Are We Done With Ray Tracing?
State-of-the-Art and Challenges in Game Ray Tracing

Colin Barré-Brisebois, @ZigguratVertigo
SEED – Electronic Arts

SEED


How did we get here?

Microsoft | DirectX Developer Blog Product Blogs DevOps Features Languages .NET More

Announcing Microsoft DirectX Raytracing!

D3D Team

March 19th, 2018

If you just want to see what DirectX Raytracing can do for gaming, check out the videos from Epic, Futuremark and EA, SEED. To learn about the magic behind the curtain, keep reading.

3D Graphics is a Lie

For the last thirty years, almost all games have used the same general technique—rasterization—to render images on screen. While the internal representation of the game world is maintained as three dimensions, rasterization ultimately operates in two dimensions (the plane of the screen), with 3D primitives mapped onto it through transformation matrices. Through approaches like z-buffering and occlusion culling, games have historically strived to minimize the number of spurious pixels rendered, as normally they do not contribute to the final frame. And in a perfect world, the pixels rendered would be exactly those that are directly visible from the camera:

PC GAMER THE GLOBAL AUTHORITY ON PC GAMES US Edition

News Reviews Hardware Indie Best Of Magazine Pro

HARDWARE

POPULAR Prime Day Deals Steam Changes Best PC Games Warframe Ne

What Microsoft's DirectX Raytracing means for gaming

By Jarred Walton March 19, 2018

Physically based graphics rendering could simplify and improve the way games look.

NVIDIA DEVELOPER Join Login

NVIDIA Developer Blog DEVELOPER NEWS SUBSCRIBE FOLLOW US

AI / DEEP LEARNING AUTONOMOUS MACHINES AUTONOMOUS VEHICLES DATA SCIENCE GRAPHICS / SIMULATION HPC SMART CITIES

GAME DEVELOPMENT GRAPHICS / SIMULATION

Introduction to NVIDIA RTX and DirectX Ray Tracing

By Martin Stich | March 19, 2018 Tags: Computer Graphics & Visualization, DXR, Game Development, NVIDIA RTX, ray tracing

"Ray tracing is the future, and it always will be!" has been the tongue-in-cheek phrase used by graphics developers for decades when [illegible]d whether real-time ray tracing will ever [illegible]

Every[illegible] is the only technology we know of that enable[illegible] renderer used in the movie industry, where comp[illegible] made immense strides over the years, and [illegible] can compute. Truly taking graphics to the [illegible] is why real-time ray tracing has long been the dream of gamers and game developers.

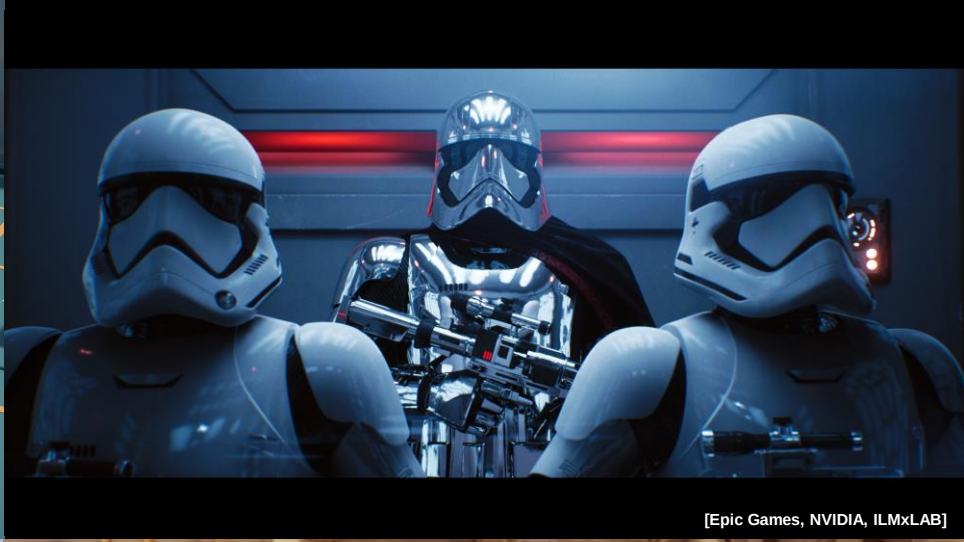
So will ray tracing always remain a dream of the future, and never arrive in the present? At GDC 2018, NVIDIA unveiled RTX, a high-performance implementation that [illegible] ray tracing APIs [illegible]ported by NVIDIA on Volta and future GPUs. At the same event, [illegible]utilizes into their industry standard DirectX APIs.

[illegible] combination that we can confidently answer the above question: the future is here! This is not a hyperbole. Leading game studios are developing upcoming titles using RTX through DirectX — today. Ray tracing in games is no longer a pipe dream. It is happening, and it will usher in a new era of real-time graphics.

GDC 2018 – DXR Unveiled

"Ray tracing is the future and ever will be"


[Electronic Arts, SEED]


[Epic Games, NVIDIA, ILMxLAB]

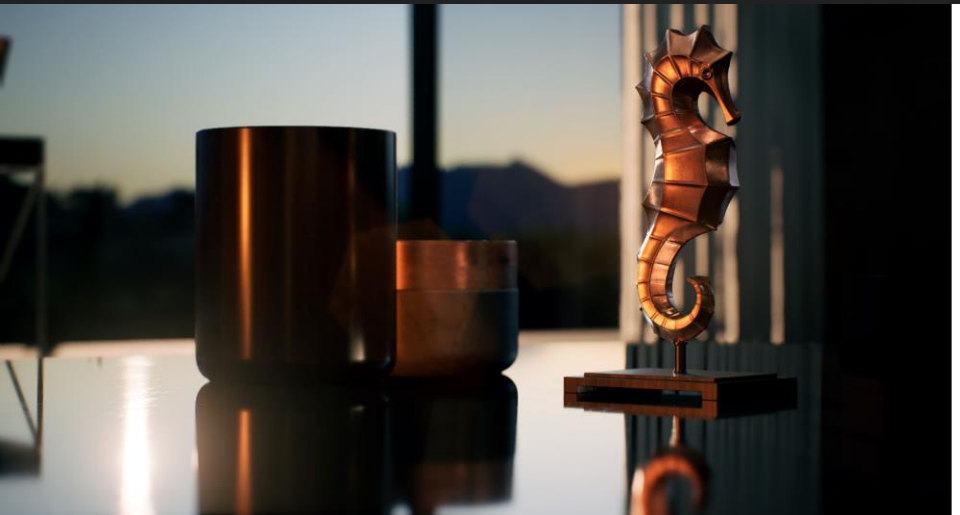

REMEDY
northlight


FUTUREMARK

Real-Time Ray Tracing in Software and Hardware


BATTLEFIELD V
NVIDIA GEFORCE RTX
BATTLEFIELD V
NVIDIA GEFORCE RTX
BATTLEFIELD V
NVIDIA GEFORCE RTX
RTX ON

Real-Time Hybrid Ray Tracing in Unreal Engine 4



[https://docs.unrealengine.com/en-US/Engine/Rendering/RayTracing/index.html]

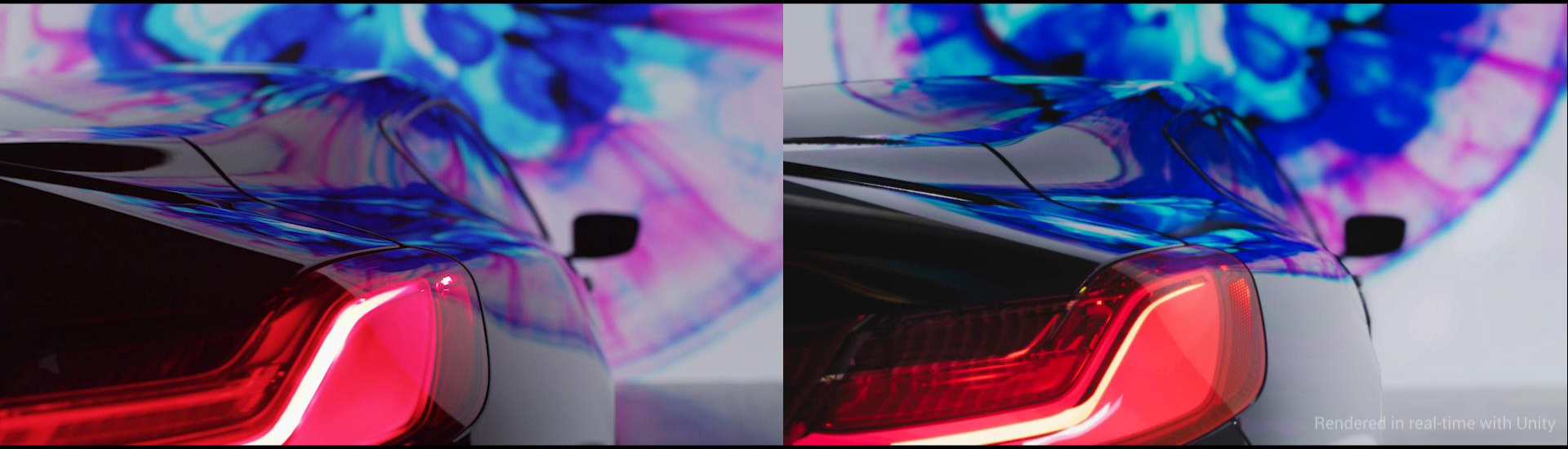
[Courtesy of Epic Games, Goodbye Kansas, Deep Forest Films]

[Courtesy of Epic Games, Goodbye Kansas, Deep Forest Films]

[Tatarchuk 2019, Courtesy of Unity Technologies]

Left: real-world footage. Right rendered with Unity

[Tatarchuk 2019, Courtesy of Unity Technologies]



Left: real-world footage. Right rendered with Unity

[Tatarchuk 2019, Courtesy of Unity Technologies]

Left: real-world footage. Right rendered with Unity

[Tatarchuk 2019, Courtesy of Unity Technologies]

[Christophe Schied, NVIDIA Lightspeed Studios™]

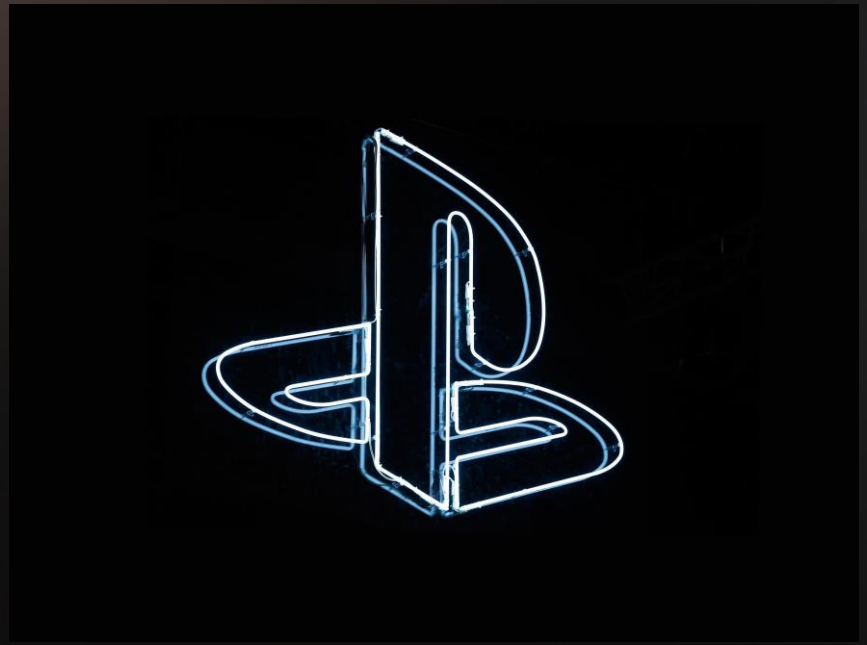
And many more…

- Assetto Corsa
- Atomic Heart
- Call of Duty: Modern Warfare
- Cyberpunk 2077
- Enlisted
- Justice
- JX3
- Mech Warrior V: Mercenaries
- Project DH
- Stay in the Light
- Vampire: The Masquerade – Bloodlines 2
- Watch Dogs: Legion
- Wolfenstein: Youngblood

Just the beginning of real-time ray tracing making its way into game products

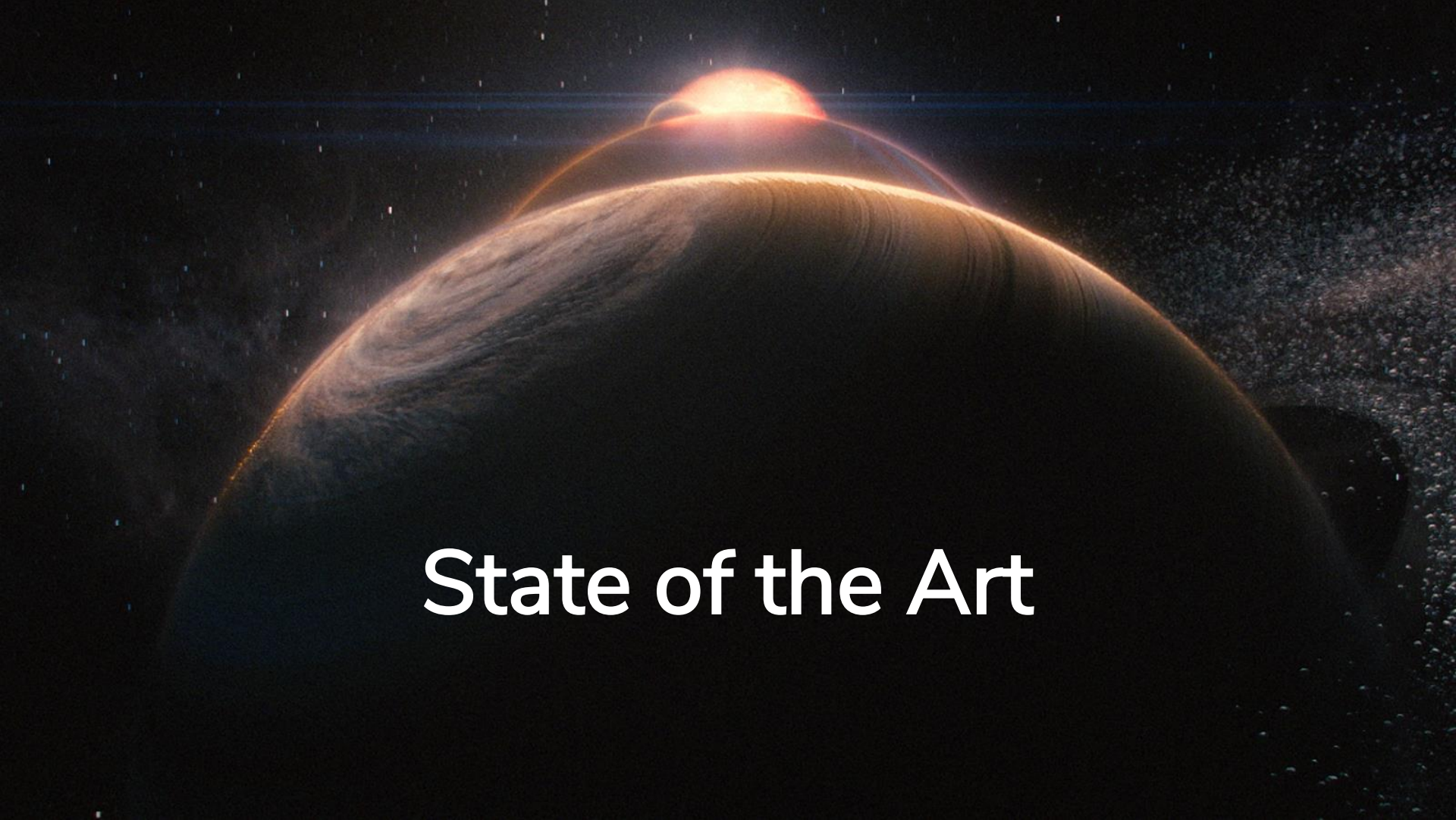
We're in for a great ride, and the work is not done! This is super exciting! ☺

Future Consoles



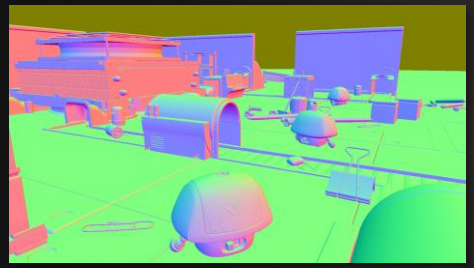
→ Microsoft: *E3 2019 Keynote*, June 9th 2019, link

→ SONY: *What to Expect From SONY's Next-Gen PlayStation*, Wired Magazine, April 16th 2019, link

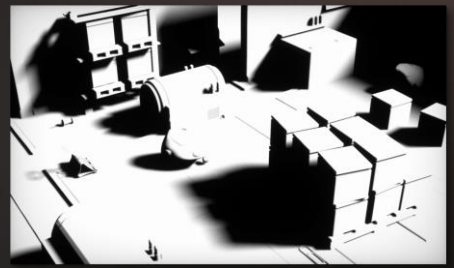


State of the Art

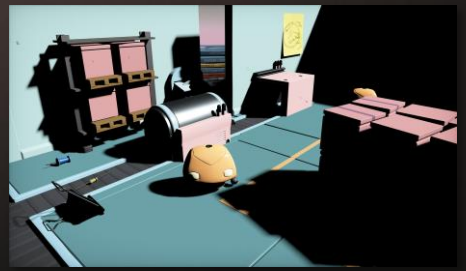
Hybrid Rendering Pipeline



Deferred shading
(raster)



Direct shadows
(ray trace or raster)



Lighting
(compute + ray trace)

Reflections
(ray trace or compute)

Global Illumination
(compute and ray trace)

Ambient occlusion
(ray trace or compute)

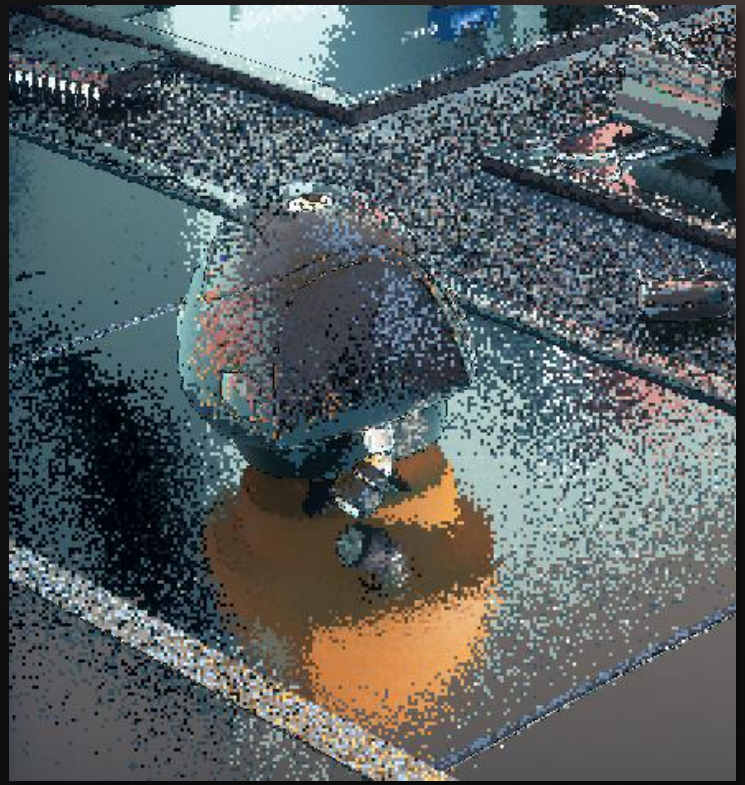
Transparency & Translucency
(ray trace and compute)

Post processing
(compute)

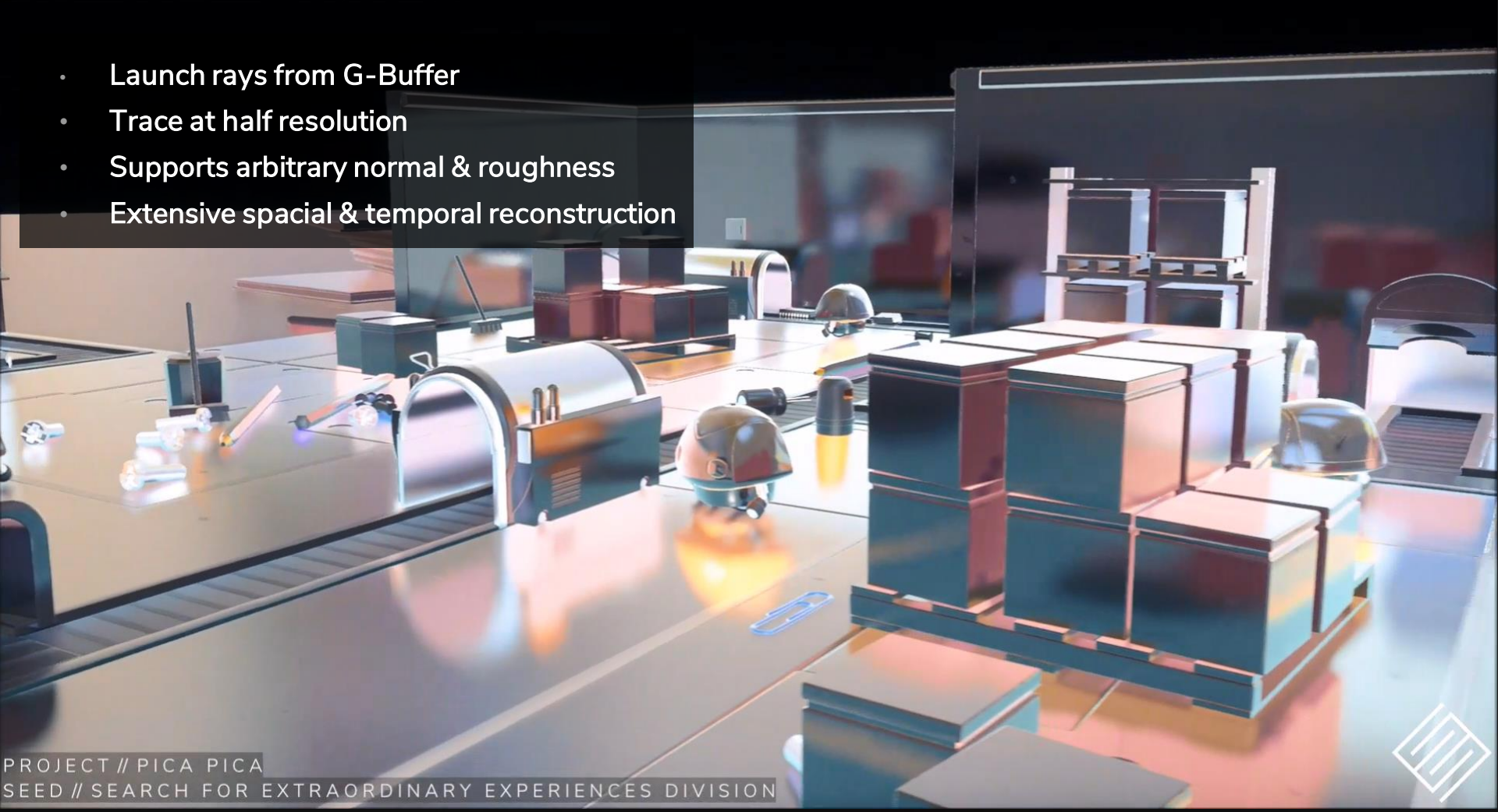
Reflections

”It Just Works”: Ray-Traced Reflections in ’Battlefield V’ [Deligiannis 2019]

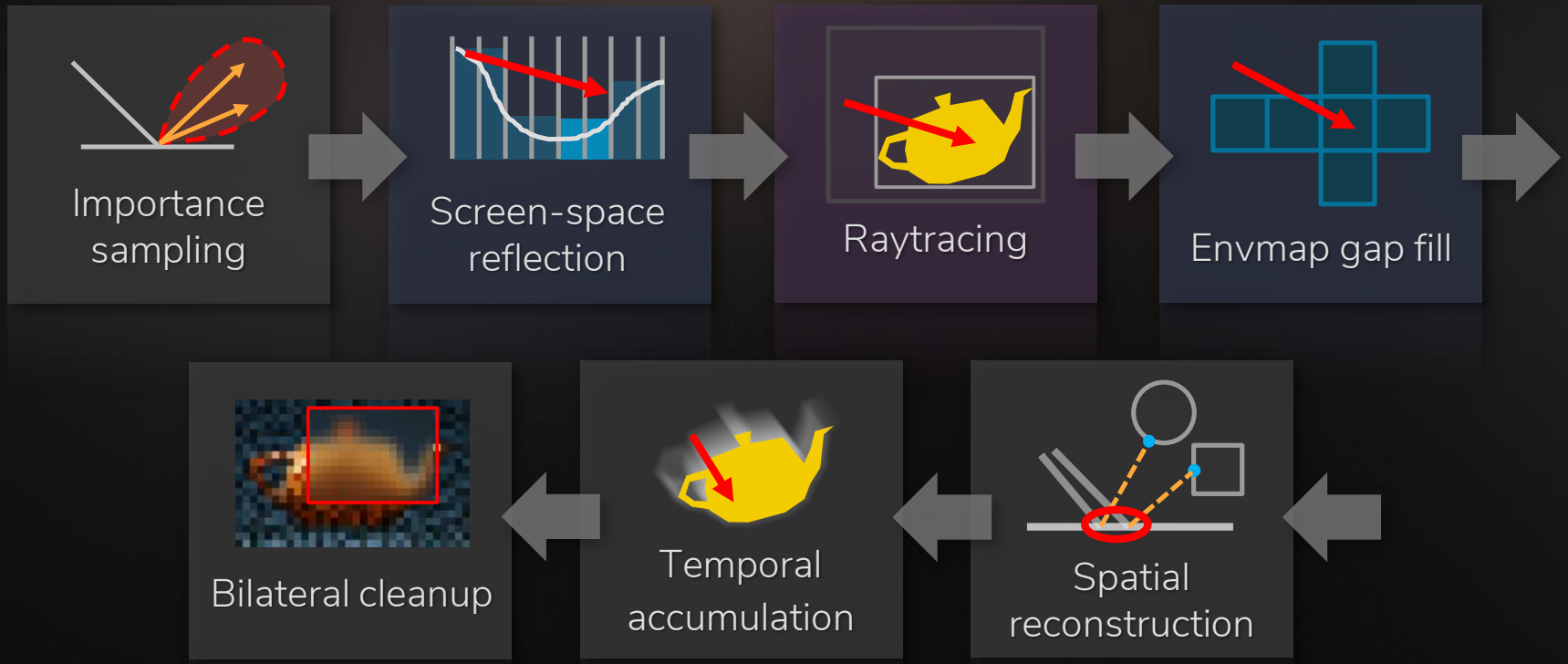
Reflections

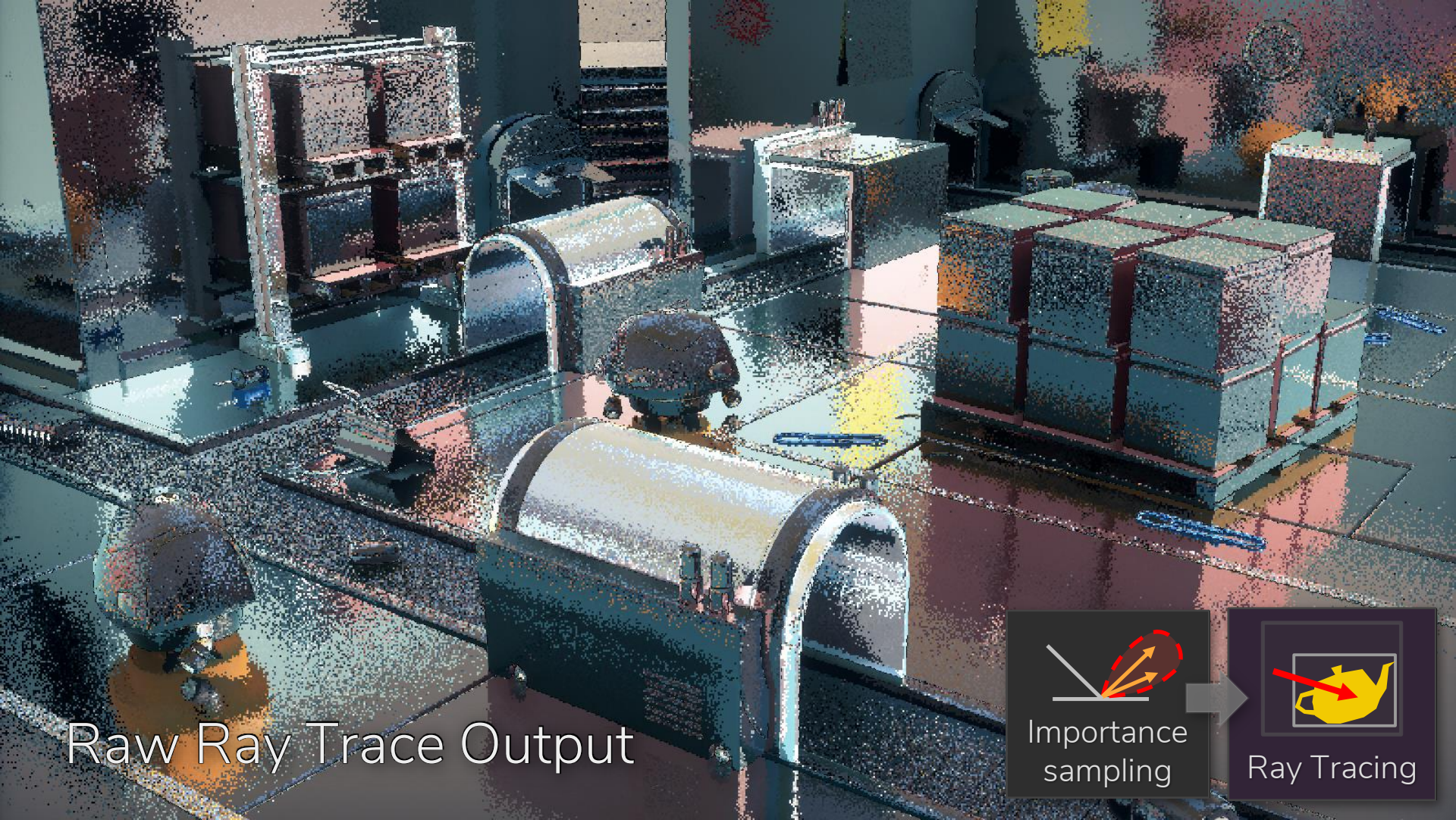


- Launch rays from G-Buffer
- Trace at half resolution
- Supports arbitrary normal & roughness
- Extensive spacial & temporal reconstruction



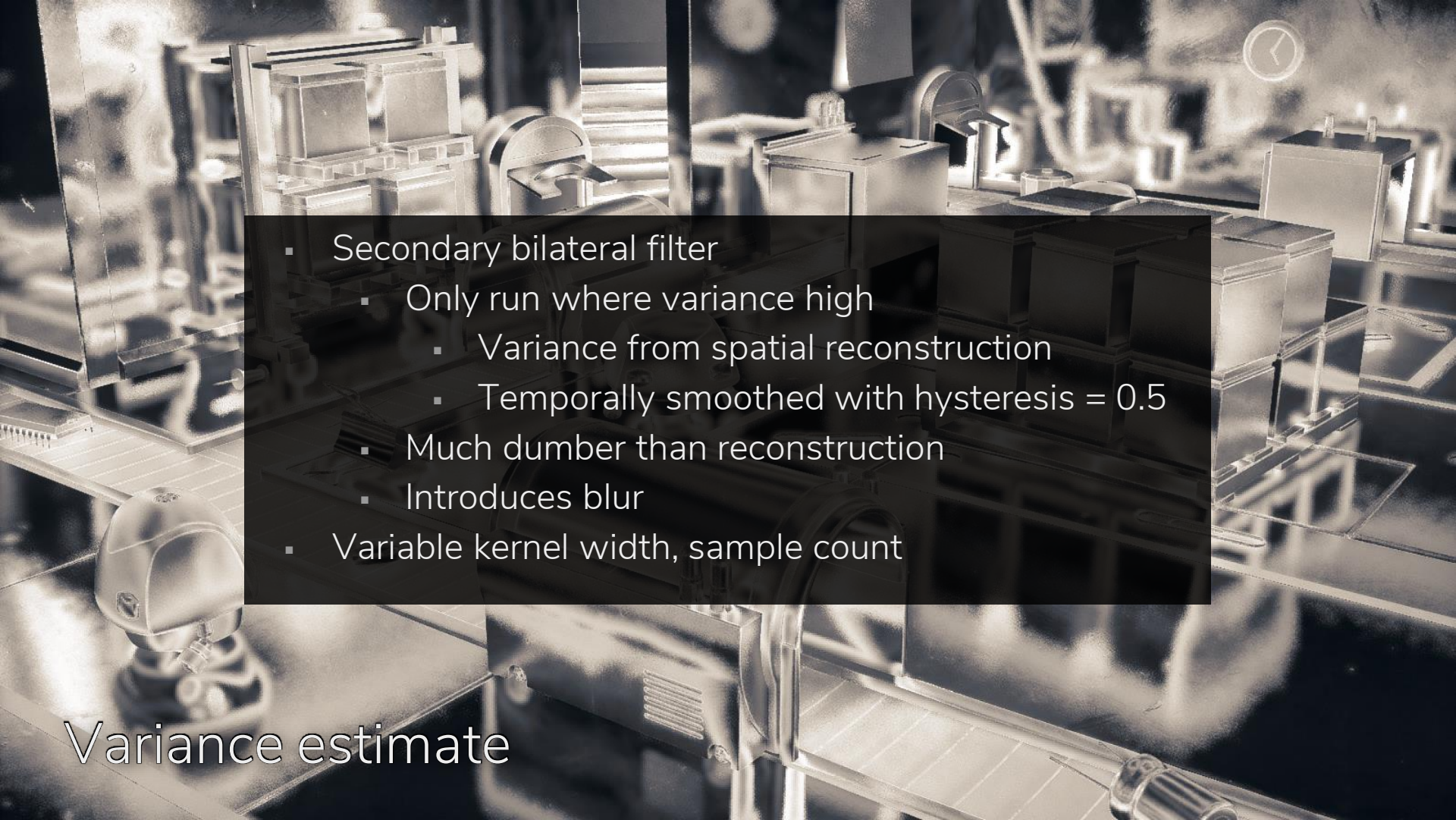
Reflection Pipeline




Raw Ray Trace Output
Importance sampling
Ray Tracing

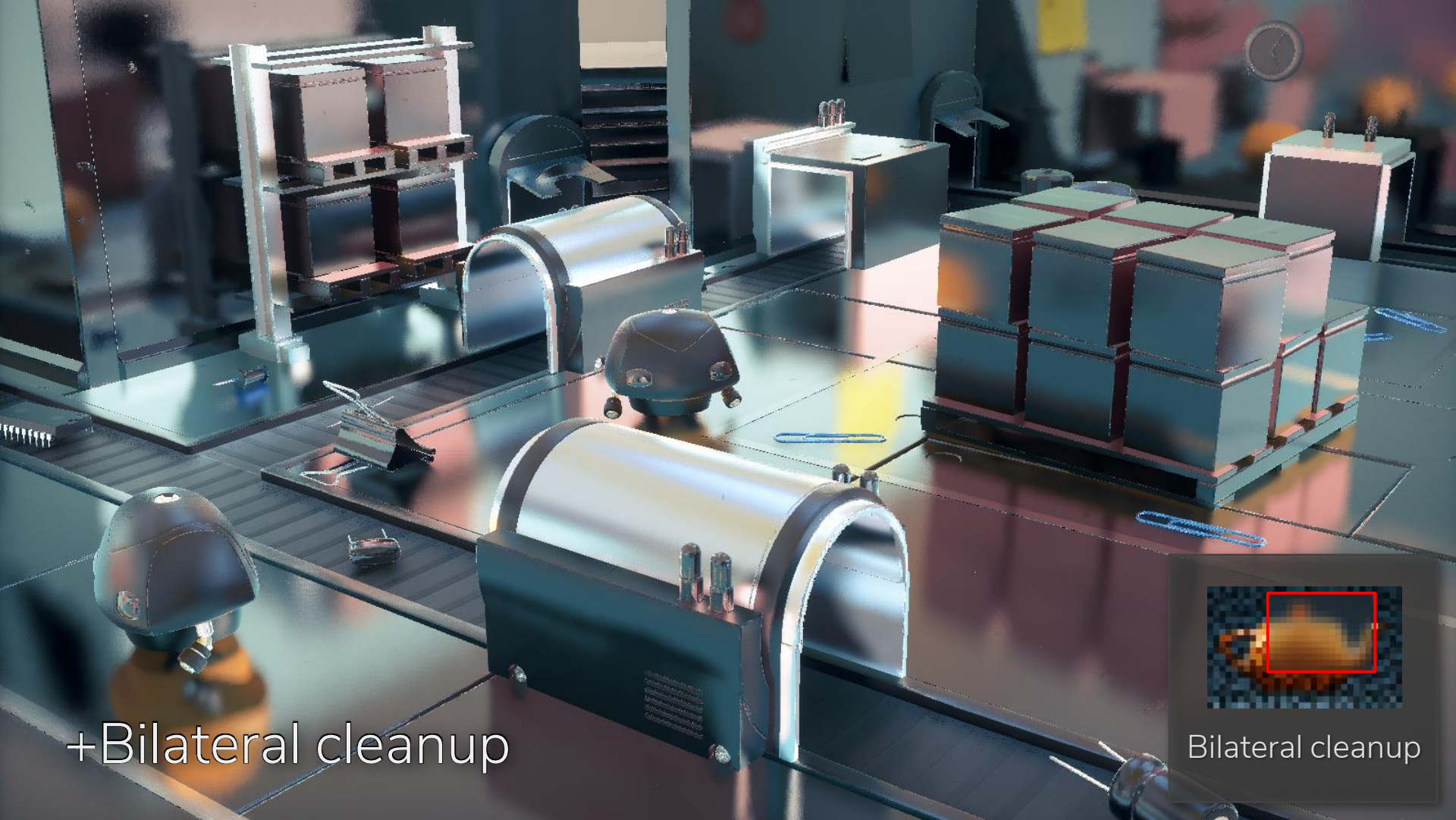

Spatial Reconstruction
Spatial
reconstruction


+Temporal Accumulation
Temporal accumulation

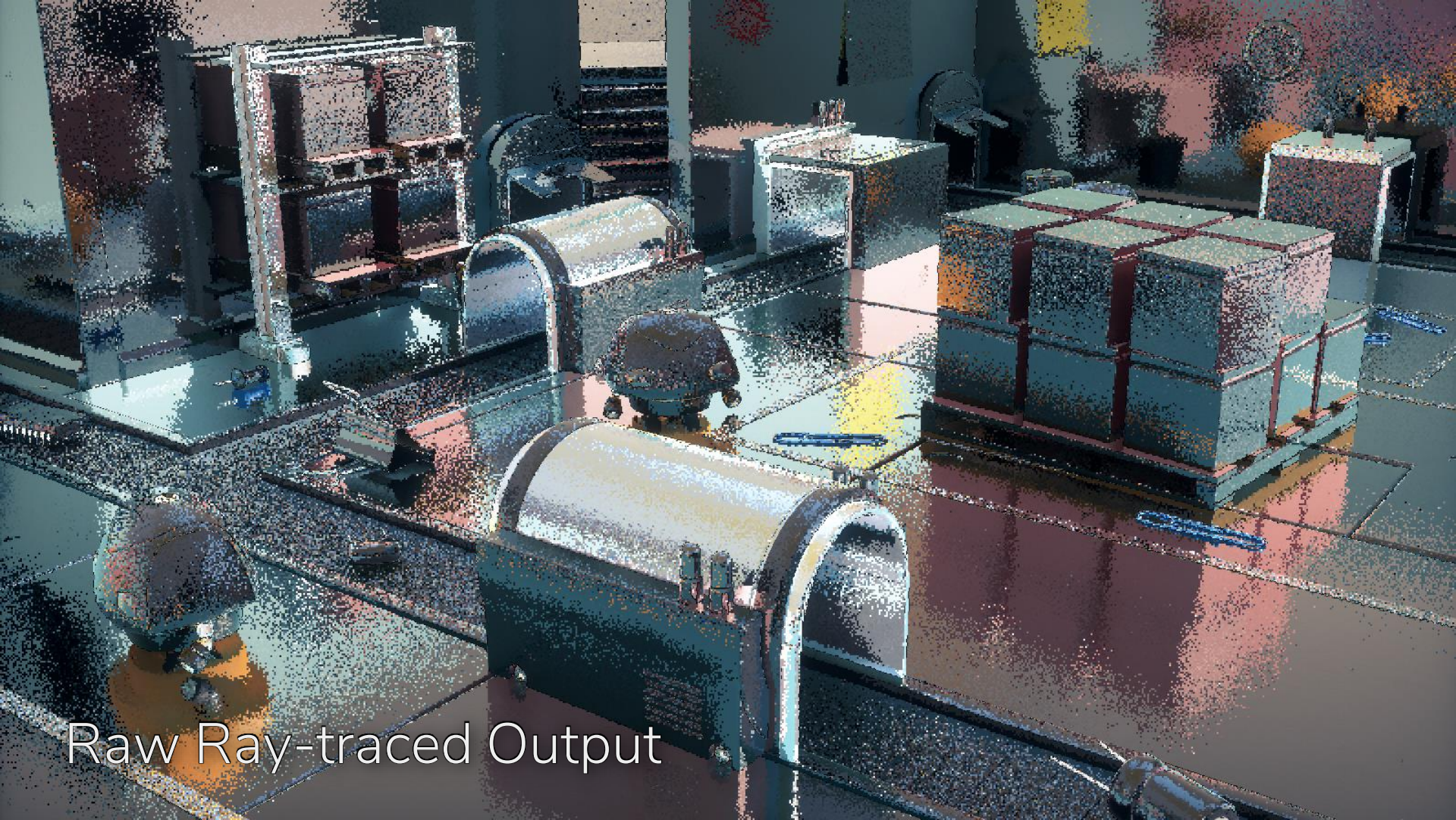


Variance estimate

- Secondary bilateral filter
  - Only run where variance high
    - Variance from spatial reconstruction
    - Temporally smoothed with hysteresis = 0.5
  - Much dumber than reconstruction
  - Introduces blur
- Variable kernel width, sample count


+Bilateral cleanup
Bilateral cleanup


+Temporal Anti-Aliasing

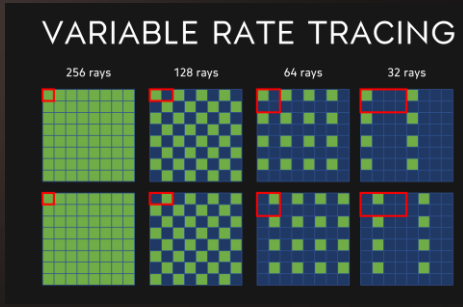

Raw Ray-traced Output

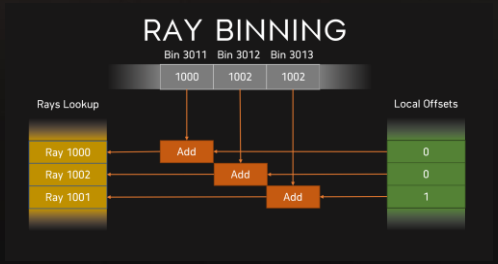
Alternatively, Hybrid RT + SSR

[Deligiannis 2019]

Hybrid RT & SSR

- Figure out which pixels can rely on screen space results
  - Otherwise, trace in world
- Performance [Deligiannis 2019]
  - *Variable Rate Tracing*
    - More rays for water & grazing angles
  - *Ray Binning*



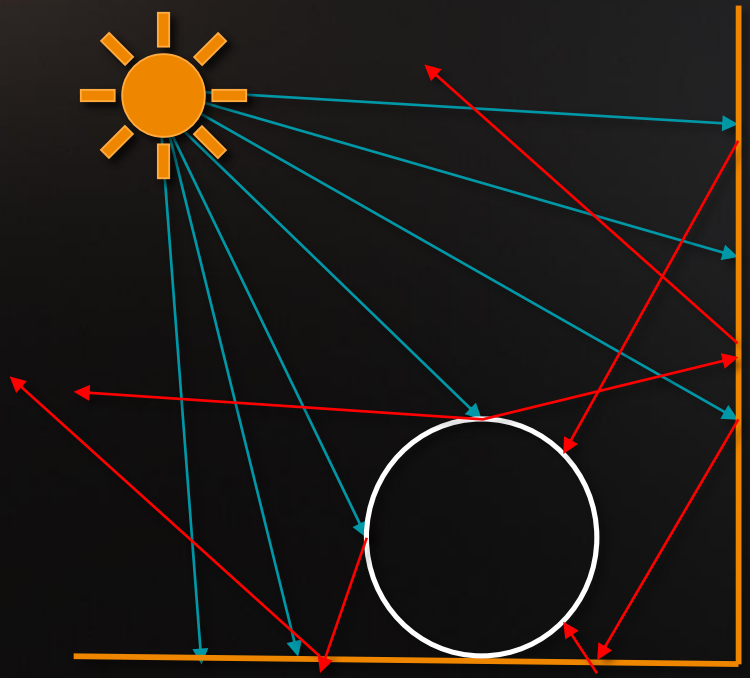


"It Just Works": Ray-Traced Reflections in 'Battlefield V' [Deligiannis 2019]

Variable Rate Tracing	Generate Rays	Ray Binning	Screen Space Hybrid	Intersect/ Material Data	Defrag	Lighting	Spatial Filter	Temporal Filter	Image Filter
0.37ms	0.19ms	0.15ms	0.36ms	1.98ms	0.08ms	0.46ms	1.45ms	0.24ms	1.00ms

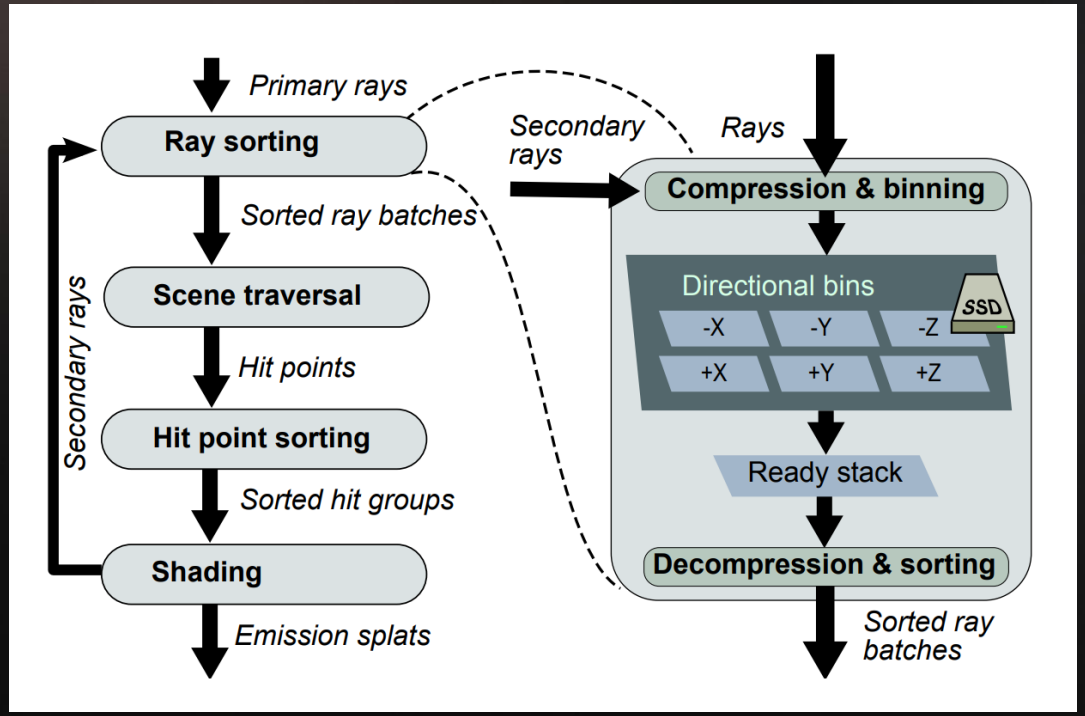
Managing Coherency

- **Coherency** is key for RTRT performance
  - Coherent → *adjacent work performing similar operations & memory access*
    - Camera rays, texture-space shading
  - Incoherent → thrash caches, kills performance
    - Reflection, shadows, refraction, Monte Carlo
- You're on your own: hardware won't take care of it for you



Managing Coherency

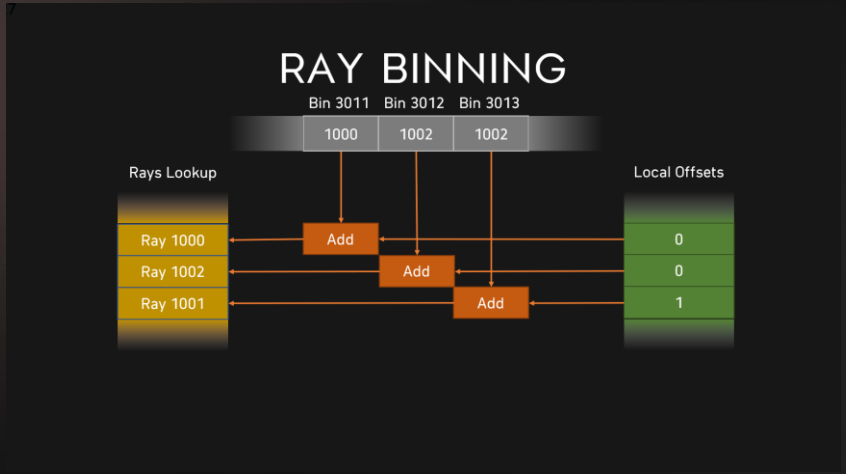
- **Inspiration from Offline**
  - Sort large out-of-core ray batches & ray-hits for deferred sharing
- A few options: [Aalto2018] [Benyoub2019] [Deligiannis 2019]
  - Use shadow maps for reflection shadows
  - Split ray tracing and shading
  - Group shading per material
  - Limit tracing on roughness



Sorted Deferred Shading for Production Path Tracing [Eisenacher 2013]

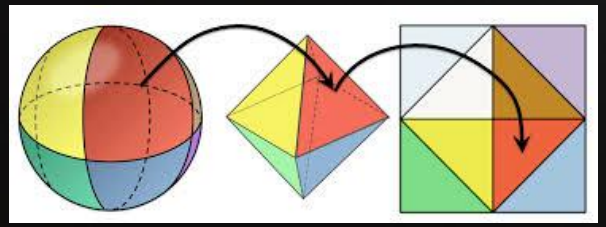
Ray Binning

Group rays that are directionally aligned to maximize coherency [Deligiannis 2019] [Benyoub 2019] [Majercik 2019]



"It Just Works": Ray-Traced Reflections in 'Battlefield V' [Deligiannis 2019]

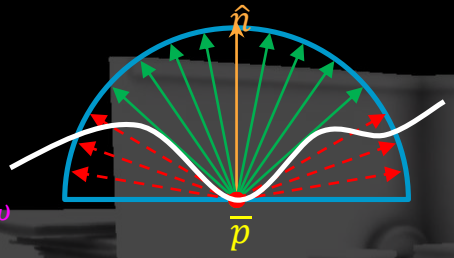
1. Split the screen in (32x32) tiles
2. Generate (random) rays
3. Sort rays in octahedral space for ray direction binning
4. For each bin, launch rays
5. Gather hit results in G-Buffer
6. Shade in Compute Shader



A Survey of Efficient Representations for Independent Unit Vectors [Cigolle 2014]

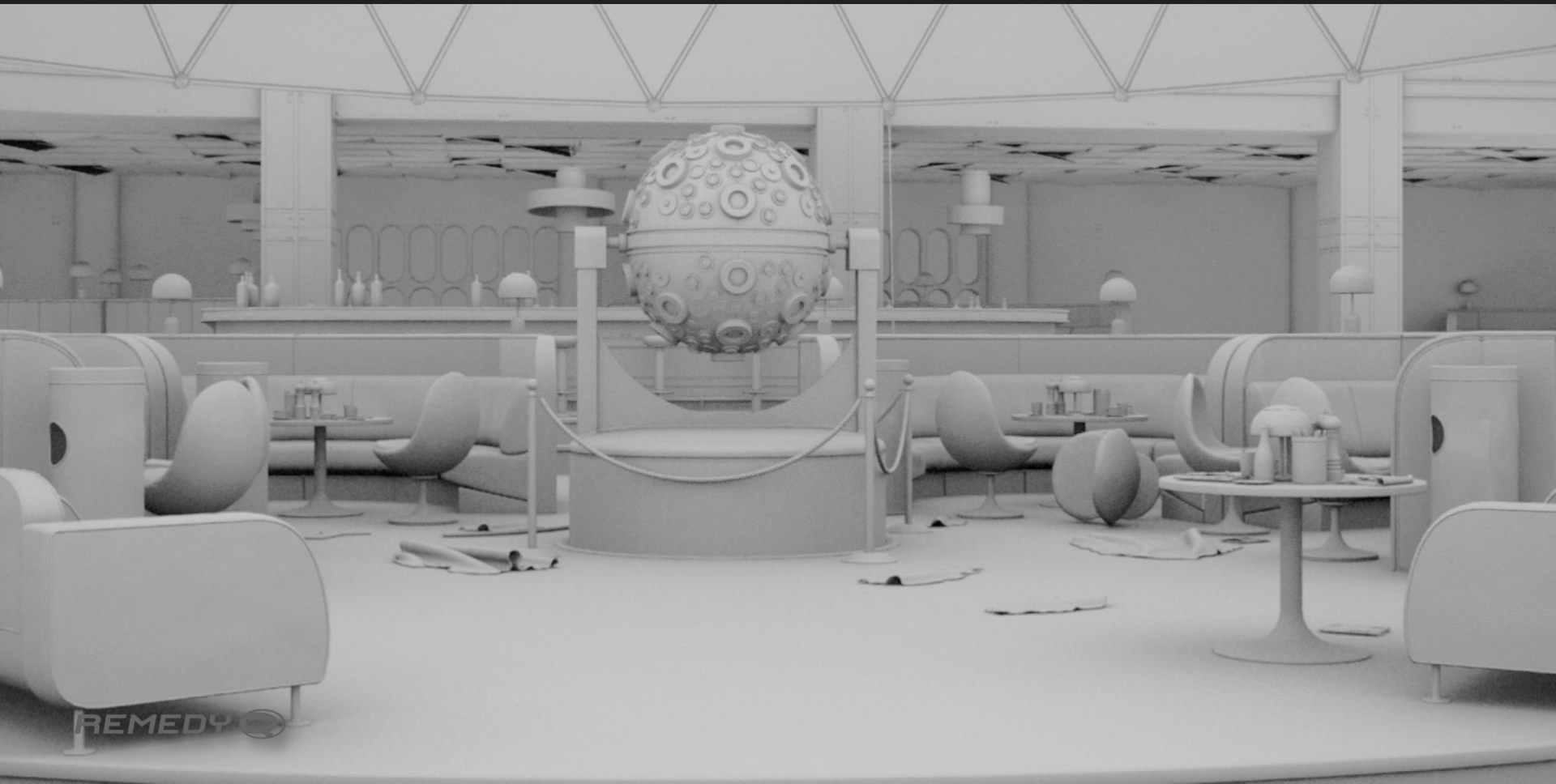
Ambient Occlusion

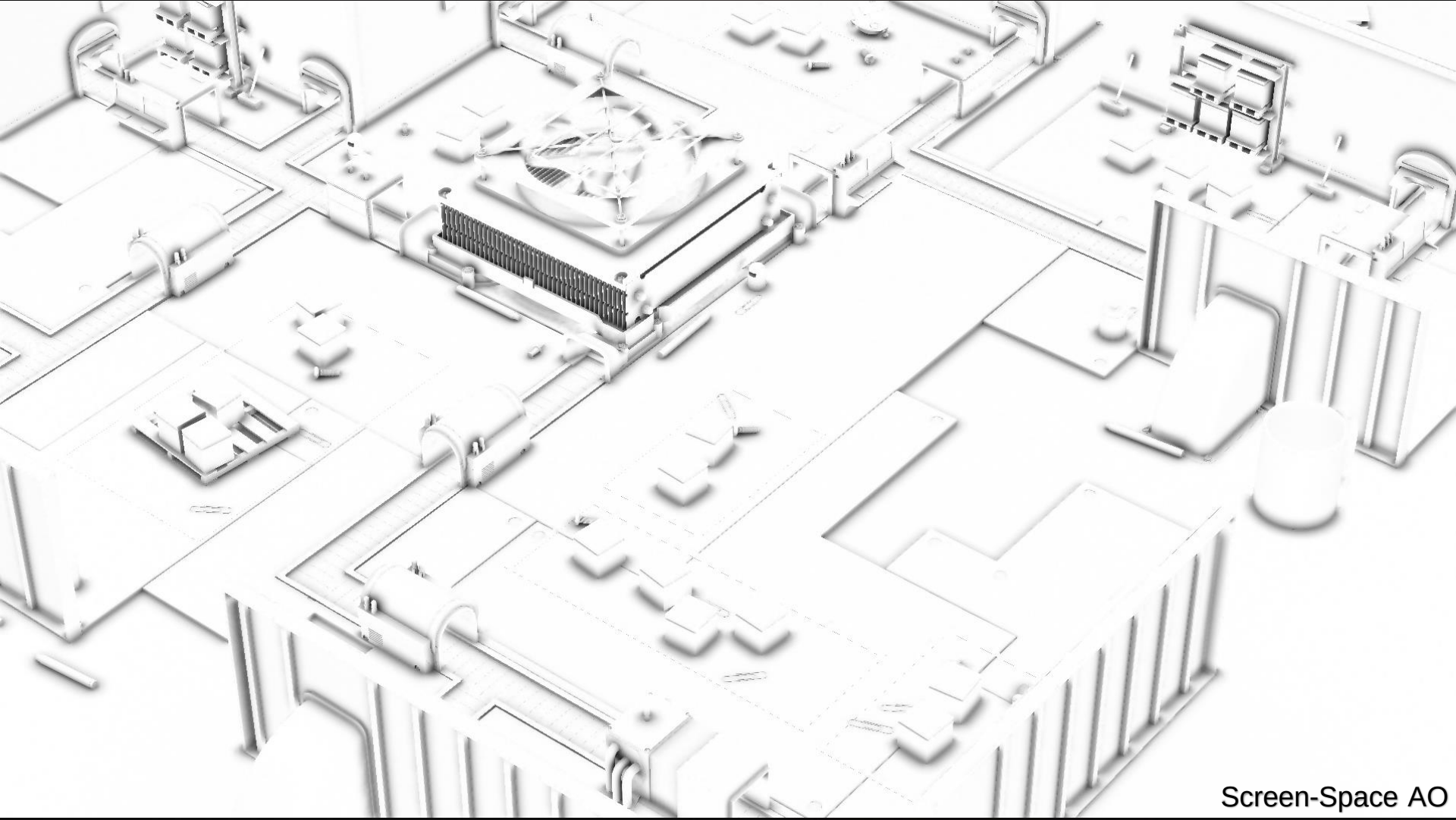
$$A_{\bar{p}} = \frac{1}{\pi}\int_{\Omega} V_{\bar{p},\hat{w}}(\hat{n}\cdot\hat{w})\,d\omega$$

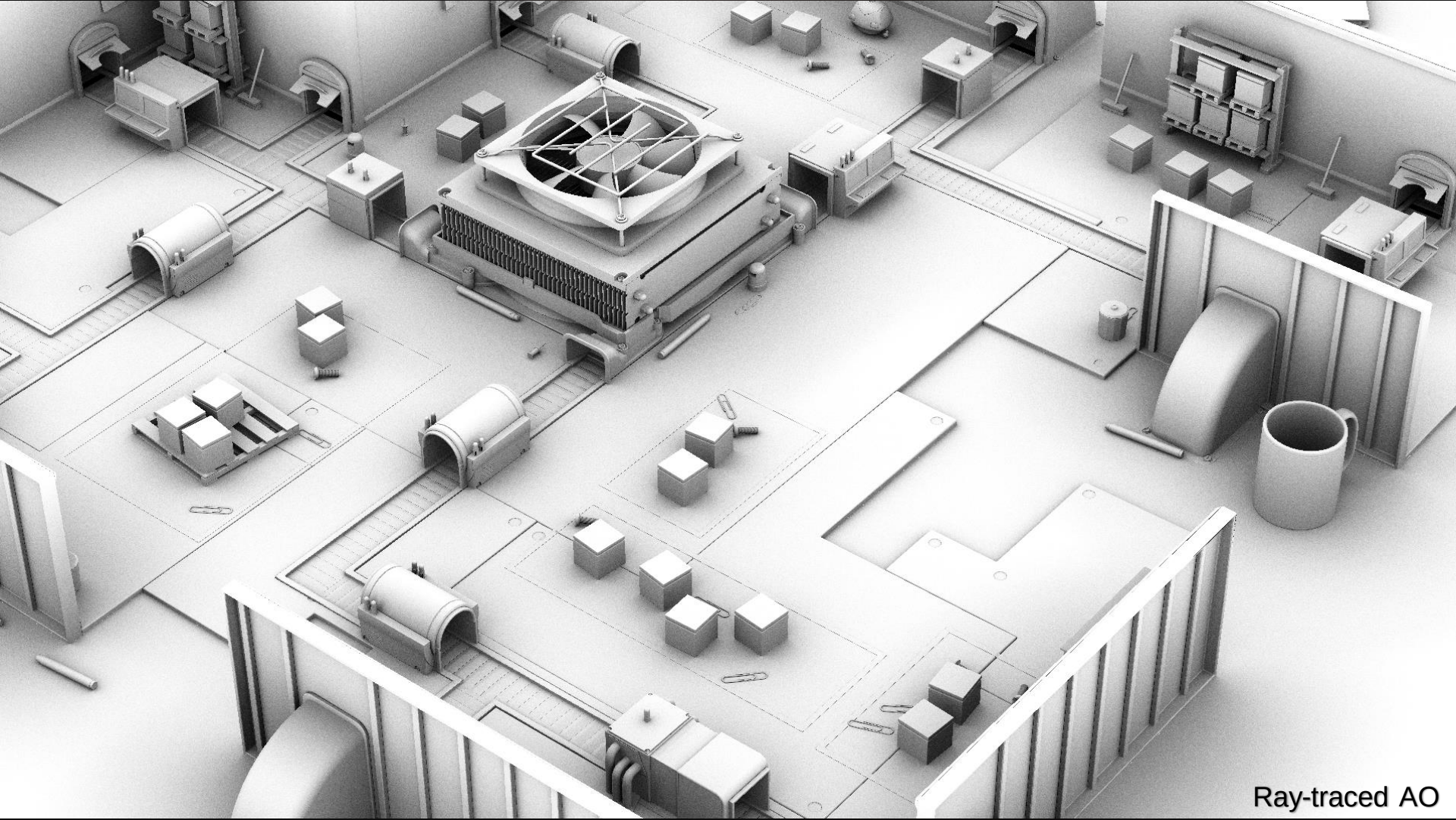


PROJECT // PICA PICA
SEED // SEARCH FOR EXTRAORDINARY EXPERIENCES DIVISION

Remedy’s Ray-traced Ambient Occlusion in Northlight Engine [Aalto2018]



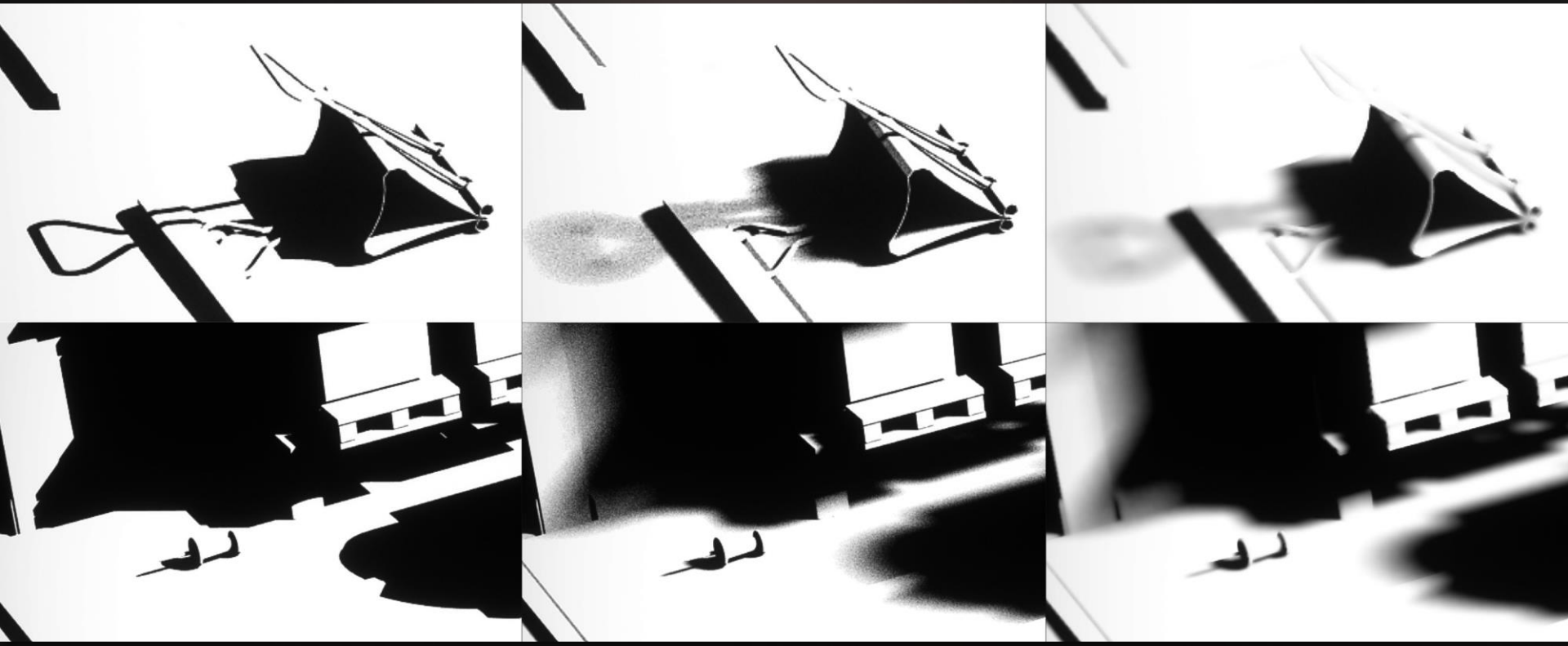

Screen-Space AO


Ray-traced AO

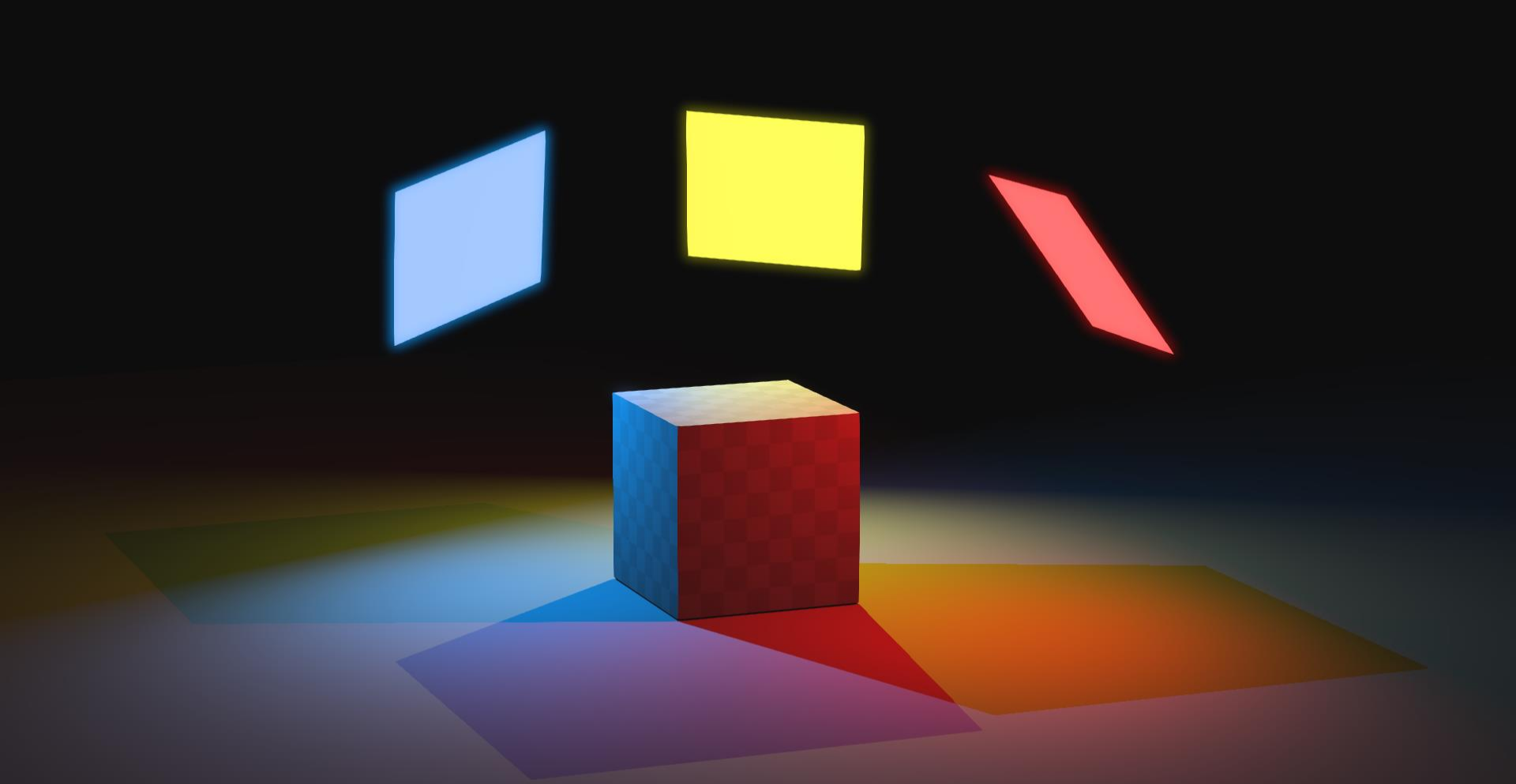

Shadows

[https://docs.unrealengine.com/en-US/Engine/Rendering/RayTracing/index.html]

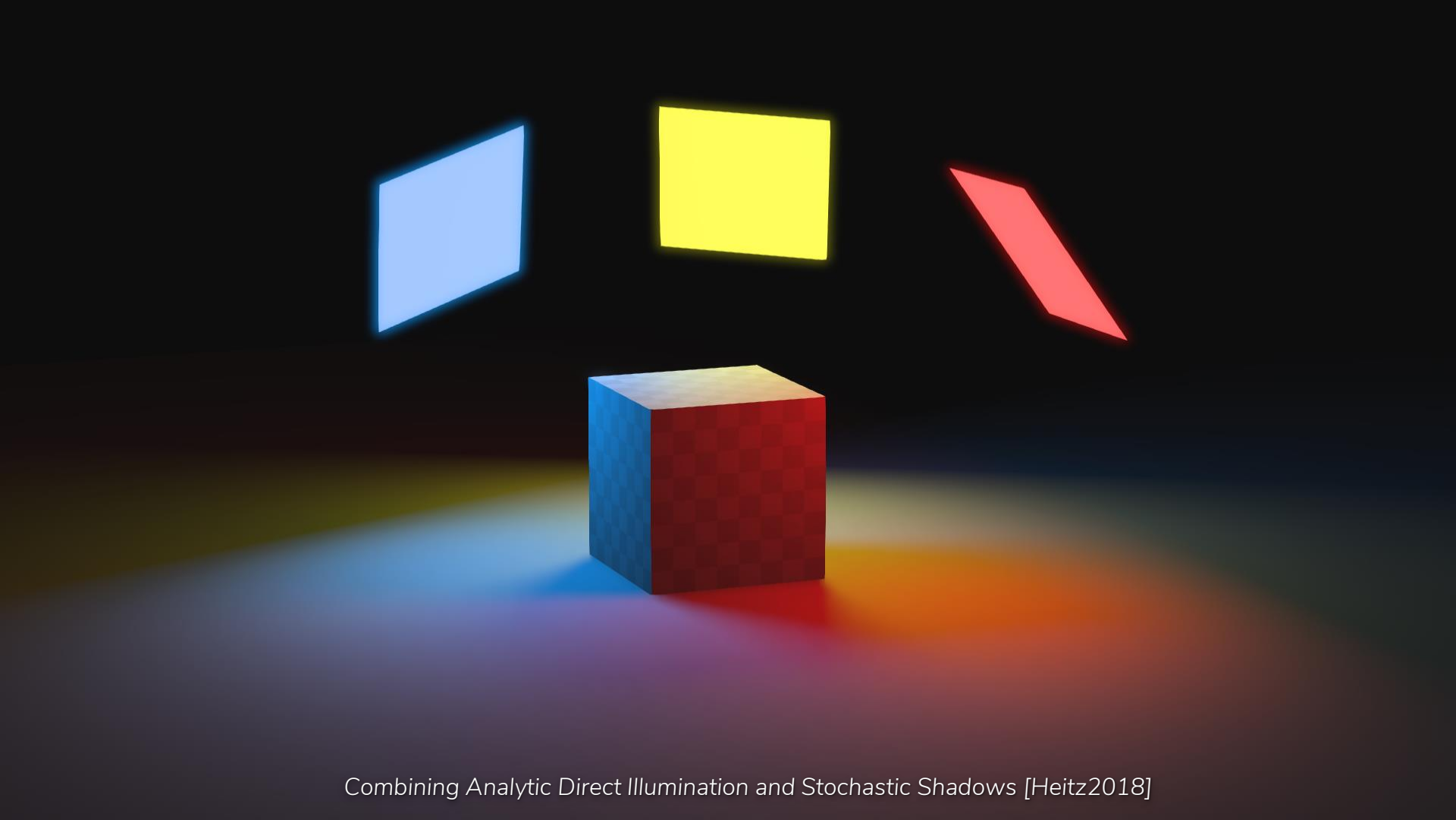
- Launch ray towards the light
- Ray misses → Not in shadow
- Soft Shadows?
  - Random cone directions. Width drives penumbra



Hard Raytraced Shadows *Soft Raytraced Shadows (Unfiltered)* *Soft Raytraced Shadows (Filtered)*



Combining Analytic Direct Illumination and Stochastic Shadows [Heitz2018]



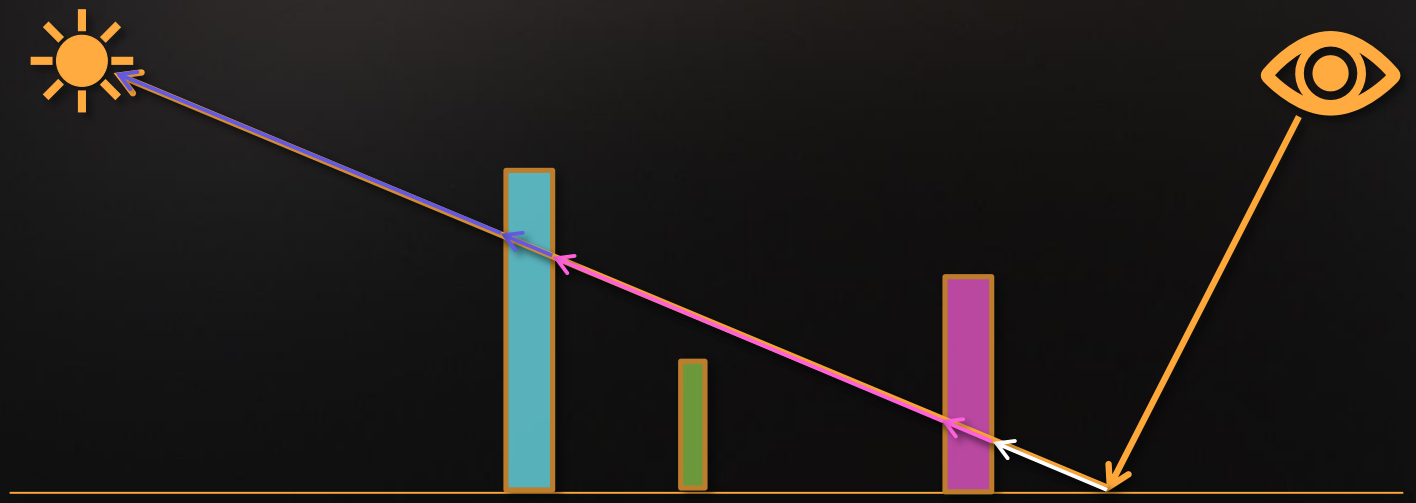
Combining Analytic Direct Illumination and Stochastic Shadows [Heitz2018]

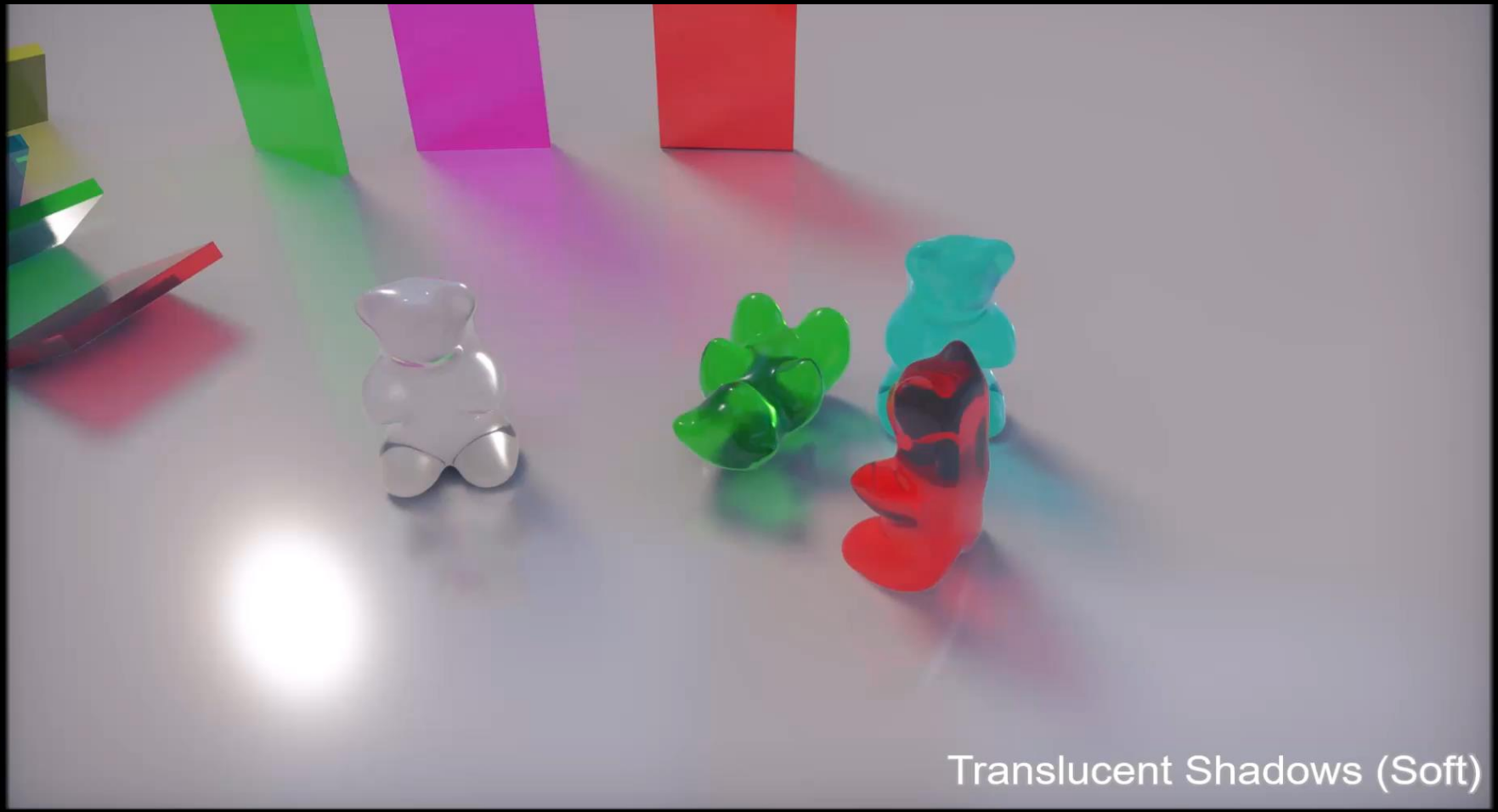
Transparent Shadows

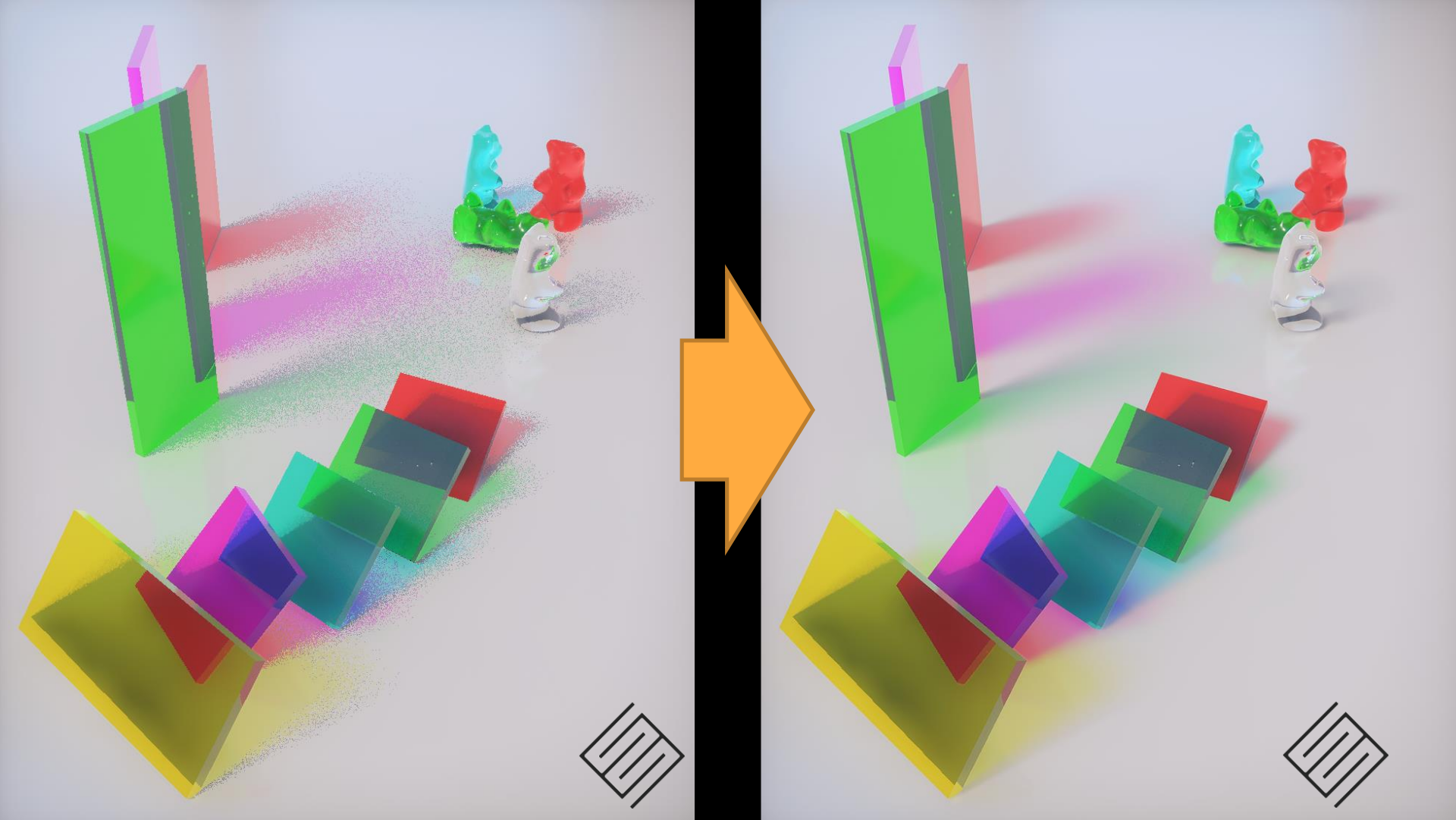
- Shadows from transparency
  - Hard to get right with raster [McGuire17]
- Let's flip it around
  - Trace towards light, like opaque
  - Accumulate absorption
    - Product of colors
    - Thin film approximation
    - Density absorption easy extension
  - Can also be soft!

Transparent Shadows (1/)

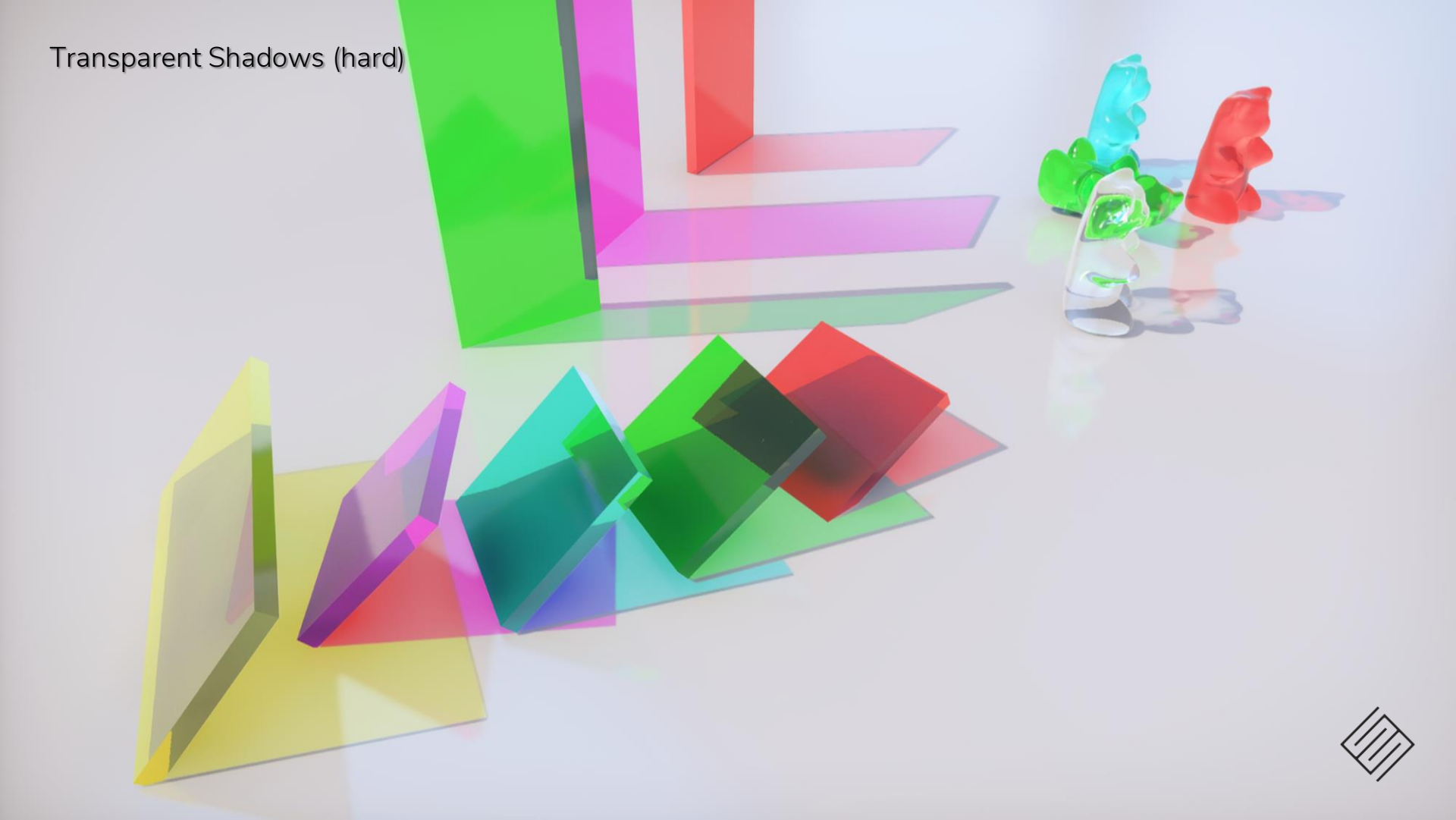
- Keep tracing until
  - Hit opaque surface or all light is absorbed or miss



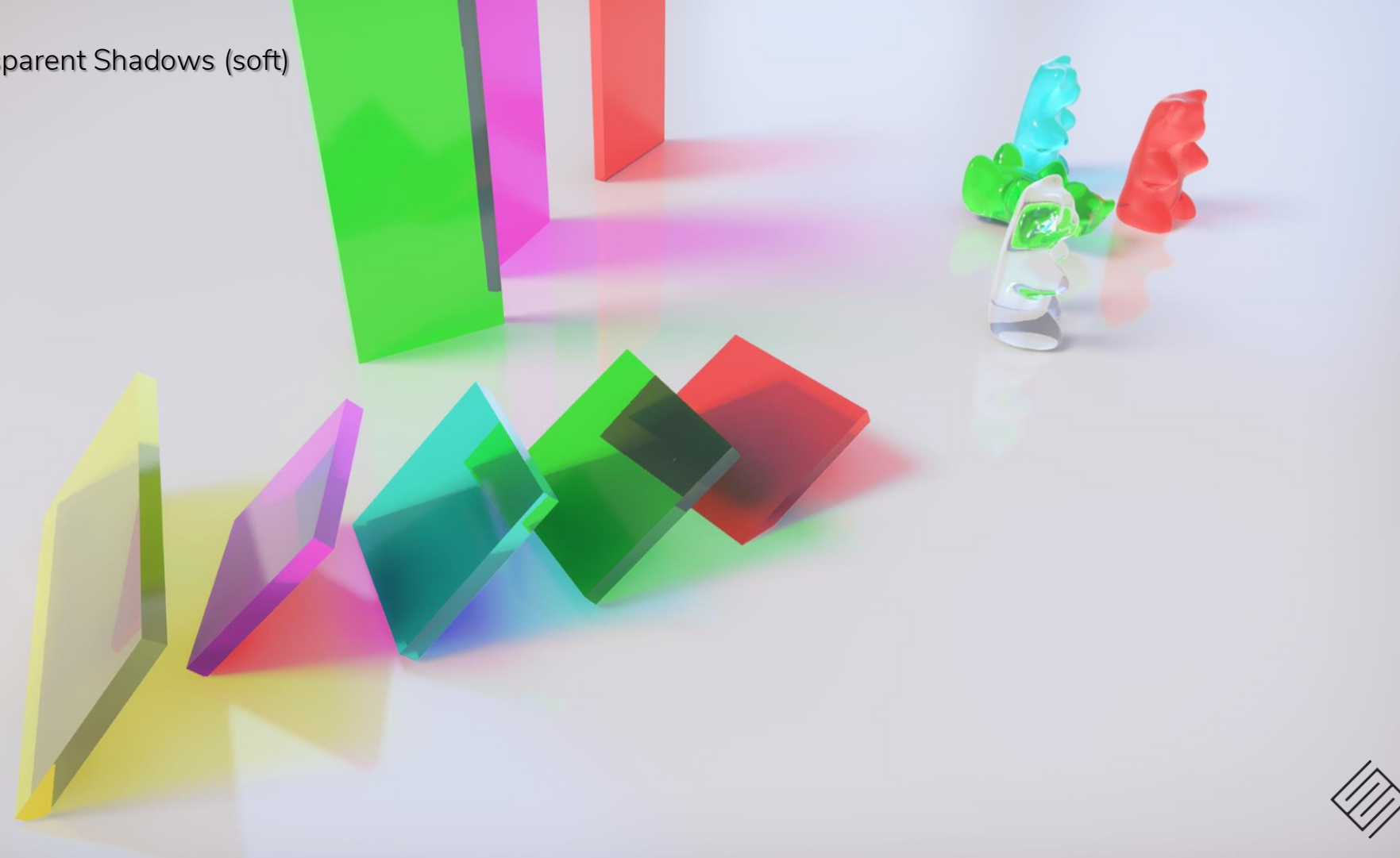

Translucent Shadows (Soft)

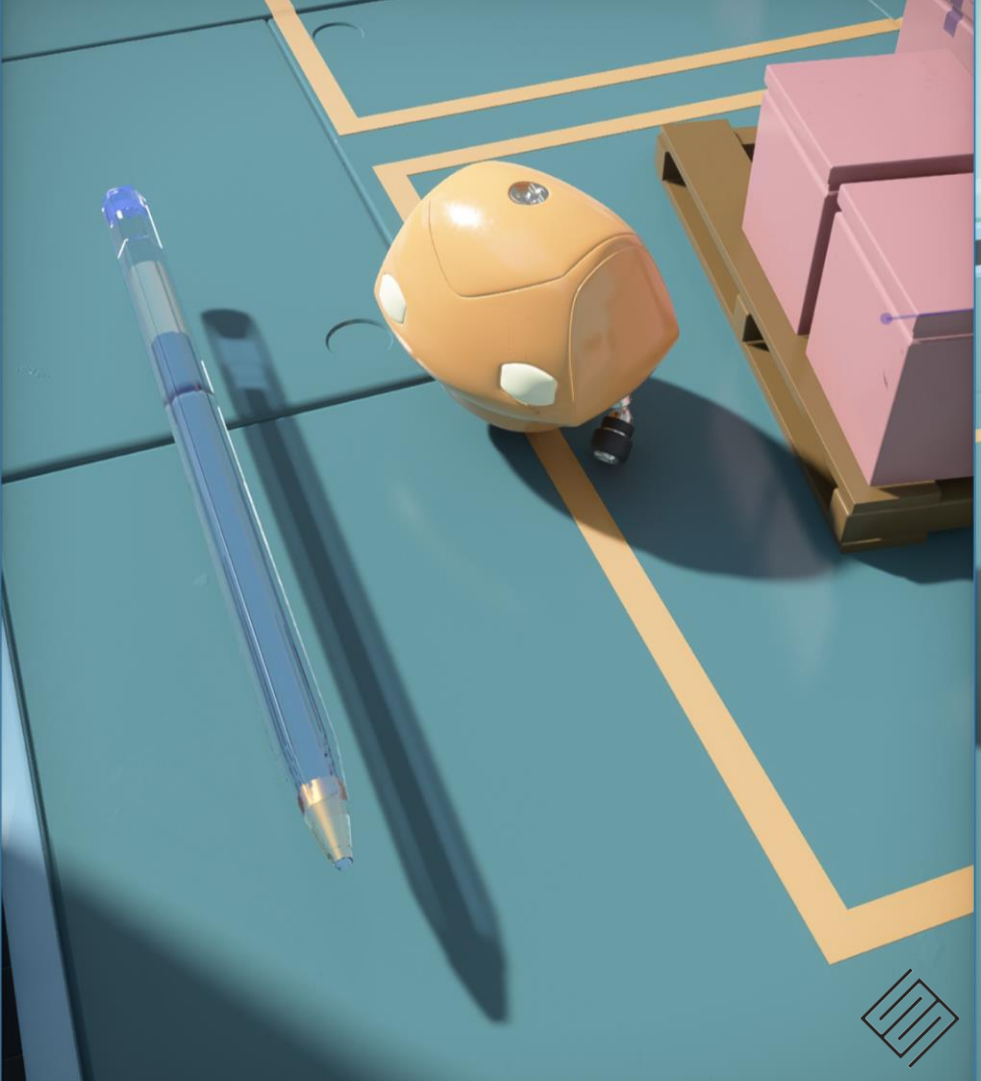


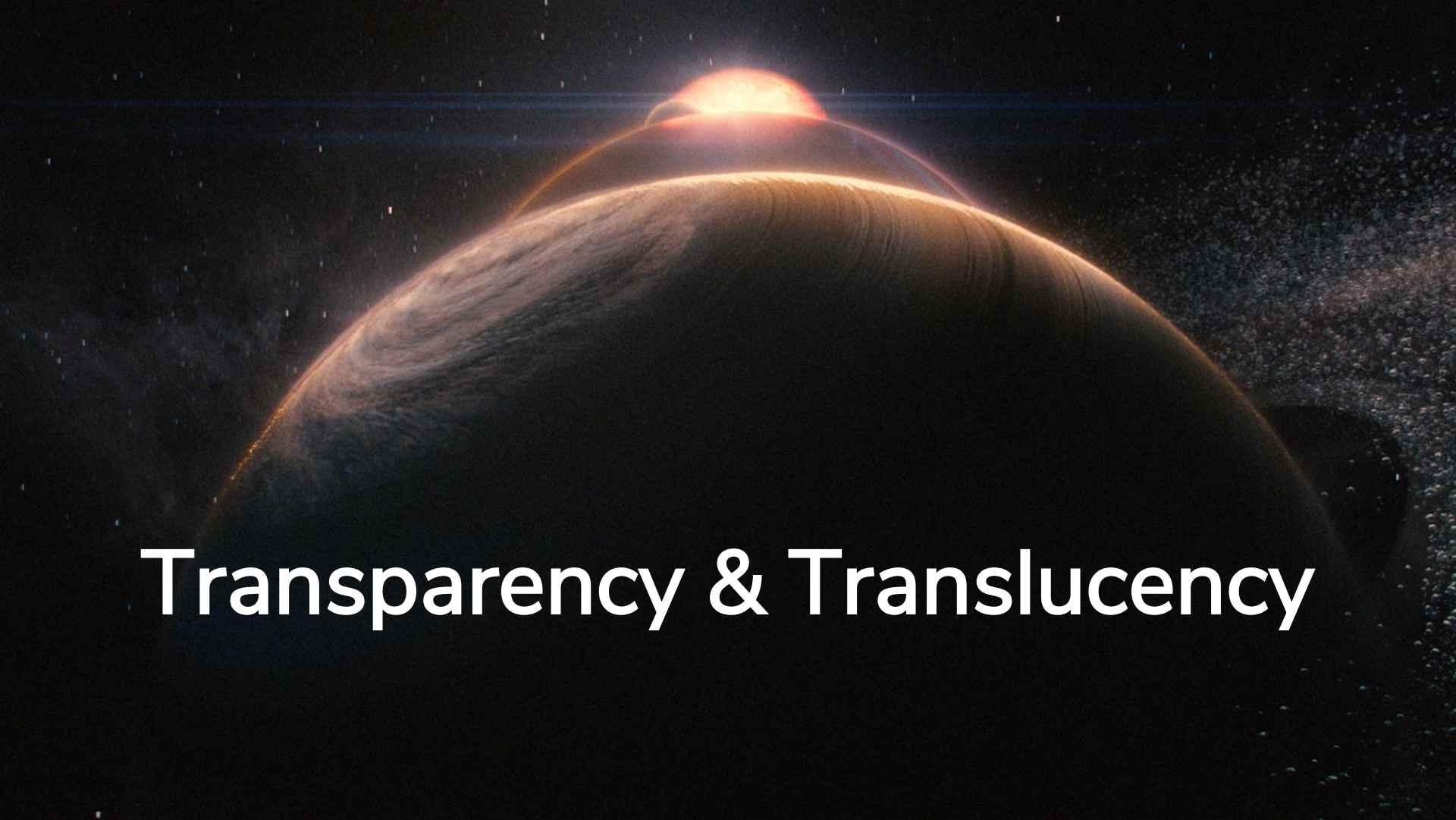
Transparent Shadows (hard)



Transparent Shadows (soft)







Transparency & Translucency

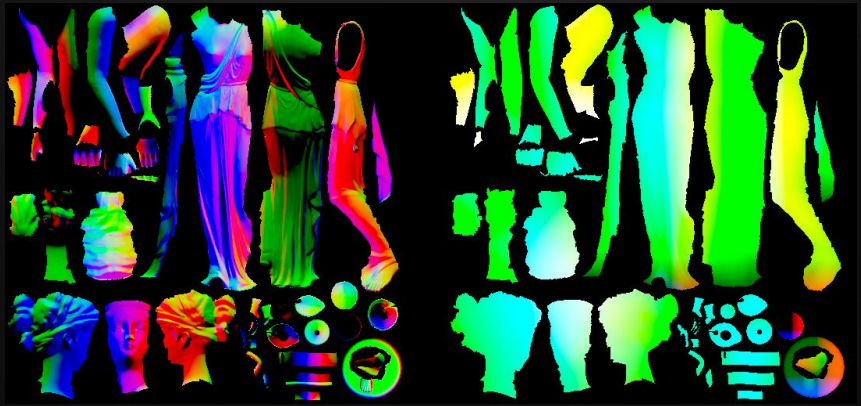
Transparency & Translucency

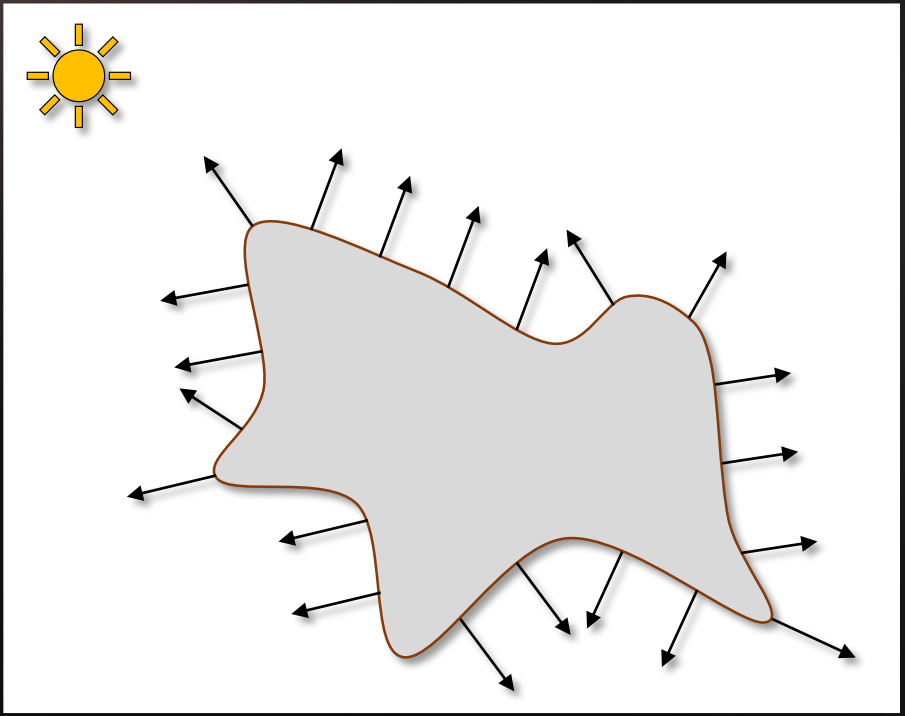
- Raytracing enables **accurate light scattering**
- **Transparency**
  - Order-independent (OIT)
  - Multiple index-of-refraction transitions
  - Variable roughness, refractions and absorption
- **Translucency**
  - Light scattering inside homogeneous medium
- We do this in **texture-space**
  - Handle view-dependent terms & dynamic changes to the environment

Texture-Space Glass and Translucency

Translucency Breakdown

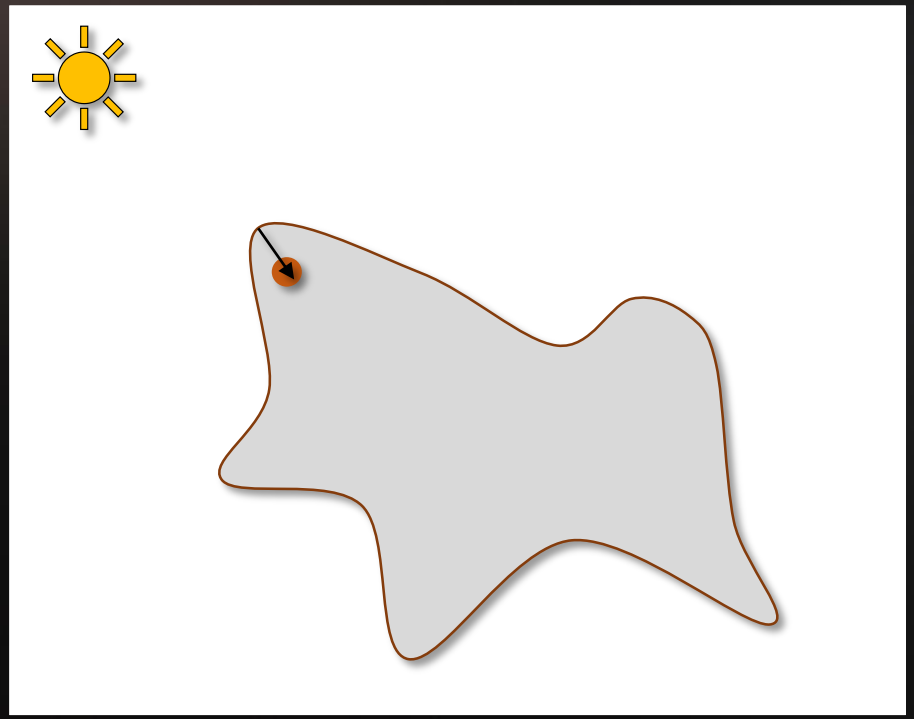
- For every valid position & normal





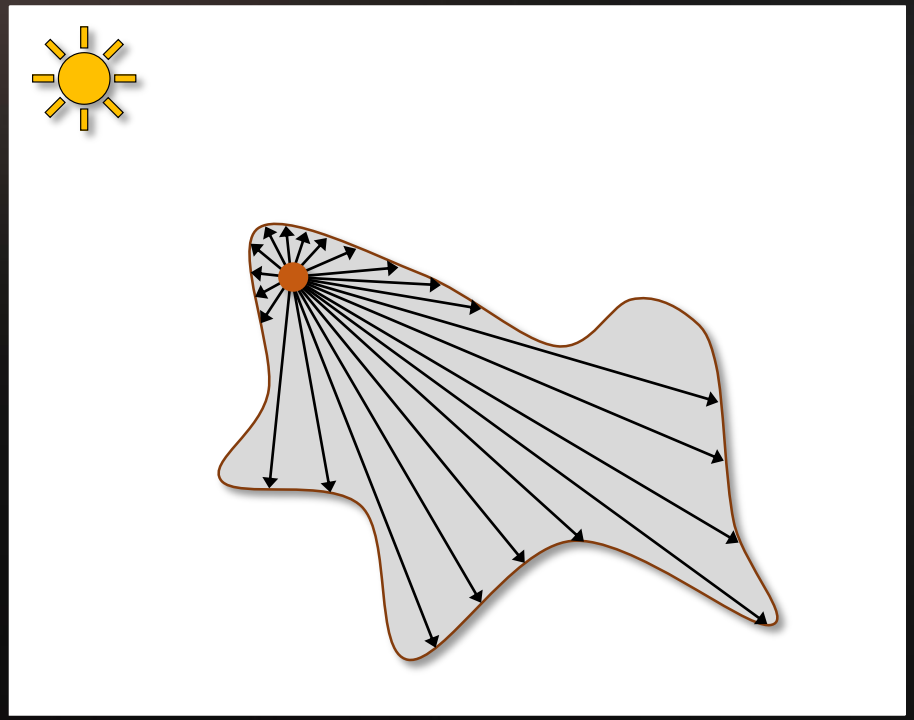
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside



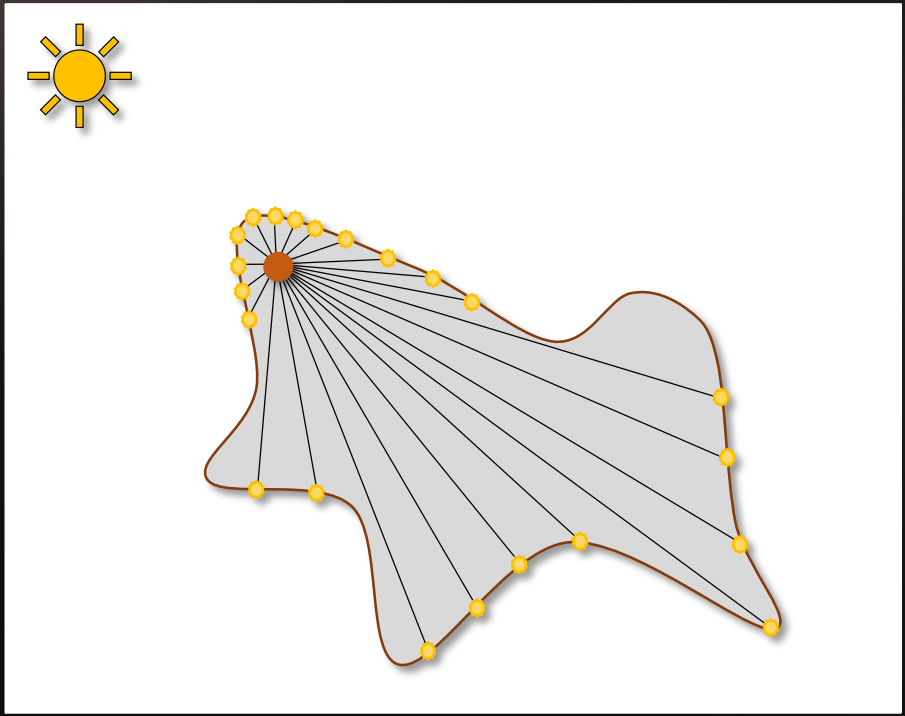
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - *(Importance-sample phase function)*



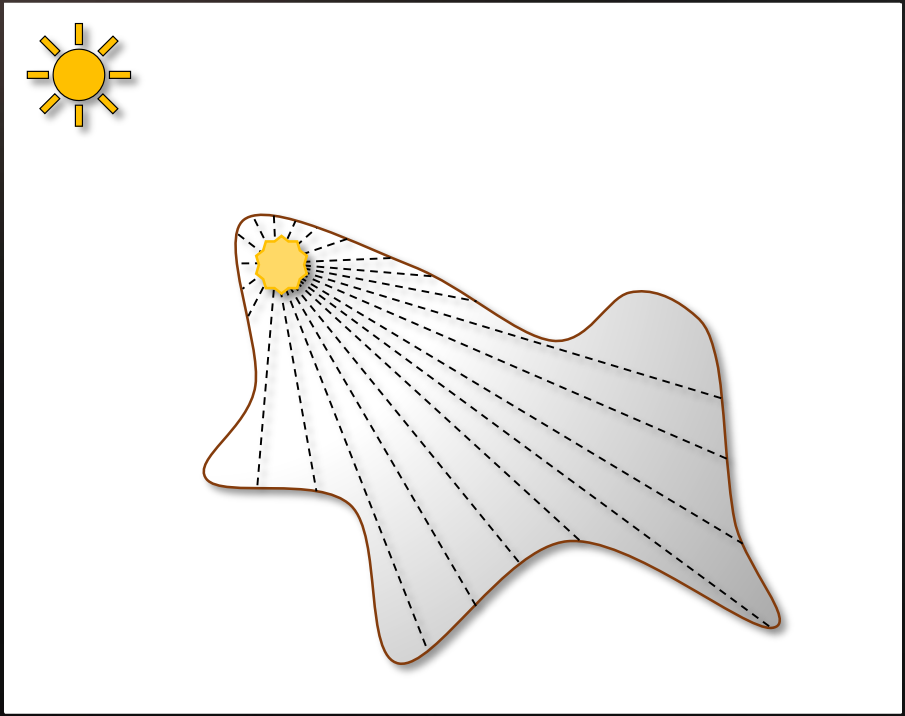
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - *(Importance-sample phase function)*
- Compute lighting at intersection



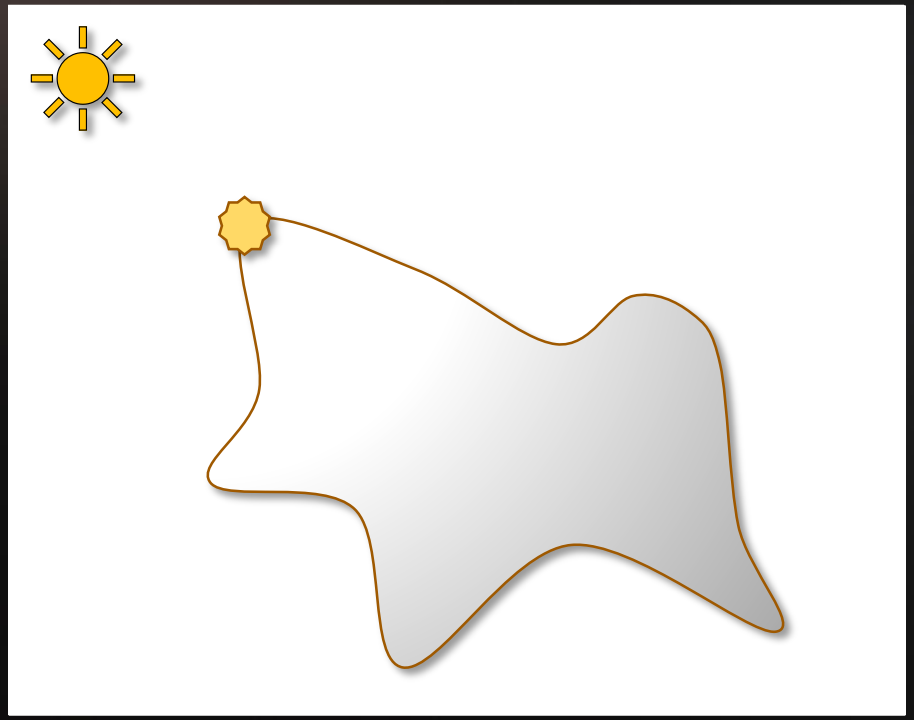
Translucency Breakdown

- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - *(Importance-sample phase function)*
- Compute lighting at intersection
- Gather all samples



Translucency Breakdown

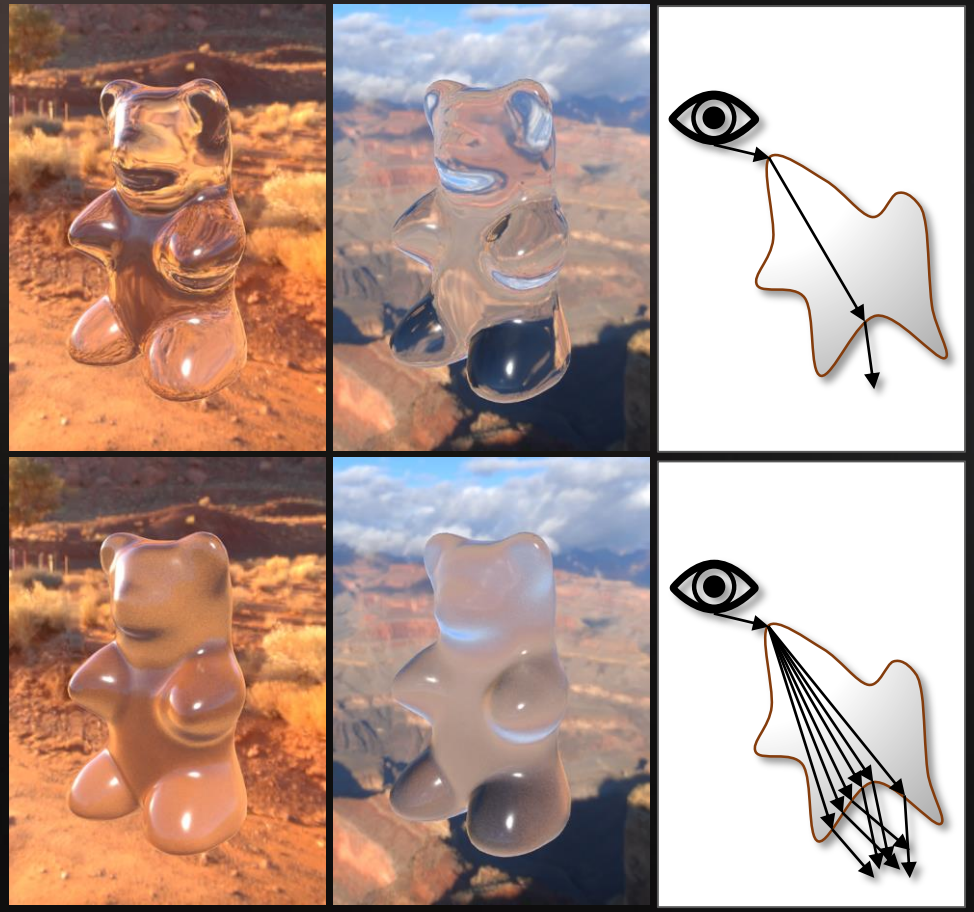
- For every valid position & normal
- Flip normal and push (ray) inside
- Launch rays in uniform sphere dist.
  - *(Importance-sample phase function)*
- Compute lighting at intersection
- Gather all samples
- Update value in texture



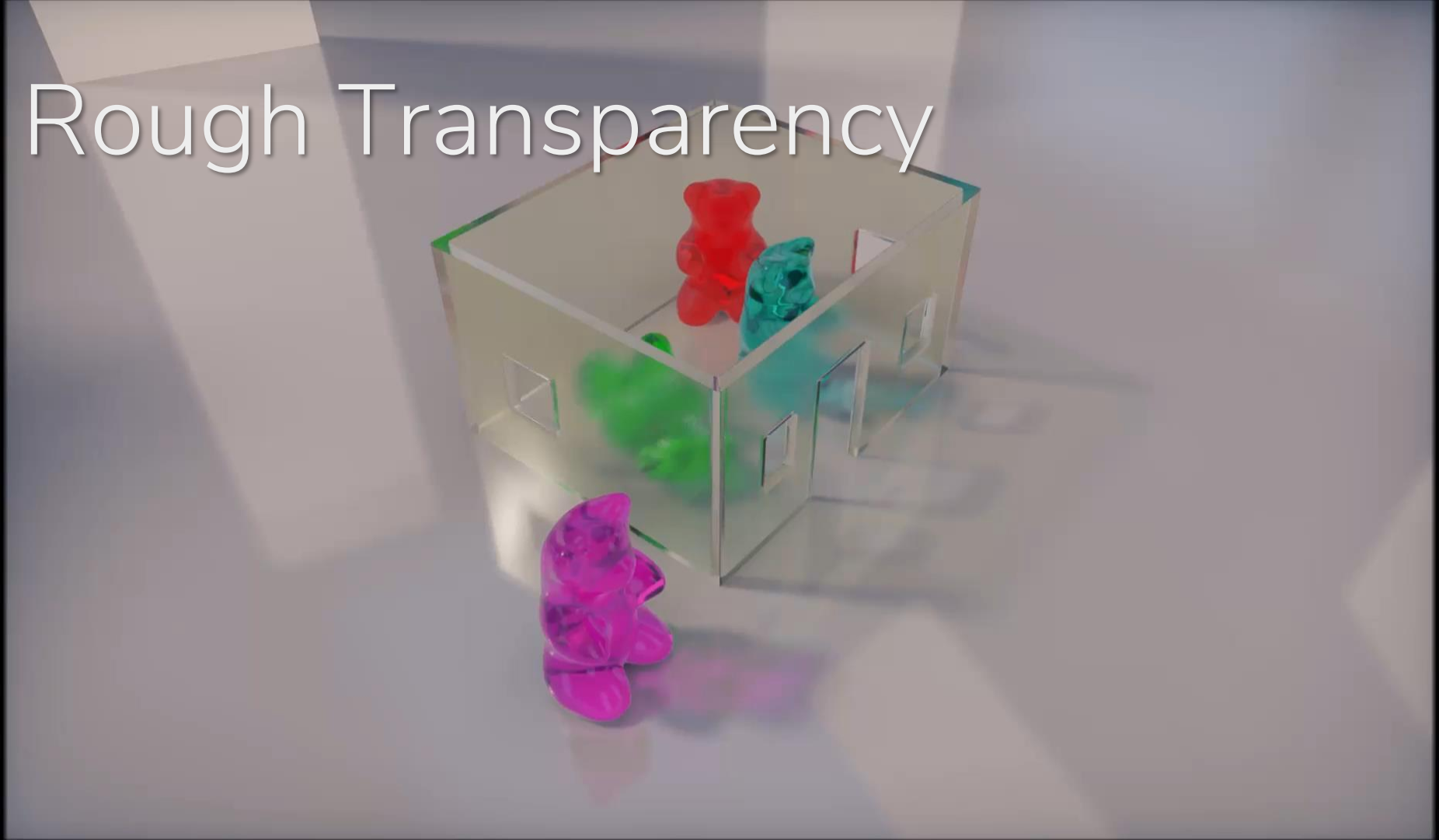
Transparency

Works for clear and rough glass

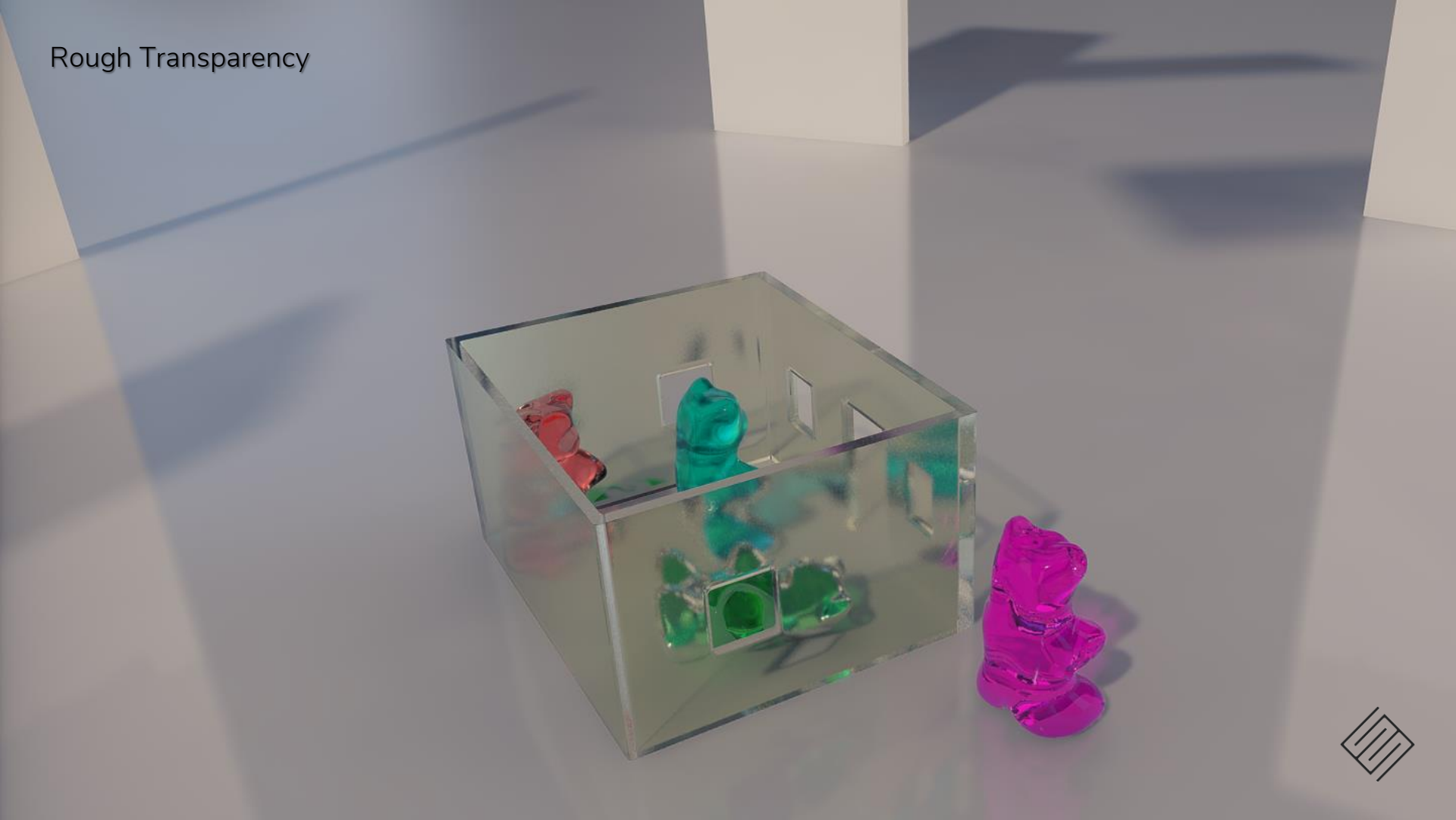
- **Clear**
  - No filtering required
- **Rough**
  - *Microfacet Models for Refraction through Rough Surfaces* [Walter2007]
  - More samples + temporal filtering
- Apply phase function & BTDF

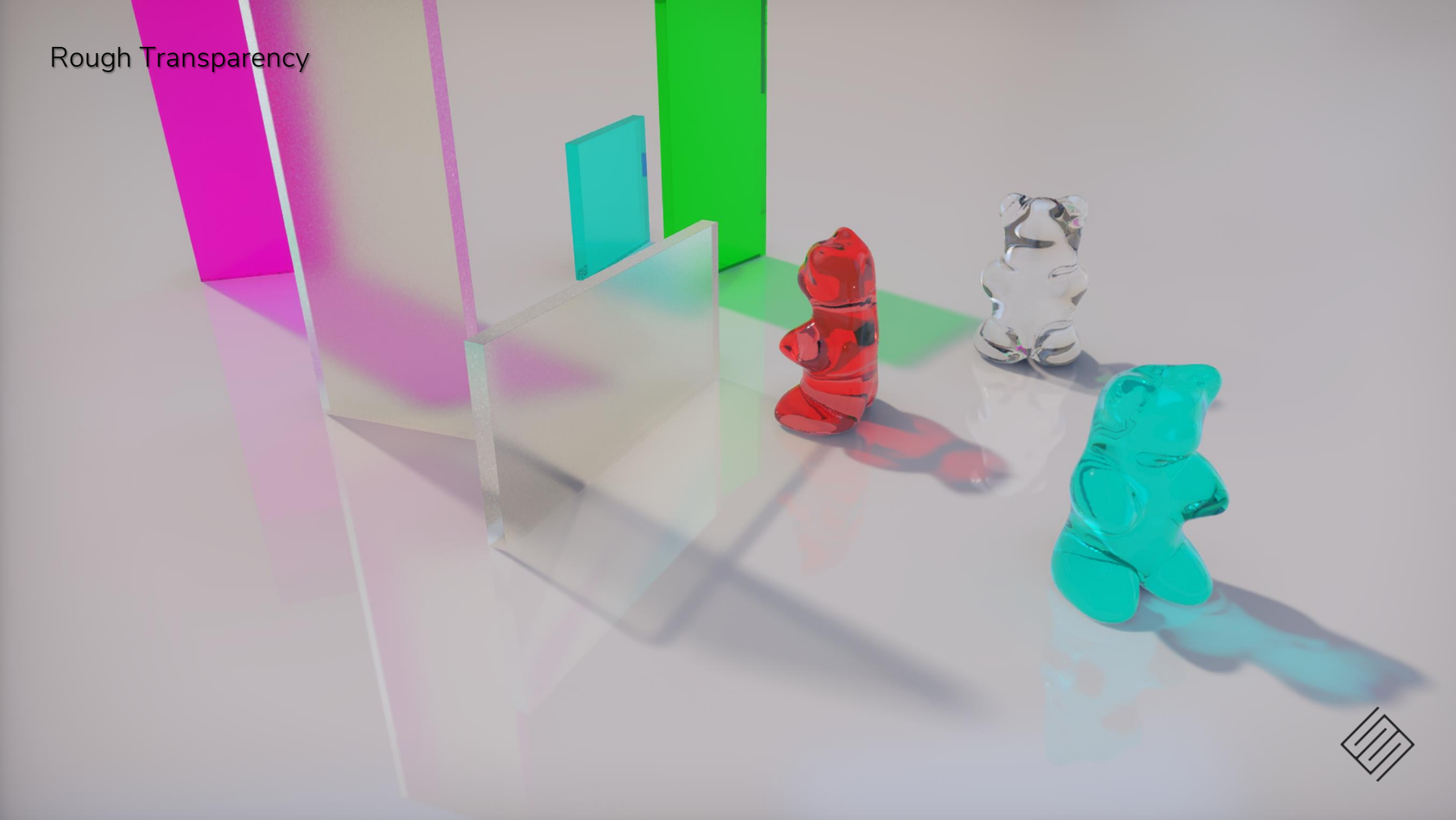


Rough Transparency

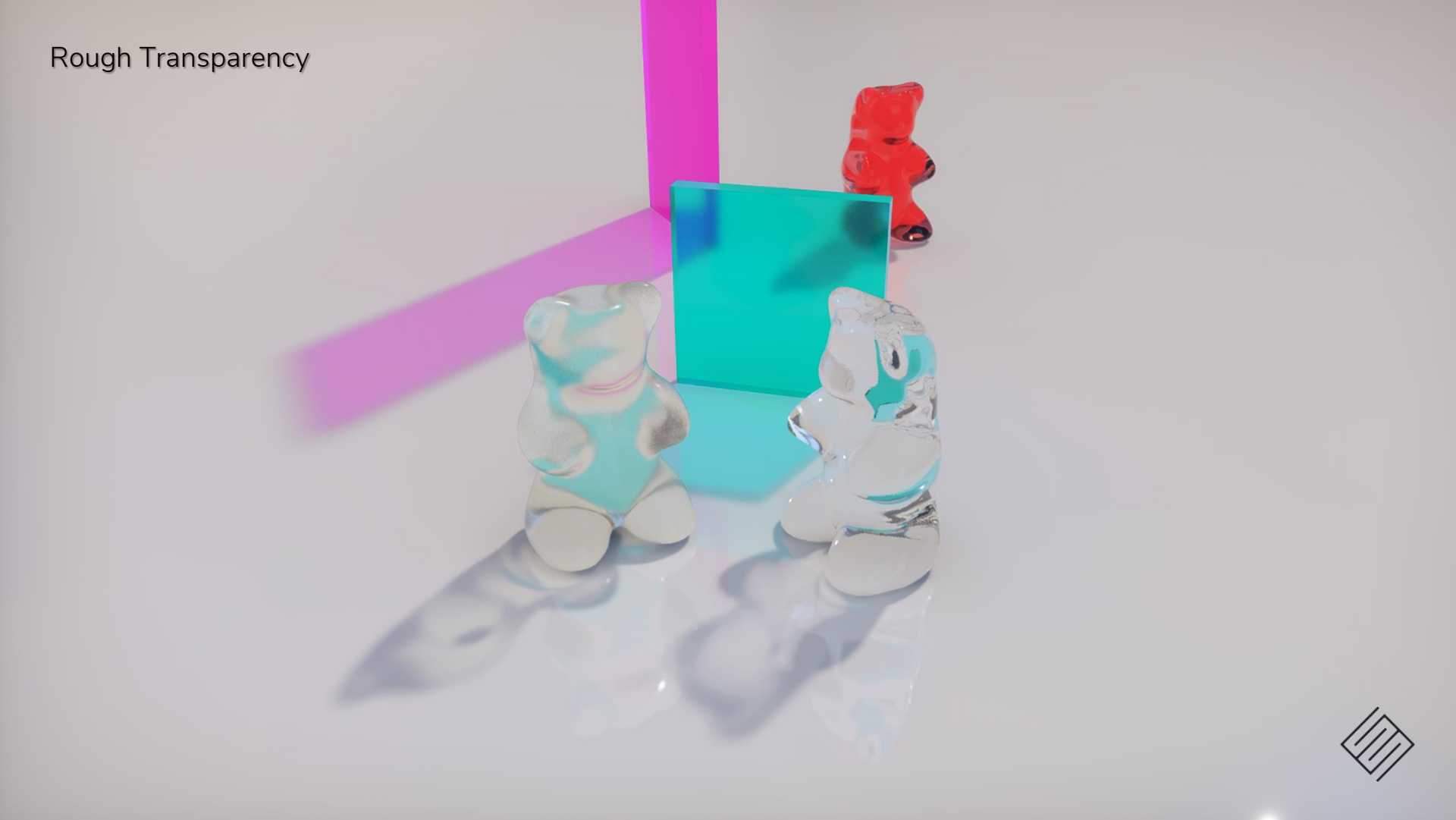


Rough Transparency




Rough Transparency

Rough Transparency



Many Lights

Many Lights

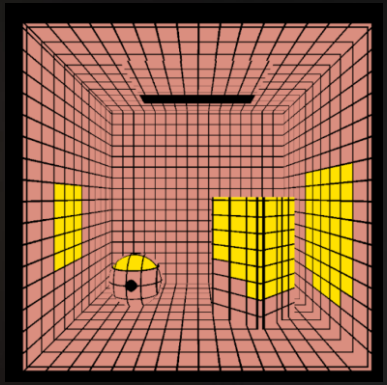
Which (most affecting) lights to choose?

Acceleration structure-based selection:

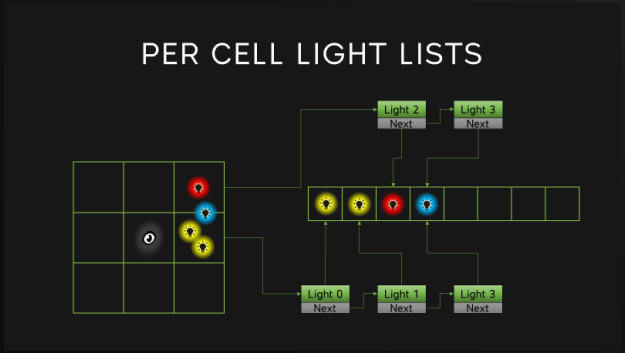
- Unity: camera-oriented acceleration structure [Benyoub 2019] [Tatarchuk 2019]
- BFV: horizontal plane light list [Deligiannis 2019]

Light Importance Sampling:

- *Dynamic Many-Light Sampling for Real-Time Ray Tracing* [Moreau 2019]
- *Stochastic Lightcuts* [Yuksel 2019]



Camera-oriented Acceleration Structure for Lights [Tatarchuk 2019] [Benyoub 2019]



”It Just Works”: Ray-Traced Reflections in ’Battlefield V’ [Deligiannis 2019]

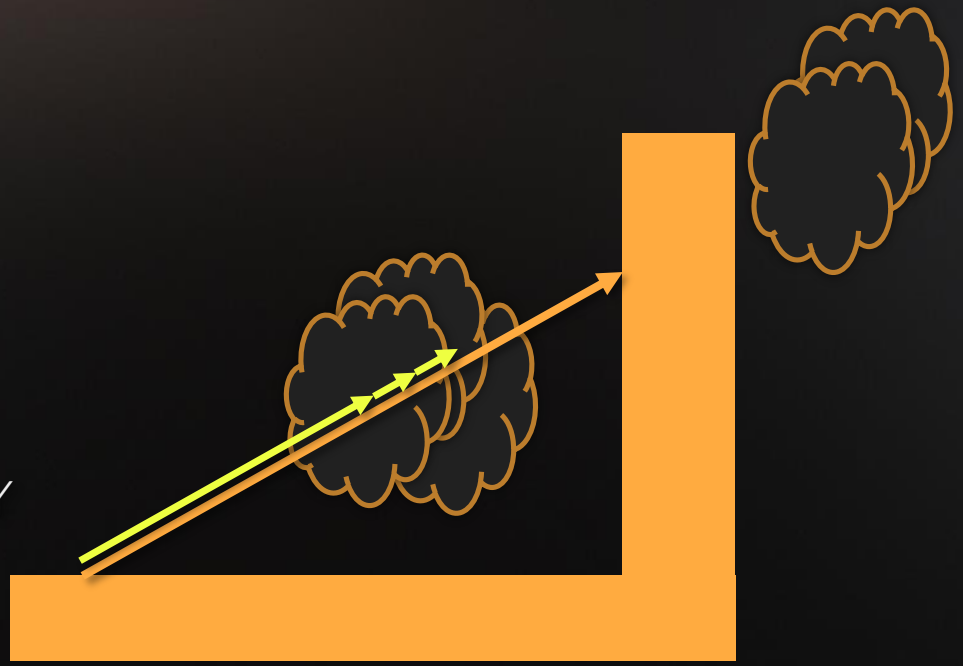
Particles?

[Deligiannis 2019]

[Deligiannis 2019]

Particles

- Particles have to be ray-aligned
  - Not perfect since VFX are often designed assuming view-aligned particles
- Battlefield V [Deligiannis 2019]
  1. Shoot ray in Opaque TLAS
  2. Shoot again in Particle TLAS
     1. Limit length from opaque hit
  3. Blend particles with opaque hit
- Rotate odd particles 90 deg. around Y



[Deligiannis 2019]

[Deligiannis 2019]

[Deligiannis 2019]

Other Things
(yet still super important!)

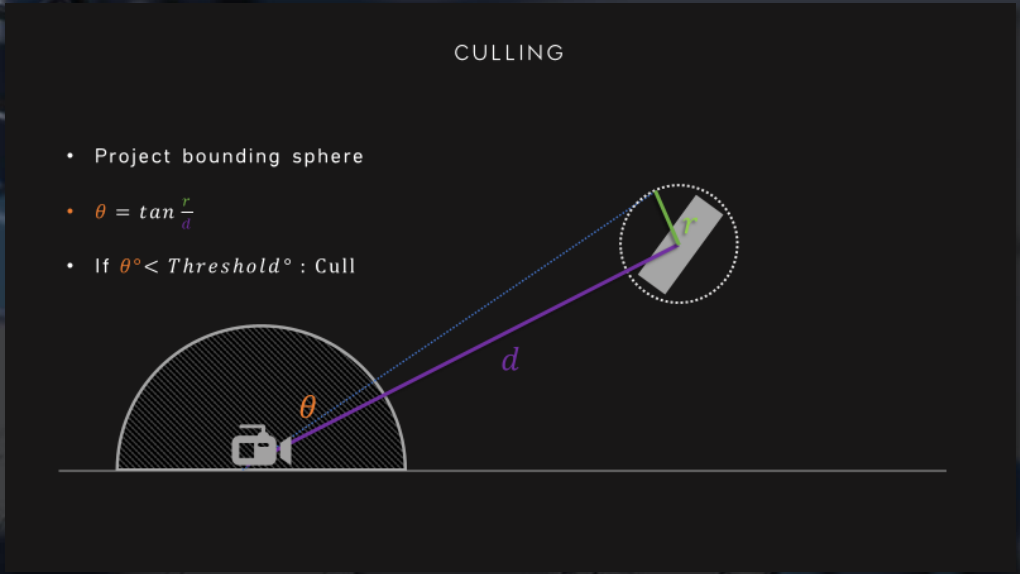
Ray-traced GI

Ray-traced approaches making their way:

- Surfels [Stachowiak 2018]
- Grid [Aalto 2018]
- Probes [Majercik 2019]

Culling

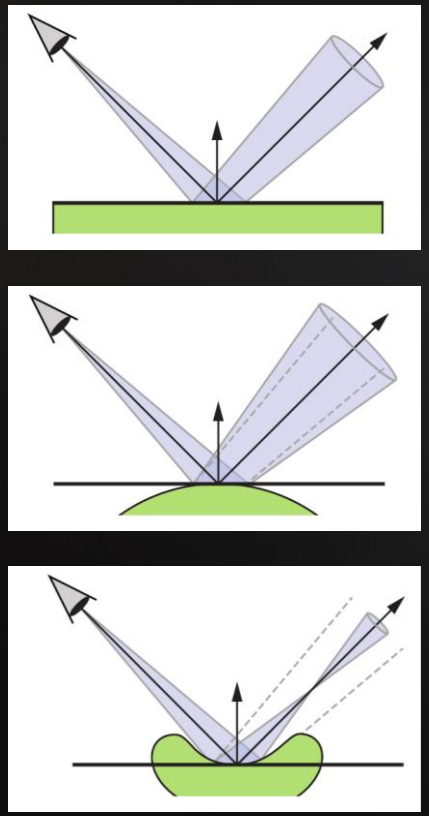
- Can't rely on Frustum Culling, since rays are in world space
- In the case one can't have all objects in the BVH, have to find new culling heuristic
- Projected bounding sphere [Deligiannis 2019]



"It Just Works": Ray-Traced Reflections in 'Battlefield V' [Deligiannis 2019]

Texture LOD

- **No automatic texture LOD with ray tracing**
  - Inherent of the rasterization pipeline: pixel quad derivatives
- In Ray Tracing Gems [Akenine-Möller 2019]
  - *Ray Differentials* [Igehy 1999] vs *Ray Cones* [Amanatides 1984]
  - Heuristic based on triangle properties, a curvature estimate, distance, and incident angle
  - Similar quality to ray differentials with single trilinear lookup. Single value stored in the payload
- Barely scratched the surface – still work to do!
  - Still some needed improvements. Come get inspired at the talk! ☺
  - *Texture Level of Detail Strategies for Real-Time Ray Tracing.* In Ray Tracing Gems 1.1 Session Room: 501AB Wed @ 2PM



Ray Cones

Performance Good Practices

- Minimize recursion: favor fire-and-forget / tail-recursive techniques

- Consider manual scheduling (sorting/binning) [Aalto 2018] [Deligiannis 2019]

- Minimize/pack payload and attributes
- Optimize your *any-hit* shaders (or don't use them at all)
- BLAS build & update on an async queue: define update vs rebuild metric for your case, over multiple frames, overlapped with raster
- TLAS: Build instead of Update & don't include the skybox (do in Miss Shader)

- More Tips & Tricks: Ray Tracing Best Practices [Dunn 2019]

Challenges

Game Constraints

Complex & Large Scale Environments in EA BioWare's Anthem

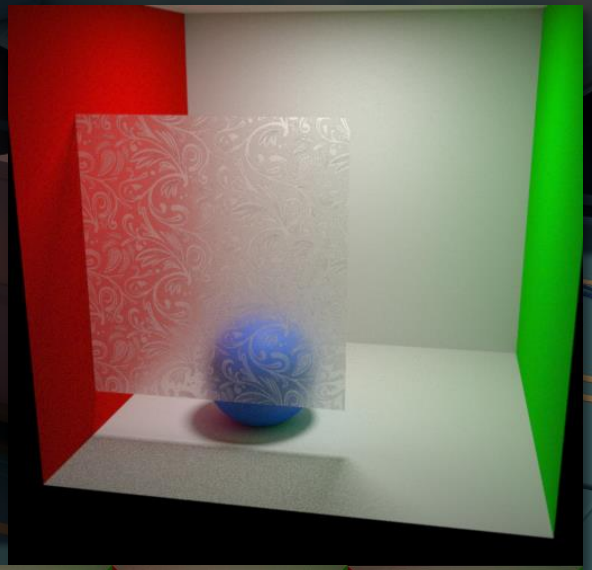
Need robust techniques

for video games:

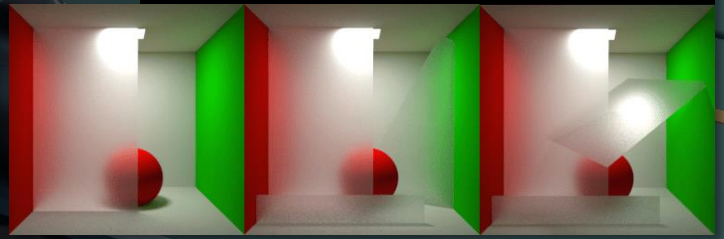
- Many animated characters
- Dynamic environments
- Moving foliage & vegetation
- Massive open worlds
- User generated content
- User created experiences

Transparency

- **Transparency** is far from being solved!
  - Glass
    - Clear/rough + filtered + shadowed
  - Particles & Volumetric Effects
    - Can use miss shader to update volumes / clipmaps
    - Ray marching in hit shaders?
    - Non-trivial blending & filtering



- PICA PICA: **texture-space OIT** with refractions and scattering
  - Not perfect, but one step closer



Transparency in the Maxwell Renderer

- Denoising techniques don't work so well with transparency (and 1-2 spp)

Partial Coverage

- **Foliage**
  - Can still do alpha test (i.e.: any-hit)
  - Animated becomes a real problem

PBRT (Matt Pharr, Wenzel Jakob, Greg Humphreys)

- **Defocus effects** such as motion blur and depth of field are still intractable

- Need partial coverage denoising @ 1-2 spp

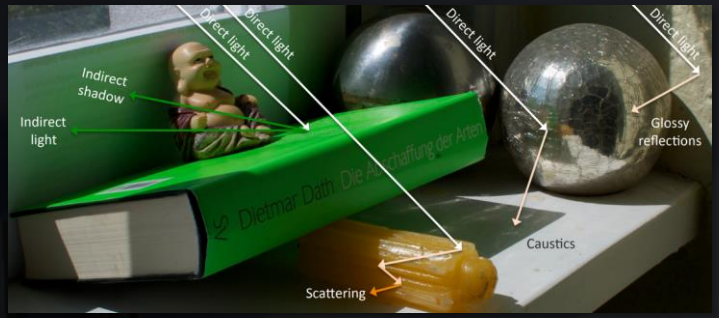
Disney / Pixar

Future Ray Tracing Research

- Literature has to **adjust to real-time game ray tracing constraints**
  - Games → budgeted amount rays/frame, rays/pixel, fixed frame times & memory budgets
  - Games → Light transport caches: surfels, voxels, lightmaps
- Moving towards decoupled shading & variable rate ray tracing
  - Texture-space Techniques
    - Caching of materials and partial solutions
  - Split the BRDF: view vs view-independent terms
- Perf R&D: Efficient sampling / integration strategies & reconstruction filtering

Global Illumination

- **Open problem even in offline rendering**
  - Variance still too high
  - Can reduce frequency
    - Prefiltering, path-space filtering
    - Denoising & reconstruction
  - Pinhole GI & lighting is not solved



[Ritchell 2011]

- **Incoherent** shading → **intractable** performance
  - Have to resort to caching to amortize shading
  - PICA PICA: caching of GI via surfels
    - Issues: only spawn surfels from what you see

- Need to solve GI for **user-generated content** →

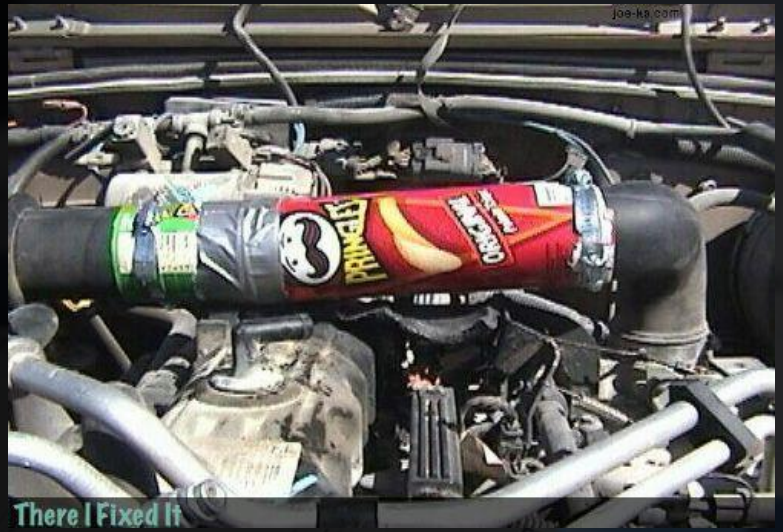
Sparse BVH Tracing

- **We assume ray-triangle intersection as the end-all-be-all, but what if we stopped the ray higher up the tree?**
  - Treat AABBs like voxels
    - Akin to beam tracing [Heckbert 1984], or "ray bundles"
  - Explore algorithms that would benefit from broad tracing
    - *Global Illumination Using Ray-Bundle Tracing* [Tokuyoshi 2012]
    - *Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields* [Majercik 2019]
    - Cone Tracing [Crassin 2011]
    - Sound Propagation
  - Acceleration is currently ray-tri & ray-AABB
    - Feels like it's all here.
    - Expose trace "LOD" controls to developers at API level

Evolving Engines for Hybrid RT

- Bindless
- Visibility-Buffer
- Texture-space Ray Tracing
- Raster still shows some advantages for primary opaque visibility over RT
- …

- Scene Management for RT
- Shader Compilation [Deligiannis 2019]



Summary

- Real-time ray tracing brings the game to another level
- Still have a lots of work to do to bridge offline and real-time
- DXR provides a playground where research and gamedevs can collaborate even more, and solve challenges together!

Thank You!

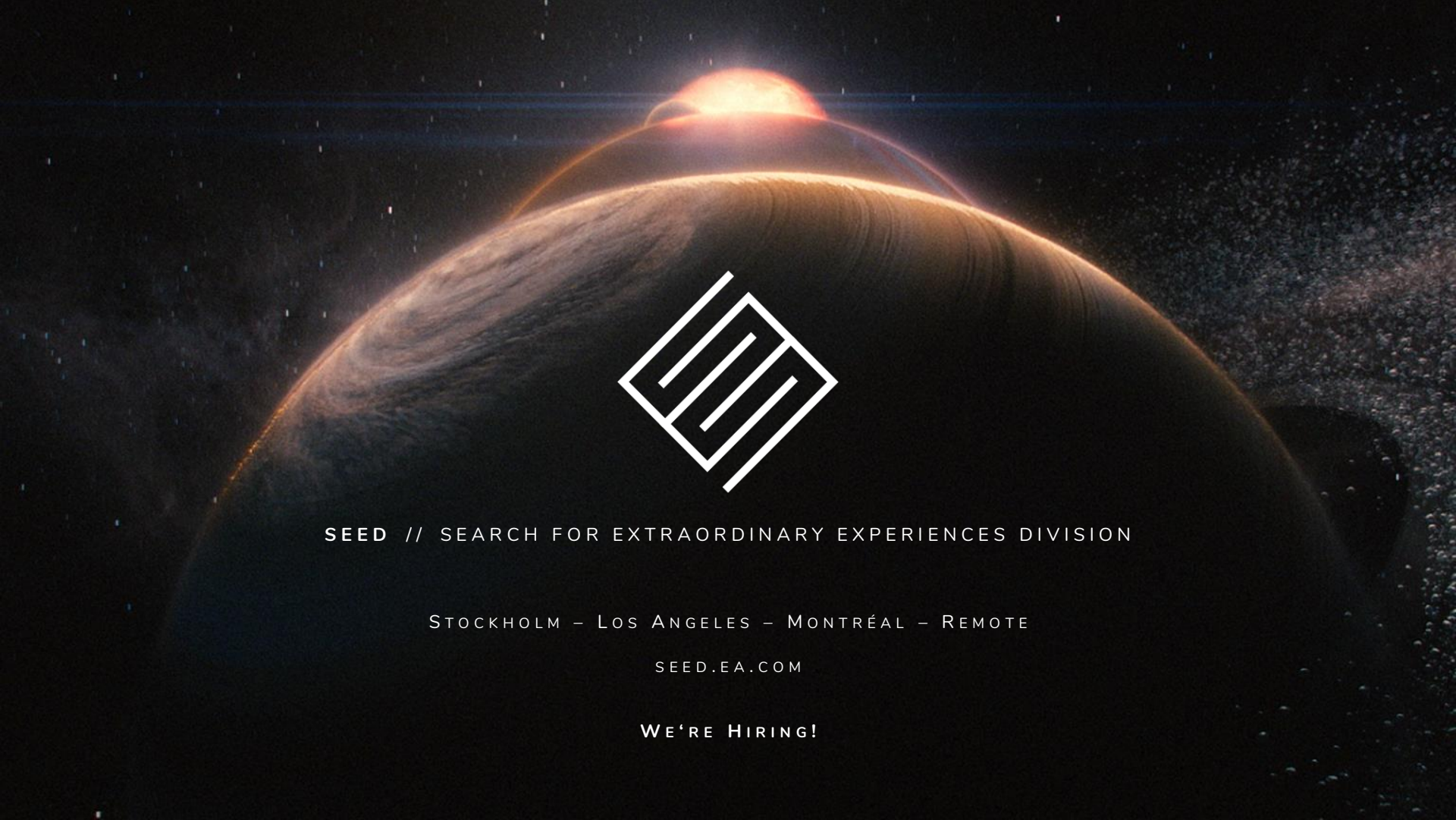
- SEED

- Alex Keller (NVIDIA)
- Natasha Tatarchuk (Unity Technologies)
- Marcus Wassmer (Epic Games)
- Jan Schmid and Johannes Deligiannis (DICE)

- Jon Greenberg (for BVH & beam tracing discussions)

References

- [Aalto 2018] Aalto, Tatu. “Experiments with DirectX Raytracing in Remedy’s Northlight Engine”, online.
- [Amanatides 1984] Amanatides, John. “Ray Tracing with Cones”, online.
- [Andersson & Barré-Brisebois 2018] Andersson, Johan and Barré-Brisebois, Colin. “Shiny Pixels and Beyond: Real-Time Ray Tracing at SEED”, online.
- [Benyoub 2019] Benyoub, Anis. “Leveraging Ray Tracing Hardware Acceleration In Unity”, online.
- [Epic, NVIDIA and ILMxLAB 2019] “Epic Games demonstrates real-time ray tracing in Unreal Engine 4 with ILMxLAB and NVIDIA” online.
- [Heckbert 1984] Heckbert, Paul and Hanrahan, Pat. *Beam Tracing Polygonal Objects*”, online.
- [Igehy 1999] Igehy, Homan. “Tracing Ray Differentials”, online.
- [Majercik 2019] Majercik, Alexander. Guertin, Jean-Philippe. Nowrouzezahrai, Derek. McGuire, Morgan. “Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields”, online.
- [Macedo 2016] de Macedo, Daniel Valente. Rodrigues, Maria Andréia F. "Real-time dynamic reflections for realistic rendering of 3D scenes", online.
- [Ritchel 2011] Ritschel, Tobias et. al. “The State of the Art in Interactive Global Illumination”, online.
- [Schied 2017] Schied, Christoph et. al. “Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path-Traced Global Illumination”, online.
- [Stachowiak 2018] Stachowiak, Tomasz. “Stochastic All The Things: Ray Tracing in Hybrid Real-Time Rendering”, online.
- [Tokuyoshi 2012] Tokuyoshi, Yusuke and Sekine, Takashi. “Global Illumination Using Ray-Bundle Tracing”, online.
- [Tatarchuk 2019] Tatarchuk , Natalya. “Towards Filmic Quality at 30 FPS: Real-Time Ray Tracing for Practical Game Engine Pipelines”, online.
- [Yuksel 2019] Yuksel, Cem. “Stochastic Lightcuts”, online.


SEED // SEARCH FOR EXTRAORDINARY EXPERIENCES DIVISION
Stockholm – Los Angeles – Montréal – Remote
seed.ea.com
We're Hiring!