hics & (depth < ≥0000

at a = nt - nc, b = nt - r at Tr = 1 - (R0 + (1 - R0 Fr) R = (D <sup>=</sup> nnt - N - (00)

= \* diffuse = true;

efl + refr)) && (depth < MAXDEPTH

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, closed If; radiance = SampleLight( &rand, I, \* + radiance v + radiance z)

v = true; at brdfPdf = EvaluateDiffuse( L, N ) Psurvis at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf) ();

andom walk - done properly, closely following See /ive)

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

## **INFOMAGR – Advanced Graphics**

Jacco Bikker - November 2021 - February 2022

# Lecture 13 - "Bidirectional"

 $\tilde{J}I(x', x'')dx'''$ 

Welcome!



tics & (depth < Modeo

z = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ∩ D, N ); ∂)

at a = nt - nc, b = nt + n at Tr = 1 - (R0 + (1 - R0 Tr) R = (D <sup>+</sup> nnt - N - (ddn

= \* diffuse = true;

-:fl + refr)) && (depth < MAXDERT

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &t, &i) a.x + radiance.y + radiance.z) > 0) &&

v = true; at brdfPdf = EvaluateDiffuse( L, N ) Psurviv at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* (((weight \* cosThetaOut) / directPdf) ();

andom walk - done properly, closely following Sour /ive)

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

### Today's Agenda:

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



### Forward

### **Backward and Forward Path Tracing**



efl \* E \* diffuse; = true;

#### AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly If; radiance = SampleLight( &rand, I, &L, \$11

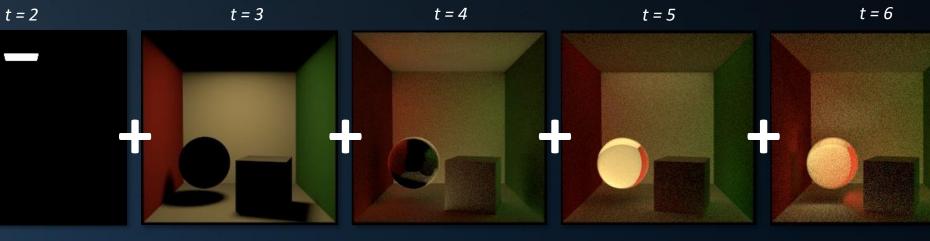
e.x + radiance.y + radiance.z) > 0) && (dot in w = true;

at brdfPdf = EvaluateDiffuse( L, N ) Psu at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L );

E \* ((weight \* cosThetaOut) / directPdf) \* (nad

andom walk - done properly, closely following Same /ive)

, t33 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, apdf urvive; .pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:







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

### Forward

tics ≹ (depth < Moder

: = inside ? 1 = 1 00 ht = nt / nc, ddn = 0 s2t = 1.0f - nnt = n 2, N ); ≫)

at a = nt - nc, b = nt + r at Tr = 1 - (R0 + (1 - R0 Fr) R = (D <sup>=</sup> nnt - N = (dd)

= \* diffuse = true;

efl + refr)) && (depth < MODEPTH

D, N ); refl \* E \* diffus = true;

AXDEPTH)

sion = true:

survive = SurvivalProbability( diffuse .estimation - doing it properly, if; radiance = SampleLight( &rand, I, &L, &llento 2.x + radiance.y + radiance.z) > 0) &&

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) (rad

andom walk - done properly, closely following Same /ive)

, t3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, apd urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf);

### Forward Path Tracing

A 'normal' path tracer works back to the lights (valid, <u>Helmholtz</u>).

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

24.20

### Forward

at a = nt

), N );

AXDEPTH)

survive = SurvivalProbability( lf: radiance = SampleLight( &rand, e.x + radiance.y + radiance.z)

v = true: at brdfPdf = EvaluateDiffuse( L, N at3 factor = diffuse \* INVPI at weight = Mis2( directPdf, brdfPdf at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf

andom walk - done properly, closely fol /ive)

at3 brdf = SampleDiffuse( diffuse, N, r1 urvive; pdf;

3.

4.

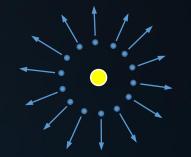
### Forward Path Tracing

A 'normal' path tracer works back to the lights (valid, <u>Helmholtz</u>).

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

Consequences / issues:

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



1 = E \* brdf \* (dot( N, R ) / pdf); sion = true:

24.20

### Forward

ics & (depth < MaxDonn

: = inside ? 1 1 1 1 ht = nt / nc, ddn bs2t = 1.0f - nnt " ∩ D, N ); ≥)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0) Fr) R = (D <sup>=</sup> nnt - N - (dd)

= \* diffuse; = true;

efl + refr)) && (depth < MOOD

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, did f; radiance = SampleLight( &rand, I, &L, & 2.x + radiance.y + radiance.z) > 0 500

v = true; at brdfPdf = EvaluateDiffuse( L

at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf at cosThetaOut = dot( N, L );

E \* ((weight \* cosThetaOut) / directPdf)

andom walk - done properly, closely following Sov /ive)

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

### Forward Path Tracing

Tracing paths from the light helps when:

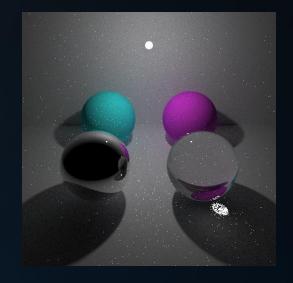
- the light is hard to reach
- the light cannot be *importance sampled* (using NEE).

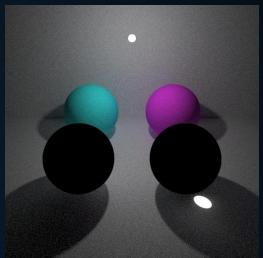
Tracing paths from the eye is better when:

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

ics & (depth < MoxDorn

: = inside ? 1 ht = nt / nc, ddn = 1 os2t = 1.0f - nnt " n 2, N ); ≥)

at a = nt - nc, b = nt + nc at Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N = (ddm)

= \* diffuse; = true;

efl + refr)) && (depth < MAXD

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly close if; radiance = SampleLight( &rand, I, &L, & e.x + radiance.y + radiance.z) > 0)

v = true; at brdfPdf = EvaluateDiffuse(

at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf at cosThetaOut = dot( N, L );

E \* ((weight \* cosThetaOut) / directPdf)

andom walk - done properly, closely following 300 /ive)

; at3 brdf = SampleDiffuse( diffuse, N, r1, r2, 8R, 8pdf urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

Forward Path Tracing

Tracing paths from the light helps whe

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

Tracing paths from the eye is better when:

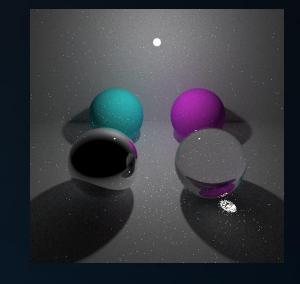
the camera is hard to reach.

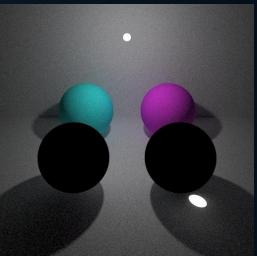
Many scenes would benefit from both approaches. Now what?

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

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?* 







### Forward

sics & (depth < Moonant

: = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ∩ n D, N ); ∂)

at a = nt - nc, b = nt + nc at Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N = (100)

= \* diffuse = true;

efl + refr)) && (depth < MAXDEPTH

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly closed ff; radiance = SampleLight( &rand, I, &L, 111 2.x + radiance.y + radiance.z) > 0) && (d)

v = true; at brdfPdf = EvaluateDiffuse( L, N ) \* Psi at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf)

andom walk - done properly, closely following /ive)

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

### Forward Path Tracing

The problem with this scene:

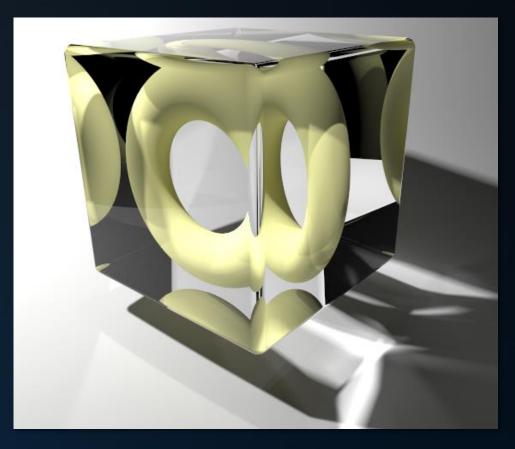
- When a path hits the torus, it can't use NEE
- This is true for light tracing and path tracing.

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



#### AXDEPTH)

survive = SurvivalProbability( diff lf; radiance = SampleLight( &rand, I, 8) .x + radiance.y + radiance.z) > 0)

#### v = true;

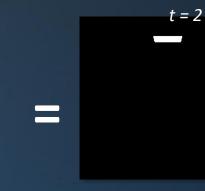
at brdfPdf = EvaluateDiffuse( L, N ) at3 factor = diffuse \* INVPI at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L );

E \* ((weight \* cosThetaOut) / directPdf)

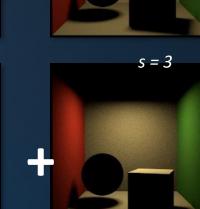
/ive)

at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, A urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf);

sion = true:







*t* = 3







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

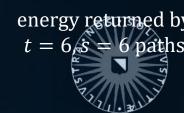
energy returned by t = 3, s = 3 paths EDL - LDE

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

t = 4

energy returned by t = 5, s = 5 paths

t = 5



t = 6

### Forward

sics & (depth < Monos⊺

: = inside ? 1 ht = nt / nc, ddn os2t = 1.0f - nmt - on 0, N ); 3)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0 Fr) R = (D - nnt - N - (00

= \* diffuse; = true;

• efl + refr)) && (depth < MOXDEPT

), N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, closed Hf; radiance = SampleLight( &rand, I, &L, &light e.x + radiance.y + radiance.z) > 0) && closed

w = true; at brdfPdf = EvaluateDiffuse( L, N ) Psurv at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf)

andom walk - done properly, closely following Soci /ive)

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

### 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

sics & (depth < MaxCor

: = inside ? 1 ( ) ) ht = nt / nc, ddn () ss2t = 1.0f - nnt " ∩ 2, N ); ð)

at a = nt - nc, b = nt = rcat Tr = 1 - (R0 + (1 - R0) fr) R = (D = nnt - N (ddn)

= \* diffuse; = true;

. efl + refr)) && (depth < MAXDEPIN

), N ); refl \* E \* diffu = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, df; radiance = SampleLight( &rand, I, &L, &ilent 2.x + radiance.y + radiance.z) > 0) &&

w = true; at brdfPdf = EvaluateDiffuse( L, N ) '

at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf) (cost

andom walk - done properly, closely following Sec. /ive)

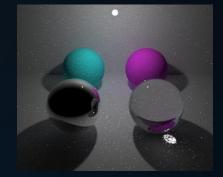
; at3 brdf = SampleDiffuse( diffuse, N, r1, r2, 8R, 8pdf urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### 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.









tics & (depth < Modeo

z = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ∩ D, N ); ∂)

at a = nt - nc, b = nt + n at Tr = 1 - (R0 + (1 - R0 Tr) R = (D <sup>+</sup> nnt - N - (ddn

= \* diffuse = true;

-:fl + refr)) && (depth < MAXDERT

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &t, &i) a.x + radiance.y + radiance.z) > 0) &&

v = true; at brdfPdf = EvaluateDiffuse( L, N ) Psurviv at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* (((weight \* cosThetaOut) / directPdf) ();

andom walk - done properly, closely following Sour /ive)

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

### Today's Agenda:

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



## VPLs

sics & (depth < ⊁0000

: = inside 7 1 ... ht = nt / nc, ddn ... bs2t = 1.0f - n⊓t ? D, N ); ∂)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0 Ir) R = (D = nnt - N

= \* diffuse; = true;

efl + refr)) && (depth < MAXDEP

), N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, df; radiance = SampleLight( &rand, I, &L, &L, e.x + radiance.y + radiance.z) > 0) && ()

w = true; at brdfPdf = EvaluateDiffuse( L, N ) Psurvis at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf)

andom walk - done properly, closely following 300 /ive)

; at3 brdf = SampleDiffuse( diffuse, N, r1, r2\*: A. Keller, Instant Radiosity. SIGGRAPH '97. pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### Instant Radiosity\*

Idea:

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

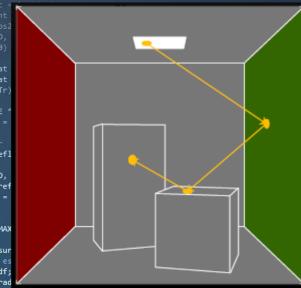
Now, render the scene with rasterization or Whitted-style ray tracing. At the first diffuse surface, use the VPLs to estimate indirect light, and the lights themselves for direct illumination.

(did we account for all light transport?)



### VPLs

### Instant Radiosity



e.x + radiance.y + radiance.z) > 0) 88 (

/ = true;

at brdfPdf = EvaluateDiffuse( L, N )

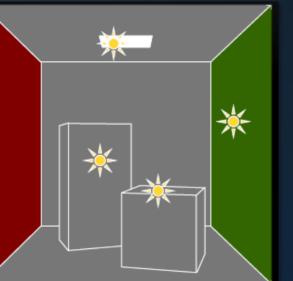
at weight = Mis2( directPdf, brdfPdf );

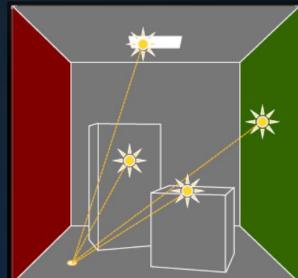
at cosThetaOut = dot( N, L );

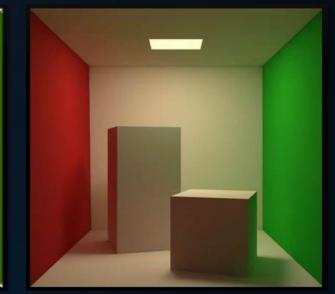
E \* ((weight \* cosThetaOut) / directPdf) = (no

ndom walk - done properly, closely following Sec. ive)

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







Images: M. Hasan, SIGGRAPH Asia '09.



## VPLs

nics & (depth < NACCO

at a = nt - nc, b = nt = n at Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N = (ddn )

= \* diffuse; = true;

efl + refr)) && (depth < □0

D, N ); refl \* E \* diffus = true;

#### AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &L, e.x + radiance.y + radiance.z) > 0) &&

v = true; at brdfPdf = EvaluateDiffuse( L, N ) \* Psurs at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf)

andom walk - done properly, closely following Sec /ive)

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

### 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</p>
- 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.





tics & (depth < Modeo

z = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ∩ D, N ); ∂)

at a = nt - nc, b = nt + n at Tr = 1 - (R0 + (1 - R0 Tr) R = (D <sup>+</sup> nnt - N - (ddn

= \* diffuse = true;

-:fl + refr)) && (depth < MAXDERT

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &t, &i) a.x + radiance.y + radiance.z) > 0) &&

v = true; at brdfPdf = EvaluateDiffuse( L, N ) Psurviv at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* (((weight \* cosThetaOut) / directPdf) ();

andom walk - done properly, closely following Sour /ive)

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

### Today's Agenda:

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



## Photons

at a = nt

efl + refr)) && (dept)

), N ); refl \* E \* diffuse;

AXDEPTH)

survive = SurvivalProbability lf: radiance = SampleLight( &rand e.x + radiance.y + radiance.z

v = true; at brdfPdf = EvaluateDiffuse at3 factor = diffuse \* INVPI at weight = Mis2( directPdf, brdf at cosThetaOut = dot( N, L ) E \* ((weight \* cosThetaOut) /

andom walk - done prop /ive)

urvive; pdf; 1 = E \* brdf \* (dot( N, R ) / pdf); sion = true:

Photon Mapping\*

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

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





at3 brdf = SampleDiffuse( diffuse, N, r1, r2\*: Henrik Wann Jensen, The photon map in global illumination. Ph.D. dissertation, 1996.



### 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.

andom walk - done properly, closely following Sout vive)

; t3 brdf = SampleDiffuse( diffuse, N, r1, r2, RR, Spdf) urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:



## Photons

nics & (depth < NACCO

t = inside 7 1 1 1 0 ht = nt / nc, ddn 0 0 ps2t = 1.0f - nπt 0 0 p, N ); ≫)

at a = nt - nc, b = nt - r at Tr = 1 - (R0 + (1 - R0 Fr) R = (D \* nnt - N \* (dd

= \* diffuse; = true;

efl + refr)) && (depth < MAXDEPTH

D, N ); refl \* E \* diffus = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, closed if; radiance = SampleLight( &rand, I, &L, &light) 2.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) (rad

andom walk - done properly, closely following Sec. /ive)

; at3 brdf = SampleDiffuse( diffuse, N, r1, r2, 8R, 8pdf urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### Photon Mapping

Phase 1: propagating flux.

Surface interaction:

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







sics & (depth < ⊅00000

: = inside ? 1 1 1 1 ht = nt / nc, ddn 1 ps2t = 1.0f - nmt 7 7 2, N ); 2)

at a = nt - nc, b = nt + nc at Tr = 1 - (R0 + (1 - R0 Ir) R = (D = nnt - N - (dd)

= \* diffuse; = true;

. :fl + refr)) && (depth < MAXDEPTH

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &L, &L) e.x + radiance.y + radiance.z) > 0) && ()

v = true; + bodf0df

at brdfPdf = EvaluateDiffuse( L, N )
at3 factor = diffuse \* INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E \* ((weight \* cosThetaOut) / directPd

andom walk - done properly, closely following /ive)

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

### Photon Mapping

Phase 1: propagating flux.

Photon storage:

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

#### struct photon

};

float3 position;
float3 power;
float3 L;

// world space position of the photon hit
// current power level for the photon
// 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.



nics & (depth < NACCO

: = inside ? 1 ( ) . ht = nt / nc, ddn bs2t = 1.0f - n⊓t 2, N ); 2)

at a = nt - nc, b = nt + nc, at Tr = 1 - (R0 + (1 - R0  $\Gamma$ r) R = (D = nnt - N - (dd)

= \* diffuse; = true;

-:fl + refr)) && (depth < MAXDEPT

D, N ); refl \* E \* diff = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &L e.x + radiance.y + radiance.z) > 0) & &

w = true; at brdfPdf = EvaluateDiffuse( L, N ) \* P at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf

andom walk - done properly, closely following /ive)

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

### 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:

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

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

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



ics & (depth < Moder

z = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ∩ D, N ); ∂)

at a = nt - nc, b = nt - n at Tr = 1 - (R0 + (1 - R0 Tr) R = (D <sup>=</sup> nnt - N

= \* diffuse = true;

. efl + refr)) && (depth < NACESS

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, close if; radiance = SampleLight( &rand, I, &L e.x + radiance.y + radiance.z) > 0) \$23

v = true; at brdfPdf = EvaluateDiffuse( L, N ) P at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf

andom walk - done properly, closely following Se /ive)

, t3 brdf = SampleDiffuse( diffuse, N, r1 urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### 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.



22

\*: https://web.cs.wpi.edu/~emmanuel/courses/cs563/write\_ups/zackw/photon\_mapping/PhotonMapping.html

nics & (depth < Modes

: = inside ? 1 1 1 nt = nt / nc, ddm os2t = 1.0f - nmt ~ 0, N ); 0)

at a = nt - nc, b = nt .... at Tr = 1 - (R0 + (1 - R0 Ir) R = (D = nnt - N

= \* diffuse = true;

. efl + refr)) && (depth < MAXDEP

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse
estimation - doing it properly, closed
if;
radiance = SampleLight( %rand, I, %1, %1)
e.x + radiance.y + radiance.z) > 0) &&

w = true; at brdfPdf = EvaluateDiffuse( L, N ) \* Psur at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf)

andom walk - done properly, closely following 30 /ive)

; at3 brdf = SampleDiffuse( diffuse, N, r1,

urvive; pdf; \_ = 5 \* bodf \* (

n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### Photon Mapping

Phase 2: irradiance estimation.

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

$$w_{pg} = lpha \left[ 1 - rac{1 - e^{-eta rac{d_p^2}{2r^2}}}{1 - e^{-eta}} 
ight]$$
 ,

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

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

The state of the s

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

## Photons

tics & (depth < NACCO

at a = nt - nc, b = nt rat Tr = 1 - (R0 + (1 R0 Tr) R = (D = nnt - N (1 R0

= \* diffuse; = true;

. efl + refr)) && (depth < MAXDEP

D, N ); refl \* E \* diffus = true;

#### AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &L, &lienti e.x + radiance.y + radiance.z) > 0) && (dot)

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) = (Pad

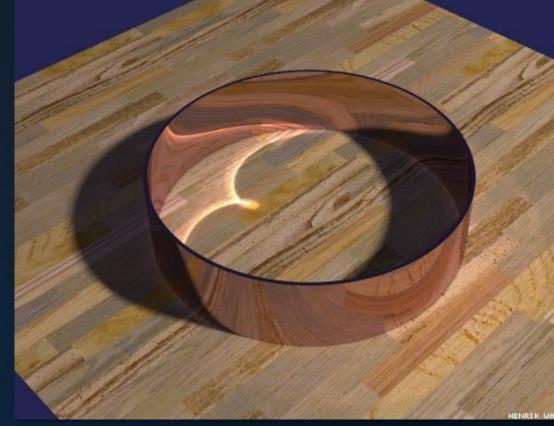
andom walk - done properly, closely following Sec. /ive)

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

### Photon Mapping

Algorithm characteristics:

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





## Photons

ics & (depth < NACCO

: = inside ? 1 ht = nt / nc, ddm - os2t = 1.0f - nnt - nn 0, N ); 3)

at a = nt - nc, b = nt - nc, b = nt - (R0 + (1 - R0))at Tr = 1 - (R0 + (1 - R0))Tr) R =  $(D^{-1} nnt - N^{-1})$ 

= \* diffuse; = true;

. efl + refr)) && (depth < MAXDEPT)

D, N ); refl \* E \* diffu = true;

AXDEPTH)

survive = SurvivalProbability( diffuse
estimation - doing it properly, closed
if;
radiance = SampleLight( &rand, I, &L, &light
e.x + radiance.y + radiance.z) > 0) && dotted

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) (psurvive);

andom walk - done properly, closely following Sec. /ive)

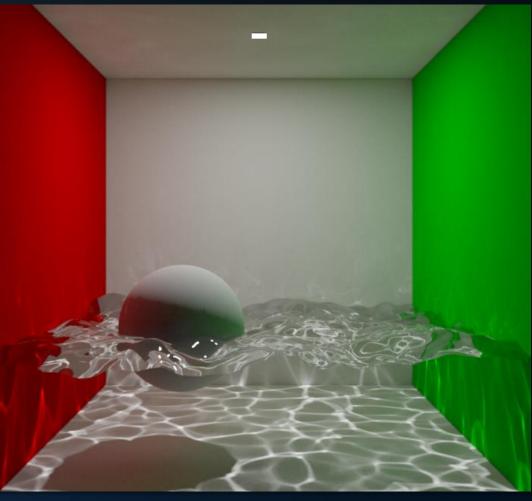
; at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### 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.





tics & (depth < Modeo

z = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ∩ D, N ); ∂)

at a = nt - nc, b = nt + n at Tr = 1 - (R0 + (1 - R0 Tr) R = (D <sup>+</sup> nnt - N - (ddn

= \* diffuse = true;

-:fl + refr)) && (depth < MAXDERT

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &t, &i) a.x + radiance.y + radiance.z) > 0) &&

v = true; at brdfPdf = EvaluateDiffuse( L, N ) Psurviv at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* (((weight \* cosThetaOut) / directPdf) ();

andom walk - done properly, closely following Sour /ive)

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

### Today's Agenda:

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



## BDPT

tics & (depth < ⊅000000

: = inside ? | ht = nt / nc, ddn os2t = 1.0f - n⊓t 2, N ); 2)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0 Fr) R = (D <sup>=</sup> nnt - N

= \* diffuse; = true;

efl + refr)) && (depth < MADER

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse
estimation - doing it properly, close
if;
radiance = SampleLight( &rand, I, &L, &
e.x + radiance.y + radiance.z) > 0) &

v = true; at brdfPdf = EvaluateDiffuse( L, N ) \* at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPd

andom walk - done properly, closely following 34 /ive)

; at3 brdf = SampleDiffuse( diffuse, N, r3 urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### 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:

- 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.



 $\boldsymbol{\chi}$ 



Eye path:

vertex  $t_1$ 

vertex  $t_2$ 

vertex  $t_0$  (eye)

**Connections:** 

*t*<sub>0</sub>..*t*<sub>1</sub>..*s*<sub>2</sub>..*s*<sub>1</sub>..*s*<sub>0</sub>

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

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

 $t_0...t_1...t_2...s_0$ 

 $t_0...t_1...s_1...s_0$ 

 $t_0...t_1...s_0$ 

 $t_0 ... s_2 ... s_1 ... s_0$ 

*t*<sub>0</sub>..*s*<sub>1</sub>..*s*<sub>0</sub>

 $t_0...s_0$ 

## BDPT

ics & (depth < Modest

: = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ° ∩ D, N ); ≥)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N = (000

= \* diffuse; = true;

. efl + refr)) && (depth < MAXD

), N ); refl \* E \* diffus = true;

#### AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, closed if; radiance = SampleLight( &rand, I, &L, &II e.x + radiance.y + radiance.z) > 0) &&

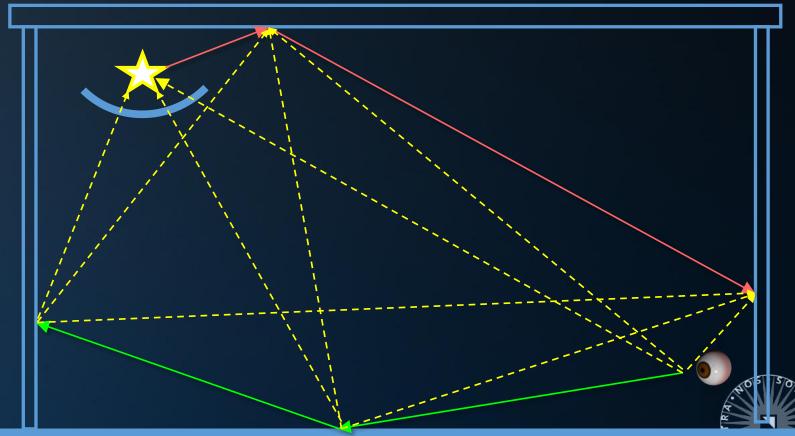
w = true; at brdfPdf = EvaluateDiffuse( L, N ) at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / direct)

andom walk - done properly, closely following Sao /ive)

t3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf )
urvive;
pdf;
n = E \* brdf \* (dot( N, R ) / pdf);
sion = true:

### Light path:

vertex  $s_0$  (light) vertex  $s_1$ vertex  $s_2$ 





## BDPT

ics & (depth < MAXDERT)

t = inside ? 1 ht = nt / nc, ddn = 0 os2t = 1.0f - nnt " n 2, N ); ≷)

at a = nt - nc, b = nt at Tr = 1 - (R0 + (1 - R0 Ir) R = (D = nnt - N

= \* diffuse; = true;

-=**fl + refr**)) && (depth

), N ); refl \* E \* diffus = true;

#### AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &L 2.x + radiance.y + radiance.z) > 0) &&

w = true; at brdfPdf = EvaluateDiffuse( L, N ) at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / direct

andom walk - done properly, closely following See /ive)

. t3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, apd ) prvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

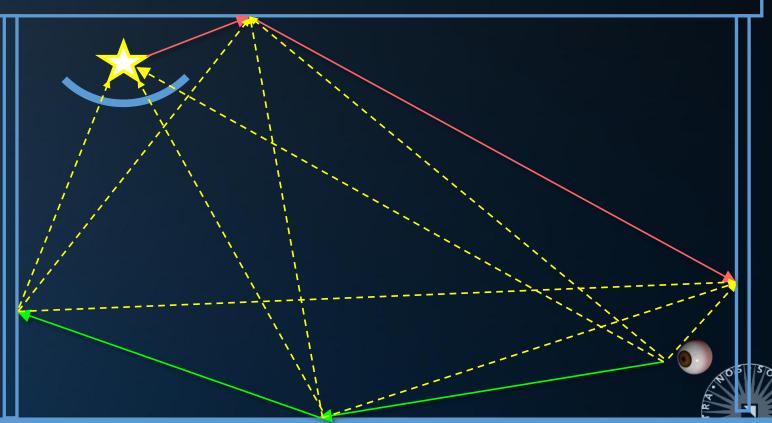
### Eye path:

vertex  $t_0$  (eye) vertex  $t_1$ vertex  $t_2$ 

### **Connections:**

(2) $t_0 ... s_0$ (3) $t_0...s_1...s_0$ (3)*t*<sub>0</sub>..*t*<sub>1</sub>..*s*<sub>0</sub> (4) $t_0 ... s_2 ... s_1 ... s_0$ (4) $t_0..t_1..s_1..s_0$ (4) $t_0...t_1...t_2...s_0$ (5) $t_0 ... t_1 ... s_2 ... s_1 ... s_0$ (5) $t_0...t_1...t_2...s_1...s_0$  $t_0 ... t_1 ... t_2 ... s_2 ... s_1 ... s_0$  (6)

Light path: vertex s<sub>0</sub> (light) vertex s<sub>1</sub> vertex s<sub>2</sub> In BDPT, paths of the same length are equivalent techniques to connect the eye to the camera. We thus combine them using MIS.





## BDPT

= true:

efl + refr)) && (de

), N );

#### AXDEPTH)

survive = SurvivalProbabilit radiance = SampleLight( &rand x + radiance.v + radiance

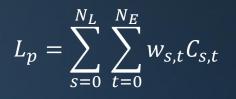
- v = true:
- at brdfPdf = EvaluateDiffuse( at3 factor = diffuse \* INVPI
- at weight = Mis2( directPdf, brdfPdf at cosThetaOut = dot( N, L );
- E \* ((weight \* cosThetaOut) / directPd

andom walk - done properly, closel /ive)

at3 brdf = SampleDiffuse( diffuse, N, r1, urvive; pdf; 1 = E \* brdf \* (dot( N, R ) / pdf);

### **Bidirectional Path Tracing\***

The BDPT estimator takes the following form:





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.

30

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

## BDPT

Bidirectional Path Tracing\*

 $N_L N_E$ 

s=0 t=0

 $w_{s,t}C_{s,t}$ .

#### When s > 0, t > 0:

Back to  $L_p =$ 

nt = nt / nc, dd os2t = 1.0f - nn 0, N ); 3)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0  $\Gamma$ r) R = (D = nnt - N = (100)

= \* diffuse = true;

. efl + refr)) && (depth < MAXDEDIII

D, N ); refl \* E \* diffuse; = true;

#### AXDEPTH)

survive = SurvivalProbability( diffuse
estimation - doing it properly, close
if;
radiance = SampleLight( &rand, I, &L, &l
e.x + radiance.y + radiance.z) > 0) &&

v = true; at brdfPdf = EvaluateDiffuse( L, N ) \* P at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L );

E \* ((weight \* cosThetaOut) / directPdf) (radi

andom walk - done properly, closely following Sec. /ive)

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

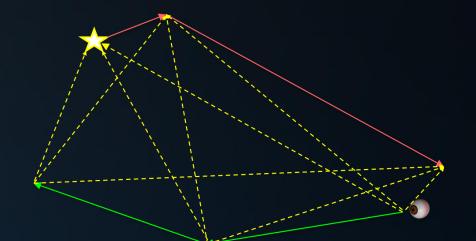
$$C_{s,t} = \frac{L_e(y_0, \overline{y_0 y_1})}{p(y_0, \overline{y_0 y_1})} G(y_0 \leftrightarrow y_1) f_r(x_t, \overline{x_t y_s}, \overline{x_t x_{t-1}}) f_r(y_s, \overline{y_s y_{s+1}}, \overline{y_s x_t})$$

$$= \frac{f_r(x_i, \overline{x_i x_{i+1}}, \overline{x_i x_{i-1}}) |N_{x_i} \cdot \overline{x_i x_{i+1}}|}{p(\overline{x_i x_{i+1}})} \left( \prod_{i=1}^{s-1} \frac{f_r(y_i, \overline{y_i y_{i+1}}, \overline{y_i y_{i-1}}) |N_{y_i} \cdot \overline{y_i y_{i+1}}|}{p(\overline{y_i y_{i+1}})} \right)$$

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

tics & (depth < Mooss

t = inside ? 1 | | | ht = nt / nc, ddn bs2t = 1.0f - nnt ? r 2, N ); ≱)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0 Fr) R = (D <sup>=</sup> nnt - N <sup>-</sup> (dd)

= \* diffuse = true;

efl + refr)) && (depth < MODEPTI

D, N ); refl \* E \* diffu = true;

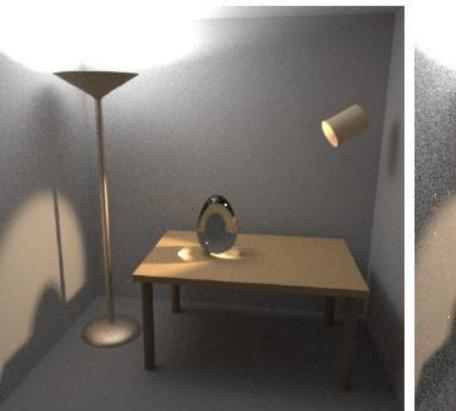
#### AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, if; radiance = SampleLight( &rand, I, &L e.x + radiance.y + radiance.z) > 0) &&

v = true; at brdfPdf = EvaluateDiffuse( L, N ) \* Ps at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf)

andom walk - done properly, closely following /ive)

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



(a) Bidirectional path tracing with 25 samples per pixel



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



tics & (depth < Modeo

z = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ∩ D, N ); ∂)

at a = nt - nc, b = nt + n at Tr = 1 - (R0 + (1 - R0 Tr) R = (D <sup>+</sup> nnt - N - (ddn

= \* diffuse = true;

-:fl + refr)) && (depth < MAXDERT

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &t, &i) a.x + radiance.y + radiance.z) > 0) &&

v = true; at brdfPdf = EvaluateDiffuse( L, N ) Psurviv at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* (((weight \* cosThetaOut) / directPdf) ();

andom walk - done properly, closely following Sour /ive)

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

### Today's Agenda:

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







## 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...

#### survive = SurvivalProbability( diff lf; radiance = SampleLight( &rand, I e.x + radiance.y + radiance.z)

at a = ni

), N );

AXDEPTH)

refl \* E \* diffuse;

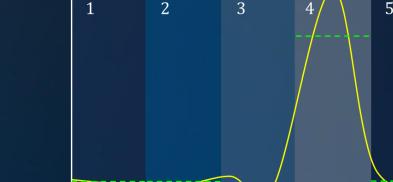
v = true: at brdfPdf = EvaluateDiffuse( at3 factor = diffuse \* INVPI at weight = Mis2( directPdf, brdfPdf at cosThetaOut = dot( N, L );

E \* ((weight \* cosThetaOut) / directPdf)

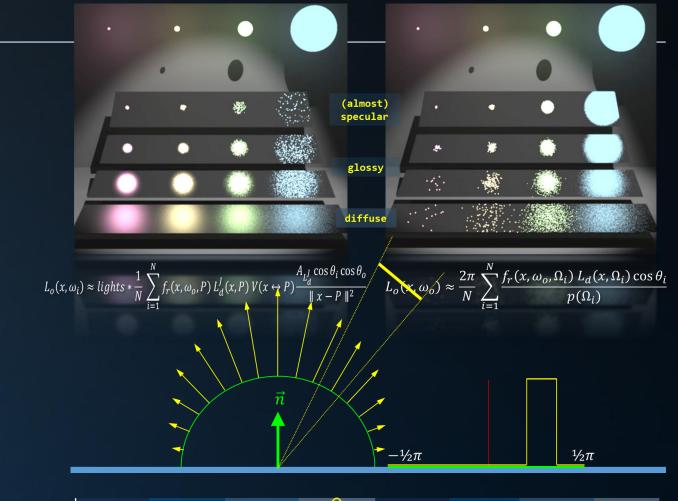
andom walk - done properly, closely follo /ive)

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

### 







### There's a new way. 😊

tics & (depth < NACC

t = inside ? 1 ht = nt / nc, ddn = 1 os2t = 1.0f - nnt " ∩ 2, N ); ≥)

at a = nt - nc, b = nt = nat Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N - (dom

= \* diffuse; = true;

. :fl + refr)) && (depth < MAXDEPTH

D, N ); refl \* E \* diffusa = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, closed if; radiance = SampleLight( &rand, I, &L, &light) 2.x + radiance.y + radiance.z) > 0) && (000)

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) \* (radJ

andom walk - done properly, closely following Sec. /ive)

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

#### TO BE CONTINUED



tics & (depth < Modeo

z = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ∩ D, N ); ∂)

at a = nt - nc, b = nt + n at Tr = 1 - (R0 + (1 - R0 Tr) R = (D <sup>+</sup> nnt - N - (ddn

= \* diffuse = true;

-:fl + refr)) && (depth < MAXDERT

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &t, &i) a.x + radiance.y + radiance.z) > 0) &&

v = true; at brdfPdf = EvaluateDiffuse( L, N ) Psurviv at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* (((weight \* cosThetaOut) / directPdf) ();

andom walk - done properly, closely following Sour /ive)

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

### Today's Agenda:

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



hics & (depth < NoCOS

: = inside ? 1 ht = nt / nc, ddn = u os2t = 1.0f - nmt = on D, N ); B)

at a = nt - nc, b = nt at Tr = 1 - (R0 + (1 - R0) Fr) R = (D <sup>+</sup> nnt - N - (dd)

= \* diffuse = true;

. efl + refr)) && (depth < MAXDEPT

D, N ); refl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( diffuse .estimation - doing it properly, const if; radiance = SampleLight( &rand, I, &L, &ilento 2.x + radiance.y + radiance.z) > 0) & \_\_\_\_\_

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)

andom walk - done properly, closely following Sec. /ive)

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

# **INFOMAGR – Advanced Graphics**

Jacco Bikker - November 2021 – February 2022

# END of "Bidirectional"

next lecture: "TAA & ReSTIR"

