南京大学
NANJING UNIVERSITY

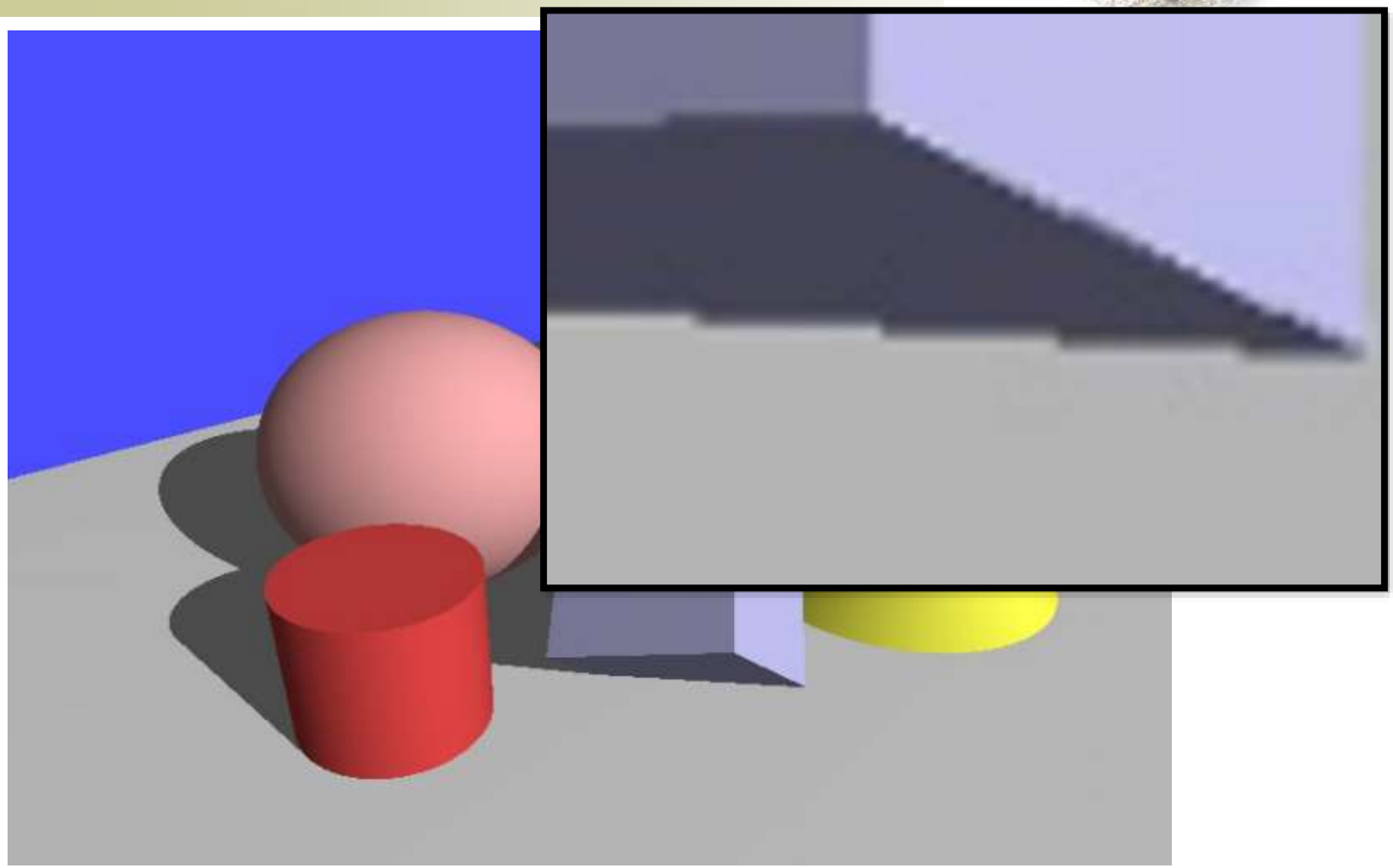
图形绘制技术（Rendering）
Chapter 4: Ray Tracing Advanced

过 洁
南京大学计算机科学与技术系
guojie@nju.edu.cn

Shading the Intersection Point

- The shading of each intersection point is the sum of contributions from all light sources
  - Cast rays from the intersection point to all light sources
- Light types:
  - Ambient light, point light, directional light, spot light, area light, volume light, etc.
- Material properties:
  - Diffusion, specular, shininess, emission, etc.

A Ray Tracing Example



Ray Tracing Fails

Distributed Ray Tracing

What is distributed ray tracing?

- Distributed ray tracing is **not** ray tracing on a distributed system.
- Distributed ray tracing **is** a ray tracing method based on randomly distributed oversampling to reduce aliasing artifacts in rendered images.

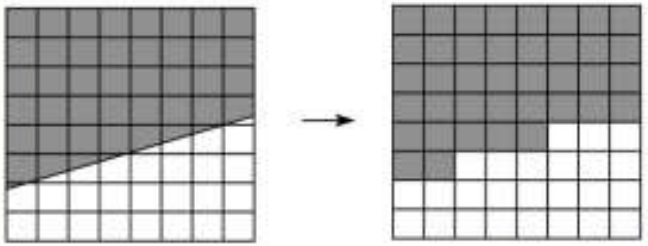
Distributed Ray Tracing

- Motivation
  - The classical ray tracing produces very clean images (look fake)
    - Perfect focus
    - Perfect reflections
    - Sharp shadows
- Main idea
  - Replace the single ray with a distribution of rays
- Add randomness to rendering
  - Antialiasing
  - Soft shadows
  - Depth-of-field
  - Motion blur
  - Glossy reflections

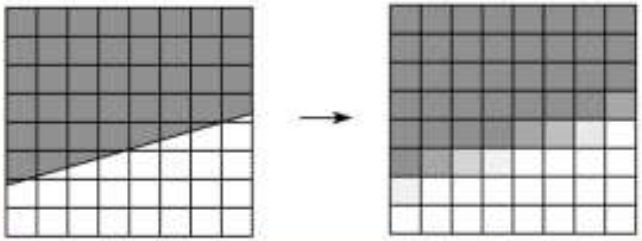
[Cook et al. 1984]

Aliasing in Rendering

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

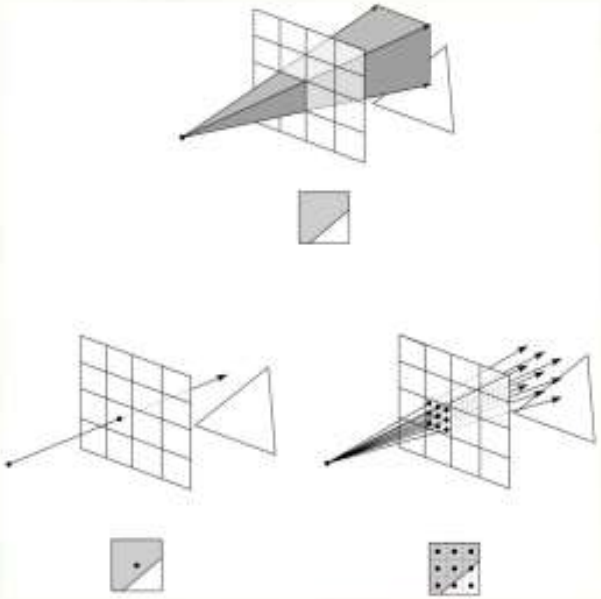


- We would instead like to get a smoother transition:



Antialiasing in a ray tracer

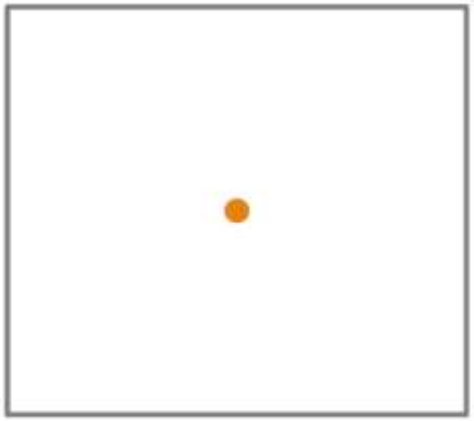
- We would like to compute the average intensity in the neighborhood of each pixel.

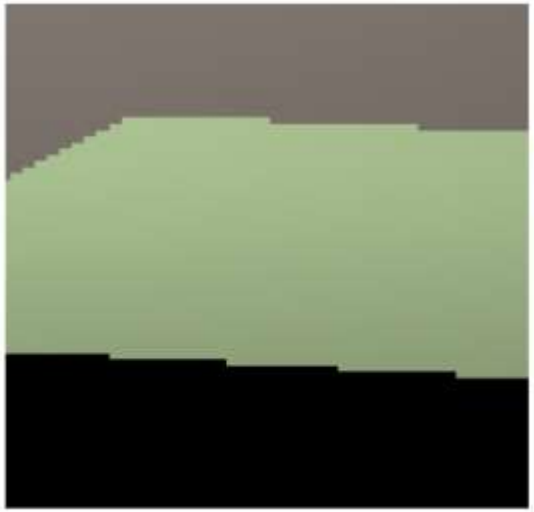


- When casting one ray per pixel, we are likely to have aliasing artifacts.
- To improve matters, we can cast more than one ray per pixel and average the result.
- A.k.a., **super-sampling and averaging down**.

Antialiasing

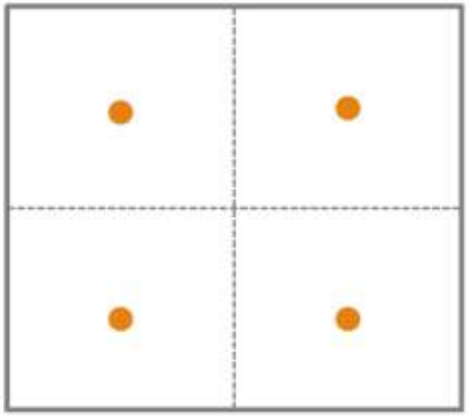
- e.g. one sample / pixel
- for each pixel (x, y) do
  - $c(x,y) = trace(x + 0.5, y + 0.5)$

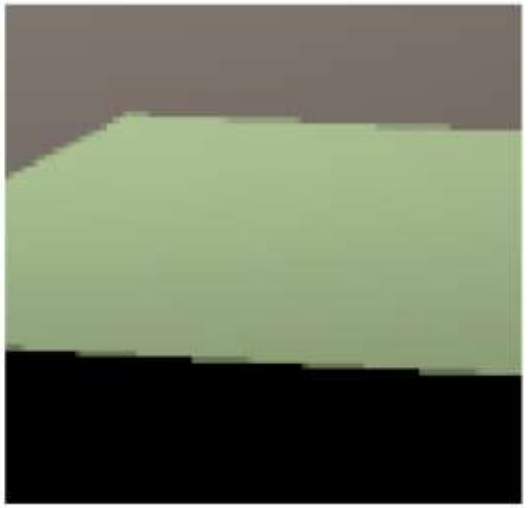




Antialiasing

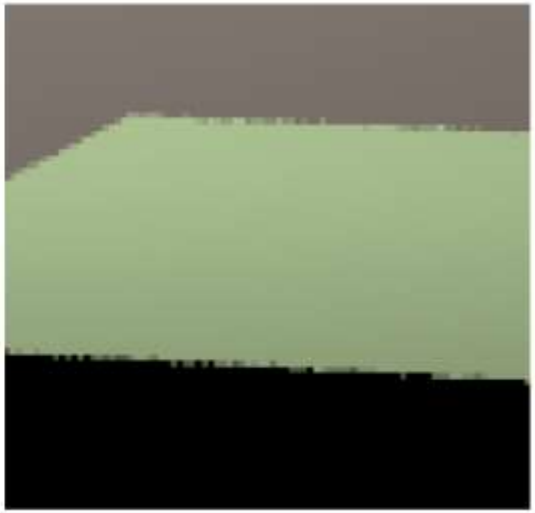
- e.g. four samples / pixel
- Regular sampling: divide each pixel area to $n \times n$ grids and generate a ray within each grid
- for each pixel (x, y) do
  - $c(x,y) = 0$
  - for p = 0 to n – 1 do
    - for q = 0 to n – 1 do
      - $c(x,y) = c(x,y) + trace\left(x + \frac{p+0.5}{n}, y + \frac{q+0.5}{n}\right)$
    - $c(x,y) = c(x,y)/n^2$
- Equal to the traditional rendering
  - Uses a larger image resolution
  - Down-sample the image to make a target resolution
  - Still make the regular pattern!





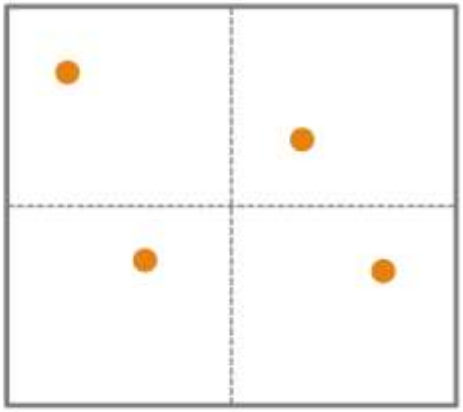
Antialiasing

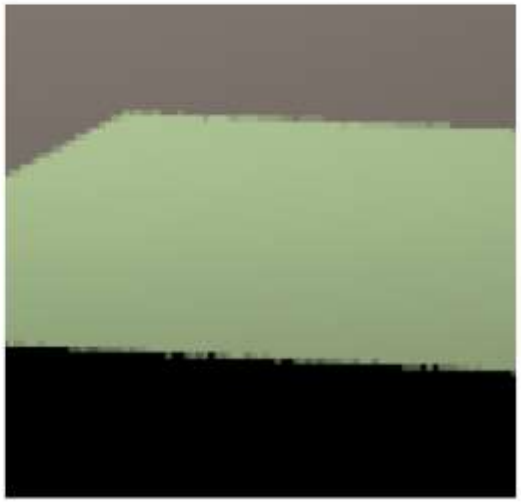
- e.g. four samples / pixel
- Random sampling: randomly generate n^2 rays
- for each pixel (x, y) do
  - $c(x,y) = 0$
  - for p = 0 to $n^2 - 1$ do
    - $c(x,y) = c(x,y) + trace(x + \epsilon, y + \epsilon)$
  - $c(x,y) = c(x,y)/n^2$
- $\epsilon \in [0,1)$ is a random number
- The regular pattern is converted into image noise



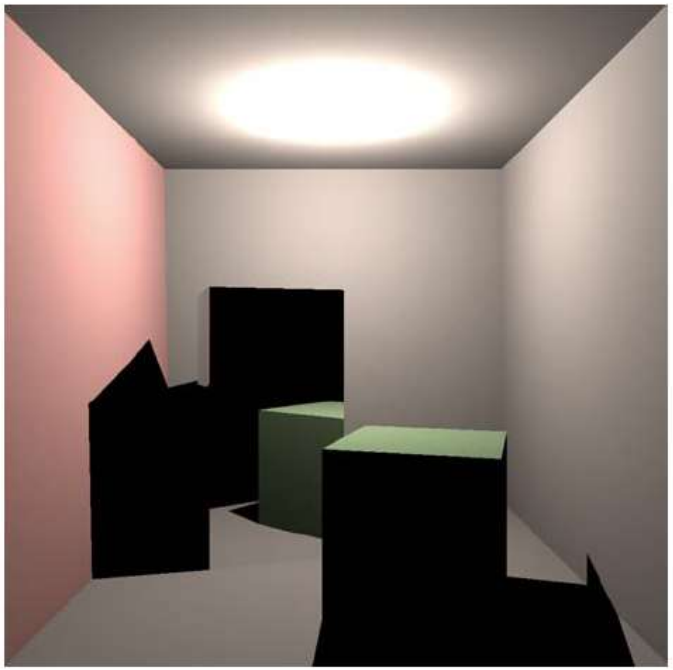
Antialiasing

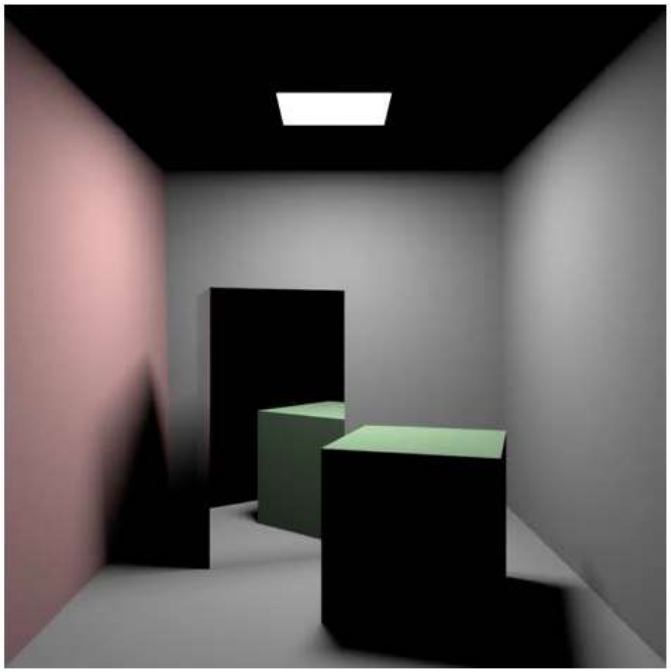
- e.g. four samples / pixel
- Jittering (stratified sampling): randomly generate a ray within each grid
- for each pixel (x, y) do
  - $c(x,y) = 0$
  - for p = 0 to n – 1 do
    - for q = 0 to n – 1 do
      - $c(x,y) = c(x,y) + trace\left(x + \frac{p+\epsilon}{n}, y + \frac{q+\epsilon}{n}\right)$
    - $c(x,y) = c(x,y)/n^2$
- This is a hybrid approach between the regular and random sampling





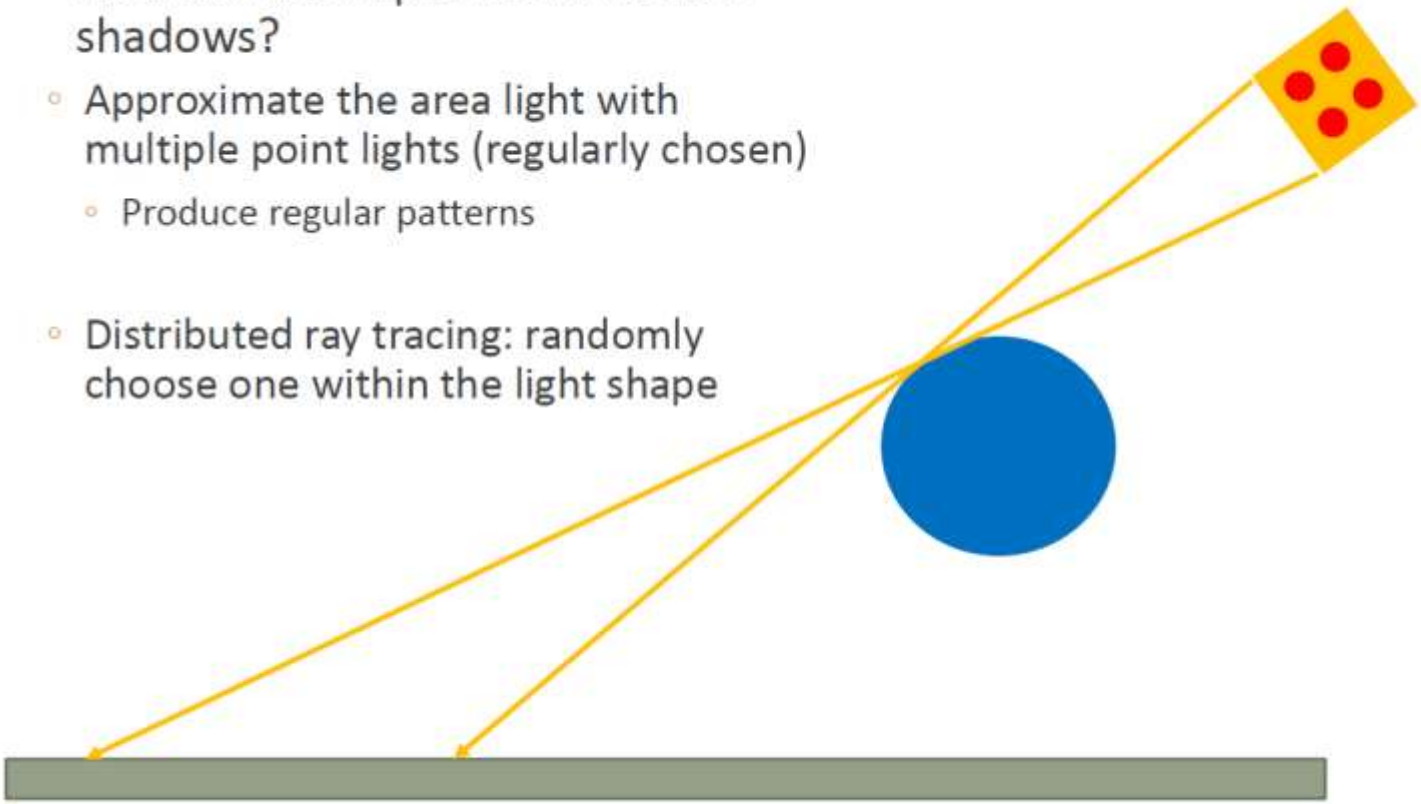
Soft Shadows





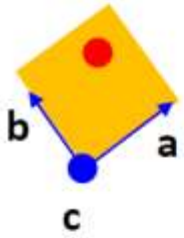
Soft Shadows

- How can we implement the soft shadows?
  - Approximate the area light with multiple point lights (regularly chosen)
    - Produce regular patterns
  - Distributed ray tracing: randomly choose one within the light shape

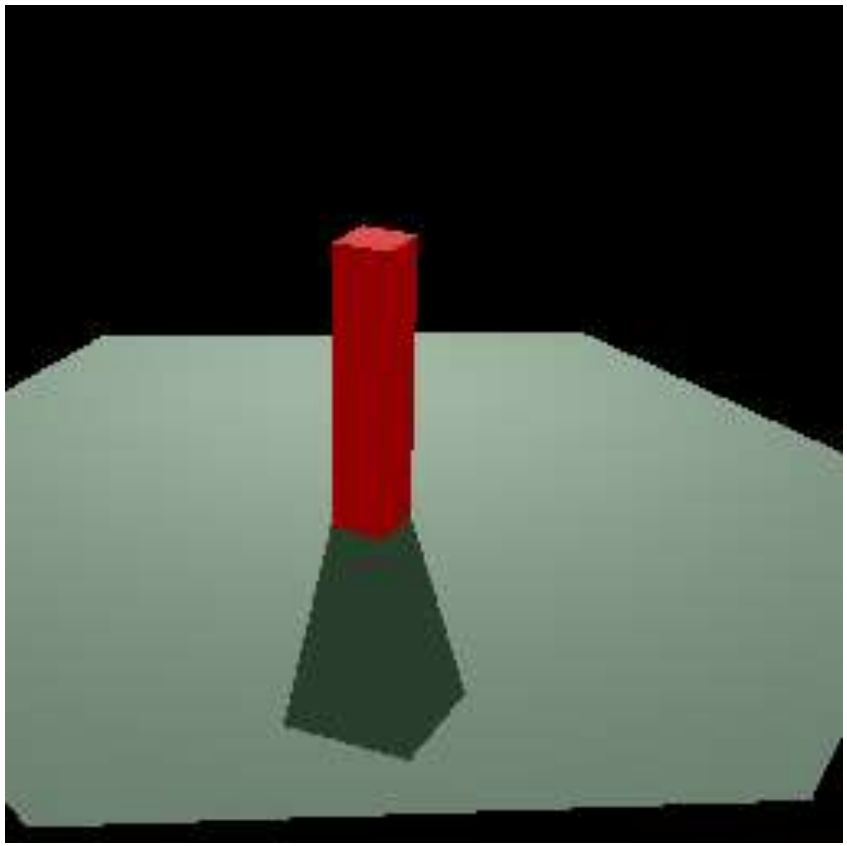


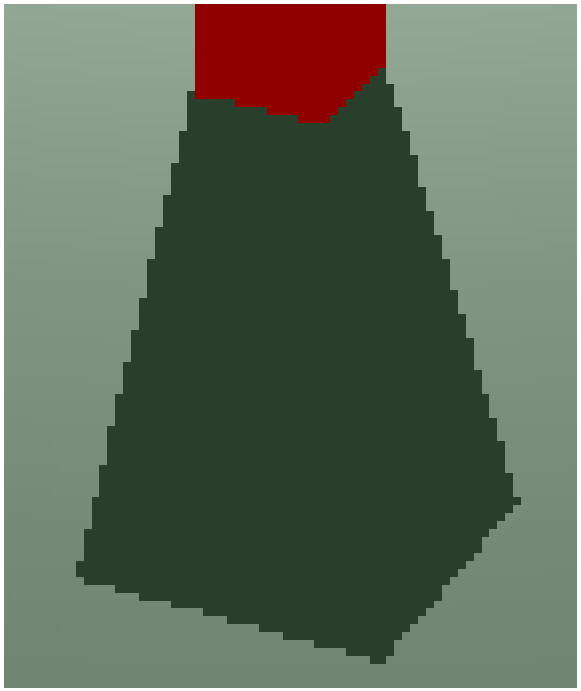
Soft Shadows

- e.g. area light defined as a parallelogram
  - Select a random point
    - $\boldsymbol{l}_{pos} = \boldsymbol{c} + \varepsilon_1 \boldsymbol{a} + \varepsilon_2 \boldsymbol{b}$
    - $\varepsilon_1, \varepsilon_2 \in [0,1)$
  - Generate a shadow ray from this point

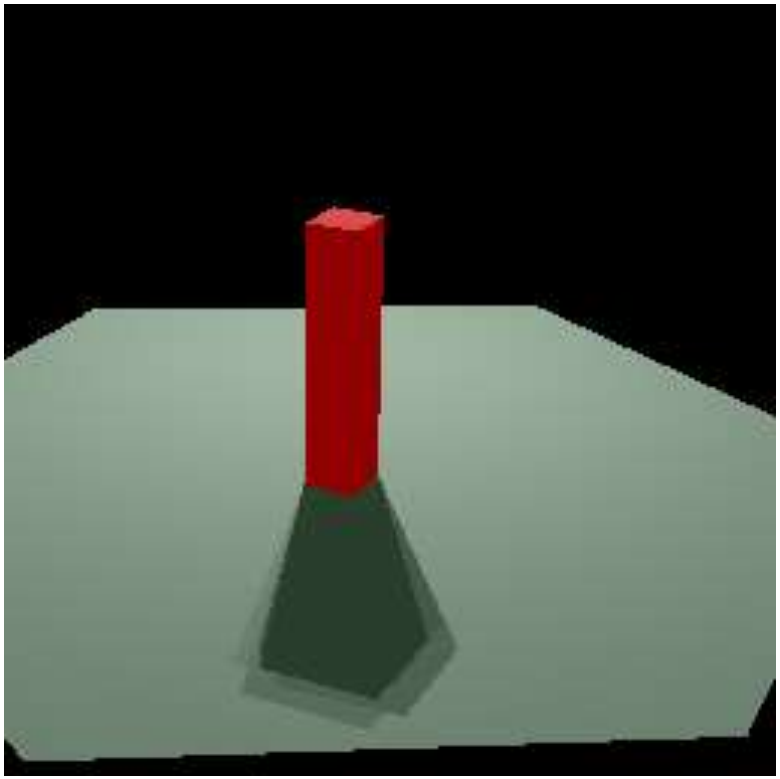


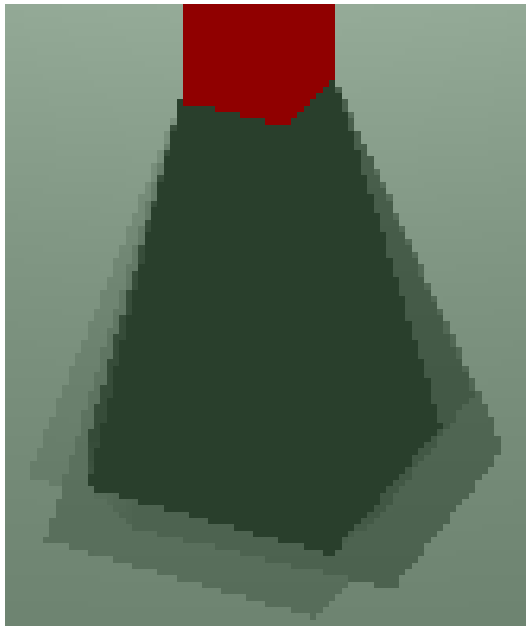
Soft Shadows



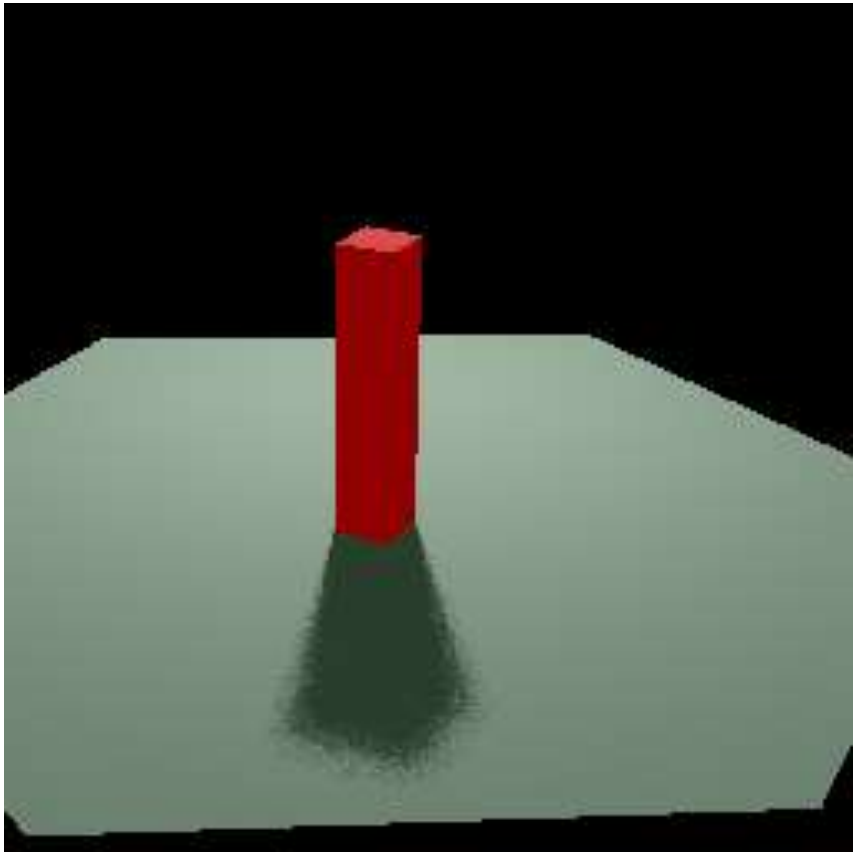


Soft Shadows

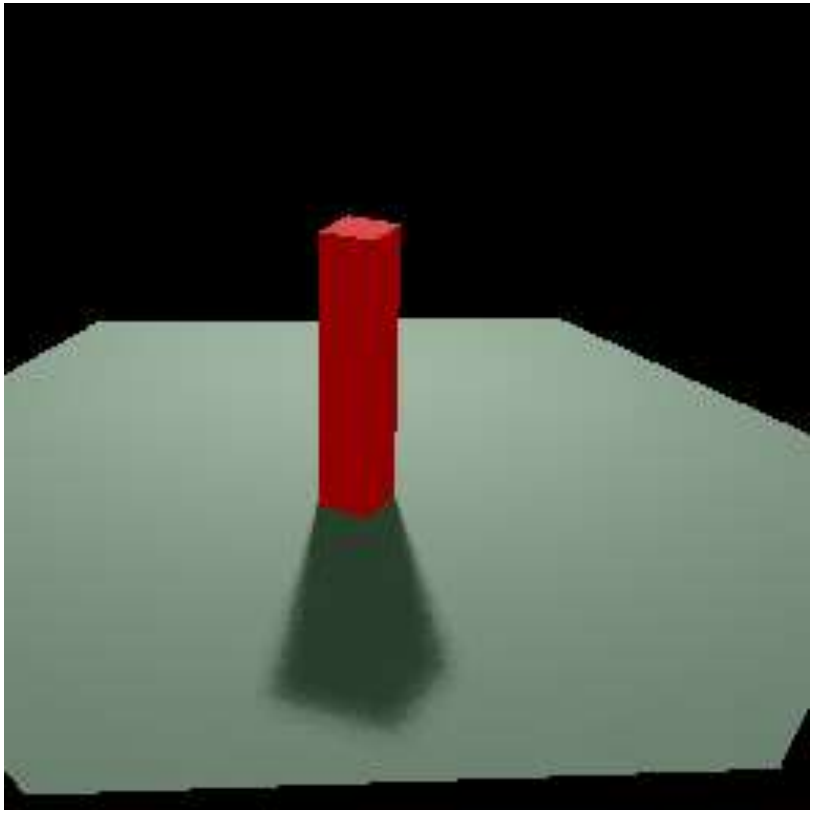


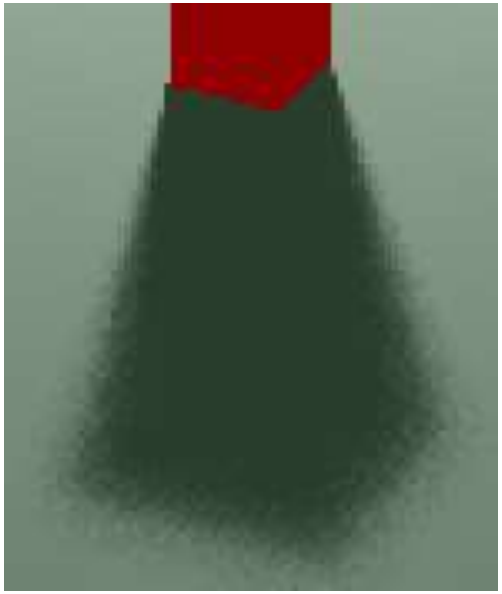


Soft Shadows



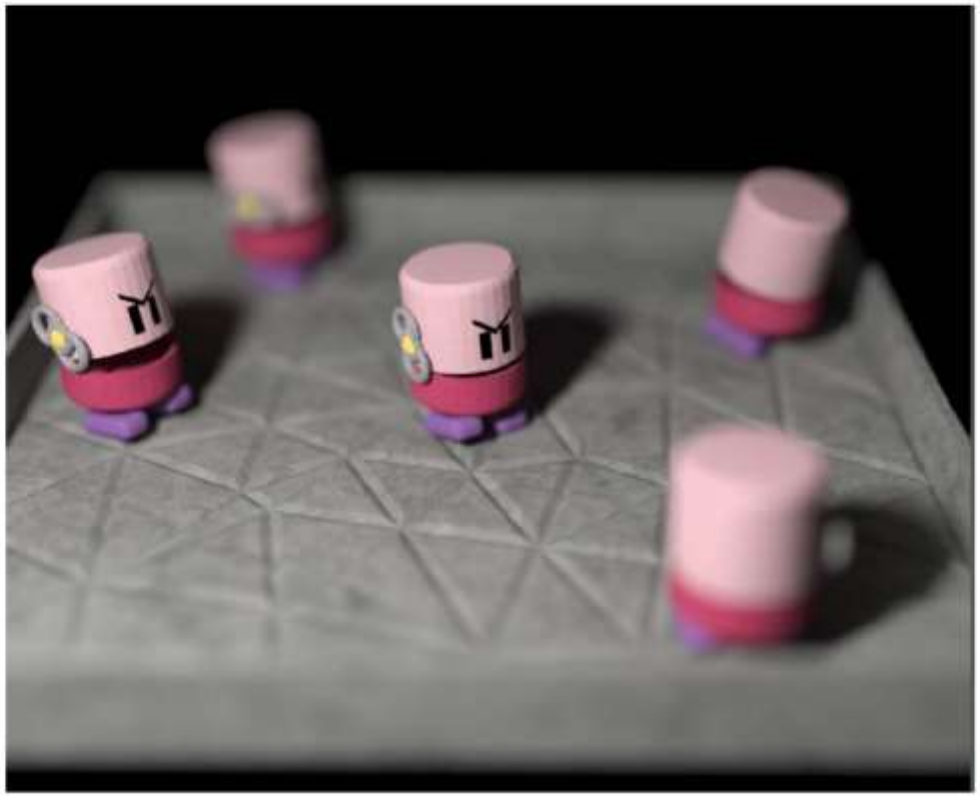
Soft Shadows





Depth of Field

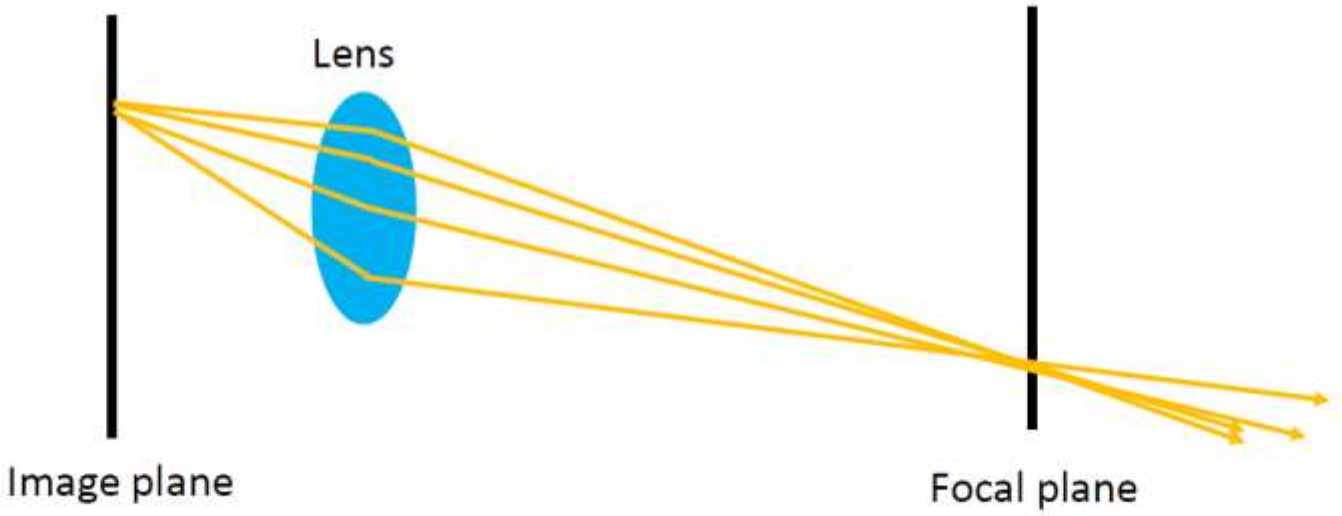
- This effect occurs when using a nonzero size "lens"
  - Approximate real cameras that have lenses with focal lengths



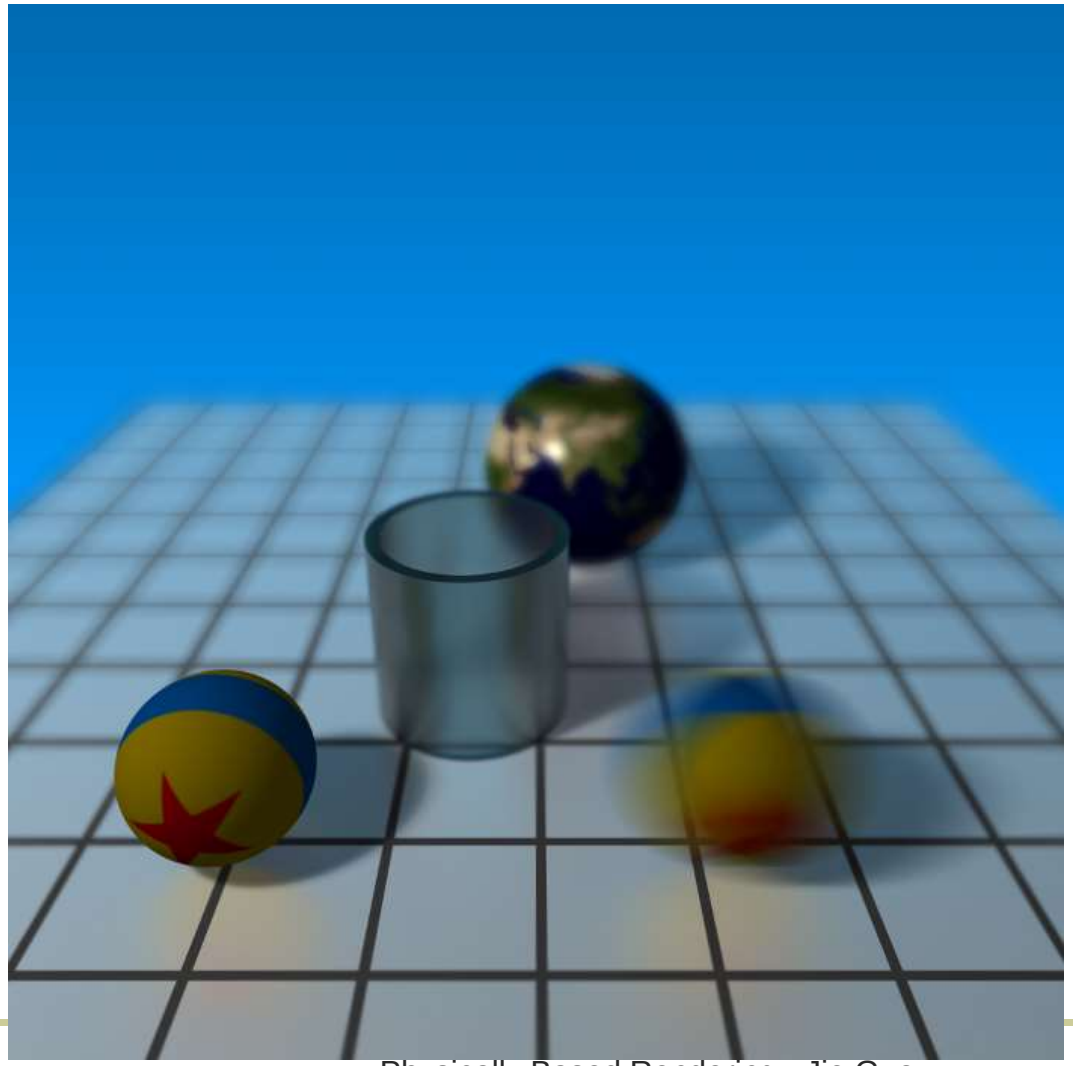
[Moon et al. 15]

Depth of Field

- This effect occurs when using a nonzero size “lens”
  - Approximate real cameras that have lenses with focal lengths
  - Can model the lens with a disk



DoF and Motion Blur

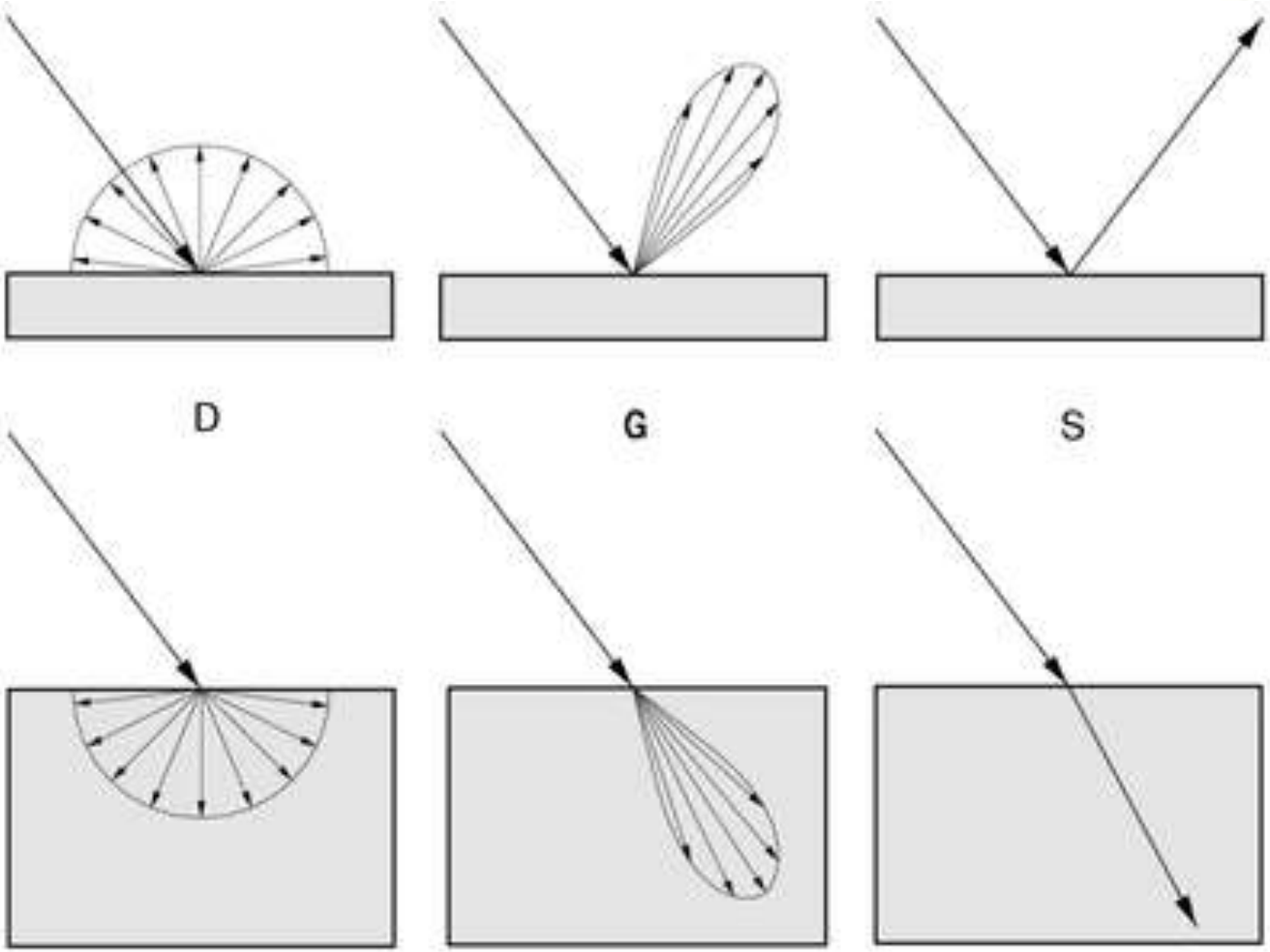


Gloss

- Standard ray tracing produces sharp reflections (mirror like).

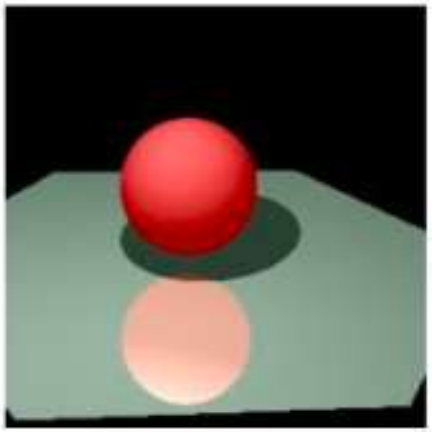
- Distributed ray tracing uses multiple rays around the reflected ray to produce a blur image of the reflected object.

A Note on Material

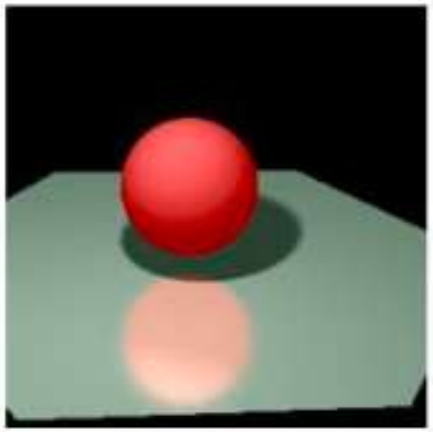


Gloss

Sharp VS Fuzzy Gloss

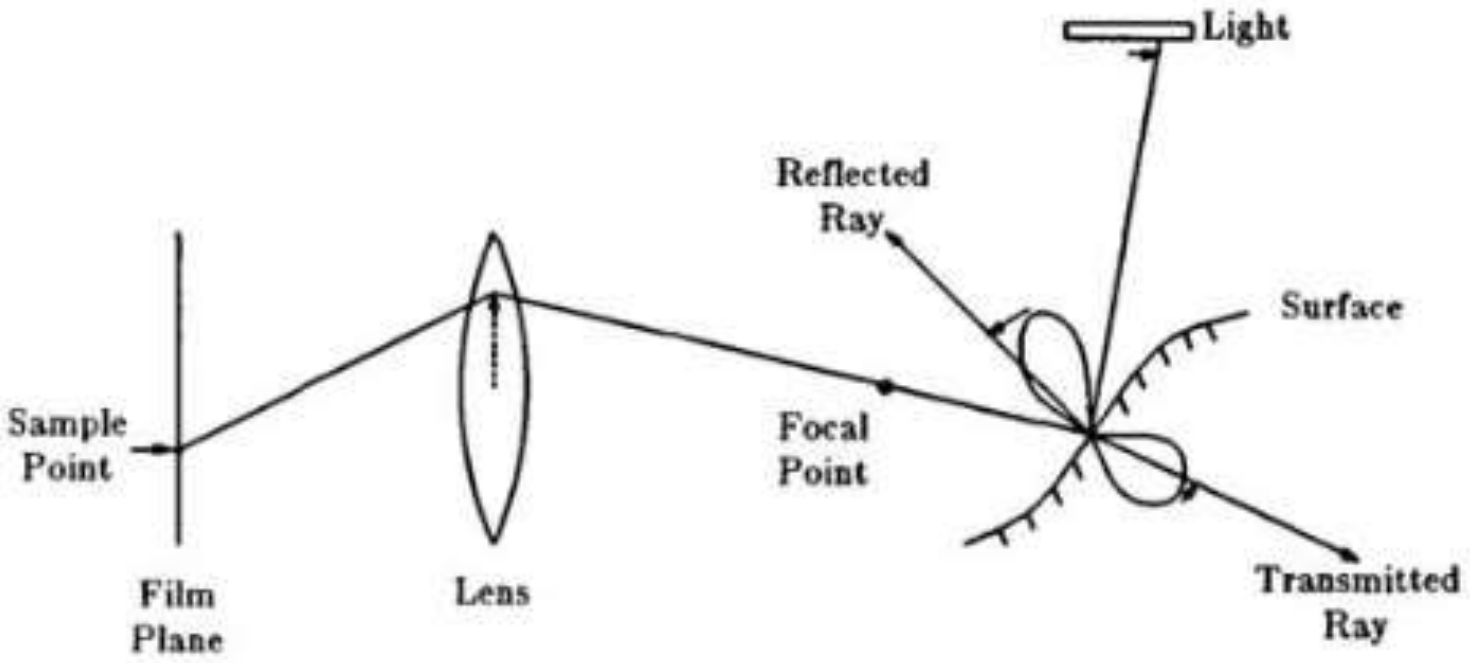


Standard Ray Tracer



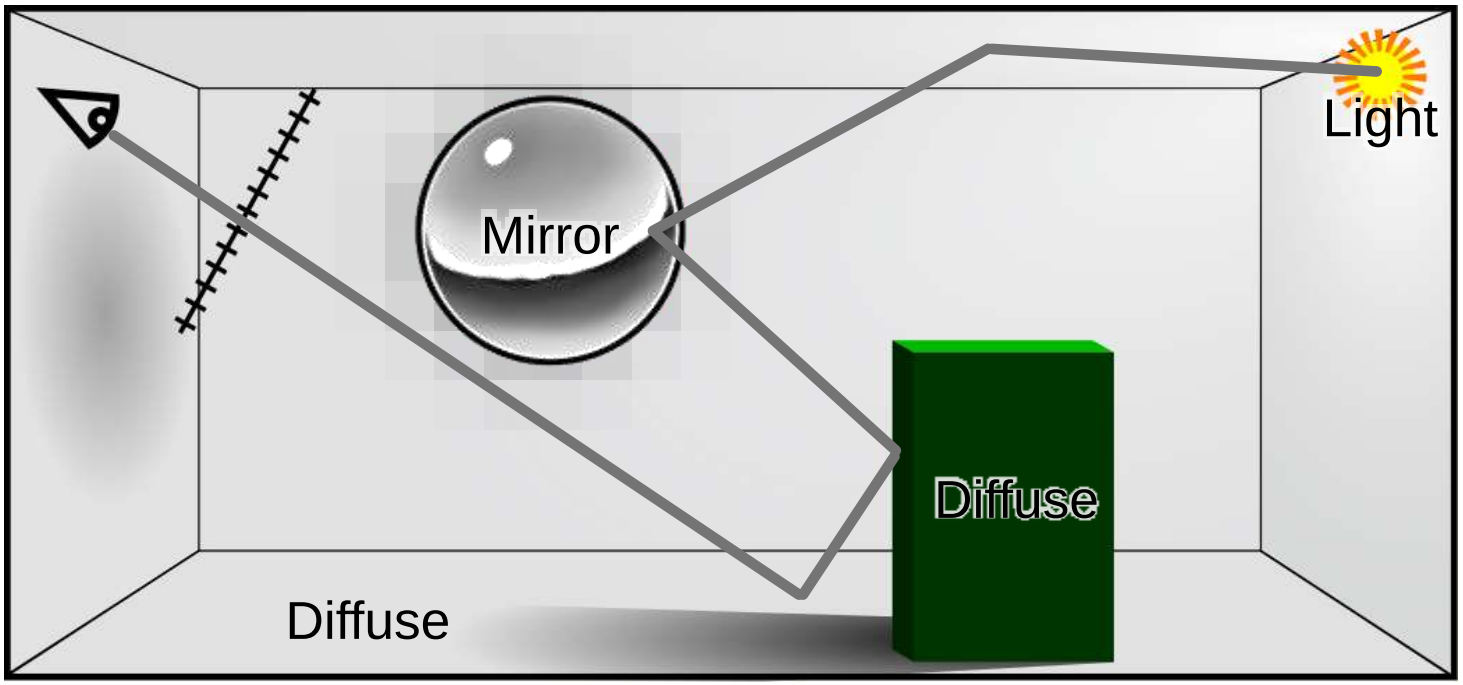
Distributed Ray Tracer
(50 Rays)

Distributed Ray Tracing: All in One



More General? Path Tracing

- Path tracing extends distributed ray tracing by stochastically sampling all possible light paths



Parallelization

- Ray tracing is inherently parallel, since the rays for one pixel are independent of the rays for other pixels
- Can take advantage of modern parallel CPU/GPU/Clusters to significantly accelerate a ray tracer
  - Threading (e.g., Pthread, OpenMP) distributes rays across cores
  - Message Passing Interface (MPI) distributes rays across processors on different machines
  - OptiX/CUDA distributes rays on the GPU

Parallelization

- Memory coherency helps when distributing rays to various threads/processors
  - Assign spatially neighboring rays (on the image plane) to the same core/processor
  - These rays tend to intersect with the same objects in the scene, and thus access the same memory

Real Time Ray-Tracing

- NVIDIA Optix…