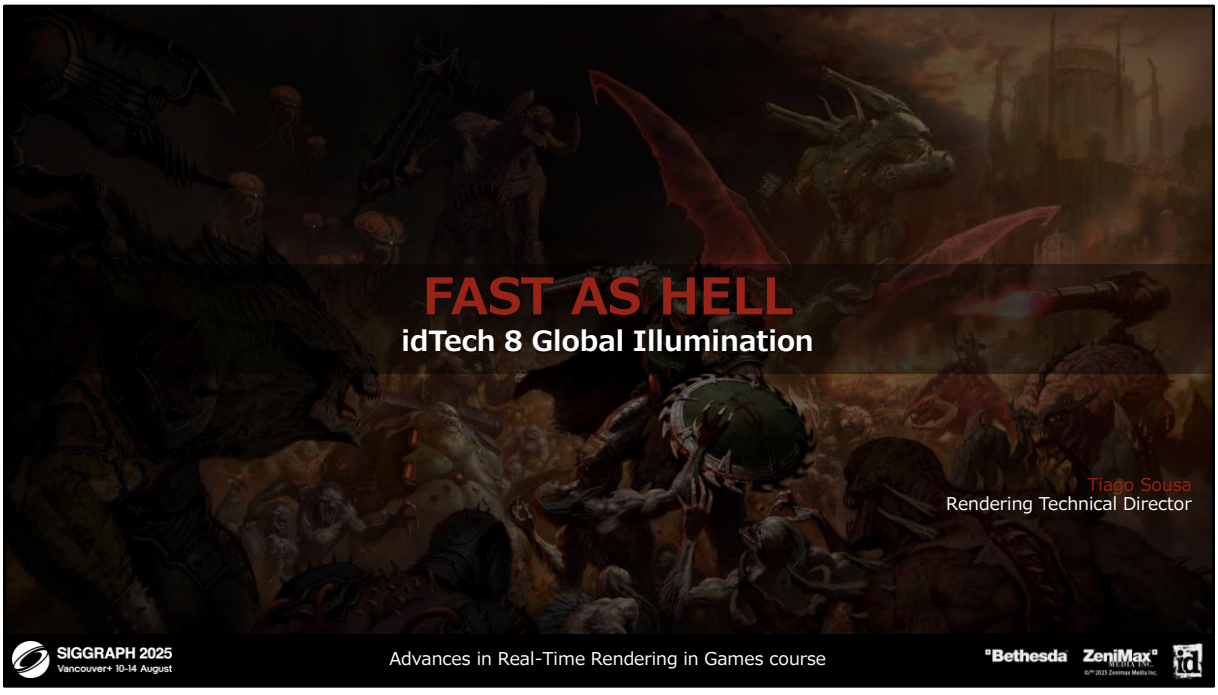
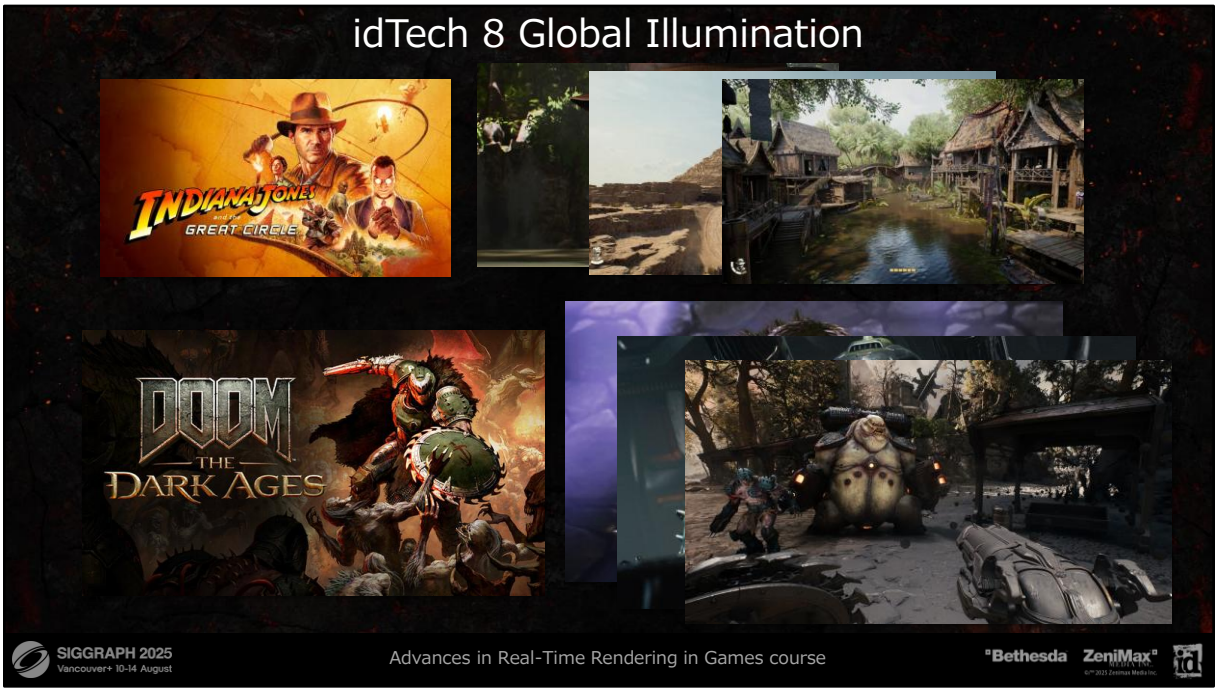

FAST AS HELL
idTech 8 Global Illumination
Tiago Sousa
Rendering Technical Director
SIGGRAPH 2025
Vancouver+ 10-14 August
Advances in Real-Time Rendering in Games course
Bethesda
ZeniMax
id




Hello everyone,

Today I'll be talking about the global illumination solution developed for idTech8.. This work has been used in 2 big projects running realttime at 60hz or higher on all platforms, and lays some of the foundations for next idTech iterations.

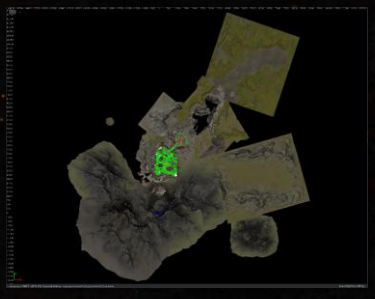
Indiana Jones and The Great Circle used an early iteration of the work we are presenting today. With the most recent iteration being used on DOOM The Dark Ages.

Both id and MG studios are relatively small ( < 200 ) for modern days crazy team sizes AAA standards. Particularly our tech teams are fairly compact, so we try share work whenever possible.


Low Spec = Xbox Series S and similar ( RT capable hw ).

## Larger levels: How much?

- DOOM The Dark Ages final statistics *
  - Approx. 5000 sq km for all levels summed together.
  - Comparing to DOOM Eternal:
    - 4x to 10x bigger levels
    - 22 single-player base levels (vs 13).



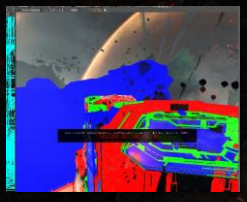
* Only accounting with walkable surface area



Single level example

## Past: Global Illumination on idTech 7

- Path-Traced Lightmaps
  - Static and opaque geometry.
- Multiple pre-processing steps
  - Unique UV unwraps.
  - BVH generation.
  - Lightmap LOD.
  - Etc.
- In-house cloud baking
  - For each machine
    - Process N instances lightmaps.
    - Dispatch N tiles (threads).
    - For each tile texel:
      - Raster tris, path-trace, update irradiance cache [Ward1988], etc.
      - HDR result, encode using 2-band SH [Ramamoorthi2001]
    - Stored 2 versions on network.
      - 1. BC6H L0 + 3 x BC1 L1 [O'Donnell 18] + Oodle Kraken.
      - 2. RGBE L0 + 3 x 32 bits L1 + Oodle Kraken.
      - Why? Different platforms encoding (no re-baking).
- Any changes? Re-bake required
  - E.g: lighting/geo/settings/etc.





Ask audience to raise hand, who here had fun shipping a complex/large multi-platform project using lightmaps ? Quick review/overview from id7

Static opaque geometry via Path-Traced Lightmaps. Dynamics/transparencies used another solution ( check extra slides ).

Multiple pre-processing steps
Lightmap lodding (e.g less resolution far, not visible from player walking areas = trim )

In house cloud “baking”:
Up to 64 machines.
2-band SH – for directional lighting results reconstruction later. Cheap / good enough. We tend to use in a lot of areas.
Switch ended up using ASTC + Kraken

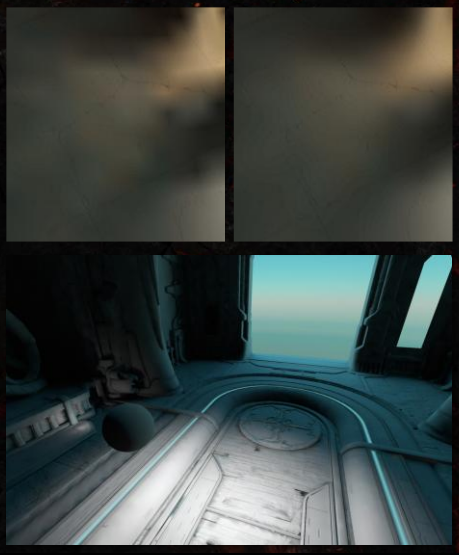
Any changes? Re-bake required.
E.g: lighting/geo/settings/etc..
Aggregate all tiles at end for final instances lightmap.

Re-bakes triggers

Lighting changes.
Quality settings changes. ( e.g. SSAA final bakes quality ).
Geometry changes.
Walkable player area changes require adjusted lodding.

## Past: Global Illumination on idTech 7

- Usual limitations
  - Leaks/seams/banding/resolution.
  - Block compression artefacts.
  - Workarounds
    - e.g. custom uvs/thick walls/decals/dithering/etc..
- Fairly optimized, but wasn't enough
  - Larger maps took 40h.
  - In the end dropped (½) x (½) res.
    - 2x2 texels per square meter default.
    - Art tunable for quality on key areas.
    - "Saved" 4x baking times / disk / network / memory.
  - Bi-cubic filtering for less egregious quality.
- Statistics for DOOM Eternal
  - Final quality bakes: >= 4h to 10h hours.
    - ~4 days turnaround time for all maps.
  - ~500MB compressed per level on disk
    - ~5GB uncompressed
  - Network storage: ~1.4TB





We spent quite some time optimizing, but even then we ended up having to drop to 1/2 resolution for more reasonable baking times + memory/disk/network requirements

## Past: Global Illumination on idTech 7

- Dynamic geometry + Transparencies TLDR
  - Irradiance Volumes (Indirect Diffuse) [Greger1998]
    - Generated from walkable player areas or hand placed.
    - Path traced, AFTER lightmaps are baked.
      - HDR + 2-band Spherical Harmonics encoding.
    - Per-vertex lookup via linear blend of 8 closest probes.
  - Reflection Probes (Indirect Specular) [Mittring2009]
    - Box Projected Cube Map Env. Mapping [Czuba2010]
    - Hand placed samples by art.
    - Generated AFTER lightmaps are baked.
    - BC6H cube map array.
    - Applied via cluster/tile binning [Olson2012][Sousa2016][Geffroy2020]
  - Re-baking triggers similar to lightmap.



Irradiance Volumes ( for Indirect Diffuse Lighting. We call it Ambient Probes internally )

Much quicker that lightmaps, but still around 10 minutes baking time usually per level, on a single machine.

Reflection Probes (for Indirect Specular. We call it Environment Probes internally)

# Past: Global Illumination on idTech 7

- Multiple systems and dependencies
  - ⇔ Could break at multiple points
    - Lightmap instance/probe looking broken/out of place?
      - Mesh changed?
      - Level layout change?
      - Lightmap LOD change?
      - Lighting change?
      - Code bug? Nans?
    - Lightmap work didn't finish on a cloud machine?
      - Crash during bake?
      - Windows Update running?
      - Anti-Virus running?
      - Out of disk space?

## Past: Global Illumination on idTech 7

- What if DOOM The Dark Ages used the exact same older tech bits?
  - Rough estimate:
    - Minimum pre-computation time per level:
      - 4 x bigger levels: ~ 4h to 10h x 4 = ~ 16h to 40h.
      - 4 x ~ 500 MB = ~ 2 GB compressed.
    - Maximum time:
      - 10 x bigger levels: ~ 4h to 10h x 10 = ~ 40h to 100h
      - 10 x ~ 500 MBs = ~ 5 GBs compressed.
    - Almost double the number of levels (~1.7x):
      - Disk requirement:
        - (min) ~ 44 GBs
        - (max) ~ 110 GBs
      - Network Storage:
        - (min) ~ 4 x 1.4TBs = ~ 9.5 TBs
        - (max) ~ 10 x 1.4TBs = ~ 24 TBs
      - Turnaround time:
        - (min) ~ 4 days = ~ 27 days
        - (max) ~ 4 days = ~ 68 days




What if we had just used the exact same idTech7 pre-computation steps on the new project ?

Clearly not a scaleable solution. Even if levels only had increased the minimum size of 4x. Would have required restraining art a fair bit / limit levels design.

And would have kept being a pain point for everyone involved, especially at end of a project where quick iteration is key.

How to scale ? Even if we would double the number of PCs on the cloud. And even if they would have double the core count - would still be slow. Doesn't fix any of the other cases. Keep investing on older tech? Or move on?

## Building on Previous idTech 7 Work

- DOOM Eternal/idTech7 RT update
  - June 2021.
- Hardware RT
  - Almost a decade old.
    - Support on all new generation consoles.
  - We view it as a visibility tool.
    - Similarly to compute shaders, can be creative.
    - idTech8 used for gameplay/other use cases.
      . Material queries (e.g. sound/decals), particle collisions, streaming, mesh spawning, tools.
- Build on our work from idTech 7
  - Try unify/simplify previous systems.
  - Can't trace many rays..
  - How to speed up workloads?
  - Can we get close to Path Tracing results?
- Extensive ongoing practical research
  - DDGI [Majercik2019]
  - Surfels [Halen2021]
  - Screen-Space Radiance Caching [Wright2021]
  - ReSTIR GI [Ouyang2021]
  - Etc



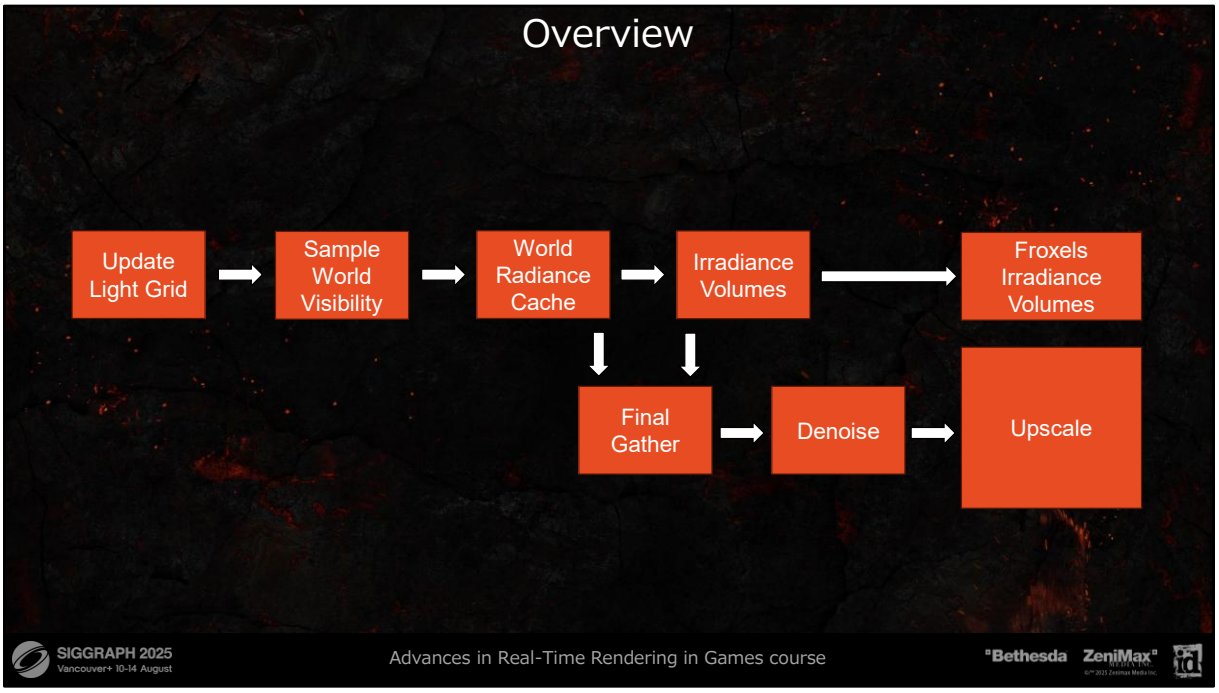





Which brings us to id8. Where iteration speed is key. Can we unify/simplify previous systems ?

We finished RT support for DOOM Eternal around June 2021 and was a good learning experience about “next-gen” ( basically current gen now ) platforms limitations. Particularly obvious: Can't trace many rays. ( low end hw can be orders of magnitude slower than faster hw ).

Back in 2021, when we started there was quite some ongoing research that looked practical (⇔ potentially usable across multiple platforms).




This is an overview of what we ended up with. ( For global illumination this is essentially what happens in a frame ).

1. We start by updating structures required, for lighting/shading things.

2. Sample world visibility, cache samples results.

3. Shade all the previous samples and store them in the “world radiance cache”

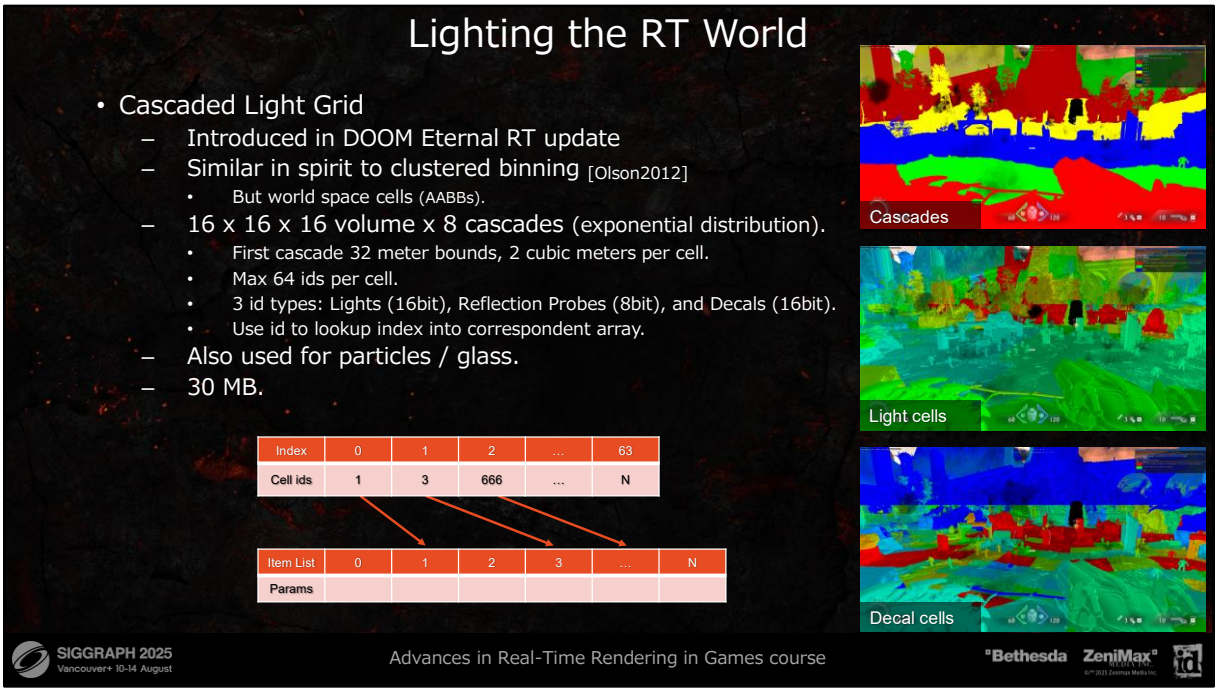
3. Using past result, update the irradiance volumes.

4 Do a final gather step, where we index into all the caches ( 0 shading involved )

5. Denoise result.

6. Last but not least, upscale the result.

For transparencies, we generate a “transparencies froxel irradiance volume”




First things first. How do we light the RT world. Can't use tile/cluster binning. These are tied to frustum.

Still a fan of cluster binning, can we do it in world space ? Introduced Cascade Light Grids for DOOM Eternal RT update.

Cost around 30 MB Vram . Most of it from having ludicrous amount of decals around camera (limited to 30k) and input arrays having to be double buffered.

We also ended up also using it for particles/glass. For example on particles using clustered binning when they are lit per vertex, could result in artefacts from vertices outside frustum.

# Lighting the RT World

- 1x Async compute dispatch
  - X, Y, (Z x Num Cascades)
    - XYZ = grid resolution / 4.
    - 4 x 4 x 4 thread group size
- Hierarchical Gather
  - CPU markup (bitmask) for items AABBs fully contained in a cascade.
    - ⇔ Don't process items not present in a cascade.
  - 1. Coarse culling
    - Each thread = world cell from cascade.
    - Cell size = bounds / thread group size
    - Gathers lights/decals/probes overlapping cell.
    - Item OBB vs cell AABB test.
      - Transform AABB cell into OBB local space.
      - Check for AABB cell vs identity AABB.
    - Gather results into group shared memory, flat bit array buckets [Drobot2017]
      - E.g. groupshared uint lightsFlatBitArray[ 32 ];
      - One bucket per item type
      - Flat bit array is ordered ( important for decals/reflection probes pre-sorted results).
  - 2. Detailed culling
    - Cell size = bounds / grid resolution
    - Loop through flat bit array buckets + item OBB vs cell AABB.
    - Write final buffers.
  - ~0.2ms consoles (~0.1ms on PC)



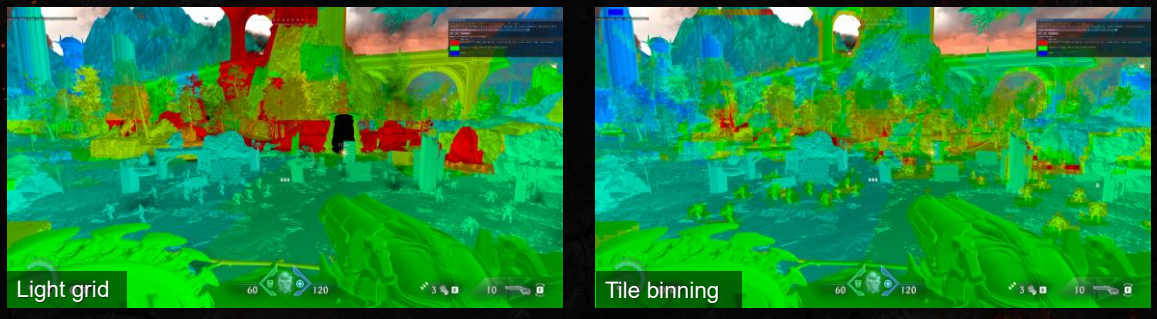

Limits
4k lights (retail) / 8k lights ( dev )
30k decals (retail) / 54k decals ( dev )

(Drobot2017) Improved Culling For Tiled And Clustered Rendering

Fairly fast for amount of data It processes, but could be improved. There was bigger battles.

## Lighting the RT World

- Similar limitation to clustered binning
  - Increased overdraw at distance.
  - Can mitigate with higher cascades resolution.






Similar limitation to clustered binning: the further cascades the more overdraw due larger volume size.

Can mitigate with higher density cascades resolution. For us the 16 x 16 x 16 volumes were a sweet spot for memory/perf.

Note: on right image/ on raster side characters use what we call “light rigs”, an art/design choice to keep a character always visible. We skip those on RT world.

## Sampling the World

- Cascaded Irradiance Volumes + Local Irradiance Volumes
  - 16 x 16 x 16 x 6 cascades. 10 x 10 x 10 for local volumes.
  - Sample Position = Cells center.
  - Interleaved updates.
    - Per frame: 1 cascade + 1 volume.
  - Begin by tracing only visibility.
    - Each probe traces N visibility ray queries.
    - Per frame: 64 / 32 / 16 rays.





Cascaded irradiance volumes + hand placed local volumes.

Only visibility at start.

16 x 16 x 16 x 6 cascades. 10 x 10 x 10 for local volumes.

lower specs cascades would be, 12 x 12 x 12.

Interleaved updates

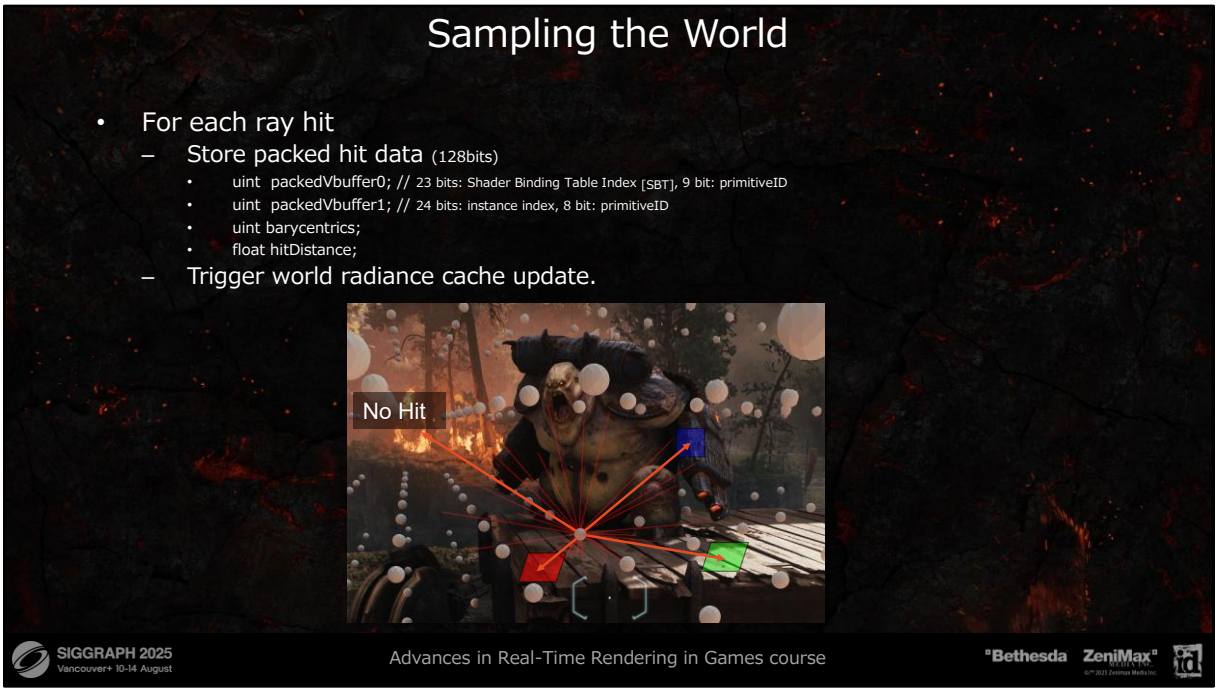
Per frame: 1 cascade + 1 volume.

Each probe in volume traces N visibility rays, per frame.

64 / 32 / 16 rays on depending on platform/settings.

RayOrigin = Probe Position; // The cell center.

RayEnd = RayOrigin + Ray Dir * MaxDist;




For each ray hit:

Store 128bits of packed data.

uint packedVbuffer0; // ( 23 bits: Shader Binding Table Index [check SBT ref ] ), ( 9 bit: primitiveID ). ( part 1 of 17 bit primitive ID )

uint packedVbuffer1; // ( 24 bits: instance index ), ( 8 bit: primitiveID ) ( part 2 of 17 bit primitive ID )

uint barycentrics;

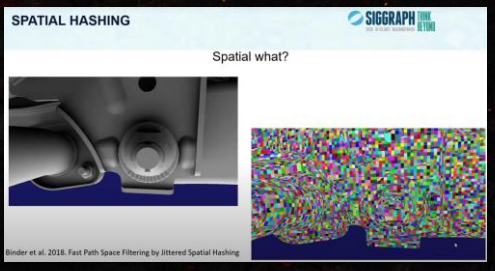
float hitDistance;

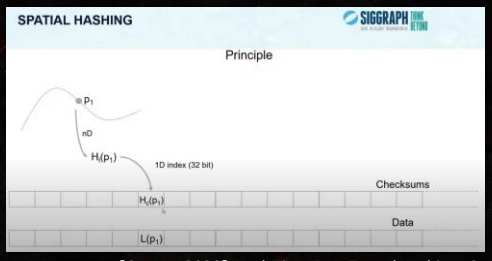
Trigger world radiance cache update

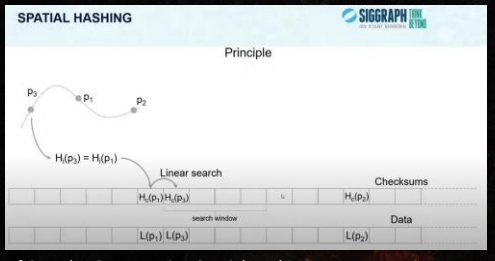
More ahead

## Sampling the World

- Each ray hit triggers cache update
  - Cache uses spatial hashing [Gautron2020]
    - 1D array.
    - N-Dimensional hash function.
      - Use hash value to index array.
    - 25cm$^3$ cells quantization + cells LODs.
    - 14 MB.




*[Gautron2020] Real-Time Ray-Traced Ambient Occlusion of Complex Scenes using Spatial Hashing*



For cache, spatial hashing [Gauttron2020] caught our attention early on due to it's simplicity ( always a plus in my book ).

Allows compact storage of data (for our case, volumetric data) in a 1D Array.

To index, compute hash function over multiple components (e.g. quantized positions, distance to camera for lodding, temporal, normal, etc. Dealer's choice ). Use the hash value to index.

Hash collisions will happen at times, those are handled by doing a linear search over next N elements. (Fingers crossed we fit, else we have to drop some results ).

# Sampling the World

- Each ray hit triggers cache update
  - Atomic increment active frame cells counter.
    - One counter and one list per dispatch index.
      - Dispatch index ⇔ shader index. Retrieved from SBT.
    - Store active cell hash index on last item.
    - Also store rayID and probeID at hash index.
      - In next steps, use to fetch packed hit data.


```
uint lod = uint( exp2( floor( log2( 1.0f + ( distToCam / RAD_CACHE_LOD_DIST ) ) ) ) );
float cellSize = RAD_CACHE_CELL_SIZE * lod;
int3 posInt = int3( floor( pos.xyz / cellSize ) );
uint hash = Hash( lod + Hash( posInt.z + Hash( posInt.y + Hash( posInt.x ) ) ) );
uint hashIndex =  hash % RAD_CACHE_MAX_NUM_CELLS;

atomicAddOut( radCacheActiveCellsRW[ dispatchIndex ], uint( 1 ), numActiveCells );
radCacheActiveCellsRW[ dispatchOffset + numActiveCells ] = hashIndex;
radCacheStateRW[ hashIndex ] = packedRayData;
```



Example code for inspiration. Check Gautron 2020 for in depth explanation of spatial hashing/dealing with hash collisions.

## World Radiance Cache Update

- Shade each active cache entry
  - Use active frame cell counters to setup indirect dispatches.
    - Async Indirect dispatch per shader.
  - Use rayID + probeID + ray hit packed data to fetch required inputs.
  - Mostly same code for lighting/shading loop as raster.
    - Previous frame irradiance volumes for multiple bounces.
  - ~20k cache entries updated per frame. Re-use for N frames.




We then shade each active cache entry

Using async indirect dispatch per each shader type. Not that many shaders ( less than 10 )

From probeID and rayID can derive global rayID. Lookup ray hit packed data (SBT, etc -> triangle data -> use barycentrics, etc )

Shading/lighting: access light grid/bindless resources, use inputs computed from ray hit packet data. Do light & shading loop.

## Irradiance Volumes Update

1x Compute Dispatch

- Only process current frame volumes.
  - 8x8 thread group size ⇔ up to 64 rayIDs.
- Load world radiance cache results
  - Index via probeID and rayID.
    - No hit ? Lookup sky.
    - Store into group shared memory.
  - Compute irradiance for each texel [Greger1998]
  - Add previous frame.
- Store into probe image atlas
  - Octahedron env mapping [Dachsbacher2008]
  - Similar storage to DDGI [Majercik2019]
    - RGB9E5 color + RG16F visibility.
  - 2560 x 1585.
  - 8x8 tiles (10 x 10 with borders)
  - 64 MB (30MB lower specs)
  - 0.08ms

Probe Image Atlas

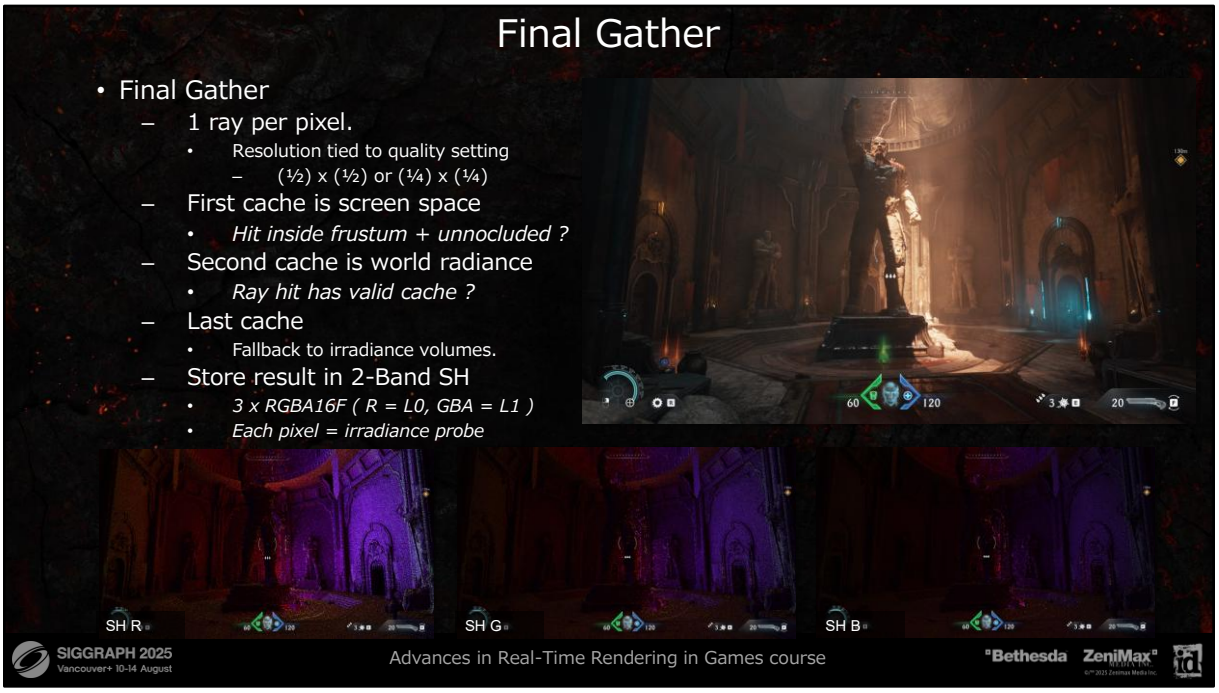
Irradiance Volumes



With the radiance cache updated. Can now compute irradiance volumes.
Load radiance cache results
8x8 tiles. Nicely fits into 64 thread group size.

From probeID and rayID can derive global rayID. Intersection point = probe center + rayDir * rayHitDistance. Compute hash, lookup radiance cache.

Octahedron env mapping, conveniently maps into one image tile => simple lookup / memory efficient.
2560 X 1585 for 6 cascades + max of 100 local volumes.




Final Gather

1 ray per pixel. Cosine weighted ray directions ( along hemisphere defined by pixel normal ), use blue noise for ray direction variation.

Resolution depending on quality settings

½ or ¼ resolution

0 shading, only use caches.

First cache is screen space

*If ray hit is inside frustum and unnocluded.*

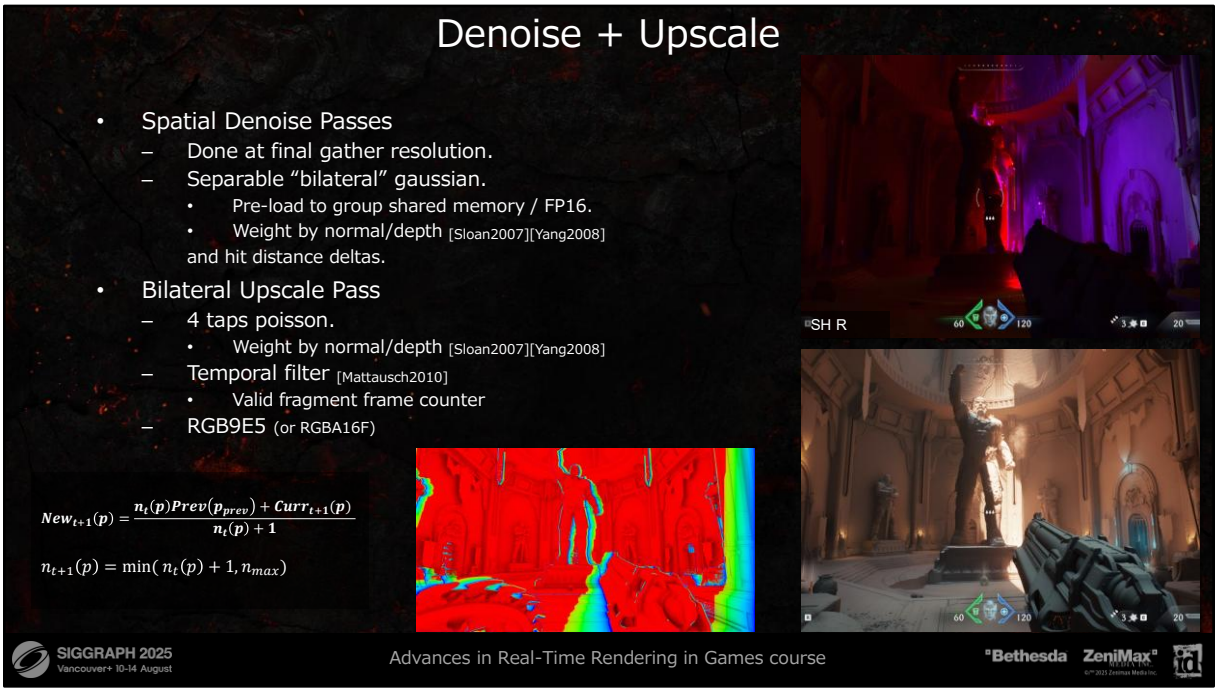
Second cache is world radiance.

*If ray hits valid cache.*

Last cache, fallback to irradiance volumes.

Store 2-Band SH, want to try maintain normal maps details (similarly to lightmaps).

Essentially each pixel is an irradiance probe in SH space.




Spatial Denoise passes

“Denoise” is a fancy word for sharing similar neighbors results to mitigate for lack of samples.

Done at same resolution as final gather.

Separable “bilateral” gaussian.

Weight by normal/depth and ray hit distance deltas.

Not really separable, but good for perf.

Group shared memory for performance.

Upscale

*4 taps poisson.*

*Weight by normal/depth.*

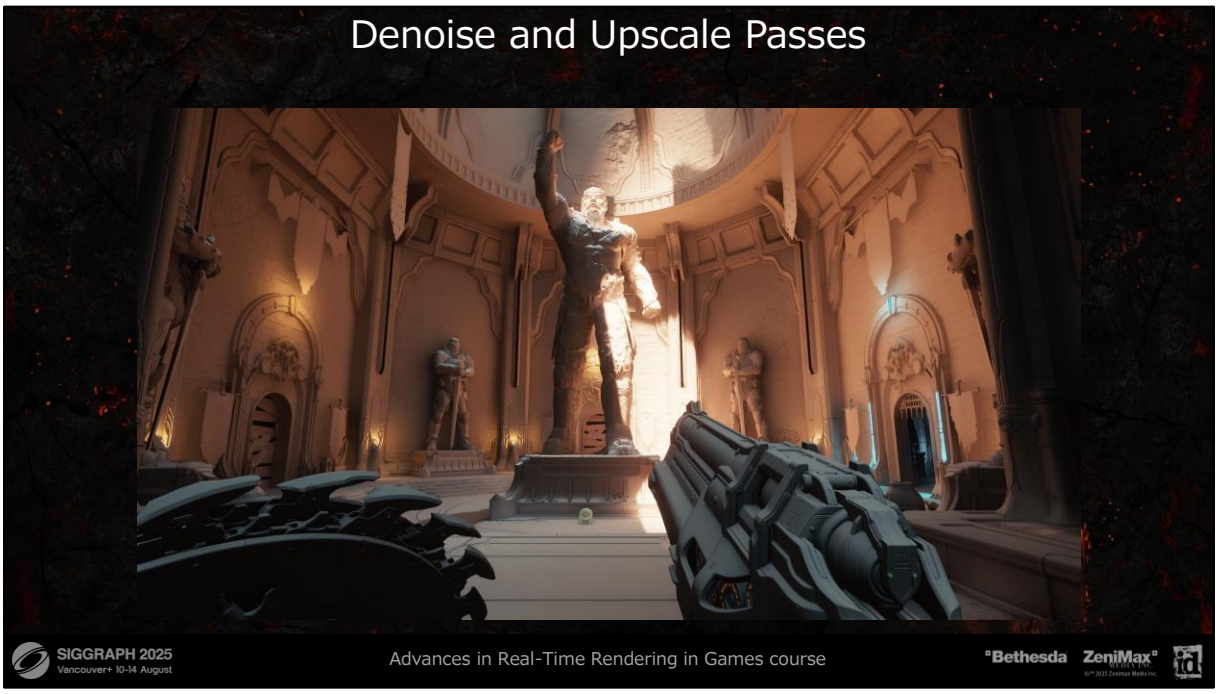
*Temporal filter.*

*RGB9E5 on platforms that fully support*

*Else RGBA16F*

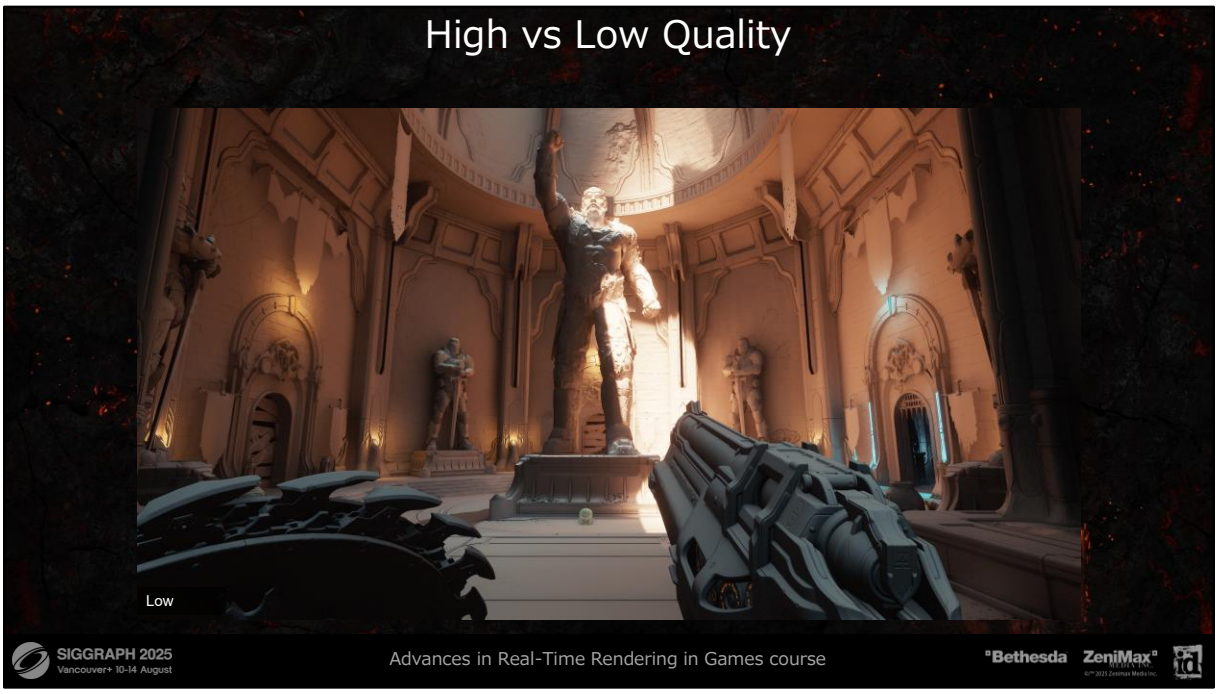
*Avoid R11G11B10F ⇔ color shift/banding, no pure white.*

TAA/others acts as another temporal filter at end of frame.




Visualizing individual steps.

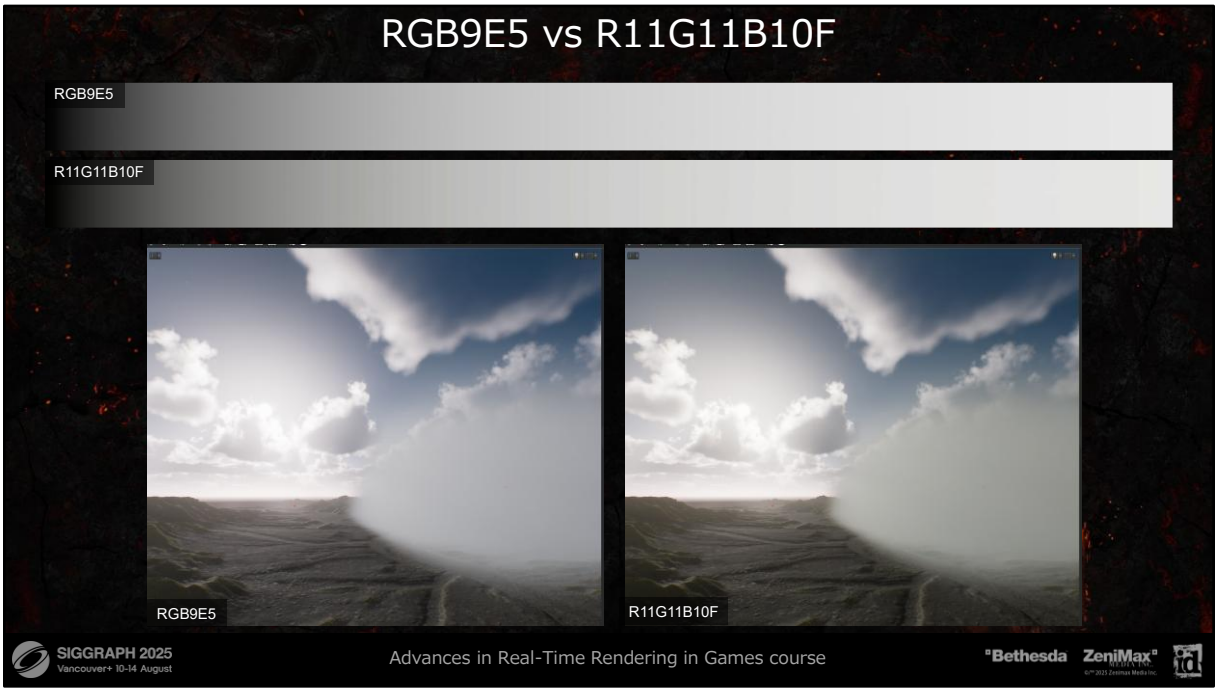
1. No filtering
2. Denoise passes
3. Temporal Filter
4. Adding direct lighting




Visualizing quality settings

1. High
2. Low

Relatively close, bit softer result on low quality. Loses some directional lighting details due lower resolutions.




Pause for nitpick: Can every HW Fully support RGB9E5 in future please. We want lean nice quality 32 bit formats. (Plus memory is not growing that much unfortunately.. )
Notice the green/yellow shift/no pure white. Some banding as well ( can be hidden with dither, but )


Reflection Probes

No need to wait for lightmap baking. Just do it.

Similar to what we've been doing for past decade (slide 7), BC6H cube map array applied at runtime, depending on surface time, opaque will use tile binning.

Transparencies either clustered or light grid.

Dropped AMDs Cubemap Gen

Re-implemented for baking perf (GGX filtering, multi-threaded (tile based))

Re-fit reflection probes result [Lazarov2013][Hobson 2019].

Fit results to environment. nicer localized details ( e.g. no glowing corners, pick up bounce color ).

On our case, we use the frame irradiance as a replacement term for the probe irradiance.

## Indirect Specular

- Almost same as in idTech7
  - SSR when possible then RT.
    - Fallback to probes based on smoothness.
  - RT uses own temporal upscale.
    - Upscale depends on quality settings
      - ½ res, ¼ res
    - Ended up disabling on consoles for perf.
      - Usually fairly fast (~2ms).
      - Ran out of time to optimize further.
  - Thresholded > min smoothness
    - Only runs on very smooth surfaces.
    - Ran out of time for more complex BRDFs.



Ran out of time to optimize further.

SSR + Probes was always fallback plan.

## Transparencies

- Froxels Irradiance Volume
  - Volumetric fog [Wronski2014] uses irradiance volumes.
  - Share previous work for all transparencies.
    - ⇔ No need to re-do lookup work for transparencies.
    - Stored during light scattering froxel dispatch.
  - 2 band SH encoded.
  - 50MB (25MB on lower specs)
  - Used in particles, glass, etc.

Re-fit environment probes on blended



Volumetric fog based on Bart Wronski 2014 work.

Re-fit: use Froxels Irradiance Volume as a replacement for probe irradiance.

## Compositing Results

- Directional Occlusion [Landis2002][Ritschel2009]
  - Used for high frequency details.
    - Don’t have all data in BLASes.
    - POM [Tatarchuk2005]
  - Contact shadows for every light.
  - Specular occlusion.
  - Ray-march along hemisphere
    - Store average unnocluded direction.
    - Denoise + temporal filter [Mattausch2010]



For compositing we rely on directional occlusion. This bring back some high frequency details that we miss, due to not having every geometry/upper lod details on BLASes.

(Mattausch2010) High-Quality Screen Space Ambient Occlusion using Temporal Coherence


on/off on paralax occlusion mapped surface.

## Hotspot Performance

| | Xbox Series S (900p) | Xbox Series X (1440p) | PS5 (1440p) | PS5 Pro (1800p) | PC (4k) |
|---|---|---|---|---|---|
| World Sampling | 0.38 ms | 0.27 ms | 0.45 ms | 0.083 ms | 0.091 ms |
| Radiance Cache | 0.2 ms | 0.21 ms | 0.109 ms | 0.11 ms | 0.109 ms |
| Irradiance Volumes | 0.08 ms | 0.08 ms | 0.046 ms | 0.035 ms | 0.07 ms |
| Final Gather | 0.6 ms | 0.535 ms | 0.489 ms | 0.385 ms | 0.5 ms |
| Denoise | 0.208 ms | 0.18 ms | 0.266 ms | 0.22 ms | 0.52 ms |
| Upscale | 0.65 ms | 0.59 ms | 0.69 ms | 0.72 ms | 0.445 ms |
| Cost ( serial ) | ~2.11 ms | ~1.9 ms | ~2.05 ms | ~1.923 ms | ~1.7 ms |
| Cost ( async ) | ~1.71 ms | ~1.4 ms | ~1.55ms | ~1.82 ms | NA |

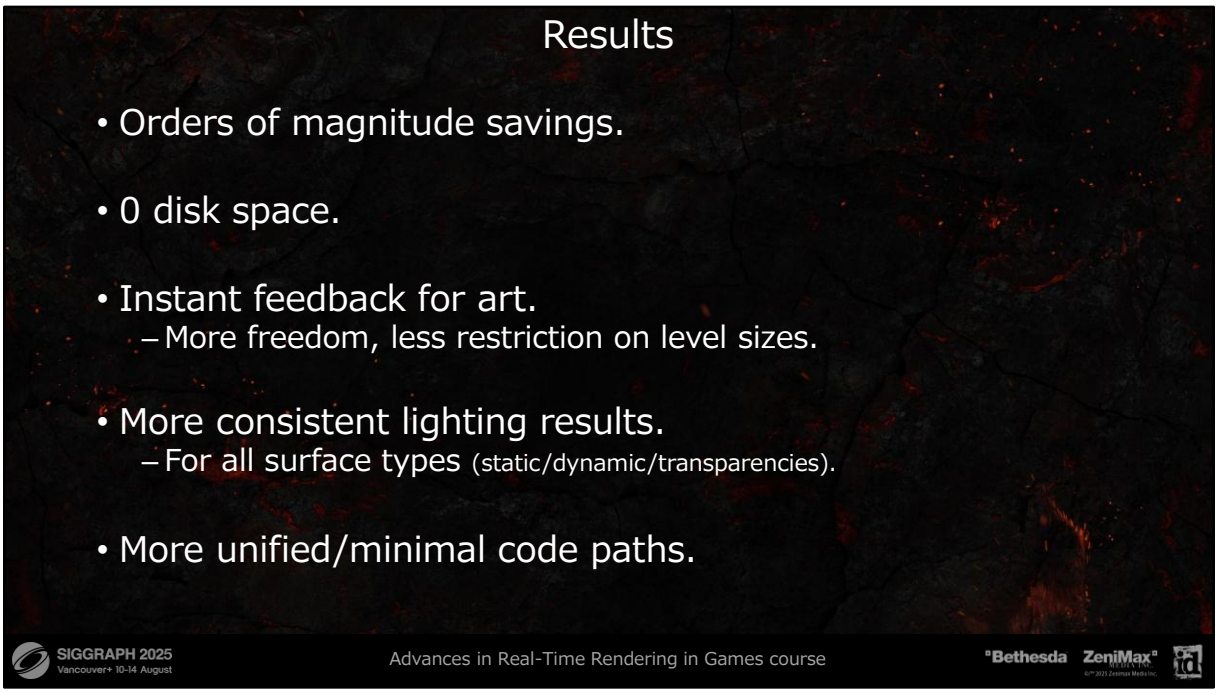


Some numbers, on a common hotspot we have in the game ( mission 4, essentially a large vista, lots of enemies, vegetation, particles, etc ).
These are the values of what we shipped with.

(Should mention, when I did this talk at CEDEC 2025 couple weeks ago, made a mistake with the captures comparison - didn't compare exact same cascades/volumes being updated, there can be perf variations.
Fixed that for Siggraph also added more platforms for sake of completeness.)

Consoles are relatively close. Series X slightly faster, due 20% extra TFLOPS ( ~12TFLOPS vs ~10TFLOPS ). On this machine had a 4080, which roughly 4x the TFLOPS of consoles.
Async saved around 0.5ms on Series X and PS5. About 0.4ms on Series S. Not so much on PS5 Pro case, since the async dispatch cases actually run fairly fast.

Our async workload is actually pretty full, we try to keep GPU as busy as possible. Don't think we wouldn't have been able to achieve our performance target without.




Orders of magnitude savings (Hours down to milliseconds.)

Instant feedback for art.

0 disk space.

Unified/minimal code paths.

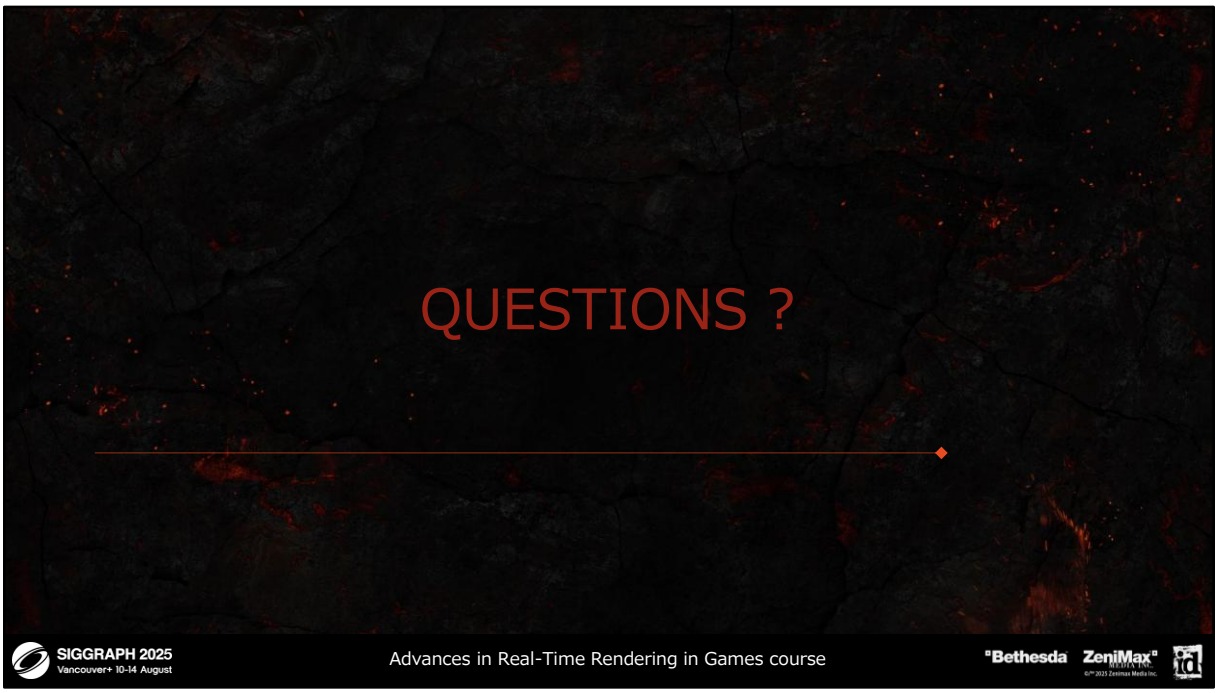
Consistent lighting results.
For every surface type.

# Special Thanks


- id Software:
  - Marty Stratton, Billy Khan, Bogdan Coroi, Darin McNeil, William Schilthuis, Oliver Fallows, Jean Geffroy, Phillip Hammer, Dominik Lazarev, Yixin Wang, Johan Donderwinkel, Regan Carver, Mel-Frederic Fidorra, Ian Malerich.
  - Our awesome art teams, for constantly pushing limits.
- Natalya Tatarchuk.
- All our fans !

# Thank you

- Tiago Sousa
  - tiago.sousa@idsoftware.com
  - X: @idSoftwareTiago
  - BlueSky: @idSoftwareTiago.bsky.social


QUESTIONS ?
SIGGRAPH 2025
Vancouver+ 10-14 August
Advances in Real-Time Rendering in Games course
Bethesda
ZeniMax

# References


[ Vahl2025 ] Pushing 60hz. Shipping Indiana Jones and The Great Circle

[ Ouyang2021 ] ReSTIR GI: Path Resampling For Real-Time Path Tracing

[ Halen2021 ] Global Illumination Based on Surfels

[ Wright2021 ] Radiance Caching for Real-Time Global Illumination

[ Geffroy2020 ] Rendering The Hellscape of DOOM Eternal

[ Gautron2020 ] Real-Time Ray-Traced Ambient Occlusion of Complex Scenes using Spatial Hashing. https://youtu.be/oza36AqcLW8

[ Majercik2019 ] Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields

[ Hobson2019 ] The Indirect Lighting of God of War

[ O'Donnell2018 ] Precomputed Global Illumination in Frostbite

[ Drobot2017 ] Improved Culling For Tiled And Clustered Rendering

[ Sousa2016 ] The Devil Is In The Details, idTech 666

[ Wronski2014] Assassin's Creed 4: Road To Next Gen Graphics

[ Lazarov2013 ] Getting More Physical in Call of Duty Black Ops 2

[ Olson2012 ] Clustered Deferred and Forward Shading

[ Mattausch2010 ] High-Quality Screen Space Ambient Occlusion using Temporal Coherence

# References


[ Czuba2010 ] Box Projected Cube-Map Environment Mapping http://devlog.behc.pl/ (http://web.archive.org)

[ Mittring2009 ] A bit more deferred – CryEngine 3

[ Ritschel2009 ] Approximating Dynamic Global Illumination In Image Space

[ Dachsbacher2008 ] Octahedron Environment Maps

[ Sloan2007 ] Image-Based Proxy Accumulation for Real-Time Soft Global Illumination

[ Tatarchuk2005 ] Parallax Occlusion Mapping: Self-shadowing, Perspective Correct Bump Mapping using Reverse Height Map Tracing

[ Landis2002 ] Production-Ready Global Illumination

[ Ramamoorthi2001 ] An efficient representation for irradiance environment maps

[ Greger1998 ] The Irradiance Volume

[ Ward1988 ] A Raytracing Solution For Diffuse Inter-reflections

[ SBT ] The Shader Binding Table Demystified, Will Usher, Ray Tracing Gems 2

[ Oodle Kraken ] https://www.radgametools.com/oodlekraken.htm