

I3D 2019 | Montréal, QC | 22 May 2019

DDGI: DYNAMIC DIFFUSE GLOBAL ILLUMINATION WITH RAY-TRACED IRRADIANCE FIELDS

Zander Majercikⁿ, Jean-Philippe Guertin^u, Derek Nowrouzezahrai^m, and Morgan McGuire^{n,m}
Journal of Computer Graphics Techniques (JCGT), vol. 8, no. 2, 1-30, 2019



Thank you. Two of my coauthors are here today, Derek...and the first author, Zander.



Let us begin with breakdown of the lighting factors in this scene



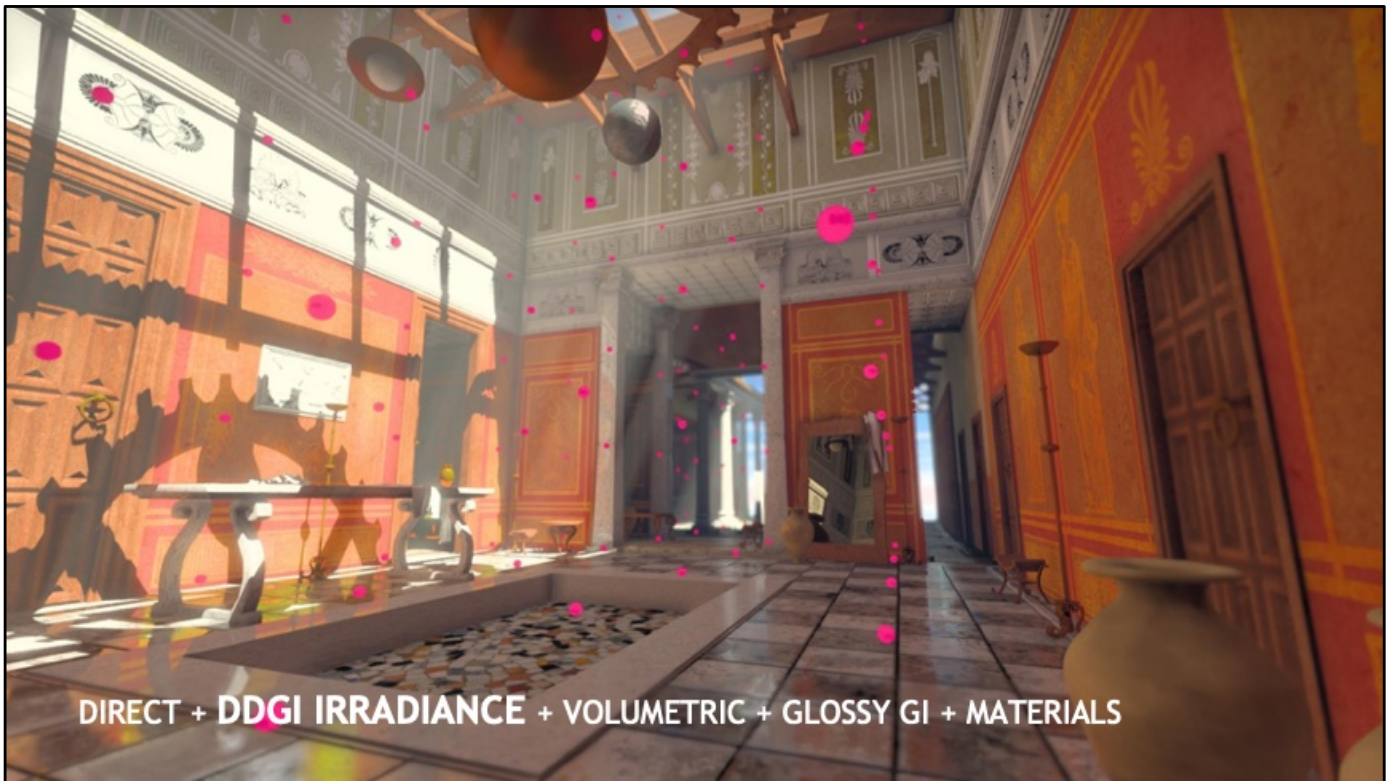
Here's the scene with direct illumination and no materials.

What I want you to notice is that most of the scene is in shadow with respect to the sun and receives zero direct illumination.

Instead, the dominant lighting for this scene



Is multiple-bounce indirect global illumination from the sun and the sky, reflected from matte surfaces towards the camera. Here is the lighting computed by our technique.



Our method works by ray-tracing radiance irradiance probes in real time.

The pink spheres are a visualization of the locations of the probes.

Unlike previous light probes, these have **visibility** information that makes them robust

And are computed continuously and **incrementally**

Here is a comparison of the new method to...



Flat ambient light instead of global illumination



To classic irradiance probes...



And our result. This is a very similar data structure to the irradiance probes used extensively today.

The improvement in quality is because of that visibility information, which ensures that lighting is interpolated between probes in ways that respect the scene geometry.

And this operates in a few milliseconds per frame ...



...with dynamic cameras, geometry, and lights.

DYNAMIC DIFFUSE GLOBAL ILLUMINATION

“Diffuse Global Illumination” = **Irradiance** due to indirect light

$$E_e(X) = \int_{\Omega} L(X, \omega) \max(0, n \cdot \omega) d\omega$$



What is now popularly called “diffuse global illumination” is technically irradiance *field*.

The **irradiance** at a point X with surface normal n is the integral of incident radiance over all directions ω weighted by the projected area.

This is a term often modulated by one minus Fresnel reflection and applied as the matte/Lambertian lobe of the material’s scattering function. It describes most of the light in typical scenes.

Note that we are NOT talking about Radiosity here. Radiosity is the outgoing light in a scene that is entirely Lambertian. Irradiance is incoming light from a scene with all kinds of materials...it just describes the term that you need for the diffuse part of the last bounce, where you compute the glossy part using some kind of environment map or ray tracing and have a full lighting solution.

It was first described as a field to be sampled for rendering by Greger, Don Greenberg, and Pete Shirley in 1998, and there’s been a lot of work since on good data structures for computing or storing it...

SELECTED RELATED WORK

Light maps	[Quake97, Mitchell06]
Virtual point lights	[Keller97, Kaplanyan10, Ding14, Xu16, Luksch19]
Reflective shadow maps	[Dachsbacher05, Kaplanyan10, Ding14, Malmros17, Xu16]
Light propagation volumes	[Kaplanyan09, Kaplanyan10, Boeckmann19]
Sparse voxel cone tracing	[Crassin11, McLaren16]
Denoised ray tracing	[Mara17, Schied17, Metro19, Archard19]
Irradiance probes/voxels	[Greger98, Tatarchuk05, Ramamoorthi11, Gilabert12]

11 

Because diffuse GI is so important, there have been a lot of techniques for simulating it and every real-time system uses one.

Let me focus on one problem that many of these have in common, which is handling visibility when lighting is sampled into an intermediate world, camera, or screen-space data structure...which all of these do.

This is not specific to irradiance probes, but I will use that example because it is what our algorithm builds on...

IRRADIANCE PROBES

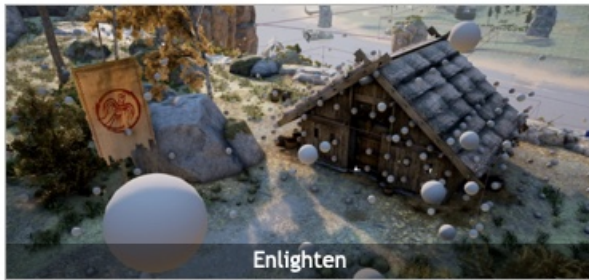


Image Credits: Geomerics and Ninja Theory, <https://unity3d.com/learn/tutorials/topics/graphics/probe-lighting>,
<https://docs.unrealengine.com/en-us/Engine/Rendering/LightingAndShadows/IndirectLightingCache>, <https://unity3d.com/learn/tutorials/topics/graphics/probe-lighting>

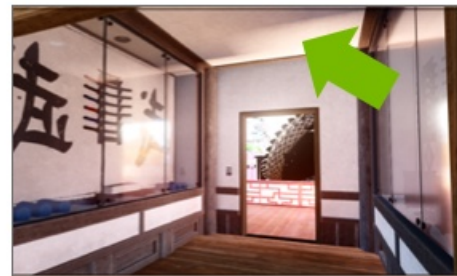
Irradiance probes have been with us since 1998 and are supported by most engines today. Fill the world with small probes that measure and store diffuse GI. Usually prebaked, although Enlighten can update them at runtime.

Probes might be cube maps, spherical harmonics, octahedral maps, etc.

The quality is excellent in the best case.

However, they “leak” light and shadow when sampled away from the probe centers...

LIGHT & SHADOW LEAKS



[Hooker 2016]

These slides are from a Treyarch presentation at SIGGRAPH. Everybody has the same problem.

If the lighting changes radically near a probe because of a wall, they can bleed light inside of a room from outside sun.

Or bleed darkness outside. For the image on the top, the bright area on the ceiling is sunlight on the roof bleeding in.

If the probe lands inside of a wall, then all that it sees is darkness and it bleeds shadow everywhere. The dark shadow on the door on the lower right image is because the probe behind the door is bleeding shadow out.

The state of the art is to have artists manually move each probe to a good location and manually place **blocker** geometry to divide inside and outside. That is another huge workflow cost. This is what Hooker's talk was actually about: the tool that they created to help artists manually inject visibility information.

Moving the probes also doesn't help if you want to get to runtime dynamic probe lighting. If you update lighting at runtime, then dynamic geometry or a character might be covering up a probe no matter where you place it.

I'm explaining the leaking issue for probes because that's where we're going to fix it today. But leaking is a problem for all GI solutions...

LIGHT & SHADOW LEAKS



[Iwanicky 2013]



[Rakhtenko 2018]

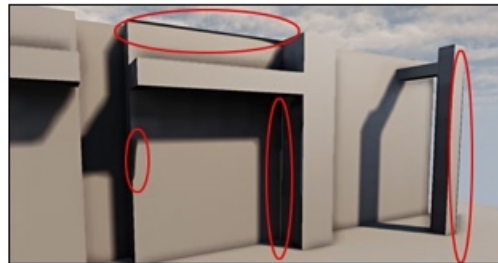


Image Sources: <https://forums.unrealengine.com/development-discussion/content-creation/18712-need-help-how-to-fix-the-light-under-walls>, <https://answers.unrealengine.com/questions/336484/light-leaking-problem-solid-geometry.html>, <https://www.worldofleveldesign.com/categories/udk/udk-lightmaps-03-how-to-fix-light-shadow-lightmap-bleeds-and-seams.php>

For example, here are shadow maps leaking due to interpolation problems at corners and parameterization seams. The same thing happens for voxels, light propagation volumes, reflective shadow maps, virtual point lights, and even per-pixel path tracing when denoising on the result.

Our contribution is to avoid this leaking...

CONTRIBUTIONS

Avoid leaks with visibility data + heuristics

Dynamic irradiance in 1-2M rays/frame

Noiseless, incremental, infinite-bounce

Evaluation of precision and resolution parameters

Forward, deferred, transparent, volumetric, etc.

Eliminate surface parameterization/manual probe placement requirement

Open-source C++/OpenGL/OptiX reference implementation

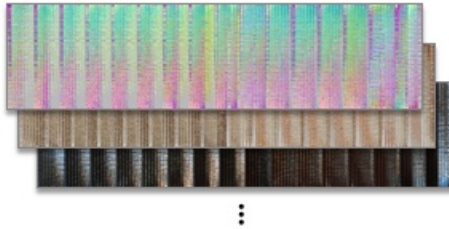
15 

We also support fully dynamic scenes with hardware-accelerated ray tracing.
The reference code is available online at JCGT.



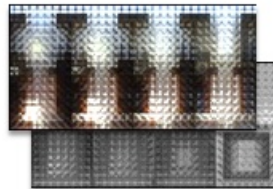
DDGI IRRADIANCE PROBES

Independent of framerate and screen resolution



1. Ray Trace & Shade

Trace and shade packed rays from active probes.
Uses previous iteration for shading: infinite bounce GI.



2. Filtered Blend

Blend irradiance and depth into probes.
Duplicate probe border texels for fast bilinear.



3. Sample

Volumetric sample based on 3D position,
visibility, and normal.

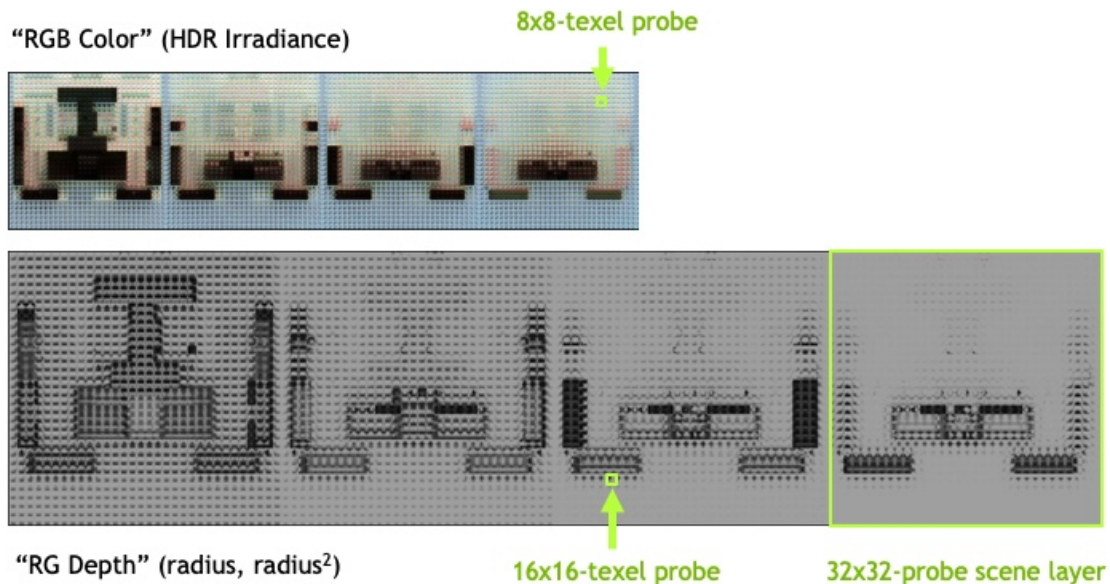
For each probe, we trace a set of rays in a low discrepancy spherical Fibonacci pattern. The pattern rotates every frame to avoid aliasing.

We pack the ray hits together into a ray traced G-buffer and then run the usual deferred shading pass on it.

Note that we're using the full scene geometry with no LOD for the trace and the full deferred shader.

Here is a close view of the layout of the texture data structures in step 2

DATA STRUCTURE



This is an octahedral projection of each probe's sphere into a square
We then pack thousands of those tiny squares into two texture maps.

Irradiance is low frequency with respect to angle, so is stored at lower resolution than the depth data.

Every texel here represents the convolution with a cosine hemisphere for irradiance and a power-cosine for depth.

We store radial distance and distance *squared*...

RADIAL GAUSSIAN DEPTH

Mean:

$$\mu = \sum r / n$$

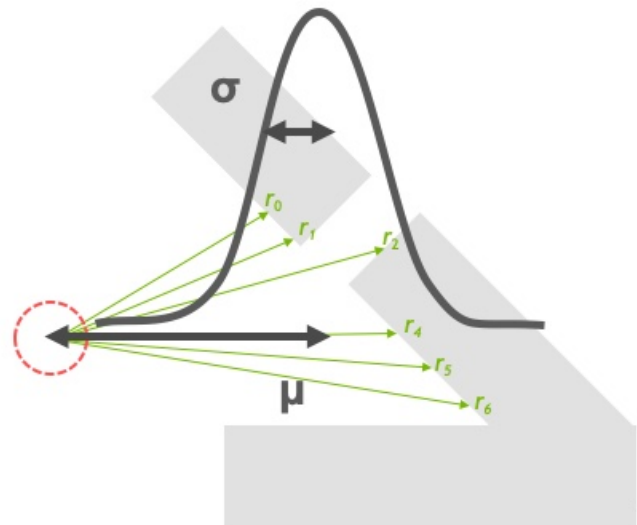
Standard Deviation:

$$\begin{aligned} \sigma &= \sqrt{\sum (r - \mu)^2 / n} \\ &= \sqrt{(\sum r^2) / n - \mu^2} \end{aligned}$$

Each probe texel stores $(\sum r, \sum r^2)$

Irradiance blurs over a cosine-weighted hemisphere in a gather pass.

Blur depth over a *power-cosine* to capture variation but retain some sharpness.



Because that allows us to reconstruct a gaussian model of depth *within a texel* and filter it via exponentially-weighted moving mean and variance.

Here I'm showing a top view of one probe in pink and some gray geometry. The green rays all contribute to a single texel. The resulting mu and sigma describe the distribution of depths within that texel.

This compact model allows us to perform a Chebychev statistical visibility test when interpolating shading.

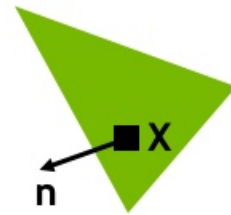
Similar ideas have been used for variance shadow maps, moment shadow maps, moment order-independent transparency.

Some problems with leaking on those because shadow maps simultaneously cover huge depth ranges (tens to thousands of meters) and have tiny texels in world space (few cm)

Works better for irradiance because the grid is much coarser than the geometry. Probes are on a 1m grid and only need visibility accurate within 1m.

SAMPLING IRRADIANCE

```
// float3 n = shading normal, X = shading point
```



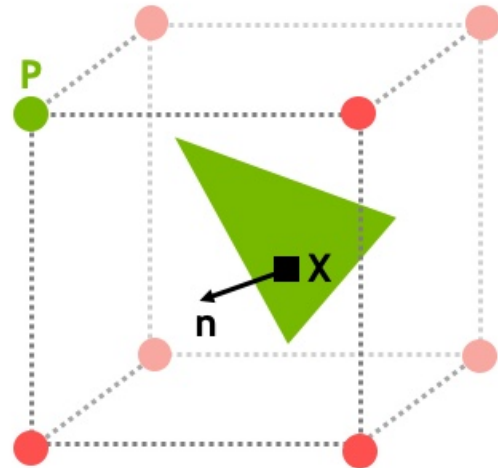
20 

You have a point on some triangle that you're trying to shade. Forward, deferred, ray traced, volumetric marching, etc.

Let's call it X and the surface normal n .

SAMPLING IRRADIANCE

// float3 n = shading normal, X = shading point, P = probe location



21 NVIDIA

There are eight probes forming a cube around the point. It may be a distorted square if the optimizer or artist moved the probes, but we compute this in regular grid space.

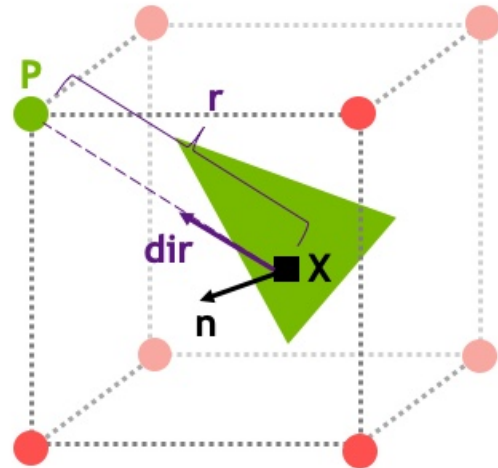
Iterate over the probes. For each probe P:

SAMPLING IRRADIANCE

// float3 n = shading normal, X = shading point, P = probe location

```
float4 irradiance = float4(0);
for (each of 8 probes around X) {
    float3 dir = P - X;
    float r = length(dir);
    dir *= 1.0 / r;

    1 // smooth backface
    float weight = (dot(dir, n) + 1) * 0.5;
```



22 

Compute a smooth **backface** weight. This is a really inexpensive visibility approximation based on wrap shading. It reduces the impact of lighting sampled *behind* the surface on the front face., since the surface itself probably changes the lighting.

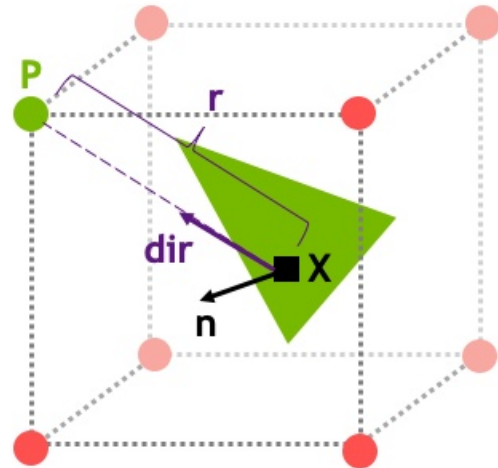
SAMPLING IRRADIANCE

// float3 n = shading normal, X = shading point, P = probe location

```
float4 irradiance = float4(0);
for (each of 8 probes around X) {
    float3 dir = P - X;
    float r = length(dir);
    dir *= 1.0 / r;

    1 // smooth backface
    float weight = (dot(dir, n) + 1) * 0.5;

    2 // adjacency
    weight *= trilinear(P, X);
}
```



23 

Then compute an adjacency weight by trilinear interpolation across the three axes of the cue of probes. This allows us to smoothly transition between probes in 3D space.

SAMPLING IRRADIANCE

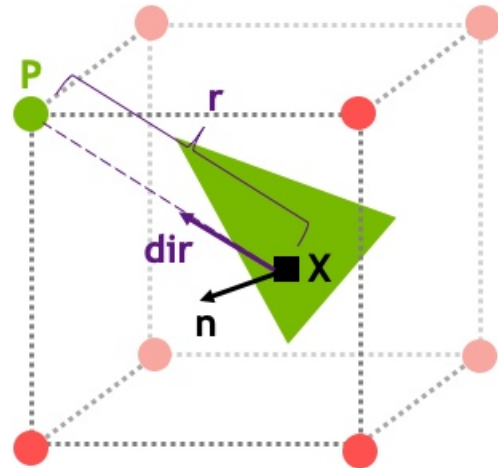
// float3 n = shading normal, X = shading point, P = probe location

```
float4 irradiance = float4(0);
for (each of 8 probes around X) {
    float3 dir = P - X;
    float r = length(dir);
    dir *= 1.0 / r;

    1 // smooth backface
    float weight = (dot(dir, n) + 1) * 0.5;

    2 // adjacency
    weight *= trilinear(P, X);

    // visibility (Chebyshev)
    3 float2 temp = texture(depthTex, probeCoord).rg;
    float mean = temp.r, mean2 = temp.g;
    if (r > mean) {
        float variance = abs(square(mean) - mean2);
        weight *= variance / (variance + square(r - mean));
    }
}
```



24 

Finally, the big one. Read back the gaussian depth model and apply Chebyshev's statistical test. This tells us how much of the gaussian is past a certain point. In our case, we want to know the fraction of the probe is obscured by walls.



RESULTS

PERFORMANCE

	Time (ms)
Ray generation	0.1
Ray cast	0.8
Ray shade	0.4
Probe update	0.7 (0.3 color + 0.4 depth)
Sample irradiance probes for primary ray shading	0.5
Deferred direct shading	0.1
Total	2.6
Add brute-force ray-traced glossy	+2.4

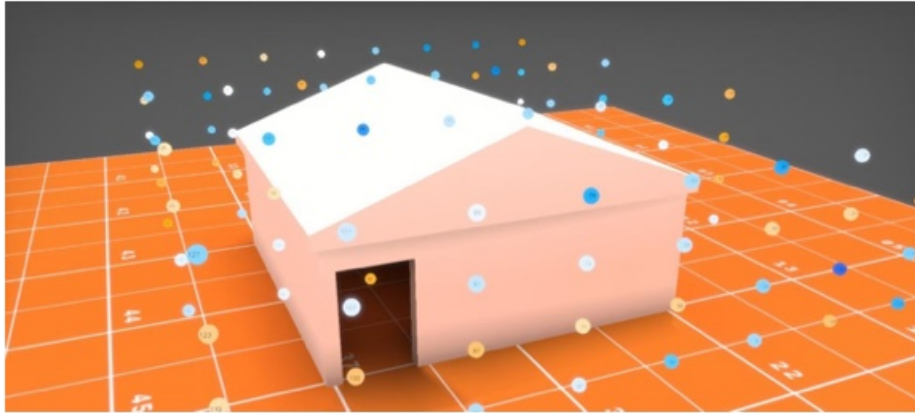
1920x1080, 32x8x32 probes, 64 rays/probe, RTX 2080 Ti,
OptiX + OpenGL, 8x8 GL_RGB10A2 color, 16x16 GL_RG16F depth

26 

The performance is highly tunable because you can change the resolution of the grid, the screen, and the textures as well as control the number of rays per probe per frame.

With the settings from the results that I am about to show, the fully-dynamic GI takes 2.6 milliseconds per frame.

AVOIDS LEAKS



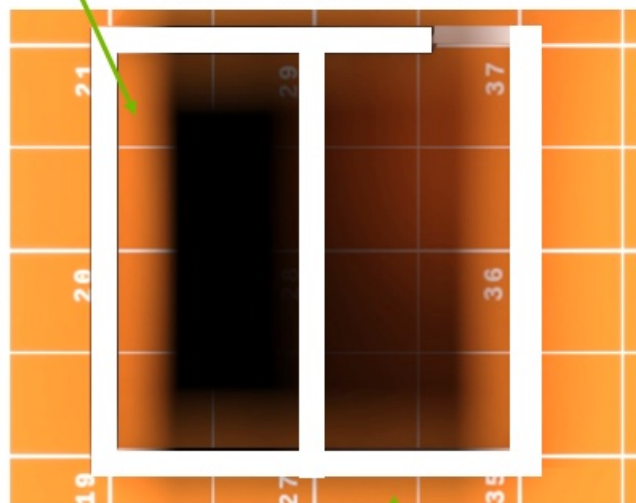
AVOIDS LEAKS



AVOIDS LEAKS

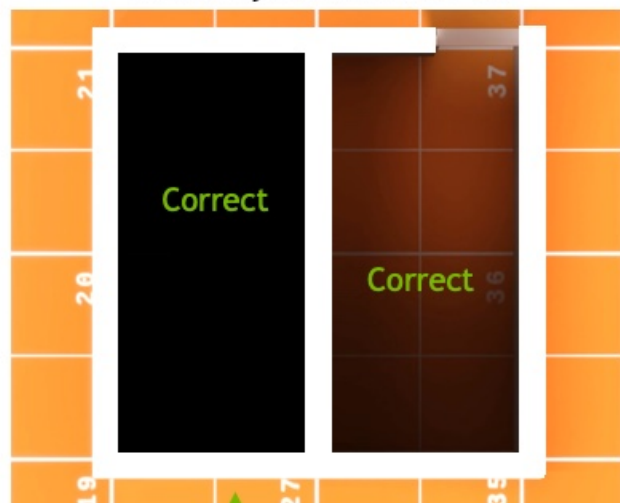
Light leak

Before: Classic Probes



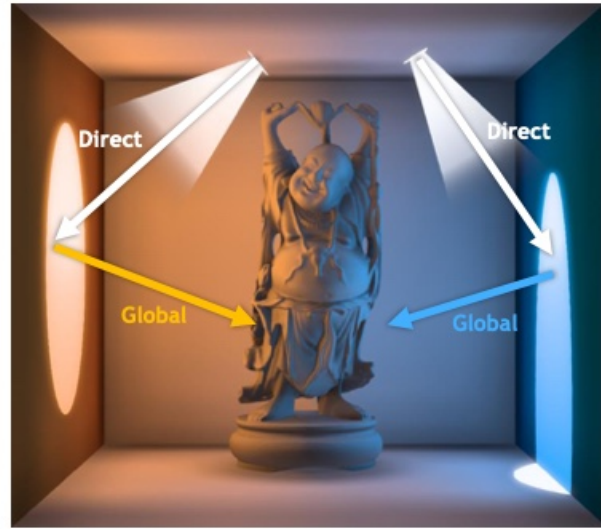
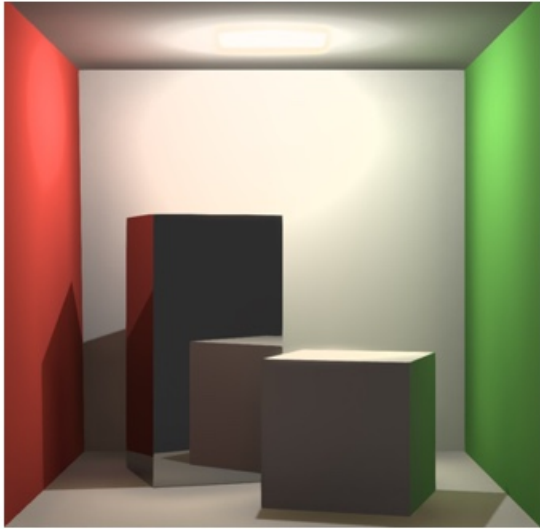
Shadow leak

After: Dynamic Diffuse GI



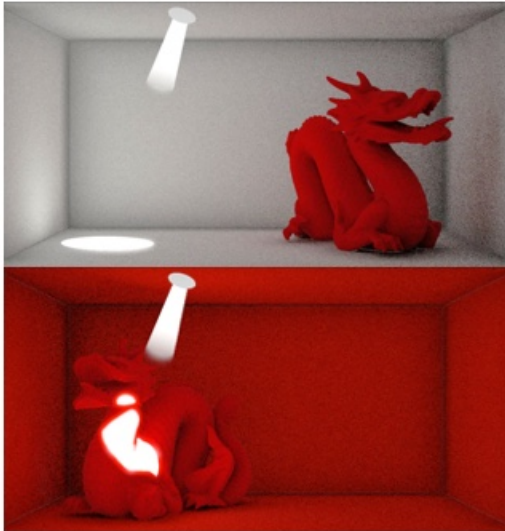
Correct

ACCURATE & NOISE FREE

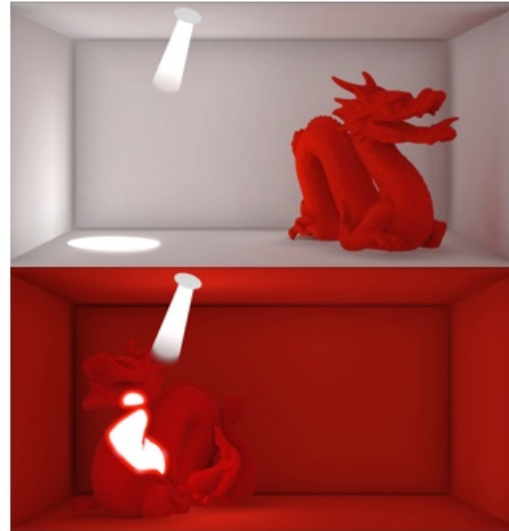


ACCURATE & NOISE FREE

Path Tracing



DDGI Irradiance Probes



PROBE GRID RESOLUTION



(a) 4×4 probes, AO



(b) 4×4 probes, no AO



(c) 8×8 probes



(d) 16×16 probes



(e) 64×64 probes



(f) Path-traced reference

REALISTIC

Direct Illumination Only



+ Dynamic Diffuse GI



DYNAMIC LIGHTING

Afternoon



Evening



LARGE SCENES



DYNAMIC GEOMETRY



DYNAMIC GEOMETRY



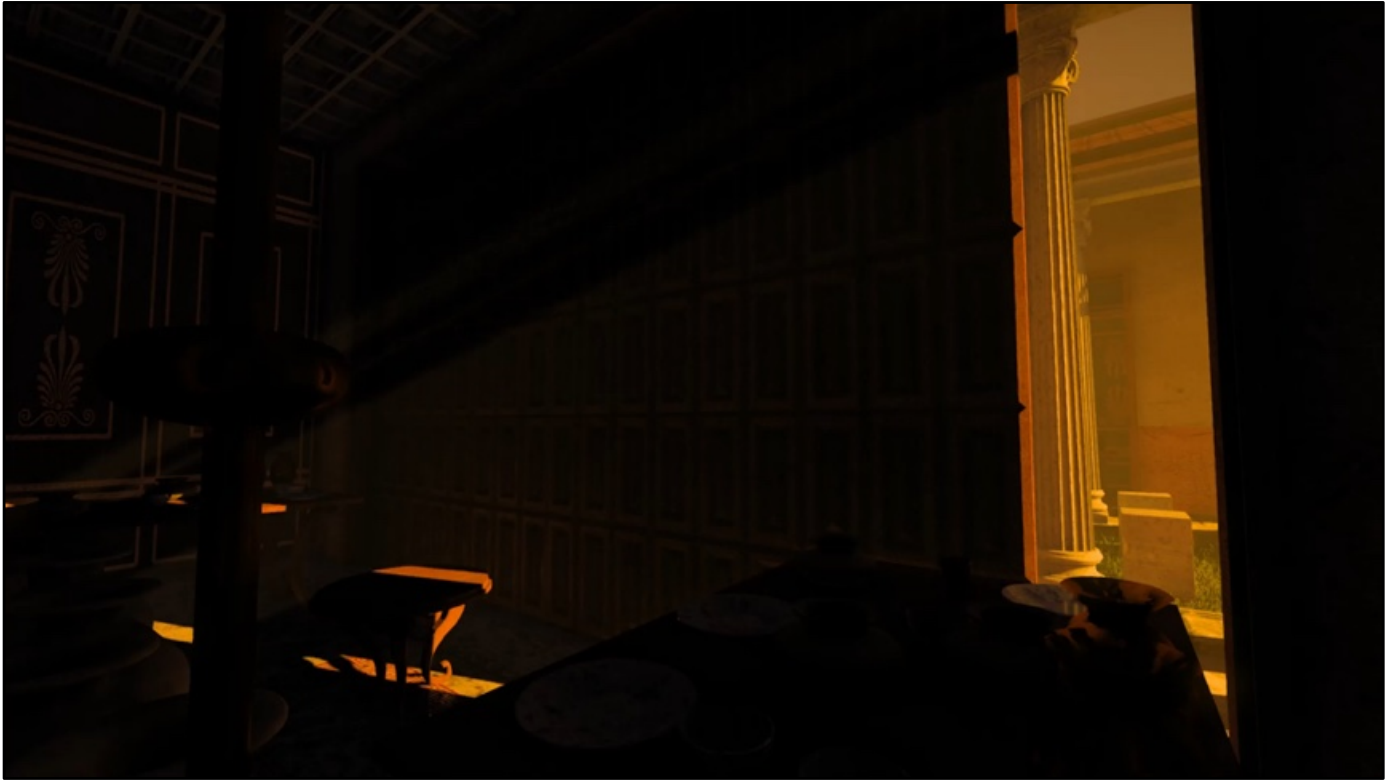












LIMITATIONS & CONTINUING WORK

Low-frequency (combine with short screen-space GI [Mara2016])

Thin corridors & sharp corners

Probes *just* inside walls

Convergence rate (faster → more rays → lower hysteresis)

44 

Here are some limitations of the work as published which we're now working on as follow-up.

In areas where there are two facing walls without a probe in between some fallback approximation is needed because there are no samples. Sharp concave corners are an extreme example of this.

Because of the amortization, if there is a radical change in direct illumination it can take several frames for the indirect to update.

By constant factor performance optimization, we can increase the rays per probe and reduce the hysteresis within the same budget.

I showed you 64 rays per probe in 2.5 ms; our latest implementation is up to 256 rays in 1.5 ms.

SUMMARY

Real-time ray tracing of the irradiance *field*

Amortized no high resolution radiance ever stored

Fast, noise free, parameterization free

In the paper:

- Exhaustive exploration of parameter space
- Open source reference implementation

MORE INFORMATION

Contact mcguire@nvidia.com, @CasualEffects

Blog Post <https://morgan3d.github.io/articles/2019-04-01-ddgi>

Paper <http://jcgf.org/published/0008/02/01/>

Thanks to Dylan Lacewell, Mike Mara, Dan Evangelakos, Sam Donow, and Corey Taylor for their work on the implementation infrastructure; and Alice Peters, Andrew Wilmot, and Marc Olano at JCGT for support on the paper.

Built using the open-source G3D Innovation Engine <https://casual-effects.com/g3d>



BIBLIOGRAPHY

Greger et al., *The Irradiance Volume*, IEEE CG&A 1998

Gilabert and Stefanov, *Deferred Radiance Transfer Volumes: Global Illumination in Far Cry 3*, GDC 2012

Tatarchuk, *Irradiance Volumes for Games*, GDC 2005

Kaplanyan, *Light Propagation Volumes in CryEngine 3*, SIGGRAPH Advances in Real-Time Rendering Course, 2009

Kaplanyan and Dachsbacher, *Cascaded Light Propagation Volumes for Real-Time Indirect Illumination*, I3D 2010

Asirvatham and Hoppe, *Terrain Rendering Using GPU-Based Geometry Clipmaps*, GPU Gems 2, Pharr & Fernando eds., 2005

Donow, *Light Probe Selection Algorithms for Real-Time Rendering of Light Fields*, Williams College Thesis, 2016

Evangelakos, *A Light Field Representation for Real Time Global Illumination*, Williams College Thesis, 2015

Mara et al., *An Efficient Denoising Algorithm for Global Illumination*, HPG 2017

Mara et al., *Deep G-buffers for Stable Global Illumination Approximation*, HPG 2016

Wang et al., *Fast Non-Uniform Radiance Probe Placement and Tracing*, I3D 2019

Keller, *Instant Radiosity*, SIGGRAPH 1997

Ritschel et al., *Imperfect Shadow Maps*, SIGGRAPH Asia 2008

Dachsbacher and Stamminger, *Reflective Shadow Maps*, I3D 2005

Donnelly and Lauritzen, *Variance Shadow Maps*, I3D 2006

Hooker, *Volumetric Global Illumination at Treyarch*, SIGGRAPH Advances in Real-Time Rendering, 2016

Cigolle et al., *Survey of Efficient Representations for Independent Unit Vectors*, JCGT 2014

BIBLIOGRAPHY

- Luksch et al., Incrementally baked Global Illumination, I3D 2019
- Stachowiak, Stochastic Screen-Space Reflections, SIGGRAPH Advances in Real-Time Rendering, 2015
- Valient, Reflections and Volumetrics of Killzone Shadow Fall, SIGGRAPH Advances in Real-Time Rendering, 2014
- McGuire and Mara, Efficient GPU Screen-Space Ray Tracing, JCGT 2014
- Majercik et al., Dynamic Diffuse Global Illumination with Ray Traced Irradiance Fields, JCGT 2019
- Mara, CloudLight: A Distributed Global Illumination System for Real-Time Rendering, Williams College Thesis, 2012
- McGuire et al. Real-Time Global Illumination using Precomputed Light Field Probes, I3D 2017
- Silvennoinen and Lehtinen, Real-time Global Illumination by Precomputed Local Reconstruction from Sparse Radiance Probes, ACM ToG 2017
- Schied et al., Spatiotemporal Variance-Guided Filtering: Real-Time Reconstruction for Path Traced Global Illumination, HPG 2017
- Chajdas et al., Assisted Environment Map Probe Placement, SIGRAD 2011
- Rakhteenko, Baking artifact-free lightmaps on the GPU, 2018 <https://ndotl.wordpress.com/2018/08/29/baking-artifact-free-lightmaps/>
- Crassin et al., Interactive Indirect Illumination Using Voxel Cone Tracing, INRIA 2011
- Crassin et al., CloudLight: A System for Amortizing Indirect Lighting in Real-Time Rendering, JCGT 2015
- Mitchell et al., Shading in Valve's Source Engine, SIGGRAPH Advanced Real-Time Rendering in 3D Graphics and Games, 2006
- O'Donnell, Precomputed Global Illumination in Frostbite, GDC 2018
- Iwanicki, Lighting Technology of "The Last of Us", SIGGRAPH Advances in Real-Time Rendering, 2013
- Dimitrov, Cascaded Shadow Maps, NVIDIA 2007 https://developer.download.nvidia.com/SDK/10.5/opengl/src/cascaded_shadow_maps/doc/cascaded_shadow_maps.pdf

BIBLIOGRAPHY

The “personal communication” references below are primary sources of developers who worked on these games.

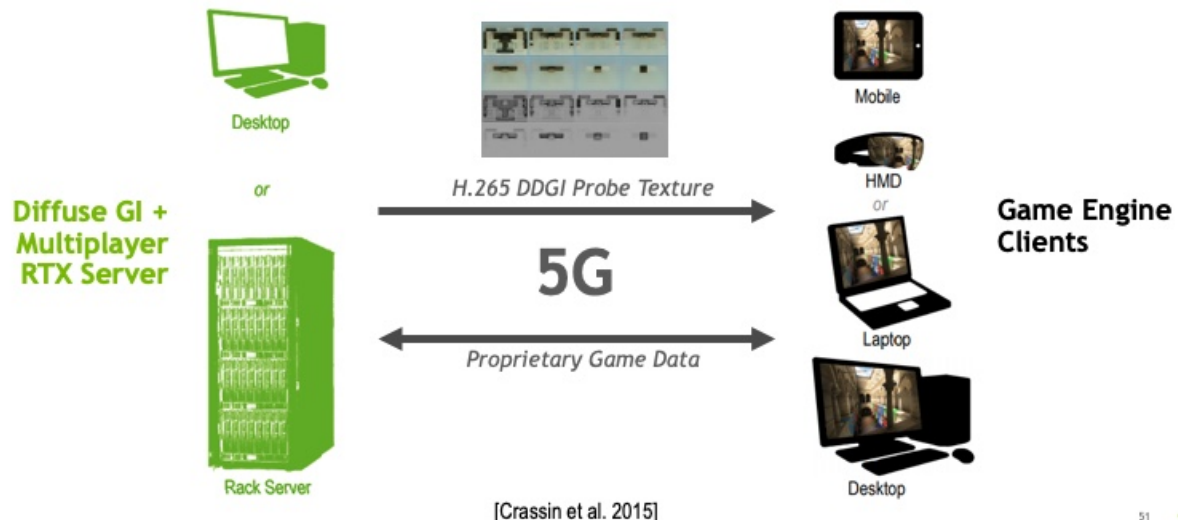
- [Archard19] Archard, Zhdan, Shyshkovtsov, and Karmalsky, Exploring Raytraced Future in Metro Exodus, GDC Talk, 2019
- [Asirvatham05] Asirvatham and Hoppe, Terrain Rendering Using GPU-Based Geometry Clipmaps, GPU Gems 2, Pharr and Fernando eds., 2005
- [Barrett19] Barrett, Personal communication: Just Cause 3 and 4 used RSM, May 3, 2019
- [Metro19] Battaglia, Interview with Ben Archard on Metro: Exodus, Eurogamer, Feb 17, 2019
- [Blinn76] Blinn and Newell, Texture and reflection in computer generated images, CACM, Vol. 19, No. 10, pages 542-547, October 1976
- [Crassin11] Crassin, Neyret, Sainz, Green, and Eisemann, Interactive Indirect Illumination Using Voxel Cone Tracing, Pacific Graphics, 2011
- [Chajdas11] Chajdas et al., Assisted Environment Map Probe Placement, SIGRAD 2011
- [Dachsbacher05] Dachsbacher and Stamminger, Reflective Shadow Maps, I3D, 2005
- [Deligiannis19] Deligiannis, Schmid, and Uludag, It Just Works: Ray-Traced Reflections in Battlefield V, GTC Talk, 2019
- [Ding] Ding, In-Game and Cinematic Lighting of The Last of Us, GDC Presentation, 2014
- [Gilbert12] Gilbert and Stefanov, Deferred Radiance Transfer Volumes: Global Illumination in Far Cry 3, Talk at GDC, 2012
- [Kaplanyan10] Kaplanyan, Real-time Diffuse Global Illumination in CryENGINE 3, SIGGRAPH Presentation, 2010
- [Keller97] Keller, Instant radiosity, SIGGRAPH, 1997
- [Malmros17] Malmros, Gears of War 4: custom high-end graphics features and performance techniques, SIGGRAPH Talk, 2017
- [McLaren16] McLaren, Graphics Deep Dive: Cascaded voxel cone tracing in The Tomorrow Children, Gamasutra, November 28, 2016

BIBLIOGRAPHY

- [Mitchell06] Mitchell, McTaggart, and Green, Shading in Valve's Source Engine, SIGGRAPH Advances in Real-Time Rendering Course, 2006
- [Quake] ID Software, Quake II, 1998
- [Qian19] Qian and Yang, RTX in Justice, GDC Talk, 2019
- [Rakhteenko18] Rakhteenko, Baking artifact-free lightmaps on the GPU, Blog 2018
- [Ramamoorthi11] Ramamoorthi and Hanrahan, An Efficient Representation for Irradiance Environment Maps, SIGGRAPH, 2001
- [Sassone] Sassone, Personal communication: Dirt Showdown used VPLs, May 3, 2019
- [Schied17] Schied, Kaplanyan, Wyman, Patney, Chaitanya, Burgess, Liu, Dachsbacher, Lefohn, and Salvi, Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination, HPG '17, July 27, 2017, 12 pages
- [Sousa11] Sousa, Kasyan, and Schulz, Secrets of CryENGINE 3 graphics technology. SIGGRAPH Advances in Real-Time Rendering, 2011
- [Stachowiak15] Stachowiak, Stochastic Screen-Space Reflections, SIGGRAPH Advances in Real-Time Rendering, 2015
- [Tatarchuk05] Tatarchuk, Irradiance Volumes for Games, GDC Presentation, 2005
- [Valient14] Valient, Taking Killzone Shadow Fall image quality into the next generation, Presentation at GDC, 2014
- [Wang19] Wang et al., Fast Non-Uniform Radiance Probe Placement and Tracing, I3D 2019
- [White19] White, Personal communication: Gears of War 4 used RSM + VPLs, May 3, 2019
- [Wronski14] Wronski, Assassin's Creed 4: Black Flag, the road to next-gen graphics, GDC Presentation, 2014
- [Xu16] Xu, Temporal Antialiasing In Uncharted 4, SIGGRAPH Advances in Real-Time Rendering Course, 2016

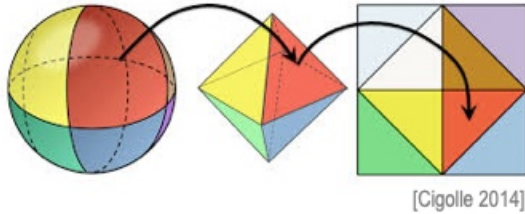
The Ultimate Optimization

STREAMING GI



The biggest optimization is to offload the probe rendering to the cloud.
 Local desktop + mobile VR HMD = high quality dynamic GI on low-end hardware.
 For a multiplayer environment, server side computation amortizes over multiple users.

PROBE PACKING



Sphere → Octahedron → Square

Pack XZ squares, layer in Y

Including border texels for fast bilinear:

Depth: 16x16 RG16F = 1024 bytes

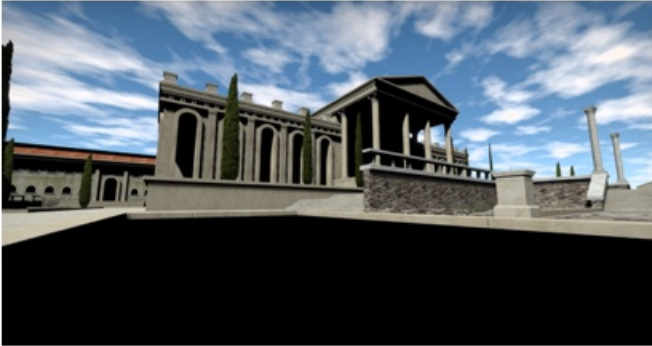
Irradiance: 6x6 R11G11B10F = 144 bytes

= **1168** bytes/probe



REALISTIC

Direct Illumination Only



+ DDGI Irradiance Probes + Glossy Rays



PROBE PLACEMENT

Grid

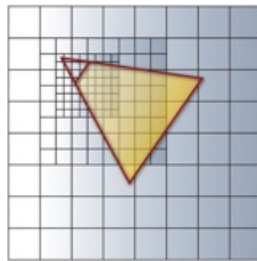
Optionally optimize around static geo [Chajdas 2011, Donow 2016, Wang et al. 2019, Unity]

Artists may override placement

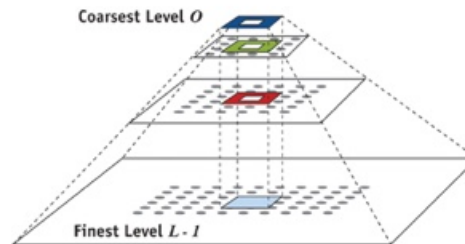
Cascades

$32 \times 4 \times 32 = 4 \text{ k}$ probes around the camera that update frequently.

Coarse cascades in space and time to scale out to big scenes.



[Kaplanyan and Dachsbacher 2010]



[Asirvatham and Hoppe 2005]



Top View

You can also run an optimizer to move them around static geometry for even better resolution, and I've cited some research explaining how to do that.

And artists *can* manually move any probe within its grid cell. Unlike classic irradiance probes, they don't HAVE to move, them, though.

Just like you would with shadow maps, terrain, or voxels, you should put the probes in cascades to cover large regions.

Coarse cascades update much less frequently and cover a large region.

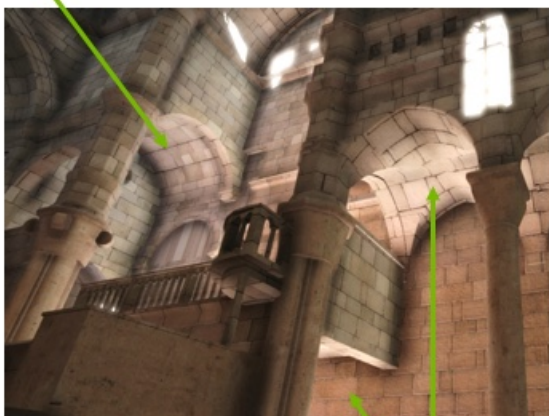
Disable visibility on very coarse cascades to halve memory and shading cost.

Let's look at the active probes around the camera

AVOIDS LEAKS

Shadow leak

Before: Classic Probes



Light leak

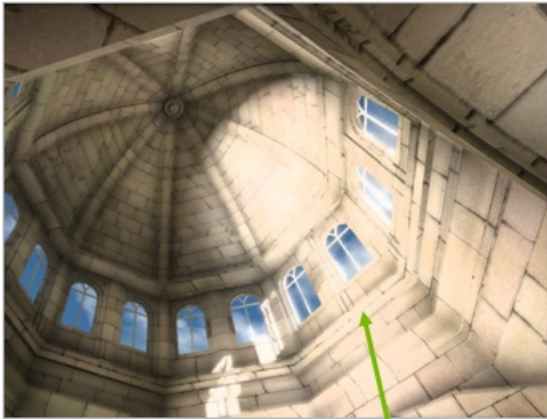
After: Dynamic Diffuse GI



Correct

AVOIDS LEAKS

Before: Classic Probes



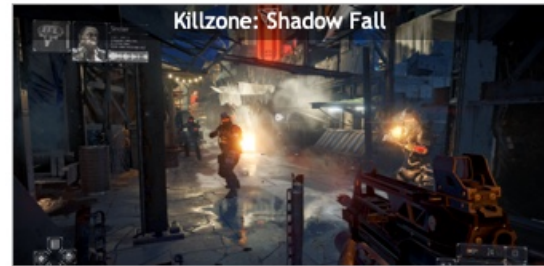
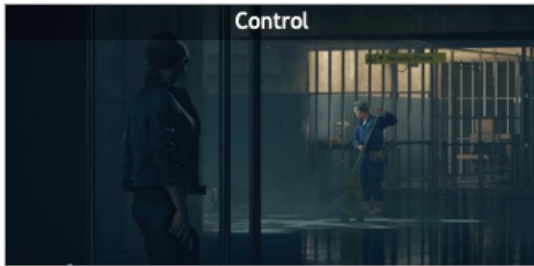
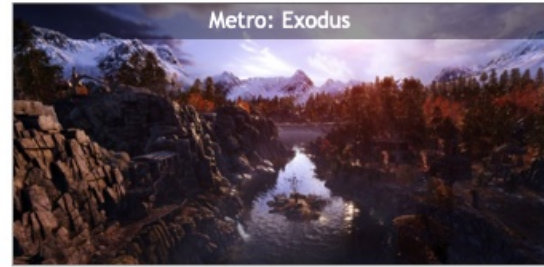
Light leak

After: Dynamic Diffuse GI

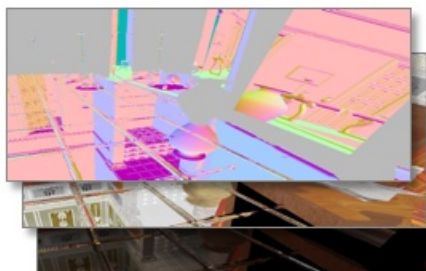


Correct

State of the Art
GLOSSY GI



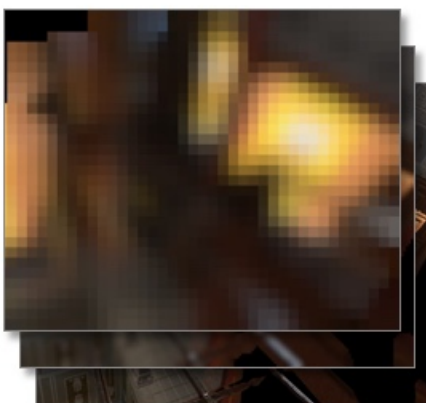
GLOSSY GI



⋮

1. Ray Trace

Trace and shade perfect mirror rays at full resolution X, $\frac{1}{2}$ resolution Y



⋮

2. Blur

Bilateral filter into MIPs, respecting edges



3. Sample

Sample in screen-space based on primary roughness and total reflection distance

1.1 ms/frame in our simple demo, including BVH update