Aliasing

Ray tracing is a form of sampling and can suffer from annoying visual artifacts...

Consider a continuous function $f(x)$. Now sample it at intervals $\Delta$ to give $f[i] = \text{quantize}[f(i\Delta)]$.

- The question is, how well does $f[i]$ approximate $f(x)$?

Consider sampling a sinusoid:

- In this case, the sinusoid is reasonably well approximated by the samples.

Aliasing (con’t)

Now consider sampling a higher frequency sinusoid:

- We get the exact same samples, so we seem to be approximating the first lower frequency sinusoid again.

- We say that, after sampling, the higher frequency sinusoid has taken on a new “alias”, i.e., changed its identity to be a lower frequency sinusoid.
Jaggies

- One of the most common rendering artifacts is the “jaggies”. Consider rendering a white polygon against a black background:

- We would instead like to get a smoother transition:

Anti-aliasing

- Q: How do we avoid artifacts caused by sampling?
- Sampling:
- Pre-filtering:
- Combination:

- Example - polygon:

Without anti-aliasing

With anti-aliasing
Antialiasing in a ray tracer

- We would like to compute the average intensity in the neighborhood of each pixel.
- When casting one ray per pixel, we are likely to have aliasing artifacts.
- To improve matters, we can cast more than one ray per pixel and average the result.
  A.k.a., super-sampling and averaging down.

Speeding it up

- Vanilla ray tracing is really slow!
- Consider: \( m \times m \) pixels, \( k \times k \) supersampling, and \( n \) primitives, average ray path length of \( d \), with 1 or 2 rays cast recursively per intersection.
- Complexity =
- For \( m=1000, k=5, n=100,000, d=8 \)...very expensive!!
- In practice, some acceleration technique is almost always used.
- We’ve already looked at reducing \( d \) with adaptive ray termination. Now we look at reducing the effect of the \( k \) and \( n \) terms.

Antialiasing by adaptive sampling

- Casting many rays per pixel can be unnecessarily costly.
- For example, if there are no rapid changes in intensity at the pixel, maybe only a few samples are needed.
- Solution: adaptive sampling.

- Q: When do we decide to cast more rays in a particular area?

Faster ray-polyhedron intersection

- Let’s say you were intersecting a ray with a polyhedron:
  - Straightforward method
    » intersect the ray with each triangle
    » return the intersection with the smallest \( t \)-value.
  - Q: How might you speed this up?
Hierarchical bounding volumes

- Intersect with largest B.V.
- Then intersect with children...
- ...until you reach the leaf nodes - the primitives.

Balanced trees with tight bounding volumes.

Hierarchical bounding volumes

Uniform spatial subdivision

- Another approach is **uniform spatial subdivision**.

  - Uniform subdivision in 2D
  - Uniform subdivision in 3D

- Idea:
  - Partition space into cells (voxels)
  - Associate each primitive with the cells it overlaps
  - Trace ray through voxel array using fast incremental arithmetic to step from cell to cell

Non-uniform spatial subdivision

- Still another approach is **non-uniform spatial subdivision**.

  - Quadtree in 2D
  - Octree in 3D

- Other variants include k-d trees and BSP trees.
- Various combinations of these ray intersections techniques are also possible. See Glassner for more.

Summary

- What to take home from this lecture:
  - The meanings of all the boldfaced terms.
  - An intuition for what aliasing is.
  - How to reduce aliasing artifacts in a ray tracer
  - An intuition for how ray tracers can be accelerated.