Photon Mapping

LUGG
Lund University Graphics Group

Michael Doggett
Department of Computer Science
Lund university

Outline

- Photon Mapping (ch. 14 in textbook)
  - Progressive
  - Stochastic

How to make light sampling faster?

- Path tracing converges slowly
- Shot rays from the light
  - Bidirectional path tracing
  - Metropolis Light Transport
    - Find a path to the light
    - ... then reuse a modified version
  - Photon Mapping

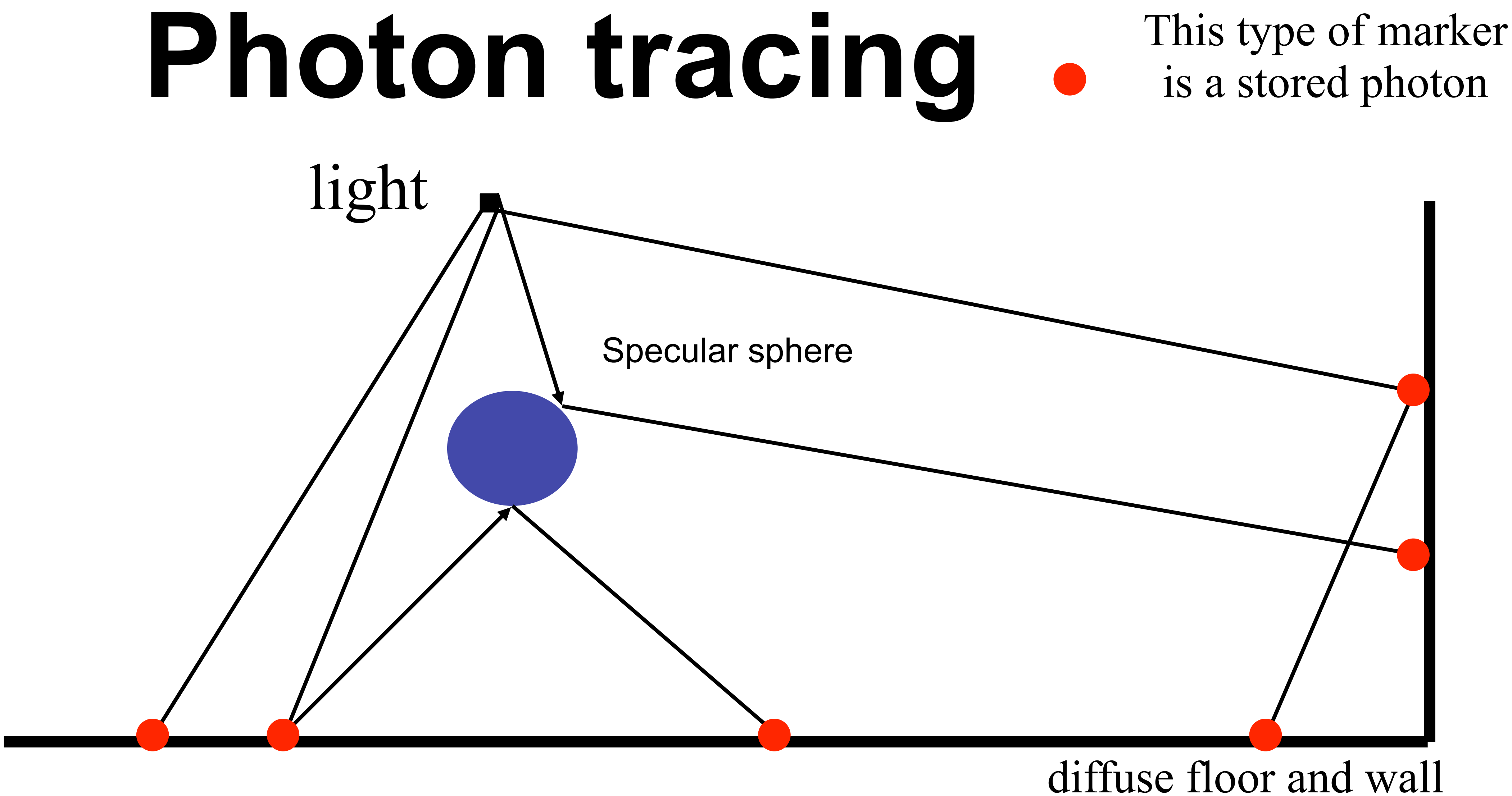
Photon mapping
State-of-the-art in GI

- Developed by Henrik Wann Jensen (started 1993)
- A clever two-pass algorithm:
  - 1: Shoot photons from light sources, and let them bounce around in the scene, and store them where they land
  - 2: ”Ray tracing”-like pass from the eye, but gather the photons from the previous pass
- Advantages:
  - Fast
  - Handles arbitrary geometry (as does path tracing)
  - All global illumination effects can be seen
  - Little noise

The first pass: Photon tracing

- Store illumination as points (photons) in a ”photon map” data structure
- In the first pass: **photon tracing**
  - Emit photons from light sources
  - Trace them through scene
  - Store them in photon map data structure
- More details:
  - Then use Russian roulette to find out whether the photon is absorbed or reflected
  - If a photon is absorbed, by a surface (that has a diffuse component), store the photon in photon map
  - If reflected, then shoot photon in new random direction

Photon tracing



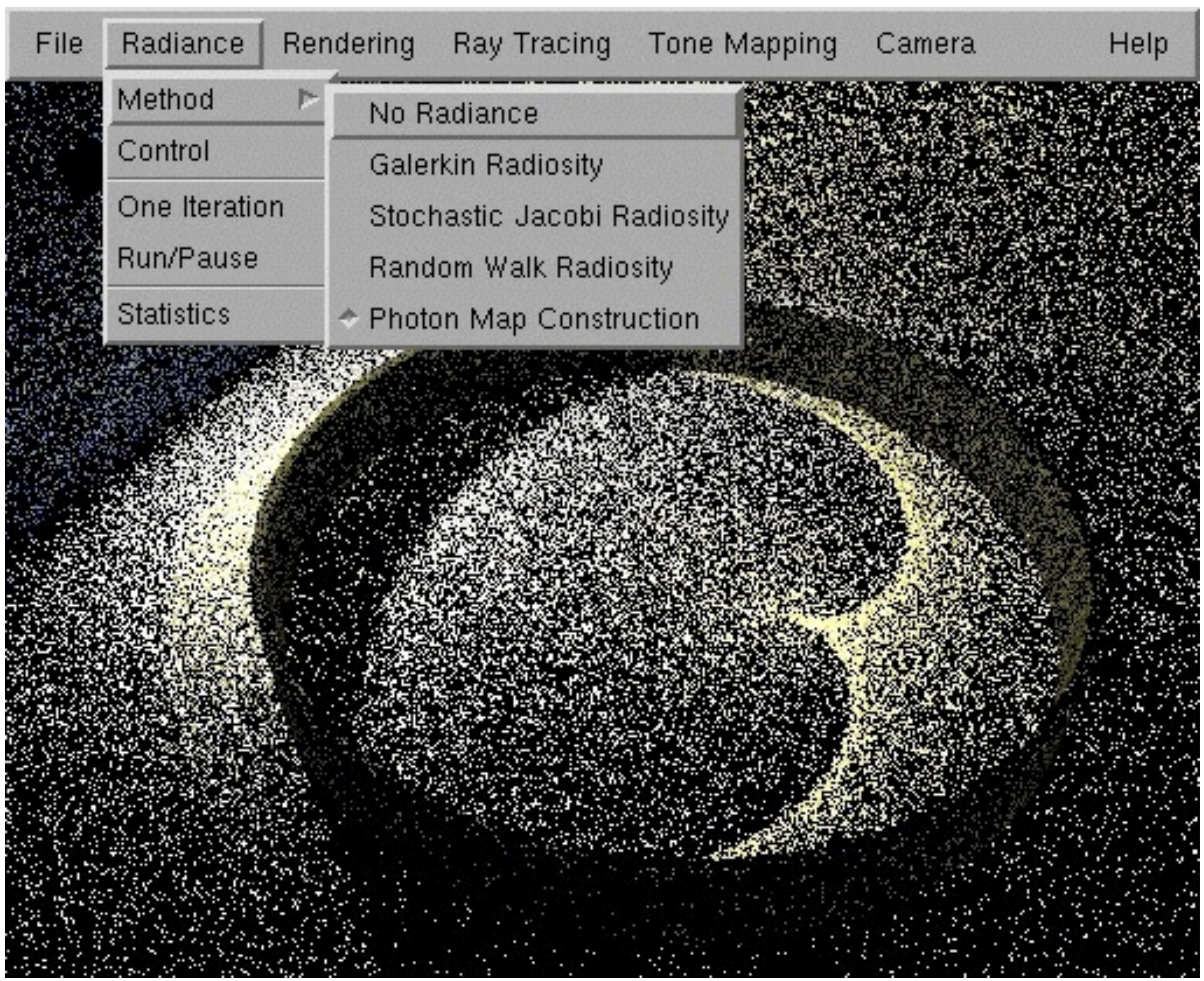
- Should not store photon at specular surfaces, because these effects are view dependent
  - –only diffuse effect is view independent
- Some diffuse photons are absorbed, some are scattered further
- A photon = the incoming illumination at a point

The photon map data structure

- Keep them in a separate (from geometry) structure
- Store all photons in kD-tree
  - Essentially an axis-aligned BSP tree, but we must alter splitting axis: x,y,z,x,y,z,x,y,z, etc.
  - Each node stores a photon
  - Needed because the algorithm needs to locate the n closest photons to a point
- A photon:
  - float x,y,z;
  - char power[4]; // RGBE, with more accuracy using shared Exponent
  - char phi,theta; // compact representation of incoming direction
  - short flag; // used by KD-tree (stores which plane to split)
- Create balanced KD-tree – simple, done once.
- Photons are stored linearly in memory:
  - Parent node at index: p, left child at: 2p , right child: 2p+1

What does it look like?

- Stored photons displayed:



Density estimation

- The density of the photons indicate how much light that point receives
- Radiance is the term for what we display at a pixel
- Rather complex derivation skipped (see Jensen's book)…
- Reflected radiance at point x:

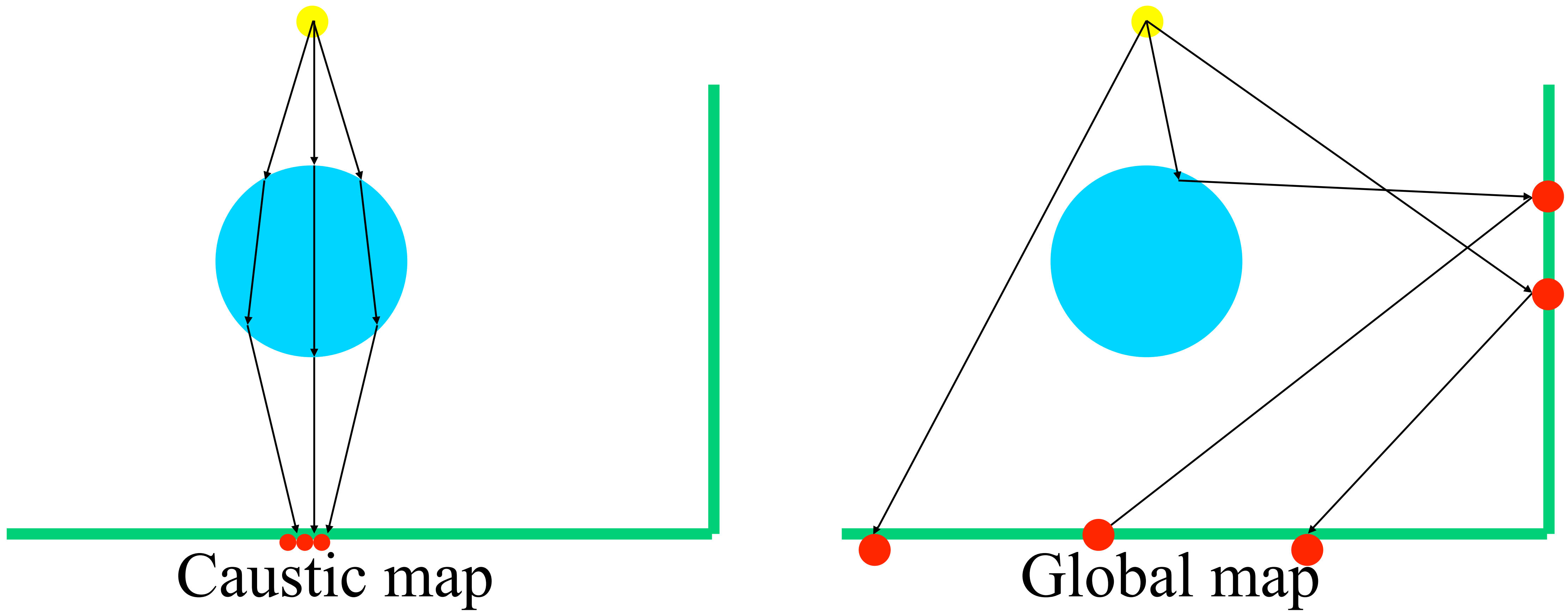
$$L(\mathbf{x},\omega) \approx \frac{1}{\pi r^2} \sum_{1}^{n} f_r(\mathbf{x},\omega_p \leftrightarrow \omega)\Phi_p(\mathbf{x},\omega_p)$$

- L is radiance at $\mathbf{x}$ in the direction of ω
- r is radius of expanded sphere
- ω_p is the direction of the stored photon
- Φ_p is the stored power of the photon (flux)
- f_r is the BRDF

Two-pass algorithm

- Already said:
  - –1) Photon tracing, to build photon maps
  - –2) Rendering from the eye using photon maps
- Pass 1:
  - –Use **two** photon maps
  - –A **caustics photon map** (for caustics)
    - Reflected or refracted via a surface to a diffuse surface
    - Light transport notation: LS+D
  - –A **global photon map** (for all illumination)
    - All photons that landed on diffuse surfaces
    - L(S | D)*D

Caustic map and global map



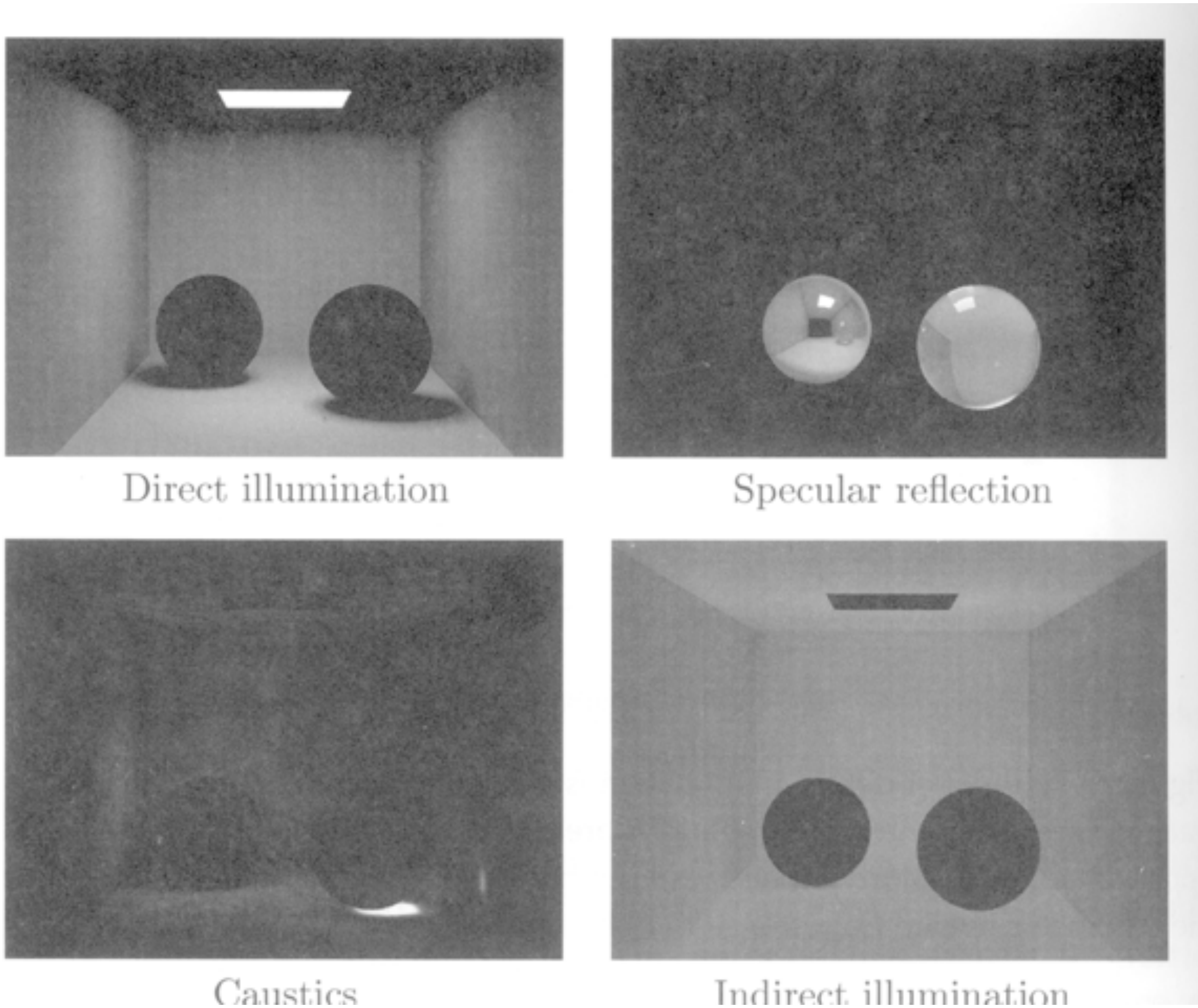
- Caustic map: send photons only towards reflective and refractive surfaces
  - –Caustics is a high frequency component of illumination
  - –Therefore, need many photons to represent accurately
- Global map - assumption: illumination varies more slowly

Pass 2: Rendering using the photon map

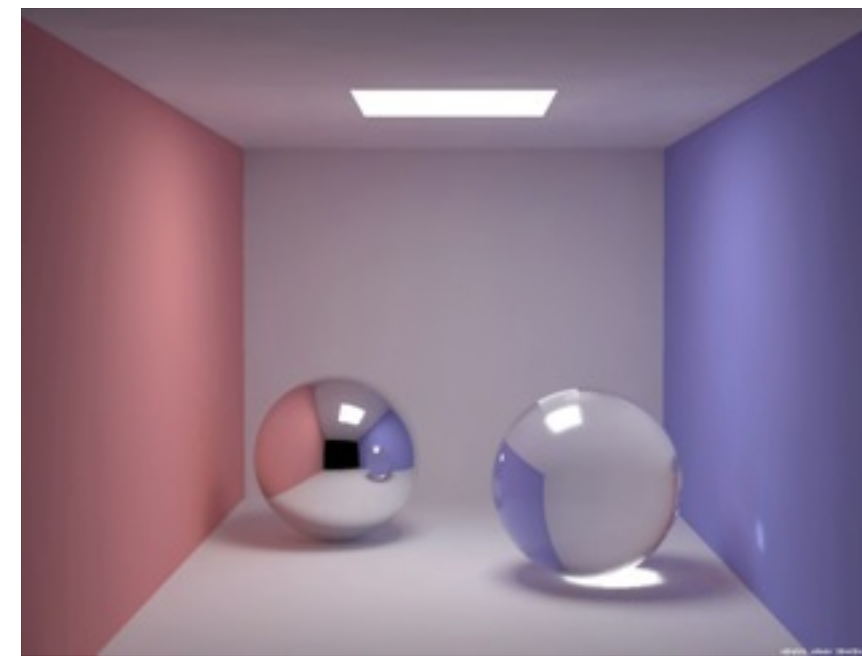
- Render from the eye using a modified ray tracer
  - –A number of rays are sent per pixel
  - –Break up into fastest way to compute different terms
  - –For each ray evaluate four terms
    - **Direct illumination** (light reaches a surface directly from light source)… may need to send many rays to area lights. Done using standard ray tracing.
    - **Specular reflection** (also evaluated using ray tracing, possibly, with many rays sent around the reflection direction)
    - **Caustics**: use caustics photon map
    - **Indirect illumination** (sample multiple diffuse reflections) (gives color bleeding): use the photon map for reflected rays.

Images of the four components

Image scanned from Henrik Wann Jensen's book

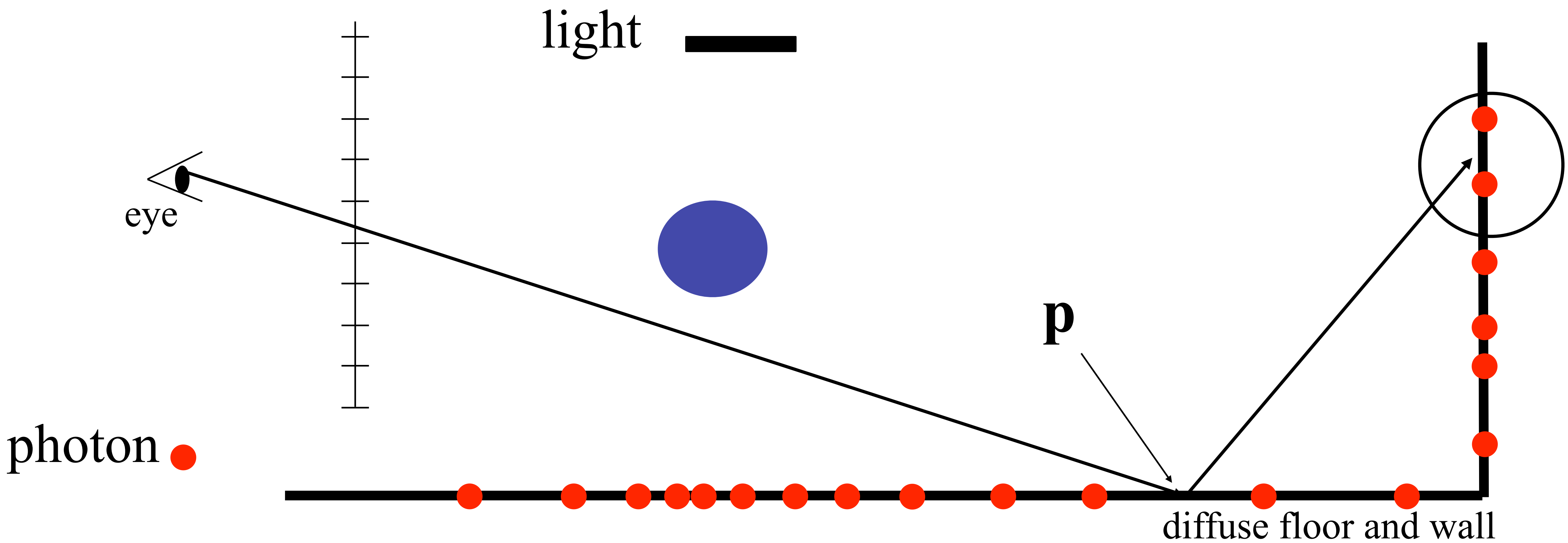


=



- These together solves the entire rendering equation!

Indirect illumination: Use the photon map



- To evaluate indirect illumination at point **p**:
  - Send several random rays out from **p**, and grow spheres at contacts
  - May need several hundreds of rays to get good results.

Locate *n* nearest photons

given the photon map, a position x and a max search distance d^2
this recursive function returns a heap h with the nearest photons.
Call with `locate_photons(1)` *to initiate search at the root of the kd-tree*

```
locate_photons( p ) {
  if ( 2p+1 < number of photons ) {
    examine child nodes
                Compute distance to plane (just a subtract)
    δ = signed distance to splitting plane of node n
    if (δ < 0) {
                  We are left of the plane - search left subtree first
      locate_photons( 2p )
      if ( δ² < d² )
        locate_photons( 2p+1 )        check right subtree
    } else {
                  We are right of the plane - search right subtree first
      locate_photons( 2p+1 )
      if ( δ² < d² )
        locate_photons( 2p )        check left subtree
    }
  }
              Compute true squared distance to photon
  δ² = squared distance from photon p to x
  if ( δ² < d² ) {              Check if the photon is close enough?
    insert photon into max heap h
                Adjust maximum distance to prune the search
    d² = squared distance to photon in root node of h
  }
}
```

from Fig4.10, photon mapping tutorial, "Advanced Global Illumination Using Photon Mapping", SIGGRAPH 2008

More literature

- Photon Mapping Tutorial, “Advanced Global Illumination Using Photon Mapping”, SIGGRAPH 2008 course notes
- Great book on global illumination and photon mapping:
  - Henrik Wann Jensen, *Realistic Image Synthesis using Photon Mapping*, AK Peters, 2001.
- Check Henrik’s website:
  - http://graphics.ucsd.edu/~henrik/
  - "Global Illumination using Photon Maps"
    Henrik Wann Jensen
    In *"Rendering Techniques '96"*. Eds. X. Pueyo and P. Schröder. Springer-Verlag, pages 21-30, 1996

Progressive Photon Mapping

LAB4

- Photon mapping limited by Photon Map
- Rearrange algorithm to remove limitation
- Progressively refine final image by tracing more photons
  - image approaches correct solution

"Progressive Photon Mapping", T. Hachisuka, S. Ogaki and H. W. Jensen, ACM Transactions on Graphics (SIGGRAPH Asia 2008)

Progressive Photon Mapping

LAB4

- 1st pass - find eye intersection positions
- 2nd pass - collect photons at intersection positions

View hit points

Photons

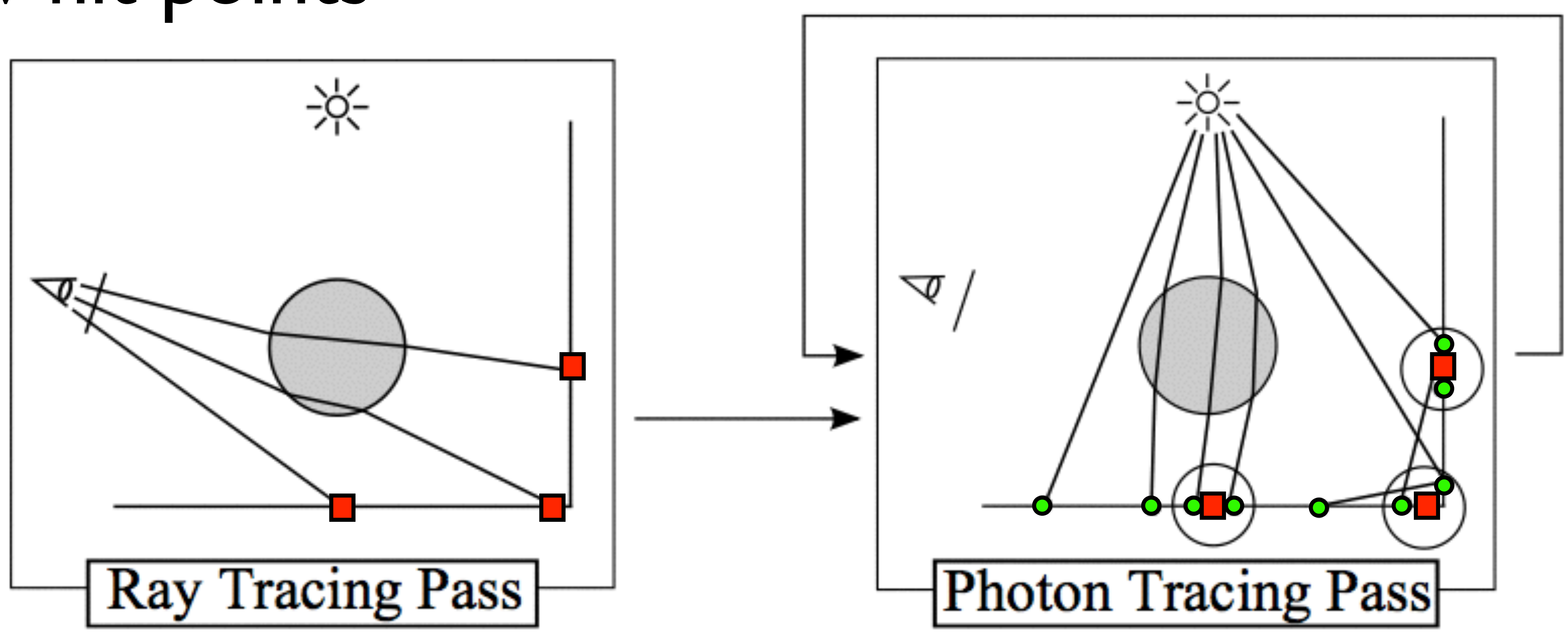
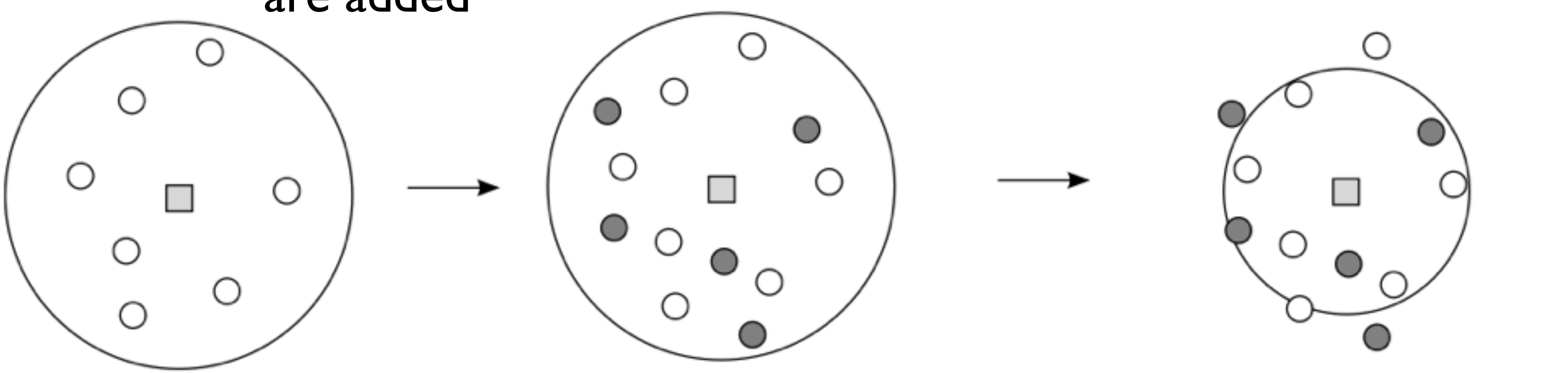


image from "Progressive Photon Mapping", T. Hachisuka, S. Ogaki and H.W. Jensen, ACM TOG 2008

Progressive Photon Mapping

LAB4

- 1st pass - find eye intersection positions
- 2nd pass - collect photons at intersection positions
  - Reduce radius of gathered photons as more photons are added



Radius: R(x) Photons: N(x) → Radius: R(x) Photons: N(x) + M(x) → Radius: R(x) - dR(x) Photons: N(x) + αM(x)

image from "Progressive Photon Mapping", T. Hachisuka, S. Ogaki and H. W. Jensen, ACM TOG 2008

Progressive Photon Mapping

LAB4

- Equal time image : 22 hours
  - PM : 20 million photons, shorter runtime due to memory limit
  - PPM :165 million photons

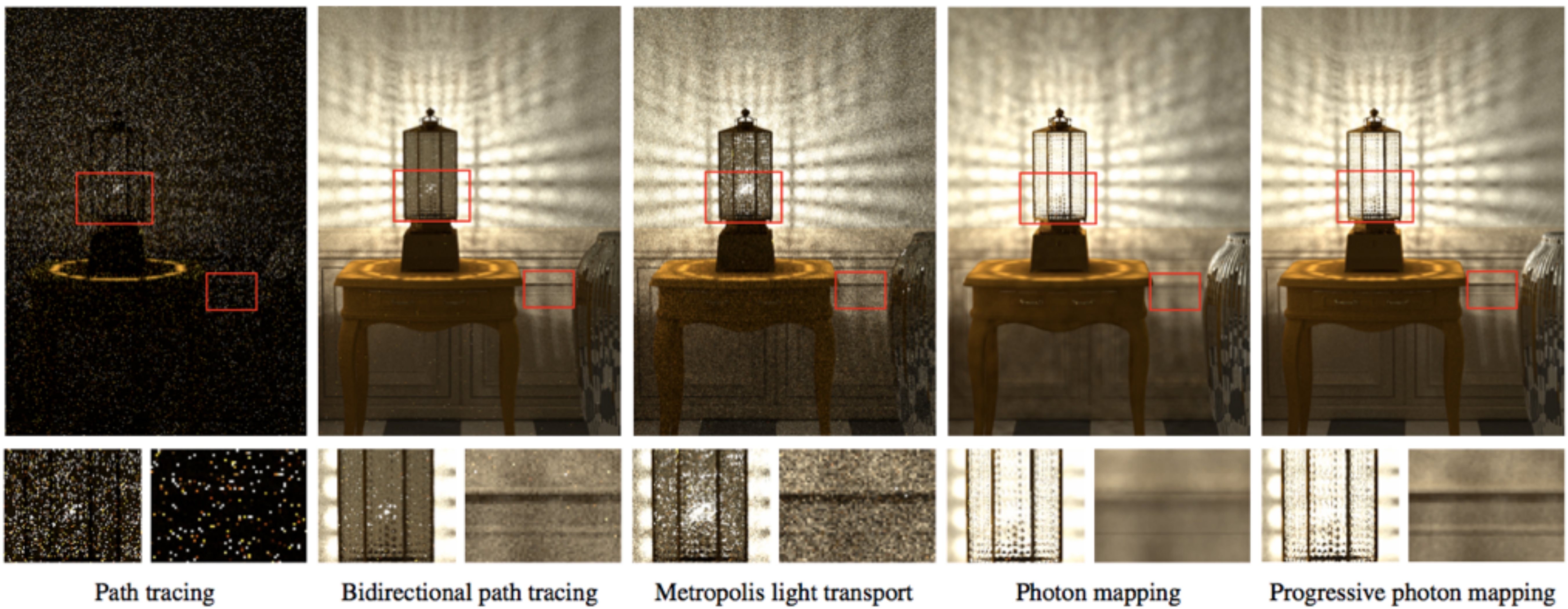


image from "Progressive Photon Mapping", T. Hachisuka, S. Ogaki and H.W. Jensen, ACM TOG 2008

Stochastic Progressive Photon Mapping

- Add extra pass
  - Randomly generate hit points

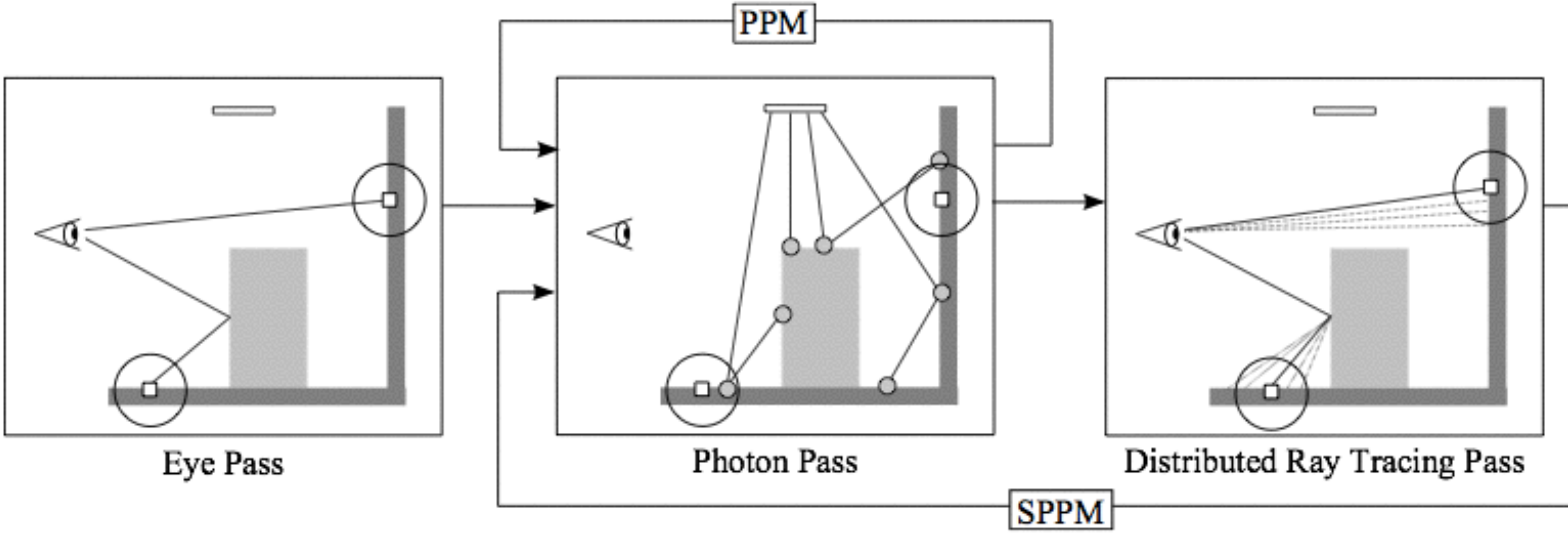


image from "Stochastic Progressive Photon Mapping", T. Hachisuka, and H. W. Jensen, ACM TOG 2009

Stochastic Progressive Photon Mapping

- BDPT (left), PPM (middle), and SPPM (right)
- Same rendering time with Depth-of-field and caustics

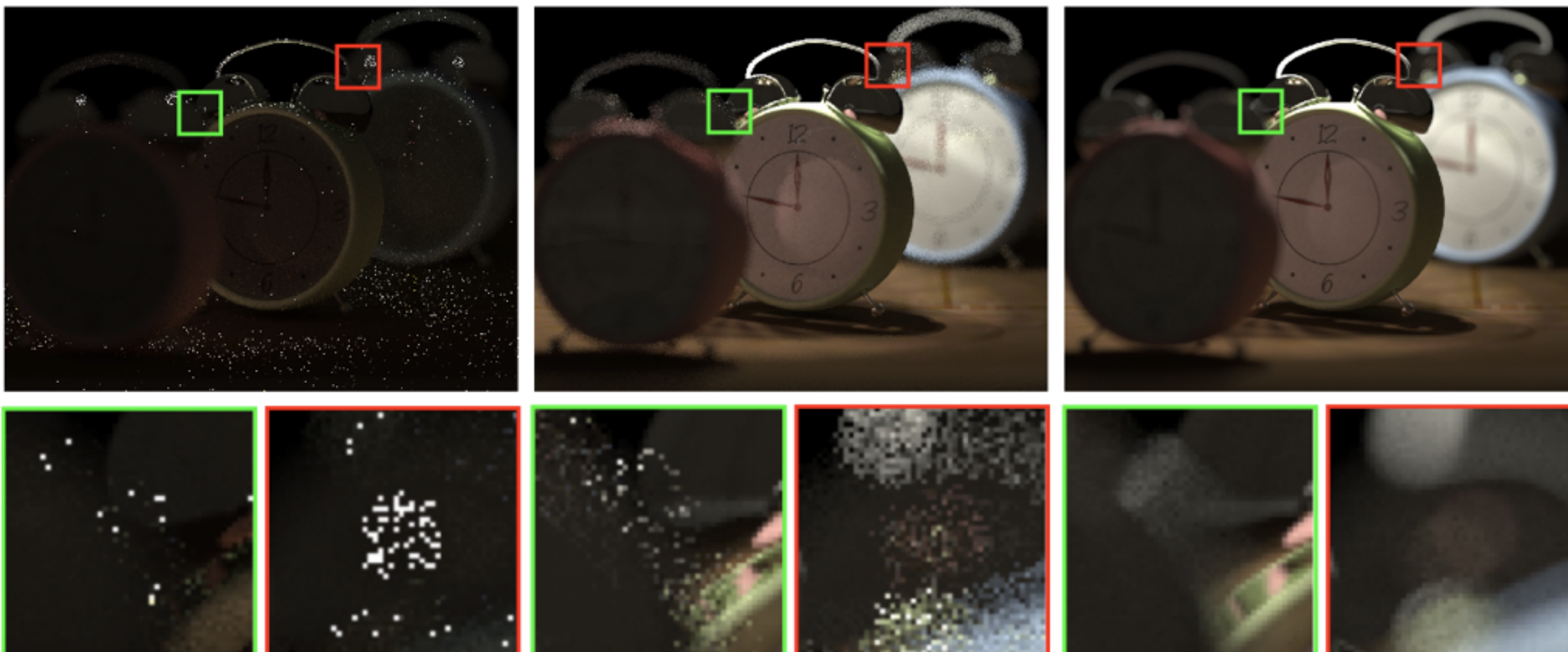


image from "Stochastic Progressive Photon Mapping", T. Hachisuka, and H.W. Jensen, ACM TOG 2009

Stochastic Progressive Photon Mapping

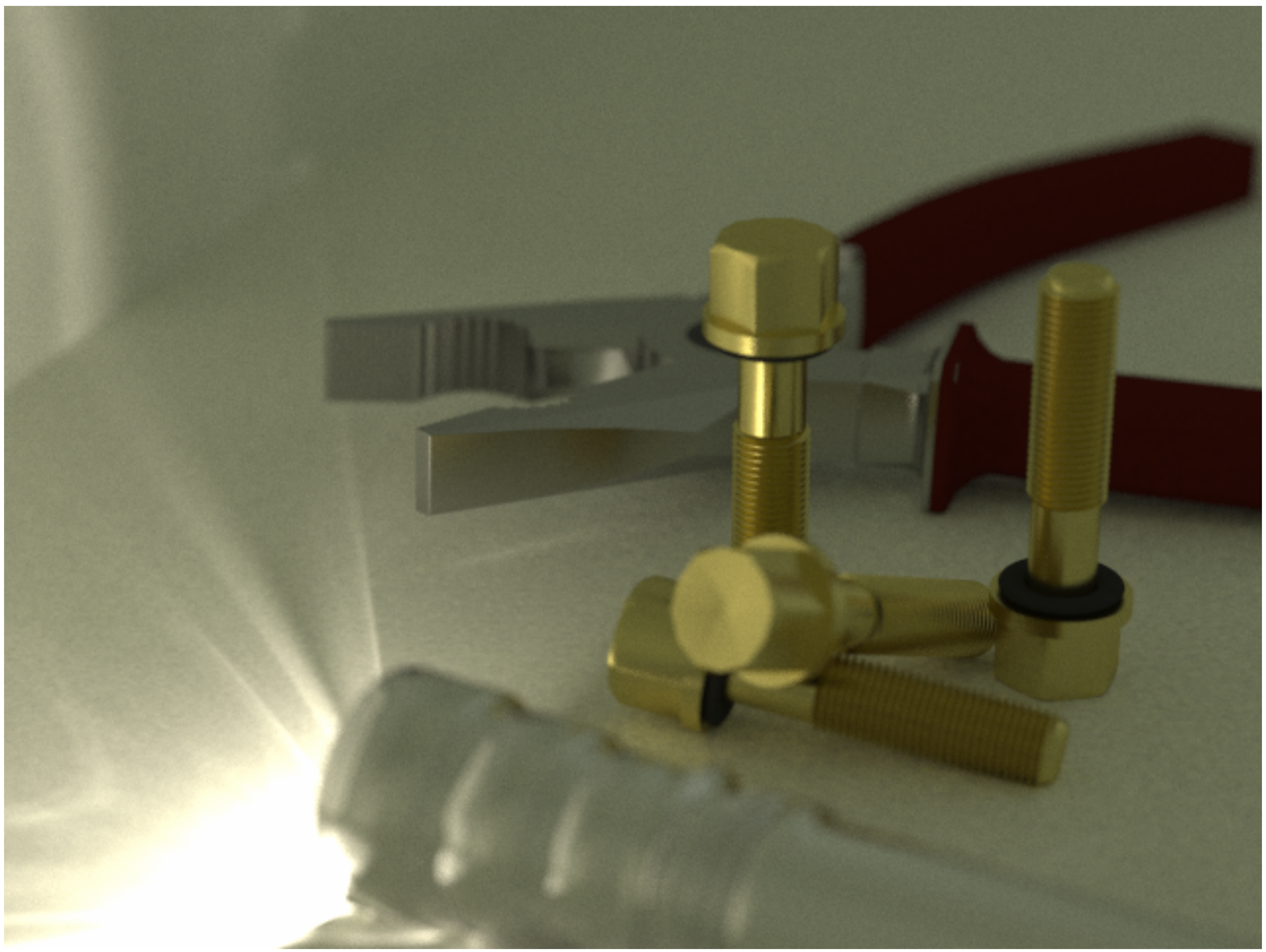
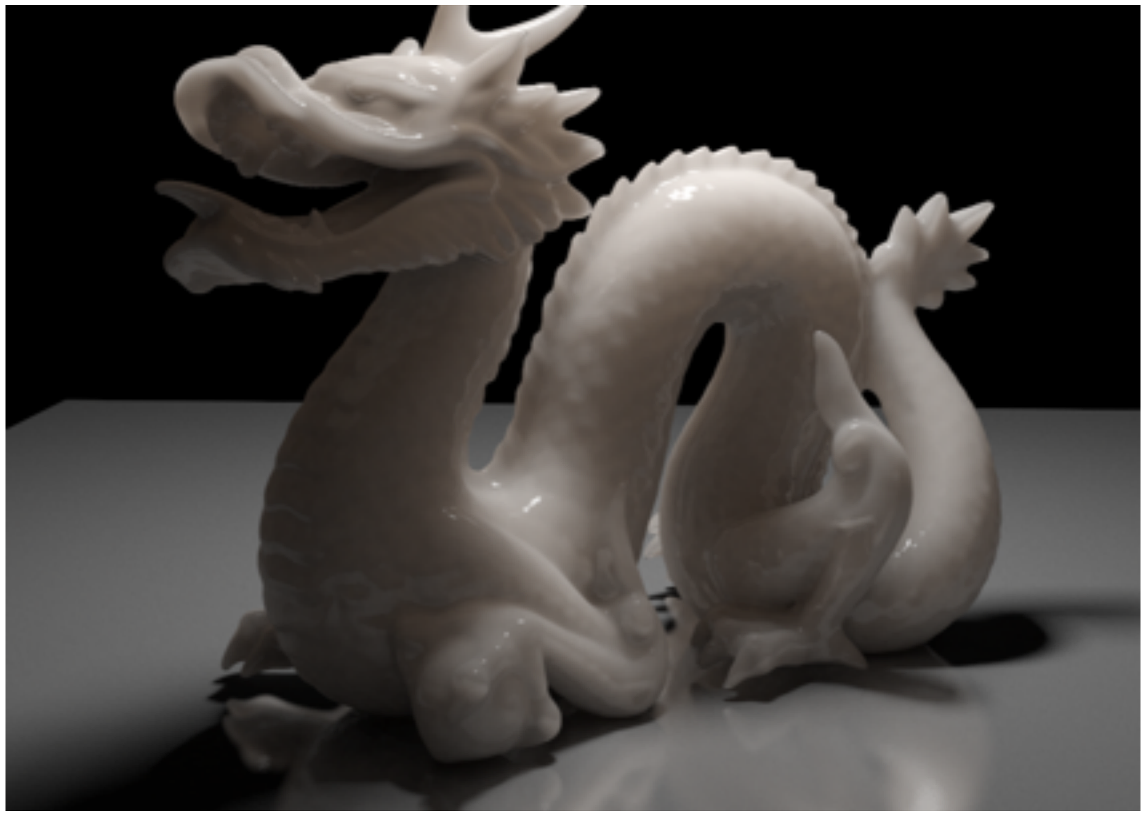


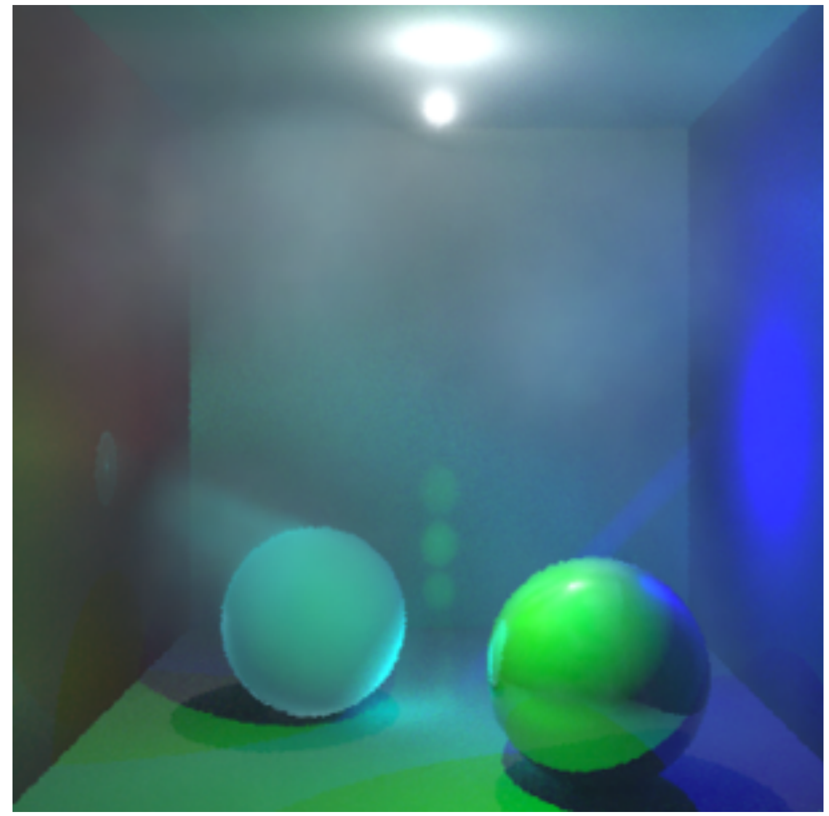
image courtesy Toshiya Hachisuka

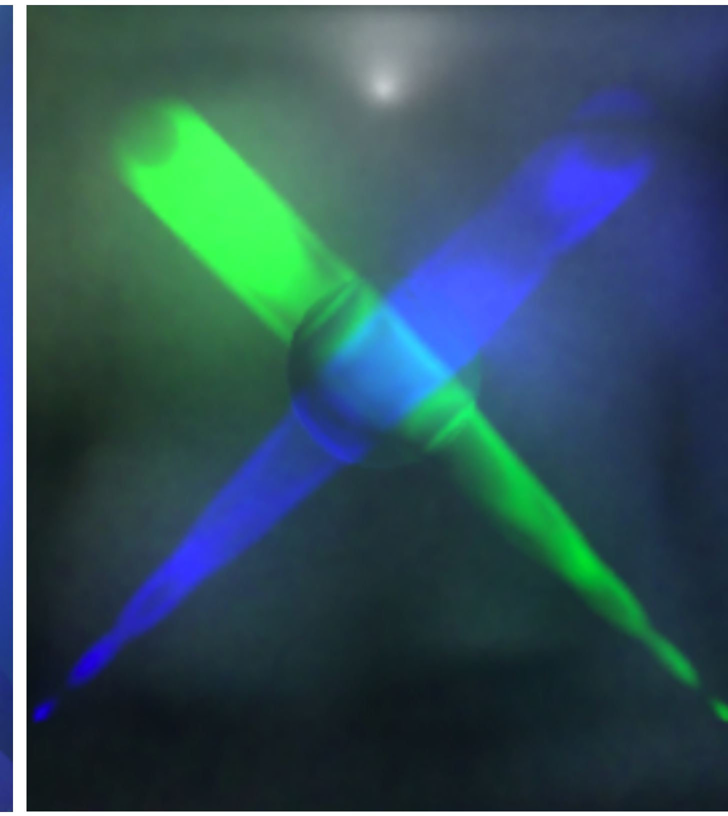
Elective

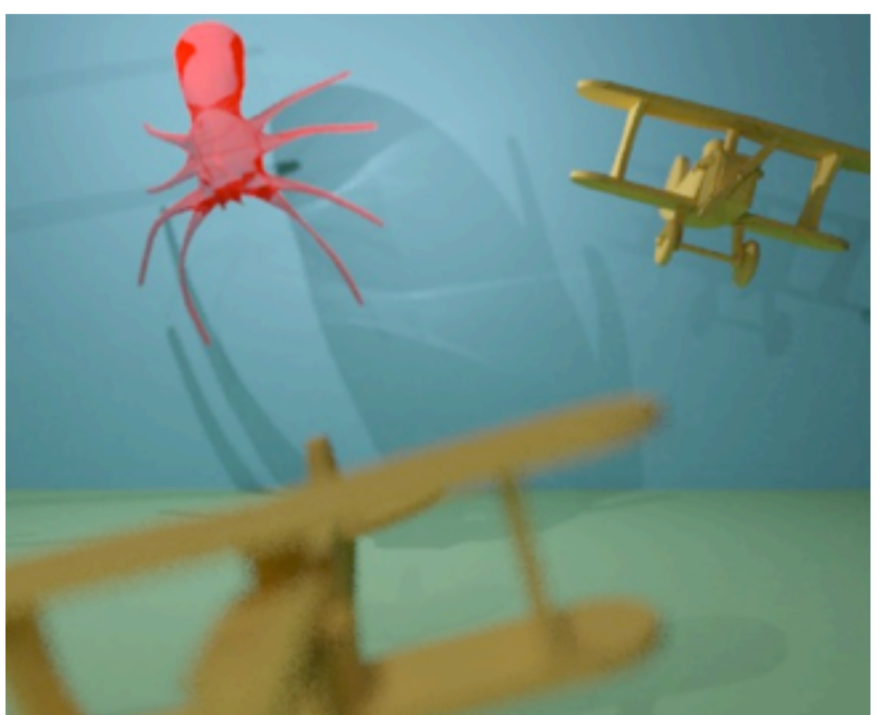
- Create a photorealistic image with an interesting effect
- AKA Assignment 5
- Choose your own topic
- Present an image at last lecture
  - 2 weeks
  - Describe what is interesting and how you did it

Electives from 2011

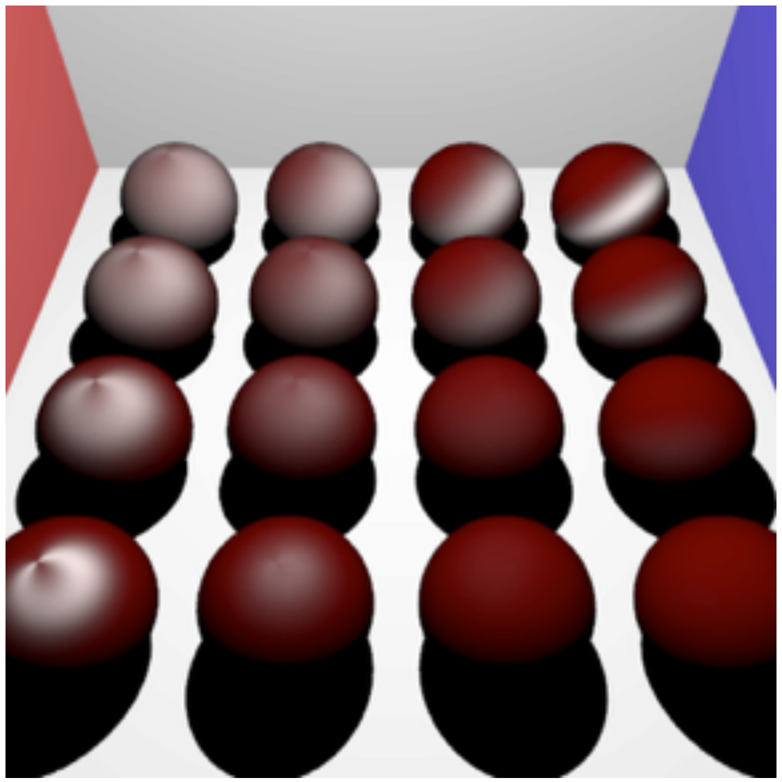


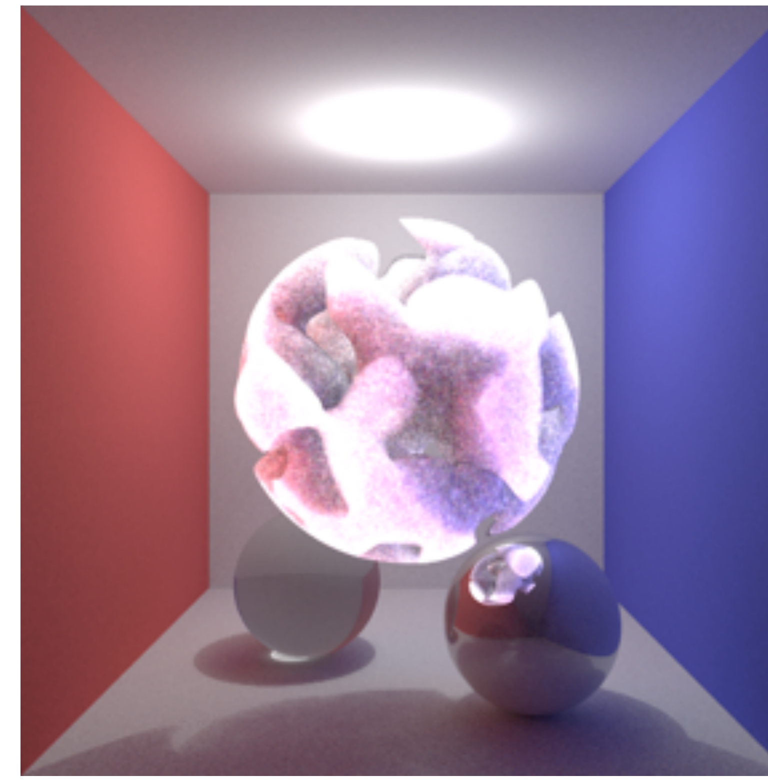


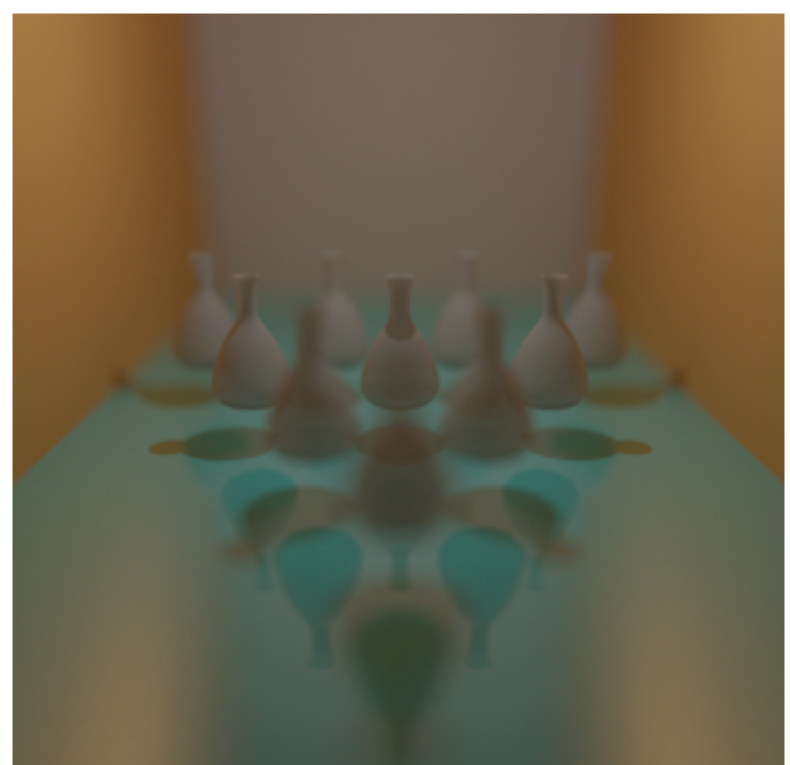
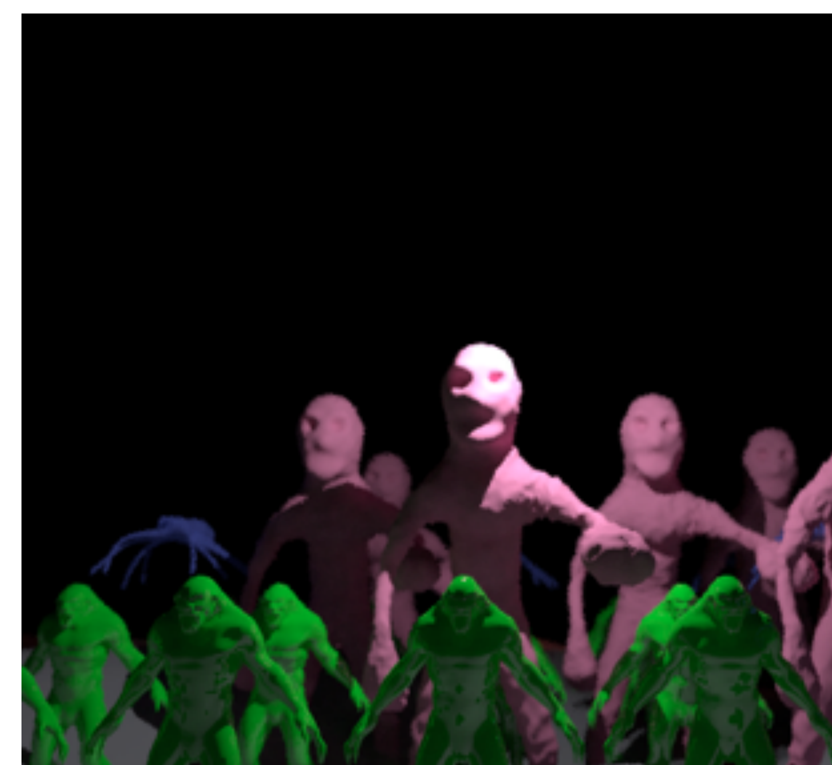




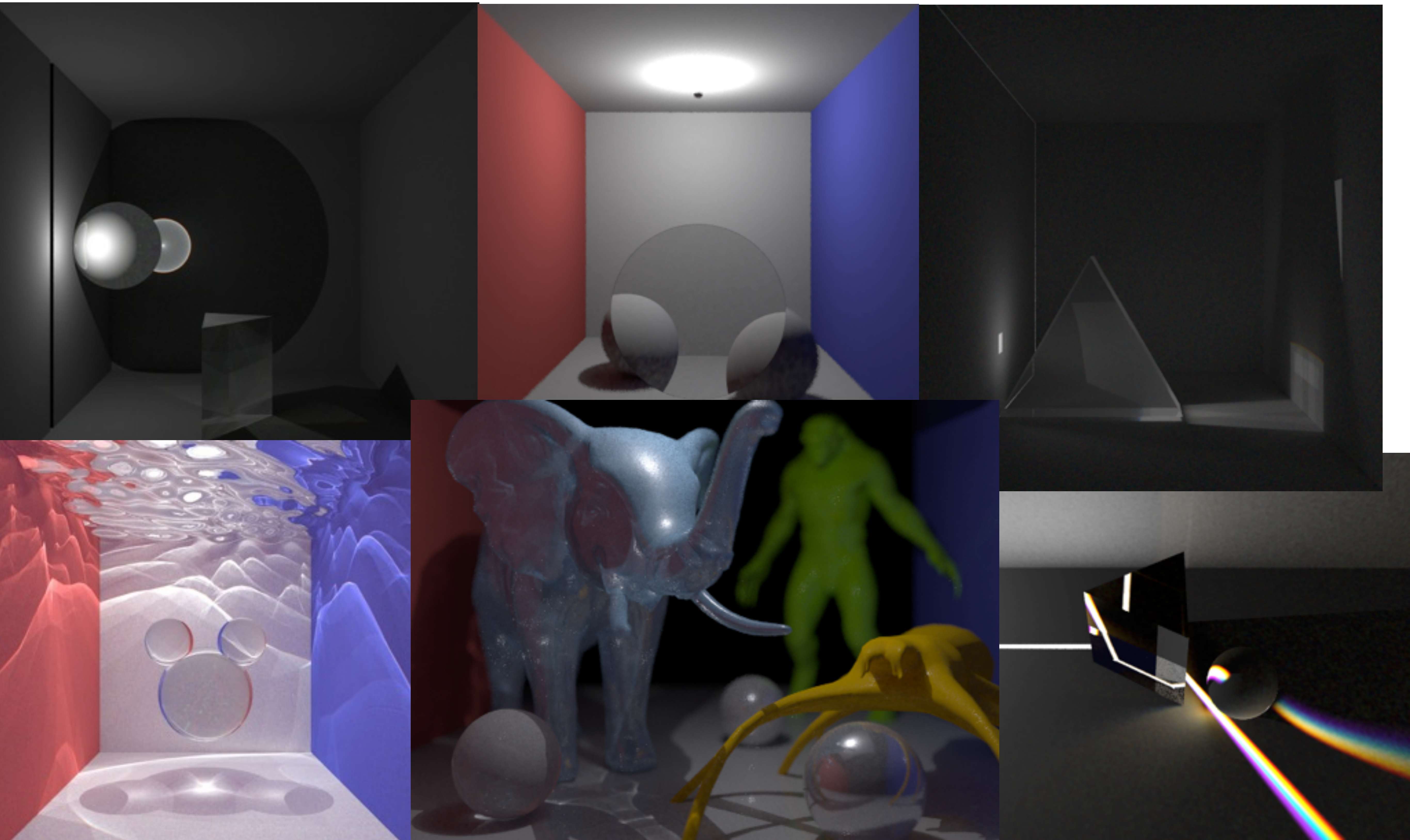
Elective from 2012





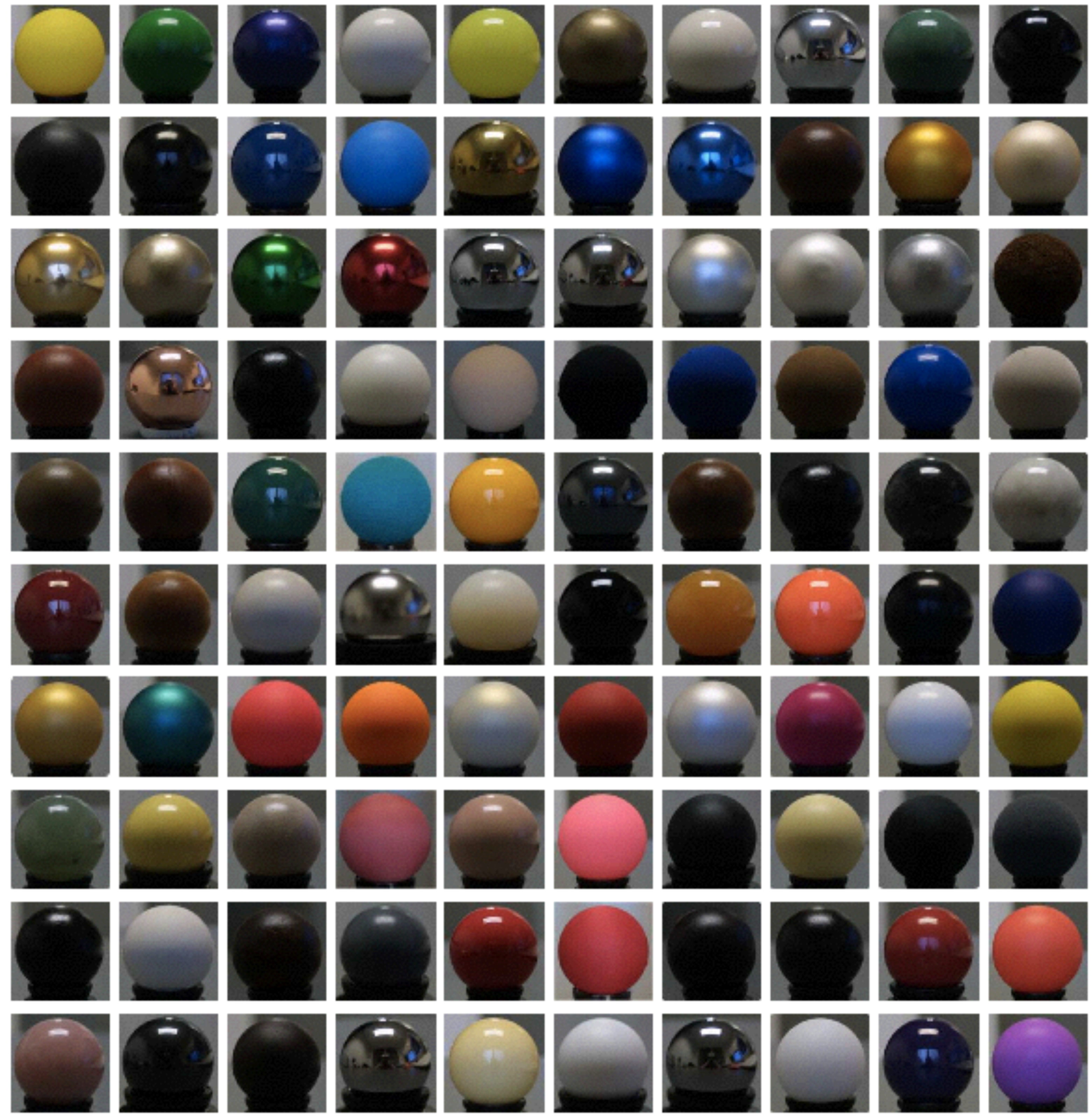



Elective from 2013



BRDFs

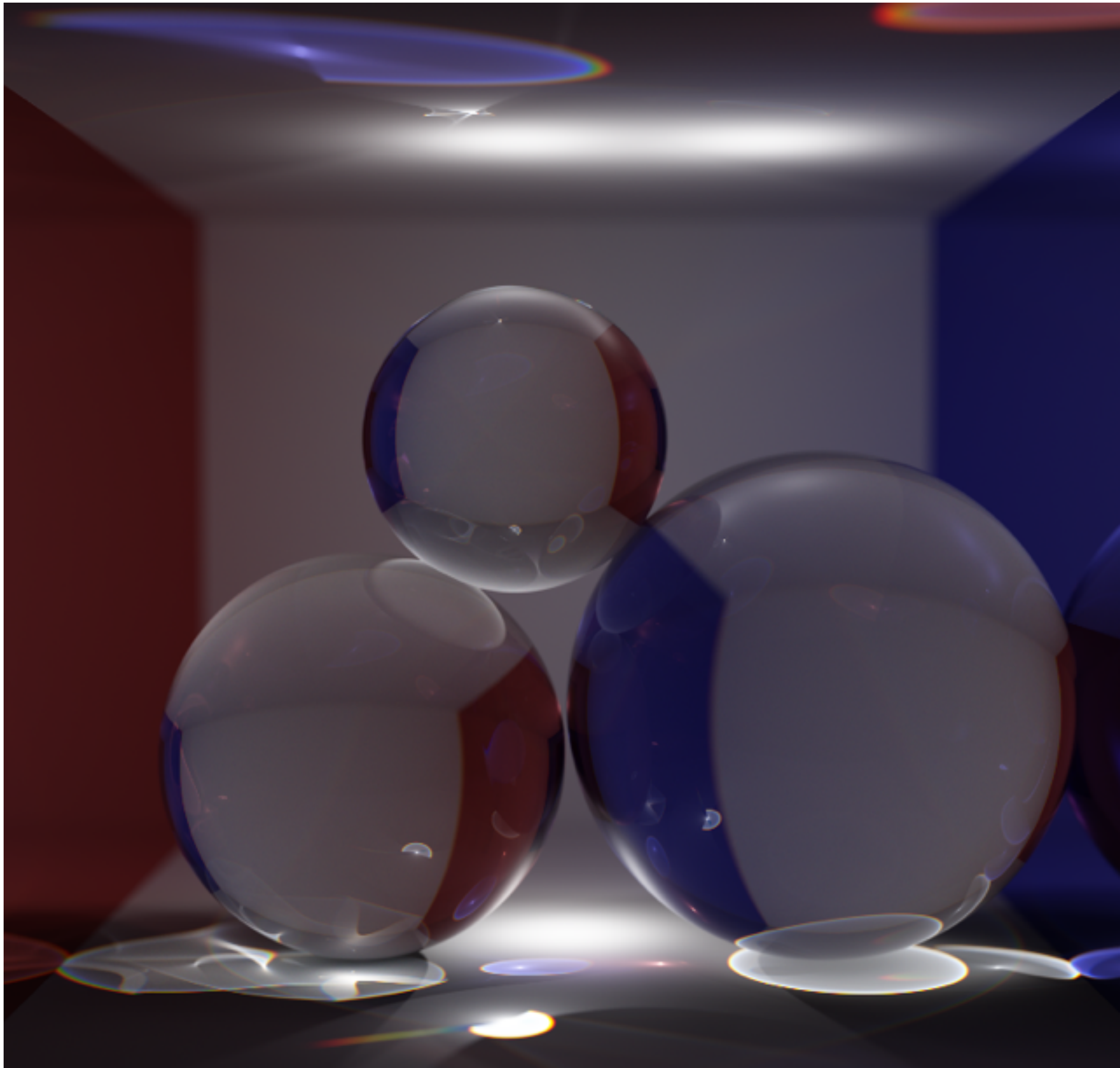
- Need more accurate physically correct model than Phong
- Measurement and analytical models
  - Schlick
  - Cook-Torrance '81
  - Ward '92
  - many others



from "A Data-Driven Reflectance Model", Wojciech Matusik, Hanspeter Pfister, Matt Brand and Leonard McMillan, ACM Transactions on Graphics 22, 3(2003)

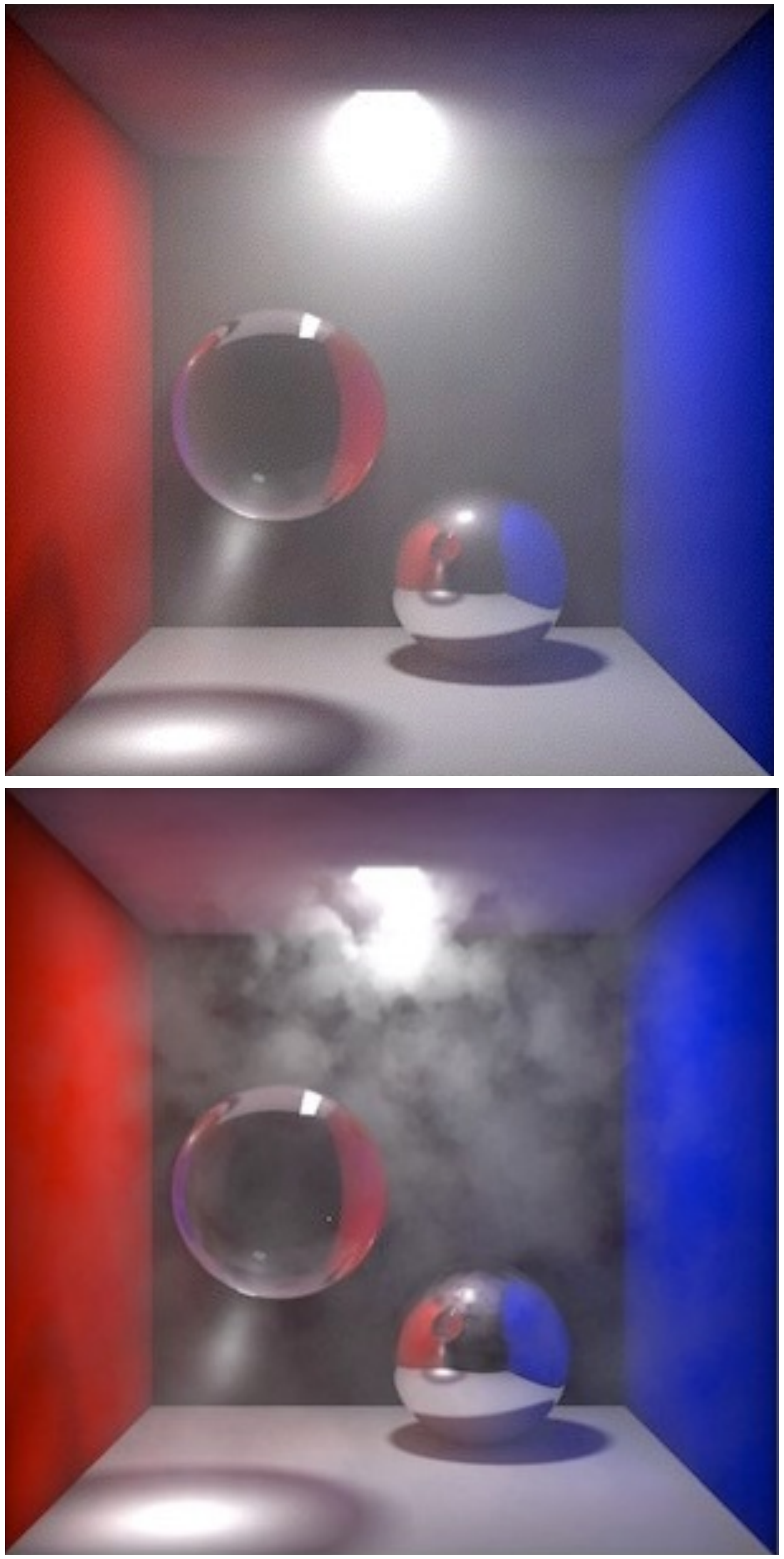
Diffraction

- CDs, prisms
- Need to compute refraction for different wavelengths



Inspiration...

- Subsurface scattering
- Participating media



Images courtesy of Henrik Wann Jensen

Even more inspiration

- Have a look at Stanford University Rendering Competition for more ideas
- http://www-graphics.stanford.edu/courses/cs348b-competition/cs348b-09/

Next

- Friday Lab 3: Path Tracing
- In 4 weeks
  - Monday Seminar: Progressive Photon Mapping
  - Wednesday Lecture: Advanced Topics