INFOGR – Computer Graphics

J. Bikker - April-July 2016 - Lecture 13: “Ground Truth”

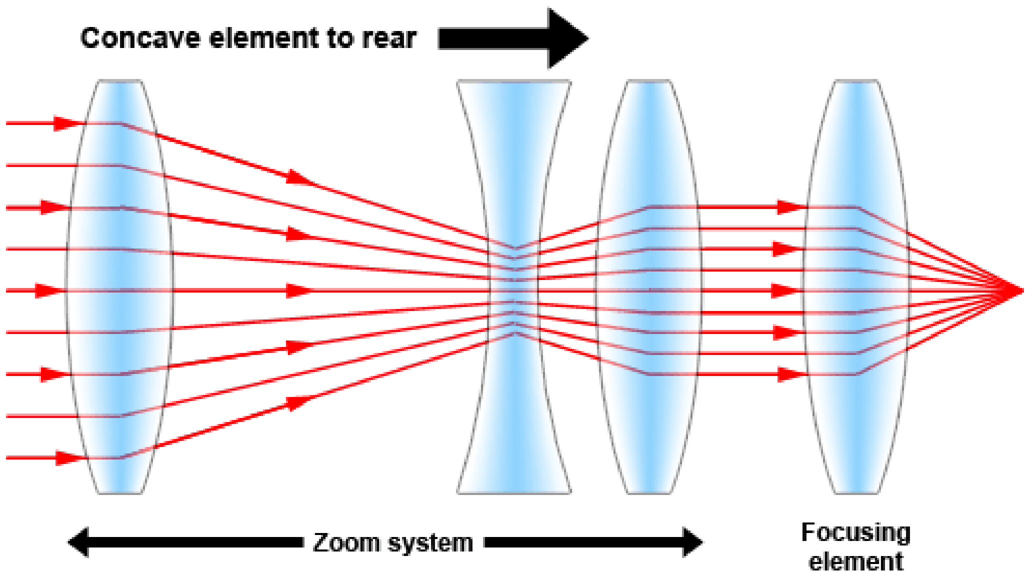
Welcome!

Today's Agenda:

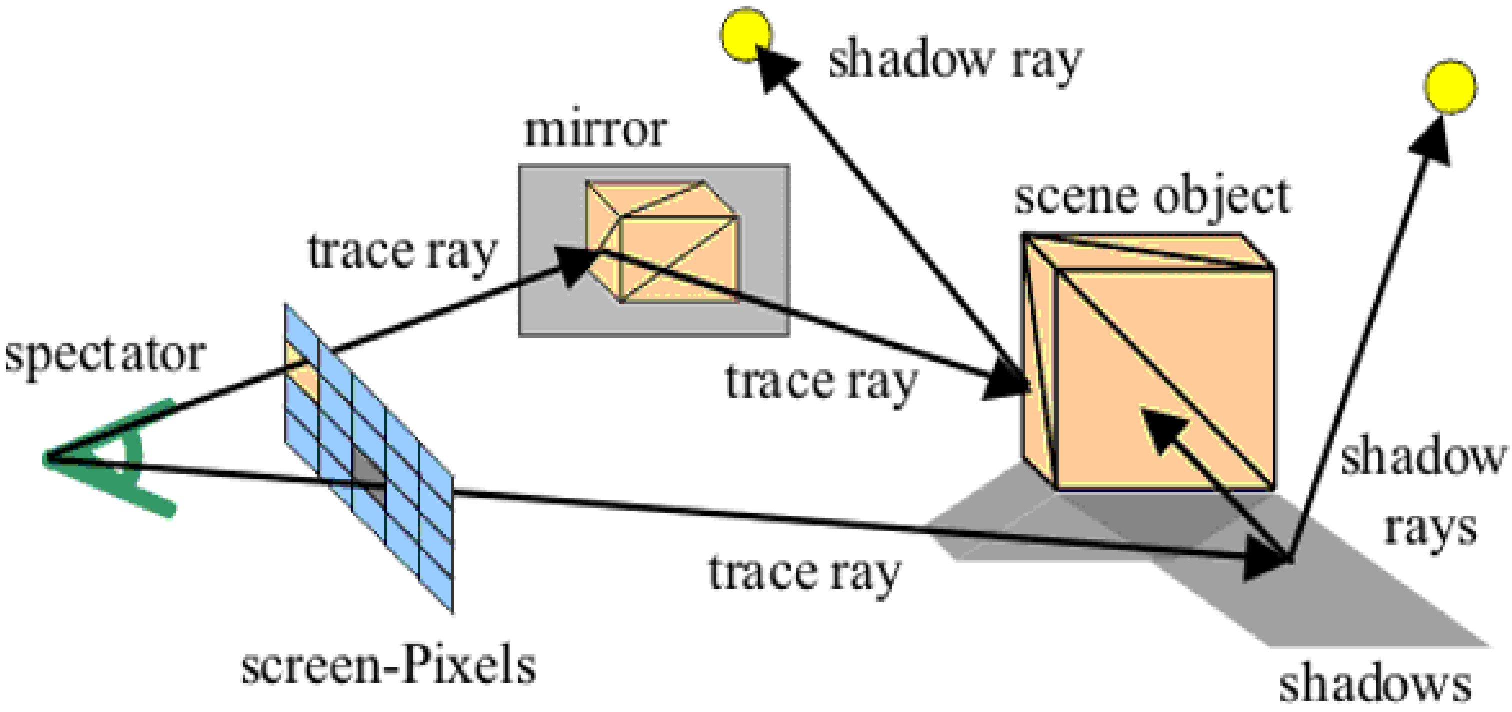
- Deterministic Rendering
- Monte Carlo
- Path Tracing

Deterministic

Whitted-style Ray Tracing



Zoom lens at wide-angle setting



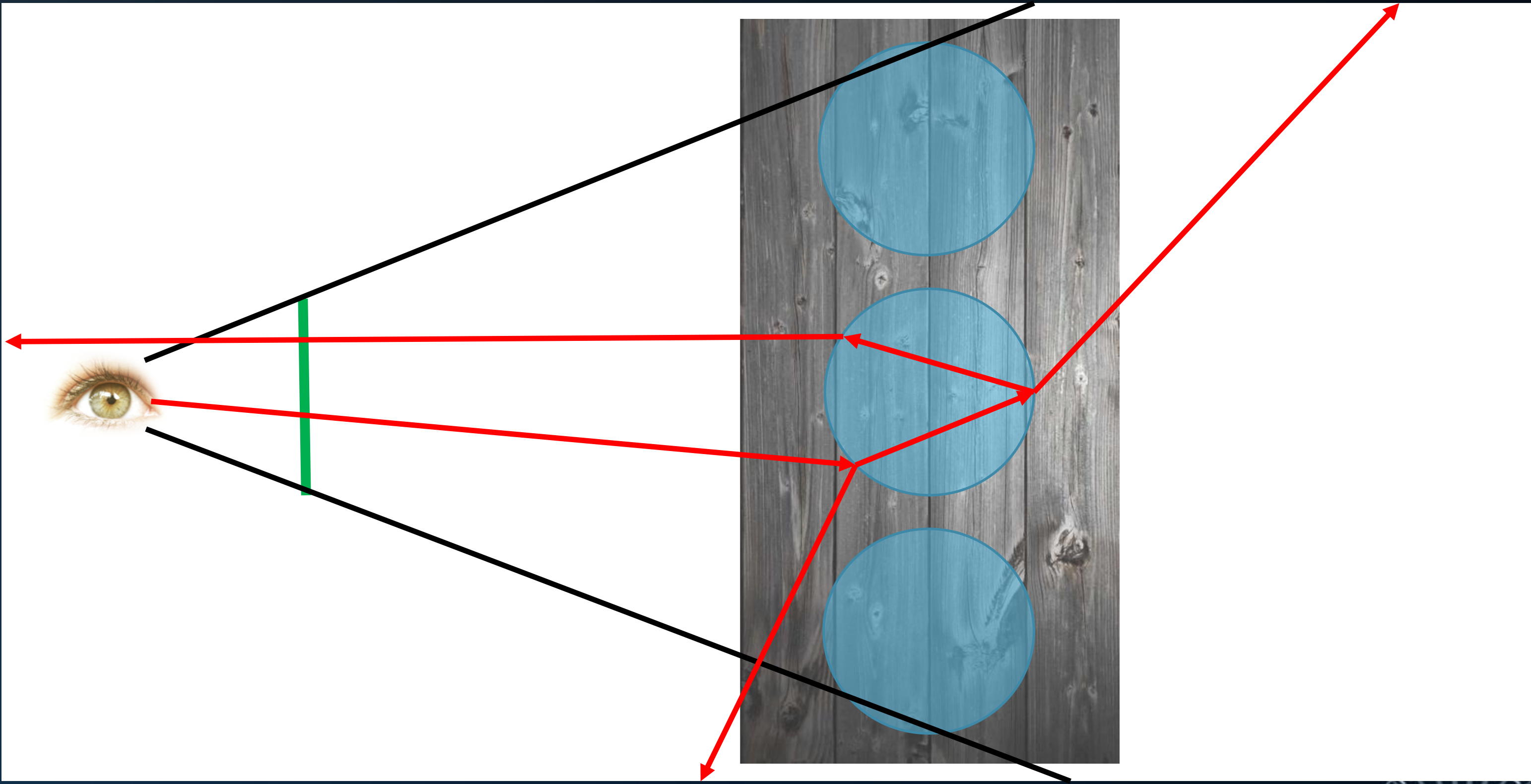
Deterministic

Whitted-style Ray Tracing

Color at pixel:

- sphere material color * refracted ray
- + sphere material color * reflected ray

This is a recursive process.



Deterministic

Whitted-style Ray Tracing

Color at pixel:

- sphere material color * refracted ray
- + sphere material color * reflected ray

This is a recursive process.

```
color Trace( O, D )
   I, N, mat = NearestIntersection( O, D )
   if (mat == DIFFUSE)
      return mat.color *
             DirectIllumination( I, N )
   if (mat == MIRROR)
      return mat.color *
             Trace( I, reflect( D, N ) )
   if (mat == GLASS)
      return mat.color *
             (X * Trace( I, reflect( D, N ) ) +
             (1-X) * Trace( I, refract( D, N ) ) )
```

Fresnel equations

Snell’s law

angle of incidence = angle of reflection

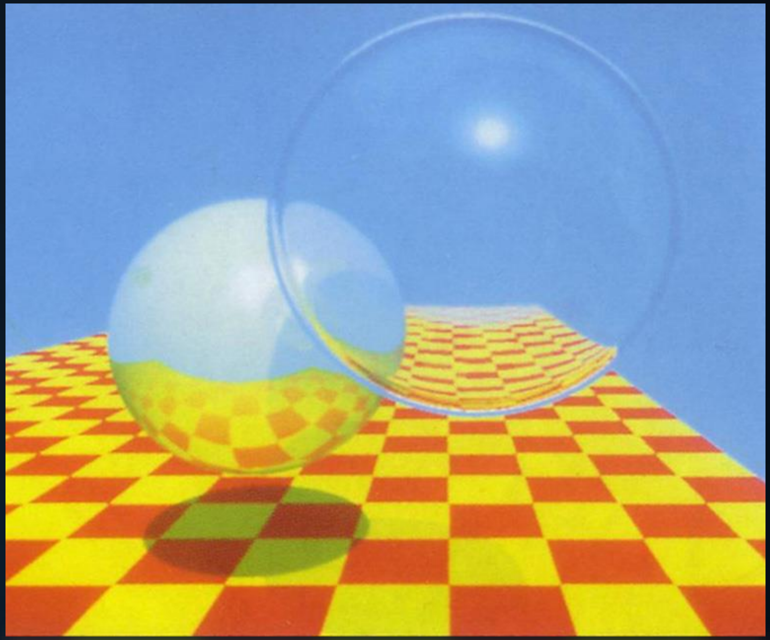
Deterministic

Whitted-style Ray Tracing

Shading model:

- Based on classical ray optics

Whitted-style ray tracing is deterministic: it cannot simulate area lights, glossy reflections, and diffuse reflections.



Deterministic

Rasterization

```
color Shade( P, N )
   if (mat == DIFFUSE)
      return mat.color *
             DirectIllumination( P, N )
   if (mat == MIRROR)
      return mat.color *
             EnvironmentMap( … )
   if (mat == GLASS)
      return mat.color *
             HackyGlassEffect( … )
```

Deterministic

	Rasterization	Ray Tracing
Lights	Point	Point
Reflection	Perfect specular	Perfect Specular
Indirect	Can’t do	Can’t do
Motion blur	Hacks	Tons of rays
Depth of field	Hacks	Tons of rays

Why are certain features hard or impossible?
Because they require *integration*.

Today's Agenda:

- Deterministic Rendering
- Monte Carlo
- Path Tracing

Monte-Carlo

Distributed Ray Tracing*

Problem:

Ray tracing is currently limited to sharp shadows, sharp reflections, and sharp refraction.

Goal:

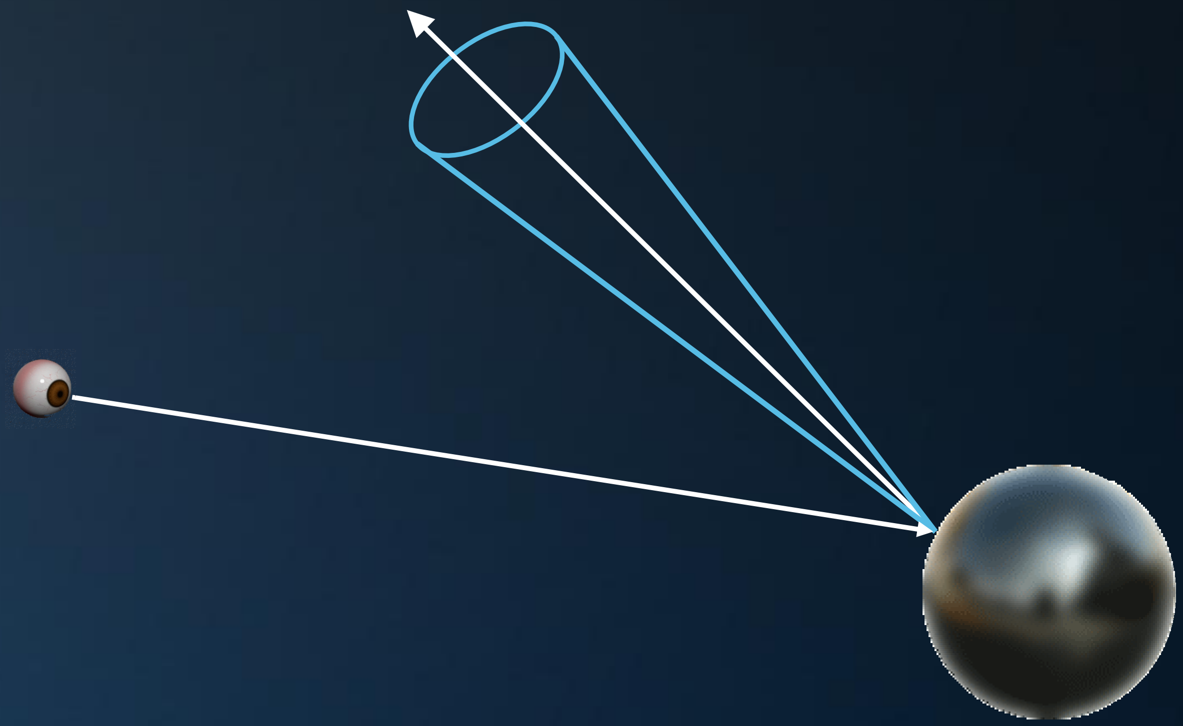
- Augment Whitted-style ray tracing with glossy reflections and refractions, as well as soft shadows.

*: “Distributed Ray Tracing”, Cook et al., 1984.


Photoshop

Monte Carlo

Monte Carlo



Monte Carlo

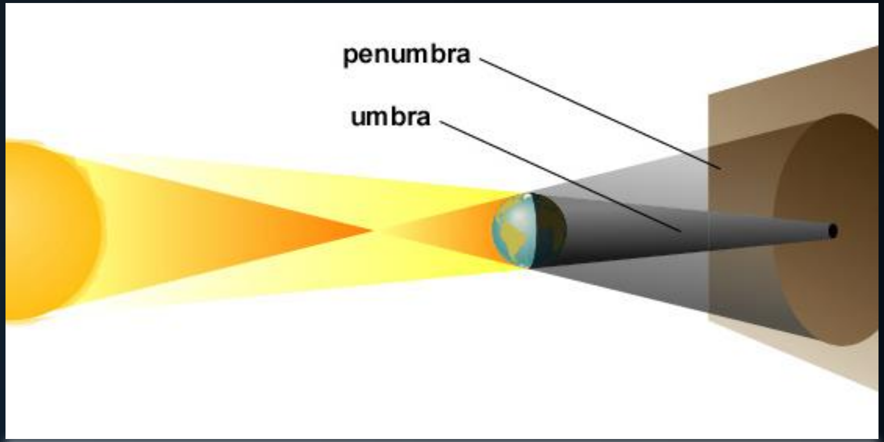
Analytic Soft Shadows

Anatomy of a shadow – regions

- Fully occluded area: *umbra*
- Partially occluded area: *penumbra*

A soft shadow requires an area light source.

In nature, all light sources are area lights (although some approximate point lights).



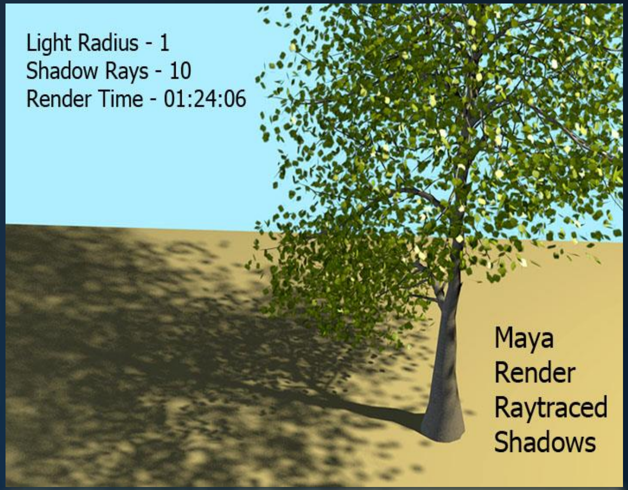
Monte Carlo

Analytic Soft Shadows

Surface points in the penumbra are lit by a part of the light source.

Rendering soft shadows requires that we determine the visible portion of the light source.

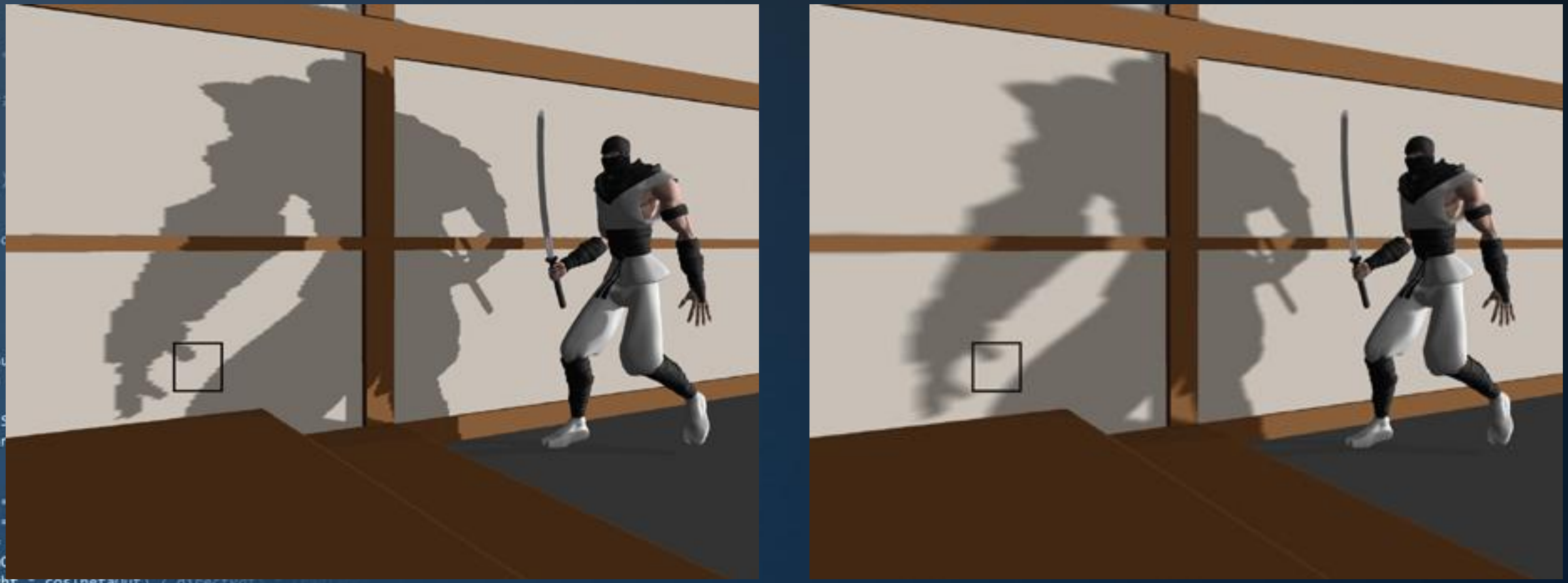
In most cases, this is a very hard problem.



Monte Carlo

Approximate Soft Shadows

When using shadow mapping, we can simulate soft shadows by blurring the shadow map.



In this example, filter kernel radius is adjusted based on the distance from the occluder.

Monte Carlo

Calculating Accurate Soft Shadows

"Rendering soft shadows requires that we determine the visible portion of the light source."

In other words:

The amount of light cast on a surface point P by area light L is determined by the integral of the visibility between P and L over the surface of the light source:

$$I_{L\rightarrow P} = \int_{A_L} V(P, L)$$

Monte-Carlo Integration

To solve this integral for the generic case, we will use Monte-Carlo integration.

Using Monte-Carlo, we replace the integral by the expected value of a stochastic experiment.

Monte Carlo

Stochastic shadows

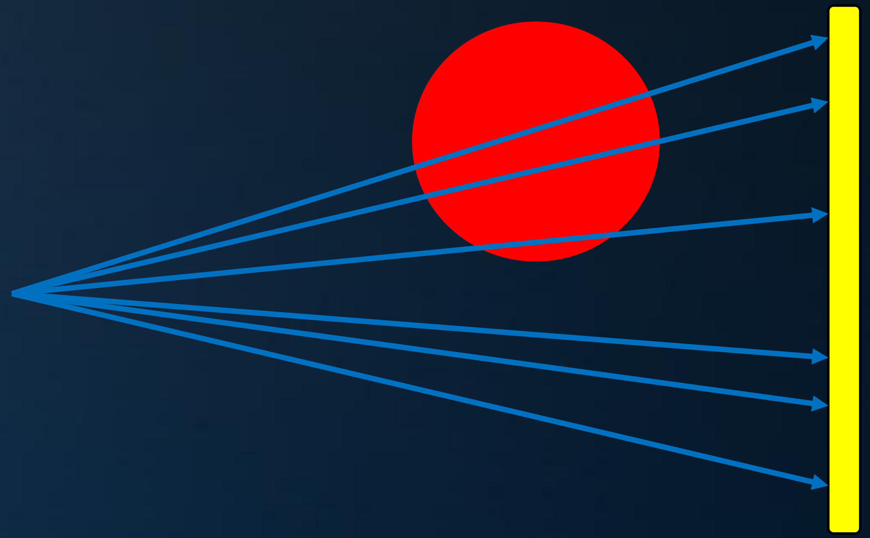
For soft shadows, we want to know the visible area of a light source, which can be 0..100%.

The light source could be (partially) obscured by any number of objects.

We can approximate the visibility of the light source using a number of random rays.

Using 6 rays:

$$V \approx \frac{1}{6}\sum_{i=1}^{6} V_i$$



Monte Carlo

Stochastic shadows

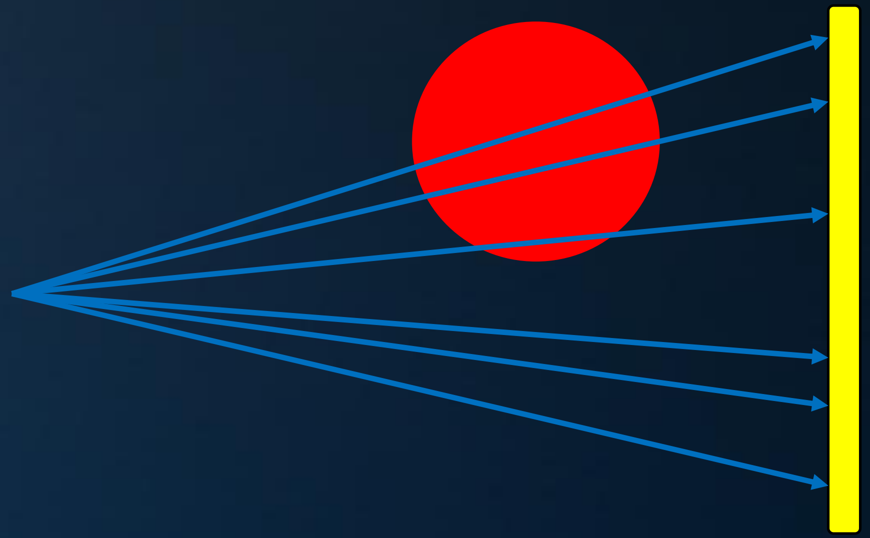
For soft shadows, we want to know the visible area of a light source, which can be 0..100%.

The light source could be (partially) obscured by any number of objects.

We can approximate the visibility of the light source using a number of ***random*** *rays.*

Using N rays:

$$V \approx \frac{1}{N} \sum_{i=1}^{N} V_i$$



Monte Carlo

Stochastic shadows

$$V \approx \frac{1}{N}\sum_{i=1}^{N} V_i$$

As N approaches infinity, the result becomes equal to the expected value, which is the integral we were looking for.

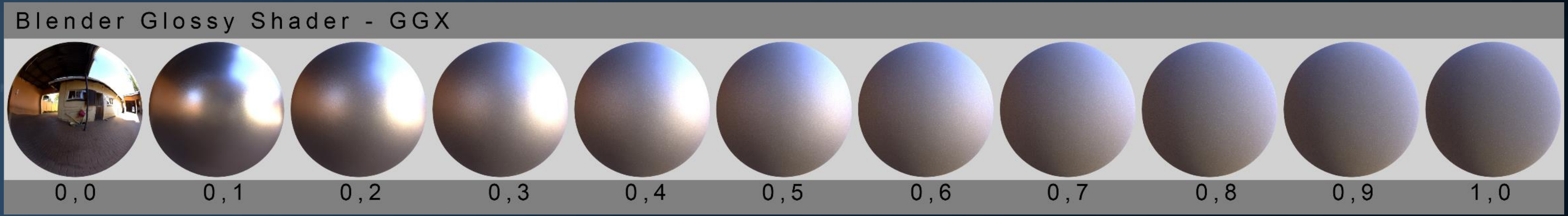
Before that, the result will exhibit *variance*.
In the case of soft shadows, this shows up as noise.

Monte Carlo

Approximate Diffuse Reflections

When rendering diffuse reflections, we face a similar problem:

A glossy surface reflects light arriving from a range of directions.



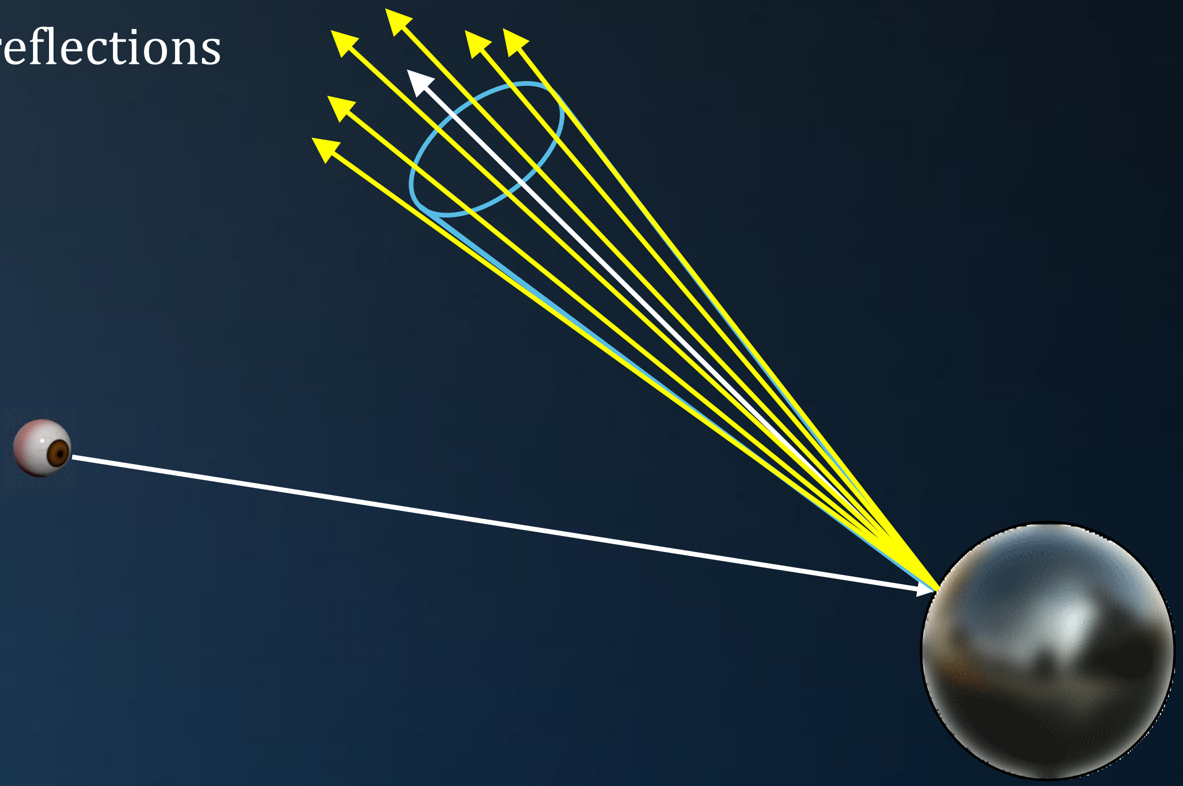
In rasterization, we can achieve this by blurring the environment map.

Monte Carlo

Note that a correct glossy reflection requires a filter kernel size based on distance to the reflected object.

Monte Carlo

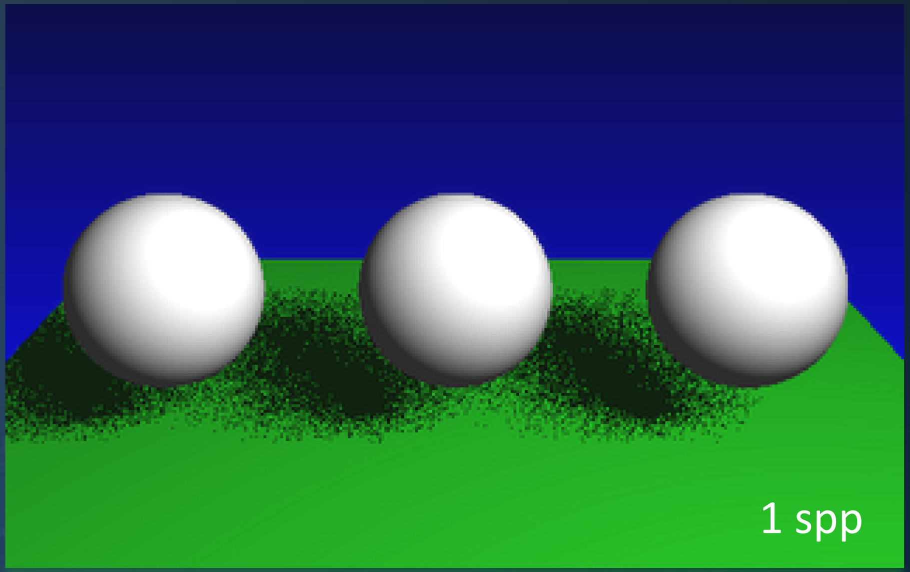
Stochastic reflections

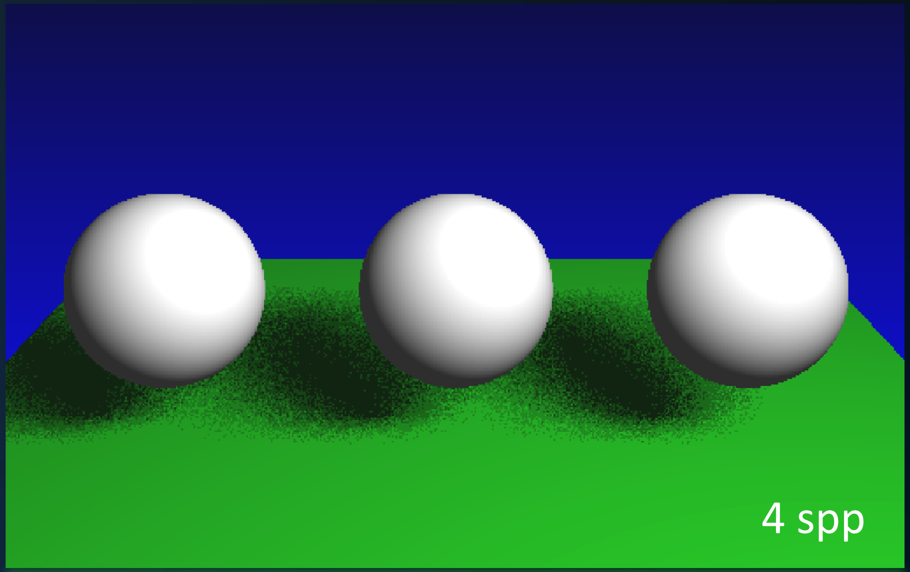


Monte Carlo

Variance

As long as we don’t take an infinite amount of samples, the result of the stochastic process exhibits variance.

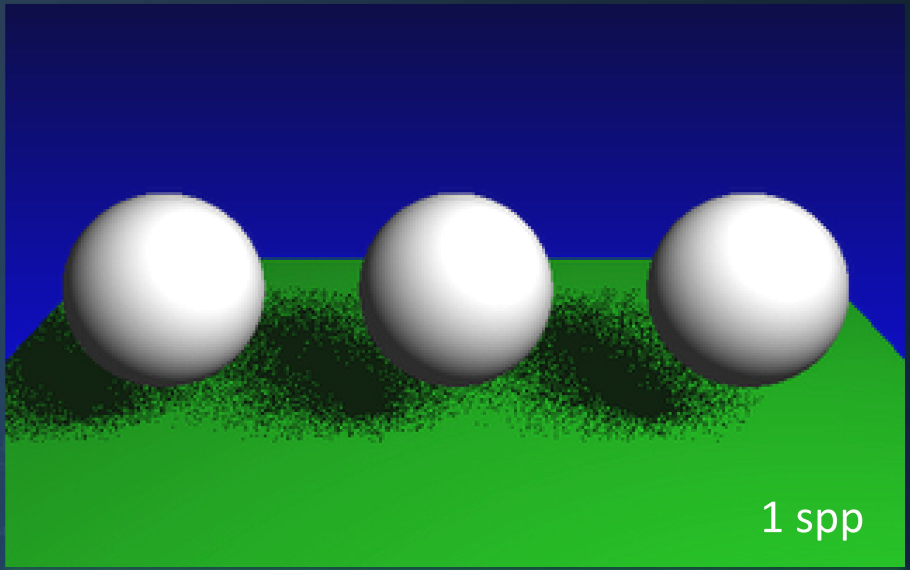


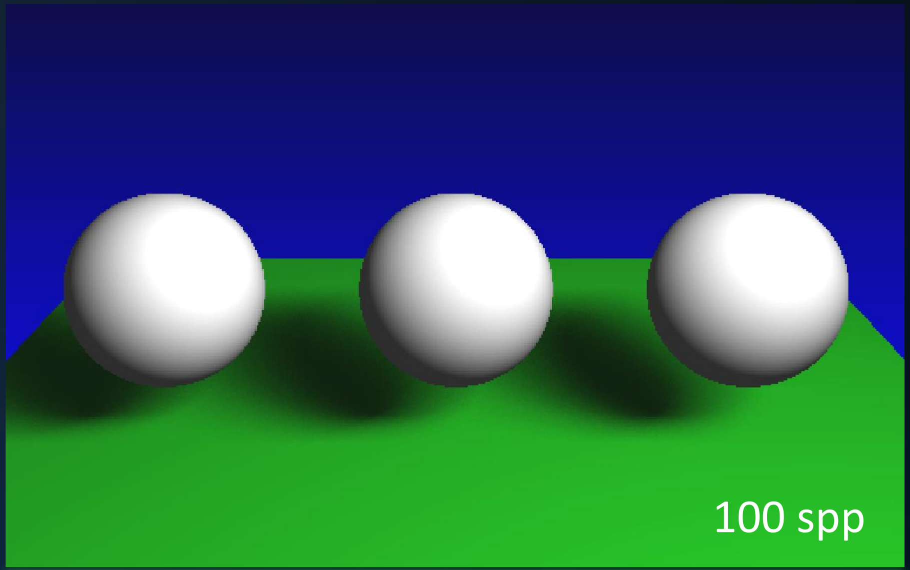


Monte Carlo

Variance

As long as we don't take an infinite amount of samples, the result of the stochastic process exhibits variance.



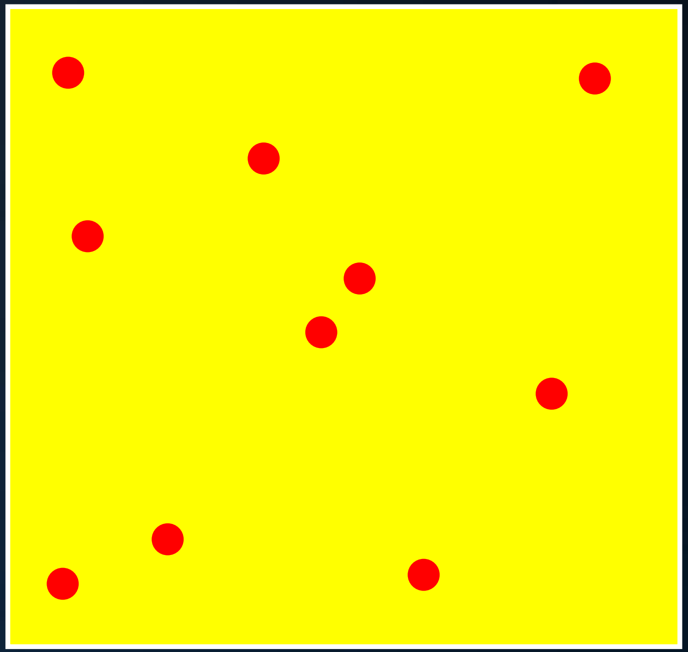


Monte Carlo

Variance reduction: stratification

The variance in random sampling can be reduced using *stratification*.

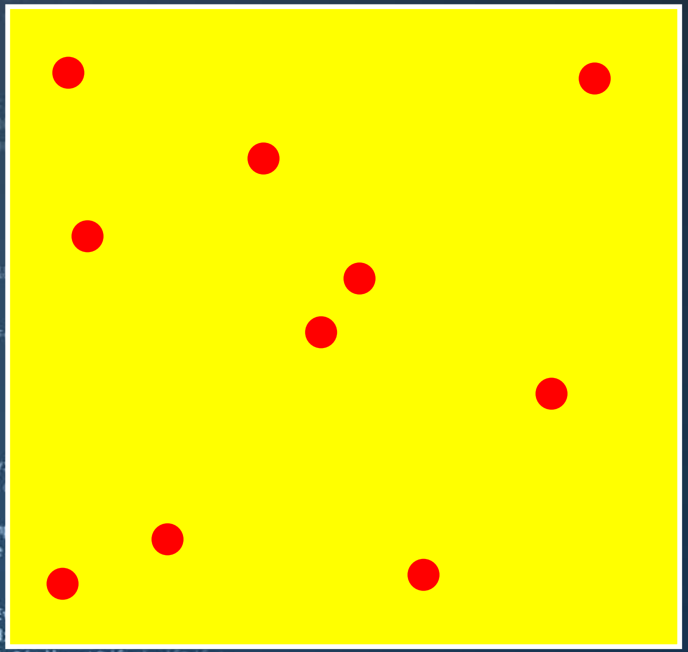
N=16

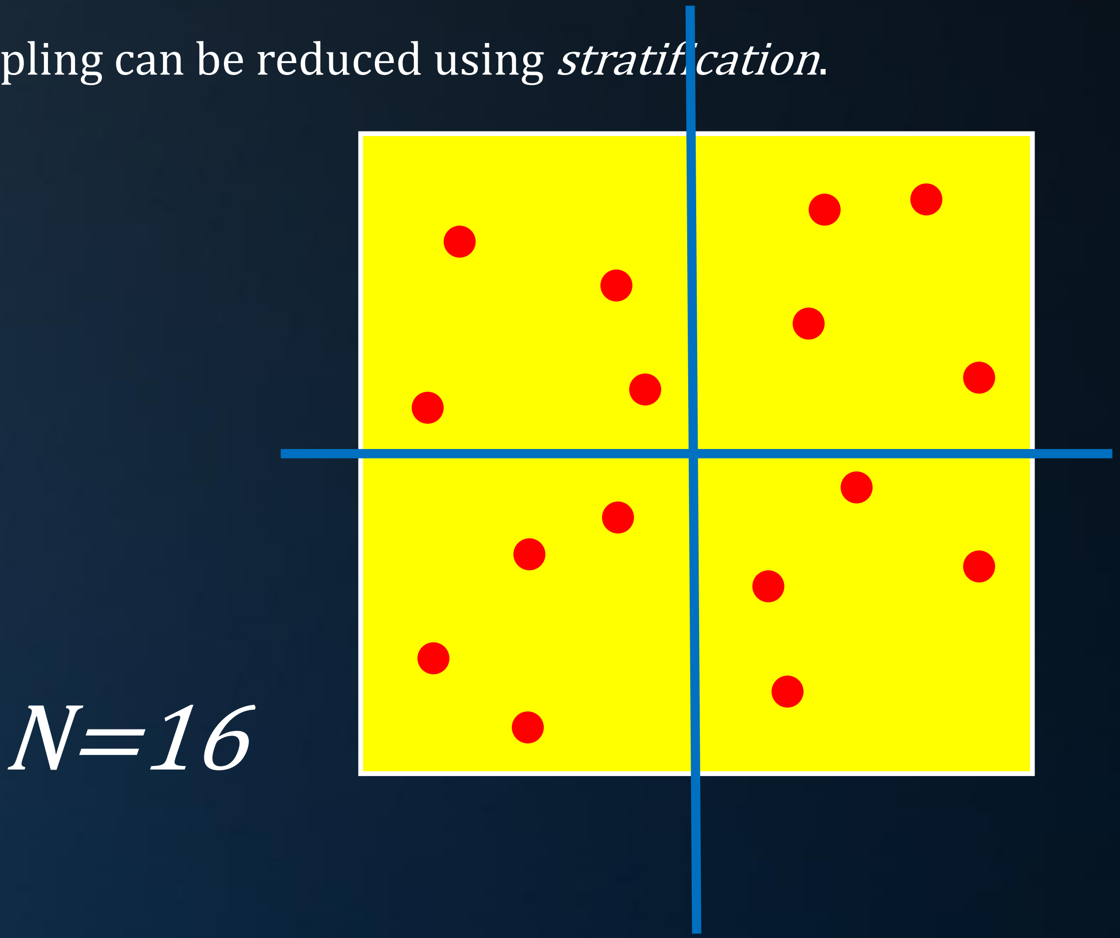


Monte Carlo

Variance reduction: stratification

The variance in random sampling can be reduced using *stratification*.

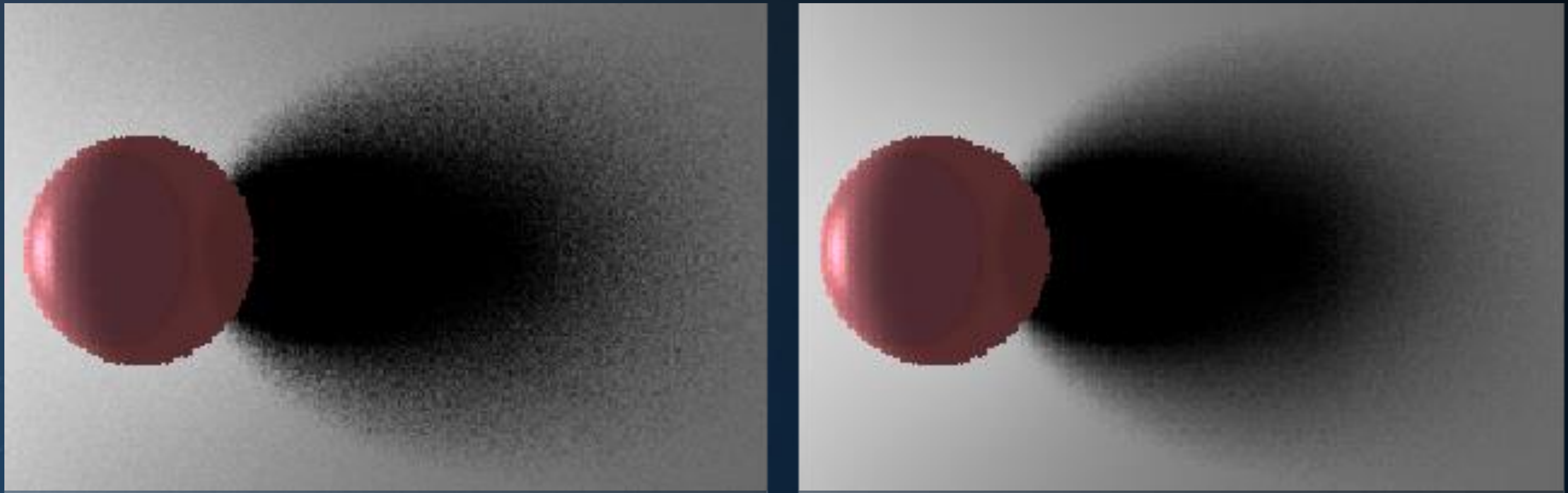




Monte Carlo

Variance reduction: stratification

The variance in random sampling can be reduced using *stratification*.



Uniform vs stratified, 36 samples, 6x6 strata

Monte-Carlo

Distributed Ray Tracing

Integrating over area of light sources: soft shadows

Integrating over reflection cone: glossy reflections

Integrating over pixel: anti-aliasing

Integrating over time: motion blur

Integrating over lens: depth of field

Integrating over wavelength: dispersion

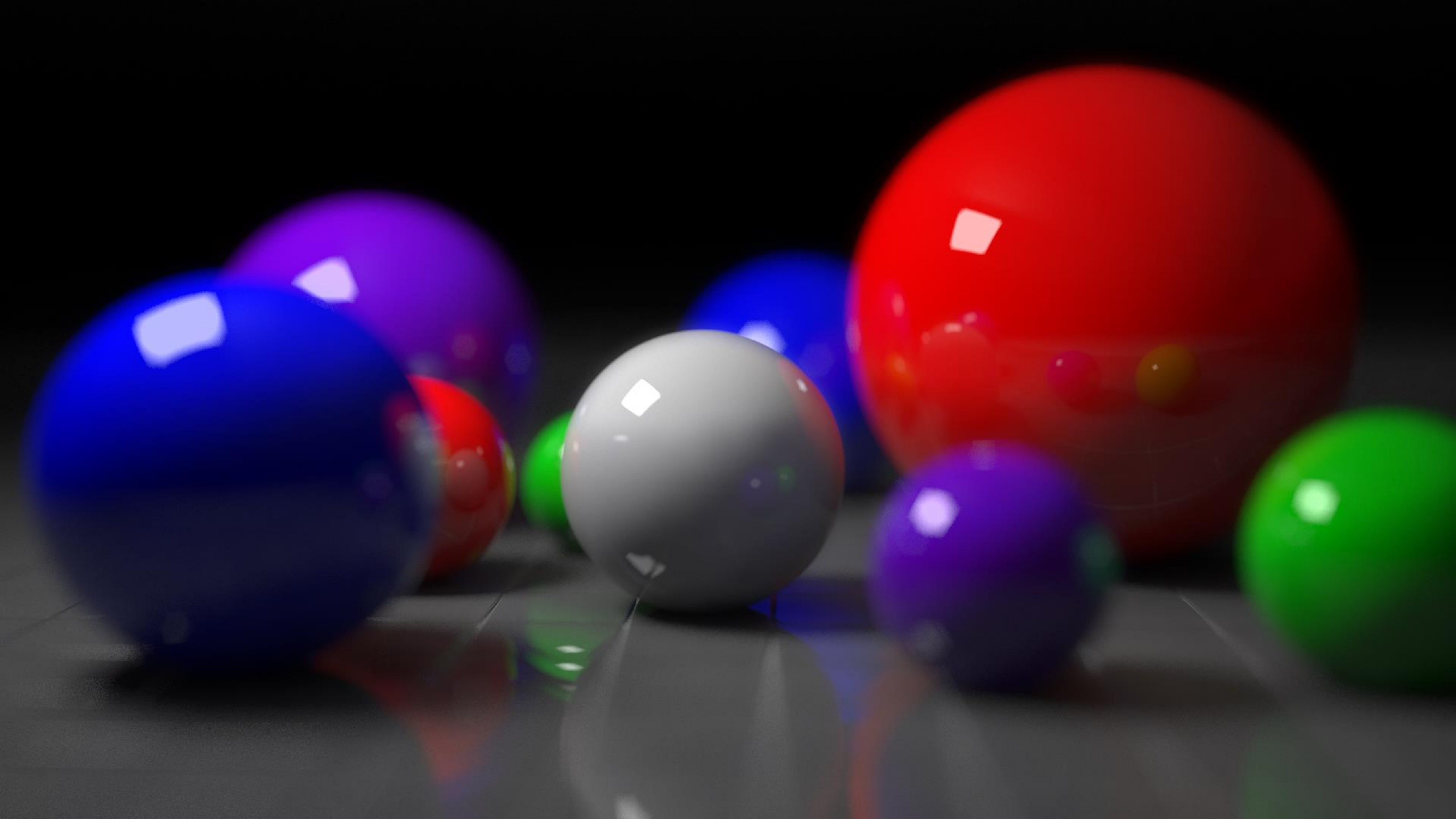
Monte Carlo

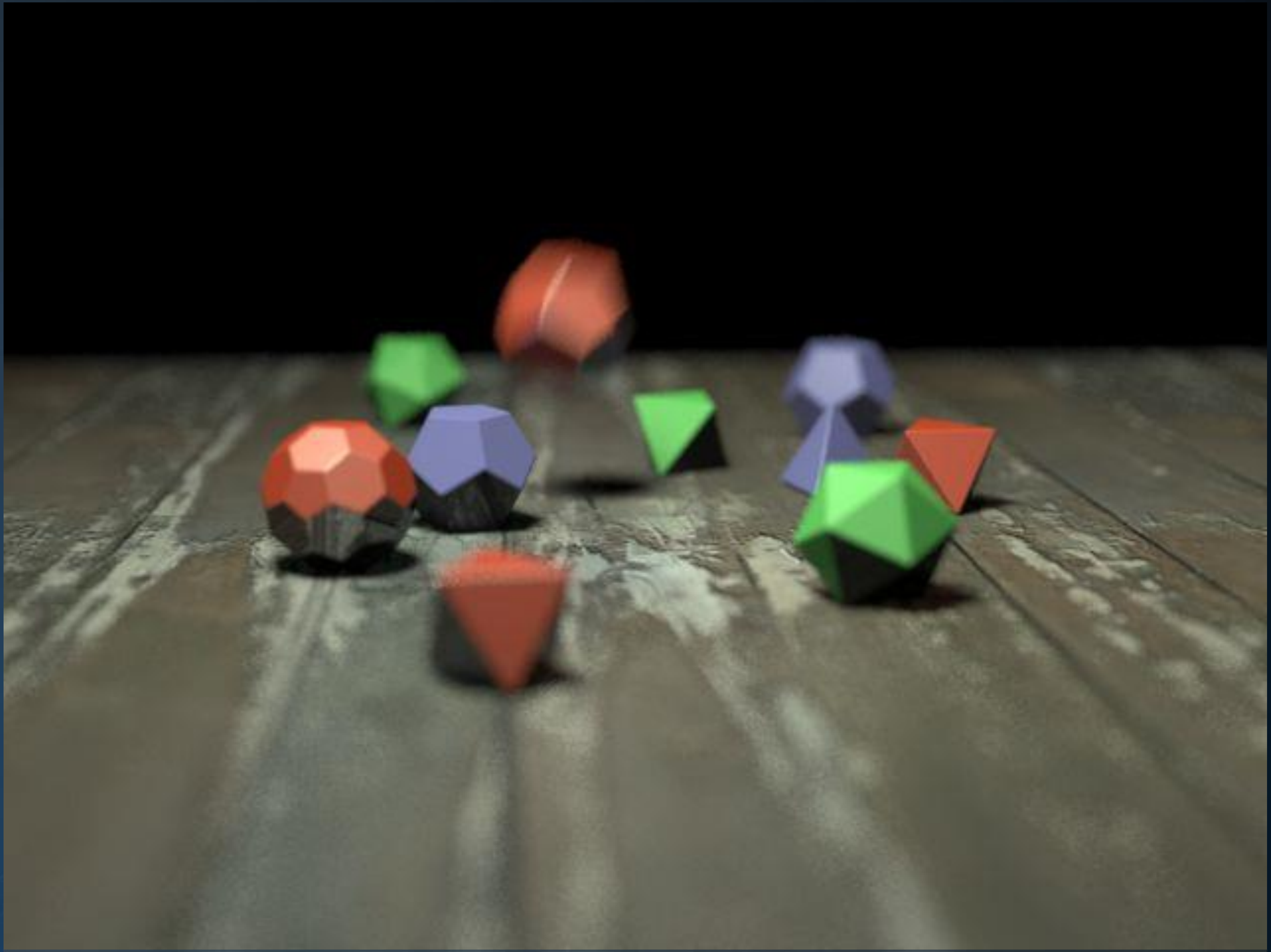
Distributed Ray Tracing

Improved model:

- Still based on classical ray optics
- Combined with probability theory to solve integrals

Distributed ray tracing requires many rays to bring down variance to acceptable levels.



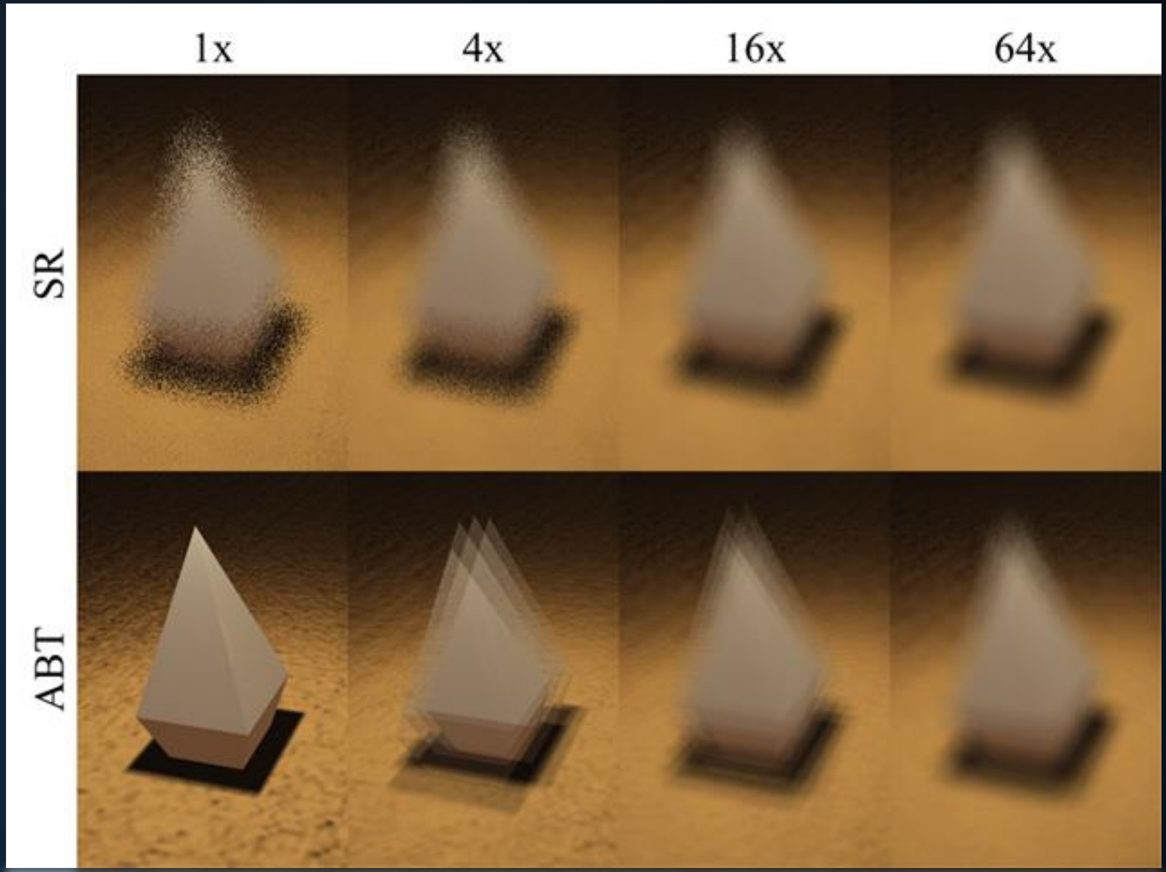


Monte Carlo

Monte Carlo in Rasterization

“Stochastic Depth of Field using Hardware Accelerated Rasterization”,

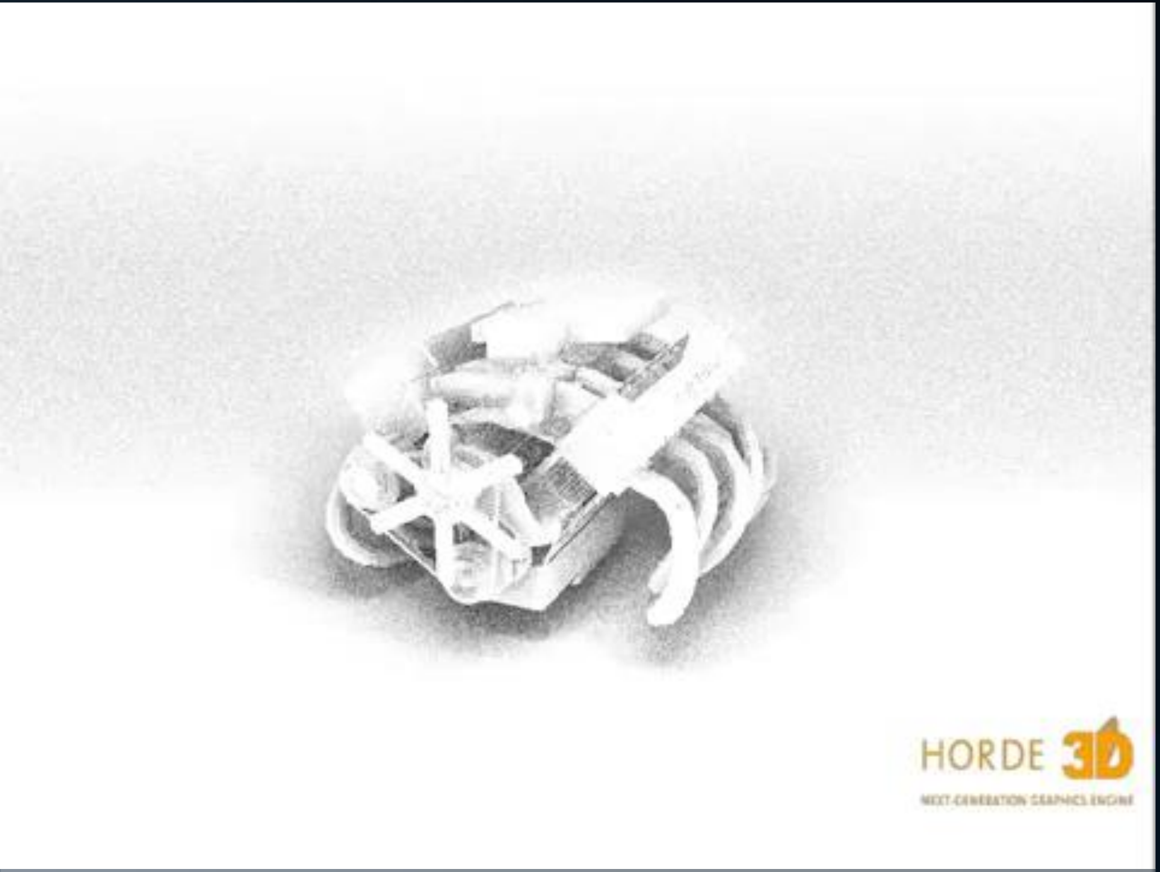
Robert Toth & Erik Lindler, 2008



Monte Carlo

Monte Carlo in Rasterization

Screen Space Ambient Occlusion,
CryEngine 2, 2007.



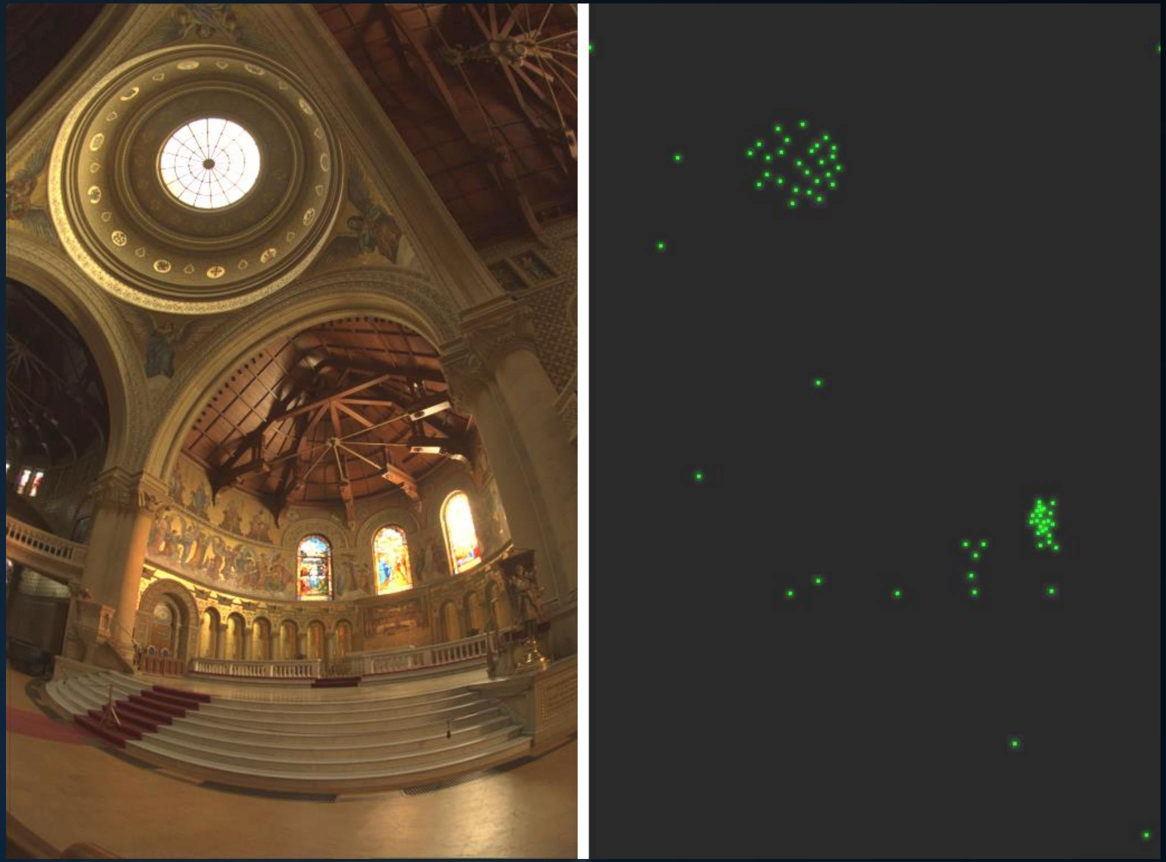
Monte Carlo

Monte Carlo in Rasterization

Light from an environment map, from:

"Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions",

Clarberg et al., 2005.



Monte Carlo

Cost of Distributed Ray Tracing

Distributed Ray Tracing is an expensive process:

- Per primary hit point, we need ~64 shadow rays *per light*
- Per primary hit point on a glossy surface, we need ~64 reflection rays,
  - ...and, for each reflection ray hit point, we need ~64 shadow rays per light.

If we use 4x4 anti-aliasing per pixel, multiply the above by 16.

Now imagine a glossy surface reflects another glossy surface...

Monte Carlo

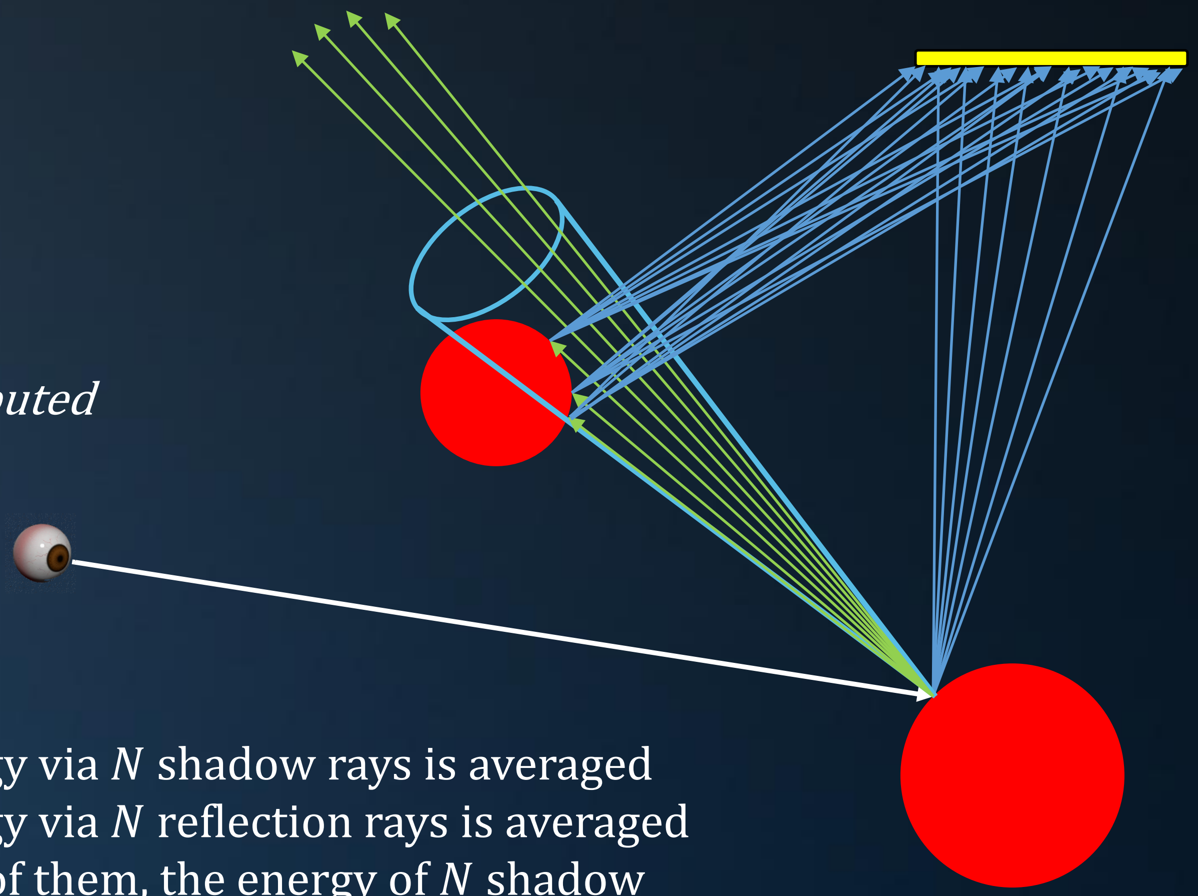
Today's Agenda:

- Deterministic Rendering
- Monte Carlo
- Path Tracing

Physically Based

Ray Tree

Using distributed ray tracing:



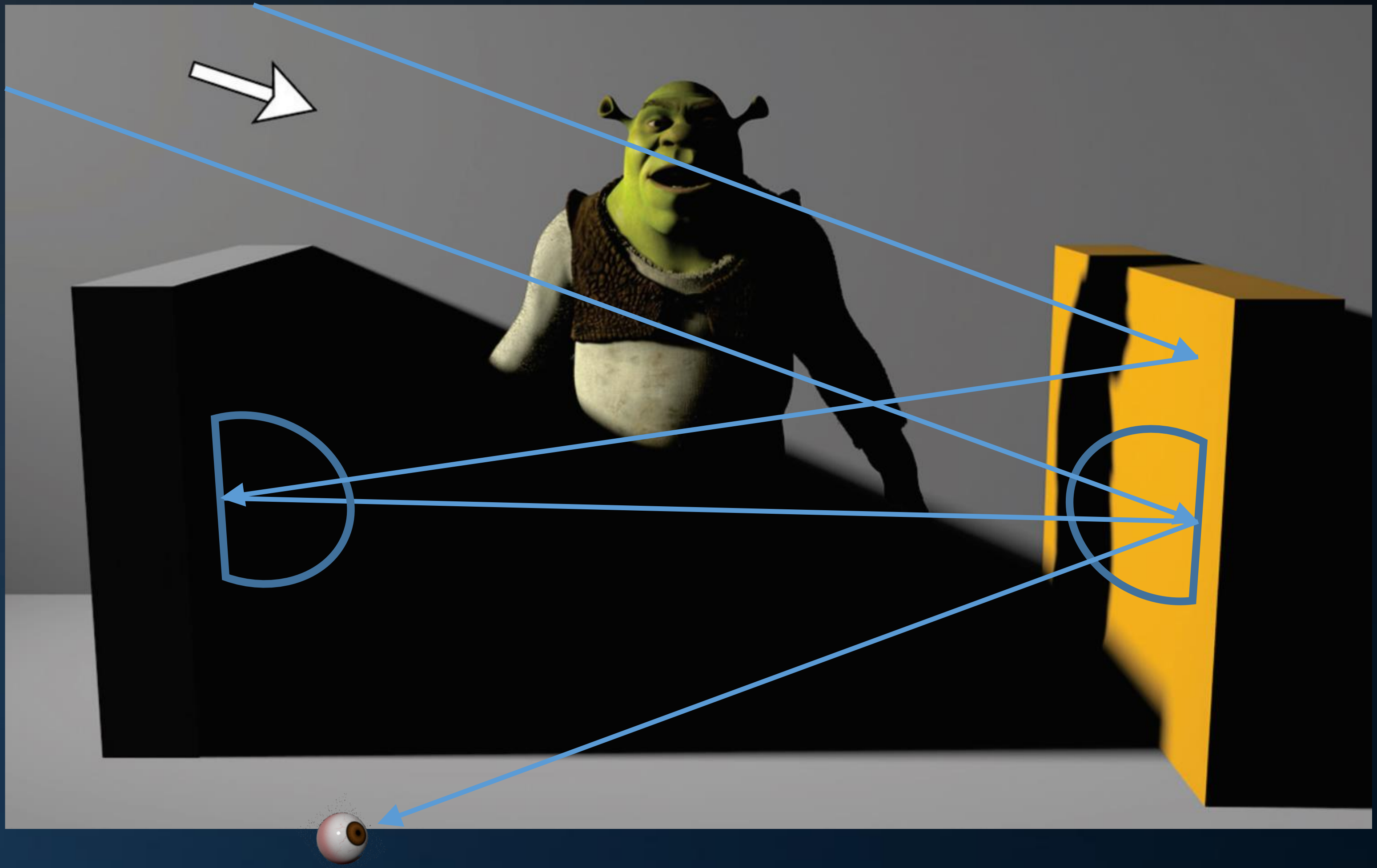
- The energy via *N* shadow rays is averaged
- The energy via *N* reflection rays is averaged
- For each of them, the energy of *N* shadow rays is averaged.

➔ *The energy via each shadow ray is very low.*

Physically Based

Diffuse reflections

Apart from specular and glossy materials, diffuse materials also reflect light.

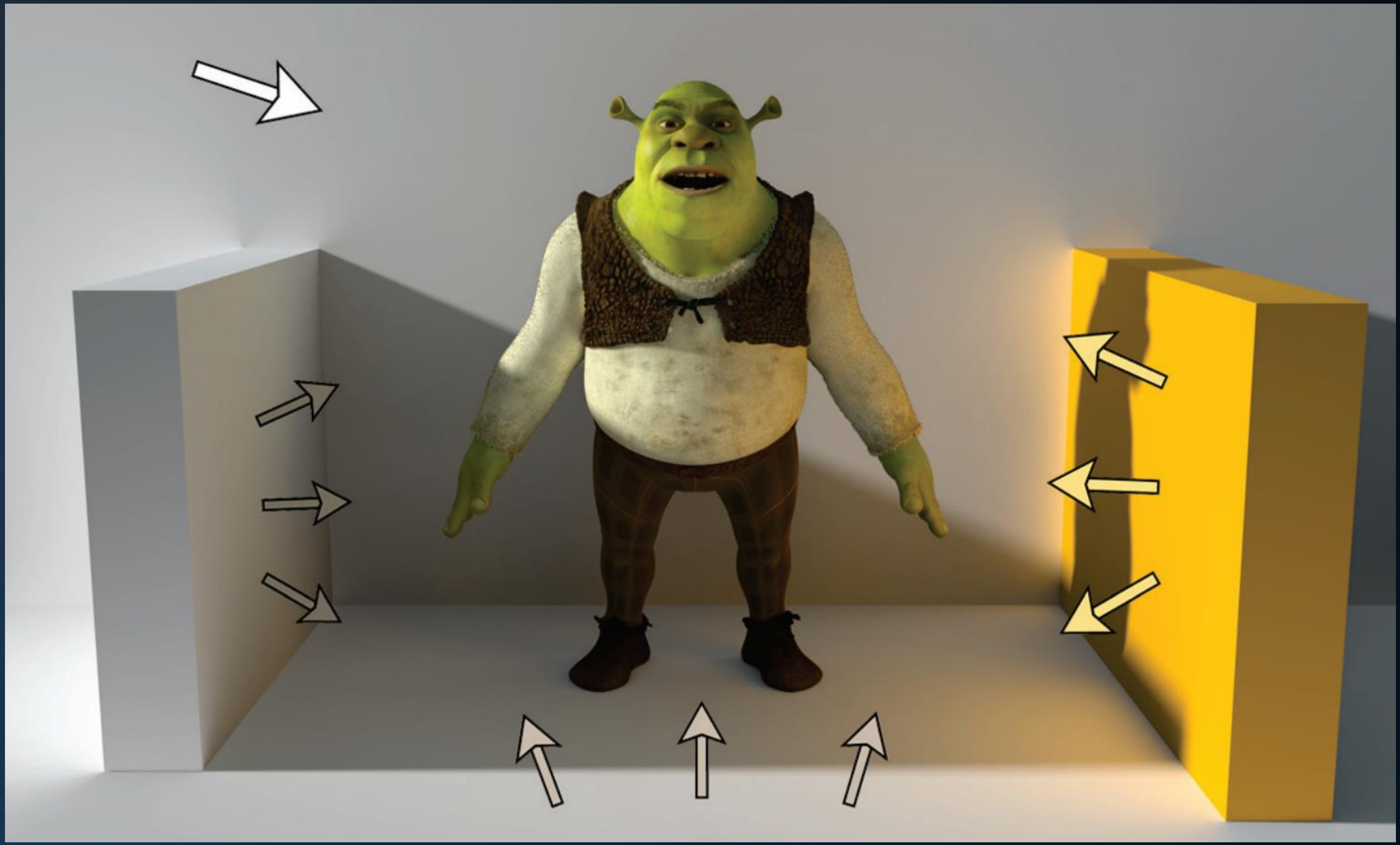


Physically Based

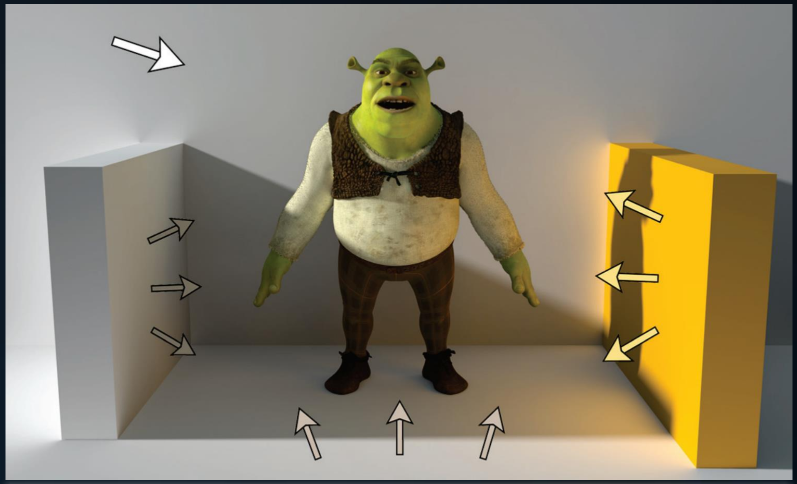
Diffuse reflections

Apart from specular and glossy materials, diffuse materials also reflect light.

This is why a shadow is seldom black.



Physically Based



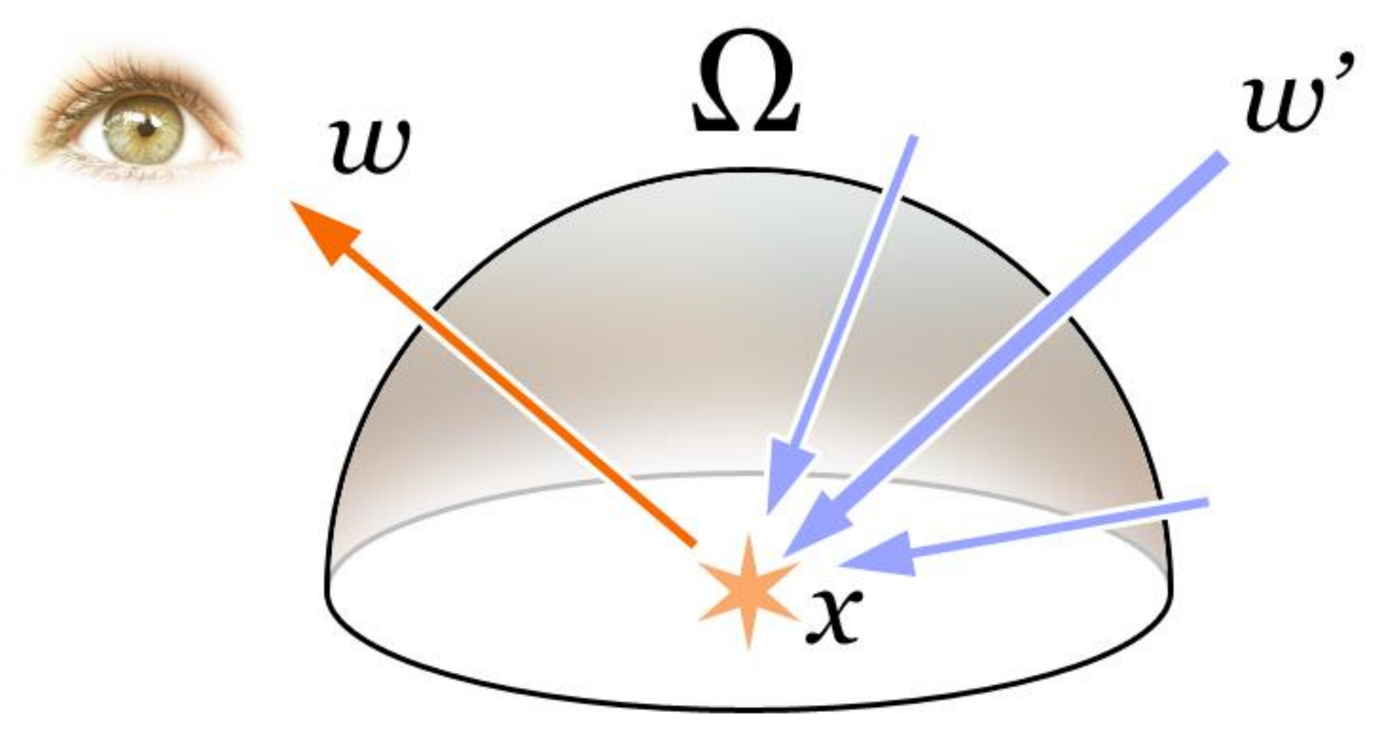
Physically based rendering

Calculating <u>all</u> light transport from the light sources to the camera, directly or via scene surfaces.

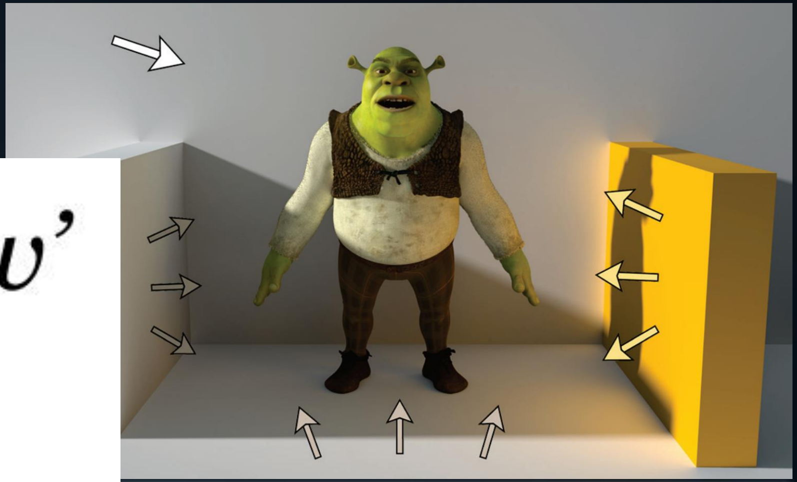
Nature solves this using a “random walk”: a large number of photons travelling through space from lights to sensors.

$$L_o(\mathbf{x},\mathbf{w}) = L_e(\mathbf{x},\mathbf{w}) + \int_{\Omega} f_r(\mathbf{x},\mathbf{w}',\mathbf{w}) L_i(\mathbf{x},\mathbf{w}')(-\mathbf{w}'\cdot\mathbf{n})d\mathbf{w}'$$

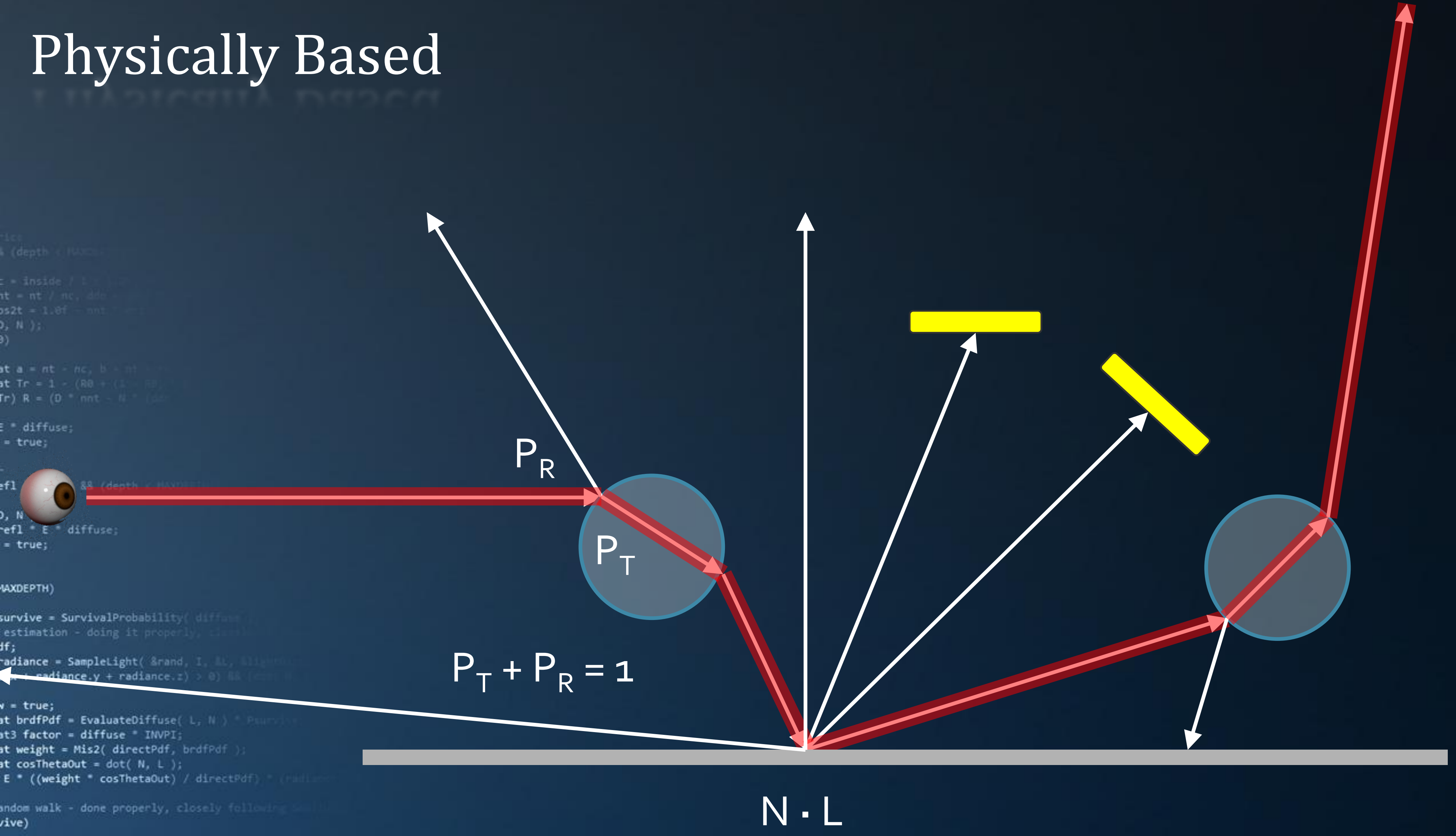
Physically Based



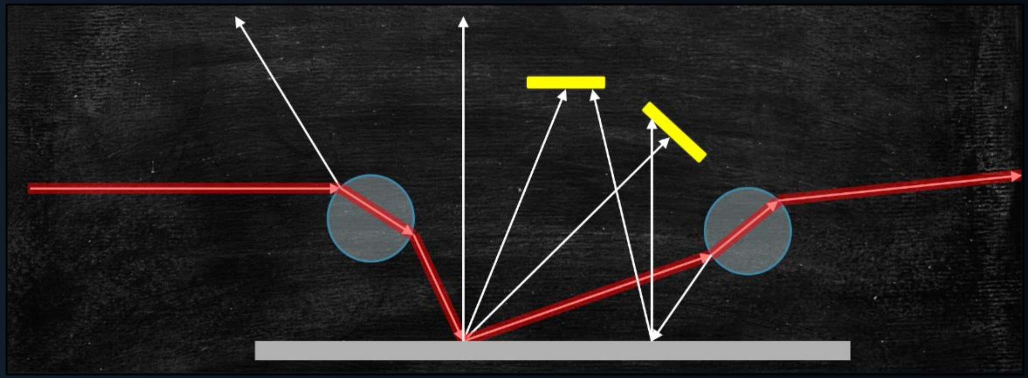
$$L_o(\mathbf{x},\mathbf{w}) = L_e(\mathbf{x},\mathbf{w}) + \int_{\Omega} f_r(\mathbf{x},\mathbf{w}',\mathbf{w}) L_i(\mathbf{x},\mathbf{w}')(-\mathbf{w}'\cdot\mathbf{n})d\mathbf{w}'$$



Physically Based



Physically Based

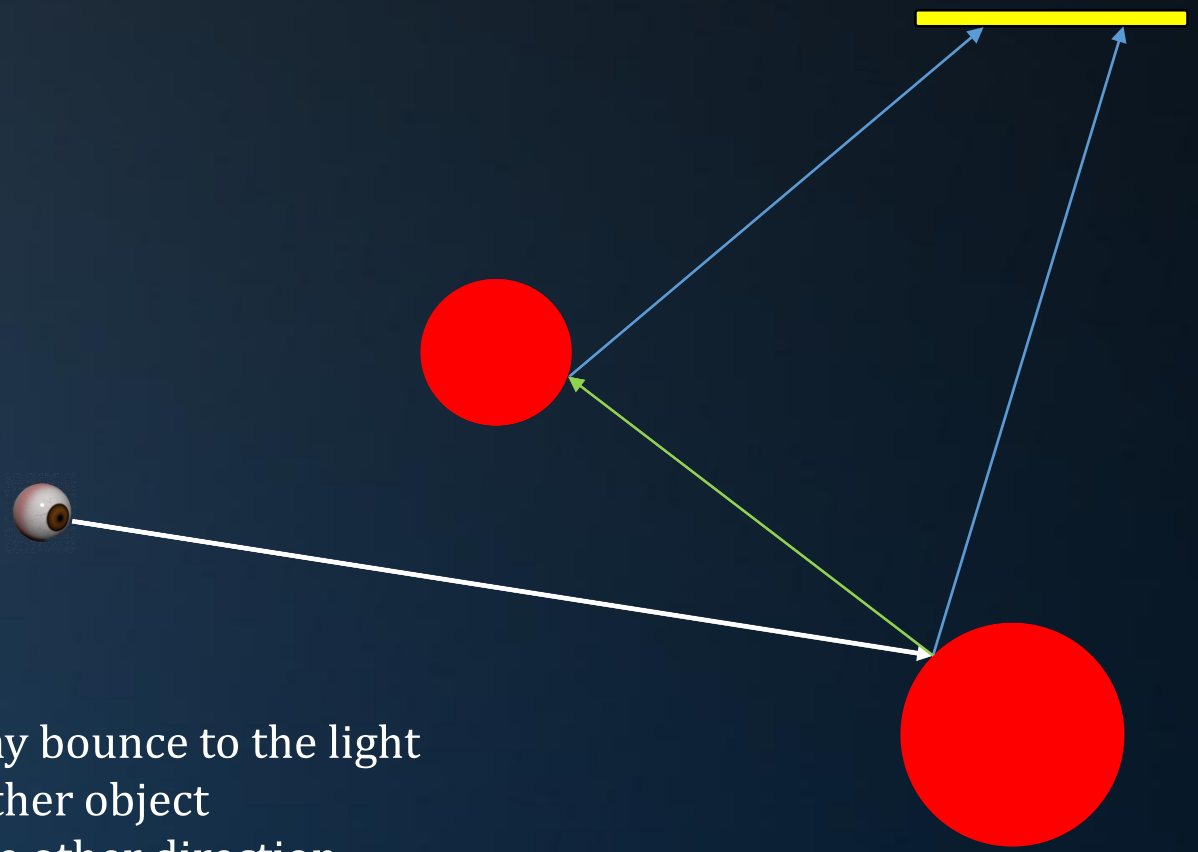


Path Tracing

```
Color Trace( vec3 O, vec3 D )
{
   I,N,mat = Intersect( O, D );
   if (mat.IsLight()) return mat.emissive;
   vec3 R = RandomReflection( N );
   BRDF = mat.color;
   return BRDF * dot( N, R ) * Trace( I, R );
}
```

Physically Based

Ray Tree



- A path may bounce to the light
- Or to another object
- Or in some other direction

Physically Based

Path Tracing

Tracing ‘photons’ backwards, from the camera to the light source, by performing a random walk.

- Instead of splitting the path, we randomly evaluate one branch.
- By using many paths, we explore all possible branches.
- We have the same number of primary rays as we have ‘shadow rays’.

Today’s Agenda:

- Deterministic Rendering
- Monte Carlo
- Path Tracing

INFOGR – Computer Graphics

J. Bikker - April-July 2016 - Lecture 13: “Ground Truth”

END of “Ground Truth”

next lecture: “Grand Recap”