Vulkan - Happy (Belated) Birthday 😊

DESTINATIONS

DOOM

DOTA 2

THE TALEOS PRINCIPLE

VAIN GLORY

unity

UNREAL ENGINE

STEAM VR

Vulkan

Android

Linux

Steam

Windows

GDC

www.gameworks.nvidia.com
1) Mathias - Vulkan Extensions for Maxwell & Pascal
2) Tristan - GPU Work Creation
3) Kyle - Vulkan in Nsight
4) Nuno - GameWorks HBAO+ on Vulkan
New Khronos, Multi-Vendor and NVIDIA functionality!

NVIDIA GDC Vulkan developer driver

Version 376.98 for Windows & Linux

https://developer.nvidia.com/vulkan-driver

LunarG SDK

Version 1.0.42.0

https://www.lunarg.com/vulkan-sdk/
NVIDIA GPU architectures

Kepler (2012)
- GTX 600 series
- GTX 700 series

Maxwell (2014)
- GTX 900 series

Pascal (2016)
- GTX 1000 series
API usability (Kepler+)

VK_KHR_push_descriptor, VK_KHR_descriptor_update_template,
VK_KHR_get_physical_device_properties2, VK_KHR_maintenance1,
VK_KHR_shader_draw_parameters

Cross process memory sharing & synchronization (Kepler+)

VK_KHX_external_memory/semaphore*

Explicit Multi-GPU for AFR, SFR, VR (Kepler+)

VK_KHX_device_group/creation

Multi-View for VR (Kepler+)

VK_KHX_multiview
Multi-Vendor Vulkan extensions

Cross lane shader intrinsics (Kepler+)
  
  VK_EXT_shader_subgroup_vote
  
  VK_EXT_shader_subgroup_ballot

Additional discard rectangles (Kepler+)
  
  VK_EXT_discard_rectangles
Improved Memory Management (Kepler+)

VK_NV_dedicated_allocation

https://developer.nvidia.com/what’s-your-vulkan-memory-type

MSAA improvements (Maxwell+)

VK_NV_sample_mask_override_coverage

GPU Work creation (Kepler+)

VK_NVX_device_generated_commands
NVIDIA extensions (2/2)

Viewport Broadcast (Maxwell+)
- VK_NV_viewport_array2
- VK_NV_geometry_shader_passthrough

Viewport Swizzle (Maxwell+)
- VK_NV_viewport_swizzle

Simultaneous Multi Projection (Pascal)
- VK_NV_clip_space_w_scaling
- VK_NVX_multiview_per_view_attributes
• **New**: broadcast into multiple viewports
  • Same clip space position!
  • Can scale and offset x,y via viewport transformation
  • N = 16
#extension GL_ARB_shader_viewport_layer_array : require
#extension GL_NV_viewport_array2 : require

void main()
{
    // VK 1.0: write in GS, read in FS
    // GL_ARB_shader_viewport_layer_array: write in VS, TS, GS
    gl_ViewportIndex = 2;

    // GL_NV_viewport_array2: write in VS, TS, GS
    gl_ViewportMask[0] = 0x07; // viewports 0, 1 and 2
}
VK_NV_passthrough_geometry_shader (1/2)

Regular geometry shader

- Flexible, expand geometry, has performance impact
- Common use case “pass through” triangle into viewport / render target layer

**New:** explicit pass-through geometry shader

- No geometry expansion
- Can vary some per primitive data e.g. viewport layer / mask (for culling)

**Multi Resolution Shading**

- 3x3 grid viewports of varying resolutions
- Compute overlapping viewports & broadcast
// regular geometry shader

layout(triangles) in;

in Inputs {vec2 texcoord;} v_in[];

layout(triangle_strip,max_vertices=3) out;
out Outputs{vec2 texcoord;} gl_in[];

in gl_PerVertex {vec4 gl_Position;} gl_in[];

void main() {
    int layer = compute_layer();
    for (int i = 0; i < 3; i++) {
        gl_Position = gl_in[i].gl_Position;
        texcoord = v_in[i].texcoord;
        gl_Layer = layer;
        EmitVertex();
    }
}

// passthrough geometry shader

#extension GL_NV_geometry_shader_passthrough: require

layout(triangles) in;
layout(passthrough) in Inputs {vec2 texcoord;} v_in[];

layout(passthrough) in gl_PerVertex {vec4 gl_Position;} gl_in[];

void main() {
    gl_Layer = compute_layer();
}
VK_NV_clip_space_w_scaling (1/2)

- \( v_{clip} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \)

- **New viewport state**: \( w' = w + Ax + By \)

- \( v_{NDC} = \begin{pmatrix} x \\ w' \\ x \\ w' \\ x \\ w' \end{pmatrix} \)

- **Lens Matched Shading**
  - 2x2 viewports, per viewport A & B
  - Compute overlapping viewports & broadcast
VkViewportWScalingNV wCoeffs[1] = { 4.0f, /* A */ 0.2f /* B */};

VkPipelineViewportWScalingStateCreateInfoNV vpWScalingInfo = {
    VK_STRUCTURE_TYPE_PIPELINE_VIEWPORT_W_SCALING_STATE_CREATE_INFO_NV, nullptr,
    VK_TRUE, /* viewportWScalingEnable */
    1, /* viewportCount */
    wCoeffs /* pViewportWScalings */
};

VkPipelineViewportStateCreateInfo vpStateInfo = {
    VK_STRUCTURE_TYPE_PIPELINE_VIEWPORT_STATE_CREATE_INFO, &vpWScalingInfo, ...
};

VkDynamicState dynStates[] = { VK_DYNAMIC_STATE_VIEWPORT_W_SCALING_NV };
vkCmdSetViewportWScalingNV(cmd, 0, 1, wCoeffs);
VK_NV_viewport_swizzle (1/2)

\[ v_{\text{clip}} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \]

**New:** coordinate swizzle viewport state \( v_{\{x,y,z,w\}} \rightarrow \begin{pmatrix} \pm x \\ \pm y \\ \pm z \\ \pm w \end{pmatrix} \)

**Single-pass Voxelization**

3 viewports, swizzled to XY, XZ and/or YZ for dominant plane

**Single-pass cube map rendering**

6 per face viewports with separate swizzle
VK_NV_viewport_swizzle (2/2)

```c
VkViewportSwizzleNV swizzles[1] ={
    VK_VIEWPORT_COORDINATE_SWIZZLE_NEGATIVE_Y_NV, /* x */
    VK_VIEWPORT_COORDINATE_SWIZZLE_POSITIVE_Z_NV, /* y */
    VK_VIEWPORT_COORDINATE_SWIZZLE_NEGATIVE_X_NV, /* z */
    VK_VIEWPORT_COORDINATE_SWIZZLE_POSITIVE_W_NV /* w */
};

VkPipelineViewportSwizzleStateCreateInfoNV swizzleInfo = {
    VK_STRUCTURE_TYPE_PIPELINE_VIEWPORT_SWIZZLE_STATE_CREATE_INFO_NV, nullptr, 0,
    1, /* viewportCount */
    swizzles /* pViewportSwizzles */
};

VkPipelineViewportStateCreateInfo vpStateInfo = {
    VK_STRUCTURE_TYPE_PIPELINE_VIEWPORT_STATE_CREATE_INFO, &swizzleInfo
};
```
VRWorks Building Blocks: **Vulkan** flavor

Accelerating Your VR Games with VRWorks

Cem Cebenoyan
Edward Liu
Daniel Price

Today 12:15 PM - 1:15 PM in this room

www.gameworks.nvidia.com
Vulkan Device Generated Commands

Tristan Lorach
Motivation - What do we try to solve

• Transfer CPU & Driver work ➔ GPU... when GPU can do it faster
• Avoid synchronization / serialization of GPU⇔CPU
• Avoid Memory transfer: stay in Video Memory

• Beneficial for:
  • Occlusion culling
  • Object sorting
  • LODs
  • Compute generating drawcalls: CGG
  • ...
Draw Indirect & Multi-Draw Indirect

Set States → Bind Shaders → Bind Vertex Buffers → Bind Index Buffers → Bind Uniforms

Outside ➔ Can’t change them from inside [M]DI

Bind Framebuffers → Bind Textures

glDraw[Arrays | Elements]Indirect( ▲, P₀)

... CPU/GPU Memory

vertices

indices

struct {
  uint count;
  uint primCount;
  uint first[ ];
} Draw[Elements|Arrays]IndirectCommand;

glMultiDraw[Indexed]Indirect( ▲▲, N )

N=4

vertices₀ → vertices₁ → vertices₂ → vertices₃

... CPU/GPU Memory

indices₀ → indices₁ → indices₂ → indices₃

www.gameworks.nvidia.com
Command-Lists

Set States → Bind Shaders → Bind Vertex Buffers → Bind Index Buffers → Bind Uniforms

glDrawCommandsStatesNV(...)

Bind Framebuffers → Bind Textures

Token buffer
- Special States
- VBO address ...
- EBO address ...
- Uniform Matrix
- Uniform Material
- DrawElements
- EBO address ...
- Uniform Matrix
- Uniform Material
- DrawElements

vertices
indices
Uniforms

GPU Memory

State Object 1
State Object 2
Captured states In the Driver
Relies on Bindless (direct GPU addresses)

Framebuffer₀
Framebuffer₁
...
Array in CPU memory

GDC
Device-Generated-commands (DGC)

- Takes Advantage of Vulkan’s Graphics-Pipeline (== PSO)
  - At last: no more complex state-machine to handle (Command-Lists was limited by that)
  - Can change shaders: shaders are defined in the PSO
- Indirect reference to Vulkan Objects
  - No need for Bindless (No direct GPU address use)
  - Generic ➔ not NVIDIA-specific
- Allows to modify everything from the GPU
  - Traditional Vulkan: vkCmd...() are CPU code (+Multithread)
  - DGC’s are tokens + Arguments + Object references ➔ GPU kernel/shader can do it too
Create an **IndirectCommandsLayout**

- **Layout used to replicate ‘N’ identical kinds of Sequences**
- **Arbitrary sequence of bindings / pushconstants / Drawcalls defined in this Layout**
- **Each can refer to different resources / draw arguments**

**Layout Info**

- Token Pipeline
- Token Idx Buffer
- Token Vtx Buffer
- Token DSet
- Token Draw Indexed
- ...

![Diagram](www.gameworks.nvidia.com)

**Device Generated Command-buffer**

- 1. Bind Pipeline
- 2. Bind Idx Buffer
- 3. Bind Vtx Buffer
- 4. Bind DSet
- 5. Draw Indexed...

**‘N’ times...**

vkCmdProcessCommands(...)
Reserve Space For Commands

• DGC will create commands in a 2ndary Command Buffer
• Driver needs a hint to know the Maximum needed size

\textit{vkCmdReserveSpaceForCommandsNVX}

• Later ➔ Command Generation will populate it \((\textit{vkCmdProcessCommandsNVX})\)
Referencing Resources

Many Vtx/Idx Buffers; Pipeline states and Descriptor Sets to expose

Avoid NVIDIA-Specific Bindless: hide resources behind Reference Ids

Need ‘Max Sizes’ for each Type of Token Operation

```
VkBuffer myIdxBuf
VkBuffer myVtxBuf
vkRegisterObjects(...)
VkPipeline myPSO
VkDescriptorSet myDSet
```
Build Resource Bindings: SoA style

<table>
<thead>
<tr>
<th>Device Generated Command-buffer</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Pipeline Id Refs</th>
<th>VtxBuff Id Refs</th>
<th>IdxBuff Id Refs</th>
<th>DSet Id Refs</th>
<th>Draw Indirect Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>RefId0 Offset</td>
<td>RefId0 Offset</td>
<td>Id0 Offset</td>
<td>Id0 Offset</td>
<td>Idx cnt First Instance</td>
</tr>
<tr>
<td>RefId1 Offset</td>
<td>RefId1 Offset</td>
<td>Id0 Offset</td>
<td>Id0 Offset</td>
<td>Idx cnt First Instance</td>
</tr>
<tr>
<td>RefId2 Offset</td>
<td>RefId2 Offset</td>
<td>Id0 Offset</td>
<td>Id0 Offset</td>
<td>Idx cnt First Instance</td>
</tr>
<tr>
<td>RefId3 Offset</td>
<td>RefId3 Offset</td>
<td>Id1 Offset</td>
<td>Id1 Offset</td>
<td>Idx cnt First Instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Id1 Offset</td>
<td>Id2 Offset</td>
<td>Idx cnt First Instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Id1 Offset</td>
<td>Id2 Offset</td>
<td>Idx cnt First Instance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Id1 Offset</td>
<td>Id2 Offset</td>
<td>Idx cnt First Instance</td>
</tr>
</tbody>
</table>

Stored in Arbitrary Buffers + Offset (GPU Memory)

VkBuffer A

VkBuffer B

Device Generated Command-buffer

Generates commands in targeted Cmd Buffer

vkCmdProcessCommandsNVX(...)

Reserved area

Generates commands in targeted Cmd Buffer

www.gameworks.nvidia.com
Flexible Sequencing

Pipeline Id Refs

Draw Indirect Arguments

Sequence buffer

Device Generated Command-buffer

vkCmdProcessCommandsNVX(…)

www.gameworks.nvidia.com
**Additional remarks**

- You can rebuild DGC Command Buffer every frame
- You can modify object references in buffers
- But use a barrier `VK_PIPELINE_STAGE_COMMAND_PROCESS_BIT_NVX`
- Each Array of data can have their own frequency divisor

<table>
<thead>
<tr>
<th>Pipeline Id Refs</th>
<th>RefId0 Offset</th>
<th>RefId1 Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>VtxBuff Id Refs</td>
<td>RefId0 Offset</td>
<td>RefId1 Offset</td>
</tr>
<tr>
<td>Draw Indirect Arguments</td>
<td>Idx cnt</td>
<td>First Instance</td>
</tr>
</tbody>
</table>

- `vkCmdProcessCommandsNVX` Optimizes binding/state changes
Conclusion

• This extension is an alternate approach to VK command-buffers

• gives full control to GPU for creation (after Graphics-Pipeline / resources are referenced in tables)

• CPU-friendly

• Multi-thread friendly

• Still under evaluation (hence NVX)
References

Blog:


Samples:

https://github.com/nvpro-samples/gl_vk_threaded_cadscene


Extension spec:

VK_NV[X]_device_generated_commands

https://www.khronos.org/registry/vulkan/specs/1.0-extensions/html/vkspec.html#VK_NVX_device_generated_commands
NVIDIA Nsight Vulkan Support

Kyle Spagnoli
Nsight

What is Nsight VSE?

Understand CPU/GPU interaction
Explore and debug your frame as it is rendered
Profile your frame to understand hotspots and bottlenecks
Save your frame for targeted analysis and experimentation
Leverage the Microsoft Visual Studio platform
Nsight

New features in version 5.3

Vulkan 1.0.42 support
Vulkan extensions
Vulkan serialization
Vulkan shader reflection
Vulkan descriptor view
Bug squishing

Theme support
OpenVR support
New shaders view
Microsoft Hybrid support
D3D11 / D3D12 point releases
## Event View - API Trace

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
<th>Object</th>
<th>CPU µs</th>
<th>GPU µs</th>
<th>Thread</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>vkAcquireNextImageKHR(VkDevice device = 0x2a635d00)</td>
<td>0x2a635d00</td>
<td>3</td>
<td>-</td>
<td>3500</td>
</tr>
<tr>
<td>19</td>
<td>// Recorded 21 commands to primary command buffer; VkCommandBuffer = 0x2a673700, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>// Recorded 82 commands to primary command buffer; VkCommandBuffer = 0x2a7bfcb0, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>// Recorded 702 commands to primary command buffer; VkCommandBuffer = 0x2a901370, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>// Recorded 143 commands to primary command buffer; VkCommandBuffer = 0x2a9007f0, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>// Recorded 244 commands to primary command buffer; VkCommandBuffer = 0x2a89f5d0, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>// Recorded 1083 commands to primary command buffer; VkCommandBuffer = 0x2a823200, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>// Recorded 21 commands to primary command buffer; VkCommandBuffer = 0x2a822870, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>26</td>
<td>// Mapped memory update; VkDeviceMemory = 0x3f94b0, ranges = 0x6f158 @ 21899024, ...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>// Mapped memory update; VkDeviceMemory = 0x3f9f070, ranges = 0x7f336e @ 2195808, 6...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td>vkFlushMappedMemoryRanges(VkDevice device = 0x2a635d00, uint32_t memoryRangeCount = 0x2a635d00, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>29</td>
<td>// Recorded 17 commands to primary command buffer; VkCommandBuffer = 0x2a82d230, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>// Mapped memory update; VkDeviceMemory = 0x3f945330, ranges = 0x3f945330</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>31</td>
<td>vkFlushMappedMemoryRanges(VkDevice device = 0x2a635d00, uint32_t memoryRangeCount = 0x2a635d00, Vk...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>32</td>
<td>vkResetFences(VkDevice device = 0x2a635d00, uint32_t fenceCount = 1, VkFence pFences = 0x...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>33</td>
<td>// Mapped memory update; VkDeviceMemory = 0x3f945330, ranges = 0x3f945330</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>34</td>
<td>vkQueueSubmit(VkQueue queue = 0x41437090, uint32_t submitCount = 1, VkSubmitInfo pSubmit...</td>
<td>204</td>
<td>-</td>
<td>-</td>
<td>3500</td>
</tr>
<tr>
<td>35</td>
<td>vkBeginCommandBuffer(VkCommandBuffer commandBuffer = 0x2a82d230, VkCommandBuffer...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>36</td>
<td>vkCmdPipelineBarrier(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>37</td>
<td>vkCmdCopyBufferToImage(VkCommandBuffer commandBuffer = 0x2a82d230, VkBuffer srcBuffer...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>38</td>
<td>vkCmdCopyBufferToImage(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>39</td>
<td>vkCmdCopyBufferToImage(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>vkCmdCopyBufferToImage(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>41</td>
<td>vkCmdCopyBufferToImage(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>42</td>
<td>vkCmdCopyBufferToImage(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags...</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Command buffer construction**

**Memory updates**

**Command buffer execution**
Event Scrubber

- Multi-queue / multi-thread
- State buckets & VK_EXT_debug_markers
- Synchronization
Current Target - Draws, Clears, & Blits

Color(s), depth, & stencil target

Wireframe highlights rendered geometry
API Inspector - All the render state

- Pipeline
- Render Pass
- Framebuffer
- Input Assembly
- Shaders
- Viewport
- Raster
- Pixel Ops.
- Misc.
API Inspector - Shader Reflection

Names from SPIRV decorations

Uniform values
API Inspector

Integration with SPIRV-Cross to get human readable, efficient GLSL representation of shaders.

SPIRV decorations for uniforms & interfaces if available
Geometry Viewer - Graphical

Control attributes

Control rendering modes
Geometry Viewer - Vertex Data

Index buffer ordering

Vertex attribute values
Resource View

Thumbnail previews

Graphics / memory previews

Resource information

Revision information

www.gameworks.nvidia.com
Resource View - Tagging

- Used in shader
- Copy/blit source
- Index buffer
- Etc.

Tag a resource’s consumption on the scrubber
Device Memory

- Memory objects
- Contained resources
- Raw memory
- Mini-map view
## Descriptor Sets

<table>
<thead>
<tr>
<th>Set</th>
<th>Layout</th>
<th>Pool</th>
<th>Consumptions</th>
<th>Binding</th>
<th>Type</th>
<th>Stage</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x001100</td>
<td>0x002100</td>
<td>0x003100</td>
<td>0x004100</td>
<td>0x005100</td>
<td>0x006100</td>
<td>0x007100</td>
<td>0x008100</td>
</tr>
<tr>
<td>0x009100</td>
<td>0x00a100</td>
<td>0x00b100</td>
<td>0x00c100</td>
<td>0x00d100</td>
<td>0x00e100</td>
<td>0x00f100</td>
<td></td>
</tr>
</tbody>
</table>

### All descriptors objects with usage counts

### Associated resources

### Selected resource information

**Pool information**

[www.gameworks.nvidia.com](http://www.gameworks.nvidia.com)
C/C++ Serialization - Save to Disk

Example C/C++ code:
```c
void createsourcecode()
{
    // Example code
    ...
}
```

Human readable C/C++ code

Loop frame in insolation
C/C++ Serialization - Challenges Solved

Portability

```c
typedef struct VKMemoryAllocateInfo {
    VkStructureType type;
    const void* pNext;
    VkDeviceSize allocationSize;
    uint32_t memoryTypeIndex;
} VKMemoryAllocateInfo;
```

Frame looping

*Where are my particles!?*

- Acquire/render/present flow
- Multi-threading
- Synchronization
- Command buffer re-recording
- Pre-frame barriers
- Multi-buffering
- Respect object model
- Unnecessary finish calls
- Missing extensions
Supported Vulkan Extensions

VK_KHR_surface
VK_KHR_swapchain
VK_KHR_display
VK_KHR_display_swapchain
VK_KHR_win32_surface
VK_EXT_debug_report
VK_NV_glsl_shader
VK_KHR_sampler_mirror_clamp_to_edge
VK_IMG_filter_cubic
VK_EXT_debug_marker
VK_NV_dedicated_allocation
VK_EXT_validation_flags*
VK_KHR_get_physical_device_properties2*
VK_KHR_shader_draw_parameters*
VK_KHR_shader_subgroup_ballot*
VK_KHR_shader_subgroup_vote*
VK_KHR_maintenance1*
VK_KHR_maintenance1*

*new for Nsight 5.3
Roadmap

Profiler & Performance Analysis
Android & Linux Support
Shader Editing
Sparse Texture Support
Improved Resource Barrier Visualization
Future Extensions & Core Releases
Download Nsight with Vulkan Support Today

Version 5.2 right now

Version 5.3 soon after GDC

http://www.nvidia.com/object/nsight.html
GameWorks HBAO+ on Vulkan

Nuno Subtil
Horizon-Based Ambient Occlusion +

- Bavoil, L., Sainz, M., Image-Space Horizon-Based Ambient Occlusion, Siggraph 2008

- HBAO+ improves upon existing Ambient Occlusion techniques to add richer, more detailed, more realistic shadows around objects that occlude rays of light

- Compared to previous techniques, HBAO+ is faster, more efficient, and significantly better
HBAO+ on Vulkan

• Most requested GameWorks port
  • GL, DX11, DX12 ports already exist
• Vulkan is important for NVIDIA
  • Industry-leading driver stack for Vulkan
  • Library effort ramping up
• Prioritization based on developer feedback
HBAO+ Interface: Vulkan vs DX12

• Very similar APIs:
  • Explicit and verbose
  • Application and library code responsible for synchronization

• Logistical differences:
  • Vulkan allows no queries on objects; all object information must be explicit
  • Context creation requires handshaking between app and library for extension support

```c
struct GFSDK_SSAO_ShaderResourceView_D3D12
{
    ID3D12Resource* pResource;
    GFSDK_SSAO_UINT64 GpuHandle;
};
```

```c
struct GFSDK_SSAO_Image.VK
{
    uint32_t Width;
    uint32_t Height;
    VkImageView View;
    VkFormat Format;
    VkImageLayout Layout;
    VkSampleCountFlagBits MultiSampleBit;
};
```
Context creation handshake

Vulkan extensions required by middleware must be enabled at context creation time

1. Query Vulkan runtime for list of available device extensions

2. Call HBAO+ library with supported extensions, request list of extensions to enable

3. Create Vulkan device context that includes all extensions requested by HBAO+ library

4. Create HBAO+ context
Synchronization

• Vulkan API allows (and requires) fine-grained synchronization

• HBAO+ library needs to know:
  • Which GPU engine last operated on a given input/output surface
  • What kind of operation (read? write?) was performed

• App requires the same information after calling into HBAO+
Results
Implementation details

• Reused all of the API independent code in HBAO+
  • Added Vulkan-specific backend

• Reused HBAO+ HLSL shaders
  • `glslc` can compile all of our shaders to SPIR-V
    • Supports `#include`, command-line preprocessor definitions
  • Very few Vulkan-specific modifications required
    • Workarounds for missing language features
    • Shader Model 5.1 maps well to SPIR-V
How can I use it?

• Productization effort underway
  • Expected release around Summer
  • Same release model as other versions of HBAO+

• Can work with early adopters
  • Be prepared for some rough edges...
  • Get in touch: gameworks@nvidia.com
Thank you for attending! 😊

Mathias Schott - mschott@nvidia.com
Tristan Lorach - tlorach@nvidia.com
Kyle Spagnoli - kspagnoli@nvidia.com
Nuno Subtil - nsubtil@nvidia.com