CSC 4356
Interactive Computer Graphics
Lecture 24: Advanced Ray Tracing

Jinwei Ye

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

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

Distributed Ray Tracing (DRT)

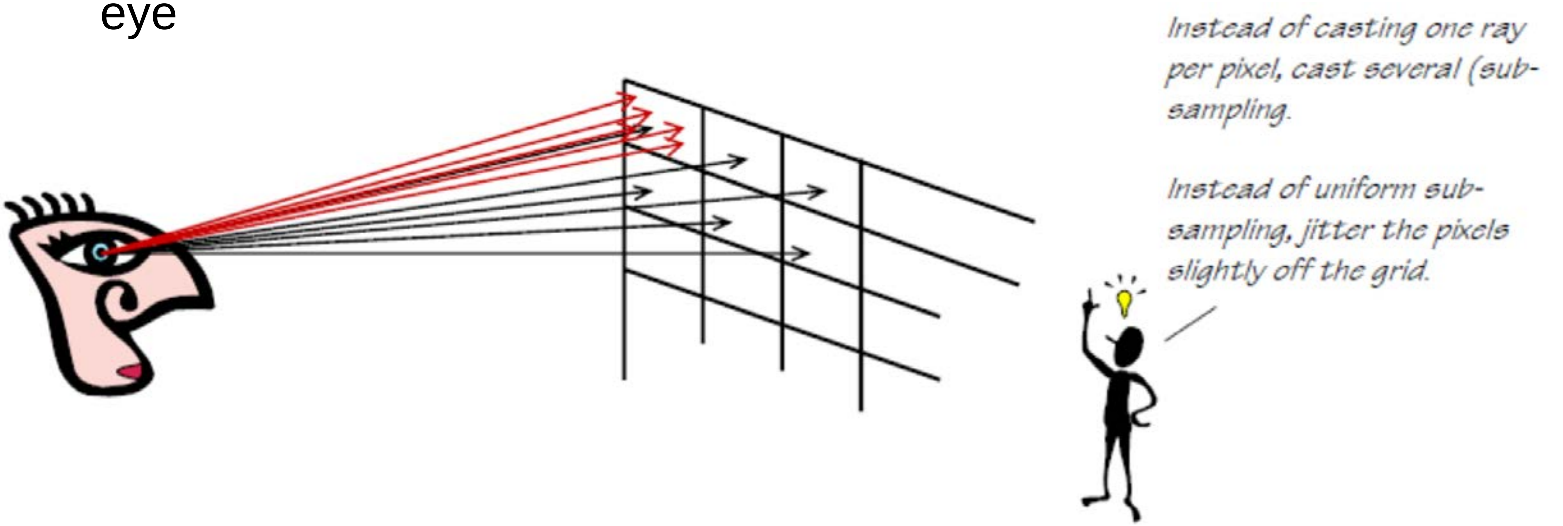
- Cook & Porter, in their classic paper “Distributed Ray Tracing” realized that ray-tracing, when combined with randomized sampling, which they called “jittering”, could be adapted to address a wide range of rendering problems:

Distributed Ray Tracing (DRT)

- **Main idea**: Replace our single ray approximations with a *distribution of rays*
- Improvements to this image:
  - Anti-aliased edges
  - Objects in/out of focus according to a lens
  - Motion blur of fast moving objects
  - Soft shadows
  - Glossy reflection
  - “Glossy” translucency

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

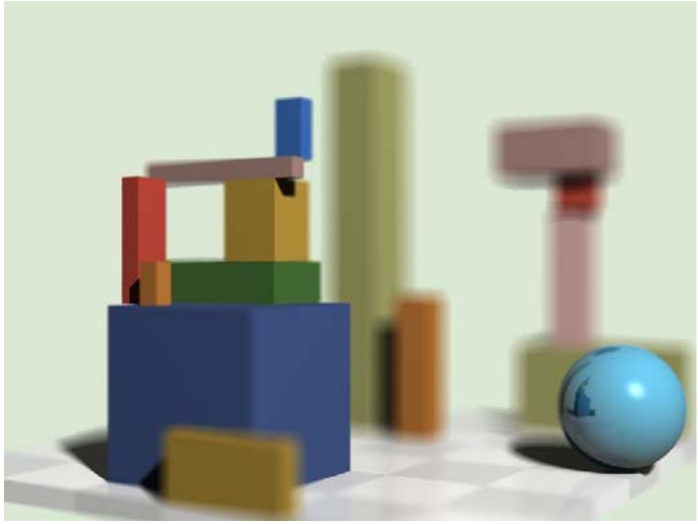


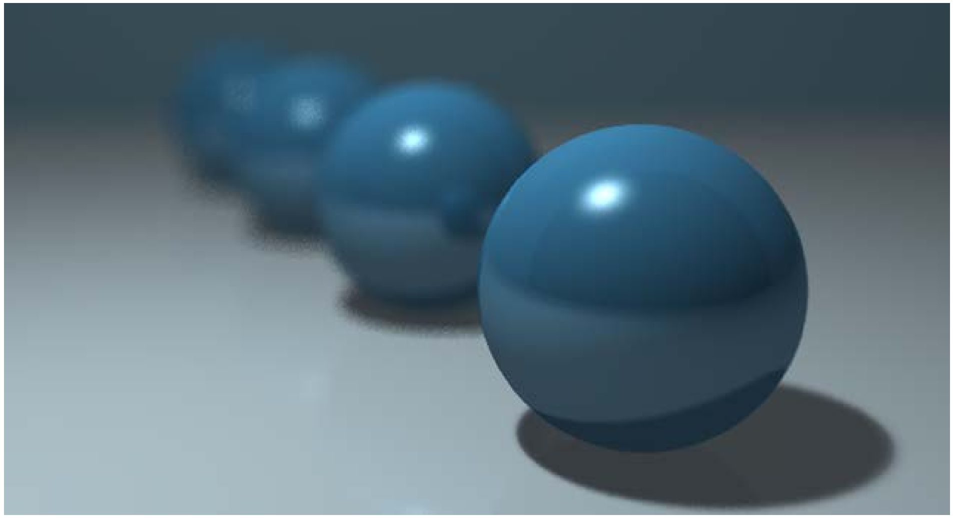
Camera Model

- Assumption: Pinhole camera model
  - **Realistic: Real cameras uses lenses to gather more light**

Depth-of-Field

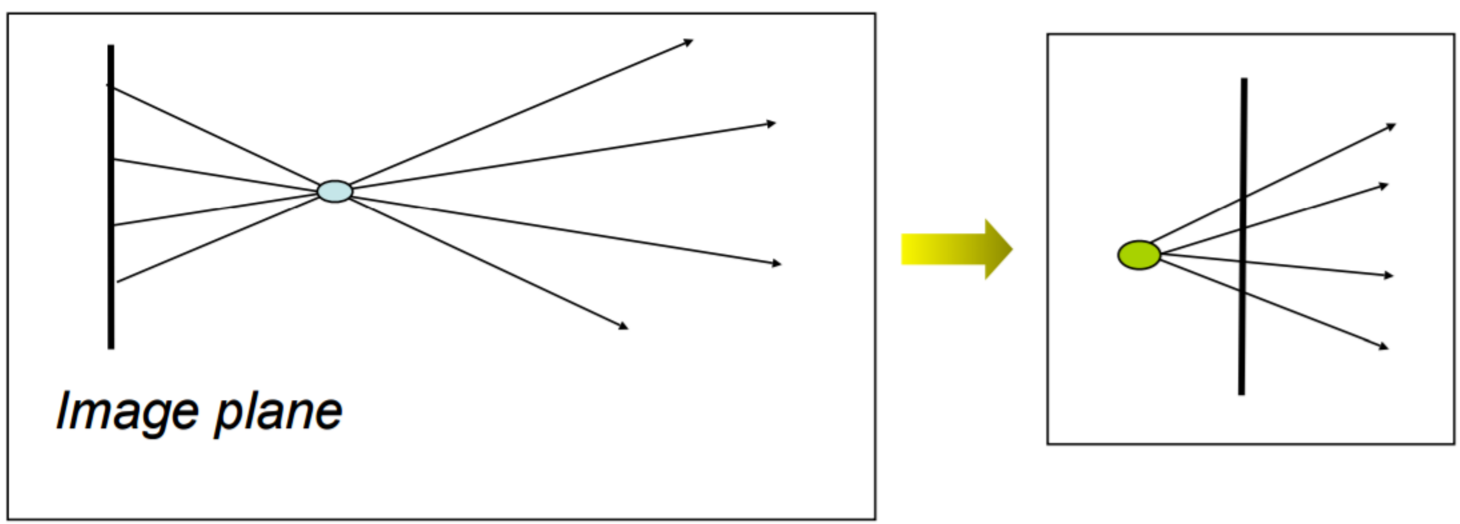
- Rays don’t have to all originate from a single point.
- Real cameras collects rays over an aperture
  - can be modeled as a disk
  - final image is blurred away from the focal plane.
  - gives rise to depth-of-field effects.





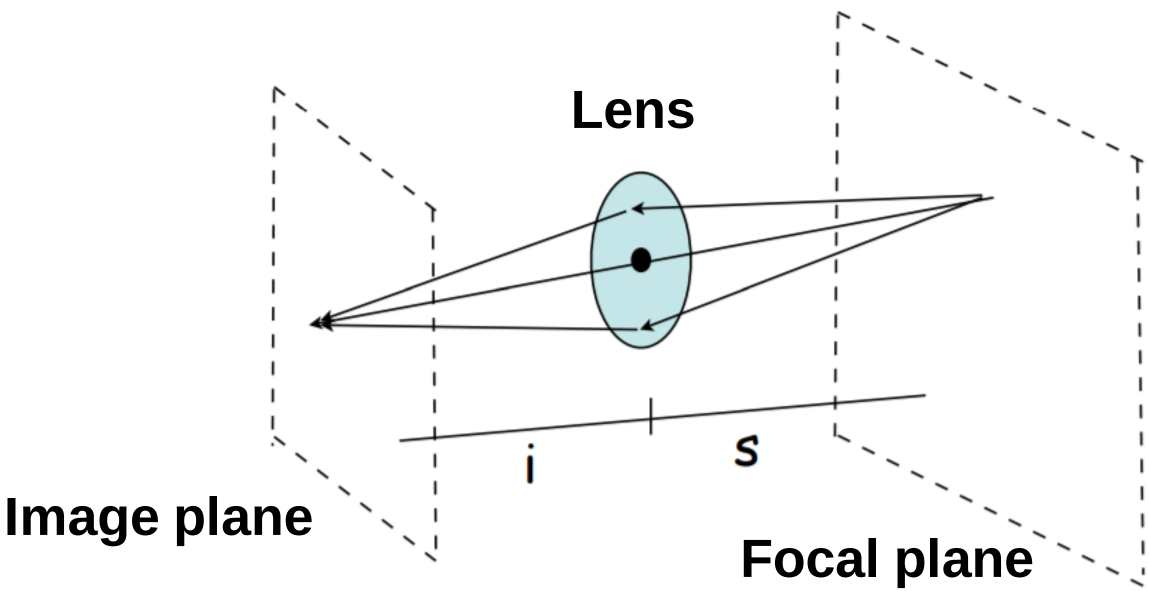
Pinhole Camera

- When using pinhole camera, the “lens” is just a point to project light from the scene onto the image plane
  - Everything is in focus



Thin-lens Camera

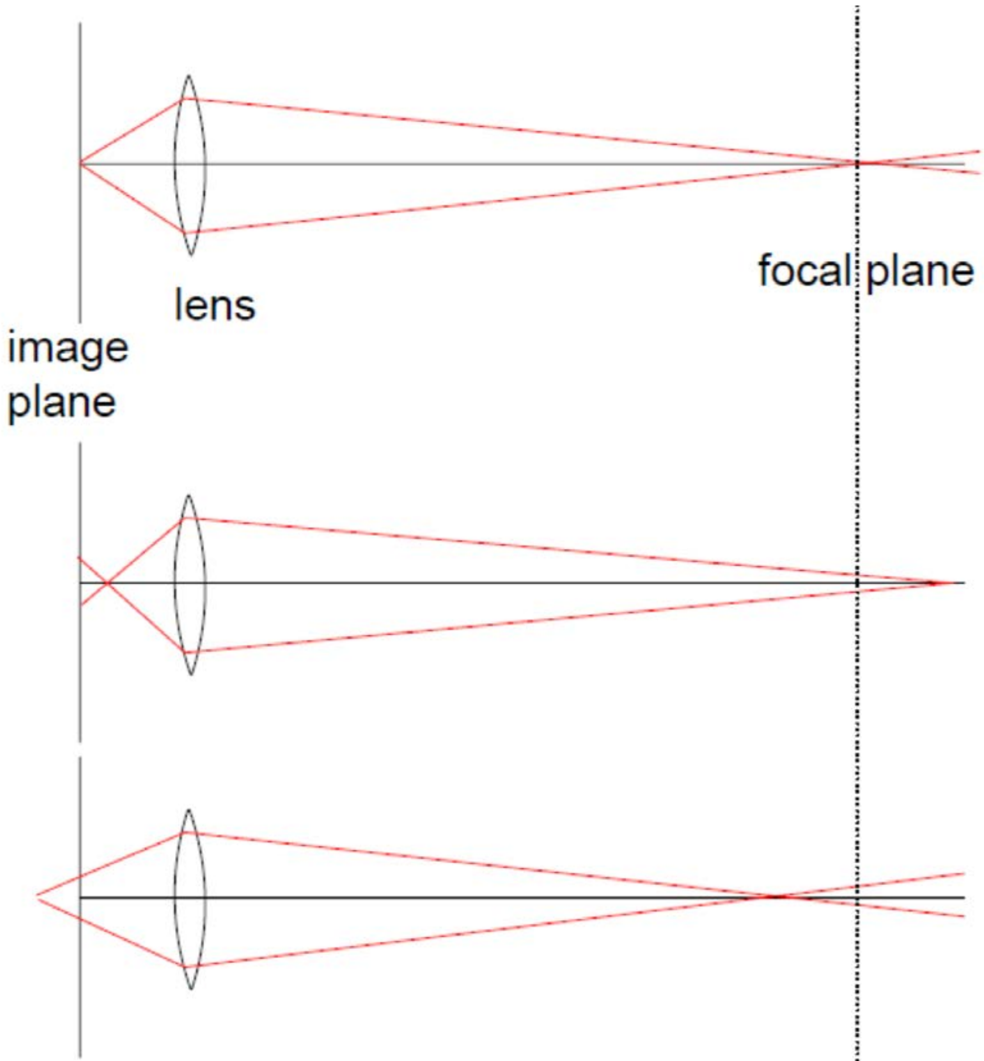
- Depth-of-field can be simulated using a thin-lens camera
- A thin-lens camera replace the pinhole by a disk-shaped thin-lens
  - – Lens lets more light into the camera



1/s+1/i = 1/f
f: focal length

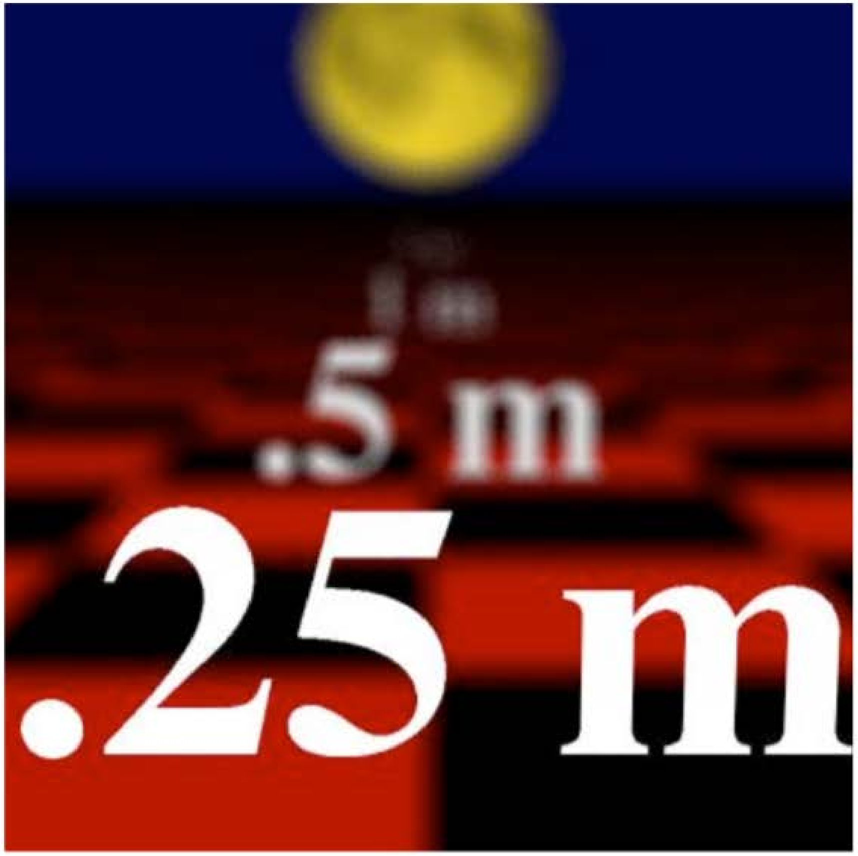
when s = infinity
f = i

Defocus Blur



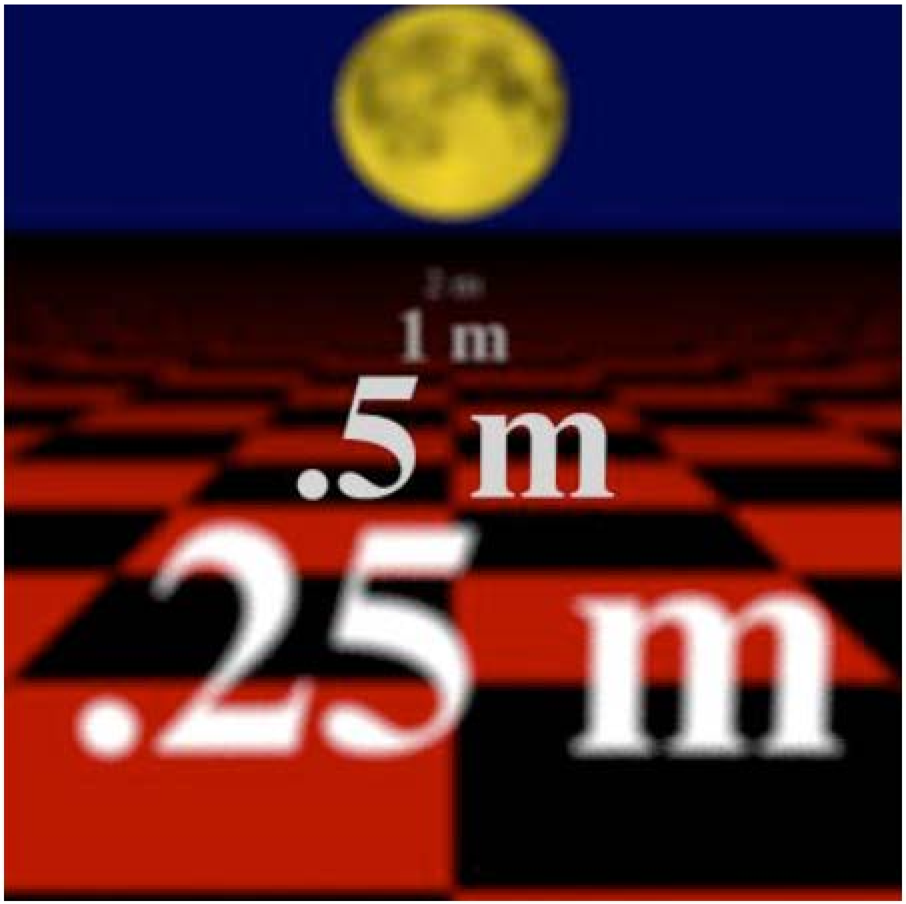
Changing the Focal Length

Pinhole Camera

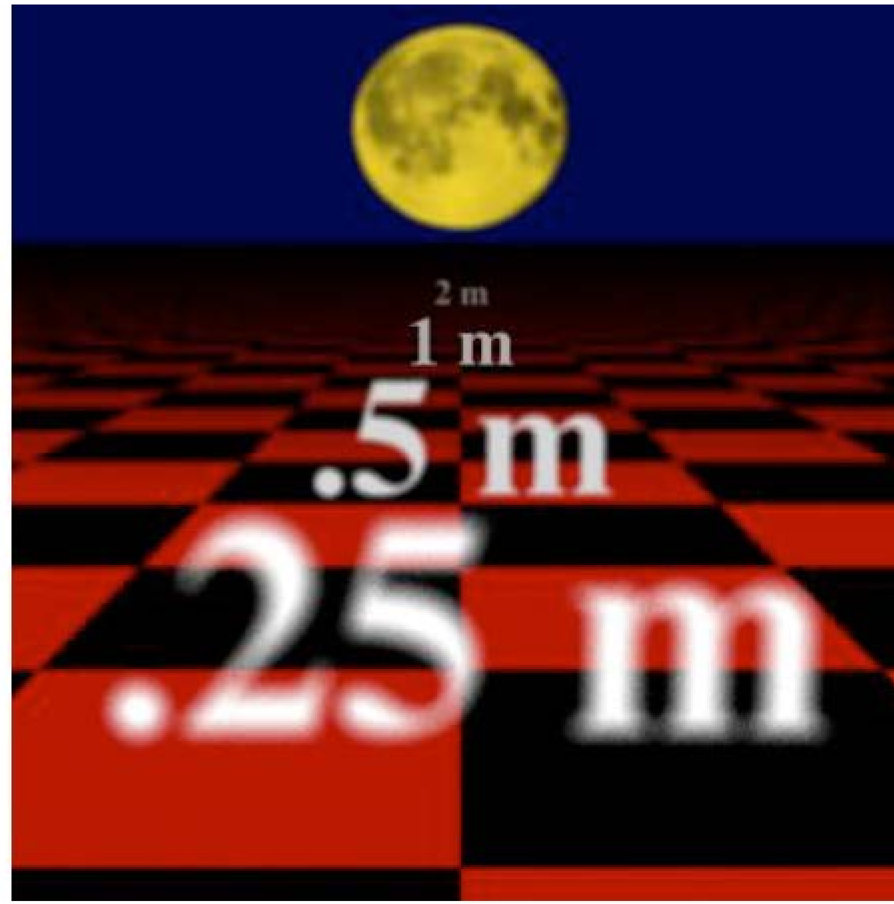


.25 m Focal Length

Changing the Focal Length

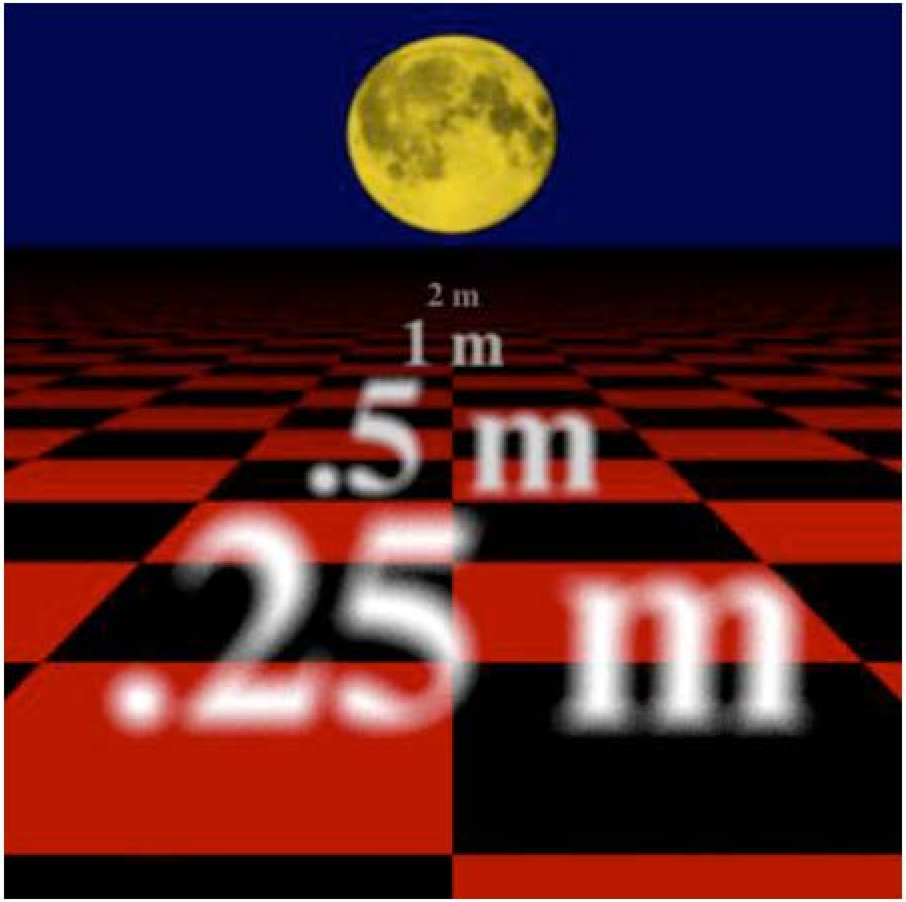


0.5 m Focal Length

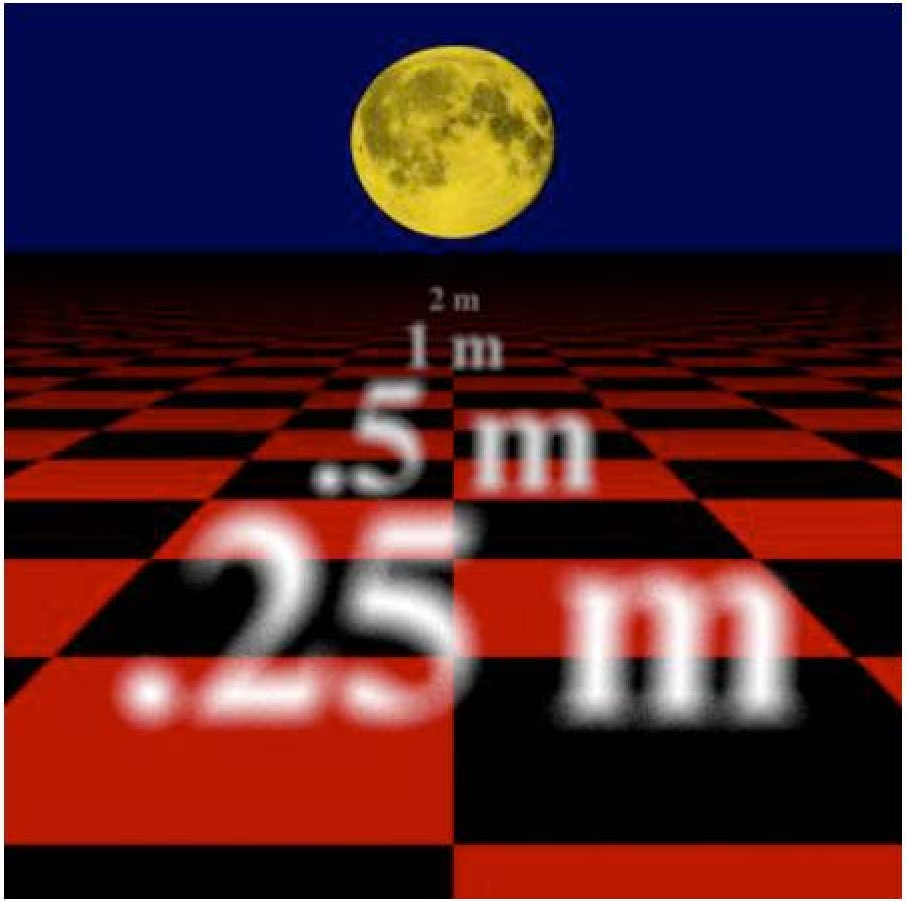


1 m Focal Length

Changing the Focal Length



2 m Focal Length

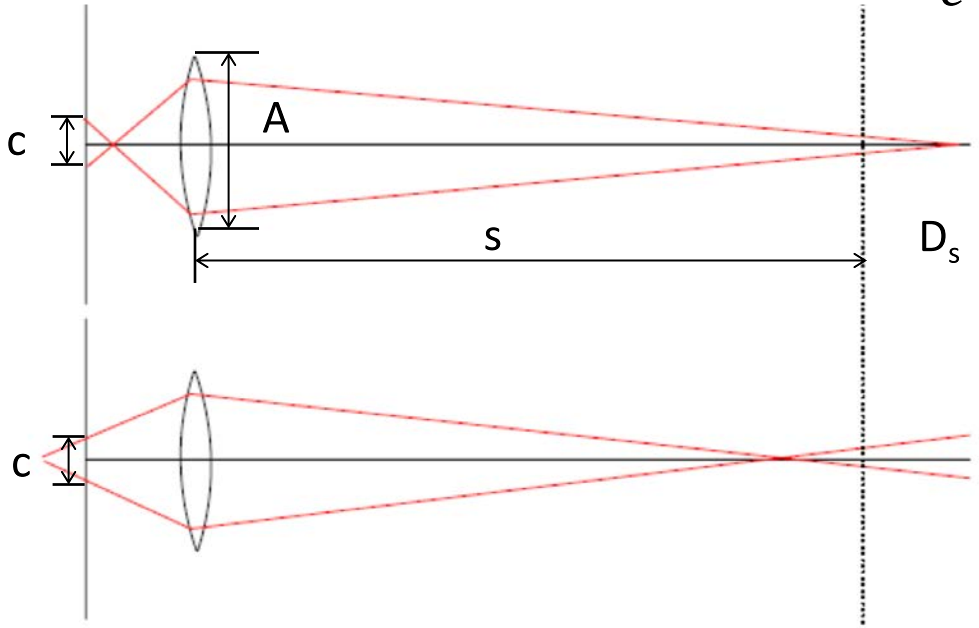


Infinite Focal Length

Circle-of-Confusion

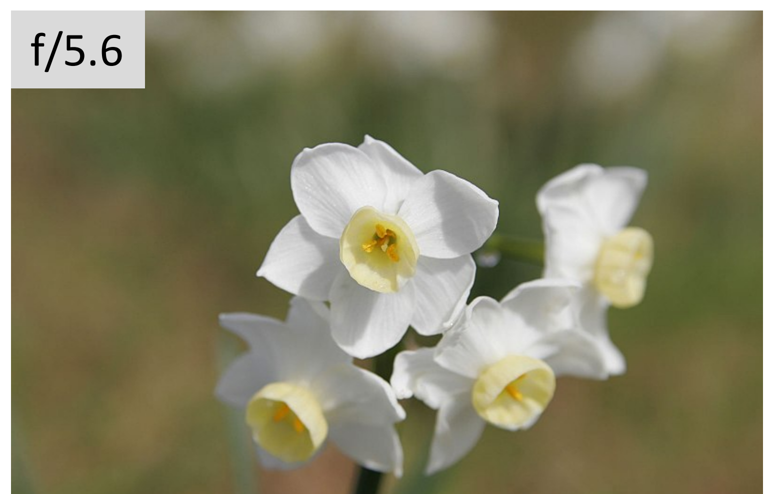
- The circle of confusion determines a scene point's contribution to the image plane

$$c = \frac{D_s f}{s(s + D_s - f)} A$$



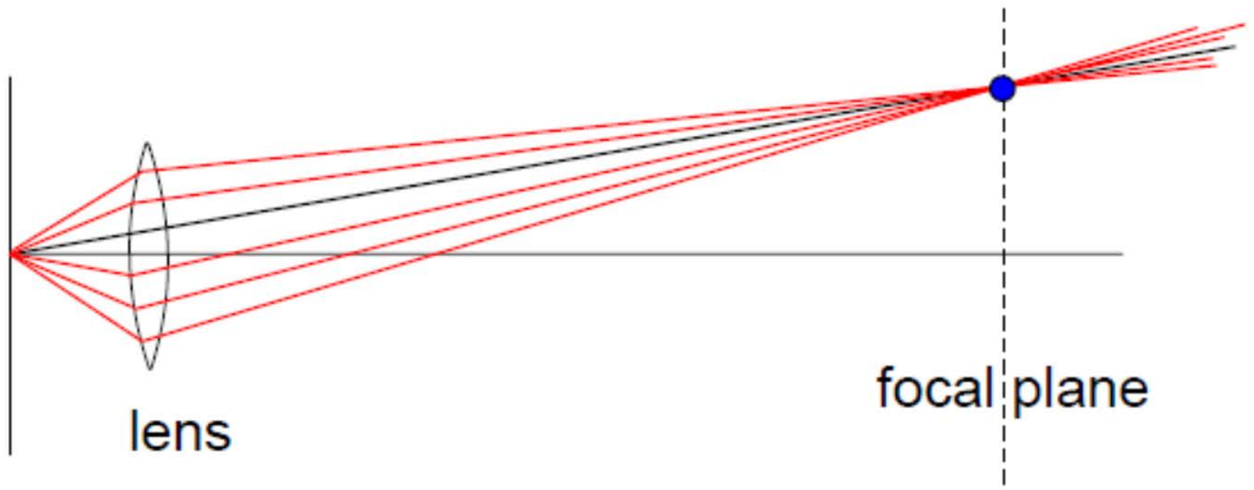
Depth-of-Field

- Depth-of-field: focus range or effective focus range
  - Circle-of-confusion smaller than a pixel
- Larger aperture size smaller (or shallower) depth-of-field



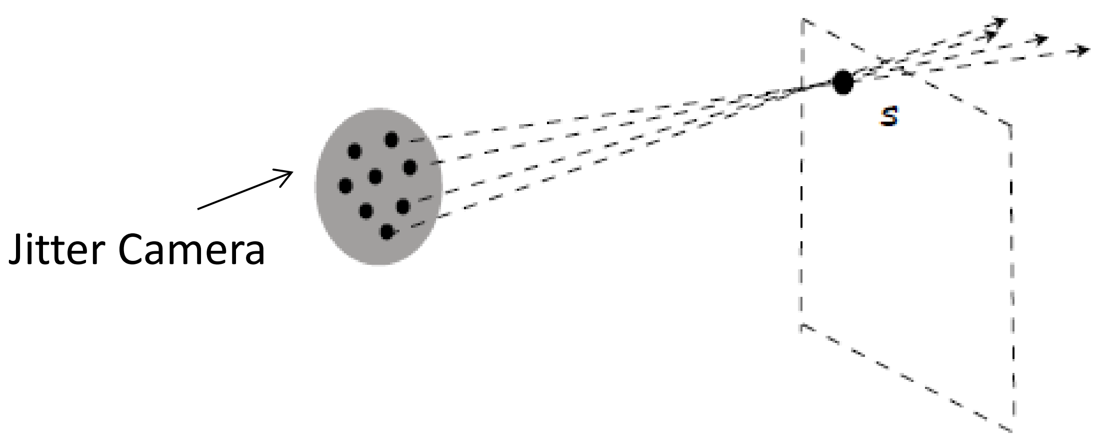
DRT for Depth-of-Field

- 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

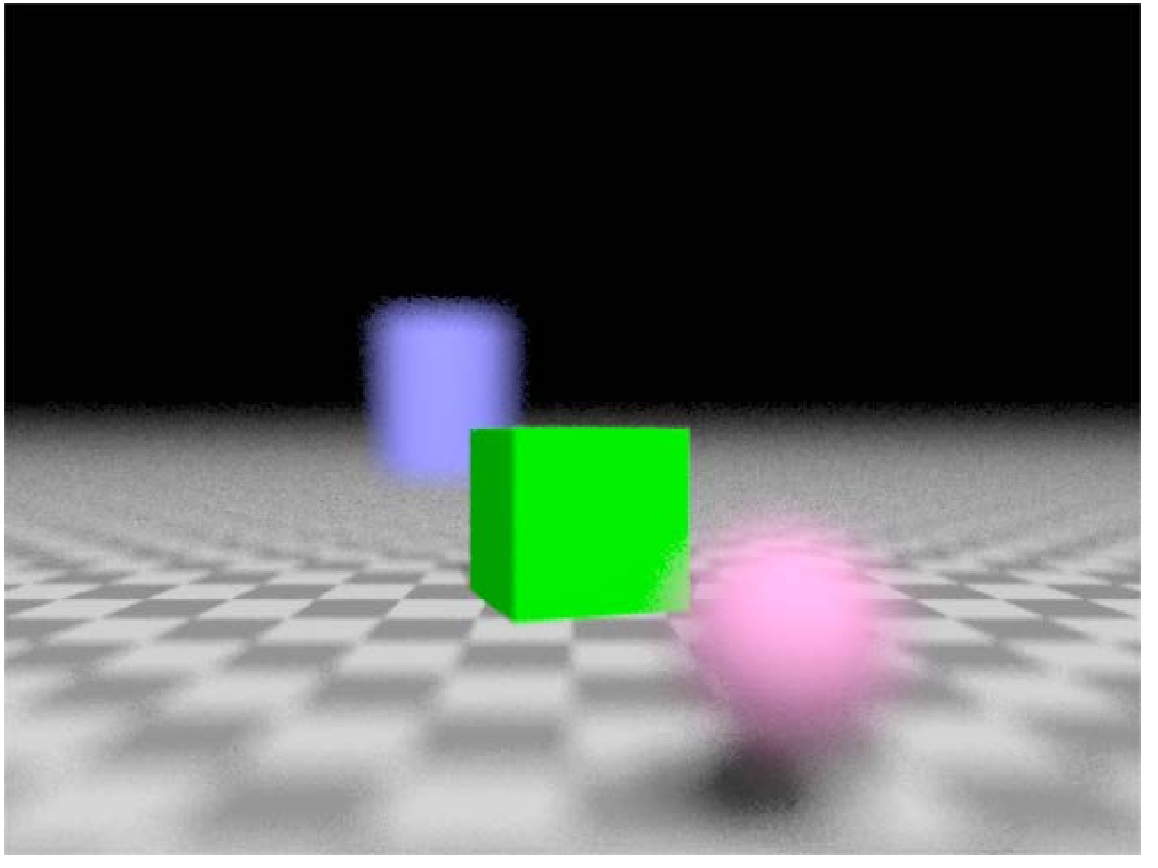


Implementation Details

- Place your image S distance away, where you have the complete focus
- Assume the radius of the lens is R, for each pixel, randomly select N points within a disk around the camera (the disk is perpendicular to the camera view direction). Use those N points as your camera position and shoot rays
- Average the N colors from the rays and assign it to the pixel



Depth-of-Field Example



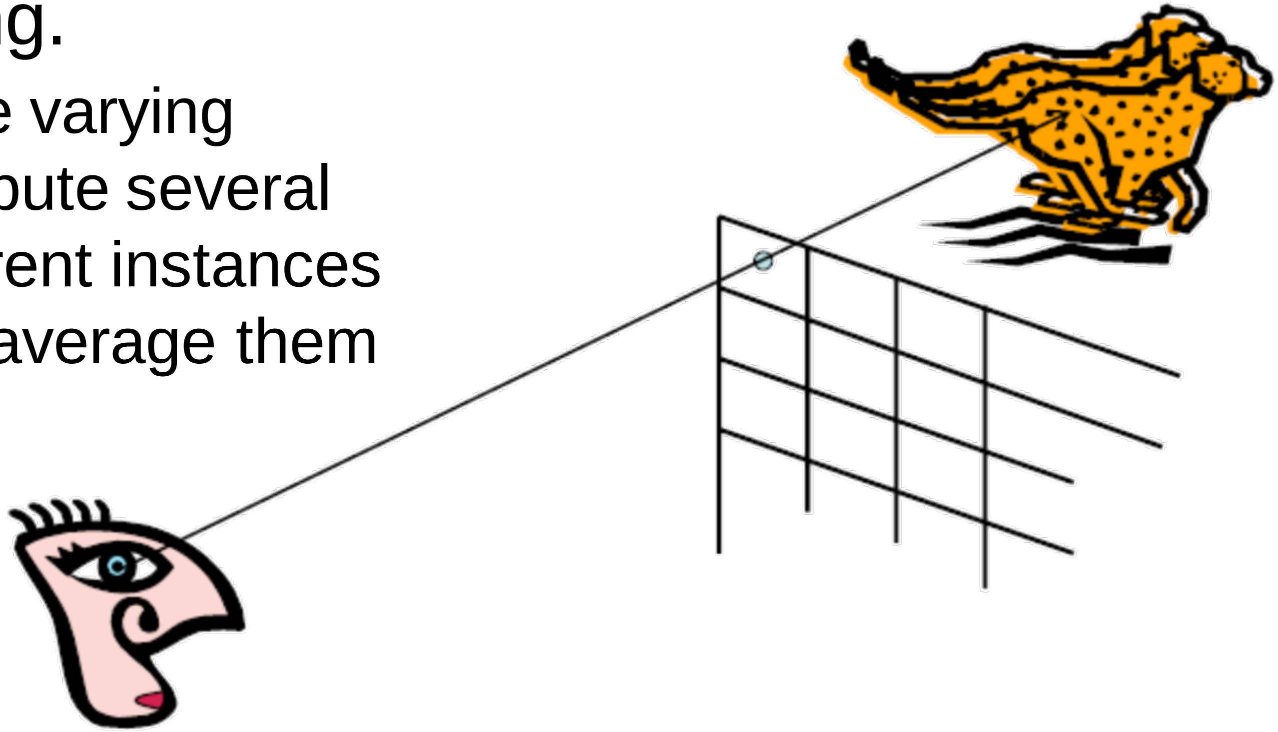
Rendered by Pov-Ray

Motion Blur

- Problem: Object (or camera) motion requires an exposure (samples over time or shutter speed) rather than a single sample in time

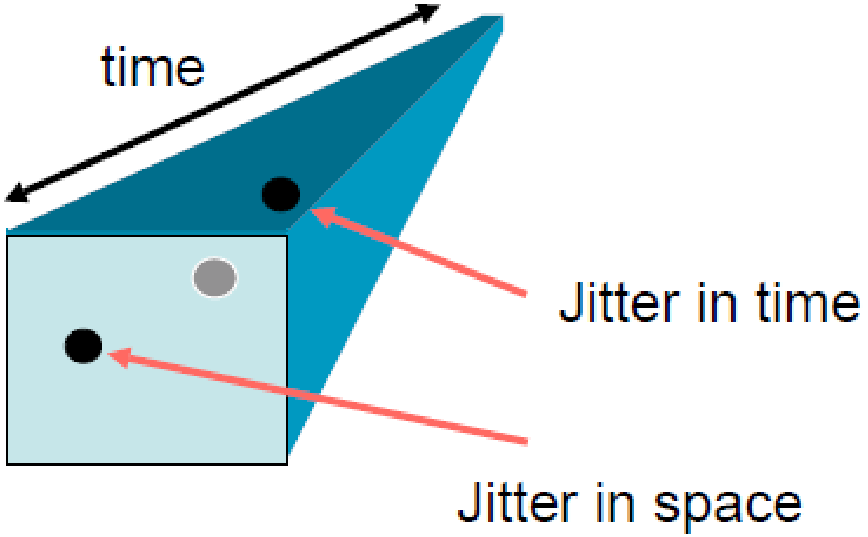
DRT for Motion Blur

- 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.

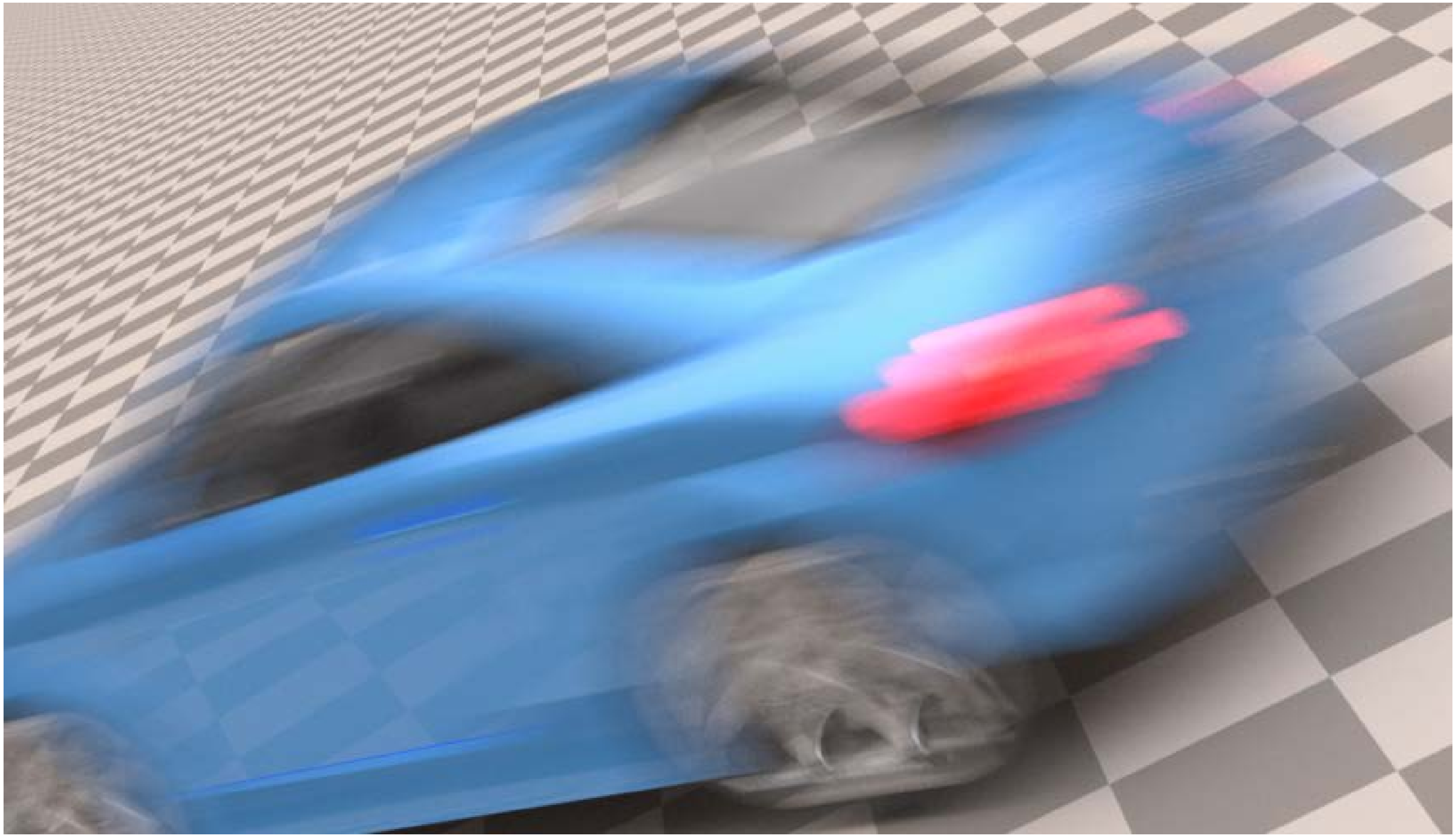


DRT for Motion Blur

- Jitter distributed rays over time
  - Sample objects temporally
  - $T = T0 + \xi(T1 - T0)$

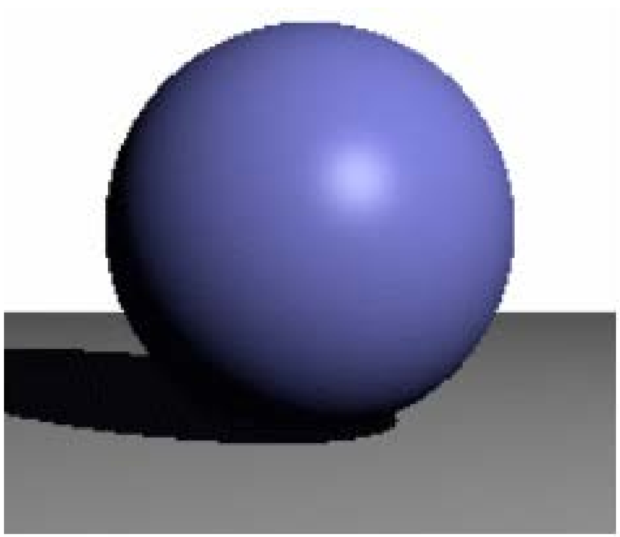


Motion Blur Example

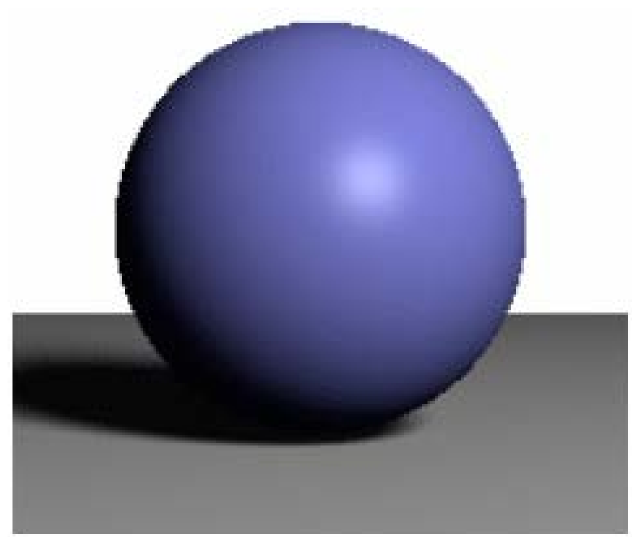


Shadow

- Assumption: The light source is a point
  - **Realistic: Soft shadows**



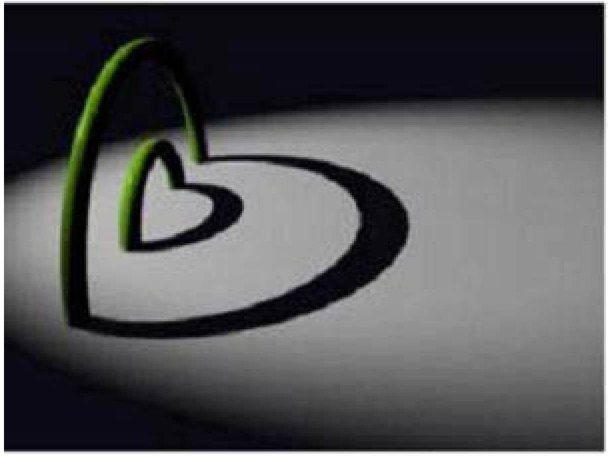
Point Light Source

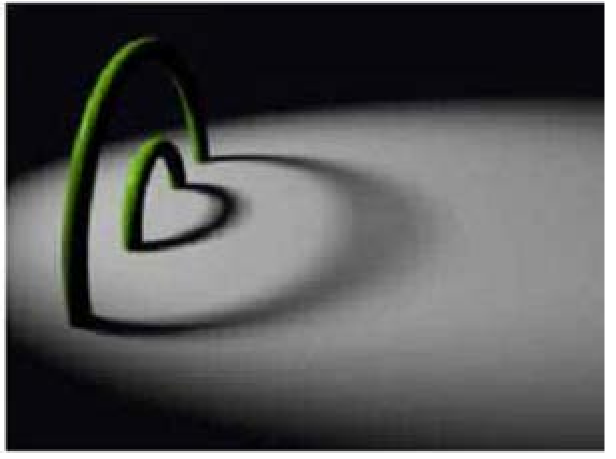


Area Light Source

Soft shadows

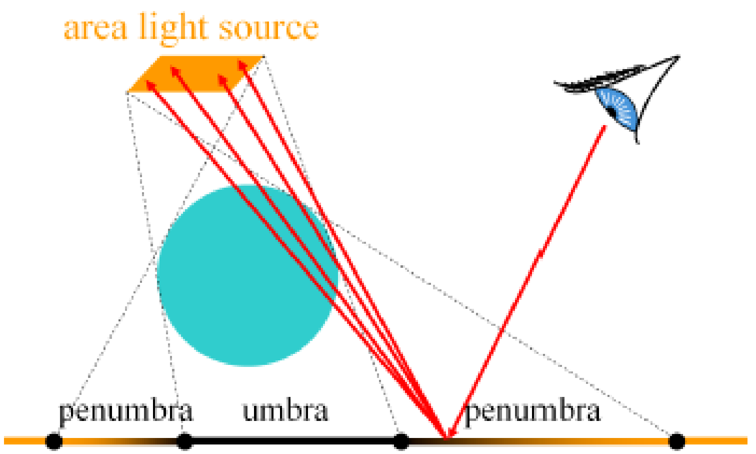
- For point light sources, sending a single shadow ray toward each is reasonable
  - But this gives hard-edged shadows
- Simulating soft shadows
  - Model each light source as sphere
  - Send multiple jittered shadow rays toward a light sphere; use **fraction** that reach it to attenuate color
  - Similar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



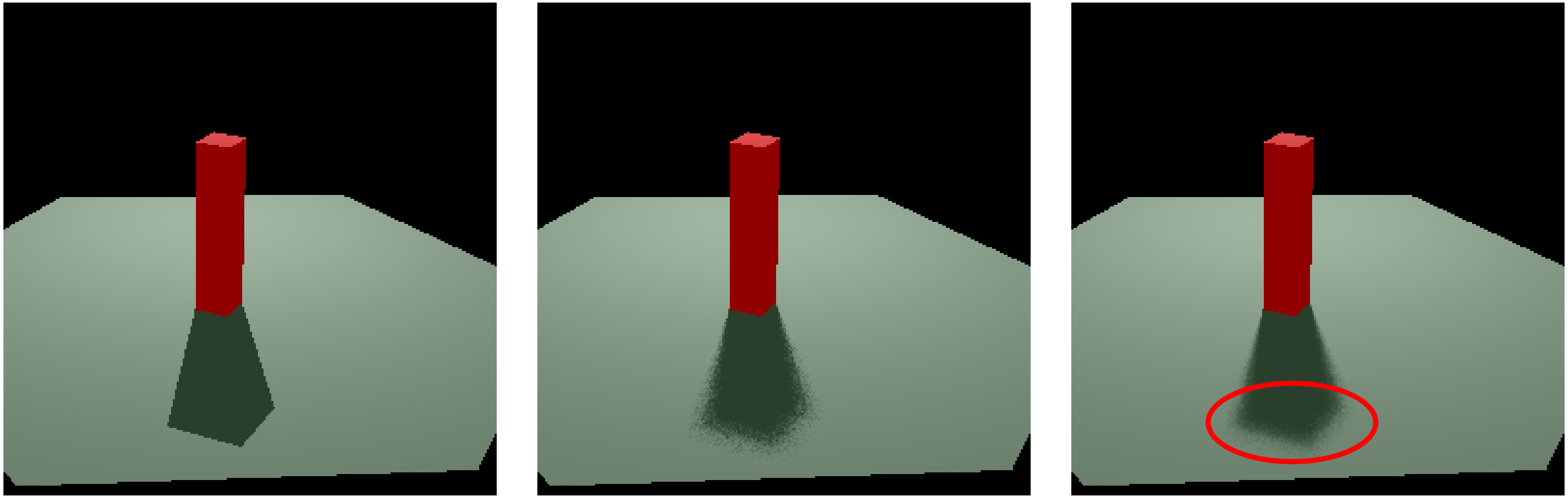


Sampling the Area Light

- Usually, the light source is modeled as a plane oriented towards the scene
- The light source may be treated as a sphere and random positions chosen on the sphere's surface to send a population of shadow rays
- Stochastic sampling on the light source's surface provides antialiasing in the soft shadow



Soft Shadow Examples

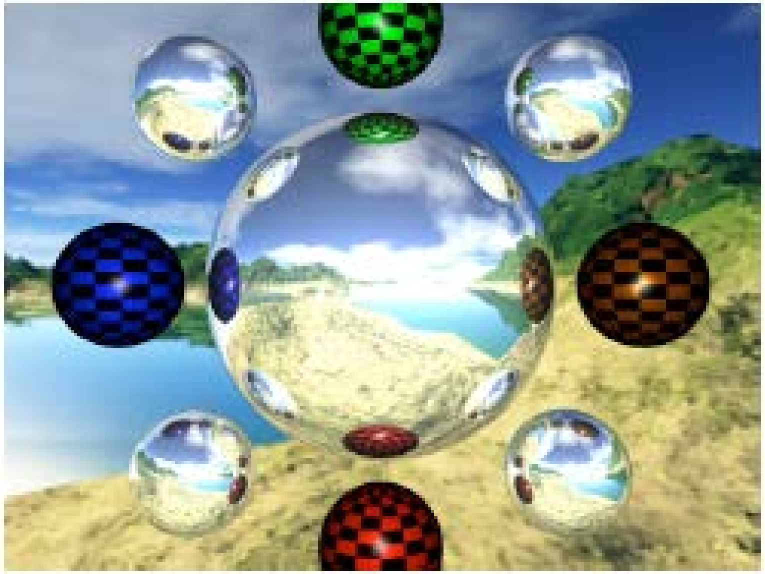


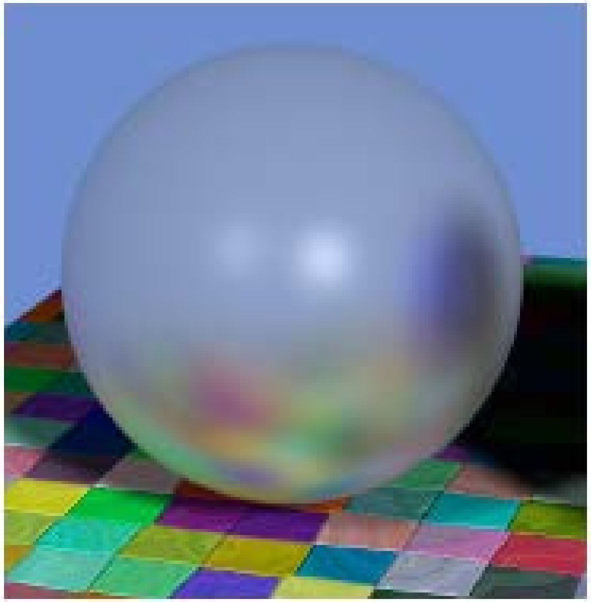
1 shadow ray   10 shadow rays   50 shadow rays

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

Reflections

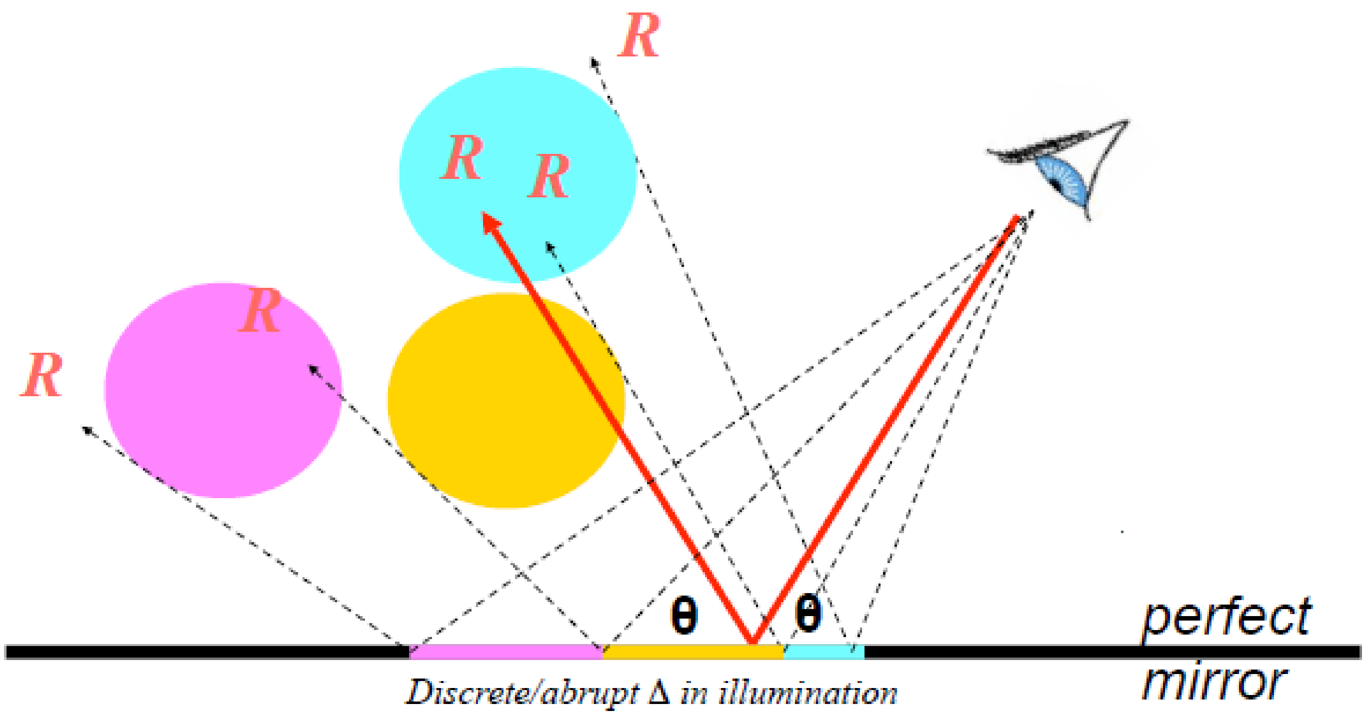
- Assumption: The surface is a perfect mirror, so the only reflection on a surface comes from the reflection vector
  - **Realistic: Glossy reflection**





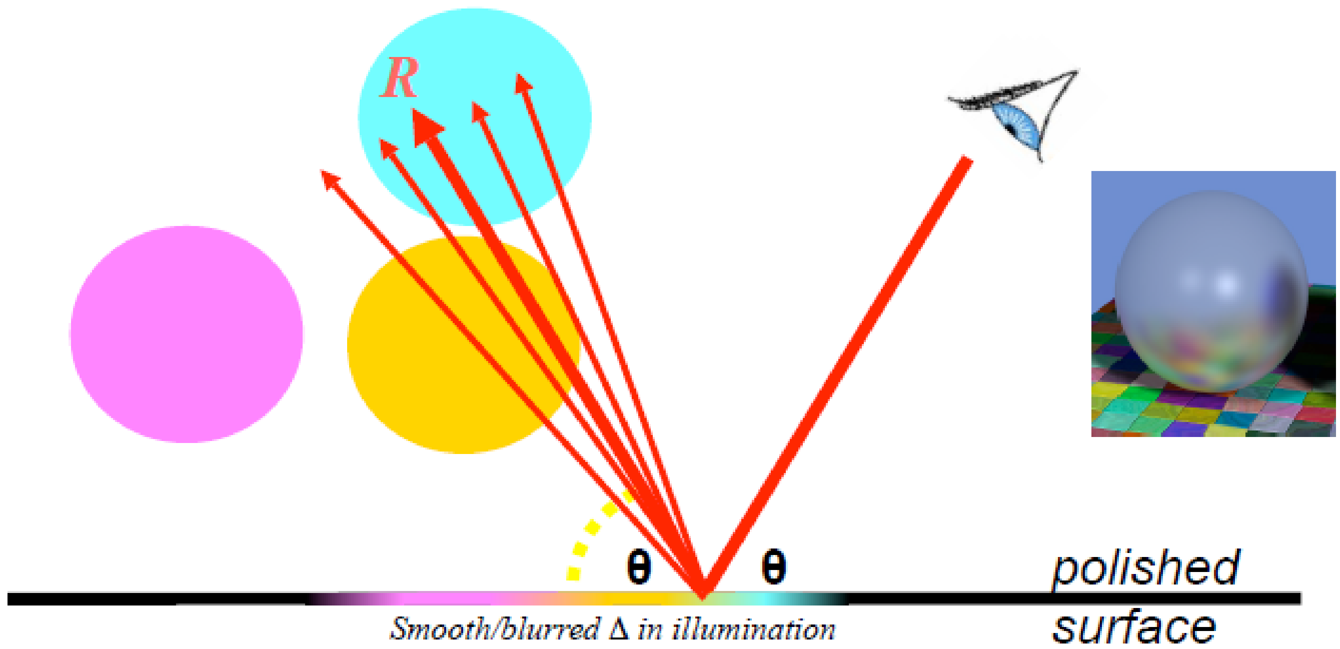
Mirror Reflections

- Contribution only comes from the reflection vector



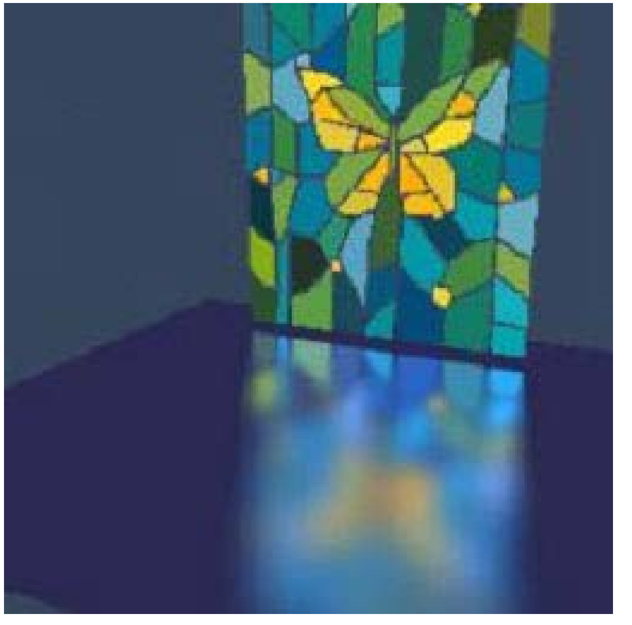
Glossy Reflection

- Integrate over additional rays defined about the reflection vector



DRT for Glossy Reflections

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



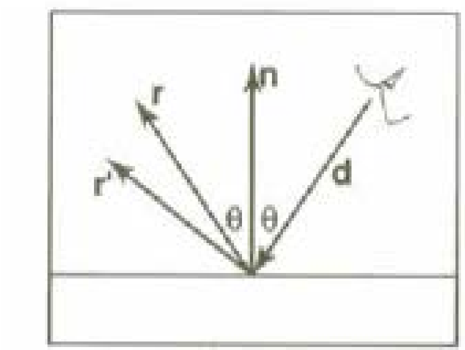
Why is the reflection sharper at the top?

Implementation

- Define each ray r' as a perturbation from r
- To do this:
  - Create an orthonormal uvw basis with w = r
  - Create a random point in the 2D square with side length a centered at the origin
  - Create u,v: u = -a/2 + ε a; v = -a/2 + ε' a with random ε and ε' in [0,1]
  - Then r' = r + u u + v v

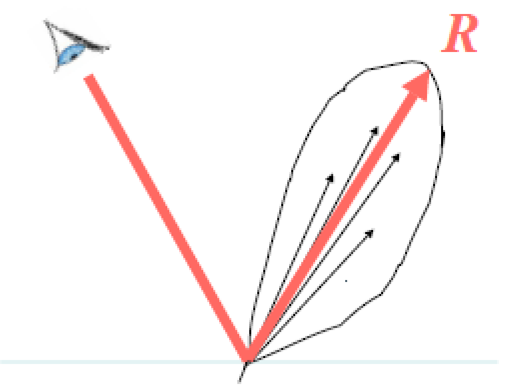
a – blur control

ξ - random value



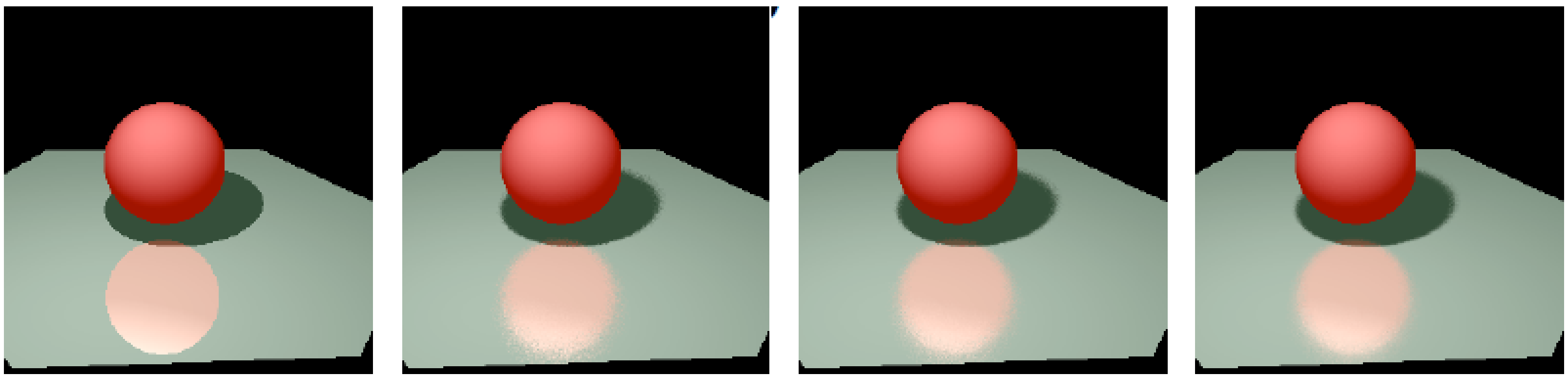
Implementation

- Integrate Over the Population of Reflection
  - Let's utilize the same function used when determining specular highlight intensity
  - Weight the each ray *R*' according to a lobe, i.e. the cosine of the angle between *R* and *R*'

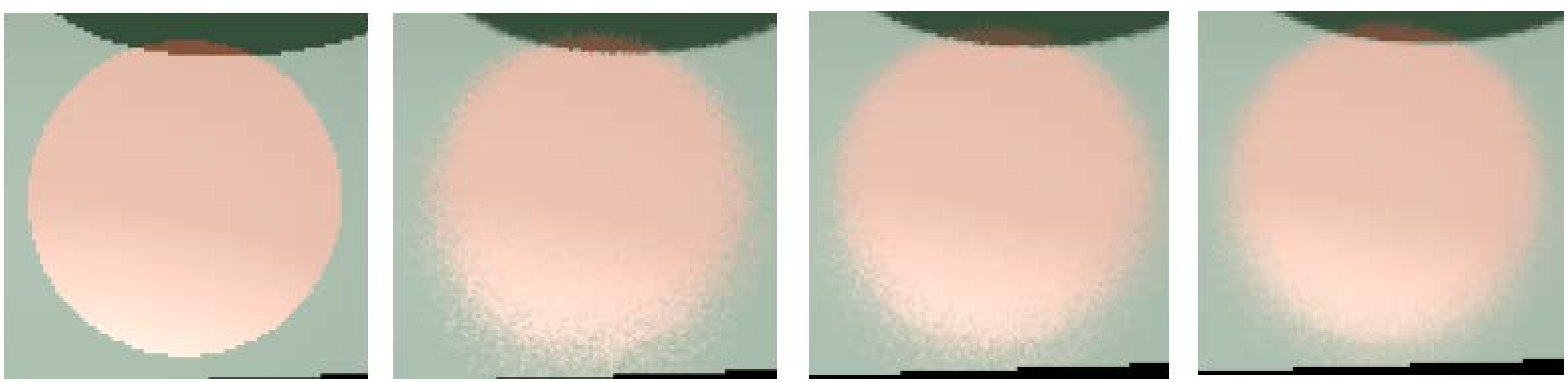


Glossy Reflection

Glossy Reflection Examples

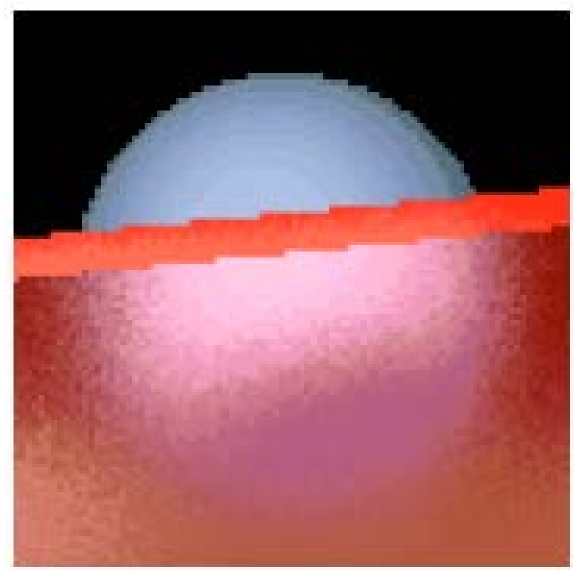


1 Ray | 10 Rays | 20 Rays | 50 Rays



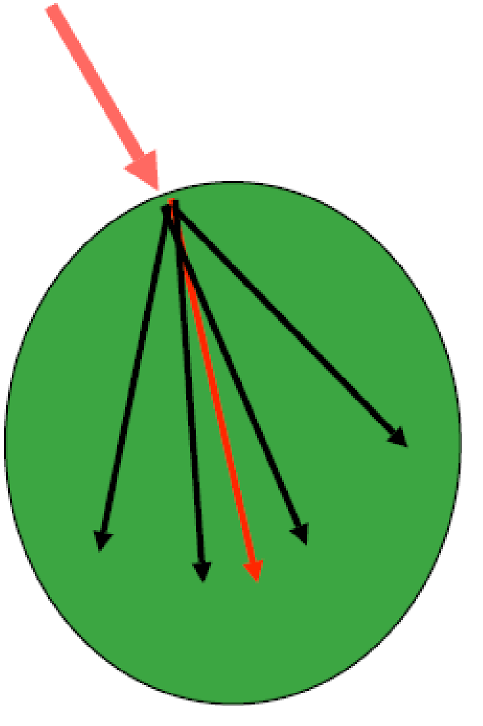
Refraction

- Assumption: Perfectly clear material, so the only refraction contribution comes from the transmittance vector
- **Realistic: "Blurry" refraction**

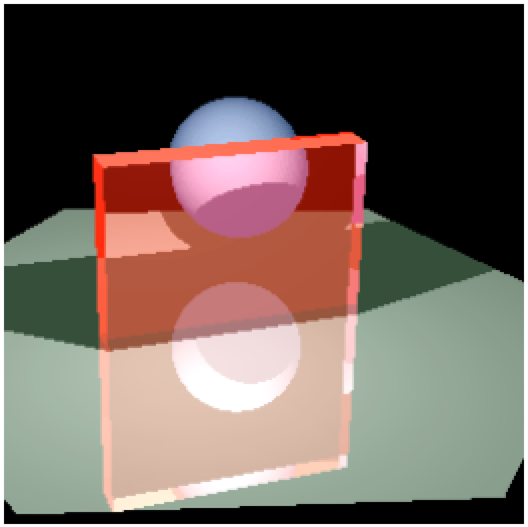


DRT for Translucency

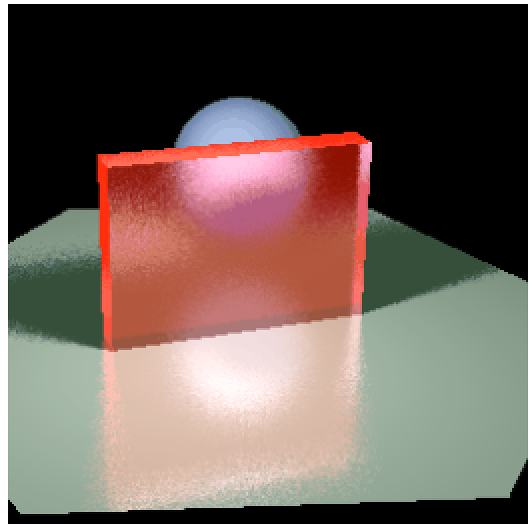
- Solution: Same solution as glossy reflection, except use the transmittance vector *T* and integrate over the hemisphere behind the surface



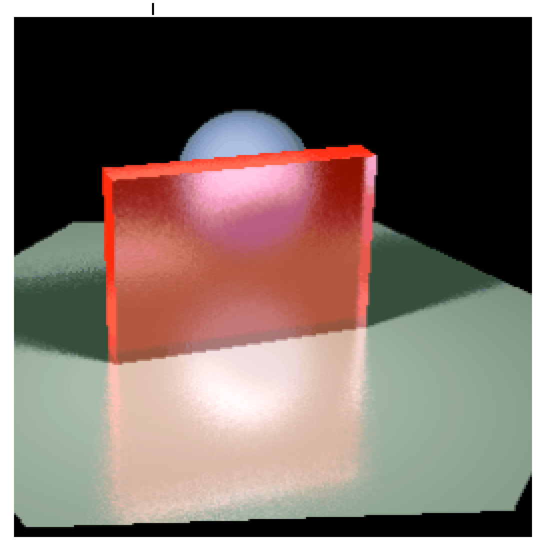
Translucency Examples



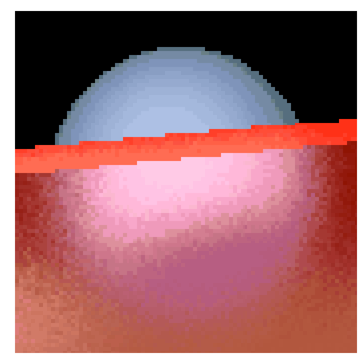
1 Ray



10 Rays

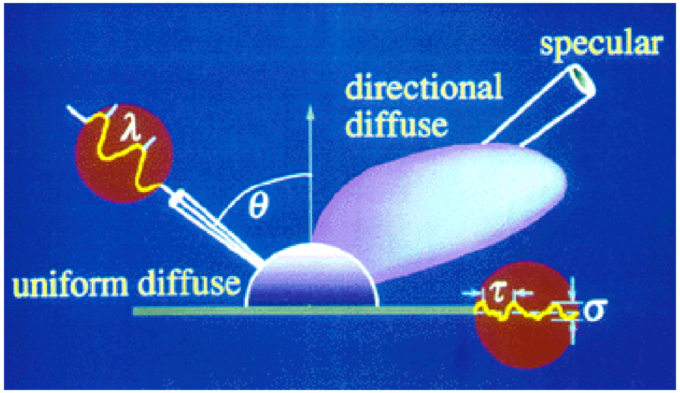


20 Rays

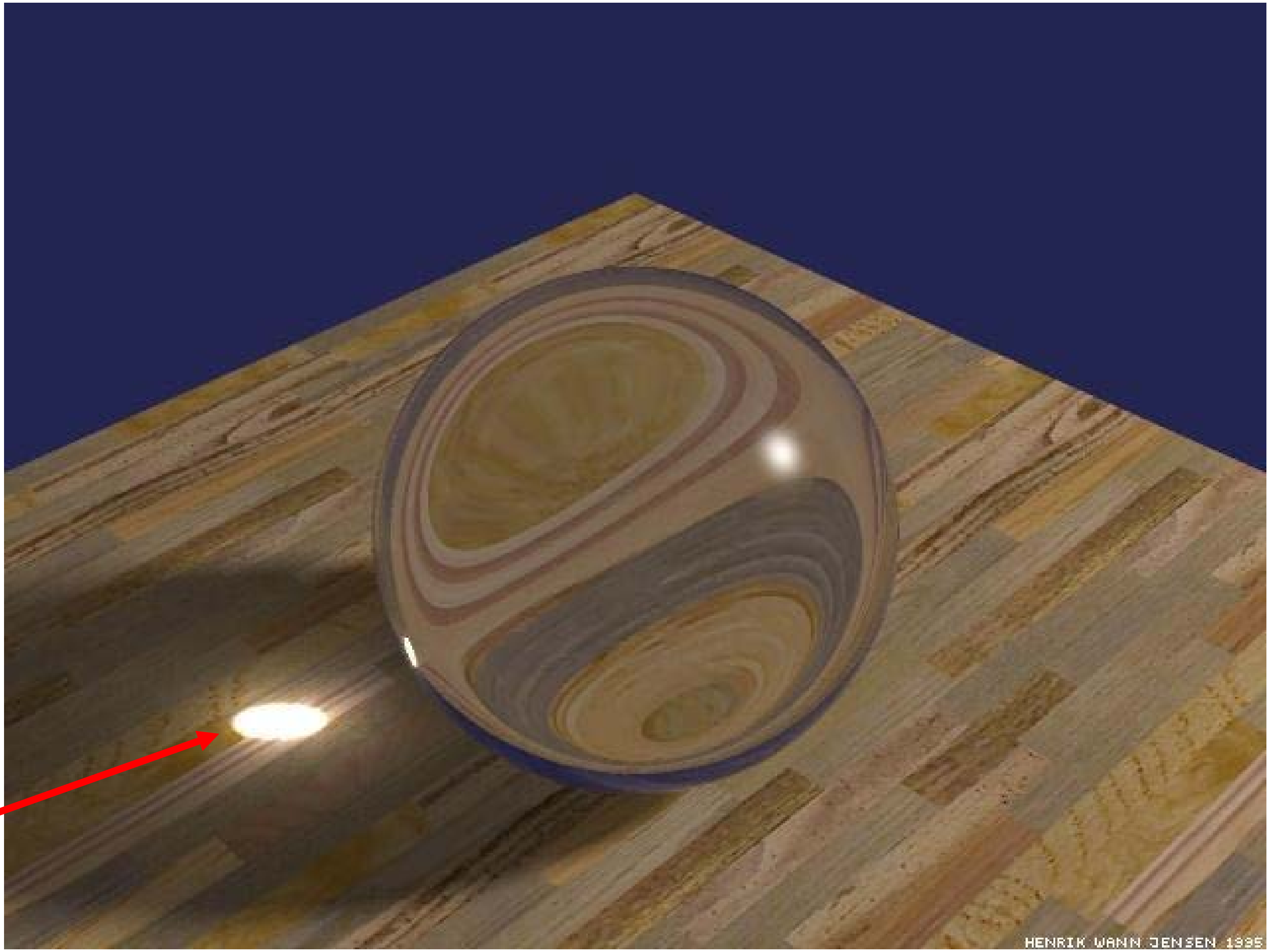


Complex Interreflection

- Model true reflection behavior as described by a full BRDF
- Randomly sample rays over the hemisphere, weight them by their BRDF value, and add them together
- Generate ray samples from a distribution that matches the BRDF for the given incident direction and average them samples together
  - This technique is called “Monte Carlo Integration”.



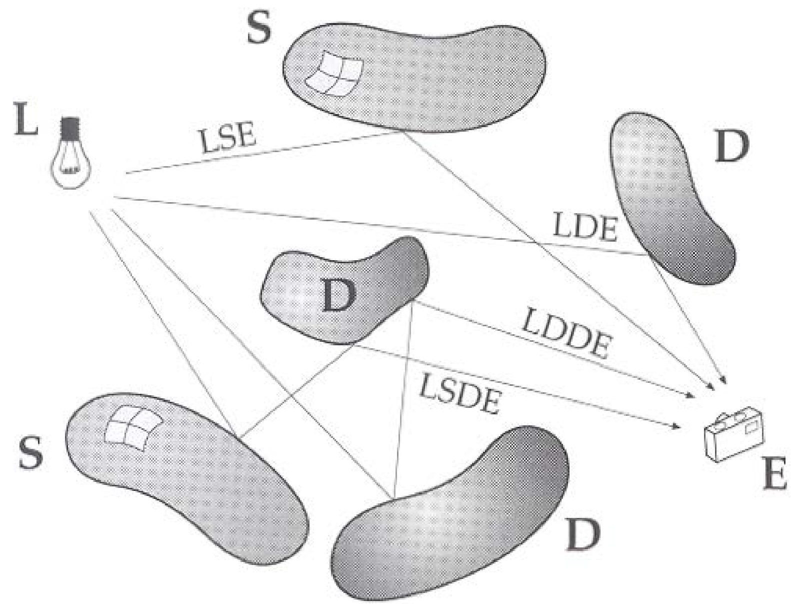
Can Ray Tracing Do This?



caustic

Light Paths

- Consider the path that a light ray might take through a scene between the light source **L** and the eye **E**
- It may interact with multiple diffuse (**D**) and specular (**S**) objects along the way
  - Includes reflections, refractions
- We can describe this series of interactions with the *regular expression* **L** (**D** | **S**) * **E**
  - (If a surface is a mix of **D** and **S**, the combination is additive so it is still OK to treat in this manner)



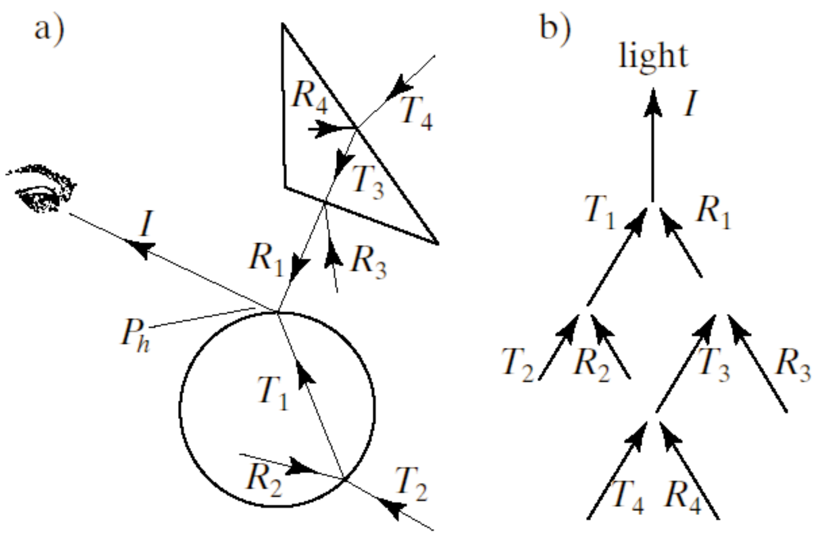
from Sillion & Puech

Light Path Notations

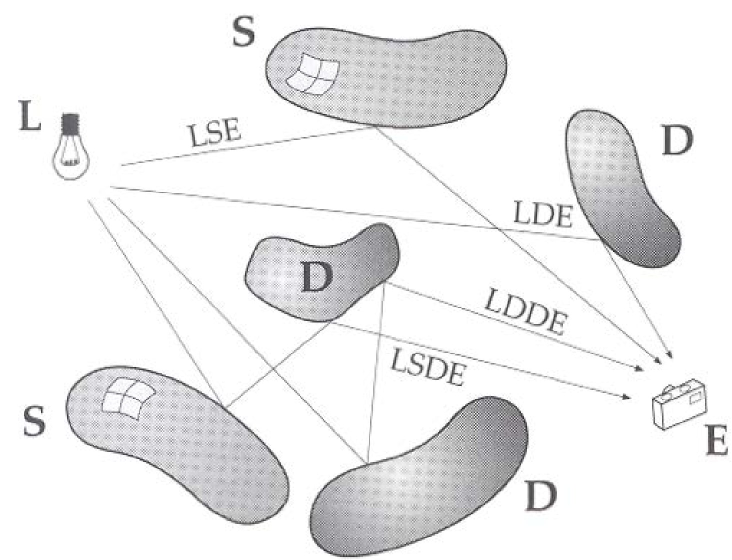
- **L**: Light source
- **E**: Eye
- **S**: Specular reflection
- **D**: Diffuse reflection
- (k)+: one or more k events
- (k)*: zero or more of k events (iterations)
- (k|k'): a k or k' event

Light Paths: Examples

- Direct visualization of the light: **LE**
- Local illumination: **LDE**, **LSE**
- Ray tracing: **LSE**, **LDS*E**



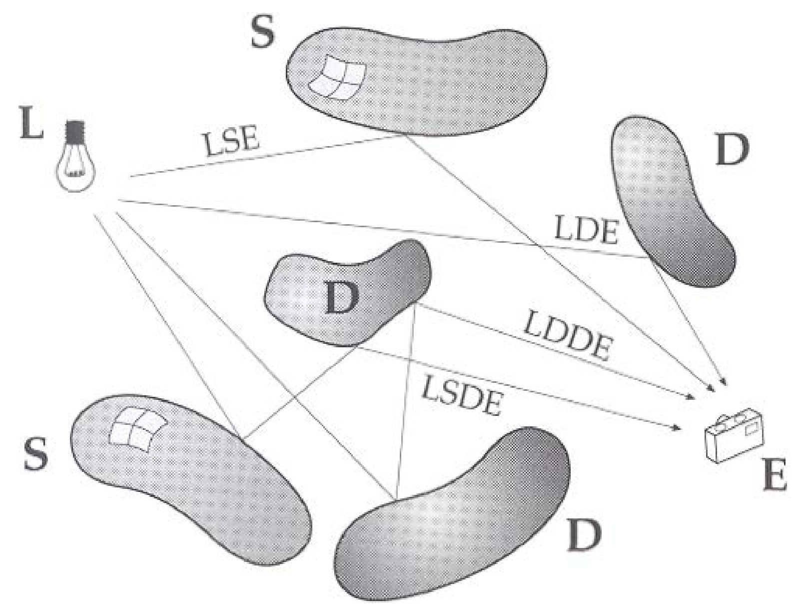
Ray tracing light paths



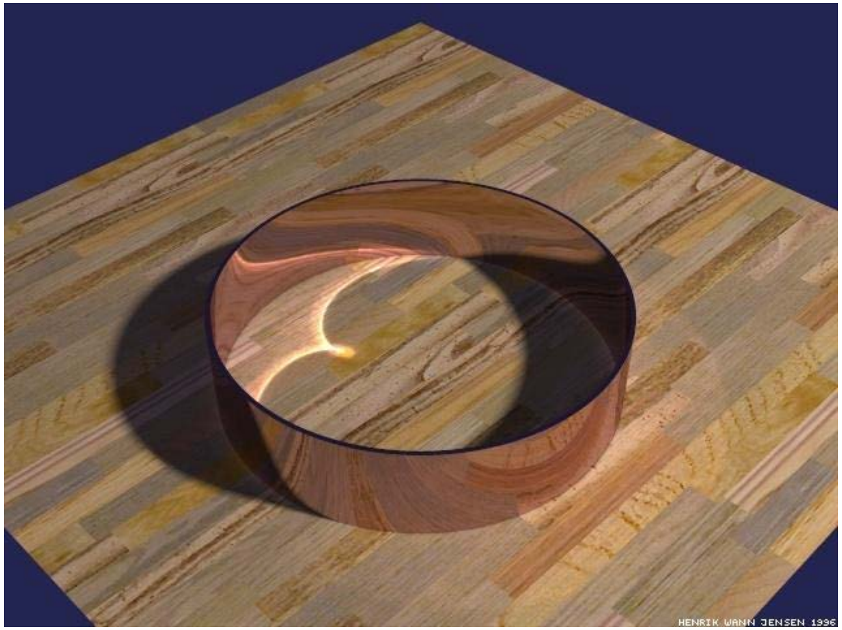
General light paths

Caustics

- Definition: (Concentrated) specular reflection/refraction onto a diffuse surface
  - In simplest form, follow an **LS+DE** path
- Standard ray tracing **cannot** handle caustics—only paths described by **LDS*E**

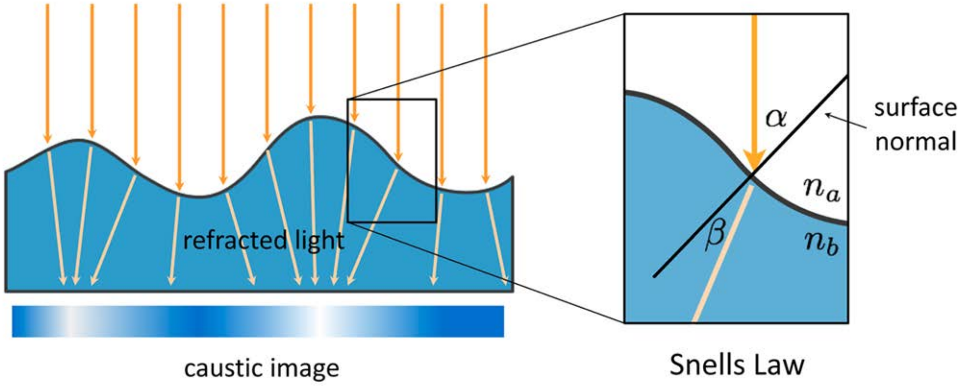


from Sillion & Puech



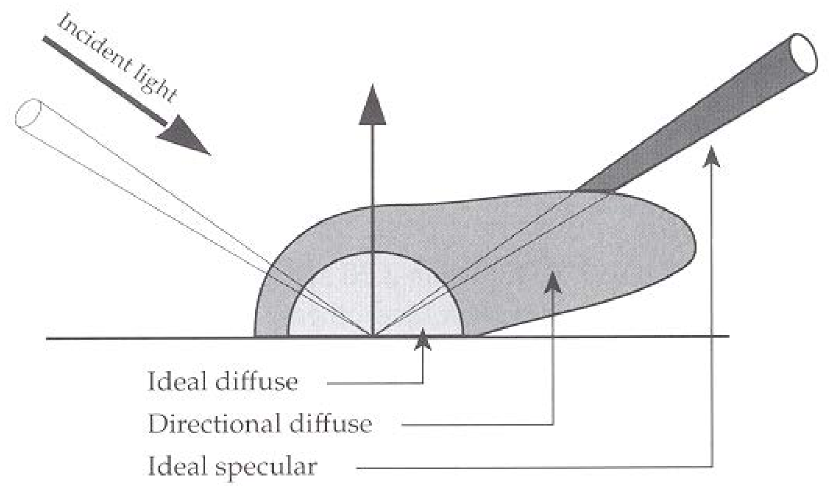
courtesy of H. Wann Jensen

Caustics: Examples

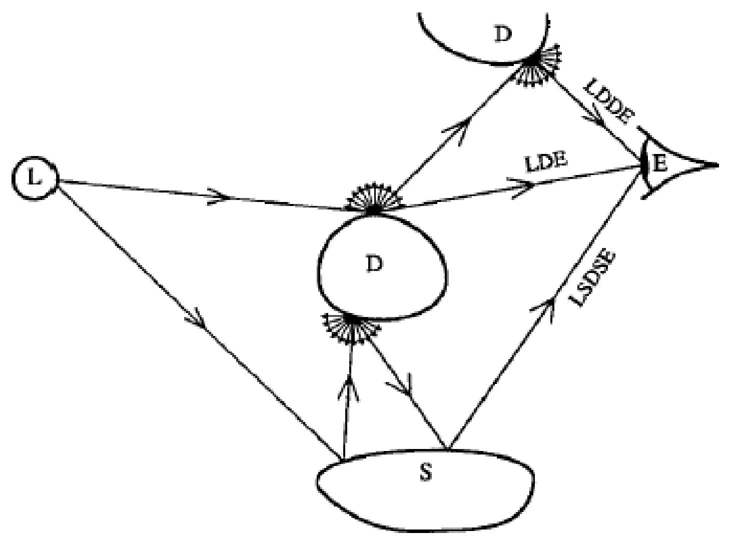


The Problem with Diffuse Surfaces

- For specular surfaces, we “know” where the photon will go (= “came from”, if going backwards), whereas for diffuse surfaces there’s much more uncertainty
  - If we’re tracing a ray from the eye and we hit a diffuse surface, this uncertainty means that the source of the photon could be anywhere in the hemisphere
  - Conventional ray tracing just looks for lights at this point, but for **LS$^+$DE** paths we need to look for other specular surfaces
    - How to find them?



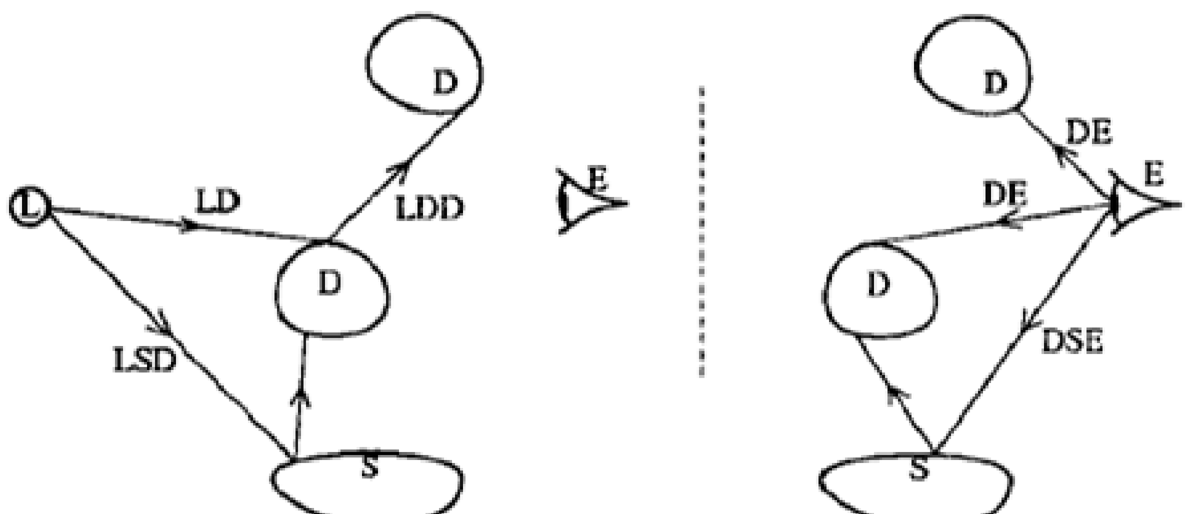
from Sillion & Puech



from P. Heckbert

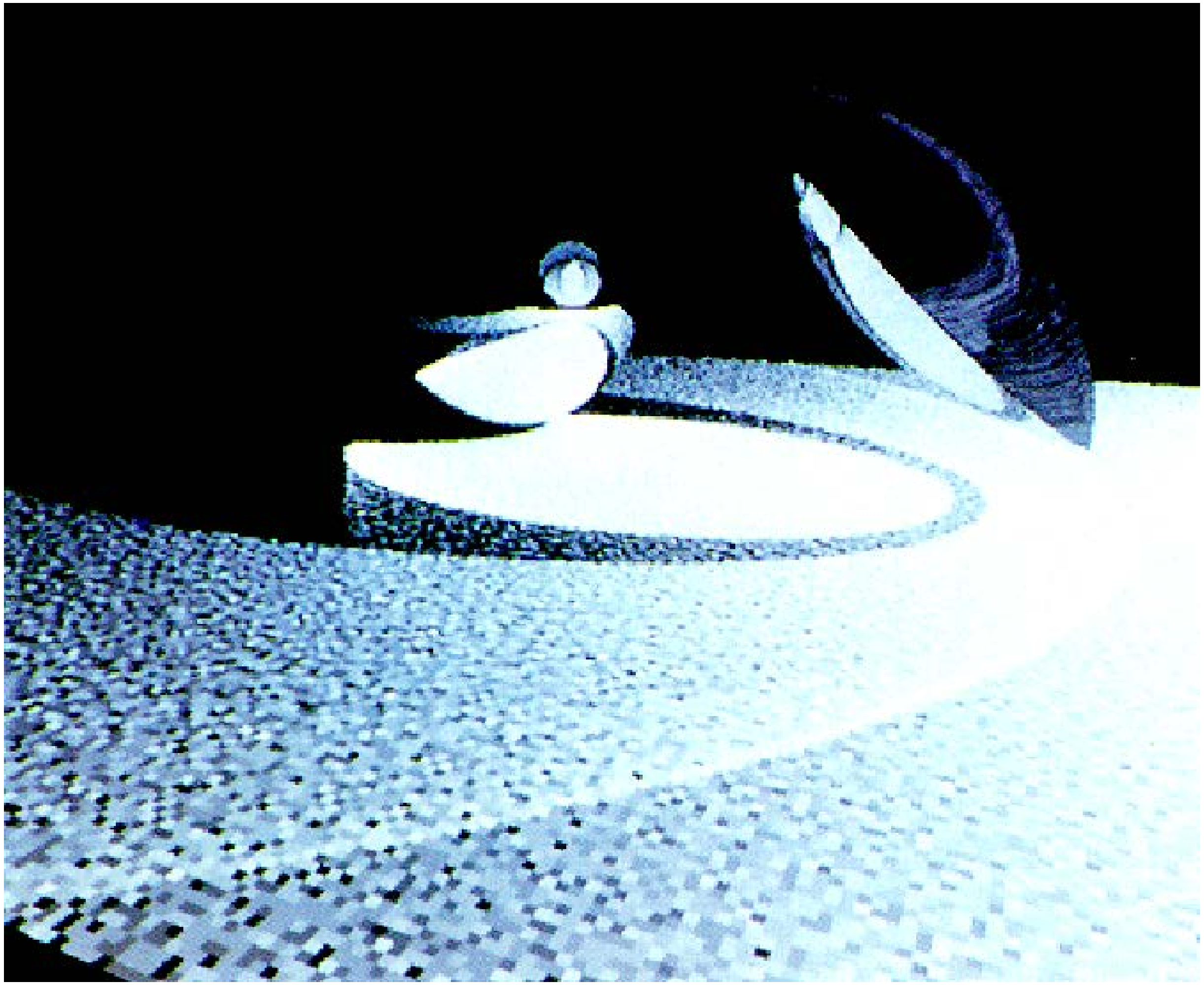
Bidirectional Ray Tracing (P. Heckbert, 1990)

- Idea: Trace forward **light rays** into scene as well as backward **eye rays**
- At diffuse surfaces, light rays additively "deposit" photons in **radiosity textures**, or "rexes", where they are accessed by eye rays
  - Summation approximates integral term in radiance computation
  - Light rays carry information on specular surface locations—they have no uncertainty



from P. Heckbert

Bidirectional Ray Tracing: Results



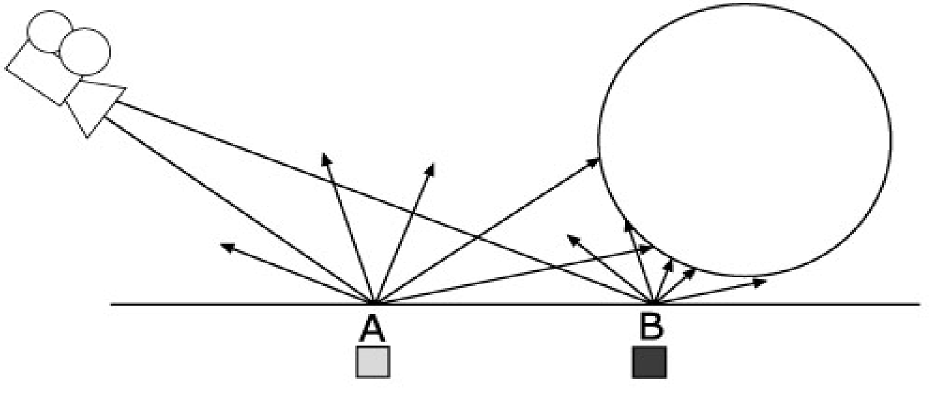
Lens, mirrored sphere, and diffuse surface with caustic of focused light

Ambient Occlusion (AO)

- **A**mbient **O**cclusion (**AO**)
  - Shadowing of ambient light
  - Darkening of the ambient shading contribution

How to realize AO?

- Main idea: Cast multiple random rays (a “distribution of rays”) from each rendered surface point to estimate percent of sky hemisphere that is visible
  - Limit length of rays so distant objects have no effect
  - Cosine weighting/distribution for foreshortening
- Developers of this idea won a technical Oscar in 2010



Object Space Ambient Occlusion

- AO does not depend on light direction
- Precompute AO for static objects using ray casting
  - How many rays?
  - How far do they go?
  - Local objects? Or all objects?

Object Space Ambient Occlusion

- The integral of the occlusion contributed from inside a hemisphere of a given radius **R**, centered at the current surface point **P** and oriented towards the normal **n** at **P**

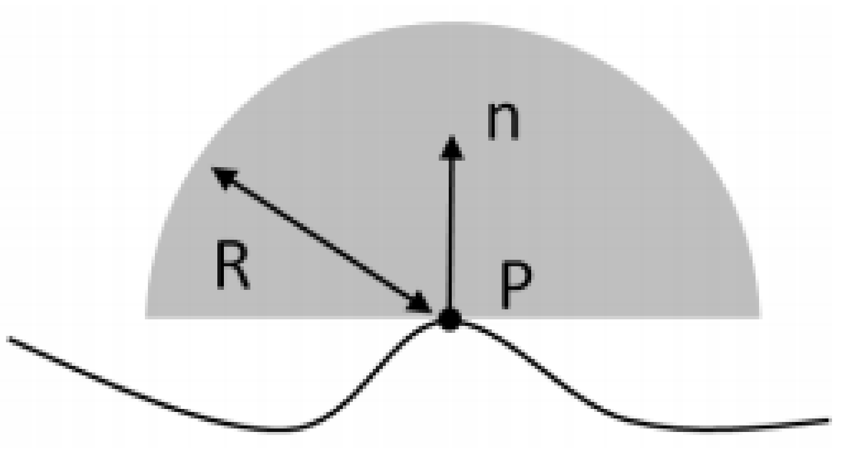
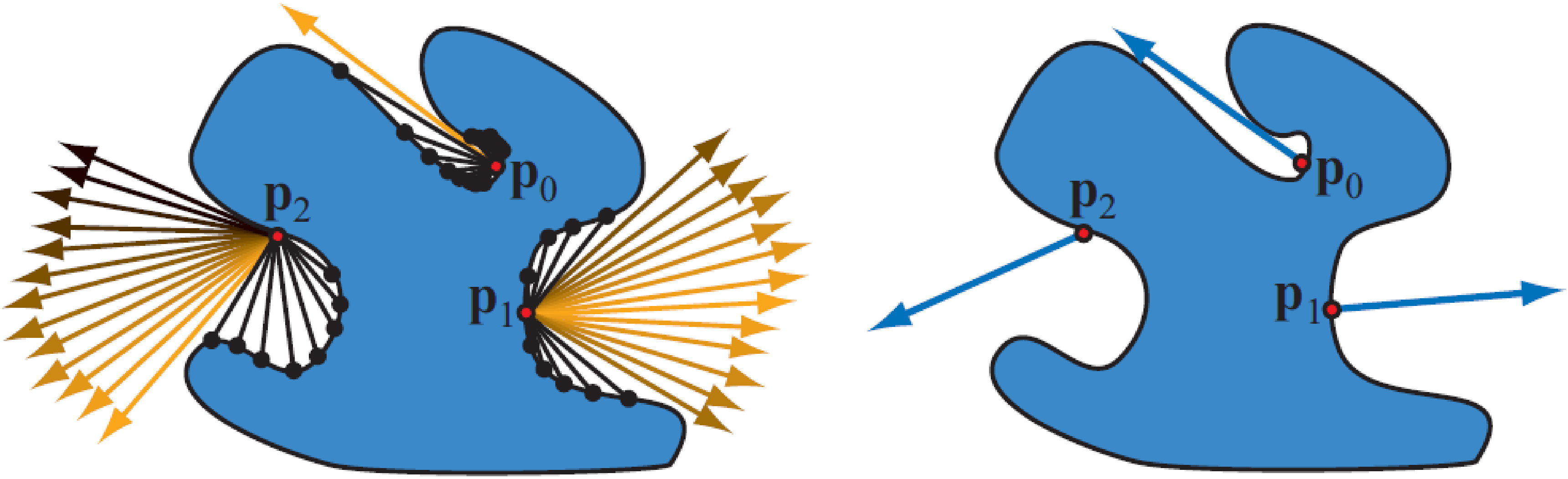


Figure 2. Hemisphere Ω around a surface point P.

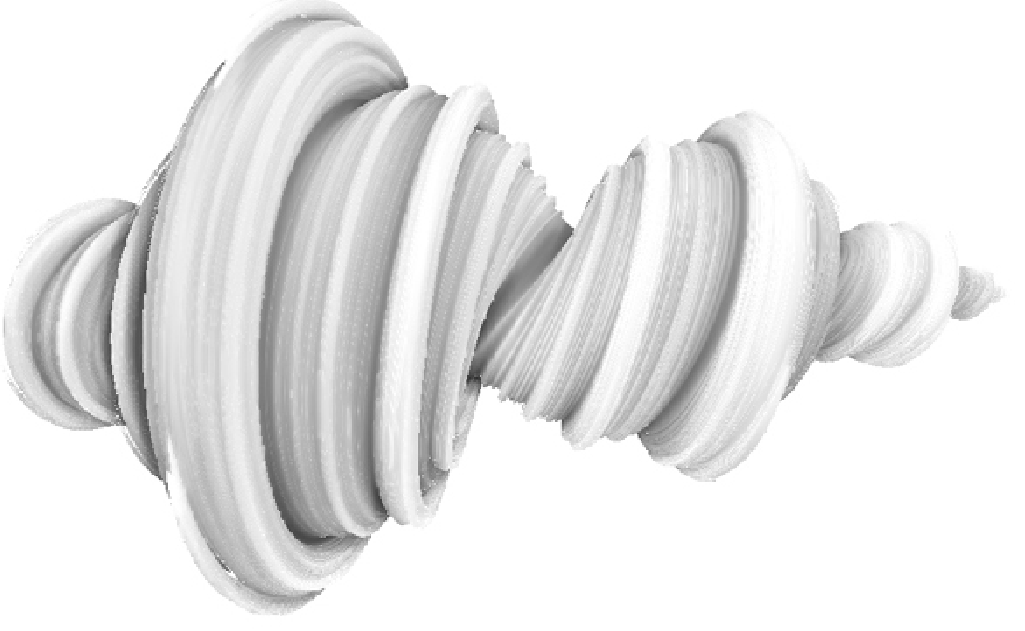
Object Space Ambient Occlusion



- Cosine weight rays
  - or use *importance sampling*: cosine distribute number of rays

Object Space AO: Notes

- Depends on scene complexity
- AO values are stored in textures or vertices



Next Time …

- No class on Thursday (Thanksgiving)

- Next meeting will be on Tuesday (11/28)
  - – Image-based rendering