CSC 4356
Interactive Computer Graphics
Lecture 23: Ray Tracing (part 3)

Jinwei Ye

http://www.csc.lsu.edu/~jye/CSC4356/

Tue & Thu: 10:30 - 11:50am
218 Tureaud Hall

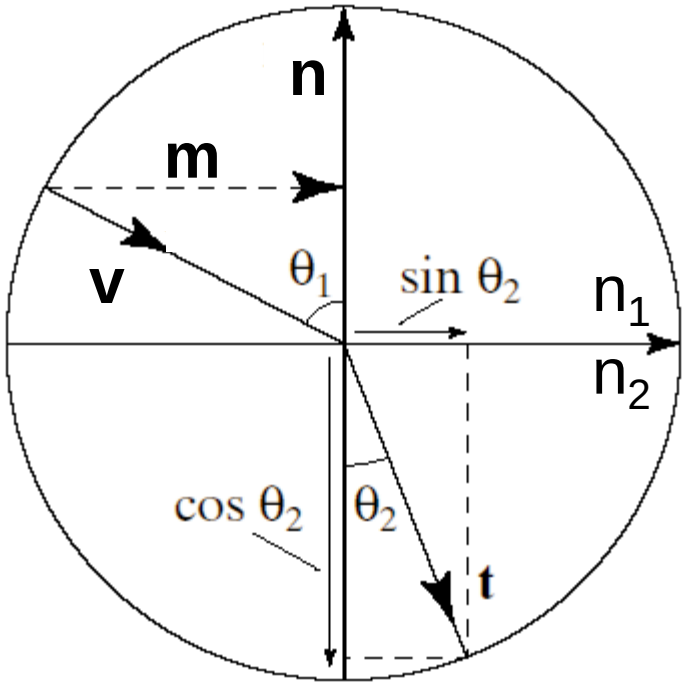
Computing the Transmission Direction **t**

$$n = \frac{n_1}{n_2}$$

$$c_1 = \cos\theta_1 = -\mathbf{v}\cdot\mathbf{n}$$

$$c_2 = \cos\theta_2 = \sqrt{1 - n^2(1 - c_1^2)}$$

$$\mathbf{t} = n\mathbf{v} + (nc_1 - c_2)\mathbf{n}$$



Total internal reflection happens when the term in the square root above isn't positive, which is when

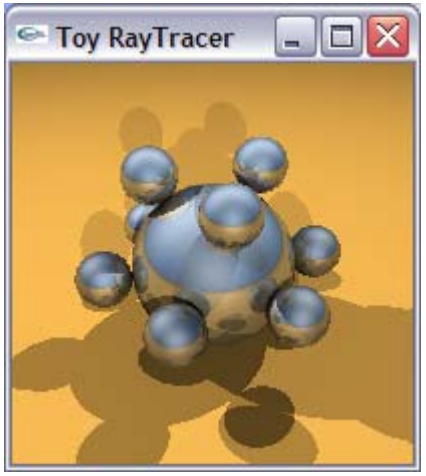
$$n^2(1 - c_1^2) \geq 1$$

Basic Ray Tracing: Notes

- Global illumination effects simulated by basic algorithm are shadows, purely specular reflection/transmission
- Some outstanding issues
  - Aliasing, aka jaggies
  - Shadows have sharp edges, which is unrealistic
  - No diffuse reflection from other objects
- Intersection calculations are expensive, and even more so for more complex objects
  - Not currently suitable for real-time (i.e., games)

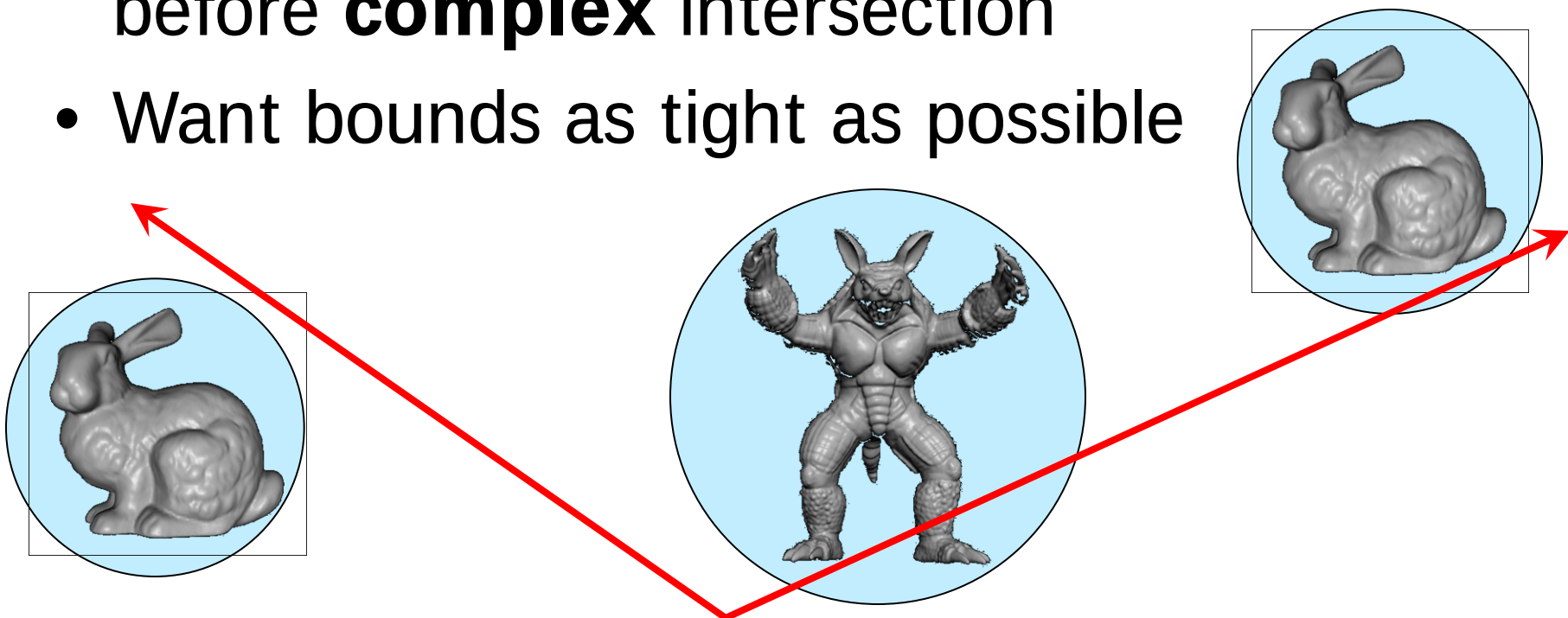
Acceleration Methods

- Render time for a ray tracer depends on the number of ray intersection tests per pixel
  - roughly dependent on the number of primitives in the scene times the number of pixels.
- Early efforts focused on accelerating the ray-object intersection tests
- More advanced methods required to make ray tracing practical
  - Bounding Volumes
  - Spatial Subdivision



Bounding Volumes for Efficiency

- Idea: enclose complex objects (i.e., .obj models) in simpler ones (e.g., spheres, boxes) and test **simple** intersection before **complex** intersection
- Want bounds as tight as possible

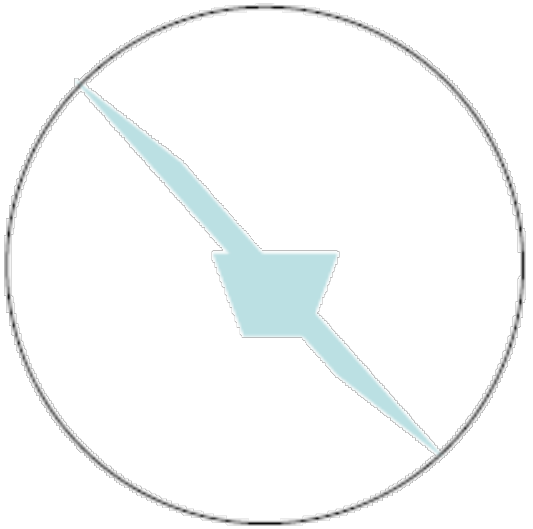


Bounding Volumes

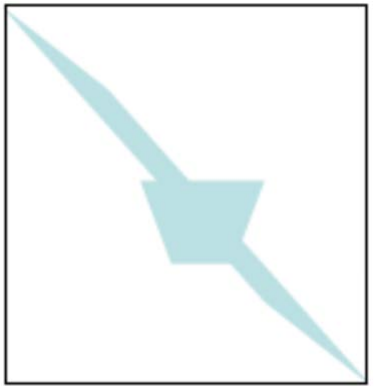
- Enclose complex objects within a simple-to-intersect objects.
  - If the ray does not intersect the simple object then its contents can be ignored
  - The likelihood that it will strike the object depends on how tightly the volume surrounds the object.
- Spheres are simple, but not tight
- Axis-aligned bounding boxes often better
  - can use nested or hierarchical bounding volumes

Bounding Volumes

- Sphere [Whitted80]
  - Cheap to compute
  - Cheap test
  - Potentially very bad fit

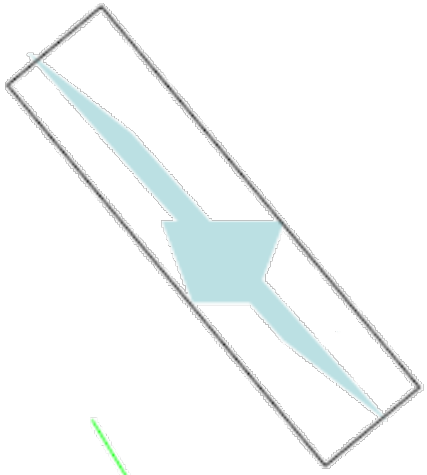


- Axis-Aligned Bounding Box
  - Very cheap to compute
  - Cheap test
  - Tighter than sphere

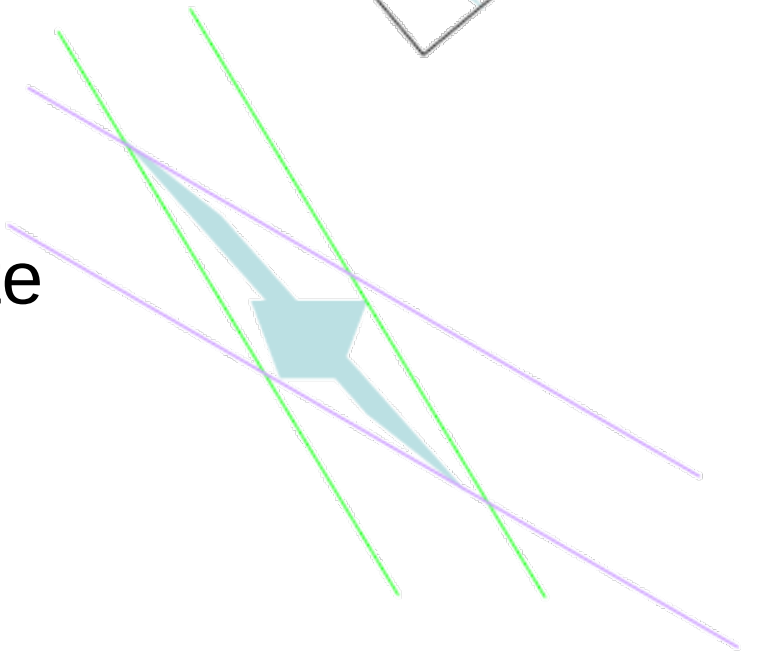


Bounding Volumes

- Oriented Bounding Box
  - Fairly cheap to compute
  - Fairly Cheap test
  - Generally fairly tight

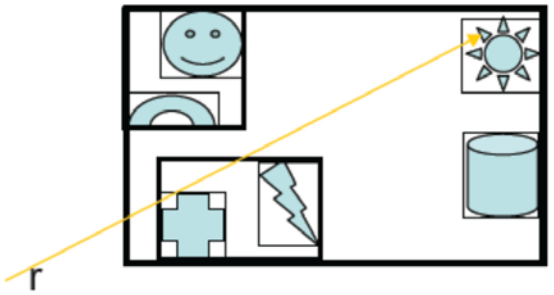


- Slabs / K-dops
  - More Expensive to compute
  - Fairly Cheap test
  - Can be tighter than OBB

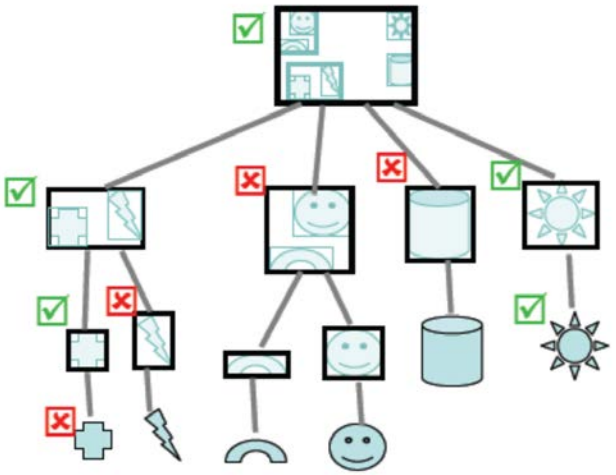


Hierarchical Bounding Volumes

- Organize bounding volumes as a tree
- Each ray starts with the scene BV and traverses down through the hierarchy



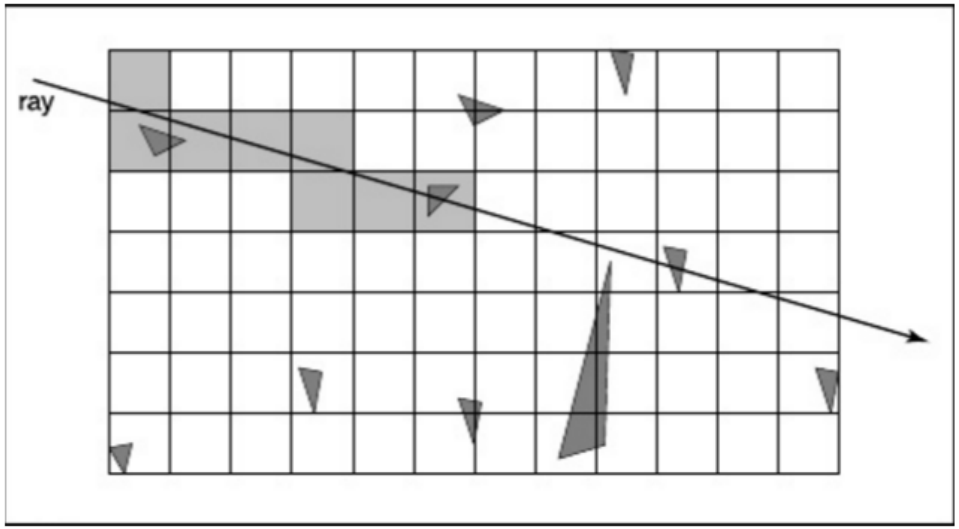
Nested boxes impose a *hierarchy...*



...that allow a more efficient recursive tree search

Spatial Subdivision

- **Idea**: Divide space in to sub-regions
  - Place objects within a sub-region into a list
  - Only traverse the lists of sub-regions that the ray passes through
  - Sub-space types
    - Regular grid
    - Octree
    - BSP tree
    - kd-tree



Other Optimizations

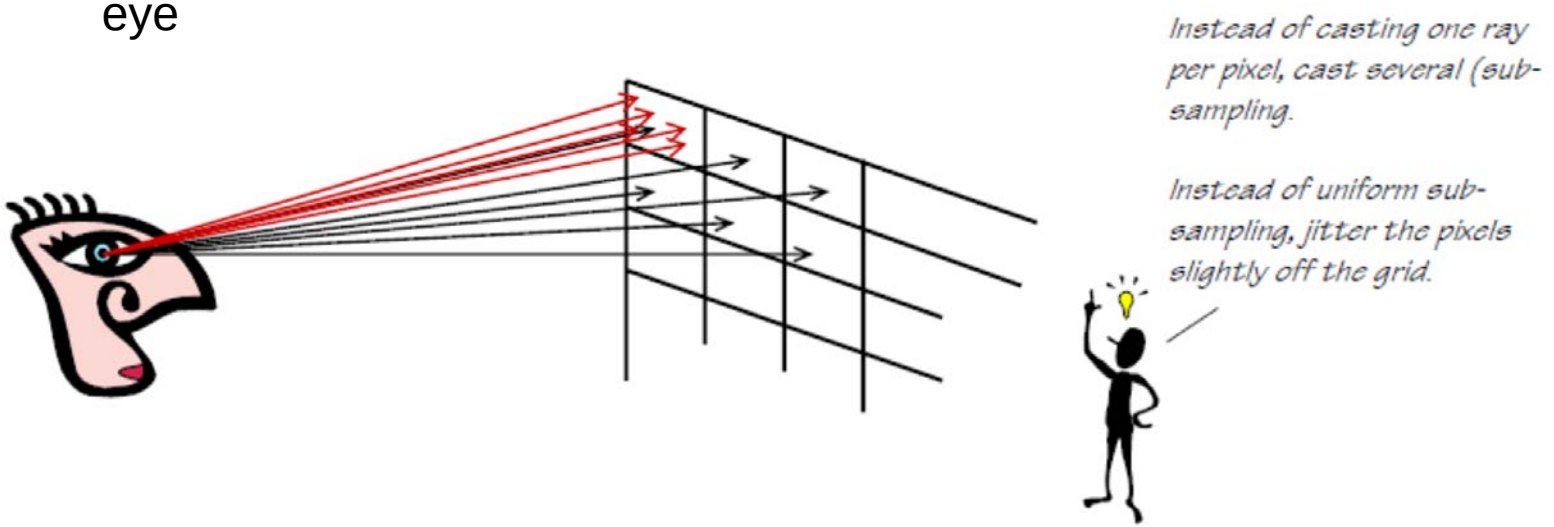
- Shadow cache
  - due to coherence the last object intersected will likely be intersected on the next ray
  - save last hit object and test it first for next ray
- Adaptive depth control
  - limit the depth of recursion
  - the color of secondary rays gets modulated in lighting equations. Stop recursing when contribution falls below a threshold
- Lazy geometry loading/creation
  - for very complex models or procedural models we can supply a bounding volume and defer the actual loading/creation of the geometry until a ray hits the bounding volume

Distribution Ray Tracing

- Cook & Porter, in their classic paper “Distributed Ray Tracing” realized that ray-tracing, when combined with randomized sampling, which they called “jittering”, could be adapted to address a wide range of rendering problems:

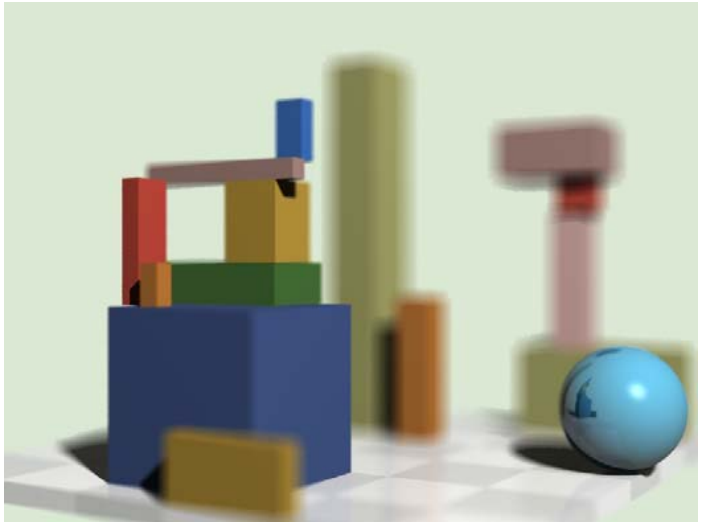
Antialiasing

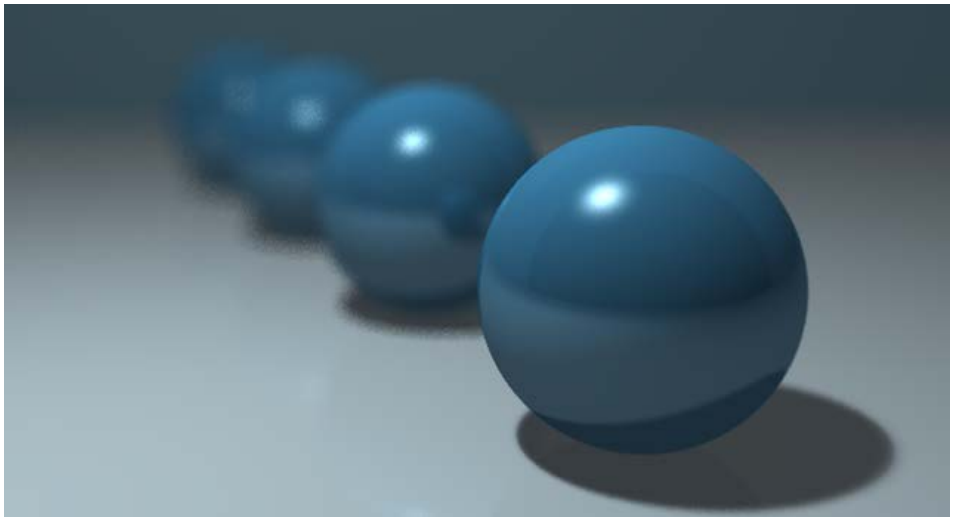
- The need to sample is problematic because sampling leads to aliasing
- Solution 1) **super-sampling**
  - increases sampling rate, but does not completely eliminate aliasing
  - difficult to completely eliminate aliasing without pre-filtering because the world is not band-limited
- Solution 2) **distribute the samples randomly**
  - converts the aliasing energy to noise which is less objectionable to the eye



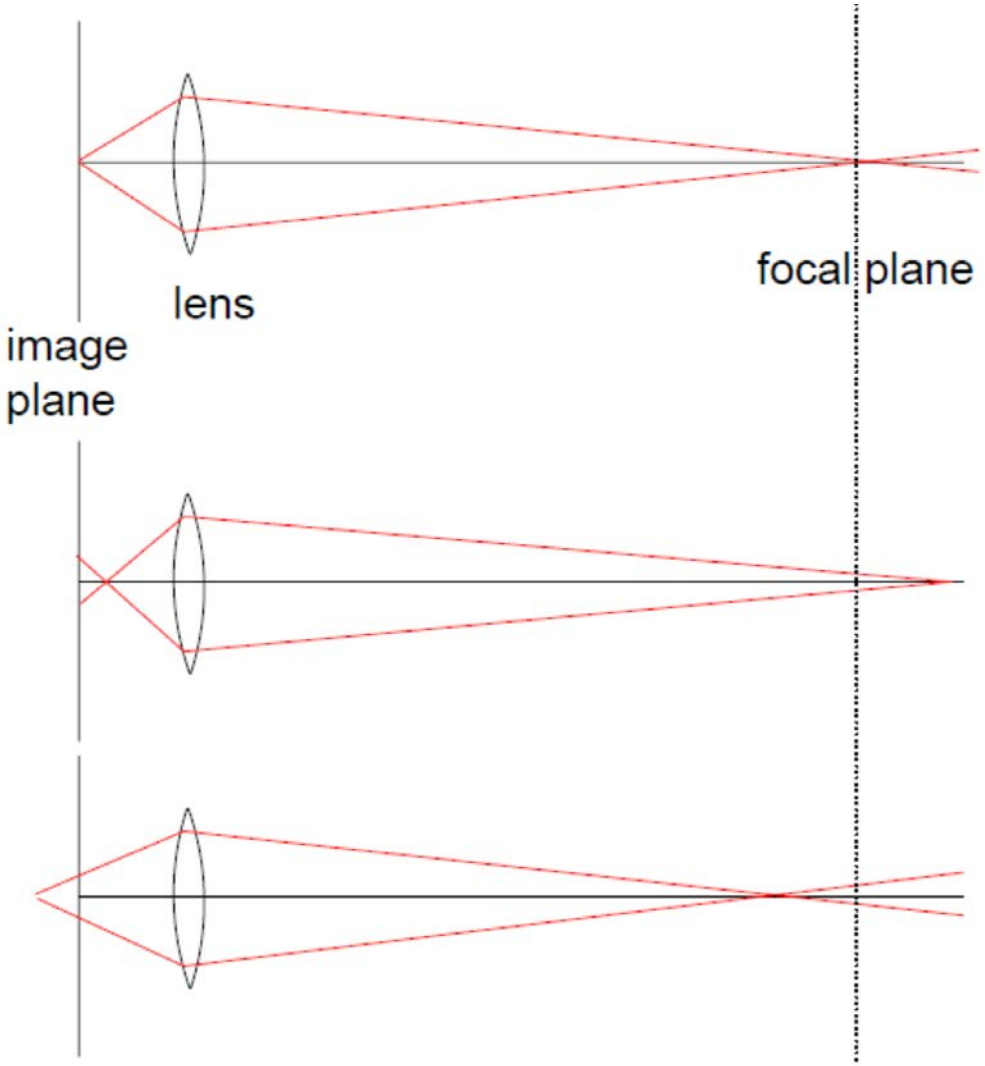
Depth-of-Field

- Rays don’t have to all originate from a single point.
- Real cameras collects rays over an aperture
  - can be modeled as a disk
  - final image is blurred away from the focal plane.
  - gives rise to depth-of-field effects.



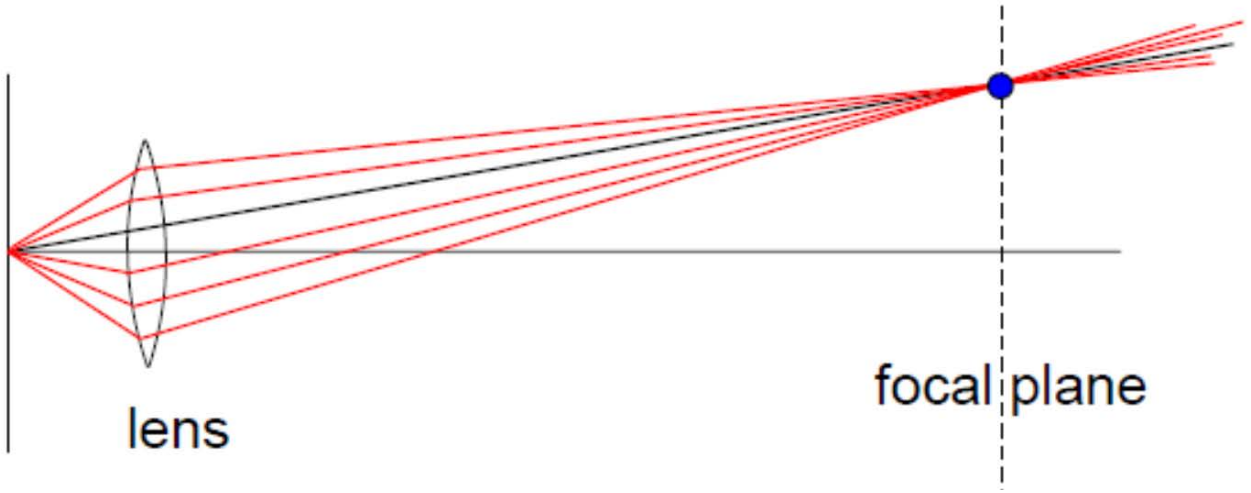


Depth-of-Field



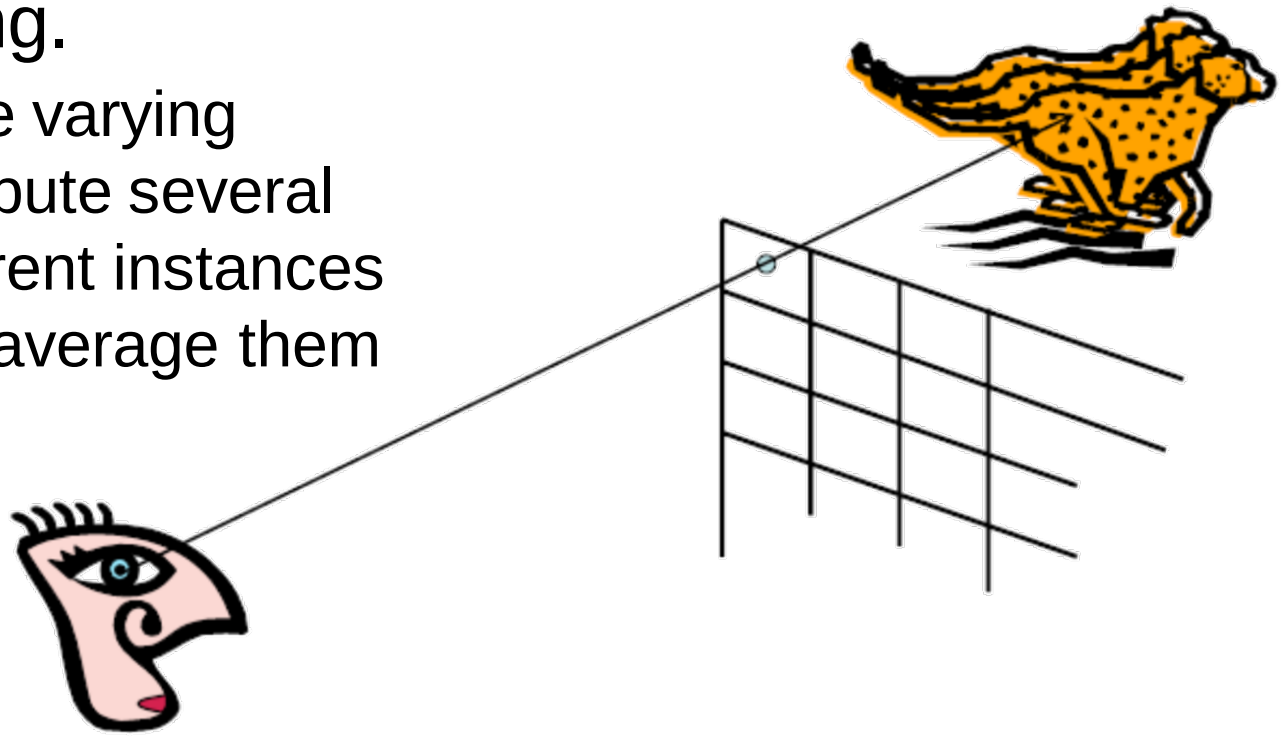
Depth-of-Field

- Start with normal eye ray and find intersection with focal plane
- Choose jittered point on lens and trace line from lens point to focal point

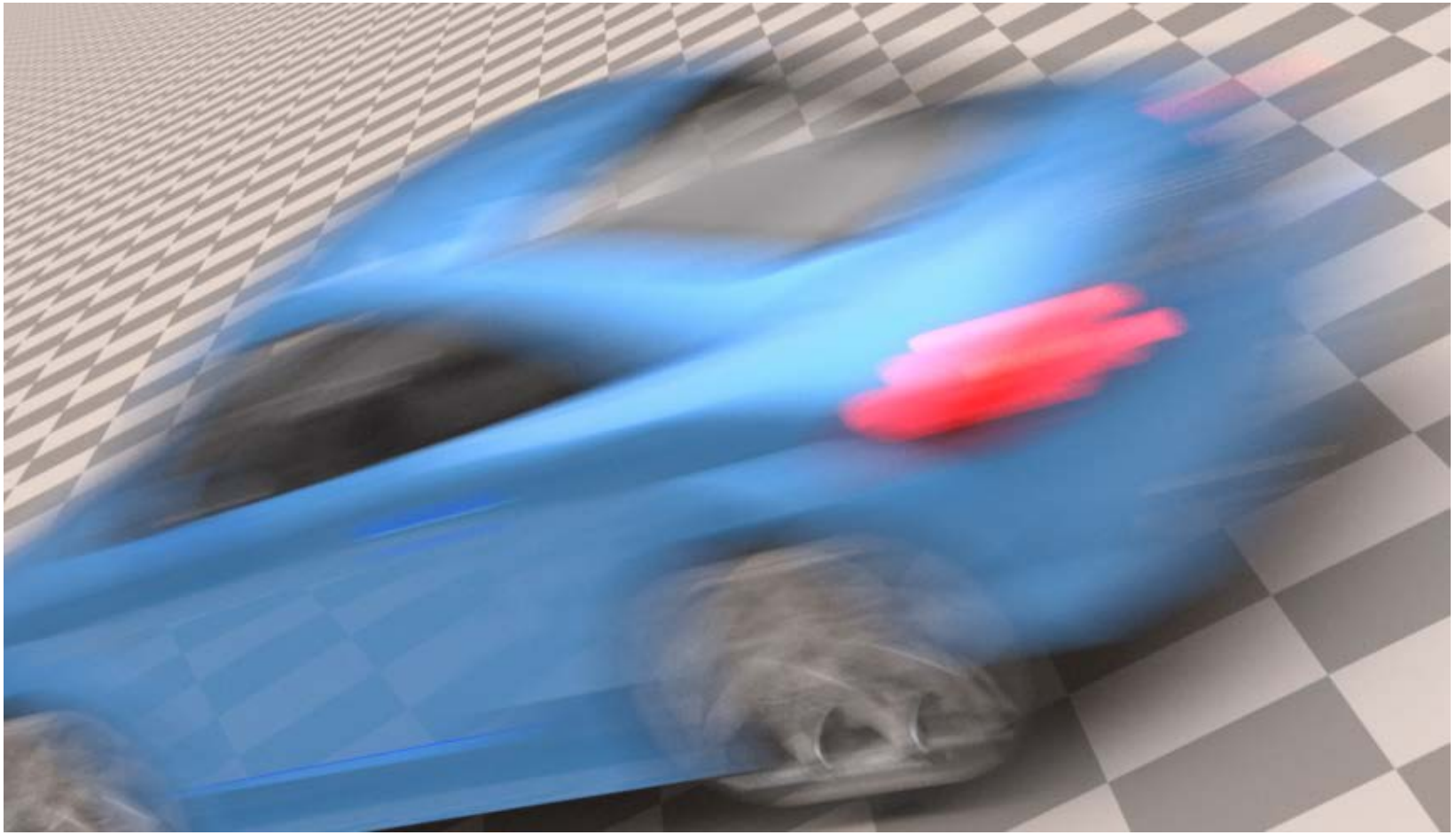


Motion Blur

- You can also jitter samples through time to simulate the finite interval that a shutter is open on a real camera. This produces motion blur in the rendering.
  - Given a time varying model, compute several rays at different instances of time and average them together.

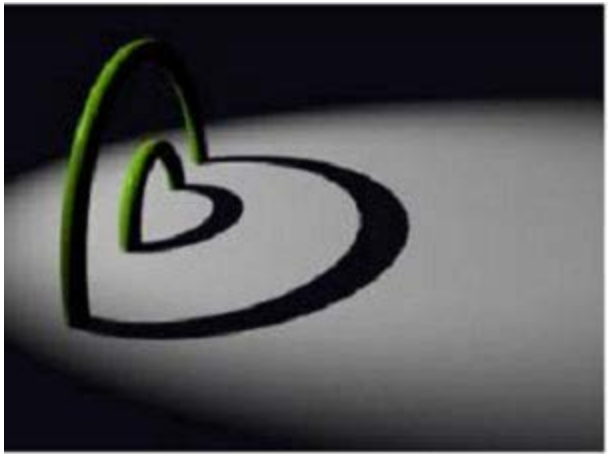


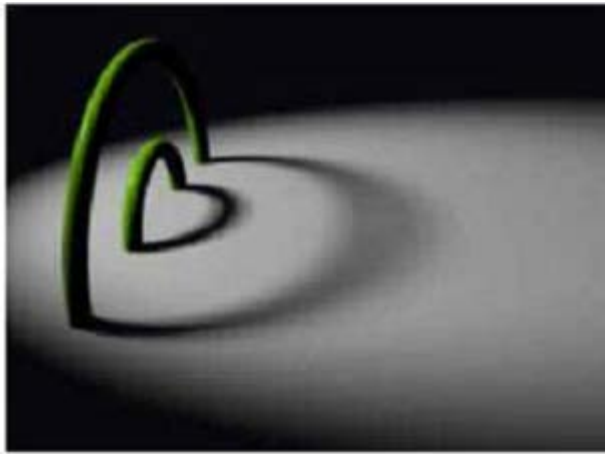
Motion Blur Example



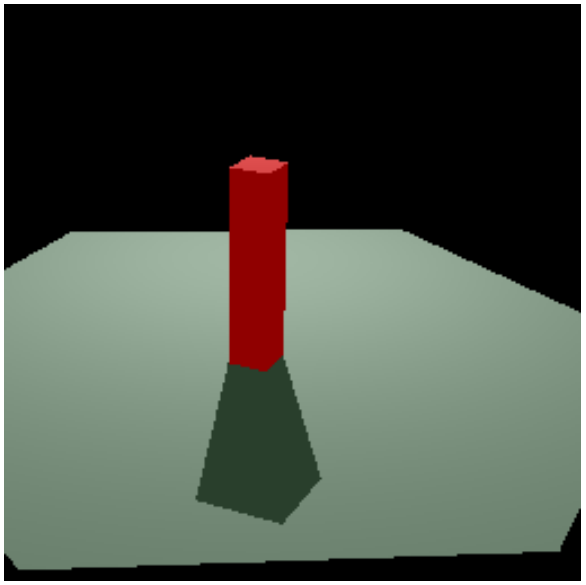
Soft shadows

- For point light sources, sending a single shadow ray toward each is reasonable
  - But this gives hard-edged shadows
- Simulating soft shadows
  - Model each light source as sphere
  - Send multiple jittered shadow rays toward a light sphere; use **fraction** that reach it to attenuate color
  - Similar to ambient occlusion, but using list of light sources instead of single hemisphere

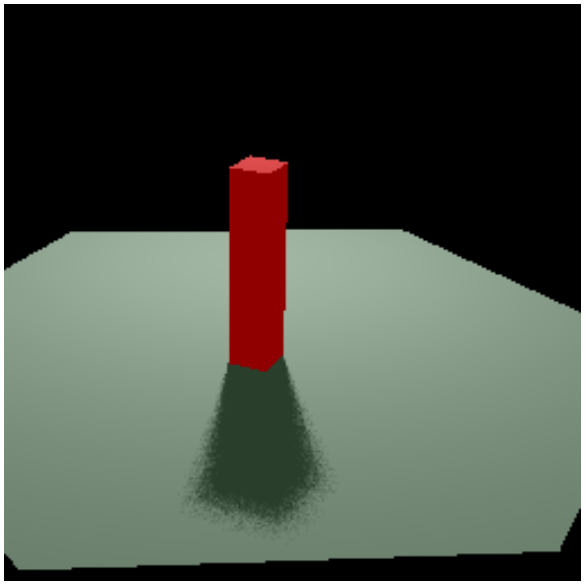




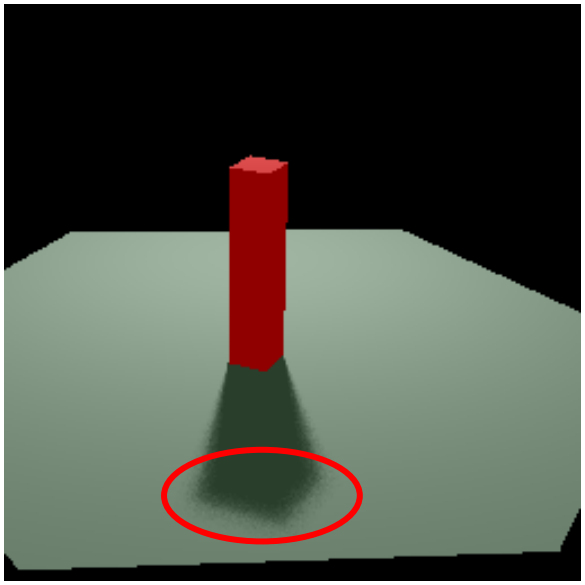
Soft Shadows: Example



1 shadow ray



10 shadow rays

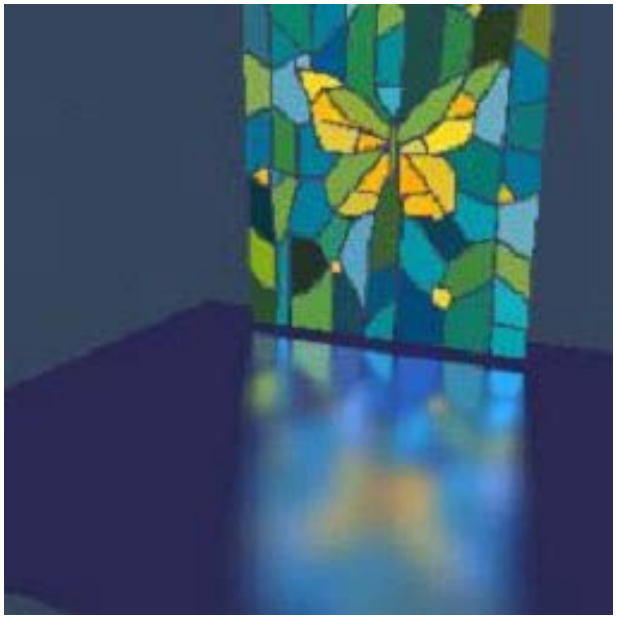


50 shadow rays

Note discrete "shadow points" -- need post-processing to smooth into contiguous region

Glossy Reflections

- Analog of hard shadows are “sharp reflections”—every reflective surface acts like a perfect mirror
- To get glossy or blurry reflections, send out multiple *jittered reflection rays* and average their colors



Why is the reflection sharper at the top?