CS6630 Realistic Image Synthesis

Monte Carlo illumination

Steve Marschner

Fall 2022

Monte Carlo illumination overview

Estimating illumination

- BRDF sampling
- Light source sampling

Variance reduction

- importance sampling
- stratified sampling
- blue noise / quasi MC
- sampling in software

Multiple Importance Sampling

- basic idea of MIS
- MIS with area sources

Convergence rate

- **We can get a better estimate of the expected value of *g* by generating several values and averaging them.**

$$G_n = \frac{1}{N}\sum_{i=1}^{n} g(x_i) \quad \text{where } x_i \sim p$$

- **As *n* increases, the variance of G_n decreases**

$$\sigma^2\left\{\sum_{i=1}^{n} g(x_i)\right\} = \sum_{i=1}^{n} \sigma^2\{g\} = N\sigma^2\{g\}$$

$$\sigma\{G_n\} = \frac{\sigma\{g\}}{\sqrt{N}}$$

Illumination with uniform sampling

- **If we select directions uniformly over the hemisphere…**
  - then:

    $$p(\omega_i) \sim 1/(2\pi)$$

    (see notebook for how…)
  - 2π because that is the area (solid angle) of the hemisphere; that way, probability integrates to 1
  - the correct estimator is:

$$g(\omega_i) = \frac{L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}|}{p(\omega_i)}$$
$$= 2\pi L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}|$$

cosine-proportional sampling

- **If we select directions proportional to** $|\omega_i \cdot \mathbf{n}|$ (see notebook for how…)
  - then:

$$p(\omega_i) \sim |\omega_i \cdot \mathbf{n}|/\pi$$

  - factor of π needed so that probability integrates to 1
  - the correct estimator is:

$$g(\omega_i) = \frac{L_i(\omega_i) f_r(\omega_i, \omega_r) |\omega_i \cdot \mathbf{n}|}{p(\omega_i)}$$
$$= \pi L_i(\omega_i) f_r(\omega_i, \omega_r)$$

Rendering any surface, env. lighting

- **Start with integral**

$$L_r = \int_{S^2_+} L_i(\mathbf{w}) \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n}) \, d\sigma(\mathbf{w})$$

  - one choice of pdf: proportional to BRDF

$$f(\mathbf{w}) = L_i(\mathbf{w}) \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n}) \qquad p(\mathbf{w}) = \frac{f_r(\mathbf{v}, \mathbf{w})}{M(\mathbf{v})} \qquad g(\mathbf{w}) = \frac{f(\mathbf{w})}{p(\mathbf{w})} = M(\mathbf{v}) L(\mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n})$$

  - another choice: proportional to environment brightness

$$f(\mathbf{w}) = L_i(\mathbf{w}) \cdot f_r(\mathbf{v}, \mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n}) \qquad p(w) = \frac{1}{M} L_i(\mathbf{w}) \qquad g(\mathbf{w}) = \frac{f(\mathbf{w})}{p(\mathbf{w})} = M f(\mathbf{v}, \mathbf{w}) \cdot (\mathbf{w} \cdot \mathbf{n})$$

Code: any surface / environment light

```
Color shade(x, V, brdf, N) {
  result = black;
  w, p_w = environment.sample(N);
  if !shadow(x, w) {
    L_env = environment.eval(w)
    f_r = brdf.eval(V, w);
    result += L_env * f_r
      * dot(w, N) / p_w;
  }
  return result;
}
```

sampling by lighting environment

```
Color shade(x, V, brdf, N) {
  result = black;
  w, p_w = brdf.sample(N);
  if !shadow(x, w) {
    L_env = environment.eval(w)
    f_r = brdf.eval(V, w);
    result += L_env * f_r
      * dot(w, N) / p_w;
  }
  return result;
}
```

sampling by BRDF

Rendering any surface, area light sampling

- **Start with integral**

$$L_r = \int_S L_s \cdot f_r(\mathbf{v}, \mathbf{w}(\mathbf{y})) \cdot \frac{|\mathbf{n_x} \cdot \mathbf{w}(\mathbf{y})|\,|\mathbf{n_y} \cdot \mathbf{w}(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|^2} dA(\mathbf{y})$$

  - choose a probability density on S: uniform

$$f(\mathbf{y}) = L_s \cdot f_r(\mathbf{v}, \mathbf{w}(\mathbf{y})) \cdot \frac{|\mathbf{n_x} \cdot \mathbf{w}(\mathbf{y})|\,|\mathbf{n_y} \cdot \mathbf{w}(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|^2}$$

$$p(\mathbf{y}) = \frac{1}{A}$$

$$g(\mathbf{y}) = \frac{f(\mathbf{y})}{p(\mathbf{y})} = L_s A \cdot f_r(\mathbf{v}, \mathbf{w}(\mathbf{y})) \cdot \frac{|\mathbf{n_x} \cdot \mathbf{w}(\mathbf{y})|\,|\mathbf{n_y} \cdot \mathbf{w}(\mathbf{y})|}{\|\mathbf{x} - \mathbf{y}\|^2}$$

Code: any surface / area light sampling

```
Color shade(x, V, brdf, N) {
  result = black;
  for light in lights {
    y, p_y = light.sample(x);
    if !shadow(x, y) {
      L = normalize(y - x);
      f_r = brdf.eval(V, L);
      result += light.radiance * f_r
        * dot(L, N) * dot(-L, light.normal)
        / distSqr(x, y)
        / p_y;
    }
  }
  return result;
}
```


Area light by solid angle sampling

- **Start with integral**

$$L_r = \int_{S^2_+} L_i(\mathbf{w}) f_r(\mathbf{v}, \mathbf{w}) \, |\, \mathbf{w} \cdot \mathbf{n_x} \,| \, d\sigma(\mathbf{w})$$

  - choose a direction by choosing a point on the light

$$p(\mathbf{y}) = \frac{1}{A}$$

$$\mathbf{w} = \frac{\mathbf{y} - \mathbf{x}}{\|\mathbf{y} - \mathbf{x}\|} \qquad p(\mathbf{w}) = \frac{\|\mathbf{y} - \mathbf{x}\|^2}{|\, \mathbf{w} \cdot \mathbf{n_y} \,|} p(\mathbf{y})$$

$$g(\mathbf{w}) = \frac{f(\mathbf{w})}{p(\mathbf{w})} = L_s \, f_r(\mathbf{v}, \mathbf{w}) \, |\, \mathbf{w} \cdot \mathbf{n_x} \,| \frac{|\, \mathbf{w} \cdot \mathbf{n_y} \,|}{\|\mathbf{y} - \mathbf{x}\|^2} A$$

$$= L_s A \, f_r(\mathbf{v}, \mathbf{w}) \frac{|\, \mathbf{w} \cdot \mathbf{n_x} \,| \, |\, \mathbf{w} \cdot \mathbf{n_y} \,|}{\|\mathbf{y} - \mathbf{x}\|^2}$$

same estimator as area sampling
so code is the same!

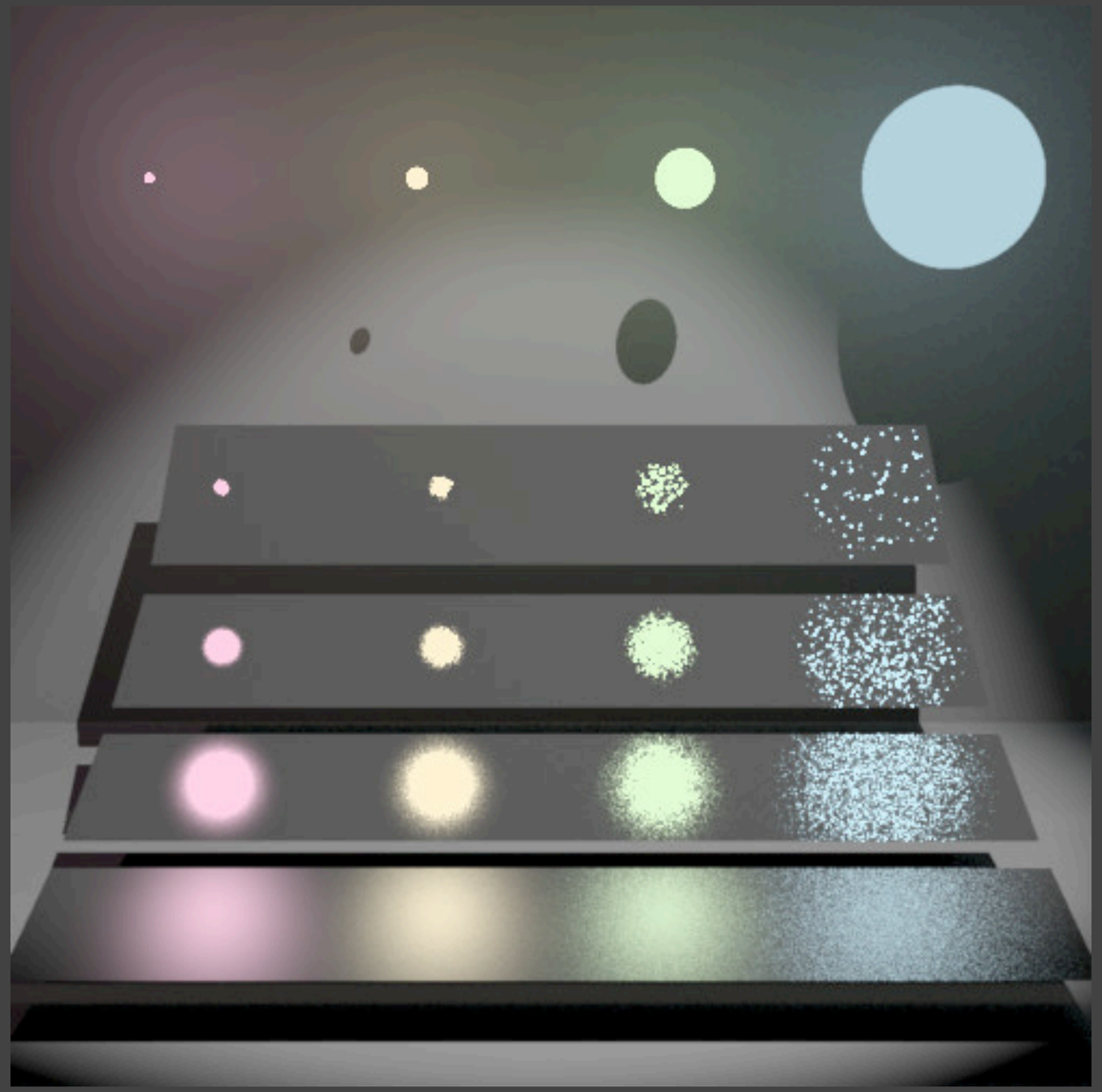
Code: any surface / area light

```
Color shade(x, V, brdf, N) {
  result = black;
  y, p_y = light.sample(N);
  if !shadow(x, y) {
    w = normalize(y - x);
    p_w = p_y * distSqr(x, y) / dot(-L, light.normal)
    f_r = brdf.eval(V, w);
      result += light.radiance * f_r
        * dot(w, N) / p_w;
  }
  return result;
}
```

sampling by area on light

```
Color shade(x, V, brdf, N) {
  result = black;
  w, p_w = brdf.sample(N);
  if (y = light.intersect(x, w)) and !shadow(x, y) {
    L = light.radiance
    f_r = brdf.eval(V, w);
    result += L_env * f_r
      * dot(w, N) / p_w;
  }
  return result;
}
```

sampling by BRDF



Veach thesis, 1997

sampling the luminaires

Veach thesis, 1997

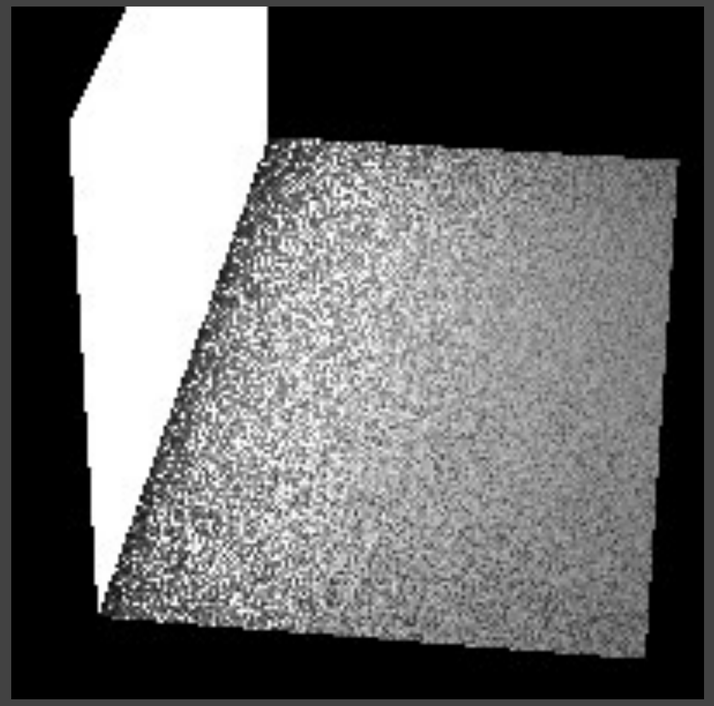
sampling the BRDF

Veach thesis, 1997

sampling sum of pdfs (balance heuristic)

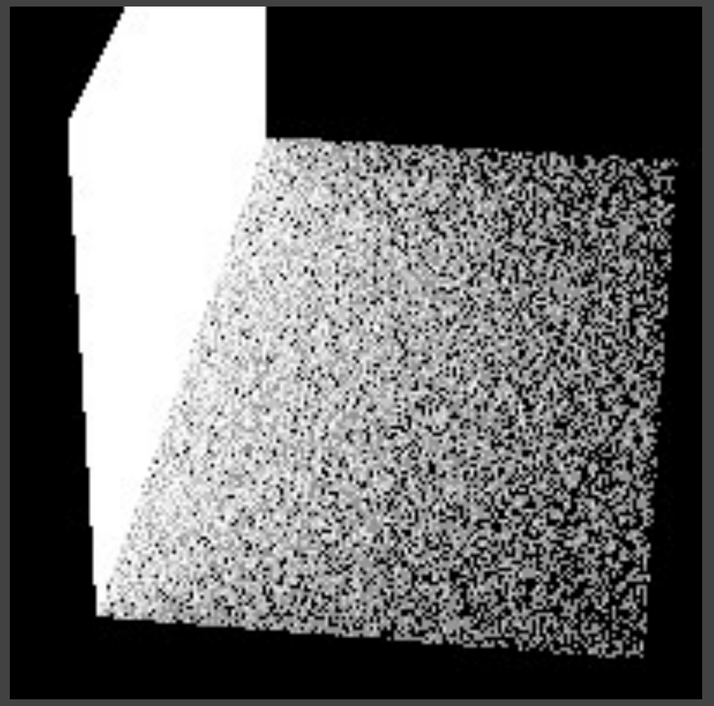
Veach thesis, 1997

combining samples with power heuristic



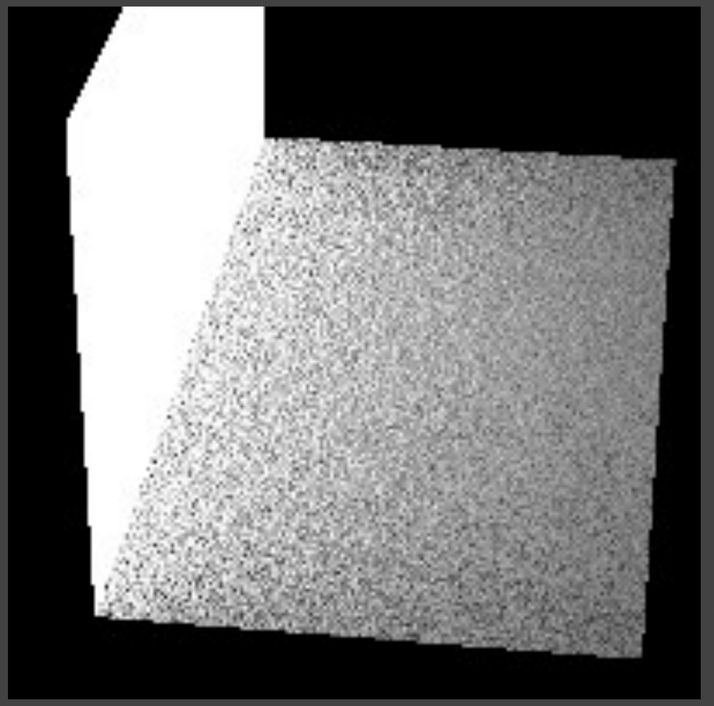
source area sampling

r^{-2} term causes high variance at left



cosine sampling

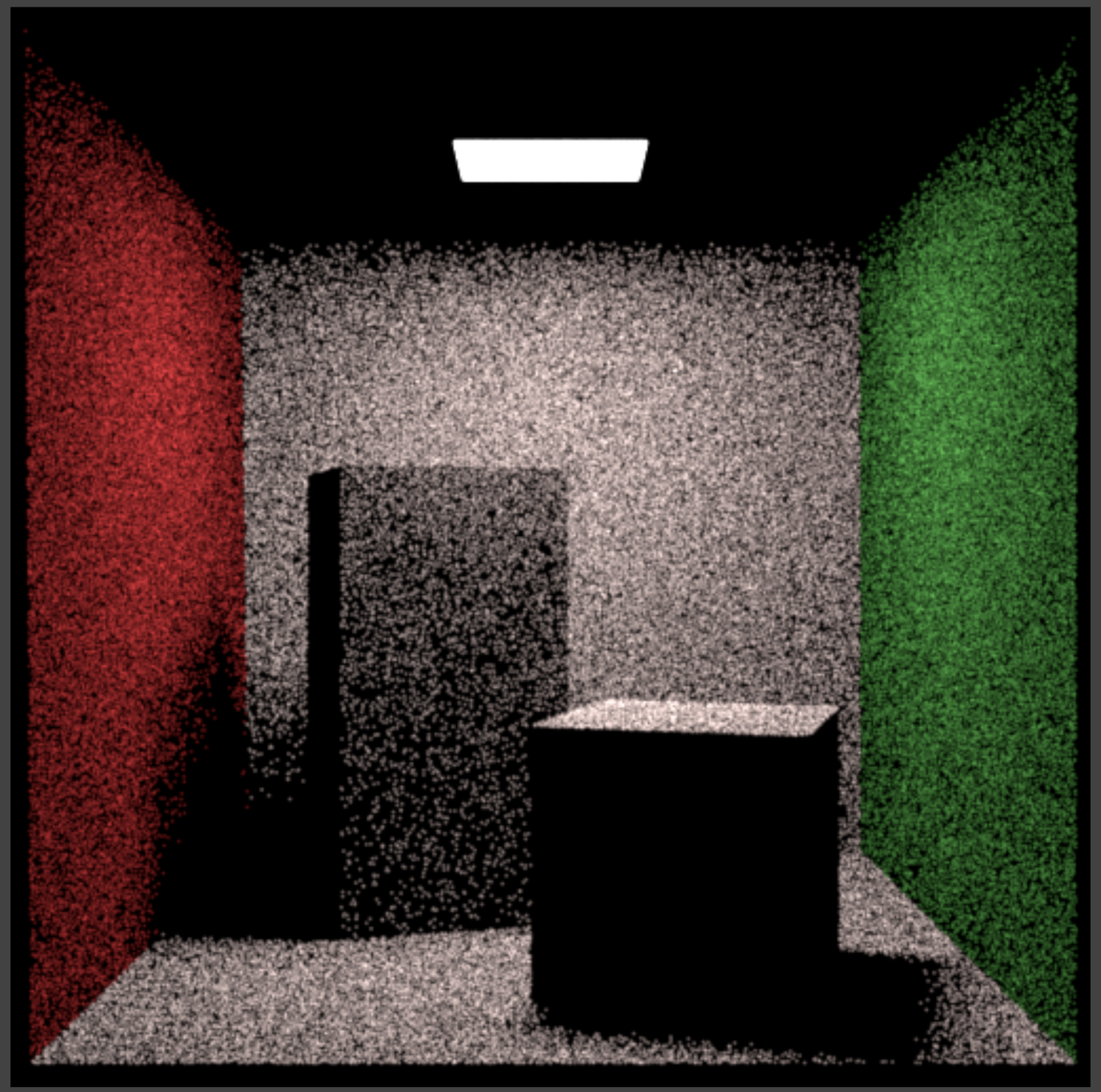
source solid angle causes high variance at right



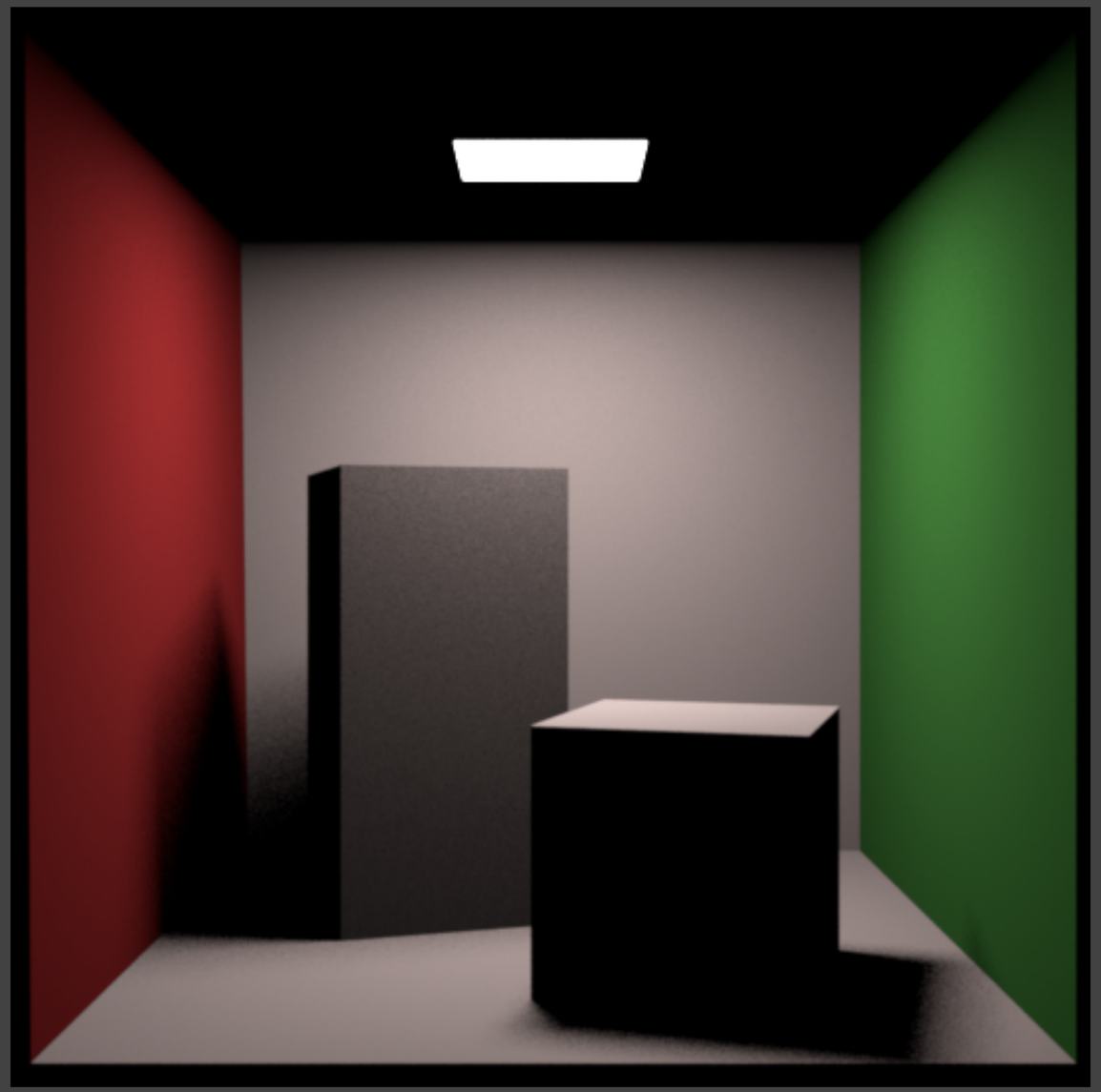
MIS

moderate variance everywhere

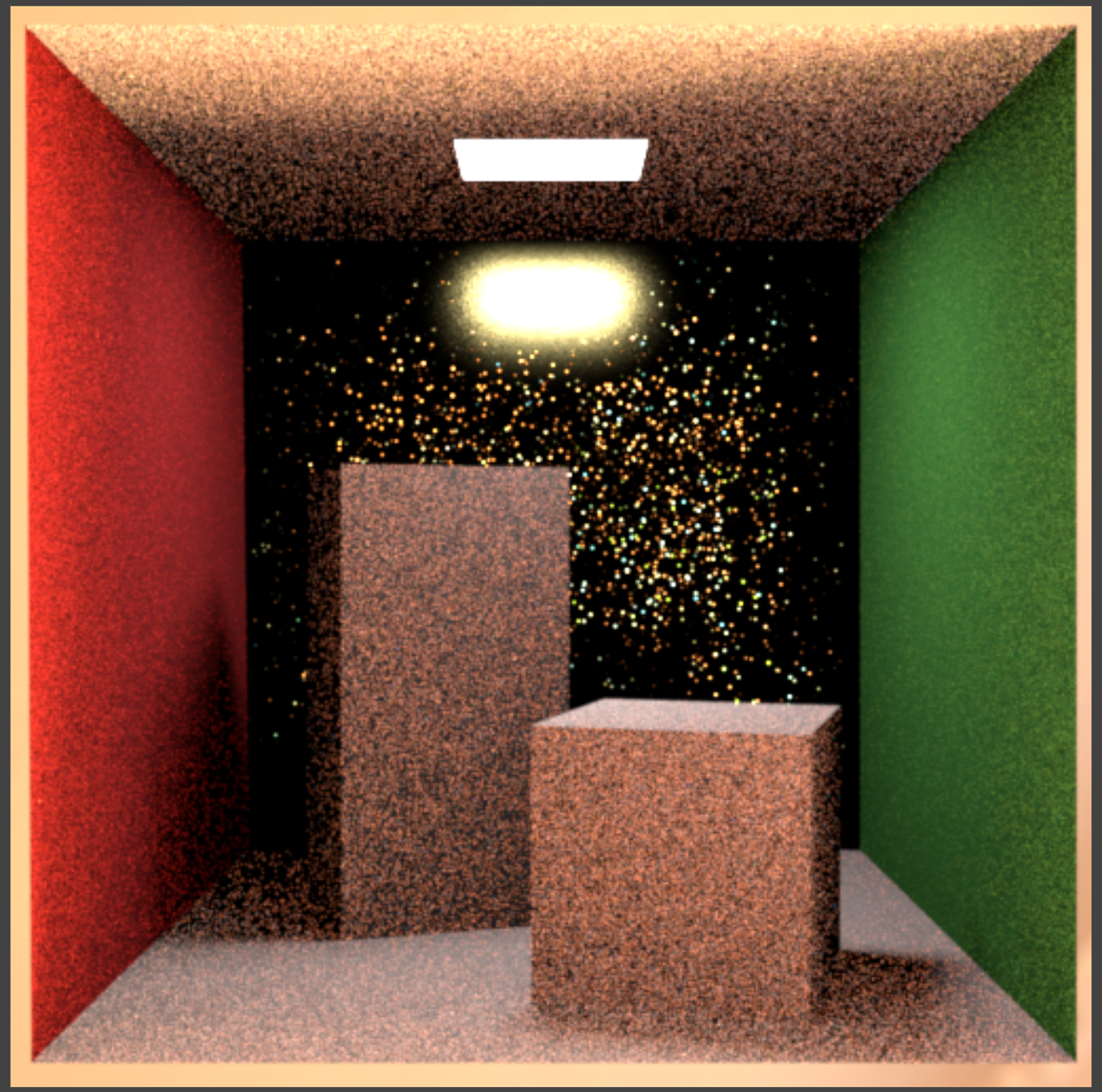
Veach thesis, 1997



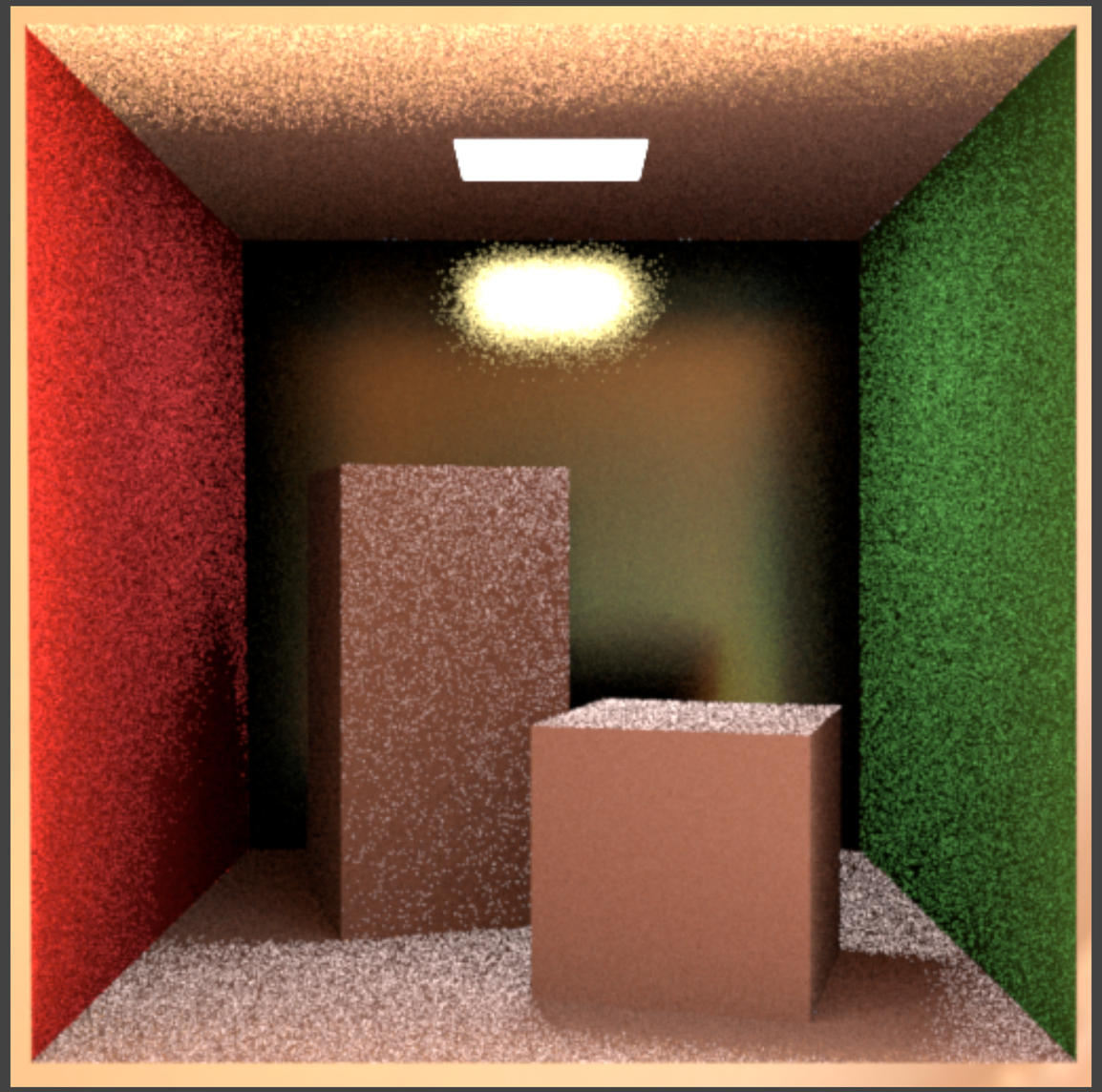
sampling the BRDF



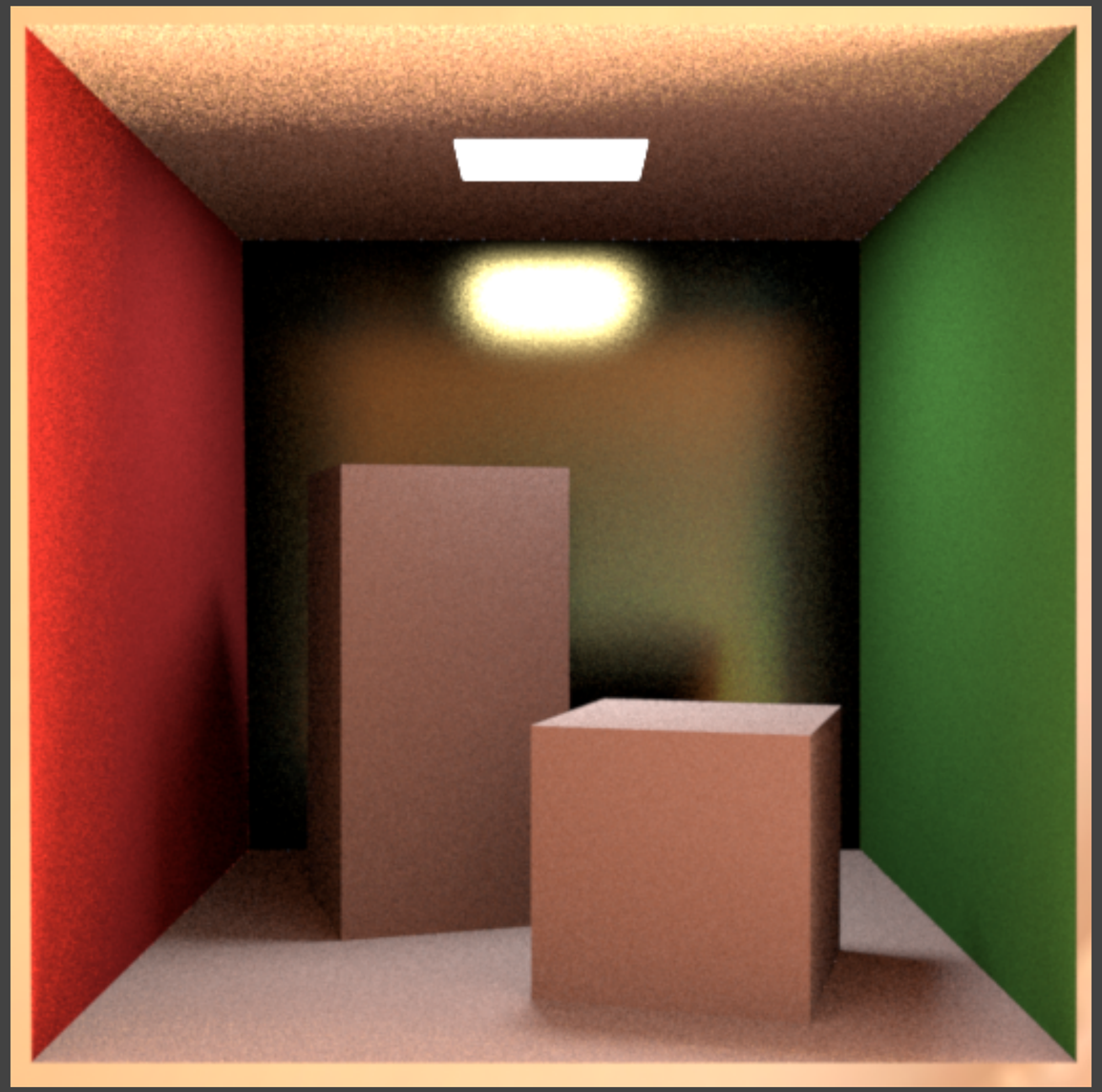
sampling the luminaires



sampling the luminaires



sampling the BRDF



combining samples with power heuristic

Code: multiple importance sampling

```
Color shade(x, V, brdf, N) {
  result = black;
  w, p_w = environment.sample(N);
  if !shadow(x, w) {
    L_env = environment.eval(w)
    f_r = brdf.eval(V, w);
    result += L_env * f_r
      * dot(w, N) / p_w;
  }
  return result;
}
```

sampling by lighting environment

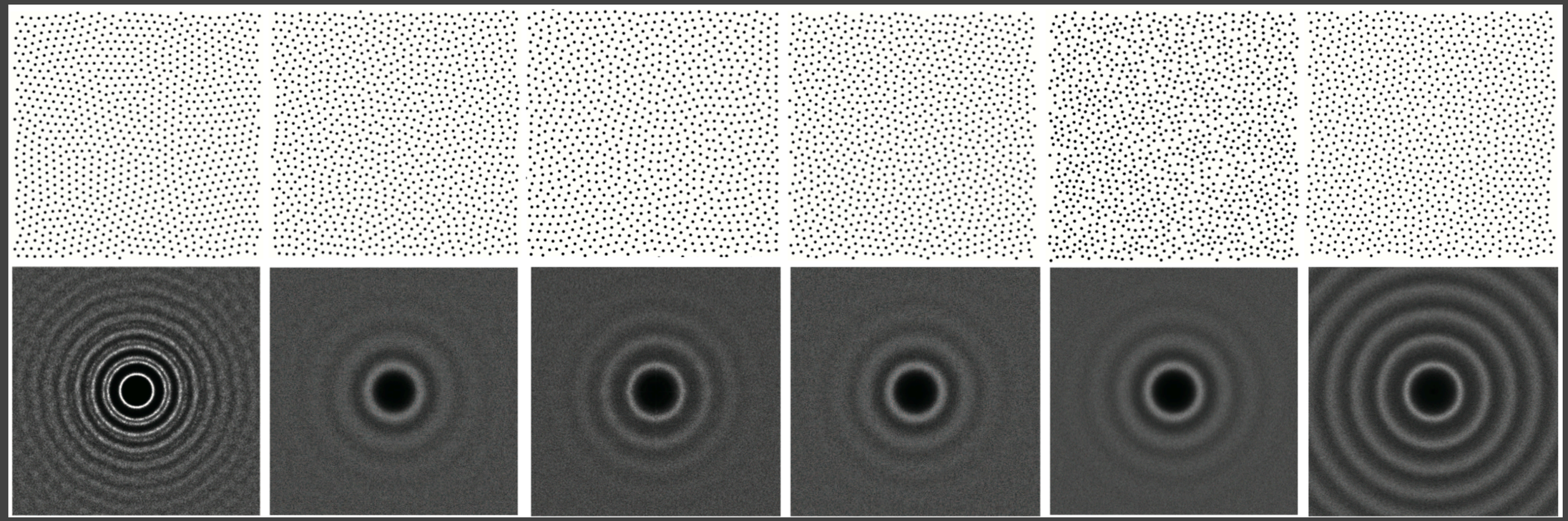
```
Color shade(x, V, brdf, N) {
  result = black;
  w, p_w = brdf.sample(N);
  if !shadow(x, w) {
    L_env = environment.eval(w)
    f_r = brdf.eval(V, w);
    result += L_env * f_r
      * dot(w, N) / p_w;
  }
  return result;
}
```

sampling by BRDF

```
Color shade(x, V, brdf, N) {
  result = black;
  wl, p_wll = environment.sample(N);
  p_wlb = brdf.pdf(N, wl)
  wb, p_wbb = brdf.sample(N);
  p_wbl = environment.pdf(N, wb)
  if !shadow(x, wl) {
    L = environment.eval(wl)
    f_r = brdf.eval(V, wl);
    result += L * f_r
      * dot(wl, N) / (p_wll + p_wlb);
  }
  if !shadow(x, wb) {
    L = environment.eval(wb)
    f_r = brdf.eval(V, wb);
    result += L * f_r
      * dot(wb, N) / (p_wbl + p_wbb);
  }
  return result;
}
```

multiple importance with 2 samples

Blue noise point sets



Dong-Ming Yan et al. 2015

Quasi Monte Carlo sequences

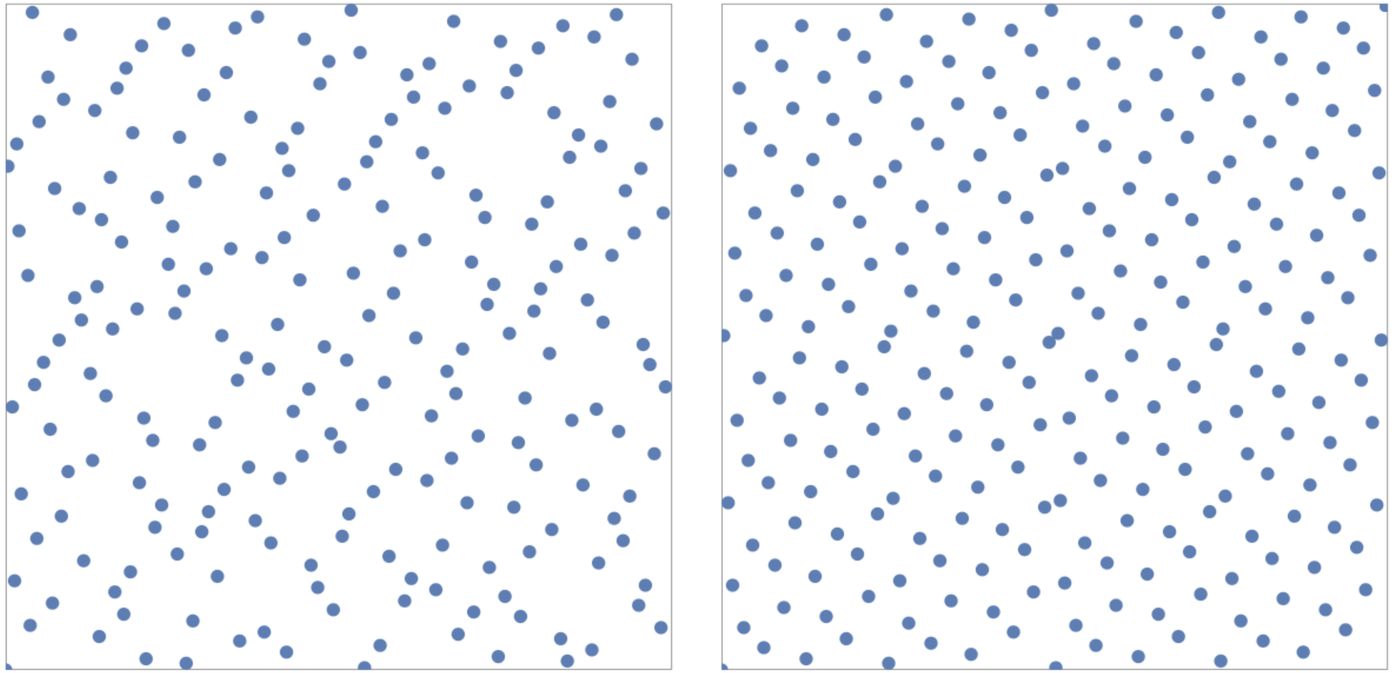


Figure 7.25: The First Points of Two Low-Discrepancy Sequences in 2D. (a) Halton (216 points), (b) Hammersley (256 points).