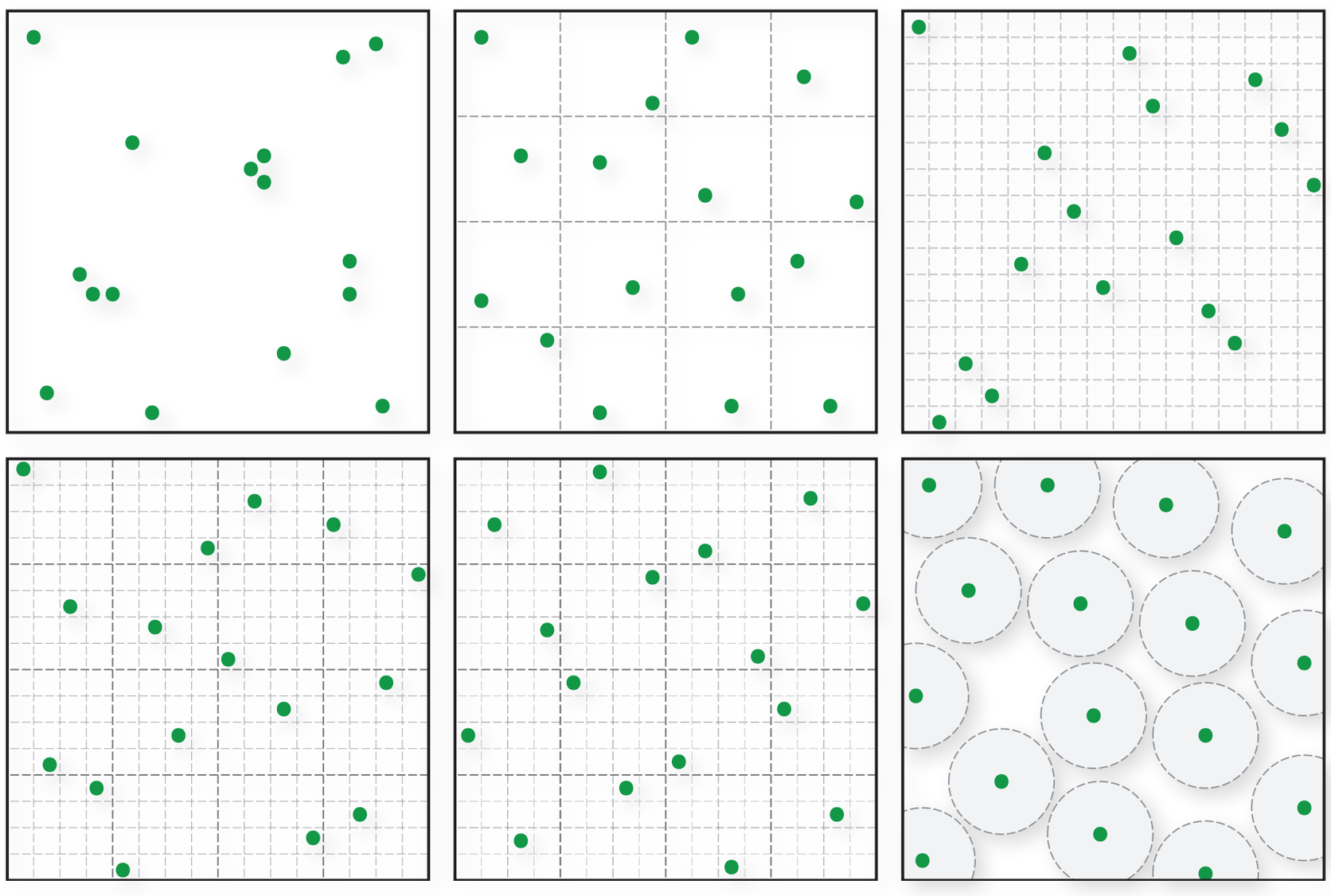
CS 87/187, Spring 2016

# Rendering Algorithms

## Monte Carlo II: Improved Sampling & Variance Reduction




Prof. Wojciech Jarosz

wojciech.k.jarosz@dartmouth.edu

Dartmouth

# Last Lecture

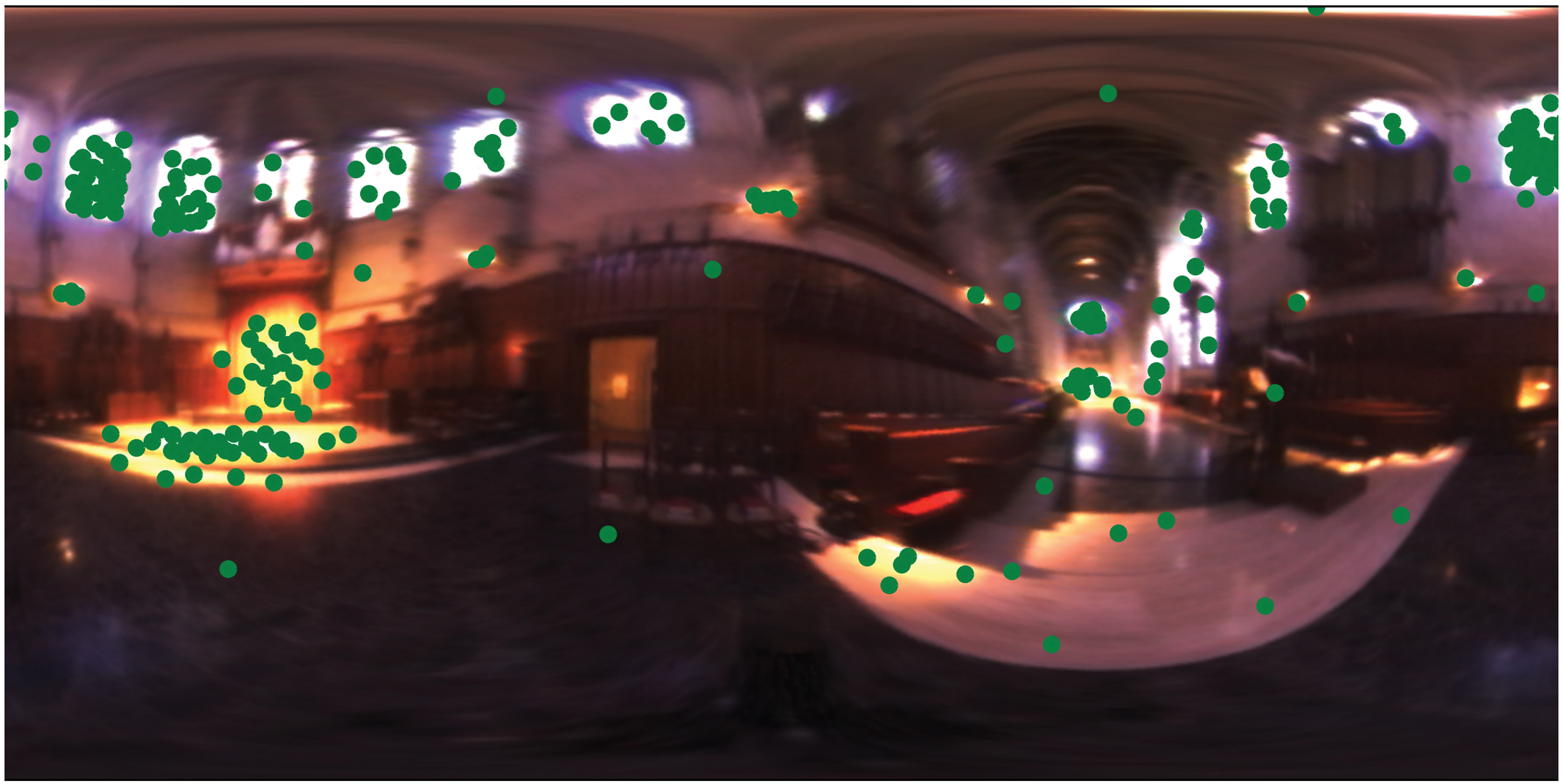
- Realistic lighting & realistic cameras

# Environment Lighting

# Environment Lighting

$$L_r(\mathbf{x}, \vec{\omega}_r) = \int_\Omega f_r(\vec{\omega}_i, \vec{\omega}_r) L_{\text{env}}(\vec{\omega}_i) V(\mathbf{x}, \vec{\omega}_i) \cos\theta_i \, d\vec{\omega}_i$$

# Importance Sampling $L_{\text{env}}$



$$p(\vec{\omega}_i) \propto L_{\text{env}}(\vec{\omega}_i)$$

# Marginal/Conditional CDF

- Assume the lat/long parameterization
- Draw samples from joint $p(\theta, \phi) \propto L_{\text{env}}(\theta, \phi) \sin\theta$
  - Step 1: create scalar version $L'(\theta, \phi)$ of $L_{\text{env}}(\theta, \phi) \sin\theta$
  - Step 2: compute marginal PDF

$$p(\theta) = \int_0^{2\pi} L'(\theta, \phi)\, d\phi$$

  - Step 3: compute conditional PDF

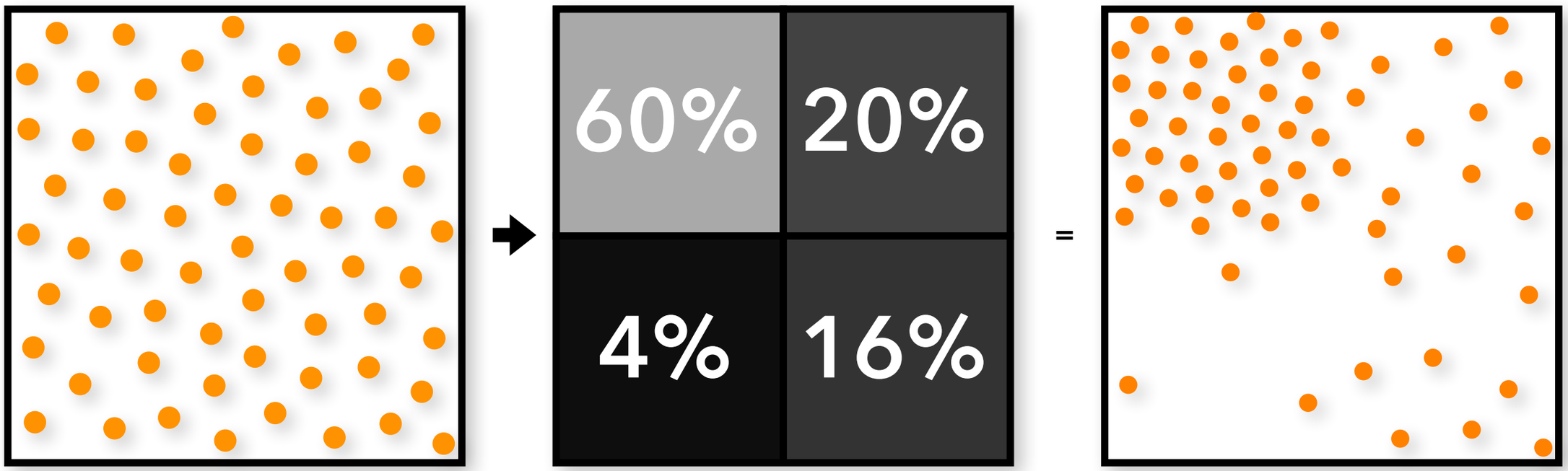
$$p(\phi|\theta) = \frac{p(\theta, \phi)}{p(\theta)}$$

  - Step 4: draw samples $\theta_i \sim p(\theta)$ and $\phi_i = p(\phi|\theta)$

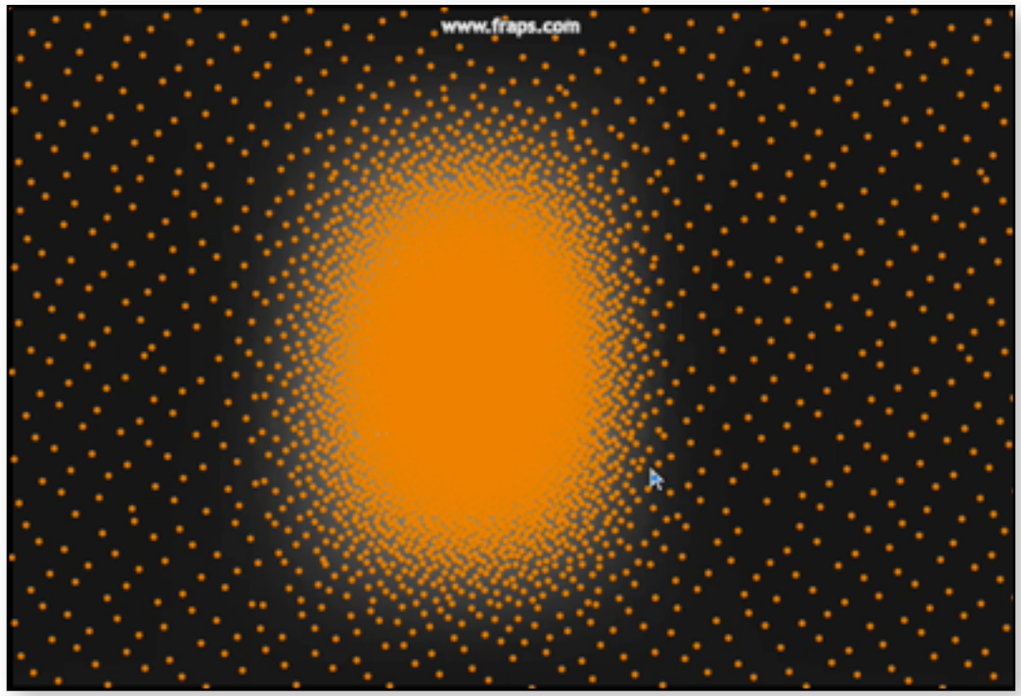
# Hierarchical Sample Warping

Clarberg, Jarosz, Akenine-Möller, Jensen. "Wavelet Importance Sampling," 2005.

- Given:
  - input point set, and
  - hierarchical representation of density function (mip-map)
- Recursively warp point set to match the importance function

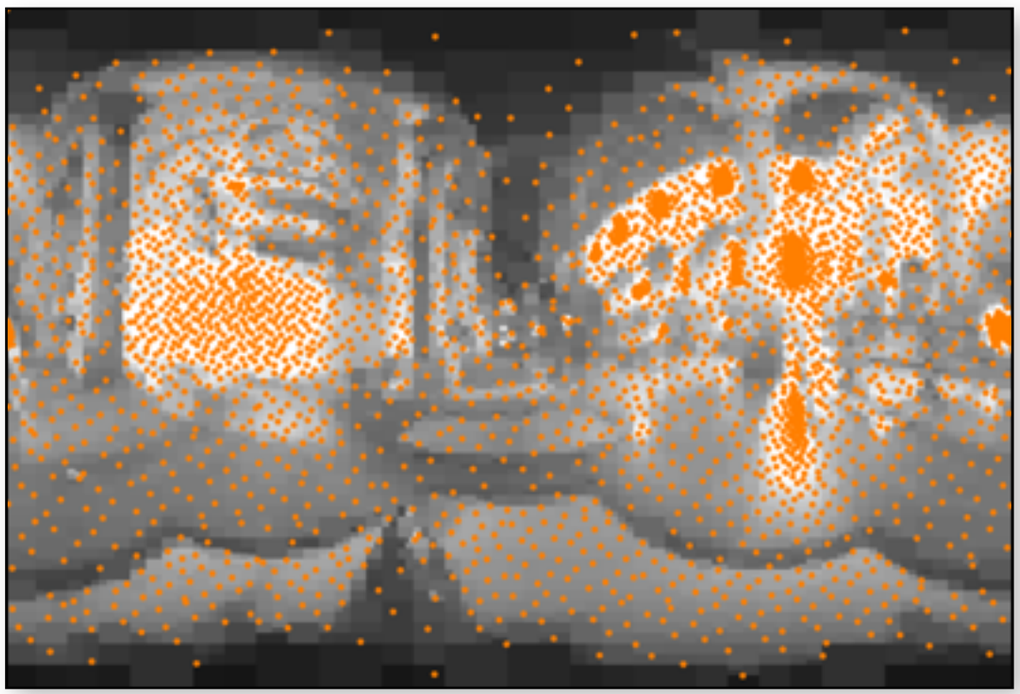


# Product Sampling



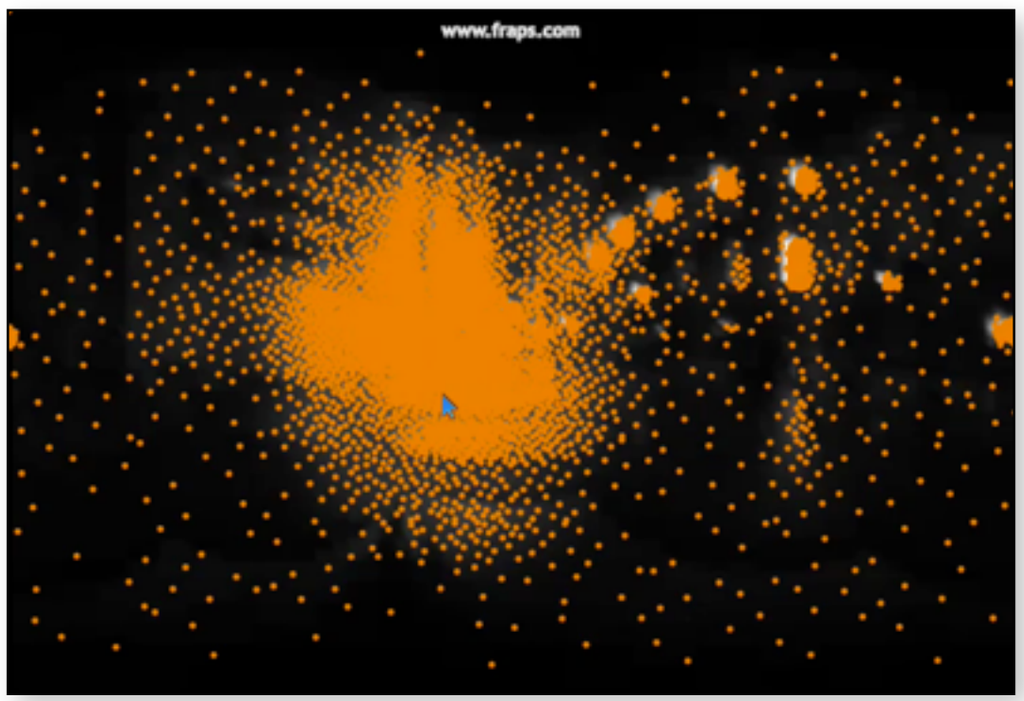

BRDF

x



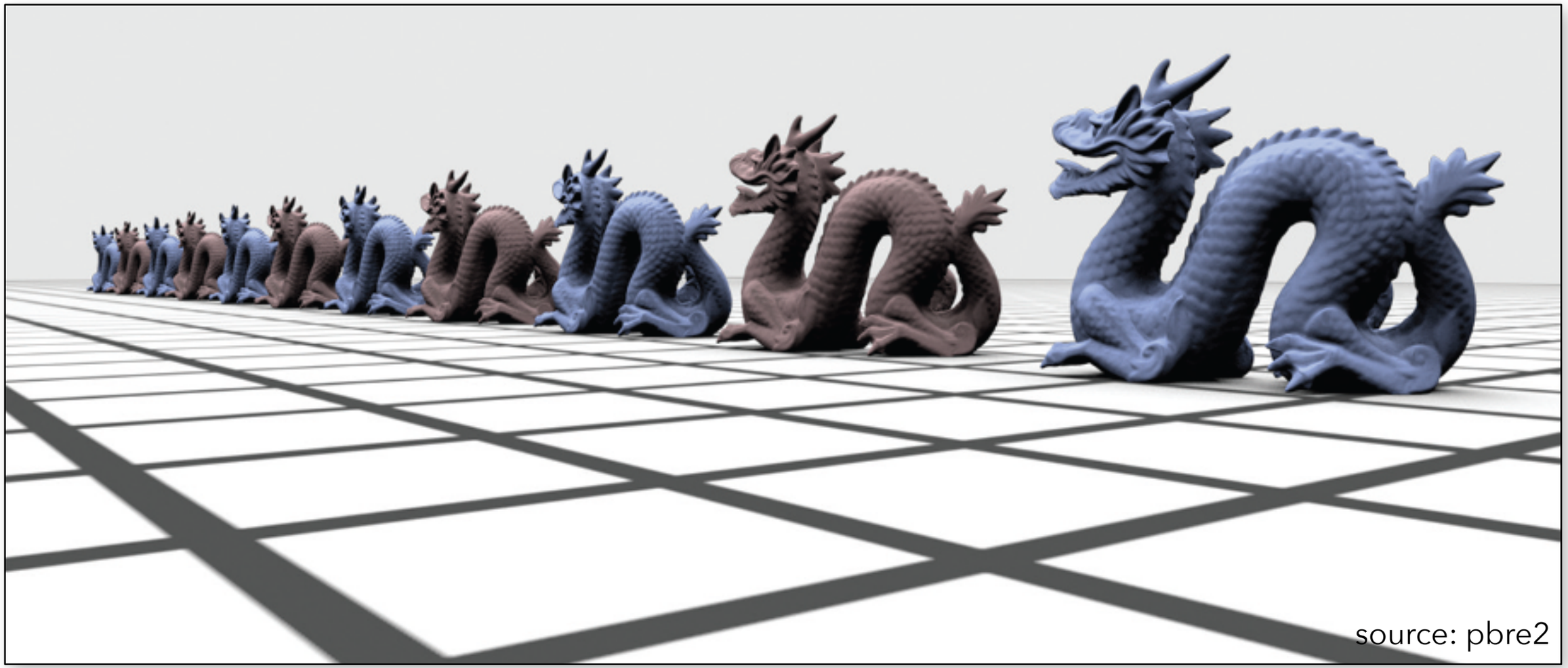
Lighting

=

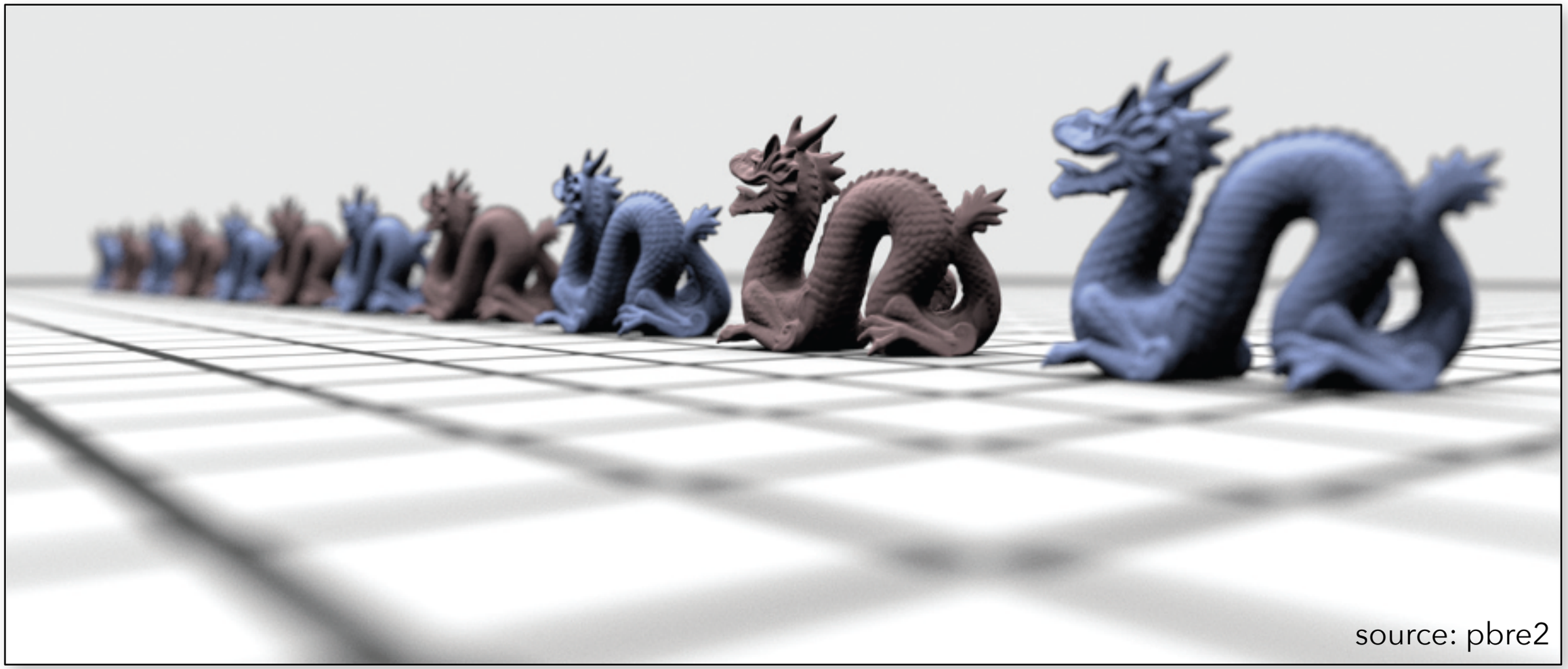



BRDF x Lighting

# Pinhole Camera



# Thin-Lens Camera (larger aperture)



# Monte Carlo Integration Recap

- Goal: evaluate integral $\int_a^b f(x)dx$

- Random variable $X_i \sim p(x)$

- Monte Carlo Estimator $F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}$

- Expectation $E[F_N] = \int_a^b f(x)dx$

# Today's Menu

- Variance reduction techniques
  - Control variates
- Better random variables using
  - Stratified Sampling
  - Quasi Monte Carlo

# Monte Carlo

- The standard MC estimator:

$$F = \int_{\mu(x)} f(x)\ \mathrm{d}\mu(x)$$

$$\left\langle F^N \right\rangle = \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(X_i)}{\mathrm{pdf}(X_i)}$$

# Monte Carlo

- The standard MC estimator:

$$F = \int_{\mu(x)} f(x)\, \mathrm{d}\mu(x)$$

$$V\left[\left\langle F^N \right\rangle\right] = V\left[\frac{1}{N}\sum_{i=0}^{N-1}\frac{f(X_i)}{\mathrm{pdf}(X_i)}\right]$$

# Monte Carlo Error

- Variance:

$$V\left[\left\langle F^N \right\rangle\right] = V\left[\frac{1}{N}\sum_{i=0}^{N-1}\frac{f(X_i)}{\mathrm{pdf}(X_i)}\right]$$

← assume uncorrelated samples

$$= \frac{1}{N^2}\sum_{i=0}^{N-1} V\left[\frac{f(X_i)}{\mathrm{pdf}(X_i)}\right]$$

$$= \frac{1}{N^2}\sum_{i=0}^{N-1} V\left[Y_i\right]$$

$$= \frac{1}{N} V\left[Y\right]$$

# Monte Carlo Error

- Variance:

$$V\left[\left\langle F^N \right\rangle\right] = V\left[\frac{1}{N}\sum_{i=0}^{N-1}\frac{f(X_i)}{\mathrm{pdf}(X_i)}\right]$$

$$= \frac{1}{N^2}\sum_{i=0}^{N-1} V\left[\frac{f(X_i)}{\mathrm{pdf}(X_i)}\right]$$

$$= \frac{1}{N^2}\sum_{i=0}^{N-1} V\left[Y_i\right]$$

$$= \frac{1}{N} V\left[Y\right]$$

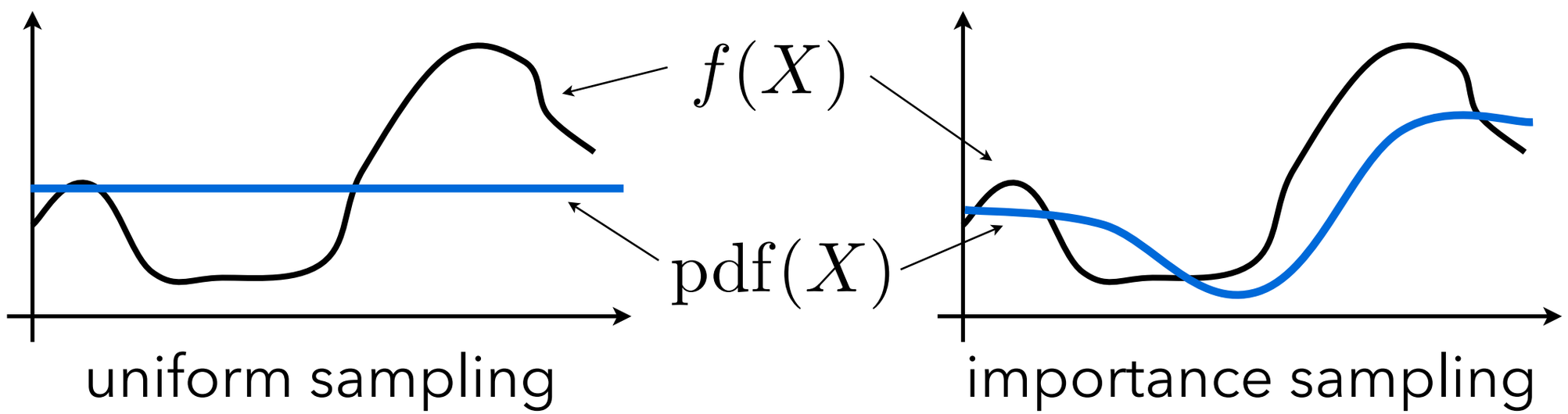
- Std. deviation:

$$\sigma\left[\left\langle F^N \right\rangle\right] = \boxed{\frac{1}{\sqrt{N}}\sigma\left[Y\right]}$$

# Strategies for Reducing Variance

$$\sigma\left[\left\langle F^N \right\rangle\right] = \frac{1}{\sqrt{N}}\sigma\left[Y\right]$$

- Reduce the variance of $Y$
  - Importance sampling



# Strategies for Reducing Variance

$$\sigma\left[\left\langle F^N \right\rangle\right] = \frac{1}{\sqrt{N}}\sigma\left[Y\right]$$

- Reduce the variance of $Y$
  - Importance sampling
  - Control Variates

# Control Variates

- Say you have a function $g$, which you can efficiently (e.g. analytically) integrate to $G$

$$F = \int_a^b f(x)\,\mathrm{d}x,$$

- Control variates estimator:

$$\left\langle F_c^N \right\rangle = \left( \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(X_i) - g(X_i)}{\mathrm{pdf}(X_i)} \right) + G$$

# Control Variates

- Control variates estimator:

$$\left\langle F_c^N \right\rangle = \left( \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(X_i) - g(X_i)}{\mathrm{pdf}(X_i)} \right) + G$$

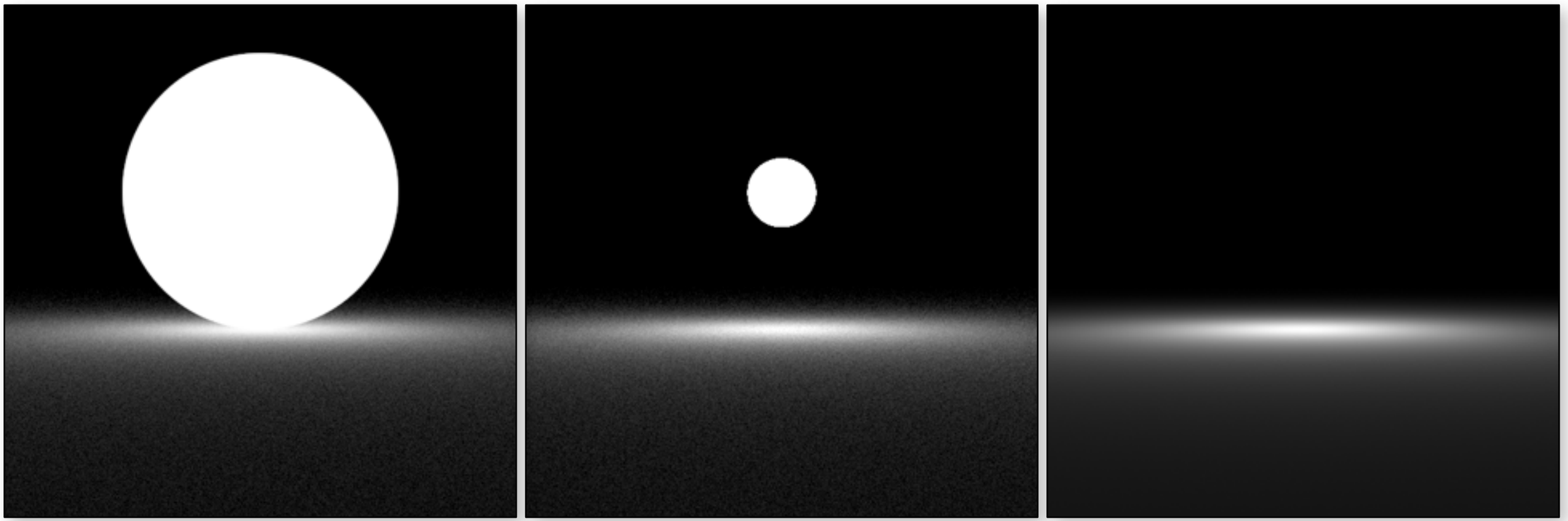
- Recall that:

$$\sigma \left[ \left\langle F^N \right\rangle \right] = \frac{1}{\sqrt{N}} \sigma \left[ Y \right]$$

  - and in this case:

$$Y = \frac{f(X) - g(X)}{\mathrm{pdf}(X)}$$

# Sphere Light



A sphere light

A smaller sphere light

A point light

shading of diffuse surface independent of sphere light radius

idea: use analytic point light as control variate for sphere lights

# Control Variates (Sphere light)

- Exploiting analytic sphere lights:

$$L_r(\omega_o) = \int_\Omega f_r(\omega_i, \omega_o) L_i(\omega_i, \omega_o) V(\omega_i) \cos\theta_i \, \mathrm{d}\omega_i$$

$$\langle L_r(\omega_o) \rangle = \left( \frac{1}{N} \sum_{j=0}^{N-1} \frac{f(\omega_{i,j}) - g(\omega_{i,j})}{\mathrm{pdf}(\omega_{i,j})} \right) + G$$

  - where:

$$f(\omega_i) = f_s(\omega_i, \omega_o) L_i(\omega_i, \omega_o) V(\omega_i) \cos\theta_i$$

$$g(\omega_i) = L_i(\omega_i, \omega_o) \cos\theta_i$$

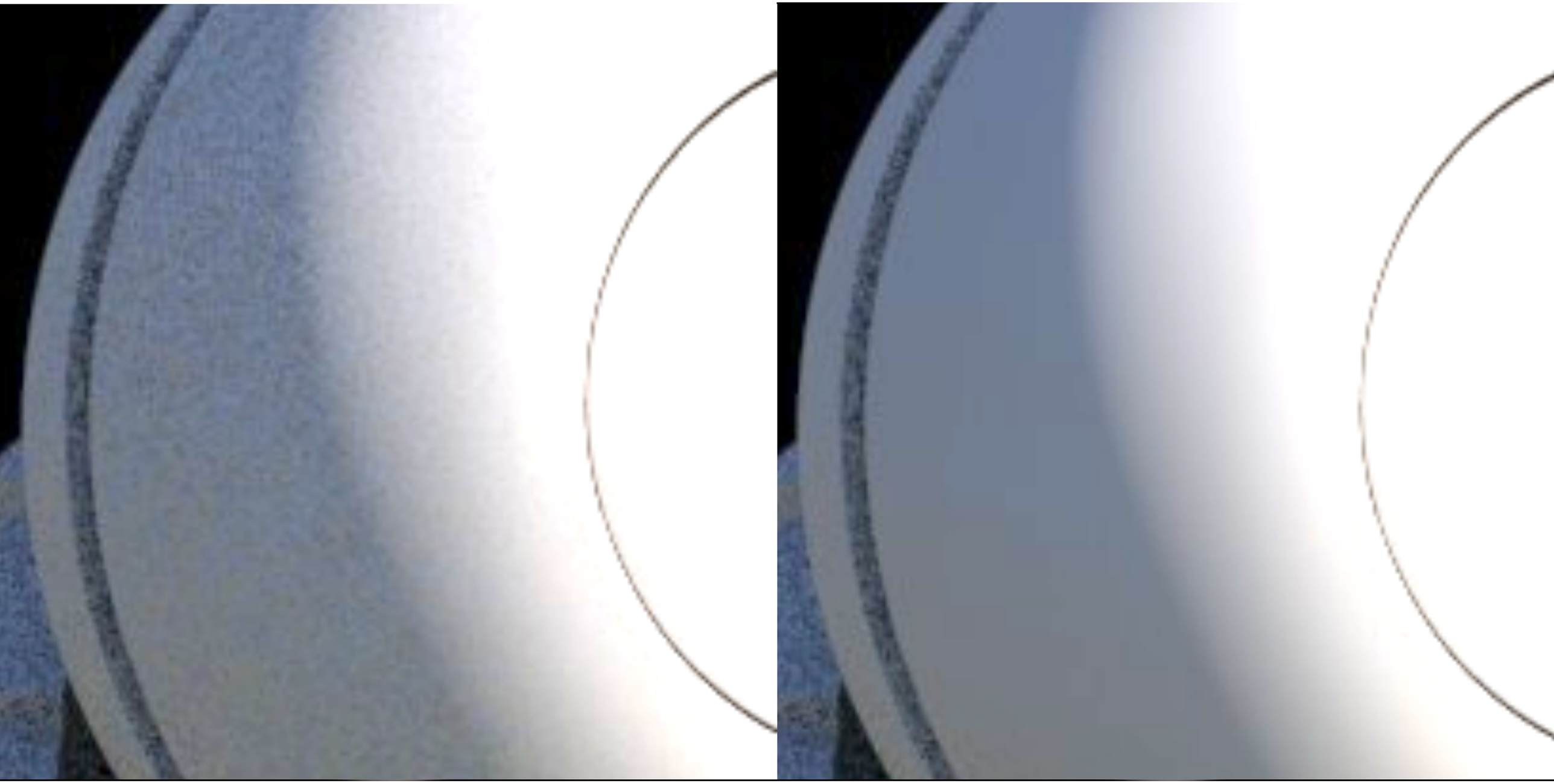
$$G = \frac{\Phi_e \cos\theta}{4\pi r^2}$$

For sphere lights clipping the horizon see: Hery and Villemin. “Physically Based Lighting at Pixar.” SIGGRAPH 2013.

# Control Variates (Sphere light)

Control variates off

Control variates on

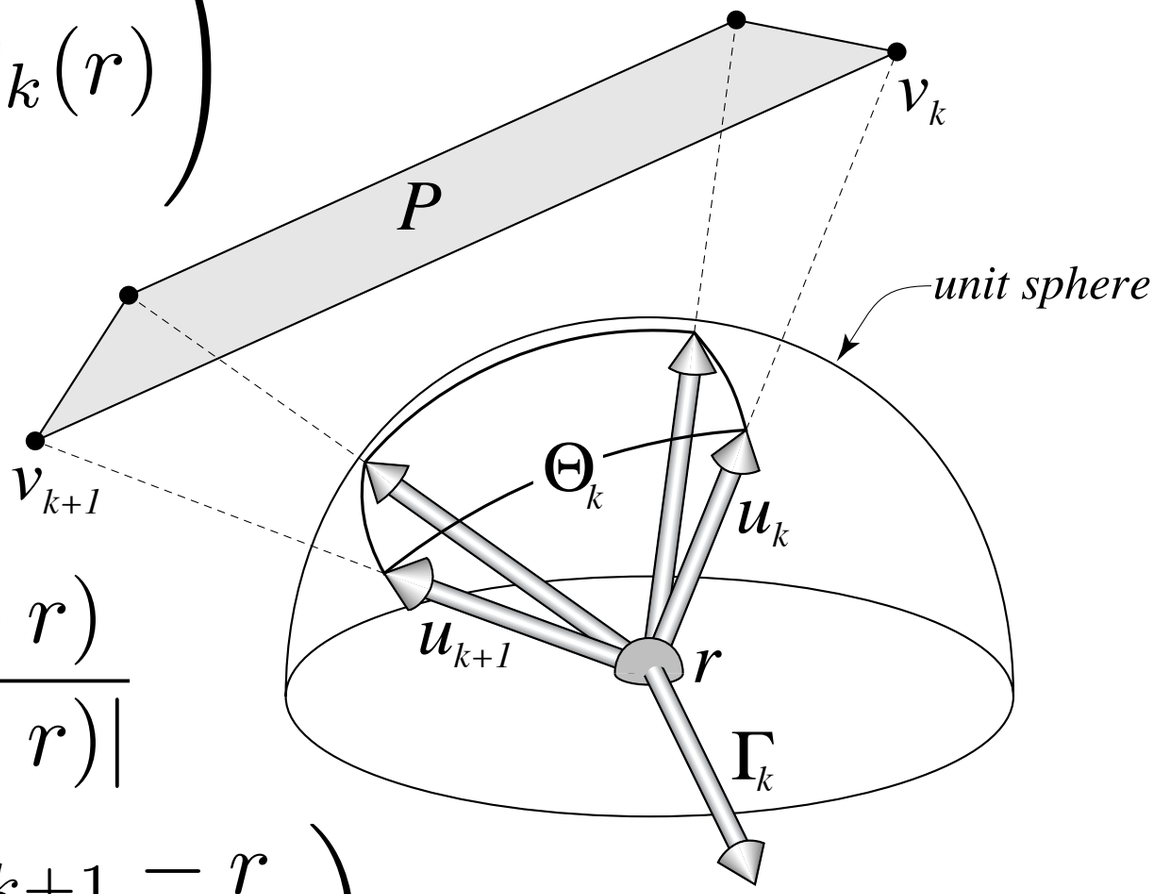


from Hery and Villemin. “Physically Based Lighting at Pixar.” SIGGRAPH 2013.

# Analytic polygon light

- Can obtain the irradiance due to a diffuse, polygonal source in closed form using Stoke's theorem

$$E(r) = -\mathbf{n} \cdot \left( \frac{M}{2\pi} \sum_{k=1}^{n} \Theta_k(r)\, \Gamma_k(r) \right)$$




$$\Gamma_k(r) = \frac{(v_k - r) \times (v_{k+1} - r)}{\left|(v_k - r) \times (v_{k+1} - r)\right|}$$

$$\Theta_k(r) = \cos^{-1}\left( \frac{v_k - r}{|v_k - r|} \cdot \frac{v_{k+1} - r}{|v_{k+1} - r|} \right)$$

J. Arvo. 1994. "The Irradiance Jacobian for Partially Occluded Polyhedral Sources"

# Control Variates (Polygon light)

- Exploit analytic polygonal irradiance

$$L_r(\omega_o) = \int_\Omega f_r(\omega_i, \omega_o) L_i(\omega_i, \omega_o) V(\omega_i) \cos\theta_i \, \mathrm{d}\omega_i$$

$$\langle L_r(\omega_o) \rangle = \left( \frac{1}{N} \sum_{j=0}^{N-1} \frac{f(\omega_{i,j}) - g(\omega_{i,j})}{\mathrm{pdf}(\omega_{i,j})} \right) + G$$

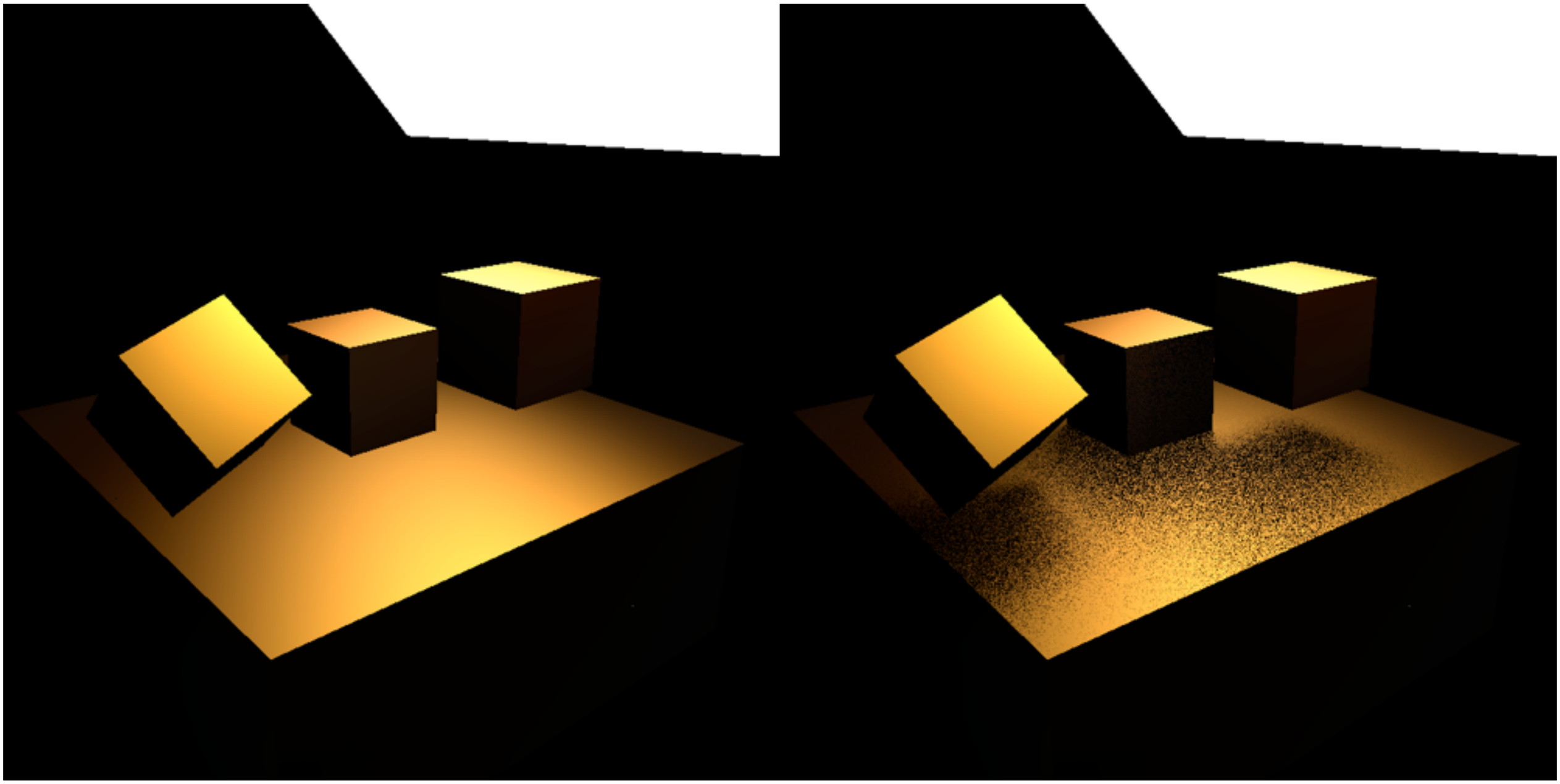
  - where:

$$f(\omega_i) = f_s(\omega_i, \omega_o) L_i(\omega_i, \omega_o) V(\omega_i) \cos\theta_i$$

$$g(\omega_i) = L_i(\omega_i, \omega_o) \cos\theta_i$$

$$G = -\mathbf{n} \cdot \left( \frac{M}{2\pi} \sum_{k=1}^{n} \Theta_k \, \Gamma_k \right)$$

# Analytic area lights as control variates



analytic result

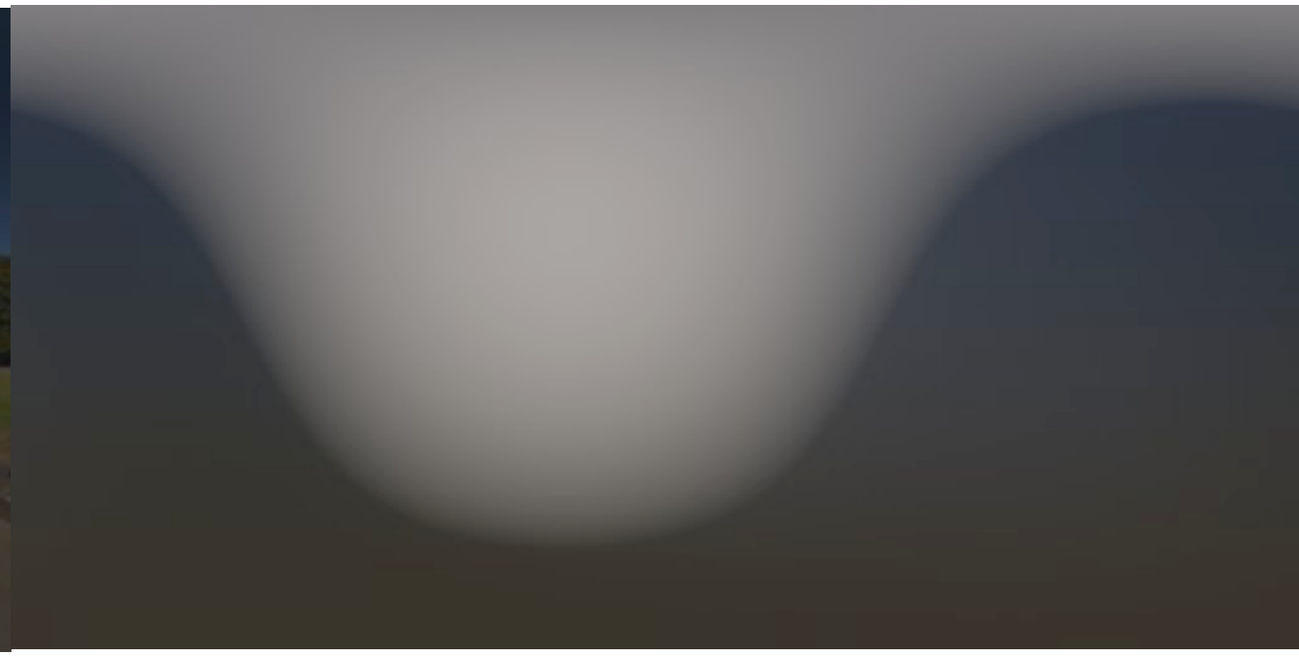
control variate +
MC shadows

# Control Variates (Env map)

$$L_r(\omega_o) = \int_\Omega f_r(\omega_i, \omega_o) L_{\text{env}}(\omega_i) V(\omega_i)(\omega_i \cdot \vec{n}) \, \mathrm{d}\omega_i$$

- Precompute diffuse convolution for an env map

$$L_{\text{env}}(\omega_i) \qquad G(\vec{n}) = \int_\Omega L_{\text{env}}(\omega_i)(\omega_i \cdot \vec{n}) \, \mathrm{d}\omega_i$$



# Control Variates (Env map)

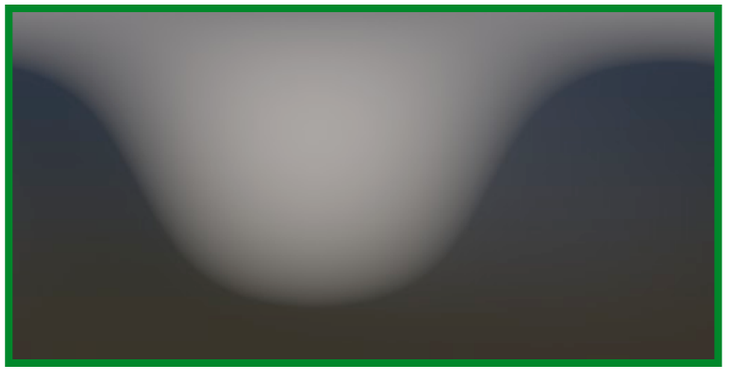
$$L_r(\omega_o) = \int_\Omega f_r(\omega_i, \omega_o) L_{\text{env}}(\omega_i) V(\omega_i)(\omega_i \cdot \vec{n}) \, \mathrm{d}\omega_i$$

$$\langle L_r(\omega_o) \rangle = \left( \frac{1}{N} \sum_{j=0}^{N-1} \frac{f(\omega_{i,j}) - \frac{\rho}{\pi} g(\omega_{i,j})}{\text{pdf}(\omega_{i,j})} \right) + \frac{\rho}{\pi} G(\vec{n})$$

$$f(\omega_i) = f_r(\omega_i, \omega_o) L_{\text{env}}(\omega_i) V(\omega_i)(\omega_i \cdot \vec{n})$$

$$g(\omega_i) = L_{\text{env}}(\omega_i)(\omega_i \cdot \vec{n})$$

$$G(\vec{n}) = \int_\Omega L_{\text{env}}(\omega_i)(\omega_i \cdot \vec{n}) \, \mathrm{d}\omega_i$$



# Variance reduction summary

- If I have a function $g$ which is "similar" to $f$, should I use it for importance sampling or control variates?
  - If $f$-$g$ is nearly constant, use $g$ as a control variate
  - If $f/g$ is nearly constant, use $g$ as a PDF

- Importance sampling (IS)
  - must be able to sample from $g$ (typically involves integrating to CDF and *inverting*)

- Control variates (CV)
  - only requires being able to evaluate $g$ and efficiently compute $G$ (does not actually need to be analytic!)

# Variance reduction summary

- If I have a function $g$ which is "similar" to $f$, should I use it for importance sampling or control variates?
- Can I employ both IS and CV?
  - Yes, but recall…

$$\left\langle F_c^N \right\rangle = \left( \frac{1}{N} \sum_{i=0}^{N-1} \frac{f(X_i) - g(X_i)}{\mathrm{pdf}(X_i)} \right) + G$$

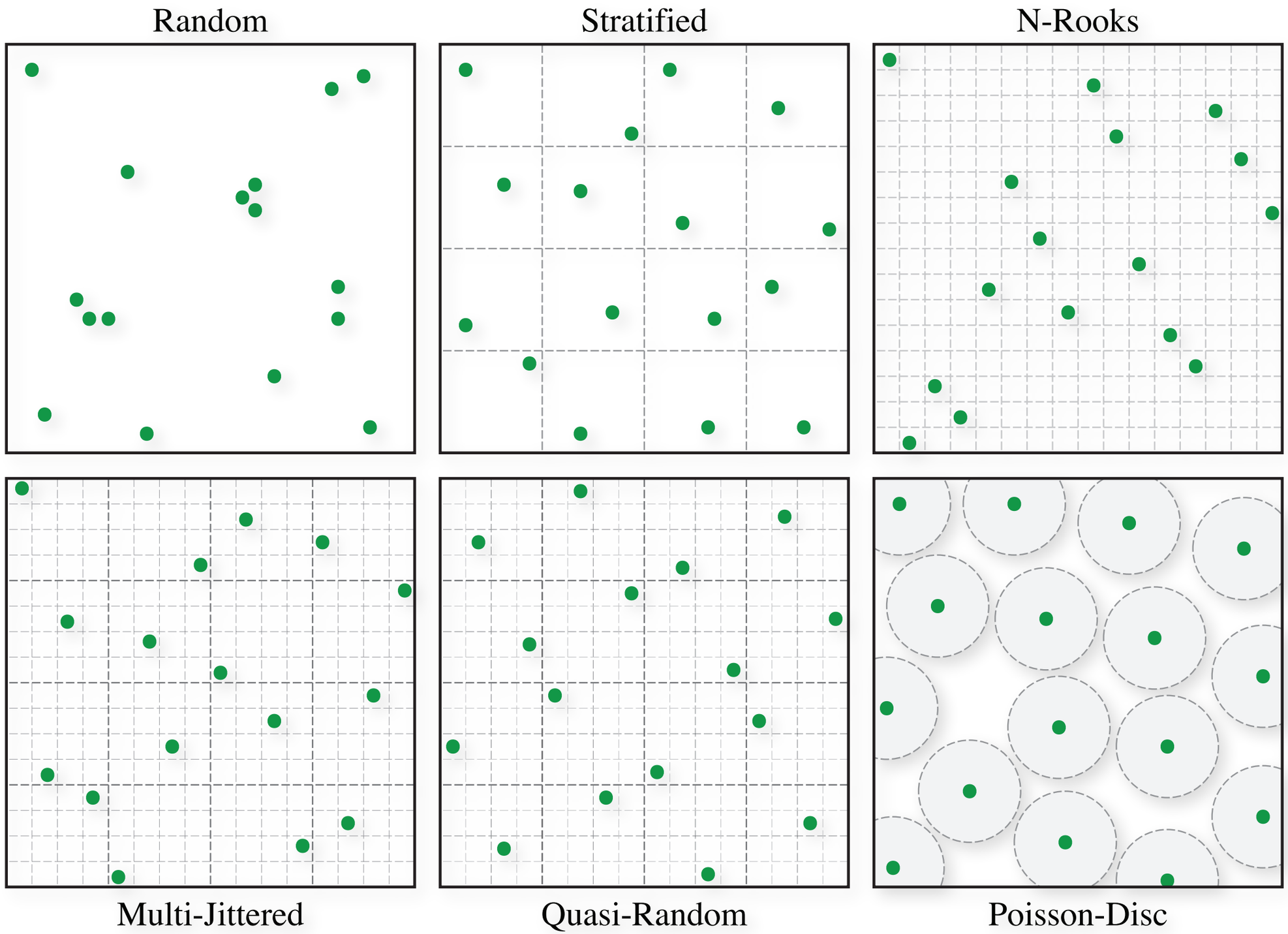
$$Y = \frac{f(X) - g(X)}{\mathrm{pdf}(X)} \qquad \sigma\left[\left\langle F^N \right\rangle\right] = \frac{1}{\sqrt{N}} \sigma\left[Y\right]$$

# Strategies for Reducing Variance

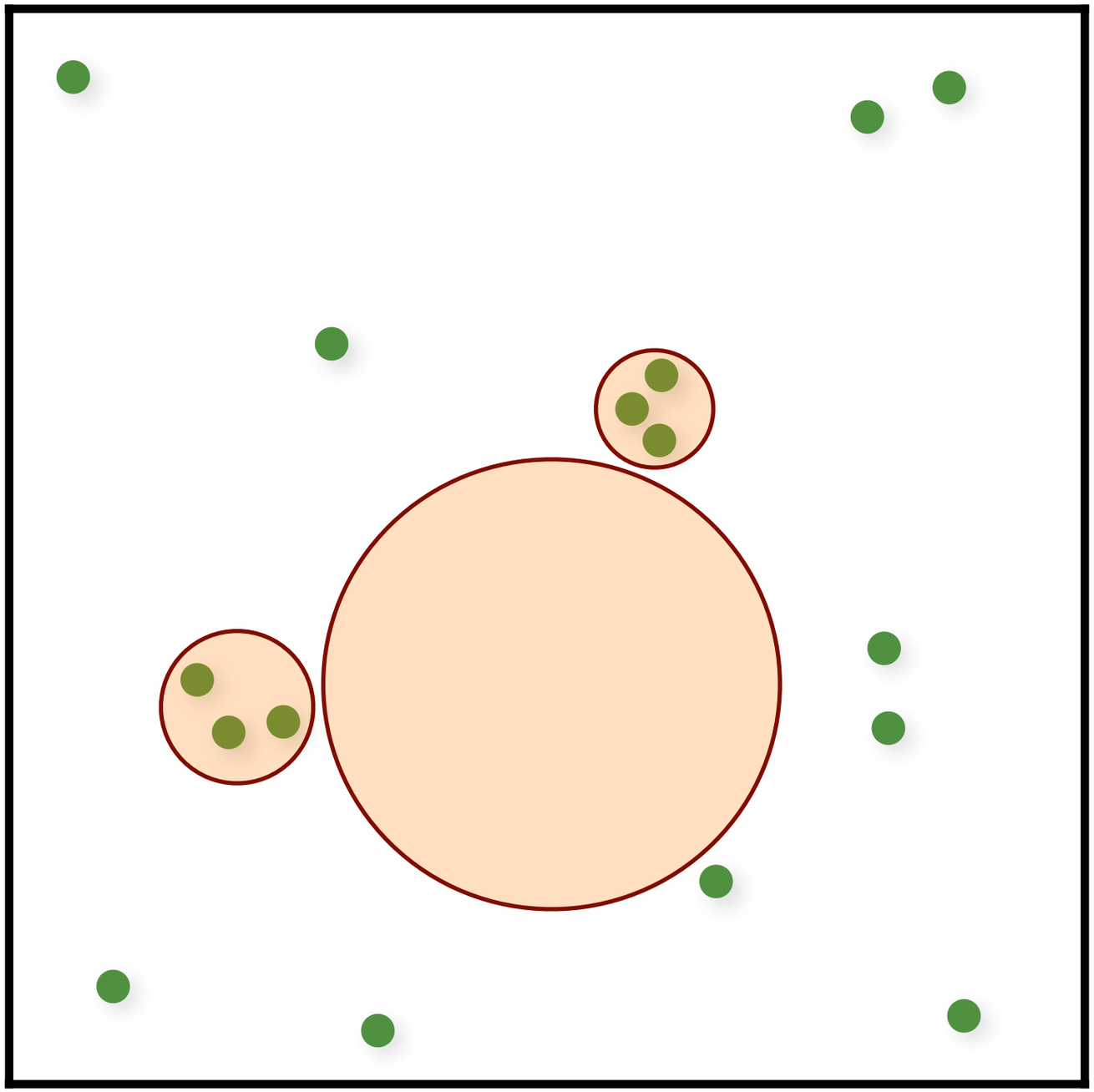
$$\sigma\left[\left\langle F^N \right\rangle\right] = \frac{1}{\sqrt{N}}\sigma\left[Y\right]$$ ← remember, this assumed uncorrelated samples

- Reduce the variance of $Y$
  - Importance sampling
  - Control variates
- Relax assumption of uncorrelated samples

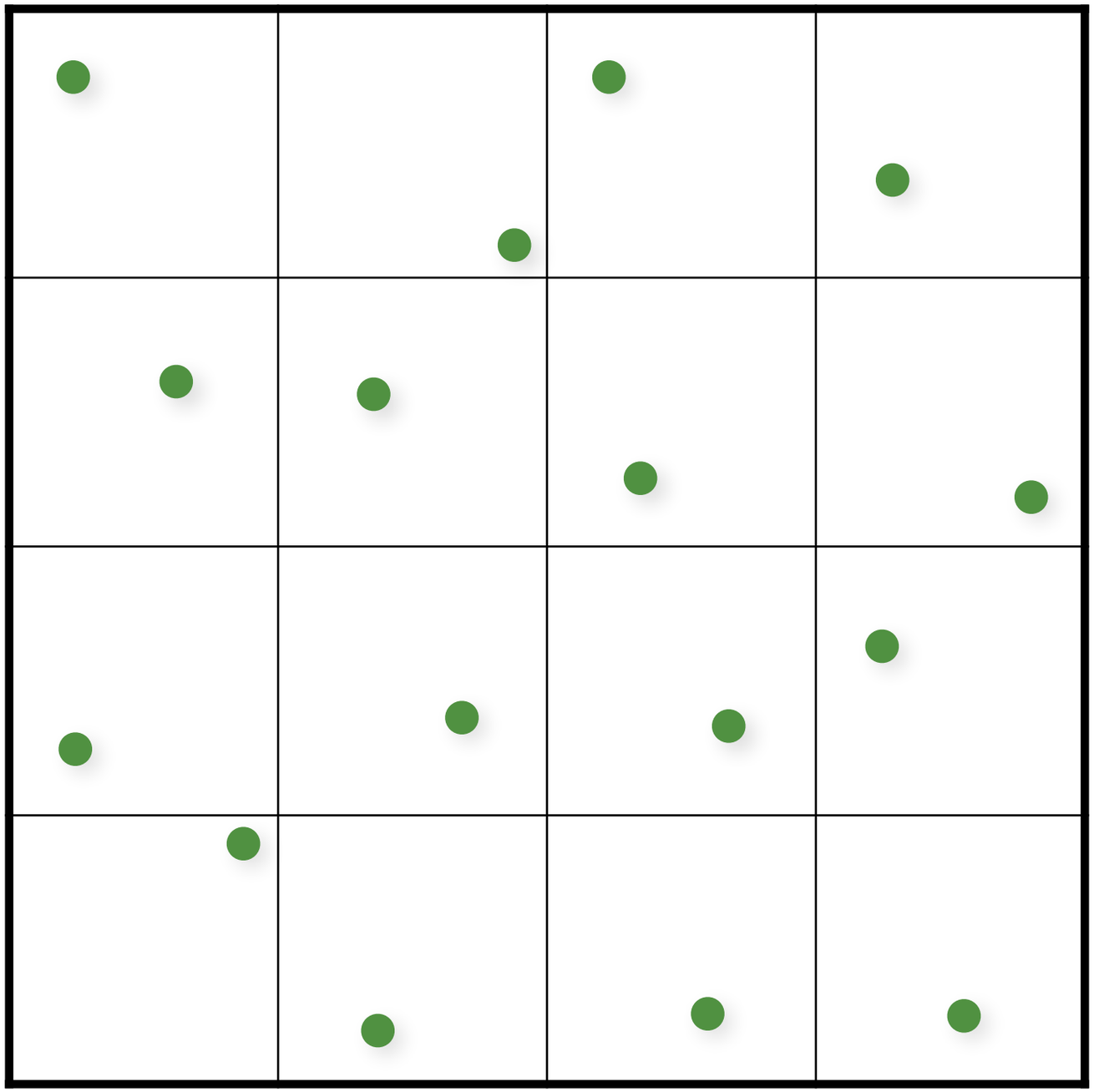
# Careful Sample Placement



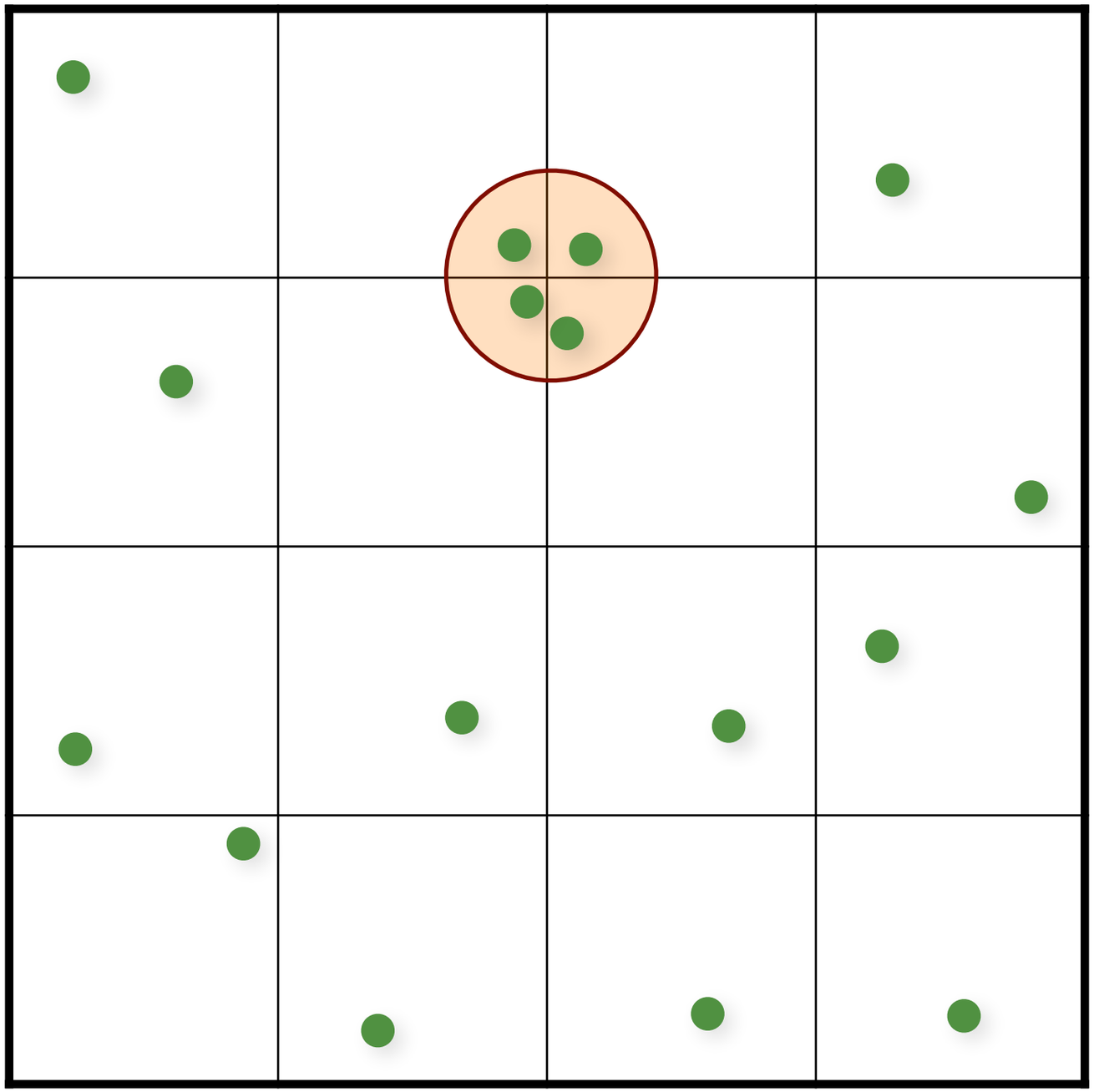
# Random Sampling



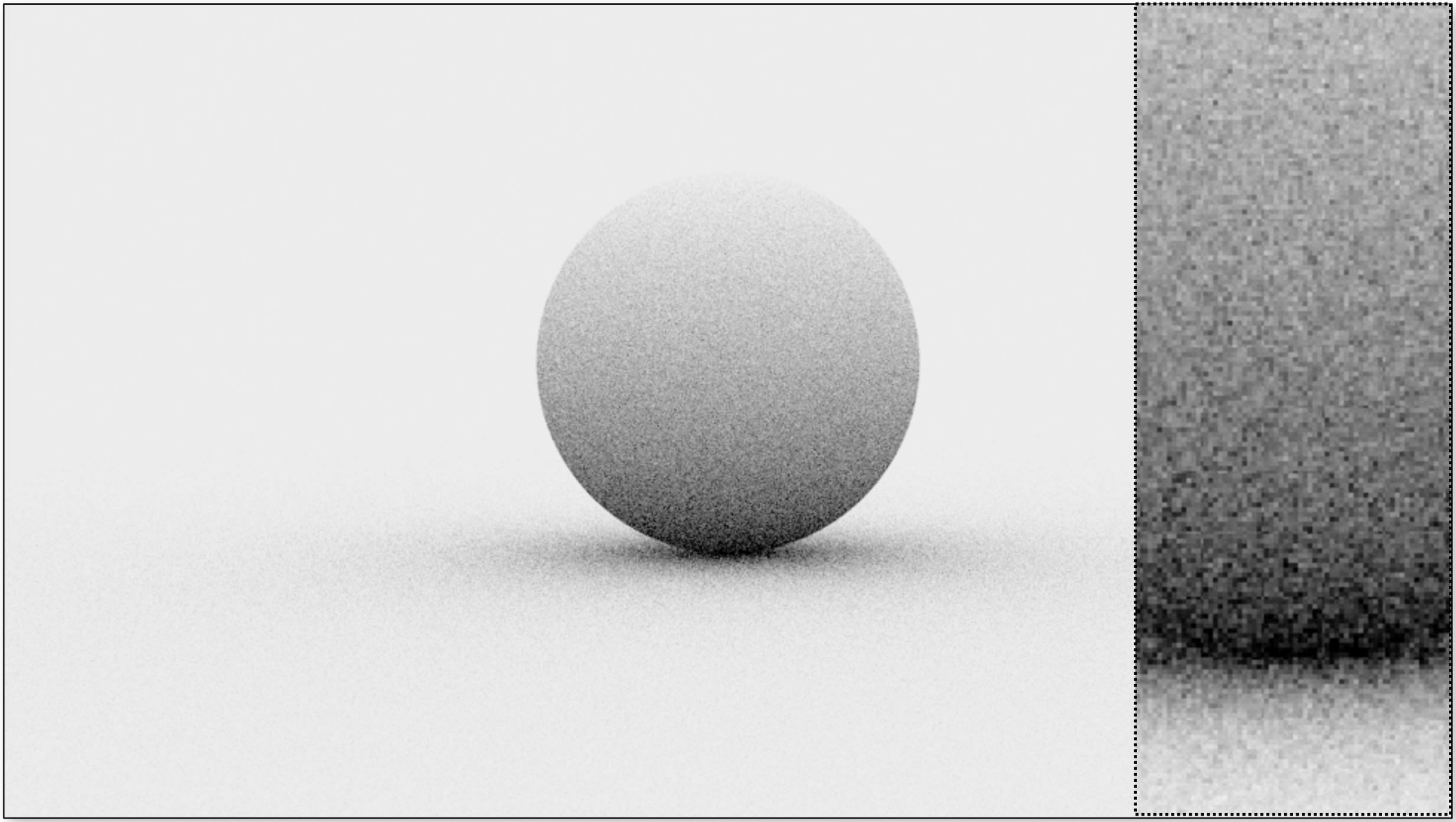
# Stratified (Jittered) Sampling



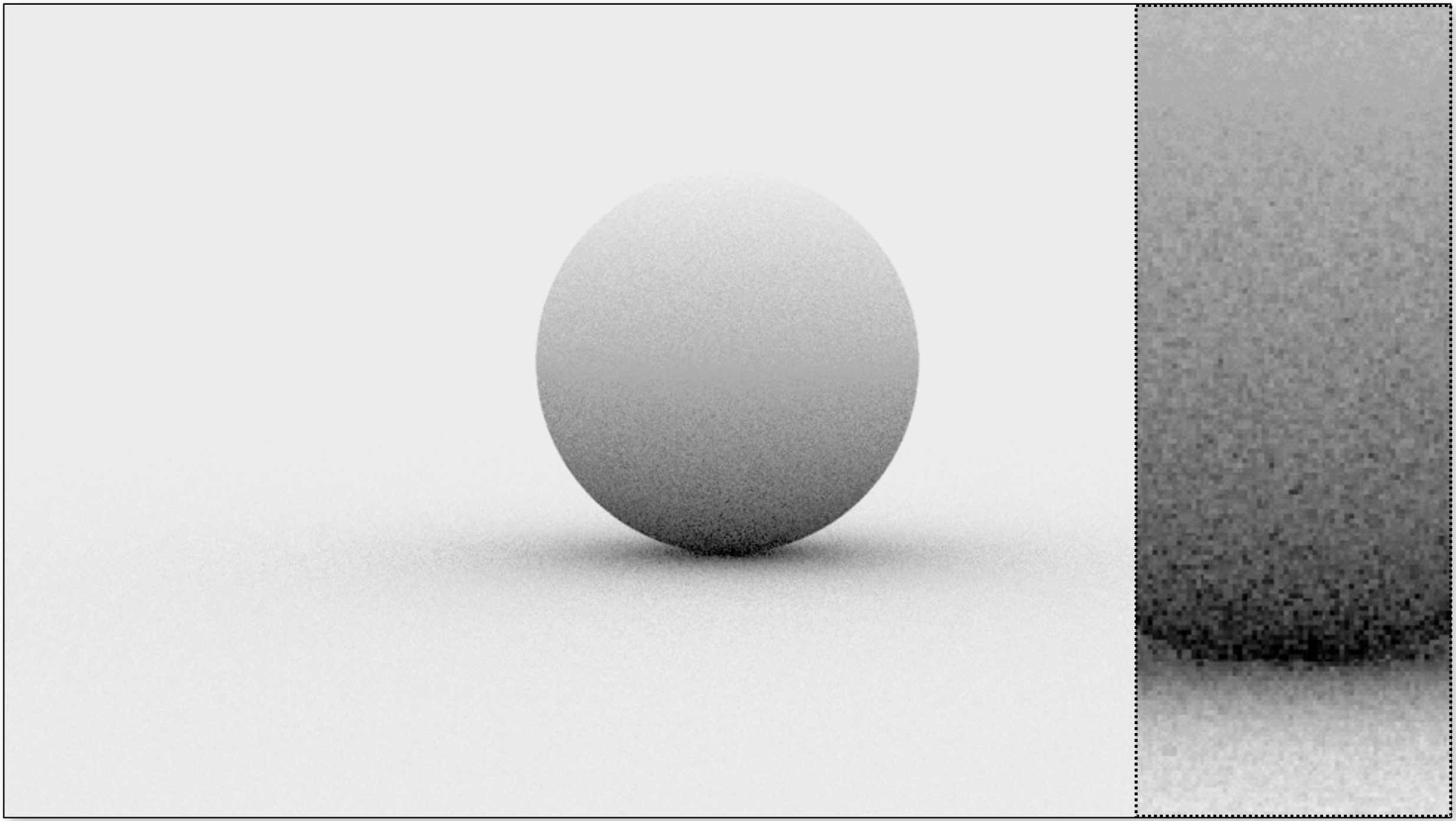
# Stratified (Jittered) Sampling



# Monte Carlo (16 random samples)

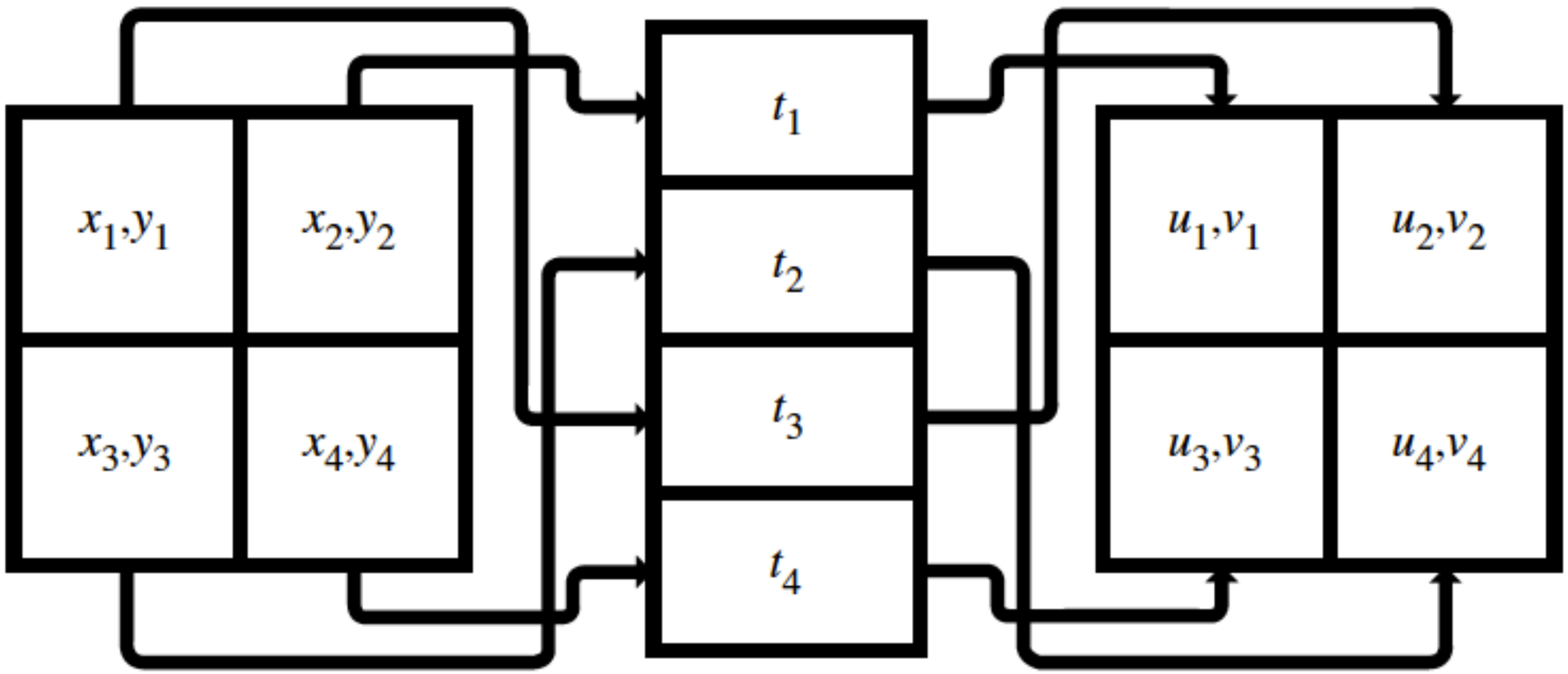


# Monte Carlo (16 stratified samples)

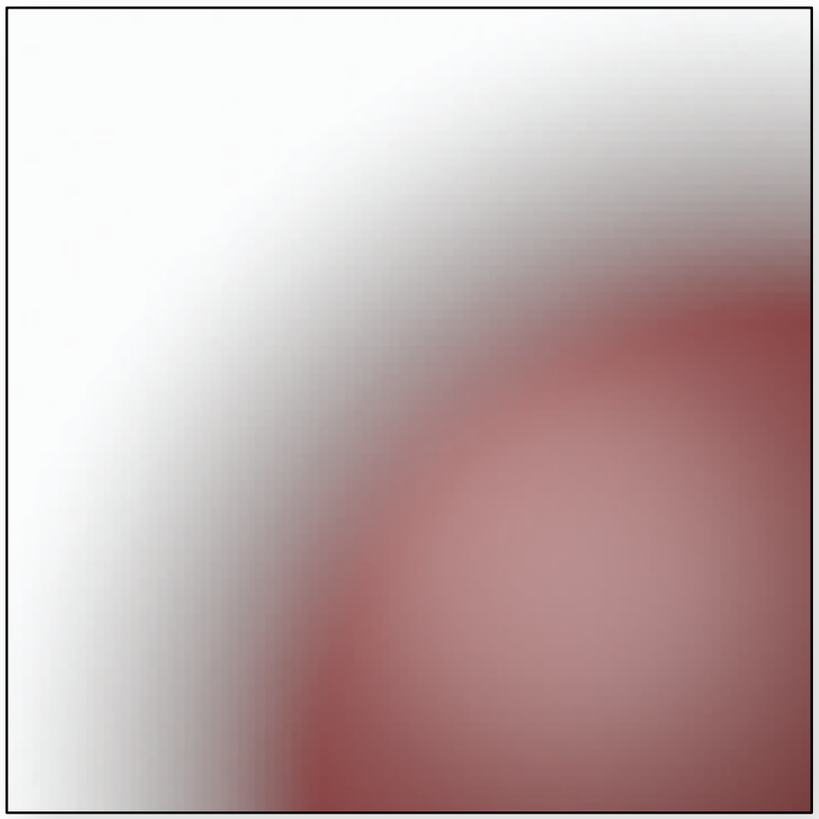


# Stratifying in Higher Dimensions

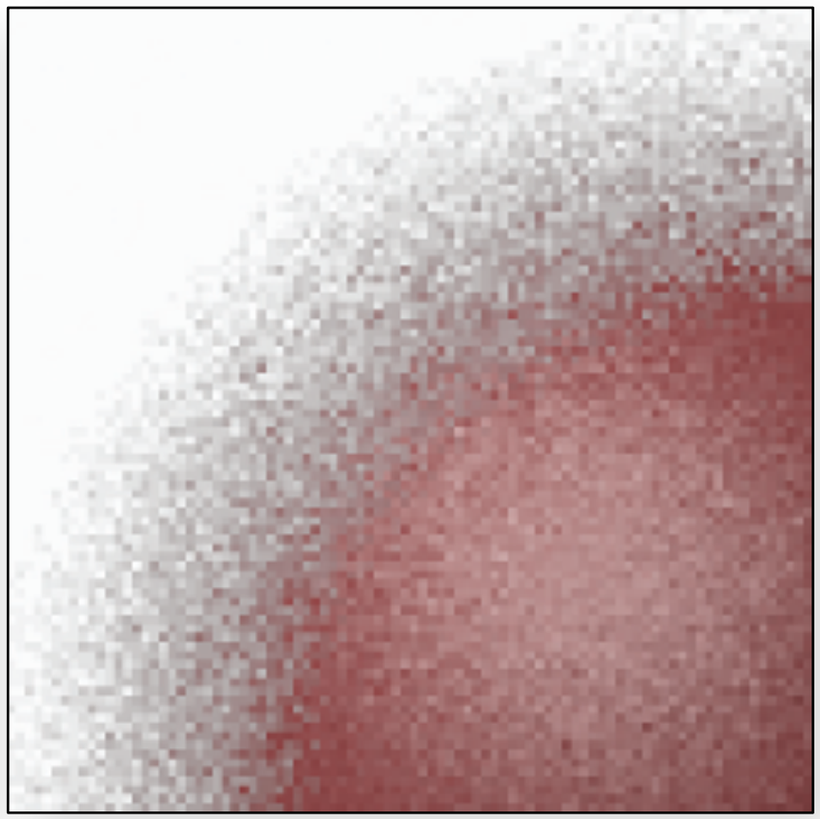
- Stratification requires $N^d$ samples.
- Inconvenient for large $d$
- Compute stratified samples in lower-dimensions (e.g. (x,y), t, (u,v)), and randomly combine



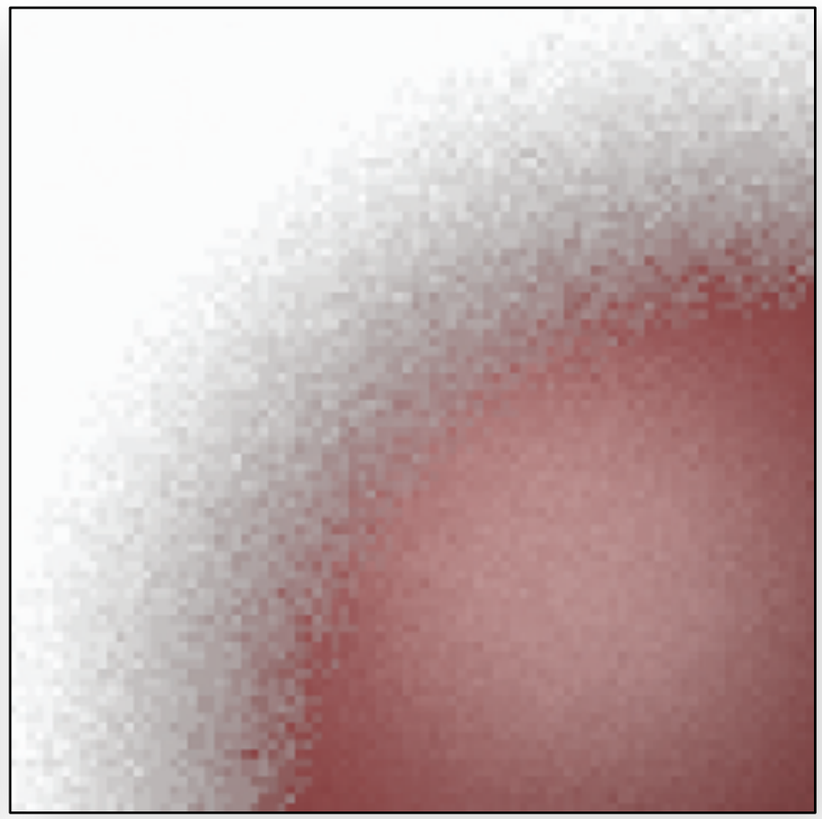
# Depth of Field (4D)



Reference

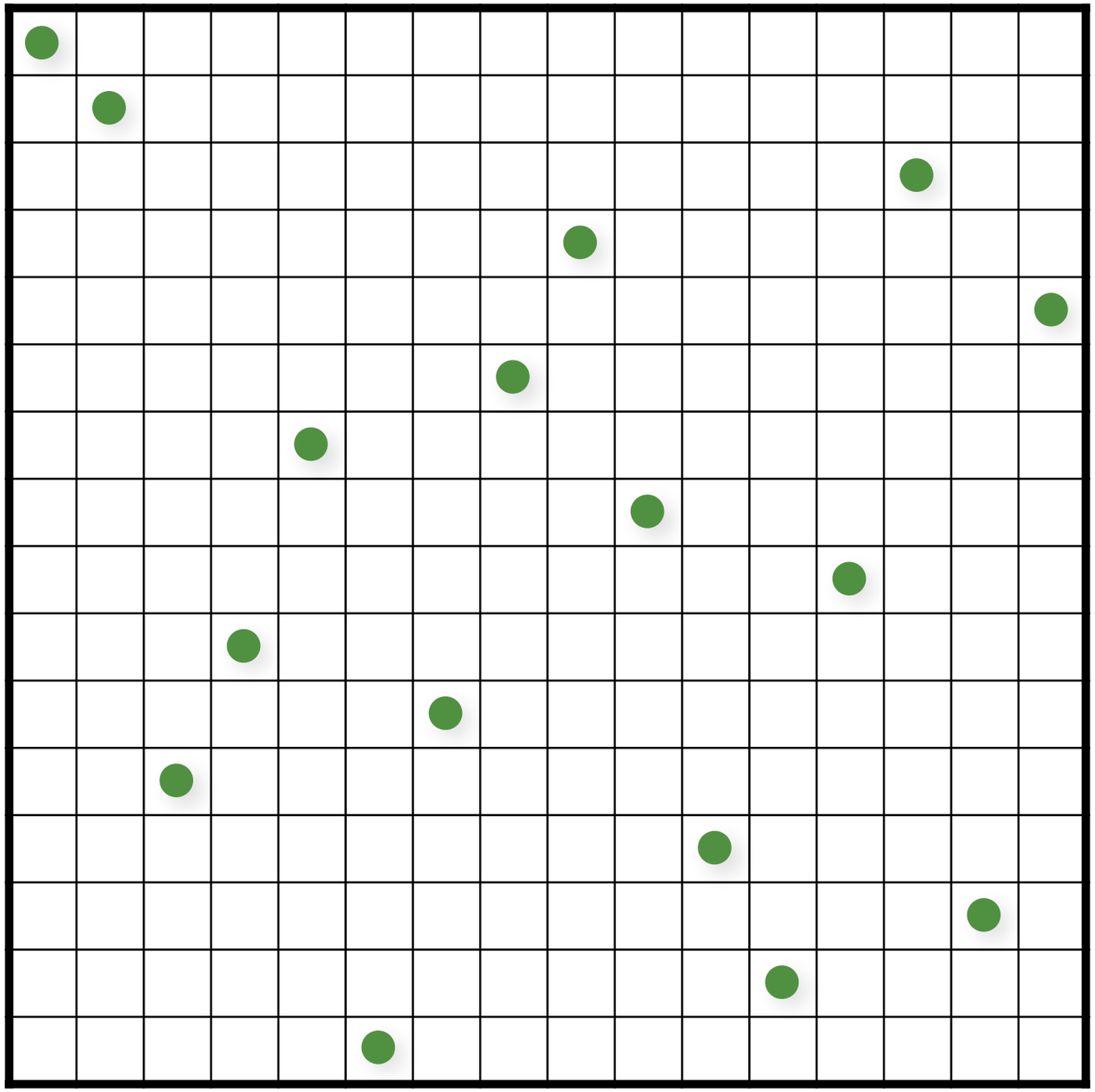


Random Sampling

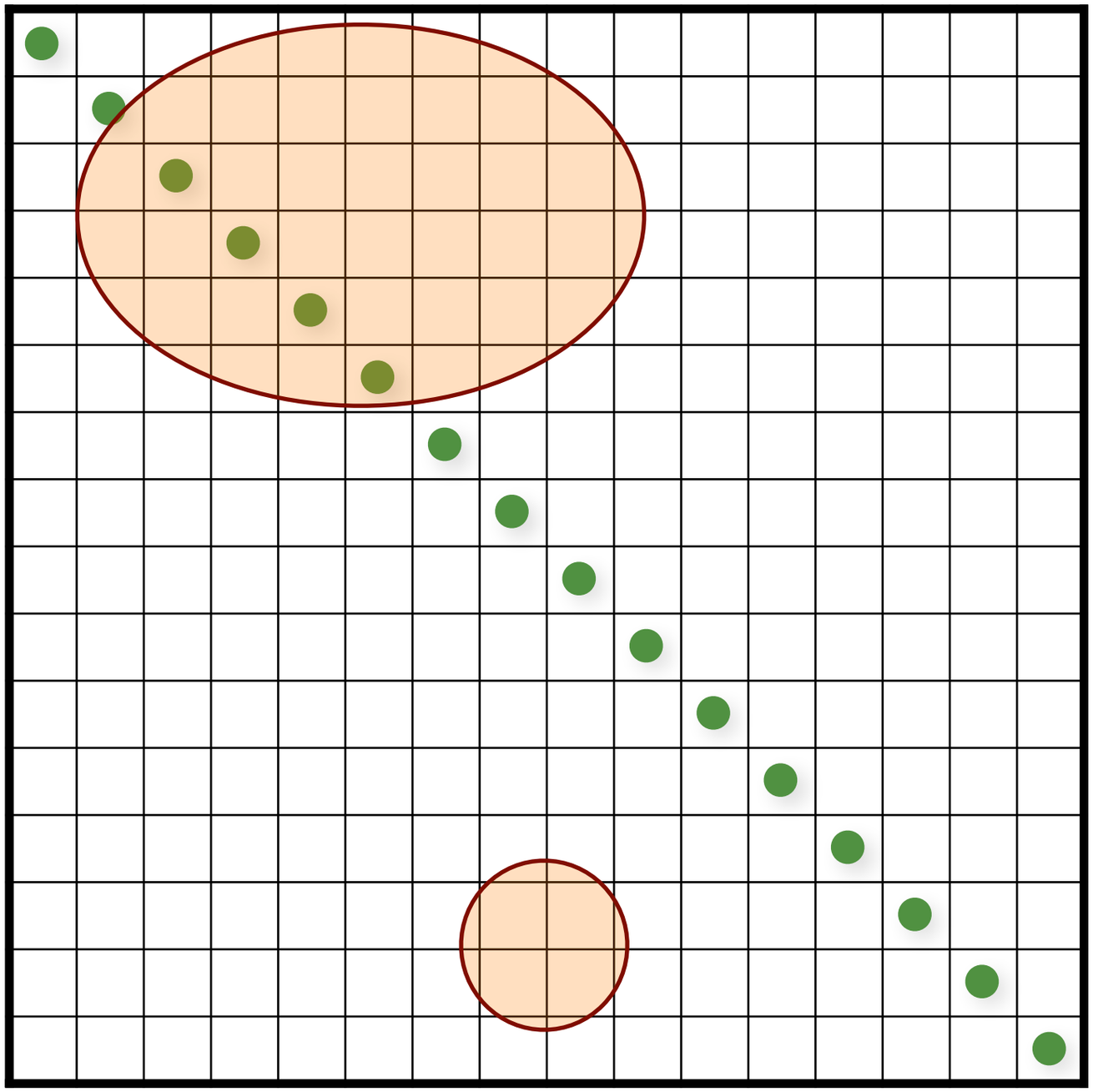


Stratified Sampling

# Latin Hypercube (N-Rooks) Sampling



# Latin Hypercube (N-Rooks) Sampling

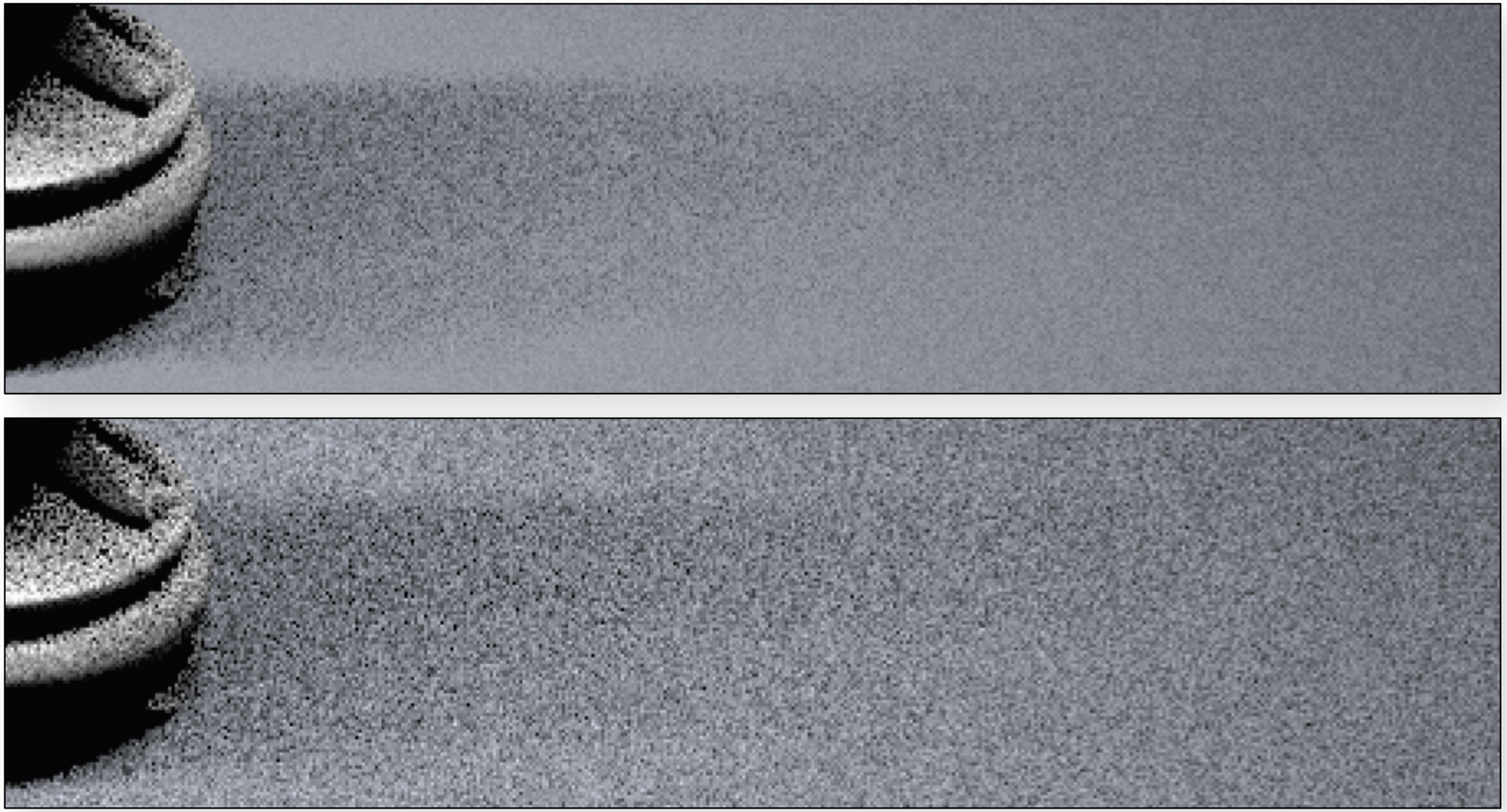


Shuffle columns

# Soft Shadows

# Soft Shadows: Sample Allocation

1 image sample x 16 light samples

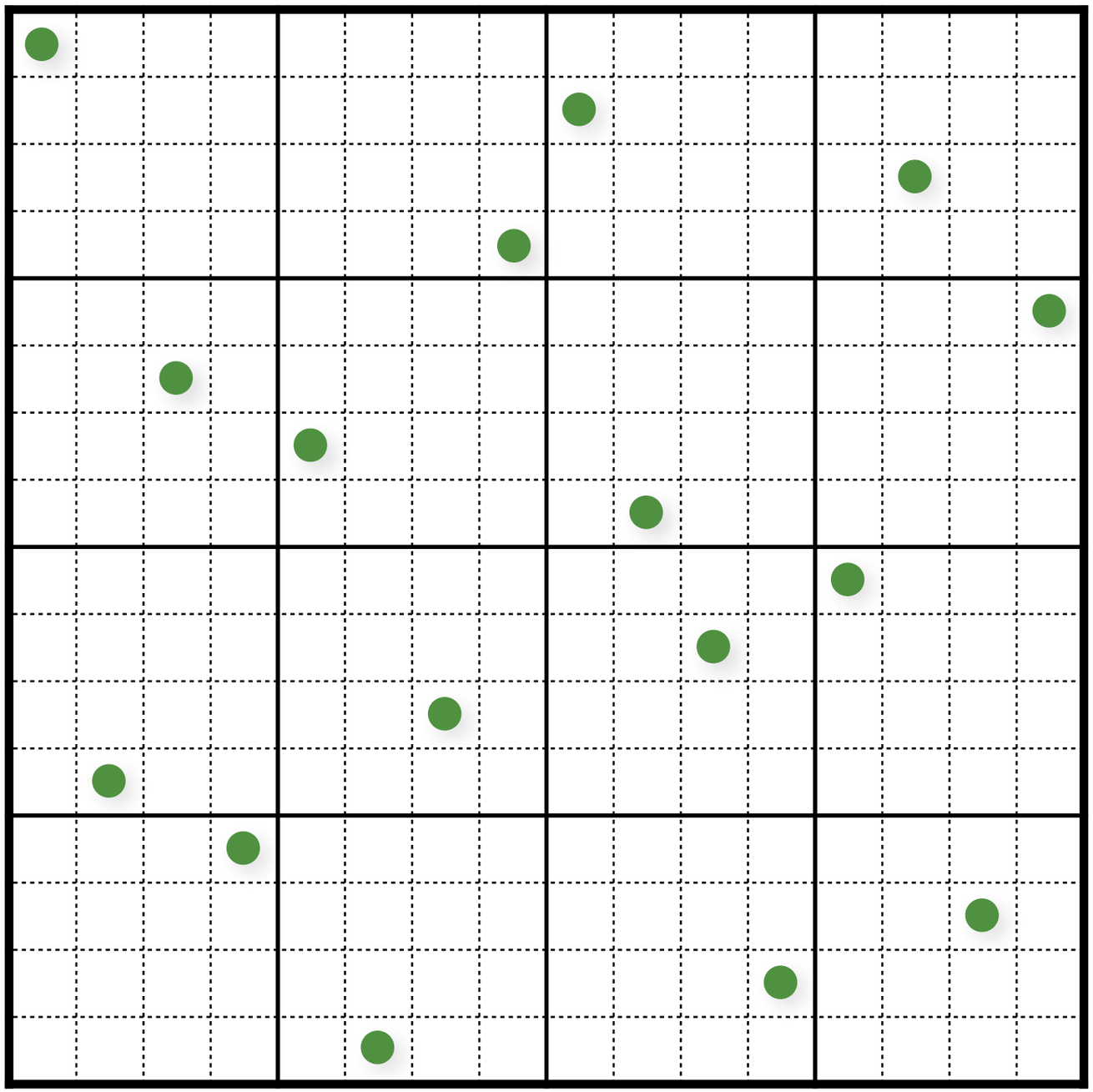


16 image samples x 1 light sample

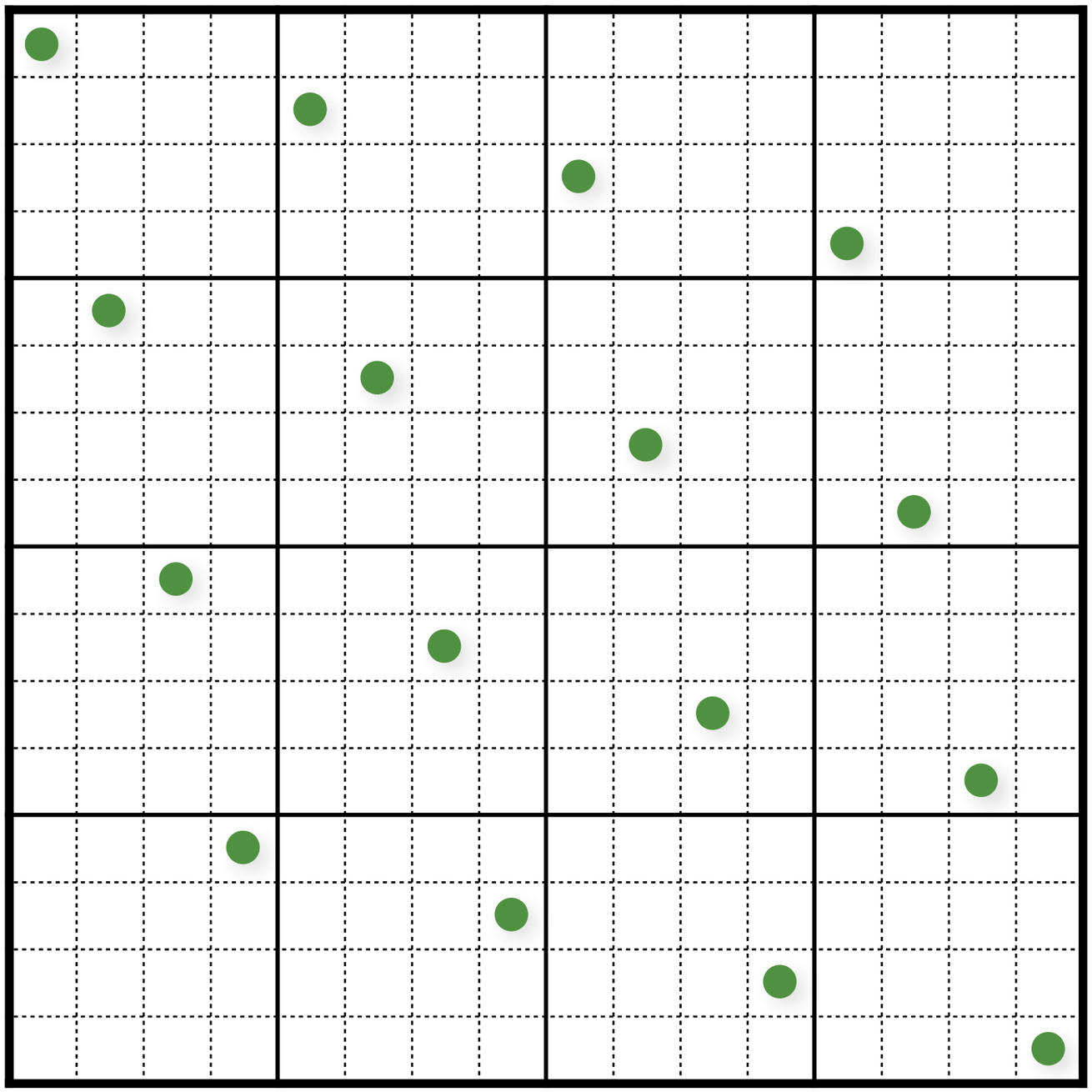
# Multi-Jittered Sampling

- Kenneth Chiu, Peter Shirley, and Changyaw Wang. "Multi-jittered sampling." In *Graphics Gems IV*, pp. 370–374. Academic Press, May 1994.
  - combine N-Rooks and Jittered stratification constraints

# Multi-Jittered Sampling

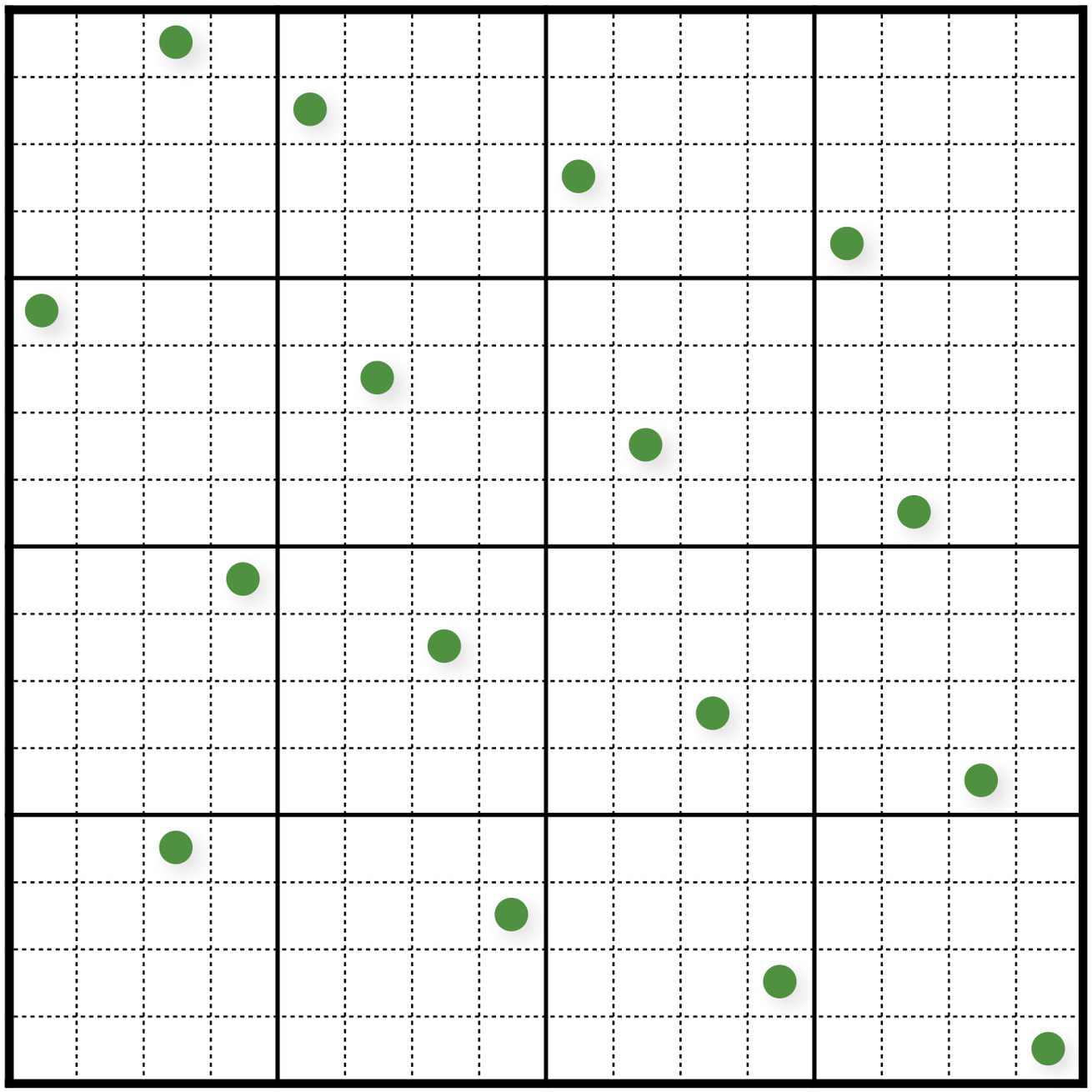


# Multi-Jittered Sampling



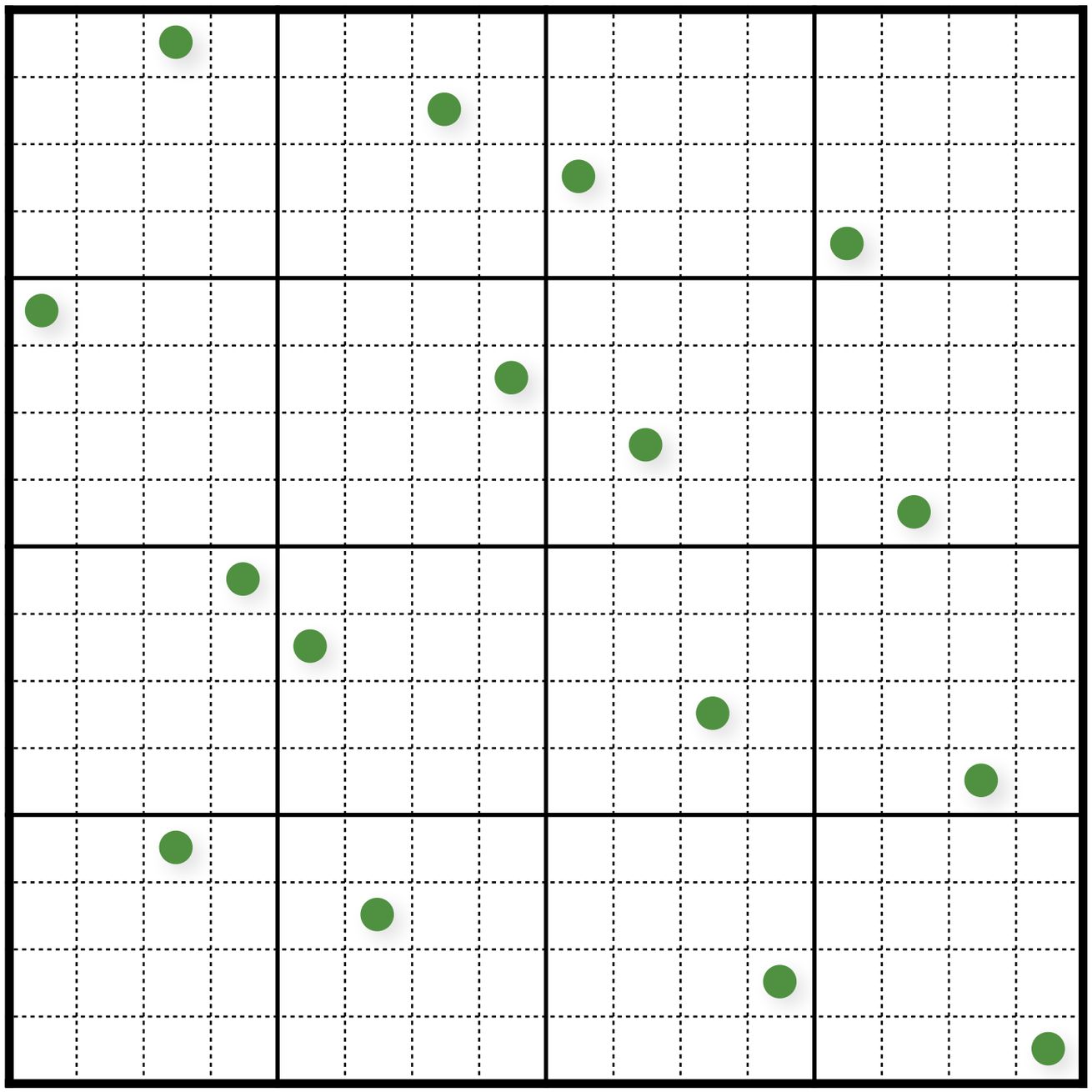
Shuffle x-coords

# Multi-Jittered Sampling



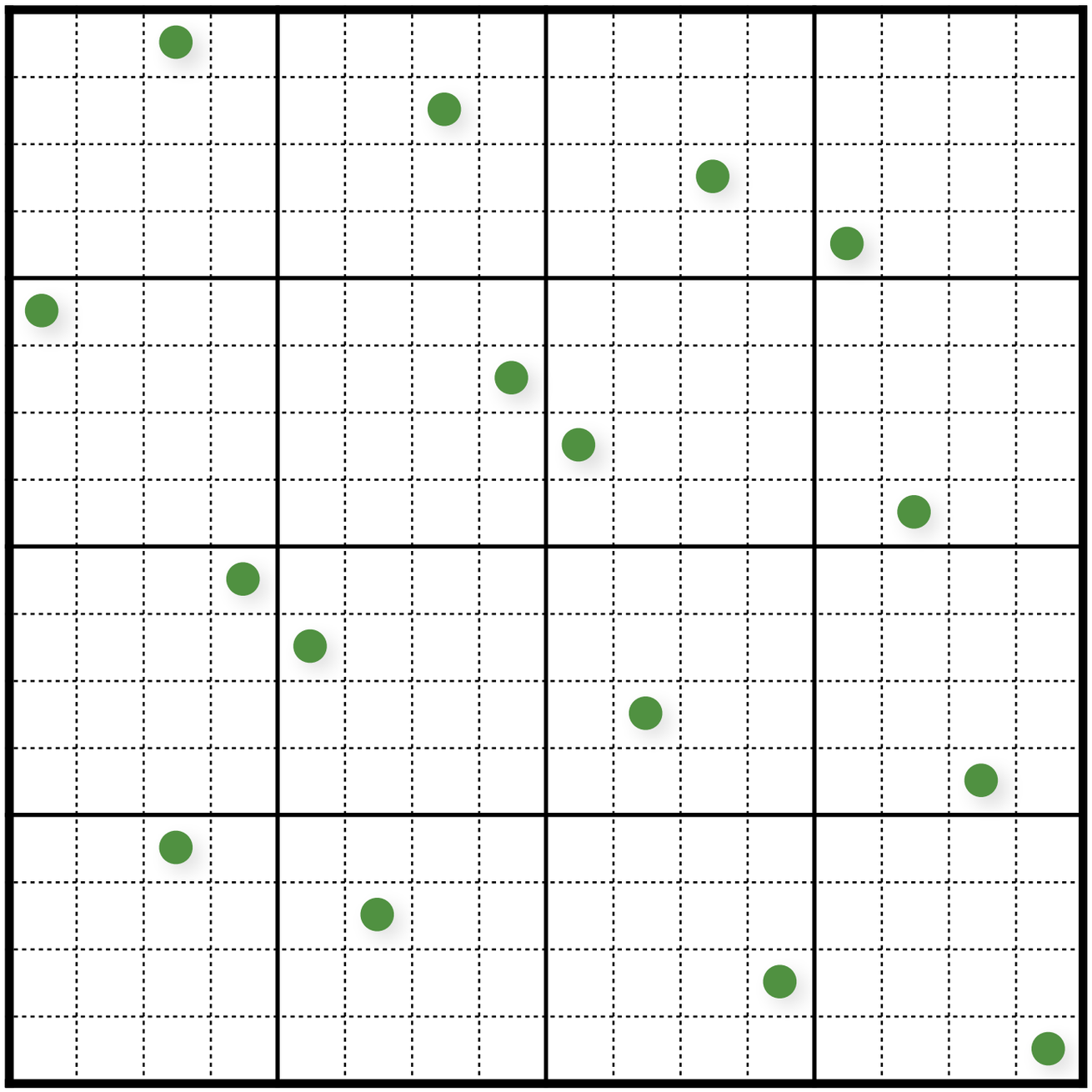
Shuffle x-coords

# Multi-Jittered Sampling



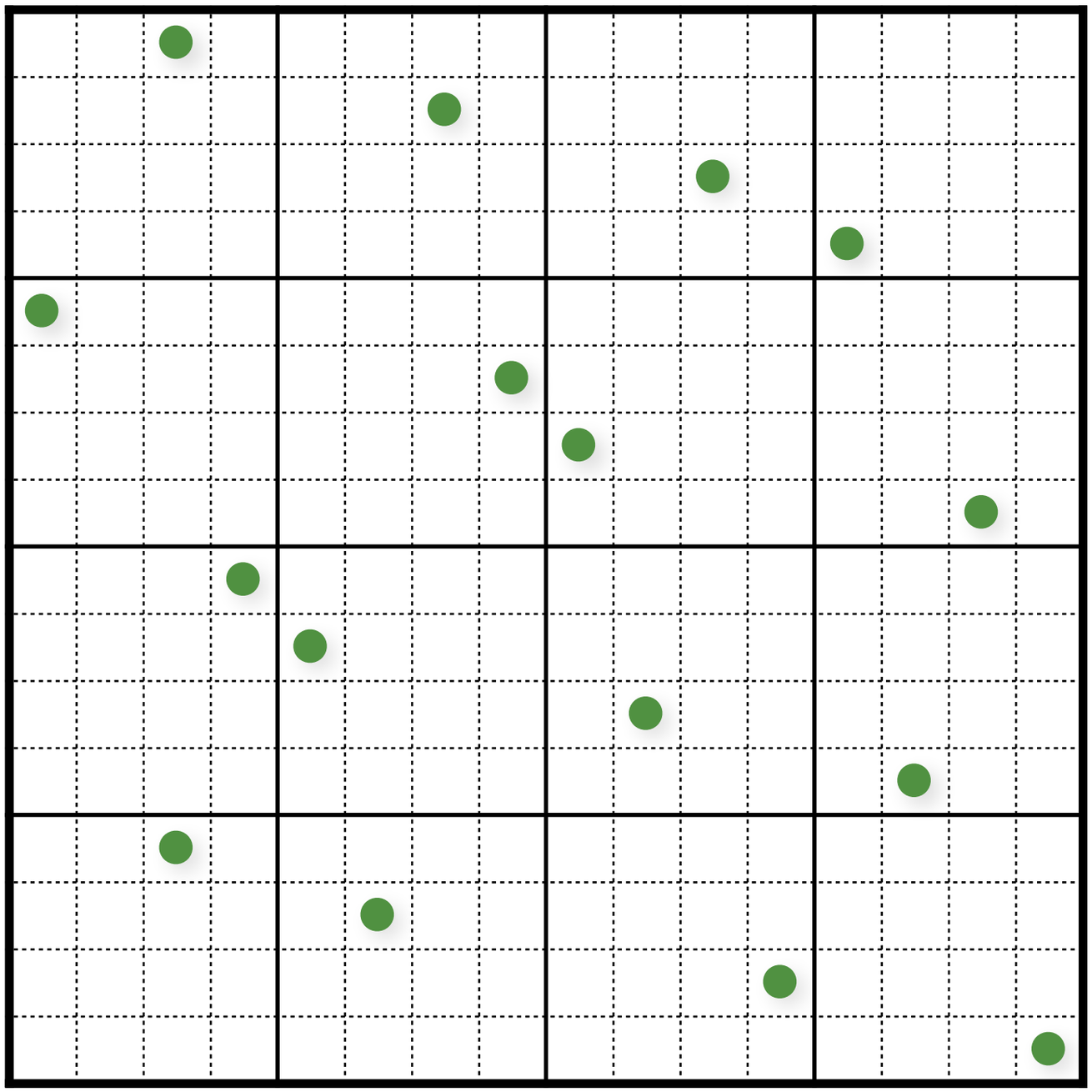
Shuffle x-coords

# Multi-Jittered Sampling



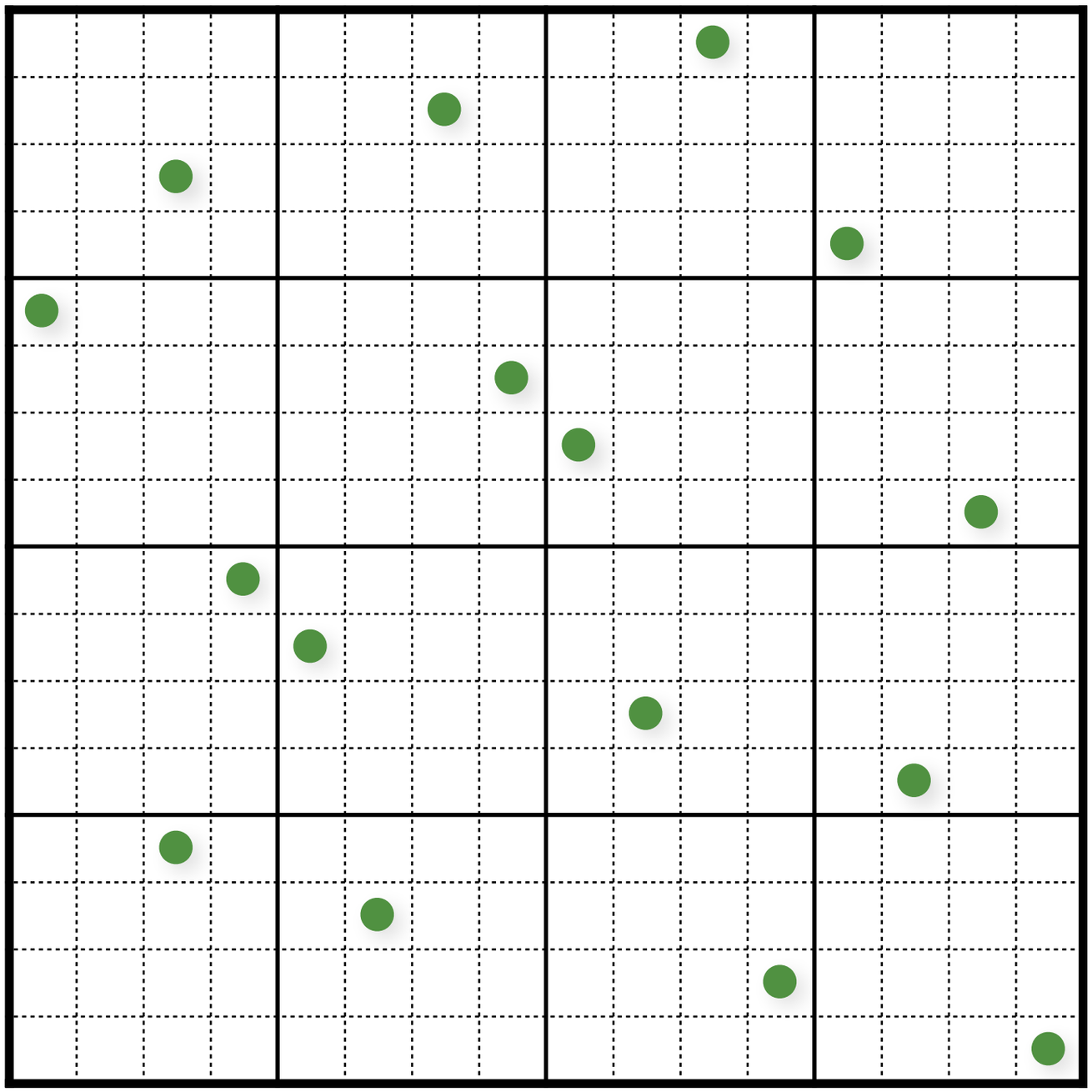
Shuffle x-coords

# Multi-Jittered Sampling



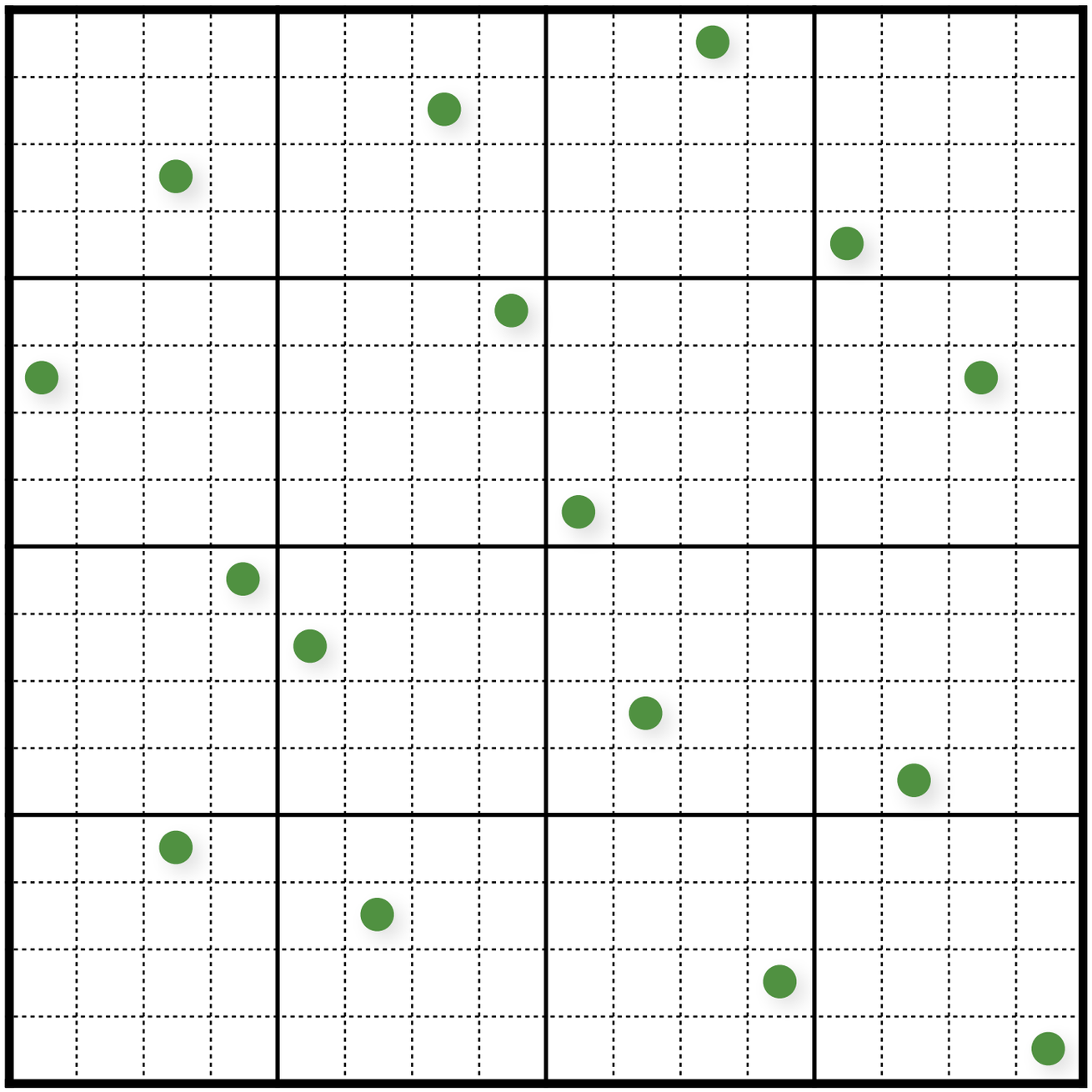
Shuffle ~~x~~y-coords

# Multi-Jittered Sampling



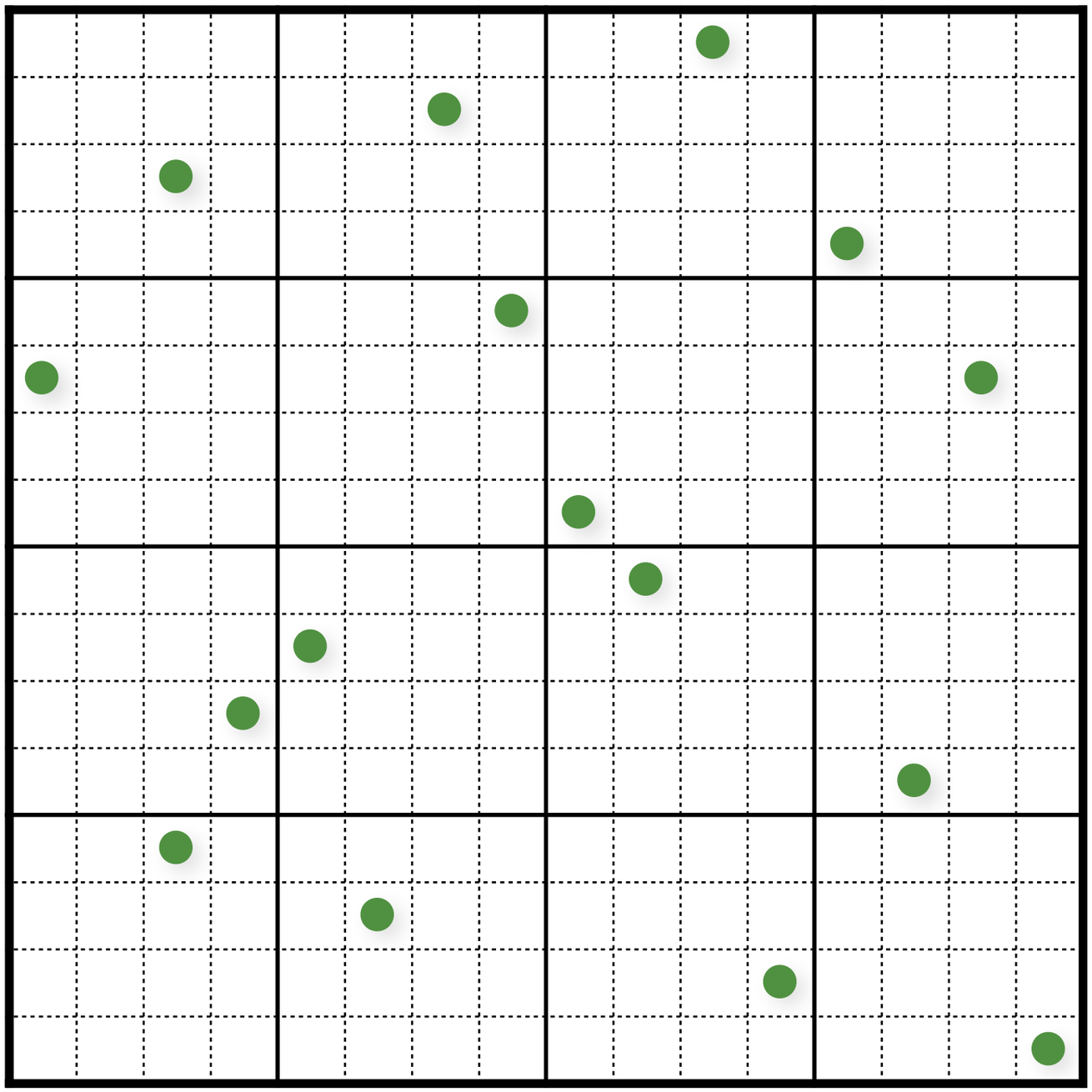
Shuffle y-coords

# Multi-Jittered Sampling



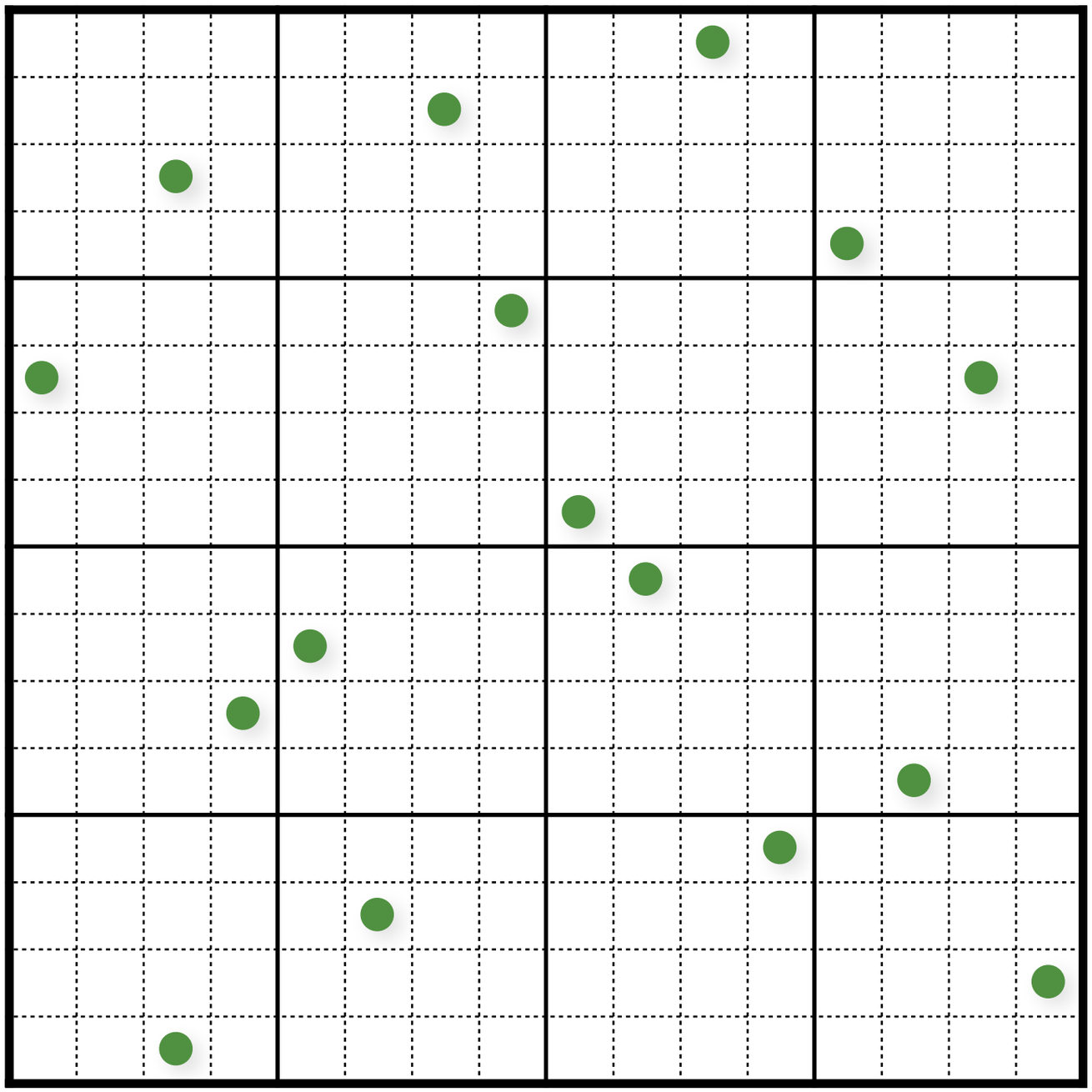
Shuffle y-coords

# Multi-Jittered Sampling



Shuffle y-coords

# Multi-Jittered Sampling



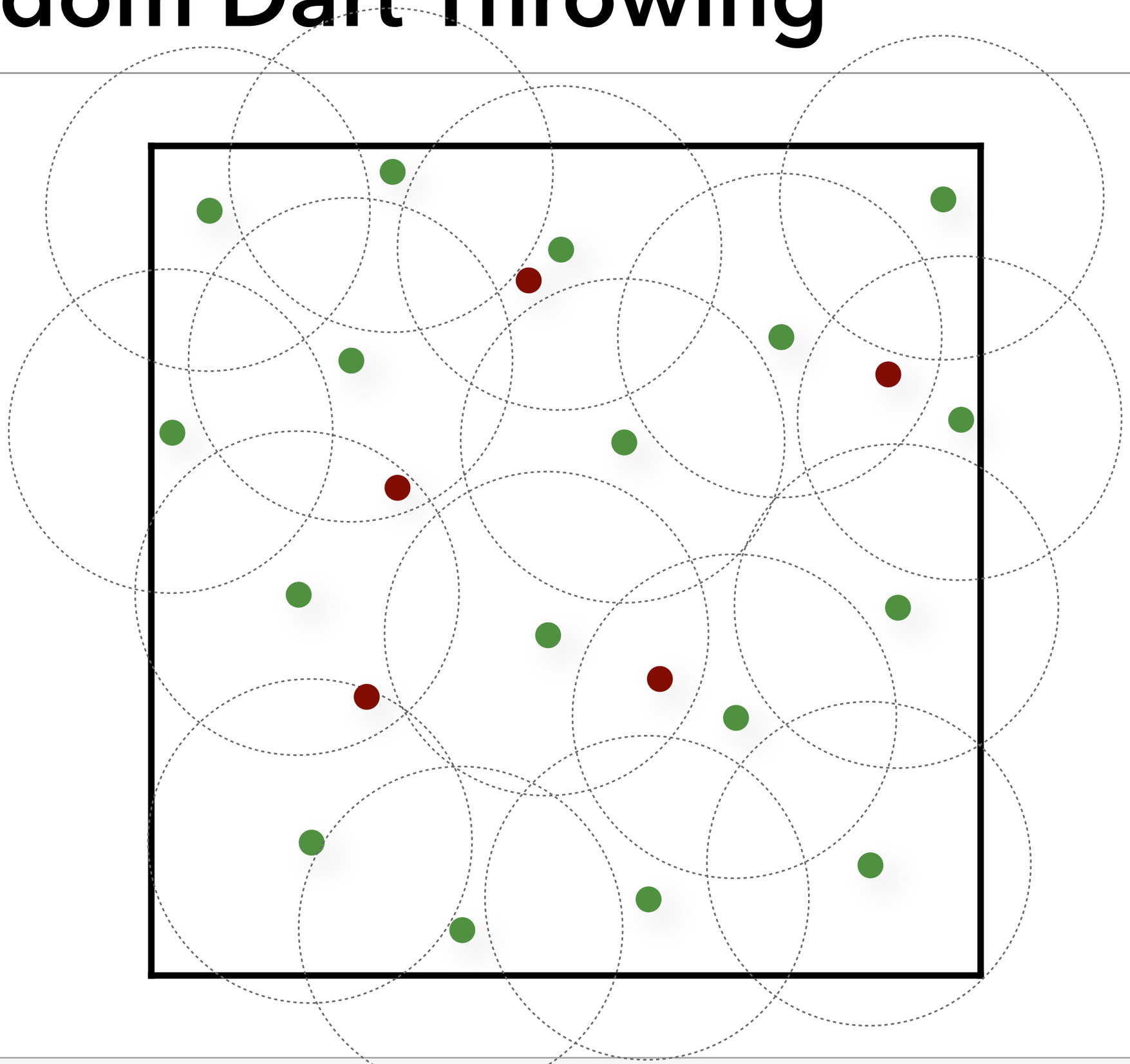
Shuffle y-coords

# Visual Break

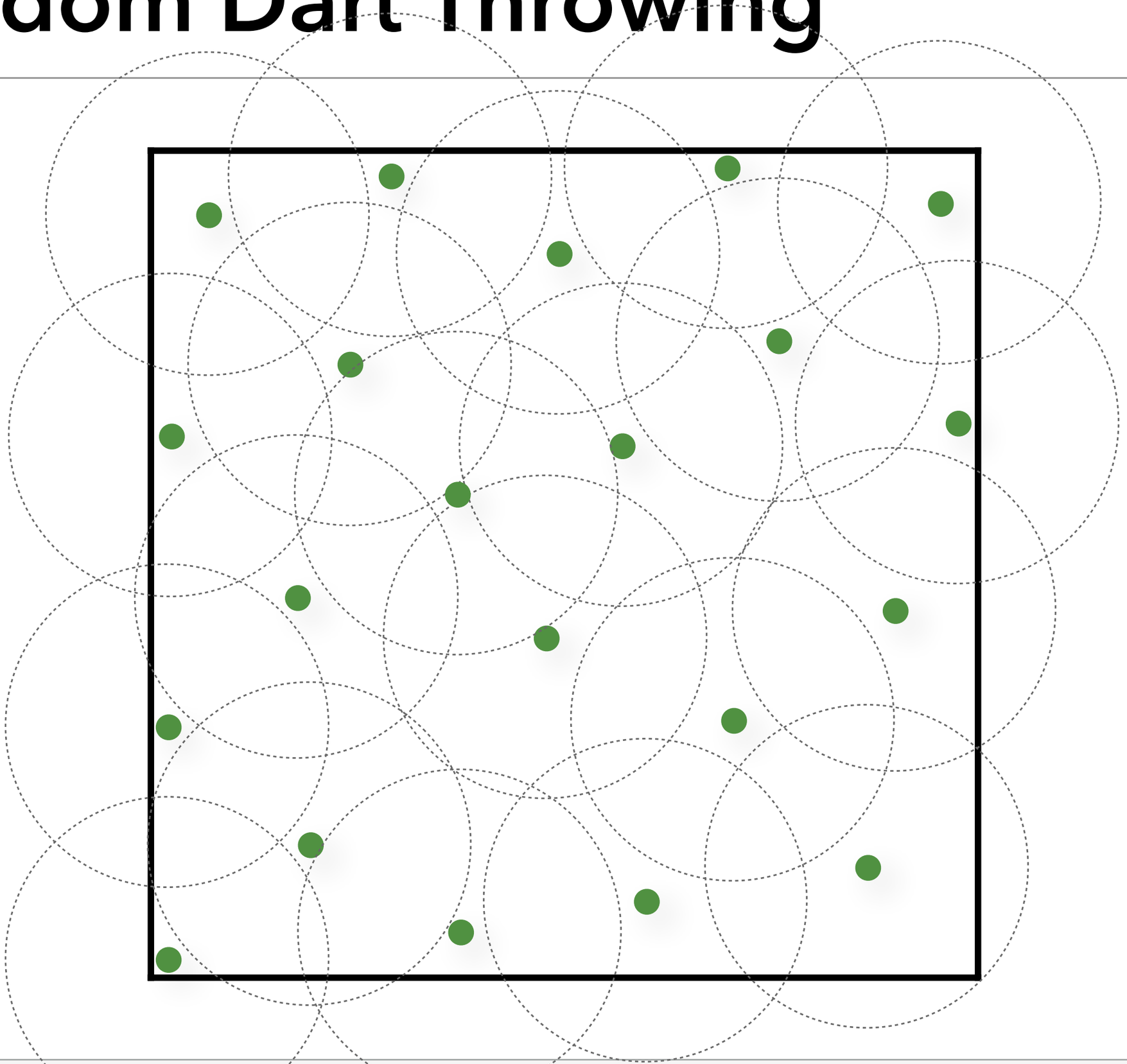
# Poisson-Disk/Blue-Noise Sampling

- Enforce a minimum distance between points
- Poisson-Disk Sampling:
  - Robert L. Cook. "Stochastic sampling in computer graphics." *ACM Transactions on Graphics*. 1986.
  - Ares Lagae and Philip Dutré. "A comparison of methods for generating Poisson disk distributions." *Computer Graphics Forum*, 2008.

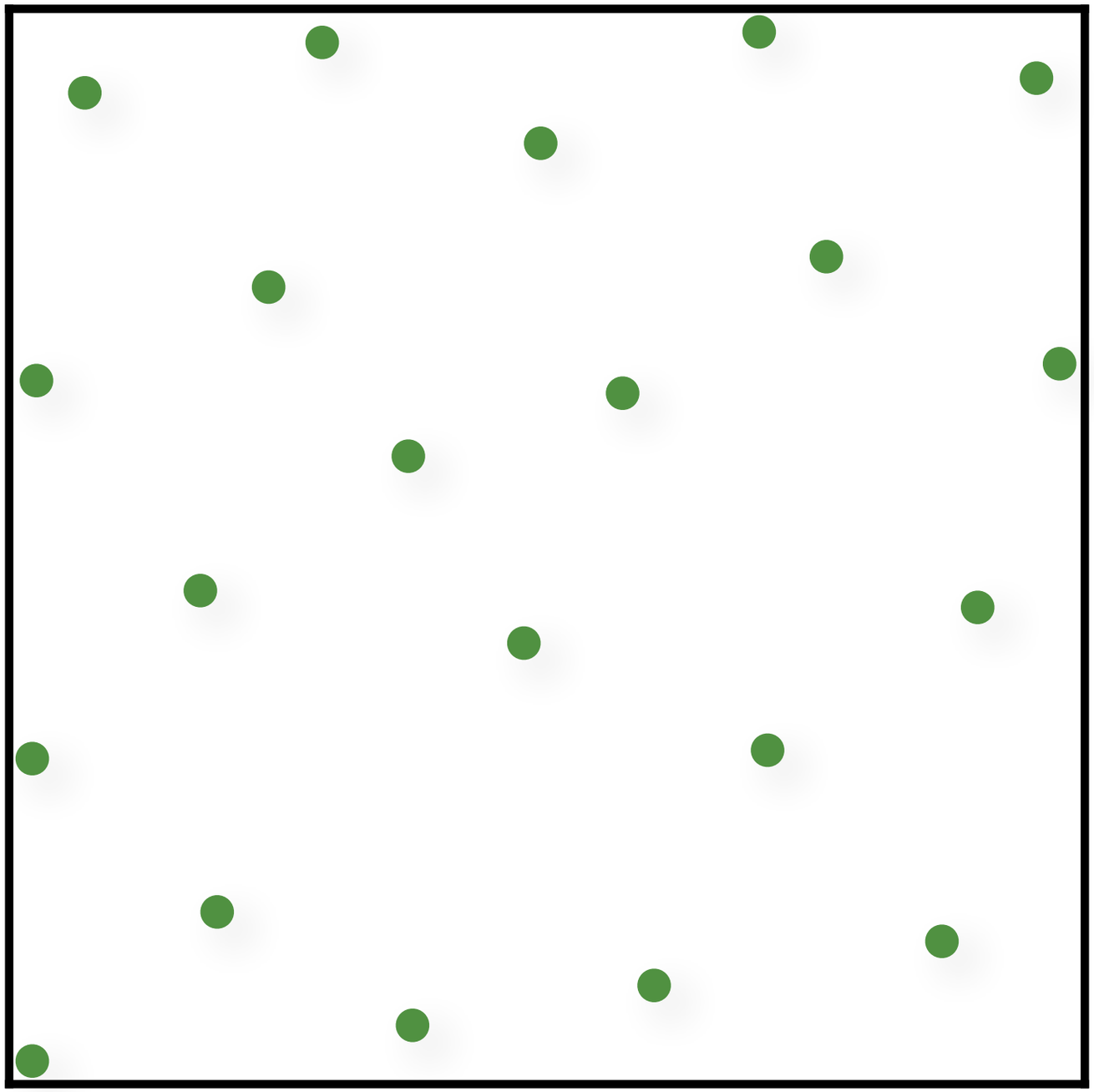
# Random Dart Throwing



# Random Dart Throwing



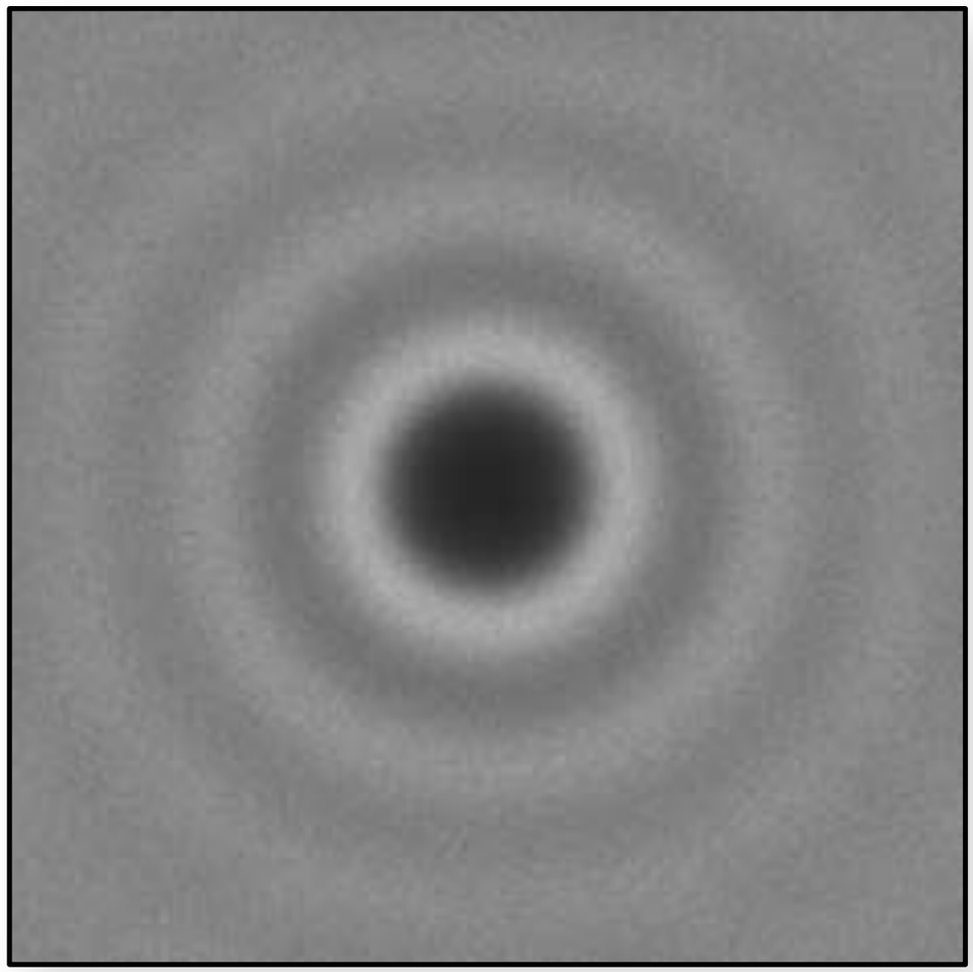
# Random Dart Throwing

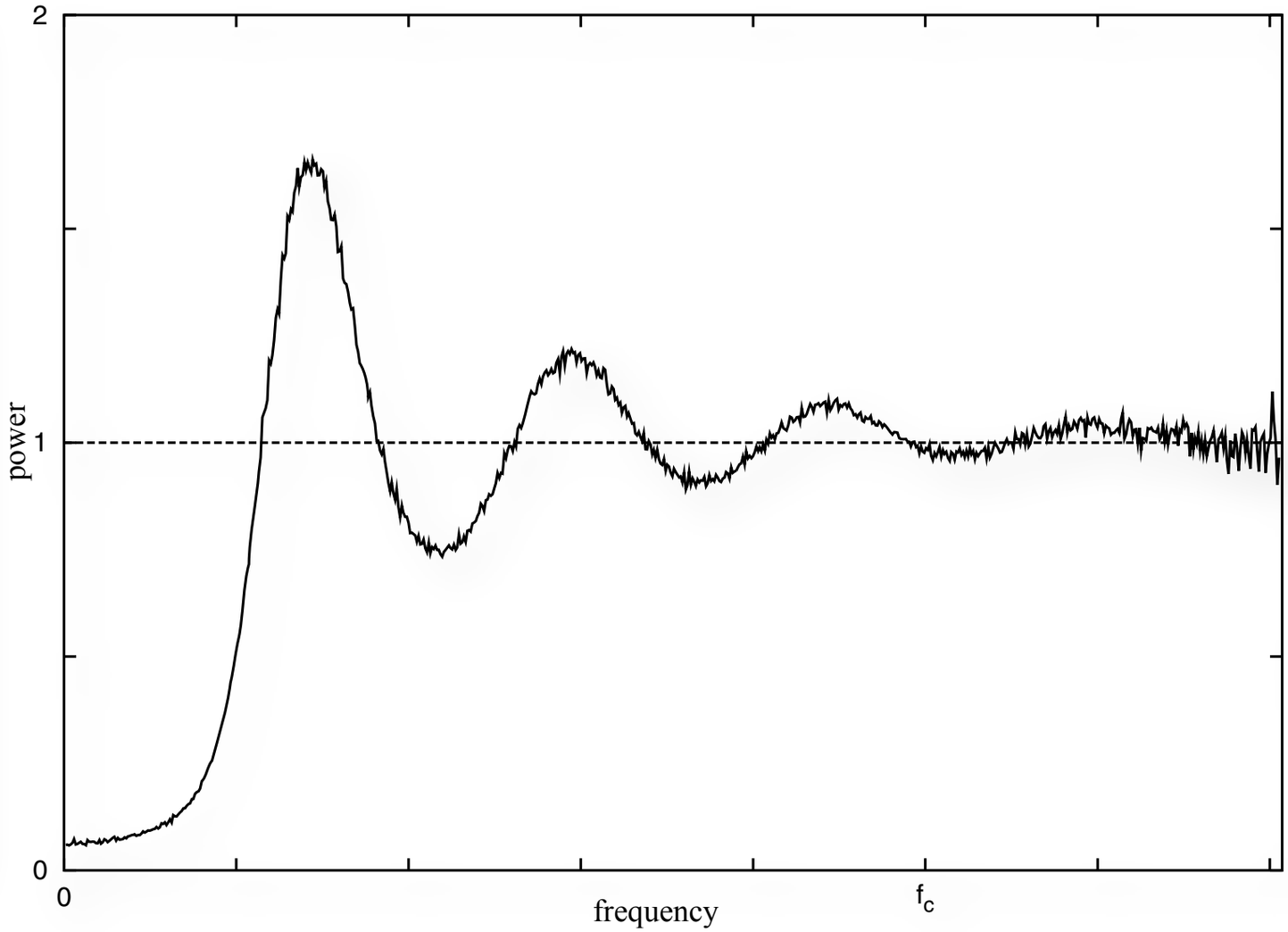


# Stratified Sampling

# Best Candidate Sampling

# Power Spectrum



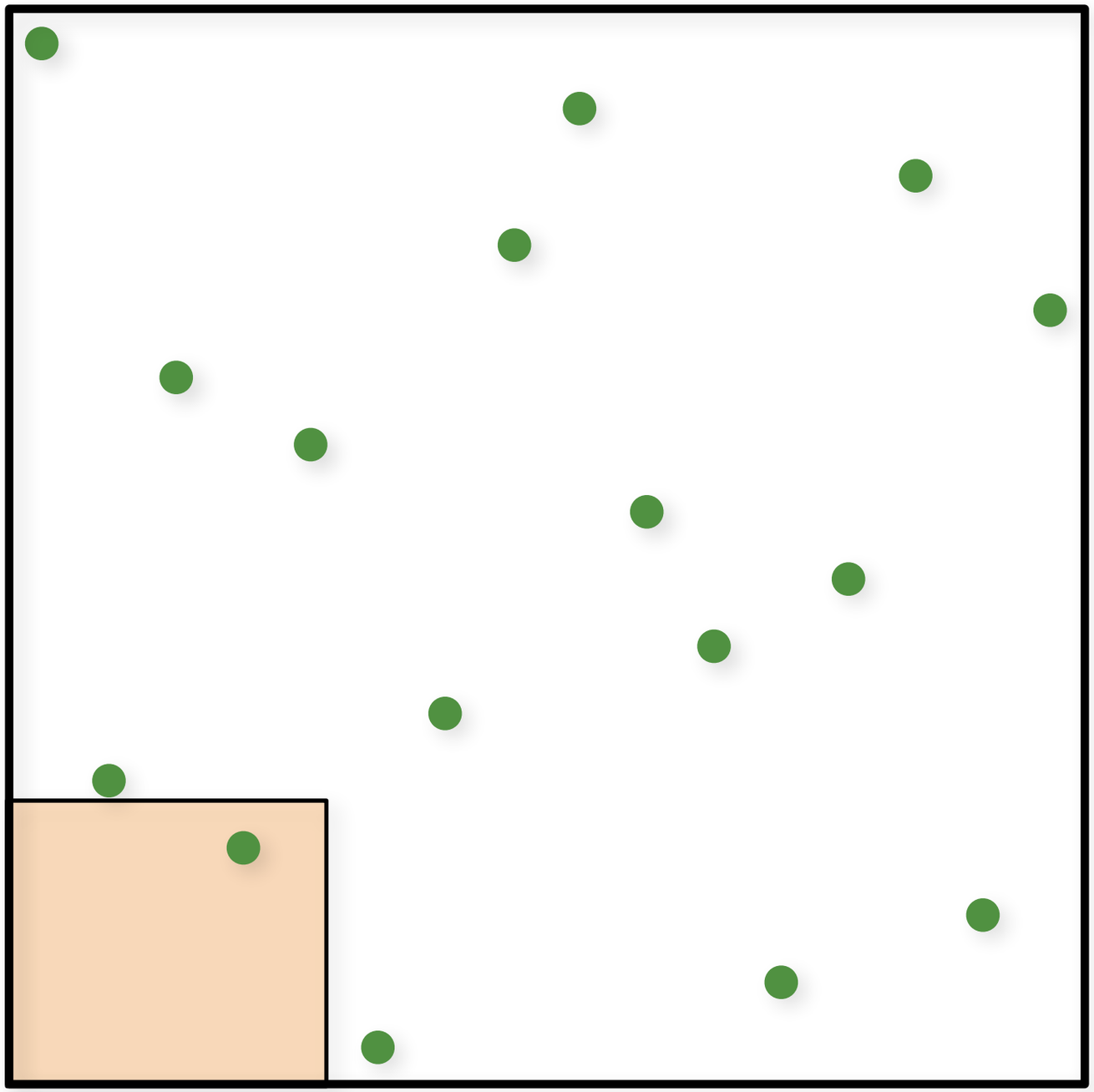


# Visual Break

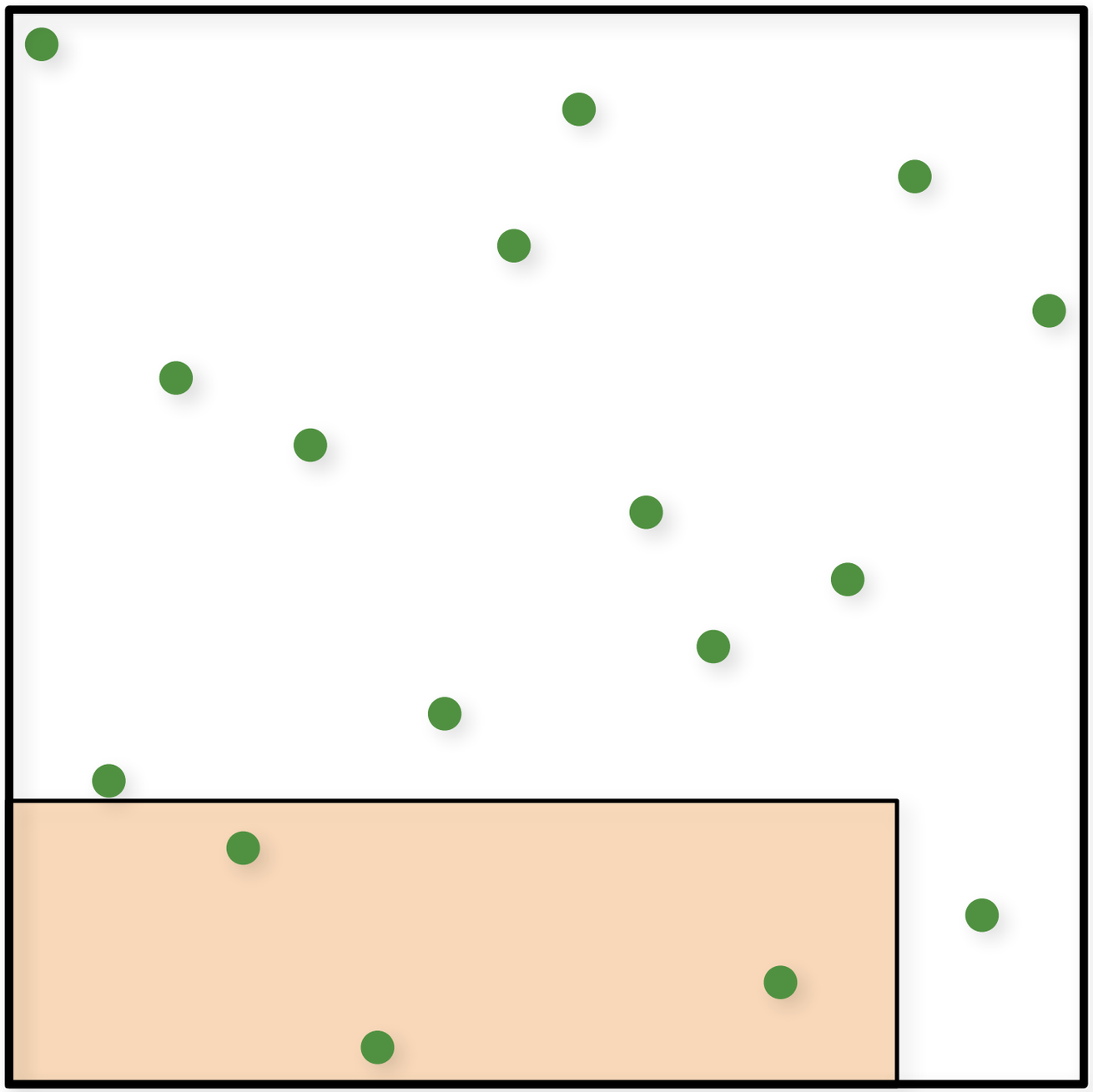
# Discrepancy

- Previous stratified approaches try to minimize "clumping"
- "Discrepancy" is another possible formal definition of clumping: **$D^*(x_1, .., x_n)$**
  - for every possible subregion compute the maximum absolute difference between:
    - fraction of points in the subregion
    - volume of containing subregion

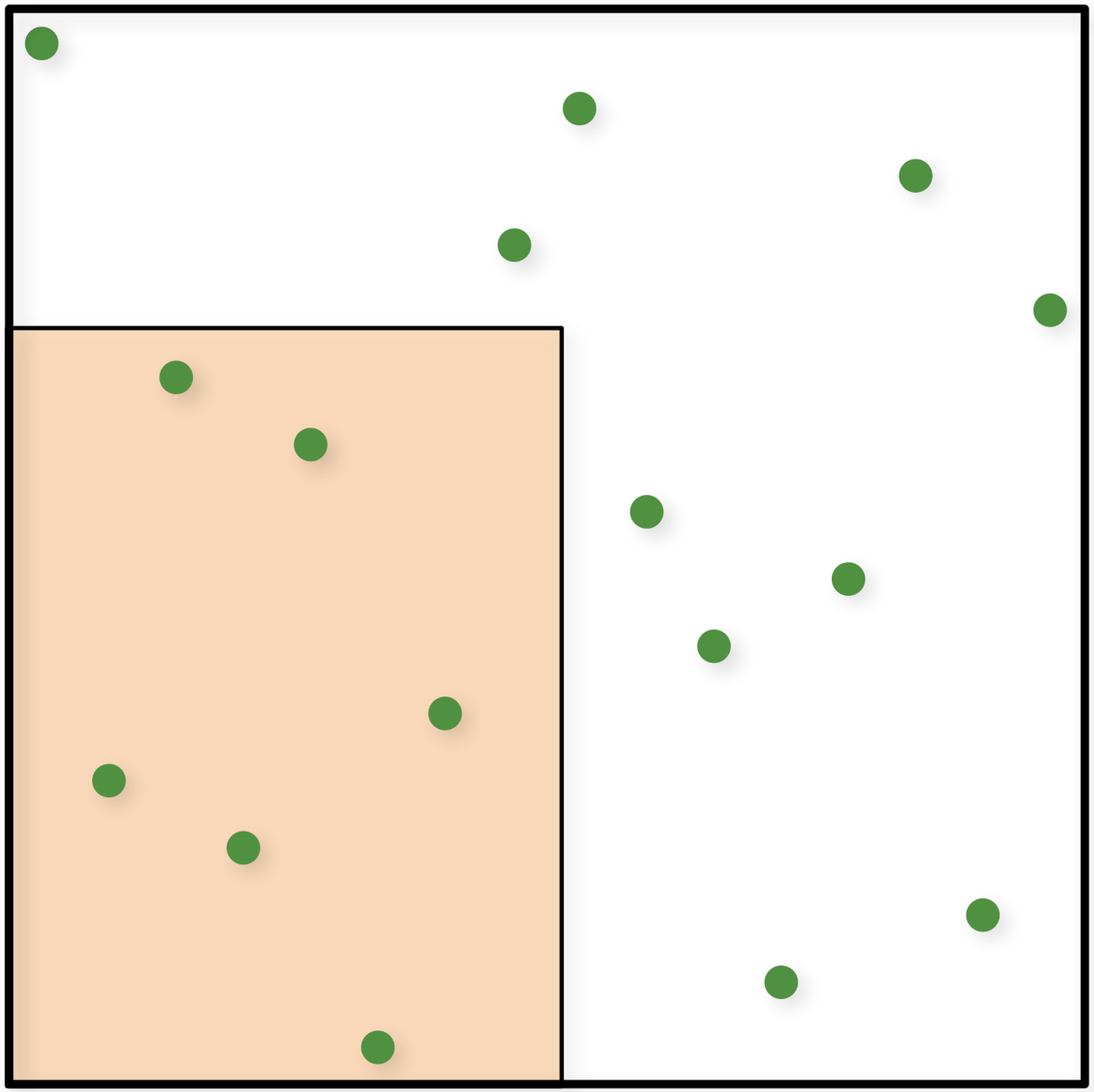
# Discrepancy



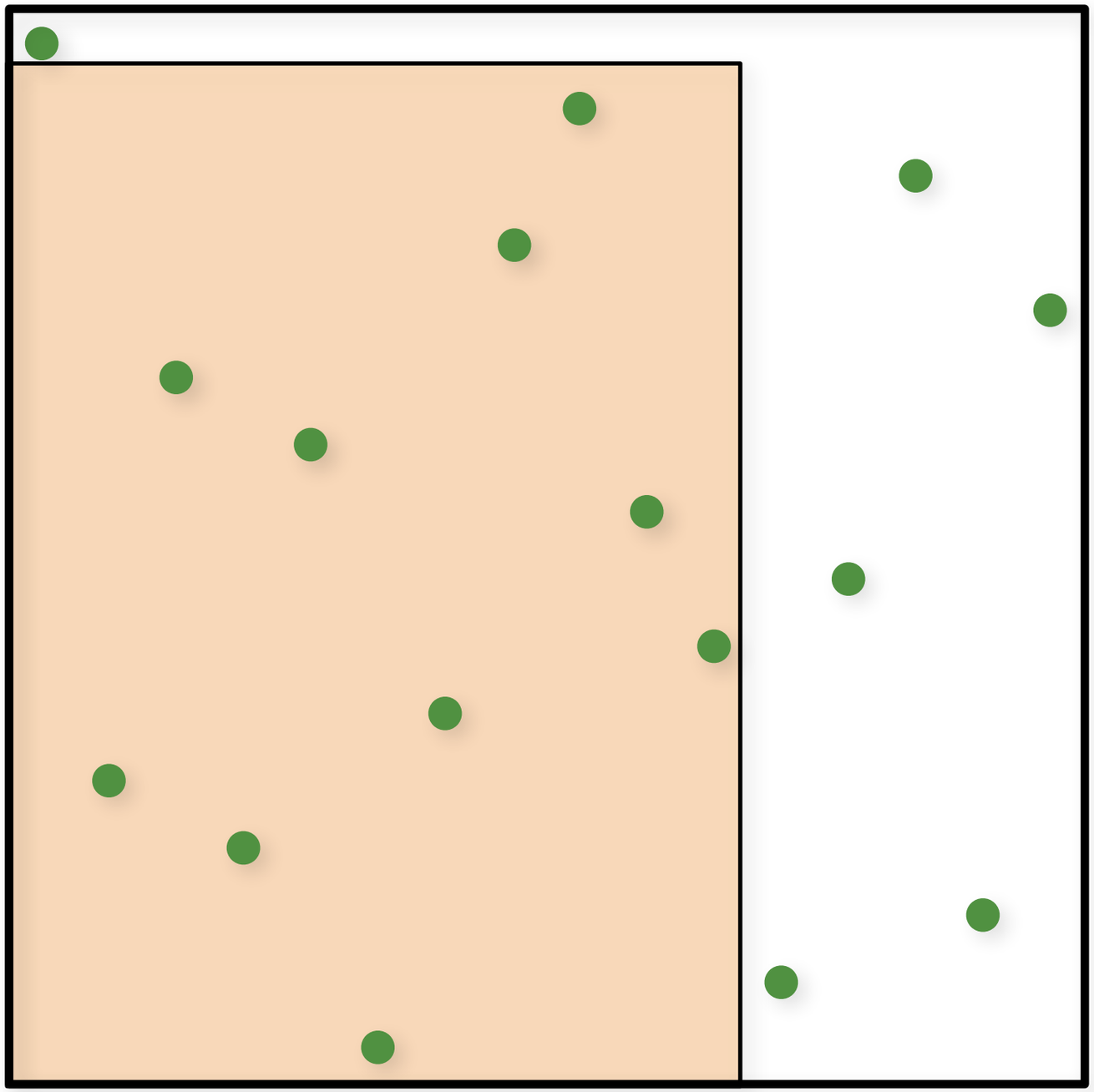
# Discrepancy



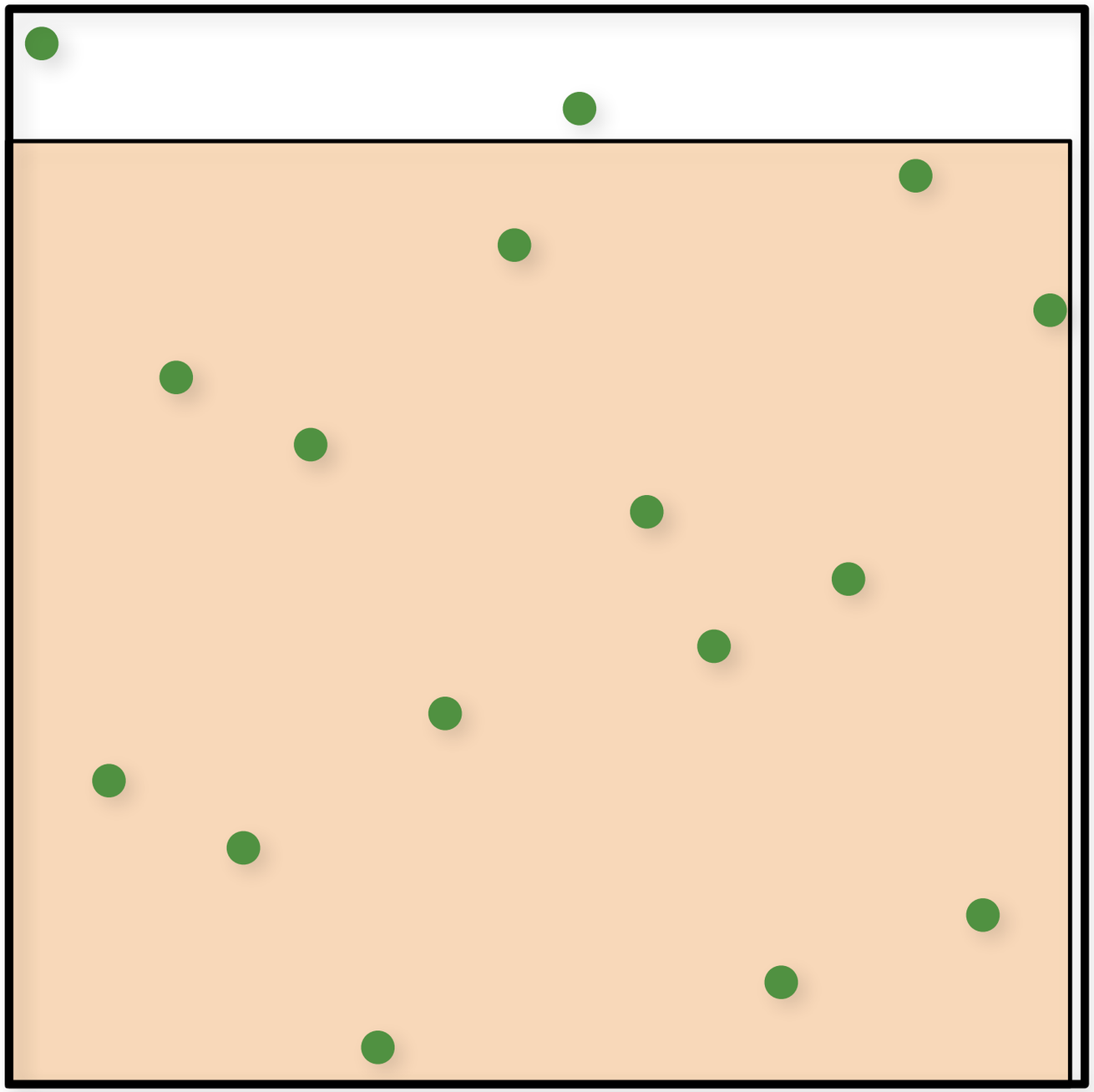
# Discrepancy



# Discrepancy



# Discrepancy



# Low-Discrepancy Sampling

- Low-Discrepancy sequences are specially crafted to have small discrepancy values.
- Entire field of study called Quasi-Monte Carlo (QMC)

# Koksma-Hlawka inequality

$$\left| \frac{1}{n} \sum_{i=1}^{n} f(x_i) - \int f(u)\,\mathrm{d}u \right| \leq V(f) D^*(x_1, \ldots, x_n)$$

# The Radical Inverse

- A positive integer value $n$ can be expressed in a base $b$ with a sequence of digits $d_m...d_2d_1$ uniquely determined by:

$$n = \sum_{i=1}^{\infty} d_i b^{i-1}$$

- The radical inverse function $\Phi_b$ in base $b$ converts a nonnegative integer $n$ to a floating-point value in [0, 1) by reflecting these digits about the decimal point:

$$\Phi_b(n) = 0.d_1 d_2 \ldots d_m$$

- Subsequent points "fall into biggest holes"

# The Radical Inverse

```
float radicalInverse(int n, int base, float inv)
{
    float v = 0.0f;
    for (float p = inv; n != 0; p *= inv, n /= base)
        v += (n % base) * p;
    return v;
}

float radicalInverse(int n, int base)
{
    return radicalInverse(n, base, 1.0f / base);
}
```

More efficient version available for base 2

# The Van der Corput Sequence

- Radical Inverse in base two

| n | Base 2 | $\Phi_b$ |
|---|---|---|
| 1 | 1 | .1 = 1/2 |
| 2 | 10 | .01 = 1/4 |
| 3 | 11 | .11 = 3/4 |
| 4 | 100 | .001 = 1/8 |
| 5 | 101 | .101 = 5/8 |
| 6 | 110 | .011 = 3/8 |
| 7 | 111 | .111 = 7/8 |
| ... | | |

Discrepancy:

$$D^*(x_1,\ldots,x_n) \in O\left(\frac{\log N}{N}\right)$$

# The Radical Inverse (Base 2)

```
float vanDerCorputRIU(uint n)
{
    n = (n << 16) | (n >> 16);
    n = ((n & 0x00ff00ff) << 8) | ((n & 0xff00ff00) >> 8);
    n = ((n & 0x0f0f0f0f) << 4) | ((n & 0xf0f0f0f0) >> 4);
    n = ((n & 0x33333333) << 2) | ((n & 0xcccccccc) >> 2);
    n = ((n & 0x55555555) << 1) | ((n & 0xaaaaaaaa) >> 1);
    return n / float (0x100000000LL);
}
```

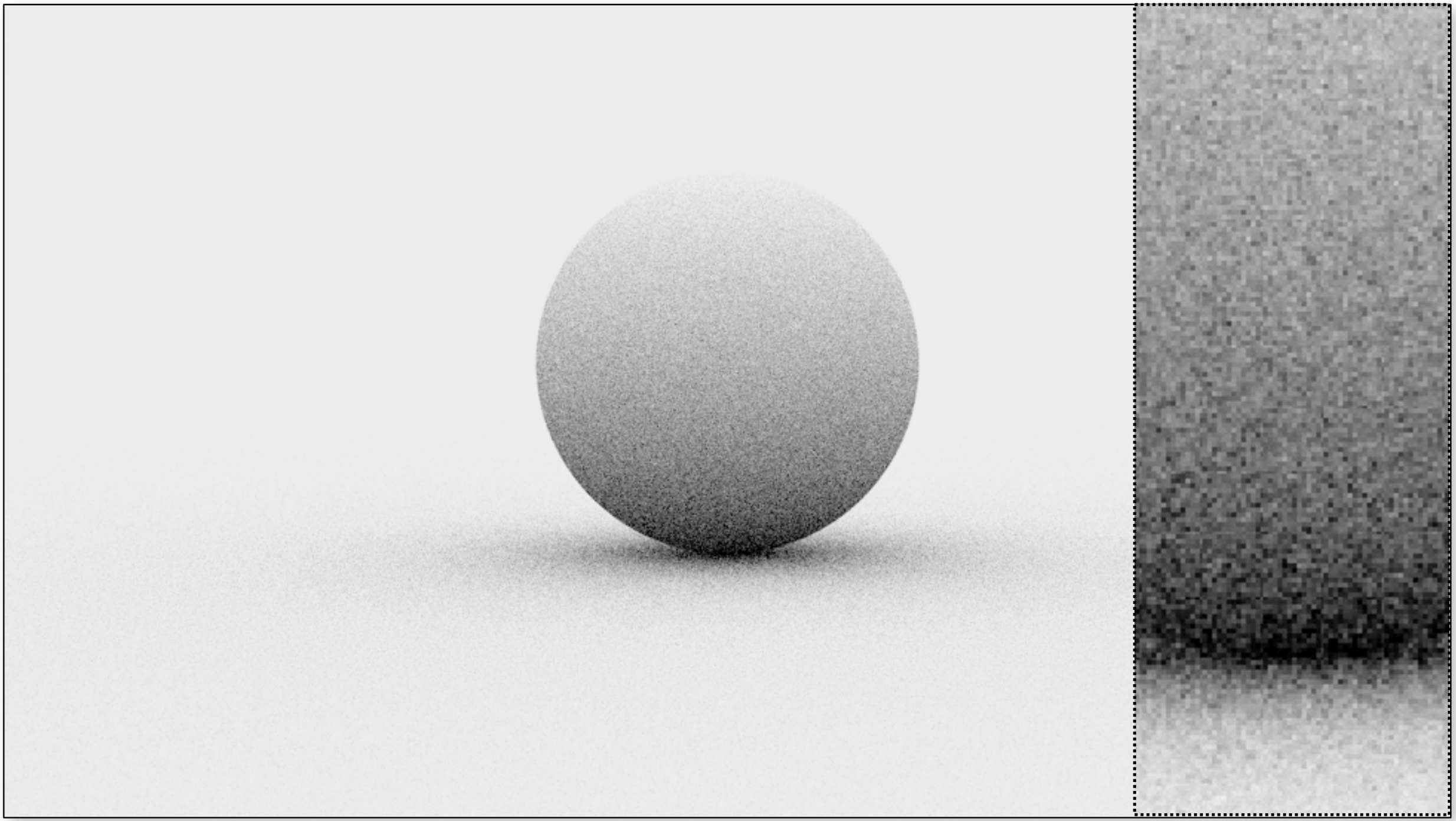
# The Halton Sequence

- An n-dimensional Halton sequence uses the radical inverse with a different base $b$ for each dimension of the pattern.
- The bases must all be relatively prime.

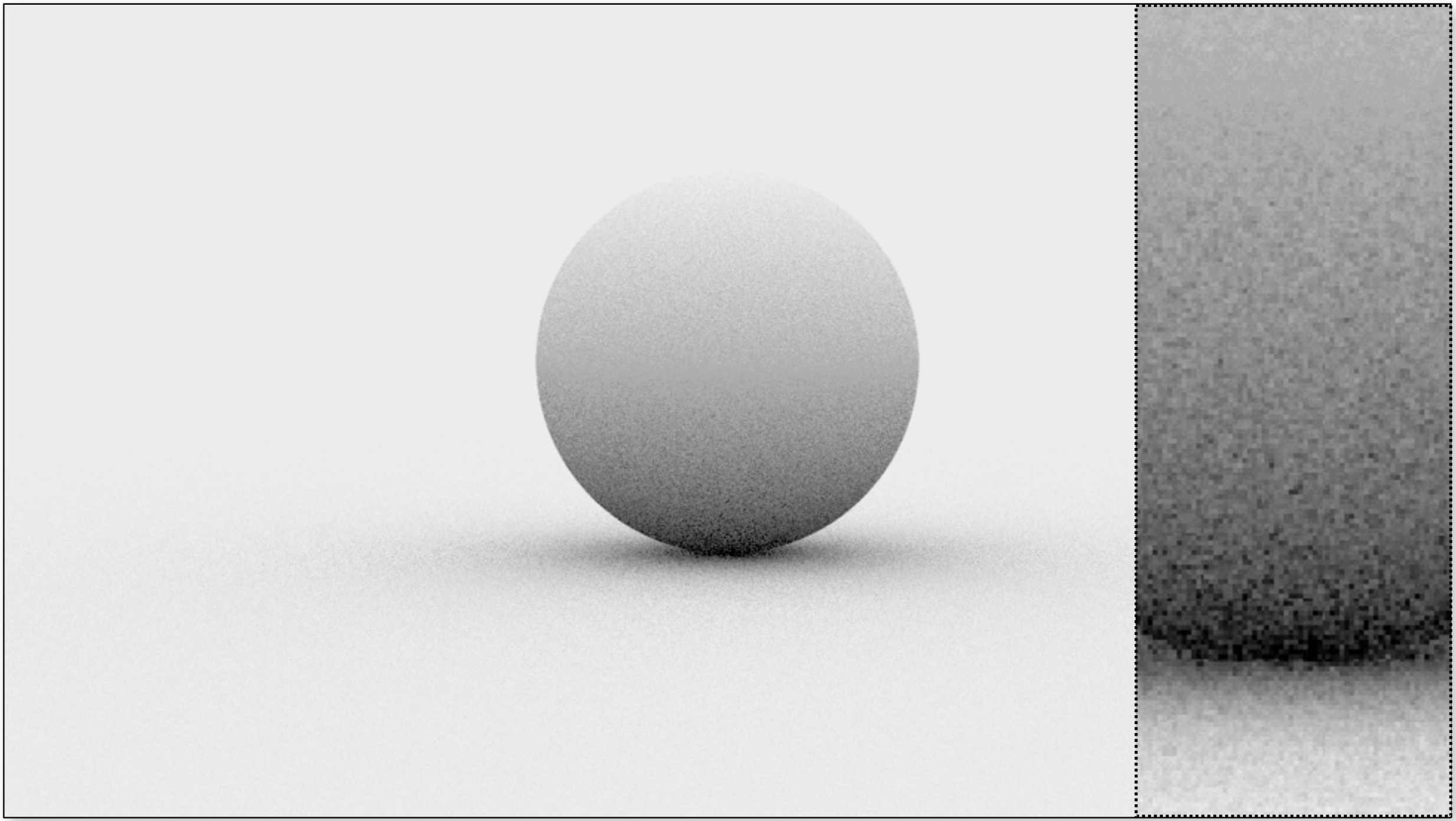
$$x_i = (\Phi_2(i), \Phi_3(i), \Phi_5(i), \ldots, \Phi_{p_n}(i))$$

- Incremental/progressive generation of samples

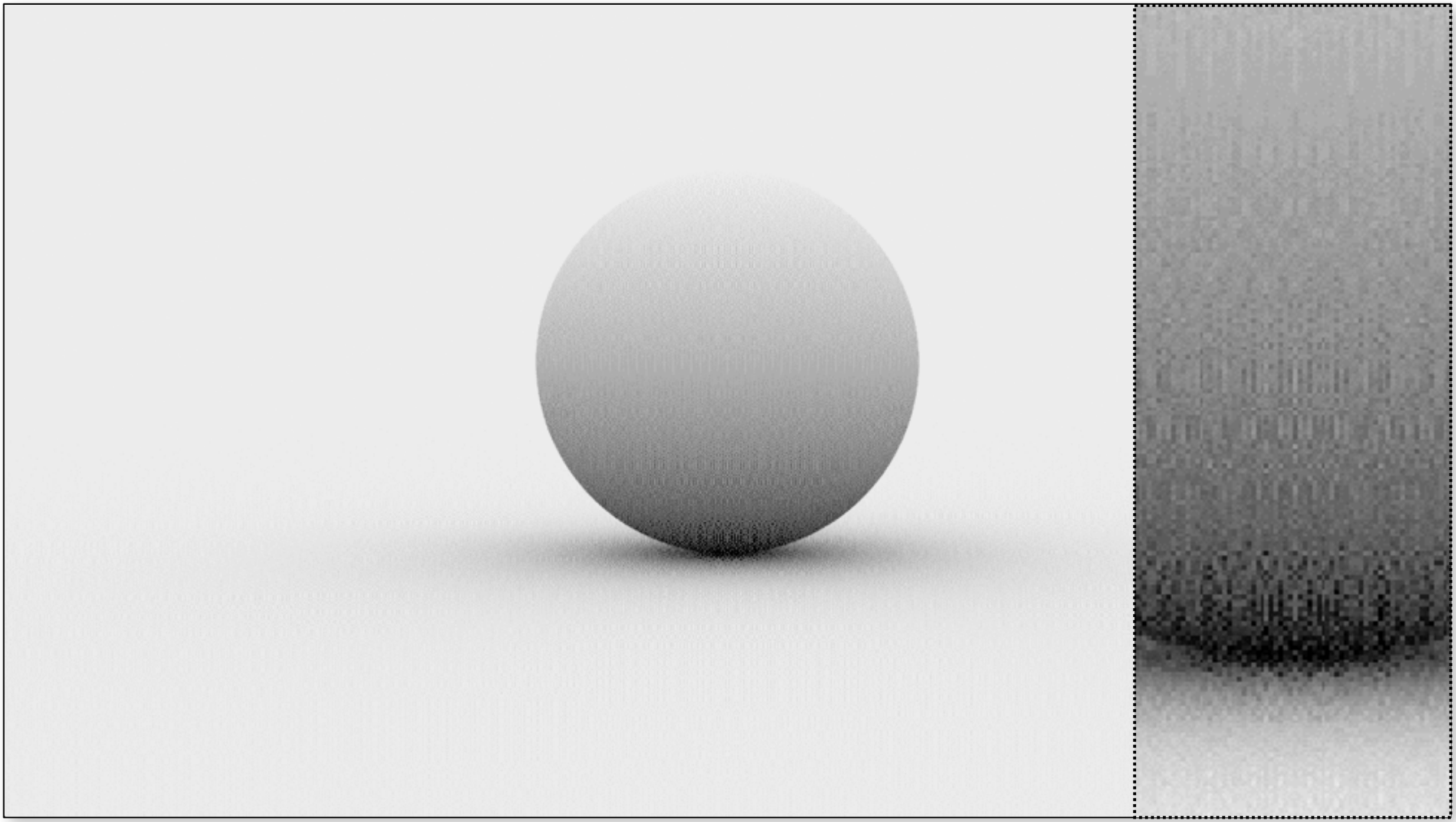
# Monte Carlo (16 random samples)



# Monte Carlo (16 stratified samples)



# Quasi-Monte Carlo (16 Halton samples)

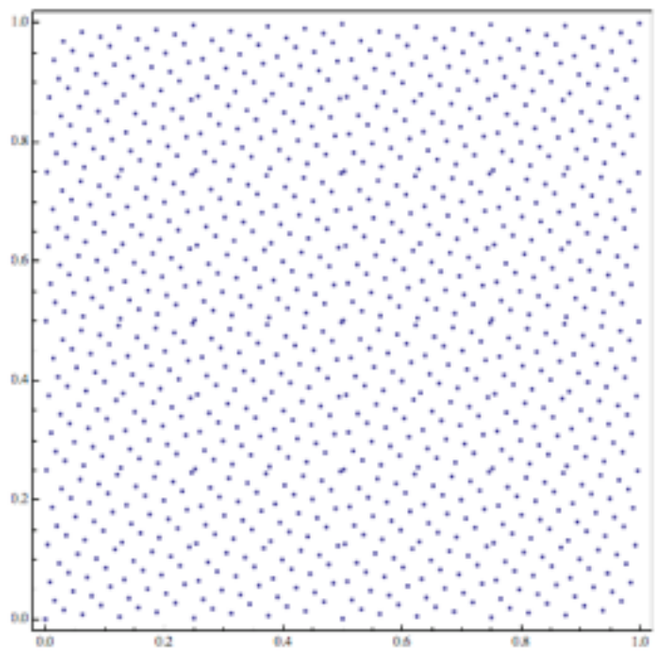


# The Hammersley Sequence

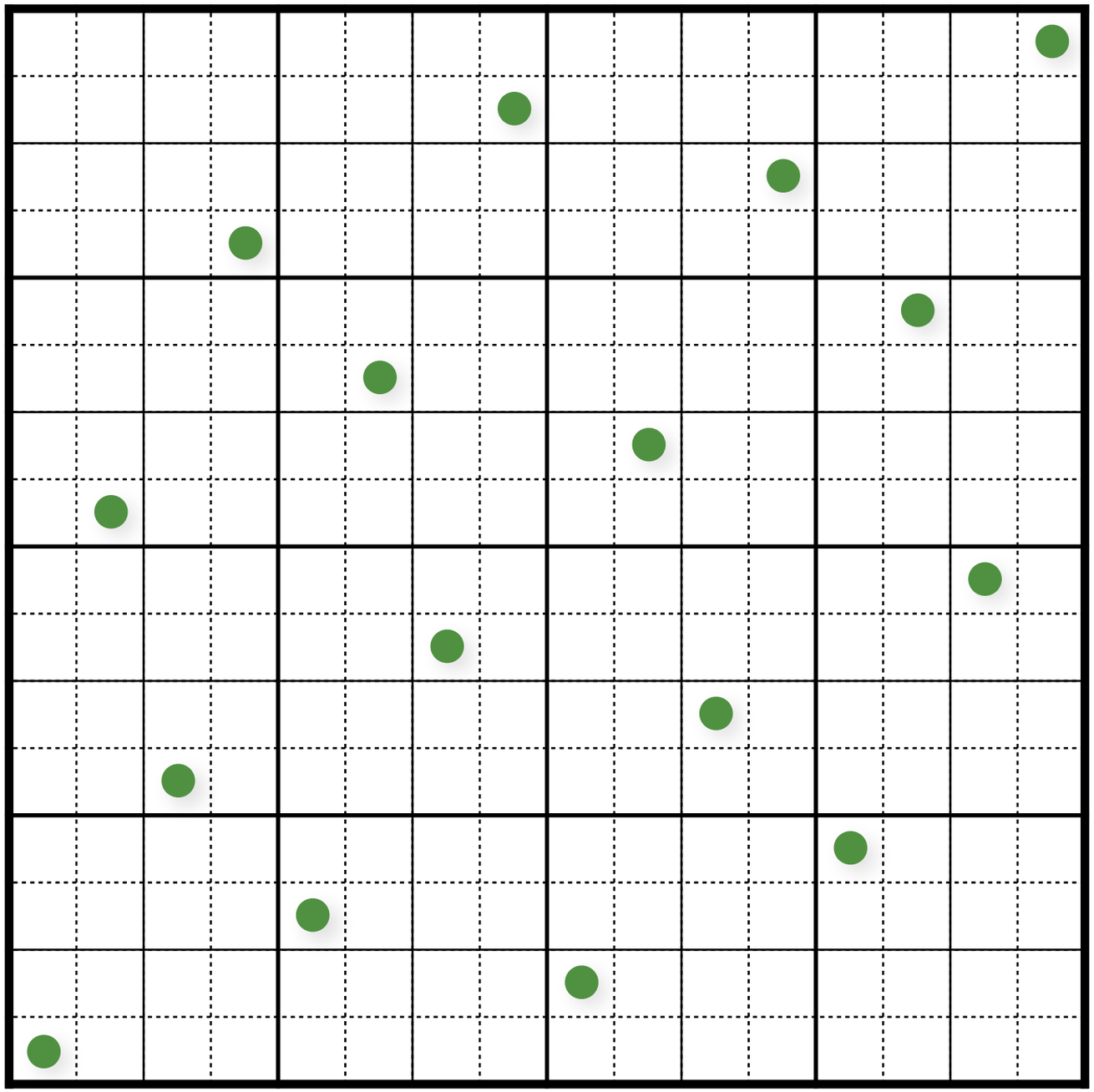
- Same as Halton, but uses $i/N$ for first dimension:

$$x_i = \left( \frac{i}{N}, \Phi_2(i), \Phi_3(i), \ldots, \Phi_{p_n}(i) \right)$$

- Provides slightly lower discrepancy
- Not incremental, need to know total number of samples, $N$, in advance

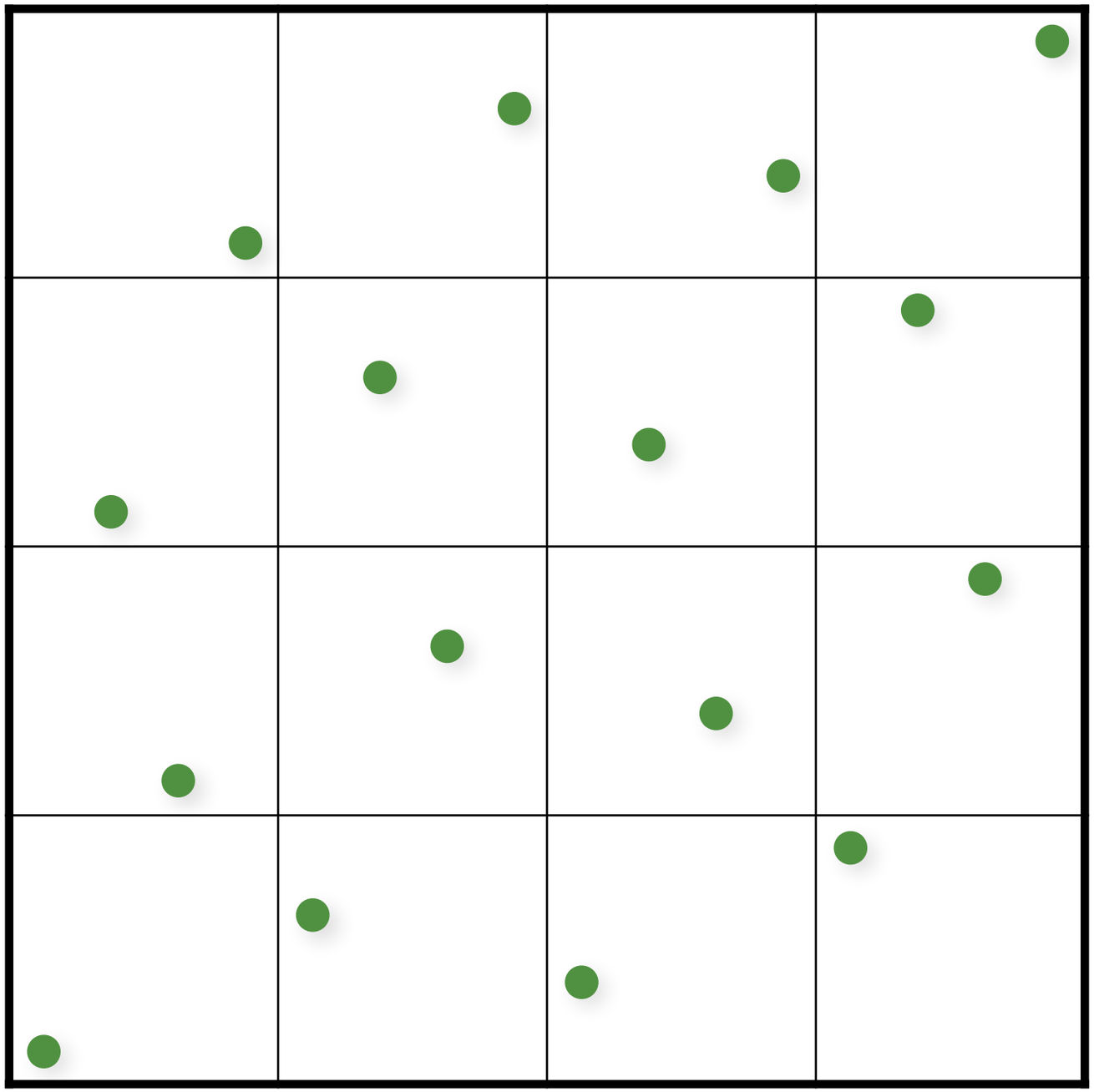


# The Hammersley Sequence



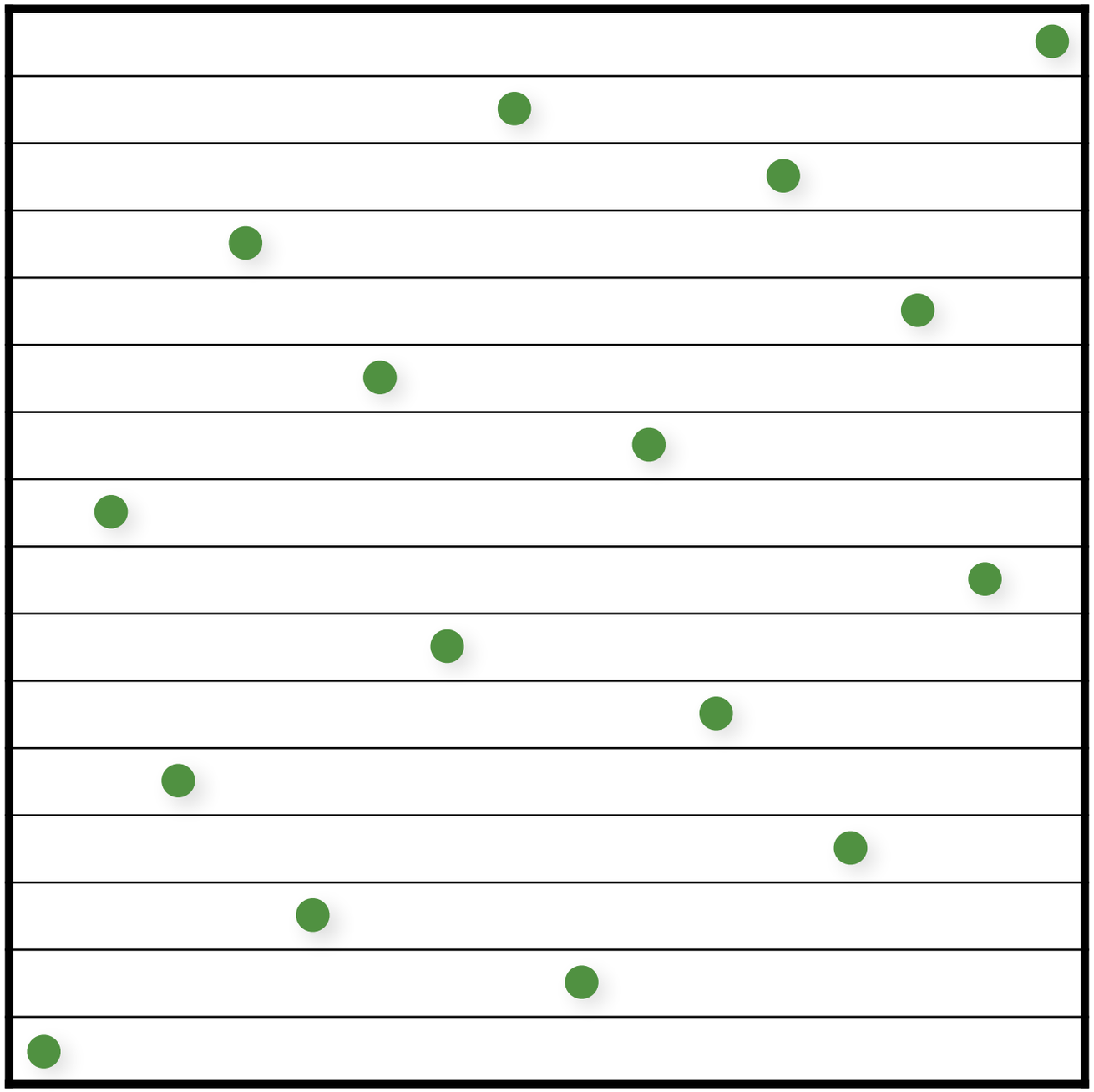
1 sample in each “elementary interval”

# The Hammersley Sequence



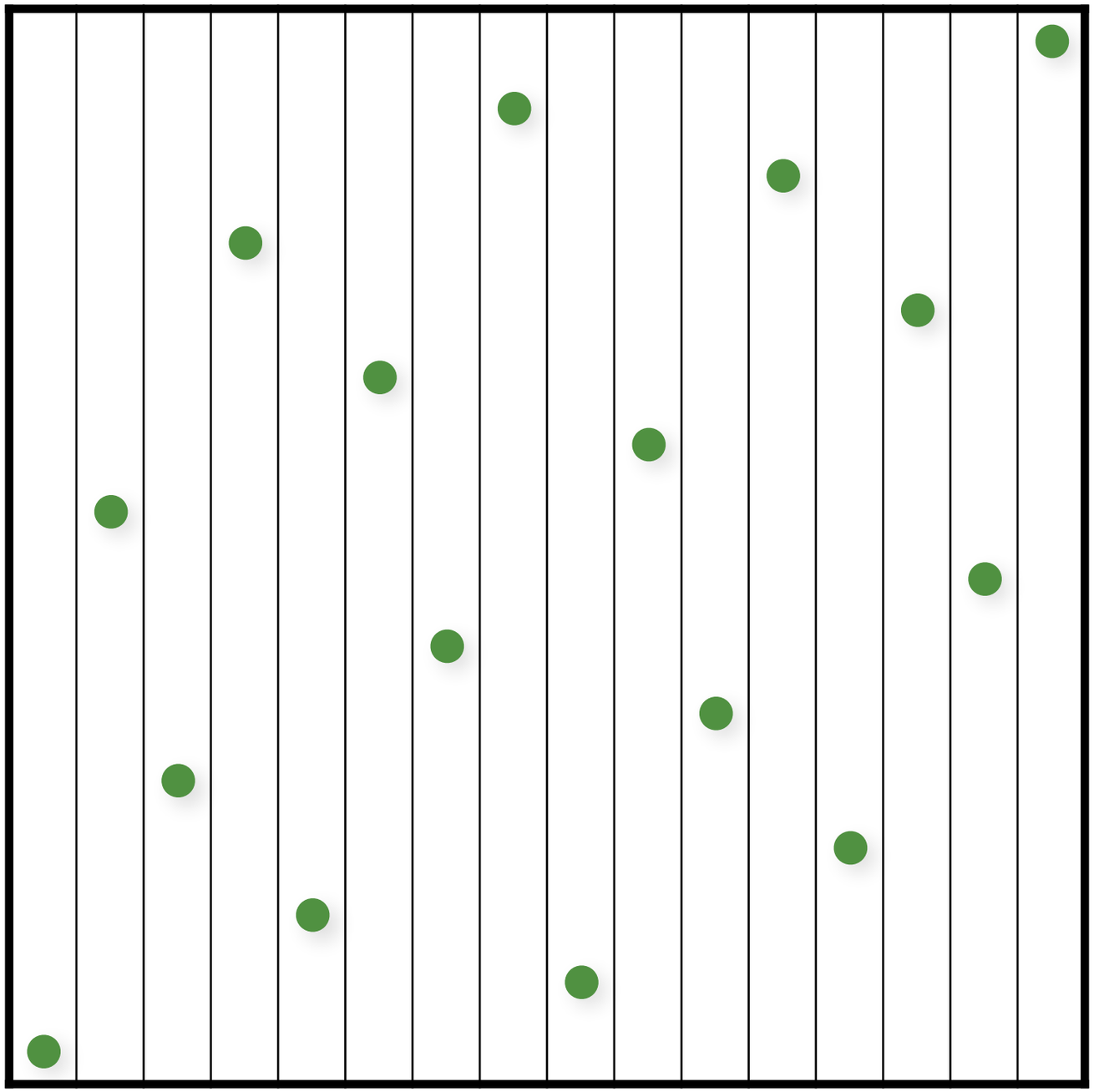
1 sample in each “elementary interval”

# The Hammersley Sequence



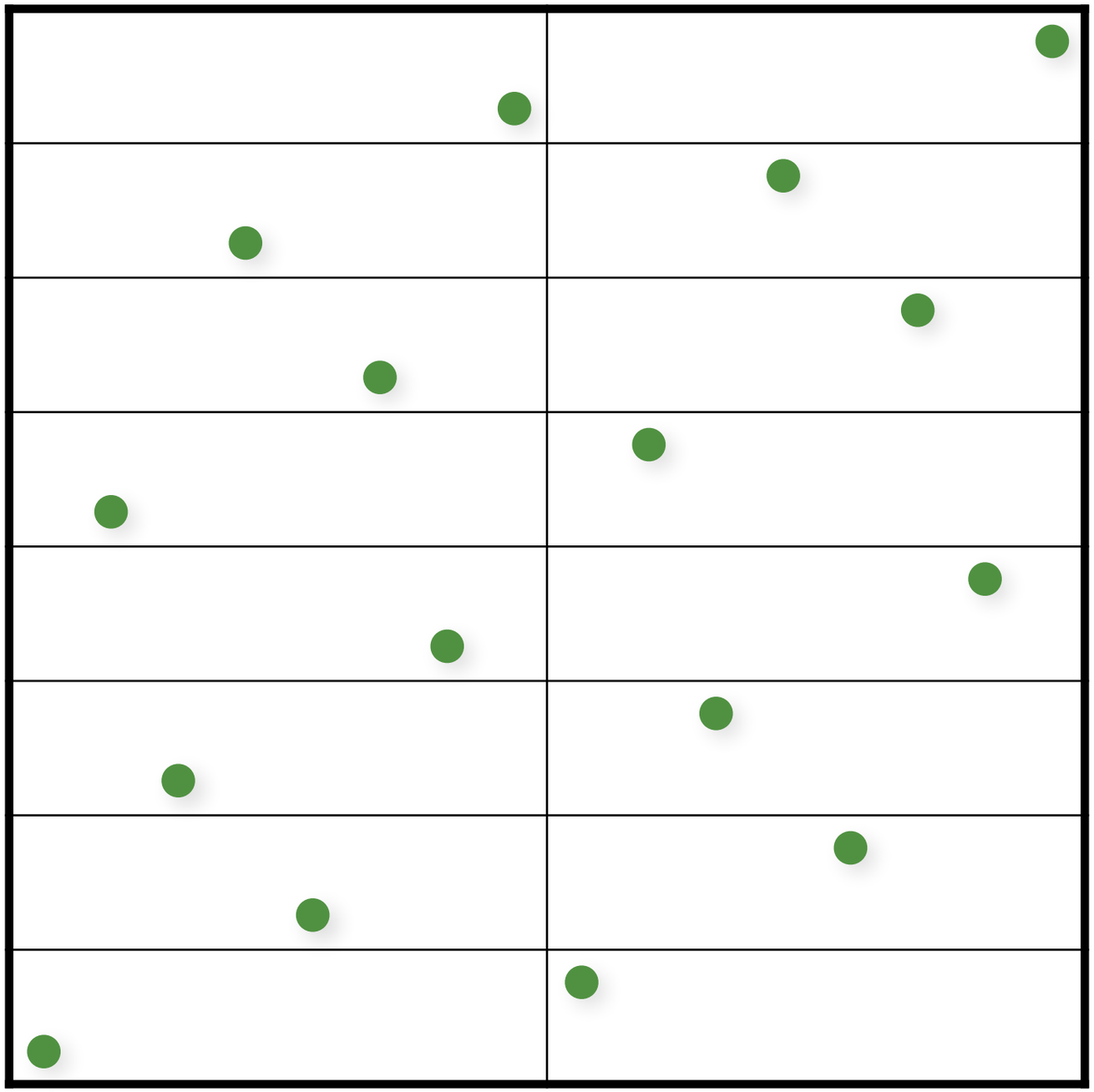
1 sample in each “elementary interval”

# The Hammersley Sequence



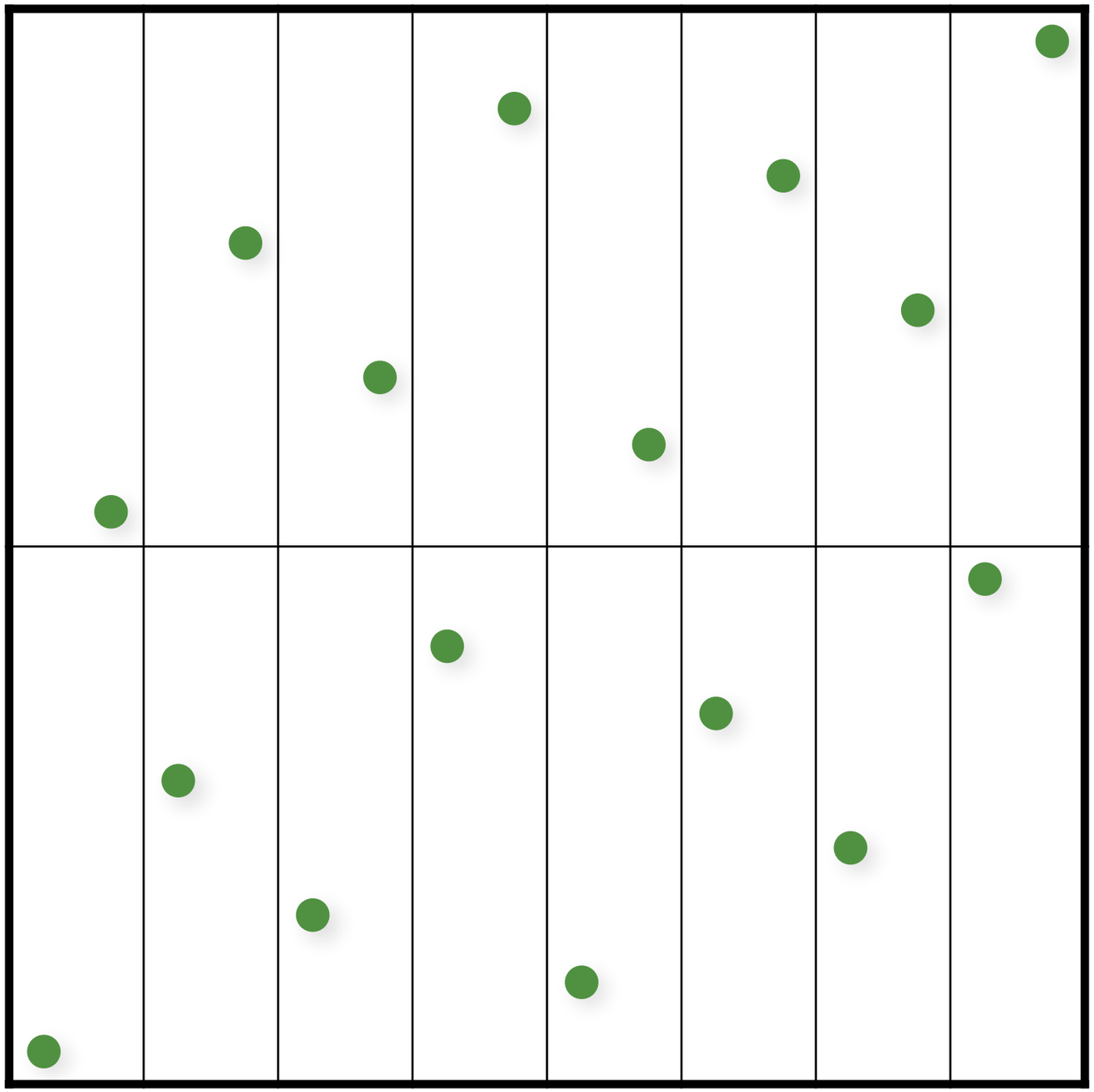
1 sample in each “elementary interval”

# The Hammersley Sequence



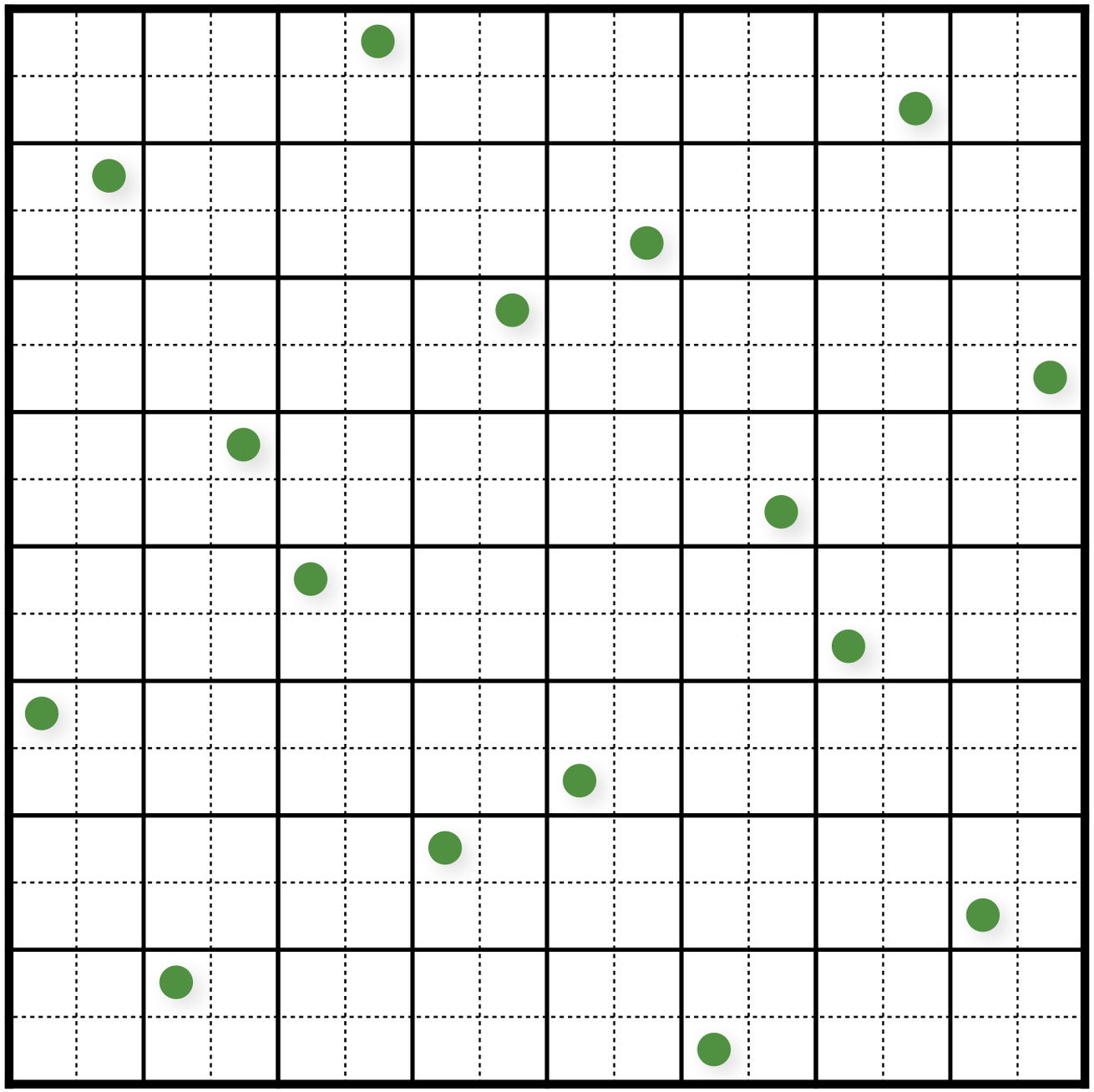
1 sample in each “elementary interval”

# The Hammersley Sequence



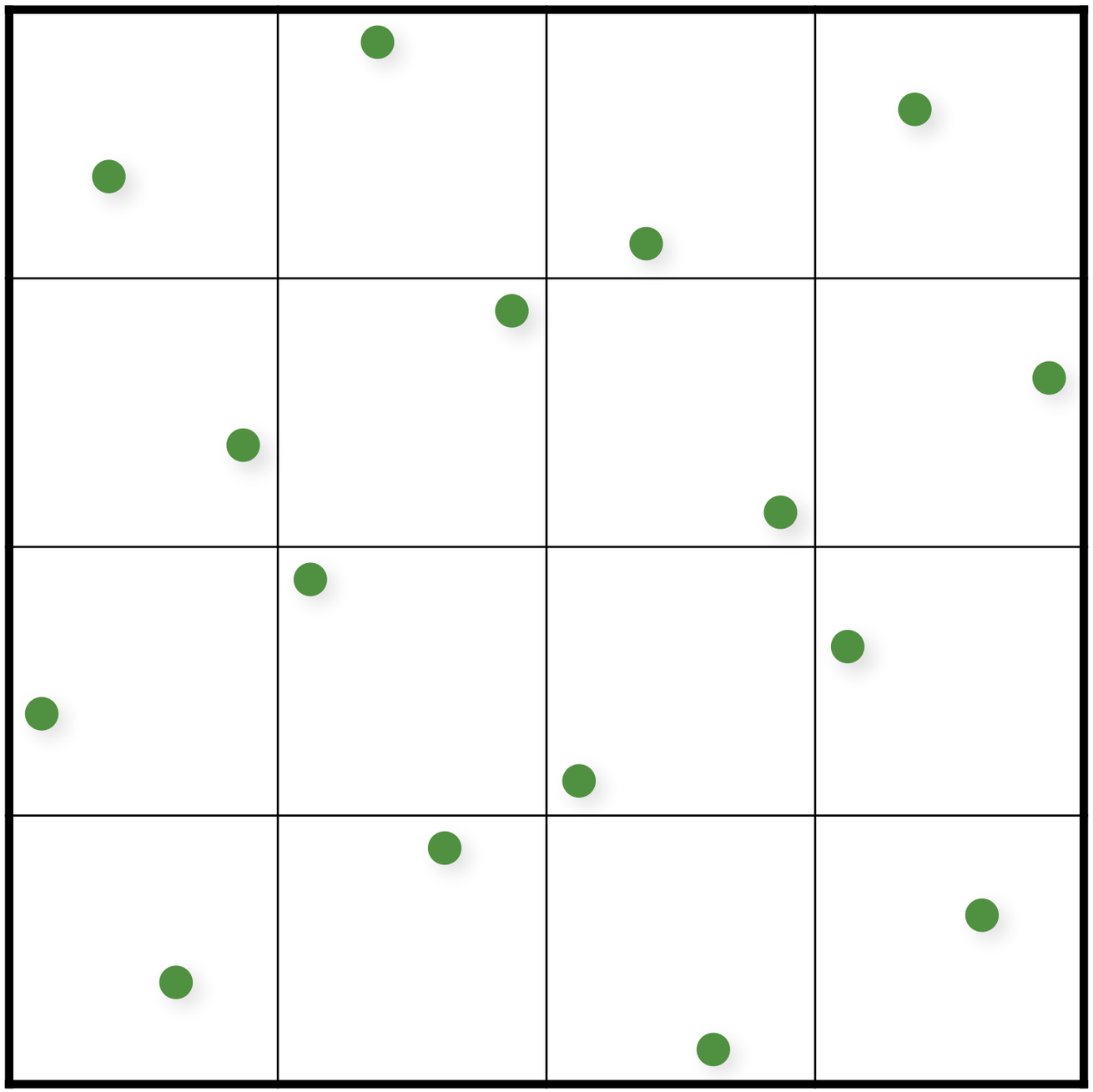
1 sample in each “elementary interval”

# (0,2)-Sequences



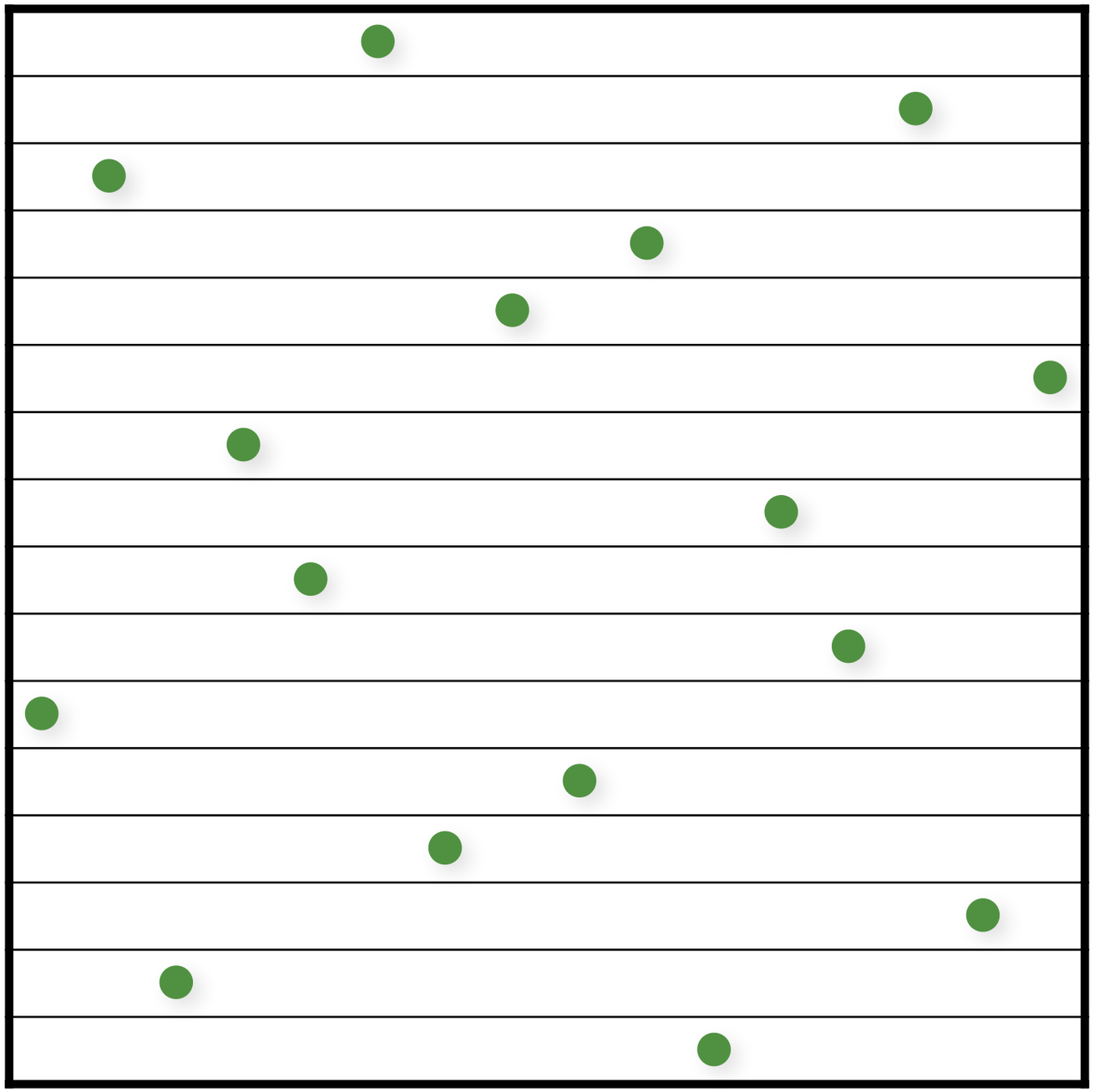
1 sample in each “elementary interval”

# (0,2)-Sequences



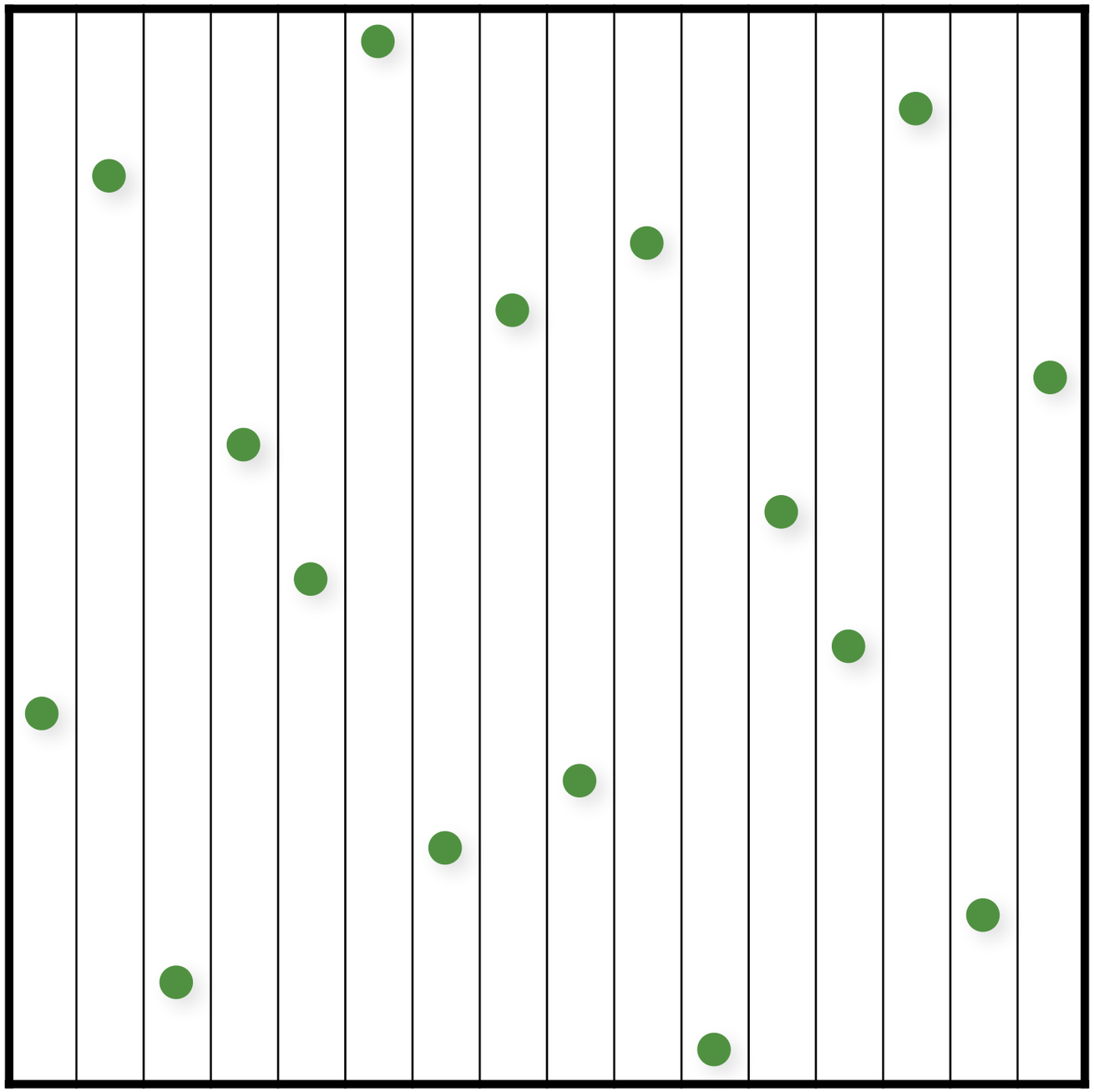
1 sample in each “elementary interval”

# (0,2)-Sequences



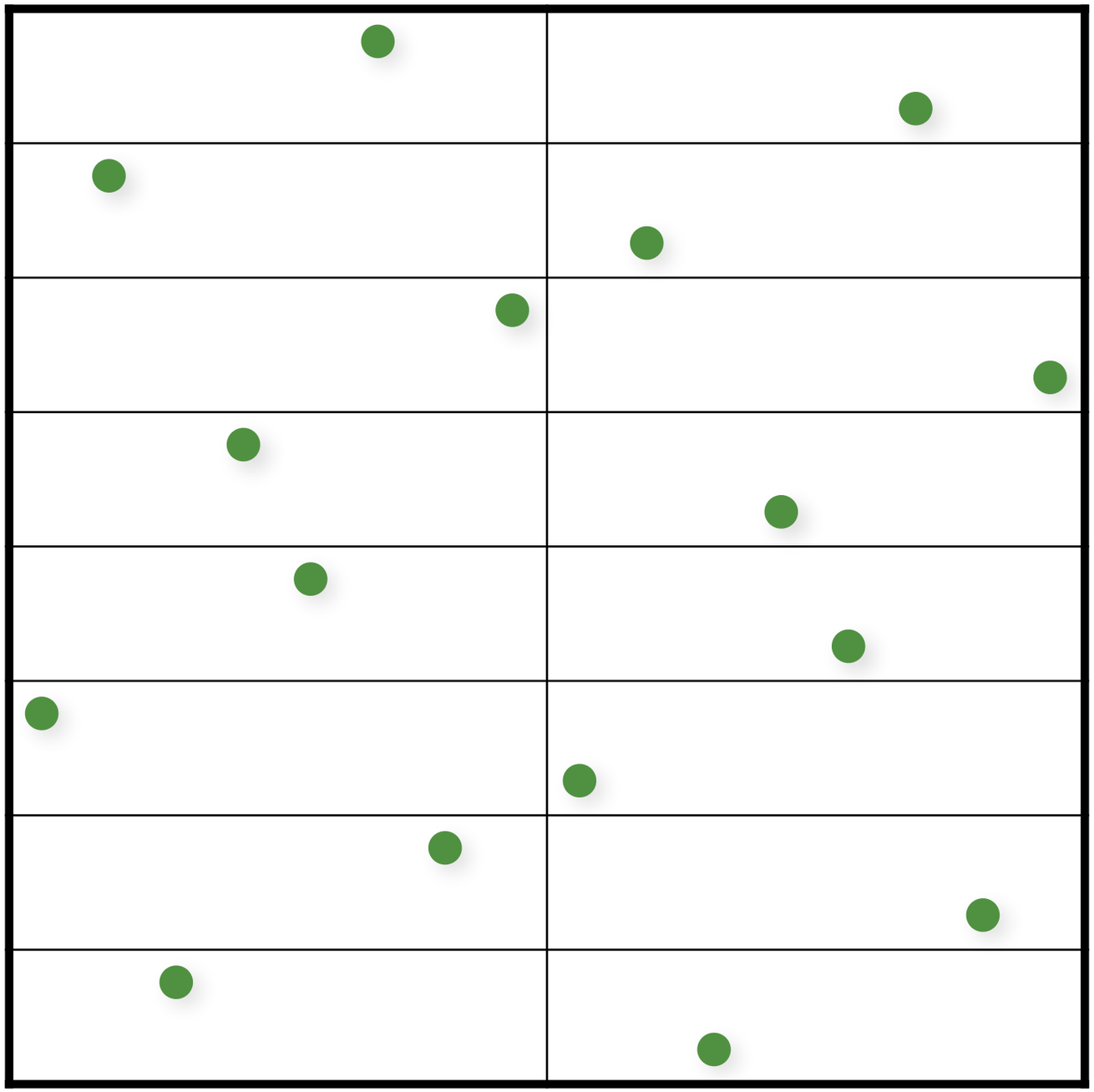
1 sample in each “elementary interval”

# (0,2)-Sequences



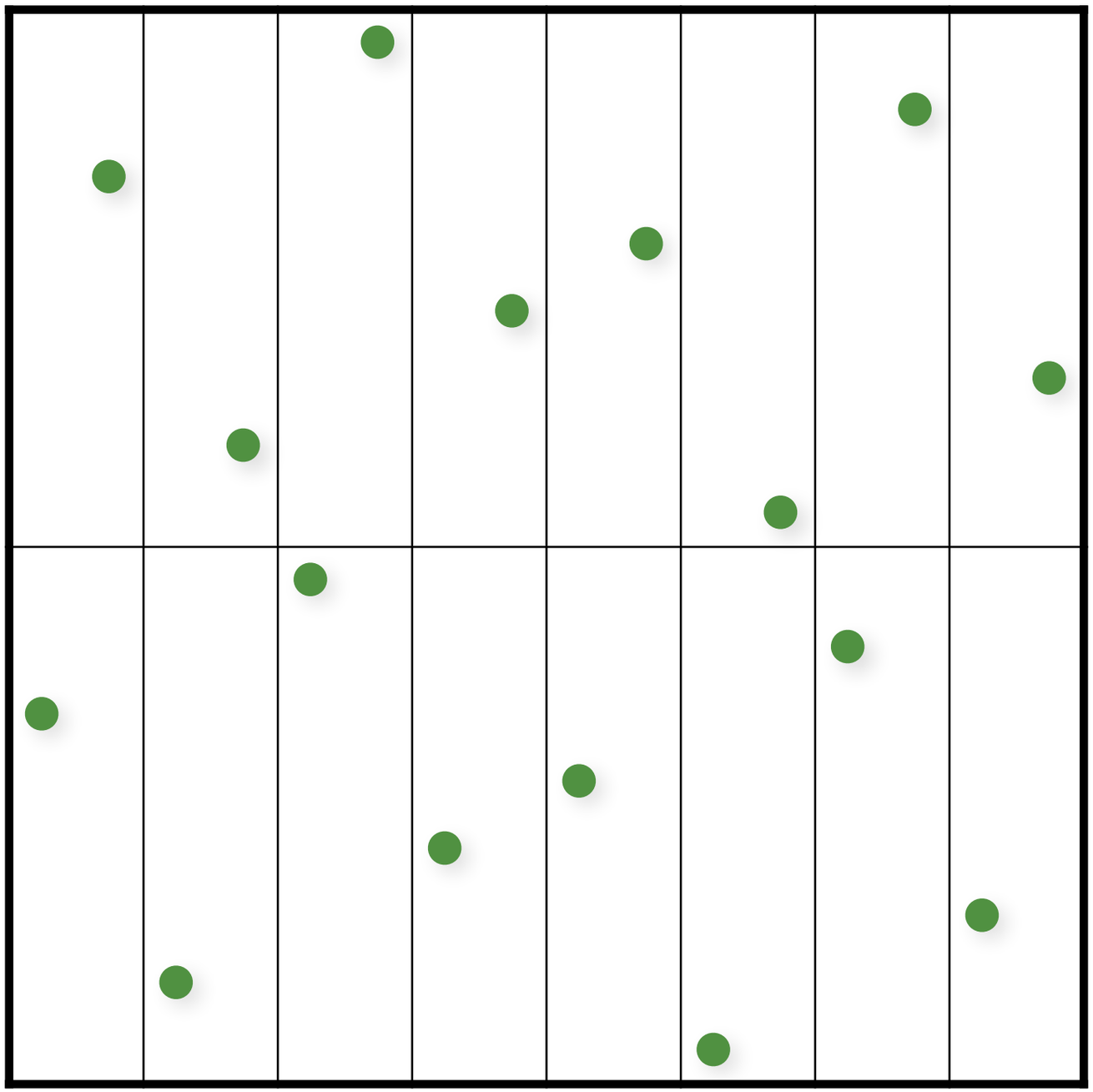
1 sample in each “elementary interval”

# (0,2)-Sequences



1 sample in each “elementary interval”

# (0,2)-Sequences



1 sample in each “elementary interval”

# Additional resources

"Enumerating Quasi-Monte Carlo Point Sequences in Elementary Intervals."
L. Grünschloß, M. Raab and A. Keller. In *Monte Carlo and Quasi-Monte Carlo Methods 2010*

http://gruenschloss.org/

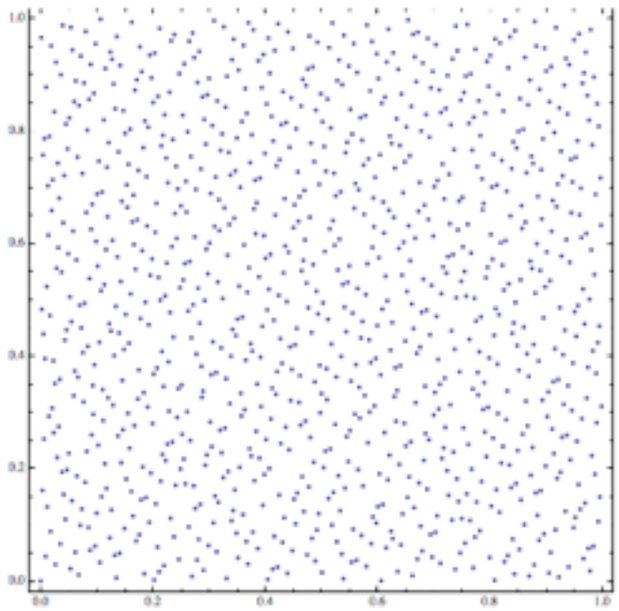
# Many more...

- Sobol
- Faure
- Larcher-Pillichshammer
- Folded Radical Inverse
- (t,s)-sequences & (t,m,s)-nets
- Scrambling/randomization
- much more...

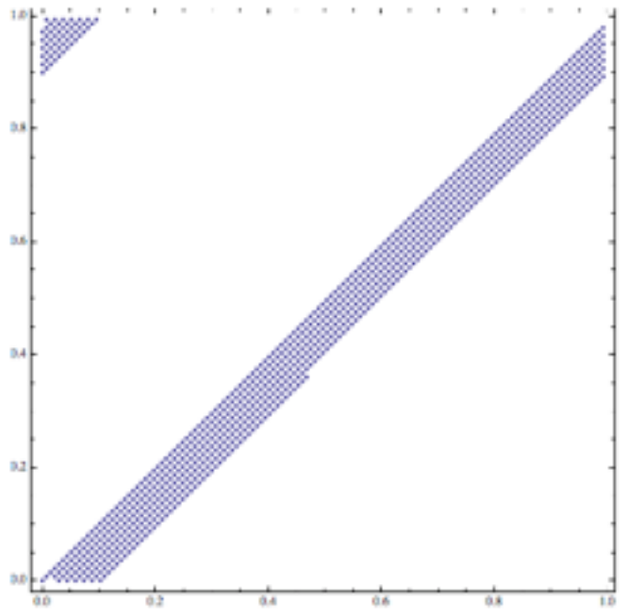
# Challenges

- LD sequence identical for multiple runs
  - cannot average independent images!
  - no “random” seed
- Quality decreases in higher dimensions

## Halton Sequence



Dimensions 1 and 2



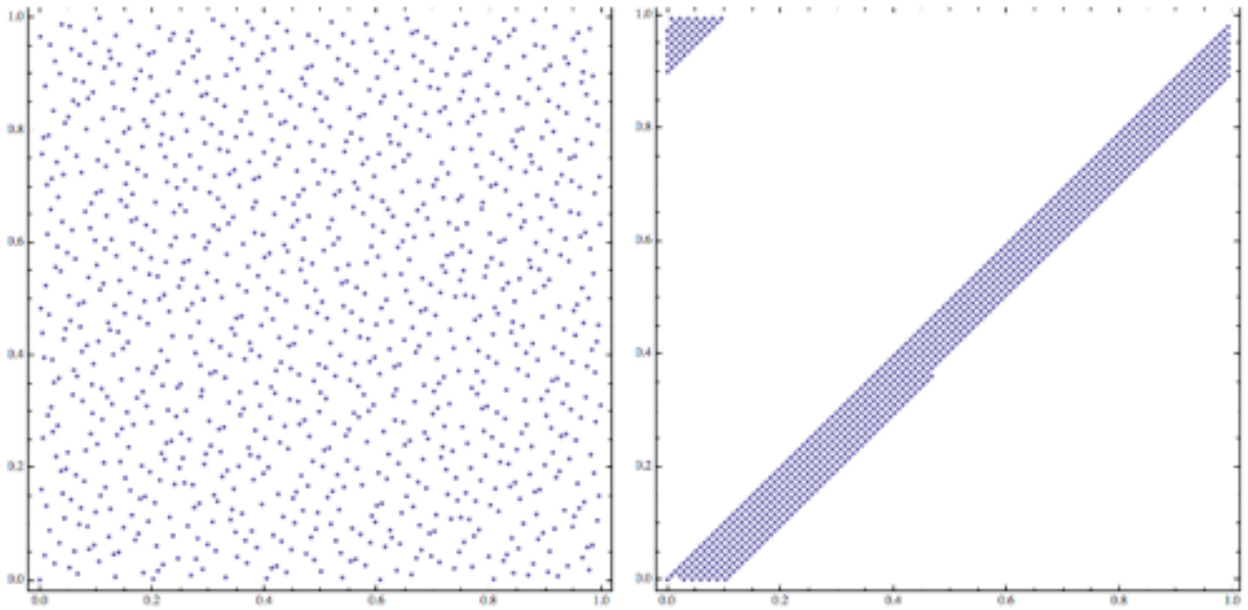
Dimensions 32 and 33

# Randomized/Scrambled Sequences

- Random permutations: compute a permutation table for the order of the digits and use it when computing the radical inverse

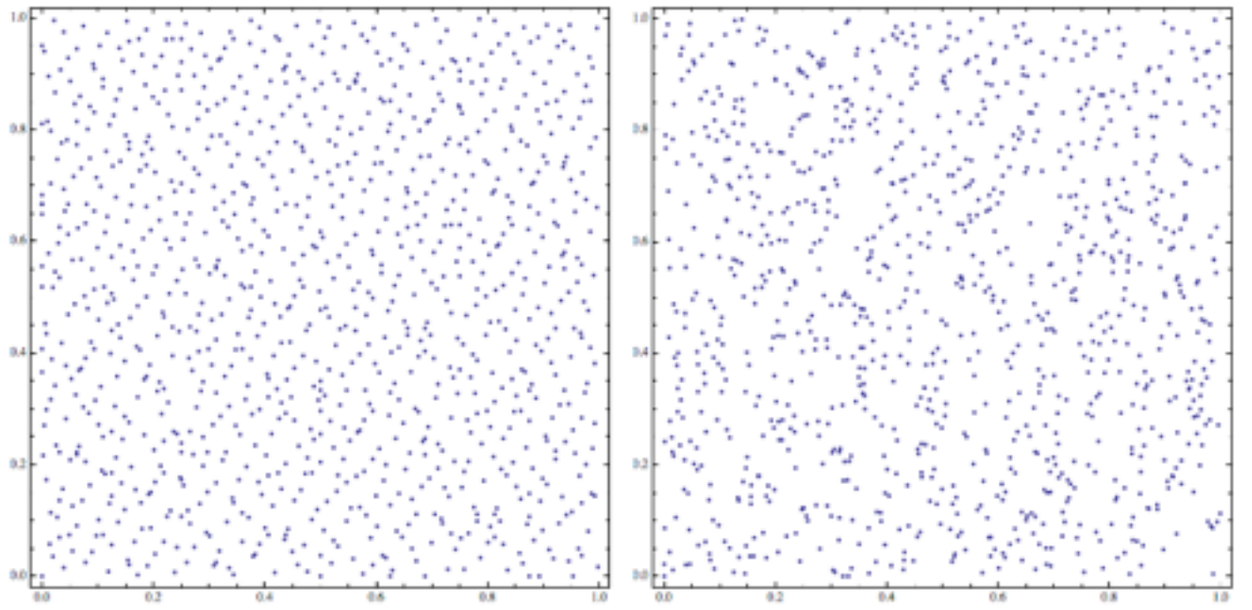
$$\Phi_b(n) = 0.\pi(d_1)\pi(d_2)...\pi(d_m)$$

Without scrambling



Dimensions 1 and 2 Dimensions 32 and 33

With scrambling



Dimensions 1 and 2 Dimensions 32 and 33

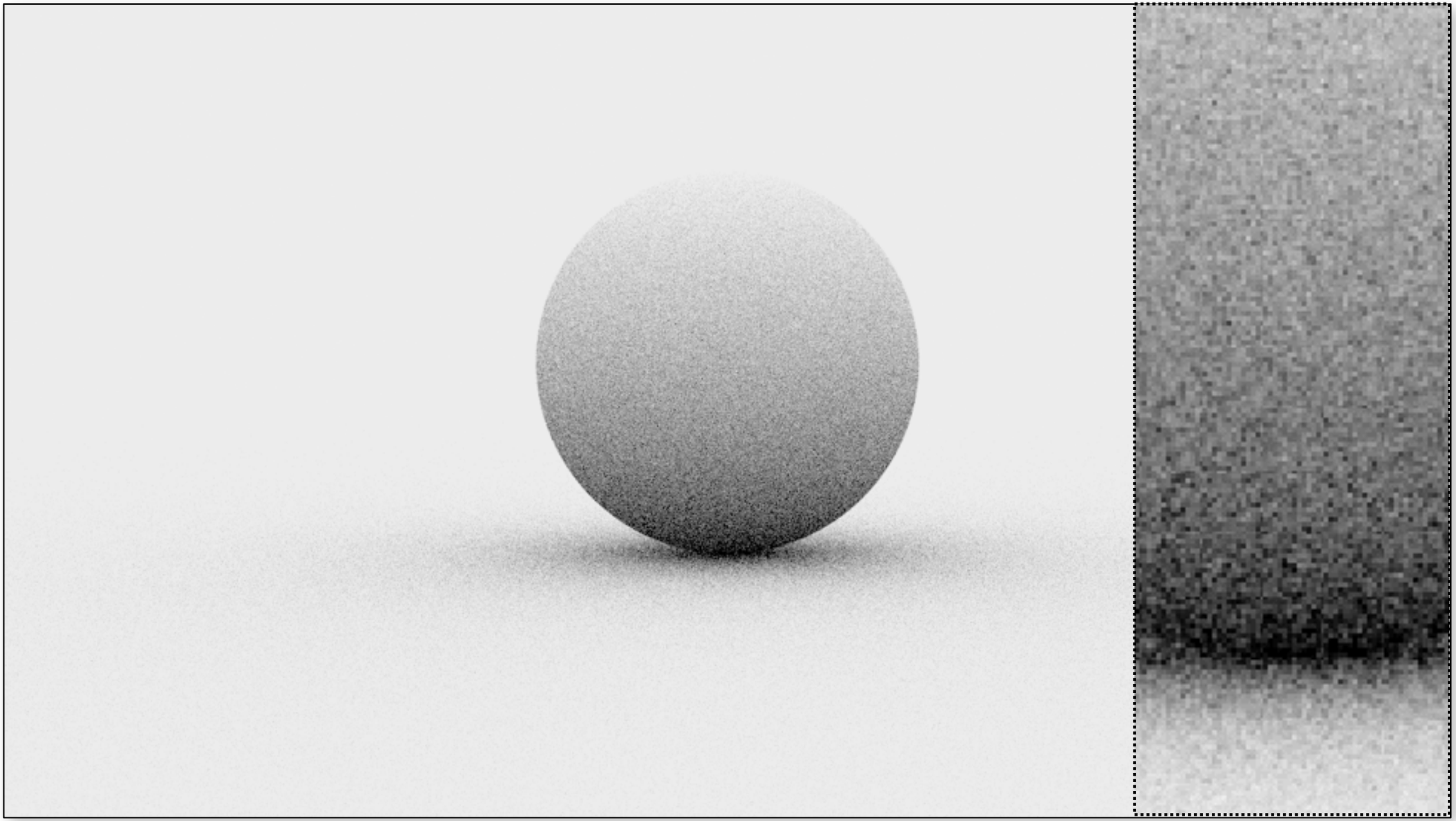
# Randomized/Scrambled Sequences

- Random permutations: compute a permutation table for the order of the digits and use it when computing the radical inverse
  - Can be done very efficiently for base 2 with XOR operation
- See PBRe2 Ch7 for details

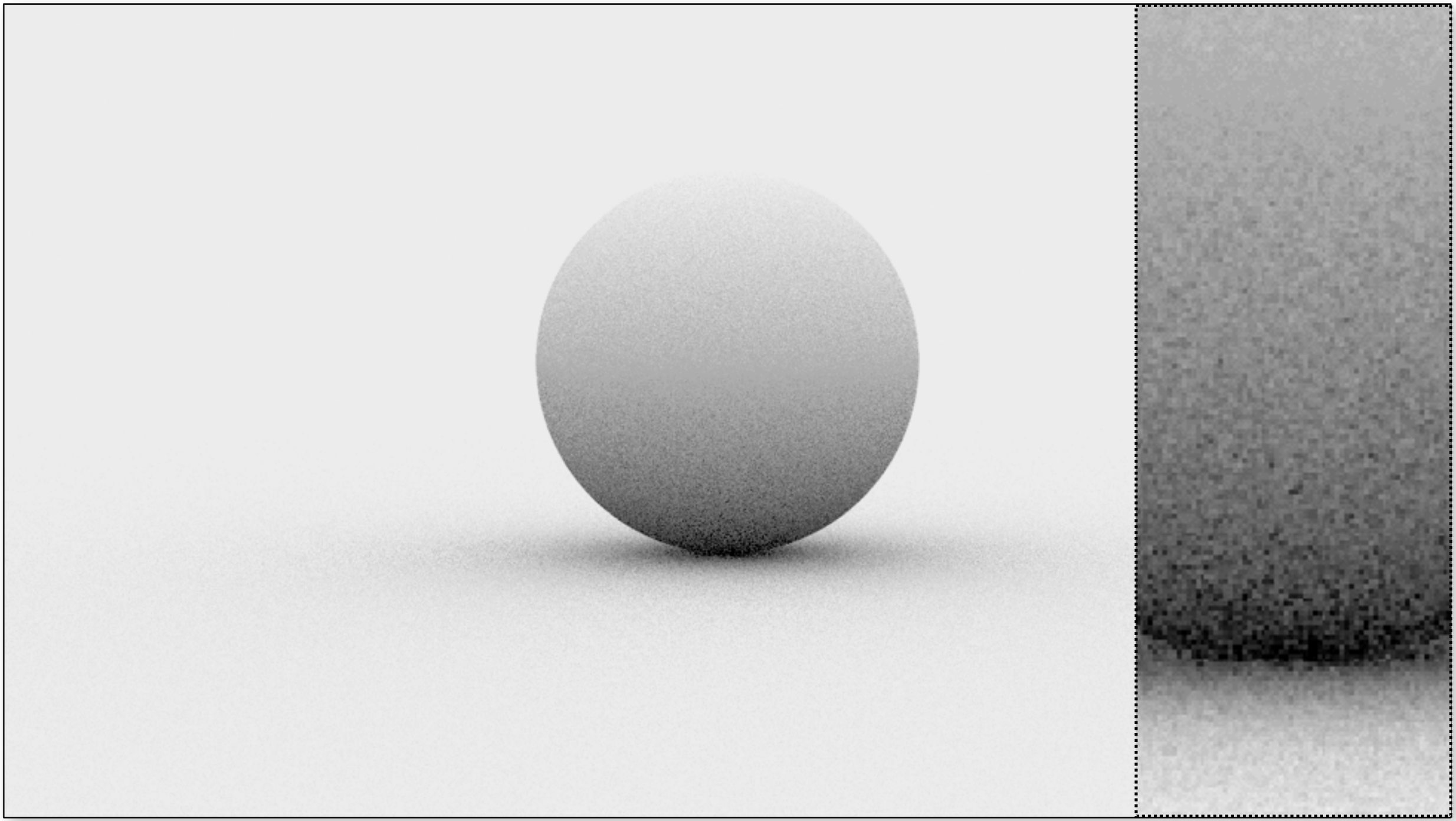
# Scrambled Radical Inverse (Base 2)

```
float vanDerCorputRIU(uint n, uint scramble = 0)
{
    n = (n << 16) | (n >> 16);
    n = ((n & 0x00ff00ff) << 8) | ((n & 0xff00ff00) >> 8);
    n = ((n & 0x0f0f0f0f) << 4) | ((n & 0xf0f0f0f0) >> 4);
    n = ((n & 0x33333333) << 2) | ((n & 0xcccccccc) >> 2);
    n = ((n & 0x55555555) << 1) | ((n & 0xaaaaaaaa) >> 1);
    n ^= scramble;
    return n / float (0x100000000LL);
}
```

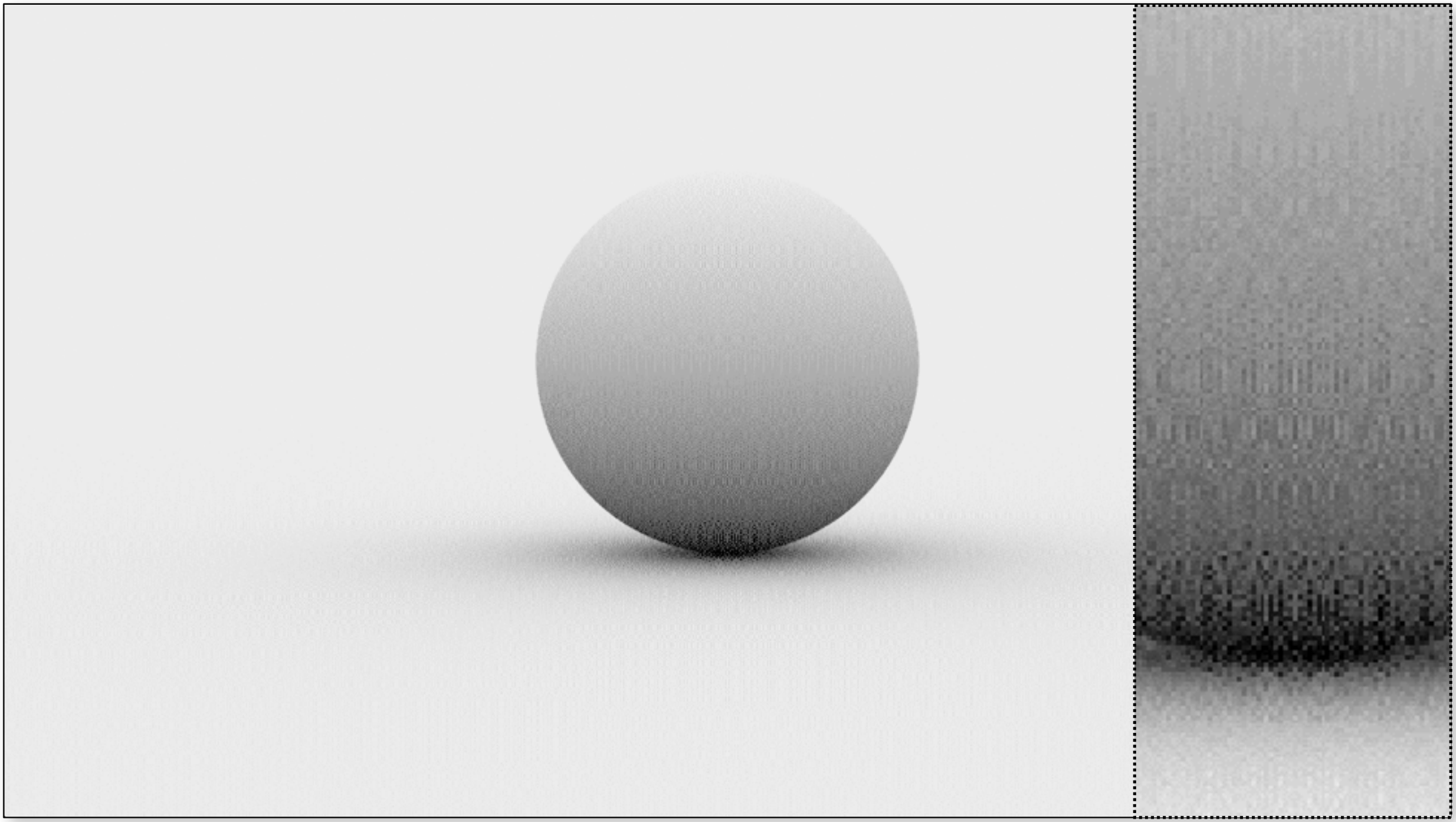
# Monte Carlo (16 random samples)



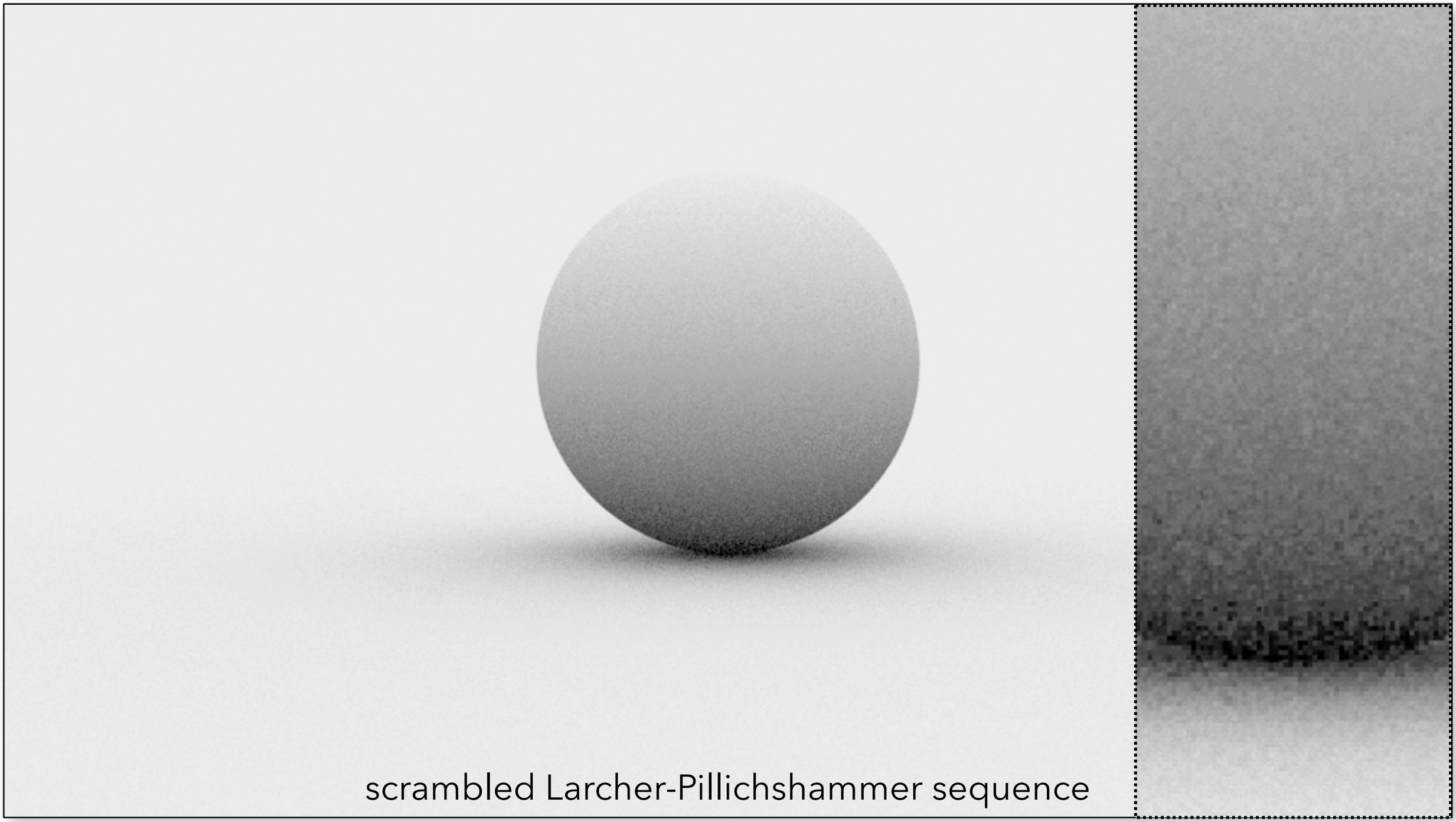
# Monte Carlo (16 stratified samples)



# Quasi-Monte Carlo (16 Halton samples)



# Scrambled Quasi-Monte Carlo



scrambled Larcher-Pillichshammer sequence

# Implementation tips

- Using QMC can often lead to unintuitive, difficult-to-debug problems.
  - Always code up MC algorithms first, using random numbers, to ensure correctness
  - Only after confirming correctness, slowly incorporate QMC into the mix

# Questions?