hics & (depth < ≥0000

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

= * diffuse = true;

efl + refr)) && (depth < MODEPTI

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

AXDEPTH)

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

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 Sec /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 4 - "Light Transport"

Welcome!



ics & (depth < Mox000

= inside ? 1 1 1 0 ht = nt / nc, ddn bs2t = 1.0f - nnt = n D, N); B)

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

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:

Today's Agenda:

- Introduction
- The Rendering Equation
- Light Transport



Whitted



e.x + radiance.y + radiance.z

v = true;

at brdfPdf = EvaluateDiffuse(L, N) Paulat 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)

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





Whitted



radiance = SampleLight(&rand, I, &L,) 2.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) * (ra

andom walk - done properly, closely following 3001 /ive)

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



Advanced Graphics – Light Transport

Introduction

nics & (depth < ⊅00000

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

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

= * diffuse; = true;

-•fl + refr)) && (depth < MAX

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

AXDEPTH)

survive = SurvivalProbability(diff estimation - doing it properly, cl if;

radiance = SampleLight(&rand, I, &L, &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) / 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:

Whitted

Missing:

- Area lights
- Glossy reflections
- Caustics
- Diffuse interreflections
- Diffraction

- Polarization
- Phosphorescence
- Temporal effects
 - Motion blur
 - Depth of field
 - Anti-aliasing







Advanced Graphics – Light Transport

Introduction

tics & (depth < MANDER

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Psurat3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

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

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

Anti-aliasing

Adding anti-aliasing to a Whitted-style ray tracer:

Send multiple primary rays through each pixel, and average their result.

Problem:

- How do we aim those rays?
 - What if all rays return the same color?









sics & (depth < MaxDer

: = inside ? 1 + 1.0 ht = nt / nc, ddn → bs2t = 1.0f - n∺t ↑ 0, N); ð)

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

= * diffuse; = true;

efl + refr)) && (depth < NAXDEPTH

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

AXDEPTH)

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

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) (cost

andom 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:

Anti-aliasing – Sampling Patterns

Adding anti-aliasing to a Whitted-style ray tracer:

Send multiple primary rays through each pixel, and average their result.

Problem:

- How do we aim those rays?
- What if all rays return the same color?





Anti-aliasing – Sampling Patterns

tics & (depth < ™000000

c = inside / i nt = nt / nc, r os2t = 1.0f - r O, N); ð)

ata = nt - nc, b = atTr = 1 - (R0 + 1) Fr)R = (D = nnt - N

= * diffuse = true:

efl + refr)) && (de

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

AXDEPTH)

survive = SurvivalProbabili estimation - doing it prop Hf; radiance = SampleLight(&ra e.x + radiance.y + radiance

v = true; at brdfPdf = EvaluateDiffus at3 factor = diffuse * INVF at weight = Mis2(directPdf at cosThetaOut = dot(N, L E * ((weight * cosThetaOut E * ()

andom walk - done pro /ive)

; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, 4R,) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:





ALCONTRACTOR OF A

Anti-aliasing – Sampling Patterns

tics & (depth < ≫ocoori

z = inside 7 1 ht = nt / nc, ddn bs2t = 1.0f - nnt D, N); D)

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

= * diffuse; = true;

. :fl + refr)) && (dept)

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

AXDEPTH)

survive = SurvivalProbabili estimation - doing it prop if; radiance = SampleLight(&ra 2.x + radiance.y + radiance

w = true; at brdfPdf = EvaluateDiffus at3 factor = diffuse * INVF at weight = Mis2(directPdf at cosThetaOut = dot(N, L E * ((weight * cosThetaOut

andom walk - done properl /ive)

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



Anti-aliasing – Sampling Patterns

hics & (depth < Moore

nt = nt / nc, c ps2t = 1.0f - n D, N); D)

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

= * diffuse; = true;

. :fl + refr)) && (dept

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

AXDEPTH)

survive = SurvivalProbabili estimation - doing it prof if; radiance = SampleLight(&r: 2.x + radiance.y + radiance

w = true; at brdfPdf = EvaluateDiffus at3 factor = diffuse * INVF at weight = Mis2(directPdf at cosThetaOut = dot(N, L E * ((weight * cosThetaOut

andom walk - done properly,..... /ive)

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





sics & (depth < ∧oxoon

: = inside ? 1 1 1 2 ht = nt / nc, ddn bs2t = 1.0f - nnt * 1 2, N); 3)

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

= * diffuse; = true;

efl + refr)) && (depth < NAXDEPTH

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

AXDEPTH)

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

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 s /ive)

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

Anti-aliasing – Sampling Patterns

Adding anti-aliasing to a Whitted-style ray tracer:

Send multiple primary rays through each pixel, and average their result.

Problem:

- How do we aim those rays?
- What if all rays return the same color?

More info: https://mynameismjp.wordpress.com/2012/10/24/msaa-overview





Advanced Graphics – Light Transport

Introduction

nics & (depth < ⊅00000

z = inside ? 1 1.0 ht = nt / nc, ddm bs2t = 1.0f - n⊓t = on 0, N); 8)

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDE

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

AXDEPTH)

survive = SurvivalProbability(diffus estimation - doing it properly if; radiance = SampleLight(&rand, I, &L

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

w = 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 Sovi /ive)

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

Whitted

Missing:

- Area lights
- Glossy reflections
- Caustics
- Diffuse interreflections
 - Diffraction
- Polarization
- Phosphorescence
- Temporal effects

Motion blur
 Depth of field
 Anti-aliasing





Distribution Ray Tracing*

tics & (depth < MODEFT

: = inside ? 1 | ... ht = nt / nc, ddn | ... bs2t = 1.0f - nnt " n 2, N); ≷)

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

= * diffuse; = true;

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

), 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) && (dottoor)

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

^{andom walk - done properly, closely following set ^{vive)} *: Distributed Ray Tracing, Cook et al., 1984}

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



Soft shadows



Distribution Ray Tracing*

nics & (depth < M00000

c = inside 7 1 1 1 7 ht = nt / nc, ddn bs2t = 1.0f - nnt 7 7 0, N); 8)

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

= * diffuse; = true;

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

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) && (closed)

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 Sector *: Distributed Ray Tracing, Cook et al., 1984

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



Glossy reflections

A PARTY AND A PART

Distribution Ray Tracing*

hics & (depth < MoxDarm

c = inside 7 1 1 1 0 ht = nt / nc, ddn 9 0 ps2t = 1.0f - nnt 7 0 D, N); Ø)

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly if; radiance = SampleLight(&rand, I, &L, &L) ax + 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 Service *: Distributed Ray Tracing, Cook et al., 1984

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







Distribution Ray Tracing*

nics & (depth < →000811

c = inside 7 1 1 1 0 ht = nt / nc, ddn 9 0 ps2t = 1.0f - nnt 7 0 D, N); Ø)

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

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

AXDEPTH)

survive = SurvivalProbability(diffuse
estimation - doing it properly
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 Service *: Distributed Ray Tracing, Cook et al., 1984

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







ics & (depth < PANDSOT

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

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEDI

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.

v = true;

at brdfPdf = EvaluateDiffuse(L, N) Psurela 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); ;;;;n = true;

Distribution Ray Tracing

Whitted-style ray tracing is a *point sampling* algorithm:

- We may miss small features
- We cannot sample areas

Area sampling:

- Anti-aliasing: one pixel
- Soft shadows: one area light source
 - Glossy reflection: directions in a cone
 - Diffuse reflection: directions on the hemisphere



tics & (depth < Macca

: = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nmt 0, N); 3)

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

= * diffuse = true;

--

D, N); refl * E * dif = true;

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, N) Paul 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, apd) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Area Lights

Visibility of an area light source:

$$V_A = \int_A V(x, \omega_i) \, d\omega_i$$

Analytical solution case 1:

$$V_A = A_{light} - A_{light \cap sphere}$$

Analytical solution case 2:

 $V_A = ?$







nics & (depth < Motosa

: = inside ? 1 | 1 | 0 ht = nt / nc, ddn bs2t = 1.0f - nnt 0, N); 3)

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

= * diffuse = true;

efl + refr)) && (depth < MAX

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

AXDEPTH)

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

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 fo /ive)

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

Approximating Integrals

An integral can be approximated as a Riemann sum:

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

Note that the intervals do not need to be uniform, as long as we sample the full interval. If the intervals are uniform, then

$$\sum_{i=1}^{N} f(t_i) \, \Delta_i = \Delta_i \sum_{i=1}^{N} f(t_i) = \frac{B-A}{N} \sum_{i=1}^{N} f(t_i) \, .$$

Regardless of uniformity, restrictions apply to N when sampling multi-dimensional functions (ideally, $N = M^d$, $M \in \mathbb{N}$). Also note that aliasing may occur if the intervals are uniform.







tics & (depth < ≯VXDE

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

v = true;

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

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

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

Monte Carlo Integration

Alternatively, we can approximate an integral by taking random samples:

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

Here, $X_1 .. X_N \in [A, B]$. As *N* approaches infinity, V_A approaches the *expected value* of f(X).

Unlike in Riemann sums, we can use arbitrary *N* for Monte Carlo integration, regardless of dimension.





sics & (depth < >∨∞ccri

: = inside ? 1 1 1 1 ht = nt / nc, ddn 1 552t = 1.0f - nnt " r 2, N); 3)

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPT

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

AXDEPTH)

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

w = 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 fol /ive)

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

Monte Carlo Integration of Area Light Visibility

To estimate the visibility of an area light source, we take *N* random point samples.

In this case, 5 out of 6 samples are unoccluded:

$$V \approx \frac{1}{6} \left(1 + 1 + 1 + 0 + 1 + 1 \right) = \frac{5}{6}$$

Properly formulated using a MC integrator:

$$V = \int_{\mathcal{S}^2} V(p) \, dp \approx \frac{1}{N} \sum_{i=1}^N V(P)$$

With a small number of samples, the variance in the estimate shows up as noise in the image.

Q: Where did the $\frac{B-A}{N}$ go? A: the domain of the visibility function is [0..1], so B - A = 1.



sics & (depth < ≫occorr

: = inside ? 1 1 1 1 ht = nt / nc, ddn 4 552t = 1.0f - nnt 7 n 2, N); 3)

at a = nt - nc, b = nt = n_{c} at Tr = 1 - (R0 + (1 - R0) Tr) R = (D = nnt - N = (300)

= * diffuse; = true;

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

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) &&

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) * (reference);

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

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

Monte Carlo Integration of Area Light Visibility

We can also use Monte Carlo to estimate the contribution of multiple lights:

1. Take the average of *N* samples from each light source;

 $E(x \leftarrow) = \frac{1}{N} \sum_{i} \sum_{i} L_i V(x \leftrightarrow l_i)$

2. Sum the averages.

No averaging here: multiple lights are additive.

ln

 $\boldsymbol{\chi}$

 l_1



tics & (depth < PVCDOF

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0 Γ r) R = (D = nnt - N - (10)

= * diffuse = true;

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

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

AXDEPTH)

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

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

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

Monte Carlo Integration of Area Light Visibility

Alternatively, we can just take *N* samples, and pick a random light source for each sample.

 $E(x \leftarrow) = \frac{2}{N} \sum_{i=1}^{N} L_Q V_Q(P), \qquad Q \in \{1,2\}$

 $= \frac{1}{N} \sum_{i=1}^{N} \frac{L_Q V_Q(P)}{(0.5)} \qquad \begin{array}{c} \text{Probability of} \\ \text{sampling light } L_Q \end{array}$

 $\boldsymbol{\chi}$



tics & (depth < NACCET

c = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nmt * 2, N); 3)

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

= * diffuse = true;

efl + refr)) && (depth < NOCCEPT

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

AXDEPTH)

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

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) * (c);

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

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

Monte Carlo Integration of Area Light Visibility

We obtain a better estimate with fewer samples if we do not treat each light equally.

In the previous example, each light had a 50% probability of being sampled. We can use an arbitrary probability, by dividing the sample by this probability.

 $E(x \leftarrow) = \frac{1}{N} \sum_{i=1}^{N} \frac{L_Q V_Q(P)}{\rho_Q}, \qquad \sum \rho_Q = 1, \rho_Q > 0$



 χ

sics & (depth < ≫000000

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

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

= * diffuse; = true;

-:**fl + refr)) && (depth < MAXDE**

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

AXDEPTH)

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

v = 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) = (realized)

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:

Distribution Ray Tracing

Key concept of distribution ray tracing:

We estimate integrals using Monte Carlo integration.

Integrals in rendering:

- Area of a pixel
 - Lens area (aperture)
 - Frame time
 - Light source area
- Cones for glossy reflections
- Wavelengths

- ...





tics & (depth < Mox000

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

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPIII

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) && (doing)

v = true; at brdfPdf = EvaluateDiffuse(L, N) Pourvive 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, &R, &pdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Open Issues

Remaining issues:

- Energy distribution in the ray tree / efficiency
- Diffuse interreflections



ics & (depth < Mox000

= inside ? 1 1 1 0 ht = nt / nc, ddn bs2t = 1.0f - nnt = n D, N); B)

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

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:

Today's Agenda:

- Introduction
- The Rendering Equation
- Light Transport



nics & (depth < Motoort

z = inside ? 1 ht = nt / nc, ddn os2t = 1.0f - nnt = nnt D, N); 3)

at a = nt - nc, b = nt at Tr = 1 - (R0 + (1 - R0) Tr) R = (D = nnt - N - (30)

= * diffuse; = true;

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

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

AXDEPTH)

sion = true:

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

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

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

Whitted, Cook & Beyond

Missing in Whitted:

Cook:

- Area lights
- Glossy reflections
- Caustics
- Diffuse interreflections
- Diffraction
 - Polarization
 - Phosphorescence
 - Temporal effects
 - Motion blur Depth of field
 - Anti-aliasing

- ✓ Area lights
- ✓ Glossy reflections
- × Caustics
- × Diffuse interreflections
- × Diffraction
- Polarization
- Phosphorescence
- × Temporal effects
- ✓ Motion blur
- Depth of field
- ✓ Anti-aliasing









Whitted, Cook & Beyond

Cook's solution to rendering:

c = inside / i nt = nt / nc, ddn os2t = 1.0f - nnt 0, N); 3)

at a = nt - nc, b = nt + r 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 if; radiance = SampleLight(&rand, I, &L, &light) 2.x + radiance.y + radiance.z) > 0) && (doing)

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) (Ps

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:

$\int_{A_{pixel}} \int_{A_{lens}} \int_{T_{frame}} \int_{\Omega_{glossy}} \int_{A_{light}} \dots$

Sample the many-dimensional integral using Monte Carlo integration.

Ray optics are still used for specular reflections and refractions: The ray tree is not eliminated.



God's Algorithm

at a = nt

), N); refl * E * diffuse;

AXDEPTH)

survive = SurvivalProbability(dif lf;

radiance = SampleLight(&rand, I, e.x + radiance.y + radiance.z)

v = true;

at brdfPdf = EvaluateDiffuse(L, 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, & urvive; pdf; 1 = E * brdf * (dot(N, R) / pdf); sion = true:

1 room 1 bulb 100 watts 10²⁰ photons per second

Photon behavior:

- Travel in straight lines
- Get absorbed, or change direction:
 - Bounce (random / deterministic)
 - Get transmitted
- Leave into the void Get detected





hics & (depth < Mot00⊓

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N - (0.0

= * diffuse; = true;

efl + refr)) && (depth < MOXDEPT

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

AXDEPTH)

sion = true:

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

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) (read);

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

t3 brdf = SampleDiffuse(diffuse, N, r1, r2, 8R, 8pd rvive; pdf; n = E * brdf * (dot(N, R) / pdf);



hics & (depth < >∨∞cc

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPIN

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

AXDEPTH)

survive = SurvivalProbability(diffu .estimation - doing it properly if; radiance = SampleLight(&rand, I, &L 2.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 /ive)

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

God's Algorithm - Mathematically

A photon may arrive at a sensor after travelling in a straight line from a light source to the sensor:

 $L(s \leftarrow x) = L_E(s \leftarrow x)$

Or, it may be reflected by a surface towards the sensor:

$$L(s \leftarrow x) = \int_{A} f_{r}(s \leftarrow x \leftarrow x') L(x \leftarrow x') G(x \leftrightarrow x') dA(x')$$

Those are the options. Adding direct and indirect illumination together:

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





God's Algorithm - Mathematically

nics & (depth < MOOS−1

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

= * diffuse = true;

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

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

AXDEPTH)

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

v = 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 Sec. /ive)

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







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

& (depth < MoxConn = inside ? 1

os2t = 1.0f - n=t 0, N); 0)

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

= * diffuse; = true;

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

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

(AXDEPTH)

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

v = 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) (mad

andom walk - done properly, closely follow: *: The Rendering Equation, Kajiya, 1986

; t3 Brdf = SampleDiffuse(diffuse, N, r1, r2, 8R, 8pdf) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

The Rendering Equation*:

- Light transport from lights to sensor
- Recursive
- Physically based

The equation allows us to determine to which extend rendering algorithms approximate real-world light transport.

ightoindata n





ics & (depth < Mox000

= inside ? 1 1 1 0 ht = nt / nc, ddn bs2t = 1.0f - nnt = n D, N); B)

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

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:

Today's Agenda:

- Introduction
- The Rendering Equation
- Light Transport



Light Transport Quantities

ics (depth < MAXDEFT

: = inside ? 1 1 1 1 ht = nt / nc, ddn 4 bs2t = 1.0f - nnt 4 2, N); 3)

at a = nt - nc, b = $M_{\pm} = 0$ at Tr = 1 - (R0 + (1 - R0) Fr) R = (D = nnt - N = (3)

= * diffuse = true;

efl + refr)) && (depth < MODE

), 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) &&

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 fol /ive)

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

Light Hansport Quant

Radiant flux - Φ :

"Radiant energy emitted, reflected, transmitted or received, per unit time."

Units: watts = joules per second $W = J s^{-1}$.

Simplified particle analogy: number of photons.

Note: photon energy depends on electromagnetic wavelength: $E = \frac{hc}{\lambda}$, where h is Planck's constant, c is the speed of light, and λ is wavelength. At $\lambda = 550$ nm (yellow), a single photon carries 3.6 * 10⁻¹⁹ joules.





sics & (depth < MoOCC

: = inside ? 1 ht = nt / nc, ddn = 1 os2t = 1.0f - nnt " ∩ 2, N); 3)

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

= * 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, &L, &L) 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) (real

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

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

Light Transport Quantities

In a vacuum, radiant flux emitted by a point light source remains constant over distance:

A point light emitting 100W delivers 100W to the surface of a sphere of radius r around the light. This sphere has an area of $4\pi r^2$; energy per surface area thus decreases by $1/r^2$.

In terms of photons: the density of the photon distribution decreases by $1/r^2$.





nics & (depth < ⊅00000

: = inside ? 1 ht = nt / nc, ddn os2t = 1.0f - nnt ? 2, N); 3)

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEPT

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

AXDEPTH)

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

w = 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) (real

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

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

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.



= true:

), N); refl * E * diffuse;

AXDEPTH)

survive = SurvivalProbability lf: radiance = SampleLight(&rand x + radiance.v + 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 f /ive)

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

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.



nics & fdepth < MAXONE

: = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt n D, N); 3)

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

= * diffuse = true;

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

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

AXDEPTH)

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) (real

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

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

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 Quantities

Converting radiance to irradiance:

 $E = L \cos \theta$

= * 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) 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) (read);

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:





Light Transport

hics & (depth < Mot0€)

t = inside 7 1 1 1 1 nt = nt / nc, ddn - 1 ns2t = 1.0f - n⊓t - n 2, N); 2)

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

= * diffuse; = true;

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

AXDEPTH)

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

w = true; at brdfPdf = EvaluateDiffuse(<u>L. N) _ _ _</u>

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:

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

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

Radiance Radiance

Radiance

Irradiance

BRDF



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:

- radiance = SampleLight(&rand, I, &L, &llando .x + radiance.y + radiance.z) > 0) && (dottoo
- v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pi at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * (((weight * cosThetaOut) / directPdf)

survive = SurvivalProbability(dif

ata = nt -

), N);

AXDEPTH)

lf;

efl + refr)) && (depth

refl * E * diffuse;

andom walk - done properly, closely following rive)

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

Should be positive: f_r(ω_o, ω_i) ≥ 0
Helmholtz reciprocity should be obeyed: f_r(ω_o, ω_i) = f_r(ω_i, ω_o)
Energy should be conserved: ∫_o f_r(ω_o, ω_i) cos θ_o dω_o ≤ 1



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

Relation between real-world light transport and the RE:

- 1. Each sensor element registers an amount of photons arriving from the first surface visible though that pixel.
- 2. This surface may be emissive, in which case it produced the sensed photons.
- 3. This surface may also reflect photons, arriving from "other surfaces" in the scene.
- 4. For the "other surfaces": goto 2.

Real Provide State

radiance = SampleLight(&rand, I, &L, &L)
e.x + radiance.y + radiance.z) > 0) & 0
w = true;
at brdfPdf = EvaluateDiffuse(L, N) * P:
at3 factor = diffuse * INVPI;
at weight = Mis2(directPdf, brdfPdf);

survive = SurvivalProbability(diff)

at a = nt

), N);

AXDEPTH)

lf;

efl + refr)) && (depth < M

refl * E * diffuse;

at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * (ma

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: ics & (depth < Mox000

= inside ? 1 1 1 0 ht = nt / nc, ddn bs2t = 1.0f - nnt = n D, N); B)

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

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:

Today's Agenda:

- Introduction
- The Rendering Equation
- Light Transport



hics & (depth < NoCOS

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

= * diffuse = true;

efl + refr)) && (depth < MODEPTI

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

AXDEPTH)

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

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

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:

INFOMAGR – Advanced Graphics

Jacco Bikker - November 2021 - February 2022

END of "Light Transport"

next lecture: "The Perfect BVH"

