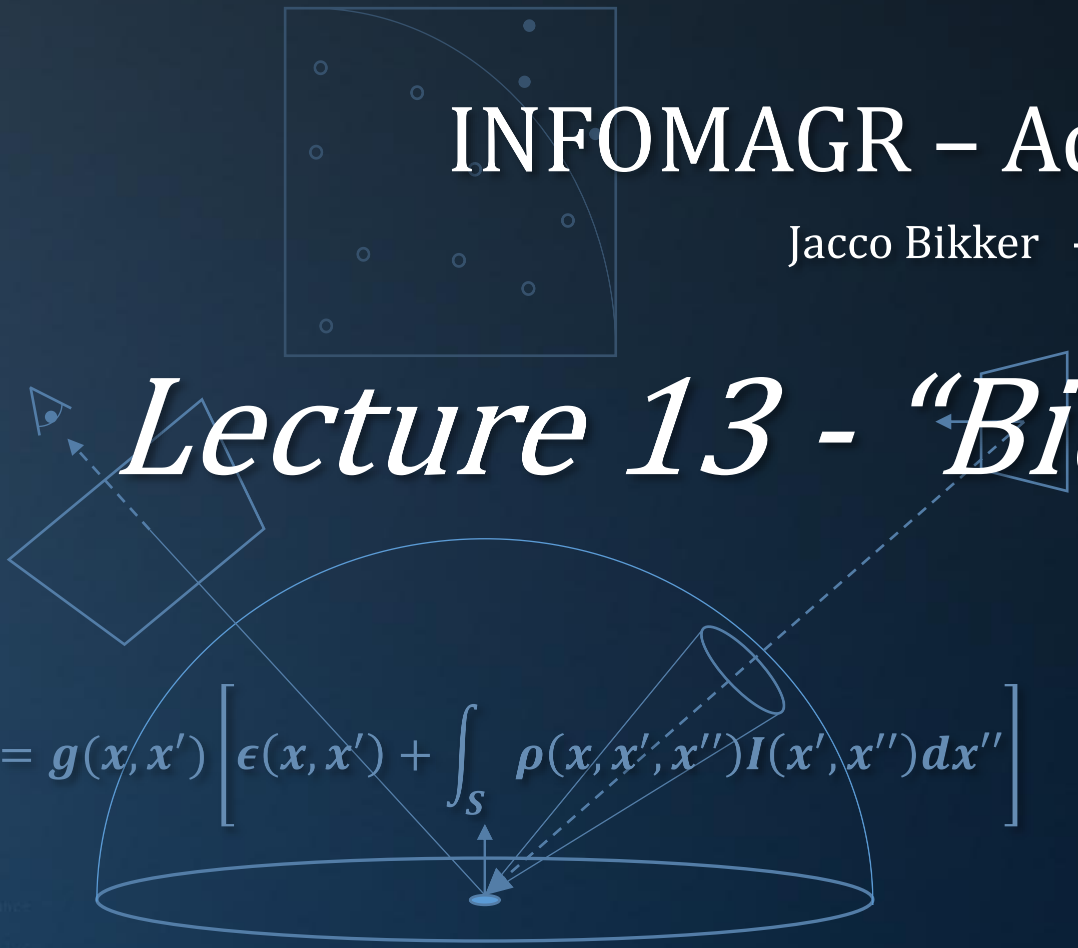
INFOMAGR – Advanced Graphics

Jacco Bikker - November 2021 - February 2022

Lecture 13 - "Bidirectional"

Welcome!

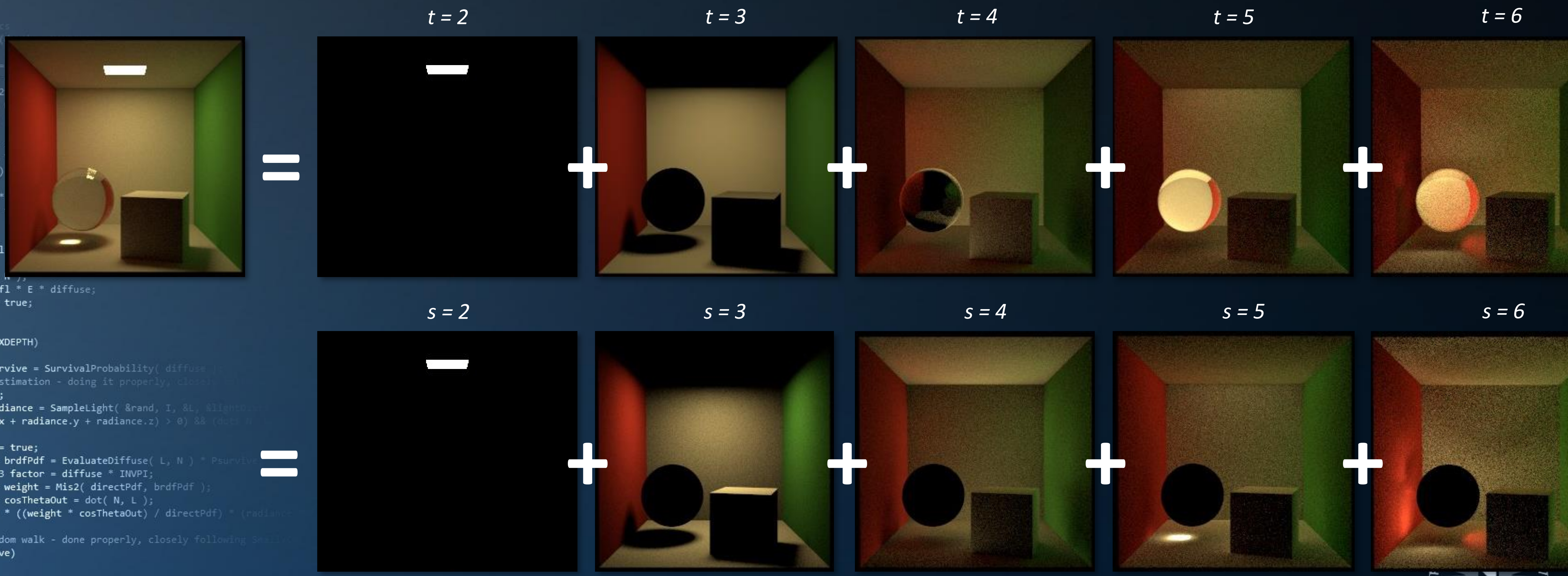

$I(x,x') = g(x,x')\left[\epsilon(x,x') + \int_S \rho(x,x',x'')I(x',x'')dx''\right]$

Today’s Agenda:

- Recap: Forward Path Tracing
- Virtual Point Lights
- Photon Mapping
- Bidirectional Path Tracing
- More

Forward

Backward and Forward Path Tracing



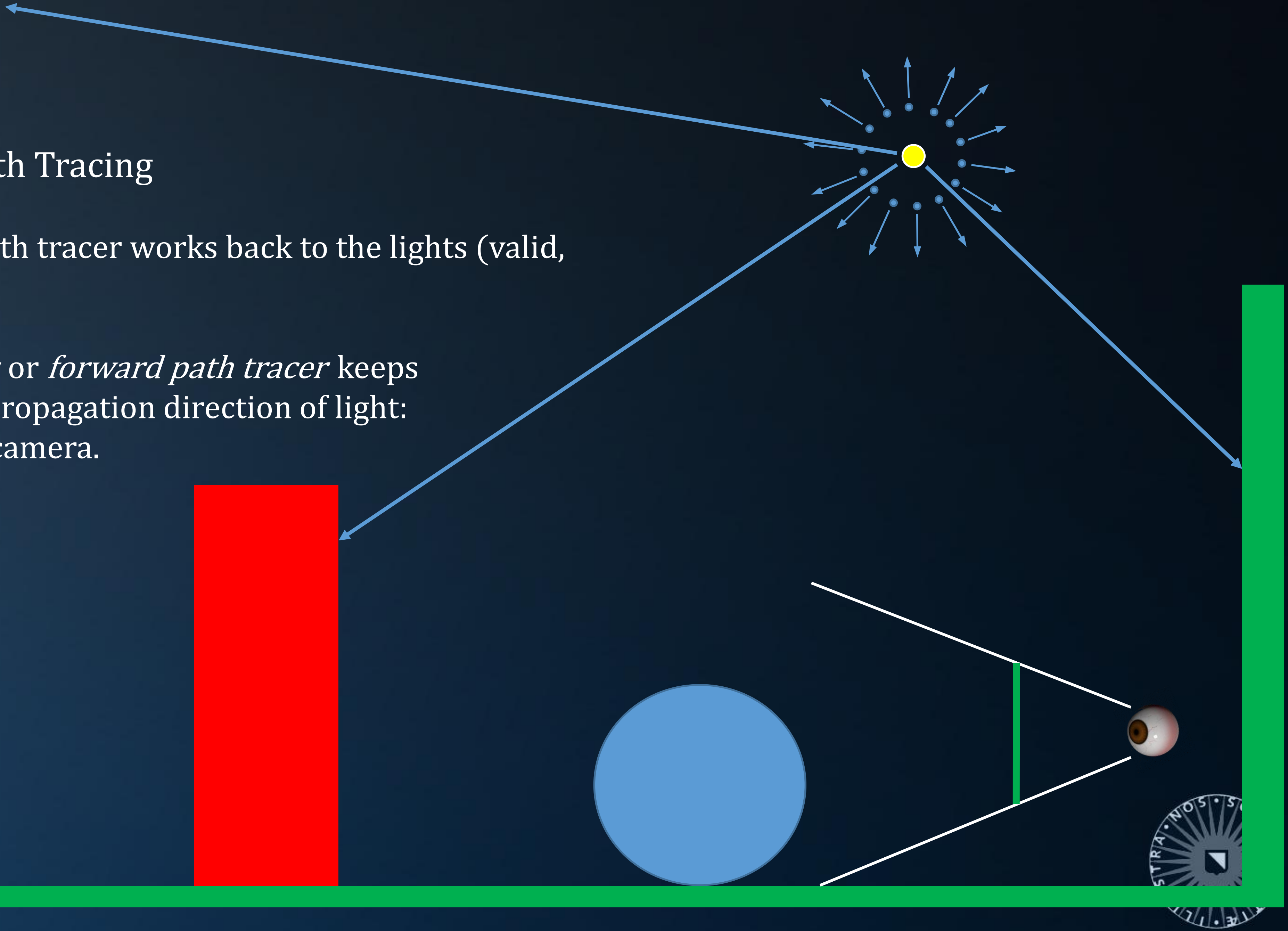
Images: Simon Brown, sjbrown.co.uk/2011/01/03/two-way-path-tracing

Forward

Forward Path Tracing

A ‘normal’ path tracer works back to the lights (valid, Helmholtz).

A *light tracer* or *forward path tracer* keeps the original propagation direction of light: towards the camera.



Forward

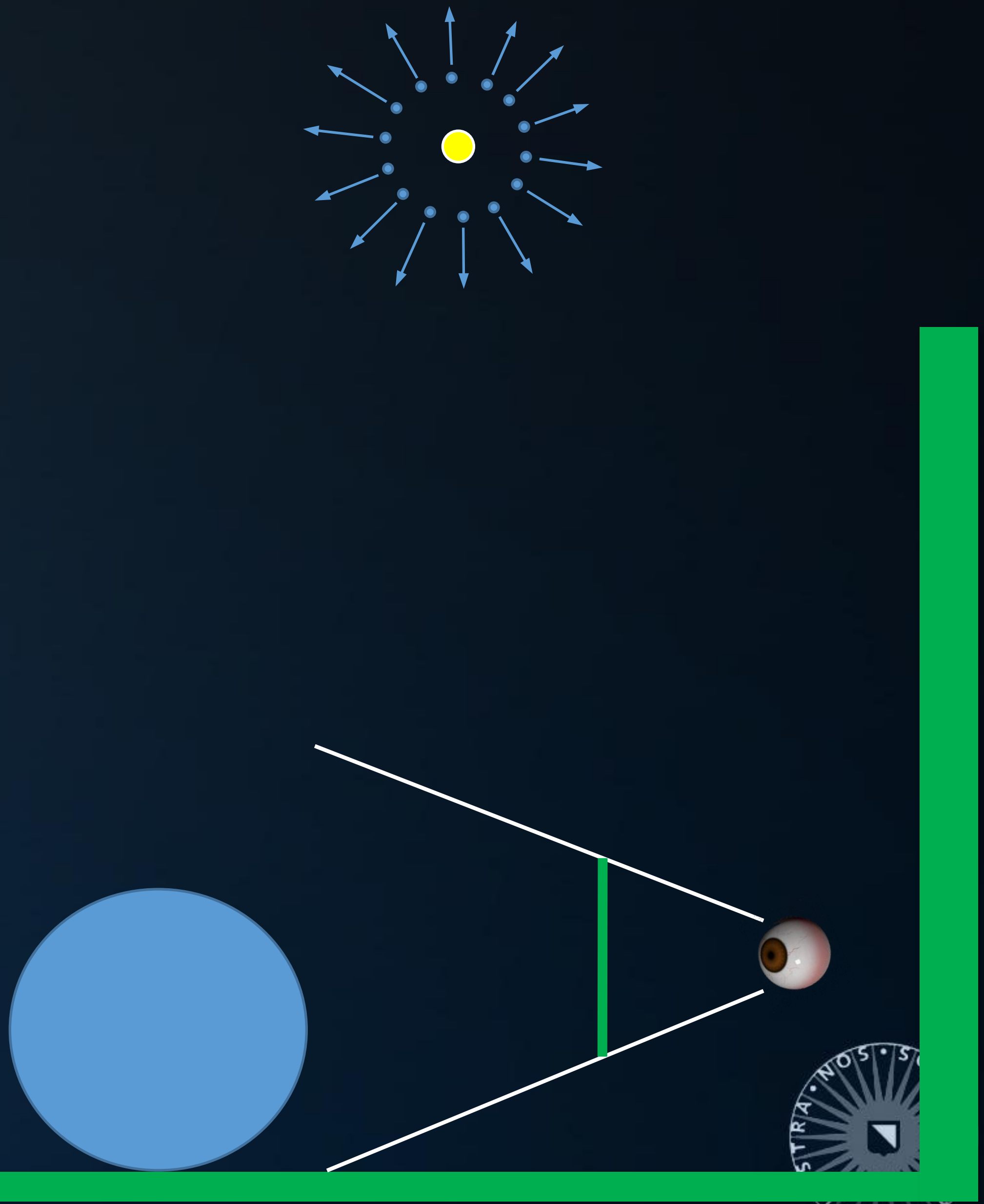
Forward Path Tracing

A ‘normal’ path tracer works back to the lights (valid, Helmholtz).

A *light tracer* or *forward path tracer* keeps the original propagation direction of light: towards the camera.

Consequences / issues:

1. ‘Eye’ must have an area.
2. Or: use Next Event Estimation.
3. If the eye sees a mirror, it will be black.
4. This is a bad idea in an open world scene.
5. Paths hit random pixels *(however, on average…).*
6. What if the camera is behind glass?



Forward

Forward Path Tracing

Tracing paths from the light helps when:

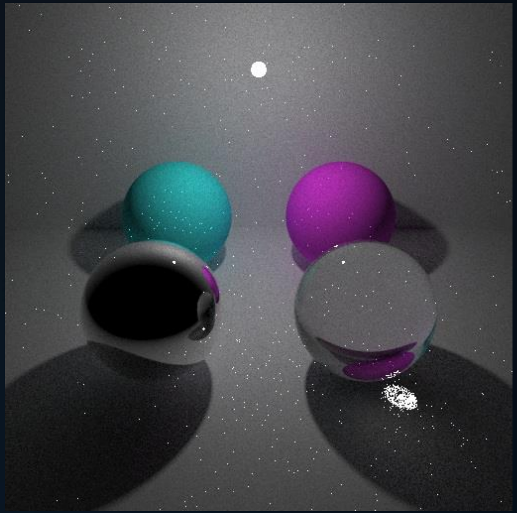
- the light is hard to reach
- the light cannot be *importance sampled* (using NEE).

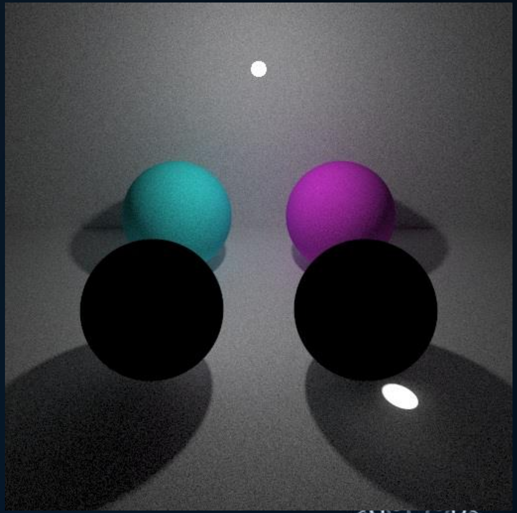
Tracing paths from the eye is better when:

- the camera is hard to reach.

Many scenes would benefit from both approaches. Now what?

- decide on a per-pixel basis?
- do both, and average? (would that even work?)
- something smarter?





Forward

Forward Path Tracing

Tracing paths from the light helps whe

- the light is hard to reach
- the light cannot be *importance sam*

So… a forward path tracer cannot correctly render a scene in which the camera directly views pure specular objects.

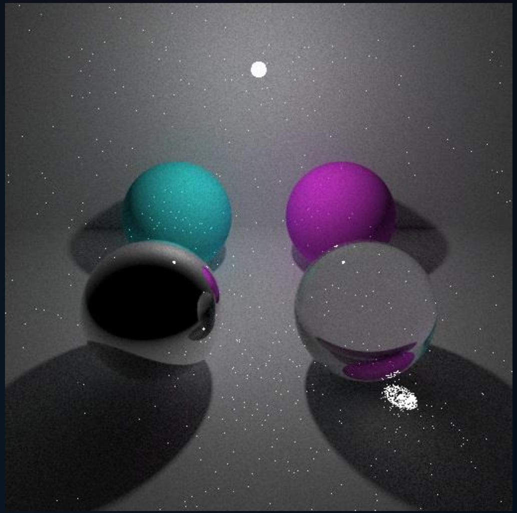
Is it possible to construct a scene that cannot be correctly rendered using a backward path tracer?

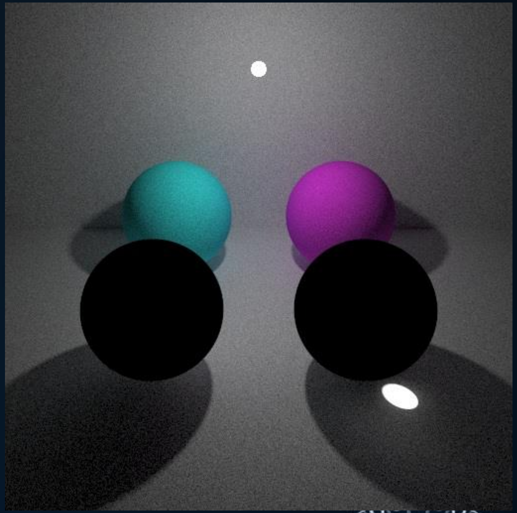
Tracing paths from the eye is better when:

- the camera is hard to reach.

Many scenes would benefit from both approaches. Now what?

- decide on a per-pixel basis?
- do both, and average? (would that even work?)
- something smarter?





Forward

Forward Path Tracing

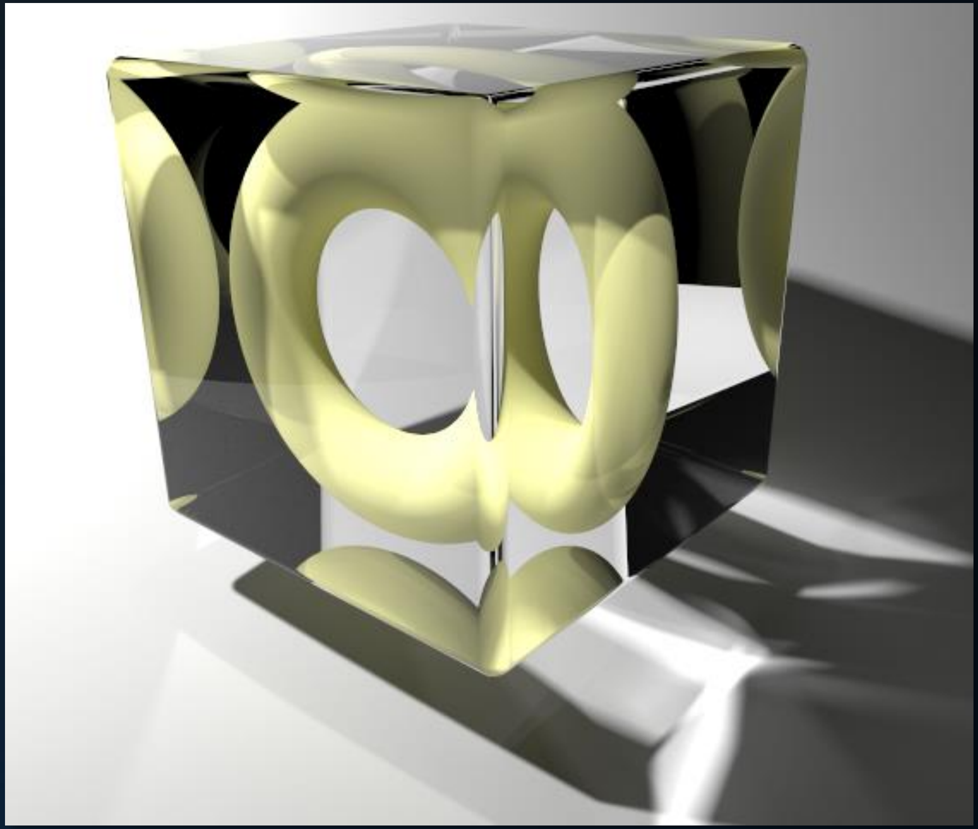
The problem with this scene:

- When a path hits the torus, it can’t use NEE
- This is true for light tracing and path tracing.

The problematic paths are SDS paths*:

E: eye
D: diffuse
S: specular
L: light

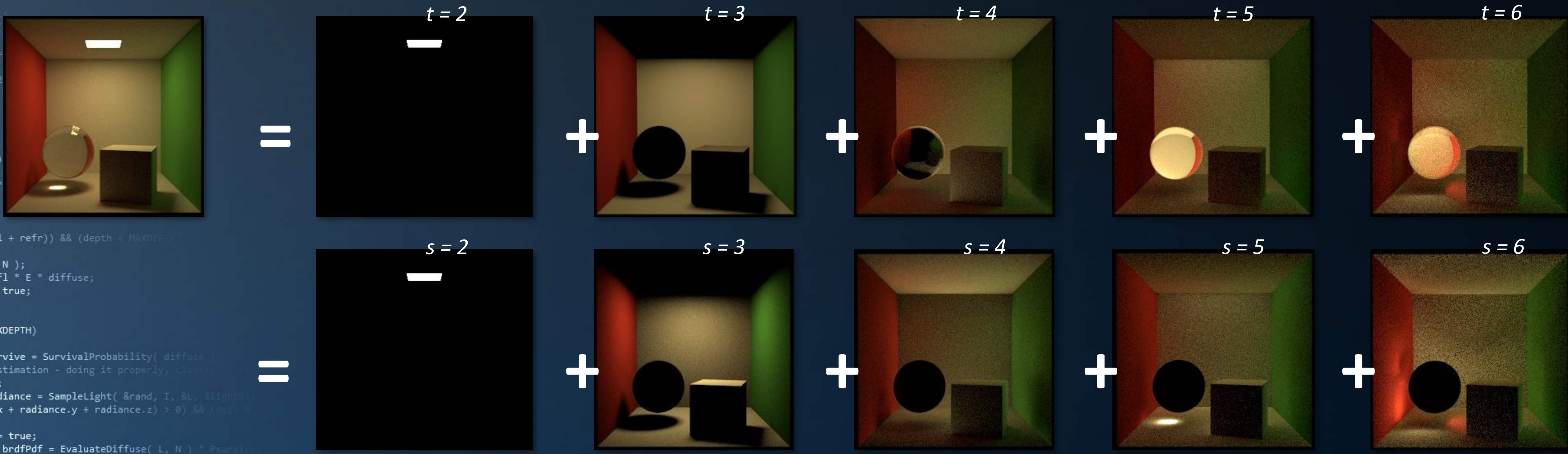
(a light tracer fails on L(…)SE paths.)



*: Heckbert, Adaptive radiosity textures for bidirectional ray tracing. SIGGRAPH 1990.

Forward

Path Classification



energy returned by
$t = 2, s = 2$ paths
EL - LE

energy returned by
$t = 3, s = 3$ paths
EDL - LDE

energy returned by
$t = 4, s = 4$ paths
E(D|S)DL
L(D|S)DE

energy returned by
$t = 5, s = 5$ paths

energy returned by
$t = 6, s = 6$ paths

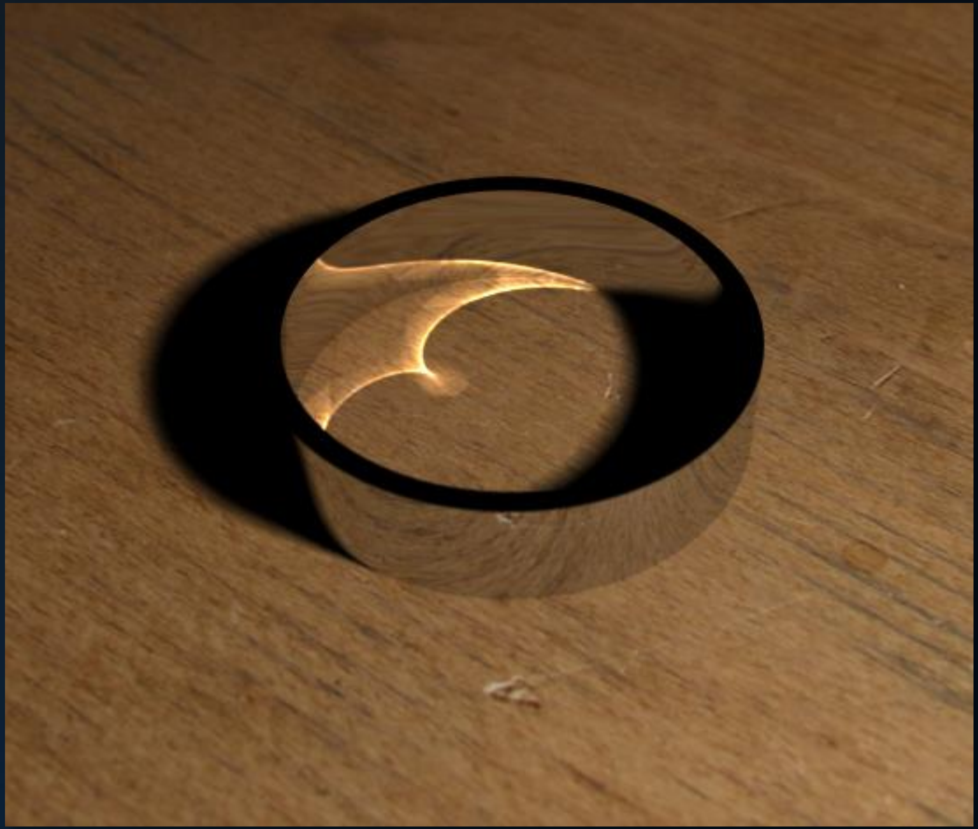
Forward

Forward Path Tracing

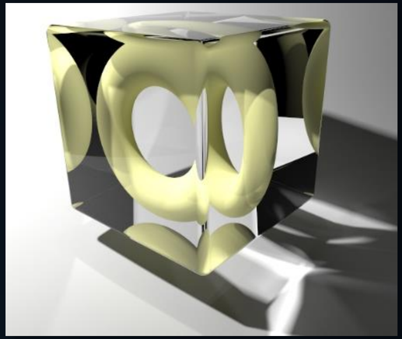
The problem with this scene:

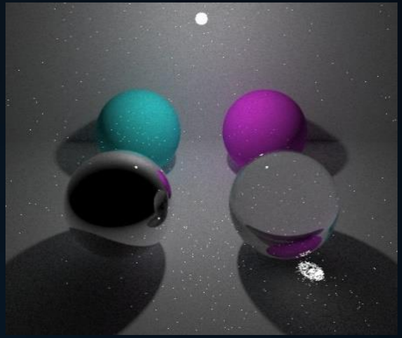
- The wood inside the ring benefits from NEE
- But sometimes much more energy arrives via the metal.

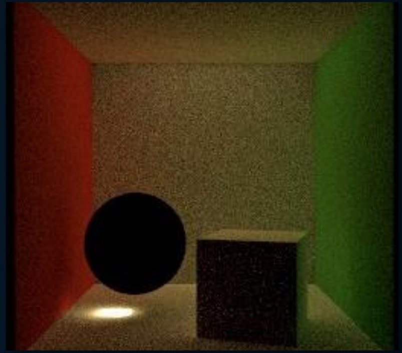
Here, NEE correctly samples the direct illumination, but the indirect illumination (via the metal) is poorly represented by the cosine pdf.



Forward







Today

Paths with high throughput and a low probability yield severe noise.

- Sometimes it’s better to trace from the light.
- Sometimes backward nor forward work well.

Bidirectional techniques aim to exploit benefits of both.

Today’s Agenda:

- Recap: Forward Path Tracing
- Virtual Point Lights
- Photon Mapping
- Bidirectional Path Tracing
- More

VPLs

Instant Radiosity*

Idea:

Trace N particles (where N is ~10^3..10^5) from the light sources, record non-specular hits.
Each recorded hit becomes a *virtual point light*.

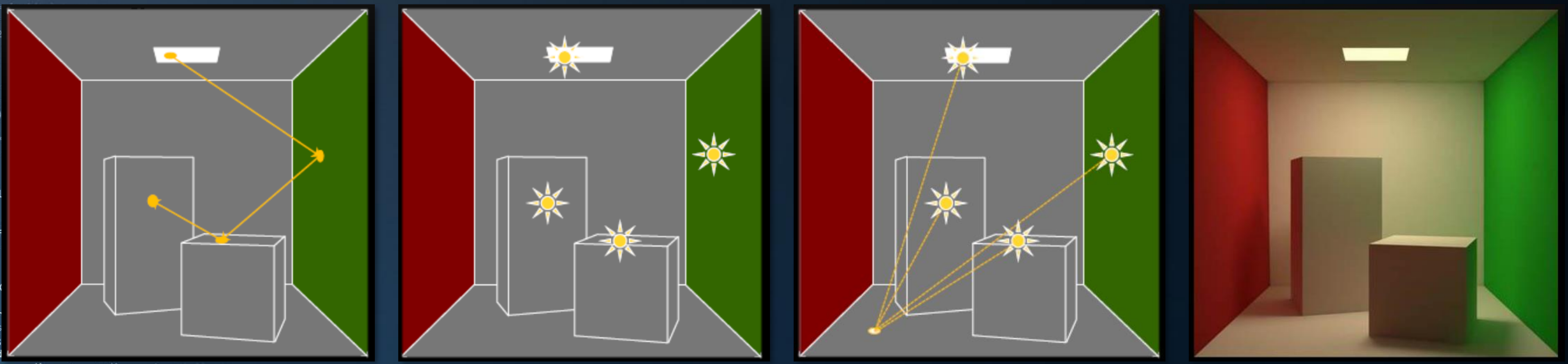
Now, render the scene with rasterization or Whitted-style ray tracing.
At the first diffuse surface, use the VPLs to estimate indirect light, and the lights themselves for direct illumination.

(did we account for all light transport?)

*: A. Keller, Instant Radiosity. SIGGRAPH ‘97.

VPLs

Instant Radiosity



Images: M. Hasan, SIGGRAPH Asia ‘09.

```
rics
& (depth < MAXDEPTH)
e.x + radiance.y + radiance.z) > 0) && (dot( N,
w = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
 E * ((weight * cosThetaOut) / directPdf) * (radiance

andom walk - done properly, closely following Small
vive)

;
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
urvive;
 pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true:
```

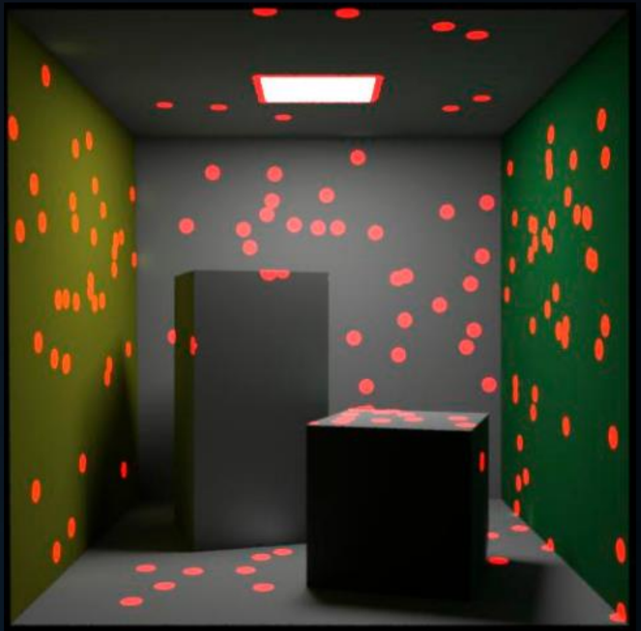
VPLs

Instant Radiosity

Using VPLs has some interesting characteristics:

- No noise! Those splotches though…
- VPLs can bounce: they can represent all indirect light
- VPLs *cannot* represent direct light
- #VPLs < #pixels
- Evaluating VPLs can be done with or without occlusion
- VPL visibility can also be evaluated using shadow maps

Instant Radiosity is a *bidirectional* technique:
we propagate **flux** when placing the VPLs,
and we propagate **importance** when connecting to them.



19232 VPLs | 356.142 VPLs/sec
Alexander Chia

Today’s Agenda:

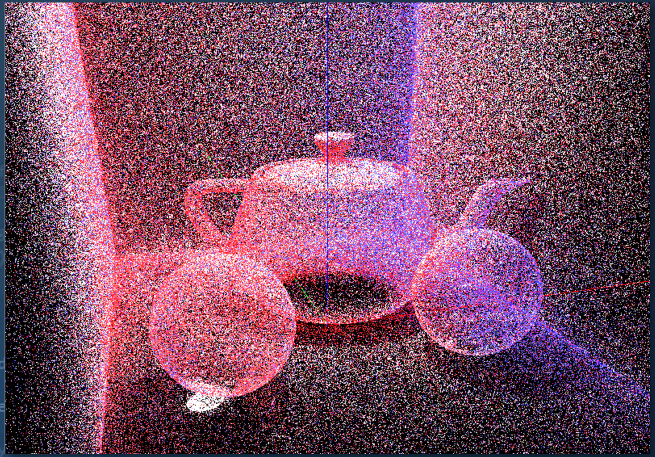
- Recap: Forward Path Tracing
- Virtual Point Lights
- Photon Mapping
- Bidirectional Path Tracing
- More

Photons

Photon Mapping*

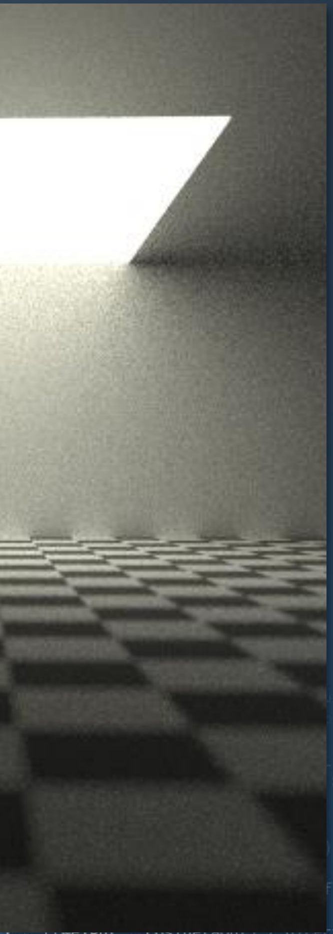
With the photon mapping algorithm, we split rendering in two phases:

- In phase 1 we deposit flux (Φ) in the scene by tracing a large number of photons;
- In phase 2, we estimate illumination using the photon map.



*: Henrik Wann Jensen, The photon map in global illumination. Ph.D. dissertation, 1996.

Photons



Photon Mapping

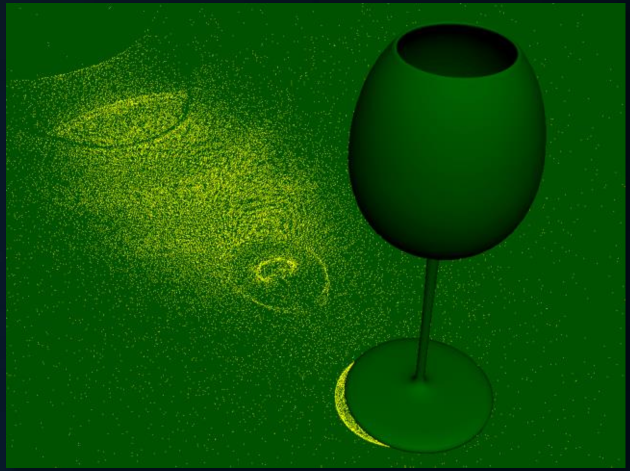
Phase 1: propagating flux.

Photon emission:

- Point light: emitted in uniformly distributed random directions from the point.
- Area light: emitted from random positions on the square, with directions limited to a hemisphere. The emission directions are chosen from a cosine distribution.

All photons have the same power: their density is the only way to express varying brightness.

Photons



Photon Mapping

Phase 1: propagating flux.

Surface interaction:

A photon that hits a surface may get absorbed or reflected.

Photons

Photon Mapping

Phase 1: propagating flux.

Photon storage:

At each non-specular path vertex we store the photon:

```
struct photon
{
    float3 position;     // world space position of the photon hit
    float3 power;        // current power level for the photon
    float3 L;            // incident direction
};
```

A photon may be stored multiple times along its path before it gets absorbed. Since the total set of photons represents the illumination, we divide photon power by the total number of stored photons.

Photons

Photon Mapping

Phase 2: radiance estimation.

In the second pass, we render the scene using rasterization or Whitted-style ray tracing to find the first diffuse surface; the photon map is then used to estimate illumination.

At each non-specular path vertex we estimate the reflected radiance:

$$L(x,\omega_o) = \int_{\Omega_x} f_r(x,\omega_i,\omega_o) \boxed{L_i(x,\omega_i)\cos\theta_i} \, d\omega_i$$

This requires information about the irradiance $L_i(x,\dots)\cos\theta_i$ arriving over the hemisphere Ω_x. We estimate this irradiance by looking at the photon density at x:

$$L(x,\omega_o) \approx \frac{1}{\pi r^2}\sum_{p=1}^{N} f_r(x,\omega_p,\omega_o)\Delta\Phi(x,\omega_p)$$

Photons

Photon Mapping

Phase 2: irradiance estimation*.

We estimate this irradiance by looking at the photon density at x:

$$L(x,\omega_o) \approx \frac{1}{\pi r^2}\sum_{p=1}^{N} f_r(x,\omega_p,\omega_o)\Delta\Phi(x,\omega_p)$$

Note:

- We gather photons on a disc of radius r.
- We assume that the gathered photons belong to the same surface.
- Each photon within radius r has the same influence on the estimate.

*: https://web.cs.wpi.edu/~emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html

Photons

Photon Mapping

Phase 2: irradiance estimation.

Instead of using the same weight for each photon we can use a filter:

$$w_{pg} = \alpha\left[1 - \frac{1 - e^{-\beta\frac{d_p^2}{2r^2}}}{1 - e^{-\beta}}\right] ,$$

where $\alpha = 0.918$, $\beta = 1.953$*. Value d_p^2 is the squared distance between photon p and x.
Now:

$$L(x, \omega_o) \approx \sum_{p=1}^{N} f_r(x, \omega_p, \omega_o)\Delta\Phi(x, \omega_p)\, w_{pg} \,.$$

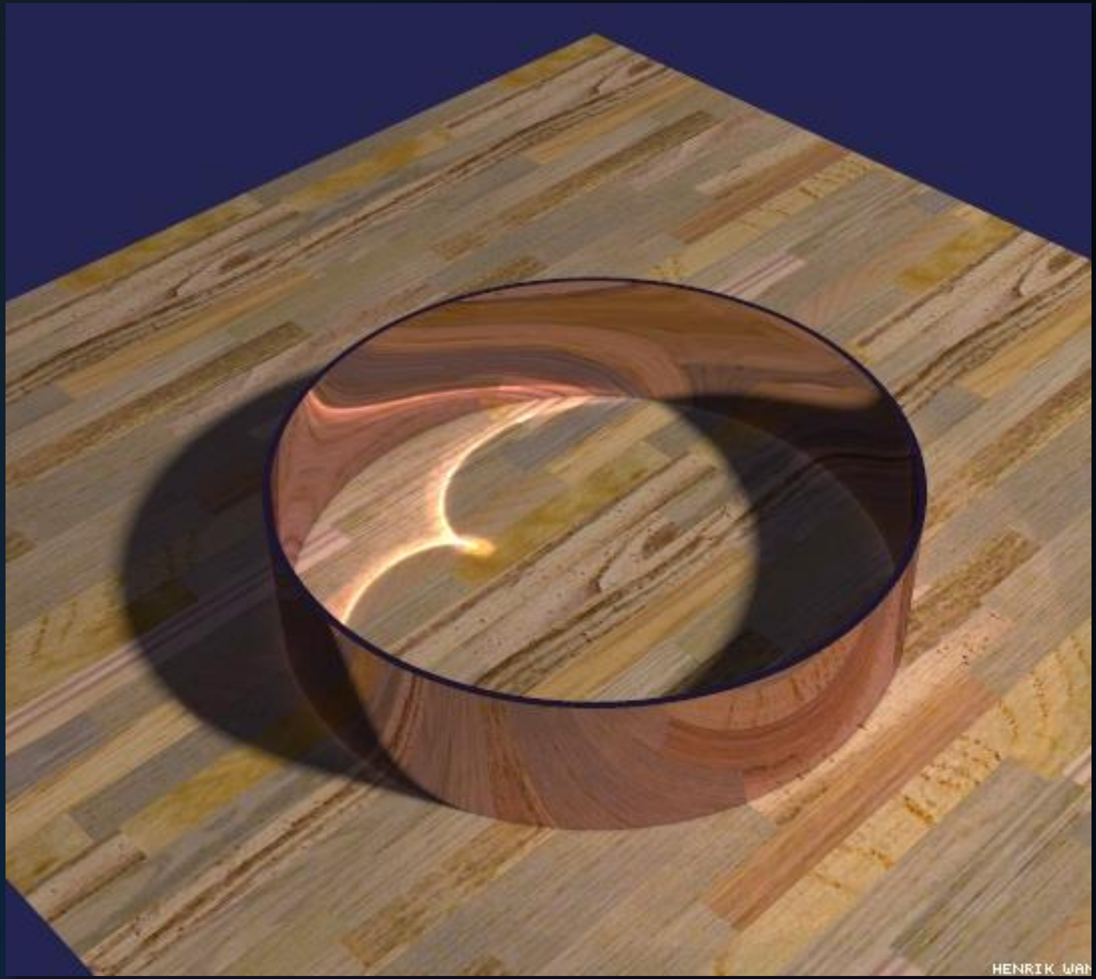
*: Mark J. Pavicic, Convenient Anti-Aliasing Filters that Minimize Bumpy Sampling. In Graphics Gems I.

Photons

Photon Mapping

Algorithm characteristics:

- Low-frequent noise
- Can be used in a rasterizer
- Can be used for direct + indirect
- Still a bidirectional technique.



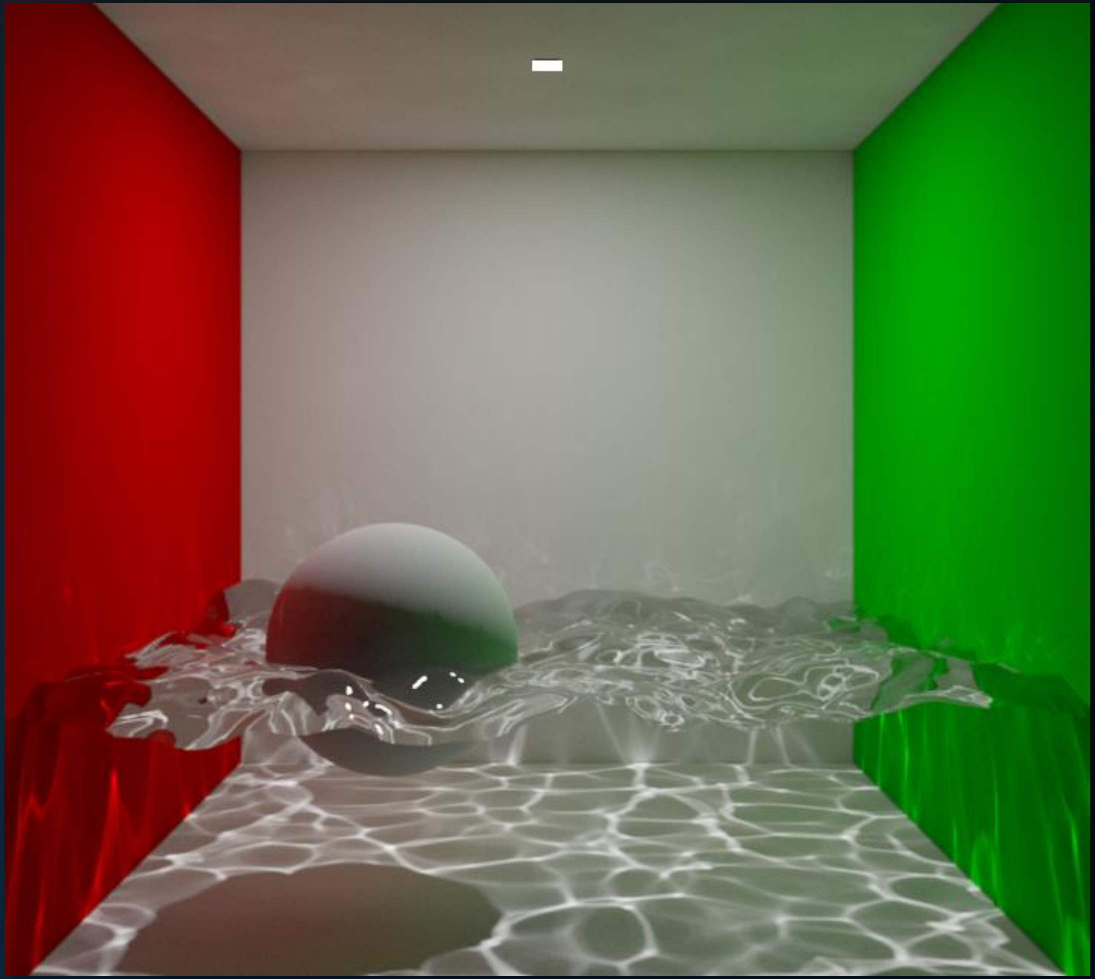
Photons

Photon Mapping

Algorithm characteristics:

- Low-frequent noise
- Can be used in a rasterizer
- Can be used for direct + indirect
- Still a bidirectional technique

Can be used to specifically replace paths a path tracer handles poorly, e.g. caustics.



Today’s Agenda:

- Recap: Forward Path Tracing
- Virtual Point Lights
- Photon Mapping
- Bidirectional Path Tracing
- More

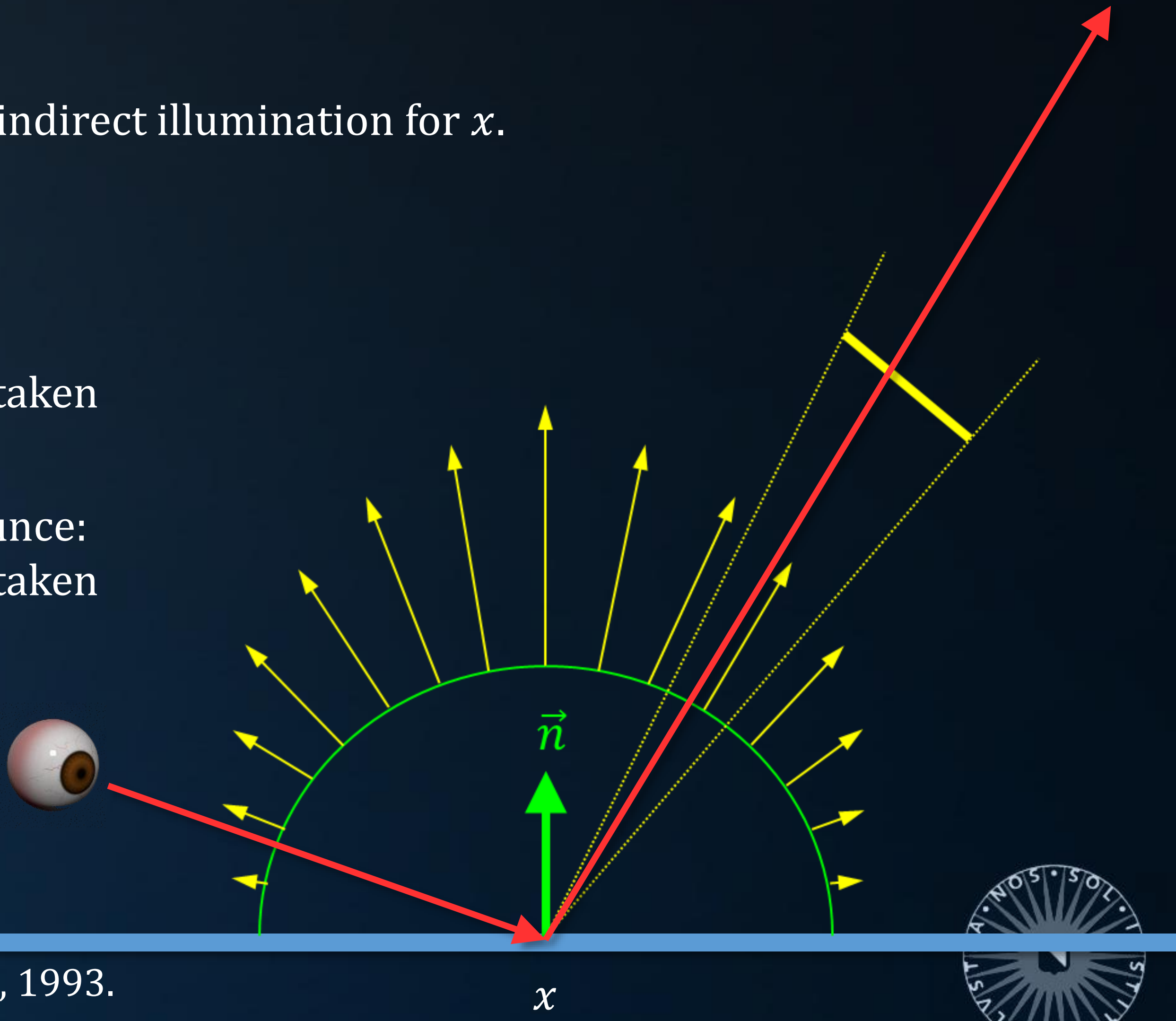
BDPT

Multiple Importance Sampling, Briefly

With NEE, we split the domain in direct and indirect illumination for x.
We use a different pdf for each subdomain.
With MIS, we combine the two pdfs:

1. When reaching a light with NEE:
   we consider the chance that would have taken
   us here via a random bounce.
2. When reaching a light with a random bounce:
   we consider the chance that would have taken
   us here via NEE.

Bidirectional Path Tracing*:
Taking this to extremes.



*: Bidirectional Path Tracing. Lafortune & Willems, 1993.
And: Veach, PhD thesis.

BDPT

Eye path:

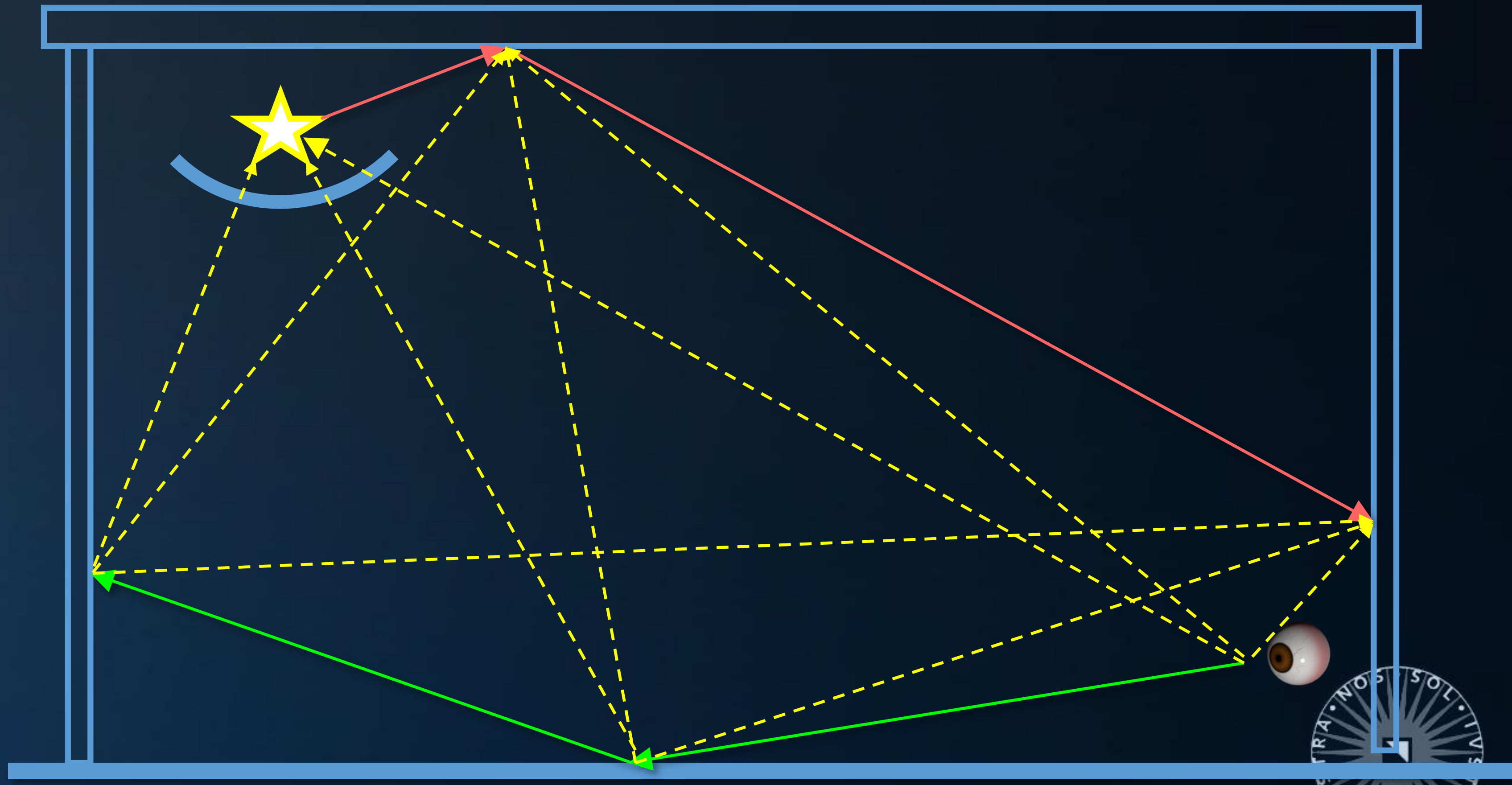
vertex t_0 (eye)
vertex t_1
vertex t_2

Light path:

vertex s_0 (light)
vertex s_1
vertex s_2

Connections:

$t_0..s_2..s_1..s_0$
$t_0..s_1..s_0$
$t_0..s_0$
$t_0..t_1..s_2..s_1..s_0$
$t_0..t_1..s_1..s_0$
$t_0..t_1..s_0$
$t_0..t_1..t_2..s_2..s_1..s_0$
$t_0..t_1..t_2..s_1..s_0$
$t_0..t_1..t_2..s_0$



BDPT

Eye path:

vertex t_0 (eye)
vertex t_1
vertex t_2

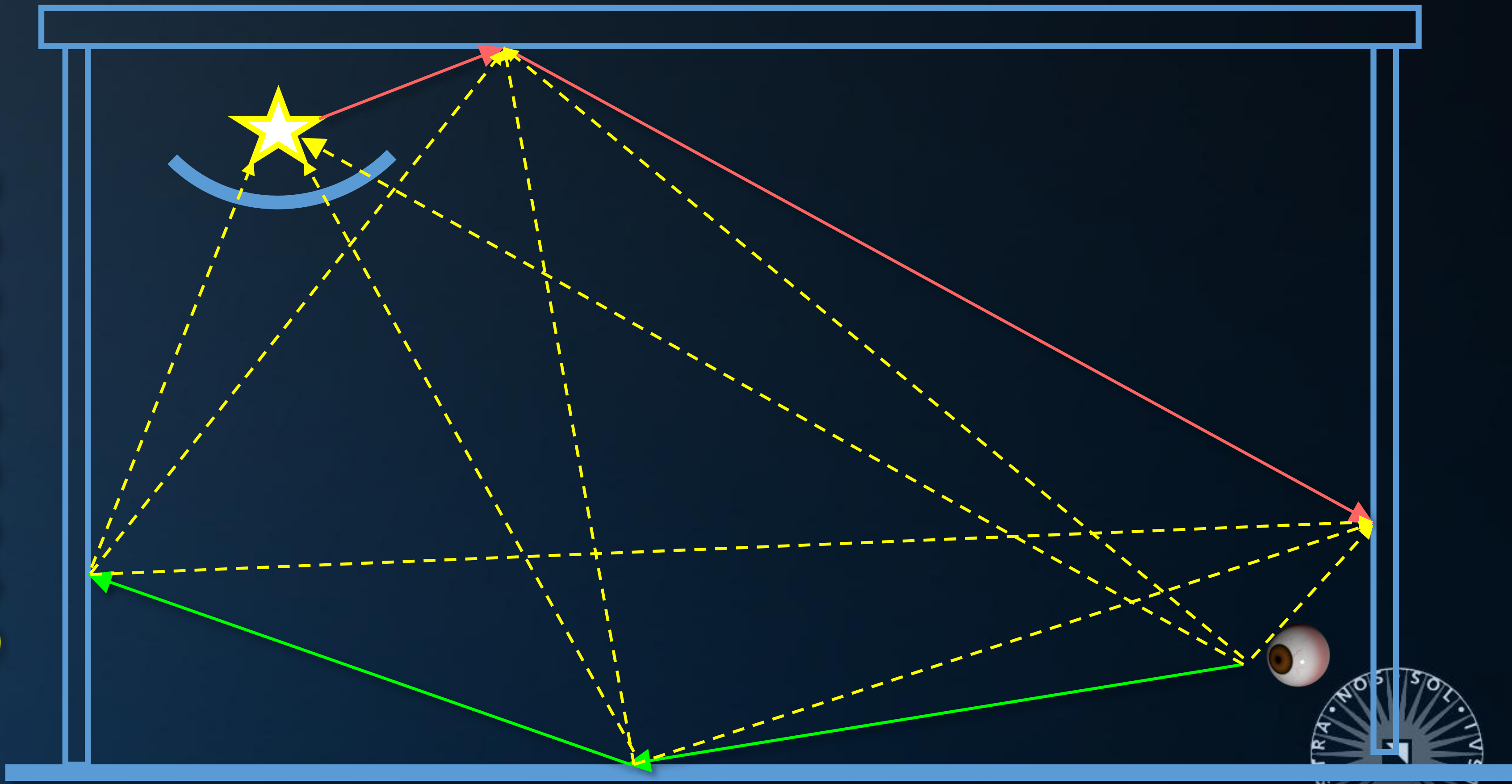
Light path:

vertex s_0 (light)
vertex s_1
vertex s_2

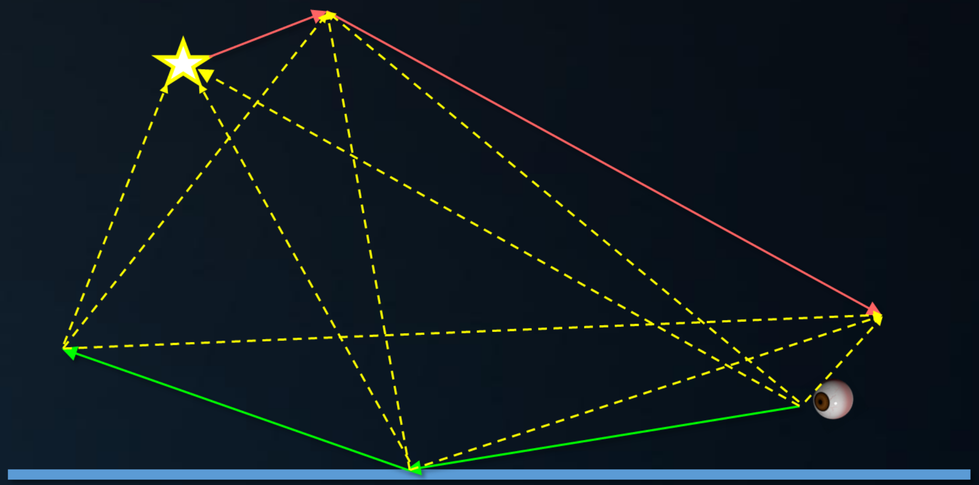
In BDPT, paths of the same length are equivalent techniques to connect the eye to the camera. We thus combine them using MIS.

Connections:

$t_0..s_0$	(2)
$t_0..s_1..s_0$	(3)
$t_0..t_1..s_0$	(3)
$t_0..s_2..s_1..s_0$	(4)
$t_0..t_1..s_1..s_0$	(4)
$t_0..t_1..t_2..s_0$	(4)
$t_0..t_1..s_2..s_1..s_0$	(5)
$t_0..t_1..t_2..s_1..s_0$	(5)
$t_0..t_1..t_2..s_2..s_1..s_0$	(6)



BDPT



Bidirectional Path Tracing*

The BDPT estimator takes the following form:

$$L_p = \sum_{s=0}^{N_L} \sum_{t=0}^{N_E} w_{s,t} C_{s,t}$$

Here, $C_{s,t}$ is the unweighted contribution of a path with s vertices on the light path and t vertices on the eye path.

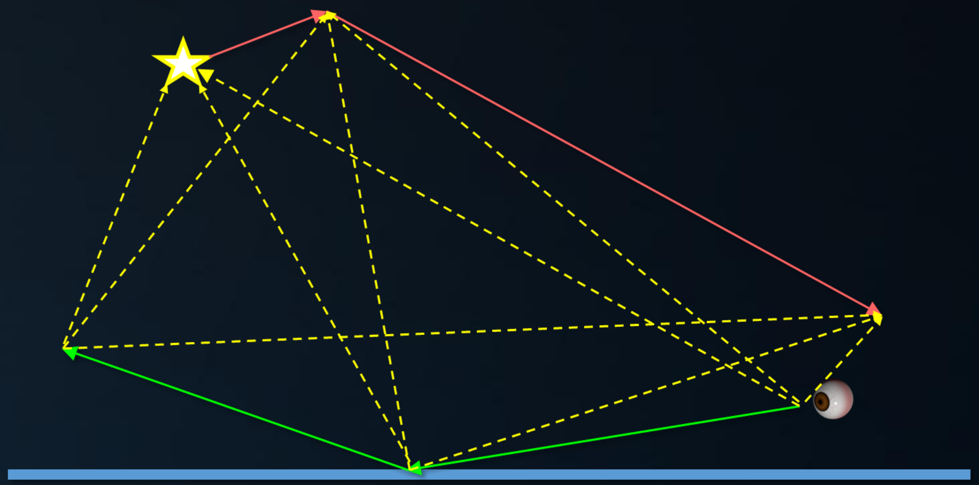
For the evaluation of $C_{s,t}$ we have several cases:

- $s > 0, t = 0$: light tracing (or forward path tracing);
- $s = 0, t > 0$: unidirectional path tracing;
- $s = 0, t = 0$: when the light is directly visible to the eye.
- And finally, all other paths: $s > 0, t > 0$.

*: Bidirectional Path Tracing. Lafortune & Willems, 1993. And: E. Veach, PhD thesis.

BDPT

Bidirectional Path Tracing*



When $s > 0, t > 0$:

$$C_{s,t} = \frac{L_e(y_0,\overrightarrow{y_0y_1})}{p(y_0,\overrightarrow{y_0y_1})} G(y_0 \leftrightarrow y_1) f_r(x_t,\overrightarrow{x_ty_s},\overrightarrow{x_tx_{t-1}}) f_r(y_s,\overrightarrow{y_sy_{s+1}},\overrightarrow{y_sx_t})$$

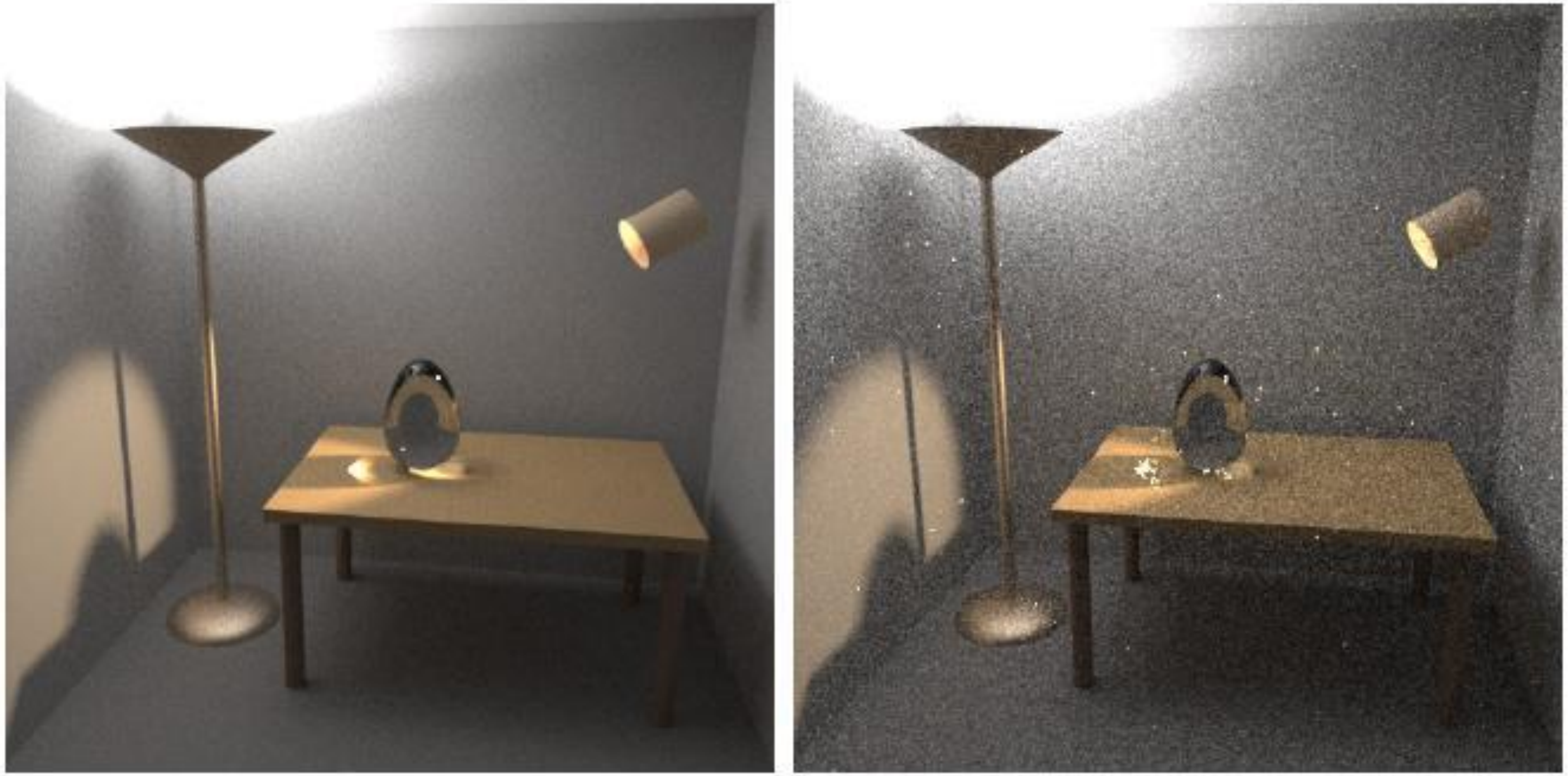
$$\left(\prod_{i=1}^{t-1} \frac{f_r(x_i,\overrightarrow{x_ix_{i+1}},\overrightarrow{x_ix_{i-1}})\left|N_{x_i}\cdot\overrightarrow{x_ix_{i+1}}\right|}{p(\overrightarrow{x_ix_{i+1}})}\right)\left(\prod_{i=1}^{s-1} \frac{f_r(y_i,\overrightarrow{y_iy_{i+1}},\overrightarrow{y_iy_{i-1}})\left|N_{y_i}\cdot\overrightarrow{y_iy_{i+1}}\right|}{p(\overrightarrow{y_iy_{i+1}})}\right)$$

Back to $L_p = \sum_{s=0}^{N_L}\sum_{t=0}^{N_E} w_{s,t}C_{s,t}$.

A few notes about w:

- Each path of $s+t$ vertices can be made in $s+t-1$ ways.
- The weights for a path length sum to 1.
- Weight calculation can be done in several ways:
- See Veach Chapter 10, PBRT book, other sources.

BDPT



(a) Bidirectional path tracing with 25 samples per pixel

(b) Standard path tracing with 56 samples per pixel (the same computation time as (a))

Today’s Agenda:

- Recap: Forward Path Tracing
- Virtual Point Lights
- Photon Mapping
- Bidirectional Path Tracing
- More

ALITA
BATTLE ANGEL

Path Guiding

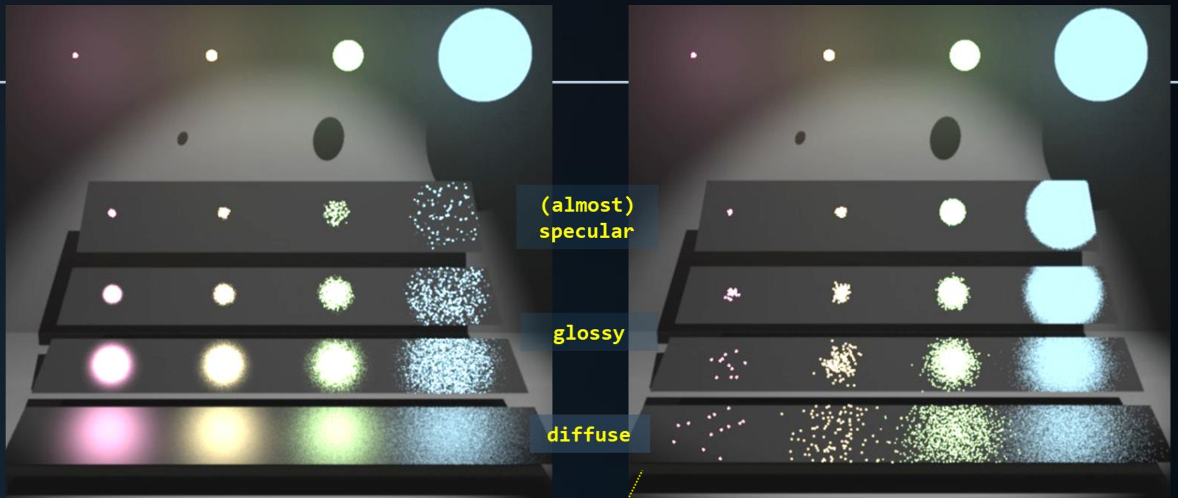
The Way of the Photon

Previously in ADVGR:

- We importance sampled
- Aiming for the important samples
- Blending strategies when needed
- Going bidirectional if all else fails.

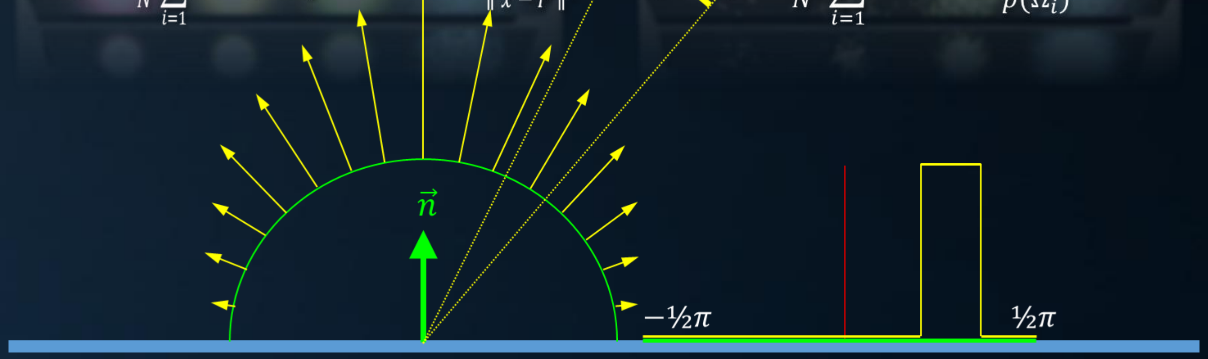
Now, what if I told you…

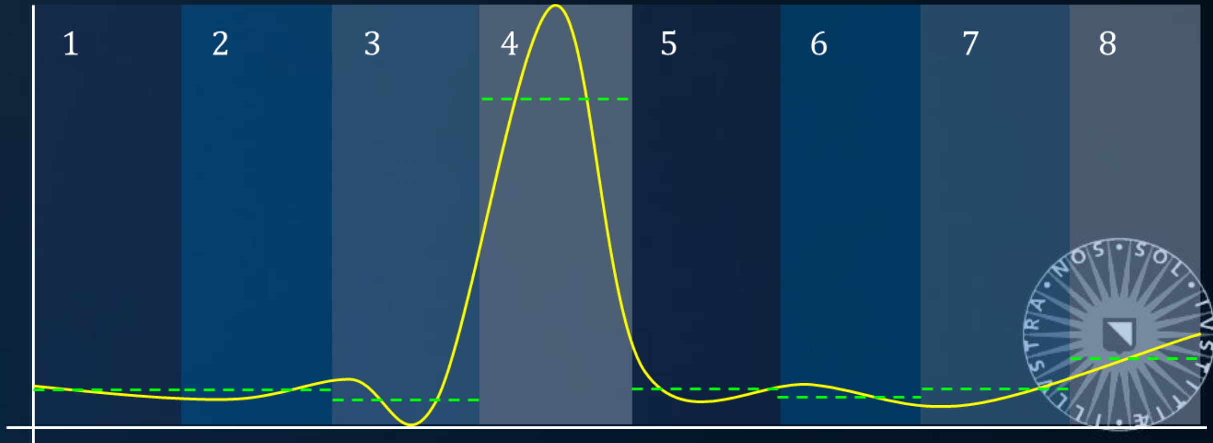
There’s a new way. ☺



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

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





```
rics
& (depth < MAXDEPTH)
c = inside ? 1 : 1.0f
nt = nt / nc, ddn
os2t = 1.0f - nnt * nnt
D, N );
0)
at a = nt - nc, b = nt + nc
at Tr = 1 - (R0 + (1 - R0)
Tr) R = (D * nnt - N * (ddn
E * diffuse;
= true;
efl + refr)) && (depth < MAXDEPTH)
D, N );
refl * E * diffuse;
= true;
MAXDEPTH)
survive = SurvivalProbability( diffuse
estimation - doing it properly, closely
df;
radiance = SampleLight( &rand, I, &L, &light
e.x + radiance.y + radiance.z) > 0) &&
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radiance
andom walk - done properly, closely following Small
vive)
;
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf );
urvive;
pdf;
n = E * brdf * (dot( N, R ) / pdf);
sion = true:
```

TO BE CONTINUED

Today’s Agenda:

- Recap: Forward Path Tracing
- Virtual Point Lights
- Photon Mapping
- Bidirectional Path Tracing
- More

INFOMAGR – Advanced Graphics

Jacco Bikker - November 2021 – February 2022

END of “Bidirectional”

next lecture: “TAA & ReSTIR”