Path Tracing

Steve Rotenberg
CSE168: Rendering Algorithms
UCSD, Spring 2014

Global Illumination

- For photoreal rendering, we are interested in simulating *global illumination*
- Global illumination refers to the modeling of all of the light interaction within a scene, including the effects of multiple diffuse and specular bounces
- Light enters the scene from the light sources, bounces around, and reaches a global equilibrium
- Global illumination is described by the *rendering equation*

Rendering Equation

$$L_{out} = L_{emitted} + L_{reflected}$$
$$L(\boldsymbol{\omega}_r) = L_e(\boldsymbol{\omega}_r) + L_r(\boldsymbol{\omega}_r)$$

$$L(\boldsymbol{\omega}_r) = L_e(\boldsymbol{\omega}_r) + \int_{\Omega} f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_r) L_i(\boldsymbol{\omega}_i) \cos\theta_i \, d\boldsymbol{\omega}_i$$

Integral Equations

- A equation that contains a function and an integral of that function is called an *integral equation*
- As with differential equations, there are many different categories of these and many different solution techniques
- The rendering equation is an inhomogeneous, linear, Fredholm integral of the second kind
- The general form of this type of equation is:

$$x(t) = g(t) + \lambda \int_a^b k(t,u)x(u)du$$

- Equations of this form were studied long before the rendering equation was identified

Recursive Integrals

- The reason the rendering equation is challenging to solve is that in order to evaluate the outgoing radiance L at a point, we need to know the incoming radiance L_i from every other point
- This leads to an infinite recursion

Neumann Expansion

$$L(\boldsymbol{\omega}_r) = L_e(\boldsymbol{\omega}_r) + \int_\Omega f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_r) L_i(\boldsymbol{\omega}_i) \cos\theta_i d\boldsymbol{\omega}_i$$

$$L = L_e + \int f_r L$$

$$L = L_e + \int f_r L_e + \int f_r \int f_r L$$

$$L = L_e + \int f_r L_e + \int f_r \int f_r L_e + \int f_r \int f_r \int f_r L$$

$$L = \cdots$$

$$L = L_e + T L_e + T^2 L_e + T^3 L_e \ldots = \sum_{m=0}^{\infty} T^m L_e$$

Light Paths

- In order to classify and compare rendering algorithms, it is convenient to introduce a notation to describe light paths
- In the real world, light starts at a light source and eventually ends up in our eye, after possibly several bounces
- We make a distinction between diffuse bounces and specular bounces, as they bring different challenges and sometimes require different approaches to handle correctly
- A light path is described using the following notation:

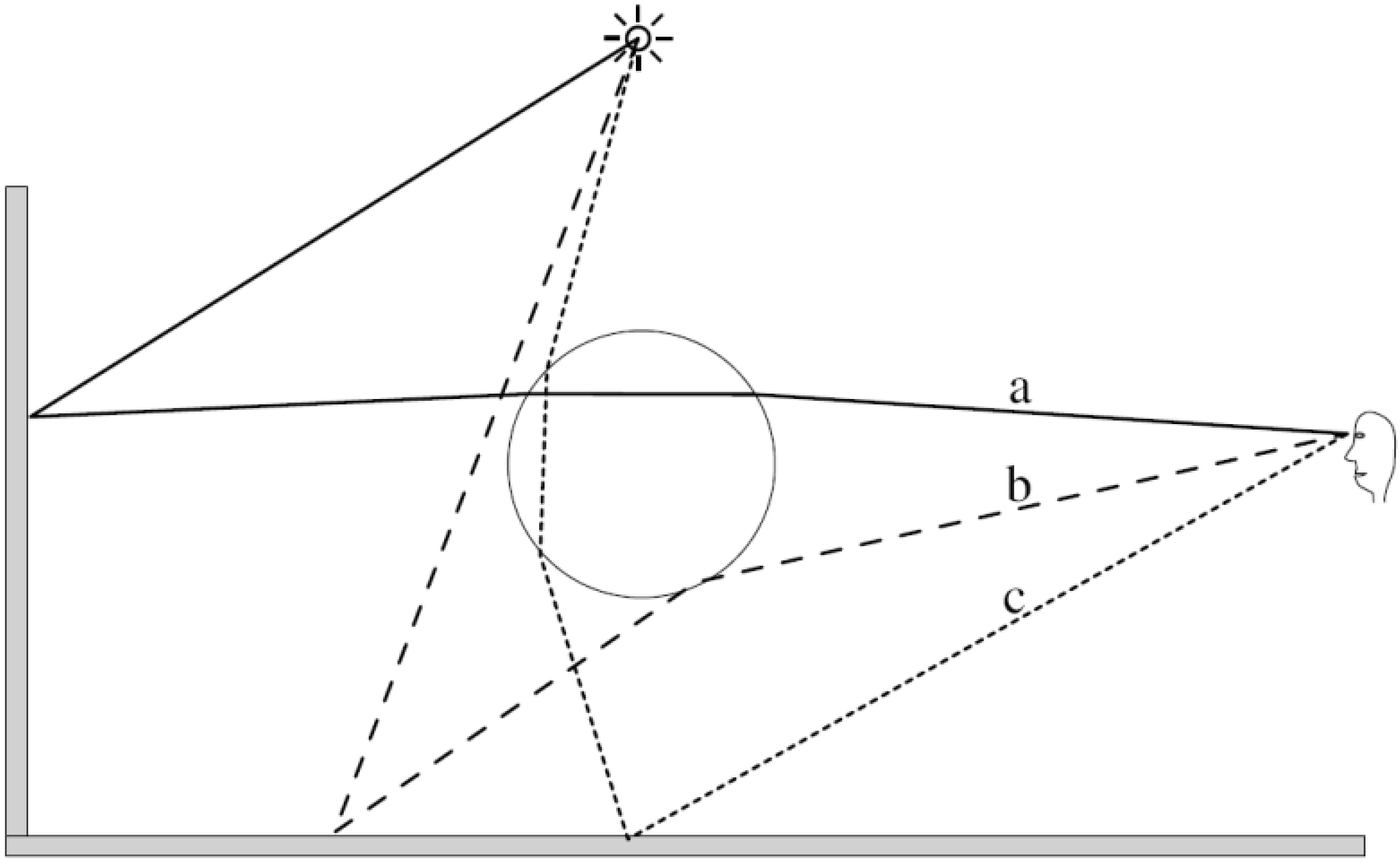
L	Light source
D	Diffuse reflection
S	Specular reflection
E	Eye

- So the direct lighting on a diffuse surface is LDE
- If we are seeing a directly lit diffuse surface through a mirror, we would be seeing the LDSE paths

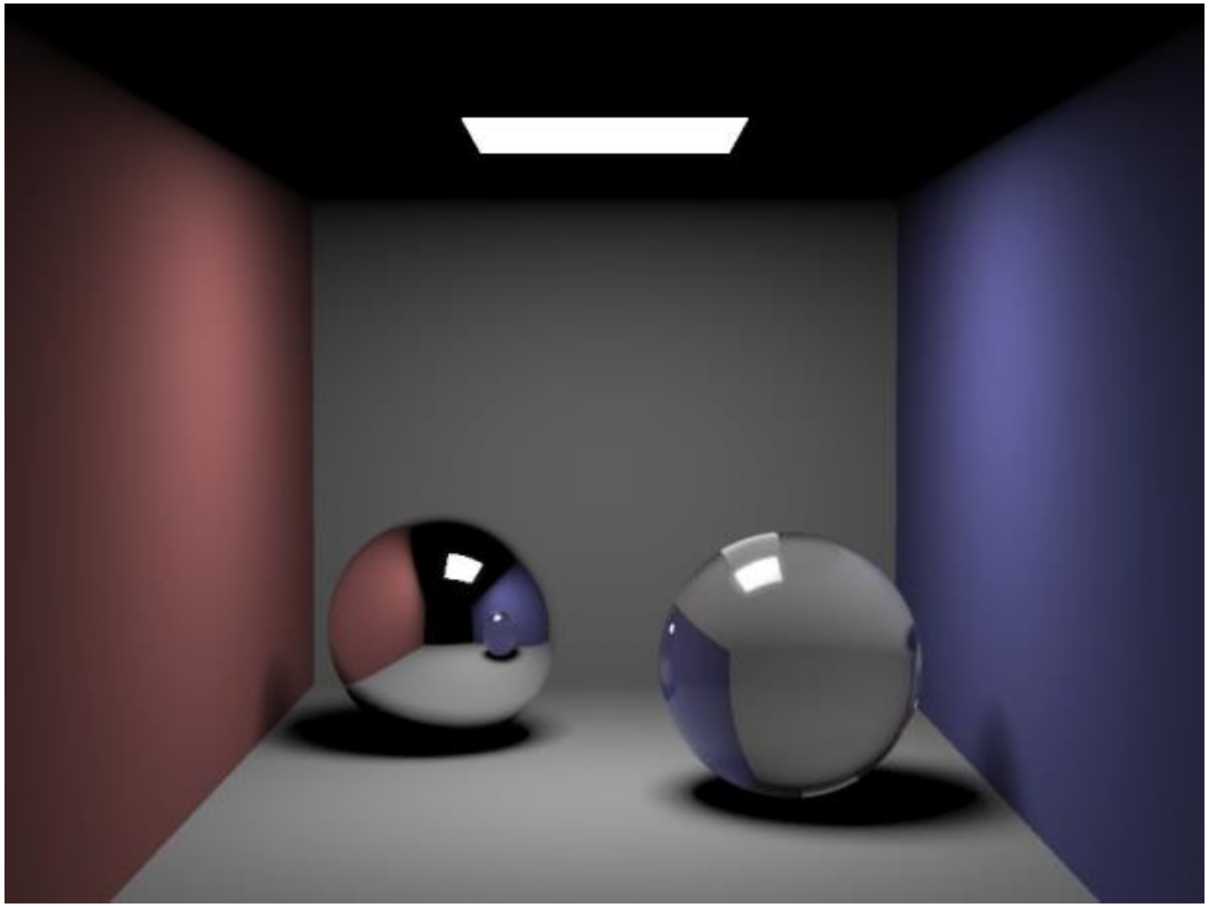
Light Paths

- We can also use some operators in our notation:
  - | or
  - * 0 or more
  - + 1 or more
  - () parenthesis
- For example, L(S|D)E refers to a path that starts at the light, hits either a diffuse or specular surface, and then hits the eye
- LS+DE refers to paths that have at least one specular bounce before hitting a diffuse surface and then hitting the eye
- Global illumination models all light paths (in theory), so would include L(S|D)*E paths

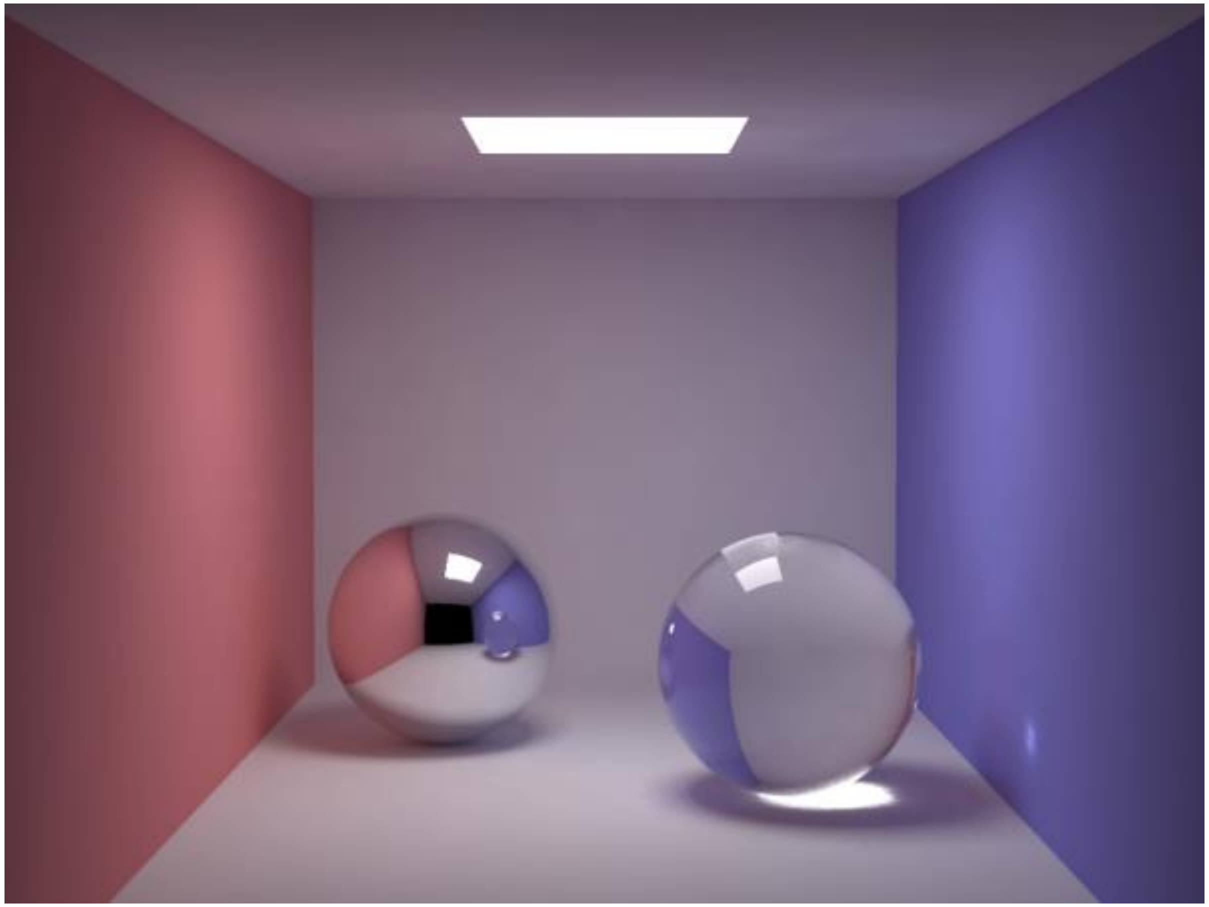
Light Paths



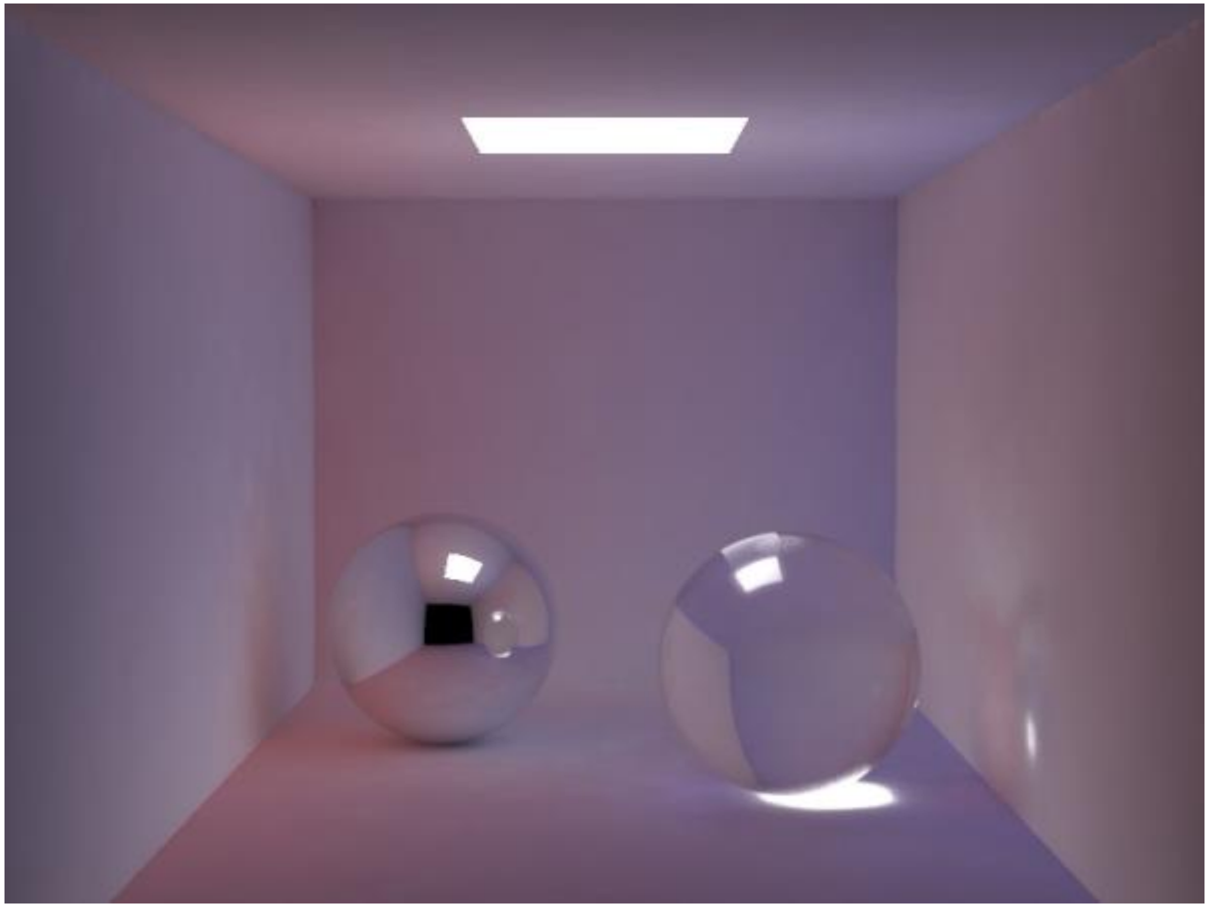
Direct Illumination



Direct + Indirect Illumination



Indirect Illumination



Singularities

- In the real world, light is emitted from surfaces of finite size, and so all lights are essentially area lights
- In computer graphics however, we often simplify lights as ideal points or single directions
- These are examples of singularities, as a point light would essentially have to emit an finite amount of light from an infinitesimal point, leading to an infinite density of light per area
- Some rendering approaches have difficulties with singularities of this type

Caustics

- *Caustics* are areas of focused light, resulting from one or more specular bounces
- Examples are the concentrated beams formed by magnifying glasses, or the animated light patterns at the bottom of swimming pools
- Generally, caustics are caused by light paths of the LS+DE form
- These can be challenging to render

Rendering Algorithms

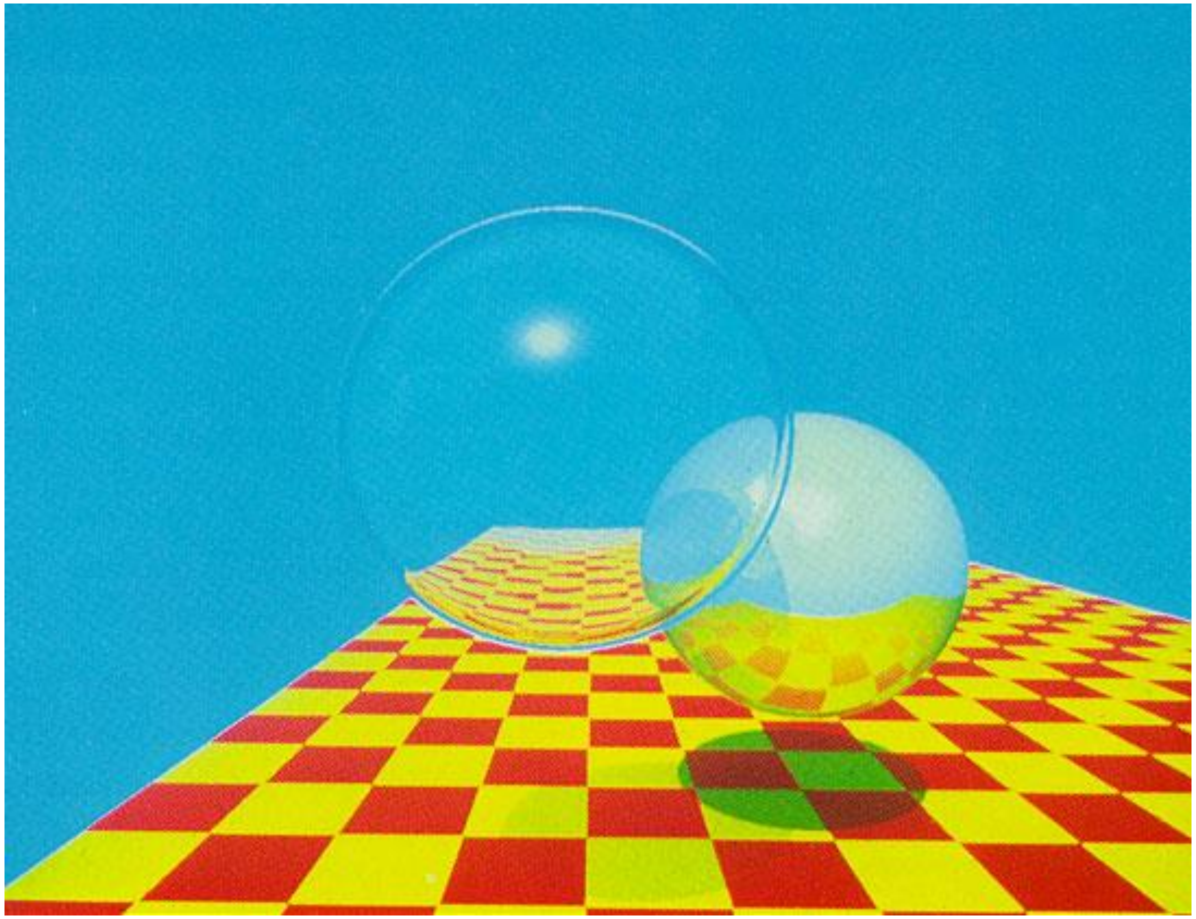
Ray Based Algorithms

- Ray-based rendering algorithms are all evolved from the original ray tracing algorithm and make use of the concept of ray intersection
- They can all be built off of the same foundation of ray intersection algorithms, spatial data structures, and BRDF models
- It is even possible to combine all of these techniques and use each for its strengths

Classical Ray Tracing

- *Classic ray tracing* traces one or more primary rays per pixel from the camera
- If a ray hits a diffuse surface, then additional shadow rays are traced to the lights and the pixel is shaded
- If a ray hits a specular surface, it reflects and/or refracts recursively until a diffuse surface is hit
- Traces LDS*E paths

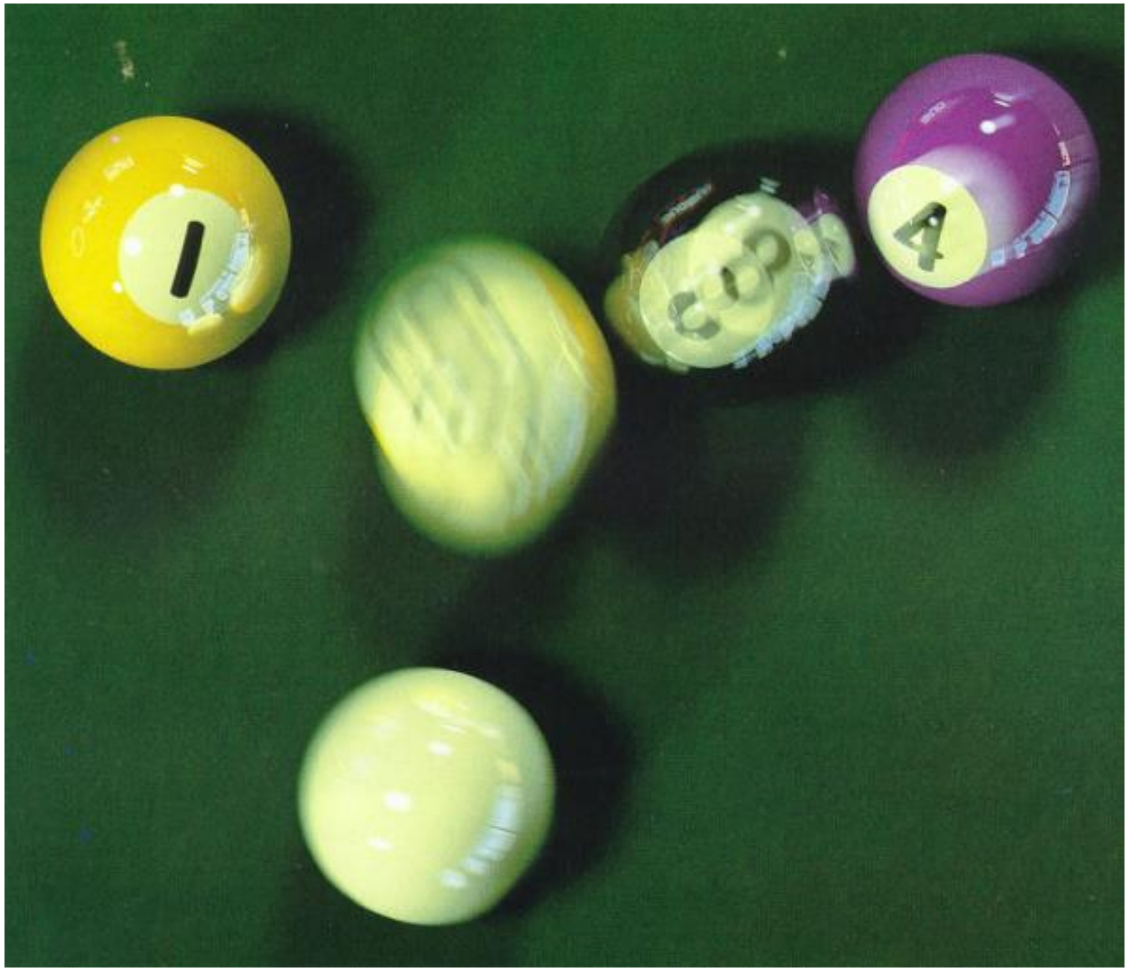
Ray Tracing



Distribution Ray Tracing

- *Distribution ray tracing* added the concept of averaging over a distribution of rays to render softer effects like glossy reflections, soft shadows, camera focus, and motion blur
- The camera test multiple rays per pixel, distributed across the lens and in time
- Reflections will spawn multiple additional rays and diffuse surfaces will test multiple shadow rays per area light
- This approach can lead to an exponential growth in the number of rays per pixel
- It traces the same LDS*E paths as classic ray tracing, but supports a wider range of material types and visual effects

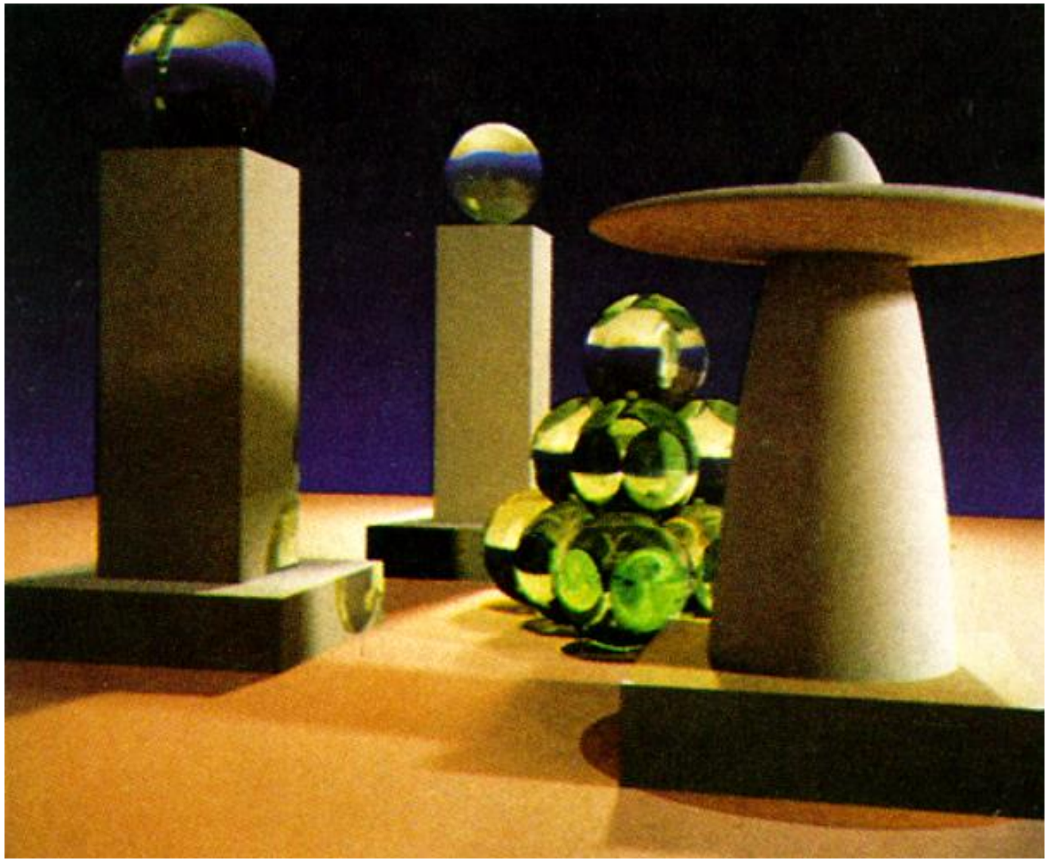
Distribution Ray Tracing



Path Tracing

- *Path tracing* traces one or more primary rays per pixel
- When a ray hits a surface, it traces one or more shadow rays towards light sources, and also spawns off one additional reflection ray
- The reflected rays form a path, where each node of the path connects to the light source
- Paths are randomly generated and their length is determined through probabilistic means
- Path tracing is capable of tracing all types of light paths L(S|D)*E, but can suffer from excessive noise in some situations
- It is generally good for LD*S*E paths, but is bad at LS*(S|D)*E paths (i.e., paths with caustics) and fails for any LS(S|D)*E paths if the light source is a singularity (i.e., a point or single direction)

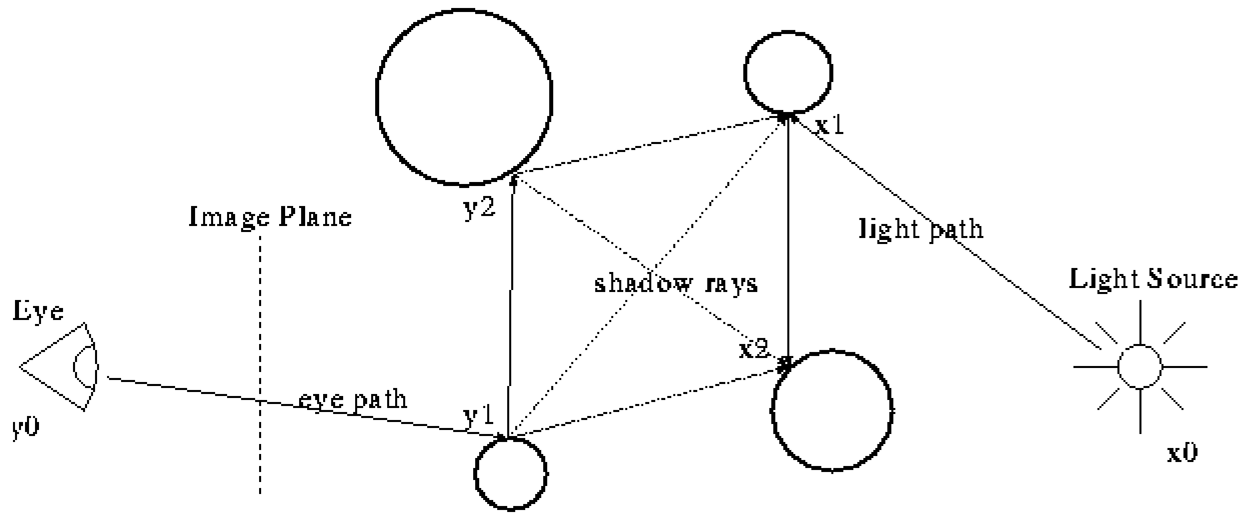
Path Tracing



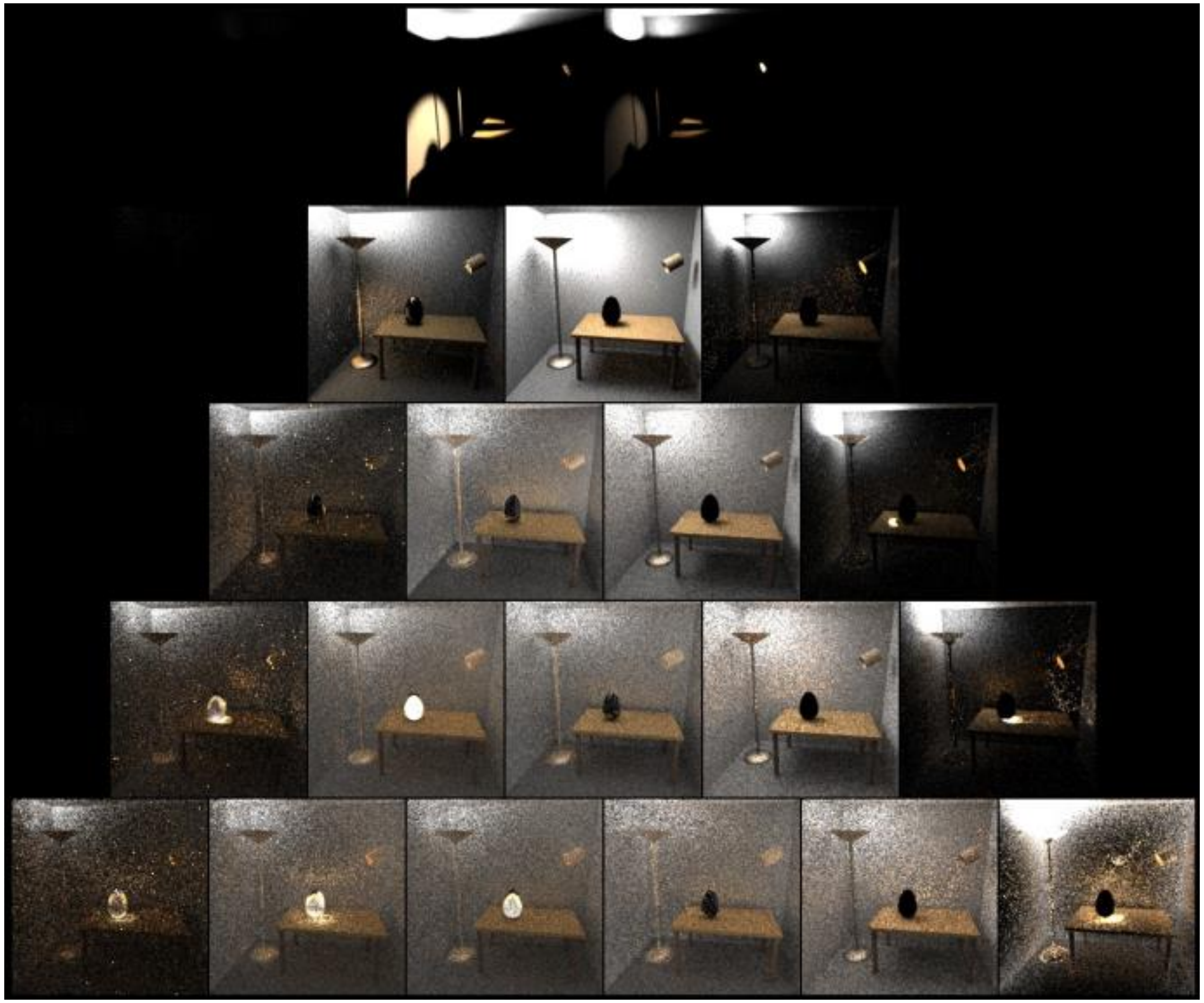
Bidirectional Path Tracing

- *Bidirectional path tracing* (BPT) is an extension to path tracing
- A path is traced from both the camera and a light source
- Then, each vertex of the eye path is connected to each vertex of the light path to form all possible combinations of the two paths
- If the connecting ray is not blocked, then the path is added to the pixel, according to its weighted probability
- BPT is better at handling indirect lighting situations, and will generally do a better job than standard path tracing in most practical cases

Bidirectional Path Tracing



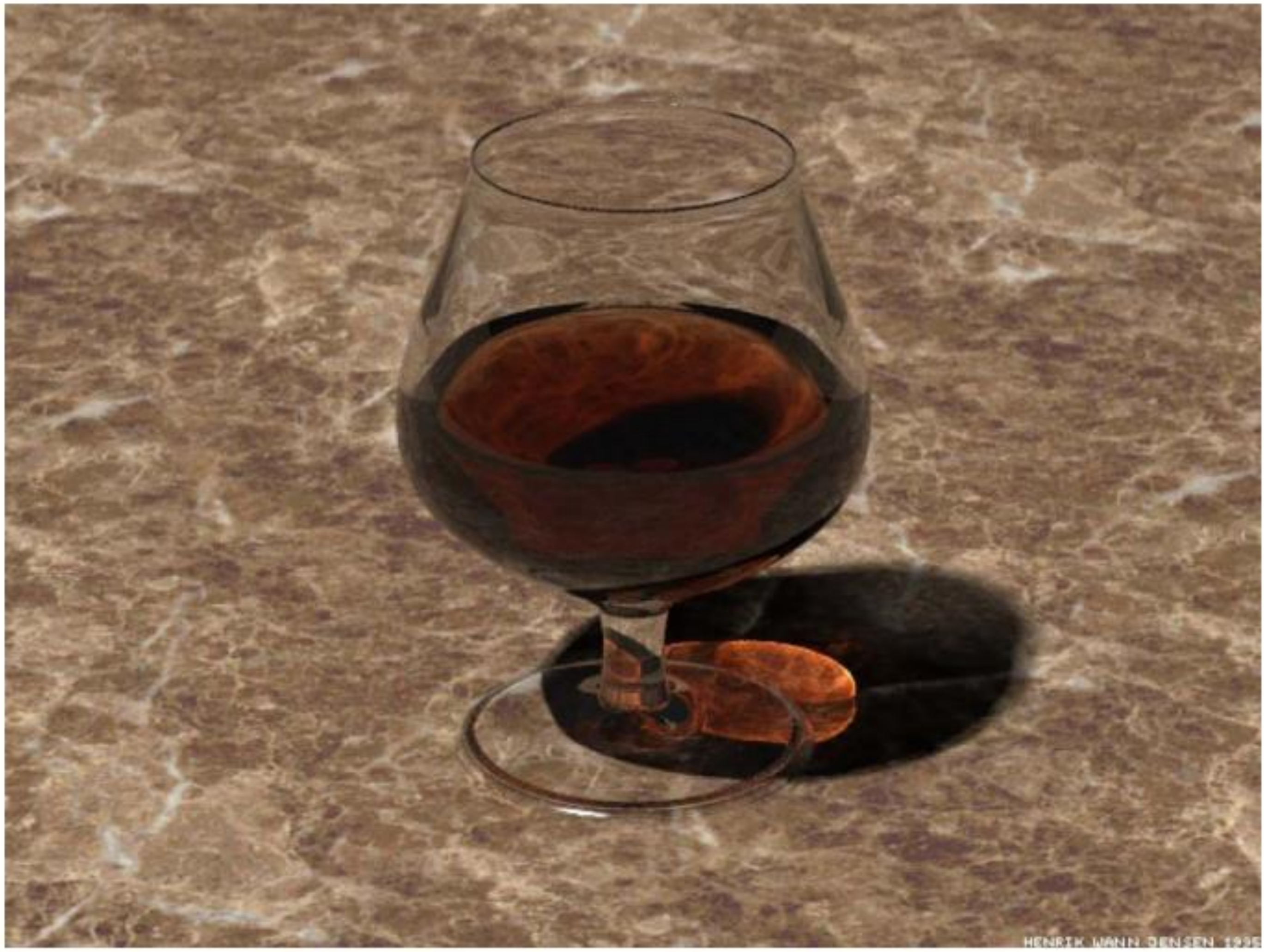
Bidirectional Path Tracing



Photon Mapping

- *Photon mapping* is a two-pass rendering system
- In the first pass, photons are emitted from the light sources and scatter through the environment
- At each bounce, they are either randomly reflected according to the BRDF of the surface or they are absorbed by the material, and their 3D position is added to the *photon map*
- Often, millions of photons are shot out, and need to be efficiently stored and managed
- In the second pass, the scene is path traced
- Certain components of the lighting are computed by with the standard path tracing approaches (such as direct lighting and specular reflections)
- Other components of the lighting can come from the photon map (such as highly specular paths like caustics)
- The photon map can also be used for the indirect diffuse components as well, and can be combined with a technique called *final gathering*
- Photon mapping combines with path tracing and can be balanced to gain the benefits that each approach offers

Photon Mapping



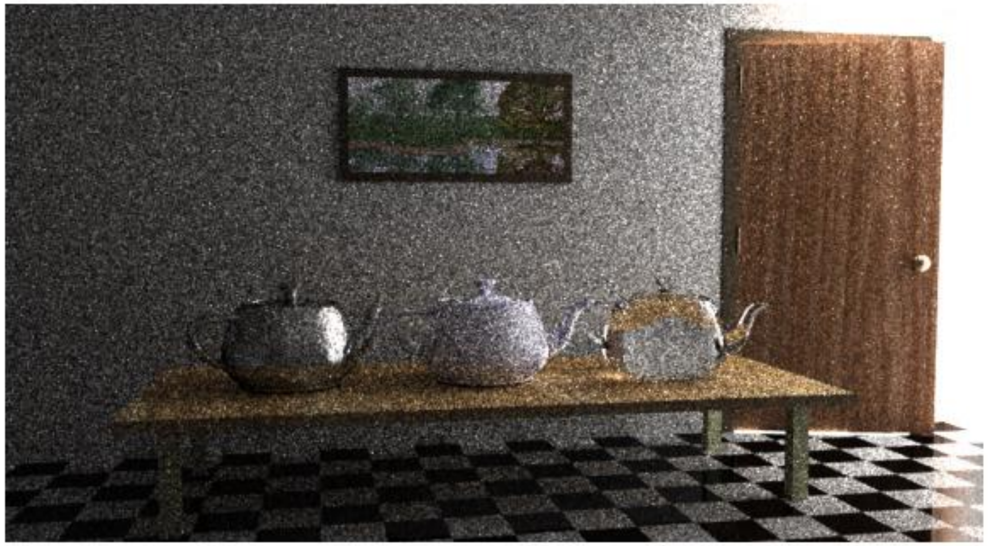
Metropolis Light Transport

- *Metropolis light transport* (MLT) is a rather exotic form of path tracing that has been gaining popularity lately, as it is notable for its ability to handle particularly complicated light paths
- For each pixel, an initial path is created that starts from the camera, bounces in the scene, and ends up at a light source
- Then, the path is modified (or *mutated*) according to a variety of possible rules
- Next, the path is either accepted or rejected according to a random decision, weighted by the relative contribution of the new path to the previous path
- The current path then adds its contribution to the pixel and the mutation process is repeated some desired number of times

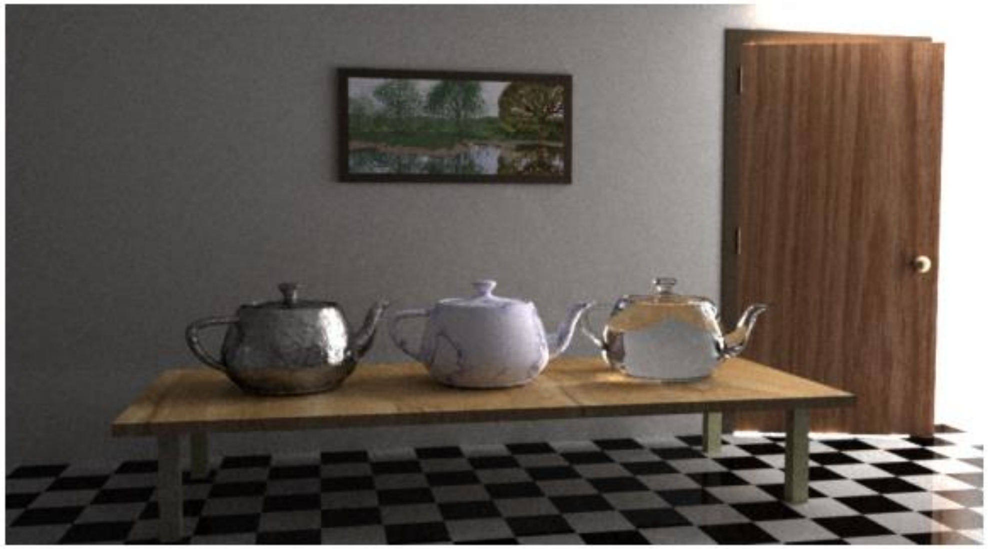
```
x = x0
for i = 1 to n
    x' = mutate(x)
    a = accept(x, x')
    if (random() < a)
        x = x'
    record(x)
```

Metropolis Algorithm

BPT: 40 samples per pixel



MLT: 250 mutations per pixel



Ray Tracing References

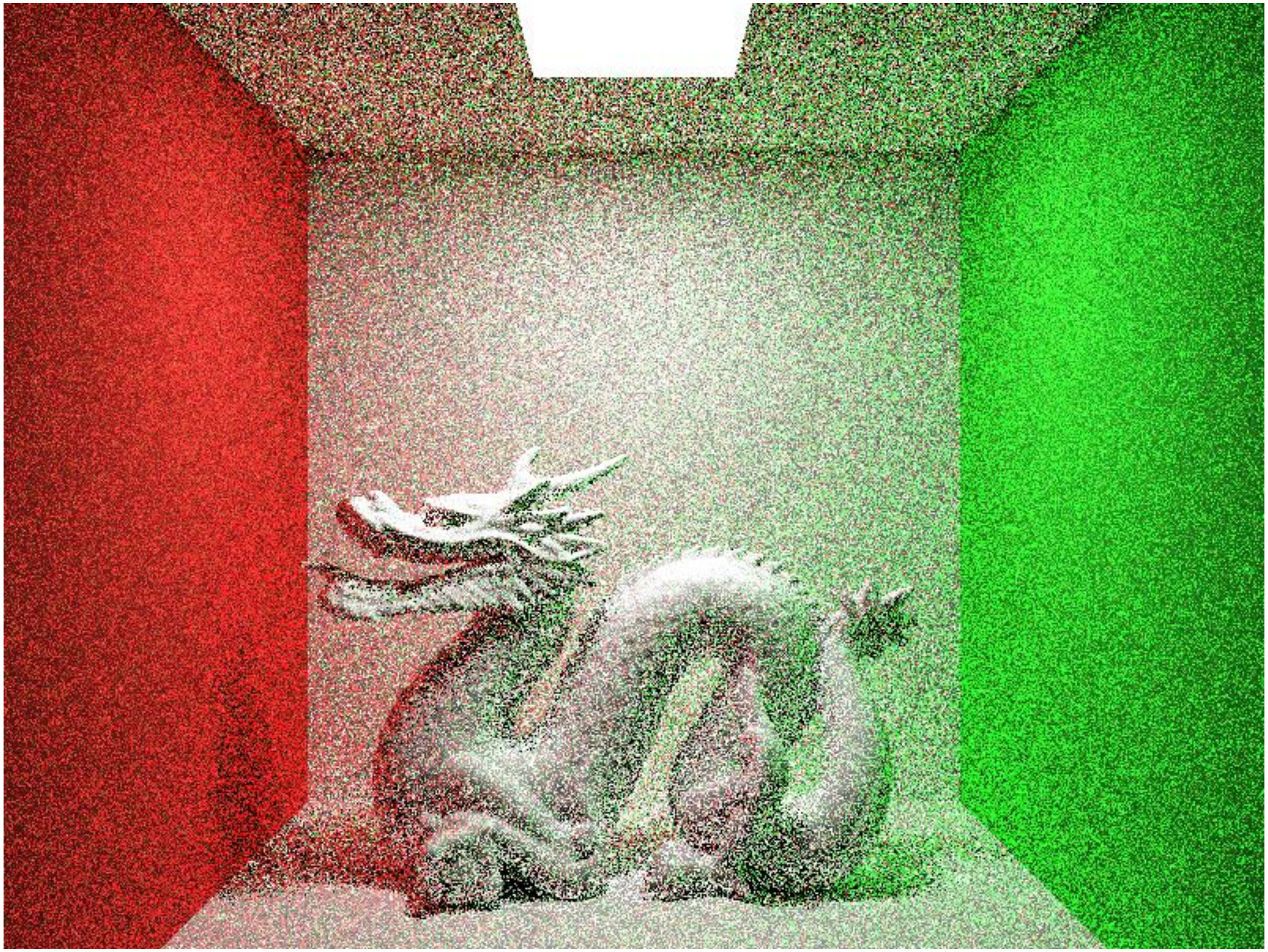
- “An Improved Illumination Model for Shaded Display”, Turner Whitted, 1980
- “Distributed Ray Tracing”, Cook, Porter, Carpenter, 1984
- “The Rendering Equation”, James Kajiya, 1986
- “Bidirectional Path Tracing”, Eric Lafortune, Yves Willems, 1993
- “Rendering Caustics on Non-Lambertian Surfaces”, Henrik Wann Jensen, 1996
- “Metropolis Light Transport”, Eric Veach, Leonidas Guibas, 1997

Non-Ray Based Methods

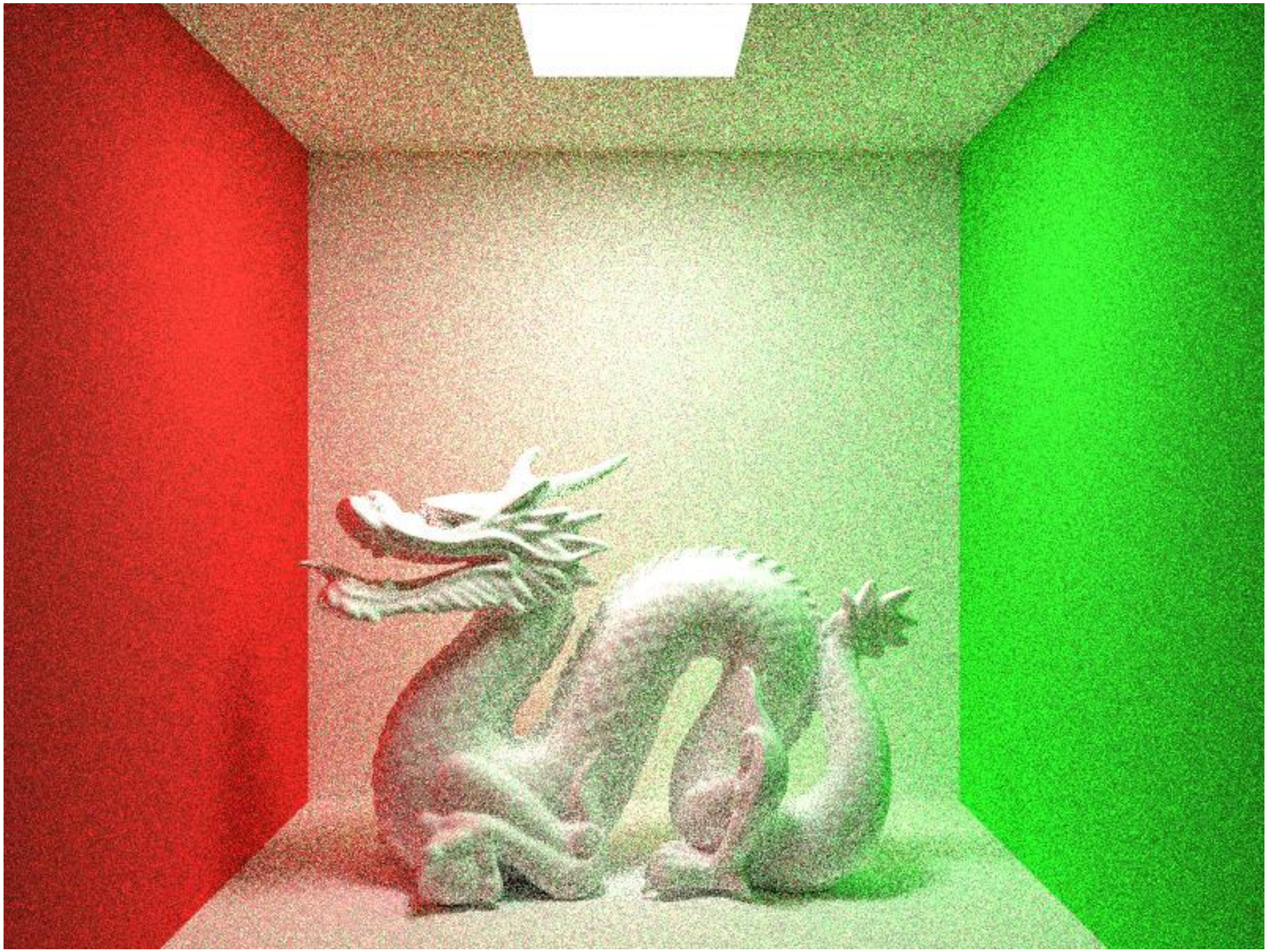
- Not all rendering algorithms are based on ray tracing
- A lot of research went into a technique called *radiosity* (or the *radiosity method*) between 1984 and around 2000
- The radiosity method starts by dividing up all of the surfaces of the scene into tiles
- A matrix of *form factors* is computed, which relates the contribution of every tile to every other tile's incoming light
- This matrix is then used to compute the total interreflection of light throughout the environment
- Radiosity is well suited for diffuse environments and can precompute all diffuse interreflection for real time viewing
- It fails however for specular reflection and requires extensive modification to be able to even attempt to handle specular situations

Path Tracing

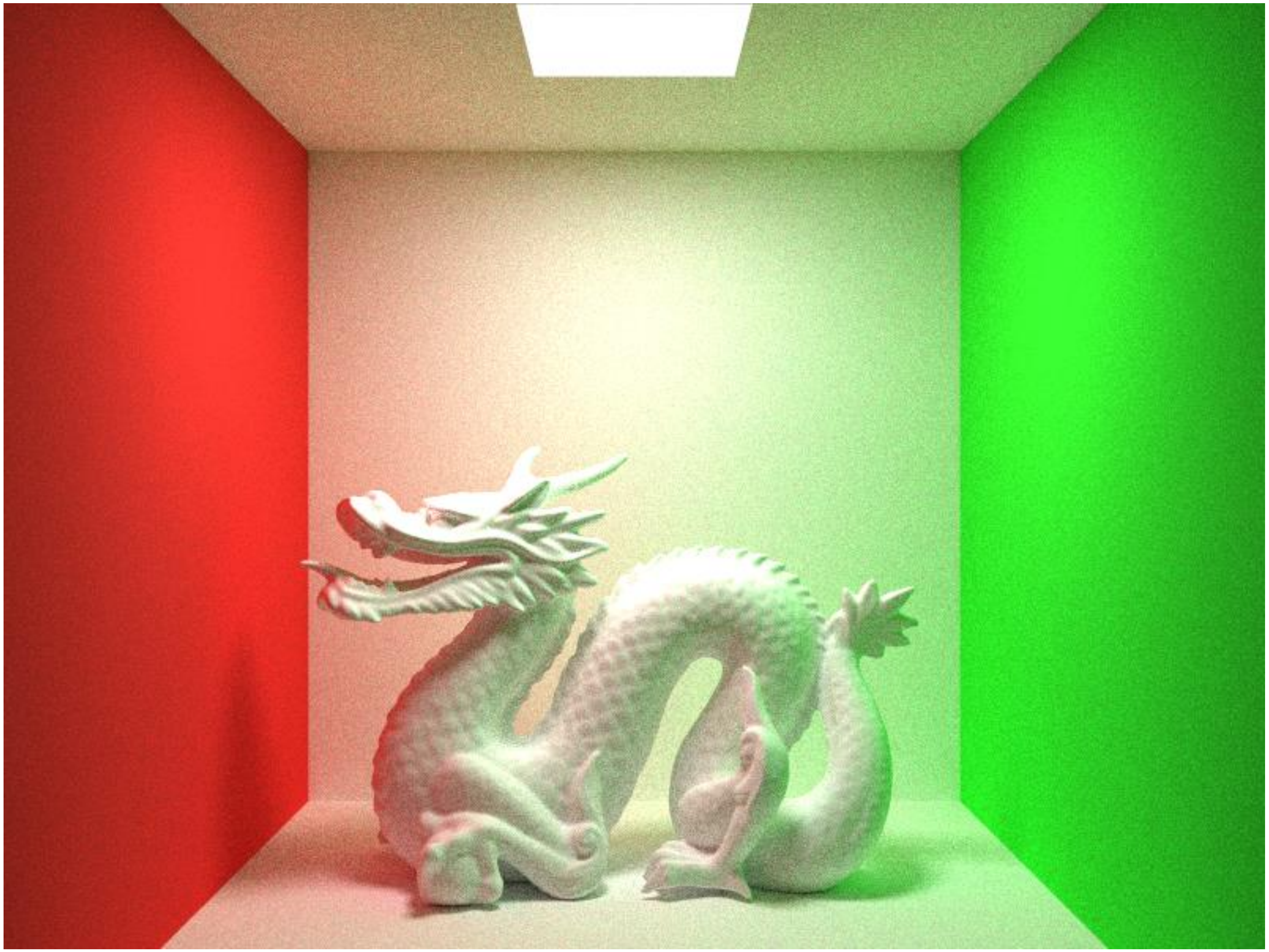
1 Path / Pixel



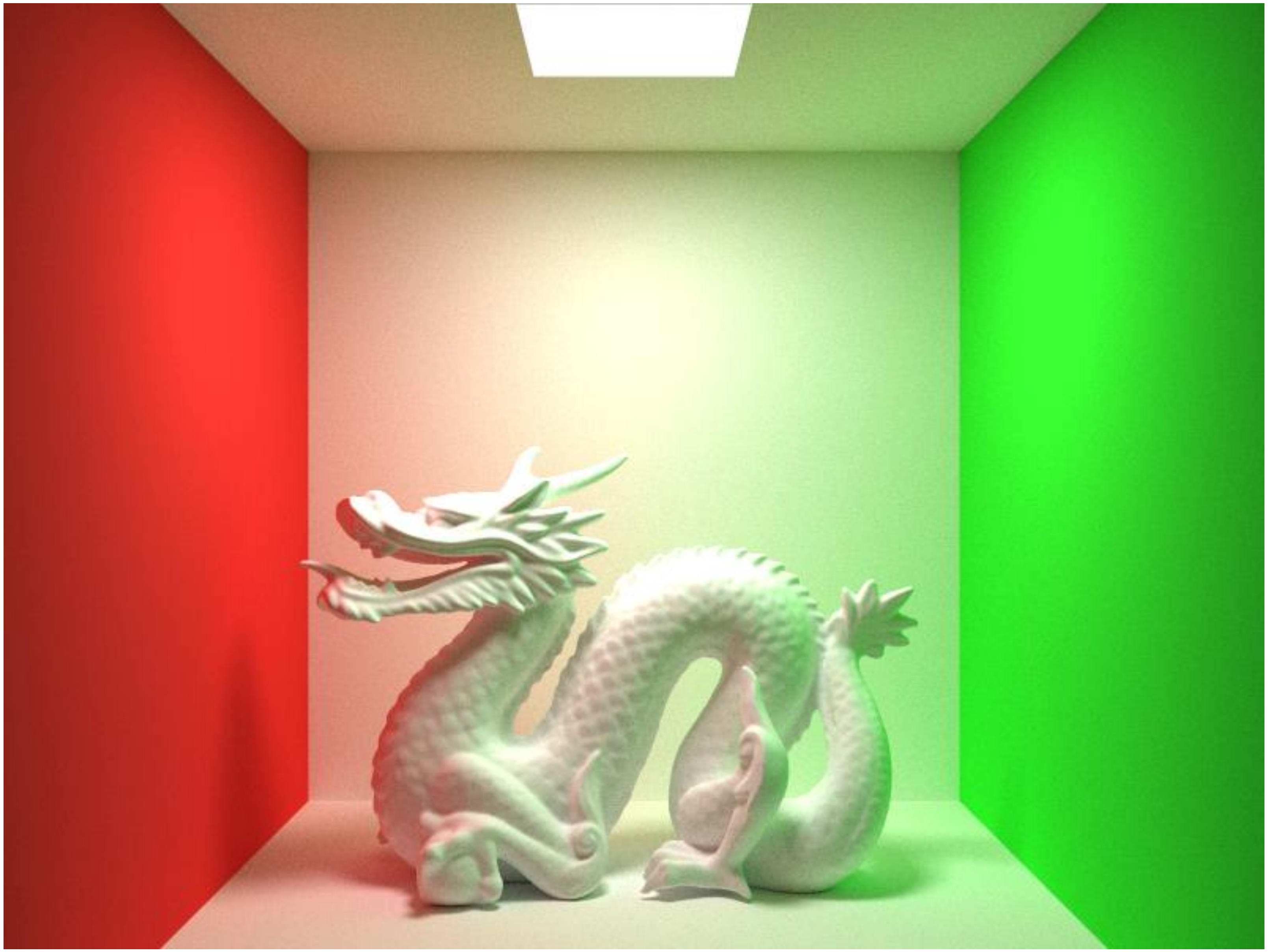
10 Paths / Pixel



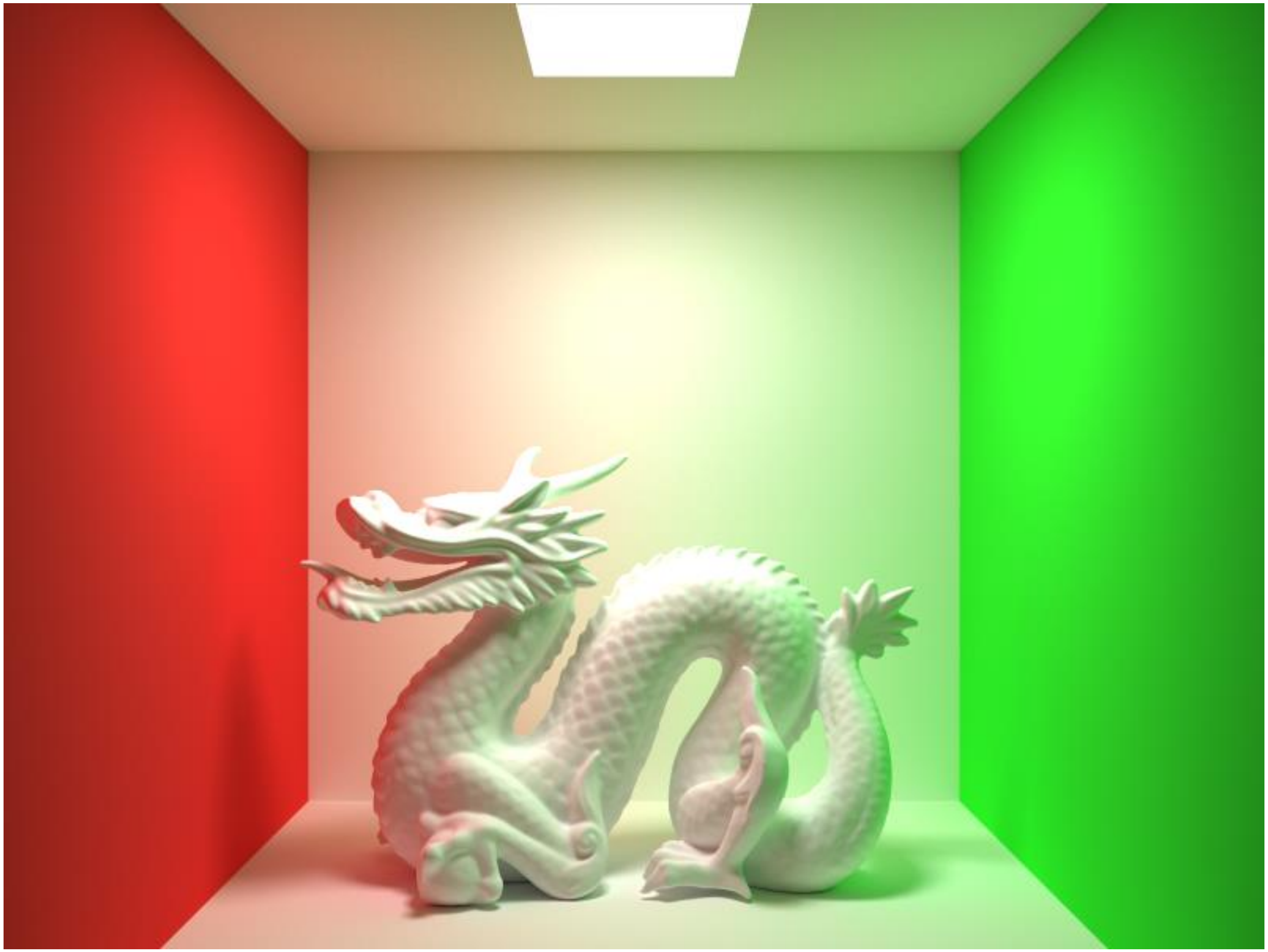
100 Paths / Pixel



1000 Paths / Pixel



10000 Paths / Pixel



Path Tracing

- There are several standard extensions to path tracing that are typically used to get the most out of the technique
- We will look at some of the more popular methods

Path Length

- Some lighting situations do not involve a lot of bounces, such as outdoor lighting, and may look fine with only two or three bounces
- Indoor scenes with bright lights and bright walls will bounce contain a lot of indirect light and will require more bounces (maybe 5 or more)
- Scenes with shiny objects and dielectrics may require many bounces to render correctly (10+)
- It is always possible to construct a situation where we need even more bounces (such as a hall of mirrors)
- How do we determine how long our paths should be without resorting to just picking some number?

Russian Roulette

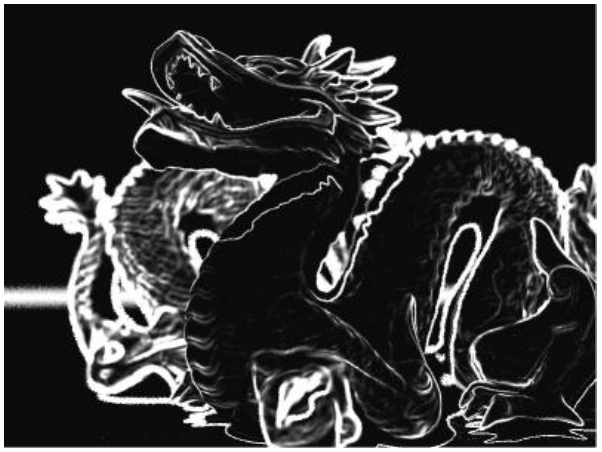
- If we limit the number of bounces to a fixed value, then we are introducing bias (consistent error) to the image as we will underestimate the total brightness
- Russian roulette is a strategy that randomly determines whether or not to terminate a path at each bounce, based on the total contribution of the path
- Above a certain brightness threshold, all bounces are accepted. Below that threshold, we assign a probability of acceptance
- If, for example, we determine that there is a $1/N$ chance of accepting a particular bounce, then we scale its contribution by N if it is accepted

Stratified Sampling

- We looked at examples of stratified (jittered) sampling for pixel antialiasing
- The same approach can be extended into higher dimensional spaces
- For example, the paths we create in path tracing can be stratified
- Each bounce represents two additional dimensions, since the reflected ray will scatter in some direction on a hemispherical surface
- A path of length 5 has 4 bounces, and could be represented with 8 numbers from 0 to 1
- To generate a set of stratified paths, we could divide up an 8 dimensional space into $2^8 = 256$ stratified paths
- Obviously, this approach can require many paths, but that is typically the case anyway
- An alternative is to use quasi-random vectors in 8D space, which has the advantage of not requiring a fixed number of paths, and is more compatible with adaptive approaches

Adaptive Sampling

- Adaptive sampling can be used to determine when enough paths have been traced to reduce the error below some desired threshold
- This allows the computation time to be spent in the areas that need it
- We looked at this in detail in a previous lecture



Importance Sampling

- *Importance sampling* refers to the technique of concentrating the sampling the integral in areas of high importance
- Rather than take a bunch of samples that contribute highly varying amounts to the final estimate, we would prefer to take samples that are all roughly equally important, so that we can make the most of each sample
- For rendering, this means that we want to concentrate our reflection rays in directions that contribute a lot to the reflected light
- In other words, we want to generate samples that correspond to the reflectivity described by the material BRDF

Integral Partitioning

- The integral portion of the rendering equation contains the product of the hemisphere of incoming light with the BRDF
- If we think about the incoming hemisphere of light, we can separate it into the direct light coming straight from the light sources and the indirect light coming from one or more bounces
- We expect that the direct light is bright and has strong boundaries, and we also have the benefit of knowing exactly where the light sources are
- We expect the indirect light to be smoother, but we do not have any knowledge ahead of time about how it is distributed
- This is a good reason to treat those two parts of the integral differently, and use different techniques to evaluate them
- *Partitioning* is the process of breaking an integral into multiple parts, evaluating each part with a different technique, and then summing up the results

Integral Partitioning

- This is what we are doing at each bounce, when we shoot a shadow ray to the light source and shoot a reflection ray
- The shadow ray is evaluating the direct lighting part and the reflection ray is evaluating the indirect light
- With point and directional lights, it would be impossible for a bounced ray to hit the light, so we are enforcing this partitioning
- With area lights, we have to make sure that if a bounced ray hits an area light, then we ignore the emission contribution from the light, as it would have already been considered in the direct lighting partition

Multiple Importance Sampling

- Depending on the surface properties, sampling the reflection (BRDF) partition or sampling the light partition may generate excessive noise
- *Multiple importance sampling* is a technique for optimally combining the two by estimating the noise of each and weighting the result towards the less noisy, allowing for a smooth transition between techniques

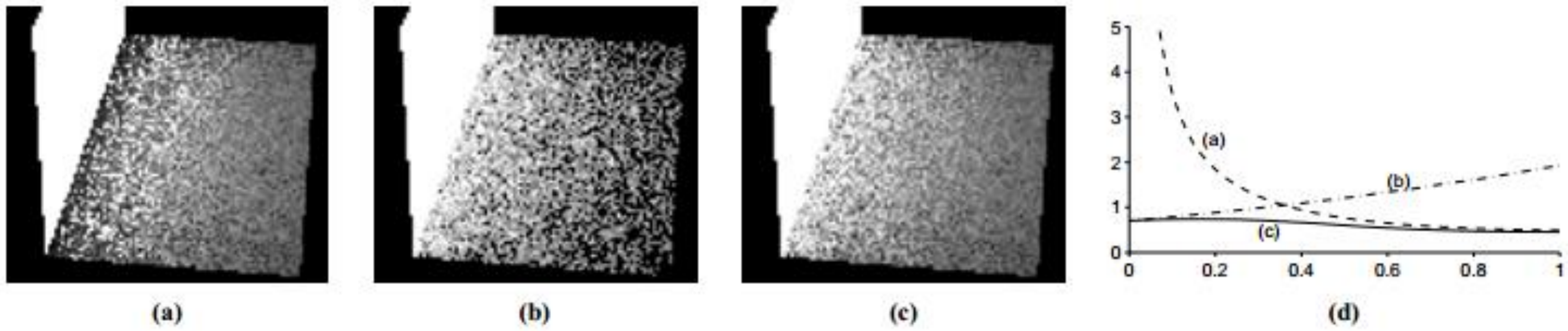
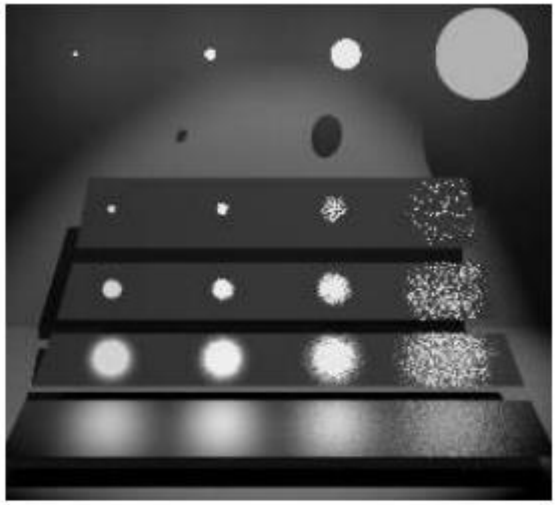
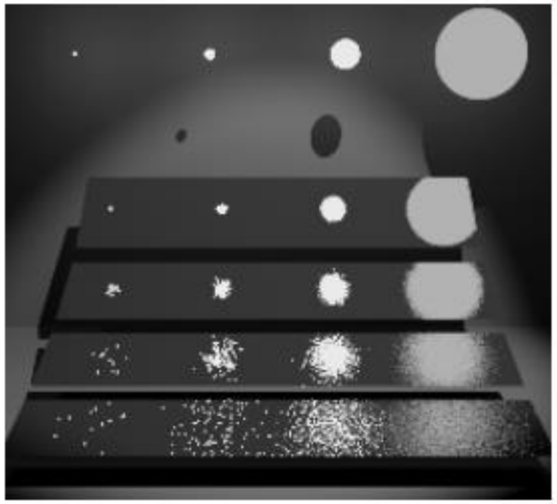
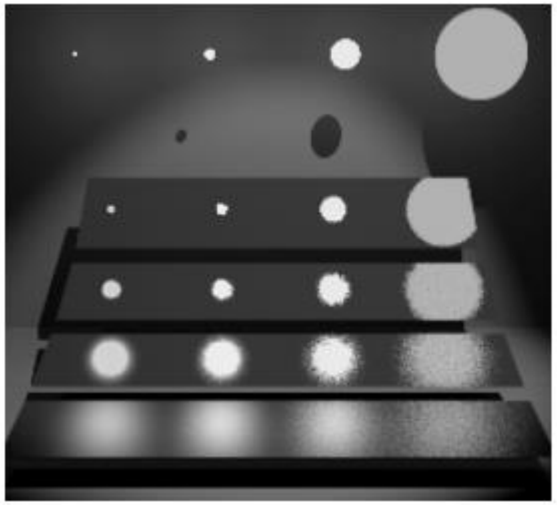


Figure 6: A simple test scene consisting of one area light source (i.e. a bright patch) and an adjacent diffuse surface. The images were computed by **(a)** sampling the light source according to emitted power, with $n_1 = 3$ samples per pixel, **(b)** sampling the hemisphere according to the *projected solid angle*[3] $\cos(\theta_o')\, d\sigma(\vec{\omega}_o')$ (see Fig. 1), with $n_2 = 6$ samples per pixel, and **(c)** a weighted combination of samples from (a) and (b) using the power heuristic with $\beta = 2$. **(d)** A plot of σ / μ (standard deviation divided by mean) as a function of distance from the light source, for $n_1 = 1$ and $n_2 = 2$.

Multiple Importance Sampling


Light sampling

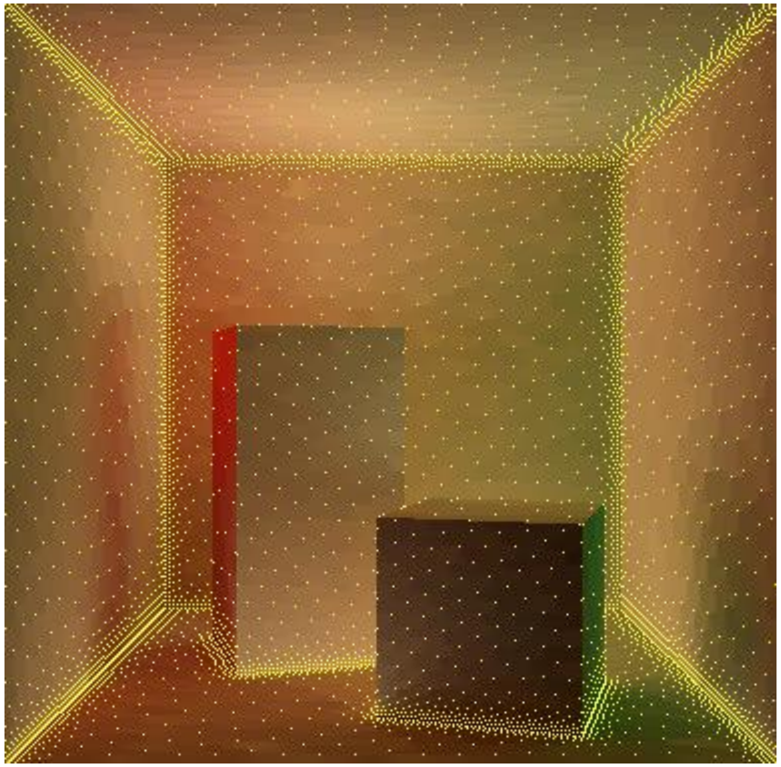
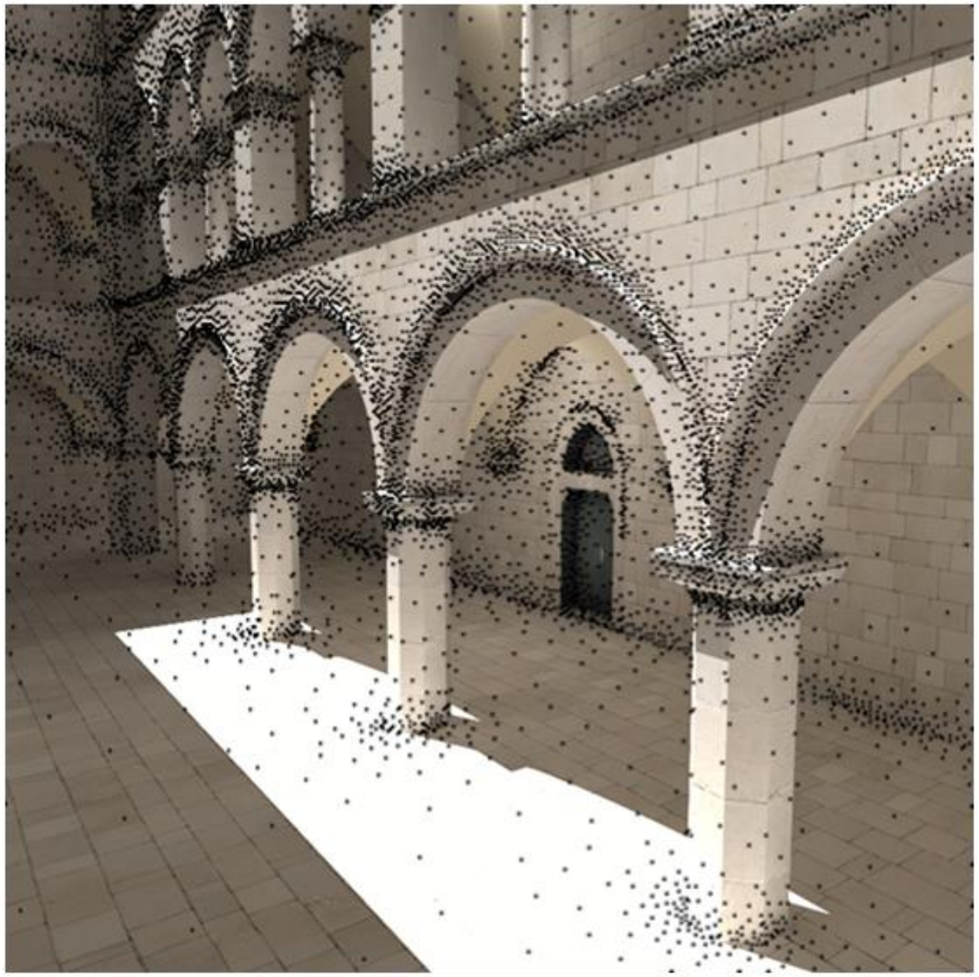

BRDF sampling


Multiple importance sampling

Irradiance Caching

- The direct illumination (LDE) in a scene typically has sharp discontinuities from shadows and angled surfaces
- The indirect illumination (LDD+E), however, is typically much smoother
- The idea behind irradiance caching is to compute and cache accurate indirect illumination at key points and use those cached values to interpolate
- There are many details in interpolating cached values and determining when to re-use nearby cache points and when to create new ones
- Irradiance caching was introduced by Greg Ward in 1988 and there have been many extensions and enhancements since then

Irradiance Caching

Limitations of Path Tracing

- Poorly suited for highly specular scenes and caustics
- Not well suited for highly indirect light
- Can be slow and take many paths to converge

Strengths of Path Tracing

- Well suited for primarily diffuse scenes with area lighting
- Relatively easy to implement
- Flexible: easy foundation to add new features
- Reference to compare other algorithms
- Becomes more and more appealing as GPUs get faster