Ray Tracing for Games

Dr. Jacco Bikker - IGAD/BUAS, Breda, February 7

Welcome!

11

Ray Tracing for Games

Wednesday
13:00 – 17:00

course intro
LH2
template
Whitted
refactoring
RT-centric games

LAB 1

Thursday
09:00 – 14:00

advanced Whitted
audio, AI & physics
faster Whitted
Heaven7

LAB 2

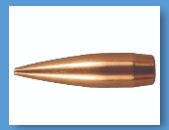
work @ home

End result day 2:

A solid Whitted-style ray tracer, as a basis for subsequent work.

Friday
09:00 – 17:00

optimization
profiling, rules of engagement
threading



LAB 3

SIMD
applied SIMD
SIMD triangle
SIMD AABB

LAB 4

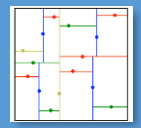
End result day 3:

A 5x faster tracer.

GLOBAL GAME JAM

Monday
09:00 – 17:00

acceleration
grid, BVH, kD-tree
SAH
binning



LAB 5

refitting
top-level BVH
threaded building

LAB 6

End result day 4:

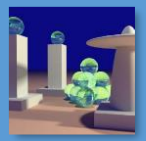
A real-time tracer.

Tuesday
09:00 – 17:00

Monte-Carlo
Cook-style
glossy, AA
area lights, DOF

LAB 7

path tracing



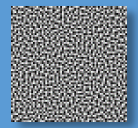
LAB 8

End result day 5:

Cook or Kajiya.

Thursday
09:00 – 17:00

random numbers
stratification
blue noise



LAB 9

importance
sampling
next event
estimation

LAB 10

End result day 6:

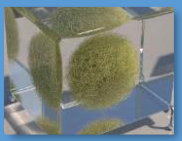
Efficiency.

Friday
09:00 – 17:00

future work

LAB 11

path guiding



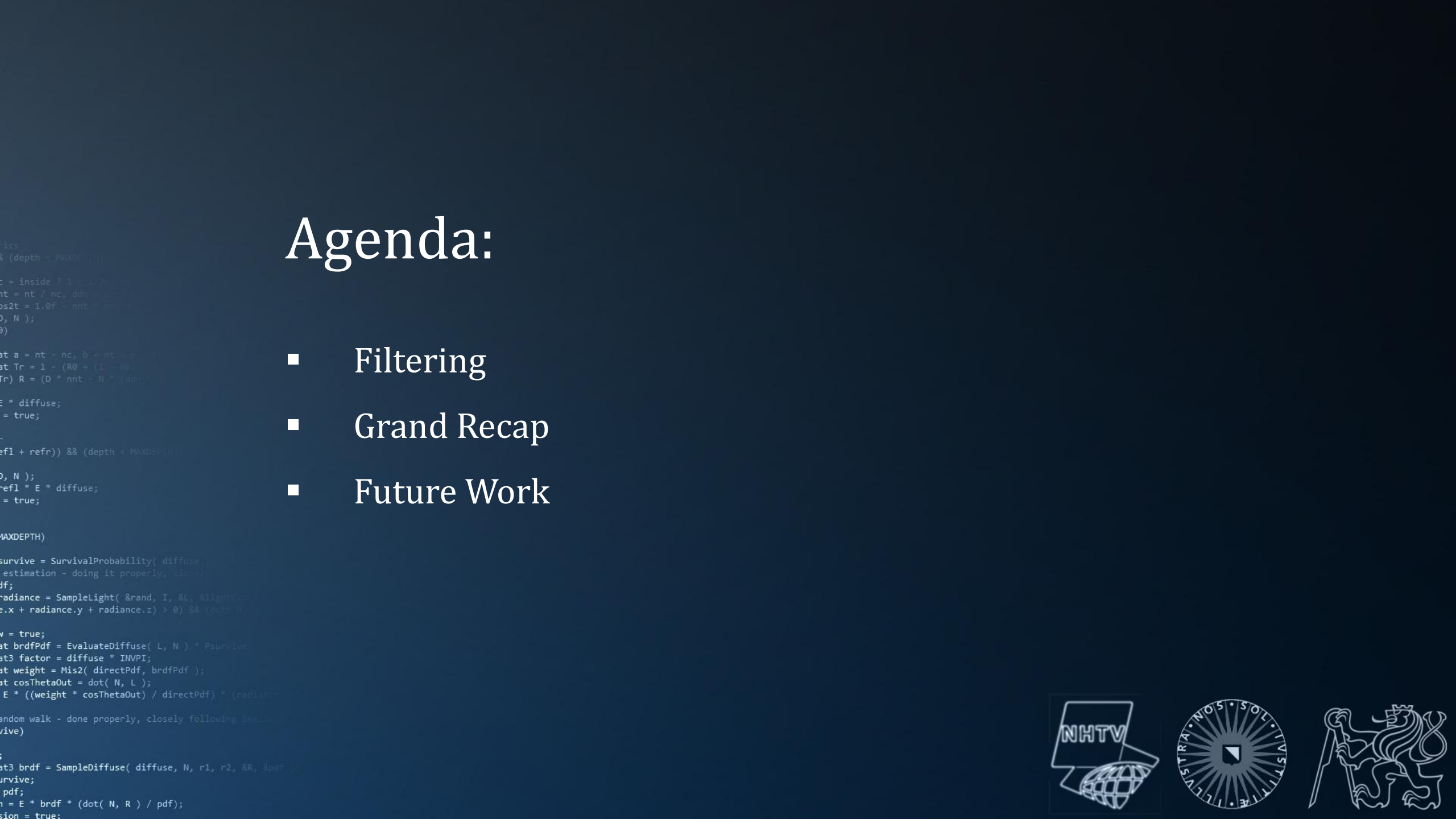
LAB 10

End result day 6:

Great product.

Agenda:

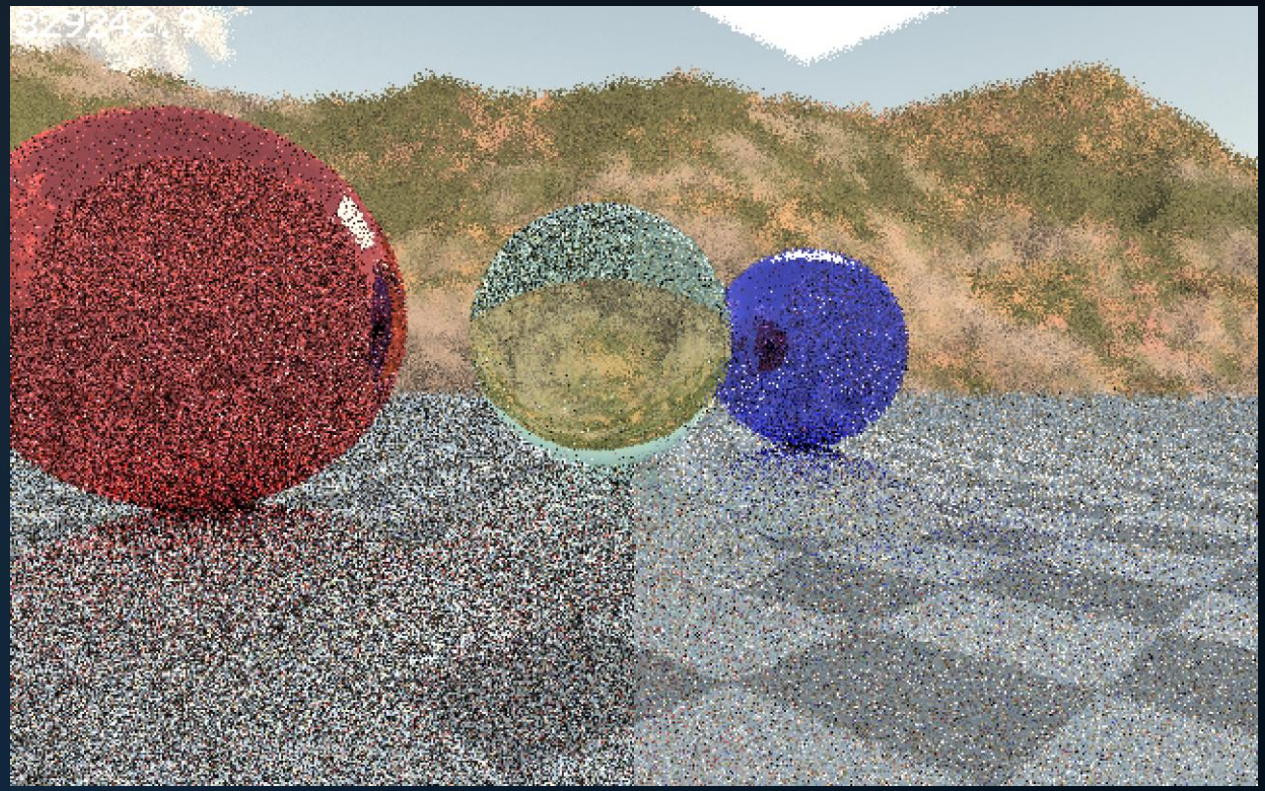
- Filtering
- Grand Recap
- Future Work



Reducing Noise in Path Tracing

We can increase ‘exposure’ in a pathtracer by taking more samples.

Or we can use Importance Sampling.



Ray Tracing for Games

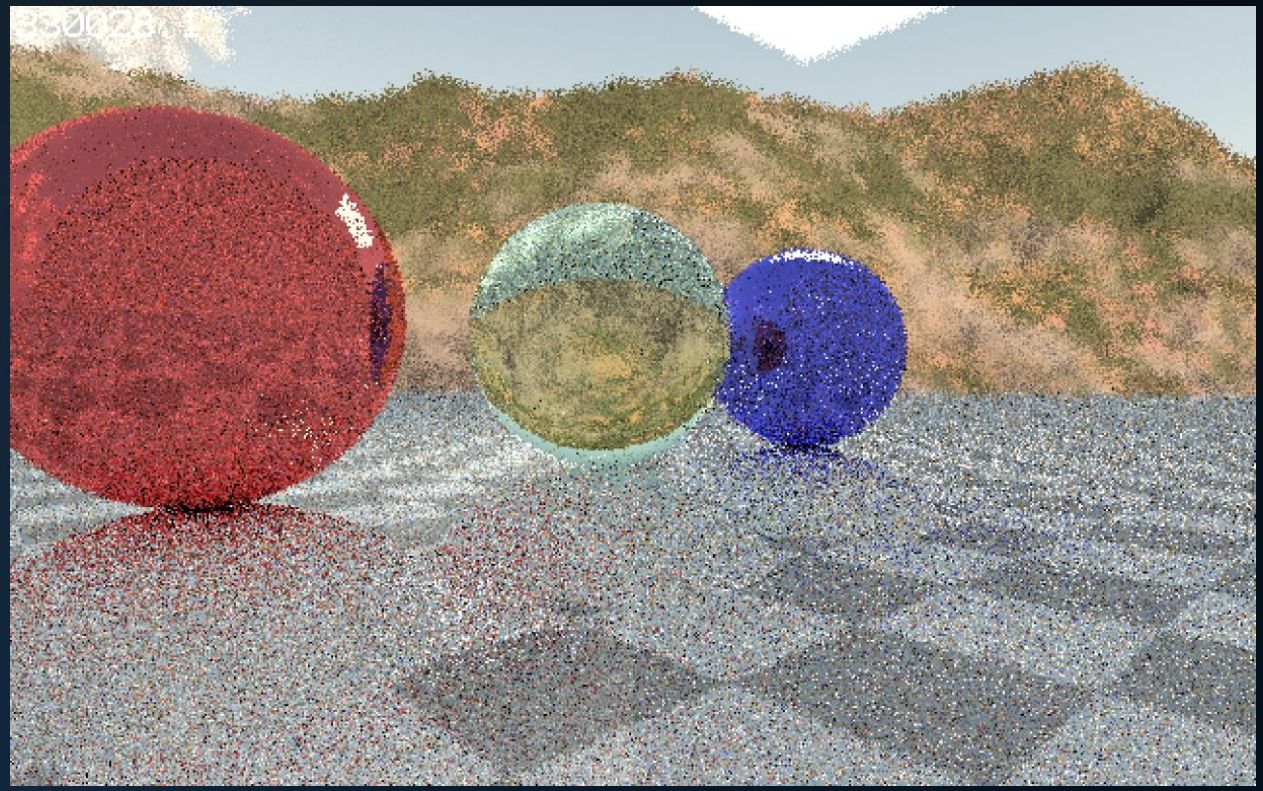
Reducing Noise in Path Tracing

We can increase ‘exposure’ in a pathtracer by taking more samples.

Or we can use Importance Sampling.

Or Next Event Estimation.

However, at interactive rates it will be quite hard to reach a converged image.



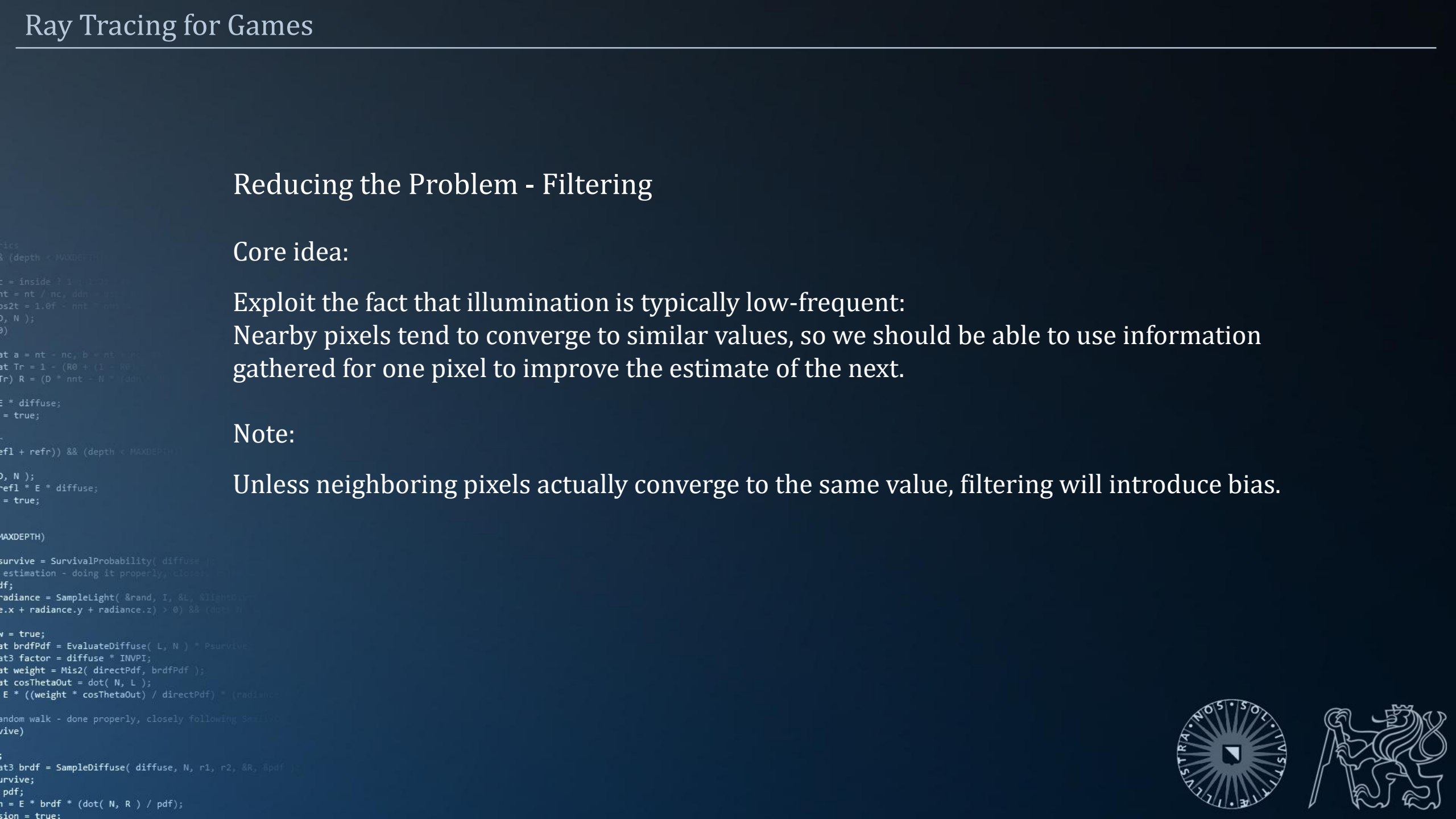
Reducing the Problem - Filtering

Core idea:

Exploit the fact that illumination is typically low-frequent:
Nearby pixels tend to converge to similar values, so we should be able to use information gathered for one pixel to improve the estimate of the next.

Note:

Unless neighboring pixels actually converge to the same value, filtering will introduce bias.



Filter kernels

For the actual filtering, we apply a kernel.

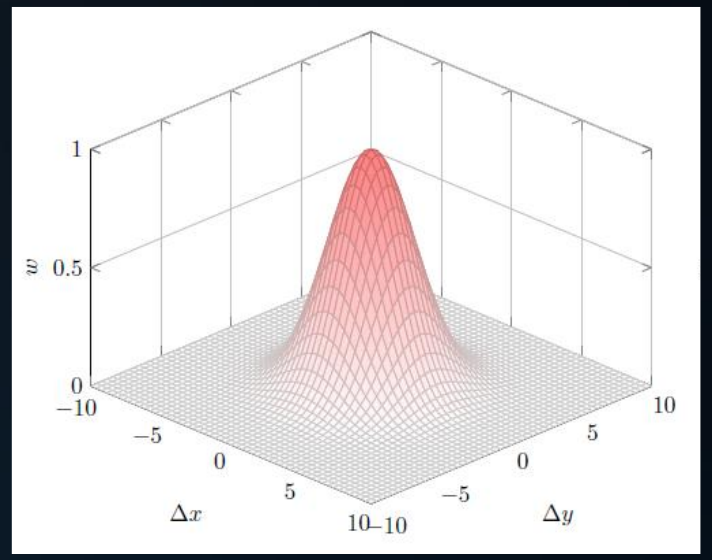
```
Pixel FilteredValue( i_x, i_y, halfWidth )
    sum = 0
    summedWeight = 0
    for j_x = i_x - halfWidth to i_x + halfWidth
        for j_y = i_y - halfWidth to i_y + halfWidth
            sum += ReadPixel( j_x, j_y ) * weight( j_x, j_y )
            summedWeight += weight( j_x, j_y )
    return sum / summedWeight
```

$$\hat{c}_i = \frac{\sum_{j \in \mathcal{N}_i} c_j w(i,j)}{\sum_{j \in \mathcal{N}_i} w(i,j)}$$

Filter kernels

For the actual filtering, we apply a kernel.

```
Pixel FilteredValue( i_x, i_y, halfWidth )
    sum = 0
    summedWeight = 0
    for j_x = i_x - halfWidth to i_x + halfWidth
        for j_y = i_y - halfWidth to i_y + halfWidth
            sum += ReadPixel( j_x, j_y ) * weight( j_x, j_y )
            summedWeight += weight( j_x, j_y )
    return sum / summedWeight
```



Here, **`weight`** or w is the weight function. We could simply use the Gaussian kernel:

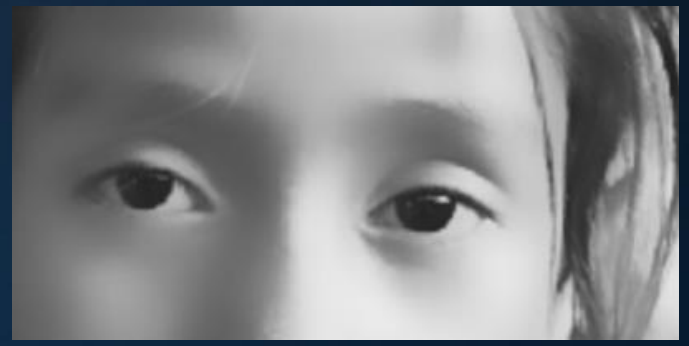
$$w(i,j) = \exp\left(\frac{-\|p_i - p_j\|^2}{2\sigma_d^2}\right),$$

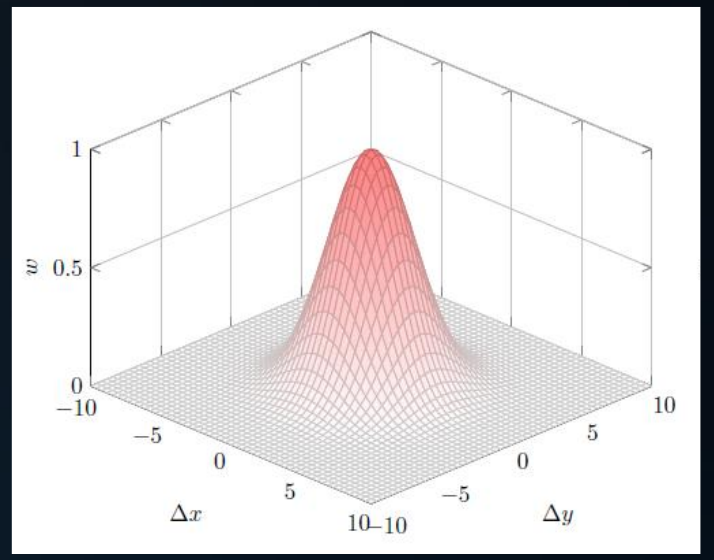
Filter kernels

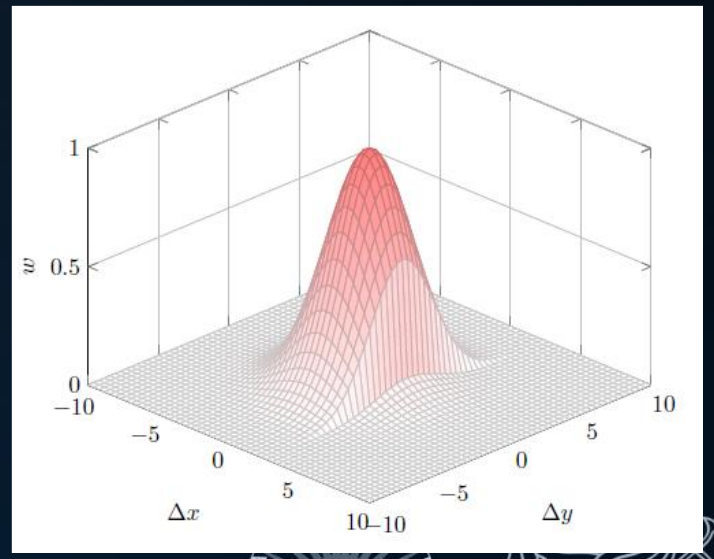
A Gaussian filter (as well as other low-pass filters) blurs out high frequency details.

We can improve on this using a non-linear bilateral filter*.

$$w(i,j) = \exp\left(\frac{-\|p_i - p_j\|^2}{2\sigma_d^2}\right) \times \exp\left(\frac{-\|c_i - c_j\|^2}{2\sigma_r^2}\right)$$







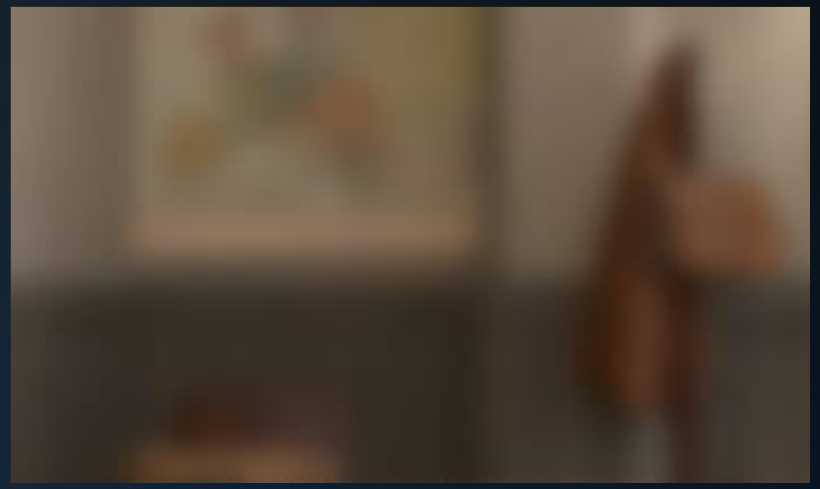
*: Tomasi & Manduchi, Bilateral filtering for gray and color images. ICCV '98.

Ray Tracing for Games

Adding this separation to an existing renderer:

- store albedo at the primary intersection (simple material property);
- at the end of the pipeline: illumination = sample / max(epsilon, albedo).

Ray Tracing for Games



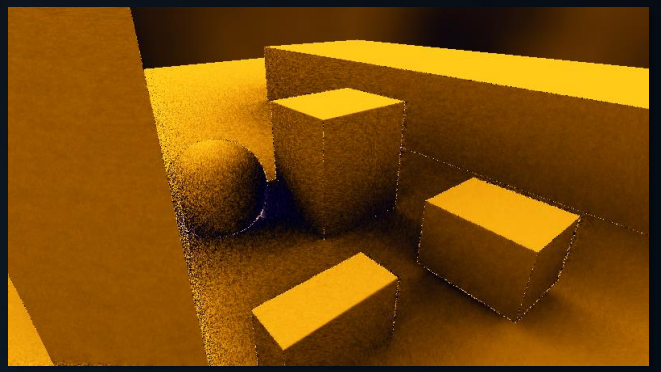
Example of denoising using a cross-bilateral filter in Blender.

sinmantyx.wordpress.com/2015/12/02/denoising-cycles-renders-in-blender-theory

Reprojection

Core idea:

In an animation, samples taken for the previous frame are meaningful for the current frame. *We can supply the filter with more data by looking back in time.*



Reprojection

Core idea:

In an animation, samples taken for the previous frame are meaningful for the current frame. *We can supply the filter with more data by looking back in time.*

Problem: in an animation, the camera and/or the geometry moves. We need to find the location of a pixel in the previous frame(s).

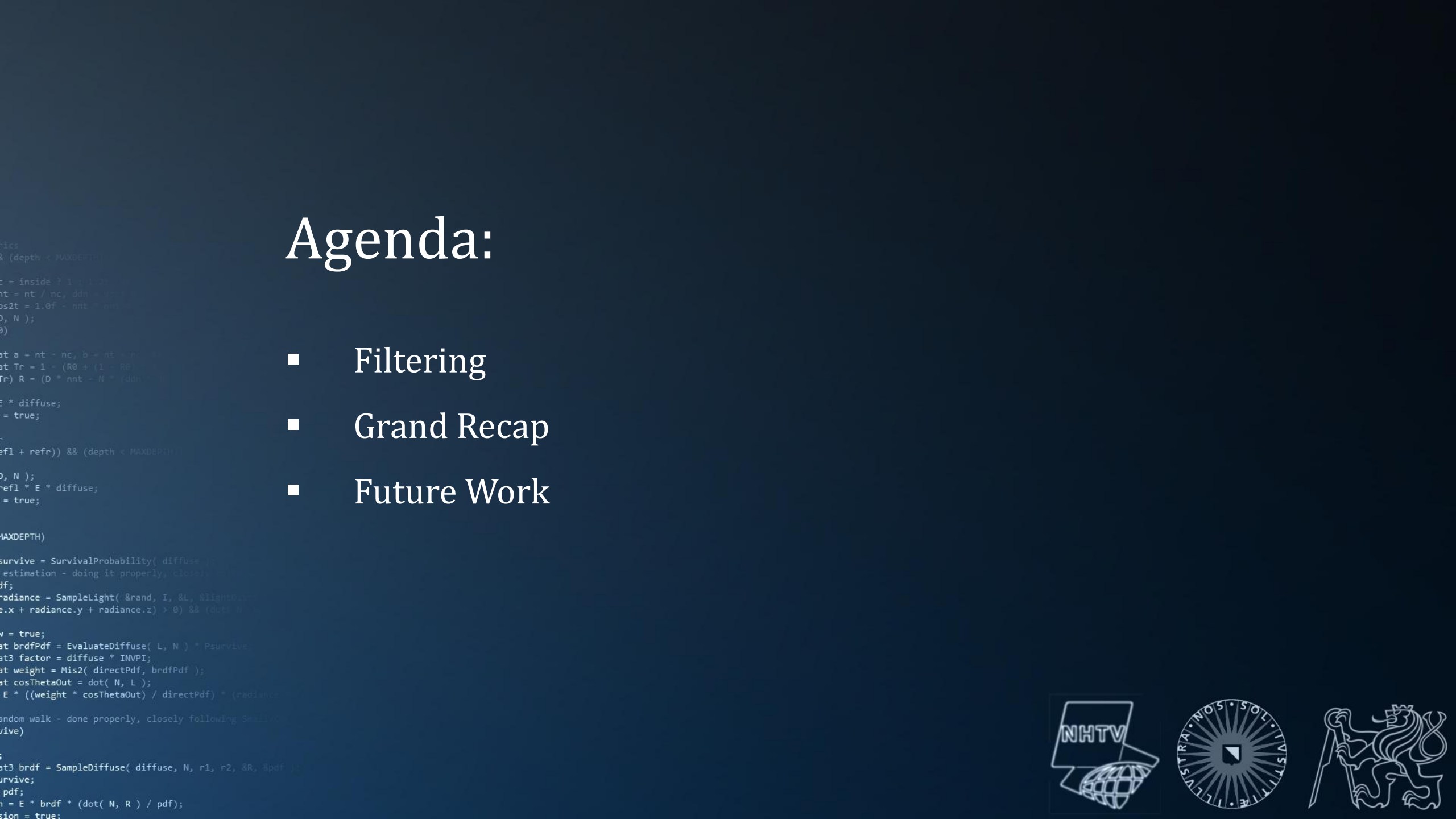
Solution: use the camera matrices.

$$M_{4x4}\begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{pmatrix} = \begin{pmatrix} x_{screen} \\ y_{screen} \\ z_{screen} \\ 1 \end{pmatrix} \quad \rightarrow \quad M_{4x4}^{-1}\begin{pmatrix} x_{screen} \\ y_{screen} \\ z_{screen} \\ 1 \end{pmatrix} = \begin{pmatrix} x_{world} \\ y_{world} \\ z_{world} \\ 1 \end{pmatrix}$$

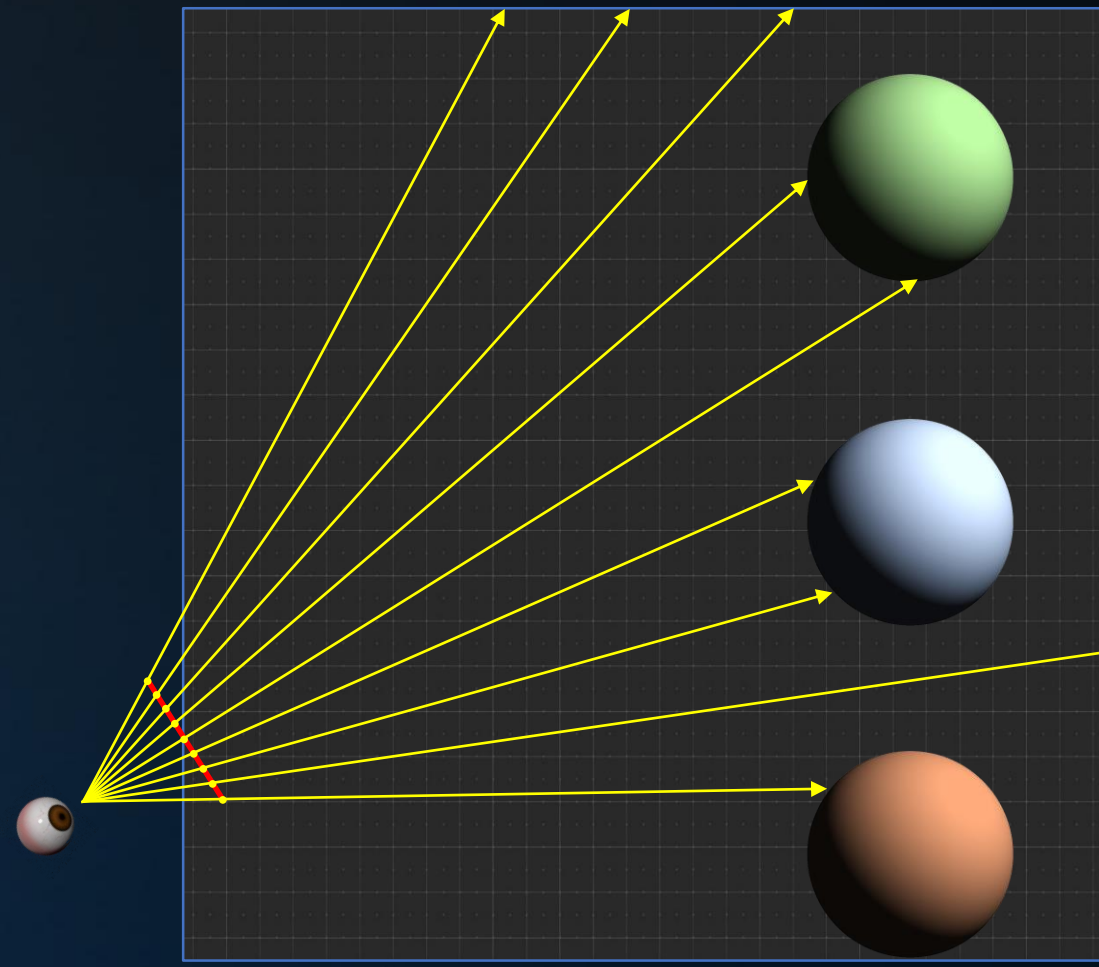
(finally, apply the matrix of the previous frame to obtain the screen location in the previous frame.)

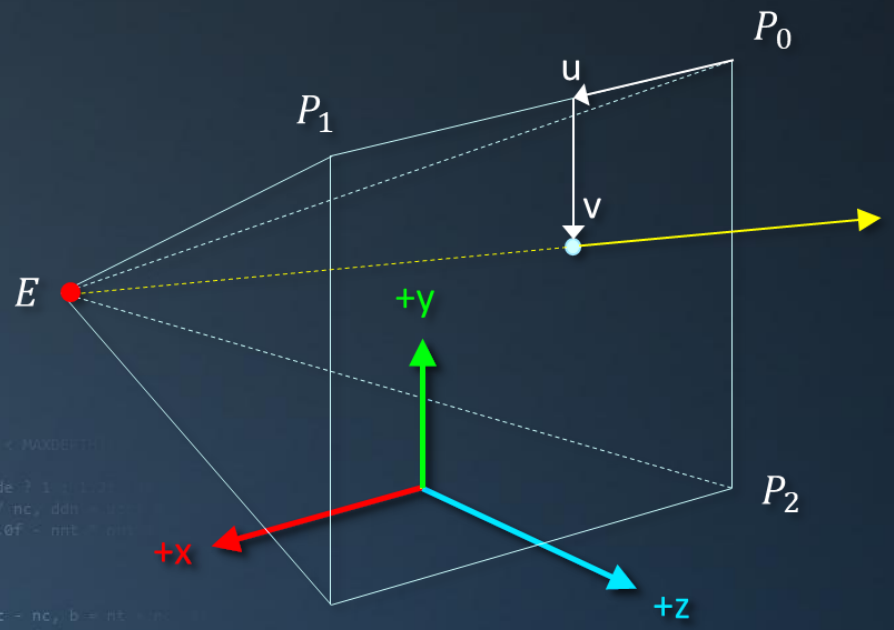
Agenda:

- Filtering
- Grand Recap
- Future Work

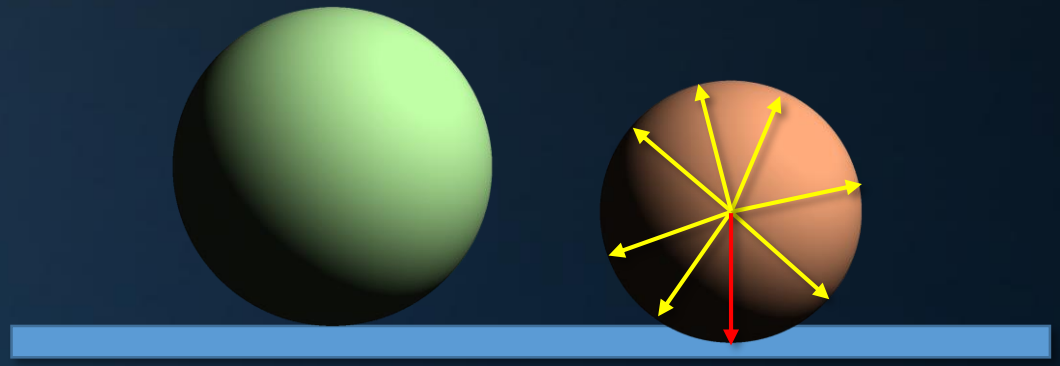



```
 E * ((weight * cosThetaOut) / directPdf) * (radiance
andom walk - done properly, closely following Se
vive)
;
at3 brdf = SampleDiffuse( diffuse, N, r1, r2, &R, &pdf
urvive;
 pdf;
 = E * brdf * (dot( N, R ) / pdf);
sion = true;
```

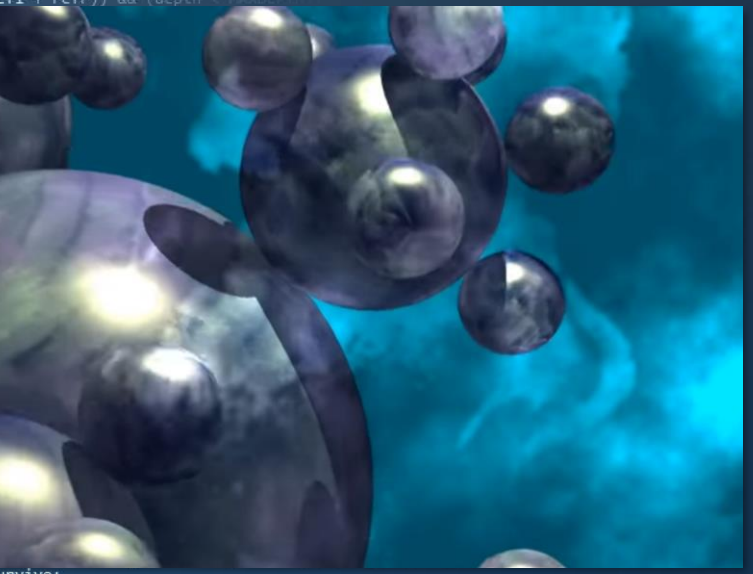


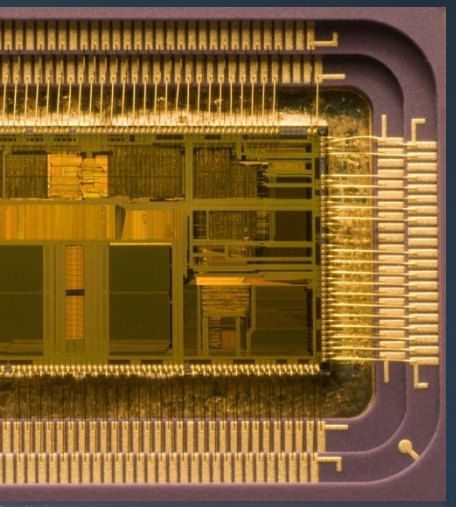


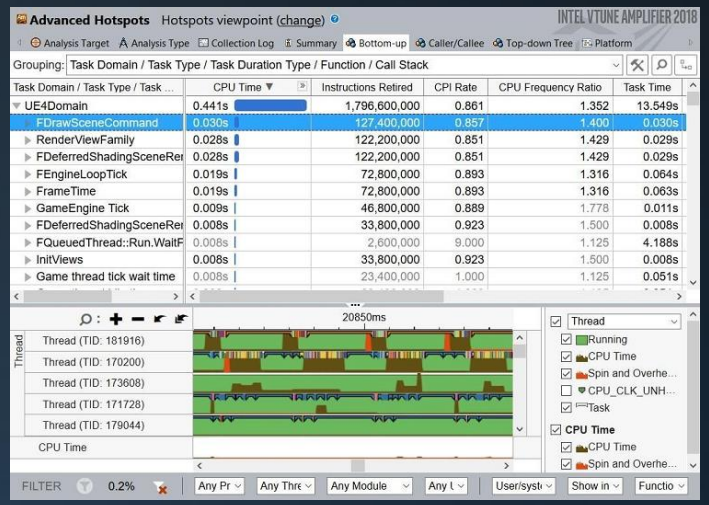
```
Color DirectIllumination(
    Color sum = BLACK
    for each light l
        L = light.pos - I
        dist2 = L.squareLeng
        if (!IsOccluded( I,
            normalize(L)
            sum += (light.col
    return sum
```

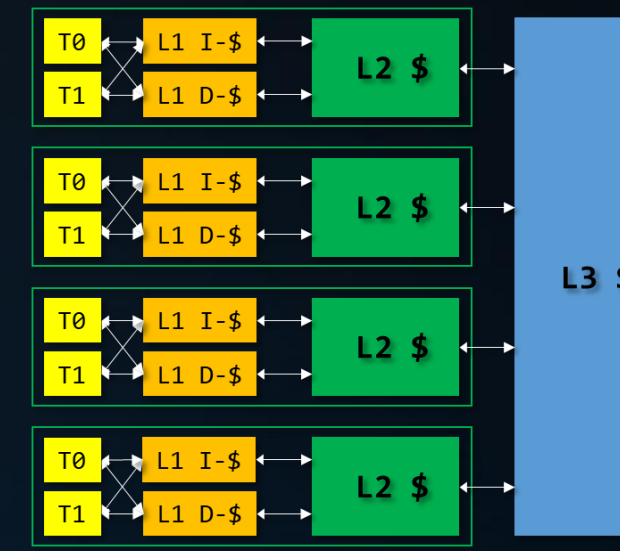


```
Color Trace( ray r )
    I, $\vec{N}$, mat = NearestInte
    if (mat == DIFFUSE)
        return mat.color * D
    if (mat == MIRROR)
        return mat.color * T
```









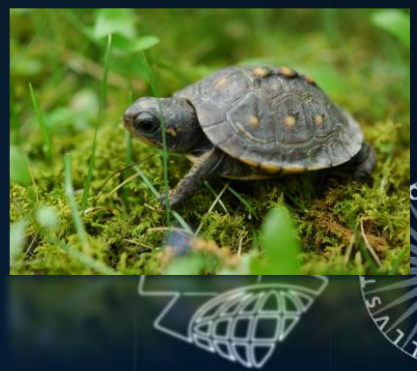
What is the 'cost' of a multiply?

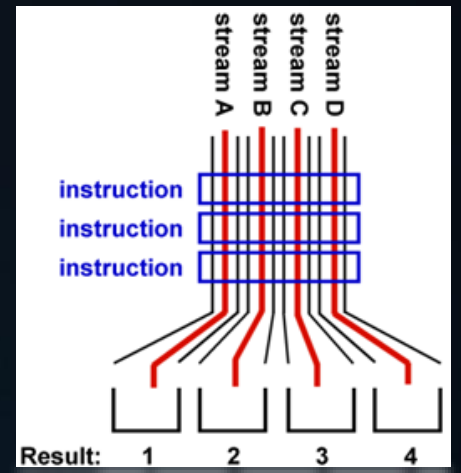
```
float x = 0, y = 0.1f;
unsigned int i = 0, j = 0x28929227;
for ( int k = 0; k < ITERATIONS; k++ )
{
	x += y, y *= 1.01f;
	i += j, j ^= 0x17737352, i >>= 1, j /= 28763;
	if (with) x *= y;
	i += j, j ^= 0x17737352, i >>= 1, j /= 28763;
	my = x + (float)i;
}
```

```
fldz
xor ecx, ecx
fld dword ptr ds:[405290h]
mov edx, 28929227h
fld dword ptr ds:[40528Ch]
push esi
mov esi, 0C350h        = 50000
```

```
add ecx, edx
mov eax, 91D2A969h
xor edx, 17737352h
shr ecx, 1
mul eax, edx
fld st(1)
faddp st(3), st

mov eax, 91D2A969h
shr edx, 0Eh
add ecx, edx
fmul st(1),st
xor edx, 17737352h
shr ecx, 1
mul eax, edx
shr edx, 0Eh
dec esi
jne tobetimed<0>+1Fh
```

$= \frac{2^{46}}{28763}$ (!!)





SIMD Friendly Data Layout

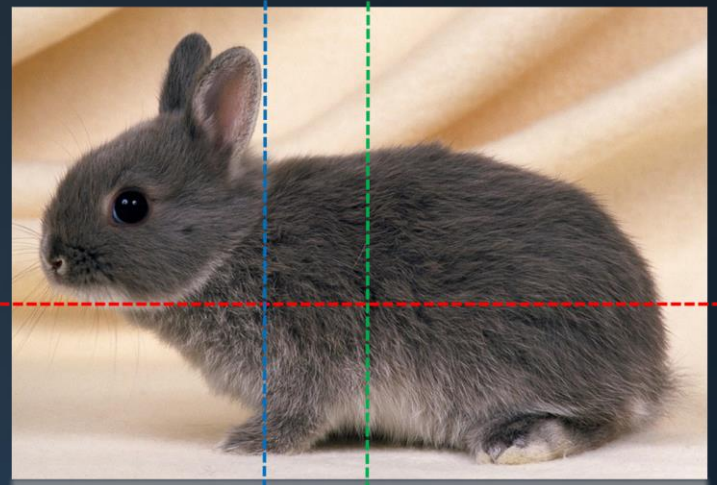
Consider the following data structure:

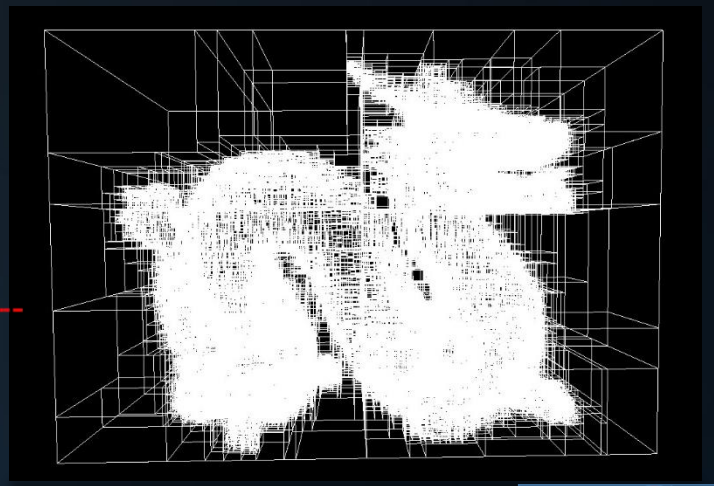
```
struct Particle
{
   float x, y, z;
   int mass;
};
Particle particle[512];
```

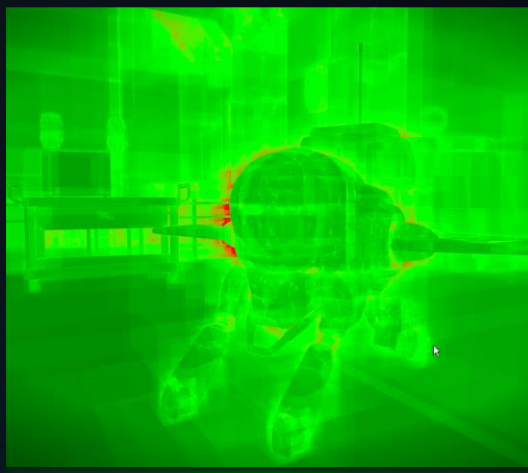
AoS

SoA

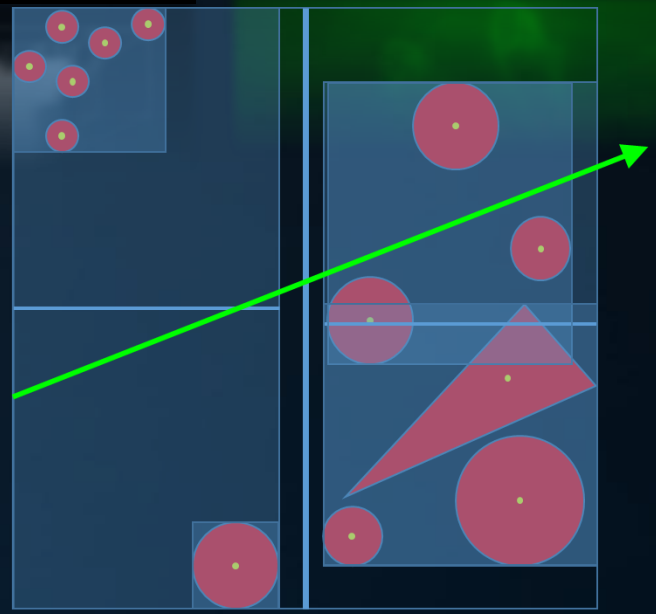
```
union { float x[512];  __m128 x4[128]; };
union { float y[512];  __m128 y4[128]; };
union { float z[512];  __m128 z4[128]; };
union { int mass[512]; __m128i mass4[128]; };
```

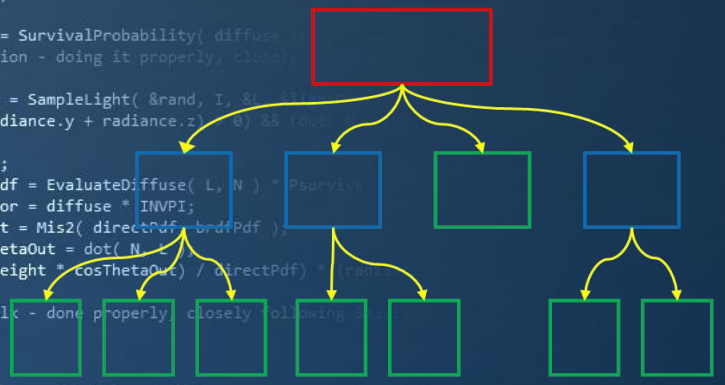


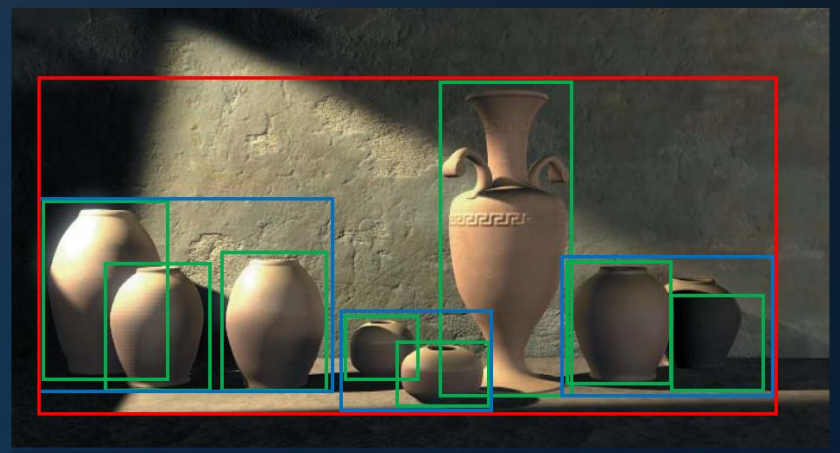


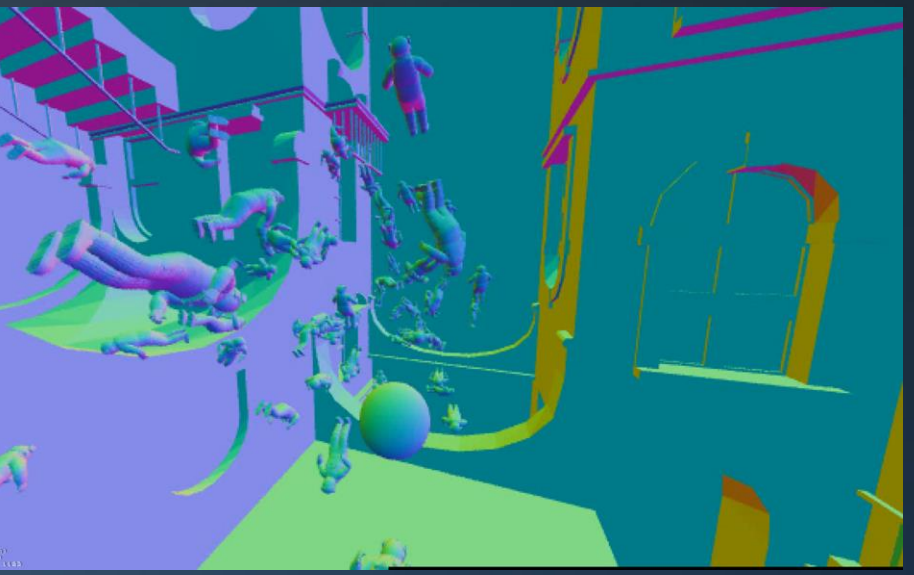


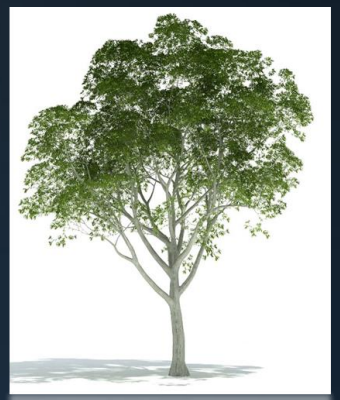
```
struct BVHNode
{
    AABB bounds;        // 24 bytes
    int leftFirst;      // 4 bytes
    int count;          // 4 bytes, total 32  ☺
};
```

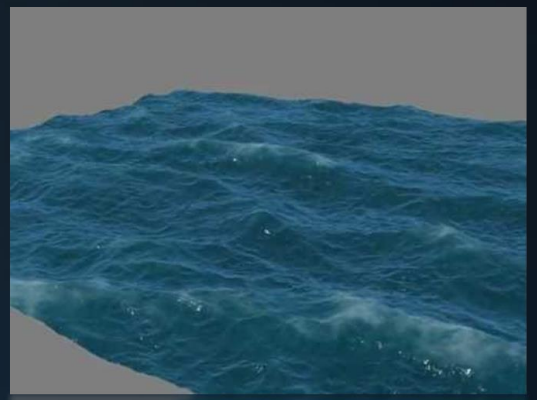


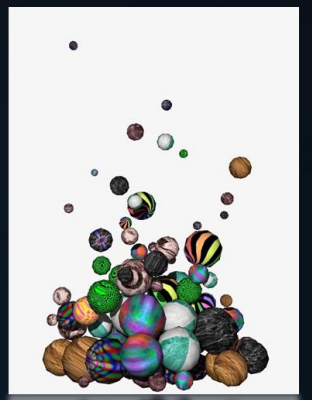


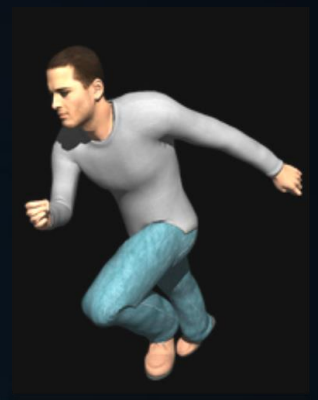




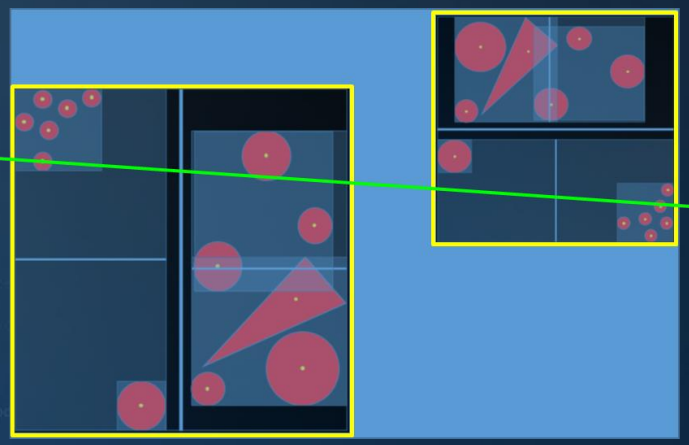


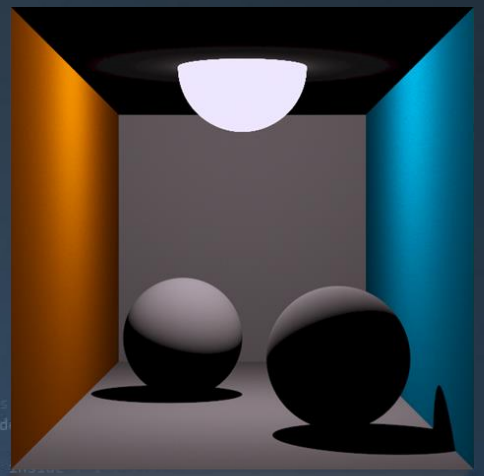
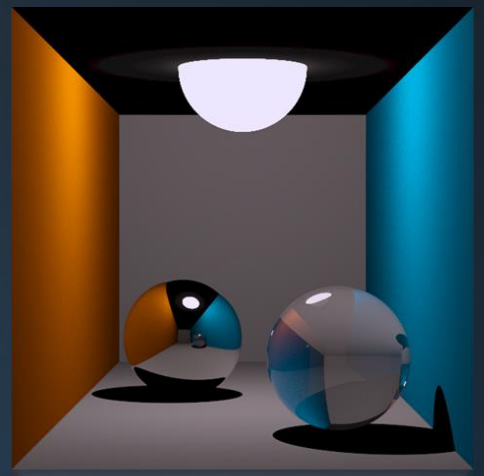
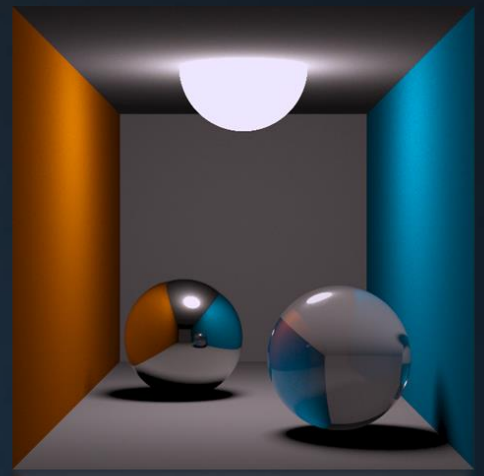
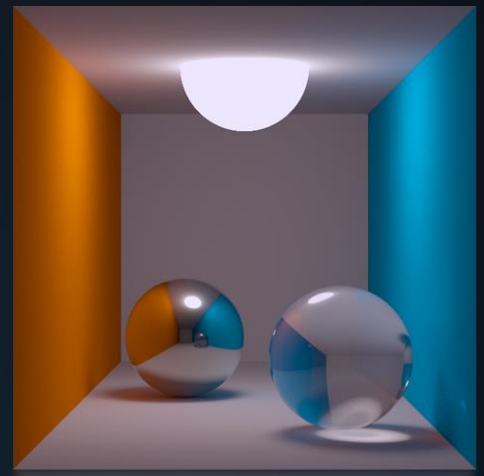






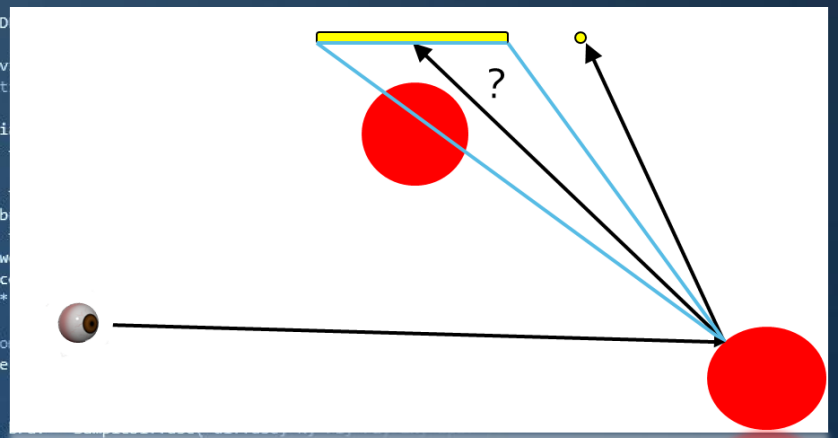
```
efl + refr)) && (depth < MAXDEPTH)

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

MAXDEPTH)

survive = SurvivalProbability( diffuse );
estimation - doing it properly, closely
df;
radiance = SampleLight( &rand, I, &L, &lightD
e.x + radiance.y + radiance.z) > 0) &&
v = true;
at brdfPdf = EvaluateDiffuse( L, N ) * Psurvive;
at3 factor = diffuse * INVPI;
at weight = Mis2( directPdf, brdfPdf );
at cosThetaOut = dot( N, L );
E * ((weight * cosThetaOut) / directPdf) * (radian

andom walk - done properly, closely following Se
vive)

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

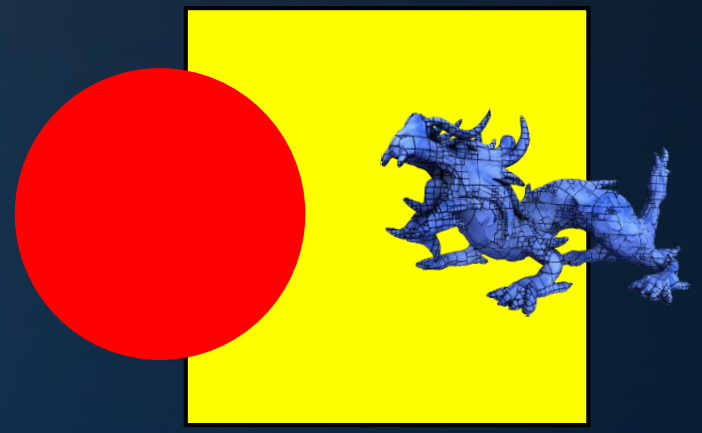


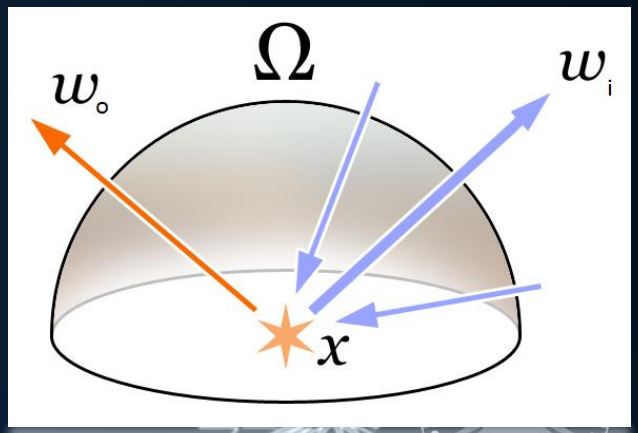





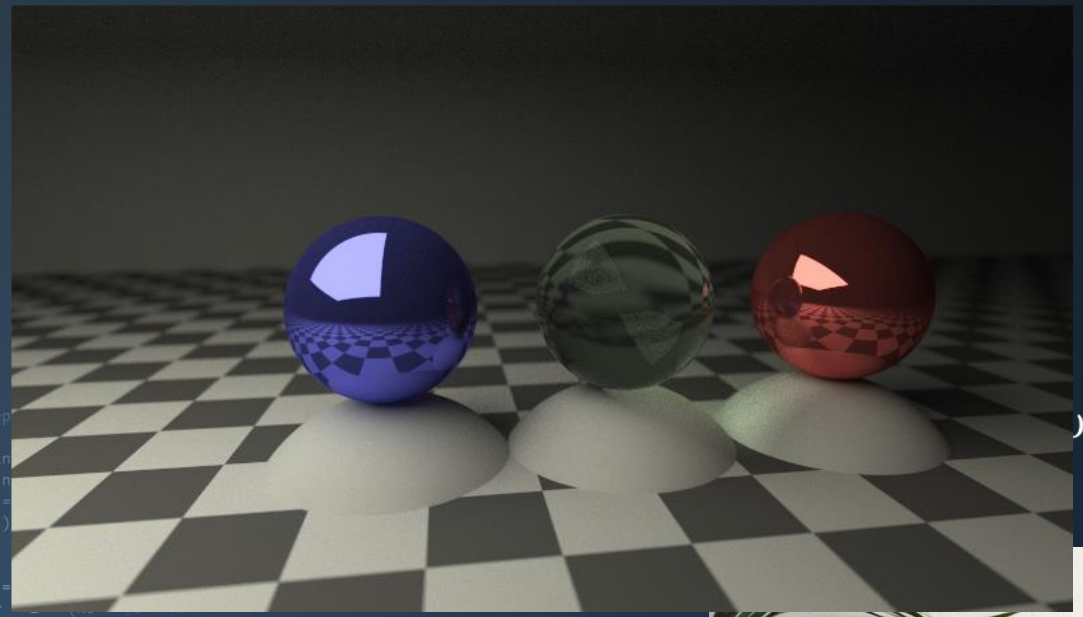

$$V_A = \int_A^B f(x)\,dx \approx \frac{B-A}{N}\sum_{i=1}^{N} f(X_i)$$

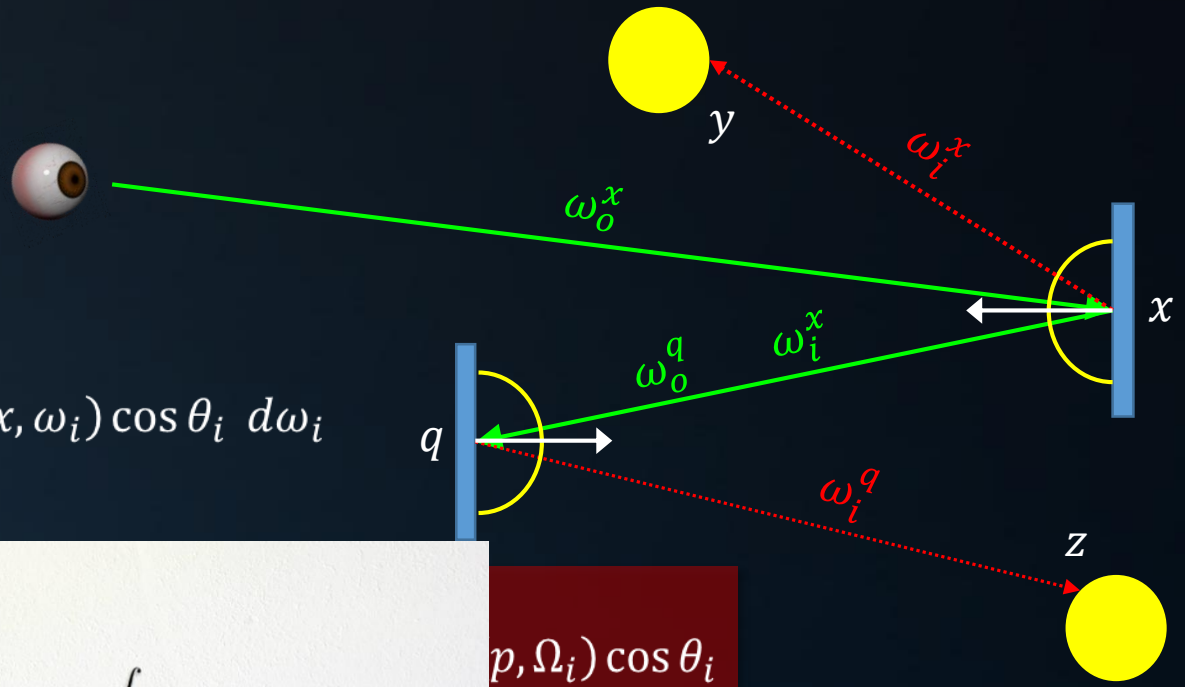
$$L(s \leftarrow x) = L_E(s \leftarrow x) + \int_A f_r(s \leftarrow x \leftarrow y)\, L(x \leftarrow y)\, G(x \leftrightarrow y)\, dA(y)$$











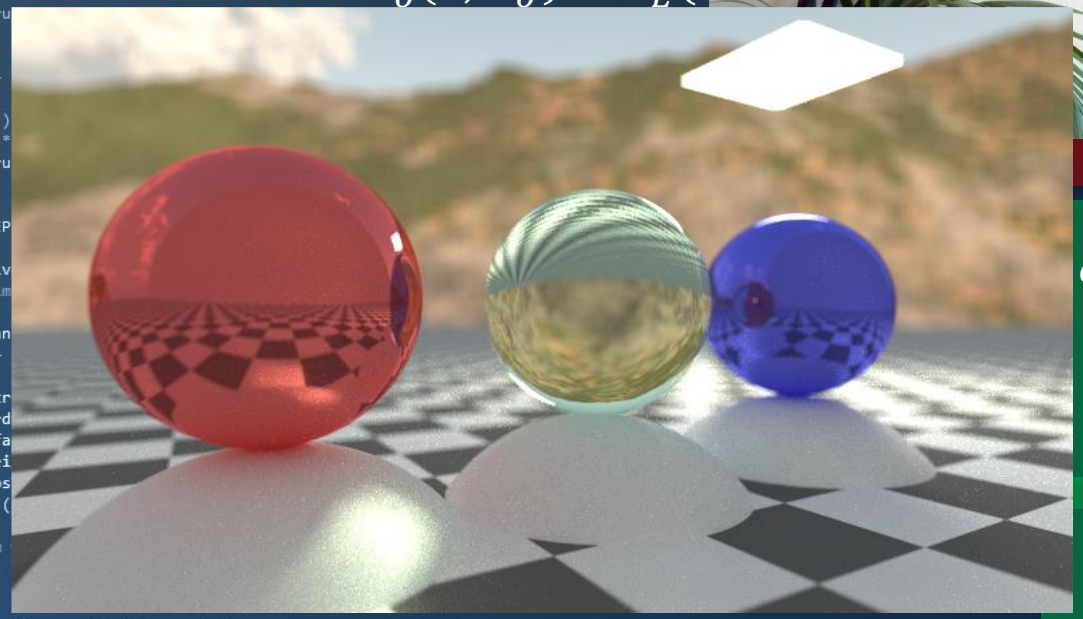
$$L_o(x, \omega_o^x) = L_E(x$$

$$\ldots_o, \omega_i)\, L_i(x, \omega_i) \cos\theta_i \; d\omega_i$$

$$\ldots p, \Omega_i) \cos\theta_i$$

$$L_o = L_e + \int_\Omega L_i \cdot f_r \cdot \cos\theta \cdot d\omega$$

direct light

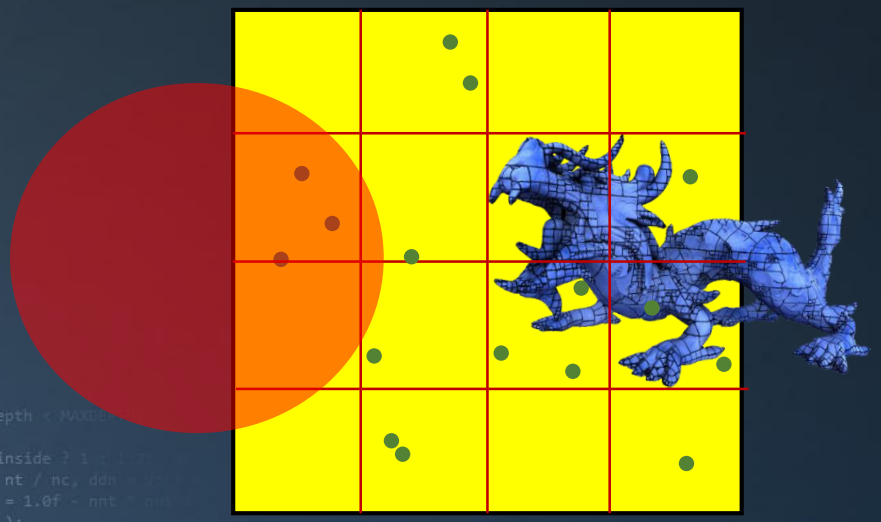


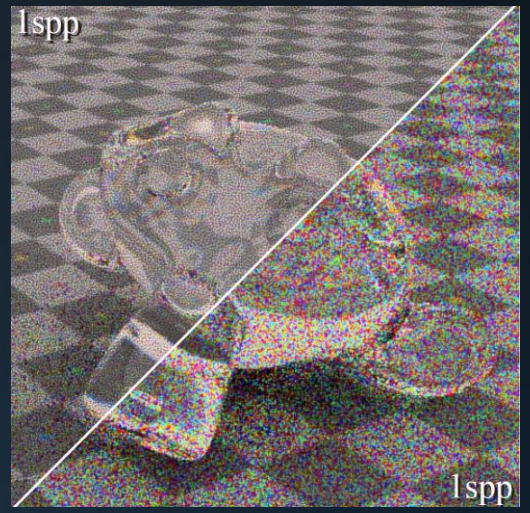
$$\ldots \omega_o^q, \omega_i^q)\; \cos\theta_i^q \; f_r(x, \omega_o^x, \omega_i^x)\; \cos\theta_i^x \; d\omega_i^x \; d\omega_i^q$$

1st bounce

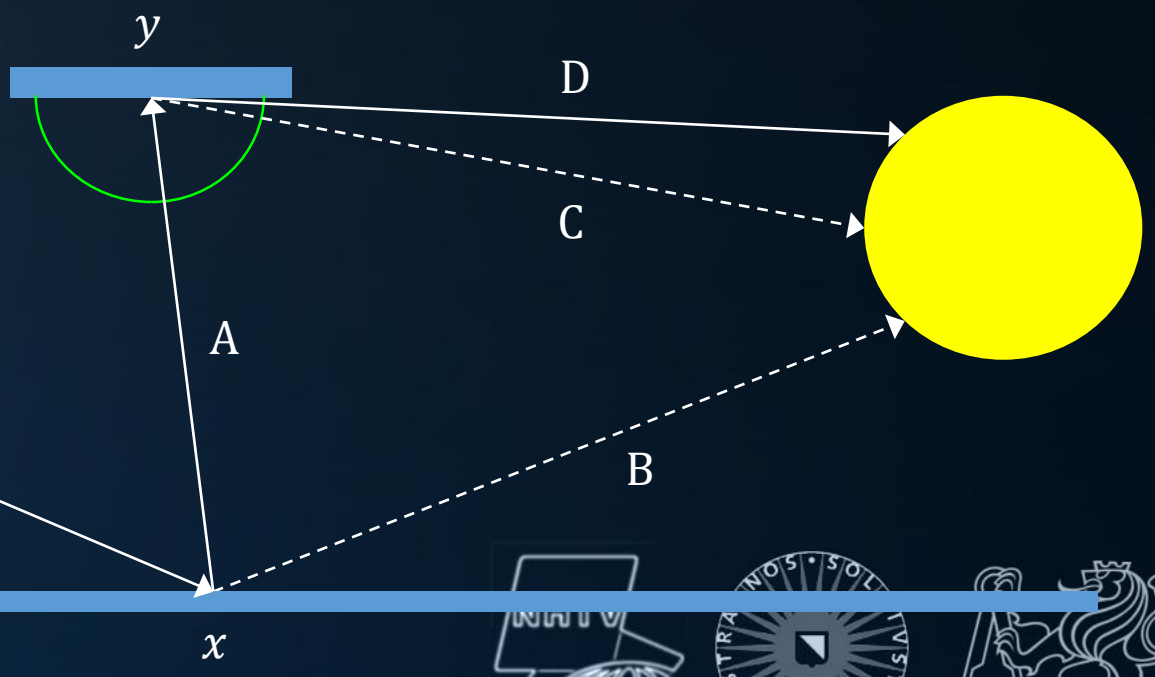
2nd bounce

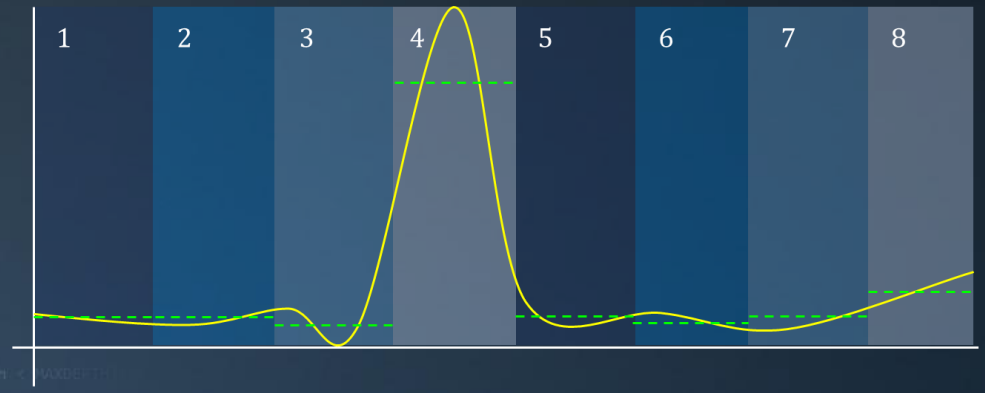
$$\approx \ldots$$

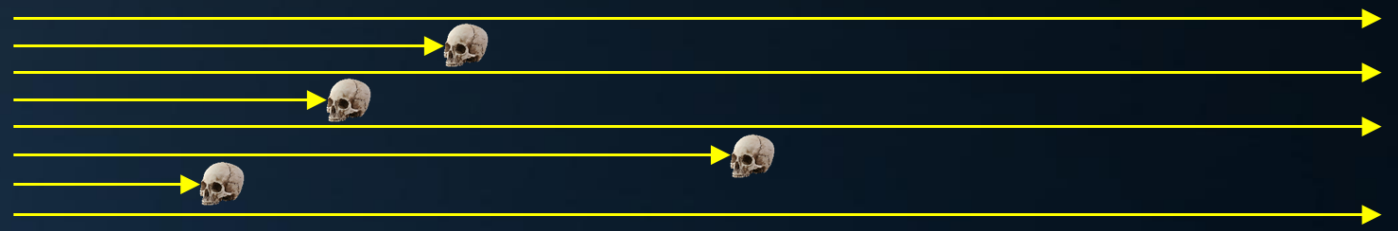




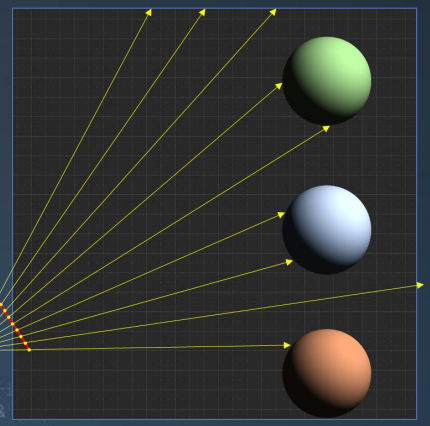
```
Color Sample( Ray ray )
{
    // trace ray
    I, N, material = Trace( ray );
    BRDF = material.albedo / PI;
    // terminate if ray left the scene
    if (ray.NOHIT) return BLACK;
    // terminate if we hit a light source
    if (material.isLight) return BLACK;
    // sample a random light source
    L, Nl, dist, A = RandomPointOnLight();
    Ray lr( I, L, dist );
    if (N·L > 0 && Nl·-L > 0) if (!Trace( lr ))
    {
        solidAngle = ((Nl·-L) * A) / dist²;
        Ld = lightColor * solidAngle * BRDF * N·L;
    }
    // continue random walk
    R = DiffuseReflection( N );
```

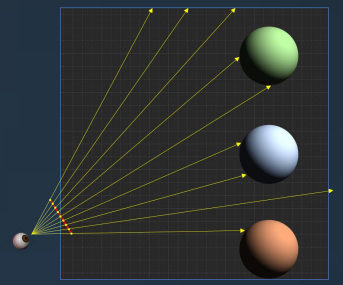


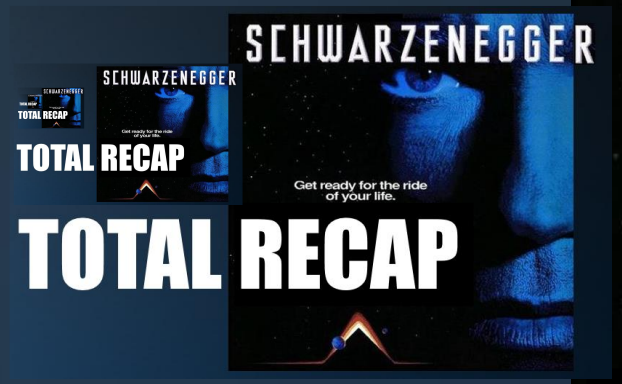


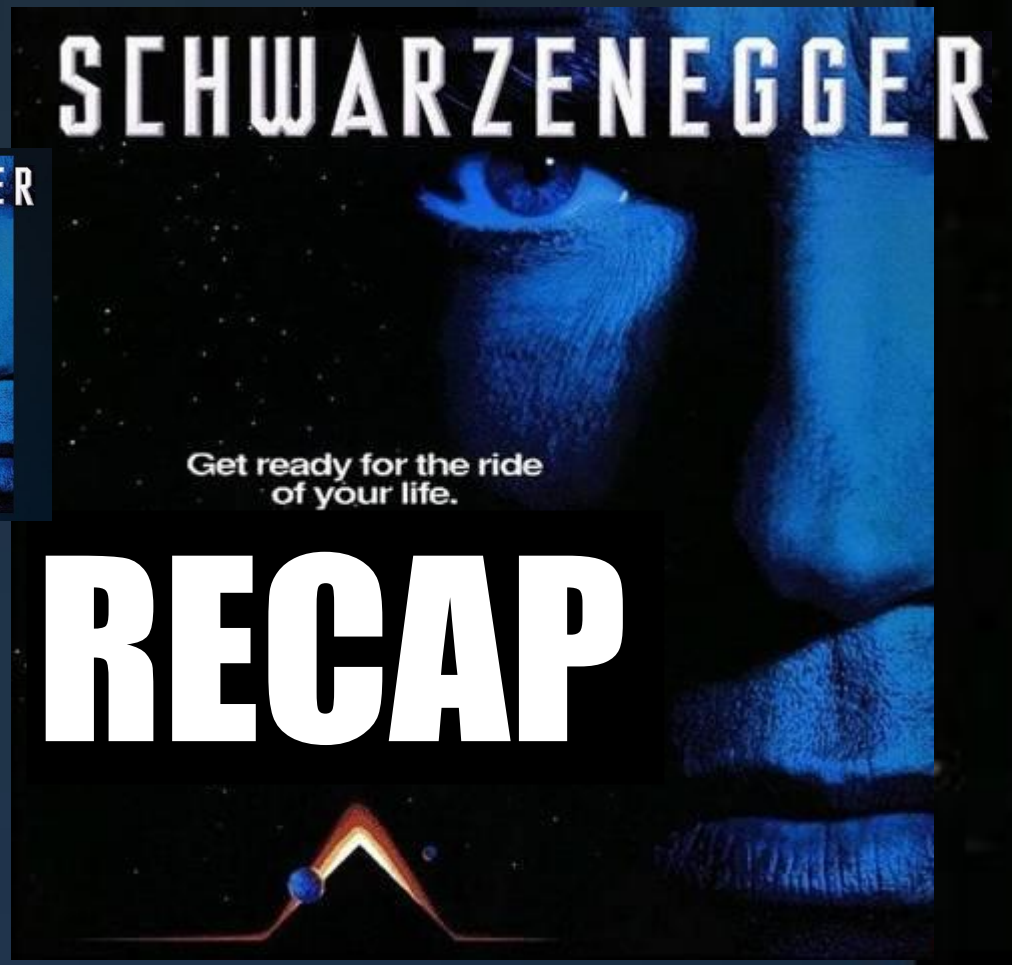


```
Color Sample( Ray ray )
{
	// trace ray
	I, N, material = Trace( ray );
	// terminate if ray left the scene
	if (ray.NOHIT) return BLACK;
	// terminate if we hit a light source
	if (material.isLight) return emittance;
	// continue in random direction
	R = CosineWeightedDiffuseReflection( N );
	Ray r( I, R );
	// update throughput
	BRDF = material.albedo / PI;
	PDF = (N·R) / PI;
	Ei = Sample( r ) * (N·R) / PDF;
	return BRDF * Ei;
}
```

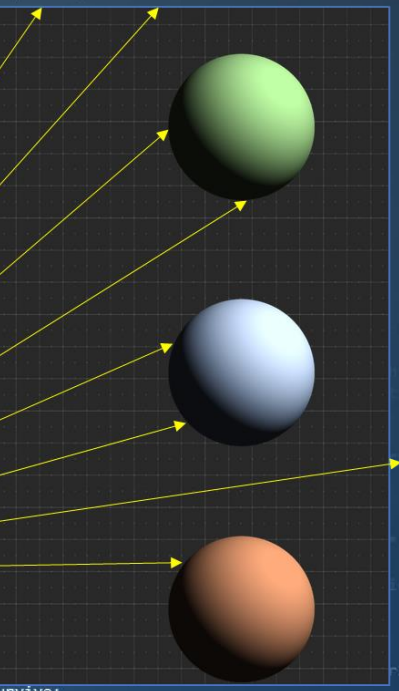


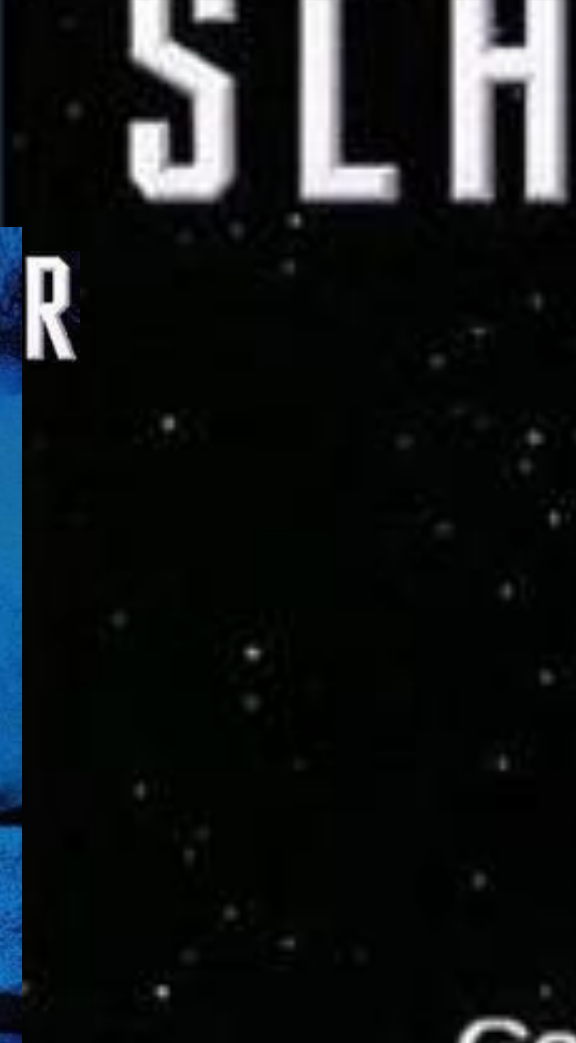






TOTAL RECAP

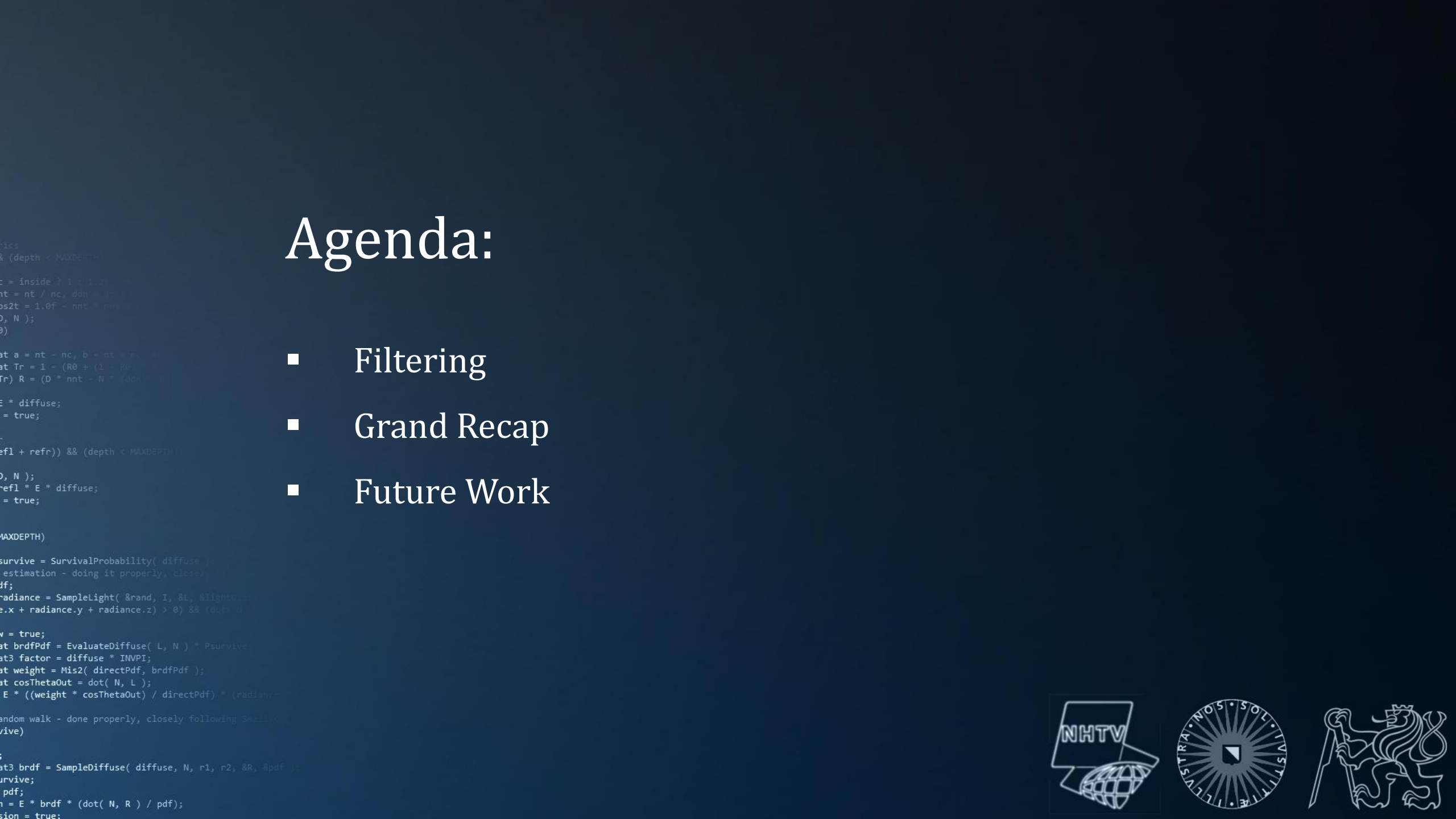




TOTAL RECAP

Agenda:

- Filtering
- Grand Recap
- Future Work



Materials

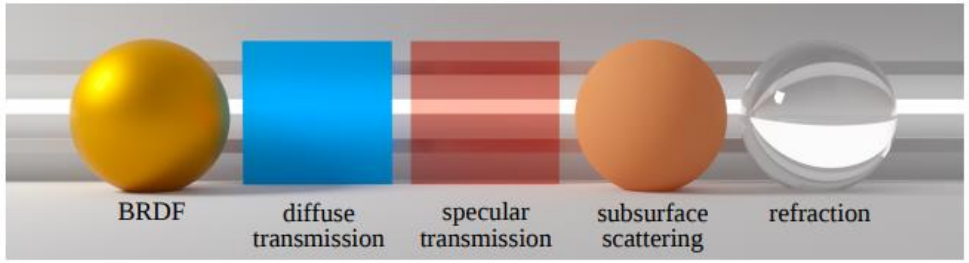


Figure 1: Rendered examples using our new BSDF.

Extending the Disney BRDF to a BSDF with Integrated Subsurface Scattering

by Brent Burley, Walt Disney Animation Studios

1 Introduction

We introduced a new physically based shading model on Wreck-It Ralph [Bur12], and we used this single, general-purpose BRDF on all materials (except hair). This model, which has come to be known as the Disney BRDF, is able to reproduce a wide range of materials with only a few parameters. For our next film, Frozen, we continued to use this BRDF unmodified, but effects like refraction and subsurface scattering were computed separately from the BRDF, and indirect illumination was approximated using point clouds. All of these effects were combined through ad hoc shading in an additive way.

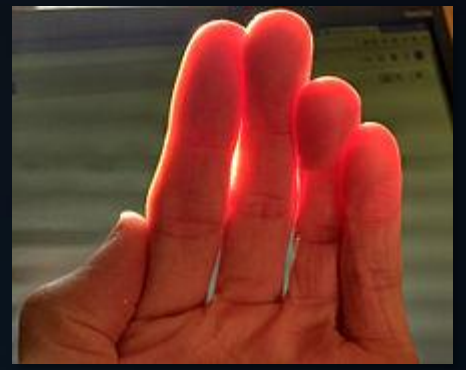
Starting with Big Hero 6 in 2014, we switched from ad hoc lighting and shading to path-traced

Other resources:

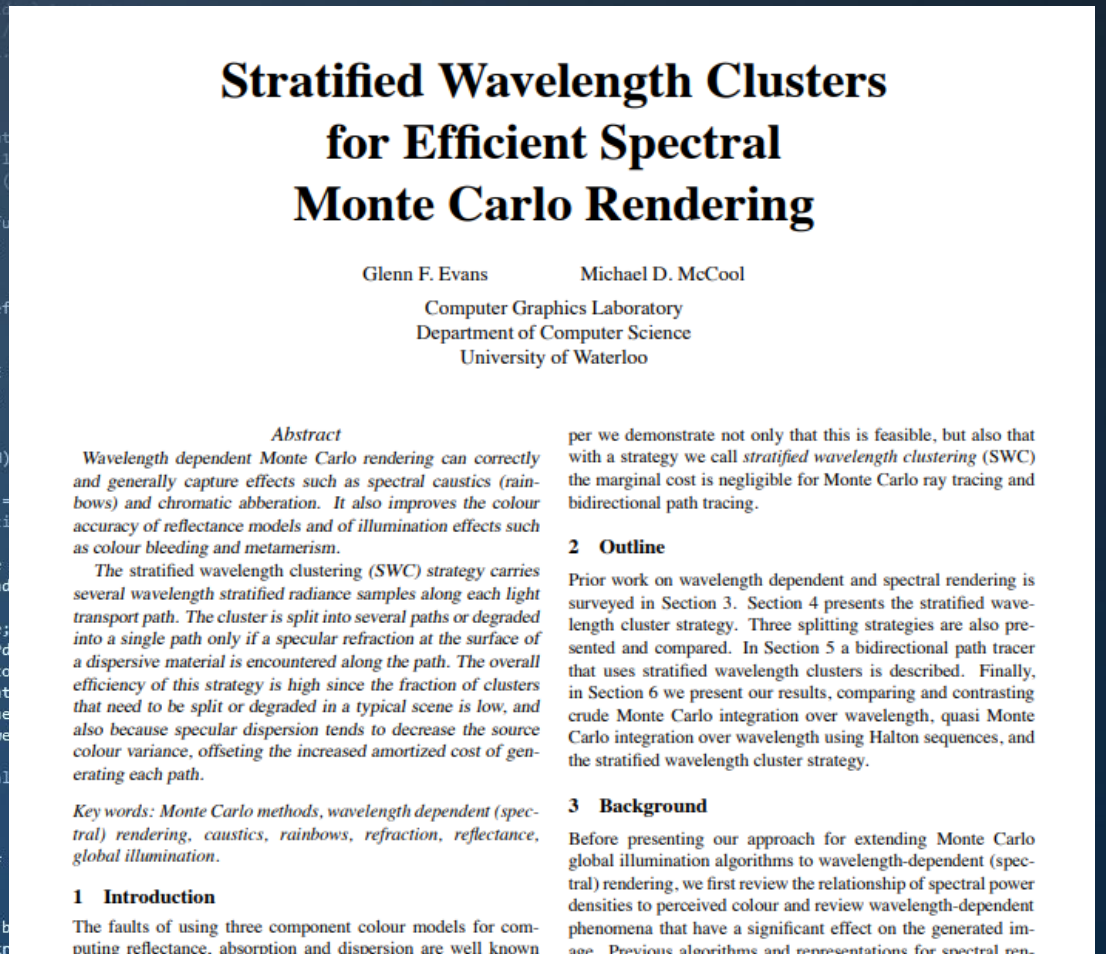
Implementing the Disney BSDF
schuttejoe.github.io/post/disneybsdf

Efficient Rendering of Layered Materials using an Atomic Decomposition with Statistical Operators. Belcour, 2018.

Eric Heitz's Research Page:
eheitzresearch.wordpress.com/research
(all his work is good)



Spectral Rendering



Stratified Wavelength Clusters for Efficient Spectral Monte Carlo Rendering

Glenn F. Evans Michael D. McCool

Computer Graphics Laboratory
Department of Computer Science
University of Waterloo

Abstract

Wavelength dependent Monte Carlo rendering can correctly and generally capture effects such as spectral caustics (rainbows) and chromatic abberation. It also improves the colour accuracy of reflectance models and of illumination effects such as colour bleeding and metamerism.

The stratified wavelength clustering (SWC) strategy carries several wavelength stratified radiance samples along each light transport path. The cluster is split into several paths or degraded into a single path only if a specular refraction at the surface of a dispersive material is encountered along the path. The overall efficiency of this strategy is high since the fraction of clusters that need to be split or degraded in a typical scene is low, and also because specular dispersion tends to decrease the source colour variance, offseting the increased amortized cost of generating each path.

Key words: Monte Carlo methods, wavelength dependent (spectral) rendering, caustics, rainbows, refraction, reflectance, global illumination.

1 Introduction

The faults of using three component colour models for computing reflectance, absorption and dispersion are well known per we demonstrate not only that this is feasible, but also that with a strategy we call *stratified wavelength clustering* (SWC) the marginal cost is negligible for Monte Carlo ray tracing and bidirectional path tracing.

2 Outline

Prior work on wavelength dependent and spectral rendering is surveyed in Section 3. Section 4 presents the stratified wavelength cluster strategy. Three splitting strategies are also presented and compared. In Section 5 a bidirectional path tracer that uses stratified wavelength clusters is described. Finally, in Section 6 we present our results, comparing and contrasting crude Monte Carlo integration over wavelength, quasi Monte Carlo integration over wavelength using Halton sequences, and the stratified wavelength cluster strategy.

3 Background

Before presenting our approach for extending Monte Carlo global illumination algorithms to wavelength-dependent (spectral) rendering, we first review the relationship of spectral power densities to perceived colour and review wavelength-dependent phenomena that have a significant effect on the generated image. Previous algorithms and representations for spectral ren-

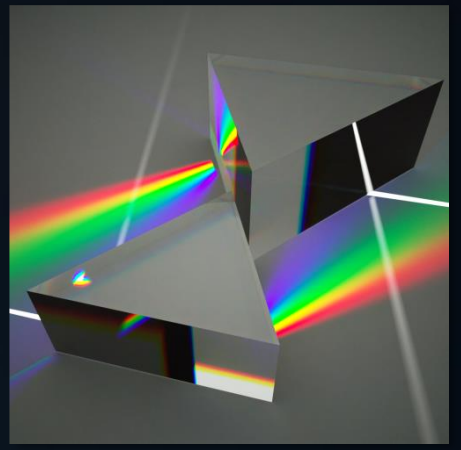
Other resources:

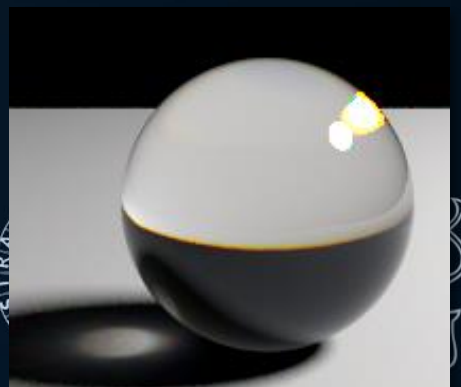
Hero Wavelength Spectral Sampling. Wilie et al., 2014.

Physically Meaningful Rendering using Tristimulus Colours. Meng et al., 2015.

PBRT. Pharr et al., 2004-2018.
https://www.pbrt.org

Mitsuba Renderer. W. Jakob, 2014.
https://www.mitsuba-renderer.org





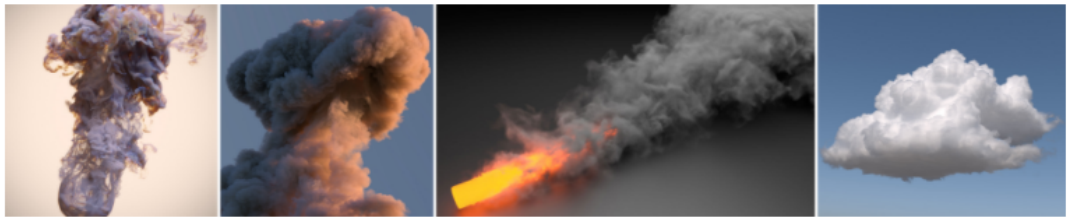
Participating Media

Monte Carlo methods for physically based volume rendering

Jan Novák[1] Iliyan Georgiev[2] Johannes Hanika[3] Jaroslav Křivánek[4] Wojciech Jarosz[5]

[1]Disney Research [2]Solid Angle [3]Karlsruhe Institute of Technology [4]Charles University, Prague [5]Dartmouth College

In *ACM SIGGRAPH Courses*, 2018



Various media rendered with Monte Carlo methods for physically based simulation of light transport in volumes. The three images on the left are courtesy of Lee Griggs.

Abstract | Downloads | Cite | Related

Abstract

We survey methods that utilize Monte Carlo (MC) integration to simulate light transport in scenes with participating media. The goal of this course is to complement a recent Eurographics 2018 state-of-the-art report providing a broad overview of most techniques developed to date, including a few methods from neutron transport, with a focus on concepts that are most relevant to CG practitioners.

The wide adoption of path-tracing algorithms in high-end realistic rendering has stimulated many diverse research initiatives aimed at efficiently rendering scenes with participating media. More computational power has enabled holistic approaches that tie volumetric effects and surface scattering together and simplify authoring workflows. Methods that were previously assumed to be incompatible have been unified to allow renderers to benefit from each method's respective strengths. Generally, investigations have shifted away from specialized solutions, e.g. for single- or multiple-scattering approximations or analytical methods, towards the more versatile Monte Carlo algorithms that are currently enjoying a widespread success in many production settings.

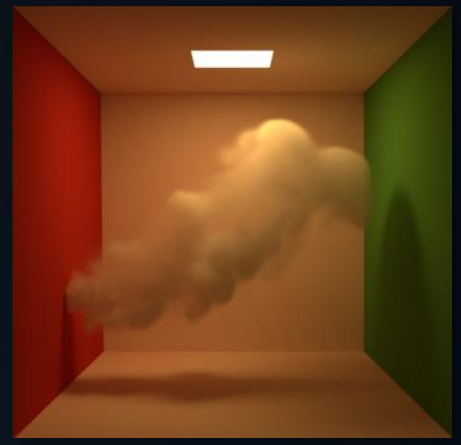
The goal of this course is to provide the audience with a deep, up-to-date understanding of key techniques for free-path sampling, transmittance estimation, and light-path construction in participating media, including those that are presently utilized in production rendering systems. We present a coherent overview of the fundamental building blocks and we contrast the various advanced methods that build on them, providing attendees with guidance for implementing existing solutions and developing new ones.

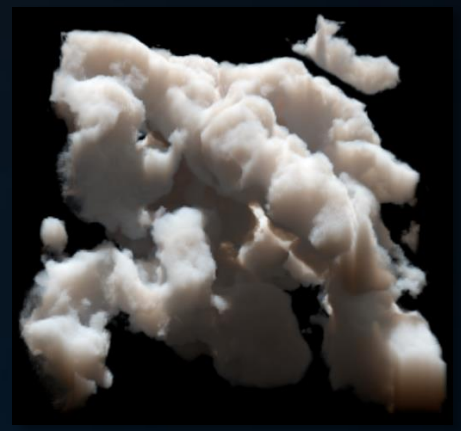
Downloads

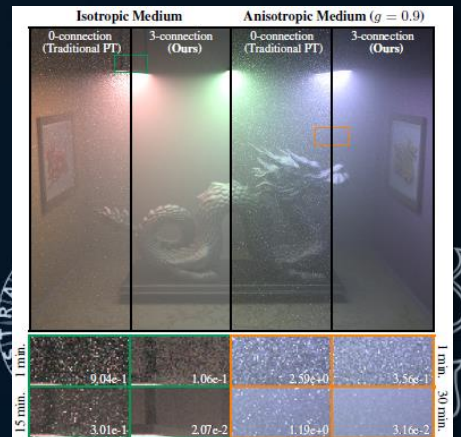
Other resources:

Importance Sampling Techniques for Path Tracing in Participating Media. Kulla and Fajardo, 2014.

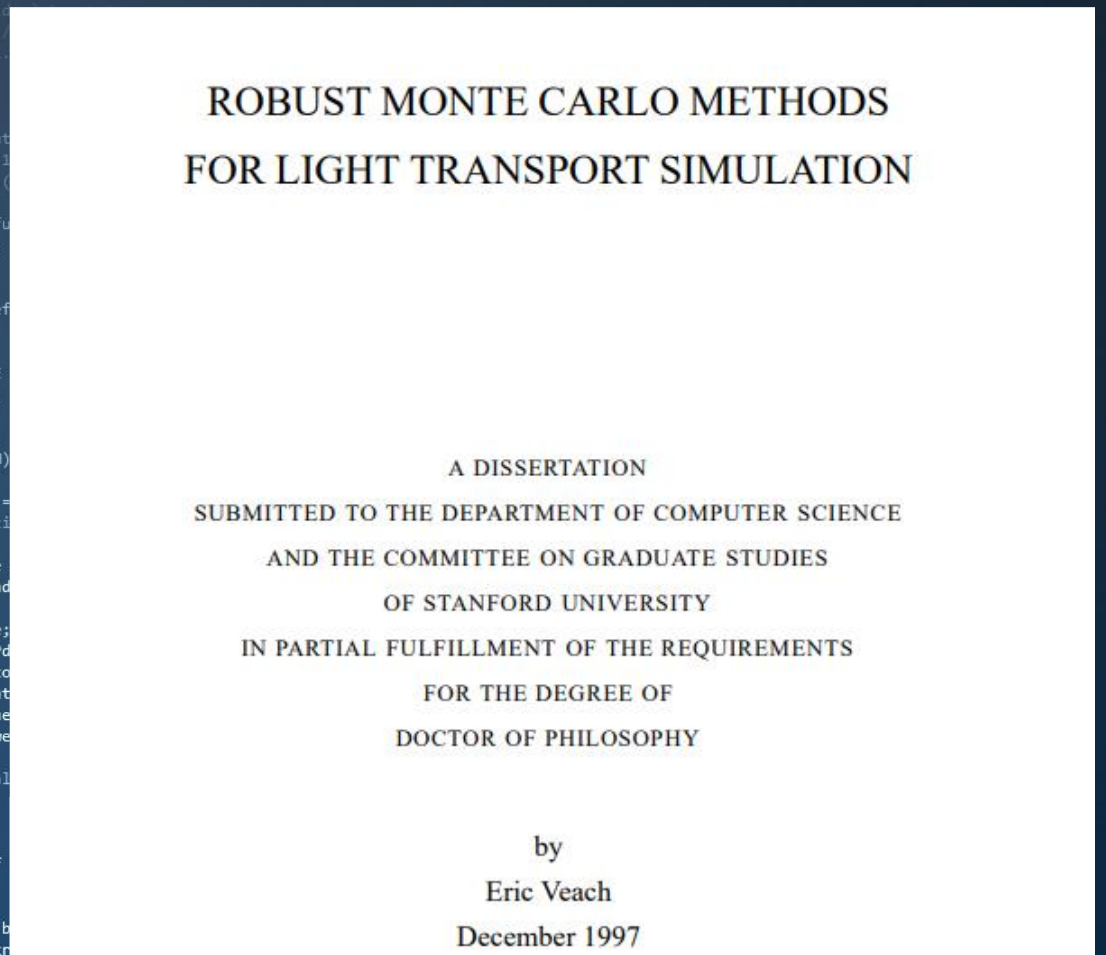
Area Light Equi-Angular Sampling on ShaderToy: https://ww.shadertoy.com/view/ldXGzS







Light Transport



ROBUST MONTE CARLO METHODS
FOR LIGHT TRANSPORT SIMULATION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

by
Eric Veach
December 1997

Other resources:

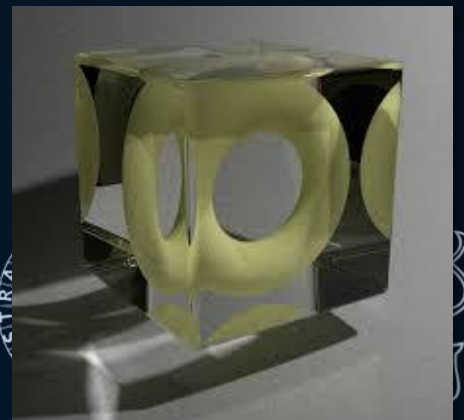
The Rendering Equation. James T. Kajiya, 1986.

Bidirectional Path Tracing. Michal Vlnas, 2018 (student paper).

Global Illumination using Photon Maps. Jensen, 1996.

Progressive Photon Mapping. Hachisuka et al., 2008.

(article on) Vertex Connection and Merging, Georgev et al., 2012. https://schuttejoe.github.io/post/vertexconnectionandmerging



Primitives

Phantom Ray-Hair Intersector

Alexander Reshetov, David Luebke
NVIDIA

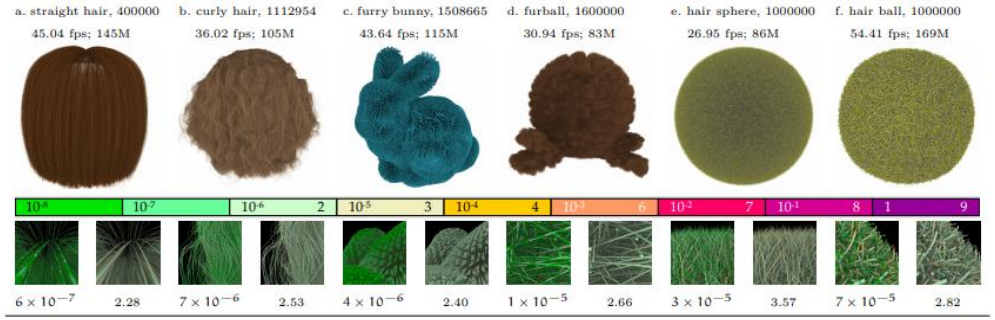


Figure 1. *Different models rendered with our technique and number of curves in the model. Second line: the achieved frame rate on a Titan Xp and the total number of rays traced per second, including one primary ray for each pixel at 1000×1000 screen resolution and four ambient occlusion rays for each hit point. For each model, the bottom row gives the average error in curve parameter* t *(left) and the number of iterations (right). The corresponding inserts show these values for each primary ray according to the heatmap strip in the middle.*

ABSTRACT

We present a new approach to ray tracing swept volumes along trajectories defined by cubic Bézier curves. It performs at two-thirds of the speed of ray-triangle intersection, allowing essentially even treatment of such primitives in ray tracing applications that require hair, fur, or yarn rendering.

At each iteration, we approximate a radially symmetric swept volume with a tangential cone. A distance from the ray-cone intersection to the cone's base is then used to compute the next curve parameter t. When this distance is zero, the ray intersects the swept volume and the cone at the same point and we stop the iterations. To enforce continuity of the iterative root finding, we introduce "phantom" intersection

CCS CONCEPTS

• **Computing methodologies** → **Ray tracing**; **Parametric curve and surface models**;

KEYWORDS

Ray tracing, Bézier curves, swept volumes, generalized cylinders, hair and fur rendering

ACM Reference Format:
Alexander Reshetov, David Luebke. 2018. Phantom Ray-Hair Intersector. In *Proceedings of HPG'18*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3233307

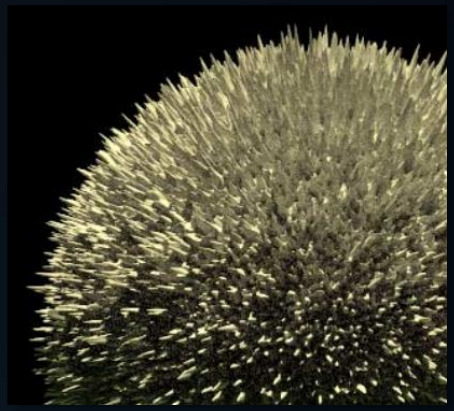
1 INTRODUCTION AND PRIOR ART

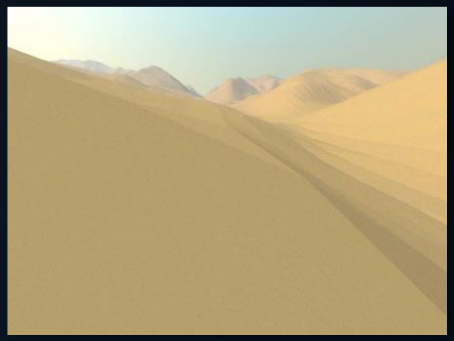
Other resources:

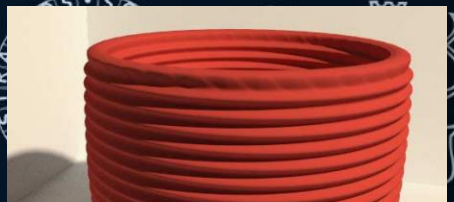
Direct Ray Tracing of Smoothed and Displacement Mapped Triangles. Smits et al., 2000.

Direct Ray Tracing of Phong Tessellation. Ogaki and Tokuyoshi, 2011.

Two-Level Ray Tracing with Reordering for Highly Complex Scenes. Hanika et al., 2010.







Render Comparison

REDSHIFT 2.6.31

ARNOLD 5.2.2.0

V-RAY NEXT, Beta 2, Build 4.11.01

HOUDINI 17.0.352 (MANTRA)

01/11/2019

Production

Arnold: A Brute-Force Production Path Tracer

ILIYAN GEORGIEV, THIAGO IZE, MIKE FARNSWORTH, RAMÓN MONTOYA-VOZMEDIANO, ALAN KING, BRECHT VAN LOMMEL, ANGEL JIMENEZ, OSCAR ANSON, SHINJI OGAKI, ERIC JOHNSTON, ADRIEN HERUBEL, DECLAN RUSSELL, FRÉDÉRIC SERVANT, and MARCOS FAJARDO, Solid Angle

Fig. 1. Arnold is a path-tracing renderer used for the production of photo-realistic and art-directed visual effects in feature films (left), commercials (middle), animated films (right), television series, music videos, game cinematics, motion graphics, and others. *Gravity ©2013 Warner Bros. Pictures, courtesy of Framestore; Racing Faces ©2016 Opel Motorsport, courtesy of The Mill; Captain Underpants ©2017 DreamWorks Animation.*

Arnold is a physically based renderer for feature-length animation and visual effects. Conceived in an era of complex multi-pass rasterization-based workflows struggling to keep up with growing demands for complexity and realism, Arnold was created to take on the challenge of making the simple and elegant approach of brute-force Monte Carlo path tracing practical for production rendering. Achieving this required building a robust piece of ray-tracing software that can ingest large amounts of geometry with detailed shading and lighting and produce images with high fidelity, while scaling well with the available memory and processing power.

Arnold's guiding principles are to expose as few controls as possible, provide rapid feedback to artists, and adapt to various production workflows. In this article, we describe its architecture with a focus on the design

ACM Reference format:
Iliyan Georgiev, Thiago Ize, Mike Farnsworth, Ramón Montoya-Vozmediano, Alan King, Brecht Van Lommel, Angel Jimenez, Oscar Anson, Shinji Ogaki, Eric Johnston, Adrien Herubel, Declan Russell, Frédéric Servant, and Marcos Fajardo. 2018. Arnold: A Brute-Force Production Path Tracer. *ACM Trans. Graph.* 37, 3, Article 32 (July 2018), 12 pages.
https://doi.org/10.1145/3182160

1 INTRODUCTION

At a purely technical level, visual realism in computer-generated imagery boils down to two areas: (1) amount of scene detail, e.g., number of geometric primitives, amount of textures, variety of ma-

Other resources:

The Iray Light Transport Simulation and Rendering System. Keller et al., 2017.

The Design and Evolution of Disney's Hyperion Renderer. Burley et al., 2018.

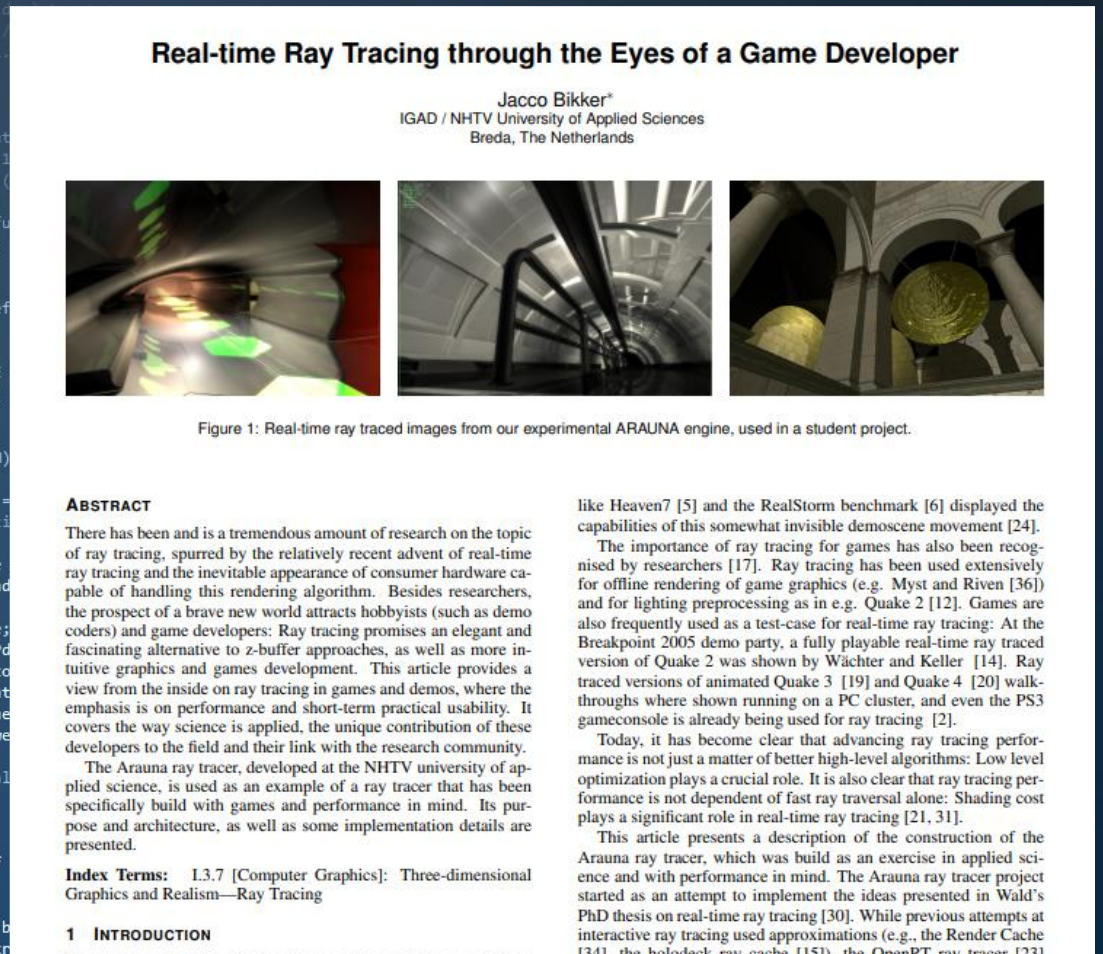
Manuka: A batch-shading architecture for spectral path tracing in movie production. Fascione et al., 2018.

Mitsuba 2: A Retargetable Forward and Inverse Renderer. Nimier-David, 2019.


ALITA
BATTLE ANGEL

Ray Tracing for Games

Games



Real-time Ray Tracing through the Eyes of a Game Developer

Jacco Bikker*
IGAD / NHTV University of Applied Sciences
Breda, The Netherlands

Figure 1: Real-time ray traced images from our experimental ARAUNA engine, used in a student project.

ABSTRACT

There has been and is a tremendous amount of research on the topic of ray tracing, spurred by the relatively recent advent of real-time ray tracing and the inevitable appearance of consumer hardware capable of handling this rendering algorithm. Besides researchers, the prospect of a brave new world attracts hobbyists (such as demo coders) and game developers: Ray tracing promises an elegant and fascinating alternative to z-buffer approaches, as well as more intuitive graphics and games development. This article provides a view from the inside on ray tracing in games and demos, where the emphasis is on performance and short-term practical usability. It covers the way science is applied, the unique contribution of these developers to the field and their link with the research community.

The Arauna ray tracer, developed at the NHTV university of applied science, is used as an example of a ray tracer that has been specifically build with games and performance in mind. Its purpose and architecture, as well as some implementation details are presented.

Index Terms: I.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism—Ray Tracing

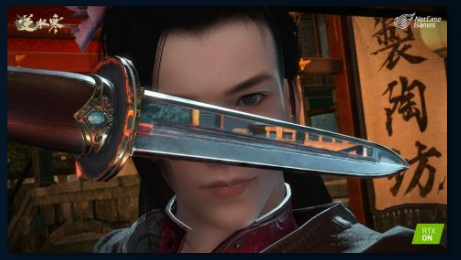
1 INTRODUCTION

like Heaven7 [5] and the RealStorm benchmark [6] displayed the capabilities of this somewhat invisible demoscene movement [24].

The importance of ray tracing for games has also been recognised by researchers [17]. Ray tracing has been used extensively for offline rendering of game graphics (e.g. Myst and Riven [36]) and for lighting preprocessing as in e.g. Quake 2 [12]. Games are also frequently used as a test-case for real-time ray tracing: At the Breakpoint 2005 demo party, a fully playable real-time ray traced version of Quake 2 was shown by Wächter and Keller [14]. Ray traced versions of animated Quake 3 [19] and Quake 4 [20] walkthroughs where shown running on a PC cluster, and even the PS3 gameconsole is already being used for ray tracing [2].

Today, it has become clear that advancing ray tracing performance is not just a matter of better high-level algorithms: Low level optimization plays a crucial role. It is also clear that ray tracing performance is not dependent of fast ray traversal alone: Shading cost plays a significant role in real-time ray tracing [21, 31].

This article presents a description of the construction of the Arauna ray tracer, which was build as an exercise in applied science and with performance in mind. The Arauna ray tracer project started as an attempt to implement the ideas presented in Wald's PhD thesis on real-time ray tracing [30]. While previous attempts at interactive ray tracing used approximations (e.g., the Render Cache [34], the holodeck ray cache [15]), the OpenRT ray tracer [23]



Other resources:

OptiX: A General Purpose Ray Tracing Engine. Parker et al., 2010.

REAL-TIME RAYTRACING
WITH NVIDIA RTX. Stich, 2018.

(not a lot of research so far…)

Massive Scenes

Moana Island scene. Heather Pritchett & Rasmus Tamstorf, 2016. *~15 billion primitives.*

Ray Tracing for Games

Ray Tracing for Games

Wednesday
13:00 – 17:00

course intro
LH2
template
Whitted
refactoring
RT-centric games

LAB 1

Thursday
09:00 – 14:00

advanced Whitted
audio, AI & physics
faster Whitted
Heaven7

LAB 2

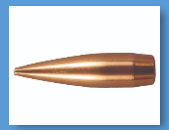
work @ home

End result day 2:

A solid Whitted-style ray tracer, as a basis for subsequent work.

Friday
09:00 – 17:00

optimization
profiling, rules of engagement
threading



LAB 3

SIMD
applied SIMD
SIMD triangle
SIMD AABB

LAB 4

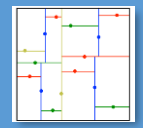
End result day 3:

A 5x faster tracer.

GLOBAL GAME JAM

Monday
09:00 – 17:00

acceleration
grid, BVH, kD-tree
SAH
binning



LAB 5

refitting
top-level BVH
threaded building

LAB 6

End result day 4:

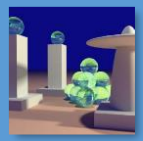
A real-time tracer.

Tuesday
09:00 – 17:00

Monte-Carlo
Cook-style
glossy, AA
area lights, DOF

LAB 7

path tracing



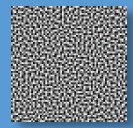
LAB 8

End result day 5:

Cook or Kajiya.

Thursday
09:00 – 17:00

random numbers
stratification
blue noise



LAB 9

importance
sampling
next event
estimation

LAB 10

End result day 6:

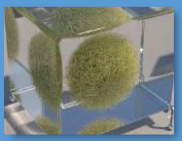
Efficiency.

Friday
09:00 – 17:00

future work

LAB

path guiding



LAB 10

End result day 6:

Great product.

End of PART 11.