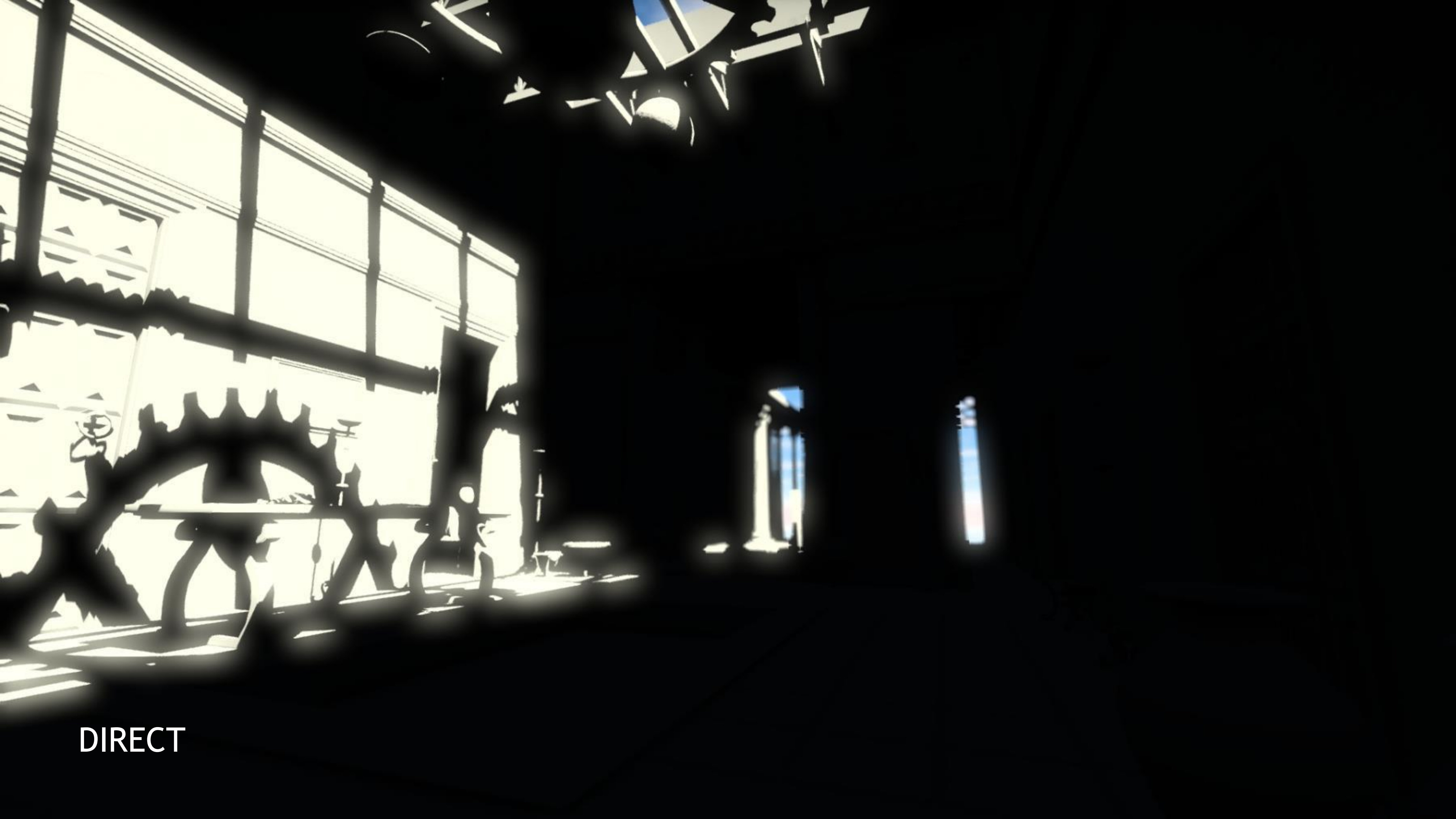
DYNAMIC DIFFUSE GLOBAL ILLUMINATION

WITH RAY-TRACED IRRADIANCE FIELDS

Morgan McGuire | April 2019



DIRECT


DIRECT + **DIFFUSE GI**

DIRECT + **DIFFUSE GI** + VOLUMETRIC


DIRECT + DIFFUSE GI + VOLUMETRIC + GLOSSY GI


DIRECT + DIFFUSE GI + VOLUMETRIC + GLOSSY GI + MATERIALS
Diffuse GI: 1.0 ms/frame
Glossy GI: 1.1 ms/frame
Throughput: 1.5 Grays/s
GeForce RTX 2080 Ti @ 1080p


Diffuse GI: 1.0 ms/frame
Glossy GI: 1.1 ms/frame
Throughput: 1.5 Grays/s
GeForce RTX 2080 Ti @ 1080p
DIRECT + DIFFUSE GI + VOLUMETRIC + GLOSSY GI + MATERIALS


BEFORE: NO GLOBAL ILLUMINATION


BEFORE: CLASSIC PROBES


AFTER: NEW DYNAMIC DIFFUSE GI


MOVING CAMERA, GEOMETRY, AND LIGHTS…

Dynamic Diffuse Global Illumination with Ray Traced Irradiance Fields

OVERVIEW

1 ms/frame dynamic diffuse global illumination on *everything* (static, dynamic, transparent, volumetric, forward, deferred)

Runs everywhere, best quality on RTX. Constant performance, varying indirect light latency across platforms.

Uses existing engine data paths, no bake time, minimizes leaks and noise. Good artist workflow.

Fresh out of the lab after six years of R&D with academic collaborators [Mara 2012, Crassin 2013, Evangelakos 2015, Donow 2016, McGuire 2017, Wang 2019, Majercik 2019]

Working with partners on game integration and art team feedback now.

No patents on the algorithm. No SDK or licensing.

AGENDA

1. Global Illumination Overview
2. Glossy GI Best Practices
3. The Diffuse GI challenge
4. **New Dynamic Diffuse GI**
5. Engine Integration
6. Examples & Demo

AGENDA

1. Global Illumination Overview
2. Glossy GI Best Practices
3. The Diffuse GI challenge
4. New Dynamic Diffuse GI
5. Engine Integration
6. Examples & Demo

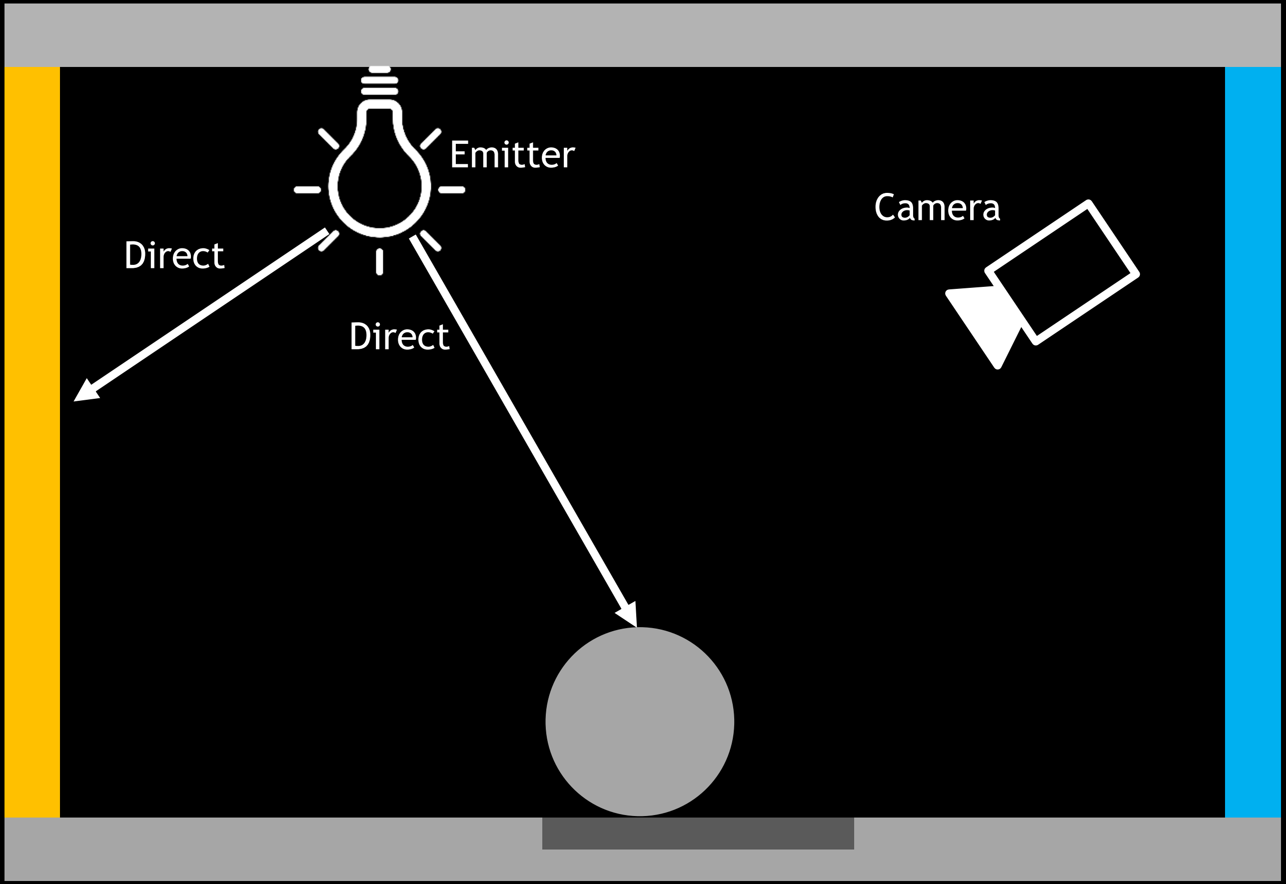
Items 1–3: Everybody

Item 4: Art Director, Project Manager

Item 5: Programmers

GLOBAL ILLUMINATION

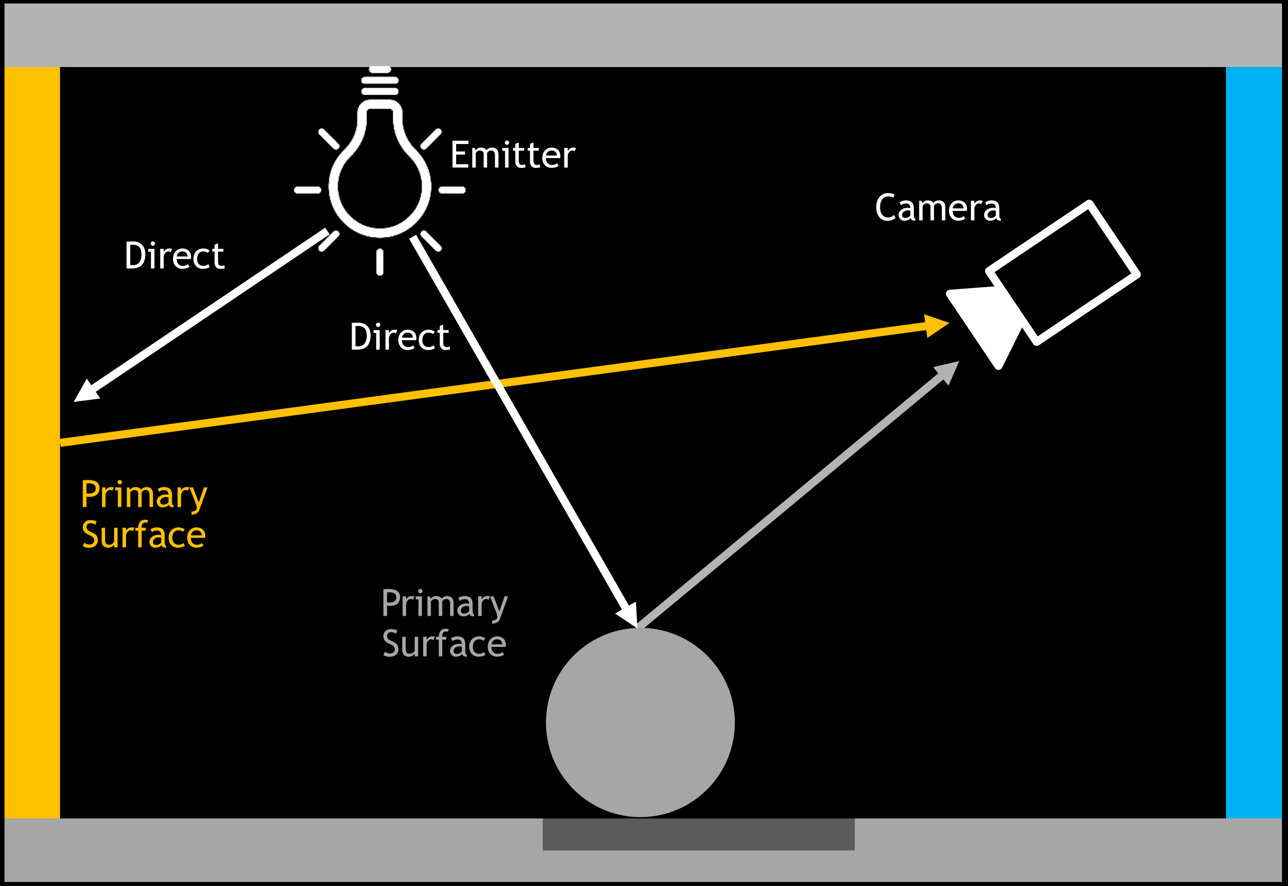
DIRECT ILLUMINATION



Direct illumination:

straight from the emitter

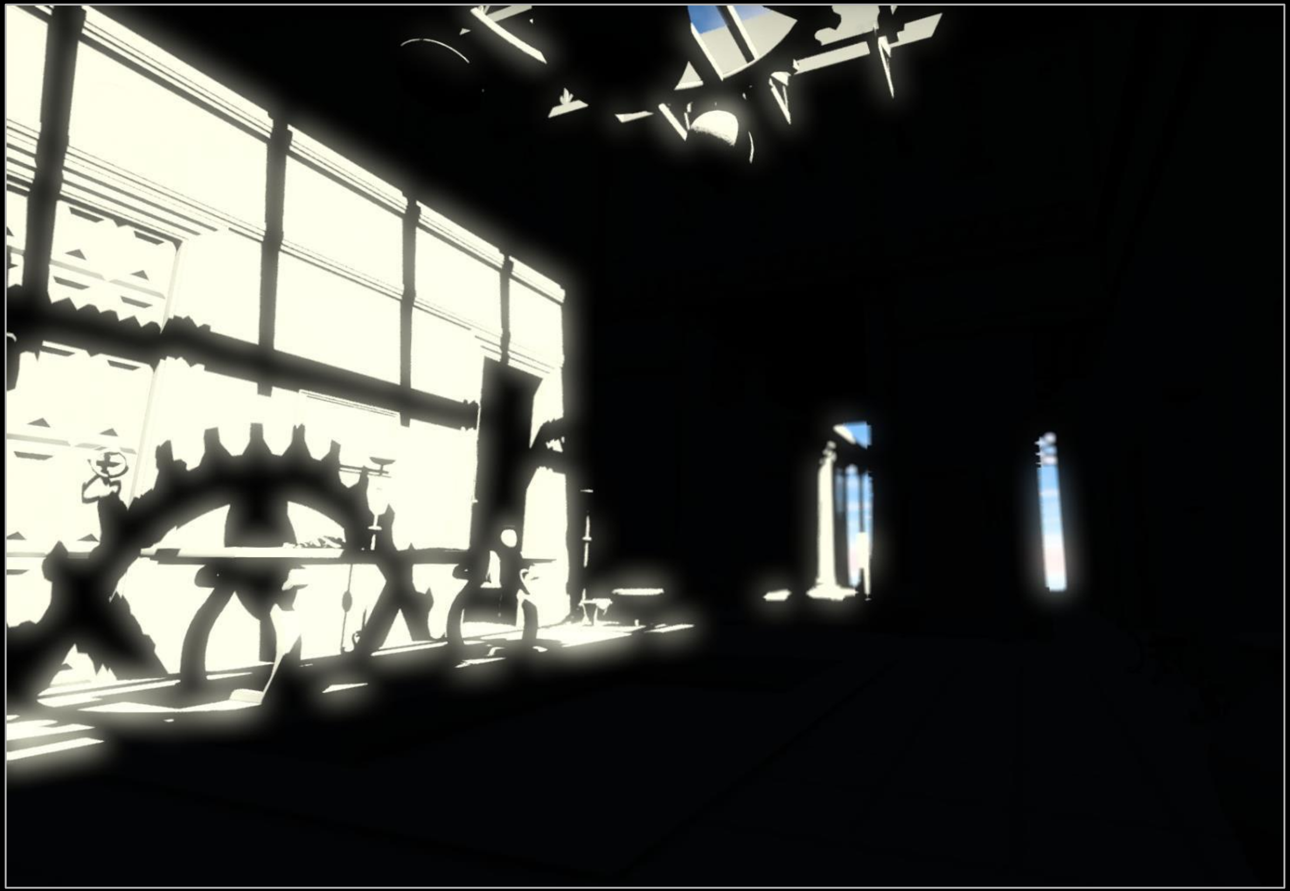
DIRECT ILLUMINATION



Direct illumination:

straight from the light emitter

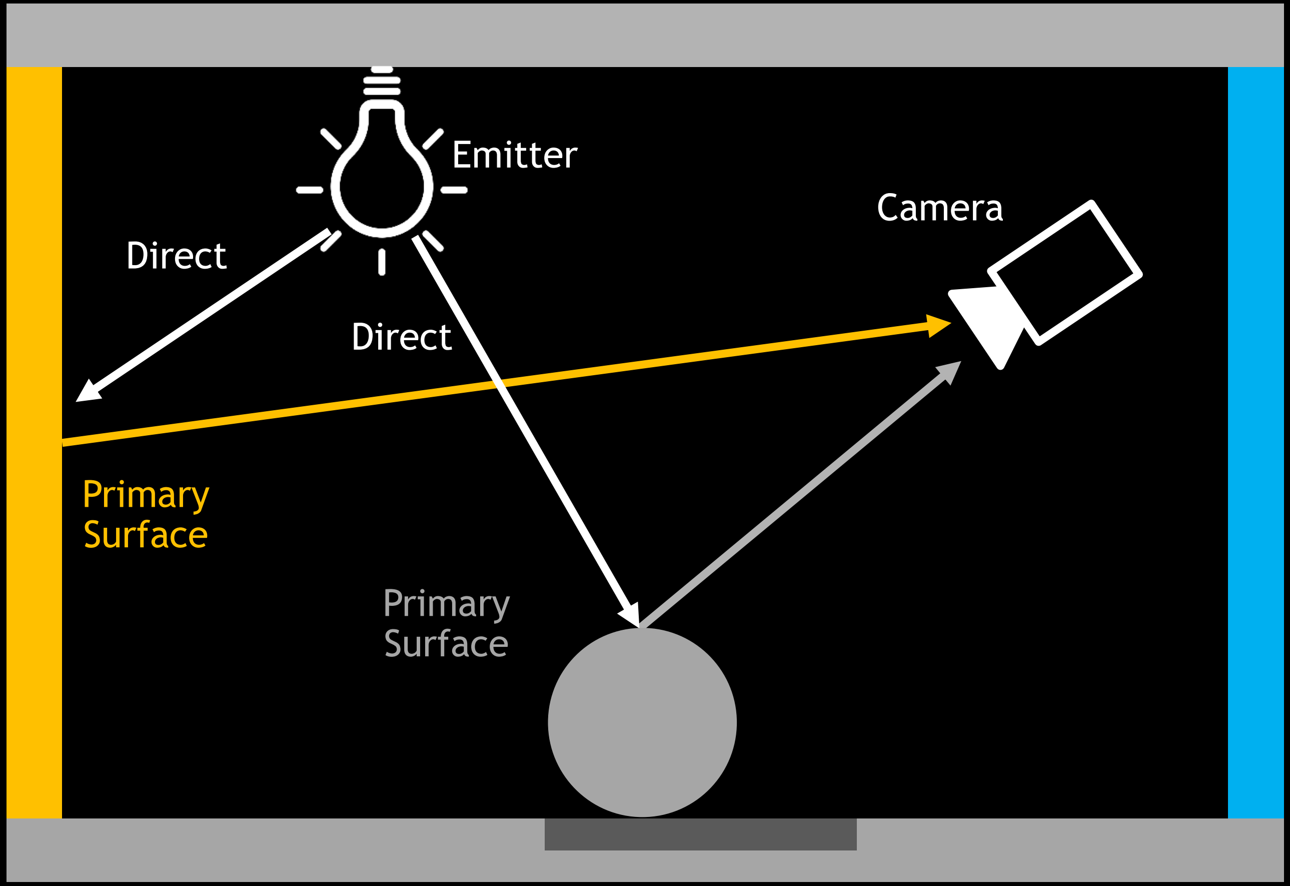
DIRECT ILLUMINATION



Direct illumination:

straight from the light emitter

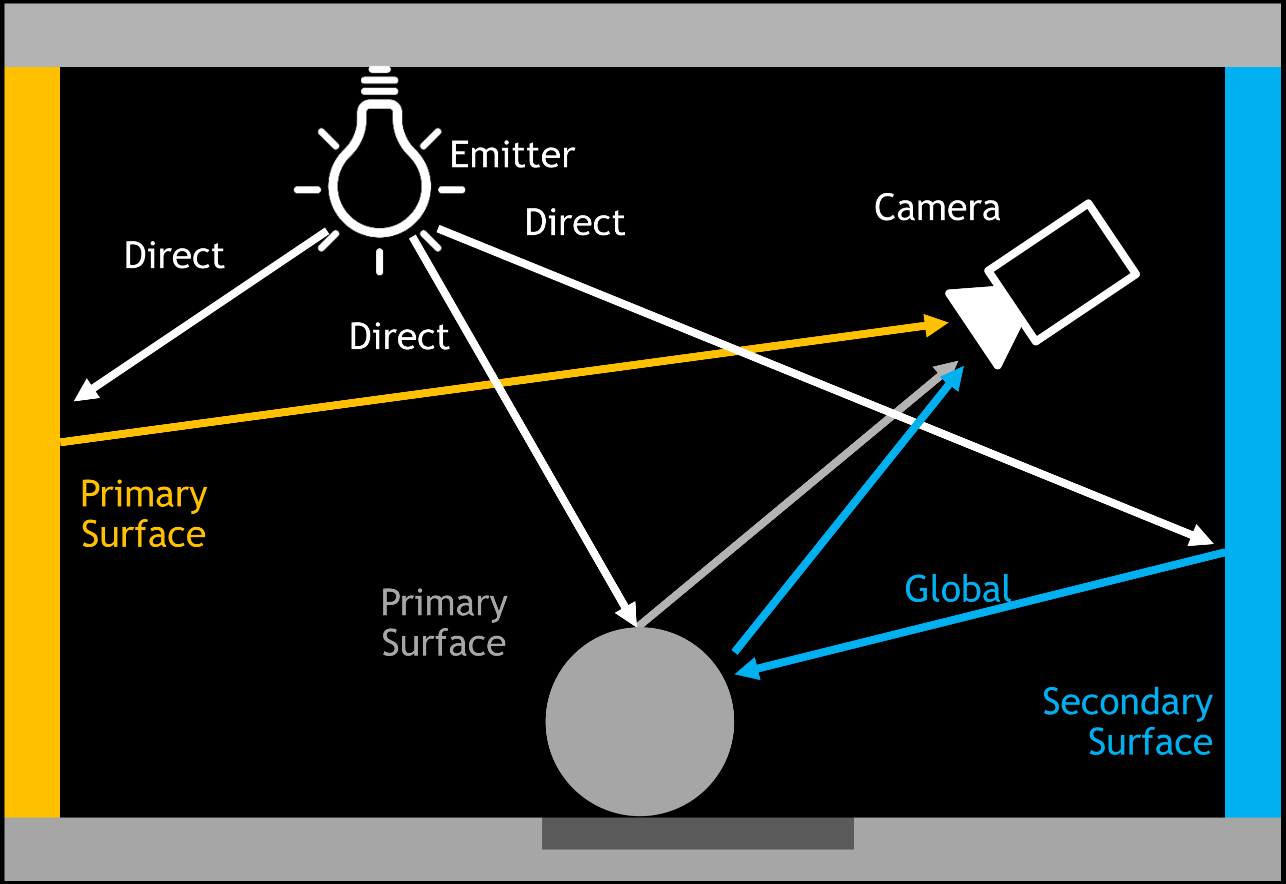
DIRECT ILLUMINATION



Direct illumination:

straight from the light emitter

GLOBAL ILLUMINATION



Direct illumination:

straight from the light emitter

Global illumination:

bounces off at least one other surface

GLOBAL ILLUMINATION

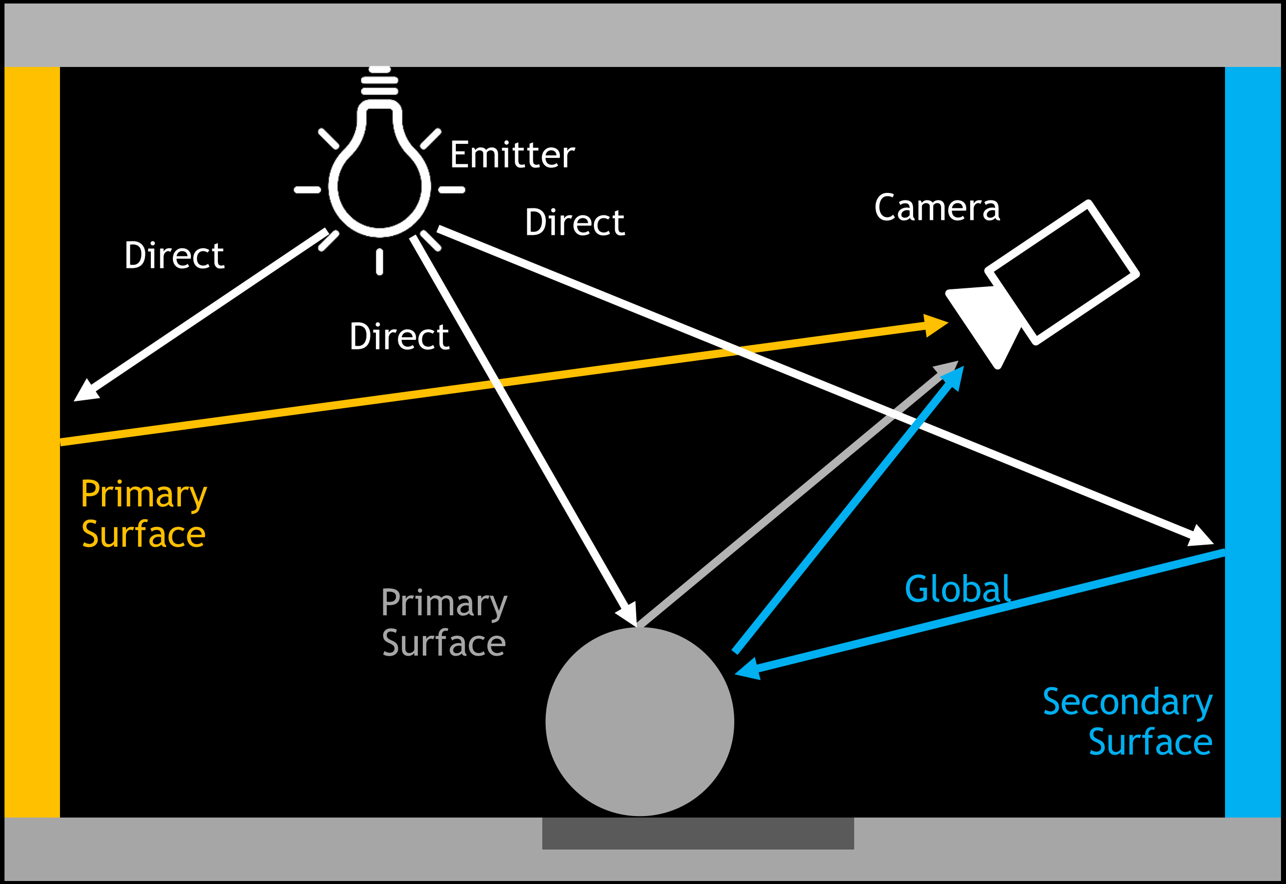
Direct illumination:

straight from the light emitter

Global illumination:

bounces off at least one other surface

GLOBAL ILLUMINATION



Direct illumination:

straight from the light source

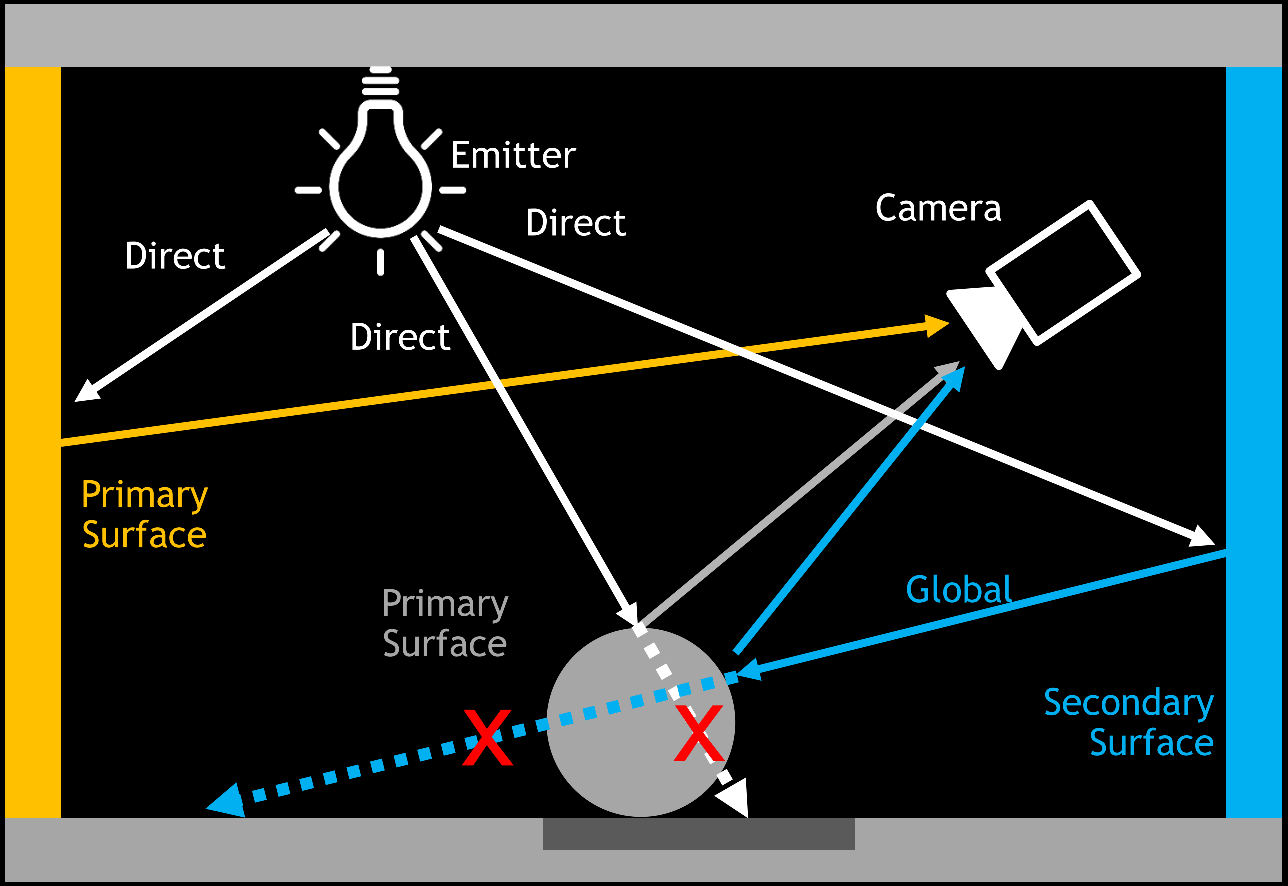
Global illumination:

bounces off at least one other surface

Visibility:

unobstructed line of sight
primary surface: visibility to camera
direct shadow: visibility to emitter
G.I. "visibility": any two points

GLOBAL ILLUMINATION



Direct illumination:

straight from the light source

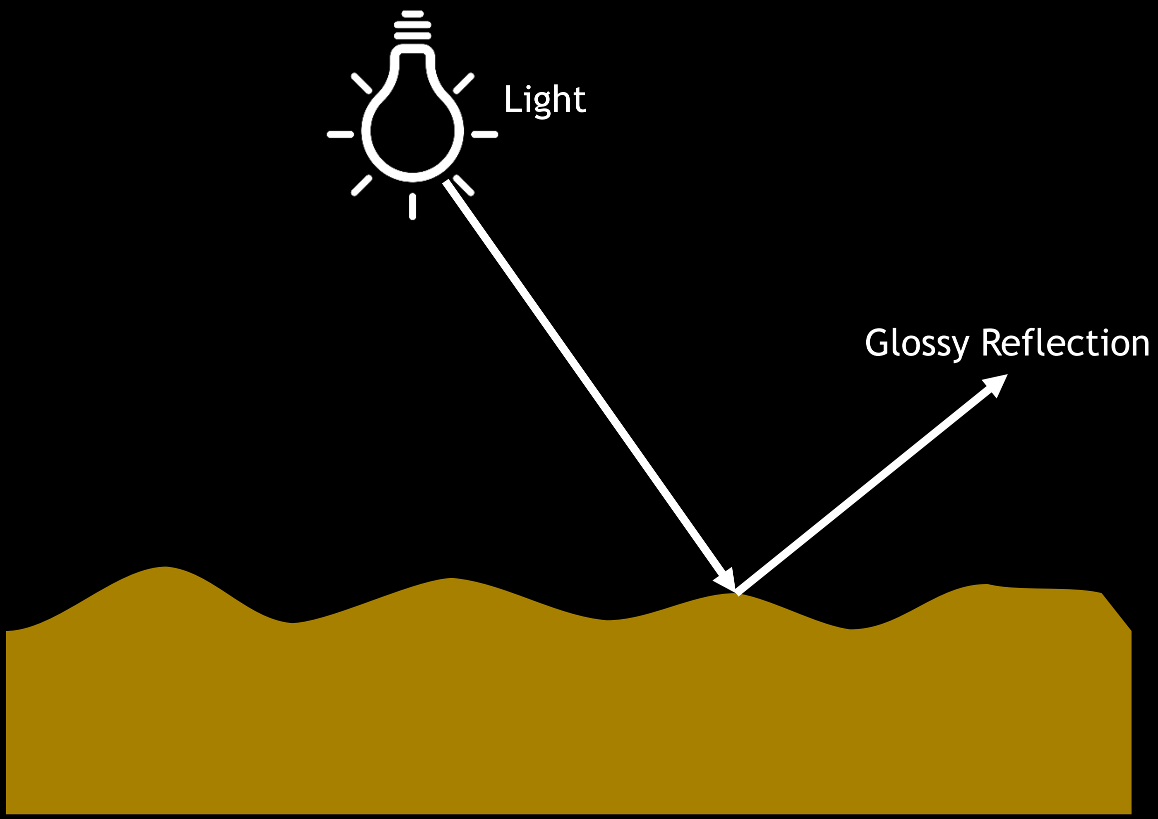
Global illumination:

bounces off at least one other surface

Visibility:

unobstructed line of sight
primary surface: visibility to camera
direct shadow: visibility to emitter
G.I. "visibility": any two points

GLOSSY REFLECTION



Glossy Reflection:

(e.g., specular, microfacet, GGX, etc.)

- reflects off the surface

- only visible near mirror angle

GLOSSY REFLECTION

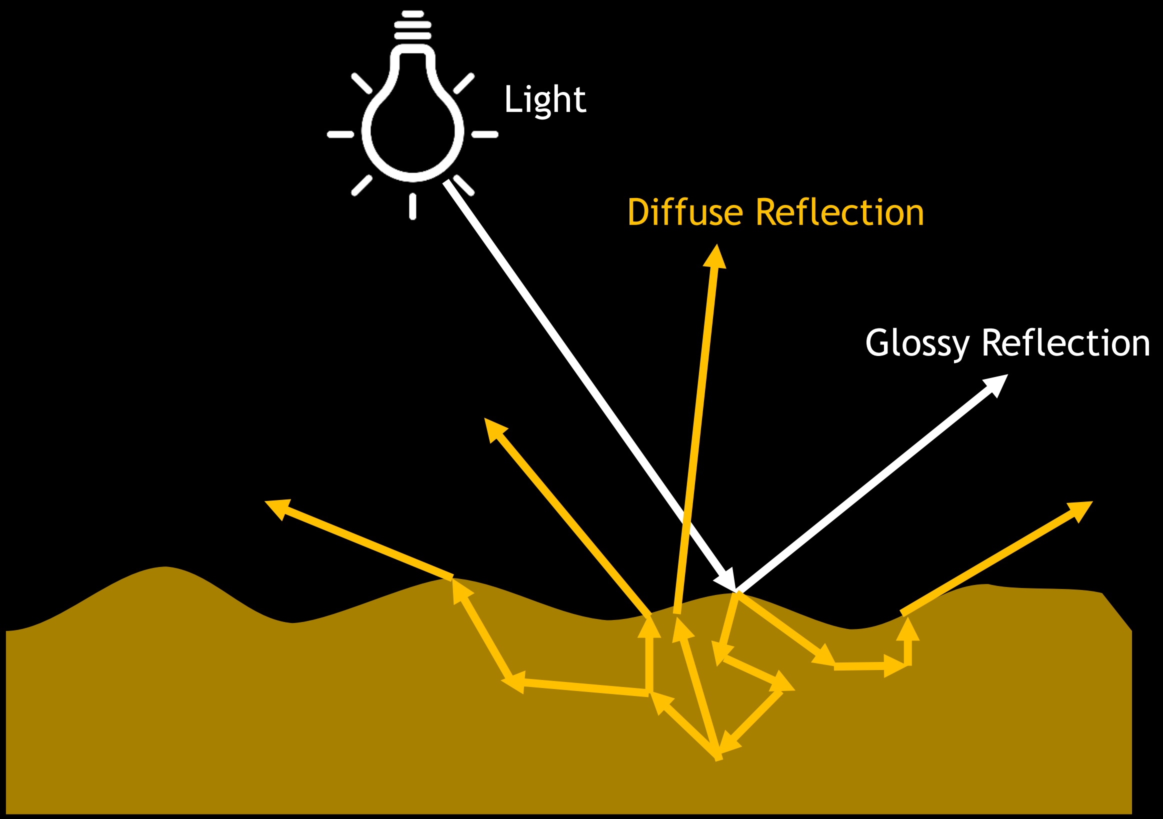
Glossy Reflection:

(e.g., specular, microfacet, GGX, etc.)

- reflects off the surface

- only visible near mirror angle

DIFFUSE REFLECTION



Glossy Reflection:

(e.g., specular, microfacet, GGX, etc.)

- reflects off the surface

- only visible near mirror angle

Diffuse Reflection:

(e.g., matte, Lambertian, etc.)

- scatters just below the surface

- visible from all directions

DIFFUSE REFLECTION

Glossy Reflection:

(e.g., specular, microfacet, GGX, etc.)
- reflects off the surface
- only visible near mirror angle

Diffuse Reflection:

(e.g., matte, Lambertian, etc.)
- scatters just below the surface
- visible from all directions

Today: Dynamic **Diffuse Global Illumination** with correct **Visibility**

Glossy Reflection:

(e.g., specular, microfacet, GGX, etc.)
- reflects off the surface
- only visible near mirror angle

Diffuse Reflection:

(e.g., matte, Lambertian, etc.)
- scatters just below the surface
- visible from all directions

GLOSSY GI

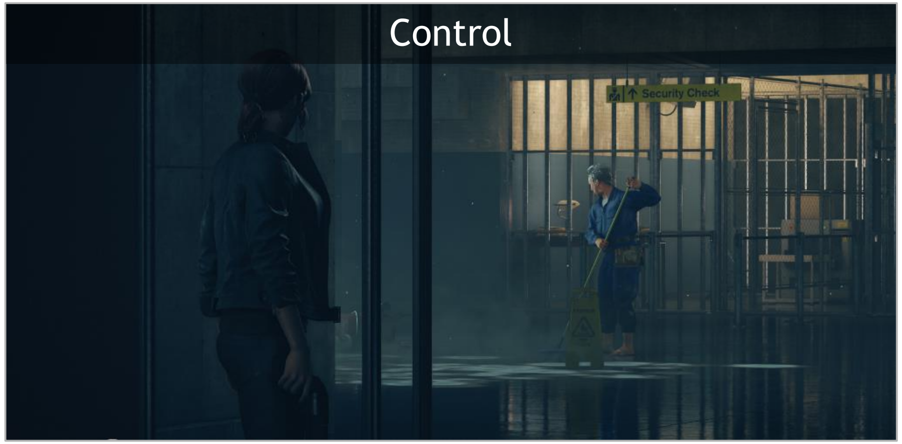
STATE OF THE ART

State of the Art

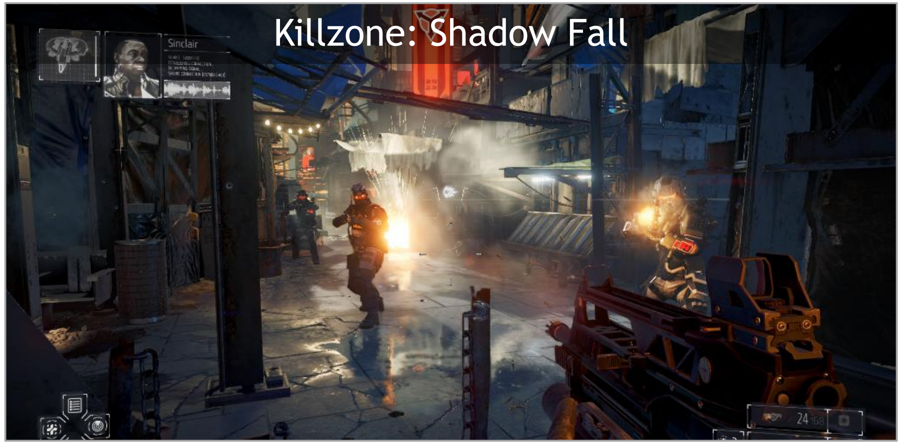
GLOSSY GI

Battlefield V

Metro: Exodus

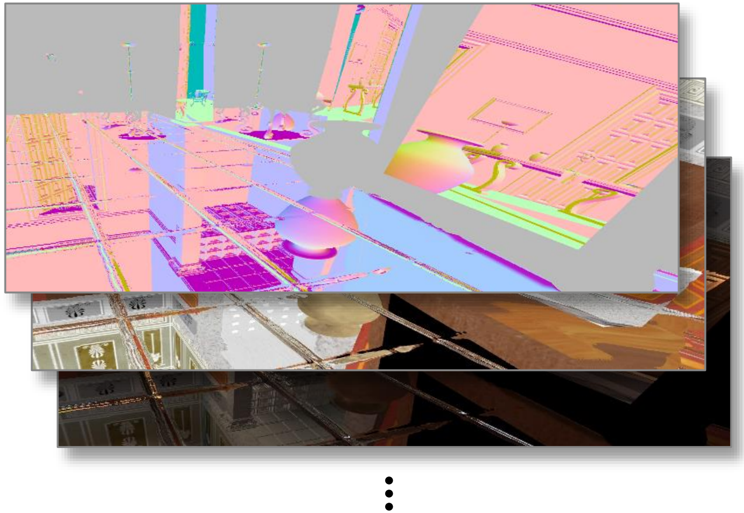


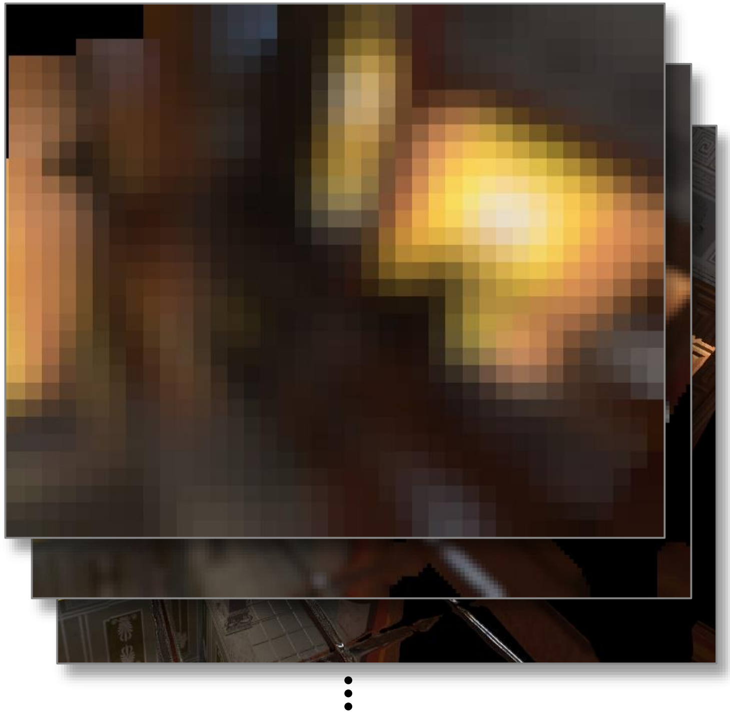
Control



Killzone: Shadow Fall

GLOSSY GI





1. Ray Trace

Trace and shade perfect mirror rays at full resolution X, ½ 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

GLOSSY GI IMPLEMENTATION

Heavy lifting is all in your existing forward or deferred shader, which runs on ray hits. Uses shadow maps, regular materials, etc. so no special shading code for the base implementation.

We use half resolution only *vertically* because that gives a good performance to quality tradeoff on high-end hardware. Most reflections are on floors, and they'll be blurred vertically in screen space anyway.

Stretch to full resolution and bilateral blur into MIP-maps. Gaussian kernel, normal & depth weighting. Expand out into untraced areas so that trilinear fetches don't hit black. MIP generation is about 0.1 ms of total time.

When rendering the camera view, compute MIP level to gather from smoothness, distance to primary surface + distance to reflected surface. Produces proper distance fading.

To improve quality: address flicker. final-frame TAA can help and hurt. Use everything you know about filtering and flickering *inside* the glossy shader: MIP bias, bump to roughness, TAA/FXAA on the glossy trace, LOD.

Can optimize down to about 0.5 ms/frame (see Battlefield V): Combine with screen-space ray tracing and environment maps, use geometric and material level of detail, apply checkerboarding plus upsampling, DLSS.

DIFFUSE GI

STATE OF THE ART

Real-Time Diffuse GI

STATE OF THE ART

Baked light maps

Light propagation volumes

Sparse voxel cone tracing

Denoised ray tracing

Baked irradiance probes

IRRADIANCE PROBES

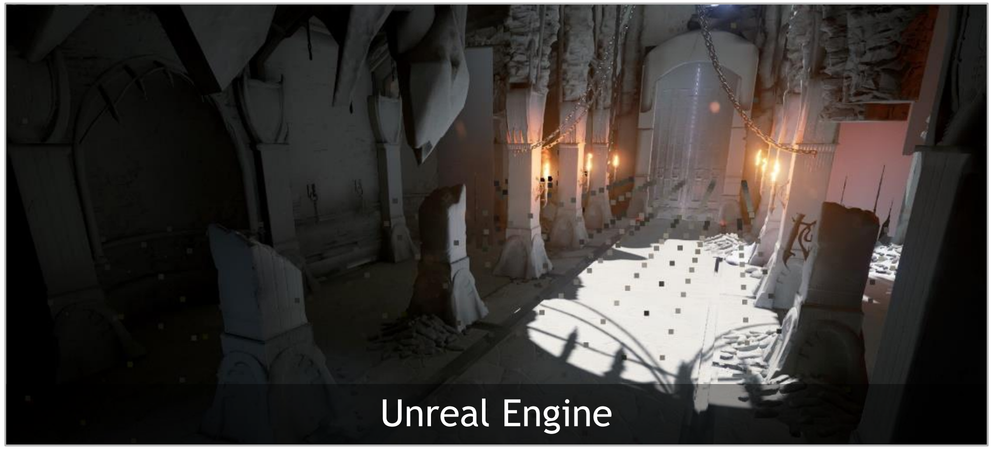
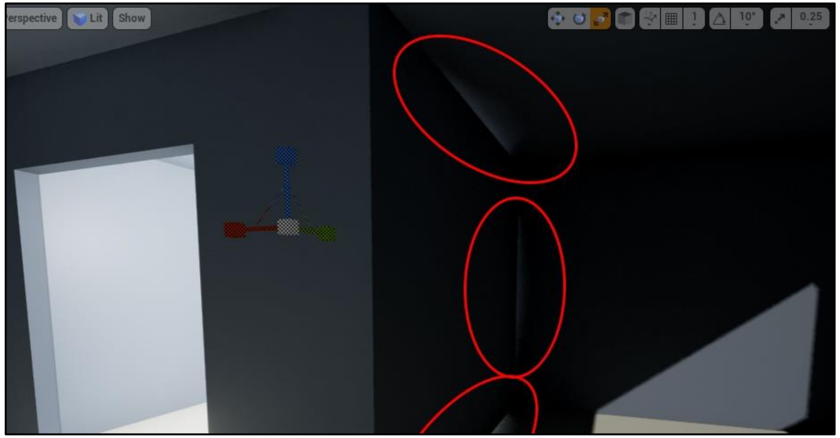


Image Credits: Geomerics and Ninja Theory, https://unity3d.com/learn/tutorials/topics/graphics/probe-lighting, https://docs.unrealengine.com/en-us/Engine/Rendering/LightingAndShadows/IndirectLightingCache, https://unity3d.com/learn/tutorials/topics/graphics/probe-lighting

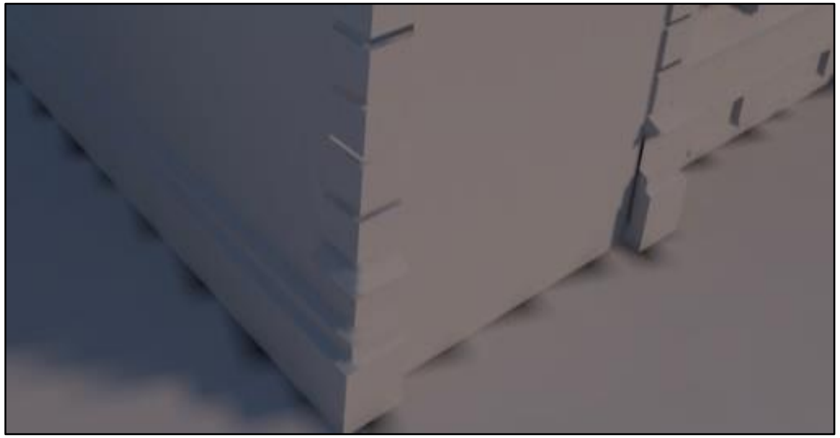
LIGHT & SHADOW LEAKS

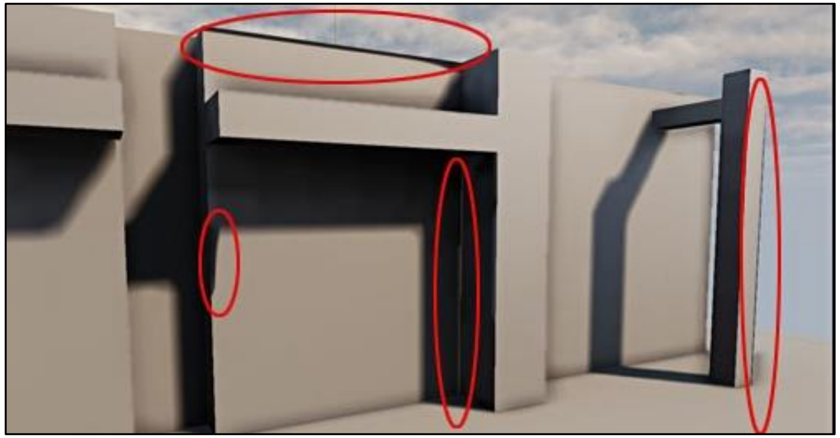
[Hooker 2016]

LIGHT & SHADOW LEAKS



[Iwanicky 2013]





[Rakhteenko 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

NEW: DYNAMIC DIFFUSE GI

WITH RAY-TRACED IRRADIANCE FIELDS

UPGRADING PROBES WITH VISIBILITY

Classic Probes: Fast to sample, noise-free, work with characters and transparents, parameterization-free, already in your engine.

Upgrade:

- **Leaks**: Store visibility information to prevent light and shadow leaking.
- **Dynamic**: Asynchronous GPU ray trace directly into low resolution probes, gather blending
- **Workflow**: Art cost is in avoiding leaks and bake time. Real-time + no leaks fixes worflow.

PROBE PLACEMENT

Grid

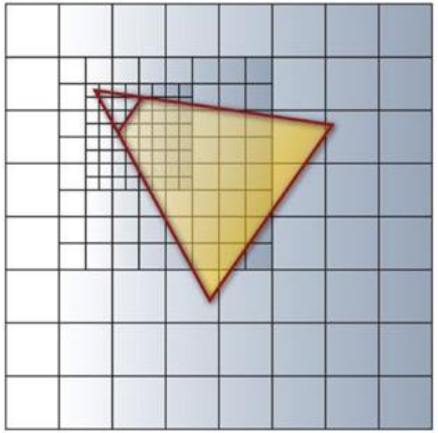
Optionally optimize around static geo [Chajdas 2011, Donow 2016, Wang et al. 2019, Unity]

Artists may override placement

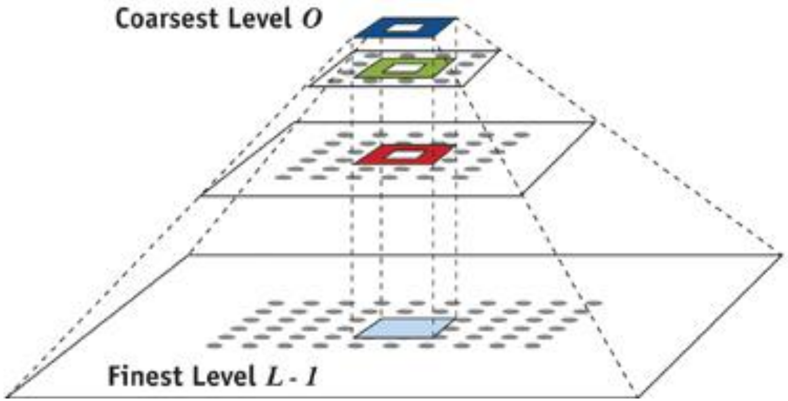
Cascades

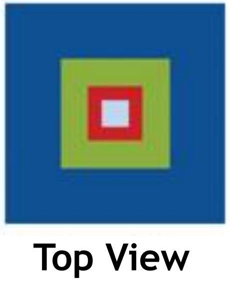
32 x 4 x 32 = 4 k probes around the camera that update frequently.

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



[Kaplanyan and Dachsbacher 2010]



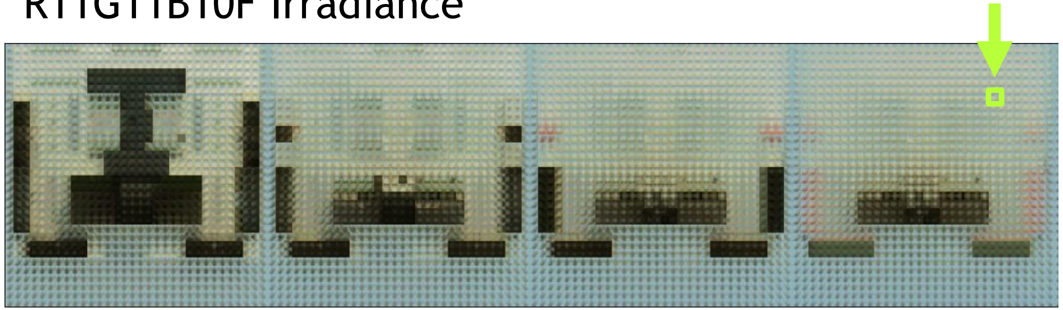


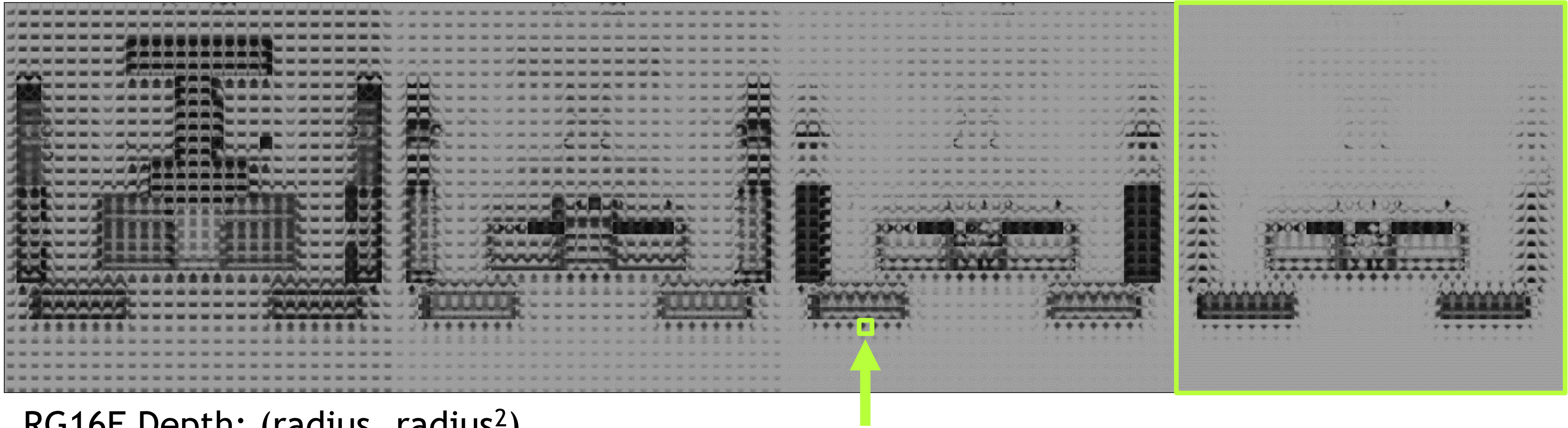
[Asirvatham and Hoppe 2005]

DATA STRUCTURE

6x6-texel probe

R11G11B10F Irradiance





RG16F Depth: (radius, $radius^2$)

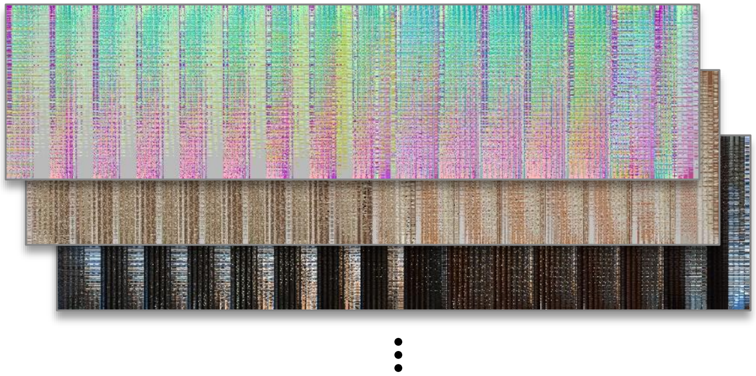
16x16-texel probe

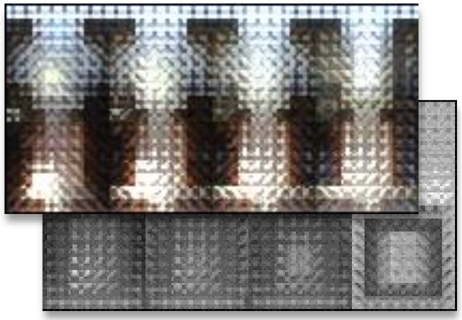
32x32-probe scene layer

5 MB GPU RAM for 8k Probes

DYNAMIC DIFFUSE GI

Independent of framerate and screen resolution





1. Ray Trace

Trace and shade packed rays from active probes.
(Pack into the bottom of the Glossy GI ray pass)

Uses previous iteration for shading: infinite bounce GI.

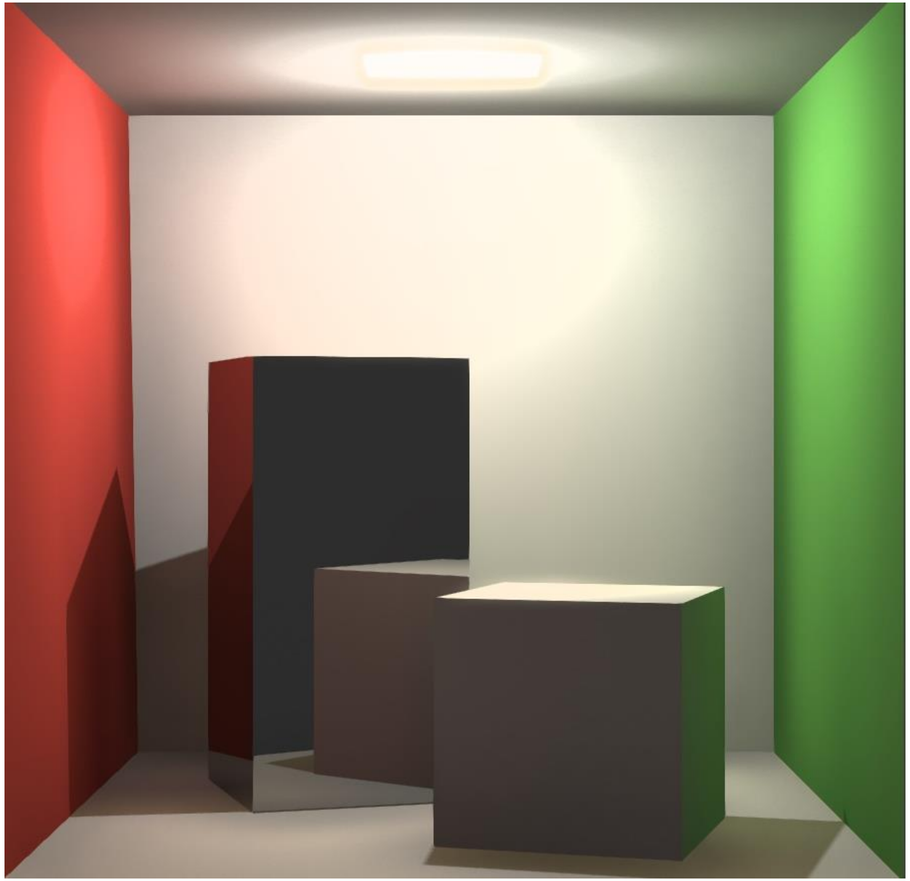
2. Blend

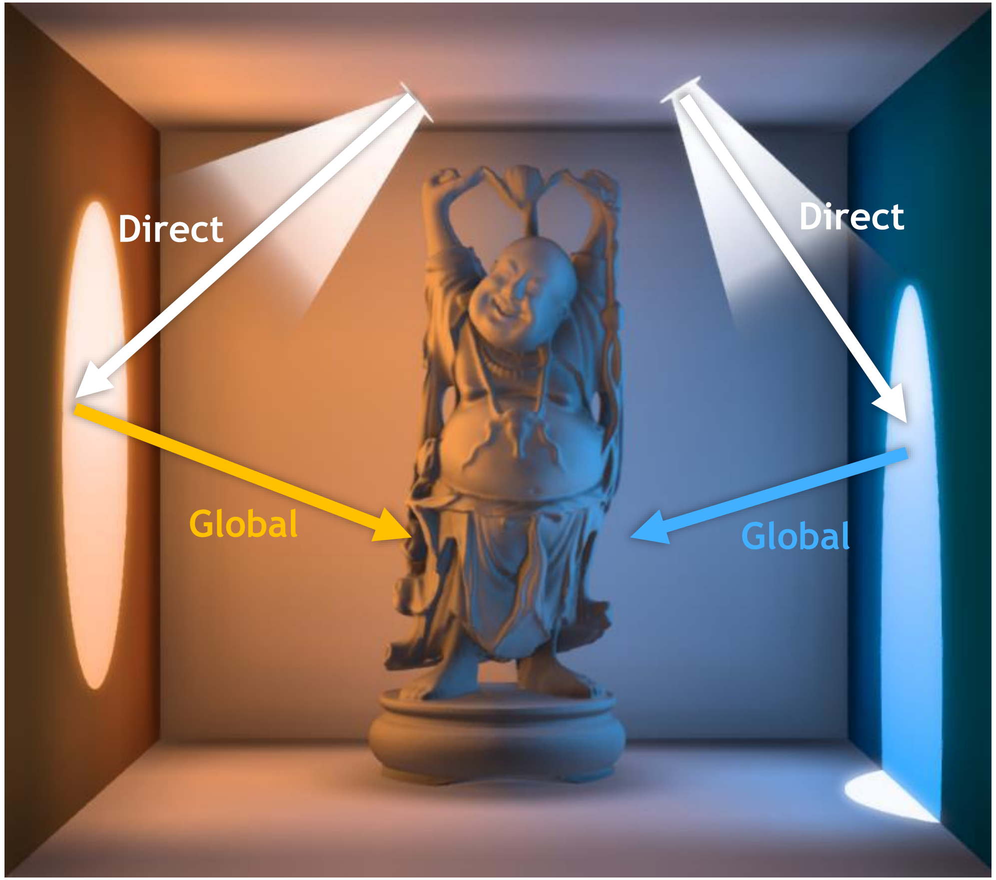
Blend irradiance and depth into probes.
Duplicate probe border texels for fast bilinear.

3. Sample

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

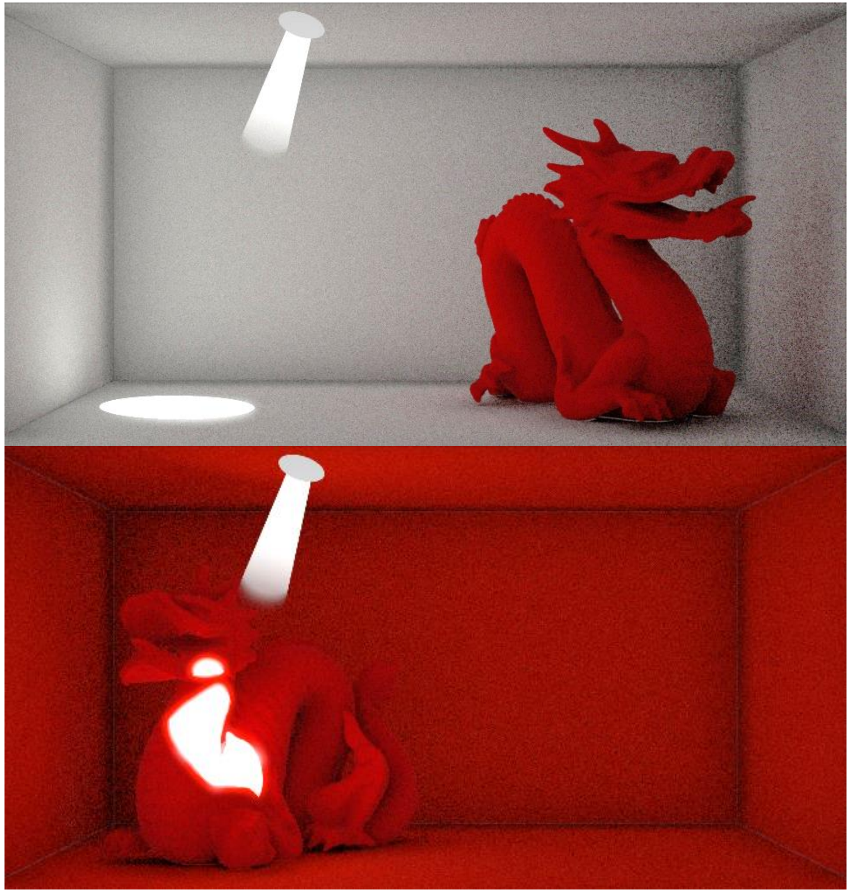
ACCURATE & NOISE FREE



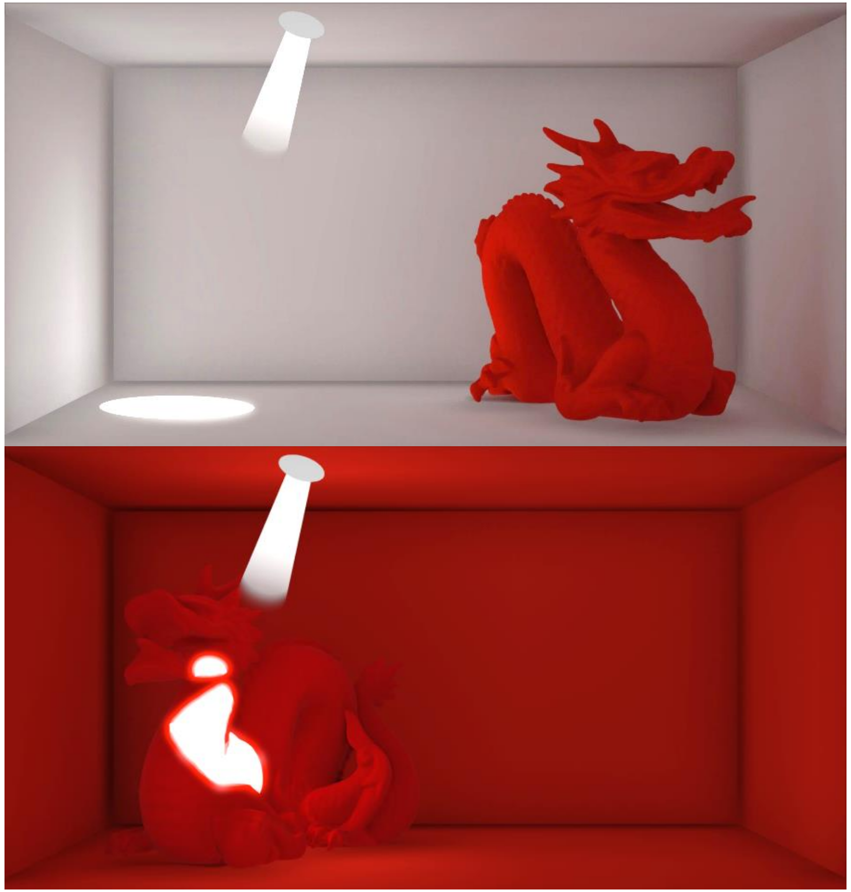


ACCURATE & NOISE FREE

Path Tracing

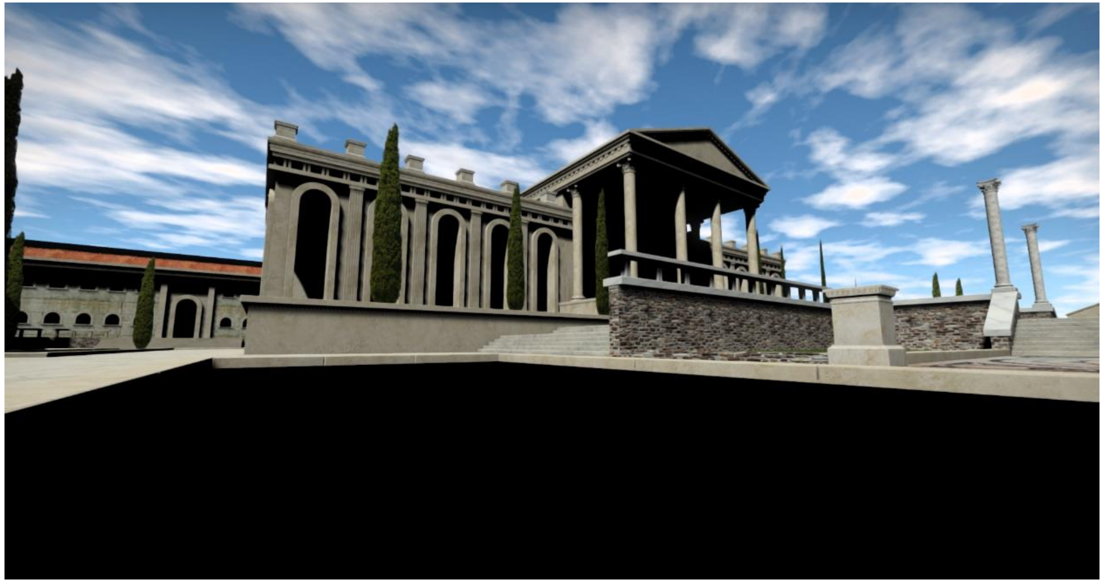


Dynamic Diffuse GI



REALISTIC

Direct Illumination Only



+ Dynamic Diffuse GI

REALISTIC

Direct Illumination Only

+ Dynamic Diffuse GI

REALISTIC

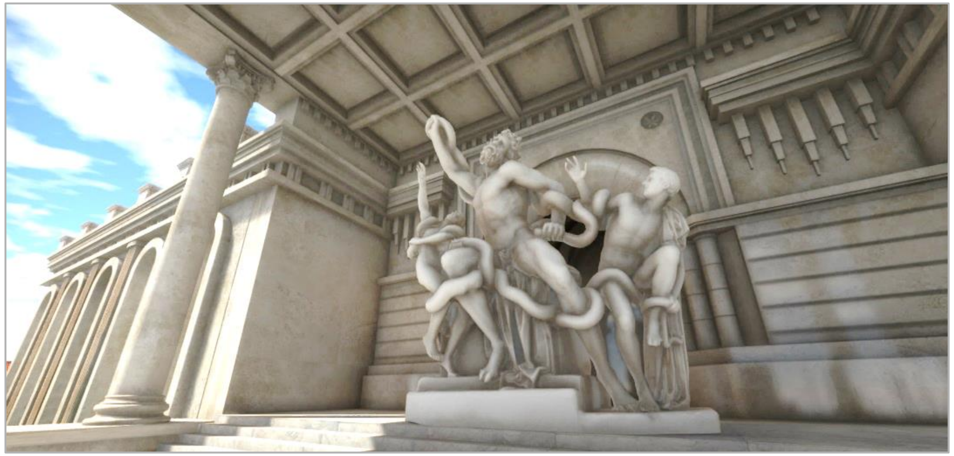
Direct Illumination Only

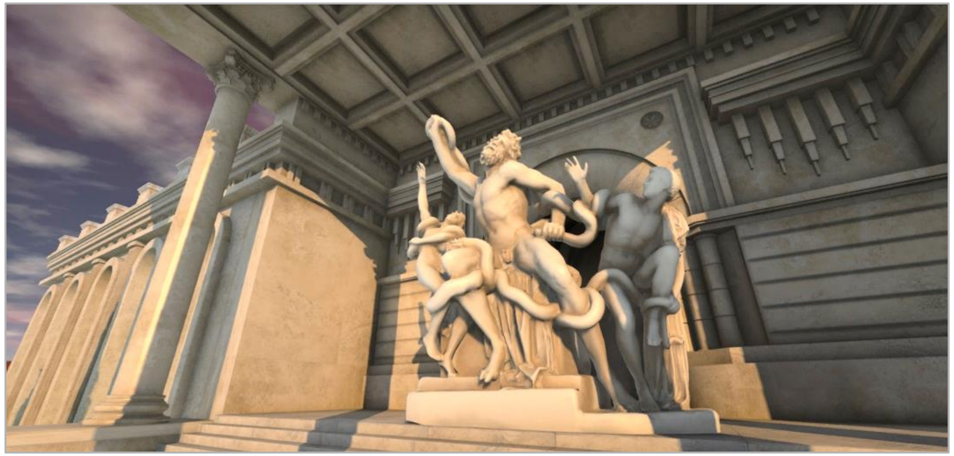
+ Dynamic Diffuse GI

DYNAMIC LIGHTING

Afternoon

Evening

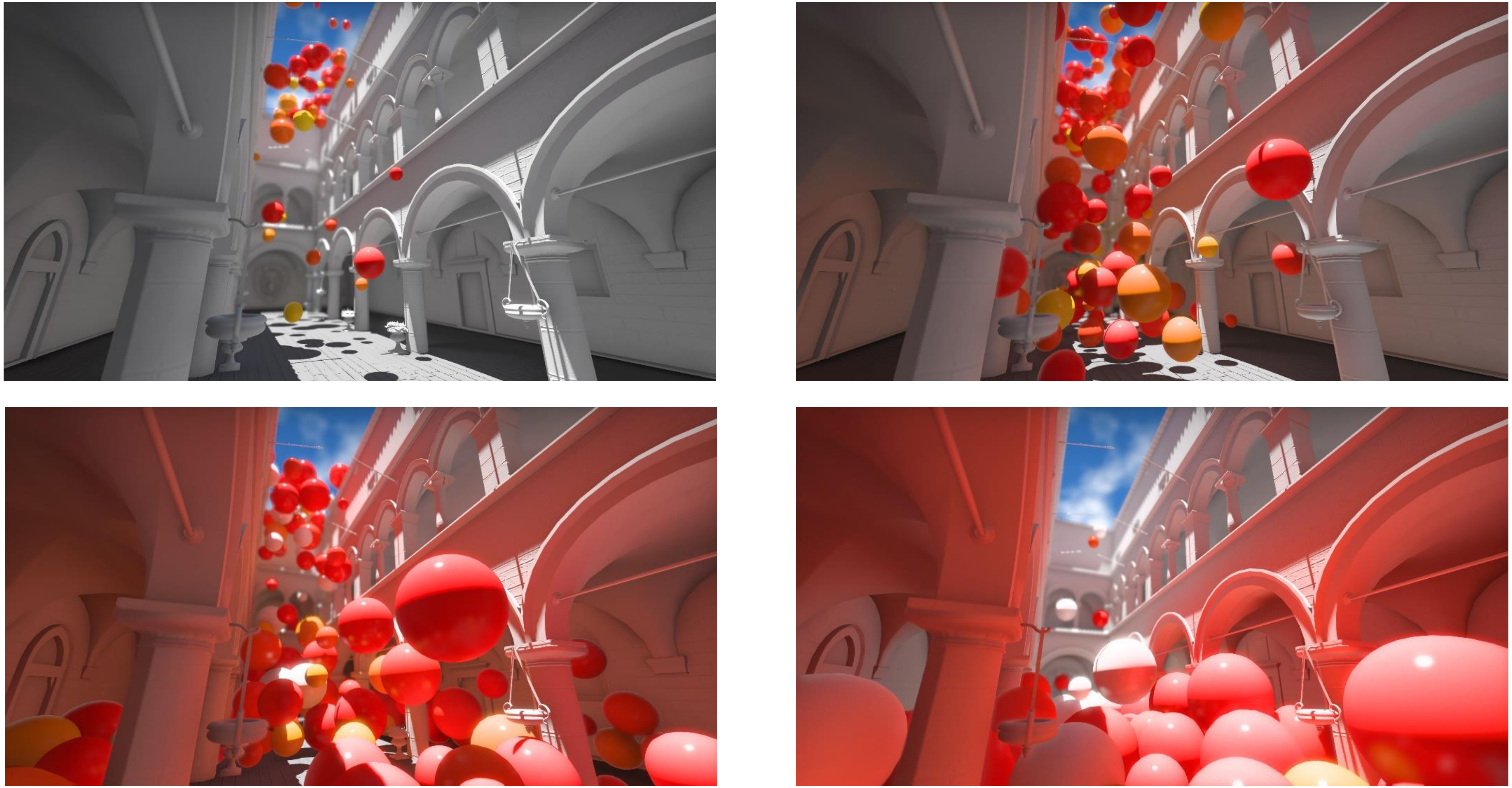




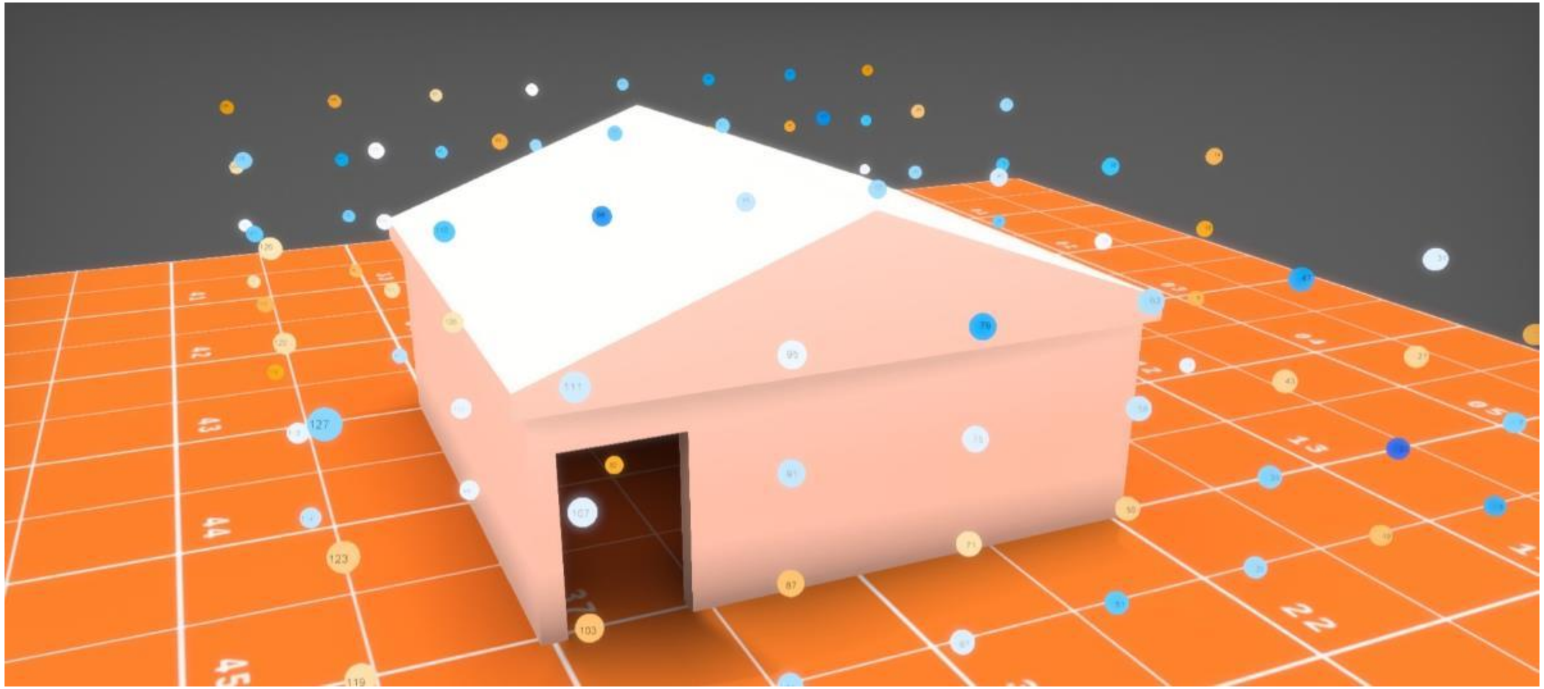
LARGE SCENES


DYNAMIC
GEOMETRY

DYNAMIC GEOMETRY



AVOIDS LEAKS

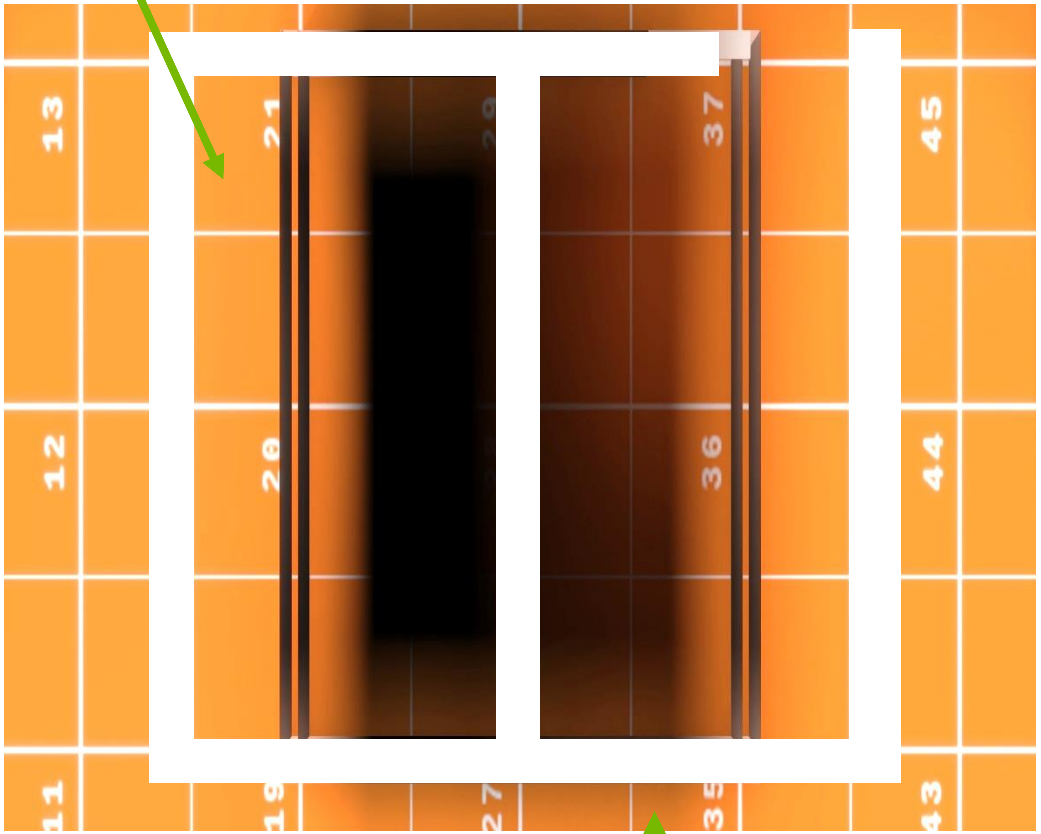


AVOIDS LEAKS

AVOIDS LEAKS

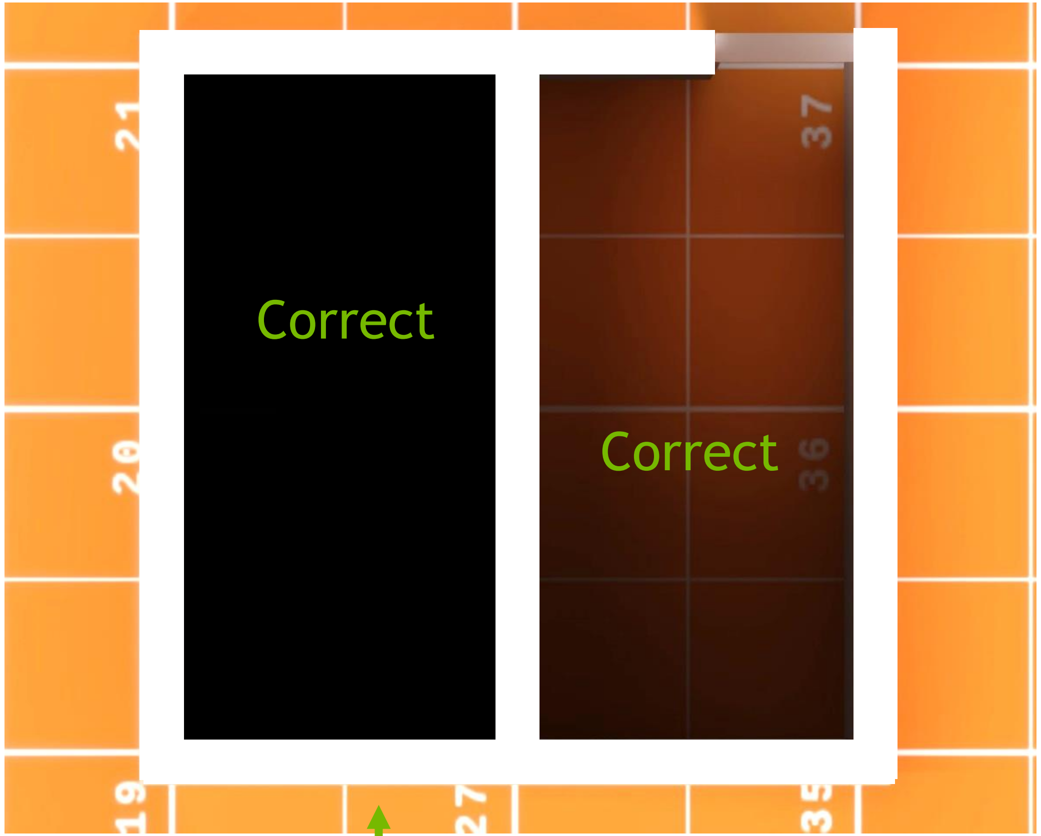
Light leak

Before: Classic Probes



Shadow leak

After: Dynamic Diffuse GI



Correct

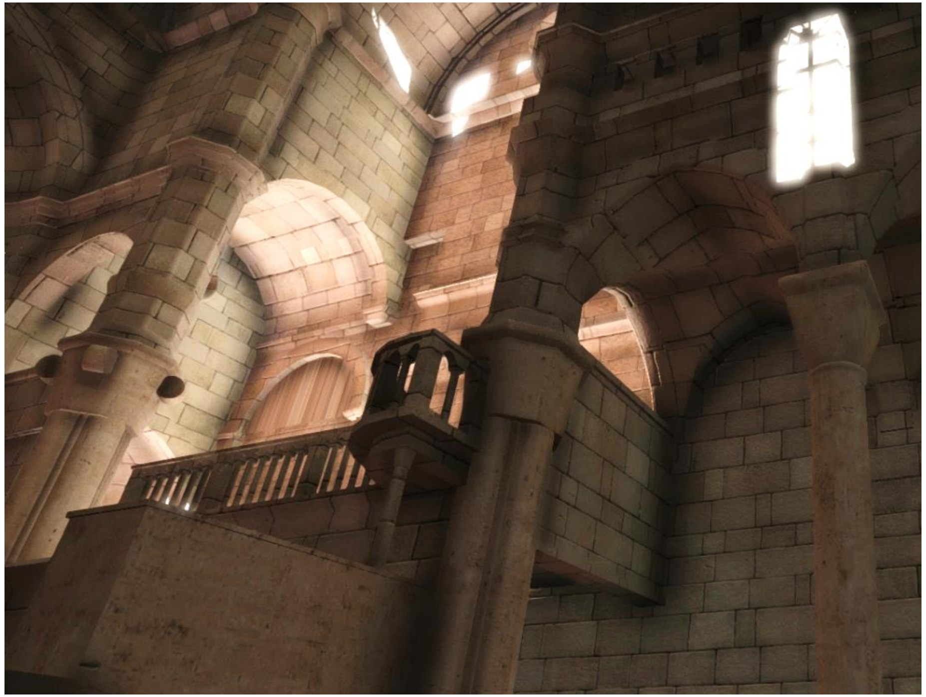
AVOIDS LEAKS

Shadow leak

Before: Classic Probes

Light leak

After: Dynamic Diffuse GI



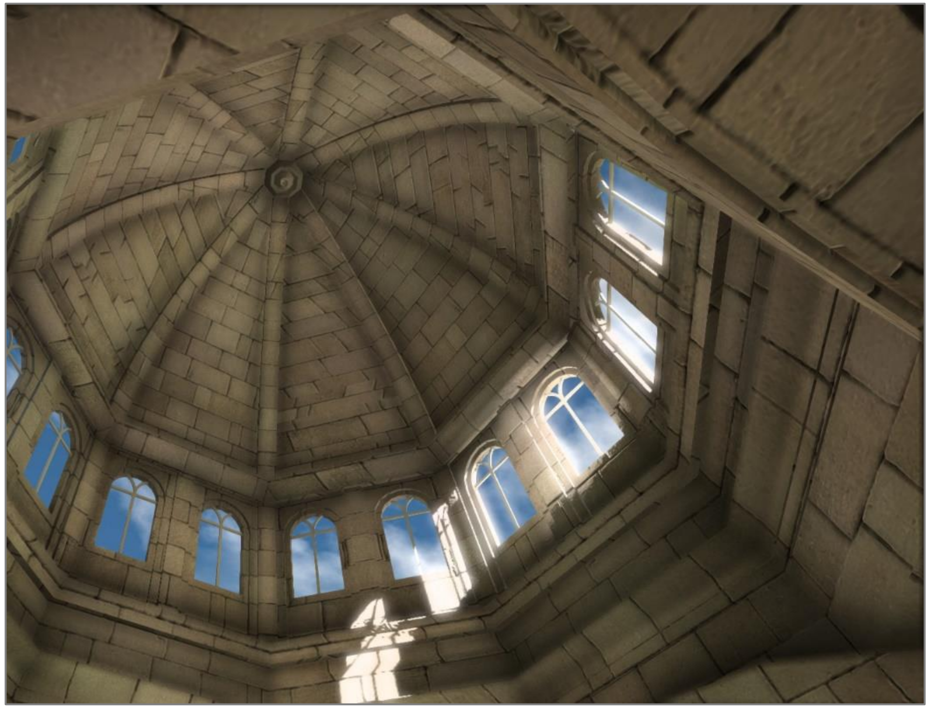
Correct

AVOIDS LEAKS

Before: Classic Probes

Light leak

After: Dynamic Diffuse GI



Correct

LIMITATIONS

Self-shadow bias must be tuned to geometry thickness

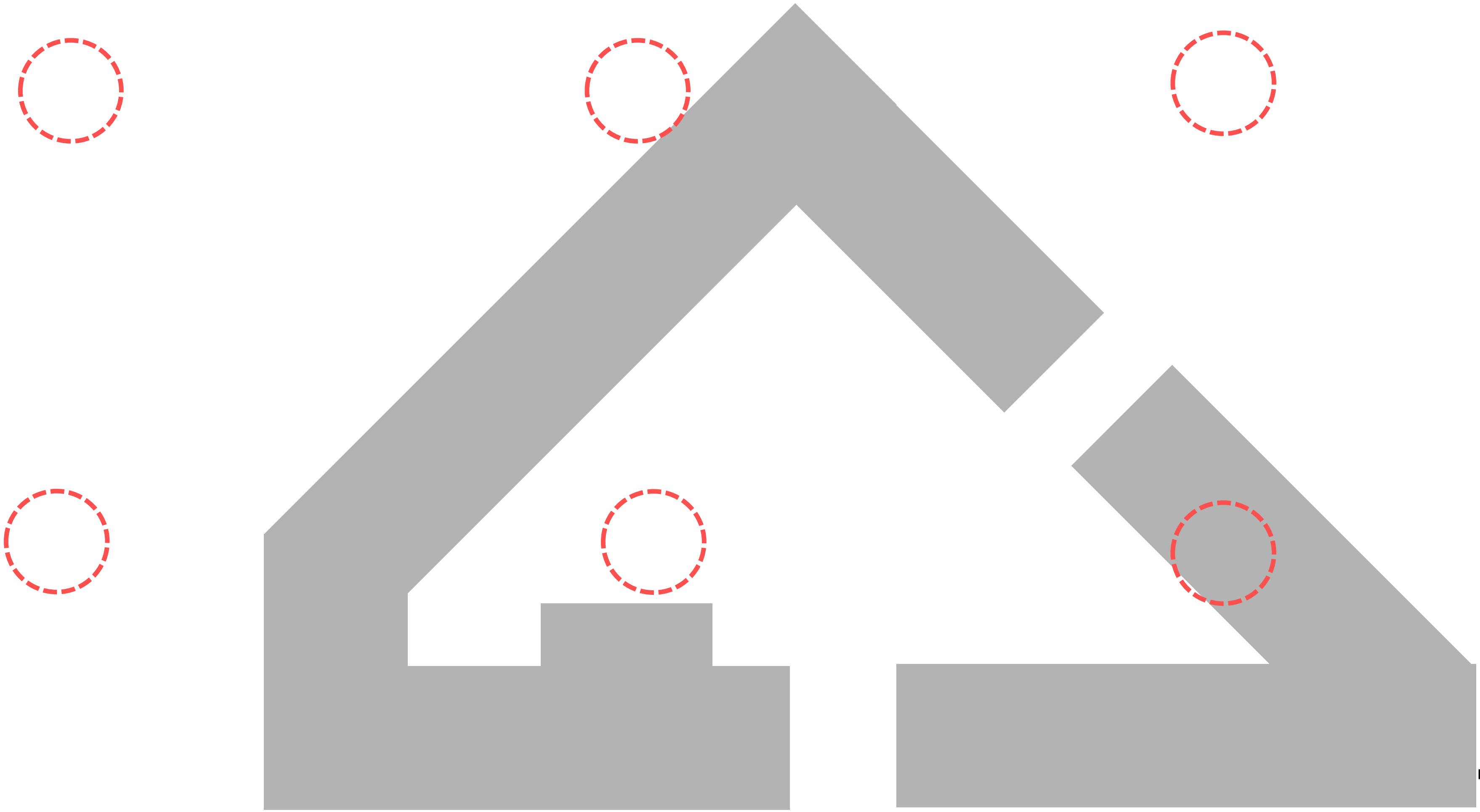
Light crossfades in time under dramatic changes

Blurrier than light maps (use **screen-space AO** for contact shadows)

ENGINE INTEGRATION

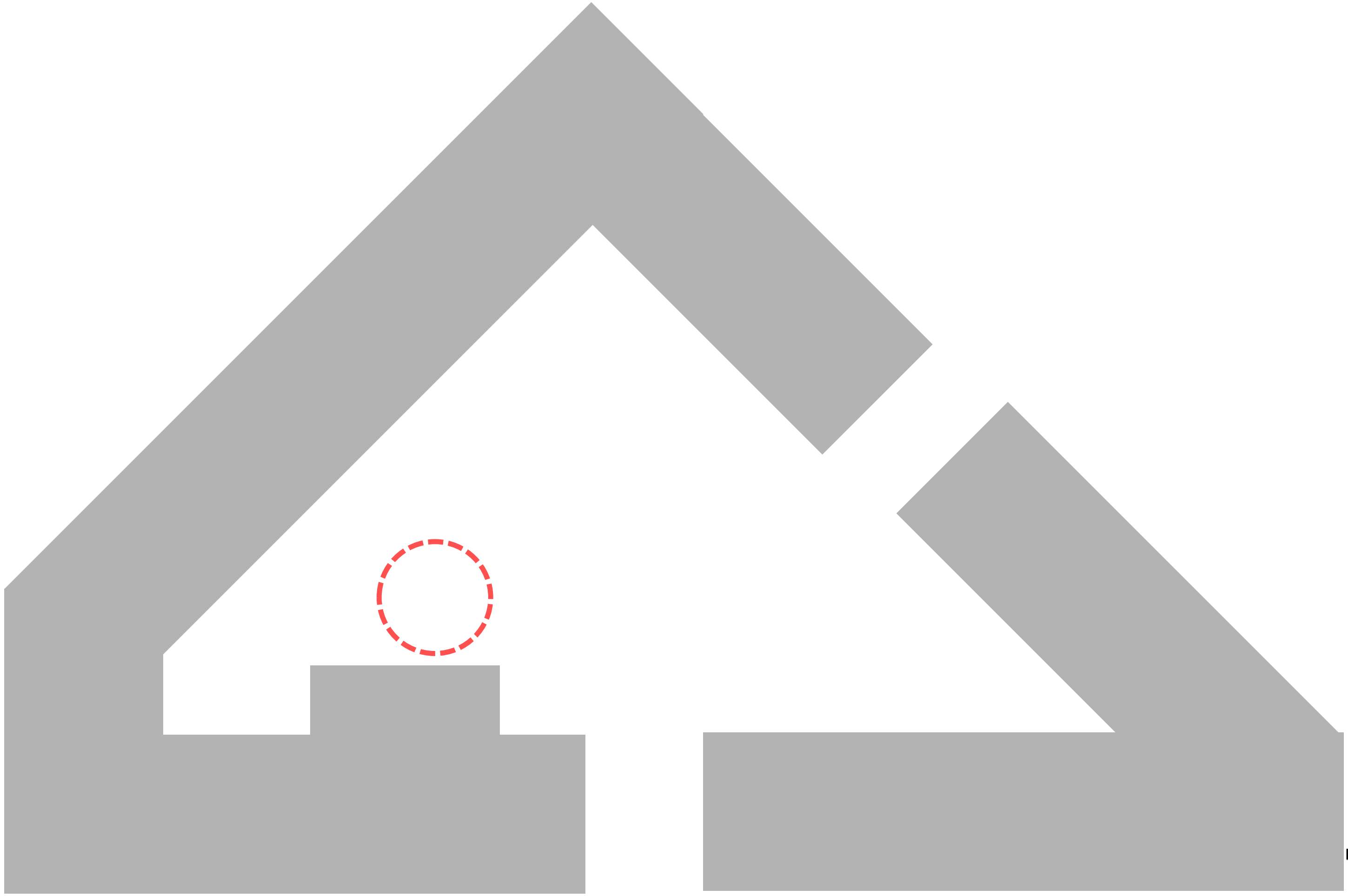
Encoding Visibility

RADIAL GAUSSIAN DEPTH



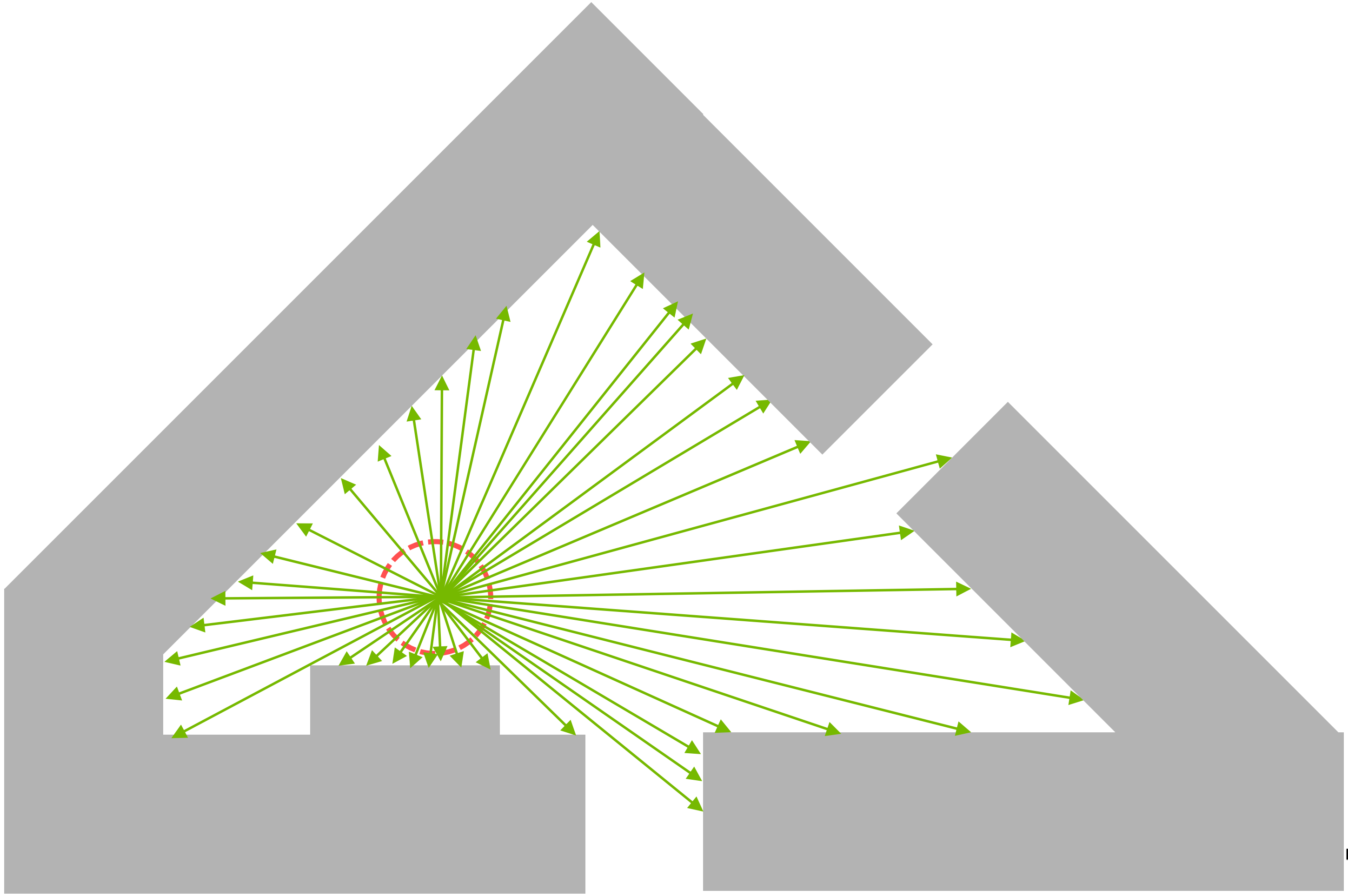
Encoding Visibility

RADIAL GAUSSIAN DEPTH



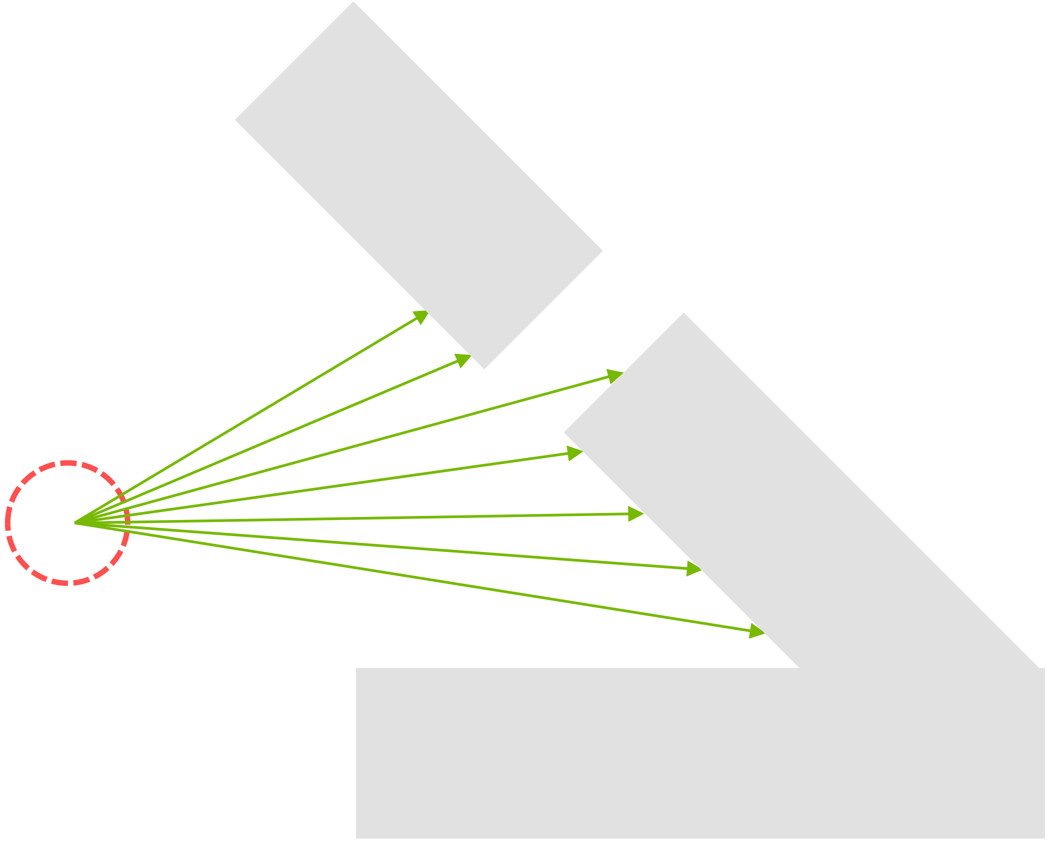
Encoding Visibility

RADIAL GAUSSIAN DEPTH



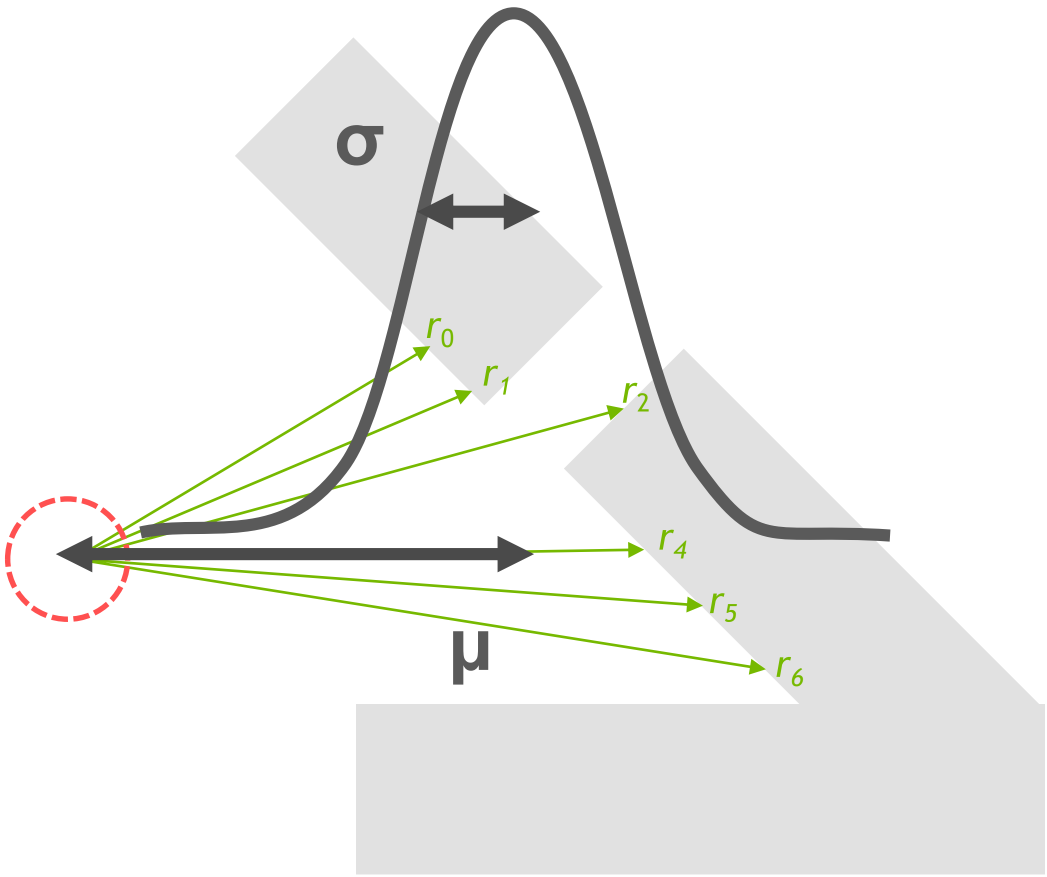
Encoding Visibility

RADIAL GAUSSIAN DEPTH



Encoding Visibility

RADIAL GAUSSIAN DEPTH



NVIDIA

Encoding Visibility

RADIAL GAUSSIAN DEPTH

Mean:

$\mu = \sum r / n$

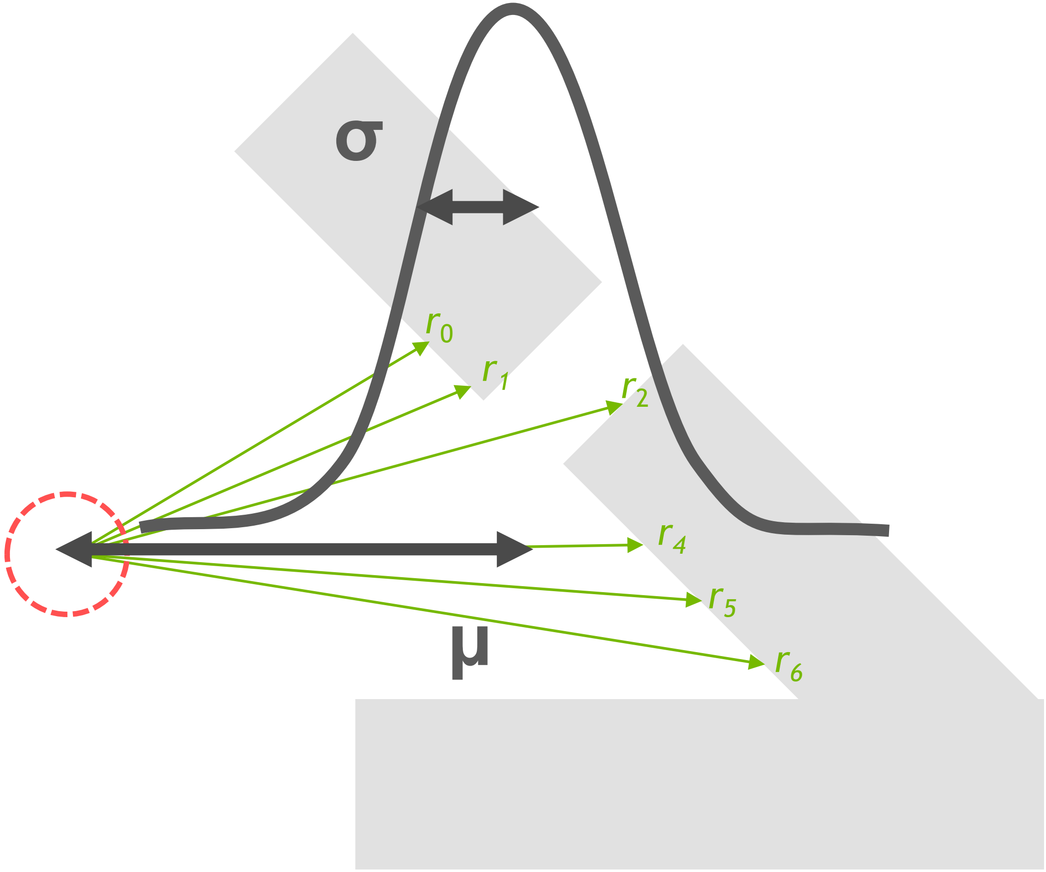
Standard Deviation:

$\sigma = \text{sqrt}(\sum(r - \mu)^2 / n)$
$= \text{sqrt}((\sum r^2) / n - \mu^2)$

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

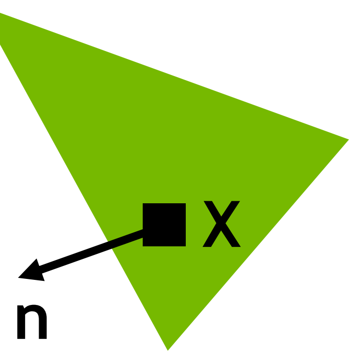
Irradiance blurs over a cosine-weighted hemisphere in a gather pass.

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



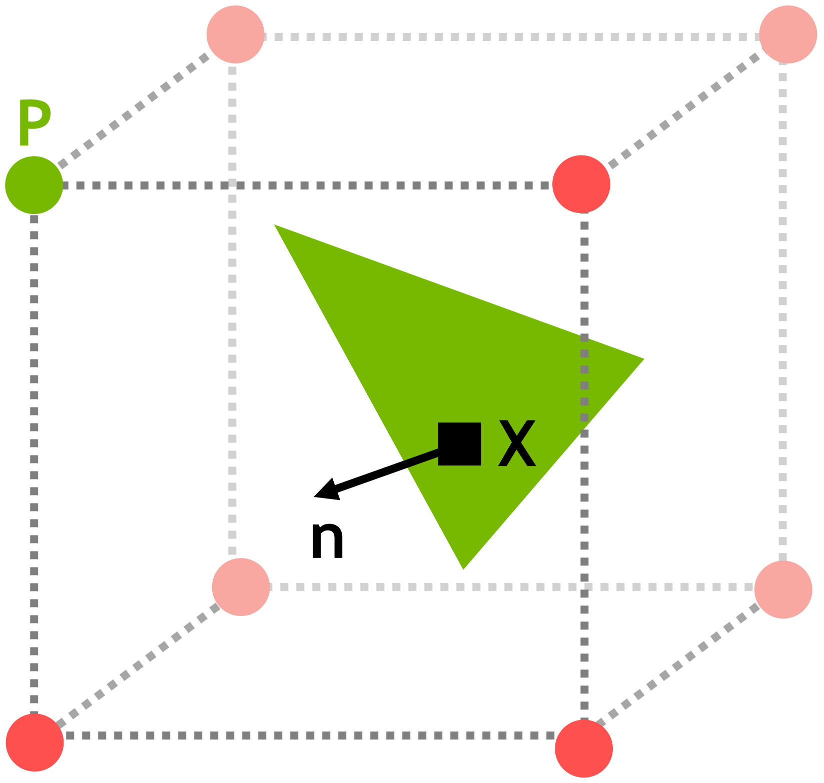
READING IRRADIANCE

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



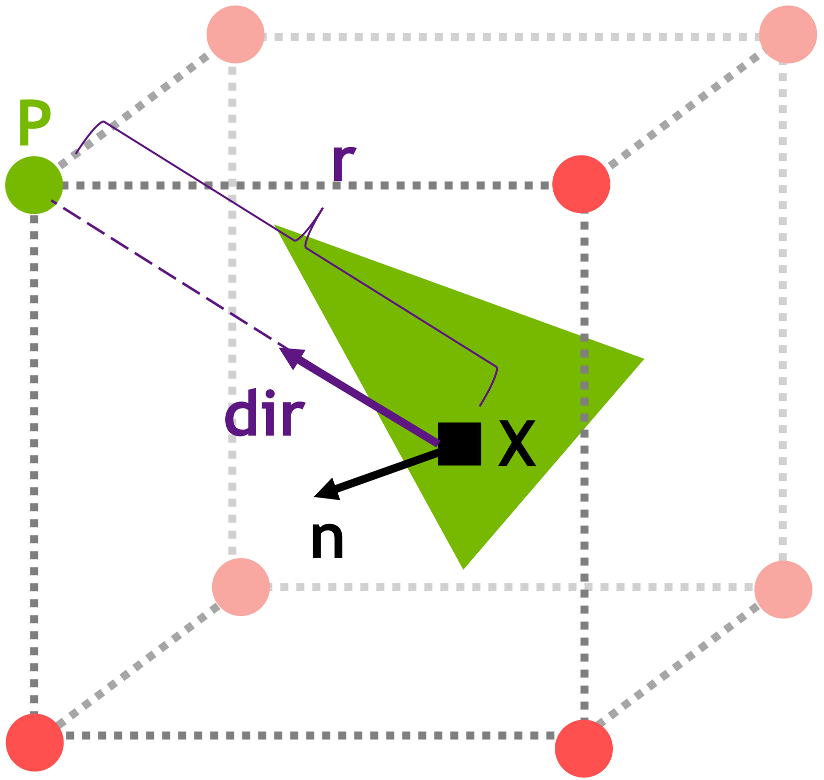
READING IRRADIANCE

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



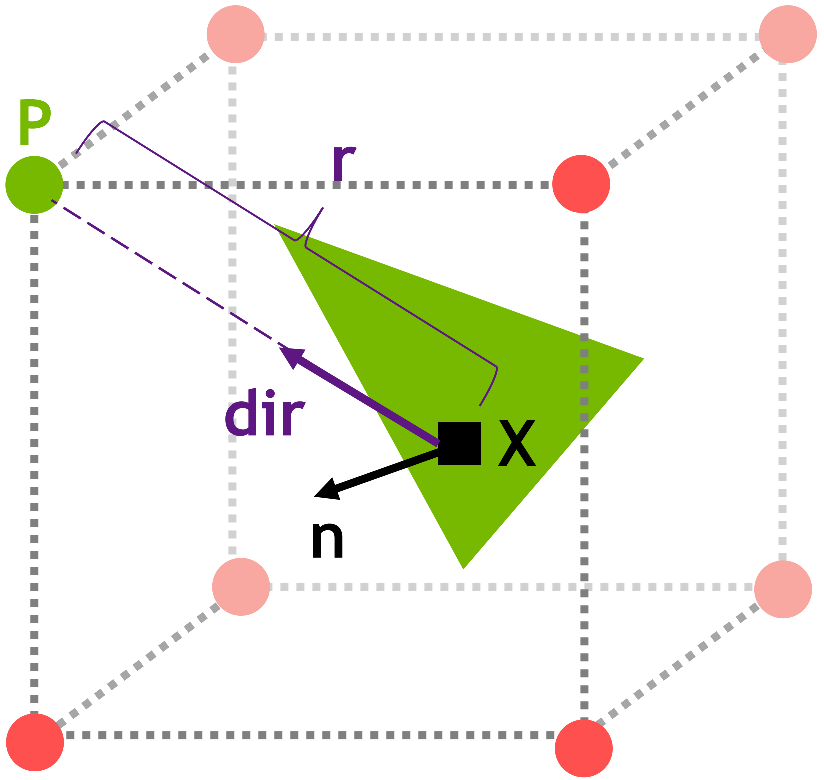
READING IRRADIANCE

```
// 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;
```



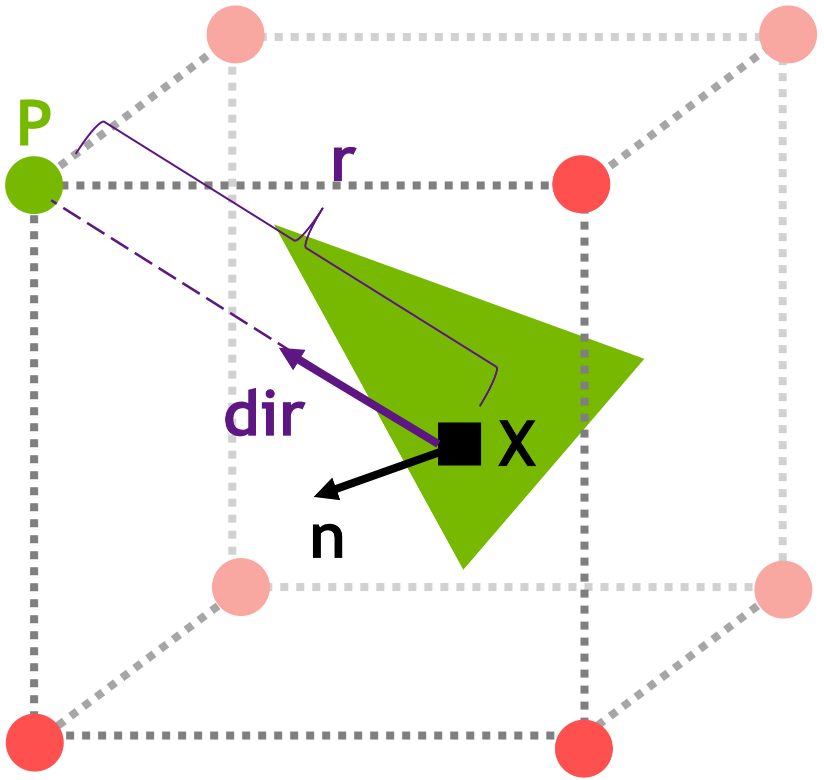
READING IRRADIANCE

```
// 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);
```



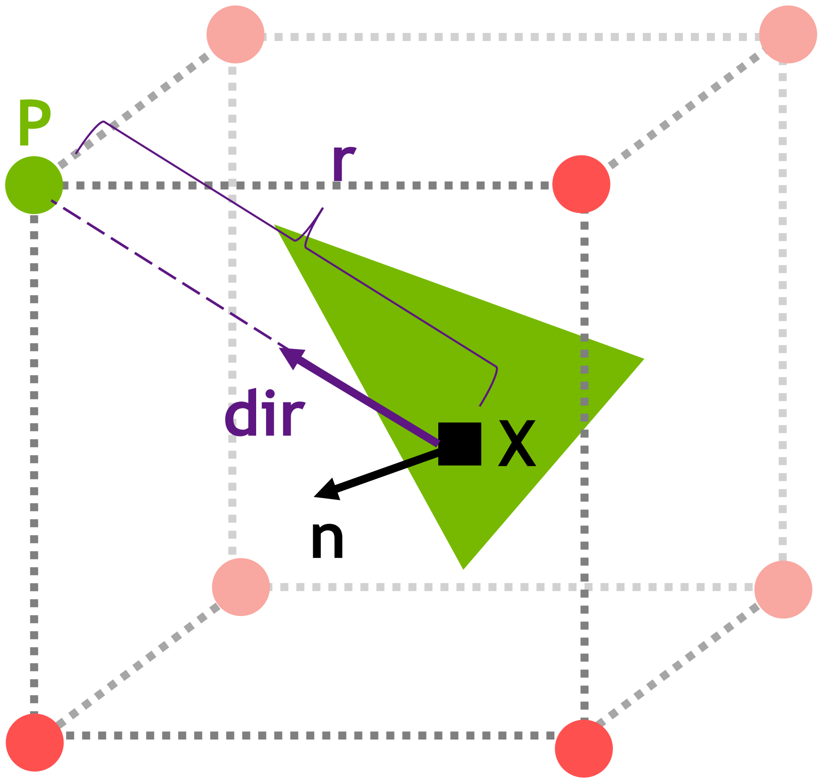
READING IRRADIANCE

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



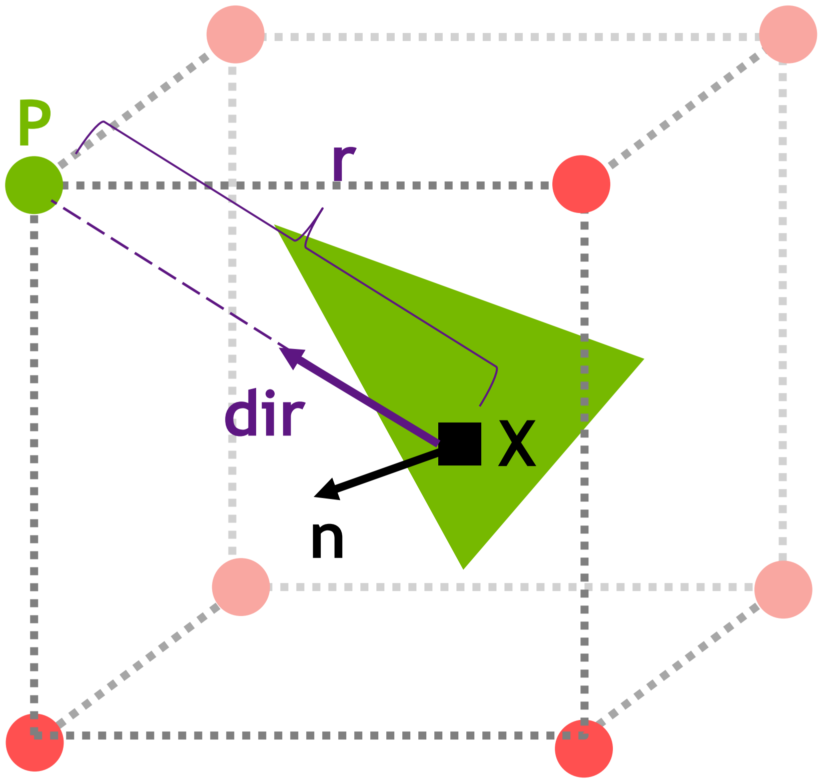
READING IRRADIANCE

```
// 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)
        float2 temp = texelFetch(depthTex, probeCoord).rg;
3       float mean = temp.r, mean2 = temp.g;
        if (r > mean)  {
            float variance = abs(square(mean) – mean2);
            weight *= variance / (variance + square(r – mean));
        }
        irradiance += sqrt(texelFetch(colorTex, probeCoord) * weight;
}

return square(irradiance.rgb * (1.0 / irradiance.a));
```

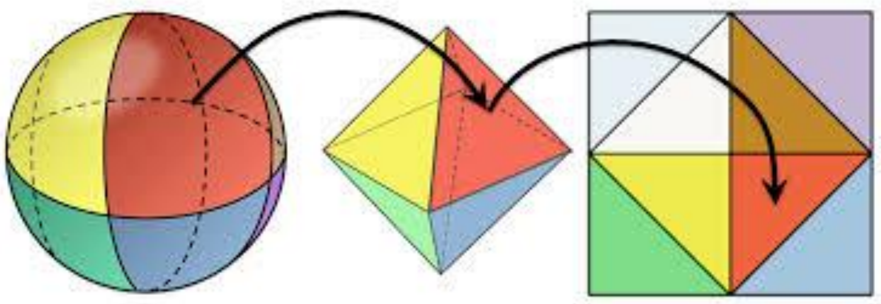


READING IRRADIANCE

```
// float3 n = shading normal, X = shading point, P = probe location
// threshold = low-perception threshold (around 0.2)
X += bias * (n - viewVector)
float4 irradiance = float4(0);
float4 irradianceNoCheb = float4(0);
for (each of 8 probes around X) {
        …

        // smooth backface
        float weight = square( (dot(dir, n) + 1) * 0.5 ) + 0.2;

        // adjacency
        weight *= trilinear(P, X) + 0.001;

        // visibility (Chebyshev)
        …
        if (weight < threshold)
           weight *= square(weight) / square(threshold);
        …
}

return lerp(square(irradianceNoCheb.rgb * (1.0 / irradianceNoCheb.a)),
            square(irradiance.rgb * (1.0 / irradiance.a)),
            saturate(irradiance.a));
```



PROBE PACKING



[Cigolle 2014]

Sphere → Octahedron → Square

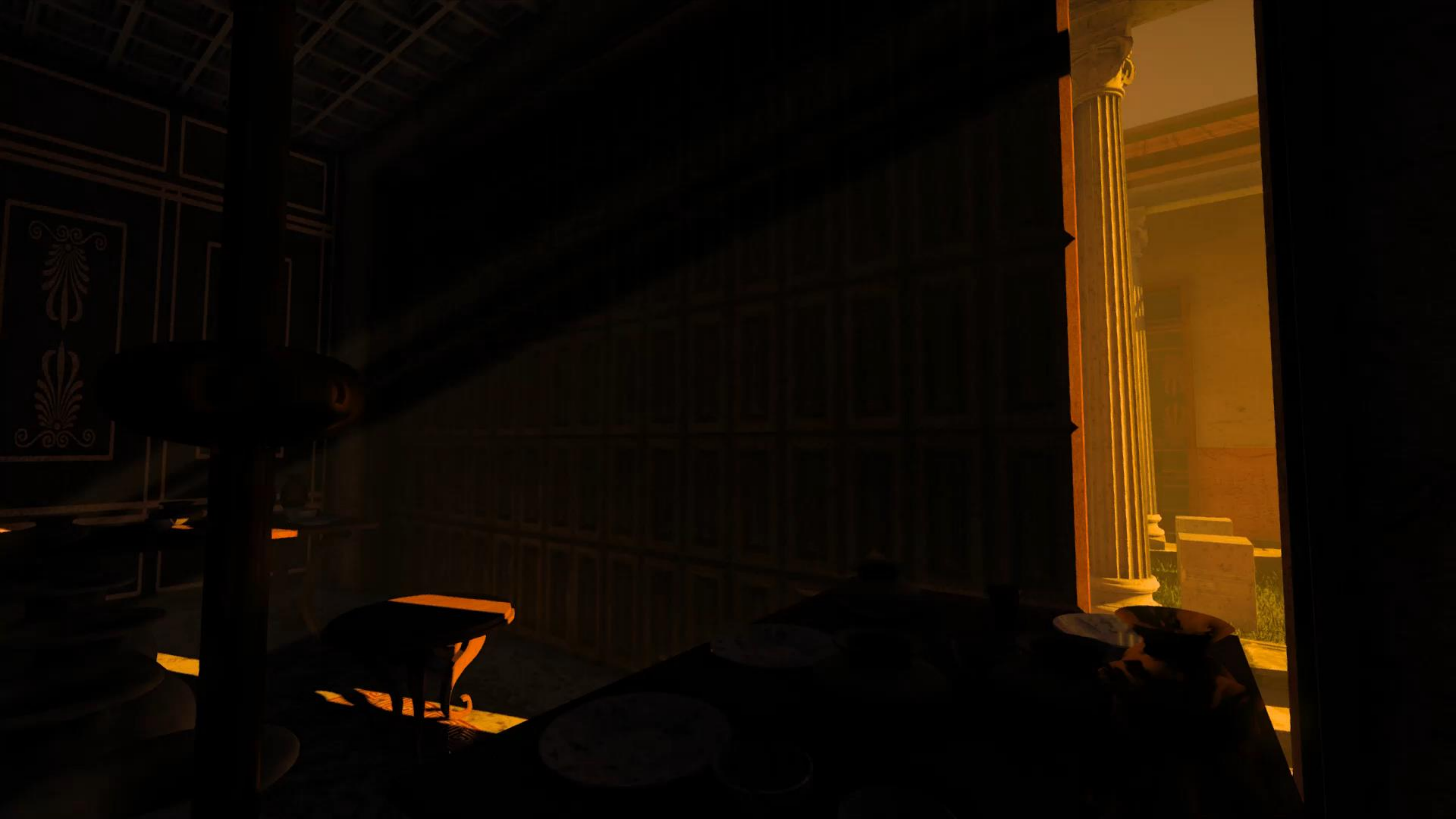
Pack XZ squares, layer in Y

Including border texels for fast bilinear:

Depth:	16x16	RG16F	= 1024 bytes
Irradiance:	6x6	R11G11B10F	= 144 bytes
			= **1168** bytes/probe

DEMO


Dynamic Diffuse
Global Illumination
NVIDIA
Render
Direct
Diffuse GI
Volumetric
Glossy GI
Materials
Visibility
Probes
Buffers
Time
5 AM
10 AM
1 PM
5 PM
8 PM
11 PM
Camera
Lobby
Ceiling
Hall
Lawn
Table
Arcade
Yard
Interact
Slide Door



MORE GI OPTIMIZATIONS

Scale down to lower-end hardware by reduce number of probes updated each frame or decreasing ray count and increasing hysteresis. Similar to what Enlighten did for their probes

Use fewer probes vertically for typical environments. Most illumination changes are due to walls. If you have two probes per room vertically that may be fine.

Order diffuse probe rays to capture coherence.

Pack diffuse probe rays and glossy into a single ray shader. Having more work in a single pass allows the scheduler to fill the machine and do more optimizations. This will get even faster over time with driver updates.

Compute ray derivatives for mip bias to reduce flicker and increase cache coherence. Force higher roughness on indirect bounces.

Clamp maximum radiance on indirect shading so that tiny reflections into light sources won't become fireflies.

Use fewer rays for diffuse in bright situations. Dim/high contrast is the case where rays get different results and you need more to avoid low-frequency flicker.

Use simplified shaders on indirect rays: no shadow map filtering, simpler BRDF model, no volumetrics.

Blurred mirror reflections are more stable than blurred stochastic reflections.

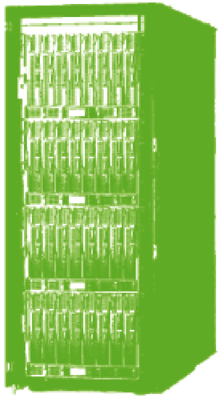
The Ultimate Optimization

STREAMING GI

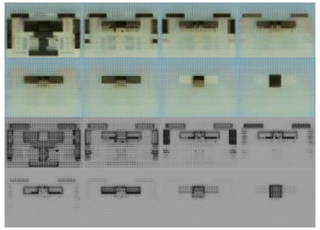
Diffuse GI + Multiplayer RTX Server

Desktop

or



Rack Server



H.265 DDGI Probe Texture →

5G

↔ *Proprietary Game Data*

Game Engine Clients

[Crassin et al. 2015]

SUMMARY

Dynamic Diffuse GI

Avoids leaks to decrease artist workload while increasing visual quality

1 ms/frame, 4.5 MB per cascade

Scales down to XboxOne by reducing update frequency

Scales up to 4k, 240 Hz, and VR

Dynamic lights and geometry, forward, deferred, transparent, volumetric

MORE INFORMATION

mcguire@nvidia.com, @CasualEffects

After the conference:

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

Technical Paper: **Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields,** Zander Majercik (NVIDIA), Jean-Philippe Guertin (University of Montreal), Derek Nowrouzezahrai (McGill University), and Morgan McGuire (NVIDIA), JCGT 2019

Thanks to Dylan Lacewell, Mike Mara, Dan Evangelakos, Sam Donow, and Corey Taylor for their work on the implementation infrastructure.

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

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

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

VR on the Exhibition Floor

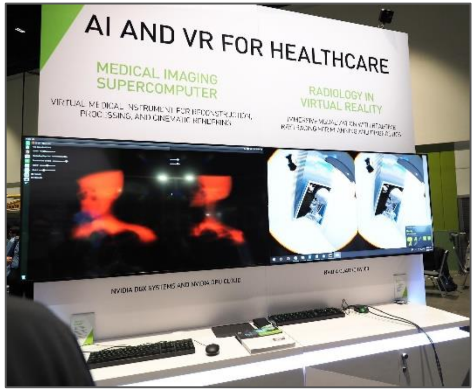
Expo Hall 3, Concourse Level

VR VILLAGE

Explore the VR Village to get hands-on with the latest advances in virtual reality

VR THEATER

Go to the VR Theater to see and experience narrated VR demos built by our partners



VR PARTNERS

Explore a great lineup of VR partners around the VR Village showcasing their groundbreaking technology

COME EXPLORE ALL THINGS VR AT GTC 2019

VR VILLAGE HOURS

Tuesday: 12:00pm - 7:00pm
Wednesday: 12:00pm - 7:00pm
Thursday: 11:00am - 2:00pm