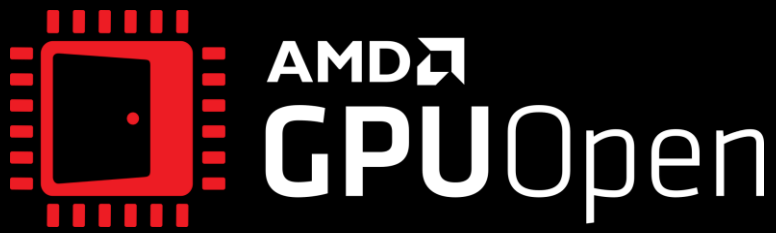
PERFORMANT REFLECTIVE BEAUTY: HYBRID RAY TRACED REFLECTIONS IN FAR CRY 6

STEPHANIE BRENHAM, UBISOFT TORONTO

IHOR SZLACHTYCZ, AMD

GDC



REFLECTIONS BACK ON THE UBISOFT TEAM

Stephanie Brenham
- 3D Team Lead Programmer
- Far Cry 6
- The Games Awards Future Class 2021
- Maya & mental ray for Maya

Anton Remezenko
- Senior 3D programmer
- Far Cry 6
- Far Cry 5

Aleksei Shevchenko – 3D programmer
Mikhail Shostak – 3D programmer

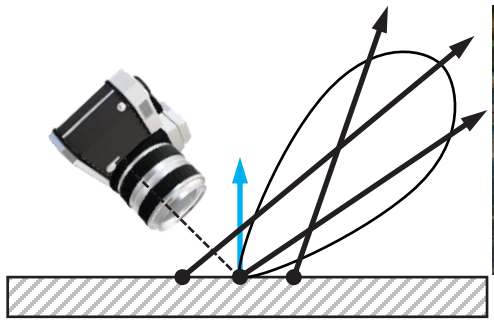
FAR CRY 6

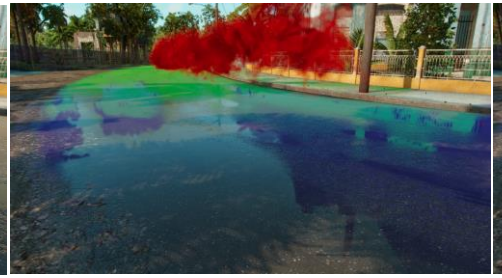
- FPS set in a fictional tropical island
- Humid tropical setting
  - See “Simulating Tropical Weather in Far Cry 6”
- Lots of wet and highly reflective surfaces

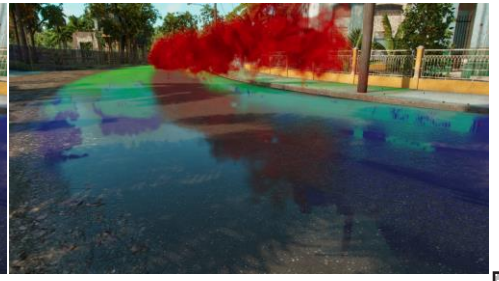

YARA
15SG9P

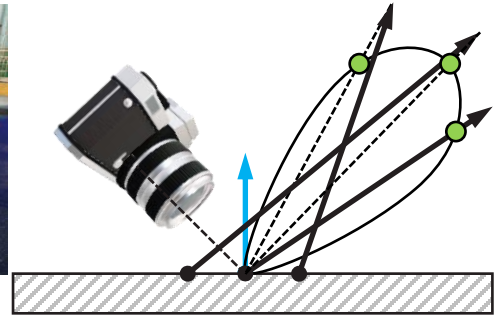
REFLECTIONS IN FAR CRY 6

Generate Rays	*SSLR	Ray Trace & Lighting	Particles Ray Trace	Integrate





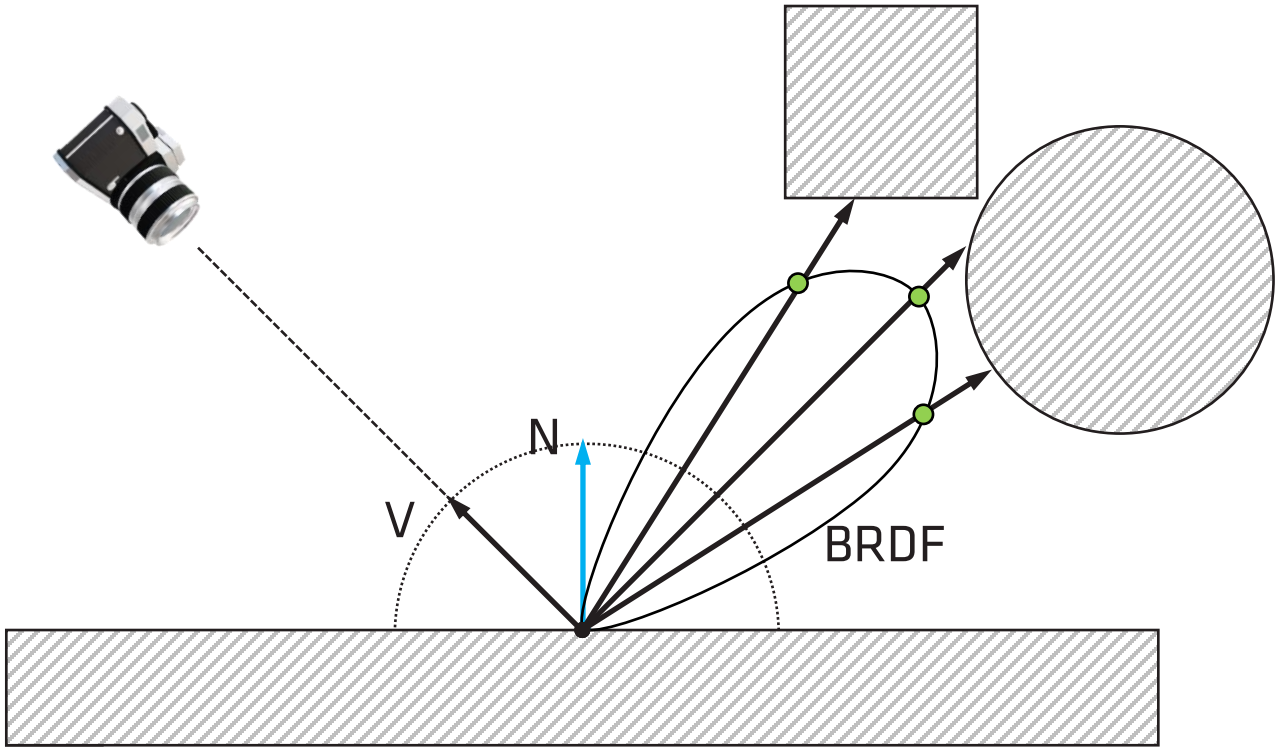




1. Generate Ray Buffer
2. SSLR
3. HW Ray trace
4. Particles HW Ray trace
5. Integrate results

*SSLR based on Tomasz Stachowiak “Stochastic Screen Space Reflections” at SIGGRAPH 2015

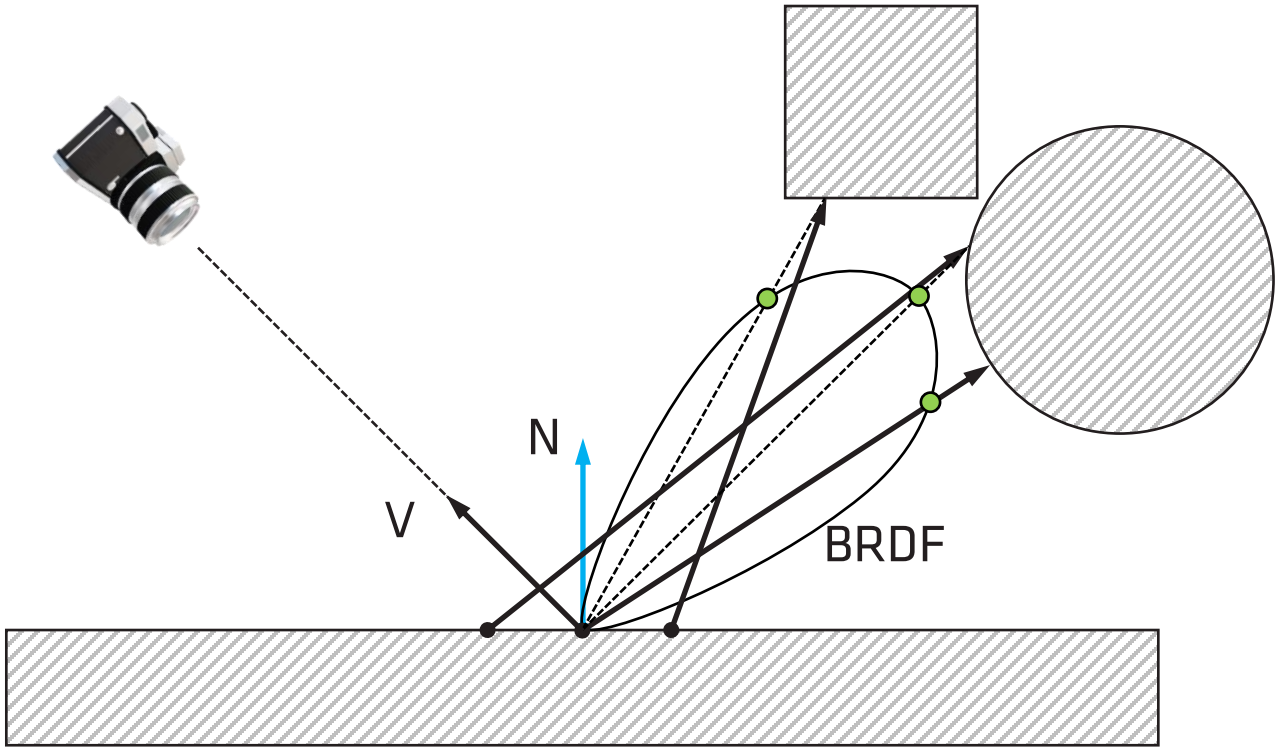
IMPORTANCE SAMPLING



- GGX BRDF to obtain reflection lobe
- Importance sampling reduction
- Pre-generated Hammersley sequence
- Store rays in buffer for later use

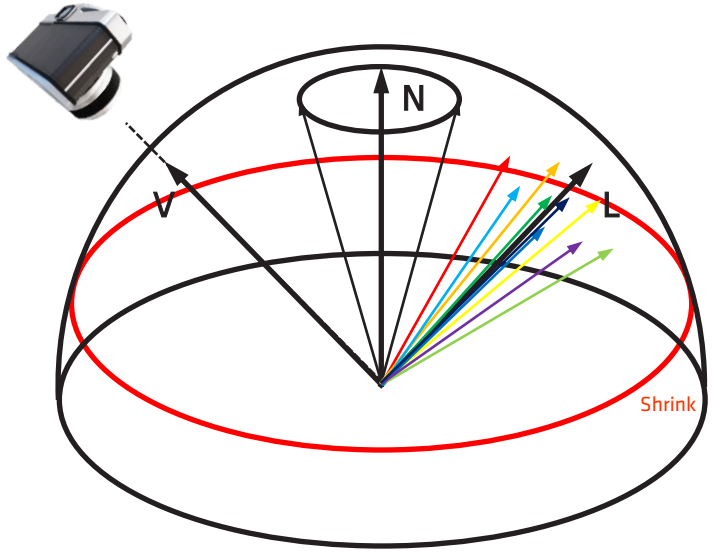
Problem:
Still too many rays per texel

SOLUTION: REUSE NEIGHBOURS

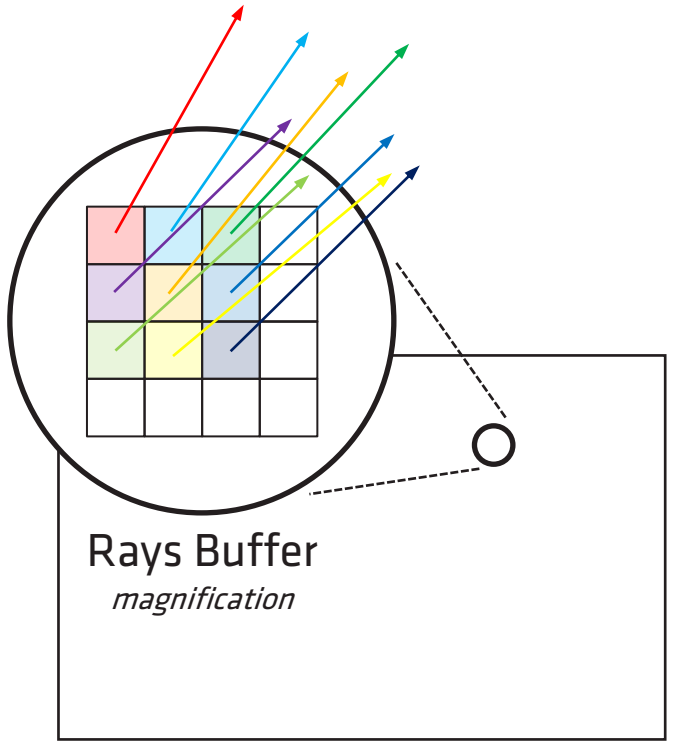


- **Wanted:** one ray per texel per frame
- Close neighbors use similar IS with Hammersley distribution
- Collect hit points from neighbors' rays
  - reuse these values in the BRDF integration

RAY BUFFER – TO BE USED LATER



Shrink Hemisphere



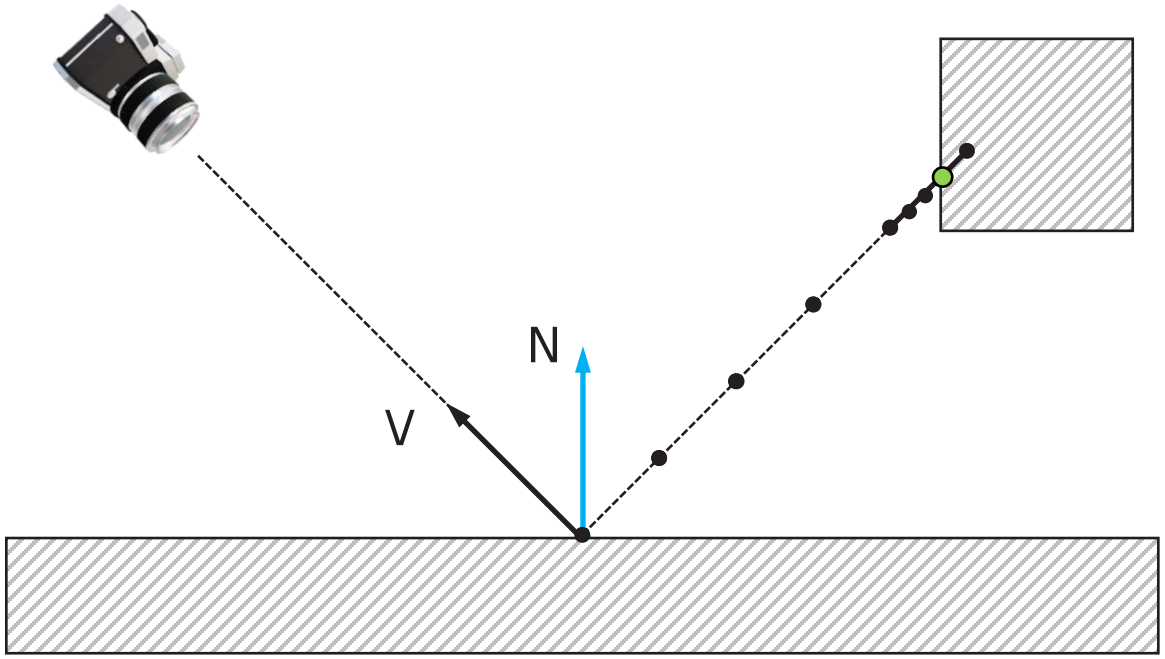
Store in half resolution ray buffer

SCREEN SPACE LOCAL REFLECTIONS

- SS trace faster than HW ray tracing
- SSLR is fallback while BVH builds
- Green shows where we use SSLR

Problem:
SSLR found to be unstable with the limited trace steps

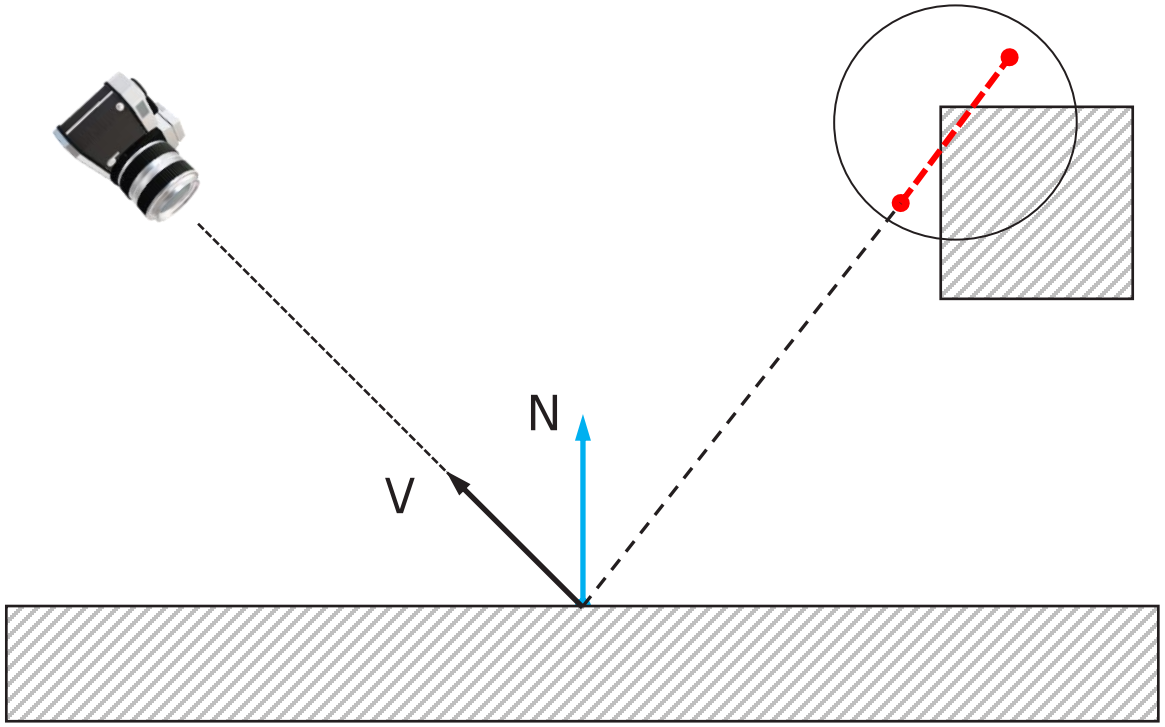
SOLUTION: LINEAR TRACE WITH REFINE



- Replaced Hierarchical Z with Linear Trace
- 64 large steps, 8 refine steps
- Stable with camera rotation
- Same performance as Hierarchical Z

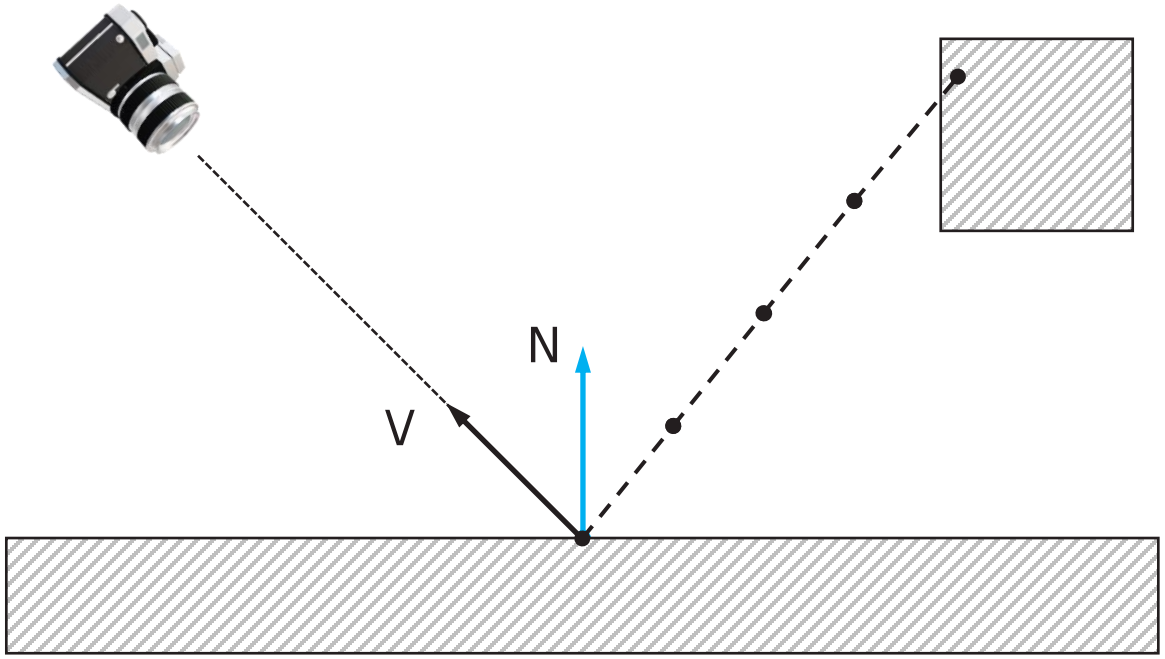
Problem:
Edges were being missed

SSLR LINEAR TRACE EDGE CASES



- The larger linear traces could step over edges
- No geometry detected: edges are lost

SOLUTION: RANDOM RAY OFFSETS



- Random ray start offsets
- Edges more stable

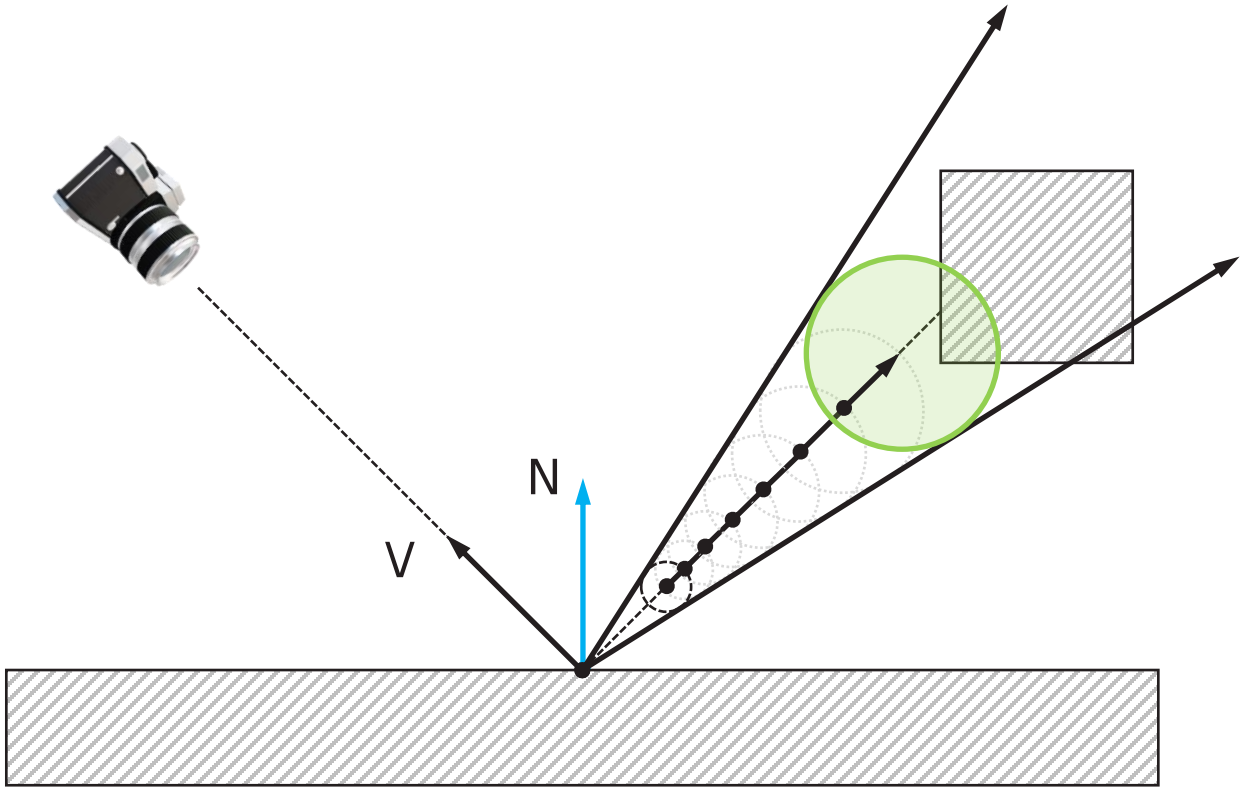
Wanted:
Better performance for glossy reflections

SS TILE CLASSIFICATIONS

Generate Rays	SSLR	Ray Trace & Lighting	Particles Trace	Integrate

- Smooth surfaces like chrome need higher precision
- **Fast trace** – **green** tiles
- **High precision trace** – **red** tiles

SS FAST CONE TRACE



- Cone angle varies by roughness
- Select the mip of the depth map according to number of pixels in the circle

Performant glossy reflection

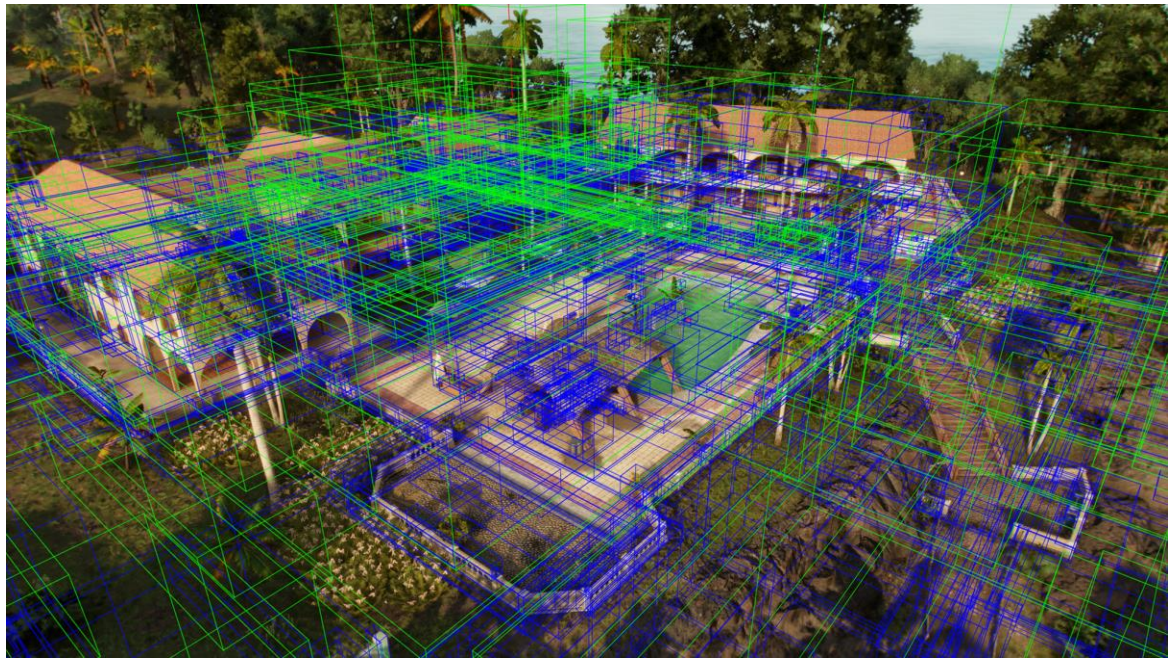
HW RAY TRACE ACCELERATION STRUCTURES

How to divide up all this space?

- BVH is the industry standard

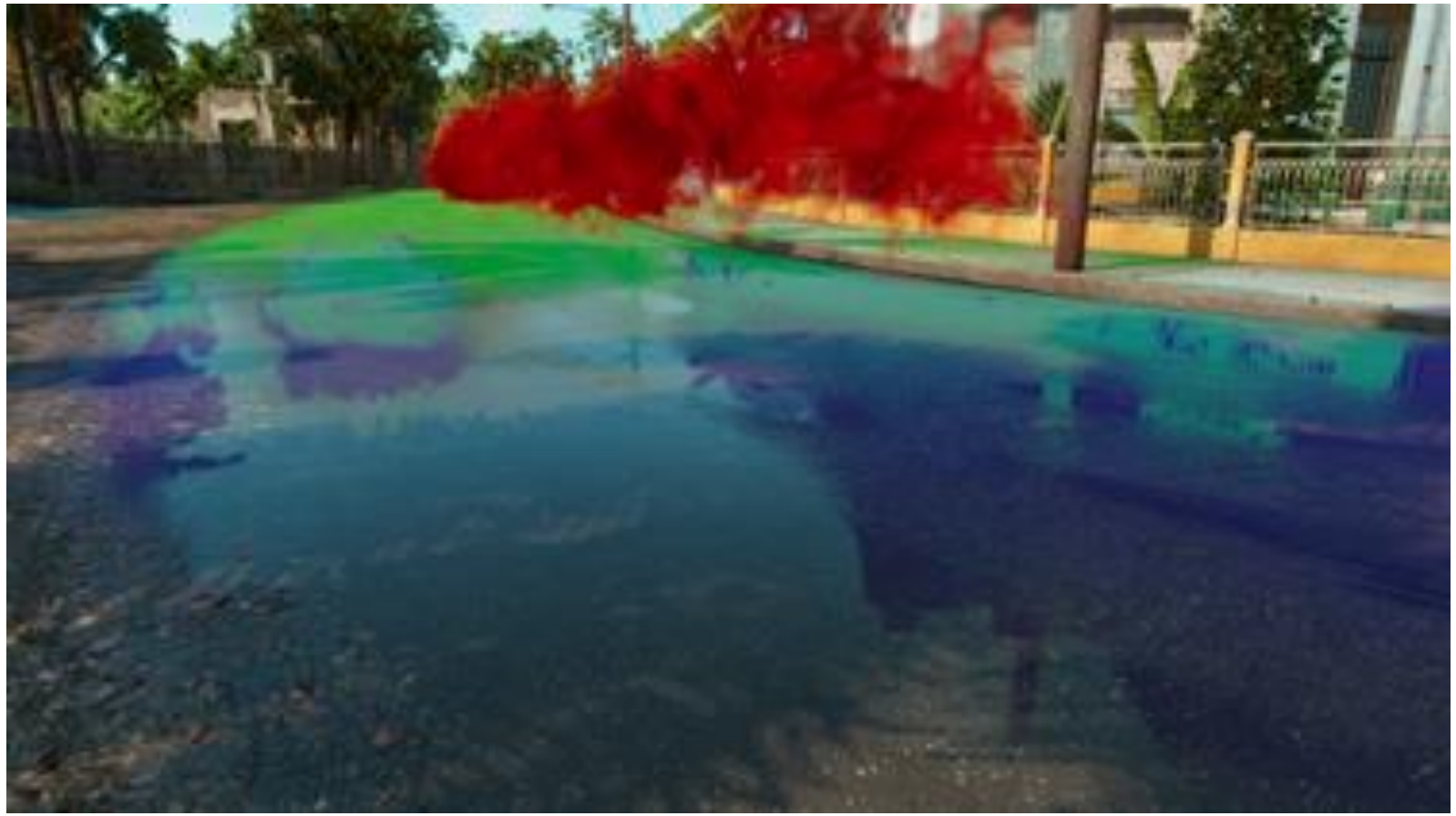
Traversing the BVH

- TLAS and BLAS



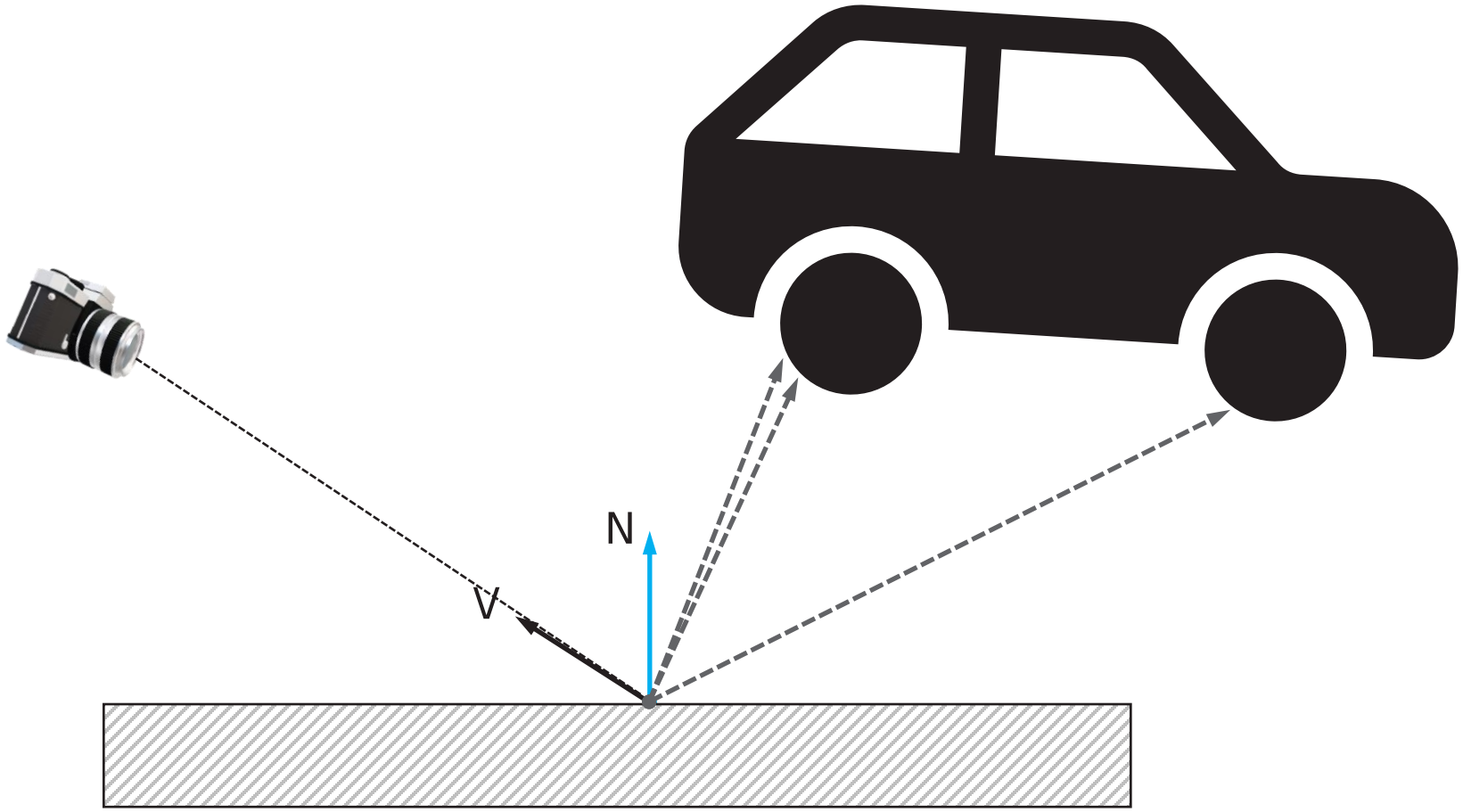
HW RAY TRACING ONLY IN NECESSARY PLACES

Generate Rays	SSLR	Ray Trace & Lighting	Particles Trace	Integrate



- **Far distances**: Only SSLR
  - BVH stops far from player
- **Close distances**: Only HW RT
  - Likely to be offscreen reflections
- **Mid distances**: it depends…

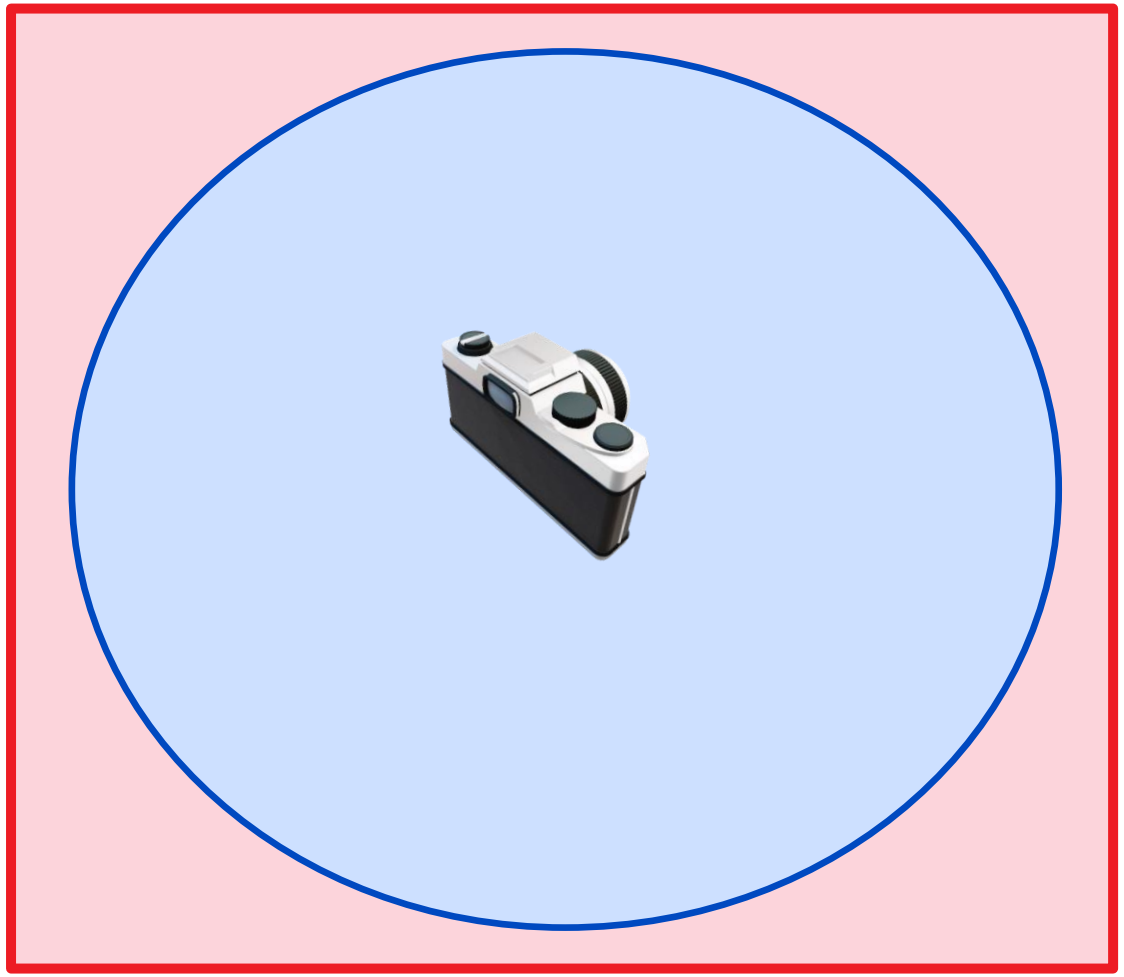
REFLECTING WITH CONFIDENCE: SSLR



- Changes in neighboring pixel depth are normally small
- Large depth changes in neighbouring pixels
  - Missing offscreen details
  - Low SSLR confidence

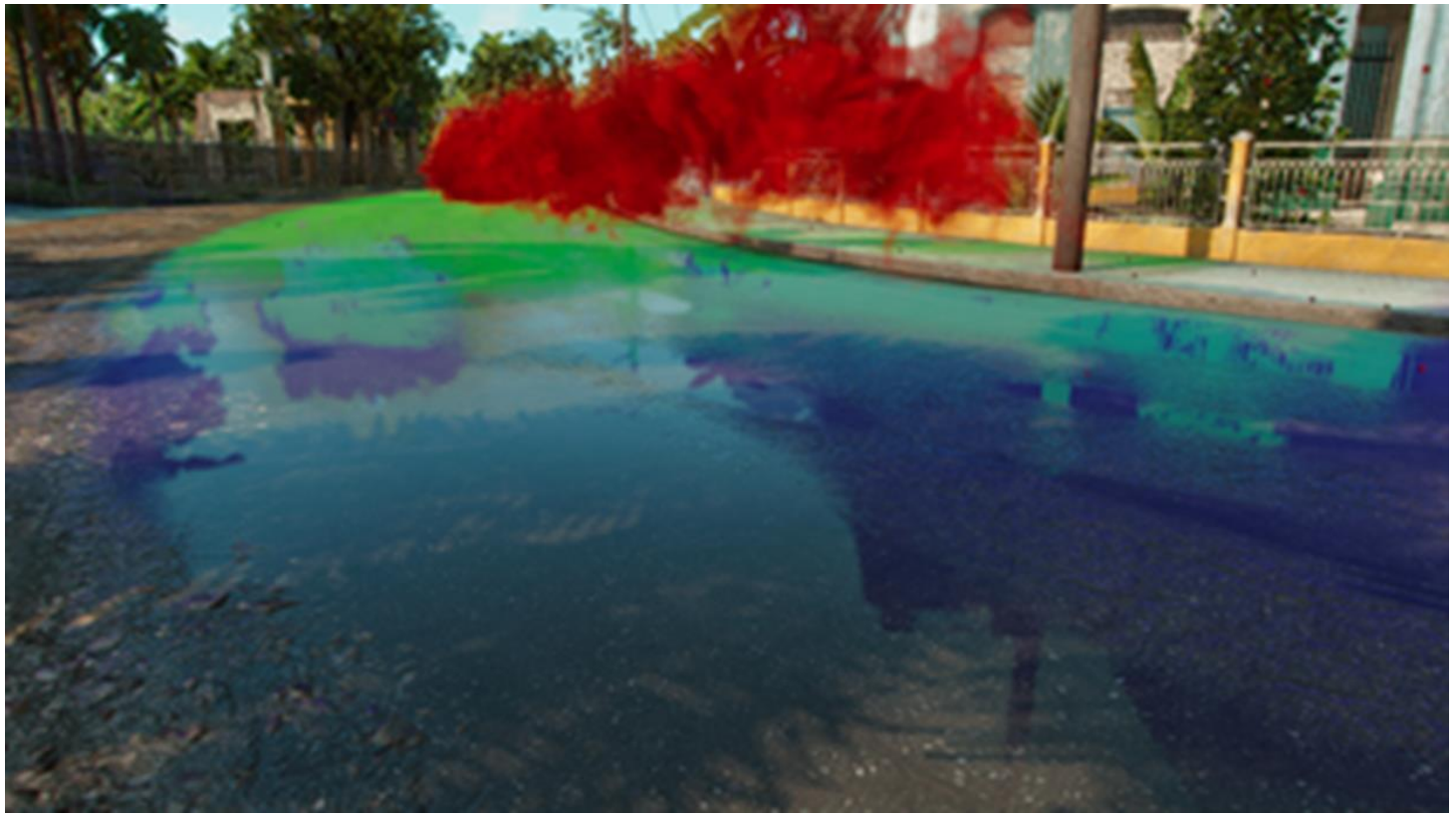
REFLECTING WITH CONFIDENCE: HW RAY TRACING

Generate Rays	SSLR	Ray Trace & Lighting	Particles Trace	Integrate



- HW Ray trace confidence 100% close to player
- Confidence gradually decreases to 0% at edge of BVH
  - Prevents popping

REFLECTING WITH CONFIDENCE: HYBRID

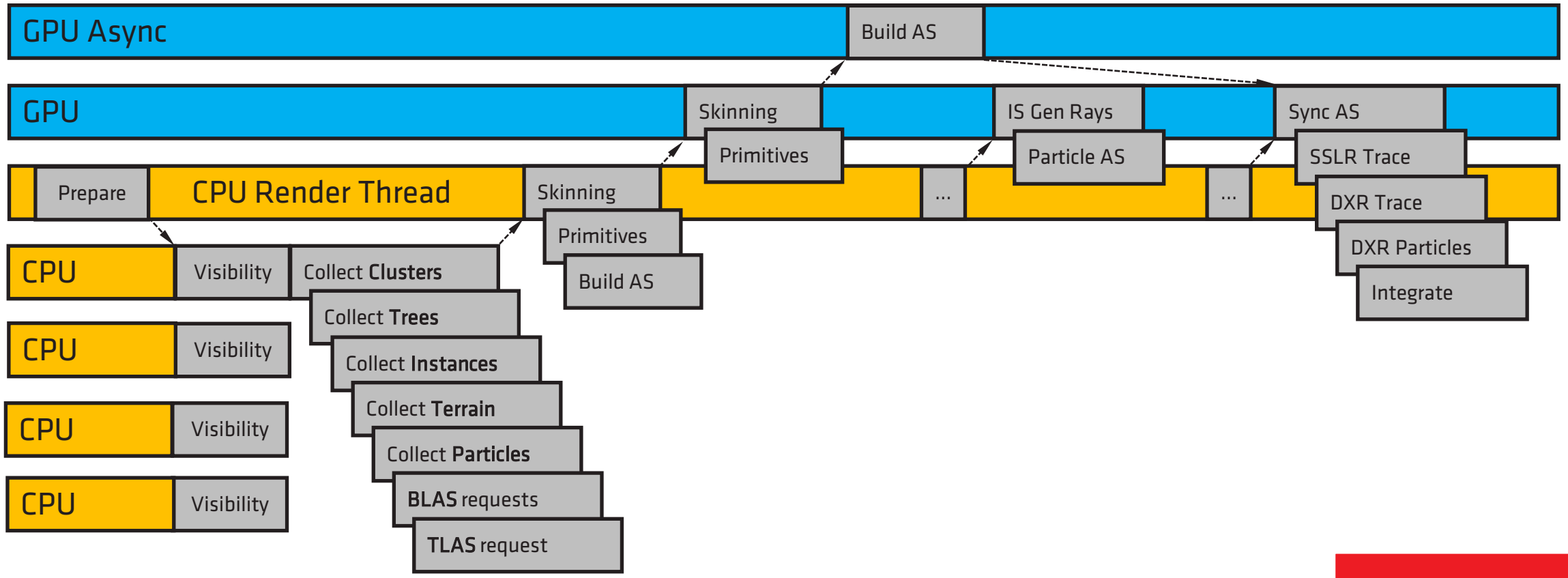


- **Green** is full confidence SSLR
  - Use only SSLR
- **Teal** is mid confidence SSLR
  - Hybrid: use trace with highest confidence
- **Blue** is no confidence SSLR
  - Use only HW RT
- **Other reflections:** no confidence in SSLR nor HW RT
  - Use environment map reflections

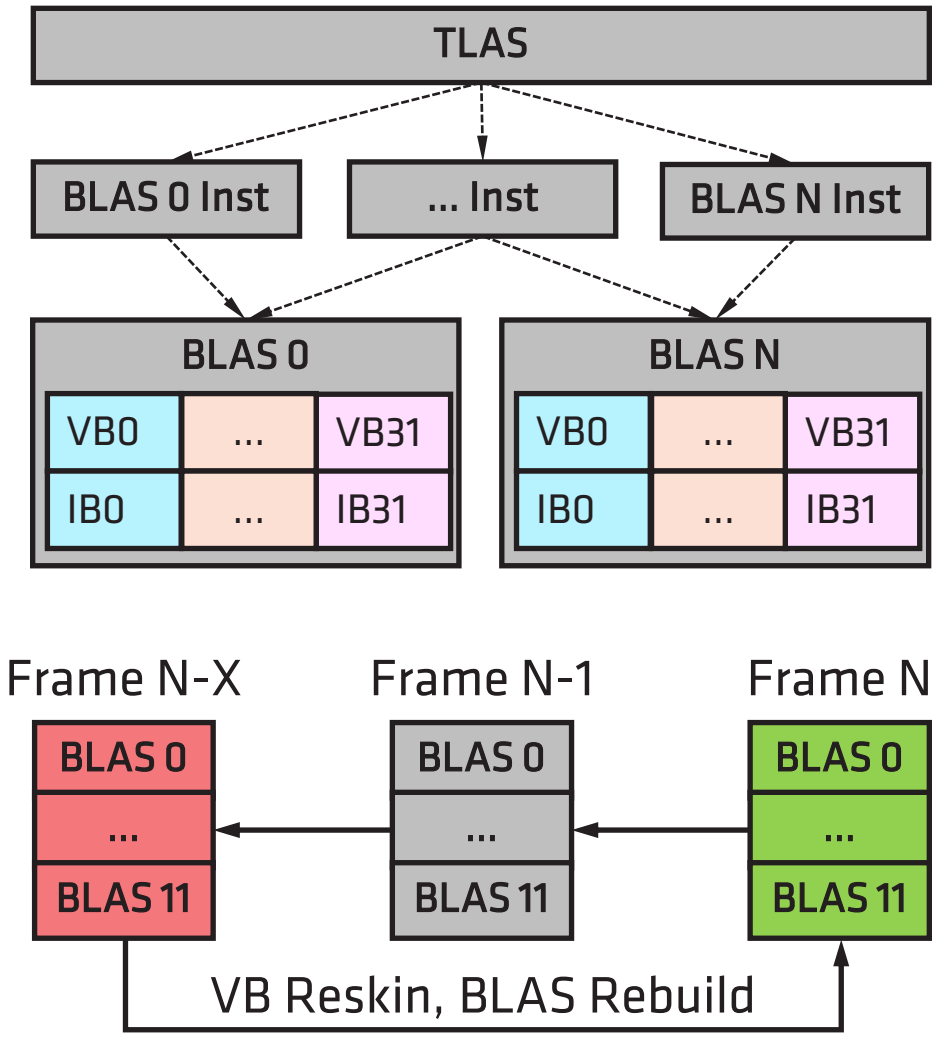
See “Simulating Tropical Weather in Far Cry 6” for more details on environment maps

PEEK BEHIND THE CURTAIN

RAYTRACE REFLECTION GRAPH

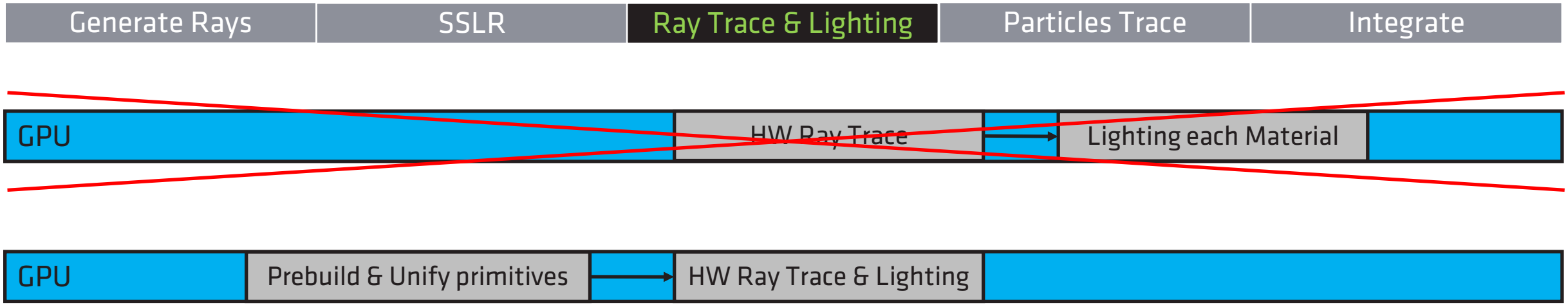


BVH TREE



- One BLAS per Object (or Material)
  - Tip: Per object is preferred
- Static BLASes are built once
- Skinned BLASes need endless rebuild loop
- First person player character rebuilds each frame
  - Player weapon & attachments as well
- 16 BLASes rebuild per frame (static/skinned)
  - Max 12 skinned

HW RAY TRACING & LIGHTING TOGETHER



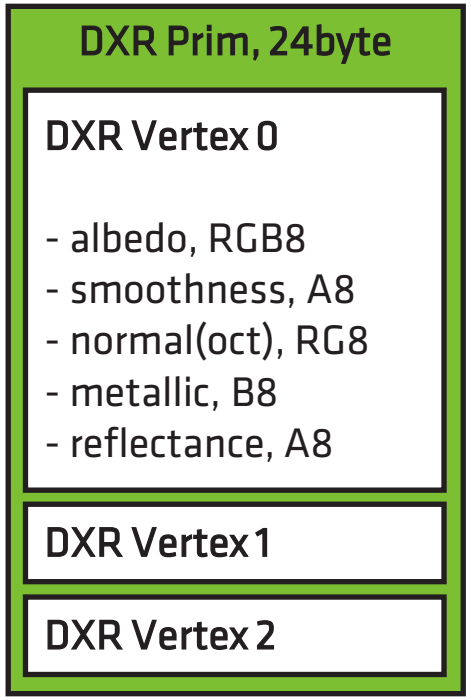
- Lighting shader and ray trace shader in one
- Parameters for lighting unified and stored per vertex
- Each material has specific shader to map its lighting to the unified lighting parameters
- Object material parsed on BLAS creation stage
  - Performance hit of shader variations is a one-time cost during BLAS creation

LIGHTING IN THE HW RAY TRACING PIPELINE

Generate Rays	SSLR	Ray Trace & Lighting	Particles Trace	Integrate

- **Graphics pipeline**: interpolated values provided by the Rasterizer to Pixel Shaders
- **Ray tracing pipeline**: interpolation done in the lighting shader
  - Unified lighting parameters stored in DXR primitives

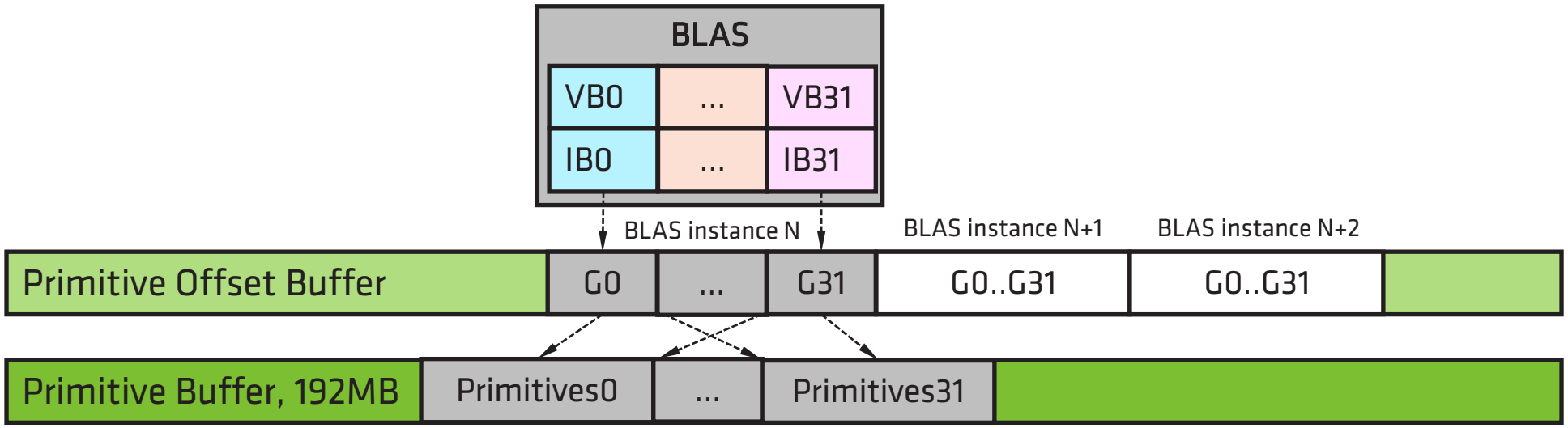
PRIMITIVE DATA



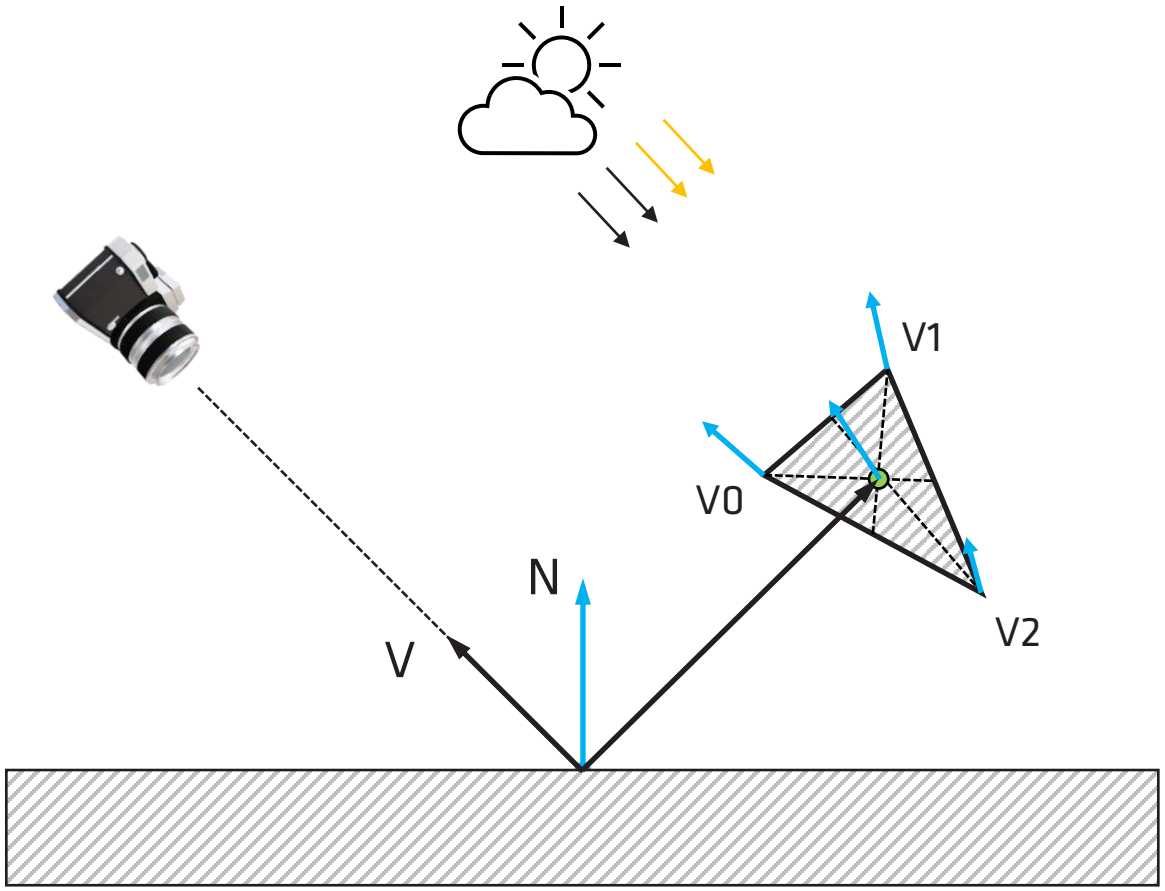
- Unwrap Vertex and Index buffers to Primitives before BLAS created
  - Causes duplication but ensures better coherency for interpolation
- Index buffer not needed for lighting computations
  - Removes an additional indirection
- Vertex data is packed

PRIMITIVE POOL

- Primitive Buffer – holds all primitive information from every BLAS
- Primitive Offset Buffer – redirection to primitives for BLAS instances in TLAS
  - Required to locate primitive data using the PrimitiveIndex supplied by DXR
- Fixed Geometry count per BLAS – max 32



LIGHTING THE PRIMITIVE



- Interpolate primitive vertex data using barycentric coordinates
- Contributors to lighting the primitive
  - Direct Lighting(Sun)
  - Indirect Lighting(GI, Env Maps)
  - Shadows

Future Work:
support dynamic lights

HW RAY TRACE BARYCENTRIC COORDINATES

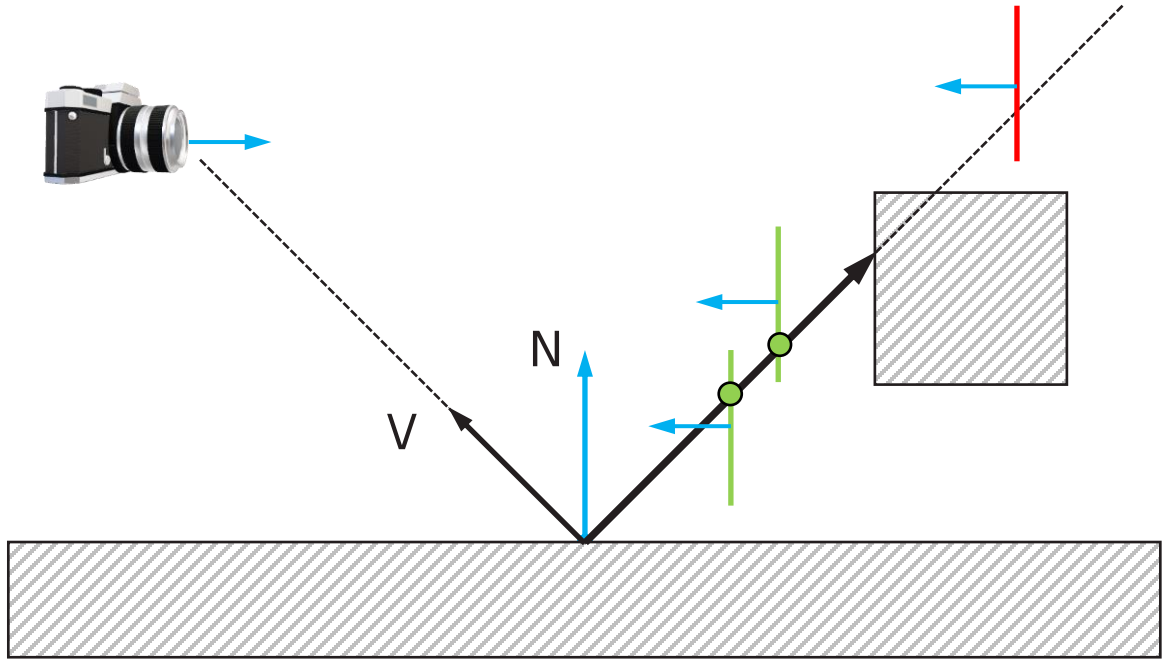
HW RAY TRACE PRIMITIVES LIGHTING

PARTICLES IN FAR CRY 6

- Far Cry games are known for
  - explosions
  - dynamic systemic fire
- New for Far Cry 6: Poison
- Particle reflections important for Far Cry 6

Problem:
Particles are just 2D billboards

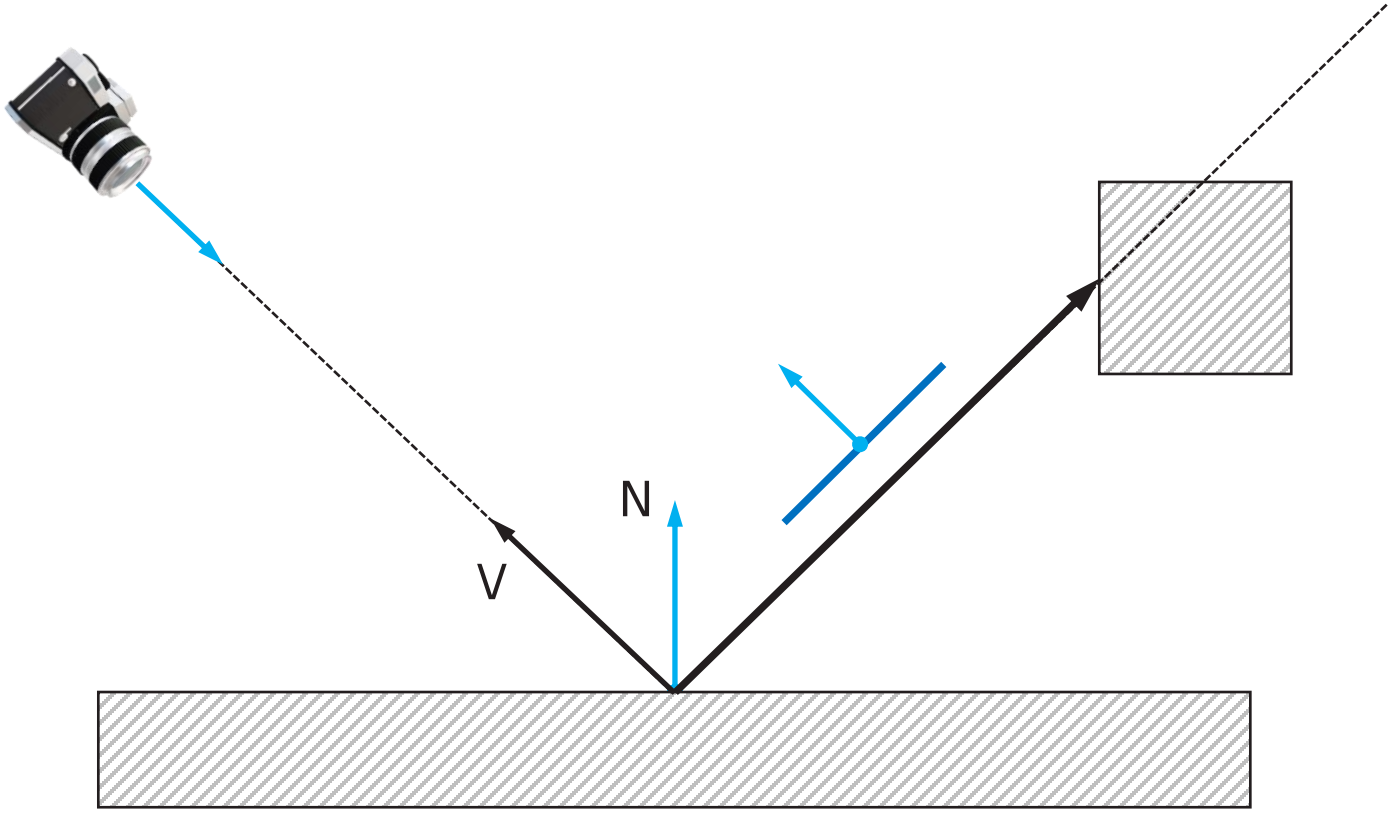
PARTICLES HARDWARE RAY TRACE



- Reuse ray length from SSLR/HW RT
  - clips invisible particles
- BLAS Instance per Particle
  - 3 BLAS instances in this diagram
- Collect all particles along the ray for blending

Performance Tip:
Build separate TLAS for particles

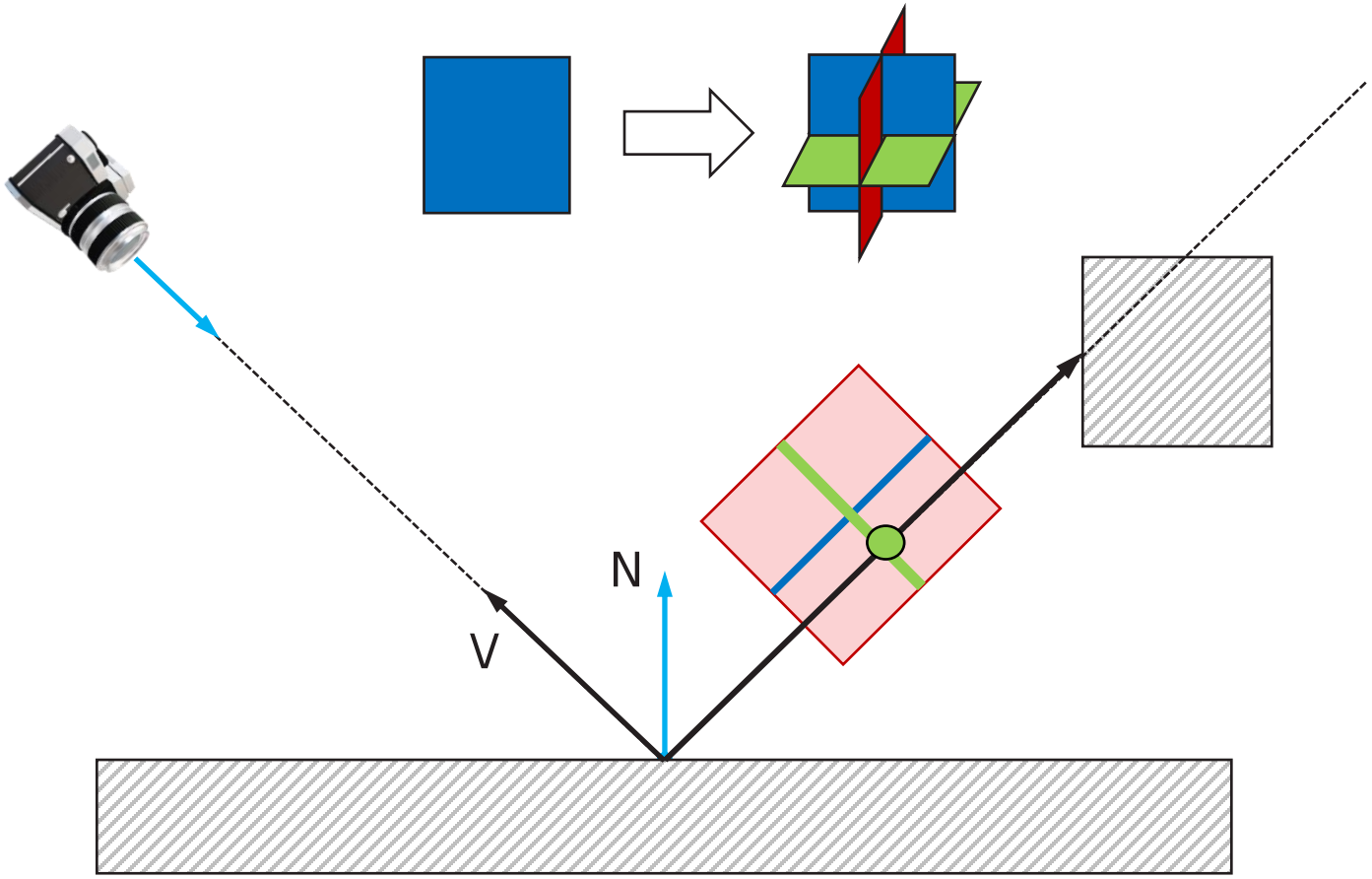
PARTICLES: THE CASE OF THE MISSING SPRITES



2D sprites are oriented to be visible to camera

Problem:
Sprites are often not visible to the reflection ray

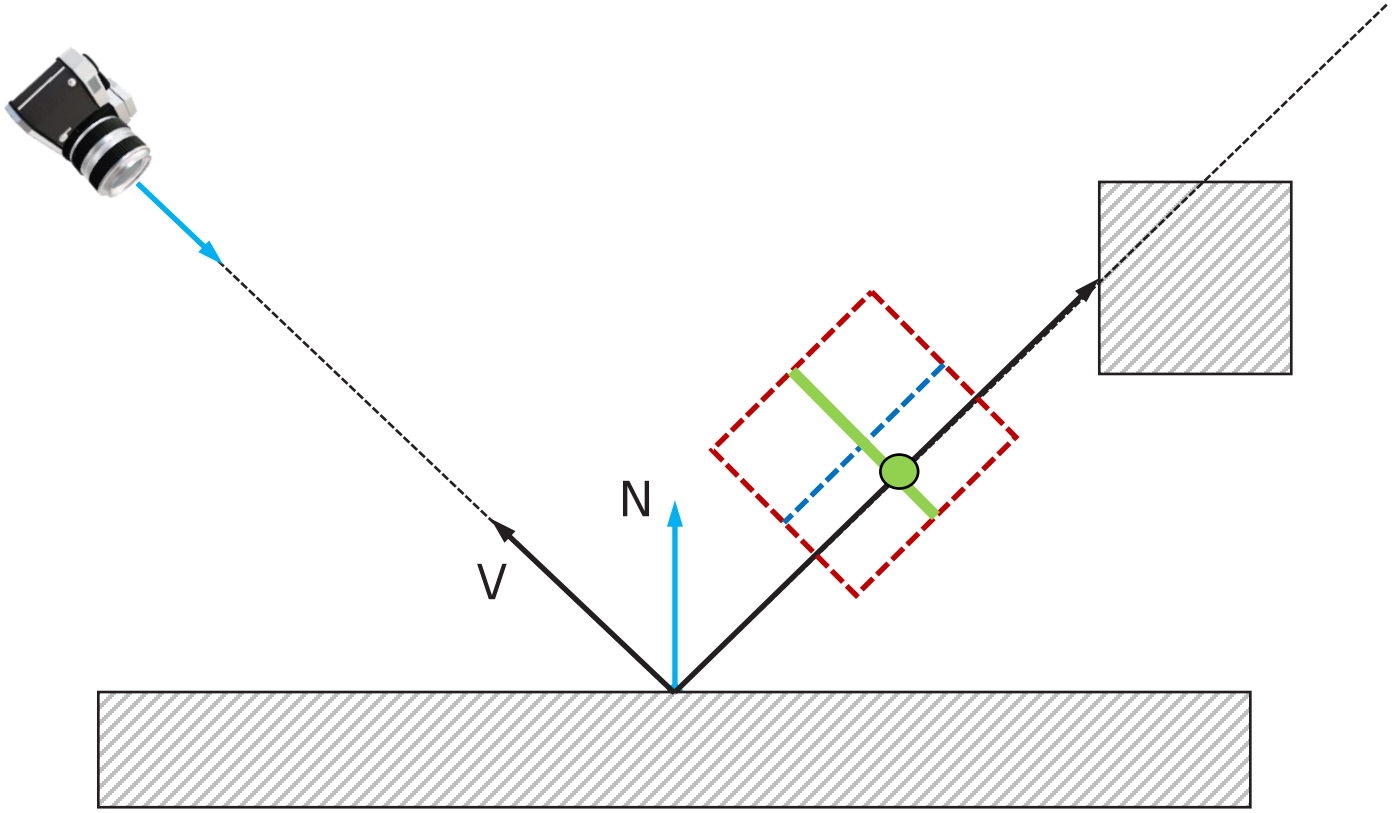
SOLUTION: EXTRA QUADS



Extra quads aligned by local space coordinate system planes(X,Y) for each particle

Problem:
expensive to test intersection on all the quads

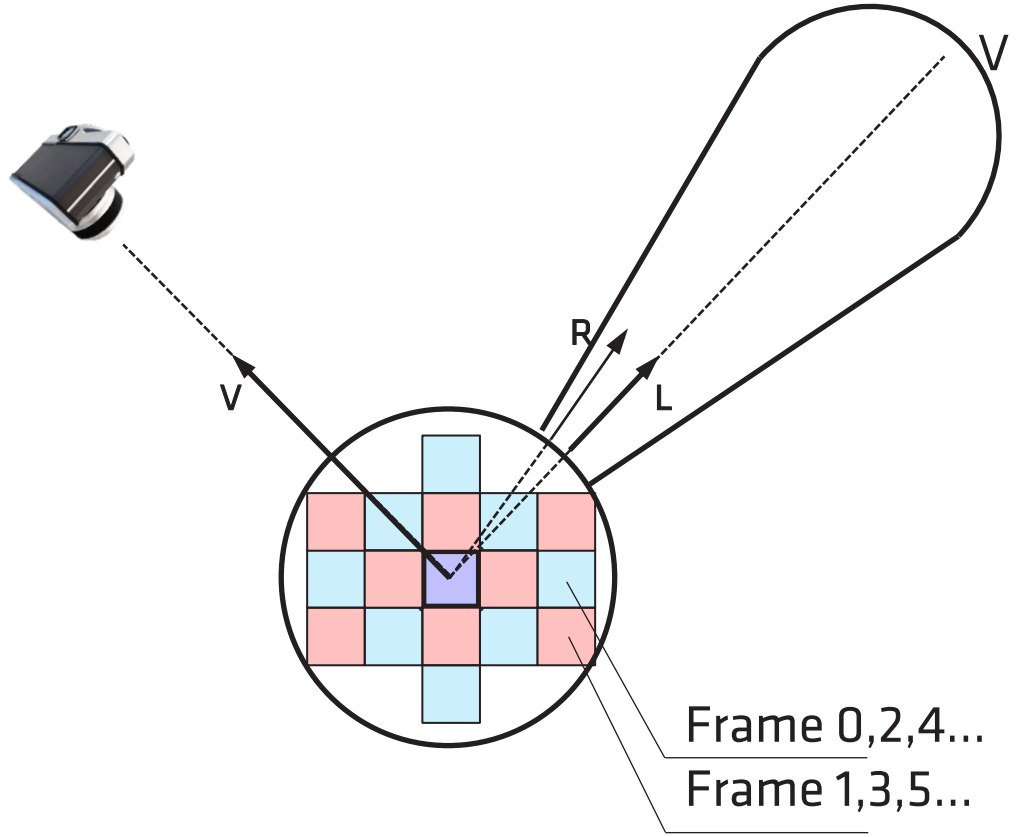
SOLUTION: MASK QUADS BY VIEW SPACE AXIS



- Quads masked by view space axis direction in BVH
- Select quad perpendicular to ray
- Simplified lighting
  - No Env Map
  - Low Quality Shadows


SSLR PARTICLES
RAY TRACED PARTICLES

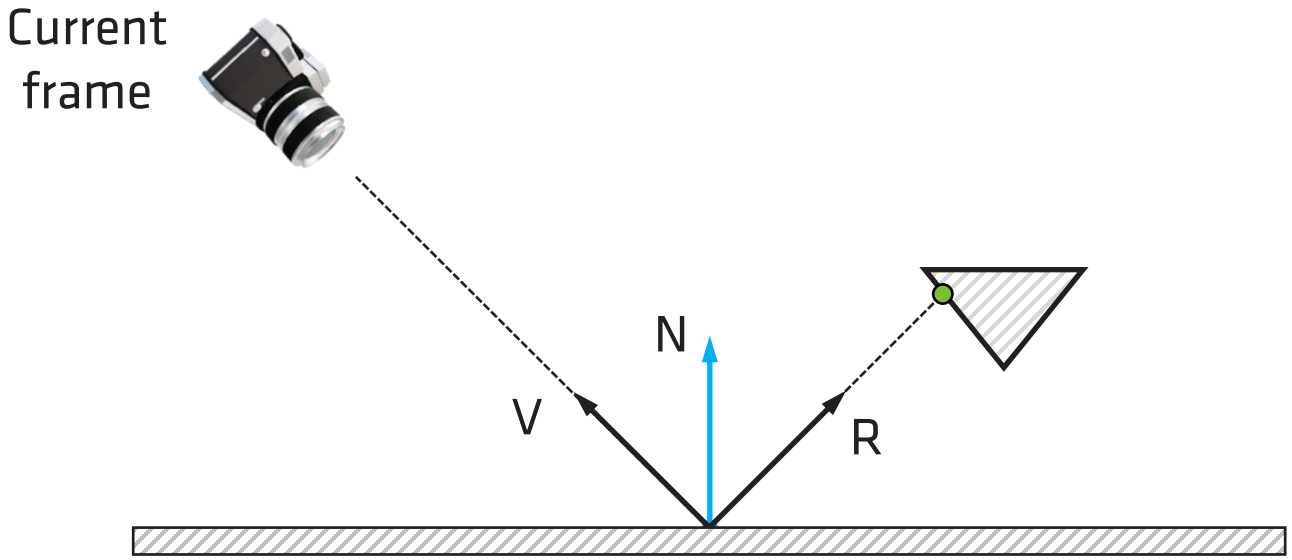
INTEGRATE



- Reuse neighbors' color results to integrate rays from multiple directions
- 9 neighbors per frame
- Central pixel required for mirror reflections
  - used every frame
- Weighted by BRDF coefficient

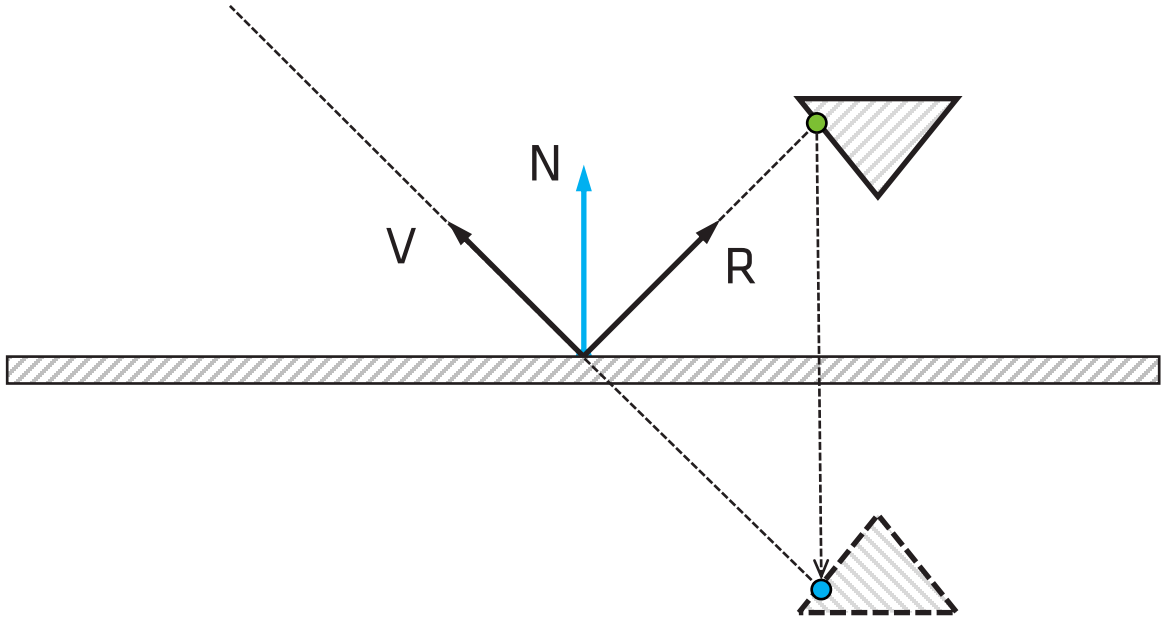
REFLECTION MOTION

Current frame



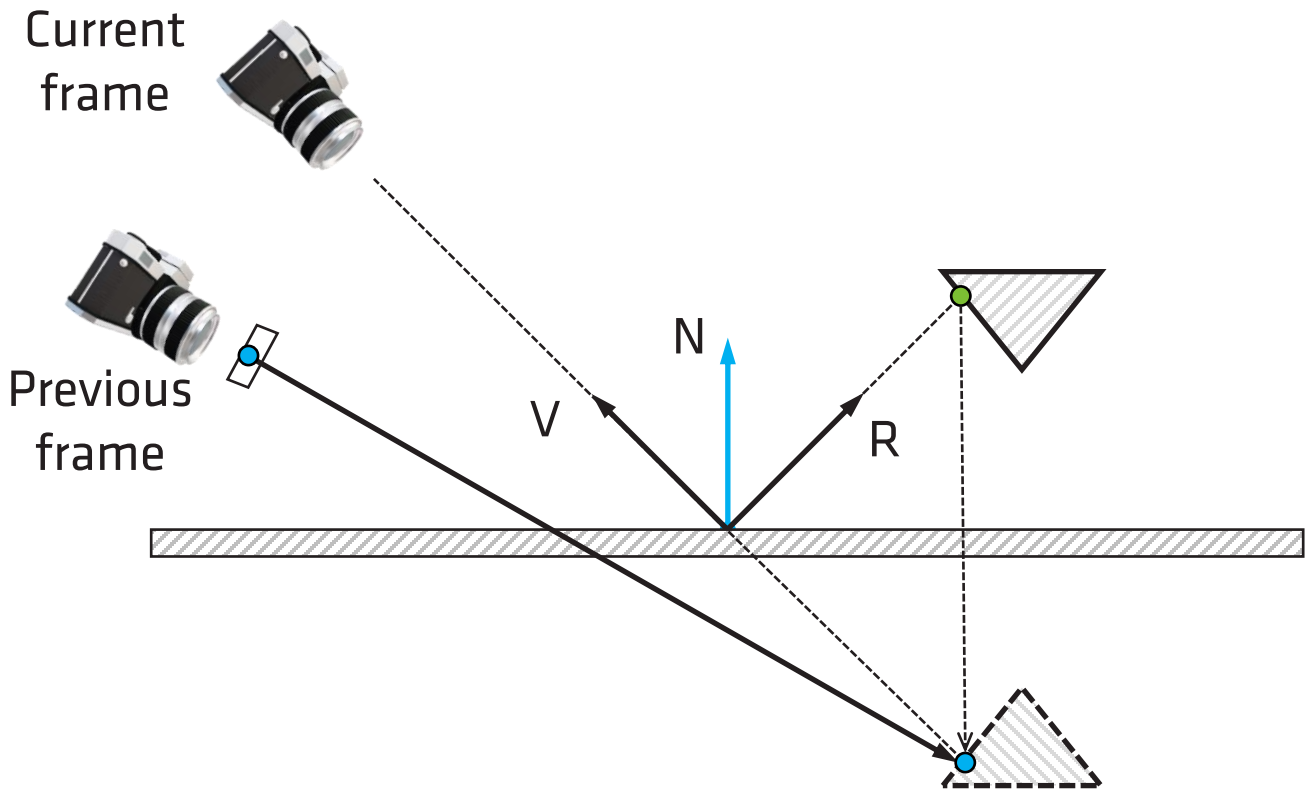
REFLECTION MOTION : REPROJECTION

Current frame



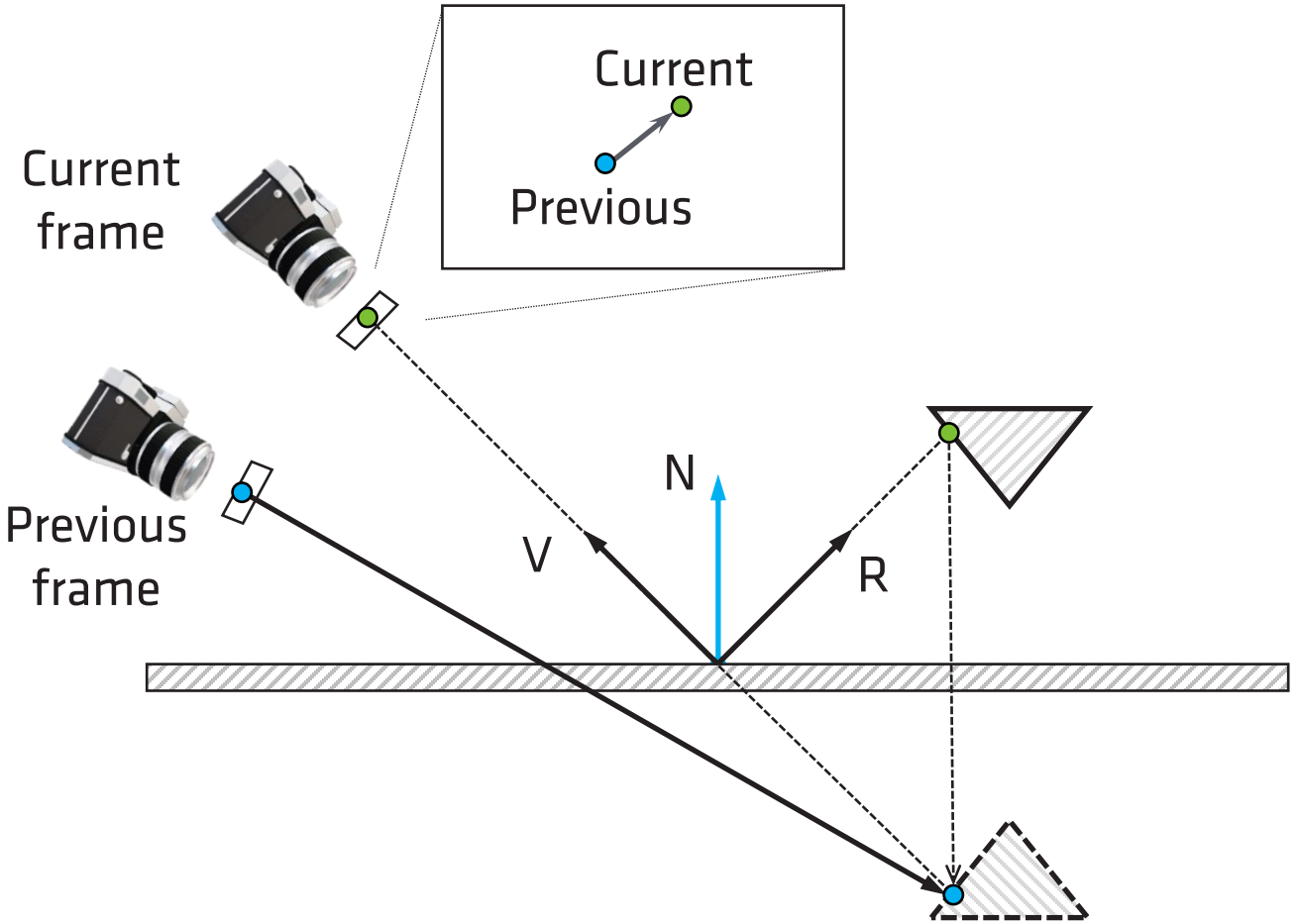
- Usually calculated with reprojection
  - Reproject reflection result through surface plane and get position in World Space

REFLECTION MOTION: PREVIOUS FRAME



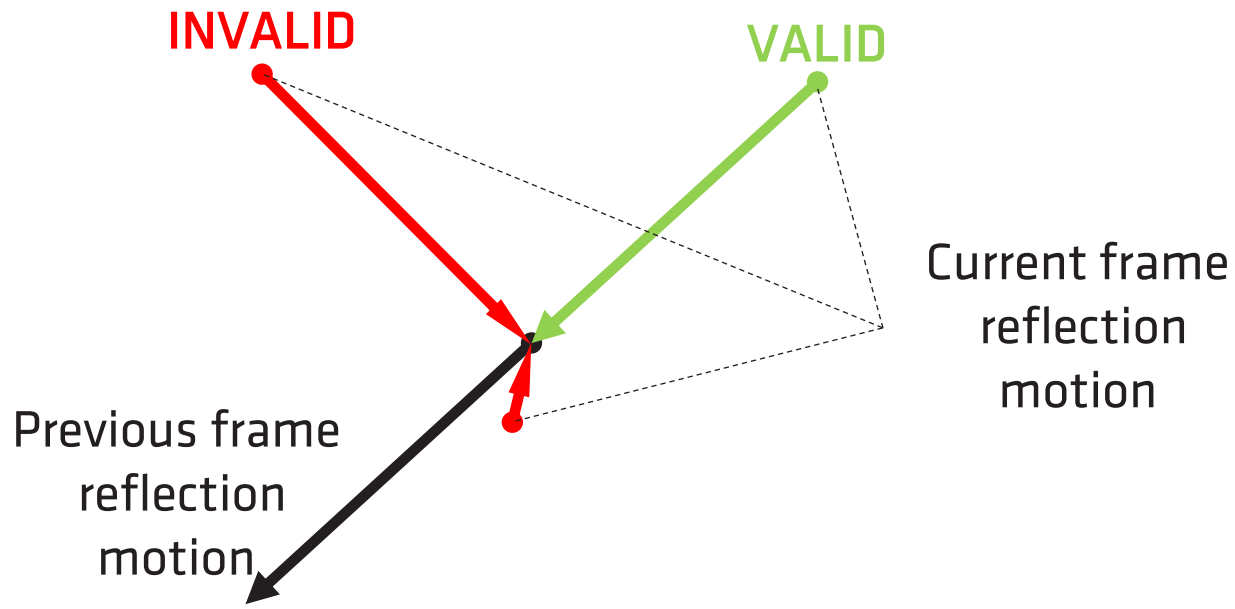
- Usually calculated with reprojection
  - Reproject reflection result through surface plane and get position in World Space
  - Put World Space position into Screen Space using previous frame's camera

REFLECTION MOTION: BACK IN SCREEN SPACE



- Usually calculated with reprojection
  - Reproject reflection result through surface plane and get position in World Space
  - Put World Space position into Screen Space using previous frame's camera
  - Take the difference in Screen Space
- For objects that move with camera
  - Use GBuffer motion

TEMPORAL ACCUMULATION



- Unlike TAA: history is not rejected based on motion vector length
  - Ignoring large reflection motion vectors can result in specular explosions on rough surfaces
- History validated by reflection motion derivative/differential
  - Discard incoherent reflection motion


SSLR
EL ESTE

SSLR/HW RAY TRACE MASK


HYBRID RAY TRACED REFLECTIONS
EL ESTE

OPTIMIZING RAYTRACING IN FAR CRY 6

GDC

AMD GPUOpen

REFLECTIONS AT THE AMD TEAM

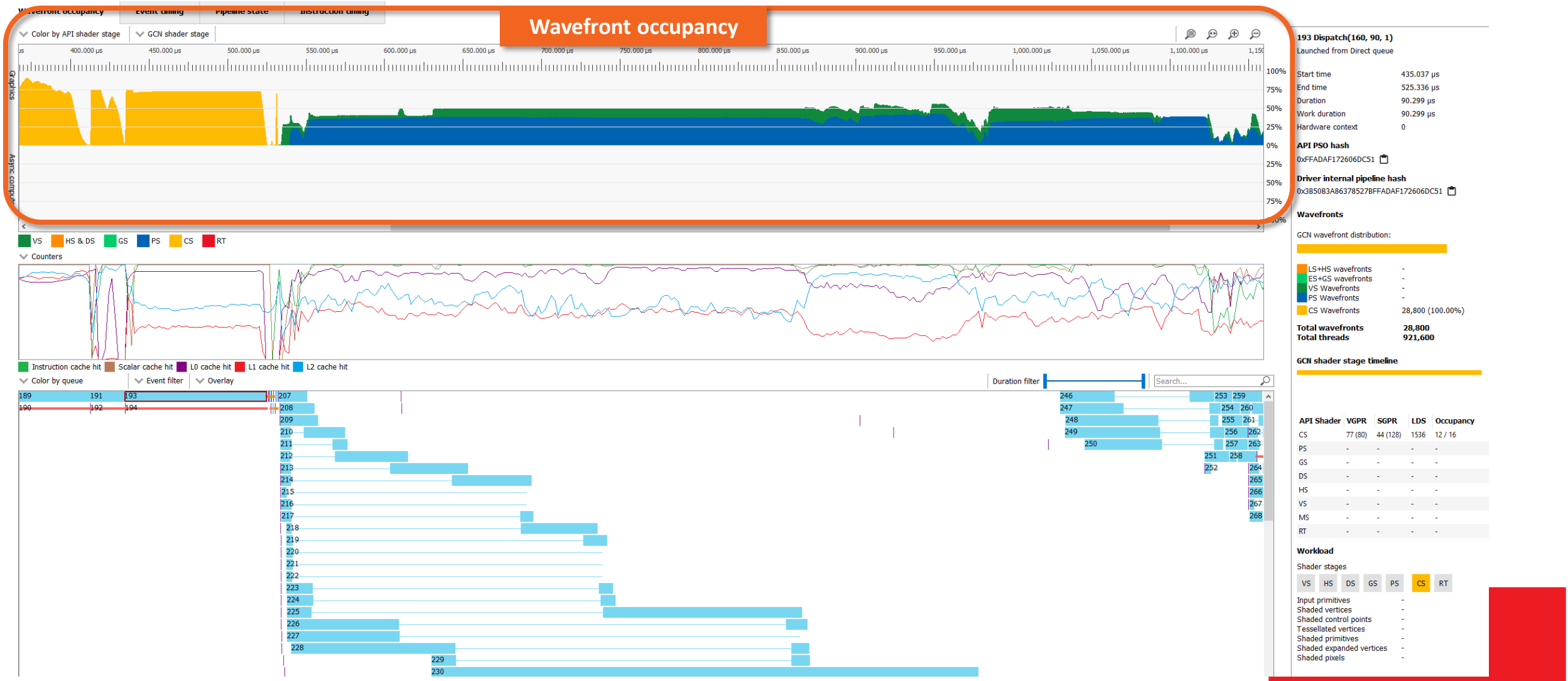
- **Ihor Szlachtycz (presenter)**
  - AMD GPU Dev Tech for Far Cry 6
- **Zhuo Chen**
  - AMD GPU Dev Tech for Far Cry 6
- **John Hartwig**
  - AMD CPU Dev Tech for Far Cry 6

OPTIMIZATION OVERVIEW

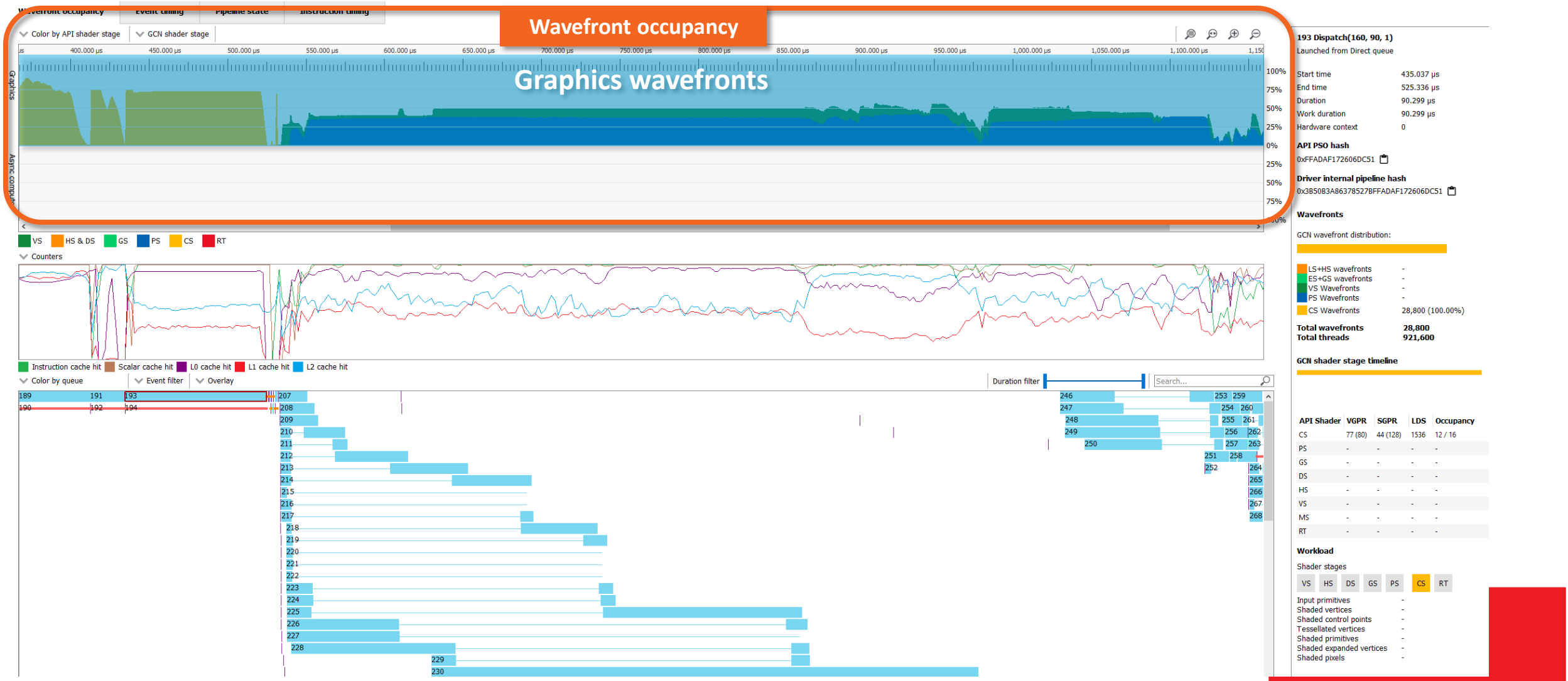
- We will be going over how Hybrid Reflections raytracing was optimized for Far Cry 6
  - Shader table management
  - Hit shader design
  - BVH building
- All performance captures for Far Cry 6 were done on a Radeon 6800XT GPU with a Ryzen 3970X Threadripper
- But first, we will give a quick overview of Radeon GPU Profiler (RGP)

RADEON GPU PROFILER (RGP) OVERVIEW

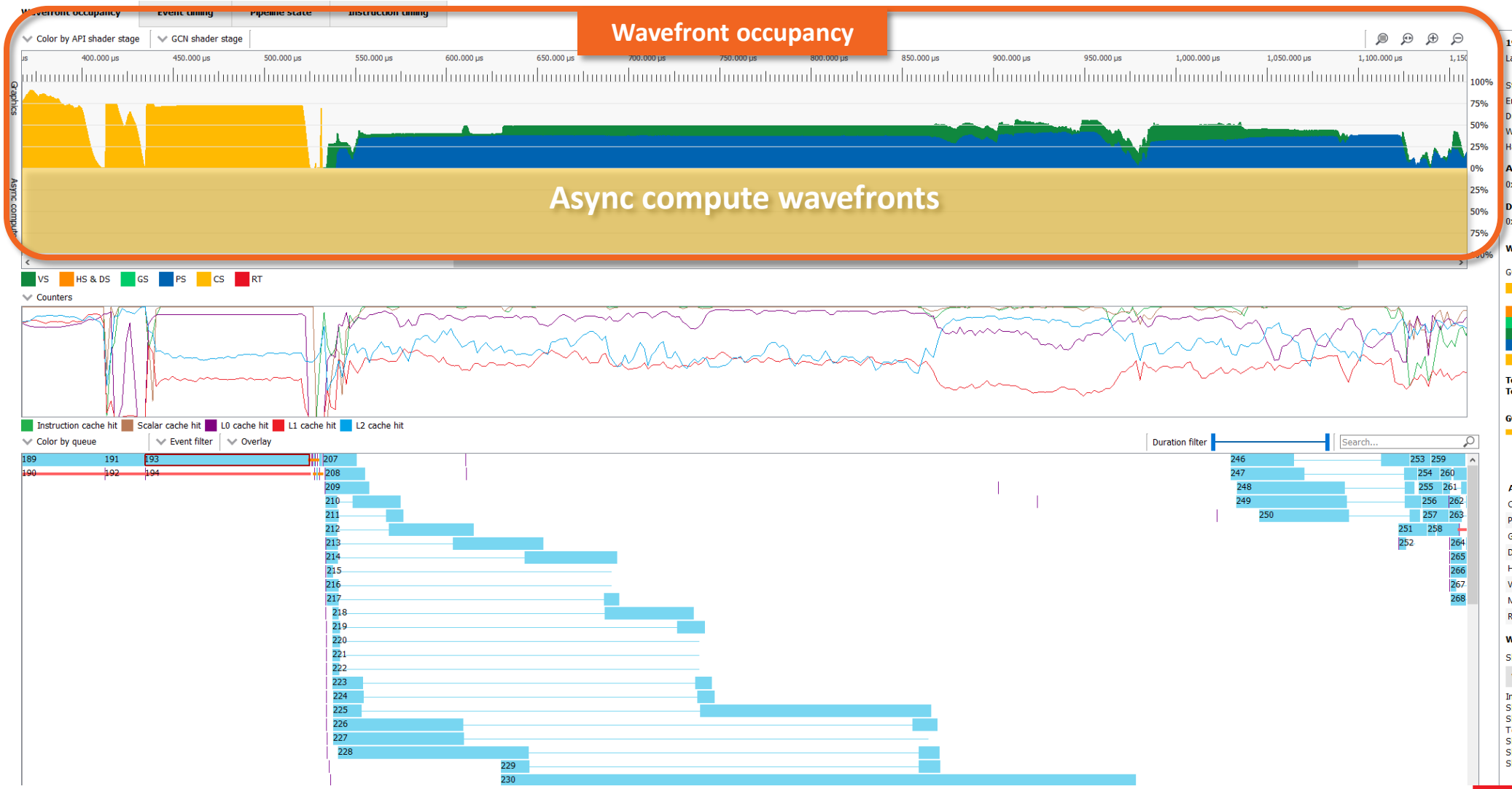
RADEON GPU PROFILER (RGP) OVERVIEW



RADEON GPU PROFILER (RGP) OVERVIEW

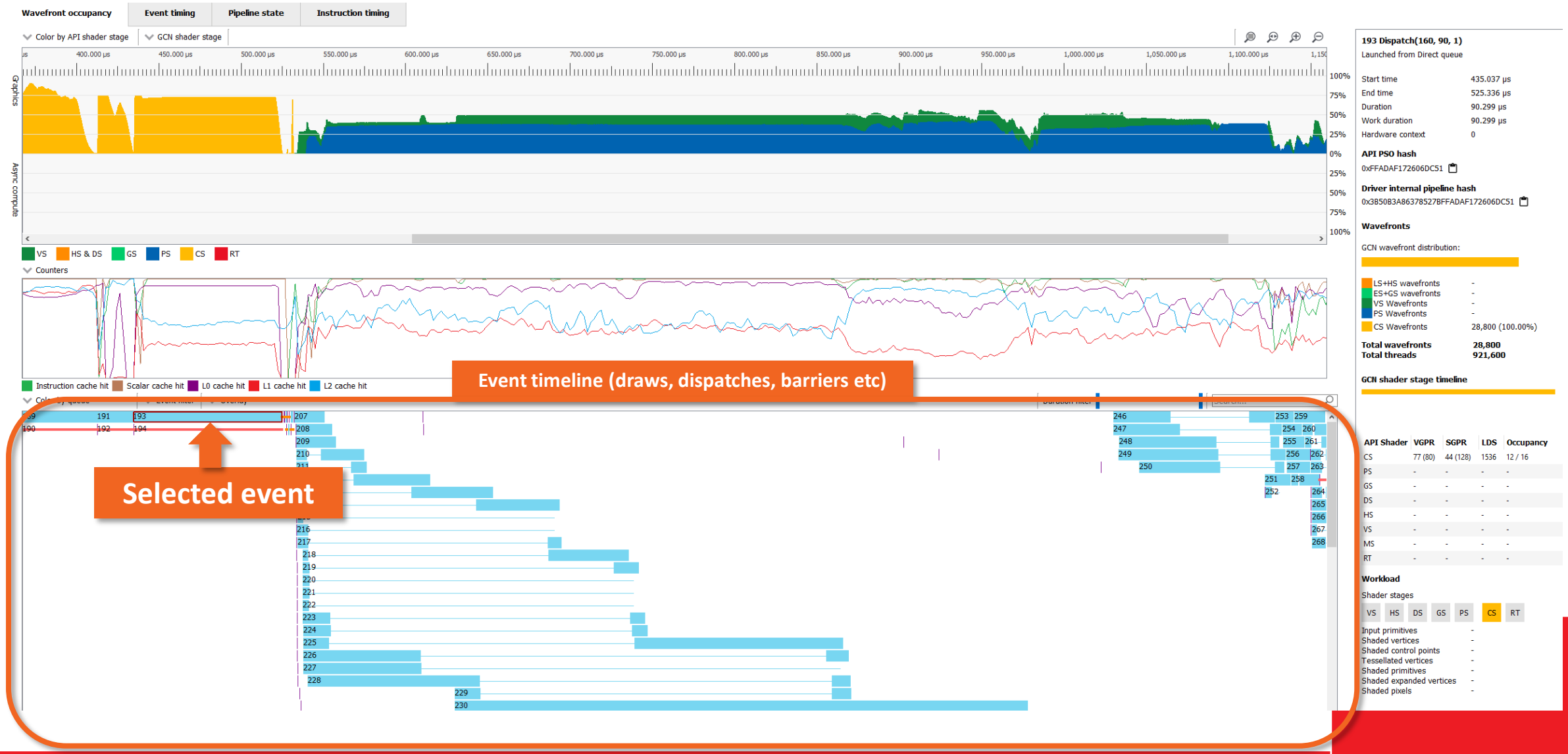


RADEON GPU PROFILER (RGP) OVERVIEW



RADEON GPU PROFILER (RGP) OVERVIEW

RADEON GPU PROFILER (RGP) OVERVIEW



RADEON GPU PROFILER (RGP) OVERVIEW

Event details

193 Dispatch(160, 90, 1)
Launched from Direct queue

Start time	435.037 µs
End time	525.336 µs
Duration	90.299 µs
Work duration	90.299 µs
Hardware context	0

API PSO hash
0xFFADAF172606DC51

Driver internal pipeline hash
0x3B50B3A86378527BFFADAF172606DC51

Wavefronts

GCN wavefront distribution:

LS+HS wavefronts	-
ES+GS wavefronts	-
VS Wavefronts	-
PS Wavefronts	-
CS Wavefronts	28,800 (100.00%)
Total wavefronts	**28,800**
Total threads	**921,600**

GCN shader stage timeline

API Shader	VGPR	SGPR	LDS	Occupancy
CS	77 (80)	44 (128)	1536	12 / 16
PS	-	-	-	-
GS	-	-	-	-
DS	-	-	-	-
HS	-	-	-	-
VS	-	-	-	-
MS	-	-	-	-
RT	-	-	-	-

Workload

Shader stages

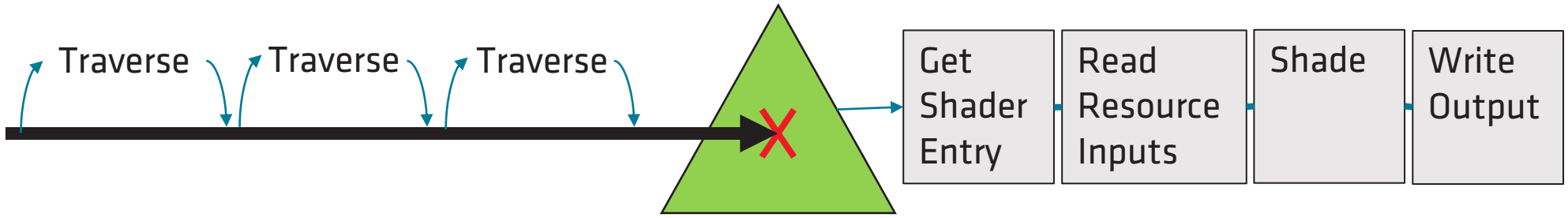
Input primitives	-
Shaded vertices	-
Shaded control points	-
Tessellated vertices	-
Shaded primitives	-
Shaded expanded vertices	-
Shaded pixels	-

GENERAL RAYTRACING PERFORMANCE

- Here we have an overview of what a RGP trace looks like with Hybrid RayTraced Reflections

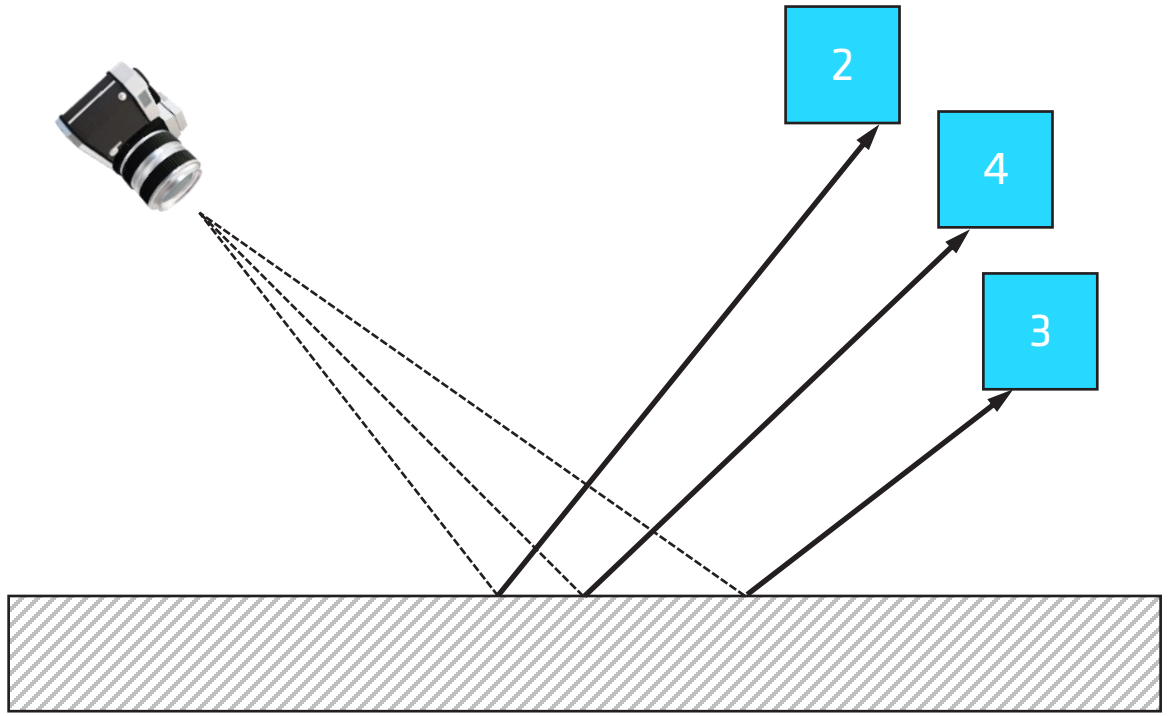
RAY TRACING DIVERGENCE

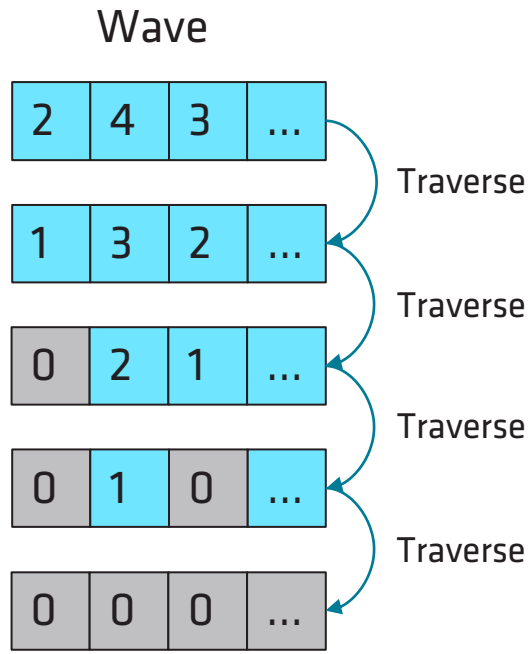
- Ray tracing is a latency heavy operation, very affected by divergence
- We look at below forms of divergence and how Far Cry 6 tackles them
  - Ray Divergence
  - Shader Table Divergence
  - Resource Divergence



RAY DIVERGENCE

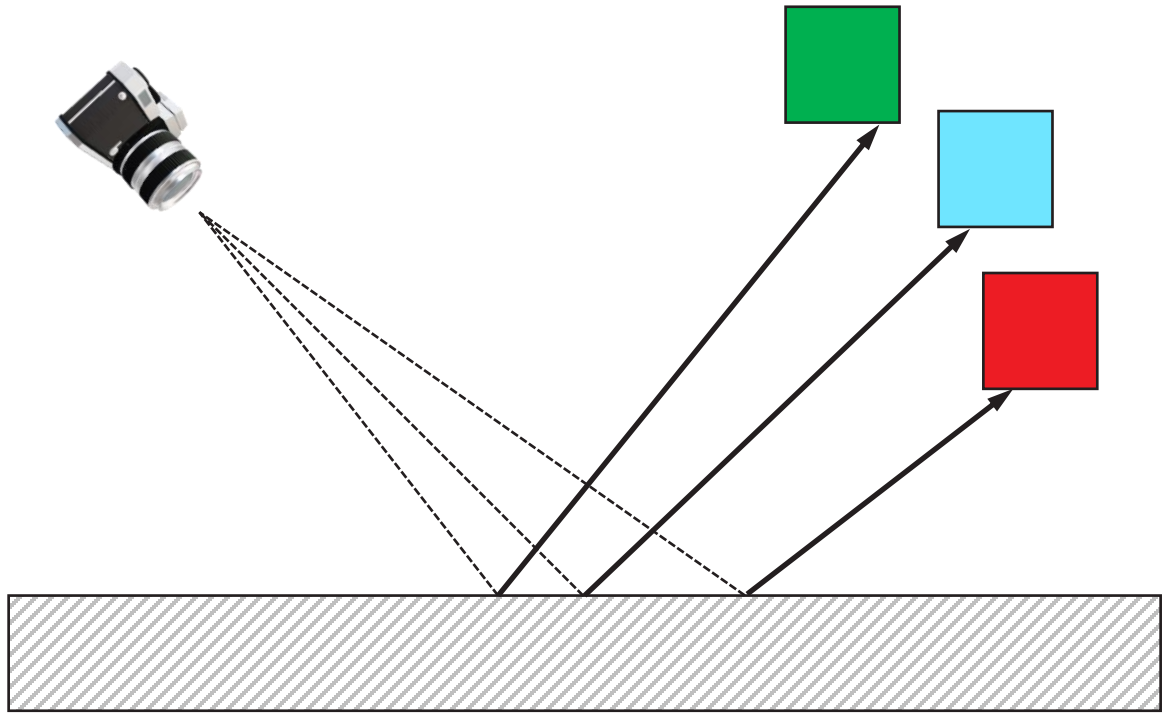
- Different rays can hit different BLAS's with number of levels that the ray has to traverse
- Executing shader will have to traverse for the worst case

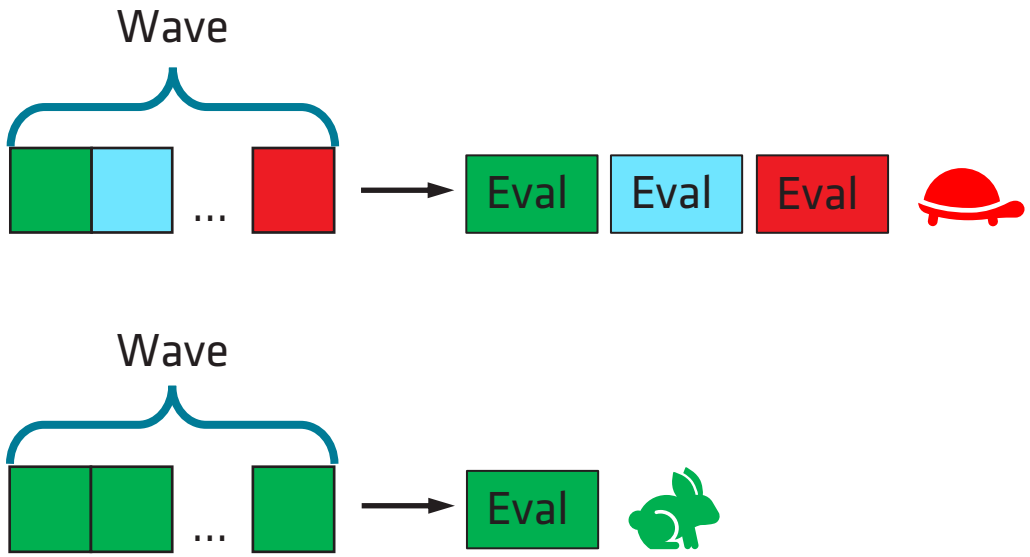




SHADER TABLE DIVERGENCE

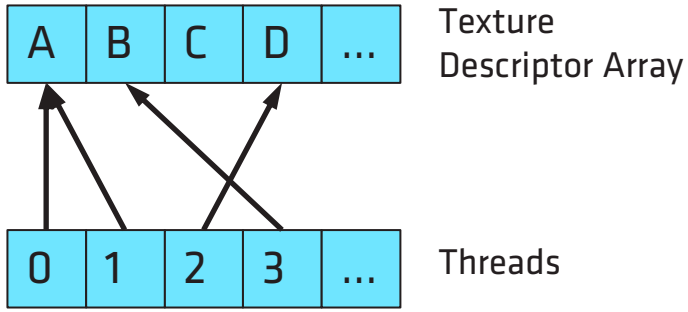
- Occurs when different rays evaluate different shader table entries for hit/miss
- We have to evaluate all the hit materials in our thread group sequentially, like a branch



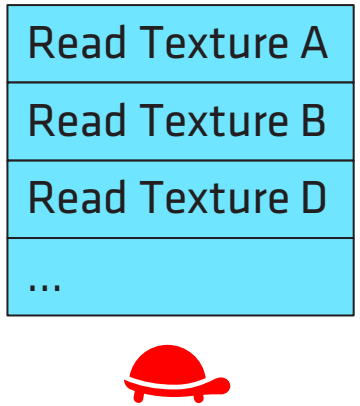


RESOURCE DIVERGENCE

- Occurs when a shader issues a resource read, but different threads in the wave can be accessing different resources
- We need to loop through all unique resources in our wave and have each issue the read for the threads using that resource
  - In HLSL, this is done via NonUniformResourceIndex



Textures[textureId].Sample



Texture.Sample

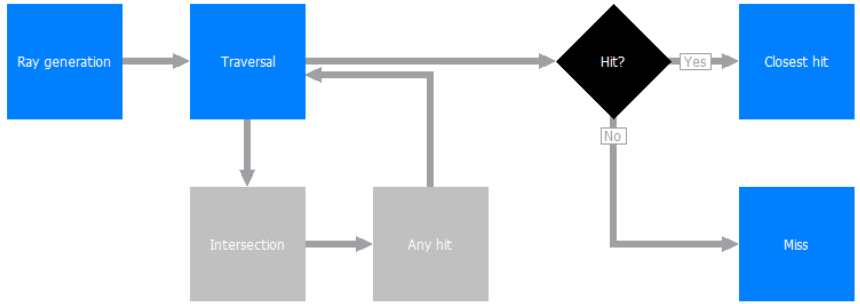
RAY TRACING HIT EXECUTION

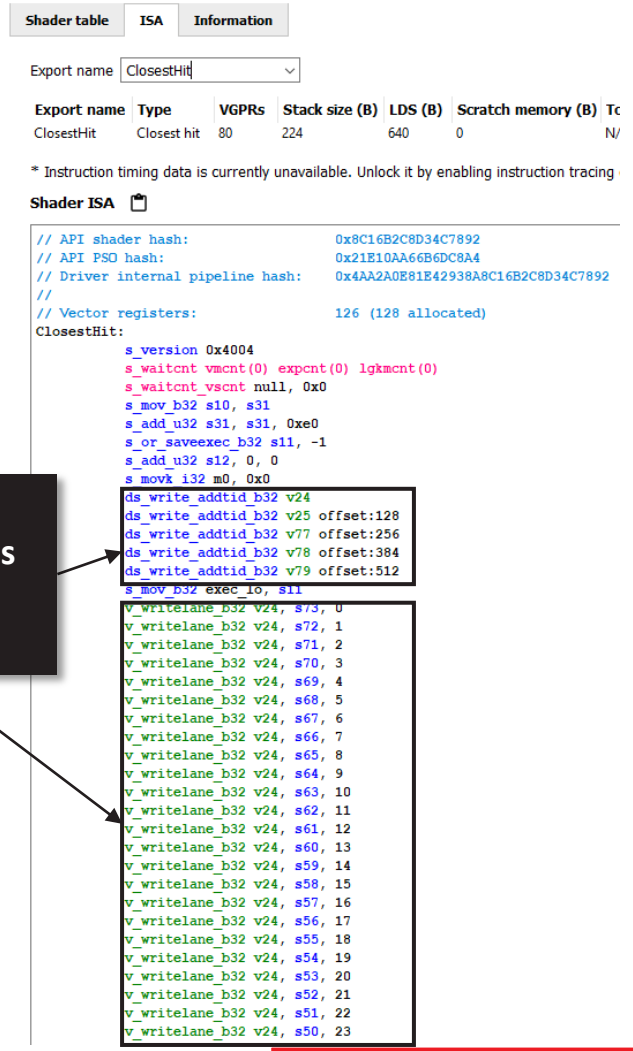
```
if (ray intersects)
{
    uint hit_table_index = calc_hit_table_index();
    switch (hit_table_index)
    {
        case 0:
            ray_result = shader_table_entry_0(ray_hit);
            break;
        case 1:
            ray_result = shader_table_entry_1(ray_hit);
            break;
        ...
        case n:
            ray_result = shader_table_entry_n(ray_hit);
            break;
    }
}
```

- Small shader table:
  - Everything gets inlined and the code behaves like a series of if statements
- Large shader table:
  - Too many shaders to inline, compiler will emulate a function calling convention with loading/storing function arguments/return values in LDS
- hit_table_index does not have to be uniform for a thread group

SHADER TABLES (NON INLINED)

- Example of what you will see in RGP when your shader table isn't inlined
- Lots of LDS usage, instruction cache misses, wasted instructions for writing/loading function arguments
- On a Radeon™ 6800XT at 4K, forcing the shader table to not be inlined is about 2-3x slower





Excess instructions and LDS Traffic

Shader Table

Allocated resources are driven by the shader table entries with the highest pressure on each resource, regardless of the call count.

Export name	Type	VGPRs	Stack size (B)	LDS (B)	Scratch memory (B)	Total latency (clks)	Instruction cost (%)	Instruction hit count	Call count *	API shader hash	Internal pipel
RayGen	Ray generation	126	240	0	0	N/A	N/A	N/A	N/A	0x2193817871F6D4F9	0xD4BE70F6FF7
TraceRaysAmdInternal	Traversal	88	64	4,096	0	N/A	N/A	N/A	N/A	0xAAA1CCC200000000FBD5CBD0CA959ED5	0xF790F03A966
ClosestHit	Closest hit	80	224	640	0	N/A	N/A	N/A	N/A	0x8C16B2C8D34C7892	0x4AA2A0E81E4
Miss	Miss	19	16	0	0	N/A	N/A	N/A	N/A	0xF78FE40F86CE1DC9	0x4CEF6A572CD
_amdgpu_cs_main	N/A	0	0	0	0	N/A	N/A	N/A	N/A	0x61DB175FE66864C9	0x301A80234BE

FAR CRY 6 SHADER TABLE

- Far Cry 6 has a single hit shader which interpolates vertex attributes that store all material properties
- Avoids lots of shader table divergence with only 1 hit and 1 empty miss shader
- Avoids resource divergence since all DXR Prims are in global buffers
  - No per material textures get accessed
- Ray divergence is partially mitigated by BVH building
  - We will get to this in next section

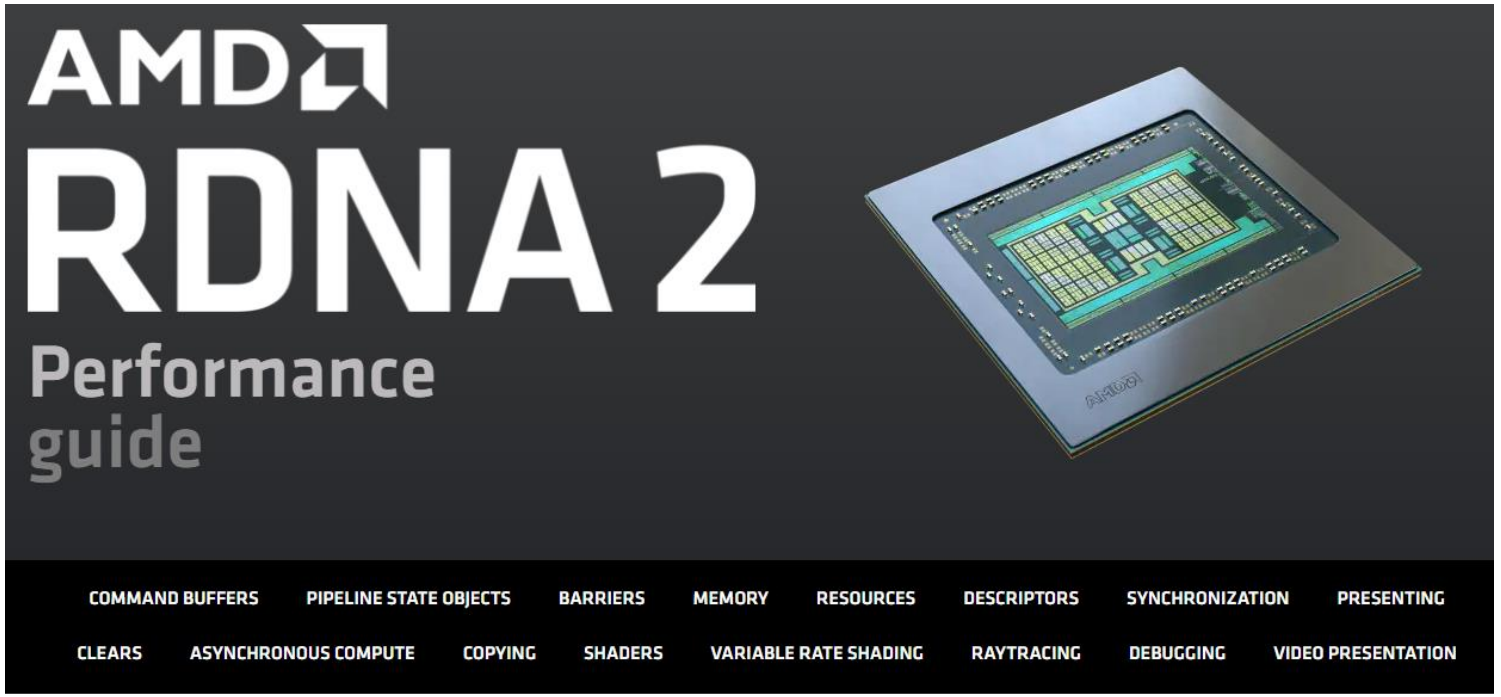
```
uint geometryOffset = instanceIndex * gs_DxrGeometriesPerInstance + geometryIndex;
uint primOffset = RTPrimOffsetBuffer[geometryOffset];

DXRPrim prim = DXRPrimUnpack(RTPrimBuffer[primOffset + primitiveIndex]);
float3 color = COMPUTE_BARYCENTRIC_POINT(barycentrics, prim.vertices[0].color, prim.vertices[1].color, prim.vertices[2].color);
// ...
float reflectance = COMPUTE_BARYCENTRIC_POINT(barycentrics, prim.vertices[0].reflectance, prim.vertices[1].reflectance, prim.vertices[2].reflectance);

payload.hitColor = ApplyLighting(WorldRayOrigin(), WorldRayDirection(), RayTCurrent(), color, ... reflectance);
```

AMD RAYTRACING PERFORMANCE GUIDE

- You can find our performance recommendations at https://gpuopen.com/performance/
- We have raytracing and BVH recommendations in the Ray Tracing section

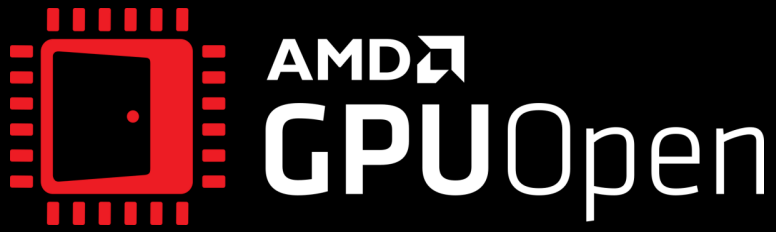


Optimizing a modern real-time renderer can be a somewhat daunting task. Explicit APIs hand you more control over how you craft your frame than ever before, allowing you to achieve higher frame rates, prettier pixels, and ultimately, better use of the hardware.

Our **AMD RDNA™ 2** Performance Guide will help guide you through the optimization process with a collection of tidbits, tips, and tricks which aim to support you in your performance quest.

BVH OVERVIEW

GDC



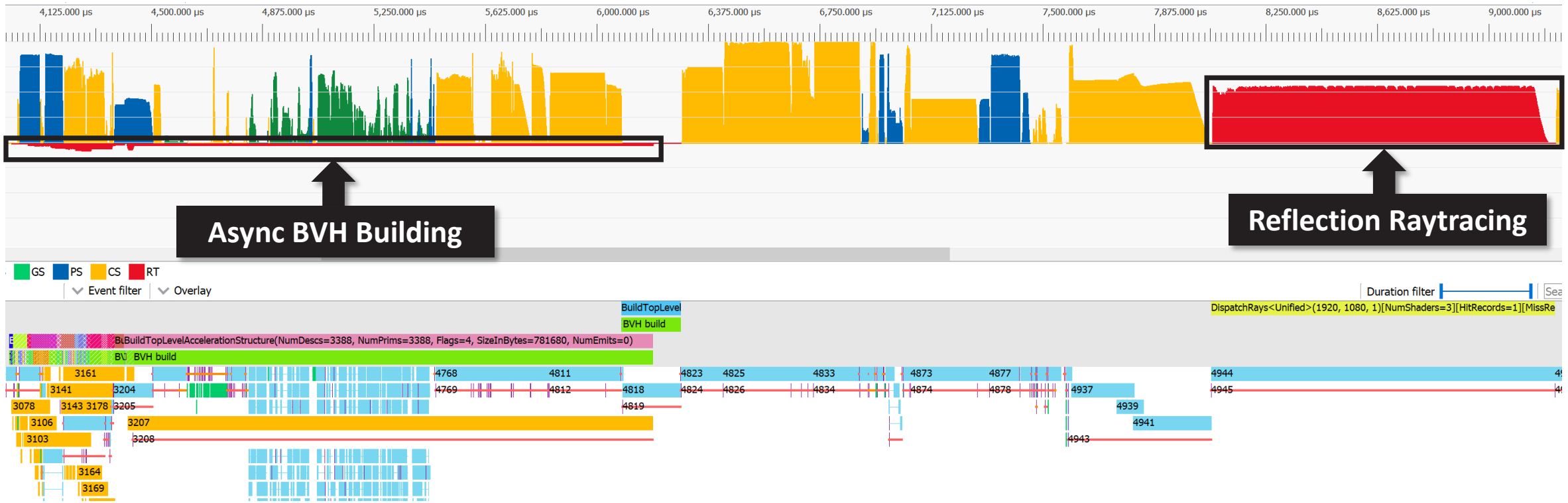
BLAS VERTEX GENERATION

- Far Cry 6 uses compute shader to generate vertices for BLAS building instead of a VS/GS/HS/DS
  - Can’t do per vertex/object culling if done in a geometry pass
  - Avoids redundancy with vertex cache miss
- Compute shader in Far Cry 6 stores material data per vertex which are calculated via reading material textures of the simplified models at the vertices
- Vertices generated by CS for BLAS can be reused for shadows and potentially other passes

	VS Write	GS Write	HS/DS Write	CS Write
Culled Geometry	No	No	No	N/A
Vertex Cache Miss	Yes	Yes	Yes	No

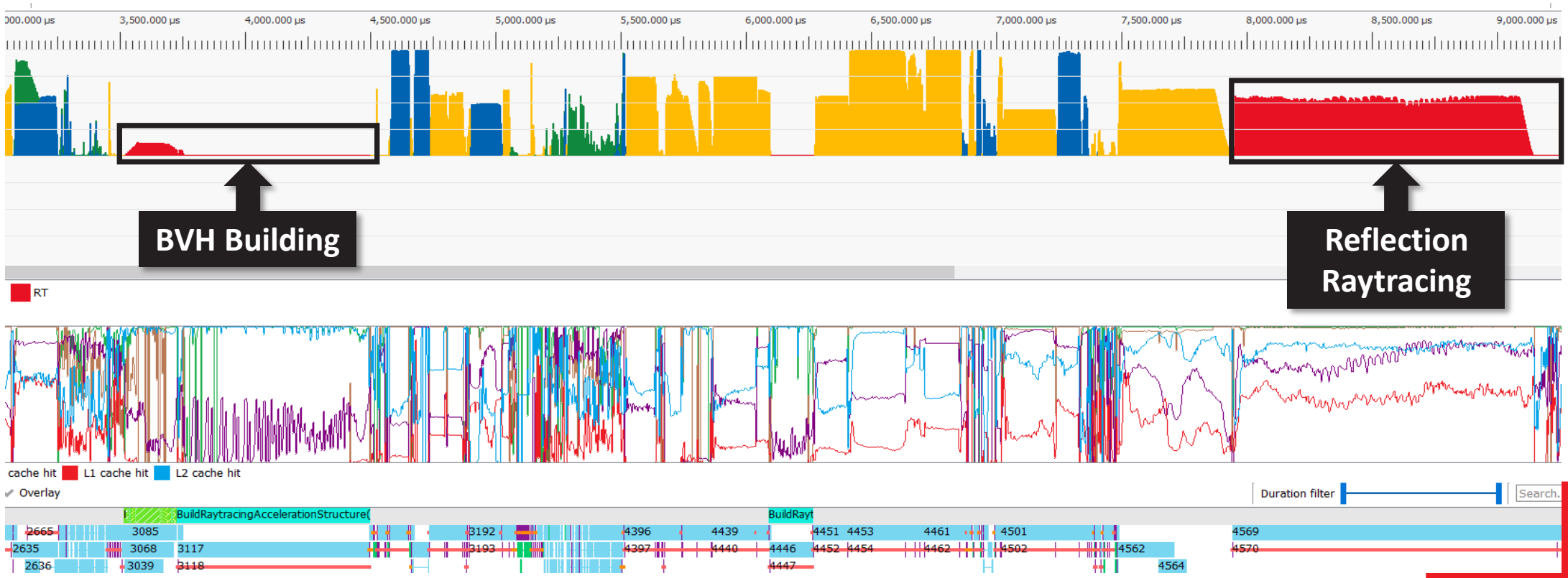
BVH BUILDING

- Looking at the RGP trace again, you can see that BVH building is executed on async compute

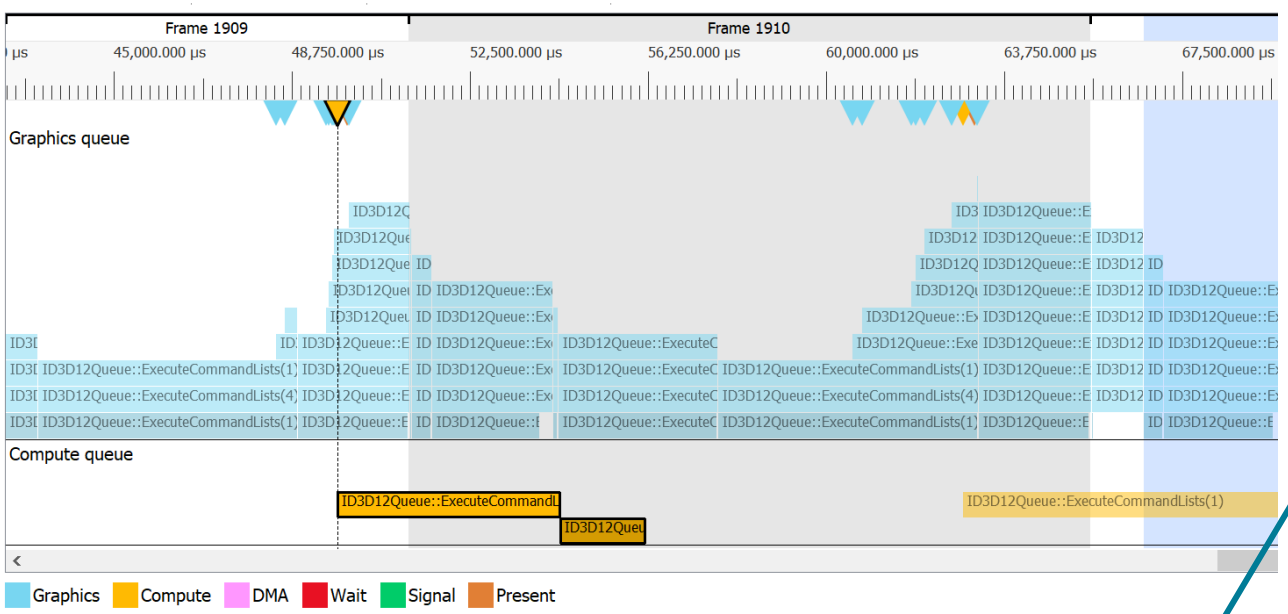


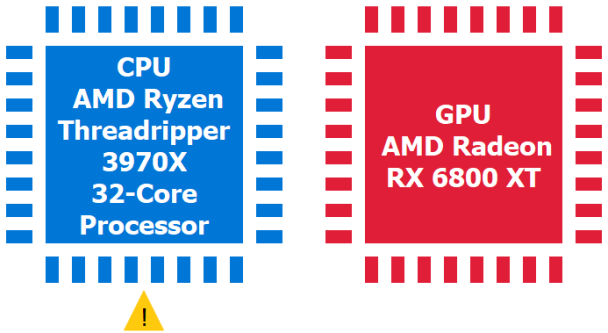
ASYNC BVH BUILDING

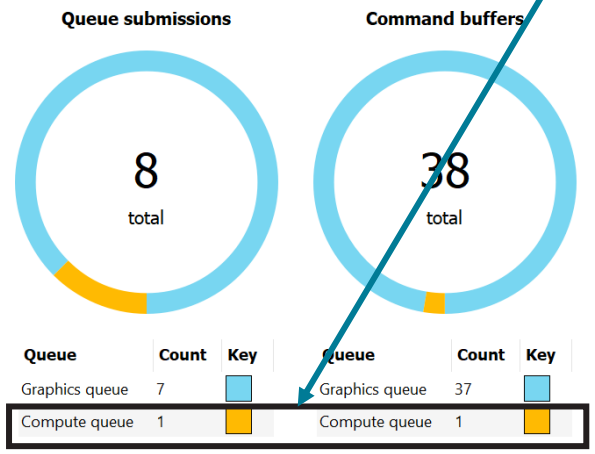
- BLAS/TLAS generation doesn't usually depend on other parts of the frame until you start ray tracing
  - Can overlap with Gbuffer pass/shadows/early frame workloads
- Below is an image of non overlapped BVH generation (0.4-0.8ms slower):



ASYNC BVH BUILDING



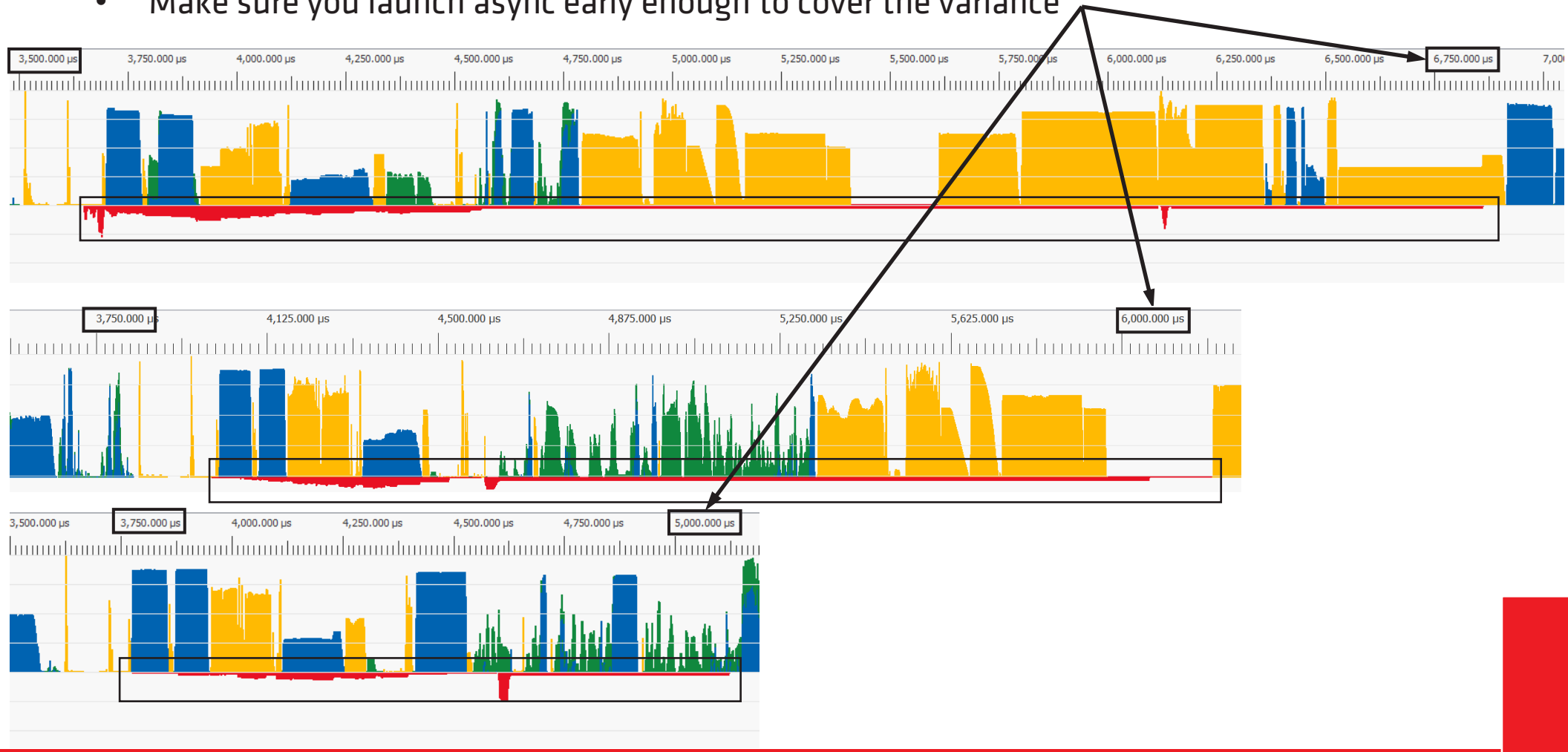




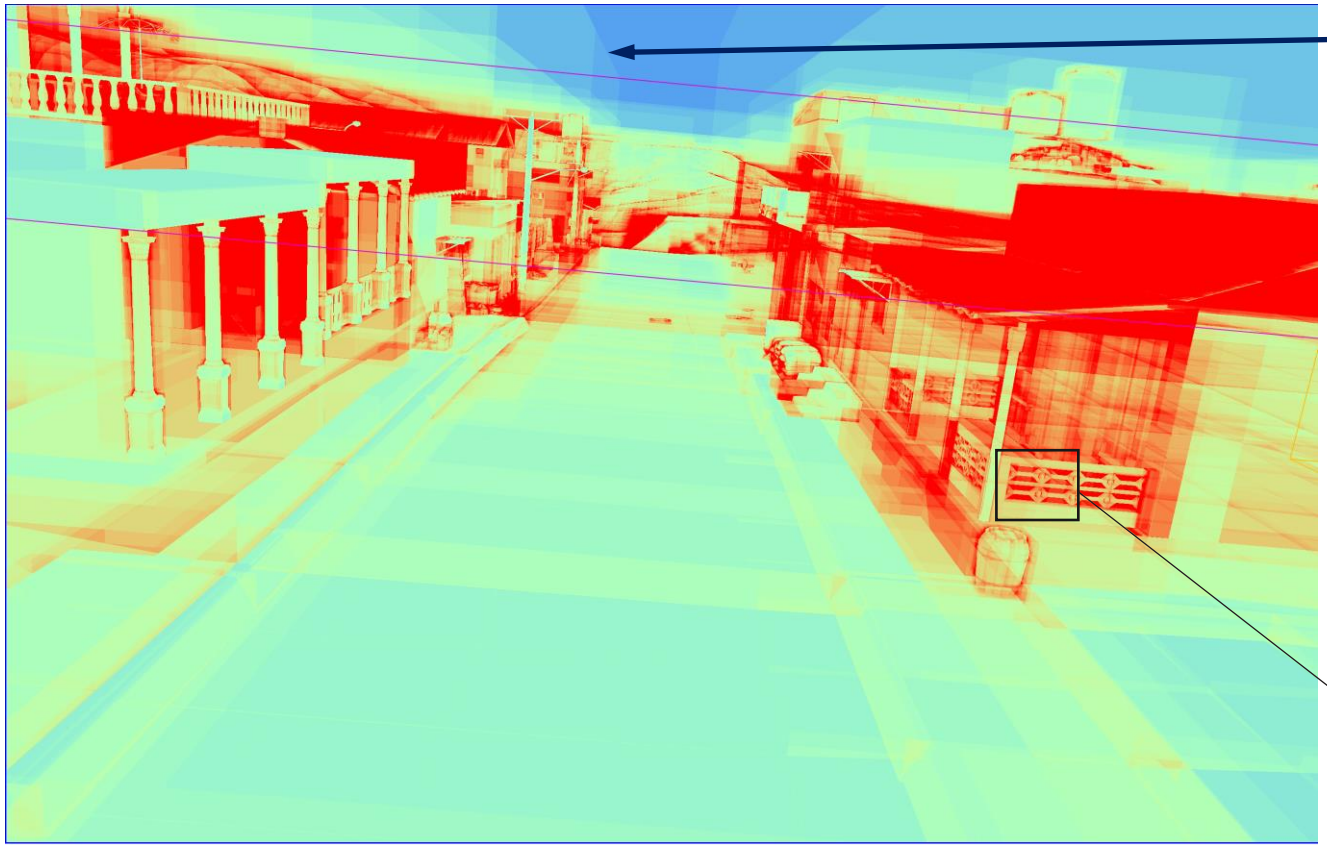
- Low amounts of synchronization are needed between graphics and compute queue
  - On PC, sync between graphics and compute queue is higher than console
- Far Cry 6 has one async compute command list with all the BVH building calls and syncs to just before SSLR

ASYNC BVH BUILDING

- Some frames have much higher BVH building times vs others, which async helps hide
  - Make sure you launch async early enough to cover the variance

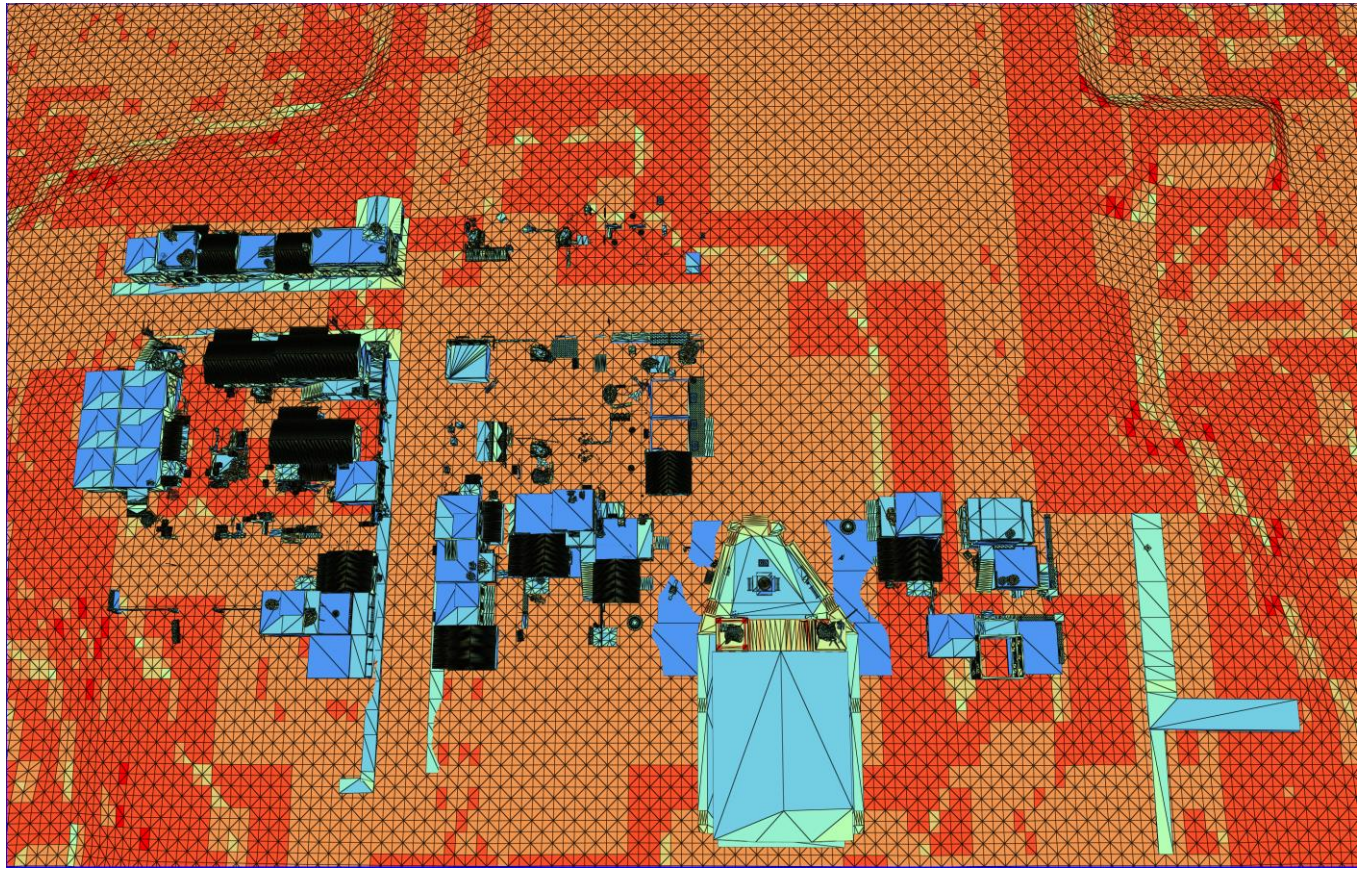


BVH TRAVERSAL COST



- Rays that miss geometry still pay the traversal cost for the TLAS hierarchy
  - Part of why having particles in separate TLAS gave good perf wins
- Best to avoid putting objects into TLAS if they are far away or unlikely to affect rendering result
- Camera is usually deep in a tree, so all rays can be affected by the higher traversal cost

BVH TRAVERSAL COST



- Far Cry 6 limits the objects that are in the TLAS to only a radius around the player
  - Terrain clip mapped is contained with a longer range
- BLAS objects are at a lower LOD, less nodes to traverse to get to triangles
  - More correct rendering for distance objects, fully detailed objects would alias
  - Helps address ray divergence

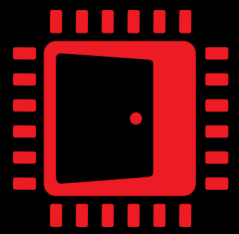
HYBRID REFLECTIONS BLAS/TLAS TIPS

- Consider using CS for vertex generation for BLAS
  - Far Cry 6 generates BLAS positions and DXR Prim attributes in a single CS pass
- We recommend trying async compute for BLAS/TLAS building
  - Easy to overlap with early/mid parts of your frame, big perf win for Far Cry 6
- Try to limit amount of geometry in your TLAS to avoid cost of traversing large TLASes per frame
  - For Far Cry 6 reflections, only objects within a radius of 100 are put into the BVH
- Consider having lower LODs for reflection BLASes for faster tracing and lower VRAM usage
  - Far Cry 6 uses a constant lower LOD for objects in BLASes

EPYC RADEON RYZEN AMD

HYBRID REFLECTIONS SAMPLE

GDC



AMD GPUOpen

HYBRID REFLECTIONS OVERVIEW

Far Cry 6 Reflections

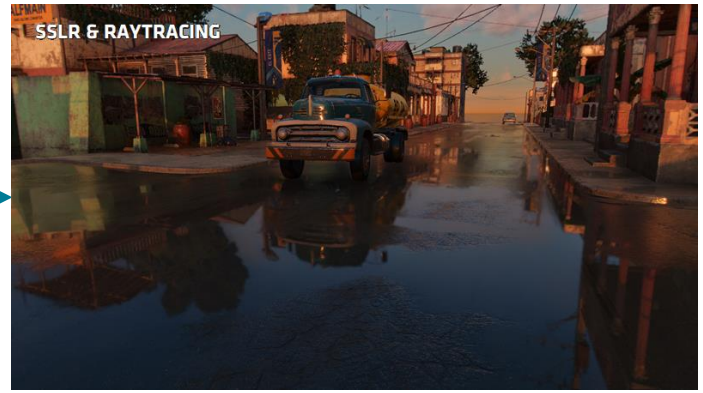
Tracing

SSLR

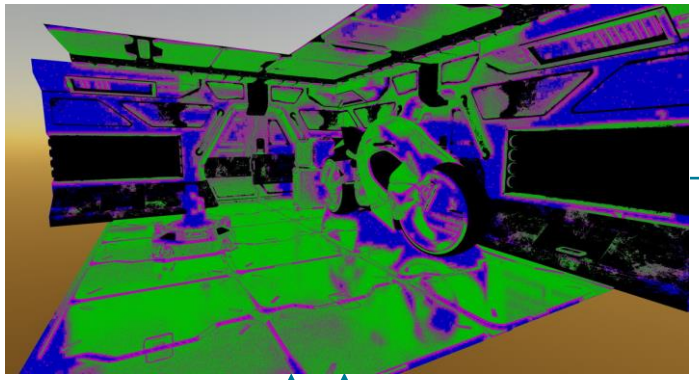
HybridRays

DXR Rays

Far Cry 6 Denoiser



Hybrid Sample Reflections



Tracing

SSLR

HybridRays

DXR Rays

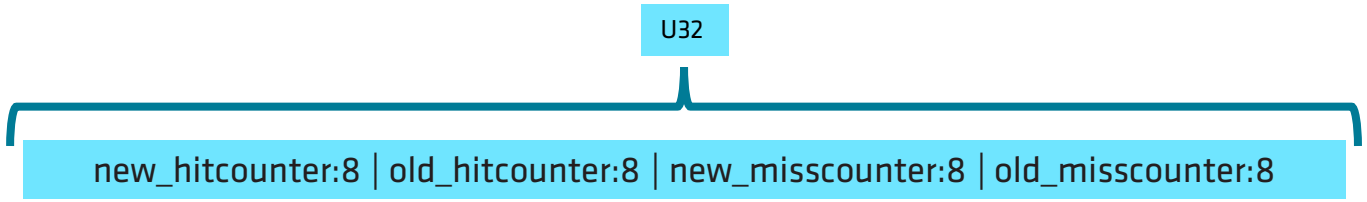
FidelityFX-Denoiser

Hit/Miss counters

Variance

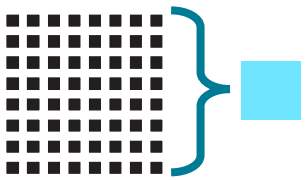
FEEDBACK COUNTERS

- We store a feedback counter for each set of 8x8 pixels (1 tile)
- The counter consists of one U32 with hit/miss counters for 2 frames of statistics



- When deciding if a ray should be Hybrid or DXR, we use reprojected counters from previous frame

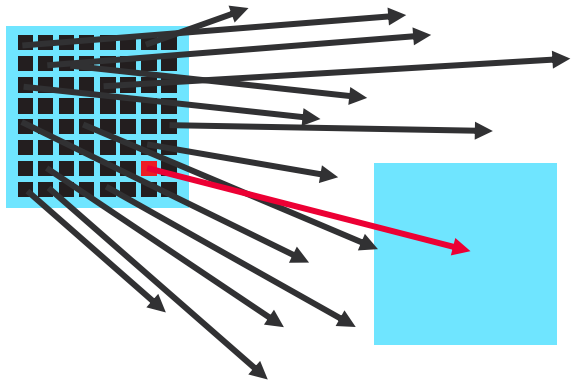
1 U32 per 8x8 tile of pixels



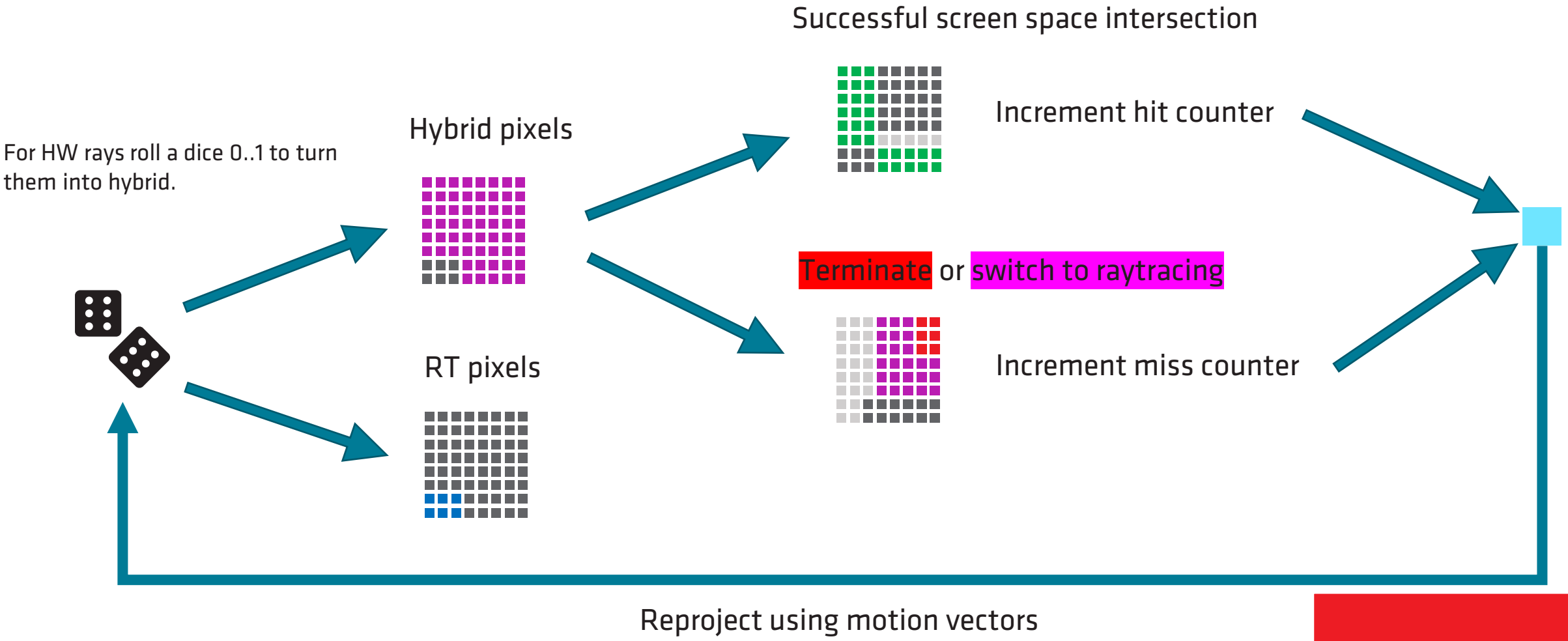
FEEDBACK COUNTERS

- Reproject counters by picking a random motion vector in the 8x8 tile
- We can switch pixels from SSLR to DXR, also need a way to switch from DXR to SSLR
  - Problem is all our DXR rays will always hit something with high confidence
- We solved this by randomly converting pixels in tiles from DXR to Hybrid

Reprojection:



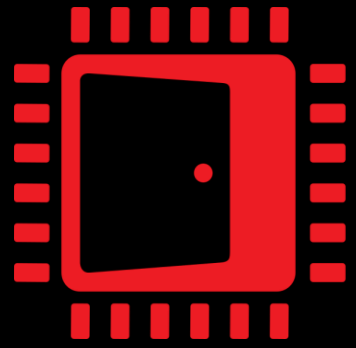
CLASSIFICATION



CONCLUSION

- Overall, Hybrid Raytraced Reflections is a powerful technique that gives a good quality boost over SSLR
- Keeping raytraced reflections hybrid really helps in making the technique viable for real time
  - Lots of performance saved by using SSLR where applicable
- Plan ahead for asset system integration and handling materials in ray tracing shaders
- Important to keep in mind how ray tracing runs on HW and its performance characteristics
- AMDs Hybrid Reflection sample is currently available at https://gpuopen.com/learn/hybrid-reflections/
  - DX12 implementation with docs and MIT licensed public source code

DISCLAIMER AND ATTRIBUTION



AMD GPUOpen

EPYC

RADEON

RYZEN

AMD