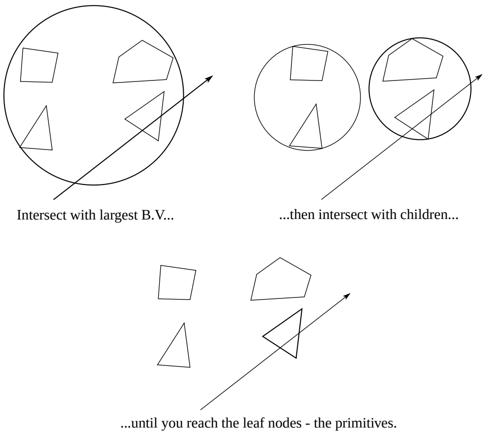
Acceleration: hierarchical bounding volumes

Vanilla ray tracing is really slow!

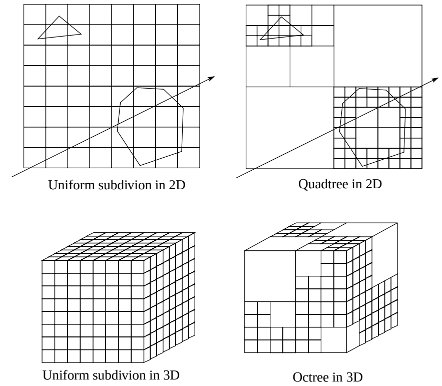
In practice, some acceleration technique is almost always used.

One approach is to use **hierarchical bounding volumes**.



Acceleration: spatial subdivision

Another approach is **spatial subdivision**.



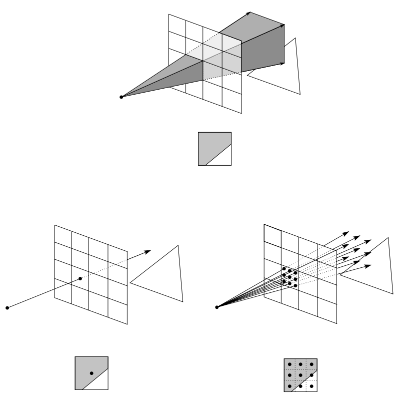
Idea:

- Partition objects spatially.
- Trace ray through voxel array.

Partition can be uniform or adaptive (e.g., octrees).

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.

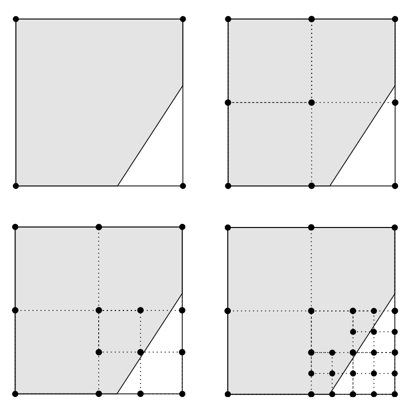
A.k.a., **super-sampling and averaging down**.

Antialiasing by adaptive sampling

Casting many rays per pixel can be unnecessarily costly.

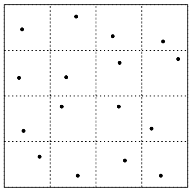
For example, if their are no rapid changes in intensity at the pixel, maybe only a few samples are needed.

Solution: **adaptive sampling**.



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

Distribution ray tracing

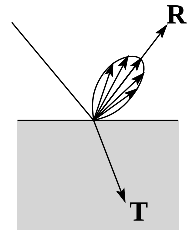


Idea:

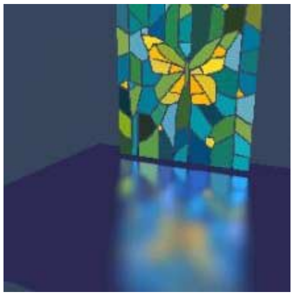
- Use non-uniform (jittered) samples.
- Replaces aliasing artifacts with noise.
- Provides additional effects if we distribute rays in other dimensions:
  - Reflection and refractions
  - Light source area
  - Camera lens area
  - Time

Originally called "distributed ray tracing," but we will call it **distribution ray tracing** so as not to confuse with parallel computing.

Distribution ray tracing, cont.



Distributing rays over reflection and/or refraction directions gives:

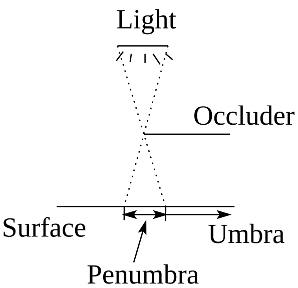


Distribution ray tracing, cont.

Operationally:

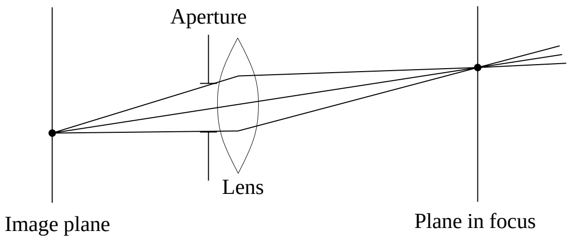
1. Partition the reflection directions into 16 angular regions. Assign each region a unique number between 1 and 16.
2. Partition each pixel into 16 regions. Assign each region a unique number between 1 and 16.
3. Select sub-pixel $m = 1$.
4. Cast a ray through sub-pixel m, jittered within its region.
5. After finding the first intersection, reflect into direction region m, jittered within that region. Repeat for future reflections.
6. Add result to current pixel total.
7. Increment m and, if $m \leq 16$, go to 4.
8. Divide by 16, store the result, choose the next pixel and go to 3.

Distribution ray tracing, cont.



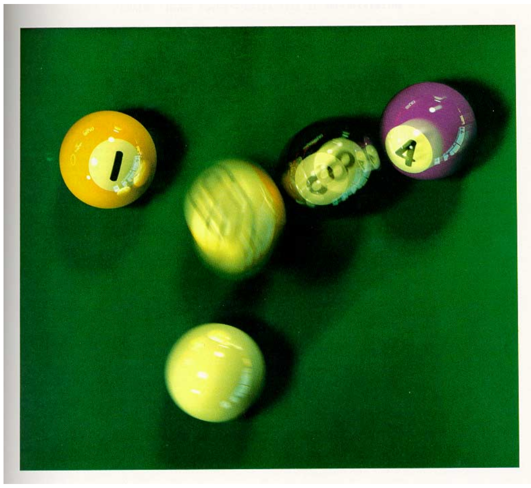
Distributing rays over light source area gives:

Distribution ray tracing, cont.



Distributing rays over a finite aperture gives:

Distribution ray tracing, cont.



Distributing rays over time gives: