Lab #4 – Path tracing (part 2)

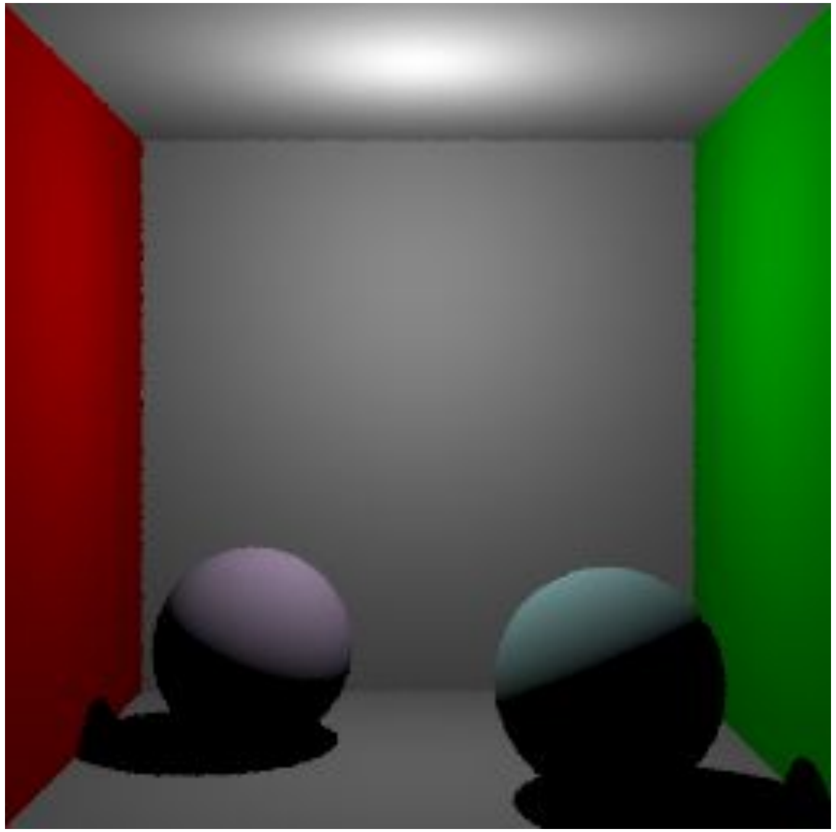
Informática Gráfica

Adolfo Muñoz - Julio Marco
Pablo Luesia - J. Daniel Subías – Óscar Pueyo

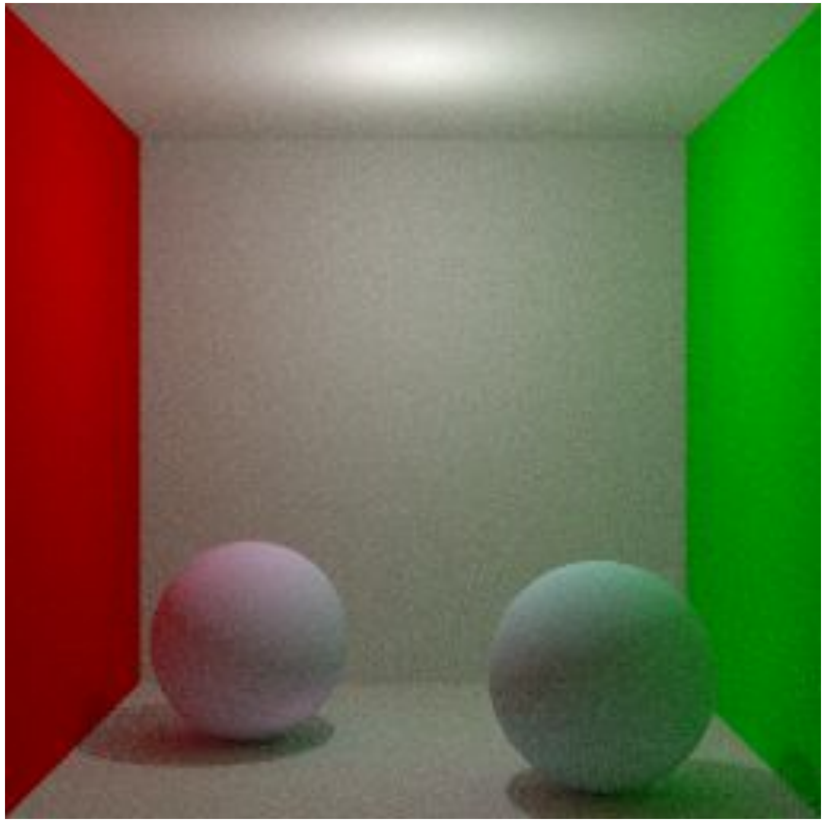
Graphics and Imaging Lab

Before we begin…

- Today: calculate direct illumination **and** indirect illumination



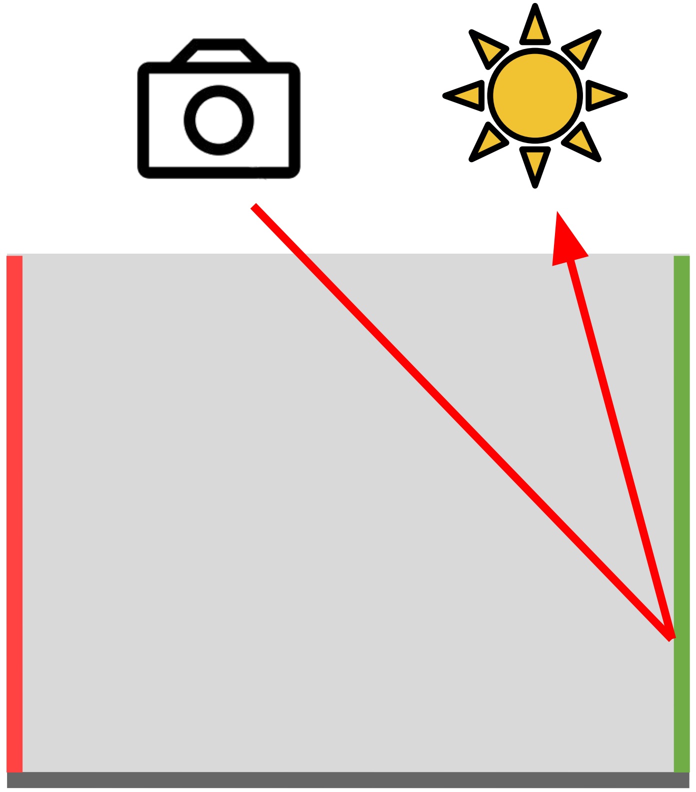
Previous session (direct light)



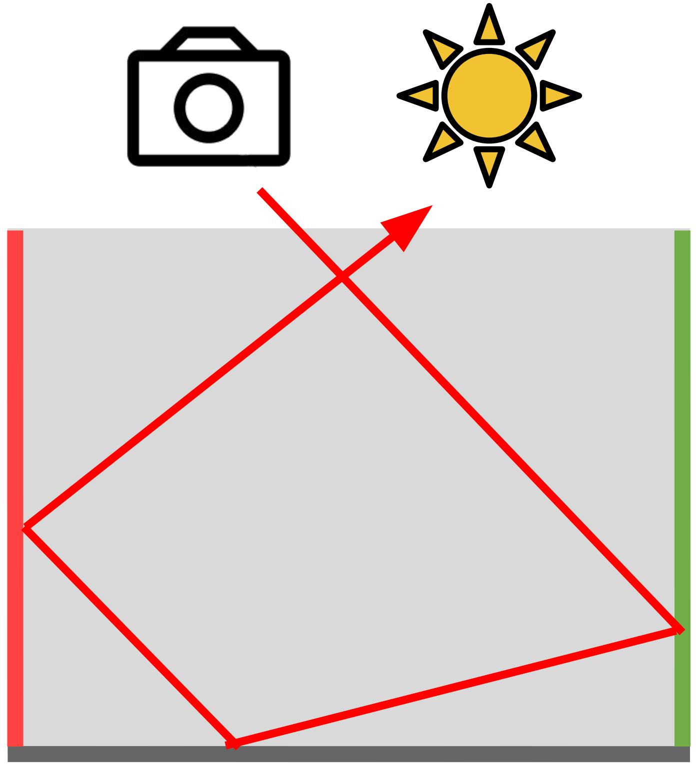
Full path tracing (+ indirect light)

Before we begin…

- Today: calculate direct illumination **and** indirect illumination



Previous session (direct light)



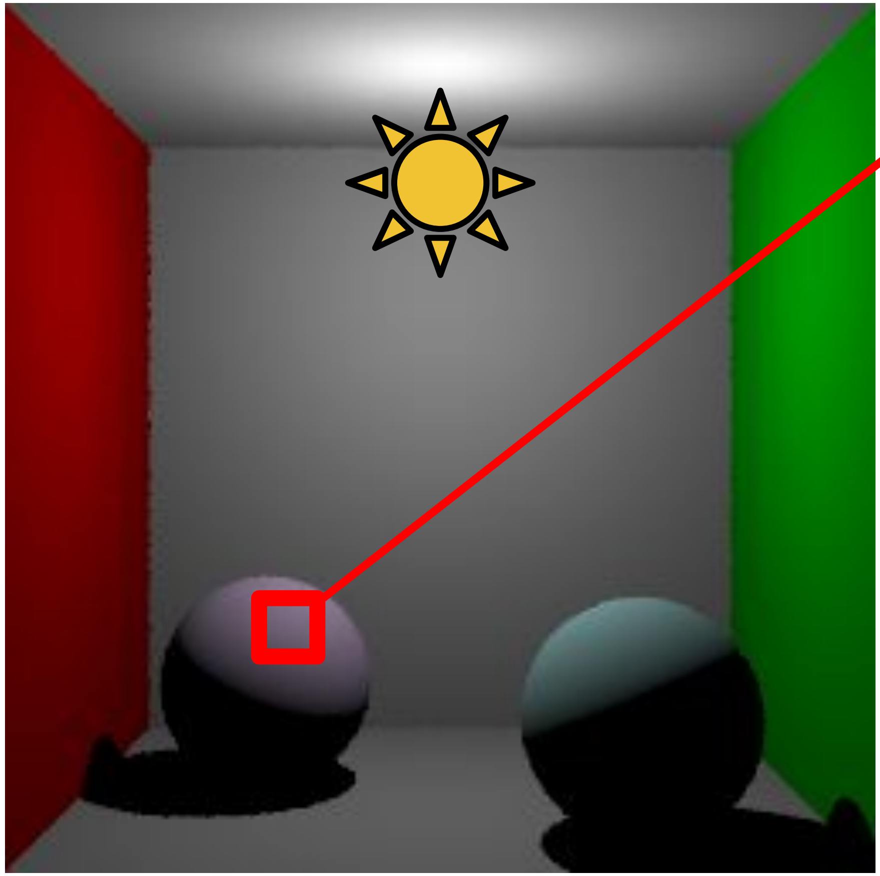
Full path tracing (+ indirect light)

Before we begin…

Graphics and Imaging Lab

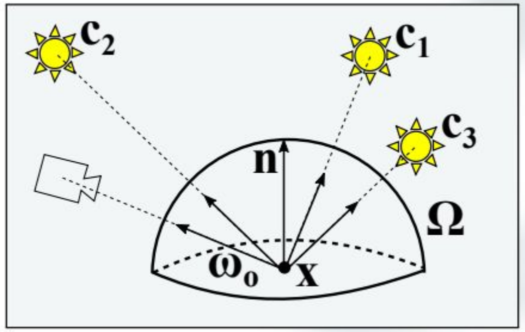
- Warning: there is a lot of theory on these slides, handle with care
    - If you want to go straight to the point/programming, go to the end
- Lab 4 (path tracing) **is the first submitted work**
    - Recommended deadline: November 13th
    - Moodle: January 11th
    - You will use most of today's code for Lab 5 (photon mapping) too
- Remember: Final work is 80% of the final grade

Previously: only direct illumination



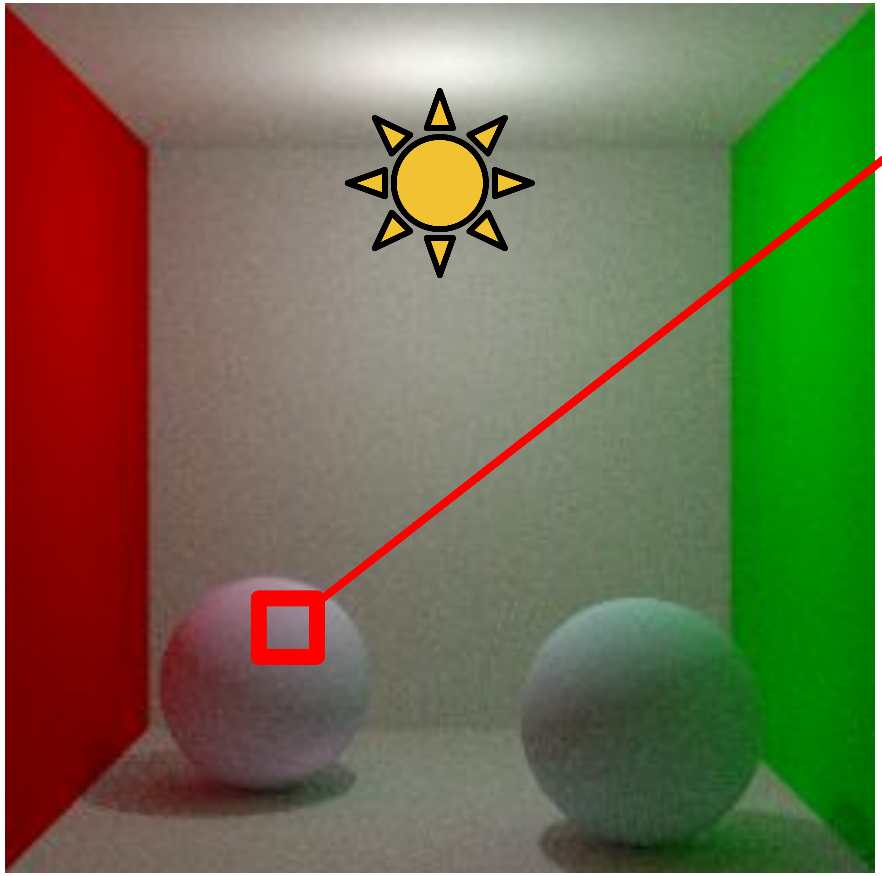
$$L_o(\mathbf{x},\omega_\mathbf{o}) = \cancel{L_e(\mathbf{x},\omega_\mathbf{o})} + \boxed{\int_\Omega L_i(\mathbf{x},\omega_\mathbf{i}) f_r(\mathbf{x},\omega_\mathbf{i},\omega_\mathbf{o}) |\mathbf{n}\cdot\omega_\mathbf{i}| d\omega_\mathbf{i}}$$

But just the direct light



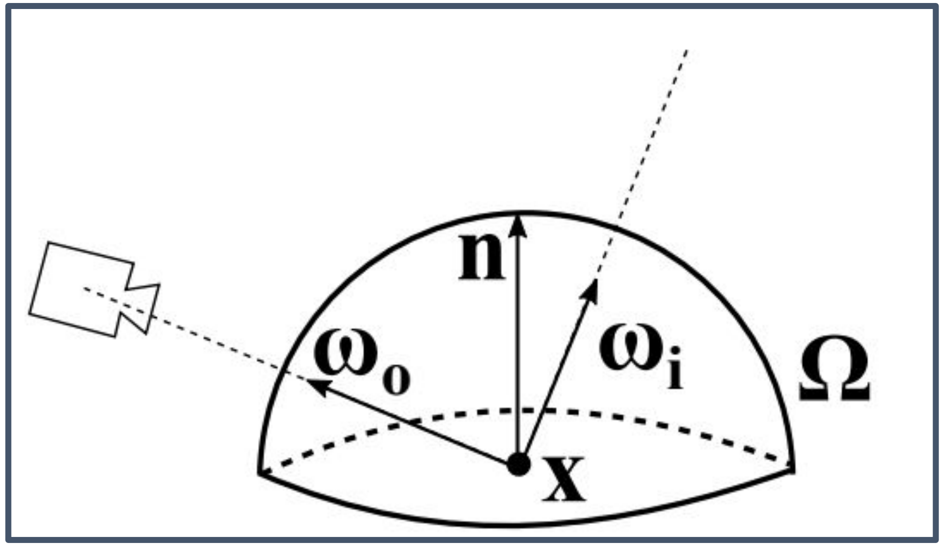
$$L_o(\mathbf{x},\omega_\mathbf{o}) = \boxed{\sum_{i=1}^{n} \frac{p_i}{|\mathbf{c}_\mathbf{i}-\mathbf{x}|^2} f_r\left(\mathbf{x}, \frac{\mathbf{c}_\mathbf{i}-\mathbf{x}}{|\mathbf{c}_\mathbf{i}-\mathbf{x}|}, \omega_\mathbf{o}\right) \left|\mathbf{n}\cdot\frac{\mathbf{c}_\mathbf{i}-\mathbf{x}}{|\mathbf{c}_\mathbf{i}-\mathbf{x}|}\right|}$$

Which color do we fill each pixel with?



$$L_o(\mathbf{x},\omega_o) = \cancel{L_e(\mathbf{x},\omega_o)} + \boxed{\int_\Omega L_i(\mathbf{x},\omega_i) f_r(\mathbf{x},\omega_i,\omega_o) |\mathbf{n}\cdot\omega_i| d\omega_i}$$

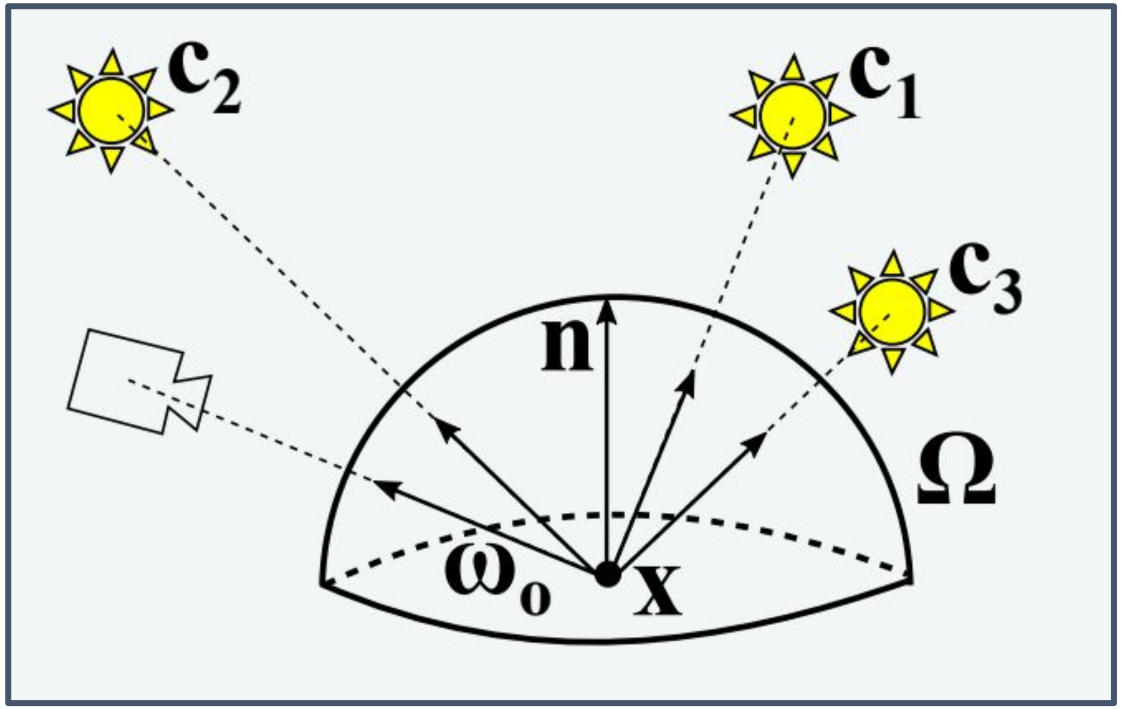
The full integral



Computing the path integral

- How do we compute $\int_{\Omega} L_i(\mathbf{x},\omega_\mathbf{i}) f_r(\mathbf{x},\omega_\mathbf{i},\omega_\mathbf{o}) |\mathbf{n}\cdot\omega_\mathbf{i}| d\omega_\mathbf{i}$?

- On the previous session (direct light):



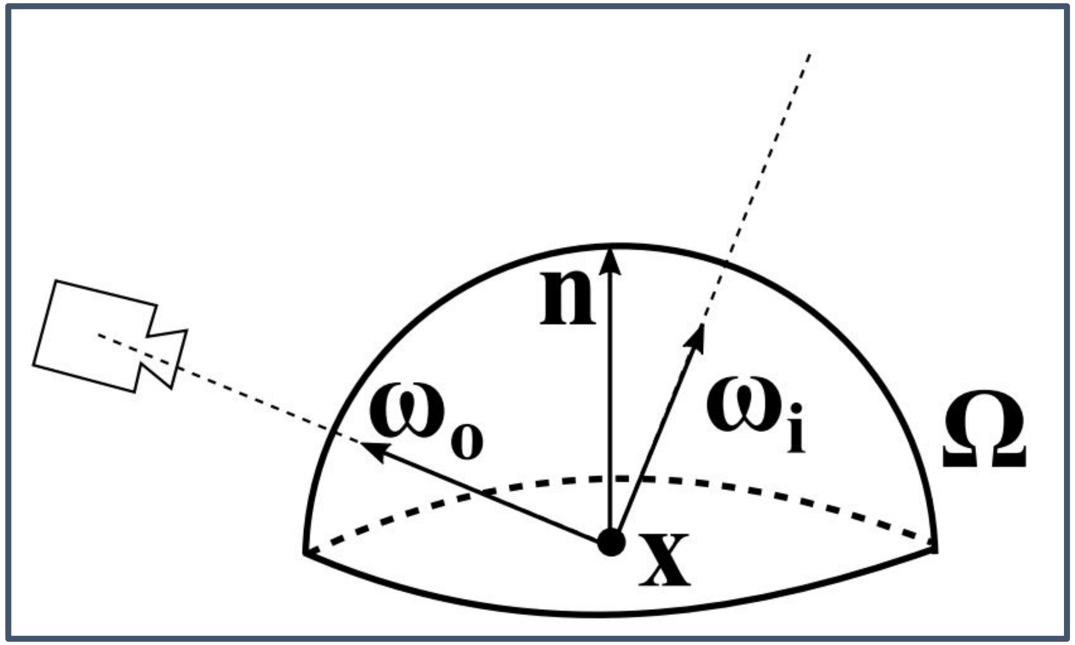
$$= \sum_{i=1}^{n} \frac{p_i}{|\mathbf{c_i}-\mathbf{x}|^2} f_r\left(\mathbf{x}, \frac{\mathbf{c_i}-\mathbf{x}}{|\mathbf{c_i}-\mathbf{x}|}, \omega_\mathbf{o}\right) \left|\mathbf{n}\cdot\frac{\mathbf{c_i}-\mathbf{x}}{|\mathbf{c_i}-\mathbf{x}|}\right|$$

- Paths have exactly one bounce
- Closed-form solution

Computing the path integral

- How do we compute $\int_{\Omega} L_i(\mathbf{x}, \omega_\mathbf{i}) f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\mathbf{n} \cdot \omega_\mathbf{i}| d\omega_\mathbf{i}$?

- Today (direct light + indirect light):



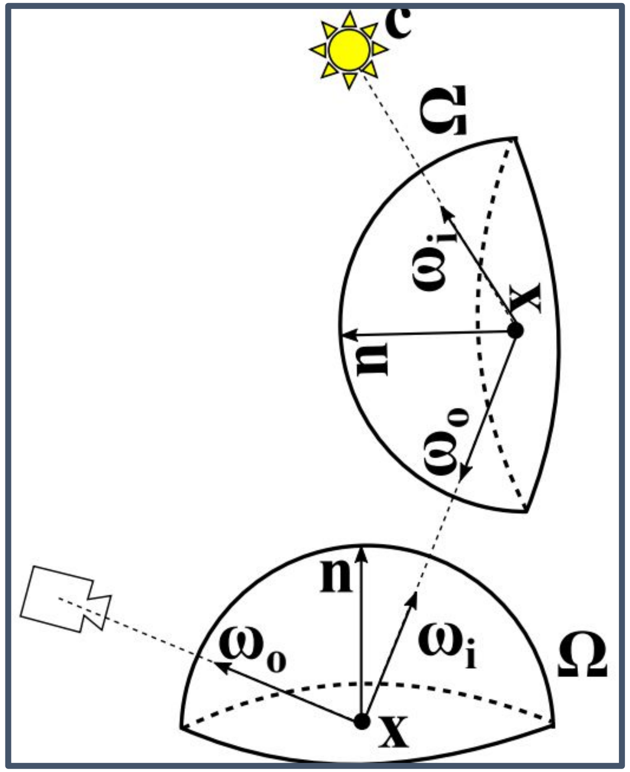
$$= \int_{\Omega_1} L_i(\mathbf{x}_1, \omega_{\mathbf{i}1}) f_r(\mathbf{x}_1, \omega_{\mathbf{i}1}, \omega_{\mathbf{o}1}) |\mathbf{n}_1 \cdot \omega_{\mathbf{i}1}| d\omega_{\mathbf{i}1}$$

- Integrate over the whole hemisphere

Computing the path integral

- How do we compute $\int_{\Omega} L_i(\mathbf{x}, \omega_\mathrm{i}) f_r(\mathbf{x}, \omega_\mathrm{i}, \omega_\mathrm{o}) |\mathbf{n} \cdot \omega_\mathrm{i}| d\omega_\mathrm{i}$?

- Today (direct light + indirect light):



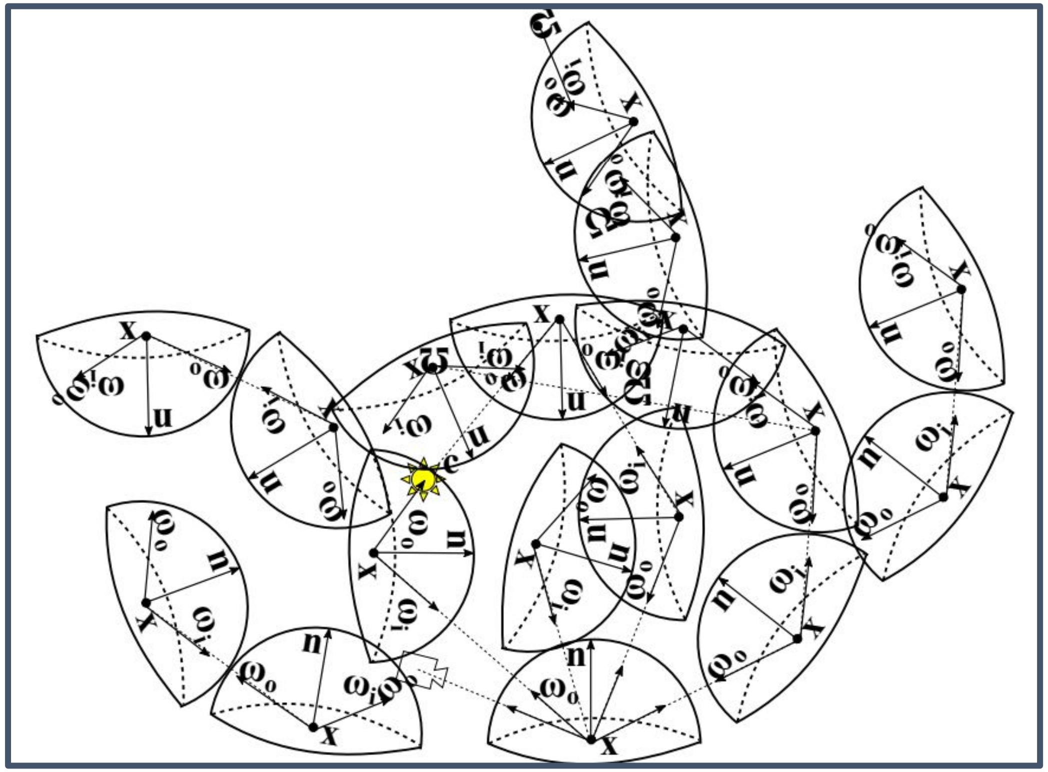
$$= \int_{\Omega_1} \left(\int_{\Omega_2} L_i(\mathbf{x}_2, \omega_{\mathrm{i2}}) f_r(\mathbf{x}_2, \omega_{\mathrm{i2}}, -\omega_{\mathrm{i1}}) |\mathbf{n}_2 \cdot \omega_{\mathrm{i2}}| d\omega_{\mathrm{i1}} \right) f_r(\mathbf{x}_1, -\omega_{\mathrm{i1}}, \omega_{\mathrm{o1}}) |\mathbf{n}_1 \cdot \omega_{\mathrm{i1}}| d\omega_{\mathrm{i1}}$$

- Integrate over the whole hemisphere^2

Computing the path integral

- How do we compute $\int_{\Omega} L_i(\mathbf{x}, \omega_\mathbf{i}) f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\mathbf{n} \cdot \omega_\mathbf{i}| d\omega_\mathbf{i}$?

- Today (direct light + indirect light):



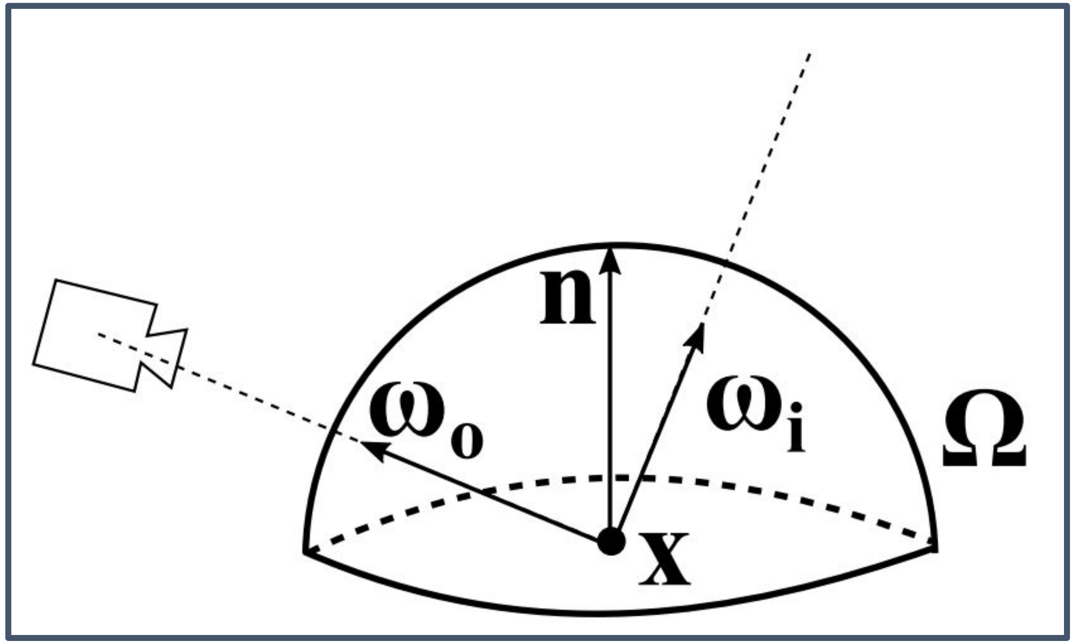
$$= \int_{\Omega_1} \cdots \int_{\Omega_N} L_i(\mathbf{x_N}, \omega_\mathbf{N2}) \left(\prod_{k=2}^{N} f_r \left(\mathbf{x_k}, \omega_\mathbf{il}, -\omega_\mathbf{i(k-1)} \right) \right) f_r(\mathbf{x_1}, \omega_\mathbf{i1}, \omega_\mathbf{o1})$$

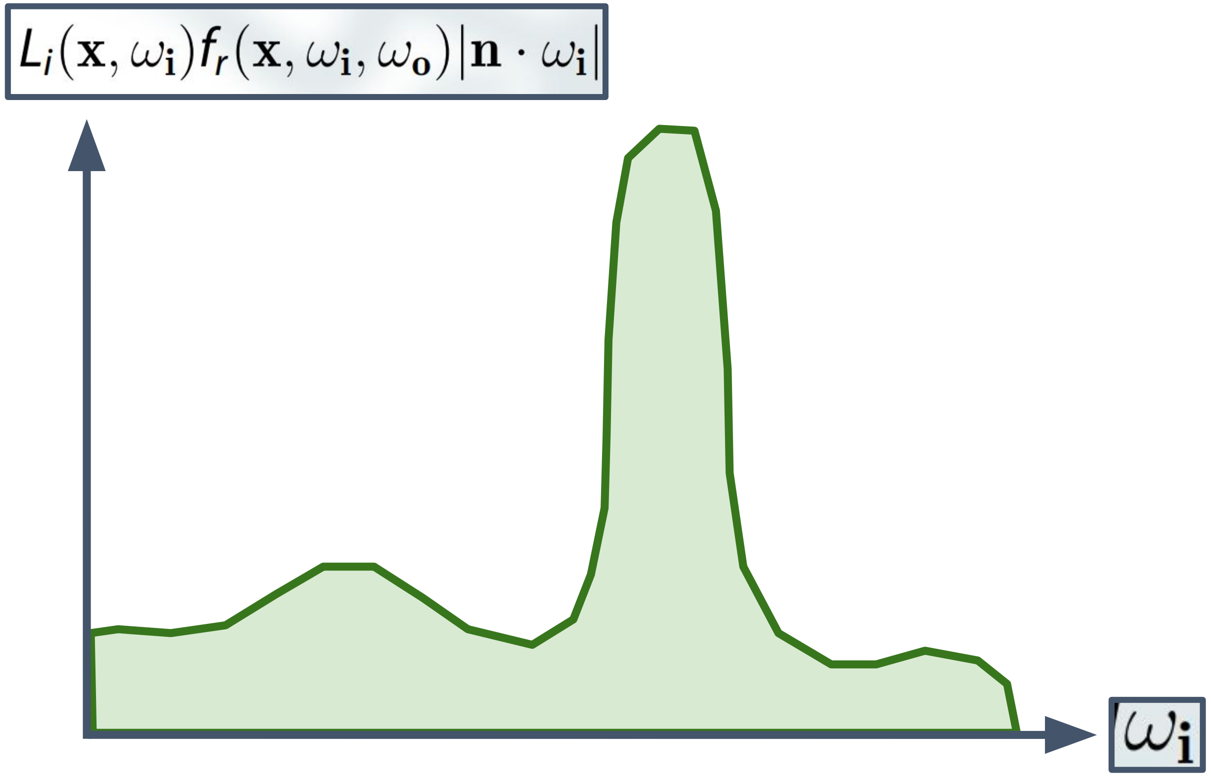
$$\left(\prod_{k=1}^{N} |\mathbf{n_k} \cdot \omega_\mathbf{ik}| \right) d\omega_\mathbf{iN} \cdots d\omega_\mathbf{i1}$$

- Paths can have 1..N bounces
- How do we solve this integral?

Approximating one integral

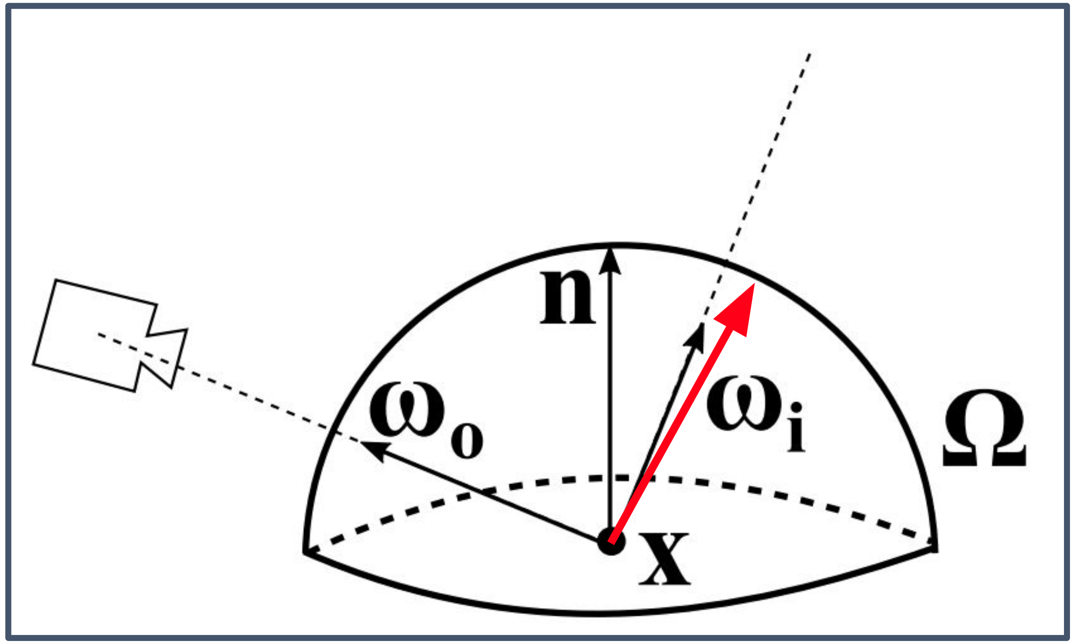
- How do we compute $\int_\Omega L_i(\mathbf{x}, \omega_\mathbf{i}) f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o}) |\mathbf{n} \cdot \omega_\mathbf{i}| d\omega_\mathbf{i}$?

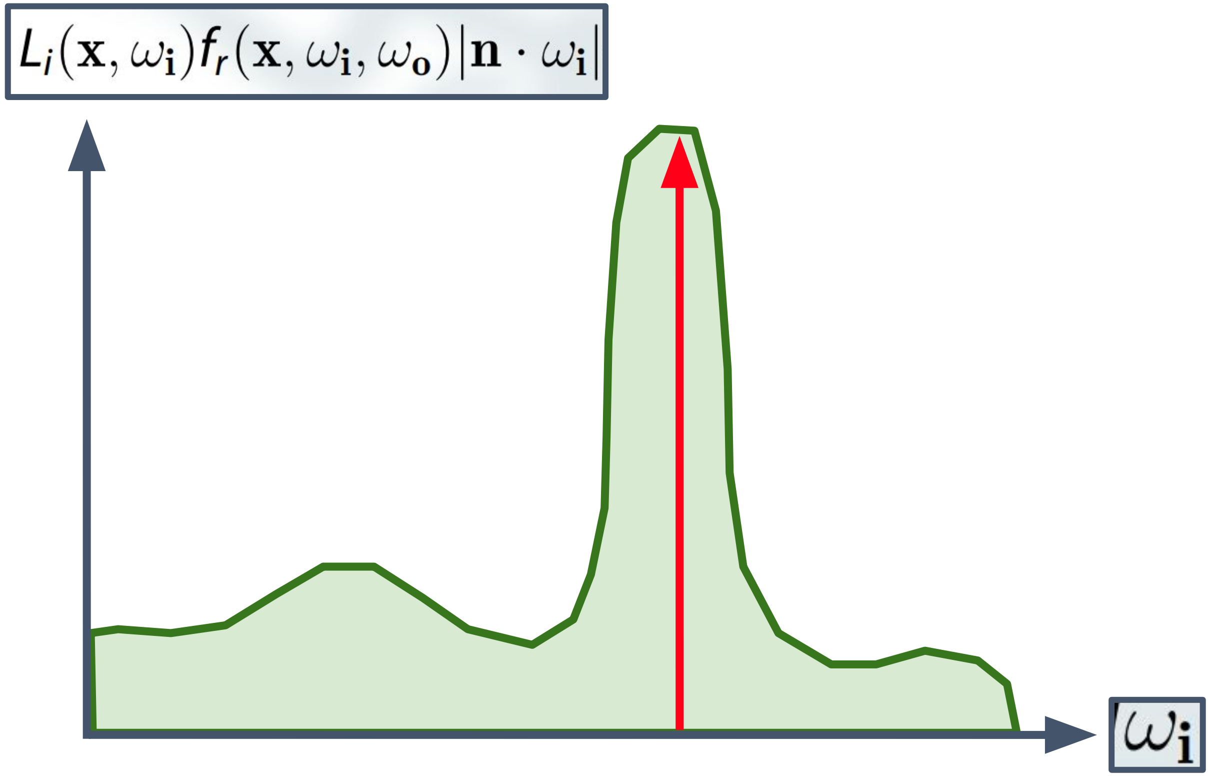




Approximating one integral

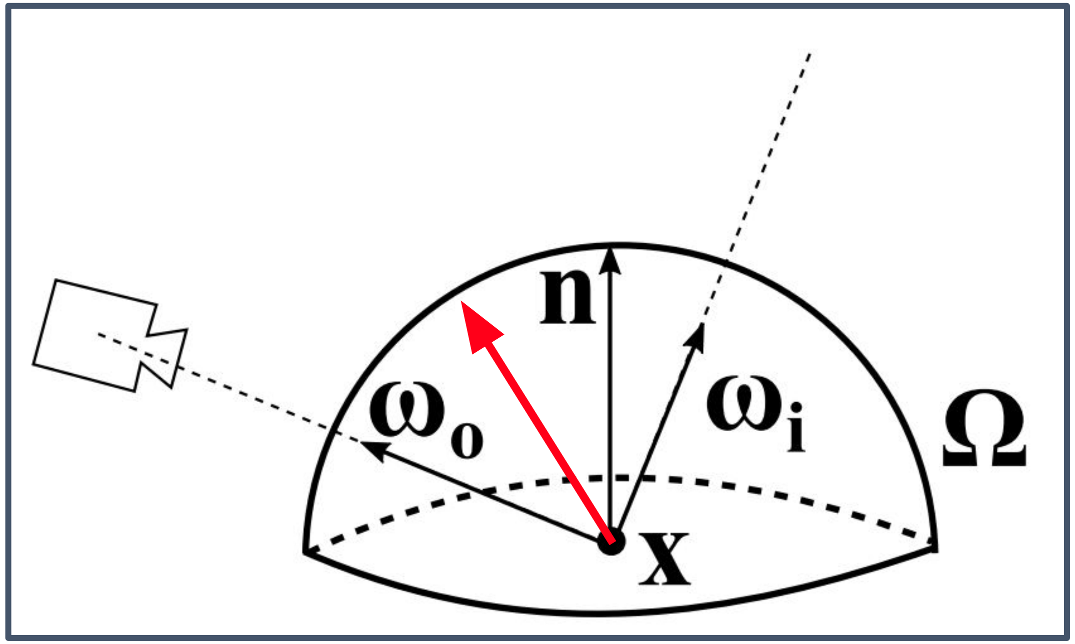
- How do we compute $\boxed{\int_\Omega L_i(\mathbf{x}, \omega_\mathrm{i}) f_r(\mathbf{x}, \omega_\mathrm{i}, \omega_\mathrm{o}) |\mathbf{n} \cdot \omega_\mathrm{i}| d\omega_\mathrm{i}}$?

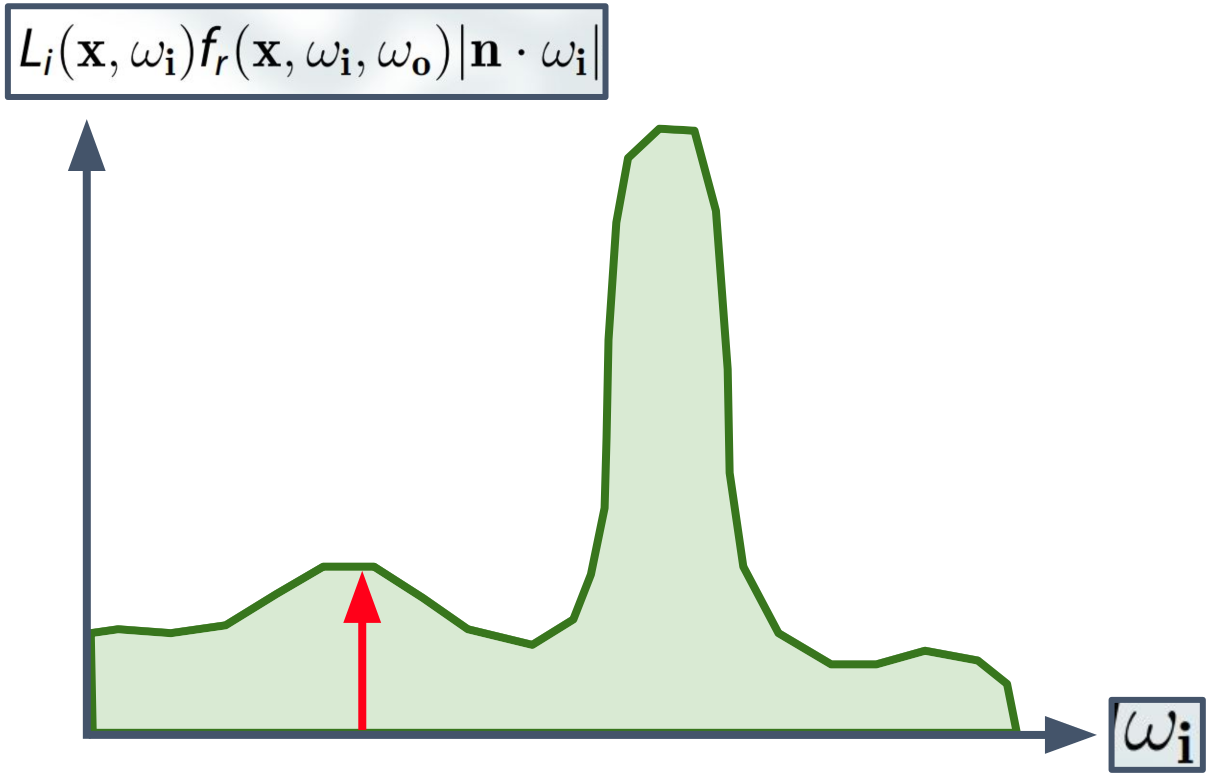




Approximating one integral

- How do we compute $\int_{\Omega} L_i(\mathbf{x}, \omega_\mathrm{i}) f_r(\mathbf{x}, \omega_\mathrm{i}, \omega_\mathrm{o}) |\mathbf{n} \cdot \omega_\mathrm{i}| d\omega_\mathrm{i}$?

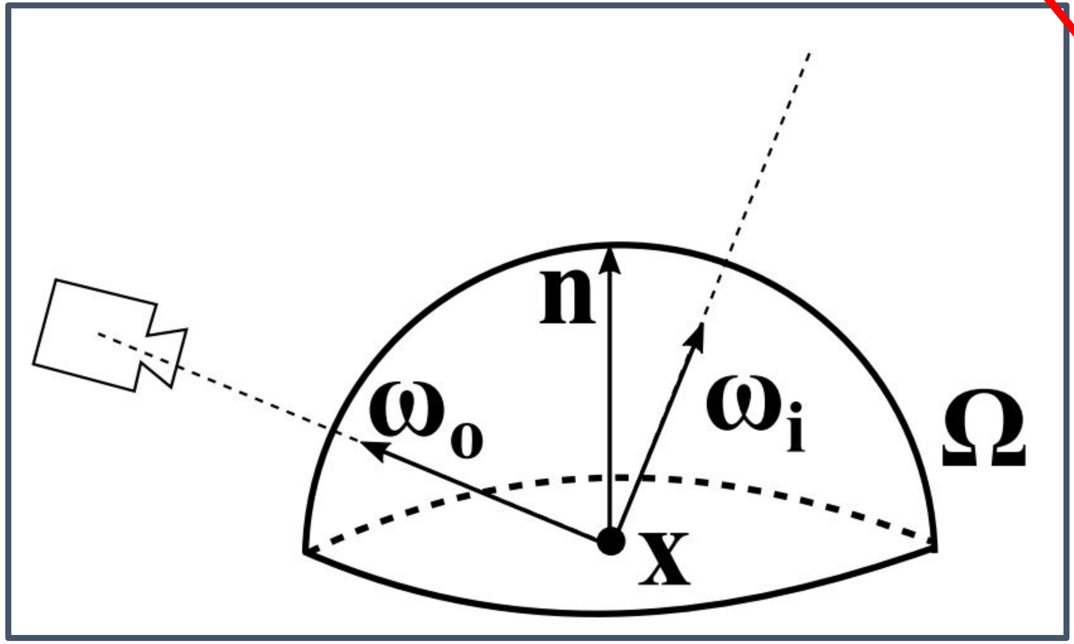


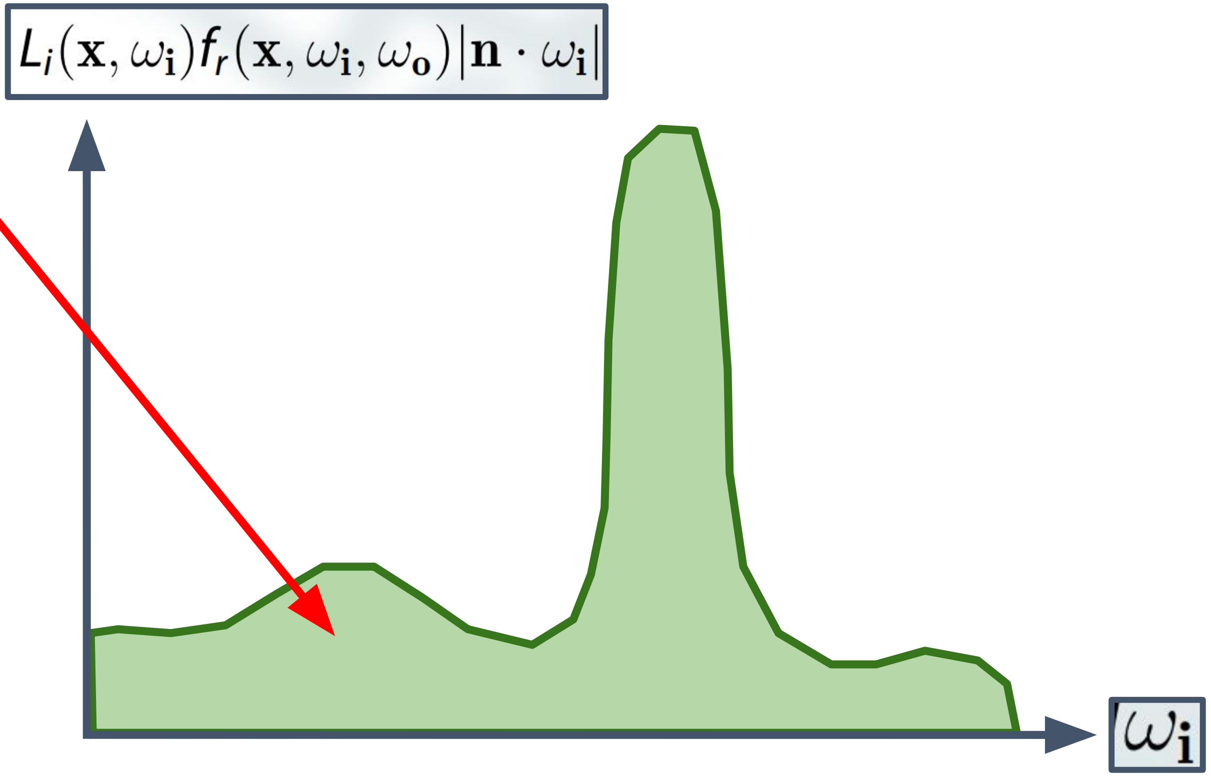


Approximating one integral

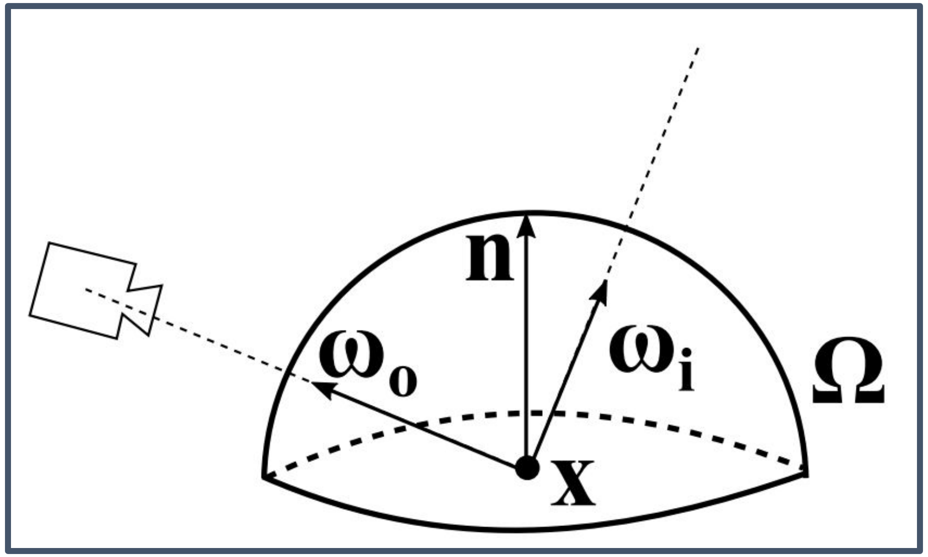
- How do we compute $\int_{\Omega} L_i(\mathbf{x},\omega_\mathbf{i}) f_r(\mathbf{x},\omega_\mathbf{i},\omega_\mathbf{o}) |\mathbf{n}\cdot\omega_\mathbf{i}| d\omega_\mathbf{i}$?

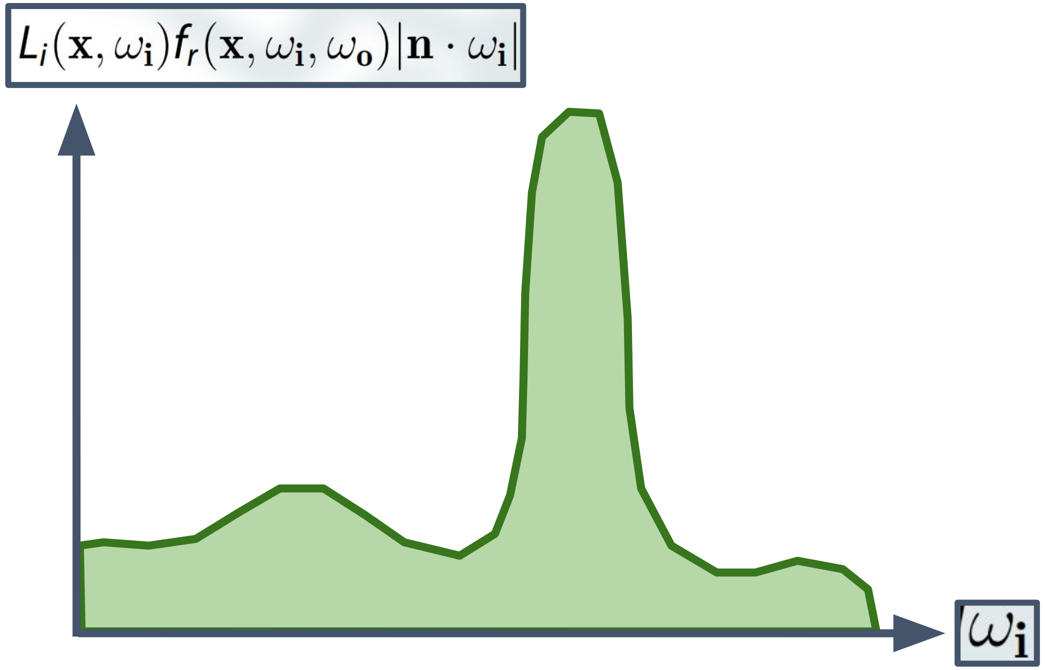
- Compute this green area under the curve





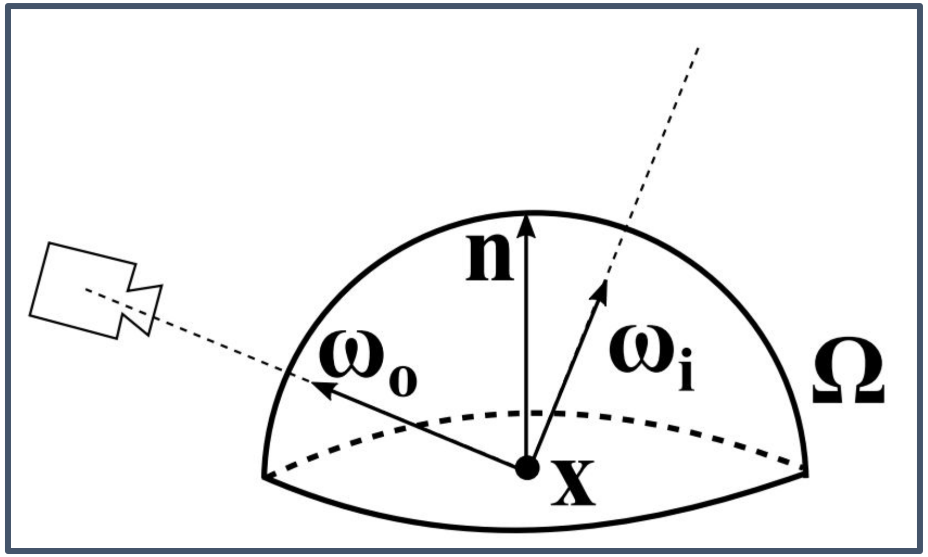
Approximating one integral

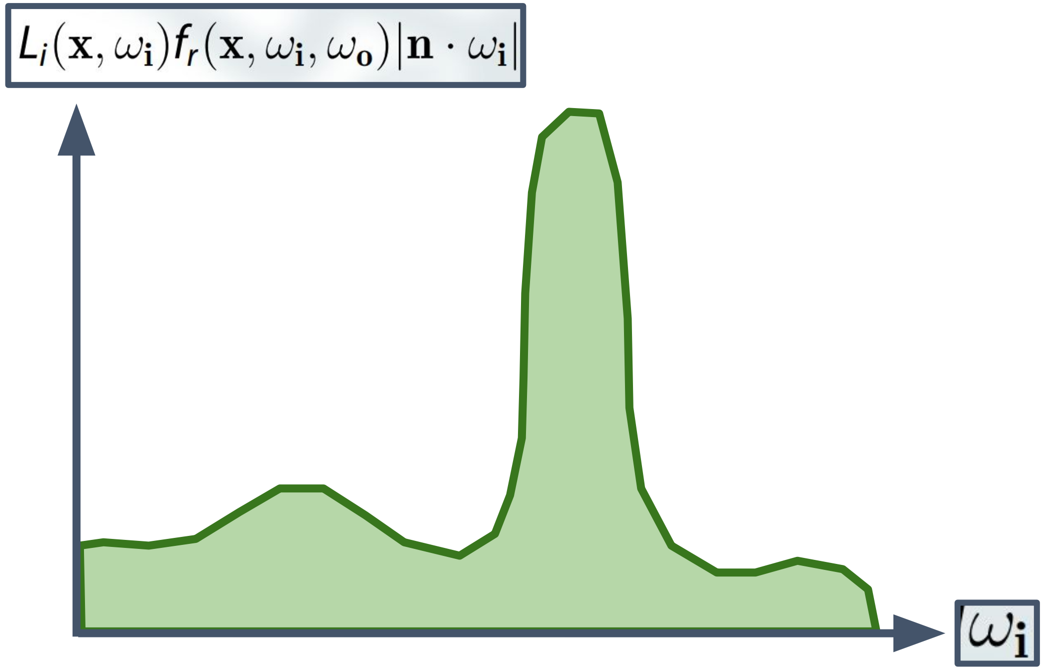




- There are infinite values for ω_i

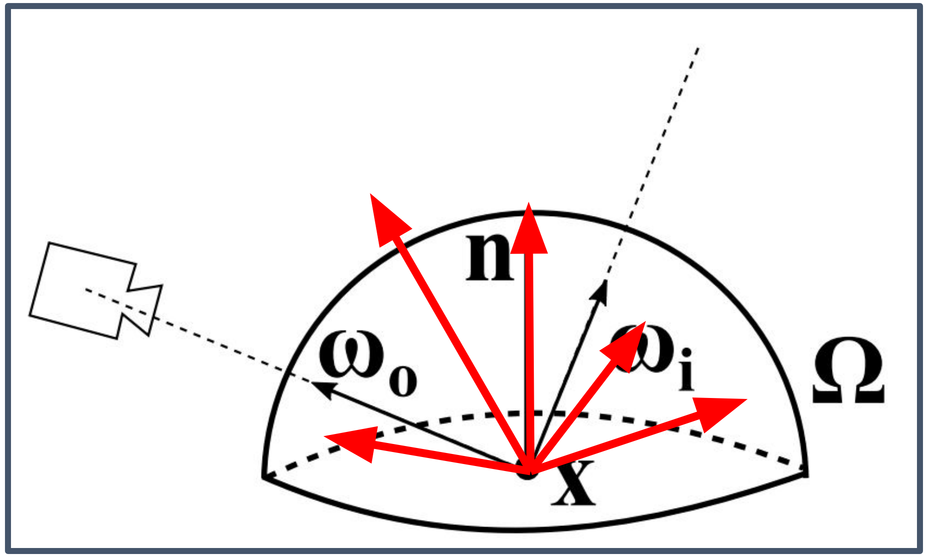
Approximating one integral

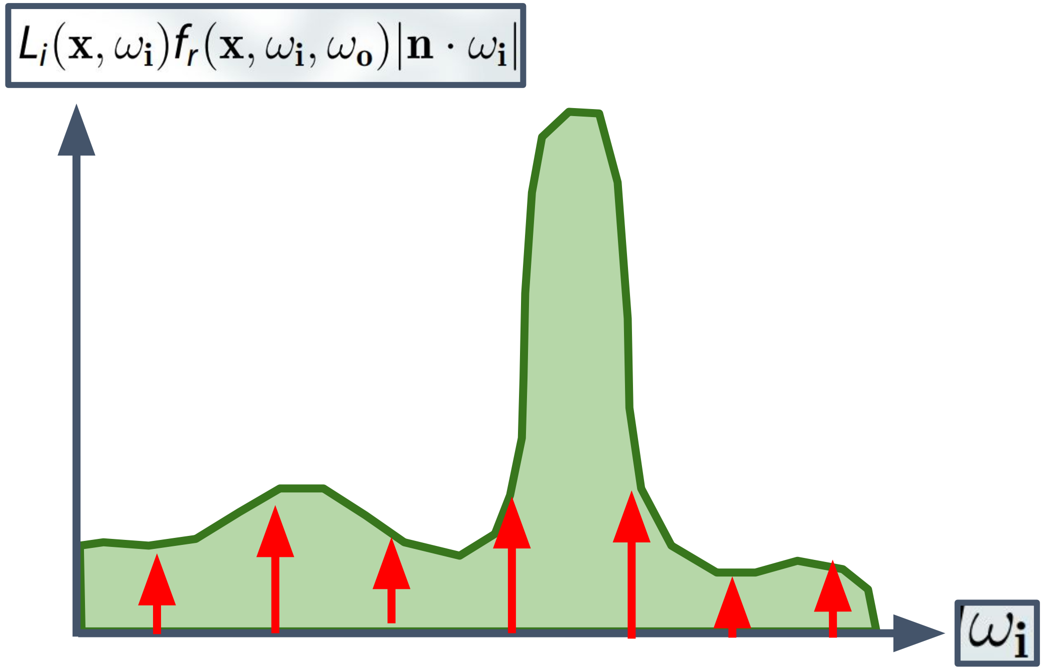




- There are infinite values for ω_i
- Idea 1: approximate using a finite amount of (evenly spaced) samples

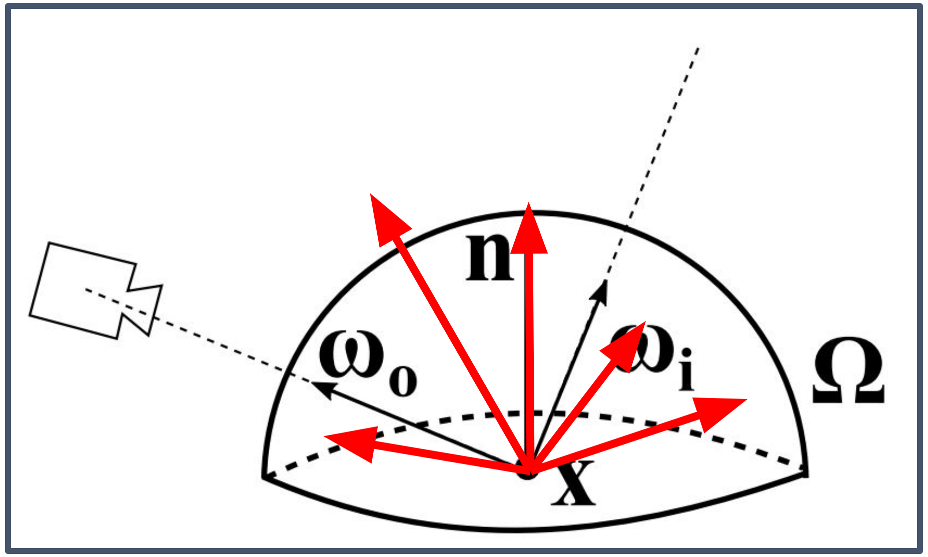
Approximating one integral

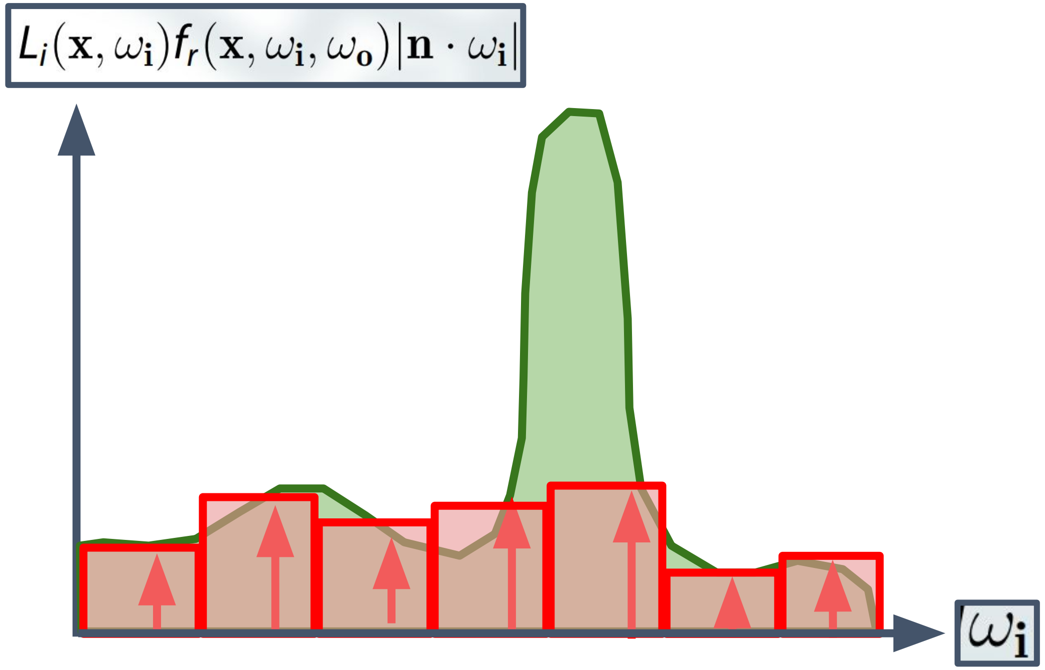




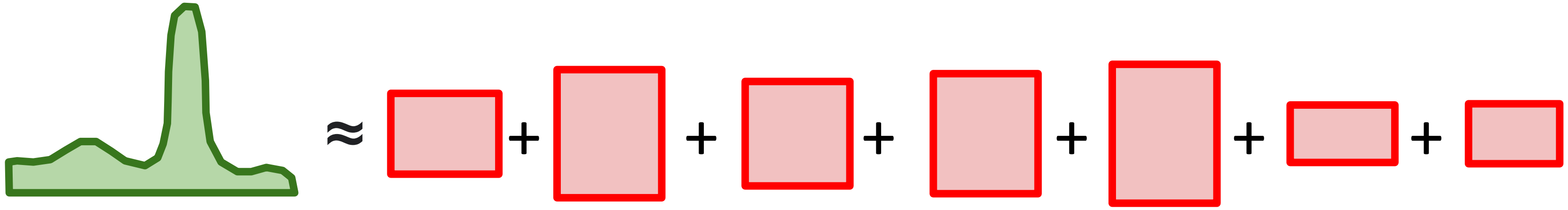
- There are infinite values for ω_i
- Idea 1: approximate using a finite amount of (evenly spaced) samples

Approximating one integral

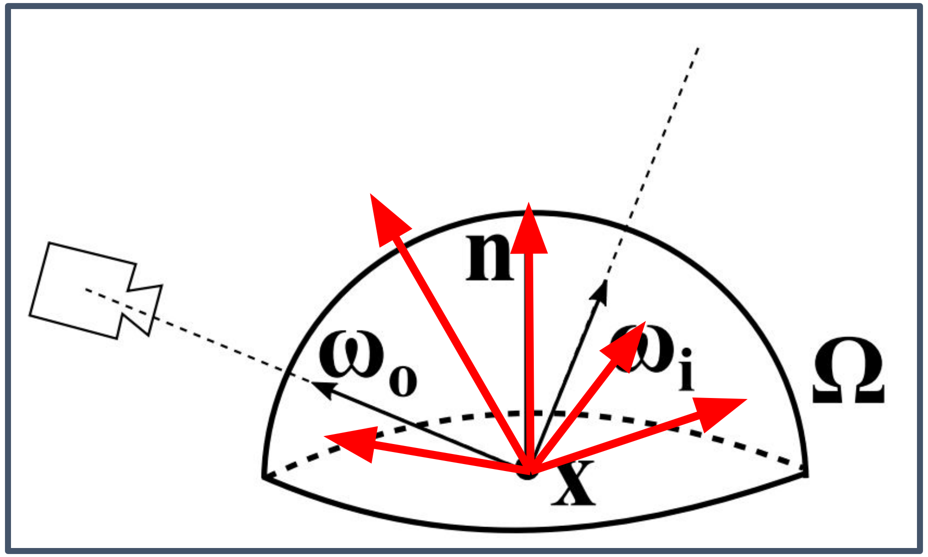


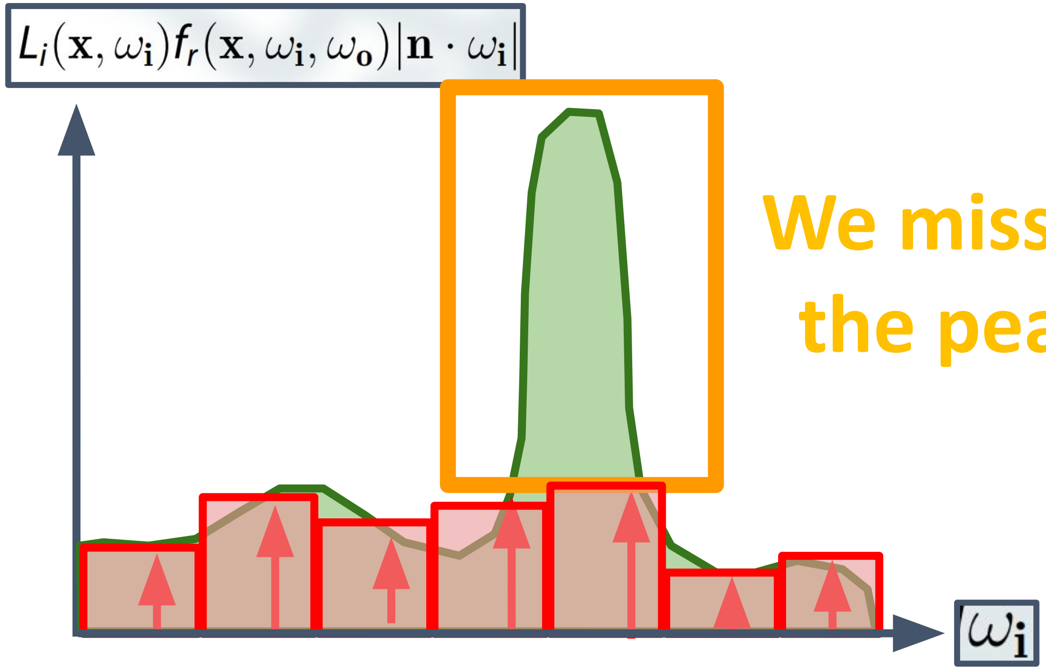


- There are infinite values for ω_i
- Idea 1: approximate using a finite amount of (evenly spaced) samples

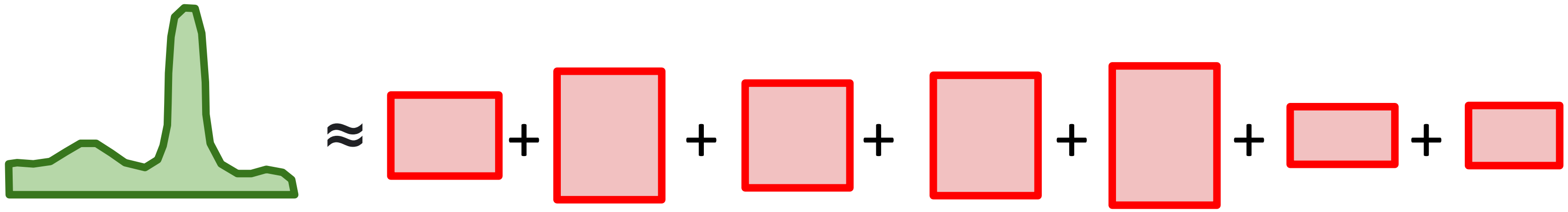


Approximating one integral

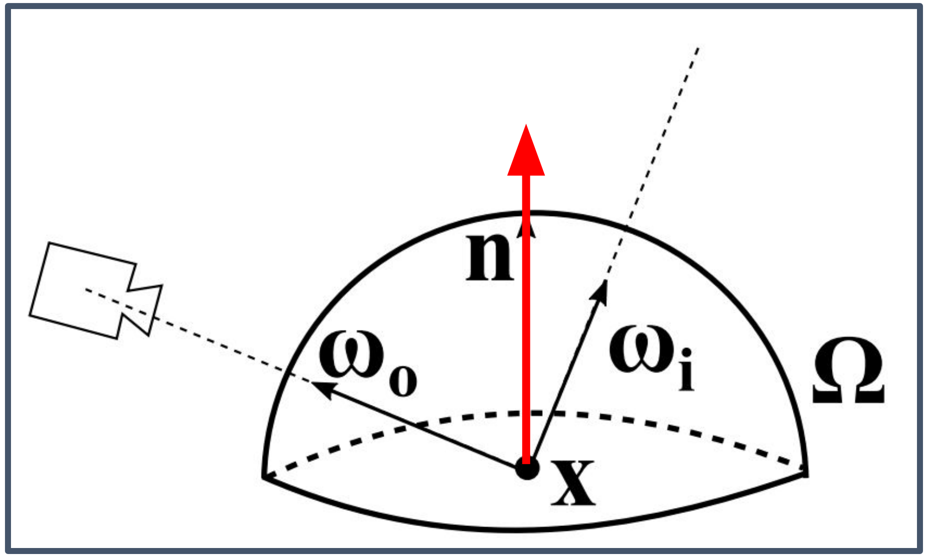


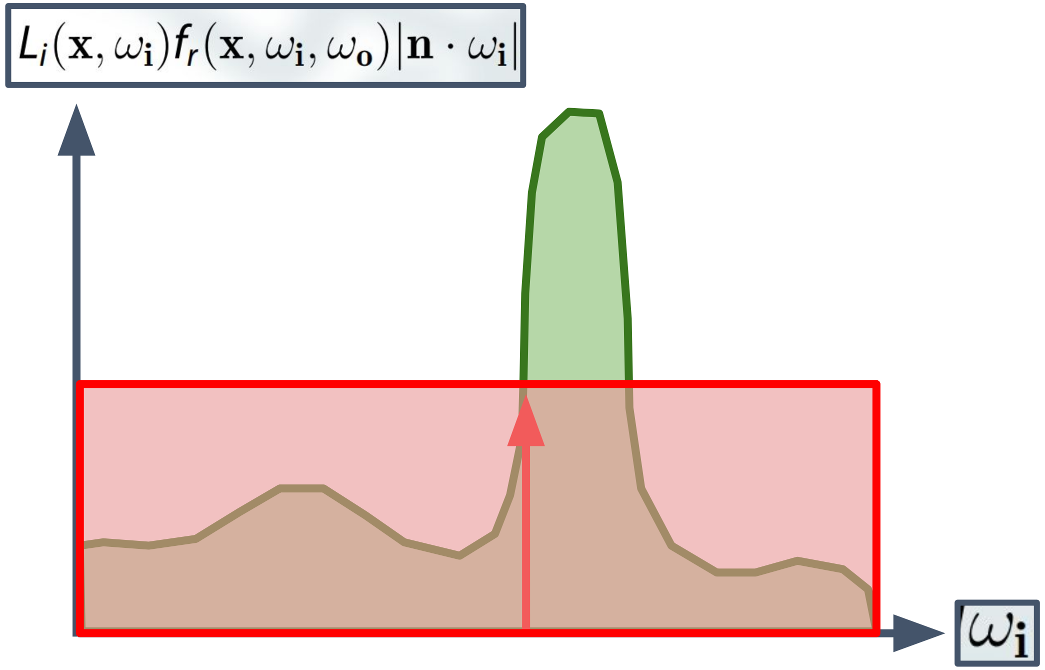


- There are infinite values for ω_i
- Idea 1: approximate using a finite amount of (evenly spaced) samples



Approximating one integral



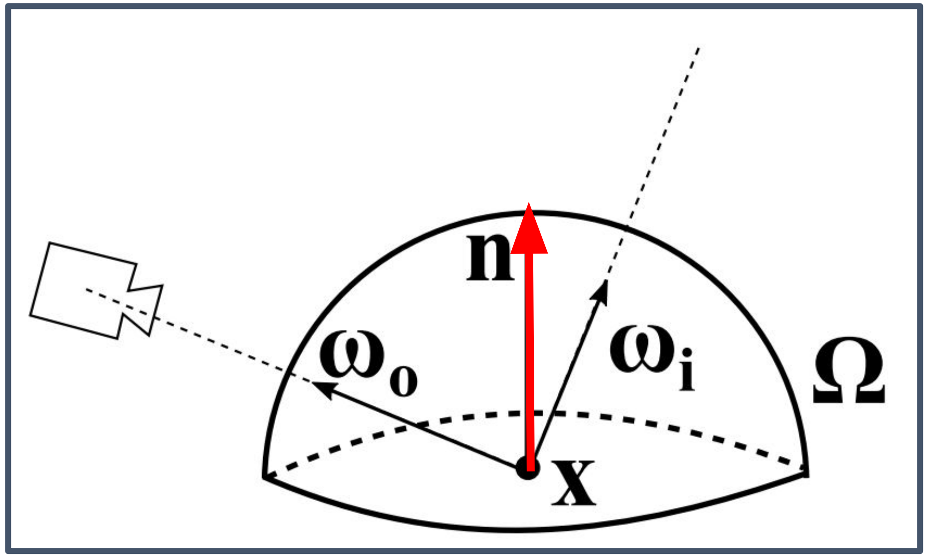


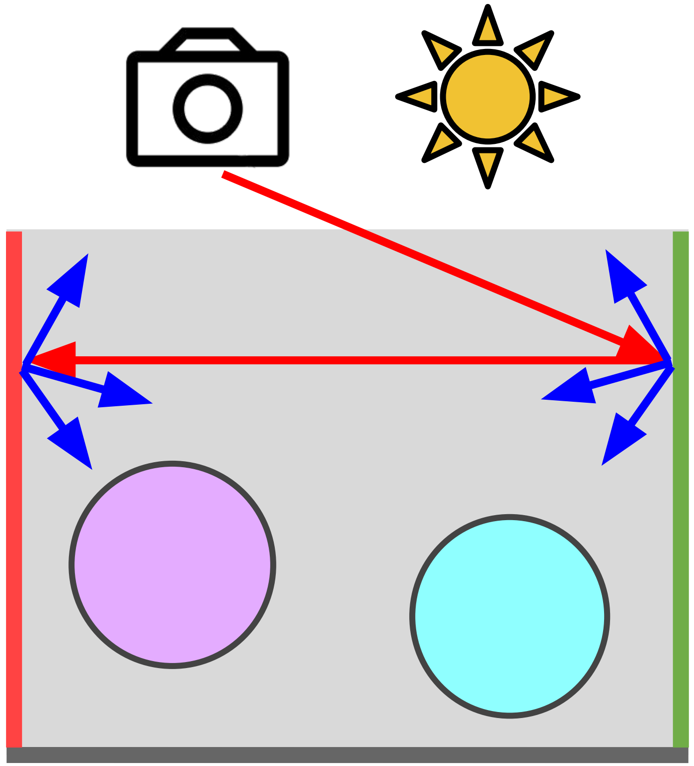
- There are infinite values for ω_i
- Idea 1: approximate using a finite amount of (evenly spaced) samples
  - Example of one deterministic sample

Approximating one integral in practice

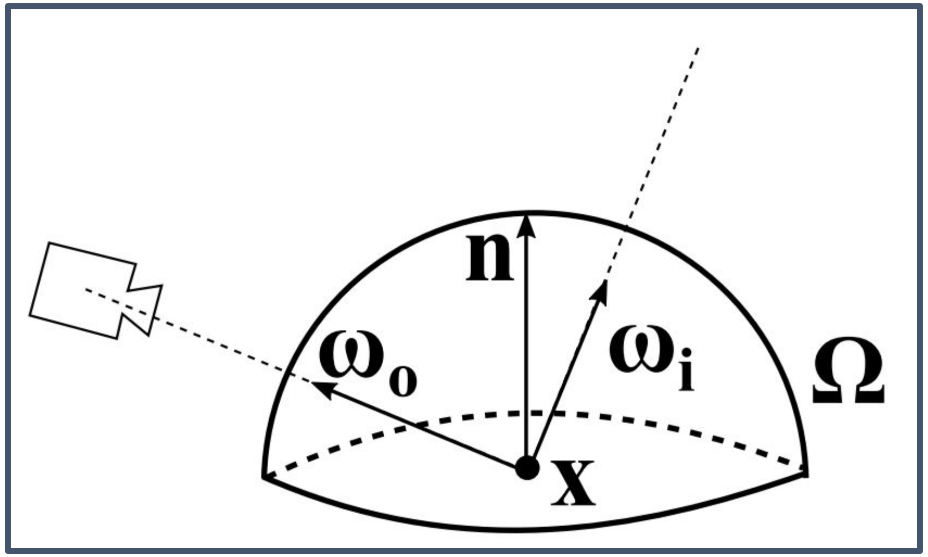
Graphics and Imaging Lab

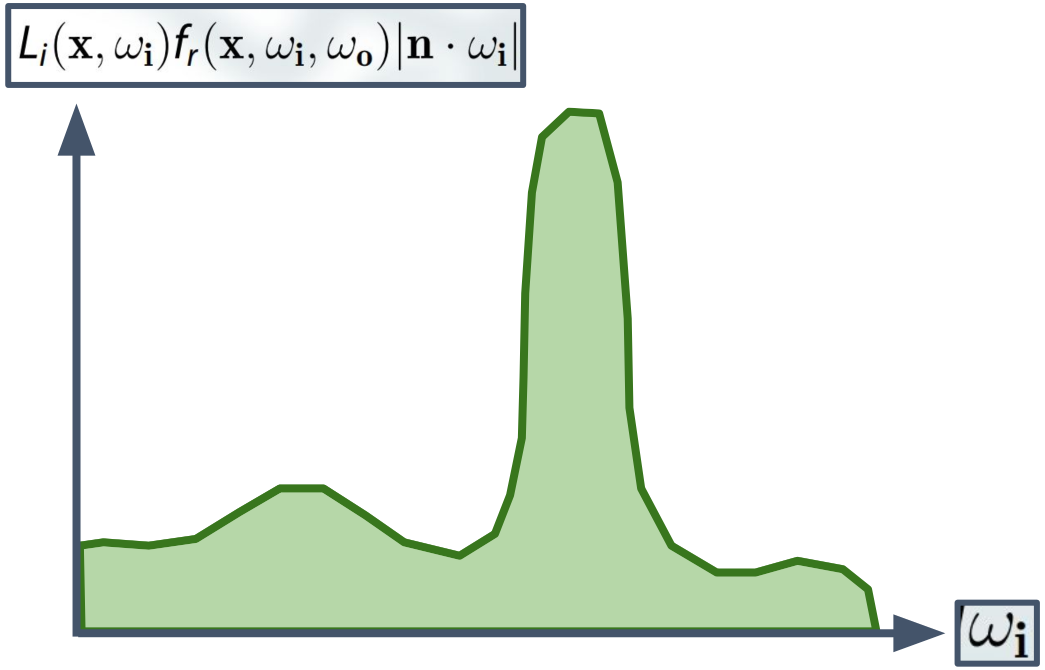
- If all rays go towards **n**, you ignore the rest of the hemisphere
  - Notice this will also happen with N samples
- Produces what's called a biased result (ignores)





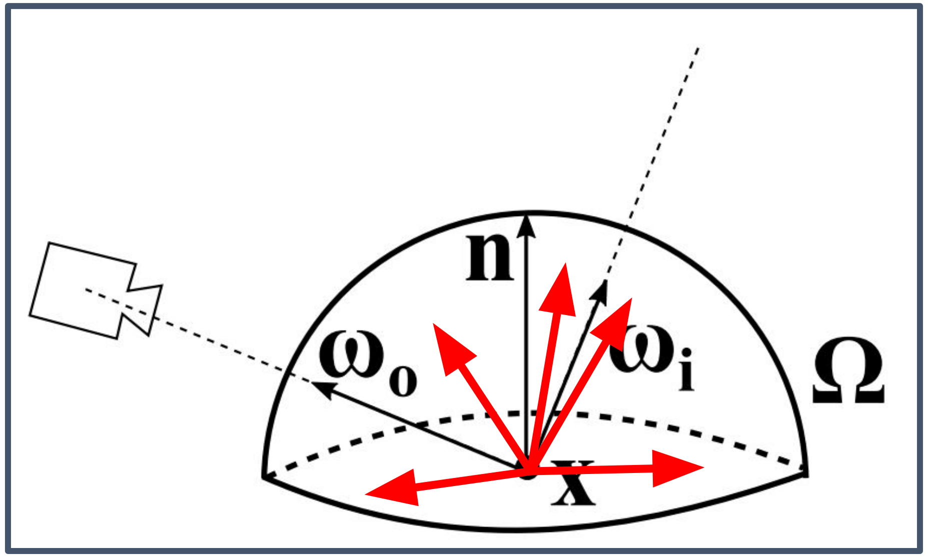
Approximating one integral

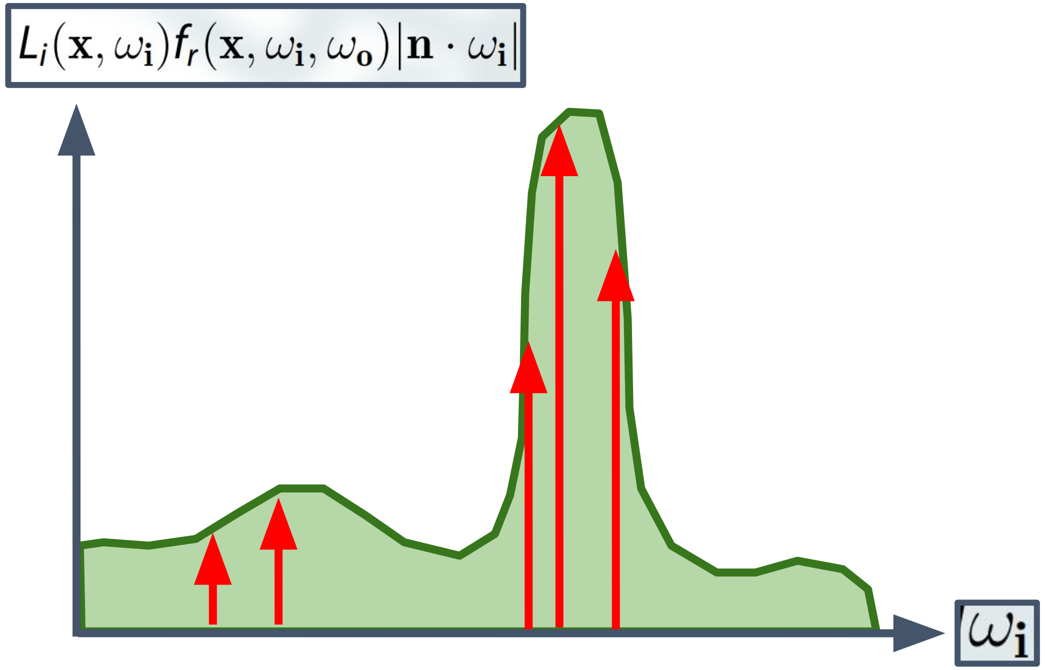




- There are infinite values for ω_i
- ~~Idea 1: approximate using a finite amount of (evenly spaced) samples~~
- Idea 2: **Monte Carlo estimator**, use the mean of **N** random samples

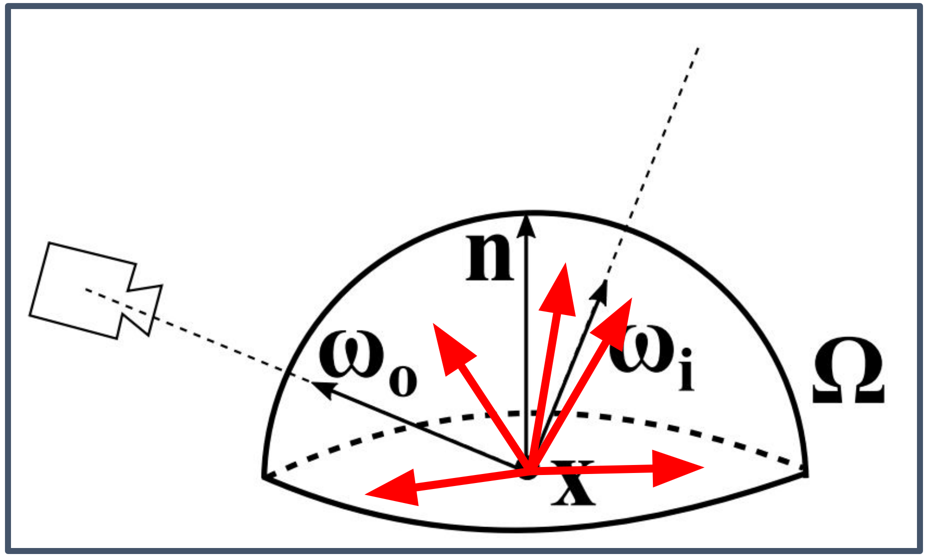
Approximating one integral

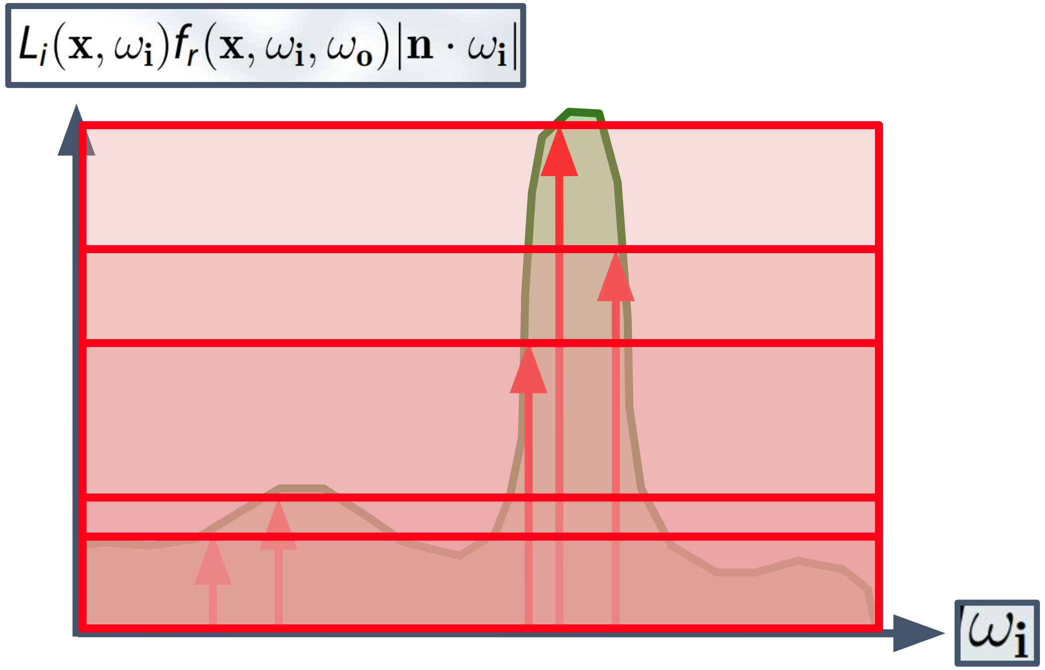




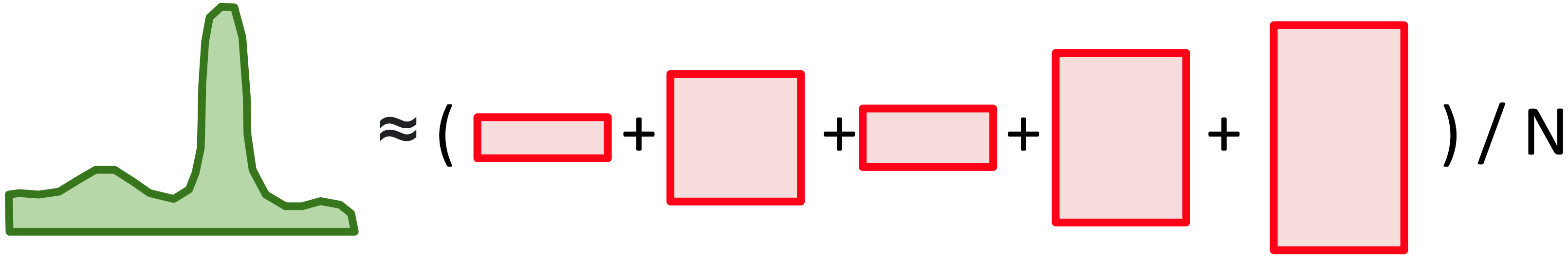
- There are infinite values for ω_i
- ~~Idea 1: approximate using a finite amount of (evenly spaced) samples~~
- Idea 2: **Monte Carlo estimator**, use the mean of **N** random samples

Approximating one integral



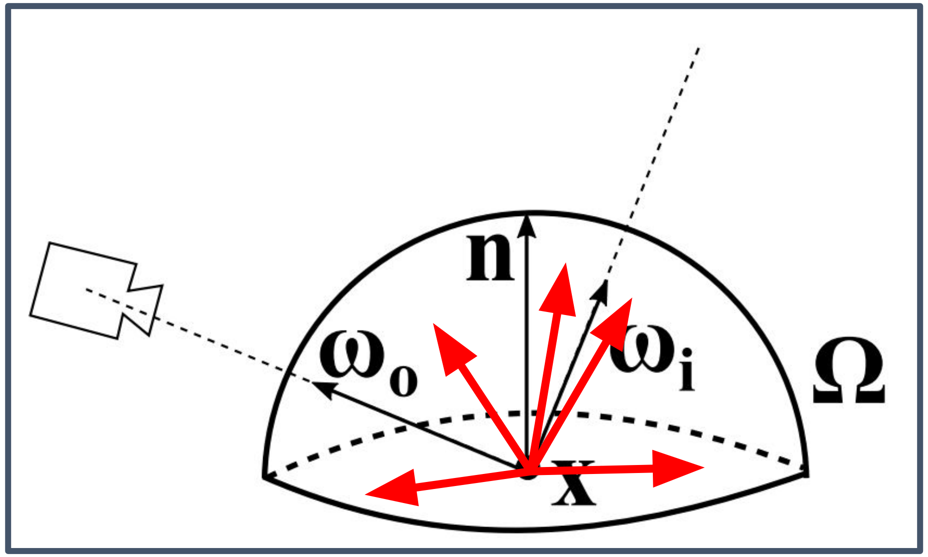


- There are infinite values for ω_i
- Idea 2: **Monte Carlo estimator**, use the mean of N random samples

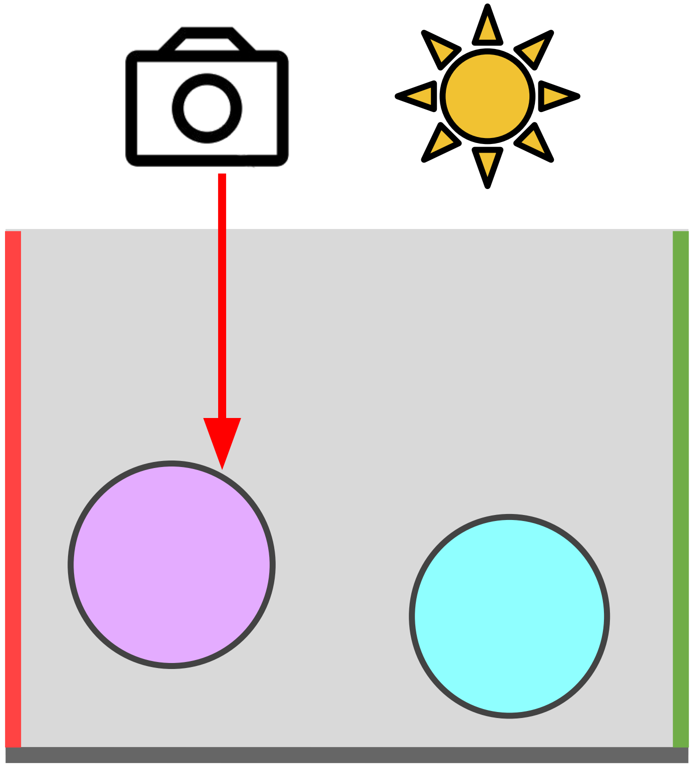


Approximating one integral in practice

- What happens if you approximate integrals using N samples?

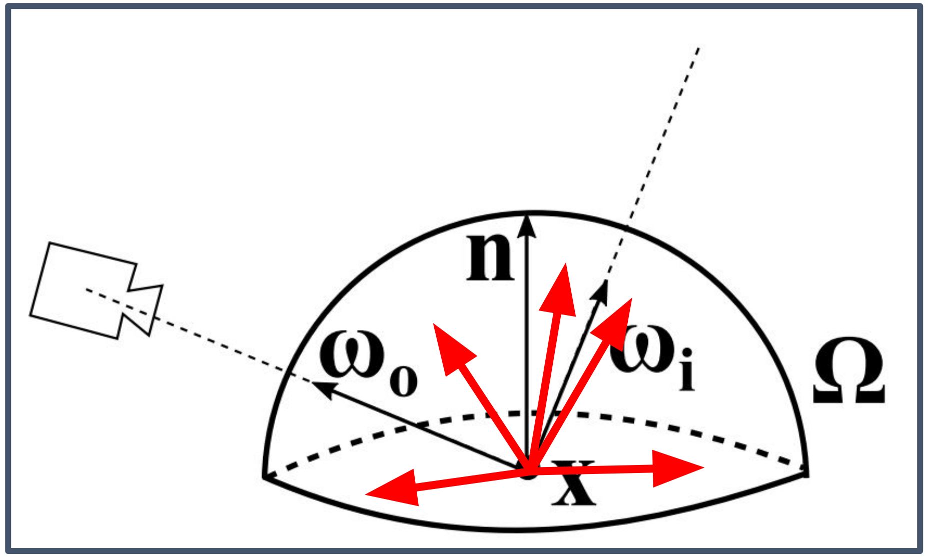


One ray from the camera

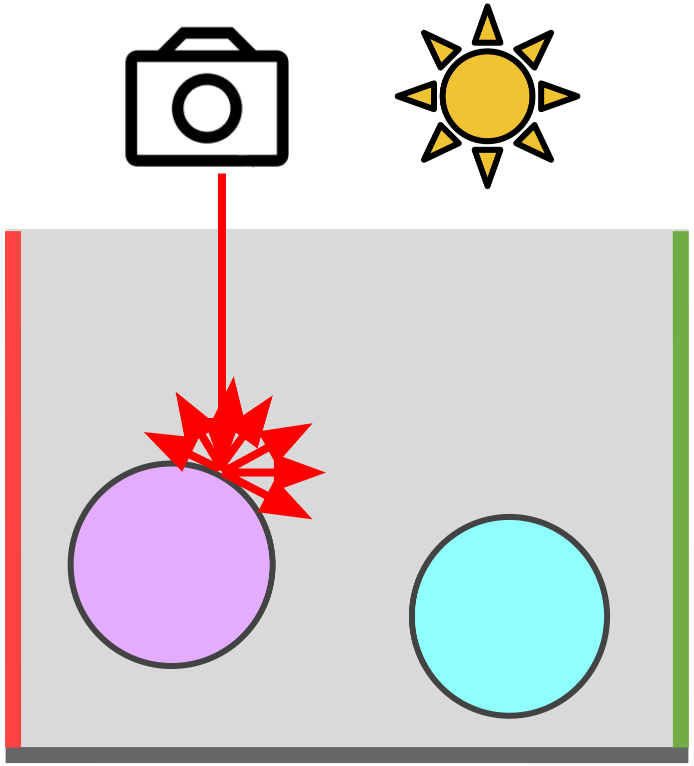


Approximating one integral in practice

- What happens if you approximate integrals using N samples?

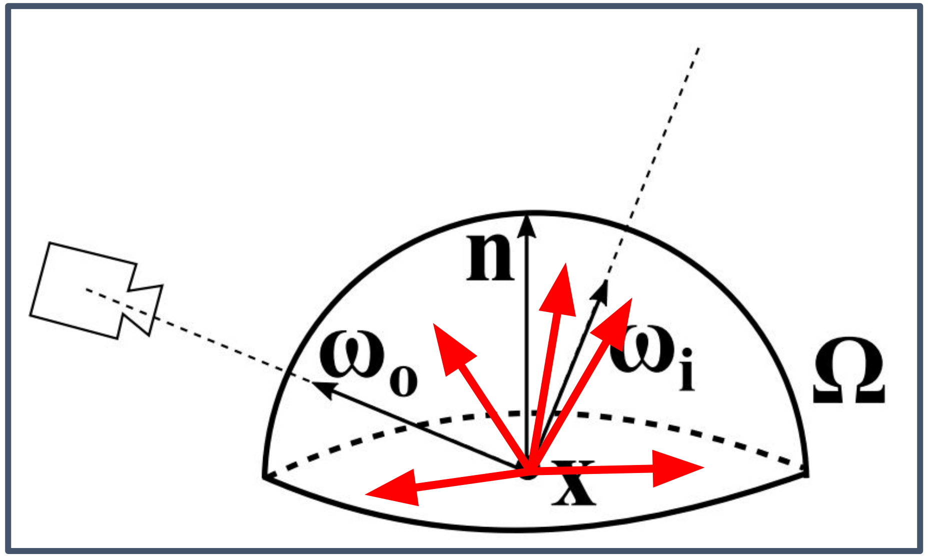


N rays for the first bounce

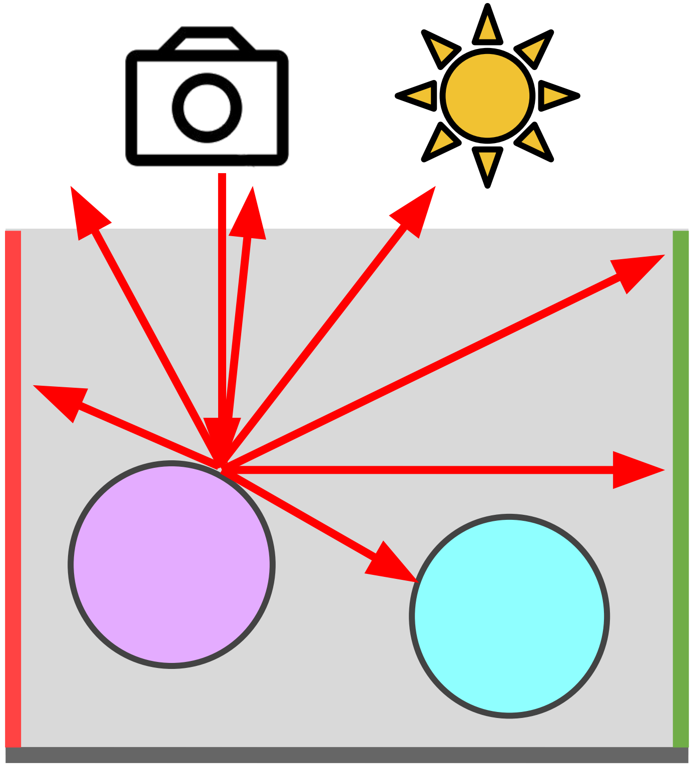


Approximating one integral in practice

- What happens if you approximate integrals using N samples?

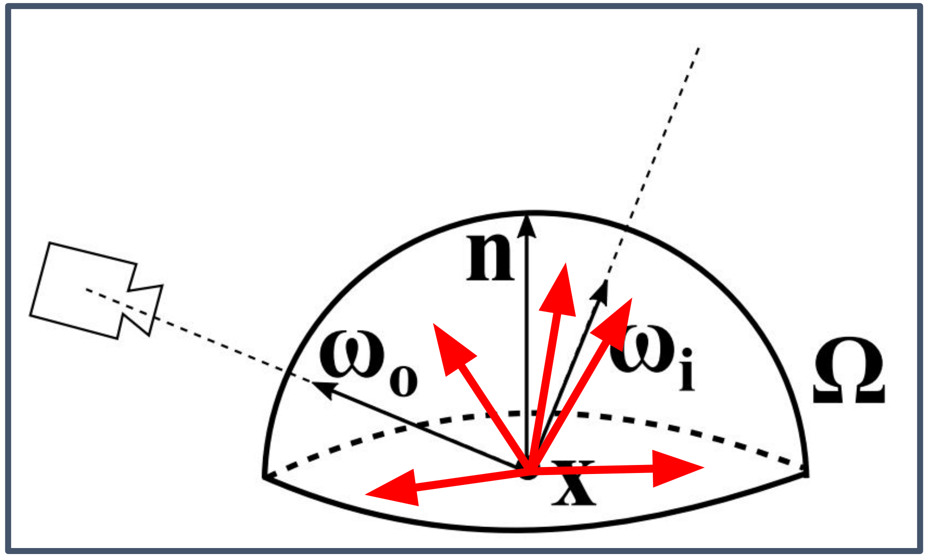


N rays for the first bounce

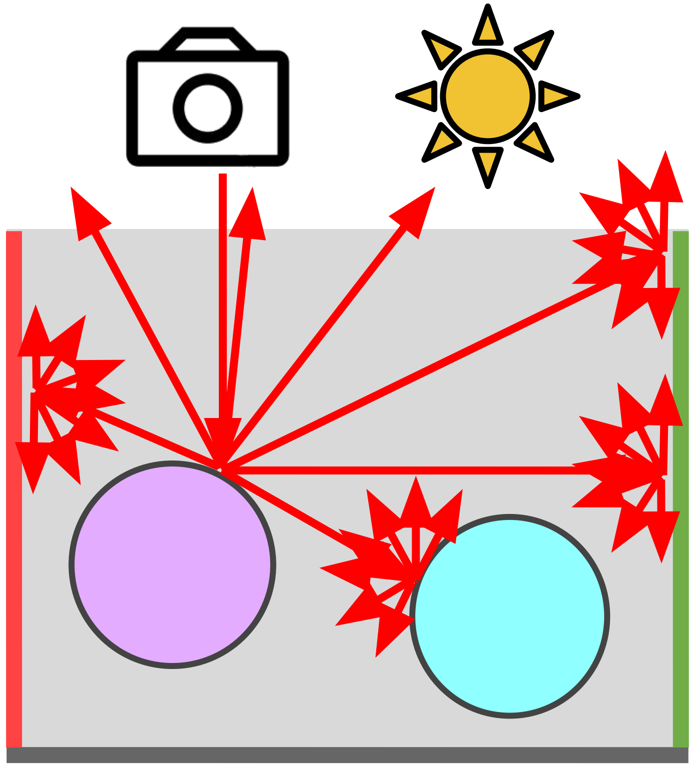


Approximating one integral in practice

- What happens if you approximate integrals using N samples?

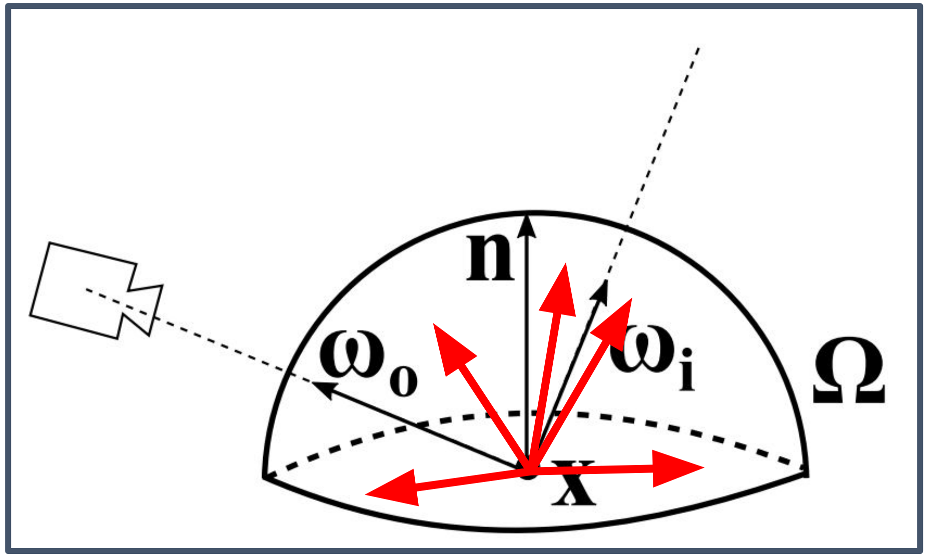


N^2 rays for the second bounce

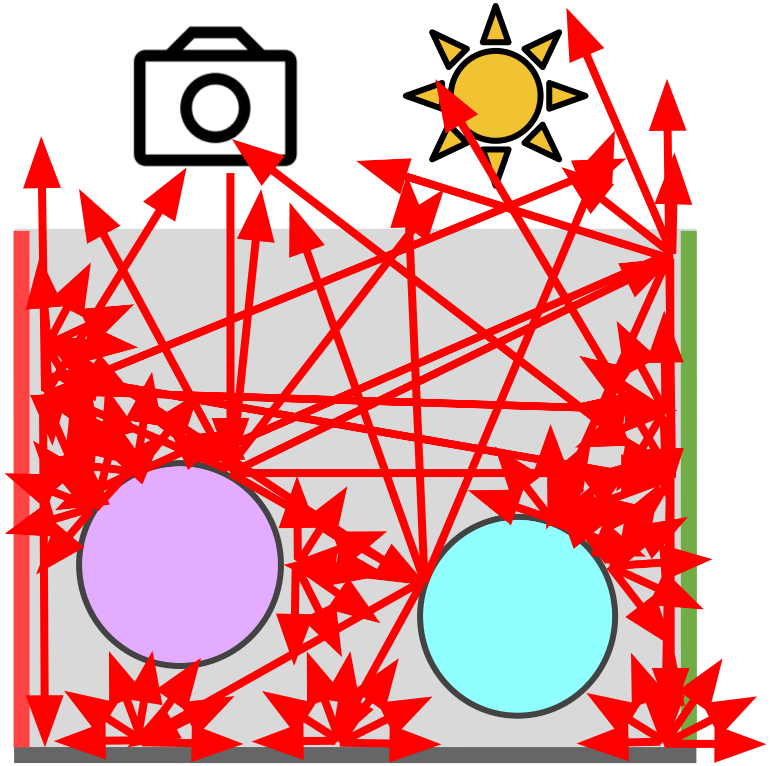


Approximating one integral in practice

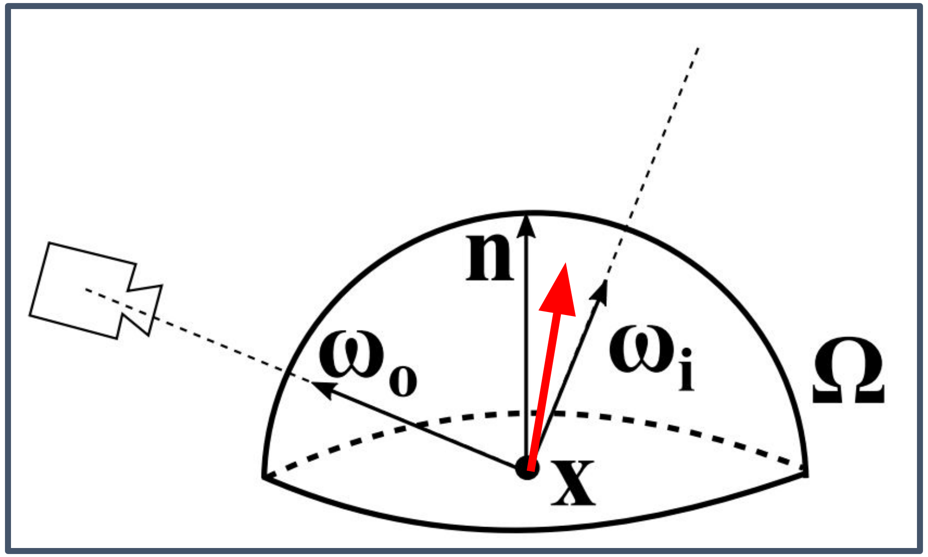
- What happens if you approximate integrals using N samples?

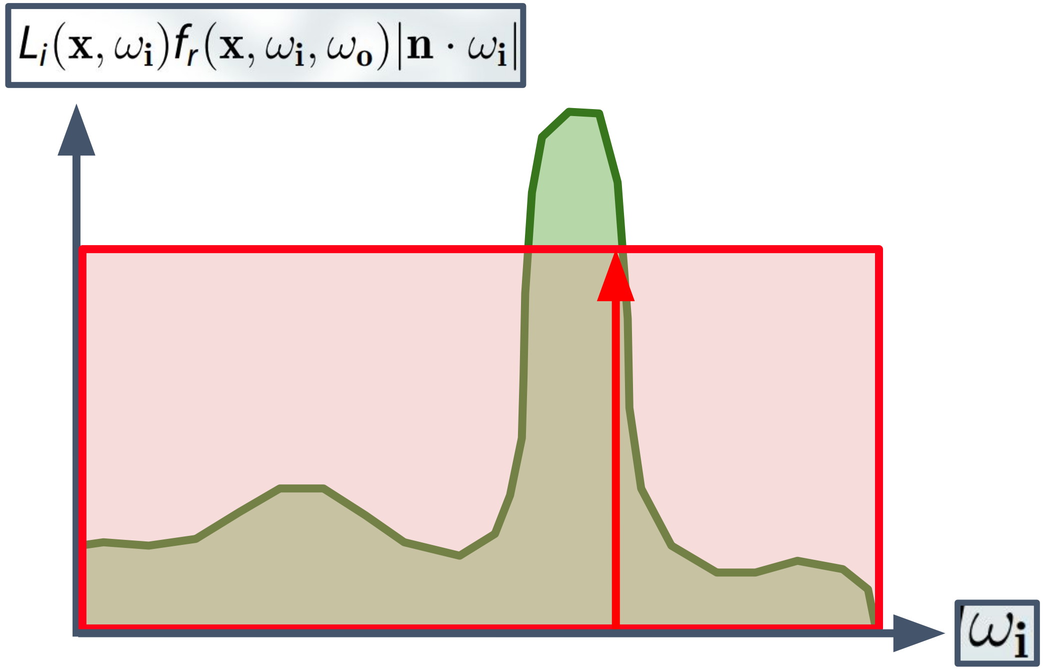


N^N rays for the Nth bounce

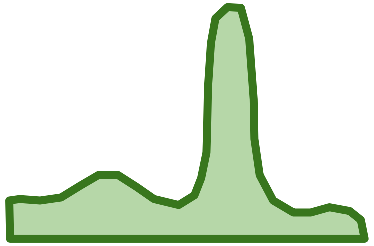
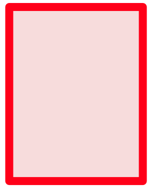


Approximating one integral



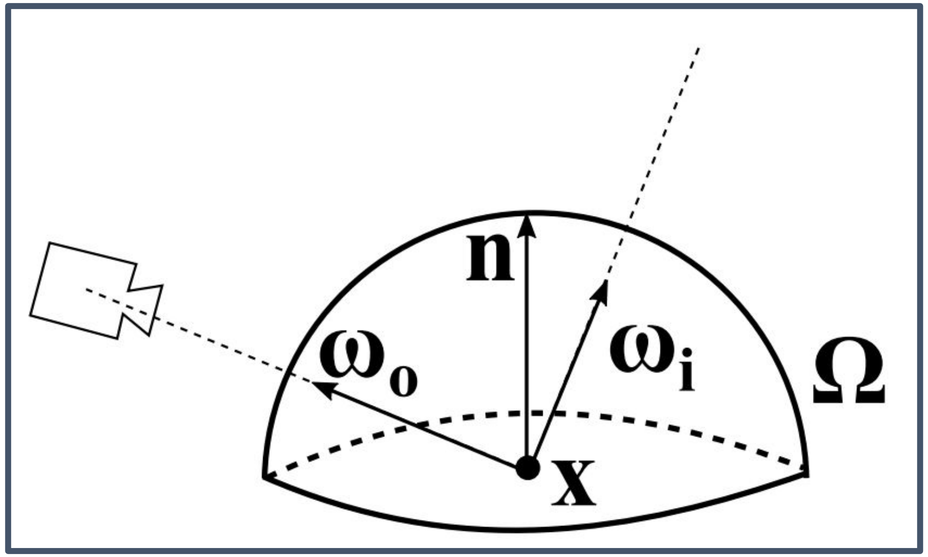


- There are infinite values for ω_i
- Idea 3: **Monte Carlo estimator**, use the mean of **N = 1** random sample

 ≈  for a randomly chosen direction

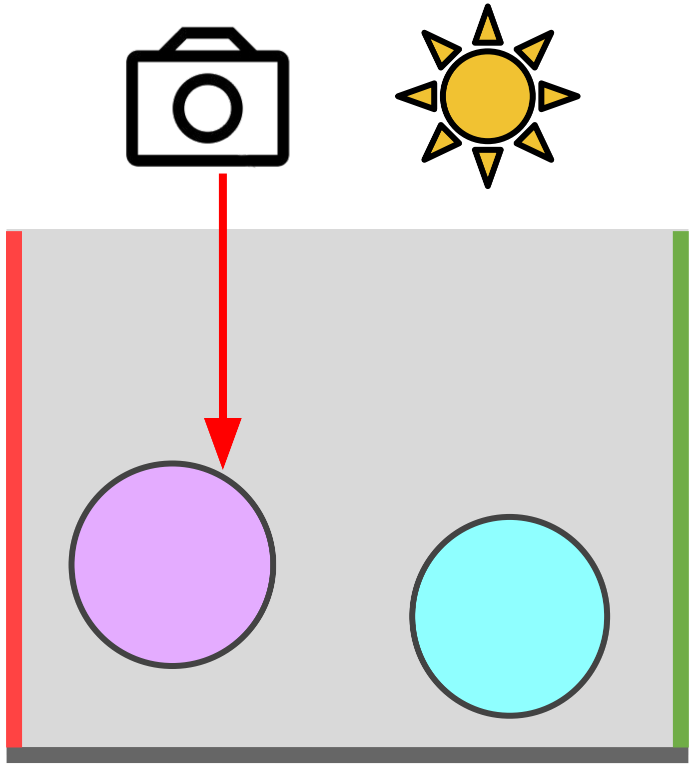
Approximating one integral in practice

- Monte Carlo estimation for the path integral



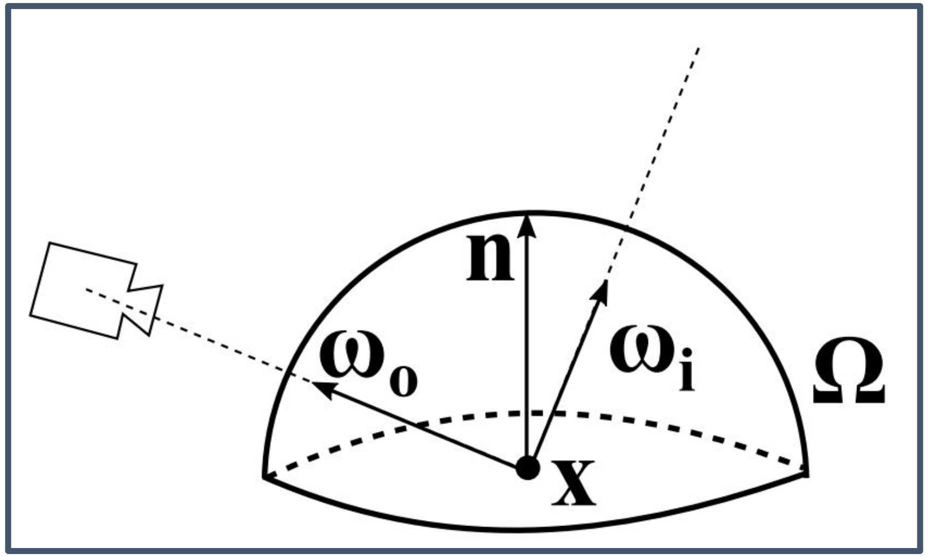
One random ω_i on each bounce

One ray from the camera



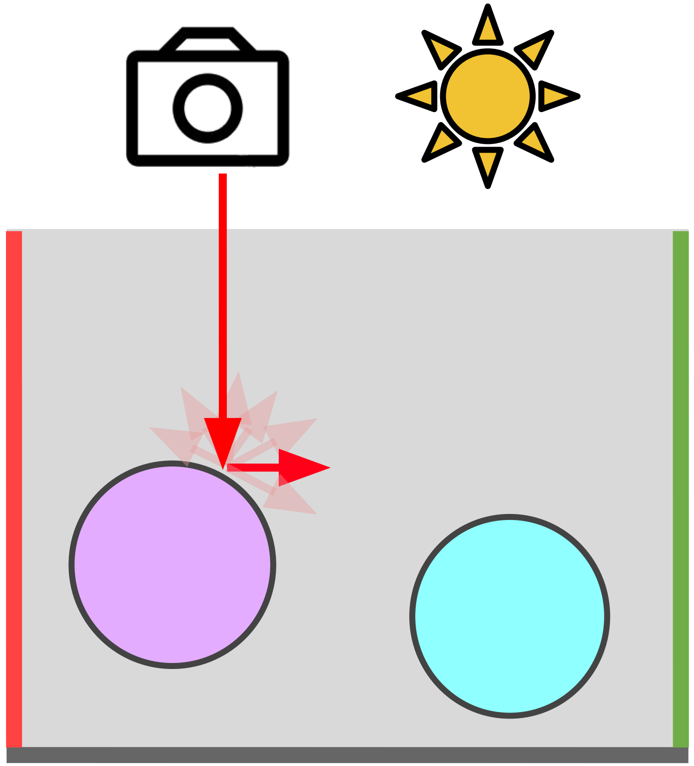
Approximating one integral in practice

- Monte Carlo estimation for the path integral



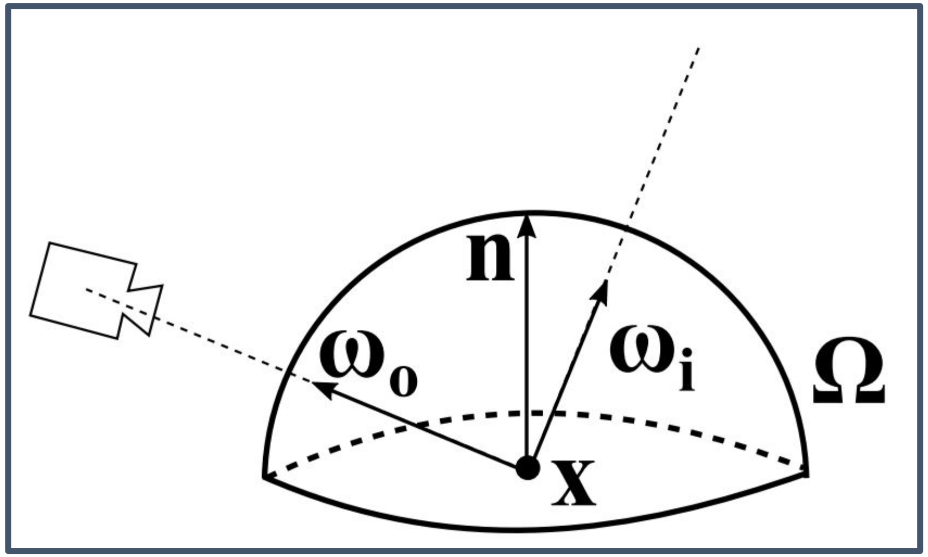
One random ω_i on each bounce

One random ray for the first bounce



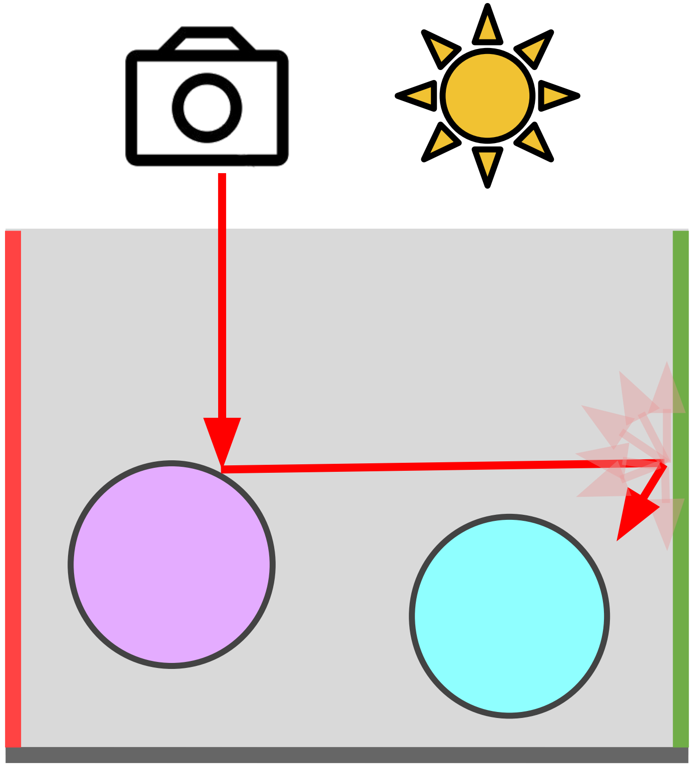
Approximating one integral in practice

- Monte Carlo estimation for the path integral



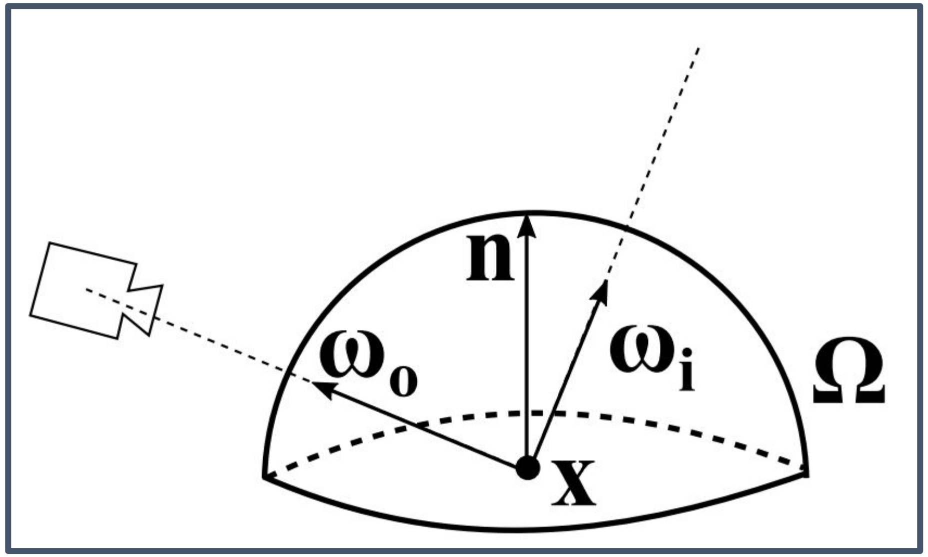
One random ω_i on each bounce

One random ray for the second bounce



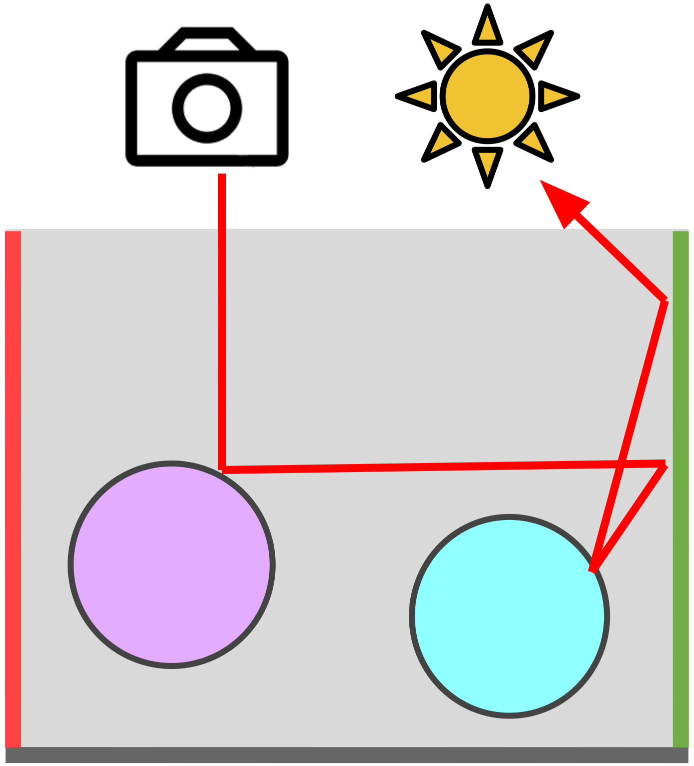
Approximating one integral in practice

- Monte Carlo estimation for the path integral



One random ω_i on each bounce

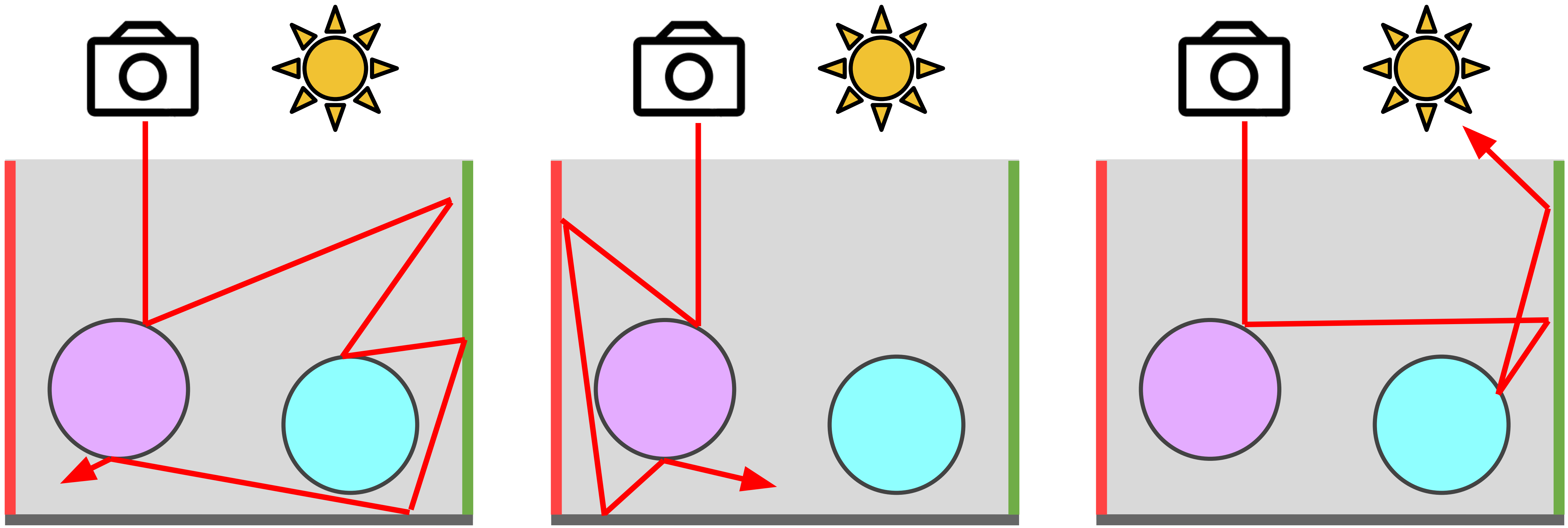
One random path



Approximating one integral in practice

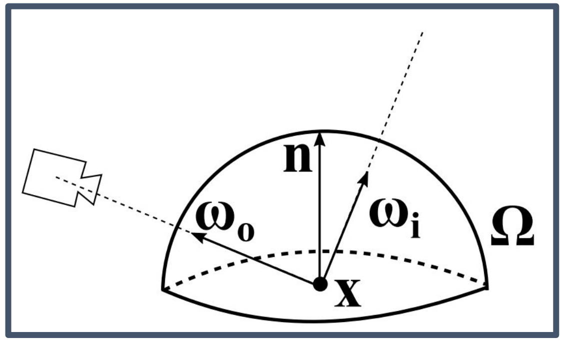
Graphics and Imaging Lab

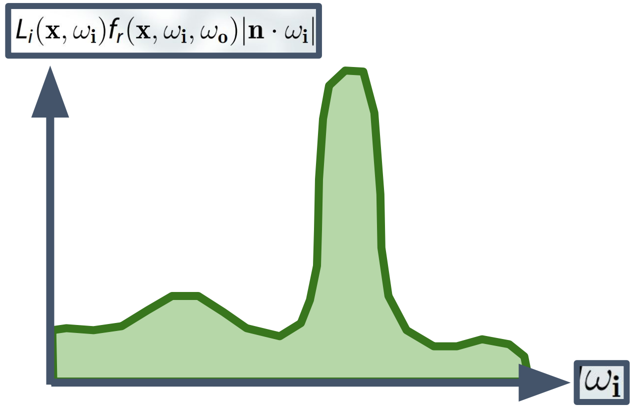
- Monte Carlo estimation for the path integral
  - Sum of multiple random paths
  - More paths ⟶ better approximation of the integral (better result)



Generating random paths

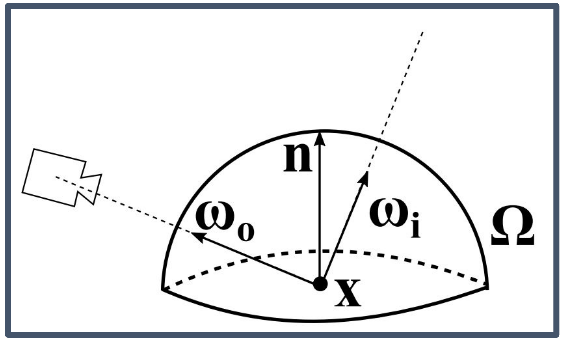
- How to generate a value for ω_i ?

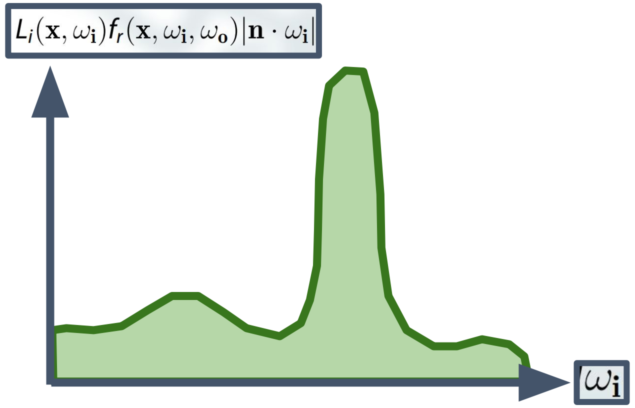




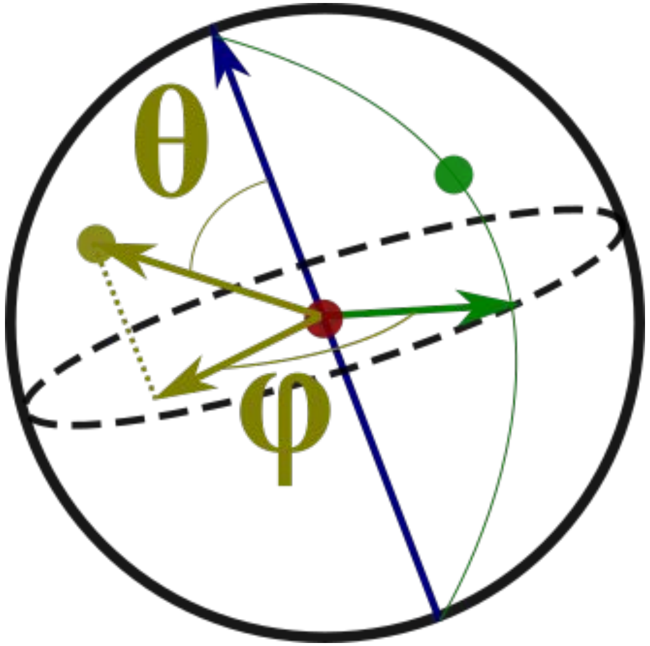
Generating random paths

- How to generate a value for $\boxed{\omega_{\mathbf{i}}}$?





- Remember your old job:



- Spherical coordinates:

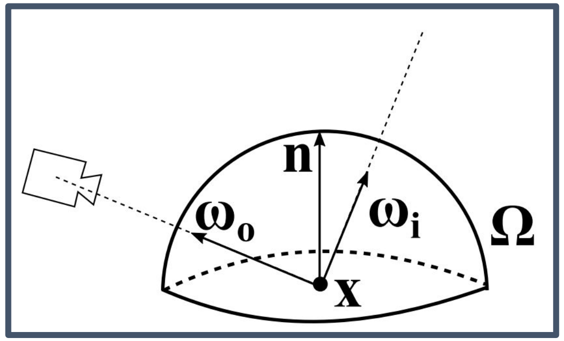
$\theta \in [0, \frac{\pi}{2})$ **(*hemi*sphere)**

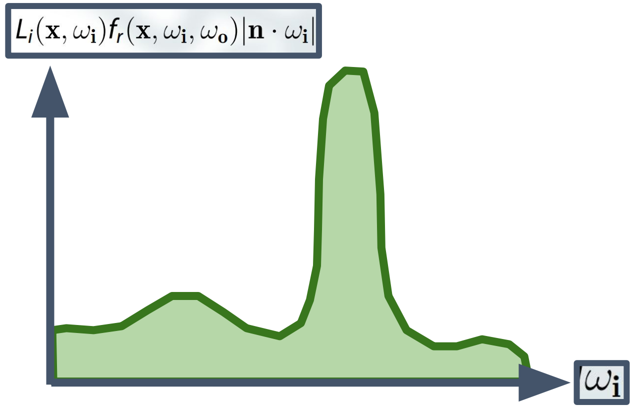
$\varphi \in [0, 2\pi)$

- Convert spherical to cartesian

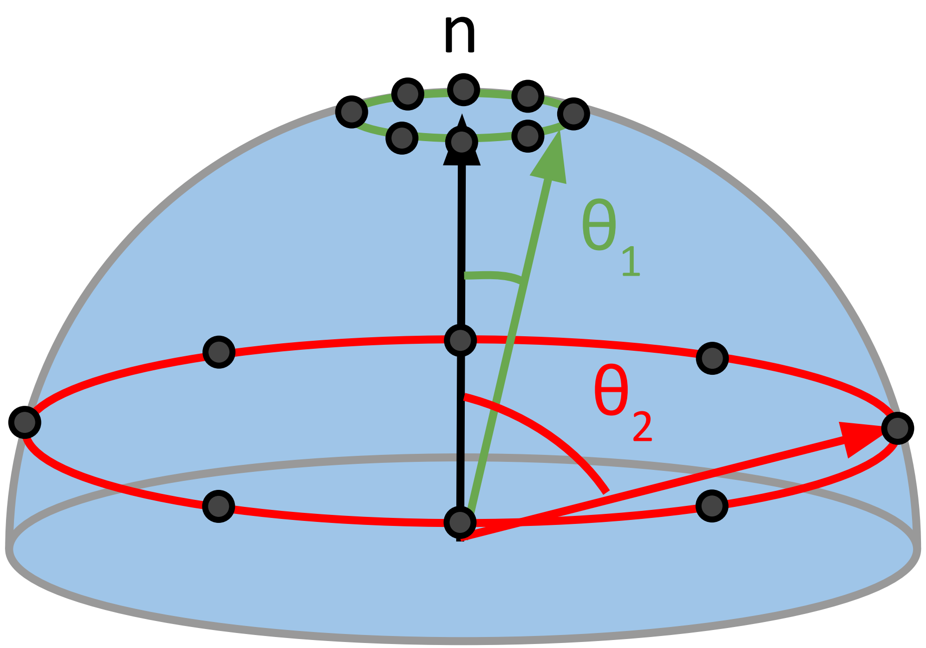
Generating random paths

- How to generate a value for ω_i ?





- Remember your old job:



- Spherical coordinates:

$$\theta \in [0, \tfrac{\pi}{2}) \quad \textbf{(\textit{hemi}sphere)}$$

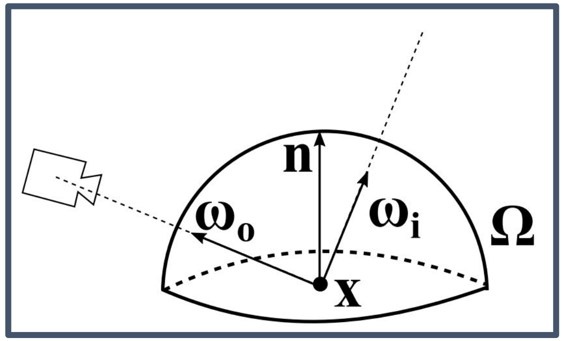
$$\varphi \in [0, 2\pi)$$

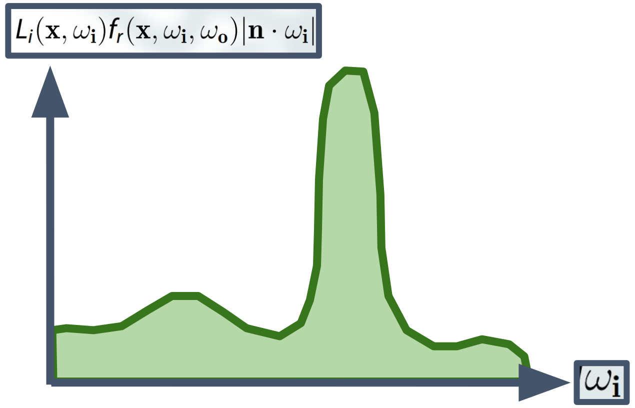
- Convert spherical to cartesian

Uniform θ, φ ≠ Uniform solid angle ω_i

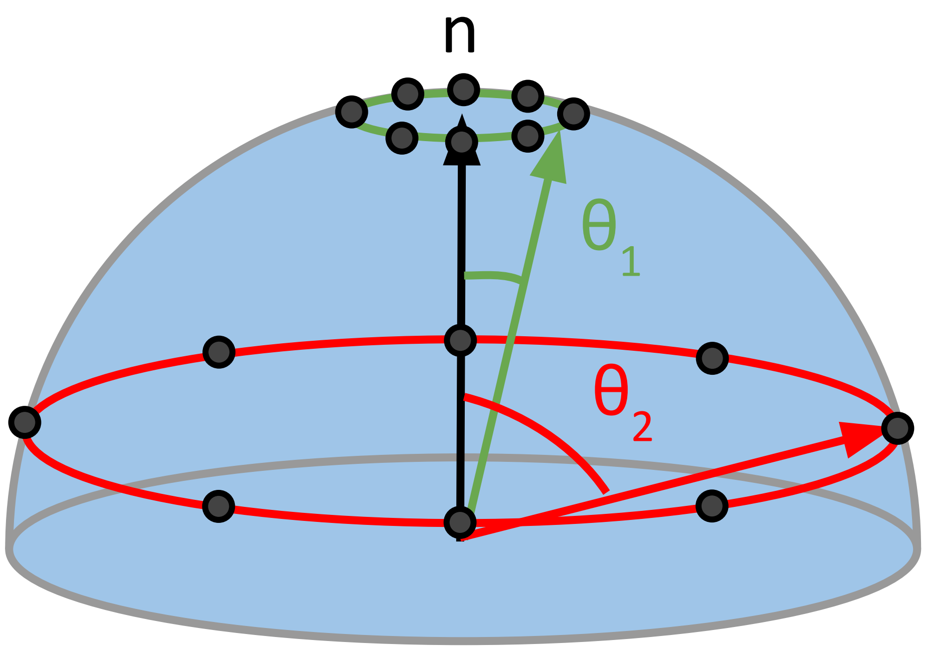
Generating random paths

- How to generate a value for $\boxed{\omega_i}$?

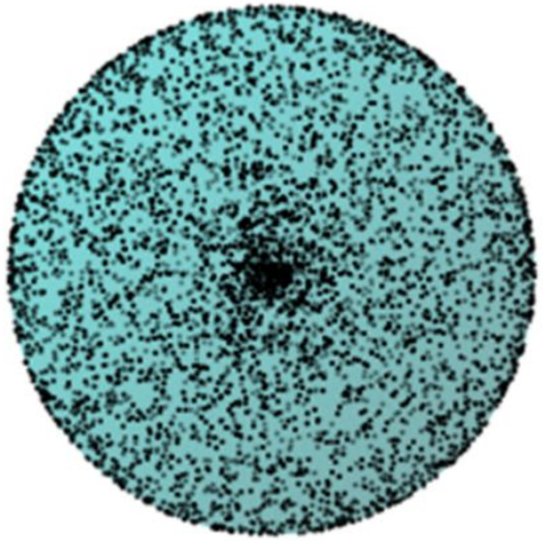




- Remember your old job:



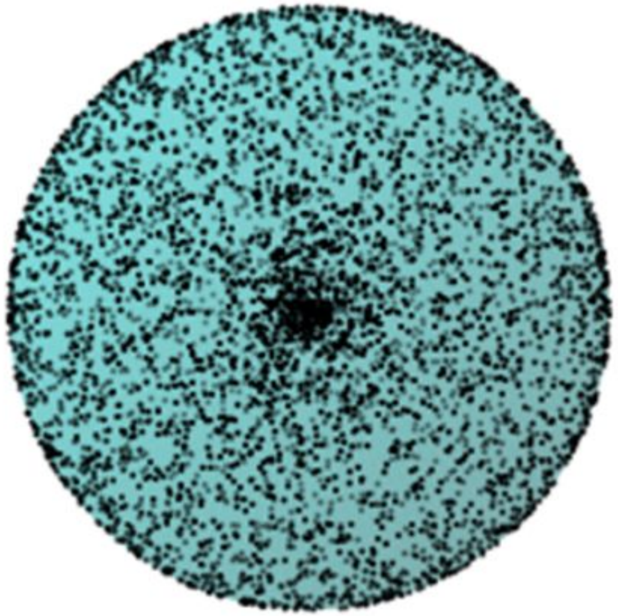
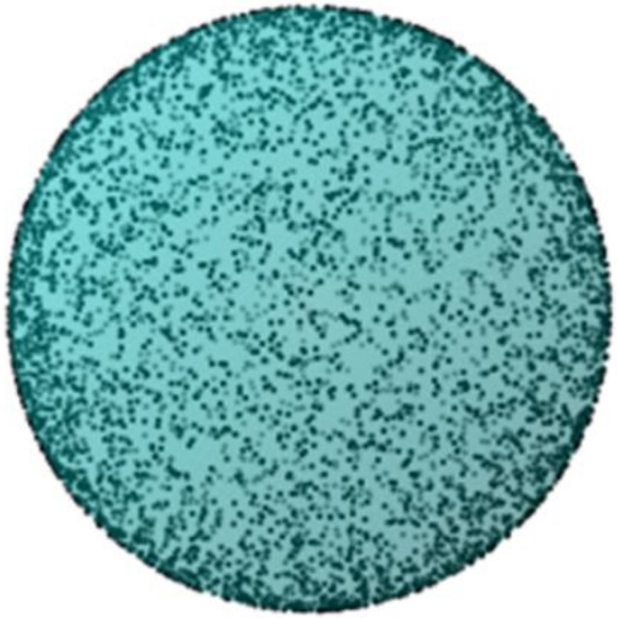
Uniform θ, φ ≠ Uniform solid angle $\boxed{\omega_i}$



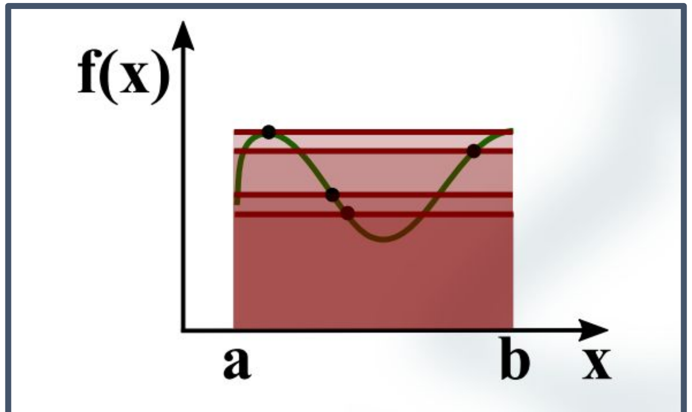
Top view (more samples near **n**)

Uniform angle vs uniform solid angle

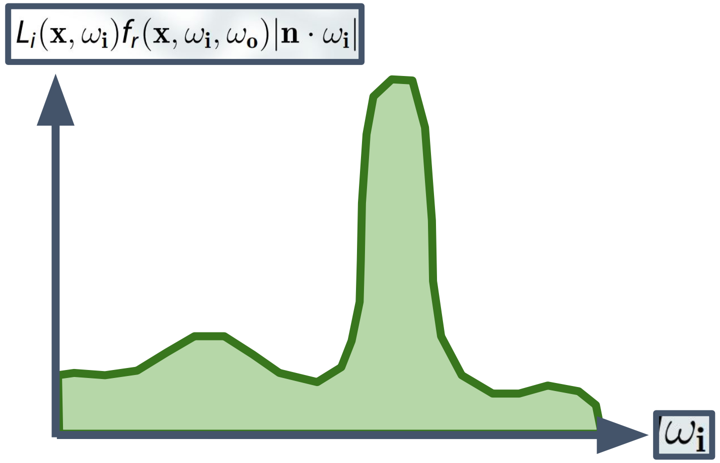
Graphics and Imaging Lab

	Uniform angle sampling	**Uniform solid angle sampling**
		
$\xi_\theta \in [0,1)$	$c^{-1}(\xi_{\theta_i}) = \frac{\pi}{2}\xi_{\theta_i}$	$c^{-1}(\xi_{\theta_i}) = \arccos \xi_{\theta_i}$
$\xi_\varphi \in [0,1)$	$c^{-1}(\xi_{\phi_i}) = 2\pi\xi_{\phi_i}$	$c^{-1}(\xi_{\phi_i}) = 2\pi\xi_{\phi_i}$

Importance sampling



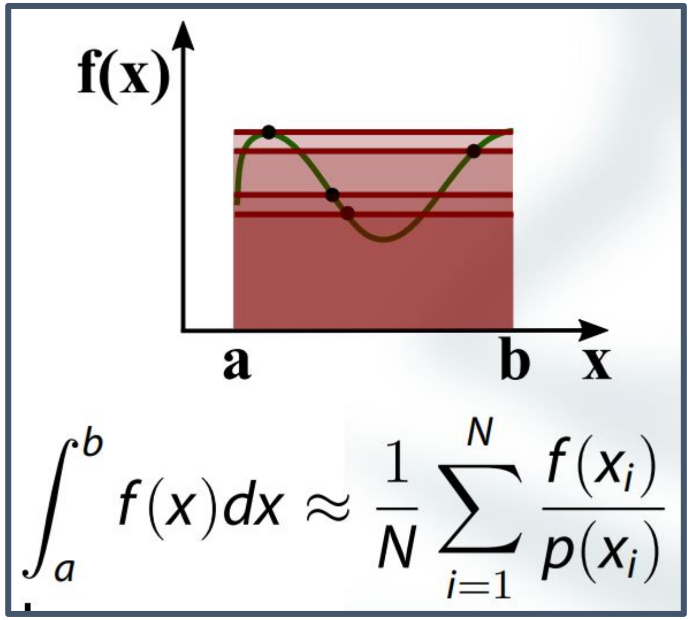
$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$

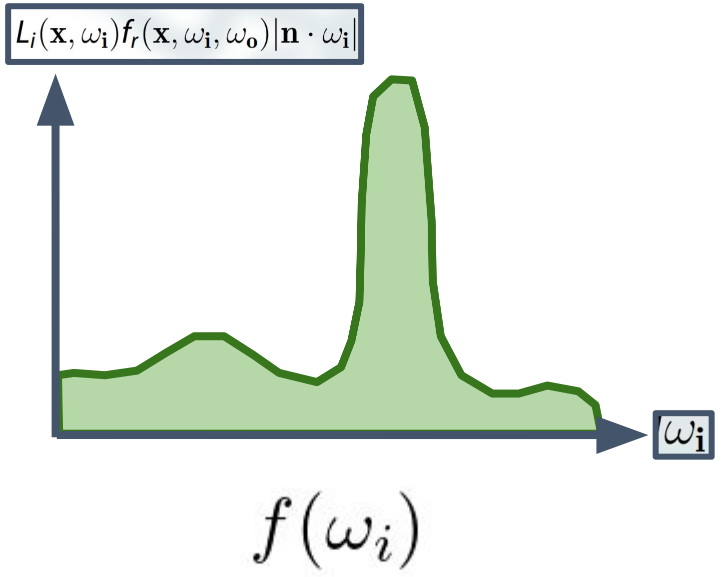


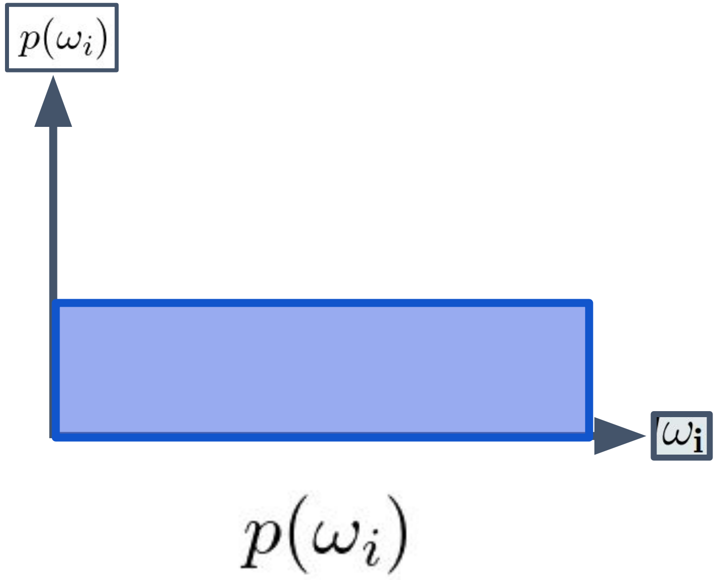
$f(\omega_i)$

$p(\omega_i)$

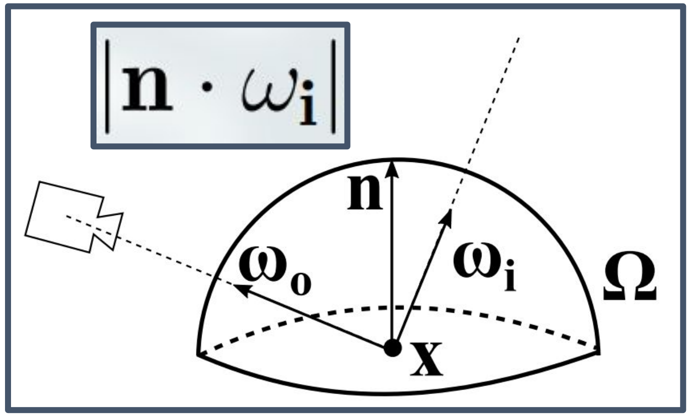
Importance sampling



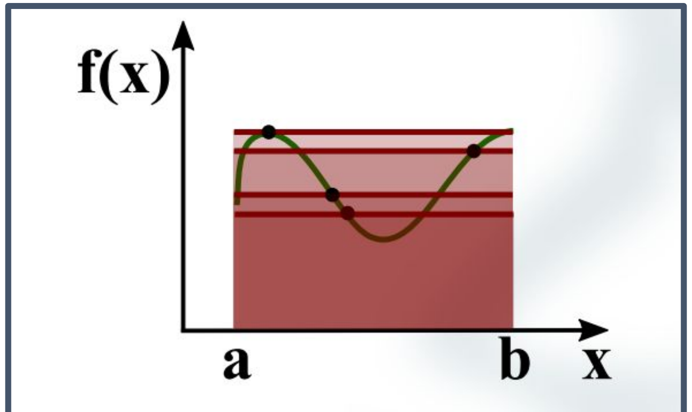




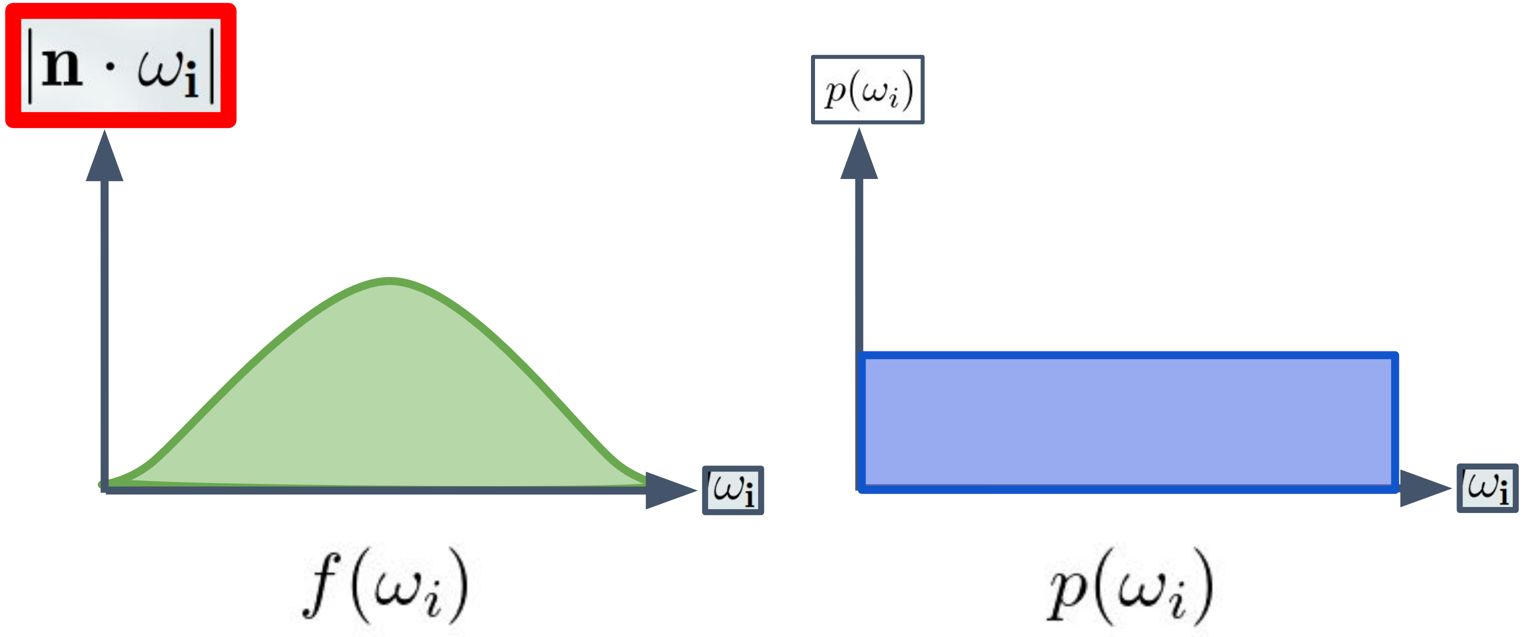
- Monte Carlo works better when *f* and *p* have similar distributions

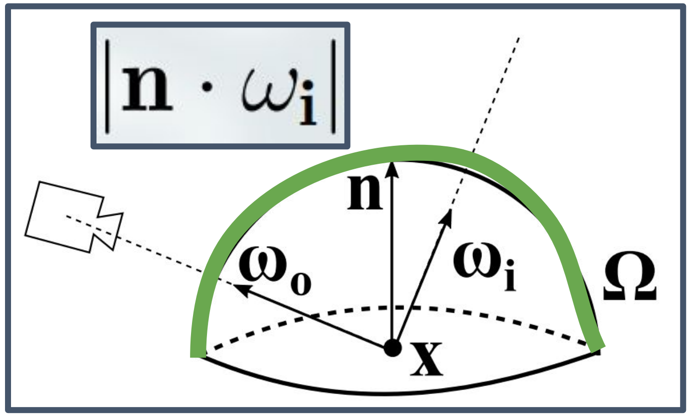


Importance sampling: diffuse materials



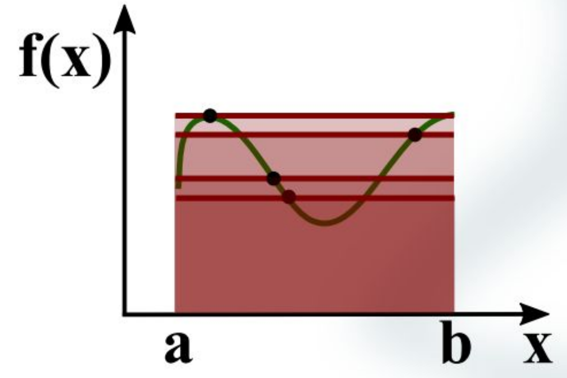
$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$



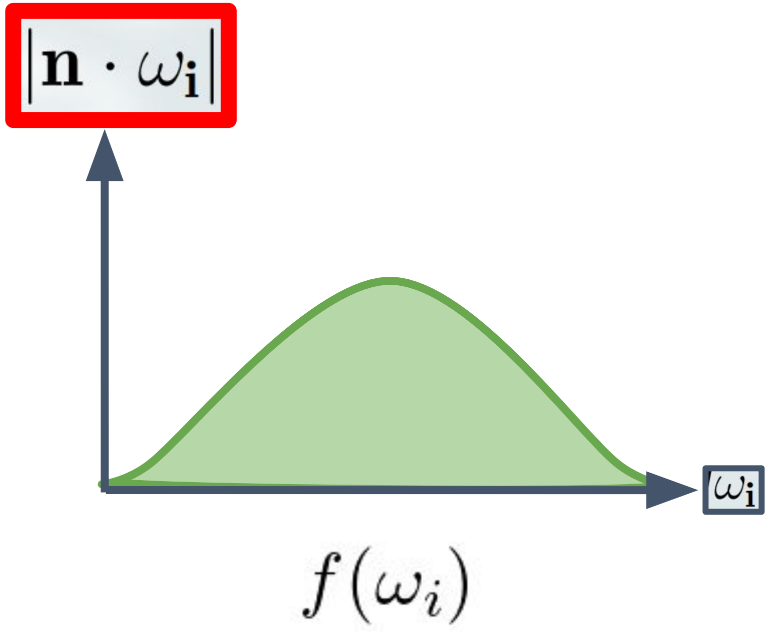


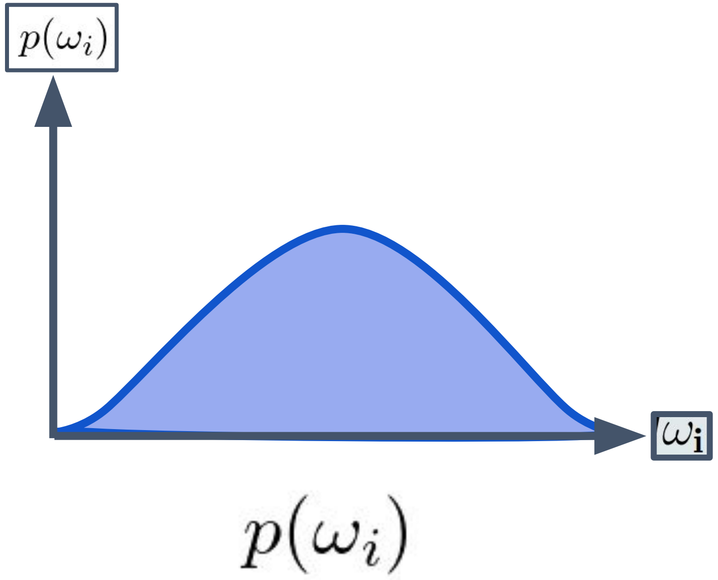
- Monte Carlo works better when *f* and *p* have similar distributions

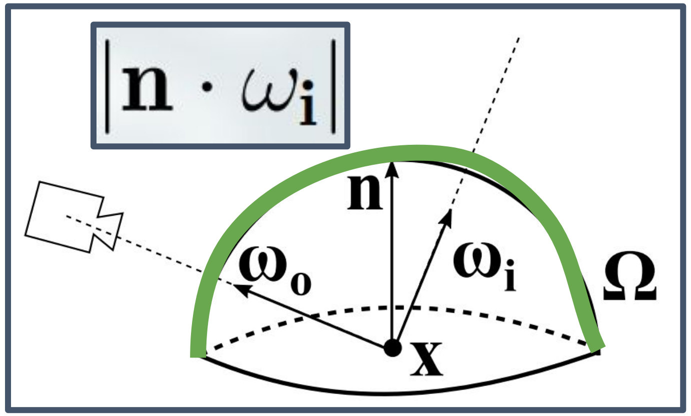
Importance sampling: diffuse materials



$$\int_a^b f(x)dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$$







- Monte Carlo works better when *f* and *p* have similar distributions
- Make *p* follow $|\mathbf{n}\cdot\omega_\mathbf{i}|$
- Cosine-weighted

$$c^{-1}(\xi_{\theta_i}) = \arccos\sqrt{1-\xi_{\theta_i}}$$

$$c^{-1}(\xi_{\phi_i}) = 2\pi\xi_{\phi_i}$$

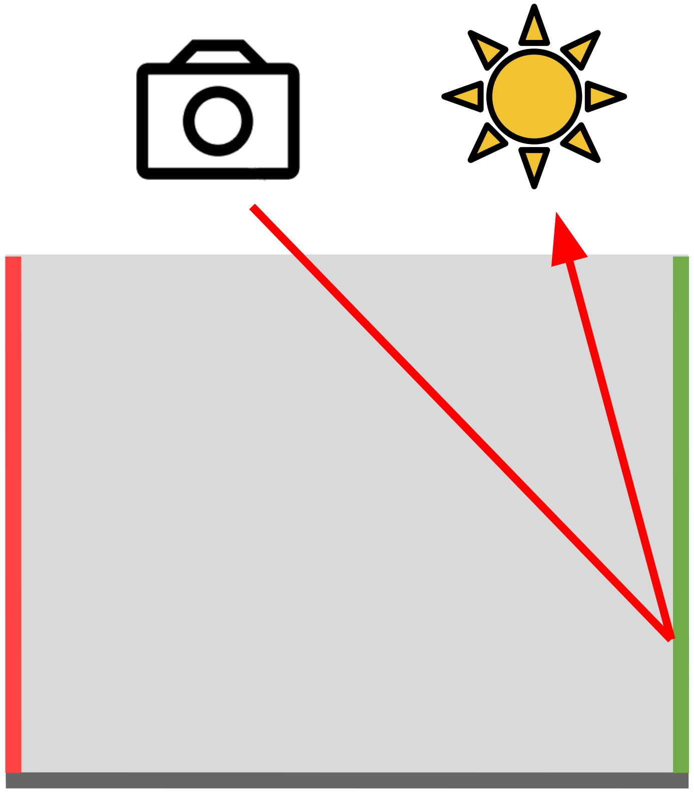
Programming recommendations

Graphics and Imaging Lab

- Enough maths, let’s code

Programming recommendations

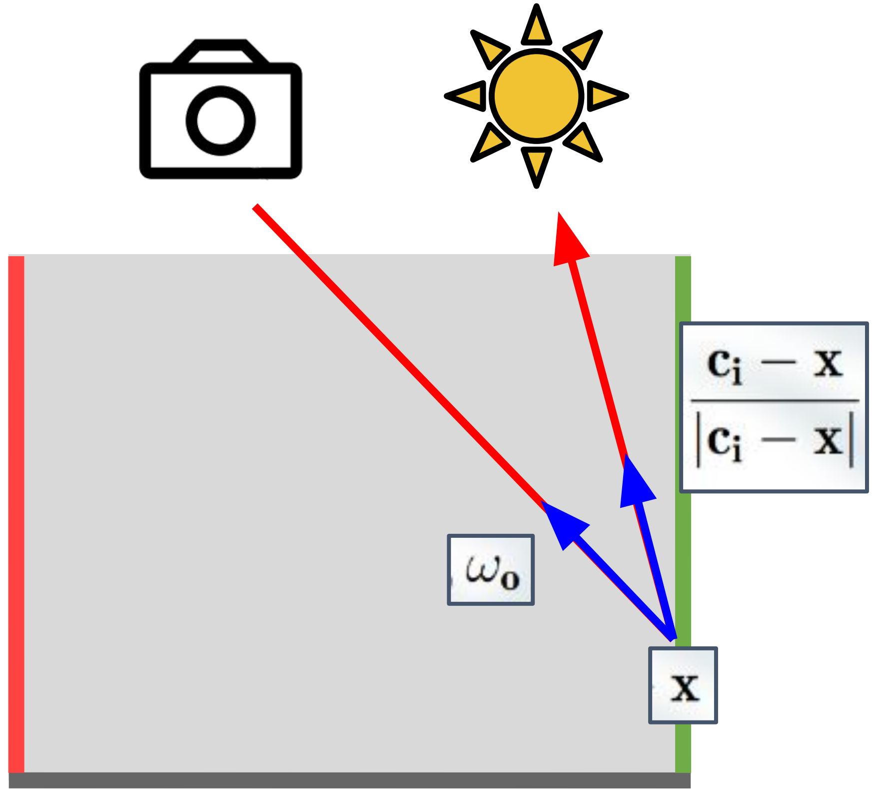
- Enough maths, let's code
- **Recommendation:** you can separate your code in functions



Programming recommendations

- Enough maths, let's code
- **Recommendation:** you can separate your code in functions
  - From previous session
    - Diffuse **BRDF evaluation** function

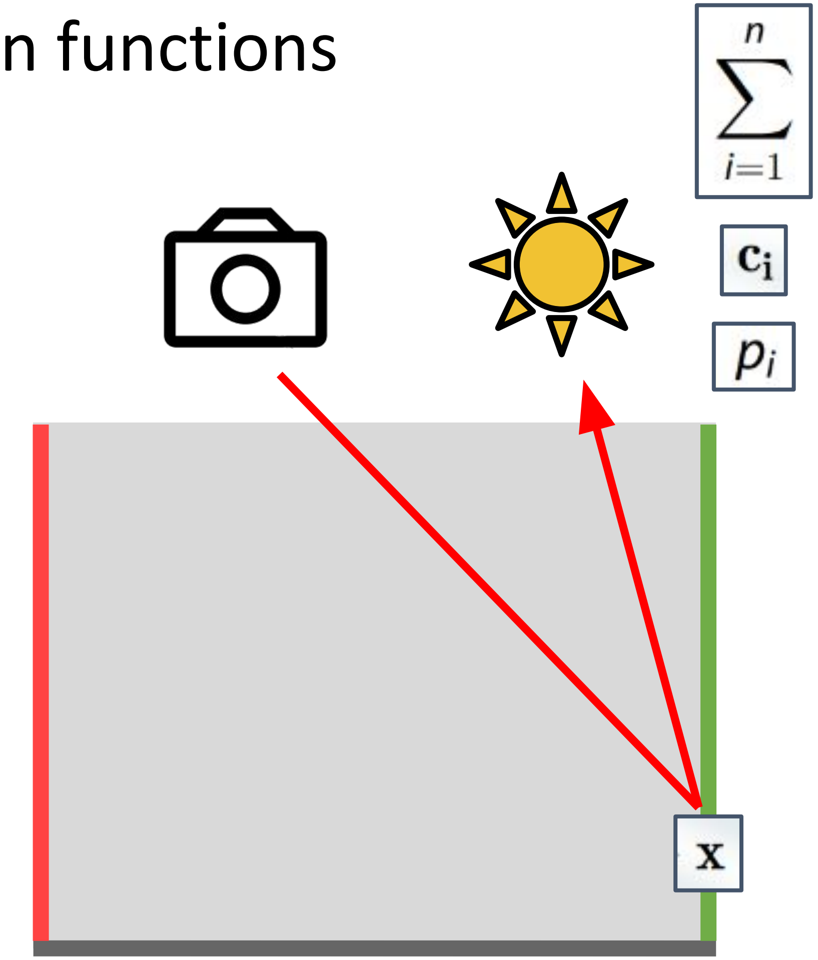
$$\boxed{f_r\left(\mathbf{x}, \frac{\mathbf{c_i} - \mathbf{x}}{|\mathbf{c_i} - \mathbf{x}|}, \omega_\mathbf{o}\right)}$$



Programming recommendations

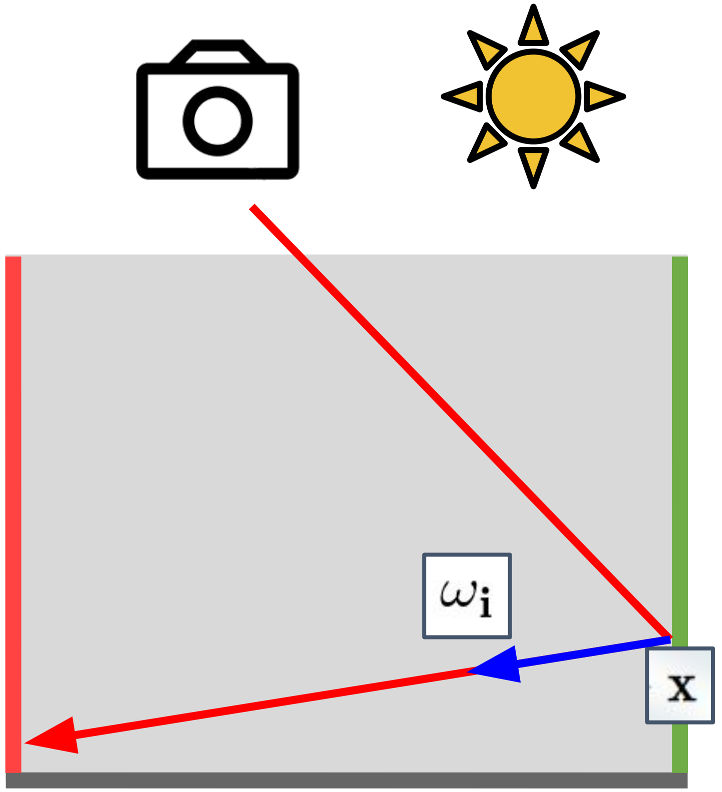
- Enough maths, let's code
- **Recommendation:** you can separate your code in functions
  - From previous session
    - Diffuse **BRDF evaluation** function
    - Next-event estimation (direct lights at $\boxed{\mathbf{x}}$)

$$\sum_{i=1}^{n} \frac{p_i}{|\mathbf{c_i} - \mathbf{x}|^2} f_r\left(\mathbf{x}, \frac{\mathbf{c_i} - \mathbf{x}}{|\mathbf{c_i} - \mathbf{x}|}, \omega_o\right) \left|\mathbf{n} \cdot \frac{\mathbf{c_i} - \mathbf{x}}{|\mathbf{c_i} - \mathbf{x}|}\right|$$



Programming recommendations

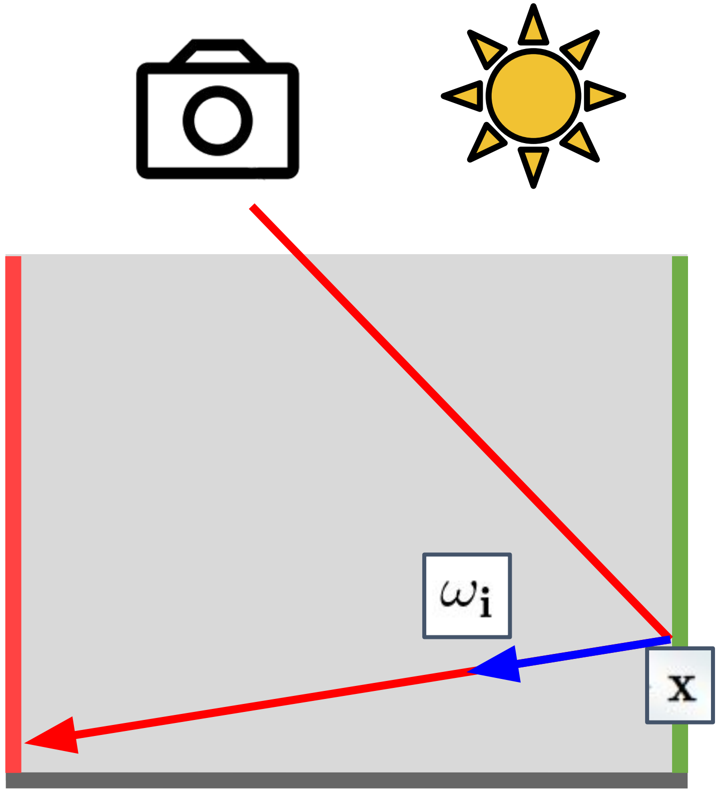
- Enough maths, let's code
- **Recommendation:** you can separate your code in functions
  - From previous session
    - Diffuse **BRDF evaluation** function
    - Next-event estimation (direct lights at $\boxed{\mathbf{x}}$)
  - Today's session
    - Diffuse **BRDF sample** function



$$\begin{matrix} \xi_\theta \in [0,1) \\ \xi_\varphi \in [0,1) \end{matrix} \Rightarrow \begin{matrix} \theta \in [0,\frac{\pi}{2}) \\ \varphi \in [0,2\pi) \end{matrix} \Rightarrow \omega_i$$

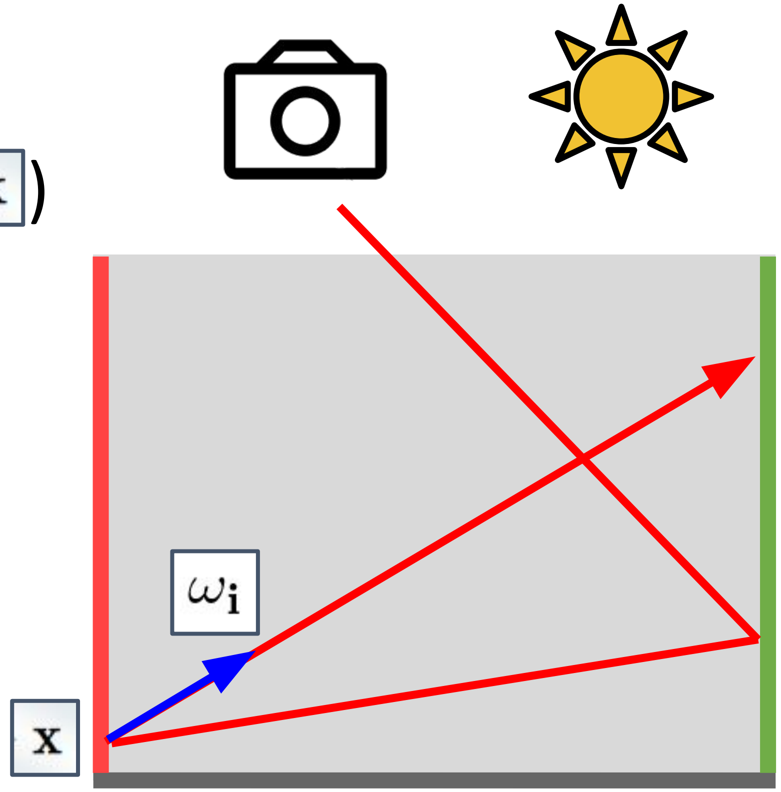
Programming recommendations

- Enough maths, let's code
- **Recommendation:** you can separate your code in functions
  - From previous session
    - Diffuse **BRDF evaluation** function
    - Next-event estimation (direct lights at $\boxed{\mathbf{x}}$)
  - Today's session
    - Diffuse **BRDF sample** function
      - Return $\boxed{\omega_\mathbf{i}}$ and $\boxed{f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o})}$
      - Call it on every bounce
      - Intersect ray ($\boxed{\mathbf{x}}$, $\boxed{\omega_\mathbf{i}}$) with geometry



Programming recommendations

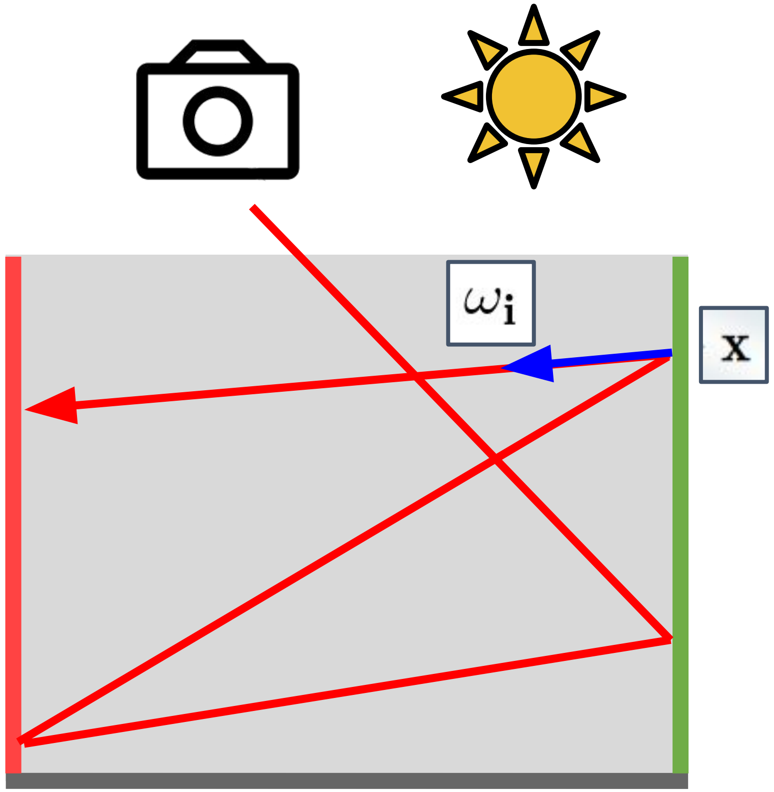
- Enough maths, let's code
- **Recommendation:** you can separate your code in functions
  - From previous session
    - Diffuse **BRDF evaluation** function
    - Next-event estimation (direct lights at $\boxed{\mathbf{x}}$)
  - Today's session
    - Diffuse **BRDF sample** function
      - Return $\boxed{\omega_\mathbf{i}}$ and $\boxed{f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o})}$
      - Call it on every bounce
      - Intersect ray ($\boxed{\mathbf{x}}$, $\boxed{\omega_\mathbf{i}}$) with geometry



Programming recommendations

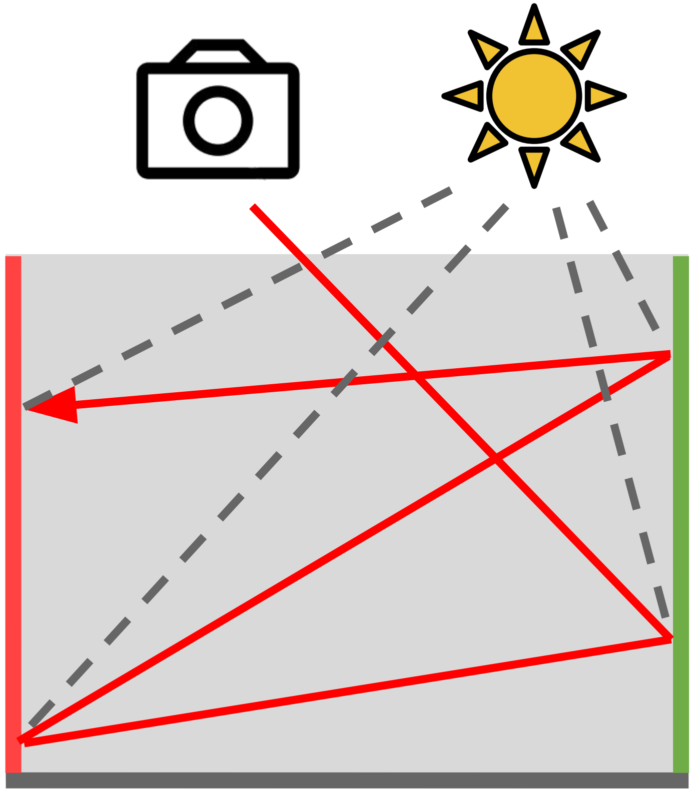
- Enough maths, let's code
- **Recommendation:** you can separate your code in functions
  - From previous session
    - Diffuse **BRDF evaluation** function
    - Next-event estimation (direct lights at $\boxed{\mathbf{x}}$)
  - Today's session
    - Diffuse **BRDF sample** function

Problem: it is **impossible** to hit a point light with a random ray ($\boxed{\mathbf{x}}$, $\boxed{\omega_i}$)



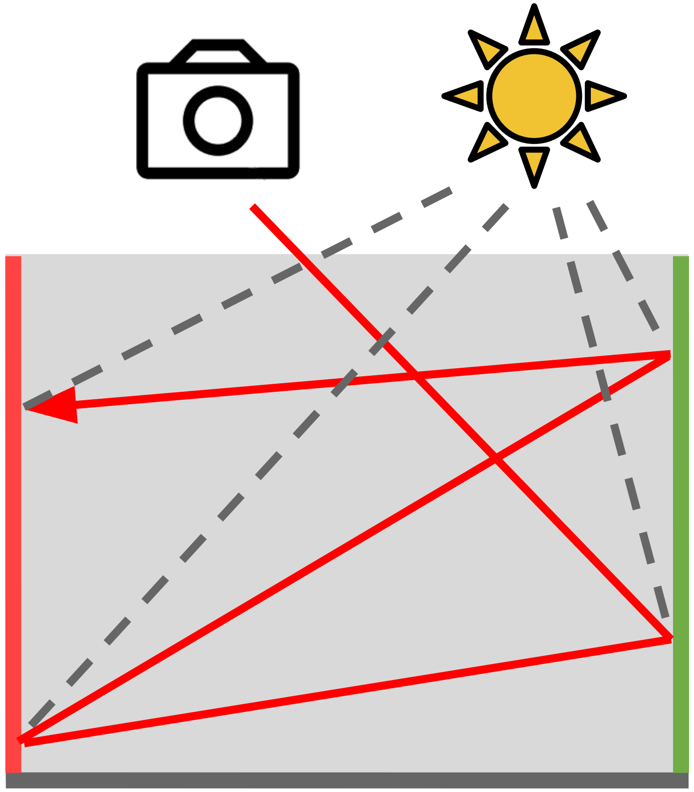
Programming recommendations

- Enough maths, let's code
- **Recommendation:** you can separate your code in functions
  - From previous session
    - Diffuse **BRDF evaluation** function
    - Next-event estimation (direct lights at $\boxed{\mathbf{x}}$)
  - Today's session
    - Diffuse **BRDF sample** function
      - Generate $\boxed{\omega_\mathrm{i}}$ and intersect ray
    - Call next-event estimation on every bounce $\boxed{\mathbf{x}}$



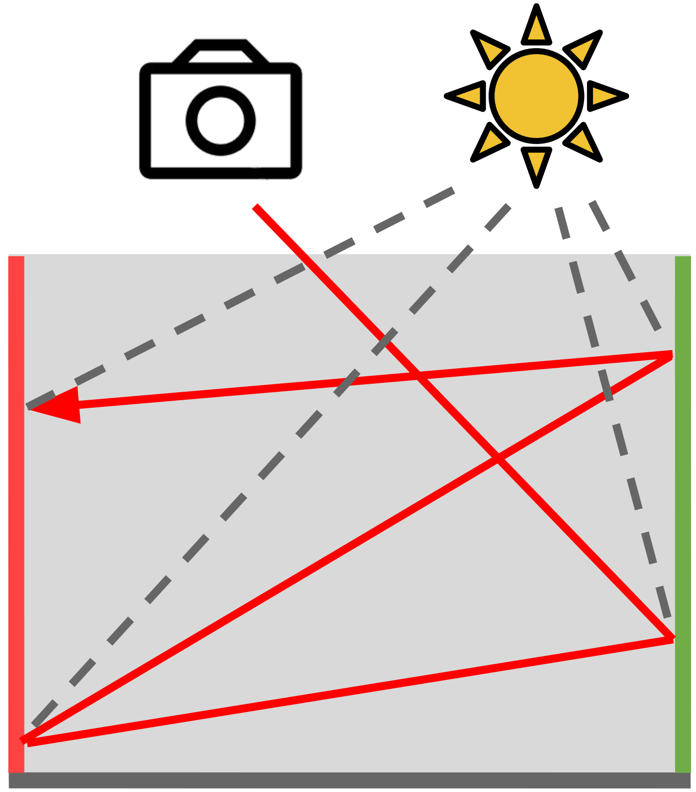
Programming recommendations

- Path tracing algorithm: Want to calculate $\boxed{L_o(\mathbf{x}, \omega_\mathbf{o})}$
  - Intersect ray with geometry at point $\boxed{\mathbf{x}}$ with its own BRDF
  - Search for light sources at $\boxed{\mathbf{x}}$
  - Sample BRDF $\boxed{\mathbf{x}}$ for a new ray ($\boxed{\mathbf{x}}$, $\boxed{\omega_\mathbf{i}}$)
  - Repeat the same steps again



Programming recommendations

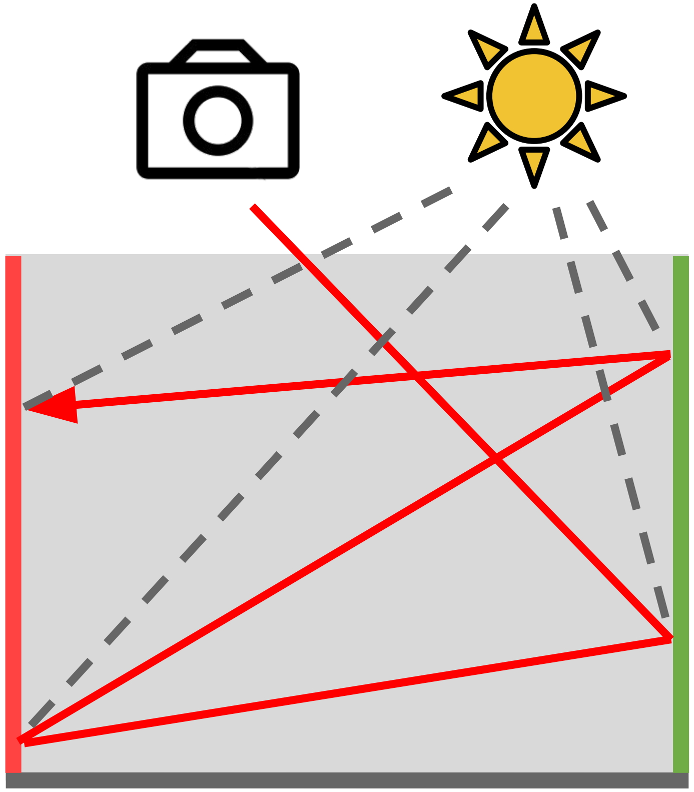
- Path tracing algorithm: Want to calculate $\boxed{L_o(\mathbf{x}, \omega_\mathbf{o})}$
  - Intersect ray with geometry at point $\boxed{\mathbf{x}}$ with its own BRDF
  - Search for light sources at $\boxed{\mathbf{x}}$
  - Sample BRDF $\boxed{\mathbf{x}}$ for a new ray ($\boxed{\mathbf{x}}$, $\boxed{\omega_\mathbf{i}}$)
  - Repeat the same steps again
- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)



Programming recommendations

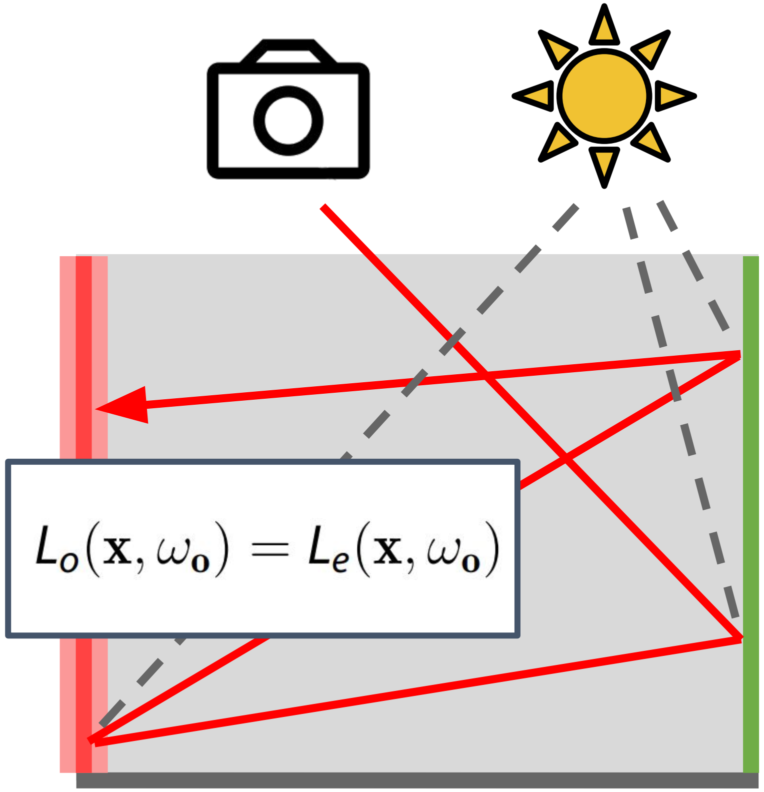
Graphics and Imaging Lab

- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- When do we stop generating rays?
  (aka. path termination conditions)



Programming recommendations

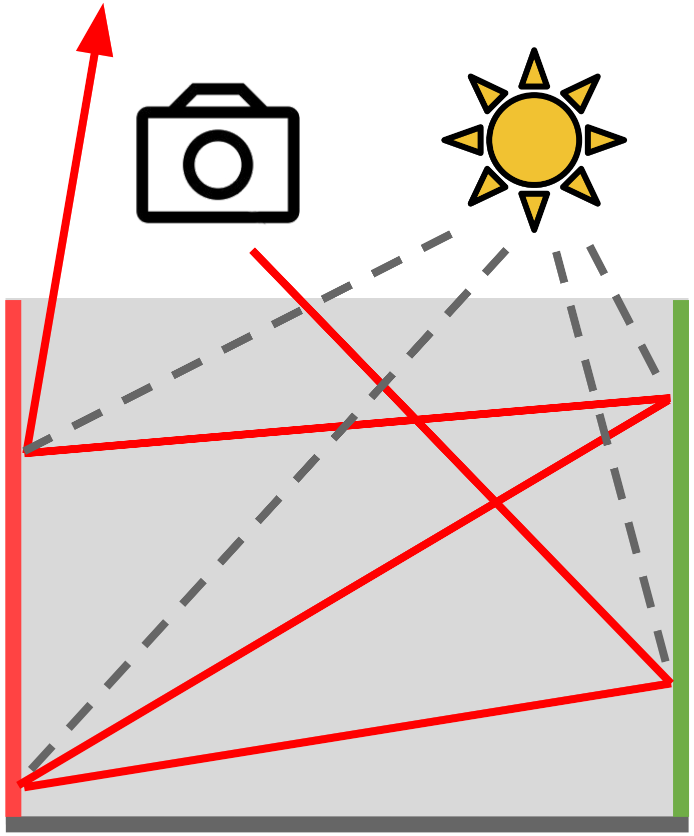
- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- When do we stop generating rays?
  (aka. path termination conditions)
  - **(1) When hitting an area light source**



Programming recommendations

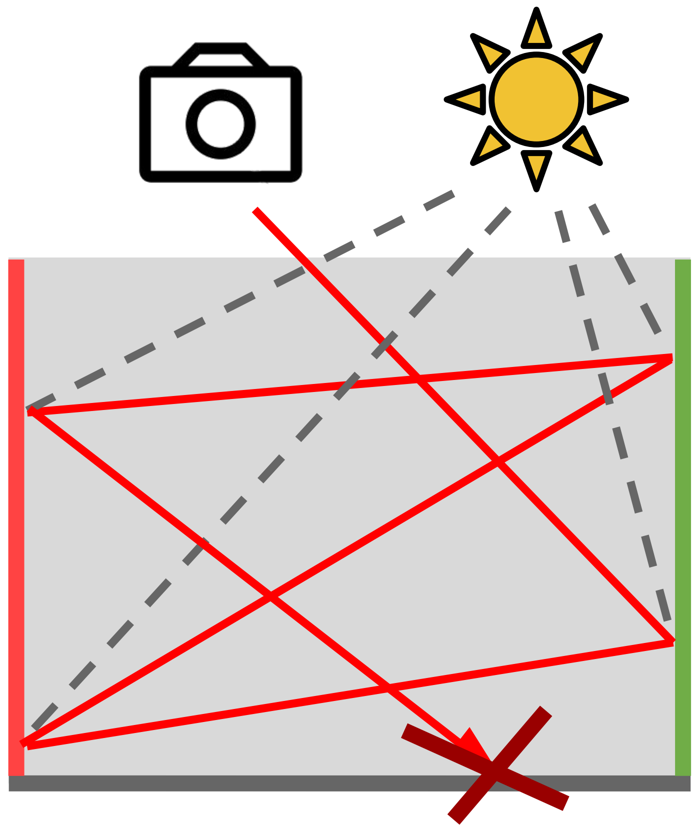
- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- When do we stop generating rays?
  (aka. path termination conditions)
  - (1) When hitting an area light source
  - **(2) When ray does not hit anything**

$$L_o(\mathbf{x}, \omega_\mathbf{o}) = 0$$



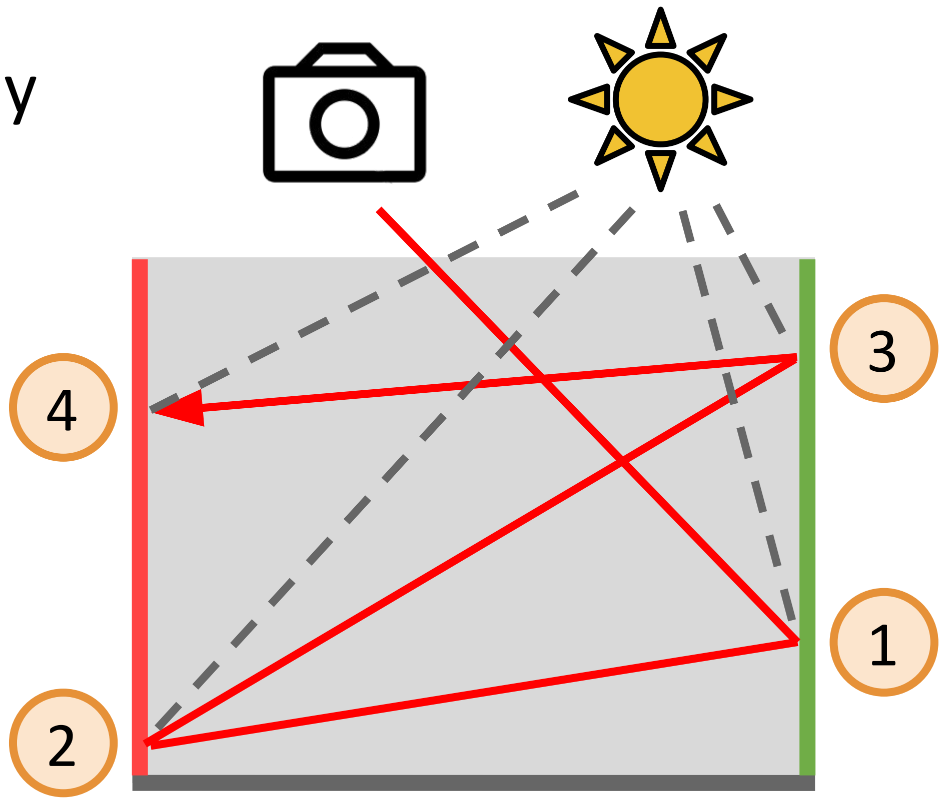
Programming recommendations

- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- When do we stop generating rays?
  (aka. path termination conditions)
  - (1) When hitting an area light source
  - (2) When ray does not hit anything
  - **(3) When there are >N bounces**



Programming recommendations

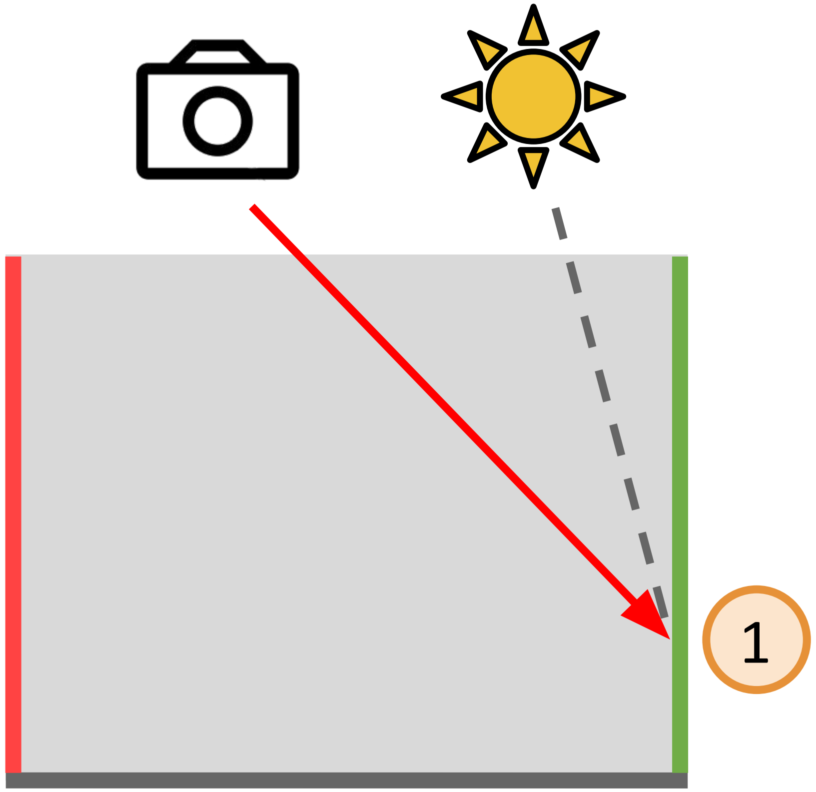
- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- Important: account for light bounces correctly
  - You should **sum** four contributions



Programming recommendations

- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- Important: account for light bounces correctly
  - In this example, you should **sum** four contributions
  - Light at 1 should be multiplied by BRDF/cosine terms at 1

$$\boxed{f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o})|\mathbf{n} \cdot \omega_\mathbf{i}|}$$

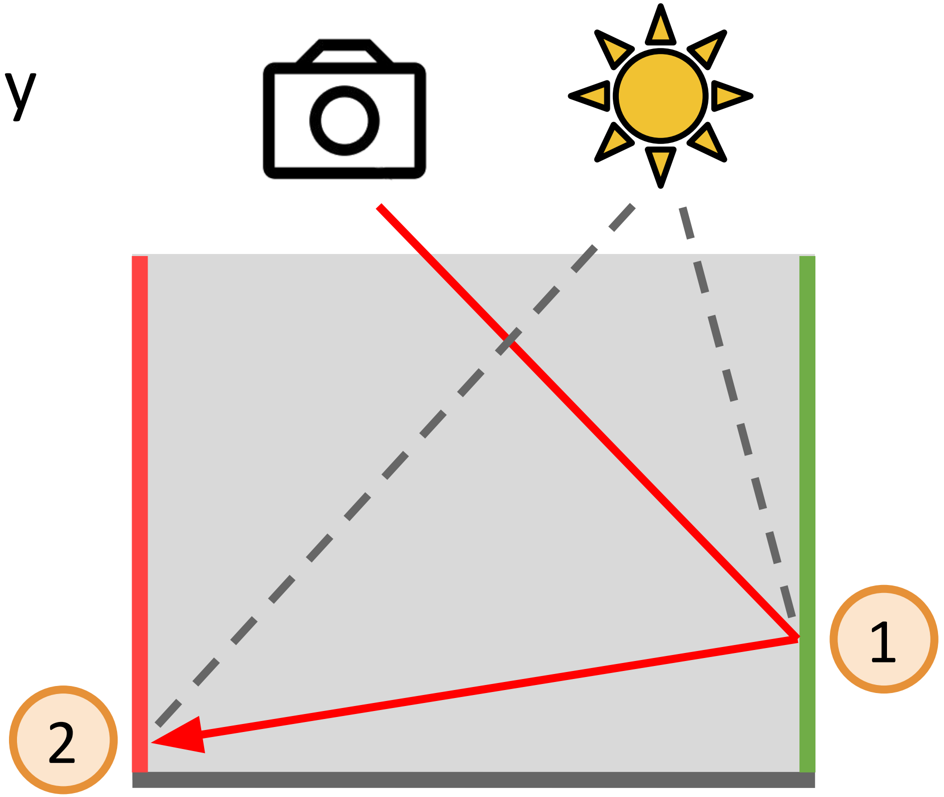


Programming recommendations

Graphics and Imaging Lab

- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- Important: account for light bounces correctly
  - In this example, you should **sum** four contributions
  - Light at 2 should be multiplied by BRDF/cosine terms at 2 1

$$f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o})|\mathbf{n} \cdot \omega_\mathbf{i}|$$

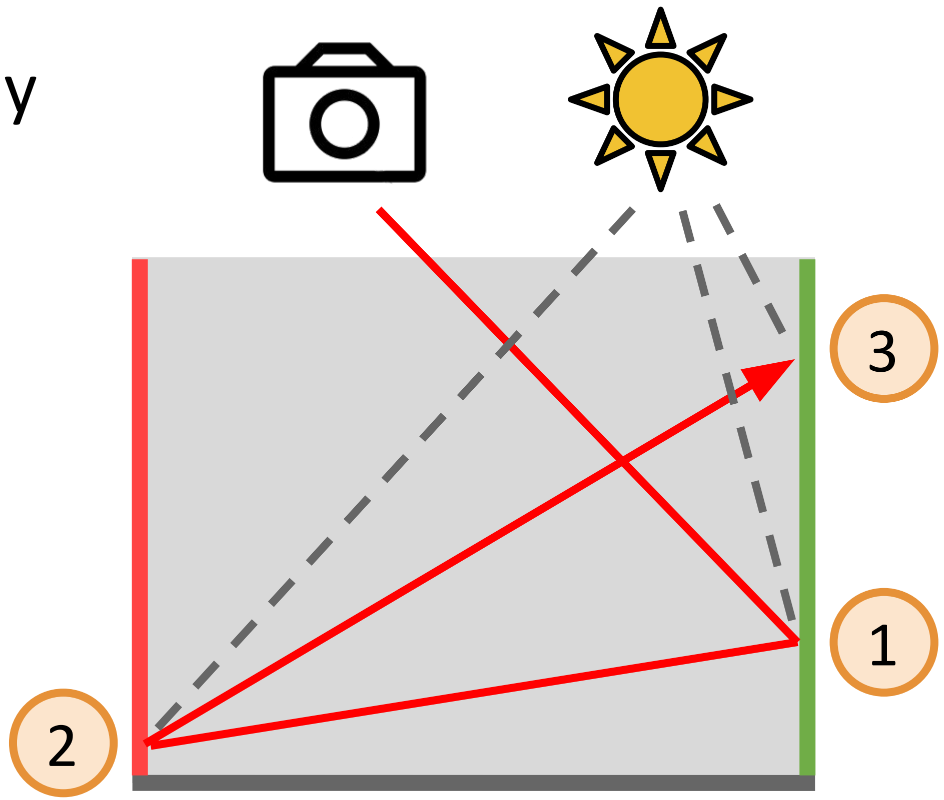


Programming recommendations

Graphics and Imaging Lab

- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- Important: account for light bounces correctly
  - In this example, you should **sum** four contributions
  - Light at 3 should be multiplied by BRDF/cosine terms at 3 2 1

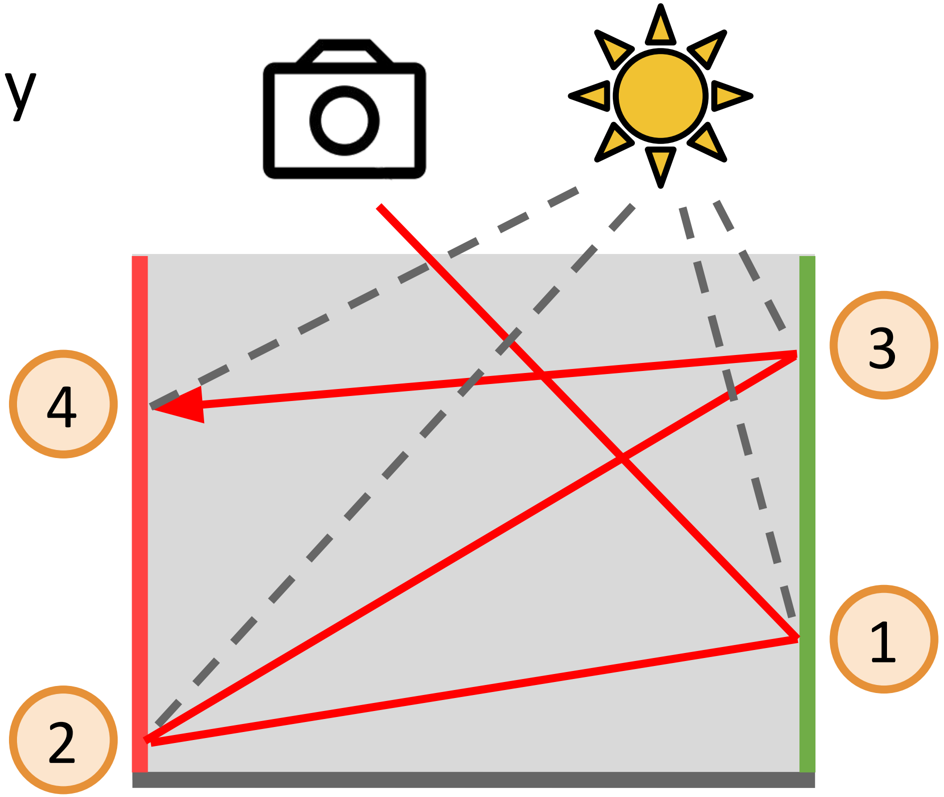
$$f_r(\mathbf{x}, \omega_\mathbf{i}, \omega_\mathbf{o})|\mathbf{n} \cdot \omega_\mathbf{i}|$$



Programming recommendations

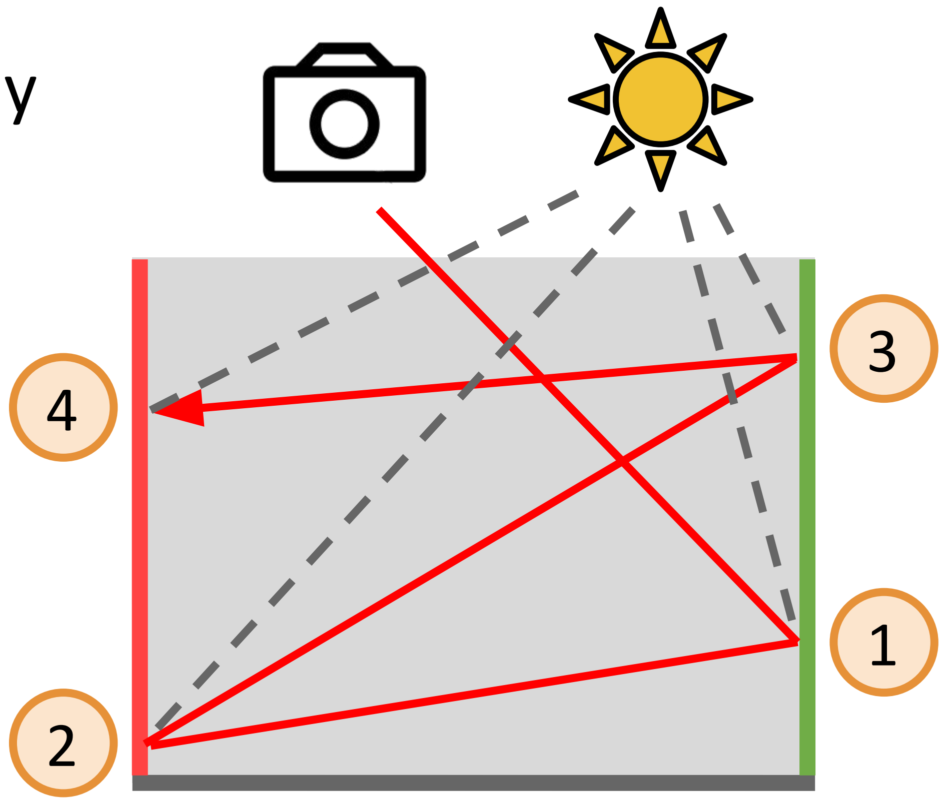
- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- Important: account for light bounces correctly
  - In this example, you should **sum** four contributions
  - Light at 4 should be multiplied by BRDF/cosine terms at 4 3 2 1

$$f_r(\mathbf{x}, \omega_{\mathbf{i}}, \omega_{\mathbf{o}})|\mathbf{n} \cdot \omega_{\mathbf{i}}|$$



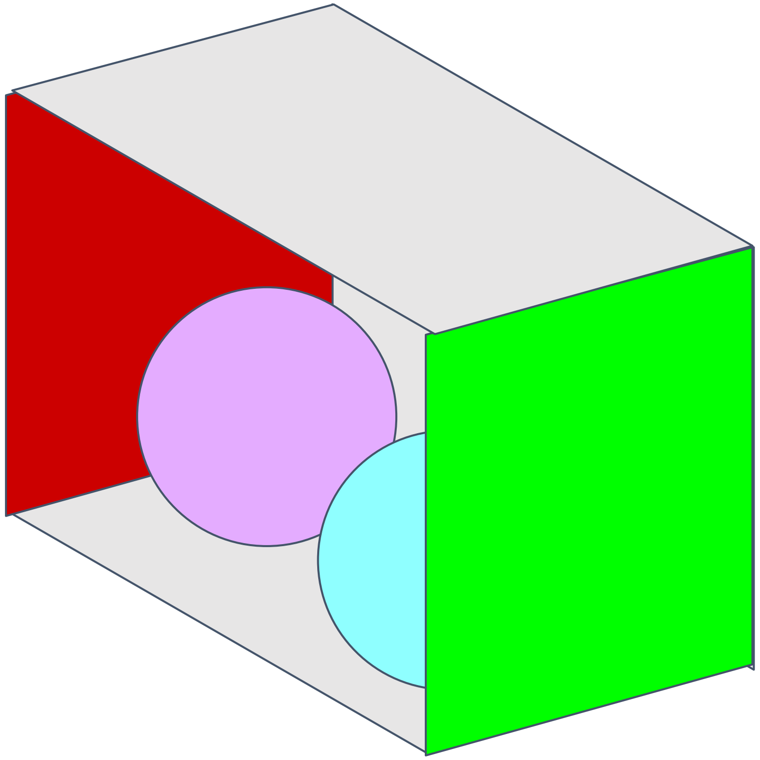
Programming recommendations

- Function can be written as:
  - Recursive (recommended)
  - Iterative (faster, caution)
- Important: account for light bounces correctly
  - In this example, you should **sum** four contributions
  - Total: 1 + 2 + 3 + 4
    - Easy to do recursively
    - Tricky to do iteratively



Example scene: Cornell Box

- **Geometry**



Planes defined by normal (n) and distance (d)

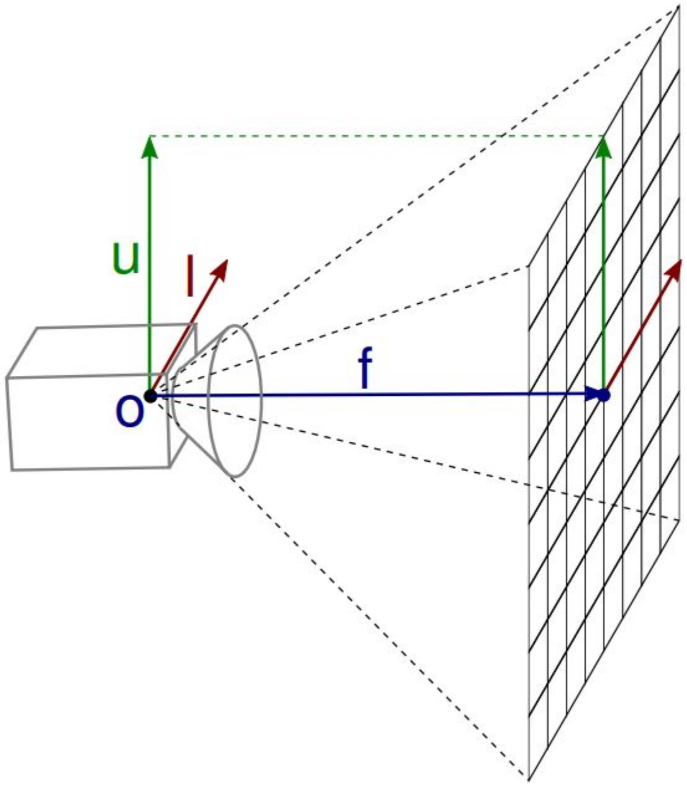
Left plane	n = (1, 0, 0), d = 1
Right plane	n = (-1, 0, 0), d = 1
Floor plane	n = (0, 1, 0), d = 1
Ceiling plane	n = (0, -1, 0), d = 1
Back plane	n = (0, 0, -1), d = 1

Spheres defined by center (c) and radius (r)

Left sphere	c = (-0.5, -0.7, 0.25), r = 0.3
Right sphere	c = (0.5, -0.7, -0.25), r = 0.3

Example scene: Cornell Box

- **Camera & light sources**

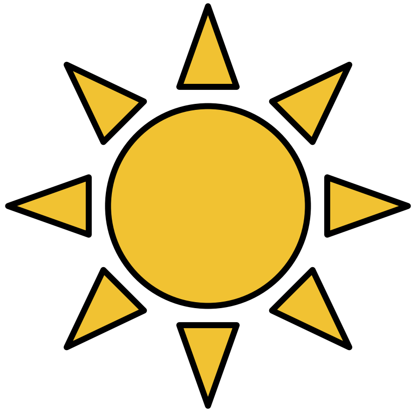


Camera and image plane defined by

Origin	O = (0, 0, -3.5)
Left	L = (-1, 0, 0)
Up	U = (0, 1, 0)
Forward	F = (0, 0, 3)
Size	256x256 pixels

Example scene: Cornell Box

- **Light sources**



Center and power (emission)

Center c = (0, 0.5, 0)

Power can be any number e.g. p = (1, 1, 1)

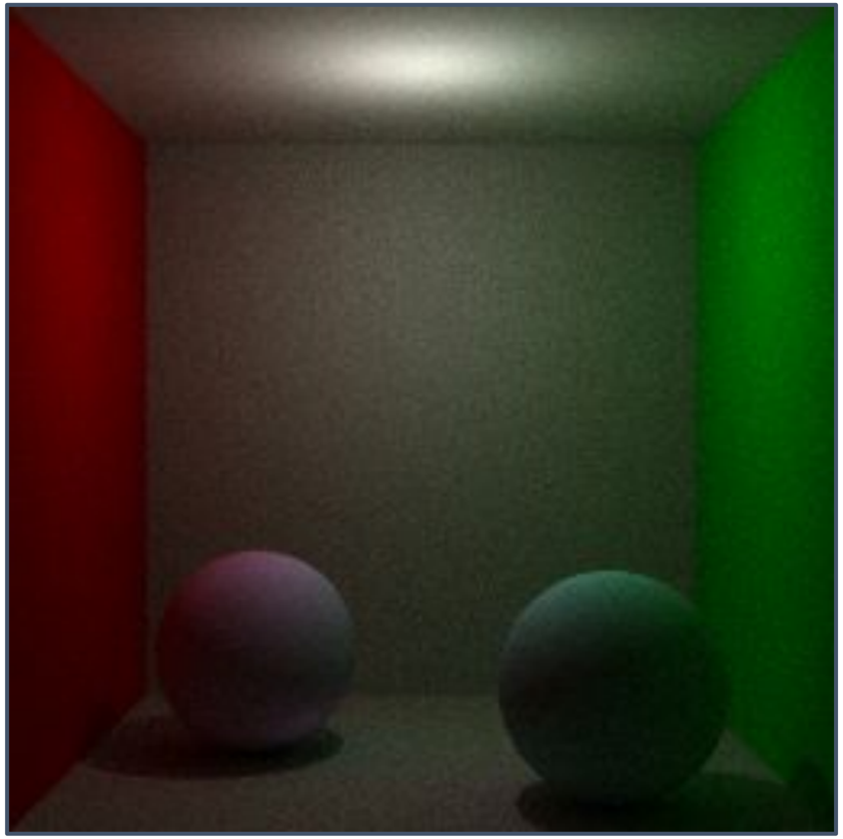
Just be careful with the #MAX

```
P3
# feep.ppm
#MAX=<maximum of your RGB memory values>
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
```

Example scene: Cornell Box

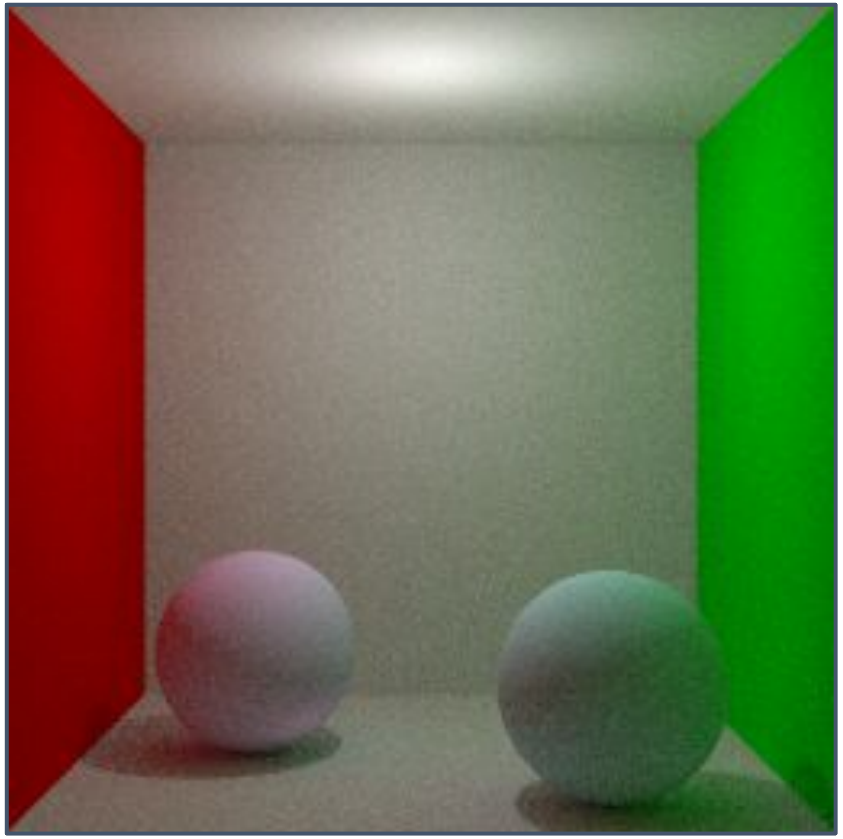
Graphics and Imaging Lab

- **Results (no area lights + point light)**



Using a point light

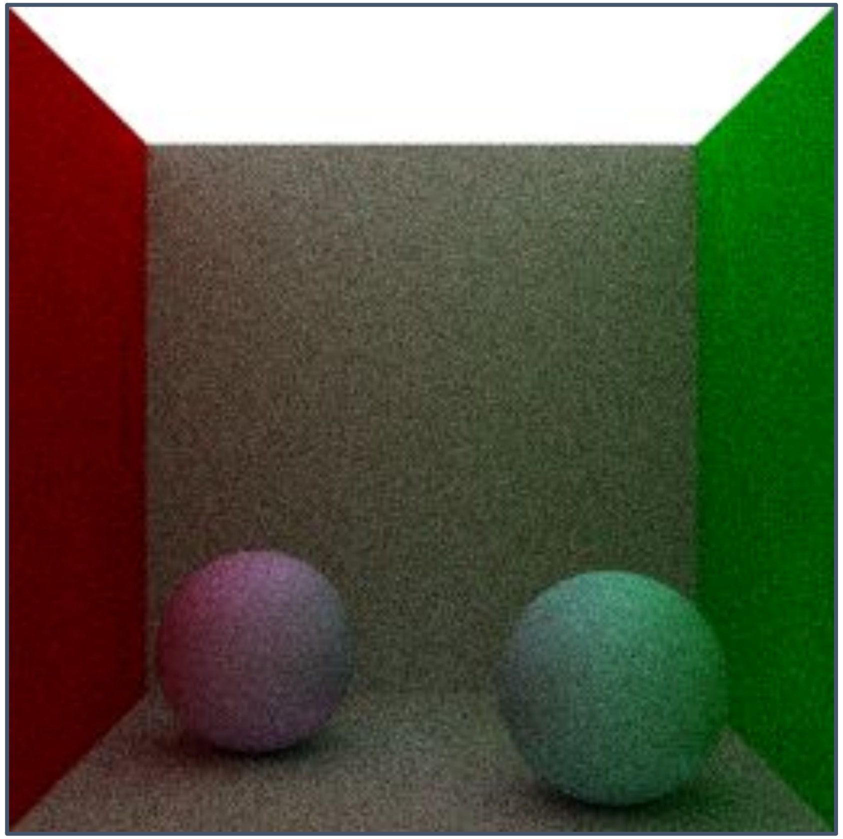
Equalization
Gamma = 2.2



With tone mapping

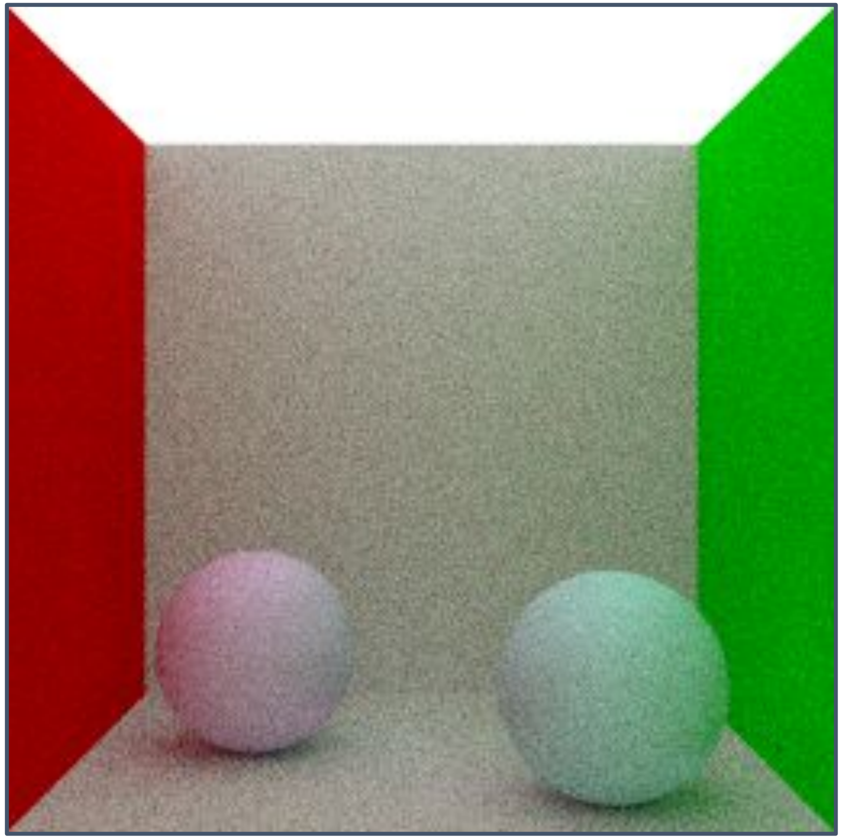
Example scene: Cornell Box

- **Results (no point light + ceiling plane is an area light)**



Using an area light

Equalization
Gamma = 2.2



With tone mapping

Questions

Graphics and Imaging Lab

DO ASK questions, either now or after the lab

But be reasonable, please :)

pluesia@unizar.es | dsubias@unizar.es | o.pueyo@unizar.es

What to expect from this session

Graphics and Imaging Lab

In the programming language of your choice implement:

- Diffuse BRDF (k_d for each material)
    - **“Evaluate” function:** Return $f(\mathbf{x}, \omega_i, \omega_o)$ (you should have already programmed this)
    - **“Sample” function:** Return random direction ω_i and $f(\mathbf{x}, \omega_i, \omega_o)$
- Find point light sources using **next-event estimation** on each bounce
- Terminate paths when: (1) no intersection (2) area light is hit (3) >N bounces
- Recommended deadline: November 13th (moodle: January 11th)
    - Extensions (do not count towards recommended deadline):
        - **Textures:** make diffuse coefficient k_d epend on hit position $\mathrm{k}_d(\mathbf{x})$
        - **Fresnel effects:** make diffuse coefficient k_d epend on viewing direction $\mathrm{k}_d(\omega_\mathbf{o})$
        - **Parallelization:** divide work between several threads, estimate time to finish execution
        - **More:** importance sampling next-event estimation, etc. (see the lab assignment or ask us)