"Le: k (depth < 100

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

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

= diffuse = true;

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

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

AXDEPTH)

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

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

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

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

INFOGR – Computer Graphics

Jacco Bikker - April-July 2016 - Lecture 2: "Graphics Fundamentals"

Welcome!



Synchronize



tice (depth < NAS

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

at a = nt - nc, b - nt - at Tr = 1 - (R0 + 11 - 77 Tr) R = (D * nnt - 77

= diffuse; = true;

efl + refr)) 絽 (depth (MAD)

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

AXDEPTH)

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

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

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

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

Today's Agenda:

- The Raster Display
- Vector Math
- Colors
- 2D Primitives
- 3D Primitives



Discretization

tic: € (depth < 10.

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

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

= diffuse = true:

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

AXDEPTH)

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

v = true;

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

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

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









Discretization

), N);

AXDEPTH)

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

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

E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely follow vive)

at3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, UR rvive; pdf; i = 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)



tine .

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

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

= diffuse; = true;

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

AXDEPTH)

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

v = true;

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

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

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

Rasterization



Improving rasterization:

1. Increase resolution;



tic: ≰ (depth < 10.55

- = inside / L it = nt / nc, ddo os2t = 1.8f - ont -2, N); 8)
- st a = nt nc, b = nt = ncst Tr = 1 - (R0 + (1 - R))Tr) R = (0 * nnt - R + 1)
- = diffuse; = true;
- D, N); ~efl * E * diffuse; = true;
- AXDEPTH)
- v = true;
- it brdfPdf = EvaluateDiffuse(L, N) * Paulo it3 factor = diffuse * INVPI; it weight = Mis2(directPdf, brdfPdf); it cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)
- andom walk done properly, closely following : /ive)
- ; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, Lord pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Rasterization



Improving rasterization:

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









Discretization

tic: ⊾(depth < N

= inside / 1
it = nt / nc, ddo
os2t = 1.0f - nnt
0, N();
3)

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

= diffuse = true:

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

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

AXDEPTH)

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

v = true;

it brdfPdf = EvaluateDiffuse(L, N.) Pare bits st3 factor = diffuse * INVPI; ot weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * Und

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

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

1									





CRT – Cathode Ray Tube

tice (depth (1995)

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

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

= diffuse = true;

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

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

AXDEPTH)

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

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

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

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

Electron beam zig-zagging over a fluorescent screen.



12

0,0

x=-1

Raster Displays

CRT – Cathode Ray Tube

y=1

0,0

v=-1

x=1

tica ≰ (depth ⊂ NASS

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

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

= diffuse; = true;

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

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

AXDEPTH)

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

v = true; at brdfPdf =

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

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

; pt3 brdf = SampleDiffuse(diffuse, N, F1, F2, UR, S 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...



tice ≰ (depth < 10.∞

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

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

= diffuse = true;

: efl + refr)) && (depth < HADDITT

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

AXDEPTH)

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

v = true;

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

sndom walk - done properly, closely following
vive)

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

Frame rate



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

tice ≰ (depth < P⊐.≂

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

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

E * diffuse; = true;

: :**fl + refr**)) && (depth < N

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it property if; radiance = SampleLight(&rand, I = 0.5 e.x + radiance.y + radiance.z) > 0.55

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

andom walk - done properly, closely following /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, loc prvive; 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 < Nov

= inside / :
it = nt / nc, ddo ss2t = 1.0f - not 0, N);
3)

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

E * diffuse; = true;

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

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

AXDEPTH)

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

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

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

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, bp3 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?



tice (depth < NAS

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

at a = nt - nc, b - nt - at Tr = 1 - (R0 + 11 - 77 Tr) R = (D * nnt - 77

= diffuse; = true;

efl + refr)) 絽 (depth (MAD)

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

AXDEPTH)

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

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

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

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

Today's Agenda:

- The Raster Display
- Vector Math
- Colors
- 2D Primitives
- 3D Primitives



2D space

tic: (depth (1935

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

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

= diffuse; = true;

.

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

AXDEPTH)

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

v = true;

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

sndom walk - done properly, closely following vive)

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

width-1

Conversion:

 $P_x = x * width$ $P_y = (1-y) * height$









Vectors

tic: ⊾ (depth < 140

: = inside / 1 it = nt / nc, dda os2t = 1.0f = nnt − D, N); B)

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

= diffuse; = true;

efl + refr)) && (depth k HANDIIII

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

AXDEPTH)

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

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

andom walk - done properly, closely following /ive)

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

In \mathbb{R}^d , a vector can be defined as an ordered *d*-tuple:







The Euclidean length or *magnitude* of a vector is calculated using:

$\|\vec{v}\| = \sqrt{v_1^2 + v_2^2 + \dots + vd^2}$

In 2D, this is similar to the Pythagorean theorem:

 $a^2 + b^2 = c^2$



Vectors

), N); efl * E * diffuse;

AXDEPTH)

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

in \mathbb{R}^3 : $\vec{v} = \begin{bmatrix} 0\\0\\0 \end{bmatrix}$ v = true; at brdfPdf = EvaluateDiffuse(L, N) st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely follow vive)

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

A *unit vector* is a vector with length = 1:

 $\|\vec{v}\| = 1$

A *null vector* is a vector with length = 0, e.g.:



0



A vector can be normalized by dividing it by its magnitude:

$$\vec{v}_{unit} = \frac{\vec{v}}{\|\vec{v}\|}$$

Can we normalize every vector?



ice

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

st a = nt - nc, b - nt - st Tr = 1 - (R0 + (1 - 1 fr) R = (D * nnt - N * 144

= diffuse; = true;

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

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

AXDEPTH)

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

v = true;

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

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

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

Vectors

A 2D vector (v_x, v_y) can be seen as the point v_x, v_y in the Cartesian plane.

A 2D vector (v_x, v_y) can be seen as an *offset* from the *origin*.



Note:

Positions and vectors in \mathbb{R}^3 can be both represented by 3-tuples (*x*, *y*, *z*), but they are not the same!



), N); efl * E * diffuse;

AXDEPTH)

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

v = true;

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

andom walk - done properly, closely follo vive)

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

Vectors

The sum of two vectors in \mathbb{R}^d ,

 $\vec{v} = (v_1, v_2, \dots, v_d)$ and $\vec{w} = (w_1, w_2, \dots, w_d)$

is defined as:

 $\vec{v} + \vec{w} =$

$$v_1 + w_1, v_2 + w_2, \dots, v_d + w_d$$



Example:

(4,1) + (1,2) = (5,3)

Vector subtraction is similarly defined.

Vector addition is *commutative* (as can be easily seen from the geometric interpretation):

$$(4,1) + (1,2) = (5,3) = (1,2) + (4,1).$$



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

AXDEPTH)

survive = SurvivalProbability(dif If; adiance = SampleLight(&rand, 1 .x + radiance.y + radiance.z)

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

andom walk - done properly, closely follo vive)

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

Vectors

The scalar multiple of a ddimensional vector \vec{v} is defined as:

 $\lambda \vec{v} = (\lambda v_1, \lambda v_2, \dots, \lambda v_d)$

Scalar multiplication can change the *length* of a vector.

It can also change the *direction* of the vector, which is reversed if $\lambda < 0$.



Two vectors \vec{v} and \vec{w} are parallel if one is a scalar multiple of the other, i.e.:

there is a λ such that $\vec{v} = \lambda \vec{w}$.



tice ≰ (depth < NAS

:= inside / i it = nt / nc, ddo os2t = 1.0f 0, N); 8)

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

= diffuse; = true;

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

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

AXDEPTH)

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

v = true;

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

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

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

Vectors

Parallel vectors are called *linearly dependent.*

If they are not parallel, vectors are *linearly independent*.

A special case is when two vectors are perpendicular to each other; in this case, each vector is a *normal vector* of the other.



In \mathbb{R}^2 , we can easily create a normal vector for (v_x, v_y) :

$$\vec{n} = (-v_y, v_x)$$

or

 $\vec{n} = (v_y, -v_x)$

Question: does this also work in \mathbb{R}^3 ?



Bases

tice (depth < 1925

: = inside / l it = nt / nc, ddo os2t = 1.0f - nn: o, N); 3)

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

= diffuse; = true;

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

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, N) = #2 factor = diffure = TANDT.

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

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

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, bp3 pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

We can use two linearly independent vectors to produce any vector:

 $\vec{a} = \lambda_1 \vec{u} + \lambda_2 \vec{v}$

This doesn't just work for perpendicular vectors.





Bases

tic: (depth < NAC

= inside / l ht = nt / nc, ddo os2t = 1.0f - nnt o, N); 3)

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

= * diffuse; = true;

: :f**l + refr)) && (depth** k HANDE

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

AXDEPTH)

survive = SurvivalProbability(diff estimation - doing it proper if; radiance = SampleLight(&rand, I . e.x + radiance.y + radiance.z) 000

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

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

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

We can use two linearly independent vectors to produce any vector:

 $\vec{a} = \lambda_1 \vec{u} + \lambda_2 \vec{v}$

This doesn't just work for perpendicular vectors.

Any pair of linearly independent vectors form a *2D basis*.

This extends naturally to higher dimensions.



Bases

tic: ≰(depth <)⊔...

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

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

E = diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

"Any pair of linearly independent vectors form a 2D basis":

The Cartesian coordinate system is an example of this.

In this case the vectors (1,0) and (0,1) form an *orthonormal* basis:



- 1. The vectors are perpendicular to each other;
- 2. The vectors are unit vectors.



tica ⊾ (depth < 155)

: = inside / l ht = nt / nc, dde os2t = 1.0f - nnt -D, N); B)

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

E * diffuse; = true;

efl + refr)) && (depth k HANDIII

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

AXDEPTH)

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

v = true;

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

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

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

Bases

A coordinate system can be *left handed* or *right handed*.

Note that this only affects the interpretation of the vectors; the vectors themselves are the same in each case.









tica ⊾ (depth ⊂ 100

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

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

E = diffuse; = true;

: :fl + refr)) && (depth is MAND

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

AXDEPTH)

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

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

andom walk - done properly, closely following /ive)

*: AKA inner product or scalar product
provive;
pdf;
n = E * brdf * (dot(N, R) / pdf);
sion = true;

Dot product

Given vectors \vec{a} , \vec{u} and \vec{v} , we know that:

 $\vec{a} = \lambda_1 \vec{u} + \lambda_2 \vec{v}$

We can determine λ_1 and λ_2 using the *dot product**.



The dot product of vector \vec{v} and \vec{w} is defined as:

$$\vec{v} \cdot \vec{w} = v_1 w_1 + v_2 w_2 + \dots + v_d w_d$$

or

 $\vec{v} \cdot \vec{w} = \sum_{i=0}^{d} v_i w_i$



tic: ⊾ (depth ∈ 100

= inside / 1 it = nt / nc, dde os2t = 1.0f - nnt 0, N); 8)

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

andom walk - done properly, closely following /ive)

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

Dot product

The dot product *projects* one vector on another.

If \vec{a} and \vec{u} are unit vectors, we can calculate the angle between them using the dot product:

 $\lambda = \cos \propto = \vec{u} \cdot \vec{a}$

or, if they are not normalized:

 $\cos \propto = \frac{\vec{u} \cdot \vec{a}}{\|\vec{u}\| \|\vec{a}\|}$



Projecting a vector on two linearly independent vectors yields a coordinate within the 2D basis.

This works regardless of the direction and scale of \vec{u} and \vec{v} , and also in \mathbb{R}^{3} .



 \vec{v}

2D Transforms

ile) k (depth is 1925

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

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

= diffuse; = true;

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

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

AXDEPTH)

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

w = true; ot brdfPdf = EvaluateDiffuse(L, N) * P ot3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

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

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

Cross product

The cross product can be used to calculate a vector perpendicular to a 2D basis formed by 2 vectors. It is defined as:

$$\times \vec{w} = \begin{pmatrix} v_2 w_3 - v_3 w_2 \\ v_3 w_1 - v_1 w_3 \\ v_1 w_2 - v_2 w_1 \end{pmatrix}$$

Note: The cross product is only defined in \mathbb{R}^3 .





INFOGR – Lecture 2 – "Graphics Fundamentals"

2D Transforms

ic: ⊾(depth ⊂ 12

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

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

= diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbabilit estimation - doing it prope if; radiance = SampleLight(%ran e.x + radiance.y + radiance.

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

andom walk - done properly, closely /ive)

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





2D Transforms

tic: ⊾(depth < NA

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

at a = nt - nc, b - nt at Tr = 1 - (80 + 11 Tr) R = (0 * nnt - 8 * 11

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

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





2D Transforms

tica ≰ (depth < 10.5

= inside / 1 it = nt / nc, ddo ss2t = 1.0f - not 5, N); 3)

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

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





tice (depth < NAS

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

at a = nt - nc, b - nt - at Tr = 1 - (R0 + 11 - 77 Tr) R = (D * nnt - 77

= diffuse; = true;

efl + refr)) 絽 (depth (MAD)

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

AXDEPTH)

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

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

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

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

Today's Agenda:

- The Raster Display
- Vector Math
- Colors
- 2D Primitives
- 3D Primitives



tic: ⊾ (depth < 10.

= inside / : it = nt / nc, dom 552t = 1.0f = not 3, N); 3)

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

= diffuse; = true;

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

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

AXDEPTH)

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

andom walk - done properly, closely following /ive)

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, brd 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).







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.

radiance = SampleLight(&rand, I, L), lie 2.x + radiance.y + radiance.z) > 0) [] []

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

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

; pt3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, 1, r pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:





Color representation

tic: k (depth < 100

t = inside / 1
it = nt / nc, d
ss2t = 1.0f
, N);
})

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

; t3 brdf = SampleDiffuse(diffuse, N, F1, F2, NR, Npf) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:



Color representation

tic: k (depth < 100

t = inside / 1
it = nt / nc, d
ss2t = 1.0f
, N);
})

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

; t3 brdf = SampleDiffuse(diffuse, N, F1, F2, NR, Npf) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:



INFOGR – Lecture 2 – "Graphics Fundamentals"

Colors

Color representation

tic: k (depth < 100

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

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

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



tic: k (depth < 100

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

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

= diffuse; = true;

efl + refr)) && (depth < NAAD

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

AXDEPTH)

survive = SurvivalProbabil estimation - doing it pro if; radiance = SampleLight(&ra e.x + radiance.y + radiance

v = true; ot brdfPdf = EvaluateDiffuse st3 factor = diffuse * INVPi st weight = Mis2(directPdf, st cosThetaOut = dot(N, L) E * ((weight * cosThetaOut)

andom walk - done properl vive)

; at3 brdf = SampleDiffuse(d rvive; pdf; o = 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.











tice € (depth < 10.5

: = inside / l it = nt / nc, dda os2t = 1.0f = nn: D, N); B)

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

= diffuse; = true;

efl + refr)) && (depth < HAADI

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

AXDEPTH)

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

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

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

; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, 48, 4); 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)



tic: ⊾ (depth < 10.

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

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

= diffuse; = true;

efl + refr)) && (depth k MANDIII

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

AXDEPTH)

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

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

; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, spin 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.



tice (depth < NAS

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

at a = nt - nc, b - nt - at Tr = 1 - (R0 + 11 - 77 Tr) R = (D * nnt - 77

= diffuse; = true;

efl + refr)) 絽 (depth (MAD)

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

AXDEPTH)

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

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

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

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

Today's Agenda:

- The Raster Display
- Vector Math
- Colors
- 2D Primitives
- 3D Primitives



tic: (depth (1935

: = inside / | nt = nt / nc, dda os2t = 1.0f = nnt 0, N); ∂)

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

= diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability(different estimation - doing it proper) ff; radiance = SampleLight(&rand, 1, 1, 1, 2.x + radiance.y + radiance.r) = 0

v = true; at brdfPdf = EvaluateDiffuse(L, N) at3 factor = diffuse = INVPI;

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

andom walk - done properly, closely foll /ive)

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

Recap

Vectors and their properties:

- Magnitude, direction
- Scalar product
- Null vector, normal
- Parallel, linear (in)depence
- Commutative addition & subtraction
- Dot product, cross product

Concepts:

- \mathbb{R}^d spaces
- (orthonormal) 2D basis, Cartesian
 - Left handed, right handed





tic: k (depth < 10.

: = inside / : ht = nt / nc, dde os2t = 1.8f - ont 0, N); 3)

at a = nt - nc, b - nt at Tr = 1 - (R0 + 11 Tr) R = (D * nnt - N *

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Implicit representation

Implicit curve:

f(x,y)=0

Function *f* maps two-dimensional points to a real value, i.e.

 $(x,y) \rightarrow f(x,y)$

The points for which this value is 0 are on the curve.

Example: circle

 $x^2 + y^2 - r^2 = 0$

If p = (x, y) is a point on the circle, and \vec{p} is a vector from the origin to p, it's length must be r, so $||\vec{p}|| = r$.



Example: circle with center c and radius r:

$$(x-c_x)^2 + (y-c_y)^2 - r^2 = 0$$



tic: ⊾(depth ∈ NA

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

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

= diffuse; = true;

-:fl + refr)) && (depth k H∧DIIII

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

AXDEPTH)

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

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

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

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

Implicit representation

Implicit curve:

f(x,y)=0

Function *f* maps two-dimensional points to a real value, i.e.

 $(x,y) \rightarrow f(x,y)$

The points for which this value is 0 are on the curve.

Example: line

Slope-intersect form:

y = ax + c

Implicit form:

-ax + y - c = 0

In general:

Ax + By + C = 0

$y = \frac{2}{3}x + 1$



 Δx



tic: € (depth < R

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

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

= diffuse; = true;

efl + refr)) && (depth < NACOT

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

AXDEPTH)

v = true;

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

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

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

Implicit line representation

Ax + By + C = 0

In this case:

 $A = -\frac{2}{3}$, B = 1, C = -1

The vector (A,B) is a *normal* of the line.



Slope-intersect form: y = ax + cImplicit form: -ax + y - c = 0General form: Ax + By + C = 0



tic: k (depth < P

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

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

= diffuse; = true;

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

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

AXDEPTH)

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

v = true;

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

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

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



Ax + By + C = 0

In this case:

 $\mathrm{A}=-rac{2}{3}$, $\mathrm{B}=1$, $\mathrm{C}=-1$

The vector (A,B) is a *normal* of the line.



We can use the normal to calculate the distance of a point to the line:

$$d = \vec{N} \cdot p + C$$

For $p = (3,3)$:
$$d = -\frac{2}{3} * 3 + 1 * 3 - 1$$
$$= -2 + 3 - 1 = 0$$

For p = (0,0): $d = -\frac{2}{3} * 0 + 1 * 0 - 1$ = -1 (?)



tic: K (depth < 12

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

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

= diffuse; = true;

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

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

AXDEPTH)

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

andom walk - done properly, closely following /ive)

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

Implicit line representation

Ax + By + C = 0

Given point p_1 and p_2 , we determine A, B and C as follows:

 $\vec{l} = p_2 - p_1$

 $\vec{N} = \left(-l_y, l_x\right)$

$$A=N_{\chi}$$
 , $B=N_{\chi}$, $C=-(ec{N}\cdot p_{1})$

It is convenient to normalize the normal: Only when $\|\vec{N}\| = 1$, |C| is the distance of the line to the origin.

(+)

Test with the line from the previous slides:

$$p_1 = (-3, -1)$$

 $p_2 = (3,3)$

p₂

$$\vec{l} = (6,4)$$

$$\vec{N} = (-4,6)$$

$$A = -4, B = 6$$

$$C = -((-4*-3) + (6*-1))$$

$$= -6$$

-4x + 6y - 6 = 0 $-\frac{2}{3}x + y - 1 = 0$



tic: ⊾ (depth < 100

at Tr = 1 - (80 + Tr) R = (D * nnt -

= diffuse = true;

-:**fl + refr))** 88 (depth < MADD

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it properly ff; radiance = SampleLight(&rand, I, I, I) e.x + radiance.y + radiance.z) > 0) #10

v = true;

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

andom walk - done properly, closely fi /ive)

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





"Le: k (depth < 100

: = inside / l it = nt / nc, dde os2t = 1.01 - nnt 0, N); 3)

st $a = nt - nc_{3} b - nt$ st Tr = 1 - (80 + (1))Tr) R = (0 + nt - n)

= diffuse = true;

: fl + refr)) 88 (depth < MADDI

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

AXDEPTH)

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

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

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

INFOGR – Computer Graphics

Jacco Bikker - April-July 2016 - Lecture 2: "Graphics Fundamentals"

END of "Graphics Fundamentals"

next lecture: "Ray Tracing (Introduction)"

