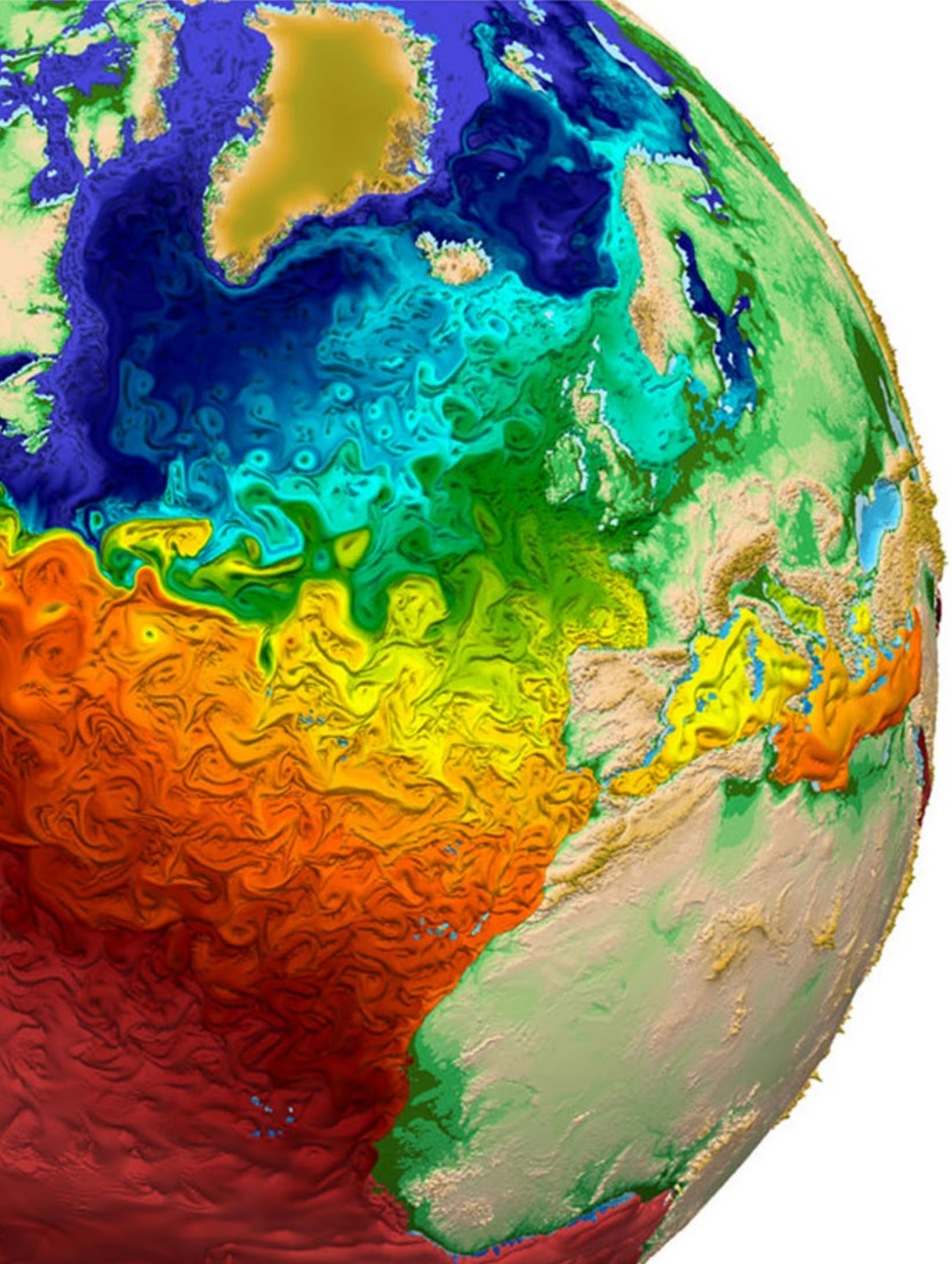


Numerical Methods

Sampling Domains

Eric Shaffer

ILLINOIS

2D Sampling

- Assume we are sampling a function on a unit square

- Good sampling
  - Not structured…involves some amount of randomness
  - Uniform(ish) distribution…avoid gaps and clumps
  - Projections into 1D along x and y are also uniform(ish)
  - There is a non-trivial minimum distance between all sample points

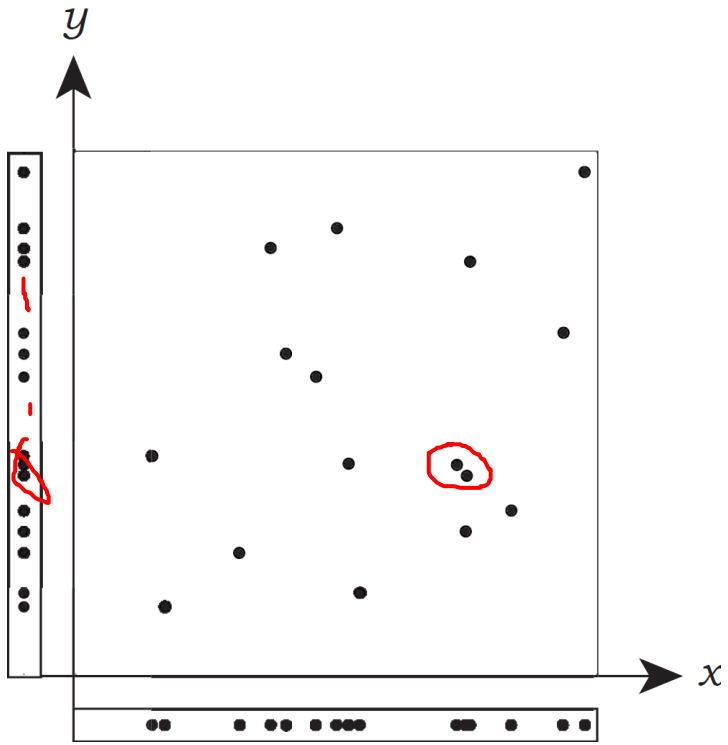
- Such a sample pattern is called *Well-Distributed*

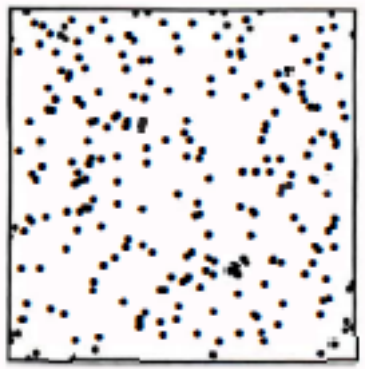
Random

Too irregular

Oversamples some areas…

Undersamples others





By random we mean a floating point values generated by a pseudo-random number generator of reasonable quality like Python's:

`random.random()`

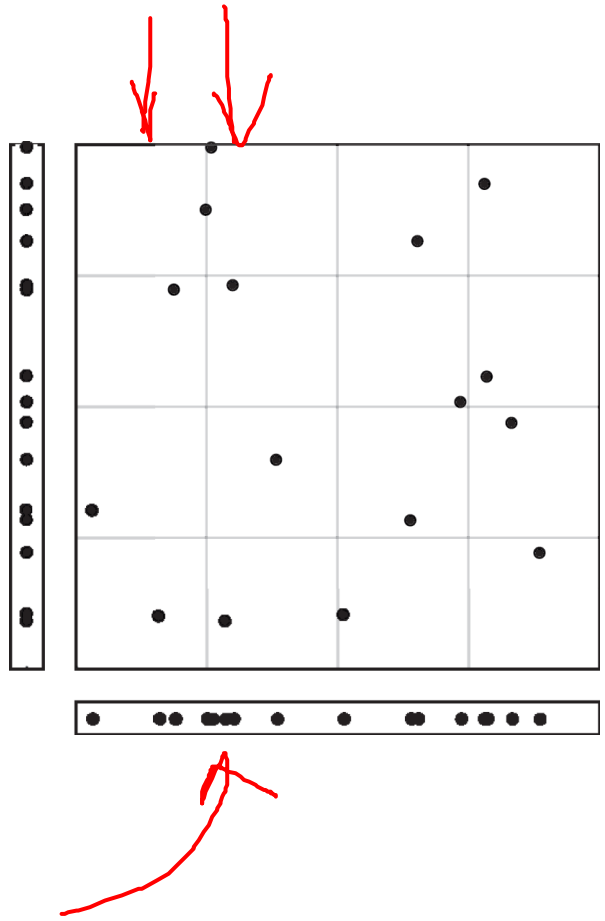
Return the next random floating point number in the range [0.0, 1.0).

Jittered

- Create a n x n grid covering the domain
  - Or a n^d uniform grid in d dimensional space
- Randomly generate a sample in each cell

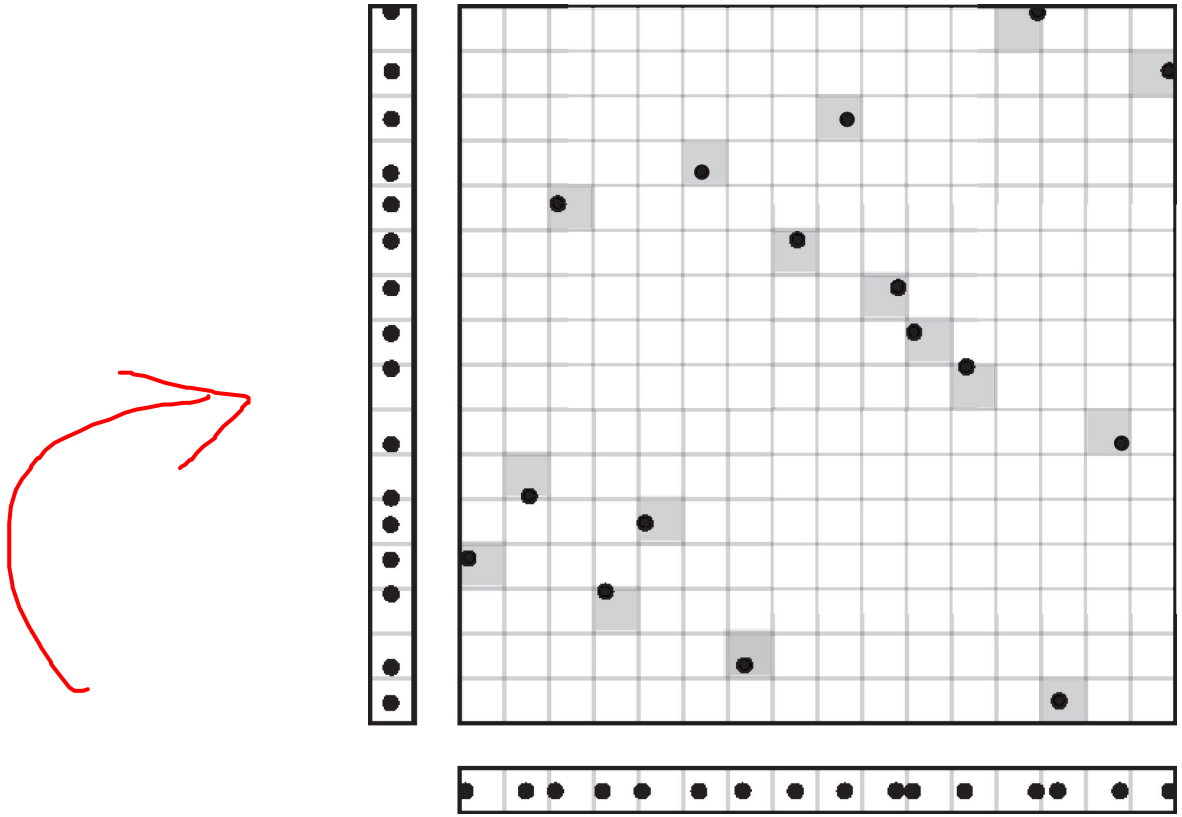
- Example of *stratified* sampling
  - Each cell is a strata

- Significantly better than random
- x-y projections can still be poorly distributed



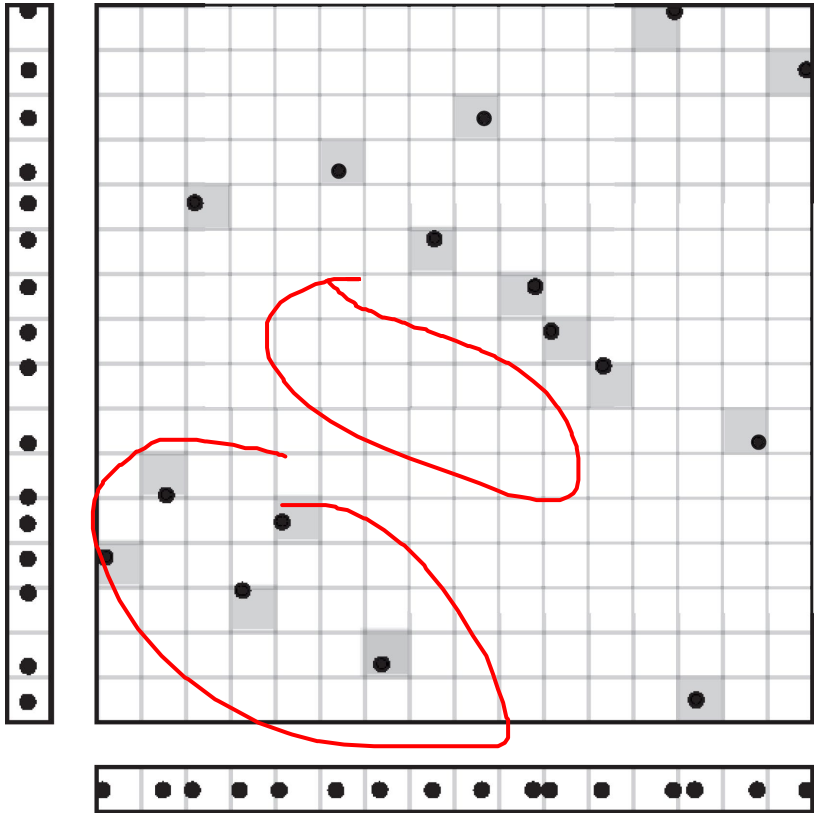
n-rooks

- Also called Latin hypercube sampling
- Use and n x n grid
- One sample exactly in each row and column
  - Again, randomly position a sample within the cell containing it
  - If samples were rooks in chess, no captures can occur



n-rooks

- Produced by random shuffle of diagonal samples
  - Must maintain the rook condition
- Use n samples instead of n^2 as in jittered
- 1D distributions are good…better than jittered
- 2D barely better than random…worse than jittered



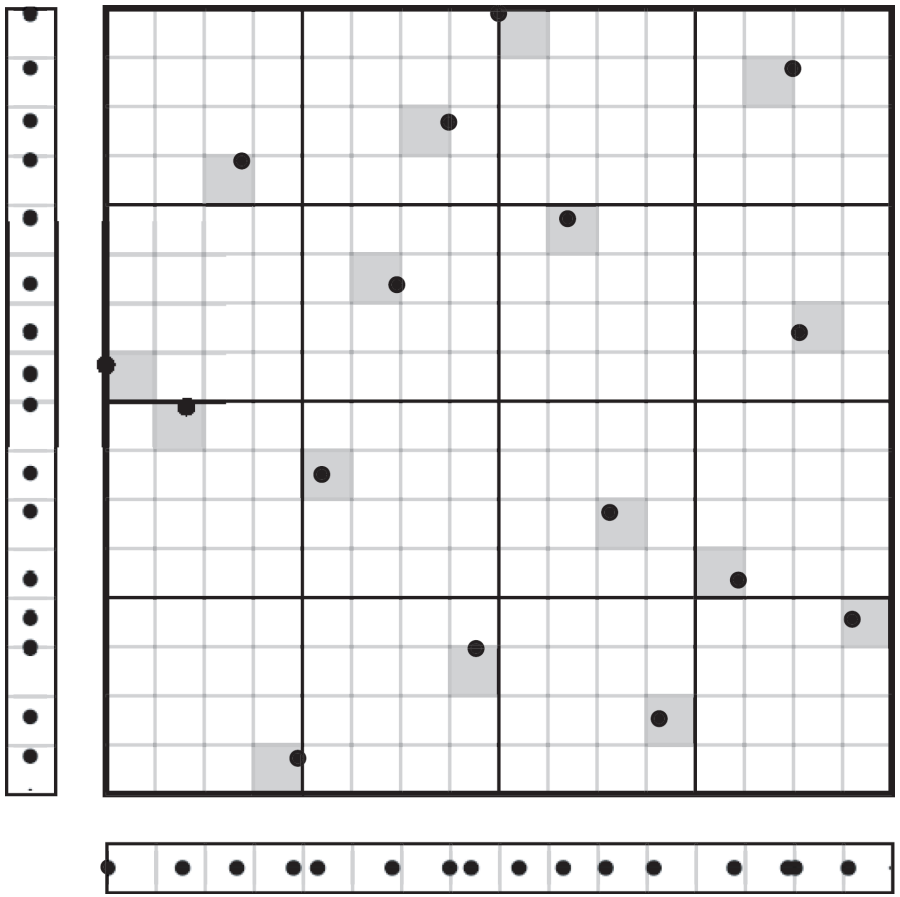
Developed by UIUC alum Pete Shirley in 1991

ILLINOIS

Multi-Jittered Sampling

- We use two grids
- For n samples with n a perfect square
  - Coarse grid is $\sqrt{n} \times \sqrt{n}$
  - Fine grid is $n \times n$
  - 1 sample per coarse grid cell
  - For each select unique row & column of fine grid
  - Randomly position sample in fine grid cell

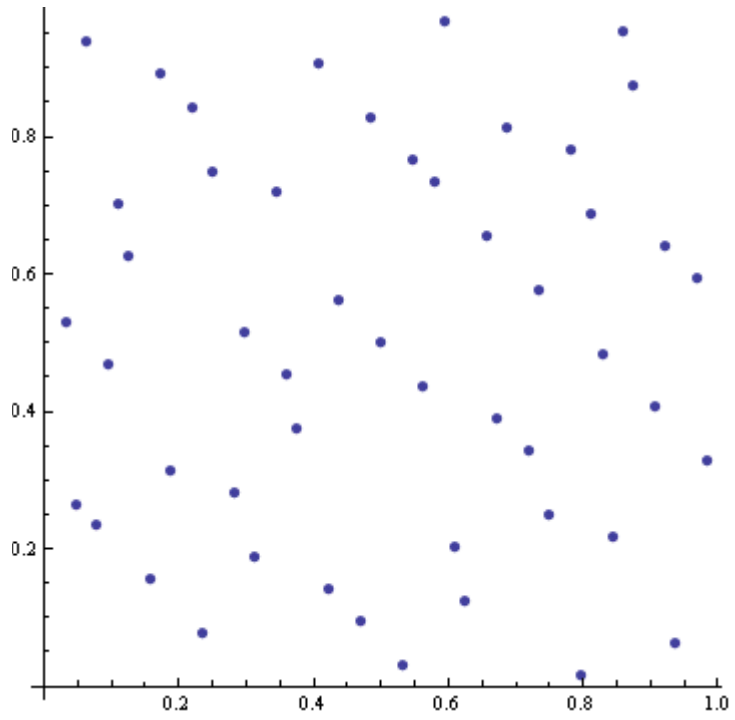
- Good 1D projections from the rook condition
- Good 2D distribution from stratification
- Very good sampling technique

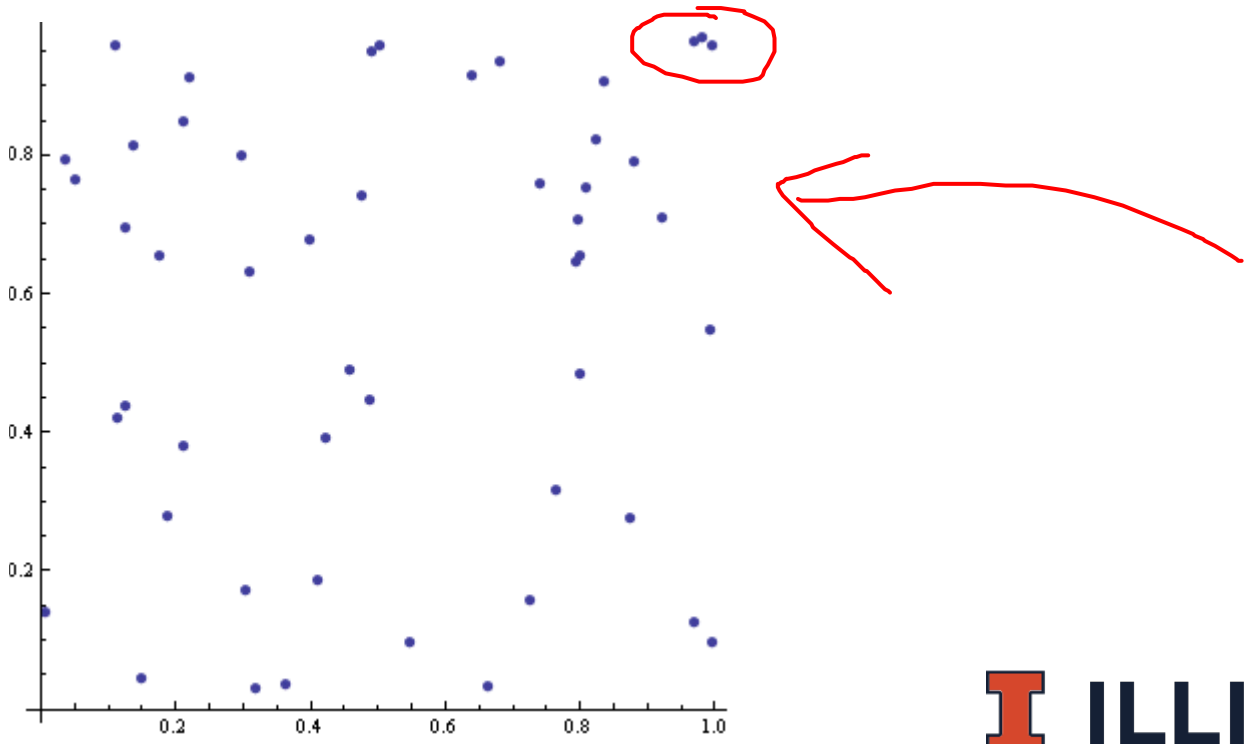


Hammersley Sampling

Deterministic quasi-random sequence

Low-discrepancy sequences are also called quasirandom sequences, due to their common use as a replacement of uniformly distributed random numbers. The "quasi" modifier is used to denote more clearly that the values of a low-discrepancy sequence are neither random nor pseudorandom.
- Wikipedia





Hammersley Sampling

$$\Phi_2(i) = \sum_{j=0}^{n} a_j(i) 2^{-j-1} = a_0(i)\frac{1}{2} + a_1(i)\frac{1}{4} + a_2(i)\frac{1}{8} \ldots.$$

Radical inverse function of integer i to base 2

- reflect binary digits of i across decimal point
- evaluate this new number now in [0,1)

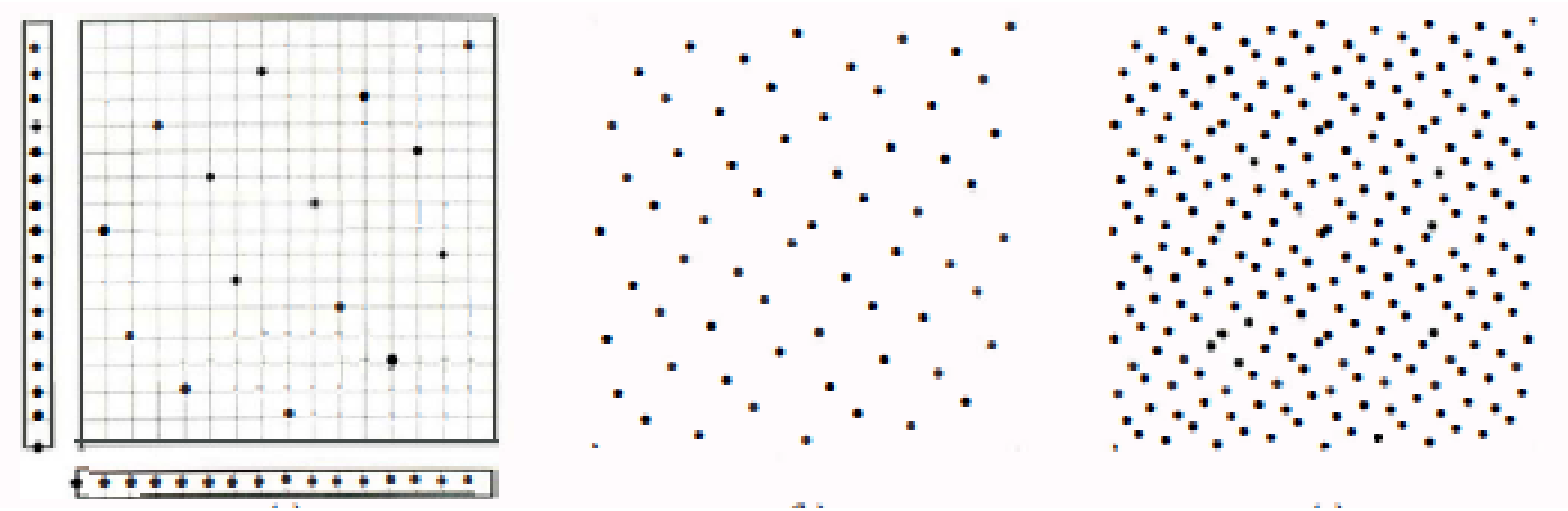
i			Reflection around the Decimal Point			$\Phi_2(i)$ (base 2)
1	=	1_2	$.1_2$	=	1/2	0.5
2	=	10_2	$.01_2$	=	1/4	0.25
3	=	11_2	$.11_2$	=	1/2 + 1/4	0.75
4	=	100_2	$.001_2$	=	1/8	0.125
5	=	101_2	$.101_2$	=	1/2 + 1/8	0.635
6	=	110_2	$.011_2$	=	1/4 + 1/8	0.325
7	=	111_2	$.111_2$	=	1/2 + 1/4 + 1/8	0.875
8	=	1000_2	$.0001_2$	=	1/16	0.0625

ILLINOIS

Hammersley Sequence in 2D

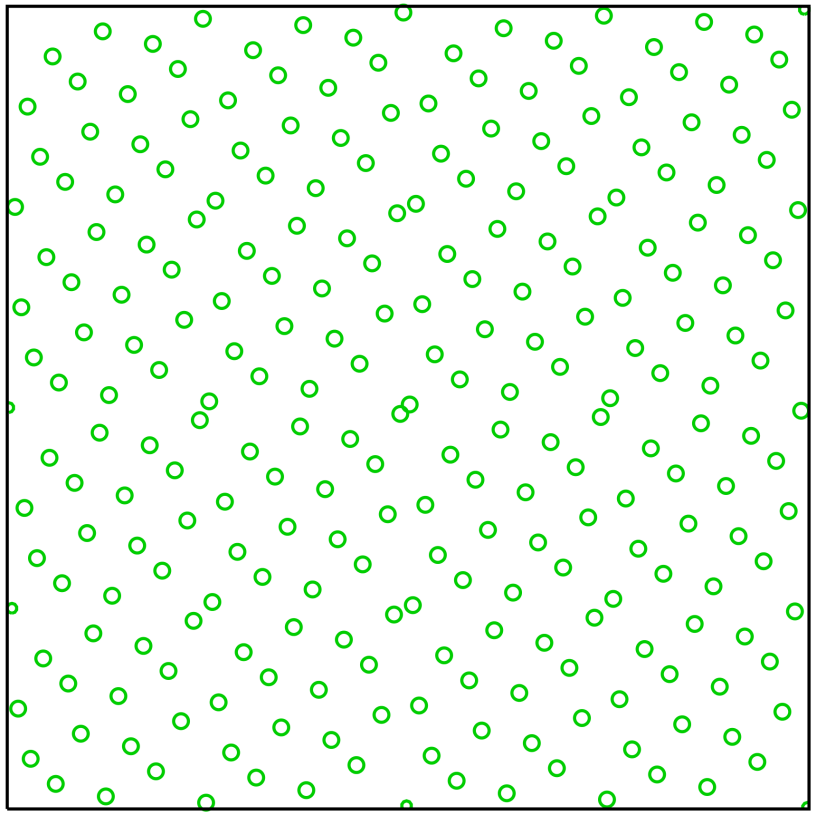
Set of n 2D samples in unit square:

$$p_i = (x_i, y_i) = [i/n, \Phi_2(i)]$$



Hammersley Issues

- Pattern is well-distributed but…
- 1D projections are regular
  - Too much structure in the pattern
- For a given n only one sequence exists



Halton Sequence

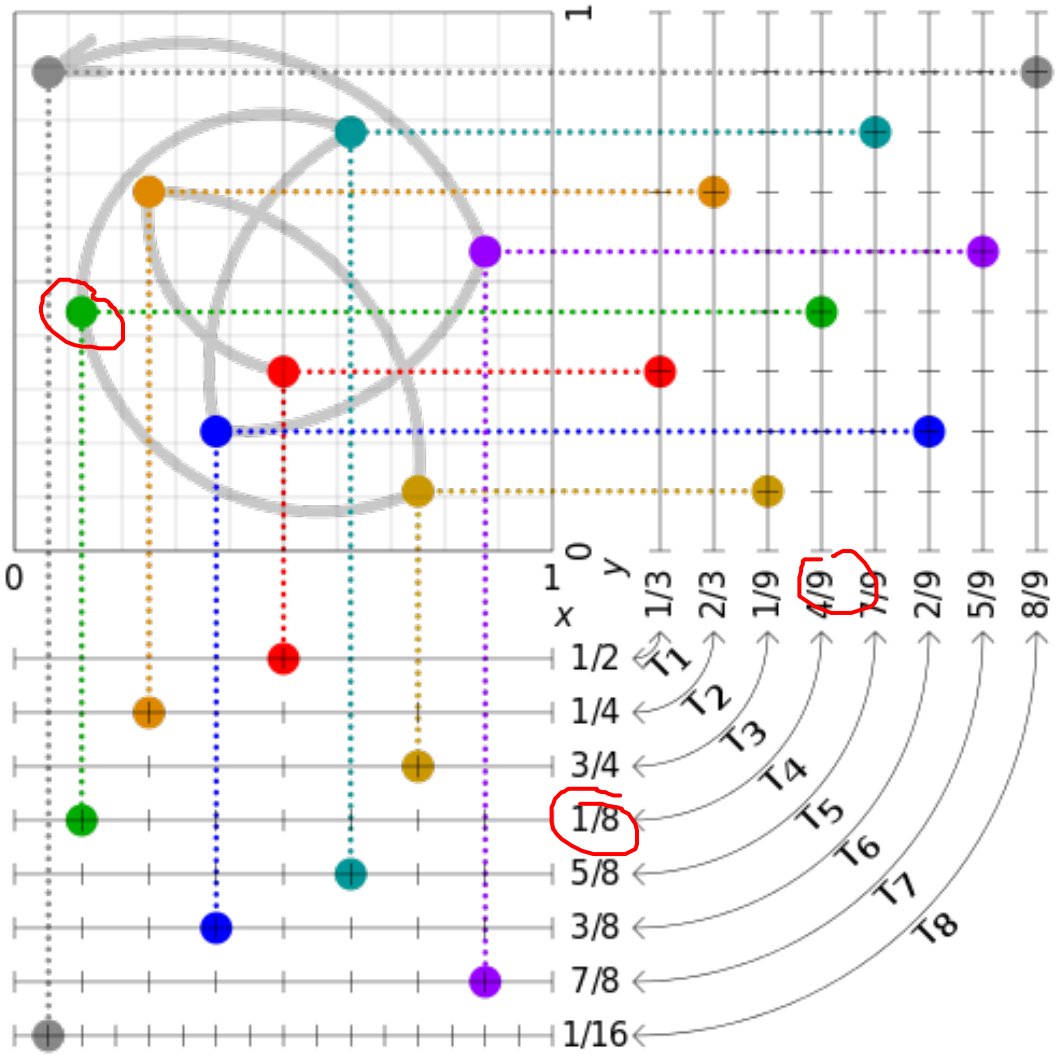
$$p_i = (\Phi_2(i), \Phi_3(i), \Phi_5(i), \ldots)$$

- Better low discrepancy sequence
- Generate n-dimensional points
  - though Hammersley can be generalized as well….
- Number of samples need not be known in advance

ILLINOIS

Halton Sequence Example

The 2,3 Halton Sequence



$\Phi_2(4) =$

.001

$\Phi_3(4) =$

0.11

Definition: Discrepancy

- We often want a low discrepancy sequence
  - e.g. Monte Carlo methods

- Imagine points in some space $S = [0,1]^n$
- Suppose we sample using K points…
- We can evaluate the quality by
  - take portion V of S
  - volume V/volume S should equal (number points in V)/(number points in S)
    - …but it generally won't
  - the difference is the *discrepancy*
- Different formal ways to measure discrepancy…

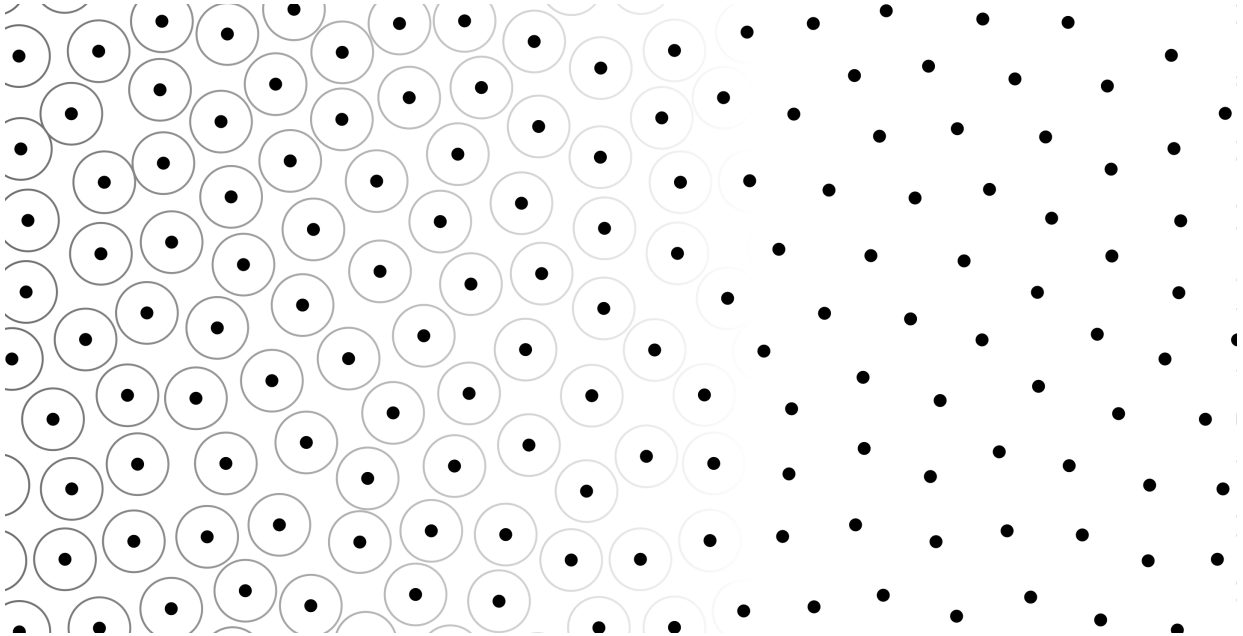
Poisson Disk Sampling

Not actually a low-discrepancy sequence

Does assure a minimum distance between points

Fast and generalizes well to higher dimensions

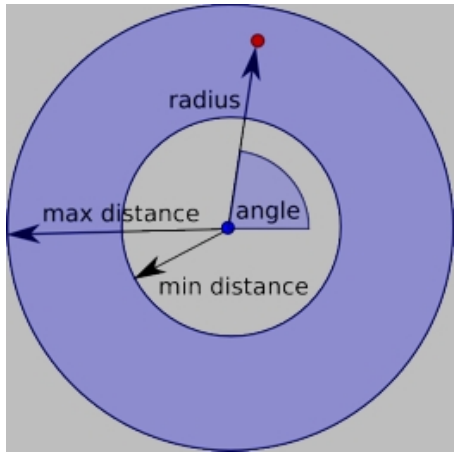
Popular in game industry



Poisson Disk Sampling

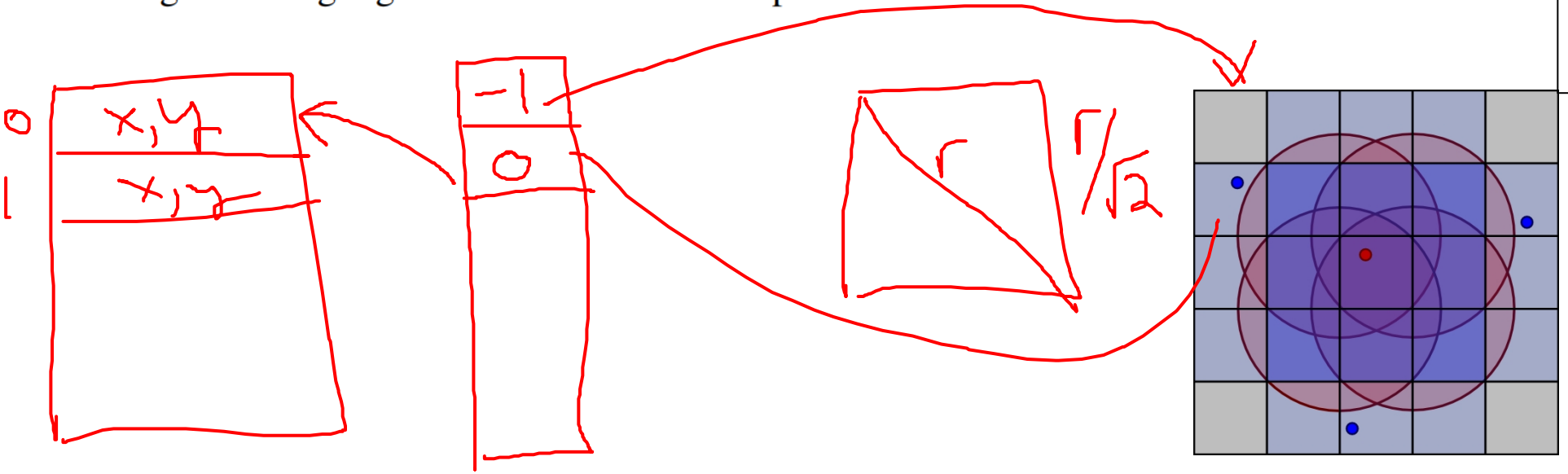
Algorithm

Step 0. Initialize an n-dimensional background grid for storing samples and accelerating spatial searches. We pick the cell size to be bounded by $r/\sqrt{n}$, so that each grid cell will contain at most one sample, and thus the grid can be implemented as a simple n-dimensional array of integers: the default -1 indicates no sample, a non-negative integer gives the index of the sample located in a cell.



r is minimum distance between two sample points

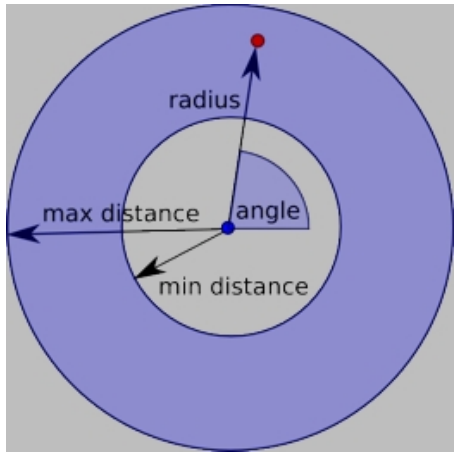
r will be the length of a cell diagonal

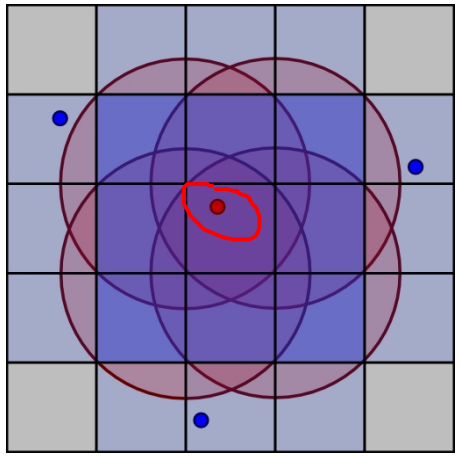


Poisson Disk Sampling

Algorithm

Step 1. Select the initial sample, x_0, randomly chosen uniformly from the domain. Insert it into the background grid, and initialize the "active list" (an array of sample indices) with this index (zero).

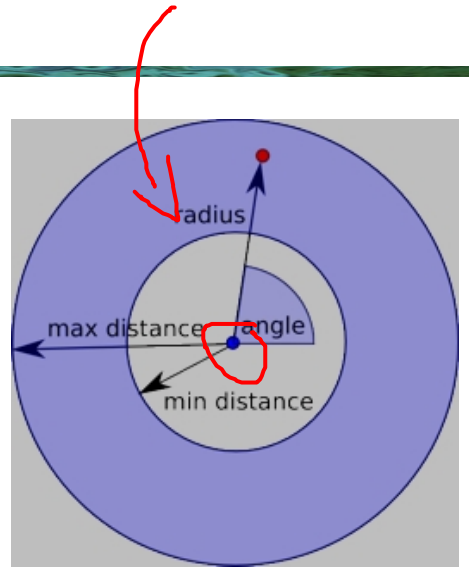




Poisson Disk Sampling

Algorithm

Step 2. While the active list is not empty, choose a random index from it (say i). Generate up to k points chosen uniformly from the spherical annulus between radius r and $2r$ around x_i. For each point in turn, check if it is within distance r of existing samples (using the background grid to only test nearby samples). If a point is adequately far from existing samples, emit it as the next sample and add it to the active list. If after k attempts no such point is found, instead remove i from the active list.



r will be the length of a cell diagonal

k=30 is a popular choice