tics & (depth < No:0

: = inside ? l | ... ht = nt / nc, ddn | ... bs2t = 1.0f - nnt | ... D, N ); B)

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

= \* diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly ff; radiance = SampleLight( &rand, I, x + radiance x + radiance z)

v = 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 Sec /ive)

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

## **INFOMAGR – Advanced Graphics**

Jacco Bikker - November 2021 - February 2022

## Lecture 5 - "The Perfect BVH"

 $\rho(x,x',x'')I(x',x'')dx''$ 

Welcome!



tics & (depth < Modeo

: = inside ? l | ... ht = nt / nc, ddn os2t = 1.0f - nnt = n O, N ); 0)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0) r) 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 H; radiance = SampleLight( &rand, I, &L, &light) e.x + radiance.y + radiance.z) > 0) && closed

w = 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:

### Today's Agenda:

- Building Better BVHs
- Refitting
- Fast BVH Construction
- The Top-level BVH



### Better BVHs

hics & (depth < Notes

t = inside ? 1 1 1 1 ht = nt / nc, ddn 1 552t = 1.0f - nnt " n 2, N ); 8)

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

= \* diffuse; = true;

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

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

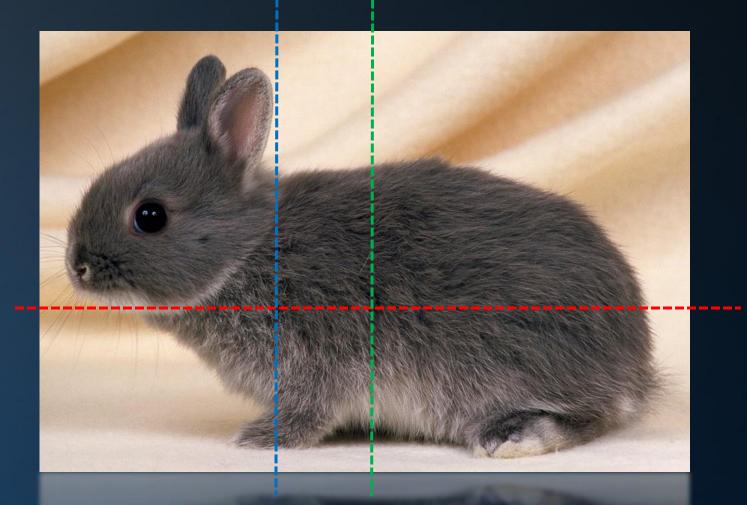
AXDEPTH)

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

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) \* (PS

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

; t3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, 6pdf urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:





### Better BVHs

tics & (depth < ™0000

t = inside ? 1 1 1 0 ht = nt / nc, ddn 0 bs2t = 1.0f - nnt ° n 2, N ); ≷)

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

= \* diffuse; = true;

efl + refr)) && (depth < MAXDEPTH

), 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) &&

v = 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) (rad

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

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





### Better BVHs

sics & (depth < Monos

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

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

= \* diffuse; = true;

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

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

AXDEPTH)

survive = SurvivalProbability( diffuse .estimation - doing it properly, closed if; radiance = SampleLight( &rand, I, &L, &light) 2.x + radiance.y + radiance.z) > 0) && closed 2.x + radiance.y + radiance.z) > 0) && closed 3.x + radiance.y + radiance.z) > 0) && closed 3.x + radiance.y + radiance.z) > 0) && closed 3.x + radiance.y + radiance.z) > 0) && closed 3.x + radiance.y + radiance.z) > 0) && closed 3.x + radiance.y + radiance.z) > 0) && closed 3.x + radiance.y + radiance.z) > 0) && closed 3.x + radiance.y + radiance.z) > 0) & & closed 3.x + radiance.y + radiance.z) > 0) & & closed 3.x + radiance.y + radiance.z) > 0) & & closed 3.x + radiance.y + radiance.z) > 0) & & closed 3.x + radiance.y + radiance.z) > 0) & & closed 3.x + radiance.y + radiance.z) > 0) & & closed 3.x + radiance.y + radiance.z) > 0) & & closed 3.x + radiance.y + radiance.z) > 0) & & closed 3.x + radiance.y + radiance.z) > 0) & & closed 3.x + radiance.z) + radiance.z) > 0) & & closed 3.x + radiance.z) + rad

v = true;

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

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

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

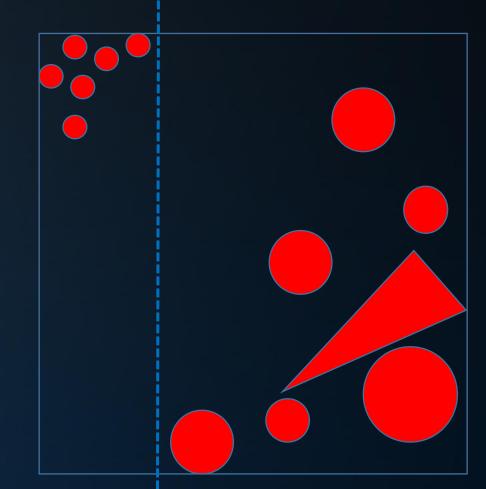
What Are We Trying To Solve?

A BVH is used to reduce the number of ray/primitive intersections.

But: it introduces new intersections.

The ideal BVH minimizes:

- # of ray / primitive intersections
- # of ray / node intersections.





### Better BVHs

hics & (depth < ™0000

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

= \* diffuse = true;

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

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

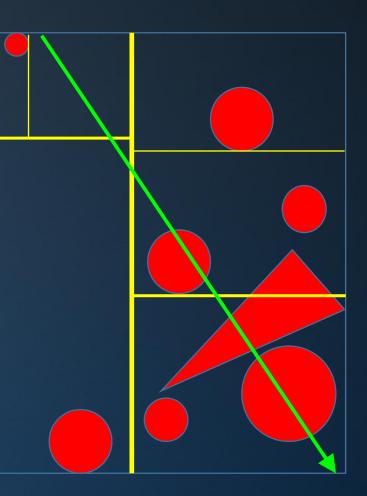
#### AXDEPTH)

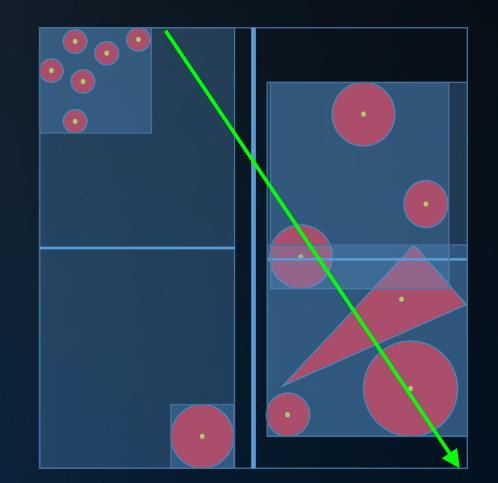
survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &L, &lis e.x + radiance.y + radiance.z) > 0) && (d)

v = 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 300. /ive)

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







### Better BVHs

sics & (depth < NADDS

: = 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 Fr) R = (D = nnt - N = (100

= \* diffuse; = true;

efl + refr)) && (depth < MOXDEPTH

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

AXDEPTH)

survive = SurvivalProbability( diffuestimation - doing it properly, if; radiance = SampleLight( %rand, I, %. 2.x + radiance.y + radiance.z) > 0) %

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

andom walk - done properly, closely following 30 /ive)

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

BVH versus kD-tree

The BVH better encapsulates geometry.

This reduces the chance of a ray hitting a node.

→ This is all about probabilities!

*What is the probability of a ray hitting a random triangle?* 

*What is the probability of a ray hitting a random node?* 

This probability is proportional to **surface area**.





### Better BVHs

2)
at a = nt
at Tr = ;
(r) R =
= true;
= true;
= true;
(r) N );
refl \* E
= true;

(AXDEPTH)

survive = estimati If; radiance e.x + rac

at3 factor = diffuse \* INVPI; at weight = Mis2( directPort at cosThetaOut = dot( N, Route 1: 10% up-time, \$1000 fine E \* ((weight \* cosThetaOut)

-

andom walk - done properly, closely foll /ive)

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



Route 2: 100% up-time, \$100 fine



### Better BVHs

sics & (depth < Mondor

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

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

= \* diffuse; = true;

efl + refr)) && (depth < MAXDEE

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

AXDEPTH)

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

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

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

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

#### **Optimal Split Plane Position**

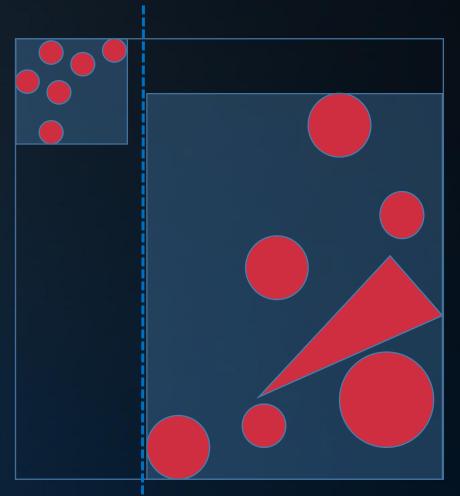
The ideal split minimizes the *expected cost* of a ray intersecting the resulting nodes.

This expected cost is based on:

- Number of primitives that will have to be intersected
- Probability of this happening

#### The cost of a split is thus:

 $A_{left} * N_{left} + A_{right} * N_{right}$ 





### Better BVHs

sics & (depth < MANDOR

: = inside ? 1 ht = nt / nc, ddn = 1 os2t = 1.0f - nnt " n 2, N ); 3)

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

= \* diffuse; = true;

efl + refr)) && (depth < MAXDEE

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

AXDEPTH)

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

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) \* (notest);

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

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

#### **Optimal Split Plane Position**

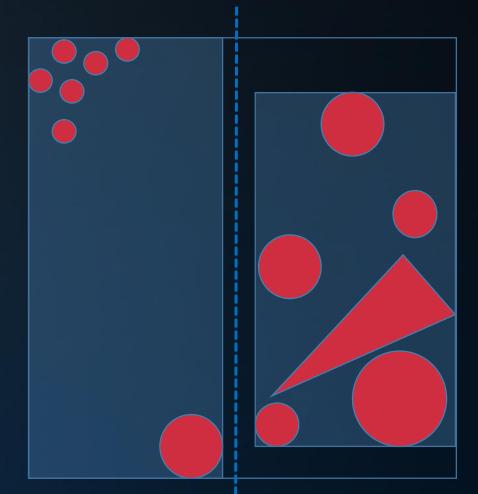
The ideal split minimizes the *expected cost* of a ray intersecting the resulting nodes.

This expected cost is based on:

- Number of primitives that will have to be intersected
- Probability of this happening

#### The cost of a split is thus:

 $A_{left} * N_{left} + A_{right} * N_{right}$ 





### Better BVHs

sics & (depth < Motoor

at a = nt - nc, b =  $m_{t}$  =  $p_{t}$ at Tr = 1 - (R0 + (1 - R0 Fr) R = (D = nnt - N = (dd)

= \* diffuse; = true;

efl + refr)) && (depth < NOCCE

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

AXDEPTH)

survive = SurvivalProbability( diffuse estimation - doing it properly, close if; radiance = SampleLight( &rand, I, &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, &pdf urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

**Optimal Split Plane Position** 

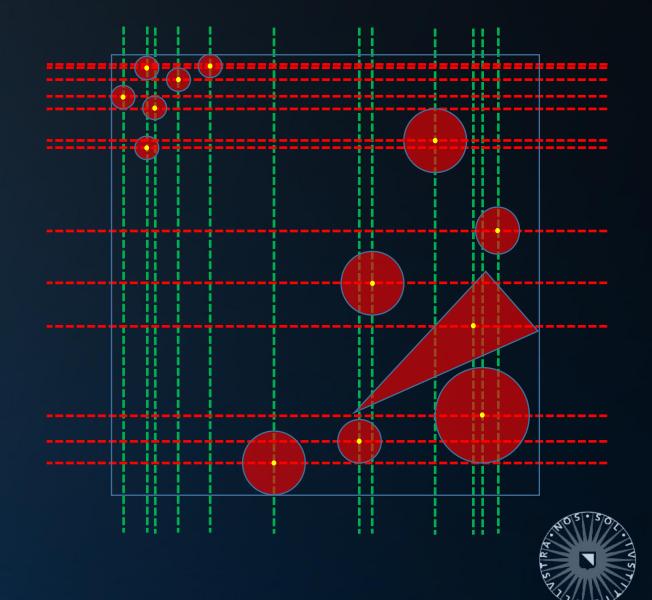
Which positions do we consider?

*Object subdivision may happen over x, y or z axis.* 

*The cost function is constant between primitive centroids.* 

→ For N primitives: 3(N - 1) possible locations

→ For a 2-level tree:  $(3(N-1))^2$  configurations



### Better BVHs

at a = nt

AXDEPTH)

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

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

andom walk - done properly, closely follo /ive)

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

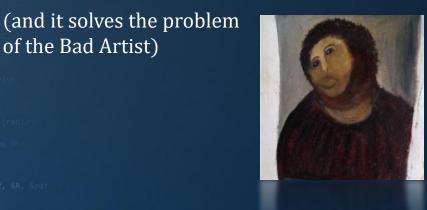
### SAH and Termination

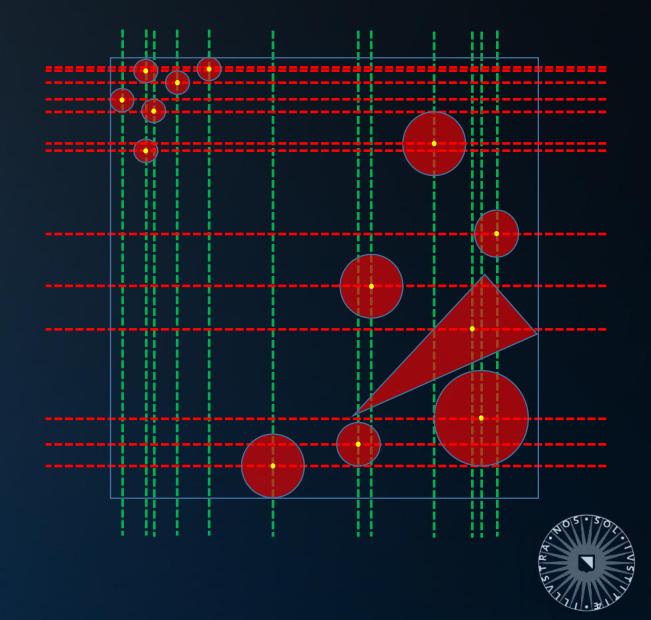
of the Bad Artist)

A split is 'not worth it' if it doesn't yield a cost lower than the cost of the parent node, i.e.:

 $A_{left} * N_{left} + A_{right} * N_{right} \ge A * N$ 

This provides us with a natural and optimal termination criterion.





### Better BVHs

nics & (depth < Noteen

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

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

= \* diffuse; = true;

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

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

\*: Heuristics for Ray Tracing using Space Subdivision, MacDonald & Booth, 1990.

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

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

# State Stat

**Optimal Split Plane Position** 

The *surface area heuristic* (SAH) is applied in a greedy manner\*.

### Better BVHs

sics & (depth < ⊅00000

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

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

= \* diffuse; = true;

∘ efl + refr)) && (depth < MAXDE

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

AXDEPTH)

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

w = true; at brdfPdf = EvaluateDiffuse( L, N ) Pau 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:

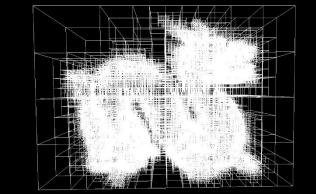
#### **Optimal Split Plane Position**

Comparing naïve versus SAH:

- SAH will cut #intersections in half;
- expect ~2x better performance.

#### SAH & kD-trees:

• Same scheme applies.





### Better BVHs

hics & (depth < ⊅000

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

at a = nt - nc, b = nt - n at Tr = 1 - (R0 + (1 - R0 Tr) R = (D <sup>+</sup> nnt - N - (d)

= \* diffus = true;

efl + refr)) && (depth < MAXDEPTO

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

#### AXDEPTH)

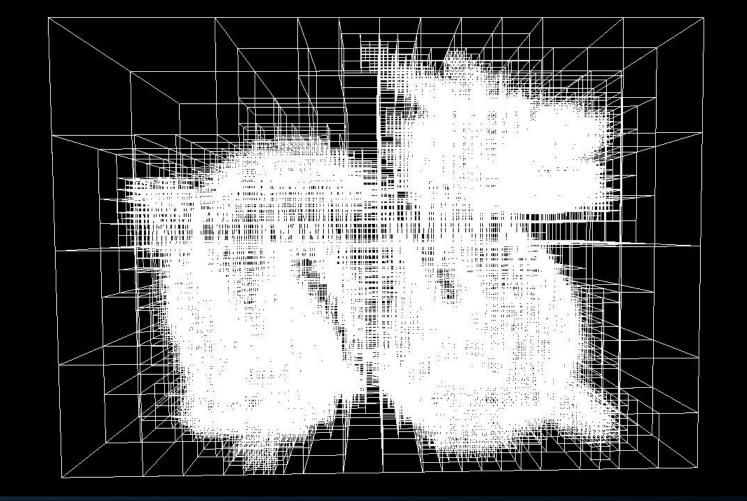
survive = SurvivalProbability( diffuse estimation - doing it properly, close if; radiance = SampleLight( &rand, I, &L,

e.x + radiance.y + radiance.z) > 0) 88

w = 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 following /ive)

; at3 brdf = SampleDiffuse( diffuse, N, c1, c2, 8R, 60; urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:



**Median Split** 



### Better BVHs

nics & (depth < 2000

: = inside / l ht = nt / nc, ddn bs2t = 1.0f - nmt / D, N ); B)

at a = nt - nc, b = nt - n at Tr = 1 - (R0 + (1 - R0 Fr) R = (D <sup>+</sup> nnt - N - (d)

= \* diffuse = true;

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

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

#### AXDEPTH)

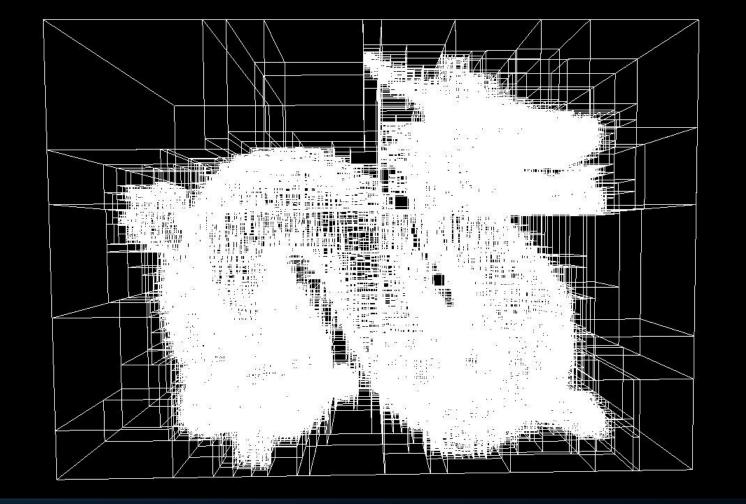
survive = SurvivalProbability( diffuse estimation - doing it properly, close if; radiance = SampleLight( &rand, I, &L,

e.x + radiance.y + radiance.z) > 0) 88

w = 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 following /ive)

; at3 brdf = SampleDiffuse( diffuse, N, rl, r2, 8R, 60; urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true;



### **Surface Area Heuristic**



### Better BVHs

nics & (depth < MADD

it = nt / nc, ddl
ss2t = 1.0f - nm
), N );
))

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

= \* diffuse = true;

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

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

#### AXDEPTH)

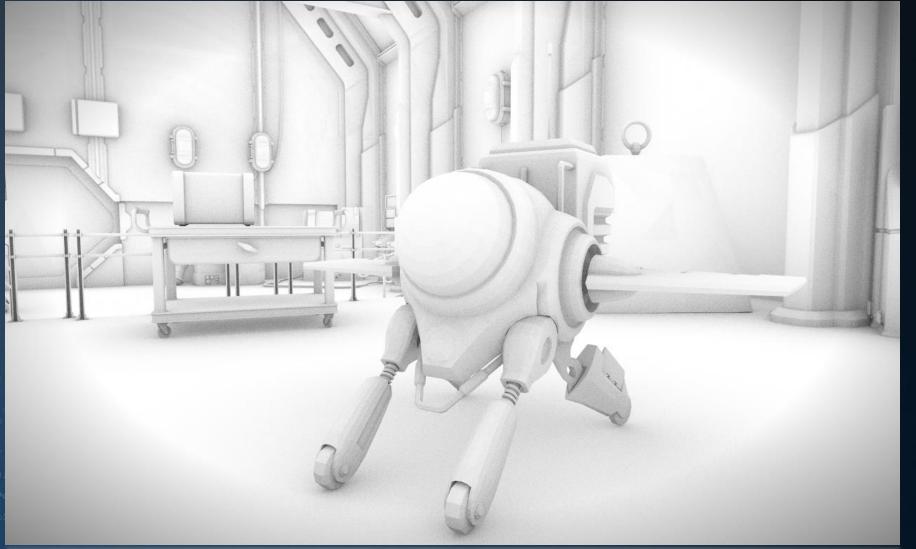
survive = SurvivalProbability( diffu estimation - doing it properly, class if; radiance = SampleLight( &rand, I, &L e.x + radiance.y + radiance.z) > 0) 8

v = true;

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

andom walk - done properly, closely vive)

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





### Better BVHs

nics & (depth < MODS

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

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

= \* diffuse; = true;

efl + refr)) && (depth < MAXDEPT

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

#### AXDEPTH)

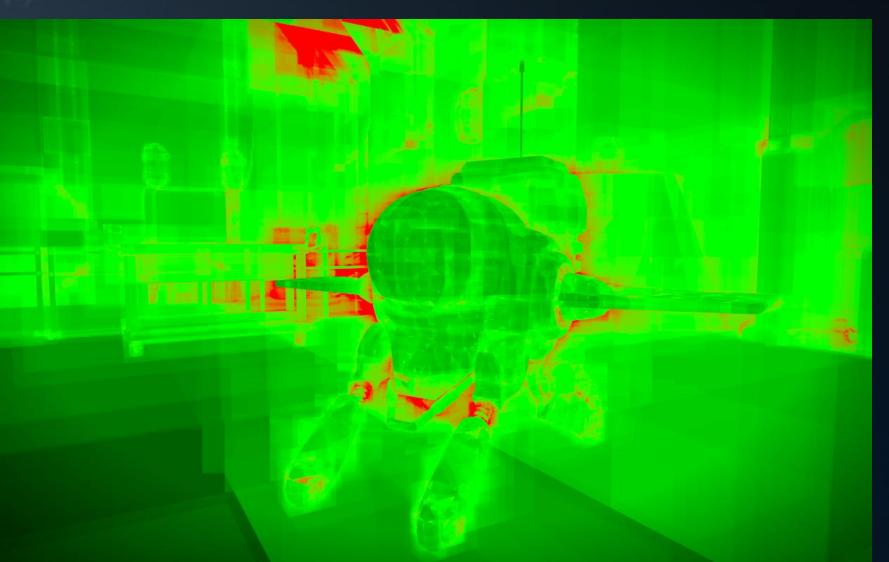
survive = SurvivalProbability( diffuse estimation - doing it properly, close if; radiance = SampleLight( &rand, I, &L, e.x + radiance.y + radiance.z) > 0) &&

v = true;

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

andom walk - done properly, closely fo /ive)

, H33 brdf = SampleDiffuse( diffuse, N, r1, r2, 8R, 800) urvive; .pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:





### Better BVHs

nics & (depth < ⊅0000

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

= \* diffuse; = true;

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

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

#### AXDEPTH)

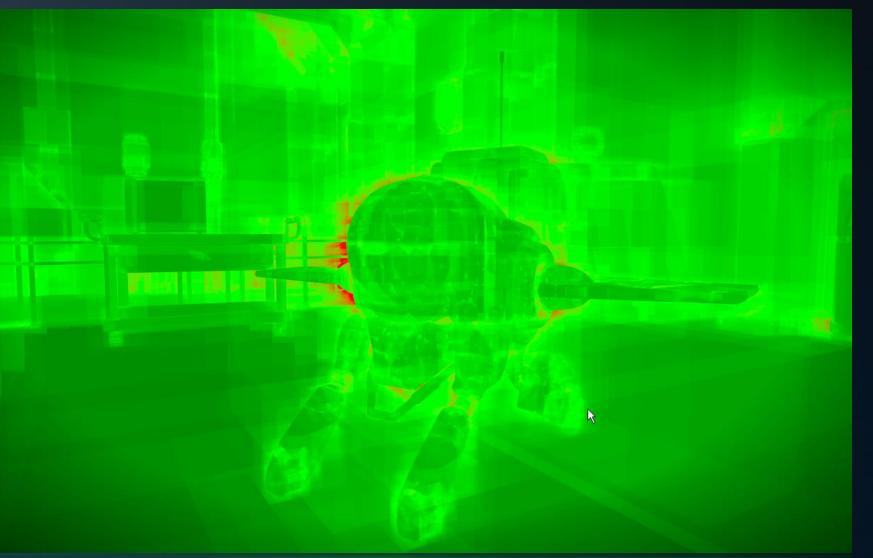
survive = SurvivalProbability( diffuse estimation - doing it properly if; radiance = SampleLight( &rand, I, &L 2.x + radiance.y + radiance.z) > 0) &&

v = true;

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

andom walk - done properly, closely f /ive)

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





tics & (depth < Modeo

: = inside ? l | ... ht = nt / nc, ddn os2t = 1.0f - nnt = n O, N ); 0)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0) r) 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 H; radiance = SampleLight( &rand, I, &L, &light) e.x + radiance.y + radiance.z) > 0) && closed

w = 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:

### Today's Agenda:

- Building Better BVHs
- Refitting
- Fast BVH Construction
- The Top-level BVH



### Refitting

sics & (depth < Monost

: = 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 = (ddn

= \* diffuse; = true;

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

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

AXDEPTH)

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

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

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

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

Summary of BVH Characteristics

A BVH provides significant freedom compared to e.g. a kD-tree:

- No need for a 1-to-1 relation between bounding boxes and primitives
- Bounding boxes may overlap
- Bounding boxes can be altered, as long as they fit in their parent box
- A BVH can be very bad but still valid

Some consequences / opportunities:

- We can rebuild part of a BVH
- We can combine two BVHs into one
- We can *refit* a BVH



### Refitting

nics & (depth < ∧∧∞oo

: = inside ? 1 ( ) ) ht = nt / nc, ddn ss2t = 1.0f - n⊓t \* ∩ 2, N ); ≫)

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

= \* diffuse; = true;

• efl + refr)) && (depth < MAXDEPTH

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

AXDEPTH)

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

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

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

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

### Refitting

Q: What happens to the BVH of a tree model, if we make it bend in the wind?

A: Likely, only bounds will change; the topology of the BVH will be the same (or at least similar) in each frame.

#### Refitting:

*Updating the bounding boxes stored in a BVH to match changed primitive coordinates.* 





### Refitting

sics & (depth < ∧∨∞coo

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

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

= \* diffuse; = true;

• efl + refr)) && (depth < MACD

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

AXDEPTH)

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

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

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

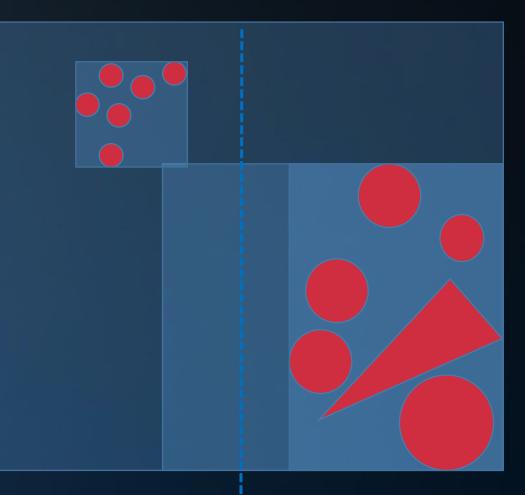
; at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf urvive; pdf; n = E \* brdf \* (dot( N, R ) / pdf); sion = true:

### Refitting

*Updating the bounding boxes stored in a BVH to match changed primitive coordinates.* 

#### Algorithm:

- 1. For each leaf, calculate the bounds over the primitives it represents
- 2. Update parent bounds





### Refitting

### Refitting - Suitability



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

ndom walk - done properly, closely following 30 rive)

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











### Refitting

#### **Refitting – Practical**

nics & (depth k PAXOE)

: = 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) Fr) R = (D \* nnt - N - (dd)

= \* diffuse; = true;

. e**fl + refr)) && (depth < MAXDE** 

D, N ); refl \* E \* diff = 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) (nud

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

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

#### BVH node array

Level 1

#### Root node

0

Therefore:

for( int i = N-1; i >= 0; i-- )
nodeArray[i].AdjustBounds();

Order of nodes in the node array:

at a position greater than X.

We will never find the parent of node X



tics & (depth < Modeo

: = inside ? l | ... ht = nt / nc, ddn os2t = 1.0f - nnt = n O, N ); 0)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0) r) 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 H; radiance = SampleLight( &rand, I, &L, &light) e.x + radiance.y + radiance.z) > 0) && closed

w = 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:

### Today's Agenda:

- Building Better BVHs
- Refitting
- Fast BVH Construction
- The Top-level BVH



### Binning

tics & (depth < NACCO

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

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

= \* 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, &II )
e.x + radiance.y + radiance.z) > 0) &&

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 Sou. /ive)

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

### Rapid BVH Construction

Refitting allows us to update hundreds of thousands of primitives in realtime. But what if topology changes significantly?

Rebuilding a BVH requires 3*NlogN* split plane evaluations.

#### Options:

- Do not use SAH (significantly lower quality BVH)
- 2. Do not evaluate all 3 axes (minor degradation of BVH quality)
- 3. Make split plane selection independent of *N*



### Binning

sics & (depth < ≥0000

t = inside ? 1 1 1 1 ht = nt / nc, ddn os2t = 1.0f - nnt " r 2, N ); ≥)

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

= \* diffuse; = true;

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

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

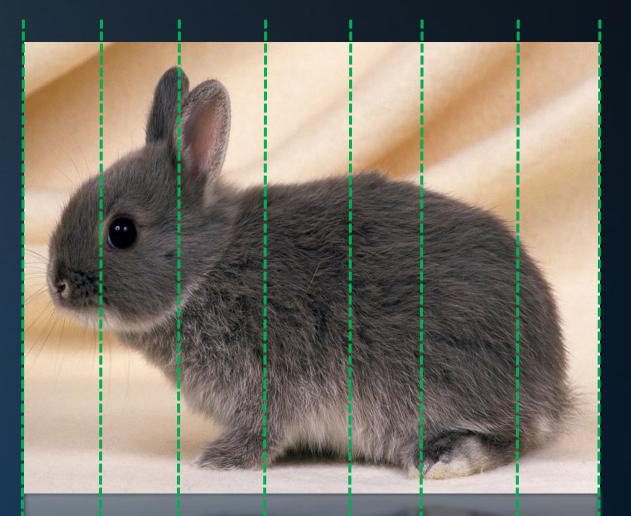
AXDEPTH)

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

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

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

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





**Binned BVH Construction\*** 

Evaluate SAH at N discrete intervals.

**Binned construction:** 

### Binning

nics & (depth < Motoon

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

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

= \* diffuse; = true;

efl + refr)) && (depth < MAXDEPT

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

#### AXDEPTH)

urvive;

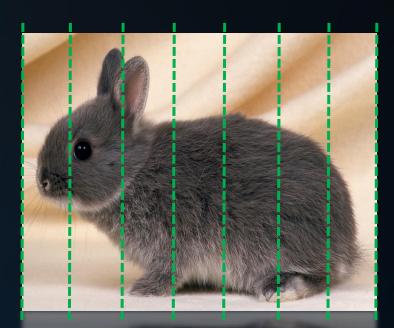
survive = SurvivalProbability( diffuse .estimation - doing it properly, if; radiance = SampleLight( &rand, I, &L, &light) 2.x + radiance.y + radiance.z) > 0) && (doing)

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

andom walk - done properly, closely following 30 /ive)

#### tt3 brdf = SampleDiffuse( diffuse, N, r1, r2, \*: On fast Construction of SAH-based Bounding Volume Hierarchies, Wald, 2007

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





### Binning

tics & (depth < ™00000

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

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

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

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

#### **Binned BVH Construction**

Performance evaluation:

472ms 7.88M triangles (12 cores @ 2Ghz)\*.





\*: Parallel BVH Construction using Progressive Hierarchical Refinement, Henrich et al., 2016.

tics & (depth < Modeo

: = inside ? l | ... ht = nt / nc, ddn os2t = 1.0f - nnt = n O, N ); 0)

at a = nt - nc, b = nt - nc at Tr = 1 - (R0 + (1 - R0) r) 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 H; radiance = SampleLight( &rand, I, &L, &light) e.x + radiance.y + radiance.z) > 0) && closed

w = 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:

### Today's Agenda:

- Building Better BVHs
- Refitting
- Fast BVH Construction
- The Top-level BVH



### Top-level BVH

ics & (depth < Mocos

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

at a = nt - nc, b = nt = n at Tr = 1 - (R0 + (1 - R0)  $\Gamma$ r) R = (D  $^{+}$  nnt - N  $^{-}$  (ddn

= \* diffuse = true;

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

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

#### AXDEPTH)

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

v = true;

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

andom walk - done properly, closely follow /ive)

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



CHARLINGUETO.



### Top-level BVH

ics ≰(depth < Notici

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 ≪ MAXDED⊺

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

#### AXDEPTH)

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

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

andom walk - done properly, closely followi /ive)

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







### Combining BVHs

sics & (depth < MODEF

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

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

= \* diffuse = true;

efl + refr)) && (depth < MANDEPTI

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

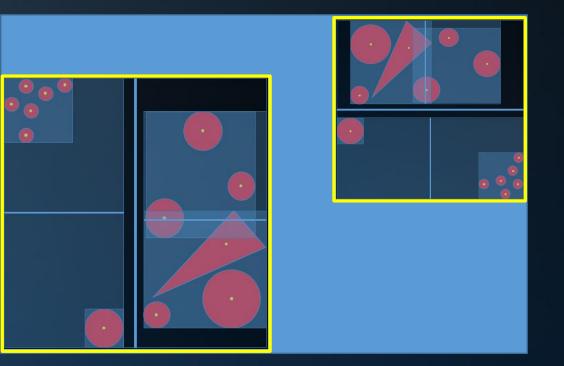
AXDEPTH)

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

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) (rad

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

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





## Top-level BVH

sics & (depth < Modean

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 \* diffuse; = true;

#### AXDEPTH)

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

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

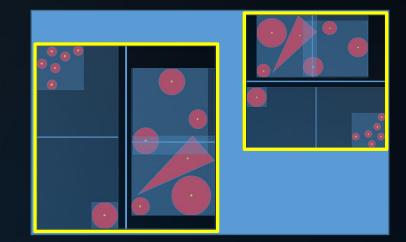
andom walk - done properly, closely following Sec. /ive)

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

### Combining BVHs

Two BVHs can be combined into a single BVH, by simply adding a new root node pointing to the two BVHs.

- This works regardless of the method used to build each BVH
- This can be applied repeatedly to combine many BVHs







## Top-level BVH

### Scene Graph

AXDEPTH)

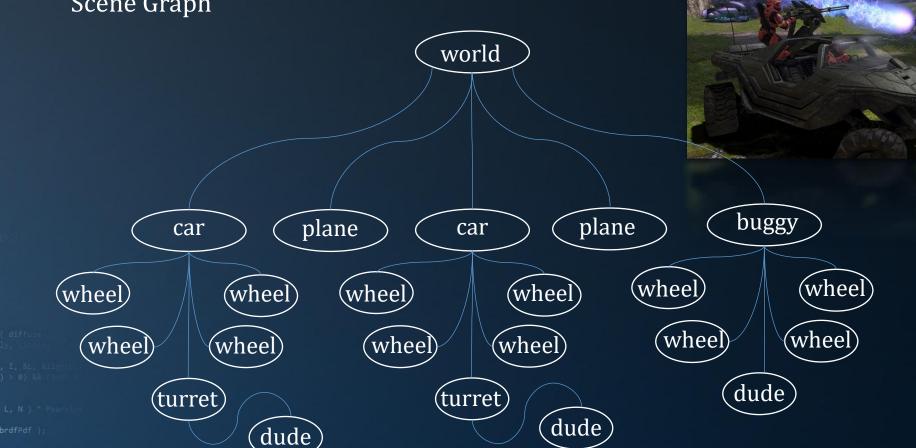
survive = SurvivalProbability( diff radiance = SampleLight( &rand, I, &L

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

E \* ((weight \* cosThetaOut) / directPdf)

andom walk - done properly, closely follow /ive)

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





## Top-level BVH

sics & (depth < Motion

t = inside 7 1 1 1 0 ht = nt / nc, ddn 0 0 bs2t = 1.0f - n⊓t 0 0 2, N ); ð)

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

= \* diffuse; = true;

efl + refr)) && (depth < MAXD

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

AXDEPTH)

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

v = 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) (Psi

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

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

### Scene Graph

If our application uses a scene graph, we can construct a BVH for each scene graph node.

The BVH for each node is built using an appropriate construction algorithm:

- High-quality SBVH for static scenery (offline)
- Fast binned SAH BVHs for dynamic scenery

The extra nodes used to combine these BVHs into a single BVH are known as the *Top-level BVH*.





42

## Top-level BVH

nics & (depth < Motoor

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

at a = nt - nc, b = nt - r at Tr = 1 - (R0 + (1 - R0 Fr) R = (D <sup>#</sup> nnt - N <sup>-</sup> (dd

= \* diffuse; = true;

efl + refr)) && (depth < MODEPTI

D, 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) \* (rac

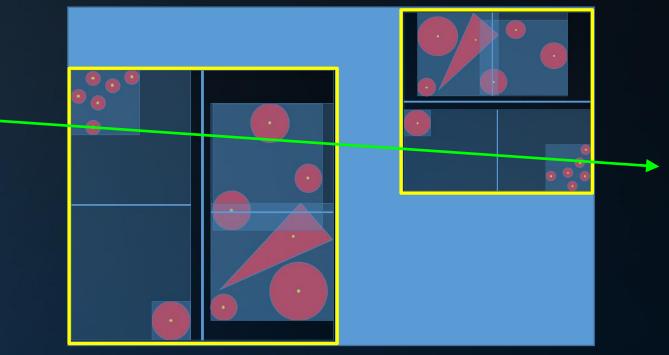
andom walk - done properly, closely following Small /ive)

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

### **Rigid Motion**

Applying rigid motion to a BVH:

- 1. Refit the top-level BVH
- 2. Refit the affected BVH





## Top-level BVH

at a = nt

), N );

AXDEPTH)

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

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

andom walk - done properly, closely fo /ive)

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

### **Rigid Motion**

Applying rigid motion to a BVH:

- Refit the top-level BVH
- Refit the affected BVH 2.

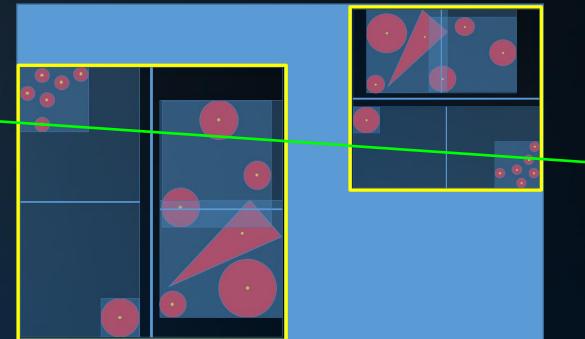
or:

#### 2. Transform the ray, not the node

Rigid motion is achieved by transforming the rays by the *inverse transform* upon entering the sub-BVH.

(this obviously does not only apply to translation)







nics & (depth < >>0000

: = inside } 1 ht = nt / nc, ddn os2t = 1.0f - nnt = c O, N ); 0)

at a = nt - nc, b = nt at Tr = 1 - (R0 + (1 - R0  $\Gamma$ r) 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 if; radiance = SampleLight( &rand, I, &L, %I) 2.x + radiance.y + radiance.z) > 0) && (close 2.x + radiance.z) && (close 2.x + radiance.z)

v = true;

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

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

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

### The Top-level BVH - Construction

Input: *list of axis aligned bounding boxes for transformed scene graph nodes* 

#### Algorithm:

- 1. Find the two elements in the list for which the AABB has the smallest surface area
- 2. Create a parent node for these elements
- 3. Replace the two elements in the list by the parent node
- 4. Repeat until one element remains in the list.

### Note: algorithmic complexity is $O(N^3)$ .



ics & (depth < 2000€

: = inside ? 1 1 1 3 ht = nt / nc, ddn 4 os2t = 1.0f - nmt 7 nm 0, N ); 0)

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

= \* diffuse; = true;

efl + refr)) && (depth < ⊨

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

AXDEPTH)

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

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

#### The Top-level BVH – Faster Construction\*

#### Algorithm:

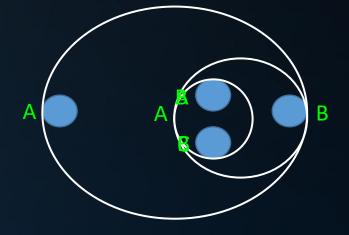
Node A = list.GetFirst();
Node B = list.FindBestMatch( A );
while (list.size() > 1)

Node C = list.FindBestMatch( B );
if (A == C)

list.Remove( A ); list.Remove( B ); A = new Node( A, B ); list.Add( A ); B = list.FindBestMatch( A );

else A = B, B = C;

\*: Fast Agglomerative Clustering for Rendering, Walter et al., 2008





The Top-level BVH – Traversal

The leafs of the top-level BVH contain the sub-BVHs.

When a ray intersects such a leaf, it is transformed by the inverted transform matrix of the sub-BVH. After this, it traverses the sub-BVH.

Once the sub-BVH has been traversed, we transform the ray again, this time by the transform matrix of the sub-BVH.

For efficiency, we store the inverted matrix with the sub-BVH root.

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

at a = nt

), N );

AXDEPTH)

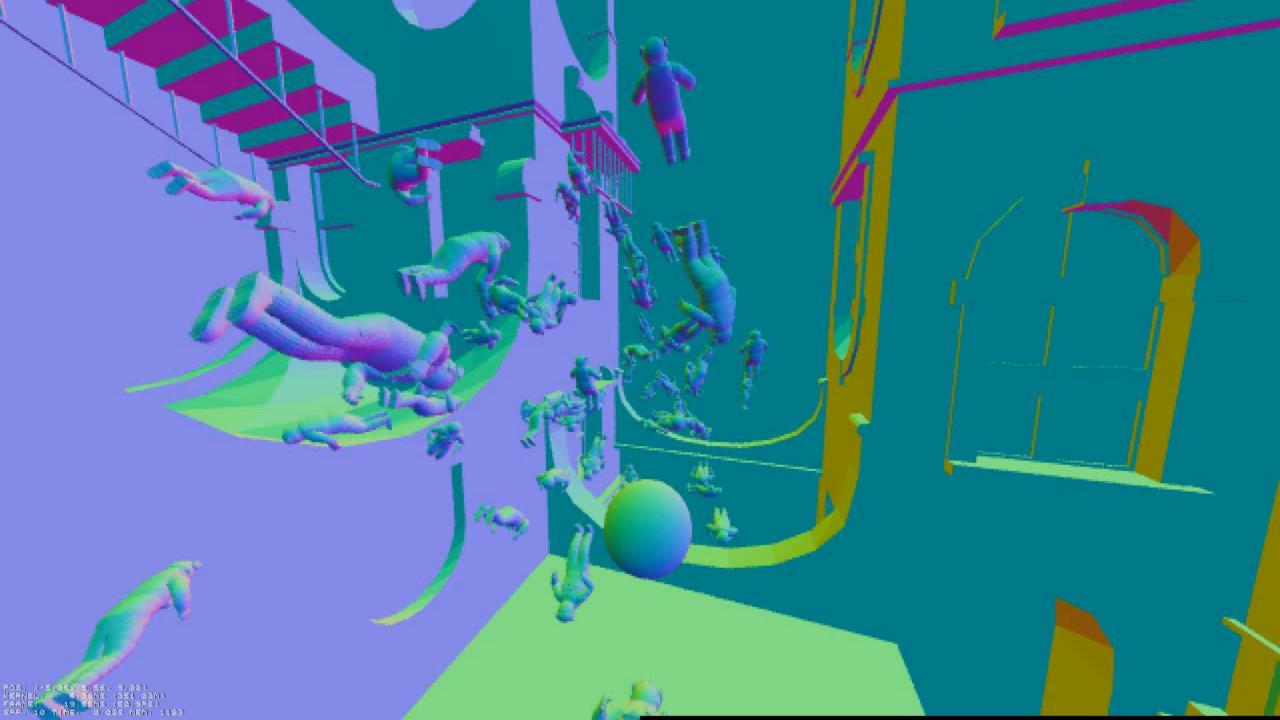
refl \* E \* diffuse;

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) (psurviv)

andom walk - done properly, closely following 3000 /ive)

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





The Top-level BVH – Summary

The top-level BVH enables complex animated scenes:

- for static objects, it contains high-quality sub-BVHs;
- for objects undergoing rigid motion, it also contains high-quality sub-BVHs, with a transform matrix and its inverse;
- for deforming objects, it contains sub-BVHs that can be refitted;
- for arbitrary animations, it contains lower quality sub-BVHs.

Combined, this allows for efficient maintenance of a global BVH.

survive = SurvivalProbability( diff

at a = nt

), N );

AXDEPTH)

if; radiance = SampleLight( &rand, I, &L, &li 2.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)

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:

# We share a start way a start w

hics & (depth < Not00⊓

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

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

= \* diffuse; = true;

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

D, N ); refl \* E \* diffuse; = 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 ) Pourvive 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:

## **INFOMAGR – Advanced Graphics**

Jacco Bikker - November 2021 - February 2022

## END of "The Perfect BVH"

next lecture: "Path Tracing"

