sics & (depth < MODE)

c = inside 7 1 ht = nt / nc, ddn os2t = 1.0f - nmt 0, N); 2)

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

= * diffuse = true;

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

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 <u>+</u> radiance.<u>z</u>) 0 8.8 _____

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 Sec. /ive)

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

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2019

Lecture 5: "Graphics Fundamentals"

Welcome!



ics & (depth < POODEFT

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Today's Agenda:

Rasters

- Colors
- Ray Tracing
- Assignment P2



Discretization

hics & (depth < Mox000

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

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

= * diffuse = true:

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

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

AXDEPTH)

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

v = true;

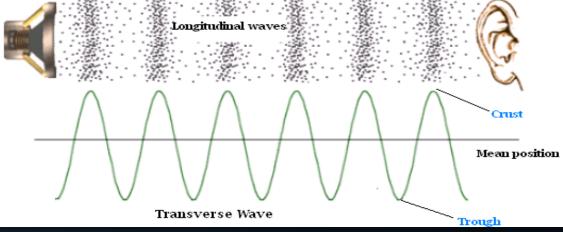
at brdfPdf = EvaluateDiffuse(L, N) * Psurvis at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) (see

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

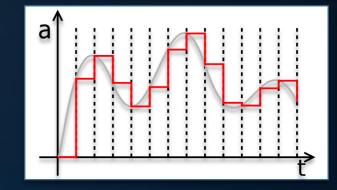
.

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

	li E									









Discretization

tics & (depth < Mox000

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

= * diffuse; = true;

• •fl + refr)) && (denth < MAXDER)

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

AXDEPTH)

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

v = true;

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

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

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

Rasterization:

"Converting a vector image into a raster image for output on a video display or printer or storage in a bitmap file format."

(Wikipedia)



Rasterization

hics & (depth < MODEFT

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

= * diffuse; = true;

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

AXDEPTH)

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

v = true;

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

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

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

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

Improving rasterization:

1. Increase resolution;



Rasterization

nics & (depth < MODEFT

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

= * diffuse; = true;

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

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

AXDEPTH)

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

v = true;

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

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

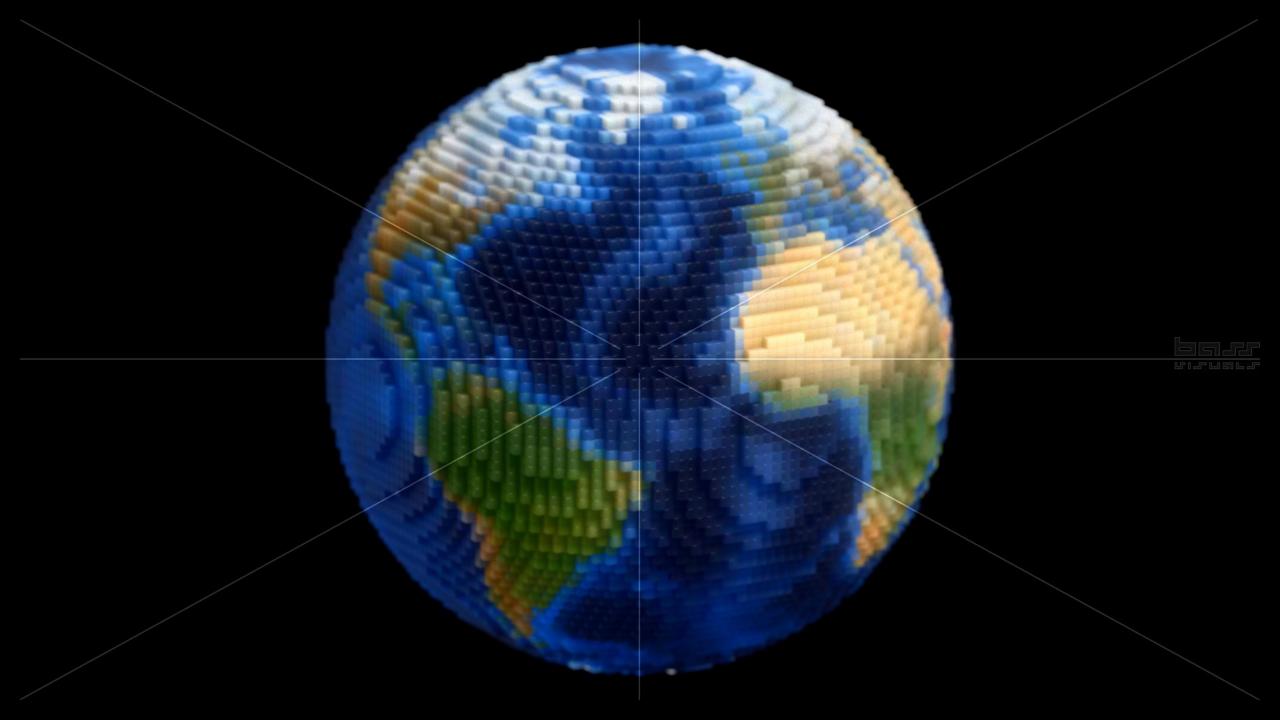
andom walk - done properly, closely following Same /ive)

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

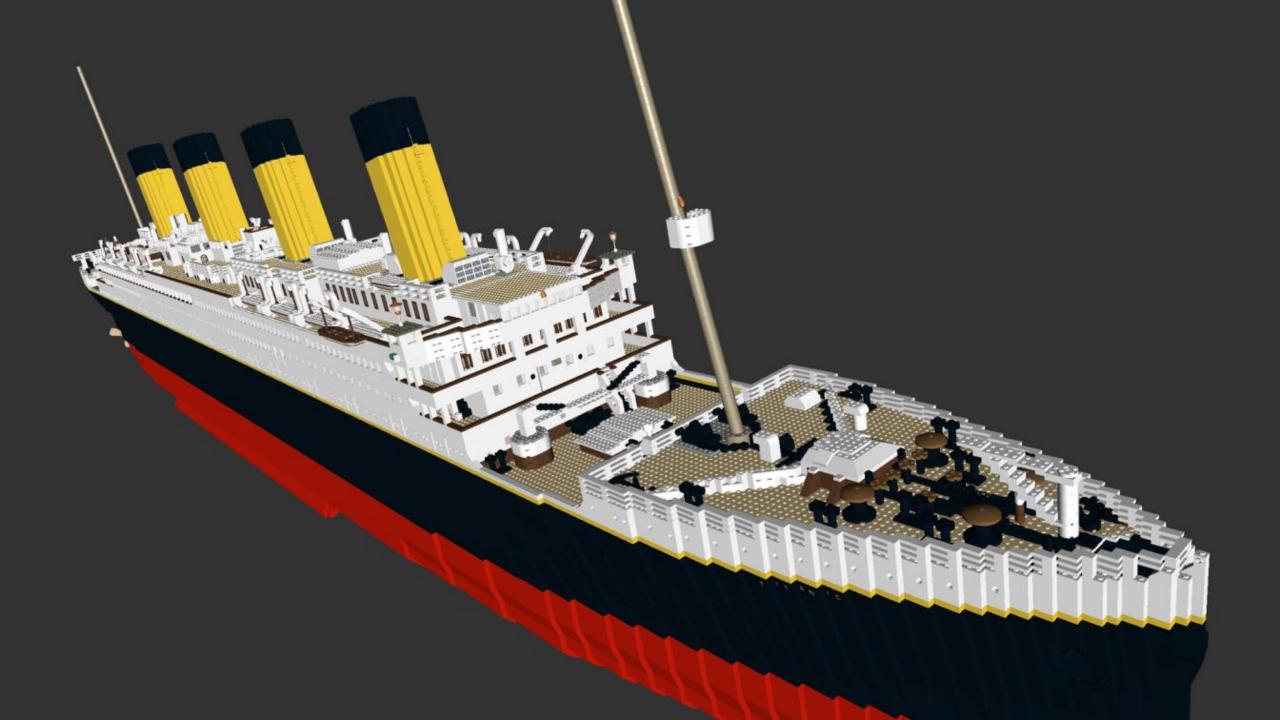
Improving rasterization:

- 1. Increase resolution;
- 2. Anti-aliasing;
- 3. Animation.









Discretization

tics & (depth < ™0000

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

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

= * diffuse = true;

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

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

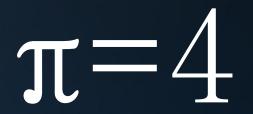
v = true;

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

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:





 $a^{1+}b^{1}=c^{1}$



CRT – Cathode Ray Tube

nics & (depth < ≫000000

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

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

= * diffuse = true;

. efl + refr)) && (depth < MAXDEP⊺

), N); ∙efl * E * diffi = true;

AXDEPTH)

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

v = true; at brdfPdf

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 Sou. /ive)

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

Physical implementation – origins

Electron beam zig-zagging over a fluorescent screen.



CRT – Cathode Ray Tube

tics & (depth < PVCDEF

c = inside ? 1 ht = nt / nc, ddn os2t = 1.0f - nnt ? D, N); D)

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPT

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

AXDEPTH)

survive = SurvivalProbability(differences)
estimation - doing it properly, club;
if;
radiance = SampleLight(&rand, I, &L,
e.x + radiance.y > _0) &&

v = true;

at brdfPdf = EvaluateDiffuse(L, N) * Psuevi 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)

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

P,O X

Physical implementation – consequences

- Origin in the top-left corner of the screen
- Axis system directly related to pixel count



CRT – Cathode Ray Tube

tics & (depth < NOCC⊂

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

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEDI

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

AXDEPTH)

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

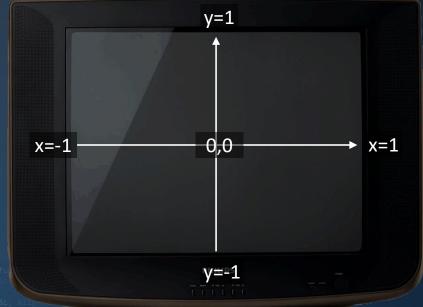
; at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &pdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Physical implementation – consequences

- Origin in the top-left corner of the screen
- Axis system directly related to pixel count

Not the coordinate system we expected...





CRT – Cathode Ray Tube

nics & (depth < NACOS

: = inside ? l : . . ht = nt / nc, ddn os2t = 1.0f - nmt ° 0, N); ∂)

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEDI

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

AXDEPTH)

survive = SurvivalProbability(diffe estimation - doing it properly coo df; radiance = SampleLight(&rand, T, &t e.x + radiance.y + radiance.z) > 01_24

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) = (Pd

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:

Proper screen coordinates

- Pixel coordinates are only relevant for the final step: plotting pixels
- Decouple the 2D screen coordinates in your game / app from the physical mapping.



Frame rate

nics & (depth < MoxOC⊂

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

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

= * diffus = true:

efl + refr)) && (depth < MODEPIII

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

AXDEPTH)

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

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

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



PAL: 25fps NTSC: 30fps (actually: 29.97) Typical laptop screen: 60Hz High-end monitors: 120-240Hz Cartoons: 12-15fps

Human eye: 'Frame-less' Not a raster.

How many fps / megapixels is 'enough'?



Frame rate

tics & (depth < Not0000

: = inside 7 1 1 1 1 ht = nt / nc, ddn - 1 bs2t = 1.0f - n⊓t - 1 2, N); ≥)

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

= * diffuse; = true;

efl + refr)) && (depth < MA

), N); refl * E * diffus = 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 following : /ive)

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

0 ms20 ms40 ms60 msFrame 1Frame 2Frame 3Sim 1Sim 2Sim 3Input 1Input 2Input 3

Even 100 frames per second may result in a noticeable delay of 30ms.

A very high frame rate minimizes the response time of the simulation.





Generating images

tics & (depth < Modes

t = inside ? 1 1 1 0 ht = nt / nc, ddn os2t = 1.0f - nnt ⊂ n 2, N); ≷)

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEPII

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, N) at3 factor = diffuse * INVPI; built = win0(discont for the state built = win0(discon

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

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

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

Rendering: <u>on a raster</u> *"The process of generating an image from a 2D or 3D model by means of a computer program."* (Wikipedia)

Two main methods:

1. Ray tracing: for each pixel: what color do we assign to it?

2. Rasterization: for each triangle, which pixels does it affect?



ics & (depth < POODEFT

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Today's Agenda:

Rasters

- Colors
- Ray Tracing
- Assignment P2



nics & (depth < Maxim

c = inside ? 1 : . . ht = nt / nc, ddn ss2t = 1.0f - nnt 2, N); 3)

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

= * diffuse; = true;

efl + refr)) && (depth < MOXDEPTH

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

AXDEPTH)

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

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

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:

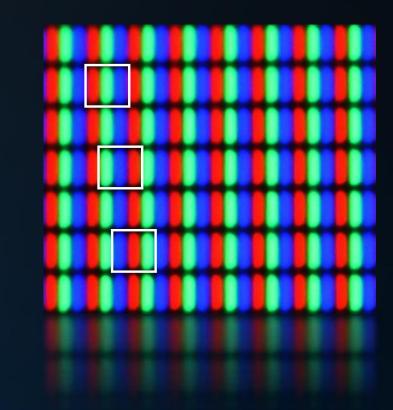
Color representation

Computer screens emit light in three colors: red, green and blue.

By additively mixing these, we can produce most colors: from black (red, green and blue turned off) to white (red, green and blue at full brightness).

In computer graphics, colors are stored in discrete form. This has implications for:

- Color resolution (i.e., number of unique values per component);
- Maximum brightness (i.e., range of component values).







radiance = SampleLight(&rand, I, &L, &light0) 2.x + radiance.y + radiance.z) > 0) && (dot)

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 Sec. vive)

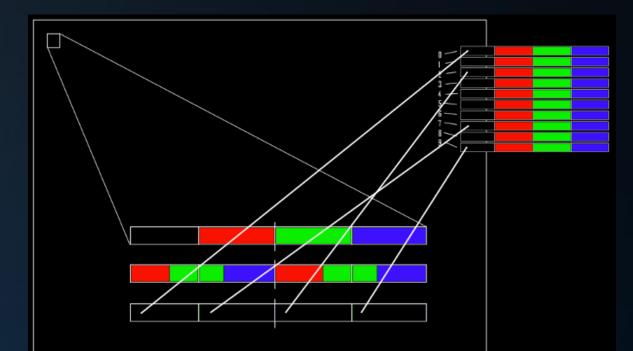
; at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &pdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Color representation

The most common color representation is 32-bit ARGB, which stores red, green and blue as 8 bit values (0..255).

Alternatively, we can use 16 bit for one pixel (RGB 565),

or a color palette. In that case, one byte is used per pixel, but only 256 unique colors can be used for the image.





Color representation

sics & (depth < Notion

: = inside ? | ... ht = nt / nc, ddn os2t = 1.0f - nmt ? 2, 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 < MAXDEPTH

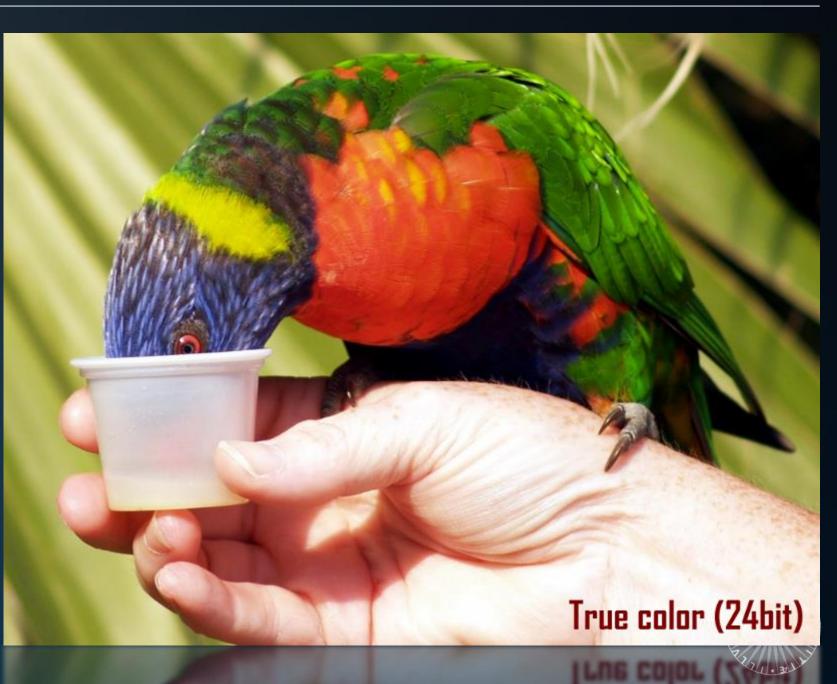
), N); refl * E * diffus: = true;

AXDEPTH)

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 Soul /ive)

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



Color representation

sics & (depth < MODE=1

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

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

survive = SurvivalProbability(diffuse
estimation - doing it properly,
if;
radiance = SampleLight(&rand, I, &L, &light(&rand, &

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) (nad

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

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



INFOGR – Lecture 5 – "Graphics Fundamentals"

Colors

Color representation

hics & (depth < ≯voces

: = inside ? | ht = nt / nc, ddn bs2t = 1.0f - nmt 2, 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 < MAXDEPTH

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

AXDEPTH)

survive = SurvivalProbability(diffuse .estimation - doing it properly, df; radiance = SampleLight(&rand, I, &L, &light(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) (rad

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

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



21.30)

ics & (depth < MoxOOFT

: = inside ? 1 : . . ht = nt / nc, ddn . 552t = 1.0f - nmt ?), 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 < MADE

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

AXDEPTH)

survive = SurvivalProbabilit
estimation - doing it prop#f;
radiance = SampleLight(&ran
e.x + radiance.y + radiance.

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

andom walk - done properly **/ive)**

; at3 brdf = SampleDiffuse(d urvive; pdf; n = E * brdf * (dot(N, R)

sion = true:

Color representation

Textures can typically safely be stored as palletized images.

Using a smaller palette will result in smaller compressed files.



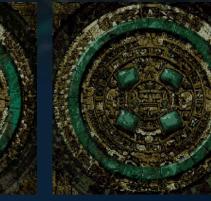














ics & (depth < 2000000

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEPT

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

AXDEPTH)

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

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

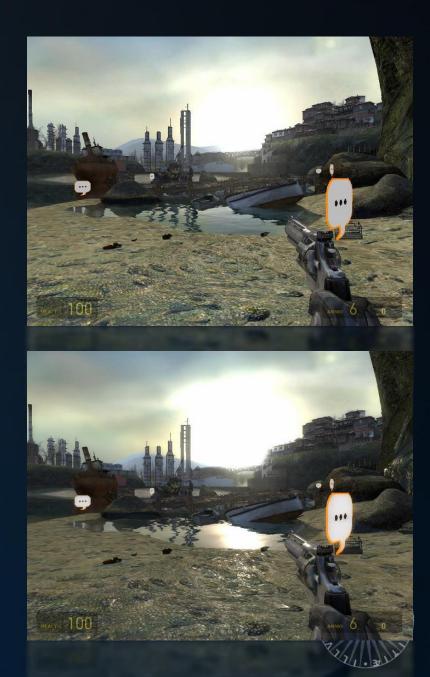
Color representation

Using a fixed range (0:0:0 ... 255:255:255) places a cap on the maximum brightness that can be represented:

- A white sheet of paper: (255,255,255)
- A bright sky: (255,255,255)

The difference becomes apparent when we look at the sky and the sheet of paper through sunglasses.

(or, when the sky is reflected in murky water)



ics & (depth < PACES)

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

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

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

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

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

Color representation

For realistic rendering, it is important to use an internal color representation with a much greater range than 0..255 per color component.

HDR: High Dynamic Range;

We store one float value per color component.

Including alpha, this requires 128bit per pixel.



ics & (depth < PAXDOR

: = inside ? 1 ()) ht = nt / nc, ddn ss2t = 1.0f - n⊓t * ∩ 2, N); ≫)

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

= * diffuse; = true;

efl + refr)) && (depth < MODEPTI

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly, closed Hf; 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) * (nad

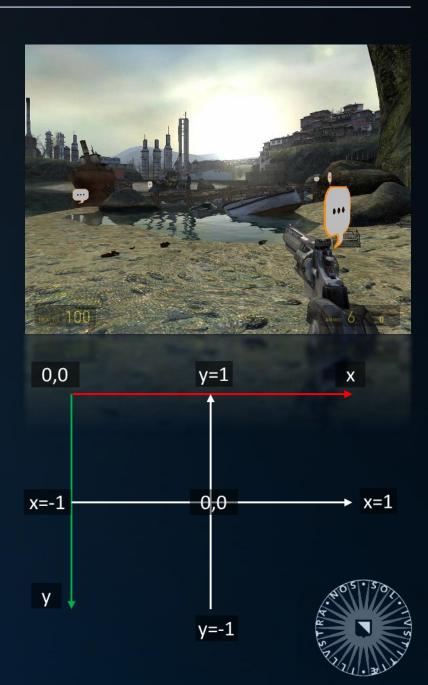
andom walk - done properly, closely following Same /ive)

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

Color representation

Like pixel coordinates, pixel colors on the physical screen are only useful for final pixel plotting:

Do not use integer colors clamped to [0..255] internally, unless you have a good reason for this.



ics & (depth < POODEFT

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Today's Agenda:

Rasters

- Colors
- Ray Tracing
- Assignment P2



ics Filenth - 1117

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

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

= * diffuse; = true;

efl + refr)) && (depth < MODEPILL

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

AXDEPTH)

survive = Surviv estimation - do Hf; radiance = Sampl e.x + radiance.y

w = true; at brdfPdf = Eva at3 factor = dif at weight = Mis2 at cosThetaOut = E * ((weight *

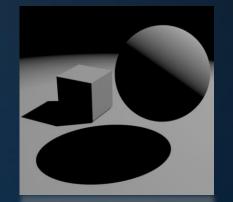
andom walk -/ive)

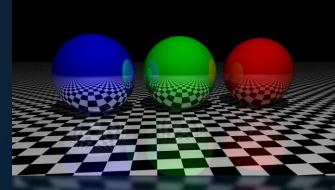
; at3 brdf = SampleDiffuse(diffuse, N, r1, r2. urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

PART 1: Introduction & shading (today, Thursday)

PART 2: Reflections, refraction, absorption (next week)

PART 3: Path Tracing (later)







Ray Tracing:

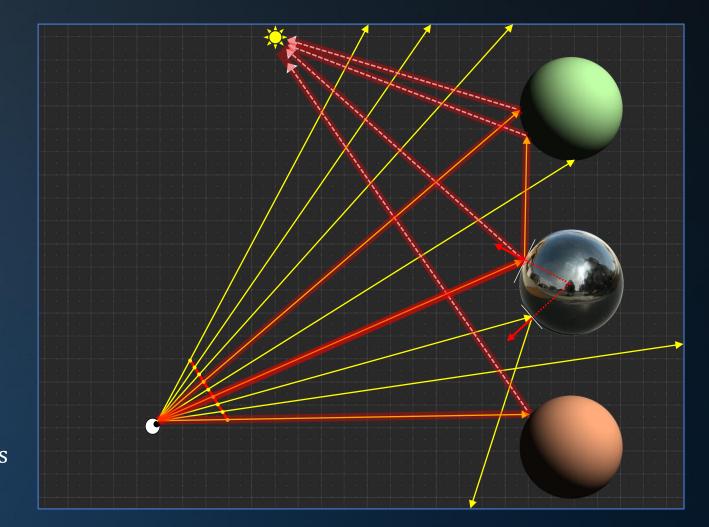
World space

- Geometry
- Eye

at a = nt

AXDEPTH)

- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays
- survive = SurvivalProbability(diffuse estimation - doing it proper Hf; radiance = SampleLight(&rand, ison transport e.x + radiance.y + radiance.z) > 0 & &
- w = true; at brdfPdf = EvaluateDiffuse(L, N Extension rays at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L)Light transport E * ((weight * cosThetaOut)Light transport
- andom walk done properly, closely following Sov /ive)
- ; at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &pdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:





Ray Tracing:

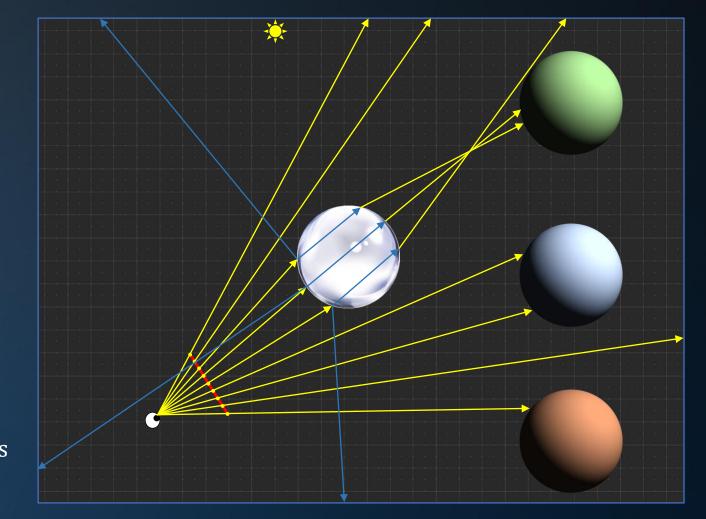
World space

- Geometry
- Eye

at a = nt

AXDEPTH)

- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays
- survive = SurvivalProbability(diffuse estimation - doing it property H; Light transport adiance = SampleLight(&rand, is a stransport e.x + radiance.y + radiance.z) > 0
- w = true; at brdfPdf = EvaluateDiffuse(L, N Extension rays at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L)Light transport E * ((weight * cosThetaOut)Light transport
- andom walk done properly, closely following Sec /ive)
- ; at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, 8pdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:







Ray Tracing:

World space

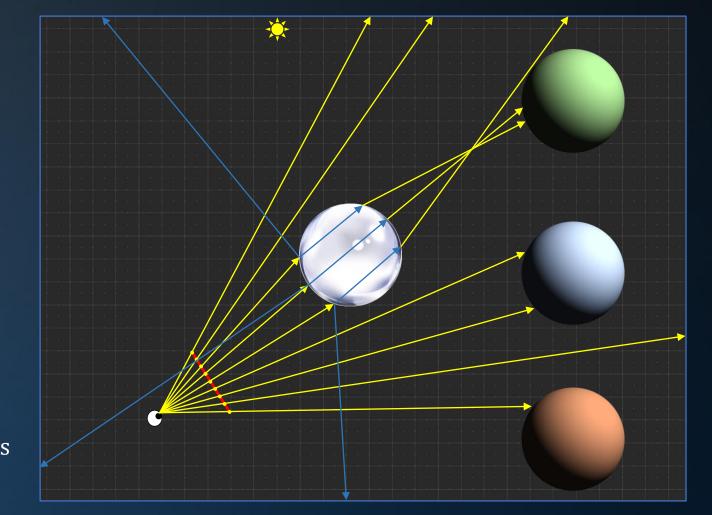
- Geometry
- Eye

at a = nt

AXDEPTH)

lf;

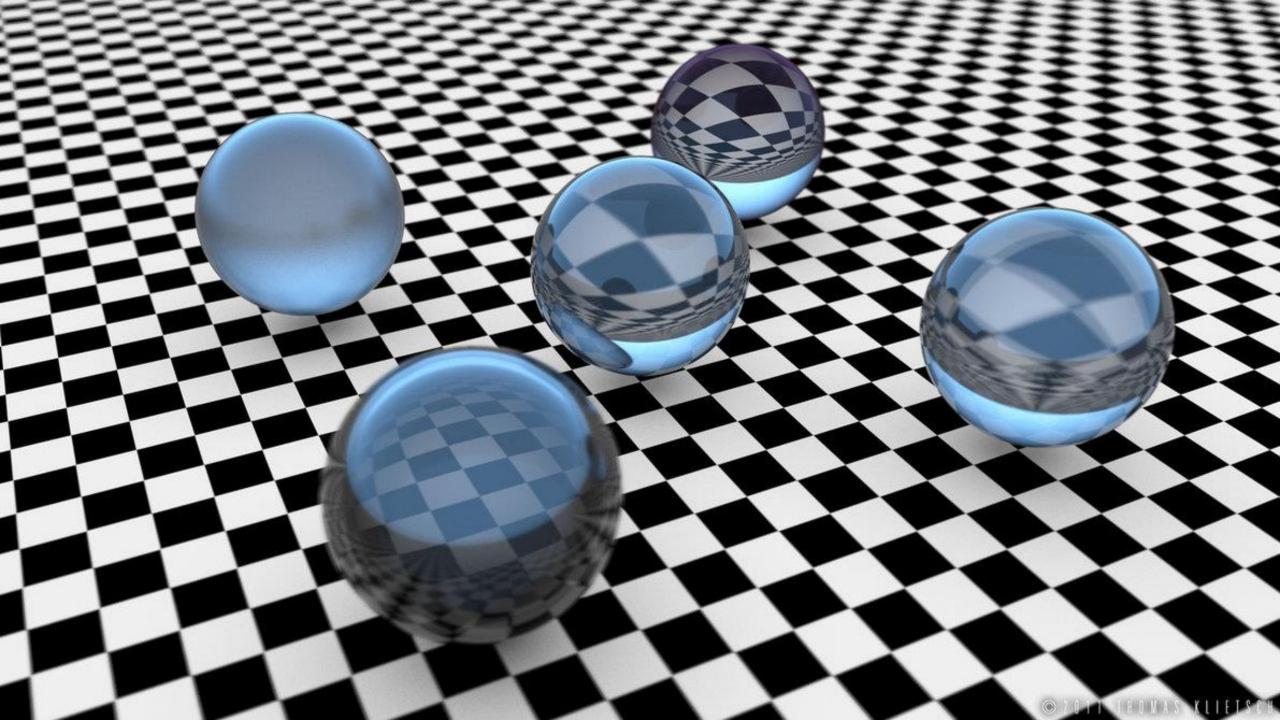
- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
- Shadow rays
- survive = SurvivalProbability Light transport radiance = SampleLight(&ran .x + radiance.y + radiance.z
- Extension rays t brdfPdf = EvaluateDiffuse(L at3 factor = diffuse * INVPI at weight = Mis2(directPdf, brdfPdf at cosThetaOut = dot(N, L) Light transport E * ((weight * cosThetaOut)
- /ive)
- at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, & urvive; pdf; 1 = E * brdf * (dot(N, R) / pdf); sion = true:





We are calculating light transport backwards.





ics k (depth < No) : = inside } 1

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

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

= * diffuse = true;

. efl + refr)) && (dep

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

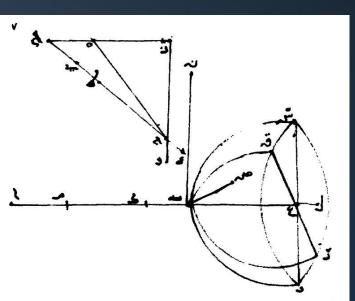
AXDEPTH)

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

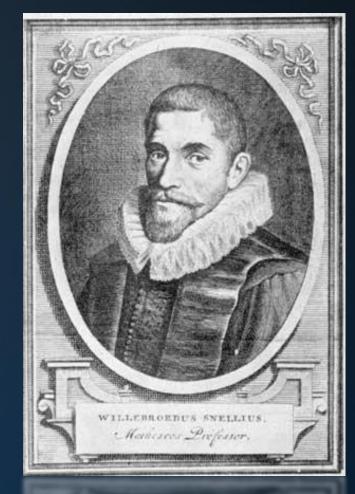
w = true; at brdfPdf = EvaluateD at3 factor = diffuse * at weight = Mis2(dire at cosThetaOut = dot(E * ((weight * cosThet

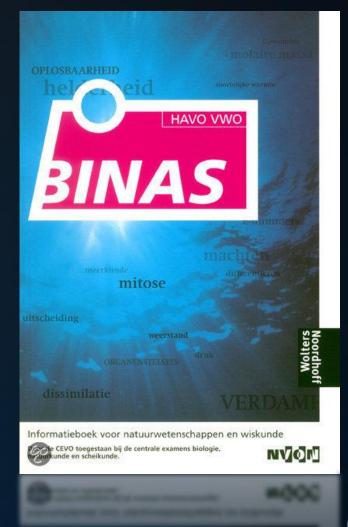
andom walk - done properly, **/ive)**

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



لاندان ماننده عليه اسط مستوغيره فلان هذا الشطح يقط سط بزص عنقطة ت فلابة مؤلز يقط احلحل ب ن م فلكن ذلك الخط مت والفصل المشترك بين فذا السطح وبين طح قط ق حط مستر فلات هذا السط يا ترضيط مسطى فتط ت غنط مستريط ترفيط قت د على تعلق مت وكذ كخط مستو ف المحال فلا يا ترضبط مسطى مناطقة مسطح مستو غير سطح مستو في من

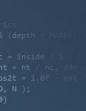






INFOGR – Lecture 5 – "Graphics Fundamentals"

Ray Tracing



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

= * diffus = true;

. :fl + refr)) &&

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

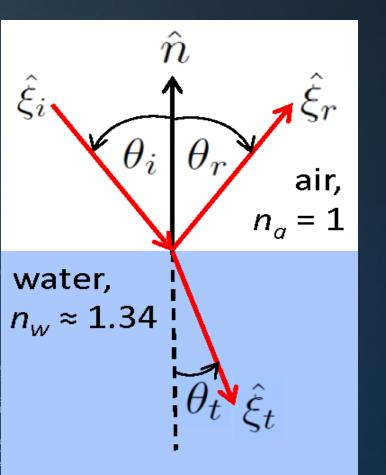
AXDEPTH)

survive = SurvivalProl
estimation - doing if
if;
radiance = SampleLight
e.x + radiance.y +

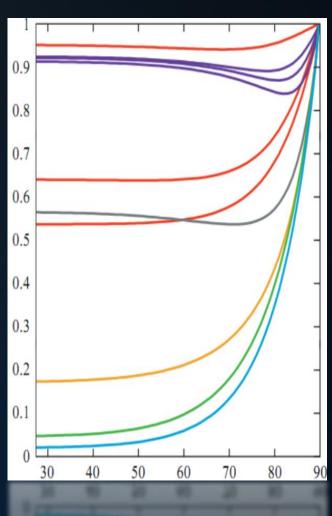
v = true; at brdfPdf = EvaluateD at3 factor = diffuse * at weight = Mis2(dire at cosThetaOut = dot(E * ((weight * cosThetaOut))

andom walk - done properly, closely following /ive)

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









Ray Tracing

sics & (depth < Monor

: = inside ? 1 ht = nt / nc, ddn ss2t = 1.0f - nnt ° n 2, N); ≫)

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

= * diffuse; = true;

efl + refr)) && (depth < NA

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

AXDEPTH)

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

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

andom walk - done properly, closely following /ive)

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

Physical basis

Ray tracing uses *ray optics* to simulate the behavior of light in a virtual environment.

It does so by finding light transport paths:

From the 'eye'

- Through a pixel
- Via scene surfaces
- To one or more light sources.

At each surface, the light is modulated. The final value is deposited at the pixel (simulating reception by a sensor).

T. Whitted, "An Improved Illumination Model for Shaded Display", ACM, 1980.



Intersections

nics & (depth < Monor

: = inside ? 1 ht = nt / nc, ddm os2t = 1.0f - nnt = nn D, N); 3)

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

= * diffuse; = true;

. :fl + refr)) && (depth < MAXDEP

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

AXDEPTH)

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

v = true;

at brdfPdf = EvaluateDiffuse(L, N) Psurdi 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)

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

Ray definition

A ray is an infinite line with a start point:

 $p(t) = 0 + t\vec{D}$, where t > 0.

struct Ray

};

float3 0; // ray origin
float3 D; // ray direction
float t; // distance

The ray direction is generally *normalized*.



sics & (depth < MaxDor

t = inside 7 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 - R0) Tr) R = (D = nnt - N - (dd)

= * diffuse; = true;

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

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) && co

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) (r

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:

Ray setup

A ray is initially shot through a pixel on the screen plane. The screen plane is defined in world space:

Camera position:E = (0,0,0)View direction: \vec{V} Screen center: $C = E + d\vec{V}$ Screen corners: $p_0 = C + (-1,1,0),$

From here:

- Change FOV by altering *d*;
 - Transform camera by multiplying \overline{E} , p_0 , p_1 , p_2 with the camera matrix.

 $= C + (1,1,0), p_2 = C + (-1,-1)$



Only if $\vec{V} = (0,0,1)$ of course.

-1,0)

sics & (depth < MANDOR

: = inside ? 1 1 1 0 ht = nt / nc, ddn 0 0 ps2t = 1.0f - nnt 0 0 2, N); ≥)

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

= * diffuse; = true;

efl + refr)) && (depth < NOCCEPTIO

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

AXDEPTH)

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

w = 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) = (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); sion = true:

Ray setup

Point on the screen:

 $p(u,v) = p_0 + u(p_1 - p_0) + v(p_2 - p_0), \ u, v \in [0,1)$

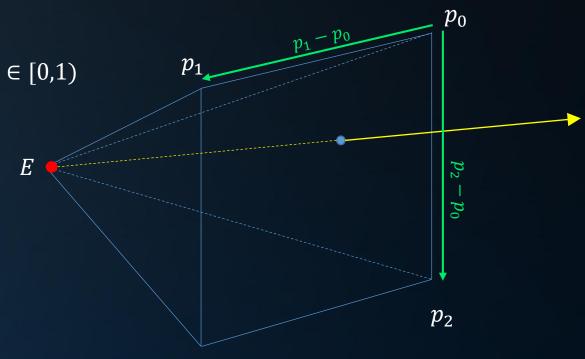
Ray direction (before normalization):

$$\vec{D} = p(u, v) - E$$

Ray origin:

O = E





tics & (depth < ™0000

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

= * diffuse; = true;

efl + refr)) && (depth < M0

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

AXDEPTH)

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

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

andom walk - done properly, closely followin /ive)

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

Ray intersection

Given a ray $p(t) = 0 + t\vec{D}$, we determine the smallest intersection distance t by intersecting the ray with each of the primitives in the scene.

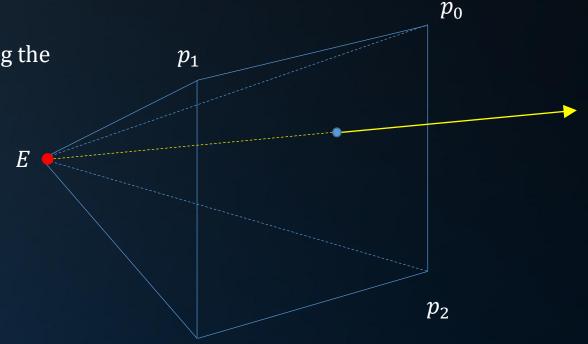
Ray / plane intersection:

Plane:
$$\mathbf{p} \cdot \vec{N} + d = 0$$

Ray: $p(t) = 0 + t\vec{D}$

Substituting for p(t), we get

 $\begin{pmatrix} 0 + t\vec{D} \end{pmatrix} \cdot \vec{N} + d = 0$ $t = -(0 \cdot \vec{N} + d) / (\vec{D} \cdot \vec{N})$ $P = 0 + t\vec{D}$





 $a = \vec{D} \cdot \vec{D}$

Intersect

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, 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 urvive; pdf;

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

Ray intersection

Ray / sphere intersection:

Sphere: $(p - C) \cdot (p - C) - r^2 = 0$ Ray: $p(t) = 0 + t\vec{D}$

Substituting for p(t), we get

$$(0 + t\vec{D} - C) \cdot (0 + t\vec{D} - C) - r^2 = 0 \vec{D} \cdot \vec{D} t^2 + 2\vec{D} \cdot (0 - C) t + (0 - C)^2 - r^2 = 0$$

$$ax^2 + bx + c = 0 \rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

 $b = 2\vec{D} \cdot (O - C)$ $c = (0 - C) \cdot (0 - C) - r^2$

Negative: no intersections

E

 p_1



 p_0

 p_2

ics & (depth < NAXDE-

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely follo /ive)

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

Note:

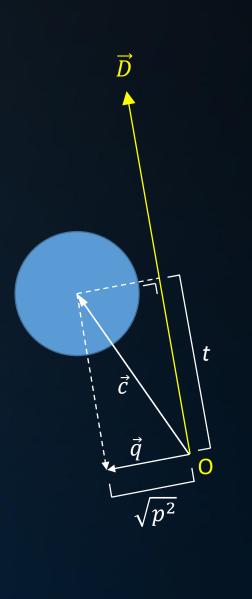
Ray Intersection

Efficient ray / sphere intersection:

void Sphere::IntersectSphere(Ray ray)

vec3 c = this.pos - ray.0; float t = dot(c, ray.D); vec3 q = c - t * ray.D; float p² = dot(q, q); if (p² > sphere.r²) return; t -= sqrt(sphere.r² - p²); if ((t < ray.t) && (t > 0)) ray.t = t; // or: ray.t = min(ray.t, max(0, t));

This only works for rays that start outside the sphere.





ics & (depth < POODEFT

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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:

Today's Agenda:

Rasters

- Colors
- Ray Tracing
- Assignment P2



Checkpoint

0, N);)) at a = nt - n

it Tr = 1 - (Tr) R = (D =

= true;

fl + refr)) && (o

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

AXDEPTH)

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

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

andom walk - done properly, closely follow /ive)

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



Math Basics

- vector != point
- planes, normals
- spheres
- dot product, cross product

Assignment P1

- template
- rgb colors in 32-bit
- coordinate systems in practice
- OpenGL in C#
- vertex buffers & shaders

Checkpoint 1: MIDTERM on May 16nd Checkpoint 2: P2 on May 28th



ics & (depth < Mox00000

: = inside ? 1 ht = nt / nc, ddn os2t = 1.0f - nmt - on 0, 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 < MAX

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

AXDEPTH)

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

v = true;

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

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:

Assignment 2: "Write a 2D ray tracer."

- Using the template
- Floating point pixels
- Pretty easy minimal specs
- Tons of extra challenges



https://ncase.me/sight-and-light



tics & (depth < NAXDS=1

at a = nt - nc, b = nt at Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N - (300

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

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

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

v = true;

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

at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

e ((weight cosmetadue) / directron) (h

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

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

Assignment 2: "Write a 2D ray tracer."

- Work in C#, C++, Java, ...
- Work alone or in pairs
 - Special challenges

Fastest ray tracer

Smallest ray tracer





fics & (depth < MoCOS

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

= * diffuse; = true;

. :fl + refr)) && (depth < MOXDEPT

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

AXDEPTH)

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

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

andom walk - done properly, closely /ive)

; at3 brdf = SampleDiffuse(diffuse, N, r urvive; pdf; a = F * brdf * (dot(N R) / odf);

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

www.youtube.com/watch?time_continue=37&v=iIIu7kPCN-8 medium.com/@jerry.ylilammi/making-of-newton-protocol-e9ccde41af30

ics & (depth < Motosini

= inside ? 1 1.00 ht = nt / nc, ddn os2t = 1.0f - nπt α 0, N); 3)

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

= * 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, &light e.x + radiance.y + radiance.z) > 0) && (dot)

v = true;

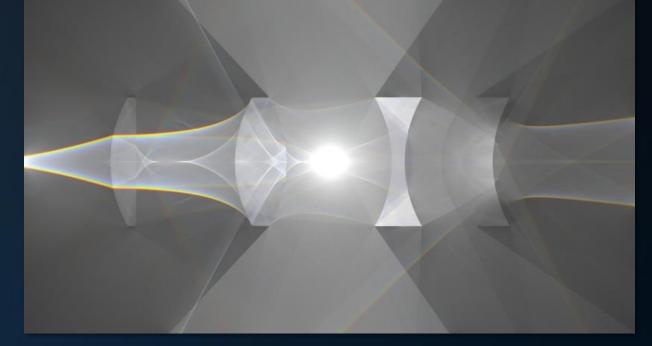
at brdfPdf = EvaluateDiffuse(L, N) Psuccisi at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * (Pad

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

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

Assignment 2: "Write a 2D ray tracer."

- Assignment now online
- Deadline: May 28.





ics & (depth < POODEFT

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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:

Today's Agenda:

Rasters

- Colors
- Ray Tracing
- Assignment P2



tics & (depth < NOCOD

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

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPIN

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

AXDEPTH)

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

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 Sec. /ive)

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

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2019

END OF lecture 5: "Graphics Fundamentals"

Next lecture: "More on Ray Tracing"

