

tics & (depth < PV00500

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

at a = nt - nc, b = nt at Tr = 1 - (R0 + (1 - R0) Tr) R = (D = nnt - N - (10)

= * diffuse; = true;

efl + refr)) && (depth < MAXDEPTI

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

AXDEPTH)

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

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

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

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

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2019

Lecture 7: "Ray Tracing (2)"

Welcome!



ics & (depth < Modern

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

= * diffuse = true;

• •fl + refr)) && (depth < MAXDEPILL

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

AXDEPTH)

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

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

Today's Agenda:

- Recap
- Normals
- Reflections
- Recursion
- Shading models
- TODO



Recap

tics & (depth < ™0000

: = inside ? 1 ht = nt / nc, ddn os2t = 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;

• •fl + refr)) && (depth < MAXDE

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

AXDEPTH)

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

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







INFOGR – Lecture 7 – "Ray Tracing (2)"

Recap

AXDEPTH)

v = true;

/ive)

urvive; pdf;

sion = true

lf;

survive = SurvivalProbability(diff

radiance = SampleLight(&rand, I, e.x + radiance.y + radianc<u>e.z) > 0</u>

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 fol

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

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



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.

$A \circ B = \begin{bmatrix} A_x B_x \\ A_y C_y \\ A_z C_z \end{bmatrix} \tag{6}$

('entrywise product')

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*.

Reflected light:

$$E_{to_eye} = \sum_{i=1}^{\#lights} \frac{(N \cdot L)}{dist^2} \circ E_{light} \circ c_{material}$$



Recap

ics & (depth < Modoon

: = inside ? 1 ()) ht = nt / nc, ddn bs2t = 1.0f - n⊓t ~ (0, 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 < №

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

AXDEPTH)

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

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

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

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

Efficient ray / sphere intersection:

void Sphere::IntersectSphere(Ray ray)

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

Note:

This only works for rays that start outside the sphere.





INFOGR – Lecture 7 – "Ray Tracing (2)"

Recap

hics & (depth < Maco

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

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

= * diffuse = true:

efl + refr)) && (depth < NACED

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

AXDEPTH)

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

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







ics & (depth < Modern

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

= * diffuse = true;

• •fl + refr)) && (depth < MAXDEPILL

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

AXDEPTH)

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

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

Today's Agenda:

- Recap
- Normals
- Reflections
- Recursion
- Shading models
- TODO



sics & (depth < Modern

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

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

= * diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly close If; adiance = SampleLight(&rand, I, &L, & 2.x + radiance.y + radiance.z) > 0) 38

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

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

; at3 brdf = SampleDiffuse(diffuse, N, r1, r2, &R, &pdf) urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true: Fun fact: $\begin{pmatrix} A \\ B \\ C \end{pmatrix}$ *is* the normal.

 \vec{N}

We Need a Normal

For a plane, we already have the normal.

$$Ax + By + Cz + D = 0$$
 or $(P \cdot \vec{N}) + D = 0$

A plane is the set of points that are at distance 0 from the plane.
Distance increases when we move away from the plane.
We move away from the plane by moving in the direction of the normal.

Distance attenuation: $1/r^2$

Angle of incidence: $N \cdot L$



sics & (depth < ≫xcco

: = inside ? 1 () . ht = nt / nc, ddn os2t = 1.0f - n⊓t 0, N); ∂)

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

= * diffuse; = true;

efl + refr)) && (depth < MADD

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

AXDEPTH)

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

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) (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:

We Need a Normal

Question:

How does light intensity relate to scene size? i.e.: if I scale my scene by a factor 2, what should I do to my lights?

 \rightarrow Distance attenuation requires scaling light intensity by 2^2

→ Scene scale does not affect $N \cdot L$.



sics & (depth < Motos)

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

We Need a Normal

Question:

What happens when a light is near the horizon?

→ Angle approaches 90°; cos α approaches 0 (and so does $\vec{N} \cdot \vec{L}$)

→ Light is distributed over an infinitely large surface (so, per unit area it becomes 0)

Note: below the horizon, $\cos \alpha$ becomes negative. \rightarrow Clamp $\vec{N} \cdot \vec{L}$ to zero.





nics & (depth < NADDO

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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) (read);

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:

We Need a Normal

Normals are also used to *prevent* shadow rays.

Situation:

A light source is behind the surface we hit with the primary ray:

 $\vec{N} \cdot \vec{L} < 0$

In this case, visibility is 0, and we do not cast the shadow ray.



tics & (depth < →∞000

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

at a = nt - nc, b = 11 at Tr = 1 - (R0 + (1 - 10 Ir) R = (D = nnt - N

= * diffuse = true;

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

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

AXDEPTH)

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

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 Sec. /ive)

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

We Need a Normal

Normals for spheres:

The normal for a sphere at a point *P* on the sphere is parallel to the vector from the center of the sphere to *P*.

$$\vec{N}_P = \frac{P-C}{||P-C||}$$
 or (cheaper): $\vec{N}_P = \frac{P-C}{r}$





sics & (depth < MAXDE

: = inside ? 1 1 ... nt = nt / nc, ddm os2t = 1.0f - nmt m D, N); 3)

at a = nt - nc, b = nt + nc at Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N + 600

= * diffuse = true;

efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

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

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

We Need a Normal

Normals for spheres:

When a sphere is hit from the inside, we need to *reverse* the normal.

 $\vec{N}_P = \frac{C - P}{||P - C||}$

To detect this situation:

1. Calculate the normal in the usual manner (P - C); 2. If $\vec{N}_P \cdot \vec{D}_{ray} > 0$ then $\vec{N}_P = -\vec{N}_P$.



 \vec{N}_{P}

tics & (depth < NOCCS)

: = inside ? 1 1 1 0 ht = nt / nc, ddn 0 0 ss2t = 1.0f - n⊓t 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 < MODEP

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

AXDEPTH)

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

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

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

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

Normal Interpolation

Simulating smooth surfaces using normal interpolation:

1. Generate *vertex normals*.

A vertex normal is calculated by averaging the normals of the triangles connected to the vertex and normalizing the result.

2. Interpolate the normals over the triangle.

In a ray tracer, use barycentric coordinates to do this. Normalize the interpolated normal.





ics & (depth < Moder

: = inside ? 1 | 1 | 1 ht = nt / nc, ddm | ps2t = 1.0f - nmt | r 2, N); 3)

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

= * diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbabil estimation - doing it pro Hf; radiance = Sampl e.x + radiance.y

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

andom walk - do **/ive)**

Normal Interpolation

Using the interpolated normal:

- Use the interpolated normal in the $\vec{N} \cdot \vec{L}$ calculation.
- Use the original face normal when checking if a light is visible.



ics & (depth < Modern

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

= * diffuse = true;

• •fl + refr)) && (depth < MAXDEPILL

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

AXDEPTH)

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

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

Today's Agenda:

- Recap
- Normals
- Reflections
- Recursion
- Shading models
- TODO



hics & (depth < Moder

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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





nics & (depth < Maccor

c = inside ? 1 1 1 1
nt = nt / nc, ddn 4
ps2t = 1.0f - nnt 4
p, N);

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDERIU

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

AXDEPTH)

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

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 Sec /ive)

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

Light Transport

We introduce a *pure specular* object in the scene.

Based on the normal at the primary intersection point, we calculate a new direction. We follow the path using a *secondary ray*.

At the primary intersection point, we 'see' what the secondary ray 'sees'; i.e. the secondary ray behaves like a primary ray.

We still need a shadow ray at the new intersection point to establish light transport.





nics & (depth < NACCO

c = inside ? 1 1 1 2 nt = nt / nc, ddn 4 4 os2t = 1.0f - nnt 5 0, 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 < MAXDEP

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

AXDEPTH)

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

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

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

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

Light Transport

For a pure specular reflection, the energy from a single direction is reflected into a single outgoing direction.

- We do not apply $\vec{N} \cdot \vec{L}$
- We <u>do</u> apply absorption

Since the reflection ray requires the same functionality as a primary ray, it helps to implement this recursively.

vec3 Trace(Ray ray)

I, N, material = scene.GetIntersection(ray);
if (material.isMirror)
 return material.color * Trace(...);
return DirectIllumination() * material.color;





nics & (depth < MADS

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

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

= * diffuse; = true;

efl + refr)) && (depth < MODEPTI

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

AXDEPTH)

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

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

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

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

Light Transport

For pure specular reflections we do not cast a shadow ray.

Reason:

Light arriving from that direction cannot leave in the direction of the camera.



sics & (depth < NOCCS

nt = nt / nc, dd ns2t = 1.0f - nm), N);))

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

= * diffuse; = true;

. :fl + refr)) && (depth

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

AXDEPTH)

survive = SurvivalProbab estimation - doing it p if; radiance = SampleLight(e.x + radiance.y + radia

w = true; at brdfPdf = EvaluateDif at3 factor = diffuse * I at weight = Mis2(direct at cosThetaOut = dot(N, E * ((weight * cosThetaOut) / direct

andom walk - done properly, closely fol /ive)

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





sics & (depth < MARCET

: = inside ? 1 1 1 1 ht = nt / nc, ddn bs2t = 1.0f - nmt ? r 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 < MAXDEF

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

AXDEPTH)

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

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

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

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

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

Partially Reflective Surfaces

We can combine pure specularity and diffuse properties.

Situation: our material is only 50% reflective.

In this case, we send out the reflected ray, and multiply its yield by 0.5. We also send out a shadow ray to get direct illumination, and multiply the received light by 0.5.







nics & (depth < MACOST

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Reflecting a HDR Sky

A dark object can be quite bright when reflecting something bright.

E.g., a bowling ball, pure specular, color = (0.01, 0.01, 0.01); reflecting a 'sun' stored in a HDR skydome, color = (100, 100, 100).

For a collection of HDR probes, visit Paul Debevec's page:

http://www.pauldebevec.com/Probes





ics & (depth < Modern

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

= * diffuse = true;

• •fl + refr)) && (depth < MAXDEPILL

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

AXDEPTH)

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

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

Today's Agenda:

- Recap
- Normals
- Reflections
- Recursion
- Shading models
- TODO



tics & (depth < Modoor

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

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEPT

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

AXDEPTH)

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

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

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

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

Whitted-style Ray Tracing, Pseudocode

Color Trace(vec3 0, vec3 D)

I, N, mat = IntersectScene(0, D);
if (!I) return BLACK;
return DirectIllumination(I, N) * mat.diffuseColor;

Todo:

Implement IntersectSceneImplement IsVisible

Color DirectIllumination(vec3 I, vec3 N)

```
vec3 L = lightPos - I;
float dist = length( L );
L *= (1.0f / dist);
if (!IsVisibile( I, L, dist )) return BLACK;
float attenuation = 1 / (dist * dist);
return lightColor * dot( N, L ) * attenuation;
```



ics & (depth < Modes)

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

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

= * diffuse; = true;

efl + refr)) && (depth < MANDEPTH

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

AXDEPTH)

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

v = true;

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

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

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

Whitted-style Ray Tracing, Pseudocode

Color Trace(vec3 0, vec3 D)

I, N, mat = IntersectScene(0, D);

if (!I) return BLACK;

if (mat.isMirror())

```
return Trace( I, reflect( D, N ) ) * mat.diffuseColor;
```

else

return DirectIllumination(I, N) * mat.diffuseColor;

Todo:

Handle partially reflective surfaces.



ics & (depth < Maxim

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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)

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

Whitted-style Ray Tracing, Pseudocode

```
Color Trace( vec3 0, vec3 D )
```

```
I, N, mat = IntersectScene( 0, D );
```

if (!I) return BLACK;

if (mat.isMirror())

```
Todo:
```

- Implement reflect
- Implement refract
- Implement Fresnel
- Cap recursion

```
return Trace( I, reflect( D, N ) ) * mat.diffuseColor;
```

```
else if (mat.IsDielectric())
```

```
f = Fresnel( ... );
return (f * Trace( I, reflect( D, N ) )
    + (1-f) * Trace( I, refract( D, N, ... ) ) ) * mat.DiffuseColor;
```

else

return DirectIllumination(I, N) * mat.diffuseColor;



Spheres: pure specular

ics ≰(depth < NOCCO

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

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

= * diffuse; = true;

efl + refr)) && (depth < MODEPTI

), 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) && (000)

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)

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

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





Spheres: 50% specular

ics & (depth < MODELL

: = inside ? 1 ()) ht = nt / nc, ddn ss2t = 1.0f - nnt *), N); >)

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

= * diffuse; = true;

efl + refr)) && (depth < MODEPTI

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

AXDEPTH)

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

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)

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

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





Spheres: one 50% specular, one glass sphere

hics & (depth < Modern

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

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

= * diffuse = true;

efl + refr)) && (depth < MODEPTI

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

AXDEPTH)

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

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

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

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





tics & (depth < Mox000

t = inside 7 1 ht = nt / nc, ddn = 0 os2t = 1.0f - nnt ™ n D, N); ≥)

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

= * diffuse; = true;

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

>, N); refl * E * diffuse;



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

ALC: NO.



sics & (depth < Moore

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

= * diffuse; = true;

• efl + refr)) && (depth < MAXDEPT

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

AXDEPTH)

survive = SurvivalProbability(diffuse estimation - doing it properly, closed if; adiance = 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)

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

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

Ray Tree

Recursion, multiple light sampling and path splitting in a Whitted-style ray tracer leads to a structure that we refer to as the *ray tree*.

All energy is ultimately transported by a single primary ray.

Since the energy does not increase deeper in the tree (on the contrary), the average amount of energy transported by rays decreases with depth.

ics & (depth < Modern

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

= * diffuse = true;

• •fl + refr)) && (depth < MAXDEPILL

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

AXDEPTH)

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

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

Today's Agenda:

- Recap
- Normals
- Reflections
- Recursion
- Shading models
- TODO



INFOGR – Lecture 7 – "Ray Tracing (2)"

Shading

AXDEPTH)

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

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

andom walk - done properly, closely follow /ive)

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

Diffuse Material

A diffuse material scatters incoming light in all directions.

Incoming:

Absorption:

Reflection:

 $E_{light} * \frac{1}{dist^2} * \vec{N} \cdot \vec{L}$

 $(\vec{V}\cdot\vec{N})$

 $(\vec{V}\cdot\vec{N})$

Eye sees:

 $C_{material}$

terms cancel out.

A diffuse material appears the same regardless of eye position.



tics & (depth < ⊅00000

: = inside ? 1 () ht = nt / nc, ddn os2t = 1.0f - n⊓t " n∩ 0, 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 < MODEPTI

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

AXDEPTH)

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

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

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

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

Specular Material

A specular material reflects light from a particular direction in a single outgoing direction.





tics & (depth < >∨∞000

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

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

= * diffus = true;

. efl + refr)) && (

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

AXDEPTH)

survive = SurvivalPr estimation - doing Hf; radiance = SampleLig e.x + radiance.y + r

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

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





tics & (depth < Notice

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

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

= * diffuse; = true;

efl + refr)) && (depth < MOXDE

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

AXDEPTH)

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

v = true;

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

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

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

Glossy Material

A glossy material reflects *most* light along the reflected vector.

 $\vec{R} = \vec{L} - 2(\vec{L}\cdot\vec{N})\vec{N}$

For other directions, the amount of energy is:

 $(\vec{V} \cdot \vec{R})^{\alpha}$, where exponent α determines the specularity of the surface.



 \vec{L}

 \vec{N}

at a = nt

efl + refr)) && (depth -

refl * E * diffuse;

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse(| at3 factor = diffuse * INVPI at weight = Mis2(directPdf, brdf 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:

Phong Shading

Complex materials can be obtained by blending diffuse and glossy.

$$V = C_{material} * \left((1 - f_{spec}) (\vec{N} \cdot \vec{L}) + f_{spec} (\vec{V} \cdot \vec{R})^{\alpha} \right)$$

where

α

Ispec

is the specularity of the glossy reflection; is the glossy part of the reflection; $1 - f_{spec}$ is the diffuse part of the reflection.

Note that the glossy reflection *only* reflects light sources.



ics & (depth < Modern

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

= * diffuse = true;

• •fl + refr)) && (depth < MAXDEPILL

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

AXDEPTH)

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

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

Today's Agenda:

- Recap
- Normals
- Reflections
- Recursion
- Shading models
- TODO



Limitations of Whitted-style

nics & (depth < PAXDE-

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

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

= * diffuse = true;

efl + refr)) && (depth < 1

), N); ∵efl * E * diff = true;

AXDEPTH)

survive = SurvivalProbabilif
estimation - doing it prop
f;
radiance = SampleLight(&rar
e.x + radiance.y + radiance

v = true; at brdfPdf = EvaluateDiffuse tt3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely following /ive)

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



NOS-SOP

Limitations of Whitted-style

hics & (depth < Not00⊓

c = inside ? 1 ht = nt / nc, ddn os2t = 1.0f - nmt 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 < MAXDEPIN

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

AXDEPTH)

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

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

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

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





Limitations of Whitted-style

ics & (depth < Modern

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse t3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

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

, t33 brdf = SampleDiffuse(diffuse, N, r1, r2, 8R, 5pdf urvive; .pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:





Limitations of Whitted-style

tics & (depth < MODEF)

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

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

= * diffuse; = true;

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

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

AXDEPTH)

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

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

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





Limitations of Whitted-style

tics & (depth < ⊁000)

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

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

= * diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbab
estimation - doing it p
if;
radiance = SampleLight(.
e.x + radiance.y + radia

v = true; at brdfPdf = EvaluateDif.usc(_2,...,) at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf

andom walk - done properly, closely following /ive)

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







sics & (depth < Notion

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

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

= * diffuse = true;

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

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

AXDEPTH)

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

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

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

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

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2019

END OF lecture 7: "Ray Tracing (2)"

Next up: "Midterm Exam"



Incoming: MidTerm

What to Study:

- Math lecture material (slides, annotated notes)
- Practice: tutorial sheet 1..4 (answers available)
- Check your skills using the practice midterm exam
- Extra: take the 2016/2017 or 2017/2018 midterm
- Read and memorize the NeedToKnow sheets (four, including intro)

The exam is a blend of math questions and theory: 75% math, 25% theory (approx.).

Points will be awarded for partially correct answers.

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)

at a = nt

), N);

AXDEPTH)

lf;

efl + refr)) && (dept

refl * E * diffuse;

survive = SurvivalProbability(

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

andom walk - done properly, closely following Sec vive)

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

