CS 419: Production Rendering

Introduction to
Aliasing and Sampling

Eric Shaffer
Aliasing is an effect caused by discrete sampling.

With digital images:
- We have a finite number of pixels
- We have a finite number of colors
- ...which will not always be able to render a scene accurately

Some common aliasing phenomena are:
- jaggies
- moire patterns
- loss of small details in textures
Mario...

Filthy Jagged Original Emulation

Glorious Anti-Aliased PC Emulation
Minecraft with anti-aliasing...

...maybe not as easy to see in a world made of blocks
Imagine a yellow polygon in a scene.

We have a 5x7 array of pixels and shoot rays through the pixel centers.

We get a jagged edge....

How could we generate a better approximation?
Moire patterns

\[ f(x, y) = \frac{1}{2} (1 + \sin(x^2 y^2)) \]

Second images show what happens when we try to squeeze a bigger domain into the same 512x512 pixels

As an aside....how do you ray cast \( f(x,y) \) ?
**Anti-aliasing**

- One remedy to aliasing is to shoot more rays per-pixel.

- We use an $n$ by $n$ regular sub-grid and shoot through the sub-pixels.

- Color is the average color returned by the sub-samples.

- Image shows one pixel with 25 sub-samples.
A Ray Tracing Example

One ray per pixel          16 rays per pixel             Enlarged view
Problems with Regular Sampling

- Still often leads to “regular” artifacts
- Humans are great at perceiving induced regular patterns
Random Sampling

- Could use N random locations in the pixel
- Often makes things look noisy, but...
- ...people prefer noise to aliasing visually
Jittered Sampling

- Use a regular grid of n by n subpixels
- Use random location within each subpixel
Pixels on the horizon cover an infinite area
Projected size of a square becomes infinitely small at horizon
We cannot sample enough to preserve infinite detail
Filtering

- In ray-tracing this means using rays outside pixel boundary as well as inside to generate the color.
- The weight scheme of the samples determines the type of filter:
  - Box: All rays equally weighted.
  - Tent: Sample importance decreases linearly away from center.
  - Cubic: Importance decrease as cubic polynomial.
  - Gaussian: Importance decreases exponentially.
Filters

Box Filter

Gaussian Filter
Some Quick Definitions

- BRDF: Bidirectional reflectance distribution function
  - 4D function modeling light reflected at an opaque surface.

- BTDF: Bidirectional transmittance distribution function
  - used for subsurface scattering among other things
Sampling

- We need to sample and reconstruct lots of things
  - For depth of field you need to model and sample a finite area lens
  - Area lights and soft shadows require sampling the light surfaces
  - For glossy reflection and transmission you need to sample BRDFs & BTRFs
2D Sampling

- Assume we are sampling a function on a unit square
- Good sampling
  - 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
- Ultimately we want a sampling pattern that
  - Generates a quality result with a minimum number of samples
  - …i.e. approximation converges more quickly…less rendering time
Random

- Fails

*Figure 5.8. (a) 16 random samples with x- and y-projections; (b) 256 random samples.*
Jittered

- Example of *stratified* sampling
- Significantly better than random

Figure 5.9. (a) 16 jittered samples with x- and y-projections; (b) 256 jittered samples.
n-Rooks

- Also called Latin hypercube sampling
- Use and n by n grid
- One sample exactly in each row and column
  - i.e. if samples were rooks in chess, no captures can occur

*Figure 5.10.* (a) 16 n-rooks samples in their initial positions; (b) the same samples shuffled in the x- and y-directions; (c) 256 samples.
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
- 2D not better than random...worse than jittered
Multi-Jittered Sampling

- We use two grids
  - Coarse grid with 1 sample per cell
  - Fine grid on which we enforce the rook condition

*Figure 5.11. (a) 16 multi-jittered samples in the initial distribution; (b) after shuffling in the x- and y-directions; (c) 256 multi-jittered samples.*
Multi-Jittered Sampling

- Good 1D projections from the rook condition
- Good 2D distribution from stratification
- For $n$ samples with $n$ a perfect square
  - Coarse grid is $\sqrt{n} \times \sqrt{n}$
  - Fine grid is $n \times n$
- Very good sampling technique
Hammersley Sampling

- Radical inverse function of integer $i$ to base 2
  - reflect binary digits of $i$ across decimal point
  - evaluate this new number now in $[0,1)$

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

<table>
<thead>
<tr>
<th>$i$</th>
<th>$1 _ _ _ 1$</th>
<th>$1 _ _ _ 1$</th>
<th>$1 _ _ _ 1$</th>
<th>$1 _ _ _ 1$</th>
<th>$1 _ _ _ 1$</th>
<th>$1 _ _ _ 1$</th>
<th>$1 _ _ _ 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1_2$</td>
<td>.1_2</td>
<td>.1_2</td>
<td>.1_2</td>
<td>.1_2</td>
<td>.1_2</td>
<td>.1_2</td>
</tr>
<tr>
<td>2</td>
<td>$10_2$</td>
<td>.01_2</td>
<td>.01_2</td>
<td>.01_2</td>
<td>.01_2</td>
<td>.01_2</td>
<td>.01_2</td>
</tr>
<tr>
<td>3</td>
<td>$11_2$</td>
<td>.11_2</td>
<td>.11_2</td>
<td>.11_2</td>
<td>.11_2</td>
<td>.11_2</td>
<td>.11_2</td>
</tr>
<tr>
<td>4</td>
<td>$100_2$</td>
<td>.001_2</td>
<td>.001_2</td>
<td>.001_2</td>
<td>.001_2</td>
<td>.001_2</td>
<td>.001_2</td>
</tr>
<tr>
<td>5</td>
<td>$101_2$</td>
<td>.101_2</td>
<td>.101_2</td>
<td>.101_2</td>
<td>.101_2</td>
<td>.101_2</td>
<td>.101_2</td>
</tr>
<tr>
<td>6</td>
<td>$110_2$</td>
<td>.011_2</td>
<td>.011_2</td>
<td>.011_2</td>
<td>.011_2</td>
<td>.011_2</td>
<td>.011_2</td>
</tr>
<tr>
<td>7</td>
<td>$111_2$</td>
<td>.111_2</td>
<td>.111_2</td>
<td>.111_2</td>
<td>.111_2</td>
<td>.111_2</td>
<td>.111_2</td>
</tr>
<tr>
<td>8</td>
<td>$1000_2$</td>
<td>.0001_2</td>
<td>.0001_2</td>
<td>.0001_2</td>
<td>.0001_2</td>
<td>.0001_2</td>
<td>.0001_2</td>
</tr>
</tbody>
</table>

Table 5.1. Binary representations and radical inverse functions for the integers 1–8.
Hammersley Sequence

Set of n 2D samples in unit square:

\[ p_i = (x_i, y_i) = \left[ \frac{i}{n}, \Phi_2(i) \right] \]

Book Formula on page 108 is wrong
Hammersley Issues

- Pattern is well-sampled but...
- 1D projections are regular
  - Too much structure in the pattern
- For a given n only one sequence exists (read section 5.1)
Better low discrepancy sequence
Generate n-dimensional points
  though Hammersley can be generalized as well....
Number of samples need not be known in advance

\[ p_i = (\Phi_2(i), \Phi_3(i), \Phi_5(i), ...) \]
- The 2,3 Halton Sequence
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 $V$)/$K$
    - ...but it generally won’t
  - the difference is the discrepancy

Different formal ways to measure discrepancy...see PBR
Some Results

Regular, 1 and 256 samples per pixel

Random, 1 and 256 samples per pixel
Some Results

Jittered

N-Rooks
Some Results

Multi-jittered

Hammersley