Raise your VR game with NVIDIA® GeForce® Tools

Yan An – Graphics Tools QA Manager

AGENDA

Introduction & tour of Nsight

Analyze a geometry corruption bug

VR debugging

System Analysis Tracing

GPU Range Profiling

UNREAL ENGINE

Compile Debug Profile

Microsoft DirectX · OpenGL · NVIDIA CUDA · OpenGL ES · Android NDK · GNU C/C++

Trace

OpenCV · NVIDIA VISIONWORKS · cuDNN · NVTX NVIDIA Tools eXtension

IDE Integration

Visual Studio · eclipse

Standalone and CLI

Getting Started...

JetPack

Hardware Support

NSIGHT VISUAL STUDIO EDITION 5.1

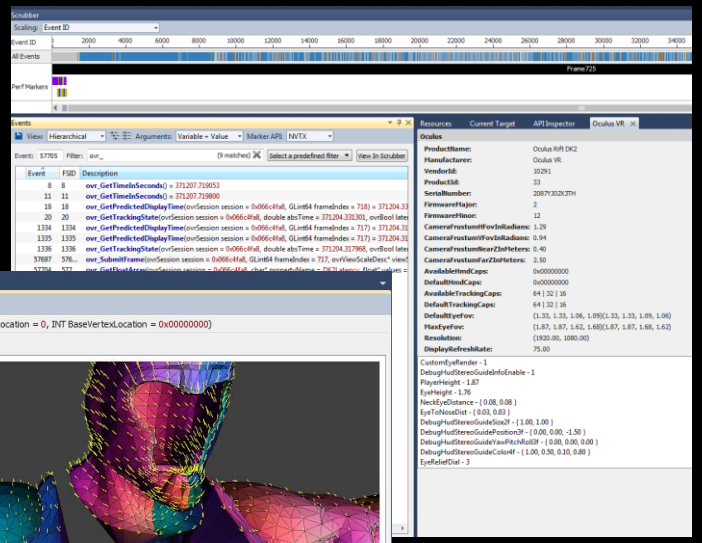
VR Development and Advanced Graphics Profiling

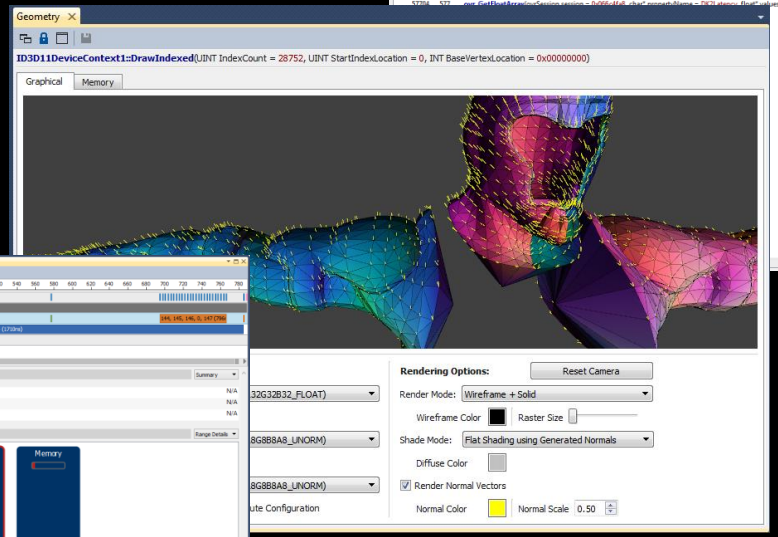
New Range Profiler, including DirectX12

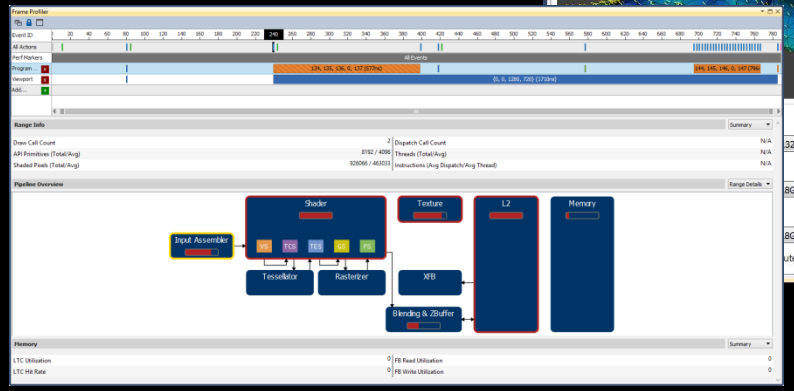
New Geometry View

Oculus VR SDK support

CUDA 8.0 support







UE4's VR Sample - Showdown

Released at Oculus Connect

Immersive firefight experience ala The Matrix

The engine completes a render pass on the left eye, then proceeds to the right

The same render target is used for each eye

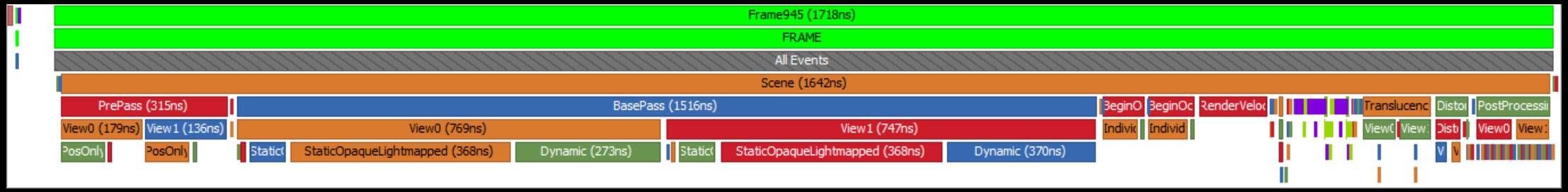
UE4's VR Engine

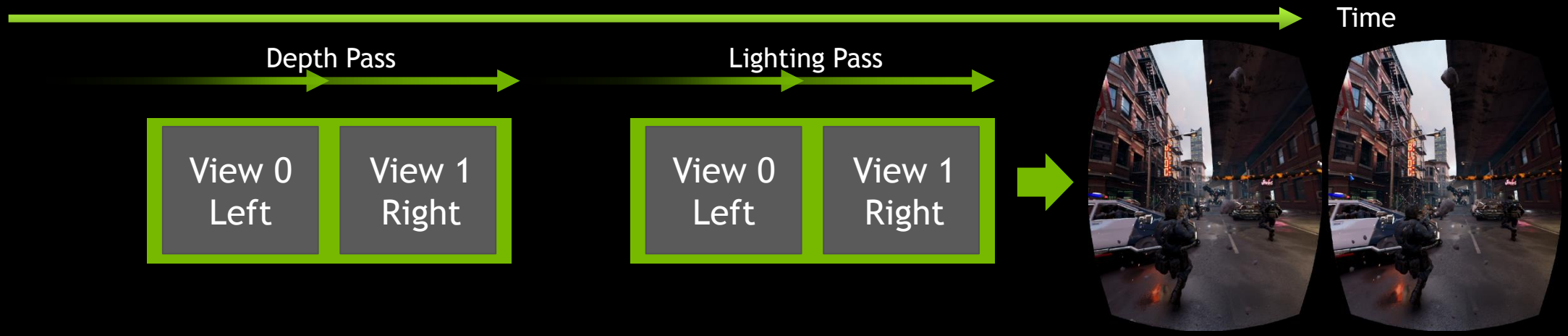
VR Processing

Once complete, the frame is submitted to Oculus for processing

UE4's VR Engine

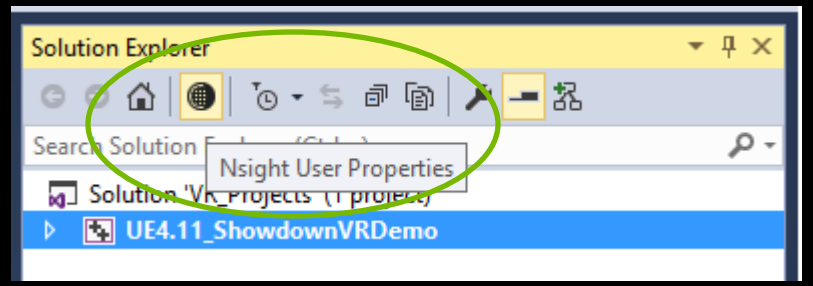
Render pass per eye





Walkthrough

Walkthrough

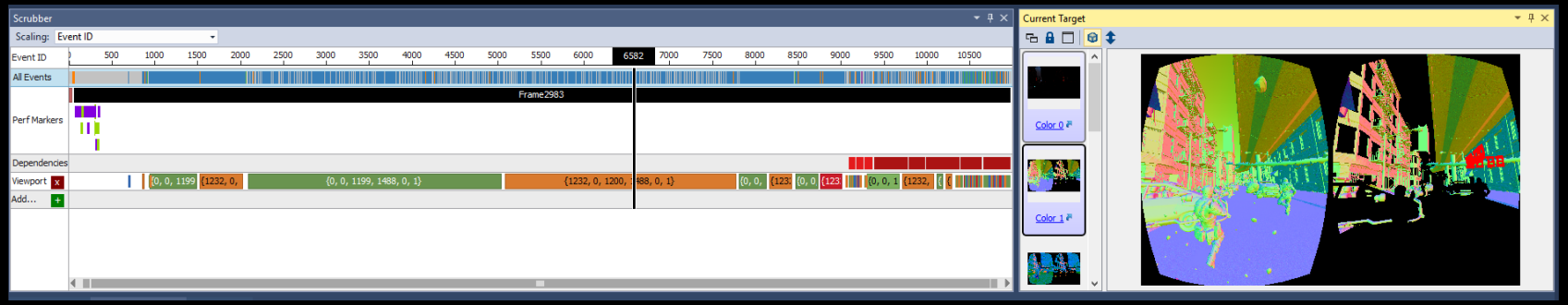


- Configure your project's Nsight User Settings
- Nsight -> Start Graphics Debugging
- Look at the HUD
  - Status – FPS, etc.
  - Performance Graphs
- Hit hot key Ctrl-Z – now Nsight has keyboard
  - Can move graphs around, etc.

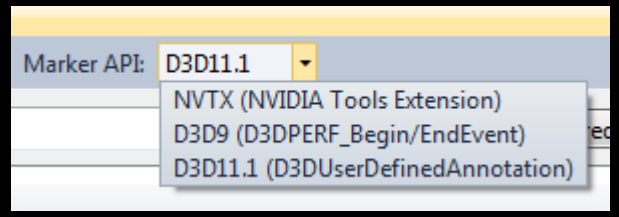
Walkthrough

- Hit space bar to capture frame
- Real time capture, app is still running, and pause the loop
- HUD capabilities
  - Scrub scene
  - Inspect textures & RTs
  - All of these features are available w/o VS, can drag & drop to desktop icon
  - Can serialize a frame to generated C++ source code, support VR API!

Walkthrough



- Alt+Tab back to VS
- Scrubber
  - Scrub through scene, get context via Perf Markers
- Event View
  - Display API call with parameters
  - Connected to scrubber
  - Filtering – ‘draw|pssetshader\(‘
  - If you are leveraging middleware, you may not have a choice of the perfmarker that they use. You can specify your perfmarker to investigate your own code.
- Current Targets - view all bound, can switch different RTVs and DSVs



Walkthrough

- Make sure you are on a draw call, let's look at the "API Inspector"
- Basic Graphics/GPU pipeline on left
- Each stage contains the state of the API at current draw call
  - IA: all of your vertex setup
  - VS, HS, DS, PS, CS: shader stages
  - SO
  - Raster
  - OM

API Inspector

Input Assembler

IA
VS
HS
DS
GS
SO
RS
PS
OM
CS

ID3D11DeviceContext1::DrawIndexed(UINT IndexCount = 6012, UINT S

Global Settings

PrimitiveTopology: D3D_PRIMITIVE_TOPOLOGY_TRIANGLELIST

Input Layout

Slot	SemanticName	SemanticIndex	Format
0	ATTRIBUTE	0	DXGI_FORMAT_R32G32B32_FLOA
1	ATTRIBUTE	1	DXGI_FORMAT_R8G8B8A8_UNOR
2	ATTRIBUTE	2	DXGI_FORMAT_R8G8B8A8_UNOR
3	ATTRIBUTE	4	DXGI_FORMAT_R16G16B16A16_F

Vertex Buffers

Slot	Stride	Offset	Size	Usage	Bind
0	12	0	93348	D3D11_USAGE_DEFAULT	D3D11_BIND_VE
1	16	0	124464	D3D11_USAGE_DEFAULT	D3D11_BIND_VE
2	0	0	4	D3D11_USAGE_DEFAULT	D3D11_BIND_SH

Walkthrough

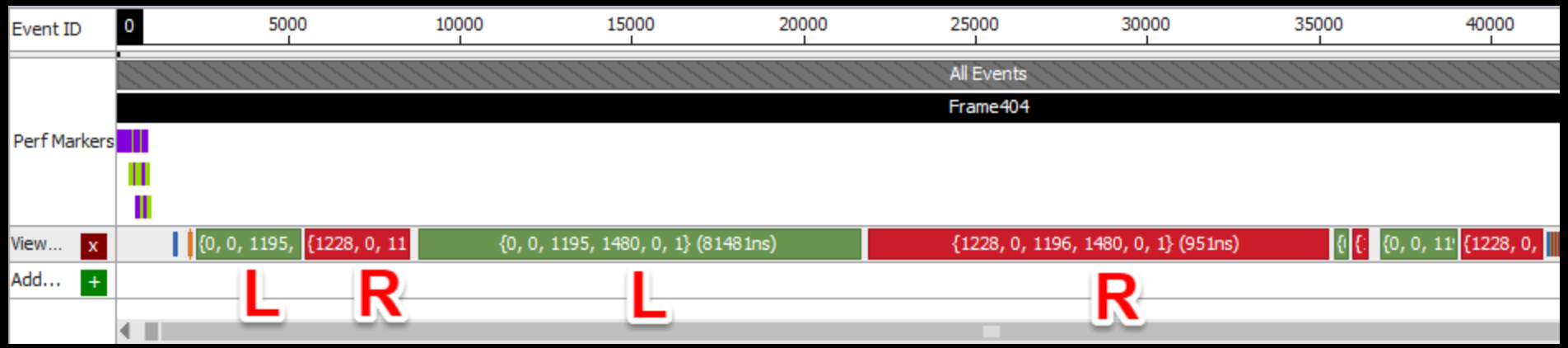
- Open up the “API Statistics” view just over 1k draw calls but almost 30k “other”
- AddRef/Release is ~13k calls!
- If you filtered for “GetDesc”…600 Calls but Descs are immutable. This might be worth caching
- Sort the object column
- Look at the textures object ids; note that some are called 8-12 times
- Caching/storage cost for D3D11_TEXTURE2D_DESC is 44 bytes

Summary

Draws:	1185	**Clears:**	44	**Presents:**	1	**Misc. Data Update:**	0	**Other:**	39205
Dispatches:	2	**Blits:**	1147	**Command List Executes**	0	**Non-API:**	1	**Total:**	41585

Walkthrough

- Some of the normal features in Nsight are useful for VR Debugging
- New scrubber feature: add rows with ranges that have different characteristics
- Click the "+" button and select "viewport", offers a high-level view of each eye's render pass
- This is especially useful if you haven't configured your own perfmarkers



WALKTHROUGH

Welcome to Nsight!

HUD performance graphs & scrubbing

Visual Studio integrated views

- Scrubber, Events & Current Target
- API Inspector & API Statistics
- Geometry & Resources

Utilize basic views to find potential CPU perf issues

Bad Geometry

Bad Geometry

- Start Nsight as we described in “Walkthrough” section
- Not a real bug from UE, but representative of a class of problems users have reported: what happens to my render result?

Bad Geometry

- Pause and capture the frame, scrub on the HUD to the **bad draw call**
- Many causes to this type of problem, use Nsight to check things that can go wrong
  - API Inspector – VS stage
  - API Inspector – RS stage
  - Etc.

Bad Geometry

- API Inspector – VS stage, collapse all and check buffers, is anything wrong?

API Inspector

Vertex Shader: IA | VS | HS | DS | GS | SO | RS | PS | OM | CS

ID3D11DeviceContext1::DrawIndexed(UINT IndexCount = 93144, UINT StartIndexLocation = 194592, INT BaseVertexLocation = 0x00000000)

VertexShader uid:5422

Source code (Main)

Shader Resource Views

SRV uid:1254 (Slot 0)

View

Name:	SRV uid:1254
Format:	DXGI_FORMAT_R32G32B32A32_FLOAT
ViewDimension:	D3D_SRV_DIMENSION_BUFFER
ElementOffset:	0
ElementWidth:	192

Resource Description

Name:	Buffer uid:1253
ByteWidth:	3072
Usage:	D3D11_USAGE_DYNAMIC
BindFlags:	D3D11_BIND_SHADER_RESOURCE | D3D11_BIND_VERTEX_BUFFER
CpuAccessFlags:	D3D11_CPU_ACCESS_WRITE
MiscFlags:	0x00000000u
StructureByteStride:	0

Constant Buffers

Slot	HLSL Buffer Name	HLSL Variables	Width	Stride	Object Name	FirstConstant	NumConstants	Usage	BindFlags	AccessFlags	MiscFlags
0	$Globals	11	512	0	Buffer uid:43	0	4096	D3D11_USAGE_DEFAULT	D3D11_BIND_CONSTANT_BUFFER	0x00000000u	0x00000000u
1	View	114	4096	0	Buffer uid:5384	0	4096	D3D11_USAGE_DYNAMIC	D3D11_BIND_CONSTANT_BUFFER	D3D11_CPU_ACCESS_WRITE	0x00000000u
2	Primitive	18	512	0	Buffer uid:3756	0	4096	D3D11_USAGE_DYNAMIC	D3D11_BIND_CONSTANT_BUFFER	D3D11_CPU_ACCESS_WRITE	0x00000000u

Samplers

Slot	Filter	Address U	Address V	Address W	MipLODBias	MaxAnisotropy	ComparisonFunc	RGBA
0	D3D11_FILTER_MIN_MAG_MIP_POINT	D3D11_TEXTURE_ADDRESS_WRAP	D3D11_TEXTURE_ADDRESS_WRAP	D3D11_TEXTURE_ADDRESS_WRAP	0.00	0	D3D11_COMPARISON_NEVER	
1	D3D11_FILTER_MIN_MAG_MIP_POINT	D3D11_TEXTURE_ADDRESS_CLAMP	D3D11_TEXTURE_ADDRESS_CLAMP	D3D11_TEXTURE_ADDRESS_CLAMP	0.00	0	D3D11_COMPARISON_NEVER	
2	D3D11_FILTER_MIN_MAG_MIP_POINT	D3D11_TEXTURE_ADDRESS_CLAMP	D3D11_TEXTURE_ADDRESS_CLAMP	D3D11_TEXTURE_ADDRESS_CLAMP	0.00	0	D3D11_COMPARISON_NEVER	

Bad Geometry

- API Inspector – RS stage, check fill mode, cull mode, depth bias, etc. is anything wrong?

API Inspector

Rasterizer

IA | VS | HS | DS | GS | SO | RS | PS | OM | CS

ID3D11DeviceContext1::DrawIndexed(UINT IndexCount = 93144, UINT StartIndexLocation = 194592, INT BaseVertexLocation = 0x00000000)

Rasterizer State

FillMode:	D3D11_FILL_SOLID	**DepthBias:**	0	**DepthClipEnable:**	TRUE	**AntialiasedLineEnable:**	FALSE
CullMode:	D3D11_CULL_BACK	**DepthBiasClamp:**	0.00	**ScissorEnable:**	TRUE		
FrontCounterClockwise:	TRUE	**SlopeScaledDepthBias:**	0.00	**MultisampleEnable:**	TRUE		

Viewports

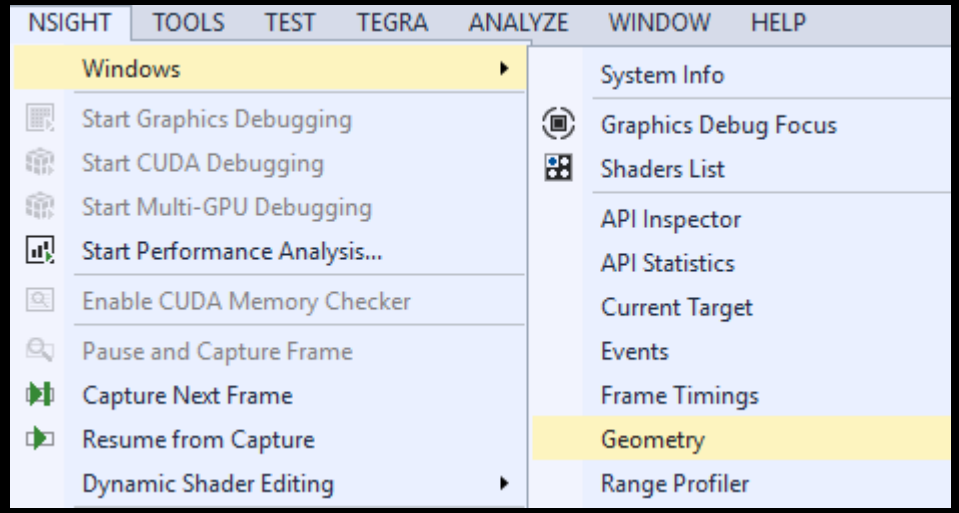
	TopLeftX	TopLeftY	Width	Height	MinDepth	MaxDepth
0	0.00	0.00	1195.00	1480.00	0.00	1.00

Scissors

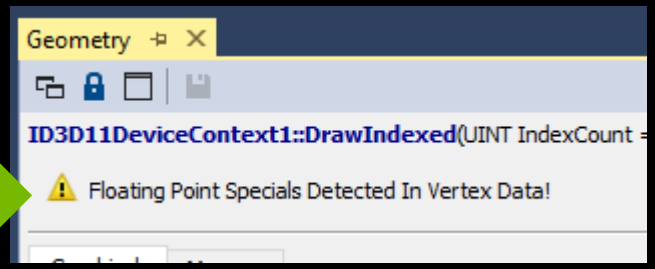
	Left	Top	Right	Bottom
0	0	0	16384	16384

Bad Geometry

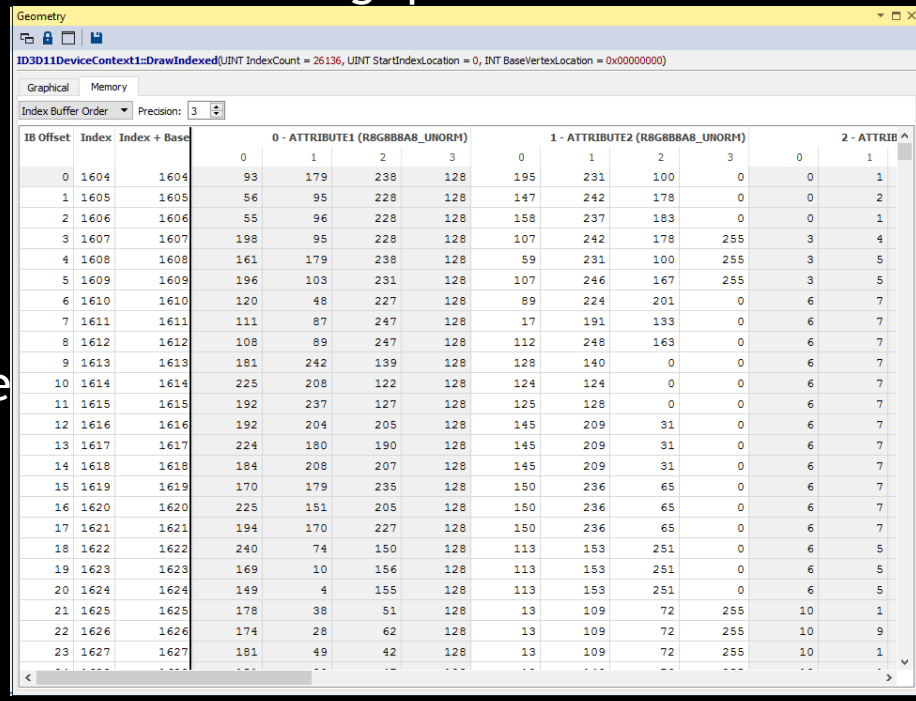
- Looks like our processing is set up properly, going to guess it might be a data input problem
- Let us open Geometry View!



Bad Geometry



- Open the Geometry View [new view with Nsight 5.1]
  - Can already see that Nsight detected some FP specials (let us discuss later)
- Geometry Graphical View
  - Ability to map any attribute to one of 3 characteristics for the rendering: pos/color/normal
  - Visualize things like normals or UV as color
  - Settings persisted according to the vertex layout
- Geometry Memory Table View
  - Each vertex is a row with the columns as each attribute
  - Can see in index buffer order or vertex buffer order
  - Takes into account any offsets being used



IB Offset	Index	Index + Base	0 - ATTRIBUTE1 (R8G8B8A8_UNORM)				1 - ATTRIBUTE2 (R8G8B8A8_UNORM)				2 - ATTRIB	
			0	1	2	3	0	1	2	3	0	1
0	1604	1604	93	179	238	128	195	231	100	0	0	1
1	1605	1605	56	95	228	128	147	242	178	0	0	2
2	1606	1606	55	96	228	128	158	237	183	0	0	1
3	1607	1607	198	95	228	128	107	242	178	255	3	4
4	1608	1608	161	179	238	128	59	231	100	255	3	5
5	1609	1609	196	103	231	128	107	246	167	255	3	5
6	1610	1610	120	48	227	128	89	224	201	0	6	7
7	1611	1611	111	87	247	128	17	191	133	0	6	7
8	1612	1612	108	89	247	128	112	248	163	0	6	7
9	1613	1613	181	242	139	128	128	140	0	0	6	7
10	1614	1614	225	208	122	128	124	124	0	0	6	7
11	1615	1615	192	237	127	128	125	128	0	0	6	7
12	1616	1616	192	204	205	128	145	209	31	0	6	7
13	1617	1617	224	180	190	128	145	209	31	0	6	7
14	1618	1618	184	208	207	128	145	209	31	0	6	7
15	1619	1619	170	179	235	128	150	236	65	0	6	7
16	1620	1620	225	151	205	128	150	236	65	0	6	7
17	1621	1621	194	170	227	128	150	236	65	0	6	7
18	1622	1622	240	74	150	128	113	153	251	0	6	5
19	1623	1623	169	10	156	128	113	153	251	0	6	5
20	1624	1624	149	4	155	128	113	153	251	0	6	5
21	1625	1625	178	38	51	128	13	109	72	255	10	1
22	1626	1626	174	28	62	128	13	109	72	255	10	9
23	1627	1627	181	49	42	128	13	109	72	255	10	1

Bad Geometry

- Back to the “FP specials detection”
- Could use the “tag” feature in the resources view to show all uses of this buffer in the scene. Not only tag the vertex buffers, but also support any resources, like texture

Geometry

ID3D11DeviceContext1::DrawIndexed(UINT IndexCount = 25056, UINT StartIndexLocation = 96240, INT BaseVertexLocation = 0x00000000)

Floating Point Specials Detected In Vertex Data!

Graphical | Memory

Index Buffer Order | Precision: 3 | Filter To Specials

IB Offset	Index	Index + Base	2 - ATTRIBUTE3 (R8G8B8A8_UINT)				3 - ATTRIBUTE4 (R8G8B8A8_UNORM)				4 - ATTRIBUTE0 (R32G32B32_FLOAT)			5 - ATTRIBUTE8 (R16G.	
			0	1	2	3	0	1	2	3	0	1	2	0	1
0	220...	22082	4	0	0	0	255	0	0	0	116.805	nan	204.358	1.525	0.49
1	220...	22088	4	0	0	0	255	0	0	0	nan	55.767	197.995	1.576	0.47
2	220...	22089	4	0	0	0	255	0	0	0	116.143	56.535	197.428	1.576	0.49
3	220...	22081	4	0	0	0	255	0	0	0	113.202	57.592	204.868	1.525	0.47
4	220...	22090	4	0	0	0	255	0	0	0	111.499	52.312	198.907	1.576	0.44
5	220...	22088	4	0	0	0	255	0	0	0	nan	55.767	197.995	1.576	0.47
6	220...	22081	4	0	0	0	255	0	0	0	113.202	57.592	204.868	1.525	0.47
7	220...	22088	4	0	0	0	255	0	0	0	nan	55.767	197.995	1.576	0.47
8	220...	22082	4	0	0	0	255	0	0	0	116.805	nan	204.358	1.525	0.49
9	220...	22075	4	0	0	0	255	0	0	0	117.024	60.949	210.377	1.479	0.49
10	220...	22081	4	0	0	0	255	0	0	0	113.202	57.592	204.868	1.525	0.47
11	220...	22082	4	0	0	0	255	0	0	0	116.805	nan	204.358	1.525	0.49
12	220...	22083	4	0	0	0	255	0	0	0	112.184	54.229	206.098	1.525	0.44
13	220...	22090	4	0	0	0	255	0	0	0	111.499	52.312	198.907	1.576	0.44
14	220...	22081	4	0	0	0	255	0	0	0	113.202	57.592	204.868	1.525	0.47
15	220...	22081	4	0	0	0	255	0	0	0	113.202	57.592	204.868	1.525	0.47
16	220...	22074	4	0	0	0	255	0	0	0	113.417	60.127	210.799	1.479	0.47
17	220...	22083	4	0	0	0	255	0	0	0	112.184	54.229	206.098	1.525	0.44

BAD GEOMETRY RECAP

Corruption in vertex data

API Inspector to check pipeline setup

Geometry View to inspect pre-transform data

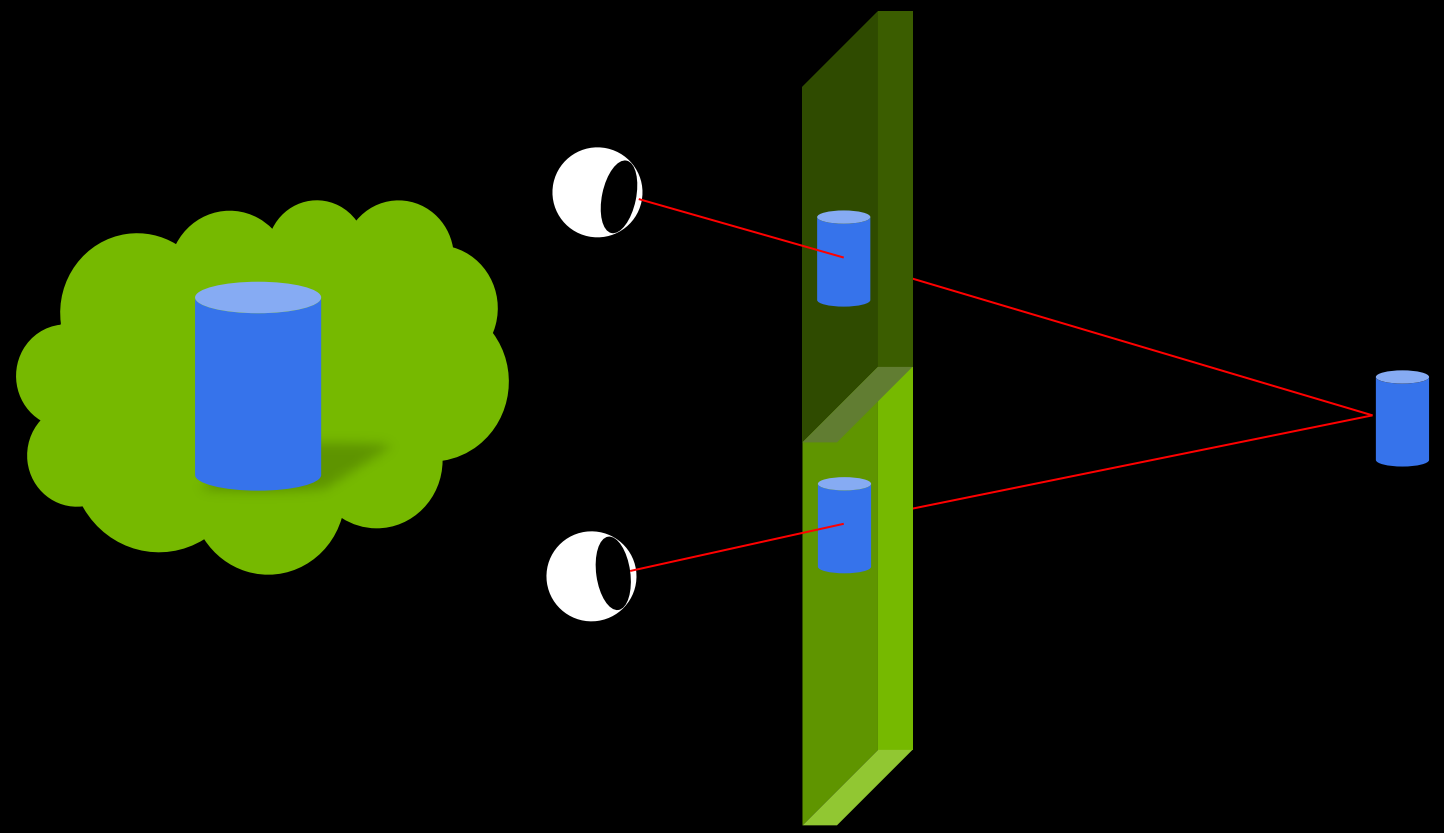
Resources View + ‘tag’ to show all uses

VR…not so Reality

Parallax & Binocular Vision

- pick an object in the foreground, like your figner or someone's head
- Now alternate covering your left and right eye
- Notice how the object 'moves' relative to the background as you switch eyes
- This is called "parallax" and is part of the nature of our binocular vision system
- This gives us the ability to perceive depth, and is what is used in stereo/VR to give a much more immersive experience

Parallax & Binocular Vision



VR…not so Reality

- Sometimes, stereo can break down in your VR rendering and kill the experience
- Ever seen cheesy “3D” where everything just looks like several different depths of billboards?
- In this scene, look at the cement chunks that are flying through the air
- If you had an HMD, you would see them floating unnaturally in space, almost like a HUD element or reticle site


16.2 FPS, 61.78 ms/frame
Active shaders: Original
HUD activated
Geom Busy
Shader Busy
Texture Busy
25.0
12.5
0.0
GPU Busy
50.0
25.0
0.0
Histogram: prims per draw
750
375
0
HW

VR…not so Reality

- Start Nsight as we described in “Walkthrough” section
- Capture the frame (Ctrl-Z and Space as shortcut keys)
- Let’s use the Resources View to dig into the texture used here to confirm what we are seeing
- First, let us find the drawcalls for the “not so reality” effect

VR…not so Reality

- Go to event that render the left eye
- Hit Ctrl+F2 to bookmark it in Event View
- Go to event that render the right eye
- Hit Ctrl+F2 to bookmark it in Event View
- Again, if we didn’t have Perf Markers, you could use a new Viewport row on the scrubber to confirm that these calls go to different eyes
- Go to CTV (Current Target View) and click on the depth texture, it will open Resources View for detail inspect

Scrubber

Scaling: Event ID

Event ID 500 1000 1500 2000 2500 3000 3500 4000 4500 5000 5500 6000 6500 7000 7780 8500 9000 9500 10000 10500

All Events

Frame2983

Perf Markers

Dependencies

Viewport {0, 0, 1199 {1232, 0, {0, 0, 1199, 1488, 0, 1} {1232, 0, 1200, 1488, 0, 1}

Add...

Current Target

Color 5

Depth

Geometry API Inspector Events

View: Hierarchical Arguments: Variable + Value Marker API: NVTX

Event: 7780 Filter: Enter a filter or select a predefined one on the right Select a predefined filter

Event	Description	Object	CPU µs	GPU µs	Thread
7769	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 0, UINT NumBuf	0x000001F...	0	-	8524
7770	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 2, UINT NumBuf	0x000001F...	<1	-	8524
7771	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 3, UINT NumBuf	0x000001F...	0	-	8524
7772	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 4, UINT NumBuf	0x000001F...	0	-	8524
7773	ID3D11DeviceContext1::IASetIndexBuffer(ID3D11Buffer* pIndexBuffer = 'Bu	0x000001F...	0	-	8524
7774	ID3D11DeviceContext1::DrawIndexedInstanced(UINT IndexCountPerInstance	0x000001F...	0	10	8524
7775	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 0, UINT NumBuf	0x000001F...	0	-	8524
7776	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 2, UINT NumBuf	0x000001F...	0	-	8524
7777	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 3, UINT NumBuf	0x000001F...	<1	-	8524
7778	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 4, UINT NumBuf	0x000001F...	0	-	8524
7779	ID3D11DeviceContext1::IASetIndexBuffer(ID3D11Buffer* pIndexBuffer = 'Bu	0x000001F...	0	-	8524
7780	ID3D11DeviceContext1::DrawIndexedInstanced(UINT IndexCountPerInstance	0x000001F...	0	13	8524
7781	ID3D11DeviceContext1::IASetInputLayout(ID3D11InputLayout* pInputLayou	0x000001F...	<1	-	8524
7782	ID3D11DeviceContext1::VSSetShader(ID3D11VertexShader* pVertexShader =	0x000001F...	0	-	8524
7783	ID3D11DeviceContext1::PSSetShader(ID3D11PixelShader* pPixelShader = 'Pi	0x000001F...	0	-	8524
7784	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 0, UINT NumBuf	0x000001F...	<1	-	8524
7785	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 1, UINT NumBuf	0x000001F...	<1	-	8524
7786	ID3D11DeviceContext1::IASetVertexBuffers(UINT StartSlot = 2, UINT NumBuf	0x000001F...	0	-	8524
7787	ID3D11DeviceContext1::VSSetConstantBuffers(UINT StartSlot = 1, UINT Numl	0x000001F...	0	-	8524
7788	ID3D11DeviceContext1::PSSetConstantBuffers(UINT StartSlot = 3, UINT Num	0x000001F...	0	-	8524
7789	ID3D11DeviceContext1::VSSetConstantBuffers(UINT StartSlot = 4, UINT Numl	0x000001F...	0	-	8524
7790	ID3D11DeviceContext1::VSSetConstantBuffers(UINT StartSlot = 3, UINT Numl	0x000001F...	<1	-	8524
7791	ID3D11DeviceContext1::PSSetShaderResources(UINT StartSlot = 0, UINT Num	0x000001F...	<1	-	8524

VR…not so Reality

- Check the current DSV
- Note that you can see just black image.
- Open up the histogram and renormalize (click the little arrow on the bottom right side of texture display), and drag the max arrow to adjust the range of your depth, so that you can see details
- Note the depth of the chunk for each eye

Resources

Textures (365) | Buffers (1146) | Thumbnails

Filter: Enter a filter or select a pred | Select a predefined filter

Texture2D 775 | Texture2D 776 | Texture2D 777 | Texture2D 778
Texture2D 779 | Texture2D 780 | Texture2D 781 | Texture2D 785
Texture2D 790 | Texture2D 798 | Texture2D 800 | Texture2D 802
Texture2D 830 | Texture2D 830 | Texture2D 830 | Texture2D 850

Available Views (2)

Name:	DSV 81
Format:	DXGI_FORMAT_D24_UNORM_S8_UINT
ViewDimension:	D3D11_DSV_DIMENSION_TEXTURE2D
MipSlice:	0

Graphical | Memory

Viewer Focus: Mip Level
Mip Level = 0

0.000000 Min | Max 0.014795

Pos: (1837, 850) Raw Value: 0x9626e Normalize: By Histogram View Stencil

Revision Information

Resource Info

Type:	2D Texture
Name:	Texture2D 830
Revision:	843
Size:	2432 x 1488
Memory:	13.80 MB
Format:	DXGI_FORMAT_R24G8_TYPELESS
# Mips:	1
# Consumptions:	76

Available Revisions (891)

Rev: 832 Evt: 7692 | Rev: 833 Evt: 7698 | Rev: 834 Evt: 7704 | Rev: 835 Evt: 7710 | Rev: 836 Evt: 7716 | Rev: 837 Evt: 7722 | Rev: 838 Evt: 7728 | Rev: 839 Evt: 7746 | Rev: 840 Evt: 7762 | Rev: 841 Evt: 7768 | Rev: 842 Evt: 7774 | Rev: 843 Evt: 7780 | Rev: 844 Evt: 7795 | Rev: 845 Evt: 9034 | Rev: 846 Evt: 9282 | Rev: 847 Evt: 9435

VR…not so Reality

- Use API Inspector as in previous section and confirm the setup of rendering, and that it is all OK.
- What's else? We need to look at the VR setup!
- Open VR Inspector view
- You will see we show Swap Chains, Mirror Textures, Render Desc Queries, info about the HMD

VR Inspector

Show API Usage

Swap Chains

	Resources	CurrentIndex	Type	Format	ArraySize	Width	Height	MipLevels	SampleCount	StaticImage	MiscFlags
0	Texture2D 342	0	ovrTexture_2D	OVR_FORMAT_B8G8R8A8_UNORM_SRGB	1	16	16	5	1	ovrTrue	ovrTextureMisc_AllowGenerateMip
1	Texture2D 343 Texture2D 344 Texture2D 345	0	ovrTexture_2D	OVR_FORMAT_B8G8R8A8_UNORM_SRGB	1	2432	1488	1	1	ovrFalse	ovrTextureMisc_DX_Typeless

Mirror Textures

	Resource	Format	Width	Height	MiscFlags
0	Texture2D 346	OVR_FORMAT_B8G8R8A8_UNORM_SRGB	1920	1080	ovrTextureMisc_DX_Typeless

Render Desc Queries

	Eye	Fov	DistortedViewport	PixelsPerTanAngleAtCenter	HmdToEyeOffset
0	ovrEye_Left	(1.33, 1.33, 1.06, 1.09)	(0.00, 0.00) (960.00, 1080.00)	(549.62, 549.62)	(-0.03, 0.00, 0.00)
1	ovrEye_Left	(1.33, 1.33, 1.09, 1.06)	(0.00, 0.00) (960.00, 1080.00)	(549.62, 549.62)	(-0.03, 0.00, 0.00)

HMD Description

Type:	ovrHmd_DK2
ProductName:	Oculus Rift DK2
Manufacturer:	Oculus VR
VendorId:	10291
ProductId:	33
SerialNumber:	209QKS04EVHM
FirmwareMajor:	2
FirmwareMinor:	12
RuntimeVersion:	1.6.0
AvailableHmdCaps:	0x00000000u
DefaultHmdCaps:	0x00000000u
AvailableTrackingCaps:	ovrTrackingCap_Position | ovrTrackingCap_MagYawCorrection | ovrTrackingCap_Orientation
DefaultTrackingCaps:	ovrTrackingCap_Position | ovrTrackingCap_MagYawCorrection | ovrTrackingCap_Orientation
DefaultEyeFov:	(1.33, 1.33, 1.06, 1.09) (1.33, 1.33, 1.09, 1.06)
MaxEyeFov:	(1.87, 1.87, 1.62, 1.68) (1.87, 1.87, 1.68, 1.62)
Resolution:	(1920.00, 1080.00)
DisplayRefreshRate:	75.00
LayerHudShowAll:	ovrFalse
DebugHudStereoGuideInfoEnable:	ovrTrue
PlayerHeight:	1.78
EyeHeight:	1.67
NeckEyeDistance:	{ 0.08, 0.08 }
EyeToNoseDist:	{ 0.03, 0.03 }
DebugHudStereoGuideSize2f:	{ 1.00, 1.00 }
DebugHudStereoGuidePosition3f:	{ 0.00, 0.00, -1.50 }
DebugHudStereoGuideYawPitchRoll3f:	{ 0.00, 0.00, 0.00 }
DebugHudStereoGuideColor4f:	{ 1.00, 0.50, 0.10, 0.80 }
PerfHudMode:	0
LayerHudMode:	0
LayerHudCurrentLayer:	0
DebugHudStereoMode:	0
User:	Unknown
Name:	Unknown
Gender:	Male

VR…not so Reality

- Click through to ‘Show API Usage’ to show VR APIs in the events view

Event ID | All Events | Perf Markers | Dependencies | Add…

Geometry | API Inspector | Events

View: Hierarchical | Arguments: Variable + Value | Marker API: NVTX

Event: 7780 | Filter: ovr_

Event	Description	Object
2	ovr_GetTimeInSeconds() = 569.551015	-
4	ovr_GetTimeInSeconds() = 569.551023	-
59	ovr_GetTimeInSeconds() = 569.577314	-
61	ovr_GetTimeInSeconds() = 569.577321	-
63	ovr_GetTimeInSeconds() = 569.577323	-
65	ovr_GetTimeInSeconds() = 569.577323	-
67	ovr_GetPredictedDisplayTime(ovrSession session = 0xb5cace38, int64_t frameIndex = 0x000009ac) = 0.000000	-
68	ovr_GetTrackingState(ovrSession session = 0xb5cace38, double absTime = 436.565595, ovrBool latencyMarker = ovrFalse) = { 'Tracking State' }	-
690	ovr_GetPredictedDisplayTime(ovrSession session = 0xb5cace38, int64_t frameIndex = 0x000009ab) = 0.000000	-
691	ovr_GetPredictedDisplayTime(ovrSession session = 0xb5cace38, int64_t frameIndex = 0x000009ab) = 0.000000	-
692	ovr_GetTrackingState(ovrSession session = 0xb5cace38, double absTime = 436.552261, ovrBool latencyMarker = ovrTrue) = { 'Tracking State' }	-
10932	ovr_CommitTextureSwapChain(ovrSession session = 0xb5cace38, ovrTextureSwapChain chain = 0xb59d8fc0) = 0x00000000	-
10933	ovr_GetTextureSwapChainCurrentIndex(ovrSession session = 0xb5cace38, ovrTextureSwapChain chain = 0xb59d8fc0, int32_t* out_Index = 0x000000A17D0FDAB4) = 0x00	-
10984	ovr_SubmitFrame(ovrSession session = 0xb5cace38, int64_t frameIndex = 0x00000000, ovrViewScaleDesc* viewScaleDesc = 0x00007FF66F613C9C, ovrLayerHeaderPtr* laye	-
10985	ovr_GetTimeInSeconds() = 569.726791	-
10986	ovr_GetTimeInSeconds() = 569.726792	-
10987	ovr_GetTimeInSeconds() = 569.730898	-
10988	ovr_GetTimeInSeconds() = 569.730898	-
10989	ovr_GetTextureSwapChainCurrentIndex(ovrSession session = 0xb5cace38, ovrTextureSwapChain chain = 0xb59d8fc0, int32_t* out_Index = 0x00007FF66F5E8498) = 0x000	-

VR Inspector

Show API Usage

Swap Chains

	Resources	CurrentIndex	Type	Format	ArraySize	Width
0	Texture2D 342	0	ovrTexture_2D	OVR_FORMAT_B8G8R8A8_UNORM_SRGB	1	16
1	Texture2D 343 Texture2D 344 Texture2D 345	0	ovrTexture_2D	OVR_FORMAT_B8G8R8A8_UNORM_SRGB	1	2432

Mirror Textures

	Resource	Format	Width	Height	MiscFlags
0	Texture2D 346	OVR_FORMAT_B8G8R8A8_UNORM_SRGB	1920	1080	ovrTextureMisc_DX_Typeless

Render Desc Queries

	Eye	Fov	DistortedViewport	PixelsPerTanAngleAtCenter	HmdToEyeOffset
0	ovrEye_Left	(1.33, 1.33, 1.06, 1.09)	(0.00, 0.00) (960.00, 1080.00)	(549.62, 549.62)	(-0.03, 0.00, 0.00)
1	ovrEye_Left	(1.33, 1.33, 1.09, 1.06)	(0.00, 0.00) (960.00, 1080.00)	(549.62, 549.62)	(-0.03, 0.00, 0.00)

HMD Description

Type:	ovrHmd_DK2	LayerHudShowAll:	
ProductName:	Oculus Rift DK2	DebugHudStereoGuideInfoE	
Manufacturer:	Oculus VR	PlayerHeight:	
VendorId:	10291	EyeHeight:	
ProductId:	33	NeckEyeDistance:	
SerialNumber:	209QKS04EVHM	EyeToNoseDist:	
FirmwareMajor:	2	DebugHudStereoGuideSize2	

VR…not so Reality

- Look closer at the Render Desc Queries section
- Note that the app queried for the left eye twice
- Must have been a typo where they meant to query for the right eye…copy paste strikes again
- Change to query the right eye, problem solved!

Render Desc Queries

	Eye	Fov	DistortedViewport	PixelsPerTanAngleAtCenter	HmdToEyeOffset
0	ovrEye_Left	(1.33, 1.33, 1.06, 1.09)	(0.00, 0.00) (960.00, 1080.00)	(549.62, 549.62)	(-0.03, 0.00, 0.00)
1	ovrEye_Left	(1.33, 1.33, 1.09, 1.06)	(0.00, 0.00) (960.00, 1080.00)	(549.62, 549.62)	(-0.03, 0.00, 0.00)

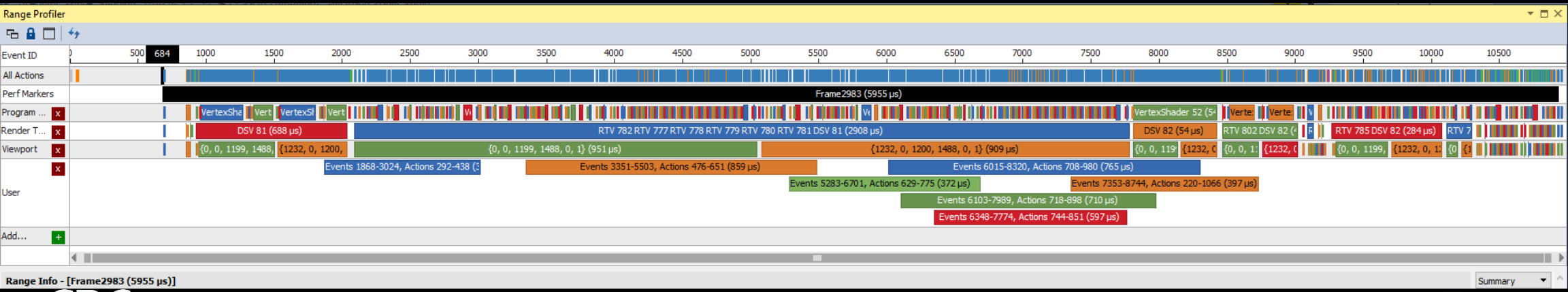
VR…NOT SO R RECAP

Bookmarking in the Events View
Resources View image inspection
Histogram viewer and remapping
API Inspector for the rendering setup
VR Inspector for the VR setup

Where’s the bottleneck?

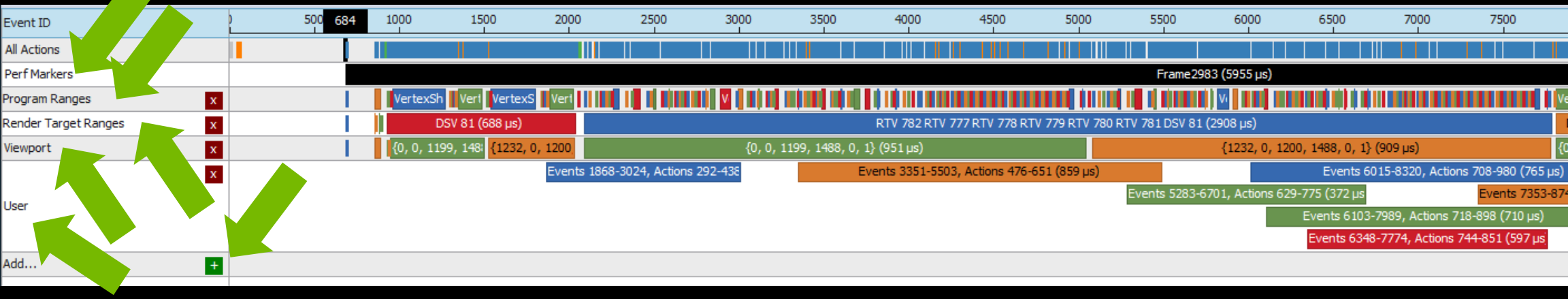
Where’s the bottleneck?

- Start Nsight as we described in “Walkthrough” section
- If you are GPU bound, you want to use a profiler to determine what part of the pipeline is overtaxed and leading to your performance drop
- Right click on scrubber and select “Profile Frame” to open “Range Profiler” view
- Profiler is draw call centric, allowed grouping of draw calls into state buckets
- New is “range” centric: a consecutive list of draw calls.



Where's my bottleneck?

- New paradigm is "ranges", which can be based on rendering state
  - Perf Markers
  - Program Ranges (all 6 shader stages)
  - Viewport (as we saw in the VR section)
  - Render targets (see how expensive each one is)
  - User – create a range on the fly



Where's my bottleneck?

- There are 3 sections for more detail profiler information
  - Range Info
    - Shows basic information about current range (primitives, pixels, compute shader, etc.)
    - Combo in upper right about 'Action Details' allows you to select per draw call info
    - One of the key improvement is the ability to selectively profile different sections for just the data you want. This will help in modern APIs where the draw call counts are going from a few thousand to 10s of thousands

Range Info - [Frame2983 (7442 µs)]

Action Details

Event	GPU Time	Draw Call Count	GPU Primitive Count	Shaded Pixels
684	96	1	2	49104
854	4640	1	16	32
857	128	1	16	16
861	0	1	16	80
864	0	1	16	80
868	0	1	16	16
885	3488	1	16	32

Event	GPU Time	Dispatch Call Count	Inst Executed
9232	240000	1	6.99606e+08
9243	240288	1	7.02002e+08

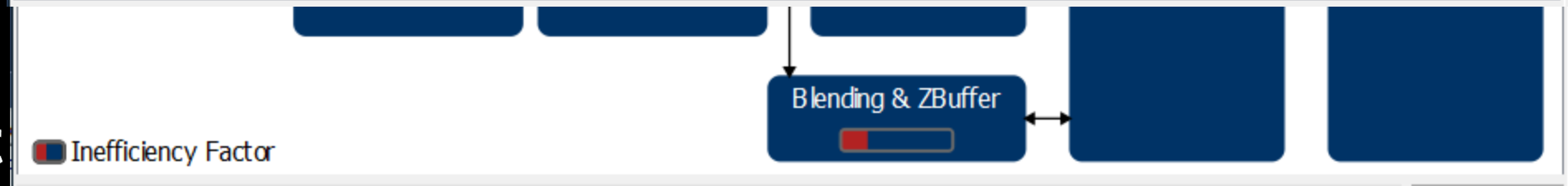
Where’s my bottleneck?

- Pipeline Overview
  - “Summary”, We show you the top 4 inefficient or bottlenecked stages
  - “Range Details” shows the classic pipeline diagram with each unit’s bottleneck
  - “Action Details” shows the detail information per drawcall

Pipeline Overview

Action Details

Event	VS Cycle Ratio	TCS Cycle Ratio	TES Cycle Ratio	GS Cycle Ratio	PS Cycle Ratio	CS Cycle Ratio	Input Assembler Inefficiency	Shader Inefficiency	Texture Inefficiency	L
684	0	0	0	0	15	0	1	36.5	4	1
854	1	0	0	0	4	0	0	13	4	6
857	1	0	0	0	2	0	0	10	6	5
861	0	0	0	0	8	0	0	16	7	5
864	1	0	0	0	9	0	0	18	4	4
868	1	0	0	0	2	0	0	11	6	4
885	0	0	0	0	7	0	0	15	5	5

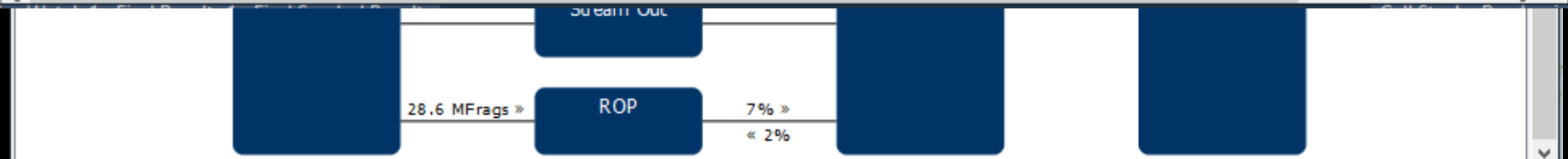


Where’s my bottleneck?

- Memory
  - “Summary”, we show shows some basic memory statistics including L2 usage, etc.
  - “Range Details” shows the requests/bytes between each one of the stages that utilize the memory subsystem
  - “Action Details” shows more detail info per draw call

Memory — Action Details

Event	L2->IA	Shader->Texture	Texture->Shader	Shader->ROP	Texture->L2	L2->Texture	Xfb->L2	ROP->L2	L2->ROP	L2->Memory	Memory->L2	L2 Hit Rate	L2 utilizat
684	0	0	0	0	0	0	0	3	0	3	1	91	1
854	0	0	0	32	0	0	0	0	0	0	0	49	0
857	0	0	0	16	0	0	0	0	0	0	1	56	0
861	0	0	0	80	0	0	0	0	0	0	0	64	0
864	0	0	0	80	0	0	0	0	0	0	0	32	0
868	0	0	0	16	0	0	0	0	0	0	0	23	0
885	0	0	0	32	0	0	0	0	0	0	0	48	0



Where's my bottleneck?

- Range Profiler and Dynamic Shader Edit
  - Pick really expensive draw call, make a user range around it and select
  - Go to API Inspector
  - Click shader link, see source code
  - Right click and use Shader Edit, code for shader optimization
  - Lock profiler and Clone a new profiler view
  - Rerun profiler and make sure the values for that draw call have dropped as your optimization

BOTTLENECK RECAP

Range Profiler

- Scrubber with ranges
- New data view(s)

Could optimize and rerun profile

- edit your shader on the fly, make it more efficient
- rerun Profiler, prove the optimize works as expected

Questions?

- Support forum of Nsight:
  - https://devtalk.nvidia.com/default/board/84/nsight-visual-studio-edition/
- Support Email:
  - devtools-support@nvidia.com
- Any question, discussion, suggestion is welcome!

Thanks! ☺