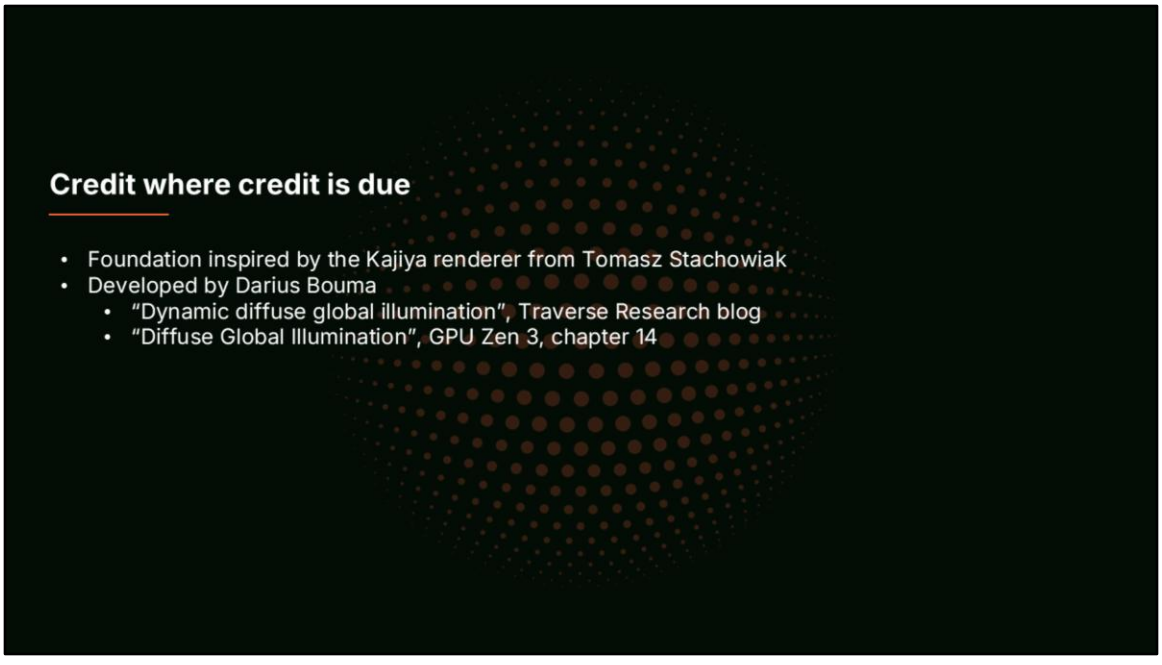
Hey everyone, I will be talking about our real-time dynamic global illumination system on mobile. Sooo, lets dive right in.

First, I will quickly cover Evolve, our benchmarking software. Then I will go through our mobile hybrid renderer and the main components. Then the clipmap irradiance cache we use. Followed by our ReSTIR GI implementation tailored to mobile devices. Then I will go over some performance numbers and figures. And lastly, I will quickly go over one of the next directions we are exploring for dynamic global illumination.



First of, credit where credit is due. A large part of our global illumination is inspired by the Kajiya renderer from Tomasz Stachowiak. If you know how that works, you will see many familiar systems during this talk.

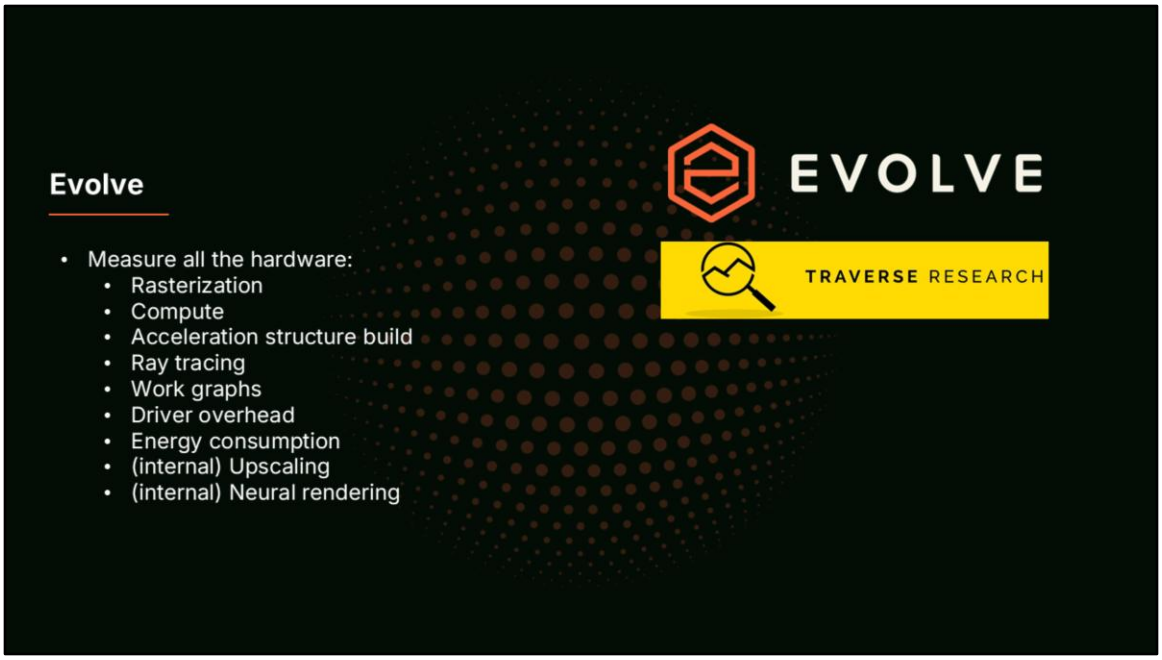
Then, the dynamic diffuse global illumination system is largely developed by Darius Bouma, so I will mostly be presenting his work. In the past Darius has written two articles about the diffuse GI system, one in the form of a blog post, and the other was published in GPU Zen 3.

This talk will combine knowledge from the articles, and give a deep dive into the final version of our GI system that we shipped in Evolve.

Evolve is a multi-platform benchmark targeting Windows, MacOS, Android and Linux. Our render backend currently supports vulkan, directx12 and metal. And we have different quality presets along with separate leaderboards for desktop and mobile.

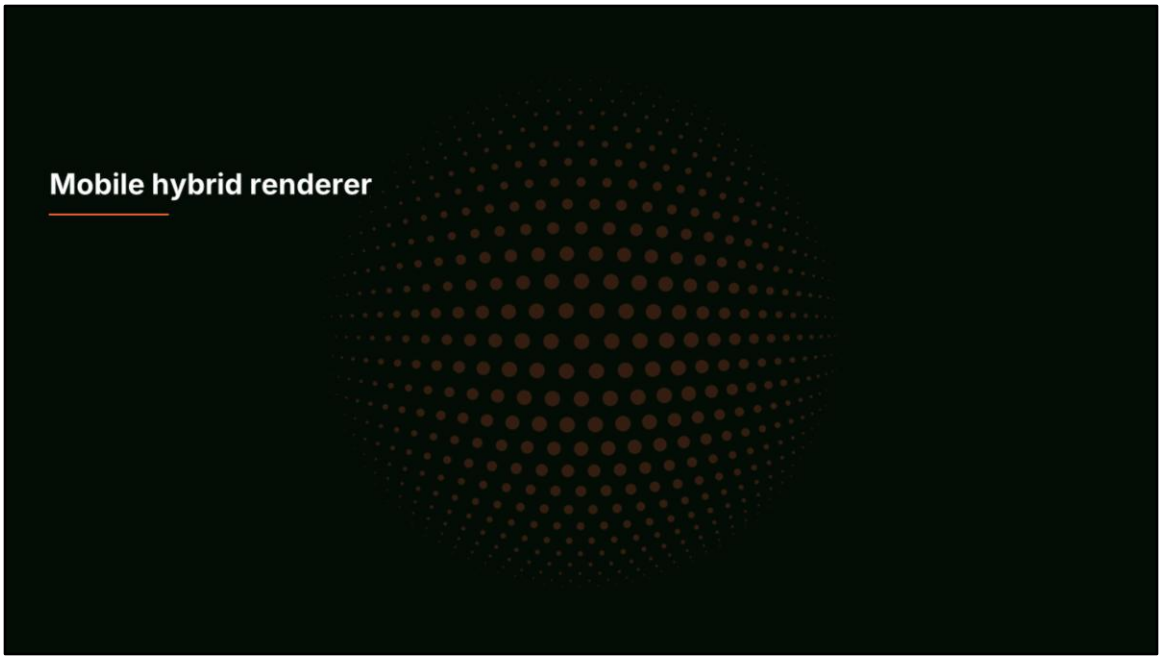
The benchmark is developed by Traverse Research, and we are targeting the latest advancements in hardware and software.

For example, our render graph has had a fully bindless design since 2019. We have multiple ray tracing systems all implemented using both inline ray tracing, through ray query, and pipeline ray tracing. We have internal builds with multiple different neural rendering techniques, with which we can benchmark new features like cooperative vectors. And we will soon be releasing a benchmark which measures both upscaling performance and quality across multiple vendor specific upscalers.

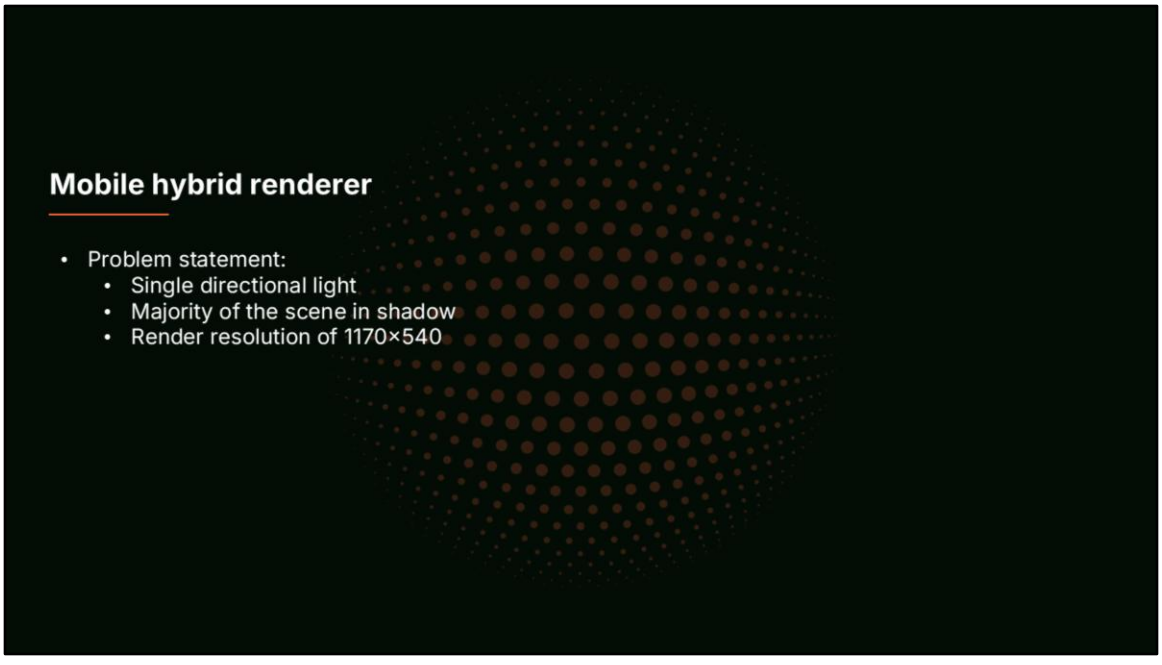


Since we are a benchmark, we are interested in measuring things, and unlike most other benchmarks we split our scoring into the following categories. Rasterization, compute, acceleration structure build times, ray tracing, work graphs, driver overhead and energy consumption. And we have internal builds for upscaling quality and performance, and neural rendering.

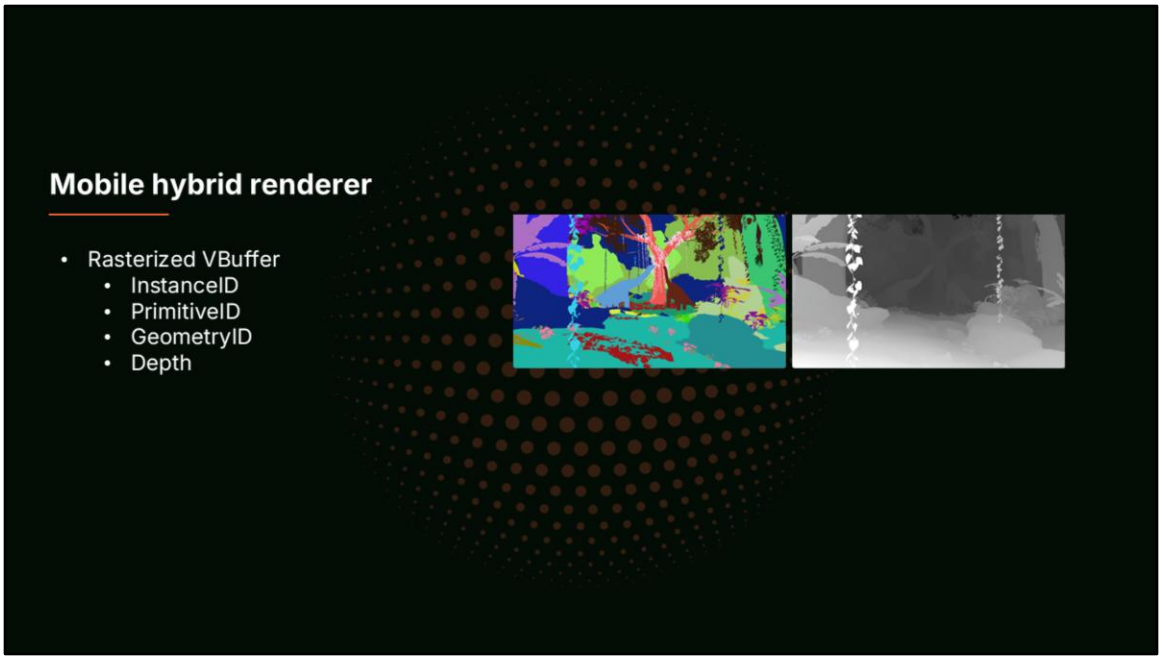
Lets have a look at the trailer first, to get an idea for the scene that we use. This trailer is captured on desktop, we will later show captures of the mobile renderer.



Let’s start by doing a quick rundown of our rendering pipeline



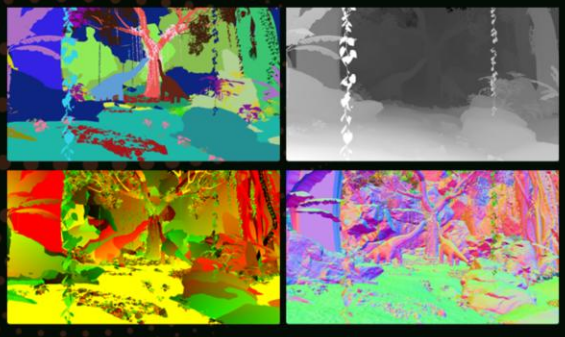
First, the problem statement we are dealing with. For Evolve we use a scene with a single directional light, but with a significant amount of foliage, which results in large parts of the scene being occluded from direct light, which makes high quality global illumination very important. On mobile specifically we target a rendering resolution of 1170x540.



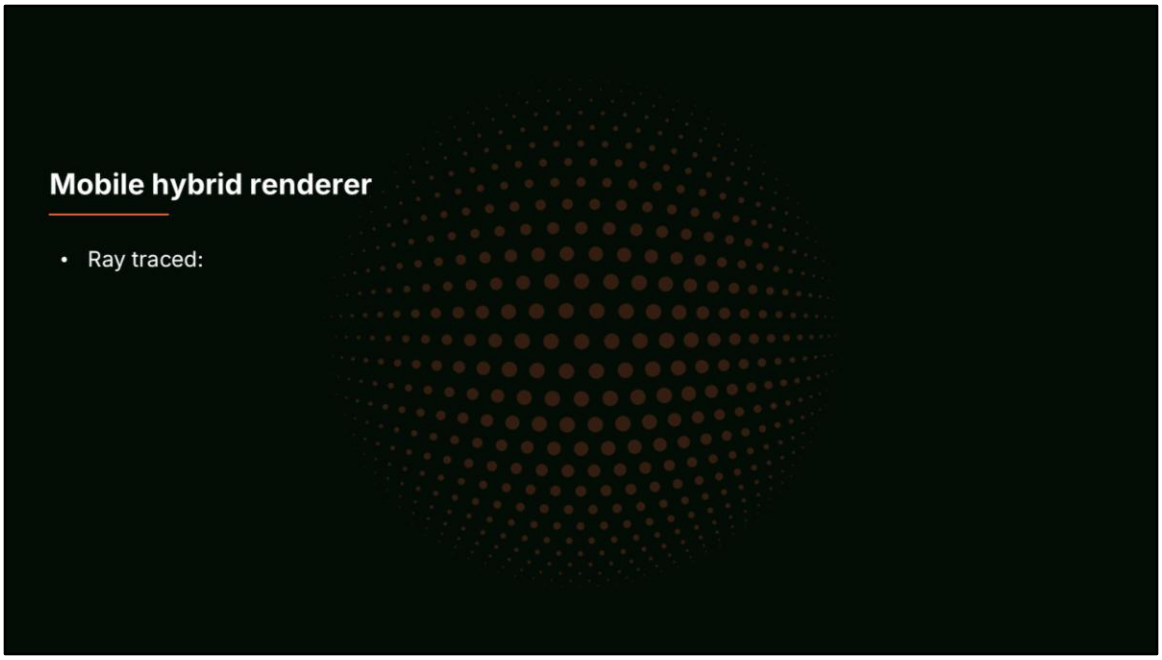
Evolve employs deferred rendering. We first rasterize the scene into a visibility buffer. This writes out InstanceID, PrimitiveID, GeometryID and depth.

Mobile hybrid renderer

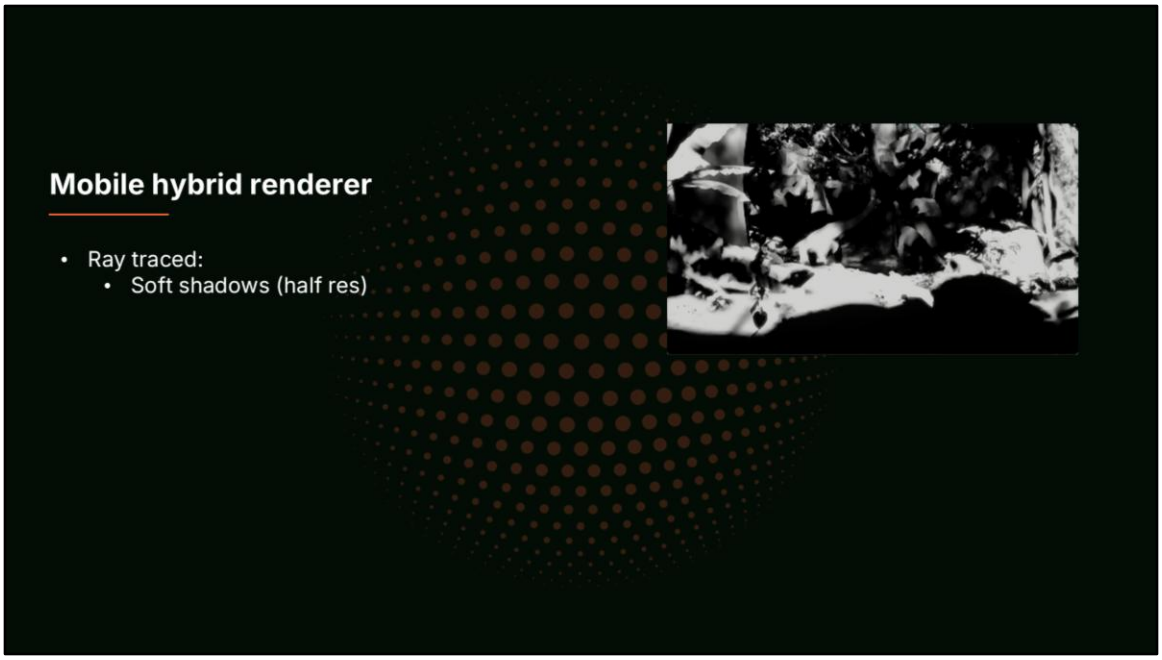
- Rasterized VBuffer
  - InstanceID
  - PrimitiveID
  - GeometryID
  - Depth
- Generate GBuffer
  - UV's
  - Normals
  - Velocity buffer
  - Stencil data (sss, skin, backfacing)



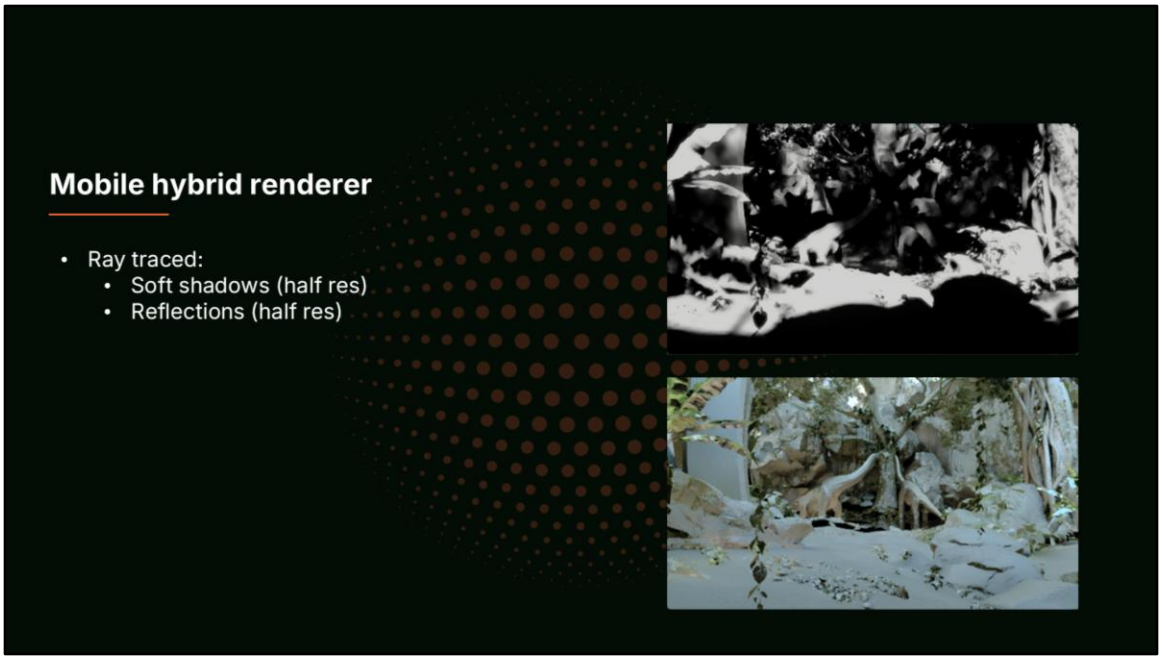
Then, from this we generate a g buffer, which writes the UV's, normals, velocity buffer and stencil data.



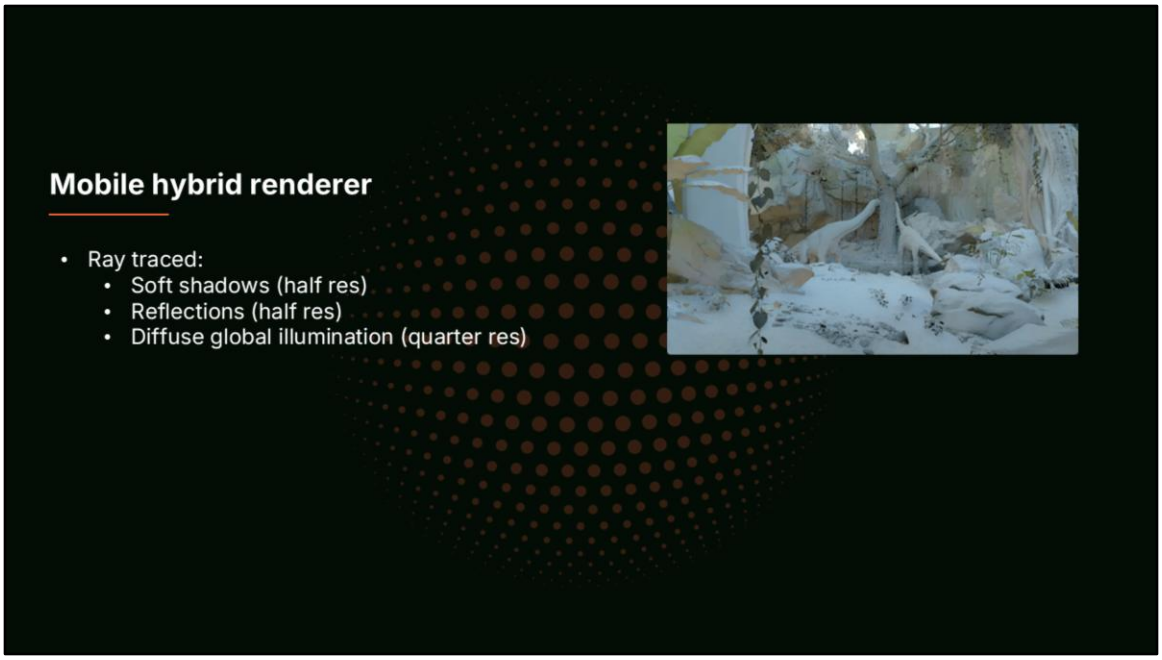
After our Gbuffer layout, we move onto the lighting. At the moment all of our scene lighting in Evolve comes from ray tracing systems.



Soft shadows are traced in half-res on mobile, which comes down to a resolution of 585 by 270.



Reflections are also traced at half res.

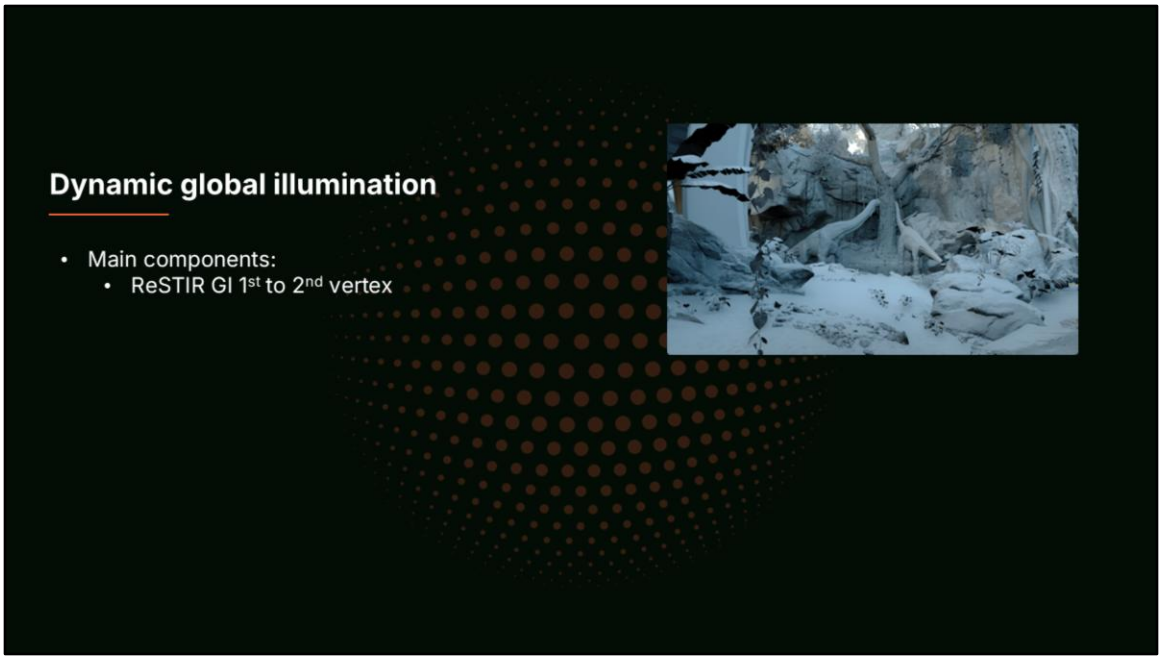


GI is instead traced at quarter res, as it’s a lower frequency signal and we can get away with less rays. This comes down to a resolution of 292 by 135.

Mobile hybrid renderer

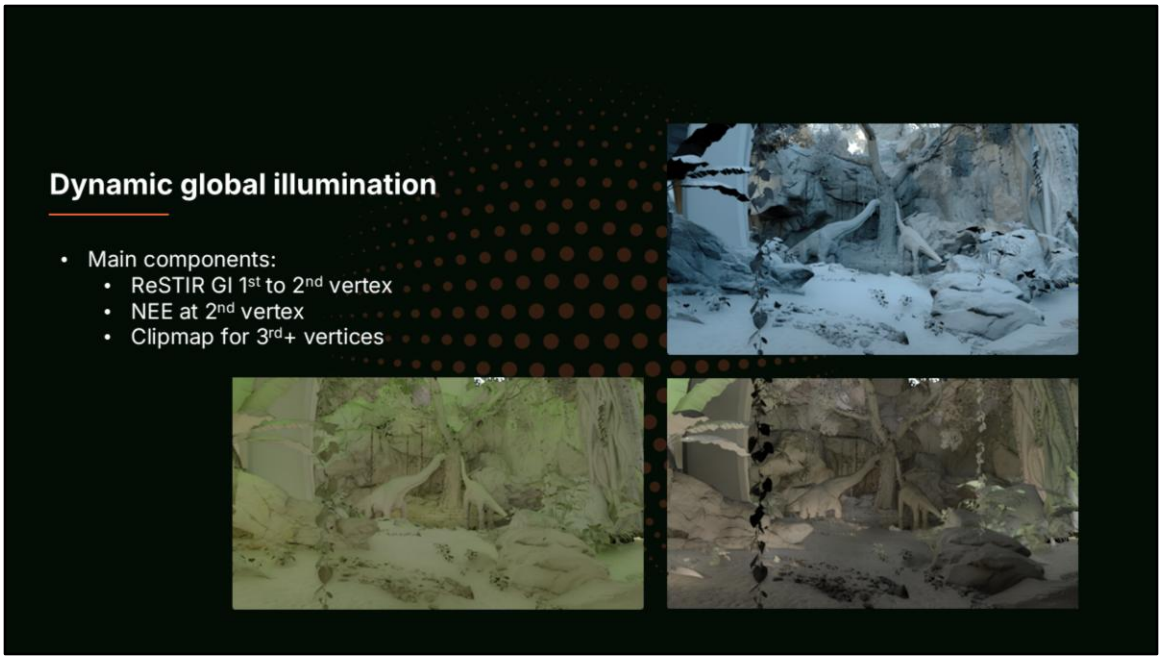
- Ray traced:
  - Soft shadows (half res)
  - Reflections (half res)
  - Diffuse global illumination (quarter res)
- Composite

These are then composited together at the end of the frame, along with other effects like sky and water.

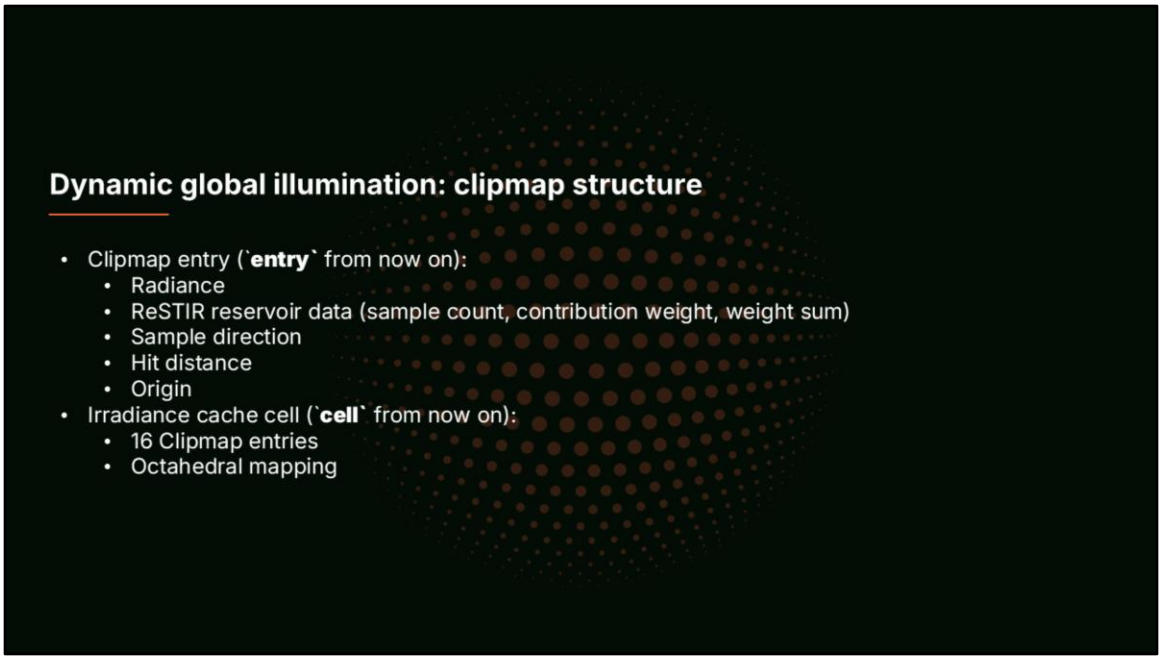


Now, the three main components of the diffuse GI system are (1) ReSTIR GI, for keeping track of high-quality 1st to 2nd vertex traces. In the image on the right we show the irradiance throughout the scene if we exclusively sample radiance from this single ReSTIR GI trace, specifically, we only show sky contribution.

To get a more complete approximation of the indirect illumination we perform next event estimation traces from the 2nd vertex. This ensures that the ReSTIR GI system receives accurate lighting contribution from the most influential path vertex, namely the 2nd.

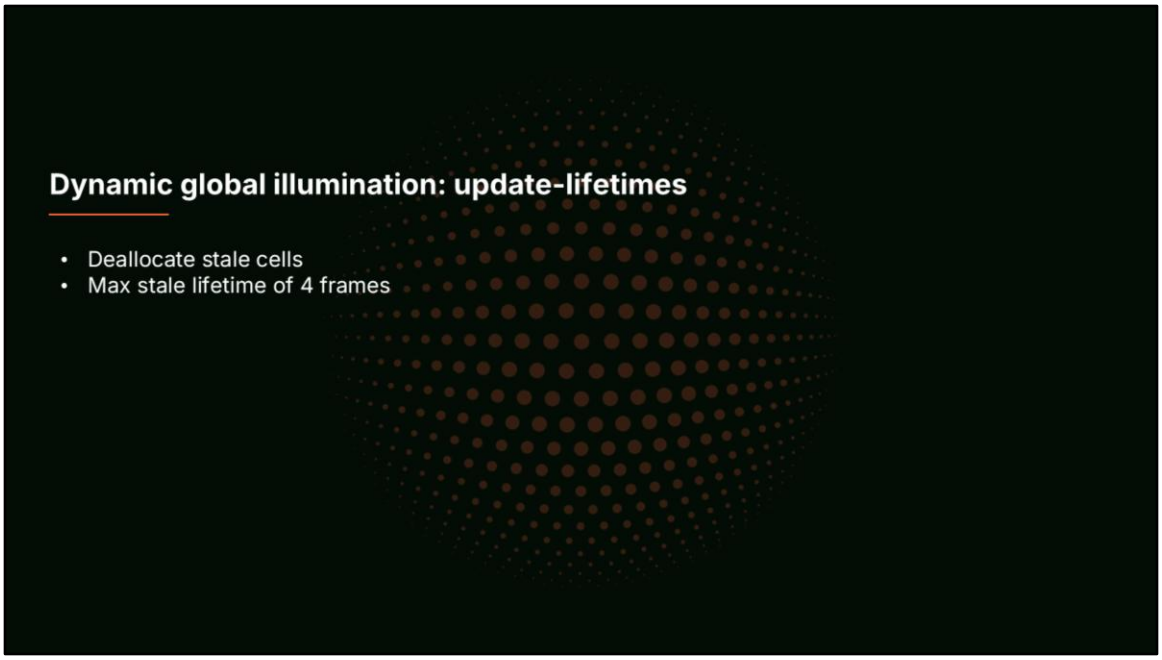


And lastly, we use a clipmap containing a rough approximation of the multiple bounce lighting. We sample this at the 2nd vertex, but as this cache does not include direct illumination, it is responsible for adding radiance that originates from the 3rd vertex or beyond. The images you see here have adjusted exposure, to highlight the effect of each of these components.

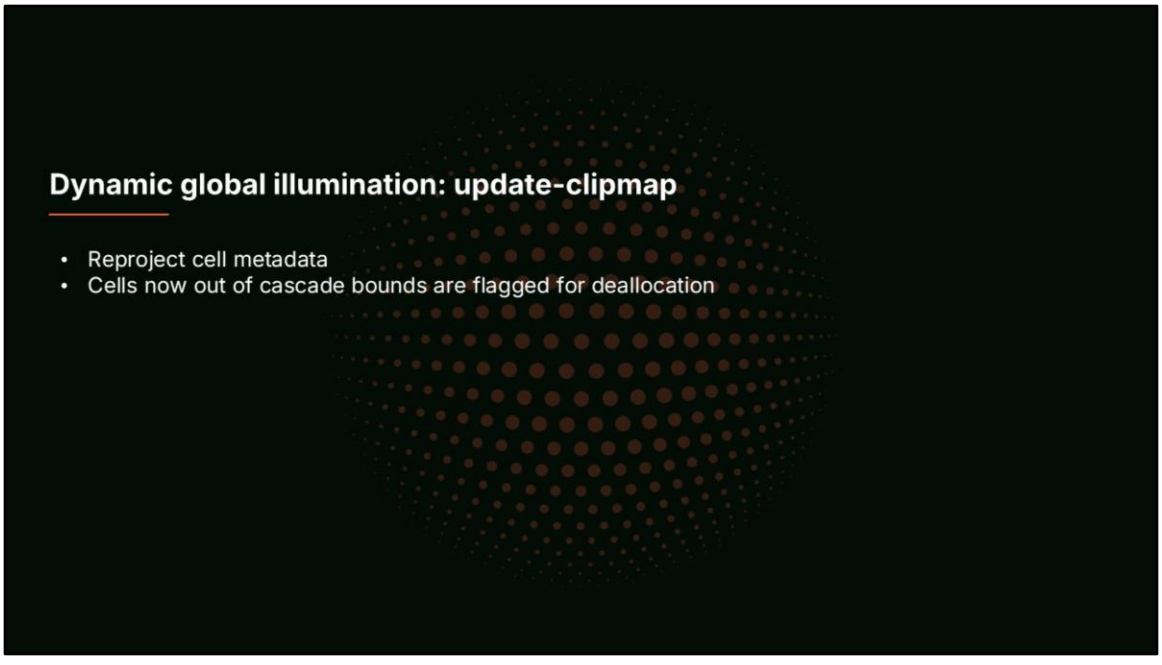


Let us first quickly cover some terminology. The clipmap consists of two main structures. First we have the clipmap entry. This stores radiance, reservoir data, a sample direction, hit distance and origin. Second, we have irradiance cache cells, each of these cells contains 16 clipmap entries mapping to different directions. At the end of our clipmap update, each cell will be represented by one set of L1 spherical harmonics which describe the irradiance at the cell location.

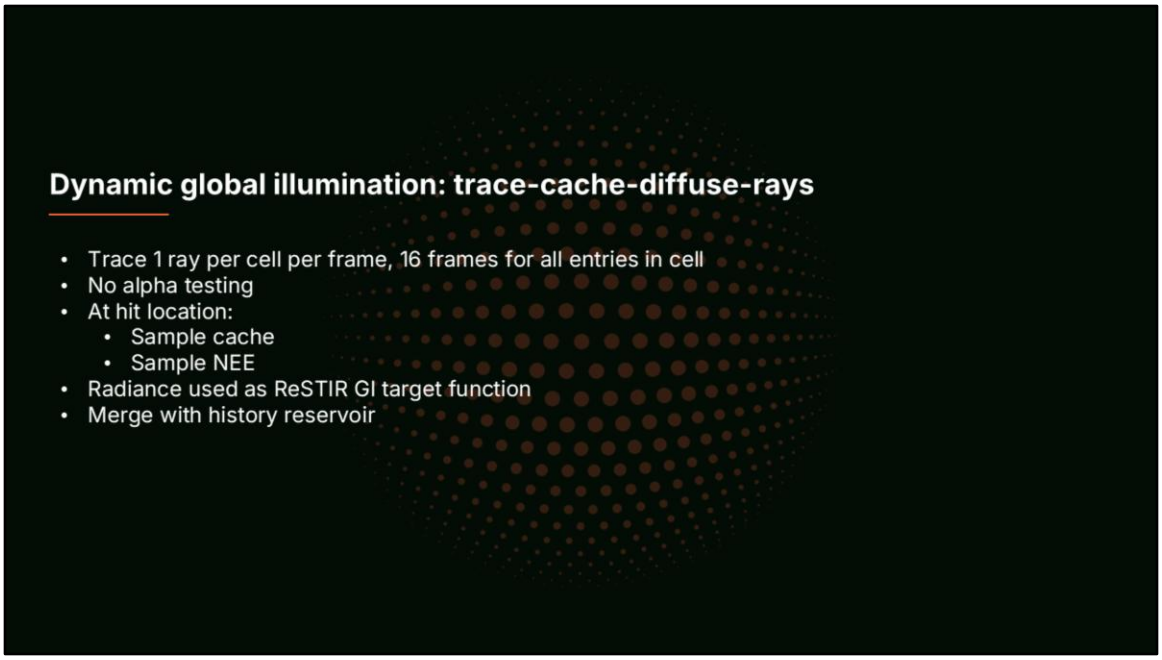
Now lets get started with the different passes related to the clipmap cache. We will walk through them in order of execution during the frame.



We start with updating the lifetimes of our cells. This is a fairly simple pass. If we have not sampled a cell in the previous frame, we start incrementing its stale lifetime, once a cell has been stale for 4 frames, we deallocate it. This keeps the amount of alive cells manageable. We have a hard cap of 65535 cells, so we must recycle them if we want to stay well below that cap.



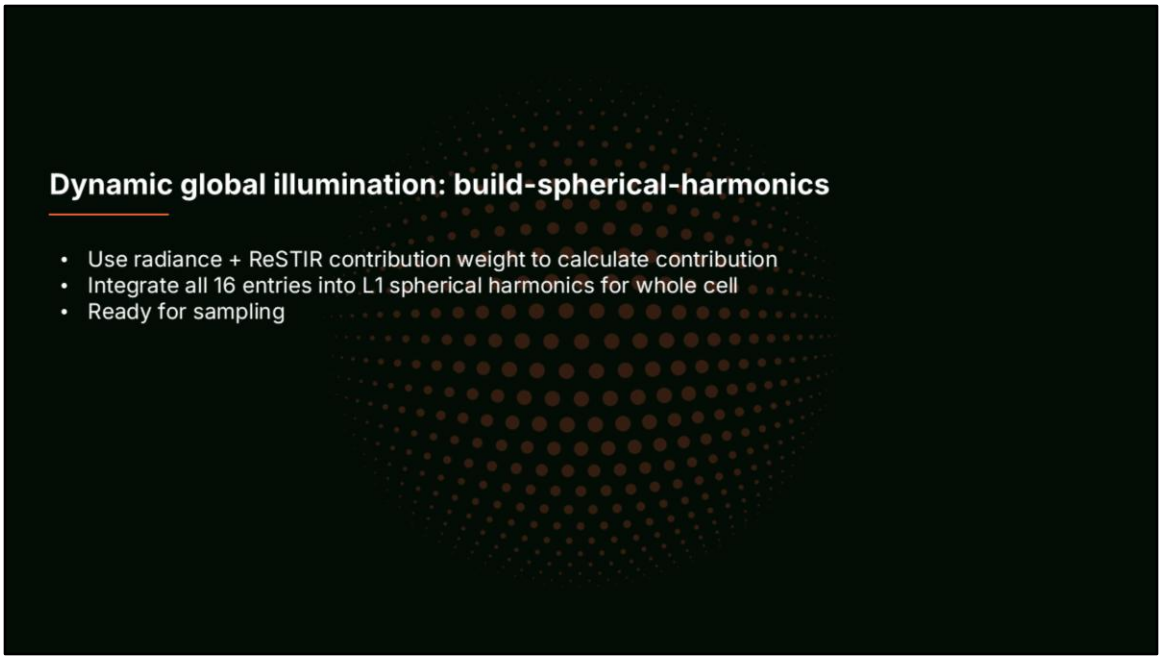
Next, update clipmap. This pass reprojects cell metadata in a scrolling fashion. Any cells that are out of the cascade bounds will be flagged for deallocation. This can happen for example when the camera moves further away and we switch to a coarser cascade.



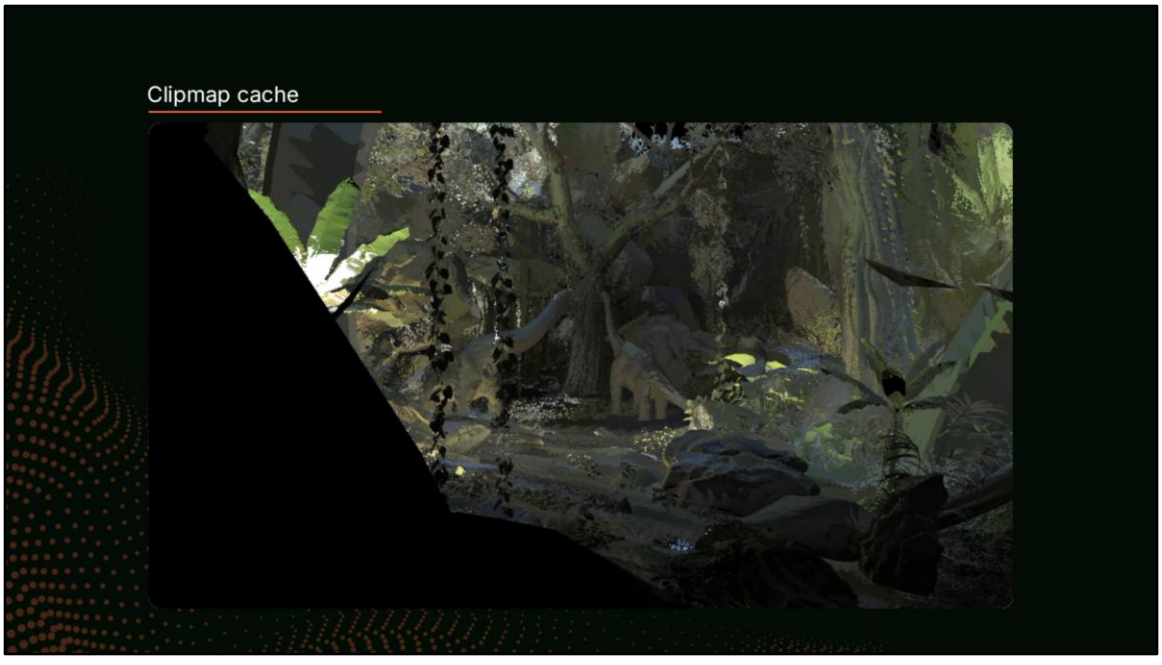
Next we get the most expensive clipmap pass. Trace diffuse rays. This pass updates one entry per cell by tracing one ray. This means that after 16 frames we will have updated all entries within a cell. For this trace we skip alpha testing as the clipmap cache is very coarse anyway. Which could result in some darkening, but we can live with that for the perf we get back.

After tracing, we have to calculate the radiance from the hit location. We do this by combining a recursive cache sample with a next event estimation sample. This way of recursively sampling the cache imitates infinite bounce light.

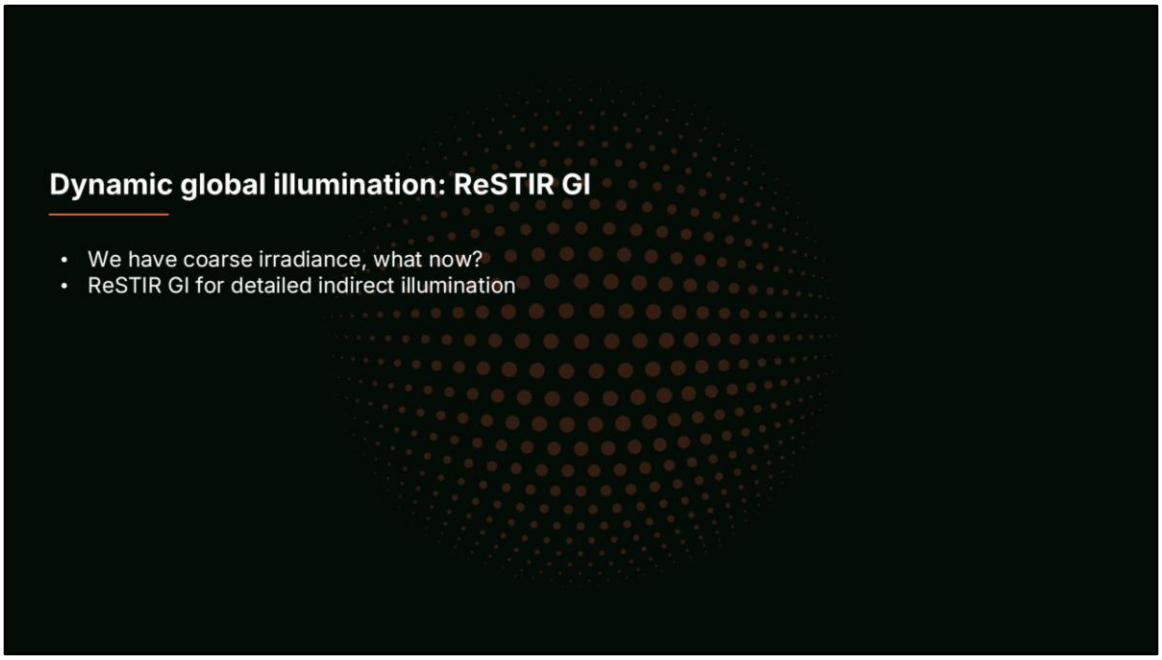
Once we have the radiance, we can use this as our ReSTIR GI target function. Then at the end of this pass, we merge our new candidate reservoir with our history reservoir. This way each entry keeps track of important contributors within its designated direction.



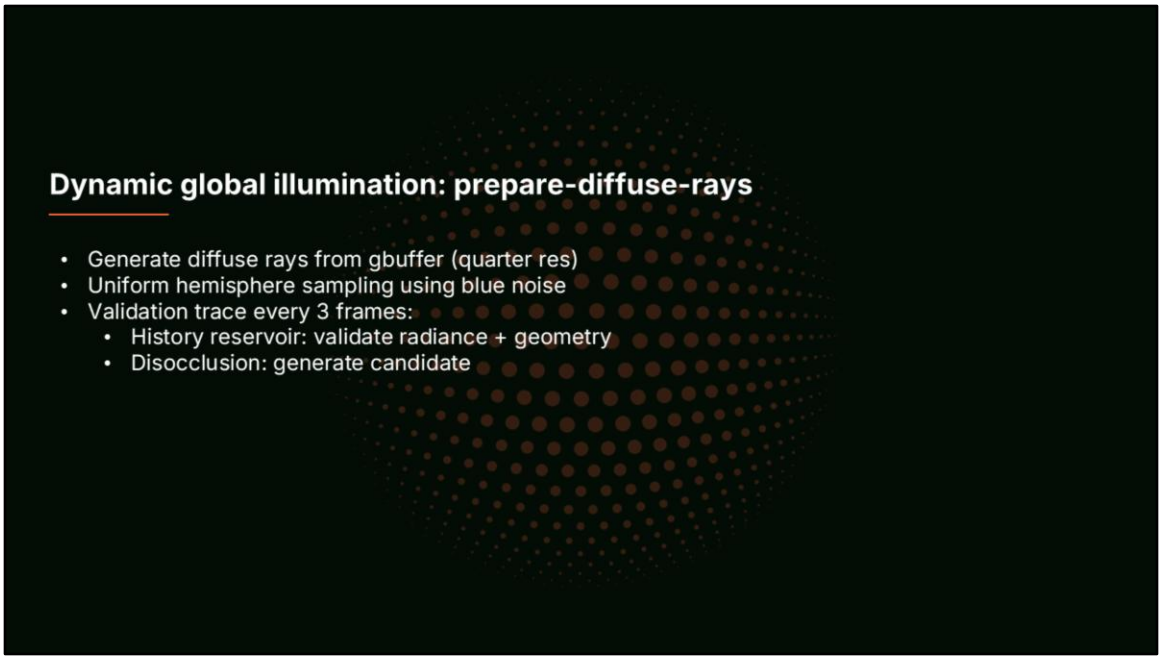
Next we construct L1 spherical harmonics from each cell. This calculates the contribution of all 16 entries using the radiance and the reservoir contribution weight, and integrates them into one set of L1 spherical harmonics. After this pass we are ready to sample the irradiance from the cache.



This is roughly what the clipmap cache then looks like. As you can see, the cells are not exactly cubic, that is because we apply a periodic shift to the sample location, which causes the wobble. Described during the Advances in spatial hashing talk at siggraph 2022.

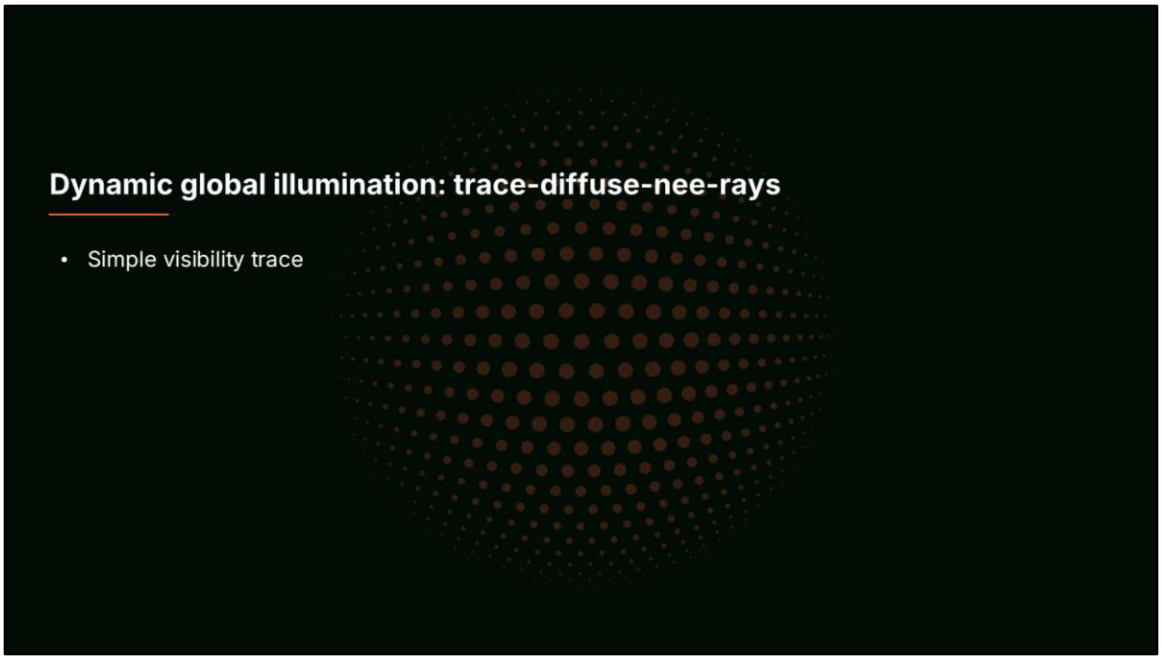


Now we have a coarse approximation of the irradiance throughout the scene. But this is far from what we can composite. As said before, we will use ReSTIR GI for high quality 1st to 2nd vertex paths, which gets us most of the high frequency detail from indirect illumination.

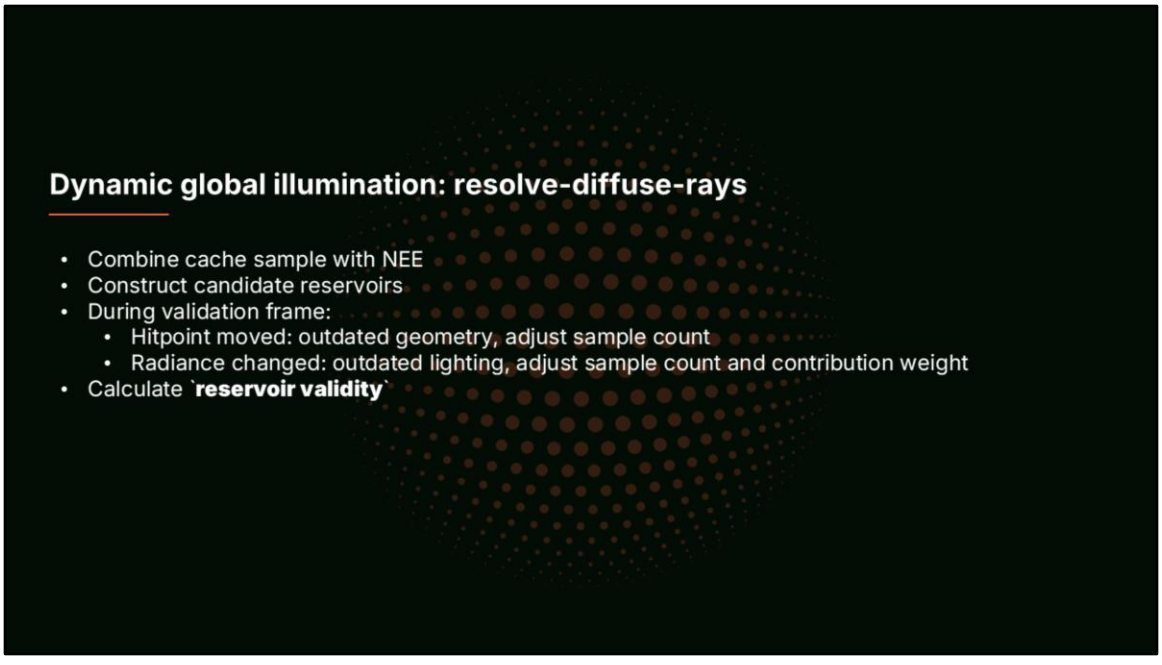


Our first pass sets up the rays. We always trace from the gbuffer, but which direction we trace depends on if we are currently doing a validation trace. The regular candidate traces are directed using uniform hemisphere sampling, as described in the ReSTIR GI paper. But every 3 frames we validate the reservoirs that we already have and re-trace in the direction that we have stored in our reservoir. This keeps the system responsive to changes in radiance and moving geometry. However, on disocclusions our reservoirs will not be valid either way, so instead we always trace a regular candidate ray. This pass only writes out a description of the ray.

Then we have to actually trace our rays. If we do not hit something we simply sample the sky. But if we do hit something we sample the clipmap and construct a next event estimation trace from the hit location. In this pass we do have to perform brdf eval’s and for this we always sample materials at a coarse mip level, to reduce bandwidth.

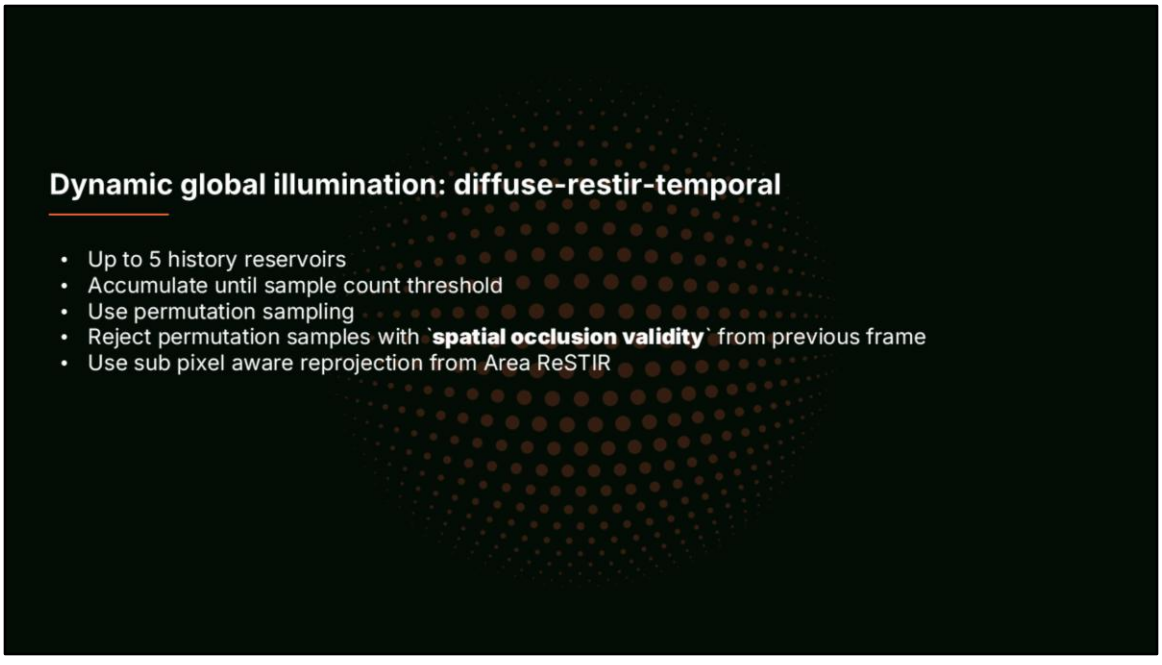


Now we have a separate pass for tracing the NEE rays. This is a simple visibility trace and writes the NEE contribution.

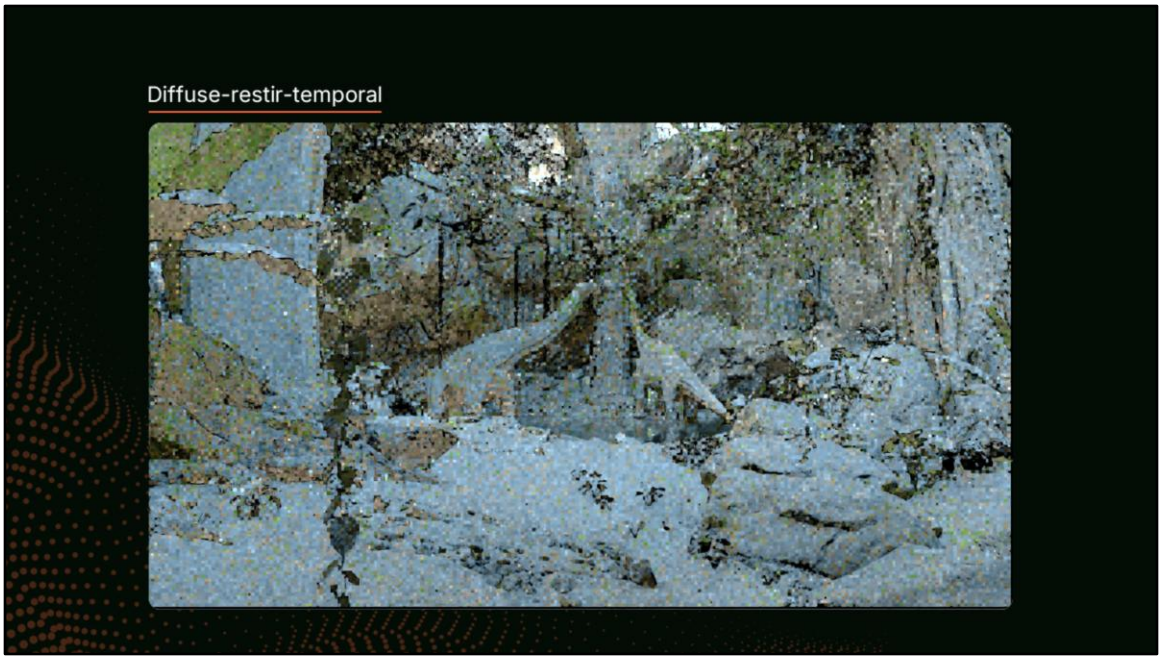


Now we have all the data we need to construct our candidate reservoirs. So on a regular frame we construct our candidate reservoir, but on a validation frame we adjust our history reservoir where needed. If the hit point moved, that means our entry is pointing at old geometry, in this case we adjust the sample count of our reservoir, to reduce its potency in following reuse passes as it is likely inaccurate. If the radiance changed, it also means that the scene has likely changed in some way. In this case we not only lower the sample count, but also the contribution weight to immediately adjust the reservoir output. At the end of this pass we also write out a reservoir validity value, which we will use to guide our temporal denoising pass.

After this resolve, we end up with our initial ReSTIR GI candidates which look roughly like this. Soooo we have a lot of work to do before we can actually use this.

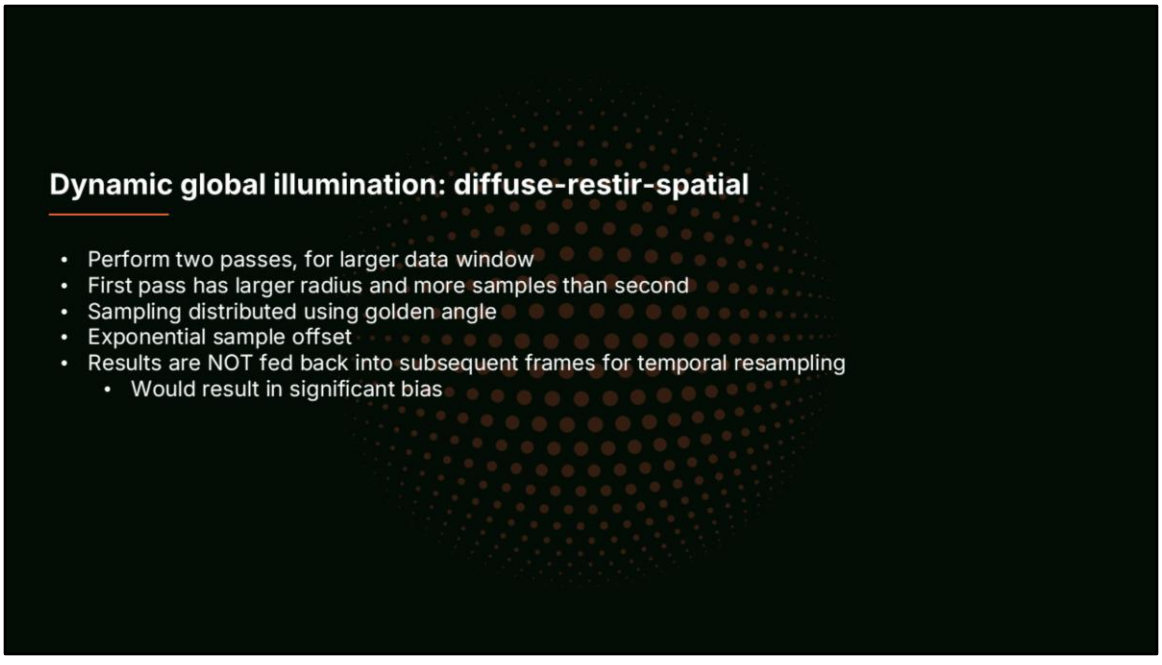


First, we have a temporal pass with which we increase the number of samples each reservoir has seen. Right now, each reservoir has seen one sample, the initial candidate. In this pass, we reproject into the previous frame's reservoirs, and at this location, we use permutation sampling to sample up to 5 history reservoirs, to try and reach a specific sample count threshold. This functions as both a temporal as well as a spatial reuse. We additionally guide this permutation sampling using a spatial occlusion validity mask, generated during previous frame's trace-spatial-validation-rays, which I will cover later. This mask describes areas that we can spatially resample without introducing too much bias. Another important detail of this pass is that for the canonical sample we use sub pixel aware reprojection as described in the Area ReSTIR paper to prevent common reprojection artefacts resulting from point sampled temporal reuse.

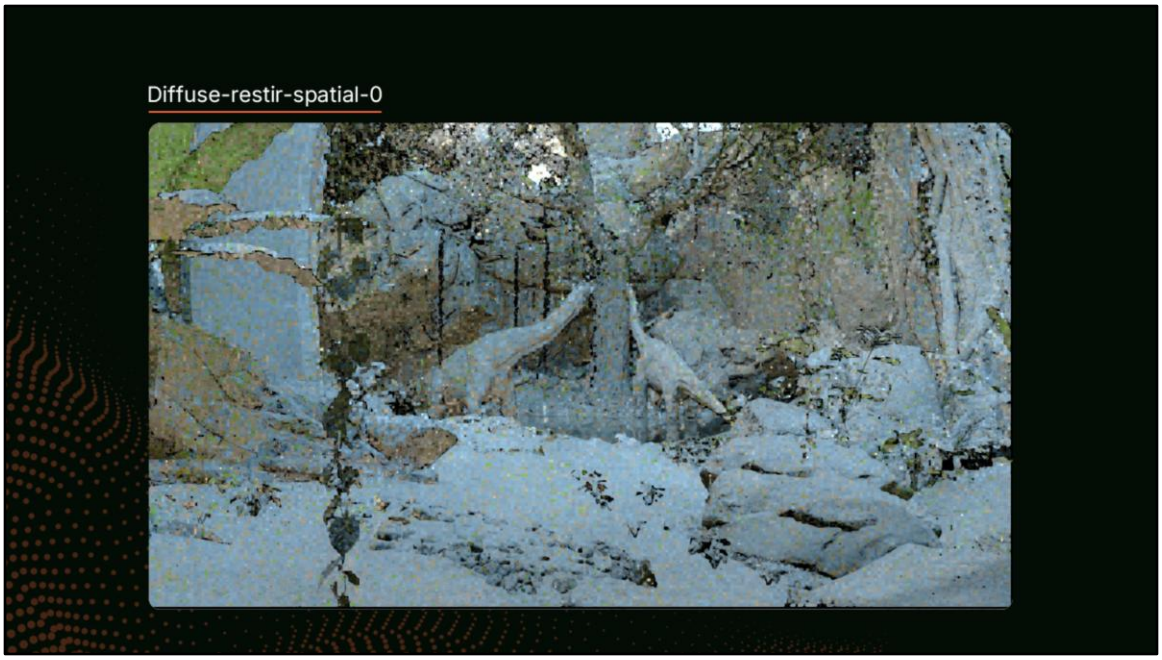


The result of this restir temporal pass we see here. This is already much better than just the candidates.

Here we see the benefit of permutation sampling. The top half has permutation sampling enabled, the bottom half does not. In this image we are moving the camera to the left, which results in disocclusion noise due to low sample counts. Even with permutation sampling enabled we get some noise at newly disoccuded pixels, but very quickly get at least some data in our reservoirs. Albeit quite blurry, but that is much better than noisy.

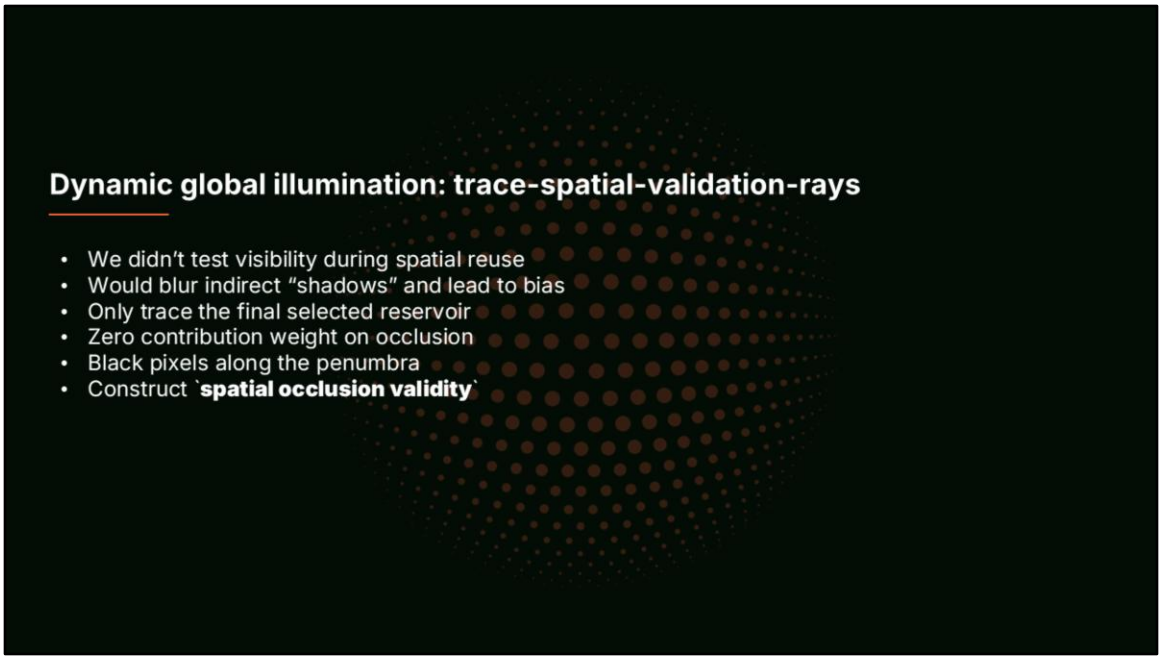


Next, we have two spatial restir passes. First one with a large radius taking 8 samples, and then one with a smaller radius taking 5 samples. The samples are distributed using the golden angle with an exponential radius. We do not trace validation rays in this pass as that would be prohibitively expensive, which means that the output is very biased. Because of this we do not feed the resulting reservoirs back into the ReSTIR system for the next frame.

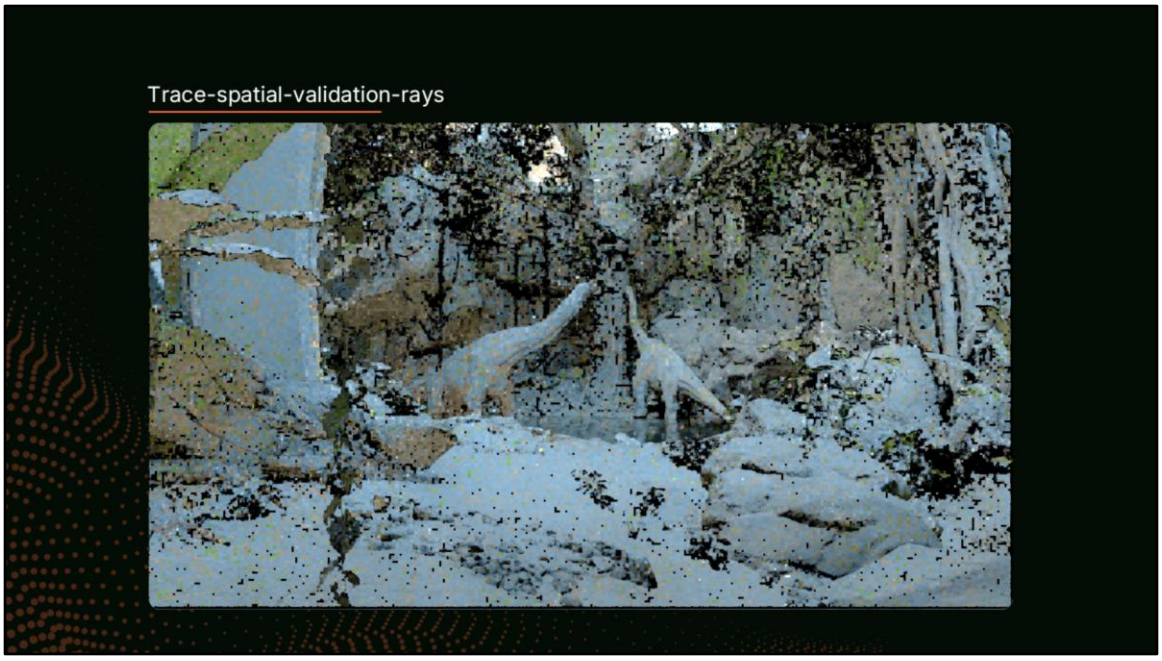


Here we see the result of the first spatial pass.

And here the second.

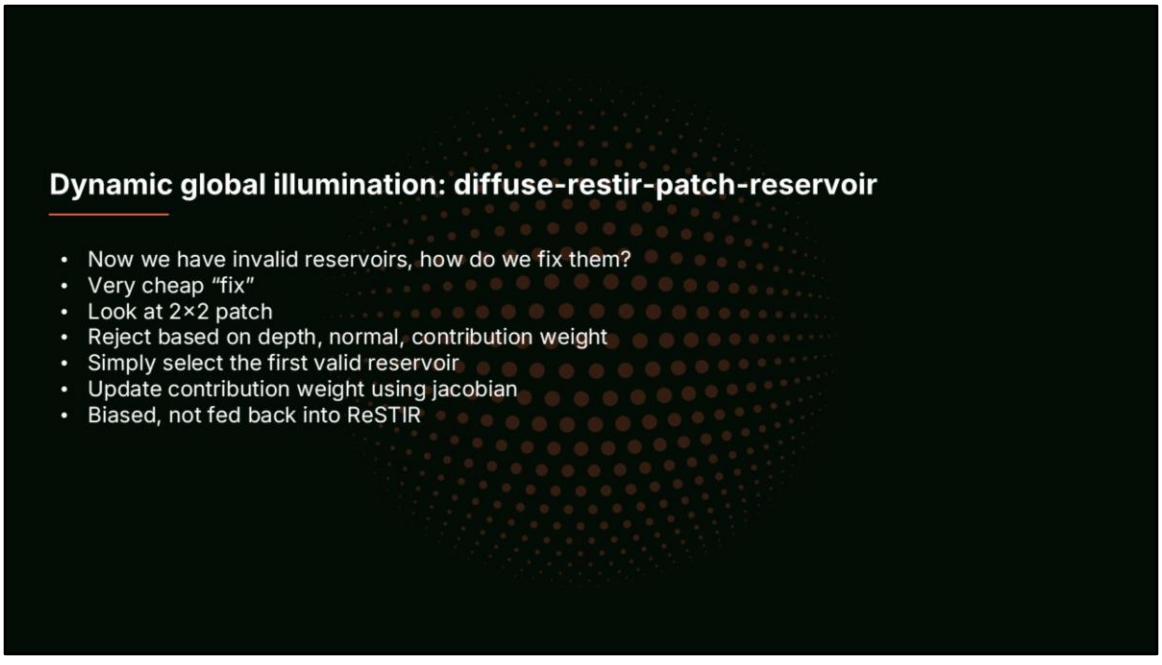


As mentioned, we did not trace visibility during our spatial passes, which means that we could have blurred across visibility boundaries. This pass aims to resolve that. We do that by tracing the final selected reservoir and zeroing the contribution weight on occlusions. This will mostly reject samples around the penumbra of shadows, and this information we additionally use to construct a spatial occlusion validity mask to guide the next frame's permutation sampling, and the current frame temporal denoising.

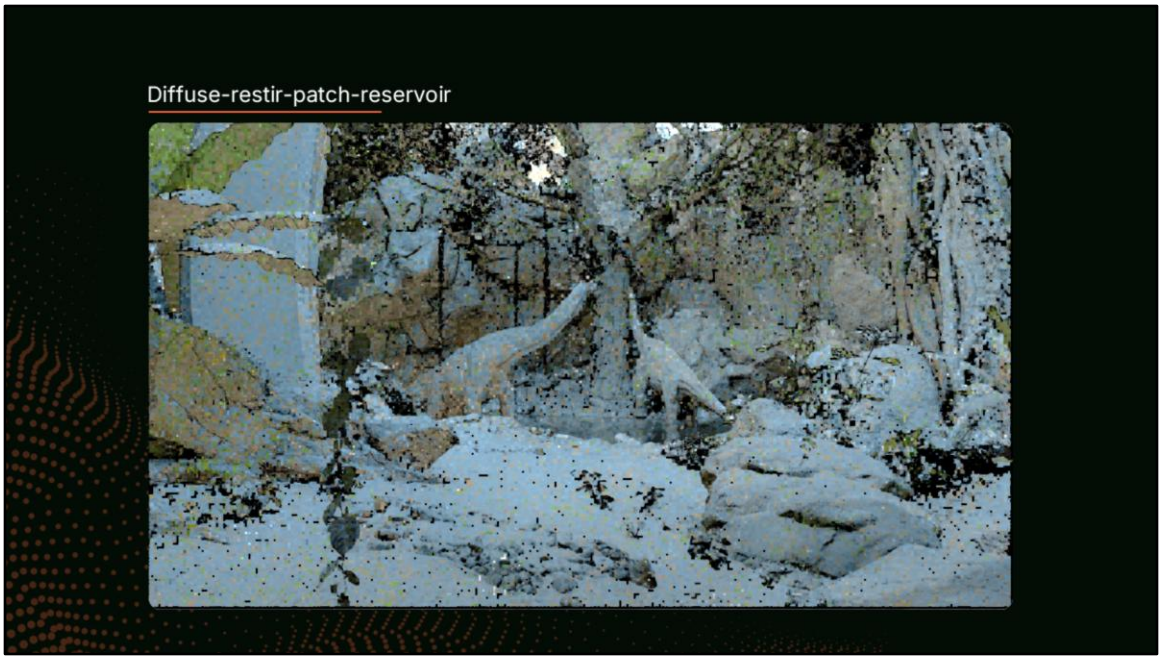


As you can see, we end up with quite a lot of pepper noise, especially around occlusions.

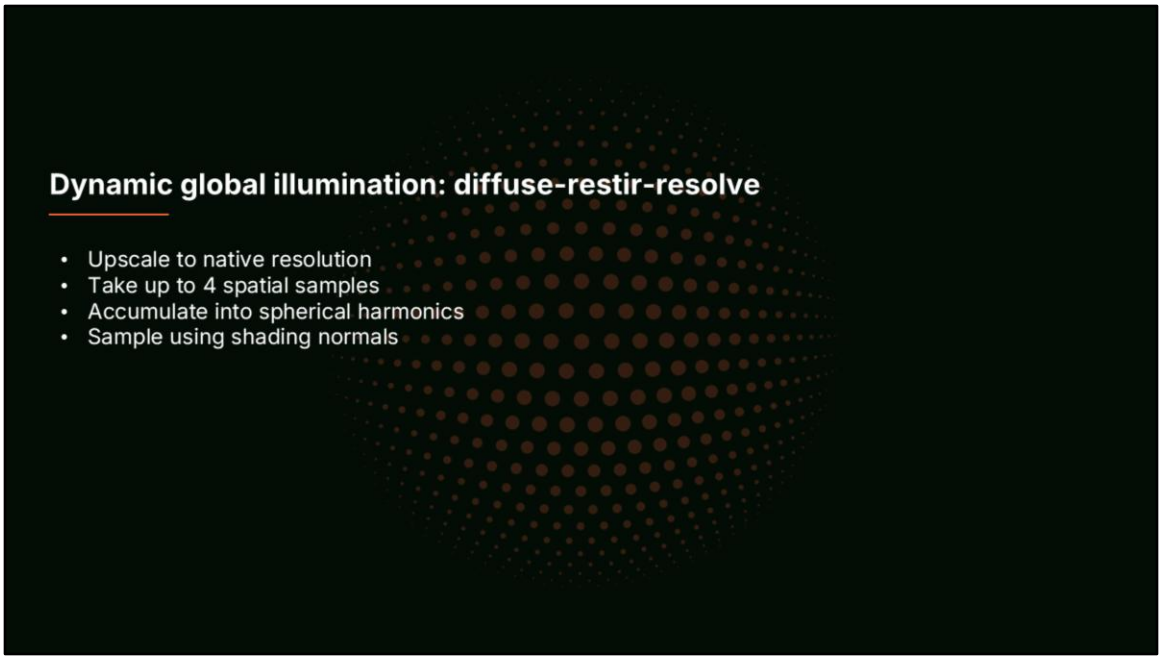
If we accumulate and temporally reproject this spatial occlusion validity, we end up with a mask like this. Here you can see that underneath this plant there is low spatial validity, which we use during permutation sampling to reduce the contribution of neighbors. This keeps indirect “shadows” sharper. And we use it to apply a more aggressive temporal denoise at the end, to reduce boiling.



Now, to resolve our newly created pepper noise, we apply a simple fix. For each of the rejected reservoirs, we simply look through a 2x2 neighborhood and find the first reservoir that passes some simple rejection criteria for depth and normal similarity, and contribution weight. Of course, we still have to update the contribution weight using the jacobian. And once again, this pass is quite biased, so we do not feed it back into the ReSTIR system for the next frames.

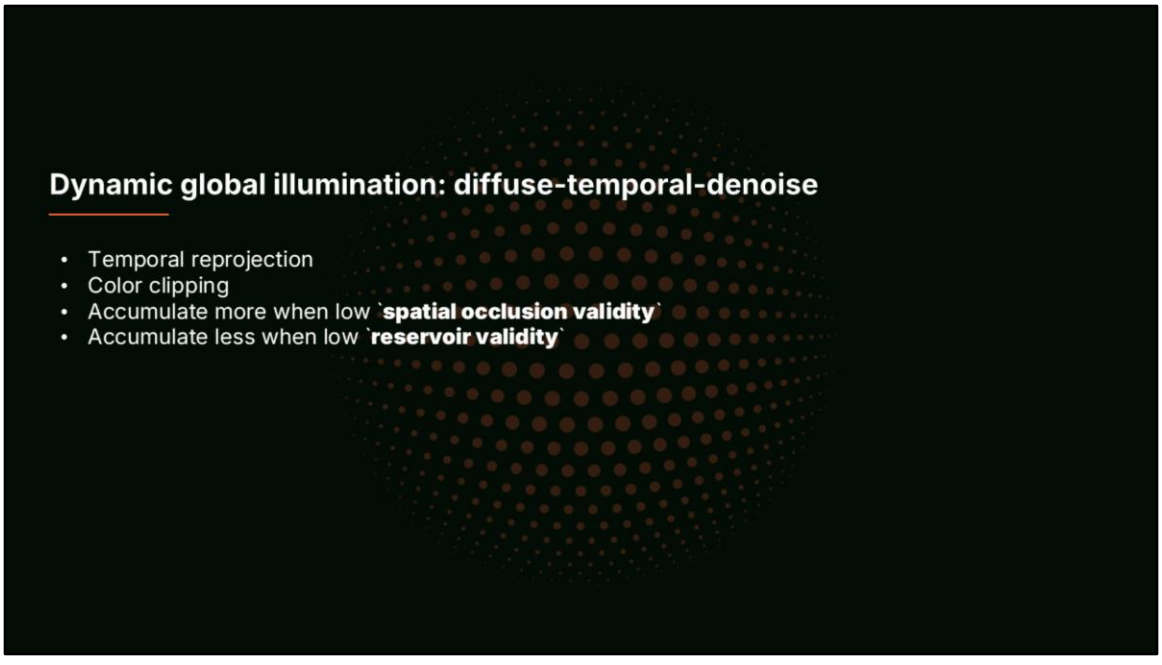


This patch eliminates some of the pepper noise, although not all of it.



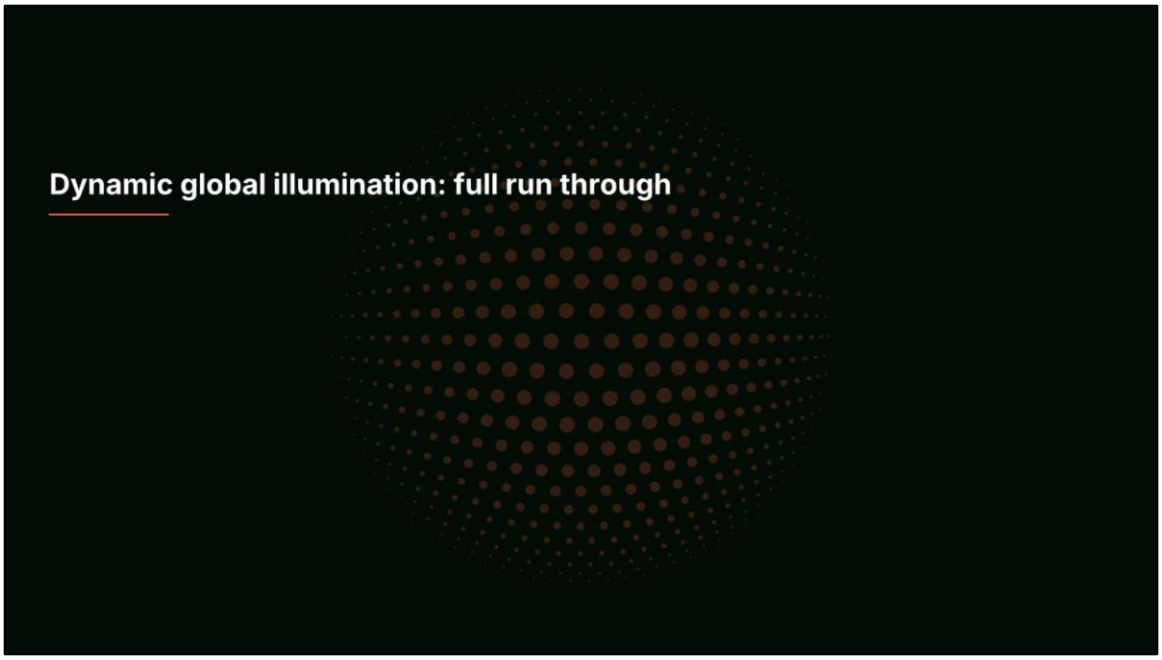
Next we have our restir resolve. This actually upscales to the rendering resolution. In this pass we take a 4 sample weighted average of the reservoirs, again using golden angle, but with a small radius this time. Then we construct a set of L1 spherical harmonics from these samples, which smoothes out the signal, and we sample these using the shading normals.

So this is what we have after going to full res. However, there is still a bit of noise which we want to get rid of.



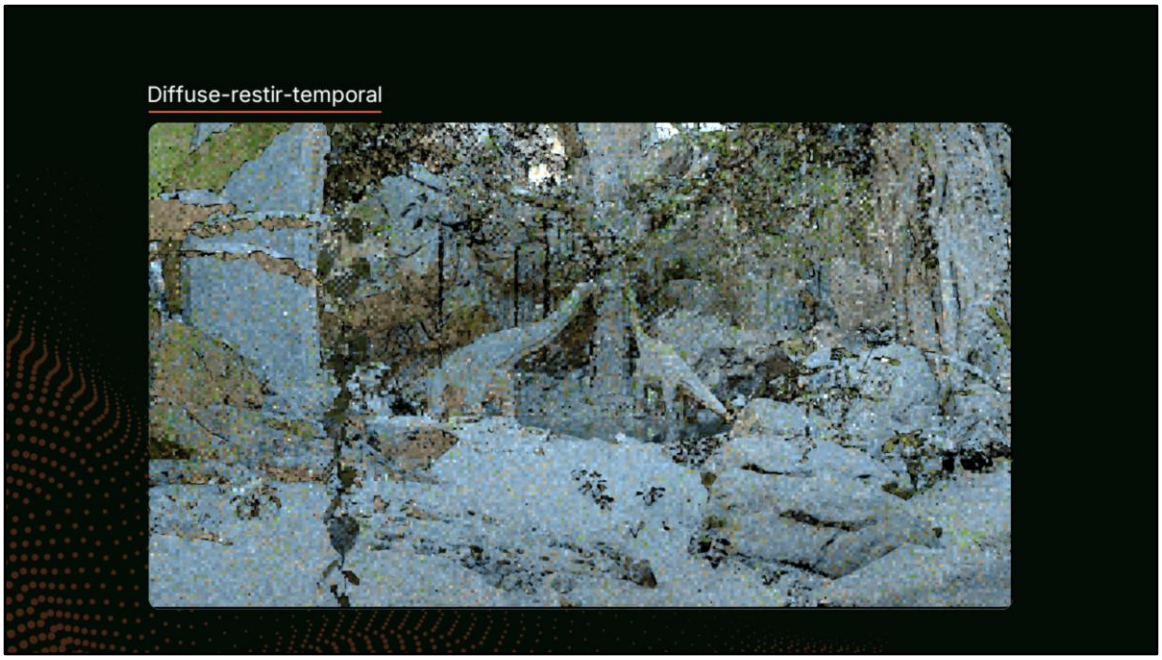
And that is what our final pass is for. This is a simple temporal reprojection pass with color clipping, similar to TAA. Here we use both the `spatial occlusion validity` to accumulate more aggressively along visibility edges. And we use the `reservoir validity` to reduce accumulation when we detect changes in the scene geometry or lighting. This makes the temporal denoise a bit more responsive.

And here we see the final result. At this point the GI is a fairly stable signal, ready for compositing.

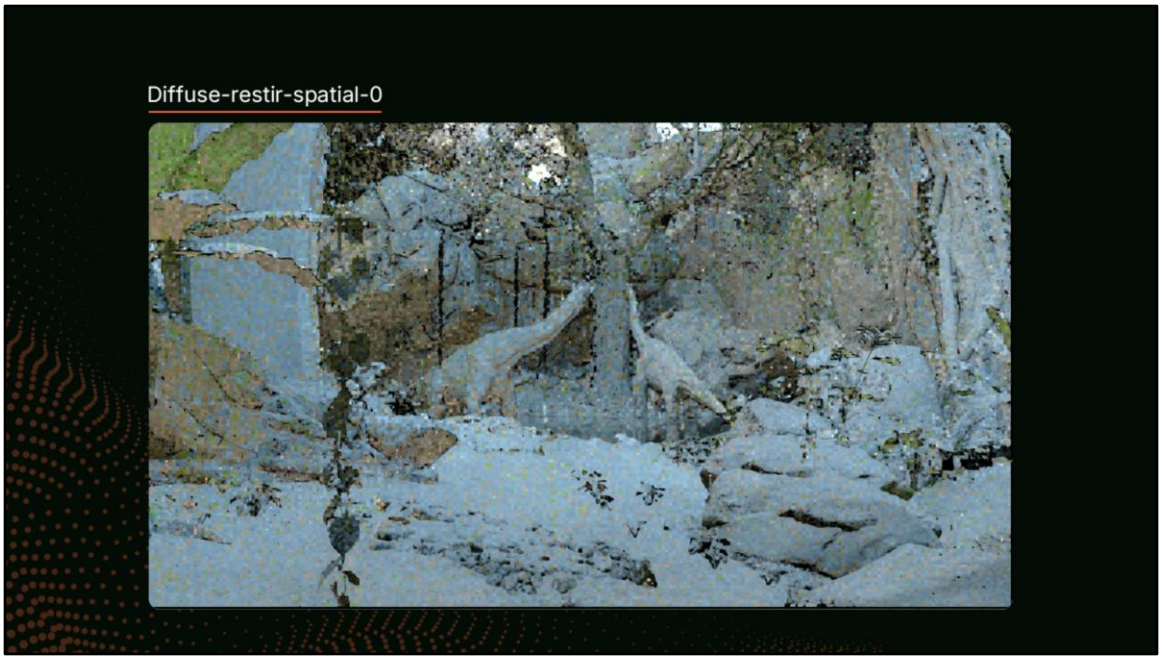


So, before showing you what the end result looks like. I want to quickly go over the intermediate debug renders again as a final overview highlighting the effect of each of the steps we took.

First we started with our candidates.

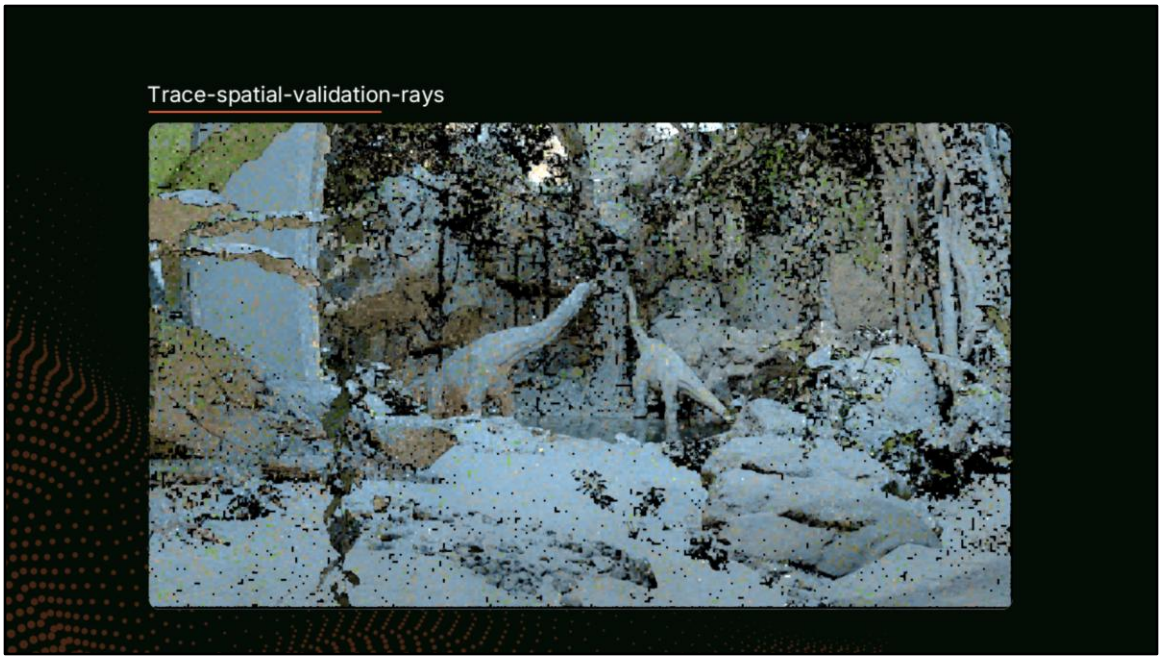


Next, we apply temporal restir.

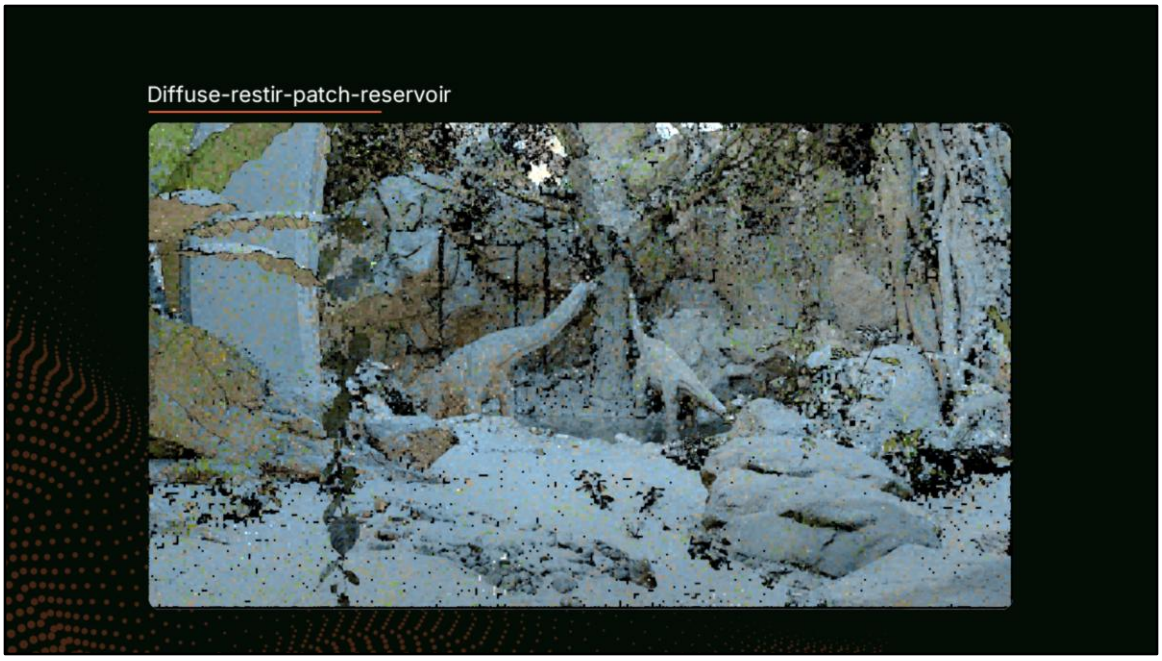


Next, our first large radius spatial filter.

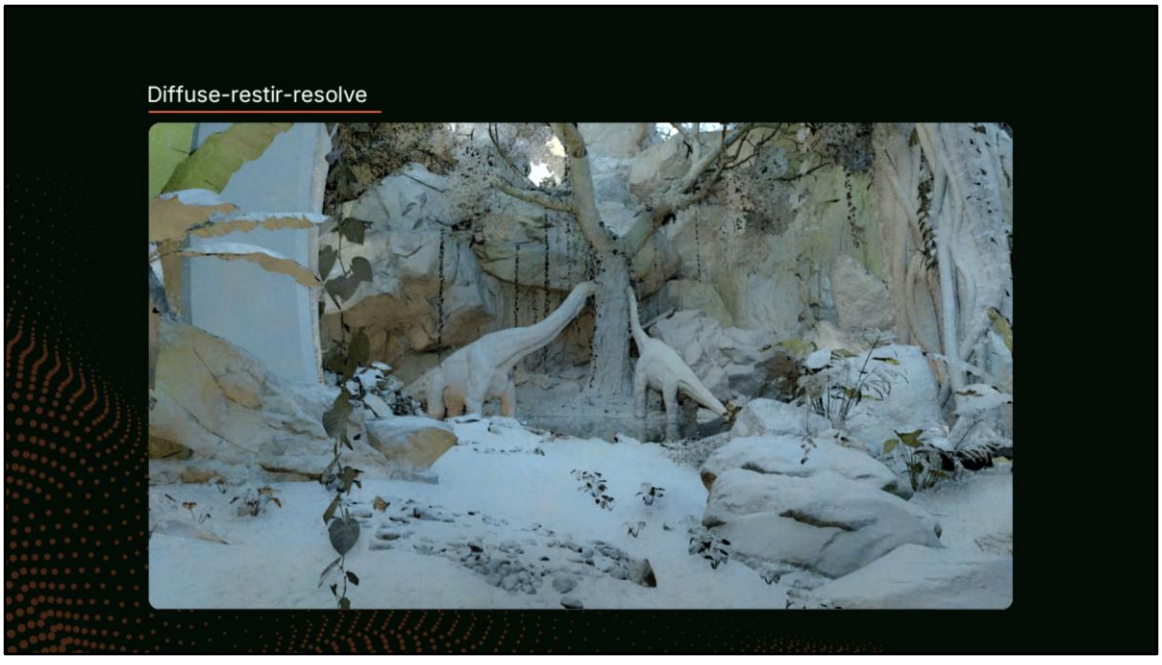
Then our smaller scale spatial pass.



Then trace validity of our spatial resampling, which adds back occlusion detail, at the cost of some pepper noise.

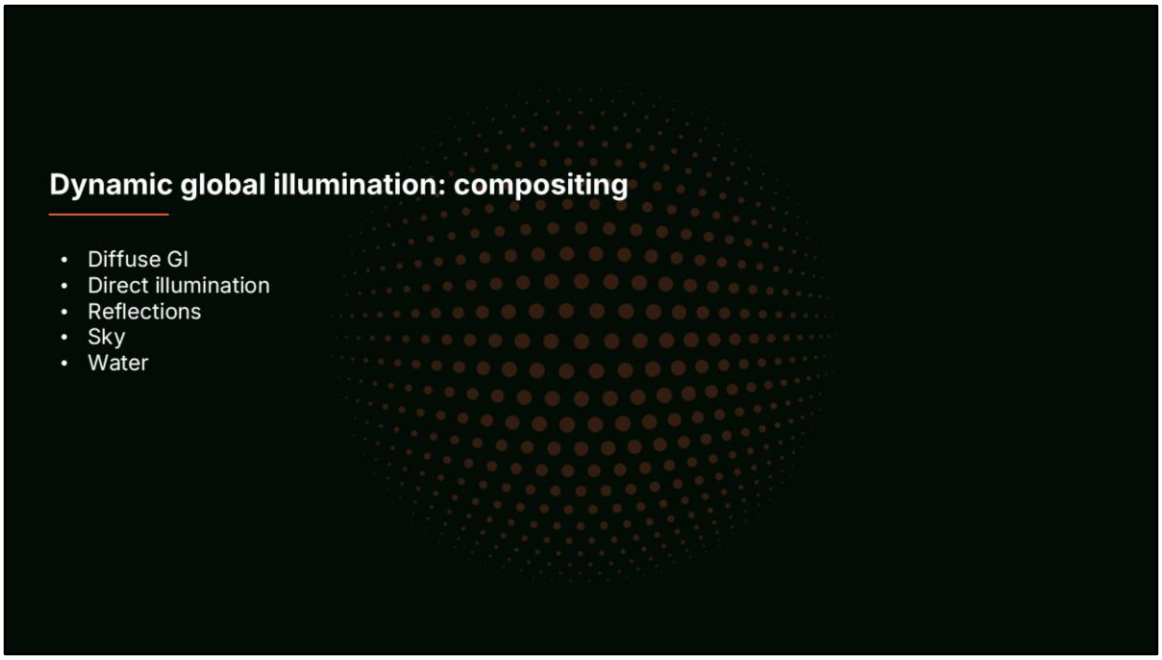


Then we fix part of the pepper noise, but not all of it.

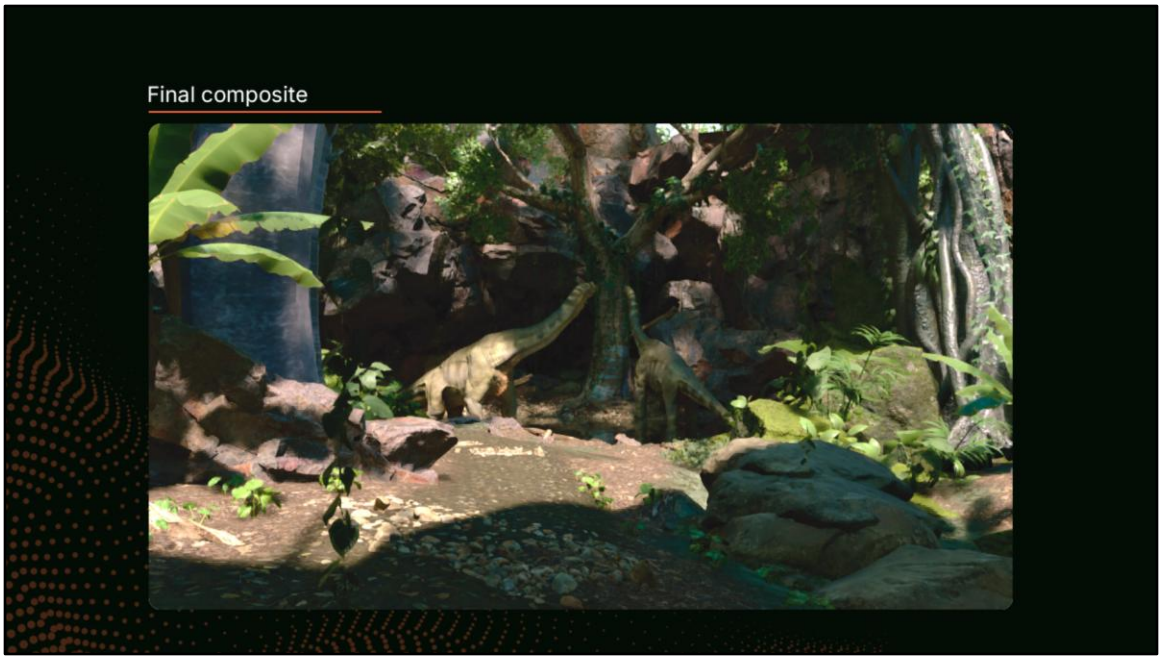


Then we resolve to full screen.

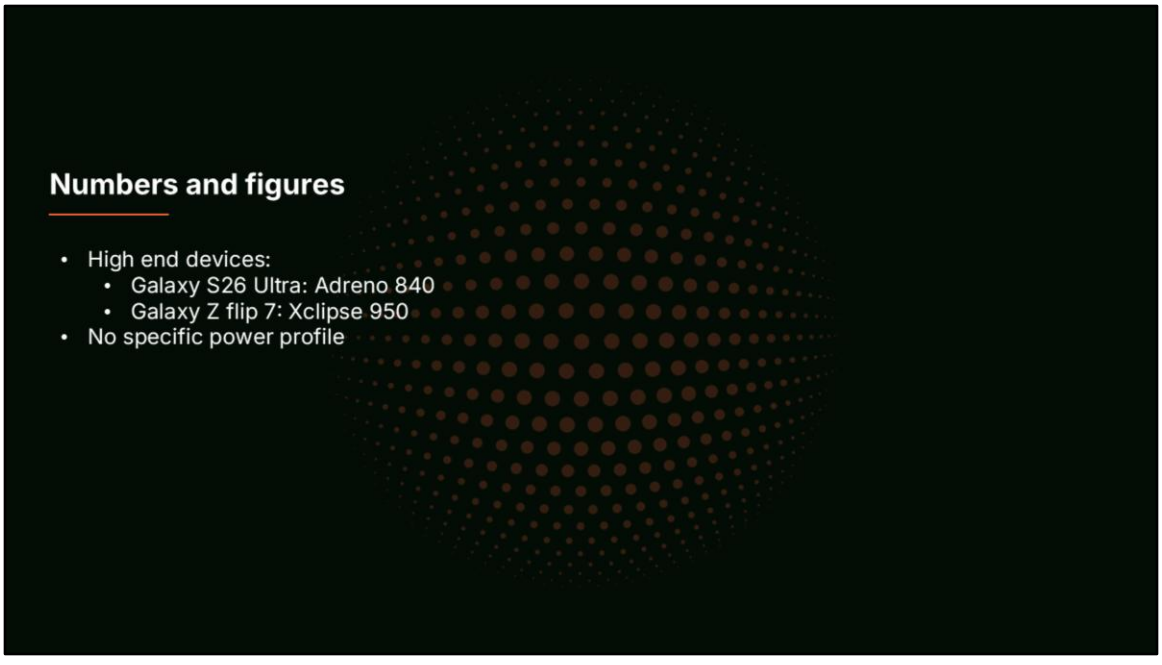
And finally the temporal denoise pass.



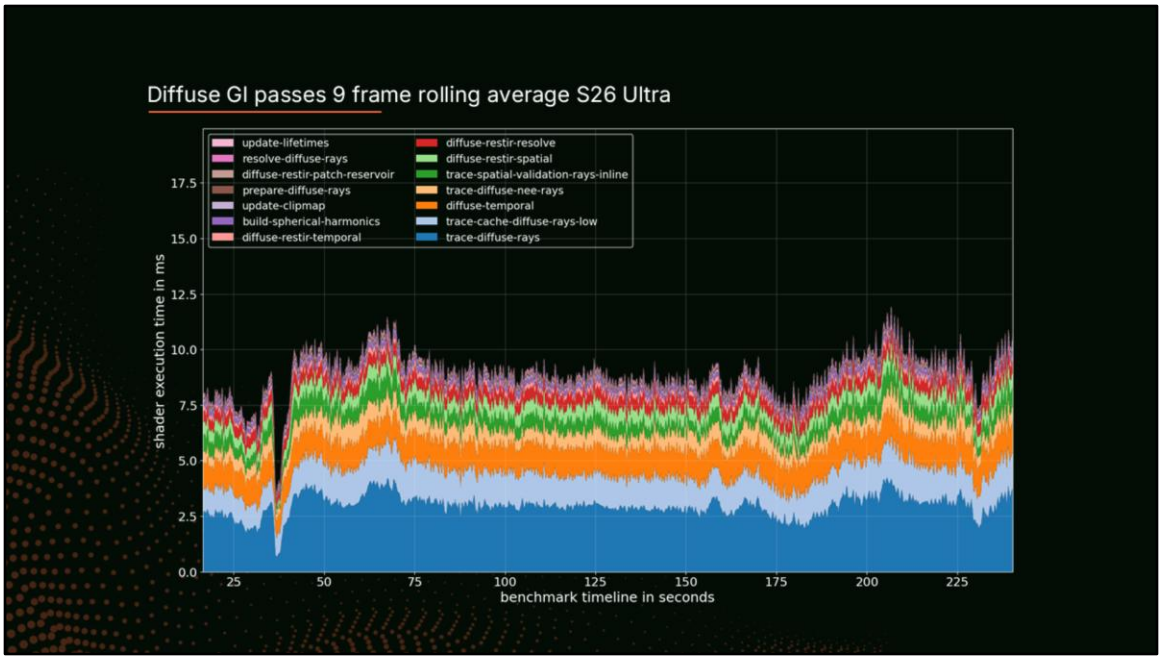
Now we have our diffuse GI, and we can composite it together with direct illumination, reflections, sky and water.



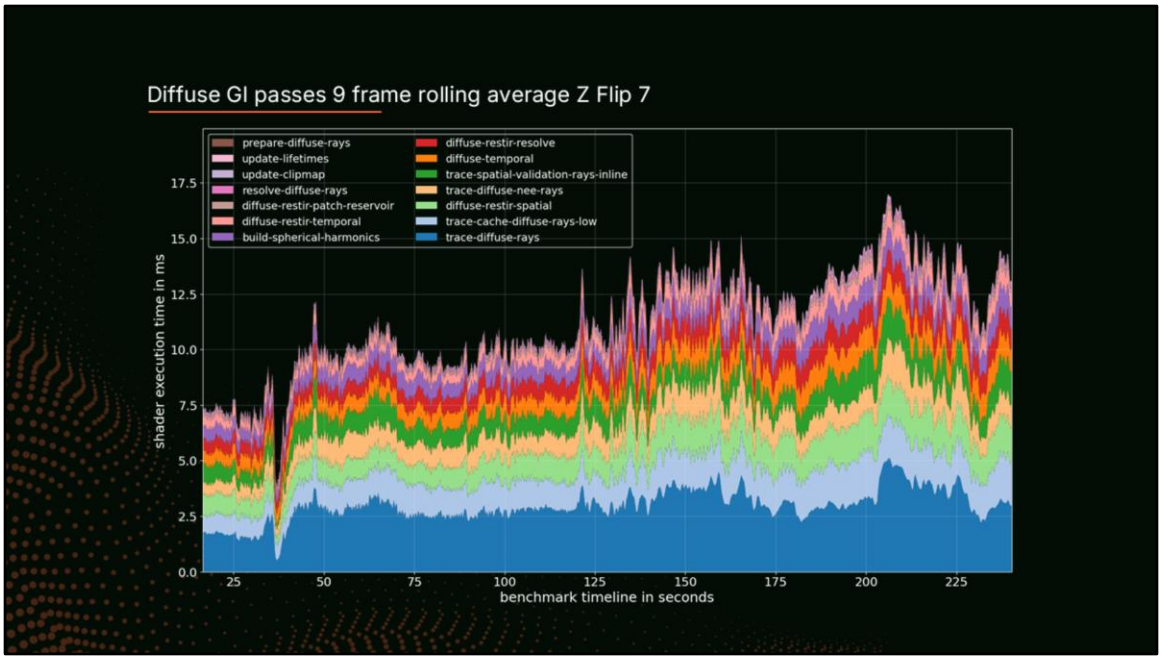
And then we end up with an image like this.



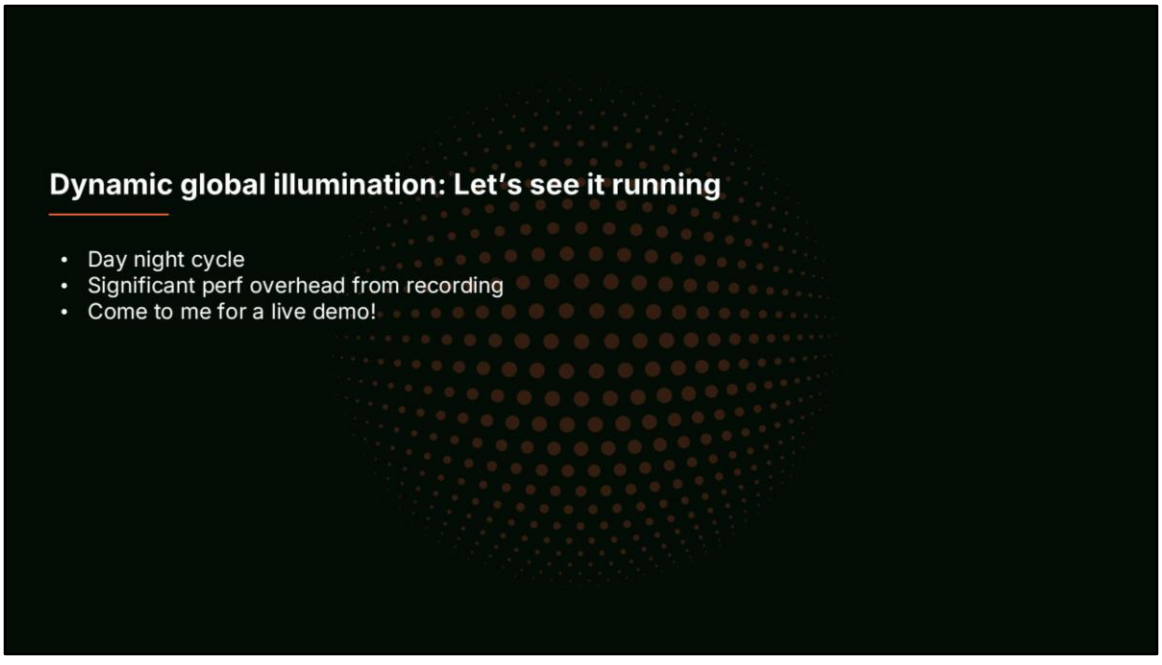
Now, onto some numbers and figures. Next I will be showing measurements on two high end mobile devices, namely the Samsung galaxy S26 Ultra with an Adreno 840 GPU, and the Samsung galaxy Z flip 7 with an Xclipse 950 GPU. We did not set any specific power profile.



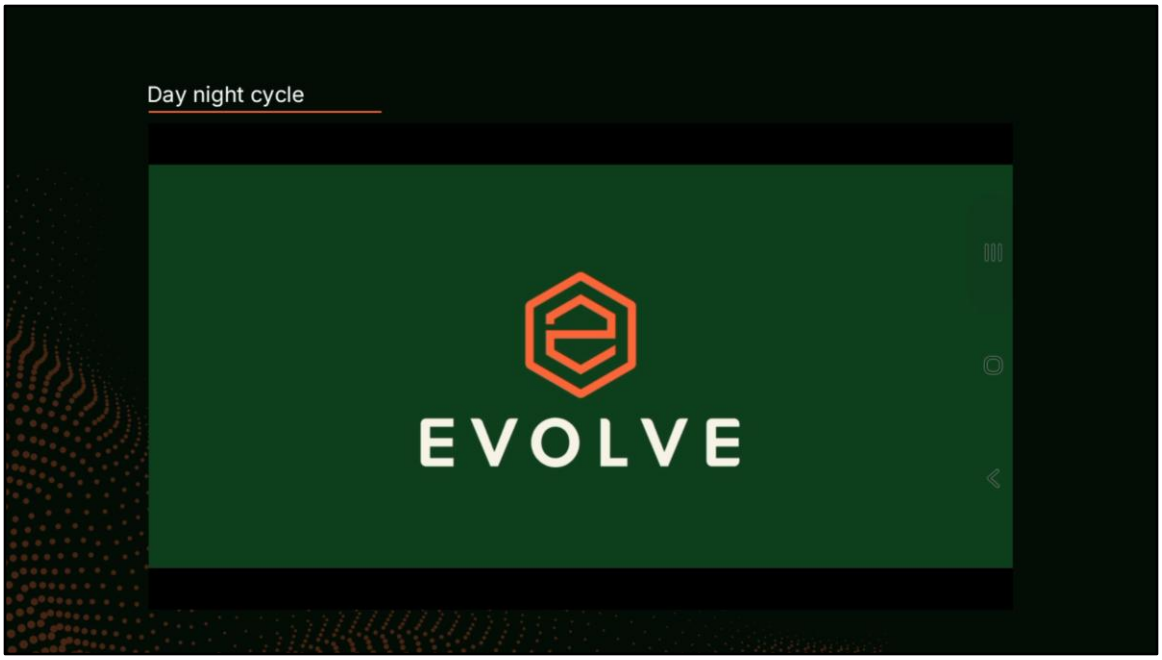
First, this graph shows a stackplot of a 9 frame rolling average of all diffuse GI passes on the S26 Ultra, throughout the Evolve benchmark. On the x axis we have the timeline of the benchmark, and on the y axis we show the cumulative shader execution time in milli seconds. As can be seen, for most of the timeline we sit below 10ms for the entire diffuse GI system, which shows that with some further optimizations, and with the next generation of mobile devices we might be able to run systems like ReSTIR GI to some success on mobile.



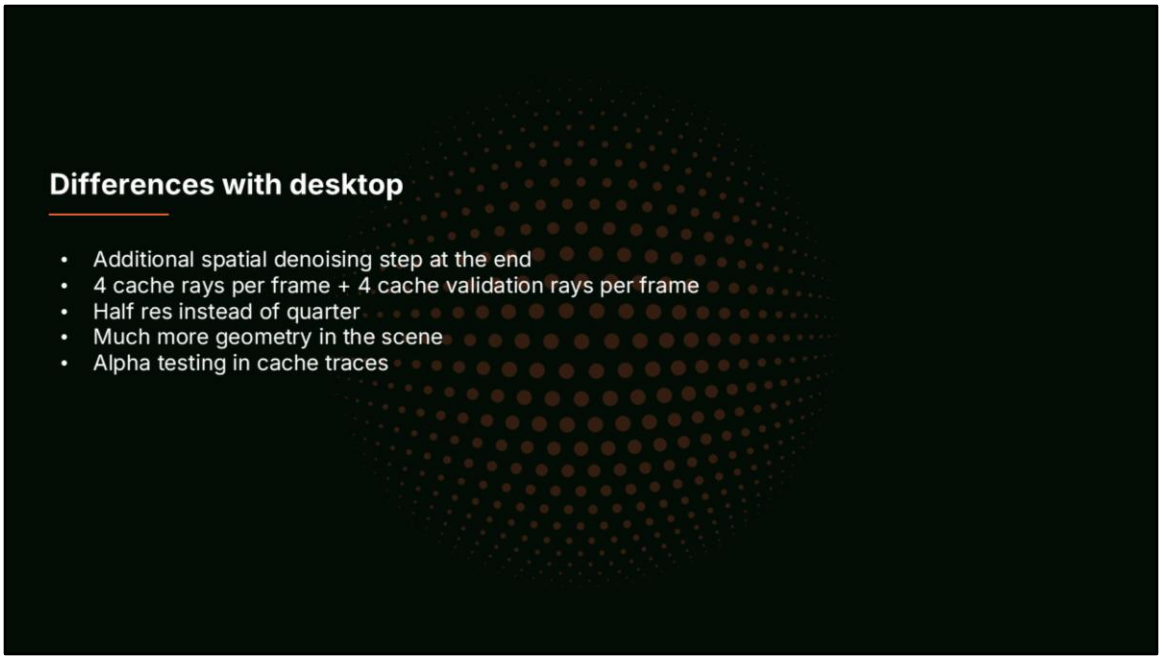
Now, we also took a similar capture of the Z flip 7. This device seemed to have stayed around 10ms, until about halfway throughout the benchmark where it gets throttled. Interestingly, the diffuse-restir-spatial pass was much more expensive, while the diffuse-temporal-denoise pass was cheaper compared to the S26 Ultra.



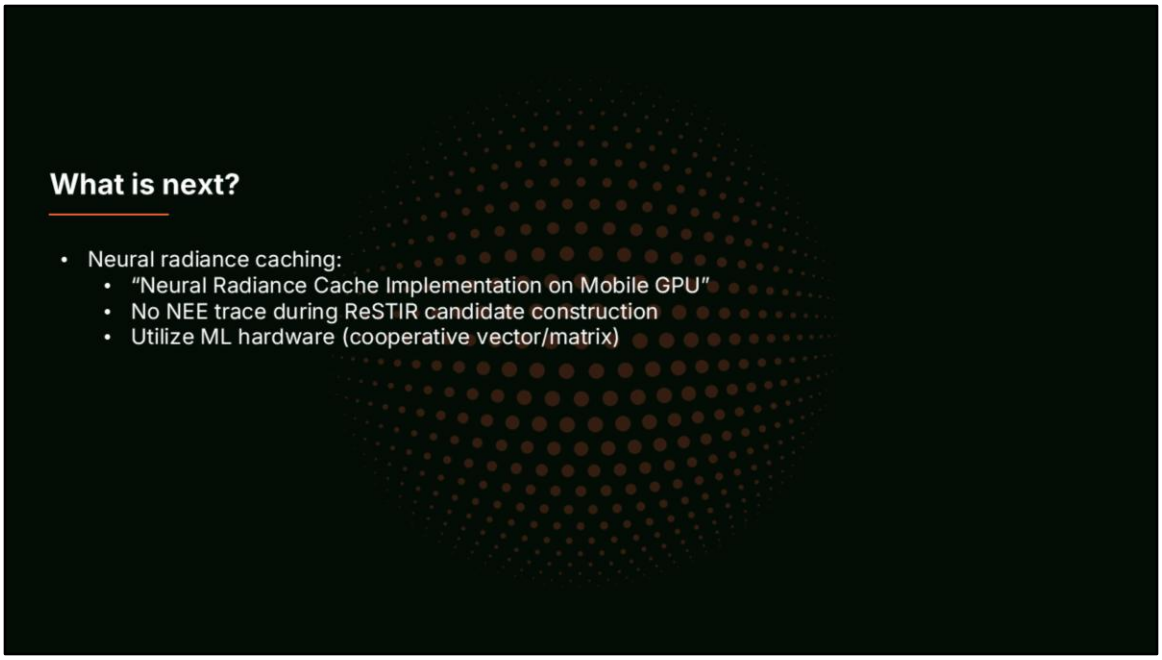
We recorded a little demo with a day night cycle. However, getting a representative recording was much harder than we thought. So, in the video it looks like the framerate is much lower than it is on the device. But we have both the z flip and the S26 Ultra with us, so please come by after the talk to have a look.



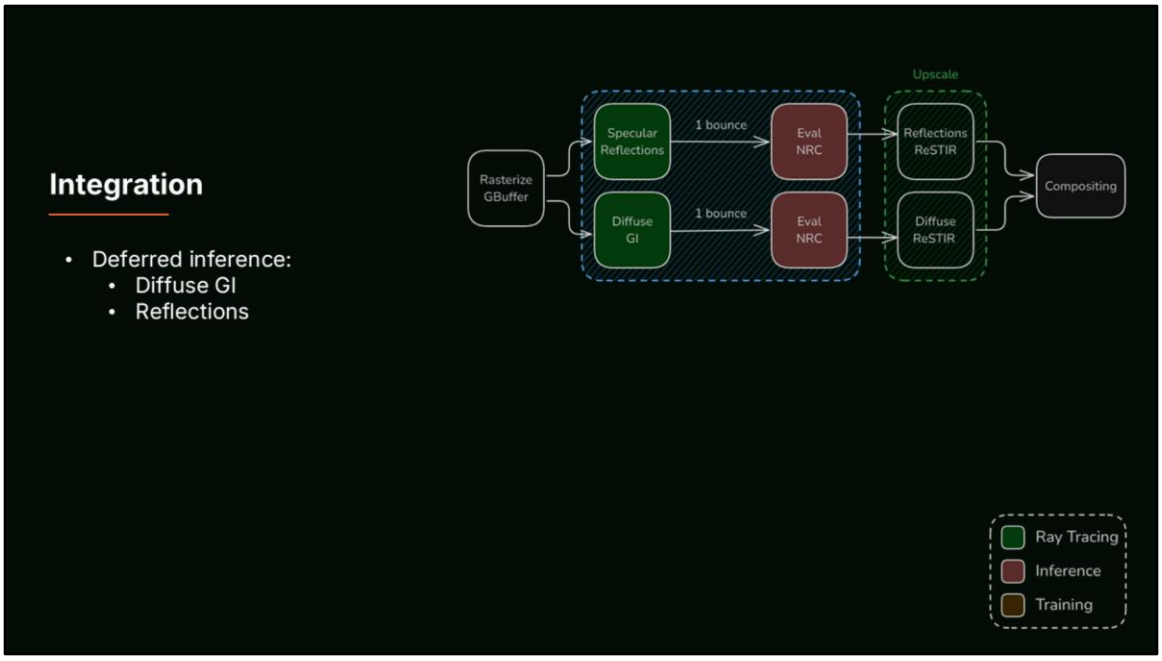
Now just imaging this running at a steady 27 frames per second. We start with a sunrise. See the light shine through the leaves. And finally end with a sunset.



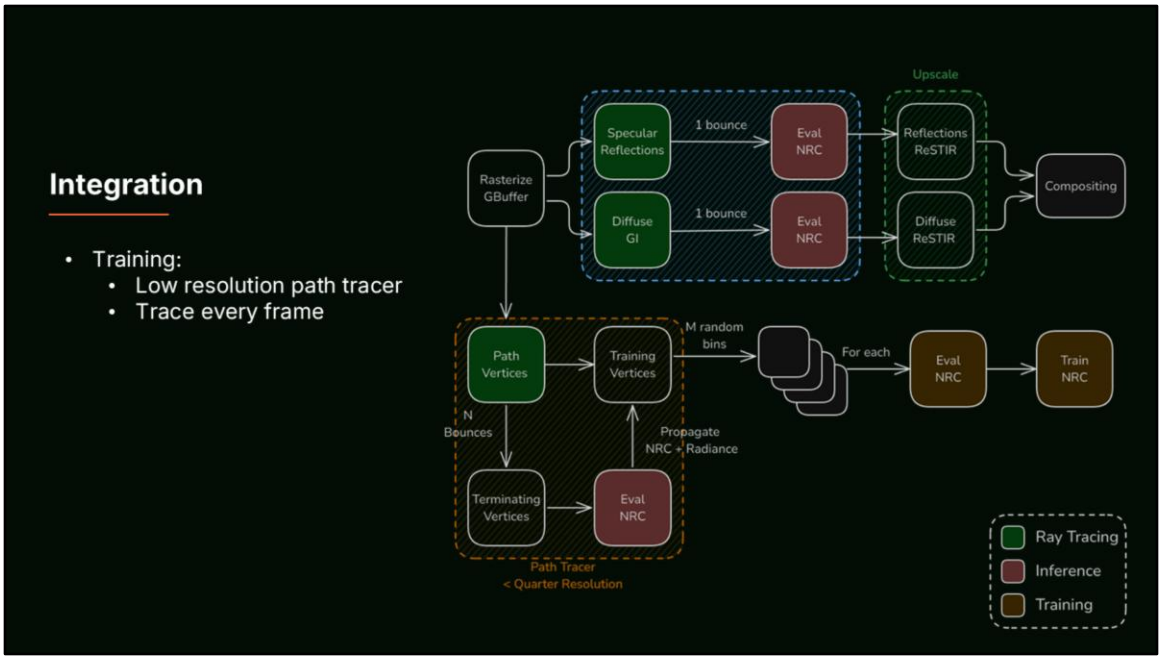
As mentioned at the start, Evolve runs both on desktop and mobile. And I mostly discussed techniques that are traditionally only used on desktop, so lets quickly cover the most important differences between our mobile and desktop version of the benchmark. On desktop, at the end after our temporal denoising pass, we have one additional spatial denoising pass. On desktop we trace multiple rays per clipmap cache cell. Instead of 1 trace per frame, we trace 4, and additionally we trace 4 validation rays. Along with this we also trace global illumination on half res instead of quarter res. All these traces are simply too much for mobile, and in our scene the increased level of responsiveness was not the priority. Next on desktop we have much more geometry. And we perform alpha testing for cache traces. Which, as said before, might darken the scene a bit, but the performance tradeoff is worth it for us.



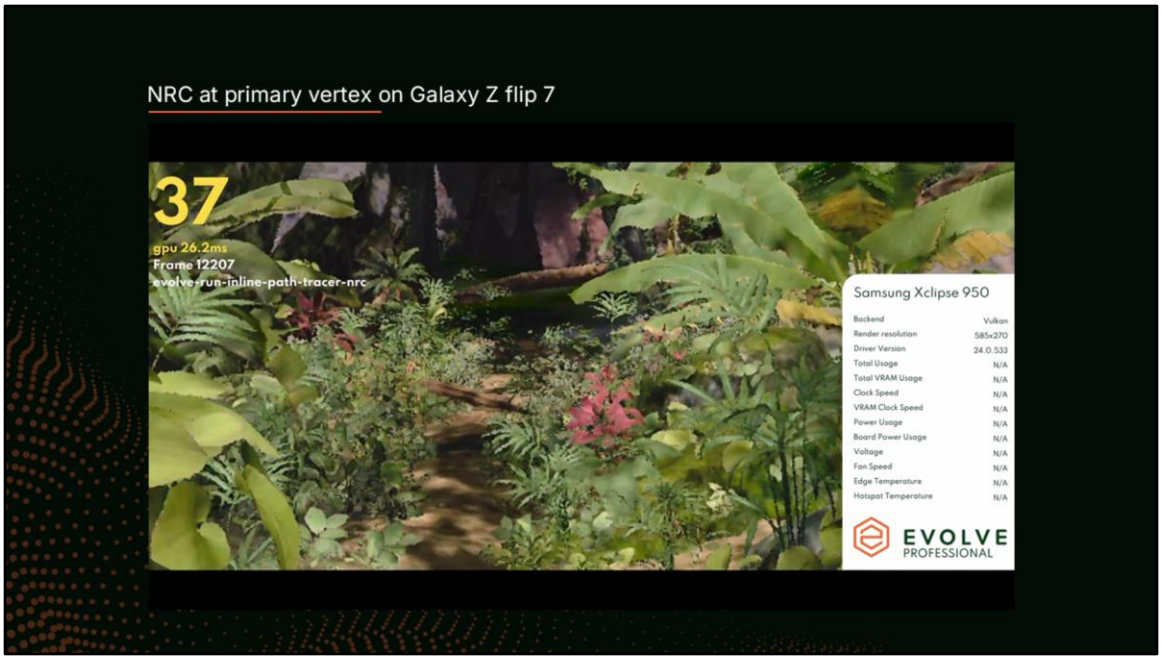
I will only quickly cover what's next, as I presented this during siggraph asia last year. One of the next steps we are investigating is neural radiance caching. The idea being that we train a neural network to predict radiance throughout the scene. In our experiments this is not of high enough quality to sample directly on the primary vertex/on the gbuffer, but definitely can be sampled at the 2nd vertex, similar to the clipmap cache, but with possibly a better quality to performance tradeoff. We have seen that NRC can effectively store direct illumination, so we can get rid of the NEE trace during restir candidate construction for example. And another major benefit is that we can use the high throughput ML hardware.



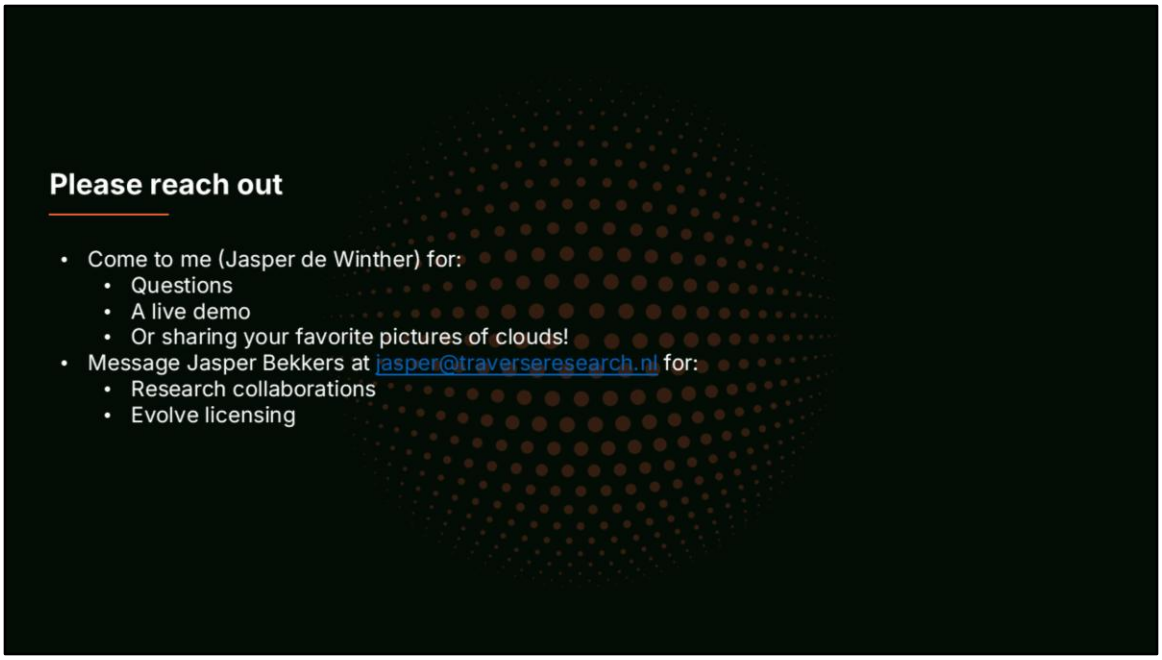
I will quickly showcase what the NRC hybrid renderer integration currently looks like for us. In the diagram in the top right you can see that on the left we start with our gbuffer. Then inside the reflection and diffuse GI systems we trace one ray, from the 1st to the 2nd vertex. After which, instead of sampling the clipmap, we evaluate NRC, the result of which we the use in the following restir passes. But this is only half of the system, we still have to train the network for it to adapt on the fly.



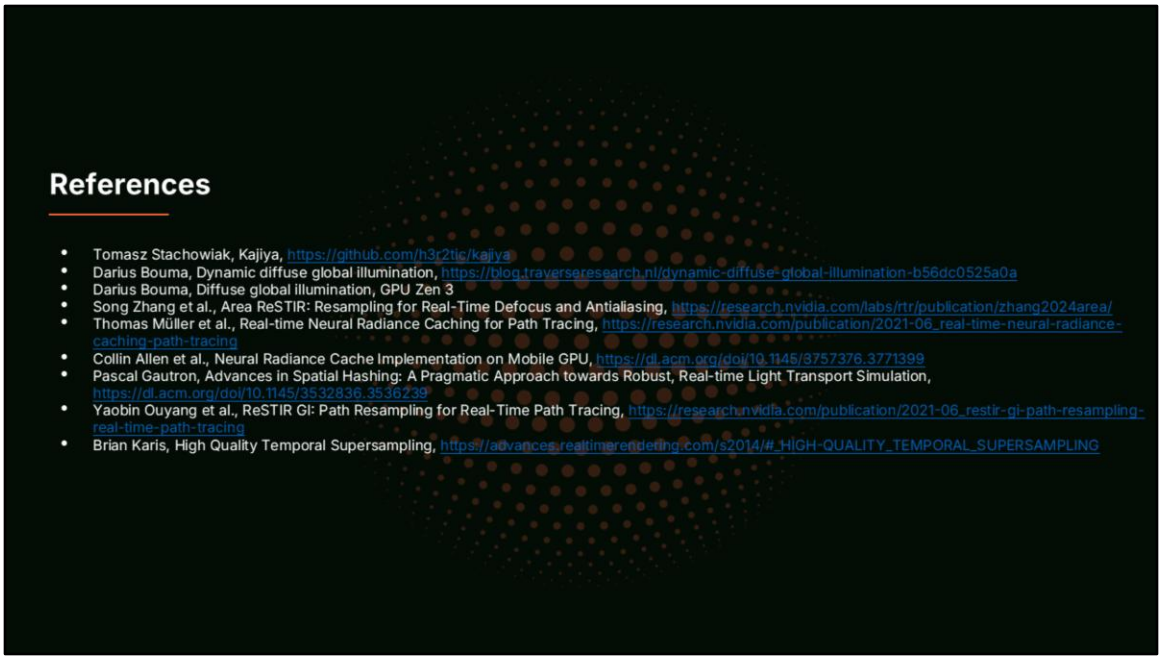
So, every frame we run a very low-resolution path tracer, and just like the clipmap at the end of each trace we recursively sample its own cache to get the “infinite bounce” effect. This path tracer then stores the vertices it encountered, and this is our training data. We then calculate the loss, do backpropagation and adjust the weights of the network, which gets us closer to predicting the actual radiance throughout the scene.



Here we have a video of this happening in real time. We show a debug render of the NRC being evaluated at the primary surface, which as I said is not what you should composite, you would sample it at the 2nd vertex, but the video illustrates the kind of quality we can currently achieve with NRC on mobile. As can be seen, this NRC is able to reconstruct direct as well as indirect lighting, and even specular highlights. This was captured on the Z flip 7.



Finally, please reach out! Come to me, Jasper de Winther, if you have any remaining questions, or if you want to see a live demo, or if you simply have some awesome pictures of clouds. And feel free to message Jasper Bekkers if you are interested in research collaborations or Evolve licensing.



These are the most important sources we used throughout the presentation.

Thank you.