Monte-Carlo Path tracing

Today

- Solving integrals (approximately)
  - What did ray-tracing do?
  - Analytic vs. Numeric
  - Monte Carlo method
- Solving the RE using Monte Carlo
- Variance reduction
  - Good Sampling patterns
  - Importance sampling

Why is this hard to solve?

- It involves an integral with no analytic solution
- It is an integral equation, so the RHS contains the LHS in an integrand

(Spherical) integration

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + \int f_r(\mathbf{x}, \omega_i, \omega)\, L(\mathbf{y}, -\omega_i) \cos \theta_i \, d\omega_i$$

Same thing!

How ray-tracing solves the RE

- Many got suspicious about ray-tracing
- Example: Does metal have finite gloss?
  - Yes, cause otherwise I cant see highlight!
  - No, reflections are not blurry in steel balls!
- Contradiction!

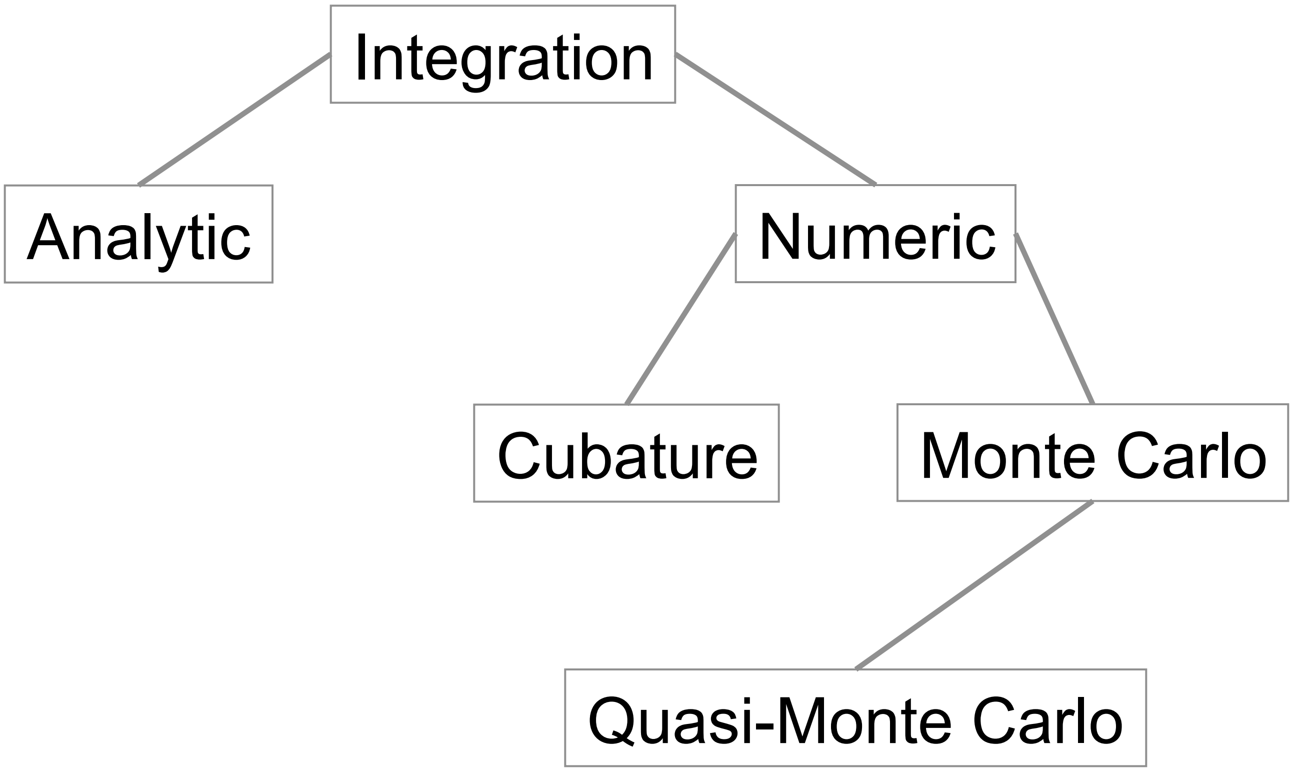
How ray-tracing solves the RE

- The integral is split into a sum of two:
  - A Dirac-paths, solved by binary recusing
  - A path connecting to a point light, can evaluate without recursion

Solving integrals

- Analytic (accurate)
  - Given the function f, try to find F by symbolic manipulation
- Numeric (approximate)
  - Cubature
  - The Monte Carlo method
  - Both are approximate

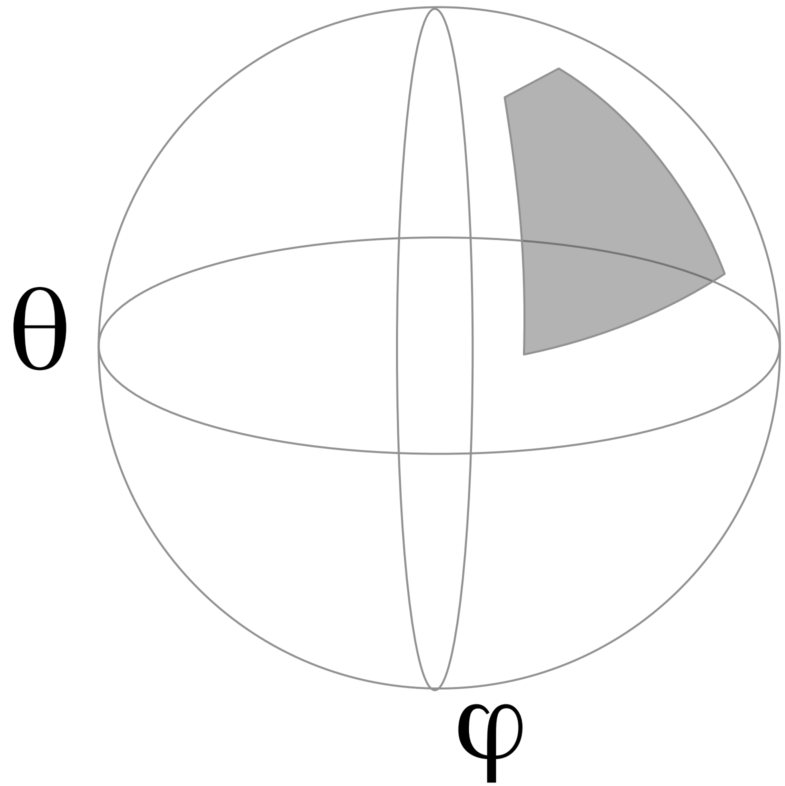
Classfication



Analytic

- Examples
  - $f(x) = \mathrm{x}$, $\quad F(x) = ½\, x^2$
  - $f(x) = \sin(\mathrm{x})$, $\quad F(x) = -\cos(x)$
- Difficult for a function such as we have
  - Impossible, as: The input is not even analytic (what is f for a bunny in the sun?)
  - Difficult, as: Recursion
  - Also difficult: Spherical domain

Some analytic things work



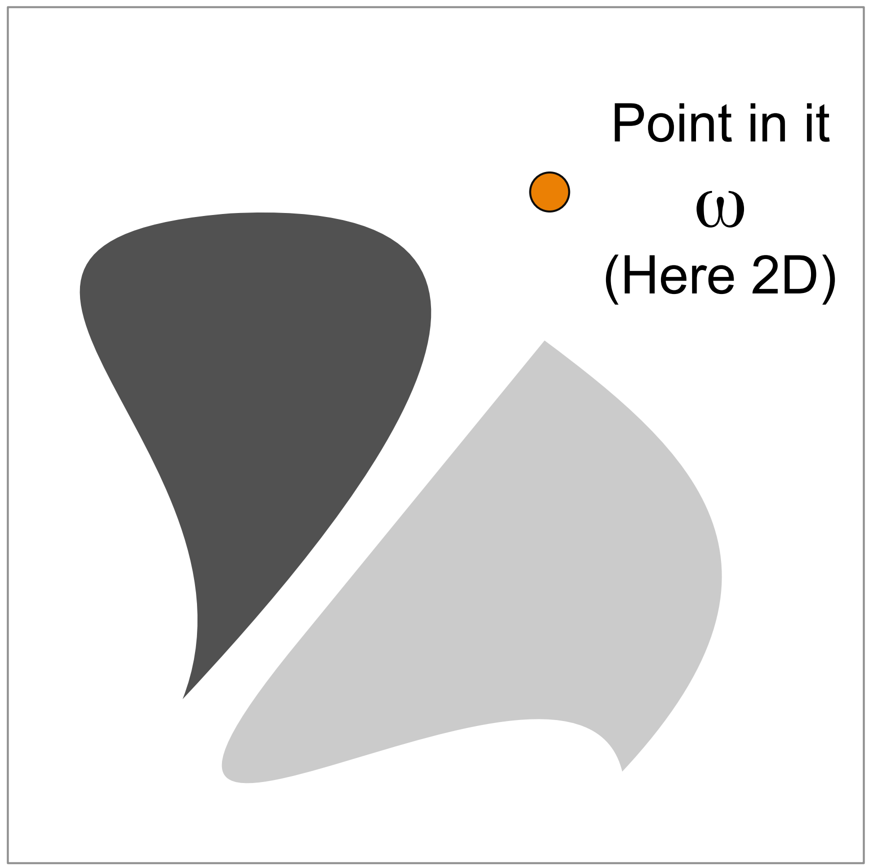
$$f(\boldsymbol{\omega}) = f(\theta, \varphi) = \begin{cases} c \text{ if } 2 < \theta < 3 \text{ and } -1 < \varphi < 1 \\ 0 \text{ otherwise} \end{cases}$$

Forget about spheres for now

Integrand

$f(\omega)$

Dark = low value
Bright = high value



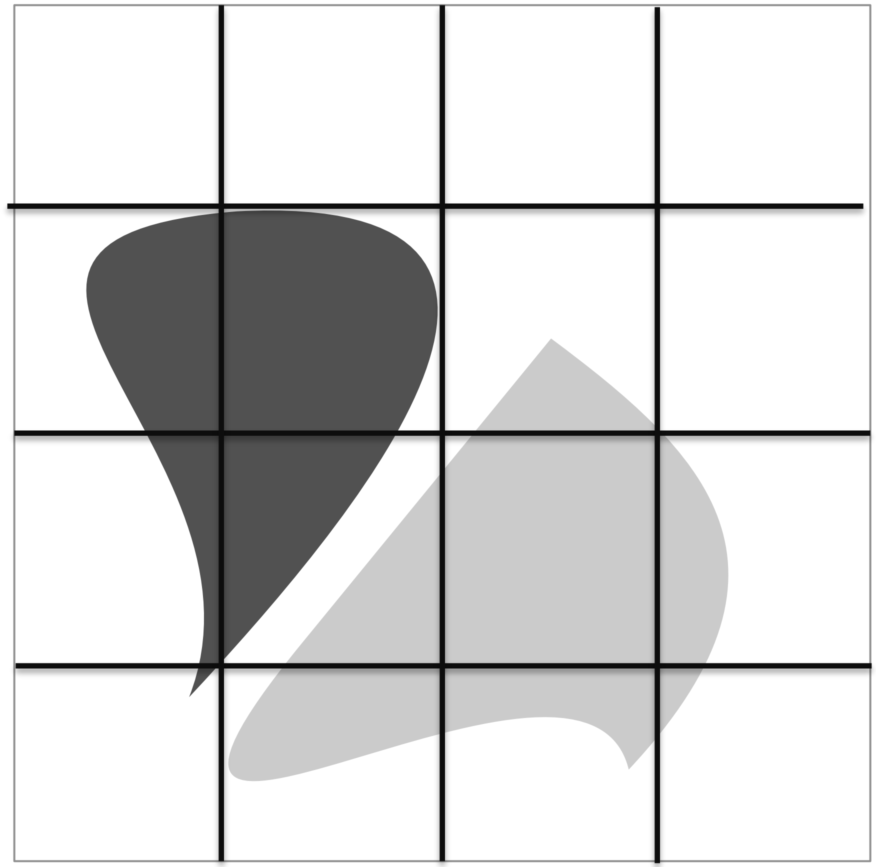
Domain

Ω

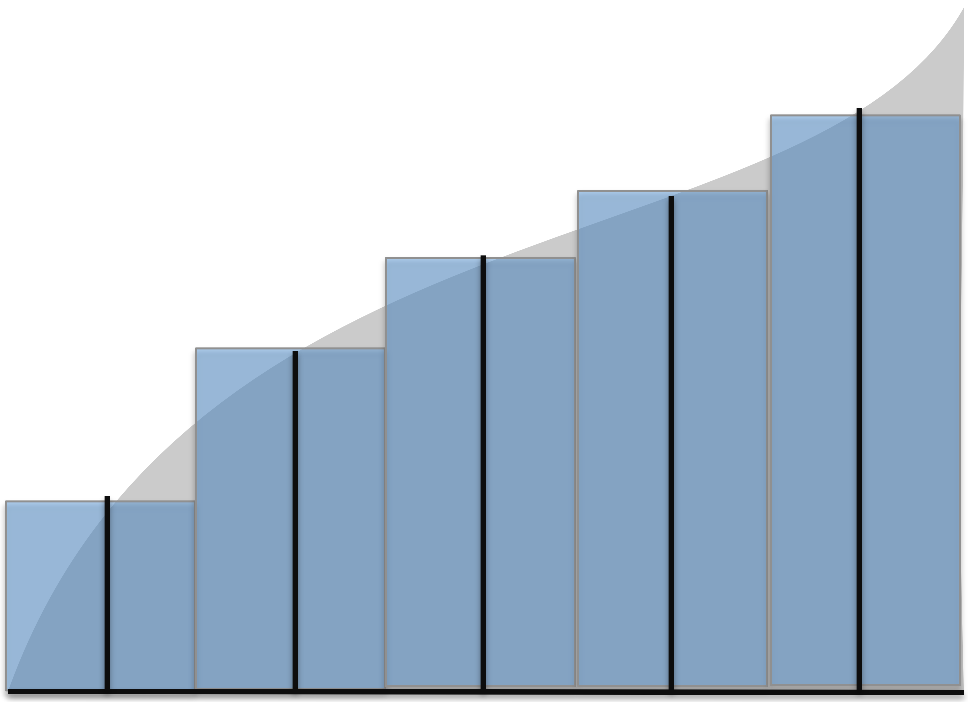
(Here 2D rectangle)

Solution $F = \int f(\omega) d\omega = 0.85$

Numerical integration



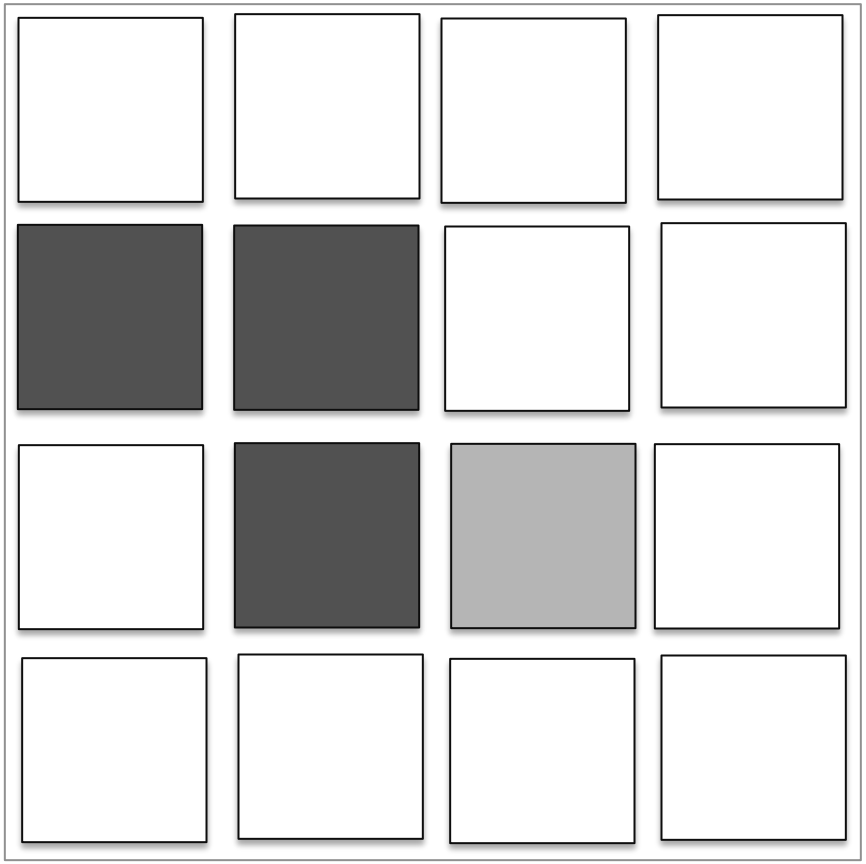
Numerical integration



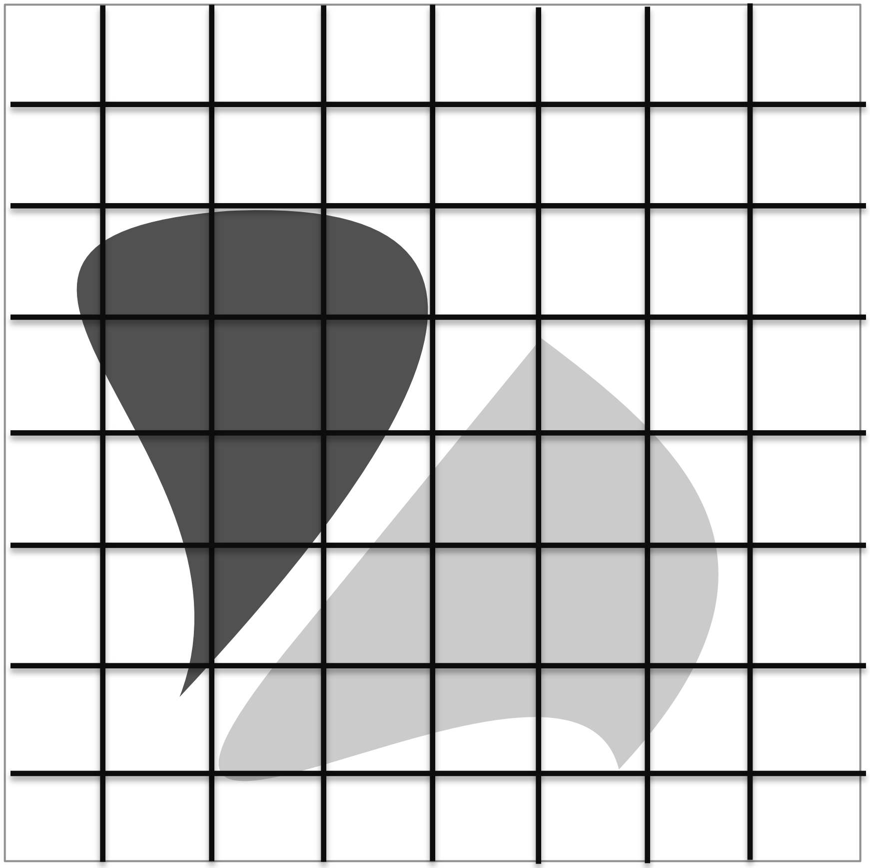
You should recall from high school analysis course.

Numerical integration

1 + 1 + 1 + 1 +
0.2 + 0.2 + 1 + 1 +
1 + 0.2 + 0.7 + 1 +
1 + 1 + 1 + 1
➔
13.3 / 16 = 0.83125
~
0.85



Numerical integration



Numerical integration

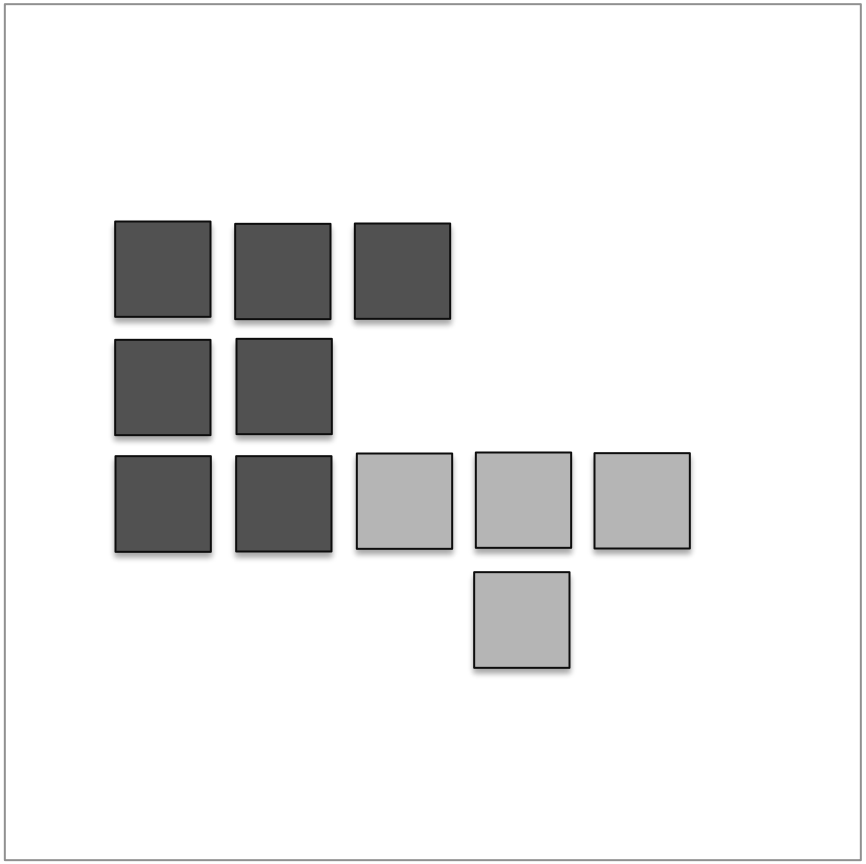
…

➔

54.2 / 64 = 0.846

~

0.85



Cubature: A bit more formal

$$F(\boldsymbol{\omega}) \text{ on } \Omega = \int f(\boldsymbol{\omega}) = \Sigma f(\boldsymbol{\omega}_i) / \mathrm{N}$$

To find the integral F of a function f, decompose Ω it into N as-small-as-possible cubes, evaluate f on each (**sample**) and average.

Is this it?

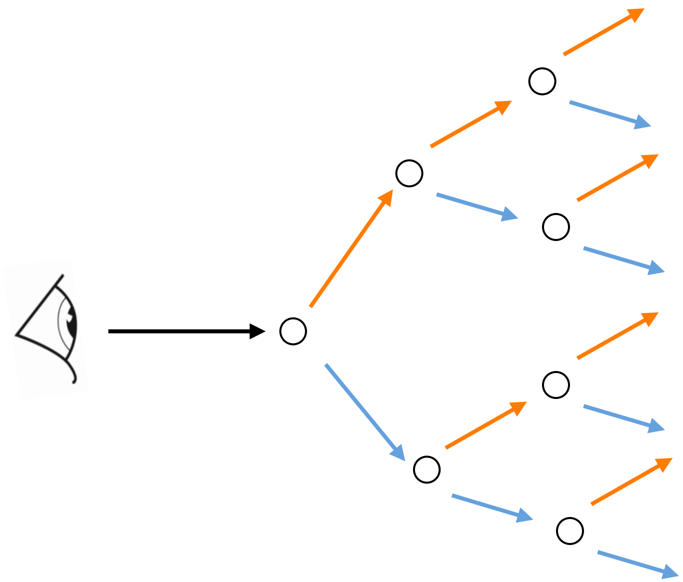
- Very simple method!
  - To get $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$
  - Subdivide hemisphere Ω above $\mathbf{x}$ into N strata $\boldsymbol{\omega}_i$
  - Evaluate integrand, i.e., send a ray
    $f = L(\mathbf{y}, \mathbf{-}\boldsymbol{\omega}_i) f_\mathrm{r}(\mathbf{x}, \boldsymbol{\omega}_{i,}\ \boldsymbol{\omega}_\mathrm{o})\cos(\theta)$
    - Triple product of light, svBRDF and geometric term
  - Average
  - Done!

Problem: Course of dimensionality

- L appears on both sides
  - For $L(\mathbf{y}, -\boldsymbol{\omega}_i)$ need to solve another integral
  - OK, lets go to $\mathbf{y}$ and also compute $L(\mathbf{y}, -\boldsymbol{\omega}_i)$
    - For $L(\mathbf{z}, -\boldsymbol{\omega}_i)$ need to solve another integral
    - OK, lets go to $\mathbf{z}$ and also compute $L(\mathbf{z}, -\boldsymbol{\omega}_i)$
      - For $L(\mathbf{z}_2, -\boldsymbol{\omega}_i)$ need to solve another integral
      - OK, lets go to $\mathbf{z}_2$ and also compute $L(\mathbf{z}_2, -\boldsymbol{\omega}_i)$
        - » For $L(\mathbf{z}_3, -\boldsymbol{\omega}_i)$ need to solve another integral
        - » OK, lets go to $\mathbf{z}_3$ and also compute $L(\mathbf{z}_3, -\boldsymbol{\omega}_i)$
          - For $L(\mathbf{z}_2, -\boldsymbol{\omega}_i)$ need to solve another integral
          - OK, lets go to $\mathbf{z}_2$ and also compute $L(\mathbf{z}_2, -\boldsymbol{\omega}_i)$

Recursion

- Recursion in CS is not evil
- Worked well in ray-tracing:



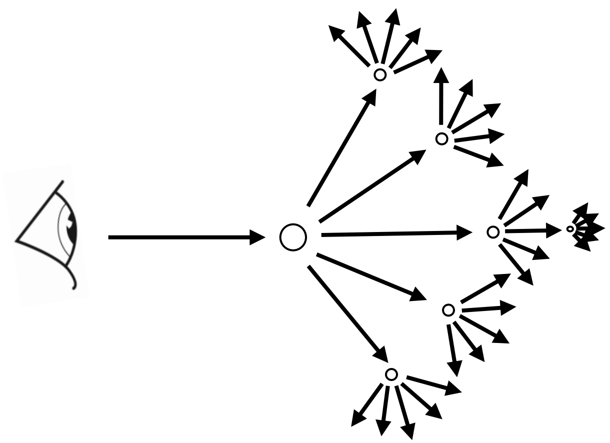
For depth d and reflection/refraction the number of rays is

$$2^d$$

So three-bounce is
$2^3 = 8$ rays / pixel

Recursion

- Recursion in CS is not evil
- Impractical for the real RE
- Typical N is maybe 100 to 1000

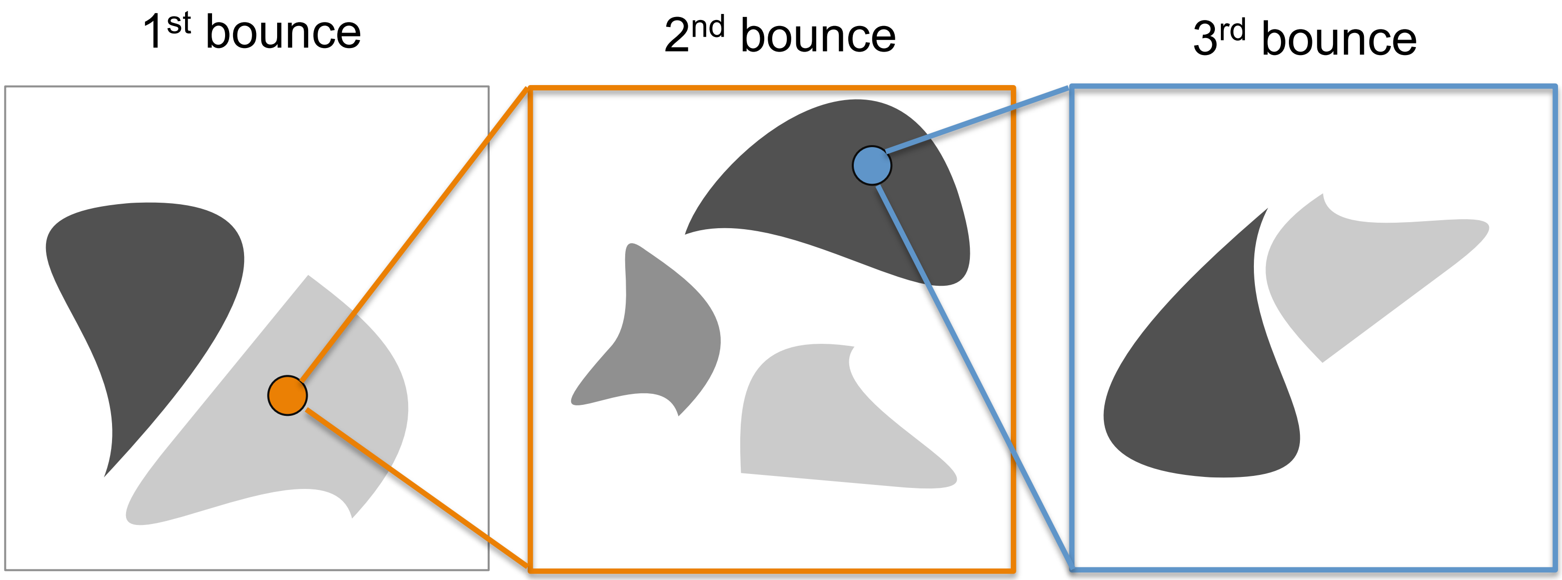


For depth d and N strata the number of rays is

$$N^d$$

So three-bounce is
1000^3 = 1B rays / pixel

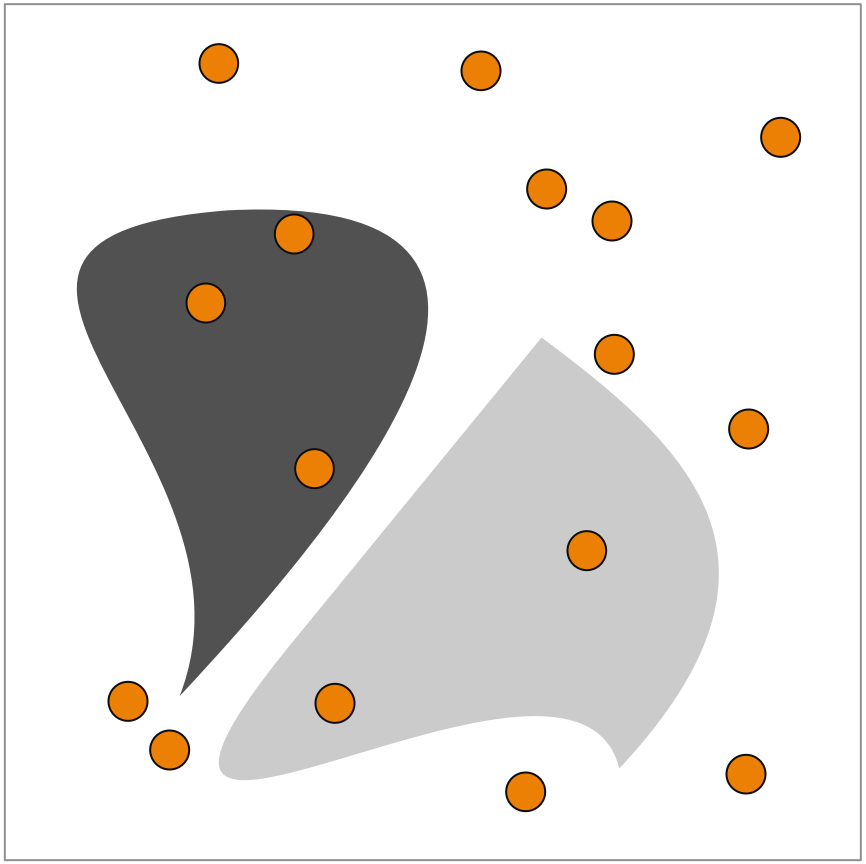
Another way to imagine it



Every time we want any accurate value (ornge dot) at any bounce we need to resolve exponentially many others below it to proceed

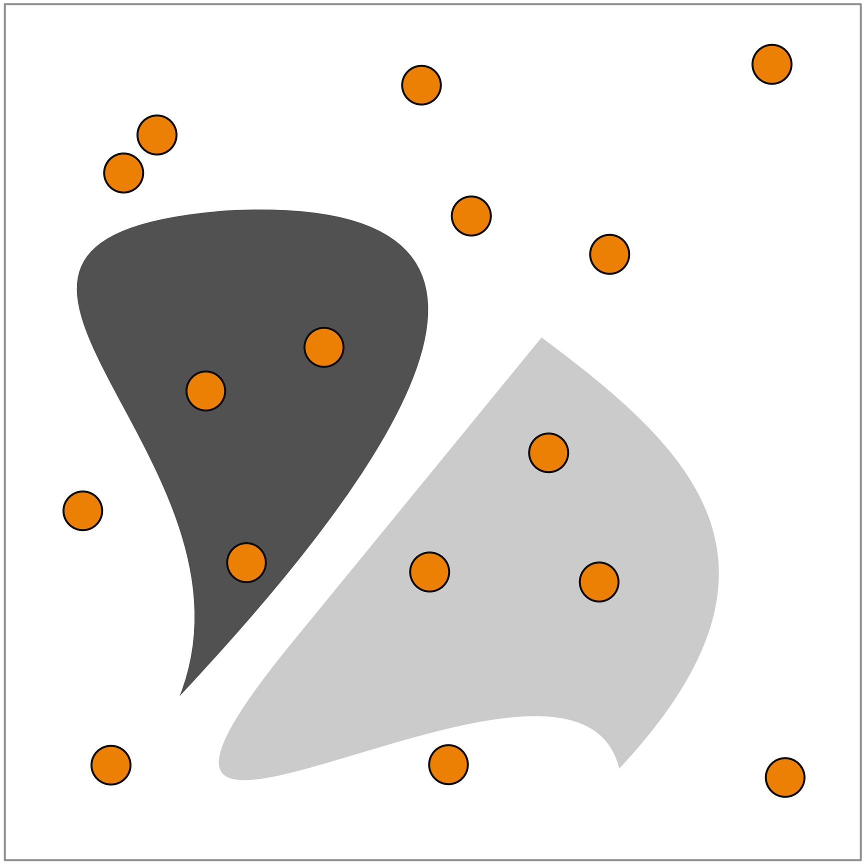
Alternative: Random samples

11 * 1.0 +
3 * 0.2 +
2 * 0.7
➔
13.0 / 16 = 0.8125
~
0.85

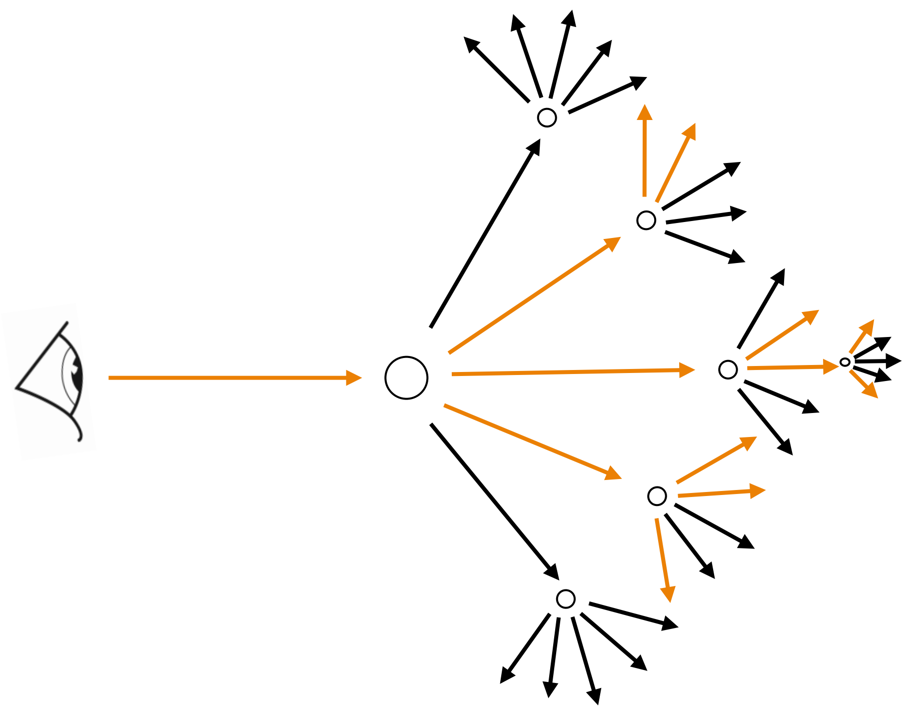


Alternative: Random samples

11 * 1.0 +
3 * 0.2 +
3 * 0.7
➔
13.7 / 16 = 0.856
~
0.85



What do we get from random?

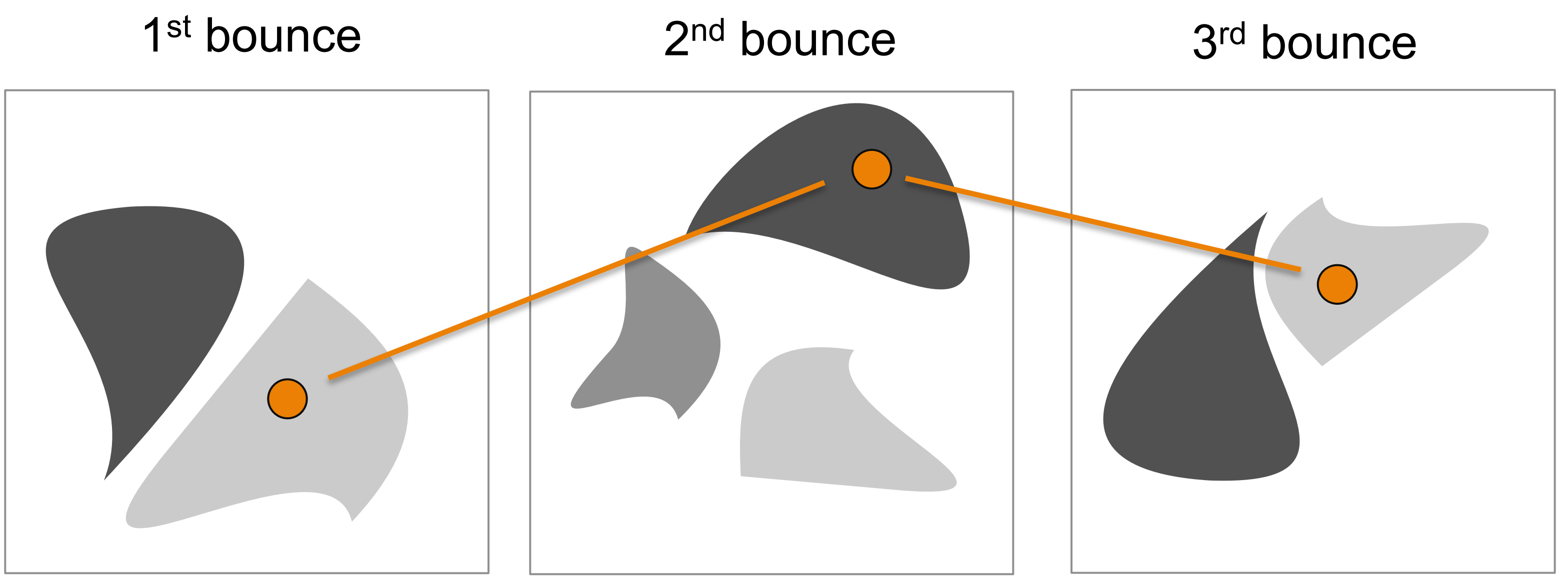


For depth d and N samples the number of rays is

$$N \times d$$

So three-bounce is
3×1000 = 3K rays / pixel

Another way to imagine it



As we just need an approximate value, we can proceed with any point without looking at all ohers

Monte Carlo: A bit more formal

$$F(\boldsymbol{\omega}) \text{ on } \Omega = \int f(\boldsymbol{\omega}) = \Sigma f(\boldsymbol{\omega}_i) / N$$

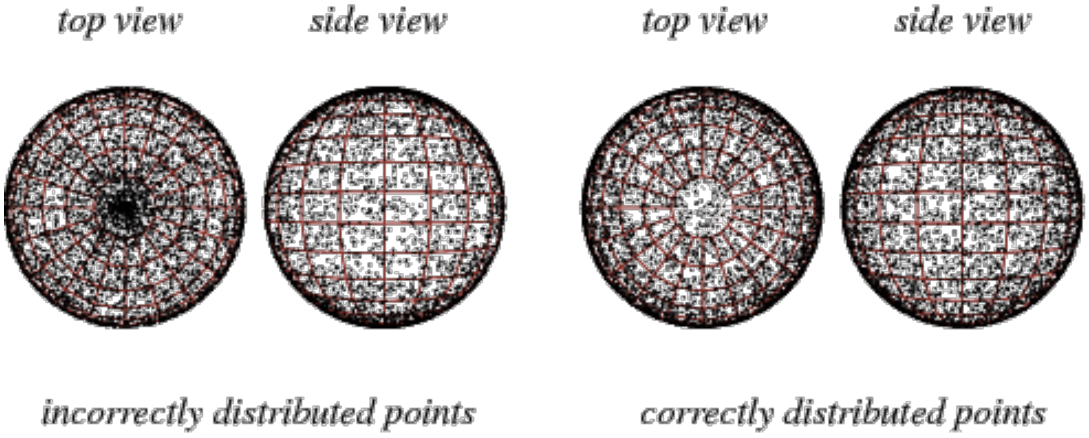
To find the integral F of a function f, place as many samples N onto Ω, evaluate f on each and average.

This is it

- Still very simple method!
  - To get $L(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{o}})$
  - N times
    - random directions $\boldsymbol{\omega}_{i,0}$ above $\mathbf{x},$ $\boldsymbol{\omega}_{i,1}$ above $\mathbf{y},$ etc.
    - Evaluate integrand, i.e. send a ray
      $f = L(\mathbf{y}, \mathbf{-}\boldsymbol{\omega}_i) f_{\mathrm{r}}(\mathbf{x}, \boldsymbol{\omega}_{i,}\ \boldsymbol{\omega}_{\mathrm{o}})\cos(\theta)$
      - Triple product of light, svBRDF and geometric term
  - Done!

How to pick random directions?

- How to pick a random ray in 3D?
- `normalize(vec3(frand(), frand(), frand()))`?
- Clumps on axis and diagonals
- Bias in result!



When to stop?

- Multiple options
- Popular:
  - After a fixed depth
  - When contribution falls below a threshold

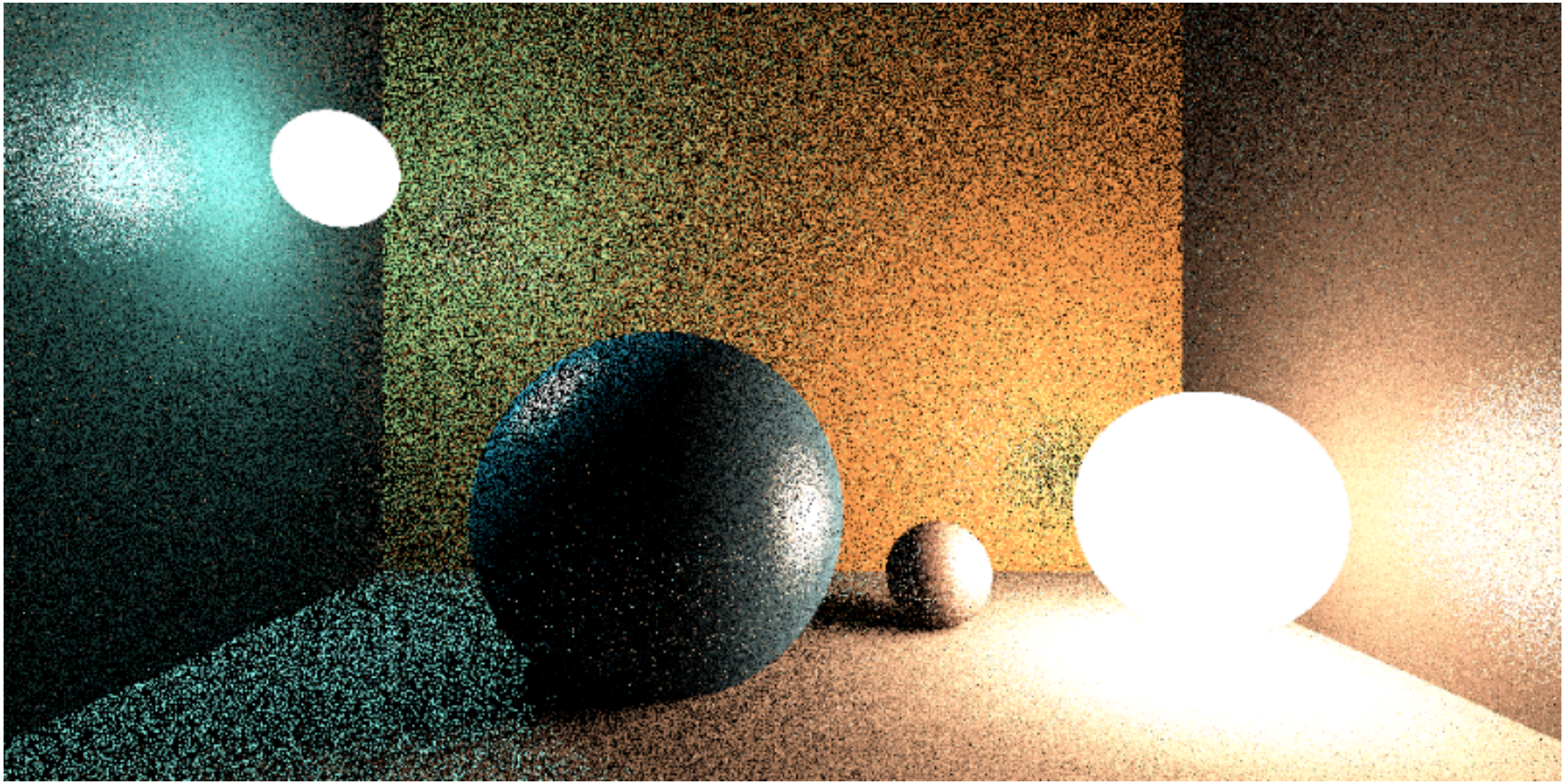
Progressiveness

1 Sample

Progressiveness

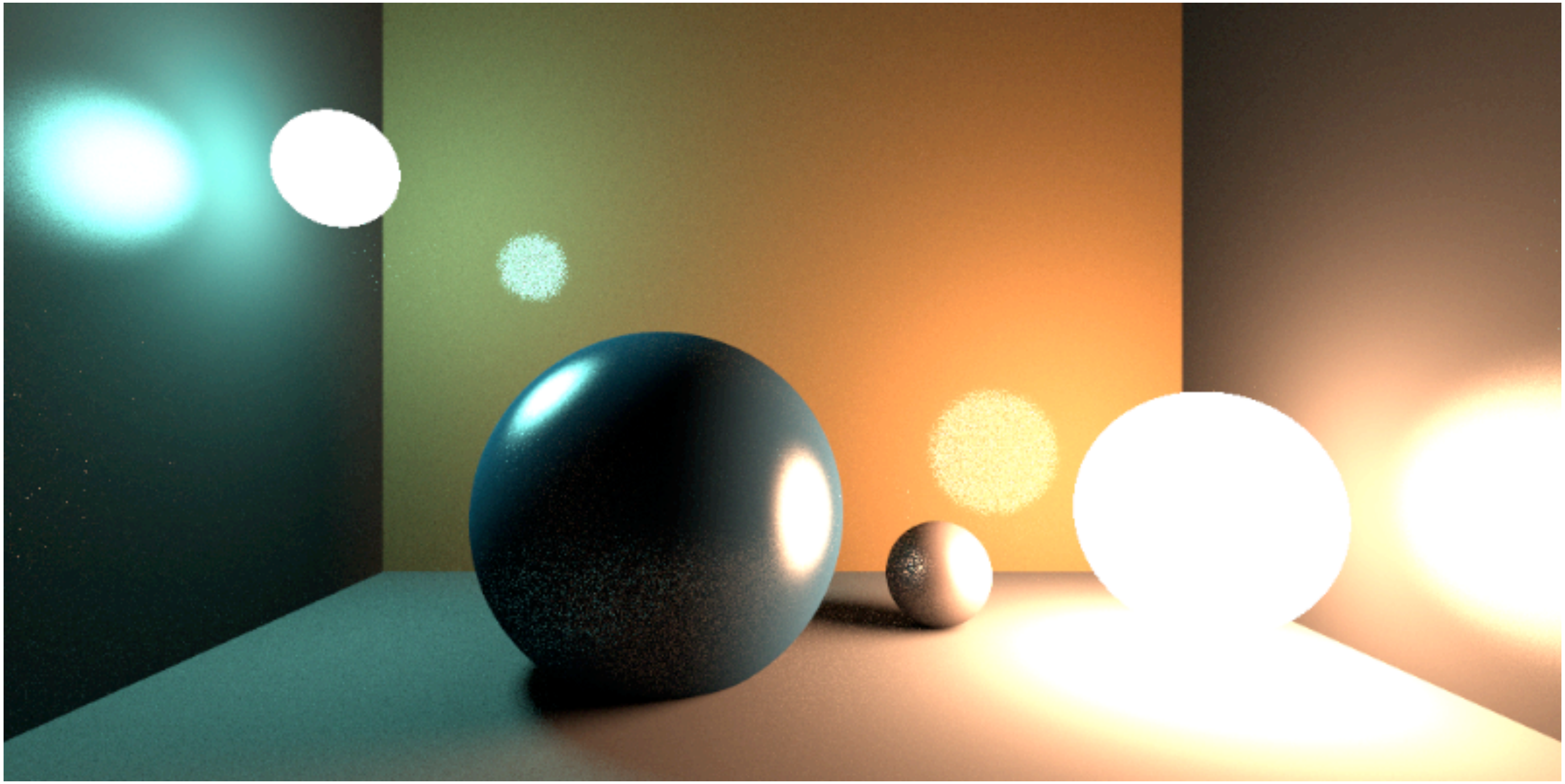
2 Samples

Progressiveness



10 Samples

Progressiveness



1000 Samples

Progressivenes

- After N samples we get an image
- After $2N$ samples, we get an even better one
- This is called **progressive**
- Very useful for previews

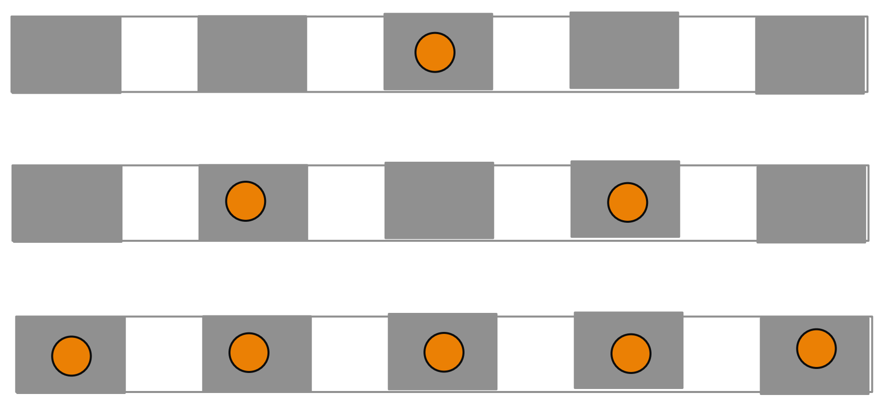
Random has another reason

- **Aliasing** is the second reason for random
- Consider this integrand, not even recursive:

Cubature with 1 sample = 0.0

Cubature with 2 samples = 0.0

Cubature with 5 samples = 0.0



Ground truth = 0.5

Random has another reasons

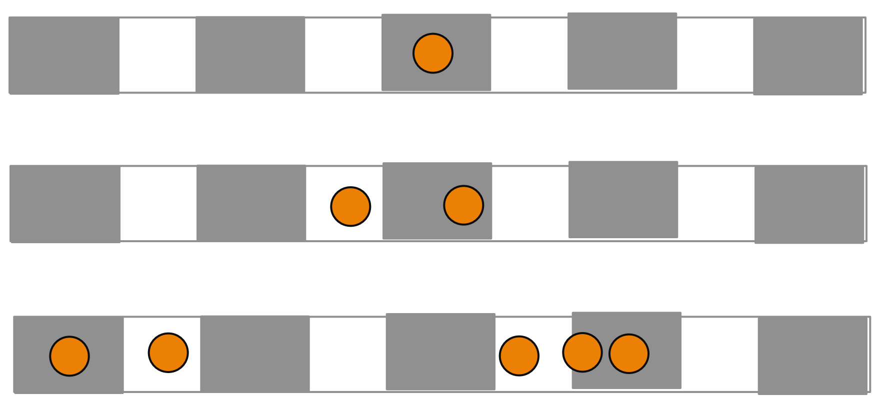
- **Aliasing** is the second reason for random
- Consider this integrand, not even recursive:

Monte Carlo with 1 sample = 0.0

Monte Carlo with 2 samples = 0.5

Monte Carlo with 5 sampls = 0.4

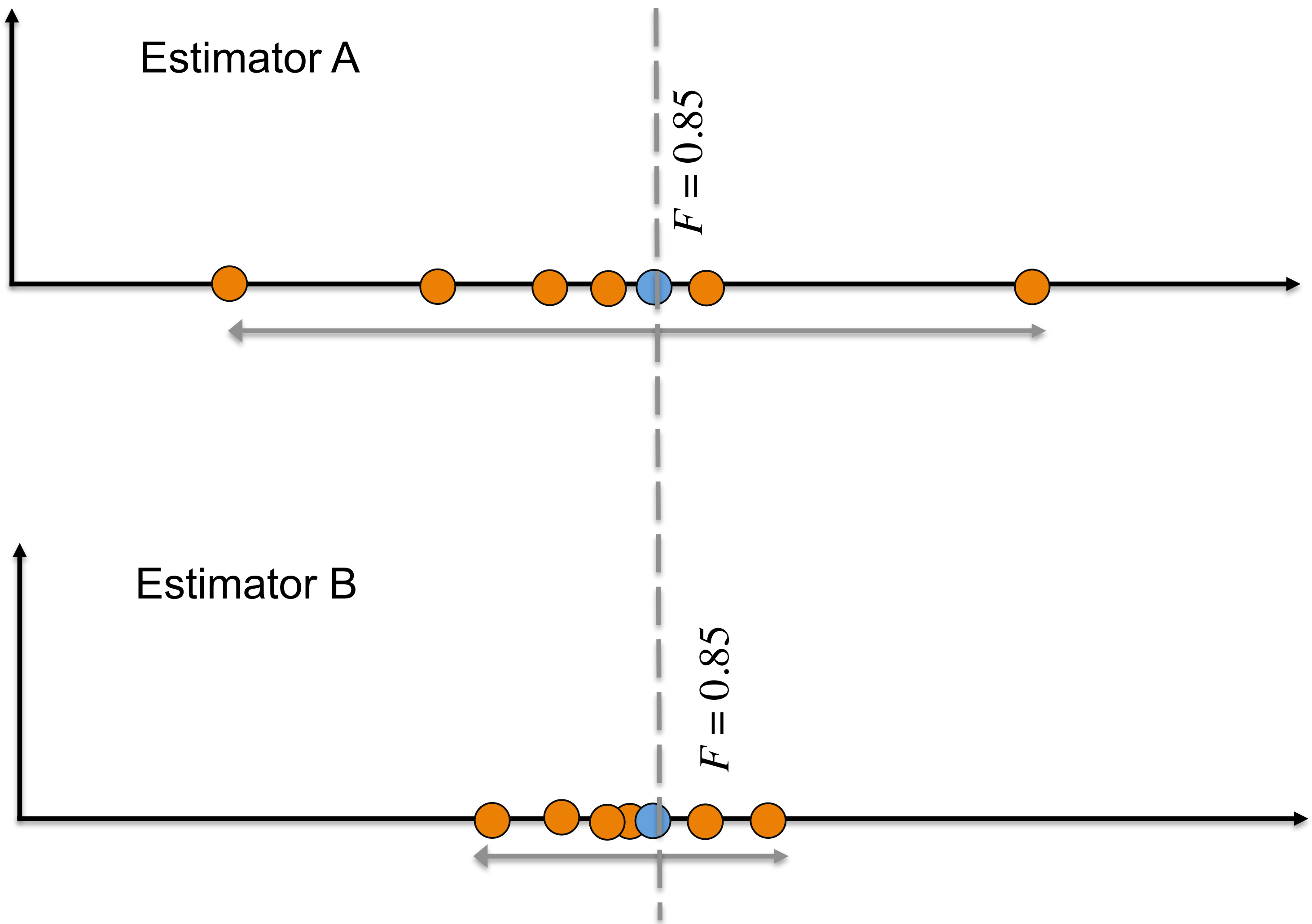
Ground truth = 0.5



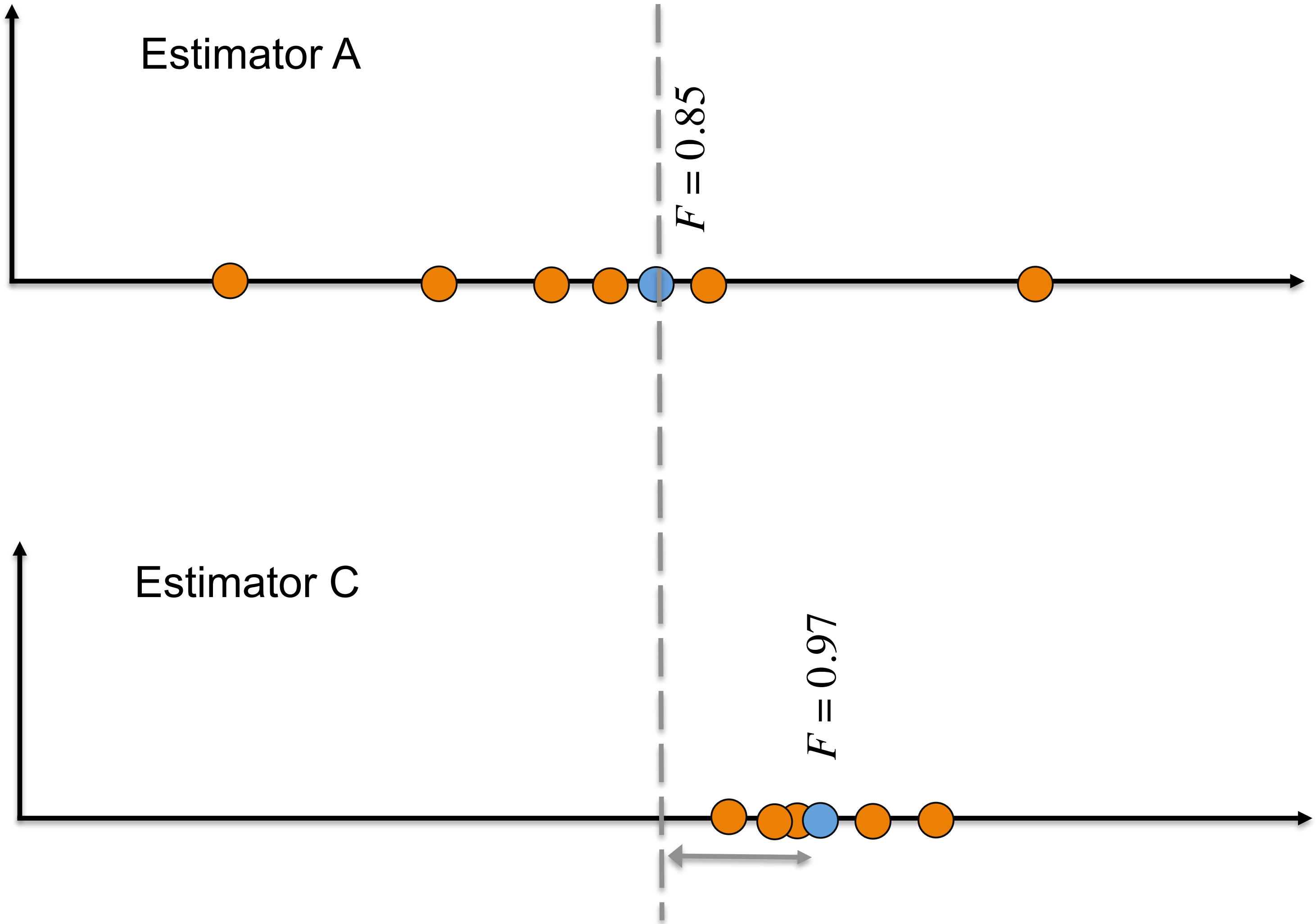
Estimator/Variance/Bias

- We get a new value for every random seed
- We map these into an **estimate**
- This is the value of the integral
- If there is a deviation, we call it **bias**
- Around this exists a distribution of values
- This distribution has a **variance**

Variance of an estimator



Bias of an estimator



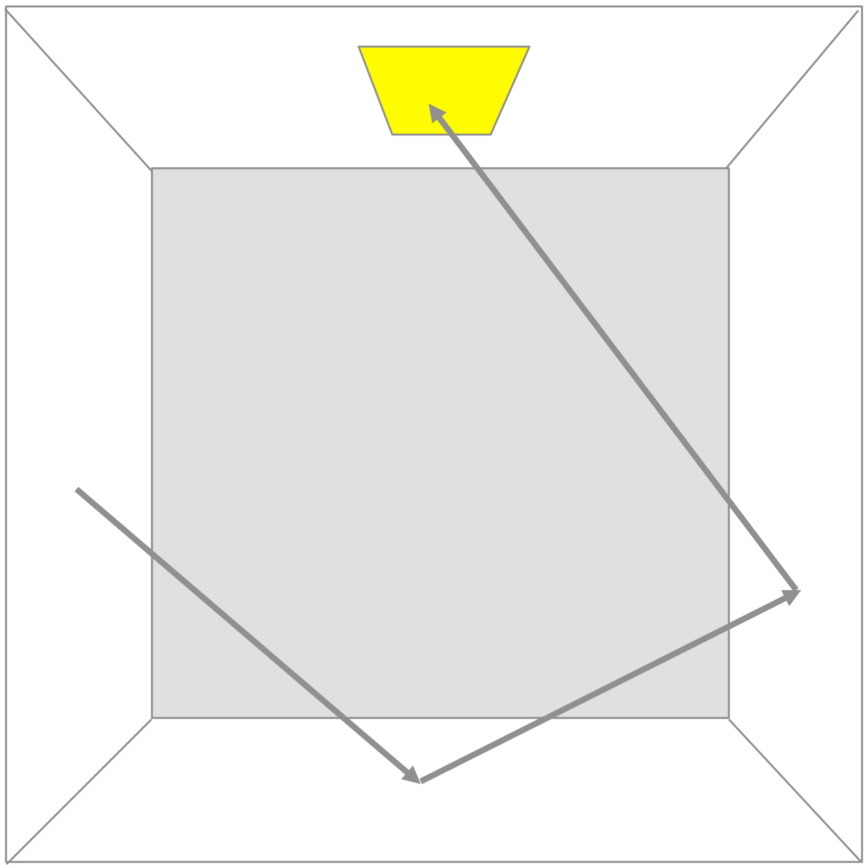
Desiderata variance

- Also it has no bias, i.e., the expected value is really the solution of the integrand
- OpenGL and CW1 ray-tracing: All biased
- A good estimator has a low **variance**
- Whenever we render, we will get a value close to the true value
- To this end, we do **variance reduction**

Variance reduction

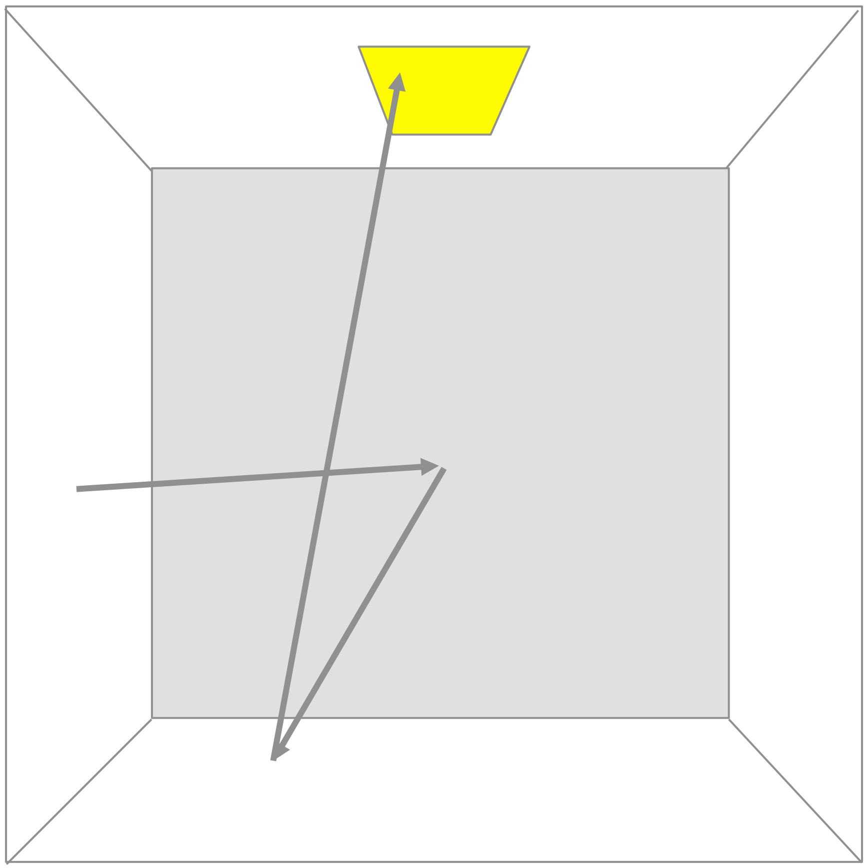
- Next-event estimation
- Good sample patterns
  - Jittered
  - Quasi-Monte Carlo
  - Blue noise
- Importance Sampling

Path tracing - High hopes



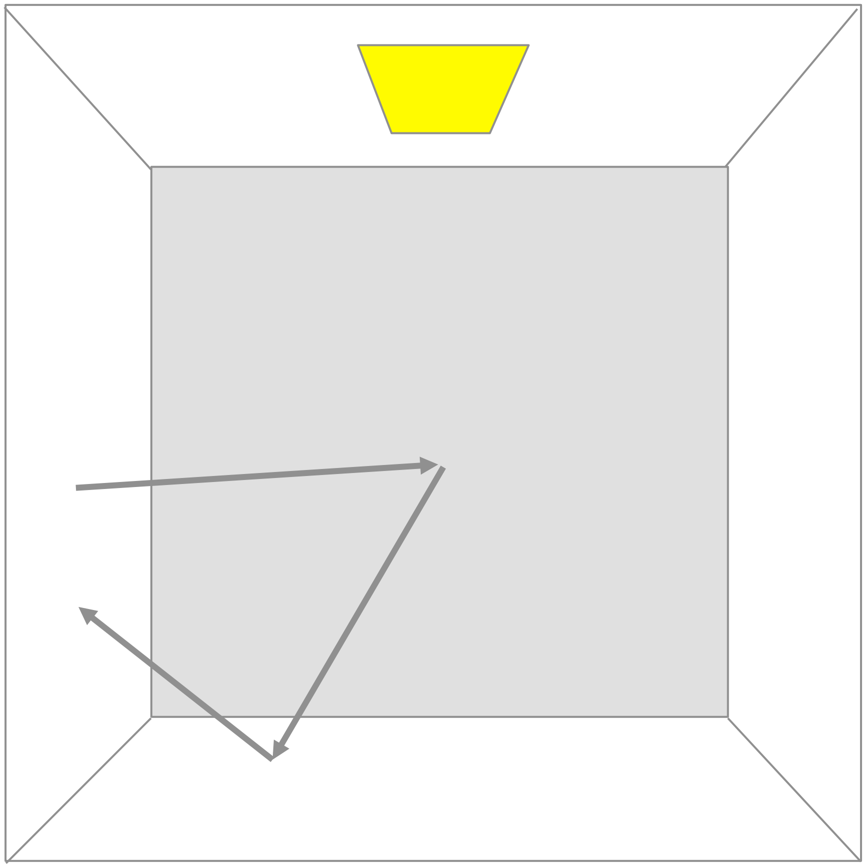
We hope for this …

Path tracing – High hopes



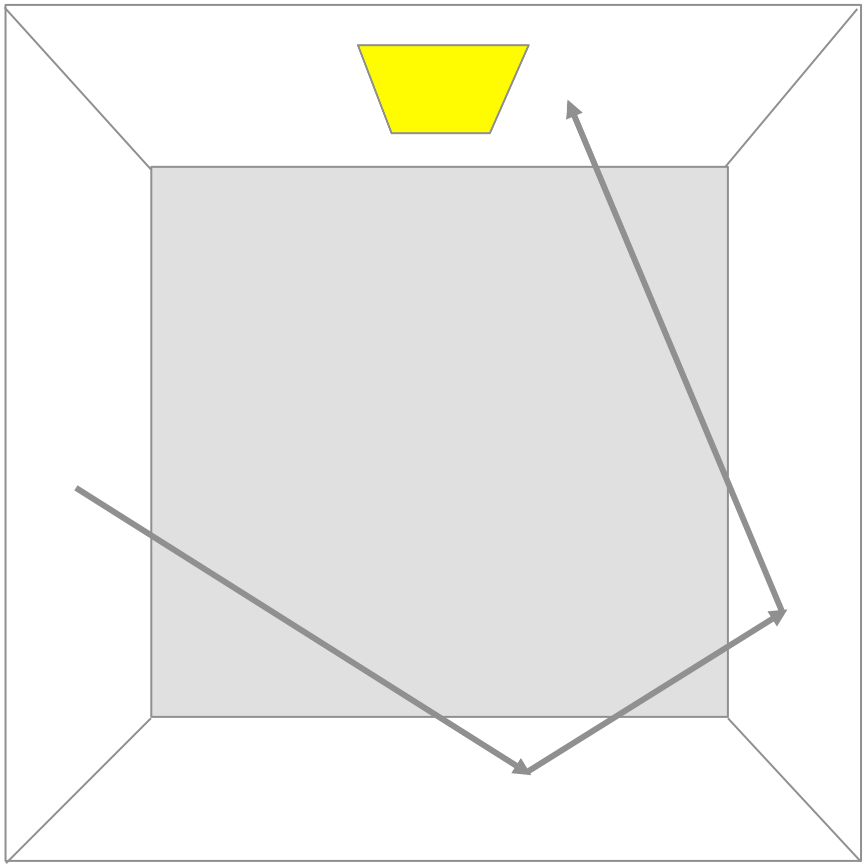
We hope for this …

Path tracing - Reality



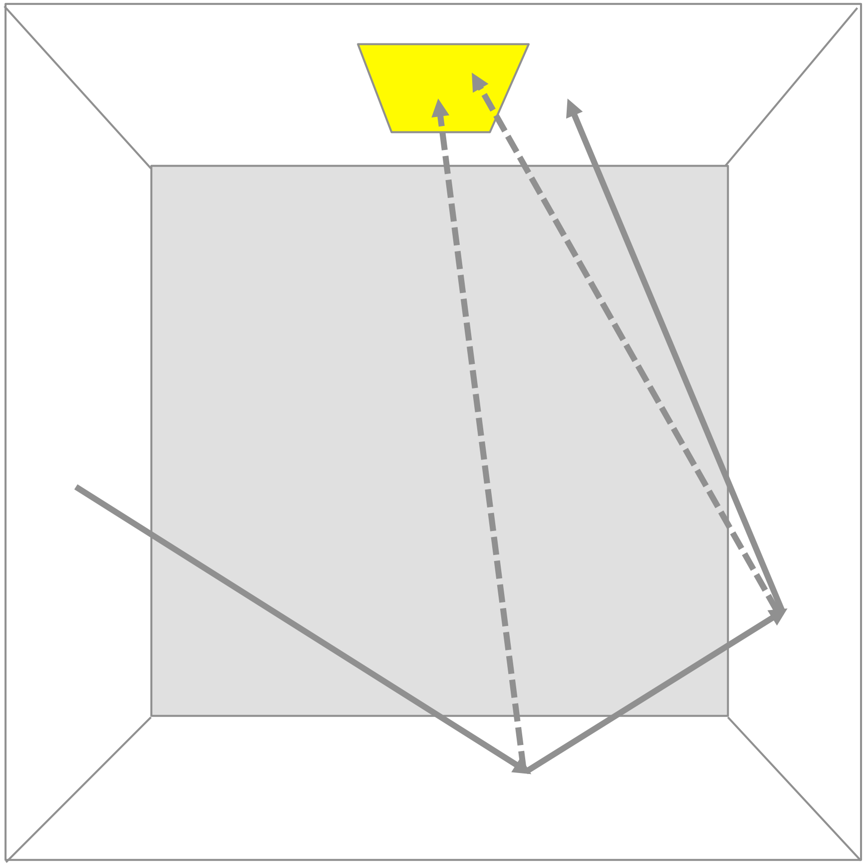
But what we get is

Path Tracing - reality



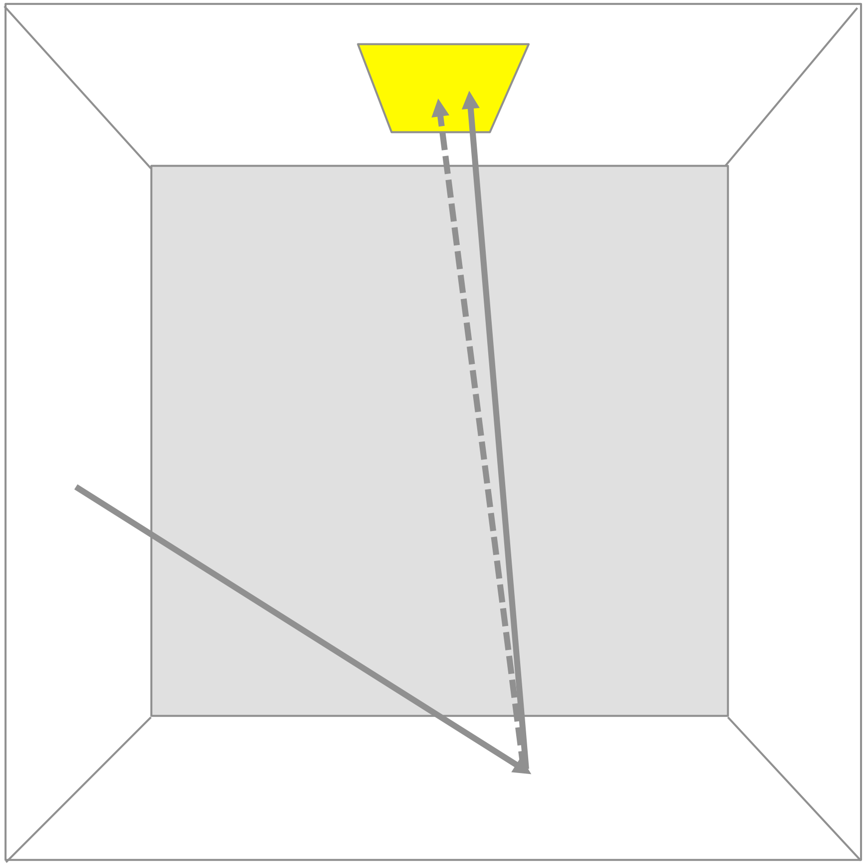
But what we get is

Next-event estimation



But what we get is

Problem: Double-accounting



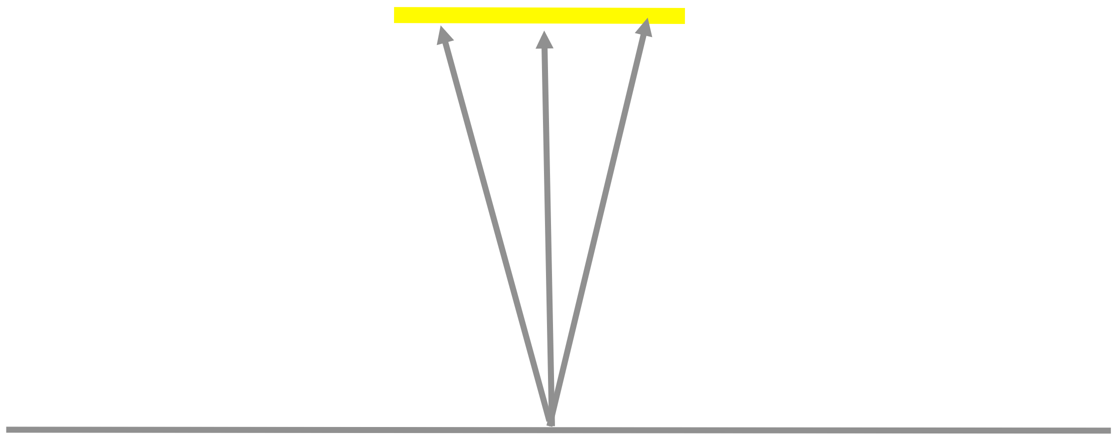
There are now **two ways** to hit the light.
Simple solution: Simply only take emission from NEE.

Next event estimation

- Two simple changes
  - Add a random ray in the direction to the light
  - Remove adding in emission L_{e} on all other paths
- Best for **small** light sources
- Result:
  - Will never miss direct light at any point
  - Still have all benefits of MC
- Glorified Whitted-style CW 1 ray-tracing

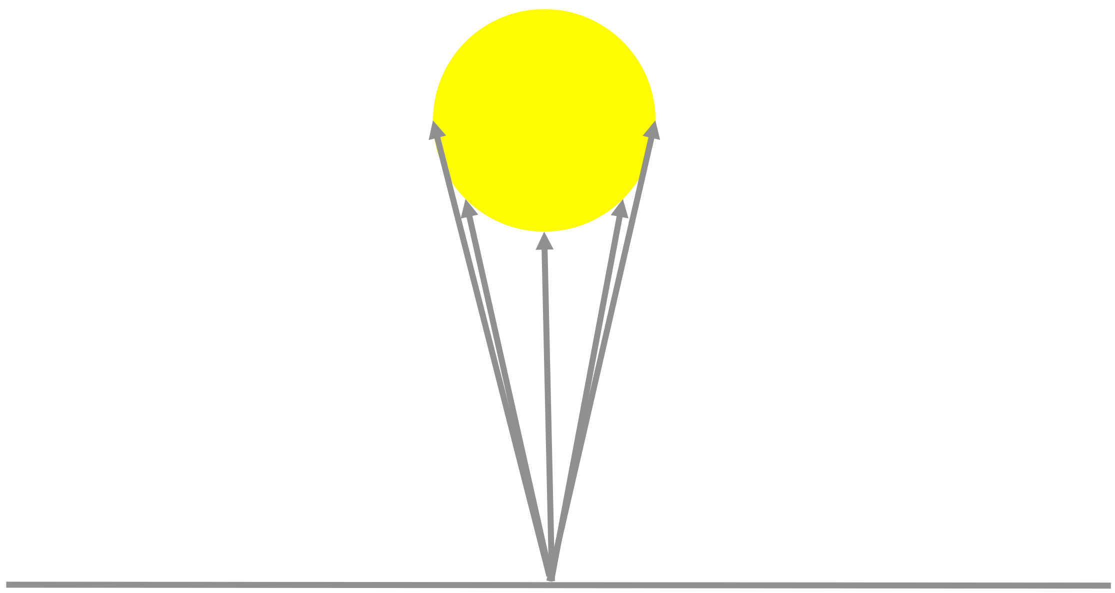
Endpoint choice

- Simple for flat area lights



Endpoint choice

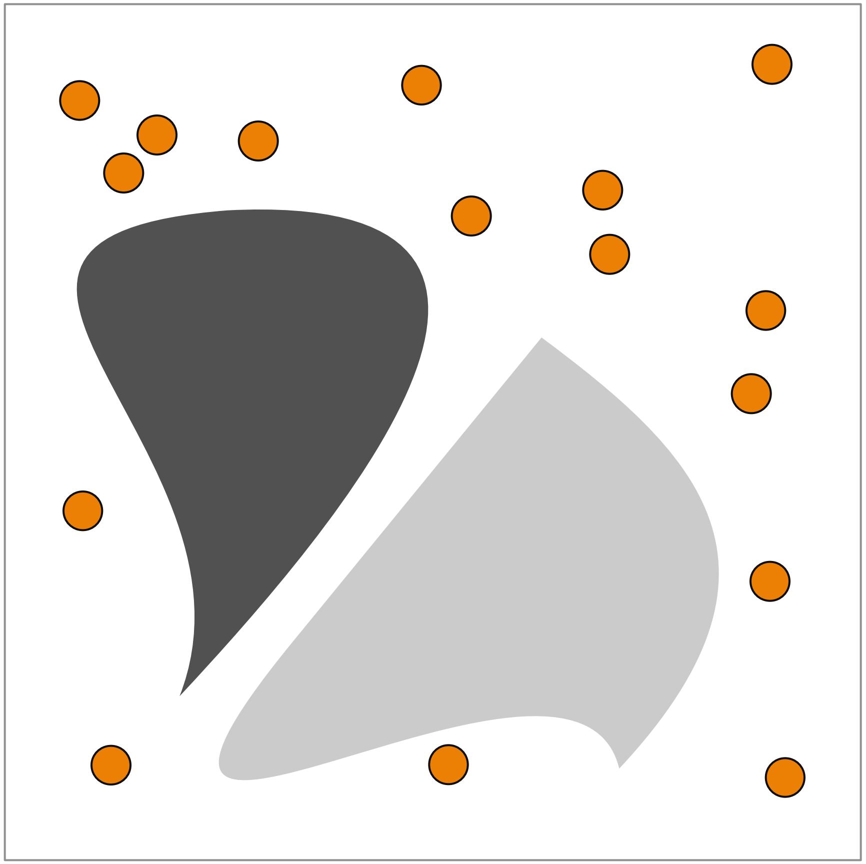
- Hard for e.g., spheres



Uniform random can go bad

16
➔
16 / 16 = 1
~
0.85

(not so cool)



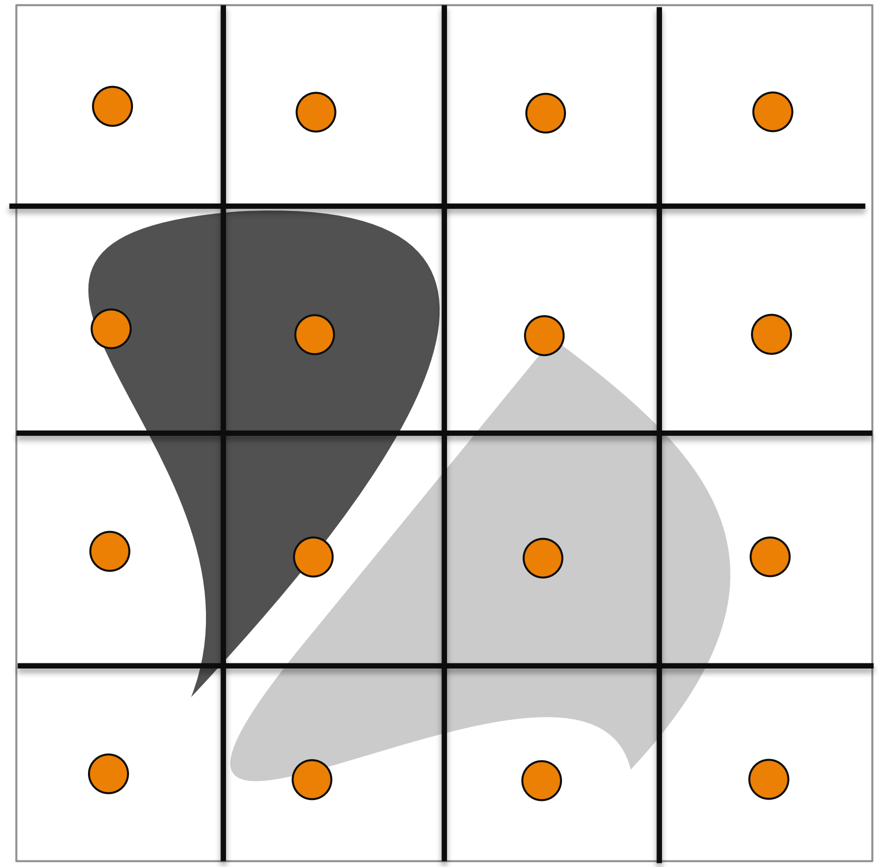
The ideal sample pattern

- Two contradicting goals:
  1. Maybe not be regular
  2. But always cover the domain uniformly, not only in the limit

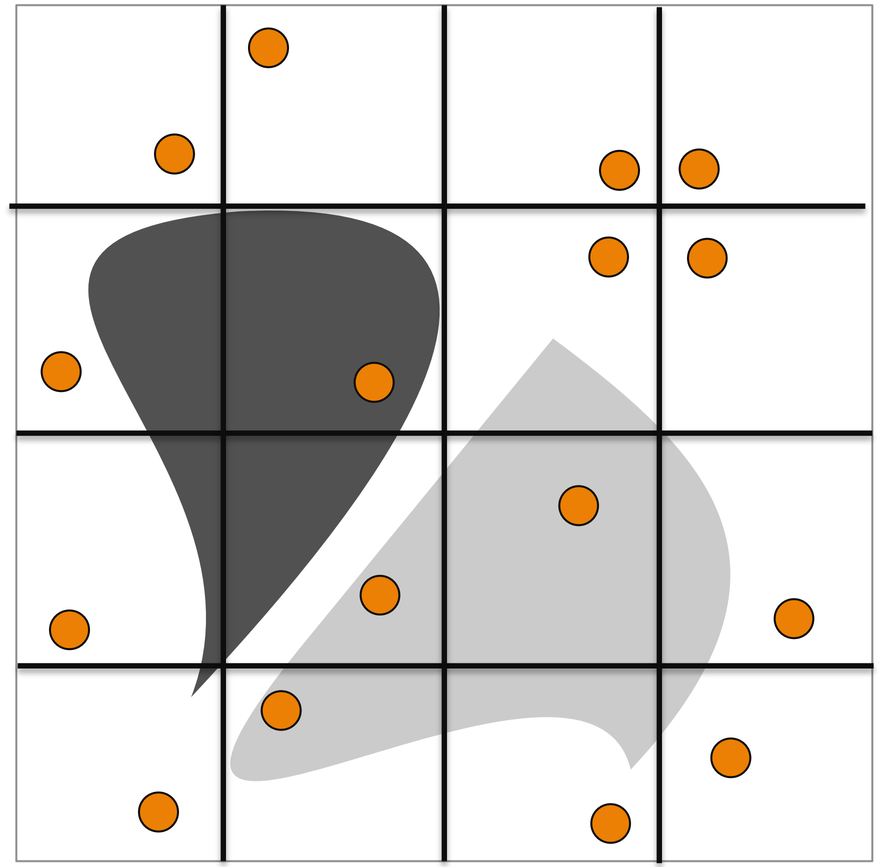
Jittering

- Back to the future:
  - Do cubature first
  - Then jitter every sample inside its cell
- Suffers from the curse of dimensionality
- Prevents aliasing
- Applicable if dimensionality is low
  - Example: Area light sampling

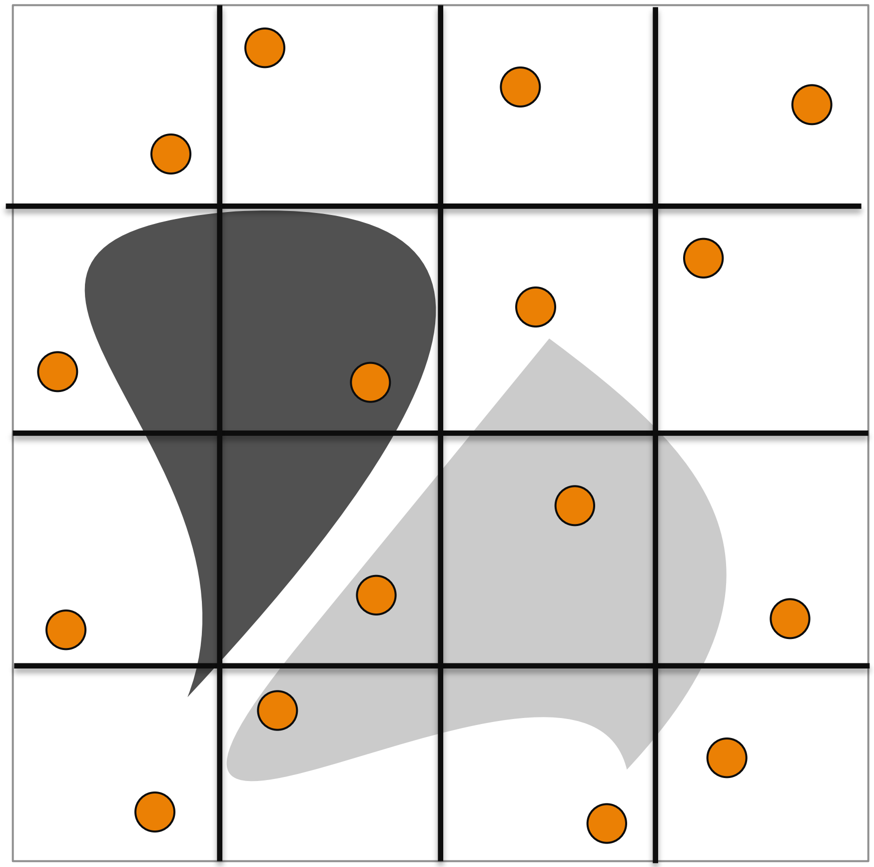
Regular (recap)



Jittered



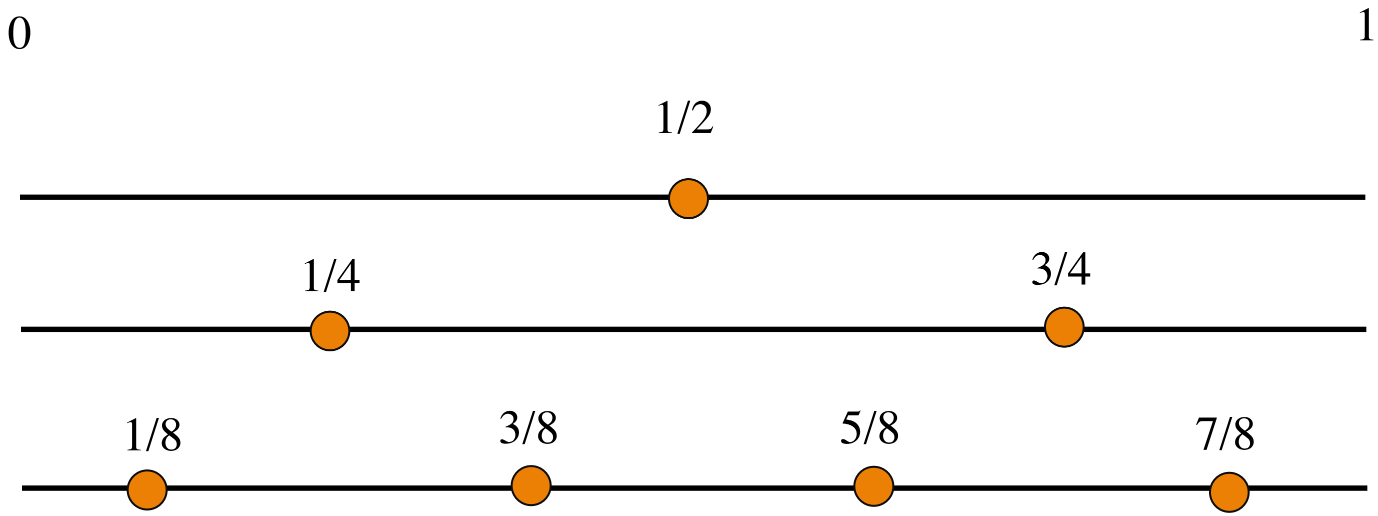
Multi-Jittered



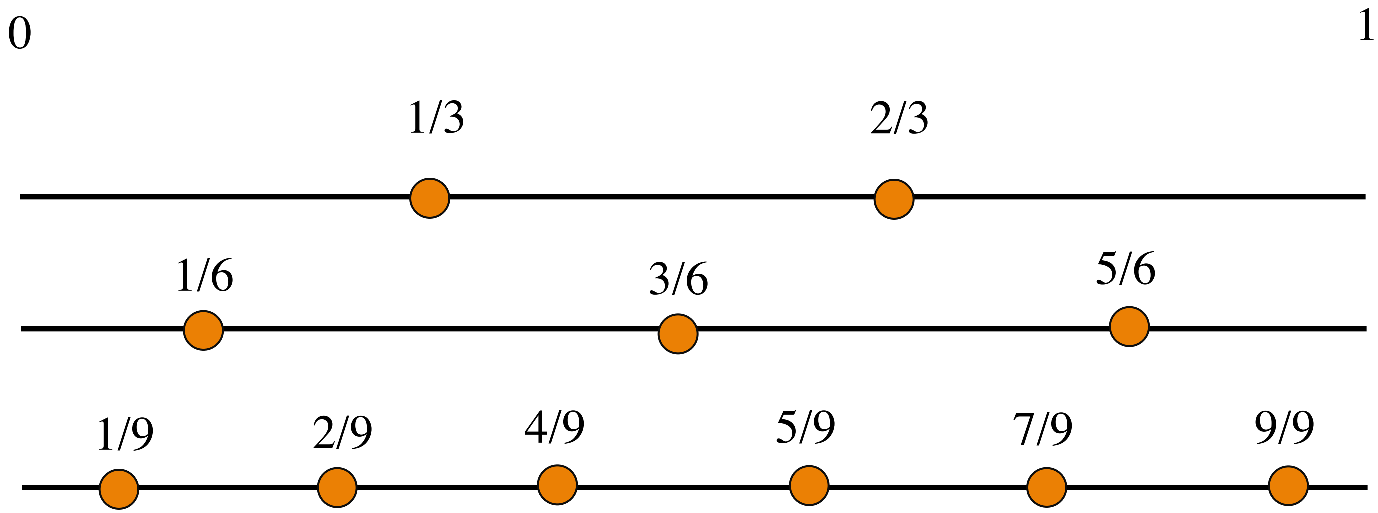
Halton

- A way to place samples
  - Somewhat uniform
  - Without structure
  - In high dimensions
- A typical work-horse solution for rendering
- Defined on the unit hypercube

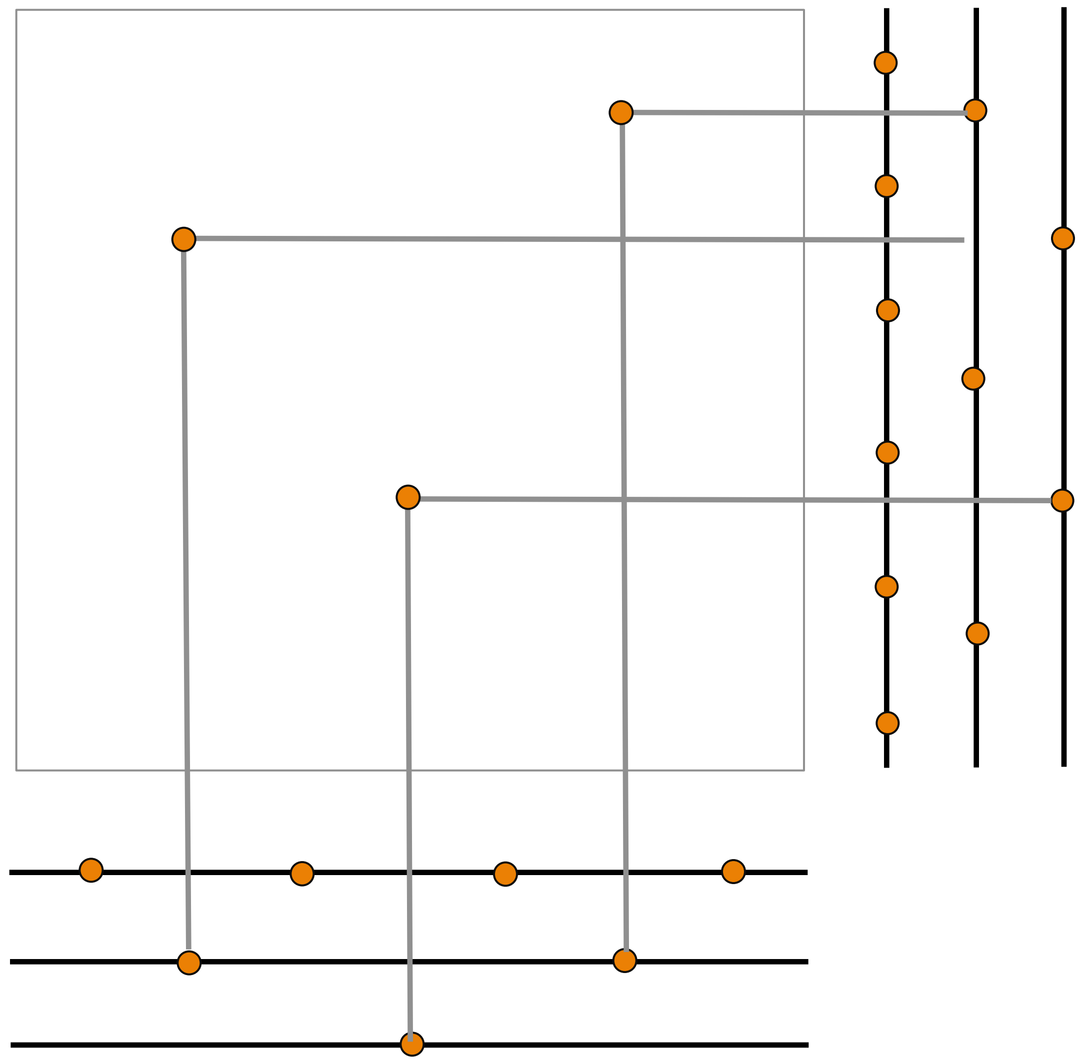
Radical inverse base $n = 2$



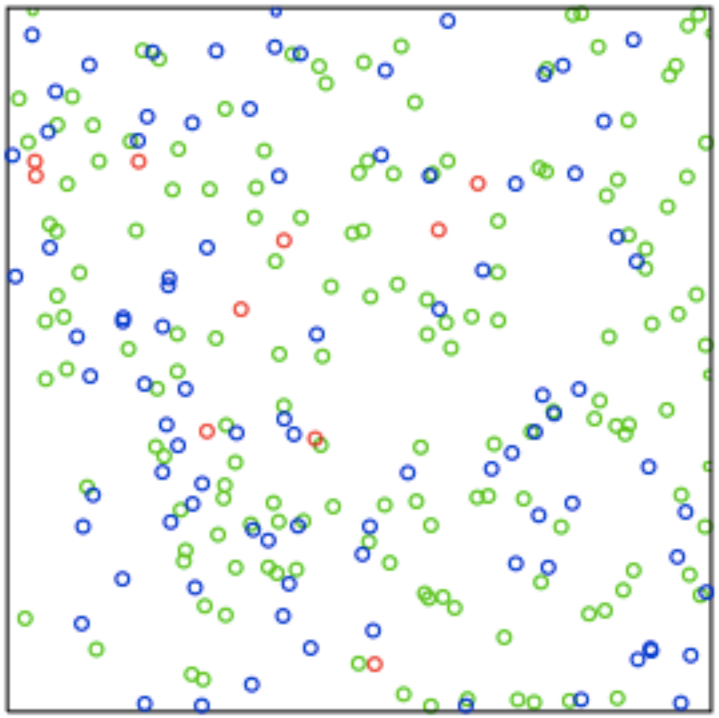
Radical inverse base $n = 3$

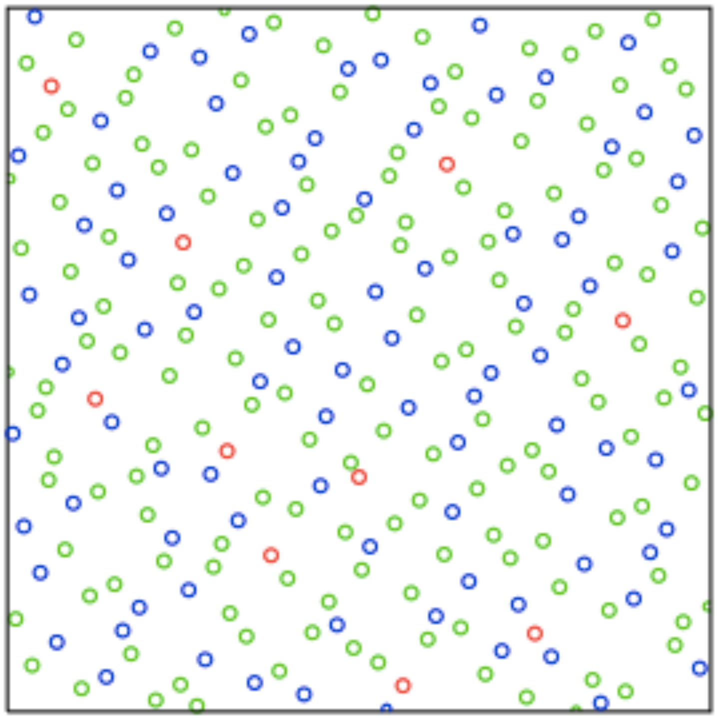


Halton in $2, 3$ i.e. $\pi(0), \pi(1)$



Random vs. Halton





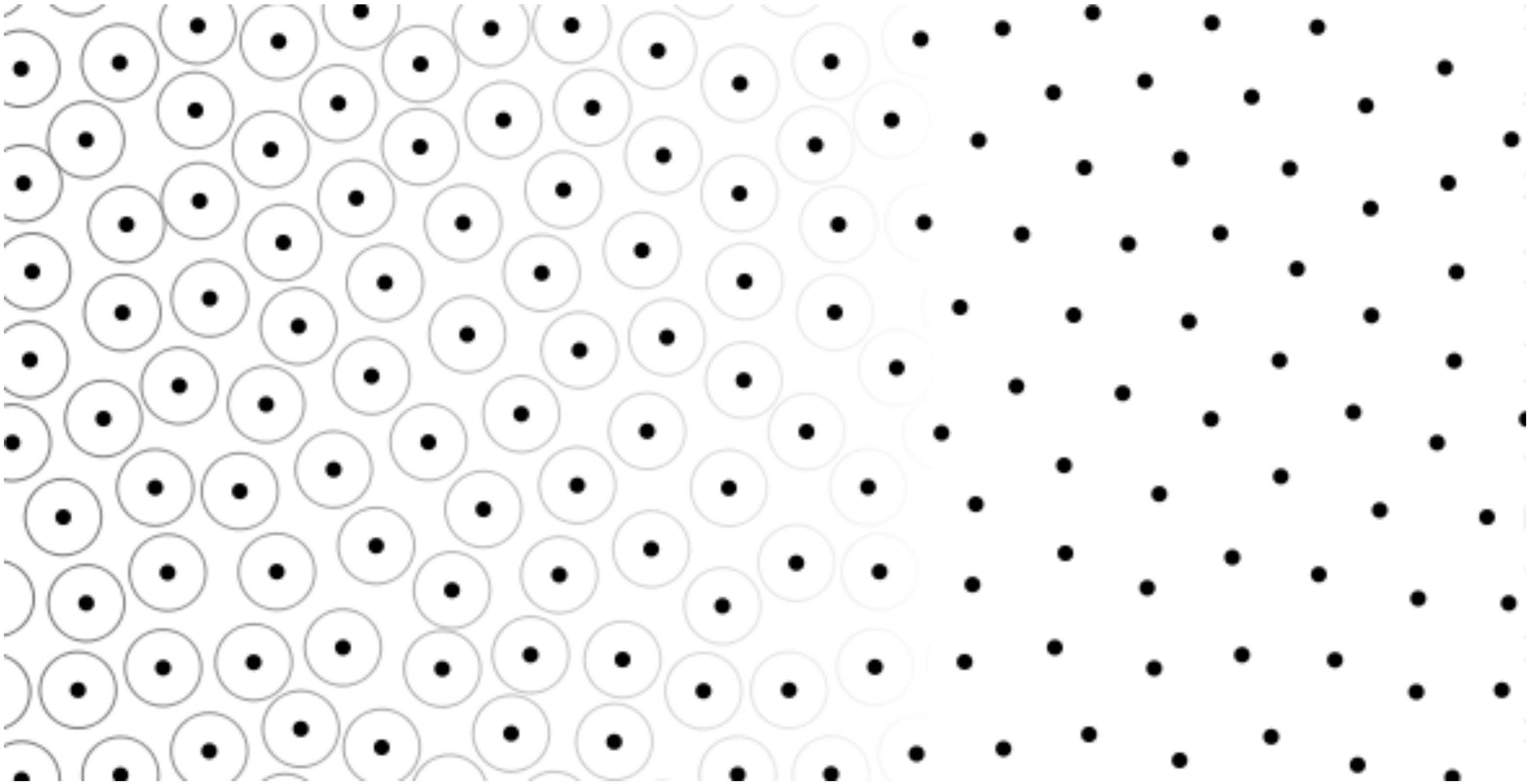
Quasi-Monte Carlo sampling

- To get an n-dimensional Halton pattern
- Build the radical inverse in the co-prime basis $\pi(0, .., n\text{-}1)$
- Build tuples in the order they occur in each sequence
- Improvement: **Hammersley** (reguar 1st dim.)
- De-correlation: **Cranely patterson** rotation

Poisson disk / Blue noise

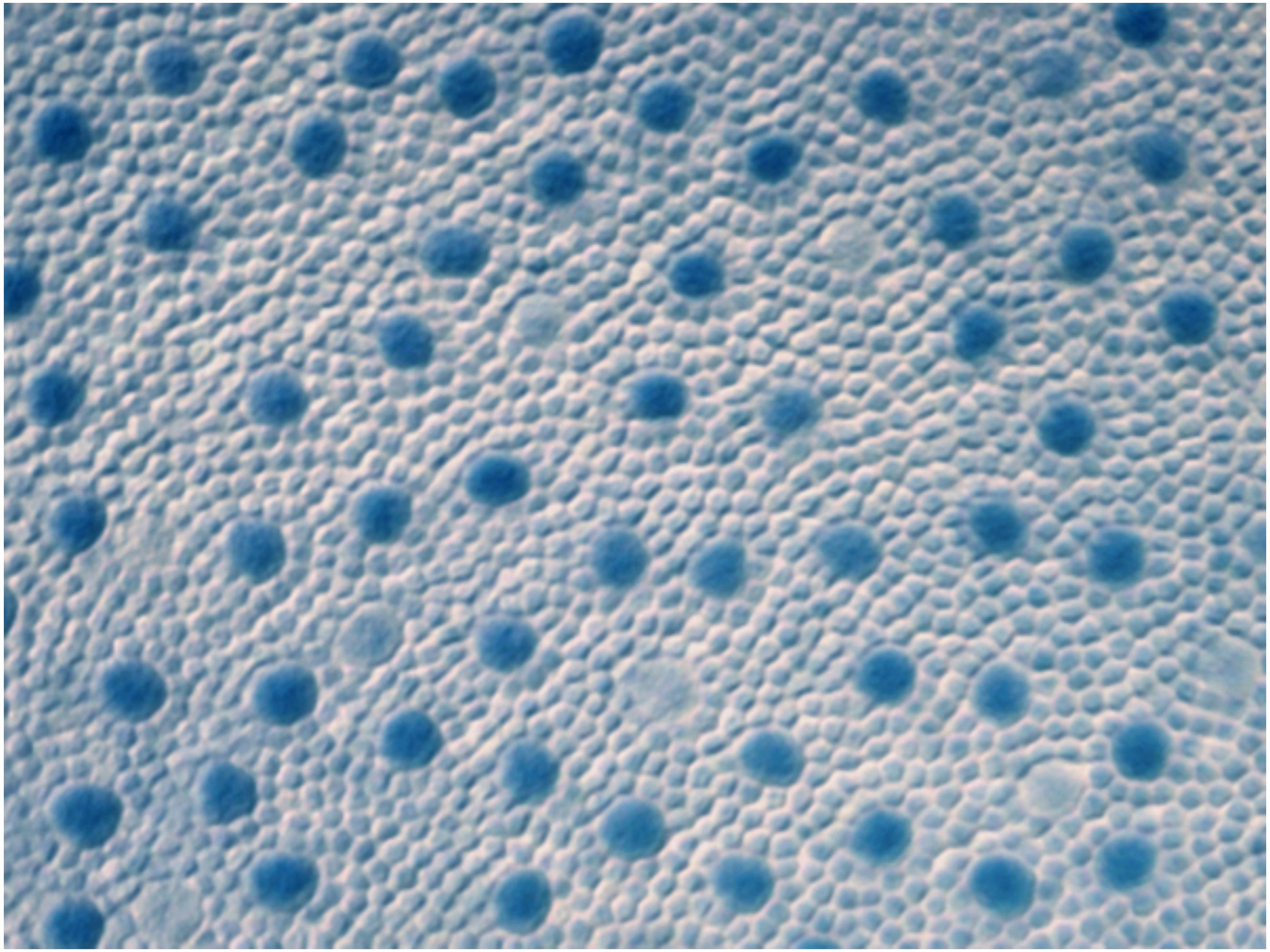
- Patterns with a maximal minimal distance between all points is called **Poisson disk**
- Another way is to see the spectrum of the distribution of distances: It is blue, i.e. no small minimal distances

Poisson disk



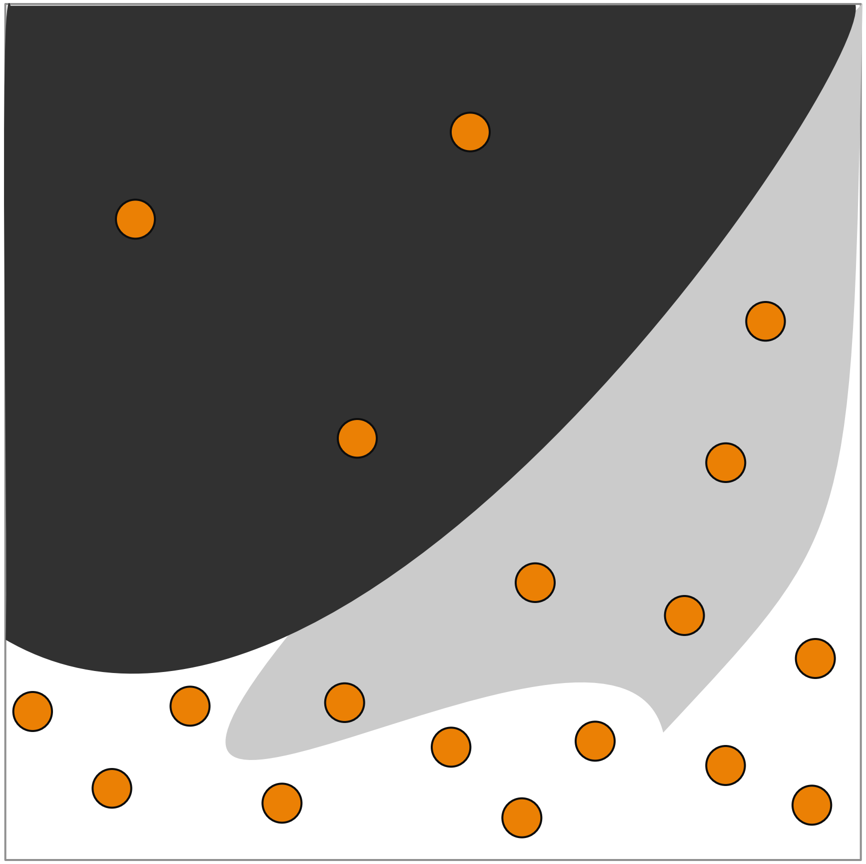
Note: The smallest circle of all circles around each point is quite large

Blue noise



Receptor distribution on the macaque retina prevents aliasing

Importance sampling



Put more samples where the integrand is high, as here the errors have the largest effect.

Importance sampling

- Placing the samples non-uniformly will introduce bias
- Fortunately, random uniform is just a special case of a more general estimator formulation we will see next
  - Before we took $\boldsymbol{\omega}_i$ uniform, so $p(\boldsymbol{\omega}_i) = 1 / |\Omega|$
  - Any other p will work as well
  - Ideall $p \sim f$

MC with importance sampling

$$F(\boldsymbol{\omega}) \text{ on } \Omega = \int f(\boldsymbol{\omega}) = 1/N \sum f(\boldsymbol{\omega}_i) / p(\boldsymbol{\omega}_i)$$

To find the integral F of a function f, place as many samples N onto Ω according to a distribution p, evaluate f and p on each and divide.

What can be p?

- Recall the integrand is
  $f = L(\mathbf{y}, -\boldsymbol{\omega}_i) f_{\mathrm{r}}(\mathbf{x}, \boldsymbol{\omega}_{i,} \ \boldsymbol{\omega}_{\mathrm{o}}) \cos(\theta)$
- We could sample for
  - Light: Hard, integral equation itself.
  - BRDF: Not so hard, done analytically
  - Geometric term: Even easier analytically
  - Products of all of the above: Even harder then any alone, but doable

Exampe: IS for direct light

Same amount of rays ©U Virginia

Recap

- Rendering is solving an integral equation
- Analytic and some numeric methods no-go
- Monte Carlo is the method of choice
- Suffers from noise (variance)
- Need to use variance reduction methods
  - Sample patterns
  - Importance sampling