tics ⊾ (depth () ().)

= inside / 1 it = nt / nc, dde os2t = 1.01 - ... D, N); B)

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

= diffuse = true;

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

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

AXDEPTH)

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

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

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

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

INFOGR – Computer Graphics

Jacco Bikker - April-July 2015 - Lecture 3: "Geometry"

Welcome!



tic: € (depth < 1555

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

= diffuse; = true;

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

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

AXDEPTH)

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

v = true;

at brdfPdf = EvaluateDiffuse(L, N) Paur 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 a /ive)

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

Today's Agenda:

- 2D Primitives
- 3D Primitives
- Textures



2D Primitives

Recap

tic: K (depth ⊂ 1935

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

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

= diffuse; = true;

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

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

AXDEPTH)

v = true;

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

indom walk - done properly, closely fell
vive)

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

Last time: 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





2D Primitives

tic: k (depth < 100

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

st a = nt - nc, b - nt - st Tr = 1 - (R8 + (1 - - - -Tr) R = (D * nnt - N - - -

E = diffuse; = true;

-:fl + refr)) && (depth ⊂ NACOIOI

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pourst3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

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

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, bod 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:

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



2D Primitives

tica ⊾ (depth ∈ 100

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

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

= diffuse; = true;

-:fl + refr)) && (depth ⊂ NACOIOI

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

AXDEPTH)

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

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

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

; st3 brdf = SampleDiffuse(diffuse, N, r1, r2, UR, sr 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: line

Slope-intersect form:

y = ax + c

Implicit form:

-ax + y - c = 0

Ax + By + C = 0

In general:

 $\begin{array}{c} c \\ \Delta y \\ \Delta x \end{array}$

 $a = \frac{\Delta y}{\Delta x}$



2D Primitives

tic: ⊾(depth ⊂ N

: = inside / 1 it = nt / nc, dde os2t = 1.0f - nnt 0, 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 HAND)

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

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

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

Implicit line representation

Ax + By + C = 0

In this case:

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

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



Slope-intersect form: y = ax + c

Implicit form:

-ax + y - c = 0

General form:

Ax + By + C = 0



2D Primitives

tic: ⊾ (depth < R

: = inside / 1 ht = nt / nc, ddo os2t = 1.0f - nn: 0, 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 HAND)

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;

st brdfPdf = EvaluateDiffuse(L, N) Pause 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, L, H pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Implicit line representation

Ax + By + C = 0

In this case:

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

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:

$$d = N \cdot p + C$$

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 (?)



2D Primitives

tica ⊾ (depth ⊂ 100

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

= diffuse; = true;

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

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.r) = 0

w = 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, brd prvive; 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:

 $\vec{l} = p_2 - p_1$

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

$$A = N_x$$
 , $B = N_y$, $\mathcal{C} = -(ec{N} \cdot p_1)$

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



2D Primitives

tic: € (depth < 14.5

= inside / 1 it = nt / nc, dde os2t = 1.0f = ont 5, N); 8)

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

= diffuse; = true;

efl + refr)) && (depth is HARDITT

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

AXDEPTH)

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

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

E * ((weight * cosThetaOut) / directPdf) andom walk - done properly, closely follo

/ive)

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

Parametric representation

Parametric curve:

$$\binom{x}{y} = \binom{g(t)}{h(t)}$$

Example: line

$$p_0 = (x_{p0}, y_{p0}), p1 = (x_{p1}, y_{p1})$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{p0} \\ y_{p0} \end{pmatrix} + t \begin{pmatrix} x_{p1} - x_{p0} \\ y_{p1} - y_{p0} \end{pmatrix}$$

$$p(t) = p_0 + t(p_1 - p_0), t \in \mathbb{R}.$$

In this example:

 p_1 is the *support vector;* $p_1 - p_0$ is the *direction vector.*





2D Primitives

tic: ⊾(depth < 19

= inside / l nt = nt / nc, dde os2t = 1.8f - nnt 0, N); 3)

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

= diffuse; = true;

efl + refr)) && (depth < MADINI

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

AXDEPTH)

survive = SurvivalProbability(difference of the section - doing it properly for the section of the section of

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pour st3 factor = diffuse * INVPI; bt 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, Spr urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Slope-intercept:

y = ax + c

Implicit representation:

-ax + y - c = 0Ax + By + C = 0

Parametric representation:

$$(t) = p_0 + t(p_1 - p_0)$$





2D Primitives

tice ⊾ (depth ⊂ NAS

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

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

= diffuse = true;

-:fl + refr)) && (depth k HAO1000

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; tbrdfPdf = EvaluateDiffuse(L, N) * P at3 factor = diffuse * INVPI; at 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, SR, Soff pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Circle - parametric

$$\binom{x}{y} = \binom{x_c + r\cos\phi}{y_c + r\sin\phi}$$

opposite

y

espinet and the second

adjacent

 $\boldsymbol{\chi}$

Ф





SOH CAH TOA



tic: € (depth < 1555

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

= diffuse; = true;

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

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

AXDEPTH)

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

v = true;

at brdfPdf = EvaluateDiffuse(L, N) Paur 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 a /ive)

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

Today's Agenda:

- 2D Primitives
- 3D Primitives
- Textures



3D Primitives

tic: k (depth < 10

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

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

= diffuse; = true;

-**:fl + refr)) && (depth k HAAD**

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

AXDEPTH)

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

v = true; tbrdfPdf = EvaluateDiffuse(L, N,) = Purch 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, brain pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Circle – sphere (implicit)

Recall: the implicit representation for a circle with radius r and center c is:

 $(x - xc)^{2} + (y - yc)^{2} - r^{2} = 0$ or: $|| p - c ||^{2} - r^{2} = 0 \implies || p - c || = r$

In \mathbb{R}^3 , we get:

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

or: || p - c || = r





3D Primitives

tice (depth o No

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

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Line – plane (implicit)

Recall: the implicit representation for a line is:

Ax + By + C = 0

In \mathbb{R}^3 , we get a plane:

Ax + By + Cz + D = 0





3D Primitives

Parametric surfaces

st a = nt

), N); efl * E * diffuse;

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L. N

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

andom walk - done properly, closely f vive)

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

A parametric surface needs two parameters:

x = f(u, v),y = g(u, v),z = h(u, v).

For example, a sphere:

 $x = r \cos \phi \sin \theta,$ $y = r \sin \phi \sin \theta,$ $z = r \cos \theta$.

Doesn't look very convenient (compared to the implicit form), but it will prove useful for texture mapping.





3D Primitives

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

AXDEPTH)

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

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 fell vive)

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

Parametric planes

Recall the parametric line definition:

 $p(t) = p_0 + t(p_1 - p_0)$

For a plane, we need to parameters:

 $p(s,t) = p_0 + s(p_1 - p_0) + t(p_2 - p_0)$

or: $\overline{p(s,t)} = p_0 + s\vec{v} + t\vec{w}$

where:

- p_0 is a point on the plane;
- \vec{v} and \vec{w} are two linearly independent vectors on the plane;

s, $t \in \mathbb{R}$.







tic: € (depth < 1555

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

= diffuse; = true;

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

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

AXDEPTH)

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

v = true;

at brdfPdf = EvaluateDiffuse(L, N) Paur 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 a /ive)

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

Today's Agenda:

- 2D Primitives
- 3D Primitives
- Textures





Textures

tica ⊾ (depth ic NA)

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

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

= diffuse; = true;

efl + refr)) && (depth < HANDIII

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

AXDEPTH)

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

v = true;

at 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 following : /ive)

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

Back to the world of graphics...

 $\vec{1}$

Given a plane: y = 0 (i.e., with a normal vector (0,1,0)).

Two vectors on the plane define a basis:

Using these vectors, any point on the plane can be reached: We can now use λ_1 , λ_2 to define a color at P:

 \vec{u}

 $\vec{u} = (1,0,0) \text{ and } \vec{v} = (0,0,1).$ $P = \lambda_1 \vec{u} + \lambda_2 \vec{v}.$ $F(\lambda_1, \lambda_2) = \cdots.$

Textures

tic: ⊾ (depth < 19

= inside / l ht = nt / nc, ddo hs2t = 1.0f - nnt h; N); B)

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Example:

 $F(\lambda_1, \lambda_2) = \sin(\lambda_1)$

Another example:

 $F(\lambda_1, \lambda_2) = ((int)(2 * \lambda_1) + (int)\lambda_2) \& 1$



Textures

tica € (depth < 10.5

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

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

= diffuse; = true;

efl + refr)) && (depth k HAADII

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

AXDEPTH)

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

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

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

Other examples (not explained here):

Perlin noise Details: <u>http://www.noisemachine.com/talk1</u>

Voronoi / Worley noise Details: "A cellular texture basis function", S. Worley, 1996.







Textures

tic: k (depth < 100

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

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

= diffuse; = true;

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

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.r) = 0

v = true;

at brdfPdf = EvaluateDiffuse(L, N) *
st3 factor = diffuse * INVPI;
bt 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, st urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Obviously, not all textures can be generated procedurally.

For the generic case, we lookup the color value in a pixel buffer.

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} P \cdot \vec{u} \\ P \cdot \vec{v} \end{pmatrix} * \begin{pmatrix} T_{width} \\ T_{height} \end{pmatrix}$$

Note that we find the pixel to read based on *P*; we don't find a '*P*' for every pixel of the texture.



v v



Textures

tica k (depth < 10.5

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

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

E * diffuse; = true;

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

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

AXDEPTH)

v = true;

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

andom walk - done properly, closely followin vive)

t3 brdf = SampleDiffuse(diffuse, N, r1, r2, NR, Not urvive; pdf; v = E * brdf * (dot(N, R) / pdf); cien = true;

Retrieving a pixel from a texture:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} P \cdot \vec{u} \\ P \cdot \vec{v} \end{pmatrix} * \begin{pmatrix} T_{width} \\ T_{height} \end{pmatrix}$$

We don't want to read outside the texture. To prevent this, we have two options:

1. Clamping

2. Tiling

$$\binom{x}{y} = \begin{pmatrix} clamp(P \cdot \vec{u}, 0, 1) \\ clamp(P \cdot \vec{v}, 0, 1) \end{pmatrix} * \binom{T_{width}}{T_{height}}$$

 $\binom{x}{y} = \binom{frac(P \cdot \vec{u})}{frac(P \cdot \vec{v})} * \binom{T_{width}}{T_{height}}$

Tiling is efficiently achieved using a bitmask. This requires texture dimensions that are a power of 2.





Textures

Texture mapping: oversampling

ic: K (depth < 100

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

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

= diffuse; = true;

efl + refr)) && (depth & NADIII

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

AXDEPTH)

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

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

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

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





Textures

Texture mapping: undersampling

tic: **(depth** ⊂ Pas

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

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

= diffuse; = true:

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

AXDEPTH)

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

v = true;

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

andom walk - done properly, closely foll /ive)

; st3 brdf = SampleDiffuse(diffuse, N, F1, urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:







Textures

tic: k (depth (155)

= inside / 1 it = nt / nc, ddo -552t = 1:0f - nni -5, N); 8)

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

= diffuse; = true;

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

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

AXDEPTH)

v = true; t brdfPdf = EvaluateDiffuse(L, N.) Promise 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:

Fixing oversampling

Oversampling: reading the same pixel from a texture multiple times. Symptoms: blocky textures.

Remedy: bilinear interpolation: Instead of clamping the pixel location to the nearest pixel, we read from four pixels.

$$\begin{split} & w_{p1} \colon (1 - frac(x)) * (1 - frac(y)) \\ & w_{p2} \colon frac(x) * (1 - frac(y)) \\ & w_{p3} \colon (1 - frac(x)) * frac(y) \\ & w_{p4} \colon 1 - (wP_{1+} wP_{2+} wP_{3}) \end{split}$$





Textures

), N); efl * E * diffuse;

AXDEPTH)

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

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

vive)

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

Fixing oversampling







Textures

tic: K (depth < 192

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

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

= diffuse; = true;

efl + refr)) && (depth & MAXDIIII

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

AXDEPTH)

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

w = true; st brdPdf = 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, So urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Fixing undersampling

Undersampling: skipping pixels while reading from a texture. Symptoms: Moiré, flickering, noise.

Remedy: MIP-mapping.

The texture is reduced to 25% by averaging 2x2 pixels. This is repeated until a 1x1 image remains.

When undersampling occurs, we switch to the next MIP level.





NO MIPMAPS

WITH MIPMAPS

Textures

Trilinear interpolation: blending between MIP levels.

tic: k (depth < 12

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

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

= diffuse = true;

-: =**fl + refr))** && (death < NA

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

AXDEPTH)

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

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

andom walk - done properly, closely following vive)

, t3 brdf = SampleDiffuse(diffuse, N, r1, r2, R, 1997 pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:





tic: € (depth < 1555

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

= diffuse; = true;

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

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

AXDEPTH)

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

v = true;

at brdfPdf = EvaluateDiffuse(L, N) Paur 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 a /ive)

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

Today's Agenda:

- 2D Primitives
- 3D Primitives
- Textures



tic: ⊾ (depth (100)

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

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

E ⁼ diffuse = true;

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

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(L, N.) * Proceed st3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * 1999

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

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

INFOGR – Computer Graphics

Jacco Bikker - April-July 2015 - Lecture 3: "Geometry"

END of "Geometry"

next lecture: "3D Engine Fundamentals"

