tic: € (depth < 100

= inside / 1 nt = nt / nc, dde os2t = 1.8f - nnt o, N); 3)

at a = nt - nc, b - mt at Tr = 1 - (80 + (1 - 7) fr) R = (D * nnt - 8 *

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

S C H W A R Z E N E G G E R

Get ready for the ride of your life.



tic: (depth < NAX

:= inside / L it = nt / nc, dd os2t = 1.01 0, N); 3)

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

= diffuse = true;

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

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

AXDEPTH)

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

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

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

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

INFOGR – Computer Graphics

Jacco Bikker & Debabrata Panja - April-July 2017

Lecture 15: "Grand Recap"

Welcome!



RECAP

tice ≰ (depth < 10.5

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

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

= diffuse = true;

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

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

AXDEPTH)

v = true; at brdfPdf = EvaluateDiffuse(L, N) * Pours) at3 factor = diffuse * INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * 000

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

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

Lecture 1: Rasters, Vectors, Colors

Concepts: Raster, discretization, anti-aliasing, rasterization, frame rate, vertical retrace, 'frame-less', RGB colors, 16-bit, palletized, HDR.











RECAP

tice k (depth ⊂ 100

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

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

E ⁼ diffuse = true;

efl + refr)) && (depth is HANDER

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

AXDEPTH)

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

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

andom walk - done properly, closely following /ive)

; st3 brdf = SampleDiffuse(diffuse, N, F1, F2, UR, L, L pdf; n = E * brdf * (dot(N, R) / pdf); sion = true;

Lecture 3 – part 2: Ray Tracing Intro

Concepts:

The "God Algorithm": light transport in nature, light transport in a ray tracer, ray tracing versus rasterization, convex / concave, reflection and shadows in a rasterizer, global data, ray optics, refraction, reflection, Fresnel, Snell, Whitted-style (recursive) ray tracing, ray equation, ray setup, normalization, ray/plane intersection, ray/sphere intersection, primary rays, shadow rays, shadow acne, distance attenuation, absorption, N dot L.









RECAP

tice k (depth < 100

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

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

= diffuse = true;

efl + refr)) && (depth & NADIII

D, N); refl * E * diffus = 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) = Pour st3 factor = diffuse = INVPI; at weight = Mis2(directPdf, brdfPdf); at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) = 0000

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

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

Lecture 3 – part 2: Ray Tracing Intro

Make sure you can:

- Explain why the efficient ray/sphere intersection code on slide 30 will not work for glass spheres;
- Setup a proper ray given a view direction, FOV and up vector;
- Explain why you need an up vector.







RECAP

tic: K (depth < 100

= inside / 1 nt = nt / nc, dda 552t = 1.8f - nn 5, N); 3)

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

E * diffuse; = true;

-: efl + refr)) && (depth & MADIIII

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property if; radiance = SampleLight(&rand I = 1, e.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)

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

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

Lecture 5 – Ray Tracing (2)

Concepts:

Calculating normals, vertex normal, normal interpolation, normal for a plane and a sphere, partially reflective surfaces, HDR sky dome, ray tree, diffuse material, Lambert, glossy, Phong, dielectrics, limitations of Whitted-style ray tracing.





Pliaht

RECAP

tic: k (depth < 10.)

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

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

= diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability(different estimation - doing it properly if; radiance = SampleLight(&nand, I, M, SampleLight(&nand, I, M, SampleLight(&nand, I, M, SampleLight(&nand, I, M))

v = true; t brdfPdf = EvaluateDiffuse(L, N.) Promote 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, R, R, r pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Lecture 5 – Ray Tracing (2)

Make sure you can:

- Construct a vector reflected in a plane;
- Explain why a bathroom mirror is (close to) white;
- Explain why we need a cap on recursion;
- Explain why rays transport little energy in a deep ray tree;









RECAP

tic: € (depth < 10

: = inside / L nt = nt / nc, dd 552t = 1.0f - on 5, N); 8)

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

E * diffuse; = true;

efl + refr)) && (depth < HAX

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

AXDEPTH)

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)

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

; t3 Brdf = SampleDiffuse(diffuse, N, F1, F2, 48, 555 pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Lecture 6 – Accelerate

Concepts:

AABB, culling, bounding volume hierarchy, conservative tests, false negatives, early out, precalculate, loop hoisting, incremental rendering, rasterization, z-buffer, global data, slab test, ray tracing versus rasterization, spaces, scene graph, engine design.

Make sure you can:

- Construct an AABB for a triangle, sphere, mesh, ... ;
- Intersect a ray and a triangle;
- Intersect a ray and an AABB using the slab test;
- Cull a sphere and an AABB against a frustum.









RECAP

tic: K (depth (⊂)04

= inside / 1 nt = nt / nc. dd ps2t = 1.0f - nn 2, N); 3)

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

= diffuse; = true;

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

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

AXDEPTH)

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)

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

; t3 brdf = SampleDiffuse(diffuse, N, r1, r2, R2, lp); urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Lecture 8: OpenGL

Concepts:

OpenGL, ARGB, Khronos Group, Fahrenheit, Glide, 3dfx, VooDoo, HW T&L, OpenGL coordinate system, right handed, immediate mode, retained mode, state machine, GPU, streaming processor, caches, GPU model, state changes, vertex buffer objects, pixel shader, vertex shader.











Right Handed Coordinates

RECAP

tica k (depth < 100

= inside / 1 nt = nt / nc, dda 552t = 1.8f - nn 5, N); 3)

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

= diffuse = true;

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

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

AXDEPTH)

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

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

; st3 brdf = SampleDiffuse(diffuse, N, F1, F2, UR, body urvive; pdf; n = E * brdf * (dot(N, R) / pdf); sion = true:

Lecture 10: Shading Models

Concepts:

World space, camera space, model space, uniform variables, glossy, Phong, Phong exponent, ambient, planar reflections, environment maps, normal maps, tangent space, fur shells.









RECAP

tic: € (depth () (0)

= inside / 1 nt = nt / nc, dd ps2t = 1.0f p, N); 3)

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

= diffuse = true;

≃ efl + refr)) && (depth < HACOPOT

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

AXDEPTH)

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

v = true; t brdfPdf = EvaluateDiffuse(L, N.) Provident st3 factor = diffuse * INVPI; st weight = Mis2(directPdf, brdfPdf); st cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf) * 1

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

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

Lecture 11: Visibility

Concepts:

Painter's, overdraw, BSP traversal (back-to-front, front-to-back), z-buffer, values in the z-buffer, z-fighting, Sutherland-Hodgeman clipping, n-gons, guard bands, back-face culling, frustum culling, hierarchical bounding volume culling, culling using a grid, portals: visibility, mirrors, 'portals'.

Questions?







RECAP

tici k (depth < 100

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

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

= diffuse = true;

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

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

AXDEPTH)

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

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

andom walk - done properly, closely following /ive)

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

Lecture 12: Post Processing

Concepts:

Post processing, camera / sensor behavior, lens flares, vignetting, chromatic aberration, noise / grain, HDR bloom and glare, tone mapping / exposure control, color correction / grading, gamma, gamma correction, depth of field, circle of confusion, ambient occlusion, screen space AO, bilateral filtering, screen space reflections, limitations of screen space approaches.









Questions?

SCHWARZENEGGER

Get ready for the ride of your life.

RE(CAP

TOTAL RECAP

TOTAL RECAP

TOTA

RECAP

tic: k (depth < NASS

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

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

= diffuse; = true;

-: efl + refr)) && (depth < HANDIIII

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

AXDEPTH)

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

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

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

SCHWARZENEGGER SCHWARZENEGGER

Ge

Get ready for the ride of your life.

What's Next?

Upcoming Attractions:



at a = nt

efl + refr)) && (depth is HAD

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

AXDEPTH)

survive = SurvivalProbability(dif if; adiance = 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) at cosThetaOut = dot(N, L); E * ((weight * cosThetaOut) / directPdf)

andom walk - done properly, closely follow vive)

at3 brdf = SampleDiffuse(diffuse, N, r1, r2, HR, H) rvive; pdf; 1 = E * brdf * (dot(N, R) / pdf); sion = true:

- Final Exam: Thursday June 29, 13:30
- P3 deadline: Tuesday June 27, 23:59
- Retake Exam: Thursday July 13, 13:30

Master:

- **Optimization & Vectorization**
- Advanced Graphics





tic: k (depth < 100

: = inside / L it = nt / nc, dde os2t = 1.01 0, N); 0)

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

= diffuse = true;

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

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

AXDEPTH)

survive = SurvivalProbability(difference estimation - doing it properly ff; radiance = SampleLight(%rane.x + radiance.y + radiance

w = true; at brdfPdf = Evalua at3 factor = diffuse at weight = Mis2(dire. at cosThetaOut = dot(N, E * ((weight * cosThetaOut))

andom walk - done properly, closely follo /ive)

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

ARACAL.

INFOGR – Computer Graphics

Jacco Bikker - April-July 2016 - Lecture 14: "Grand Recap"

next up: "Final Exam"



