive formulation, these operators are in reverse. before, we were ‘tracing’ importons, starting from the camera.

here, on the other hand, we work on ‘light waves’. they are propagated in epochs throughout all of the scene.

But again, you can look at the problem from two directions, and you can define adjoint versions of both formulation of the rendering equation.

$$\mathbf{L} = \mathbf{L}_e + \mathbf{T}\mathbf{L}$$



these operators are linear, so cats can cook with them,
they love cooking with linear things

$$L = L_e + TL$$

$$L - TL = L_e$$

$$(I - T)L = L_e$$

$$L = (I - T)^{-1} L_e$$

Solution operator $\rightarrow S = (I - T)^{-1}$



these operators are linear, so we can do funny things with them

I is the identity

$$S = (I - T)^{-1} = \sum_{i=0}^{\infty} T^i = I + T + T^2 + \dots$$

$$L = E + TE + T^2E + \dots$$



$$|T^k| \leq 1$$

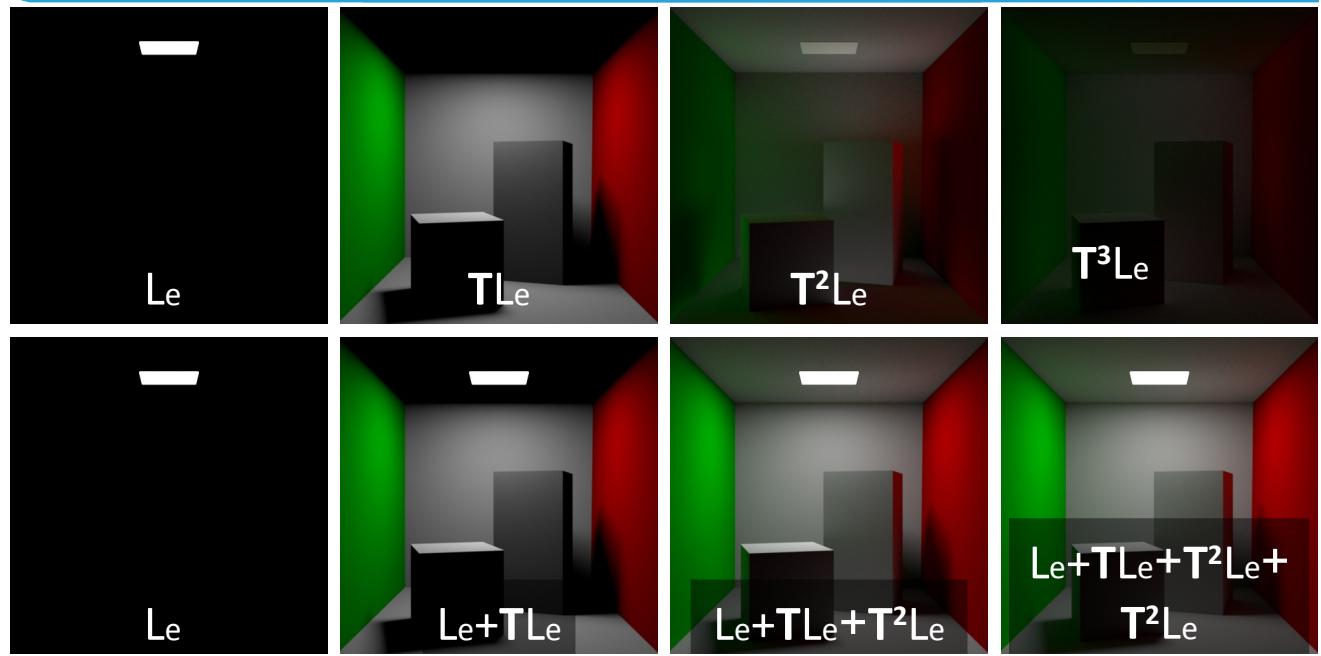
for some $k \geq 1$ and a physically valid scene model



All this works (inversion + iteration) only because ..

Let's take a look how this looks in practice

Operator Formulation (Cornell box, rendered with Nori)



see that top row becomes less bright towards the right -
> norm of $T < 1$

$$|\mathbf{T}^k| \leq 1$$

for some $k \geq 1$ and a physically valid scene model

In case of non specular materials this is even $|\mathbf{T}| < 1$

Corollary: specular materials, and in particular refractive materials, need a longer expansion

$$\mathbf{L} = \mathbf{L}_e + \mathbf{T}\mathbf{L}_e + \mathbf{T}^2\mathbf{L}_e + \dots$$



- Based on Veach 97 (PhD thesis)
- We just scratched the surface
- Veach made it quite rigorous and it's super insightful
- Do not confuse with Heckbert's notation for light paths:
 - L = light source, D = diffuse reflection, S = specular reflection, E = eye / camera
 - LDE -> denotes a direct lighting path
 - LDDE -> denotes an indirect lighting path
 - LS⁺DE -> is a caustic path. We will see later what the implications are



several slightly different things exist under this term,
some are more formal than others.

$$\mathbf{L} = \mathbf{L}_e + \mathbf{T}\mathbf{L}$$



Next: Path Integral Formulation

Adam Celarek

47

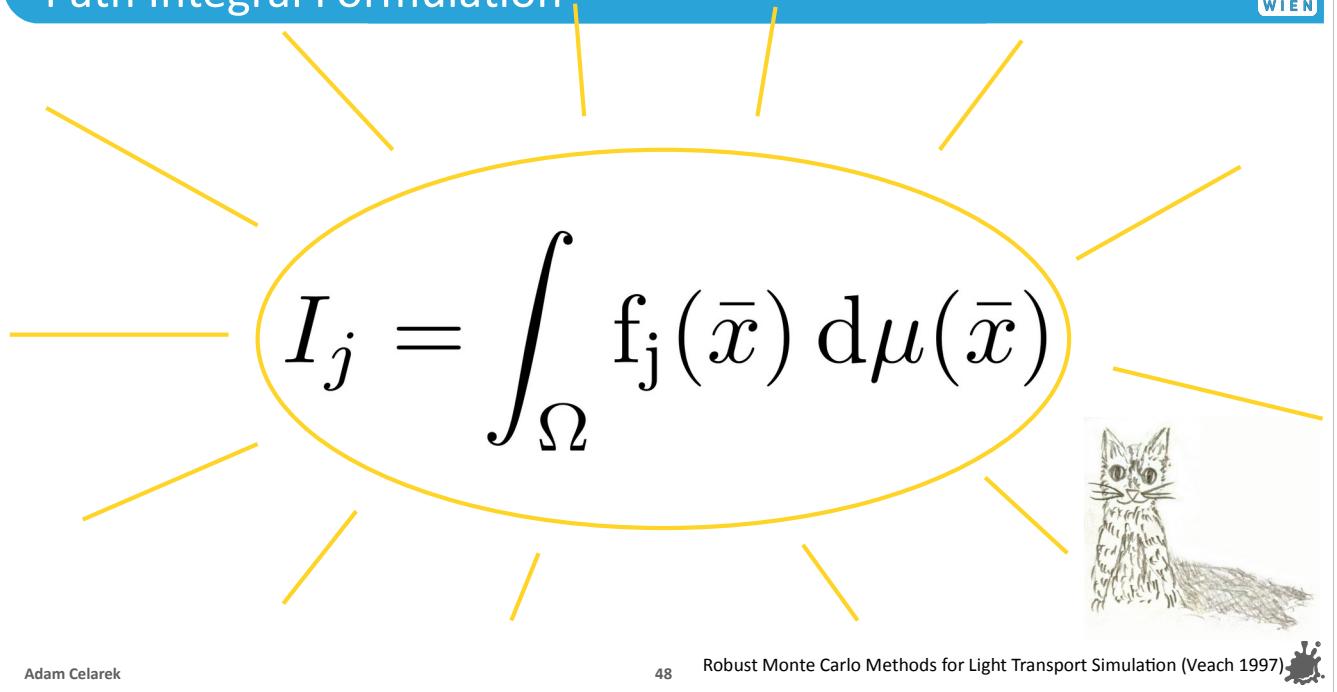
source: own work

To sum up, this is the operator formulation of light transport. We have \mathbf{L} , the light distribution in the scene, which equals \mathbf{L}_e , the emitted light and \mathbf{T} , the transport operator times \mathbf{L} . This equation reaches an equilibrium after infinite time / iterations, after which it gives us the solution for the light distribution in the scene.

It's always good to have several viewpoints on a problem, as it gives you different approaches and notations to understand and reason about a problem.

This notation is used in the radiosity method for GI, which is one of the finite element methods (FEM): The scene is discretised into small patches. Some of the patches emit light (\mathbf{L}_e). The equation is iterated several times. In every iteration we compute the outgoing light distribution for each patch. This approaches the equilibrium. We are done when the updates become small. This method was used in max Payne for instance (more details in Lehtinen's slides).

Next: Path integral formulation



Look at it with all its glory :)

Yet another reformulation of the same rendering equation.

But you probably want to know what the components are..

$$I_j = \int_{\Omega} f_j(\bar{x}) d\mu(\bar{x})$$

Measurement for a sensor element (pixel)

Measurement contribution function

Path between light source and sensor

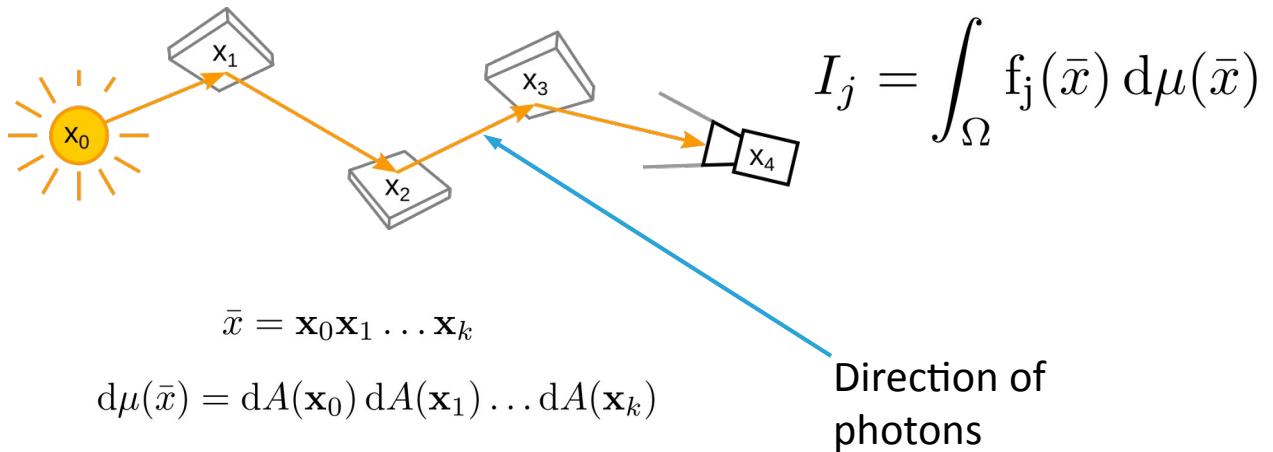
Measure on Ω

Set of all transport paths (all lengths)



This is just an overview, we will look into each component in the next slides.

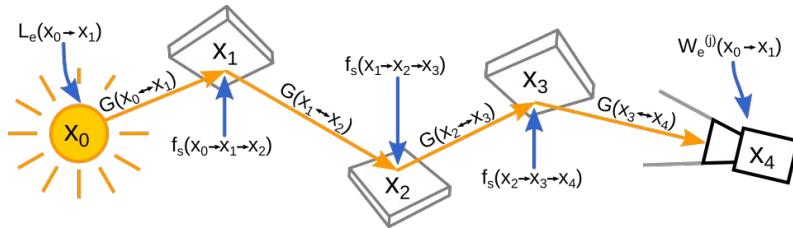
The result I_j is the measurement (that is brightness/colour) for a certain sensor pixel. The pixel is indicated by the j . We integrate over the set of all possible transport paths of all lengths. These paths are written as \bar{x} . The measure is the differential that is needed for integration. And finally, f_j is the measurement contribution function.



Here we see an example of a path. It connects the light source over 3 vertices to the camera. The light source and the camera are also vertices. We can describe a path as a list of vertices. And as said, we are integrating over the set of all possible paths. The shortest possible path would be a direct connection between the light source and the camera, consisting of just 2 vertices. The longest possible path would be of infinite length, so it's actually not possible in the computer^. The measure is a bit abstract for now. Think about it like it is responsible for generating the samples, and their pdf, which are necessary for Monte Carlo integration. Therefore it depends on the path. It can be expanded to a product of area measures (or area differentials), one area differential for each vertex. It will become clearer later.

Let's now look at the measurement contribution function f_j

Path Integral Formulation



$$f_j(\bar{x}) = L_e(x_0 \rightarrow x_1)$$

$$G(x_0 \leftrightarrow x_1) f_s(x_0 \rightarrow x_1 \rightarrow x_2)$$

$$G(x_1 \leftrightarrow x_2) f_s(x_1 \rightarrow x_2 \rightarrow x_3)$$

...

$$G(x_{k-1} \leftrightarrow x_k) W_e^{(j)}(x_{k-1} \rightarrow x_k)$$

$$I_j = \int_{\Omega} f_j(\bar{x}) d\mu(\bar{x})$$

$$\bar{x} = x_0 x_1 \dots x_k$$

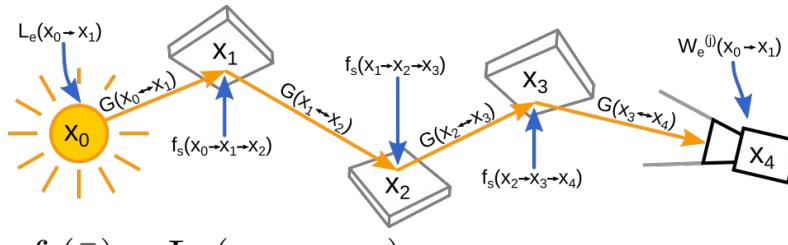
$$d\mu(\bar{x}) = dA(x_0) dA(x_1) \dots dA(x_k)$$



f_j is a product of several factors, the light emission **L_e**, which is simply the brightness of the light at position **x₀**, geometry factors between each pair of vertices **G**..

And, actually, lets look into these geometry factors, they are interesting..

Path Integral Formulation



$$f_j(\bar{x}) = L_e(x_0 \rightarrow x_1)$$

$$I_j = \int_{\Omega} f_j(\bar{x}) d\mu(\bar{x})$$

$$\bar{x} = x_0 x_1 \dots x_k$$

$$d\mu(\bar{x}) = dA(x_0) dA(x_1) \dots dA(x_k)$$

$$G(\mathbf{x} \leftrightarrow \mathbf{x}') = V(\mathbf{x} \leftrightarrow \mathbf{x}') \frac{|\cos(\theta_o) \cos(\theta'_i)|}{\|\mathbf{x} - \mathbf{x}'\|^2}$$

$$G(x_{k-1} \leftrightarrow x_k) \propto e^{-\sigma(x_{k-1} \rightarrow x_k)}$$



So G consists of a visibility term V , the sending and receiving cosine, and the distance between the vertices squared.

Huh..

G .. geometry (visibility, cos0, cos1, distance)

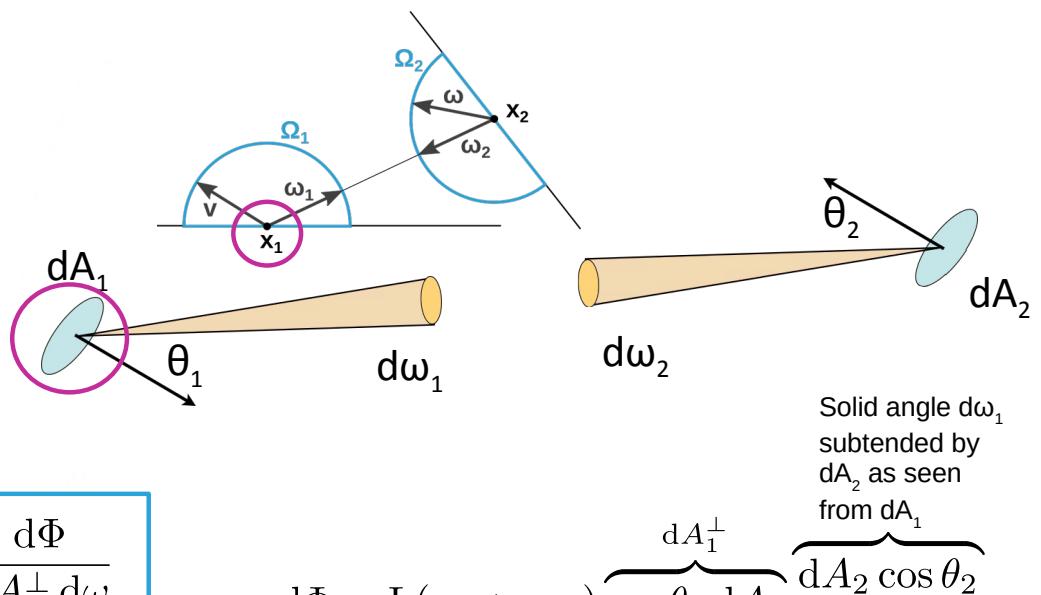
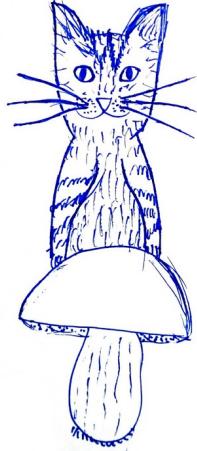
x and x' .. neighbouring vertices in the path

V .. visibility (0 or 1)

θ_o .. angle between **normal at x** and $x'-x$

θ'_i .. angle between **normal at x'** and $x-x'$

Before in the Recursive Formulation



$$L = \frac{d\Phi}{dA^\perp d\omega}$$

$$d\Phi = L(x_1 \leftarrow \omega_1) \overbrace{\cos \theta_1 dA_1}^{dA_1^\perp} \overbrace{\frac{dA_2 \cos \theta_2}{r^2}}^{d\omega_2}$$



Compare those geometry factors with what we had before in physics, when we were looking at differential flux: computing it requires the cosine on both sides and distance squared!



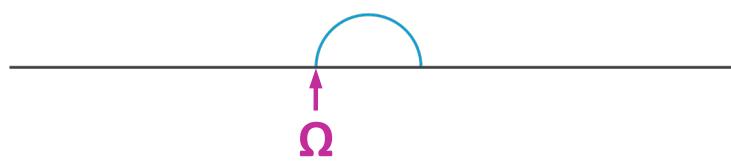
$$L(x_1 \rightarrow v) = E(x_1 \rightarrow v) + \int_{\Omega_1} f_r(x_1, \omega_1 \rightarrow v) L(x_1 \leftarrow \omega_1) \cos(\theta_x) d\omega_1$$

$L(x_1 \leftarrow \omega_1) = L(x_2 \rightarrow \omega_2)$!

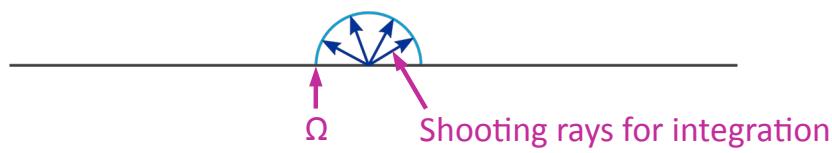


But it is nowhere to be found in the recursive formulation!

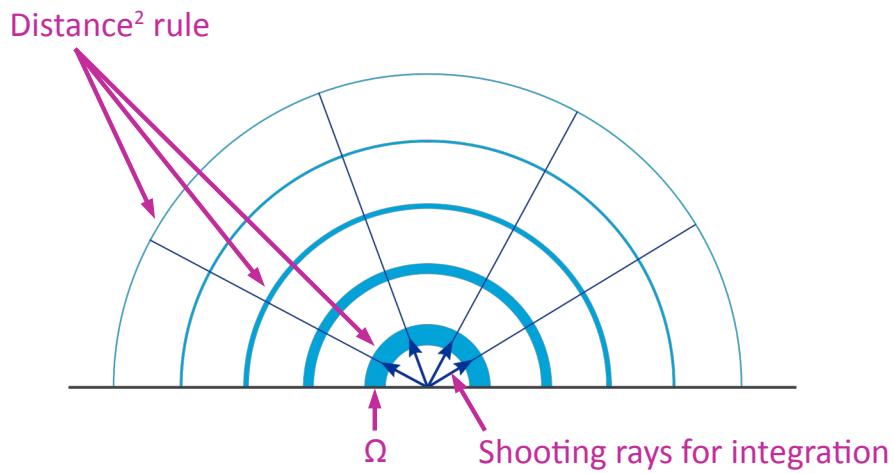
Why is that? Let's look into it!



In the recursive formulation (and also when we compute direct lighting using hemisphere sampling) we integrate over the hemisphere Ω .

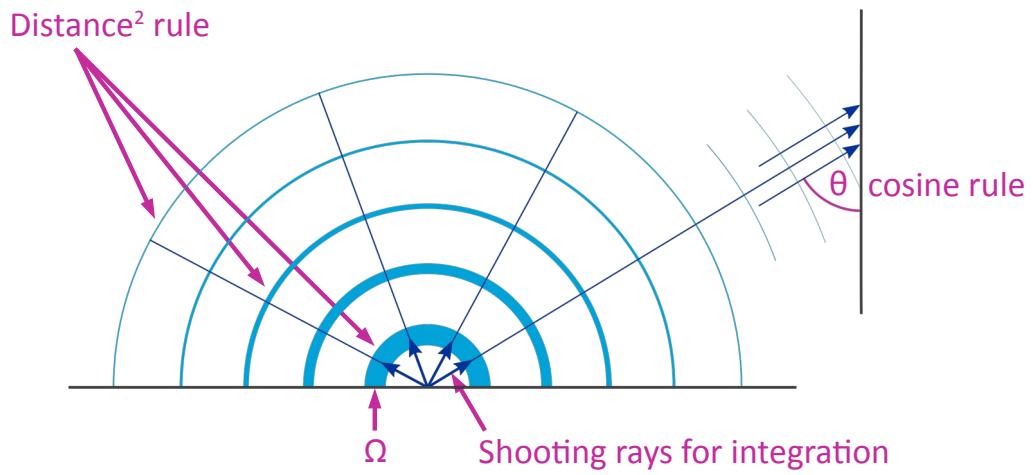


We do that by shooting ‘virtual’ rays (they are virtual, because we are still in math mode). They have to do with differentials and all this integration magic.. So we have an infinite amount of these rays, and the density of rays is continuous.

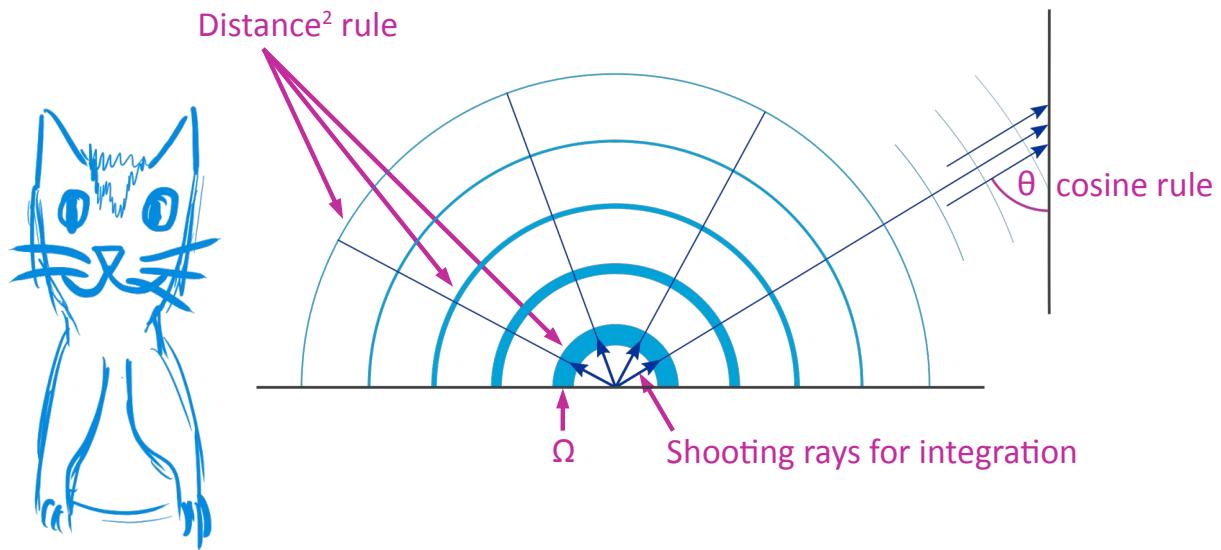


I hope the visualisation is ok, the ‘density’ of these rays is reduced by the same ‘distance squared’ law as photon density when emitting light (which I explained in the second lecture).

Before in the Recursive Formulation



When the virtual rays hit a tilted surface, the cosine rule comes into effect. Again, for the same reason as with photons. So the density is further reduced by the tiled surface..



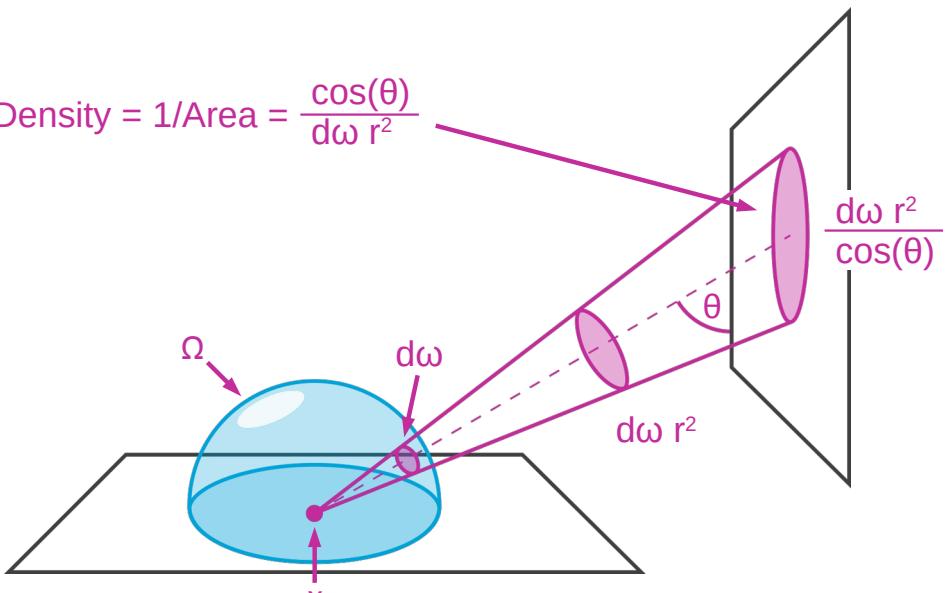
And this means, that the missing cosine and distance squared is actually embedded in ray casting!

This also fits together well with what I said about the adjoint operation of tracing photons – tracing importons. Just as photons follow the laws of distance square and cosine, importons also do.

Before in the Recursive Formulation



$$\text{Density} = 1/\text{Area} = \frac{\cos(\theta)}{d\omega r^2}$$



Adam Celarek

60



Another way to look at it is to extend the solid angle $d\omega$ as a cone. That way it ‘finds’ exactly what would be projected on the unit hemisphere. The tip is at the centre of the hemisphere (x) and at a unit distance it has a cross-section of $d\omega$. While it extends along the ray, the area becomes larger at a rate of distance squared, and when projecting it onto a tilted surface, it becomes larger by a factor of $1/\cos(\theta)$.

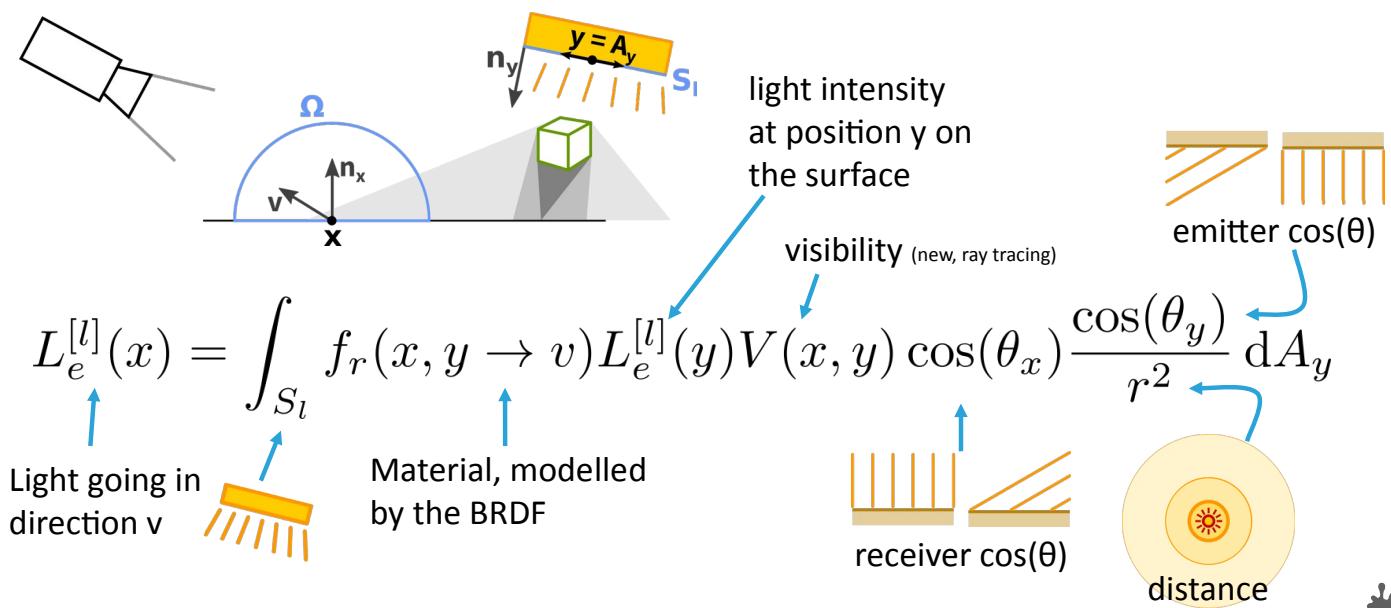
So the area at the destination is $d\omega r^2 / \cos(\theta)$. When we compute the probability density, we compute 1 over the area, which means, that we arrive at $\cos(\theta)/(r^2 d\omega)$ here again.

- Just as we can ‘map’ a sample from a surface to the hemisphere and compute its probability in the domain of the hemisphere, we can do the same thing vice versa!

- We are trying to show you the same thing from different angles, for some people one angle might work better than the other. And we hope that eventually it will make sense to you all :)

Before in the Recursive Formulation

Soft shadows (surface sampling)



Adam Celarek

61

Ok, one more:

- This page is exactly what we had in the lecture about light, when we made the change of variables and integrated over the surface of the light (hence the dA).

I'll quickly go over the factors:

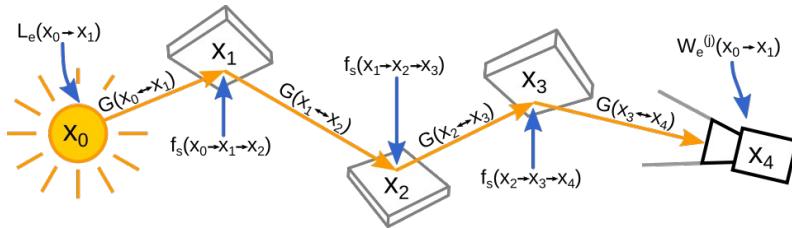
Outgoing light at x , integral over the surface, material, light emission L_e , visibility, receiver \cos , emitter \cos and distance squared.

- **And hey, actually a very similar equation could be used as a second variant of the recursive formulation. We would just have to add light emittance before the integral, and then we could plug $L_e(x)$ (exitant radiance) right into $L_e(y)$ (also exitant radiance).**

We again see a similar situation (two cosines and distance squared), even the visibility term is there. That looks pretty much like the G term.

- Ok, back to the path integral formulation, you don't know all components yet..

Path Integral Formulation



$$f_j(\bar{x}) = L_e(x_0 \rightarrow x_1)$$

$$G(x_0 \leftrightarrow x_1) f_s(x_0 \rightarrow x_1 \rightarrow x_2)$$

$$G(x_1 \leftrightarrow x_2) f_s(x_1 \rightarrow x_2 \rightarrow x_3)$$

...

$$G(x_{k-1} \leftrightarrow x_k) W_e^{(j)}(x_{k-1} \rightarrow x_k)$$

$$I_j = \int_{\Omega} f_j(\bar{x}) d\mu(\bar{x})$$

$$\bar{x} = x_0 x_1 \dots x_k$$

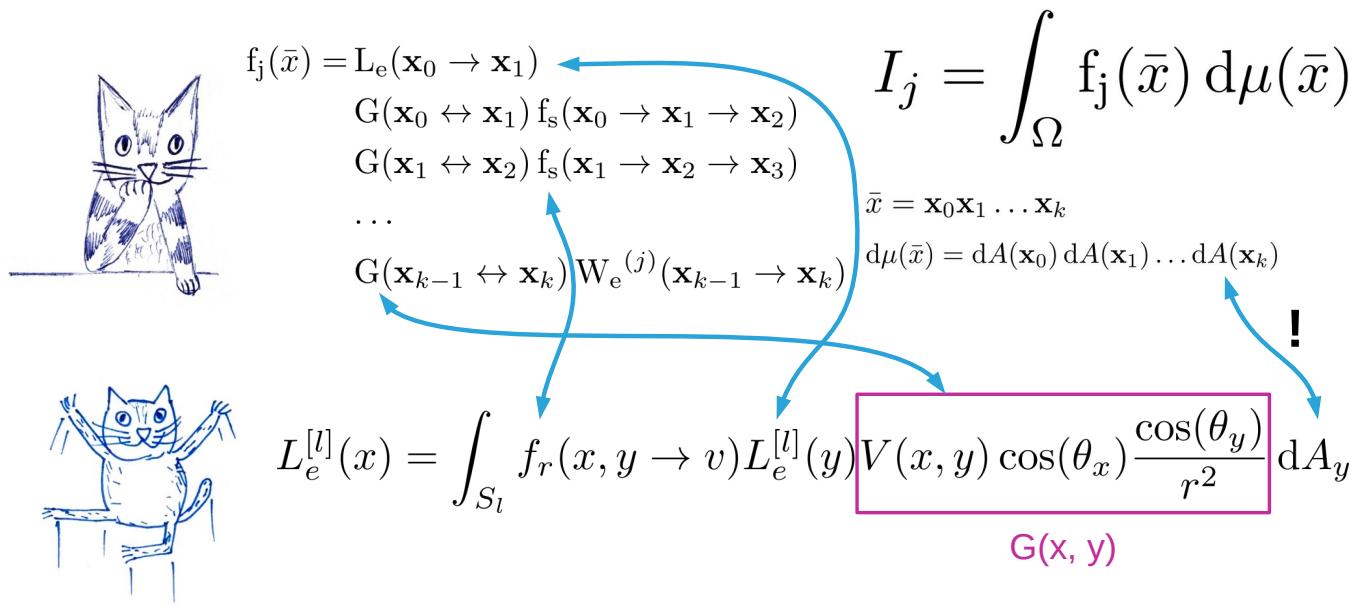
$$d\mu(\bar{x}) = dA(x_0) dA(x_1) \dots dA(x_k)$$



Again from the start..

- **f_j**, the measurement contribution function, is a product of the light emission **L_e**,
- geometry factors between each pair of vertices **G** (which we heard about at length just now), the scattering factors **f_s** for each inner vertex (reflection point), which model the material,
- and finally the importance emission from the camera **W_e**. Remember that we said that we can look at light transport from 2 different directions, either photons emitted from the light source going to the camera, or importants emitted from the camera going towards the light source. I'm not aware of a case where W_e is not 1 (tell me if you know :), but we keep it for the symmetry.

Comparison with Surface Sampling



Waaaaait, compare path integral on top with the direct light surface equation on the bottom. material, geometry term, differential for the surface of the light – the last one is the measure $dA(x)$!!!

Hah, the measure in the path integral formulation is a product of $dAs \rightarrow$
The whole integral is integrating over all surfaces at once! Similar to dA itself, which integrates over du and dv (uv coordinates), or $d\omega$ in the hemisphere, which integrates over $d\phi$ and $d\theta$ at the same time.

So the path integral formulation is really just an integral which integrates over all surfaces at the same time.

$$\bar{x} = \mathbf{x}_0 \mathbf{x}_1 \dots \mathbf{x}_k$$

$$d\mu(\bar{x}) = dA(\mathbf{x}_0) dA(\mathbf{x}_1) \dots dA(\mathbf{x}_k)$$

$$f_j(\bar{x}) = L_e(\mathbf{x}_0 \rightarrow \mathbf{x}_1)$$

$$G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) f_s(\mathbf{x}_0 \rightarrow \mathbf{x}_1 \rightarrow \mathbf{x}_2)$$

$$G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) f_s(\mathbf{x}_1 \rightarrow \mathbf{x}_2 \rightarrow \mathbf{x}_3)$$

...

$$G(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k) W_e^{(j)}(\mathbf{x}_{k-1} \rightarrow \mathbf{x}_k)$$

$$I_j = \int_{\Omega} f_j(\bar{x}) d\mu(\bar{x})$$

$$= \int_{\Omega_0} f_j(\bar{x}) d\mu(\bar{x}) + \int_{\Omega_1} f_j(\bar{x}) d\mu(\bar{x}) + \dots + \int_{\Omega_{\infty}} f_j(\bar{x}) d\mu(\bar{x})$$

L_e

$T L_e$

$T^\infty L_e$

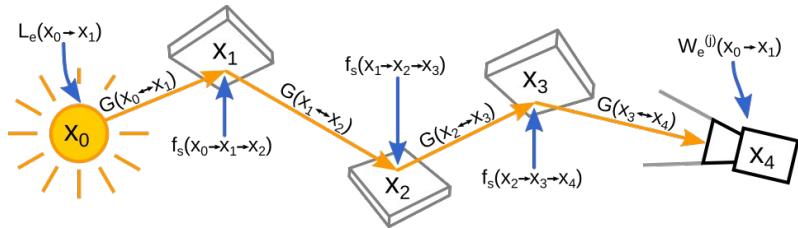


look into the integration domain

domains and operator notation not the same.

like we just learned I_j is what is rendered on the screen, i.e. contains camera filter factors. Operator notation is light in the scene.

Path Integral Formulation



$$\begin{aligned}
 I_j &= \int_{\Omega} f_j(\bar{x}) d\mu(\bar{x}) \\
 &= \int_{\Omega_0} f_j(\bar{x}) d\mu(\bar{x}) + \int_{\Omega_1} f_j(\bar{x}) d\mu(\bar{x}) + \dots + \int_{\Omega_{\infty}} f_j(\bar{x}) d\mu(\bar{x})
 \end{aligned}$$



agnostic to how the path was generated. can generate starting from the camera (path tracing, recursive formulation), or at the light. depending on that, the probabilities of generating that path are different -> MIS, BDPT. Also MLT (correlated samples, where each sample is a path).

- Cleaner notation, easier to handle with more complex algorithms than path tracing (next lecture)
 - Framework for computing probability densities on paths
 - MIS across path generation directions (Bidirectional path tracing)
 - Metropolis light transport (correlated samples / paths)
- Based on Veach 97 (PhD thesis)
- We just scratched the surface
- Veach made it quite rigorous and it's super insightful



$$I_j = \int_{\Omega} f_j(\bar{x}) d\mu(\bar{x})$$

Next Lecture: Path Tracing

Reading: Eric Veach's PhD Thesis

Adam Celarek

67

source: own work