

Dragon Age: The Veilguard – GI, RT, Character Creator and other systems.

Speakers: Kleber Garcia, Darrin Stewart & Navjot Garg

Website Abstract

We present the architectural challenges & decisions related to global illumination, ray tracing and character creator for the latest installment of the *Dragon Age* series – ***Dragon Age: The Veilguard***. First, we will cover the challenges around our probe baking system and decisions made around the team size and asset constraints. Next, we will share implementation choices taken for ray tracing considering the limitations of already existing ray tracing tech, with the additional requirement of supporting more GPUs. Lastly, we will go over our character creator tool, decisions on the technical side as well as some practical technical details making this technology fulfill the game's requirements.

Speakers: Kleber Garcia, Darrin Stewart, Navjot Garg

Agenda

- Introduction to Dragon Age: The Veilguard
- Global Illumination
- Ray Traced Reflections
- Character Creator
- Q&A

Dragon Age: The Veilguard

- Frostbite engine
 - Editor, pipeline and runtime
 - World building tools
 - Lighting & rendering pipeline
- Dragon Age: The Veilguard
 - Latest installment in Dragon Age series
 - Launched on PS5, XBSX/S and PC
 - Small level content team, 15-20 people
 - 5 rendering engineers
 - Indoor and outdoor vibrant environments
 - Fast iteration, 16x16km terrain



GI Results



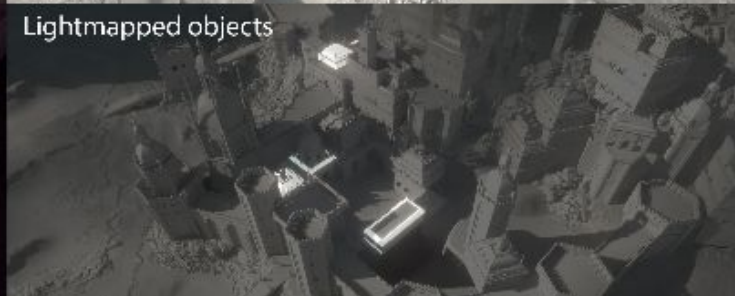
GI Results



GI Results



Lighting only



Lightmapped objects

GI Results



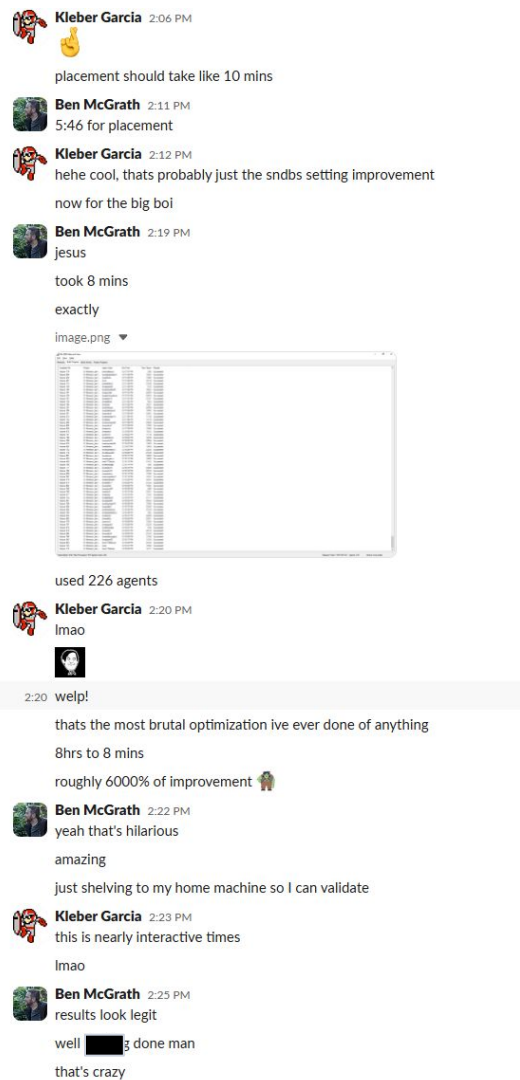
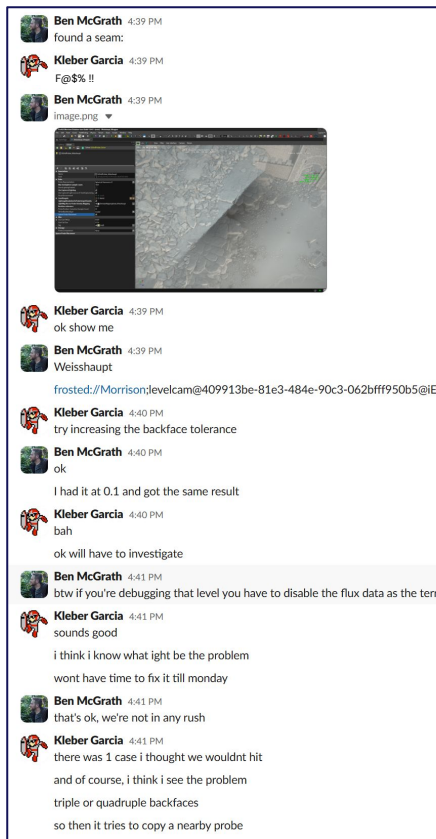
Global illumination constraints

- Content team: 15 people, 16x16km levels with huge variability
 - Realtime ray tracing GI solution still immature through prepro and production (2020)
 - Limit precomputation capacity and maintenance
- No content capacity to do much lighting
 - Rely mostly on autoplaced probes
 - Avoid lightmaps as much as possible: lightmaps still needed
- Levels highly kit bashed
 - Apply GI efficiently (see GI + Gbuffer lay down problem)
- Quick iteration
 - Bake times must be fast – bake distribution using SN-DBS
- Maintain a high quality
 - Threw memory at the problem and relied on a free streaming system
 - Smaller levels meant no need to do zone streaming, but rather camera based chunk loading

Global Illumination

Collaboration model:

- Ben & Kleber having tons of high iteration / collaboration
- Engineer: working mostly on two branches and preporting fixes. Using game team branch to validate
- Iterate vertically: from UX to runtime to SNDBs

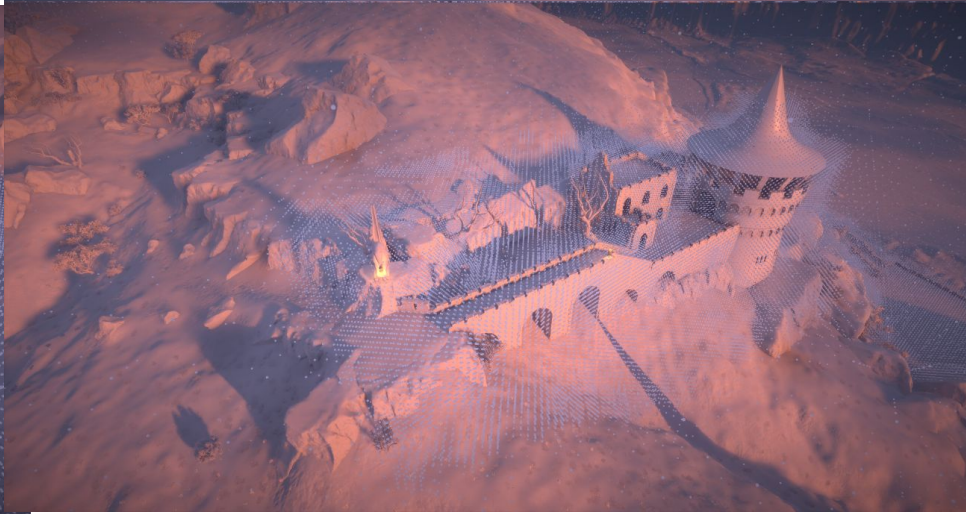


Global Illumination baking

- **Flux – frostbites radiosity suite (improvements)**
 - Radiosity path tracer suite (.dll module & inspector)
 - Probe placement and radiosity baking (lightmap and probe)
 - Headless distributed baking functionality built on top of Sony's SN-DBS
 - FluxViewer – editor / inspector and debugging
- **GiGrid – frostbites probe placement solution (improvements)**
 - Runtime: Fully probe lit, streamable volumetric probe system
 - Cook pipeline: async bake calling flux and generates game ready entities & resources
- Fast bake times – 10 mins to 15 mins per level
- Few lightmaps, mostly auto placed probes

Global Illumination baking

- 500 mb streaming budget
- Do as much in probes
- Up to 0.5m for highest resolution
- Light leaking: build chunky architecture. Thinnest wall .5m



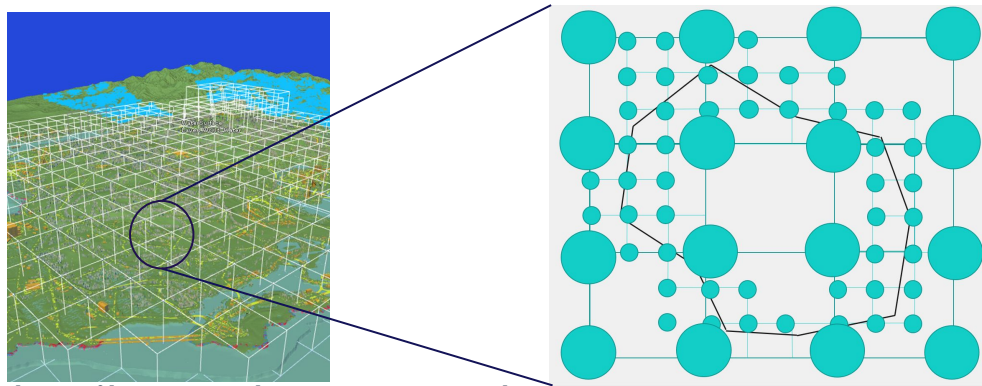
Global Illumination

- Lightmap vs probes
- Lightmap for objects that could just leak too much



Global Illumination baking

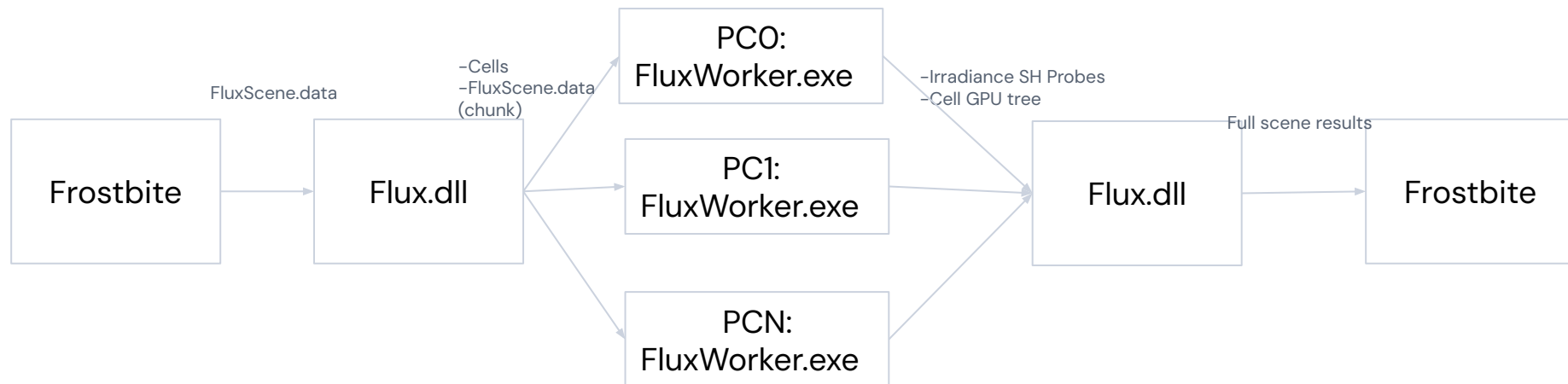
- World split into cells, each cell holding probe recursive tree structure
- Tree levels progressively stream from disk to GPU



- Further details on placement algorithm:
 - <https://dl.acm.org/doi/10.1145/3388767.3407314>

Global Illumination baking

- Flux does the placement and baking by splitting the world in cells
- Send bundle of cells to many agents, bake then reconstruct



Global Illumination baking

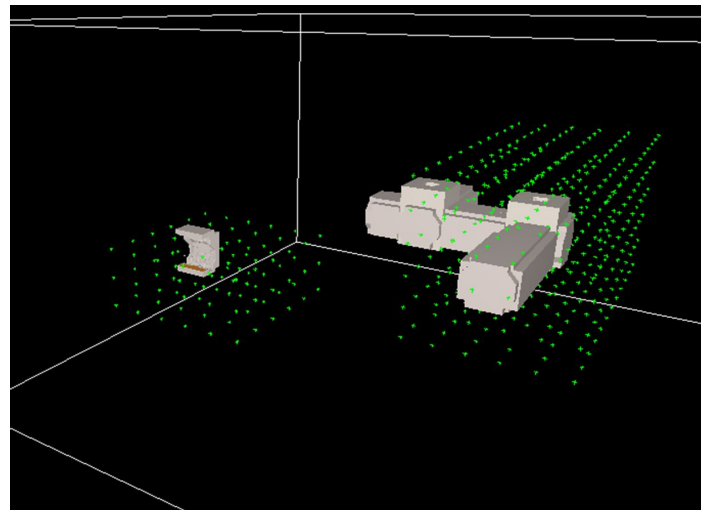
- Previous vendor increased prices significantly
- Company mandated SNDBs was a last minute requirement
- VPN + Covid was a huge issue. Mitigated by keeping machines in office.

gent	User Name	Status	Time In This State	CPU	Total RAM	Ratio
>snlbs		Excluded from build	000:00:00	2x Intel(R) Xeon(R) E5-2697A v4 @ 2.60GHz @ 2.59 GHz	8192 MB	4
11tpalamar1	tpalamar	Excluded from build	000:07:50	Intel(R) Core(TM) i7-4330K @ 3.40GHz [6 cores, HT] @ 3.4 GHz	65475 MB	11
11ndufree2	ndufree	Excluded from build	000:00:19	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65447 MB	193 (29)
11jashcroft2	jashcrofthew	Excluded from build	000:00:00	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65448 MB	188 (28)
11parisman3	parisman	Excluded from build	000:15:11	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65447 MB	15
11plohaman-10	plohaman	Excluded from build	119:35:02	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65435 MB	15
11knoone	knoone	Excluded from build	000:04:38	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65438 MB	14
11chobbs	chobbs	Excluded from build	655:05:36	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65448 MB	13
11mflewelling	mflewelling	Excluded from build	000:19:24	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65437 MB	13
11kgarcia-10		Excluded from build	040:26:36	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65446 MB	13
11pirtalato	pirtalato	Excluded from build	000:00:33	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65437 MB	13
11jcobb	jcobb	Excluded from build	000:00:06	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65435 MB	130 (26)
11katwang	katwang	Excluded from build	000:01:01	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65447 MB	123 (24)
11ecookinham	ecookinham	Excluded from build	000:00:14	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65447 MB	9
11jboocher1	jboocher	Excluded from build	000:00:23	Intel(R) Core(TM) i7-6800K @ 3.40GHz [6 cores, HT] @ 3.4 GHz	65447 MB	268 (32)
11parisman2	btrosin	Excluded from build	000:00:06	Intel(R) Core(TM) i7-4530K @ 3.40GHz [6 cores, HT] @ 3.4 GHz	65475 MB	14
11apawlowski	adpawlowski	Excluded from build	000:49:57	Intel(R) Core(TM) i7-4530K @ 3.40GHz [6 cores, HT] @ 3.4 GHz	65475 MB	133 (31)
11adobzins-r		Excluded from build	000:12:52	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65438 MB	26
11jocohen	jocohen	Excluded from build	000:46:27	Intel(R) Core(TM) i7-5820K @ 3.30GHz [6 cores, HT] @ 3.3 GHz	65437 MB	14
11jwamen-4	jwamen	Host Busy (cpu)	000:03:21	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65208 MB	245 (48)
11bhxon	bhxon	Host Busy (cpu)	000:04:22	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	225 (45)
11mcomelu1	mcomelus	Host Busy (cpu)	003:46:16	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	336 (44)
11ccampbell	colncampbell	Host Busy (cpu)	000:00:29	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	43
11ylabrador2	YLABRADOR	Host Busy (cpu)	000:03:40	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65212 MB	247 (49)
11enelwaya	enelwaya	Host Busy (disk space)	014:21:01	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65212 MB	247 (49)
11rkashyap1	rkashyap	Host Busy (disk space)	002:30:11	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65208 MB	238 (47)
11yhu	yhu	Host Busy (disk space)	002:44:19	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	44
11blindgren	blindgren	Ready	000:00:03	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65212 MB	237 (47)
11echrisman		Ready	000:38:31	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	239 (47)
11jlogsdon	jlogsdon	Ready	000:00:03	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65211 MB	23
11kgarcia1		Host Busy (Visual Studio : msbuild.exe (4348))	000:02:48	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	24
11esherwood1	esherwood	Ready	000:07:12	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	24
11pmore	patrickmore	Ready	000:09:56	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65208 MB	343 (45)
11fbreen1	fbreen	Ready	000:05:25	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65208 MB	110 (22)
11clamontag1	CLamontagne	Host Busy (cpu)	000:01:12	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65208 MB	422 (48)
11tlu	blu	Host Busy (cpu)	000:11:06	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65208 MB	119 (47)
11jgojohnson	jgojohnson	Ready	000:02:19	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	352 (46)
11trobbsins2	trobbsins	Host Busy (disk)	000:00:03	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	365 (41)
11Scalibrate		Ready	000:13:44	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65212 MB	25
11michoward	michoward	Ready	000:01:15	Intel(R) Xeon(R) W-2145 @ 3.70GHz [8 cores, HT] @ 3.7 GHz	65210 MB	249 (49)

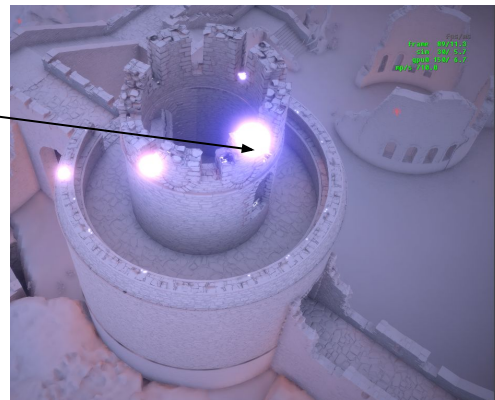
Finished Build: 0:00 Files Processed: 0 Agents Used: 0 Elapsed/Total: 0:00/0:00 Agents: 40 Broker: tib-snlbs

Global Illumination baking

- FluxViewer.exe -> inspect FluxScene, GiGrid cells and other possible artifacts
- Small stand alone application
- Added support for volumetric probes and dynamic objects to test probes

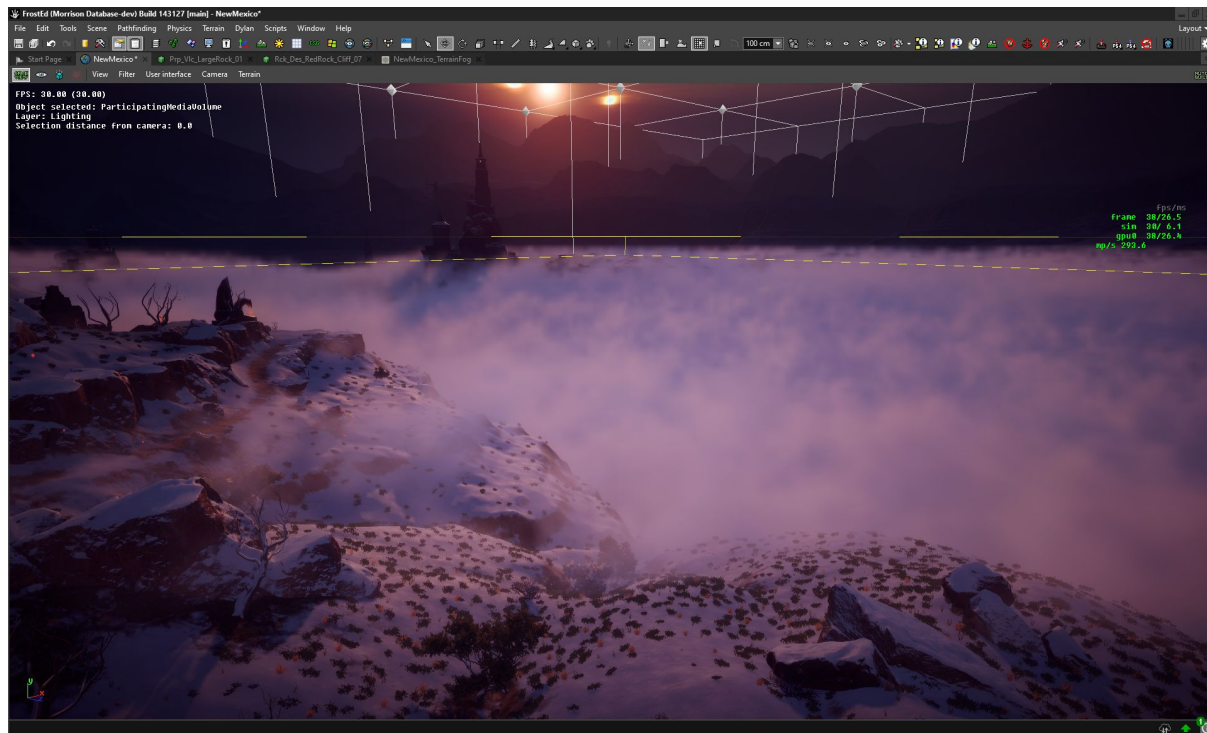


Bad normals == qnans



Global Illumination

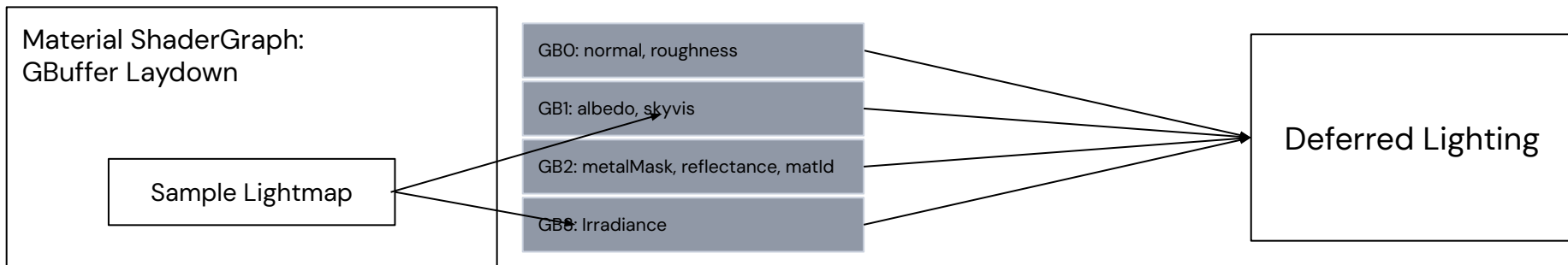
Volumetric fog support



Global Illumination

The GI - Gbuffer laydown problem -

Lightmap objects fetch lightmap during gbuffer laydown into an irradiance buffer

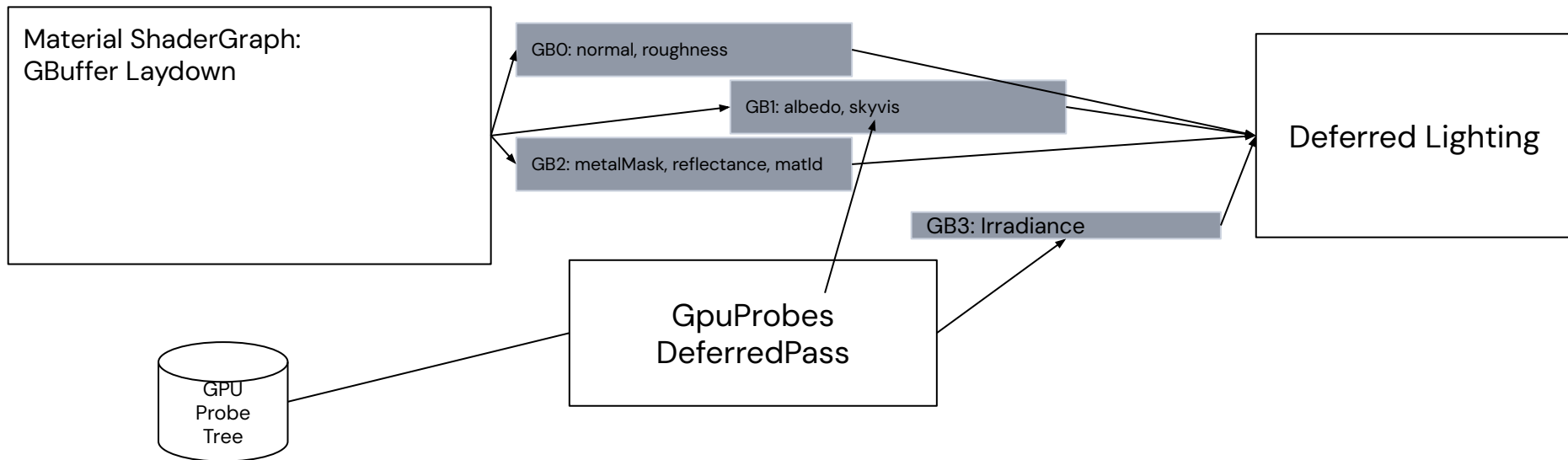


Global Illumination

The gbuffer laydown problem –

Probed objects fetch irradiance after, in a different pass.

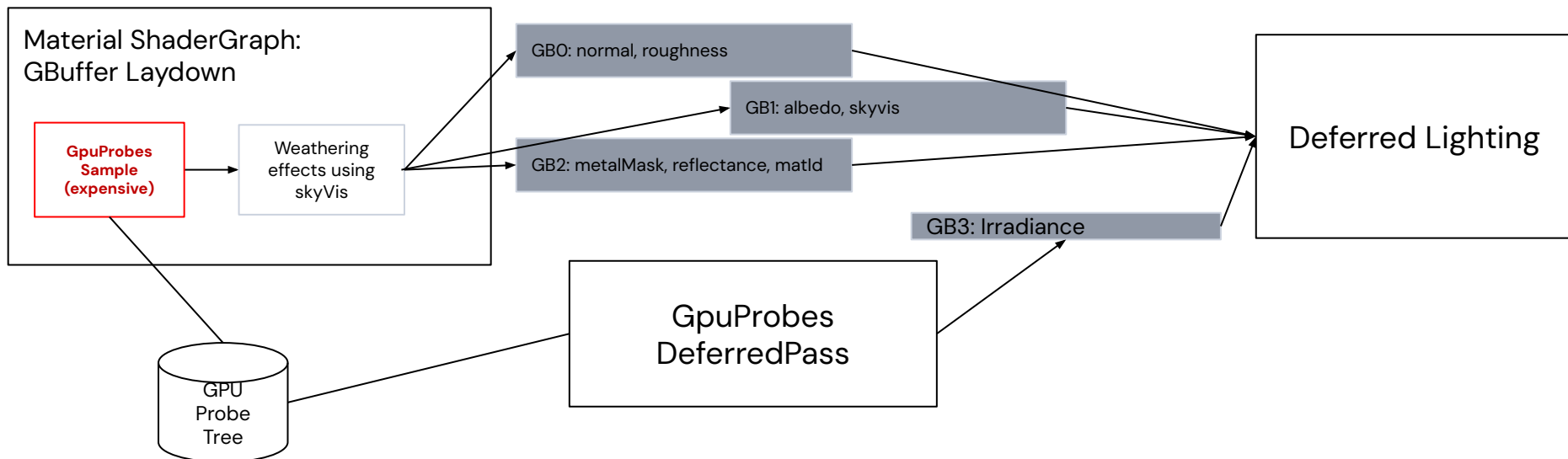
The reason is performance –> sampling a hierarchy during material laydown is expensive!



Global Illumination

The gbuffer laydown problem –

Problem: content creator uses a skyvis node in the material graph for weather effect! Adds massive VGPR cost!

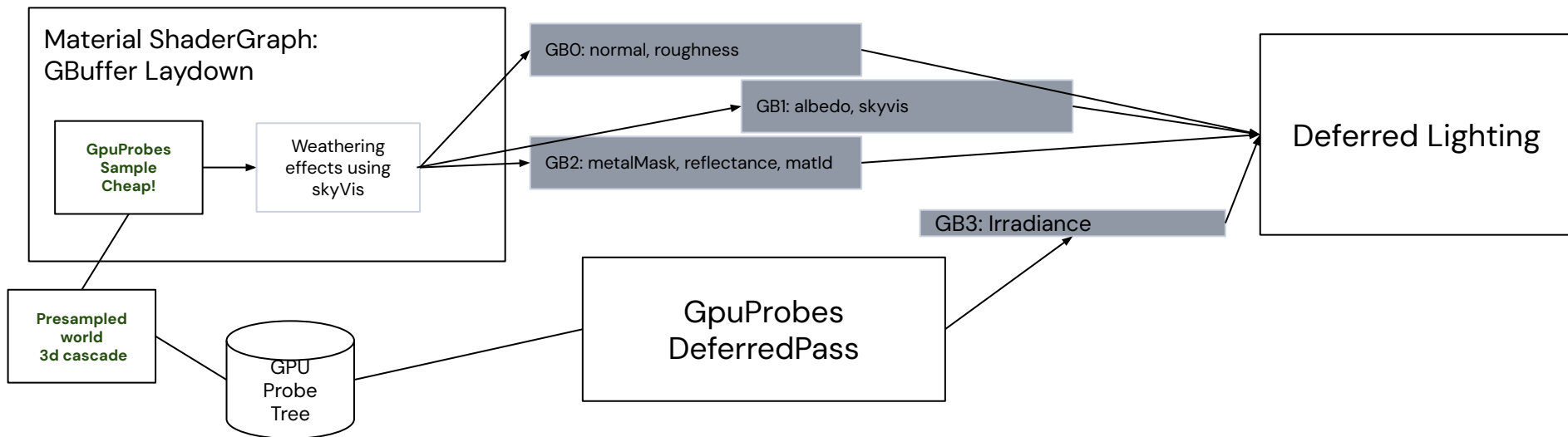


Global Illumination

The gbuffer laydown problem –

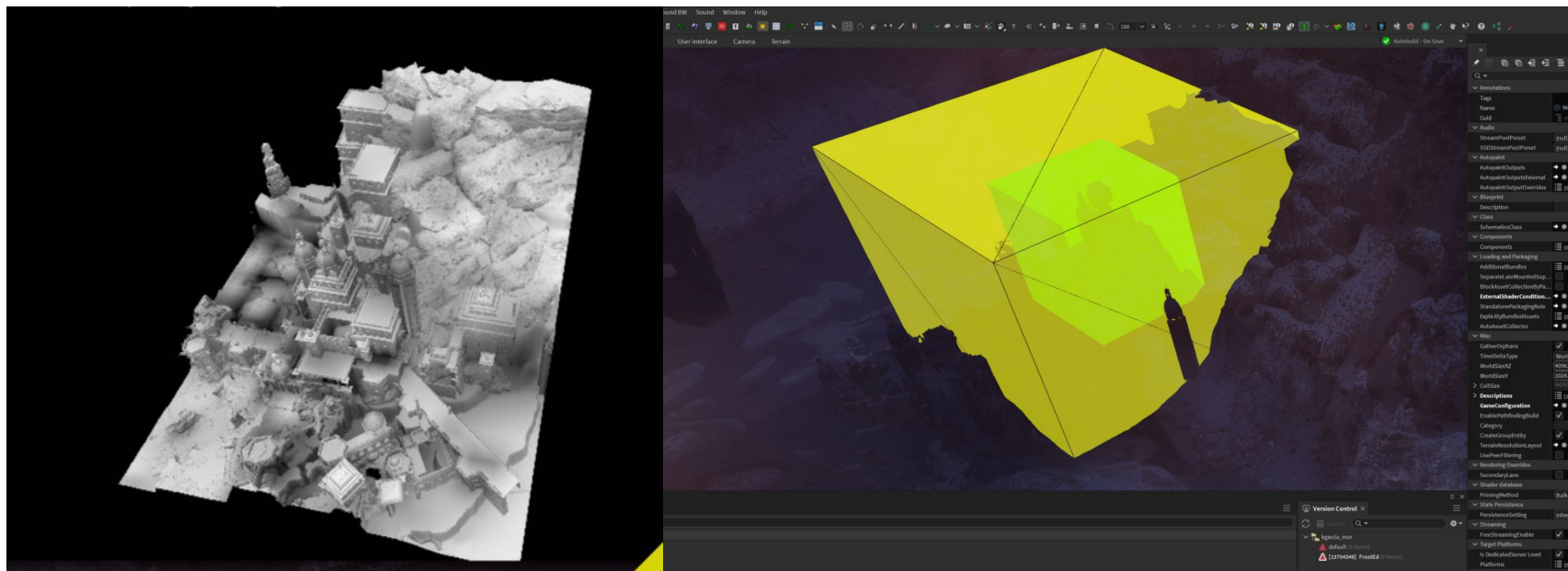
Solution: Sample cascade skyvis presampled grid -> 1 tap on 3d texture, cheaper

Lesson: Reading lighting in material graph == trouble <-> avoid this in your engine



Global Illumination

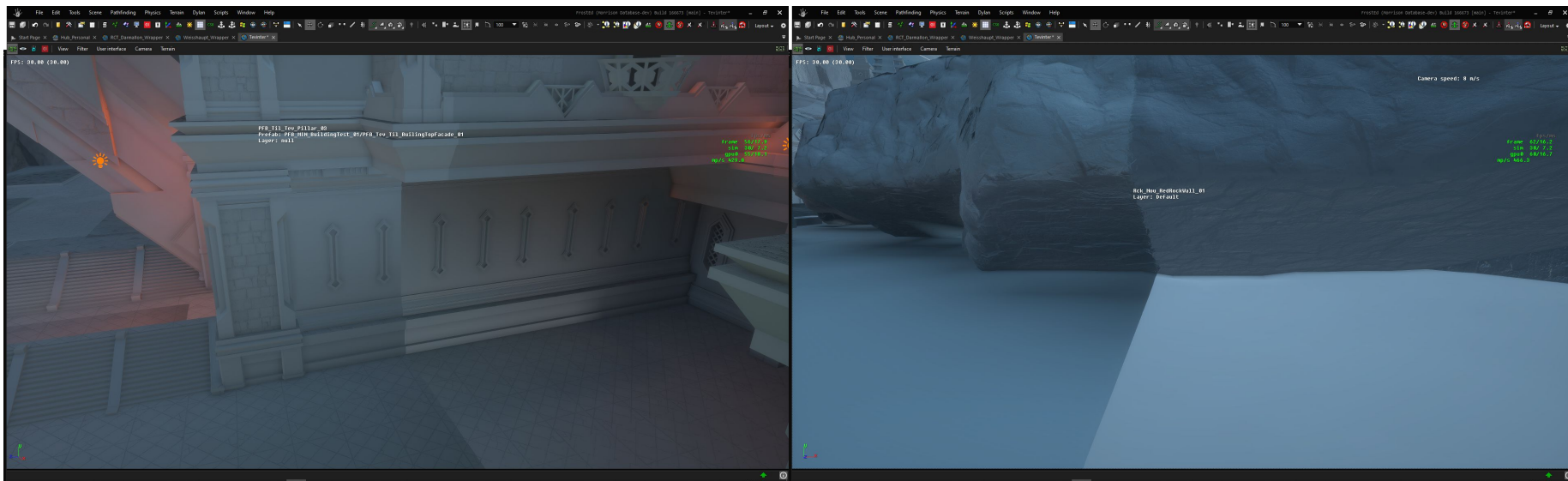
Solution: "Presample" using a weathering skyvis probe cascade around camera



Global Illumination

Fighting seams, some reasons

- Determinism issues between bake agents -> neighbor probes different
- Backface sampling strategy



Budget driven

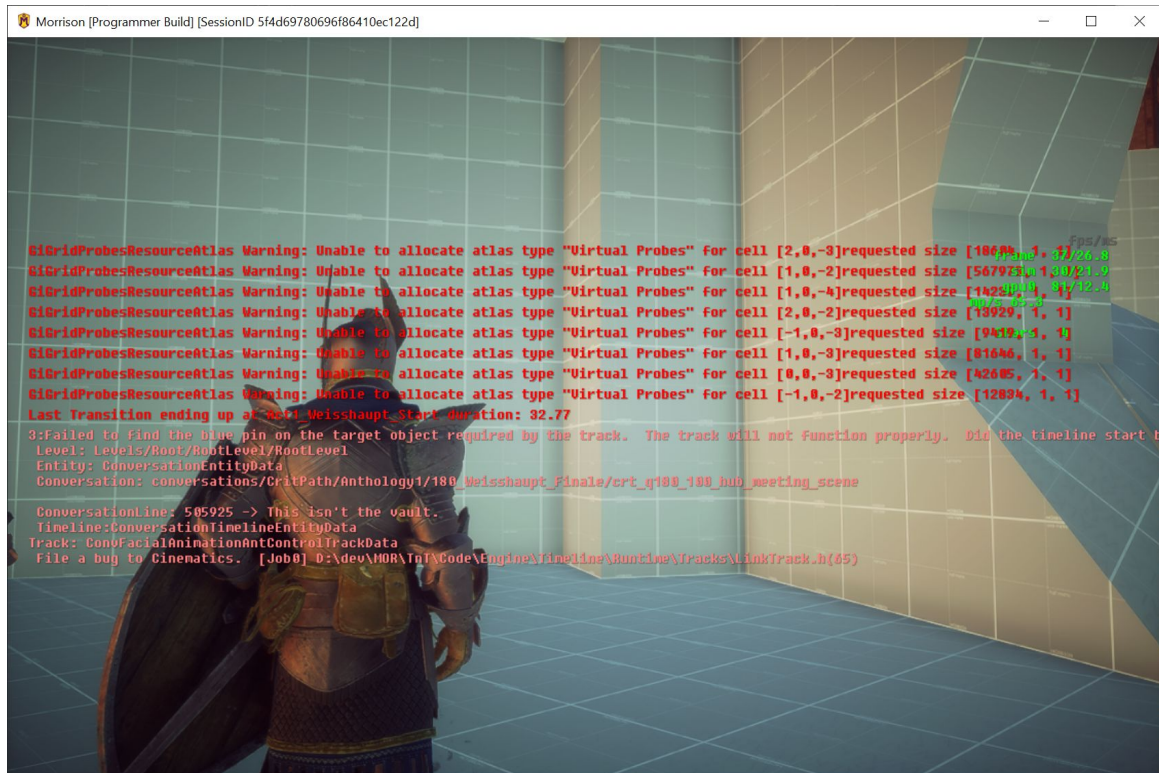
- Exposed 'high level heap' tuning
- Easy reloadable
- Not very obvious but fast iteration



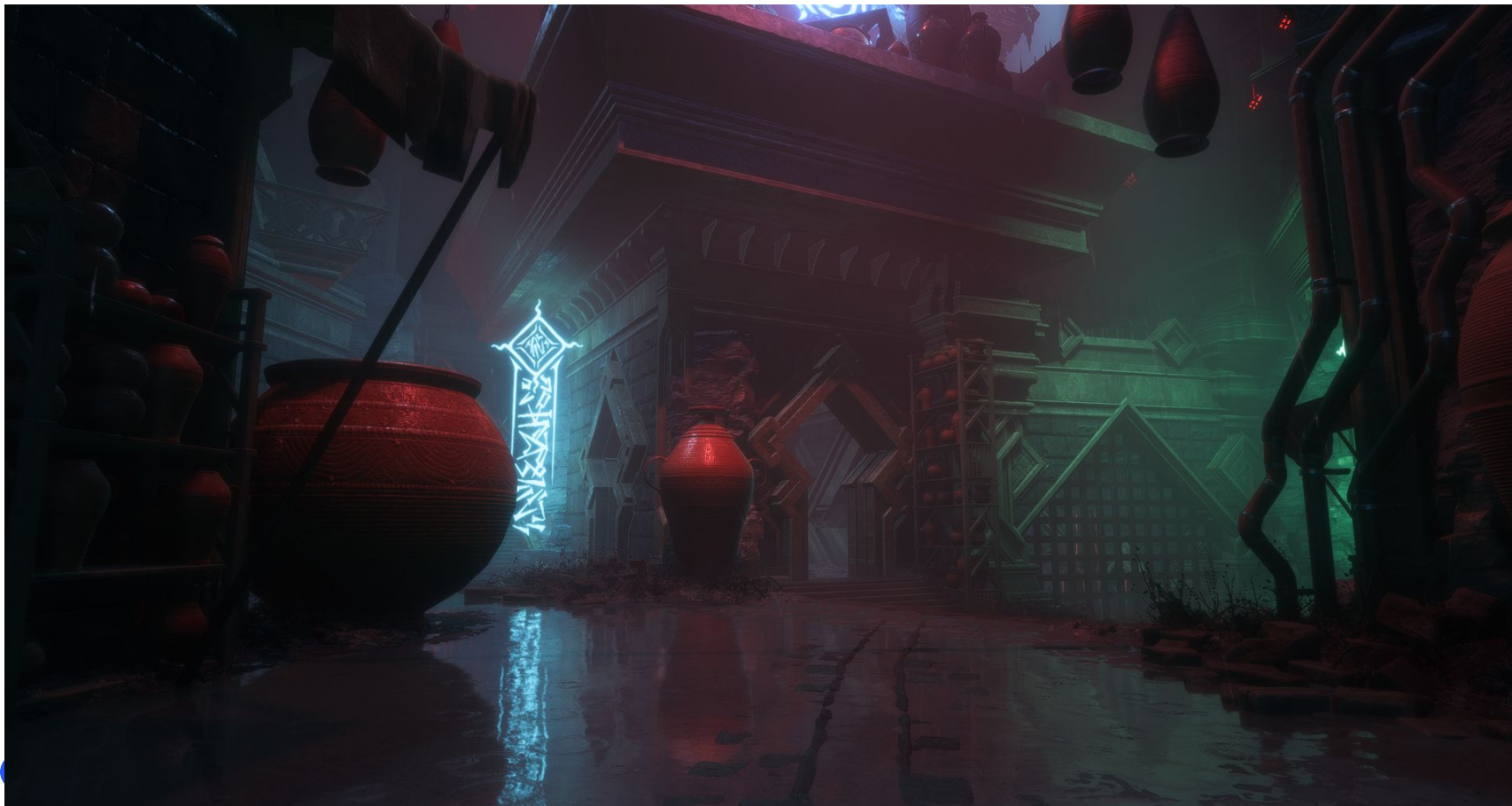
Global Illumination

Streaming failures

- Fallback to higher lod
- Mode to warn content creator



Ray-Traced Reflections



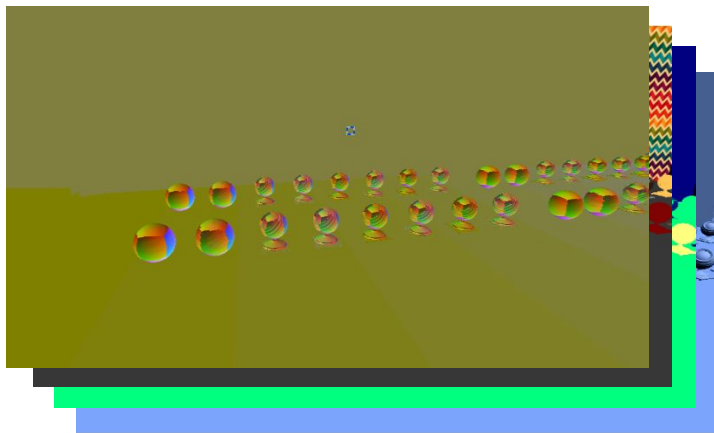
Ray-Traced Reflections

Previous work:

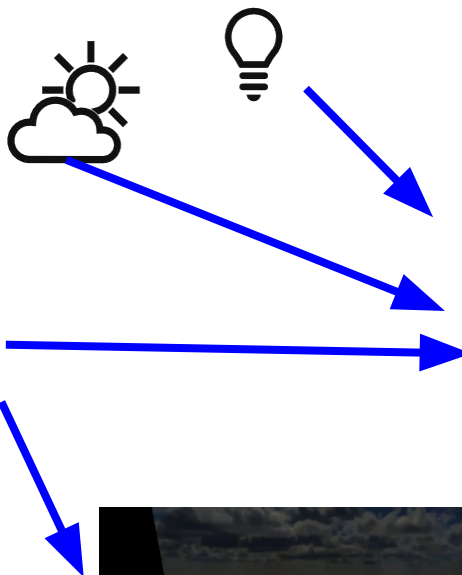
- It Just Works: RT Reflections in Battlefield V: <https://www.youtube.com/watch?v=ncUNLDQZMzQ>
- Hardware Ray Tracing as a first-class citizen in Frostbite:
<https://www.nvidia.com/en-us/on-demand/session/gtcfall21-a31411/>
- Global Illumination Based on Surfels (GIBS):
<https://www.ea.com/seed/news/siggraph21-global-illumination-surfels>
<https://advances.realtimerendering.com/s2024/index.html#gibs2>

Ray-Traced Reflections

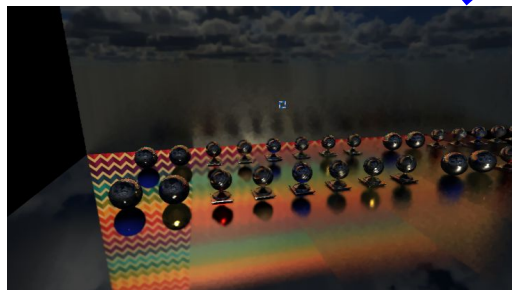
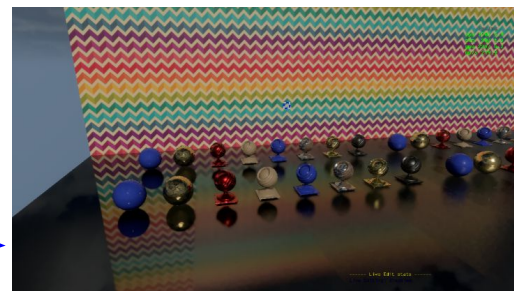
"It Just Works" recap



GBuffer



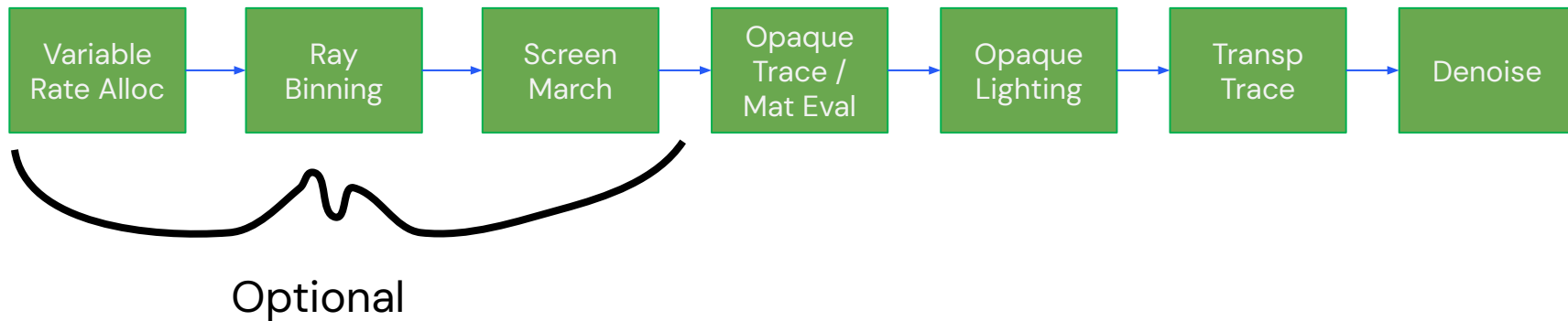
Final Result



RT Reflections

Ray-Traced Reflections

"It Just Works" Recap, pt 2



Ray-Traced Reflections

Dragon Age Challenges:

- Original RT solution not cross-platform
- Original solution built on old engine
- Upsampler vs checkerboard?
- Limited art and engineering resources
- Denoiser performance
- Original solution evaluated materials in closest hit
- No participating media volume support

Ray-Traced Reflections

Strategic decisions:

Consoles:

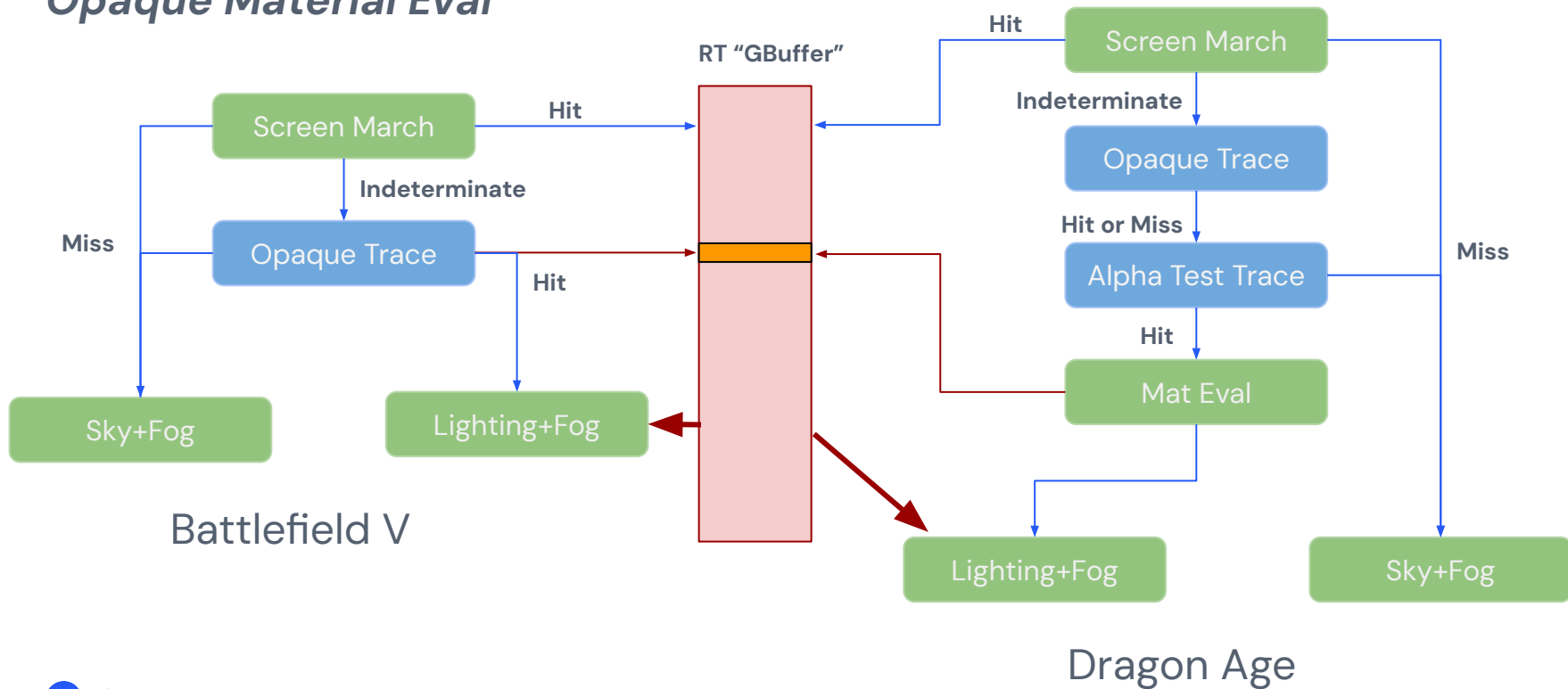
- Ray Tracing only supported in “Fidelity” mode (30 fps)
- Only one RT feature active at a time – RT Reflections or RTAO
- Half Res Support

PC:

- “Ultra” modes could have everything
- User configurable

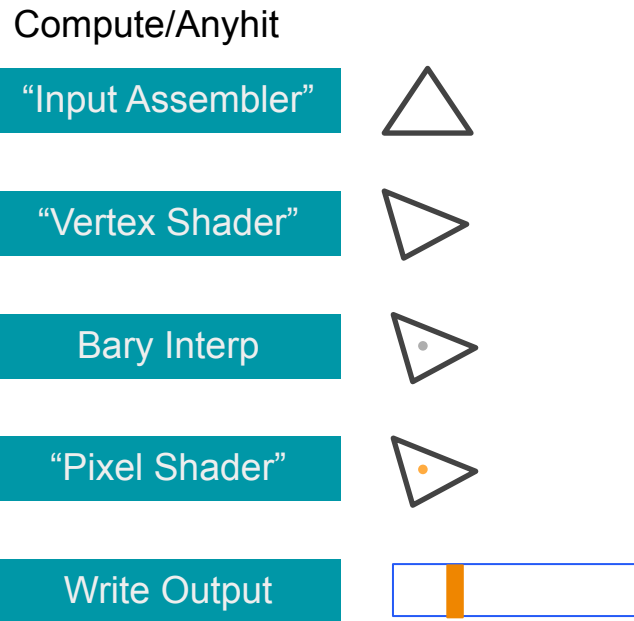
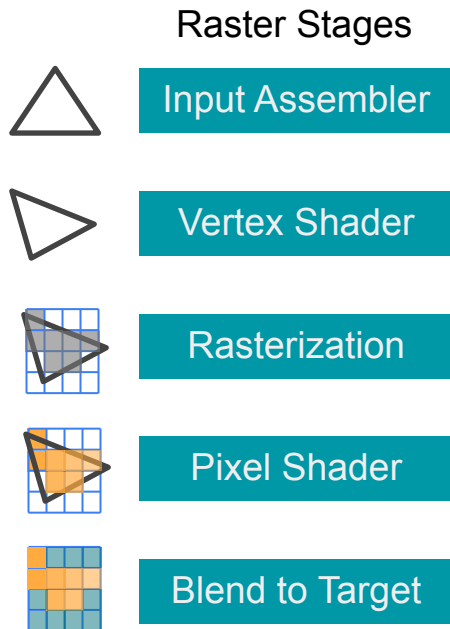
Ray-Traced Reflections

Opaque Material Eval



Ray-Traced Reflections

Compute/Anyhit



Ray-Traced Reflections

Here be dragons....

```
// 'Input Assembler' stage
IaOutput iaOut = runInputAssembler(primId, instId, startIndex);

// 'Vertex shader' and 'Raster' stage
float3 hitPos = rayOrigin + hitT * rayDirection;
VsOutput vsOut = runInterpolation(iaOut, hitPos, bc3);

// 'Pixel shader' stage
PsOutput psOut = runPixelShader(vsOut, instId, primId, hitPos);
```

```
IaOut runInputAssembler(uint primId, uint instId, uint startIndex)
{
    IaOutput iaOutput;
    for (uint edgeId = 0; edgeId < 3; ++edgeId)
    {
        // Generated code unpacks vertex data from stream(s)
    }
}
```

```
VsOutput runVertexMain(VsInput inputs)
{
    VsOutput vsOut;
    vsOut.pos      = skinPosition(inputs.pos); // 3 vgrs
    vsOutput.norm   = skinNormal(inputs.norm); // 3+3 vgrs
    vsOutput.tangent = skinNormal(inputs.tang); // 3+3+3 vgrs
    vsOutput.binorm  = skinNormal(inputs.binorm); // 3+3+3+3 vgrs
    vsOutput.uvs     = animateUVs(inputs.uvs); // 3+3+3+3+2 vgrs

    // 14 vgrs used to store 1 vertex!
    Return vsOut
}
```

```
VsOutput runInterpolation(IaOutput iaOut, float3 hitPos, float3 bc)
{
    VsOutput vsOut0 = runVertexMain(iaOut.v[0]);
    VsOutput vsOut1 = runVertexMain(iaOut.v[1]);
    VsOutput vsOut2 = runVertexMain(iaOut.v[2]);

    VsOutput vsOut;
    vsOut.position = ...; // barycentric interpolation goes here...
    // ...

    return vsOut;
}
```

42 VPGRs allocated!!!!

Ray-Traced Reflections

Worst-case scenario

```
struct VsOutput
{
    float4 h12BasisL0;
    float4 h12BasisL1;
    float4 h12BasisL2;
    half4 hPos;
    float4 preMulAlphaFog;
    float3 color;
    half3 normal;
    float3 keyDirection;
    float2 texCoords1;
    float2 alphaLevelsMinMax;
    float2 texCoords1Normalmap;
    float alpha;
    float shadow;
    float alphaLevelsExponent;
    float softParticleInvFadeDist;
    float posZ;
    float type;
    float normalBlend;
};
```


Ray-Traced Reflections

VGPR Strategies

Compute and Anyhit (Opaque Alpha Test)

- Opaque Alpha Test as a separate TLAS and trace
- Use raytrace branch node for worst-offenders
- Root node switch for opaque-ish materials (camera stippling tricks, custom lighting models)

Anyhit (Transparent)

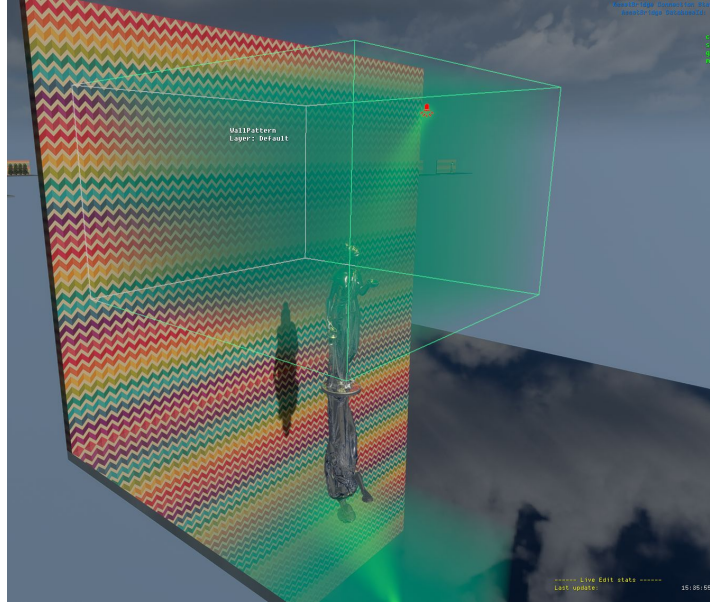
- Rewrote emitter “stack” code gen for RT (legacy VFX system)
- Separate CS pass caches vertex data for emitter “graph” (modernized VFX)

Future Work

- Vertex Data Cache for animated geo (a la [GDC Vault - Large Scale GPU-Based Skinning for Vegetation in 'Alan Wake 2'](#))
- Explore code gen for moving from VS-→PS (<https://advances.realtimerendering.com/s2024/#serac>)

Ray-Traced Reflections

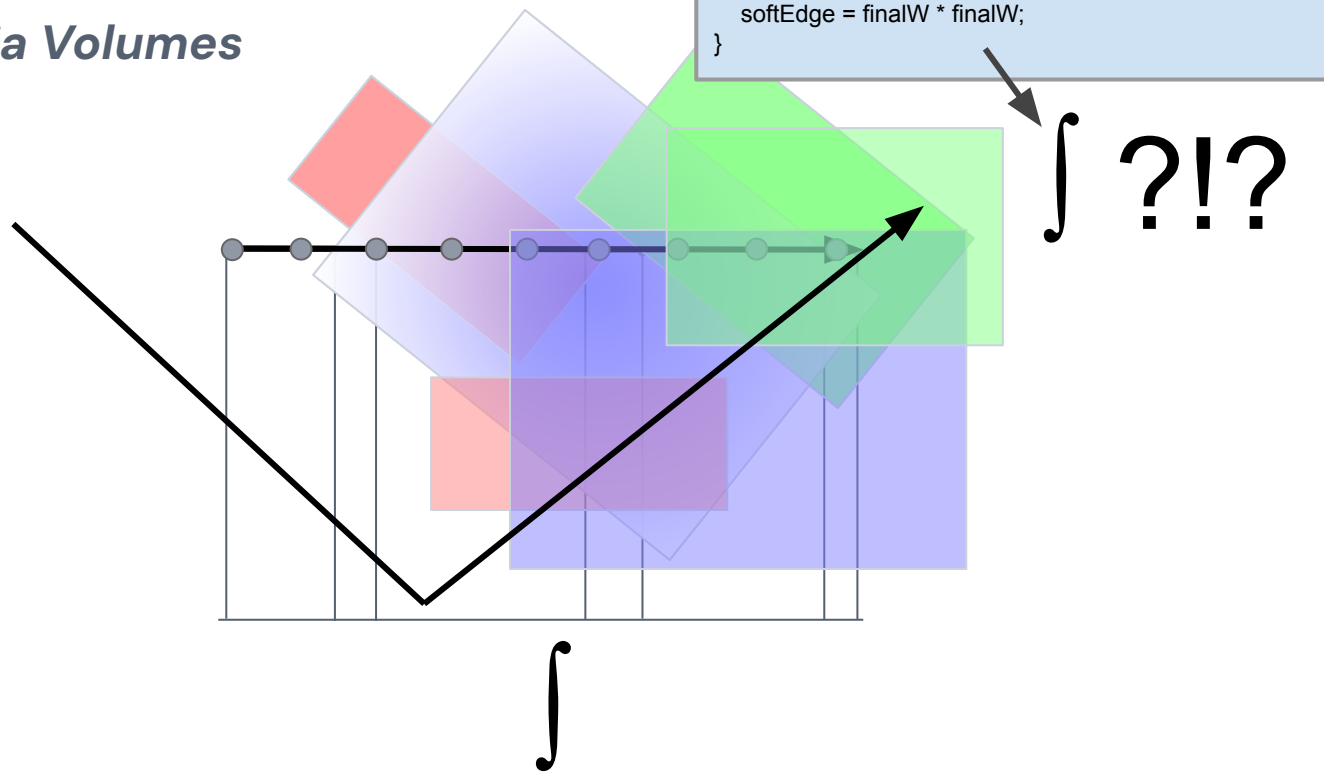
Participating Media



<https://www.ea.com/frostbite/news/physically-based-unified-volumetric-rendering-in-frostbite>

Ray-Traced Reflections

Participating Media Volumes



Ray-Traced Reflections

Denoiser

- Inspired by Nvidia's RELAX denoiser:
<https://www.nvidia.com/en-us/on-demand/session/gtcspring21-s32759/>
- But....different...
 - Tile-based
 - Ray reuse pass
 - Fewer temporal samples
 - Disocclusion?
 - Reactive mask

Character Creator



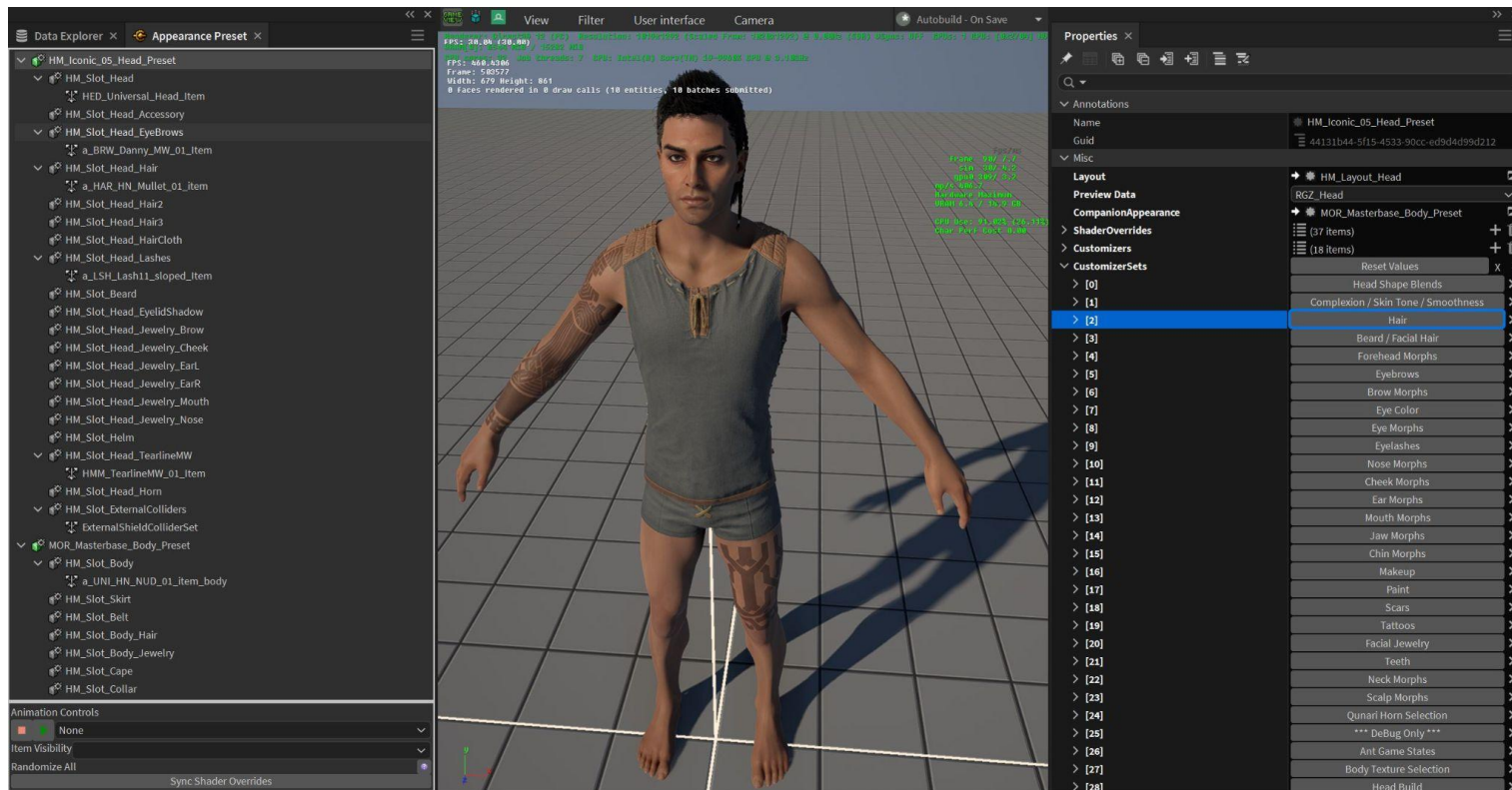
Character Creator



Goals for Character Creator

- Push the boundary on number of customization options.
- What you see in character creator is what you get in game.
- Optimal performance within memory budget.
- In-game character creator should be subset of dev tools.
- Reuse the system for infill characters

Appearance Editor



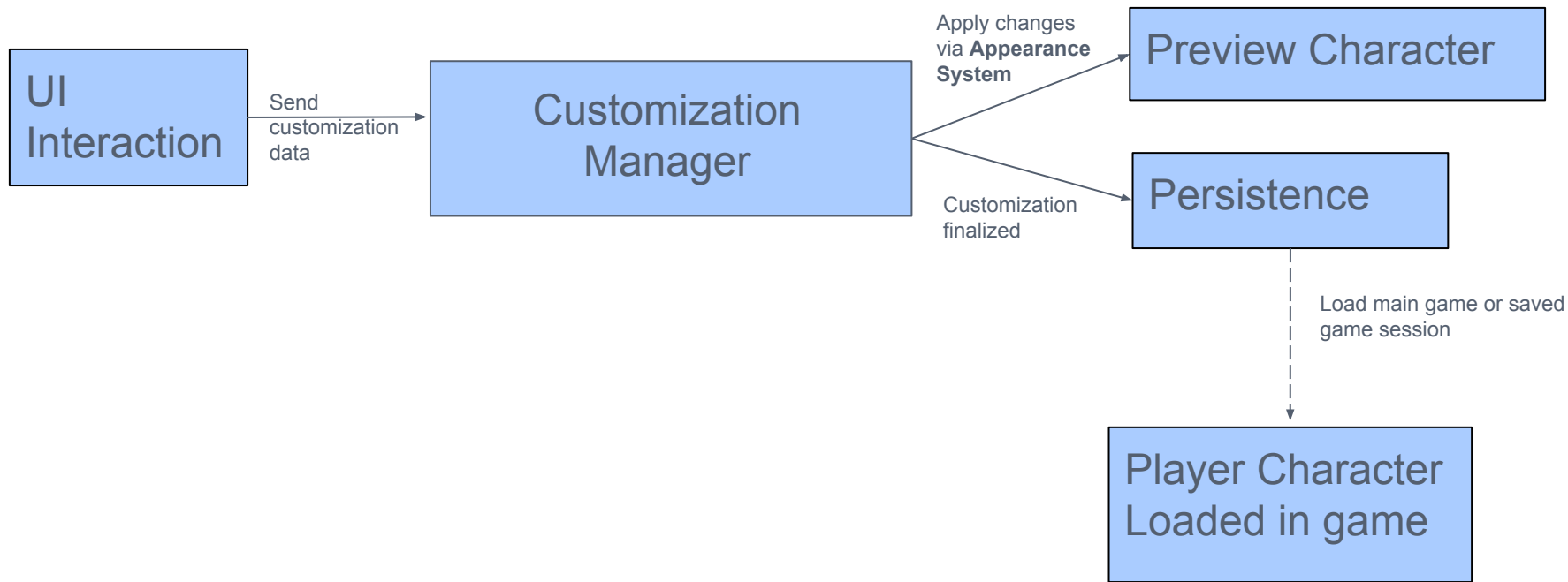
What to customize

Customizations are divided into following categories:

- Shader Params
- Morph Features
- Head Blends
- Texture Sets
- Appearance Items

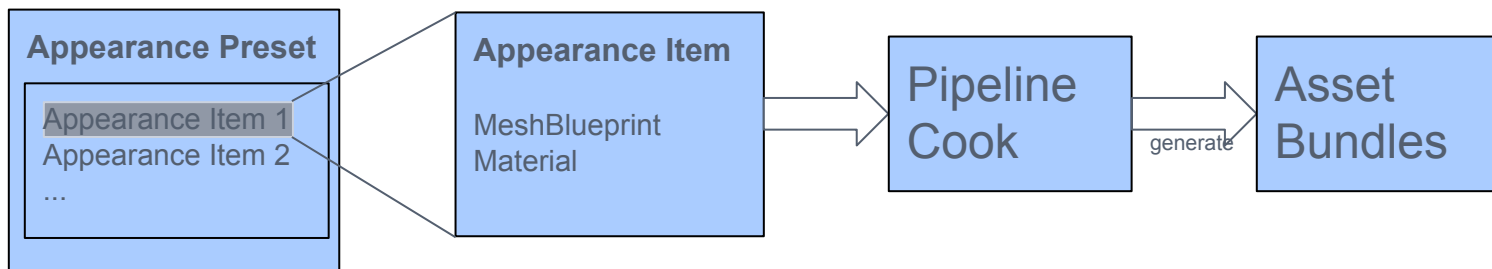


Character Creator Architecture

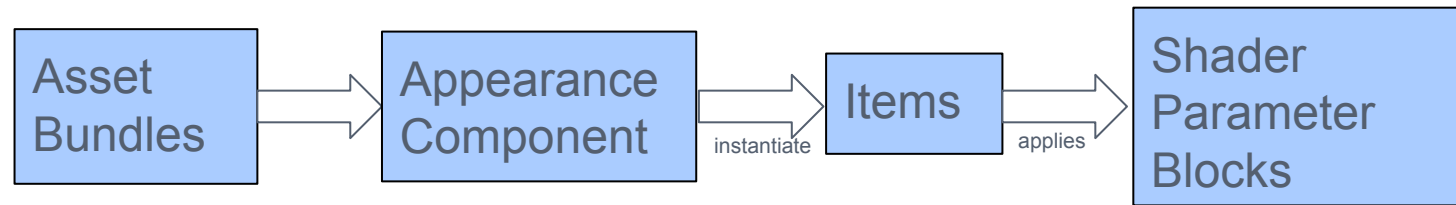


Appearance System

Asset Pipeline



Runtime



Appearance System

Appearance system is used for all character rendering that can have variations in appearance like player character, followers & infill characters.

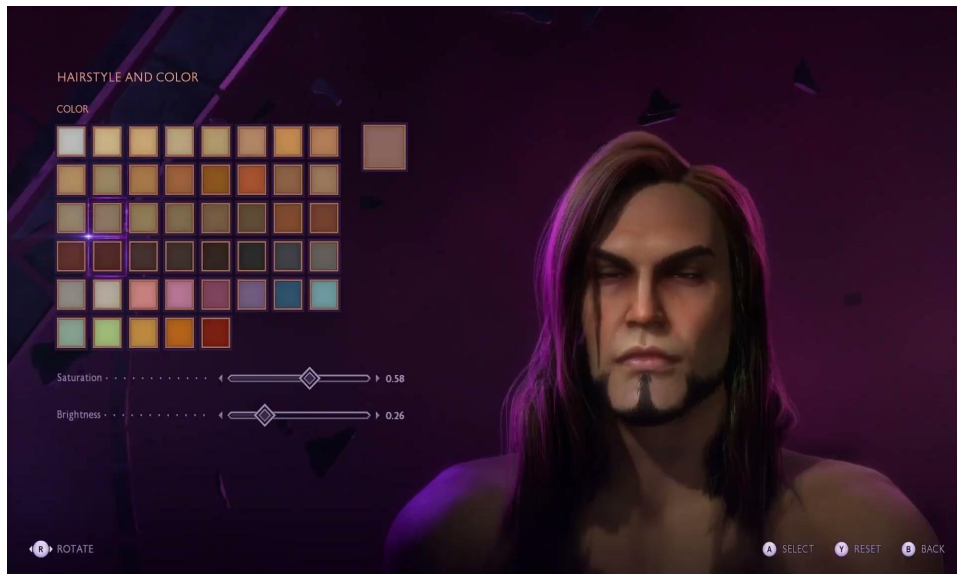
- Character Creator
- Gear System
- Infill Characters

Types of NPCs in game:

- Full NPCs
- Light NPCs
- Static NPCs
- Swarm NPCs

Shader Parameters

- Change applied to materials via shader parameters.
- Examples: skin tone, hair color, eye color, makeup etc.
- The shader graphs are set in such way that it allows all values in different character shaders are updated for one parameter name.
 - Allows matching face and body skin tone without separate process.
- All shader changes are applied as one shader parameter block to minimize performance hit.
- Shader parameter blocks can be stacked.



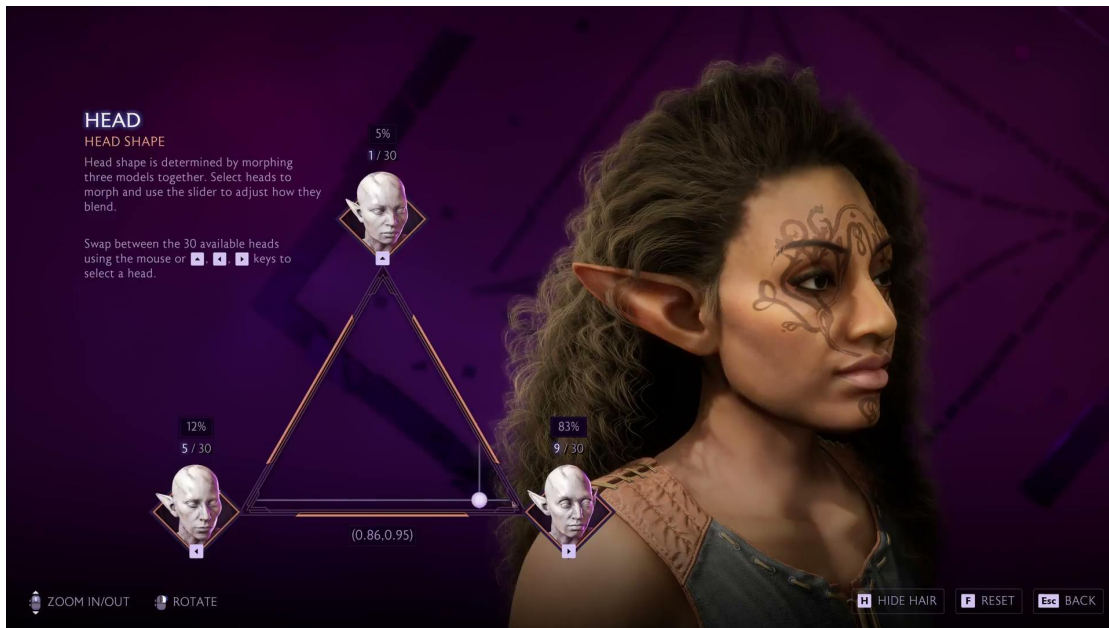
Morph Features

- Changes to facial features and body builds.
- The weights dictated how the joints in face and body rig can be changed to produce desired features.
- Weight ranges can be mapped from absolute values on rig to -1 to 1 in game.



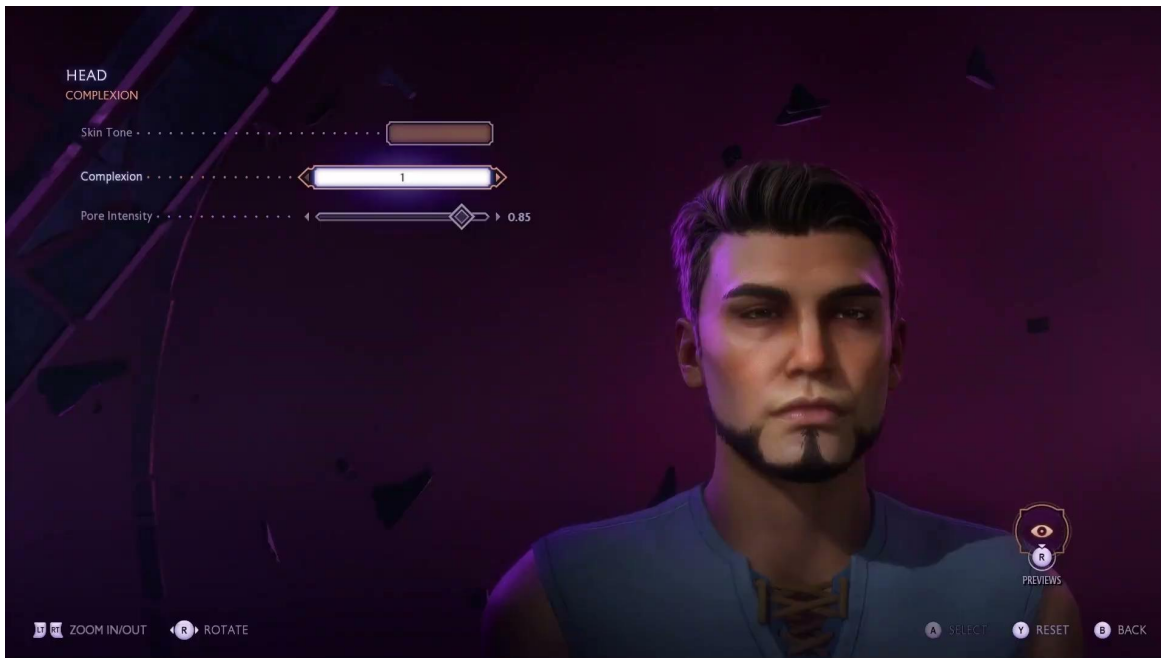
Head Blends

- 3 preset head blend shapes are interpolated between each other.
- Each race has its own collection of head blends created by artists.
- Only limited to heads, due to heavy computation.
- Memory consumption can increase based on number of head assets.
 - Solution: only keep the assets bundles that are actually used in the character and avoid duplication of asset bundles.



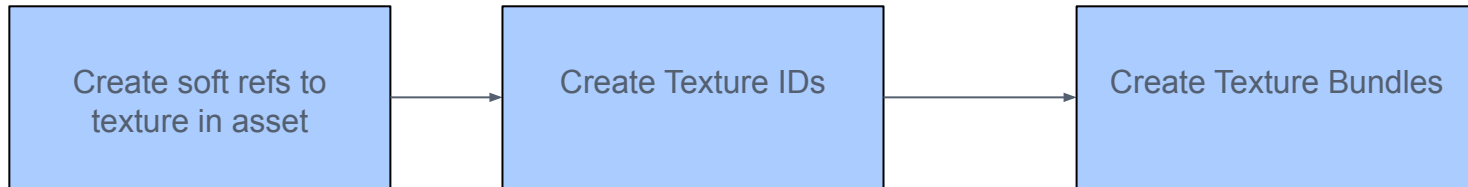
Texture Sets

- Used for complexion sets, makeup, tattoos, paint and scars in face and body.
- Higher memory consumption leading to lower resolution streaming textures.
 - Solution: unused bundles are not loaded into the memory.
 - Use of soft refs in the assets instead of hard references to texture assets.

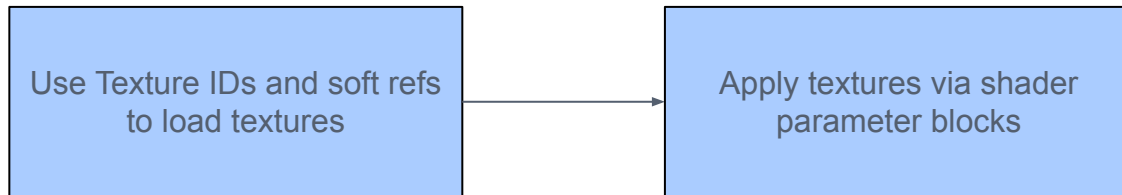


Texture Bundles

Pipeline

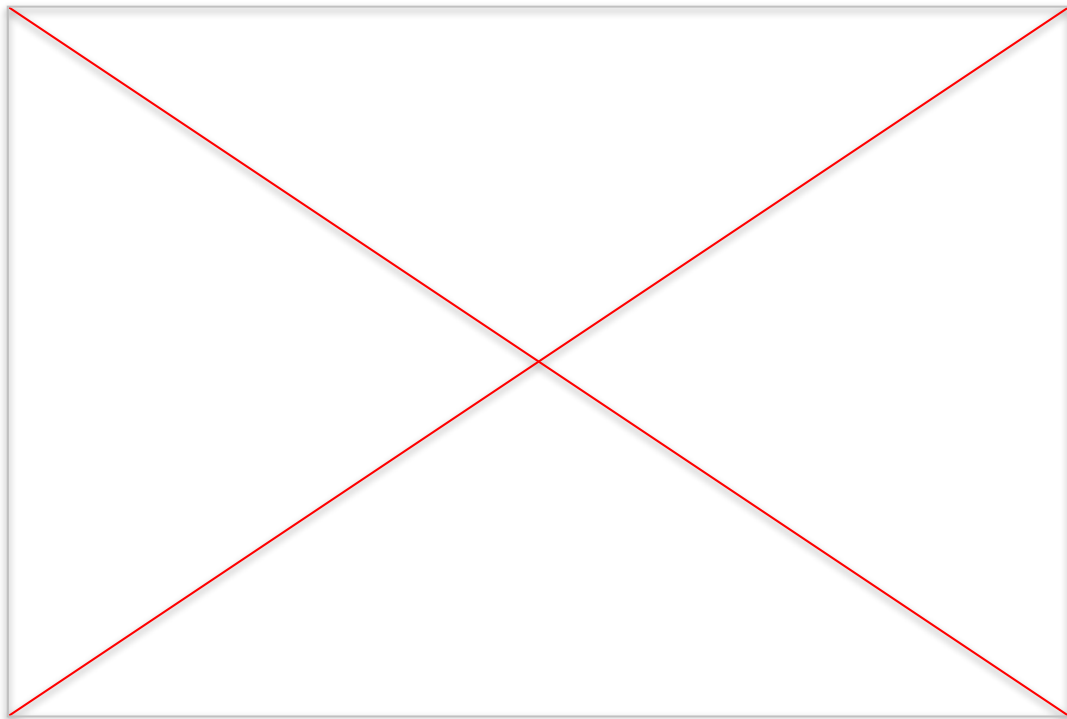


Runtime



Appearance Items

- Head presets, body presets, hair, eyebrows, eyelashes, qunari horns and gear are all appearance items.
- Appearance items automatically applies all the shader overrides as setup in data.
- Async loading of items helped in achieving better performance.
- Item bundles had to be unloaded after each category was done to keep memory within budget.



Takeaways

- Start development of character customization early.
- Start with dev tools to see what customizations are required for character design.
- Refine the options based on the impact in game. E.g. nail polish options was abandoned.
- Measure impact vs effort. Efforts for jewelry on head body blend shapes was too high with lesser impact.
- Keep an eye on memory and make adjustments.

Lighting Strand Hair

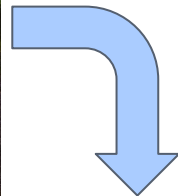
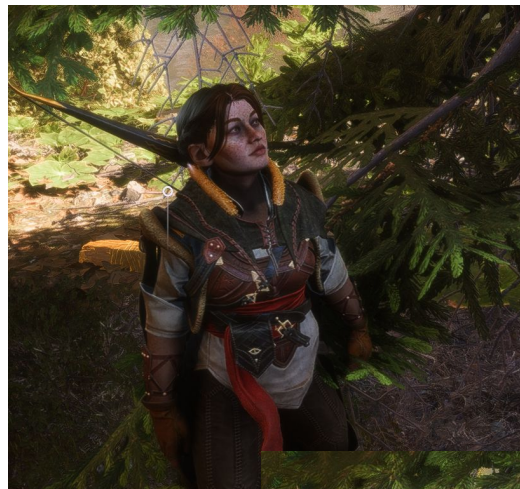
Problems:

- Varied fantasy visual environments.
- Lighter hair color tone was prone to unexpected changes.
- The hair lighting model has no diffuse component, it's purely specular

Solutions:

- Fake character lights.
- At least one shadow casting light had to be present.
- LRVs and sky sampling for correct color.

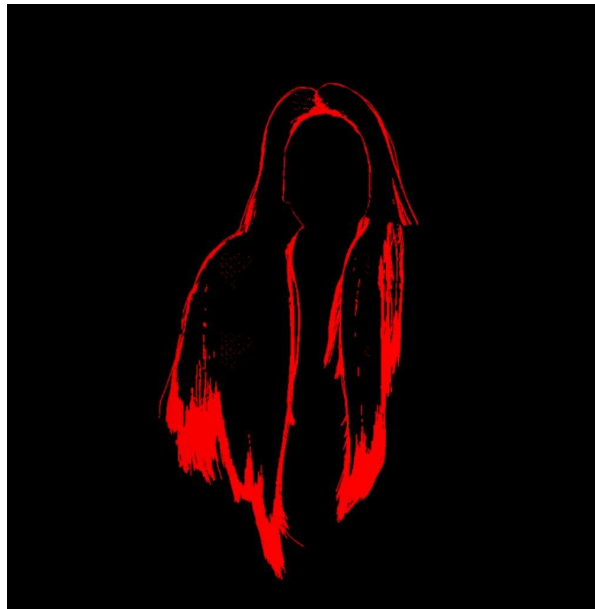
For more details check out: <https://www.ea.com/technology/news/strand-hair-dragon-age-the-veilguard>



Lighting Strand Hair

A new technique for blending hair with transparent VFX and participating media.

- Split the hair into two distinct passes, first opaque, and then transparent.
- Add alpha cutoff to the render pass that composites the hair with the world and first renders the hair that is above the cutoff (opaque), and then below the cutoff (transparent).
- Before these split passes are rendered, we render the depth of the transparent part of the strand hair which is used as a spatial barrier between transparent pixels that are “under” and “on top” of the strand hair.



Lighting Strand Hair



For more details check out: <https://www.ea.com/technology/news/strand-hair-dragon-age-the-veilguard>

Lighting Strand Hair

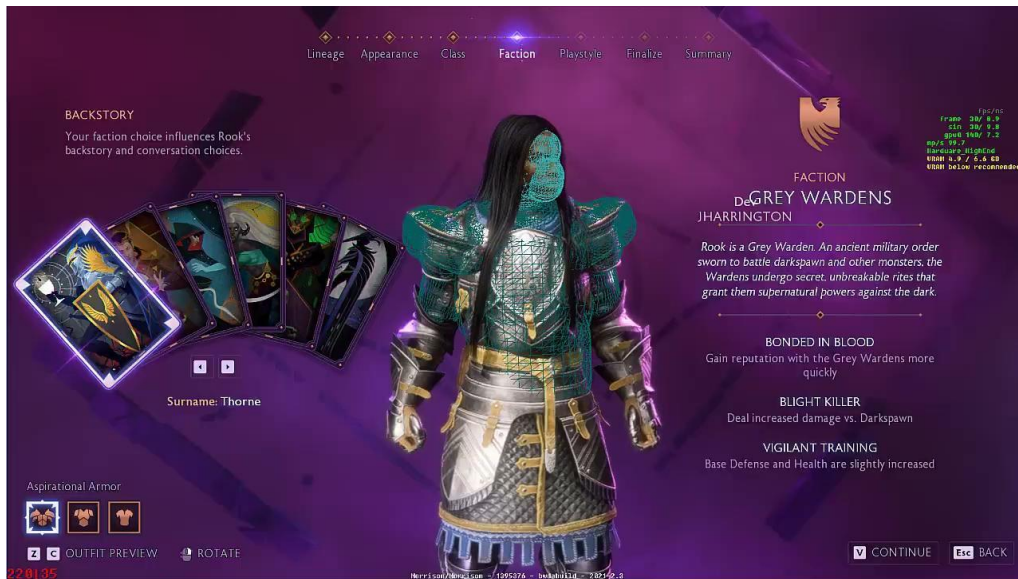
- High quality shadow maps are required in order to have good coverage of the tens of thousands of thin hair strands in the shadow map texture
- Hero shadows are rendered for every Strand Hair object and light designated as important to the shot.
- These hero shadows are generated at run time, using a tightly fitting light frustum that is adjusted to each hair's bounding box.
- As an optimization, check regular shadow maps first and if no hit, then sample strand hair hero shadows.



For more details check out: <https://www.ea.com/technology/news/strand-hair-dragon-age-the-veilguard>

Collider Sets

- Collider sets are used for preventing hair clipping with character outfits.
- Since the main player model proportions can be customized, the colliders also needed to fit the customization
- Colliders were moved based on skeletal joint transforms



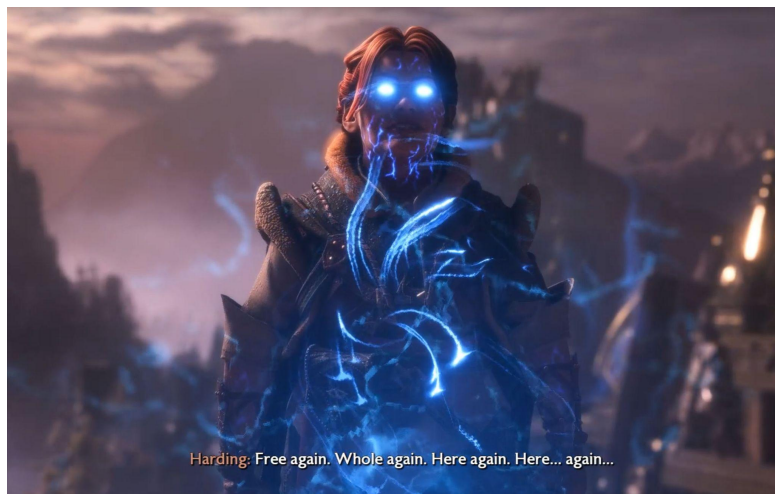
Strand Hair Performance

- **Rendering budget:**
 - 8 strand hair assets at once
 - 6.5ms for 33.3ms frametime and 3ms for 16.6ms frame time
 - Hair sim on GPU compute ~2 ms
 - Occasional spike when loading or teleporting characters.
- **Optimizations & Constraints:**
 - Only on PC high and ultra settings, PS5 and XBOX.
 - Only for head hair and beards. Eyelashes and eyebrows used card hair.
 - Only for named NPCs besides the player character.
 - Bundling the strand and card hair separately saved a lot of memory issues, especially on consoles.
 - Hair resolution control adjusts the hair resolution depending on upsampler and DRS settings.
 - Disable hair sim when off screen or far away from camera using the sim lods.
 - Decimation of render strands based on camera distance.



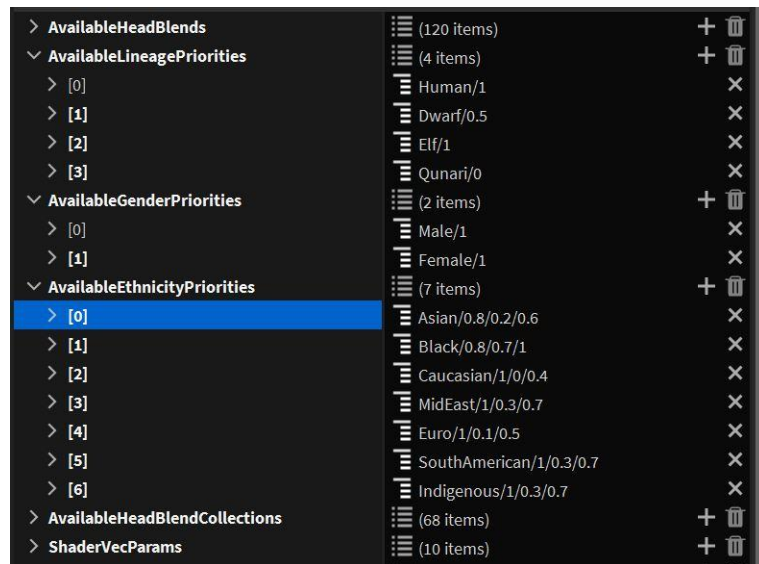
Shader Variation System

- Used for Damage Over Time effects like Poison/Burn/Freeze on players and NPCs
- Built to utilize the shader parameter blocks.
- These also can be stacked on top of the other shader customizations.
- Avoids SRV limitations and shader permutations bloat.
- Also useful for showing injuries or blighted states on character.



Infill Characters

- Values are randomized based on game lore constraints.
- These are specified in a infill character database and priority value is assigned based on required gender/lineage/ethnicity for different regions game.
- The database is parsed through the during pipeline stage and appropriate key value pairs are made from ID and asset bundles.
- At runtime, based on priority values, appropriate asset bundles are loaded and characters are rendered.
- The appearance system was reused to allow for variations in the infill characters but did not customize at runtime.



> AvailableHeadBlends	(120 items)	+
> AvailableLineagePriorities	(4 items)	+
> [0]	Human/1	x
> [1]	Dwarf/0.5	x
> [2]	Elf/1	x
> [3]	Qunari/0	x
> AvailableGenderPriorities	(2 items)	+
> [0]	Male/1	x
> [1]	Female/1	x
> AvailableEthnicityPriorities	(7 items)	+
> [0]	Asian/0.8/0.2/0.6	x
> [1]	Black/0.8/0.7/1	x
> [2]	Caucasian/1/0/0.4	x
> [3]	MidEast/1/0.3/0.7	x
> [4]	Euro/1/0.1/0.5	x
> [5]	SouthAmerican/1/0.3/0.7	x
> [6]	Indigenous/1/0.3/0.7	x
> AvailableHeadBlendCollections	(68 items)	+
> ShaderVecParams	(10 items)	+

Infill Characters Trade-offs

- Used 2 head blends instead of 3 as in Full NPCs.
- Stop blending computations at long camera distance.
- No feature morphs were used because camera never gets as close.
- Card hair only for infill NPCs.
- Shader params subset was used. For example: no makeup for infill characters.
- Texture sets subset was used for complexion.
- 1-mip offset for all textures on platforms with memory constraints was used.
- Did not make separate low poly models because LODs were usually enough for reducing the geo complexity at that distance from camera.
- Division of data by region allowed to load/unload assets as needed for that region.
 - Also provides better content management for artists

Infill Characters



Thank You!

The *Dragon Age: The Veilguard* graphics team:

James Power, Patrick Chan, Thomas Roy, Wyatt Hammond, Sam Moore(Performance Lead) & Ben McGrath(Lighting Director)

Frostbite Rendering team:

Matthias Moulin, Carlos Macarron, Matti Hietanen, Kyle Hayward, Jon Valdes, Filipe Amim, Diede Apers, Viktor Alm, Yasin Uludag, Robin Taillandier, Leo Taslaman, Sylvain Meunier, Brad Loos, Mick Beaver

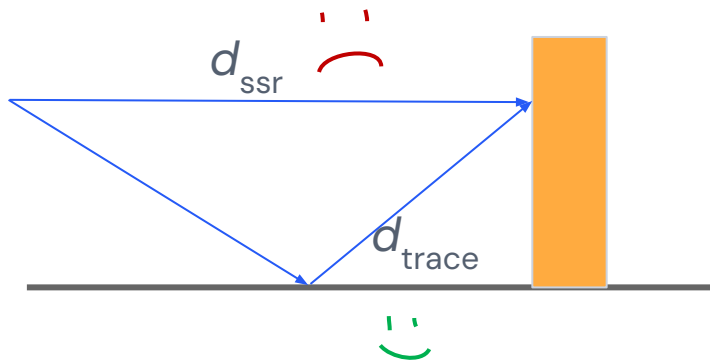
Q&A

Bonus slides

Ray-Traced Reflections (bonus)

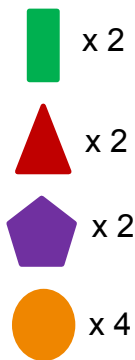
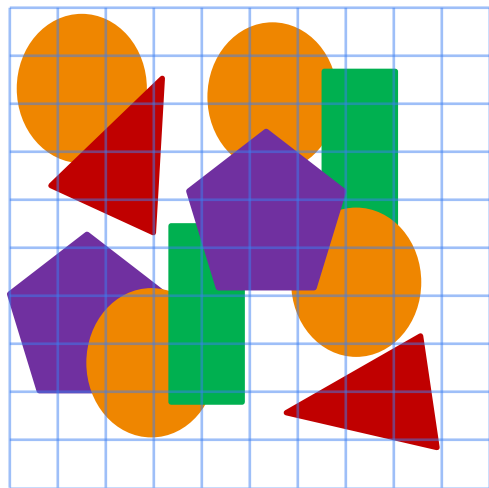
Participating Media - SSR vs RT

$$I = I_0 e^{-\mu d} \quad (\text{From } \text{Beer-Lambert law} - \text{Wikipedia})$$



Ray-Traced Reflections (bonus)

Opaque Material Evaluation



TLAS Instance ID Lookup Tables

ID->Mat/Mesh

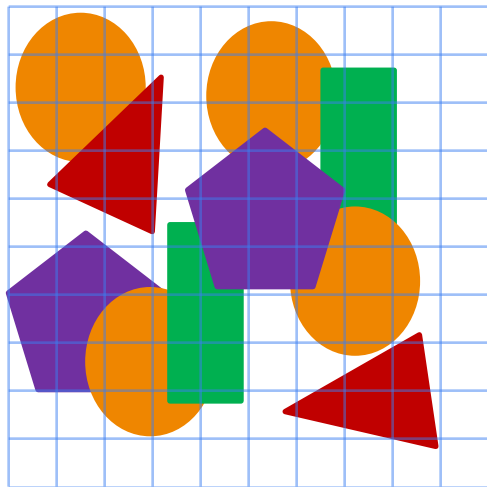
0	3
1	3
2	0
3	1
4	3
5	2
6	2
7	1
8	3
9	0

ID->Inst Index

0	0
1	1
2	0
3	0
4	2
5	0
6	1
7	1
8	3
9	1

Ray-Traced Reflections (bonus)

Sorting Opaque Hits



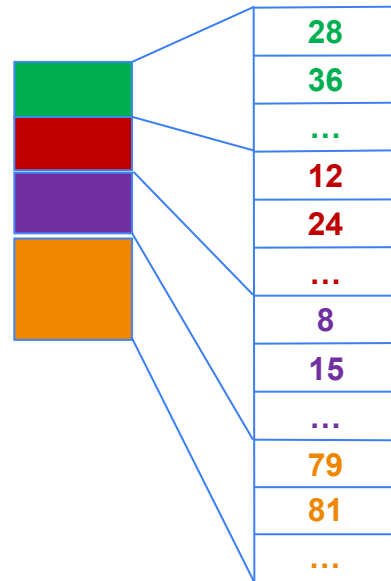
Count Hits

0	11
1	8
2	14
3	21

Prefix Sum

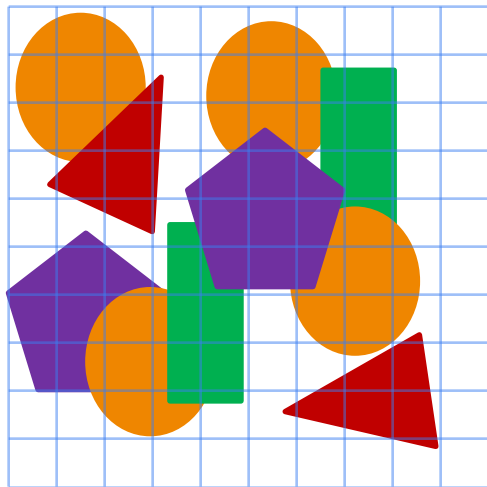
0	0
1	11
2	19
3	33

Batch Instance → RayIndex



Ray-Traced Reflections (bonus)

Opaque Material Eval Dispatch



```
ExecuteIndirect(GreenMaterial);  
ExecuteIndirect(RedMaterial);  
ExecuteIndirect(PurpleMaterial);  
ExecuteIndirect(OrangeMaterial);
```

