



NVIDIA® NSIGHT™ VSE 4.7™帮助 开发更快更炫的图形效果

阎安 软件工程师



开发者工具: NVIDIA GameWorks™

目录

NVIDIA® Nsight™ Visual Studio Edition 4.7™介绍

分析 Infiltrator (Epic Games)



在现代GPU上进行开发可能会
让您觉得云深雾绕



Shader Bug

GPU瓶颈

API状态管理

性能

API的使用

驱动

CPU和GPU交互

内存

GPU的帧时间

资源

Shader的修正

有效的
Fragments



NVIDIA开发者工具

构建 调试 性能剖析

Microsoft
DirectX

OpenGL

nVIDIA
CUDA

OpenGL|ES



GNU
C/C++

集成开发环境

Visual Studio

eclipse

独立工具



硬件支持

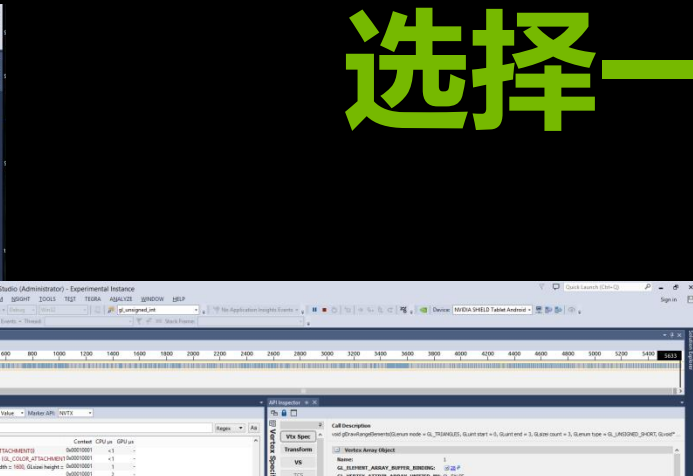
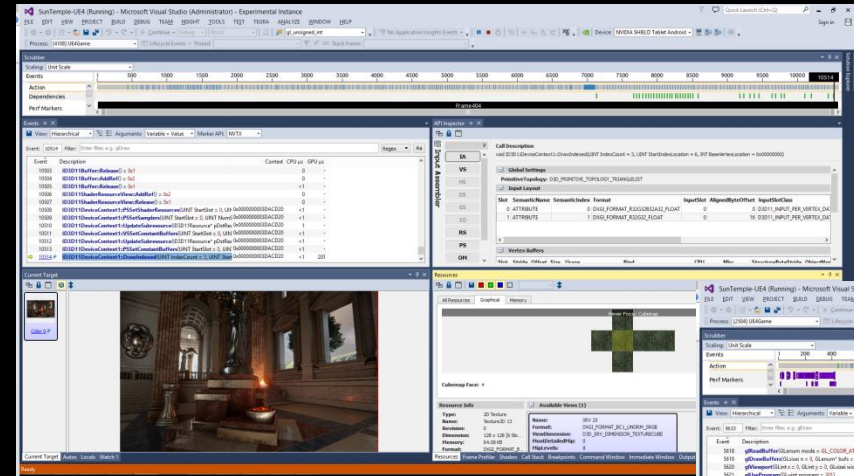
CPU和GPU的调试和性能剖析



nVIDIA
GAMMEWORKS™



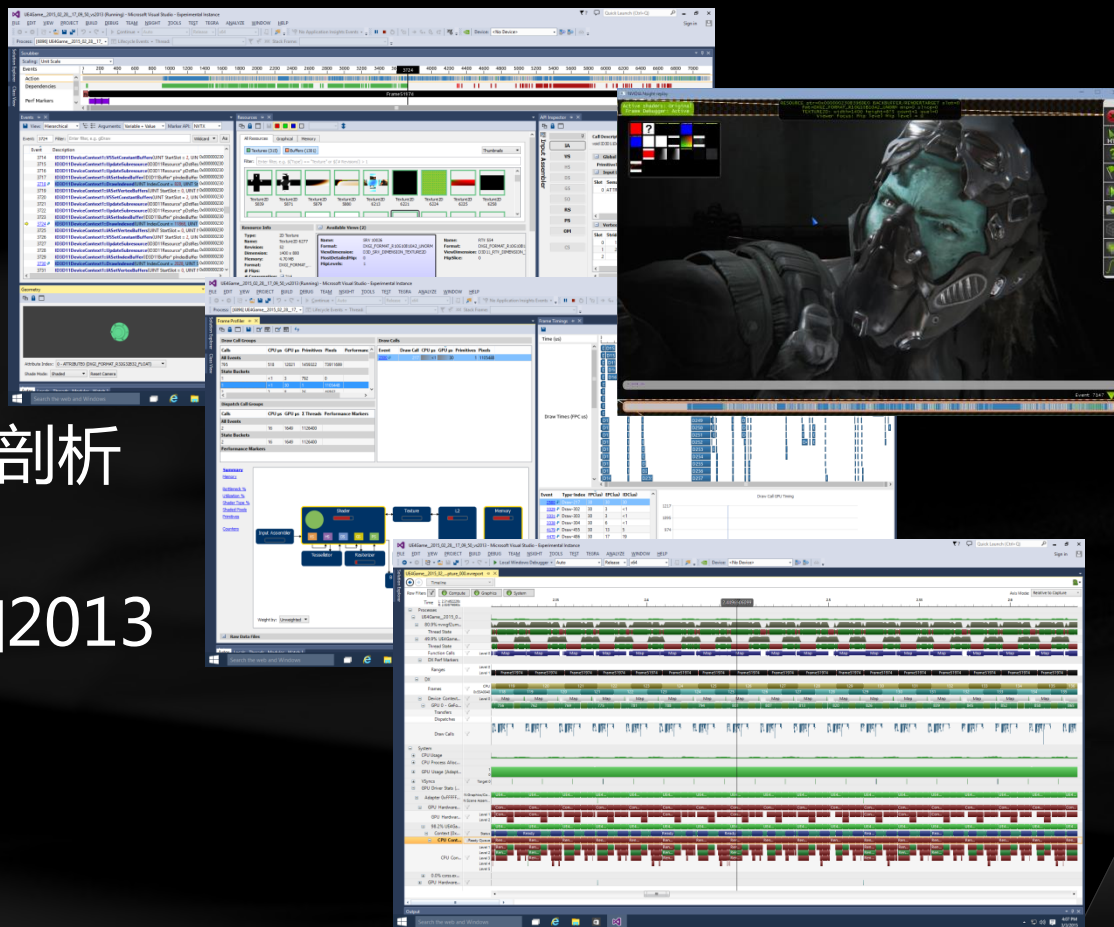
选择一个平台



NVIDIA® NSIGHT™ VISUAL STUDIO EDITION

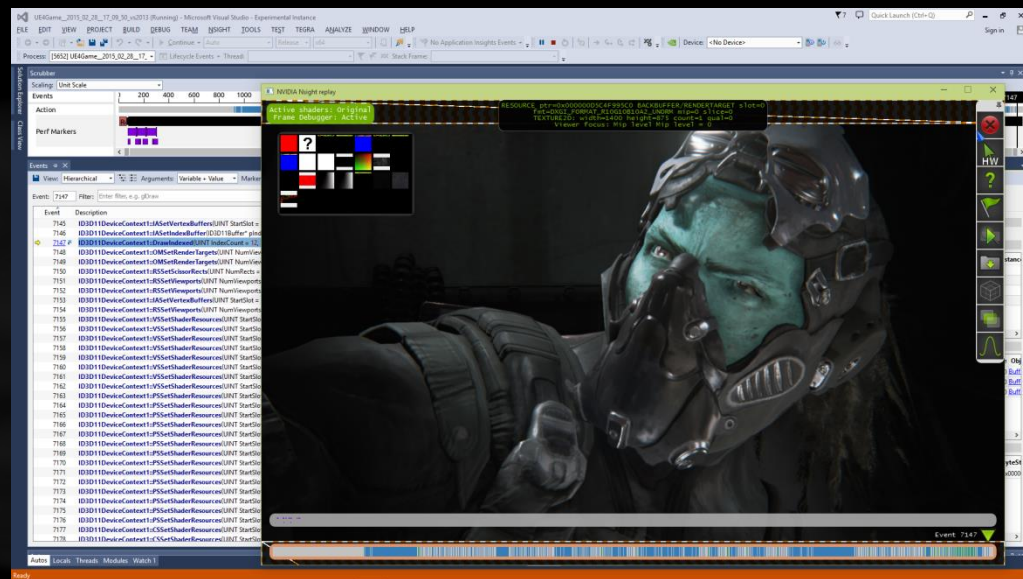
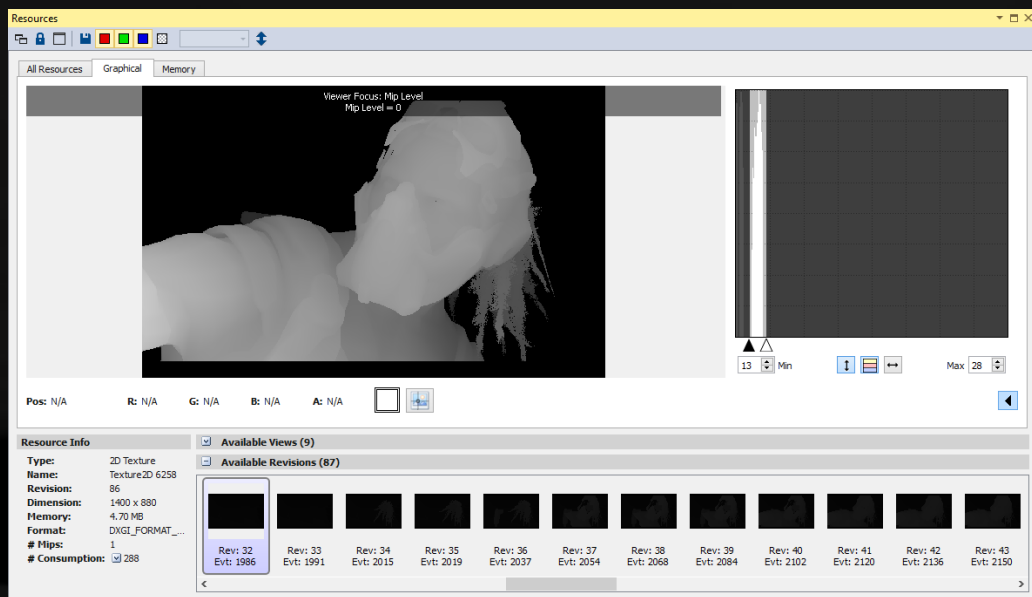
加速视觉计算的开发

- 支持DirectX 9/11和OpenGL
- 图形开发的调试和性能剖析
- 调试HLSL和GLSL着色器代码
- CUDA kernels开发的调试和性能剖析
- 系统级的性能剖析
- 支持Visual Studio 2010, 2012和2013



4.7的新功能

- Maxwell架构支持 (不含Shader Debugging功能)
- OpenGL 4.4支持 (不含ARB_sparse_texture)
- 截取OpenGL调用并生成对应源代码及工程
- 贴图的直方图显示和颜色范围重映射
- CUDA 7.5支持



INFILTRATOR的性能剖析

Active shaders: Original
Frame Debugger: Active

```
RESOURCE ptr=0x00000000D0B1BD0 BACKBUFFER/RENDERTARGET slot=0  
fmt=DXGI_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

The screenshot displays the NVIDIA Frame Debugger interface. At the top, a status bar shows 'Active shaders: Original' and 'Frame Debugger: Active'. A central window displays a game scene with a character in a dark, industrial environment. A right-side toolbar contains various icons for navigation and analysis. At the bottom, a timeline shows the current event at 37932.



INFILTRATOR的性能剖析

观看Infiltrator演示: <http://www.geforce.cn/whats-new/articles/infiltrator-gtx-680-powered-unreal-engine-4-tech-demo-unveiled>



INFILTRATOR的性能剖析 – CPU BOUND

The screenshot shows the Infiltrator application settings in Visual Studio. The 'Application Settings' section is expanded, showing the application path and working directory. The 'Activity Type' section is set to 'Trace Process Tree'. The 'Trace Settings' section is expanded, showing 'DirectX' selected under 'Domains'. The 'API Categories' section is expanded, showing 'All Categories' selected. The 'Workload Trace' section is expanded, showing 'CPU Frames', 'GPU Frames', and 'Push Buffers' selected. The 'Connection Status' section shows a green circle indicating the connection is established. The 'Session Control' section shows a red circle and 'Start' button. The 'Capture Control' section shows a red circle and 'Start', 'Stop', and 'Cancel' buttons. The 'Open Report on Stop' checkbox is checked, and the 'Summary Report' dropdown is visible.

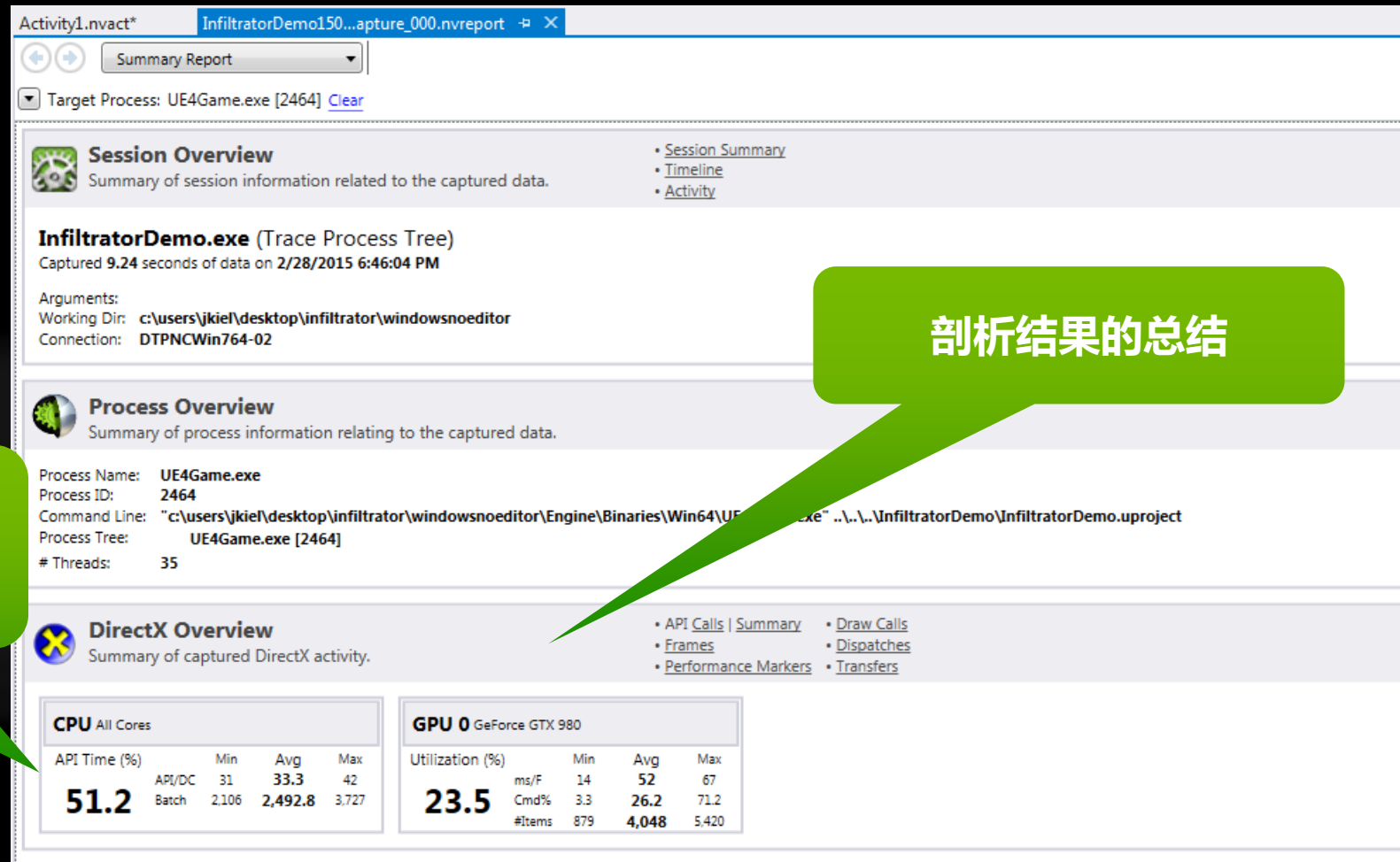
程序启动设置

性能剖析项: System和DirectX

启动!



INFILTRATOR的性能剖析 – CPU BOUND

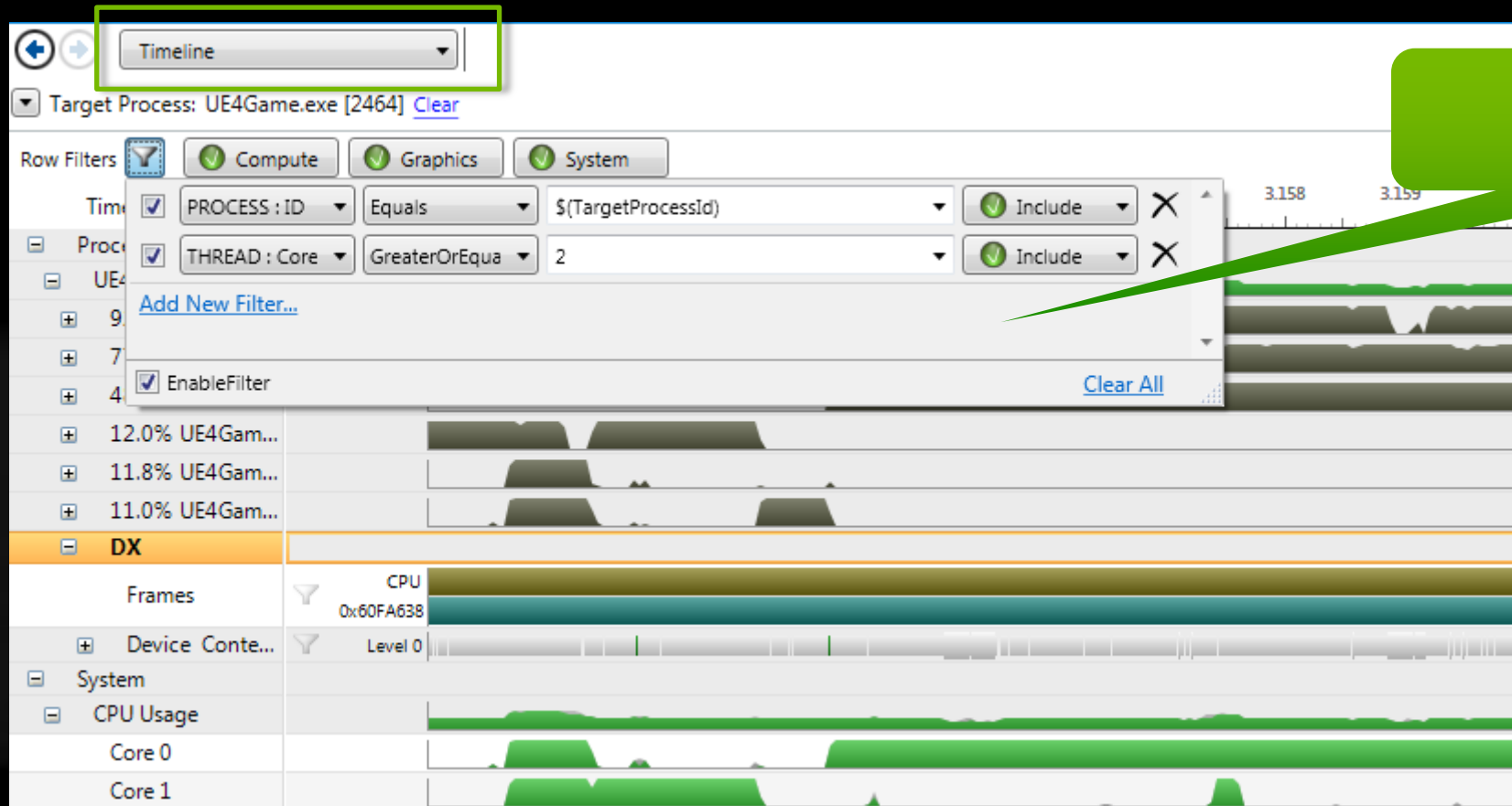


剖析结果的总结

运行时/驱动会消耗CPU时间...下一代API会帮助改进CPU的占用!



INFILTRATOR的性能剖析 – CPU BOUND

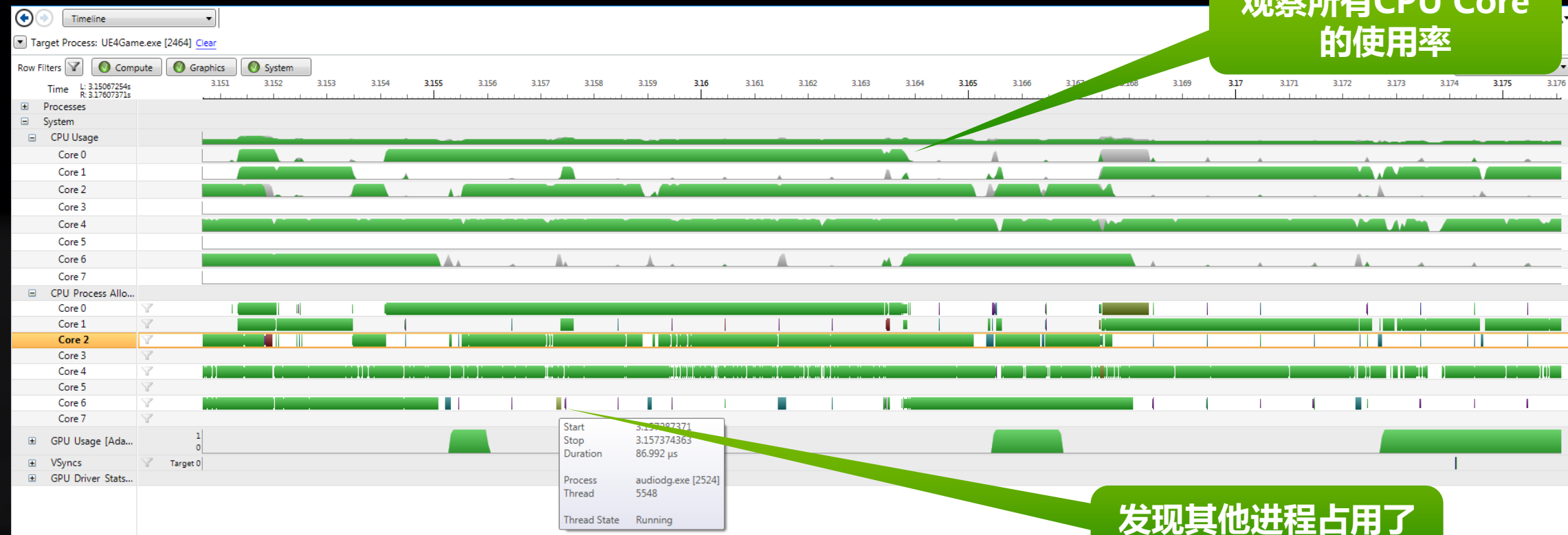


丰富的过滤选项



INFILTRATOR的性能剖析 – CPU BOUND

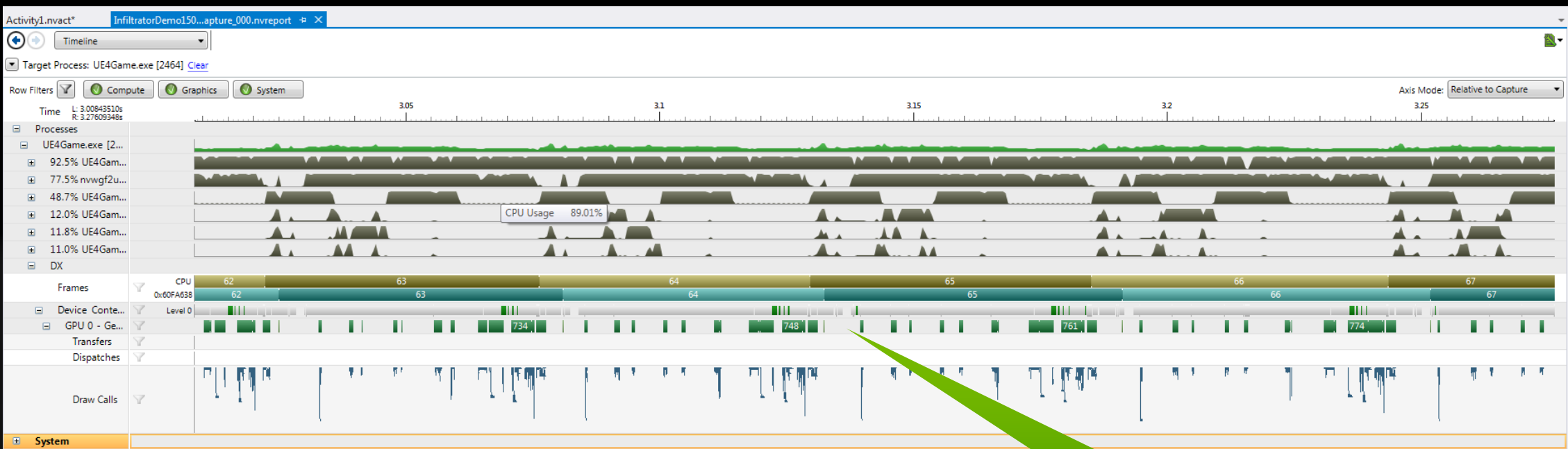
观察所有CPU Core的使用率



发现其他进程占用了CPU时间



INFILTRATOR的性能剖析 – CPU BOUND



GPU负载之间的间隙 =
CPU Bound



INFILTRATOR的性能剖析 – 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
86.0	8	8.0	8
Batch	2,560	2,560.0	2,560

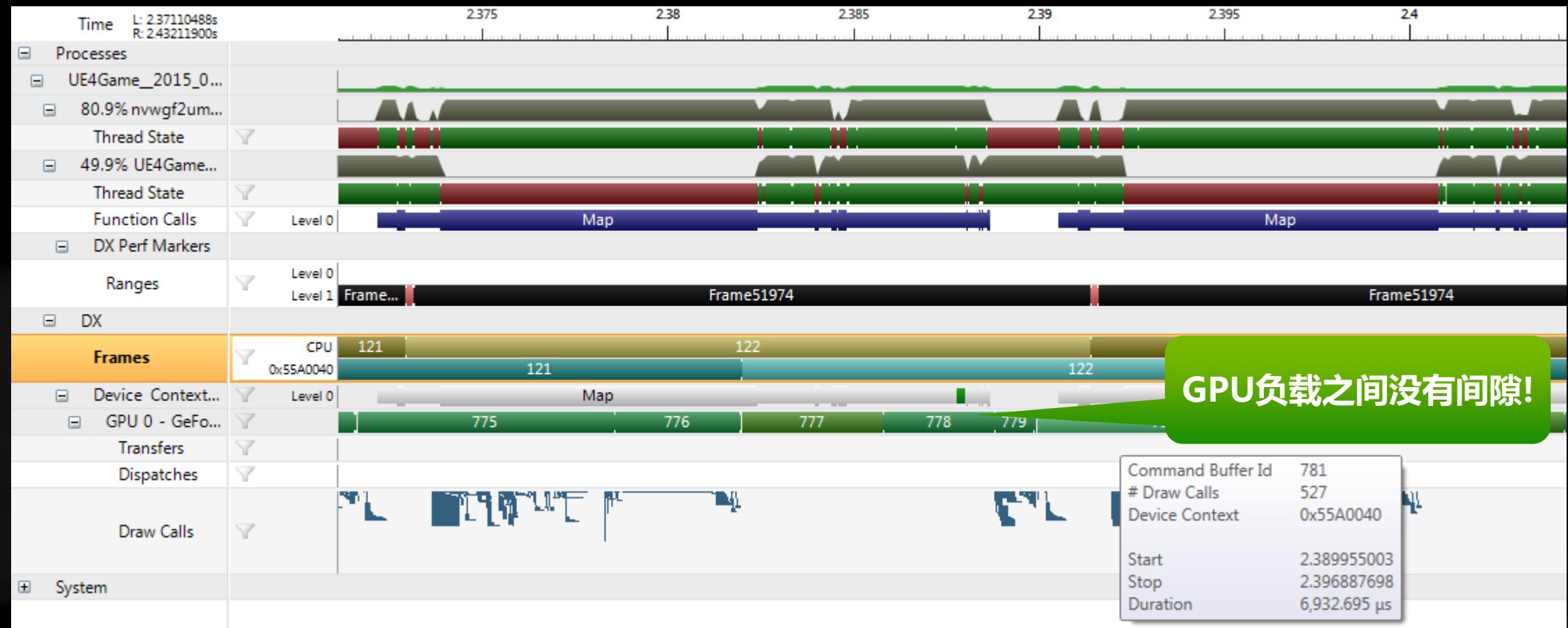
GPU 0 GeForce GTX 980			
Utilization (%)	Min	Avg	Max
96.8	18	18	23
ms/F	86.1	99.1	99.6
Cmd%	791	794	796
#Items	791	794	796

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

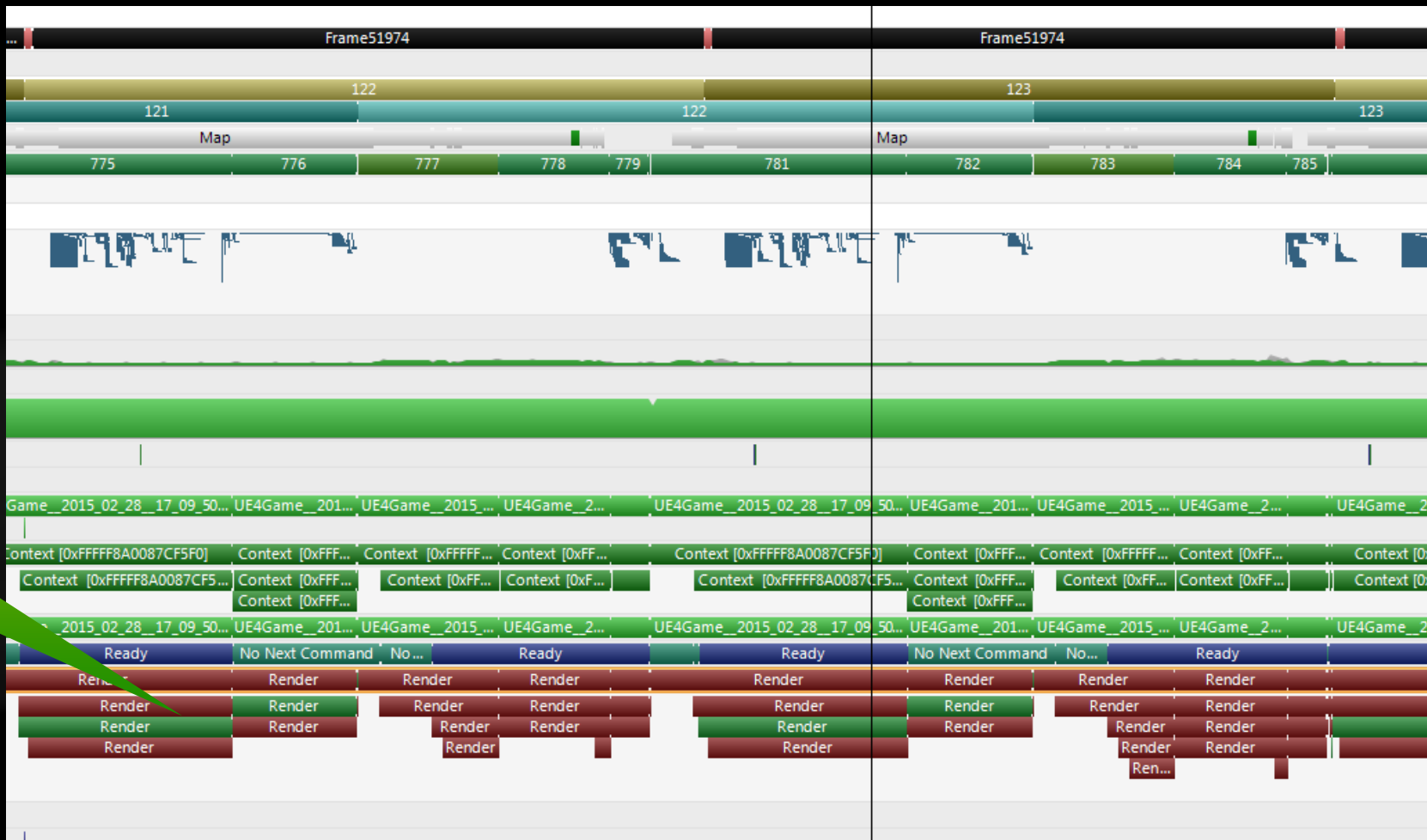
结果总结: 更多的GPU
负载



INFILTRATOR的性能剖析 – GPU BOUND



INFILTRATOR的性能剖析 – GPU BOUND



和GPUView的ETW数据相关联



INFILTRATOR的性能剖析 – GPU BOUND

CPU阻塞在Map
函数调用

跳转到对应源码

完整的Call Stack

The screenshot shows the Visual Studio IDE with the following components:

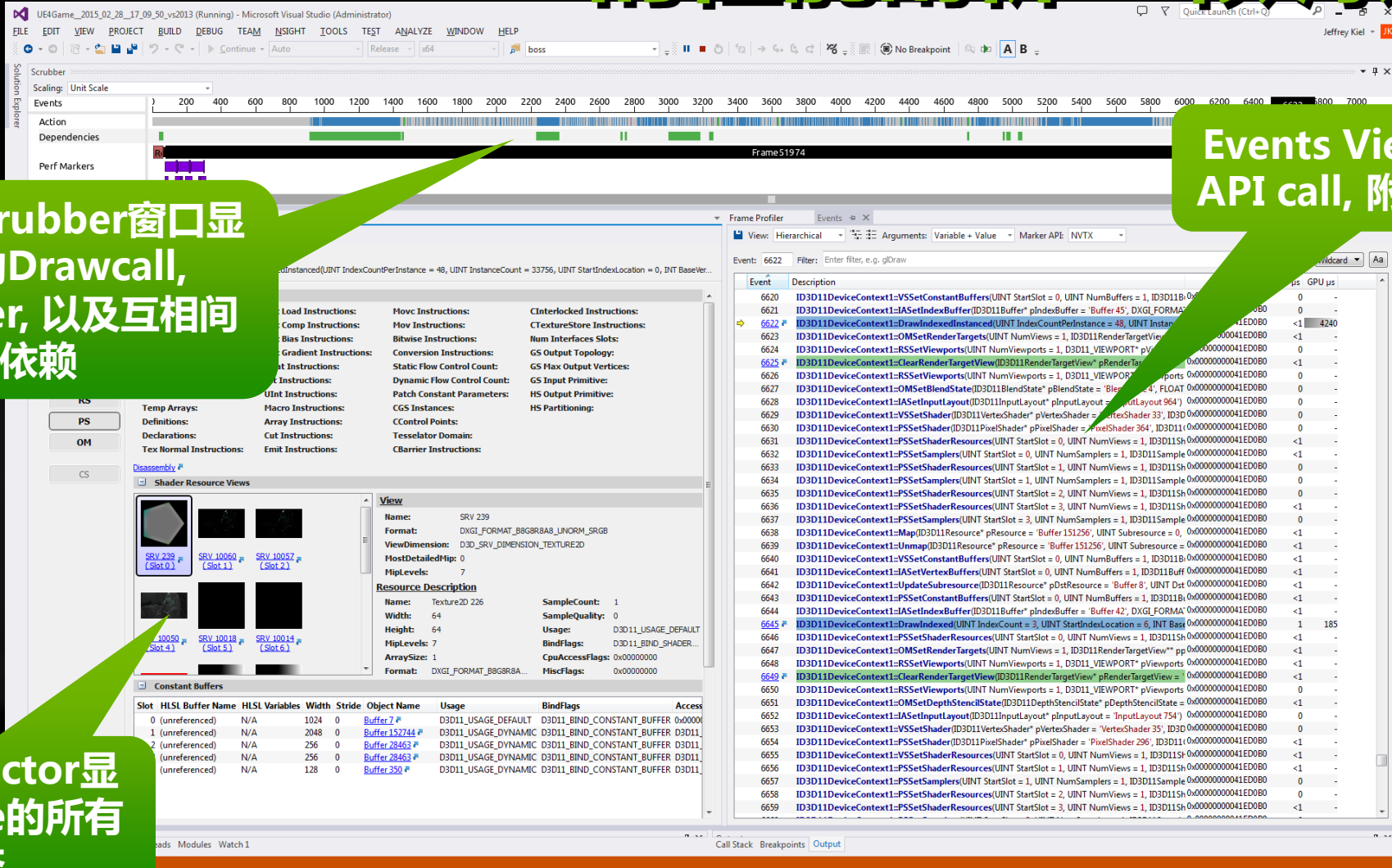
- Timeline:** Displays CPU and GPU activity over time. A green arrow points from the 'Map' function call in the call stack to the corresponding activity in the timeline.
- Call Stack:** Shows the stack of function calls. The top entry is 'Map [DirectX Function Call]' at address 0x55A0040. Below it is '3624 [Thread]' and 'Stack 1'. A green arrow points from the 'Map' entry to the source code window.
- Source Code:** Shows the source code for 'frame00.cpp'. A green arrow points from the call stack to the line: `result = pID3D11DeviceContext_uifidof_16->Map(((ID3D11Resource*)pID3D11Texture2D_uifidof_1279), 0, D3D11_MAP_READ, (D3D11_NV_CHECK_RESULT(result));`

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	0x000000013F563466	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		
	0x000000007766C841	ntdll.dll		

```
3440 NV_CHECK_RESULT(result);
3441 memcpy(D3D11_MAPPED_SUBRESOURCE_temp_259.pData, NV_GET_RESOURCE(void*, 258), 96); // Applying update to mapped data
3442 pID3D11DeviceContext_uifidof_16->Unmap(pID3D11Resource_uifidof_254680, 0);
3443
3444 // Map 259 of 464
3445 D3D11_MAPPED_SUBRESOURCE D3D11_MAPPED_SUBRESOURCE_temp_260;
3446 result = pID3D11DeviceContext_uifidof_16->Map(pID3D11Resource_uifidof_254682, 0,
3447 NV_CHECK_RESULT(result);
3448 memcpy(D3D11_MAPPED_SUBRESOURCE_temp_260.pData, NV_GET_RESOURCE(void*, 259),
3449 pID3D11DeviceContext_uifidof_16->Unmap(pID3D11Resource_uifidof_254682, 0);
3450
3451 // Map 260 of 464
3452 D3D11_MAPPED_SUBRESOURCE D3D11_MAPPED_SUBRESOURCE_temp_261;
3453 result = pID3D11DeviceContext_uifidof_16->Map(pID3D11Resource_uifidof_254684, 0, D3D11_MAP_WRITE_DISABLE,
3454 NV_CHECK_RESULT(result);
3455 memcpy(D3D11_MAPPED_SUBRESOURCE_temp_261.pData, NV_GET_RESOURCE(void*, 260), 48); // Applying update to mapped data
3456 pID3D11DeviceContext_uifidof_16->Unmap(pID3D11Resource_uifidof_254684, 0);
3457
3458 // Map 261 of 464
3459 D3D11_MAPPED_SUBRESOURCE D3D11_MAPPED_SUBRESOURCE_temp_262;
3460 result = pID3D11DeviceContext_uifidof_16->Map(pID3D11Resource_uifidof_254686, 0, D3D11_MAP_WRITE_DISABLE,
3461 NV_CHECK_RESULT(result);
3462 memcpy(D3D11_MAPPED_SUBRESOURCE_temp_262.pData, NV_GET_RESOURCE(void*, 261), 48); // Applying update to mapped data
3463 pID3D11DeviceContext_uifidof_16->Unmap(pID3D11Resource_uifidof_254686, 0);
3464 pID3D11DeviceContext_uifidof_16->OMSetRenderTargets(1, &pID3D11RenderTargetView_uifidof_1219, NULL);
3465
3466 static D3D11_VIEWPORT D3D11_VIEWPORT_temp_1 = {0.0f, 0.0f, HexToF16(0x44AF0000/*1400.0f*/), HexToF16(0x445AC000/*1400.0f*/),
3467 pID3D11DeviceContext_uifidof_16->RSSetViewports(1, &D3D11_VIEWPORT_temp_1);
3468
3469 // Map 262 of 464
3470 D3D11_MAPPED_SUBRESOURCE D3D11_MAPPED_SUBRESOURCE_temp_263;
3471 result = pID3D11DeviceContext_uifidof_16->Map(((ID3D11Resource*)pID3D11Texture2D_uifidof_1279), 0, D3D11_MAP_READ, (D3D11_NV_CHECK_RESULT(result);
3472 NV_CHECK_RESULT(result);
3473
3474 pID3D11DeviceContext_uifidof_16->Unmap(((ID3D11Resource*)pID3D11Texture2D_uifidof_1279), 0);
3475
3476 // Map 263 of 464
3477 D3D11_MAPPED_SUBRESOURCE D3D11_MAPPED_SUBRESOURCE_temp_264;
3478 result = pID3D11DeviceContext_uifidof_16->Map(((ID3D11Resource*)pID3D11Buffer_uifidof_62728), 0, D3D11_MAP_WRITE_DISABLE,
3479 NV_CHECK_RESULT(result);
3480 memcpy(D3D11_MAPPED_SUBRESOURCE_temp_264.pData, NV_GET_RESOURCE(void*, 262), 65536); // Applying update to mapped data
3481
3482 // Map 264 of 464
3483 D3D11_MAPPED_SUBRESOURCE D3D11_MAPPED_SUBRESOURCE_temp_265;
3484 result = pID3D11DeviceContext_uifidof_16->Map(((ID3D11Resource*)pID3D11Buffer_uifidof_248920), 0, D3D11_MAP_WRITE_DISABLE,
3485 NV_CHECK_RESULT(result);
3486 memcpy(D3D11_MAPPED_SUBRESOURCE_temp_265.pData, NV_GET_RESOURCE(void*, 263), 256); // Applying update to mapped data
3487 pID3D11DeviceContext_uifidof_16->Unmap(((ID3D11Resource*)pID3D11Buffer_uifidof_248920), 0);
3488
3489 // Map 265 of 464
3490 D3D11_MAPPED_SUBRESOURCE D3D11_MAPPED_SUBRESOURCE_temp_266;
3491 result = pID3D11DeviceContext_uifidof_16->Map(((ID3D11Resource*)pID3D11Buffer_uifidof_252961), 0, D3D11_MAP_WRITE_DISABLE,
3492 NV_CHECK_RESULT(result);
3493 memcpy(D3D11_MAPPED_SUBRESOURCE_temp_266.pData, NV_GET_RESOURCE(void*, 264), 64); // Applying update to mapped data
3494 pID3D11DeviceContext_uifidof_16->Unmap(((ID3D11Resource*)pID3D11Buffer_uifidof_252961), 0);
3495
3496 // Map 266 of 464
3497 D3D11_MAPPED_SUBRESOURCE D3D11_MAPPED_SUBRESOURCE_temp_267;
3498 result = pID3D11DeviceContext_uifidof_16->Map(((ID3D11Resource*)pID3D11Buffer_uifidof_255270), 0, D3D11_MAP_WRITE_DISABLE,
3499 NV_CHECK_RESULT(result);
3500 memcpy(D3D11_MAPPED_SUBRESOURCE_temp_267.pData, NV_GET_RESOURCE(void*, 265), 256); // Applying update to mapped data
3501 pID3D11DeviceContext_uifidof_16->Unmap(((ID3D11Resource*)pID3D11Buffer_uifidof_255270), 0);
3502
```



INFILTRATOR的性能剖析 - 帧调试



当前帧的Scrubber窗口显示所有的Drawcall, PerfMarker, 以及互相间的依赖

Events View显示所有API call, 附带过滤功能

API Inspector显示Pipeline的所有状态



INFILTRATOR的性能剖析 - 帧调试

The screenshot displays the Infiltrator interface with several key components:

- Timeline:** Shows a scrubber at the top with a timeline from 0 to 7000. The current frame is 4013.
- Resources:** A grid of texture thumbnails is shown, with a filter set to "Textures (315)".
- Resource Info:** A detailed view of a selected texture resource (Texture2D 6258) with the following properties:

Type:	2D Texture
Name:	Texture2D 6258
Revision:	64
Dimension:	1400 x 880
Memory:	4.70 MB
Format:	D3D11_DSV_DIMENSION_TEXTURE2D
# Mips:	1
# Consumption:	288
- Geometry:** A 3D view of a green mechanical part, which is highlighted by a callout bubble.

显示当前模型(变形前)

查看所有资源的信息



INFILTRATOR的性能剖析 – 帧调试

根据渲染的状态分类Drawcall

Calls	CPU μ s	GPU μ s	Primitives	Pixels	Performance Markers
796	489	21194	1999418	146303956	

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

Dispatch Call Groups	CPU μ s	GPU μ s	Σ Threads	Performance Markers
All Events				
2	16	3046	1126400	

State Buckets	CPU μ s	GPU μ s	Σ Threads	Performance Markers
1		833	563200	

这个Drawcall重复渲染了100倍的像素个数

显示当前Pipeline的瓶颈

当前Drawcall的瓶颈在Blend上

更加耗时的是Bokeh Filter Rendering对应的Drawcall

Weight by: GPU Time

Raw Data Files



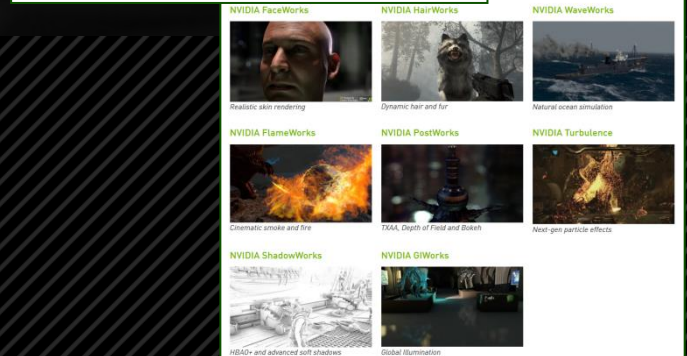
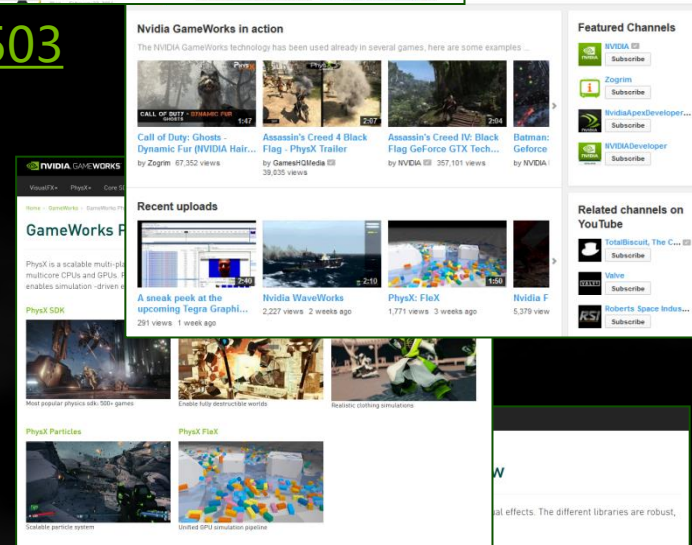
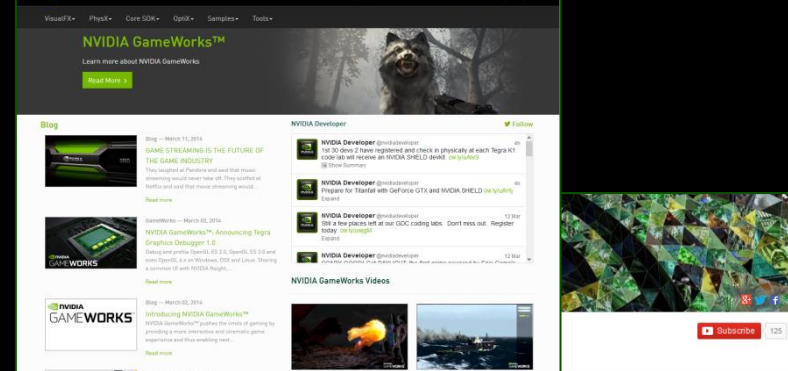
NVIDIA GAMEWORKS™

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

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

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

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



Q&A

- E-mail: devtools-support@nvidia.com
- Forum: <https://devtalk.nvidia.com/default/board/84/nsight-visual-studio-edition/>

