tics & (depth < ≥vocci

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

w = true; at brdfPdf = EvaluateDiffuse(L, N) Psurvu 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 6 - "Path Tracing"

I(x',x'')dx''

Welcome!



ics & (depth < Mox000

= inside ? l |]] ht = nt / nc, ddn bs2t = 1.0f - nnt = o D, N); B)

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPIN

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) > @______

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)

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

Path Tracing



"three-point"

= true:

efl + refr)) && (dep

), N);

AXDEPTH)

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

formulation"

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 foll /ive)

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

Previously in Advanced Graphics

The Rendering Equation:

S

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

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

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

BRDF: takes irradiance (from *x*'), calculated by: $L(x \leftarrow x') G(x \leftrightarrow x')$ returns radiance (towards s).





Introduction

Previously in Advanced Graphics

The Rendering Equation (hemispherical formulation):

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

= * diffuse; = true:

at a = nt

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, &L 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 Sovi /ive)

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

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

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

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



Introduction

sics & (depth < MANDOR

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

at a = nt - nc, b = nt + nc at Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N + 600

= * diffuse; = true;

efl + refr)) && (depth < M

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

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

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

Previously in Advanced Graphics

Particle transport:

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

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

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

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





Introduction

sics & (depth < Monorr

: = inside ? 1 1 1 2 ht = nt / nc, ddn - 1 ps2t = 1.0f - nmt - 1 2, 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 < MODEPTI

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly f; radiance = SampleLight(&rand, I, &L, &L) 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) = (red

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

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

Previously in Advanced Graphics

Probabilities:

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





Introduction

nics & (depth < Notici⊓

c = inside ? 1 1 1 1 nt = nt / nc, ddn 1 ps2t = 1.0f - nmt 1 0, N); 3)

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

= * diffuse = true;

• efl + refr)) && (depth < MAXE

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

AXDEPTH)

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

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 Sa /ive)

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

Previously in Advanced Graphics

Turning physics into code:

- 'Flux' becomes 'photons per second'
- Count becomes probability

To complicate things, radiance and irradiance are 'continuous values' or 'densities':

"At point X we have a density of 50 photons per m2"

"In direction ω_o , we have a flow of 10 photons per steradian"

Likewise, our probabilities are going to be *densities*.





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:

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

fuse); losely fo Properties:

Should be positive: $f_r(\omega_o, \omega_i) \ge 0$

Helmholtz reciprocity should be obeyed: $f_r(\omega_o, \omega_i) = f_r(\omega_i, \omega_o)$ $f_r(\omega_o, \omega_i) = f_r(\omega_i, \omega_o)$ Helmholtz reciprocity should be conserved: $\int_{\Omega} f_r(\omega_o, \omega_i) \cos \theta_o \, d\omega_o \leq 1$

diffuse, N, r1, r2, &R, &pdi

R) / pdf);

&lightDis
& (dot(N.



Bidirectional Reflectance Distribution Function

The diffuse BRDF is:

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

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

Energy conservation: $E_o \leq E_i$

(asely for Suppose we have a directional light parallel to \vec{n} , with (slightDis: (cot(intensity 1. Then: $E_i = L_i = 1$. Suppose our BRDF = $\frac{albedo}{1}$.

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

Now:
$$\int_{\Omega} \cos \omega_o \, d\omega_o = \pi \rightarrow E_o = \pi E_i$$

diffuse, N, r1, r2, &R, &pd

sely followi





Bidirectional Reflectance Distribution Function

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

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

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

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



ω_i

Ń

Introduction

Previously in Advanced Graphics

Monte Carlo integration:

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

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

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

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

andom walk - done properly, closely foll /ive)

efl + refr.)) && (de

refl * E * diffuse = true;

survive = SurvivalP

radiance = SampleLi

at weight = Mis2(directPdf at cosThetaOut = dot(N, L

), N);

AXDEPTH)

at3 factor

st3 brdf = SampleDiffuse(diffuse, N, r1
urvive;
pdf;
n = E * brdf * (dot(N, R) / pdf);
sion = true;



ics & (depth < Mox000

= inside ? l |]] ht = nt / nc, ddn bs2t = 1.0f - nnt = o D, N); B)

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPIN

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) > @______

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)

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

Path Tracing



Path Tracing

ics & (depth < Notes

: = inside ? 1 + . . ht = nt / nc, ddn os2t = 1.0f - nnt 2, 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 < MAXDEDI

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) &&

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 Soni /ive)

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

Solving the Rendering Equation

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

Let's start with direct illumination:

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

 ω_{o}

Answer:

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





tics & (depth < ™0000

: = inside ? 1 = 1.0 ht = nt / nc, ddn bs2t = 1.0f - n⊓t 2, 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 < MAXDEDI

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

AXDEPTH)

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

v = true;

at brdfPdf = EvaluateDiffuse(L, N) Psurvi 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)

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

Solving the Rendering Equation

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

Let's start with direct illumination:

For a screen pixel, diffuse surface point p with norm What is the radiance travelling via p towards the eyestimate the second state of the explanation of the second state of the sec

Answer:

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

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

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

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

In other words: the sum of radiance (scaled by $\cos \theta_i$ to convert to irradiance) arriving from all directions over the hemisphere, divided by π .

- Q: What about distance attenuation?
- A: A far-away light is found by fewer directions ω_i : it's **solid angle** on the hemisphere is smaller.
- Q: What happened to ω_o ?

 ω_o

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

Path Tracing

Direct Illumination

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

We can solve this integral using Monte-Carlo integration:

Chose N random directions over the hemisphere for p

 ω_{o}

- Find the first surface in each direction by tracing a ray
- If the surface is light emitting: add it to the sum
- Divide the sum by N and multiply by 2π

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



 ω_i

p

andom walk - done properly, closely following Same ; ; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, i ;rvive; pdf;

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

survive = SurvivalProbability(di

at brdfPdf = EvaluateDiffuse(L, M at3 factor = diffuse * INVPI;

at weight = Mis2(directPdf, brdfPdf) at cosThetaOut = dot(N, L);

E * ((weight * cosThetaOut) / directPdf)

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

at a = nt

), N);

AXDEPTH)

v = true:

lf:

refl * E * diffuse;

17

Path Tracing

Direct Illumination

efl + refr

), N = true

(AXDEPTH)

survive =

lf; radiance = samileLighL is per sr; $L_d(p, \omega_i)$ is proportional

v = true: at brdfPdf = EvaluateDiffuse(L) at a factor = diffuse from p, SO: $\sim (A_{disc}/r^2)$.

at weight = Mis2(directPdf at cosThetaOut

E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely foll /ive)

at3 brdf = SampleDiffuse(diffuse, N, r1, r2, 8 urvive; pdf; 1 = E * brdf * (dot(N, R) / pdf); sion = tru

$$f_0(p,\omega_o) \approx \frac{2\pi}{N} \sum_{i=1}^N f_r(p,\omega_o,\omega_i) L_d(p,\omega_i) \cos\theta$$

Questions:

to the solid angle of the light as seen

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

 ω_{o}

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

 ω_i

p

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



18



Path Tracing

Direct Illumination

tics & (depth < Motoort

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

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

= * diffuse; = true;

efl + refr)) && (depth < MODEP)

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) &

w = true; at brdfPdf = EvaluateDiffuse(L, N) * F 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, &pdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); ;;;n = true;

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

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

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

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

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





Direct Illumination





Using Monte-Carlo:

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

where

- estimation doing It property if; radiance = SampleLight(&rand, I, &L, &L, 2.x + radiance.y + radiance.z) > 0) &&
- v = true;
- at brdfPdf = EvaluateDiffuse(L, N at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPd
- at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directF
- andom walk done properly, closely foll /ive)
- ; at3 brdf = SampleDiffuse(diffuse, N, prvive; pdf; n = E * brdf * (dot(N, R) / pdf);

• $L_d^J(p, P)$ is the direct light towards p from random point P on random light J

- *L_d*(*p*, *r*) is the uncertaint towards *p* from random point *r* on *r*.
 V(*p* ↔ *P*) is the mutual visibility between *p* and *P*
- $A_{L_{d}}^{J}$ is the area of this light source
- $(A_{L_{d}}^{J} \cos \theta_{o})/(||p P||^{2})$ is the area of the light source projected on the
 - hemisphere, which <u>approximates</u> the solid angle of the light.



 ω_o

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

Recall:

Direct Illumination

ics (depth < NOCOST

: = inside } 1 | | | | ht = nt / nc, ddm bs2t = 1.0f - nnt = r D, N); D)

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

= * diffus = true;

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

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

AXDEPTH)

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

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

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

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

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

. By random sampling the hemisphere:

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

2. By sampling the lights directly:

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

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



at a = nt

), N);

AXDEPTH)

v = true;

/ive)

efl + refr)) && (depth

survive = SurvivalProbability(diff

radiance = SampleLight(&rand, I,) .x + radiance.y + radiance.z) > 0)

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 foll

refl * E * diffuse;

Direct Illumination

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

By random sampling the hemisphere:

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

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

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

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

at3 brdf = SampleDiffuse(diffuse, N, r1, r2, 8R, 8 urvive; pdf; 1 = E * brdf * (dot(N, R) / pdf); sion = true:

Path Tracing

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

Verification

Method 1 in a small C# ray tracing framework:

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

```
Vector3 R = RTTools.DiffuseReflection( ray.N );
Ray rayToHemisphere = new Ray( I + R * EPSILON, R, 1e34f );
Scene.Intersect( rayToHemisphere );
if (rayToHemisphere.objIdx == LIGHT)
```

```
Vector3 BRDF = material.diffuse * INVPI;
float cos_i = Vector3.Dot( R, ray.N );
return 2.0f * PI * BRDF * Scene.lightColor * cos_i;
```



at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * andom walk - done properly, closely followi /ive)

survive = SurvivalProbability(diff

radiance = SampleLight(&rand, I,

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

at brdfPdf = EvaluateDiffuse(| at3 factor = diffuse * INVPI;

at a = nt - nc,

efl + refr)) && (dept)

refl * E * diffuse;

), N);

AXDEPTH)

v = true;

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

at a = nt - nc, l

efl + refr)) && (depth

survive = SurvivalProbability(diff

radiance = SampleLight(&rand, I,)

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

at brdfPdf = EvaluateDiffuse(L, N) at3 factor = diffuse * INVPI;

at weight = Mis2(directPdf, brdfPdf) at cosThetaOut = dot(N, L);

refl * E * diffuse;

), N);

AXDEPTH)

v = true;

/ive)

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

Verification

Method 2 in a small C# ray tracing framework:

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



at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, A) urvive; pdf; 1 = E * brdf * (dot(N, R) / pdf); sion = true:

hics & (depth < 2000)

c = inside / 1 |) ht = nt / nc, ddn bs2t = 1.0f - nnt / r D, N); ð)

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

= * diffuse = true;

efl + refr)) && (depth < NAMERIA

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

AXDEPTH)

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

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





hics & (depth < 2000)

c = inside / 1 | | | ht = nt / nc, ddh ps2t = 1.0f = nnt 2, N); 2)

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

= * diffuse = true;

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

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

AXDEPTH)

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

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





hics & (depth < ⊅000

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

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

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

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)

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

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







rics & (depth < 2000

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

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

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 S /ive)

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





Path Tracing

ics & (depth < MAXON

: = inside 7 1 1 1 1 ht = nt / nc, ddn 1 1 ss2t = 1.0f - n⊓t 1 2, N); 3)

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

= * diffuse; = true;

• efl + refr)) && (depth < MODEP

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

AXDEPTH)

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) ((***

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

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

Rendering using Monte Carlo Integration

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

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

Observations:

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



Path Tracing

ics & (depth < ⊅0000

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

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

= * diffuse = true;

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

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) Psurvi 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:

Indirect Light

Returning to the full rendering equation:

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

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

$$f_{o}(p,\omega_{o}) = \int_{\Omega} f_{r}(p,\omega_{o},\omega_{i}) L_{d}(p,\omega_{i}) \cos \theta_{i} d\omega_{i}$$

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



Path Tracing

Indirect Light

Let's expand / reorganize this:

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

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

AXDEPTH)

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

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, &p urvive; pdf; i = E * brdf * (dot(N, R) / pdf); sion = true:

$$+ \int_{\Omega} L_{E}(y, \omega_{o}^{y}) f_{r}(x, \omega_{o}^{x}, \omega_{i}^{x}) \cos \theta_{i}^{x} d\omega_{i}^{x}$$
direct light on x

$$+ \int_{\Omega} \int_{\Omega} L_{E}(z, \omega_{o}^{q}) f_{r}(y, \omega_{o}^{q}, \omega_{i}^{q}) \cos \theta_{i}^{q} f_{r}(x, \omega_{o}^{x}, \omega_{i}^{x}) \cos \theta_{i}^{x} d\omega_{i}^{x} d\omega_{i}^{q}$$
1st bounce

$$+ \int_{\Omega} \int_{\Omega} \int_{\Omega} \dots$$
2nd bounce

 \approx ...

Ν

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

31

 $\boldsymbol{\chi}$

Z

1st bounce

2nd bounce

V

 $\omega_o^q \quad \omega_i^\chi$

 ω_o^{χ}

Path Tracing

ics & (depth < MODO

: = inside ? 1 | 1 / ht = nt / nc, ddm | ps2t = 1.0f - nmt | r 2, N); 3)

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

= * diffuse; = true;

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

), 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) 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 Sov /ive)

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

Indirect Light

One particle finding the light via a surface:

I, N = Trace(ray); R = DiffuseReflection(N); lightColor = Trace(new Ray(I, R)); return dot(R, N) * $\frac{albedo}{\pi}$ * lightColor * 2π ;

One particle finding the light via two surfaces:

I1, N1 = Trace(ray);
R1 = DiffuseReflection(N1);
I2, N2 = Trace(new Ray(I1, R1));
R2 = DiffuseReflection(N2);
lightColor = Trace(new Ray(I2, R2));
return dot(R1, N1) *
$$\frac{albedo}{\pi}$$
 * 2π * dot(R2, N2) * $\frac{albedo}{\pi}$ * 2π * lightColor;

х

Z

V

Path Tracing

Path Tracing Algorithm

ata = nt -

efl + refr)) && (depth c

refl * E * diffuse;

AXDEPTH)

survive = SurvivalProbability(diff radiance = SampleLight(&rand, I, &l \mathbf{r} .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 follow /ive)

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

Color Sample(Ray ray)

// trace ray

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



ν



Z

tics & (depth < ™000

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

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

= * diffuse = true:

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

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly, if; radiance = SampleLight(&rand, I, &t, &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) (r

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

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





tics & (depth < NOCD

at a = nt - nc, b = Ht = n at Tr = 1 - (R0 + (1 - R0 Fr) R = (D - nnt - N - (33)

= * diffuse = true:

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

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

AXDEPTH)

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

v = true;

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

andom walk - done properly, closely following /ive)

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





tics & (depth < ™000

t = inside } l ht = nt / nc, ddn bs2t = 1.0f - n⊓t - o D, N); ∂)

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

= * diffuse = true;

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

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

AXDEPTH)

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

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





tics & (depth < NOCD

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

= * diffuse = true;

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

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

AXDEPTH)

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

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)

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

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





Particle Transport

The random walk is analogous to particle transport:

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

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

Notice that longer paths tend to return less energy.

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



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)

at a = nt

), N);

AXDEPTH)

lf;

refl * E * diffuse;

survive = SurvivalProbability

radiance = SampleLight(&rand,

e.x + radiance.y + radiance.z)

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

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

Path Tracing

sics & (depth < Monner

: = 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) Γ r) R = (D = nnt - N - (d0)

= * 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

♥ = 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 Sec. /ive)

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

Particle Transport - Mirrors

Handling a pure specular surface:

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

Question:

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

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



Path Tracing

ics & (depth < NADD

: = inside ? 1 1 1 1 ht = nt / nc, ddn bs2t = 1.0f - nmt ? (0, N); 8)

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

= * diffuse = true;

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

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

AXDEPTH)

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

w = true; at brdfPdf = EvaluateDiffuse(L, N) Pservive 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 Sou. /ive)

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

Particle Transport - Glass

Handling dielectrics:

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

A particle must chose.

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

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



hics & (depth < MOCOS

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

= * diffuse; = true;

. efl + refr)) && (depth < MONDER []

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

AXDEPTH)

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

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





tics ≹ (depth < Mox00

: = inside ? | ht = nt / nc, ddn dd os2t = 1.0f - n∺t a D, N); ∂)

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

= * diffuse = true:

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

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

AXDEPTH)

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

v = 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)

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

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





ics & (depth < Mox000

= inside ? l |]] ht = nt / nc, ddn bs2t = 1.0f - nnt = o D, N); B)

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPIN

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) > @______

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)

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

Path Tracing



hics & (depth < NoCOS

at a = nt - nc, b = nt - nc 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, &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) (1800)

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 "Path Tracing"

next lecture: "GPU Ray Tracing (1)"

