tics ⊾ (depth () ().

: = inside / L it = nt / nc, dde os2t = 1.01 - ... ), N ); 3)

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

= diffuse = true;

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

J. Bikker - April-July 2015 - Lecture 8: "Ray Tracing"

# Welcome!



tic: € (depth < 1555

:= inside / i ht = nt / nc, dde 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 ( MAND)

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

#### AXDEPTH)

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

v = true;

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

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, R, F3, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

# Today's Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions



# Practicum

#### ic: (depth < NAC

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

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

= diffuse; = true;

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

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

#### AXDEPTH)

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

#### v = true;

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

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

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

## Exam

- Answers will be available today
- Can be discussed in tutorial (Thursday)
- Still questions?
- Make an appointment.

## P2

- Will be graded as soon as possible
  - Partner dropped? Let me know.

## Final challenge is coming up...





# Practicum

#### tic: k (depth < 10.⊂

:= inside / i it = nt / nc, dde -552t = 1.0f = nn: ), N ); ))

at a = nt - nc, b - n1 at Tr = 1 - (R0 - 1 Tr) R = (D \* nnt - 10

= diffuse; = true;

efl + refr)) && (depth is HADDIT

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

#### AXDEPTH)

survive = SurvivalProbability H.1 if: adiance = SampleLight( &rand, I H.2 e.x + radiance.y + radiance.z) H.3 v = true; at brdfPdf = EvaluateDiffuse( ) st3 factor = diffuse \* INVPI: H.4 st weight = Mis2( directPdf, brdfPdf at cosThetaOut = dot( N, L ); E \* ((weight \* cosThetaOut) / directPd H.5 andom walk - done properly, closely for /ive) H.6

#### st3 brdf = SampleDiffuse( diffuse, N, r1, r2, NR, set prvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

## P3 available today!

## Easy Assignments (choose four):

E.1	Multiple Light Sources
E.2	Spotlight
E.3	Cell Shading
E.4	Frustum Culling
E.5	Color Filter
E.6	Gaussian Blur

# Medium Assignments (choose up to two, or up to one hard one):

- M.1 Textured Light
  M.2 Shadow Mapping
  M.3 Reflection
  M.4 Cube Mapping
  M.5 Transparency
  M.6 Bloom
- M.7 God Rays (Volumetric Lighting)

Hard Assignments (choose up to one, or two medium ones):

- HDR Lighting and Tone Mapping
- Deferred Shading
- Parallax Mapping
- Screen-space Ambient Occlusion

Image-based lighting (see M.4 for description; H.5 is the "hard" variant of M.4) Ray Tracing



tic: € (depth < 1555

:= inside / i ht = nt / nc, dde 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 ( MAND)

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

#### AXDEPTH)

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

v = true;

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

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, R, F3, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

# Today's Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions



# Introduction

tic: ⊾(depth < 10

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

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

E \* diffuse; = true;

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

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

AXDEPTH)

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

w = true; st brdfPdf = EvaluateDiffuse( L, N ) Pourst3 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, F1, F2, UR, Doff pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

Rendering – "The God Algorithm"

Image synthesis in nature:

Large amounts of particles, emitted by light sources, bouncing around a scene, until they reach a sensor.

Pro: Simple.

Con: No way we can simulate that many particles.





# Introduction

tic: ⊾(depth ∈ N

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

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

= diffuse; = true;

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

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

#### AXDEPTH)

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

v = true; st 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)

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

Rendering – Rasterization

Image synthesis using rasterization:

Draw one triangle at a time, estimate the color of each pixel it covers, using available data.

*Note that this is supposed to simulate the correct solution:* 

- How much light does each fragment reflect towards the camera?
- How much light arrives at each fragment?





# Introduction

tic: k (depth < NAS⊂

- \* inside / .
it = nt / nc, dde
ss2t = 1.0f - nt
), N );
3)

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

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

**Rasterization - Reflection** 

- How much light does each fragment reflect towards the camera?
- How much light arrives at each fragment?

We can answer these questions accurately for a limited scene:

- One or more point lights
- A single convex object





# Introduction

tice ≰ (depth < 1935

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

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

= diffuse = true;

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

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

#### AXDEPTH)

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

v = true;

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

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

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

Rasterization - Reflection

For a more elaborate scene, things break:

- Occluded incoming light (shadows)
- Indirect light

It gets worse if we don't limit ourselves to point lights.





# Introduction

tic: k (depth < 100

= = inside / 1 ht = nt / nc, ddo 552t = 1.0f = nnt 5, N ); 8)

at a = nt - nc, b - nt - at Tr = 1 - (R0 + 11 - 73 Tr) R = (D \* nnt - N \*

= diffuse; = true;

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

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

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

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

**Rasterization - Reflection** 

We can simulate certain light transport paths, such as reflections:

Here, the light arriving at the bunny is stored in an 'environment map'.

This is only accurate when reflecting an infinitely far environment.





# Introduction

tic: ⊾(depth < 12

= = inside / 1 ht = nt / nc, d/m -552t = 1.0f 3, N ); 3)

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

= diffuse; = true;

efl + refr)) && (depth < MACOLOGI

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

AXDEPTH)

survive = SurvivalProbability difference estimation - doing it property ff; radiance = SampleLight( %rand, I & e.x + radiance.y + radiance.z) = 0.000

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

sndom walk - done properly, closely following vive)

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

**Rasterization - Shadows** 

We can simulate occlusion using shadow maps:

A z-buffer from the viewpoint of the light source allows us to see, per fragment, if we are closer to the light than an occluder.

This works only for point lights.

The shadow map is a raster, which is often visible.





# Introduction

11c) ⊾ (depth < 114)

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

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

= diffuse = true;

efl + refr)) && (depth k HANDIII

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse( L, N.) \* Praction 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, Npd prvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true;

Rasterization – Fundamental limitations

When drawing triangles one by one, what we lack is global data.

Environment maps and shadow maps capture some data about the environment of the triangle, but this data is a (coarse) approximation.

The core algorithm supports direct light, without occlusion:





# Introduction

		<u>: :::::::::::::::::::::::::::::::::::</u>	
		Danse (1997)	Billion and States
<u>Ö</u> dre	Wing The answer		
			Paral Marine
		E 17 Element Burnard	
		A CAPACITY OF A	

e.x + radiance v = true;

adiance = Sam

), N );

MAXDEPTH) survive = Surv

st brdfPdf = EvaluateOiffuse( L, N ) Point 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)

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





# Introduction

tice ↓ (depth ← Nord ↓ • inside / )

nt = nt / nc. os2t = 1.8f D; N ); D)

st a = nt - nc, st Tr = 1 - (R0 Fr) R = (D ° nn

= diffus = true;

. :fl + refr)) &

), N ); efl \* E \* di = true;

AXDEPTH)

survive = SurvivalP
estimation - doing
if;
radiance = SampleLi
e.x + radiance.y +

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

andom walk - done properly, closely following

; ot3 brdf = SampleDiffuse( diffuse, N, r1, r2, UR, Dot prvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true;







# Introduction

tic: ⊾(depth ⊂ NA

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

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

= diffuse; = true;

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

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

#### AXDEPTH)

survive = SurvivalProbability difference estimation - doing it properly if; radiance = SampleLight( %rand I = 1 .x + radiance.y + radiance.r) > \_\_\_\_\_\_

v = true; at brdfPdf = EvaluateDiffuse( L, N ) \* Pource 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 a /ive)

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

## Ray Tracing

This lecture: basic algorithm Next lecture: 'ground truth'

After that, we converge with rasterization again.



tic: € (depth < 1555

:= inside / i ht = nt / nc, dde 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 ( MAND)

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

#### AXDEPTH)

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

v = true;

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

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, R, F3, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

# Today's Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions



# Ray Tracing

## Ray Tracing:

## World space

- Geometry
- Eye
  - Screen plane
  - Screen pixels
  - Primary rays
  - Intersections
  - Point light
  - Shadow rays
- survive = SurvivalProbability estimation - doing it proposed #f; radiance = SampleLight( %rand e.x + radiance.y + radiance.z)
- w = true; at brdfPdf = EvaluateDiffuse( L, Extension rays at3 factor = diffuse \* INVPI; at weight = Mis2( directPdf, brdfPdf ); at cosThetaOut = dot( N, L) E \* ((weight \* cosThetaOut)Light transport
- andom walk done properly, closely following : /ive)

), N );

= true;

AXDEPTH)

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





# Ray Tracing

## Ray Tracing:

## World space

- Geometry
- Eye

- Screen plane
- Screen pixels
- Primary rays
- Intersections
- Point light
  - Shadow rays
- survive = SurvivalProbability estimation - doing it prop #f; radiance = SampleLight( & rand e.x + radiance.y + radiance.z)
- w = true; at brdfPdf = EvaluateDiffuse(L, || Extension rays st3 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 : /ive)

), N );

= true;

AXDEPTH)

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







# Ray Tracing

## Ray Tracing:

## World space

- Geometry
- Eye
  - Screen plane
  - Screen pixels
  - Primary rays
  - Intersections
  - Point light
  - Shadow rays
- survive = SurvivalProbability Light transport adiance = SampleLight( &ra e.x + radiance.y + radiance.z
- Extension rays t brdfPdf = EvaluateDiffuse( ) st3 factor = diffuse = INVPI t weight = Mis2( directPdf, brdfPd1 et cosThetaOut = dot( N, L) E • ((weight = cosThetaOut)Light transport
- andom walk done properly, closely foll vive)

), N );

= true;

AXDEPTH)

if:

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





We are calculating light transport backwards.





# **Ray Tracing**

lc: 6 (depth = )) 5 = inside /

nt = nt / nc. 552t = 1.0f 5, N ); 3)

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

diffuse = true;

. :**fl + refr**)) && (dep:

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

AXDEPTH)

survive = SurvivalProb estimation - doing it if; radiance = SampleLight e.x + radiance.y + rad

v = true; at brdfPdf = EvaluateD st3 factor = diffuse " st weight = Mis2( dire st cosThetaOut = dot( E \* ((weight \* cosTheta))

andom walk - done properly, /ive)

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



لاندان ما تند عليه اسط مستوغير و فلان هذا الشطح يقط سط برض عن تعلق ت فلا بد من لا يقط احل خلى ب ن ب فلكن ذلك الخط مستو الفصل المشترك بين هذا السط و بين مط قط ق حط مستر فلات هذا السط يما ترب سيط م على فقط ت غنط مستر على قط ق م د على نقطة مت وكذلك خط م عرف المحال فلا يا ترب بط مت على فقطة مت سط مستيو غير سلح م ب ن م م ن







# **Ray Tracing**



survive = SurvivalProl
estimation - doing if
if;
adiance = SampleLigh
e.x + radiance.y + radiance.y = ra

v = true; at brdfPdf = EvaluateD st3 factor = diffuse \* at weight = Mis2( dire at cosThetaOut = dot( E \* ((weight \* cosThetaOut) / /

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

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









# Ray Tracing

tice ≰ (depth < 10.5

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

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

andom walk - done properly, closely following /ive)

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

# Physical basis:

Ray tracing uses *ray optics* to simulate the behavior of light in a virtual environment.

It does so by finding light transport paths:

- From the 'eye'
- Through a pixel
- Via scene surfaces
- To one or more light sources.

At each surface, the light is modulated. The final value is deposited at the pixel (simulating reception by a sensor).



tic: € (depth < 1555

:= inside / i ht = nt / nc, dde 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 < MAND

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

#### AXDEPTH)

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

v = true;

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

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, R, F3, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

# Today's Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions



# Intersections

tic: K (depth < 19

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

= diffuse; = true;

: f**1 + refr))** && (depth is MAXDIII

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

AXDEPTH)

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

v = true;

at brdfPdf = EvaluateDiffuse( L, N.) \* Parts 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, soft urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

## Ray definition

A ray is an infinite line with a start point:

 $p(t) = 0 + t\vec{D}$ , where t > 0.

### struct Ray

};

float3 0; // ray origin
float3 D; // ray direction
float t; // distance

The ray direction is generally normalized.



# Intersections

ice (depth c NACC

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

= diffuse; = true;

efl + refr)) && (depth k HANDIII

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

#### AXDEPTH)

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

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

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

sndom walk - done properly, closely following vive)

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

## Ray setup

A ray is initially shot through a pixel on the screen plane. The screen plane is defined in world space:

Camera position:E = (0,0,0)View direction: $\vec{V}$ Screen center: $C = E + d\vec{V}$ Screen corners: $p_0 = C + (-1)$ 

## ter: $C = E + d\vec{V}$ ners: $p_0 = C + (-1, -1, 0), p_1 = C + (1, -1, 0), p_2 = C + (-1, 1, 0)$

## From here:

- Change FOV by altering *d*;
- Transform camera by multiplying E,  $p_0$ ,  $p_1$ ,  $p_2$  with the camera matrix.



# Intersections

tic: k (depth ∈ 100

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

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

= diffuse; = true;

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

D, N ); ~efl \* E \* diff = true;

AXDEPTH)

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

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

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

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

Ray setup

Point on the screen:

$$p(u, v) = p_0 + u(p_1 - p_0) + v(p_2 - p_0)$$

Ray direction (before normalization):

$$\vec{v} = p(u, v) - E$$

Ray origin:

 $\overline{O} = E$ 

Real Property in the second se



# Intersections

tic: ⊾ (depth ⊂ 10.

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

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

= diffuse; = true;

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

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

AXDEPTH)

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

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

andom walk - done properly, closely followin /ive)

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

## Ray intersection

Given a ray  $p(t) = 0 + t\vec{D}$ , we determine the closest intersection distance t by intersecting the ray with each of the primitives in the scene.

Ray / plane intersection:

Plane: 
$$\mathbf{p} \cdot \vec{N} + d = 0$$
  
Ray:  $p(t) = 0 + t\vec{D}$ 

Substituting for p(t), we get

 $(O + t\vec{D}) \cdot \vec{N} + d = 0$  $t = -(O \cdot \vec{N} + d) / (\vec{D} \cdot \vec{N})$  $P = O + t\vec{D}$ 





# Intersections

), 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

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

andom walk - done properly, closely vive)

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

Ray intersection

Ray / sphere intersection:

Sphere:  $(p - C) \cdot (p - C) - r^2 = 0$ 

Substituting for p(t), we get

 $(0+t\vec{D}-C)\cdot(0+t\vec{D}-C)-r^2=0$  $\vec{D} \cdot \vec{D} t^{2} + 2\vec{D} \cdot (0 - C) t + (0 - C)^{2} - r^{2} = 0$ 

 $ax^2 + bx + c = 0 \rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{-b \pm \sqrt{b^2 - 4ac}}$ 2a

 $a = \vec{D} \cdot \vec{D}$  $b = 2\vec{D} \cdot (O - C)$  $c = (0-C) \cdot (0-C) - r^2$ 

Negative: no intersections

E



 $p_0$ 

 $p_2$ 

31

 $p_1$ 

tic: € (depth < 1555

:= inside / i ht = nt / nc, dde 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 < MAND

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

#### AXDEPTH)

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

v = true;

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

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, R, F3, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

# Today's Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions



tic: k (depth < 10.

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

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

E \* diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

## Shading basics

Given a point light *L*, at distance *d* from a surface point *P* with normal  $\vec{N}$ ...

...how much light arrives at P from L?

## Depends on:

- Distance
- Angle

Visibility

ð

 $\vec{N}$ 

P



tice k (depth < 152)

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

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

= diffuse; = true;

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

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

AXDEPTH)

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 /ive)

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

## Shading basics

Given a point light *L*, at distance *d* from a surface point *P* with normal  $\vec{N}$ ...

...how much light arrives at P from L?

P

## Depends on:

- Distance
- Angle

Visibility

 $\overline{d^2}$ 

1

**Relation:** 

Distance attenuation



tic: ≹ (depth < 100

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

E = diffuse; = true;

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

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

AXDEPTH)

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

v = true; at brdfPdf = EvaluateDiffuse( L, N ) \* Punn 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, F1, F2, NR, ND) urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

## Shading basics

Given a point light *L*, at distance *d* from a surface point *P* with normal  $\vec{N}$ ...

...how much light arrives at P from L?

P

## Depends on:

- Distance
- Angle

Visibility

## Relation:







ici Admeth o Illino

= inside / i
nt = nt / nc, ddd ss2t = 1.8f - nnt 0, N );
3)

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

= diffuse; = true;

efl + refr)) && (depth is MADDI-

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

#### AXDEPTH)

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

v = true; t brdfPdf = EvaluateDiffuse( L, N ) \* Pro 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)

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

## Ray intersection

```
Color Trace( vec3 0, vec3 D )
{
    I, N, mat = IntersectScene( 0, D );
    if (!I) return BLACK;
    return DirectIllumination( I, N ) * mat.diffuseColor;
```

### Color DirectIllumination( vec3 I, vec3 N )

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



## Ray intersection

Color Trace( vec3 0, vec3 D )

```
ic:
(depth < NAS
```

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

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

= diffuse; = true;

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

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

#### AXDEPTH)

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

#### v = true;

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

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

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

# 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;



## Ray intersection

else

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

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

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

survive = SurvivalProbability( diff

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

efl + refr)) && (depth < )

refl \* E \* diffuse;

), N );

AXDEPTH)

if:

andom walk - done properly, closely following /ive)

```
;

pt3 brdf = SampleDiffuse( diffuse, N, r1, r2,

pdf;

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

sion = true:
```



# Whitted

tica ⊾ (depth ⊂ 10.

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

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

= diffuse = true;

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

D, N ); ref1 \* E \* diff: = true;

#### AXDEPTH)

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

v = true; st brdfPdf = EvaluateDiffuse( L, N ) = Paur 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, S); pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:





tic: € (depth < 1555

:= inside / i ht = nt / nc, dde 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 < MAND

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

#### AXDEPTH)

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

v = true;

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

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

; pt3 brdf = SampleDiffuse( diffuse, N, F1, F2, R, F3, pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

# Today's Agenda:

- Introduction to Ray Tracing
- Whitted-style Ray Tracing
- Intersections
- Shadows, Reflections & Refractions



tic: ≰ (depth < 100

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

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

E <sup>=</sup> diffuse = true;

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

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

AXDEPTH)

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, NR, Sch pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

# INFOGR – Computer Graphics

dr. ing. J. Bikker - April-July 2015 - Lecture 8: "Ray Tracing"

# END of "Ray Tracing"

next lecture: "Shading Models"

