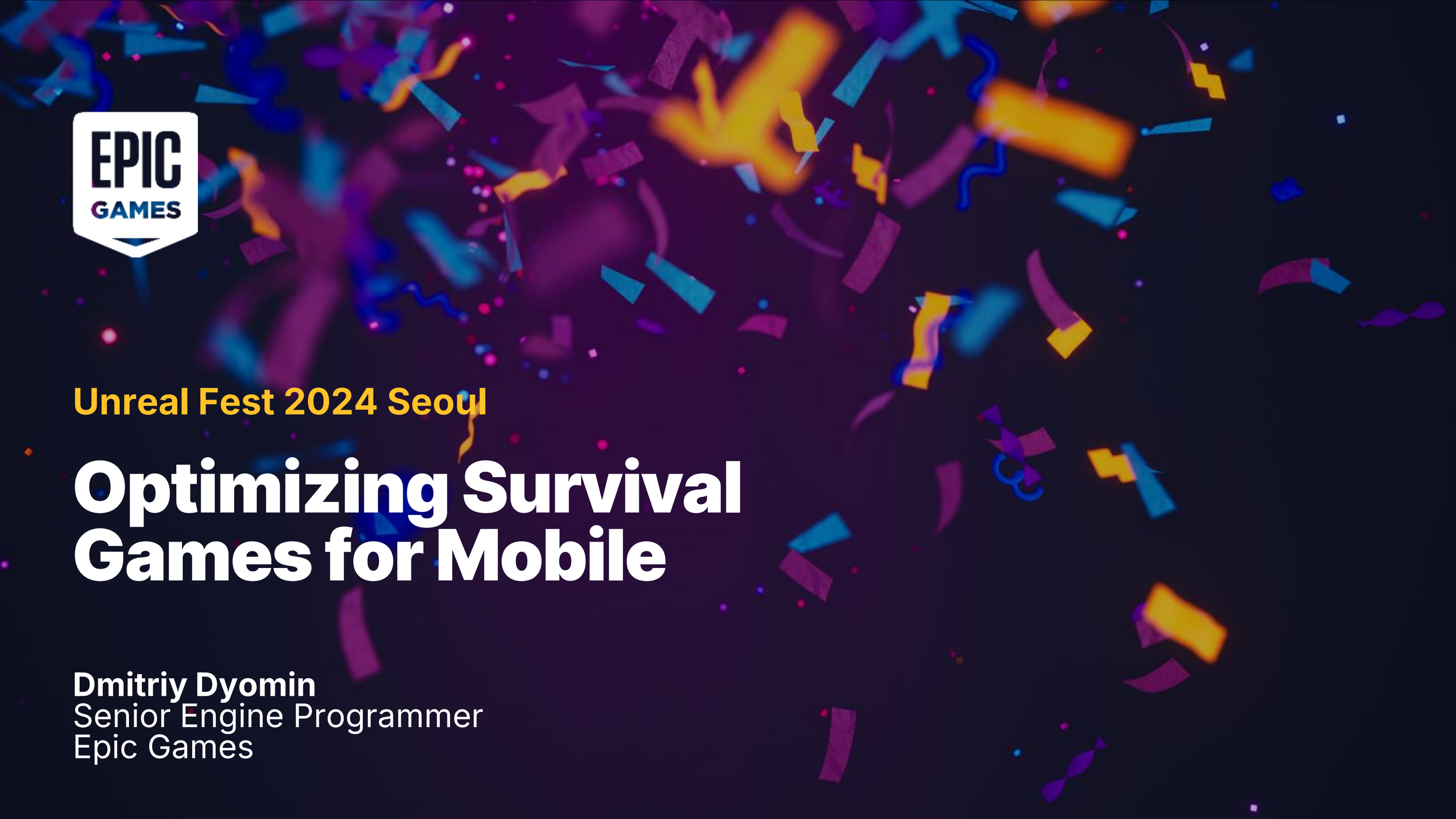


Unreal Fest 2024 Seoul

Optimizing Survival Games for Mobile

Dmitriy Dyomin
Senior Engine Programmer
Epic Games

Challenges We Hit

- Rendering heavily relies on instanced static mesh (ISM). Need to Improve culling/LOD for ISMs
- Landscape rendering performance
- Pitch-black nights, which require local lights to make it playable
- Runtime Pipeline State Objects (PSO)

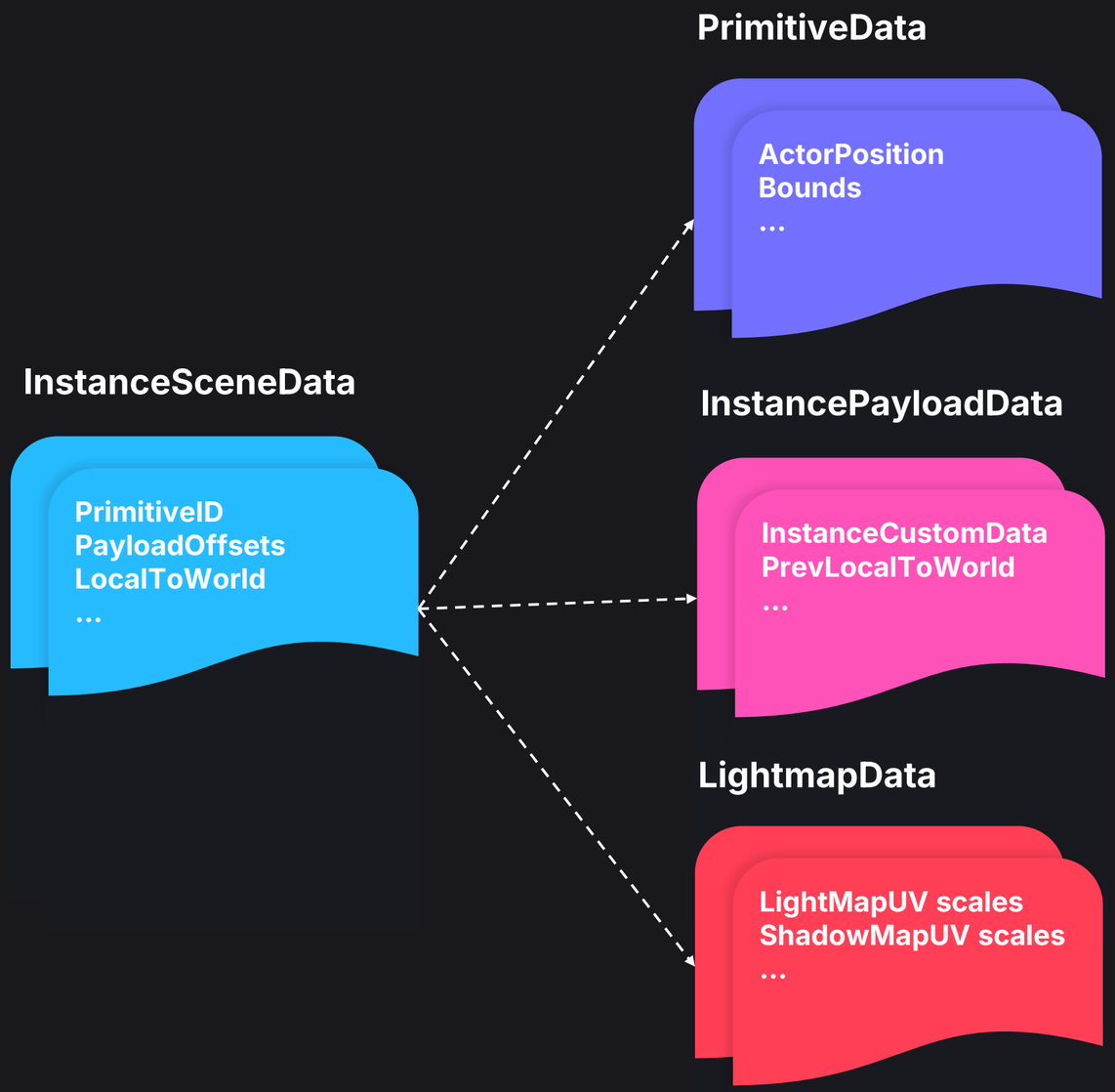
Rendering

- GPU Scene
- ISM GPU Culling and LOD
- Landscape GPU Culling
- Forward Lighting Optimization
- Runtime PSO pre-caching

GPU Scene Optimization

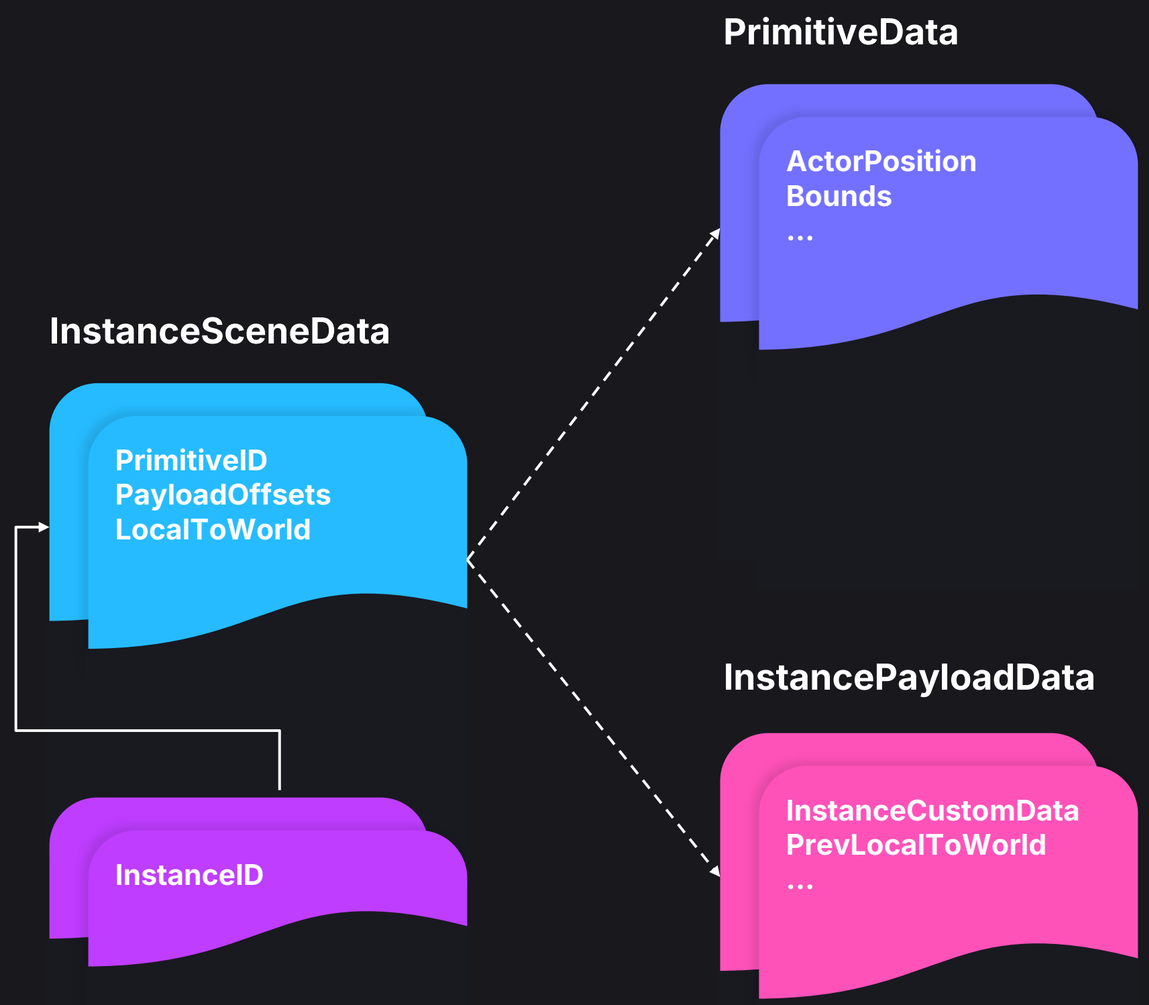
What is GPU Scene ?

- Collection of buffers
- Holds information about all primitives in the Scene
- Opens a way for a GPU Driven Rendering
- Auto-instancing



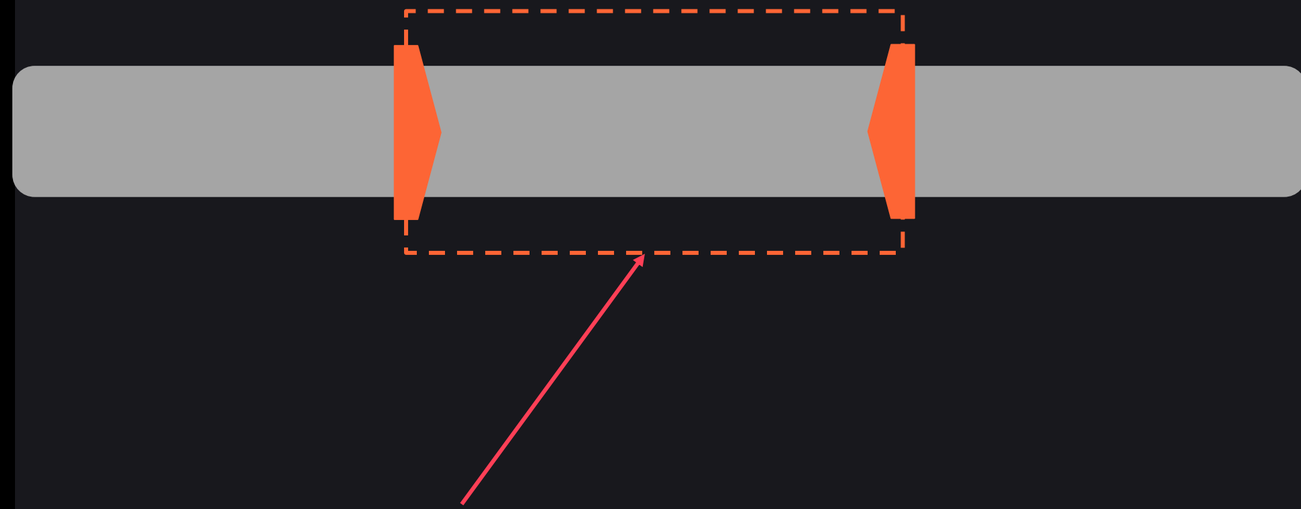
GPU Scene

- Desktop/console version runs a CS to generate a InstanceData buffer with IDs for visible instances.
- The classic implementation has up to 3 indirections. Expensive on mobile
- Some mobile GPUs do not support access to SSBO in a vertex shader



GPU Scene Optimization

- CS writes out InstanceData buffer for visible instances
  - All instance data that VS and PS might need to access
  - 512 bytes for non-ISM
  - 256 bytes per instance for ISM
- Bound as a ‘Uniform Buffer Object’ to VS and PS
  - Fast access
  - No indirections
  - Limited to 64KB on Switch and 16KB on Android
  - Up to 128 primitives or 256 instances in a single batched draw (Switch)
  - Submit batches in several draws if needed. Fast, no state changes
- Easy to switch between Desktop and Mobile specific implementations
  - DDPI - bSupportsUniformBufferObjects=true



glBindBufferRange(... Offset, Size)

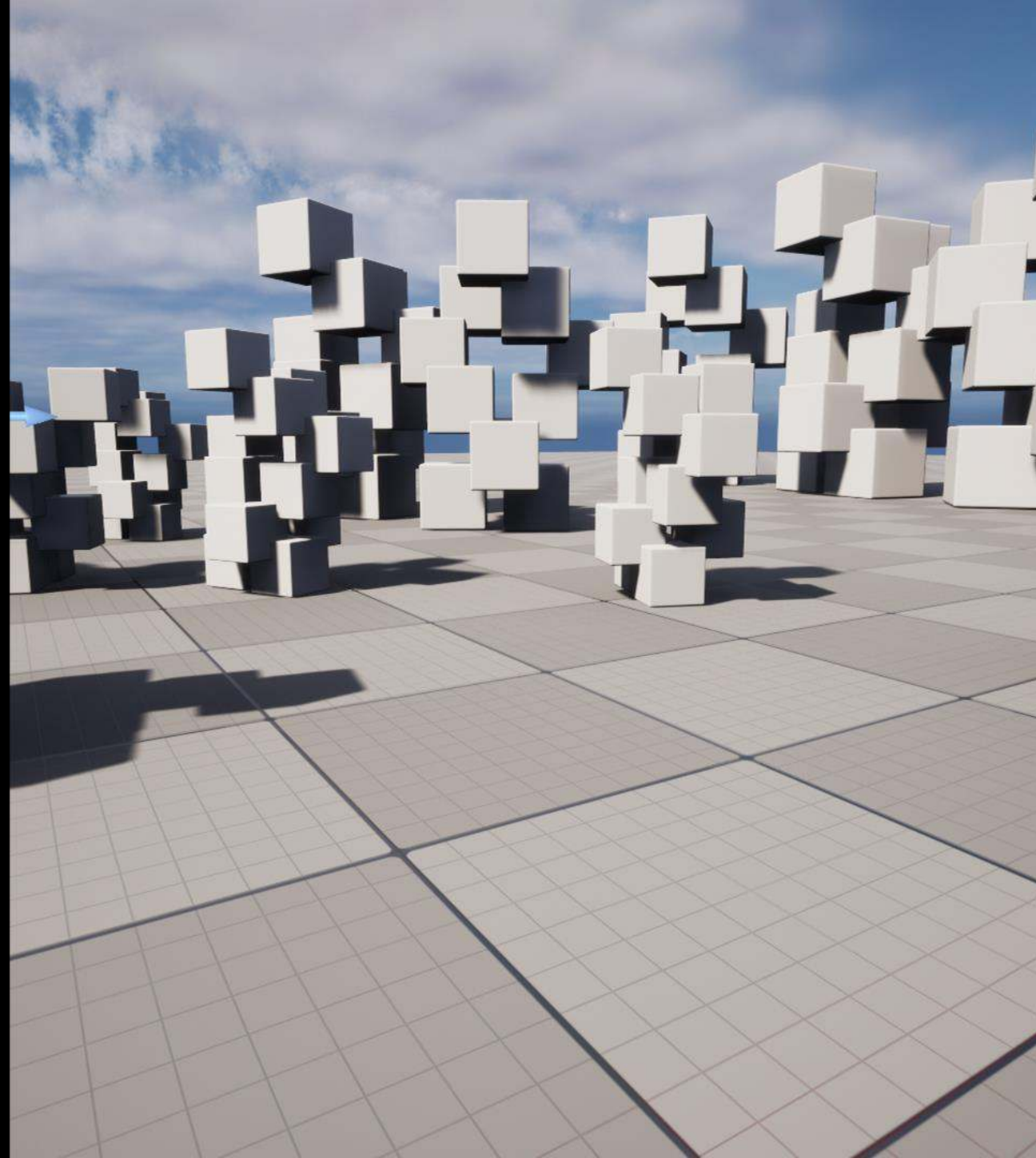
vkCmdBindDescriptorSets(... DynamicOffsets)

GPU Scene
Conclusion

So, how do you enable it?

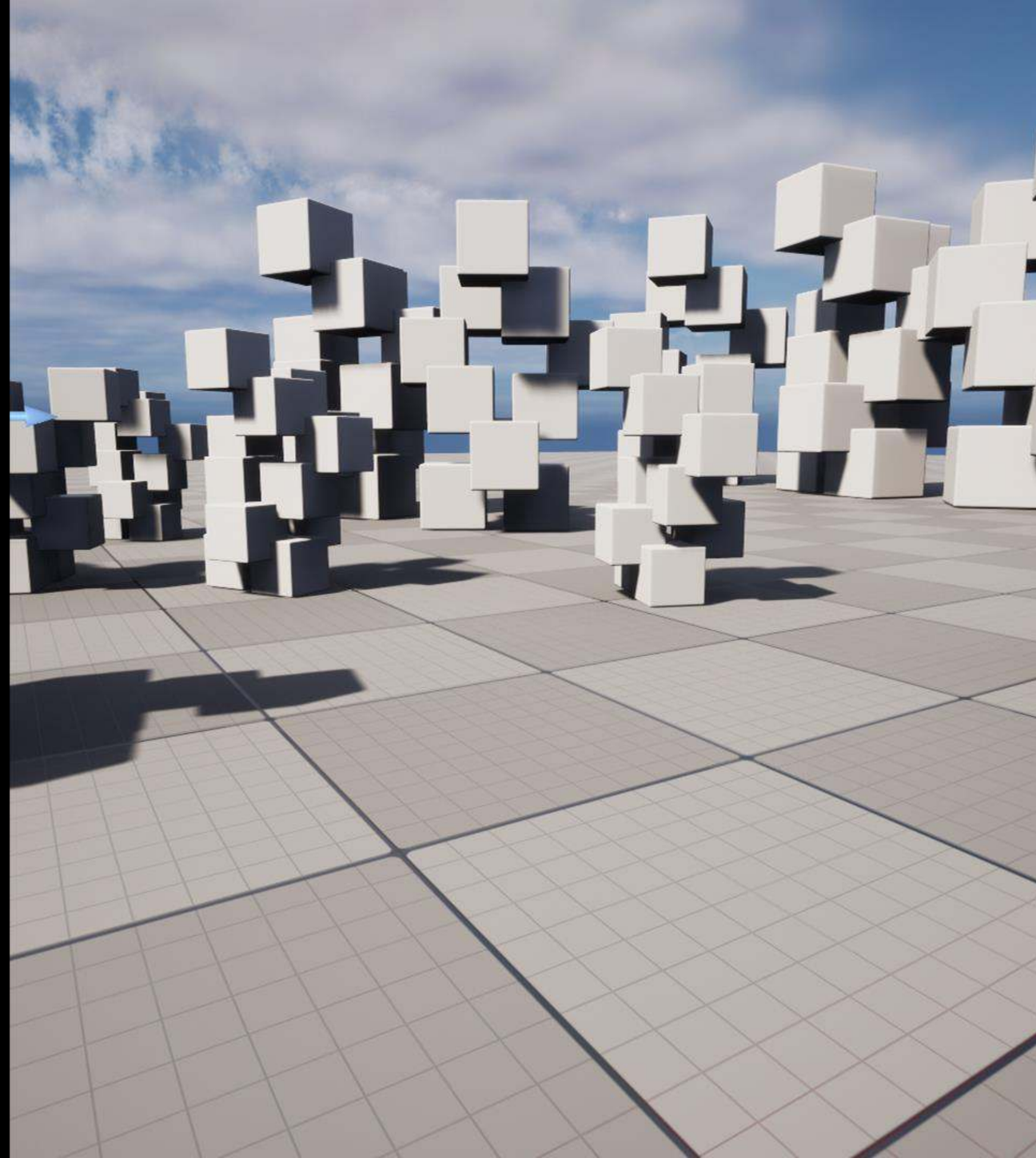
- Set r.Mobile.SupportGPUScene=1
  in your DefaultEngine.ini
- Most likely will be enabled by default in UE 5.5
- This optimization will be active for every platform that supports UBO (Android Vulkan/OpenGL, Nintendo Switch).
- iOS Metal does not use UBO optimization

Instanced StaticMesh
GPU Culling

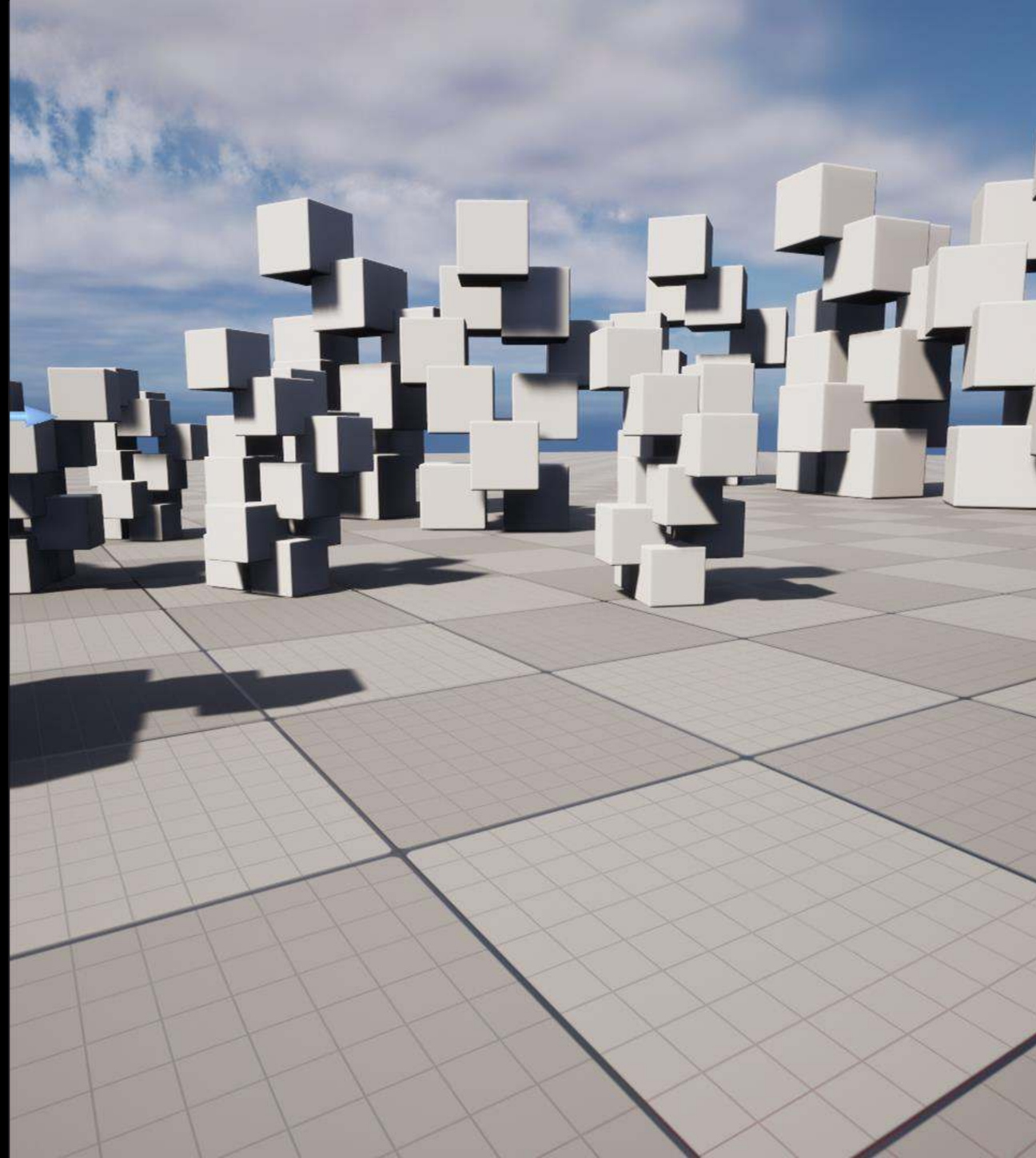


Instanced StaticMesh GPU Culling

- Requires GPU Scene
- Per-Instance culling is done while we generate InstanceData buffer in CS
- Only view frustum culling by default
- Optional per-instance occlusion culling
  - r.InstanceCulling.OcclusionCull=1
- Does not preserve instance order on mobile

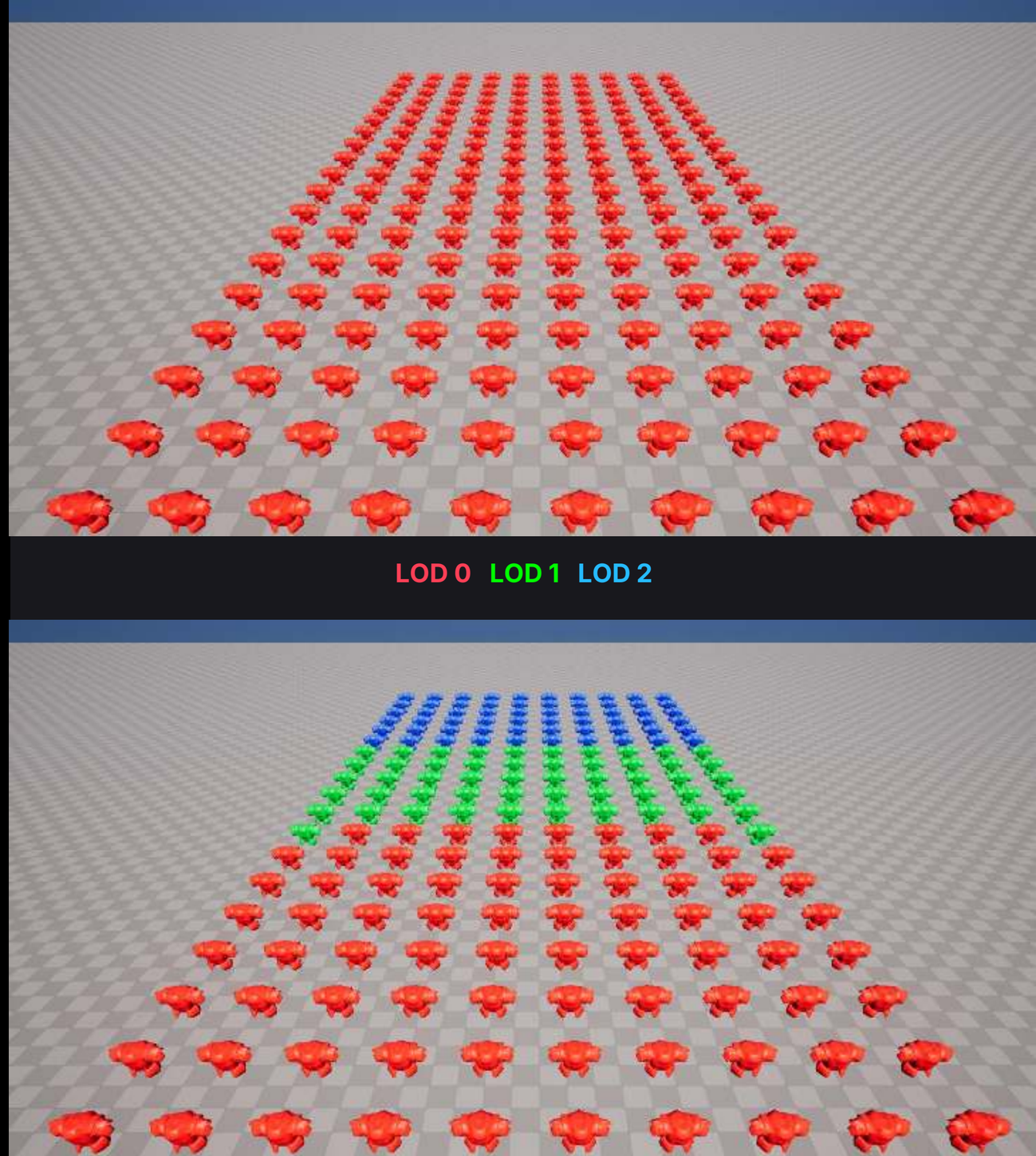


Instanced StaticMesh
GPU LOD



Instanced StaticMesh GPU LOD

- Requires GPU Scene
- LOD selection culling is done while we generate primitive buffer in CS
- On CPU we compute LOD range [LODMin, LODMax]. This determines how many draws ISM will have
- On GPU we compute and select LOD for each instance

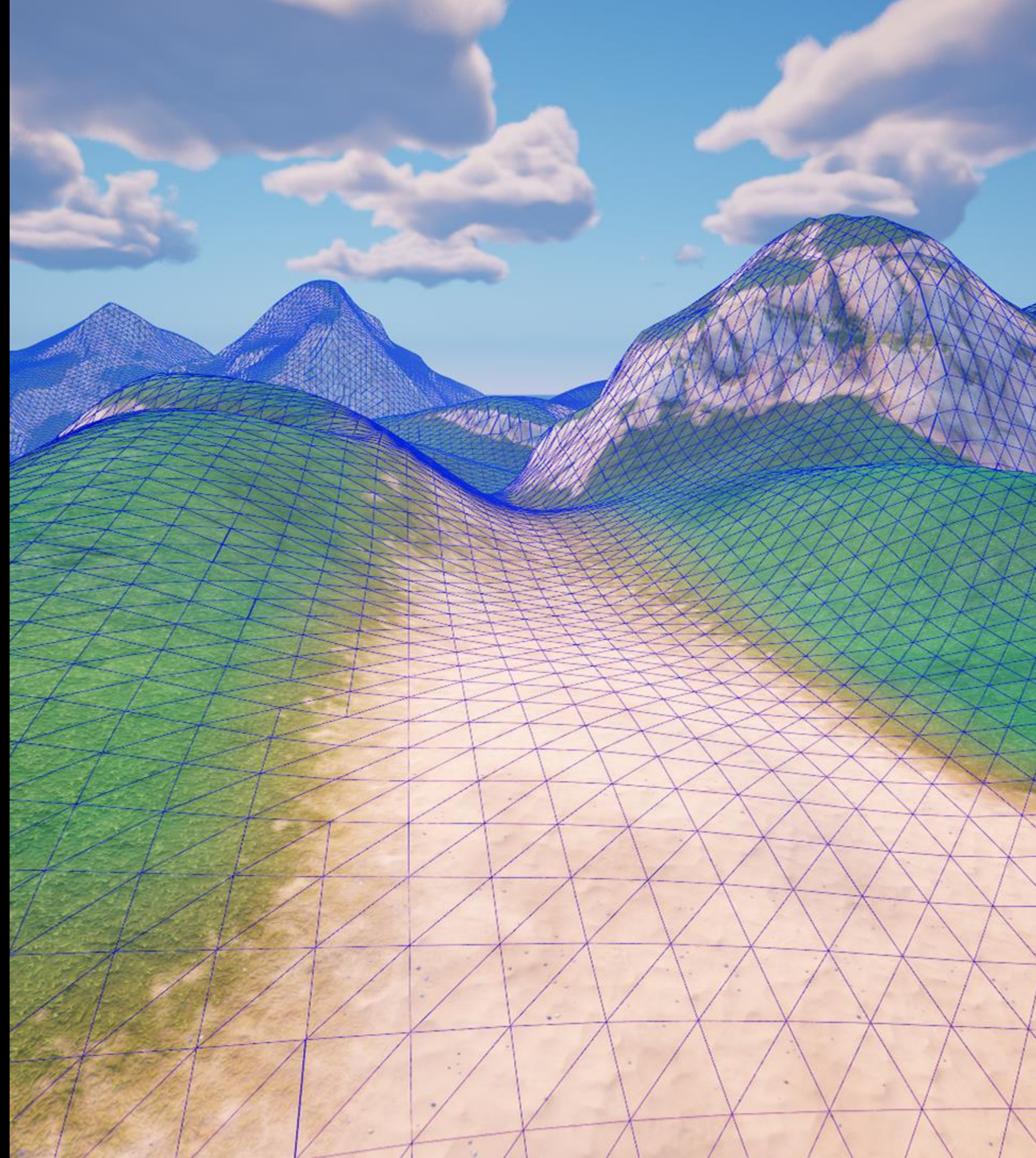


Landscape Optimization

Landscape
GPU Culling

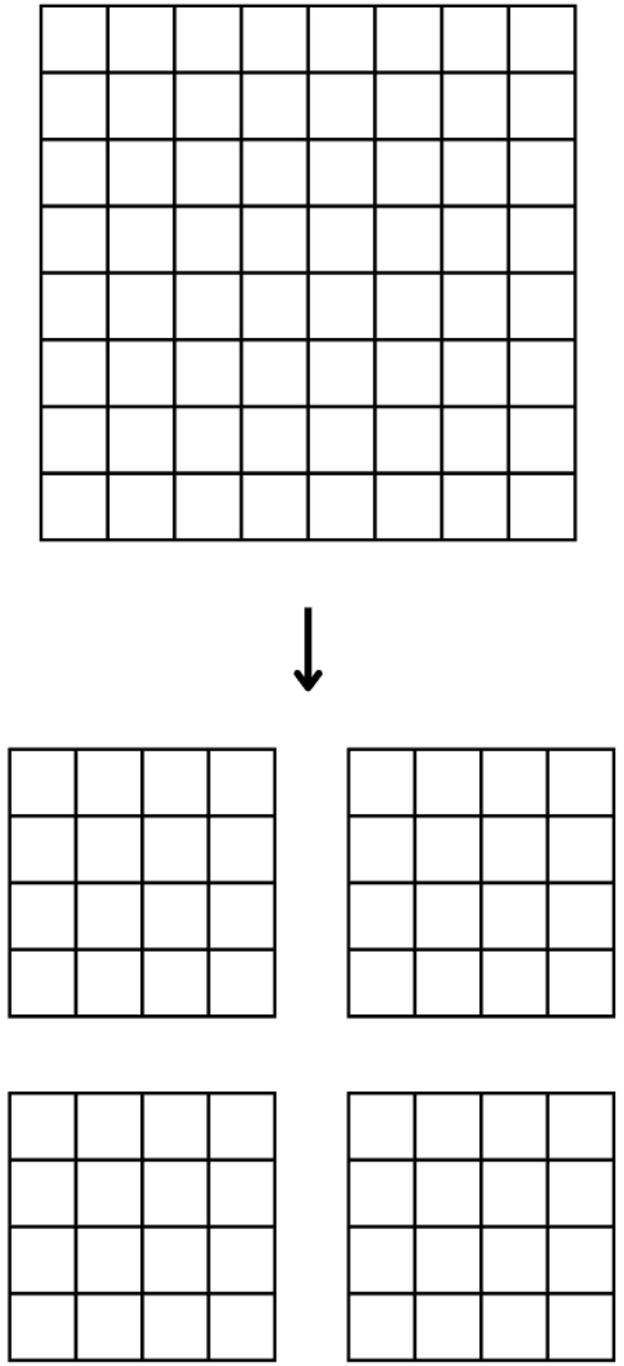
Landscape rendering

- Landscape is split to patches/components
- Each patch is [256x256 - 16x16] quads
- Typically LOD0 is 256x256 or 128x128
  - With a default 100 scale, 256x256 or 128x128 meters
- General culling works on a component level
- If any part of landscape patch in visible whole patch is rendered
- GPU has to process many landscape vertices even when they do not contribute to a visible pixels
- 256x256 patch has 131072 triangles



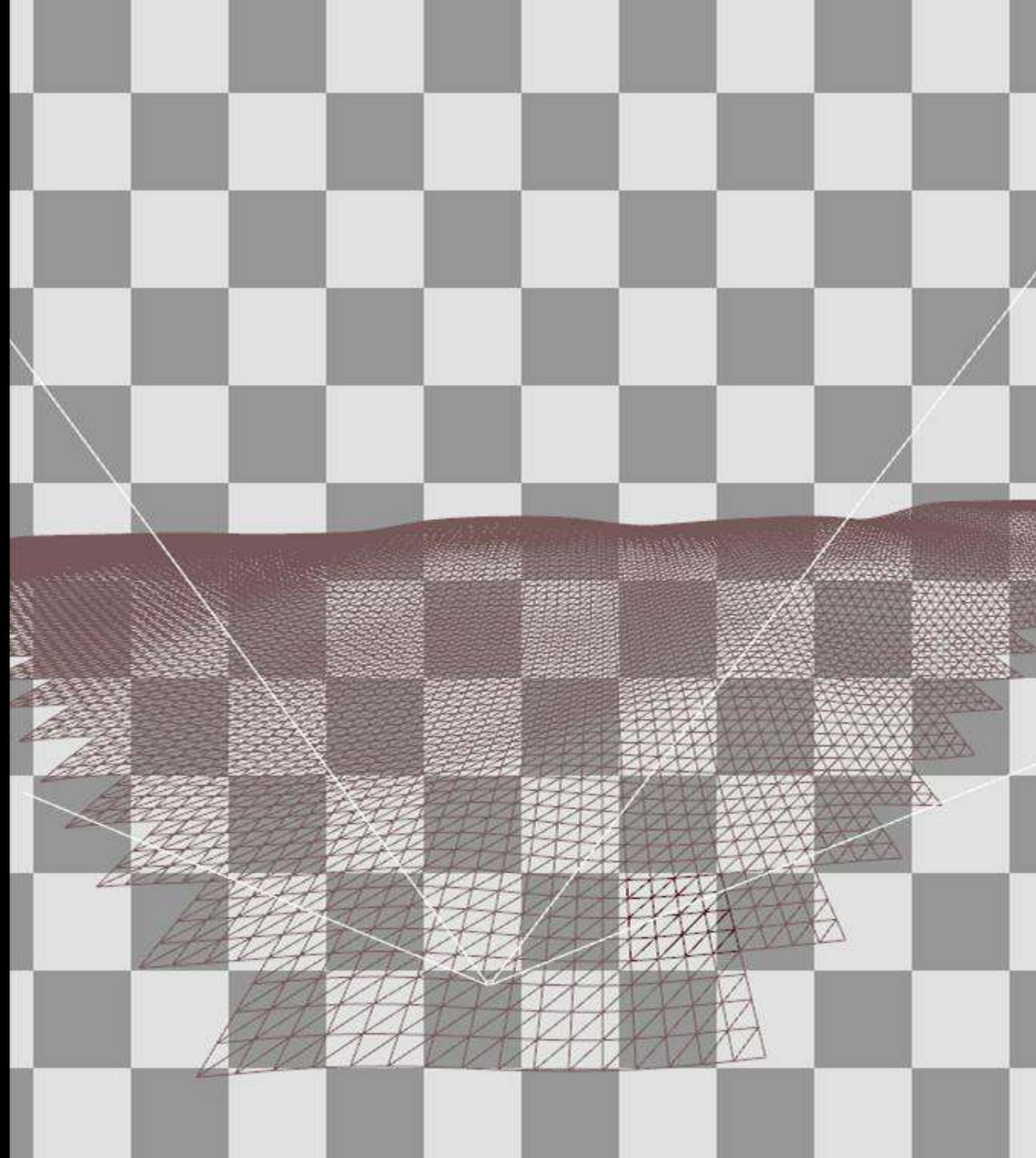
Landscape Tiled rendering

- Split landscape patch into tiles
- Tile is 4x4 quads
- 256x256 patch is 64x64 in tiles (4096 tiles)
- DrawInstanced(4096)
- Per-instance data is a position of the tile inside patch, only 4 bytes
- Tiled rendering is faster than regular because of better vertex re-use, and a smaller vertex buffer, better cache utilization



Landscape GPU culling

- For each view and landscape patch we run compute shader that does tile culling
- DrawIndirect(???)
- Right now only LOD0 is culled
- Use worst case for Height [MinZ , MaxZ] of whole patch, to avoid sampling Heightmap in a CS
- Very fast and efficient
- No support for shadow views yet
- Enabled on all platforms in UE 5.4
- Deactivated when any of these features are enabled: Landscape Nanite, Lumen or VSM



Landscape Geometry Cache

- Similar Idea to Skin Cache
- Avoid expensive logic in a landscape vertex shader
- CS outputs and caches a fully morphed landscape static mesh
- Only when camera moves enough - cache is re-computed
- Landscape static mesh is re-used between shadow, depth and base pass
- In our tests we didn’t see much difference in performance, so this feature was discarded

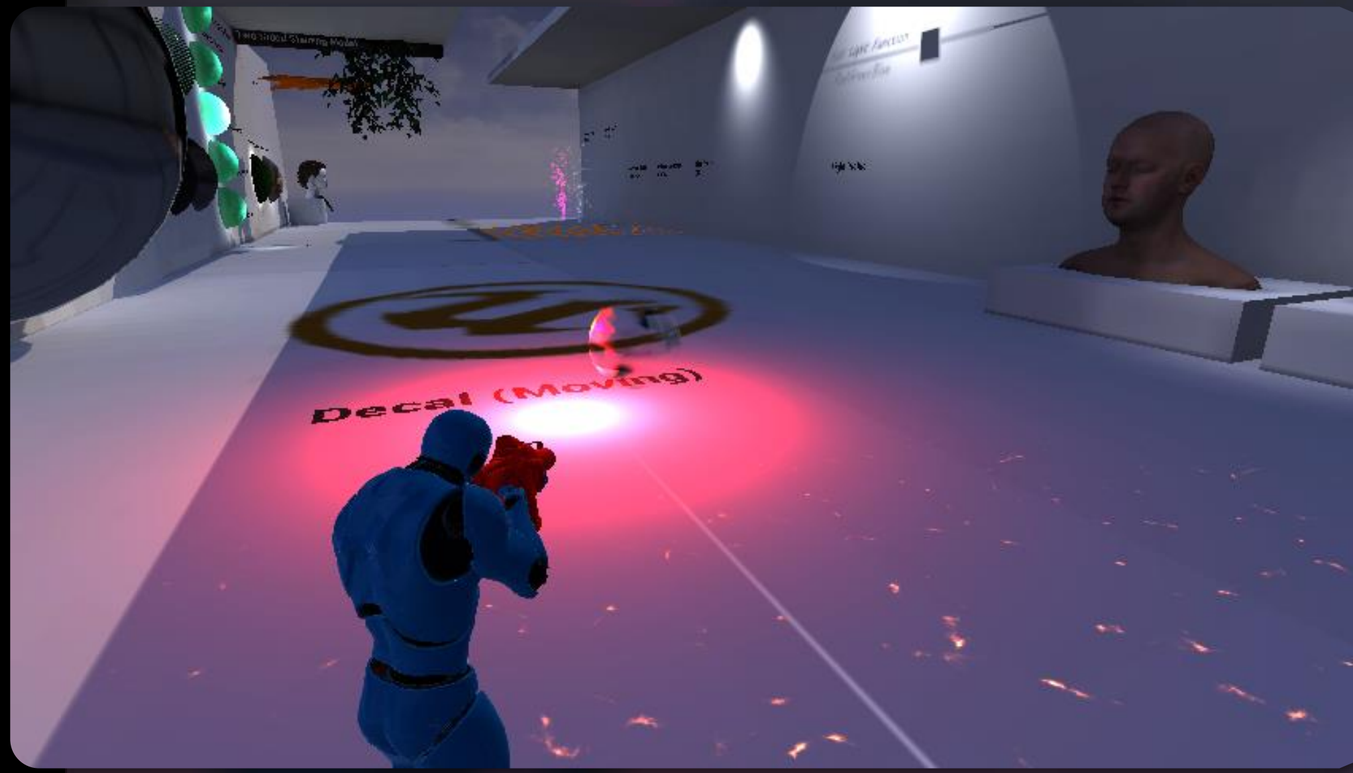
Forward Lighting Optimization

Forward Lighting Optimization

Survival game atmosphere requires lots of local lights in locations like caves.

Challenge:

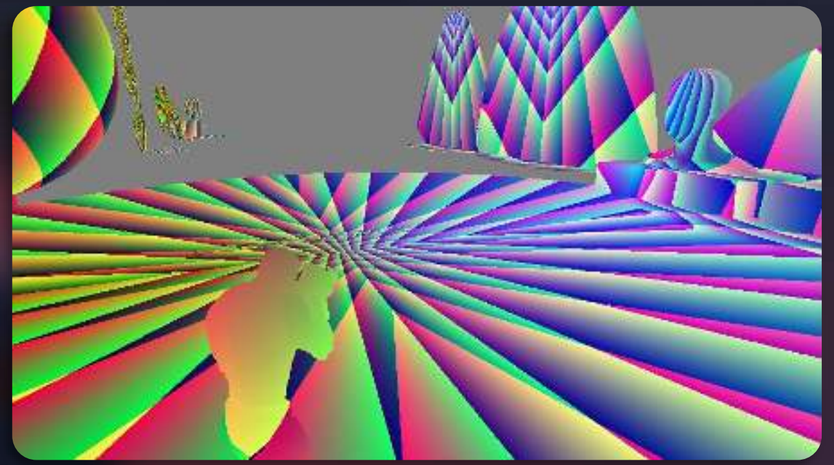
- Local lights has a poor performance in forward rendering
- Access to LightGrid is slow



Forward Lighting Optimization

Implementation:

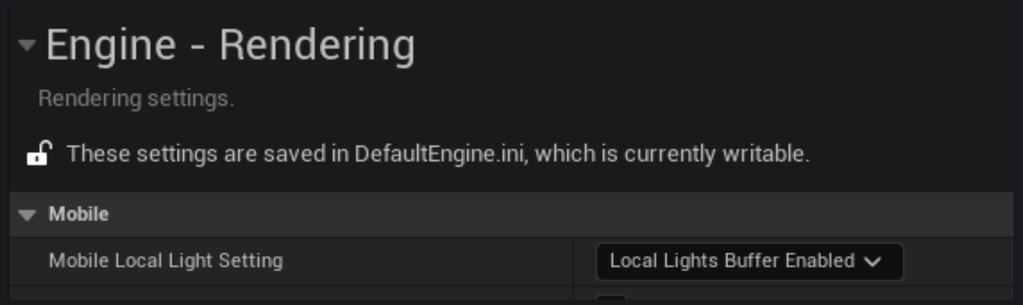
- Color merged in RG11B10
- Merged light direction is packed as RGB8
- Alpha is used for SpecularScale.
- Base pass shaders sample light buffer textures instead accessing LightGrid
- Requires full depth pre-pass
- If depth pre-pass is not active we merge lights in a base pass shader and compute BRDF once for a merged light data



Forward Lighting Optimization Conclusions

So, how do you enable it?

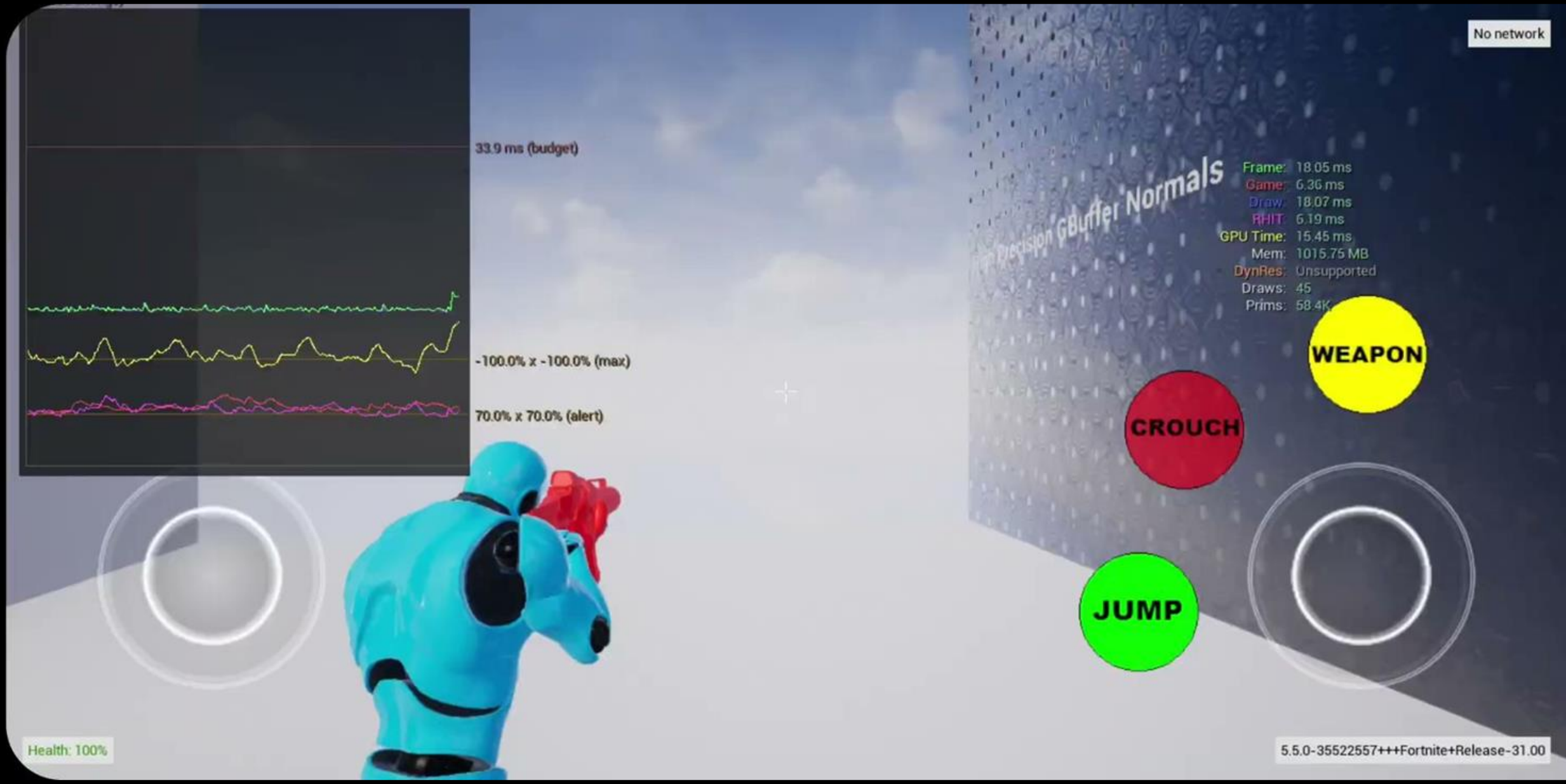
- Can enable it from the Editor, but this way it will be enabled for all platforms.
- If you want the merged lights optimization, you set r.Mobile.Forward.EnableLocalLights=2 in the <Platform>Engine.ini of the platform you want to enable this on.



Forward Lighting Optimization Conclusions

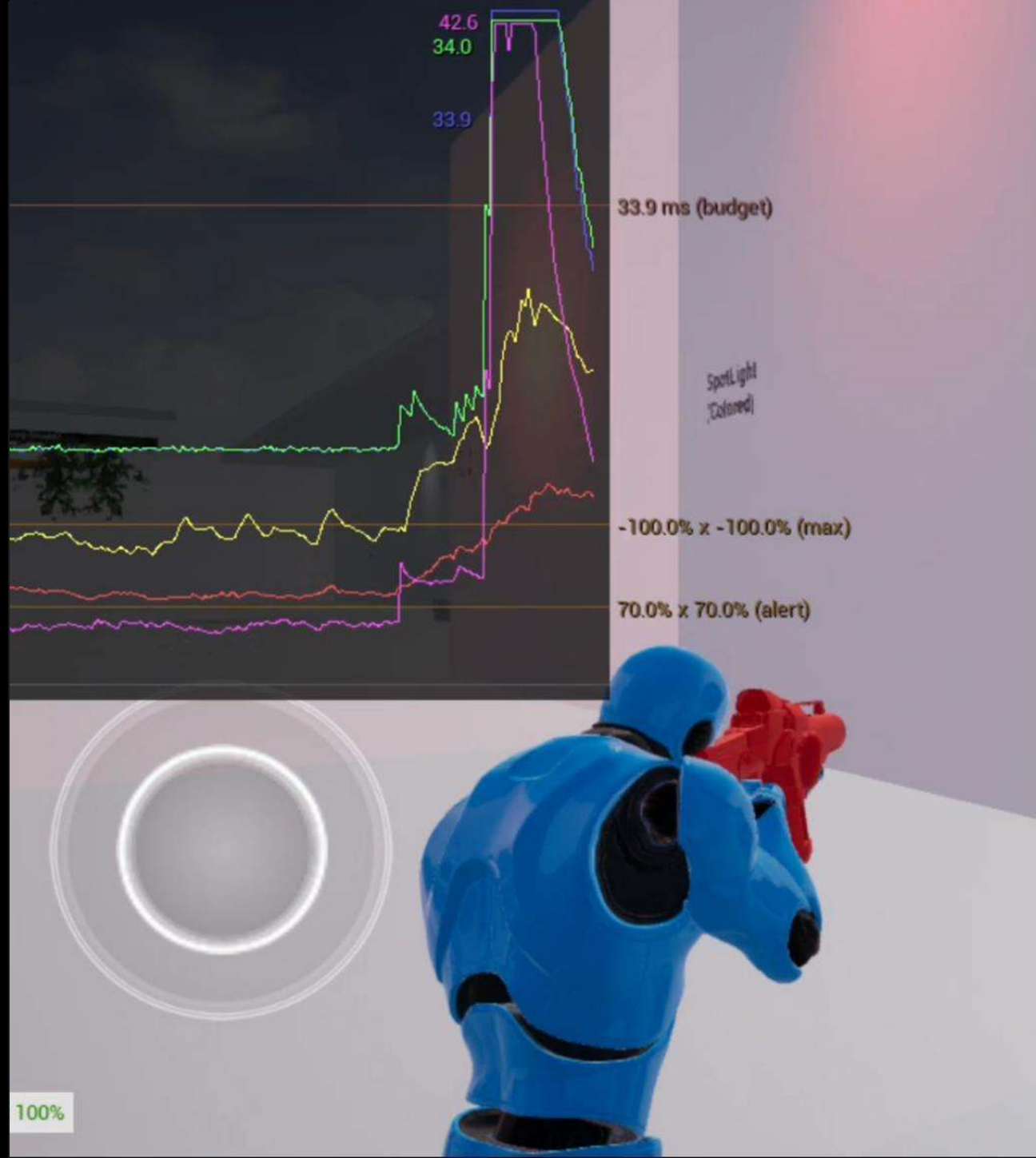
- In a latest Fortnite release we have optimized and enabled mobile deferred shading on Nintendo Switch
  - Active on Android Vulkan and iOS
- We might remove light buffer in the future
- Light mergring will remain as a performance vs quality tradeoff


Runtime
PSO pre-caching


33.9 ms (budget)
-100.0% x -100.0% (max)
70.0% x 70.0% (alert)
No network
Frame: 18.05 ms
Game: 6.36 ms
Draw: 18.07 ms
RHIT: 6.19 ms
GPU Time: 15.45 ms
Mem: 1015.75 MB
DynRes: Unsupported
Draws: 45
Prims: 58.4K
WEAPON
CROUCH
JUMP
Health: 100%
5.5.0-35522557+++Fortnite+Release-31.00

Shader compilation hitch

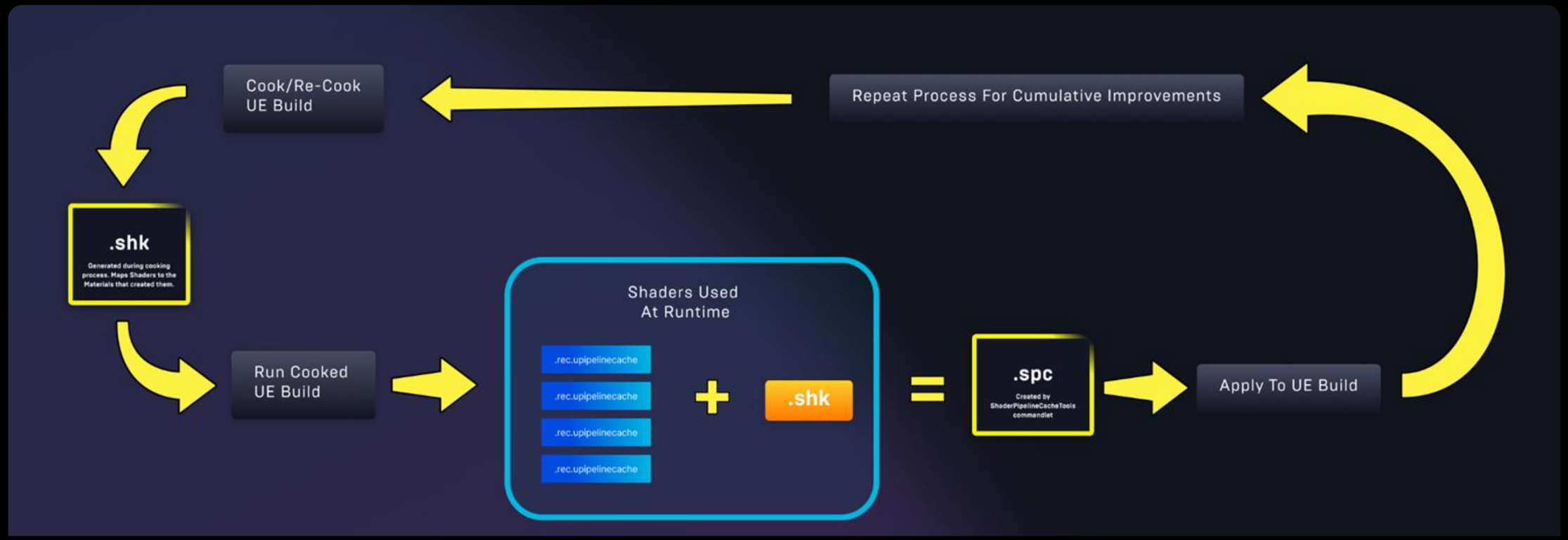
- Engine has to compile shader (create PSO) when it’s used for rendering for a first time
- Unreal Engine has a rich and flexible material system. That allows you to create complex materials with a complex shaders
- Forward shading has to use several shader permutations for each material for a better rendering performance
- In Fortnite we have thousands of materials and hundreds thousands of shaders



PSO Gathering

Challenges with current system:

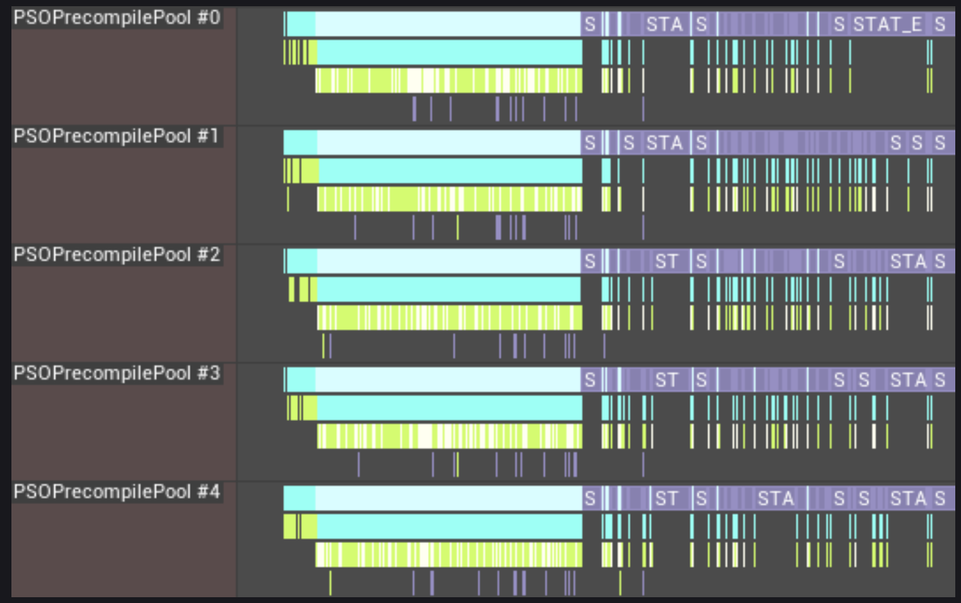
- PSO gathering can be a very complicated, error-prone process
- Most of the time, it requires a replay system in place to be efficient
- Long load times (optimizing content) to compile all the gathered PSOs

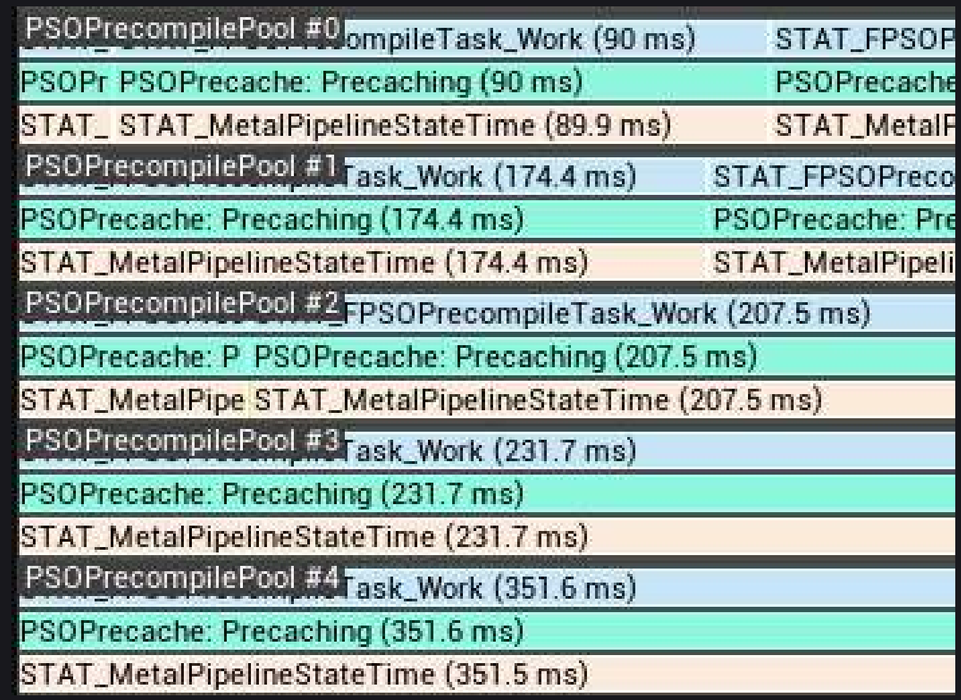


PSO pre-caching

- As early as possible - loaded assets get their PSOs queued for compilation
- PSOs get compiled on a task threads
- By the time PSO is needed most likely it will be already compiled
- Fallback strategy if PSO is not ready

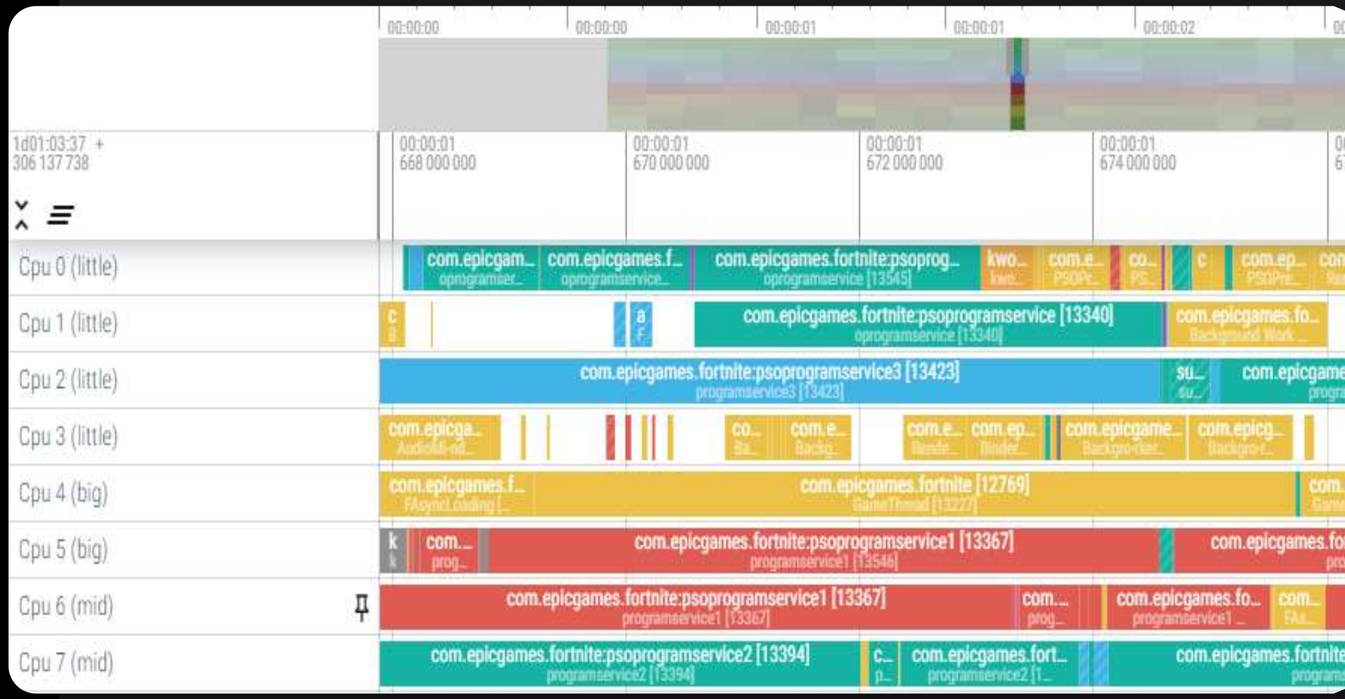
  r.PSOPrecache.ProxyCreationDelayStrategy
- Had to do some guesses to collect a correct rendering state for a PSO





Android PSO pre-caching

- OpenGL does not really support multi-threaded shader compilation
- Vulkan does in theory. In practice it depends on a driver.
  - Global mutexes and occasional crashes
- We had to implement shader compiler workers that run as separate processes. And use inter-process communication with a game
- Not ideal, as each shader compiler worker has its own PSO cache
- PSO cache gets saved to disk on app backgrounding event
- Next game sessions re-use on disk cache



Runtime PSO pre-caching

Advantages:

- Works automatically
- No need to have a gathering PSO system in place with manual steps
- Faster load times: no need to compile all the PSOs that could ever be encountered
- More configurable system: you can select what materials you want to precache
- Look for r.PSOPrecache.*

Disadvantages:

- PSO pre-caching has a runtime cost, performance and memory
- Delayed catching of bugs, probably in live
- Hitches if materials are not pre-cached or we set an incorrect rendering state

Runtime PSO Pre-caching Conclusions

So, how do you enable it?

- Set r.PSOPrecaching to 1 in the <Platform>Engine.ini of the platform you want to enable this on. It is enabled by default.
- Available on Android Vulkan/OpenGL in 5.4
- iOS Metal in 5.5
- Lots of fixes for pre-computed lighting and optimizations in 5.5
- Enabled by default on all mobile in 5.5

Thank you!

UE Mobile team