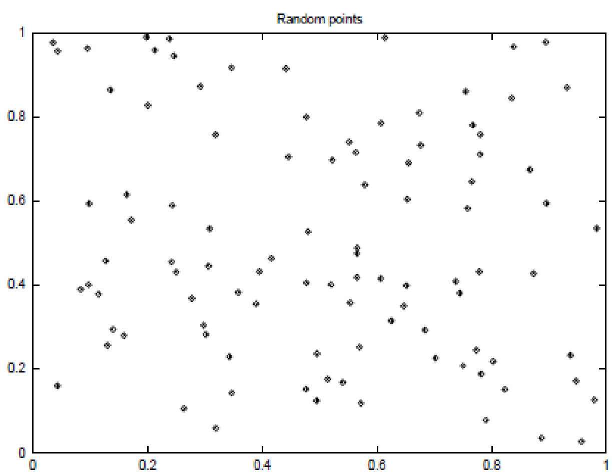
Computer graphics III – Low-discrepancy sequences and quasi-Monte Carlo methods

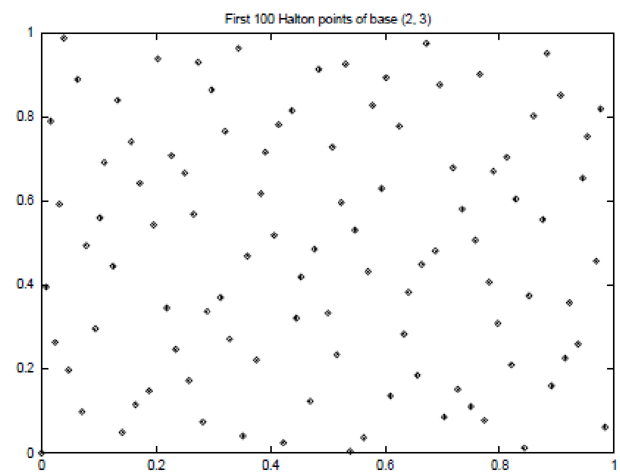
Jaroslav Křivánek, MFF UK
Jaroslav.Krivanek@mff.cuni.cz

Quasi-Monte Carlo

- Goal: Use point sequences that cover the integration domain as uniformly as possible, while keeping a 'randomized' look of the point set



High Discrepancy
(clusters of points)



Low Discrepancy
(more uniform)

Transformation of point sets

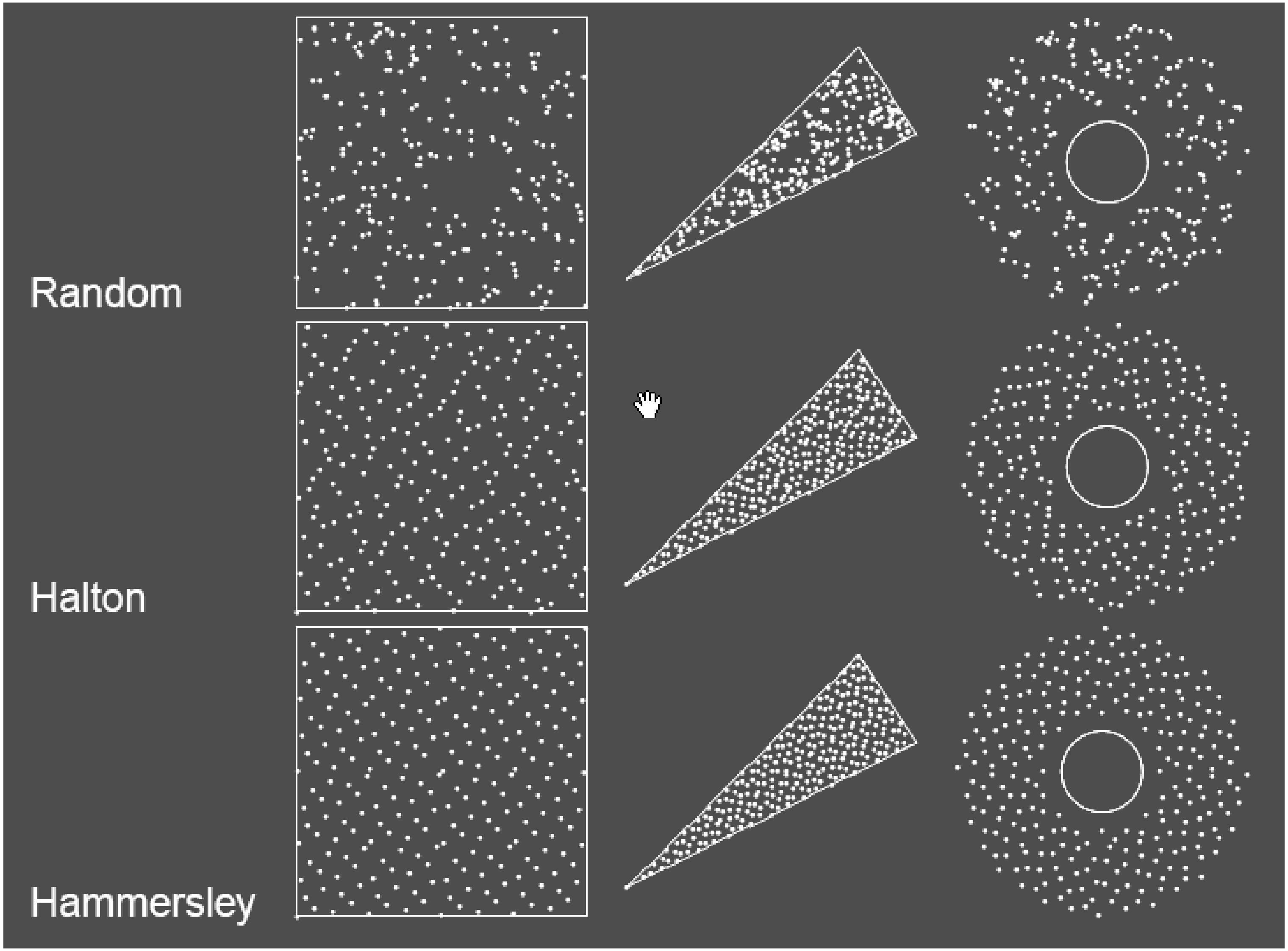


Image credit: Alexander Keller

MC vs. QMC

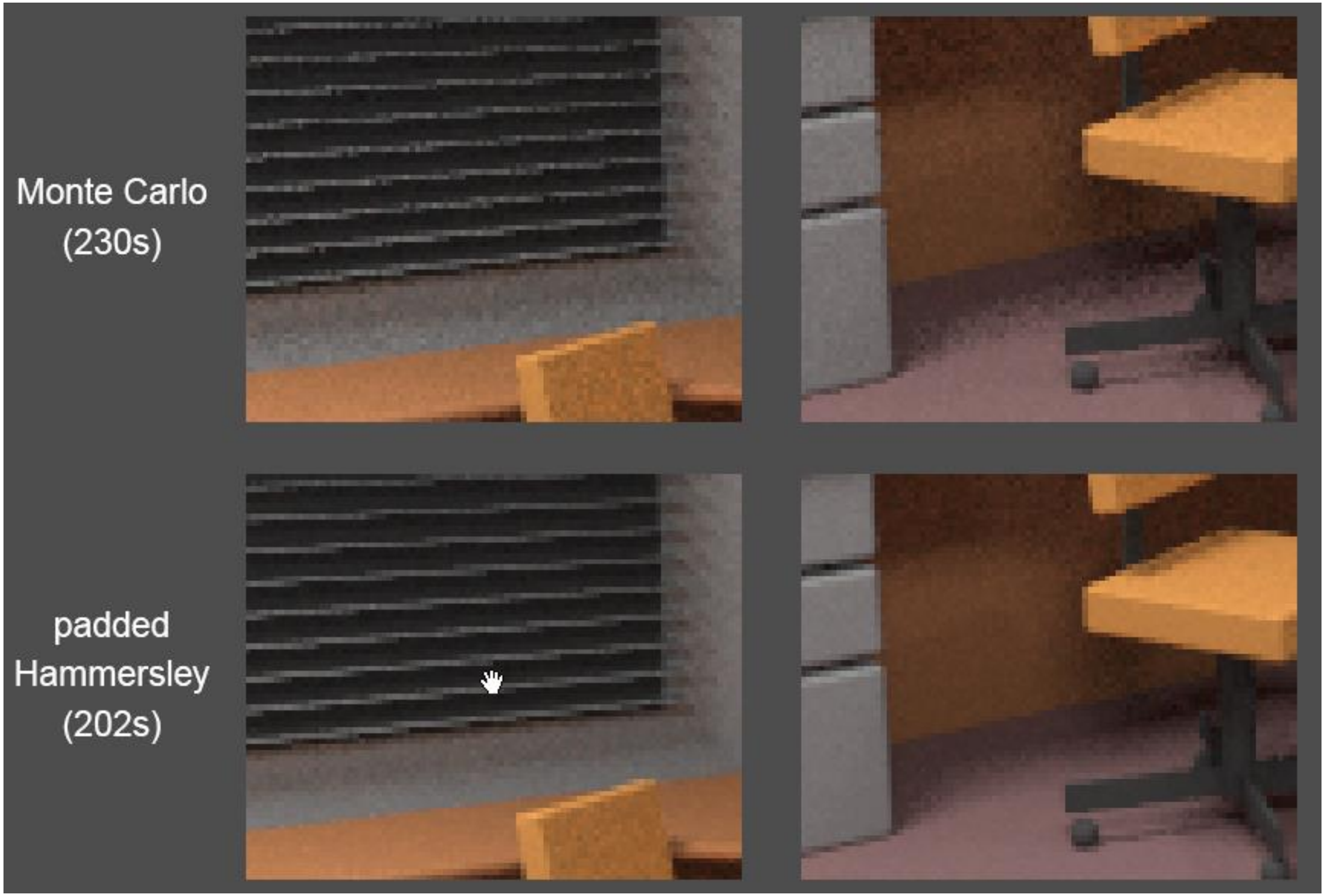


Image credit: Alexander Keller

Quasi Monte Carlo (QMC) methods

- Use of strictly deterministic sequences instead of random numbers
- All formulas as in MC, just the underlying proofs cannot reply on the probability theory (nothing is random)
- Based on sequences **low-discrepancy sequences**

Defining discrepancy

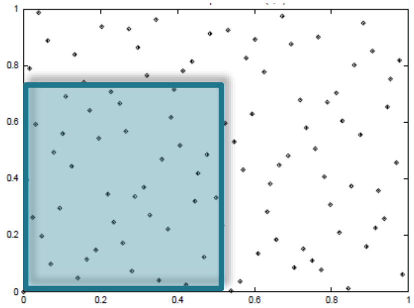
- *s*-dimensional "brick" function:

$$\mathcal{L}(\mathbf{z}) = \begin{cases} 1 \text{ if } 0 \le \mathbf{z}|_1 \le v_1, 0 \le \mathbf{z}|_2 \le v_2, \dots, 0 \le \mathbf{z}|_s \le v_s \\ 0 \text{ otherwise.} \end{cases}$$

- True volume of the "brick" function:

$$V(A) = \prod_{j=1}^{s} v_j$$

- MC estimate of the volume of the "brick":



$$\frac{1}{N}\sum_{i=1}^{N} f(\mathbf{z}_i) = \frac{m(A)}{N}$$

total number of sample points

number of sample points that actually fell inside the "brick"

Discrepancy

- Discrepancy (of a point sequence) is the maximum possible error of the MC quadrature of the “brick” function over all possible brick shapes:

$$\mathcal{D}^*(\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_N) = \sup_A \left| \frac{m(A)}{N} - V(A) \right|.$$

  - serves as a measure of the uniformity of a point set
  - must converge to zero as N -> infty
  - the lower the better (cf. **Koksma-Hlawka Inequality**)

Koksma-Hlawka inequality

- Koksma-Hlawka inequality

„variation“ of *f*

$$\left| \int_{\mathbf{z}\in[0,1]^s} f(\mathbf{z})\, d\mathbf{z} - \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{z}_i)\right| \leq \mathcal{V}_{\mathrm{HK}} \cdot \mathcal{D}^*(\mathbf{z}_1, \mathbf{z}_2, \ldots \mathbf{z}_N)$$

  - the KH inequality only applies to *f* with finite variation
  - QMC can still be applied even if the variation of f is infinite

Van der Corput Sequence (base 2)

i	binary form of i	radical inverse	H_i
1	1	0.1	0.5
2	10	0.01	0.25
3	11	0.11	0.75
4	100	0.001	0.125
5	101	0.101	0.625
6	110	0.011	0.375
7	111	0.111	0.875

- point placed in the middle of the interval
- then the interval is divided in half
- has low-discrepancy

Table credit: Laszlo Szirmay-Kalos

Van der Corput Sequence

- b … **Base**
- radical inverse

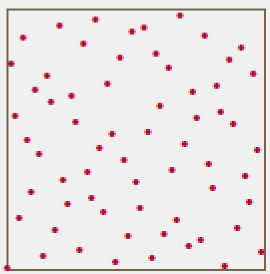
$$i = \sum_{j=0}^{\infty} a_j(i) b^j \quad \mapsto \quad \Phi_b(i) := \sum_{j=0}^{\infty} a_j(i) b^{-j-1}$$

Van der Corput Sequence (base *b*)

```
double radicalInverse(const int base, int i)
{
        double digit, radical;
        digit = radical = 1.0 / (double)base;
        double inverse = 0.0;
        while(i)
        {
                inverse += digit * (double)(i % base);
                digit *= radical;
                i /= base;
        }
        return inverse;
}
```

Sequences in higher dimension

Halton sequence $x_i := (\Phi_{b_1}(i), \ldots, \Phi_{b_s}(i))$ where b_i is the i-th prime number



Hammersley point set $x_i := \left(\frac{i}{n}, \Phi_{b_1}(i), \ldots, \Phi_{b_{s-1}}(i)\right)$

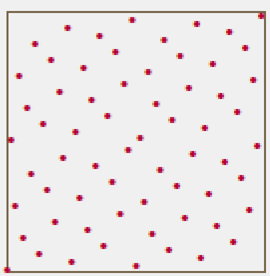


Image credit: Alexander Keller

Use in path tracing

- **Objective**: Generated paths should cover the entire high-dimensional path space uniformly

- **Approach**:
  - Paths are interpreted as “points” in a high-dimensional path space

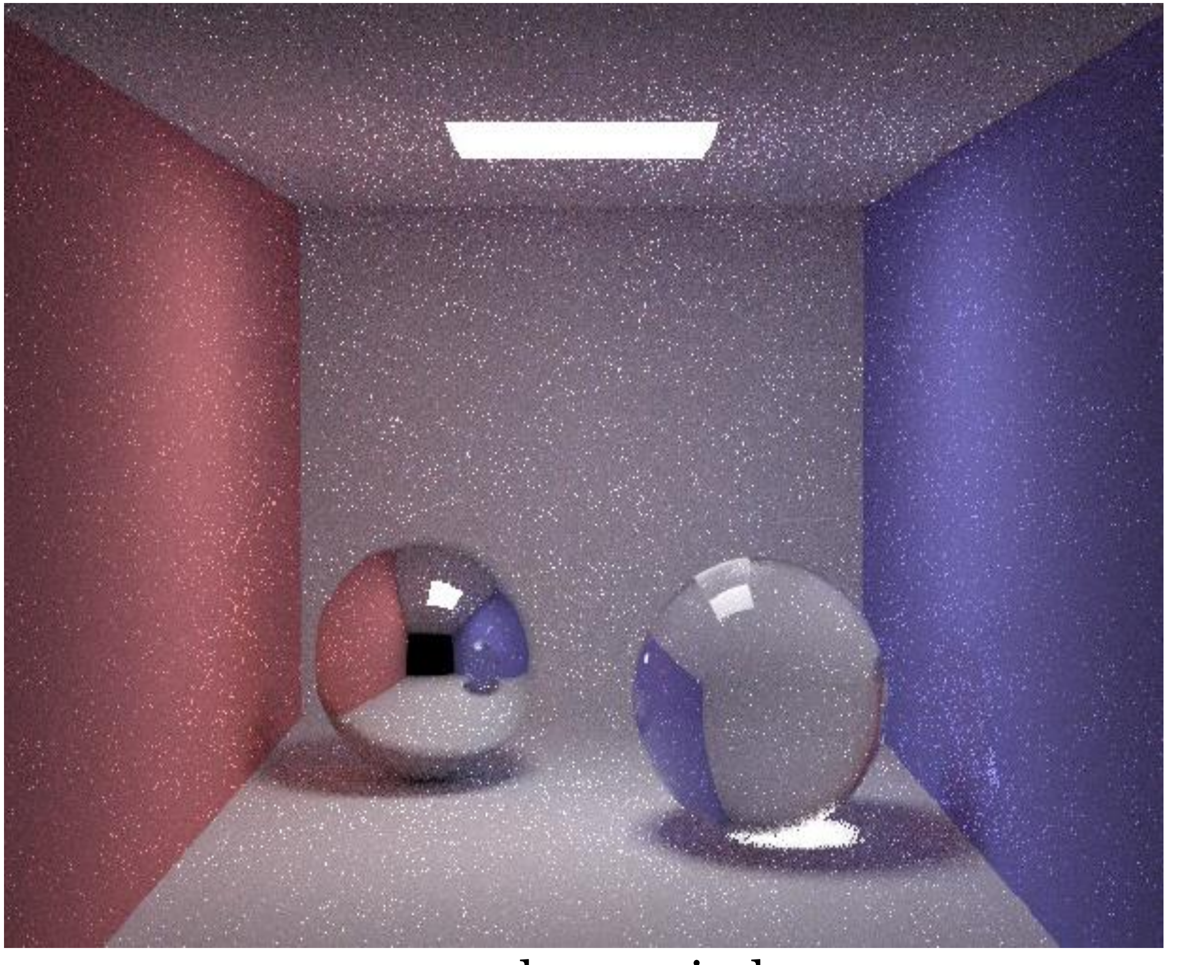
  - Each path is defined by a long vector of “random numbers”
    - **Subsequent random events** along a single path use **subsequent components** of the **same** vector

  - Only when tracing the next path, we switch to a brand new “random vector” (e.g. next vector from a Halton sequence)

Quasi-Monte Carlo (QMC) Methods

- Disadvantages of QMC:
  - Regular patterns can appear in the images (instead of the more acceptable noise in purely random MC)
  - Random scrambling can be used to suppress it

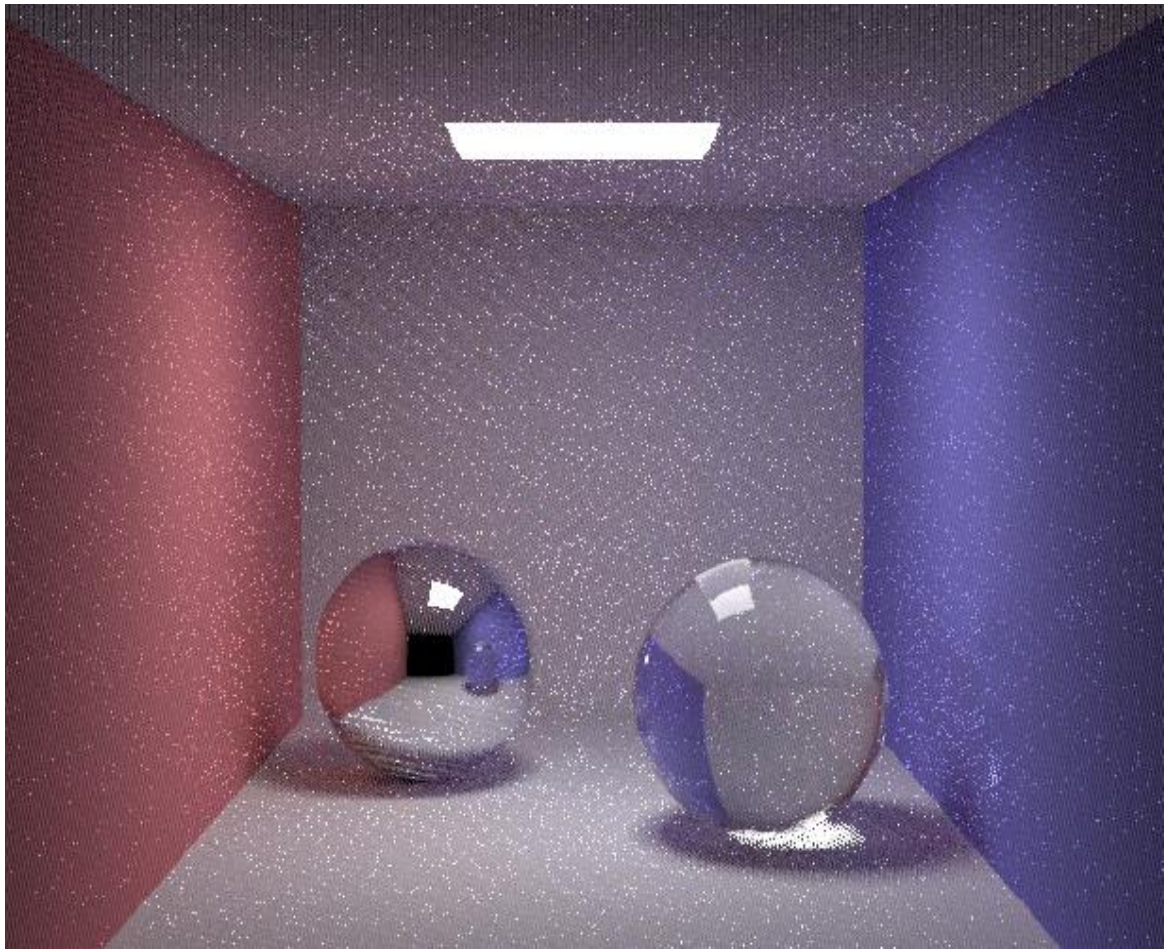
Stratified sampling



Henrik Wann Jensen

10 paths per pixel

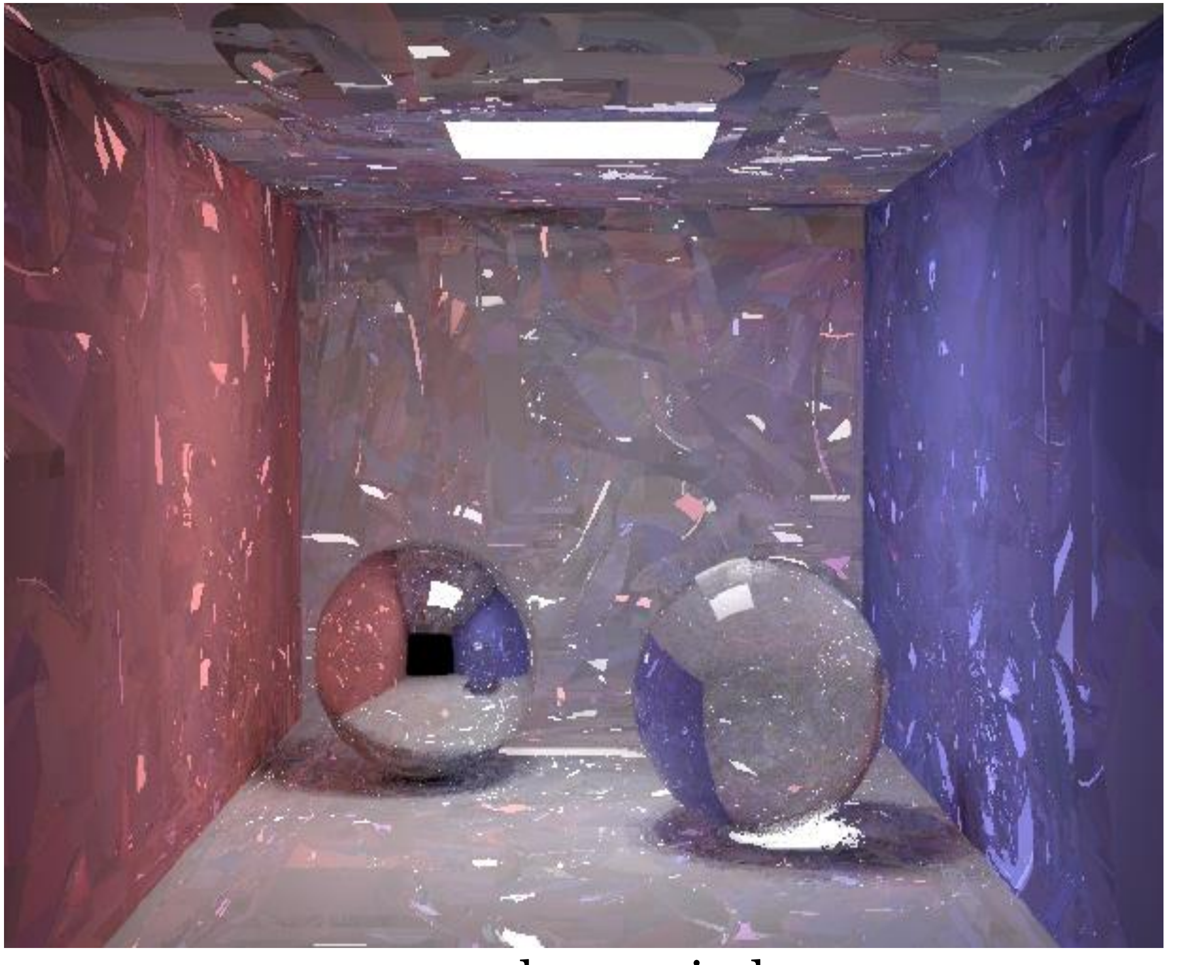
Quasi-Monte Carlo



Henrik Wann Jensen

10 paths per pixel

Fixní náhodná sekvence



Henrik Wann Jensen

10 paths per pixel