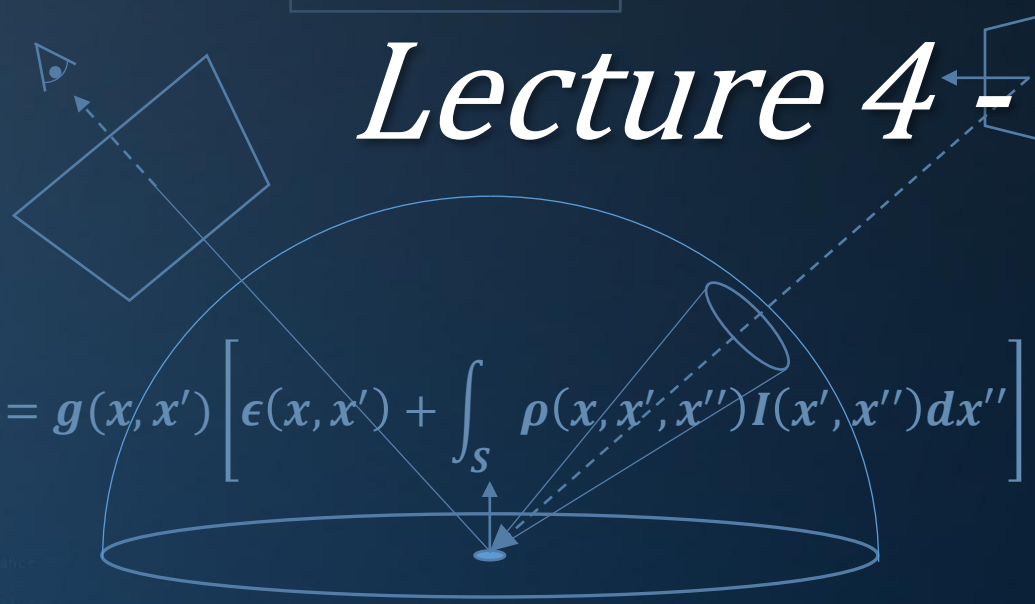
INFOMAGR – Advanced Graphics

Jacco Bikker - November 2022 - February 2023



Lecture 4 - "Path Tracing"

$$I(x,x') = g(x,x')\left[\epsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$$

Welcome!

Today’s Agenda:

- Introduction
- Path Tracing

Introduction

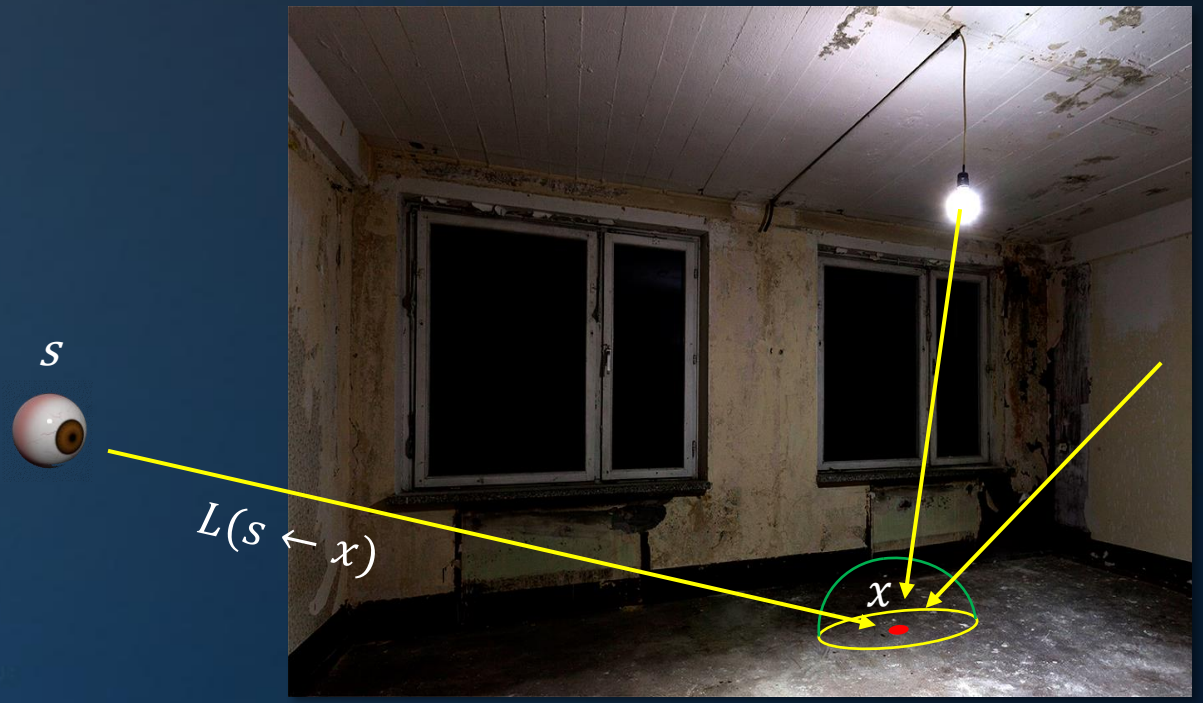
Previously in Advanced Graphics

The Rendering Equation:

$$L(s \leftarrow x) = L_E(s \leftarrow x) + \int_A f_r(s \leftarrow x \leftarrow x')\, L(x \leftarrow x')\, G(x \leftrightarrow x')\, dA(x')$$

"The light that travels from x to s is the light that x emits plus the light that x reflects."

"three-point formulation"



A: all points x' in the scene
f_r: BRDF
L: radiance
G: 'geometry factor'

BRDF: takes irradiance (from x'), calculated by:

$$L(x \leftarrow x')\, G(x \leftrightarrow x')$$

returns radiance (towards s).

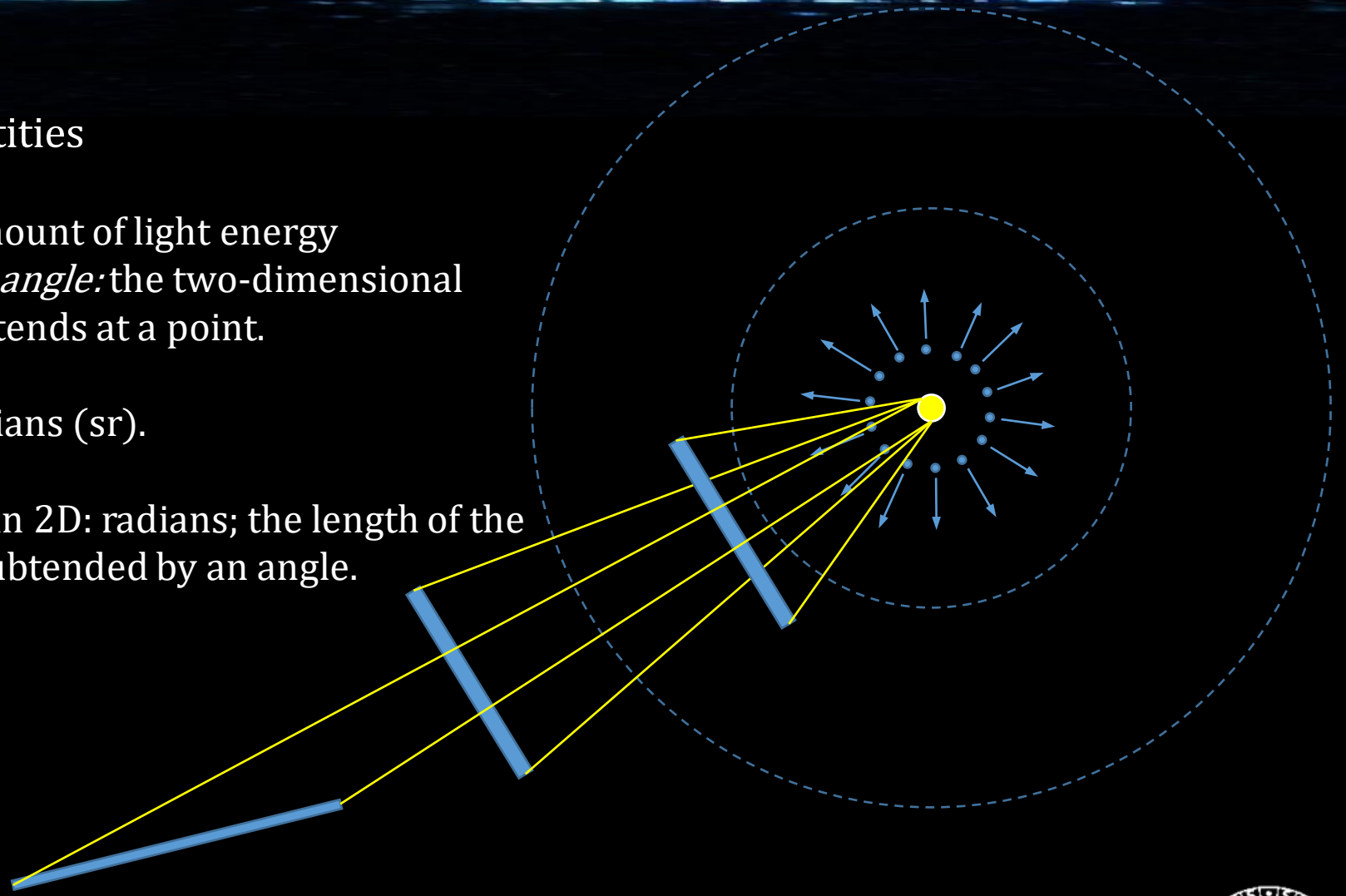
Light Transport

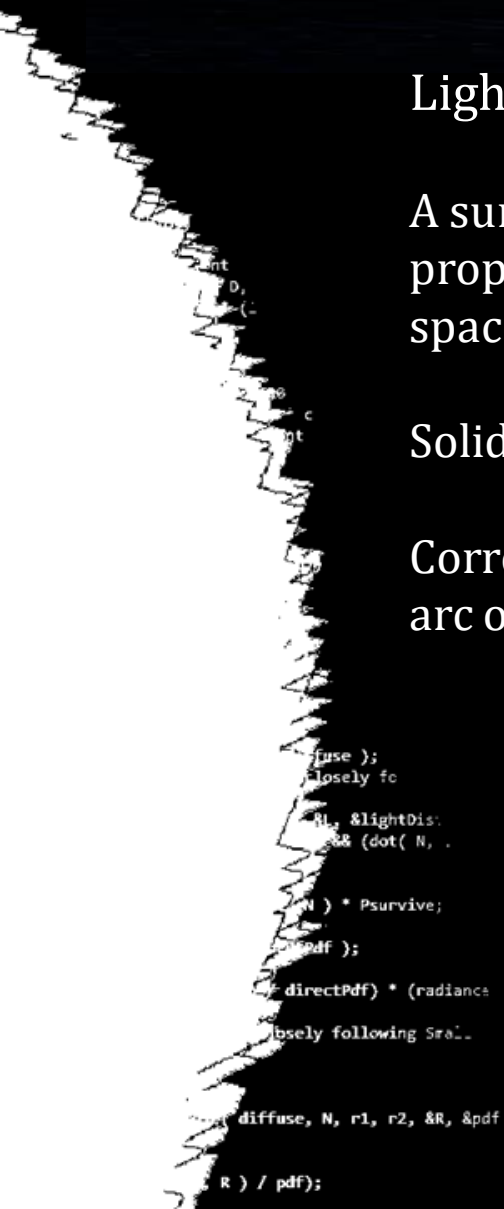
Light Transport Quantities

A surface receives an amount of light energy proportional to its *solid angle:* the two-dimensional space that an object subtends at a point.

Solid angle units: steradians (sr).

Corresponding concept in 2D: radians; the length of the arc on the unit sphere subtended by an angle.





Light Transport

Light Transport Quantities

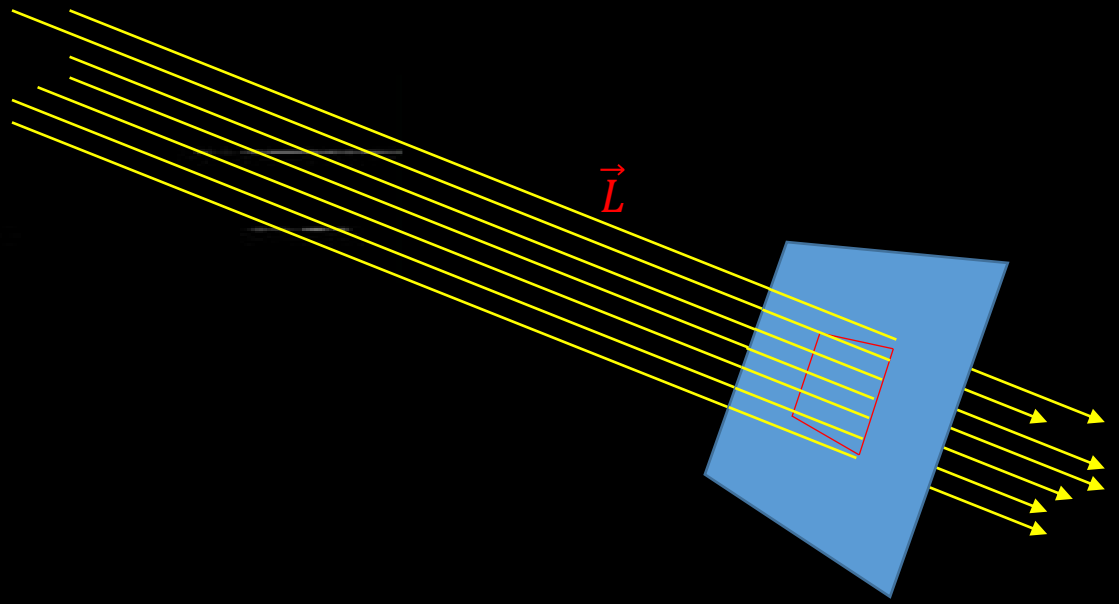
Radiance - L :

"The power of electromagnetic radiation emitted, reflected, transmitted or received per unit projected area per unit solid angle."

Units: $Wsr^{-1}m^{-2}$

Simplified particle analogy:
Amount of particles passing through a pipe with unit diameter, per unit time.

Note: radiance is a continuous value:
while flux at a point is 0 (since both area and solid angle are 0), we can still define flux per area per solid angle for that point.



Light Transport

Light Transport Quantities

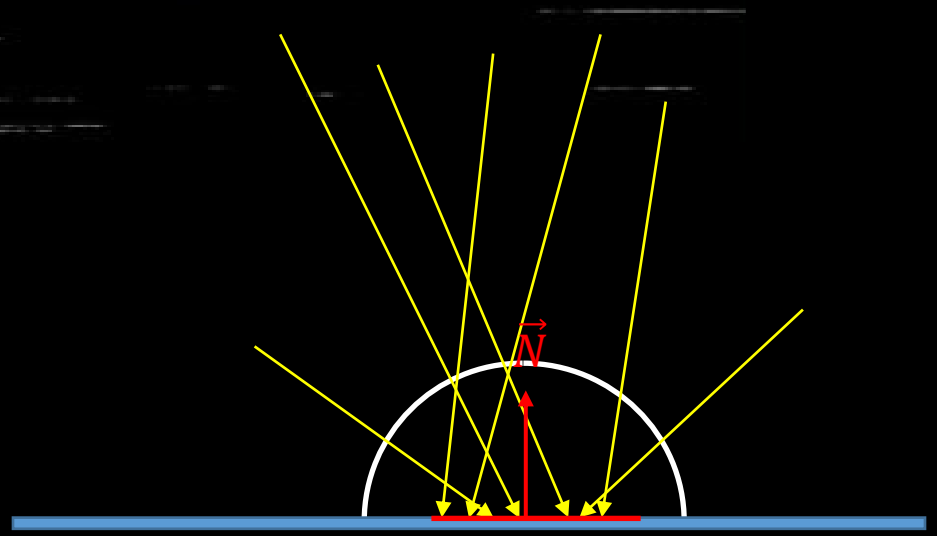
Irradiance - E :

"The power of electromagnetic radiation per unit area incident on a surface."

Units: Watts per m^2 = joules per second per m^2

$$Wm^{-2} = Jm^{-2}s^{-1}.$$

Simplified particle analogy:
number of photons arriving per unit area per unit time, from all directions.

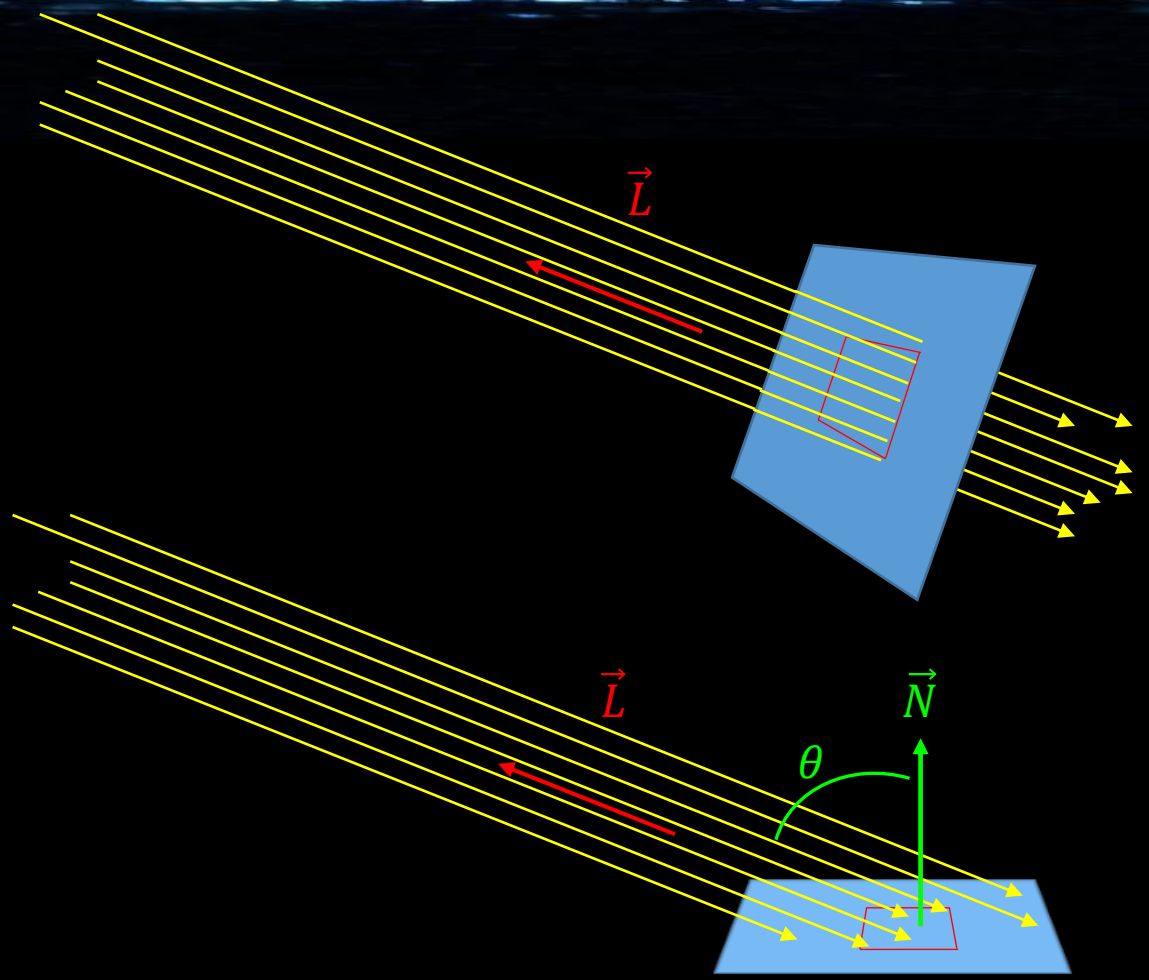


Light Transport

Light Transport Quantities

Converting radiance to irradiance:

$$E = L \cos\theta$$



Introduction

Previously in Advanced Graphics

$$G(x \leftrightarrow x') = \begin{cases} 0, & \text{if } x \leftrightarrow x' \text{ blocked} \\ \cos_i \dfrac{1}{\|x - x'\|^2}, & \text{otherwise} \end{cases}$$

The Rendering Equation:

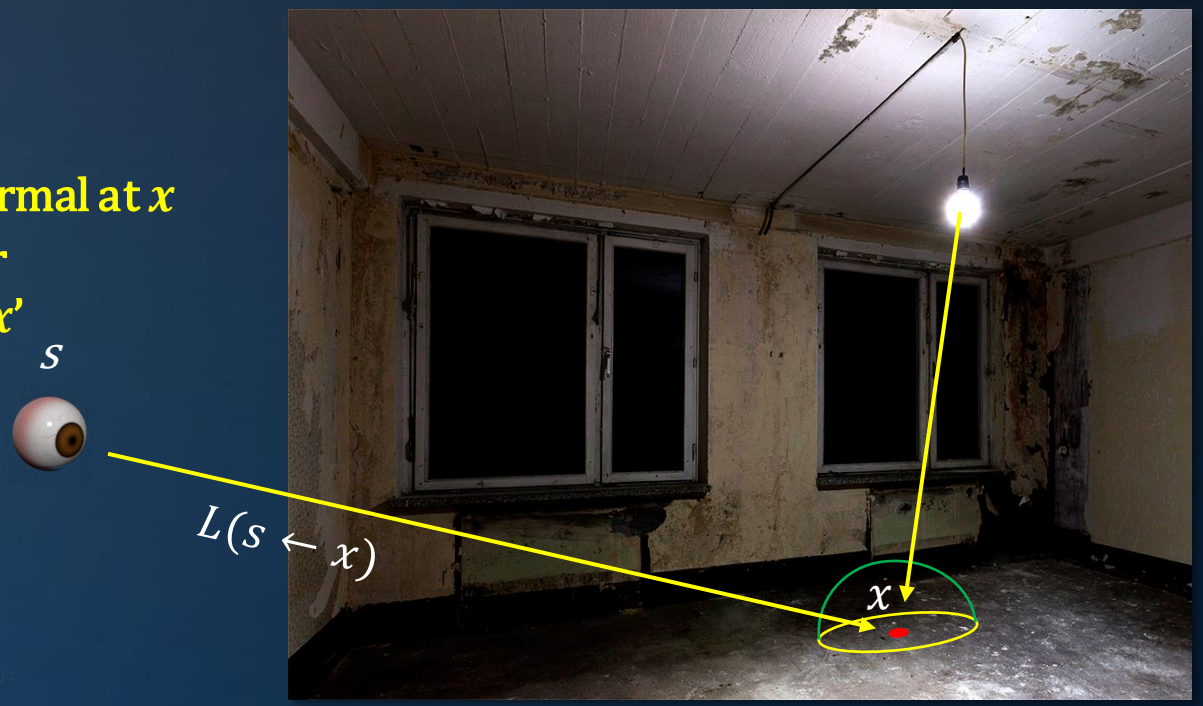
$$L(s \leftarrow x) = L_E(s \leftarrow x) + \int_A f_r(s \leftarrow x \leftarrow x')\, L(x \leftarrow x')\, G(x \leftrightarrow x')\, dA(x')$$

$\cos_i = "N \cdot L"$,

where:

$\vec{N}$ is the surface normal at x

$\vec{L}$ is the unit vector from x towards x'



***BRDF**: takes irradiance (from x'), calculated by:*

$$L(x \leftarrow x')\, G(x \leftrightarrow x')$$

returns radiance (towards s).

Introduction

The Rendering Equation (hemispherical formulation):

$$L_o(x, \omega_o) = L_E(x, \omega_o) + \int_{\Omega} f_r(x, \omega_o, \omega_i)\, L_i(x, \omega_i) \cos\theta_i \; d\omega_i$$

This time, we do not integrate over all points in the scene, but over all directions over the hemisphere Ω.

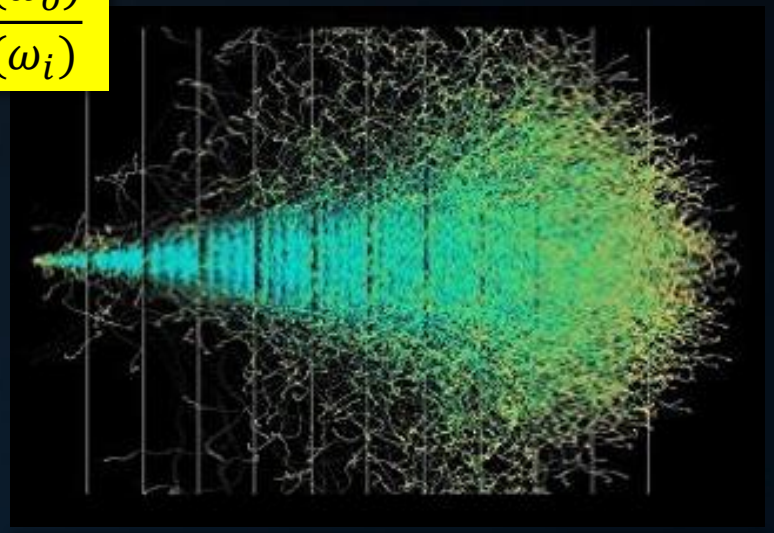
Differences:

- Visibility is now implicit: we send a ray to ω_i, it hits the first visible surface.
- Distance attenuation is implicit.
- "N dot L" is still there however: the BRDF still takes projected radiance.

$$L(s \leftarrow x) = L_E(s \leftarrow x) + \int_A f_r(s \leftarrow x \leftarrow x')\, L(x \leftarrow x')\, G(x \leftrightarrow x')\, dA(x')$$

Introduction

$$f_r(\omega_o, \omega_i) = \frac{L_o(\omega_o)}{E_i(\omega_i)}$$



Particle transport:

As an alternative to discrete flux / radiance / irradiance, we can reason about light transport in terms of particle transport.

- Flux then becomes the number of emitted photons;
- Radiance the number of photons travelling through a unit area in a unit direction;
- Irradiance the number of photons arriving on a unit area.

A BRDF tells us how many particles are absorbed, and how outgoing particles are distributed. The distribution depends on the incident and exitant direction.

Introduction

Probabilities:

We can also reason about the behavior of a single photon. In that case, the BRDF tells us the *probability* of a photon being absorbed, or leaving in a certain direction.

Introduction

Bidirectional Reflectance Distribution Function

BRDF: function describing the relation between radiance emitted in direction ω_o and irradiance arriving from direction ω_i:

$$f_r(\omega_o, \omega_i) = \frac{L_o(\omega_o)}{E_i(\omega_i)} = \frac{L_o(\omega_o)}{L_i(\omega_i)\cos\theta_i} = \frac{outgoing\ radiance}{incoming\ irradiance}$$

Or, if spatially variant:

$$f_r(x, \omega_o, \omega_i) = \frac{L_o(x, \omega_o)}{E_i(x, \omega_i)} = \frac{L_o(x, \omega_o)}{L_i(x, \omega_i)\cos\theta_i}$$

Properties:

- Should be positive: $f_r(\omega_o, \omega_i) \geq 0$
- Helmholtz reciprocity should be obeyed: $f_r(\omega_o, \omega_i) = f_r(\omega_i, \omega_o)$
- Energy should be conserved: $\int_\Omega f_r(\omega_o, \omega_i)\cos\theta_o\, d\omega_o \leq 1$

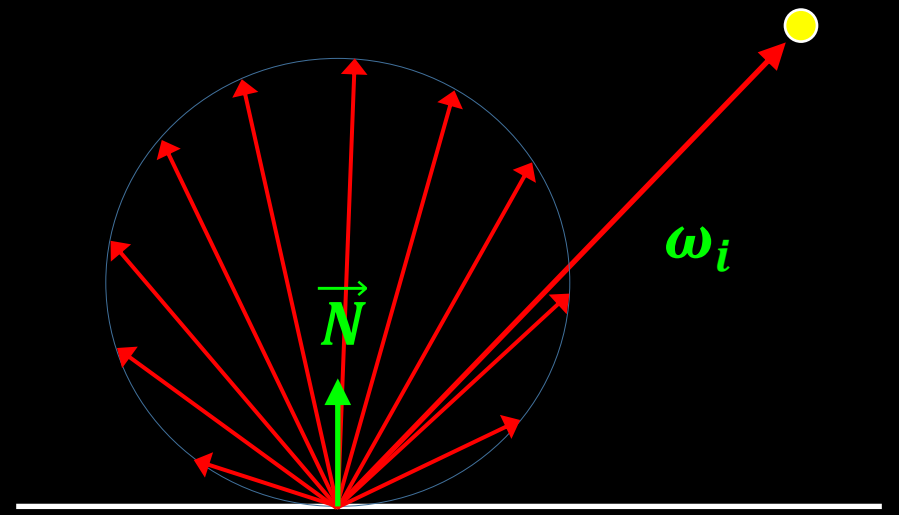
Introduction

Bidirectional Reflectance Distribution Function

The diffuse BRDF is:

$$f_r(\omega_o, \omega_i) = \frac{albedo}{\pi}$$

So, for a total irradiance E at surface point x, the outgoing radiance $L_o = E_i \frac{albedo}{\pi}$. *Why the π?*



Energy conservation: $E_o \leq E_i$

Suppose we have a directional light parallel to $\vec{N}$, with intensity 1. Then: $E_i = L_i = 1$. Suppose our BRDF $= \frac{albedo}{1}$.

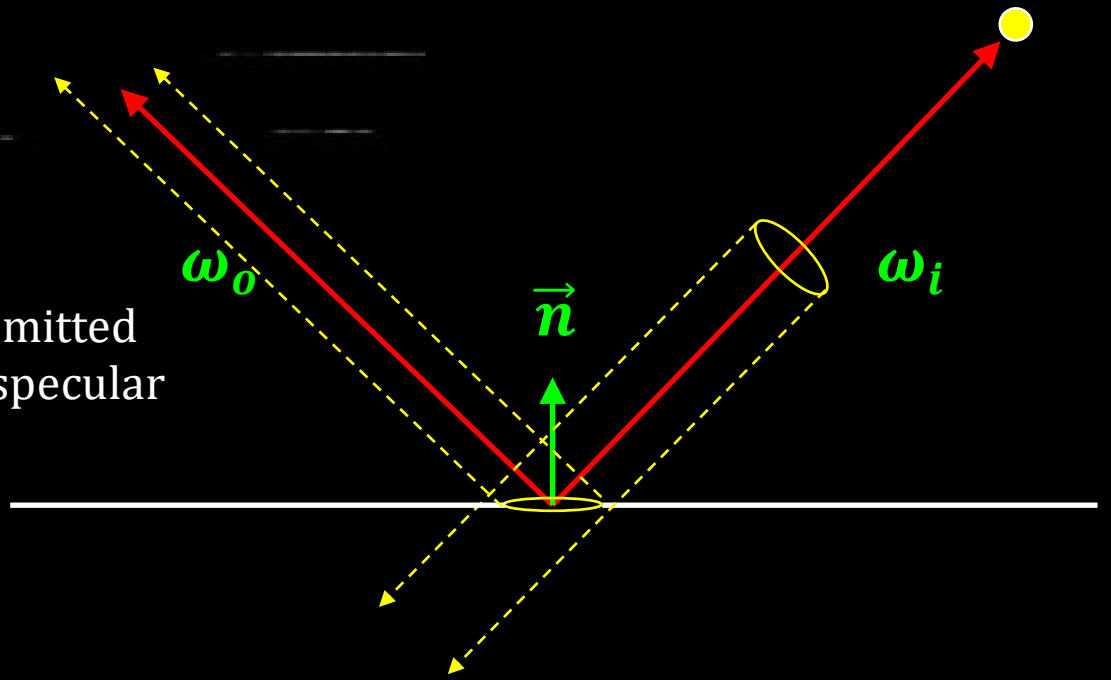
Then, for albedo $= 1$ we get: $E_o = \int_\Omega L_i \, f_r(\omega_o, \omega_i) \cos\omega_o \, d\omega_o = \int_\Omega \cos\omega_o \, d\omega_o$

Now: $\int_\Omega \cos\omega_o \, d\omega_o = \pi$ ➔ $E_o = \pi \, E_i$.

Introduction

Bidirectional Reflectance Distribution Function

Mirror / Perfect specular:
Reflects light in a fixed direction.



For a given incoming direction ω_i, all light is emitted in a single infinitesimal set of directions. The specular BRDF is a Dirac function:

$$f_r(x, \omega_o, \omega_i) = \begin{cases} \infty, \text{along reflected vector} \\ 0, \text{otherwise.} \end{cases}$$

This is not practical, and therefore we will handle the pure specular case (reflection and refraction) separately.

Introduction

Previously in Advanced Graphics

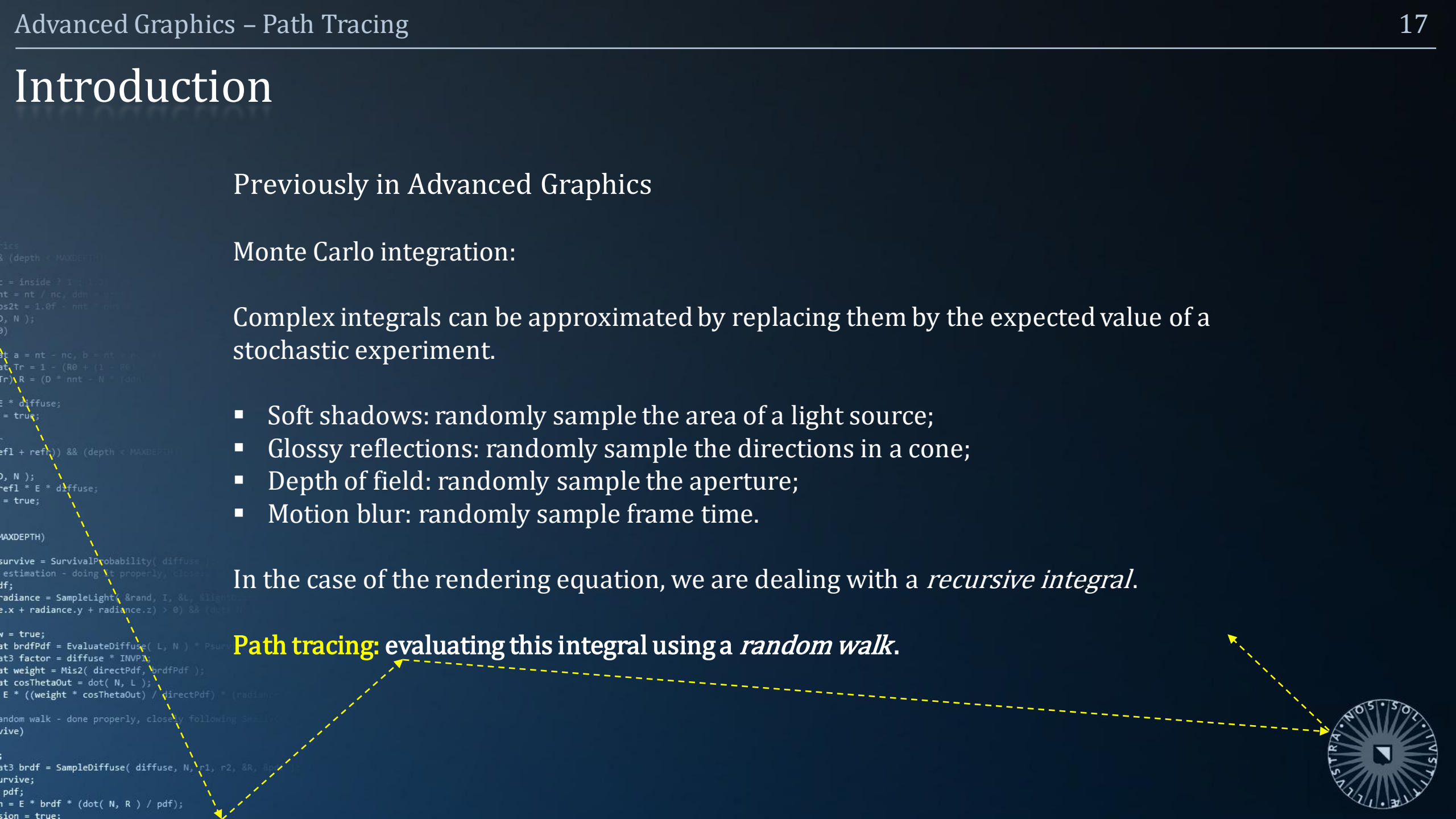
Monte Carlo integration:

Complex integrals can be approximated by replacing them by the expected value of a stochastic experiment.

- Soft shadows: randomly sample the area of a light source;
- Glossy reflections: randomly sample the directions in a cone;
- Depth of field: randomly sample the aperture;
- Motion blur: randomly sample frame time.

In the case of the rendering equation, we are dealing with a *recursive integral*.

Path tracing: evaluating this integral using a *random walk*.



Today’s Agenda:

- Introduction
- Path Tracing

Path Tracing

Solving the Rendering Equation

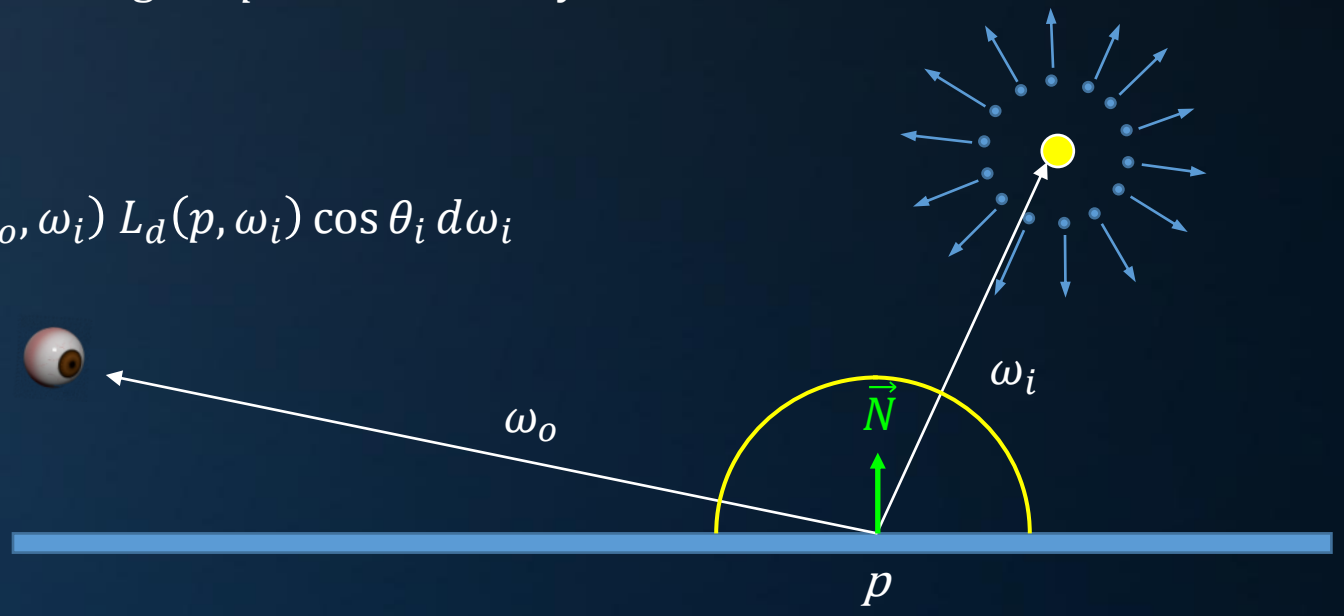
$$L_o(x,\omega_o) = L_E(x,\omega_o) + \int_\Omega f_r(x,\omega_o,\omega_i)\, L_i(x,\omega_i) \cos\theta_i \; d\omega_i$$

Let's start with direct illumination:

For a screen pixel, diffuse surface point p with normal $\vec{N}$ is directly visible.
What is the radiance travelling via p towards the eye?

Answer:

$$L_o(p,\omega_o) = \int_\Omega f_r(p,\omega_o,\omega_i)\, L_d(p,\omega_i) \cos\theta_i \, d\omega_i$$



Path Tracing

Solving the Rendering Equation

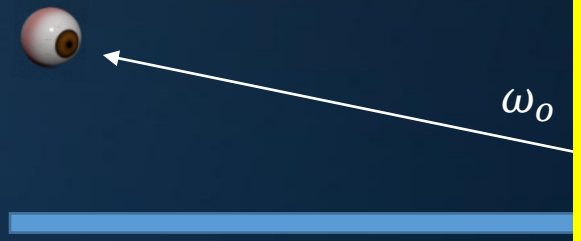
$$L_o(x, \omega_o) = L_E(x, \omega_o) + \int_\Omega f_r(x, \omega_o, \omega_i)\, L_i(x, \omega_i)\, c$$

Let's start with direct illumination:

For a screen pixel, diffuse surface point p with norm
What is the radiance travelling via p towards the ey

Answer:

$$L_o(p, \omega_o) = \int_\Omega f_r(p, \omega_o, \omega_i)\, L_d(p, \omega_i) \cos\theta_i \, d\omega_i$$



$$L_o(p, \omega_o) = \int_\Omega f_r(p, \omega_o, \omega_i)\, L_d(p, \omega_i) \cos\theta_i \, d\omega_i$$

$$= \int_\Omega \frac{albedo}{\pi}\, L_d(p, \omega_i) \cos\theta_i \, d\omega_i$$

$$= \frac{albedo}{\pi} \int_\Omega L_d(p, \omega_i) \cos\theta_i \, d\omega_i$$

In other words: the sum of radiance (scaled by $\cos\theta_i$ to convert to irradiance) arriving from all directions over the hemisphere, times $\frac{albedo}{\pi}$.

Q: What about distance attenuation?

A: A far-away light is found by fewer directions ω_i: it's **solid angle** on the hemisphere is smaller.

Q: What happened to ω_o?

A: The BRDF is independent of ω_o (it doesn't appear in the equation), but as ω_i approaches the horizon, $\cos\theta_i$ approaches zero.

Path Tracing

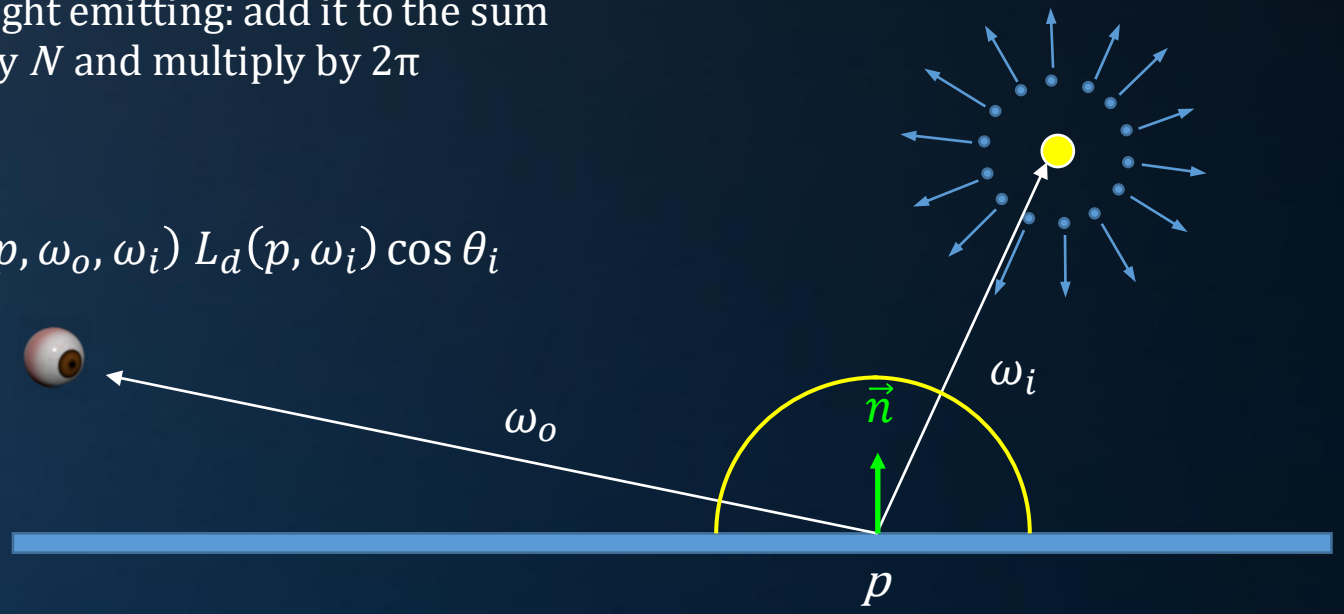
Direct Illumination

$$L_o(p,\omega_o) = \int_\Omega f_r(p,\omega_o,\omega_i)\, L_d(p,\omega_i) \cos\theta_i \, d\omega_i$$

We can solve this integral using Monte-Carlo integration:

- Chose N random directions over the hemisphere for p
- Find the first surface in each direction by tracing a ray
- If the surface is light emitting: add it to the sum
- Divide the sum by N and multiply by 2π

$$L_o(p,\omega_o) \approx \frac{2\pi}{N} \sum_{i=1}^{N} f_r(p,\omega_o,\omega_i)\, L_d(p,\omega_i) \cos\theta_i$$



Path Tracing

Direct Illumination

$$L_o(p,\omega_o) \approx \frac{2\pi}{N}\sum_{i=1}^{N} f_r(p,\omega_o,\omega_i)\, L_d(p,\omega_i)\cos\theta$$

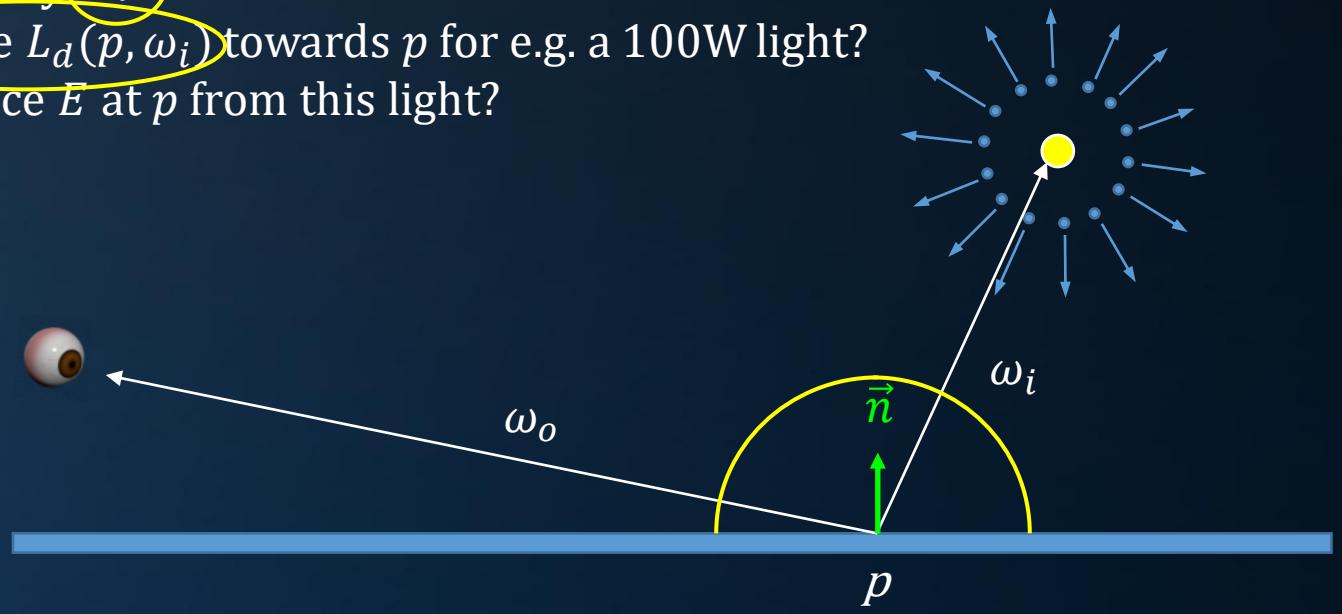
We integrate over the hemisphere, which has an area of 2π.

Do not confuse this with the $1/\pi$ factor in the BRDF, which doesn't compensate for the surface of the hemisphere, but the integral of $\cos\theta$ over the hemisphere (π).

Questions:

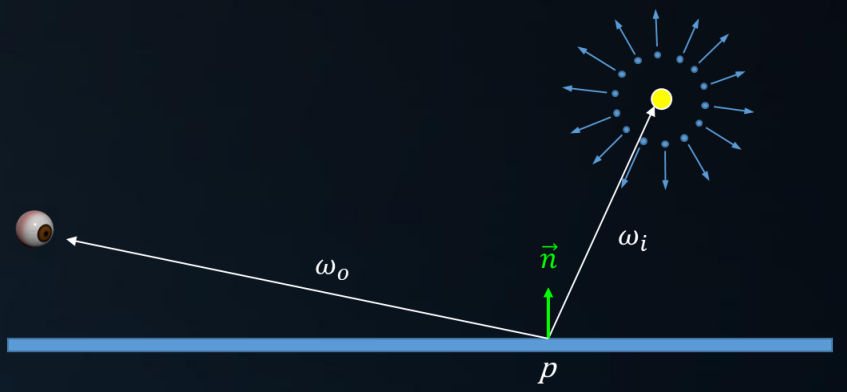
- Why do we multiply by 2π?
- What is the radiance $L_d(p,\omega_i)$ towards p for e.g. a 100W light?
- What is the irradiance E at p from this light?

L is per sr; $L_d(p,\omega_i)$ is proportional to the solid angle of the light as seen from p, so: $\sim(A_{disc}/r^2)$.



Path Tracing

Direct Illumination



$$L_o(p,\omega_o) = \int_\Omega f_r(p,\omega_o,\omega_i)\, L_d(p,\omega_i) \cos\theta_i \, d\omega_i$$

In many directions, we will not find light sources. We can improve our estimate by sampling the lights separately.

$$L_o(p,\omega_i) = \sum_{j=1}^{lights} \int_\Omega f_r(p,\omega_o,\omega_i)\, L_d^j(p,\omega_i) \cos\theta_i \, d\omega_i$$

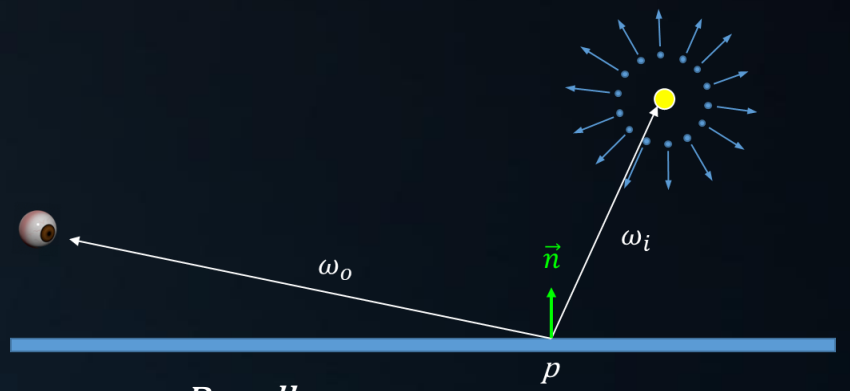
Obviously, sampling the entire hemisphere for each light is not necessary; we can sample the area of the light instead:

$$L_o(p,\omega_i) = \sum_{j=1}^{lights} \int_A f_r(p,\omega_o,\omega_i)\, L_d^j(p,\omega_i) \cos\theta_i \, d\omega_i$$

Path Tracing

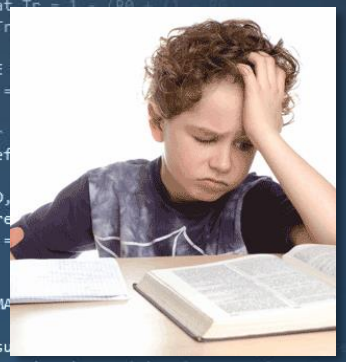
Direct Illumination

$$L_o(p,\omega_i) = \sum_{j=1}^{lights} \int_A f_r(p,\omega_o,\omega_i)\, L_d^j(p,\omega_i) \cos\theta_i \, d\omega_i$$



Recall:

$$V_A = \int_A^B f(x)\,dx \approx \frac{B-A}{N} \sum_{i=1}^{N} f(X_i)$$



Using Monte-Carlo:

$$L_o(p,\omega_i) \approx \frac{lights}{N} \sum_{i=1}^{N} f_r(p,\omega_o,P)\, L_d^J(p,P)\, V(p \leftrightarrow P) \frac{A_{L_d^J} \cos\theta_i \cos\theta_o}{\| p-P \|^2}$$

where

- $L_d^J(p,P)$ is the direct light towards p from random point P on random light J
- $V(p \leftrightarrow P)$ is the mutual visibility between p and P
- $A_{L_d^J}$ is the area of this light source
- $(A_{L_d^J} \cos\theta_o)/(\| p-P \|^2)$ is the area of the light source projected on the hemisphere, which approximates the solid angle of the light.

Path Tracing

Direct Illumination

We now have two methods to estimate direct illumination using Monte Carlo integration:

1. By random sampling the hemisphere:

$$L_o(p,\omega_o) \approx \frac{2\pi}{N}\sum_{i=1}^{N} f_r(p,\omega_o,\Omega_i)\, L_d(p,\Omega_i)\cos\theta_i$$

2. By sampling the lights directly:

$$L_o(p,\omega_i) \approx \frac{lights}{N}\sum_{i=1}^{N} f_r(p,\omega_o,P)\, L_d^J(p,P)\, V(p \leftrightarrow P)\frac{A_{L_d^J}\cos\theta_i\cos\theta_o}{\| p-P \|^2}$$

For $N = \infty$, these yield the same result.

Path Tracing

Direct Illumination

We now have two methods to estimate direct illumination using Monte Carlo integration:

1. By random sampling the hemisphere:

$$L_o(p,\omega_o) \approx \frac{2\pi}{N}\sum_{i=1}^{N} f_r(p,\omega_o,\Omega_i)\, L_d(p,\Omega_i)\cos\theta_i$$

2. By sampling the lights directly (three point formulation):

$$L_o(s \leftarrow p) \approx \frac{lights}{N}\sum_{i=1}^{N} f_r(s \leftarrow p \leftarrow Q)\, L_d^J(p \leftarrow Q)\, V(p \leftrightarrow Q)\frac{A_{L_d^J}\cos\theta_i\cos\theta_o}{\| p-Q \|^2}$$

For $N = \infty$, these yield the same result.

Path Tracing

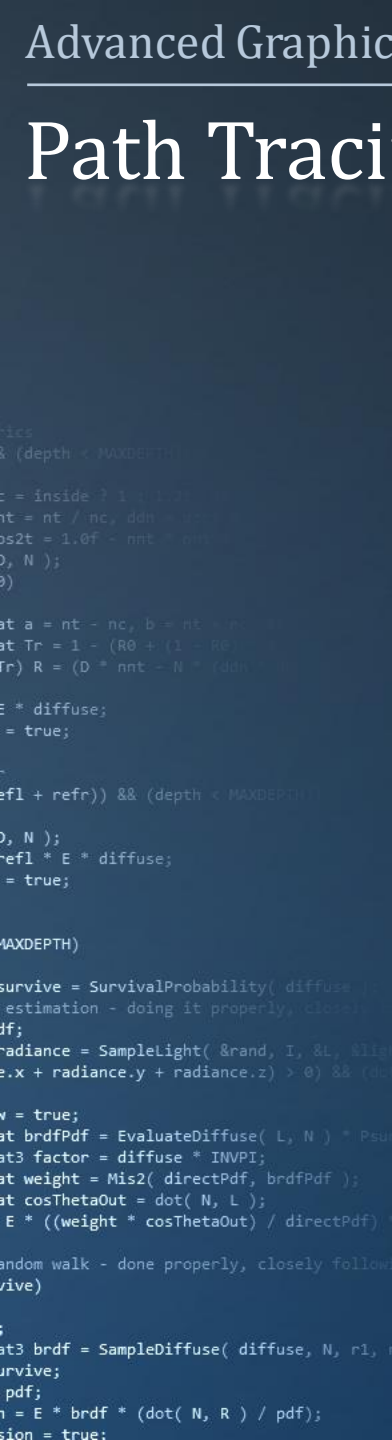
$$L_o(p,\omega_o) \approx \frac{2\pi}{N} \sum_{i=1}^{N} f_r(p,\omega_o,\Omega_i)\, L_d(p,\Omega_i) \cos\theta_i$$

Verification

Method 1 in a small C# ray tracing framework:

```
In: Ray ray, with members O, D, N, t.
Already calculated: intersection point I = O + t * D.

Vector3 R = RTTools.DiffuseReflection( ray.N );
Ray rayToHemisphere = new Ray( I + R * EPSILON, R, 1e34f );
Scene.Intersect( rayToHemisphere );
if (rayToHemisphere.objIdx == LIGHT)
{
    Vector3 BRDF = material.diffuse * INVPI;
    float cos_i = Vector3.Dot( R, ray.N );
    return 2.0f * PI * BRDF * Scene.lightColor * cos_i;
}
```



Path Tracing

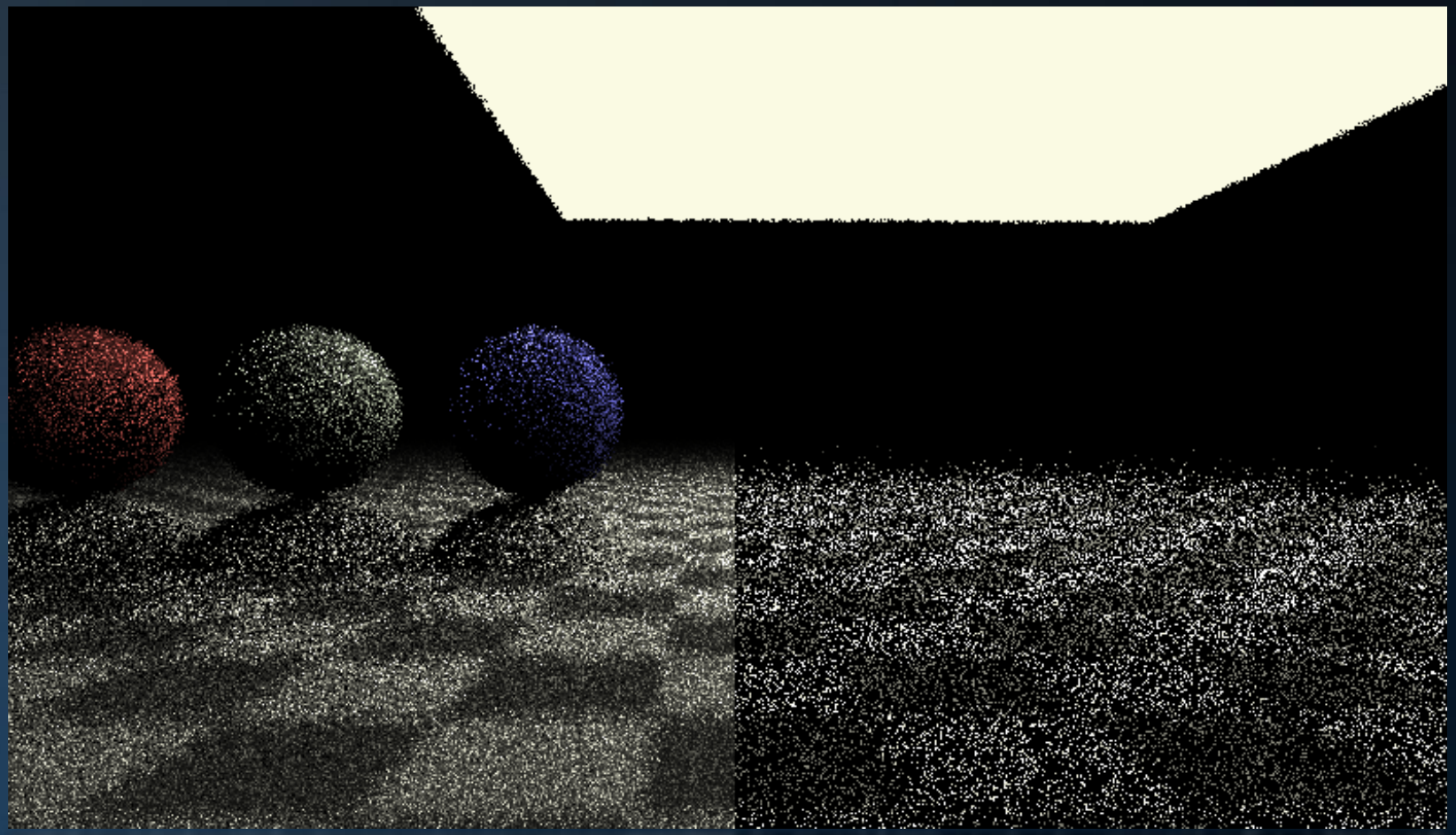
$$L_o(p,\omega_i) \approx lights * \frac{1}{N}\sum_{i=1}^{N} f_r(p,\omega_o,P)\, L_d^J(p,P)\, V(p \leftrightarrow P) \frac{A_{L_d^J} \cos\theta_i \cos\theta_o}{\| p-P \|^2}$$

Verification

Method 2 in a small C# ray tracing framework:

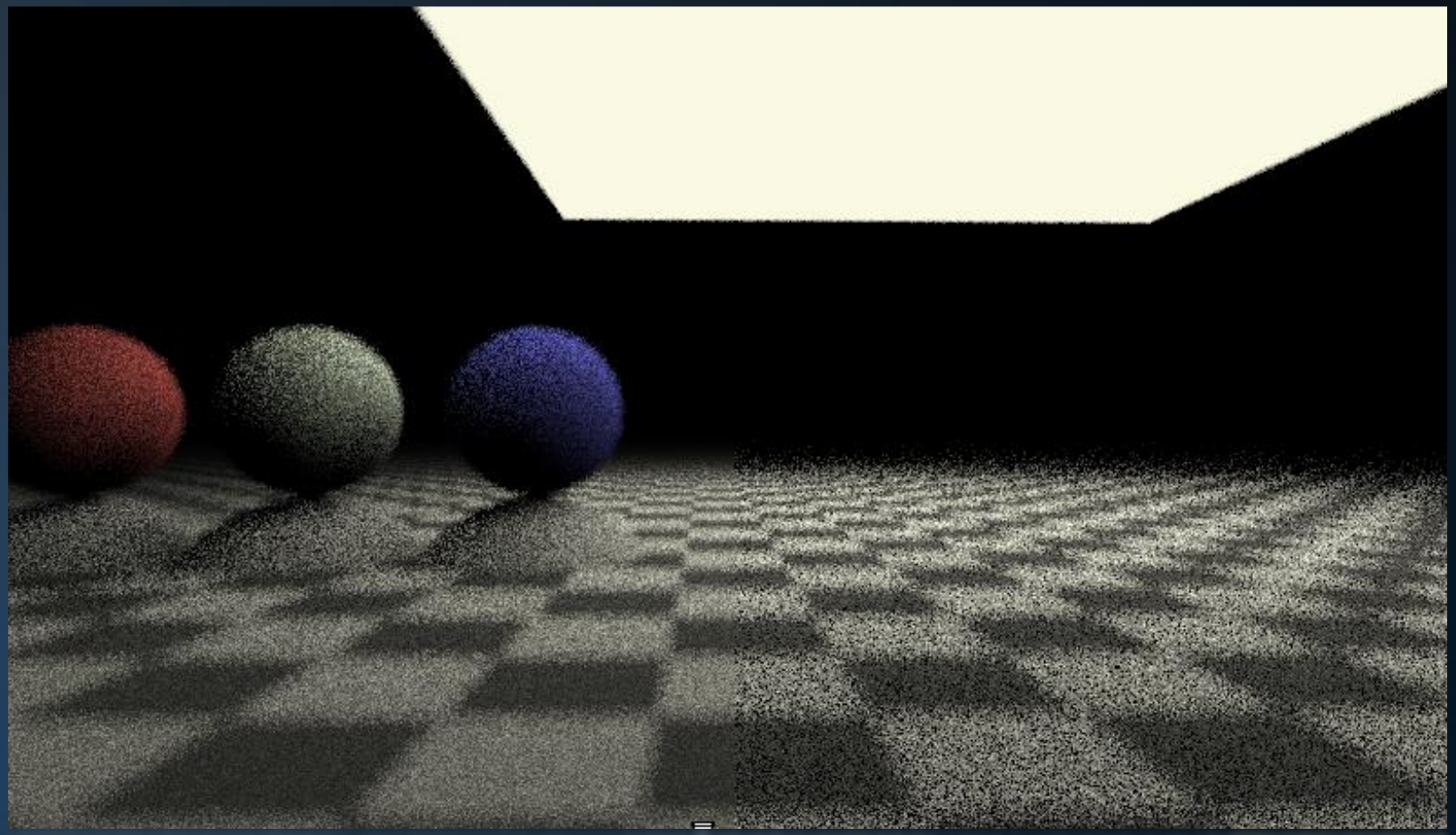
```
// construct vector to random point on light
Vector3 L = Scene.RandomPointOnLight() - I;
float dist = L.Length();
L /= dist;
float cos_o = Vector3.Dot( -L, lightNormal );
float cos_i = Vector3.Dot( L, ray.N );
if ((cos_o <= 0) || (cos_i <= 0)) return BLACK;
// light is not behind surface point, trace shadow ray
Ray r = new Ray( I + EPSILON * L, L, dist - 2 * EPSILON );
Scene.Intersect( r );
if (r.objIdx != -1) return Vector3.Zero;
// light is visible (V(p,p')=1); calculate transport
Vector3 BRDF = material.diffuse * INVPI;
float solidAngle = (cos_o * Scene.LIGHTAREA) / (dist * dist);
return BRDF * lightCount * Scene.lightColor * solidAngle * cos_i;
```

Path Tracing



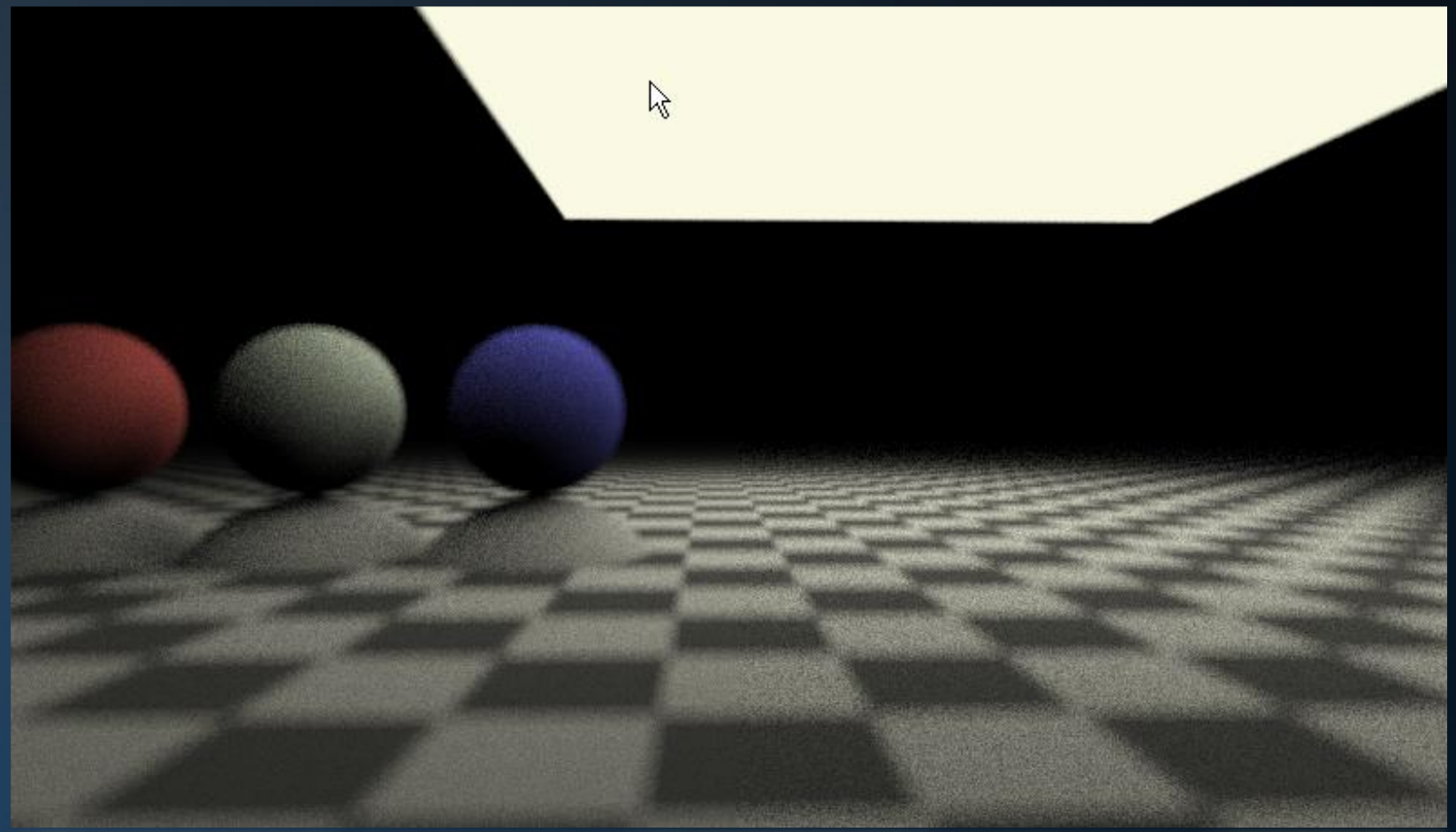
0.1s

Path Tracing



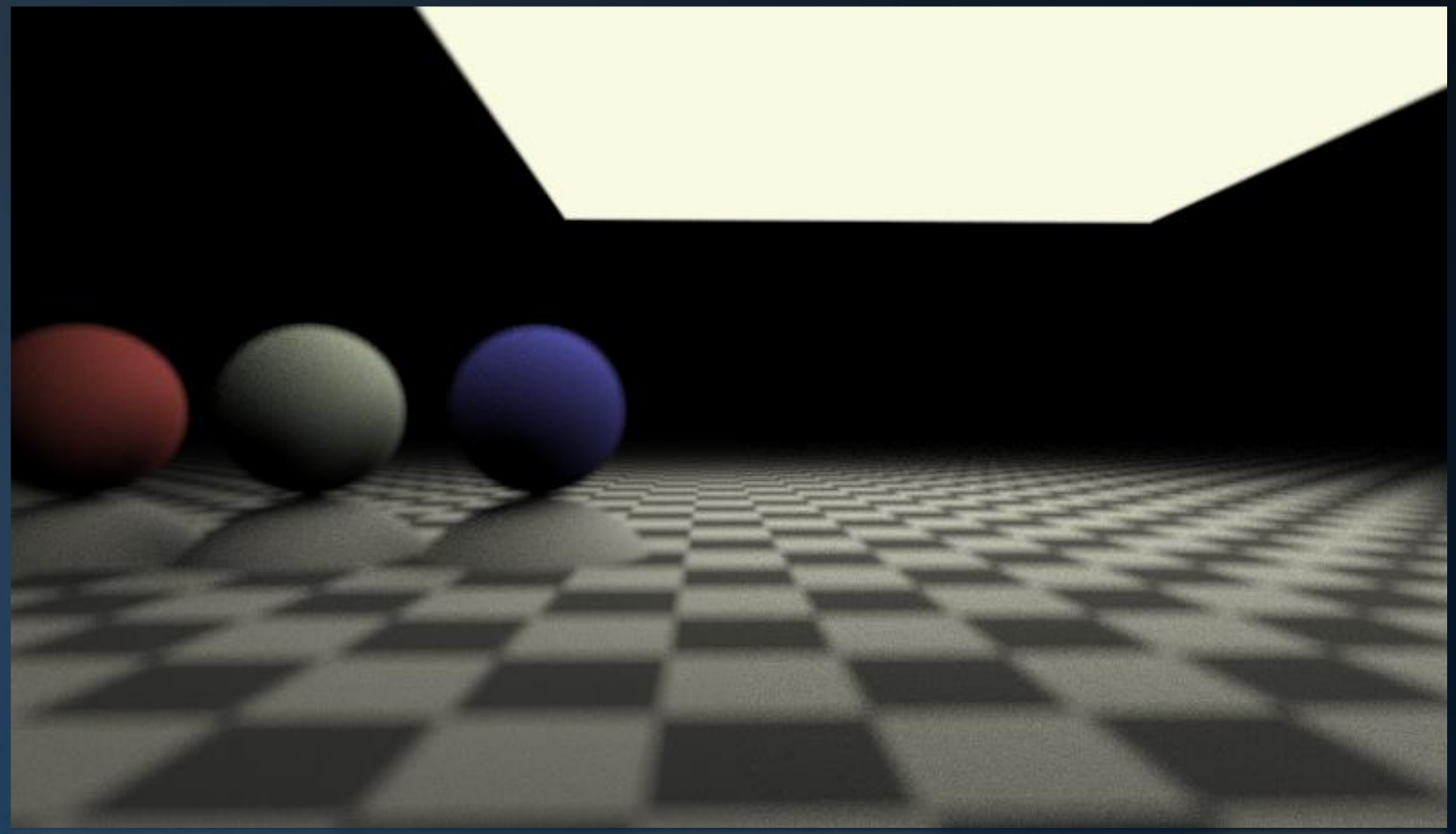
0.5s

Path Tracing



2.0s

Path Tracing



30.0s

Path Tracing

Rendering using Monte Carlo Integration

In the demonstration, we sampled each light using only 1 sample. The (very noisy) result is directly visualized.

To get a better estimate, we average the result of several frames (and thus: several samples).

Observations:

1. The light sampling estimator is much better than the hemisphere estimator;
2. Relatively few samples are sufficient for a recognizable image;
3. Noise reduces over time, but we quickly get diminishing returns.

Path Tracing

Indirect Light

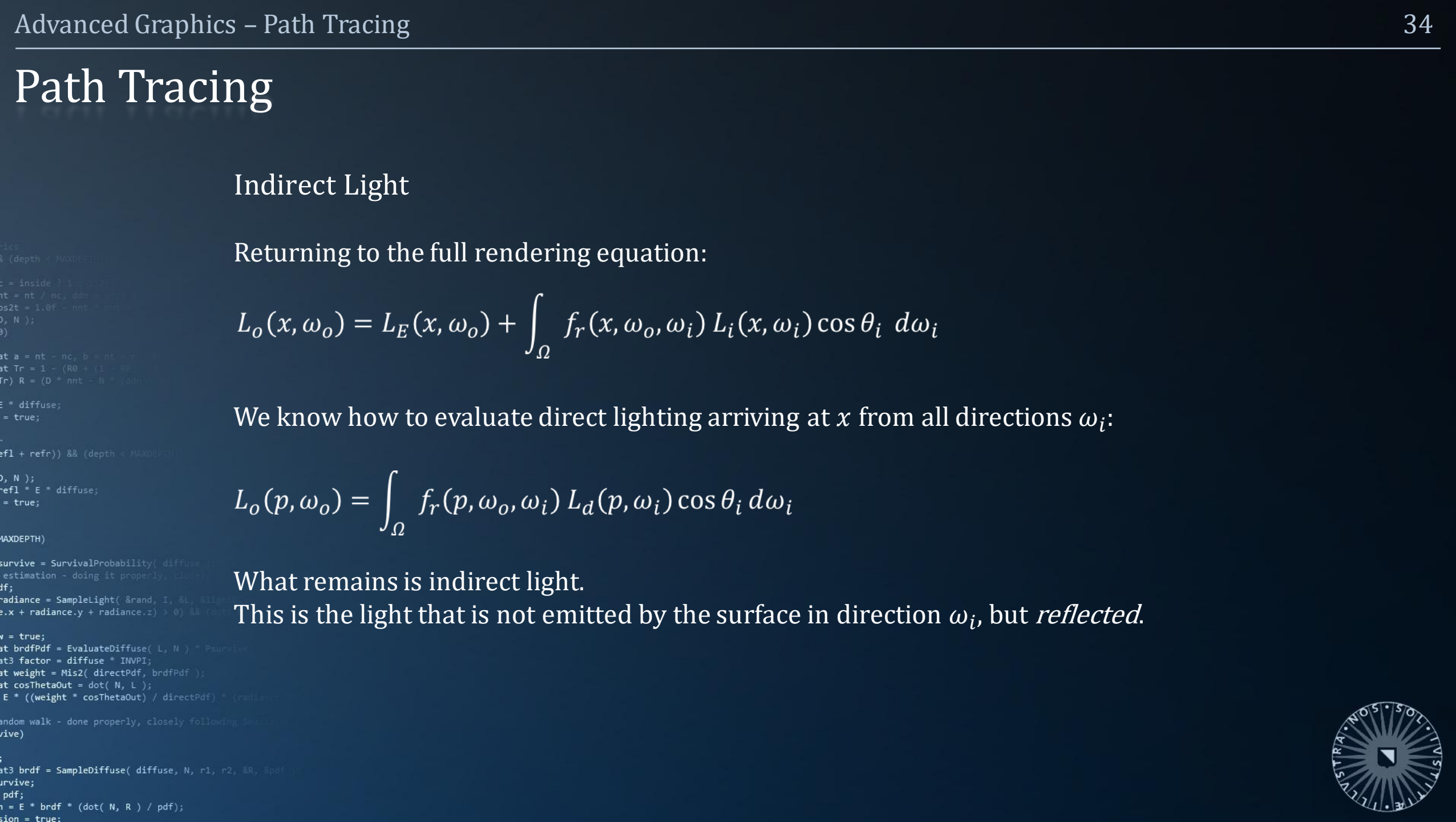
Returning to the full rendering equation:

$$L_o(x, \omega_o) = L_E(x, \omega_o) + \int_\Omega f_r(x, \omega_o, \omega_i)\, L_i(x, \omega_i) \cos\theta_i \; d\omega_i$$

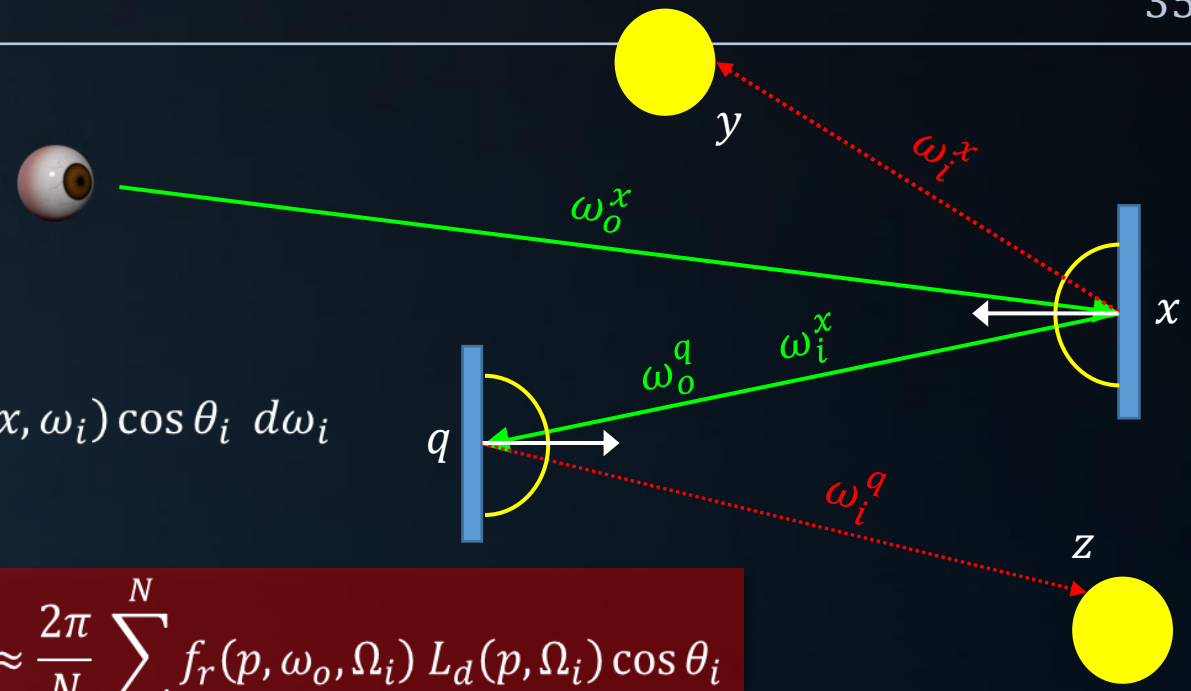
We know how to evaluate direct lighting arriving at x from all directions ω_i:

$$L_o(p, \omega_o) = \int_\Omega f_r(p, \omega_o, \omega_i)\, L_d(p, \omega_i) \cos\theta_i \, d\omega_i$$

What remains is indirect light.
This is the light that is not emitted by the surface in direction ω_i, but *reflected*.



Path Tracing



Indirect Light

$$L_o(x,\omega_o) = L_E(x,\omega_o) + \int_\Omega f_r(x,\omega_o,\omega_i)\, L_i(x,\omega_i) \cos\theta_i \; d\omega_i$$

Let's expand / reorganize this:

$$L_o(x,\omega_o^x) = L_E(x,\omega_o^x)$$

$$L_o(p,\omega_o) \approx \frac{2\pi}{N} \sum_{i=1}^{N} f_r(p,\omega_o,\Omega_i)\, L_d(p,\Omega_i) \cos\theta_i$$

$$+ \int_\Omega L_E(y,\omega_o^y)\, f_r(x,\omega_o^x,\omega_i^x) \cos\theta_i^x \; d\omega_i^x$$

direct light on x

$$+ \int_\Omega \int_\Omega L_E(z,\omega_o^q)\, f_r(y,\omega_o^q,\omega_i^q) \cos\theta_i^q \; f_r(x,\omega_o^x,\omega_i^x) \cos\theta_i^x \; d\omega_i^x \, d\omega_i^q$$

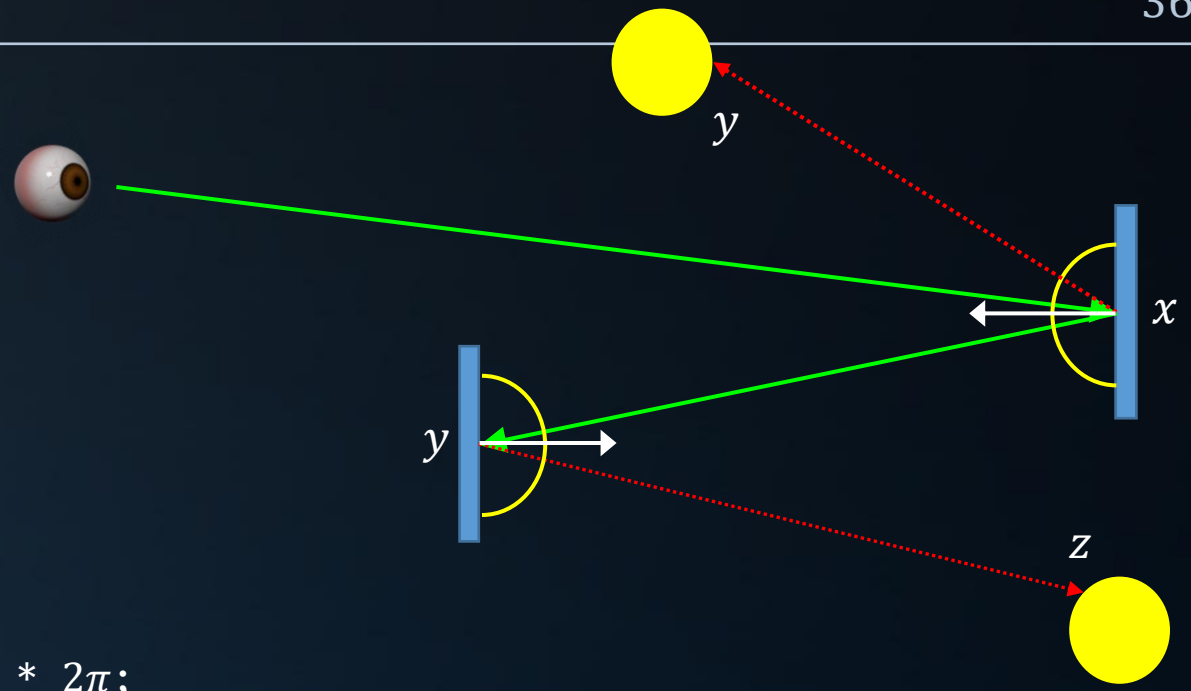
1st bounce

$$+ \int_\Omega \int_\Omega \int_\Omega \ldots$$

2nd bounce

$$\approx \ldots$$

Path Tracing



Indirect Light

One particle finding the light via a surface:

```
I, N = Trace( ray );
R = DiffuseReflection( N );
lightColor = Trace( new Ray( I, R ) );
return dot( R, N ) * albedo / π * lightColor * 2π;
```

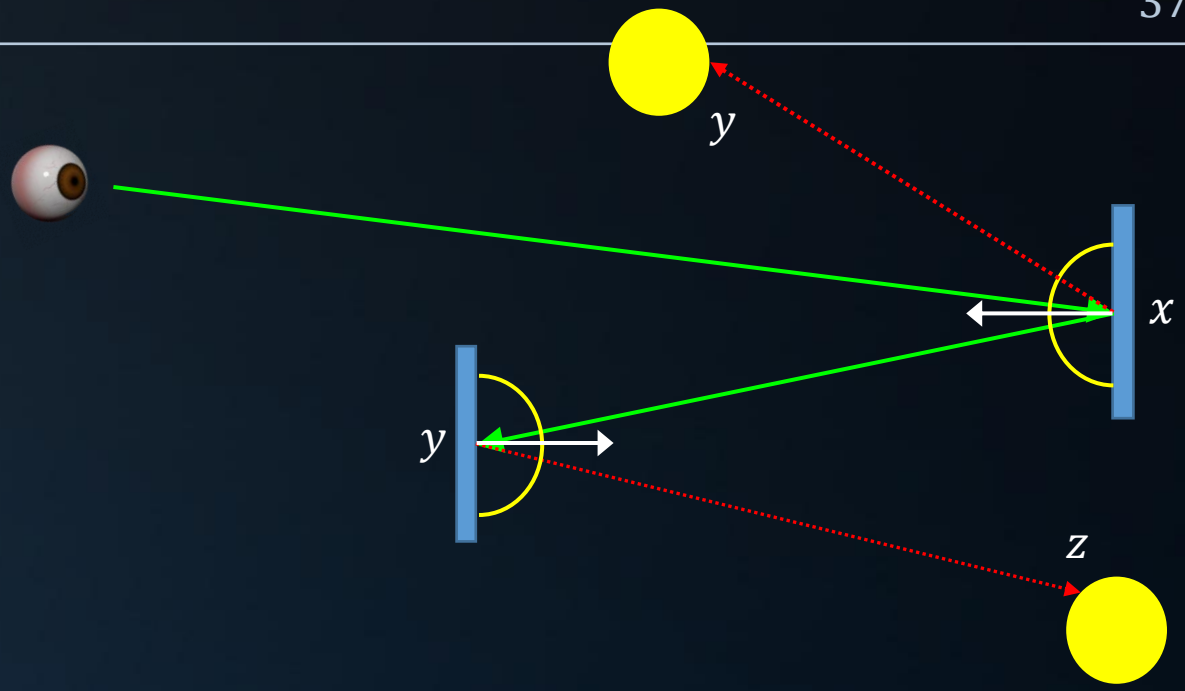
One particle finding the light via two surfaces:

```
I1, N1 = Trace( ray );
R1 = DiffuseReflection( N1 );
I2, N2 = Trace( new Ray( I1, R1 ) );
R2 = DiffuseReflection( N2 );
lightColor = Trace( new Ray( I2, R2 ) );
return dot( R1, N1 ) * albedo / π * 2π * dot( R2, N2 ) * albedo / π * 2π * lightColor;
```

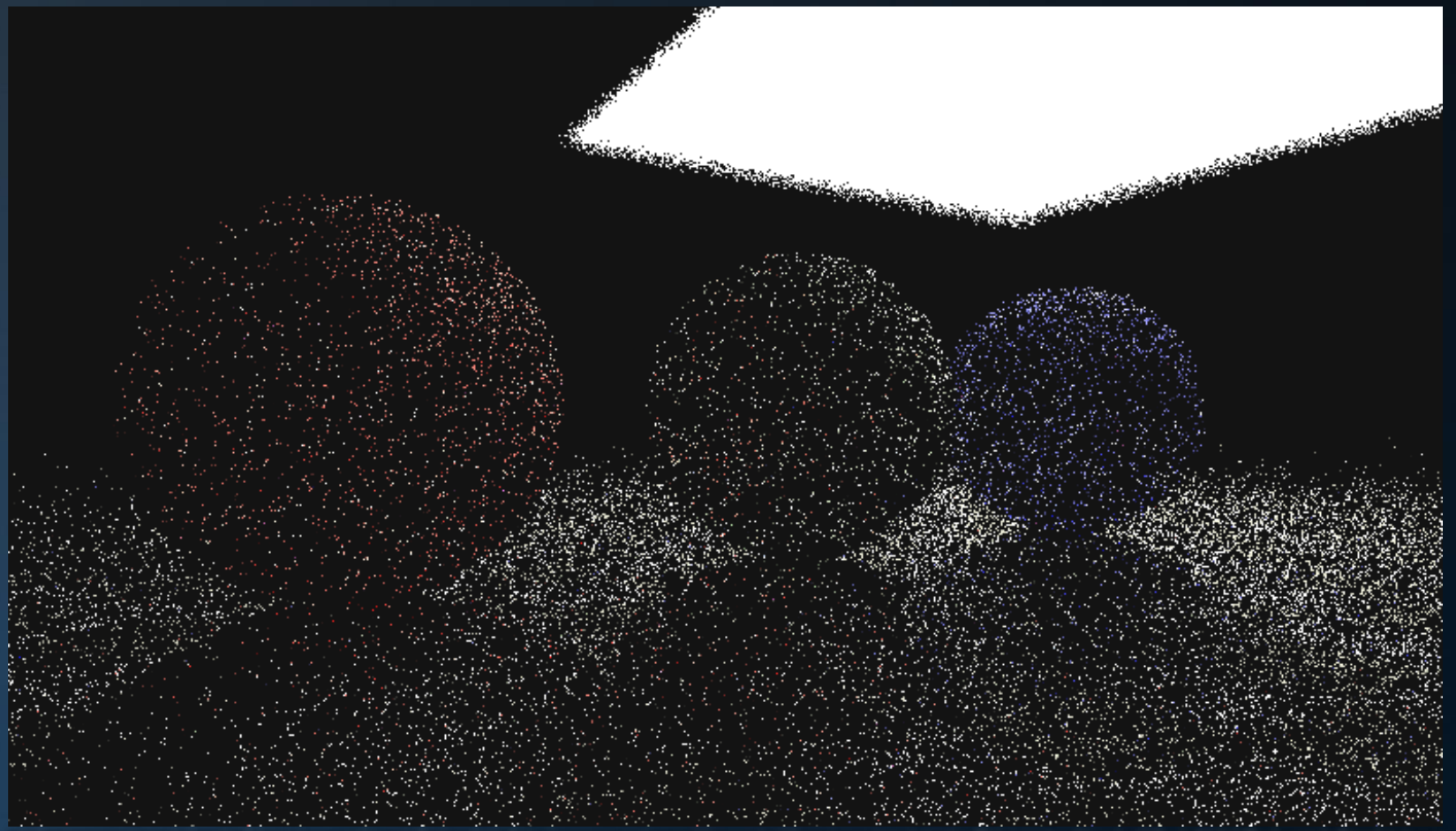
Path Tracing

Path Tracing Algorithm

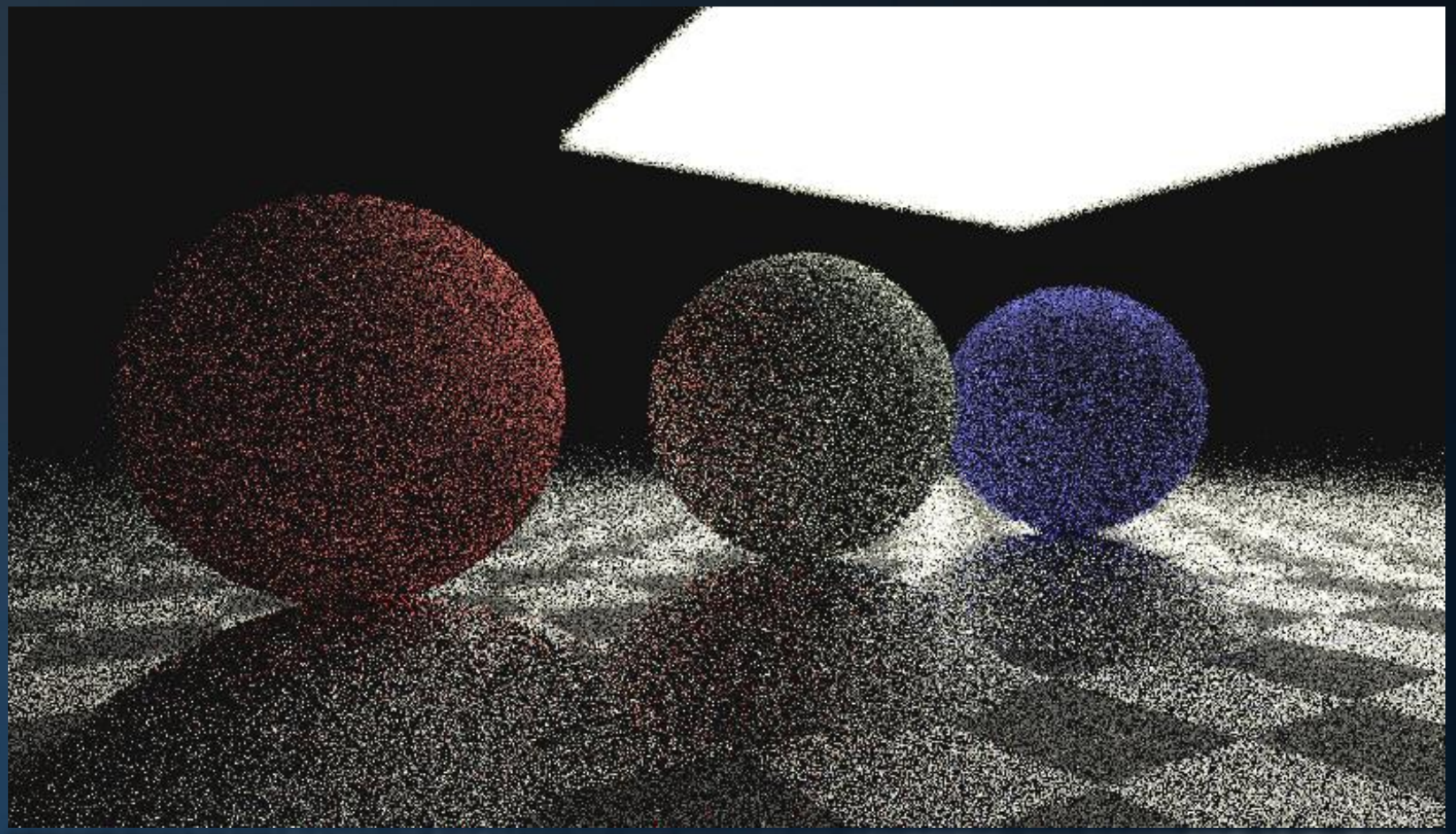
```
Color Sample( Ray ray )
{
    // trace ray
    I, N, material = Trace( ray );
    // terminate if ray left the scene
    if (ray.NOHIT) return BLACK;
    // terminate if we hit a light source
    if (material.isLight) return material.emittance;
    // continue in random direction
    R = DiffuseReflection( N );
    Ray newRay( I, R );
    // update throughput
    BRDF = material.albedo / PI;
    Ei = Sample( newRay ) * dot( N, R ); // irradiance
    return PI * 2.0f * BRDF * Ei;
}
```



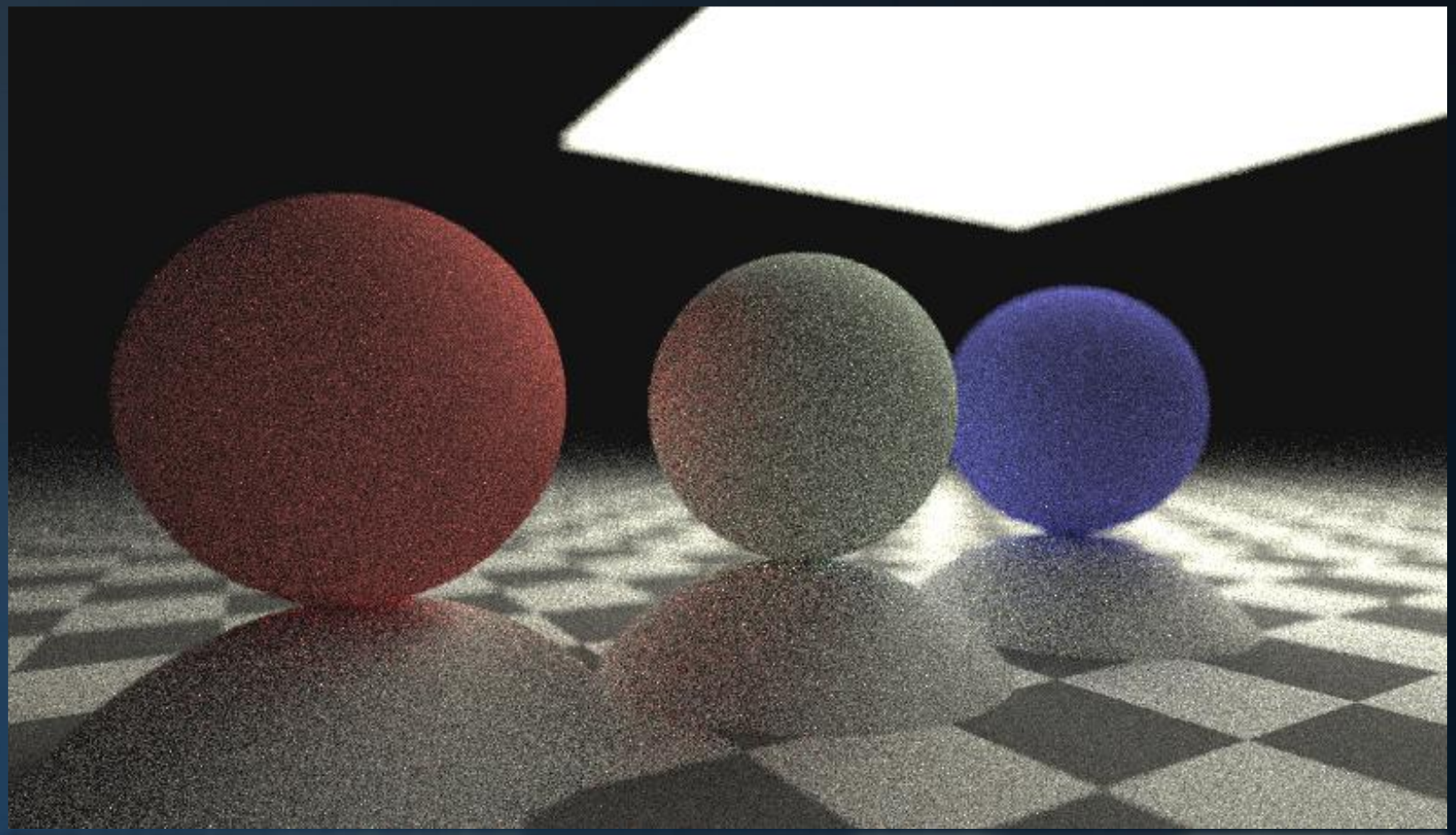
Path Tracing



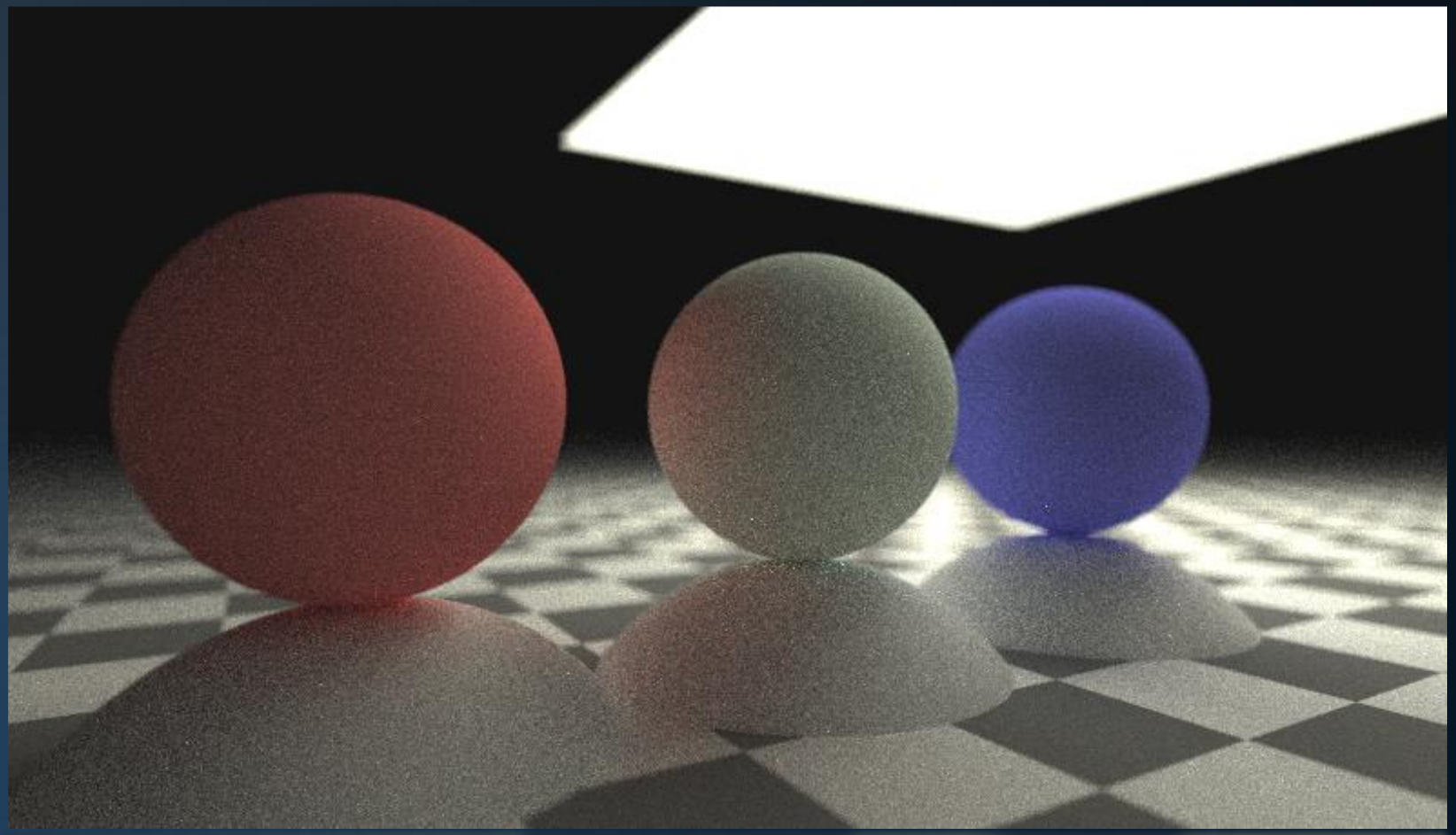
Path Tracing



Path Tracing



Path Tracing



Path Tracing

Particle Transport

The random walk is analogous to particle transport:

- a particle leaves the camera
- at each surface, energy is absorbed proportional to 1-albedo (‘surface color’)
- at each surface, the particle picks a new direction
- at a light, the path transfers energy to the camera.

In the simulation, particles seem to travel backwards. This is valid because of the Helmholtz reciprocity.

Notice that longer paths tend to return less energy.

```
Color Sample( Ray ray )
{
   // trace ray
   I, N, material = Trace( ray );
   // terminate if ray left the scene
   if (ray.NOHIT) return BLACK;
   // terminate if we hit a light source
   if (material.isLight) return emittance;
   // continue in random direction
   R = DiffuseReflection( N );
   Ray r( I, R );
   // update throughput
   BRDF = material.albedo / PI;
   Ei = Sample( r ) * (N·R);
   return PI * 2.0f * BRDF * Ei;
}
```

Path Tracing

Particle Transport - Mirrors

Handling a pure specular surface:

A particle that encounters a mirror continues in a deterministic way.

```
Color Sample( Ray ray )
{
   // trace ray
   I, N, material = Trace( ray );
   // terminate if ray left the scene
   if (ray.NOHIT) return BLACK;
   // terminate if we hit a light source
   if (material.isLight) return emittance;
   // surface interaction
   if (material.isMirror)
   {
      // continue in fixed direction
      Ray r( I, Reflect( N ) );
      return material.albedo * Sample( r );
   }
   // continue in random direction
   R = DiffuseReflection( N );
   BRDF = material.albedo / PI;
   Ray r( I, R );
   // update throughput
   Ei = Sample( r ) * (N·R);
   return PI * 2.0f * BRDF * Ei;
}
```

Path Tracing

Particle Transport - Glass

Handling dielectrics:

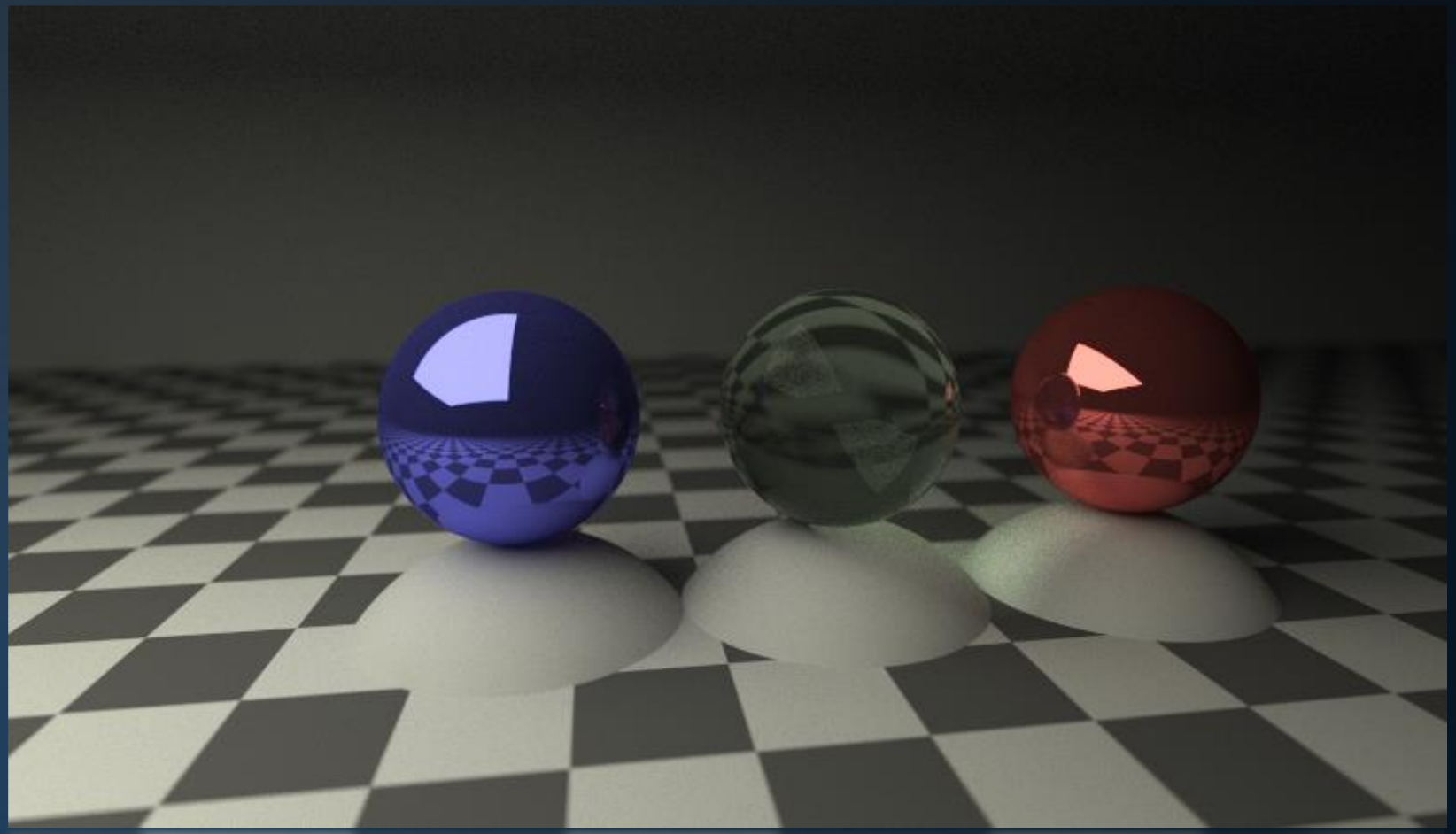
Dielectrics reflect *and* transmit light.
In the ray tracer, we handled this using two rays.

A particle must chose.

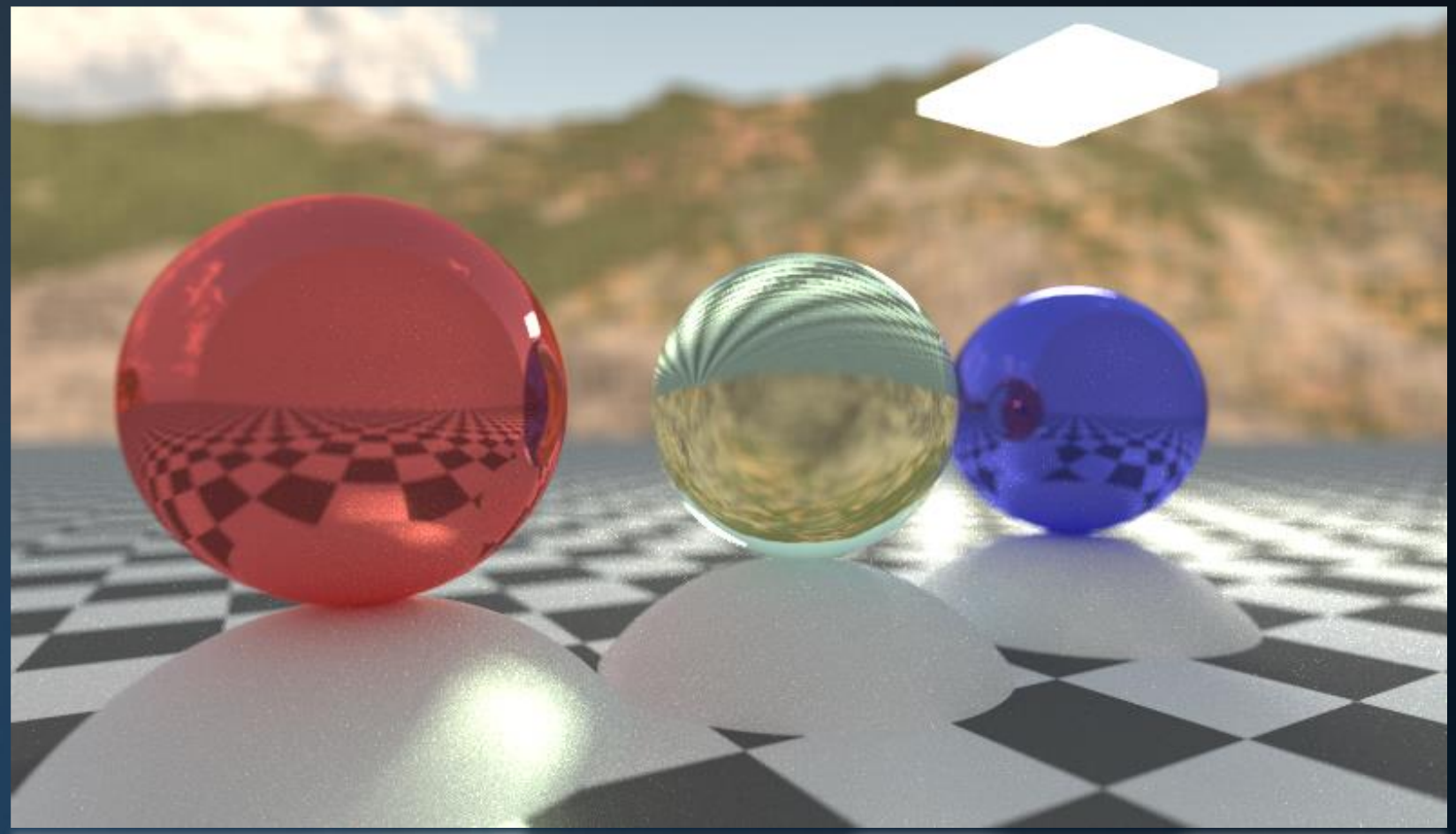
The probability of each choice is calculated using the Fresnel equations.

```
Color Sample( Ray ray )
{
   // trace ray
   I, N, material = Trace( ray );
   // terminate if ray left the scene
   if (ray.NOHIT) return BLACK;
   // terminate if we hit a light source
   if (material.isLight) return emittance;
   // surface interaction
   if (material.isMirror)
   {
      // continue in fixed direction
      Ray r( I, Reflect( N ) );
      return material.albedo * Sample( r );
   }
   // continue in random direction
   R = DiffuseReflection( N );
   BRDF = material.albedo / PI;
   Ray r( I, R );
   // update throughput
   Ei = Sample( r ) * (N·R);
   return PI * 2.0f * BRDF * Ei;
}
```

Path Tracing



Path Tracing



Today’s Agenda:

- Introduction
- Path Tracing

INFOMAGR – Advanced Graphics

Jacco Bikker - November 2022 - February 2023

END of “Path Tracing”

next lecture: “Acceleration Structures”