

Beyond Performance:

Introducing NVIDIA's New Graphics Debugger

Aurelio Reis, Director of Engineering, Graphics Developer Tools

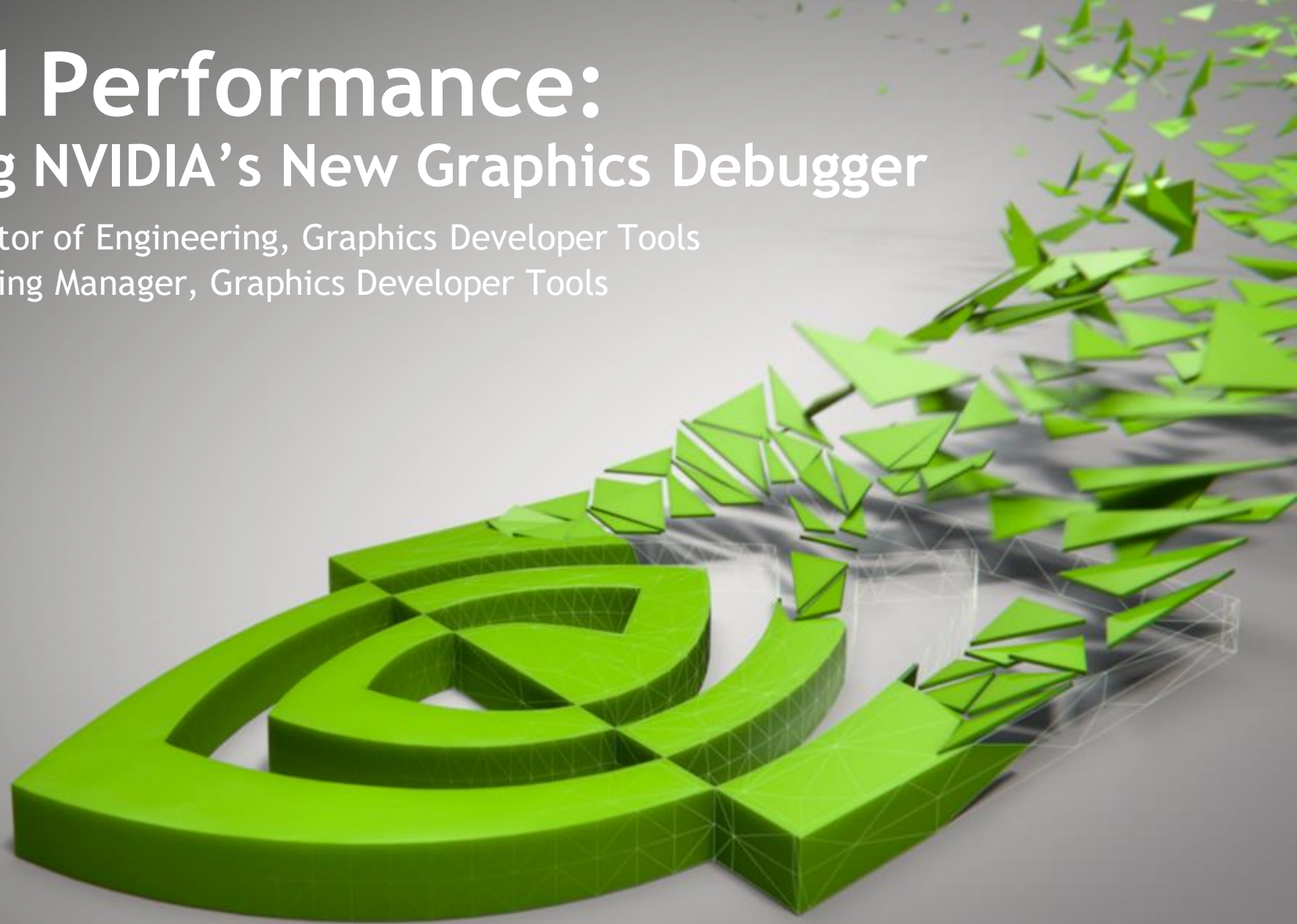
Jeff Kiel, Engineering Manager, Graphics Developer Tools

March 22



Booth #223 - South Hall

www.nvidia.com/GDC



Agenda

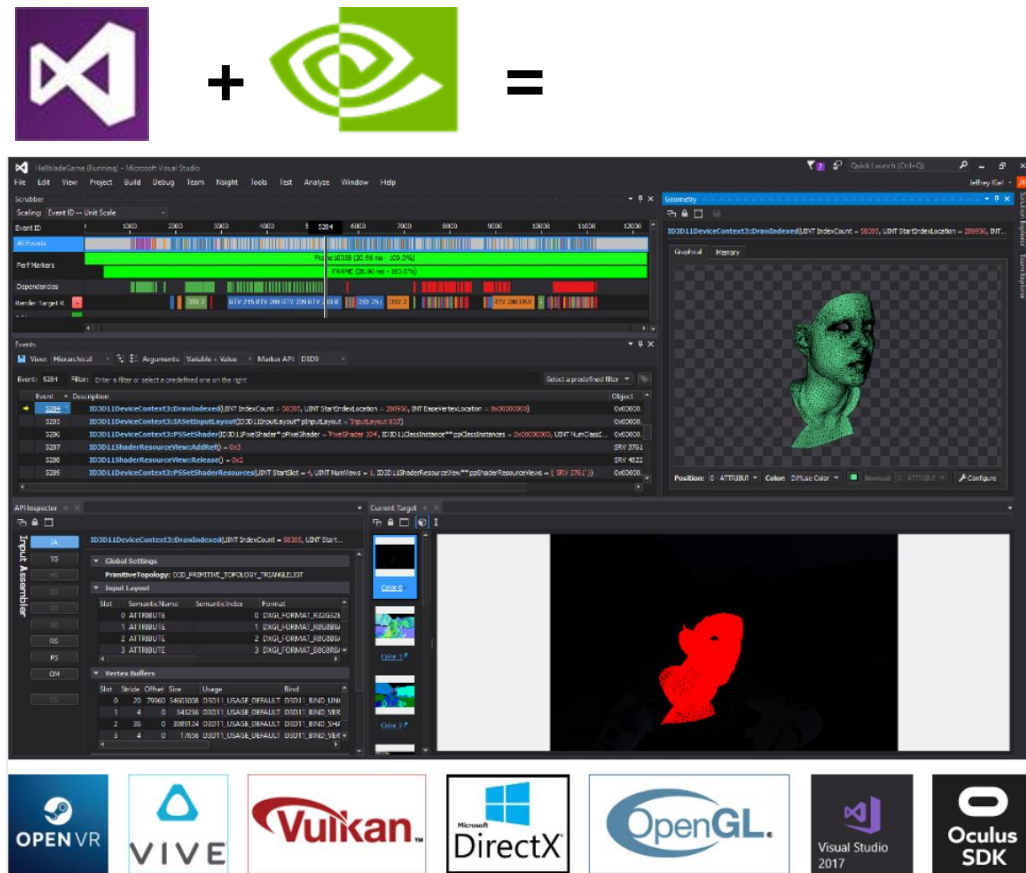
- Overview
- Demo
- Key Takeaways
- Q&A

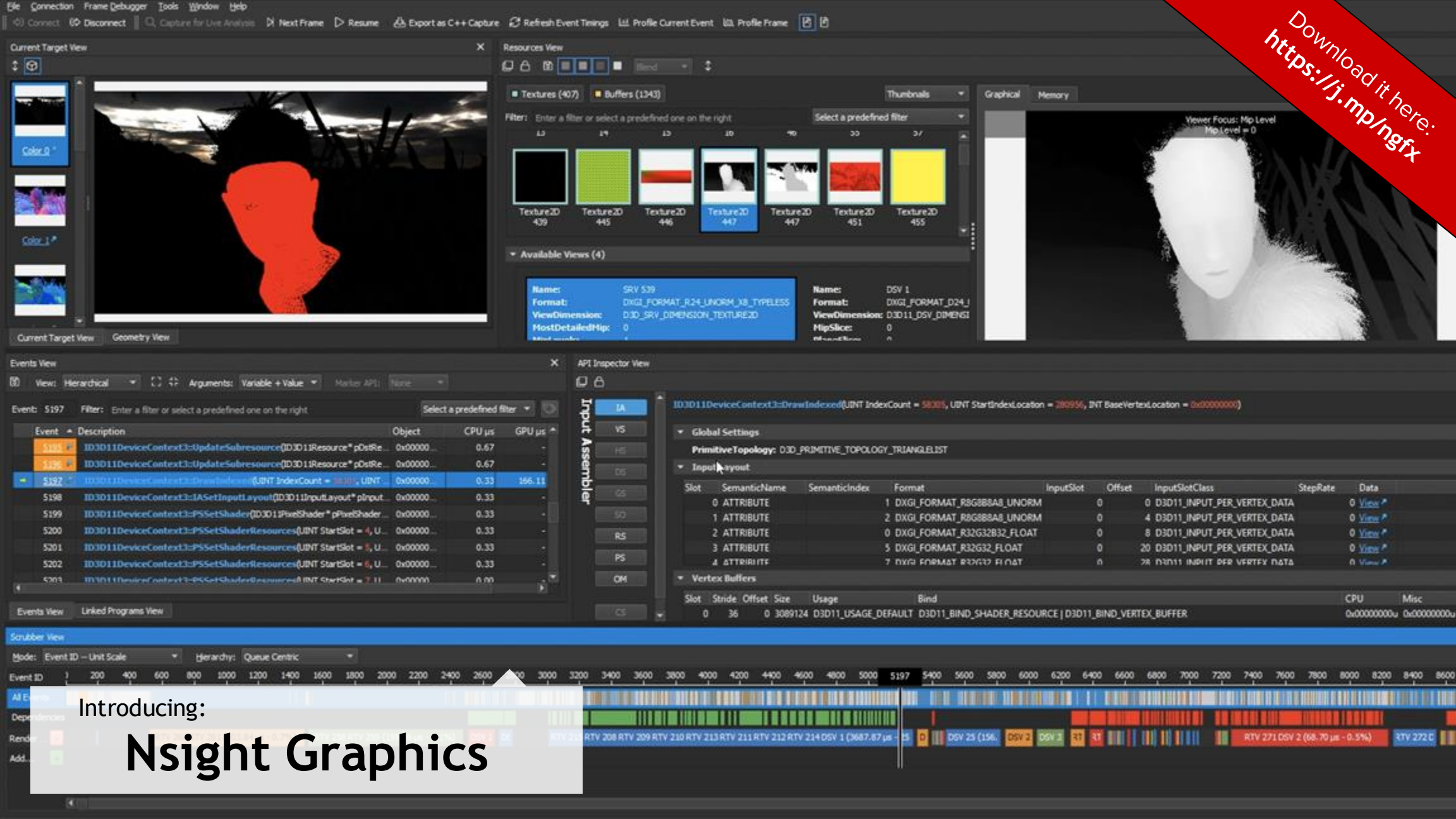
Overview

The Story So Far...

- Nsight: Visual Studio Edition
 - Fully integrated into IDE
 - Graphics
 - Compute (CUDA)
 - 9+ years of learnings
- Ideal workflow for developers, but...
 - Reliant on VS ecosystem
 - Dependencies on compute for release

How can we improve?





Download it here:
<https://j.mp/ngfx>

Introducing:
Nsight Graphics

Key Pillars

Debugging

C++ Serialized Captures
Event Timeline (Scrubber)
Shader Editing
Pixel History
Resources Viewer
Geometry Viewer
API Inspection



Profiling

Range Profiler
HW Perf Counters
Shader Stats
Event Timings
Counter Selector
API Stats
GPU Trace



Ray Tracing (RTX)

DXR Capture/Replay
DXR Frame Debugger
Shader Bind Table View

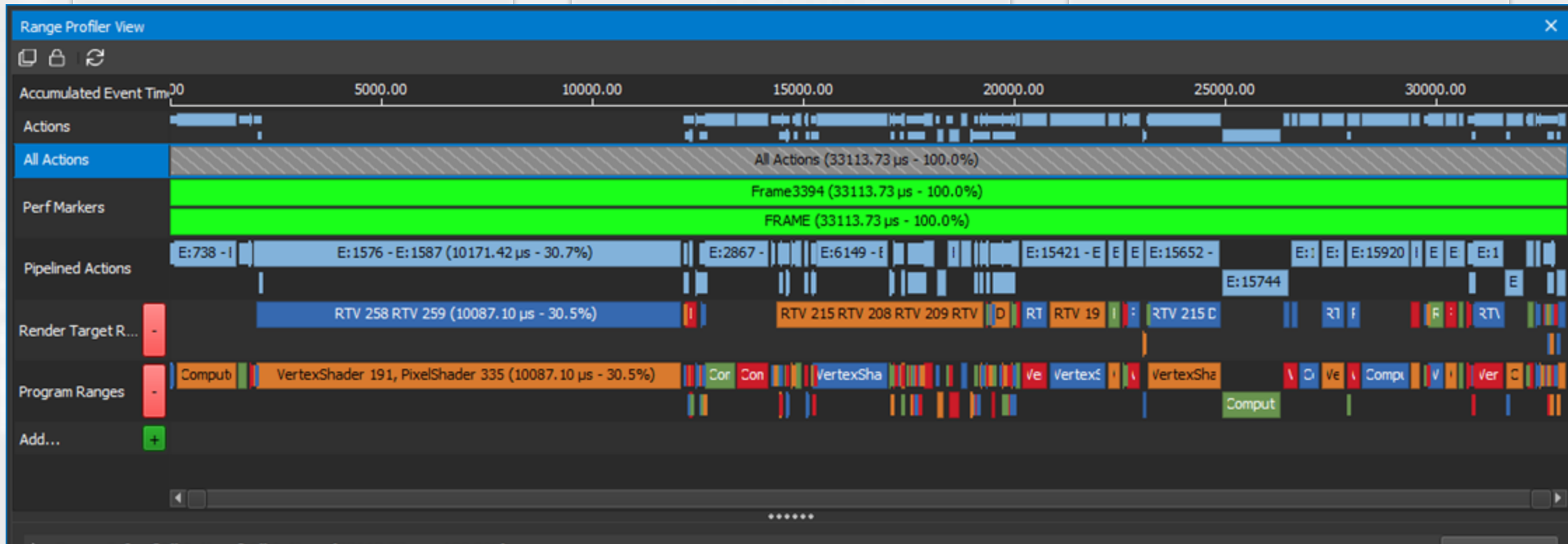


Key Pillars

Debugging

Profiling

Ray Tracing (RTX)



Key Pillars

Debugging

Profiling

Ray Tracing (RTX)



Key Pillars

Debugging

C++ Serialized
Event Timeline
Shader Editing
Pixel History
Resources View
Geometry View
API Inspection



Profiling

Ray Tracing (RTX)



Developer Choice

Nsight: VSE

- + Visual Studio integration
- + Compute tools
- + Tracing tools

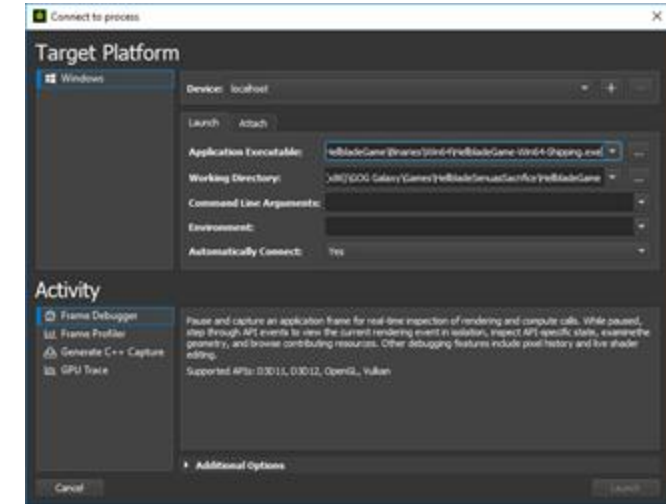
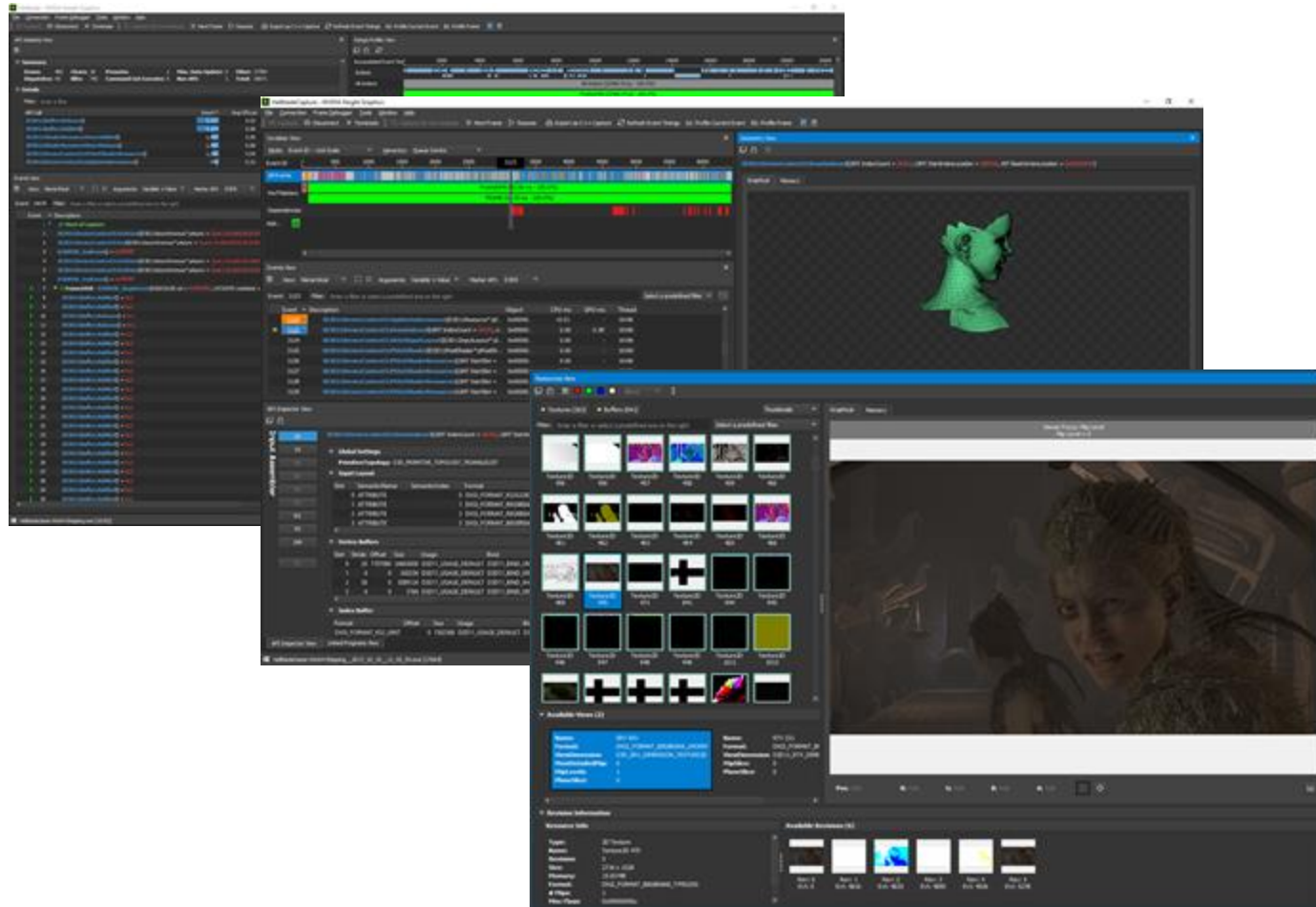
Nsight Graphics

- + More frequent releases
- + Activities system
- + Advanced profiling

Demo

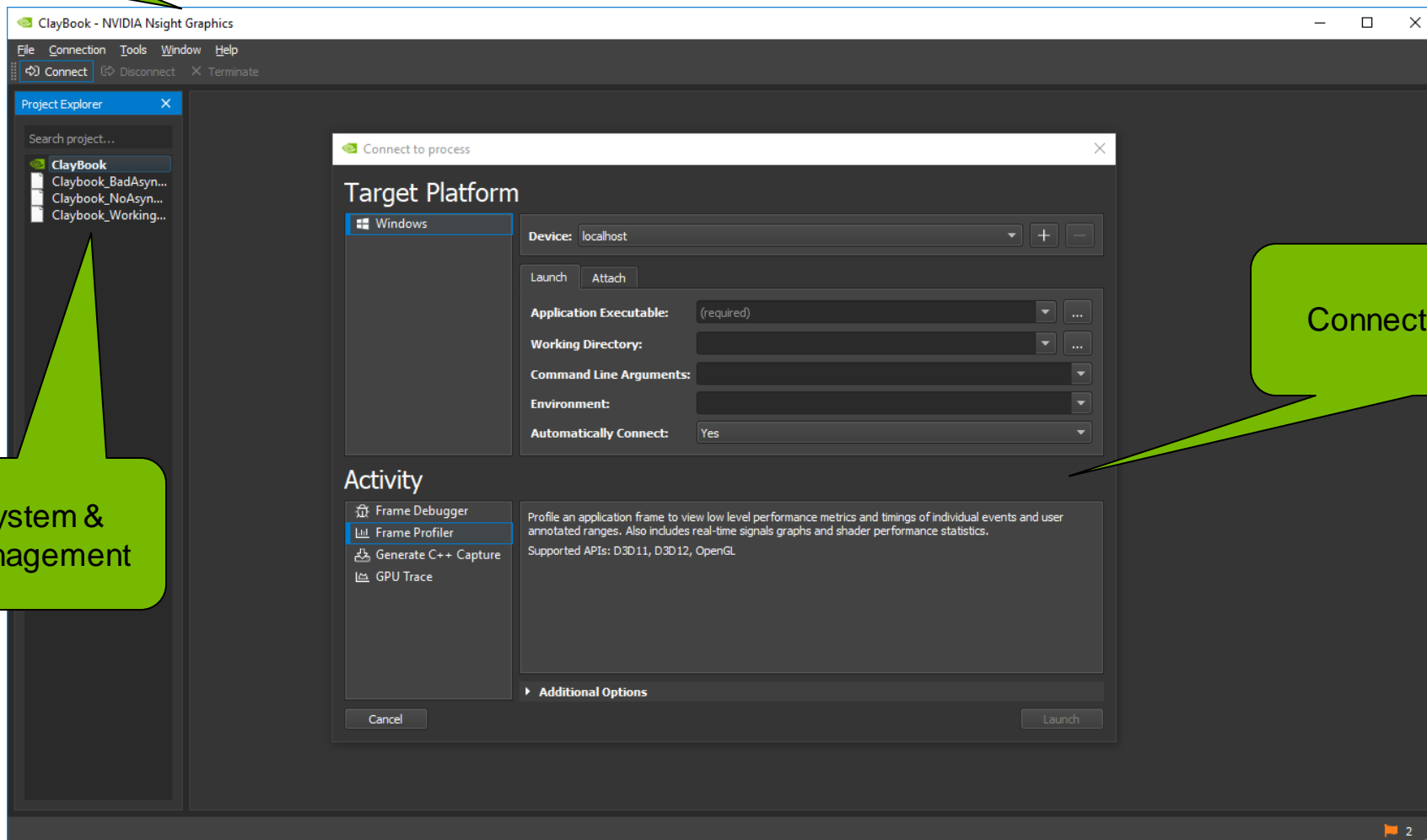
1. Nsight Graphics Tour
2. DXR Debugging
3. Optimizing Async Compute

Demo 1 - Nsight Graphics Tour



New, stand alone shell!

Nsight Graphics Tour



Project system &
artifact management

Connection settings

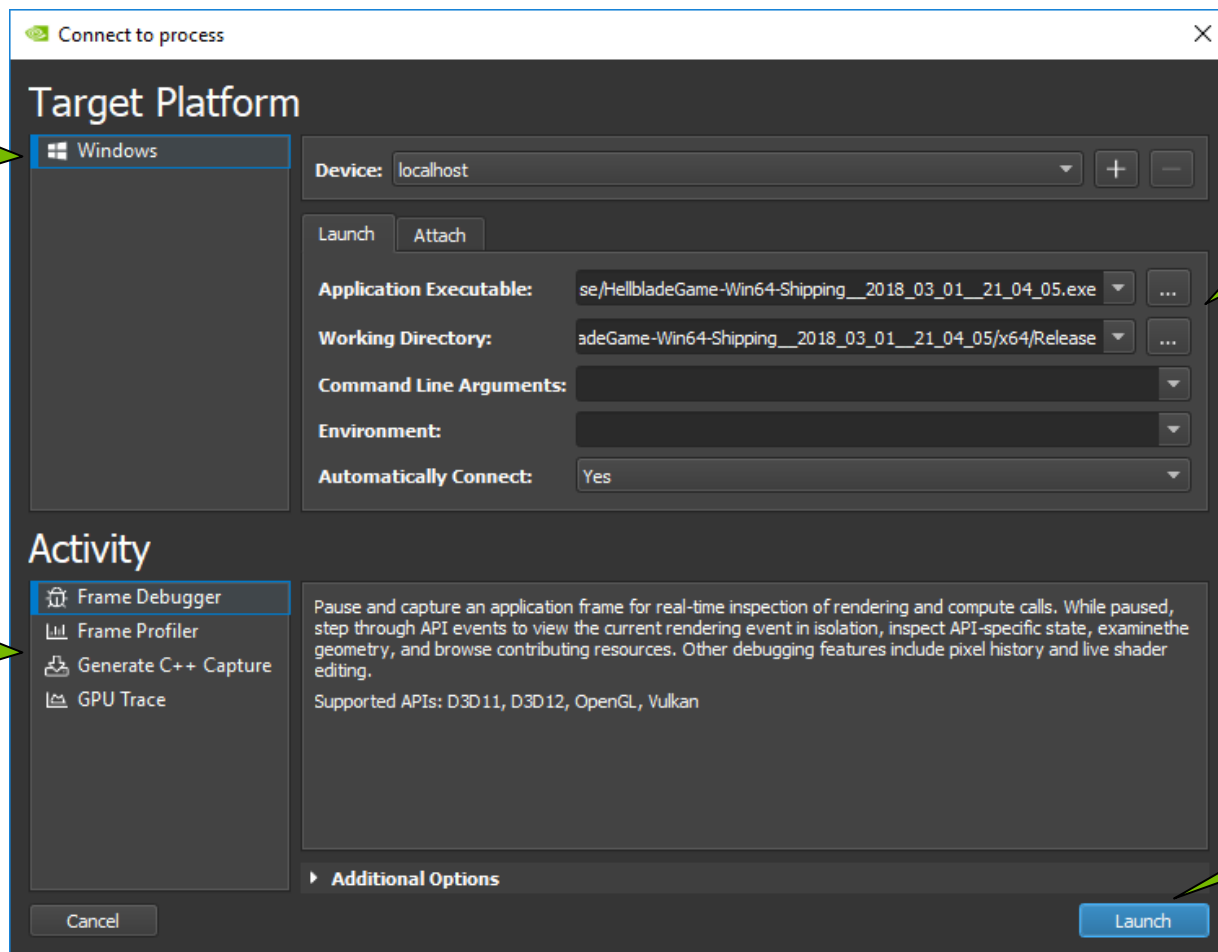
Nsight Graphics Tour

Select Platform....

...select Activity....

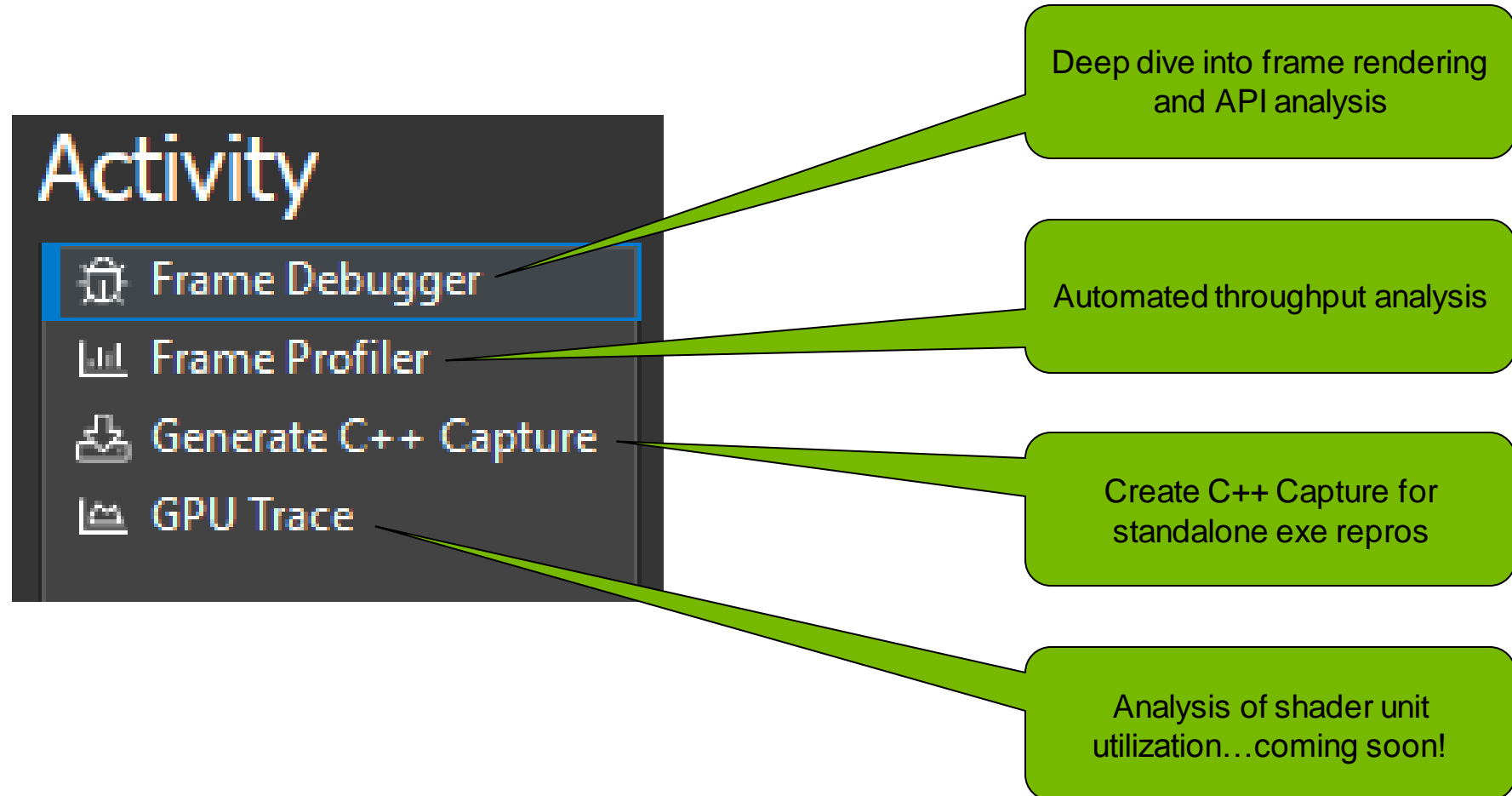
...add application parameters...

...Launch!



Nsight Graphics Tour

Targeted UI Layout Per Activity



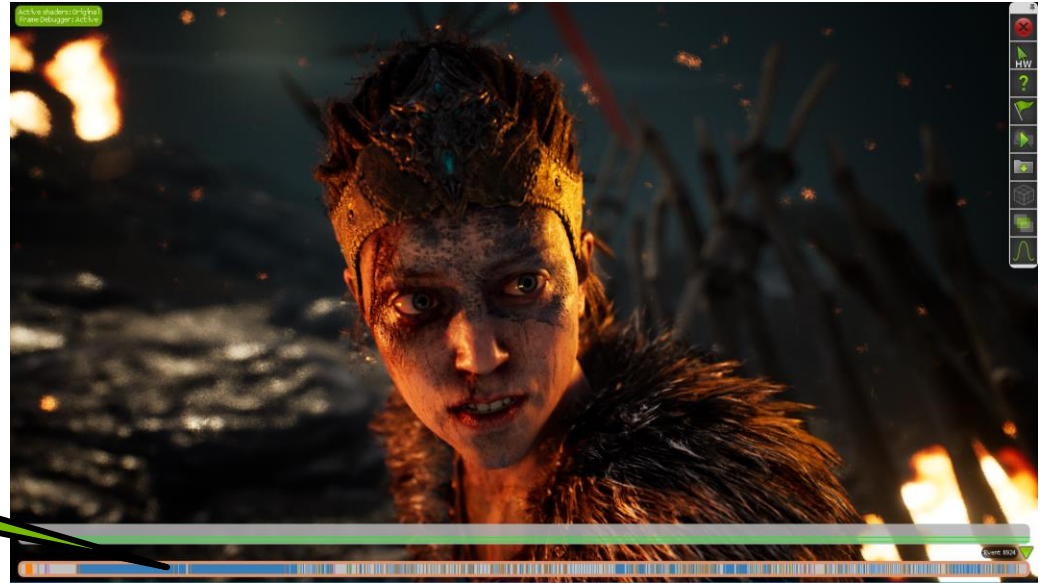
Nsight Graphics Tour



Real time performance
metrics

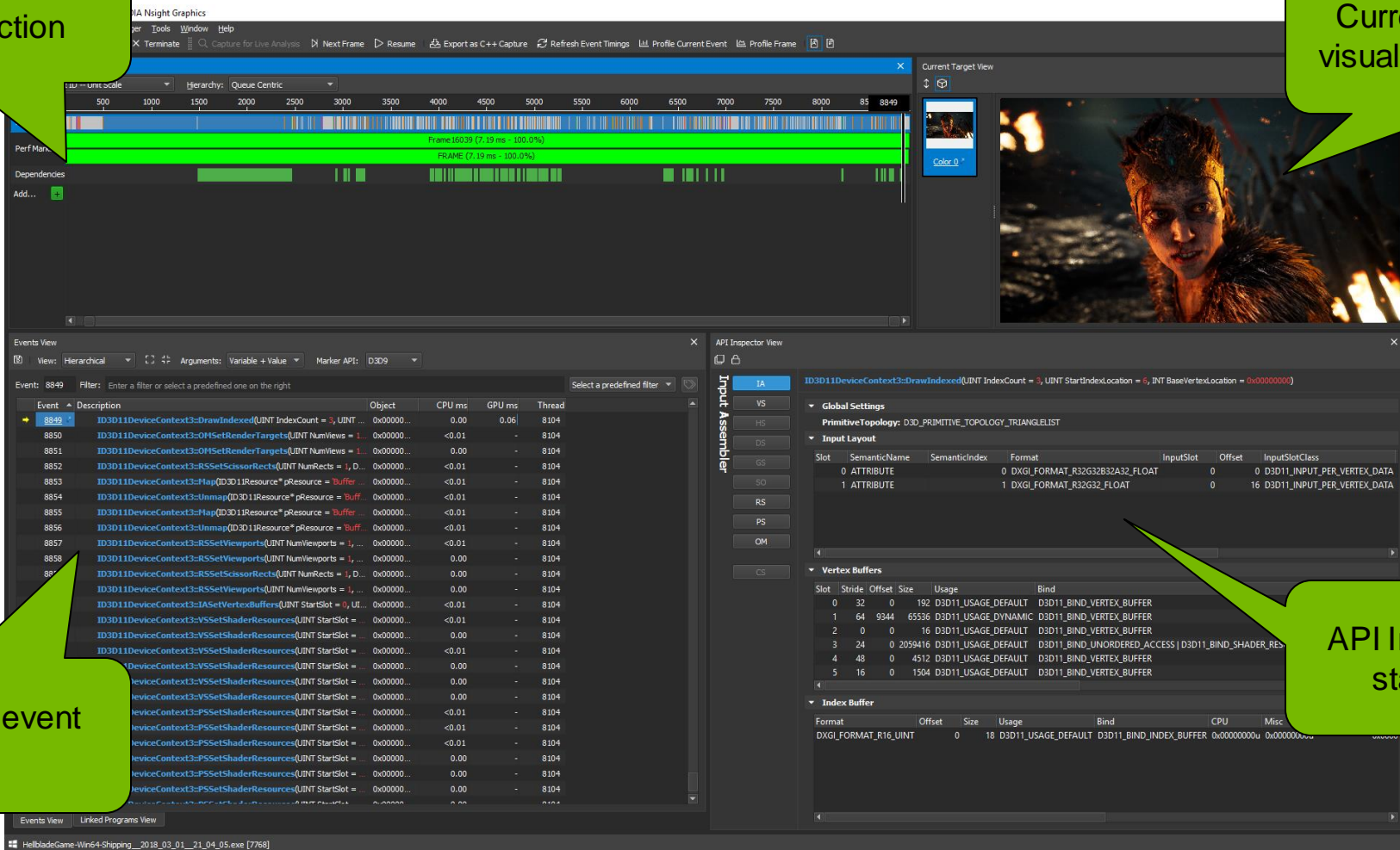
Live capture: return to
your application at will

Scrub through the scene,
better understand GPU
rendering



Scrubber: break down frame construction

Current Target View:
visualize render targets



Event List: API event trace

API Inspector: full API state inspection

Events View: filter to find problematic API usage

Nsight Graphics Tour

Frequent shader changes can impact performance

Events View

View: Hierarchical Arguments: Variable + Value Marker API: D3D9 (1271 matches) X Select a predefined filter

Event: 8849 Filter: draw[pssetshader](

Event	Description	Object	CPU ms	GPU ms	Thread
6952	ID3D11DeviceContext3::DrawInstanced(UINT VertexCountPerInsta...	0x00000...	0.00	0.00	8104
6975	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	<0.01	-	8104
6991	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 6, UINT ...	0x00000...	0.00	0.09	8104
7005	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	<0.01	-	8104
7013	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...	0.00	0.08	8104
7125	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	0.00	-	8104
7133	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...	<0.01	0.05	8104
7136	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	0.00	-	8104
7144	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...	0.00	0.03	8104
7147	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	<0.01	-	8104
7151	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...	0.00	0.02	8104
7155	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	<0.01	-	8104
7158	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	0.00	-	8104
7171	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...	0.00	0.06	8104
7178	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	0.00	-	8104
7186	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...	0.00	0.07	8104
7191	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	<0.01	-	8104
7194	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...	0.00	0.03	8104
7200	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	<0.01	-	8104
7230	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	0.00	8104
7233	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	0.00	-	8104
7249	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	<0.01	8104
7254	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	<0.01	8104
7262	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	0.00	8104
7272	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	0.00	8104
7277	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	<0.01	8104
7285	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	0.00	8104
7288	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	0.00	-	8104
7301	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	0.00	8104
7304	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	<0.01	-	8104
7315	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	<0.01	0.01	8104
7320	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	0.00	8104
7328	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...	0.00	<0.01	8104
7331	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...	0.00	-	8104

7144	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...
7147	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...
7151	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...
7155	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...
7158	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...
7171	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...
7178	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...
7186	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...
7191	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...
7194	ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT ...	0x00000...
7200	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...
7230	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...
7233	ID3D11DeviceContext3::PSSetShader(ID3D11PixelShader* pPixelSh...	0x00000...
7249	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...
7254	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...
7262	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...
7272	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...
7277	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...
7285	ID3D11DeviceContext3::DrawIndexedInstanced(UINT IndexCount...	0x00000...

Nsight Graphics Tour

Pipeline Navigator: state categorized by GPU location

Hyper-linked UI for easy navigation

State laid out for easy inspection

The screenshot displays the 'API Inspector View' window in Nsight Graphics. The left sidebar shows the 'Input Assembler' selected. The main panel shows the command `ID3D11DeviceContext3::DrawIndexed(UINT IndexCount = 3, UINT StartIndexLocation = 6, INT BaseVertexLocation = 0x00000000)`. Below this, the 'Global Settings' section shows 'PrimitiveTopology: D3D_PRIMITIVE_TOPOLOGY_TRIANGLELIST'. The 'Input Layout' section contains a table with columns: Slot, SemanticName, SemanticIndex, Format, InputSlot, Offset, InputSlotClass, StepRate, and Data. The 'Vertex Buffers' section contains a table with columns: Slot, Stride, Offset, Size, Usage, Bind, and CPU. The 'Index Buffer' section contains a table with columns: Format, Offset, Size, Usage, Bind, CPU, Misc, StructureByteStride, and ObjectName.

Slot	SemanticName	SemanticIndex	Format	InputSlot	Offset	InputSlotClass	StepRate	Data
0	ATTRIBUTE	0	DXGI_FORMAT_R32G32B32A32_FLOAT	0	0	D3D11_INPUT_PER_VERTEX_DATA	0	View
1	ATTRIBUTE	1	DXGI_FORMAT_R32G32_FLOAT	0	16	D3D11_INPUT_PER_VERTEX_DATA	0	View

Slot	Stride	Offset	Size	Usage	Bind	CPU
0	32	0	192	D3D11_USAGE_DEFAULT	D3D11_BIND_VERTEX_BUFFER	0x00000000u
1	64	9344	65536	D3D11_USAGE_DYNAMIC	D3D11_BIND_VERTEX_BUFFER	D3D11_CPU_...
2	0	0	16	D3D11_USAGE_DEFAULT	D3D11_BIND_VERTEX_BUFFER	0x00000000u
3	24	0	2059416	D3D11_USAGE_DEFAULT	D3D11_BIND_UNORDERED_ACCESS D3D11_BIND_SHADER_RESOURCE D3D11_BIND_VERTEX_BUFFER	0x00000000u
4	48	0	4512	D3D11_USAGE_DEFAULT	D3D11_BIND_VERTEX_BUFFER	0x00000000u
5	16	0	1504	D3D11_USAGE_DEFAULT	D3D11_BIND_VERTEX_BUFFER	0x00000000u

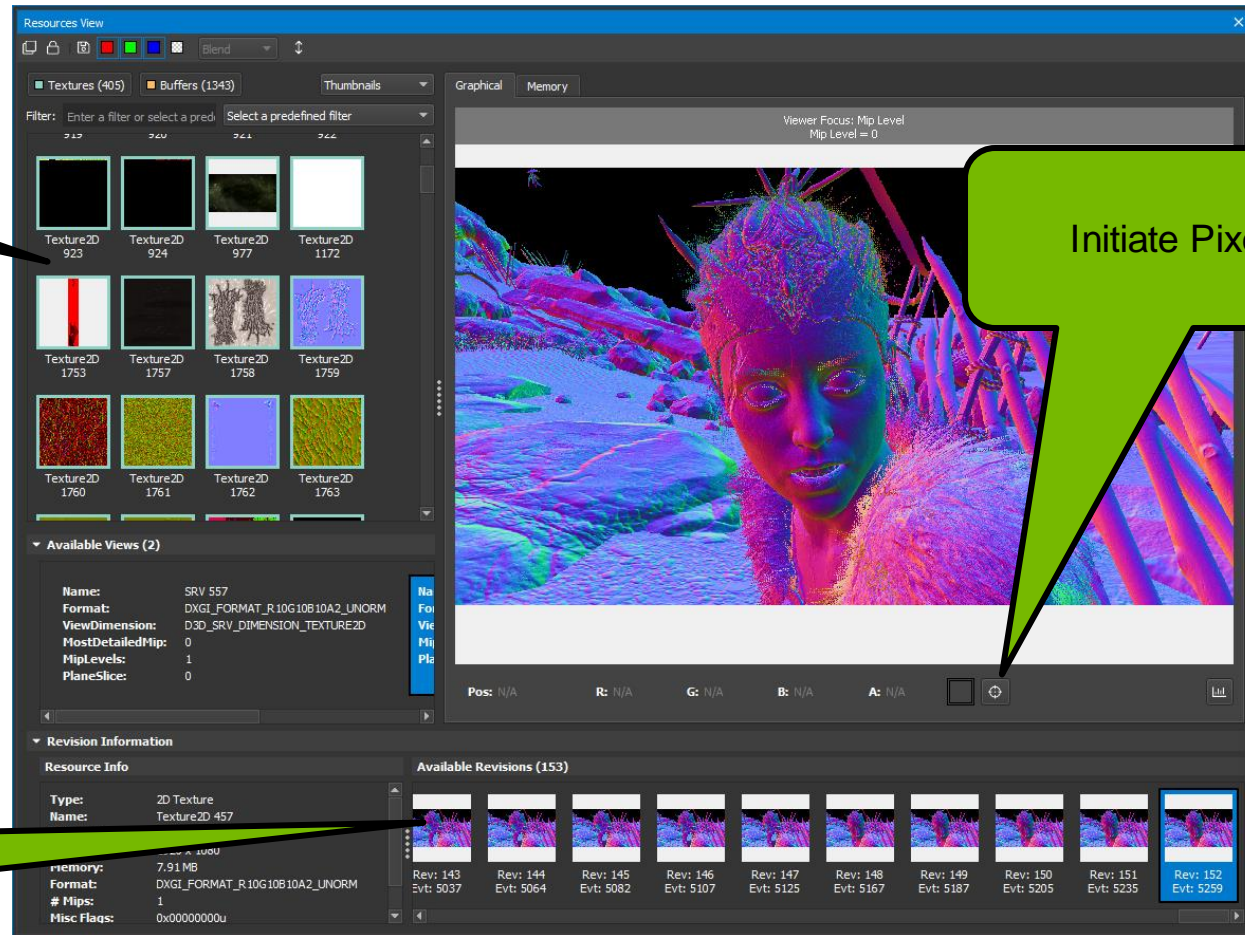
Format	Offset	Size	Usage	Bind	CPU	Misc	StructureByteStride	ObjectName
DXGI_FORMAT_R16_UINT	0	18	D3D11_USAGE_DEFAULT	D3D11_BIND_INDEX_BUFFER	0x00000000u	0x00000000u	0x00000000	Buffer 165

Nsight Graphics Tour

Inspect all resource types,
views, etc.

Initiate Pixel History!

Scrub through buffer
revisions



Nsight Graphics Tour

Show all fragments that potentially touched a pixel location

Inspect RT Before, Fragment, and RT After color values

Understand why fragments DON'T land in the frame buffer, like back facing or Z-culled

Pixel History View

Test Information

Resource: [Texture2D_457](#) Pixel: (879.00, 652.00) # Revisions: 6 # Samples Passed: 10 # Samples Failed: 33

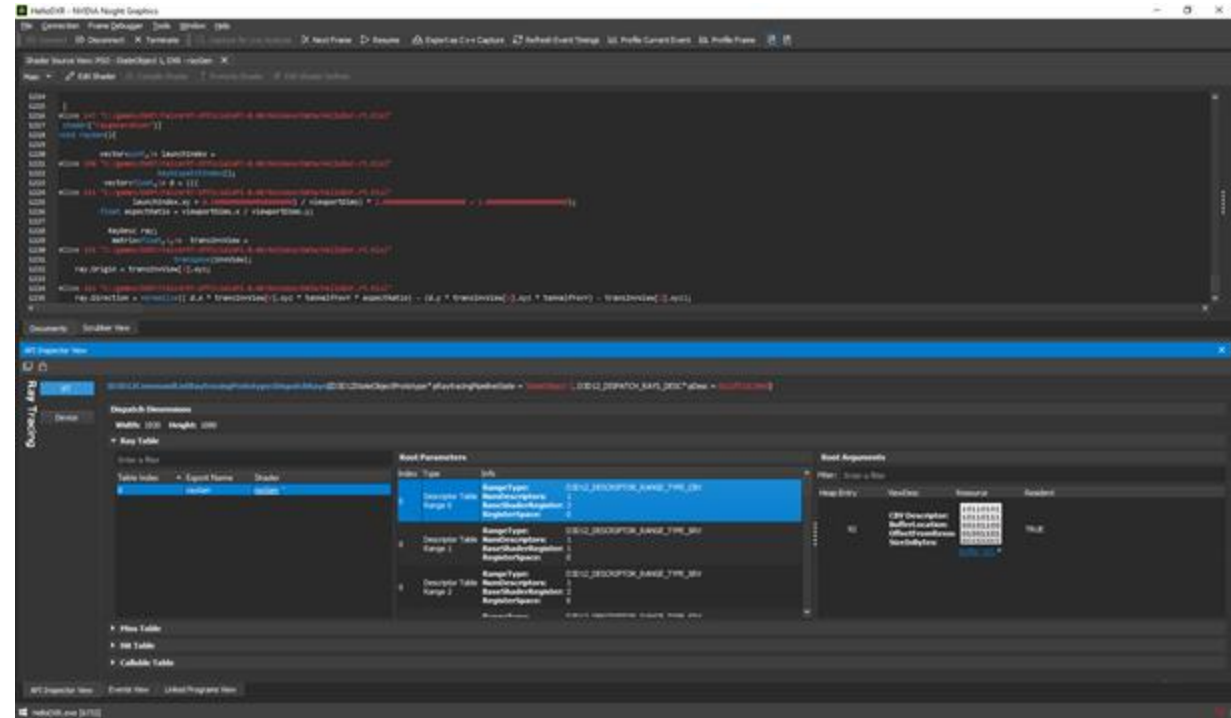
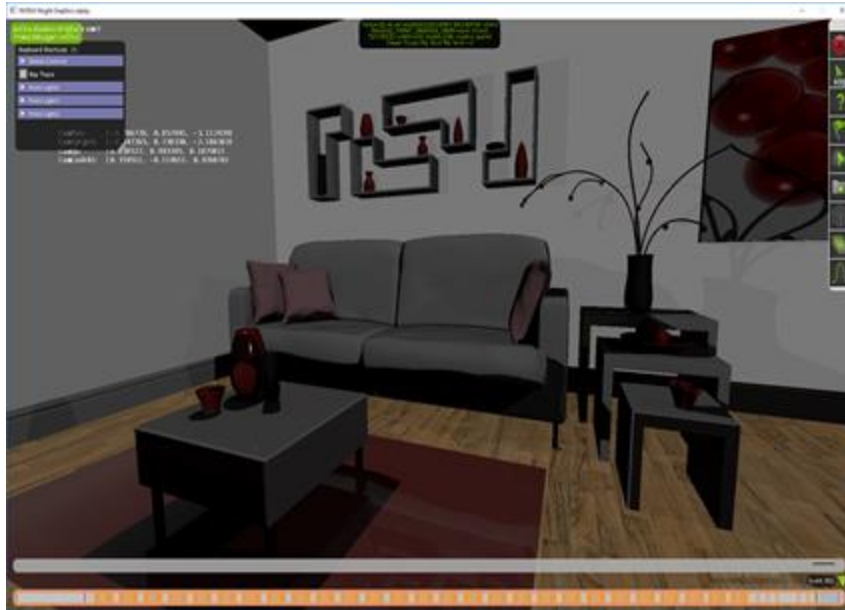
Failed pixel status is provided on a best-effort basis as the test is unable to incorporate all possible failure cases

Test Results

Filter: Enter a filter

Event	Description	Primitive	RT Before	Fragments	RT After	Status	Color Pre
4319	ID3D11DeviceContext3::DrawIndexed()	Prim: 1665 Prim: 1699	R: 443 G: 535 B: 1018 A: 0 D: 0.007 S: 0	2 Fragments	R: 85 G: 246 B: 609 A: 0 D: 0.010 S: 0	1 Passed 1 Failed Back-Face Culling Test	Texture2D_457
4378	ID3D11DeviceContext3::DrawIndexedInstance...	Inst: 0 / Prim: 254 Inst: 0 / Prim: 263 Inst: 1 / Prim: 167 Inst: 1 / Prim: 219	R: 85 G: 246 B: 609 A: 0 D: 0.010 S: 0	4 Fragments	R: 664 G: 460 B: 997 A: 0 D: 0.014 S: 0	1 Passed 1 Failed Z Test 2 Failed Back-Face Culling Test	Texture2D_457
		Inst: 0 / Prim: 254		R: 642 G: 859 B: 864 A: 0 D: 0.013 S: 0		Failed Back-Face Culling Test	
		Inst: 0 / Prim: 263		R: 664 G: 460 B: 997 A: 0 D: 0.014 S: 0		Passed	
				R: 675 G: 857 B: 851 D: 0.011 S: 0		Failed Back-Face Culling Test	
		Inst: 1 / Prim: 219		R: 745 G: 355 B: 939 A: 0 D: 0.012 S: 0		Failed Z Test	
5037	ID3D11DeviceContext3::DrawIndexed()	Prim: 1118 Prim: 1572 Prim: 1655 Prim: 1884 ...	R: 664 G: 460 B: 997 A: 0 D: 0.014 S: 0	16 Fragments	R: 487 G: 35 B: 698 A: 2 D: 0.148 S: 0	5 Passed 11 Failed Z Test	Texture2D_457

DXR Debugging



DXR Debugging

Inspect bound Ray/Miss/Hit Shaders

Visualize Root Parameters/Arguments and associated data values

Dispatch Dimensions
Width: 1920 Height: 1080

RT Root Parameters

Ray Generation

Miss Table

Hit Table

Enter a filter

Table Index	Export Name	Shader
0	RtProgramVersion0	primaryClosestHit
1	RtProgramVersion1	shadowAnyHit
2	RtProgramVersion0	primaryClosestHit
3	RtProgramVersion1	shadowAnyHit
4	RtProgramVersion0	primaryClosestHit
5	RtProgramVersion1	shadowAnyHit
6	RtProgramVersion0	primaryClosestHit
7	RtProgramVersion1	shadowAnyHit
8	RtProgramVersion0	primaryClosestHit
9	RtProgramVersion1	shadowAnyHit
10	RtProgramVersion0	primaryClosestHit
11	RtProgramVersion1	shadowAnyHit
12	RtProgramVersion0	primaryClosestHit
13	RtProgramVersion1	shadowAnyHit
14	RtProgramVersion0	primaryClosestHit
15	RtProgramVersion1	shadowAnyHit
16	RtProgramVersion0	primaryClosestHit
17	RtProgramVersion1	shadowAnyHit
18	RtProgramVersion0	primaryClosestHit
19	RtProgramVersion1	shadowAnyHit
20	RtProgramVersion0	primaryClosestHit
21	RtProgramVersion1	shadowAnyHit
22	RtProgramVersion0	primaryClosestHit
23	RtProgramVersion1	shadowAnyHit

Root Parameters

Index	Type	Info
Descriptor Table Range 0	RangeType: D3D12_DESCRIPTOR_RANGE_TYPE_UAV NumDescriptors: 1 BaseShaderRegister: 1 RegisterSpace: 0	
Descriptor Table Range 1	RangeType: D3D12_DESCRIPTOR_RANGE_TYPE_SRV NumDescriptors: 1 BaseShaderRegister: 50 RegisterSpace: 0	
Descriptor Table Range 2	RangeType: D3D12_DESCRIPTOR_RANGE_TYPE_SRV NumDescriptors: 1 BaseShaderRegister: 51 RegisterSpace: 0	
Descriptor Table Range 3	RangeType: D3D12_DESCRIPTOR_RANGE_TYPE_SRV NumDescriptors: 1 BaseShaderRegister: 52 RegisterSpace: 0	
Descriptor Table Range 4	RangeType: D3D12_DESCRIPTOR_RANGE_TYPE_SRV NumDescriptors: 1 BaseShaderRegister: 53 RegisterSpace: 0	
Descriptor Table Range 5	RangeType: D3D12_DESCRIPTOR_RANGE_TYPE_SRV NumDescriptors: 1 BaseShaderRegister: 54 RegisterSpace: 0	
Descriptor Table Range 6	RangeType: D3D12_DESCRIPTOR_RANGE_TYPE_SRV NumDescriptors: 1 BaseShaderRegister: 55 RegisterSpace: 0	
Descriptor Table Range 7	RangeType: D3D12_DESCRIPTOR_RANGE_TYPE_SRV NumDescriptors: 1 BaseShaderRegister: 56 RegisterSpace: 0	
Descriptor Table Range 8	RangeType: D3D12_DESCRIPTOR_RANGE_TYPE_CBV NumDescriptors: 1	

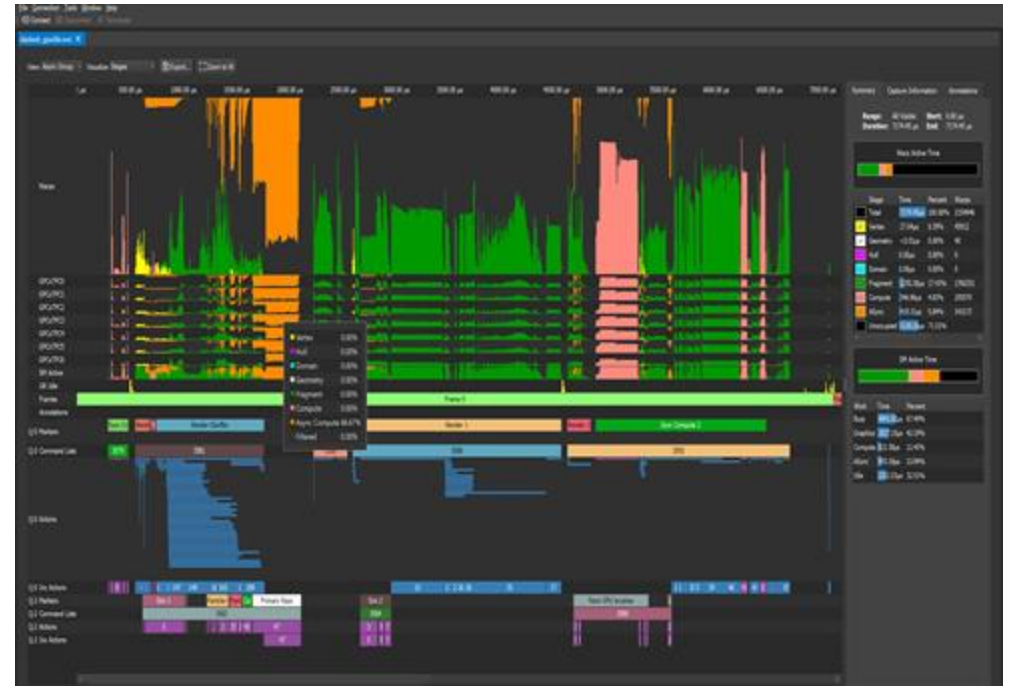
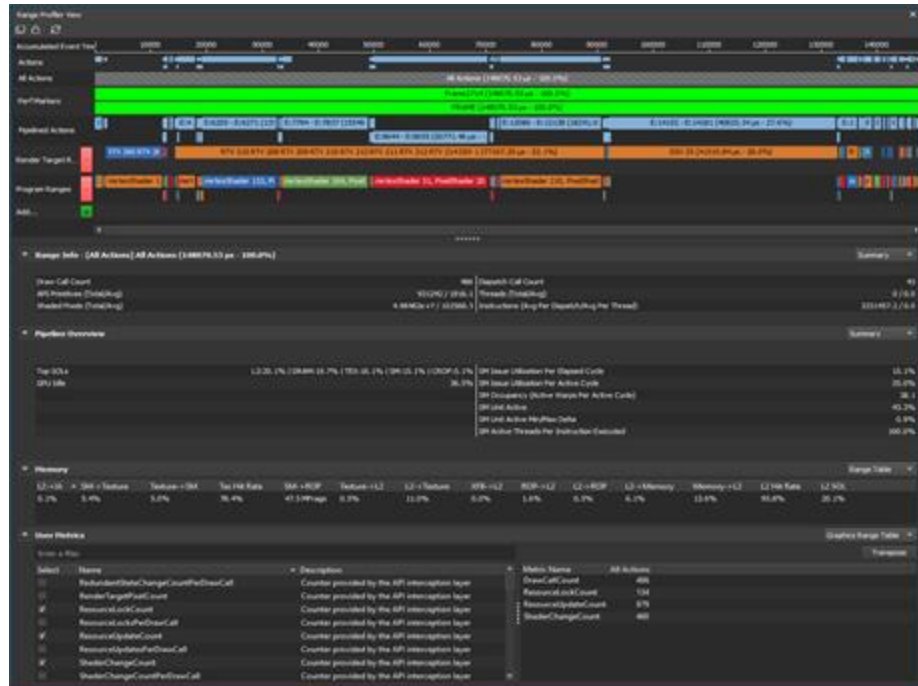
Root Arguments

Filter: Enter a filter

Heap Entry	ViewDesc	Resource	Resident
2135	SRV Descriptor: Format: 01010101 ViewDimension: 01010100 Shader4ComponentElements: 01001101 NumElements: 01010101 StructureByteStride: 6	Buffer 6	TRUE

Callable Table

Demo 3 - Profiling



“The Peak-Performance Analysis Method for Optimizing Any GPU Workload”

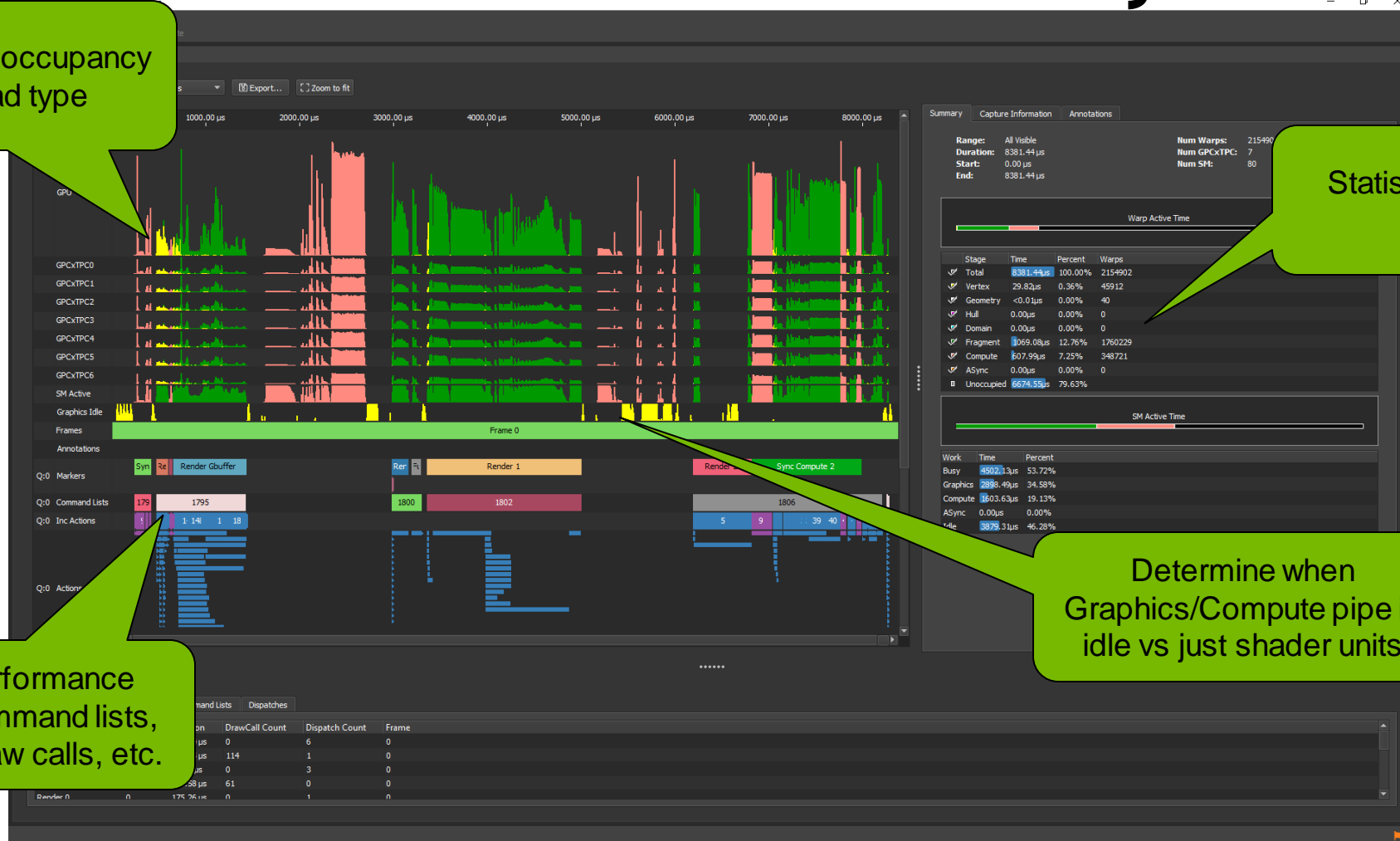
GPU Trace: How filled are my SMs?

Visualize warp occupancy by workload type

Statistics for current selection

Display performance markers, command lists, individual draw calls, etc.

Determine when Graphics/Compute pipe is idle vs just shader units

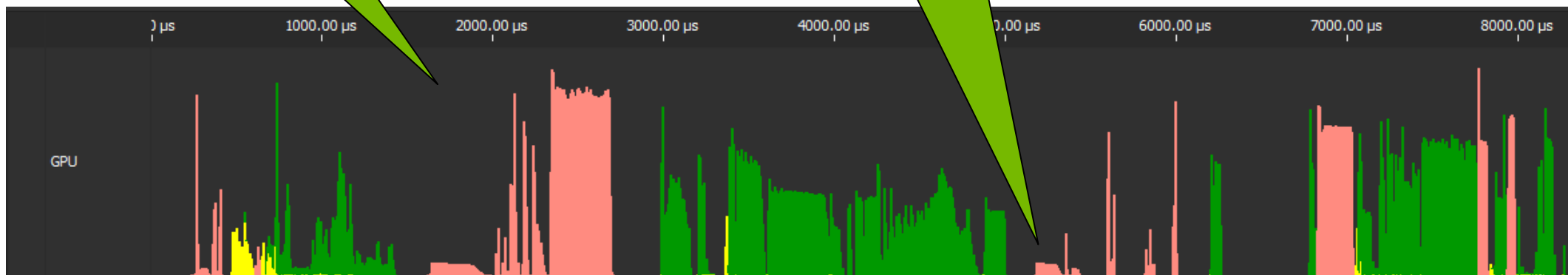


GPU Trace: How filled are my SMs?

Note graphics and compute work occurring on a single, Direct Queue

Not achieving optimal occupancy of ~65-70%

Frame time is just over 8ms



Solution? Use a Compute Queue

Construct Compute Queue

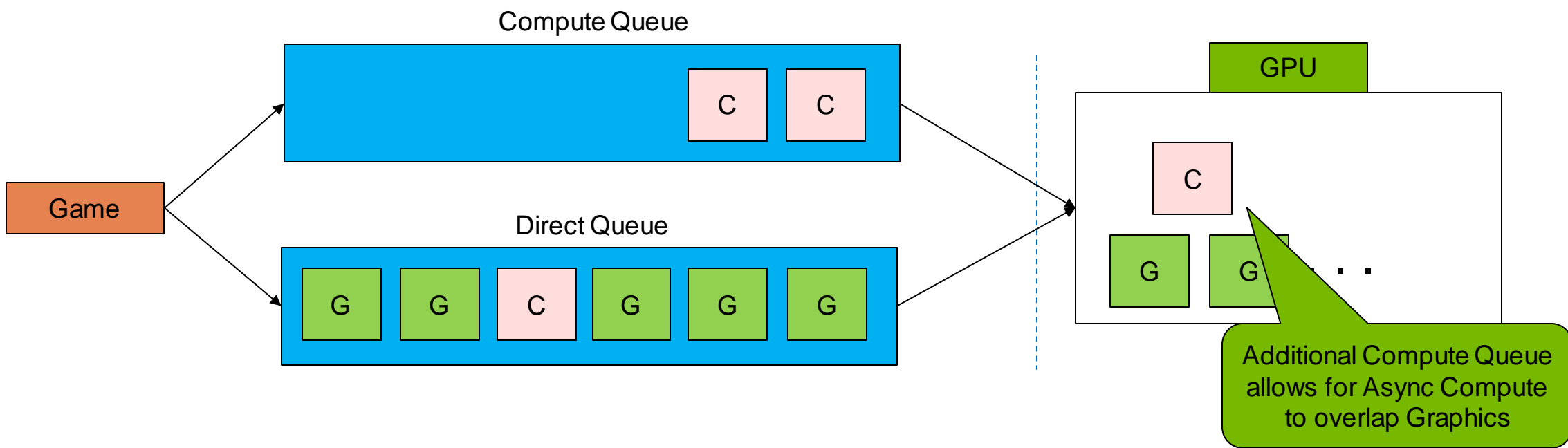
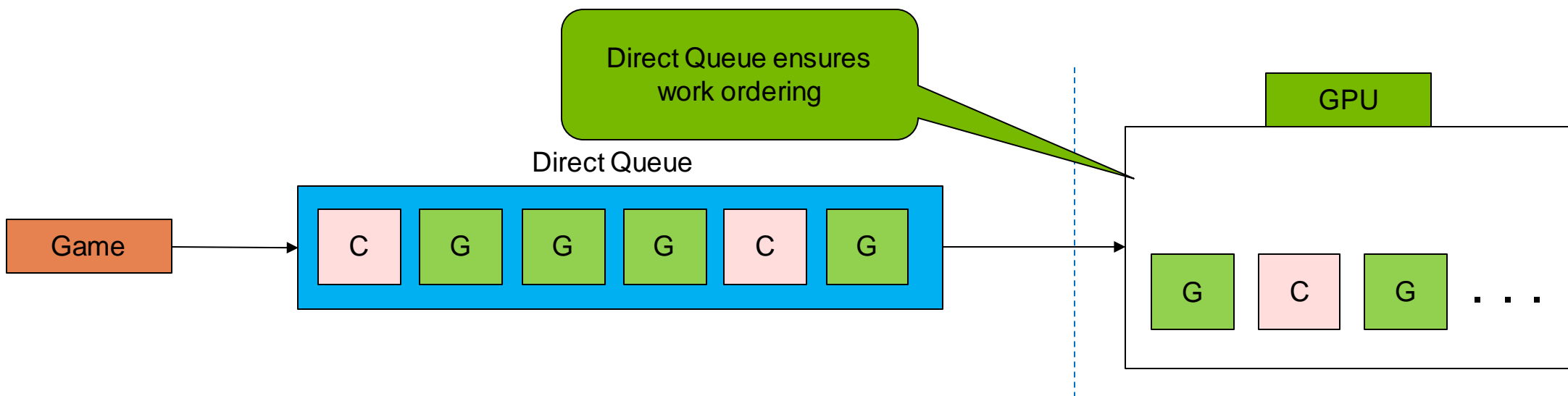
```
#include <d3d12.h>

//
// Example of creating a compute command queue, needed to get any async compute
//
D3D12_COMMAND_QUEUE_DESC computeQueueDesc =
{ D3D12_COMMAND_LIST_TYPE_COMPUTE, // Type
  0, // Priority
  D3D12_COMMAND_QUEUE_FLAG_NONE, // Flags
  0x0 // NodeMask
};

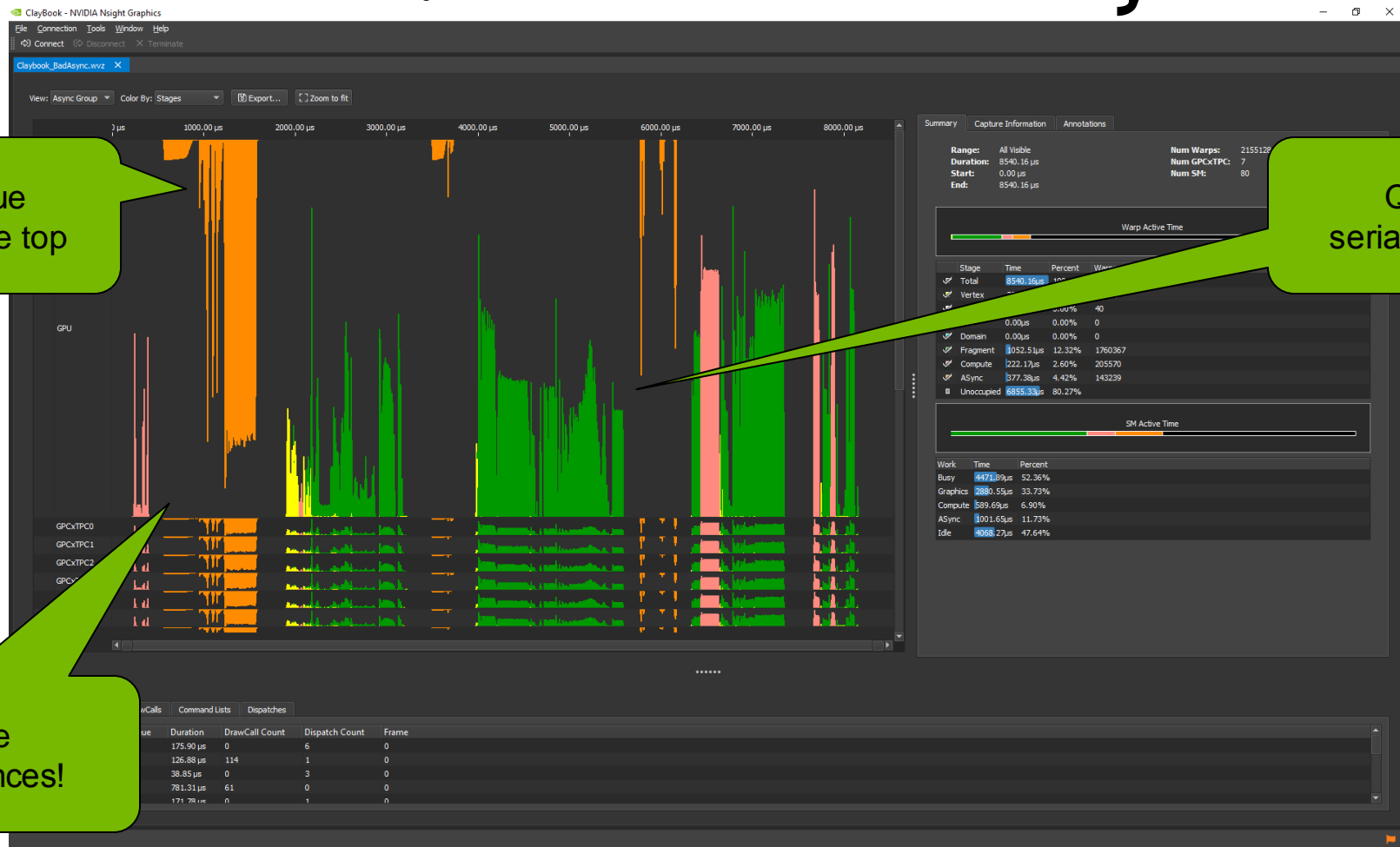
// Create a compute command queue
hResult = pD3DDevice->CreateCommandQueue(&computeQueueDesc, __uuidof(ID3D12CommandQueue), (void**) &pD3DComputeCommandQueue);

// Create a compute command allocator
hResult = pD3DDevice->CreateCommandAllocator(D3D12_COMMAND_LIST_TYPE_COMPUTE, __uuidof(ID3D12CommandAllocator),
(void**) &pComputeCommandAllocator);

// Create a compute command list
hResult = pD3DDevice->CreateCommandList(0x1, D3D12_COMMAND_LIST_TYPE_COMPUTE, pComputeCommandAllocator, NULL,
__uuidof(ID3D12GraphicsCommandList), (void**) &pComputeCommandList);
```



GPU Trace: How filled are my SMs?

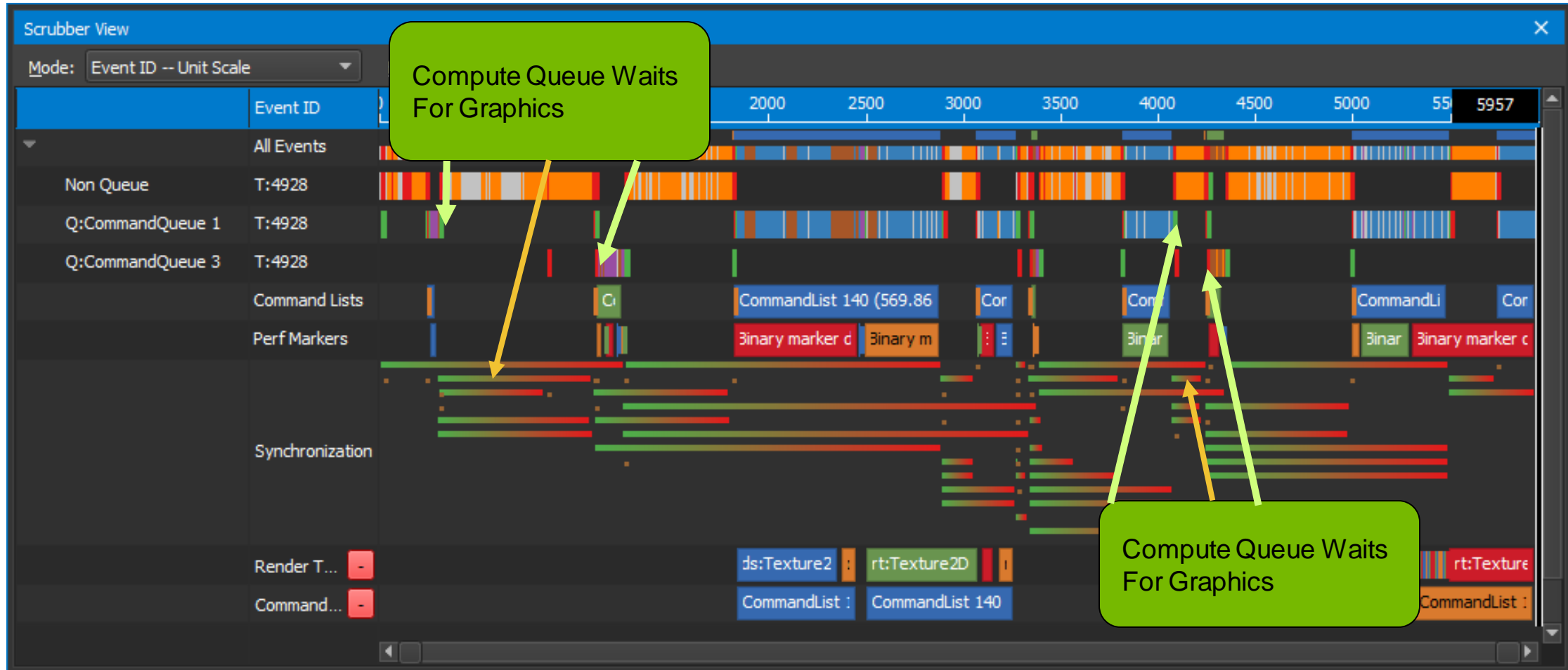


Compute Queue visualizes from the top

Queues are serialized...fences!

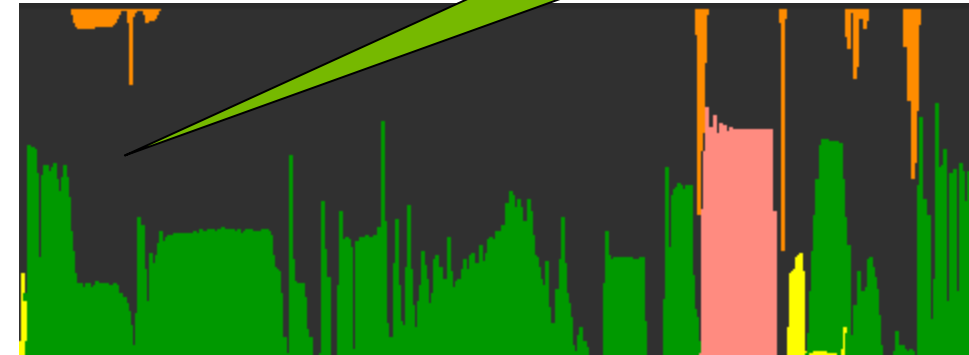
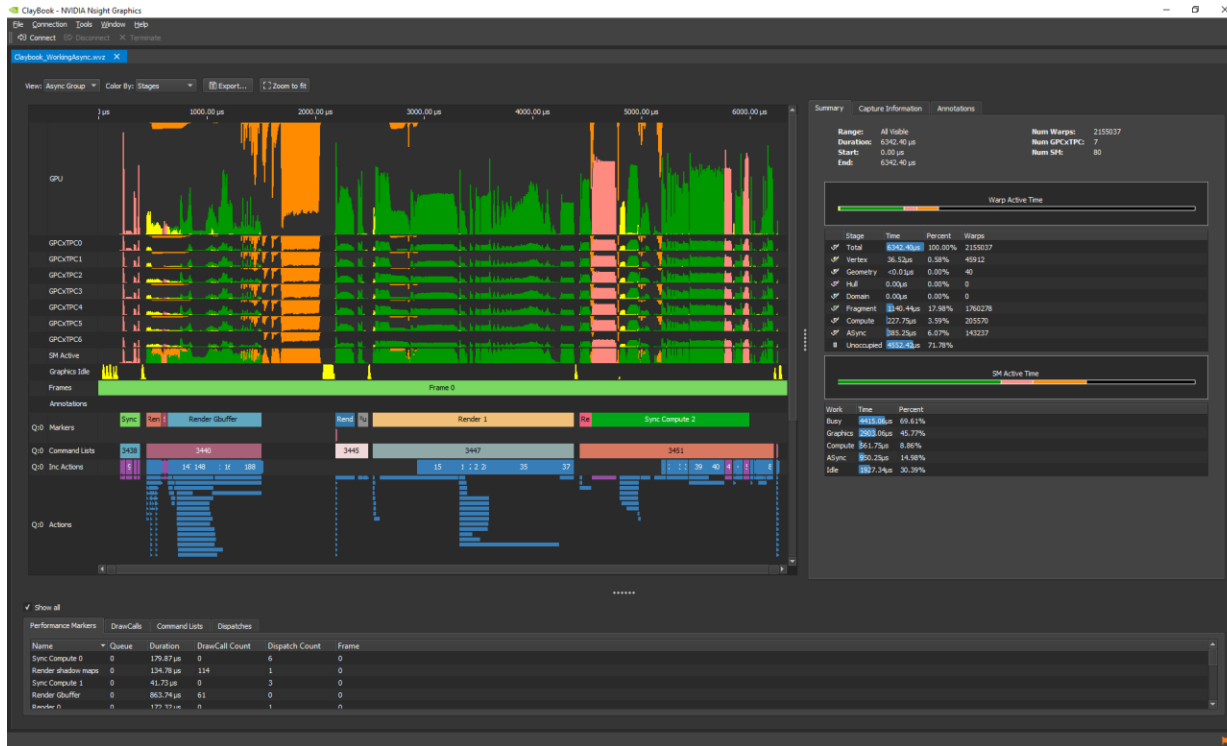
Queues are serialized...fences!

No (Real) Async: Debug with Nsight Graphics

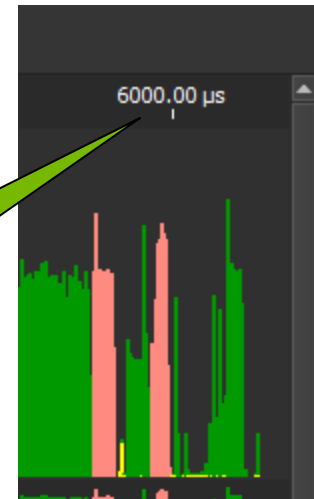


GPU Trace: Much Better!

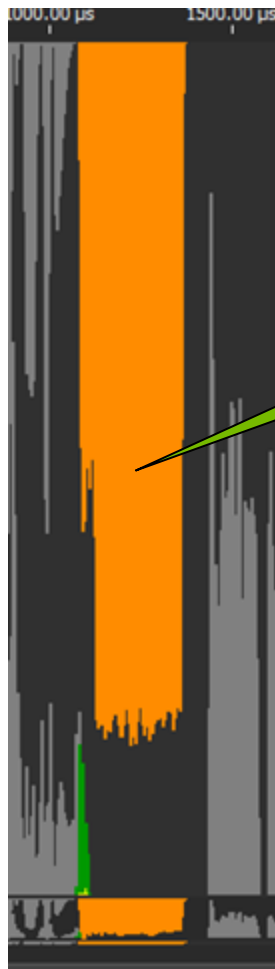
Graphics & Compute now overlap...better shader unit utilization!



Frame time now just over 6ms...~2ms saved!

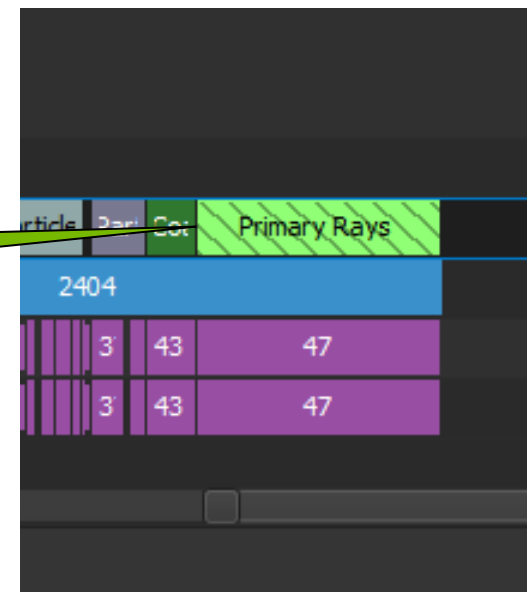


But wait...why is compute not full?

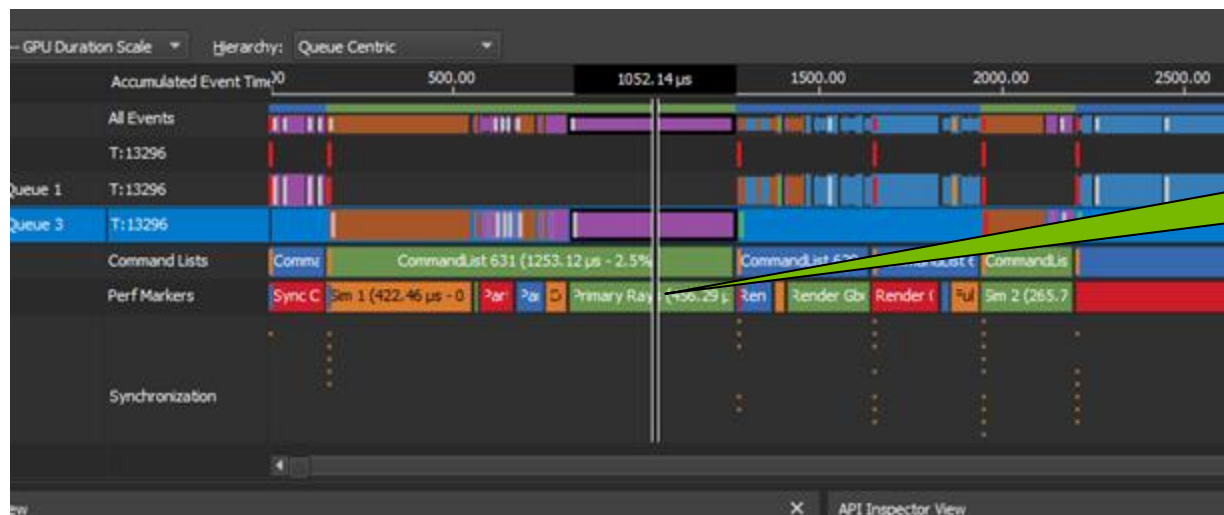


Compute work doesn't
"fill" the GPU...why?

Comes from "Primary
Rays" Performance
Marker Region

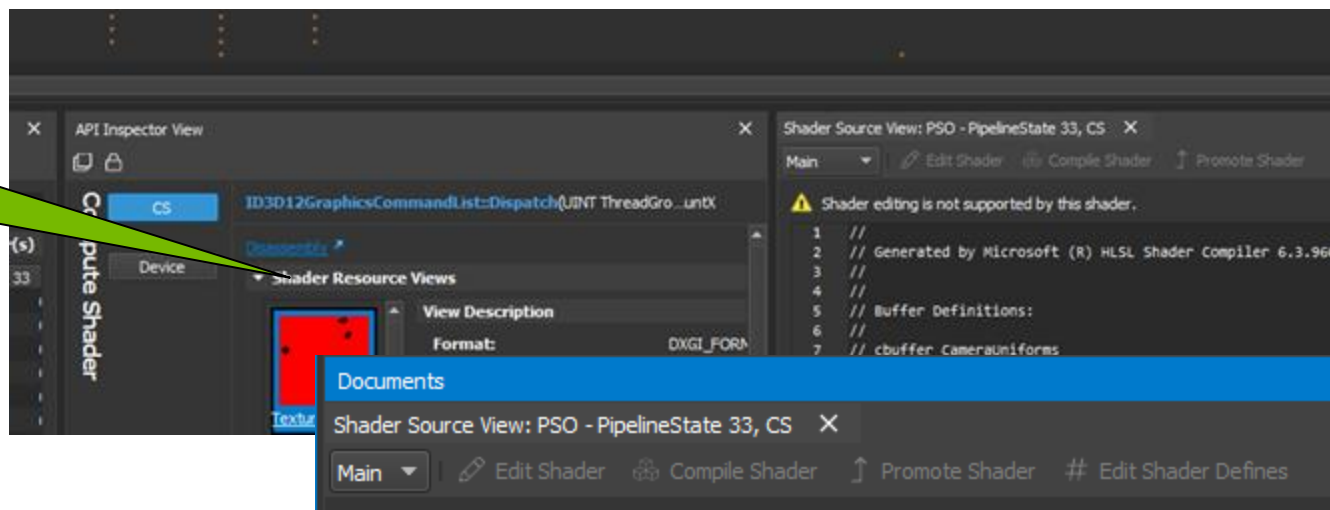


GPU Not Full: Debug with Nsight Graphics...



Find “Primary Rays” section in Scrubber

Identify shader in Scrubber/Source View

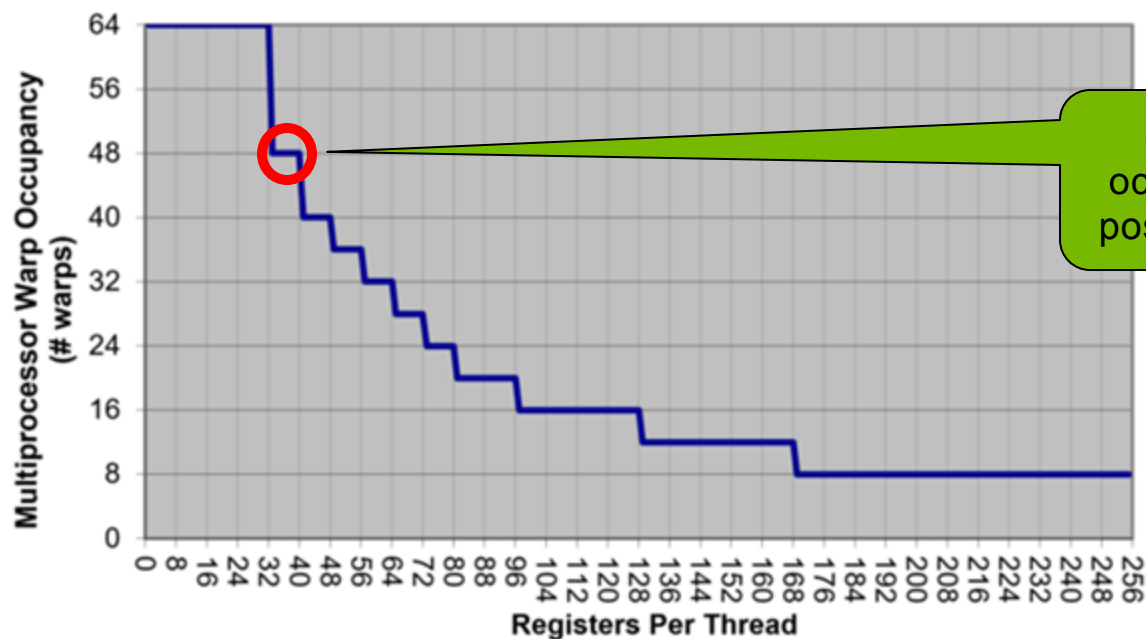


GPU Not Full: Debug with Nsight Graphics...

▶ PipelineState 31	PSO	Source + µCode	None	0	0	--	--	32
▶ PipelineState 32	PSO	Source + µCode	None	0	0	--	--	24
▶ PipelineState 33	PSO	Source + µCode	None	0	0	--	--	32
▶ PipelineState 34	PSO	Source + µCode	None	0	0	--	--	40

Shader uses 32 registers

Impact of Varying Register Count Per Thread



32 registers limits occupancy to 48 of 64 possible warps...~80%

GPU Throughput: SOL

SOL = Speed of Light

< ~10%, unit just not taxed...

0-60%

60-80%

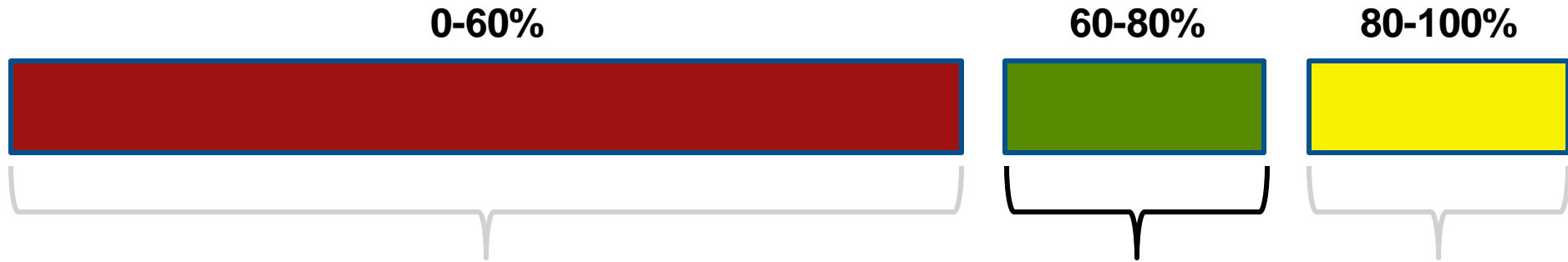
80-100%



Unit not achieving maximum throughput, analyze other metrics to determine why

GPU Throughput: SOL

SOL = Speed of Light



Grey area: can try to make unit more efficient, or could try and reduce work

GPU Throughput: SOL

SOL = Speed of Light



Unit is efficient but near max throughput, will likely need to shift work to other units improve performance

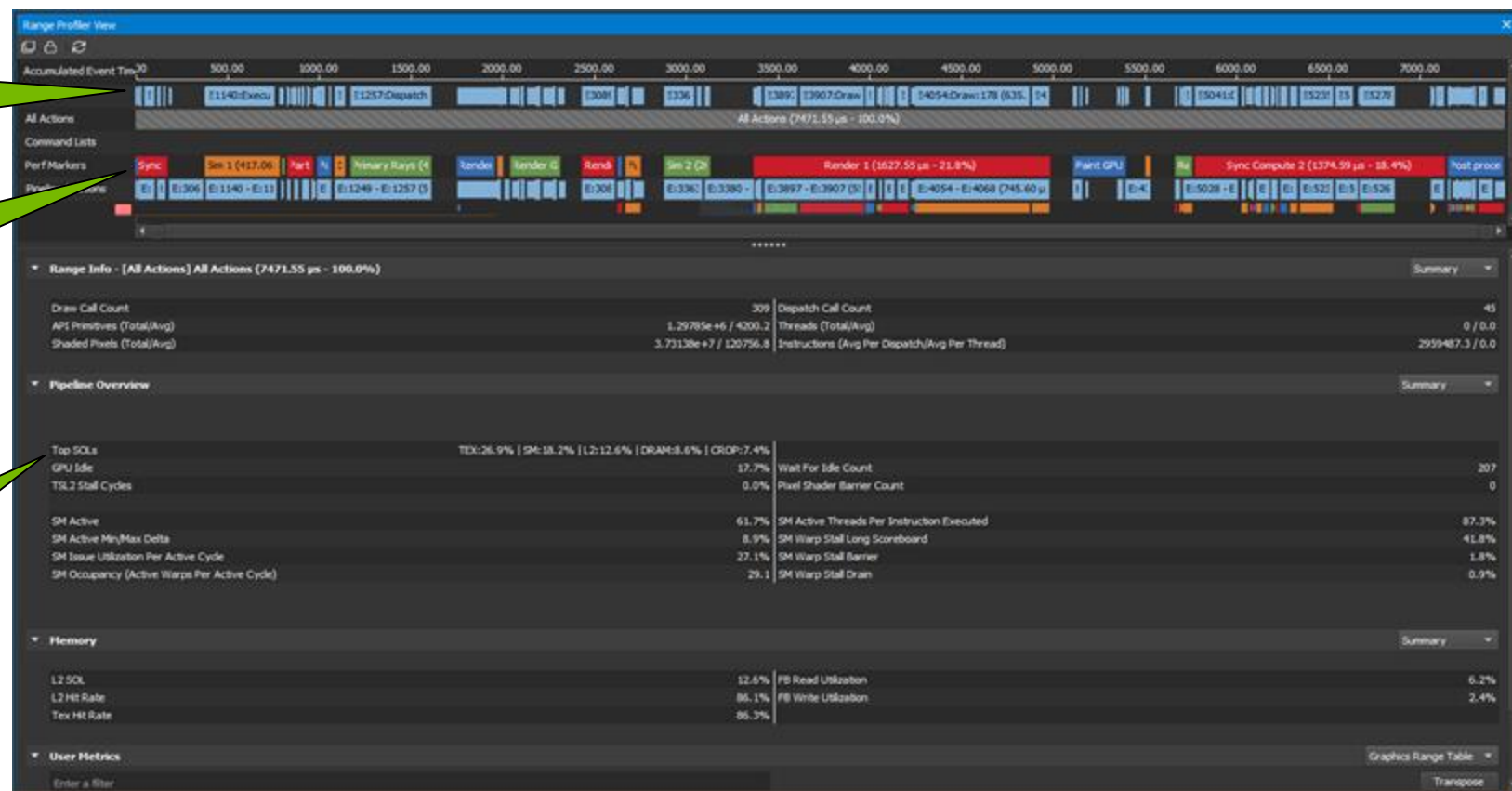
Understanding GPU Performance

Range Profiler

GPU time for every draw/dispatch

Elapsed time for performance marker ranges

Pipeline throughput/SOL values



Understanding GPU Performance

Range Profiler

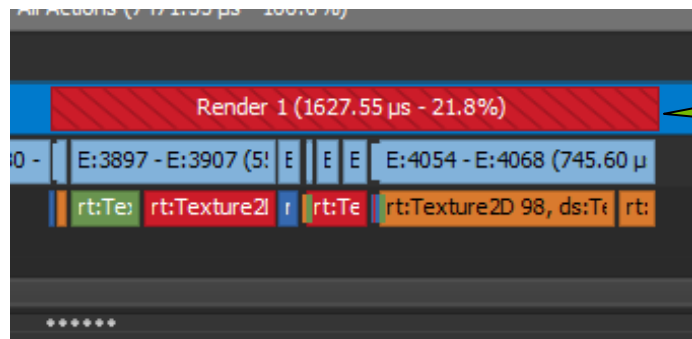
Overall frame is mainly texture limited, but better to dig into ranges that have similar workloads

TEX: 26.9% | SM: 18.2% | L2: 12.6% | DRAM: 8.6% | CROP: 7.4%

Top SOLs	TEX:26.9% SM:18.2% L2:12.6% DRAM:8.6% CROP:7.4%		
GPU Idle	17.7%	Wait For Idle Count	207
TSL2 Stall Cycles	0.0%	Pixel Shader Barrier Count	0
SM Active	61.7%	SM Active Threads Per Instruction Executed	87.3%
SM Active Min/Max Delta	8.9%	SM Warp Stall Long Scoreboard	41.8%
SM Issue Utilization Per Active Cycle	27.1%	SM Warp Stall Barrier	1.8%
SM Occupancy (Active Warps Per Active Cycle)	29.1	SM Warp Stall Drain	0.9%

Understanding GPU Performance

Range Profiler



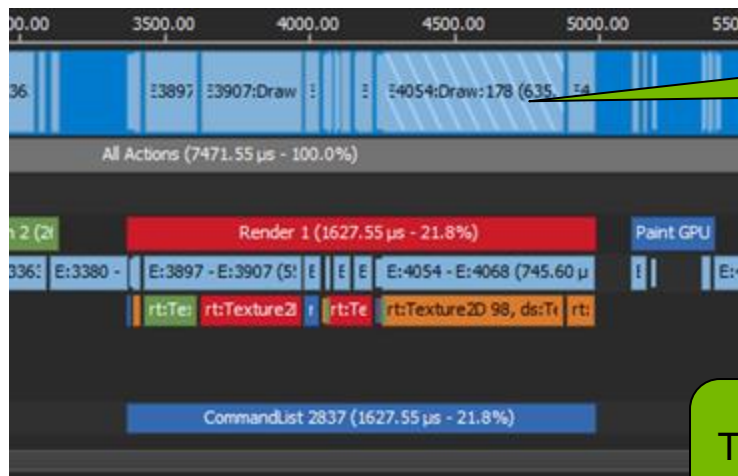
Select largest/most expensive marker range in scene...

This range is mainly texture and shader unit limited...but dig a little deeper...

TEX: 45.2% | SM: 29.5% | L2: 23.1% | DRAM: 8.4% | CROP: 6.0%

Top SOLs	TEX:45.2% SM:29.5% L2:23.1% DRAM:8.4% CROP:6.0%		
GPU Idle	11.7%	Wait For Idle Count	9
TSL2 Stall Cycles	0.0%	Pixel Shader Barrier Count	0
SM Active	74.9%	SM Active Threads Per Instruction Executed	89.4%
SM Active Min/Max Delta	11.8%	SM Warp Stall Long Scoreboard	33.4%
SM Issue Utilization Per Active Cycle	33.4%	SM Warp Stall Barrier	0.0%
SM Occupancy (Active Warps Per Active Cycle)	25.4	SM Warp Stall Drain	0.0%

Understanding GPU Performance

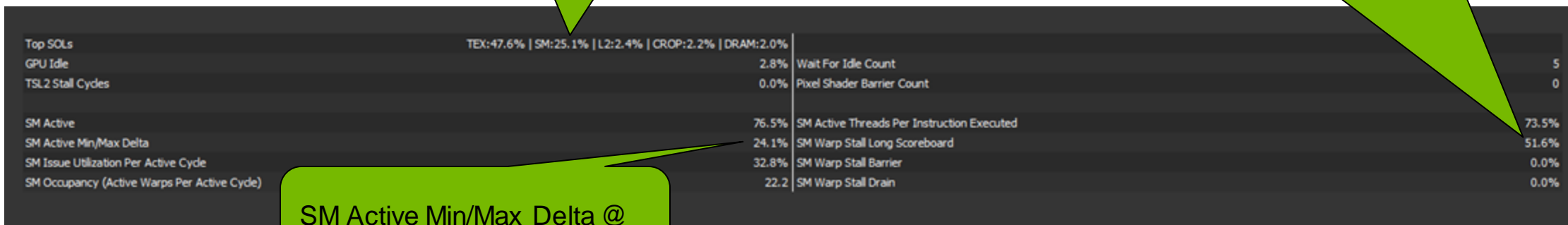


G-Buffer resolve...

Texture & Shader throughput low...

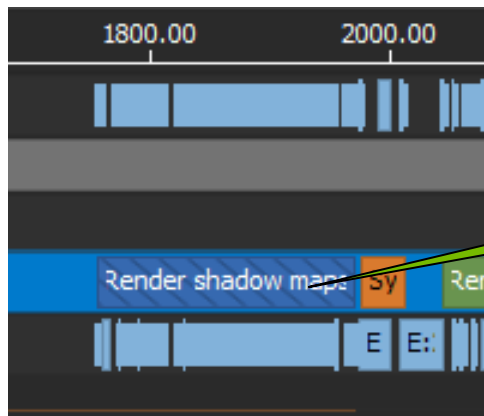
SM Active Threads Per Instruction Executed = 73.5%
SM Warp Stall Long Scoreboard = 51.6%

Shader unit waiting for texture fetches to return



SM Active Min/Max Delta @ 24.1% => shader divergence

Understanding GPU Performance



Shadow Map Rendering,
typically vertex and z limited

Top SOLs – typical signature for vertex limited work

PD: Primitive Distributor (vertex work)

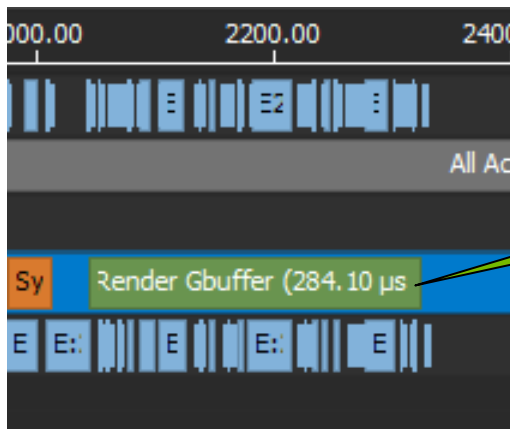
VPC: Viewport Cull/Clip (vertex work)

ZROP: Z “blending” (Z buffering)

PD: 61.8% | VPC: 28.9% | ZROP: 14.5% | L2: 7.8% | DRAM: 6.1%

Top SOLs	PD: 61.8% VPC: 28.9% ZROP: 14.5% L2: 7.8% DRAM: 6.1%	
GPU Idle	10.6%	Wait For Idle Count 8
TSL2 Stall Cycles	0.0%	Pixel Shader Barrier Count 0
SM Active	69.0%	SM Active Threads Per Instruction Executed 87.4%
SM Active Min/Max Delta	4.5%	SM Warp Stall Long Scoreboard 20.7%
SM Issue Utilization Per Active Cycle	6.4%	SM Warp Stall Barrier 0.0%
SM Occupancy (Active Warps Per Active Cycle)	6.8	SM Warp Stall Drain 0.4%

Understanding GPU Performance



Render Gbuffer typically color
blending texture, and shader limited

Top SOLs

CROP/ZROP: Blending unit (blending the 6 “fat” render targets & depth)

TEX: Texture unit (reading in diffuse textures, etc.)

SM: Shader unit (calculating Gbuffer values)

CROP: 43.4% | ZROP: 27.7% | TEX: 24.0% | SM: 19.0% | DRAM: 15.1%

Top SOLs	CROP:43.4% ZROP:27.7% TEX:24.0% SM:19.0% DRAM:15.1%	
GPU Idle	4.0%	Wait For Idle Count 5
TSL2 Stall Cycles	0.0%	Pixel Shader Barrier Count 0
SM Active	64.9%	SM Active Threads Per Instruction Executed 92.3%
SM Active Min/Max Delta	11.0%	SM Warp Stall Long Scoreboard 27.0%
SM Issue Utilization Per Active Cycle	23.4%	SM Warp Stall Barrier 0.0%
SM Occupancy (Active Warps Per Active Cycle)	11.9	SM Warp Stall Drain 0.2%

Key Takeaways

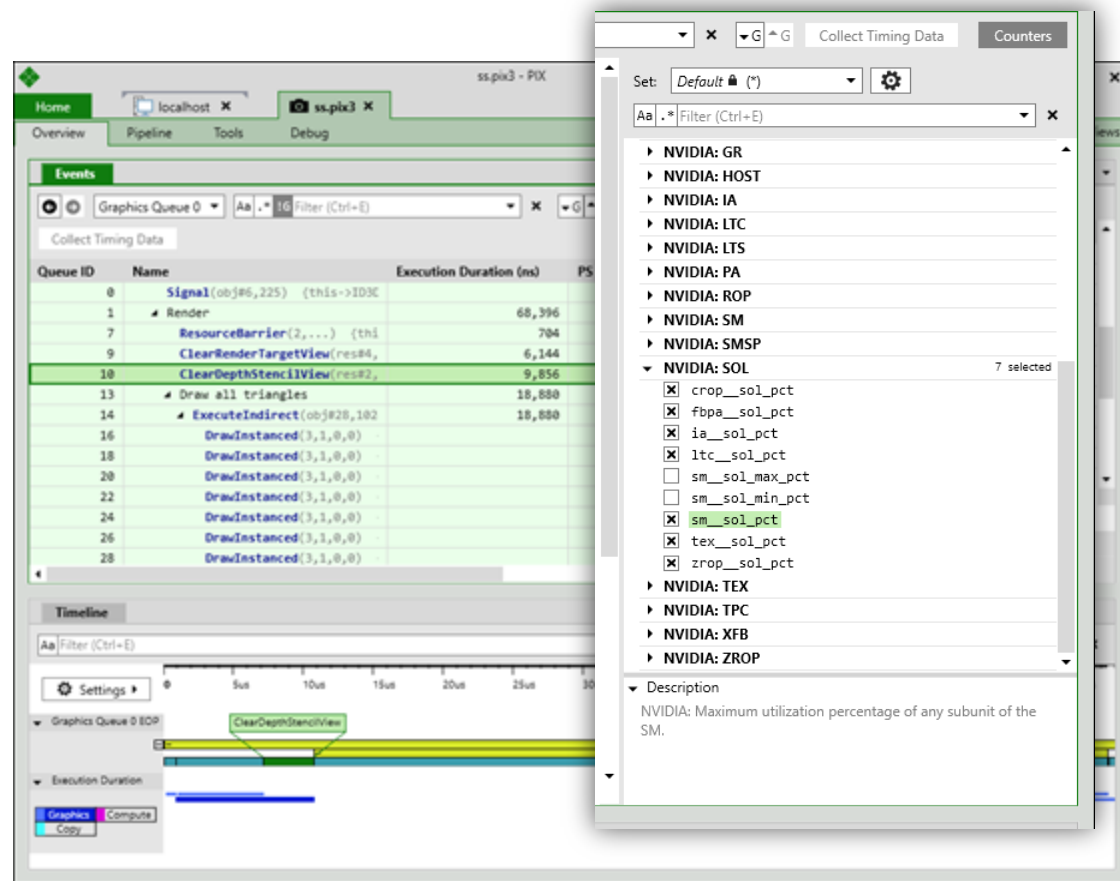
Summary

- **Nsight Graphics** is *the* next-gen graphics debugger
 - DXR Debugging
 - Range Profiler
 - GPU Trace
- **NVIDIA DevTools** is committed to...
 - increasing release frequency
 - solving developer pain points
 - improving developer productivity



Microsoft PIX for Windows

- Close collaboration to help make the best tools for our developers
 - Work history buffer i.e. timing activity
 - HW Performance Counters
 - GPU Occupancy Viewer
 - Performance warnings
 - DXR visualization



The Future of Nsight Graphics

- H1 '18
 - Vulkan 1.1 support
 - Linux Support
 - Improvements to Range Profiler
 - GPU Trace 1.0
 - DXR improvements
- H2 '18
 - GPU Trace for Vulkan
 - Pixel History (DX12 & Vulkan)
 - DXR Profiling

What's Next?

- Visit our Booth (#223) to check out *Nsight Graphics*, *DXR Debugging* and *GPU Trace*
- Check out Louis Bavoil's talk in 30 minutes to learn more about performance triaging with the *Range Profiler*
- Watch our Videos: <http://j.mp/nvidia-devtools-videos>

Join our Early Access Program
and get

Nsight Graphics here:

<https://j.mp/ngfx>

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

We're Hiring!

<http://www.nvidia.com/object/careers.html>

FORTUNE
WORLD'S MOST
ADMIRED
COMPANIES
2016

Forbes | 2016
AMERICA'S
BEST LARGE
EMPLOYERS
POWERED BY STATISTA



glassdoor
2017
BEST
PLACES
to work
Employees' Choice



BayArea
NewsGroup

"NVIDIA has made a major jump from using technology to create entertainment to using it to alter our reality and change the world."

-Rob Enderle, *Industry Analyst*



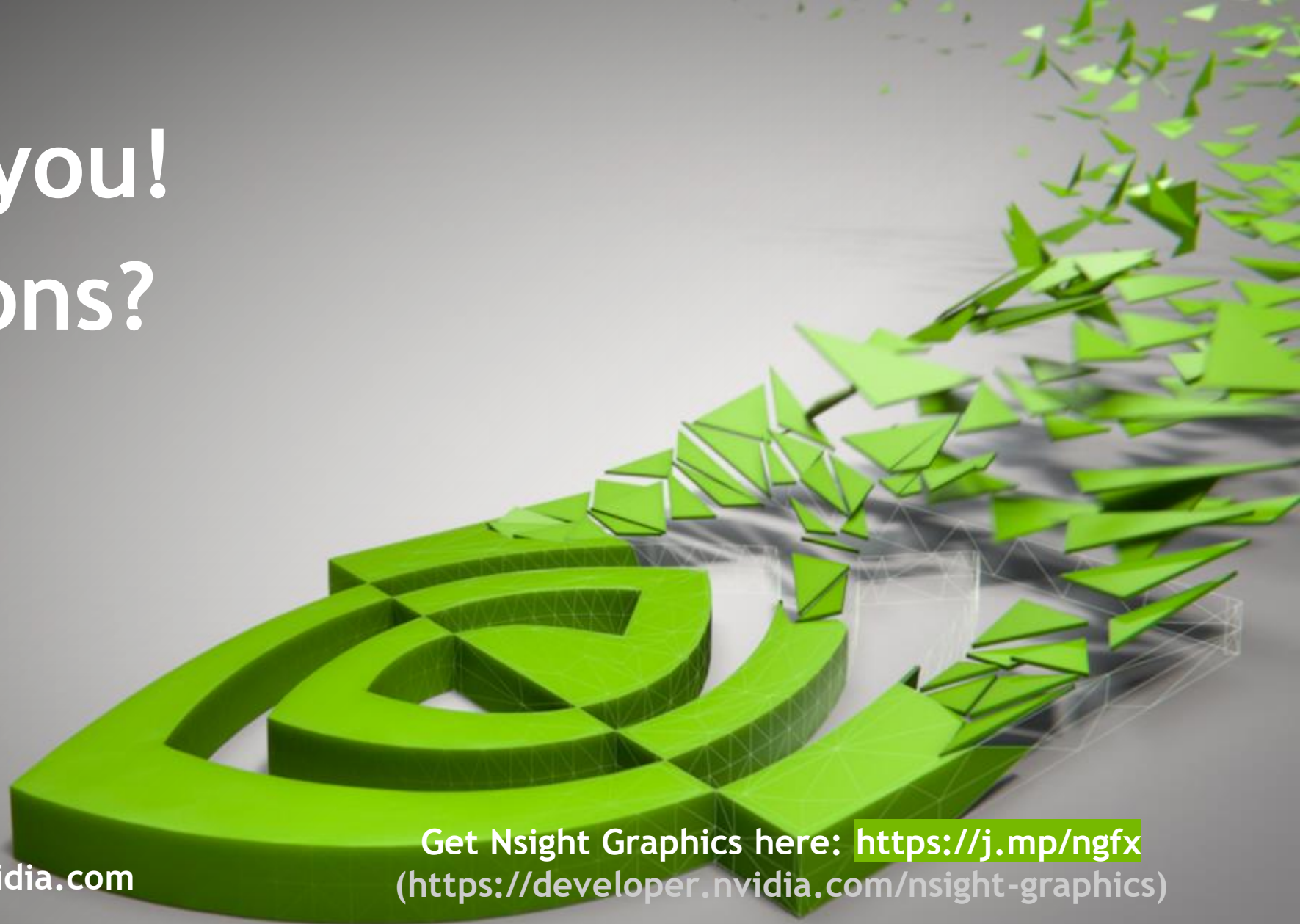
Thank you! Questions?



Booth #223 - South Hall
www.nvidia.com/GDC

Need to contact us?
NsightGraphics@nvidia.com

Get Nsight Graphics here: <https://j.mp/ngfx>
(<https://developer.nvidia.com/nsight-graphics>)



NVIDIA Sessions

Day	Time	Room	Speaker	Title
Thurs Mar 21	17:30-18:30	3022	Louis Bavoil Principal Engineer	Fixing the Hyperdrive - Maximizing Rendering Performance on NVIDIA GPUs
Fri Mar 23	10:00-11:00	3022	Nuno Subtil Senior Developer Technology Engineer	NVIDIA Vulkan Update
Fri Mar 23	11:30-12:00	3001, 3003	Alex Dunn Senior Developer Technology Engineer	Aftermath – Advances in GPU Crash Debugging
Fri Mar 23	12:15-13:15	3022	Bryan Dudash Senior Manager Developer Technology	Capture Amazing Content with NVIDIA Ansel Photo Mode and Highlights Video Capture Tool
Fri Mar 23	13:30-14:30	3022	Evan Hart, Principal Engineer	Advances in the HDR Eco-System
Fri Mar 23	15:00-16:00	3022	Cem Cebenoyan Director of Engineering	Accelerating your VR Games with VRWorks

Need to contact us?
NsightGraphics@nvidia.com

Get Nsight Graphics here: <https://j.mp/ngfx>
(<https://developer.nvidia.com/nsight-graphics>)