

NVIDIA Vulkan Update

Nuno Subtil, Sr. DevTech Engineer - GeForce

March 23, 2018



Booth #223 - South Hall

www.nvidia.com/GDC



NVIDIA Vulkan Update

- NVIDIA driver stack updates
- Libraries
- Tools

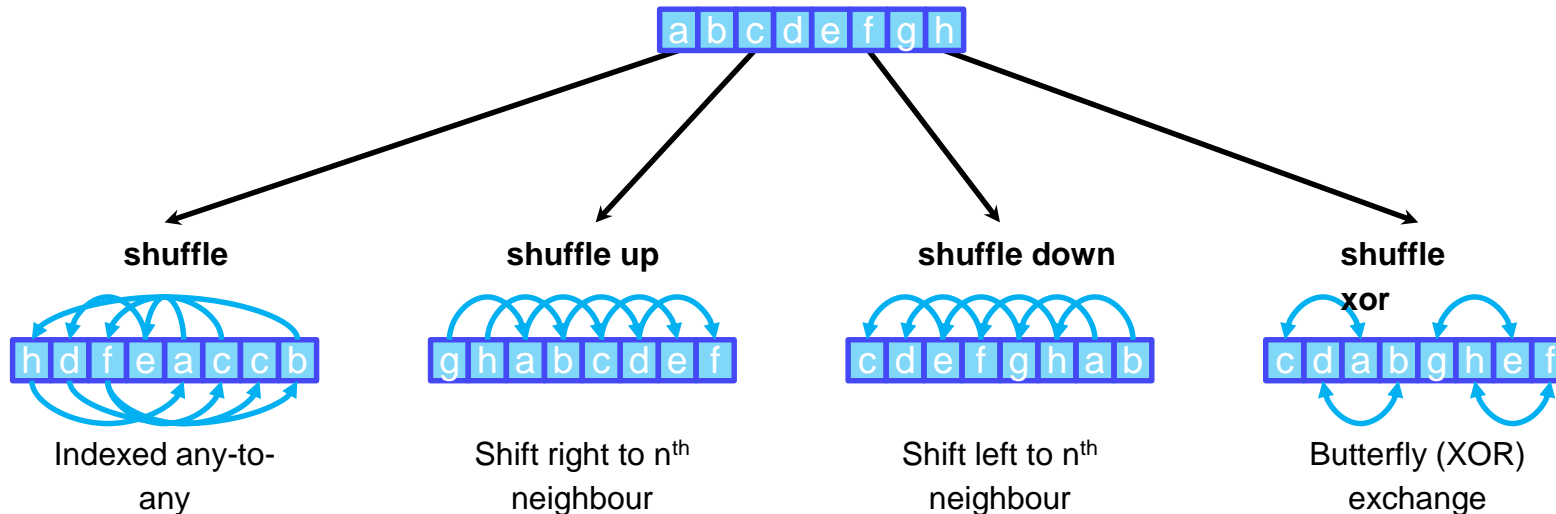


Vulkan 1.1!

- Released March 7th!
 - NVIDIA Vulkan 1.1 developer drivers out the same day
 - <https://developer.nvidia.com/vulkan-driver>
- Big ticket items:
 - Subgroups (SM6.0+)
 - Explicit Multi-GPU: support for AFR/SFR, VR applications
 - Multi-view
 - Cross-API / Cross-process synchronization primitives
 - Various “quality of life” improvements and other minor features

Subgroups

- Efficient cross-thread communication primitives
 - Exchange data between invocations of a warp/subgroup
 - Lower latency than shared memory
 - Can be used in graphics shaders
- Vulkan 1.1 supports SM6+ functionality and more



Meanwhile, at NVIDIA...

- EXT_sampler_filter_minmax:
 - Useful for voxelization algorithms, sparse textures.
- NV_fragment_coverage_to_color:
 - Output sample coverage information.
- EXT_conservative_raster
 - Cover all the pixels! Faster voxelization, raytraced shadow maps.
- EXT_depth_range_unrestricted
 - Depth can go to infinity! (But not beyond.)

NVIDIA keeps busy...

- EXT_post_depth_coverage
 - Control coverage from fragment stage.
- EXT_shader_viewport_index_layer (subset of NV_viewport_array2)
 - Pick render target layer in vertex shader
- EXT_sample_locations
 - Programmable sample locations.
- NV_fill_rectangle
 - Improved UI rendering performance, reduces fragment overhead for full-screen passes
- ...

Implementation Limits

- Developers sometimes run into our implementation-defined limits
 - Desire to have lots of descriptors bound comes up frequently
- NVIDIA has so far exposed our actual hardware limits
- We're now relaxing them
 - Implementation will handle spilling transparently where needed
 - Note that performance may degrade: existing thresholds were designed to avoid spills

```
maxPerStageDescriptorSamplers 4000 -> 1048576
maxPerStageDescriptorUniformBuffers 12 -> 15
maxPerStageDescriptorStorageBuffers 4096 -> 1048576
maxPerStageDescriptorSampledImages 16384 -> 1048576
maxPerStageDescriptorStorageImages 16384 -> 1048576
maxPerStageDescriptorInputAttachments 8 -> 1048576
maxPerStageResources 53268 -> 4294967295
maxDescriptorSetSamplers 4000 -> 1048576
maxDescriptorSetUniformBuffers 72 -> 90
maxDescriptorSetUniformBuffersDynamic 72 -> 15
maxDescriptorSetStorageBuffers 4096 -> 1048576
maxDescriptorSetSampledImages 98304 -> 1048576
maxDescriptorSetStorageImages 98304 -> 1048576
maxDescriptorSetStorageImages 8 -> 1048576
```

Shader Compiler Improvements

Original compiler path



- Original bring-up path for driver
- Leveraged NVIDIA's OpenGL shader compiler (decade+ of refinements)

Shader Compiler Improvements

Original compiler path



- Original bring-up path for driver
- Leveraged NVIDIA's OpenGL shader compiler (decade+ of refinements)
- However...
 - SPIR-V is not GLSL, translation confused the optimizer
 - Many pathological edge cases for shader performance
 - Slow compile times, high memory usage

Shader Compiler Improvements

Brand new compiler stack!



- Very simple translation between SPIR-V and NVVM (LLVM-based representation)
- Leverages modern compiler improvements in LLVM
- NVVM compiler stack shared across DX12 and Vulkan

Shader Compiler Improvements

What to expect?

- Faster shader compilation --- **3x speedup** on average
- Roughly **~50% reduction** in memory footprint
- Stable runtime shader performance, **less pathological cases**
- Expected to ship with R396 drivers (~1 month away)

- VK_NV_glsl_shader being deprecated
 - New compiler does not speak GLSL
 - Extension will be disabled soon after compiler transition

Best Practices

- Not many anti-patterns in shipping applications
 - Vulkan was designed to avoid such things --- seems to be working so far!
- Biggest concern: **use dedicated allocations** for large resources
 - Improves stability when under memory pressure
 - Can be faster in certain cases (and will never be slower)
 - Either flavor (KHR_dedicated_allocation/NV_dedicated_allocation) will work
 - Now core in Vulkan 1.1

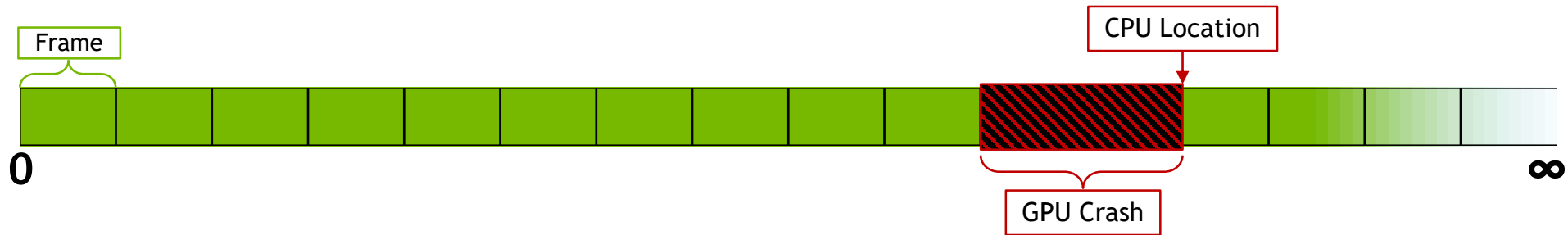
 Display driver stopped responding and has recovered  
Display driver NVIDIA Windows Kernel Mode Driver, Version stopped
responding and has successfully recovered.

How do I debug this?

 Display driver stopped responding and has recovered  
Display driver NVIDIA Windows Kernel Mode Driver, Version stopped
responding and has successfully recovered.

Debugging GPU crashes

- i. Crash detected based on error code from API (CPU)
- ii. Crash happened sometime in the last N frames of GPU commands...
- iii. CPU call stack is likely a red-herring



Not useful for debugging!

NVIDIA Aftermath

Post-mortem GPU Crash Debugging

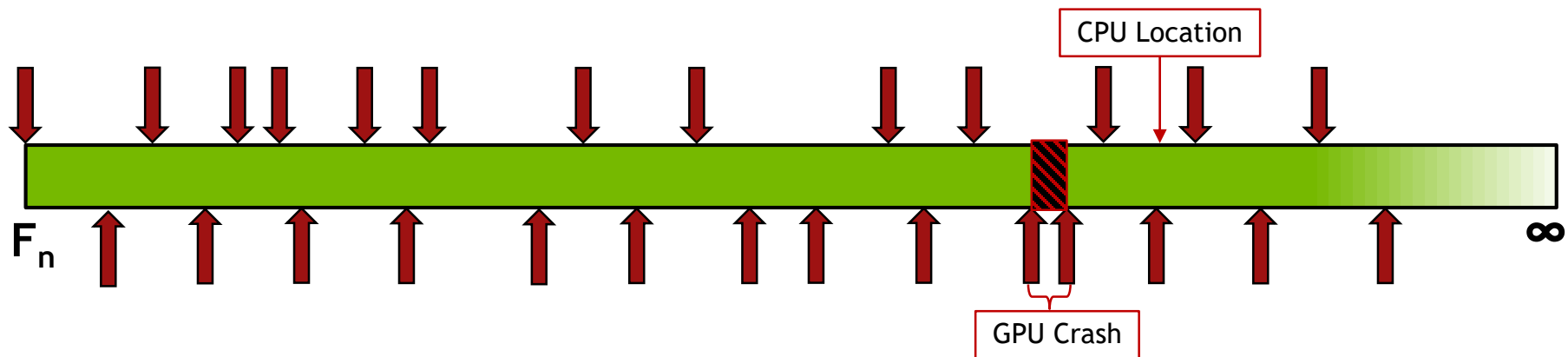
- Insert lightweight markers in the command stream
 - Can stash arbitrary app-specific data behind a marker
 - Designed such that performance impact is negligible, yet highly flexible
- After device lost, read back last marker value that the GPU executed

Debug Instrumentation with Aftermath

Checkpoints: Narrow in on GPU crash location WRT to command stream

Example:

- i. Game inserts user-defined markers in the command stream (CPU)
- ii. GPU signals each checkpoint once reached
- iii. Last marker reached indicates GPU crash location



Aftermath for Vulkan

- Device checkpoints available soon
 - Same functionality as DX12 version
 - <https://developer.nvidia.com/vulkan-driver>
- More features to come later



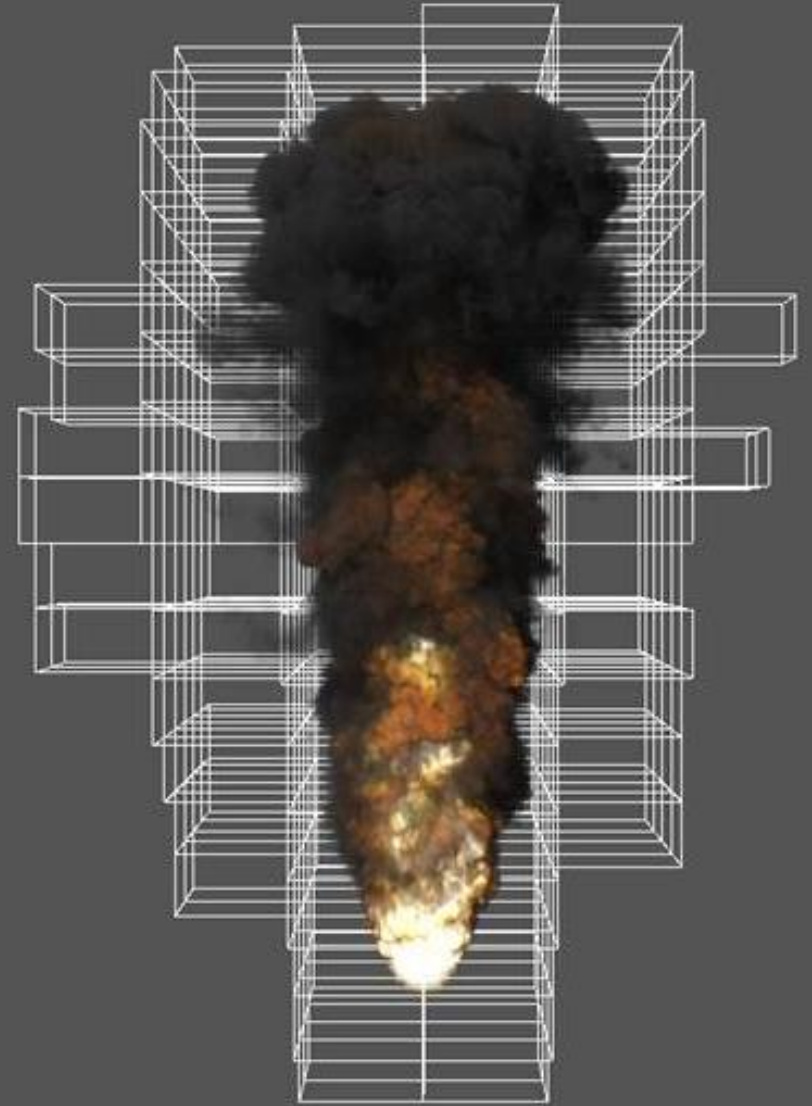
```
// VK_NV_device_diagnostic_checkpoints
typedef struct VkCheckpointDataNV {
    VkStructureType sType;
    const void* pNext;
    VkBool32      checkpointTopValid;
    void*         pCheckpointTop;
    VkBool32      checkpointBottomValid;
    void*         pCheckpointBottom;
} VkCheckpointDataNV;
```

```
void vkCmdSetCheckpointNV(
    VkCommandBuffer commandBuffer,
    const void* pCheckpointData);
```

```
VkResult vkGetCheckpointDataNV(
    VkQueue queue,
    VkCheckpointDataNV* pCheckpointData);
```

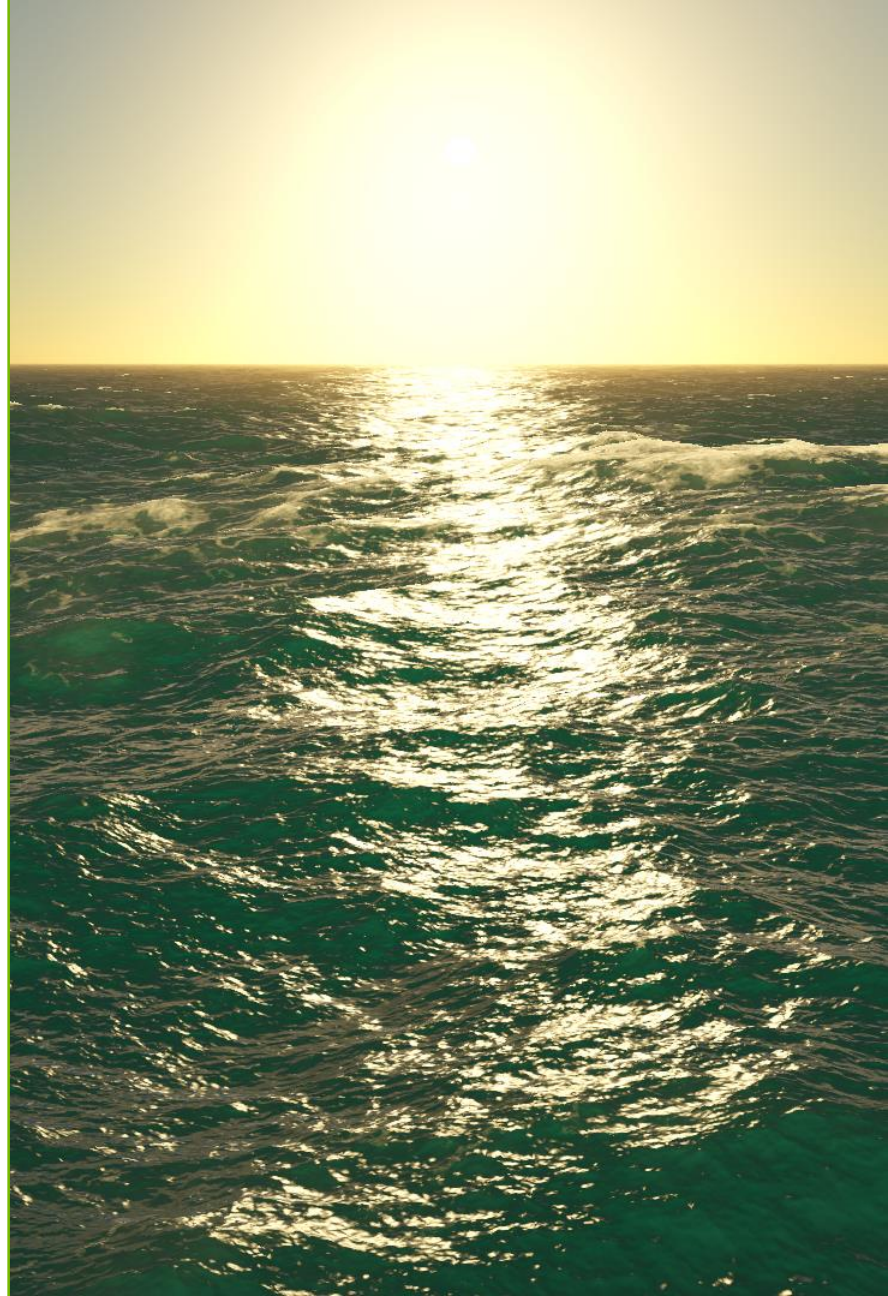
Flow

- Adaptive sparse voxel smoke/fire
- Vulkan support (in addition to DX11/DX12)
- New sparse framework, grid boundary restrictions removed
- Linux support
- Release coming soon



WaveWorks

- Cinematic-quality ocean simulation for interactive applications
- Rebuilt on cross-API abstraction layer
- Vulkan and DX11 now functional
- Linux support
- Lots of new features, release planned for later this year



Devtools Vulkan Update

Kyle Spagnoli

March 23rd 2018



Booth #223 - South Hall

www.nvidia.com/GDC



Introduction

- Nsight Graphics 1.0
- Vulkan specific features
- Vulkan specific road map

Nsight Graphics 1.0

- New standalone profiling, debugging, and analysis tool
- Builds upon technologies in Nsight Visual Studio Edition
- Vulkan, D3D11/12, OpenGL
- **Available now!**



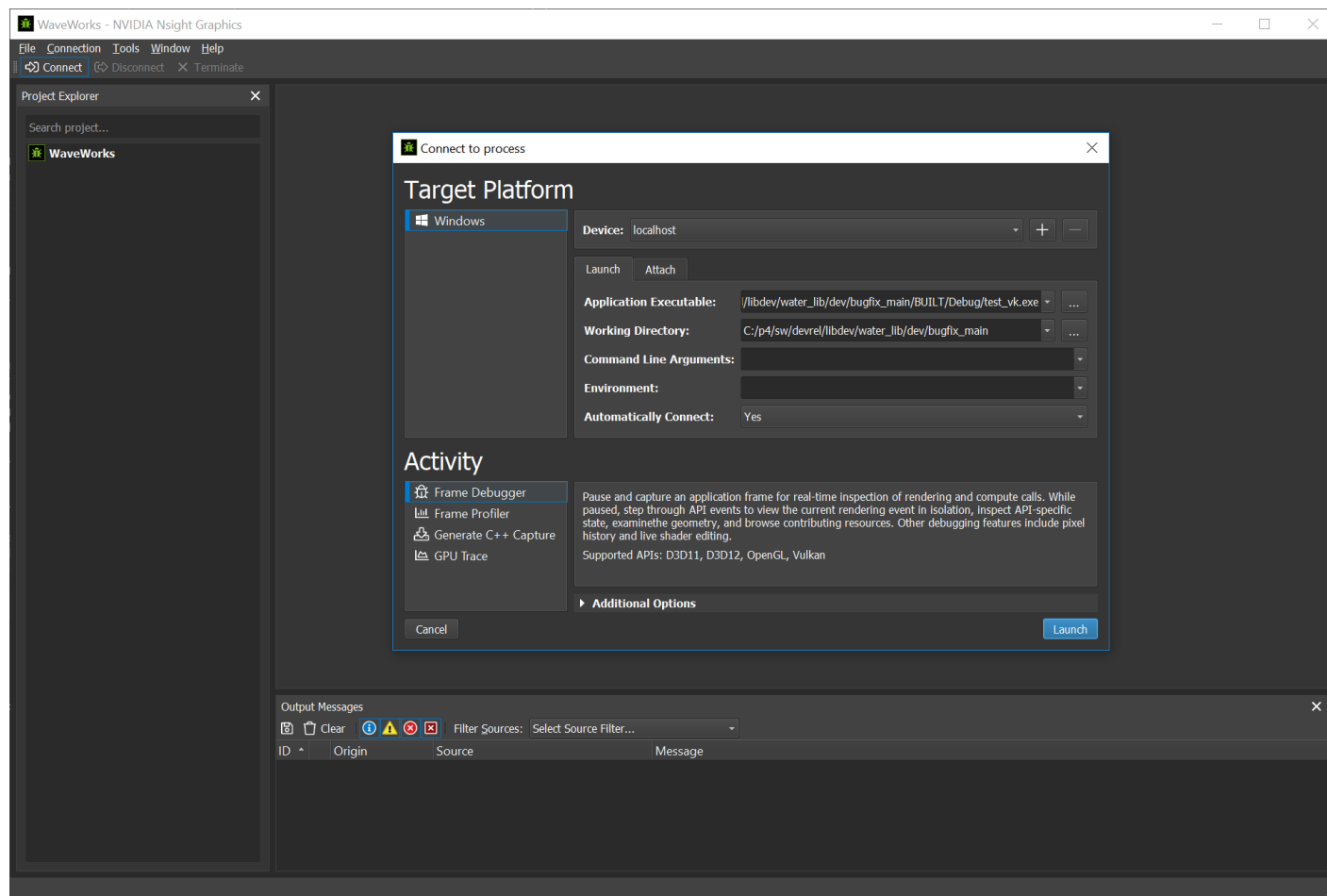
More information at “Beyond Performance: Introducing NVIDIA’s New Graphics Debugger”

NVIDIA.com > Developers > GameWorks > Tools

<https://developer.nvidia.com/nsight-graphics>

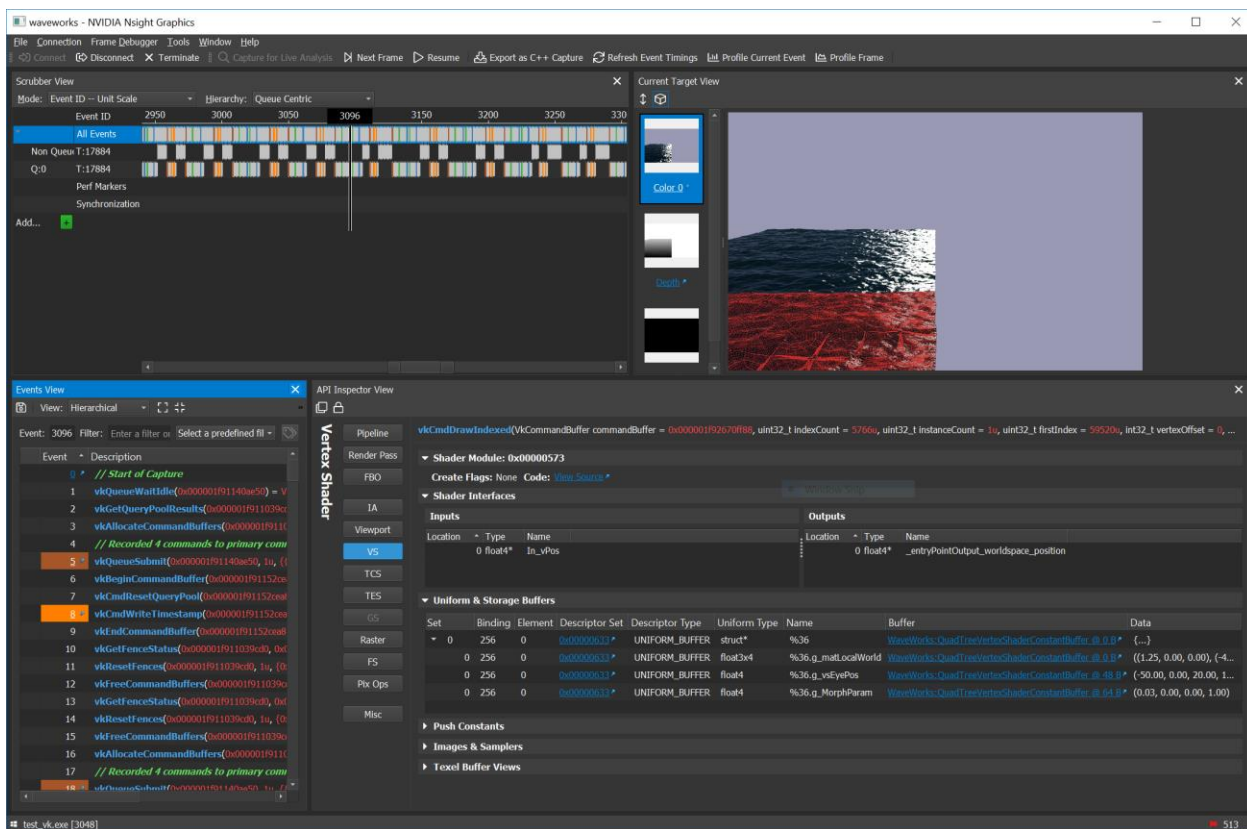
Nsight Graphics 1.0

- Independent from Visual Studio
- Streamlined launch experience
- Targeted activities
 - Frame Debugging
 - Profiling
 - C++ Capture Export
 - Tracing



Frame Debugger

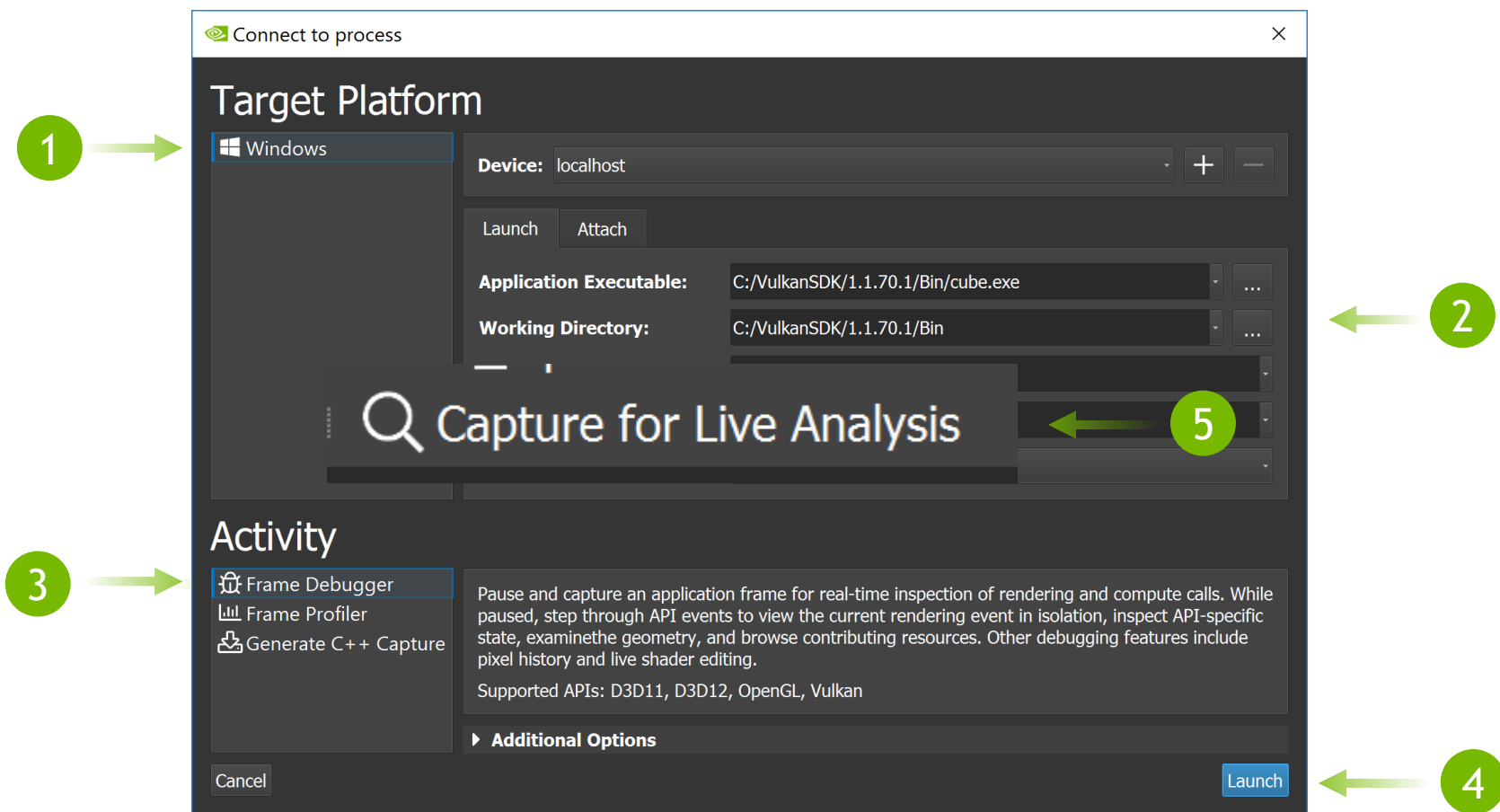
Frame Debugger - Overview



- Pause an application for live “online” analysis
- Multiple sub-windows offering different tools
- Scrub through important events
- Examine mid-frame state

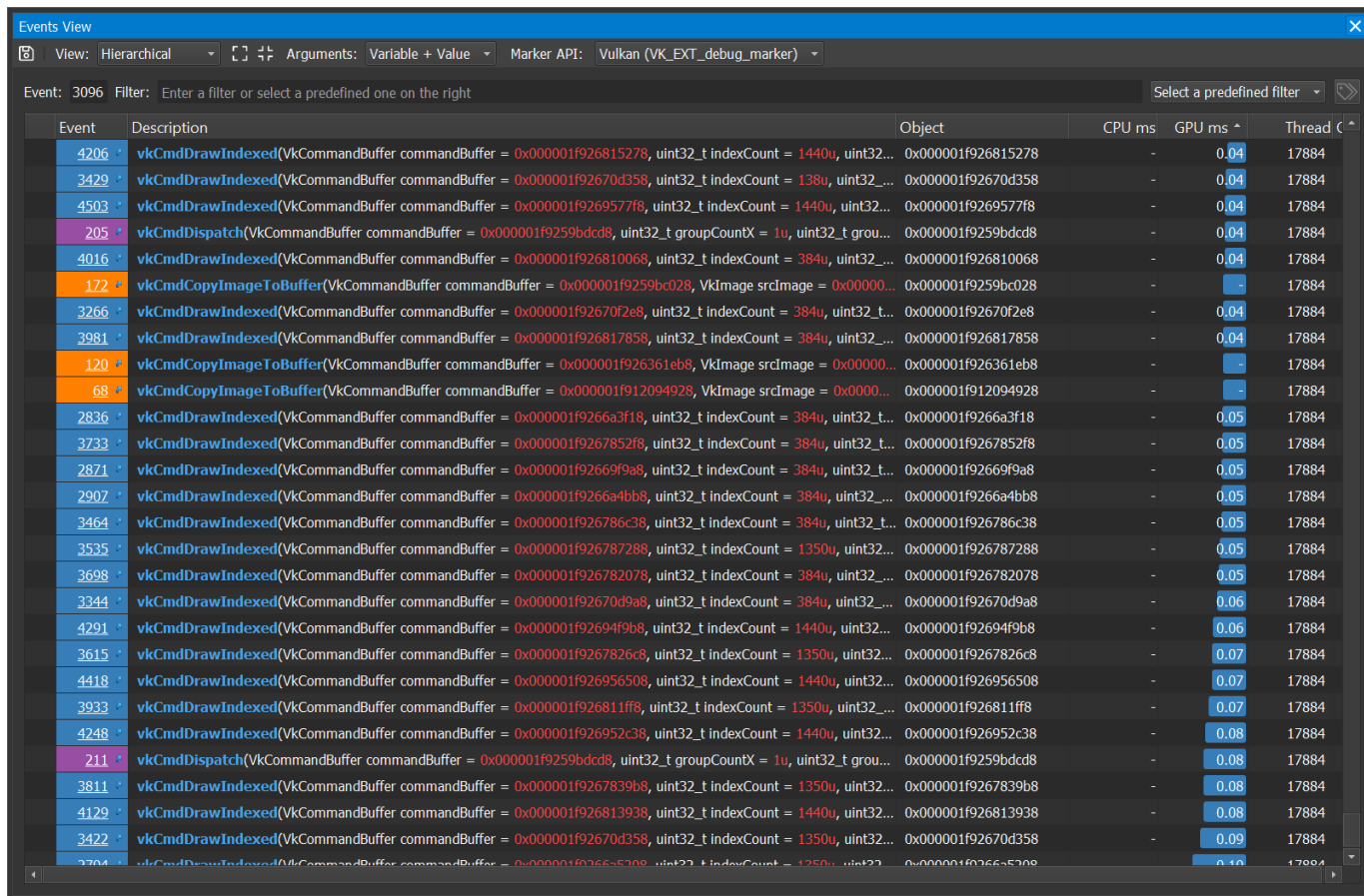
NOTE: All screenshots from WaveWorks for Vulkan

Frame Debugger - Getting Started



Frame Debugger - Event View

- Captured event stream
- Organize by command buffer, thread, queue
- Powerful filter, sort, and search systems
- Hierarchical range markers via VK_EXT_debug_marker
- Object names via VK_EXT_debug_marker

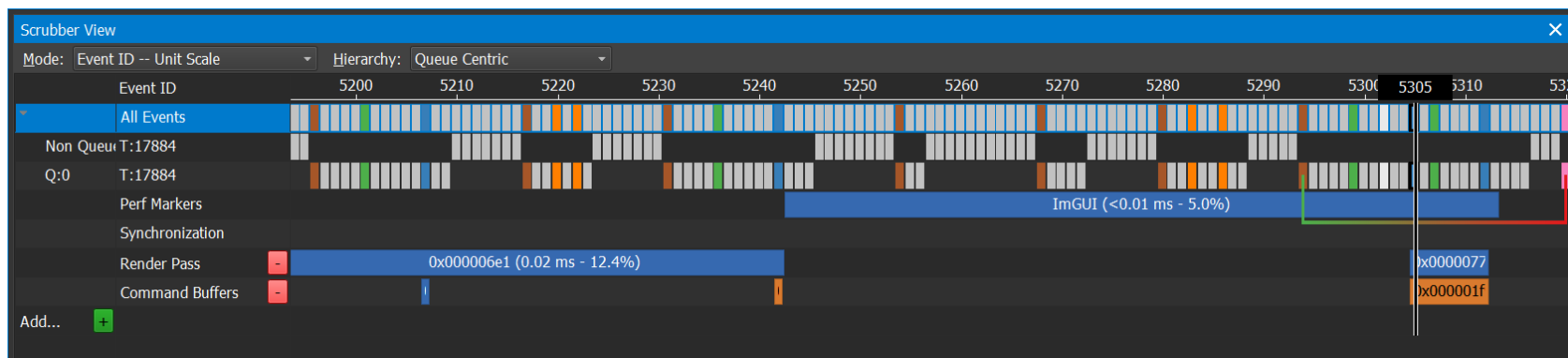


The screenshot displays the 'Events View' window of a frame debugger. The window has a blue header bar with the title 'Events View' and a close button. Below the header, there are tabs for 'View: Hierarchical', 'Arguments: Variable + Value', and 'Marker API: Vulkan (VK_EXT_debug_marker)'. A search bar is present with the text 'Event: 3096 Filter: Enter a filter or select a predefined one on the right'. A dropdown menu on the right says 'Select a predefined filter'. The main area is a table with the following columns: Event, Description, Object, CPU ms, GPU ms, and Thread. The table contains a list of Vulkan events, including vkCmdDrawIndexed, vkCmdDispatch, vkCmdCopyImageToBuffer, and vkCmdDrawIndexed. Each row has a small icon to the left of the event name, and the GPU ms column has a small blue bar indicating the duration of the event.

Event	Description	Object	CPU ms	GPU ms	Thread
4206	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926815278, uint32_t indexCount = 1440u, uint32_t...	0x000001f926815278	-	0.04	17884
3429	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f92670d358, uint32_t indexCount = 138u, uint32_t...	0x000001f92670d358	-	0.04	17884
4503	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f9269577f8, uint32_t indexCount = 1440u, uint32_t...	0x000001f9269577f8	-	0.04	17884
205	vkCmdDispatch(VkCommandBuffer commandBuffer = 0x000001f9259bdc8, uint32_t groupCountX = 1u, uint32_t grou...	0x000001f9259bdc8	-	0.04	17884
4016	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926810068, uint32_t indexCount = 384u, uint32_t...	0x000001f926810068	-	0.04	17884
172	vkCmdCopyImageToBuffer(VkCommandBuffer commandBuffer = 0x000001f9259bc028, VkImage srcImage = 0x000000...	0x000001f9259bc028	-	-	17884
3266	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f92670f2e8, uint32_t indexCount = 384u, uint32_t...	0x000001f92670f2e8	-	0.04	17884
3981	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926817858, uint32_t indexCount = 384u, uint32_t...	0x000001f926817858	-	0.04	17884
120	vkCmdCopyImageToBuffer(VkCommandBuffer commandBuffer = 0x000001f926361eb8, VkImage srcImage = 0x000000...	0x000001f926361eb8	-	-	17884
68	vkCmdCopyImageToBuffer(VkCommandBuffer commandBuffer = 0x000001f912094928, VkImage srcImage = 0x000000...	0x000001f912094928	-	-	17884
2836	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f9266a3f18, uint32_t indexCount = 384u, uint32_t...	0x000001f9266a3f18	-	0.05	17884
3733	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f9267852f8, uint32_t indexCount = 384u, uint32_t...	0x000001f9267852f8	-	0.05	17884
2871	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f92669f9a8, uint32_t indexCount = 384u, uint32_t...	0x000001f92669f9a8	-	0.05	17884
2907	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f9266a4bb8, uint32_t indexCount = 384u, uint32_t...	0x000001f9266a4bb8	-	0.05	17884
3464	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926786c38, uint32_t indexCount = 384u, uint32_t...	0x000001f926786c38	-	0.05	17884
3535	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926787288, uint32_t indexCount = 1350u, uint32_t...	0x000001f926787288	-	0.05	17884
3698	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926782078, uint32_t indexCount = 384u, uint32_t...	0x000001f926782078	-	0.05	17884
3344	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f92670d9a8, uint32_t indexCount = 384u, uint32_t...	0x000001f92670d9a8	-	0.06	17884
4291	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f92694f9b8, uint32_t indexCount = 1440u, uint32_t...	0x000001f92694f9b8	-	0.06	17884
3615	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f9267826c8, uint32_t indexCount = 1350u, uint32_t...	0x000001f9267826c8	-	0.07	17884
4418	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926956508, uint32_t indexCount = 1440u, uint32_t...	0x000001f926956508	-	0.07	17884
3933	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926811ff8, uint32_t indexCount = 1350u, uint32_t...	0x000001f926811ff8	-	0.07	17884
4248	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926952c38, uint32_t indexCount = 1440u, uint32_t...	0x000001f926952c38	-	0.08	17884
211	vkCmdDispatch(VkCommandBuffer commandBuffer = 0x000001f9259bdc8, uint32_t groupCountX = 1u, uint32_t grou...	0x000001f9259bdc8	-	0.08	17884
3811	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f9267839b8, uint32_t indexCount = 1350u, uint32_t...	0x000001f9267839b8	-	0.08	17884
4129	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f926813938, uint32_t indexCount = 1440u, uint32_t...	0x000001f926813938	-	0.08	17884
3422	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f92670d358, uint32_t indexCount = 1350u, uint32_t...	0x000001f92670d358	-	0.09	17884
3784	vkCmdDrawIndexed(VkCommandBuffer commandBuffer = 0x000001f9266a5308, uint32_t indexCount = 1350u, uint32_t...	0x000001f9266a5308	-	0.10	17884

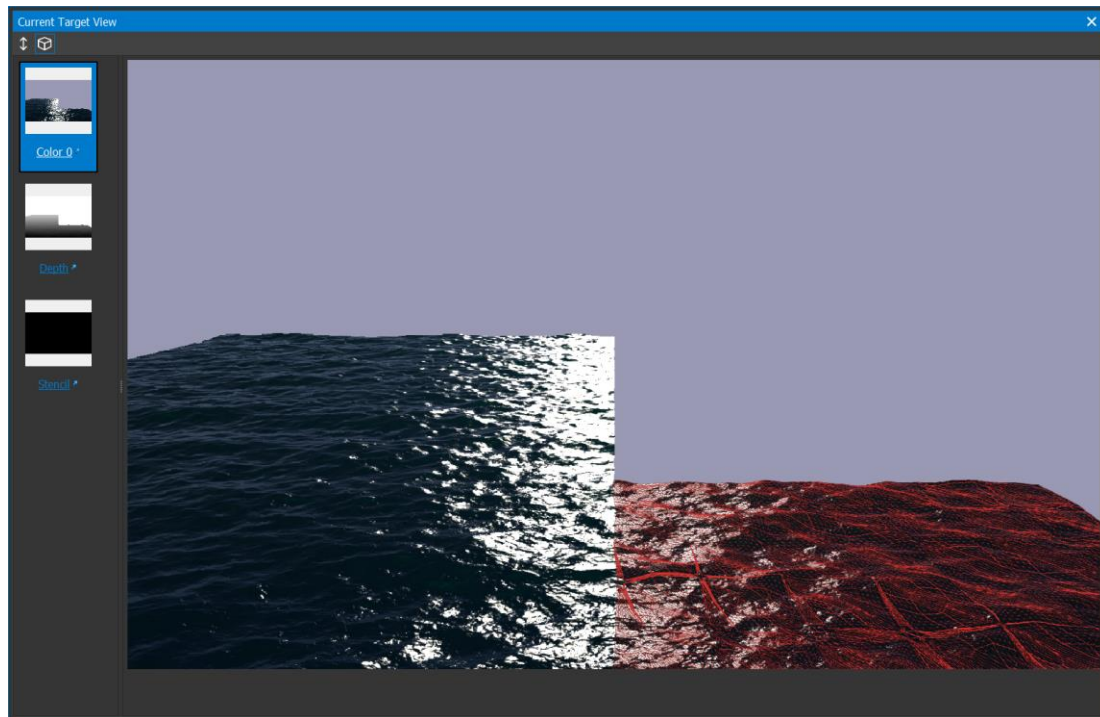
Frame Debugger - Scrubber

- Timeline view of your scene
 - Event, GPU, or CPU time scale
- Multi-queue and multi-thread aware
- Range markers via VK_EXT_debug_marker and heuristics
- Synchronization indicators

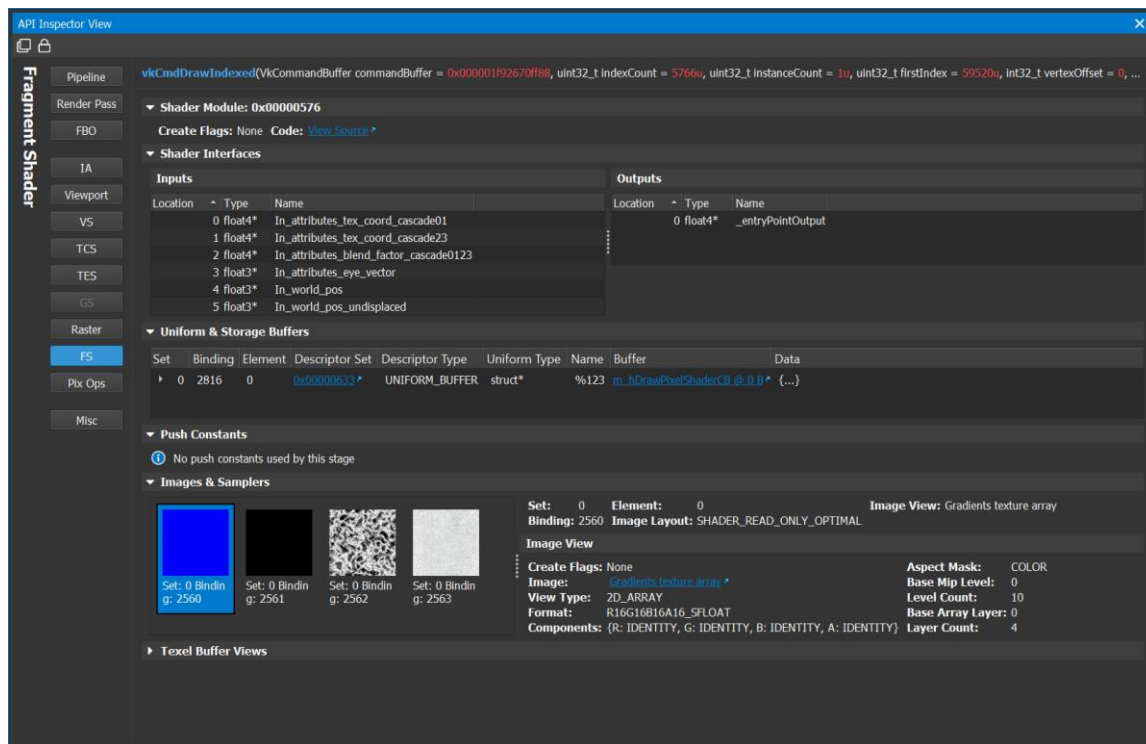


Frame Debugger - Current Target

- Visualize incremental per-draw state of your render targets
- Color, depth, stencil
- Optional overlays
- Pixel history
 - More later...



Frame Debugger - API Inspector



- Per-pipeline stage information
- See bound descriptor sets and their associated data
- Images with thumbnails
- Uniform buffers with data at-a-glance
- Reflection information if SPIRV has annotations

Frame Debugger - Descriptor Views

- Listing of all descriptors used in scene
- Sort and search by pools and layouts
- At a glance contents of each descriptor

The screenshot displays the 'Texture and Sampler Pools View' in a frame debugger. It is divided into two main panes.

Left Pane: Descriptor Sets

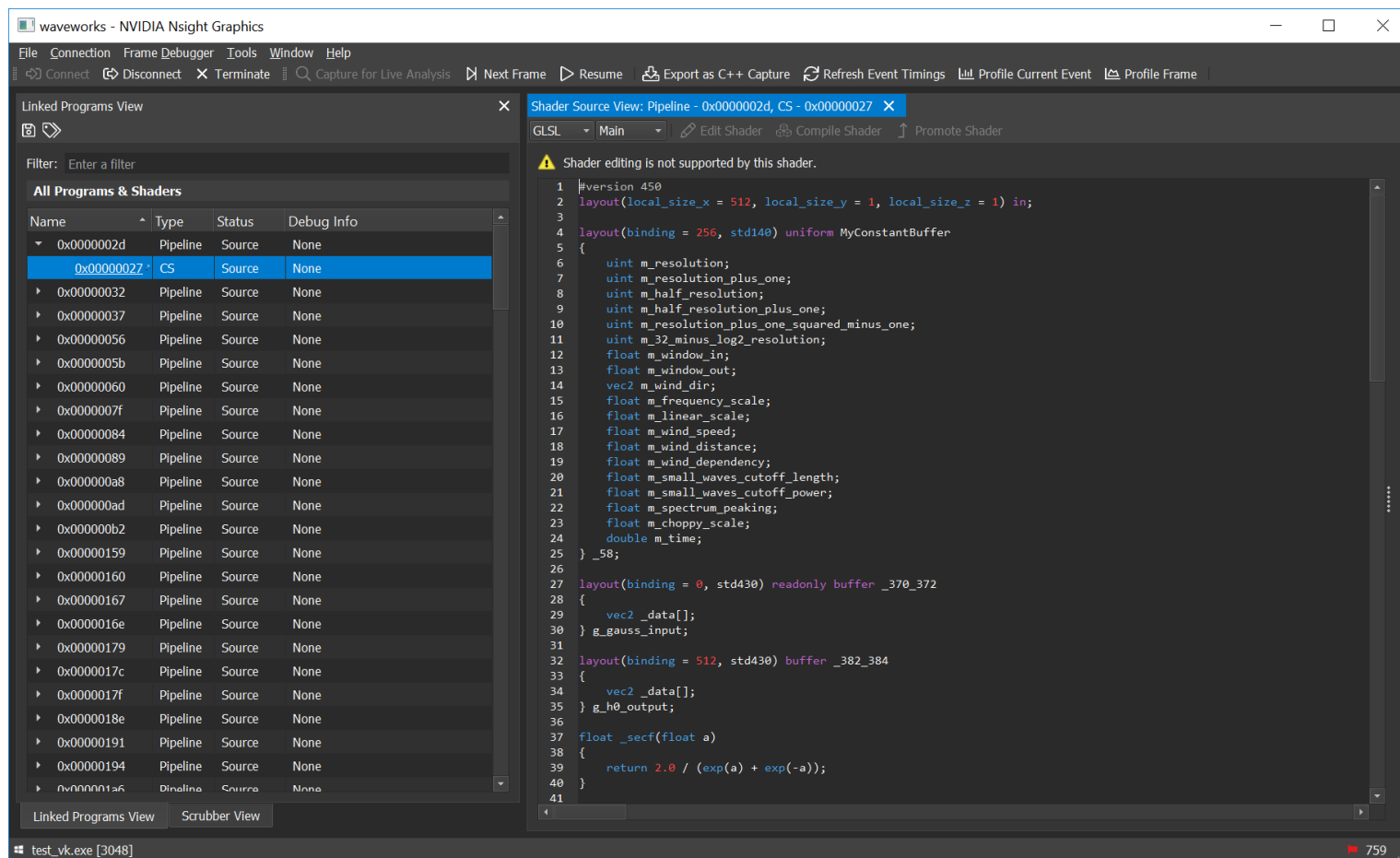
Set	Layout	Pool	Consumptions
0x00000779	0x00000775	0x00000778	1
0x00000633	0x0000062f	0x00000632	62
0x0000054e	0x00000546	0x0000054d	1
0x0000054c	0x00000543	0x0000054b	1
0x0000054a	0x00000540	0x00000549	1
0x00000539	0x00000531	0x00000538	1
0x00000537	0x0000052e	0x00000536	1
0x00000535	0x0000052b	0x00000534	1
0x00000524	0x0000051c	0x00000523	1
0x00000522	0x00000519	0x00000521	1
0x00000520	0x00000516	0x0000051f	1
0x0000050f	0x00000507	0x0000050e	1
0x0000050d	0x00000504	0x0000050c	1
0x0000050b	0x00000501	0x0000050a	1
0x000004fa	0x000004f2	0x000004f9	1
0x000004f8	0x000004ef	0x000004f7	1
0x000004f6	0x000004ec	0x000004f5	1
0x000004e5	0x000004dd	0x000004e4	1
0x000004e3	0x000004da	0x000004e2	1
0x000004e1	0x000004d7	0x000004e0	1
0x000004d0	0x000004c8	0x000004cf	1
0x000004ce	0x000004c5	0x000004cd	1
0x000004cc	0x000004c2	0x000004cb	1
0x000004bb	0x000004b3	0x000004ba	1
0x000004b9	0x000004b0	0x000004b8	1
0x000004b7	0x000004ad	0x000004b6	1
0x000004a6	0x0000049e	0x000004a5	1
0x000004a4	0x0000049b	0x000004a3	1
0x000004a2	0x00000498	0x000004a1	1
0x00000491	0x00000489	0x00000490	1
0x0000048f	0x00000486	0x0000048e	1
0x0000048d	0x00000483	0x0000048c	1
0x0000047c	0x00000478	0x0000047b	1
0x00000475	0x00000471	0x00000474	1
0x0000046e	0x0000046a	0x0000046d	1
0x00000467	0x00000463	0x00000466	1
0x00000453	0x0000044b	0x00000452	1
0x00000451	0x00000448	0x00000450	1
0x0000044f	0x00000445	0x0000044e	1
0x0000043e	0x00000436	0x0000043d	1
0x0000043c	0x00000433	0x0000043b	1
0x0000043a	0x00000430	0x00000439	1
0x00000429	0x00000421	0x00000428	1

Right Pane: Descriptor Set: 0x00000633

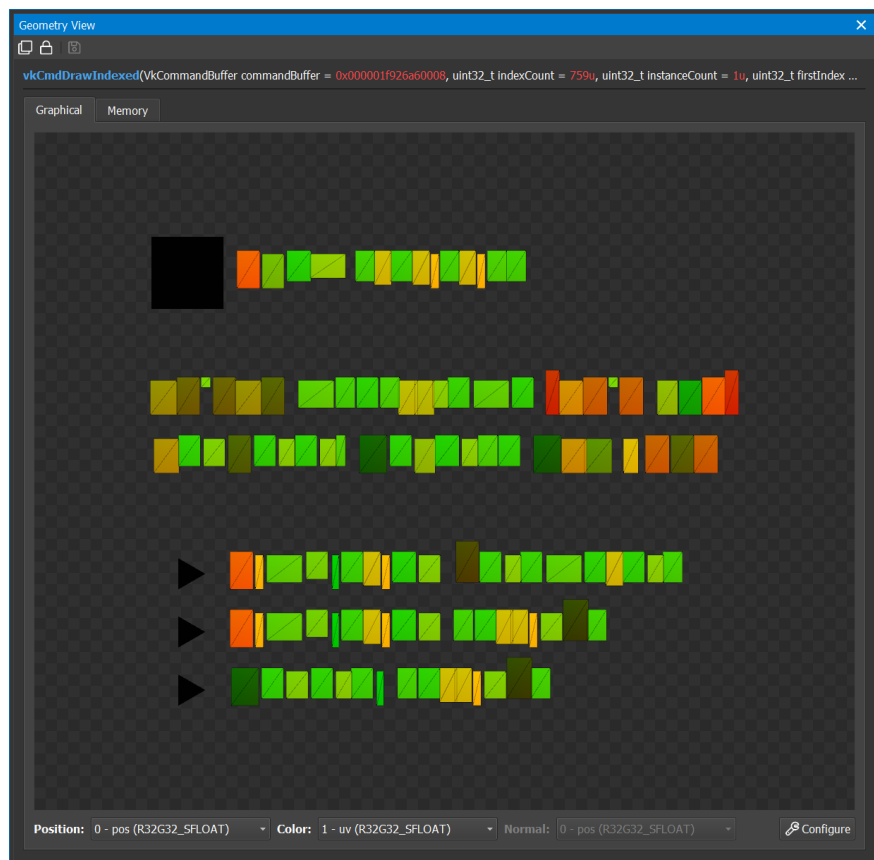
Binding	Element	Type	Stages	Properties	Preview
0	0	SAMPLED_IMAGE	VERTEX	Sampler: N/A Image View: Displacement texture array Image Layout: SHADER_READ_ONLY_OPTIMAL	
128	0	SAMPLER	VERTEX	Sampler: 0x0000062b Image View: N/A Image Layout: UNDEFINED	
256	0	UNIFORM_BUFFER	VERTEX	Buffer: WaveWorks:QuadTreeVertexShaderConstantBuffer Offset: 0 Range: 80	
896	0	UNIFORM_BUFFER	TESSELLATION_CONTROL	Buffer: WaveWorks:QuadTreeHullShaderConstantBuffer Offset: 0 Range: 32	
1280	0	SAMPLED_IMAGE	TESSELLATION_EVALUATION	Sampler: N/A Image View: Displacement texture array Image Layout: SHADER_READ_ONLY_OPTIMAL	
1408	0	SAMPLER	TESSELLATION_EVALUATION	Sampler: 0x0000062b Image View: N/A Image Layout: UNDEFINED	
1536	0	UNIFORM_BUFFER	TESSELLATION_EVALUATION	Buffer: m_hDrawVertexShaderCB Offset: 0 Range: 16	
1537	0	UNIFORM_BUFFER	TESSELLATION_EVALUATION	Buffer: MVP matrix CB Offset: 0 Range: 80	
2560	0	SAMPLED_IMAGE	FRAGMENT	Sampler: N/A Image View: Gradients texture array Image Layout: SHADER_READ_ONLY_OPTIMAL	

Frame Debugger - Shaders

- Per-pipeline and per-shader information
- View shader SPIRV source
- View as GLSL or HLSL via translation



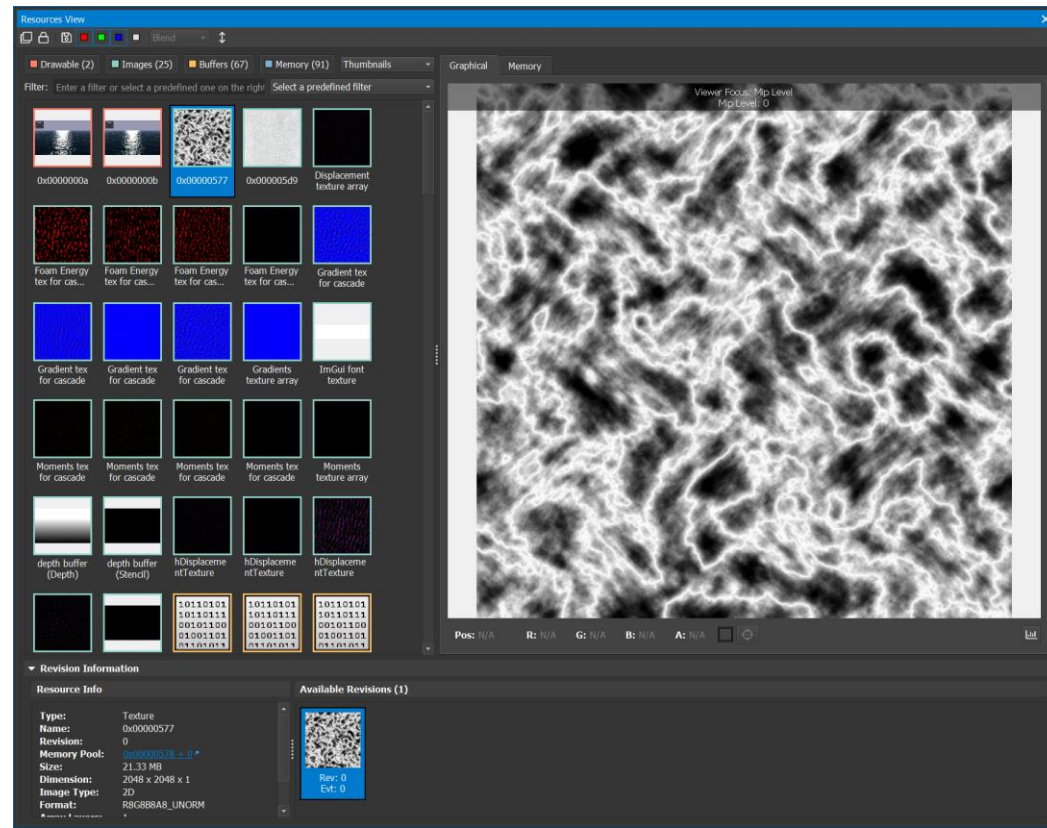
Frame Debugger - Geometry



- 3D geometry of current draw calls
- Highly configurable
 - Select attributes for position, color, and normal
 - Multiple shading modes
- Reflection information from SPIRV annotations

Frame Debugger - Resources

- At a glance view of all images and buffers
- Objects names via VK_EXT_debug_marker extension
- Powerful search and filters
- Revision and consumption tracking
- Links to backing device memory



Frame Debugger - Device Memory

Memory Pools (91)				Resources (1)		Data			
Name	Size	Entries	Flags	Mapping	Mapping Flags	Offset	Type	Size	Offset
0x00000000 512.00 KB	DEVICE_LOCAL	Unmapped	N/A	0	Buffer	0x00000000	Texture	512.00 KB	0
0x00000001 516.25 KB	DEVICE_LOCAL	Unmapped	N/A			Address	Data (0x00010000)		
0x00000005 250.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000000	9a5ca987 00000046 9408246d 343c7a24	0x00002500 9028028a	0x0000265
0x00000010 516.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000000	15077ac8 0000257a 214c48de 34893932	0x00001e5d 2574af6d	0x0000000
0x00000011 65.25 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000000	75040e4e 0000221b 1f6c0e0d 45600000	0x00000000 00000000	0x0000000
0x00000011 65.25 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000000	97f11336 00000076 2400243d 0000727b	0x00002900 2658a47d	0x0000266
0x00000021 512.00 KB	HOST_VISIBLE HOST_COHERENT	Unmapped	N/A			0x0000000000000000	1d01c1e6 000000fc 9d702892 0000005e	0x00000000 24e92c2c	0x0000000
0x00000045 512.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000000	25c0c682 00002f6d 15b8da05 00002887	0x00000000 01241888	0x0000000
0x00000059 512.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000000	1f123244 00000076 2d733810 00000060	0x0000207f 077187	0x0000000
0x00000061 250.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000000	49024555 000000ff 457c15d0 00000000	0x0000205a 2017a035	0x0000000
0x00000048 250.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000000	260fa477 00000076 20922a10 00000000	0x0000221c 9905527d	0x0000264
0x00000025 512.00 KB	HOST_VISIBLE HOST_COHERENT	Unmapped	N/A			0x0000000000000020	3271ac2b 00002c99 25113d93 00000047	0x00001e7b 014c2807	0x0000000
0x00000012 512.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000010	41a1205c 00000046 1d02933c 00000012	0x00000000 2279142d	0x0000264
0x00000013 250.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000040	42acac87 00000046 034649f9 00000000	0x00000000 11972927	0x0000000
0x00000007 516.25 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000080	27442366 0000258f 280b1a34 00000034	0x00000076 013d041a	0x0000000
0x00000003 512.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x00000000000000c0	40e0000f 00000076 25511a2d 00000000	0x00000000 00000000	0x0000000
0x00000034 65.25 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000100	73999f00 00000077 783c3245 00000000	0x00000000 27322290	0x0000265
0x00000043 516.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x00000000000001e0	18ac3e79 0000294d 00000028 00002525	0x00000000 349307b4dc	0x0000000
0x00000022 512.00 KB	HOST_VISIBLE HOST_COHERENT	Unmapped	N/A			0x0000000000000200	193f1b62 00000084 23261eaf 00000001	0x00000000 1f732000	0x0000265
0x00000017 512.00 KB	HOST_VISIBLE HOST_COHERENT	Unmapped	N/A			0x0000000000000220	40f8700c 00000076 10000000 00000000	0x00000000 00000000	0x0000000
0x00000066 65.25 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000240	0d027a78 00000088 9502a216 00000003	0x00000000 00002947	0x0000266
0x00000068 516.25 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000260	210ba87f 00000085 210ea753 00000000	0x00002019 20b8621d	0x0000264
0x00000064 512.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000000000280	1a4728f0 000028ff 100a2a39 00000000	0x0000220c 19f4406e	0x0000266
0x00000066 512.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x00000000000002a0	1600e12c 0000007a 104e0203 00000000	0x0000220c 01242000	0x0000265
0x00000066 250.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x00000000000002c0	44e2b269 000000fe 43c2d4f9 00002773	0x00000000 1f70a282	0x0000266
0x00000096 516.00 KB	DEVICE_LOCAL	Unmapped	N/A			0x0000000			

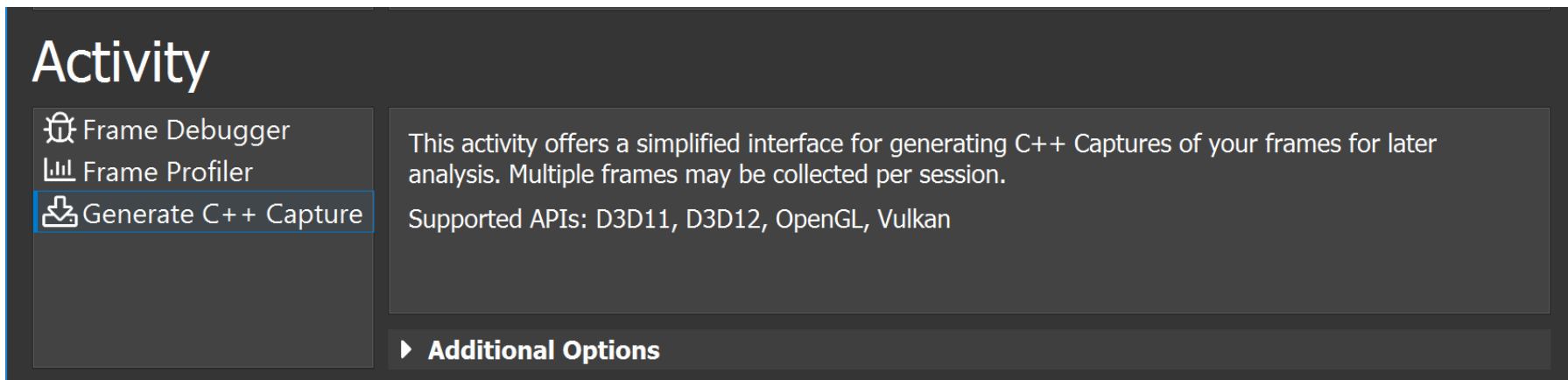
- At a glance view of all device memory objects
- Listing of bound resources
- Memory of each sub-resource
- Object layout map

NOTE: This example is only one resource per-memory region.

C++ Capture

C++ Capture - Overview

- Export an application frame as a free standing executable built from C++ source
- Previously called “Serialization” in Nsight VSE
- Debug / profile / edit & experiment with source



C++ Capture - Code Output

- Great for rapid “hack and slash” changes
- Human readable code

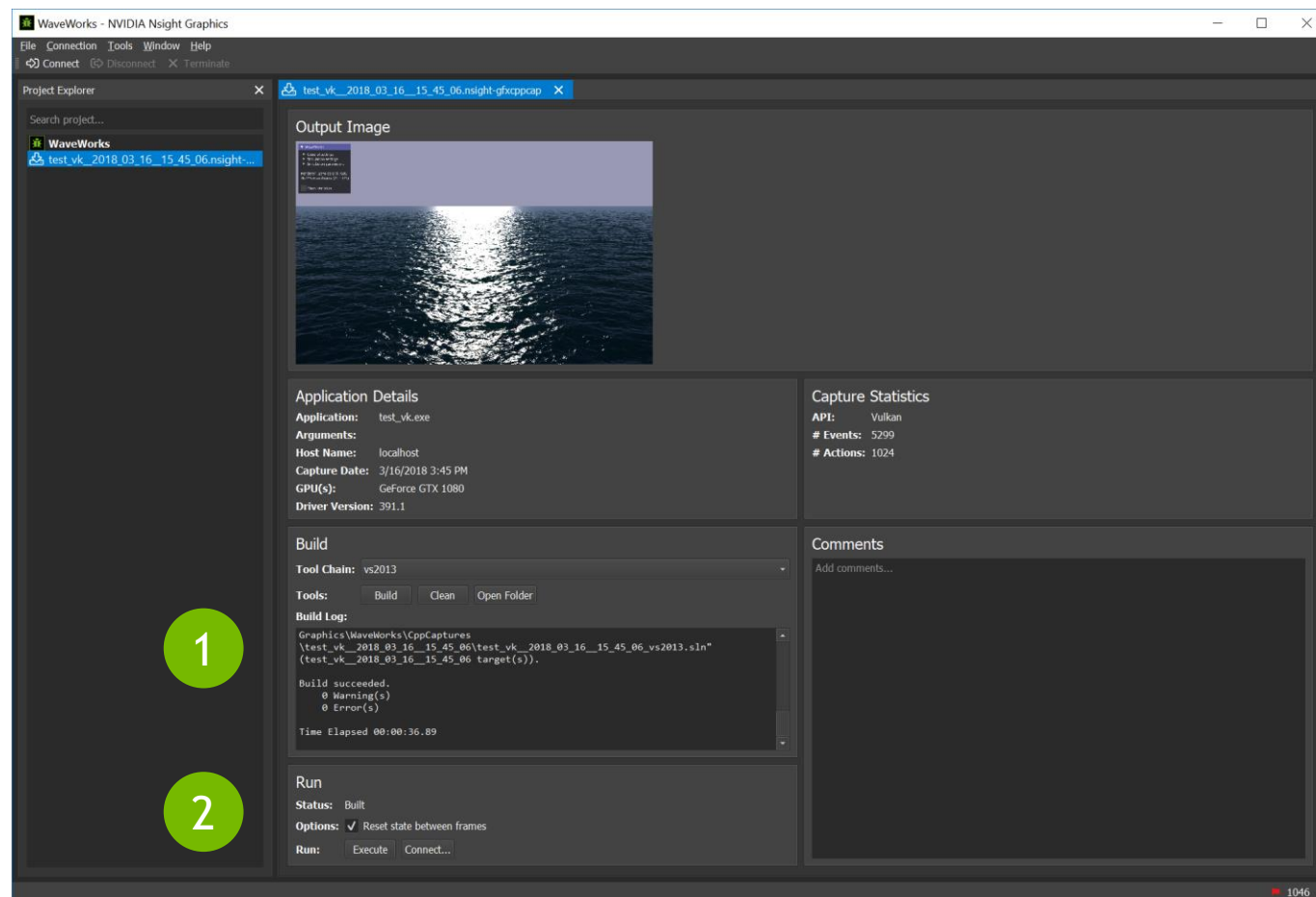
```
844 // Create VkImageView_uid_40
845 {
846     BEGIN_DATA_SCOPE();
847
848     static VkImageViewCreateInfo VkImageViewCreateInfo_temp_14[1] = { VkImageViewCreateInfo{
849         /* sType = */ VK_STRUCTURE_TYPE_IMAGE_VIEW_CREATE_INFO,
850         /* pNext = */ nullptr,
851         /* flags = */ VkImageViewCreateFlags(0),
852         /* image = */ VkImage_uid_37,
853         /* viewType = */ VK_IMAGE_VIEW_TYPE_2D,
854         /* format = */ VK_FORMAT_R8G8B8A8_UNORM,
855         /* components = */ VkComponentMapping{
856             /* >> r = */ VK_COMPONENT_SWIZZLE_R,
857             /* >> g = */ VK_COMPONENT_SWIZZLE_G,
858             /* >> b = */ VK_COMPONENT_SWIZZLE_B,
859             /* >> a = */ VK_COMPONENT_SWIZZLE_A},
860         /* subresourceRange = */ VkImageSubresourceRange{
861             /* >> aspectMask = */ VkImageAspectFlags(VK_IMAGE_ASPECT_COLOR_BIT),
862             /* >> baseMipLevel = */ 0u,
863             /* >> levelCount = */ 1u,
864             /* >> baseArrayLayer = */ 0u,
865             /* >> layerCount = */ 1u}} };
866     NV_THROW_IF(vkCreateImageView(VkDevice_uid_16, VkImageViewCreateInfo_temp_14, nullptr, &VkImageView_uid_40) != VK_SUCCESS, "A Vulkan API call was
        unsuccessful");
867 }
```

C++ Capture - Vulkan Capabilities

- Respects the acquire / record / submit / present model
 - Can't naively loop frames
 - Need to record command buffers based on results of `vkAcquireNextImageKHR`
- Best-effort to support replay on different hardware from capture
 - Dynamically patch up code
 - Some cases are impossible (e.g. missing major feature or extension)

C++ Capture - .nsight-gfxcppcap File

- Additional metafile generated with C++ export
- Screenshot, system information, statistics, etc.
- Ability to build (#1) and launch (#2) from within the GUI
- Easily debug, profile, etc. later



Vulkan Tools Roadmap

Vulkan Tools Roadmap

■ 2018 - 1st Half

- Vulkan 1.1 support
- Linux support*
- Android support
- Shader statistics
- Shader editing
- C++ export improvements

■ 2018 - 2nd Half

- Hardware profiling
- GPU trace
- Pixel history

■ Beyond...

- Support future extensions and core updates

**Already available in the NVIDIA Linux Graphics Debugger 2.3*

Roadmap - Shader Statistics

- Low level information about each pipeline and shader object
- Estimated cycles
- Register counts
- LMem counts

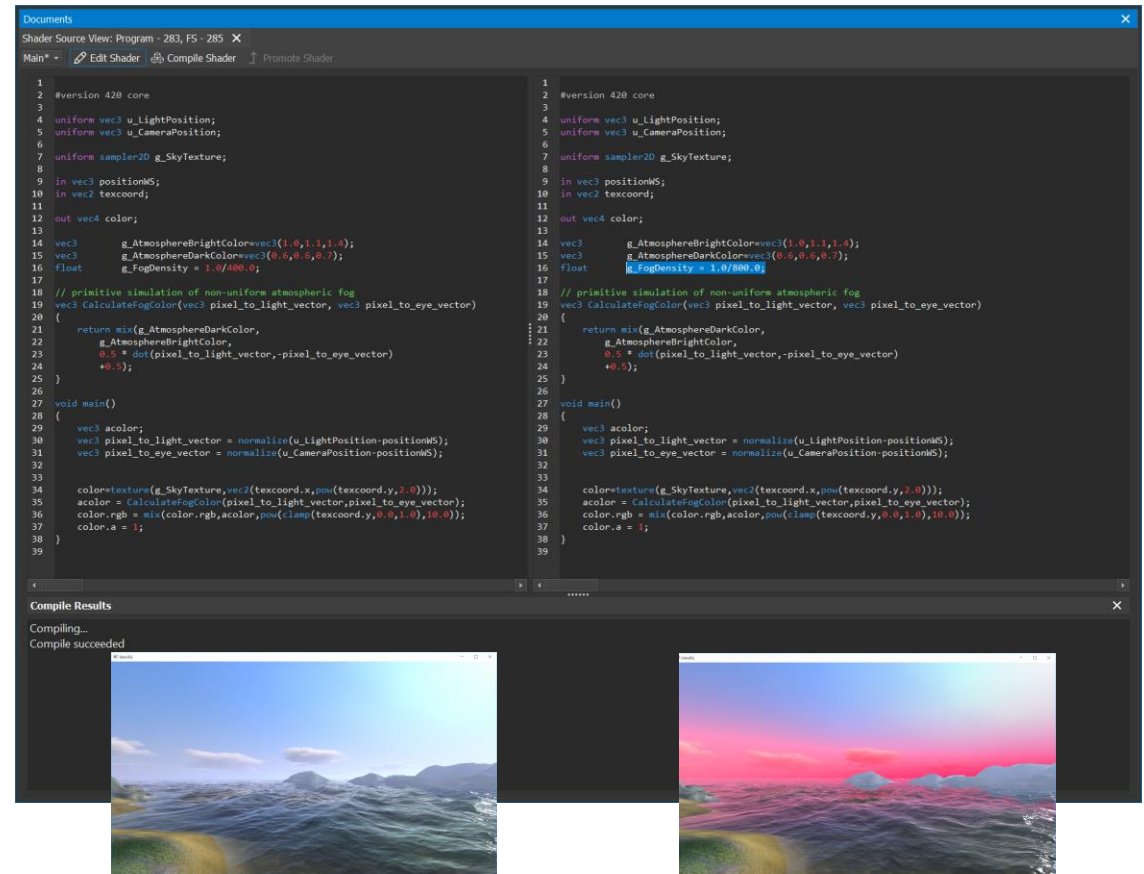
Linked Programs View										
Filter: Enter a filter										
All Programs & Shaders (19)										Selected Program(s) & Shader(s)
Name	Type	Status	Debug Info	Cycles	Avg Cycles	ALU/TEX (Inst)	ALU/TEX (Cycles)	Regs	LMem (Bytes)	
Bloom	PS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	CSMain
CSHorizFilter	CS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	Float16 0
CSMain	CS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	Float32 11
CSMain	CS	Source + µCode	None	993	44	53.00	32.16	22	0	Slow Float32 4
CSMain	CS	Source + µCode	None	940	43	61.00	21.28	22	0	Float64 0
CSVerticalFilter	CS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	Integer 41
DownScale2x2_Lum	PS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	Slow Integer 5
DownScale3x3	PS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	Interpolation 0
DownScale3x3_BrightPass	PS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	Tex 1
FinalPass	PS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	Read Generic 12
PS	PS	Source + µCode	None	280	11	4.00	7.11	10	0	Read Thread Local 0
PSDump	PS	Source + µCode	None	210	7	4.00	2.92	5	0	Read GPU Global 0
PSFinalPass	PS	Source + µCode	None	365	16	9.67	2.04	13	0	Read Attribute 0
PSFinalPassForCPUReduction	PS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	Write Generic 7
PSIntex	PS	Source	None	N/A	N/A	N/A	N/A	N/A	N/A	Write Thread Local 0
QuadVS	VS	Source + µCode	None	125	12	--	--	7	0	Write GPU Global 2
SkyboxPS	PS	Source + µCode	None	264	10	7.00	22.98	9	0	Write Attribute 0
SkyboxVS	VS	Source + µCode	None	211	19	--	--	12	0	Flow Control 28
										NOP 3

NOTE: Currently supported for D3D & OpenGL

Roadmap - Shader Editing

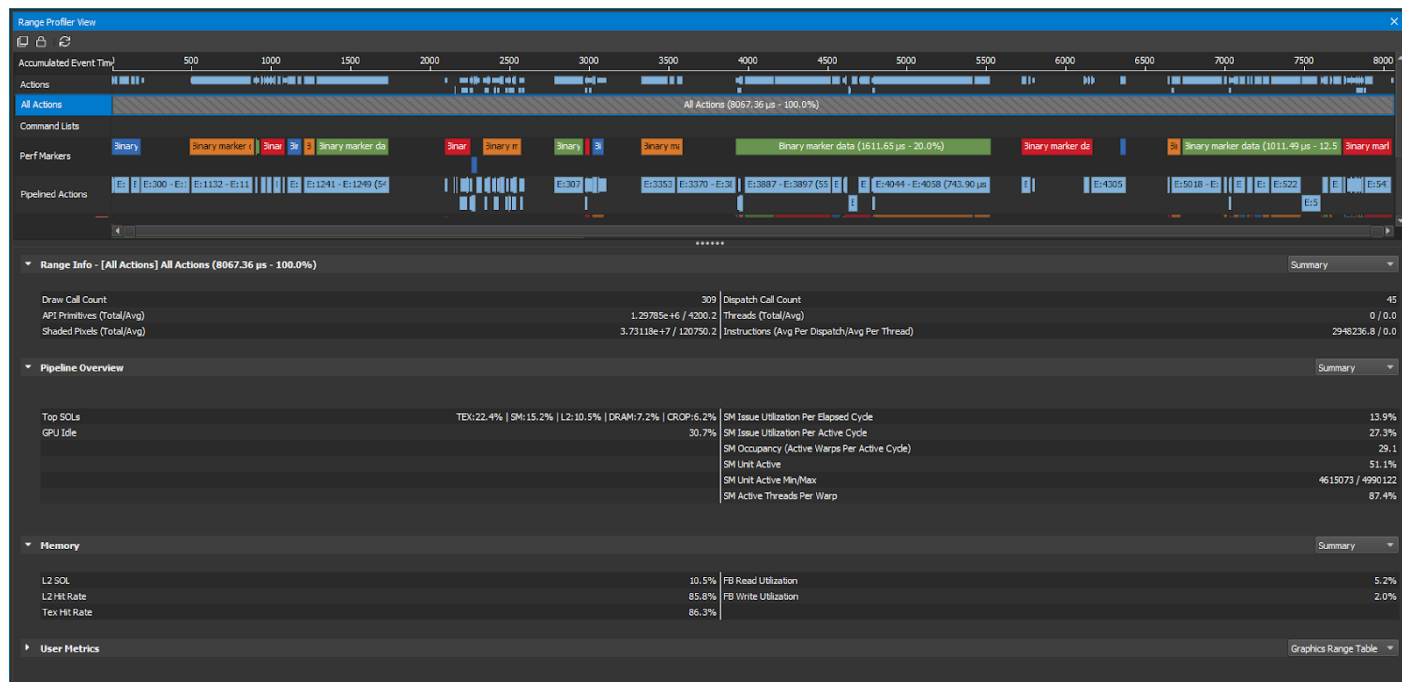
- Edit shader source and see changes reflected live in running application
- Quickly toggle between shader sets
- Edit as SPIRV, GLSL, or HLSL

NOTE: Currently supported for D3D & OpenGL



Roadmap - Hardware Profiling

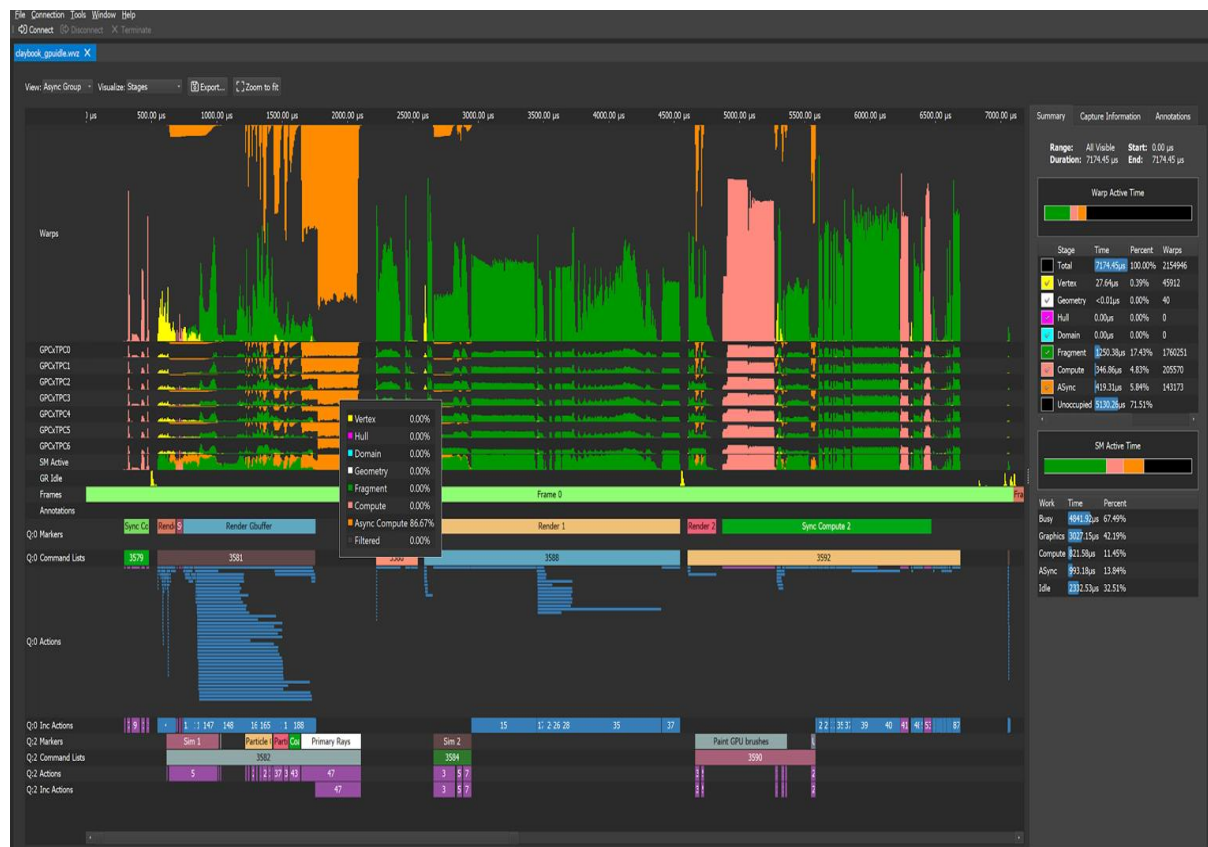
- Low level performance metrics on a range and event basis
- Find performance limiting hardware units
- Identify optimization opportunities



NOTE: Currently supported for D3D & OpenGL

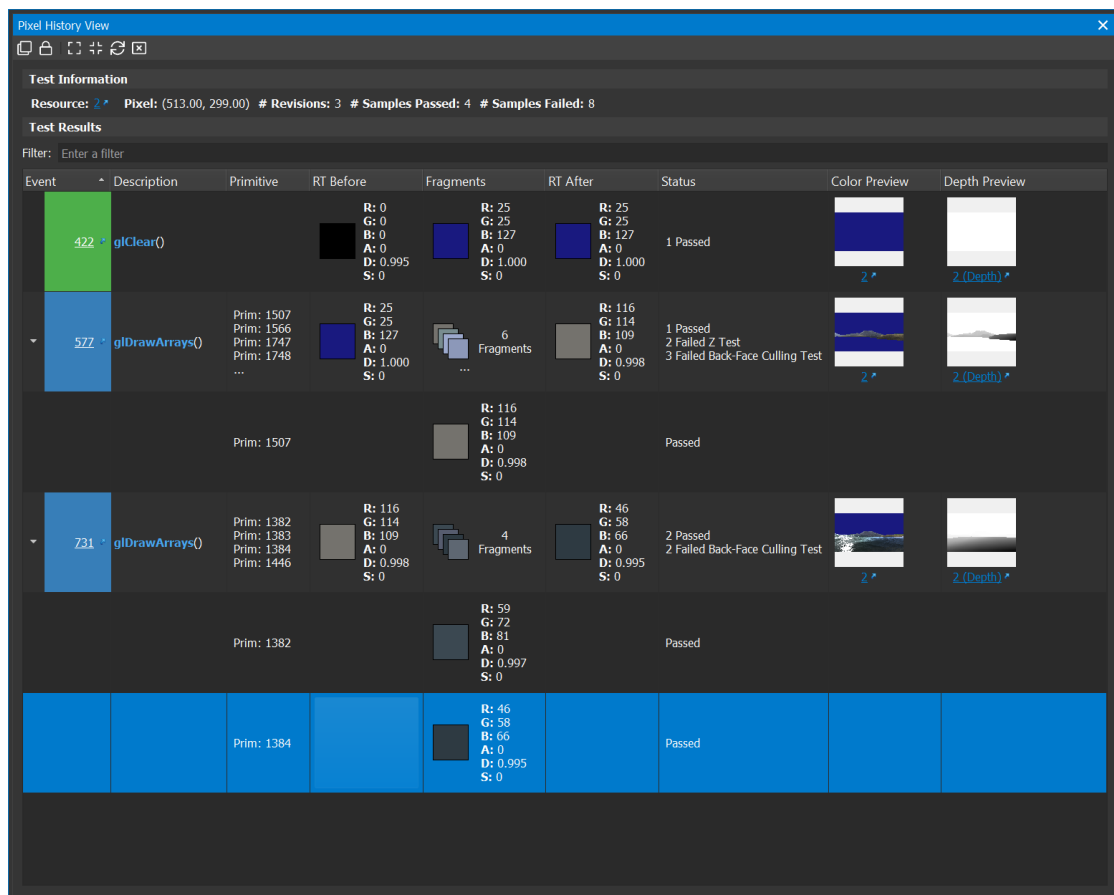
Roadmap - GPU Trace

- Low level GPU utilization
- Identify and optimize asynchronous compute opportunities



NOTE: Currently supported for D3D12 in preview

Road Map - Pixel History



Pixel History View

Test Information
Resource: Pixel: (513.00, 299.00) # Revisions: 3 # Samples Passed: 4 # Samples Failed: 8

Test Results
Filter: Enter a filter

Event	Description	Primitive	RT Before	Fragments	RT After	Status	Color Preview	Depth Preview
422	glClear()		R: 0 G: 0 B: 0 A: 0 D: 0.995 S: 0	R: 25 G: 25 B: 127 A: 0 D: 1.000 S: 0	R: 25 G: 25 B: 127 A: 0 D: 1.000 S: 0	1 Passed		
527	glDrawArrays()	Prim: 1507 Prim: 1566 Prim: 1747 Prim: 1748 ...	R: 25 G: 25 B: 127 A: 0 D: 1.000 S: 0	6 Fragments	R: 116 G: 114 B: 109 A: 0 D: 0.998 S: 0	1 Passed 2 Failed Z Test 3 Failed Back-Face Culling Test		
		Prim: 1507		R: 116 G: 114 B: 109 A: 0 D: 0.998 S: 0		Passed		
731	glDrawArrays()	Prim: 1382 Prim: 1383 Prim: 1384 Prim: 1446	R: 116 G: 114 B: 109 A: 0 D: 0.998 S: 0	4 Fragments	R: 46 G: 58 B: 66 A: 0 D: 0.995 S: 0	2 Passed 2 Failed Back-Face Culling Test		
		Prim: 1382		R: 59 G: 72 B: 81 A: 0 D: 0.997 S: 0		Passed		
		Prim: 1384		R: 46 G: 58 B: 66 A: 0 D: 0.995 S: 0		Passed		

- Trace the life of a pixel in a render target
- Draws, clears, and blits that contribute to final output pixel value
- Detailed information about failed fragments
 - Back face culling, depth, etc.

NOTE: Currently supported for D3D & OpenGL

Thank you!

- Nvidia Nsight Graphics 1.0 now available
- Nvidia Linux Graphics Debugger 2.3 (with Vulkan support) now available
- Live demos @ Booth 233 in South Hall
- We are hiring!

NVIDIA.com > Developers > GameWorks > Tools

<https://developer.nvidia.com/nsight-graphics>