INSTANT RADIOSITY

Keller (SIGGRAPH 1997)

Presented by Ivo Boyadzhiev and Kevin Matzen

BRIEF HISTORY - RADIOSITY

- Familiar FEM approach
  - Discretize geometry
  - Assume simple, Lambertian surfaces
  - Encode light transport directly
  - Solve
- Pros
  - Viewpoint independent
  - Simple, in principle
- Cons
  - Complicated form factors
  - Remeshing
  - Discretization artifacts
  - Does not capture complex materials

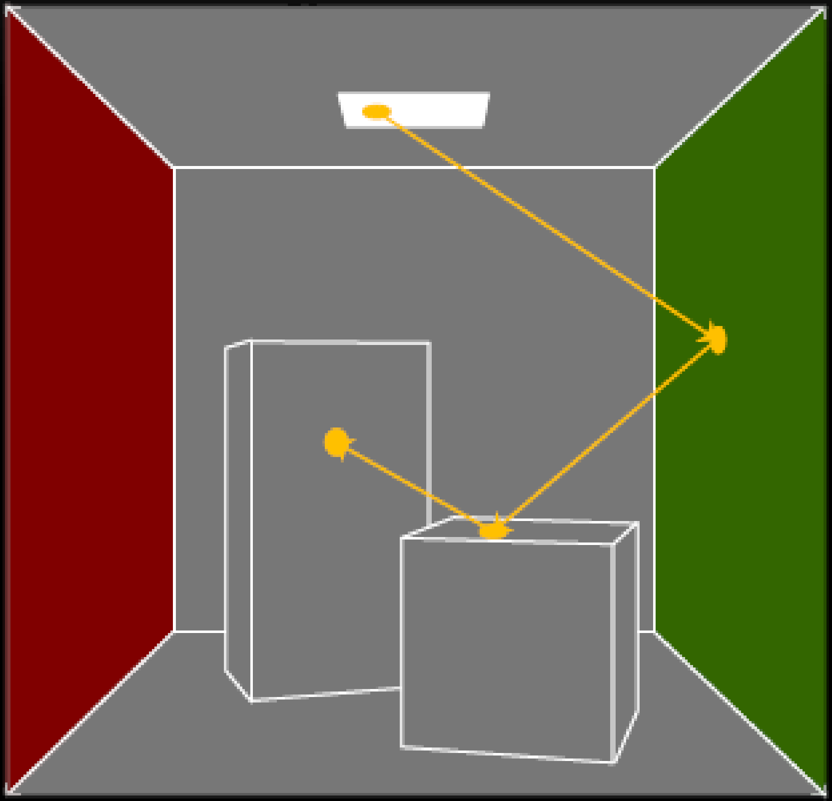
Modern CAD tools use this for interactive rendering!
(3ds Max, etc.)

Cohen et. al. (SIGGRAPH'88)

IDEA – INSTANT RADIOSITY (KELLER SIGGRAPH '97)

- Concentrate power of luminaires at samples
  - No explicit discretization
  - No complex form factors
  - Simple point lights
- Bounce energy around scene – leave virtual point lights at bounces
  - Reusable paths
- Fast HW accelerated render passes
- Still assumes Lambertian surfaces
  - Neat hack to handle ideal specular surfaces.

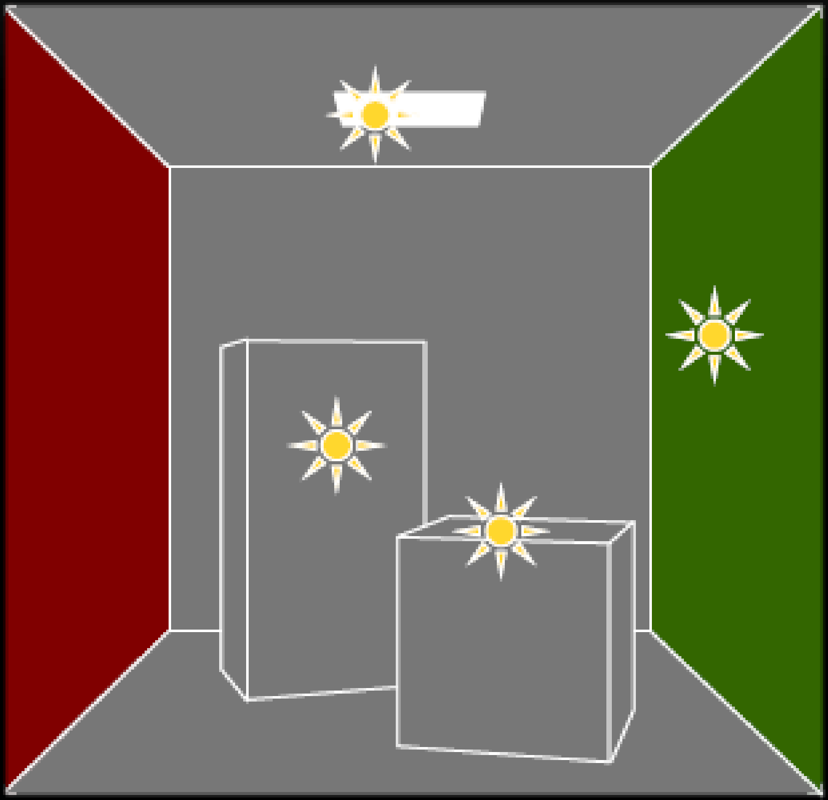
ALGORITHM BASICS



- STEP 1
  - Photons are traced from the light source into the scene.

Diagram from M. Hasan (*SIGGRAPH Asia ‘2009)*

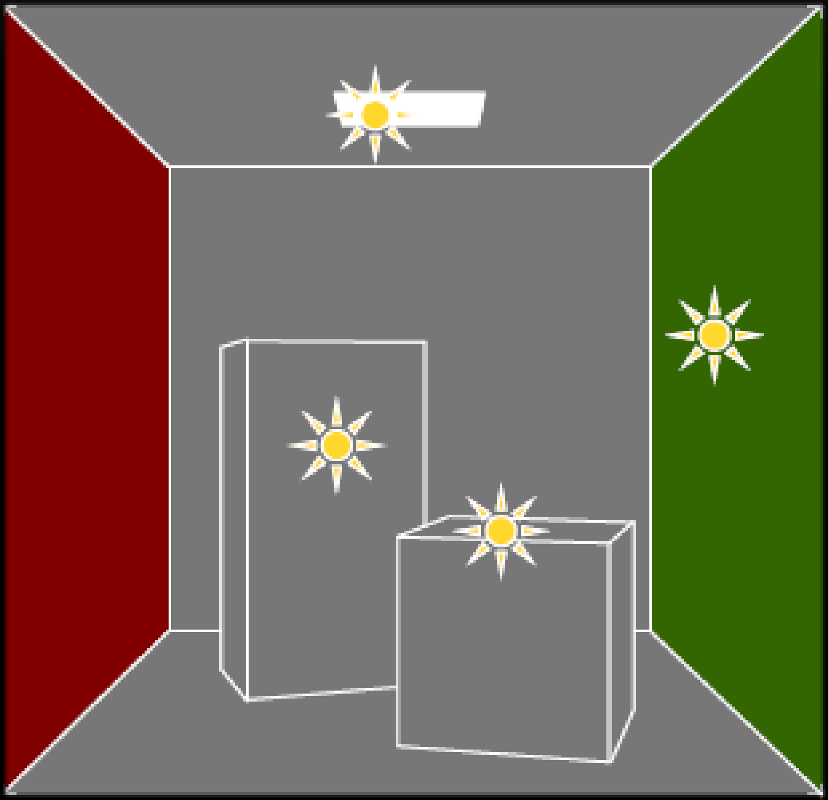
ALGORITHM BASICS



- STEP 1
  - Photons are traced from the light source into the scene.
  - Treat path vertices as Virtual Point Lights (VPLs).

Diagram from M. Hasan (*SIGGRAPH Asia '2009)*

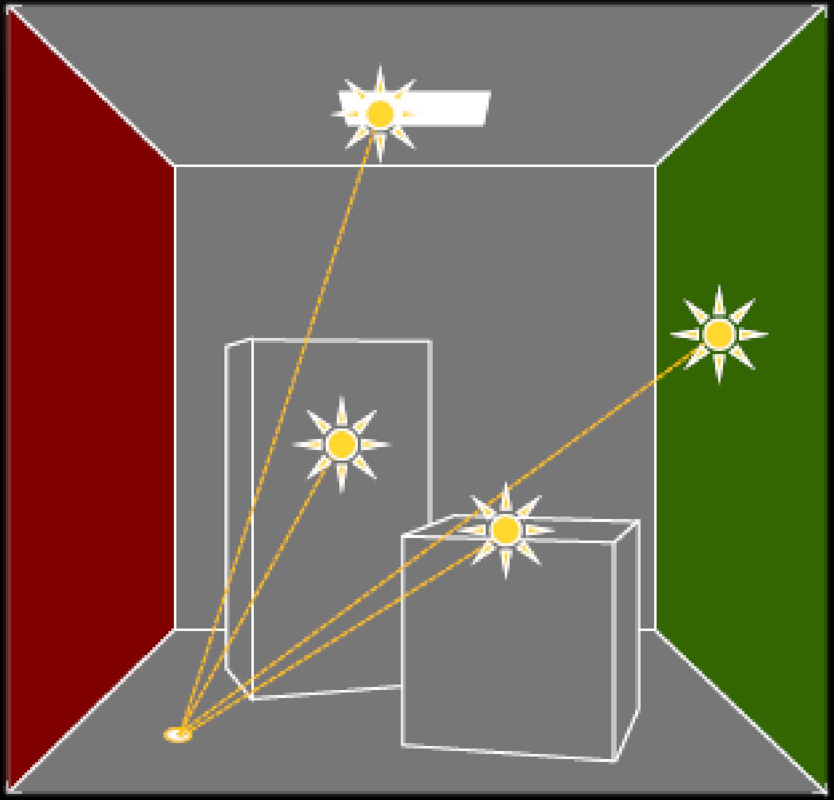
ALGORITHM BASICS



- STEP 1
  - Photons are traced from the light source into the scene.
  - Treat path vertices as Virtual Point Lights (VPLs).
  - Generates a particle approximation of the diffuse radiant, using Quasi-random walk based on quasi-Monte Carlo integration.

Diagram from M. Hasan (*SIGGRAPH Asia ‘2009)*

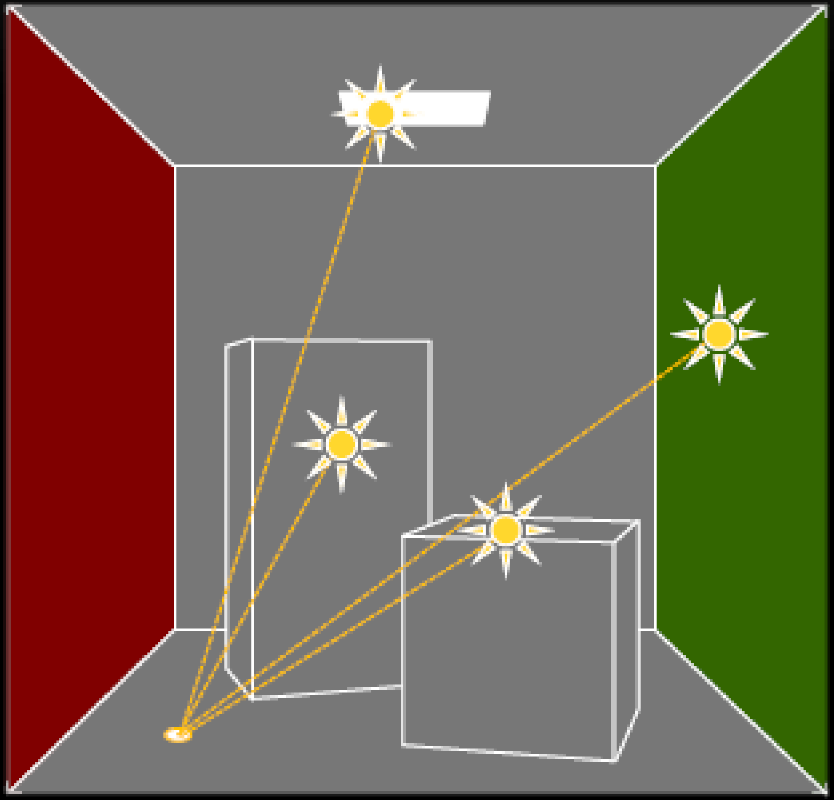
ALGORITHM BASICS



- STEP 1
  - Photons are traced from the light source into the scene.
  - Treat path vertices as Virtual Point Lights (VPLs)
  - Generates a particle approximation of the diffuse radiant, using Quasi-random walk based on quasi-Monte Carlo integration.
- STEP 2
  - The scene is rendered several times for each light source.

Diagram from M. Hasan (*SIGGRAPH Asia '2009)*

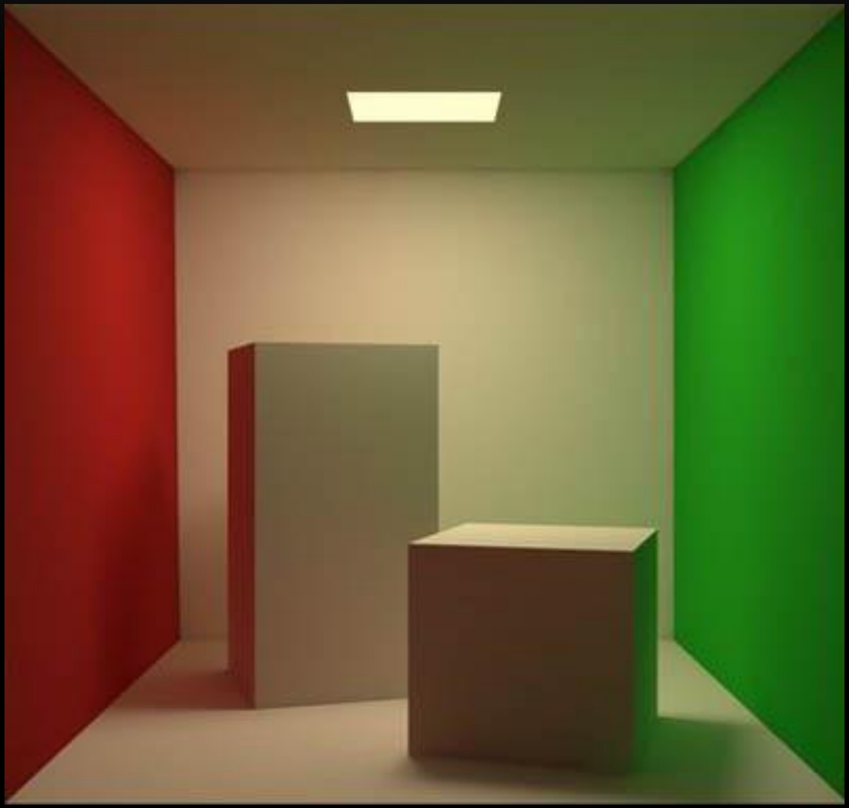
ALGORITHM BASICS



- STEP 1
  - Photons are traced from the light source into the scene.
  - Treat path vertices as Virtual Point Lights (VPLs)
  - Generates a particle approximation of the diffuse radiant, using Quasi-random walk based on quasi-Monte Carlo integration.
- STEP 2
  - The scene is rendered several times for each light source.
  - Hardware renders an image with shadows for each particle used as point light source.

Diagram from M. Hasan (*SIGGRAPH Asia '2009)*

ALGORITHM BASICS



Cornell Box, rendered using Instant Radiosity

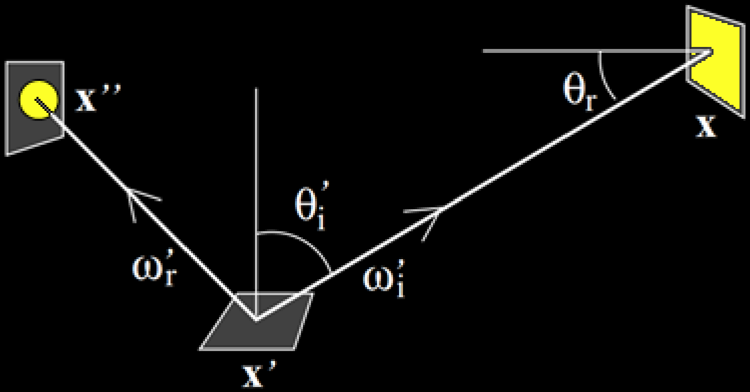
- STEP 1
  - Photons are traced from the light source into the scene.
  - Treat path vertices as Virtual Point Lights (VPLs)
  - Generates a particle approximation of the diffuse radiant, using Quasi-random walk based on quasi-Monte Carlo integration.
- STEP 2
  - The scene is rendered several times for each light source.
  - Hardware renders an image with shadows for each particle used as point light source.
  - Resulting image is composited in the accumulation buffer (hardware).

DERIVATION

Bounces from source to VPLs

$$L_r(x') = \frac{k_d(x')}{\pi} L_i(x)|\cos(\theta_i')|$$

$$L(x'') = L_e(x) \prod_{j=0}^{n} \frac{k_d(x_j)}{\pi} |\cos(\theta_j)|$$



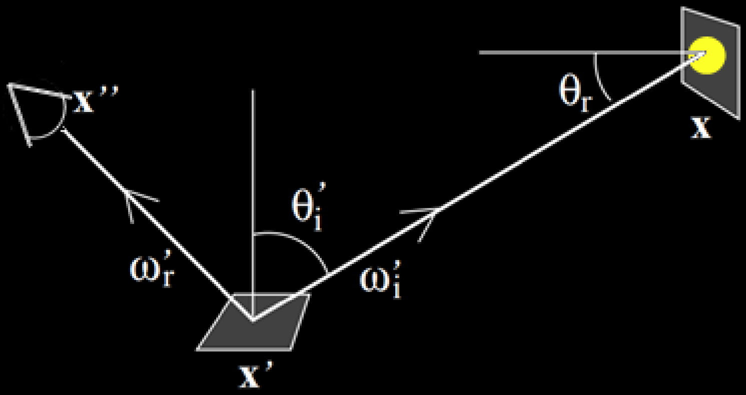
Notation from Veach and Guibas (SIGGRAPH ‘95)

DERIVATION

Bounce from VPLs to camera

$$L_r(x' \to x'') = \frac{k_d(x')}{\pi} \int_M V(x \leftrightarrow x') \frac{\cos(\theta_r)\cos(\theta_i')}{||x\ - x'||^2} L_i(x \to x') dA(x)$$

$$L(x' \to x'') = \frac{k_d(x')}{\pi} \sum_{x \in VPLs} V(x \leftrightarrow x') \frac{\cos(\theta_r)\cos(\theta_i')}{||x\ - x'||^2} L_i(x \to x')$$



Notation from Veach and Guibas (SIGGRAPH '95)

IMPLEMENTATION

Phase 1 – Quasi-Random Walk

```
foreach sample with n reflections:
  [x, pdf_x] = SampleLuminaire
  rad = L(x)/pdf_x

  for reflection in {0..n}:
    pdf_refl = pow(average_reflectivity, reflection)
    StoreVPL (x, rad/pdf_refl)
    [w, pdf_w] = SampleDirection
    rad *= $\frac{k_d(x)}{\pi}\cos(\theta)$/pdf_w
    [x] = RayTrace(x, w)
```

Notes on Keller’s implementation

Sampled by surface area (1/pdf_x = supp L)

Cosine weighted sampling
$\cos(\theta)$/pdf_w = 1

IMPLEMENTATION

Phase 1 – Quasi-Random Walk

```
foreach sample with n reflections:
  [x, pdf_x] = SampleLuminaire
  rad = L(x)/pdf_x

  for reflection in {0..n}:
    pdf_refl = pow(average_reflectivity, reflection)
    StoreVPL (x, rad/pdf_refl)
    [w, pdf_w] = SampleDirection
    rad *= $\frac{k_d(x)}{\pi}\cos(\theta)$/pdf_w
    [x] = RayTrace(x, w)
```

Phase 2 – Accumulation

```
foreach VPL in VPLs:
  [s] = ComputeSurfaceIntersections
  [v] = ComputeVisibility(s, VPL::x)
  [brdf] = EvaluateBRDF(s, VPL::x)
  Image += 1/N*v*brdf*cos*VPL::rad
```

NON-LAMBERTIAN SURFACES

- Point lights
  - Must match radiance distribution
  - Easy for Lambertian BRDF – can efficiently use fixed function pipeline

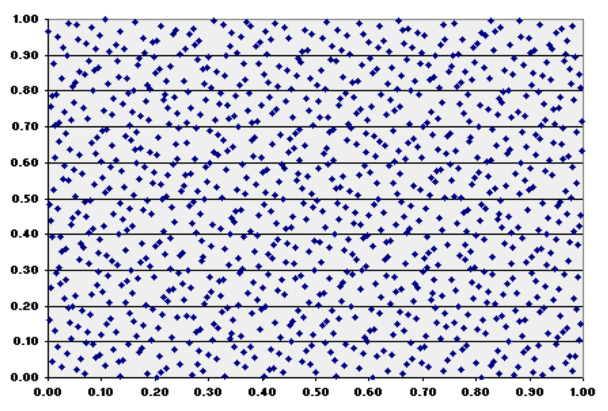
- Lambertian assumption
  - Not too important with modern programmable shaders
  - Needs to store incoming direction and delay last BRDF eval for other BRDFs
  - Can also use spot lights to simulate parametric BRDFs

- Ideal specular – not automatically compatible

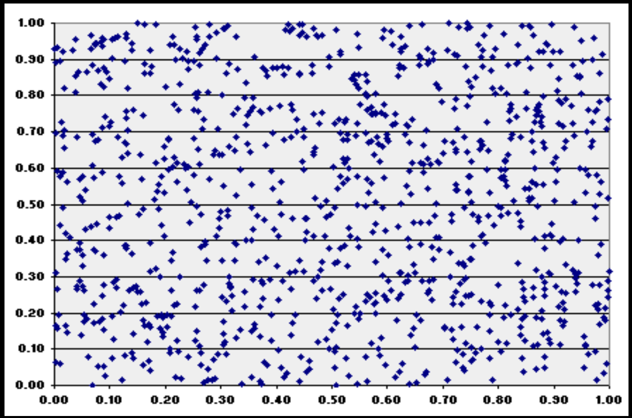
SAMPLING

QUASI-RANDOM NUMBERS

- Deterministic sequences, **that appear to be random for many purposes.**
- Quasi-random numbers may be used in Monte-Carlo simulation in the same way as pseudo-random numbers!
- Low-discrepancy: successive numbers are added in a position as far as possible from the other numbers (i.e. **avoiding clustering**).



1000 iterations, Halton sequence



1000 iterations, pseudo-random numbers

HALTON SEQUENCE (GENERATION)

- The Halton sequence in 1D is also known as the van der Corput sequence:
  1. Choose a prime base b.
  2. If n is an integer then it can be written in base b as:

$$n = \sum_{0}^{m} d_k b^k$$

  3. Then the n^{th} number in the Halton sequence of base b is given by (reflection + mapping to [0,1)):

$$\Phi_b(n) = \sum_{0}^{m} d_k b^{-(k+1)}$$

- Efficient algorithms exist for direct or **incremental** calculations [HW64].

HALTON SEQUENCE (EXAMPLE)

- The following table shows how to calculate the first 7 numbers in the Halton sequence of base 2:

n	d_2 d_1 d_0	$\Phi_2(n) =$
1	0 0 1	0*(1/8) + 0*(1/4) + 1*(1/2) = **0.5**
2	0 1 0	0*(1/8) + 1*(1/4) + 0*(1/2) = **0.25**
3	0 1 1	0*(1/8) + 1*(1/4) + 1*(1/2) = **0.75**
4	1 0 0	1*(1/8) + 0*(1/4) + 0*(1/2) = **0.125**
5	1 0 1	1*(1/8) + 0*(1/4) + 1*(1/2) = **0.625**
6	1 1 0	1*(1/8) + 1*(1/4) + 0*(1/2) = **0.375**
7	1 1 1	1*(1/8) + 1*(1/4) + 1*(1/2) = **0.875**

- Notice that the Halton sequence is essentially **filling in the largest gap** in the range (0;1), that doesn't already contain a number in the sequence: start by dividing the interval (0,1) in half, then in fourths, eighths, etc.

HALTON SEQUENCE (MULTI-DIMENSIONAL)

- For **n-dimensions**, each dimension is different van der Corput sequence:

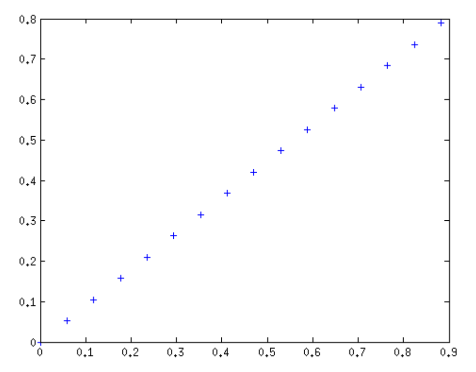
$$\mathrm{x}_i = (\Phi_2(i), \Phi_3(i), \dots, \Phi_{p_n}(i))$$

- Rate of converges for Monte Carlo integral evaluation is close to $O(N^{-\frac{n+1}{2n}})$, which is better than the random rate $O(N^{-\frac{1}{2}})$.

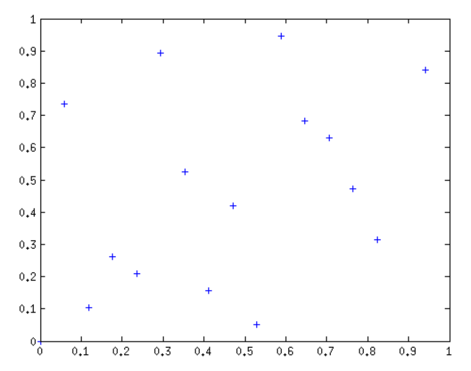
- The standard Halton sequences **perform very well in low dimensions**, however correlation problems have been noted between sequences generated from higher primes (degradation after 14 dimensions).

HALTON SEQUENCE (CURSE OF DIMENSIONALITY)

- For example if we start with the primes 17 and 19, the first 16 pairs of points would have perfect linear correlation!
  - To avoid this, it is common to drop the first few entries and/or take every other number in the sequence.
  - Or better, apply deterministic or random permutation on the digits of n, when forming $\Phi_b(n)$ (Scrambled Halton sequence).
  - Use the Sobol sequence, less correlation in higher dimensions! [Galanti & Jung '97]

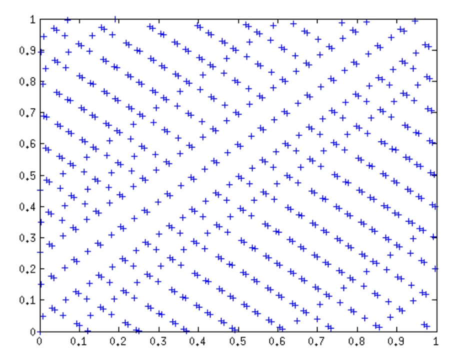


Standard Halton

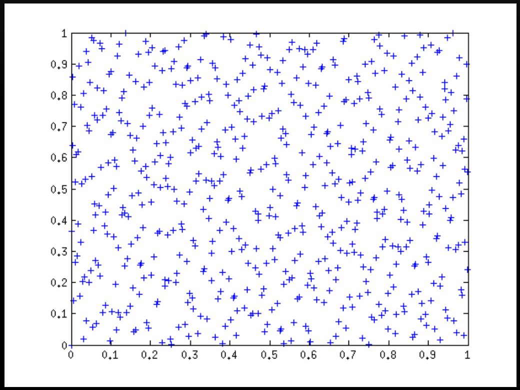


Scrambled Halton

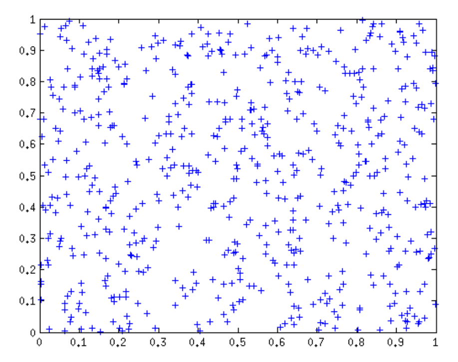
HALTON SEQUENCE (CURSE OF DIMENSIONALITY)



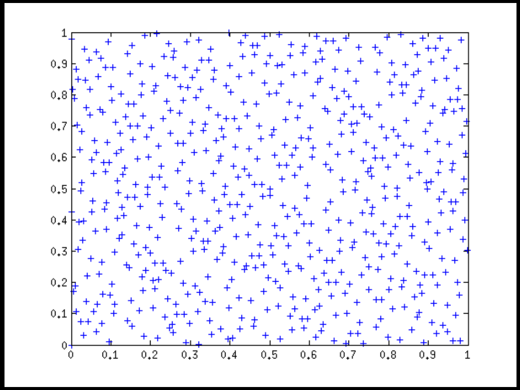
First 600 number of the standard Halton $(\Phi_{17}(i), \Phi_{19}(i))$



First 600 number of the scrambled Halton $(\Phi_{17}(i), \Phi_{19}(i))$



First 600 pair of pseudo-random numbers



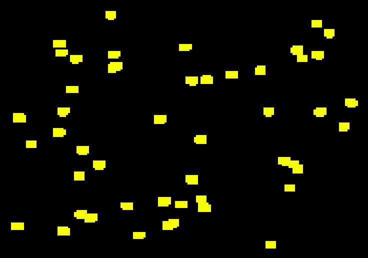
7th and 8th dimension of the 8-dimensional Sobol sequence

HAMMERSLEY SEQUENCE
(IN TWO DIMENSIONS)

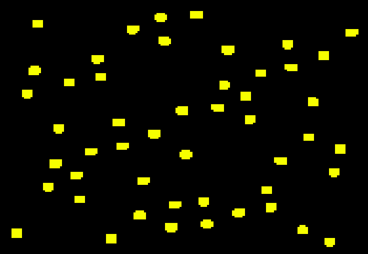
- Similar to Halton:

$$x_i = \left(\frac{i}{N}, \Phi_2(i)\right)$$

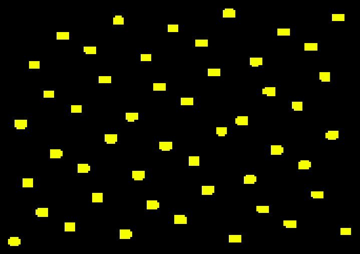
- Lower discrepancy than Halton.
- But need to know N, the total number of samples, in advance.



Random (Monte Carlo)



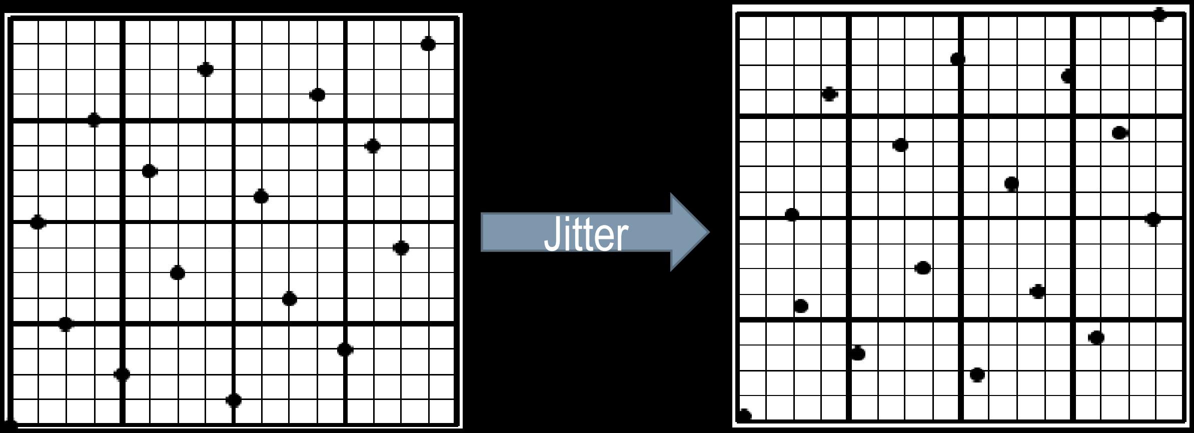
Halton Sequence
('Adaptive QMC')



Hammersley Sequence
('Constant QMC')

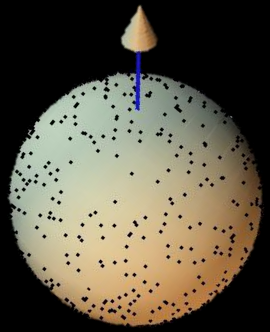
HAMMERSLEY SEQUENCE (STRUCTURE)

- The two-dimensional Hammersley sequence is aligned to a grid, which might lead to aliasing artifacts, so apply random jitter:



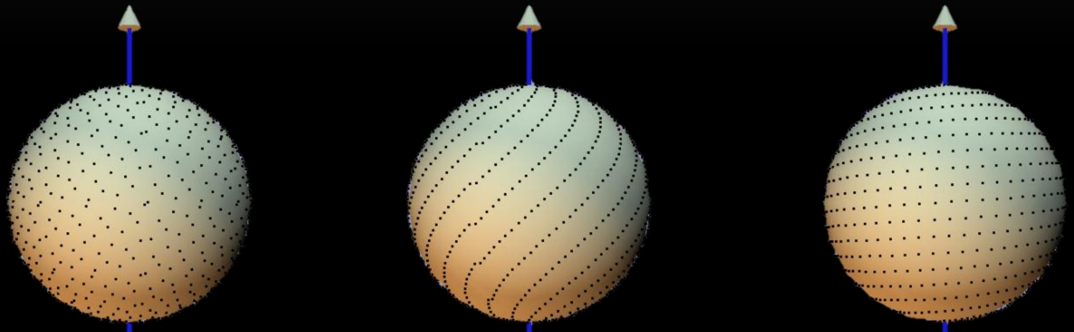
$$x_i = \left(\frac{i}{N}, \Phi_2(i) + \frac{\xi}{N}\right)$$

HAMMERSLEY SEQUENCE (LARGER BASIS)



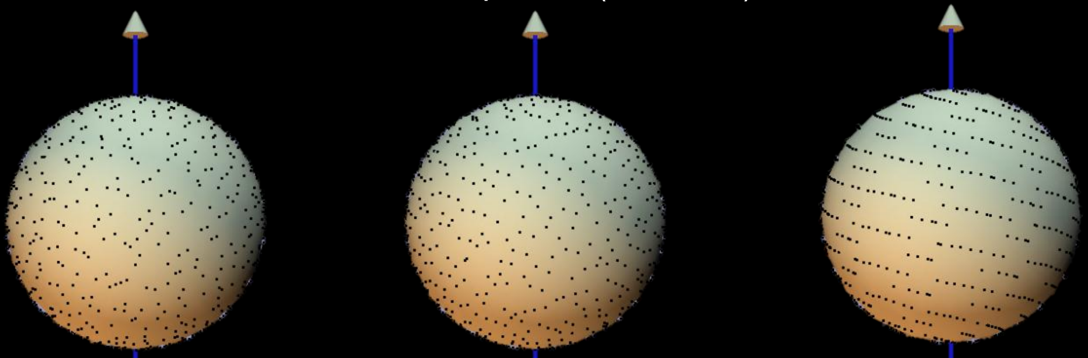
Random Points
(n = 1000)

Hammersley Sequence (*n = 1000*)



$\left(\frac{i}{N}, \Phi_2(i)\right)$ $\left(\frac{i}{N}, \Phi_7(i)\right)$ $\left(\frac{\mathbf{i}}{\mathrm{N}}, \Phi_{17}(\mathrm{i})\right)$

Halton Sequence (*n = 1000*)



$(\Phi_2(i), \Phi_3(i))$ $(\Phi_2(i), \Phi_7(i))$ $(\Phi_{17}(i), \Phi_{19}(i))$

[Wong JGT ’97]

LOW DISCREPANCY SAMPLING
AS USED IN THE IR PAPER

- **Use two-dimensional jittered Hammersley sequence for pixel super-sampling …**
  - as we usually use a predefined number of samples there.

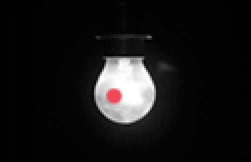
- **Use multi-dimensional Halton sequences during the quasi-random walk …**
  - as we might need more adaptive control (different number of samples).
  - watch out for degradation when the dimension is large (aka. large number of bounces)!

QUASI-RANDOM WALK USING HALTON SEQUENCES

- Each path (i) is characterized by the Halton sequence:

$$\left(\Phi_2(i), \Phi_3(i), \dots, \Phi(i)_{p_{2j+2}},\ \Phi(i)_{p_{2j+3}}, \dots, \Phi(i)_{p_{2l+3}}\right)$$

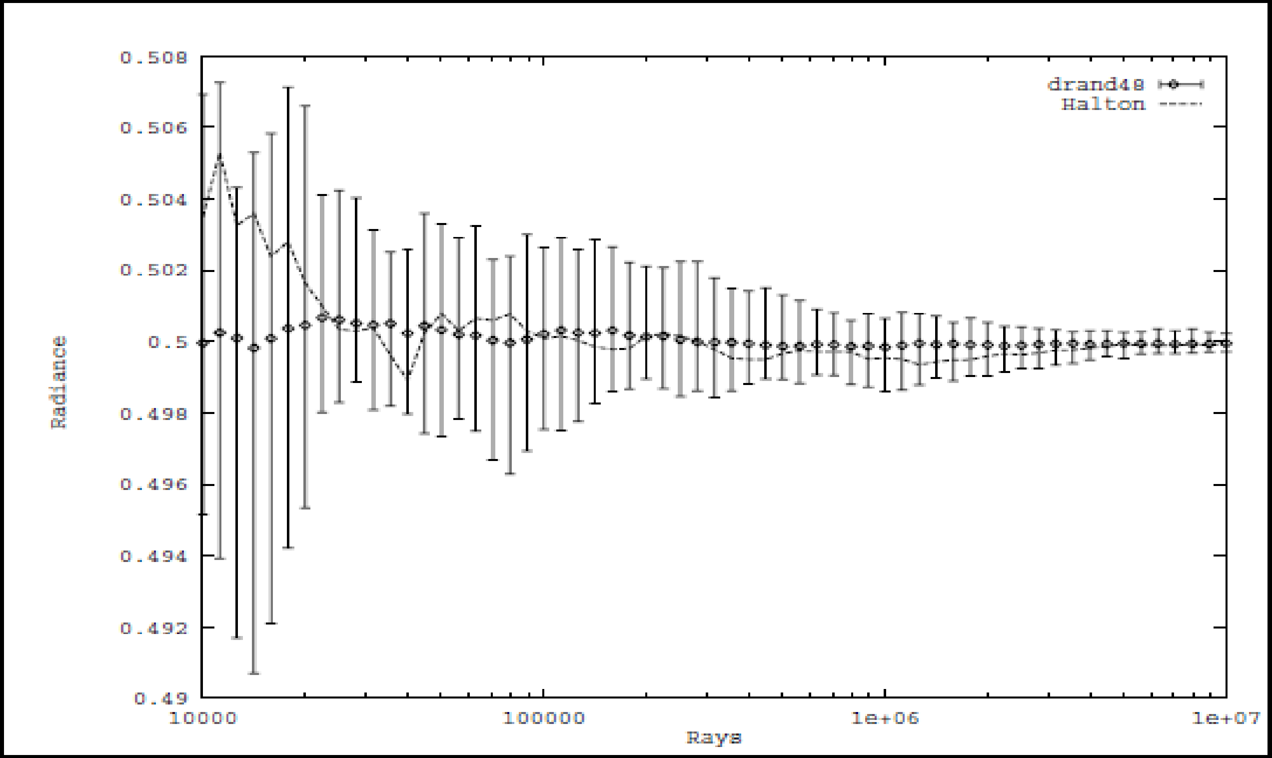
- Use $\mathrm{y} = \mathrm{y}_0(\Phi_2(i), \Phi_3(i))$ to sample starting point on the luminaire for path i.



- Use $\omega_\mathrm{j} = \left(\arcsin\sqrt{\Phi(i)_{p_{2j+2}},}\ 2\pi\Phi(i)_{p_{2j+3}}\right)$ to sample new directions for path i after j bounces.

HOW MUCH DOES THIS HELP ?

- Not shown for the Instant Radiosity method.
- Previous Keller's paper "Quasi-Monte Carlo Radiosity" gives some intuition:



ANTI-ALIASING USING HAMMERSLEY SEQUENCE

- Anti-aliasing with the Accumulation Buffer.

- A super-sampling technique is used where the entire scene is offset by small, sub-pixel amounts in screen space, and accumulated.
  - Just translate the projection matrix in x and y and re-render!

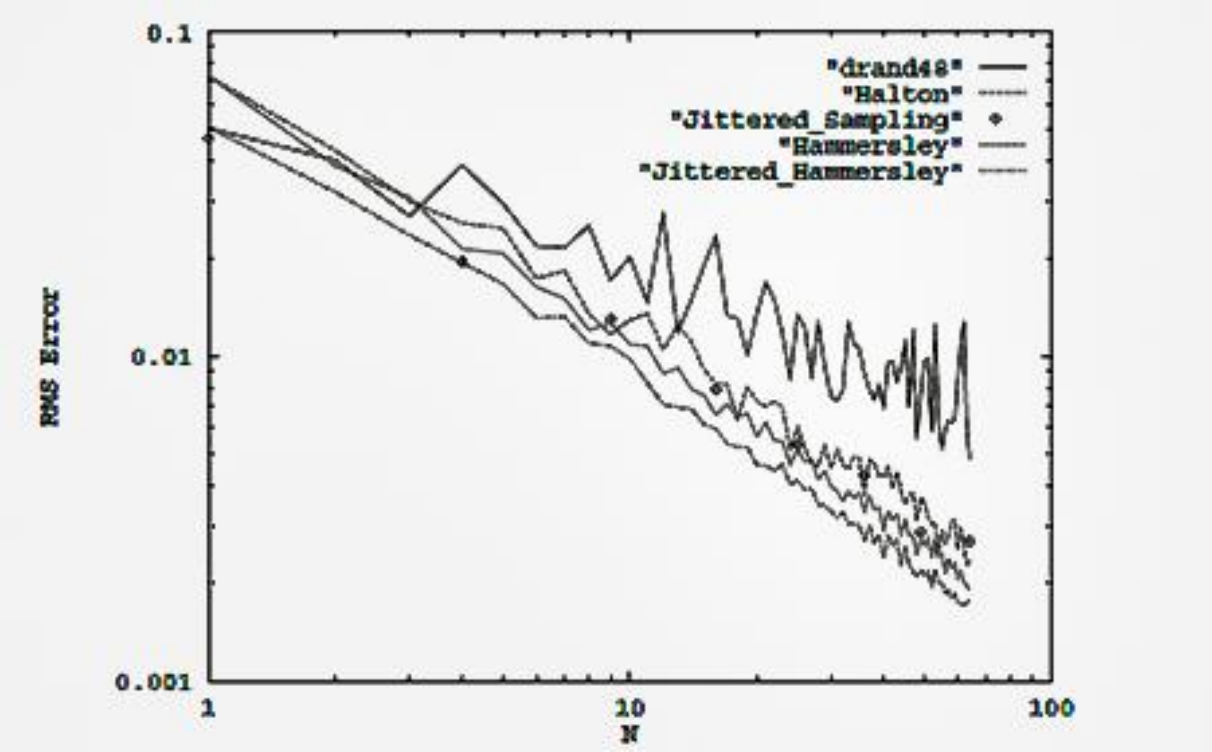
- The offset is determined by the jittered-Hammersley sequence (N is the number of lights in the scene, and x_i is the offset for the i-th VPL rendering):

$$x_i = \left(\frac{i}{N}, \Phi_2(i) + \frac{\xi}{N}\right)$$

- Hammersley numbers are suitable, as we have low-dimensional data with pre-defined number of samples!

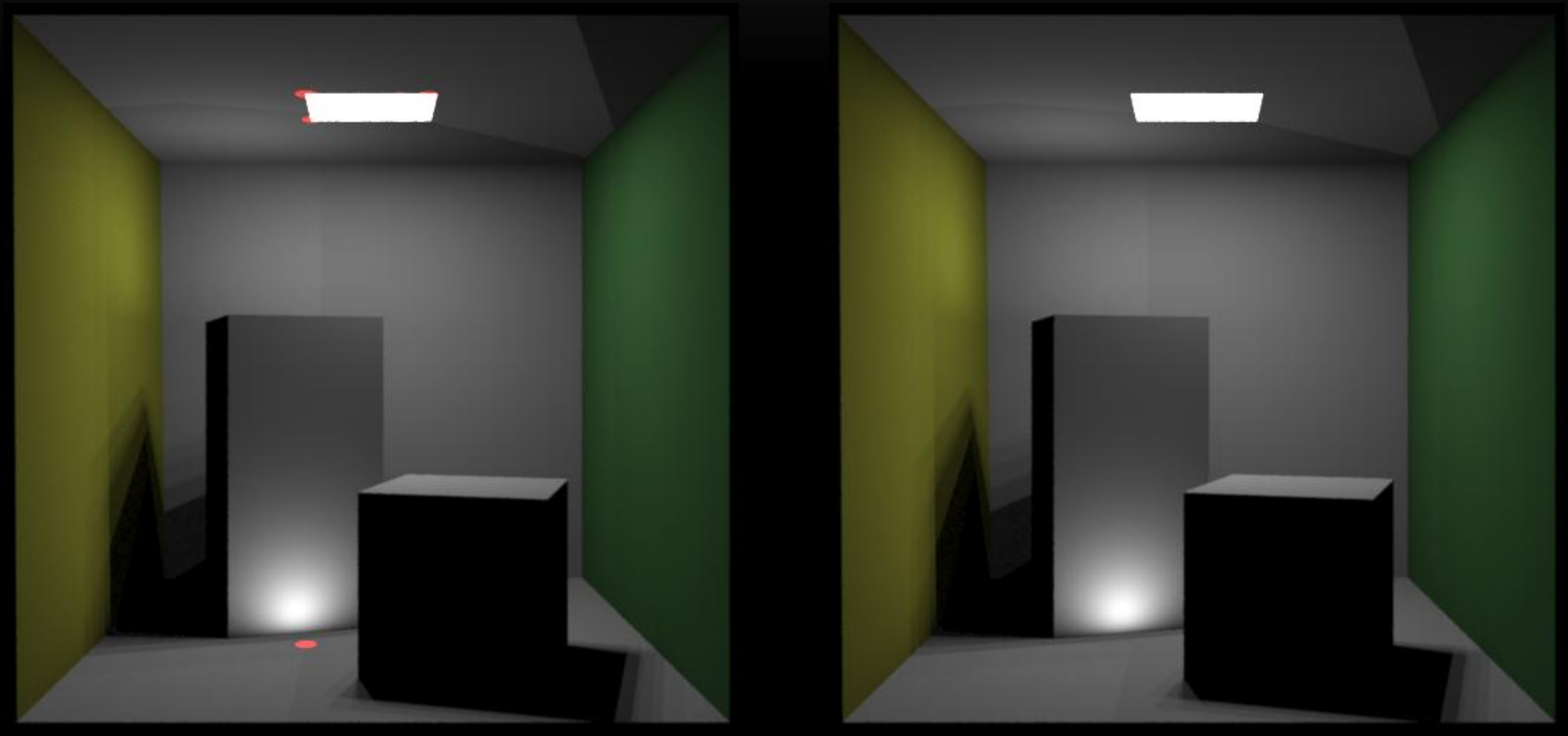
HOW MUCH DOES THIS HELP ?

- The two-dimensional jittered Hammersley sequence exposes faster convergence rates, when used for pixel super-sampling.

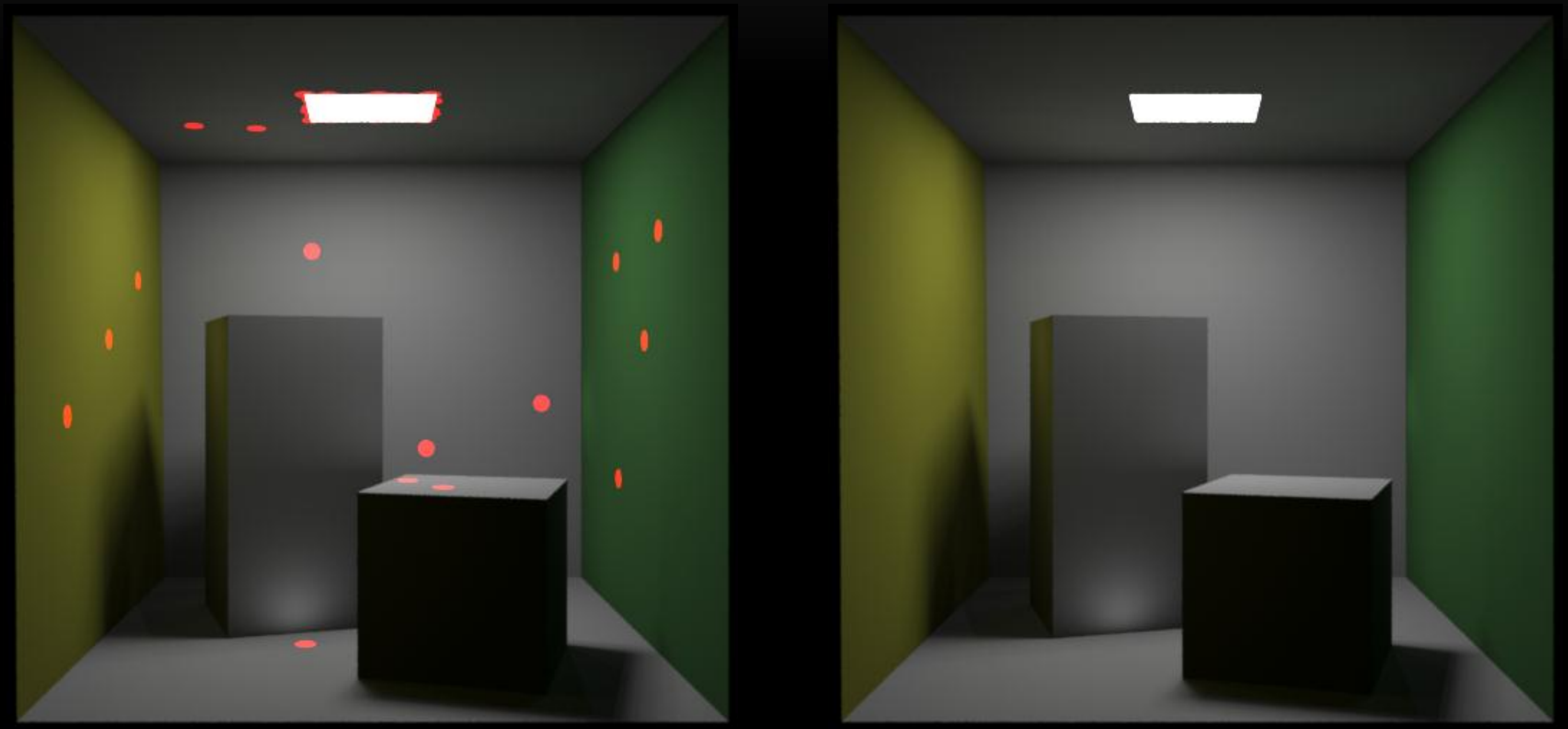


RESULTS

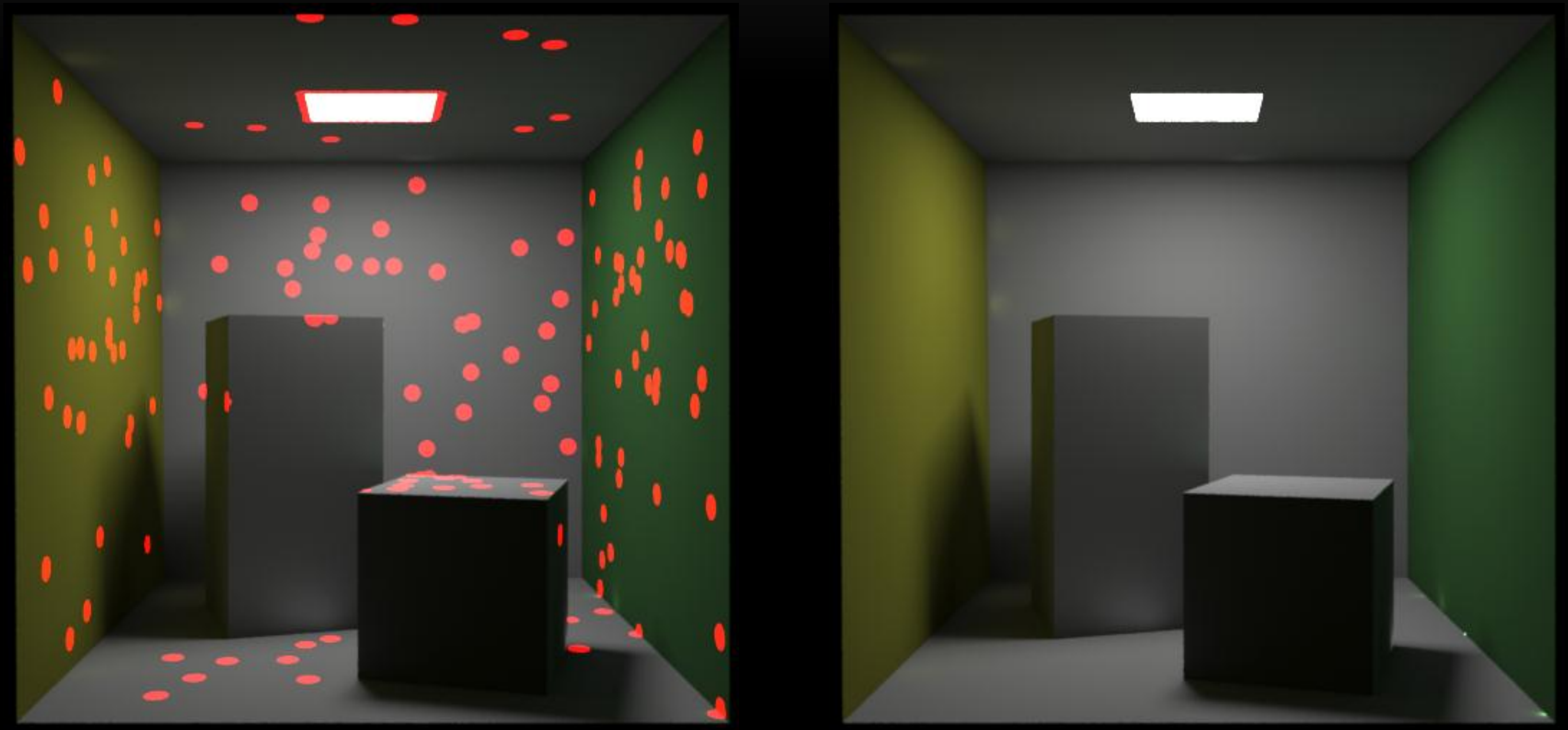
10 SAMPLES



100 SAMPLES

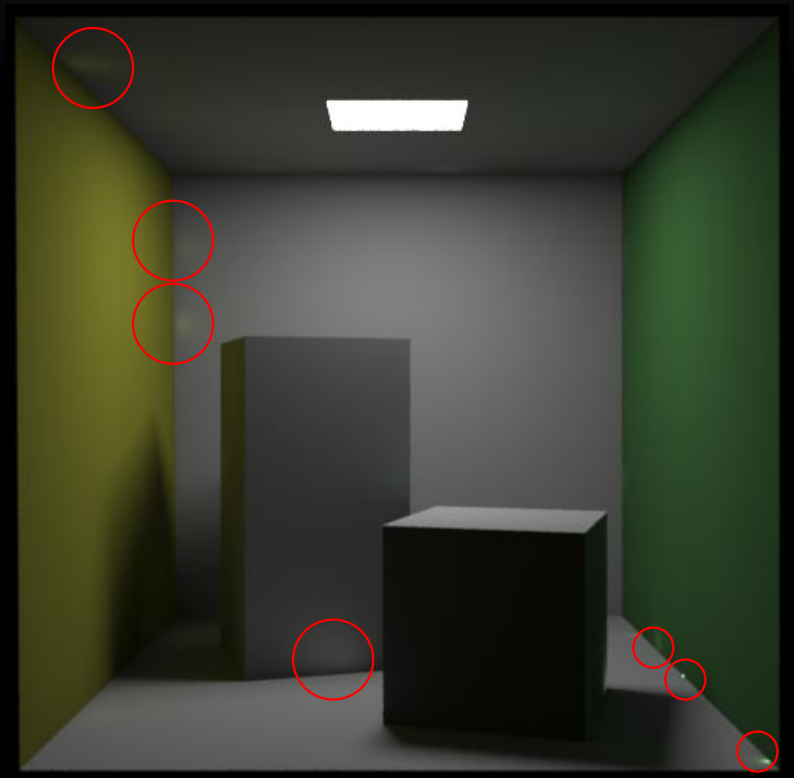


1000 SAMPLES



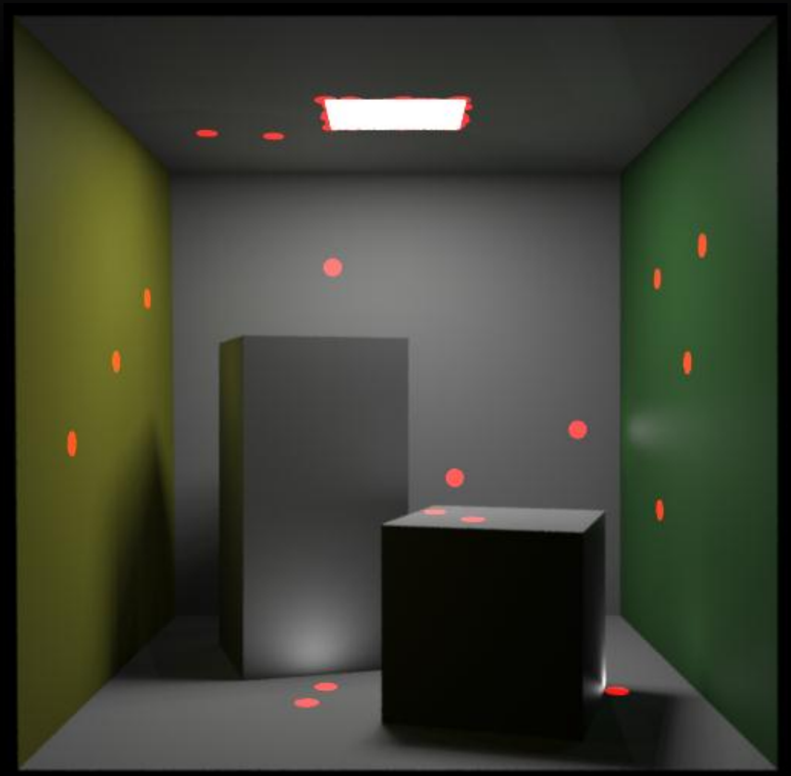
ARTIFACTS

- Unlike path tracing, not noise
- Structured hotspots
- Singularity in form factor
- Hack: clamp sample contribution
  - No longer unbiased
  - Loss of energy around edges

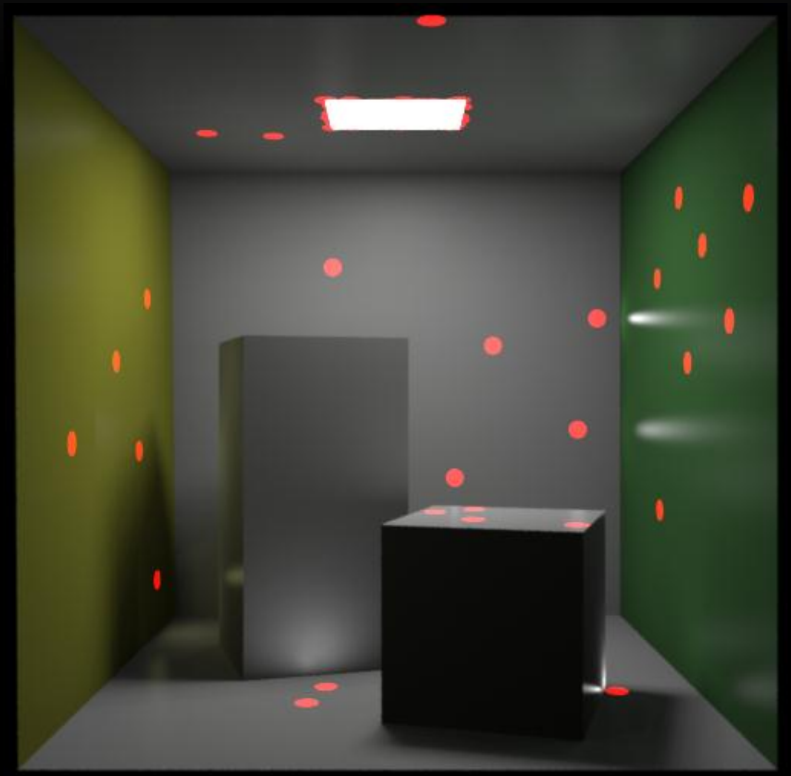


$$L(x' \to x'') = \frac{k_d(x')}{\pi} \sum_{x \in VPLs} V(x \leftrightarrow x') \frac{\cos(\theta_r)\cos(\theta_i')}{\boxed{||x - x'||^2}} L_i(x \to x')$$

GLOSSY BRDF



$\alpha = 0.25$



$\alpha = 0.1$

MODERN WORKS

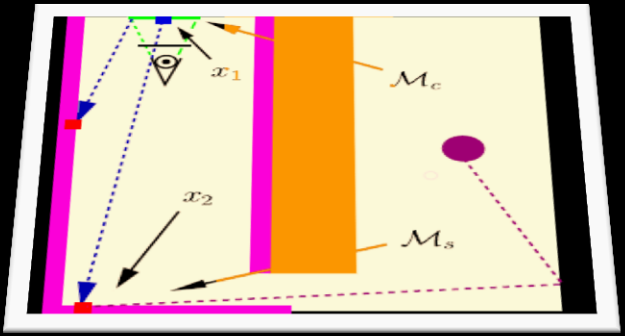
BIDIRECTIONAL INSTANT RADIOSITY

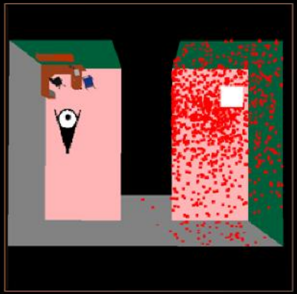
- Optimize the location of the VPLs, by finding locations which have influence on the illumination of the scene rendered from the camera.

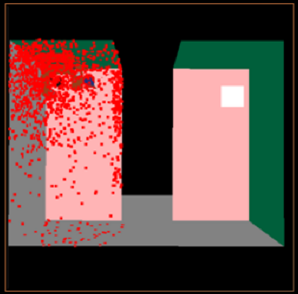
I. First, trace rays from the camera.

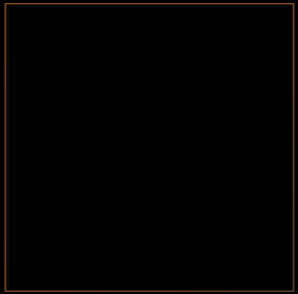
II. Second, path vertices of length 2 form the set of reverse VPL candidates.

III. Finally, connect reverse VPL points with the standard VPL points.




Standard IR VPLs


Reverse IR VPLs

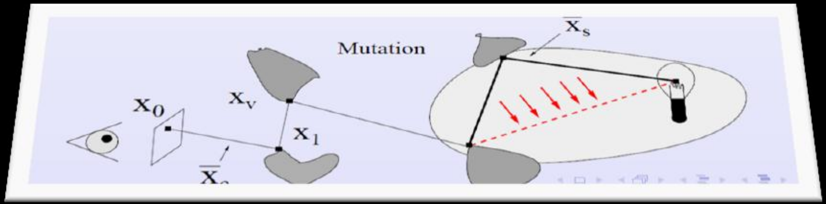

Standard IR Result


Bidirectional IR Result

Segovia et al. (*ESR '2006)*

METROPOLIS INSTANT RADIOSITY

- We must find VPLs which illuminate parts of the scene, seen by the camera.
  1. First, use the standard sequence of Metropolis Light Transport to sample VPLs (MLT part).
  2. Second, for each path, store the second point as a VPL.
  3. Accumulate all VPL contributions (IR part)



Standard IR

Bidirectional IR

Metropolis IR

Segovia et al. (EUROGRAPHICS '2007)

VPL based approaches are as good as the number of generated point lights.

Can we use millions of VPLs
in reasonable amount of time ?

Yes, Lightcuts!

QUESTIONS?