DEVELOPER DAY

DEPTH STENCIL RESOLVE

LEWIS GORDON, SAMSUNG ELECTRONICS

KHRONOS® GROUP

GDC

GDC 2019

#KhronosDevDay

Introduction

- Who am I?
- Why VK_KHR_depth_stencil_resolve was added
- How to use the extension
- Performance impact
- Availability

Why this extension was added

- Game developer request
- Useful where the depth buffer needs to be used by a later pass, such as depth fade
- Already possible to resolve colour MSAA data in the same pass without writing back the unresolved buffer
- Saves bandwidth on tile-based GPUs (mobile friendly)
- Allows depth stencil buffer from MSAA pass to be resolved in the same pass
- Already possible in OpenGL ES (GL_EXT_multisampled_render_to_texture)

Resolving depth stencil?

- vkCmdBlitImage?
- vkCmdResolveImage?
- Compute shader?
- Renderpass?
- vkDepthStencilResolveKHR

Requirements

- Vulkan was designed with extensibility in mind
- In this case though, it's not quite as straightforward, so another extension is also required
- Depth stencil resolve is dependent on vkCreateRenderPass2KHR
- Not as scary as it sounds as this still creates a VkRenderPass object

Vulkan Render Pass 2

```
typedef struct VkRenderPassCreateInfo2KHR {
    VkStructureType                         sType;
    const void*                             pNext;
    VkRenderPassCreateFlags                 flags;
    uint32_t                                attachmentCount;
    const VkAttachmentDescription2KHR*      pAttachments;
    uint32_t                                subpassCount;
    const VkSubpassDescription2KHR*         pSubpasses;
    uint32_t                                dependencyCount;
    const VkSubpassDependency2KHR*          pDependencies;
    uint32_t                                correlatedViewMaskCount;
    const uint32_t*                         pCorrelatedViewMasks;
} VkRenderPassCreateInfo2KHR;
```

Vulkan subpass description

```
typedef struct VkSubpassDescription {
    VkSubpassDescriptionFlags                   flags;
    VkPipelineBindPoint                         pipelineBindPoint;
    uint32_t                                    inputAttachmentCount;
    const VkAttachmentReference*                pInputAttachments;
    uint32_t                                    colorAttachmentCount;
    const VkAttachmentReference*                pColorAttachments;
    const VkAttachmentReference*                pResolveAttachments;
    const VkAttachmentReference*                pDepthStencilAttachment;
    uint32_t                                    preserveAttachmentCount;
    const uint32_t*                             pPreserveAttachments;
} VkSubpassDescription;
```

Vulkan subpass description 2

```
typedef struct VkSubpassDescription2KHR {
    VkStructureType                    sType;
    const void*                        pNext;       // -> VKSubpassDescriptionDepthStencilResolveKHR
    VkSubpassDescriptionFlags          flags;
    VkPipelineBindPoint                pipelineBindPoint;
    uint32_t                           viewMask;
    uint32_t                           inputAttachmentCount;
    const VkAttachmentReference2KHR*   pInputAttachments;
    uint32_t                           colorAttachmentCount;
    const VkAttachmentReference2KHR*   pColorAttachments;
    const VkAttachmentReference2KHR*   pResolveAttachments;
    const VkAttachmentReference2KHR*   pDepthStencilAttachment;
    uint32_t                           preserveAttachmentCount;
    const uint32_t*                    pPreserveAttachments;
} VkSubpassDescription2KHR;
```

Depth Stencil Resolve

```
typedef struct VkSubpassDescriptionDepthStencilResolveKHR {
    VkStructureType                         sType;
    const void*                             pNext;
    VkResolveModeFlagBitsKHR                depthResolveMode;
    VkResolveModeFlagBitsKHR                stencilResolveMode;
    const VkAttachmentReference2KHR*        pDepthStencilResolveAttachment;
} VkSubpassDescriptionDepthStencilResolveKHR;
```

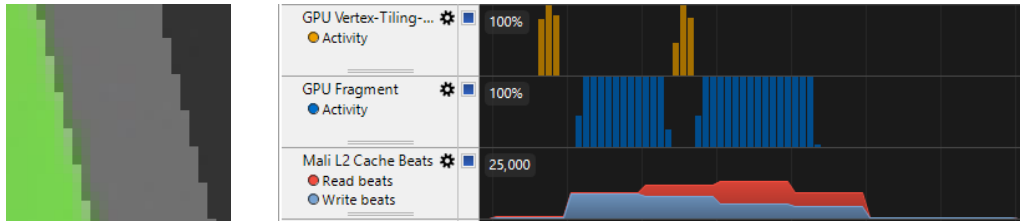
Resolve Mode

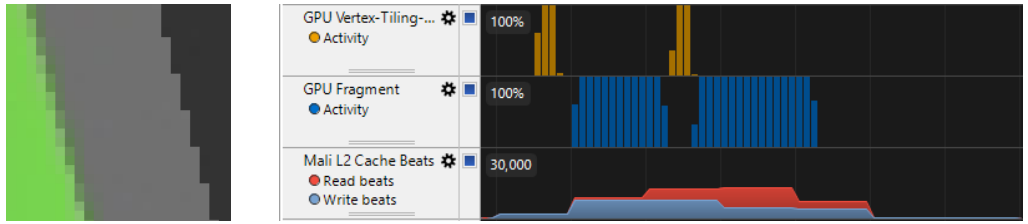
- **Need to query the device to find out what kind of resolve modes it supports:**
  - VK_RESOLVE_MODE_NONE_KHR
  - VK_RESOLVE_MODE_SAMPLE_ZERO_BIT_KHR
  - VK_RESOLVE_MODE_AVERAGE_BIT_KHR
  - VK_RESOLVE_MODE_MIN_BIT_KHR
  - VK_RESOLVE_MODE_MAX_BIT_KHR
- **Can be queried using VkPhysicalDeviceDepthStencilResolvePropertiesKHR structure passed to vkGetPhysicalDeviceProperties2KHR (now promoted to Vulkan 1.1)**

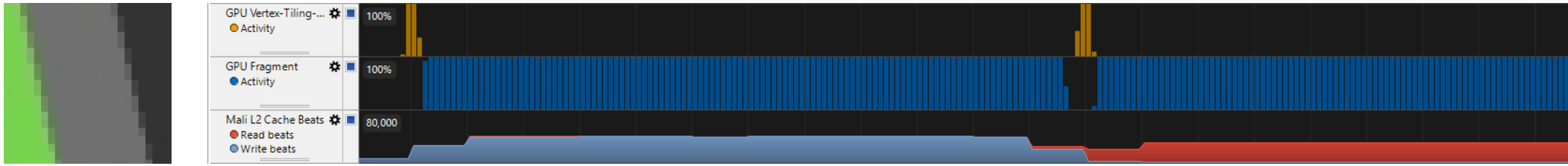
Setup

- **Vulkan requires you to request not just the extension you are interested in, but also all of its dependencies**
- **Instance extensions:**
  - VK_KHR_GET_PHYSICAL_DEVICE_PROPERTIES_2_EXTENSION_NAME
- **Device extensions:**
  - VK_KHR_MULTIVIEW_EXTENSION_NAME
  - VK_KHR_MAINTENANCE2_EXTENSION_NAME
  - VK_KHR_CREATE_RENDERPASS_2_EXTENSION_NAME
  - VK_KHR_DEPTH_STENCIL_RESOLVE_EXTENSION_NAME

Performance







Availability

- Available now in AMD and NVIDIA drivers
- Being tested now for future Android drivers

KHRONOS® GROUP GDC

GDC 2019
#KhronosDevDay