lics & (depth < >vocco

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

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

* diffuse; = true;

. efl + refr)) && (depth < MAXDEP⊺ $\epsilon(x, x')$

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly if;

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

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

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

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

INFOMAGR – Advanced Graphics

Jacco Bikker - November 2021 - February 2022

Lecture 11 - "Various"

 $\rho(x, x', x'')I(x', x'')dx''$

Welcome!



ics & (depth < MAXDE)

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

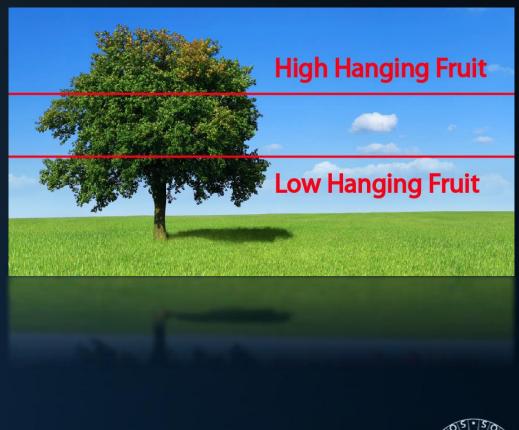
E * ((weight * cosThetaOut) / directPdf) (rac

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

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

Today's Agenda:

- Gamma Correction
- Depth of Field
- Skybox
- Spots, IES Profiles
- Microfacets





Gamma Correction

sics & (depth < NACCS

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

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

= * diffuse = true;

• efl + refr)) && (depth < MAXDER

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

AXDEPTH)

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

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

andom walk - done properly, closely follo /ive)

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

Human Eye

Digital representation of intensities is discrete: for ARGB32, we have 256 levels for red, green and blue.

The human eye is more sensitive to differences in luminance for dark shades. When encoding luminance, it is advantageous to have more detail in the lower regions, e.g.:

 $L = V^{\gamma} \Rightarrow V = L^{\frac{1}{\gamma}}$

For the human eye, $\gamma = 2.33$ is optimal^{*}.

uminance values



*: Ebner & Fairchild, Development and testing of a color space (IPT) with improved hue uniformity, 1998.

Gamma Correction

ics & (depth < 20000)

: = inside 7 1 ht = nt / nc, ddn dd os2t = 1.0f - nnt ⊂ n 0, N); ∂)

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

andom walk - done properly, closely followi /ive)

, t33 brdf = SampleDiffuse(diffuse, N, r1, r2, 8R, 8pdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

CRT Power Response

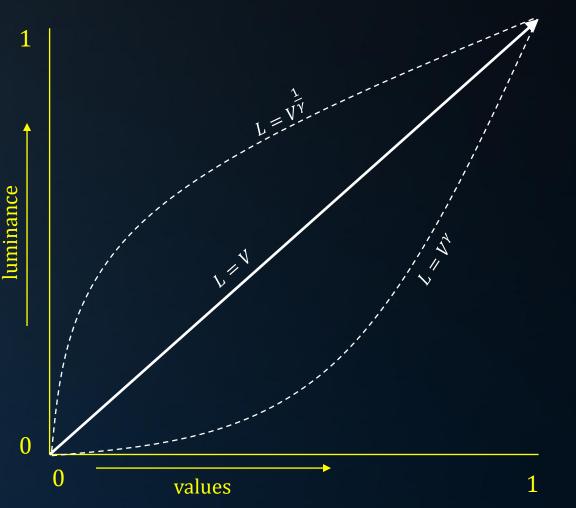
A classic CRT display converts incoming data to luminance in a non-linear way.

 $L = V^{\gamma} \Rightarrow V = L^{\frac{1}{\gamma}}$

For a typical monitor, $\gamma = 2.2$.

In other words:

- If we encode our luminance using $V = L^{\overline{\gamma}}$, it will be linear on the monitor.
- At the same time, this yields a distribution that has more detail for darker shades, which suits the human eye.





Gamma Correction

sics & (depth < Maccor

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

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

= * diffuse; = true;

efl + refr)) && (depth < MAXI

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

AXDEPTH)

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

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

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

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

Practical Gamma Correction

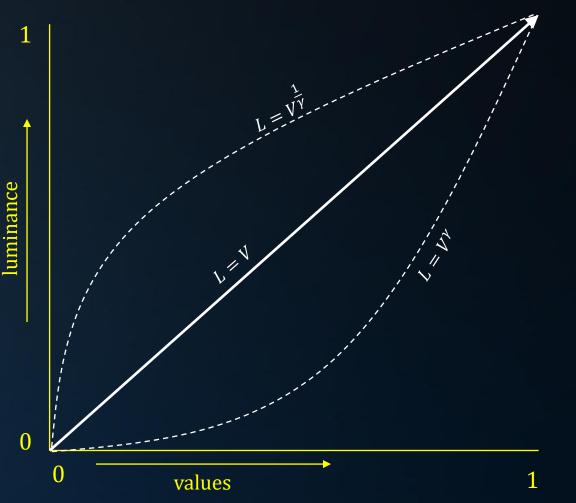
To ensure linear response of the monitor to our synthesized images, we feed the monitor adjusted data:

 $V = L^{1/2.2} \approx \sqrt{L}$

What happens if we don't do this?

1. L will be $V^{2.2}$; the image will be too dark.

A linear gradient will become a quadratic gradient; a quadratic gradient will become a cubic gradient → your lights will appear to have a very small area of influence.





Gamma Correction

tics & (depth < Notici

: = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - n⊓t 2, N); ≥)

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

= * diffuse; = true:

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

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

AXDEPTH)

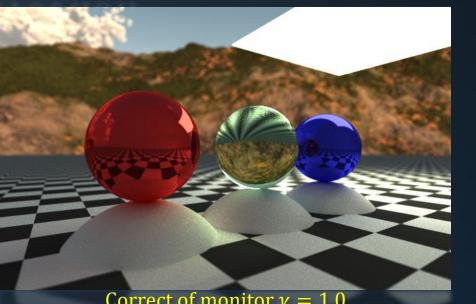
survive = SurvivalProbability(diffuse estimation - doing it properly, close If; radiance = SampleLight(&rand, I, &L, e.x + radiance.y + radiance.z) > 0) &&

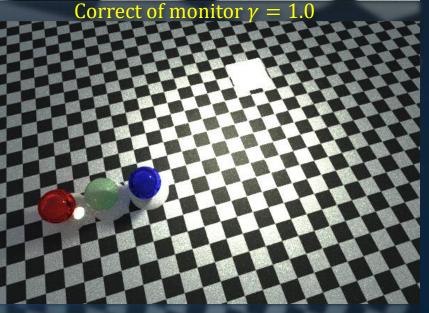
v = true; at brdfPdf = EvaluateDiffuse(L, N)

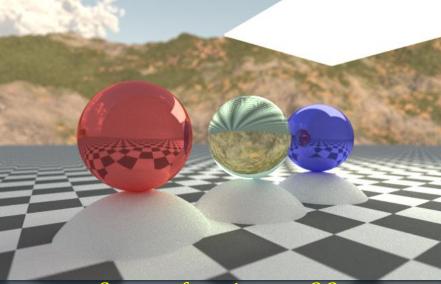
at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely follow /ive)

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











Gamma Correction

sics & (depth < Monorr

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

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

= * diffuse = true;

efl + refr)) && (depth < MAXO

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

AXDEPTH)

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

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 fo /ive)

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

Legacy

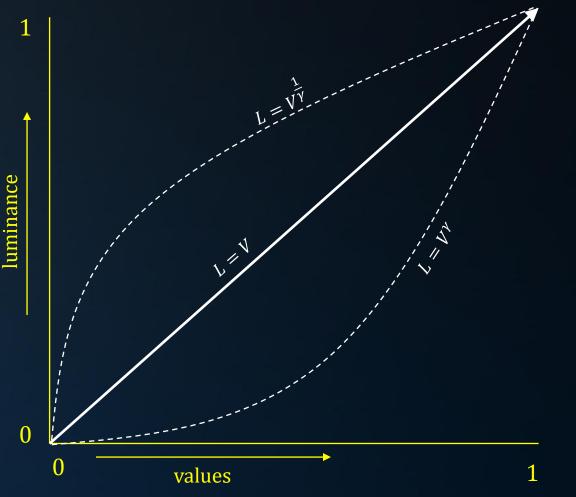
The response of a CRT is $L = V^{2.2}$; what about modern screens?

Typical laptop / desktop screens have a linear response, but expect applications to provide \sqrt{L} data... So *V* is modified (in hardware, or by the driver): $V = V^2$.

 $L \Rightarrow \sqrt{L} \Rightarrow L^2$

Not all screens take this legacy into account; especially beamers will often use $\gamma = 1$.

Gamma correct only if the hardware or video driver expects it!





Gamma Correction

tics ≹ (depth < NOCCO

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

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPT

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly, lf;

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

v = true;

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

andom walk - done properly, closely following so

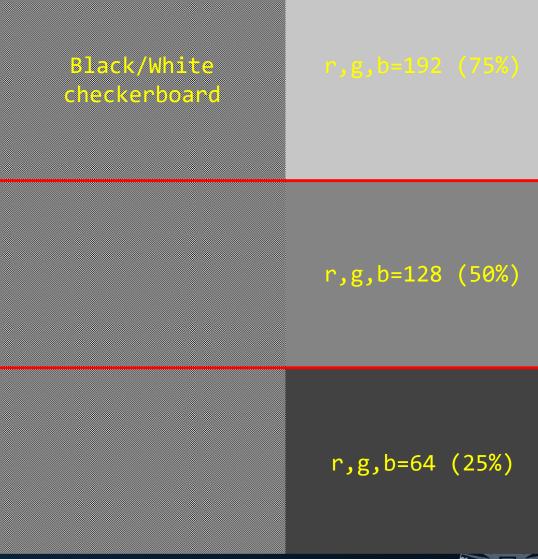
/ive)

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

Gamma Corrected Or Not?

Open gamma.gif using the windows image previewer, and zoom to the smallest level (1:1). Which bar in the right column is most similar in brightness to the right column?

	$\gamma = 1$	$\gamma = 2$
75%	0.75	0.56
50%	0.50	0.25
25%	0.25	0.06





Gamma Correction

at a = nt

), N); refl * E * diffuse;

AXDEPTH)

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

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

andom walk - done properly, closely follow /ive)

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

Consequences

How are your digital photos / DVD movies stored?

With gamma correction, ready to be sent to a display device that expects \sqrt{L}

Without gamma correction, expecting the image viewer to apply \sqrt{L}

For jpegs and mpeg video, the answer is 1: these images are already gamma corrected.

 \rightarrow Your textures may require conversion to linear space:

 $L = V^2$



ics & (depth < MAXDE)

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

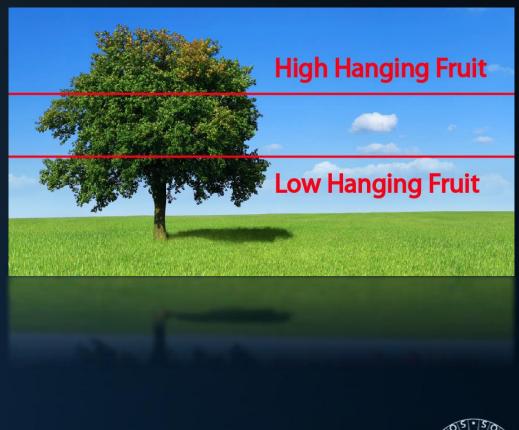
E * ((weight * cosThetaOut) / directPdf) (rac

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

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

Today's Agenda:

- Gamma Correction
- Depth of Field
- Skybox
- Spots, IES Profiles
- Microfacets

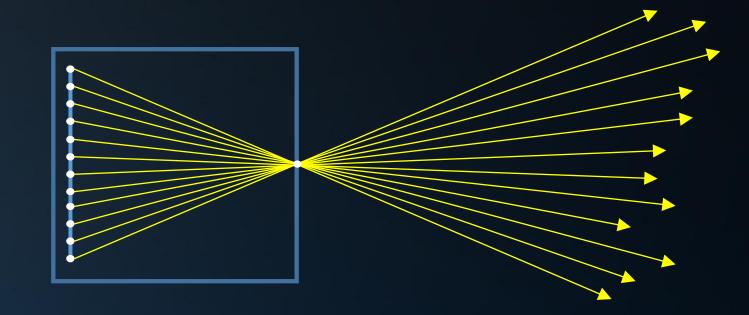




Depth of Field

Focus

A pinhole camera ensures that each pixel receives light from a single direction.





AXDEPTH)

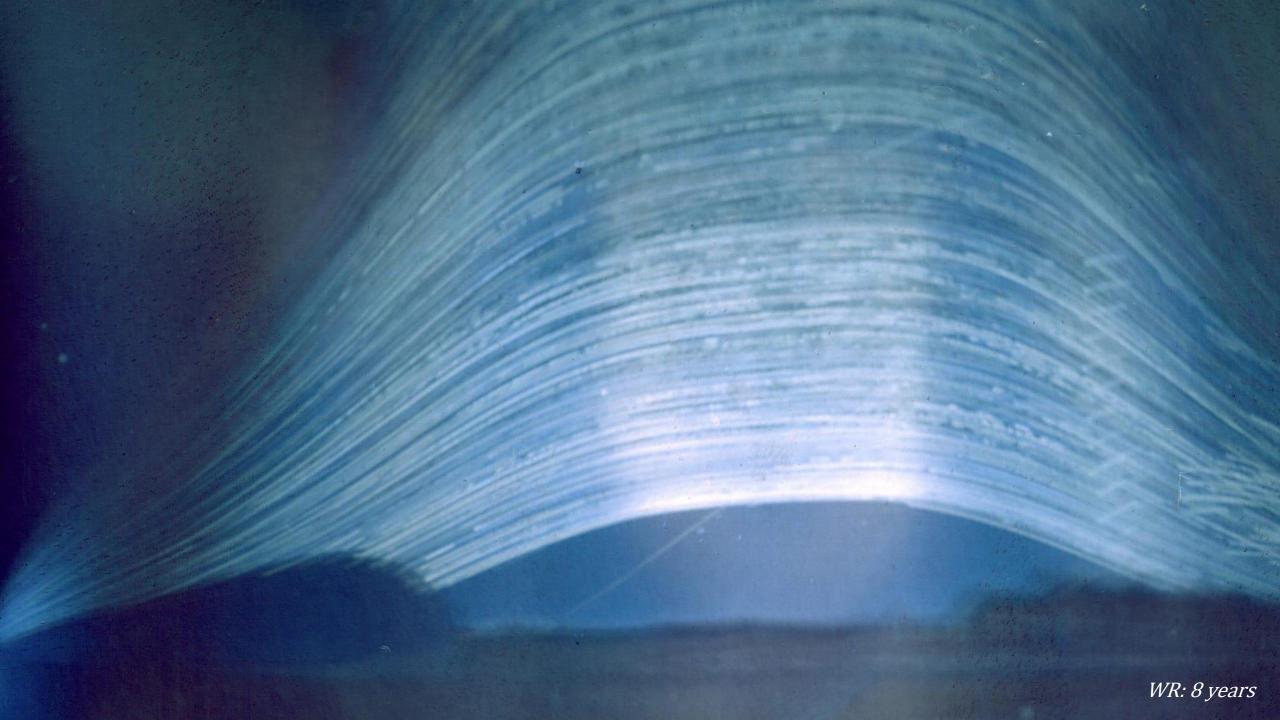
survive = SurvivalProbability(diffuse .estimation - doing it properly, close if; radiance = SampleLight(&rand, I, &L, &light) .x + radiance.y + radiance.z) > 0) &&

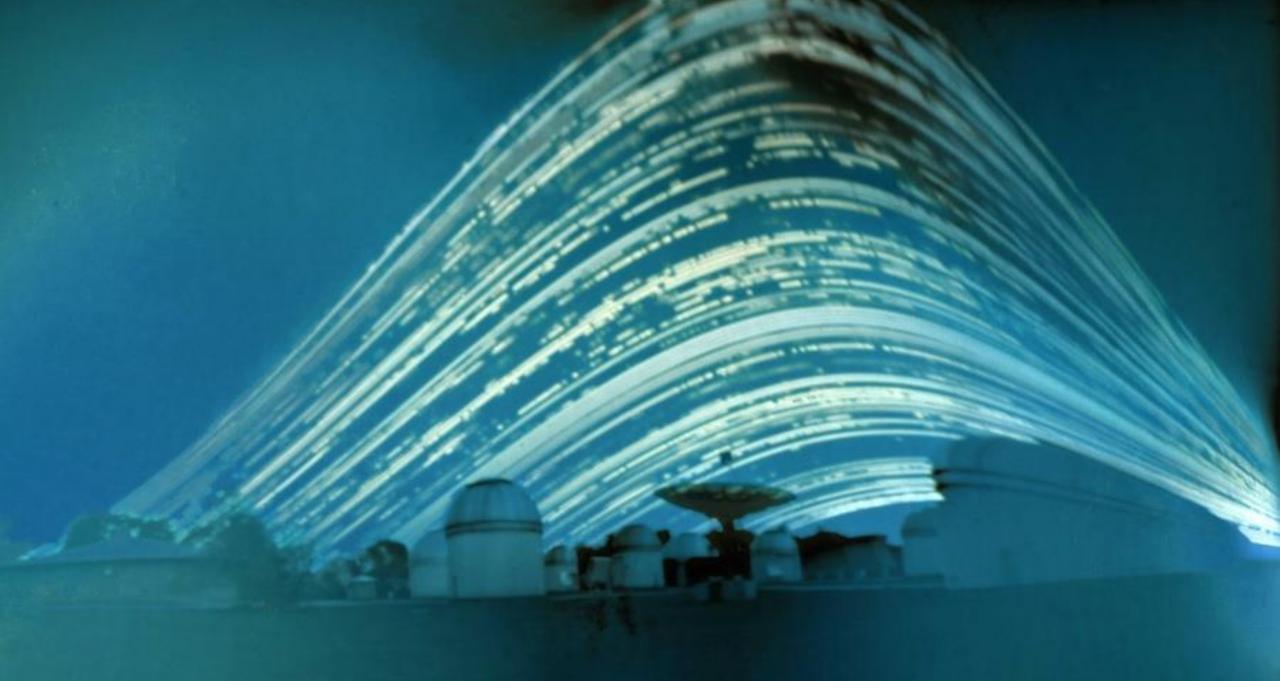
w = true; at brdfPdf = EvaluateDiffuse(L, N) Psurvive at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) (rad

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

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







Depth of Field

ics & (depth < ∧∧∞oo

: = inside ? 1 1 1 3 ht = nt / nc, ddn - 1 ps2t = 1.0f - nnt * 1 2, N); 2)

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTII

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

AXDEPTH)

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

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

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

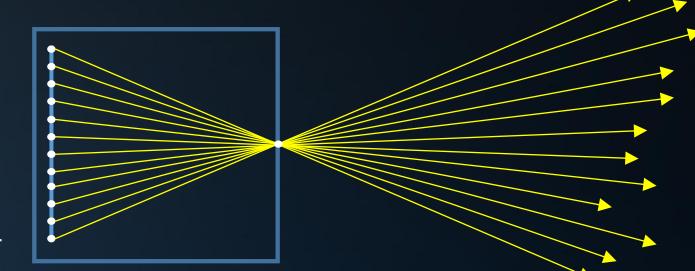
; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, apdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

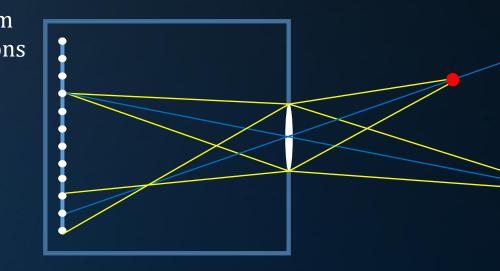
Focus

A pinhole camera ensures that each pixel receives light from a single direction.

For a true pinhole, the amount of light is zero.

Actual cameras use a lens system to direct a limited set of directions to each pixel.







Depth of Field

ics

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

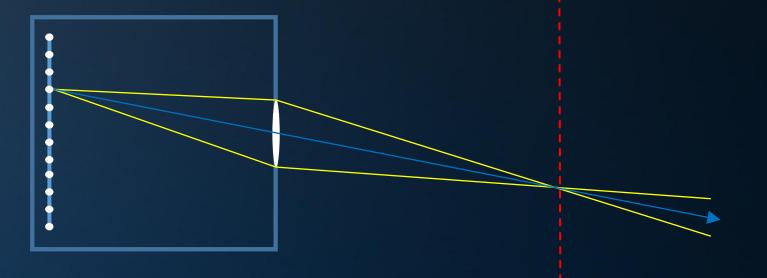
andom walk - done properly, closely following Samu /ive)

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

Focus

Objects on the focal plane appear in focus:

Light reflected from these objects to the lens end up on a single pixel on the film.





Depth of Field

nics & (depth < Notice

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

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

= * diffuse; = true;

efl + refr)) && (depth < MODEPTH

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

AXDEPTH)

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

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

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

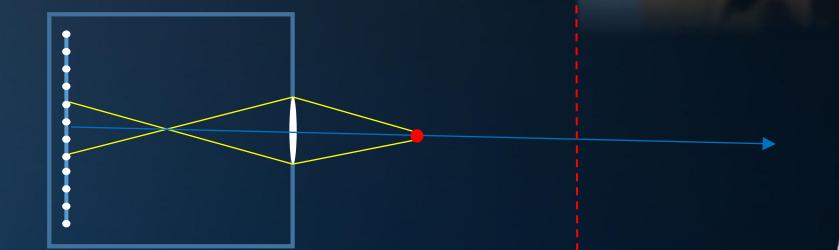
; t3 Brdf = SampleDiffuse(diffuse, N, r1, r2, 8R, 8pdf) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Focus

Objects before the focal plane appear out of focus:

Light reflected from these objects is spread out over several pixels on the film (the 'circle of confusion').







Depth of Field

ics

z = inside ? 1 (1.) ht = nt / nc, ddm os2t = 1.0f - nmt " r 2, N); 2)

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

= * diffuse; = true;

efl + refr)) && (depth < MODEPTH

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

AXDEPTH)

survive = SurvivalProbability(diffuse
estimation - doing it properly, closed
if;
adiance = SampleLight(&rand, I, &L, &light)
e.x + radiance.y + radiance.z) > 0) && (000)

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

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

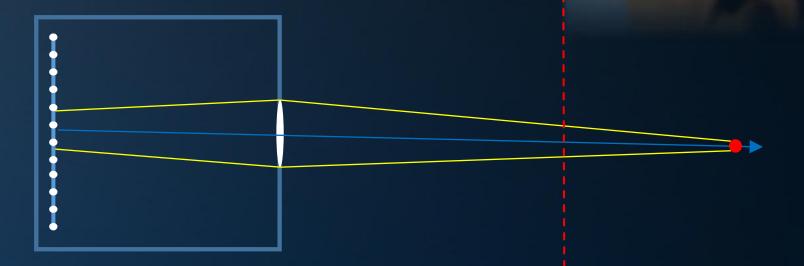
; t3 Brdf = SampleDiffuse(diffuse, N, r1, r2, &R, dpdf ; urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Focus

Objects beyond the focal plane also appear out of focus:

Light reflected from these objects is again spread out over several pixels on the film.







Depth of Field

ics & (depth < MOXOD)

: = inside ? 1 ()) ht = nt / nc, ddn (ps2t = 1.0f - nnt (), N); 3)

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEPTH)

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

AXDEPTH)

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

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

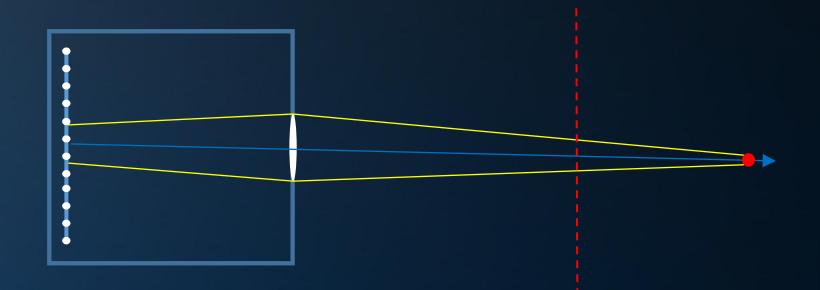
andom walk - done properly, closely following Smoot /ive)

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

Circle of Confusion

Ray tracing depth of field:

Spreading out the energy returned by a single ray over multiple pixels within the circle of confusion.





Depth of Field

Circle of Confusion

Efficient depth of field:

We place the virtual screen plane at the focal distance (from the lens). Rays are generated on the lens, and extend through each pixel.

- All rays through the pixel will hit the object near the focal plane;
- Few rays through the pixel hit the 'out of focus' objects.
- Rays through other pixels may hit the same 'out of focus' objects.

AXDEPTH)

), N);

refl * E * diffuse;

at a = ni

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

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

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

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

Depth of Field

sics & (depth < Maximum

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

at a = nt - nc, b = nt - n at Tr = 1 - (R0 + (1 - R0) Tr) R = (D $^{+}$ nnt - N $^{-}$ (d)

= * diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability(diffuse
estimation - doing it properly, closed
f;
radiance = SampleLight(&rand, I, &L, &L1
e.x + radiance.y + radiance.z) > 0) && ()

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

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

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

Generating Primary Rays

Placing the virtual screen plane at the focal distance:

Recall that a 2 \times 2 square at distance *d* yielded a FOV that could be adjusted by changing *d*.

We can adjust *d* without changing FOV by scaling the square and *d* by the same factor.

Random point on the lens: generate an (ideally uniform) random point on a disc. This is non-trivial; see Global Illumination Compendium, 19a or b. Alternatively, you can use rejection sampling.

Also nice: replace the disc with a regular n-gon.



Depth of Field

vics & (depth < >∨xxxxx

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

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

= * diffus = true;

efl + refr)) && (depth

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

AXDEPTH)

survive = SurvivalProb
estimation - doing it
if;
radiance = SampleLight
e.x + radiance.y + rad

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

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

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







ics & (depth < MAXDE)

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

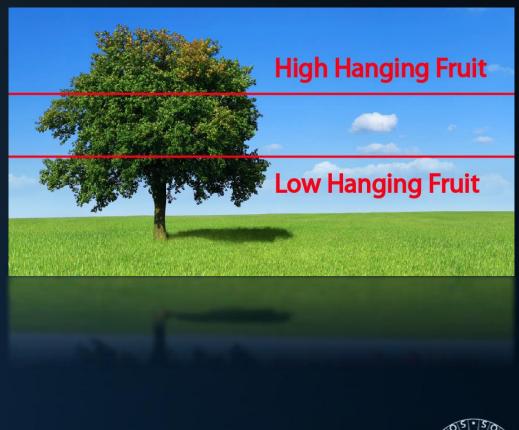
E * ((weight * cosThetaOut) / directPdf) (rac

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

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

Today's Agenda:

- Gamma Correction
- Depth of Field
- Skybox
- Spots, IES Profiles
- Microfacets





Skybox

tics & (depth < ≯VXDS)

: = inside ? 1 1 1 7 ht = nt / nc, ddn bs2t = 1.0f - nnt ? 7 2, N); 3)

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

= * diffuse; = true;

efl + refr)) && (depth < MODEPT

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

AXDEPTH)

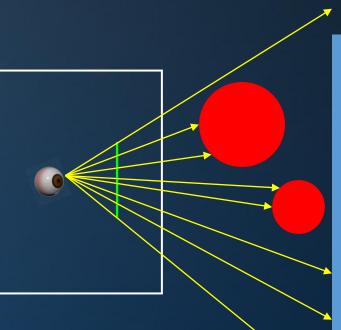
w = true; at brdfPdf = EvaluateDiffuse(L, N) * Ps at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

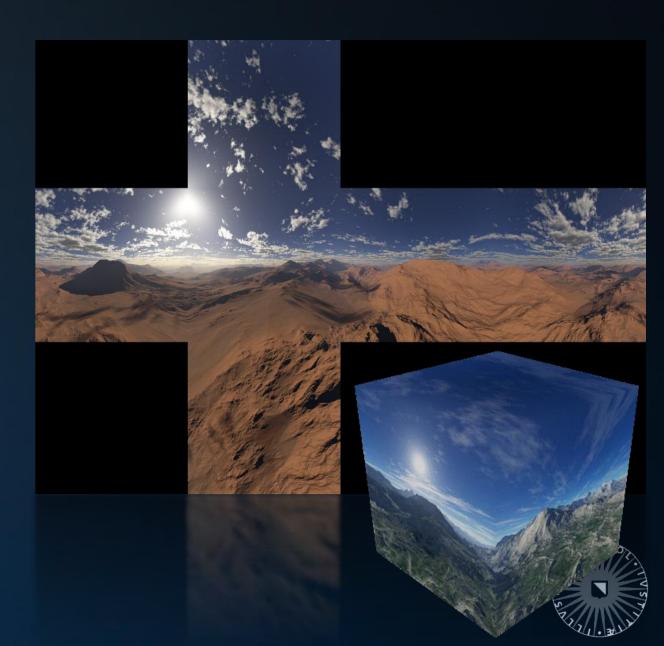
andom walk - done properly, closely foll /ive)

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

Environment Imposter

Many games use a skybox to simulate distant geometry without actually storing this geometry.





Skybox

tics & (depth < PVCDEF

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

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPT

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

AXDEPTH)

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

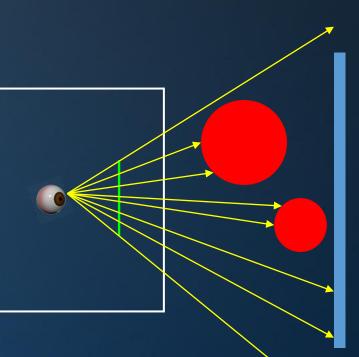
w = true; at brdfPdf = EvaluateDiffuse(L, N) * P at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely fol /ive)

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

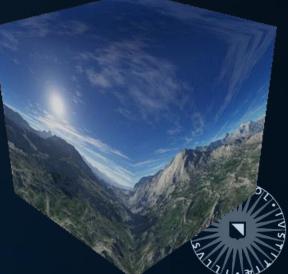
Environment Imposter

Many games use a skybox to simulate distant geometry without actually storing this geometry.



The skybox is a $1 \times 1 \times 1$ box centered around the camera: assuming the sky is at an 'infinite' distance, the location of the camera inside this box is irrelevant.

Which face of the cubemap we need to use, and where it is hit by a ray is determined on ray direction alone.



Skybox

ics & (depth < 20000

: = inside ? 1 : . . ht = nt / nc, ddn = bs2t = 1.0f - n⊓t 2, N); 2)

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDE

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

AXDEPTH)

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

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

, t33 brdf = SampleDiffuse(diffuse, N, r1, r2, 8R, 8pdf urvive; .pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

High Dynamic Range

Instead of using a skybox, we can also use an equirectangular mapping, which maps azimuth to u and elevation to v:

$$\theta = \pi(u-1), \varphi = \pi v; \ u = [0,2], \ v = [0,1].$$

Converting polar coordinates to a unit vector:

 $\vec{D} = \begin{pmatrix} \sin(\varphi)\sin(\theta) \\ \cos(\varphi) \\ -\sin(\varphi)\cos(\theta) \end{pmatrix}$

 $u, v = \begin{pmatrix} 1 + atan2(D_x, -D_z) / \pi \\ acos(D_y) / \pi \end{pmatrix}$





Skybox

hics & (depth < Not00

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

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPID

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

AXDEPTH)

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

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

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

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

High Dynamic Range

You can find HDR panoramas on Paul Debevec's page:

http://gl.ict.usc.edu/Data/HighResProbes

Note:

A HDR skydome can be used as a light source.





Skybox

nics & (depth < Motoor

: = inside ? 1 ht = nt / nc, ddn ps2t = 1.0f - nnt ... p, N); %)

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

= * diffuse; = true;

efl + refr)) && (depth < MODEC

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

AXDEPTH)

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

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

andom walk - done properly, closely fol /ive)

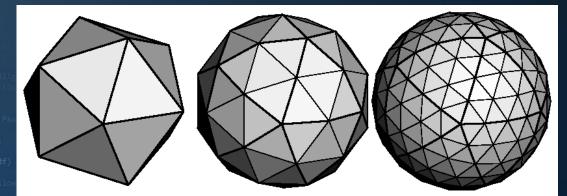
; at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, r urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Next Event Estimation for Skydomes

Useful trick:

Use the original skydome only for rays that stumble upon it.

For next event estimation, use a tessellated (hemi)sphere; assign to each triangle the average skydome color for the directions it covers.





ics & (depth < MAXDE)

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

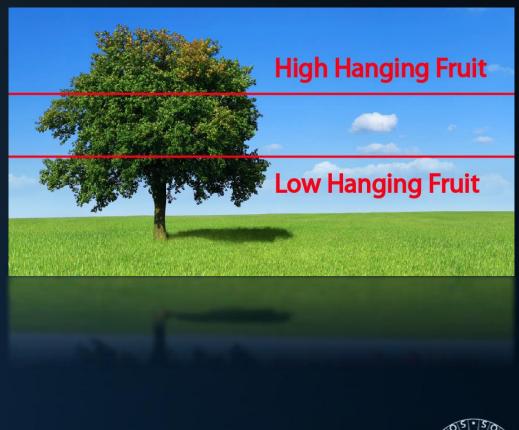
E * ((weight * cosThetaOut) / directPdf) (rac

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

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

Today's Agenda:

- Gamma Correction
- Depth of Field
- Skybox
- Spots, IES Profiles
- Microfacets





Spots & IES

Ray Tracing Spotlights

Spotlight parameters:

- Brightness
- Position, direction
- Inner angle, outer angle

We can use importance sampling for spotlights, taking into account potential contribution based on these parameters.



29

Spotlight paramotor

.

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEP)

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

AXDEPTH)

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

v = true;

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

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

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

Spots & IES

sics & (depth < MoxDer

: = inside ? 1 : . . ht = nt / nc, ddn os2t = 1.0f - n⊓t ? 2, N); 2)

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

= * diffuse = true;

efl + refr)) && (depth < MANDEPTH

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

AXDEPTH)

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

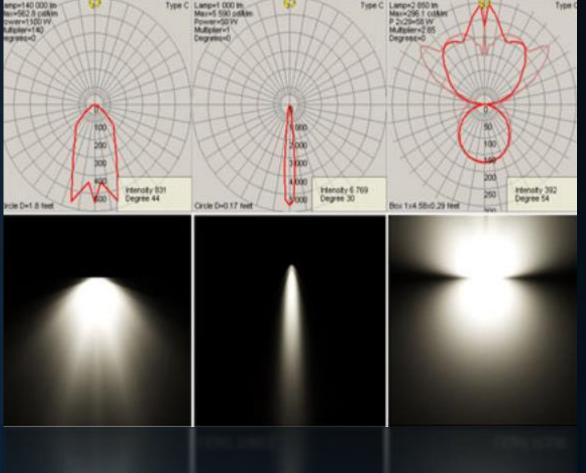
andom walk - done properly, closely following Same /ive)

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

IES Profiles

Photometric data for light sources: Measurement of the distribution of light intensity.

Can be used in e.g. 3DS Max to model lights in virtual scenes.





Lumens ⁻

Spots & IES

AXDEPTH)

survive = SurvivalProbability(diffus radiance = SampleLight(&rand, I, &L, &L e.x + radiance.y + radiance.z) > 0) 88 (v = true; at brdfPdf = EvaluateDiffuse(L, N) Psu

at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L);

E * ((weight * cosThetaOut) / directPdf) = (rate

/ive)

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

sion = true:

IESNA:LM-63-1995 [TEST] 21307						Format specification: http://lumen.iee.put.poznan.pl/kw/iesna.txt			
		HTING - PEND	OANT LUM	1INAIRE			<u> </u>		
[LUMCAT]M									
		ASTIC TUBE W		AND BOT [.]	ТОМ ОРЕ	EN			
		5 WATT INDU							
		0. LUMEN RA							
		_165W S/1 GE			ING AT 2	77 VAC A	ND 147 \	WATTS	
TILT=NONE									
1 8289 1	73 1 1 1	-1.00 0.00	1.92 1 1						
147.0000									
0 2.5 5 7.5 1	0 12.5 15 1	7.5 20 22.5 2	5 27.5 30	32.5 35 3	7.5				
40 42.5 45 4	40 42.5 45 47.5 50 52.5 55 57.5 60 62.5 65 67.5 70 72.5 75 77.5 80 82.5 85 87.5 90 92.5 95 97.5 100 102.5 105 107.5 110 Horizontal angles								
77.5 80 82.5								Horizontal angles	
112.5 115 12	17.5 120 12	2.5 125 127.5	5 130 132.	.5 135 13	7.5 140				
142.5 145 14	47.5 150 15	2.5 155 157.5	5 160 162.	.5 165 16	7.5 170				
172.5 175 17	77.5 180								Vertical angle
0									
	897.2 94		1060.8		1165.3		.1 1131		
	910.8 84		760.3	745.7	735.6	724.5	714.6	703.8	Candela values
	583.0 67		672.0	673.9	675.9	677.6	679.3	680.9	Calluela values
		2.7 681.2	678.5	676.6	680.7	684.8	683.3	680.9	
	671.2 66 ⁴		646.6	636.3	625.0	612.7	599.6	586.1	. 405.50
	558.1 54		517.8	506.6	496.4	486.5	477.0	469.6	
	482.8 502		526.0	496.0	414.4	315.6	235.7	169.7	
108.4	59.4 35.	8							Pile and



Spots & IES

hics & (depth < ™0005

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Projective Spotlight

A rectangular beam is cast from the spotlight. Illumination per direction is obtained from a bitmap.

u, *v* =?





Spots & IES

sics **& (dept**h ≪ NAC

t = inside ? 1 1 1 1 ht = nt / nc, ddn 1 552t = 1.0f - nnt " n 2, N); 8)

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPII

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly, closed if; radiance = SampleLight(&rand, I, &L, &light) 2.x + radiance.y + radiance.z) > 0) && (000)

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

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

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





Advanced Graphics – '

Spots & IES

nics & (depth < ™0000

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

= * diffuse = true;

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

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

AXDEPTH)

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

v = true;

/ive)

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

andom walk - done properly, closely following

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









ics & (depth < MAXDE)

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

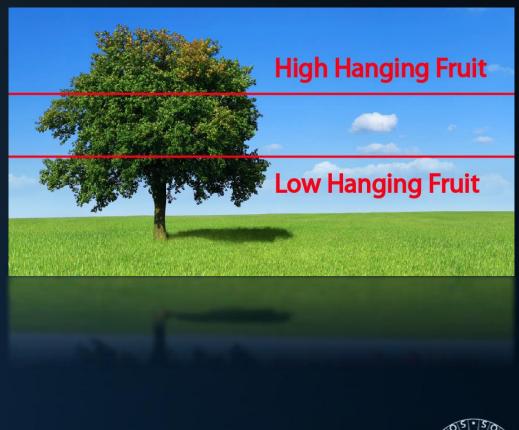
E * ((weight * cosThetaOut) / directPdf) (rac

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

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

Today's Agenda:

- Gamma Correction
- Depth of Field
- Skybox
- Spots, IES Profiles
- Microfacets





Microfacet

tics & (depth < PVXDEFT

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

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

= * diffuse = true;

efl + refr)) && (depth < MODEPTI

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

AXDEPTH)

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

v = true;

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

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

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

BRDFs Without Issues

We have two BRDFs without problems:

- I. The Lambertian BRDF
- 2. The pure specular BRDF

These are physically plausible and can be sampled. The PDF is also clear.





37

Microfacet

nics & (depth < NAXDEFT

c = inside 7 1 1 1 7 ht = nt / nc, ddn 9 7 552t = 1.0f - nnt 9 7 2, N); 8)

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

= * diffuse; = true;

efl + refr)) && (depth < MANDE

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

AXDEPTH)

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

v = true;

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

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

is brdf = SampleDiffuse(diffuse, N, r1, r2*: Torrance & Sparrow, Theory for Off-Specular Reflection from Roughened Surfaces. 1967.

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

Microfacet BRDFs*

We can simulate a broad range of materials if we assume: *at a microscopic level, the material consists of tiny specular fragments.*

- If the fragment orientations are chaotic, the material appears diffuse.
- If the fragment orientations are all the same, the material appears specular.
- Different but similar orientations yield glossy materials.





Microfacet

tics & (depth < MOCC)

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

v = true;

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

andom walk - done properly, closely following 30 /ive)

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

Microfacet BRDFs*

The Microfacet BRDF:

$$f_r(\vec{L}, \vec{V}) = \frac{F(\vec{L}, \vec{V})G(\vec{L}, \vec{V}, \vec{H})D(\vec{H})}{4(\vec{N} \cdot \vec{L})(\vec{N} \cdot \vec{V})}$$

Ingredients:

- 1. Normal distribution D
- 2. Geometry term G
- 3. Fresnel term F
- 4. Normalization





Microfacet

Normal Distribution



Microfacet BRDF, ingredient 1: $D(\vec{H}) \rightarrow$ the *normal distribution function*. Parameter \vec{H} : the halfway vector:

 $f_r(\vec{V},\vec{L}) = \cdots$

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it properly, close Hf; radiance = SampleLight(&rand, I, &L, J

v = true; at brdfPdf = EvaluateDiffu; at3 factor = diffuse * INV;

at weight = Mis2(directPdf, brdfPdf

at cosThetaOut = dot(N, L);

E * ((weight * cosThetaOut) / directPdf) = (Pd

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

; at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &pdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); ;;on = true; A microfacet that reflects \vec{L} towards \vec{V} must have a normal halfway \vec{V} and \vec{L} :

 $f_r(x,\theta_i,\theta_o) = \frac{L_o(x,\theta_o)}{L_i(x,\theta_i)\cos\theta_i}$

H = normalize(V + L).





Microfacet

Normal Distribution

Intuitive choices for D:

 $D(\vec{H}) = C$: microfacet normals are equally distributed \rightarrow diffuse material.

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

= * diffuse = true;

efl + refr)) && (depth < N

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

AXDEPTH)

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

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

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

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

Good practical choice for D: the Blinn-Phong distribution;

 $D(\vec{H}) = \begin{cases} \infty, f \text{ or } \vec{H} = (0,0,1) \\ 0, \text{ otherwise} \end{cases}$: all microfacet normals are $(0,0,1) \rightarrow$ pure specular.

 $D(\vec{H}) = \frac{\alpha + 2}{2\pi} \underline{\left(\vec{N} \cdot \vec{H}\right)}^{\alpha}$



Microfacet

tics & (depth < MOCC)

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

= * diffuse = true;

efl + refr)) && (depth < MANDEPTH

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

AXDEPTH)

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

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

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

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

Geometry Term

Microfacet BRDF, ingredient 2: $G(\vec{V}, \vec{L}, \vec{H}) \rightarrow$ the *geometry term*. It describes what fraction of a microsurface with normal \vec{H} is visible in both directions \vec{L} and \vec{V} .

Geometry Term

Microfacet BRDF, ingredient 2: $G(\vec{V}, \vec{L}, \vec{H}) \rightarrow$ the geometry term. It describes what fraction of a microsurface with normal \vec{H} is visible in both directions \vec{L} and \vec{V} .



nics & (depth < NOCC)

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

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

*: Physically Based Rendering, page 455
urvive;
pdf;
n = E * brdf * (dot(N, R) / pdf);
sign = true:

Geometry Term

Intuitive choice for G:

 $G(\vec{V}, \vec{L}, \vec{H}) = 1$: no occlusion.

Good practical choice for G*:

$$G(\vec{V}, \vec{L}, \vec{H}) = \min(1, \min\left(\frac{2(\vec{N} \cdot \vec{H})(\vec{N} \cdot \vec{V})}{\vec{V} \cdot \vec{H}}, \frac{2(\vec{N} \cdot \vec{H})(\vec{N} \cdot \vec{L})}{\vec{V} \cdot \vec{H}}\right)$$

Microfacet

sics & (depth < ⊅00000

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

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

= * diffuse; = true;

efl + refr)) && (depth < MACCE

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

AXDEPTH)

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

w = true; at brdfPdf = EvaluateDiffuse(L

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

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

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

Fresnel Term

Microfacet BRDF, ingredient 3: $F(\vec{L}, \vec{H}) \rightarrow$ the *Fresnel term*.

So far, we assumed that the light reflected by a specular surface is only modulated by the material color.

This is not true for dielectrics: here we use the Fresnel equations to determine reflection.

In nature, Fresnel does not just apply to dielectrics.



Wi

n

Microfacet

tics & (depth < Notoer)

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

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

= * diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(diffu estimation - doing it properly, class 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 f /ive)

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

Fresnel Term

Iron is specular, but reflectivity differs depending on incident angle.

Aluminum is even more interesting: reflectivity depends on wavelength. The three lines in the graph:

Top: blue, middle: green, bottom: red.

Copper takes this to extremes: at grazing angles, it appears white. The lines in the graph:

Top: red, middle: green, bottom: blue. '

(hence its reddish appearance)

0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0 20 30 40 50 60 70 80 90 10angle of incidence θ_i \equiv copper \equiv aluminum — iron — diamond — glass water From "Real-time Rendering, 3rd edition, A. K. Peters.

Microfacet

sics & (depth < MANDEr

t = inside ? 1 . . . ht = nt / nc, ddn . . bs2t = 1.0f - n⊓t . . 2, N); 2)

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

= * diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability(diffu estimation - doing it properly, co 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 S. /ive)

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

Fresnel Term

Iron

Gold

Silver

Copper

Aluminum

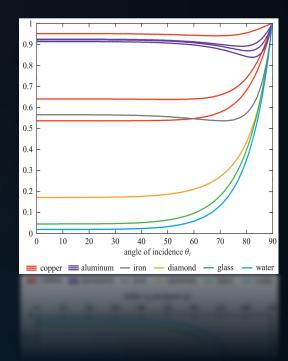
For Fresnel, we use Schlick's approximation:

 $F_r = k_{specular} + (1 - k_{specular})(1 - (\vec{L} \cdot \vec{H}))^5$

Note that this is calculated per color channel (k_{specular} is an rgb triplet).

Values for $k_{specular}$ for various materials:

0.56, 0.57, 0.58
0.95, 0.64, 0.54
1.00, 0.71, 0.29
0.91, 0.92, 0.92
0.95, 0.93, 0.88





Microfacet

Bringing it All Together

The Microfacet BRDF:

$$f_r(\vec{L}, \vec{V}) = \frac{F(\vec{L}, \vec{V})G(\vec{L}, \vec{V}, \vec{H})D(\vec{H})}{4(\vec{N} \cdot \vec{L})(\vec{N} \cdot \vec{V})}$$

$$D(\vec{H}) = \frac{\alpha + 2}{2\pi} \underline{(\vec{N} \cdot \vec{H})}^{\alpha}$$

$$G(\vec{V}, \vec{L}, \vec{H}) = \min(1, \min\left(\frac{2(\vec{N} \cdot \vec{H})(\vec{N} \cdot \vec{V})}{\vec{V} \cdot \vec{H}}, \frac{2(\vec{N} \cdot \vec{H})(\vec{N} \cdot \vec{L})}{\vec{V} \cdot \vec{H}}\right)$$

For a full derivation of the denominator of the BRDF, see Physically Based Rendering, section 8.4.2.

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

AXDEPTH)

v = true;

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

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

 $F_r = k_{specular} + (1 - k_{specular})(1 - (\vec{L} \cdot \vec{H}))^5$



hics & (depth < ™0000

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

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

= * diffuse; = true;

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

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

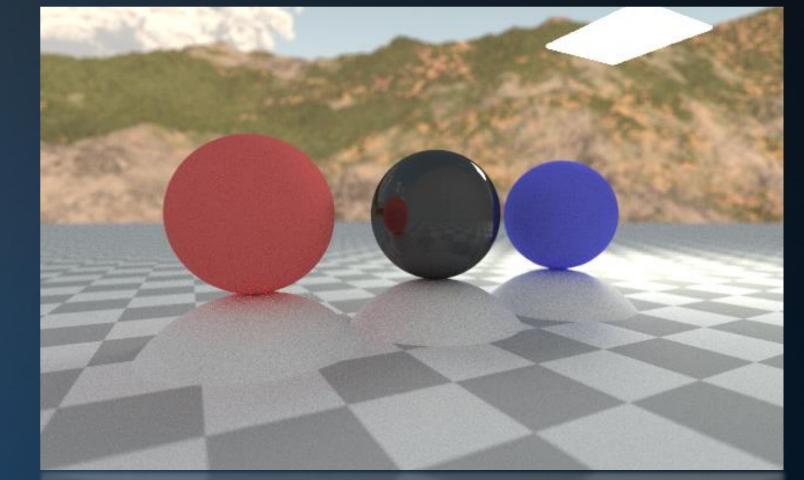
AXDEPTH)

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

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

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

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



Lambertian BRDF



tics & (depth < ™0000

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

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTO

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

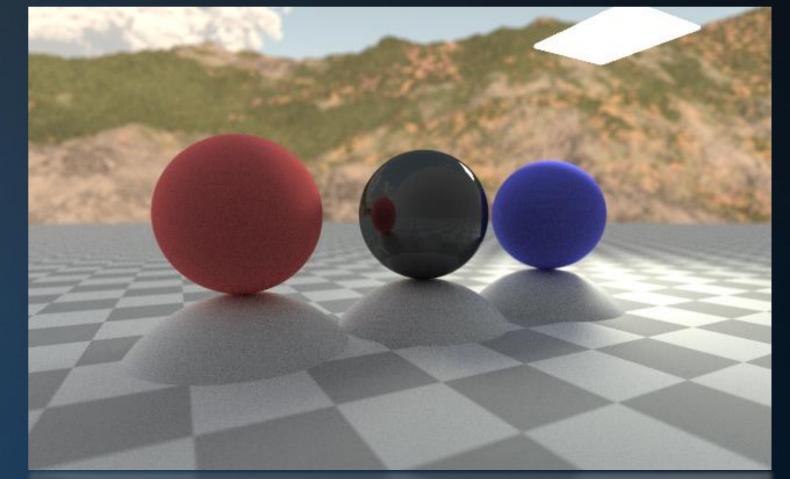
AXDEPTH)

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

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

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

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





hics & (depth < ™0000

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPOID

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

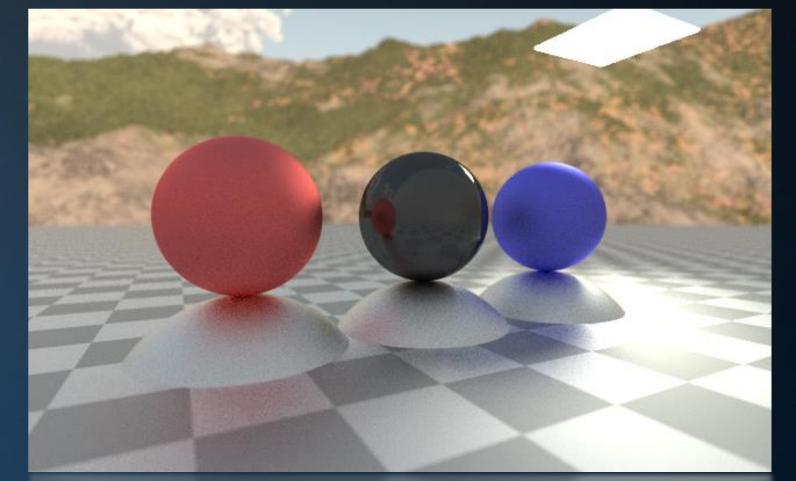
AXDEPTH)

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

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

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

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





hics & (depth < ™0000

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

= * diffuse; = true;

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

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

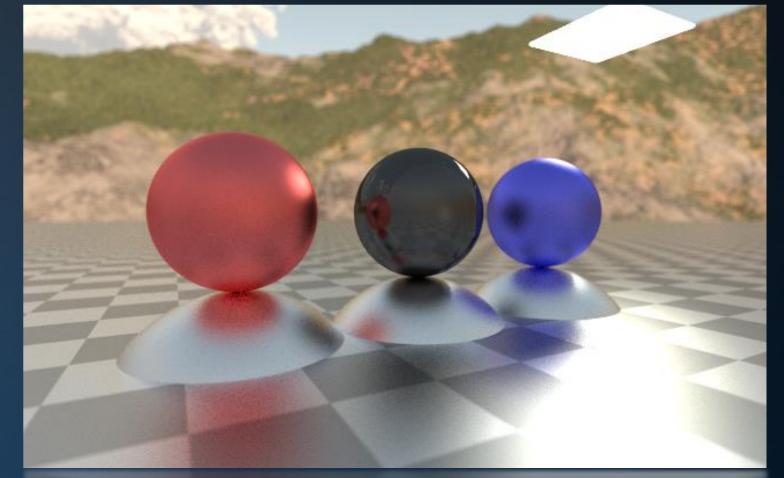
AXDEPTH)

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

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

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

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





tics & (depth < ™0000

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

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

= * diffuse; = true;

efl + refr)) && (depth < NONDERTION

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

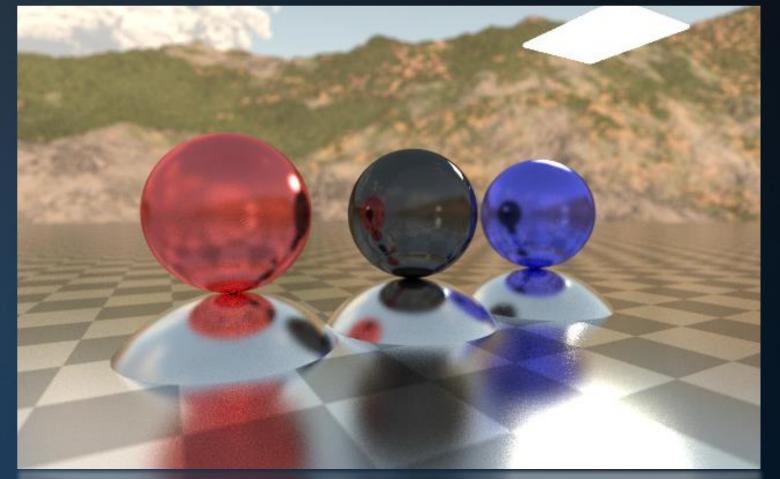
AXDEPTH)

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

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

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

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





tics & (depth < ™0000

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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





hics & (depth < ™0000

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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



Specular BRDF



ics & (depth < MAXDE)

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

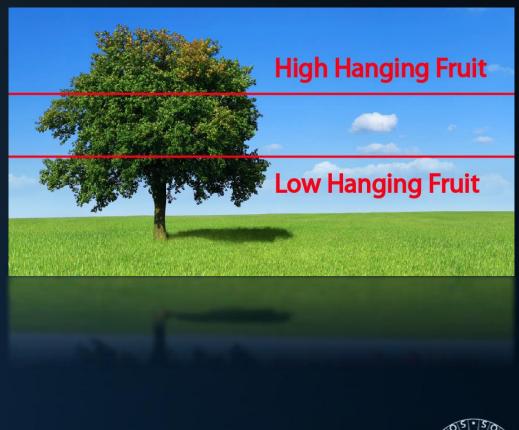
E * ((weight * cosThetaOut) / directPdf) (rac

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

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

Today's Agenda:

- Gamma Correction
- Depth of Field
- Skybox
- Spots, IES Profiles
- Microfacets





hics & (depth < Not000

: = inside ? 1 |]] ht = nt / nc, ddn =] bs2t = 1.0f - nmt = on D, N); B)

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

= * diffuse; = true;

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

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, N) Psurvive at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) (1800)

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

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

INFOMAGR – Advanced Graphics

Jacco Bikker - November 2021 - February 2022

END of "Various"

next: Assignment 2 deadline, then: Christmas Break

