Vulkanised 2026

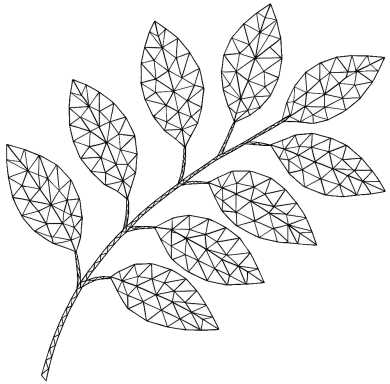
The 8th Vulkan Developer Conference
San Diego, USA | February 9-11, 2026

Accelerating Transparency: Vulkan's Opacity Micromaps

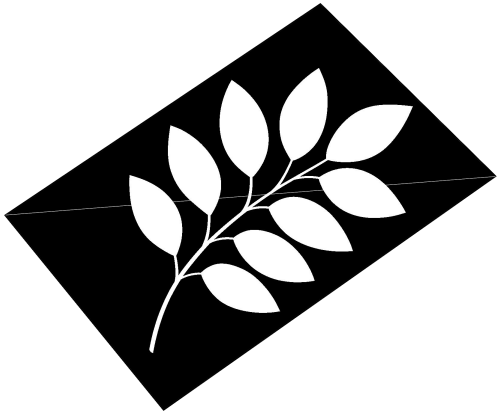
Matthew Netsch, Qualcomm Technologies, Inc

Motivation

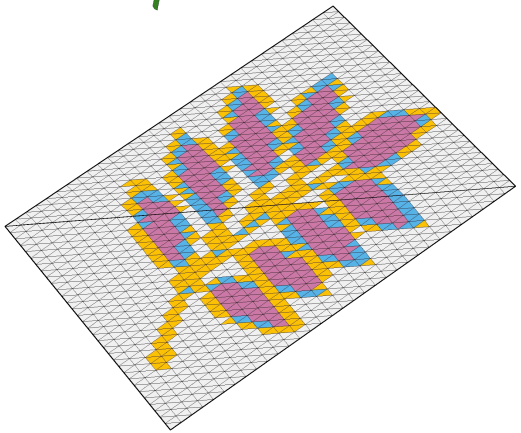
- **Many objects in scenes have complex geometry**
  - Foliage - grass, leaves
  - Buildings – fences, gills, windows
- **Traditional RT methods are quite expensive**



Complex Geometry
Excessive BVH memory cost



Alpha map
Significant AnyHit shader runtime cost



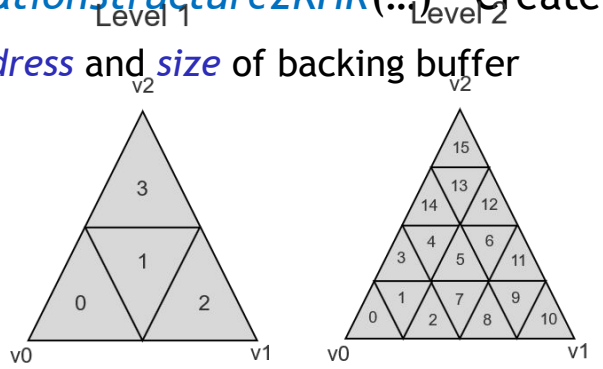
Opacity Micromap
Skips AnyHit shader invocations
More compact than full model

First look at VK_KHR_opacity_micromap

- **Not published yet, may change before release!**
- **Promotion of VK_EXT_opacity_micromap with significant API changes**
  - VkMicromapEXT object is not promoted
  - Host commands deprecated
  - Micromaps now disabled with ray query by default
  - Lossy micromaps added
  - Serialization improvements
  - …and more!
- **Micromap encoding and ray traversal effectively unchanged**
- **Slides reference ray pipeline shaders for simplicity**

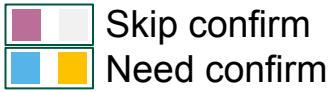
Creating micromaps

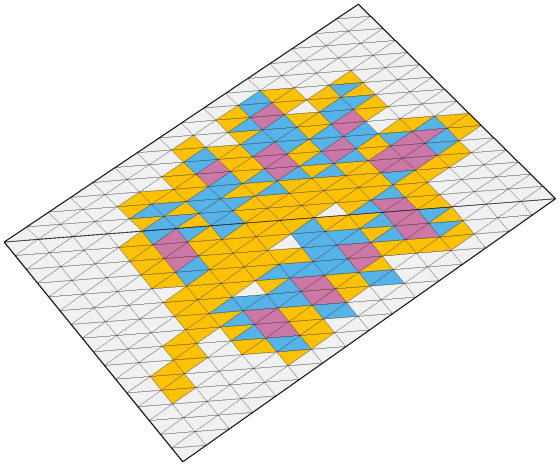
- **Micromaps are created as acceleration structures**
  - *vkGetAccelerationStructureBuildSizesKHR*(…) - Gets the size of micromap
    - Type VK_ACCELERATION_STRUCTURE_TYPE_OPACITY_MICROMAP_KHR (or GENERIC)
    - New geometry type *VkAccelerationStructureGeometryMicromapDataKHR*
      - Consumes **Shape**: *pUsageCounts* – [{*count*, *subdivisonLevel*, *format*}]
  - Create backing buffer and acquire device address
  - *vkCreateAccelerationStructure2KHR*(…) - Creates object
    - Consumes *address* and *size* of backing buffer



Micromap shape

- **Shape tradeoffs**



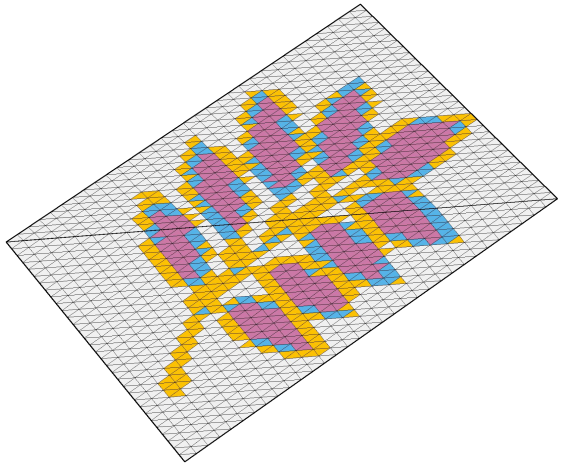


subdivisionLevel = 4

✓ Fewer memory usage

✓ May see faster traversal

✗ More AnyHit shader invocations



subdivisionLevel = 5

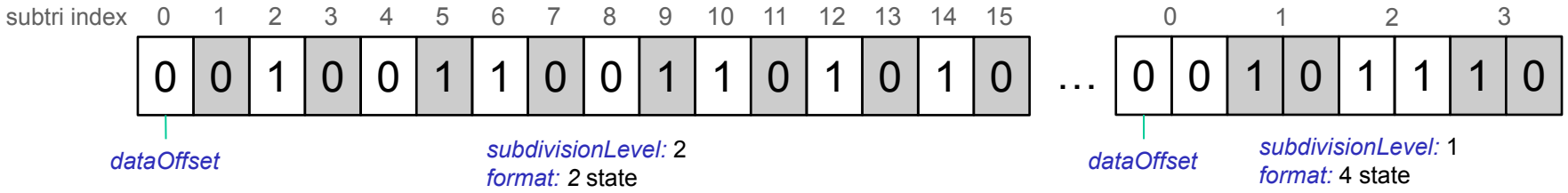
✗ More memory usage

✗ May see slower traversal

✓ Less AnyHit shader invocations

Building micromaps

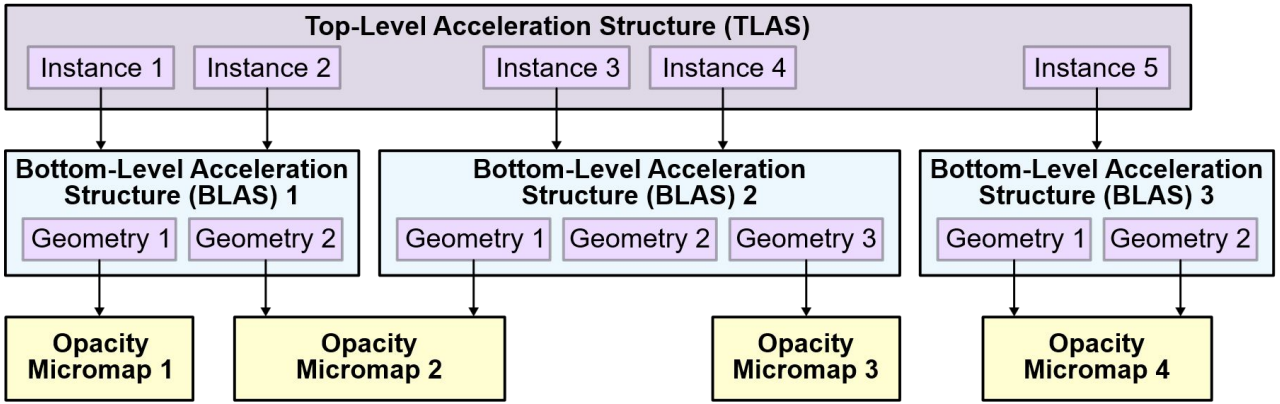
- *vkCmdBuildAccelerationStructuresKHR* (…) – Encodes opacity information to OMM
  - Type VK_ACCELERATION_STRUCTURE_TYPE_OPACITY_MICROMAP_KHR
  - Consumes **Data** in addition to **Shape:**
    - *data* - *encoded* opacity data
    - *triangleArray* - layout of data
  - Indirect/host builds not permitted

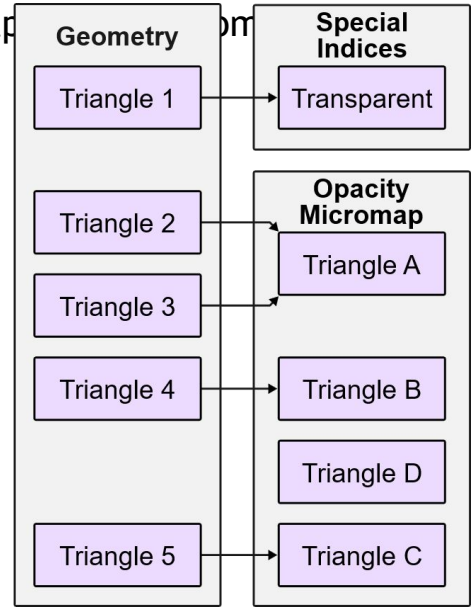


data Bitstream (little-endian bytes)

Binding micromaps

- *VkAccelerationStructureTrianglesOpacityMicromapKHR* – Binds OMM during BLAS build
  - Opacity micromaps are referenced by a BLAS
  - Triangles in the BLAS geometry are mapped to micromap triangles 1:1
  - *indexBuffer* can be used to provide n:1 mapping
  - Special indices can assign state to the whole triangle without map
  - No longer consumes **Shape** (*pUsageCounts*)



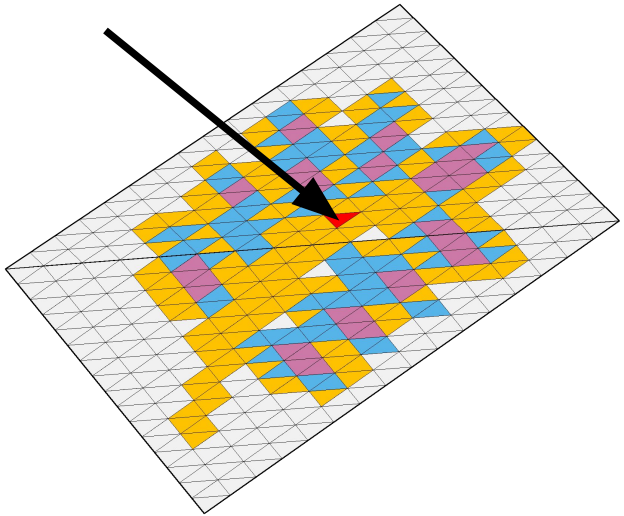


Copying Micromaps

- *vkCmdWriteAccelerationStructuresPropertiesKHR*(…) – Gets dst sizes for copy
- *vkCmdCopyAccelerationStructureKHR*(…) – Copies micromaps
  - **Clone** – identical copy
  - **Compact** – compacts micromap into more space efficient copy
- *vkCmdCopyAccelerationStructureToMemoryKHR*(…) – Serializes micromap
- *vkGetDeviceAccelerationStructureCompatibilityKHR*(…)
  - Verifies if the serialized micromap is compatible with the device
- *vkCmdCopyMemoryToAccelerationStructureKHR*(…) – Deserializes micromap
- *VK_ACCELERATION_STRUCTURE_SERIALIZED_BLOCK_TYPE_OPACITY_MICROMAP_KHR*
  - New encoded serialized header for BLAS referencing micromaps
  - Before deserializing BLAS, referenced micromap device addresses must be filled in by app

Fetching micromaps

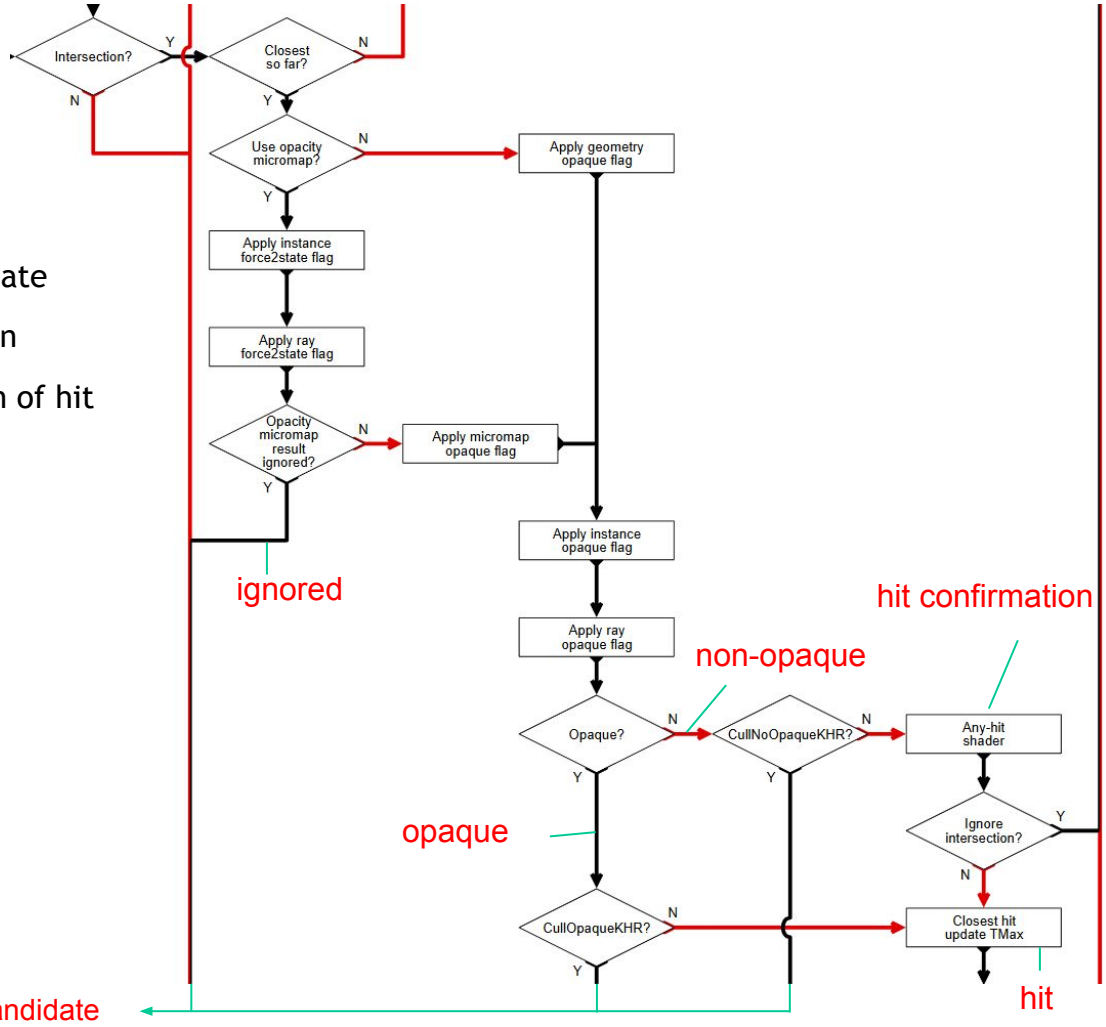
- Opacity state of intersected sub-triangle is fetched:
  - **Fully transparent** – candidate is *ignored*
  - **Fully opaque** – candidate is *opaque*
  - **Unknown** – candidate is *non-opaque* (4 state only)



4 State value	2 State value	Special index value	Result
0	0	VK_OPACITY_MICROMAP_SPECIAL_INDEX_FULLY_TRANSPARENT_KHR	Ignored
1	1	VK_OPACITY_MICROMAP_SPECIAL_INDEX_FULLY_OPAQUE_KHR	Opaque
2		VK_OPACITY_MICROMAP_SPECIAL_INDEX_FULLY_UNKNOWN_TRANSPARENT_KHR	Non-opaque
3		VK_OPACITY_MICROMAP_SPECIAL_INDEX_FULLY_UNKNOWN_OPAQUE_KHR	Non-opaque

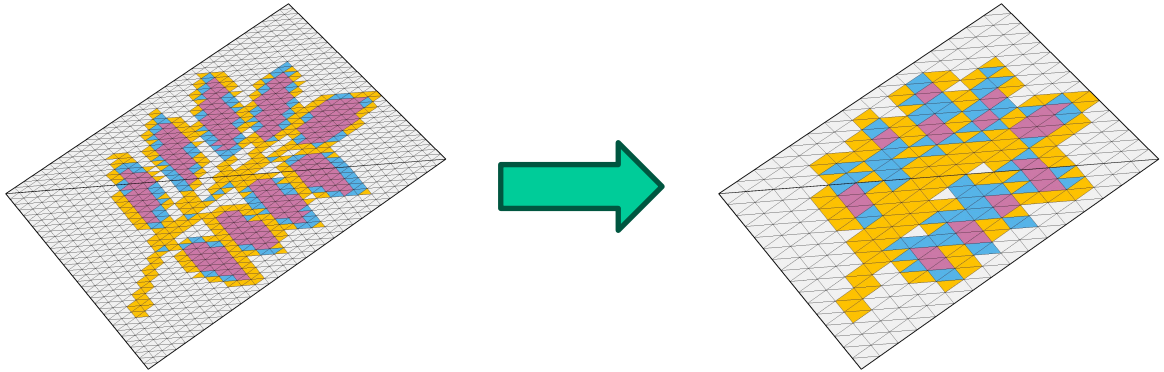
Ray Traversal

- Result of fetch:
  - ***Ignored*** – stop processing candidate
  - ***Opaque*** – automatic confirmation
  - ***Non-opaque*** – need confirmation of hit



Lossy Micromaps

- **Allows invariant substitution into unknown state**
  - Allows for lossy compression encoding internally
  - Additionally allows drivers to facilitate assisted emulation of larger subdivision levels



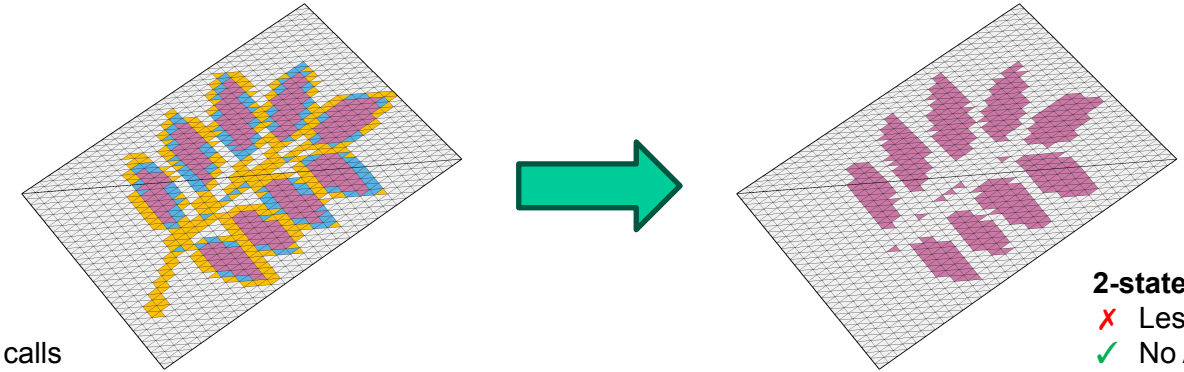
App build data
Level 5

Built micromap
Level 4*

Ex. driver which exposes:
- maxOpacity4StateSubdivisionLevel == 4
- maxOpacityLossy4StateSubdivisionLevel >= 5

force2state

- **Reduces 4 state value to 2 state**
  - 2 state formats may be useful for some cases – e.g. indirect lighting
  - Flag allows apps to reuse their 4 state micromaps without encoding a separate micromap



4-state
✓ Better quality
✗ AnyHit shader calls

2-state
✗ Less quality
✓ No AnyHit shader calls

4 State value	2 State value	Special index value	Result
0	0	VK_OPACITY_MICROMAP_SPECIAL_INDEX_FULLY_TRANSPARENT_KHR	Ignored
1	1	VK_OPACITY_MICROMAP_SPECIAL_INDEX_FULLY_OPAQUE_KHR	Opaque
2		VK_OPACITY_MICROMAP_SPECIAL_INDEX_FULLY_UNKNOWN_TRANSPARENT_KHR	Ignored
3		VK_OPACITY_MICROMAP_SPECIAL_INDEX_FULLY_UNKNOWN_OPAQUE_KHR	Opaque

Questions?

- Links (live when published):
  - https://docs.vulkan.org/features/latest/features/proposals/VK_KHR_opacity_micromap.html
  - https://docs.vulkan.org/refpages/latest/refpages/source/VK_KHR_opacity_micromap.html
  - https://github.khronos.org/SPIRV-Registry/extensions/KHR/SPV_KHR_opacity_micromap.html
  - (glsl coming too)