INFOMAGR – Advanced Graphics

Jacco Bikker - November 2016 - February 2017

Lecture 7 - “Path Tracing”

Welcome!

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

Today's Agenda:

- Introduction
- Path Tracing

Introduction

Previously in Advanced Graphics

The Rendering Equation:

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

...which models light transport as it happens in the real world, by summing:

- Direct illumination: $L_E(x,\omega_o)$
- Indirect illumination, or reflected light: $\int_\Omega f_r(x,\omega_o,\omega_i)\, L_i(x,\omega_i) \cos\theta_i \; d\omega_i$

We used quantities flux Φ (joules per second), radiance L (flux per m^2 per sr) and irradiance E (joules per second per m^2).

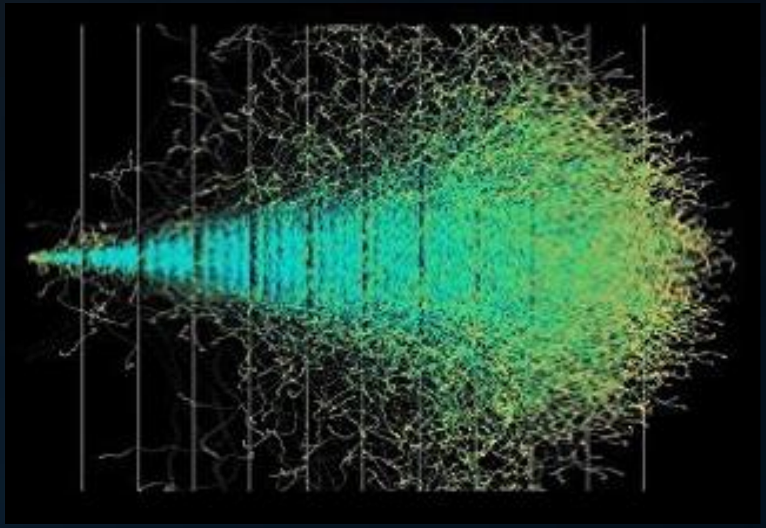
Introduction

Previously in Advanced Graphics

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

Previously in Advanced Graphics

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

Previously in Advanced Graphics

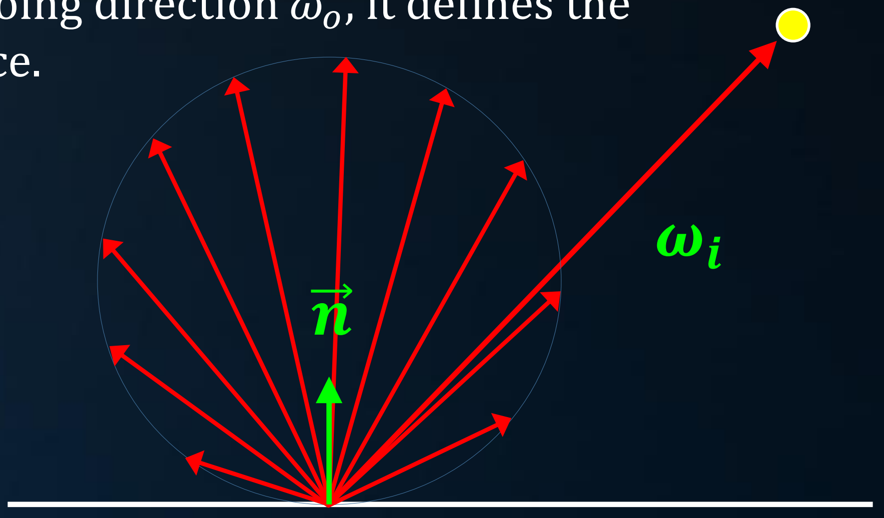
BRDFs:

The BRDF describes how incoming light is absorbed or scattered.

More accurately: for an incoming direction ω_i and an outgoing direction ω_o, it defines the relation between received irradiance and reflected radiance.

A physically based BRDF has some important properties:

- $f_r(\omega_o, \omega_i) \geq 0$
- $f_r(\omega_o, \omega_i) = f_r(\omega_i, \omega_o)$
- $\int_\Omega f_r(\omega_o, \omega_i) \cos\theta_o \, d\omega_o \leq 1$



Example: diffuse BRDF, $f_r(\omega_o, \omega_i) = \frac{albedo}{\pi}$.

Note that the diffuse BRDF is view independent; ω_o is irrelevant.
It does however take into account ω_i: this affects how radiance is converted to irradiance.

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

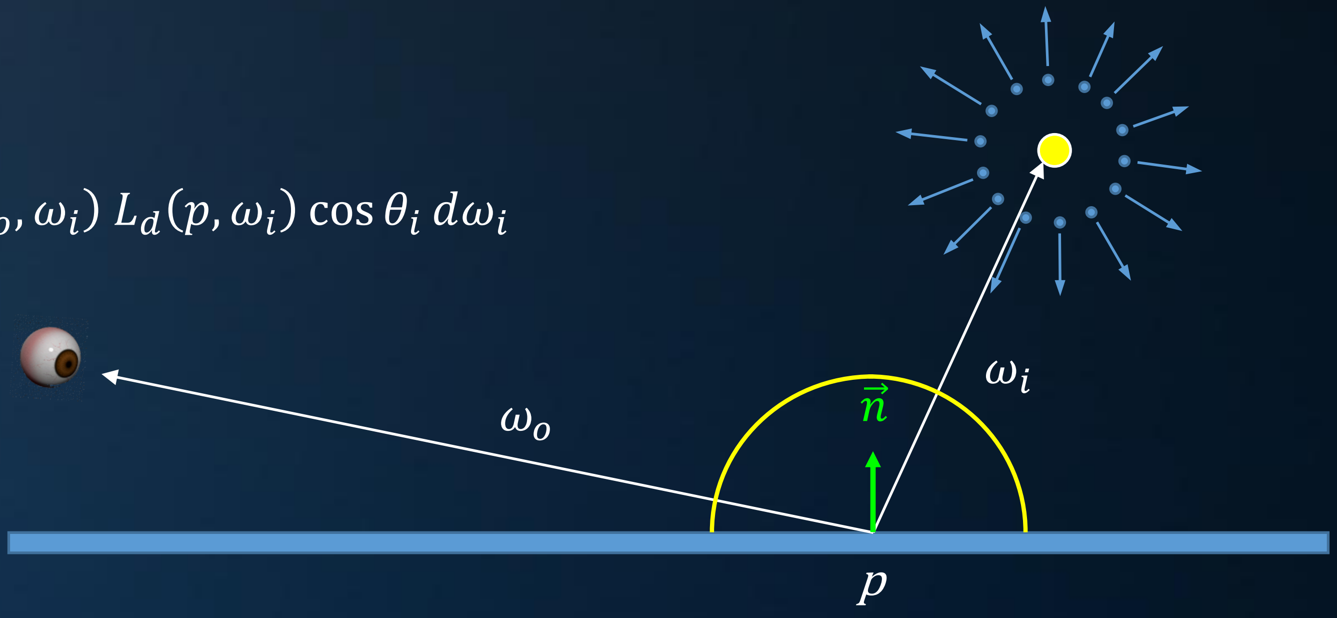
$$L_o(x,\omega_o) = L_E(x,\omega_o) + \int_\Omega f_r(x,\omega_o,\omega_i)\, L_i(x,\omega_i) \cos\theta_i \; d\omega_i$$

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

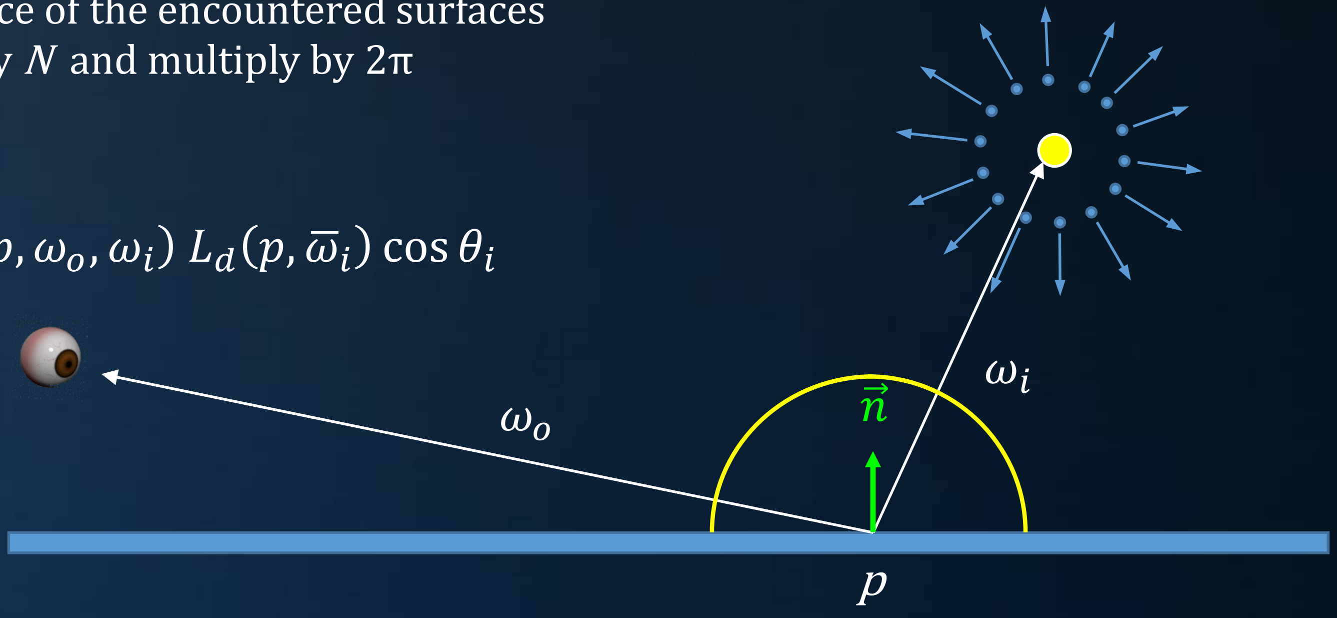
Direct Illumination

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

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
- Sum the luminance of the encountered surfaces
- Divide the sum by N and multiply by 2π

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



Path Tracing

Direct Illumination

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

Questions:

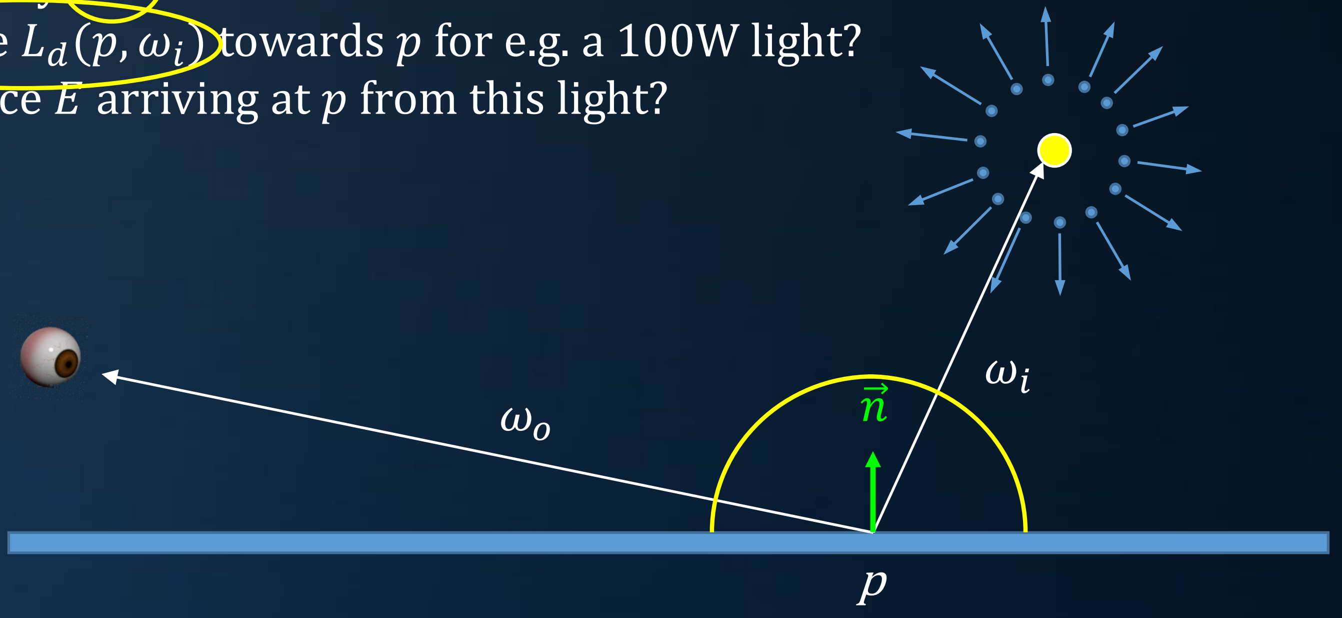
- Why do we multiply by 2π?
- What is the radiance $L_d(p,\omega_i)$ towards p for e.g. a 100W light?
- What is the irradiance E arriving at p from this light?

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).

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

Note that the 100W flux is spread out over the area; irradiance is defined per m^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:

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

Here, C compensates for the fact that we now sample the area of the light source, instead of the hemisphere. The probability of stumbling onto an unoccluded light used to be proportional to solid angle; now it is always 1. C is therefore $\sim(\cos\theta_o A_{L_d})/r^2$.

Path Tracing

Direct Illumination

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

Using Monte-Carlo:

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

where

- $L_d^{\bar{J}}$ is the direct light to p from random point $\bar{p}'$ on random light $\bar{J}$
- $A_{L_d^{\bar{J}}}$ is the area of this light source
- $V(p \leftrightarrow \bar{p}')$ is the mutual visibility between p and p'.

Path Tracing

Direct Illumination

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

1. By random sampling the hemisphere:

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

2. By sampling the lights directly:

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

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:

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

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

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

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

Path Tracing

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

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

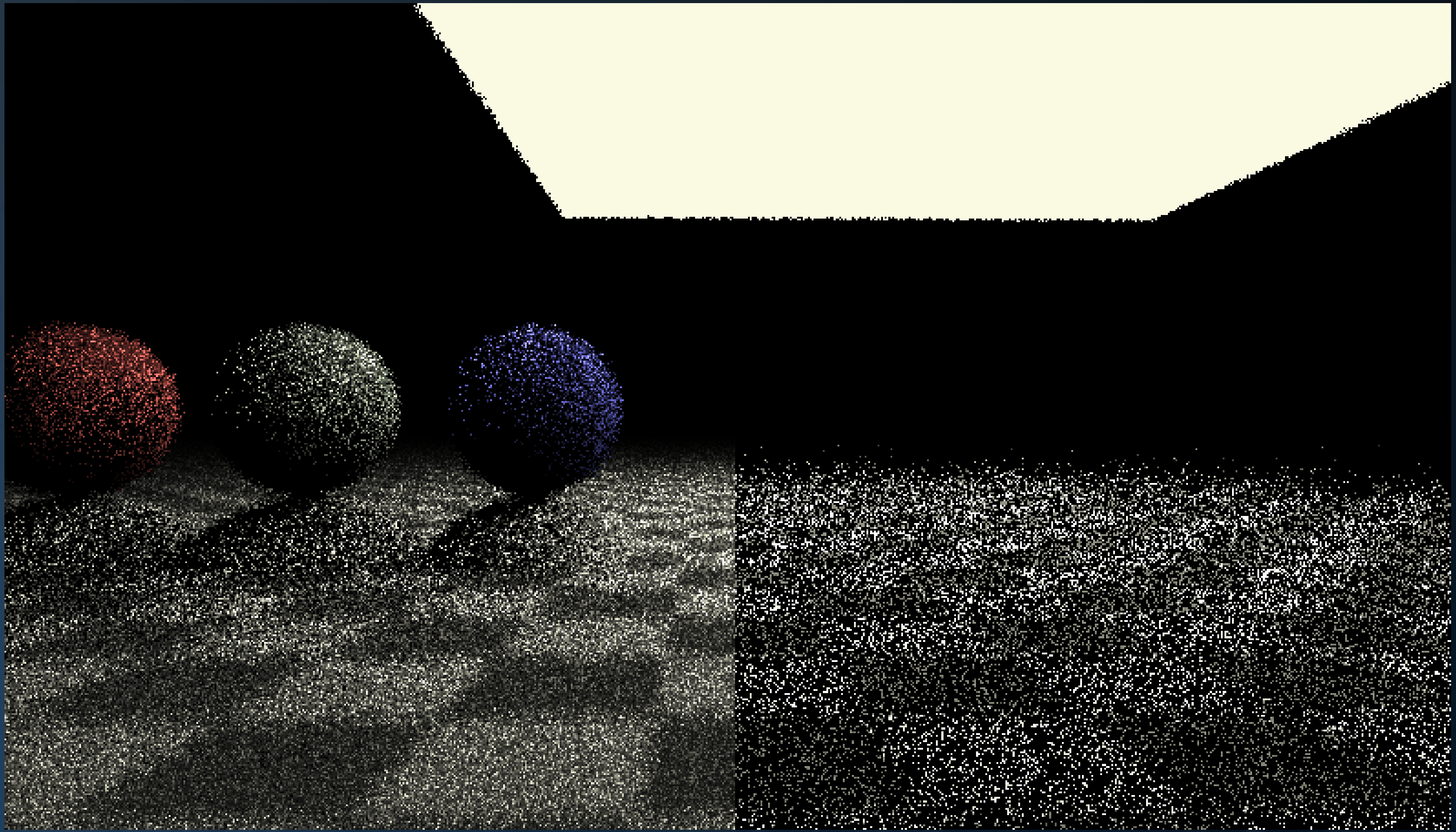
$$L_o(p,\omega_i) \approx lights * \frac{1}{N}\sum_{i=1}^{N} f_r(p,\omega_o,p')\, L_d^{\bar{j}}(p,\bar{p}')\, V(p \leftrightarrow \bar{p}')\frac{A_{L_d^{\bar{j}}}\cos\theta_i\cos\theta_o}{\| p-\bar{p}' \|^2}$$

Verification

Method 2 in a small C# ray tracing framework:

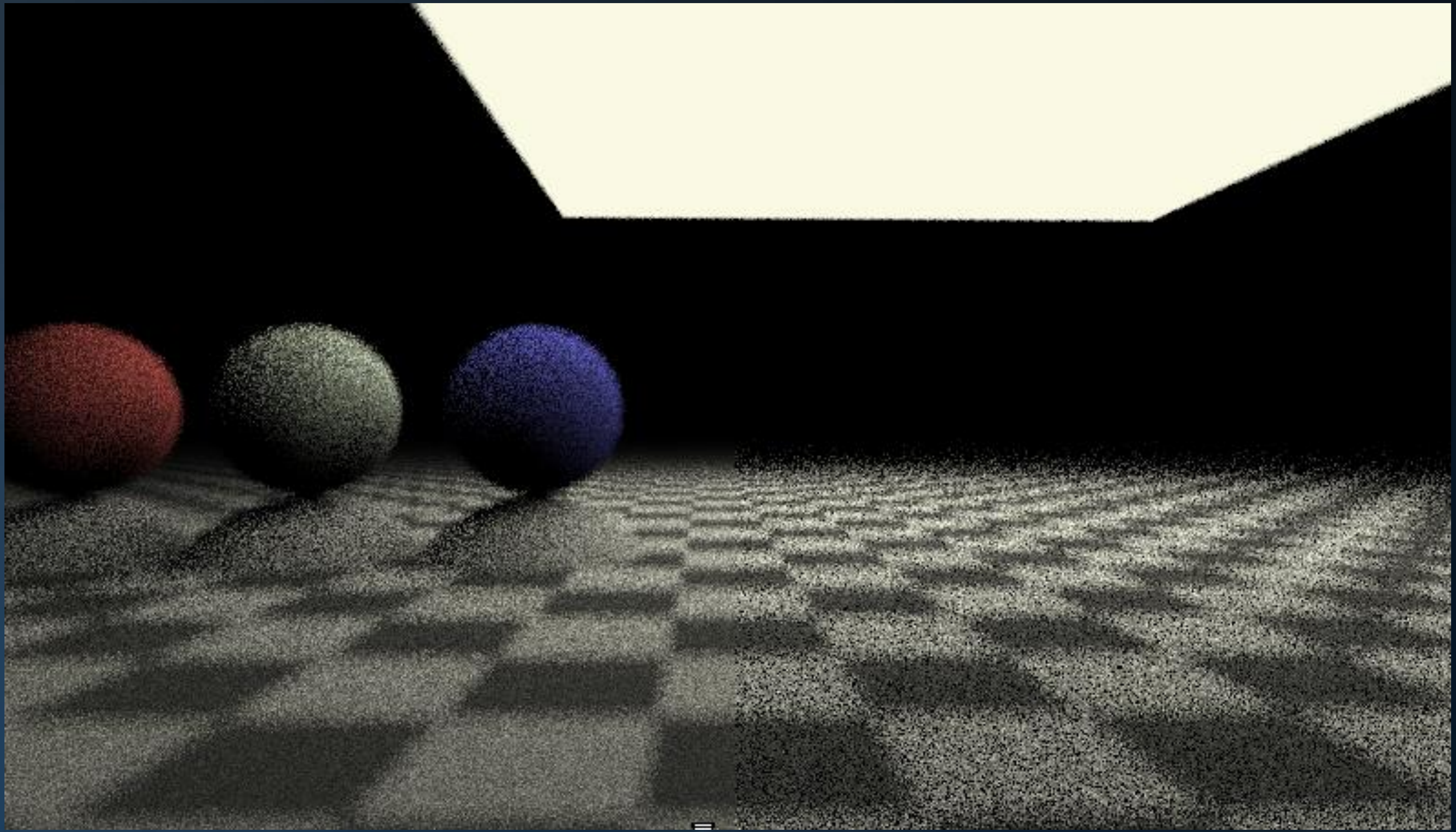
```
// construct vector to random point on light
Vector3 L = Scene.RandomPointOnLight() - I;
float dist = L.Length();
L /= dist;
float cos_o = Vector3.Dot( -L, lightNormal );
float cos_i = Vector3.Dot( L, ray.N );
if ((cos_o <= 0) || (cos_i <= 0)) return BLACK;
// light is not behind surface point, trace 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



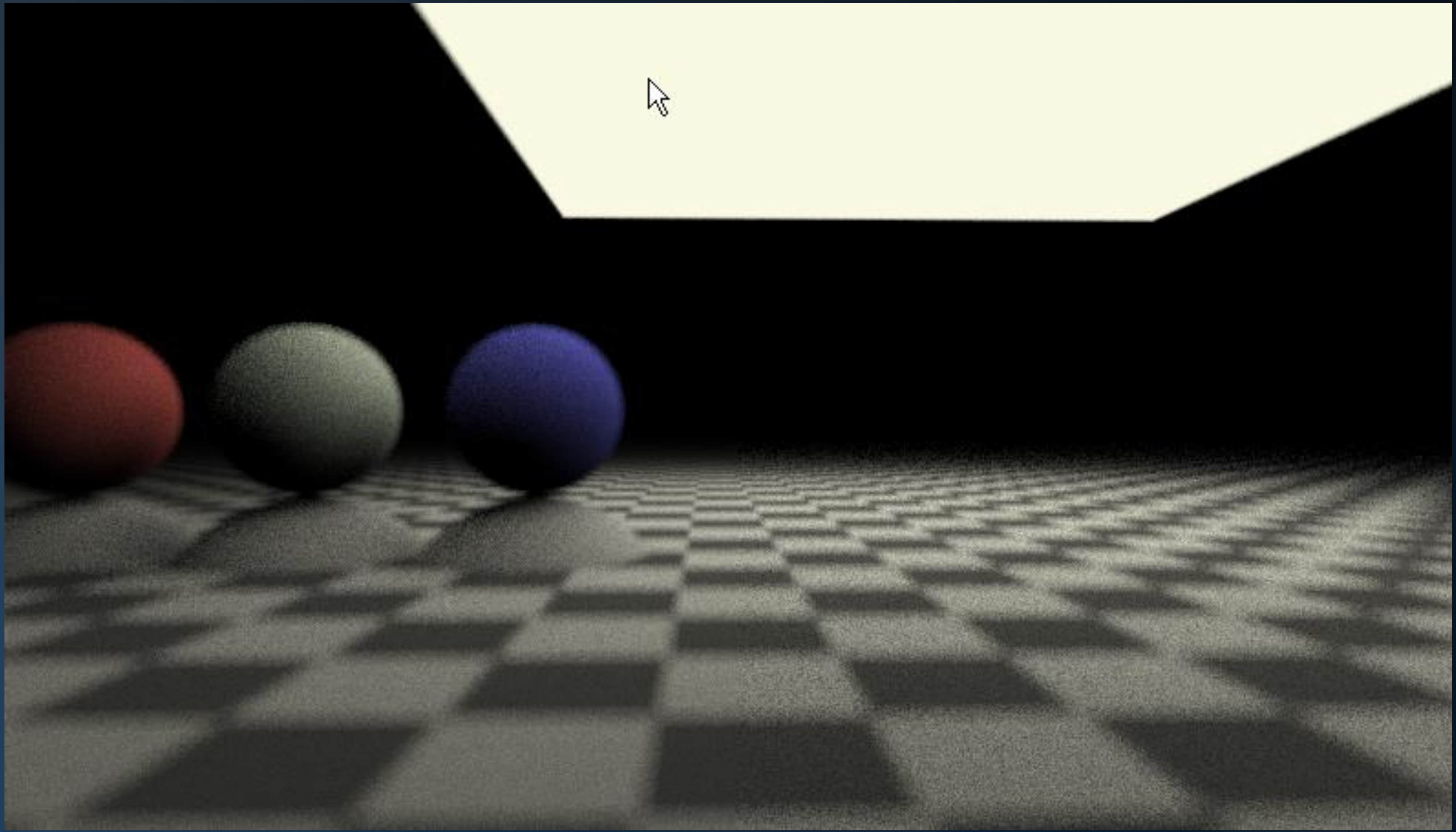
0.1s

Path Tracing



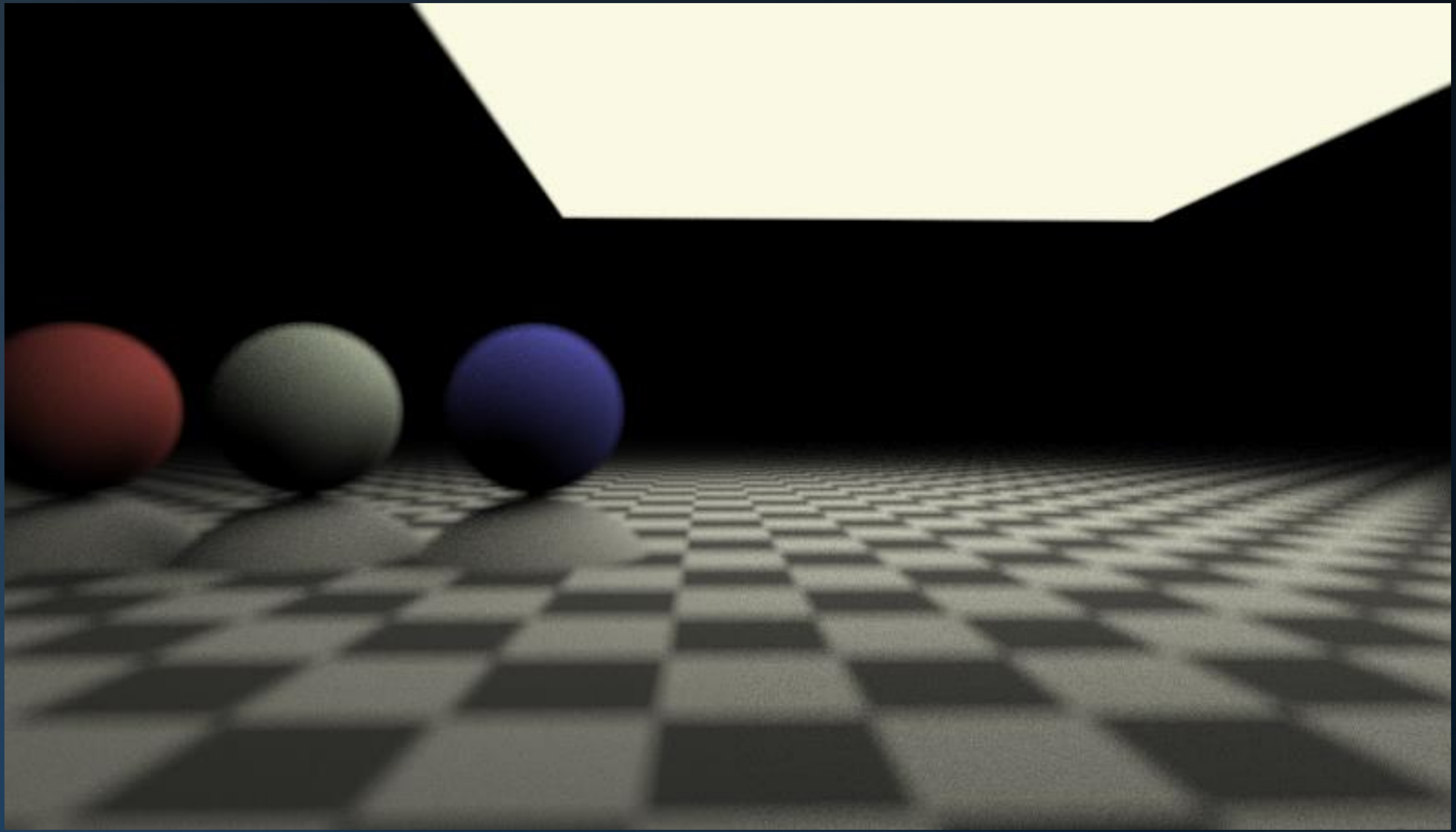
0.5s

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:

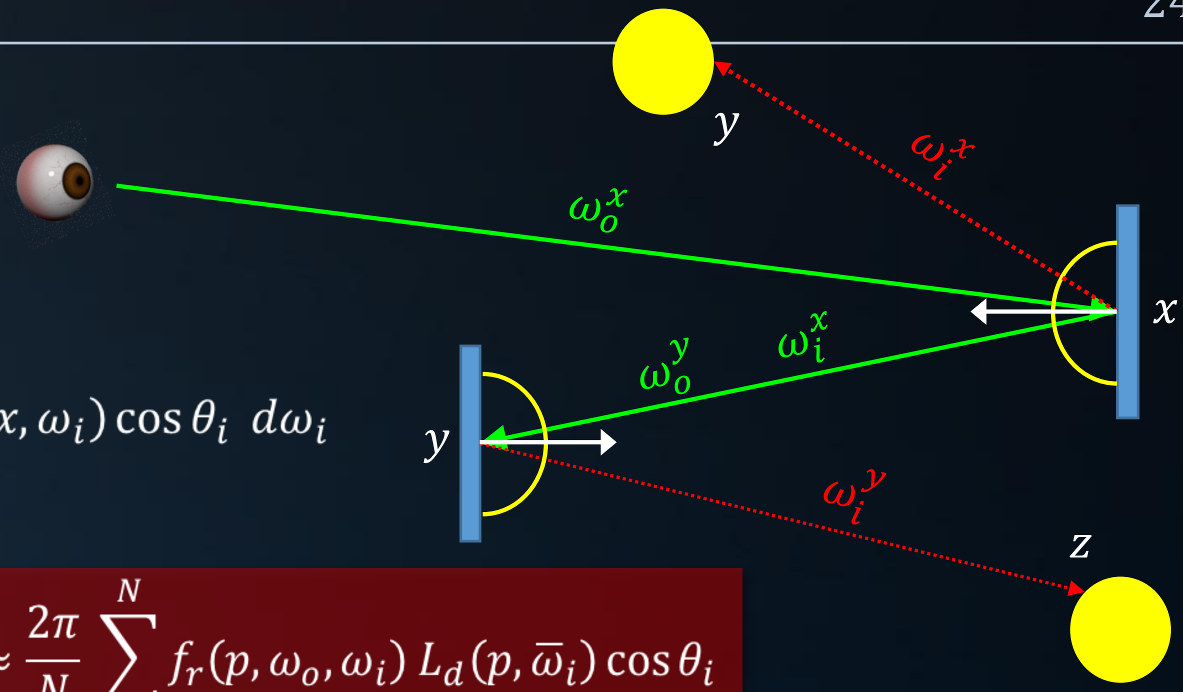
$$L_o(x,\omega_o) = L_E(x,\omega_o) + \int_\Omega f_r(x,\omega_o,\omega_i)\, L_i(x,\omega_i) \cos\theta_i \; d\omega_i$$

We know how to evaluate direct lighting:

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

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:

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

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

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

direct light

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

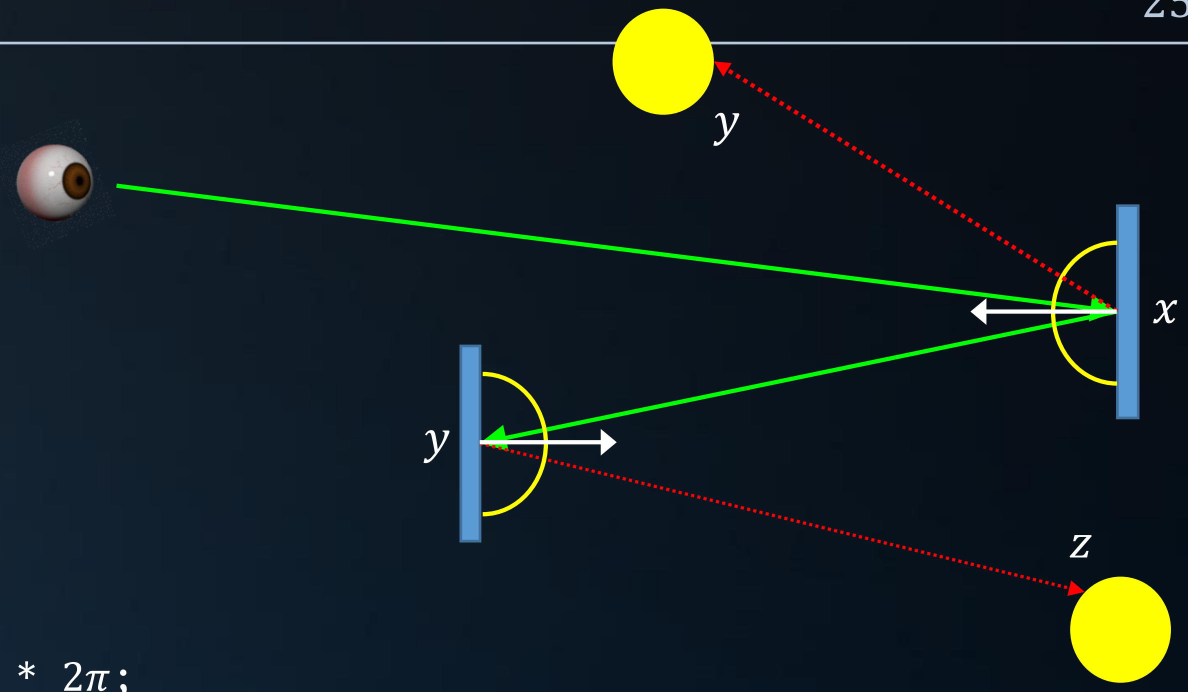
1st bounce

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

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 ) );
```

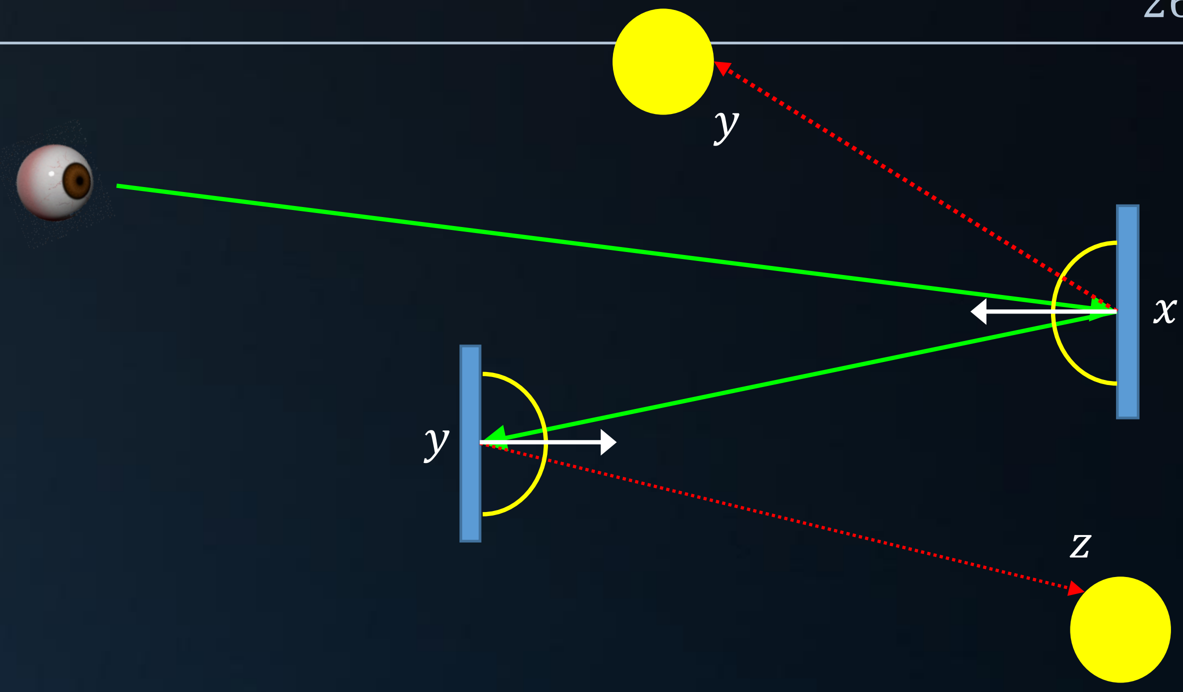
$$\texttt{return dot(R, N) * } \frac{albedo}{\pi} \texttt{ * lightColor * } 2\pi;$$

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 ) );
```

$$\texttt{return dot(R1, N1) * } \frac{albedo}{\pi} * 2\pi \texttt{ * dot(R2, N2) * } \frac{albedo}{\pi} * 2\pi \texttt{ * lightColor;}$$

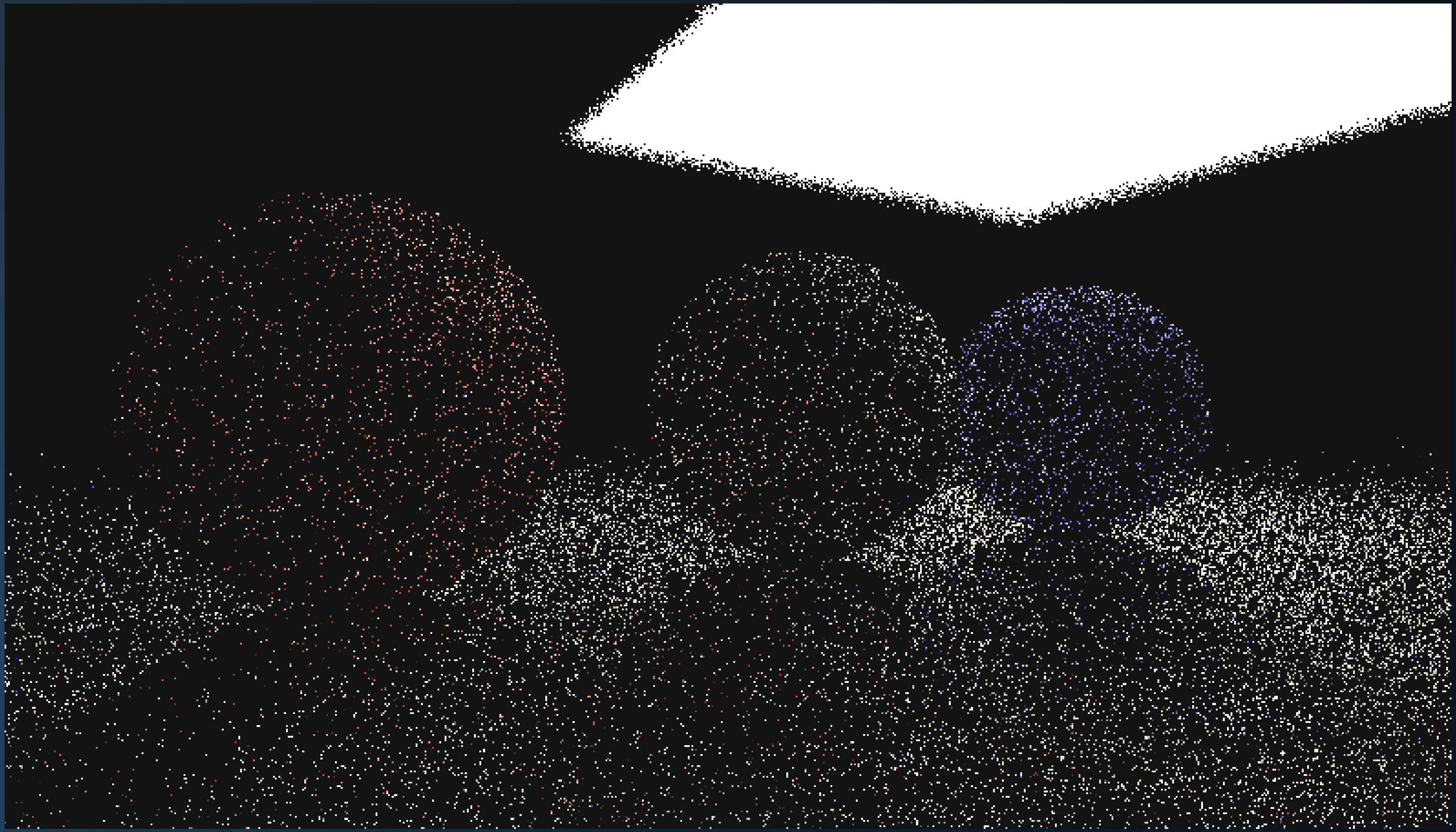
Path Tracing



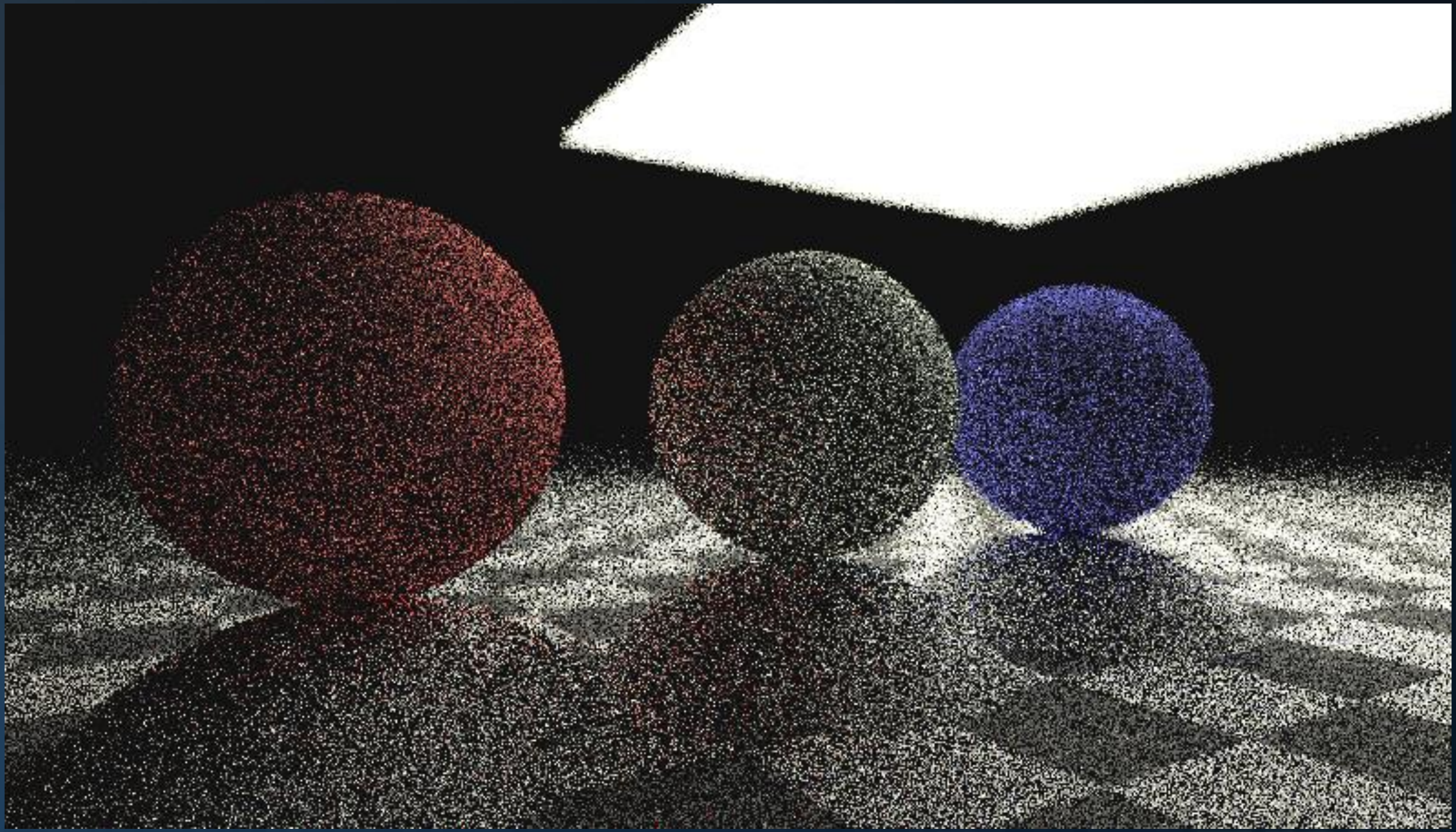
Path Tracing Algorithm

```
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;
}
```

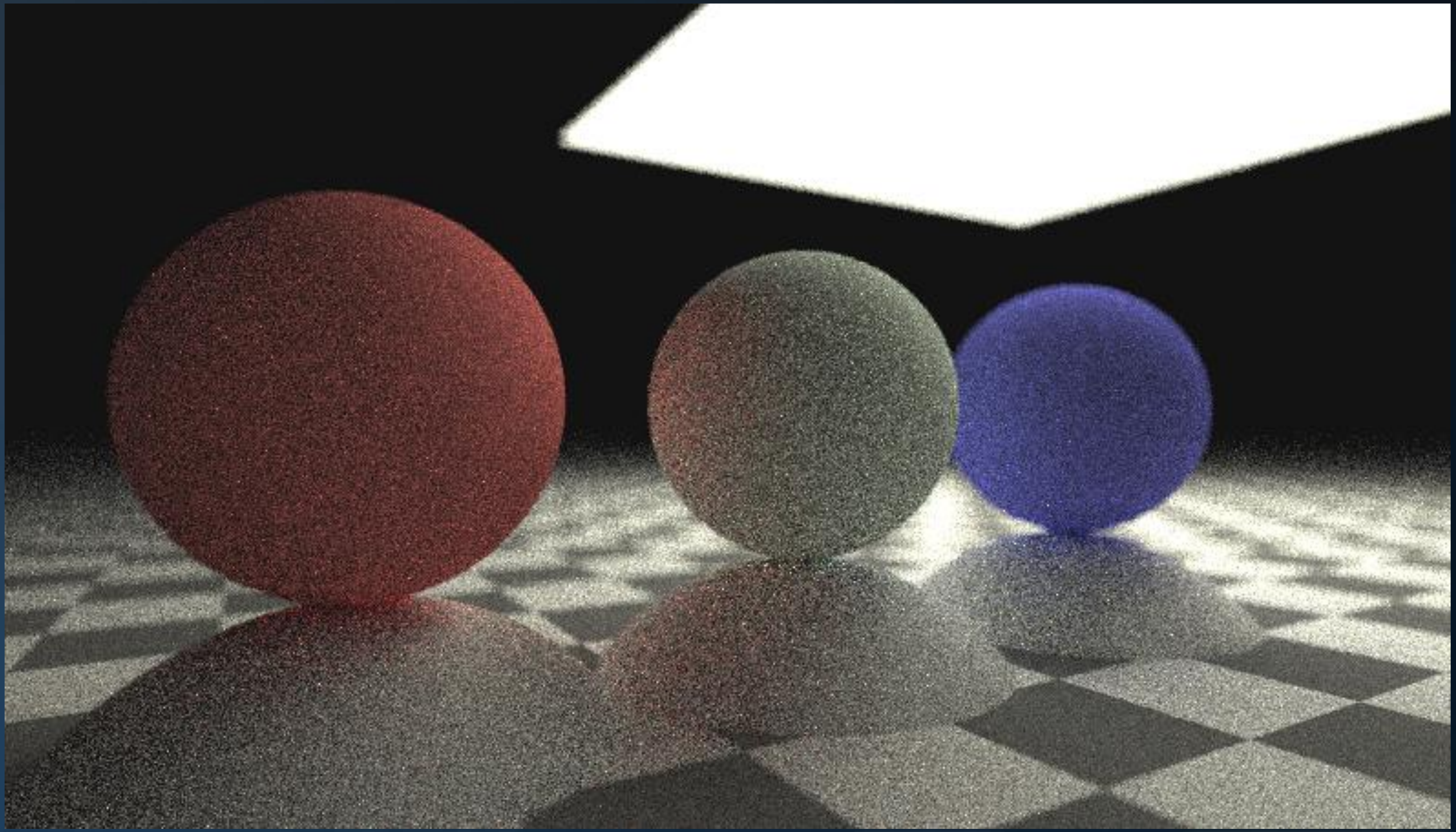
Path Tracing



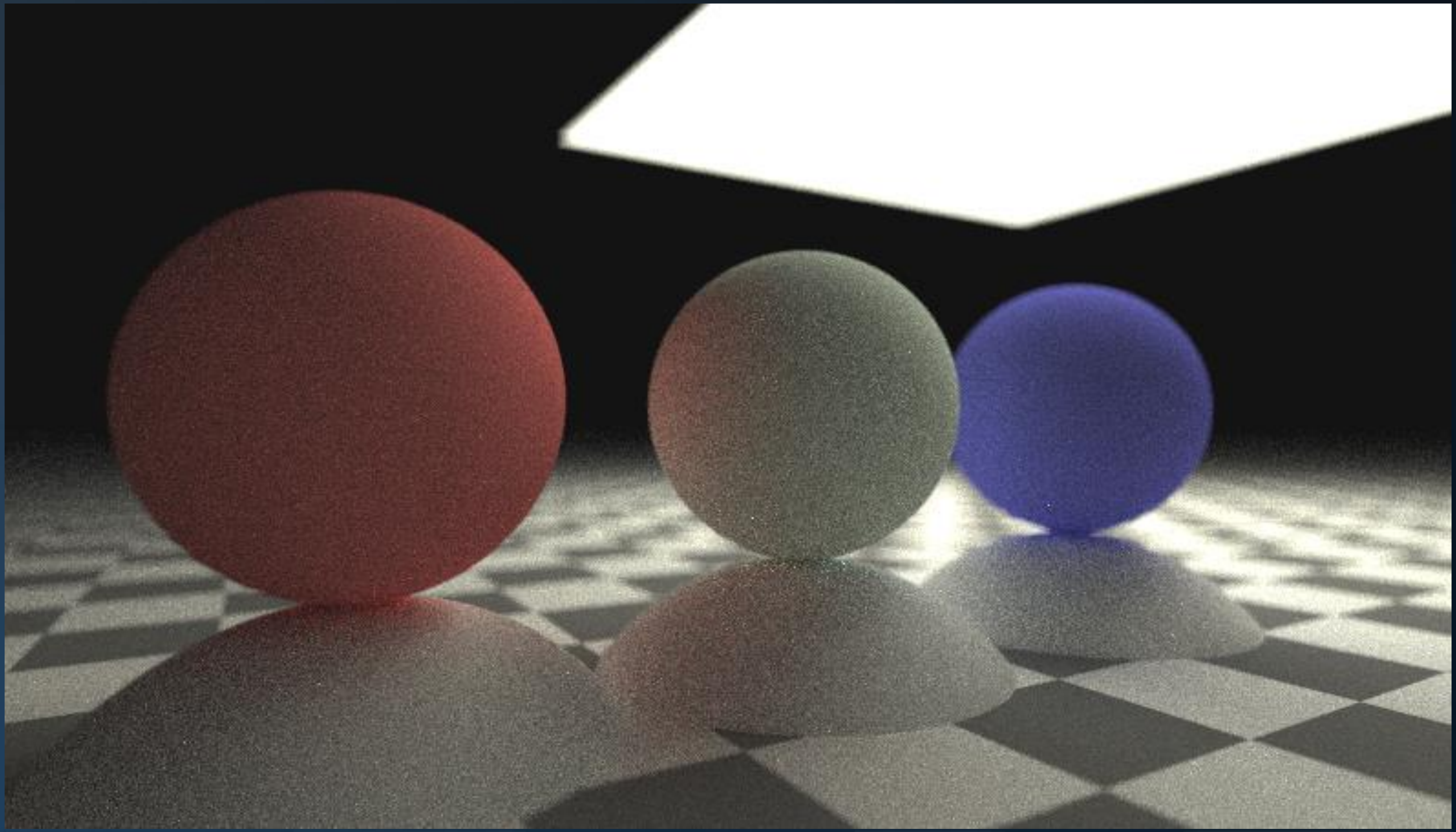
Path Tracing



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.

Question:

- What happens at a red mirror?
- What happens if a material is only half reflective?

```
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:

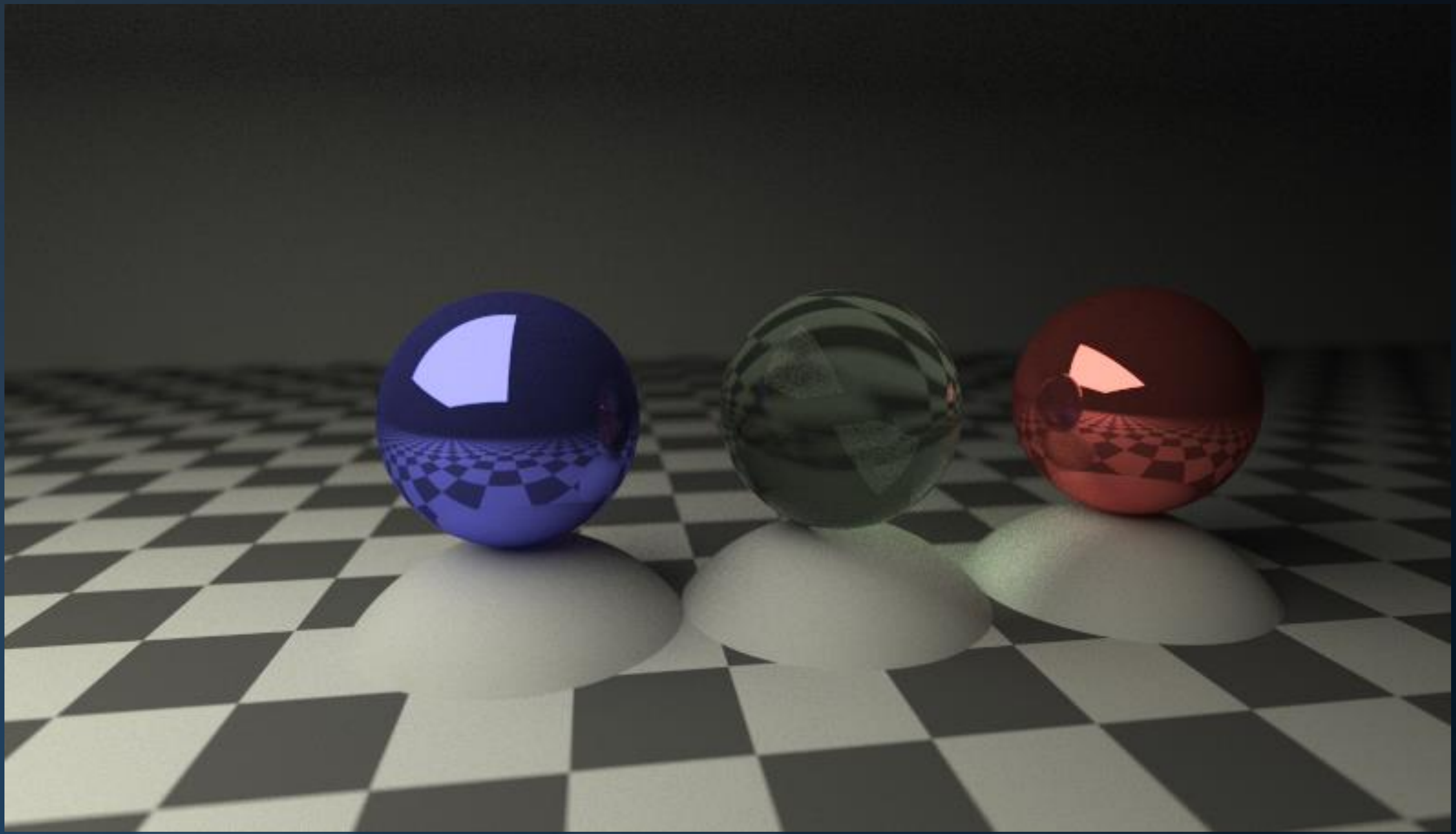
Dielectrics reflect *and* transmit light.
In the ray tracer, we handled this using two rays.

A particle must chose.

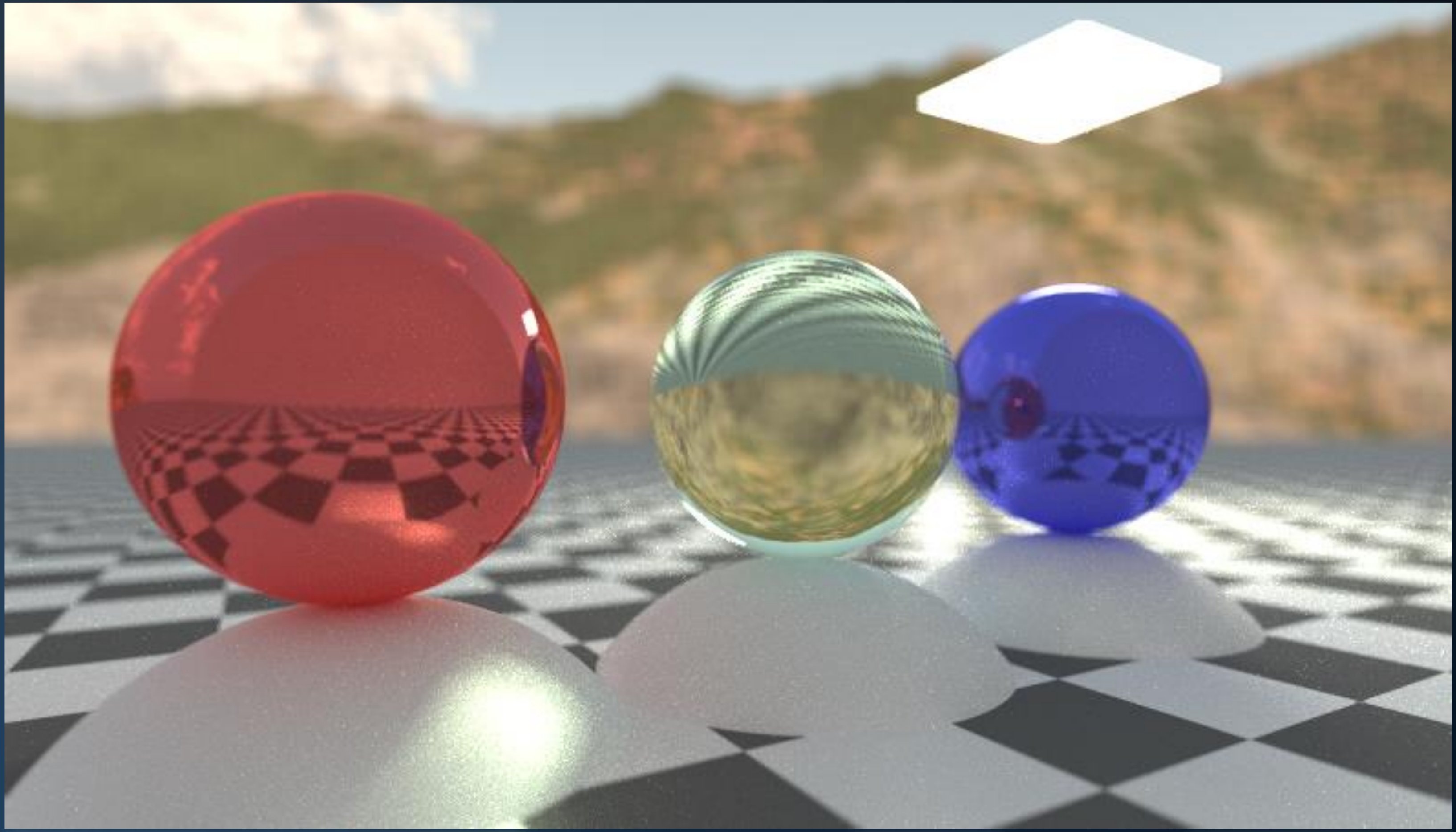
The probability of each choice is calculated using the Fresnel equations.

```
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 2016 - February 2017

END of “Path Tracing”

next lecture: “Variance Reduction”

Path Tracing

Solid Angle

A few words on solid angle:

As mentioned in lecture 5, solid angle is measured in steradians. Where radians are the length of an arc on the unit circle subtended by two directions, steradians are the surface on the unit sphere subtended by a shape.

In this lecture, we used: $\frac{A \cos\theta}{r^2}$, which is only an approximation, except when the shape is a segment of a sphere and $\cos\theta = 1$. For $r \gg A$, this approximation is 'good enough'.

Path Tracing

Additional literature:

Slides for lecture 9 of the Advanced Computer Graphics course of the University of Freiburg, by Matthias Teschner:
http://cg.informatik.uni-freiburg.de/course_notes/graphics2_09_pathTracing.pdf

Blog: A graphics guy's note:
https://agraphicsguy.wordpress.com

Monte Carlo Path Tracing, Path Hanrahan:
http://cs.brown.edu/courses/cs224/papers/mc_pathtracing.pdf

Path Tracing - Theoretical Foundation, by Vidar Nelson:
http://www.vidarnel.com/post/path_tracing

Importance Sampling for Production Rendering, SIGGRAPH 2010 course:
http://igorsklyar.com/system/documents/papers/4/fiscourse.comp.pdf

Please share additional resources on the forum.