Distribution Ray-Tracing

3D Programming Course
João Madeiras Pereira
K. Suffern; “Ray Tracing from the Ground Up”, http://www.raytracegroundup.com

- Chapter 4, 5 for Anti-Aliasing
- Chapter 6 – for Disc Sampling
- Chapter 10 – for Depth of Field
Bibliography

Peter Shirley, Michael Ashikhmin:
“Fundamentals of Computer Graphics”
Chapter 10 – Ray-Tracing
Problem with Simple Ray Tracing: Aliasing

No antialiasing
raytraced images are too "clean"

- soft shadows come from area light
  - raytracing only supports point lights

[Jason Waltman / jasonwaltman.com]
Raytraced images are “too clean”

- blurry reflections come from rough materials
  - raytracing only supports perfectly sharp mirror

[Jensen]
Raytraced images are “too clean”

- depth of field come from lens system
  - raytracing only support pinhole camera

[Jason Waltman / jasonwaltman.com]
Raytraced images are “too clean”

- motion blur come from shutter speed
  - raytracing only support infinitely fast shutter speed

[Jason Waltman / jasonwaltman.com]
Distribution Ray –Tracing (DRT)

use many rays to compute average values over pixel areas, time, area lights, reflected directions, ...
Distribution RT

- Distributed Ray Tracing, also called Distribution Ray Tracing and Stochastic Ray Tracing, is a refinement of ray tracing that allows for the rendering of "soft" phenomena.
- Averaging multiple rays distributed over an interval.
- Soft shadows can be rendered by distributing shadow rays over the light source area.
- Anti-aliasing can be rendered by distributing rays over a pixel.
- Distribute rays across the eye to simulate depth of field effect.
- Blurry reflections and transmissions can be rendered by distributing reflection and transmission rays over a solid angle about the "true" reflection or transmission direction.
- Distribute rays in time to get temporal antialiasing (motion blur).
Aliasing

Ray tracing gives a color for every possible point in the image.

But a square pixel contains an infinite number of points. These points may not all have the same color. Sampling: choose the color of one point (center of pixel). Regular sampling leads to aliasing.

jaggies
moire patterns
Supersampling

- attempts to reduce the errors by shooting more than one viewing ray into each pixel and averaging the results to determine the pixel's apparent color.

The resulting color for this pixel will be two-thirds blue and one-third green.
Regular Sampling

• Fire more than one ray for each pixel (e.g., a 4x4 grid of rays)
• Average the results (perhaps using a filter)
Regular Sampling

[Shirley]

• Replace the code

\[
\text{for each pixel } (i, j) \text{ do } \\
c_{ij} = \text{ray-color}(i + 0.5, j + 0.5)
\]

• With code that samples on a regular n x n grid:

\[
\text{for each pixel } (i, j) \text{ do } \\
c = 0 \\
\text{for } p = 0 \text{ to } n - 1 \text{ do } \\
\quad \text{for } q = 0 \text{ to } n - 1 \text{ do } \\
\qquad c = c + \text{ray-color}(i + (p + 0.5)/n, j + (q + 0.5)/n) \\
c_{ij} = c/n^2
\]
Regular Sampling Optimization

- Regular sampling takes 16 times longer to render
- Solutions:
  - Adaptive, non-recursive supersampling
Adaptive, non-recursive supersampling

- initially traces one ray per pixel. If the color of a pixel differs from its neighbors (to the left or below) by at least the set threshold value then the pixel is super-sampled by shooting a given, fixed number of additional rays. A good threshold value is 0.3
- If $r_1$, $g_1$, $b_1$ and $r_2$, $g_2$, $b_2$ are the rgb components of two pixels then the difference between pixels is computed by:
  \[
  \text{diff} = |r_1 - r_2| + |g_1 - g_2| + |b_1 - b_2|
  \]
- If the anti-aliasing threshold is 0.0 then every pixel is super-sampled. If the threshold is 3.0 then no anti-aliasing is done
Adaptive Supersampling - Monte-Carlo Sampling

• It’s a recursive technique
• It starts by tracing four rays at the corners of each pixel.
• If the colors are similar (check the threshold) then just use their average
• Otherwise recursively subdivide each cell of the grid into four sub-pixels: fire additional 5 rays – at the center and at mid of the 4 edges
• Sub-pixels are separately traced and tested for further subdivision
• Keep going until each 2x2 grid is close to uniform or limit is reached
• Filter the result
• The advantage of this method is the reduced number of rays that have to be traced.
• Samples that are common among adjacent pixels and sub-pixels are stored and reused to avoid re-tracing of rays.
Adaptive Supersampling - Monte-Carlo Sampling
Stochastic (Random) Sampling

[Shirley]

- Adaptive Supersampling still divides pixels into regular patterns of rays, and suffers from aliasing that can occur from regular pixel subdivision – Moiré patterns
- It sends a fixed number of rays into a pixel, but makes sure they are randomly distributed (but more or less evenly cover the area)
Stochastic (Random) Sampling

[Shirley]

• Code:

\[
\text{for each pixel } (i, j) \text{ do } \\
\quad c = 0 \\
\quad \text{for } p = 1 \text{ to } n^2 \text{ do } \\
\quad \quad c = c + \text{ray-color}(i + \xi, j + \xi) \\
\quad c_{ij} = c / n^2
\]

• \(\xi\) is a call that returns a uniform random number in the range \([0, 1]\)

• One interesting side effect of the stochastic sampling patterns is that they actually injects noise into the solution (slightly grainier images). This noise tends to be less objectionable than aliasing artifacts.

• Noise can be quite objectionable unless many samples are taken.

• Solution: Hybrid strategy that randomly perturbs a regular grid – Jittering or Stratified Sampling
Jittering [Shirley]

Figure 10.29. Sixteen stratified (jittered) samples for a single pixel shown with and without the bins highlighted. There is exactly one random sample taken within each bin.

for each pixel \((i, j)\) do
\[
c = 0
\]
for \(p = 0\) to \(n - 1\) do
  for \(q = 0\) to \(n - 1\) do
    \[
c = c + \text{ray-color}(i + (p + \xi)/n, j + (q + \xi)/n)
    \]
  \[
c_{ij} = c/n^2
\]
Soft shadows

Distributing rays over light source area gives:
Soft Shadows

Point light sources produce sharp shadow edges
  – the point is either shadowed or not
  – only one ray is required
With an extended light source the surface point may be partially visible to it
  – only part of the light from the sources reaches the point
  – the shadow edges are softer
  – the transition region is the penumbra
Accomplish this by
  – firing shadow rays to random points on the light source
  – weighting them by the brightness
  – the resulting shading depends on the fraction of the obstructed shadow rays
Soft Shadows

- Account for the light being an area rather than a point
- To approximate the light with a distributed set of N point lights, each with one Nth of the intensity of the base light
  - For each hit point, compute shadows and lighting
  - Average the results
- Problem: typically dozens of point lights are needed to achieve visually smooth results (slows down the performance of the RT)
- Solution: represent the area light as an infinite number of point lights and choose one at random for each viewing ray
Soft Shadows [Shirley]

Figure 10.32. The geometry of a parallelogram light specified by a corner point and two edge vectors.

\[ r = c + \xi_1 a + \xi_2 b, \]

where \( \xi_1 \) and \( \xi_2 \) are uniform random numbers in the range \([0, 1)\).
Fewer rays, more noise

More rays, less noise
Depth of Field

Distributing rays over a finite aperture gives:
Virtual and Real Camera [Suffern]

- the virtual camera (a) models the pinhole camera; the eye corresponds to the pinhole of (b)
- In (a) the view plane is between the eye and the objects
- the pinhole in (b) is in the between the objects and the film plane
- The “lens” is infiniteley small
- Real cameras have finite-aperture lens with focal distance

Figure 9.2. (a) Computer perspective viewing; (b) real pinhole camera, where the image on the film plane is inverted.
• Depth of field (DOF) is the range of distances parallel to the lens axis in which the scene is in focus
• In RT, the image can appear in focus over the range of distances where the circle of confusion is smaller than a pixel
The simulation requires a large number of rays/pixel whose origins are distributed over the surface of the lens.

**Figure 10.3.** A thin-lens camera consists of a disk for the lens, a view plane (as usual), and a focal plane, all perpendicular to the view direction.
**Depth of Field** [Suffern]

- To simulate DOF:
  - Compute the point $p$ where the center ray hits the focal plane;
  - Use $p$ and the sample point on the lens to compute the direction of the primary ray so that this ray also goes through $p$;
  - Ray-trace the primary ray into the scene; the center ray does not contribute to the pixel color

- But $p$, although in perfect focus, will not be antialiased; what to do?
DOF + Antialiasing

**Figure 10.6.** Four center rays go through different sample points on a pixel.

**Figure 10.7.** Four primary rays that start at sample points on the lens and hit the focal plane at the same points that the center rays in Figure 10.6 hit it.
Center Rays \cite{Suffern}

- We don’t have to trace the center rays

\[ p = (p_x, p_y, -f), \]
\[ p_s = (p_{sx}, p_{sy}, -d), \]
\[ l_s = (l_{sx}, l_{sy}, 0), \]
\[ p_x = p_{sx} \left( \frac{f}{d} \right), \]
\[ p_y = p_{sy} \left( \frac{f}{d} \right). \]

\[ d_r = p - l_s \]
\[ = (p_x - l_{sx})u + (p_y - l_{sy})v - fw. \]
Depth of Field

- Aperture: sample point on unit disc or unit square times the lens radius
  - Very Small Aperture
Depth of Field

- Small Aperture
Depth of Field

• Large Aperture
Depth of Field

- Very Large Aperture
DRT to simulate motion blur

- Distributing rays over time gives:
Distribution Ray Tracing

distribute rays throughout a pixel to get spatial antialiasing

distribute rays in time to get temporal antialiasing (motion blur)

distribute rays in reflected ray direction to simulate gloss

distribute rays across area light source to simulate penumbrae
  (soft shadows)

distribute rays across eye to simulate depth of field

distribute rays across hemisphere to simulate diffuse
  interreflection

also called: “distributed ray tracing” or stochastic ray tracing

aliasing is replaced by less visually annoying noise.

powerful idea! (but requires significantly more computation)