tic: ⊾ (depth < )⇔

= inside / i it = nt / nc, dd os2t = 1.0f 0, N ); 3)

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

E <sup>=</sup> diffuse = true;

--•fl + refr)) && (depth k HAAD]

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

AXDEPTH)

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

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

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

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

# INFOGR – Computer Graphics

J. Bikker - April-July 2015 - Lecture 9: "Shading Models"

# Welcome!



tic: (depth < 1925

:= inside / L it = nt / nc, ddo os2t = 1.0f - nnt 0; N ); 3)

st a = nt - nc, b - nt - n st Tr = 1 - (R0 + (1 Fr) R = (0 \* nnt - N

= diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; adiance = SampleLight( &rand, I, 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) \* read

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

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

### Today's Agenda:

- Introduction
- Light Sources
- Materials
- Sensors
- Shading





#### INFOGR – Lecture 9 – "Shading Models"

## Introduction

= true:

fl + refr)) 88 (dept

D. N \; -efl \* E \* diffuse; = true;

AXDEPTH)

 $L(p \to r) = L_e(p \to r) + \sum_{i=1}^{N_L} L(q_i \to p) f_r(q_i \to p \to r) G(q_i \leftrightarrow p)$ survive = SurvivalProbability -adiance = SampleLight( &rand x + radiance.v + radiance.

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

andom walk - done properly, closely fell /ive)

at3 brdf = SampleDiffuse( diffuse, N, r1, r2, LR, i = E \* brdf \* (dot( N, R ) / pdf);

The Quest for (Photo-)Realism

Objective in modern games

Other factors:

Important improvements when using ray tracing (more in the next lecture, 'ground truth')

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

- Material interactions
- Light models
- Sensor models





#### Material interactions

ic: k (depth < 100⊂

- - 105100 / nt = nt / nc, os2t = 1.0f 0, N ); 3)

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

= diffuse; = true;

efl + refr)) && (dept

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

AXDEPTH)

survive = SurvivalProbabi estimation - doing it pr if; radiance = SampleLight( &rand. I e.x + radiance.y + radiance.z)

v = true; t brdfPdf = EvaluateDiffuse( L, N) Provide 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)

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







#### Material interactions

ic: ≰(depth < Nac⊂-

: = inside / nt = nt / nc. >s2t = 1.0f >, N ); >)

st a = nt - n st Tr = 1 - () Tr) R = (0 \*

= diffu = true;

efl + refr))

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

AXDEPTH)

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

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

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

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



INFOGR – Lecture 9 – "Shading Models"

### Introduction

#### Material interactions



E \* ((weight \* cosThetaOut) / directPdf) \* 0

vive)

; pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, NR, 1991 prvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true;





#### Light models

tic: ≰(depth < B

it = nt / nc, c
ps2t = 1.0f
0, N );
0)

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

= diffuse; = true;

efl + refr)) 88 (depth

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

#### AXDEPTH)

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

andom walk - done properly, closely following /ive)

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







#### Light models

tic: ⊾ (depth ()

nt = nt / nc, os2t = 1.0f O, N ); B)

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

= diffuse = true;

efl + refr)) && (depth

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

#### AXDEPTH)

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

andom walk - done properly, closely following /ive)

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





### Light models

fic: k (depth < 100 z = inside / 1

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

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

= diffuse = true;

efl + refr)) && (depth )

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

AXDEPTH)

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

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

andom walk - done properly, closely following /ive)

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





#### Light models

tic: ⊾ (depth < 19

= = inside / 1 nt = nt / nc, dd >s2t = 1.0f - nn >, N ); 3)

at a = nt - nc, b - nt at Tr = 1 - (80 + 1 Tr) R = (0 \* nnt - 8

= diffus = true;

efl + refr)) && (depth & NADIII

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

#### AXDEPTH)

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

v = true; tbrdfPdf = EvaluateDiffuse( L, N ) Pu 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)

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







### Light models



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

v = true; at brdfPdf = EvaluateDiffuse( L, N.) Prost3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf)

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

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





### Light models

100 k (depth s = inside st = nt / ss2t = 1.0 b, N ); b)

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

ndom walk - done properly, closely fellowir rive)

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





### Light models

tic: (depth < NA

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

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

= diffuse; = true;

efl + refr)) && (depth is HANDIII

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

#### WXDEPTH)

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)

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





fice ⊾ (depth (⊂100)

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

at a = nt - nc, b - nt at Tr = 1 - (80 + (1 Tr) R = (D \* nnt - 6

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; adiance = SampleLight( %rand, I, I) e.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)

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





### Light models

), N );

#### AXDEPTH)

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

v = true; t brdfPdf = EvaluateDiffuse( L, N ) \* P st3 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, UR, st urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:





#### Sensor models



v = true; t brdfPdf = EvaluateDiffuse( L, N) = Pr 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)

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









lc: ≹ (depth < NA⊂

= = 1051de / 5 nt = nt / nc., dda 952t = 1.0f - no 9, N ); 9)

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

= diffuse; = true;

efl + refr)) && (depth k HANDIII

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

#### AXDEPTH)

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

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)

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

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

- 1. Light is emitted by a light source
- 2. Light interacts with the scene
- 3. Light is absorbed by a sensor

- Absorption

Scattering







tic: € (depth < 1555

:= inside / L it = nt / nc, ddo os2t = 1.0f - nnt 0; N ); 3)

st a = nt - nc, b - nt - st Tr = 1 - (R0 + 1. Fr) R = (0 \* nnt - N

= diffuse = true;

• •fl + cefn)) 88 (death ( 1000)

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

AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; adiance = SampleLight( %rand, I, I, e.x + radiance.y + radiance.r) > 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) \* Provident E \* ((weight \* cosThetaOut) / directPdf) \* Provident

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, UR, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### Today's Agenda:

- Introduction
- Light Sources
- Materials
- Sensors
- Shading



## Light Sources

tic: € (depth < 12

= inside / 1 it = nt / nc, dda -552t = 1.0f - nnt -2, N ); 8)

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

E \* diffuse; = true;

efl + refr)) && (depth < NADI-

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

AXDEPTH)

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

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

Directional lights

Directional light, such as the light from the sun:

Specified by a normalized, reversed vector  $\vec{L}$ .

Power is specified as energy travelling through a unit surface area, perpendicular to  $\vec{L}$ .

This quantity is called *irradiance*; units:  $W m^{-2}s^{-1}$ . The symbol for irradiance is *E*.

For practical purposes, we will express the energy as RGB vectors. Note that R,G,B can exceed 1, unlike e.g. colors in a painting.



## Light Sources

tic: k (depth < 10)

= inside / l it = nt / nc, dde 552t = 1.8f - nd 3, N ); 3)

at a = nt - nc, b - n1 at Tr = 1 - (R0 + fr) R = (D \* nnt - N

= diffuse; = true;

efl + refr)) && (depth k HAAD

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

#### AXDEPTH)

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

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

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

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

### Directional lights

When illuminating a surface, we need to know how much light arrives at a unit area on the surface, i.e. how much light passes through a unit surface area perpendicular to  $\vec{N}$ .

For this, we multiply by the cosine of the angle between  $\vec{N}$  and  $\vec{L}$ , i.e.  $\vec{N} \cdot \vec{L}$ .

Note that the cosine is clamped to 0, to prevent negative contributions from light arriving from the backside of the surface.

 $E = E_L \overline{cos} \theta_i$  $E = E_L \max(\vec{N} \cdot \vec{L}, 0)$ 



## Light Sources

tic: K (depth < 10.5

= inside / 1 nt = nt / nc, dde os2t = 1.0f - nnt o, N ); 3)

st  $a = nt - nc_{2} b + nt +$ st Tr = 1 - (R0 + (1 - 1) Tr) R = (D \* nnt - N \* -

= diffuse = true;

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

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse( L, N ) \* Pours 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)

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

#### Irradiance

The unit surface may receive light from many directions. For multiple lights, irradiance is additive, and represents the energy arriving over the hemisphere:

 $E = \sum_{k=1}^{n} E_{L_k} \overline{\cos} \,\theta_{i_k}$ 





tic: € (depth < 1555

:= inside / L it = nt / nc, ddo os2t = 1.0f - nnt 0; N ); 3)

st a = nt - nc, b - nt - st Tr = 1 - (R0 + 1. Fr) R = (0 \* nnt - N

= diffuse = true;

• •fl + cefn)) 88 (death ( 1000)

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

AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; adiance = SampleLight( %rand, I, I, e.x + radiance.y + radiance.r) > 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) \* Provident E \* ((weight \* cosThetaOut) / directPdf) \* Provident

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, UR, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### Today's Agenda:

- Introduction
- Light Sources
- Materials
- Sensors
- Shading



### INFOGR – Lecture 9 – "Shading Models"

### Materials

#### ic: (depth < NASS

- : = inside / : nt = nt / nc, dom os2t = 1.0f = nnt 0, N ); 3)
- at a = nt nc, b = n1 at Tr = 1 - (R0 + (1 Tr) R = (D \* nnt - 8
- = diffuse; = true;
- -:fl + refr)) && (depth < HANDIII
- 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 at3 factor = diffuse \* INVPI;
- 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)

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

### Material properties:

- Texture + detail texture
- Shader
- Normal map
- Specular map
- Color
- ...

## Used to simulate the interaction of light with a material.

#### Interaction:

- Absorption
  - Scattering











### Materials

tic: ⊾(depth < 10

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

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

E = diffuse; = true;

efl + refr)) && (depth is HADDII

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

AXDEPTH)

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

w = true; st brdfPdf = EvaluateDiffuse( L, N) Promote 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)

; pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, NR, brd 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 9 – "Shading Models"

### Materials

tic: ⊾ (depth < 100

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

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

E = diffuse; = true;

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

D, N ); -efl \* E \* diffuse; = true;

AXDEPTH)

w = true; t brdfPdf = EvaluateDiffuse( L, N ) = Purch st3 factor = diffuse = INVPI; at weight = Mis2( directPdf, brdfPdf ); st cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf) = 000

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

; pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, UR, 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

efl + refr)) && (depth

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

AXDEPTH)

survive = SurvivalProbability( diff if; -adiance = SampleLight( &rand, I. 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 vive)

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

M

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 < 1555

:= inside / L it = nt / nc, ddo os2t = 1.0f - nnt 0; N ); 3)

st a = nt - nc, b - nt - st Tr = 1 - (R0 + 1. Fr) R = (0 \* nnt - N

= diffuse = true;

• •fl + cefn)) 88 (death ( 1000)

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

AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; adiance = SampleLight( %rand, I, I, e.x + radiance.y + radiance.r) > 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) \* Provident E \* ((weight \* cosThetaOut) / directPdf) \* Provident

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, UR, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### Today's Agenda:

- Introduction
- Light Sources
- Materials
- Sensors
- Shading



tic: (depth < (U.S.)

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

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

= diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property ff; radiance = SampleLight( %rand, I & x + radiance.y + radiance.z) = 0.0000

w = true; t brdfPdf = EvaluateDiffuse( L, N ) = F at3 factor = diffuse = INVPI; t 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, NR, Doff prvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true;

Sensors typically consists of many small sensors:

- Rods and cones in the eye
- Dye particles in the film
- Pixel elements in a <u>CCD</u>
- 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: K (depth < 19

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

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

= diffus: = true;

efl + refr)) && (depth & NADIII

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it properly 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) \* Ord

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

; st3 brdf = SampleDiffuse( diffuse, N, r1, r2, NR, Norm 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 < 100

= = inside / : it = nt / nc, d/n -552t = 1.8f - nn -3, N ); 3)

at  $a = nt - hc_{1}b + nt - hc_{2}b + nt - hc_{3}b + (1 - nt) + ($ 

= \* diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; radiance = Sampletight( %rand I e.x + radiance.y + radiance.r) > 0)

v = true; t brdfPdf = EvaluateDiffuse( L, N.) \* Process st3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf) \* 0

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, R, F3, 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.

This is referred to as *radiance* :

The density of light flow per area per incoming direction.

Units:  $W m^{-2} s r^{-1} s^{-1}$ . Symbol: L



r 2

0

 $4\pi$ 

Steradians: area of surface on unit sphere



30

st a = nt

), N );

= true;

AXDEPTH)

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.





#### estimation - doing it properly ff; radiance = SampleLight( &rand, I, II, II, e.x + radiance.y + radiance.z) > 0) v = true;

survive = SurvivalProbability diff

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

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

; pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, R, set pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true: tic: € (depth < 1555

:= inside / L it = nt / nc, ddo os2t = 1.0f - nnt 0; N ); 3)

st a = nt - nc, b - nt - st Tr = 1 - (R0 + 1. Fr) R = (0 \* nnt - N

= diffuse = true;

• •fl + cefn)) 88 (death ( 1000)

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

AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; adiance = SampleLight( %rand, I, I, e.x + radiance.y + radiance.r) > 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) \* Provident E \* ((weight \* cosThetaOut) / directPdf) \* Provident

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, UR, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### Today's Agenda:

- Introduction
- Light Sources
- Materials
- Sensors
- Shading



#### tic: ⊾ (depth (182)

: = inside / l it = nt / nc, dde os2t = 1.0f - ont - o ), N ); 8)

at  $a = nt - hc_{1}b + nt - hc_{2}b + nt - hc_{3}b + (1 - nt) + ($ 

= diffuse; = true;

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

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

#### AXDEPTH)

survive = SurvivalProbability difference estimation - doing it properly if; adiance = SampleLight( @rand I. =.x + radiance.y + radiance.r) > 0\_\_\_\_\_\_

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

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  $L_o$  along the view ray  $\vec{V}$ , based on material properties and light sources.



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



## Shading

fice (depth c HAR

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

st  $a = nt - nc_{1} b - nt$ st Tr = 1 - (R0 + (1 - 1))Tr) R = (0 + nnt - 1)

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Lambert shading model

The diffuse shading model is:

$$I_{diff} = \frac{c_{diff}}{\pi} E_{L_i} \overline{cos} \theta_i$$

#### This takes into account:

- Projection of the directional light on the normal;
- Absorption due to material color *c<sub>diff</sub>*.

# Distance attenuation is represented in $E_{L_i}$ (for directional lights, this is not applicable)



### INFOGR – Lecture 9 – "Shading Models"

## Shading

tic: ⊾(depth ∈ 100

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

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

= diffuse = true;

efl + refr)) && (depth k HAAD

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

AXDEPTH)

w = true; at brdfPdf = EvaluateDiffuse( L, N ) \* Pun 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)

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





efl + refr)) && (depth

survive = SurvivalProbability( diff

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

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

i = E \* brdf \* (dot( N, R ) / pdf);

E ((weight \* cosThetaOut) / directPdf)

efl \* E \* diffuse;

), N );

= true;

AXDEPTH)

v = true;

sion = true:

if:

**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.

 $f_r(\vec{L}, \vec{V}) = \frac{dL_r(\vec{V})}{dE_i(\vec{L})}$ 



36

tice ⊾ (depth < 1935

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

st a = nt - nc, b = nt - ncst Tr = 1 - (R0 + (1 - 0) Tr) R = (D \* nnt - N

= diffuse; = true;

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

D, N ); -efl \* E \* diffuse; = true;

AXDEPTH)

survive = SurvivalProbability( different estimation - doing it properly, if; radiance = SampleLight( &rand, I, I, I) e.x + radiance.y + radiance.z) > 0) #8

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) \* (P

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

; pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, NR, bof 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: k (depth < 12

: = inside / l it = nt / nc, dd os2t = 1.0f - nd 0, N ); 3)

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

= diffuse; = true;

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

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

AXDEPTH)

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

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)

; pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, UR, 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: ⊾ (depth < 10

= inside / 1 ht = nt / nc, dr 552t = 1.0f - nn 5, N ); 3)

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

= diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it properly if; radiance = SampleLight( &rand, I. e.x + radiance.y + radiance.r) > 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) \* Provident E \* ((weight \* cosThetaOut) / directPdf) \* Provident

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

; st3 brdf = SampleDiffuse( diffuse, N, r1, r2, R, lost 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 9 – "Shading Models"

## Shading

tic: € (depth⊂c 1923

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

at  $a = nt - nc_{0} b - nt + at Tr = 1 - (80 + 1) Tr = 1 - (70 + 1) Tr = (70 + nnt - 1)$ 

= diffuse; = true;

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

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

#### AXDEPTH)

v = true; t brdfPdf = EvaluateDiffuse( L, N ) = Paur 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)

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



Want same content to look good

#### "Physically Based Shading in Unity"

http://aras-p.info/texts/files/201403-GDC UnityPhysicallyBasedShading notes.pdf



tic: € (depth < 1555

:= inside / L it = nt / nc, ddo os2t = 1.0f - nnt 0; N ); 3)

st a = nt - nc, b - nt - st Tr = 1 - (R0 + 1. Fr) R = (0 \* nnt - N

= diffuse = true;

• •fl + cefn)) 88 (death ( 1000)

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

AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; adiance = SampleLight( %rand, I, I, e.x + radiance.y + radiance.r) > 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) \* Provident E \* ((weight \* cosThetaOut) / directPdf) \* Provident

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, UR, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### Today's Agenda:

- Introduction
- Light Sources
- Materials
- Sensors
- Shading



tic: ≰ (depth < 100

= inside / L it = nt / nc, dde os2t = 1.0f 0, N ); 3)

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

E = 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 ...x + radiance.y + radiance.r) > 0

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

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

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

# INFOGR – Computer Graphics

J. Bikker - April-July 2015 - Lecture 9: "Shading Models"

# END of "Shading Models"

next lecture: "Ground Truth"

