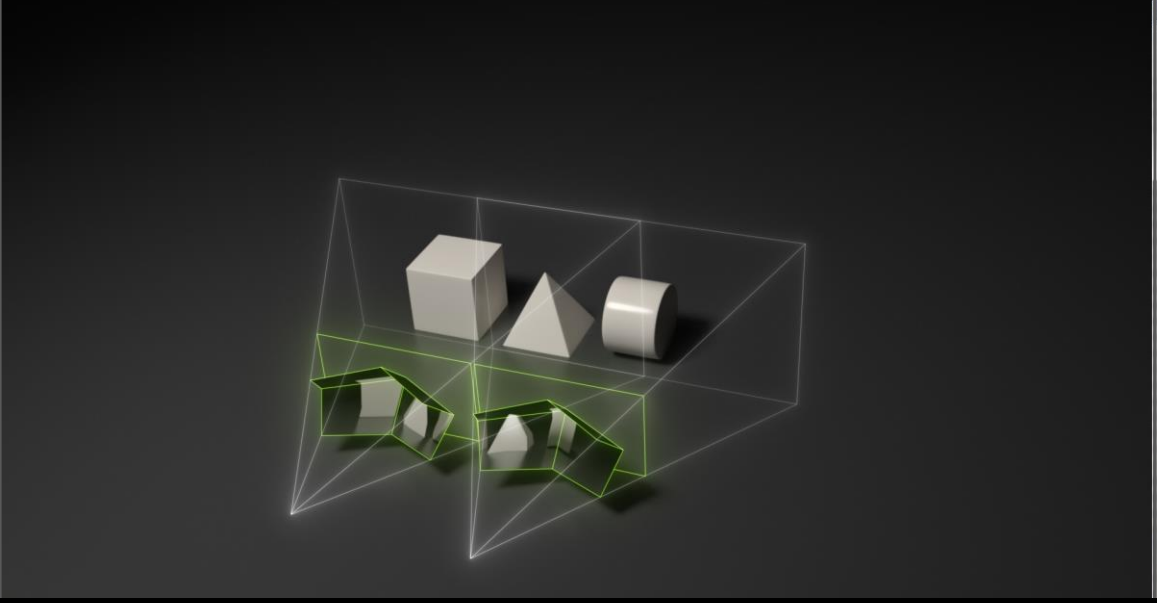
NVIDIA Vulkan® Update

March 3rd 2017

Mathias Schott
Tristan Lorach
Kyle Spagnoli
Nuno Subtil

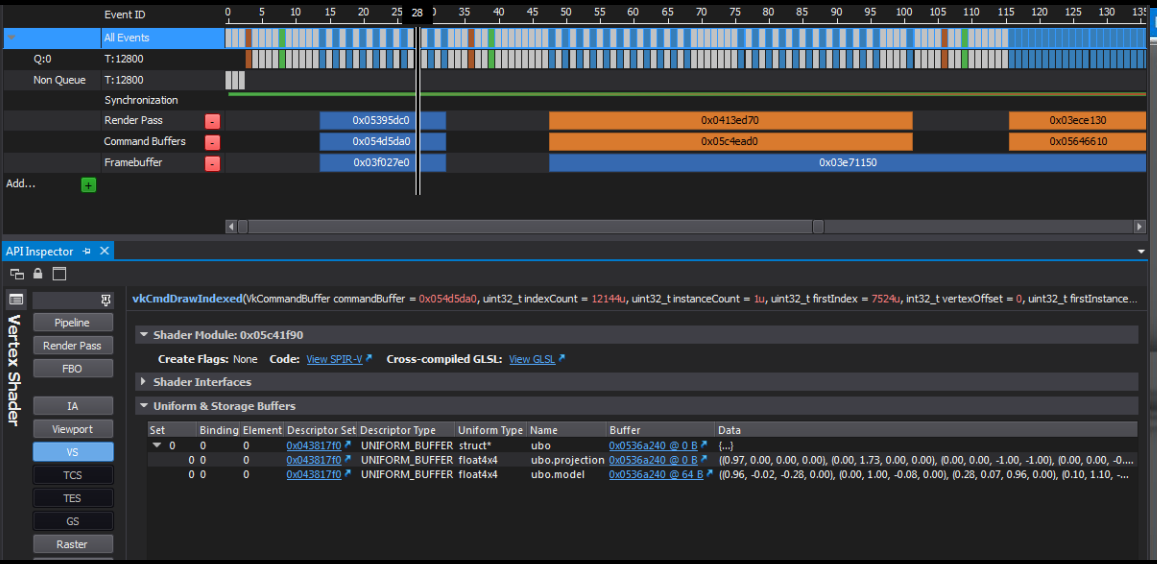
NVIDIA

Vulkan - Happy (Belated) Birthday ☺

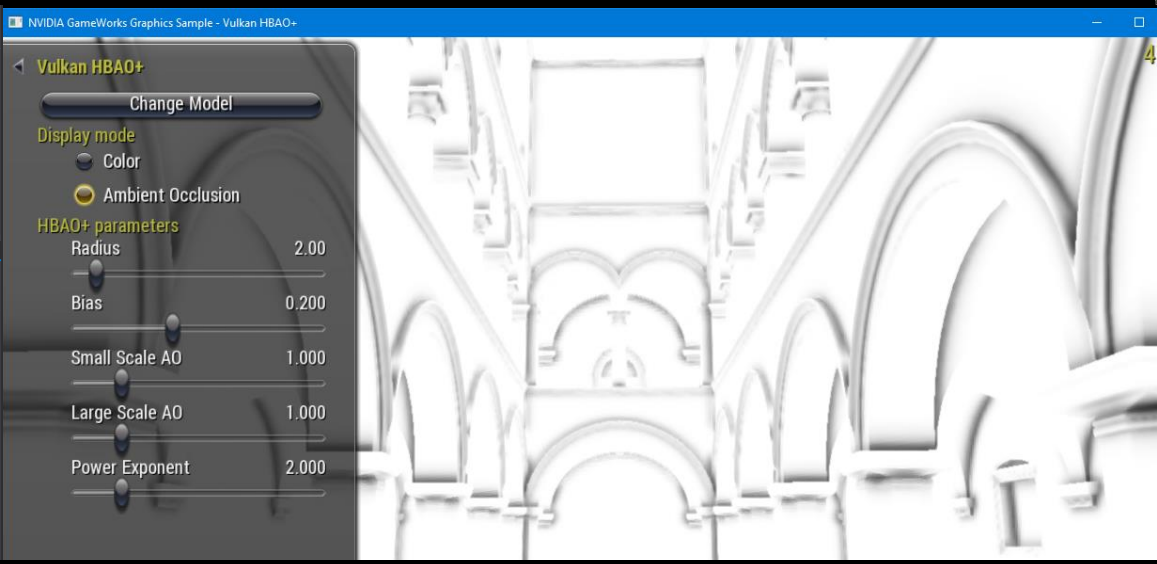


1) Mathias – Vulkan Extensions for Maxwell & Pascal

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)

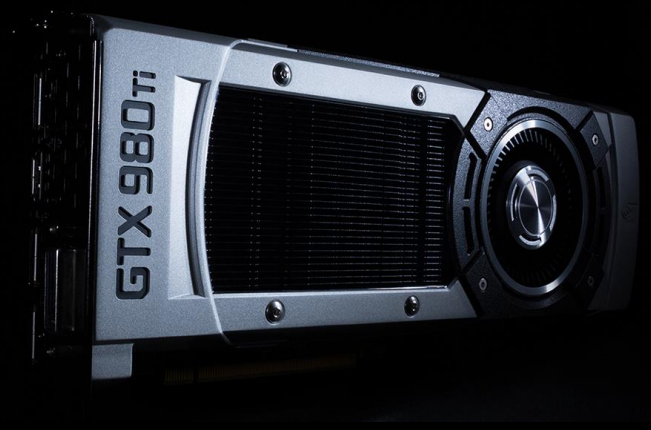
- GTX 600 series
- GTX 700 series

Maxwell (2014)

- GTX 900 series

Pascal (2016)

- GTX 1000 series



KHRONOS GROUP extensions

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

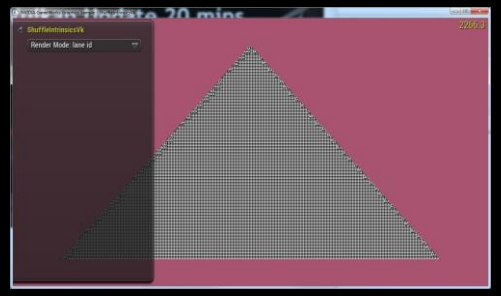
Multi-Vendor Vulkan extensions

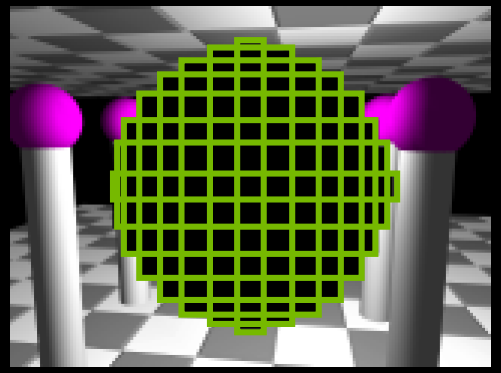
Cross lane shader intrinsics (Kepler+)

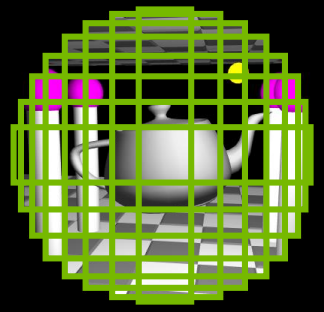
- VK_EXT_shader_subgroup_vote
- VK_EXT_shader_subgroup_ballot

Additional discard rectangles (Kepler+)

- VK_EXT_discard_rectangles



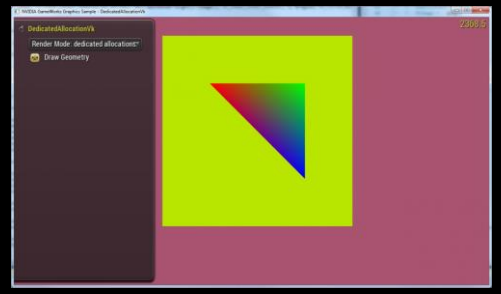




NVIDIA extensions (1/2)

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+)

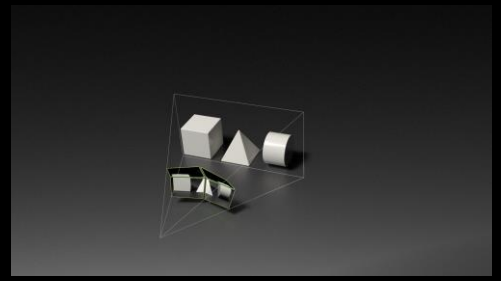
- VK_NV_viewport_array2
- VK_NV_geometry_shader_passthrough

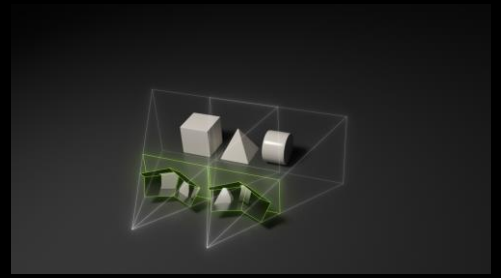
Viewport Swizzle (Maxwell+)

- VK_NV_viewport_swizzle

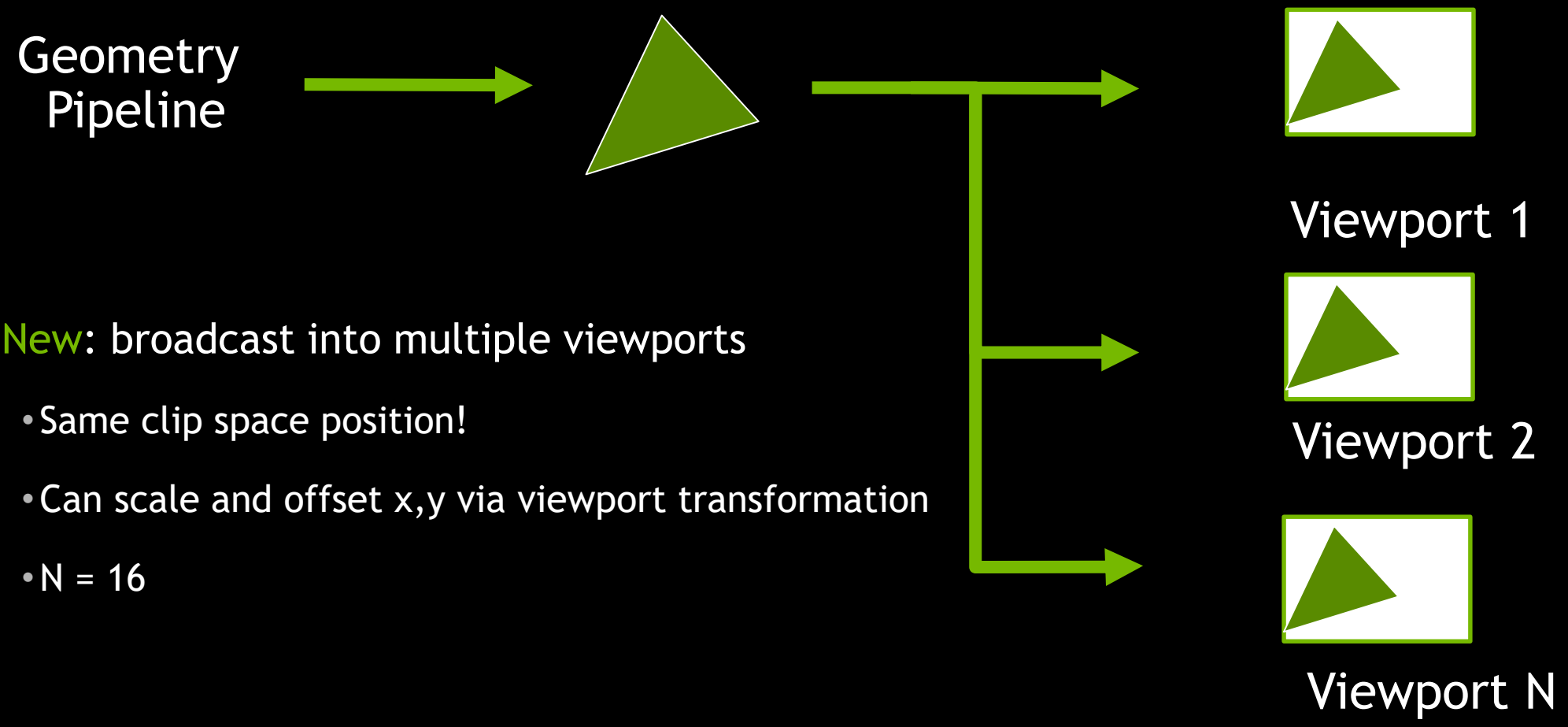
Simultaneous Multi Projection (Pascal)

- VK_NV_clip_space_w_scaling
- VK_NVX_multiview_per_view_attributes





VK_NV_viewport_array2 (1/2)



- New: broadcast into multiple viewports
  - Same clip space position!
  - Can scale and offset x,y via viewport transformation
  - N = 16

VK_NV_viewport_array2 (2/2)

```
#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 / rendertarget 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

VK_NV_passthrough_geometry_shader (2/2)

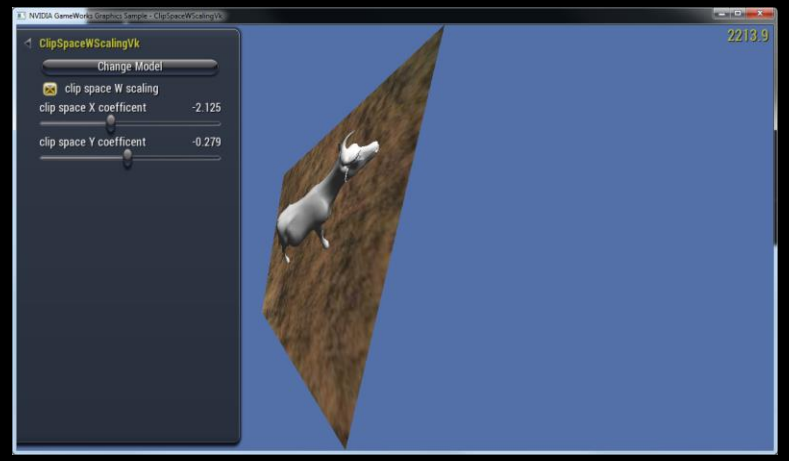
```
// regular geometry shader


layout(triangles) in;

in Inputs  {vec2 texcoord;} v_in[];

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

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: $w' = w + Ax + By$
- $v_{NDC} = \begin{pmatrix} \frac{x}{w'} \\ \frac{x}{w'} \\ \frac{x}{w'} \end{pmatrix}$
- Lens Matched Shading
  - 2x2 viewports, per viewport A & B

  Compute overlapping viewports & broadcast

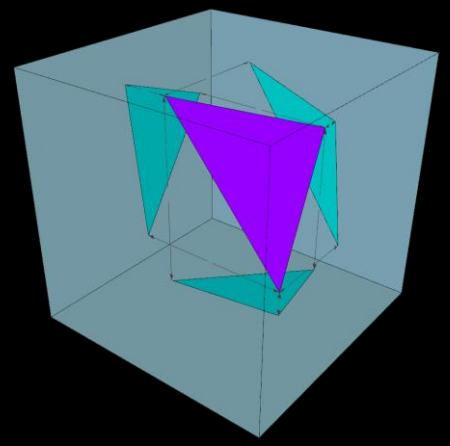


VK_NV_clip_space_w_scaling (2/2)

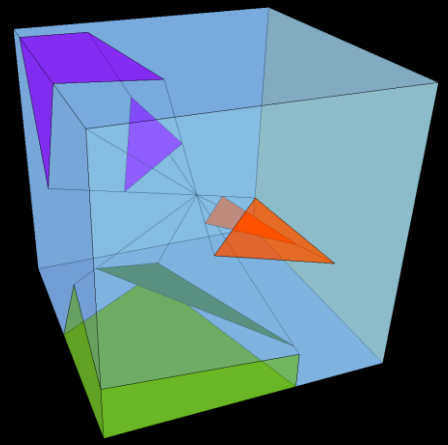
```
VkViewportWScalingNV wCoeffs [1] = {  4.0f, /*A */ 0.2f  /* B */};

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)

$$v_{clip} = \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$



New: coordinate swizzle viewport state $v_{\{x,y,z,w\}} \rightarrow \begin{cases} \pm x \\ \pm y \\ \pm z \\ \pm w \end{cases}$



Single-pass Voxelization

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

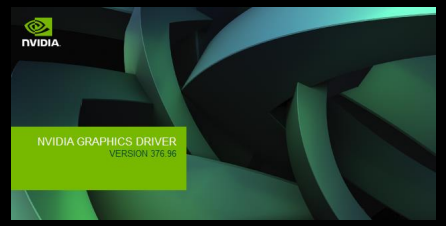
Single-pass cube map rendering

- 6 per face viewports with separate swizzle

VK_NV_viewport_swizzle (2/2)

```
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

NVIDIA

Vulkan Device Generated Commands

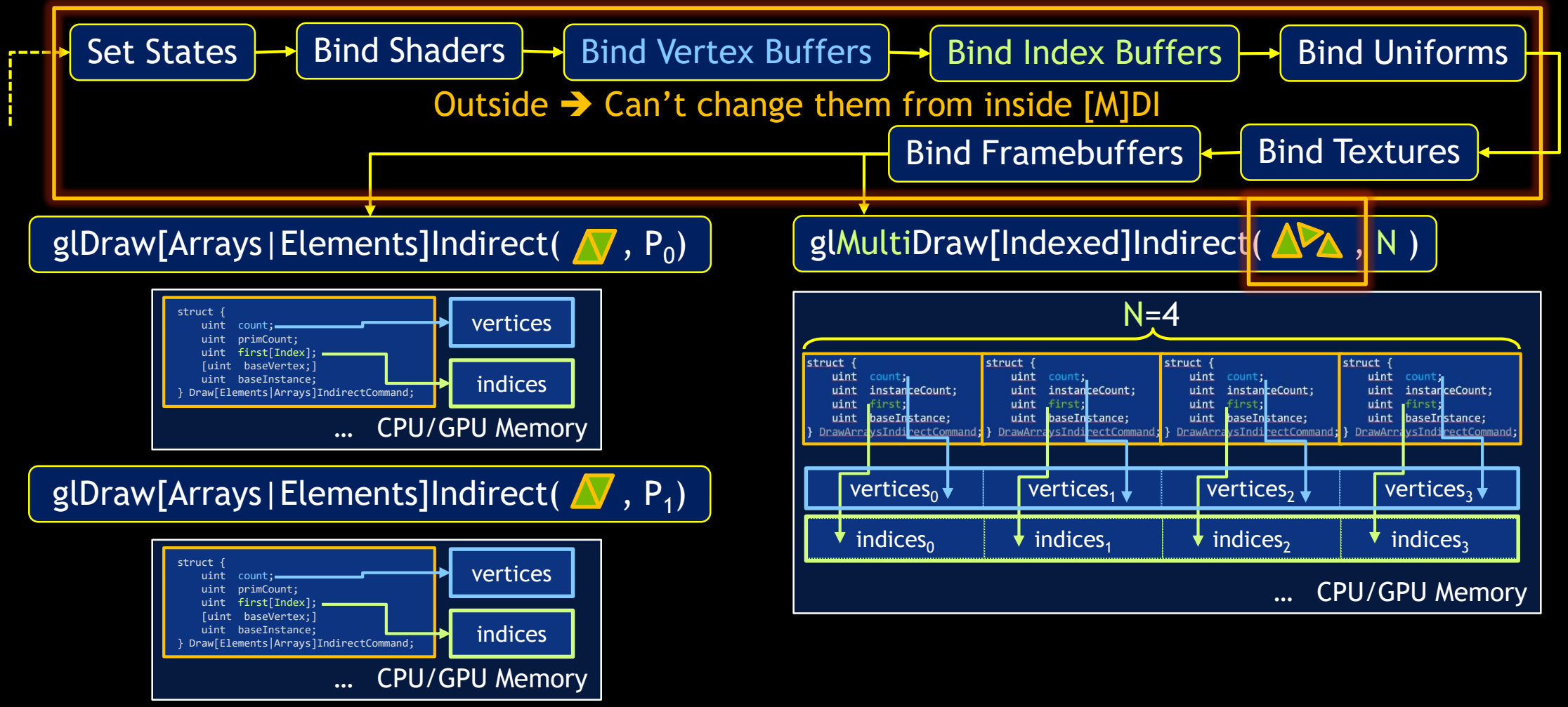
Tristan Lorach

NVIDIA

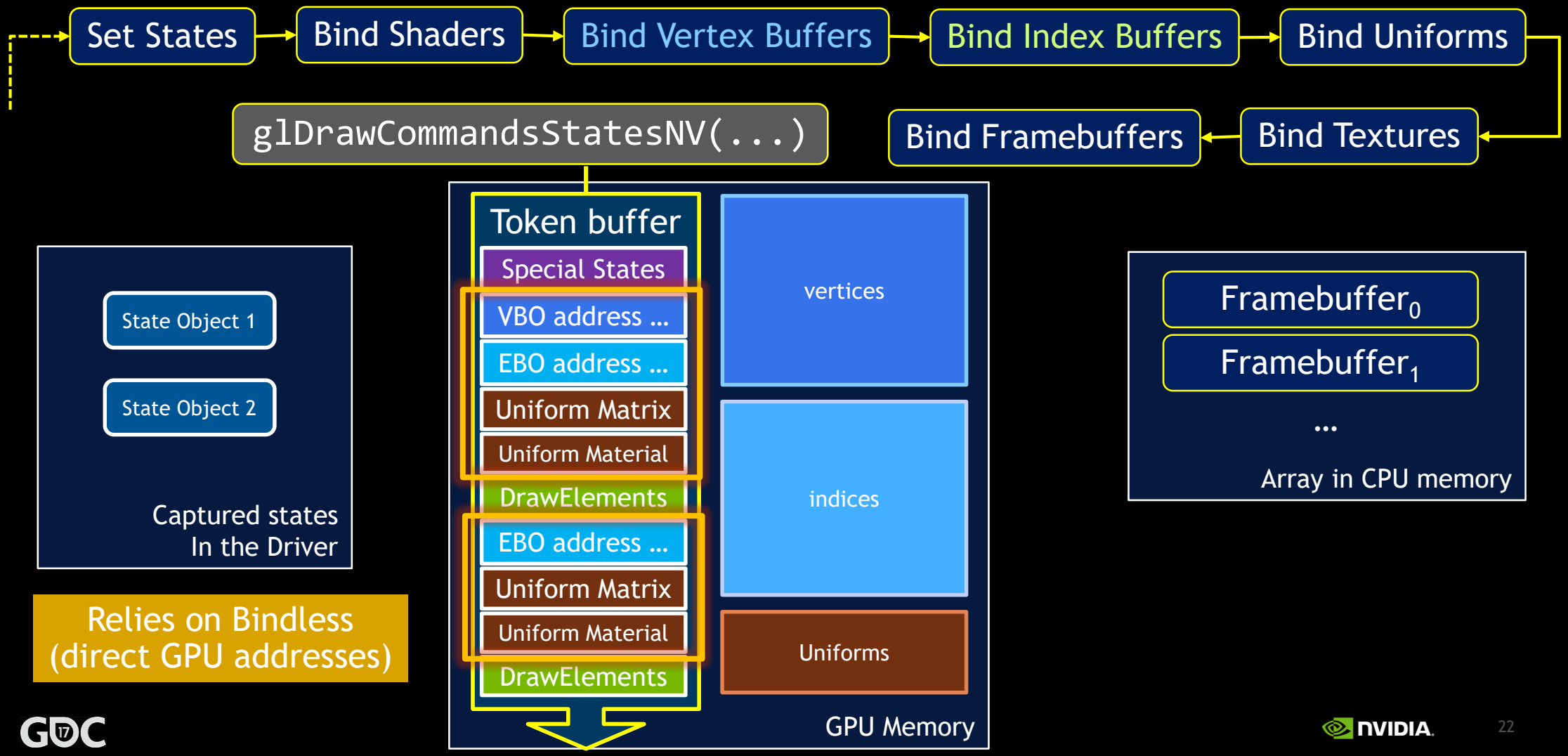
Motivation - What do we try to solve

- 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



Command-Lists



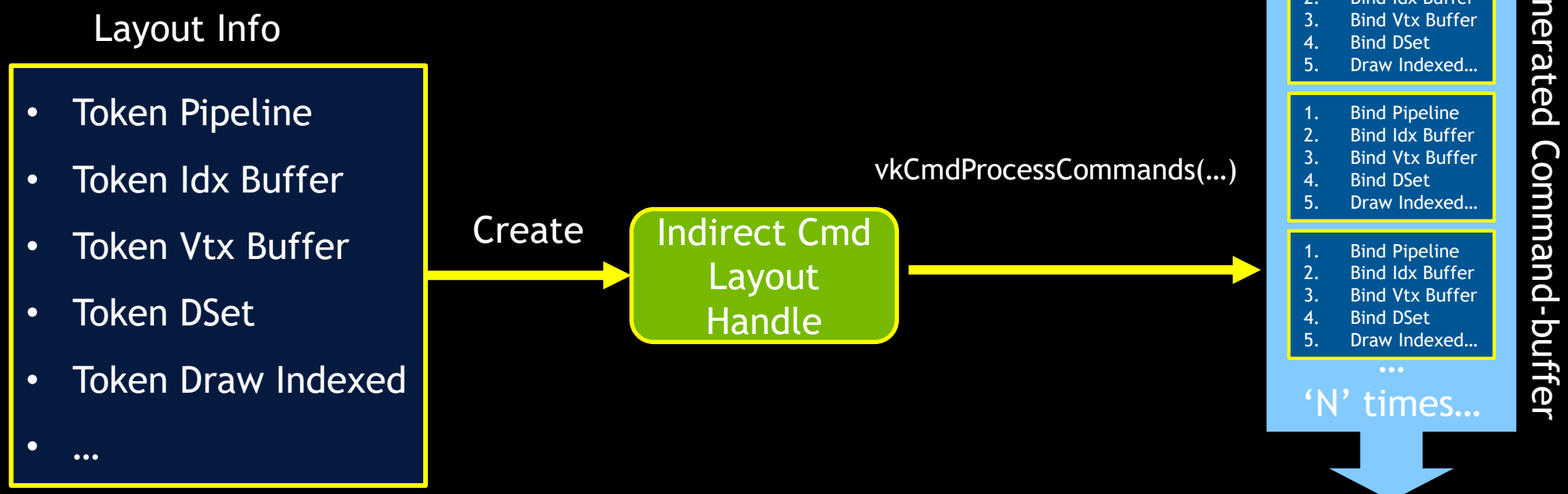
Relies on Bindless
(direct GPU addresses)

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



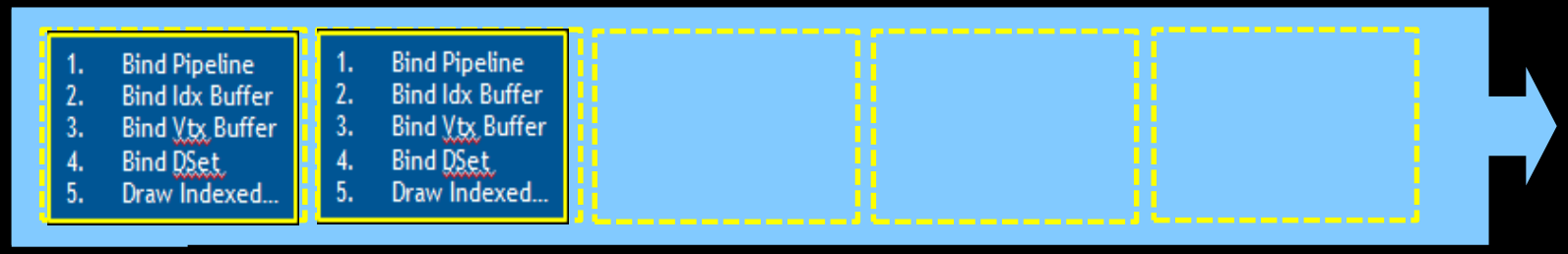
Reserve Space For Commands

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

vkCmdReserveSpaceForCommandsNVX

- Later ➔ Command Generation will populate it *(vkCmdProcessCommandsNVX)*

vkCmdReserveSpaceForCommandsNVX(…)



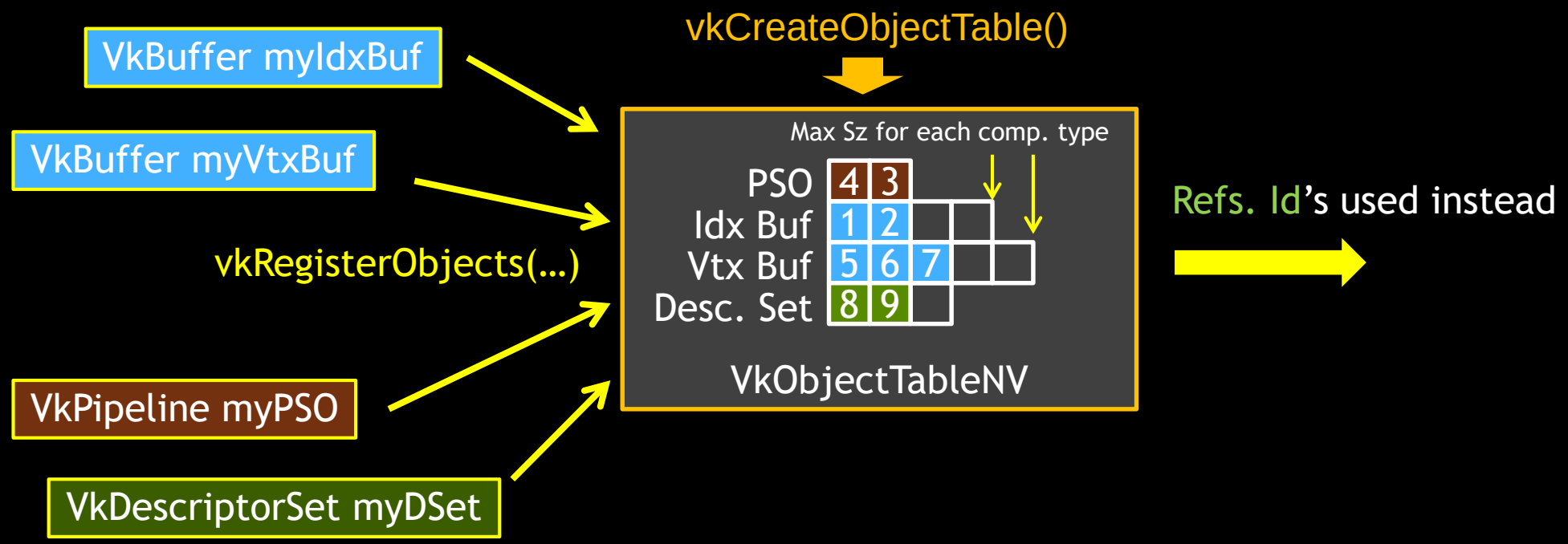
2ndary Command-buffer

Referencing Resources

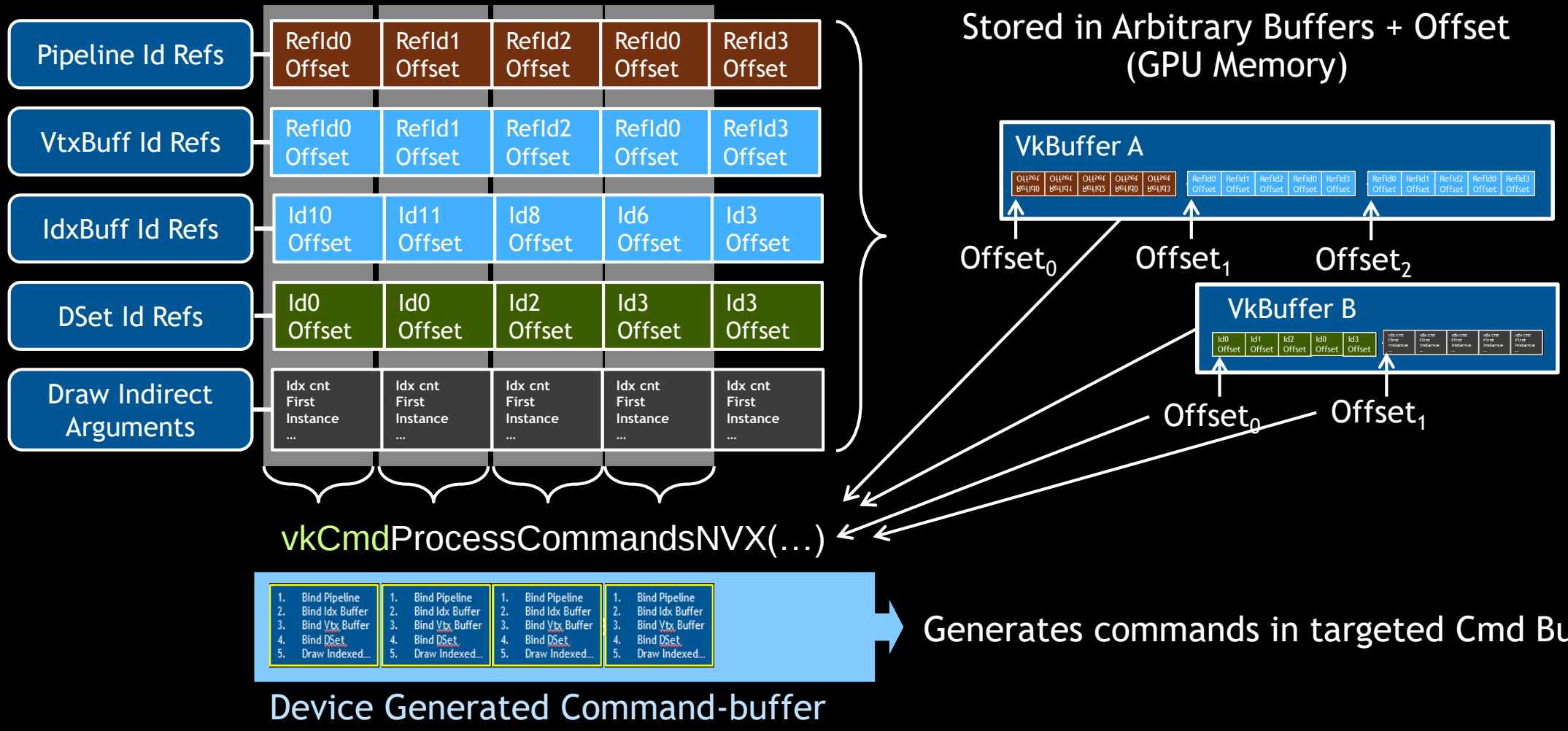
Many Vtx/Idx Buffers; Pipeline states and Descriptor Sets to expose

Avoid NVIDIA-Specific Bindless: hide resources behind Reference Ids

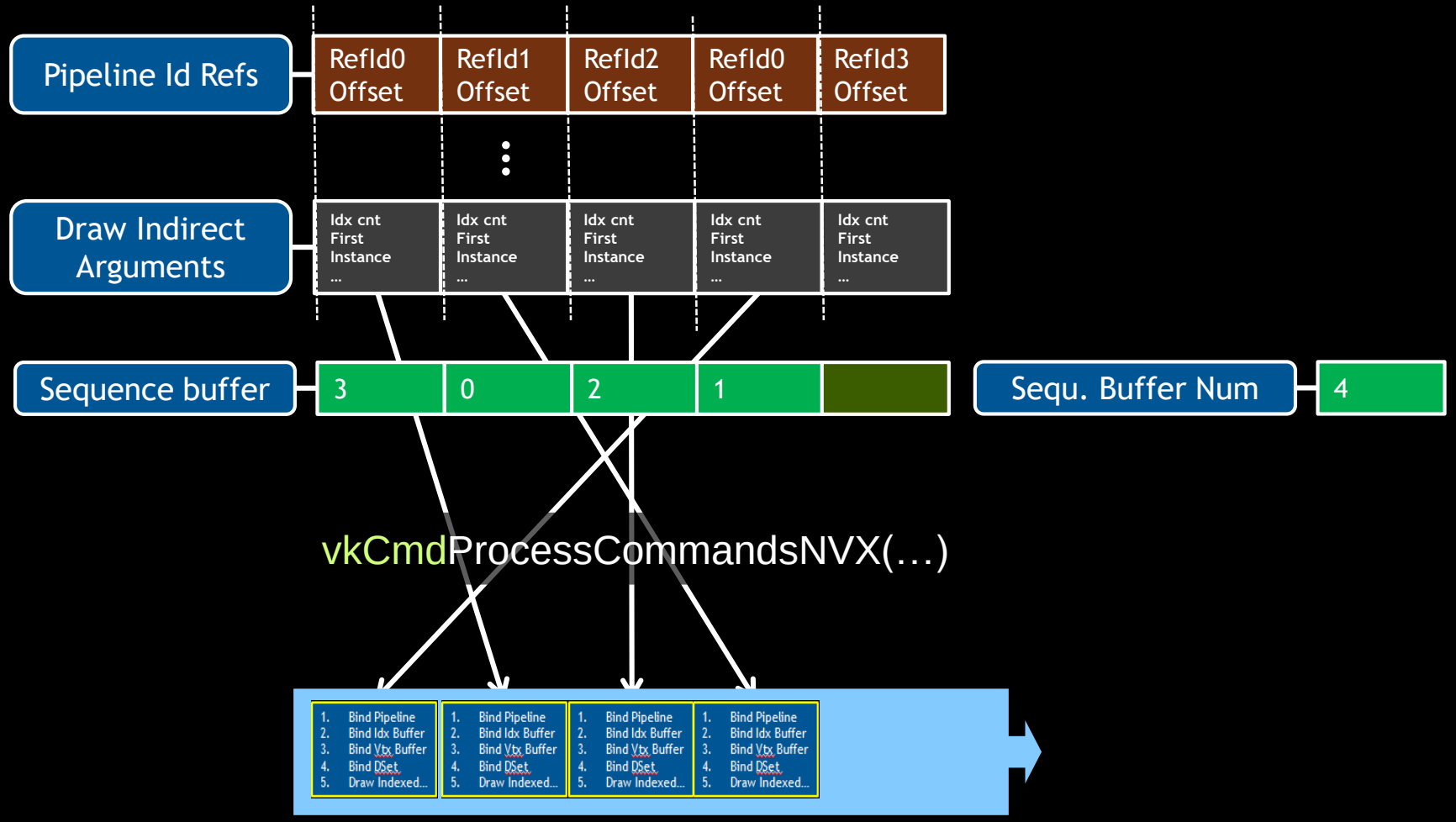
Need ‘Max Sizes’ for each Type of Token Operation



Build Resource Bindings : SoA style



Flexible Sequencing

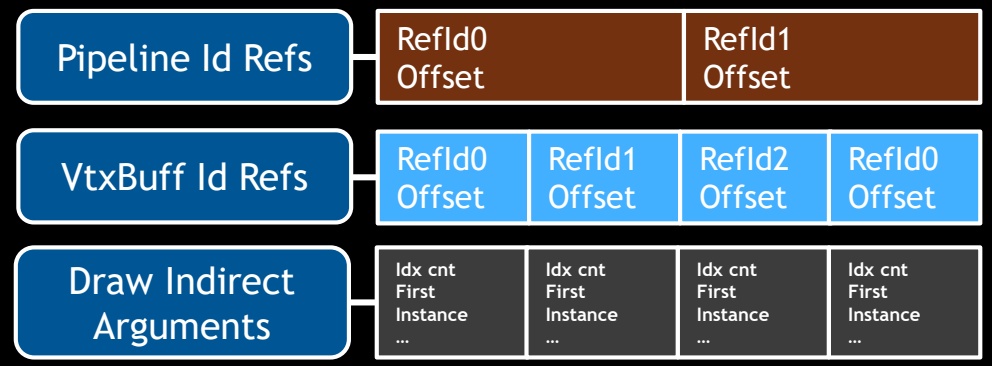


Device Generated Command-buffer

www.gameworks.nvidia.com

Additional remarks

- You can rebuilding 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



- 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:

https://developer.nvidia.com/device-generated-commands-vulkan

Samples:

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

https://developer.nvidia.com/driver-and-new-sample-vknvxdevicegeneratedcommands

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

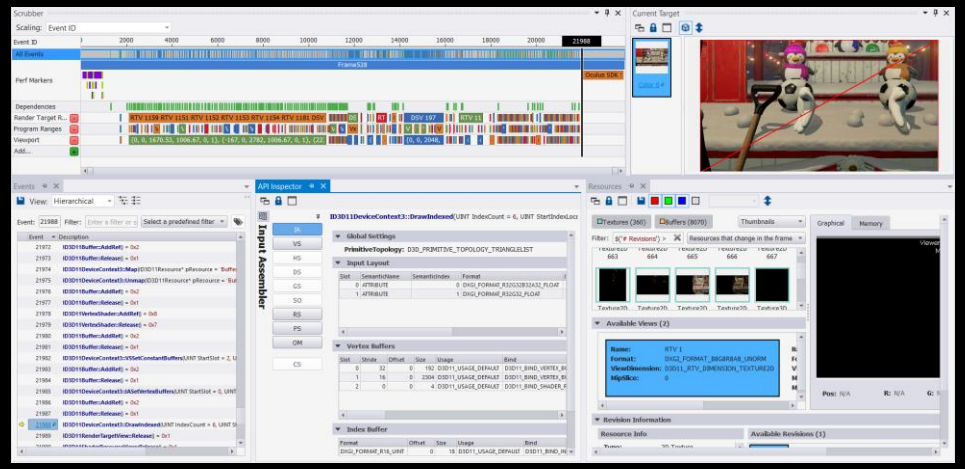
NVIDIA Nsight Vulkan Support

Kyle Spagnoli

NVIDIA

Nsight

What is Nsight VSE?



Understand CPU/GPU interaction

Explore and debug your frame as it is rendered

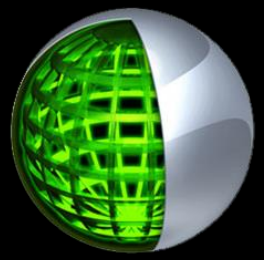
Profile your frame to understand hotspots and bottlenecks

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

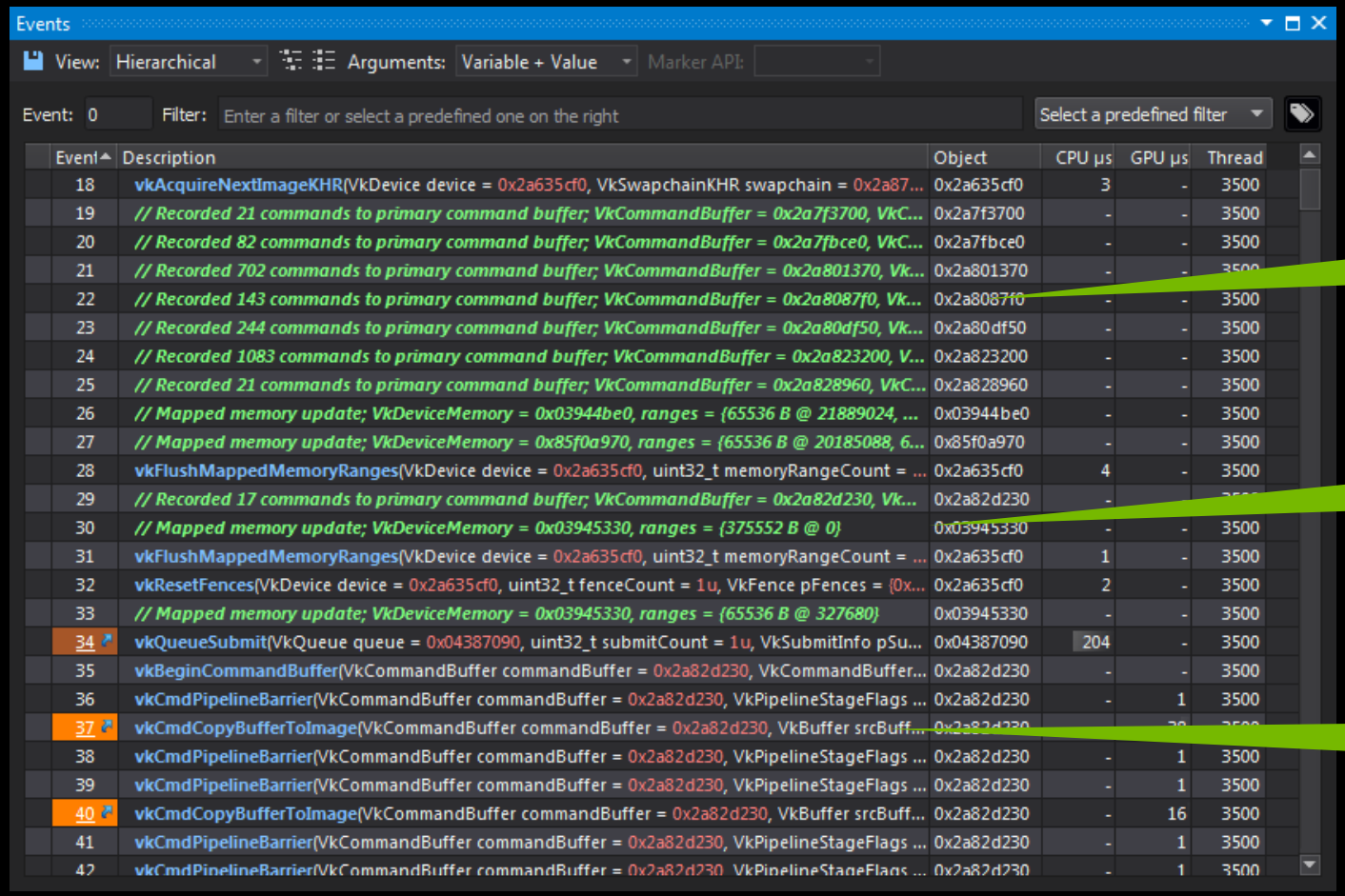
OpenVR support

New shaders view

Microsoft Hybrid support

D3D11 / D3D12 point releases

Event View – API Trace



Events

View: Hierarchical Arguments: Variable + Value Marker API:

Event: 0 Filter: Enter a filter or select a predefined one on the right Select a predefined filter

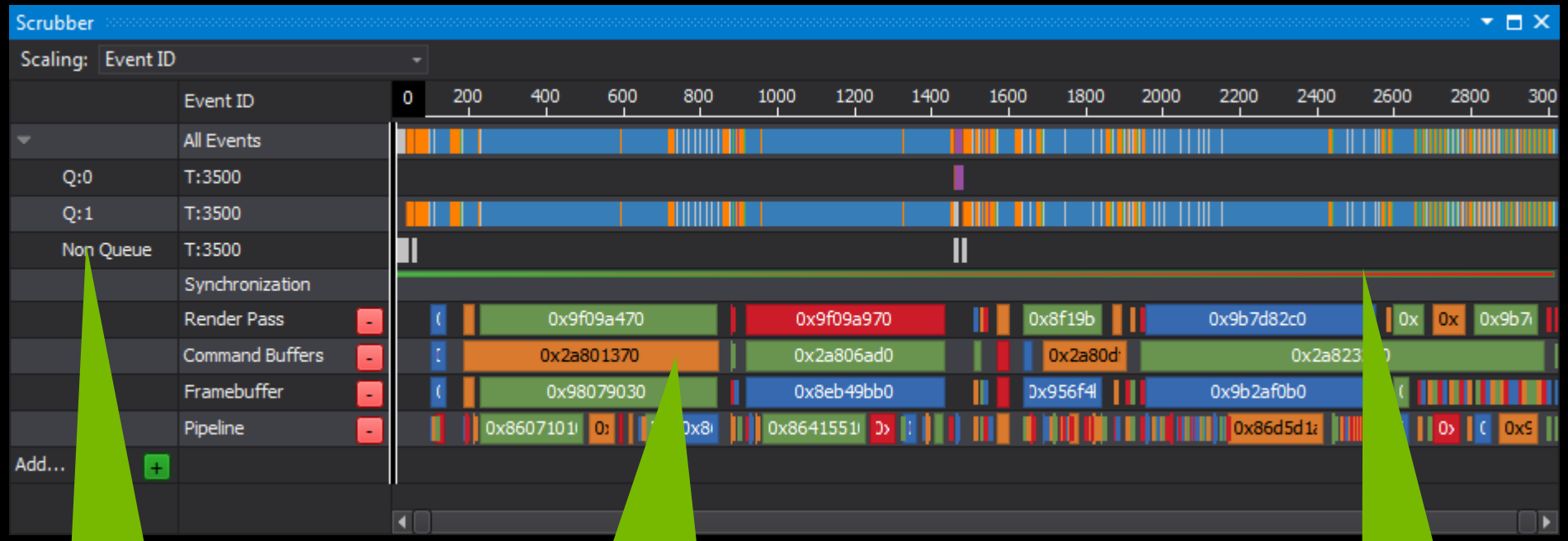
Event	Description	Object	CPU µs	GPU µs	Thread
18	vkAcquireNextImageKHR(VkDevice device = 0x2a635cf0, VkSwapchainKHR swapchain = 0x2a87...	0x2a635cf0	3	-	3500
19	// Recorded 21 commands to primary command buffer; VkCommandBuffer = 0x2a7f3700, VkC...	0x2a7f3700	-	-	3500
20	// Recorded 82 commands to primary command buffer; VkCommandBuffer = 0x2a7fbce0, VkC...	0x2a7fbce0	-	-	3500
21	// Recorded 702 commands to primary command buffer; VkCommandBuffer = 0x2a801370, Vk...	0x2a801370	-	-	3500
22	// Recorded 143 commands to primary command buffer; VkCommandBuffer = 0x2a8087f0, Vk...	0x2a8087f0	-	-	3500
23	// Recorded 244 commands to primary command buffer; VkCommandBuffer = 0x2a80df50, Vk...	0x2a80df50	-	-	3500
24	// Recorded 1083 commands to primary command buffer; VkCommandBuffer = 0x2a823200, V...	0x2a823200	-	-	3500
25	// Recorded 21 commands to primary command buffer; VkCommandBuffer = 0x2a828960, VkC...	0x2a828960	-	-	3500
26	// Mapped memory update; VkDeviceMemory = 0x03944be0, ranges = {65536 B @ 21889024, ...	0x03944be0	-	-	3500
27	// Mapped memory update; VkDeviceMemory = 0x85f0a970, ranges = {65536 B @ 20185088, 6...	0x85f0a970	-	-	3500
28	vkFlushMappedMemoryRanges(VkDevice device = 0x2a635cf0, uint32_t memoryRangeCount = ...	0x2a635cf0	4	-	3500
29	// Recorded 17 commands to primary command buffer; VkCommandBuffer = 0x2a82d230, Vk...	0x2a82d230	-	-	3500
30	// Mapped memory update; VkDeviceMemory = 0x03945330, ranges = {375552 B @ 0}	0x03945330	-	-	3500
31	vkFlushMappedMemoryRanges(VkDevice device = 0x2a635cf0, uint32_t memoryRangeCount = ...	0x2a635cf0	1	-	3500
32	vkResetFences(VkDevice device = 0x2a635cf0, uint32_t fenceCount = 1u, VkFence pFences = {0x...	0x2a635cf0	2	-	3500
33	// Mapped memory update; VkDeviceMemory = 0x03945330, ranges = {65536 B @ 327680}	0x03945330	-	-	3500
34	vkQueueSubmit(VkQueue queue = 0x04387090, uint32_t submitCount = 1u, VkSubmitInfo pSu...	0x04387090	204	-	3500
35	vkBeginCommandBuffer(VkCommandBuffer commandBuffer = 0x2a82d230, VkCommandBuffer...	0x2a82d230	-	-	3500
36	vkCmdPipelineBarrier(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags ...	0x2a82d230	-	1	3500
37	vkCmdCopyBufferToImage(VkCommandBuffer commandBuffer = 0x2a82d230, VkBuffer srcBuff...	0x2a82d230	[illegible]	[illegible]	3500
38	vkCmdPipelineBarrier(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags ...	0x2a82d230	-	1	3500
39	vkCmdPipelineBarrier(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags ...	0x2a82d230	-	1	3500
40	vkCmdCopyBufferToImage(VkCommandBuffer commandBuffer = 0x2a82d230, VkBuffer srcBuff...	0x2a82d230	-	16	3500
41	vkCmdPipelineBarrier(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags ...	0x2a82d230	-	1	3500
42	vkCmdPipelineBarrier(VkCommandBuffer commandBuffer = 0x2a82d230, VkPipelineStageFlags ...	0x2a82d230	-	1	3500

Command buffer construction

Memory updates

Command buffer execution

Event Scrubber



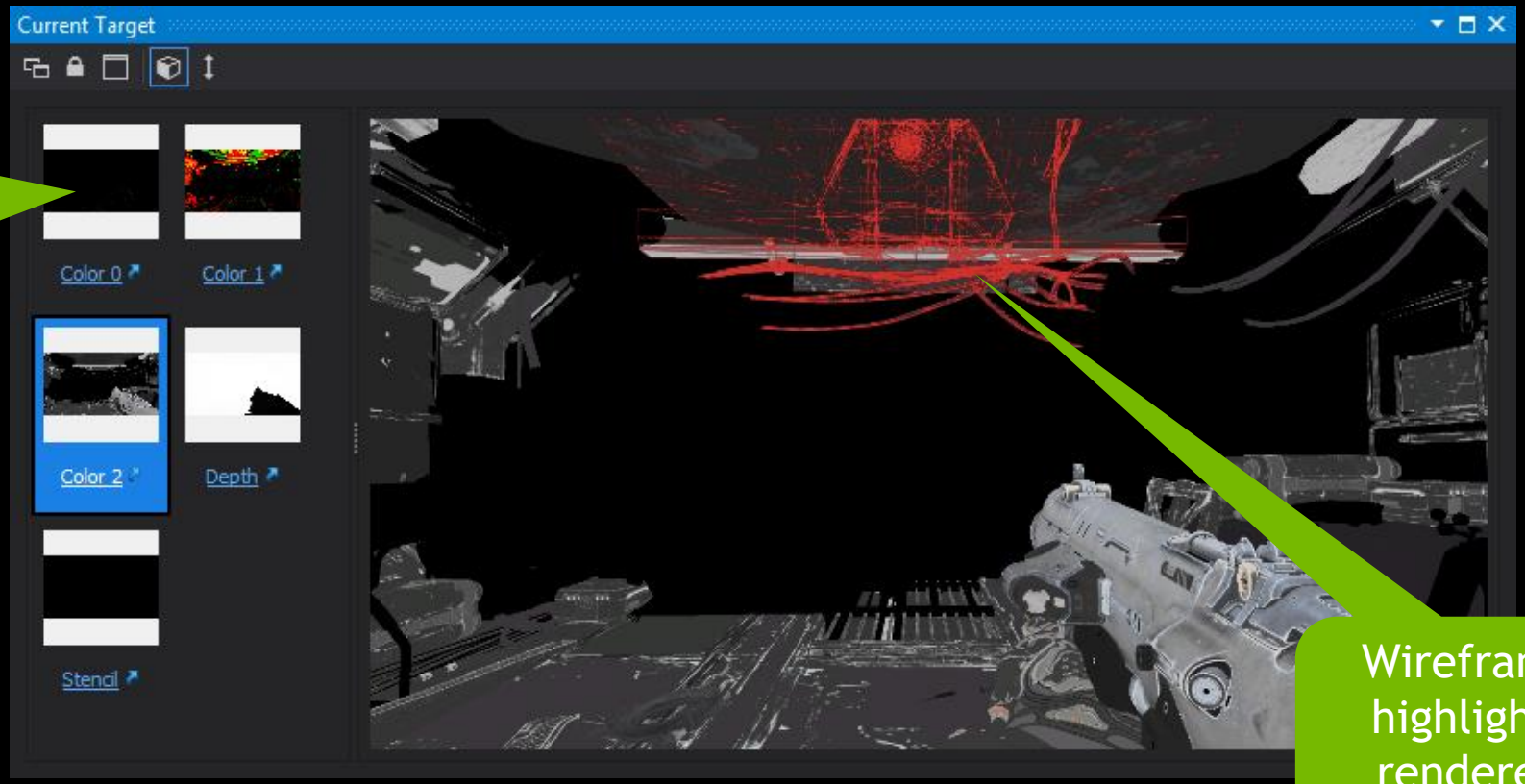
Multi-queue / multi-thread

State buckets & VK_EXT_debug_markers

Synchronization

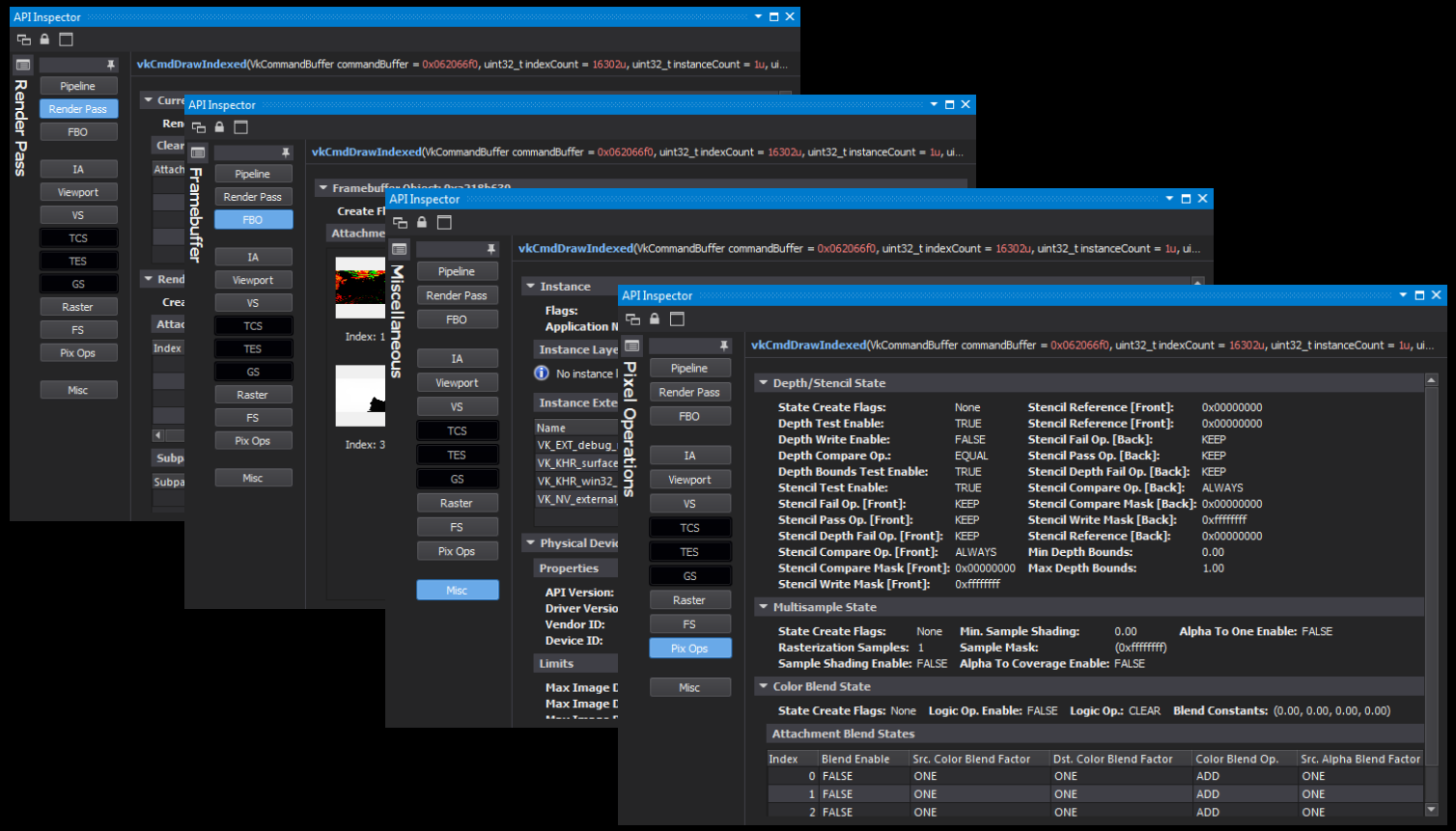
Current Target – Draws, Clears, & Blits

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

API Inspector

Vertex Shader: Pipeline | Render Pass | FBO | IA | Viewport | VS | TCS | TES | GS | Raster | FS | Pix Ops | Misc

vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x062066f0, uint32_t indexCount = 16302u, uint32_t instanceCount = 1u, uint32_t firstIndex = 0u, int32_t vertexOffset = 0, uint32_t firstInstance = 0u)

Shader Module: 0x7ae5c980

Create Flags: None **Code:** View SPIR-V **Cross-compiled GLSL:** View GLSL

Shader Interfaces

Inputs

Location	Type	Name
0	float4*	in_Position
1	float2*	in_TexCoord
2	float3*	in_PackedTnputs
12	float2*	in_VmtrTC
13	float4*	in_VmtrSB

Outputs

Location	Type	Name
2	float4*	vofi_TexCoord0
3	float4*	vofi_TexCoord1
4	float4*	vofi_TexCoord2
5	float4*	vofi_TexCoord3
6	float4*	vofi_TexCoord4
7	float4*	vofi_TexCoord5

Uniform & Storage Buffers

Set	Binding	Element	Descriptor Set	Descriptor Type	Uniform Type	Name	Buffer	Data
0	0	0	0x36f12500	UNIFORM_BUFFER_DYNAMIC	struct*	freqHigh_vertexUniforms	0x2ad54ca0 @ 1704448	{...}
0	0	0	0x36f12500	UNIFORM_BUFFER_DYNAMIC	float4	freqHigh_vertexUniforms.mvpmatrixx	0x2ad54ca0 @ 1704448	(1.00, 0.00, 0.00, 0.00)
0	0	0	0x36f12500	UNIFORM_BUFFER_DYNAMIC	float4	freqHigh_vertexUniforms.mvpmatrixy	0x2ad54ca0 @ 1704464	(0.09, 0.22, 0.97, 133.62)
0	0	0	0x36f12500	UNIFORM_BUFFER_DYNAMIC	float4	freqHigh_vertexUniforms.mvpmatrixz	0x2ad54ca0 @ 1704480	(1.77, -0.13, -0.14, -167.15)
0	0	0	0x36f12500	UNIFORM_BUFFER_DYNAMIC	float4	freqHigh_vertexUniforms.mvpmatrixw	0x2ad54ca0 @ 1704496	(-0.06, -0.97, 0.22, 244.99)
0	0	0	0x36f12500	UNIFORM_BUFFER_DYNAMIC	float4	freqHigh_vertexUniforms.modelmatrixx	0x2ad54ca0 @ 1704512	(-0.06, -0.97, 0.22, 247.94)
0	0	0	0x36f12500	UNIFORM_BUFFER_DYNAMIC	float4	freqHigh_vertexUniforms.modelmatrixy	0x2ad54ca0 @ 1704528	(0.09, 0.19, 0.98, -387.19)
0	0	0	0x36f12500	UNIFORM_BUFFER_DYNAMIC	float4	freqHigh_vertexUniforms.modelmatrixz	0x2ad54ca0 @ 1704544	(-0.11, -0.97, 0.20, -1535.83)

Push Constants

Images & Samplers

Texel Buffer Views

Names from SPIRV decorations

Uniform values

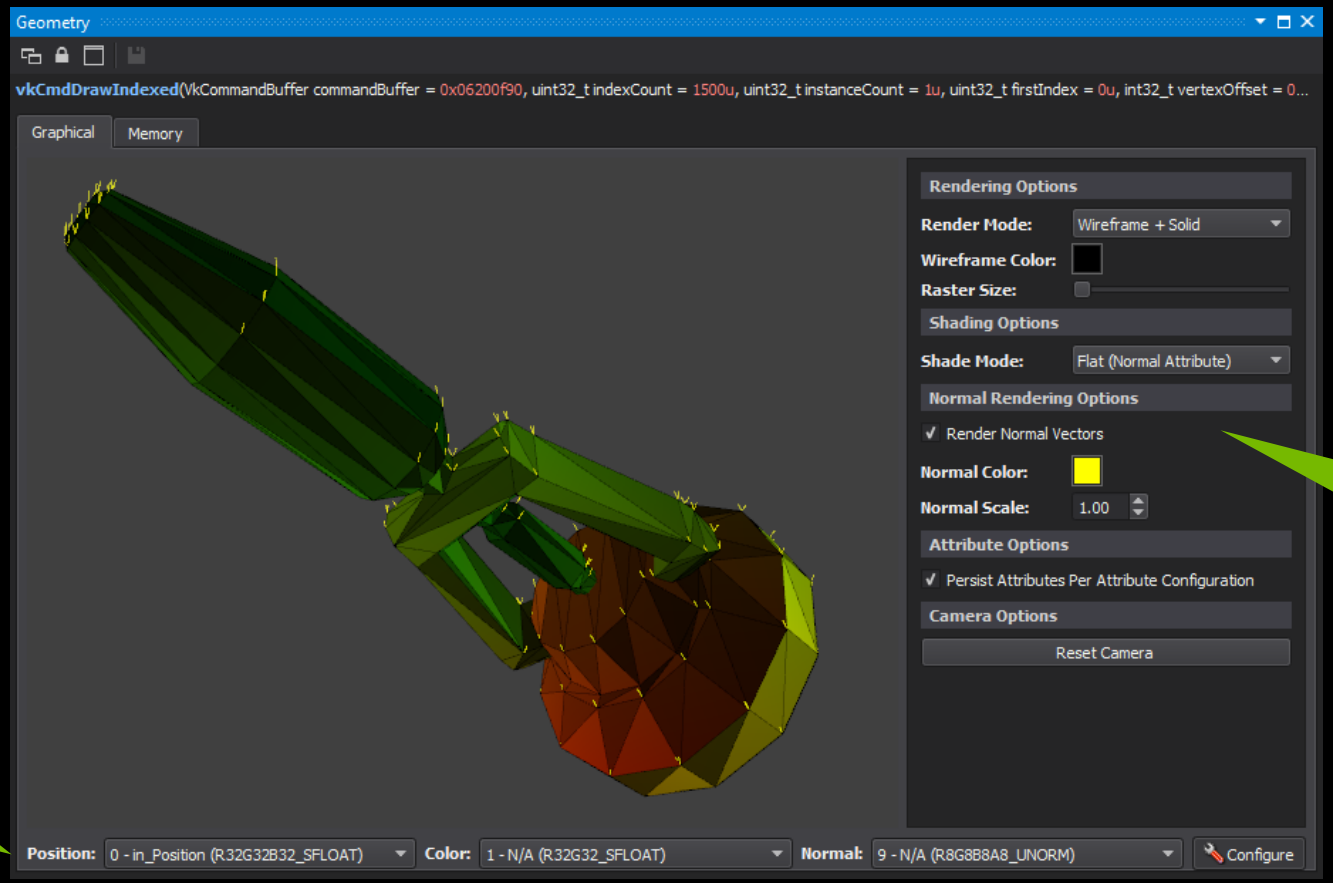
API Inspector

```
#version 450

layout(binding = 1, std140) uniform freqLow_vertexUniforms_ubo
{
    vec4 vertexstscalebias;
    vec4 vertexxyzscale;
    vec4 vertexxyzbias;
    vec4 globalvieworigin;
} freqLow_vertexUniforms;

layout(binding = 0, std140) uniform freqHigh_vertexUniforms_ubo
{
    vec4 mvpmatrixx;
    vec4 mvpmatrixy;
    vec4 mvpmatrixz;
    vec4 mvpmatrixw;
    vec4 modelmatrixx;
    vec4 modelmatrixy;
    vec4 modelmatrixz;
    vec4 mvpmatrixdeterminantsign;
} freqHigh_vertexUniforms;

layout(location = 2) out vec4 vofi_TexCoord0;
layout(location = 1) in vec2 in_TexCoord;
layout(location = 12) in vec2 in_VmtrTC;
layout(location = 0) in vec4 in_Position;
layout(location = 2) in vec3 in_PackedInputs;
layout(location = 3) out vec4 vofi_TexCoord1;
layout(location = 4) out vec4 vofi_TexCoord2;
layout(location = 5) out vec4 vofi_TexCoord3;
layout(location = 13) in vec4 in_VmtrSB;
layout(location = 7) out vec4 vofi_TexCoord5;
layout(location = 6) out vec4 vofi_TexCoord4;

uint asuint(float x)
```

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

SPIRV decorations for uniforms & interfaces if available

Geometry Viewer – Graphical



Control rendering modes

Control attributes

Geometry Viewer - Vertex Data

Geometry

vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x06200f90, uint32_t indexCount = 1500u, uint32_t instanceCount = 1u, uint32_t firstIndex = 0u, int32_t vertexOffset = 0...

Graphical | Memory

Index Buffer Order | Precision: 3

IB Offset	Index	Index + Base	0 - in_Position (R32G32B32_SFLOAT)			1 - N/A (R32G32_SFLO...		2 - N/A (R8G8B8A8_UNORM)			
			0	1	2	0	1	0	1	2	3
0	0	0	2.295	1.325	51.193	0.287	0.639	182	201	218	0
1	1	1	2.295	1.325	53.291	0.274	0.639	201	156	27	0
2	2	2	0.000	2.650	53.291	0.274	0.657	141	205	27	0
3	2	2	0.000	2.650	53.291	0.274	0.657	141	205	27	0
4	3	3	0.000	2.650	50.407	0.292	0.657	116	243	181	0
5	0	0	2.295	1.325	51.193	0.287	0.639	182	201	218	0
6	3	3	0.000	2.650	50.407	0.292	0.657	116	243	181	0
7	2	2	0.000	2.650	53.291	0.274	0.657	141	205	27	
8	4	4	-2.295	1.325	53.291	0.274	0.674	67	178	27	0
9	4	4	-2.295	1.325	53.291	0.274	0.674	67	178	27	0
10	5	5	-2.295	1.325	51.193	0.287	0.674	49	190	207	0
11	3	3	0.000	2.650	50.407	0.292	0.657	116	243	181	0
12	5	5	-2.295	1.325	51.193	0.287	0.674	49	190	207	0
13	4	4	-2.295	1.325	53.291	0.274	0.674	67	178	27	0
14	6	6	-2.295	-1.325	53.291	0.274	0.691	55	100	27	0
15	6	6	-2.295	-1.325	53.291	0.274	0.691	55	100	27	0
16	7	7	-2.295	-1.325	51.193	0.287	0.691	74	55	218	0
17	5	5	-2.295	1.325	51.193	0.287	0.674	49	190	207	0
18	7	7	-2.295	-1.325	51.193	0.287	0.691	74	55	218	0
19	6	6	-2.295	-1.325	53.291	0.274	0.691	55	100	27	0
20	8	8	-0.000	-2.650	53.291	0.274	0.709	115	51	27	0
21	8	8	-0.000	-2.650	53.291	0.274	0.709	115	51	27	0
22	9	9	-0.000	-2.650	50.407	0.292	0.709	141	13	181	0

Index buffer ordering

Vertex attribute values

Resource View

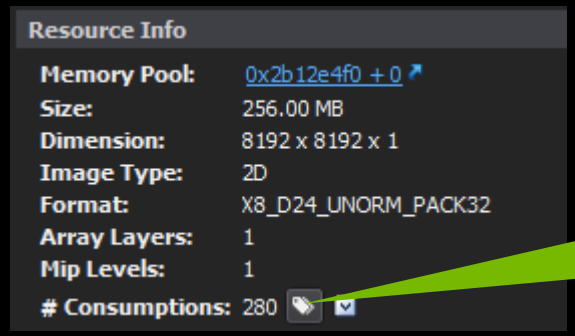
Thumbnail previews

Graphics / memory previews

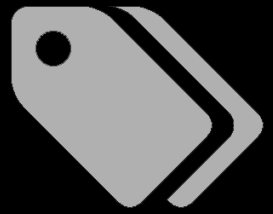
Resource information

Revision information

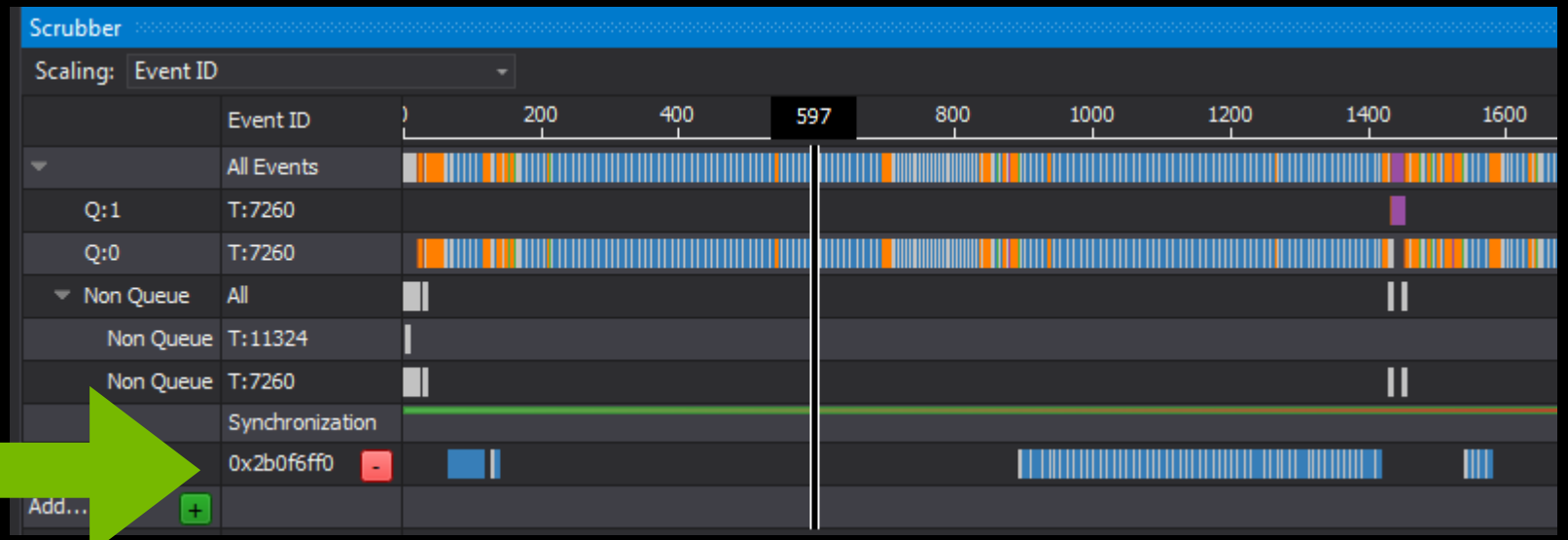
Resource View - Tagging



Tag a resource's consumption on the scrubber



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



Device Memory

Memory objects

Contained resources

Raw memory

Mini-map view

Device Memory

Memory Pools (90)

Name	Size	Entries	Flags
0x2b319510	384.00 MB	1	DEVICE_LOCAL
0x2b12e4f0	256.00 MB	1	DEVICE_LOCAL
0xab1e3130	128.00 MB	62	DEVICE_LOCAL
0x03bd54e0	128.00 MB	843	DEVICE_LOCAL
0x2b3196c0	128.00 MB	1	DEVICE_LOCAL
[illegible]e210b0	128.00 MB	1512	DEVICE_LOCAL
[illegible]	128.00 MB	2930	DEVICE_LOCAL
0x8514b430	128.00 MB	2685	DEVICE_LOCAL
0xab1e4f00	128.00 MB	2268	DEVICE_LOCAL
0xab1e4e70	128.00 MB	1999	DEVICE_LOCAL
0xab1e4de0	128.00 MB	2610	DEVICE_LOCAL
0x03b7d9f0	85.34 MB	1	DEVICE_LOCAL
0x2b12eb20	85.34 MB	1	DEVICE_LOCAL
0x03b71ea0	80.00 MB	1	HOST_VISIBLE | I
0x03b71d80	80.00 MB	1	HOST_VISIBLE | I
0x03bd6020	64.00 MB	200	HOST_VISIBLE | I
[illegible]	[illegible]	1	DEVICE_LOCAL
0xab1da1c0	64.00 MB	5183	HOST_VISIBLE | I
0x2b12efa0	48.00 MB	1	DEVICE_LOCAL
0x2b12f150	32.00 MB	1	DEVICE_LOCAL
0xab1e35b0	22.51 MB	1	DEVICE_LOCAL
0x2b3178f0	22.50 MB	1	DEVICE_LOCAL

Resources (843)

Offset	Type	Name	Size	Overlaps W
1024	Texture	0x527be250	6.00 KB	
7168	Texture	0x527bdf80	5.00 KB	
12288	Texture	0x527bccc0	87.50 KB	
102400	Texture	0x527bcae0	87.50 KB	
192512	Texture	0x529f9c20	23.50 KB	
217088	Texture	0x529f9a40	13.00 KB	
230400	Texture	0x529f9860	23.50 KB	
254976	Texture	0x529f8960	7.50 KB	
263168	Texture	0x529f8780	1024.00 B	
264192	Texture	0x529f85a0	1024.00 B	
265216	Texture	0x529f83c0	87.50 KB	
355328	Texture	0x529f81e0	1024.00 B	
356352	Texture	0x529f8000	343.50 KB	
708608	Texture	0x529f7e20	685.00 KB	
1410048	Texture	0x529f7c40	13.00 KB	
1423360	Texture	0x529f7a60	7.50 KB	
1431552	Texture	0x529f7880	7.50 KB	
1439744	Texture	0x529f76a0	685.00 KB	
2141184	Texture	0x529f74c0	1.34 MB	
3542016	Texture	0x529f72e0	13.00 KB	
3555328	Texture	0x529f7100	512.00 B	
3556352	Texture	0x529f6f20	173.00 KB	
[illegible]	[illegible]	[illegible]	343.00 KB	

Data

Offset: 1024 Columns: 4 Precision: 5 64-bit Int

Hexadecimal

Address	Data (Hash: 0xcb5e5ddd)	
0x0000000000000400	0000000000000000	0000000000000000 00
0x0000000000000420	0000000000000000	0000000000000000 00
0x0000000000000440	0000000000000000	0000000000000000 00
0x0000000000000460	0000000000000000	afff555500000001 2f
0x0000000000000480	d5fdbf2f00000823	55fdaf0b00011047 00
0x00000000000004a0	0000000000000000	0000000000000000 00
0x00000000000004c0	0000000000000000	0000000000000000 00
0x00000000000004e0	0000000000000000	0000000000000000 00
0x0000000000000500	00ff555500000825	0aff555500001048 00
0x0000000000000520	0000000000000000	0000000000000000 00
0x0000000000000540	0000000000000000	0000000000000000 00
0x0000000000000560	0aff55550001208c	00ff5555000128af 00
0x0000000000000580	d5aa000028b37a5f	55ab0200411a9b1f 55
0x00000000000005a0	0000000000000000	0000000000000000 00
0x00000000000005c0	0000000000000000	0000000000000000 00

Revisions

Revision 107 of 341

Resource Map

Resource

Descriptor Sets

Descriptor Sets

Descriptor Sets (765)

Set	Layout	Pool	Consumptions	Binding
0x36f11d20	0x8bf3ee80	0x2af13e10	8	-
0x36f11690	0x979c19d0	0x2af13e10	8	-
0x36f11000	0x8b85b0d0	0x2af13e10	8	-
0x36f10040	0x8b85ace0	0x2af13e10	8	-
0x36f17120	0x8bc07060	0x2af13e10	6	-
0x36f16d30	0x8bc07060	0x2af13e10	6	-
0x36f15ad0	0x87dce9f0	0x2af13e10	6	-
0x36f130d0	0xa185a500	0x2af13e10	6	-
0x36f12b90	0x8c0234f0	0x2af13e10	6	-
0x36f12a40	0x8bf2b210	0x2af13e10	6	-
0x36f12260	0x8bed2690	0x2af13e10	6	-
0x36f16fd0	0x8bc07060	0x2af13e10	4	-
0x36f16550	0x8bed2690	0x2af13e10	4	-
0x36f16160	0x87dce9f0	0x2af13e10	4	-
0x36f138b0	0x8bb3b	0x2af13e10	4	-
0x36f12110	0xa18	0x2af13e10	4	-
0x36f11fc0	0x	0x2af13e10	4	-
0x36f117e0		0x2af13e10	4	-
		0x2af13e10	4	-

Descriptor Pool: 0x2af13e10

Sets:		384 / 8192
Combined Image Samplers:		2802 / 262144
Storage Images:		201 / 65536
Uniform Buffers:		1181 / 65536
Storage Buffers:		478 / 65536
Uniform Buffers (Dynamic):		1535 / 65536

Descriptor Set: 0x36f10c10

Binding	Element	Type	Stages	Properties	Preview
					link
1	0	UNIFORM_BUFFER_DYNAMIC	FRAGMENT	Buffer: 0x2ad54ca0 Offset: 0 Range: 32	10110101 10110111 00101100 01001101 link
2	0	COMBINED_IMAGE_SAMPLER	FRAGMENT	Sampler: 0x03b2e930 Image View: 0xb451e430 Image Layout: SHADER_READ_ONLY_OPTIM...	link

Sampler: 0x03b2 0

Flags:		Address Mode U:	CLAMP_TO_EDGE	Anisotropy Enable:	FALSE	Min Lod:	0.00
Mag Filter:		Address Mode V:	CLAMP_TO_EDGE	Max Anisotropy:	1.00	Max Lod:	13.00
Min Filter:		Address Mode W:	REPEAT	Compare Enable:	FALSE	Border Color:	FLOAT_TRANSPARENT_BLACK
Mipmap Mode:		Mip Lod Bias:	0.00	Compare Op:	NEVER	Unnorm. Coordinates:	FALSE

Pool information

All descriptors objects with usage counts

Associated resources

Selected resource information

C/C++ Serialization – Save to Disk

Serialize Capture to C/C++

```
12
13  //--------------------------------------------------------------------------
14  // CreateResources004
15  //--------------------------------------------------------------------------
16  void CreateResources004()
17  {
18      // Create VkBuffer_uidof_11098
19      {
20          static VkBufferCreateInfo VkBufferCreateInfo_temp_1414[1] = { VkBufferCreateInfo{
21          /* sType = */ VK_STRUCTURE_TYPE_BUFFER_CREATE_INFO,
22          /* pNext = */ nullptr,
23          /* flags = */ VkBufferCreateFlags(0),
24          /* size = */ 33312ull,
25          /* usage = */ VkBufferUsageFlags(VK_BUFFER_USAGE_TRANSFER_DST_BIT | VK_BUFFER_USAGE_VERTEX_
26          /* sharingMode = */ VK_SHARING_MODE_EXCLUSIVE,
27          /* queueFamilyIndexCount = */ 0u,
28          /* pQueueFamilyIndices = */ nullptr} };
29          NV_THROW_IF(vkCreateBuffer(VkDevice_uidof_3, VkBufferCreateInfo_temp_1414 llptr, &VkBuff
30      }
31
32      // Create VkBuffer_uidof_11099
33      {
34          static VkBufferCreateInfo VkBufferCreateInfo_temp_1415[1] = { VkBuff Info{
35          /* sType = */ VK_STRUCTURE_TYPE_BUFFER_CREATE_INFO,
36          /* pNext = */ nullptr,
37          /* flags = */ VkBufferCreateFlags(0),
```

Human readable C/C++ code

Loop frame in insolation

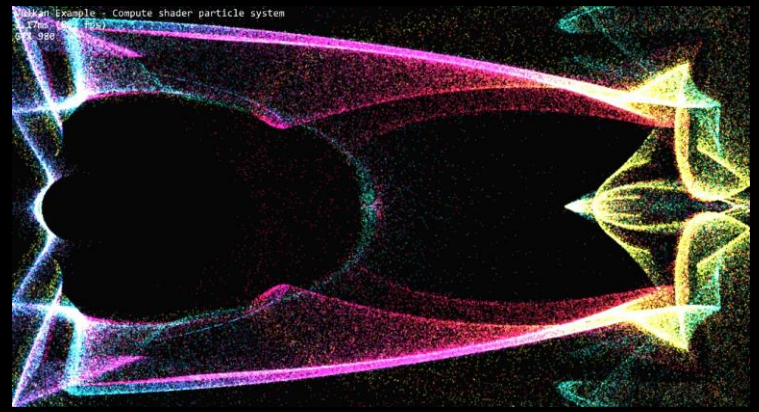
C/C++ Serialization – Challenges Solved

Portability

```
typedef struct VkMemoryAllocateInfo {
    VkStructureType     sType;
    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_EXT_shader_subgroup_ballot*

VK_EXT_shader_subgroup_vote*

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

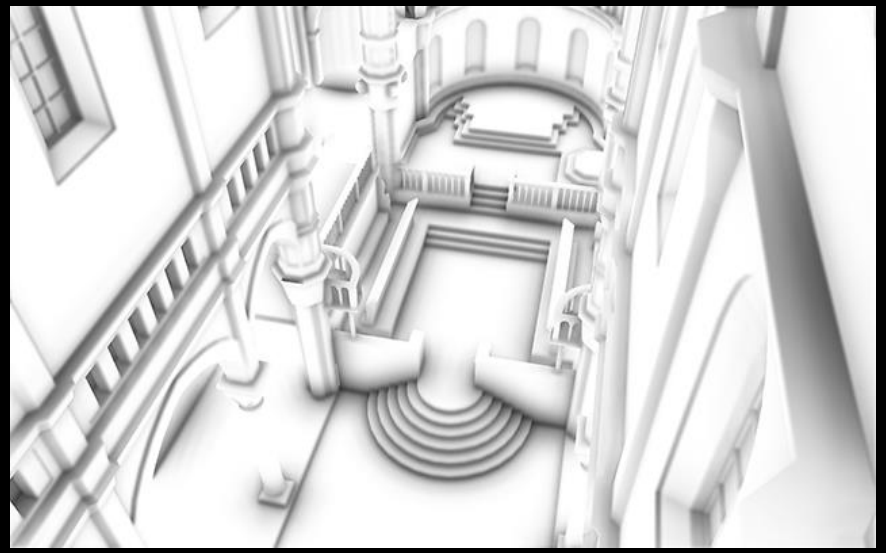
GameWorks HBAO+ on Vulkan

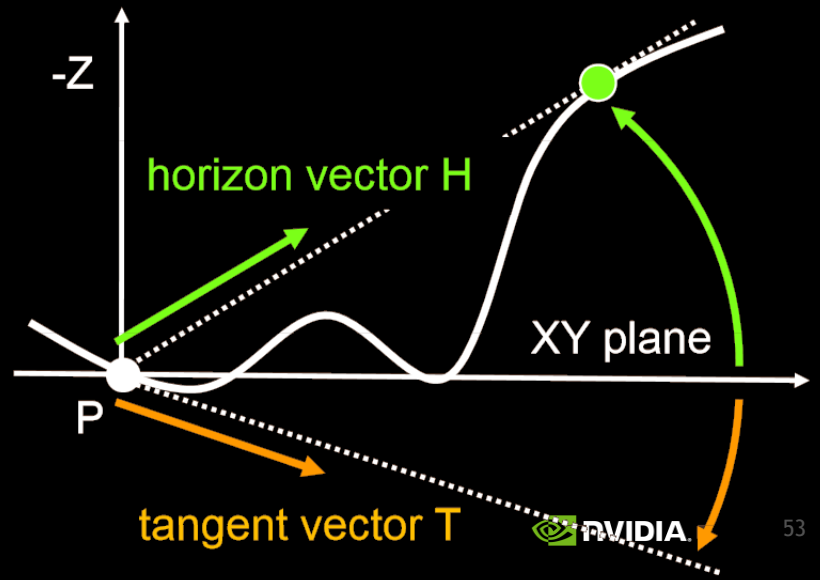
Nuno Subtil

NVIDIA

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

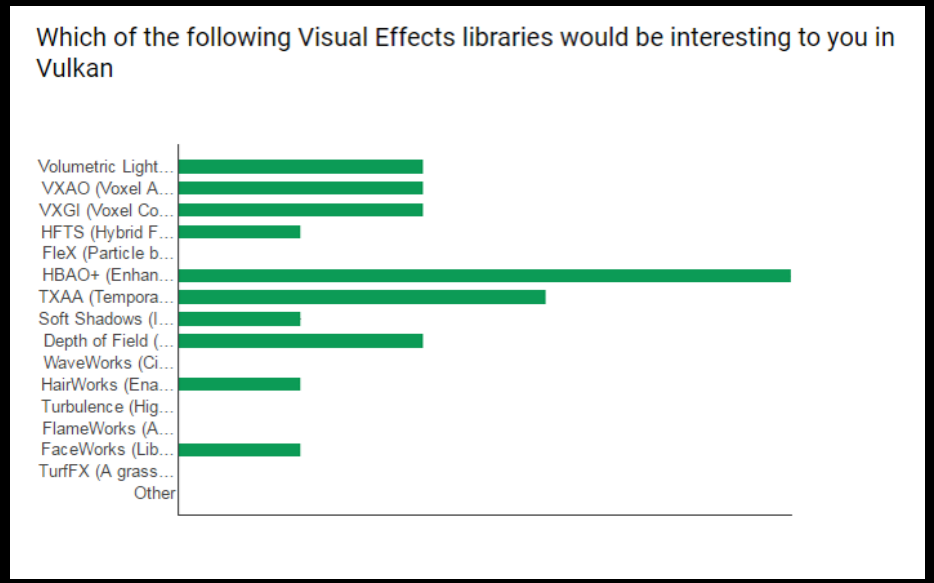




NVIDIA GAMEWORKS™

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

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

```
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

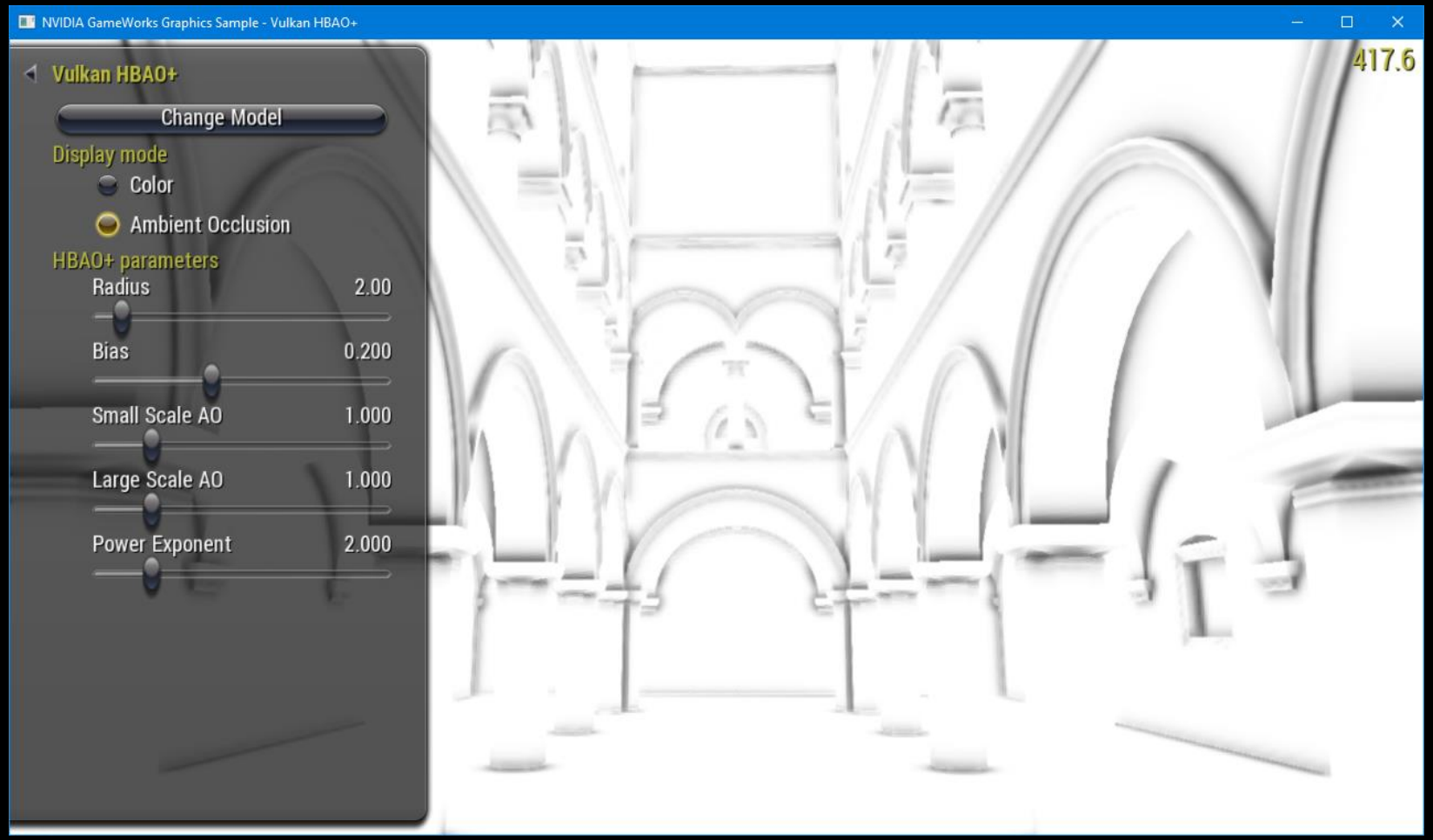
Synchronization

- 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


NVIDIA GameWorks Graphics Sample - Vulkan HBAO+
417.6
Vulkan HBAO+
Change Model
Display mode
Color
Ambient Occlusion
HBAO+ parameters
Radius 2.00
Bias 0.200
Small Scale AO 1.000
Large Scale AO 1.000
Power Exponent 2.000

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

NVIDIA Vulkan Update

Thank you for attending! ☺

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

NVIDIA