INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2019

Lecture 6: "Ray Tracing"

Welcome!



tics & (depth < PACOD

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

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

= * diffuse = true;

efl + refr)) && (depth < MAXDEPT

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

AXDEPTH)

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

w = true; at brdfPdf = EvaluateDiffuse(L, N) Promotive 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: tics & (depth < Mot08=

: = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt 0, N); 8)

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTI

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

AXDEPTH)

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

v = true;

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

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

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

Today's Agenda:

- Ray Tracing Recap
- Shading
- Textures









Ray Tracing

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, 8R, 8pdf urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:







5

tics & (depth < Mot08=

: = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt 0, N); 8)

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTI

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

AXDEPTH)

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

v = true;

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

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

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

Today's Agenda:

- Ray Tracing Recap
- Shading
- Textures









tics & (depth < Moon

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

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

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) * (rad

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

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





ics & (depth < ∧ooccit

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

= * diffuse; = true;

efl + refr)) && (depth < NOXDEPTH

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

AXDEPTH)

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

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) = (r8

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:

The End

We used *primary rays* to find the *primary intersection* point.

Determining light transport:

- Sum illumination from all light sources
- ...If they are *visible*.

We used a primary ray to find the object visible through a pixel: Now we will use a *shadow ray* to determine visibility of a light source.



AXDEPTH)

survive = SurvivalProbability(diff) radiance = SampleLight(&rand, I, &L, e.x + radiance.y + radiance.z) > 0) 8

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

andom walk - done properly, closely follow /ive)

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

Shadow Ray

Constructing the shadow ray:

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

Ray origin: the primary intersection point *I*.

Ray direction: $P_{light} - I$ (normalized)

Restrictions on *t*: $0 < t < ||P_{light} - I||$





tics & (depth < NOCCCT)

: = inside ? 1 1 1 7 ht = nt / nc, ddn 4 7 bs2t = 1.0f - nnt 7 7 D, N); &)

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

= * diffuse; = true;

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

), 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) &&

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) / <u>directPdf</u>

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

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

Shadow Ray

Equally valid:

Direction of the shadow ray: $\frac{P}{\parallel P}$

 $\frac{P_{light}-I}{\|P_{light}-I\|}$

 $\frac{I - P_{light}}{\|P_{light} - I\|} \text{ Or } \frac{I - P_{light}}{\|I - P_{light}\|}$

Note that we get different intersection points depending on the direction of the shadow ray.

It doesn't matter: the shadow ray is used to determine *if* there is an occluder, not *where*.

This has two consequences:

- 1. We need a dedicated shadow ray query;
- 2. Shadow ray queries are (on average) twice as fast. (why?)





ics & (depth < PACOSET

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Shadow Ray

"In theory, theory and practice are the same. In practice, they are not."

Problem 1:

Our shadow ray queries report intersections at $t = \sim 0$. Why?

Cause: the shadow ray sometimes finds the surface it originated from as an occluder, resulting in *shadow acne*.

Fix: offset the origin by 'epsilon' times the shadow ray direction.

Note: don't forget to reduce t_{max} by epsilon.



ics & (depth < PACCO

: = inside ? 1 1 1 1 ht = nt / nc, ddm bs2t = 1.0f - nmt 2, N); 2)

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

Shadow Ray

"In theory, theory and practice are the same. In practice, they are not."

Problem 2:

Our shadow ray queries report intersections at $t = t_{max}$. Why?

Cause: when firing shadow rays from the light source, they may find the surface that we are trying to shade.

Fix: reduce t_{max} by 2 * *epsilon*.



ics & (depth < MAXDS⊡

: = inside 7 1 1 1 0 ht = nt / nc, ddn 0 0 ss2t = 1.0f - nnt 0 n 2, N); ≫)

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPID

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

AXDEPTH)

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

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) (rac

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:

Shadow Ray

"The most expensive shadow rays are those that do not find an intersection."

Why?

(because those rays tested every primitive before concluding that there was no occlusion)



ics & (depth < >>>0000000

: = inside ? 1 a 1.3 ht = nt / nc, ddn = 3 s2t = 1.0f - nnt ⊂ n 2, N); 8)

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

= * diffuse; = true;

efl + refr)) && (depth < MANDE

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

AXDEPTH)

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

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) * (reference);

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:

Transport

The amount of energy travelling from the light via the surface point to the eye depends on:

- The brightness of the light source
- The distance of the light source to the surface point
- Absorption at the surface point
- The angle of incidence of the light energy



nics & (depth < NOCCO

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

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEPTI

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

AXDEPTH)

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

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:

Transport

Brightness of the light source:

Expressed in *watt (W)*, or *joule per second (J/s or Js*⁻¹).

Energy is transported by photons.

Photon energy depends on wavelength; energy for a 'yellow' photon is $\sim 3.31 \cdot 10^{-19}$ J.

A 100W light bulb thus emits $\sim 3.0 \cdot 10^{21}$ photons per second.



ics & (depth < MoxOC)

t = inside } 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ∩ D, N); D)

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

= * diffuse; = true;

• efl + refr)) && (depth < MODEPT

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(

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

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

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

Transport

Energy at distance *r*:

For a point light, a brief pulse of light energy spreads out as a growing sphere. The energy is distributed over the surface of this sphere.

Energy per unit area is therefore proportional to the inverse area of the sphere at distance r, i.e.:

$$E/m^2 = E_{light} \frac{1}{4\pi r^2}$$

Light energy thus dissipates at a rate of $\frac{1}{r^2}$. This is referred to as *distance attenuation*.



at a = ni

), N); refl * E * diffuse;

AXDEPTH)

survive = SurvivalProbability(lf: radiance = SampleLight(&rand e.x + radiance.y + radiance.z

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

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

at3 brdf = SampleDiffuse(diffuse, N, r1, r2, 8R, 1 urvive; pdf; 1 = E * brdf * (dot(N, R) / pdf); sion = true

Transport

Absorption:

Most materials absorb light energy. The wavelengths that are not fully absorbed define the 'color' of a material.

The reflected light is thus:

 $E_{reflected} = E_{incoming} \circ C_{material}$

Note that $C_{material}$ cannot exceed 1; the reflected light is never *more* than the incoming light.

C_{material} is typically a vector: we store r, g, b for all light transport.

 $A \circ B = \begin{vmatrix} A_x B_x \\ A_y C_y \\ A_z C_z \end{vmatrix}$ ('entrywise product')



ics & (depth < MAXDON

: = inside ? 1 + 1.0 ht = nt / nc, ddn bs2t = 1.0f - nnt 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 < MAXDI

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly, closed Hf; radiance = SampleLight(&rand, I, &L, &LL 2.x + radiance.y + radiance.z) (20)

w = true; at brdfPdf = EvaluateDiffuse(L, N) Psurvey at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) (Page 1);

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

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

Transport

Energy arriving at an angle:

A small bundle of light arriving at a surface affects a larger area than the cross-sectional area of the bundle.

Per m^2 , the surface thus receives less energy. The remaining energy is proportional to:

 $\cos \alpha$ or: $\vec{N} \cdot \vec{L}$.



sics & (depth < Mocco

: = 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 < MAXDEPIN

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

AXDEPTH)

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

w = true; at brdfPdf = EvaluateDiffuse(L, N) * Psurvive at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * (P8

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:

Transport

All factors:

- Emitted light : defined as RGB color, floating point
- Distance attenuation: $\frac{1}{r^2}$
- Absorption, modulate by material color
- N dot L





tics & (depth < Notes

t = inside 7 1 1 1 0 nt = nt / nc, ddn os2t = 1.0f - n⊓t ∩ n 0, N); 3)

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

= * diffuse = true;

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

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

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

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







tics & (depth < Mot08=

: = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt 0, N); 8)

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTI

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

AXDEPTH)

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

v = true;

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

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

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

Today's Agenda:

- Ray Tracing Recap
- Shading
- Textures









ics & (depth < ⊅000000

: = inside ? 1 ht = nt / nc, ddn ss2t = 1.0f - n⊓t 2, N); 3)

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

= * diffuse; = true;

efl + refr)) && (depth < NODEPTI

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly, close If; radiance = SampleLight(&rand, I, 81, 81) 2.x + radiance.y + radiance.z) > 0) 88

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 following SOU. /ive)

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

Texturing a Plane

 $\vec{1}$

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:

 \vec{u}

• P

We can now use λ_1 , λ_2 to define a color at P:

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



ics & (depth < ⊅0000

: = inside ? l |]] ht = nt / nc, ddn |] bs2t = 1.0f - nmt | n D, 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 < MODEPTH

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

AXDEPTH)

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

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

Example:

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

Another example:

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

sics & (depth < NOCCO

c = inside ? 1 = 5 3 ht = nt / nc, ddn = 5 552t = 1.0f - nmt = 5 2, N); 3)

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

= * diffuse; = true;

efl + refr)) && (depth < MODEPT

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

AXDEPTH)

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

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

andom walk - done properly, closely follo /ive)

, H33 brdf = SampleDiffuse(diffuse, N, r1, r2, dR, redu urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Other examples (not explained here):

Perlin noise

Details: http://www.noisemachine.com/talk1

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











ics & (depth < Model

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

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

= * diffuse; = true;

efl + refr)) && (depth < NODE

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly if; radiance = SampleLight(&rand, I, &L, &Le 2.x + radiance.y + radiance.z) >_0) &&

v = true;

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

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

it3 I

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

Obviously, not all textures can be generated procedurally.

Finding P using basis vectors \vec{u} , \vec{v} and parameters λ_1 , λ_2 :

$$\begin{pmatrix} P_{\chi} \\ P_{y} \end{pmatrix} = \lambda_{1}\vec{u} + \lambda_{2}\vec{v} = \begin{pmatrix} \lambda_{1}\vec{u}_{\chi} + \lambda_{2}\vec{v}_{\chi} \\ \lambda_{1}\vec{u}_{y} + \lambda_{2}\vec{v}_{y} \end{pmatrix}$$

...But what about the opposite, i.e. can we find λ_1 , λ_2 given P?







tics & (depth < MADDOT

= inside 7 1 g L . ht = nt / nc, ddn os2t = 1.0f - n∺t ° 0, N); 8)

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

= * diffuse = true;

• •fl + refr)) && (depth < MAXDED

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

AXDEPTH)

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

v = true;

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

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

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

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.





nics & (depth < ⊅00000

: = inside ? 1 1 1 1 ht = nt / nc, ddn 1 552t = 1.0f - nnt " n 2, N); 3)

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

= * diffuse; = true;

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

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

AXDEPTH)

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

v = true;

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

E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely following /ive)

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

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

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

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





28

Texture mapping: oversampling

20120

17

tics ≹ (depth < POCOS

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

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

= * diffuse; = true;

. efl + refr)) && (depth ≪ MAXDED⊺

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

AXDEPTH)

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

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





Texture mapping: undersampling

hics & (depth < NOCCS−1

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

= * diffuse; = true:

-•fl + refr)) && (denth < MAXIE

), N); •efl * E * diffuse; = true;

AXDEPTH)

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

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

v = true;

sion = true:

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

andom walk - done properly, closely foll /ive)

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







ics & (depth < NOCCO

at a = nt - nc, b = nt + nc at Tr = 1 - (R0 + (1 - R0 Tr) R = (D = nnt - N + (d0)

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

; at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &pdf 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.

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





AXDEPTH)

survive = SurvivalProbability(diffu radiance = SampleLight(&rand, I, & $\mathbf{x} + \mathbf{radiance.y} + \mathbf{radiance.z}) > 0$

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

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

/ive)

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

Fixing oversampling







tics & (depth < Mot000

t = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt ⊂o 2, N); ≷)

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly, closed ff; radiance = SampleLight(&rand, I, &L, &II e.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) (Psi

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:

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

Trilinear interpolation: blending between MIP levels.

tics & (depth < PVCDEF

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

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

= * diffuse = true;

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

AXDEPTH)

survive = SurvivalProbability(diffu estimation - doing it properly, clo If; radiance = SampleLight(&rand, I, &L e.x + radiance.y + radiance.z) > 0) 8

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

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

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





tics & (depth < Mot08=

: = inside ? 1 ht = nt / nc, ddn bs2t = 1.0f - nnt 0, N); 8)

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

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTI

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

AXDEPTH)

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

v = true;

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

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

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

Today's Agenda:

- Ray Tracing Recap
- Shading
- Textures









sics & (depth < MODE)

c = inside } 1 ht = nt / nc, ddn bs2t = 1.0f - nnt D, N); 2)

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

= * diffuse; = true;

efl + refr)) && (depth < MOXDEPTI

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly close If; radiance = SampleLight(&rand, I, &L, &L) 2.x + radiance.y + radiance.z) > 0<u>.88</u>

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) = 1000

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

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

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2019

END OF Lecture 6: "Ray Tracing"

Next lecture: "Ray Tracing (2)"

