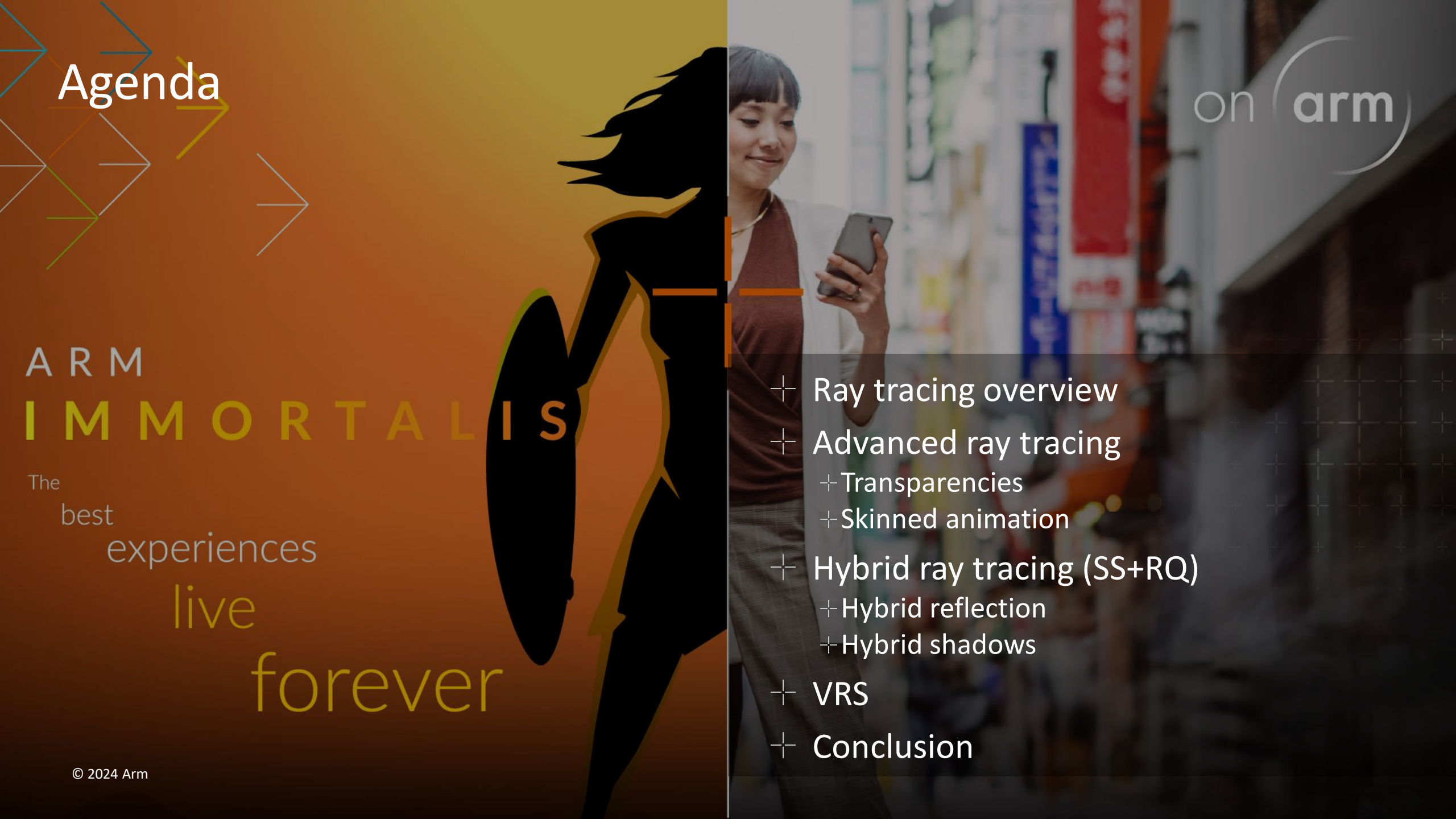
Vulkanised 2024
The 6th Vulkan Developer Conference
Sunnyvale, California | February 5-7, 2024

Realistic graphics with Ray Tracing on Mobile

Iago Calvo Lista, Arm

Agenda



- Ray tracing overview
- Advanced ray tracing
  - Transparencies
  - Skinned animation
- Hybrid ray tracing (SS+RQ)
  - Hybrid reflection
  - Hybrid shadows
- VRS
- Conclusion

Ray tracing on mobile is here!

- Last year's talk
  - Devices with HW ray tracing just appeared
  - 40-60 FPS
  - Introduction to shadows, reflections and refractions

- Today
  - More devices with HW ray tracing
  - 60-90 FPS
  - More data and advanced techniques to share

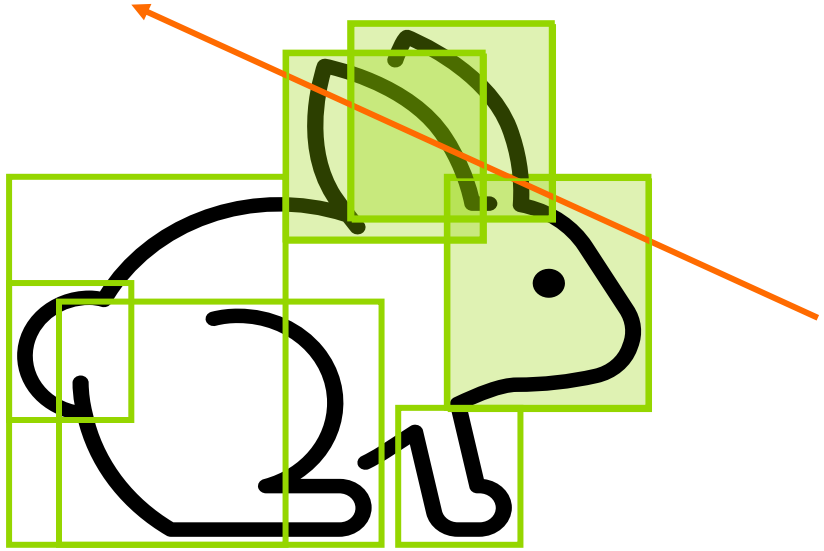
Model	SoC	GPU
Vivo X90	D9200	Immortalis-G715
Vivo X90s	D9200+	Immortalis-G715
Oppo Find X6	D9200	Immortalis-G715
Vivo X100	D9300	Immortalis-G720

arm

Ray tracing overview

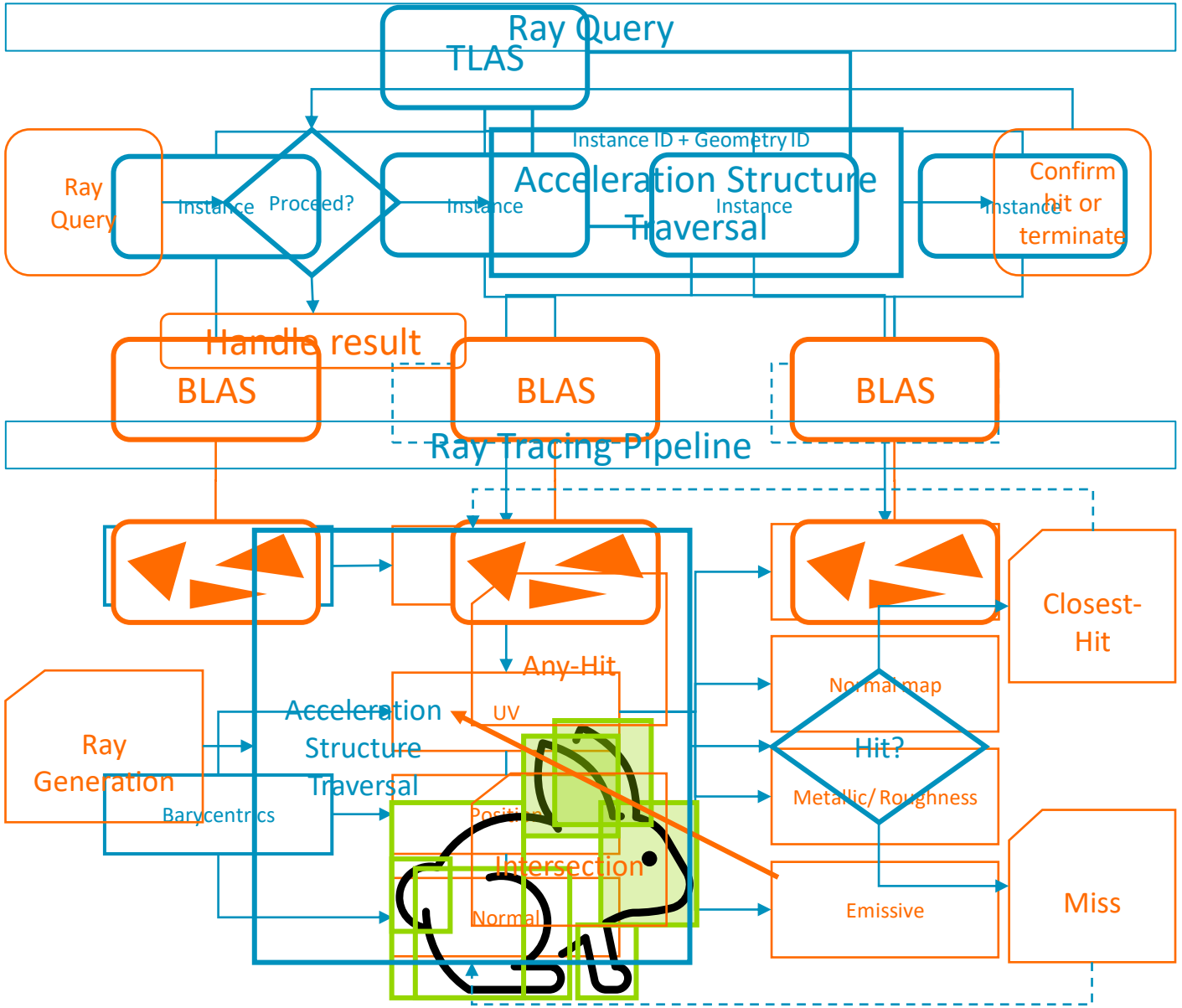
What is Ray tracing

- API for ray intersection tests
  - User launches a ray
  - Test the ray against the scene geometry
  - Returns the ray closest hit
- Main use case: rendering
  - Other: physics simulations



Ray tracing in Vulkan

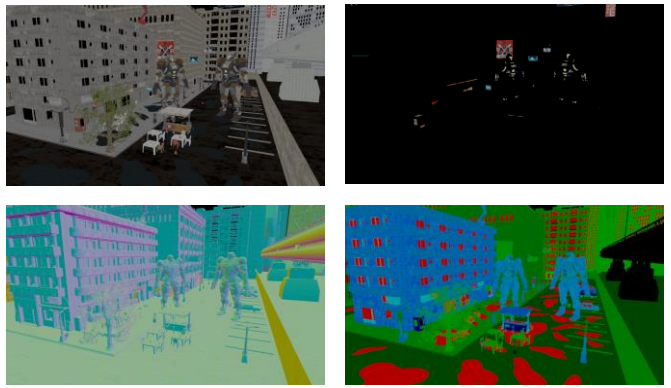
- VK_KHR_acceleration_structure
  - Optimised data structure
  - Bottom Level AS (BLAS)
  - Top Level AS (TLAS)
- Two options
  - VK_KHR_ray_query
    - Existing shaders
  - VK_KHR_ray_tracing_pipeline
    - Driver managed
- VK_EXT_descriptor_indexing
  - Bindless resources
  - Access the material of intercepted hit
  - Core Vulkan, not part of ray tracing



Ray tracing demo

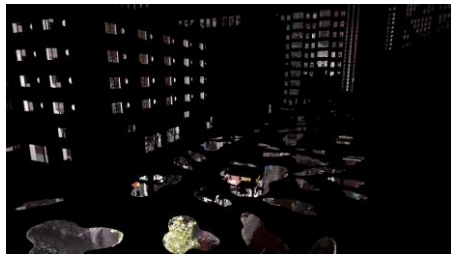
- Use of PBR rendering
  - Easy integration with ray tracing
- Hybrid rendering:
  - Rasterized Deferred GBuffer
  - Ray tracing effects as post-processing (1 render pass per effect)
- Shadows (hard vs soft)
- Reflections (glossy and mirror)
- Refractions

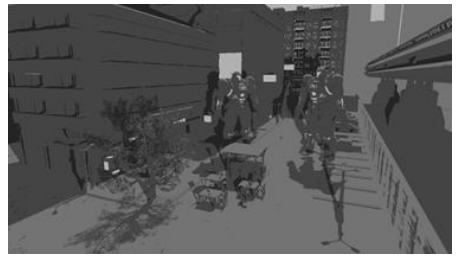
GBuffer



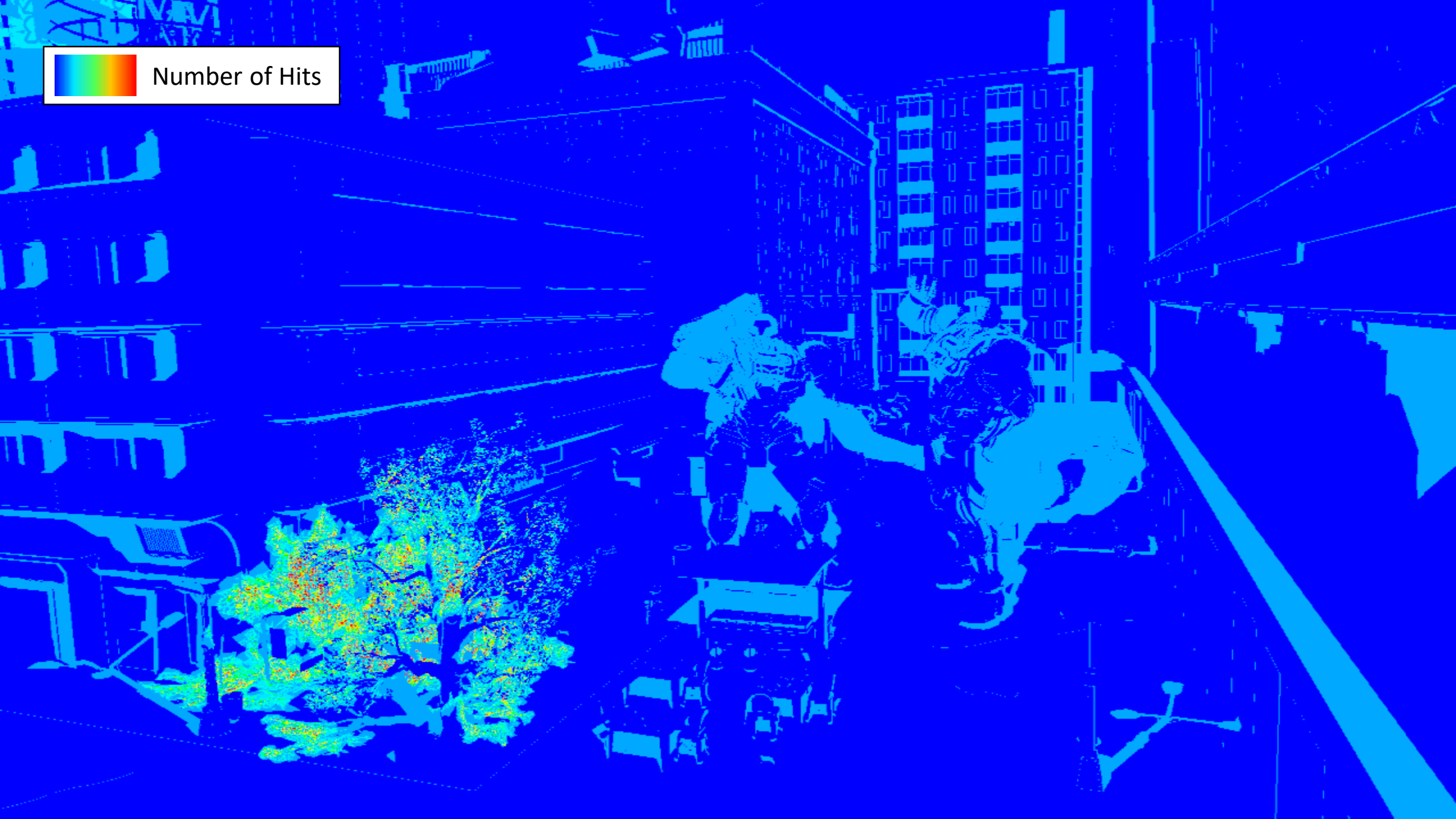
Lighting

RT Effects

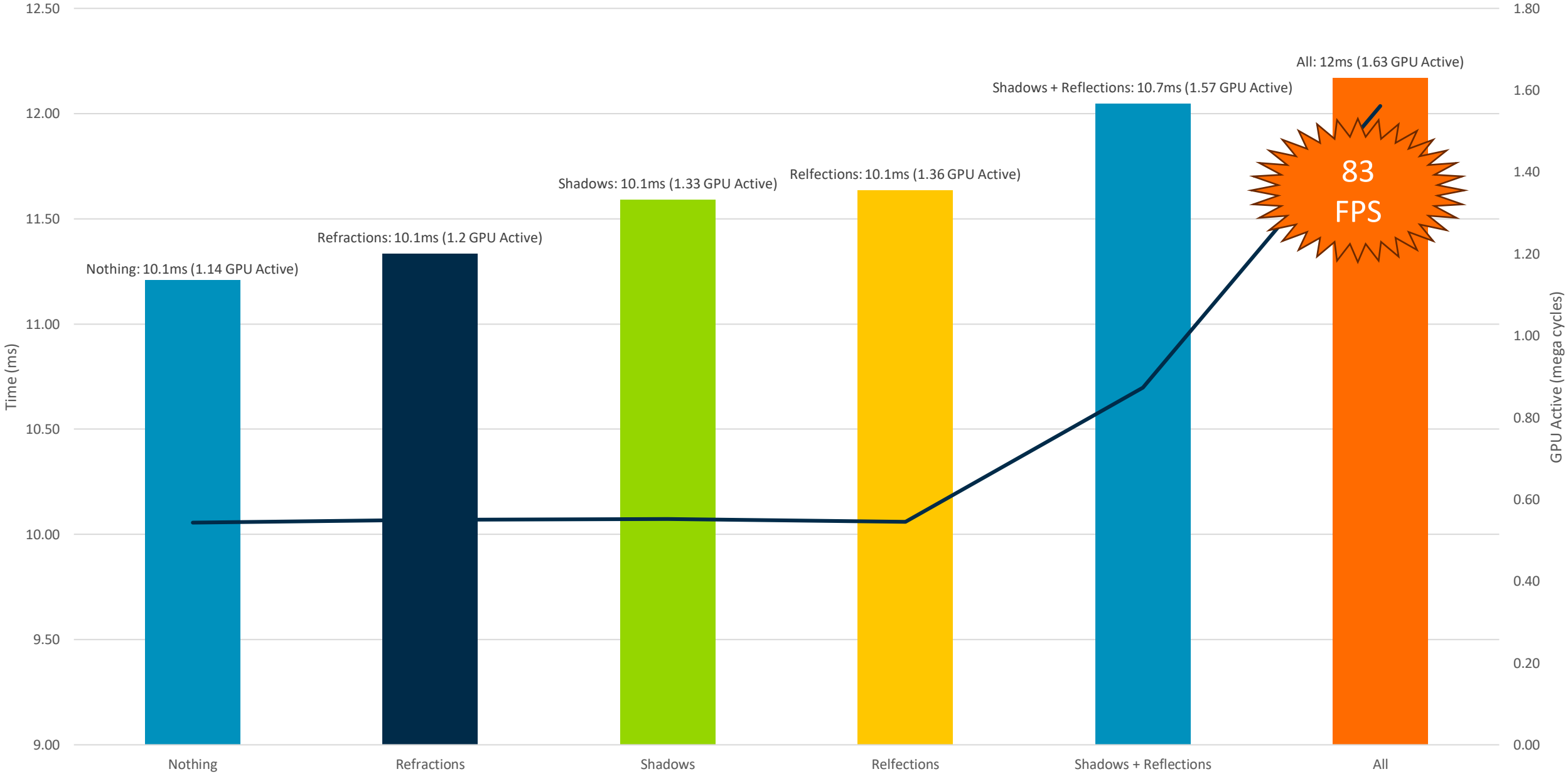




Composition


Number of Hits

Number of objects	1102
Number of meshes	517
Number of triangles	1.5M
Number of triangles without instancing	1.65M
Number of vertices	762K
Number of vertices without instancing	1.07M
Number of materials	190
Number of Lights	16
Number of casting shadow lights	1 punctual
Resolution	1600x900
Reflections resolution	1600x900
Shadows resolution	800x600


12.50
12.00
11.50
11.00
10.50
10.00
9.50
9.00
Time (ms)
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
GPU Active (mega cycles)
Nothing: 10.1ms (1.14 GPU Active)
Refractions: 10.1ms (1.2 GPU Active)
Shadows: 10.1ms (1.33 GPU Active)
Relfections: 10.1ms (1.36 GPU Active)
Shadows + Reflections: 10.7ms (1.57 GPU Active)
All: 12ms (1.63 GPU Active)
83
FPS
Nothing
Refractions
Shadows
Relfections
Shadows + Reflections
All

Advanced Ray Tracing

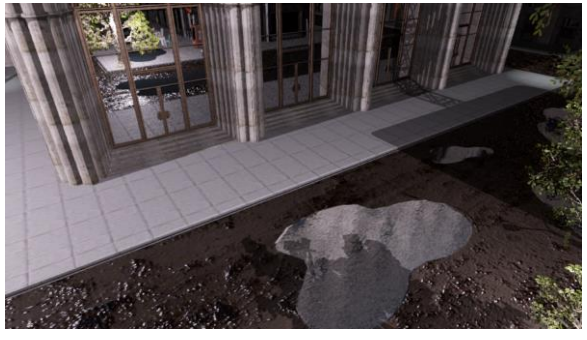
Ray Query Shadows - Alpha test OFF

Ray Query Shadows - Alpha test ON

Alpha test

- Access to bindless materials
  - Extra memory bandwidth
  - Only use transparencies necessary
- RayQueryProceedEXT will return false once it has a confirmed hit
  - Non opaque materials need confirmation to be considered
  - The first hit candidate might not be the closest
- RT pipeline invokes Any-hit shader
- For opaque materials set:
  - GEOMETRY_OPAQUE_BIT_KHR (low priority)
  - GEOMETRY_INSTANCE_FORCE_OPAQUE_BIT_KHR
  - gl_RayFlagsOpaqueEXT (priority, all materials)

```
rayQueryInitializeEXT(…);
#if defined(USE_RQ_ALPHA_TEST)
void main() {
  // Retrieve material ID, barycentrics and
  // sample bindless alpha texture
  const bool opaqueHit = bindless_alpha_test(rayQuery);
  if (!opaqueHit) {
    ignoreIntersectionEXT;
  }
}
#else
rayQueryProceedEXT(rayQuery);
#endif
```



Skinned animations

- Update the BLAS
  - Expensive
  - Minimize number of skinned meshes
- Distribute update across multiple frames
- VkBuildAccelerationStructureFlagBitsKHR
  - FAST_TRACE Only static geometry
  - FAST_BUILD Dynamic geometry
- vkCmdBuildAccelerationStructuresKHR
  - Build a new AS is slower
  - Try update/refit an existing one

Hybrid Ray tracing

Reflections

- Try to minimize number of rays

Retrieve GBuffer info

Roughness for number of rays

Generate ray info

PBR data

Trace ray

RQ/RT Pipeline API

Hit

Fail

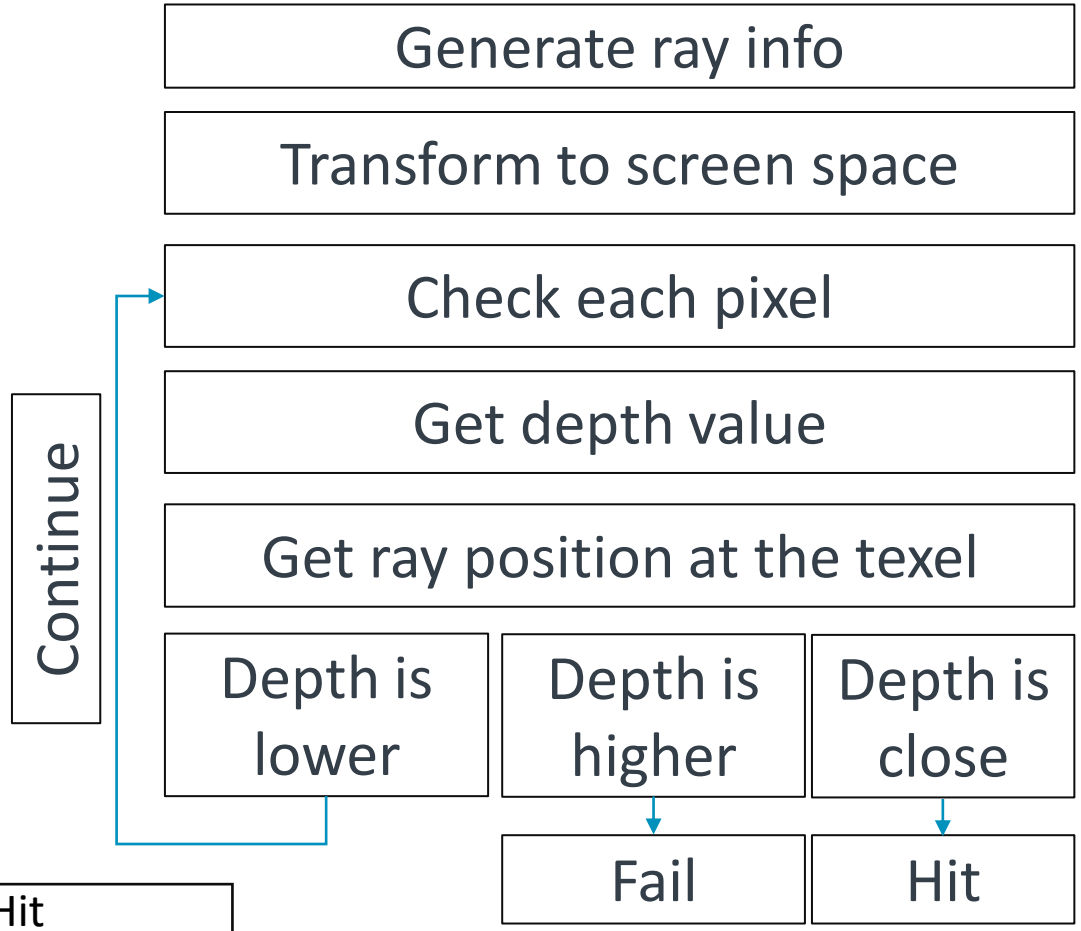
Bindless material

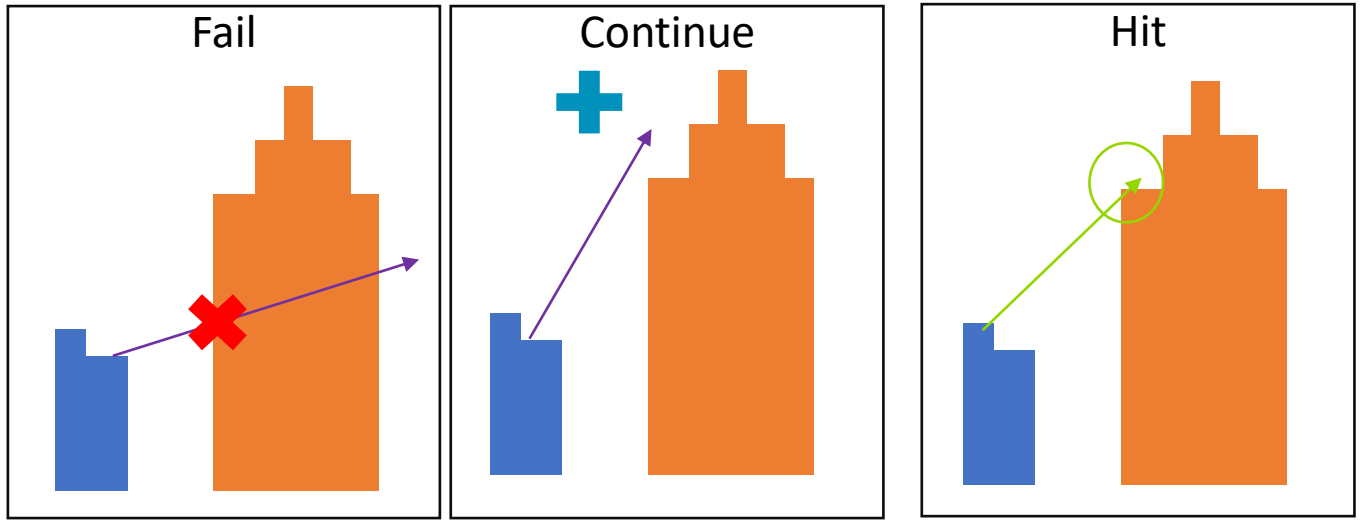
Environment map/ skybox

Multiple Bounces

Screen Space Reflections

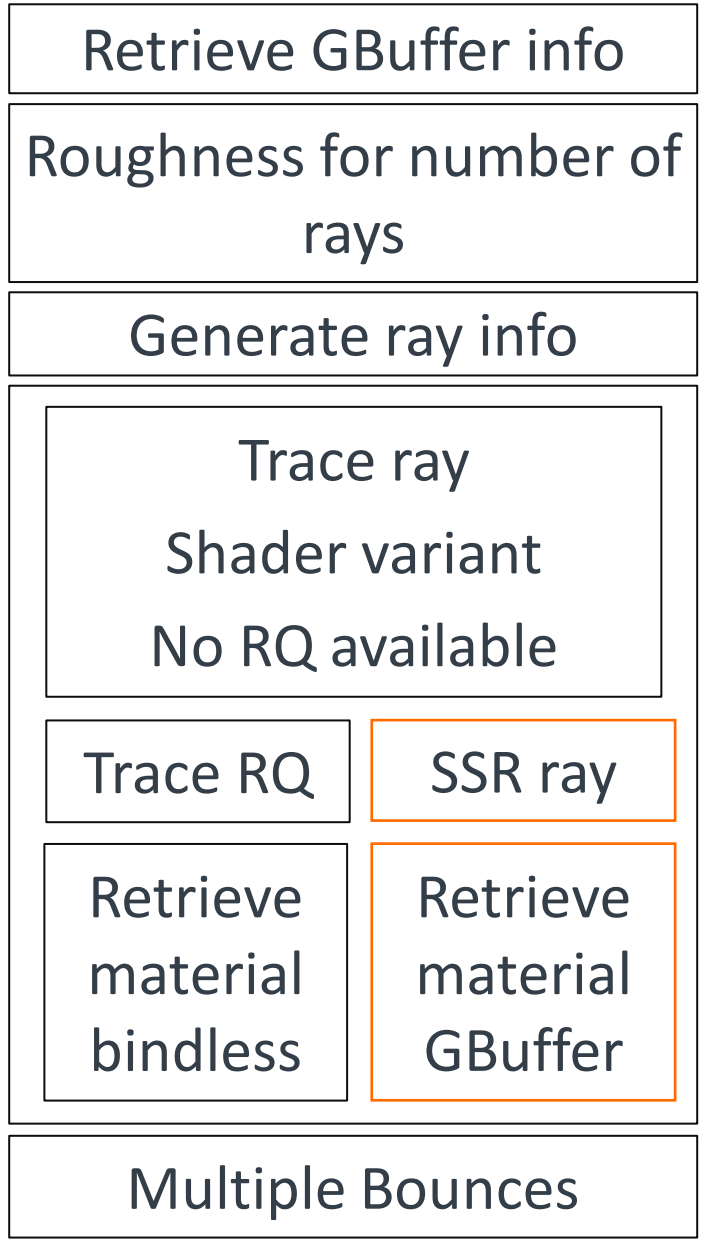
- Implementation
  - We use a stochastic/physically based approach
  - Transform ray origin and direction to screen space
  - Use the depth buffer to check each pixel
  - Advance ray and check each pixel
  - Compare ray position and depth buffer
    - Values are close – Hit
    - Depth is lower – Continue
    - Depth is higher – Not enough information (fail)





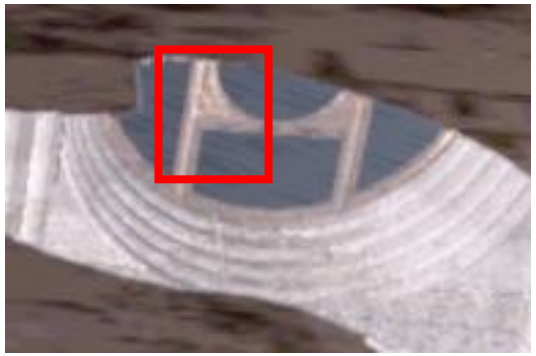
Screen Space Reflections

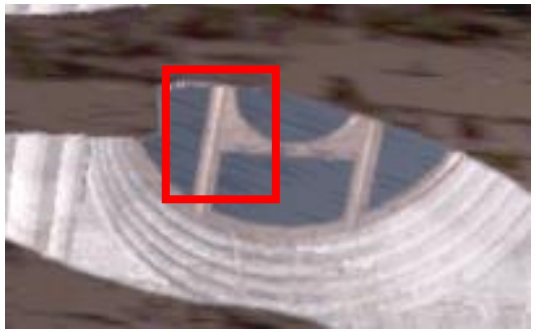
- Can be integrated easily in PBR
  - Fallback for RT reflections
- Compared to RT reflections
  - Better performance
  - Lower quality - Only objects on screen
- More difficult (more magic numbers)
- Consider Hierarchical SSR
  - Mipmap with maximum value of depth-buffer
  - Allows to skip a lot of empty space

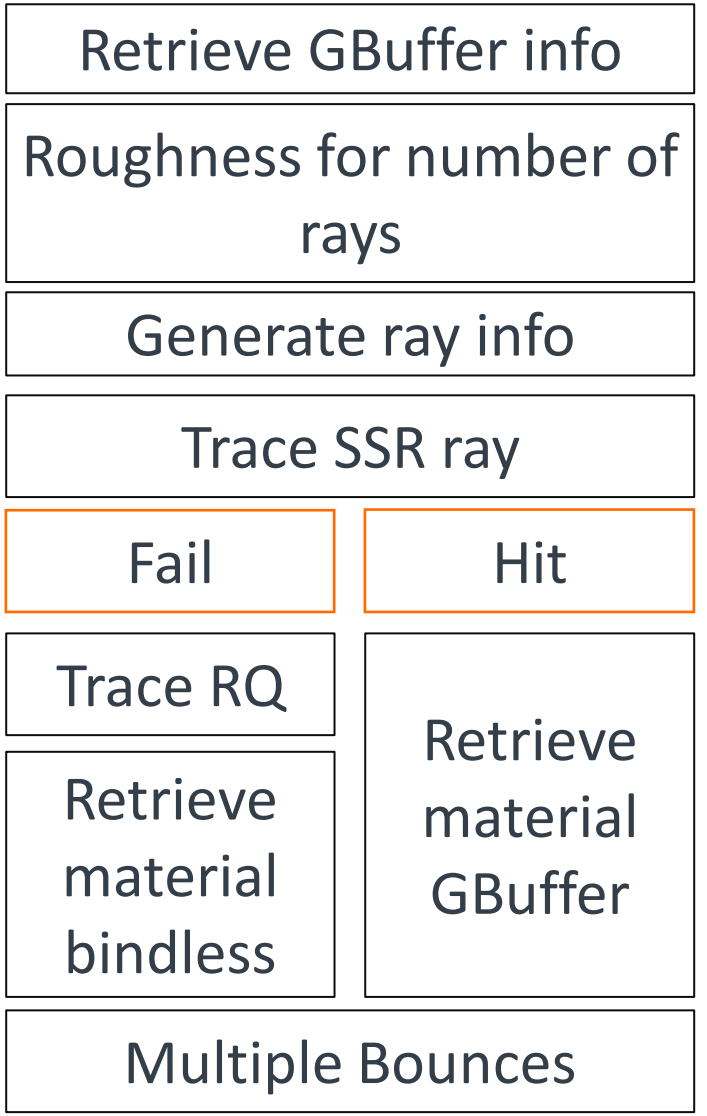


Hybrid reflections (SSR+RQ)

- We can retrieve SSR hit material from G-buffer
  - Avoid bindless
- Different view direction
  - Don’t reuse illuminated result for SSR
- Terminate (as fail) if occluded
- Mali offline compiler
  - Ensure data is reused and not copied
  - Easy to create a lot of spilling





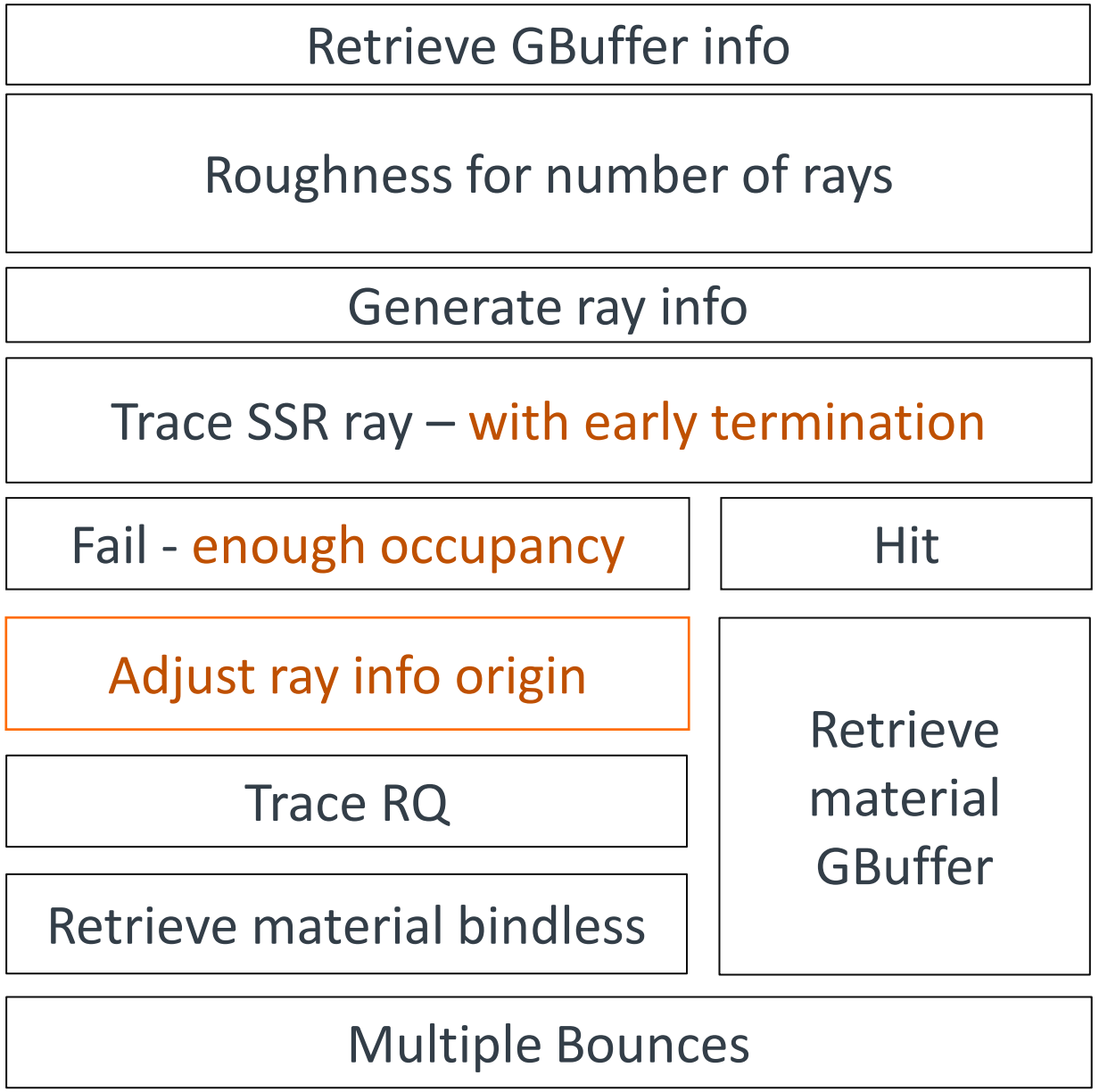


Hybrid reflections (SSR+RQ) improvements

- Self intersection
  - Common problem when using G-buffer to launch rays
- Solutions
  - gl_RayFlagsCullFrontFacingTrianglesEXT
    - Extra GPU work
  - Modify ray - Increase t_min or add bias to ray origin
  - Use a Screen space ray
- Hybrid reflections always launch a SSR ray
  - We know that we cannot intersect anything until the ray fails
  - Relaunch a ray query ray from the last texel visited by SSR
  - Transform from screen space to world space
- Useful in model with transparencies or high geometry
  - Vegetation, hair, etc
  - Allows to skip the close objects

Hybrid reflections (SSR+RQ)

- Glossy reflections
  - Roughness to control direction
  - Warp divergence in ray directions
- One single ray doing can slow down the entire warp
  - Less relevant in mirror reflections
- We can use subgroup operation to control occupancy
  - WaveActiveCount(true)
  - subgroupBallotBitCount(subgroupBallot(true))
- Check during SSR iterations to see occupancy
- Check occupancy before launching a RQ ray



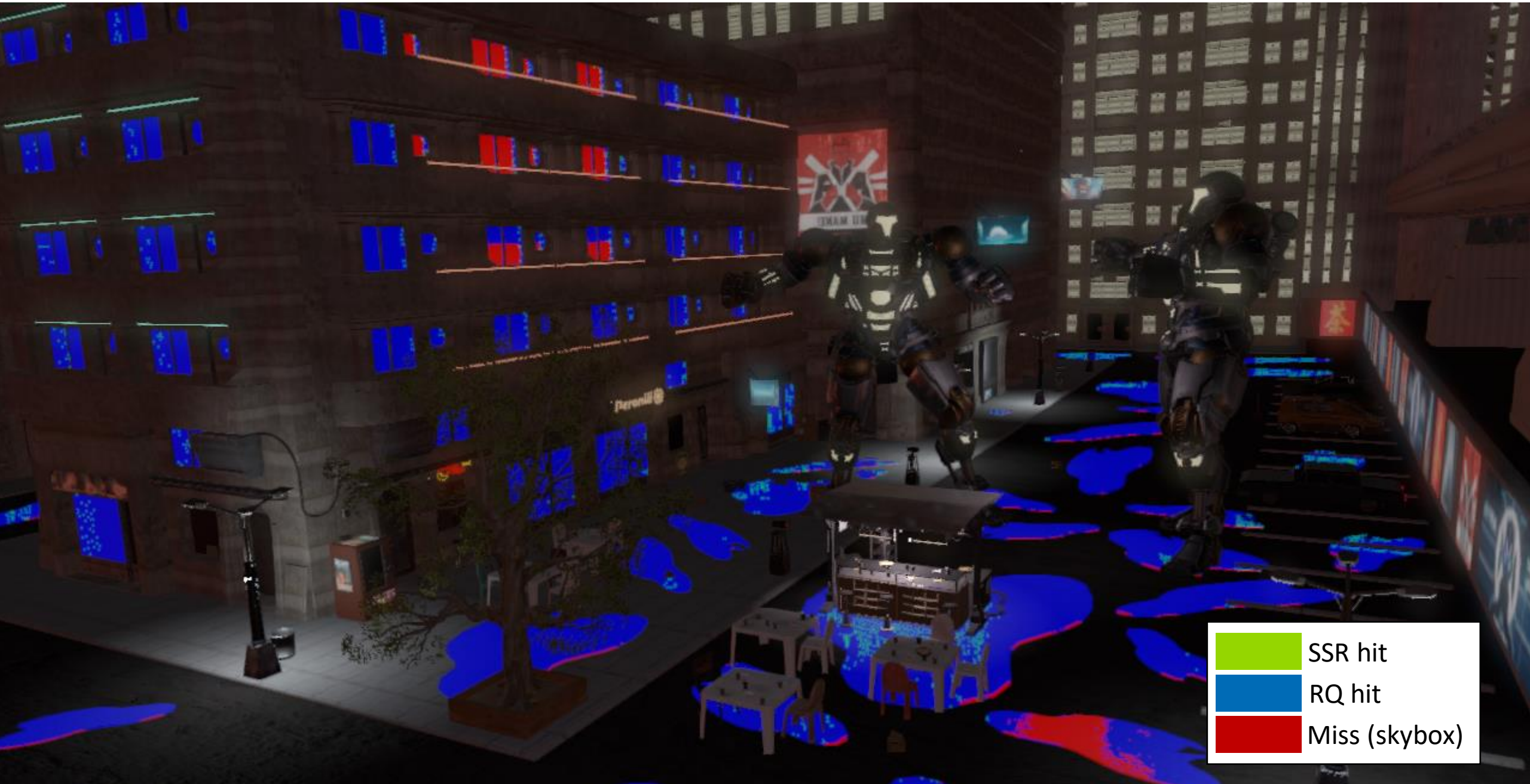
No reflections

Screen Space Reflections (SSR)

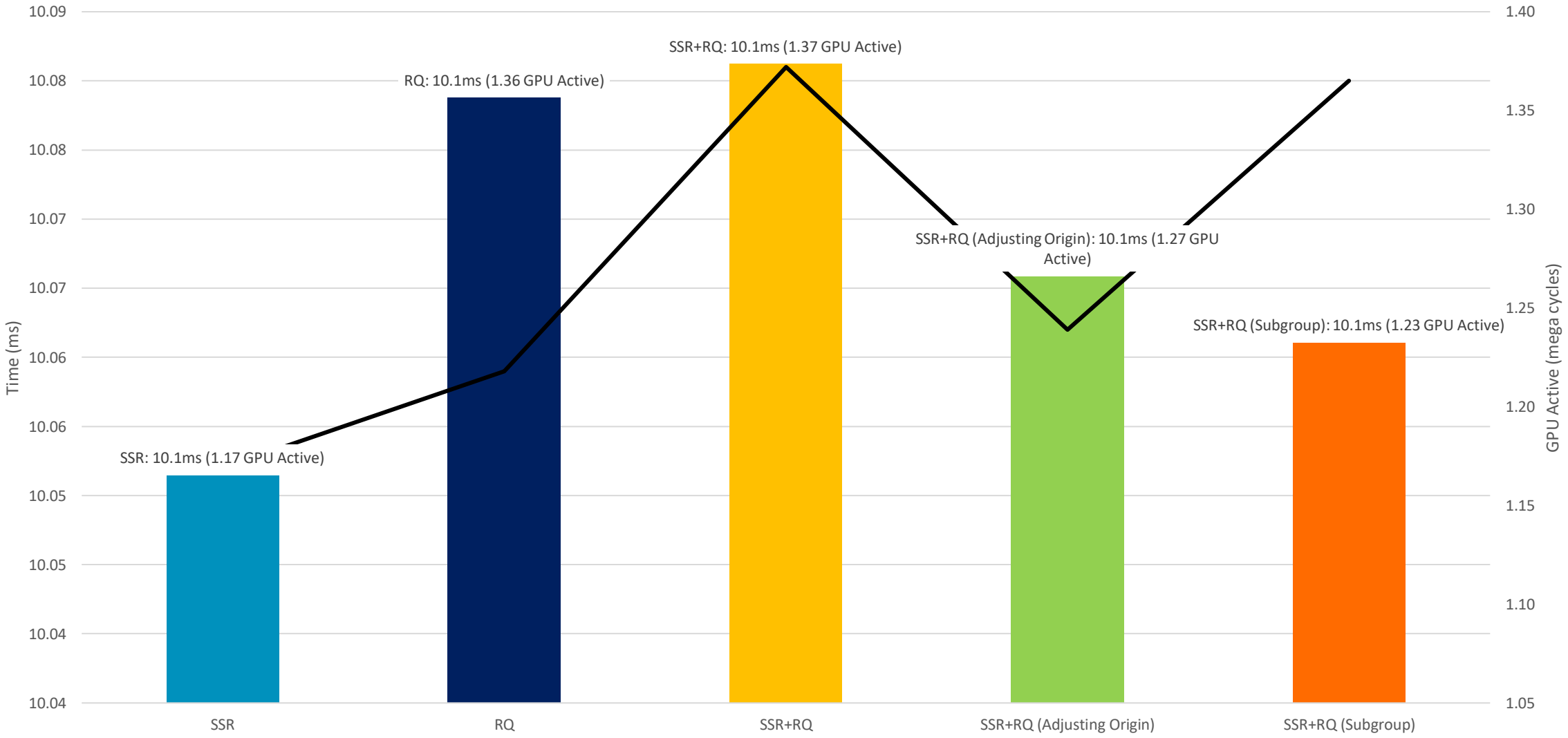
RQ reflections

Hybrid (RQ+SSR) reflections

Hybrid (RQ+SSR) reflections



Hybrid Reflections



Shadows

- Avoid unnecessary rays if the light is behind
- We do not need the closest hit - gl_RayFlagsTerminateOnFirstHitEXT

Retrieve GBuffer info

Generate ray info

PBR data

Behind the light

In shadow – no ray

```
if (dot(N, light_direction) <= 0) {
    visibility *= MAX_SHADOW_STRENGTH;
    continue;
}
```

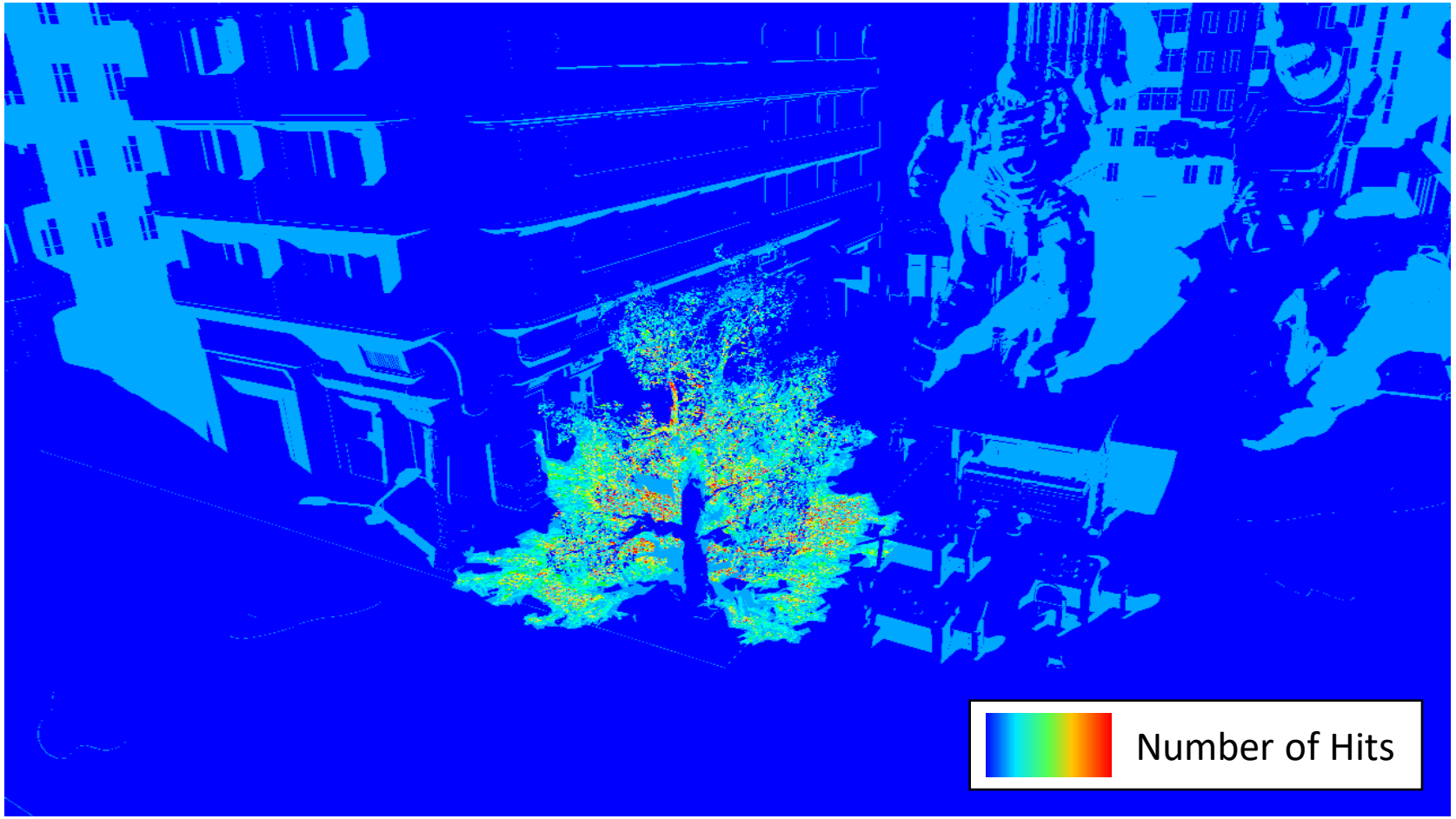
Trace ray

RQ/RTPipeline API

Hit	Fail
In shadow	Illuminated

Hybrid shadows

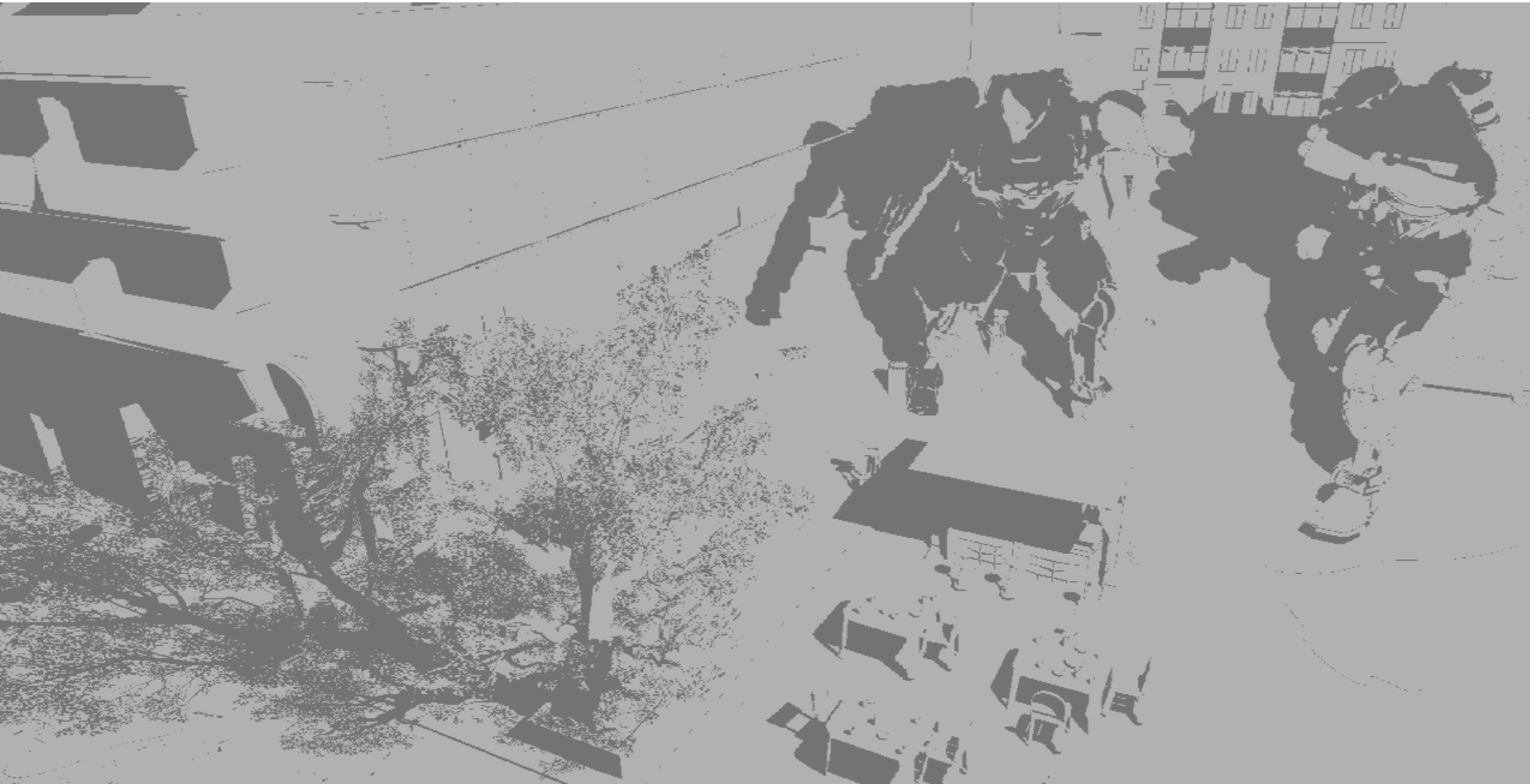
- Hybrid reflections are faster to traverse
- Shadows follow a similar algorithm
- Can the same idea be applied to shadows



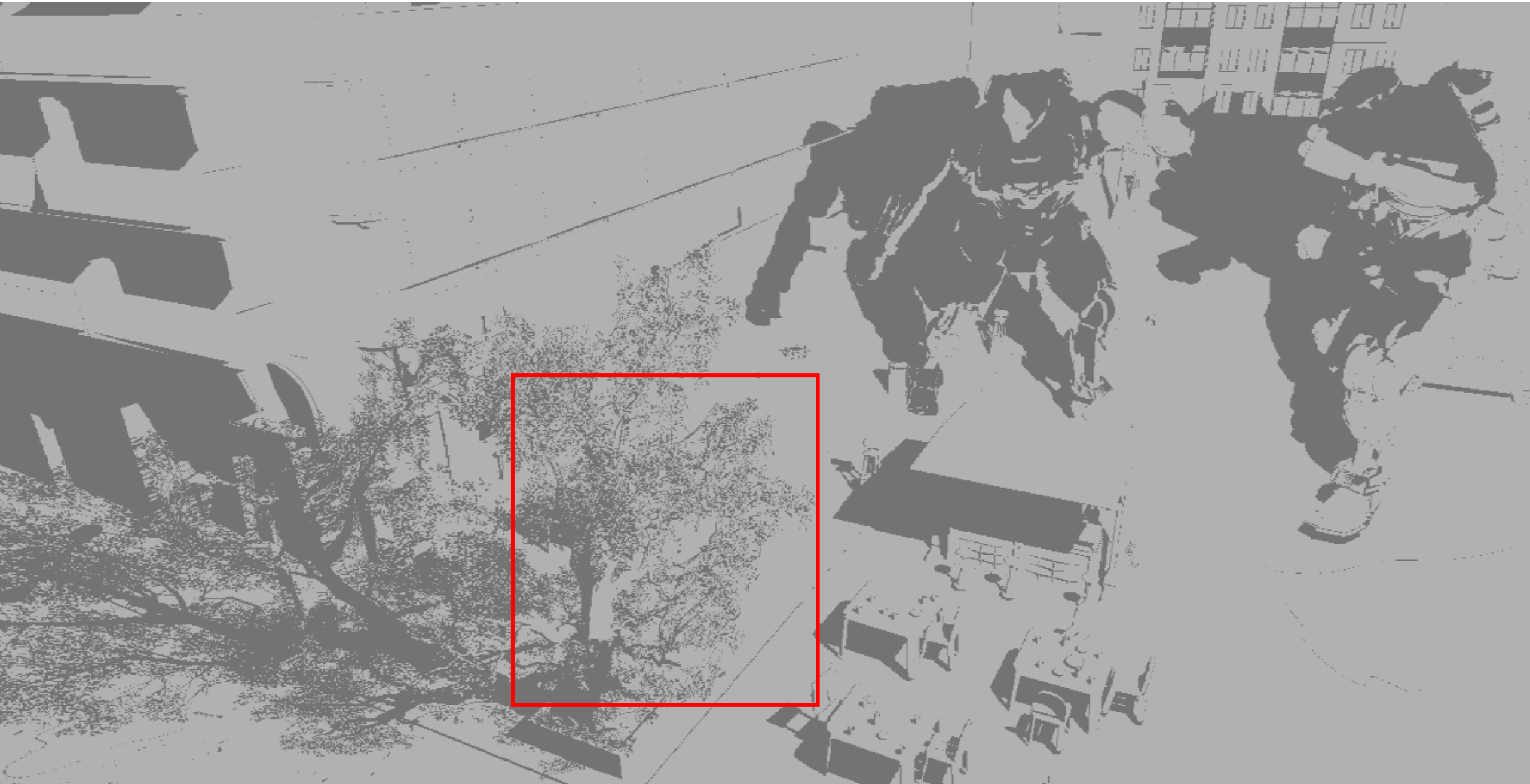
Facing away from the light

Screen Space Shadows (SSS)

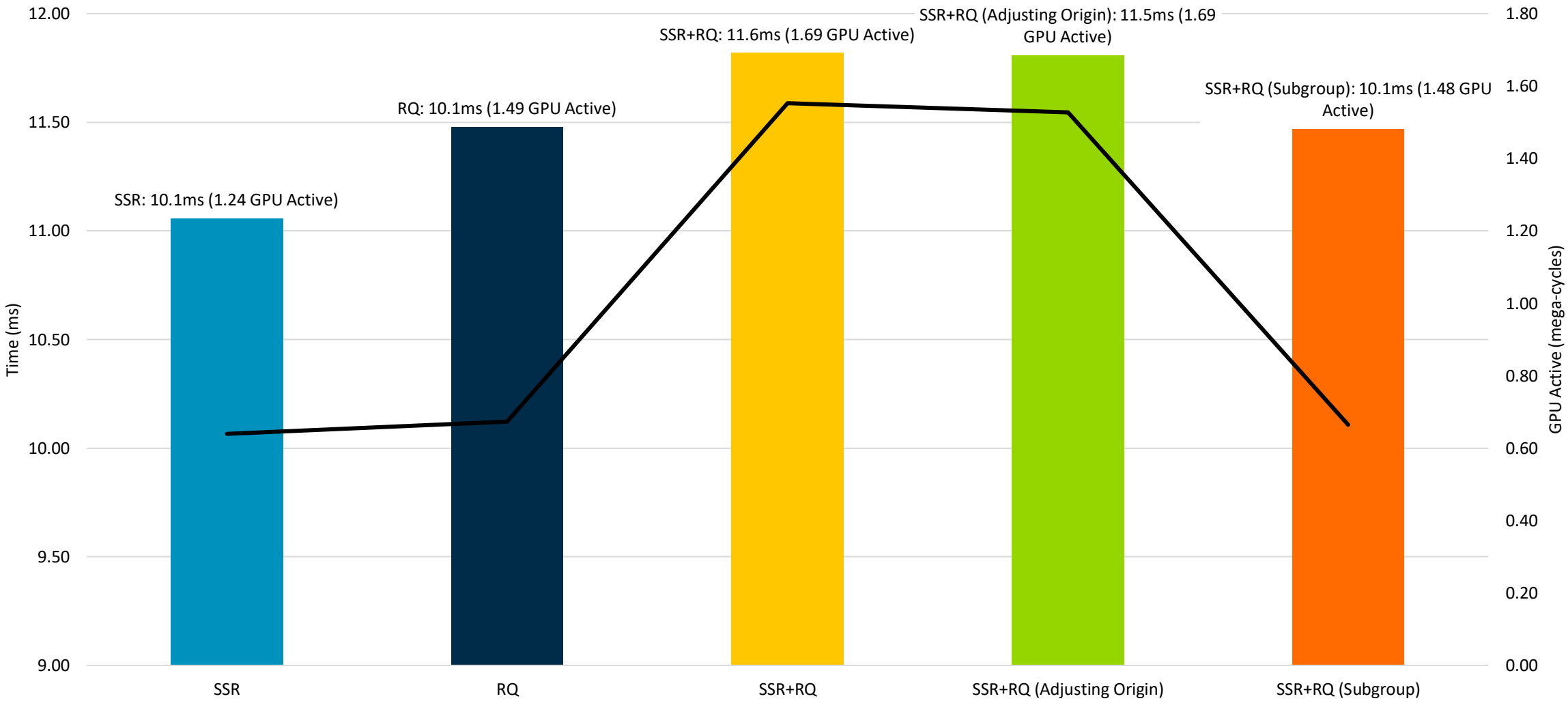
RQ Shadows



Hybrid (SSS+RQ) Shadows



Hybrid shadows



arm

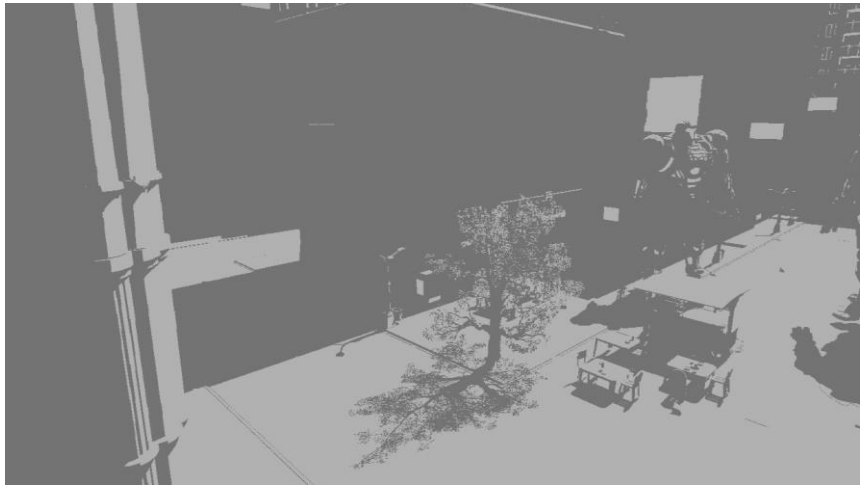
VRS + RT

Variable Rate Shading (VRS)

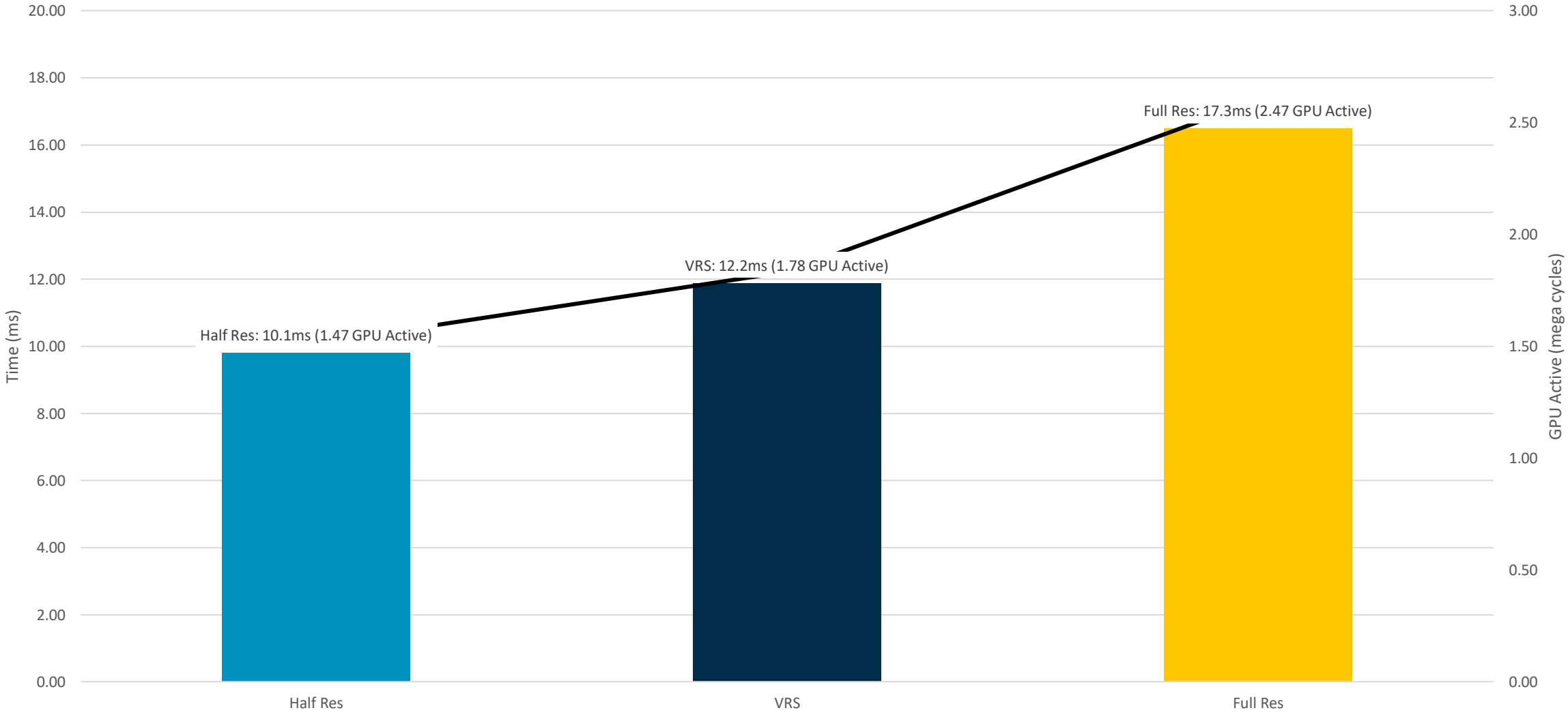
- VK_KHR_fragment_shading_rate extension
- Render some elements at lower resolution
  - Concentrate Fragment density were is noticeable
- Multiple modes
  - Per draw call
  - Per primitive
  - Shading rate attachment (most flexible)
- Option to combine multiple methods

VRS + RT

- We can combine VRS and RT
- Screen-space shading attachment
  - Generated by a compute pass
- Choose a heuristic to lower rate in less noticeable areas
- IE: Luminance to have lower rate in darker areas
  - Detail is less noticeable in darker areas
  - Useful for reflections
- IE: Concentrate in edge of shadows
  - Use previous frame
  - Light shadowed areas should remain relatively constant
  - Concentrate shading rate in borders
  - Big enough margins avoids flickering



Ray Query shadows



Conclusion

- Ray tracing can provide a huge increase of visual quality to games
- Ray tracing makes it easier to implement complex realistic effects
- Devices in the market are capable to run ray tracing content
- Vulkan uses a flexible API for RT
  - Lots of space for optimization for your application
  - Same API for mobile and desktop – easy to port your content

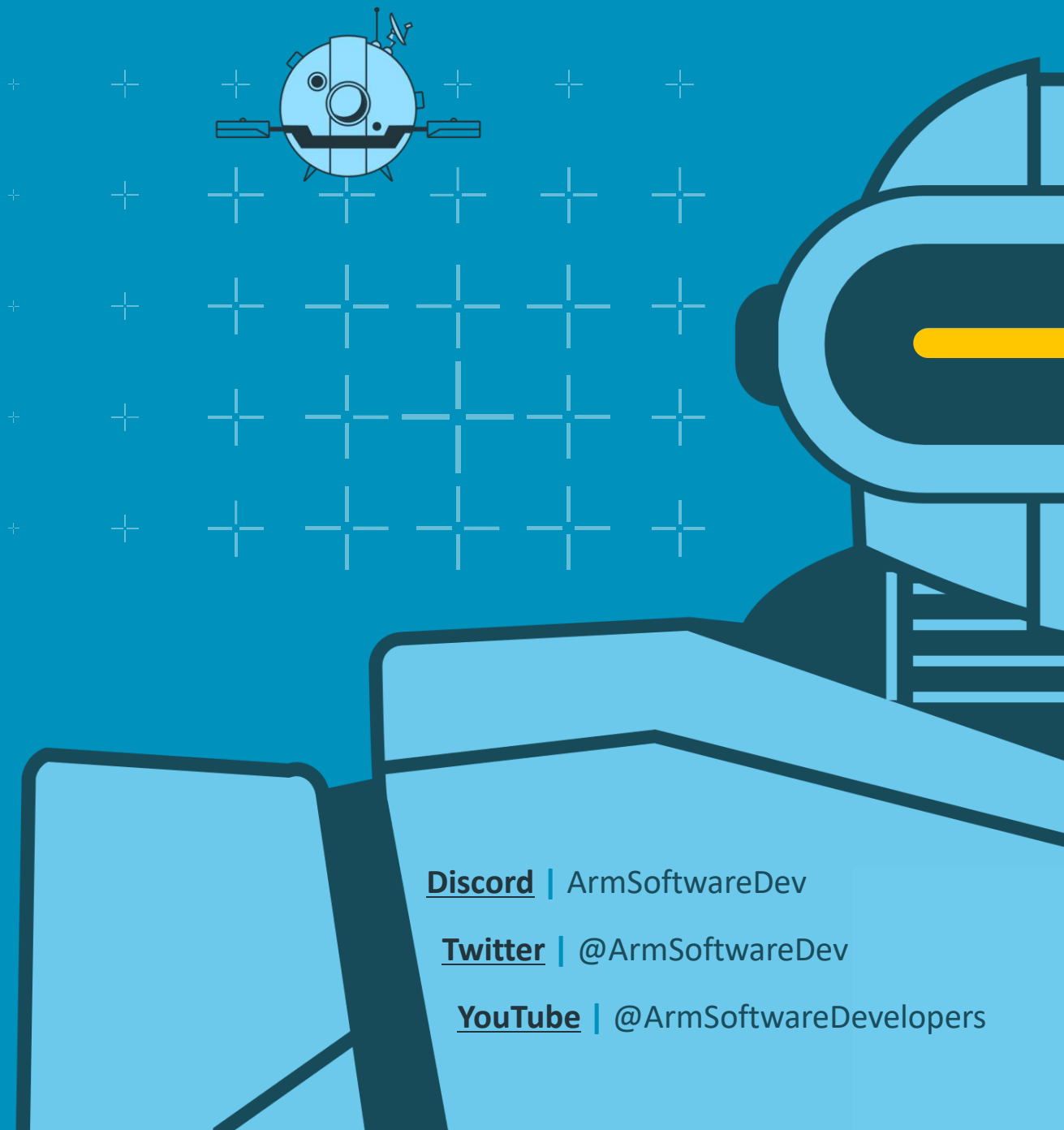
arm Developer Program

Build the future on Arm

- Join our developer community
- Connect with like-minded developers
- Get fresh insights from Arm experts
- **Sign up:** arm.com/developerprogram

Find out more

- **Talk to us via Discord** ArmSoftwareDev
- **Docs & Resources:** arm.com/vulkan
- **Forums and blogs:** community.arm.com



Discord | ArmSoftwareDev

Twitter | @ArmSoftwareDev

YouTube | @ArmSoftwareDevelopers

arm

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה