DISTRIBUTED RAY TRACING

some implementation notes

Theory v. practice

• Brute force – multiple rays at every sampling opportunity
• Implementation – for each pixel subsample, randomize at each opportunity

DRT components

• anti-aliasing and motion blur
  – supersampling - in time and space: anti-aliasing
  – jitter sample in time and space: motion blur
• depth of field - sample lens: blurs
• shadows – sample light source: soft shadows
• reflection – sample reflection direction: rough surface
• transparency – sample transmission direction: translucent surface

Replace Camera Model

• shift from pinhole camera model to lens camera model
• picture plane at -w, not +w
• camera position becomes lens center
• picture plane is behind 'pinhole'
• negate u, v, w, trace ray from pixel to camera
ANTI-ALIASING:
Organizing subpixel samples
Options

1. Do each subsample in raster order
2. do each pixel in raster order, do each subsample in raster order
3. do each pixel in raster order, do all subsamples in temporal order
4. keep framebuffer, do all subsamples in temporal order

SPATIAL JITTERING

• for each pixel 200x200, i,j
• for each subpixel sample 4x4 s.t
• Jitter sample

MOTION BLUR
TEMPORAL JITTERING

• for subsample
• get delta time from table
• jitter delta +/- 1/2 time division
• move objects to that instant in time

DEPTH OF FIELD

• generate ray from subsample through lens center to focal plane
• generate random sample on lens disk - random in 2D u,v
• generate ray from this point to focal plane point
VISIBILITY - as usual

- intersect ray with environment
- find first intersection at point $p$ on object $o$ with normal $n$

Shadows

- generate random vector on surface of light
  - random on sphere

Reflections

- compute reflection vector
- generate random sample in sphere at end of $R$
  - Importance sampling
- Randomize $n$ instead of $R$

Transparency

- compute transmission vector
- generate random sample in sphere at end of $T$
  - Importance sampling
- Randomize $n$ instead of $T