

Agenda

Agenda

Agenda:

- Background
- Acceleration structures
- Raytracing
- Denoising

• All timings for NVIDIA RTX 2070 @ 1440p

Raytraced Shadows in Call of Duty: Modern Warfare -00-07 Agenda 2020-— Agenda

• Agentis: • Backgrown (• Accelonation sectores • Byrouching • Danaking • All similing for WIELA RTX 2071 @ 1440 p

Agenda

• Here is our agenda for today.

- We will start by describing the motivation behind raytracing, and why we chose shadows.
- Next, we want to walk you through some problems we encountered during development. Some of those are specific to our title, but we mostly want to focus on general problems that most people will encounter when doing raytracing, and give you solutions that worked for us and that hopefully will work for you as well.
- Those topics include raytracing in general (from shader perspective), how to feed it with optimal data and how to denoise multiple lights per pixel at the same time.
- Any timings given during this presentation were collected on NVIDIA RTX 2070 at 1440p.

© 2020 Activision Publishing, Inc 2 / 75



Background

Background

Michał Olejnik & Paweł Kozłowski @ Digital Dragons 2020 🛛 Raytraced Shadows in Call of Duty: Modern Warfare 👘

© 2020 Activision Publishing, Inc 3/75

Background

• Beginning of our raytracing R&D

- Let's start with basics!
- Visibility only
- Shadows for local lights easiest?
- Separate acceleration structures per light
 - keeps performance problems local

```
Raytraced Shadows in Call of Duty: Modern Warfare
Background
Background
```



Background

- For the start of our DXR adventure, we wanted to keep things simple.
- We felt that sticking to visibility only algorithms will save us a lot of time, since we don't have to care about shading.
- This left 3 main candidates: ambient occlusion, sun shadows and local light shadows.
- We picked local lights, since we thought using per-light acceleration structures will help us deliver better performance, by keeping acceleration structures small, and any problems will be easier to contain. As it later turned out, this greatly helped with achieving high raytrace performance.

Background

© 2020 Activision Publishing, Inc 4 / 75

Background

Raytraced local lights

• Obvious benefits vs shadow mapping:

- area light shadows,
- contact hardening shadows,
- pixel perfect,
- physically correct.
- We already had analytical area light shading
- Shading and visibility are treated as separable
 - Not physically accurate [Heitz18]

```
Raytraced Shadows in Call of Duty: Modern Warfare

—Background
```

Raytraced local lights

-00-00

2020-

a O boloan kerefita on skadow mapping:
 arra Ightshadow,
 consets hadoning hadow,

Raytraced local lights

corners bardening shadows,
 pital particle,
 physically corners.
 We all waty had an adjetical a walling to shading
 Shading and shirility and twatted as reparable
 Nor physically accorne (bida 201

- Unlike a hacky approximation provided by shadow maps, raytraced shadows are physically correct and give multiple immediately visible improvements.
- Since we already had proper shading for area lights, inconsistencies with shadow maps were even more obvious.
- Unfortunately, to keeps things managable, we had to operate under assumption that visibility and shading are separable, which isn't true. However, raytraced shadows still provide vastly improved visuals.

© 2020 Activision Publishing, Inc 5 / 75

Background

Goals

- Must work with existing content:
 no per-light raytracing settings
 - multiple lights per pixel
- Must work with existing tile-based culling:
 different set of lights for each pixel
- Stable performance
 - constant ray count per frame
 - variable ray count per pixel/light

```
Raytraced Shadows in Call of Duty: Modern Warfare
Background
Goods
```

Must work with existing content:
 a real-plate space and graving a
 a real-plate lighter per yield
 a Must work with existing tills haved calling
 a filterene content light for a chained
 a Stable performance
 a manume space content for an
 a stable performance
 a minimum content for an

Goak

- We started our work after most maps were created, so we had to make sure raytracing will work without having to redo lighting for those maps.
- That meant no specific raytracing settings in editor, just use exisiting physical lights sizes.
- We also had to make sure that we handle multiple lights we have a limit of 256 lights per frame, and each pixel can have different set of lights. Even if there are many overlapping, they cannot flicker or disappear.
- We care about predictability of the frame time, so we want to keep raytracing execution time constant, while allowing for picture quality to scale depending on load.

© 2020 Activision Publishing, Inc 6 / 75

Goals

New tool for lighting artists:

- improved lighting quality
- higher lighting frequency than pre-baked

Background

- stick to ground truth, avoid bias
- prefer temporal filtering to spatial

Raytraced Shadows in Call of Duty: Modern Warfare -00-07 -Background 2020-└─ Goals

Now tool for fig hting actists:
 improved lighting quality
 high or lighting frequency than pro-baked
 noisk as good creat, an oil bins
 profer compared filtering or qualid

Goak

- We wanted to evaluate quality improvements given by area shadows. Evaluating them at real-time gives much higher frequency detail than what we can store in lightmaps (due to storage concerns).
- For this purpose we wanted to avoid aggressive spatial filters, because they usually need to have a big radius, which adds bias (especially when traycing in lower than native resolution) we chose to rely mainly on temporal accumulation.

© 2020 Activision Publishing, Inc 7/75



So how do we go from this...



to this - without killing performance?

Raytraced Shadows in Call of Duty: Modern Warfare Acceleration structures

Acceleration structures

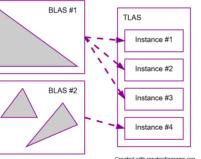
Acceleration structures

Michał Olejnik & Paweł Kozłowski @ Digital Dragons 2020 Raytraced Shadows in Call of Duty: Modern Warfare 📀 2020 Activision Publishing, Inc 10/75

Acceleration structures

Bottom/top level (BLAS/TLAS)

- Separation necessary to amortize build times
- TLAS sees BLAS only as it's top-level bounding box



Created with renderdiagrams.org

Raytraced Shadows in Call of Duty: Modern Warfare

└─Acceleration structures

2020-

a Bottom/toplovel (BLAS/TLAS) a Separation accessory to a months half times a TLAS was BLAS only as it's top-level hourding too

Acceleration structures

- There are two levels of acceleration structures in DirectX Raytracing API.
- Ideally, we would build one AS for the whole scene. However, for real-time purposes AS build times are non-negligible.
- So we usually build individual surfaces as separate BLASes and join them together in one big TLAS (or in our case, multiple TLASes).
- This way we have reasonable granularity individual BLAS build times and sizes are managable, they can be amortized between frames and removed after source meshes are streamed out.
- However, too naive approach can have significant performance implications. During TLAS building, the TLAS does not have access to insides of BLAS, it just sees its bounding box.
- This degree of separation is necessary, TLAS build times have to be kept on the low side, since they are rebuild every frame.

• **Problem** - What if multiple BLASes overlap?

- ClosestHit need to evaluate all overlapping BLASes
- AcceptHitAndEndSearch keeps evaluating until first hit found
- Either way: orders of magnitude slower
- Solution: build multiple geometries as one BLAS

Raytraced Shadows in Call of Duty: Modern Warfare 주 스Acceleration structures

BLAS merging

2020-

12/75

Produce - What if inditiale BLAS is over day? Charace Hermond to inclusive all methops ing BLAS is e AcceptibleArt ExtSorter - know moduling and the bicfored with the way, outers of magnitude shows: Solution: half all geometries as one BLAS

BLAS mensing

- The problems materializes when multiple BLASes have overlapping bounding boxes.
- Regardless whether you need closest or any hit, what usually happens is that ray traversal has to test against multiple BLASes, so performance degredates extremely fast.
- Fortunately, DXR API allows for providing a list of multiple index/vertex buffers during build, so we can build multiple surfaces as one BLAS.



Here is a worst-case scenario from our game. A light with not much geometry around.



Here it is zoomed out. How many objects do you think fall into this light's influence?



- Here you can see color-coded surfaces. Probably a bit more than you expected.
- What you see is geometry brushes used by level designers.
- Map compiler aggressively merges those surfaces for more efficient work submission.
- In most cases this means 1 drawcall for all triangles with the same material.
- Which means large bounding boxes and a lot of overlap.
- For DXR that is less than ideal.

2020-09-07

| Geometries per BLAS | Raytrace [ms] |
|---------------------|---------------|
| 1 | 9.12 |
| 2 | 7.08 |
| 4 | 4.47 |
| 8 | 2.78 |
| 16 | 2.14 |
| 32 | 1.73 |
| 64 | 1.43 |
| Empty TLAS | 0.88 |
| | |

Figure: BLAS merging performance

Raytraced Shadows in Call of Duty: Modern Warfare

└─BLAS merging

2020-09-07

16/75



BLAS menting

Here you can see actual performance for different number of surfaces per BLAS. And it would be even worse without AcceptHitAndEndSearch.

Instancing happens when passing BLASes to TLAS

• i.e. BLAS merging 'unrolls' instancing

• For our title, for each TLAS we have:

• brushes (huge, self-intersecting, low poly, no LODs) - merged • models (small, non-intersecting, high poly, use LODs) - instanced

Raytraced Shadows in Call of Duty: Modern Warfare 2020-09-07 Acceleration structures

BLAS merging

17/75

a lasta gine la succe a las anajas BLASes to TLAS wig. BIAS menting 'arrolls' instancing a For our title, for each TLAS on have: k rashes [k age, adf-in screecing, has p dy, no 10 Di] - morged a models break no in screecing, bird as b, and 10 Di] - in screecing.

BLAS mensing

• The downside of BLAS merging is that you can't reuse geometries as instances. • For us, especially since we have separte TLASes per light, the solution was rather clear.

- At the moment we pay the cost of brush+model trace
 - Acceptable compromise
- Naive merging won't scale for global TLAS:
 - Need for constants rebuilds due to streaming
- Solution:
 - Enforce non-overlapping surfaces between streamed chunks
 - Merge surfaces for each chunk separately

Raytraced Shadows in Call of Duty: Modern Warfare ⊖Acceleration structures

\sqcup BLAS merging

2020-

18/75

At the moment we pay the cost of its its model to companie
 Array and he companie
 Array and he companie
 Note may dip work task for plackal TLAS:

 Note for company task for plackal TLAS:
 Note for company task for a working
 Solution:
 Enforce more company samples and constrained when the Normal Works for constrained task.

BLAS mensing

- We did not fully get rid of overlap. In the case shown on screenshots, we will have one big brush BLAS covering almost entire map, so it will overlap every model inside it. This is an accepable performance trade-off.
- This approach won't scale when doing whole-scene TLAS.
- Fortunately, our streamer divides world into non-overlapping chunks, which we will be able to use for BLAS merging in the future.

Scaling

- **Problem** BLAS builds scale badly for smaller poly count
- Most apparent on CPU
- Multithreaded command lists advised
- Keeps improving with newer drivers

Raytraced Shadows in Call of Duty: Modern Warfare Acceleration structures

u <mark>Prodium</mark> - BLAS builds scale buily for smaller polycoust u Host approach or CPU u Hildsbud et commund Taba at daved u Hildsbud et commund Taba at daved

Sca ing

- It's important to note that CPU cost is not trivial. Consider recording multiple command lists in parallel when calling BuildRaytracingAccelerationStructure.
- This is something that keeps improving with newer drivers, so hopefully it will soon be a non-issue.

© 2020 Activision Publishing, Inc 19/75

Scaling

| Lod | Surfaces | Triangles [k] | CPU [ms] | GPU [ms] |
|-----|----------|---------------|----------|----------|
| 0 | 43 | 93 | 0.65 | 1.14 |
| 1 | 43 | 45 | 0.63 | 0.67 |
| 2 | 39 | 18 | 0.58 | 0.45 |
| 3 | 37 | 8 | 0.54 | 0.36 |
| 4 | 37 | 3 | 0.54 | 0.33 |
| 5 | 24 | 2 | 0.36 | 0.26 |
| 0 | 4 | 45 | 0.12 | 0.74 |
| 1 | 4 | 19 | 0.12 | 0.52 |
| 2 | 3 | 10 | 0.10 | 0.46 |
| 3 | 3 | 5 | 0.10 | 0.40 |
| 4 | 3 | 2 | 0.09 | 0.31 |
| 5 | 2 | 1 | 0.08 | 0.28 |

Figure: Skinned BLAS build performance before and after content optimizations

Raytraced Shadows in Call of Duty: Modern Warfare

 \square Scaling

2020-09-07

| Scaling | | | | | | | |
|---------|--------------|------|-------------|---------------|------------------|-------------|---------------|
| | | | | | | | |
| | | La d | Sattace | Diarghi (k) | | G # U [# 1] | |
| | | - | 41 | 11 | 1.15 | 1.14 | |
| | | | 41 | 45 | 0.03 | 1.17 | |
| | | 2 | 38 | | 0.51 | 1.45 | |
| | | 3 | 37 | | 0.54 | 0.11 | |
| | | 4 | 37 | 3 | 0.54 | 0.11 | |
| | | 5 | 24 | 3 | 0.11 | 1.21 | |
| | | _ | 4 | 45 | 1.12 | 1,74 | |
| | | | 4 | | 8.12 | 0.52 | |
| | | 2 | 3 | | 1.11 | 1.41 | |
| | | 3 | 3 | 5 | 1.11 | 1.41 | |
| | | 4 | 3 | 3 | 1.11 | 0.31 | |
| | | 5 | 2 | | 1.11 | 1.21 | |
| | figere: Skin | nd B | LA 5 E vill | performance b | effore and after | | / dista di su |

• Here you have CPU and GPU timings for BLAS building of different LODs for one of our character models. This is both before and after content optimizations.

• As you can see, performance does not scale linearly with vertex/surface count. Tiny surfaces are quite heavy.

Michał Olejnik & Paweł Kozłowski @ Digital Dragons 2020 🛛 Raytraced Shadows in Call of Duty: Modern Warfare

© 2020 Activision Publishing, Inc 20/75



| - | | | |
|----|-----|-----|----|
| Ra | vtr | acu | nø |
| | | | |

Raytracing goals

• For best raytracing performance:

- Avoid multiple shaders (divergence, instruction cache issues)
- Avoid AnyHit shaders (extra work during ray traversal)
- Avoid recursion (increases register pressure)
- Keep ray payload size low (to minimize memory bandwidth)
- Use AcceptHitAndEndSearch when possible (allows rays to terminate earlier)

Easy with shadows!

Raytraced Shadows in Call of Duty: Modern Warfare

22 / 75

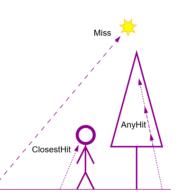
For hist sytuating preform usit:
 Antil multiple data in Histopra, investoir cach i ku of
 Antil Artylfe data in province and t dring processful
 Antil Artylfe data in province spectra province
 Antil Artylfe data in province spectra province

Ravtracing goals

· Fann mit bis had own

- To achieve high raytracing performance, we should avoid using expensive features unless necessary.
- Which is precisely where raytraced shadows shine: we do not need per-material shaders, recursion, and we don't need closest occluder any occluder will do.

| | Raytracing |
|--|------------|
| Raytracing in practice | |
| | |
| | |
| • 1x Raygen | |
| • 1× Miss | |
| 1x ClosestHit | |
| • 2x AnyHit: | |
| alpha testing | ClosestHit |
| ray bias for mismatched geometry | etry |
| | // |
| | |



Created with renderdiagrams.org

23 / 75

Raytraced Shadows in Call of Duty: Modern Warfare └─Raytracing

Raytracing in practice

-00-00

2020-

ix Reggen
 ix Min
 ix Min
 ix Closed Hit
 ix Any Hit
 ix Any Hit
 ix Any Hit
 ix Any Hit

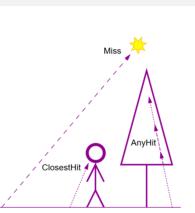
Ravtracing in practice

- Keeping divergence low is easy, since we only need a very specific set of shaders.
- There two caveats though: we need to handle alpha tested geometry, which necessitates use of not-so-performant AnyHit shader.
- Same type of shader is also used to introduce ray bias for acceleration structures which geometries do not exactly match positions reconstructed from rasterized depth. But more on that later.

| | Raytracing | |
|------------------------|------------|--|
| Raytracing in practice | | |
| | | |
| | | |

Raycast from pixel to light

- no recursion
- Max trace distance = distance to light
 - no need for light geometry



Created with renderdiagrams.org

24 / 75

Raytraced Shadows in Call of Duty: Modern Warfare

Raytracing in practice

-00-00

2020-

a Reycast from give to light

 n reconting

 Max turn distance = distance to light

 a mod for light gammer;

Raytracing in practice

Assuming we have acceleration structures built, raycast is easy - using reconstructed world space position we cast rays towards the light, by importance sampling a capsule shape.
We do not need to build BLAS for light shape, since we can simply set max trace distance to distance to our importance sampled point on light - the miss shader will be called if no intersection between two points is found.

| ĸ | avtr | acir | 1g |
|---|------|------|----|
| | | | |

Raytracing in practice

Payload needs:

- 1 bit for visibility
- Couple bits for shadow caster velocity (for denoising)
- Miss shader flips visibility bit
- ClosestHit still needed for velocity
 - Encoded in unused bits of InstanceID (24 bits total, 10 bits for velocity)
 - One less memory fetch

Raytraced Shadows in Call of Duty: Modern Warfare

Raytracing in practice

-07

00

2020-

Raytracing in practice

Poplari recht:
 Crait für infähig
 Crait für infähig
 Marin für infähig
 Marin für infähig
 Marin für infähig
 Marin für infähig
 Crantifik all merint für obsidig
 Der kommen für i

- In simplest case, we would only need 1 bit in ray payload and a miss shader to determine visibility, since DXR API allows for not providing a ClosestHit shader at all.
- However, our denoiser uses shadow caster velocity for tuning temporal convergence (more on that later).
- To avoid any extra memory fetches, we encode compressed velocity in unused bits of InstanceID.
- Note that we use AcceptHitAndEndSearch so we aren't guaranteed to get velocity of actual shadow caster, but we haven't encountered any cases in practice where this would be an issue. Usually, a moving object does not have an overlapping bounding box with static geometry, so we get an actual closest hit even if didn't specif that we wanted to.

Raytracing

Alpha testing

Problem - Alpha testing

affects ray traversal

cannot be done in Raygen

- dependent fetches (indices, uvs, texture)
- slowdown from 1.5ms to ~4.5ms
- Unsolved for now:
 - use high poly opaque instead?
 - measure TLAS complexity or raygen duration and adjust ray count accordingly?
 - promising research on alpha-testing has been published recently [Gruen20]

```
Raytraced Shadows in Call of Duty: Modern Warfare
```



• Unfortunately, we were unable to solve performance issues with alpha testing.

- There is no easy solution here alpha testing requires dependent memory fetches, and even if it is able to be properly amortized by other threads on GPU, it's still a substantial amount of additional work.
- Using high-poly opaques has been an object of research, however there is no guarantee of it being faster, and it is going to take more storage.
- We have also thought of detecting if given TLAS has any alpha tested geometry, and adjusting ray count accordingly, but this is more of a workaround than solution and it won't work with global TLAS.
- In similar manner, we could adjust ray count based on previous frame raygen execution time.
- Fortunately, since we shipped, some promising research on potential alpha-testing hardware support has been published.

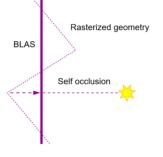
| | Raytracing |
|---------------------------------|------------|
| Mismatched geometry | |
| | |
| | |
| Problem Mismatched geometry | В |

Problem - Mismatched geometry

 e.g.: vertex animation, tesselation, subdiv

 no indirect builds in API

need to know index/vertex count on CPU



Created with renderdiagrams.org

-00-00

2020-

Raytraced Shadows in Call of Duty: Modern Warfare

└─ Mismatched geometry

Position - Minutched georety • e.g.: ener animatics, matchin, activity • e.initate balls is: API • e.end as bravit des/ener concert CPU

Mis matched geo metry

- As previously mentioned, we have cases where geometries in accelerations structures do not match the geometries used with rasterizer.
- This concerns geometries generated using vertex and geometry shaders, but even after porting them to compute it still would be impossible to build them.
- DXR API lacks indirect builds, so we would need to know index and vertex count on CPU, which is not possible for GPU driven data.

© 2020 Activision Publishing, Inc 27/75

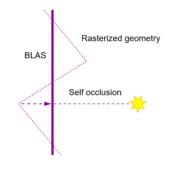
| | Raytracing |
|---------------------|------------|
| Mismatched geometry | |
| | |

Workaround - ray bias

- could bias minT in Raygen, based on a stencil bit...
- ... but what if we need to set bias per object?

not enough bits!

• Solution: ray bias in AnyHit shader



Created with renderdiagrams.org

Raytraced Shadows in Call of Duty: Modern Warfare

└─ Mismatched geometry

-00-00

2020-

Works word - my kins
 could him mit? in Region, hand on a sound him
 could him mit? in Region, hand on a sound him
 could him him
 a sound him him
 a sound him him

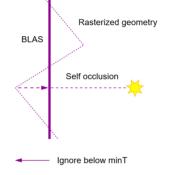
Mis matched geo metry

- We thought of biasing minimum trace distance, if we only could determine object type per-pixel, like e.g. from stencil.
- However, we did not have enough stencil bits left for our needs, so we came up with different solution.
- By using bias in AnyHit shader, we can have different biases on even per object basis.

Raytracing

Mismatched geometry

- if (RayTCurrent() < tBias) IgnoreHit()
- Different tBias depending on geometry type
- Encoded tBias index in unused InstanceID bits
 avoids extra loads, no performance delta
- Note that:
 - stencil bias per receiver
 - AnyHit bias per caster



Created with renderdiagrams.org

29/75

Raytraced Shadows in Call of Duty: Modern Warfare └─ Raytracing

— Mismatched geometry

-07

00

2020-



Mis matched geo metry

- We previously scrutinized performance of AnyHit shaders in relation to alpha testing, however they are not that slow if you don't do too much work in them.
- We use AnyHit shader to reject hits below certain threshold.
- There are couple different hardcoded thresholds for different types of features (subdiv, tesselation etc.), and we encode index to them in unused bits of InstanceID to avoid any extra memory indirections.
- Of course, the two approaches described are a bit different, since once biases by receiver, the other by caster.
- In our case it was mostly used to avoid self-shadowing issues on subdiv character faces and tesselated terrain, so the AnyHit method was actually preferable.



Here we can see tesselated terrain with black spots due to mismatched geometry.



And here it is after applying bias.





Denoising

Denoising

Michał Olejnik & Paweł Kozłowski @ Digital Dragons 2020 Raytraced Shadows in Call of Duty: Modern Warfare 🧿 2020 Activision Publishing, Inc 32 / 75

Denoising

Denoising

• Requirements:

- Forward + pipeline
- No gbuffers except for vertex normal
- Lots of material shader permutations
 - Cannot afford extra permutations just for DXR
- Final denoised texture must be easy to sample by F+

Raytraced Shadows in Call of Duty: Modern Warfare -00-00 Denoising 2020-Denoising

. Ren alte mente! Farman de pipeline
 Na schaffer auszen ofte nerves normal Los of material shader a compaction. · Connect affect and a new explanation to the Ref. Final demained propage many his easy to sample by

Denoising

- Call of Duty: Modern Warfare uses forward+ shading, which means we have a lot of specialized material shader permutations.
- Shaders require recompiling after each driver or game update, so we didn't want to bloat game with even more permutations.
- Any new code added to F+ shaders would be present everywhere even if inactive including our min spec, so we had to make sure it won't add extra register pressure.
- We aimed to have a simple texture that can be point sampled, with each texel containing floating point visibility for multiple lights.

33 / 75

| | | 1.1.1 | |
|---|-----|-------|------|
| D | enc |)ISI | ng |
| _ | | | ···o |

• 4 main passes:

- Light indices resolve + load balancer
- Raytrace
- Temporal denoiser
- Spatial denoiser + upsampler

```
Raytraced Shadows in Call of Duty: Modern Warfare
2020-09-07
      -Denoising
          Denoising
```

• 4 main passes: Light indiges results + load balancer Rayman
 Terra real distribution · Spacial denniner + a sparra let

Denoising

• Here you can see the high-level outline of our whole gpu-side implementation.

• Before we go into details, let's see what it looks like in practice.

Denoising

© 2020 Activision Publishing, Inc 34 / 75



Exponential shadow maps - leaking, missed details.



Raytraced pointlight.



Raytraced sphere light - raw input, before denoising.



Without denoising, but with TAA. Pretty good...



... but dedicated shadow temporal denoiser is better.



Adding delicate spatial blur and TAA completes the final look.

Denoising

• Everything done in half-resolution

- Denoise up to 4 separate visibility signals
 - Light indices can differ between pixels
 - Up to 3 non-denoised lights as a fallback (1 bit each)
 - Data for all lights stored in the same UAV for best performance [Aaltonen16]
- Variance-based filtering [Schied17]

```
Raytraced Shadows in Call of Duty: Modern Warfare
```

Denoising

- Note that all UAVs and compute dispatch sizes are half-resolution (25% pixel count).
- For each texel, we store 4 separate channels containing light visibility, which means we can denoise up to 4 lights per pixel.
- 4 lights per pixel are plenty, but in rare cases there can be more.
- We didn't want to force our lighting artists to redo lighting for DXR, so we added a fallback that stores not-denoised visibility for extra 3 lights (7 in total).
- Thankfully, our temporal anti-aliasing is pretty good at denoising by itself, so those places are not too jarring.
- We wanted to keep all our data in the same UAV according to multiple benchmarks one fetch of 4x wider texture is faster than sampling 4 textures separately.
- We use variance-based filtering, based on the ASVGF paper, which is also used in Quake 2 RTX.

Data format

Fat UAVs:

- RGBA32 sampled in temporal and spatial
- RG32 sampled in temporal, only for center sample in spatial

Lots of data:

- 1st and 2nd visibility moments for each channel
- Light indices
- Temporal history length
- Max shadow caster velocity
- Normal, depth

```
Raytraced Shadows in Call of Duty: Modern Warfare
```

42 / 75



Data format

- 4 lights per pixel requires a lot of storage, so our UAVs are quite fat.
- Fortunately, we were able to avoid sampling all data at all times, i.e. spatial denoiser does not have to sample auxiliary RG32 texture when sampling neighbourhood.
- To compute variance, we need to store 1st and 2nd visibility moments for each channel separately.
- After first pass, we also keep per-pixel normals and depth in the same UAV that way we don't have to do extra memory fetches.
- Other data is needed for various heuristics that determine temporal and spatial filters' strengths.

Indices resolve

• Indices pass translates tile-culled lights bitmask to per pixel 8-bit indices [Drobot17]

- Applies spot shadow falloff to minimize number of rays cast
 - \bullet This solves lack of culling refinement step that is done normally in F+
- Outputs to RG32, read by raytrace pass
- Full-resolution also read during upsampling

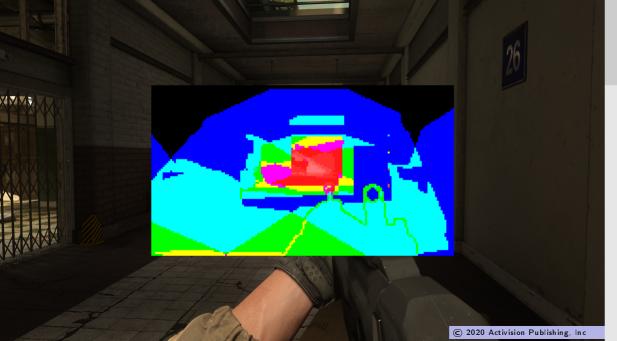
```
Raytraced Shadows in Call of Duty: Modern Warfare
```

43/75



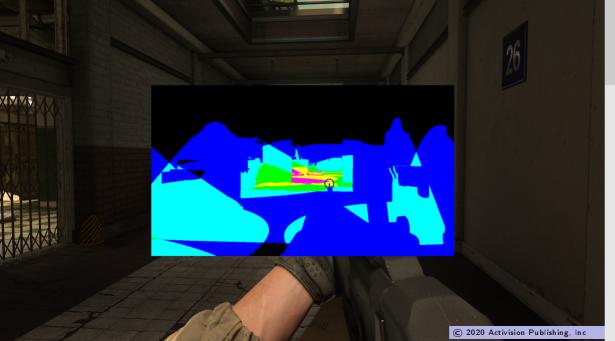
Indices resolve

- We have a highly optimized light culling scheme for F+.
- However, it does not check for spot shadow falloff during culling itself to avoid fetching light descriptors falloff is applied during F+ which is faster overall.
- But with DXR we want to skip any unnecessary rays, so we want to apply falloff before raycast.
- Indices resolve pass reads light culling bitmask, applies falloff and writes to an intermediate RG32 UAV, which is later read by raytrace pass and upsampling pass.
- We output indices in full-resolution to use later during upsampling.



Heatmap of tile-culled lights.

2020-



Heatmap after spot shadow falloff.

2020-

Load balancing

- Atomic reduce add on all per-pixel light counts
 - Thread group downsample during indices resolve pass
 - Extra downsample chain until we have only 1 pixel extra ~0.02ms
- Total sum of lights used for determining per-tile ray count during raytrace pass
- Target of 4 rays per pixel (in half-resolution), e.g.:
 - 1 full-screen light => 4 rays per light per pixel
 - 4 full-screen lights => 1 ray per light per pixel
 - 1 half-screen light => 8 rays per light per pixel

```
Raytraced Shadows in Call of Duty: Modern Warfare
```

46 / 75

 Ansist on a start of a start provint layer cannot - Tord grave frammels for information on - for a drawing bit information of the start of the start - for a drawing bit in or both the start (information) - Tord and grave and the start ording product system (range - a drawing bit in bit ordinal (range) - a drawing bit is bit ordinal (range) - a drawing bit is bit ordinal (range) - bit drawing bit is - 1 or are still bit or plot - bit drawing bit is - 1 or are still bit or plot

Load balancing

- We also do an atomic add reduction to count all lights on screen
- This is used to dynamically adjust ray count per pixel to maintain constant raytracing performance.

Temporal denoiser

- Combines output of current frame raytrace pass with previous frame results
- Temporal reprojection using existing motion vectors
- Bilinear filtering, but with manual gather and interpolation
 - Needed per sample rejection based on depth, normal and indices
- Shadow caster velocity affects convergence
 - Turn off temporal for moving shadow casters
- Accumulates both 1st and 2nd visibility moments
- Output used by spatial and next frame temporal

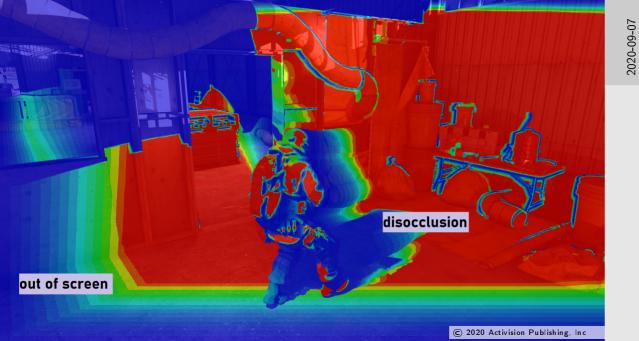
```
Raytraced Shadows in Call of Duty: Modern Warfare
```

47 / 75

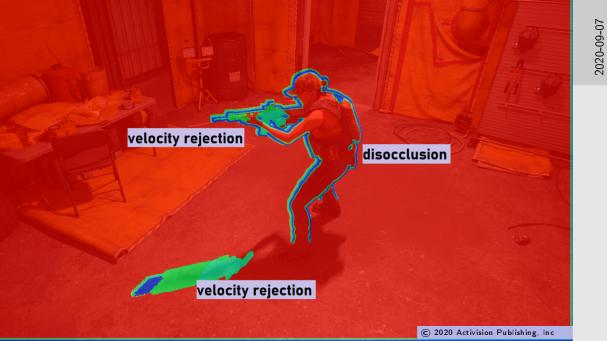
• Combine and pit of some option provide produce of the south • Tormpool comparison and producing methods works a • Billiner Sheling pit works in most of pits or at internalistic • Billiner Sheling pit works pits to and or pits or at internalistic • Billiner Sheling pits pitters may any any any • Stremmon or any sheling pits of the source of the source of the source • Atorematics task bits of 1 internalistic • Output out by source of the source of the source of the source • Output out by source of the source of the source of the source • Output out by source of the source

Temporal demoiser

- Raytrace pass uses indices to determine light visibility and passes this information to temporal denoiser.
- It's a pretty standard temporal accumulation scheme with feedback loop. There are couple caveats though.
- We want to apply rejection per each bilinear sample e.g. we do not want to reject based on depth after bilinear filtering - so we have to implement bilinear filter ourselves.
- To avoid ghosting on moving shadows, we tune out temporal filtering when shadow caster velocity is non-zero.
- We accumulate both 1st and 2nd moments so that spatial denoiser can estimate variance over time.



Heatmap of temporal history length, with character and camera in motion.



Heatmap of temporal history length, with idling character and static camera.

| en | | | |
|----|--|--|--|
| | | | |

Temporal denoiser

- **Problem** can't query for texels touched by hardware Gather
 - May result in being off by 1 pixel using weights for different texels
 - Temporal accumulation completely explodes
 - Avoid Gather, load each sample separately

ABCD or CDEF? Created with renderdiagrams org

A C

В D

Е

Raytraced Shadows in Call of Duty: Modern Warfare -00-00 -Denoising 2020-└─Temporal denoiser



Temporal demiser

- One extra issue we ran into is that we can't make assumption about exact math done by hardware when picking samples for Gather.
- We can't query hardware for texels touched during Gather with given uv we have to choose them ourselves
- We have 2 UAVs with 6 channels, so instead of doing 6 gathers we ended up doing 8 loads, which lowered performance a bit.

Michał Olejnik & Paweł Kozłowski @ Digital Dragons 2020 Raytraced Shadows in Call of Duty: Modern Warfare

© 2020 Activision Publishing, Inc.

50 / 75

| ID) | enoising | |
|-----|----------|--|
| | | |

Temporal denoiser

- **Problem** how to match light indices efficiently?
- Lights evaluated at every pixel can be different
- This causes dynamic indexing of registers
 - Inefficient codegen
- Solution: do not unpack dynamically indexed data ahead of time

```
Raytraced Shadows in Call of Duty: Modern Warfare
2020-09-07
      Denoising
```

— Temporal denoiser

n Postlam, bout to match light indices efficiently · Lights evaluated at every sizel can be different. a This causes do not introduce of registers · Inefficient conferen A Solution, do not a next dynamically independents a head of time

Temporal demoiser

- One interesting performance caveat is how to match light indices between samples. This applies both to temporal and spatial denoisers.
- Each pixel can have a different set of light affecting it, so even if two pixels share same lights, it's not guaranteed they will be in the same order.
- This isn't something that can be precomputed, so we ended up with dynamic register indexing.
- This is something that is supported neither by DXC nor actual hardware. DXC ended up producing code that either forced data to go through memory, or a lot of inefficient ALU magic involving conditional masks and swaps.
- This is usually easily remedied by manually using LDS, but in this case we can do even better

uint4 packedData = texture[samplePos];

```
// BAD:
float unpackedChannels[4] = Unpack4UnormsFromUint( packedData.x );
float visibility = unpackedChannels[dynamicIndex];
```

// GOOD: float visibility = UnpackUnormFromUint(packedData.x, dynamicIndex); // ((packedData.x >> (dynamicIndex * 8)) & 255) / 255.0;

```
Raytraced Shadows in Call of Duty: Modern Warfare
```

```
uint4 packedData = textur(ismpldfas);
// Bob;
// Bob;
// Bob;
// Soc.uspldfastes/schemesi/dynamicsinder);
// soc.uspldfastes/schemesi/dynamicsinder);
// soc.uspldfastes/schemesi/dynamicsinders);
// ( ( packedData > 0 ( dynamicsinder + 1) ) & 205 ) / 355.4;
```

- Thankfully, our visibility for 4 channels in stored in one dword.
- Instead of unpacking it ahead of time, we can unpack it 'on demand'.
- Unpacking 1 byte float from a dword involves a bit-shift by 0, 8, 16, or 24. We can simply make the bitshift amount dependent on the matched light index.
- This avoid dynamic indexing and reduces register pressure.

© 2020 Activision Publishing, Inc 52 / 75

-07

60

2020-

Spatial denoiser

Spatial poisson disk blur

- Fixed-radius pattern, sample weights depend on variance:
 - Variance computed from 1st and 2nd moment (temporal estimate only)
 - Longer temporal history => lower variance => lower filter strength
 - Penumbra => high variance => strong filter
 - Contact shadows or point light shadows => zero variance => no filter

Raytraced Shadows in Call of Duty: Modern Warfare

└──Spatial denoiser

70-00

2020-

- Our spatial denoiser is a simple, non-separable, poisson disk blur.
- We apply weighting heuristics that include depths, normals and variance.
- Since we denoise multiple lights at the same time, we do not use variance to scale sampling radius.
- Instead, we modify sample weights based on variance.
- Spatial filtering introduces bias to our shadows so we want to keep it's strength low if it is not needed.
- As temporal accumulation collects more frames, variance goes down, so we fade out the strength of spatial filter.
- Outside of that, variance filtering allows us to avoid overblurring of contacts shadows or point light shadows.

Michał Olejnik & Paweł Kozłowski @ Digital Dragons 2020 Raytraced Shadows in Call of Duty: Modern Warfare © 2020 Activision Publishing, Inc.

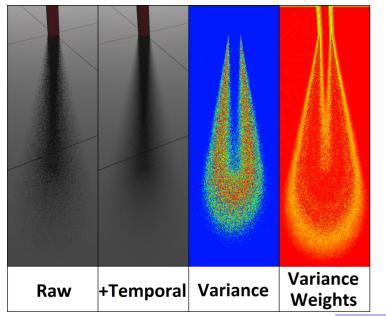
53 / 75

Spatial denoiser

• Statial poisson disk blan a First action action as well and state to and converting a · Variation compared from the and that moments from actual opinion on b Longer commental history in a longer surjection of history filter strategilt

· Perambra => high pariance => score fike

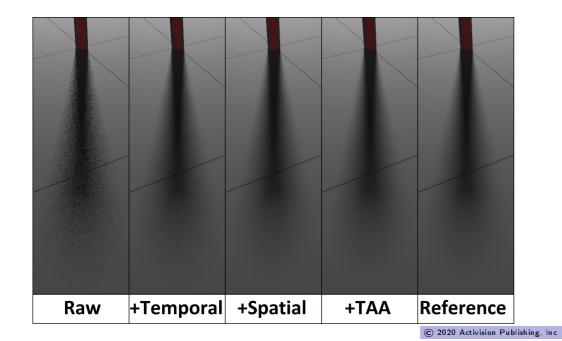
Concerns adult concerns for light of all concerns, rate and an end of the



Here is a heat map of variance (noisy, because variance estimation is only temporal) and resulting heatmap of variance weights during spatial filter.

© 2020 Activision Publishing, Inc

2020-09-07



2020-09-07

Whole denoising process - if temporal denoiser has time to converge, spatial blur and TAA will only add minor refinement.

Spatial denoiser

- Fixed-strength filter for first couple frames after disocclusion
 - No variance filtering => more bias, no contact shadows
 - Not noticeable for disocclusion events
- Noticeable noise for wide penumbra from dynamic shadows
 - Maximum spatial filter strength, but still noisy
 - Mostly rely on end-frame TAA for clean-up
- Need to investigate spatial variance estimation
 - Extra pass extra cost
 - Can we do smooth transition between spatial and temporal variance?
 - Can we avoid overblurring hard shadows while simultaneously handling wide penumbras?

```
Raytraced Shadows in Call of Duty: Modern Warfare
```

56 / 75

Floristicsgli files for that may be transition in the chains is contrast, the last sense in the construction of is a contrast, the last sense is the construction of which may and the construction of the construction is the construction of the construction of the construction is the construction of the construction of the construction is the construction of the construction of the construction is the construction of the construction of the construction is the construction of the construction of the construction is the construction of the construction of the construction is the construction of the construction of the construction is the construction of the constructio

Spatial denoiser

- Our variance estimation is temporal only, so we are left without variance knowledge if disocclusion occurs.
- In such cases, we simply turn off variance filtering and employ a fixed filter.
- This results in very arbitrary filtering, so artifacts can be visible.
- Fortunately, our temporal anti-aliasing is very good at denoising, so you won't notice that in motion, unless you encounter an extremely big light with dynamic occluders around.
- We considered an extra spatial variance estimation pass, but to truly make it worth the cost we would need to ensure high quality without sacrificing performance.

Upsampling

- Upsampling done at the end of spatial denoiser
- Bilinear upsample using groupshared memory
- Uses full-resolution indices
- Final full-resolution output for sampling in F+ contains:
 - 4x 8 bits (denoised) packed in 1st dword
 - 3x 1 bit (not-denoised) packed 2nd dword

```
Raytraced Shadows in Call of Duty: Modern Warfare
```

57 / 75

Upper opfing does at the red of spatial decision
 Billion reprinciple of age oppihe of memory
 Upper differentiation interface
 Field fill-conduction entry for for sampling in Factorial
 at the filter interface for standard and the statement

Ussampling

- To amortize denoising cost, we do upsampling in the same pass as spatial denoiser.
- We use groushared memory to share visibility values between neighbours and do bilinear filtering for missing texeles.
- Full resolution indices from the first compute pass are used, to avoid any artifacts.
- The final texture sampled by F+ is in full-resolution, and contains visibility for both denoised and non-denoised channels.

Upsampling

- What about artifacts on compute group edges?
- Could make compute group borders overlap (extra threads, extra cost), but...
 - Add jitter when picking half-res sample (4x)
 - Projection jitter from TAA (2x)
 - 8 unique temporal samples => errors barely noticeable

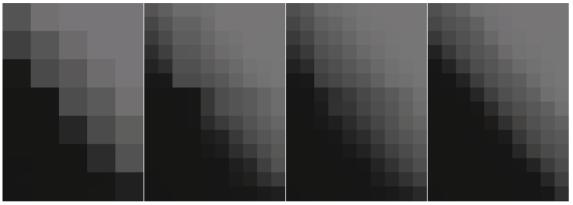
```
Raytraced Shadows in Call of Duty: Modern Warfare
```

58 / 75

What a best artflects or compute group edges? Could make compute group head to use help locate through, extra coult, hat... Add (() no also () high (a) head on match () a) A hydron () no form (AA ()) A matching and () a > a come heady redicable

Ussampling

- Since we cannot access visibility from different compute groups, we have to guess values for border texels.
- We could make compute groups overlap by duplicating work of border threads, but in practice we found it to be an overkill.
- We already jittered in time which sample of 2x2 neighbourhood we pick for our half-res textures, and with 2x TAA jitter we have 8 unique samples in 8 consecutive frames, which makes all errors imperceptible.



Half-res +Bilinear +Jitter Full-res

Raytraced Shadows in Call of Duty: Modern Warfare

Note the doubled pixels on compute group edges before adding jitter. This can be a visibly distracting pattern, though only on flat surfaces spanning multiple tiles.

© 2020 Activision Publishing, Inc

2020-09-07

2020-09-07

Summary

Summary

• Open problems:

- Alpha testing performance
- Mismatched geometry
- Better spatial denoiser

```
Raytraced Shadows in Call of Duty: Modern Warfare
```



- During the talk we outlined couple problems that we still need to solve.
- Alpha testing will most likely have to be replaced with high poly opaque BLASes, but actual performance/mesh complexity balance is yet to be measured.
- Mismatched geometry is the biggest concern, since modern vertex shading pipeline seems to evolve in orthogonal way to raytracing with advent of mesh shaders.
- And finally, we need to improve our spatial denosing scheme. NVIDIA Raytracing Gameworks provides a good spatial denoiser, but it is not suited to denoising multiple lights. It's worth investigating how it could be integrated with our framework in performant way.

Summarv

© 2020 Activision Publishing, Inc 61 / 75

Summary

Summary

Achievements:

- High performance thanks to local TLASes
- Constant performance regardless of light count
- No content adjustments needed
 - All lights are raytraced

```
Raytraced Shadows in Call of Duty: Modern Warfare
```



- However, many things did went well for us. Barring alpha testing, our raytracing is very performant and it scales well with content.
- The execution cost is more or less constant regardless of light count.
- Most importantly it works well for all lights, our artists did not have to redo lighting or turn off raytracing for individual lights.

Michał Olejnik & Paweł Kozłowski @ Digital Dragons 2020 Raytraced Shadows in Call of Duty: Modern Warfare 📀 2

© 2020 Activision Publishing, Inc 62/75

Performance

| Light count | 0 | 1 | 4 | |
|-------------|------|-------|-------|--|
| Indices | 0.21 | 0.30 | 0.47 | |
| Raytrace | 0.10 | ~1.50 | ~1.50 | |
| Temporal | 0.11 | 0.37 | 0.38 | |
| Spatial | 0.30 | 0.38 | 0.41 | |
| Total | 0.72 | 2.55 | 2.76 | |

Figure: Raytracing performance in [ms], NVIDIA RTX 2070 @ 1440p

Raytraced Shadows in Call of Duty: Modern Warfare

-Performance

2020-09-07



Performance

• Here you can see average raytracing performance.

- Load balancing maintains constant cost.
- Raytrace is usually about 1.5ms, rarely goes above 2ms, except for locations with heavy use of alpha testing.

Michał Olejnik & Paweł Kozłowski @ Digital Dragons 2020 Raytraced Shadows in Call of Duty: Modern Warfare © 2020 Activision Publishing, Inc

63 / 75



Here are some in-game examples of our denoiser in action. Proper area light shadows give desired soft look.



Here are some in-game examples of our denoiser in action. Proper area light shadows give desired soft look.



They help to ground objects in their surroundings.



They help to ground objects in their surroundings.



They do not experience leaking and ensure penumbra widening with distance from shadowcaster.



They do not experience leaking and ensure penumbra widening with distance from shadowcaster.









| <u> </u> |
|----------|
| Summary |

Acknowledgments:

- Oleg Kuznetsov
- Tim Cheblokov

Contact:

- molejnik@infinityward.com, Twitter: @olej3d
- pkozlowski@nvidia.com

Thanks!

```
Raytraced Shadows in Call of Duty: Modern Warfare
```

Arismuleignests:
 Org. Korrean
 Tim Chekkins
 Costat
 enskihi @istingund.com, Turkon: @ilajld
 etterketi@indik.com

Summary

- We wanted to thank Oleg and Tim from NVIDIA for their help with our work.
- If you have any questions, feel free to contacts us.
- Thank you for your time!

74 / 75

Summarv

Bibliography

[Heitz18] Eric Heitz, Stephen Hill and Morgan McGuire

Combining Analytic Direct Illumination and Stochastic Shadows https://eheitzresearch.wordpress.com/705-2/

[Gruen20] Holger Gruen, Carsten Benthin, Sven Woop Sub-Triangle Opacity Masks for Faster Ray Tracing of Transparent Objects https://youtu.be/prZJ8FBG9BI?t=11416

[Aaltonen16] Sebastian Aaltonen

PerfTest
https://github.com/sebbbi/perftest#nvidia-turing-rtx-2080-ti

[Schied17] Christoph Schied et al.

Spatiotemporal Variance-Guided Filtering

https://research.nvidia.com/publication/2017-07_Spatiotemporal-Variance-Guided-Filtering%3A

[Drobot17] Michał Drobot Improved Culling for Tiled and Clustered Rendering http://advances.realtimerendering.com/s2017/2017_Sig_Improved_Culling_final.pdf Raytraced Shadows in Call of Duty: Modern Warfare

\square Bibliography

-07

60

2020-

Bibliography

(Hitz II) Eic Hitz, Stephen Hilland Morgen McGain Combining Analytic Direct III aminution and Stochastic Shatowa https://ebsitzressarch.wordpress.com/785-2/

Granz II, Heljer Gram, Carrier Benkin, Sim Wrap Sal-Thingle Opacity Wark for Norice Ray Tucing at Transporter Object https://yeeku.be/pr2307050027t=11416

https://github.com/sebbbi/perftestHnvidia-turing-rts-2000-ti

[Schief2] Cohuman Schie et al. Syn inter pull VainceCohi at filming https://reserve.astin.com/publication/2017-07_Spatiotemporal-Variance-Guided-Filterin [Data 10] Vakial Data.

Improved Calling for Tiles and Claused Rendering http://dvances.waltimerendering.com/s2017/2017_Sig_Improved_Calling_final.pdf