tic: ≰ (depth < NA

: = inside / l it = nt / nc, dd os2t = 1.0f - nn 0, N); 8)

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

= diffuse = true;

efl + refr)) && (depth k HANDII

D, N); -efl * E * diffus = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it properly if; adiance = SampleLight(%rand, I, I) e.x + radiance.y + radiance.y) = 0.000

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pourch st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * 0000

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

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

INFOGR – Computer Graphics

J. Bikker - April-July 2016 - Lecture 10: "Shading Models"

Welcome!



tic: € (depth < 1000

: = inside / 1 it = nt / nc, dde os2t = 1.8f - nn: 0, N); 0)

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

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I, I, 2.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * read

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, sta urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Introduction
- Light Transport
- Materials
- Sensors
- Shading





INFOGR – Lecture 10 – "Shading Models"

Introduction

= true:

fl + refr)) && (depth

), N); efl * E * diffuse; = true:

AXDEPTH)

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

v = true: st brdfPdf = EvaluateDiffuse(L, N st3 factor = diffuse * INVPI at weight = Mis2(directPdf, brdfPdf at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely felle /ive)

at3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, N rvive; 1 = E * brdf * (dot(N, R) / pdf);

The Quest for (Photo-)Realism

Objective in modern games

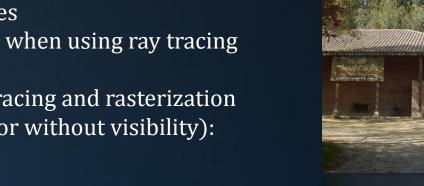
Other factors:

Important improvements when using ray tracing

The core algorithms of ray tracing and rasterization model light transport (with or without visibility):

 $L(p \to r) = L_e(p \to r) + \sum_{i=1}^{N_L} L(q_i \to p) f_r(q_i \to p \to r) G(q_i \leftrightarrow p)$

- Material interactions
- Light models
- Sensor models







Material interactions

tic: k (depth < 10.⊂

: = inside / ... nt = nt / nc. os2t = 1.0f), N); 3)

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

= diffuse = true;

efl + refr)) && (dept

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

AXDEPTH)

survive = SurvivalProbabi estimation - doing it pr if; radiance = SampleLight(&mand, 1 e.x + radiance.y + radiance.z) = 0

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

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, s) pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:







Material interactions

fic: k (depth < 10000 -

: = inside / 1 ht = nt / nc, bs2t = 1.0f b, N); b)

st a = nt - nc, st Tr = 1 - (88 Tr) R = (0 * nn

= diffu = true;

efl + refr))

), N); ~efl * E * di = true;

AXDEPTH)

survive = Surviva estimation - doi ff; radiance = Sample e.x + radiance.y

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Promise st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) *

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

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





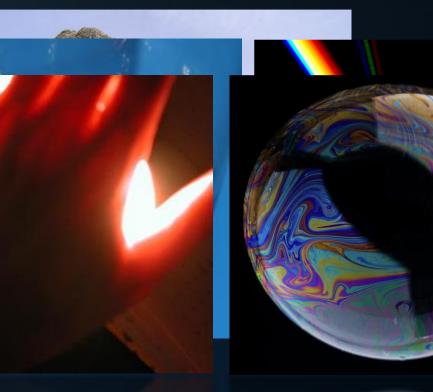
Material interactions



st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) *

andom walk - done properly, closely following /ive)

; ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, IR, los) pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;





Light models

tc: k (depth < ™

nt = nt / nc, / os2t = 1.0f 0, N); 0)

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

= diffuse = true;

:fl + refr)) && (depth

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

AXDEPTH)

survive = SurvivalProbabi
estimation - doing it pr
if;
radiance = SampleLight(&
e.x + radiance.y + radian

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

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







7

Light models

tic: ↓ (depth = 0 = = inside /

nt = nt / nc, os2t = 1.0f), N); 3)

st a = nt - nc, b st Tr = 1 - (80 fr) R = (D * nnt

= diffuse = true;

efl + refr)) && (depth

D, N); ~efl * E * dif = true;

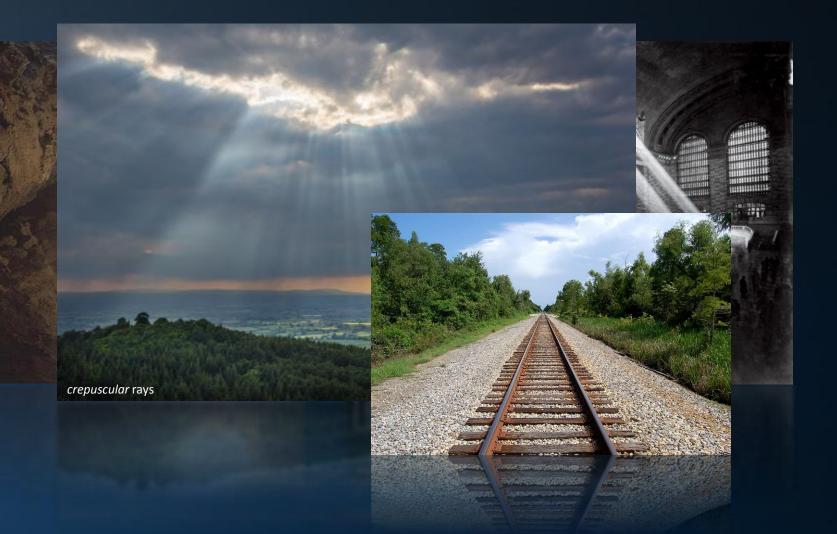
AXDEPTH)

survive = SurvivalProbabi
estimation - doing it pr
if;
radiance = SampleLight(&
e.x + radiance.y + radian

v = true; at brdfPdf = EvaluateDiff st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, SR, SC urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;





Light models

(depth / 1005 = inside / 1 it = nt / nc, /

os2t = 1.0f), N); })

st a = nt - nc, b st Tr = 1 - (80 + fr) R = (0 * nnt

= diffuse = true;

fl + refr)) && (depth o

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

AXDEPTH)

survive = SurvivalProbabl
estimation - doing it pr
if;
radiance = SampleLight(&
e.x + radiance.y + radian

v = true; at brdfPdf = EvaluateDiff st3 factor = diffuse = INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely follo vive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;





Light models

tic: € (depth < 10

: = inside / L ht = nt / nc, dr 552t = 1.0f - nn 5, N); 3)

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

= diffus = true;

-: :fl + refr)) && (depth & HADDET

D, N); ref1 * E * diff = true;

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pu st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely following /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, D) pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:







Light models



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

v = true; at brdfPdf = EvaluateDiffuse(L, N.) Probat3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

indom walk - done properly, closely following :
/ive)

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





Light models

i (depth i
 = inside
 it = nt /
is2t = 1.0
}, N);
))

at a = nt at Tr = 1 -Tr) R = (D

= diff = true;

efl + refr)

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

AXDEPTH)

survive = S estimation if; radiance = e.x + radia

v = true; tbrdfPdf st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf;

andom walk - done properly, closely follow: /ive)

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





Light models

tic: € (depth < 100

= inside / : it = nt / nc, do os2t = 1.0f = --0, N); 0)

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

= diffuse; = true;

efl + refr)) && (depth is HANDIII

), N); ~efl * E * diffus = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I) .x + radiance.y + radiance.z) 00 #

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pour st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Could

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

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







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

= diffuse = true;

-:fl + refr)) && (depth k HANDIIII)

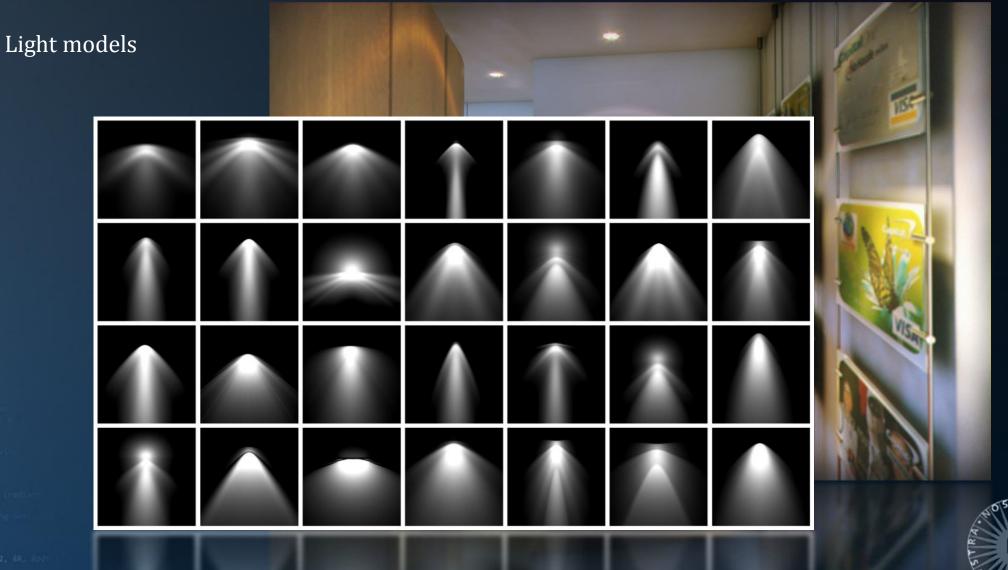
), N); refl * E * diffu = true;

AXDEPTH)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * read

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

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





Light models

tic: K (depth < 100

: = inside / 1 ht = nt / nc, dd os2t = 1.0f - nn 0, N }; 3)

at a = nt - nc, b - nt at Tr = 1 - (80 + (1 Tr) R = (D * nnt - N

= diffuse = true;

-: :fl + refr)) && (depth is HARDIIII)

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; adiance = SampleLight(%rand, I =.x + radiance.y + radiance.z) > 0) %

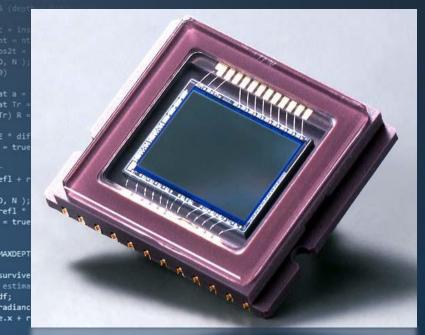
v = true; at brdfPdf = EvaluateDiffuse(L, N,) * Provident st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * ...

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

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, 4R, 4P urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:



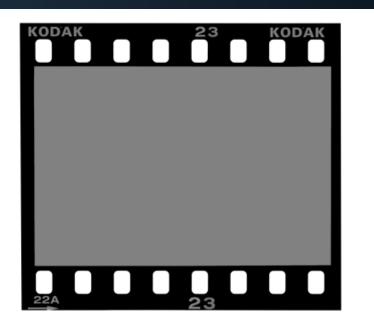
Sensor models



w = true; st brdfPdf = EvaluateDiffuse(L, N) Pr st3 factor = diffuse = INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely followin /ive)

, t3 Brdf = SampleDiffuse(diffuse, N, r1, r2, R, 10) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:









- tic: ≹ (depth < 10.⊂
- = * inside / L nt = nt / nc, dde os2t = 1.0f - nn 0, N); 0)
- at a = nt nc, b nt at Tr = 1 - (80 + (1 Tr) R = (0 * nnt - N
- = diffuse = true;
- efl + refr)) && (depth k HANDIII)
- D, N); refl * E * diffus = true;
- AXDEPTH)
- survive = SurvivalProbability(difference estimation - doing it properly if; radiance = SampleLight(%rand, I e.x + radiance.y + radiance.z) = 0) %
- v = true; st brdfPdf = EvaluateDiffuse(L, N) * Paircles st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Paircles
- andom walk done properly, closely following -/ive)
- ; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, pr pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Light is emitted by a light source
 Light interacts with the scene Absorption
 Light is absorbed by a sensor Scattering





tic: € (depth < 1000

: = inside / 1 it = nt / nc, dde os2t = 1.8f - nn: 0, N); 0)

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

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I, I, 2.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * read

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, sta urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Introduction
- Light Transport
- Materials
- Sensors
- Shading





Light Transport Quantities

at a = nt

fl + refr)) && (depth

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

AXDEPTH)

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

v = true: at brdfPdf = EvaluateDiffuse() st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPd

andom walk - done properly, closely for vive)

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

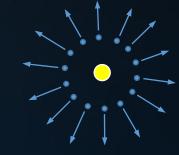
Radiant flux - Φ :

"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.*





tic: ⊾(depth < 10

: = inside / 1 it = nt / nc, dde ss2t = 1.0f = nnt -5, N); 3)

st a = nt - nc, b + nt st Tr = 1 - (80 + (1 - 1 Tr) R = (0 * nnt - 1

E ⁼ diffuse = true;

efl + refr)) && (depth is HANDE

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property ff; radiance = SampleLight(&rand, I e.x + radiance.y + radiance.r)

v = true; at brdfPdf = EvaluateDiffuse(at3 factor = diffuse * INVPI:

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

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

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, RR, D) 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 $1/r^2$.





tice k (depth ⊂ NAS

= = inside / 1 nt = nt / nc, dde os2t = 1.8f = nnt 0, N); 0)

at a = nt - nc, b - nt at Tr = 1 - (80 + 1) Tr) R = (D * nnt - N

= diffuse; = true;

-:fl + refr)) && (depth & MADIEI

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

AXDEPTH)

w = true; st brdfPdf = EvaluateDiffuse(L, N) Pour st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * (0)

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

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, R 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.



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.



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

at Tr = 1

= true:

), N);

= true;

MAXDEPTH)

v = true:

/ive)

efl * E * diffuse

survive = SurvivalProbability

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

st brdfPdf = EvaluateDiffuse(L, |
st3 factor = diffuse * INVPI;

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

andom walk - done properly, closely

22

tic: ⊾(depth ⊂ 10

= inside / 1 it = nt / nc, ddo os2t = 1.8f - nnt), N); 8)

st a = nt - nc, b - nt s st Tr = 1 - (R0 + (1 Tr) R = (D * nnt - N *

E ⁼ diffuse = true;

efl + refr)) && (depth k HANDIII

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property ff; radiance = SampleLight(&rand I &x + radiance.y + radiance.r) = 0

v = true;

at brdfPdf = EvaluateDiffuse(L, N) = Dearch st3 factor = diffuse = INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E = ((weight * cosThetaOut) / directPdf) = 0000

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, soft urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

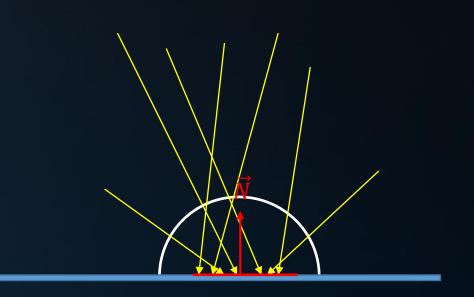
Light Transport Quantities

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.





tic: ≰ (depth < 100

: = inside / 1 it = nt / nc, dde os2t = 1.0f - nn: 0, N); 0)

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

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; adiance = SampleLight(%rand, I e.x + radiance.y + radiance.z) > 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Provident st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Paul

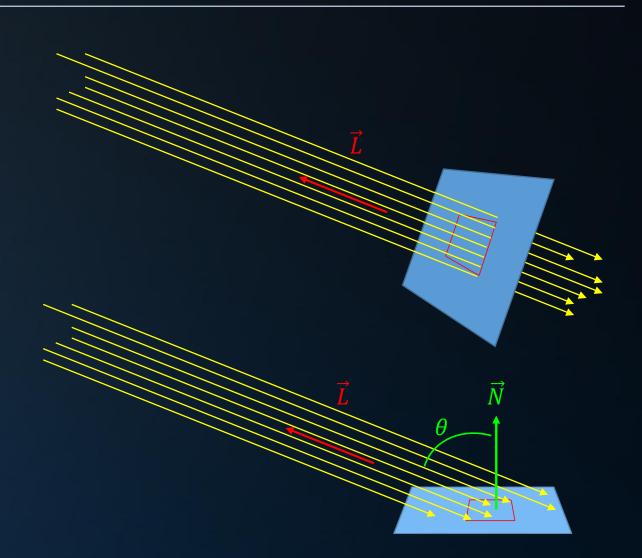
andom walk - done properly, closely following a /ive)

; st3 brdf = SampleDiffuse(diffuse, N, F1, F2, UR, S pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Light Transport Quantities

Converting radiance to irradiance:

 $E = L \cos \theta$





tice ≰ (depth ≤ 1000

: = inside / l ht = nt / nc, dde 552t = 1.0f - nnt 5, N); 3)

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

= diffuse; = true;

efl + refr)) && (depth is MADDE

D, N); ~efl * E * diffu = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; radiance = SampleLight(&rand I = 1) e.x + radiance.y + radiance.r) = 0

v = true; vt brdf0df

st brdfPdf = EvaluateDiffuse(L, N) Paurol st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

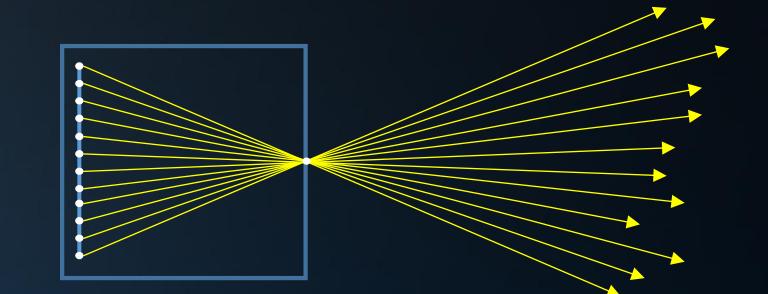
andom walk - done properly, closely following : /ive)

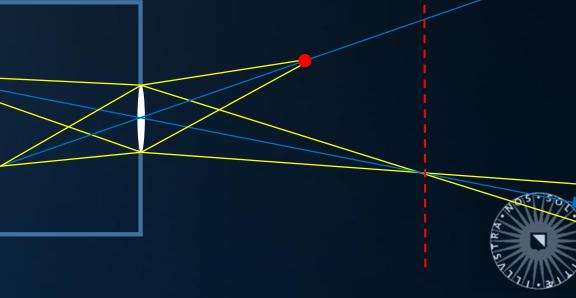
; t33 brdf = SampleDiffuse(diffuse, N, r1, r2, RR, R, r urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Pinhole Camera

A camera should not accept light from all directions for a particular pixel on the film. A pinhole ensures that only a single direction is sampled.

In the real world, an aperture with a lens is used to limit directions to a small range, but only on the focal plane.





tic: € (depth < 1000

: = inside / 1 it = nt / nc, dde os2t = 1.8f - nn: 0, N); 0)

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

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I, I, 2.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * read

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, sta urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Introduction
- Light Transport
- Materials
- Sensors
- Shading





INFOGR – Lecture 10 – "Shading Models"

Materials

at a = nt

), N); = true;

AXDEPTH)

v = true;

if;

efl + refr)) && (depth <

survive = SurvivalProbability(dff)

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

at brdfPdf = EvaluateDiffuse(L, N

st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf) at cosThetaOut = dot(N, L);

- Shader
- Normal map
- Specular map
- Color

light with a material.

Interaction:

- Absorption

andom walk - done properly, closely follo vive)

E * ((weight * cosThetaOut) / directPdf)

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

Material properties:

- Texture + detail texture

- ...

Used to simulate the interaction of

- Scattering







Materials

iic: ⊾ (depth < N

: = inside / L ht = nt / nc, ddo bs2t = 1.0f b, N); B)

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

E * diffuse; = true;

-:fl + refr)) && (depth C HANDER

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; radiance = SampleLight(%rand, I e.x + radiance.y + radiance.z) > 0) %

v = true; t brdfPdf = EvaluateDiffuse(L, N) = Pour st3 factor = diffuse = INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely following /ive)

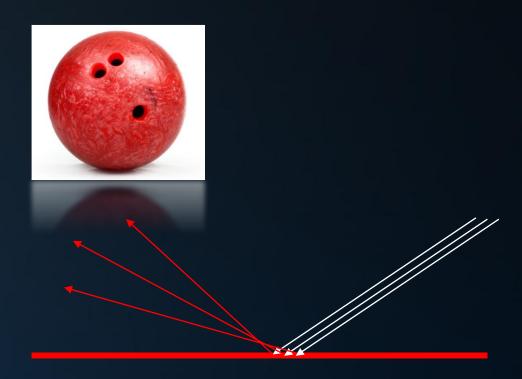
; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, N, st urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Absorption:

Happens on 'optical discontinuities'.

Light energy is converted in other forms of energy (typically heat), and disappears from our simulation.

Materials typically absorb light with a certain wavelength, altering the color of the scattered light. This is how we perceive material color.





INFOGR – Lecture 10 – "Shading Models"

Materials

tic: k (depth < 10.

: = inside / l it = nt / nc, dde os2t = 1.0f = nnt), N); 3)

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

E * diffuse; = true;

-:fl + refr)) && (depth & HAADIII

D, N); ref1 * E * diff(= true;

AXDEPTH)

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

v = true; t brdfPdf = EvaluateDiffuse(L, N) = Paars st3 factor = diffuse = INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

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

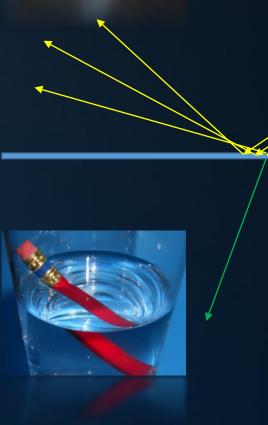
; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, Soft urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Scattering

Happens on 'optical discontinuities'.

Scattering causes light to change direction. Note that the amount of energy does not change due to scattering.

Light leaving the hemisphere can never exceed light entering the hemisphere, unless the material is emissive.





Materials

tica ⊾ (depth () IUS

: = inside / l it = nt / nc, ddo os2t = 1.0f - nnt ' D, N); D)

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

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; radiance = SampleLight(%rand, I & x + radiance.y + radiance.z) = 0)

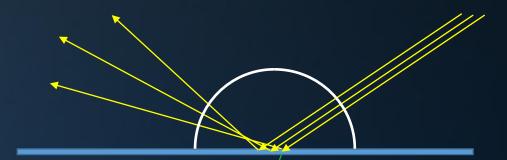
v = true; at brdfPdf = EvaluateDiffuse(L, N,) * P at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely (/ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, for pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Light / surface interaction

In: irradiance (E), from all directions over the hemisphere. Out: exitance (M), in all directions over the hemisphere.



The relation between *E* and *M* is linear: doubling irradiance doubles exitance.

 $\frac{M}{F}$ must be in the range 0..1.



tic: € (depth < 1000

: = inside / 1 it = nt / nc, dde os2t = 1.8f - nn: 0, N); 0)

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

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I, I, 2.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * read

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, sta urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Introduction
- Light Transport
- Materials
- Sensors
- Shading





tic: € (depth < NAS⊂

= inside / 1 it = nt / nc, dde os2t = 1.0f - nnt -D, N); B)

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

= diffuse; = true;

efl + refr)) && (depth is HANDERTH

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

AXDEPTH)

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

w = true; at brdfPdf = EvaluateDiffuse(L, N) = P bat3 factor = diffuse = INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely following /ive)

; ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, Dod prvive; pdf; n = E * brdf * (dot(N, R) / pdf); n = true;

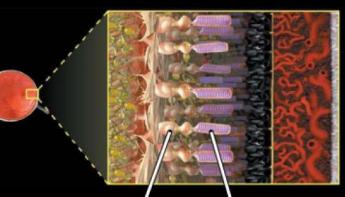
Sensors typically consists of many small sensors:

- Rods and cones in the eye
- Dye particles in the film
- Pixel elements in a CCD
- A ray in a ray tracer
- A fragment in a rasterizer

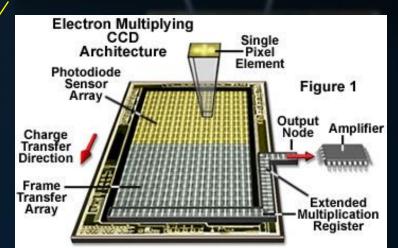
Note that we cannot use irradiance to generate an image:

irradiance is a measure for light arriving from all directions.

ENLARGED CROSS-SECTION OF RETINA



RÓD CÔNE





tic: € (depth < P

= inside / 1 nt = nt / nc, dd ps2t = 1.0f p, N); 3)

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

= diffus = true;

-: :fl + refr)) && (depth k HAADIII

D, N); ~efl * E * diffus = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; radiance = SampleLight(&rand I .x + radiance.y + radiance.r) = 0 = 0

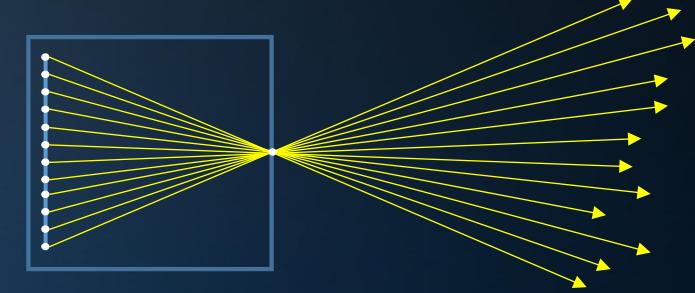
v = true; t brdfPdf = EvaluateDiffuse(L, N) Promote st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Ind

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

; at3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, 1997 pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Pinhole camera

To capture light from a specific direction, we use a camera with a small opening (the aperture), so that each sensor can 'see' a small set of incoming directions.





tic: ⊾(depth c.100

= inside / 1 it = nt / nc, dde -552t = 1.8f - nnt -5, N); 8)

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

= diffuse; = true;

-:fl + refr)) && (depth C HADDE

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; radiance = SampleLight(&rand, I e.x + radiance.y + radiance.r) = 0

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pump st3 factor = diffuse * INVPI; bt weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, bod urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

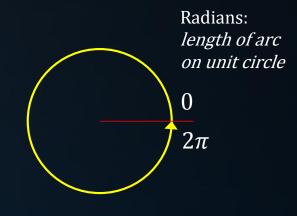
Radiance

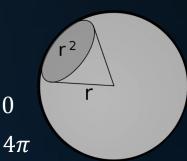
Using a pinhole camera, the sensors become directionally specific:

they average light over a small area, and a small set of incoming directions.

Recall that this is referred to as *radiance (L)*:

The density of light flow per area per incoming direction, in $W m^{-2} sr^{-1} s^{-1}$.





Steradians: area of surface on unit sphere



st a = nt

), N);

= true;

AXDEPTH)

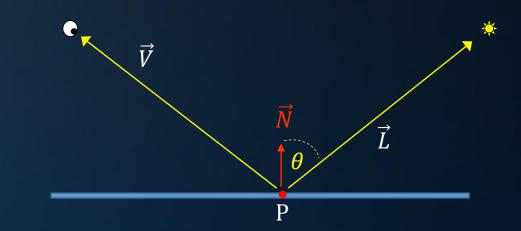
if;

efl + refr)) && (depth

efl * E * diffuse;

Summing it up:

- Light arrives from all light sources on point *P*;
- The energy flow per unit area, perpendicular to \vec{L} is projected on a surface perpendicular to \vec{N} . This is *irradiance*, or: *E*.
- Exitant light *M* is scattered over all directions on the hemisphere.
- Light scattered towards the eye arrives at a sensor.
- The sensor detects radiance: light from a specific set of directions.





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

survive = SurvivalProbability(diff

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

andom walk - done properly, closely following vive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Soci urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true: 35

tic: € (depth < 1000

: = inside / 1 it = nt / nc, dde os2t = 1.8f - nn: 0, N); 0)

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

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I, I, 2.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * read

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, sta urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Introduction
- Light Transport
- Materials
- Sensors
- Shading





tice E death a liter

: = inside / L ht = nt / nc, ddo os2t = 1.0f - nn: D; N); B)

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

= diffuse = true;

efl + refr)) && (depth & HANDIII

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(%rand, I, M) e.x + radiance.y + radiance.z) > 0) %

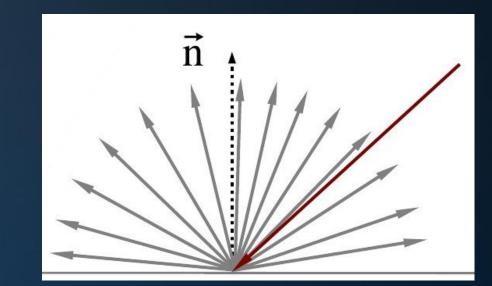
v = true; t brdfPdf = EvaluateDiffuse(L, N) = Pour of st3 factor = diffuse = INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) = 0000

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

; ot3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Npd) prvive; pdf; n = E * brdf * (dot(N, R) / pdf); nion = true;

Definition

Shading: the process of using an equation to compute the outgoing radiance along the view ray \vec{V} , based on material properties and light sources.



Diffuse or *Lambert* BRDF, also called "N dot L shading"



INFOGR – Lecture 10 – "Shading Models"

Shading

iles • (depth is NASS

: = inside / 1 it = nt / nc, dde os2t = 1.0f - ...), N); 3)

st a = $nt - nc_{1}$ b - ntst Tr = 1 - (R0 + (1 - 1)fr) R = (0^{-6} mnt - R^{-1}

= * diffuse; = true;

-:fl + refr)) && (depth is MANDITIN

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

AXDEPTH)

survive = SurvivalProbability estimation - doing it proper if; radiance = SampleLight(&rand, I = x + radiance.v + radiance.v)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Punc. st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) *

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, Doff urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Lambert shading model

The diffuse shading model is:

$$M_{diff} = \frac{c_{diff}}{\pi} L \,\overline{\cos}\theta_i$$

This takes into account:

- Projection of the direction of the incoming light on the normal;
- Absorption due to material color *c_{diff}*.

Distance attenuation is represented in *L*.

Practical implementation:

dist=light.pos-fragment.pos; L=normalize(light.pos-fragment.pos); N=fragment_normal; // interpolated radiance=light.color/(dist*dist); irradiance=radiance*dot(N,L); M=(material.color / PI)*irradiance;

The reflected energy M is what the camera will receive via the ray arriving at the fragment (i.e., the 'color' of the fragment).



INFOGR – Lecture 10 – "Shading Models"

Shading

tic: K (depth < NA

: = inside / 1 it = nt / nc, dde ss2t = 1.0f = nnt), N); 3)

at a = nt - nc, b - nt at Tr = 1 - (R0 + Tr) R = (0 * nnt - N

= diffuse = true;

efl + refr)) && (depth is MAAD]

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; radiance = SampleLight(%rand, I, %), e.x + radiance.y + radiance.z) = 0.000

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Puus st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) *

andom walk - done properly, closely following /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, brd pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

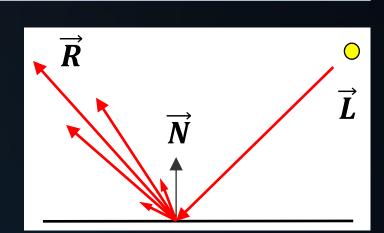
Phong shading model

The Phong shading model combines a diffuse reflection with a glossy one, and adds an ambient factor.

$$M_{phong} = c_{ambient} + c_{diff} (\vec{N} \cdot \vec{L}) L_{diff} + c_{spec} (\vec{V} \cdot \vec{R})^{S} L_{spec}$$

The Phong shading model is an 'empirical model', and has many problems:

- It doesn't guarantee that $M \leq E$;
- It doesn't take irradiance as input;
- It requires many (unnatural) parameters;
- That ambient factor...





at a = nt

>, N); ref1 * E = true;

AXDEPTH)

if;

efl + refr)) && (depth

BRDF – Bidirectional Reflectance Distribution Function

Defines the relation between *irradiance* and *radiance*.

Or, more accurately:

The BRDF represents the ratio of reflected radiance exiting along \vec{V} , to the irradiance incident on the surface from direction \vec{L} .

Note that the BRDF takes two parameters: an incoming and an outgoing direction.

$$f_r(\vec{L}, \vec{V}) = \frac{dL_{reflected}(\vec{V})}{dE_{incoming}(\vec{L})}$$



e.x + radiance.y + radiance.z) > 0) ## (control w = true; ot brdfPdf = EvaluateDiffuse(L, N) Public st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * (0)

survive = SurvivalProbability(dff)

adiance = SampleLight(&rand, I, L)

andom walk - done properly, closely following vive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Doffurvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true: 40

eler.

: = inside / 1
it = nt / nc, dde
ss2t = 1.0f = nnt
), N);
))

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

= * diffuse; = true;

efl + refr)) && (depth & NADIII

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, N) * 1 st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPd

andom walk - done properly, closely f /ive)

; t3 brdf = SampleDiffuse(diffuse, N urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true: **BRDF** – Bidirectional Reflectance Distribution Function

Diffuse BRDF:

$$f_r(\vec{L}, \vec{V}) = \frac{dL_{reflected}(\vec{V})}{dE_{incoming}(\vec{L})} = \frac{material\ color}{\pi}$$

where $E_{incoming}(\vec{L})$ is irradiance arriving from the light source, i.e. light color, times attenuation, times N dot L.

The diffuse BRDF scatters light equally in all directions. \vec{V} is not used in the equation. The diffuse BRDF is view independent.

Also note that \vec{N} and \vec{L} do not occur in this equation: N dot L is simply used to convert from radiance to irradiance.

Practical use of the BRDF:

Input for a BRDF is irradiance. This means that we already have processed attenuation and N dot L.

The fragment color, using a BRDF and a point light at distance r is thus:

$$M = f_r(\vec{L}, \vec{V}) \ \overline{\cos \theta_i} \ \frac{light \ color}{r^2}$$



BRDF – Bidirectional Reflectance Distribution Function

Phong BRDF:

at a = nt - nc, b - n† at Tr = 1 - (R0 + (1) fr) R = (D * nnt - R *

= diffuse; = true;

efl + refr)) && (depth & HADDIII

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; radiance = SampleLight(&rand, I & x + radiance.y + radiance.z) = 0

v = true; t brdfPdf = EvaluateDiffuse(L, N) = F ot3 factor = diffuse = INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, Sof urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

 $f_r(\vec{L}, \vec{V}) = \frac{dL_{reflected}(\vec{V})}{dE_{incoming}(\vec{L})} = color + color \,\overline{\cos \alpha}^m$

Where α is the angle between \vec{V} and \vec{R} , \vec{R} is \vec{L} reflected in \vec{N} , and m is the Phong exponent.

Note that the division by π is missing; it doesn't make sense for the specular reflection...

Also note that the ambient color is missing: this factor is constant and does not depend on irradiance.



tice € (depth () 100:

= inside / 1 it = nt / nc, dde -552t = 1.0f - nnt -5, N); 3)

= diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it properly if; radiance = SampleLight(&rand, I, L, L) e.x + radiance.y + radiance.z) > 0) #4

v = true; tbrdfPdf = EvaluateDiffuse(L, N,) = Pun st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

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

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, bod urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

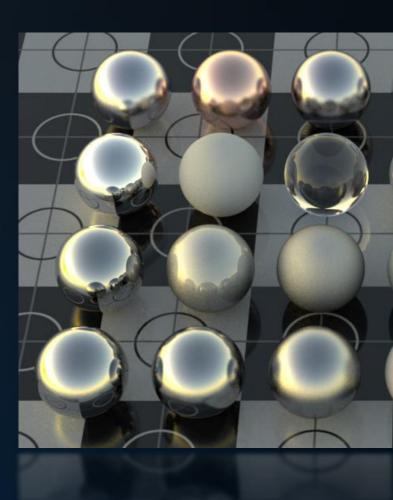
BRDF – Bidirectional Reflectance Distribution Function

BRDFs formalize the interaction of light / surface interaction, and allow us to do so in a physically correct way.

Games are switching to physically based models rapidly:

- To increase realism;
- To reduce the number of parameters in shaders;
- To have uniform shaders for varying lighting conditions.

More on this in Advanced Graphics!





tic: ⊾ (depth (⊂))

: = inside / l nt = nt / nc, ddf os2t = 1.0f 0, N); 3)

st a = nt - nc, b - nt st Tr = 1 - (80 + (1 fr) R = (D * nnt - N *

= diffuse = true;

efl + refr)) && (depth & HANDIIII

D, N); ~efl * E * diffuse; = true;

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property ff; radiance = SampleLight(&rand, I e.x + radiance.y + radiance.r) = 0.000

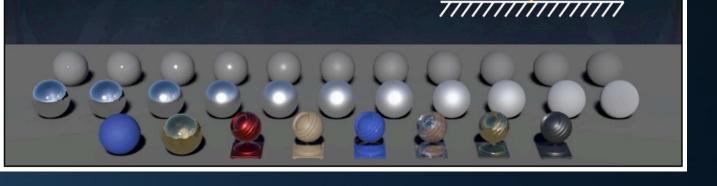
v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Pour st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, Soft urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Lighting – Analytical Lights

- Sun light
 - Units: Illuminance (lux)
 - Facing disk with non-null solid angle



"Moving Frostbite to PBR"

ttp://www.frostbite.com/wp-content/uploads/2014/11/s2014_pbs_frostbite_slides.pd



tic: k (depth < 100

: = inside / l nt = nt / nc, dd 552t = 1.0f - nn 5, N); 3)

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

= diffuse; = true;

efl + refr)) && (depth & MANDERT

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it properly if; radiance = SampleLight(@rand I = 1, e.x + radiance.y + radiance.r) = 0.000

v = true; at brdfPdf = EvaluateDiffuse(L, N.) = Pour st3 factor = diffuse = INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) = Pour E * ((weight * cosThetaOut) / directPdf)

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

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Soft urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Lighting Killzone : Shadow Fall

Michal Drobot Senior Tech Programmer

Guerrilla Games

"Lighting Killzone : Shadow Fall"

ttp://www.guerrilla-games.com/presentations/Drobot_Lighting_of_Killzone_Shadow_Fall.pdf



INFOGR – Lecture 10 – "Shading Models"

Shading

efl + refr)) && (depth k HA

), N); = true;

AXDEPTH)

survive = SurvivalProbability(diffe adiance = SampleLight(&rand, I, LL, e.x + radiance.y + radiance.z) > 0) |

v = true; at brdfPdf = EvaluateDiffuse(L, N) st3 factor = diffuse * INVPI; st 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, UR, Up urvive; pdf; 1 = E * brdf * (dot(N, R) / pdf); sion = true:



Lighting Conditions

Want same content to look good

"Physically Based Shading in Unity"



tic: € (depth < 1000

: = inside / 1 it = nt / nc, dde os2t = 1.8f - nn: 0, N); 0)

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

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; adiance = SampleLight(&rand, I, I, I, 2.x + radiance.y + radiance.z) = 0)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pauro st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * read

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, sta urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Today's Agenda:

- Introduction
- Light Transport
- Materials
- Sensors
- Shading





tic: ≰ (depth < NA

:= inside / l it = nt / nc, dd os2t = 1.0f - nn 0, N); 0)

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

= diffuse = true;

-: :fl + refr)) && (depth & MAXDIIII

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

AXDEPTH)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Puncture st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * File

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

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Sol pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

INFOGR – Computer Graphics

J. Bikker - April-July 2016 - Lecture 10: "Shading Models"

END of "Shading Models"

next lecture: "Visibility"

