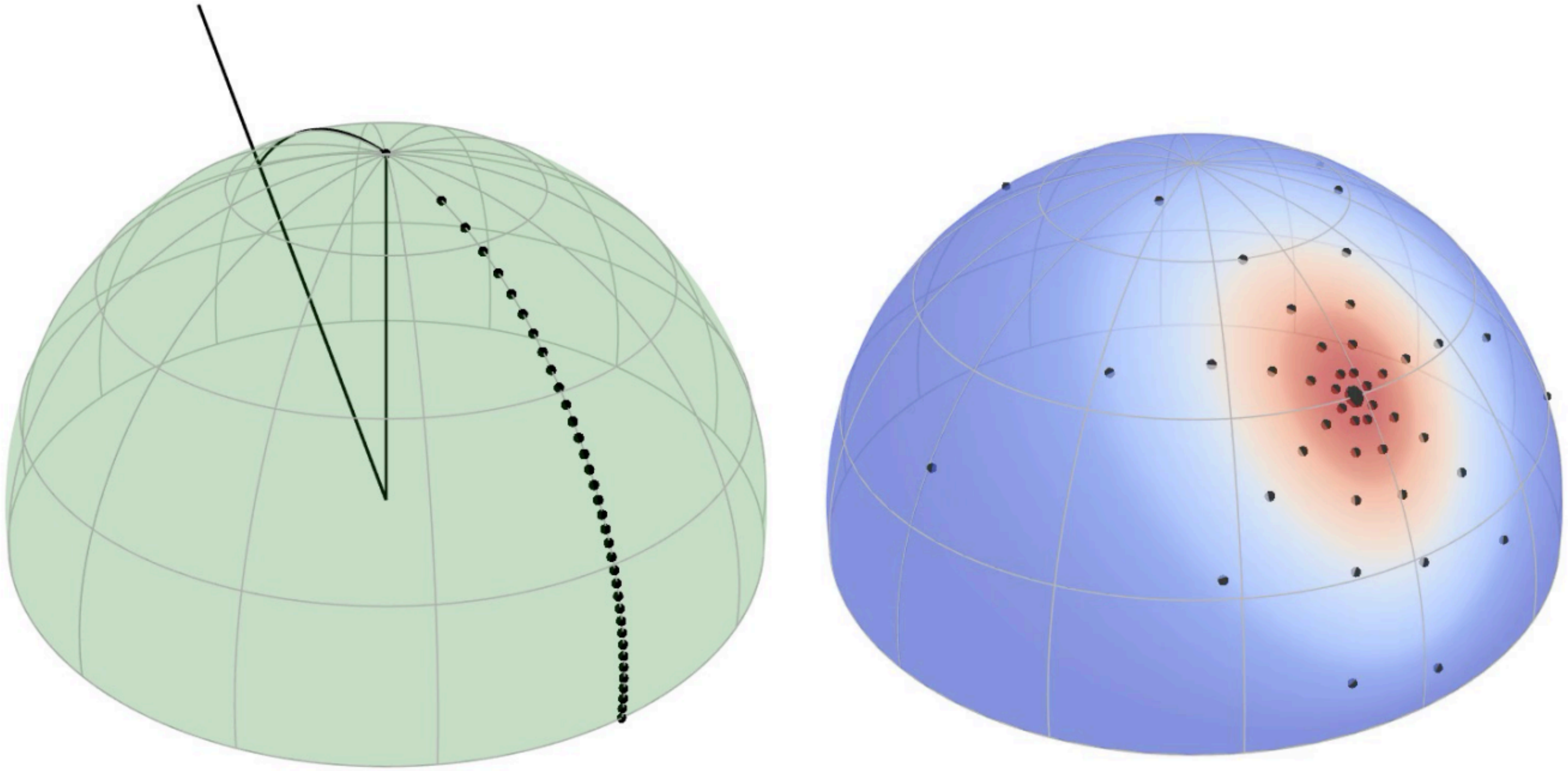
Summary of the course and weird tricks for your renderers

UCSD CSE 272
Advanced Image Synthesis
Tzu-Mao Li

What have we covered so far?

What have we covered so far?

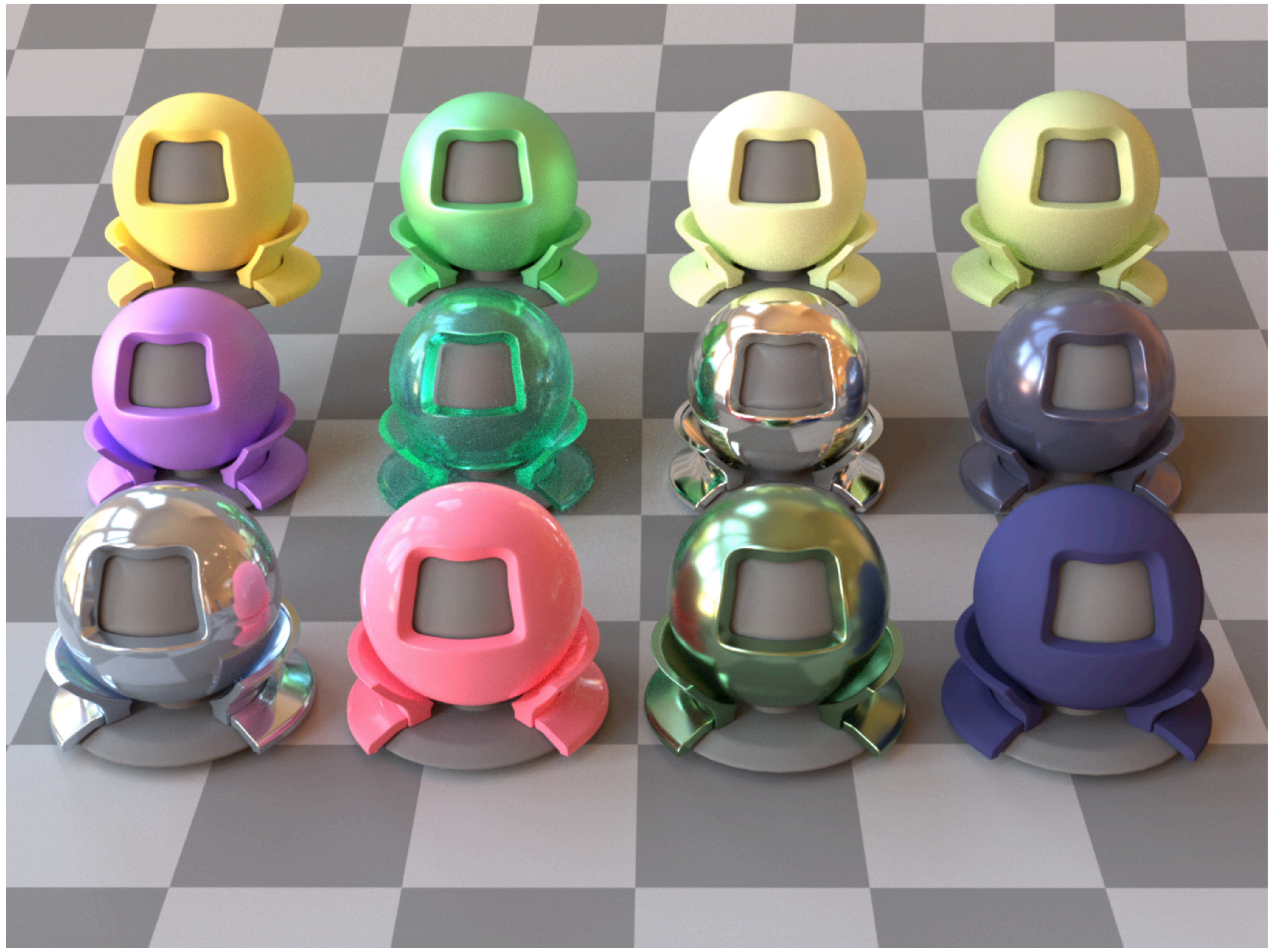
- BSDF measurement & microfacet theory, Schlick approximation



http://onrendering.com/data/papers/powitacq/slides/powitacq.html

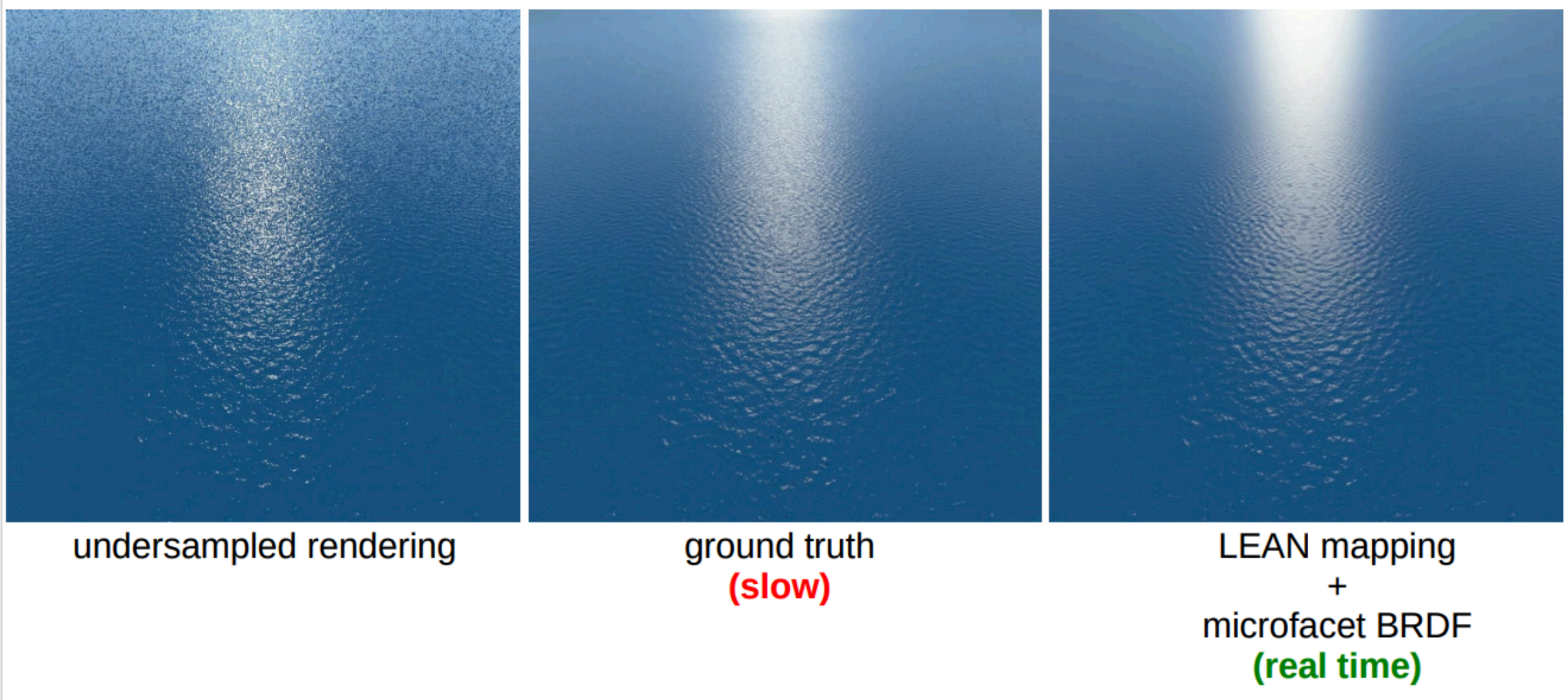
What have we covered so far?

- Uber BSDF



What have we covered so far?

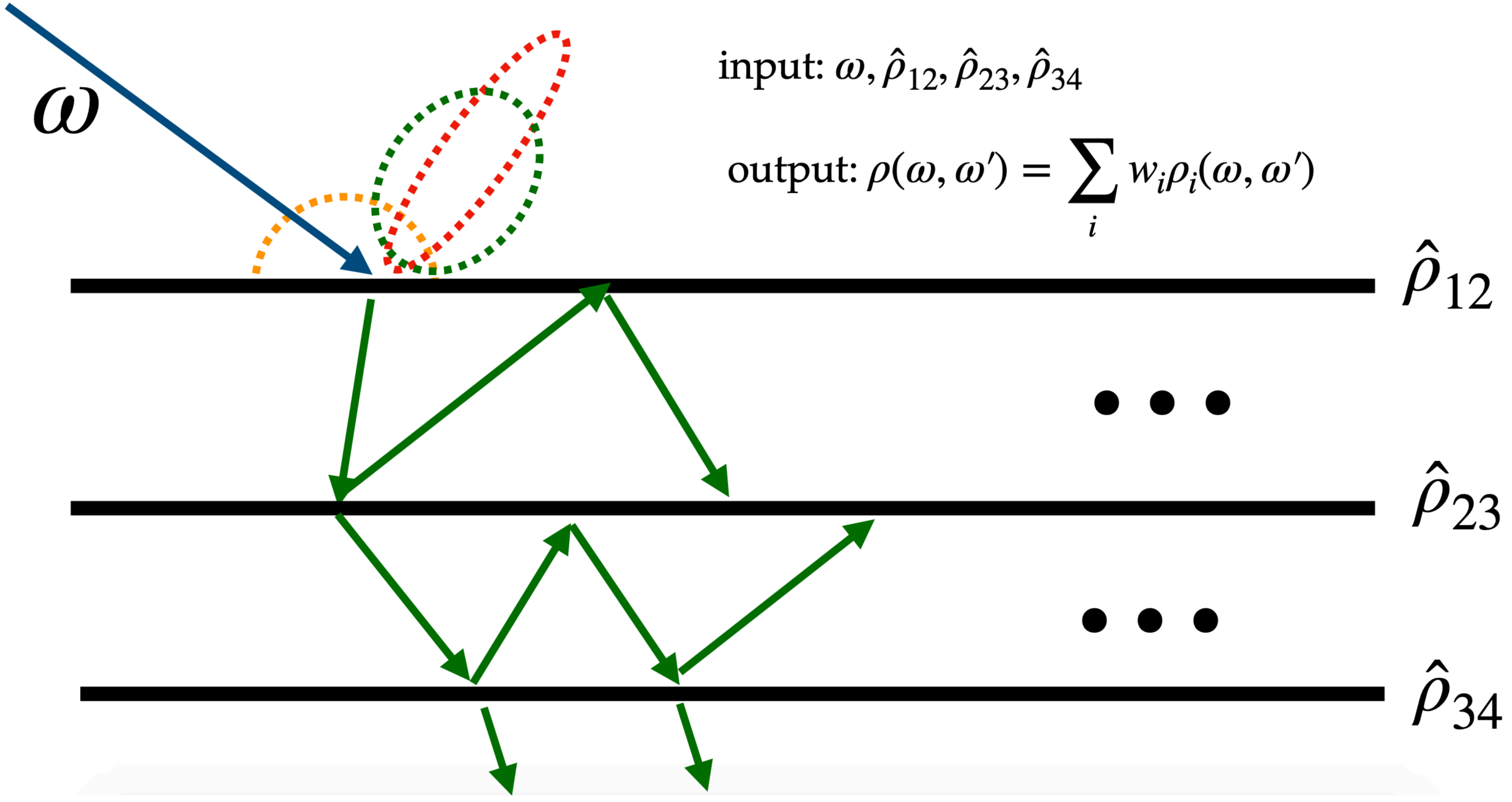
- normal map filtering



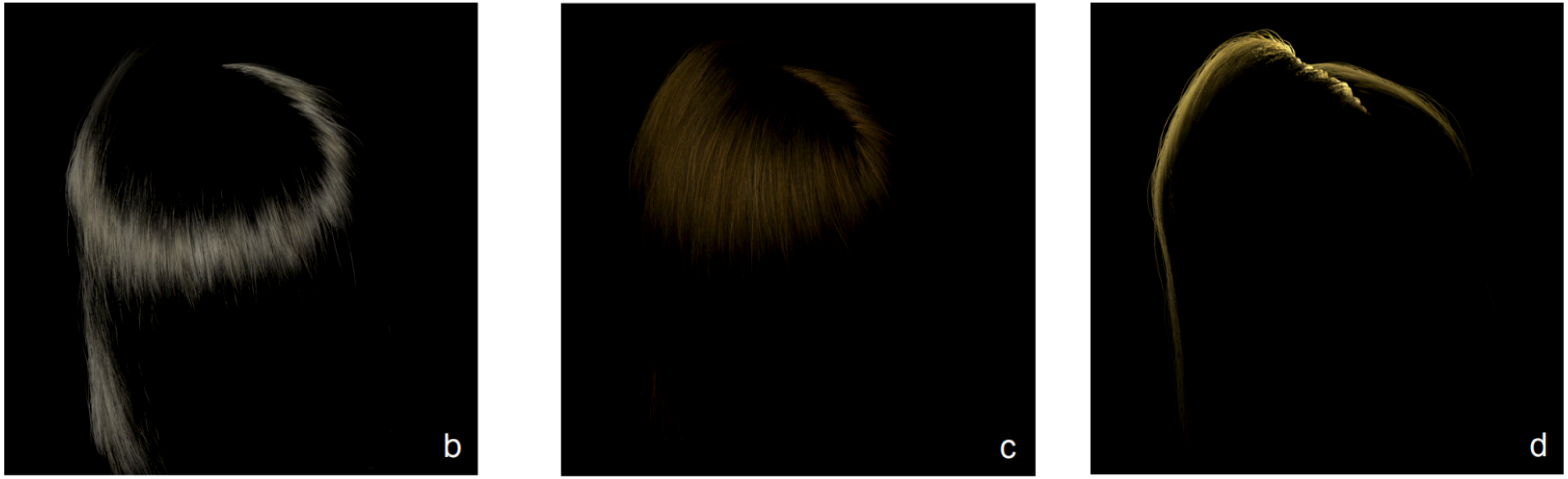
https://blog.selfshadow.com/publications/s2014-shading-course/dupuy/s2014_pbs_leadr_slides.pdf

What have we covered so far?

- layered BSDFs

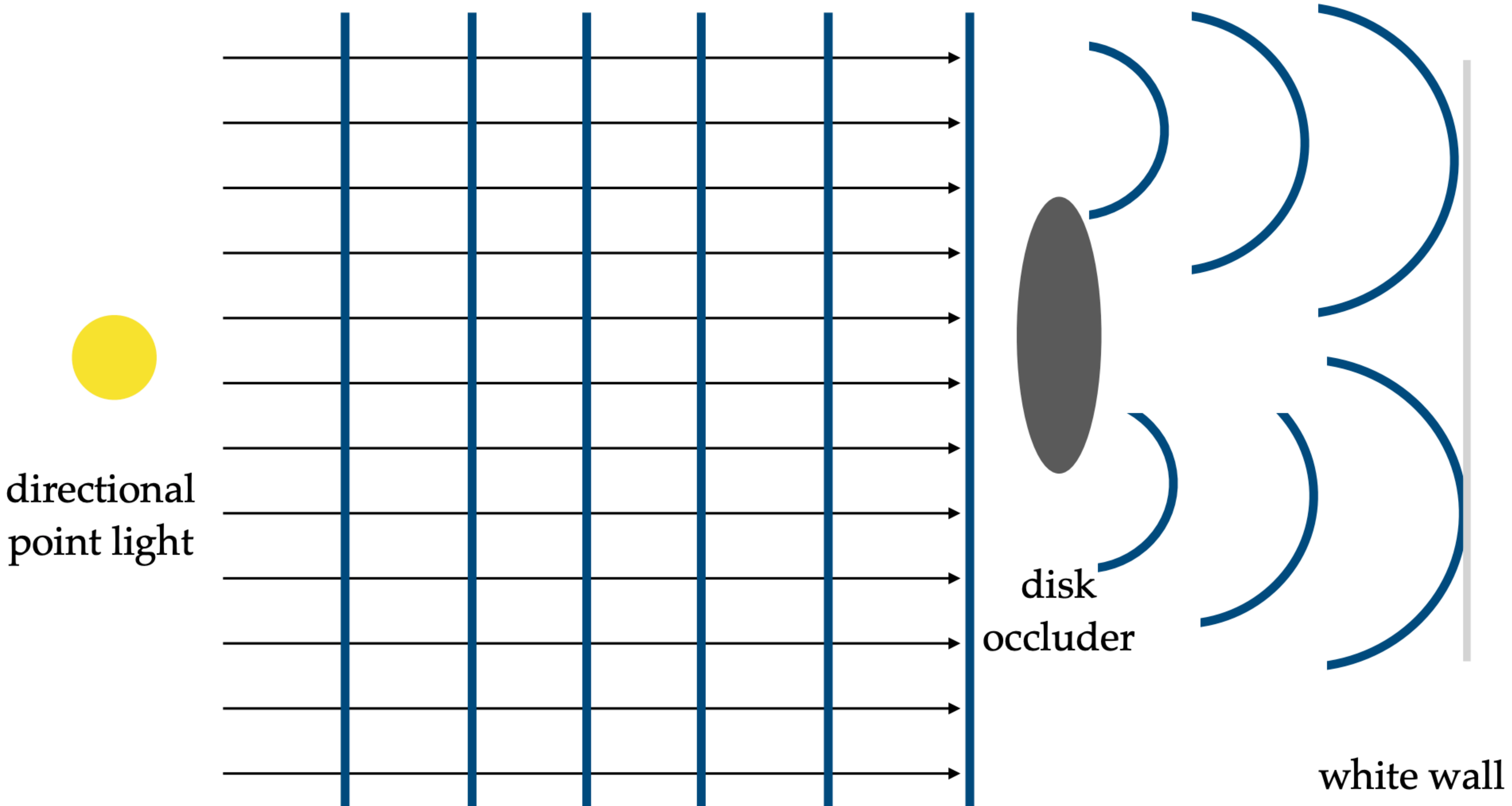


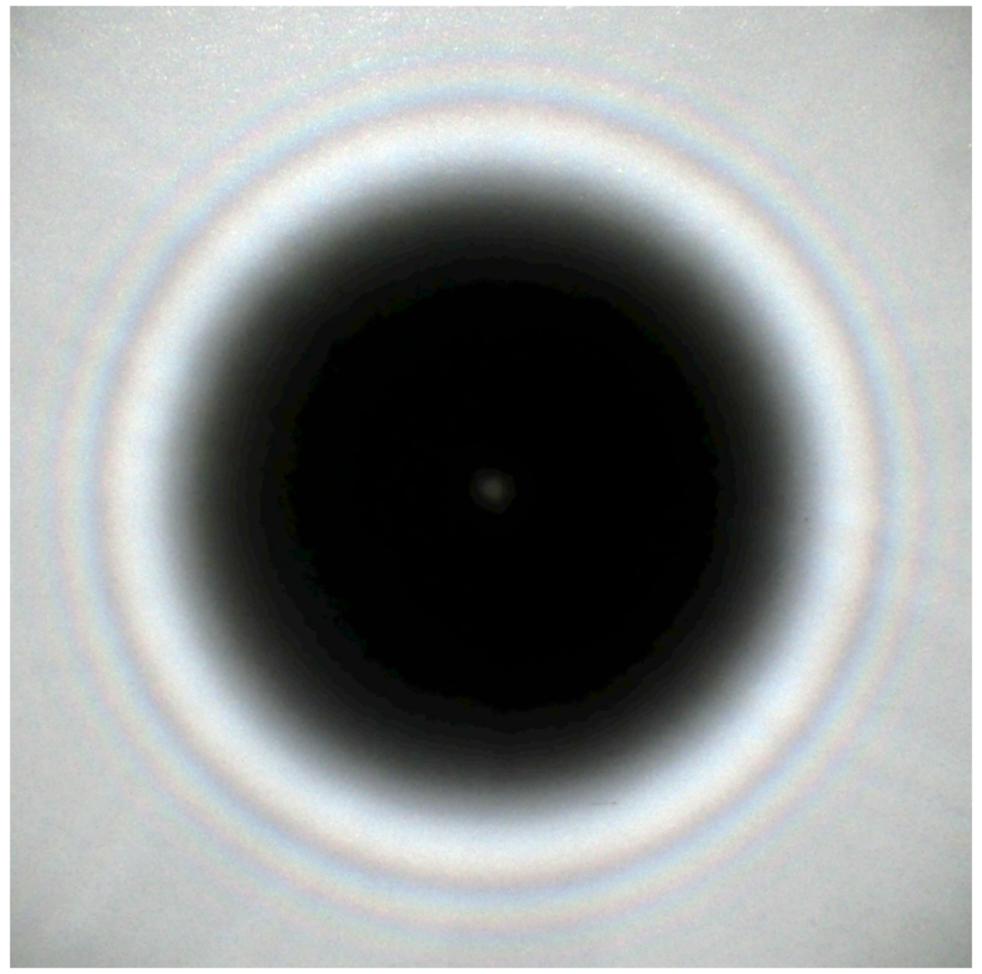
Hair and cloth rendering



R *TRT* *TT*

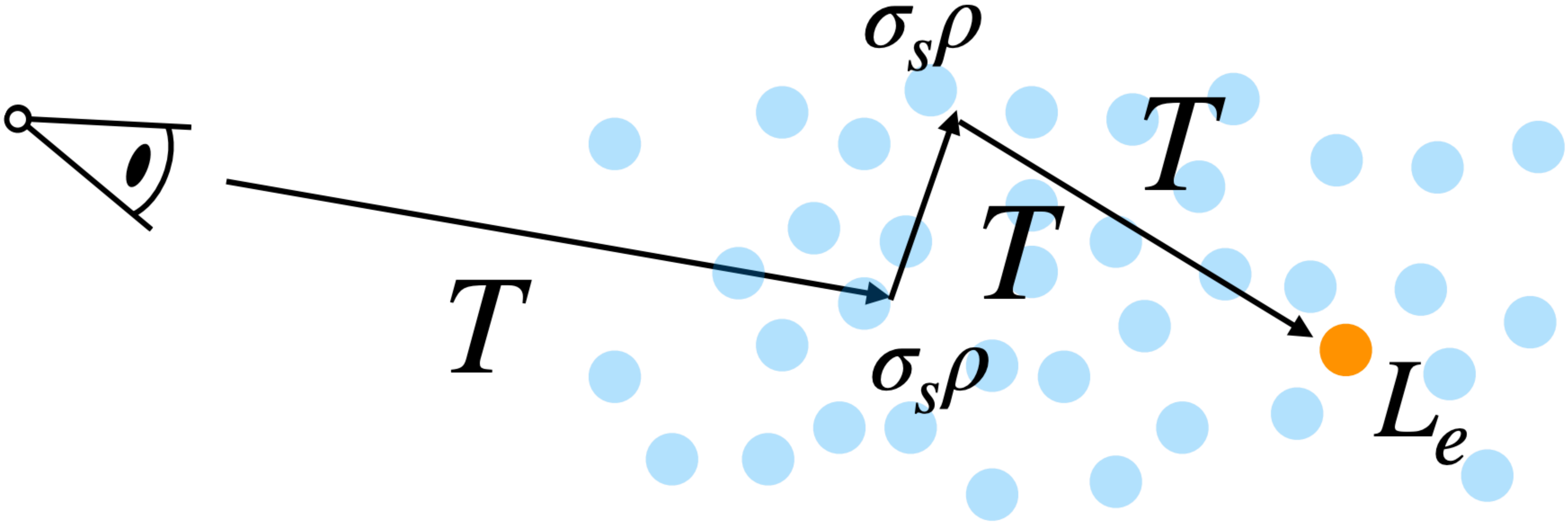
Wave optics



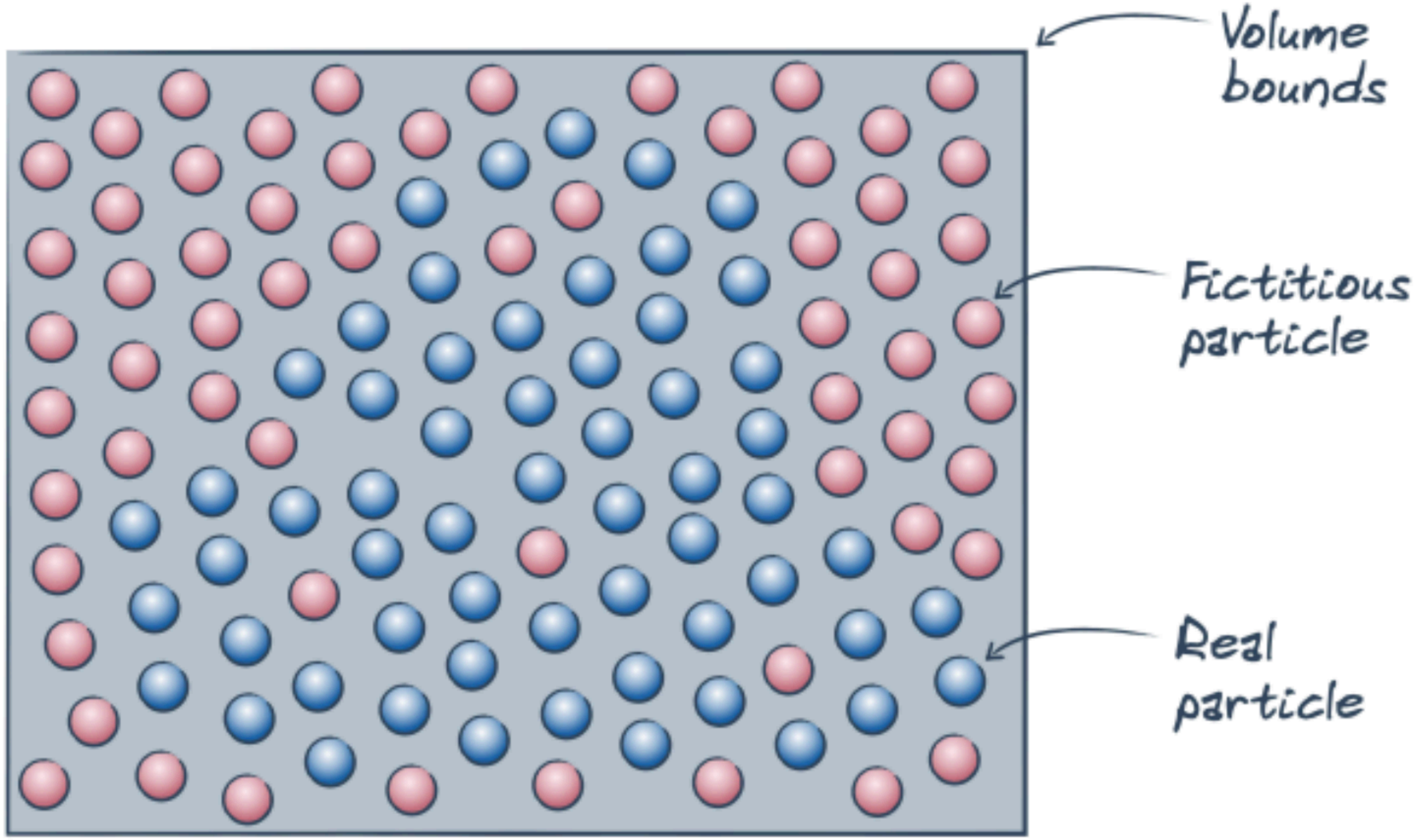


$$E_1 + E_2 = \Re\left(\left(A_1 e^{i\phi_1} + A_2 e^{i\phi_2}\right) e^{i\omega t}\right)$$

Radiative transfer equation

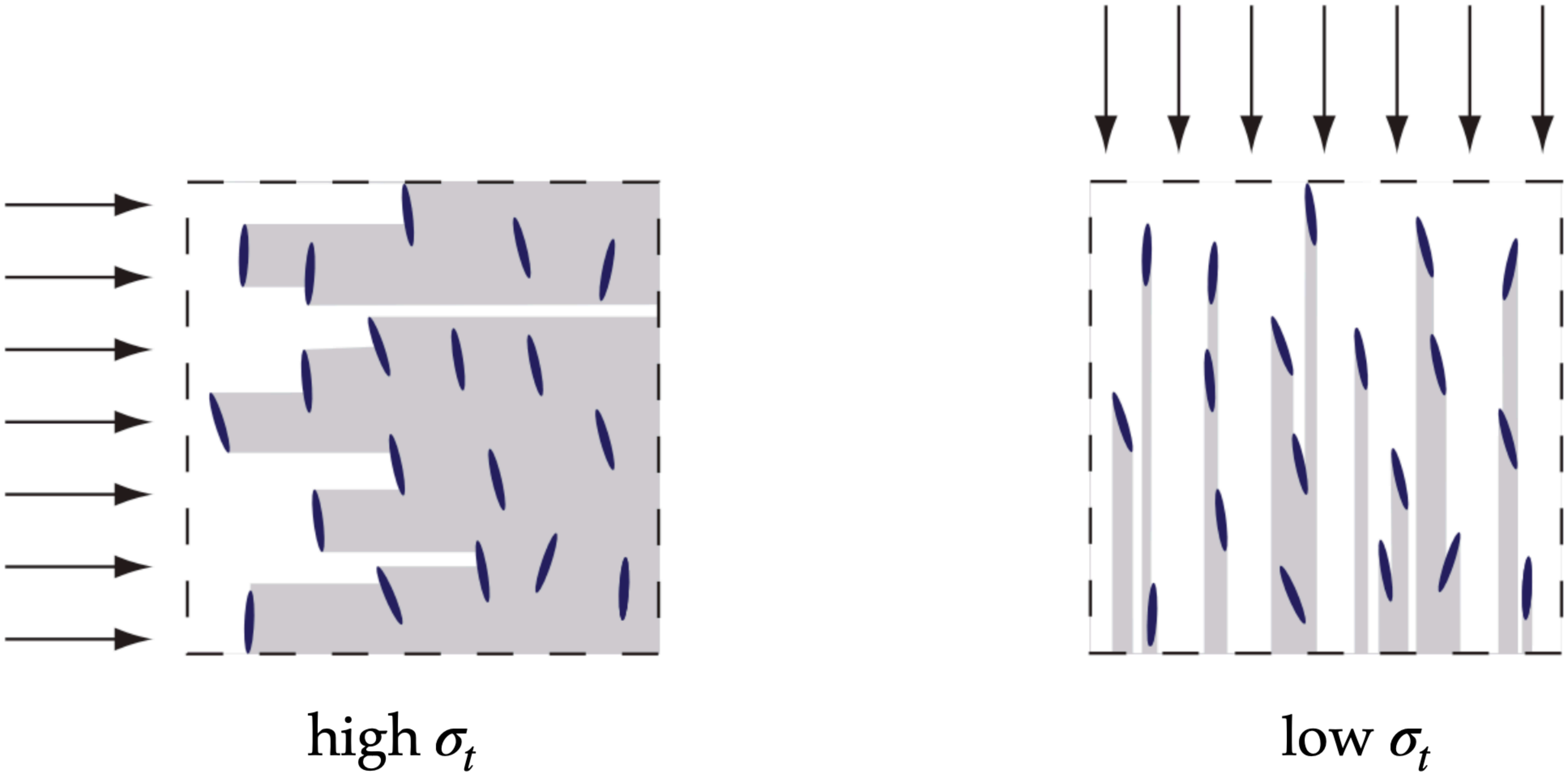


Transmittance estimation & null scattering



https://cs.dartmouth.edu/~wjarosz/publications/novak18monte-sig-slides-3-distance-sampling-notes.pdf

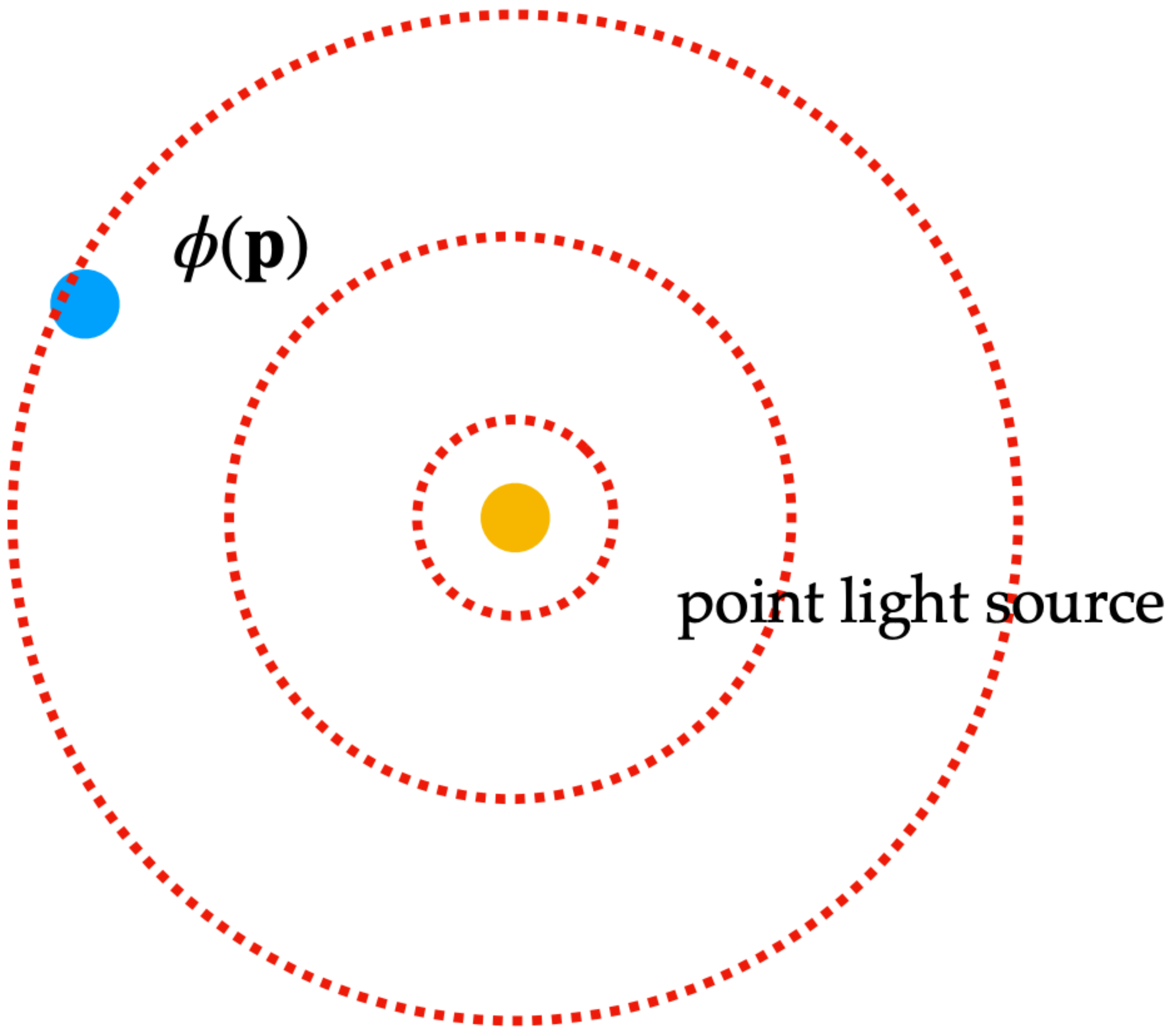
Microflake theory & SGGX



https://rgl.s3.eu-central-1.amazonaws.com/media/papers/Jakob2010Radiative_1.pdf

Diffusion approximation

$$\frac{1}{3\sigma_t}\Delta\phi(\mathbf{p}) = \sigma_a\phi(\mathbf{p}) - Q_0(\mathbf{p}) + \frac{1}{\sigma_t}\nabla\cdot Q_1(\mathbf{p})$$

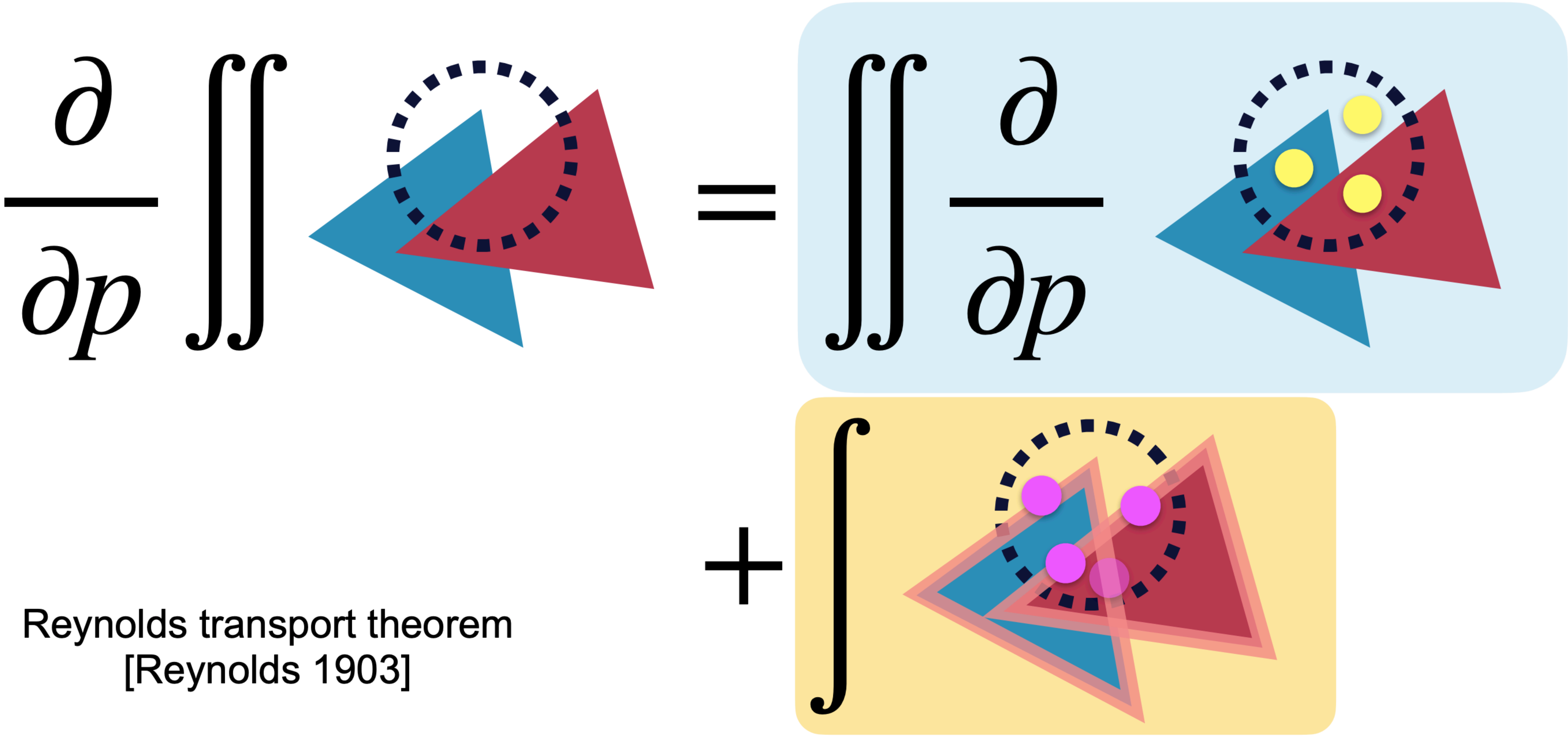


$$Q_0 = \delta(\mathbf{p})$$

$$Q_1 = 0$$

Differentiable rendering: edge sampling

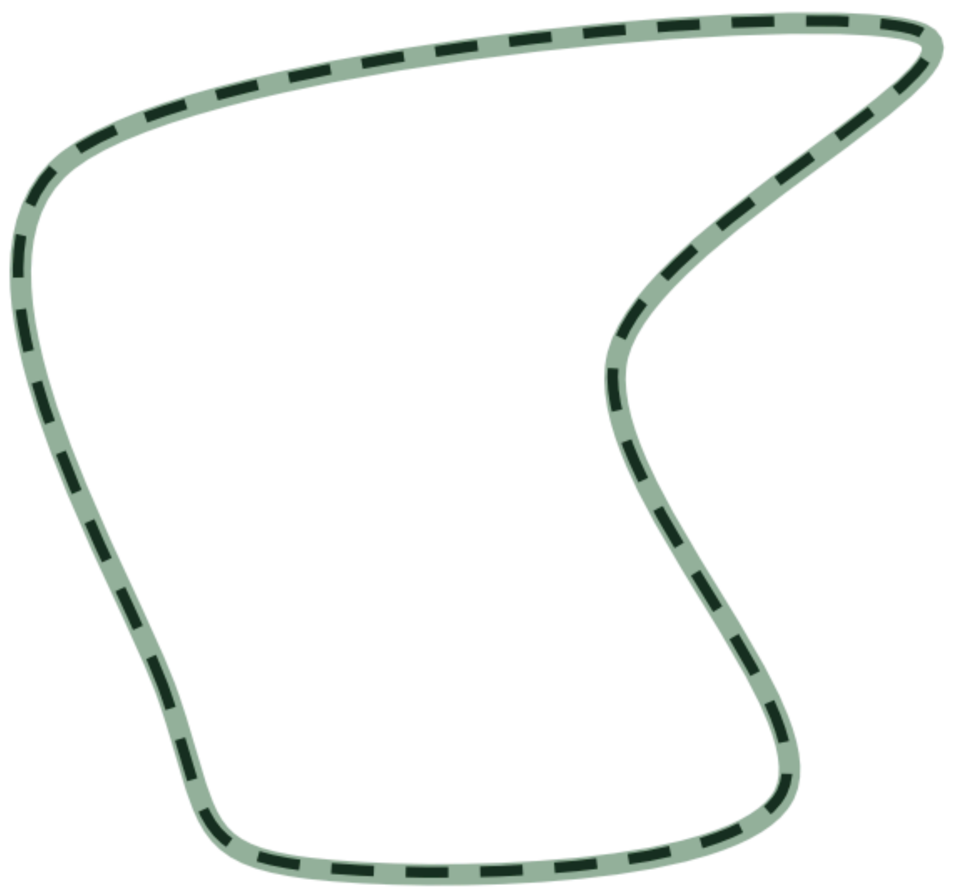
interior derivative

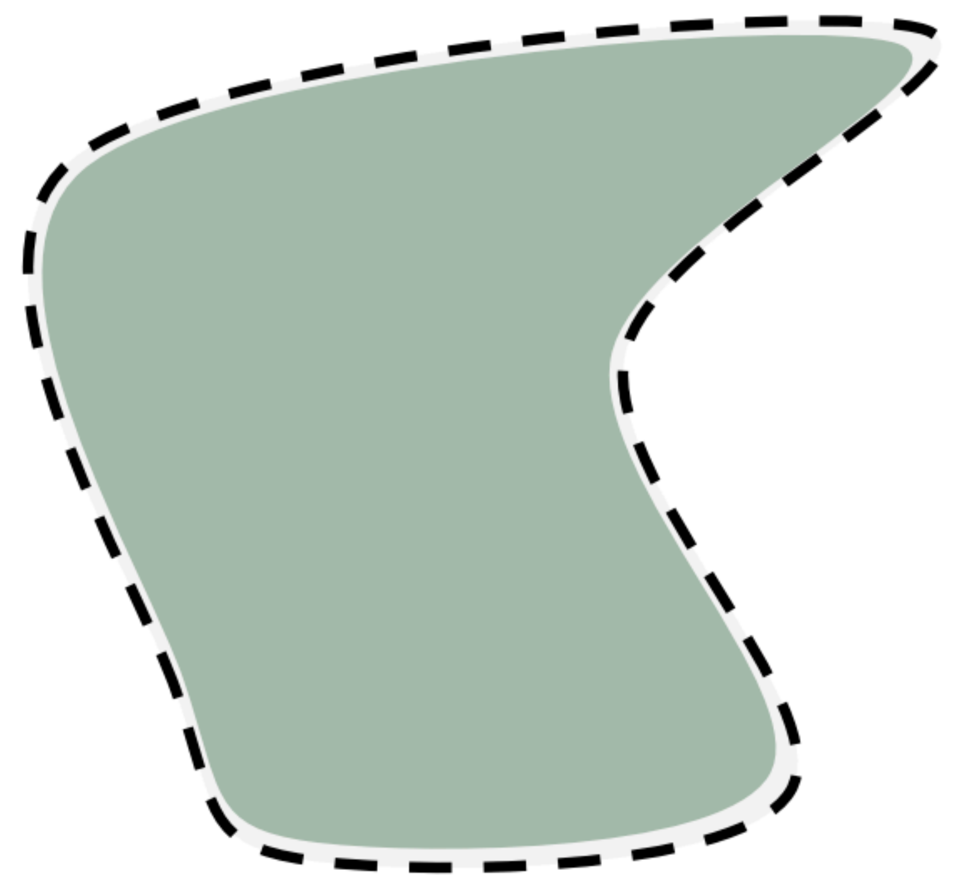


Reynolds transport theorem
[Reynolds 1903]

boundary derivative

Differentiable rendering: warped-area sampling





$$\int_{\partial \mathrm{D}} \vec{f} \cdot \vec{n} = \int_{\mathrm{D}} \nabla \cdot \vec{f}$$

Differentiable rendering: importance sampling

$$\int f(x,p)\,\mathrm{d}x$$

change of variable

$$\int f(T(u,p),p)\left|T'(u,p)\right|\,\mathrm{d}u$$

differentiation

$$\int \frac{\mathrm{d}}{\mathrm{d}p}\left[f(T(u,p),p)\left|T'(u,p)\right|\right]\mathrm{d}u$$

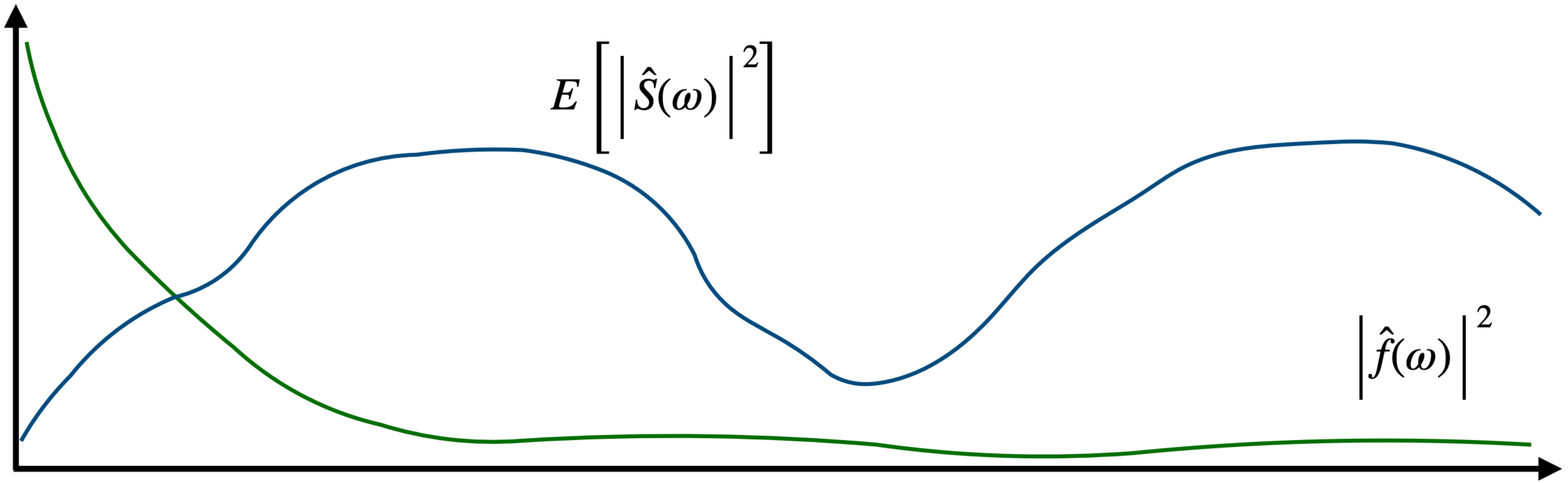
differentiation

$$\int \frac{\mathrm{d}}{\mathrm{d}p} f(x,p)\,\mathrm{d}x$$

change of variable

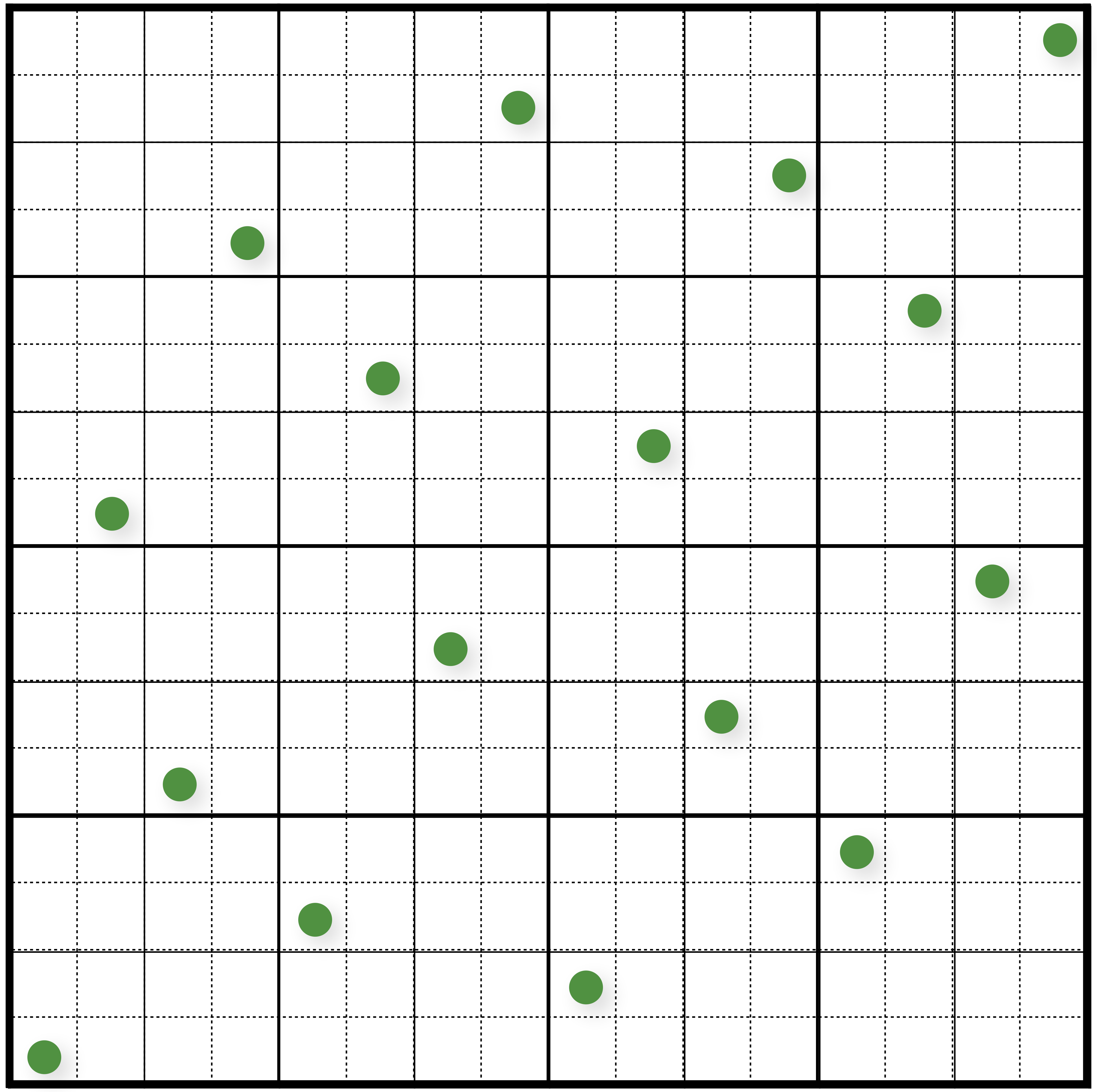
$$\int \frac{\mathrm{d}}{\mathrm{d}p}\left[f(x,p)\right]_{x\to T(u,p)}\left|T'(u,p)\right|\,\mathrm{d}u$$

Stratification

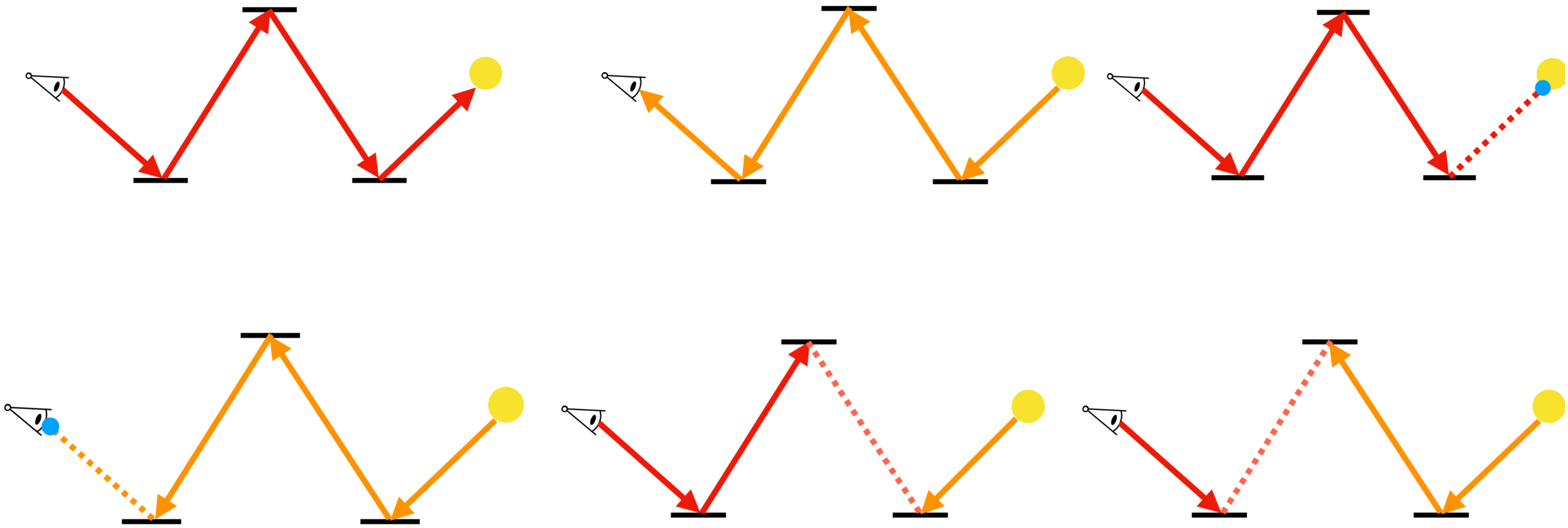


$$\text{variance} = \int \left|\hat{f}(\omega)\right|^2 E\left[\left|\hat{S}(\omega)\right|^2\right] \mathrm{d}\omega$$

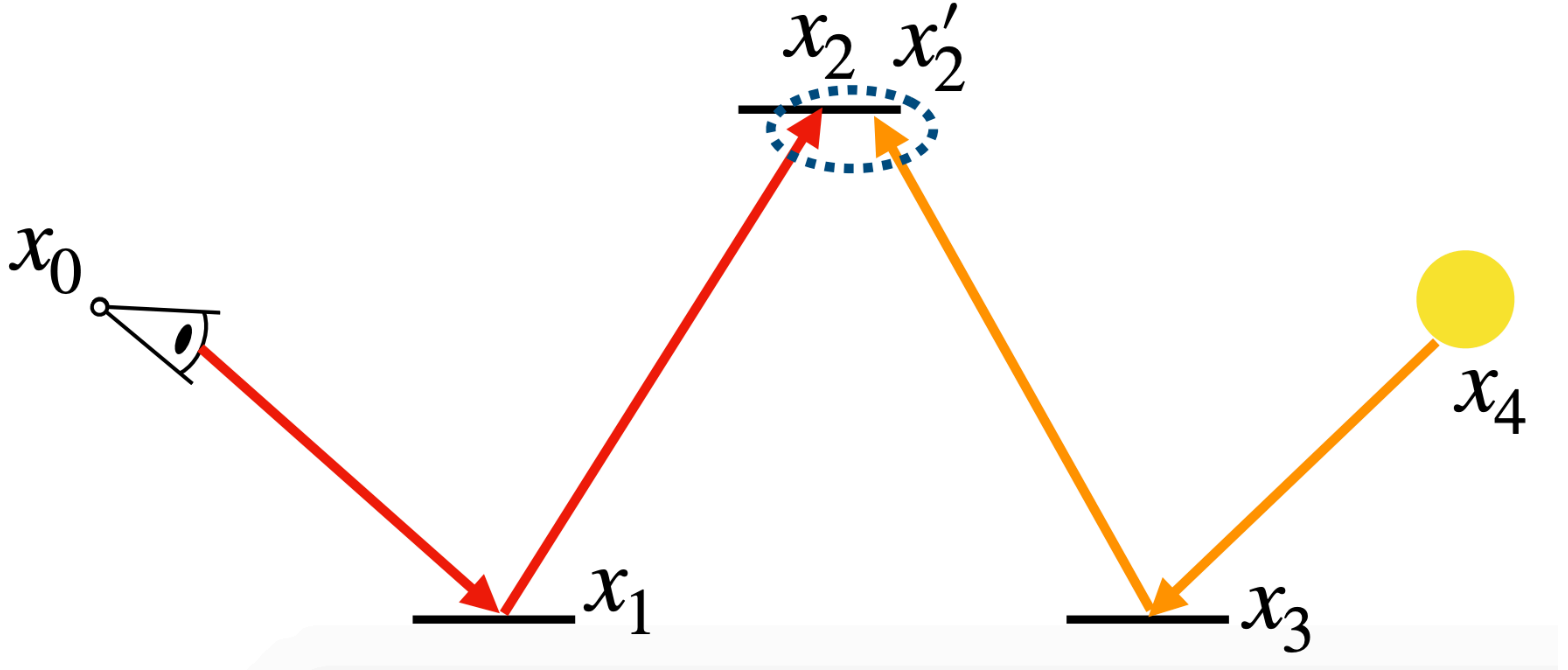
Low-discrepancy sequences



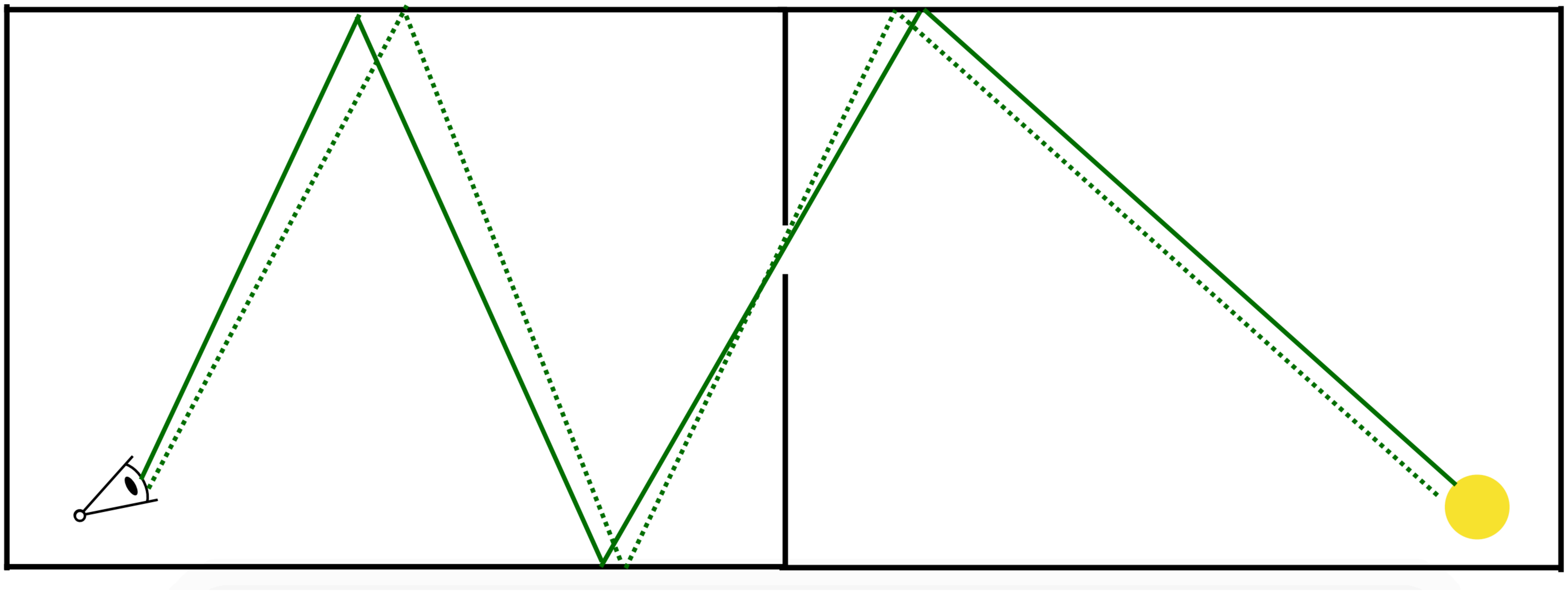
Path-space



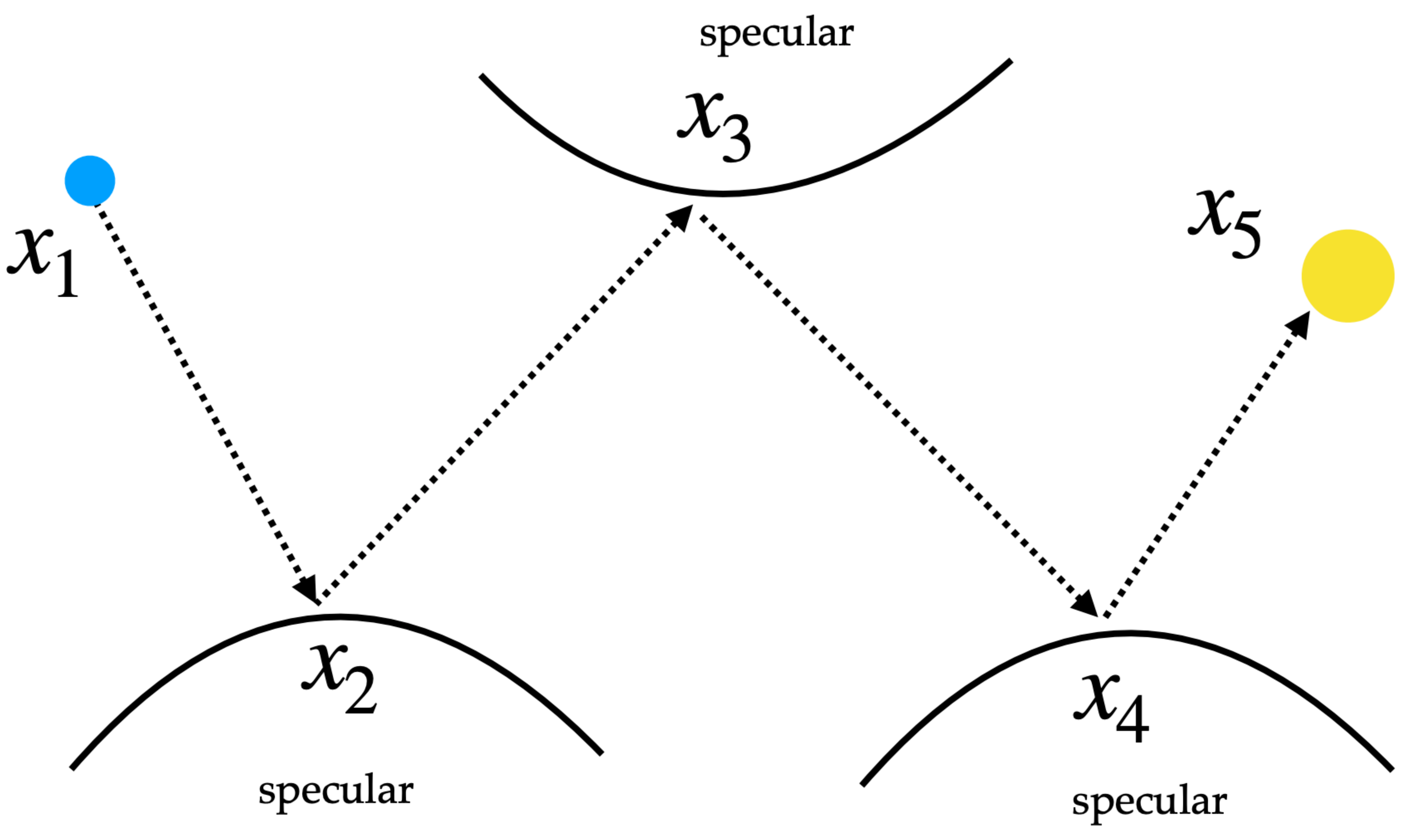
Photon mapping



Metropolis light transport



Specular light path rendering



Multiple importance sampling++

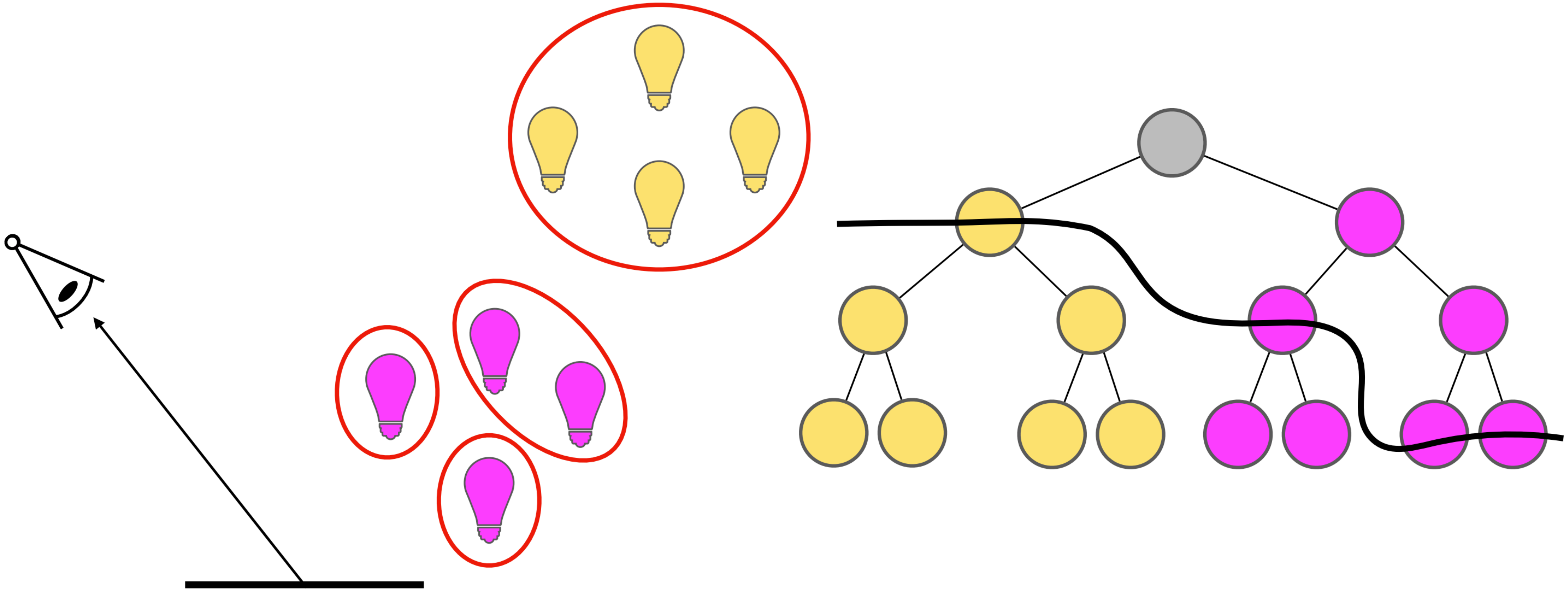
$$w_i = \frac{p_i}{f} a_i - p_i \frac{\sum_j \frac{p_j}{f} a_j - 1}{\sum_j p_j}$$

$$A_{ij} = \int \frac{p_i p_j}{\sum_k p_k}$$

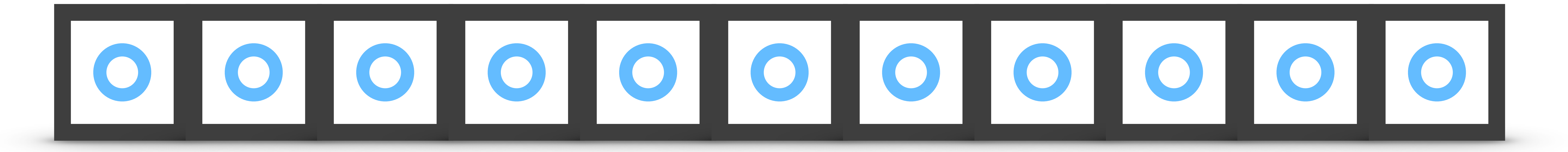
$$\begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

$$b_i = \int \frac{f p_i}{\sum_k p_k}$$

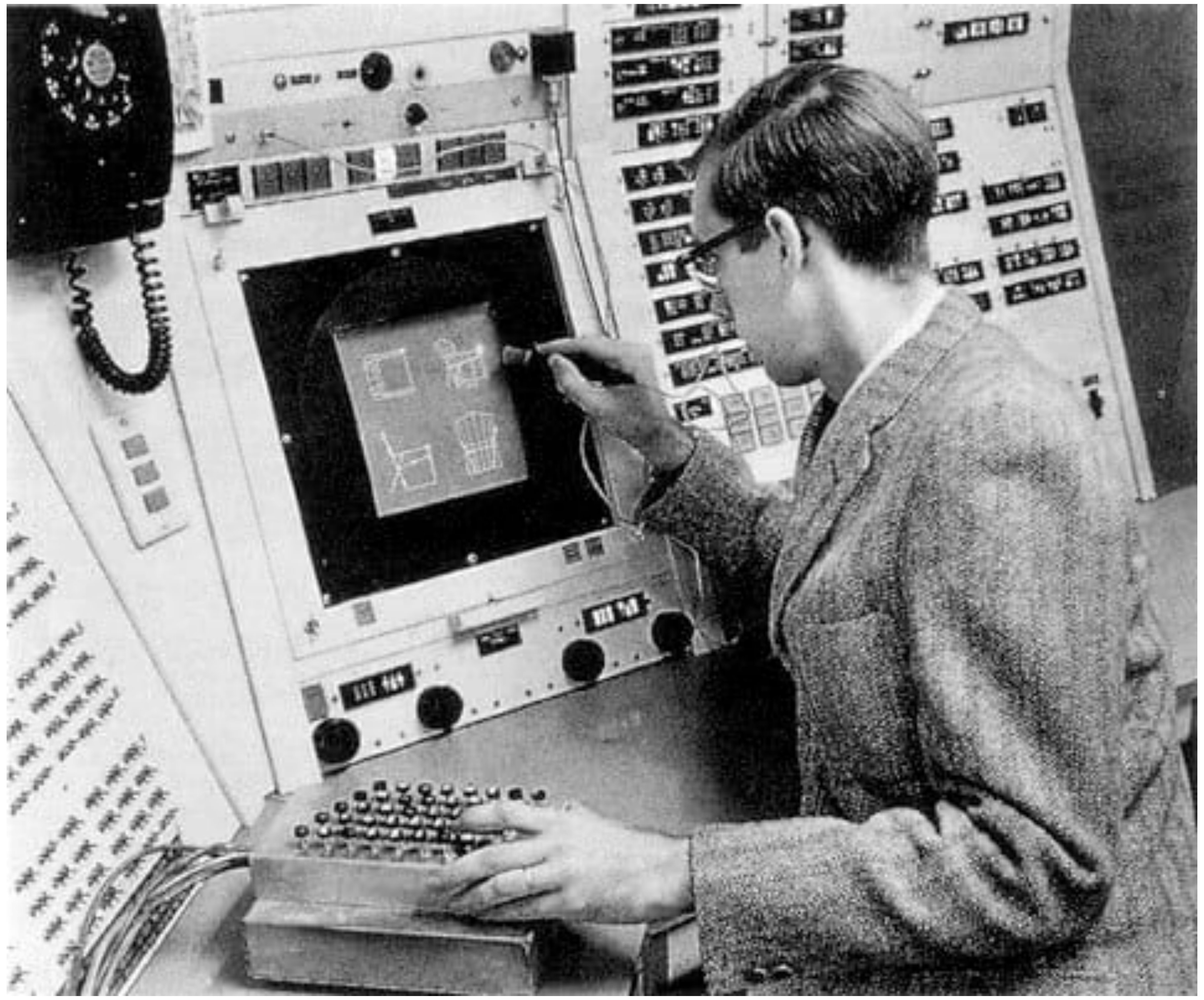
Many-lights rendering



ReSTIR



History of Computer Animation



Production rendering

RenderMan: An Advanced Path Tracing Architecture for Movie Rendering

PER CHRISTENSEN, JULIAN FONG, JONATHAN SHADE, WAYNE WOOTEN, BRENDEN SCHUBERT, ANDREW KENSLER, STEPHEN FRIEDMAN, CHARLIE KILPATRICK, CLIFF RAMSHAW, MARC BANNISTER, BRENTON RAYNER, JONATHAN BROUILLAT, and MAX LIANI, Pixar Animation Studios

The Design and Evolution of Disney's Hyperion Renderer

BRENT BURLEY, DAVID ADLER, MATT JEN-YUAN CHIANG, HANK DRISKILL, RALF HABEL, PATRICK KELLY, PETER KUTZ, YINING KARL LI, and DANIEL TEECE, Walt Disney Animation Studios

Vectorized Production Path Tracing

Mark Lee
DreamWorks Animation

Brian Green
DreamWorks Animation

Feng Xie
DreamWorks Animation

Eric Tabellion
DreamWorks Animation

Sony Pictures Imageworks Arnold

CHRISTOPHER KULLA, Sony Pictures Imageworks
ALEJANDRO CONTY, Sony Pictures Imageworks
CLIFFORD STEIN, Sony Pictures Imageworks
LARRY GRITZ, Sony Pictures Imageworks

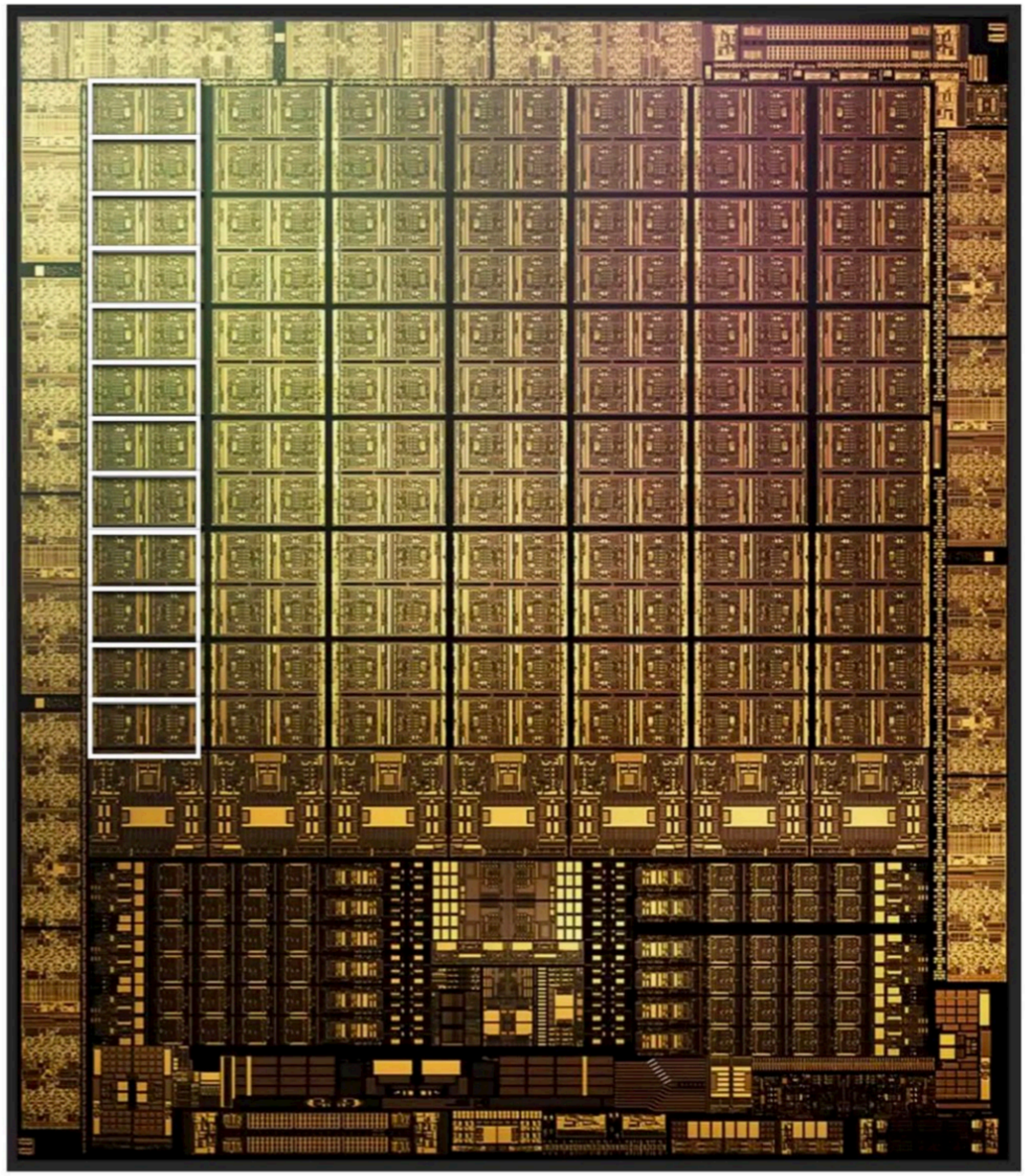
Arnold: A Brute-Force Production Path Tracer

ILIYAN GEORGIEV, THIAGO IZE, MIKE FARNSWORTH, RAMÓN MONTOYA-VOZMEDIANO, ALAN KING, BRECHT VAN LOMMEL, ANGEL JIMENEZ, OSCAR ANSON, SHINJI OGAKI, ERIC JOHNSTON, ADRIEN HERUBEL, DECLAN RUSSELL, FRÉDÉRIC SERVANT, and MARCOS FAJARDO, Solid Angle

Manuka: A batch-shading architecture for spectral path tracing in movie production

LUCA FASCIONE, JOHANNES HANIKA, MARK LEONE, MARC DROSKE, JORGE SCHWARZHAUPT, TOMÁŠ DAVIDOVIČ, ANDREA WEIDLICH, and JOHANNES MENG, Weta Digital

GPU architectures



Nanite

https://docs.unrealengine.com/5.0/en-US/RenderingFeatures/Nanite/

What haven't we covered?

- closed-form/analytical methods
- transient rendering
- speckle rendering
- refractive radiative transfer equation
- light baking/precomputed radiance transfer (in CSE 168)
- Monte Carlo denoising (in CSE 168)
- adaptive importance sampling (path guiding) (in CSE 168)

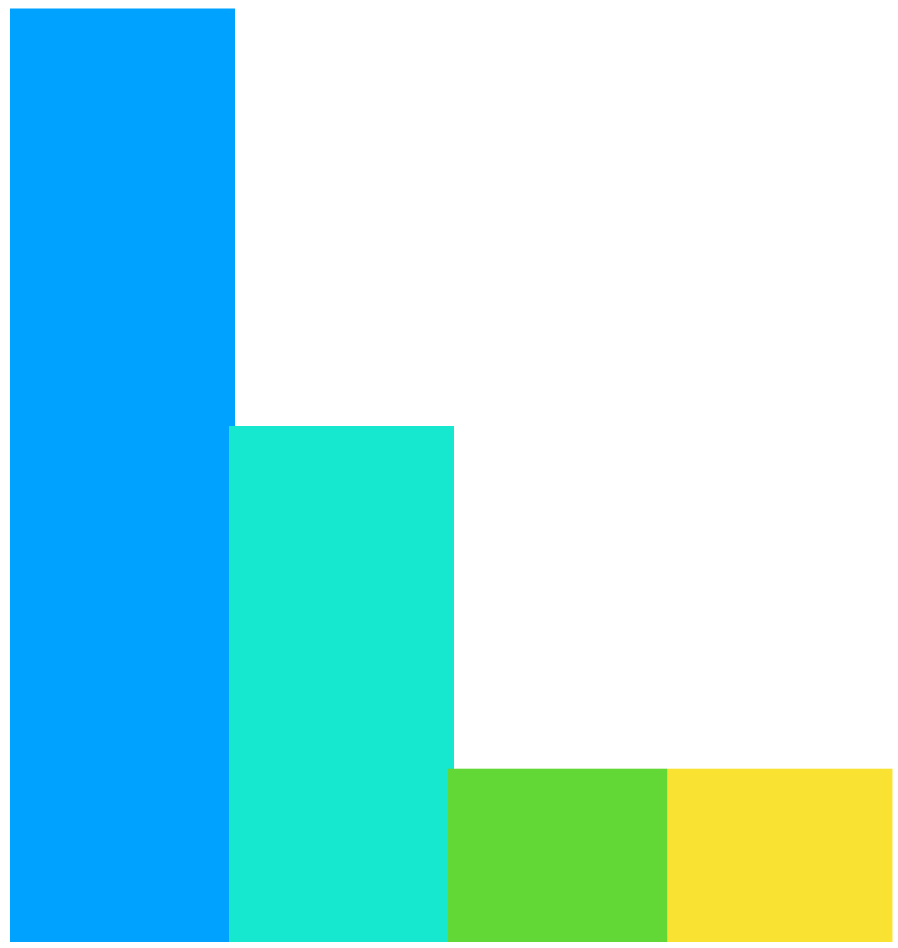
Fill the teaching evaluation!

- https://academicaffairs.ucsd.edu/Modules/Evals/
- important for future students and my career : >

Random tricks and stuff

Alias method

- goal: sampling from a discrete probability distribution
- lajolla's current implementation takes O(n) time to construct the CDF, O(log(n)) time to query
- alias method takes O(n) time to construct an "alias table", and takes O(1) time to query



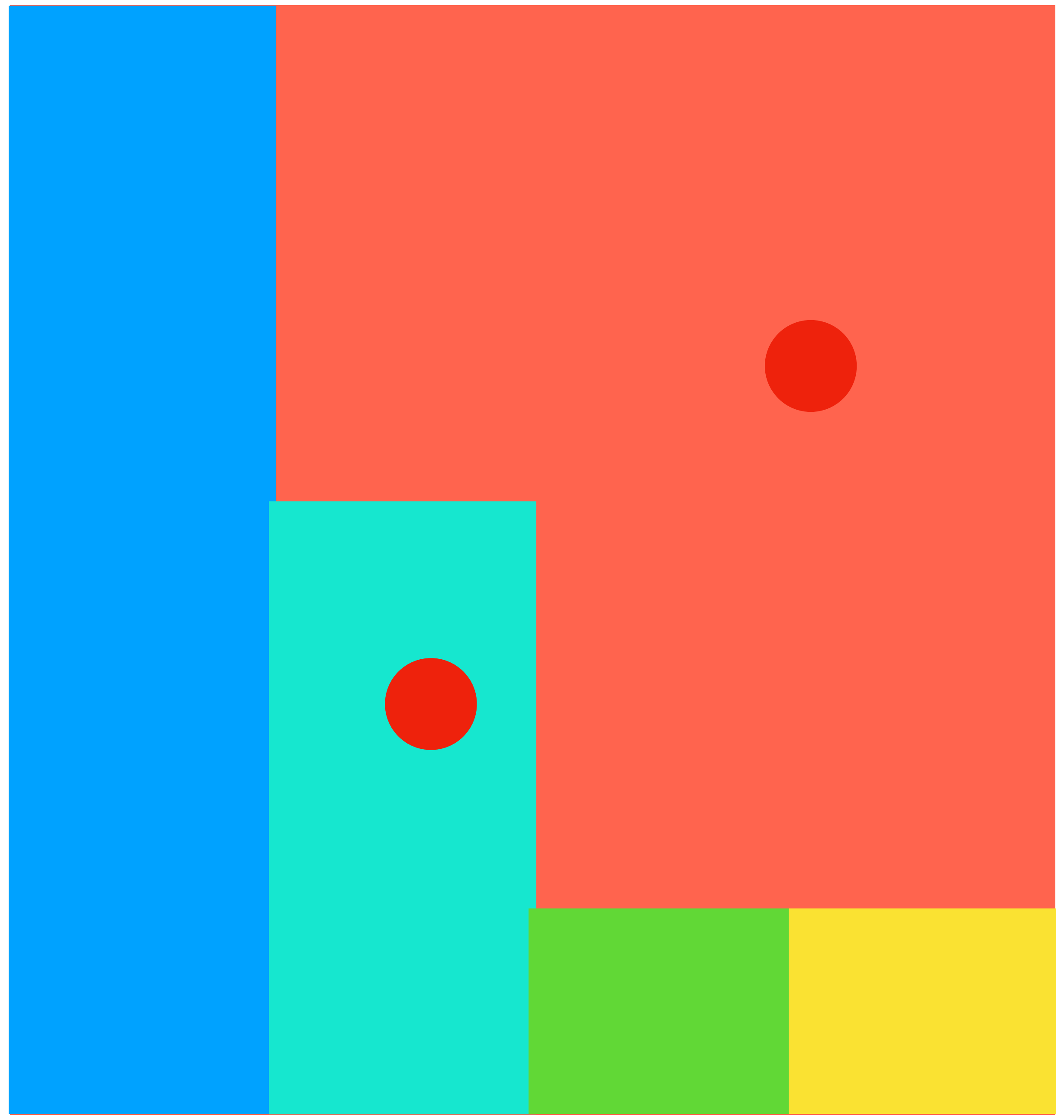
https://www.keithschwarz.com/darts-dice-coins/

Idea: sample from this rectangle

1. pick a point horizontally

2. accept/reject the sample by picking a point vertically

3. repeat until accept

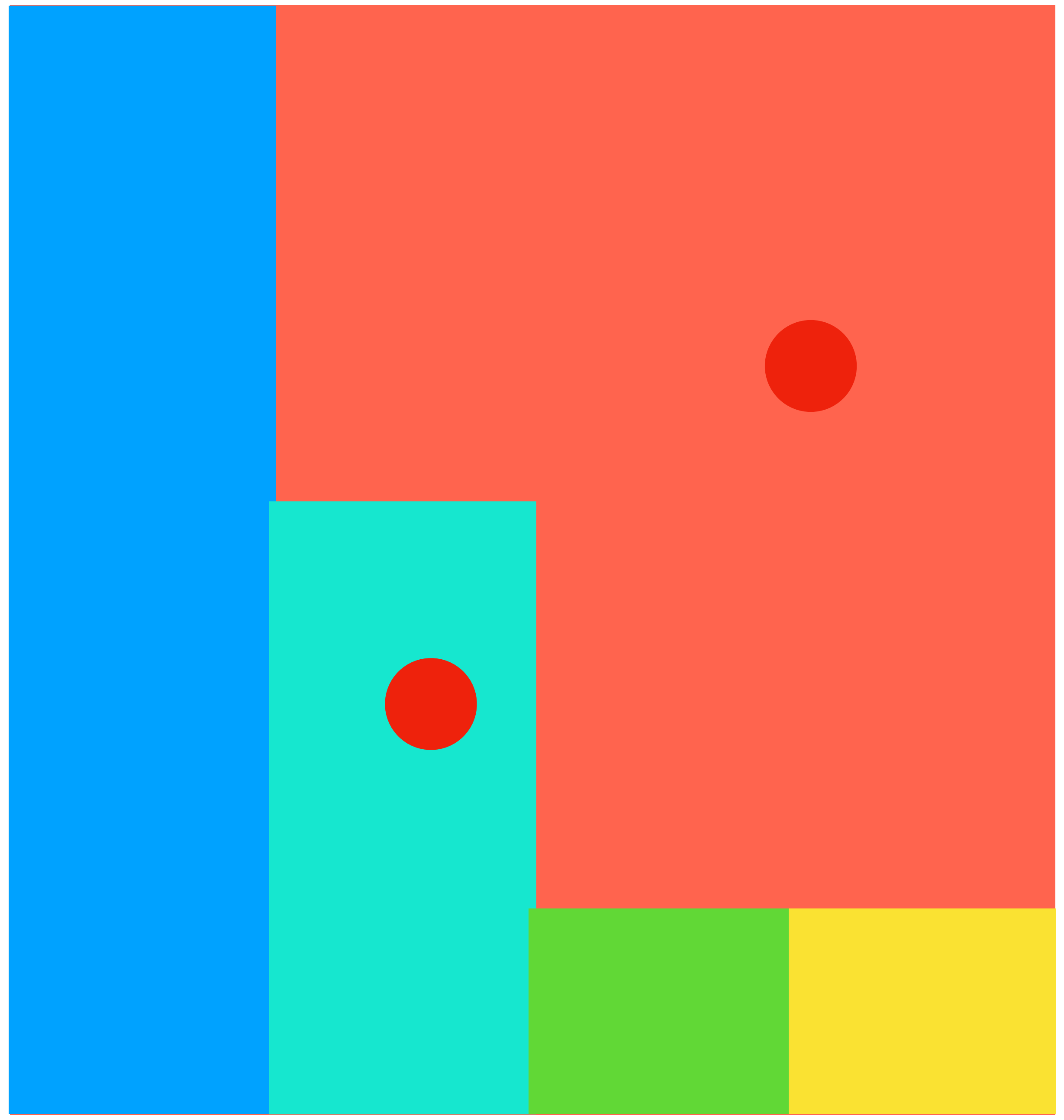


Idea: sample from this rectangle

1. pick a point horizontally

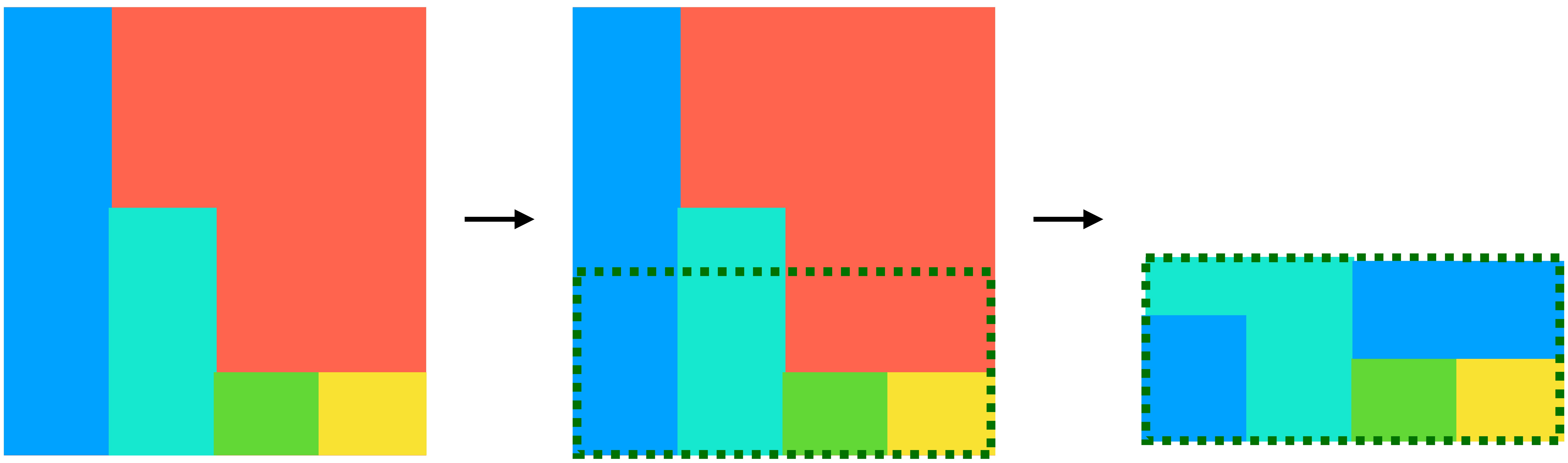
2. accept/reject the sample by picking a point vertically

3. repeat until accept

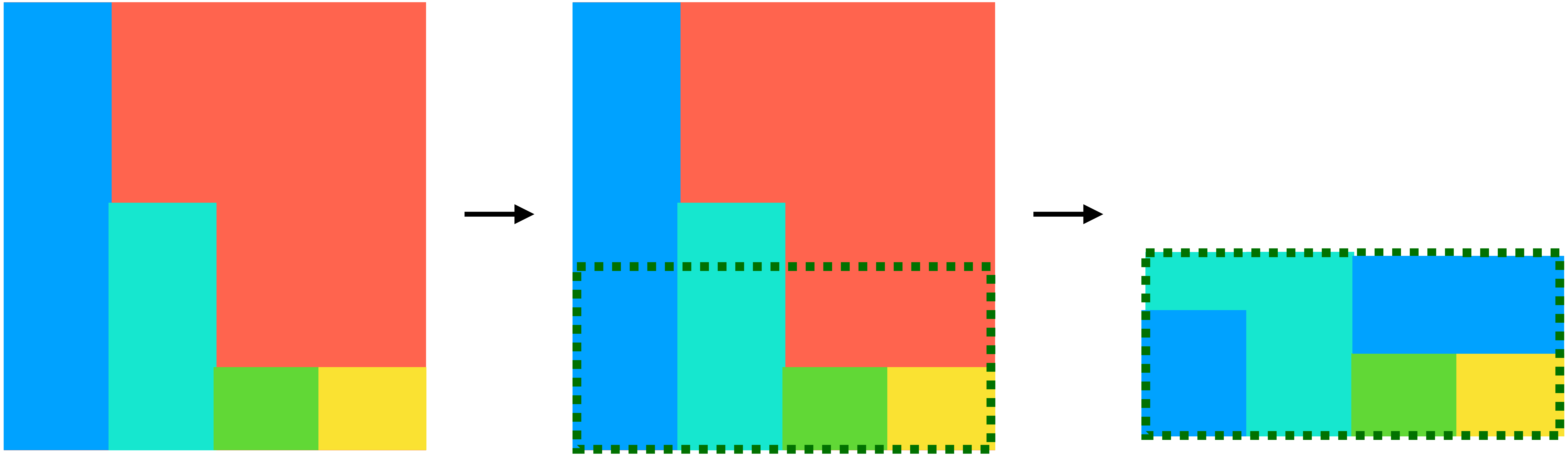


inefficient — can we improve this?

Idea: cut the rectangle and redistribute



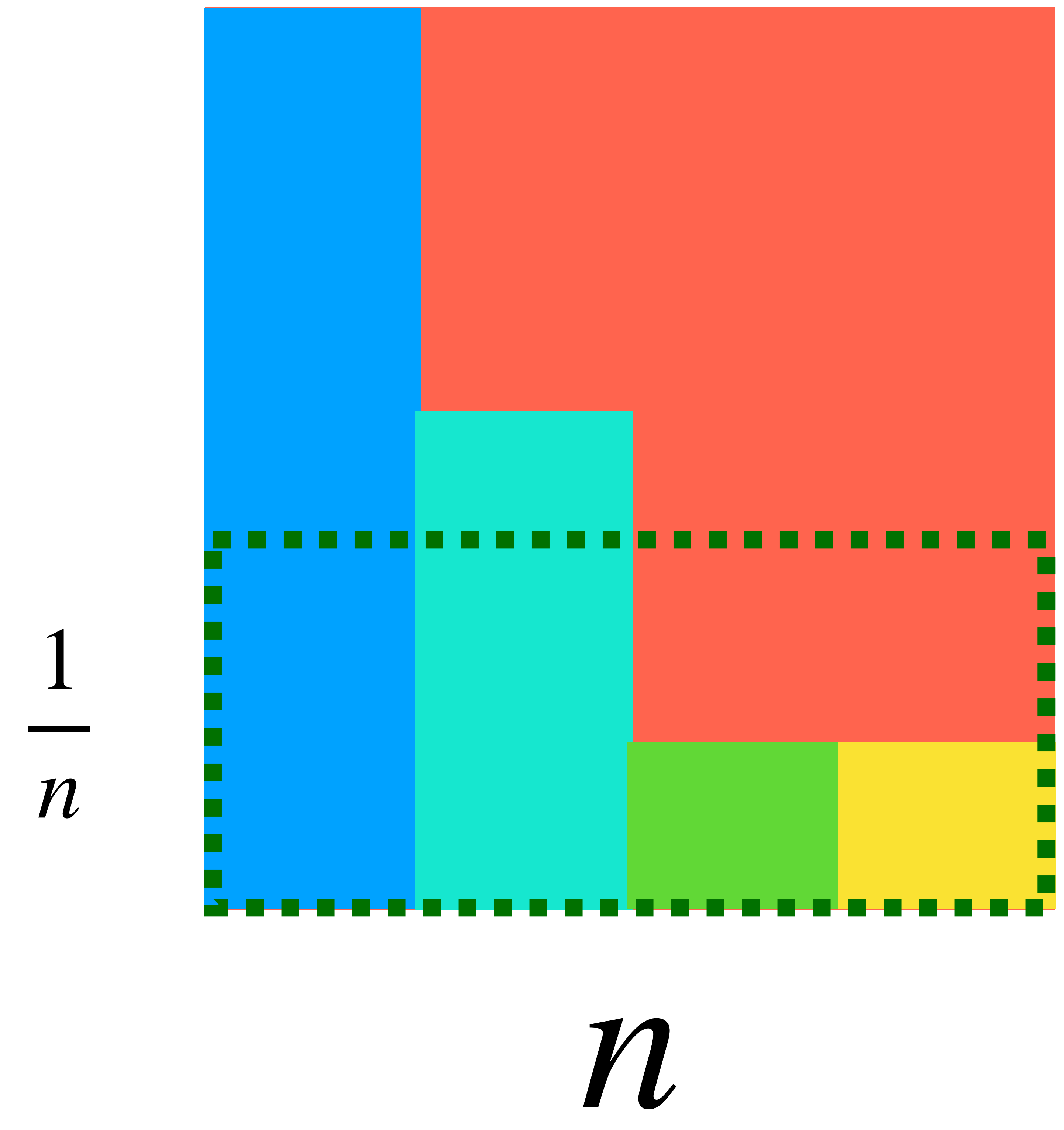
Idea: cut the rectangle and redistribute



- how large should the cut be?
- how do we redistribute?

How large should the cut be?

- we know the total probabilities sum to 1
- this means that to not waste space, if the width of the rectangle is n, the height must be $\frac{1}{n}$

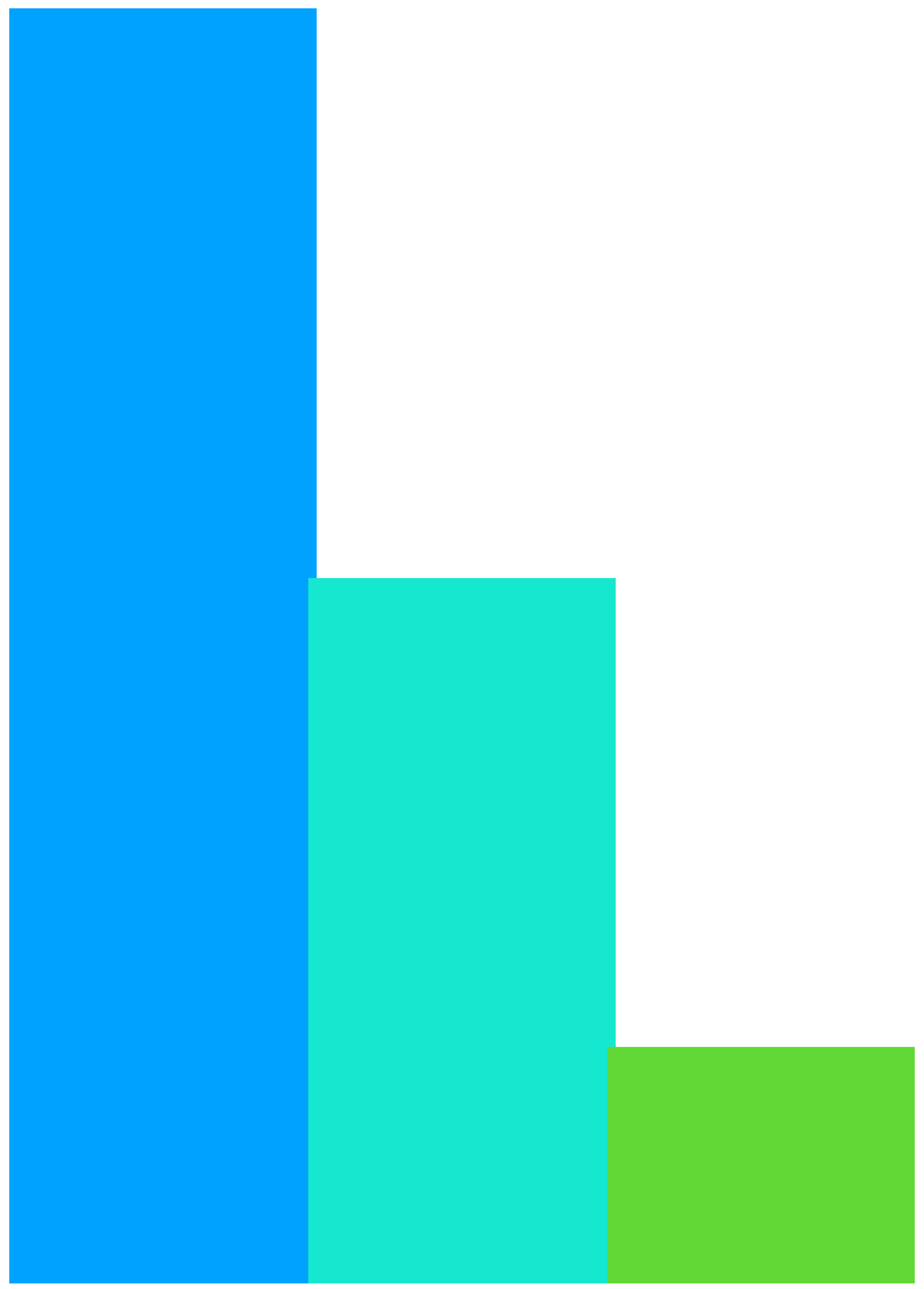


How do we redistribute?

- start from an empty rectangle

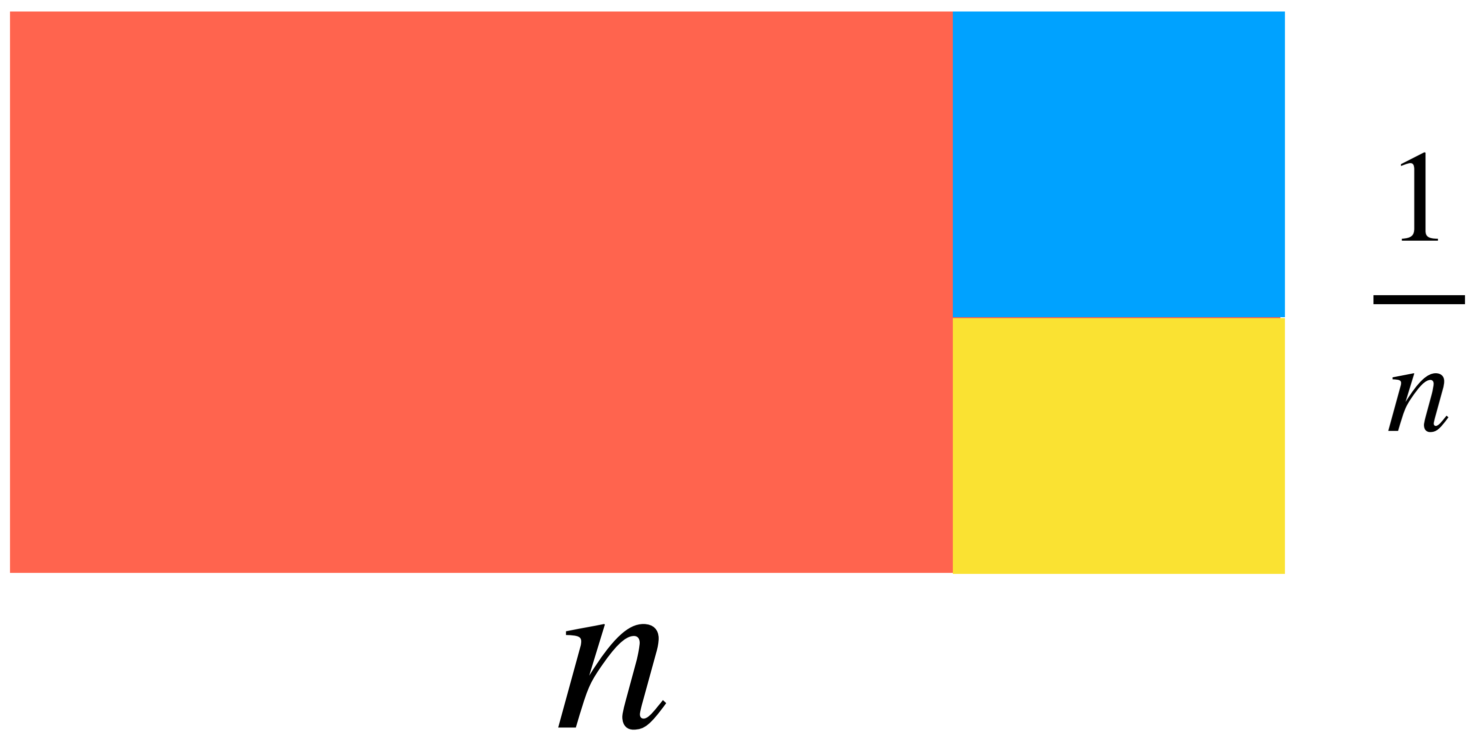
How do we redistribute?

- find a probability $< \frac{1}{n}$, put it on the rectangle



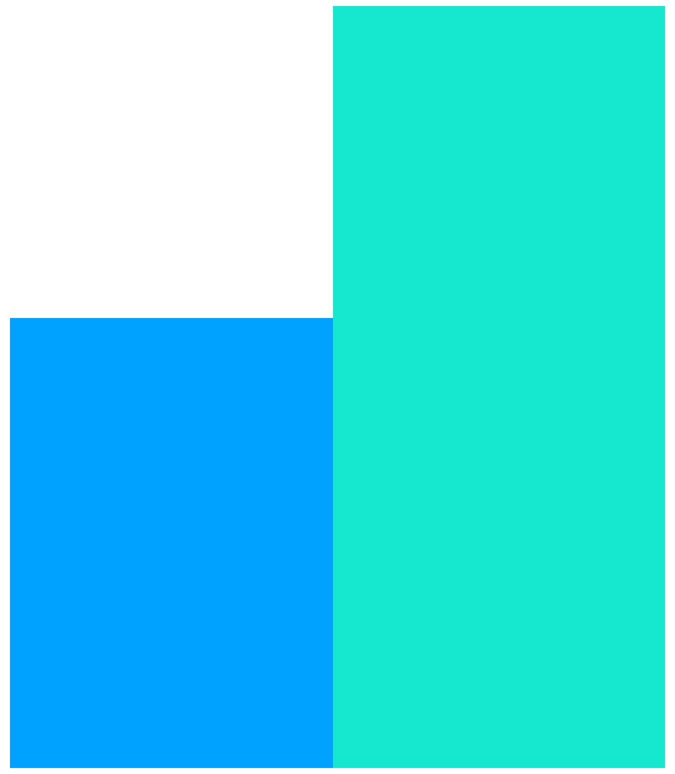
How do we redistribute?

- find a probability $\leq \frac{1}{n}$, put it on the rectangle
- find a probability $> \frac{1}{n}$, cut it an put it on the rectangle



How do we redistribute?

- find a probability $\leq \frac{1}{n}$, put it on the rectangle
- find a probability $> \frac{1}{n}$, cut it an put it on the rectangle
- repeat



$\frac{1}{n}$

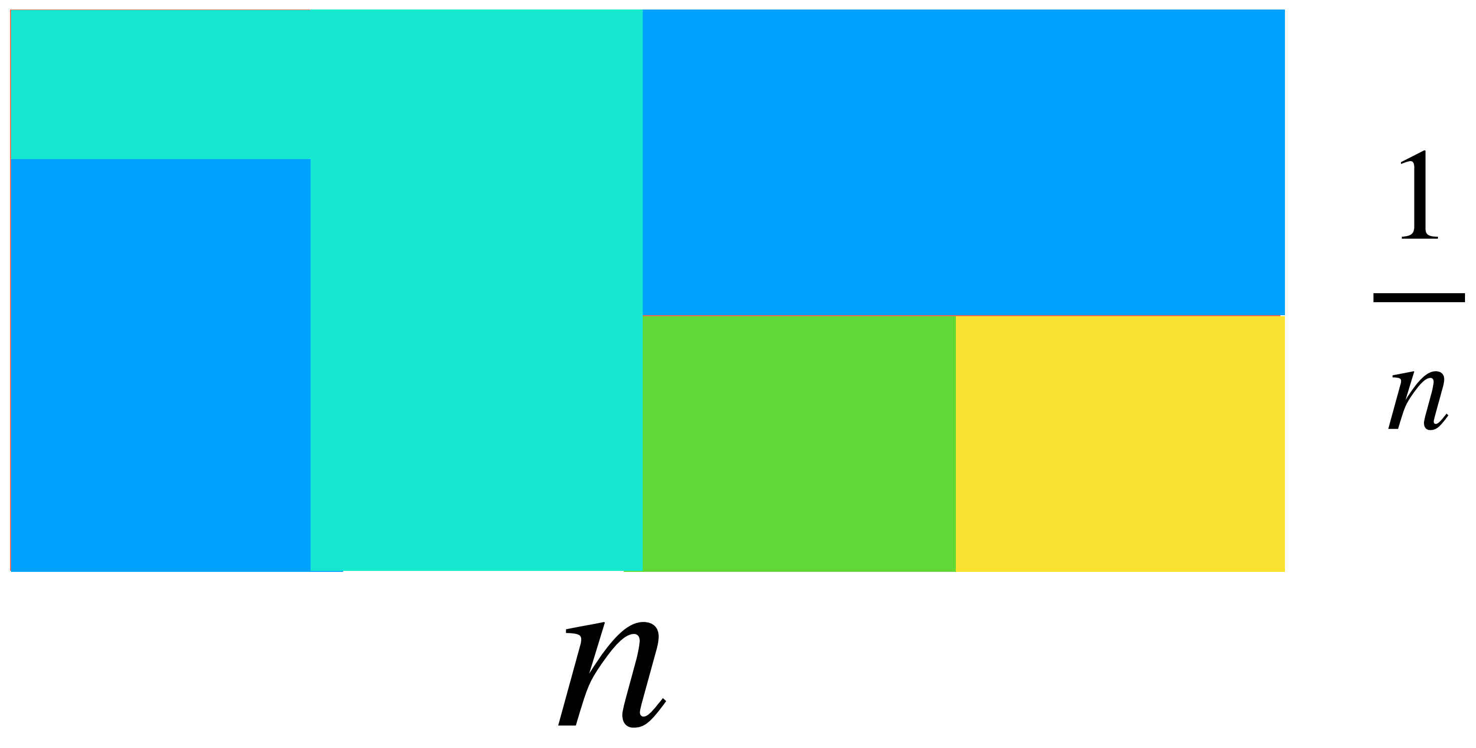
n

How do we redistribute?

- find a probability $\leq \frac{1}{n}$, put it on the rectangle
- find a probability $> \frac{1}{n}$, cut it an put it on the rectangle
- repeat

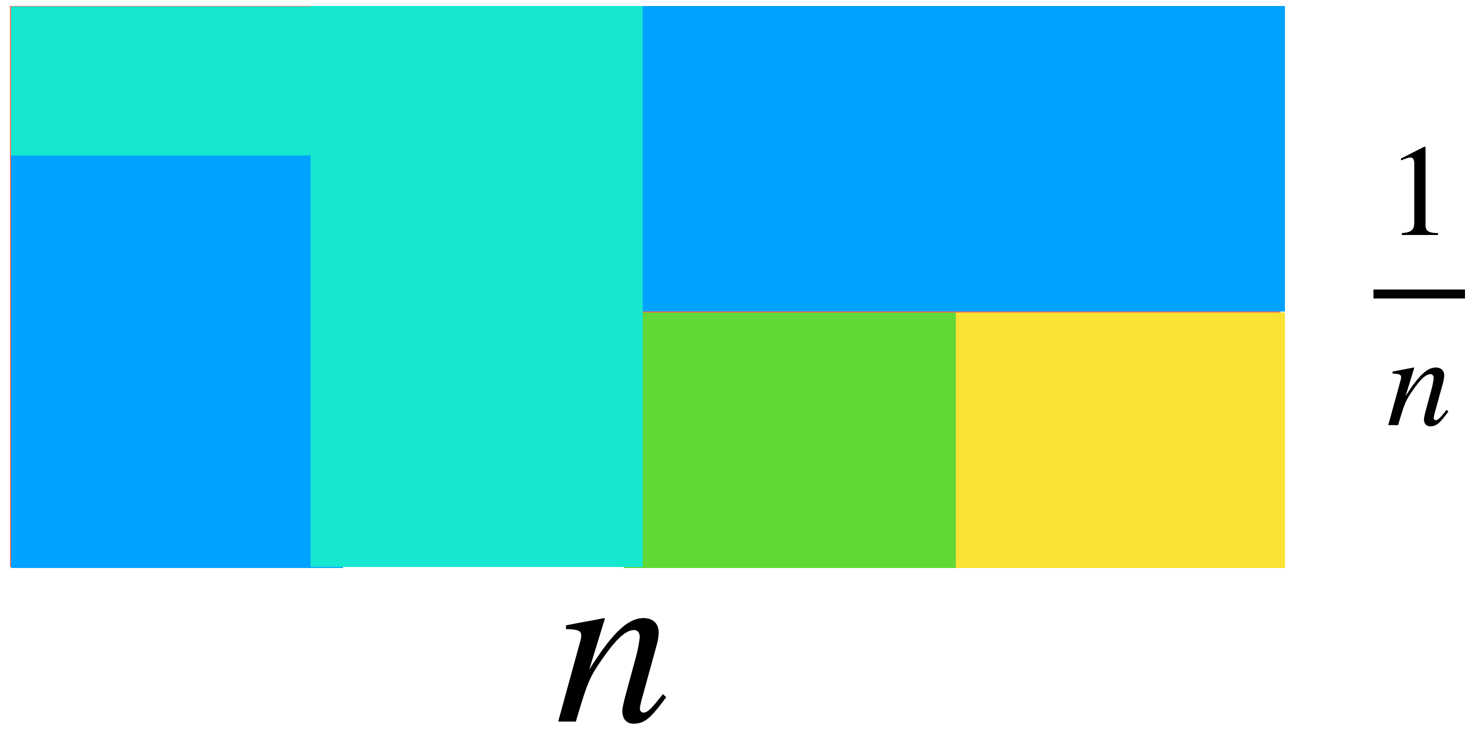
How do we redistribute?

- find a probability $\leq \frac{1}{n}$, put it on the rectangle
- find a probability $> \frac{1}{n}$, cut it an put it on the rectangle
- repeat



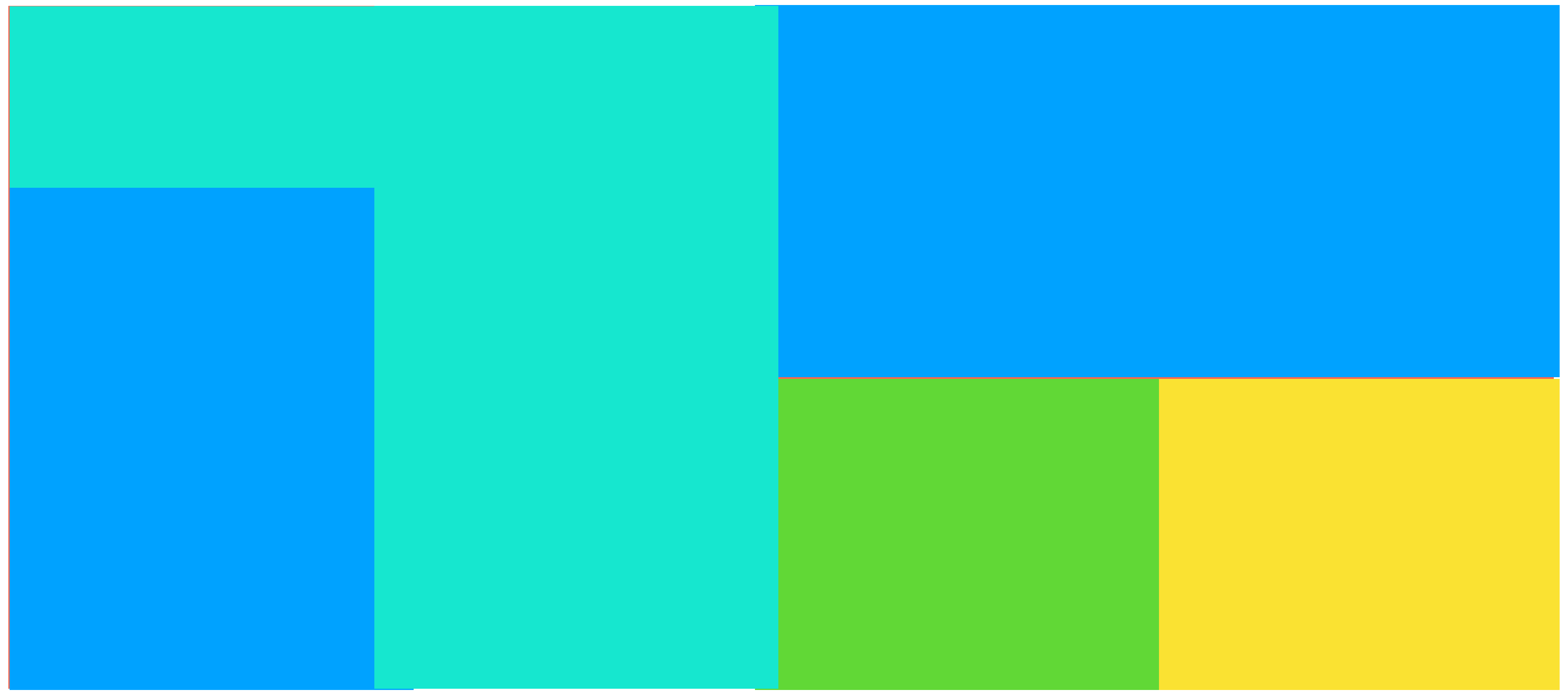
How do we redistribute?

- find a probability $\leq \frac{1}{n}$, put it on the rectangle
- find a probability $> \frac{1}{n}$, cut it an put it on the rectangle
- repeat
- can be done in O(n) time if we keep track of which entry is $\leq \frac{1}{n}$ and which is not



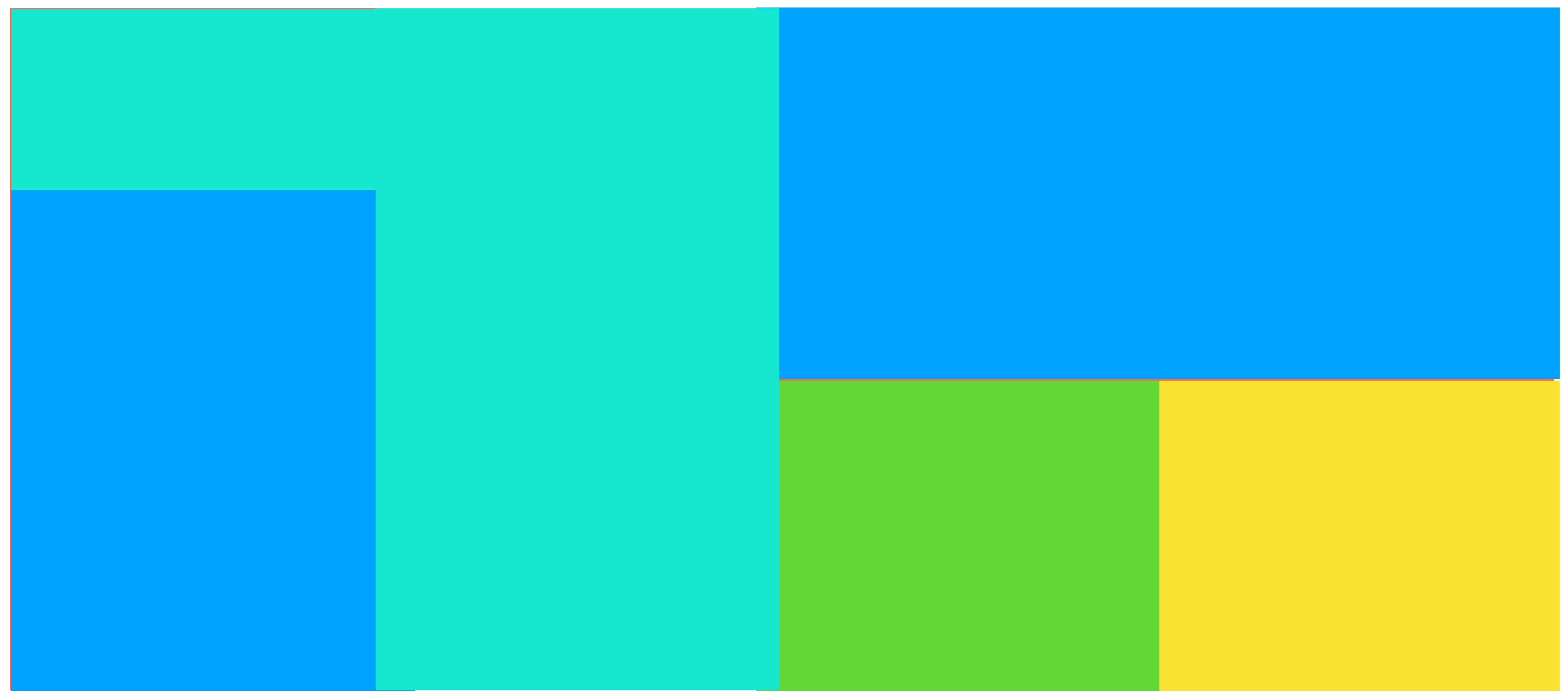
Alias method: pros and cons

- pro(s):
- con(s):



Alias method: pros and cons

- pro(s): fast
- con(s): stratification



Russian roulette debiasing

- goal: turn any consistent estimator into an unbiased estimator

$$\lim_{i \to \infty} A_i = A$$

Russian roulette debiasing

- goal: turn any consistent estimator into an unbiased estimator

$$\lim_{i \to \infty} A_i = A$$

idea: rewrite A as a **telescoping sum**

$$A = A_0 + \left(A_1 - A_0\right) + \left(A_2 - A_1\right) + \cdots = A_0 + \sum_{i=1}^{\infty} \left(A_i - A_{i-1}\right)$$

Russian roulette debiasing

- goal: turn any consistent estimator into an unbiased estimator

$$\lim_{i \to \infty} A_i = A$$

idea: rewrite A as a **telescoping sum**

$$A = A_0 + \left(A_1 - A_0\right) + \left(A_2 - A_1\right) + \cdots = A_0 + \sum_{i=1}^{\infty} \left(A_i - A_{i-1}\right)$$

sample N with probability p(N), estimate A as $\frac{A_0 + \sum_{i=1}^{N} \left(A_i - A_{i-1}\right)}{p(N)} = \frac{A_N}{p(N)}$

Russian roulette debiasing

- special case: a non-linear transformation of an integral

$$g\left(\int f\right)$$

Russian roulette debiasing

- special case: a non-linear transformation of an integral

$$g\left(\int f\right)$$

can be estimated using Taylor expansion at $F = \int f$

$$g(x) = g(F) + g'(F)(x - F) + g''(F)\frac{(x - F)^2}{2!} + \cdots$$

Russian roulette debiasing: applications

- unbiased estimation for non-linear transformation of integrals

$$\frac{1}{\int f} \qquad \exp\left(\int f\right) \qquad \left|\int f\right| \qquad \max\left(\int f, 0\right)$$

Russian roulette debiasing: applications

read this paper for more detail!

Unbiased and consistent rendering using biased estimators

ZACKARY MISSO, Dartmouth College, USA
BENEDIKT BITTERLI, Dartmouth College, USA and NVIDIA, USA
ILIYAN GEORGIEV, Autodesk, United Kingdom
WOJCIECH JAROSZ, Dartmouth College, USA

Tagged pointer

- in lajolla, we used `std::variant` for dynamic polymorphism

```
struct Foo {
  int type;
  union {
    TypeA a;
    TypeB b;
    …
  };
};
```

this is somewhat memory consuming —
can we save some memory?

https://pharr.org/matt/blog/2018/08/03/moana-reader-mail

Tagged pointer

- hack: on x86 machines, only 48-bits out of 64-bits of a pointer are used
  - use the 16-bits to store the type information!

```
struct TaggedPointer {
  uint32_t tag() const { return ((bits & tag_mask) >> tag_shift); }
  void* ptr() const { return reinterpret_cast<void*>(bits & ptr_mask); }

  uint64_t bits;
  static constexpr int tag_shift = 48;
  static constexpr int tag_bits = 64 - tagShift;
  static constexpr uint64_t tag_mask = ((1ull << tag_bits) - 1) << tag_shift;
  static constexpr uint64_t ptr_mask = ~tag_mask;
};
```

https://github.com/mmp/pbrt-v4/blob/master/src/pbrt/util/taggedptr.h

Numerically stable cross product

```
// a * b - c * d
difference_of_products(a, b, c, d) {
  cd = c * d
  err = fma(-c, d, cd) // -c*d + cd
  dop = fma(a, b, -cd) // a*b - cd
  return dop + err
}
```

https://pharr.org/matt/blog/2019/11/03/difference-of-floats

Numerically stable quadratic solve

$$ax^2 + bx + c = 0$$

$$x_1 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \qquad x_2 = \frac{2c}{-b - \sqrt{b^2 - 4ac}} \qquad b > 0$$

$$x_1 = \frac{2c}{-b + \sqrt{b^2 - 4ac}} \qquad x_2 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad b \leq 0$$

https://people.csail.mit.edu/bkph/articles/Quadratics.pdf

Cosine-weighted hemisphere sampling in 4 lines without building a frame

```
vec3 lambertNoTangent(in vec3 normal, in vec2 uv) {
    float theta = 6.283185 * uv.x;
    uv.y = 2.0 * uv.y - 1.0;
    vec3 spherePoint = vec3(sqrt(1.0 - uv.y * uv.y) * vec2(cos(theta), sin(theta)), uv.y);
    return normalize(normal + spherePoint);
}
```

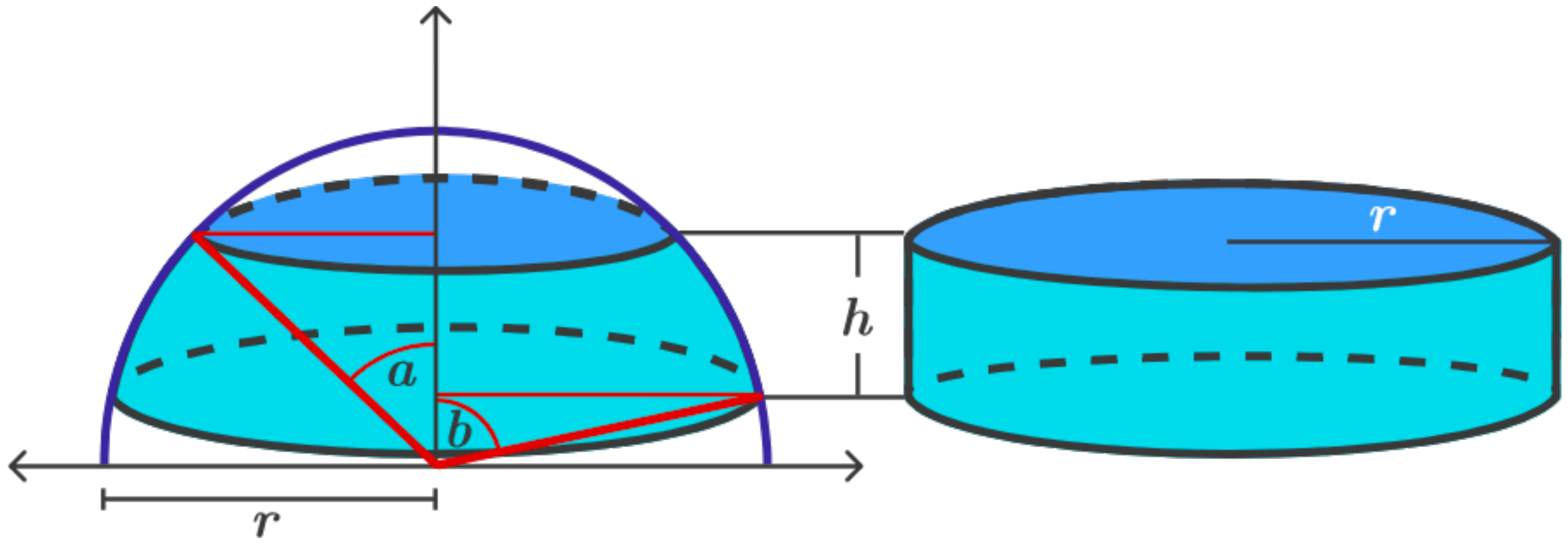
credit: Edd Biddulph

https://web.archive.org/web/20170610002747/http://amietia.com/lambertnotangent.html

Cosine-weighted hemisphere sampling in 4 lines without building a frame

```
vec3 lambertNoTangent(in vec3 normal, in vec2 uv) {
    float theta = 6.283185 * uv.x;
    uv.y = 2.0 * uv.y - 1.0;
    vec3 spherePoint = vec3(sqrt(1.0 - uv.y * uv.y) * vec2(cos(theta), sin(theta)), uv.y);
    return normalize(normal + spherePoint);
}
```

Archimedes' Hat-Box Theorem:
the area of a spherical section
= the area of a cylinder with the same radius



https://brilliant.org/wiki/surface-area-sphere/#archimedes-hat-box-theorem

credit: Edd Biddulph

https://web.archive.org/web/20170610002747/http://amietia.com/lambertnotangent.html

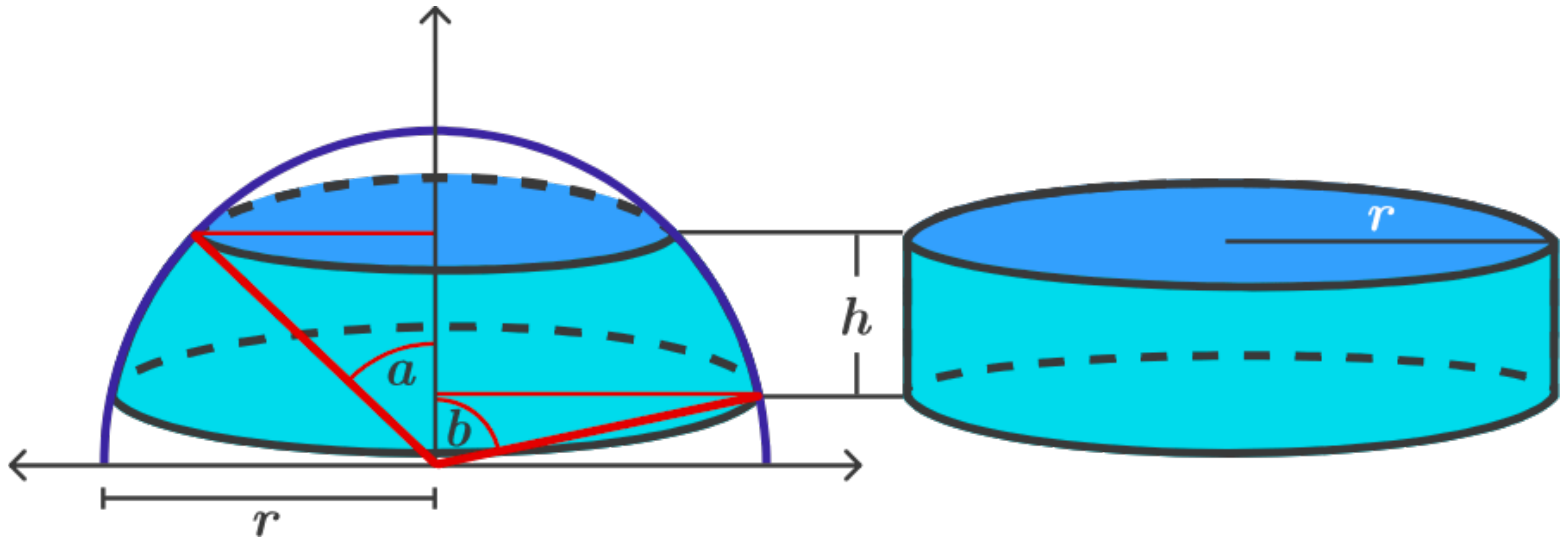
Cosine-weighted hemisphere sampling in 4 lines without building a frame

```
vec3 lambertNoTangent(in vec3 normal, in vec2 uv) {
    float theta = 6.283185 * uv.x;
    uv.y = 2.0 * uv.y - 1.0;
    vec3 spherePoint = vec3(sqrt(1.0 - uv.y * uv.y) * vec2(cos(theta), sin(theta)), uv.y);
    return normalize(normal + spherePoint);
}
```

Archimedes' Hat-Box Theorem:
the area of a spherical section
= the area of a cylinder with the same radius

↓

uniformly sampling a sphere
= uniformly sampling on concentric rings of a disk



https://brilliant.org/wiki/surface-area-sphere/#archimedes-hat-box-theorem

credit: Edd Biddulph

https://web.archive.org/web/20170610002747/http://amietia.com/lambertnotangent.html

Cosine-weighted hemisphere sampling in 4 lines without building a frame

```
vec3 lambertNoTangent(in vec3 normal, in vec2 uv) {
    float theta = 6.283185 * uv.x;
    uv.y = 2.0 * uv.y - 1.0;
    vec3 spherePoint = vec3(sqrt(1.0 - uv.y * uv.y) * vec2(cos(theta), sin(theta)), uv.y);
    return normalize(normal + spherePoint);
}
```

algorithm:

- sampling on a unit sphere uniformly
- project onto a unit disk
- scale their distance to the origin to make it a uniform sampling of a disk
- project back onto the hemisphere (Malley's method)

credit: Edd Biddulph

https://web.archive.org/web/20170610002747/http://amietia.com/lambertnotangent.html

What next?