tic: ⊾ (depth < 144

= inside / L it = nt / nc, dde 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 HADDE

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 10: "Ground Truth"

# Welcome!



tic: € (depth < 1555

:= inside / i ht = nt / nc, dde os2t = 1.0f - nnt 0; N(); 3)

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

= diffuse; = true;

efl + refr)) && (depth k HAADIIII

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) %

v = true;

at brdfPdf = EvaluateDiffuse( L, N) Process at3 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)

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

## Today's Agenda:

- Limitations of Whitted-style Ray Tracing
- Monte Carlo
- Path Tracing



## Whitted Recap

tic: ⊾ (depth ∈ Has⊂

= inside / 1 it = nt / nc, dde ss2t = 1.0f - nnt 7 5, N ); 3)

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

andom walk - done properly, closely following /ive)

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

### Whitted-style Ray Tracing

In 1980, "State of the Art" consisted of:

- Rasterization
- Shading: either diffuse (N · L) or specular ((N · H)<sup>n</sup>), both not taking into account fall-off (Phong)
- Reflection, using environment maps (Blinn & Newell \*)
- Stencil shadows (Williams \*\*)

#### Goal:

Solve reflection and refraction

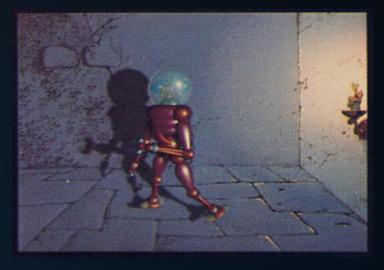
#### Improved model:

Based on classical ray optics

\* : Blinn, J. and Newell, M. 1976. Texture and Reflection in Computer Generated Images. Communications of the ACM 19:10 (1976), 542—547.

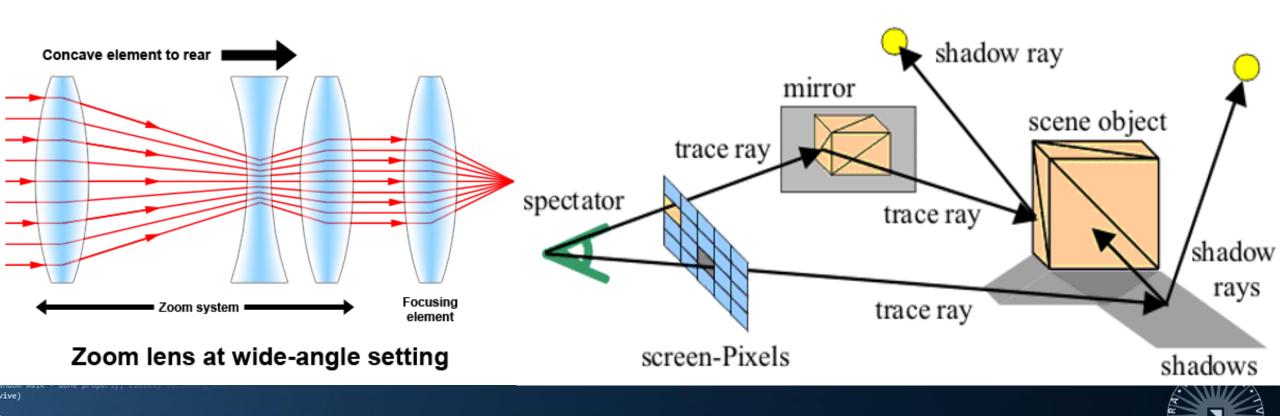
\*\* : Williams, L. 1978. Casting curved shadows on curved surfaces. In Computer Graphics (Proceedings of SIGGRAPH 78), vol. 12, 270–274.





## Whitted Recap

### Whitted-style Ray Tracing



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

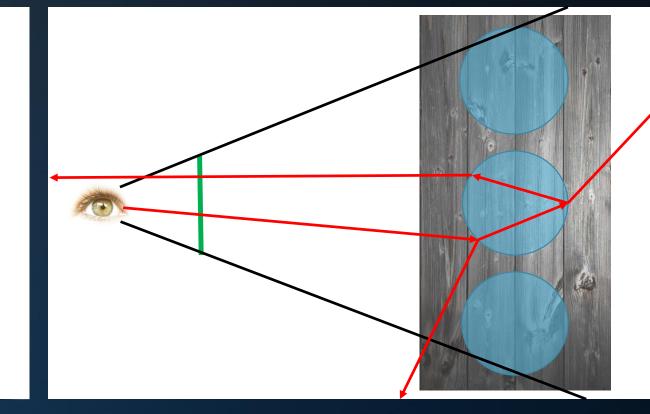
## Whitted Recap

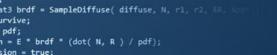
### Whitted-style Ray Tracing

### Color at pixel:

- sphere material color \* refracted ray
- + sphere material color \* reflected ray

#### This is a recursive process.



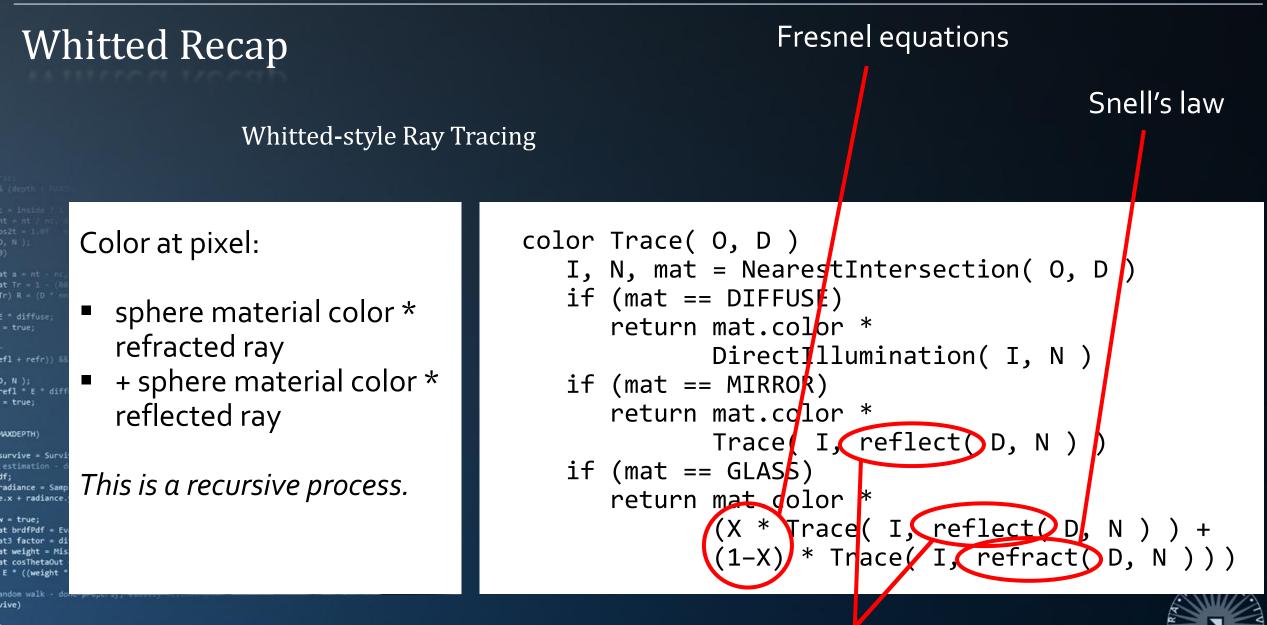


), N );

AXDEPTH)

adiance = Samp e.x + radiance. v = true; at brdfPdf = Ev at3 factor = di at weight = Mis at cosThetaOut E = ((weight \* andom walk - do vive)





, t33 brdf = SampleDiffuse( diffuse, N, r1, r2, 48, sp urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true: angle of incidence = angle of reflection

## Whitted Recap

tica ⊾ (depth < 1000

= inside / : it = nt / nc, dde ss2t = 1.0f - nnt -5, N ); 8)

st  $a = nt - nc_{3} b - nt - nc_{3}$ st Tr = 1 - (R0 + (1 - 1))Tr ) R = (D \* nnt - N - 1)

= diffuse; = true;

efl + refr)) && (depth < NADD

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

AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; radiance = Sampletight( &rand, I, e.x + radiance.y + radiance.z) 0) 88

v = true; t brdfPdf = EvaluateDiffuse( L, N ) Prost3 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, Upd) prvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

Whitted-style Ray Tracing

#### Improved model:

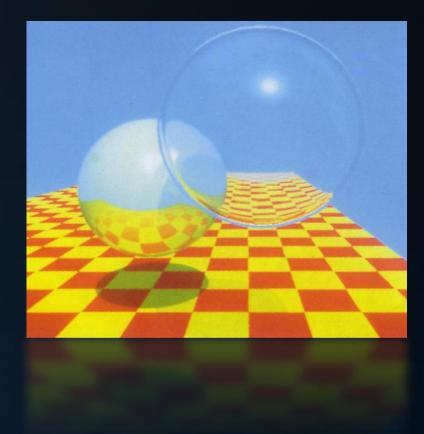
Based on classical ray optics

Dust off your physics books.

Physical basis of Whitted-style ray tracing:

Light paths are generated (backwards) from the camera to the light sources, using rays to simulate optics.

Whitted-style ray tracing is deterministic: it cannot simulate area lights, glossy reflections, and diffuse reflections.





tic: (depth < Not

= inside / 1 it = nt / nc, dde os2t = 1.01 2, N ); 3)

= diffuse; = true:

efl + refr)) && (depth k HANDIIII

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

#### AXDEPTH)

survive = SurvivalProbability( differe estimation - doing it properly if; radiance = SampleLight( &rand, I, &... e.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)

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







tic: € (depth < 1555

:= inside / i ht = nt / nc, dde os2t = 1.0f - nnt 0; N(); 3)

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

= diffuse; = true;

efl + refr)) && (depth k HAADIIII

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) %

v = true;

at brdfPdf = EvaluateDiffuse( L, N) Process at3 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)

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

## Today's Agenda:

- Limitations of Whitted-style Ray Tracing
- Monte Carlo
- Path Tracing



### Monte-Carlo

fice ⊾ (depth ∈ 1925

= inside / 1 tt = nt / nc, dde -552t = 1.0f = nnt -5, N ); 8)

st a = nt - nc, b - nt - st Tr = 1 - (R0 + (1 - 7) Tr) R = (D \* nnt - N \* - -

= diffuse; = true;

efl + refr)) && (depth k HANDIIII

D, N ); =efl \* 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; at brdfPdf = EvaluateDiffuse( L, N ) = Pour st3 factor = diffuse = INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf) = 000

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:

### **Distributed Ray Tracing**\*

#### Problem:

Ray tracing is currently limited to sharp shadows, sharp reflections, and sharp refraction.

#### Goal:

 Augment Whitted-style ray tracing with glossy reflections and refractions, as well as soft shadows.





11

\*: "Distributed Ray Tracing", Cook et al., 1984.





tic: € (depth ( 155

= inside / 1 nt = nt / nc. dda 552t = 1.0f - no 5, N ); 3)

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

= diffuse; = true;

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

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

#### AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; adiance = SampleLight( %rand, I, M) = 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, F1, F2, UR, body pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true: 

tic: k (depth ∈ 155

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

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

= diffuse; = true;

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

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

#### AXDEPTH)

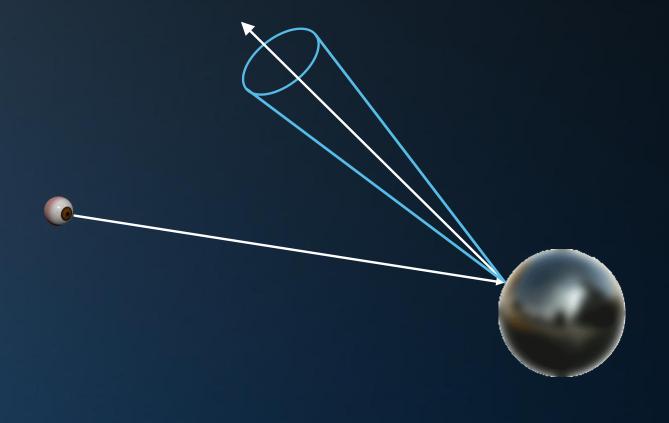
survive = SurvivalProbability difference estimation - doing it properly if; radiance = SampleLight( @rand I deference e.x + radiance.y + radiance.r) = 0.000

v = true; at brdfPdf = EvaluateDiffuse( L, N ) ↑ Po

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

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:





## Monte Carlo

tic) k (depth < 100⊂

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

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

= diffuse; = true;

efl + refr)) && (depth & NADIIII

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

AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it properly if; adiance = SampleLight( %rand, I, ) e.x + radiance.y + radiance.z) > 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) = 000

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

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

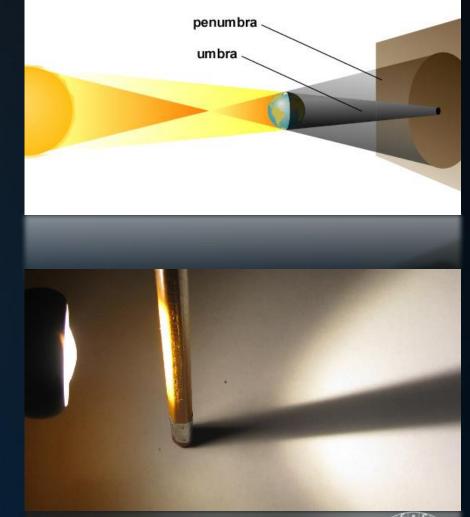
### Analytic Soft Shadows

### Anatomy of a shadow – regions

- Fully occluded area: *umbra*
- Partially occluded area: *penumbra*

#### A soft shadow requires an area light source.

In nature, all light sources are area lights (although some approximate point lights).





### Monte Carlo

tice k (depth < 155

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

st  $a = nt - hc_1 b - nt + s$ st Tr = 1 - (R0 + (1 - 1))(r) R = (0 - nnt - R - 1)

= diffuse; = true;

: :fl + refr)) && (depth k HANDIII

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

AXDEPTH)

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

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) \* 100

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

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

Analytic Soft Shadows

Surface points in the penumbra are lit by a part of the light source.

Rendering soft shadows requires that we determine the visible portion of the light source.

In most cases, this is a very hard problem.



## Monte Carlo

Approximate Soft Shadows

When using shadow mapping, we can simulate soft shadows by blurring the shadow map.

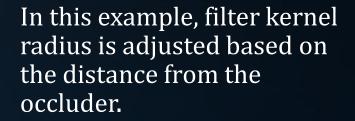
w = true; at brdfPdf = at3 factor = at weight = at cosTheta( E \* ((weight <u>- cos:netauut)</u>

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

MXDEPTH) survive = Su estimation if; adiance = S e.x + radiar

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

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







## Monte Carlo

tica ⊾ (depth ⊂ NAS⊂

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

= diffuse = true;

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

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

#### AXDEPTH)

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

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

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

### **Calculating Accurate Soft Shadows**

*"Rendering soft shadows requires that we determine the visible portion of the light source."* 

#### In other words:

The amount of light cast on a surface point P by area light L is determined by the integral of the visibility between P and L over the surface of the light source:

 $I_{L \to P} = \int_{A_I} V(P, L)$ 

### **Monte-Carlo Integration**

To solve this integral for the generic case, we will use Monte-Carlo integration.

Using Monte-Carlo, we replace the integral by the expected value of a stochastic experiment.



### Monte Carlo

fic: ⊾ (depth ⊂ 1935

= = inside / : it = nt / nc, dde ss2t = 1.8f = nnt 3, N (); 3)

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

= diffuse; = true;

efl + refr)) && (depth k HANDIIII

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

AXDEPTH)

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

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

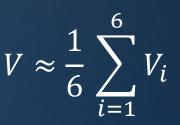
; t33 brdf = SampleDiffuse( diffuse, N, r1, r2, HR, hor urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

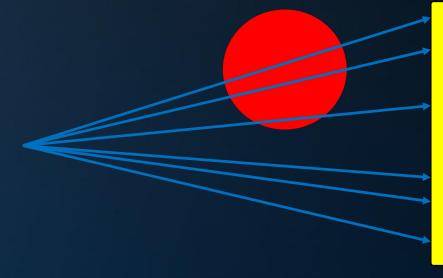
### Stochastic shadows

For soft shadows, we want to know the visible area of a light source, which can be 0..100%. The light source could be (partially) obscured by any number of objects.

We can approximate the visibility of the light source using a number of <u>random</u> rays.

Using 6 rays:







### Monte Carlo

fic: ⊾ (depth ⊂ 1935

= = inside / : it = nt / nc, ddo ss2t = 1.8f = nnt 3, N ); 3)

st a = nt - nc, b - nt - st Tr = 1 - (80 + (1 fr) R = (D \* nnt - 8

= = diffuse; = true;

efl + refr)) && (depth k HARDIII)

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

AXDEPTH)

survive = SurvivalProbability different estimation - doing it propert ff; radiance = SampleLight( &rand I e.x + radiance.y + radiance.r) > 0)

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

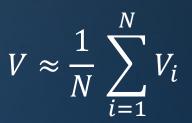
; t33 brdf = SampleDiffuse( diffuse, N, r1, r2, 48, 5pr) urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

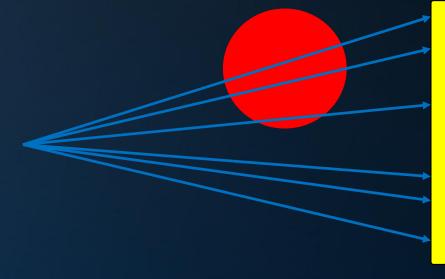
#### Stochastic shadows

For soft shadows, we want to know the visible area of a light source, which can be 0..100%. The light source could be (partially) obscured by any number of objects.

We can approximate the visibility of the light source using a number of <u>random</u> rays.

Using N rays:







"Le: k (depth < 100

= inside / L it = nt / nc, dde ss2t = 1.8f - nnt 5, N ); 3)

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

= diffuse; = true;

efl + refr)) && (depth k HANDIIII

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

AXDEPTH)

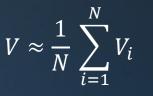
survive = SurvivalProbability difference estimation - doing it property if; radiance = SampleLight( %rand, I e.x + radiance.y + radiance.z) > 0) %

v = true; st 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)

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

#### Stochastic shadows



As *N* approaches infinity, the result becomes equal to the expected value, which is the integral we were looking for.

Before that, the result will exhibit *variance*. In the case of soft shadows, this shows up as noise.



### Monte Carlo

"ic: k (depth < 10.55

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

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

= \* diffuse; = true;

efl + refr)) && (dept

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

AXDEPTH)

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

v = true;

it brdfPdf = EvaluateDiffuse( L, N.) Paurol it3 factor = diffuse " INVPI; ot weight = Mis2( directPdf, brdfPdf ); it cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf) \* ;

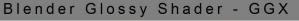
andom walk - done properly, closely following : rive)

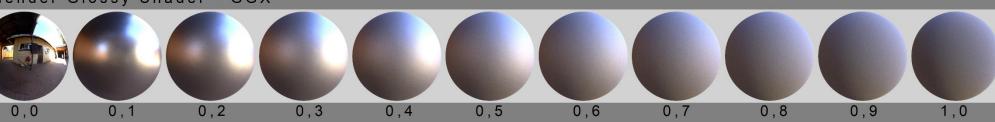
; at3 brdf = SampleDiffuse( diffuse, N, r1, r2, UR, bra urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

#### Approximate Diffuse Reflections

When rendering diffuse reflections, we face a similar problem:

A glossy surface reflects light arriving from a range of directions.





### In rasterization, we can achieve this by blurring the environment map.



Note that a correct glossy reflection requires a filter kernel size based on distance to the reflected object.

E = diffuse; = true;

st a = nt - n

efl + refr)) && (depth is Hood III

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

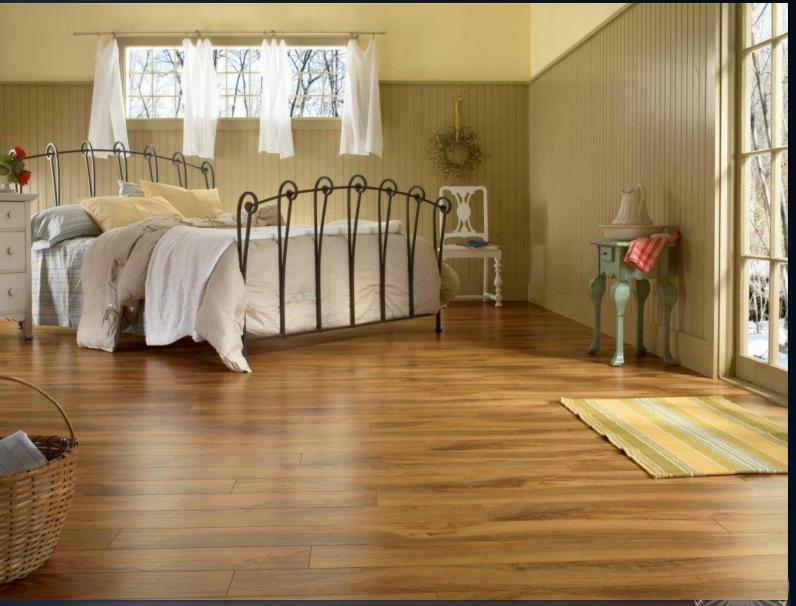
#### AXDEPTH)

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

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) \* Orden

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

, H33 brdf = SampleDiffuse( diffuse, N, F1, F2, RR, Soff urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:





#### Stochastic reflections

tic: ⊾ (depth < N2

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

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

= diffuse; = true;

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

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 bound st3 factor = diffuse = INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf) = Poul.

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:

25

## Monte Carlo

tic: K (depth < 10.5

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

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

= diffuse; = true;

efl + refr)) && (depth k HANDIIII

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

#### AXDEPTH)

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

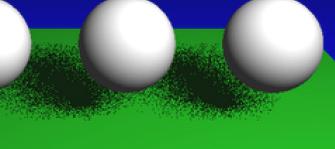
v = true; at brdfPdf = EvaluateDiffuse( L, N.) \* Pi st3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf

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

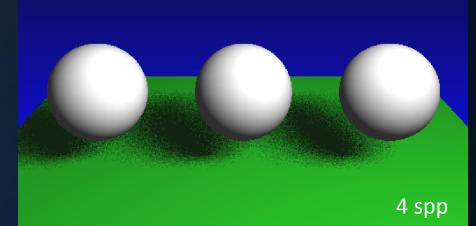
; pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, UR, Doff urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

#### Variance

As long as we don't take an infinite amount of samples, the result of the stochastic process exhibits variance.



1 spp





## Monte Carlo

fic: ⊾ (depth ⊂ 1925

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

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

= diffuse; = true;

= efl + refr)) && (depth k HANDIIII

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

#### AXDEPTH)

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

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

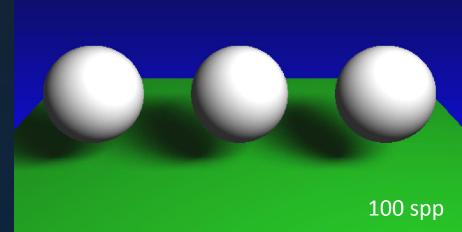
andom walk - done properly, closely following o /ive)

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

#### Variance

As long as we don't take an infinite amount of samples, the result of the stochastic process exhibits variance.

1 spp





tic: ⊾ (depth ():00

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

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

= diffuse = true;

efl + refr)) && (depth is HANDIS

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

#### AXDEPTH)

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

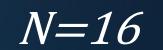
v = true; t brdfPdf = EvaluateDiffuse( L, N ) = Pour bast st3 factor = diffuse = INVPI; st weight = Mis2( directPdf, brdfPdf ); st cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf) = 0000

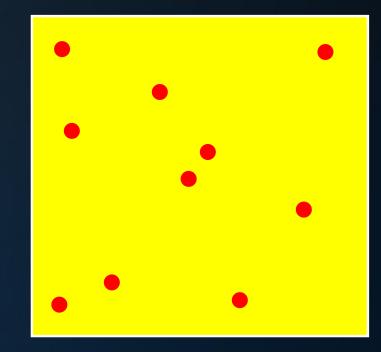
andom walk - done properly, closely following a /ive)

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

#### Variance reduction: stratification

#### The variance in random sampling can be reduced using *stratification*.







tice ⊾ (depth () Har

= inside / 1 it = nt / nc, dde >s2t = 1.0f - nnt >, N ); >)

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

= diffuse = true;

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

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

#### AXDEPTH)

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

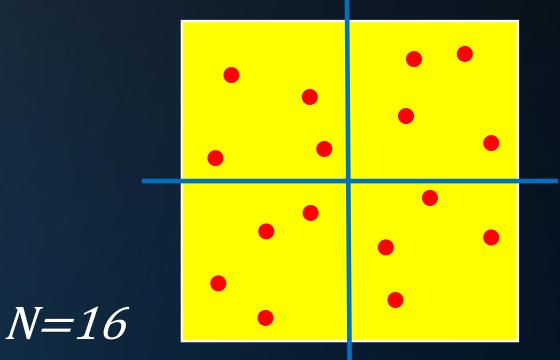
v = true; at brdfPdf = EvaluateDiffuse( L, N ) \* Pour st3 factor = diffuse \* INVPI; st weight = Mis2( directPdf, brdfPdf ); st cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf) \* 000

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

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

### Variance reduction: stratification

#### The variance in random sampling can be reduced using *stratification*.





## Monte Carlo

), N );

#### AXDEPTH)

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

v = true; st 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 felle vive)

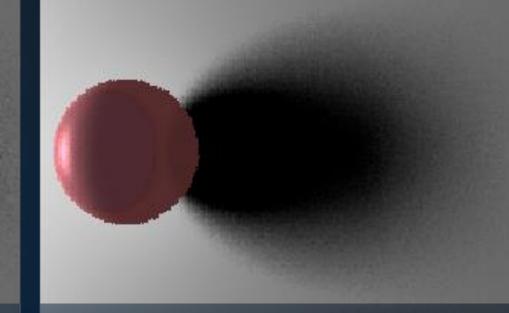
at3 brdf = SampleDiffuse( diffuse, N, r1, r2 rvive; pdf; 1 = E \* brdf \* (dot( N, R ) / pdf); sion = true:

#### Variance reduction: stratification

The variance in random sampling can be reduced using *stratification*.

### Uniform vs stratified, 36 samples, 6x6 strata





### Monte-Carlo

tica ⊾ (depth < NAS

= = inside / 1 ht = nt / nc, d/m -552t = 1.0f - ----3, N ); 3)

E \* diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability( d.f. estimation - doing it properly if; adiance = SampleLight( %rand, I. 1. 2.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, S pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### **Distributed Ray Tracing**

Integrating over area of light sources: soft shadows Integrating over reflection cone: glossy reflections Integrating over pixel: anti-aliasing

Integrating over time: motion blur Integrating over lens: depth of field Integrating over wavelength: dispersion





31

### Monte Carlo

tice € (depth < 1000

= inside / :
it = nt / nc, dde
>s2t = 1.0f = ont
>, N );
>)

= diffuse; = true;

efl + refr)) && (depth & MADITIE

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; 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 : /ive)

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

### **Distributed Ray Tracing**

#### Improved model:

- Still based on classical ray optics
- Combined with probability theory to solve integrals

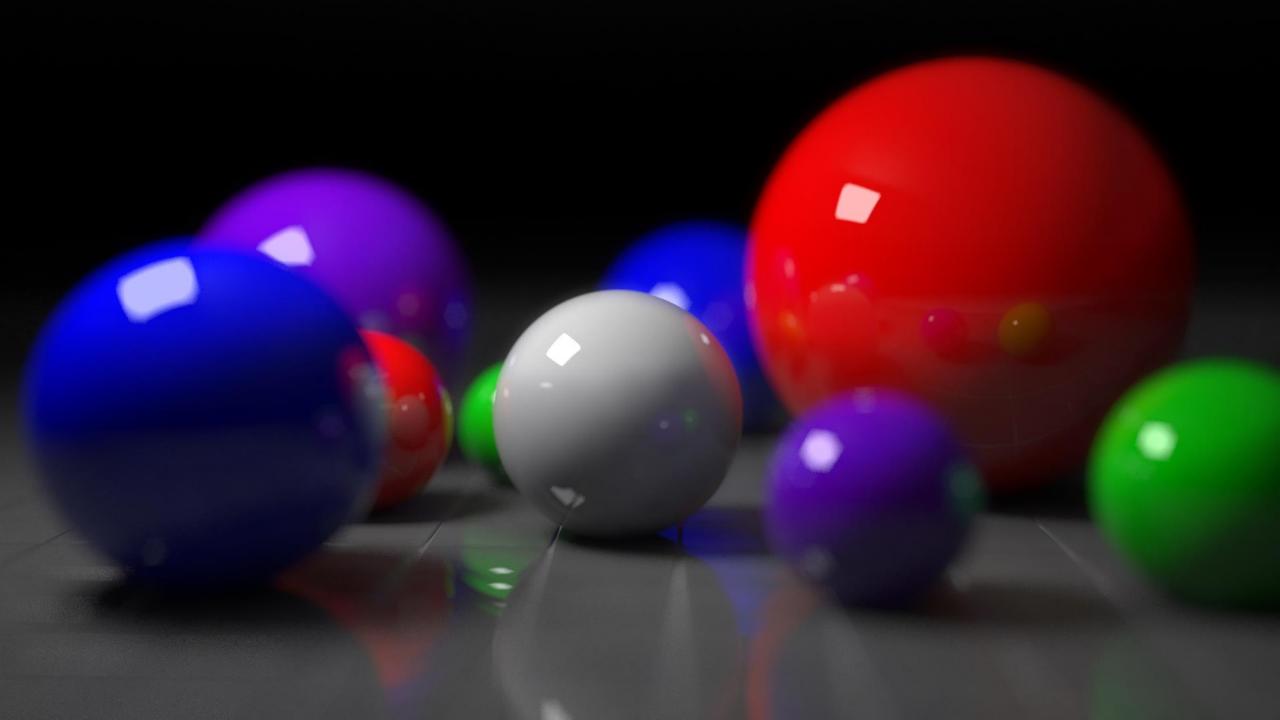
Physical basis of distributed ray tracing:

Light paths are generated (backwards) from the camera to the light sources, using rays to simulate optics.

Distributed ray tracing requires many rays to bring down variance to acceptable levels.







tic: k (depth < 100

: = inside / : nt = nt / nc, de ss2t = 1.0f - n 2, N ); 3)

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

= diffuse = true:

: :fl + refr)) && (depth ( MAXO)

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

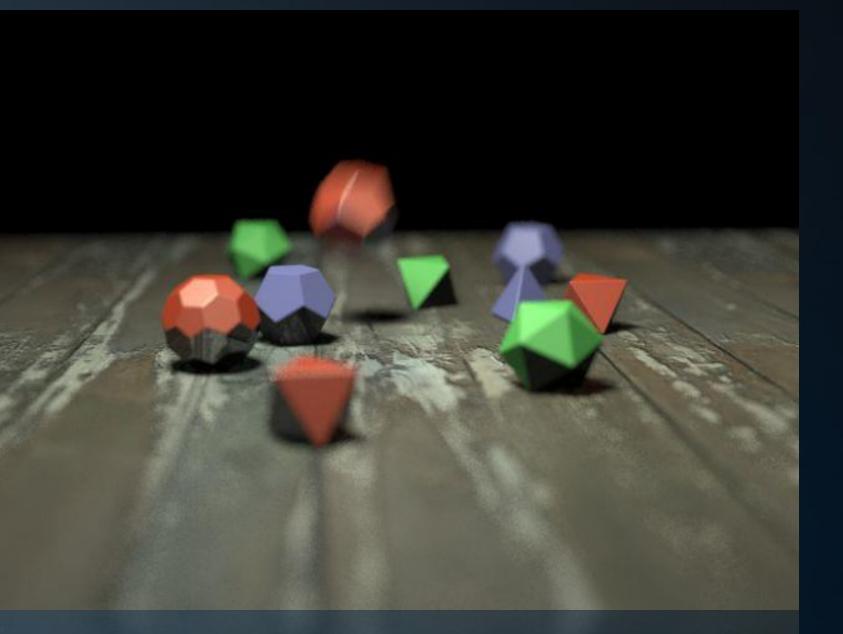
#### AXDEPTH)

survive = SurvivalProbability( difference estimation - doing it property ff; radiance = SampleLight( &rand, 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)

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

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







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

## Monte Carlo

te: K (depth < 10.55

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

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

= diffuse = true;

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

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

#### AXDEPTH)

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

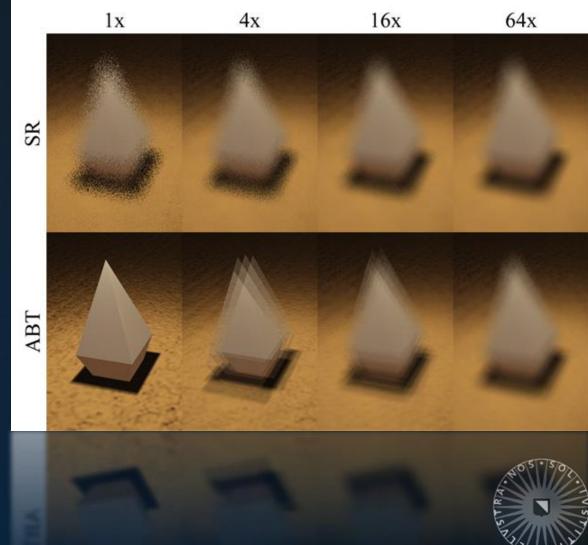
andom walk - done properly, closely following a /ive)

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

#### Monte Carlo in Rasterization

"Stochastic Depth of Field using Hardware Accelerated Rasterization",

#### Robert Toth & Erik Lindler, 2008



### Monte Carlo

tica **i (depth** i⊂ 1000

= = 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 \* - -

= diffuse = true;

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

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

#### AXDEPTH)

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

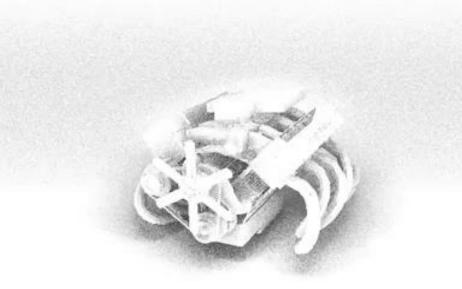
v = true; t brdfPdf = EvaluateDiffuse( L, N, ) Process st3 factor = diffuse \* INVPI; st weight = Mis2( directPdf, brdfPdf ); st cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPdf)

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

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

#### Monte Carlo in Rasterization

Screen Space Ambient Occlusion, CryEngine 2, 2007.







# Monte Carlo

tice ≰ (depth < 10.5

= = inside / 1 it = nt / nc, dde -552t = 1.0f = nnt -5, N ); 8)

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

= diffuse; = true;

efl + refr)) && (depth & HANDISIN

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; radiance = SampleLight( &rand, I , L e.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) = 000

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

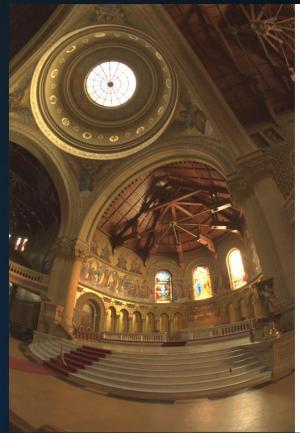
; pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, NR, D) pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

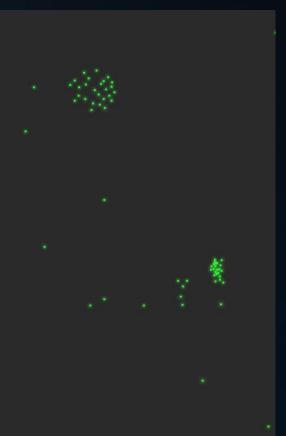
#### Monte Carlo in Rasterization

Light from an environment map, from:

"Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions",

Clarberg et al., 2005.







### Monte Carlo

tice ⊾ (depth is 1925

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

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

E \* diffuse; = true;

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

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

AXDEPTH)

w = true; st brdfPdf = 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 vive)

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

Cost of Distributed Ray Tracing

Distributed Ray Tracing is an expensive process:

- Per primary hit point, we need ~64 shadow rays *per light*
- Per primary hit point on a glossy surface, we need ~64 reflection rays,
  - ...and, for each reflection ray hit point, we need ~64 shadow rays per light.

If we use 4x4 anti-aliasing per pixel, multiply the above by 16. Now imagine a glossy surface reflects another glossy surface...



### Monte Carlo

tic: € (depth < P

= inside / L nt = nt / nc, de 552t = 1.0f - nc 5, N ); 3)

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

= diffuse = true;

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

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

#### WXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse( L, N.) \* Pours at3 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, es pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

# We need to go deeper



tic: € (depth < 1555

:= inside / i ht = nt / nc, dde os2t = 1.0f - nnt 0; N(); 3)

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

= diffuse; = true;

efl + refr)) && (depth k HAADIIII

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) %

v = true;

at brdfPdf = EvaluateDiffuse( L, N) Process at3 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)

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

### Today's Agenda:

- Limitations of Whitted-style Ray Tracing
- Monte Carlo
- Path Tracing



### Ray Tree

*Using distributed ray tracing:* 

= diffuse; = true;

st a = nt

efl + refr)) && (depth & HADD

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

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

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

• The energy via *N* shadow rays is averaged

- The energy via *N* reflection rays is averaged
- For each of them, the energy of N shadow rays is averaged.

→ The energy via each shadow ray is very low.



#### Diffuse reflections

Apart from specular and glossy materials, diffuse materials also reflect light.

= diffuse; = true;

efl + refr)) && (depth & MADICI

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

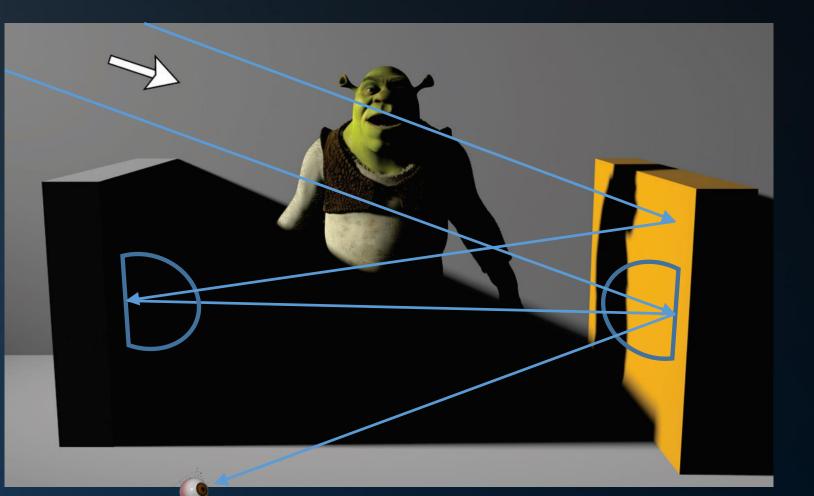
AXDEPTH)

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

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





#### Diffuse reflections

Apart from specular and glossy materials, diffuse materials also reflect light.

This is why a shadow is seldom black.

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

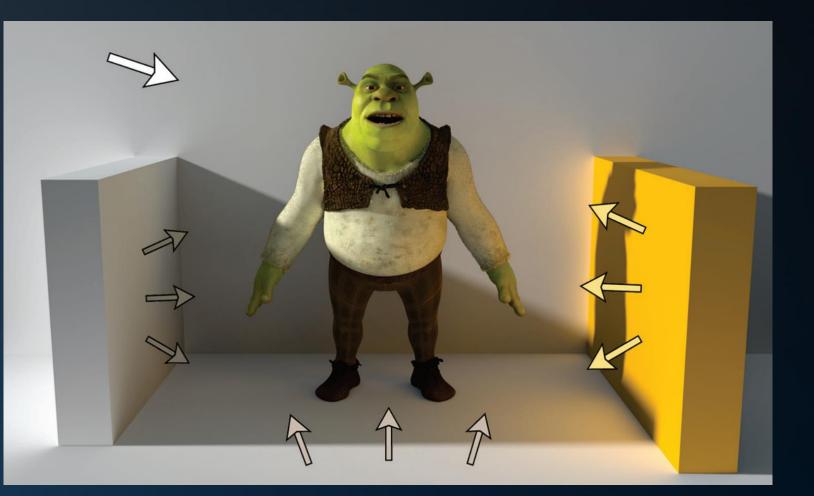
WXDEPTH)

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

v = true; t brdfPdf = EvaluateDiffuse( L, N ) \* Pour bast 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 a /ive)

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





### **Physically Based**

tic: ⊾(depth < 1000

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

st  $a = nt - hc_{2} b + nt - hc_{3}$ st Tr = 1 - (R0 + (1 - 77))(r) R = (0 - nnt - 77)

= diffuse = true;

efl + refn)) && (depth is MARDITIN

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

AXDEPTH)

v = 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 followin /ive)

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

#### Physically based rendering

Calculating <u>all</u> light transport from the light sources to the camera, directly or via scene surfaces.

Nature solves this using a "random walk": a large number of photons travelling through space from lights to sensors.



 $L_o(\mathbf{x}, \mathbf{w}) = L_e(\mathbf{x}, \mathbf{w}) + \int_{\Omega} f_r(\mathbf{x}, \mathbf{w}', \mathbf{w}) L_i(\mathbf{x}, \mathbf{w}') (-\mathbf{w}' \cdot \mathbf{n}) d\mathbf{w}'$ 



iic: K (depth < 10

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

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

= diffuse
= true;

: :fl + refr)) && (depth k MAXDIII

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

#### AXDEPTH)

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

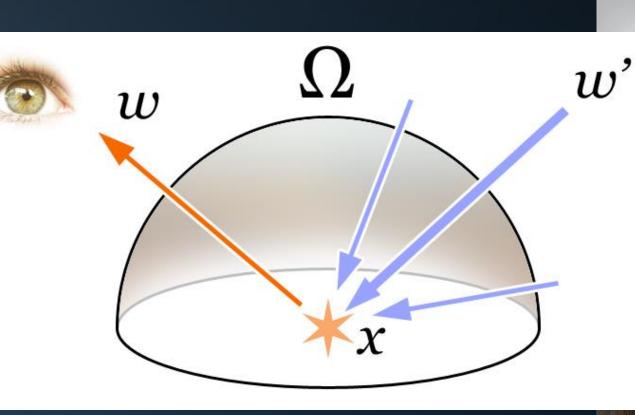
v = true; at brdfPdf = EvaluateDiffuse( L, M ) ^ P

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

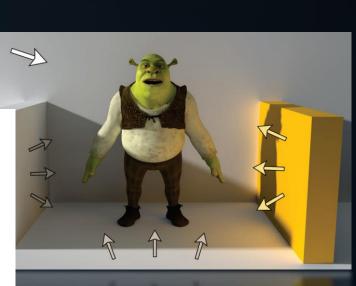
E \* ((weight \* cosThetaOut) / directPdf)

sndom walk - done properly, closely followin /ive)

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



$$L_o(\mathbf{x}, \mathbf{w}) = L_e(\mathbf{x}, \mathbf{w}) + \int_{\Omega} f_r(\mathbf{x}, \mathbf{w}', \mathbf{w}) L_i(\mathbf{x}, \mathbf{w}') (-\mathbf{w}' \cdot \mathbf{n}) d\mathbf{w}$$





tica ≰ (depth ⊂ Norm

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

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

= diffuse; = true;

efi se (depth -

refl \* E \* diffuse; = true;

AXDEPTH)

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

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

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

, t33 brdf = SampleDiffuse( diffuse, N, F1, F2, SR, Spec urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:  $P_T + P_R = 1$ 

 $\mathsf{P}_{\mathsf{T}}$ 

 $\mathsf{P}_\mathsf{R}$ 

N • L



tic: k (depth < 10.

= inside / : it = nt / nc, dde os2t = 1:0f - ...; 3)

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

= diffuse; = true;

= efl + refr)) && (depth k HADIIII

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

#### AXDEPTH)

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

v = true;

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

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

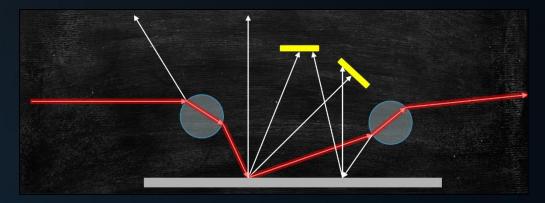
; pt3 brdf = SampleDiffuse( diffuse, N, r1, r2, NR, brd pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

#### Path Tracing

}

Color Trace( vec3 0, vec3 D )
{

```
I,N,mat = Intersect( O, D );
if (mat.IsLight()) return mat.emissive;
vec3 R = RandomReflection( N );
BRDF = mat.color;
return BRDF * dot( N, R ) * Trace( I, R );
```





#### Ray Tree

tic: ⊾(depth ( ))

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

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

E = diffuse; = true;

efl + refr)) && (depth & MANDIIII

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

#### AXDEPTH)

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

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

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

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

• A path may bounce to the light

- Or to another object
- Or in some other direction



## **Physically Based**

tic: K (depth ⊂ 1925

int = nt / nc, ddm ss2t = 1.0f - nt ); N );

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

= diffuse; = true;

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

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

AXDEPTH)

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)

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

### Path Tracing

Tracing 'photons' backwards, from the camera to the light source, by performing a random walk.

- Instead of splitting the path, we randomly evaluate one branch.
- By using many paths, we explore all possible branches.
- We have the same number of primary rays as we have 'shadow rays'.





tic: € (depth < 1555

:= inside / i ht = nt / nc, dde os2t = 1.0f - nnt 0; N(); 3)

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

= diffuse; = true;

efl + refr)) && (depth k HAADIIII

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) %

v = true;

at brdfPdf = EvaluateDiffuse( L, N) Process at3 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)

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

### Today's Agenda:

- Limitations of Whitted-style Ray Tracing
- Monte Carlo
- Path Tracing



tic: ⊾ (depth < 155

= inside / 1 it = nt / nc, do os2t = 1.0f 0, N ); 3)

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

E = 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. ... e.x + radiance.y + radiance.r) @\_\_\_\_\_

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

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:

# INFOGR – Computer Graphics

J. Bikker - April-July 2015 - Lecture 10: "Ground Truth"

# END of "Ground Truth"

next lecture: "Accelerate"

