



# BRIGHTER & FASTER GRAPHICS WITH NVIDIA® NSIGHT™ VSE 4.5™

Jeff Kiel, Manager Graphics Tools

Russ Kerschner, Sr. SW Engineer

Developer Tools: NVIDIA GameWorks™



Introducing NVIDIA® Nsight™ Visual Studio Edition 4.5™

## AGENDA

Profiling Infiltrator from Epic Games

Sneek Peek at DirectX12 support in 5.0!

Thanks for our friends at



Developing on modern GPUs can be a cloudy business...



Shader Bugs

GPU Bottlenecks

API State Management

Performance

API Usage

Driver

CPU<->GPU Interaction

Memory

GPU Frametime

Resources

Shader Modification

Contributing Fragments



# NVIDIA DEVELOPER TOOLS

BUILD. DEBUG. PROFILE.

Microsoft  
DirectX

OpenGL

nVIDIA  
CUDA

OpenGL|ES



GNU  
C/C++

IDE INTEGRATION

Visual Studio

eclipse

STANDALONE TOOLS



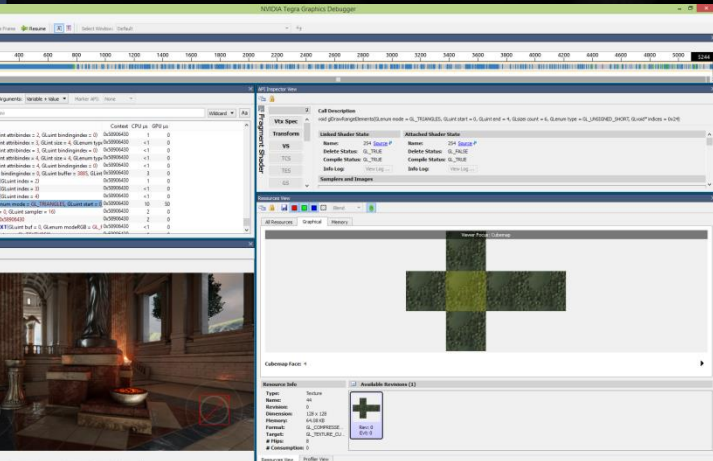
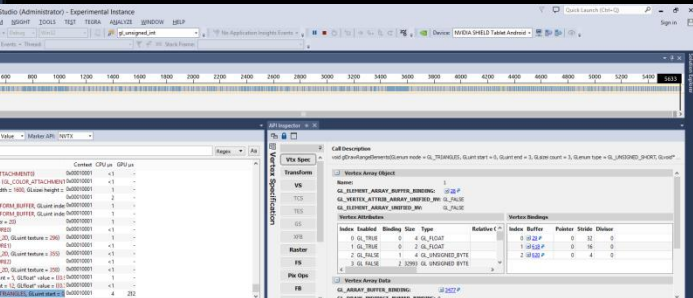
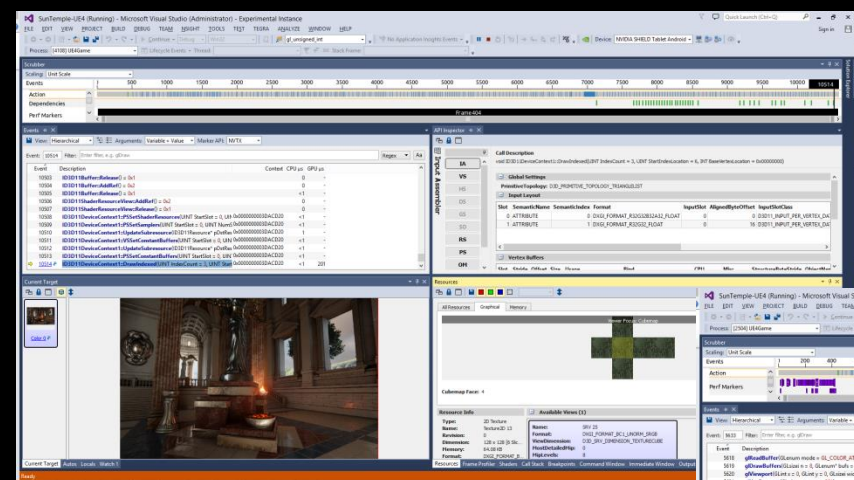
HARDWARE SUPPORT  
CPU AND GPU DEBUGGING & PROFILING



nVIDIA  
GAMEWORKS™



# PICK A PLATFORM/API

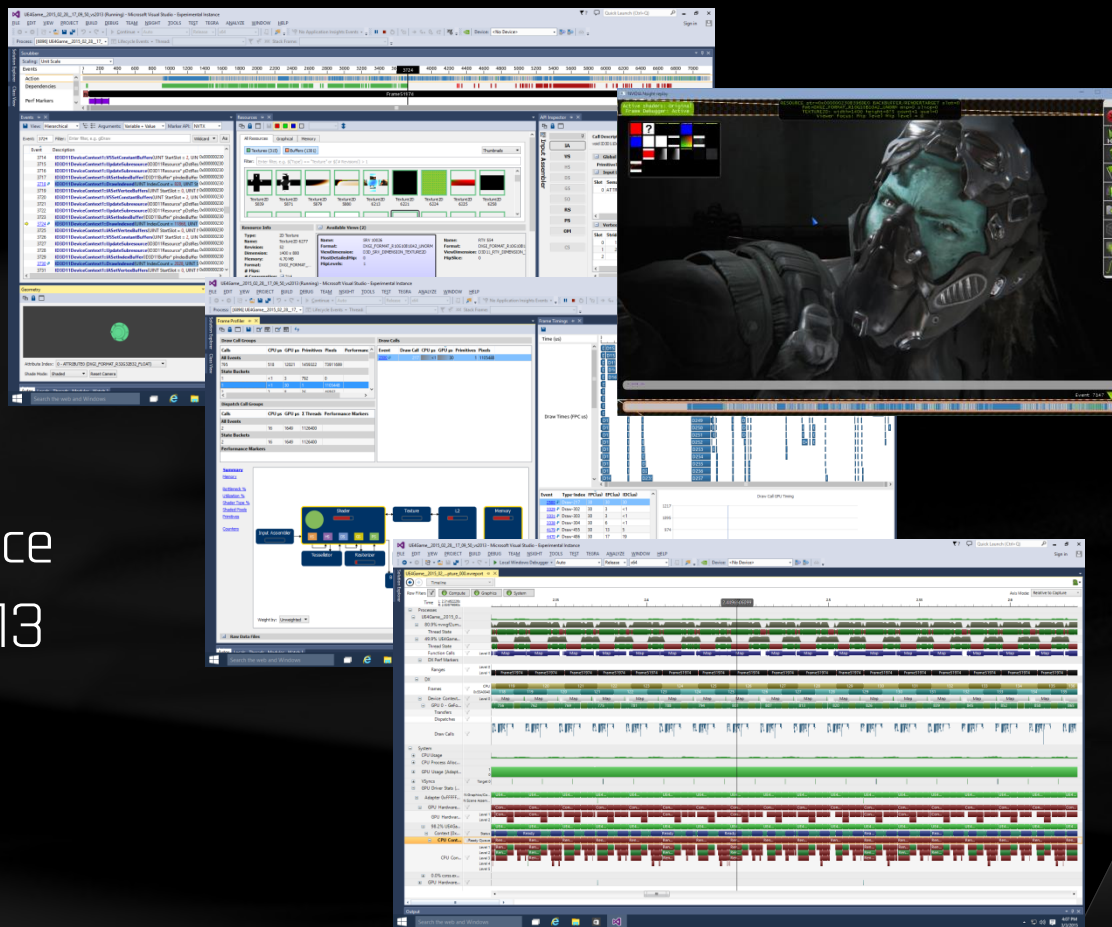




# NVIDIA® NSIGHT™ VISUAL STUDIO EDITION

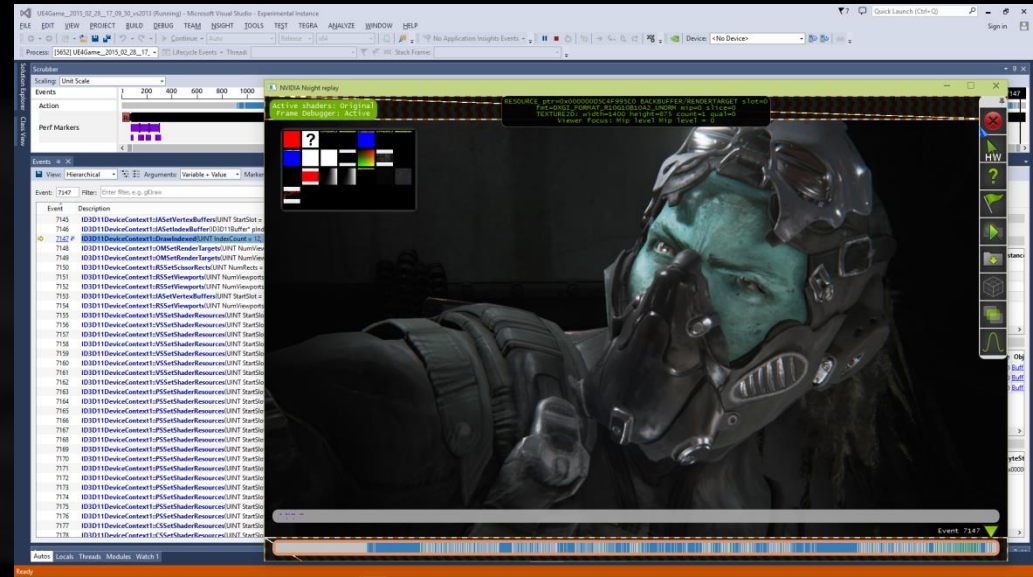
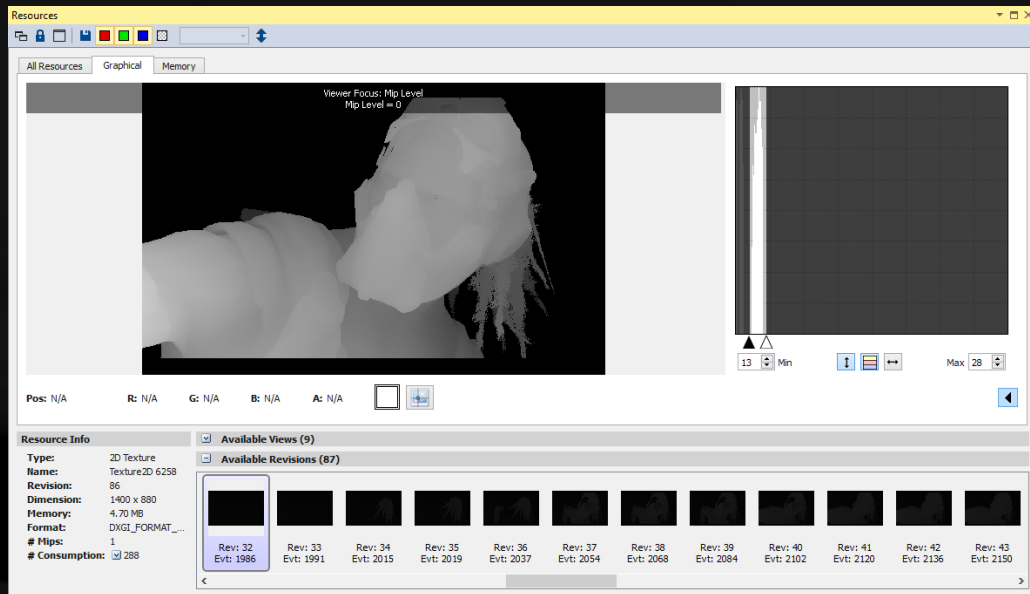
## Accelerating Visual Computing Development

- Supports DirectX 9/11 and OpenGL
- Debug and profile graphics workloads
- Debug HLSL and GLSL shaders
- Debug and profile CUDA kernels
- Platform level profiling with system trace
- All in Visual Studio 2010, 2012 and 2013



# NEW IN VERSION 4.5

- Maxwell architecture support (minus shader debugging)
- OpenGL 4.4 (minus ARB\_sparse\_texture)
- OpenGL capture with source code generation
- Texture histogram and range remapping
- CUDA 7.0 support



# PROFILING INFILTRATOR

Active shaders: Original  
Frame Debugger: Active

```
RESOURCE ptr=0x00000000D0B1BD0 BACKBUFFER/RENDERTARGET slot=0  
fmt=D3D11_FORMAT_R10G10B10A2_UNORM mip=0 slice=0  
TEXTURE2D: width=1400 height=875 count=1 qual=0  
Viewer Focus: Mip level Mip level = 0
```

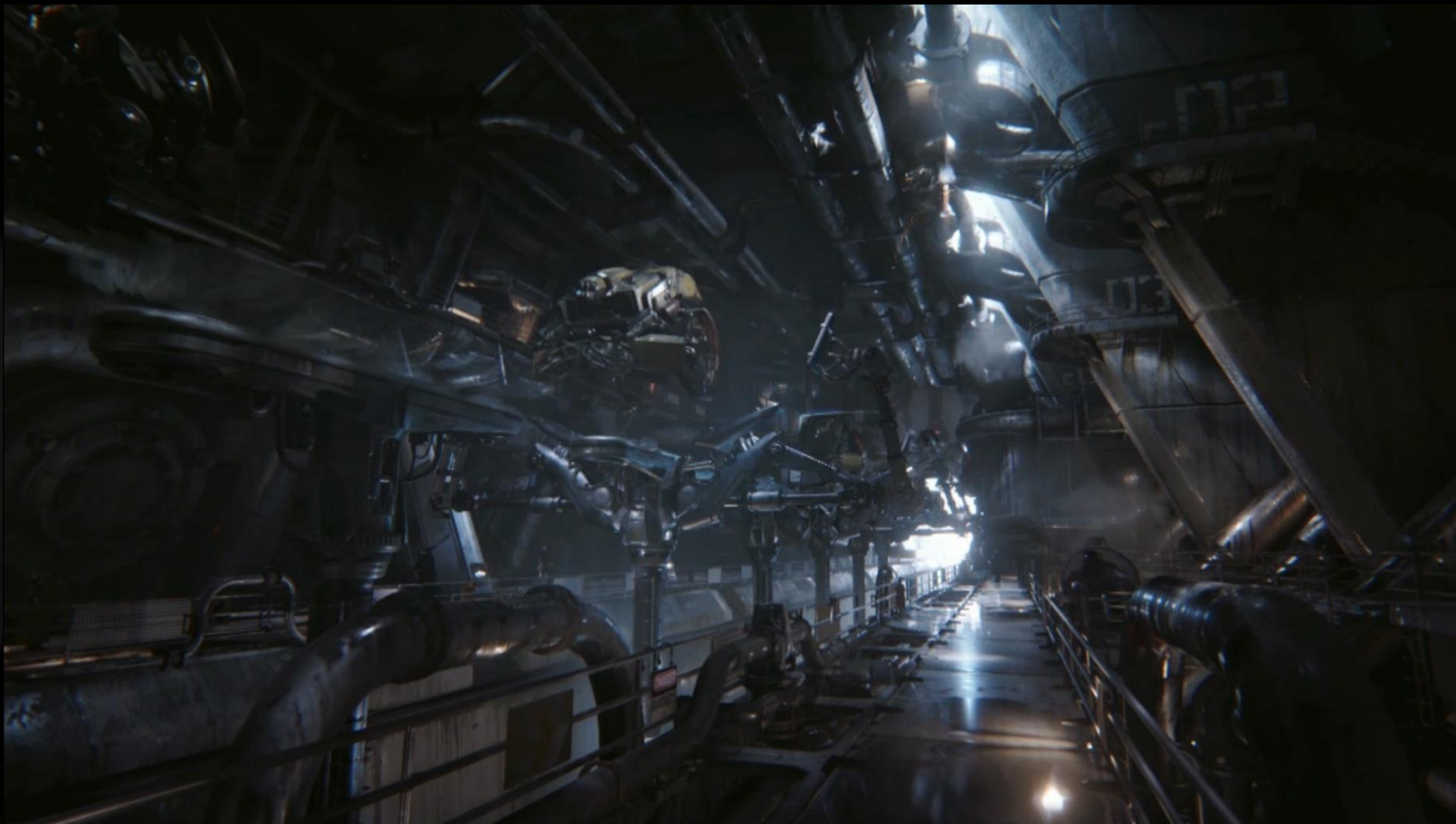
Event 37932





# PROFILING INFILTRATOR

View Infiltrator at <https://www.youtube.com/watch?v=d02rM-l-vdQ>



# PROFILING INFILTRATOR - CPU BOUND

The screenshot shows the Visual Studio interface with the Application Settings window open. The Application Name is 'InfiltratorDemo150...capture\_000.nvreport'. The Application path is 'C:\Users\jkiehl\Desktop\GDC2015\Infiltrator\_Capture\_GPUBound\x64\Release\UE4Game\_2015\_02\_28\_17\_09\_50.exe'. The Working Directory is 'C:\Users\jkiehl\Desktop\GDC2015\Infiltrator\_Capture\_GPUBound\'. The Activity Type is 'Trace Process Tree'. Under Trace Settings, 'DirectX' is selected, and 'API Categories' is set to 'All Categories'. Under Workload Trace, 'CPU Frames', 'GPU Frames', 'Push Buffers', 'Draw Calls', 'Dispatches', and 'Transfers' are all checked. The Session Control section shows a red 'Start' button and a 'Stop' button. The Capture Control section shows a red 'Start' button, 'Stop', and 'Cancel' buttons, with 'Open Report on Stop' checked and 'Summary Report' selected in the dropdown.

Application Settings

Trace System & DirectX

Launch!



# PROFILING INFILTRATOR - CPU BOUND

Activity1.nvact\* InfiltratorDemo150...apture\_000.nvreport

Summary Report

Target Process: UE4Game.exe [2464] Clear

### Session Overview

Summary of session information related to the captured data.

- Session Summary
- Timeline
- Activity

### InfiltratorDemo.exe (Trace Process Tree)

Captured 9.24 seconds of data on 2/28/2015 6:46:04 PM

Arguments:  
Working Dir: c:\users\jkiel\desktop\infiltrator\windowsnoeditor  
Connection: DTPNCWin764-02

### Process Overview

Summary of process information relating to the captured data.

Process Name: UE4Game.exe  
Process ID: 2464  
Command Line: "c:\users\jkiel\desktop\infiltrator\windowsnoeditor\Engine\Binaries\Win64\UE4Game.exe" ..\..\..\InfiltratorDemo\InfiltratorDemo.uproject  
Process Tree: UE4Game.exe [2464]  
# Threads: 35

### DirectX Overview

Summary of captured DirectX activity.

- API Calls | Summary
- Draw Calls
- Frames
- Dispatches
- Performance Markers
- Transfers

CPU All Cores			
API Time (%)	Min	Avg	Max
<b>51.2</b>	API/DC 31	33.3	42
	Batch 2,106	2,492.8	3,727

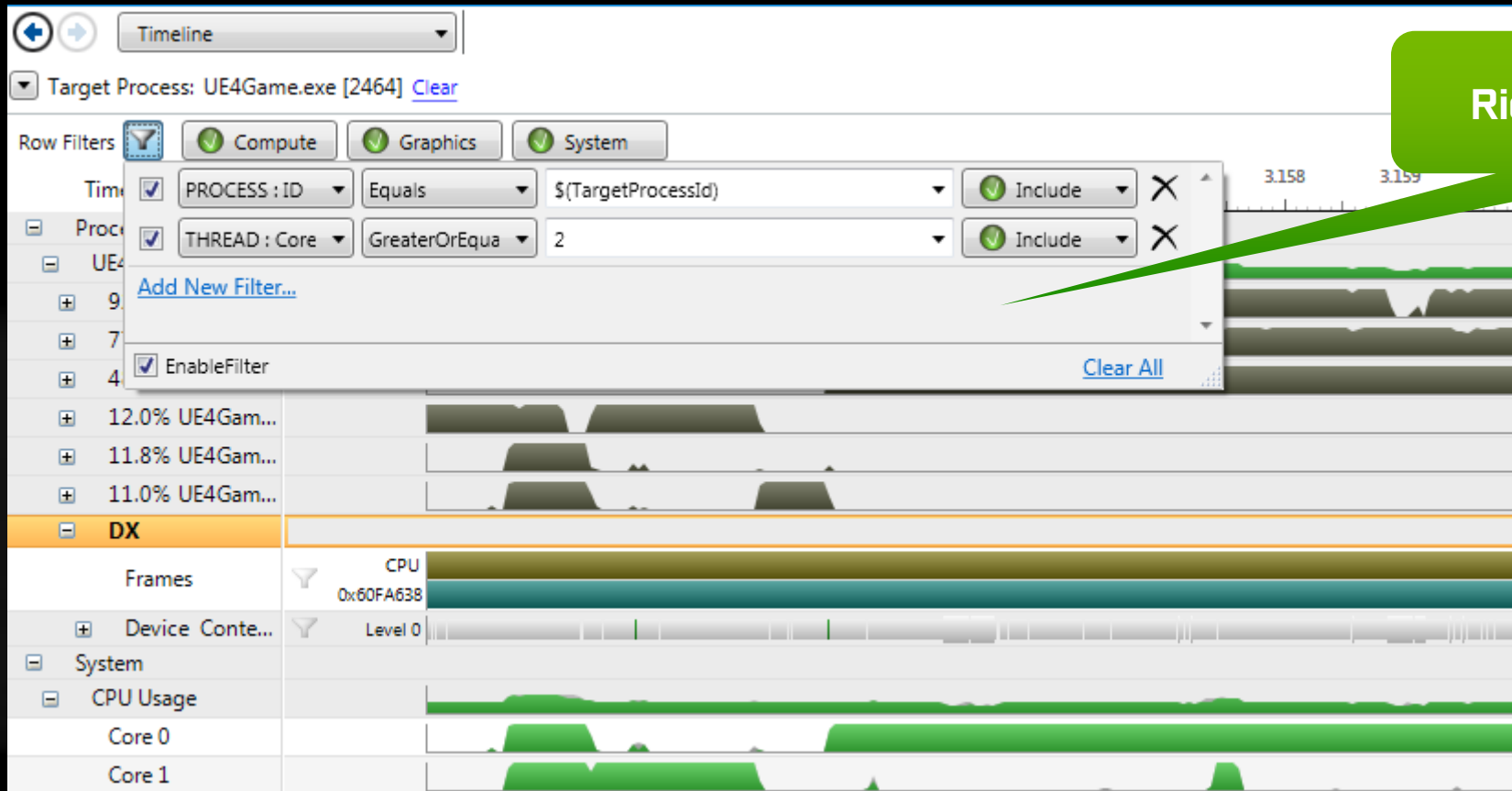
GPU 0 GeForce GTX 980			
Utilization (%)	Min	Avg	Max
<b>23.5</b>	ms/F 14	52	67
	Cmd% 3.3	26.2	71.2
	#Items 879	4,048	5,420

Summary Screen Results

Runtime/Driver Consumes CPU Time...Next Generation APIs Should Help!



# PROFILING INFILTRATOR - CPU BOUND



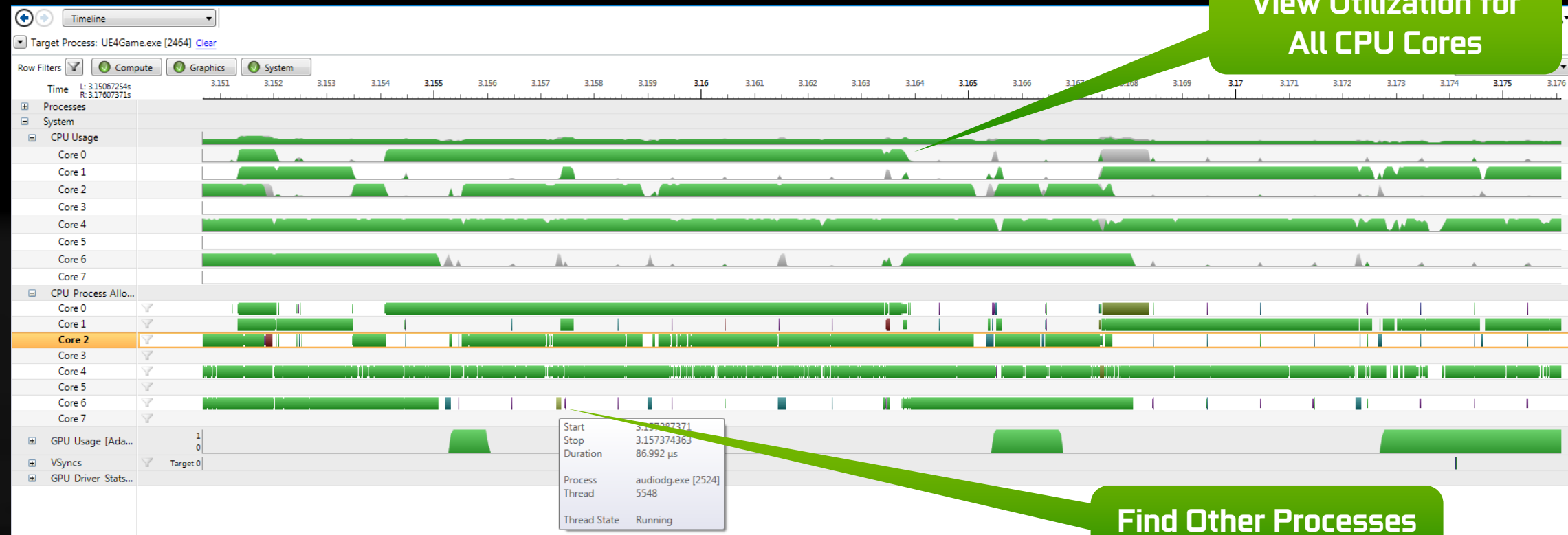
Rich Filtering Capabilities





# PROFILING INFILTRATOR - CPU BOUND

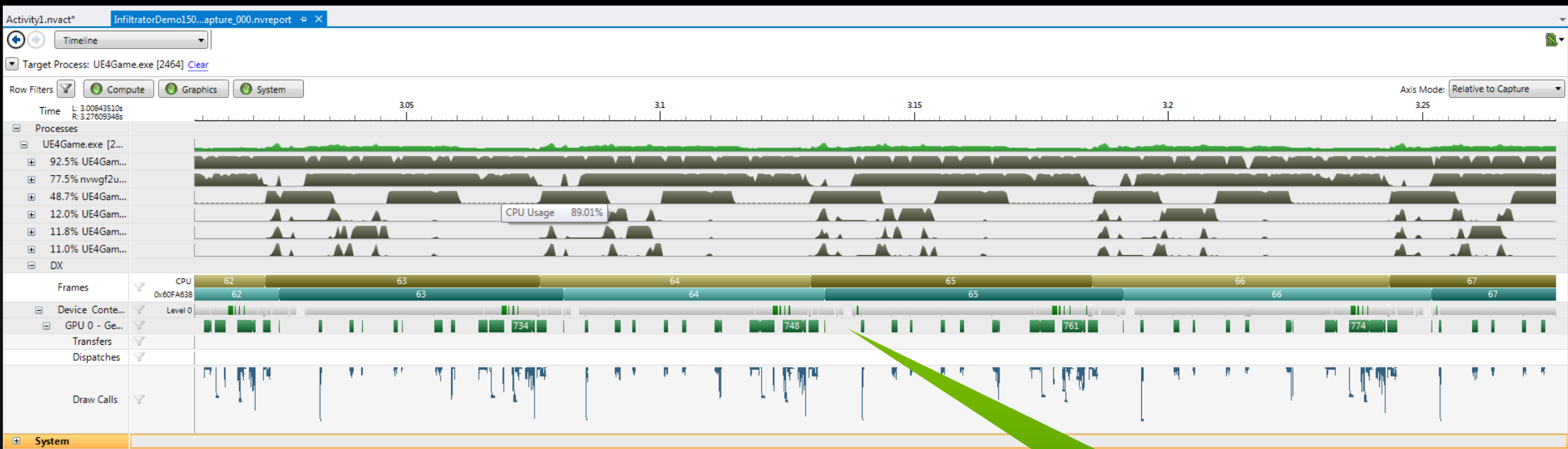
View Utilization for All CPU Cores



Find Other Processes That Steal CPU Time



# PROFILING INFILTRATOR - CPU BOUND



Gaps In GPU Workloads  
= CPU Bound



# PROFILING INFILTRATOR - GPU BOUND

UE4Game\_2015\_02\_...pture\_000.nvreport

Summary Report

## Session Overview

Summary of session information related to the captured data.

- [Session Summary](#)
- [Timeline](#)
- [Activity](#)

### UE4Game\_2015\_02\_28\_17\_09\_50.exe (Trace Application)

Captured 9.00 seconds of data on 2/28/2015 7:22:52 PM

Arguments:  
Working Dir: C:\Users\jkiel\Documents\NVIDIA Nsight\Captures\UE4Game\_2015\_02\_28\_17\_09\_50\  
Connection: localhost

## DirectX Overview

Summary of captured DirectX activity.

- [API Calls | Summary](#)
- [Draw Calls](#)
- [Frames](#)
- [Dispatches](#)
- [Performance Markers](#)
- [Transfers](#)

CPU All Cores			
API Time (%)	Min	Avg	Max
<b>86.0</b>	8	<b>8.0</b>	8
Batch	2,560	<b>2,560.0</b>	2,560

GPU 0 GeForce GTX 980			
Utilization (%)	Min	Avg	Max
<b>96.8</b>	18	<b>18</b>	23
ms/F	86.1	<b>99.1</b>	99.6
Cmd%	791	<b>794</b>	796
#Items	791	<b>794</b>	796

⚠ There are 481 CPU frames, but only 480 GPU frames.

Summary Results: Lots Of GPU Work!



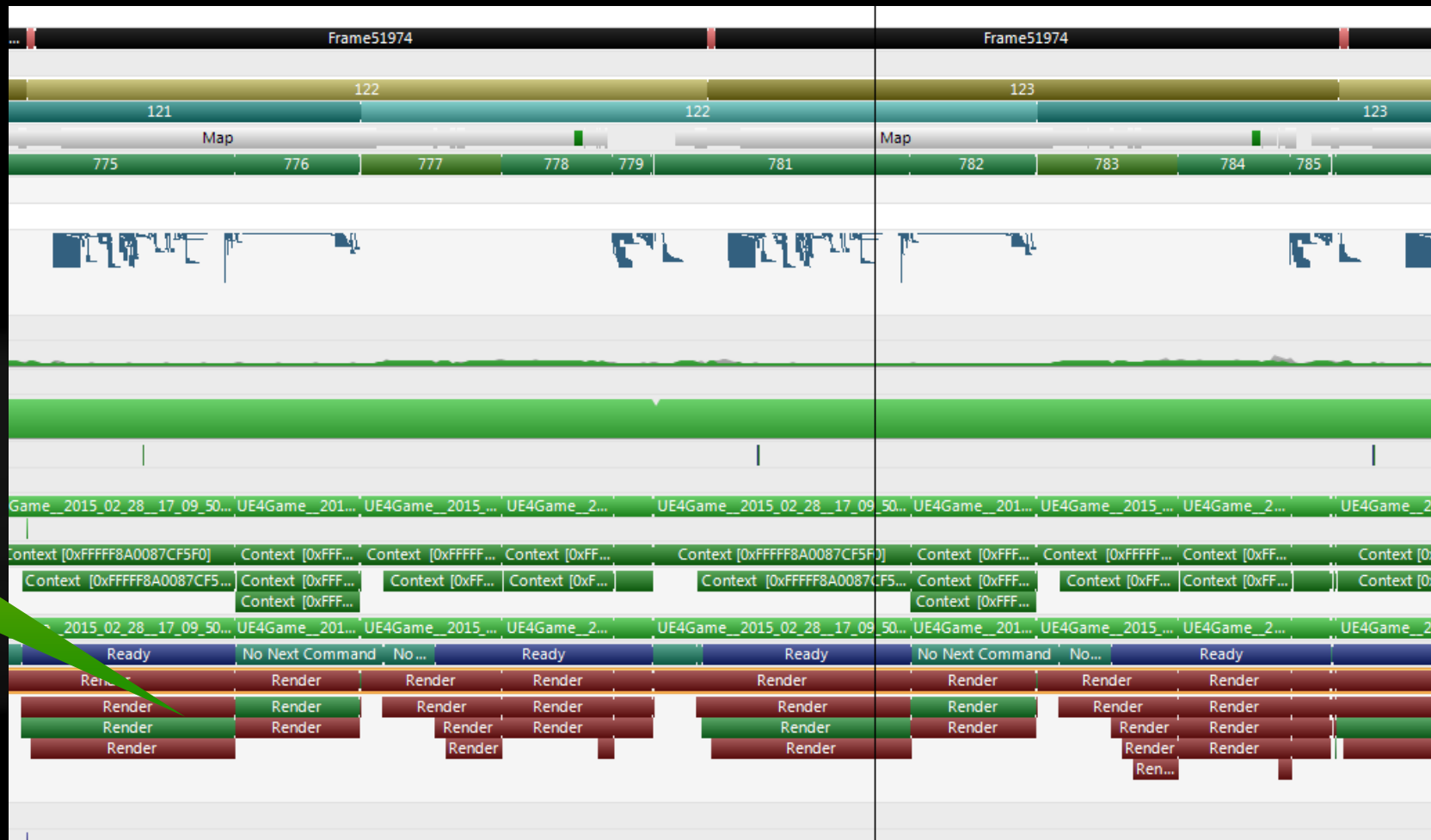
# PROFILING INFILTRATOR - GPU BOUND





# PROFILING INFILTRATOR - GPU BOUND

Correlate With  
GPUView ETW Data



# PROFILING INFILTRATOR - GPU BOUND

The screenshot displays the Visual Studio Profiling tool interface for UE4Game. The top toolbar shows 'Compute', 'Graphics', and 'System' filters. The main area is divided into several sections:

- Timeline:** Shows a performance timeline with markers for 'Map' operations. A callout 'Blocking Map Call On CPU' points to a specific 'Map' event.
- Function Calls:** A tree view showing the call stack. A callout 'Full Stack Trace' points to the 'Map [DirectX Function Call]' entry.
- Source Code:** The right pane shows the source code for 'frame0part00.cpp'. A callout 'Jump To Source' points to the line: `result = pID3D11DeviceContext_uidof_16->Map(((ID3D11Resource*)pID3D11Texture2D_uidof_1279), 0, D3D11_MAP_READ, (D3D11_NV_CHECK_RESULT(result));`
- Row Information Table:** A table at the bottom provides details for the selected function call.

Row Information	Address	Module Name	Function	File Name
Function Calls (Function Calls Row)	0x000007FEF141E2A3	Nvda.Graphics.Interception.Analysis.dll		
Mouse Information	0x000007FEF144146C	Nvda.Graphics.Interception.Analysis.dll		
Cursor Information	0x000000013F56346E	UE4Game_2015_02_28_17_09_50.exe	void __cdecl RunFrame0Part00(void)	c:\users\jkie\documents
	0x000000013F58CF95	UE4Game_2015_02_28_17_09_50.exe	WinMain	c:\users\jkie\documents
	0x000000013F660835	UE4Game_2015_02_28_17_09_50.exe	__tmainCRTStartup	fd:\dev\tools\crt\crtw32
	0x00000000774395ED	kernel32.dll		
	0x000000007756C841	ntdll.dll		

Blocking Map Call On CPU

Jump To Source

Full Stack Trace



# PROFILING INFILTRATOR - PROFILING

The screenshot displays the NVIDIA Infiltrator interface with three main components highlighted by callouts:

- Scene Scrubber:** Located at the top, it features a timeline from 0 to 7000 frames. Below the timeline are sections for 'Action', 'Dependencies', and 'Perf Markers'. A callout points to this area with the text: "Scene Scrubber With Perf Markers and Dependencies".
- API Inspector:** Located in the bottom-left, it shows a 'Shader Description' for a vertex shader. It includes a 'Shader Resource Views' section with a grid of texture slots (SRV\_10050 to SRV\_10057) and a 'Constant Buffers' table at the bottom. A callout points to this area with the text: "API Inspector - View All State".
- Events View:** Located on the right, it shows a list of GPU events with columns for 'Event', 'Description', and 'GPU μs'. A callout points to this area with the text: "Events View With Filtering".

Scene Scrubber  
With Perf Markers  
and Dependencies

Events View With  
Filtering

API Inspector -  
View All State



# PROFILING INFILTRATOR - PROFILING

The screenshot displays the Unreal Engine 4 Profiler interface. At the top, a timeline shows the execution of a game session, with a scrubber and various event tracks. Below the timeline, the 'Resources' panel is active, showing a grid of texture thumbnails. A callout bubble points to this panel with the text 'Inspect All Resources'. The 'API Inspector' window is also visible, showing details for a selected texture resource. To the right, the 'Geometry' panel displays a 3D model of a green mechanical component. A callout bubble points to this panel with the text 'Visualize Pre-transform Geometry'. The bottom of the interface shows the 'Autos Locals Memory1 Threads Modules Watch1' and 'Call Stack Breakpoints Output' tabs.

Inspect All Resources

Visualize Pre-transform Geometry





# PROFILING INFILTRATOR - PROFILING

The screenshot displays the NVIDIA Frame Profiler interface. At the top, a green callout bubble points to the 'Collect Draw Calls By Shared State' button. Below this, the 'Draw Call Groups' table shows a single group with 796 calls. The 'State Buckets' table highlights bucket 1 as the most expensive, with a CPU time of <1, GPU time of 4240, 540096 primitives, and 72479863 pixels. A second callout bubble points to a specific event in the 'Draw Calls' table, noting a 100X GPU time overdraw. The bottom section features a pipeline diagram with callouts for 'Visualize Pipeline Bottleneck' (pointing to the L2 cache) and 'This Call Is Blending Bound' (pointing to the Blending & ZBuffer stage). A third callout bubble points to the 'Summary' section, stating that the most expensive bucket and draw call is for Bokeh Filter Rendering.

Calls	CPU $\mu$ s	GPU $\mu$ s	Primitives	Pixels	Performance Markers
796	489	21194	1999418	146303956	

State Buckets	CPU $\mu$ s	GPU $\mu$ s	Primitives	Pixels
1	<1	4240	540096	72479863
117	56	2177	212595	15668991

Draw Call	CPU $\mu$ s	GPU $\mu$ s	Primitives	Pixels	
6622	756	<1	4240	540096	72479863

**Collect Draw Calls By Shared State**

**Call Has 100X Overdraw!**

**Most Expensive Bucket & Draw Call is Bokeh Filter Rendering**

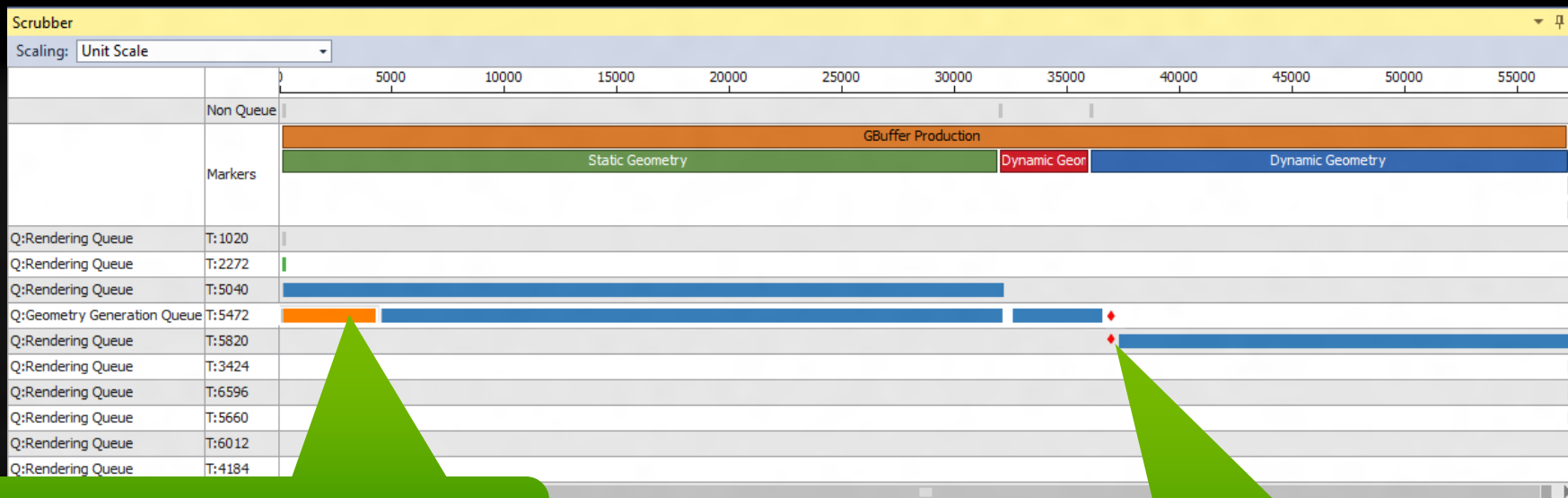
**Visualize Pipeline Bottleneck**

**This Call Is Blending Bound**



# NEW API, NEW CHALLENGES

- More parallelism!
  - Multi-thread : parallel CPU work...more parallel than D3D11
  - Multi-queue: parallel, potentially asymmetric, GPU work



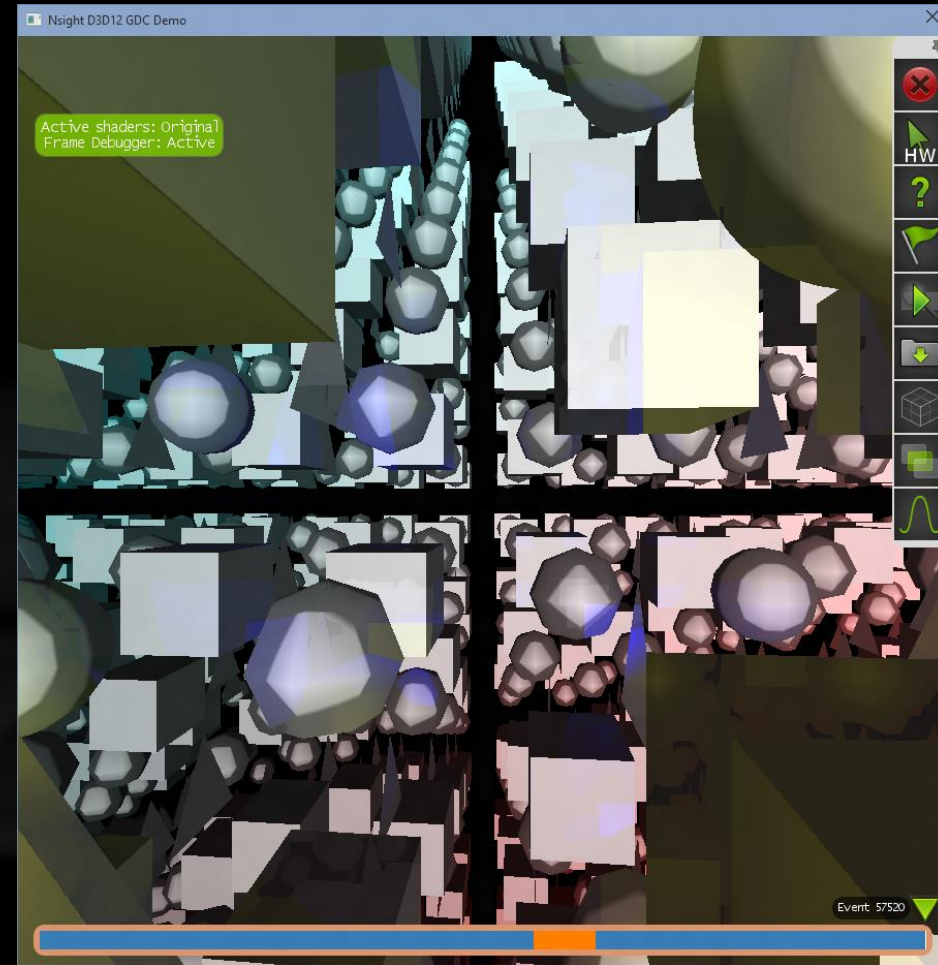
Conceptually parallel queues:  
nothing to prevent concurrency

D3D12 fence blocks the final  
batch of GPU work



# NEW API, NEW CHALLENGES

- Application is now responsible!
  - CPU/GPU synchronization
  - Usability state of resources via barriers
  - Memory heaps via placed resources
  - SRVs, UAVs, constants, samplers, all opaque at runtime
  - Root description table for binding, another layer of indirection



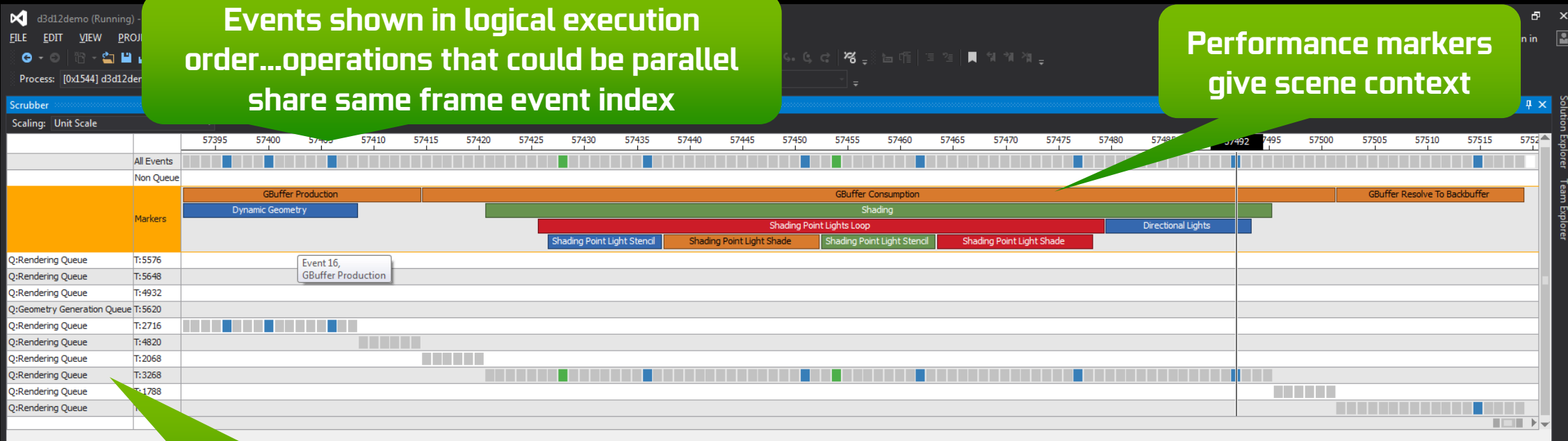
# DIRECT3D12 SNEEK PEEK (DEMO)

## Multi-queue Event Scrubber

Events shown in logical execution order...operations that could be parallel share same frame event index

Performance markers give scene context

Show all active queues





# DIRECT3D12 SNEEK PEEK (DEMO)

## Heaps View Improves API Transparency

The screenshot displays the Heaps View tool interface. On the left, a 'Heaps' list shows details for the 'Static Geometry Vertex Data Heap'. The main area features a 'Placed Resources (4)' table with columns for Resource, Dimension, Offset, Size, and Overlaps With. Below this is a 'Data' table showing memory addresses and their corresponding hex values. A 'Heap Map' at the bottom visualizes resource overlaps and usage.

Resource	Dimension	Offset	Size	Overlaps With
<a href="#">Static Geometry Whole Heap Vertex Buffer</a>	BUFFER	0	8388608	Static Geometry Triangle Pl... Static Geometry Cube Place... Static Geometry Sphere Pla...
<a href="#">Static Geometry Triangle Placed Vertex Buffer</a>	BUFFER	0	65536	Static Geometry Whole Hea... Static Geometry Cube Place... Static Geometry Sphere Pla...
<a href="#">Static Geometry Cube Placed Vertex Buffer</a>	BUFFER	65536	65536	Static Geometry Whole Hea... Static Geometry Triangle Pl... Static Geometry Sphere Pla...
<a href="#">Static Geometry Sphere Placed Vertex Buffer</a>	BUFFER	131072	65536	Static Geometry Whole Hea... Static Geometry Triangle Pl... Static Geometry Cube Place...

Address	Data
0x00000000	3f80000000000000 0000000000000000 bf80000000000000 bf800000
0x00000020	0000000000000000 bf80000000000000 bf800000bf800000 00000000
0x00000040	bf80000000000000 0000000000000000 0000000000000000 00000000
0x00000060	0000000000000000 0000000000000000 0000000000000000 00000000
0x00000080	0000000000000000 0000000000000000 0000000000000000 00000000
0x000000a0	0000000000000000 0000000000000000 0000000000000000 00000000
0x000000c0	0000000000000000 0000000000000000 0000000000000000 00000000

Resource relevant visualizer

All resources in the heap

All available heaps

Visualize resource overlaps and heap usage by type








# DIRECT3D12 SNEEK PEEK (DEMO)

## Descriptor Heap View Improves API Transparency

Descriptor Heaps

Set	Name	Properties	Revisions
✓		Type: D3D12_CBV_SRV_UAV_DESCRIPTOR_HEAP NumDescriptors: 10000 Flags: D3D12_DESCRIPTOR_HEAP_SHADER_VISIBLE	1
-		Type: D3D12_RTV_DESCRIPTOR_HEAP NumDescriptors: 10 Flags: D3D12_DESCRIPTOR_HEAP_NONE	1
-		Type: D3D12_DSV_DESCRIPTOR_HEAP NumDescriptors: 10 Flags: D3D12_DESCRIPTOR_HEAP_NONE	1
✓		Type: D3D12_SAMPLER_DESCRIPTOR_HEAP NumDescriptors: 10 Flags: D3D12_DESCRIPTOR_HEAP_SHADER_VISIBLE	1

Revision 1 of 1

Heap Entry	ViewDesc	Resource	Resident
0	SRV Descriptor Format: DXGI_FORMAT_B8G8R8A8_UNORM ViewDimension: D3D12_SRV_DIMENSION_TEXTURE2D Shader4ComponentMapping: (COMPONENT_0, COMPONENT_1, COMPONENT_2, COMPONENT_3) MostDetailedMip: 0 MipLevels: 1 ResourceMinLODClamp: 0.00	 GBuffer Resu ltTexture	true
1	SRV Descriptor Format: DXGI_FORMAT_R32G32B32A32_FLOAT ViewDimension: D3D12_SRV_DIMENSION_TEXTURE2D Shader4ComponentMapping: (COMPONENT_0, COMPONENT_1, COMPONENT_2, COMPONENT_3) MostDetailedMip: 0 MipLevels: 1 ResourceMinLODClamp: 0.00	 GBuffer Posit ionTexture	true
2	SRV Descriptor Format: DXGI_FORMAT_R32G32B32A32_FLOAT ViewDimension: D3D12_SRV_DIMENSION_TEXTURE2D Shader4ComponentMapping: (COMPONENT_0, COMPONENT_1, COMPONENT_2, COMPONENT_3) MostDetailedMip: 0 MipLevels: 1 ResourceMinLODClamp: 0.00	 GBuffer Nor malTexture	true

All descriptor  
heaps

Heap specific  
details (SRVs)



# DIRECT3D12 SNEEK PEEK (DEMO)

## Descriptor Heap View Improves API Transparency

Set	Name	Properties	Revisions
✓		Type: D3D12_CBV_SRV_UAV_DESCRIPTOR_HEAP NumDescriptors: 10000 Flags: D3D12_DESCRIPTOR_HEAP_SHADER_VISIBLE	1
-		Type: D3D12_RTV_DESCRIPTOR_HEAP NumDescriptors: 10 Flags: D3D12_DESCRIPTOR_HEAP_NONE	1
-		Type: D3D12_DSV_DESCRIPTOR_HEAP NumDescriptors: 10 Flags: D3D12_DESCRIPTOR_HEAP_NONE	1
✓		Type: D3D12_SAMPLER_DESCRIPTOR_HEAP NumDescriptors: 10 Flags: D3D12_DESCRIPTOR_HEAP_SHADER_VISIBLE	1

Heap Entry	ViewDesc
0	Filter: D3D12_FILTER_MIN_MAG_MIP_POINT AddressU: D3D12_TEXTURE_ADDRESS_CLAMP AddressV: D3D12_TEXTURE_ADDRESS_CLAMP AddressW: D3D12_TEXTURE_ADDRESS_CLAMP MipLODBias: 0.00 MaxAnisotropy: 1 ComparisonFunc: D3D12_COMPARISON_NEVER BorderColor: (0,0,0,0) MinLOD: 0.00 MaxLOD: 1.00
1	Filter: D3D12_FILTER_MIN_MAG_MIP_POINT AddressU: D3D12_TEXTURE_ADDRESS_CLAMP AddressV: D3D12_TEXTURE_ADDRESS_CLAMP AddressW: D3D12_TEXTURE_ADDRESS_CLAMP MipLODBias: 0.00 MaxAnisotropy: 1 ComparisonFunc: D3D12_COMPARISON_NEVER BorderColor: (0,0,0,0) MinLOD: 0.00 MaxLOD: 1.00
[2 - 9]	Unused Descriptor

All descriptor  
heaps

Heap specific  
details (Samplers)



# DIRECT3D12 SNEEK PEEK (DEMO)

## Root Parameters View Improves API Transparency

The screenshot shows two panels from a graphics debugger. The left panel, titled 'Root Parameters', displays a table with columns for Index, Type, Info, and Visibility. The right panel, titled 'Root Arguments', displays a table with columns for Heap Entry, ViewDesc, Resource, and Resident. A green callout points to the 'Info' column in the Root Parameters table, and another green callout points to the 'Resource' column in the Root Arguments table.

Index	Type	Info	Visibility
0	Descriptor Table Range 0	RangeType: D3D12_DESCRIPTOR_RANGE_SRV NumDescriptors: 4 BaseShaderRegister: 0 RegisterSpace: 0	D3D12_SHADER_VISIBILITY_PIXEL
1	Descriptor Table Range 0	RangeType: D3D12_DESCRIPTOR_RANGE_SAMPLER NumDescriptors: 1 BaseShaderRegister: 0 RegisterSpace: 0	D3D12_SHADER_VISIBILITY_PIXEL
2	Descriptor Table Range 0	RangeType: D3D12_DESCRIPTOR_RANGE_CBV NumDescriptors: 1 BaseShaderRegister: 0 RegisterSpace: 0	D3D12_SHADER_VISIBILITY_ALL
3	Constants	ShaderRegister: 1 RegisterSpace: 0 Num 2BitValues: 4	D3D12_SHADER_VISIBILITY_ALL

Heap Entry	ViewDesc	Resource	Resident
1	Format ViewDimension Shader4Component MostDetail MipLevels ResourceName		true
2	Format ViewDimension Shader4Component MostDetail MipLevels ResourceName		true
3	Format ViewDimension Shader4Component MostDetail MipLevels ResourceName		true
4	Format ViewDimension Shader4Component MostDetail MipLevels ResourceName		true

Root signature layout and population

Type specific details for parameter entry



# NVIDIA GAMEWORKS™

## Links...

Web: <http://ameworks.nvidia.com>

Latest info and download options

Survey: <https://developer.nvidia.com/developer-tools-survey-201503>

YouTube: <https://www.youtube.com/user/nvidiaGameWorks>

Tools, effects, and game integration videos

Twitter: <https://twitter.com/nvidiadeveloper>

Catch up on up to the minute happenings

