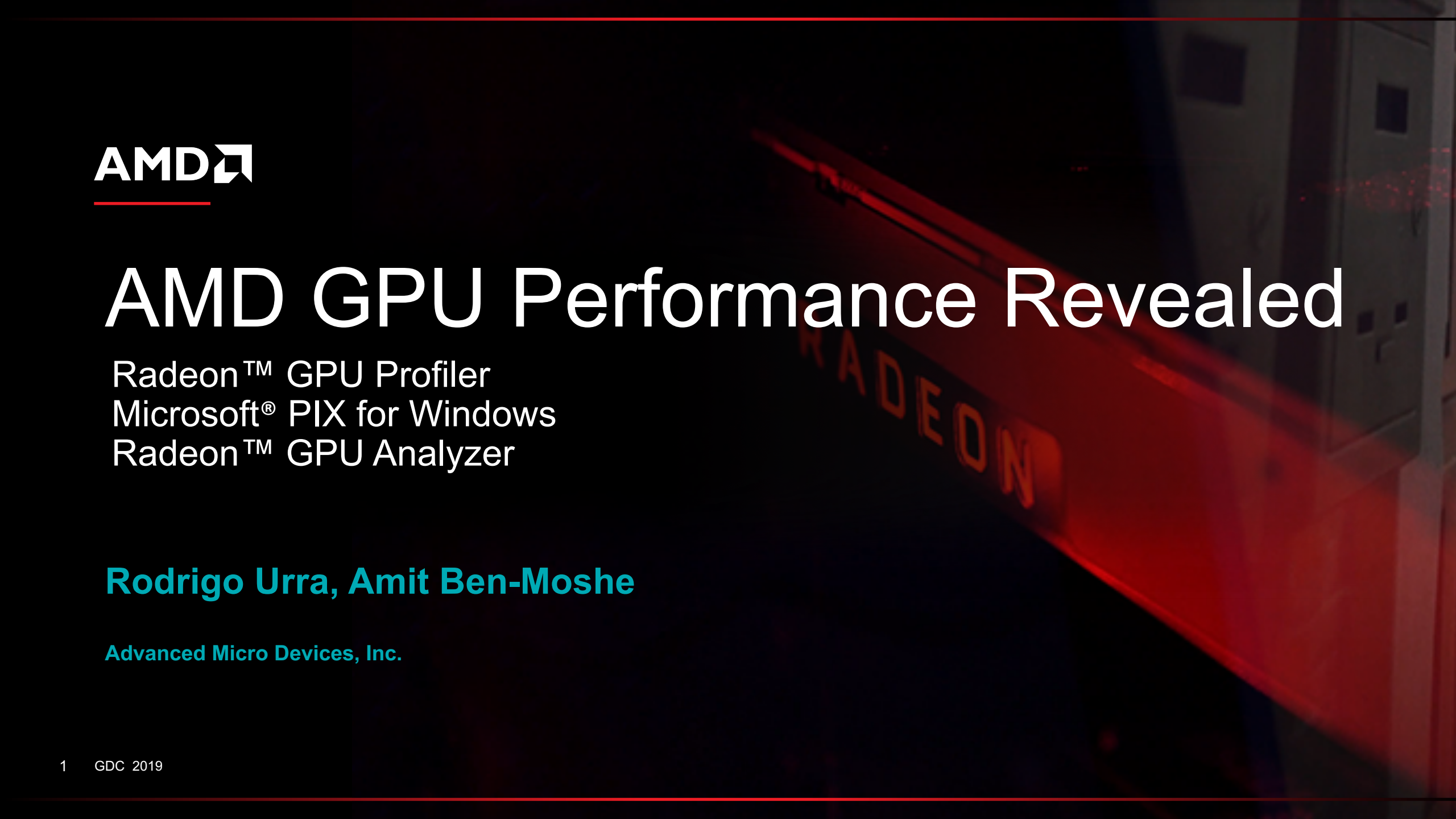


AMD GPU Performance Revealed

Radeon™ GPU Profiler
Microsoft® PIX for Windows
Radeon™ GPU Analyzer

Rodrigo Urra, Amit Ben-Moshe

Advanced Micro Devices, Inc.

AGENDA

Radeon™ GPU Profiler

Rodrigo Urra

Introduction

Tour of last year

What's new and coming

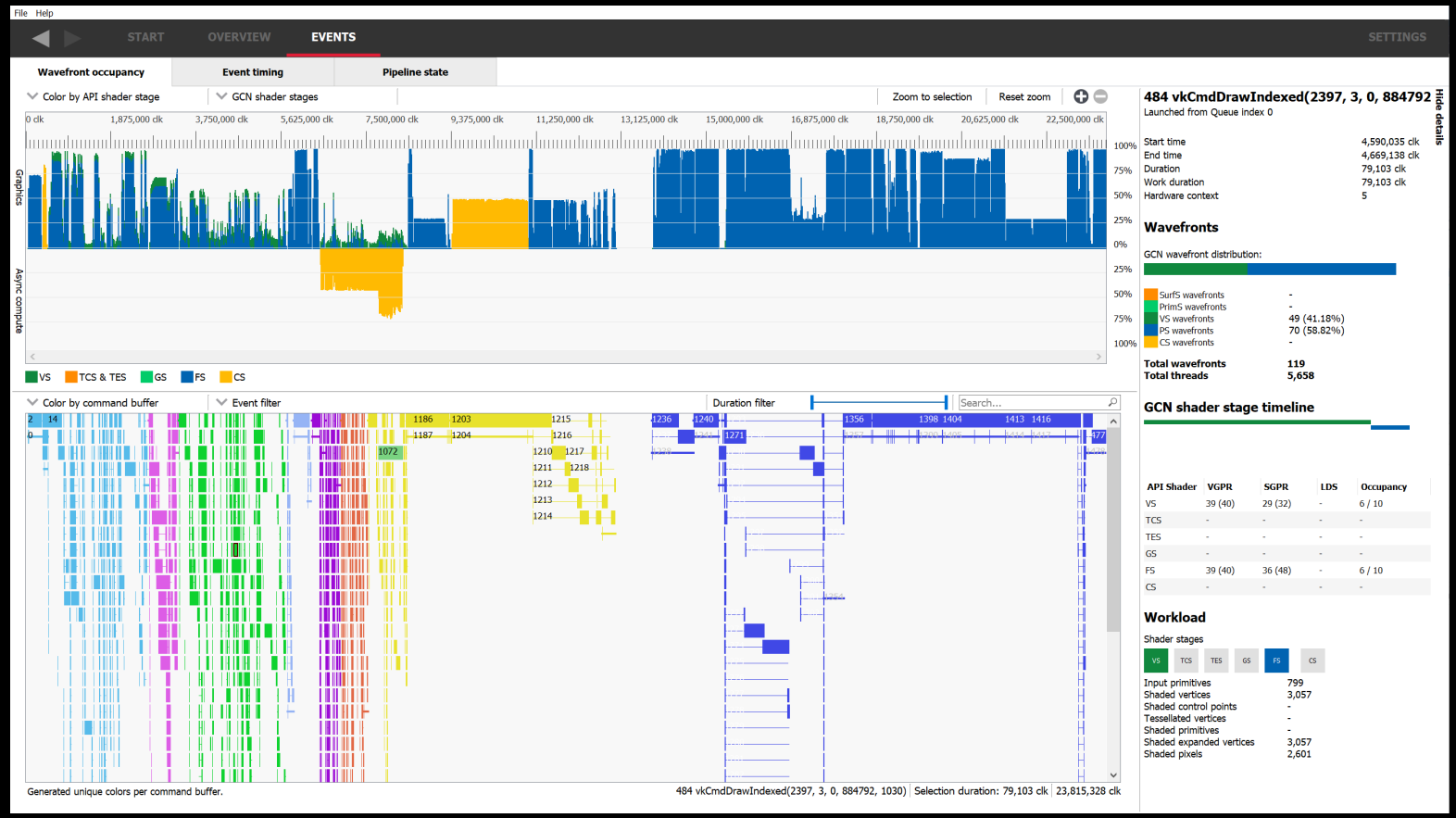
What is Radeon GPU Profiler?

GPU performance analysis tool

- Shows low level profiling data
- Visualizes GPU workloads
- Designed to identify performance bottlenecks
- Helps bridge the gap between explicit APIs and GCN

Designed to support

- Linux® and Windows®
- Vulkan® and DirectX® 12



GDC 2018 - RGP 1.2

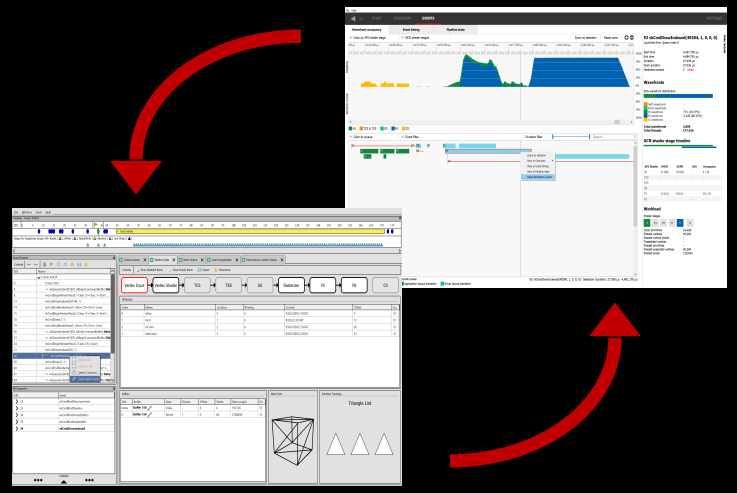
Overview

- Frame summary
- Most expensive events
- Barriers
- Context rolls

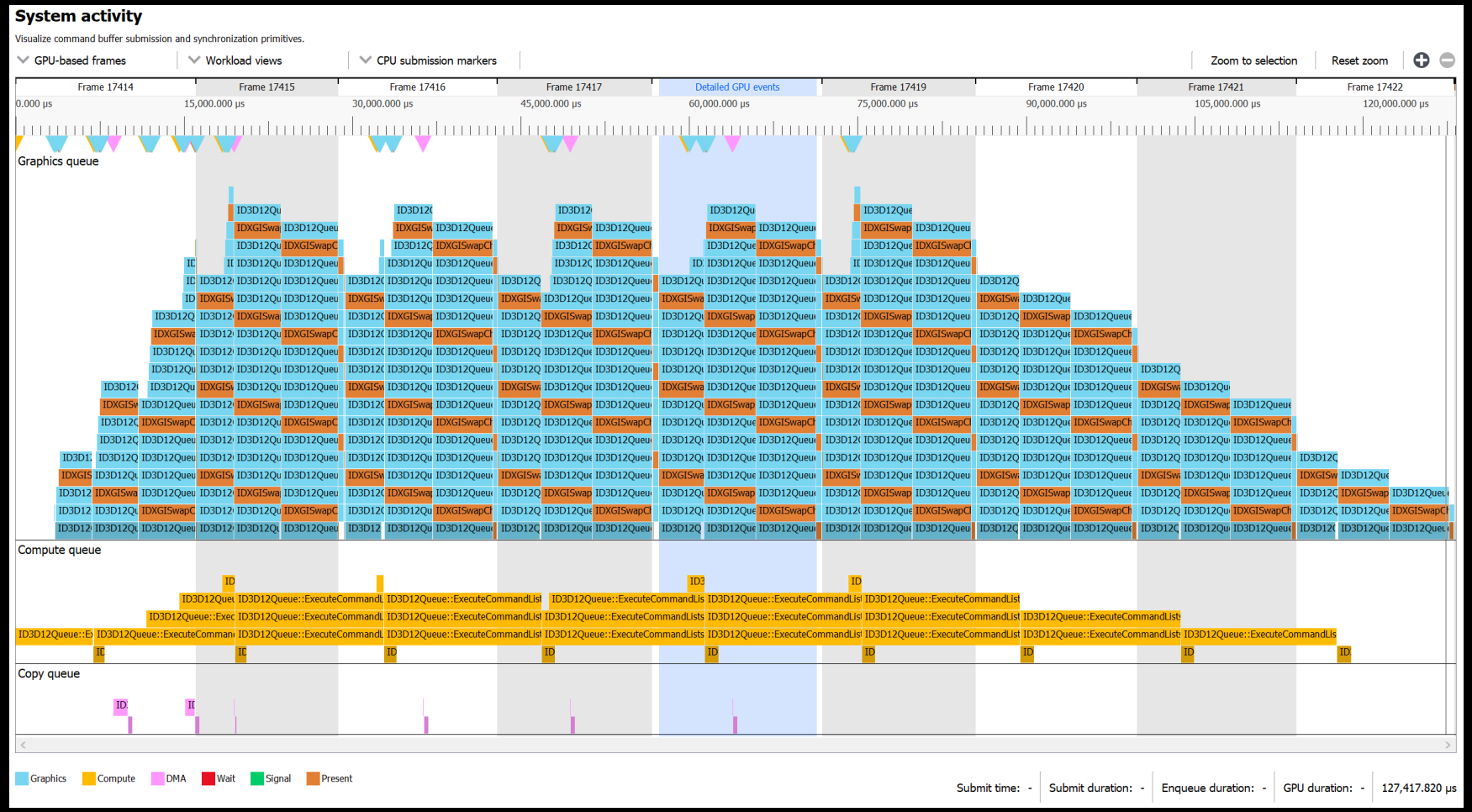
Events

- Wavefront occupancy
- Event timing
- Pipeline state

RenderDoc interop

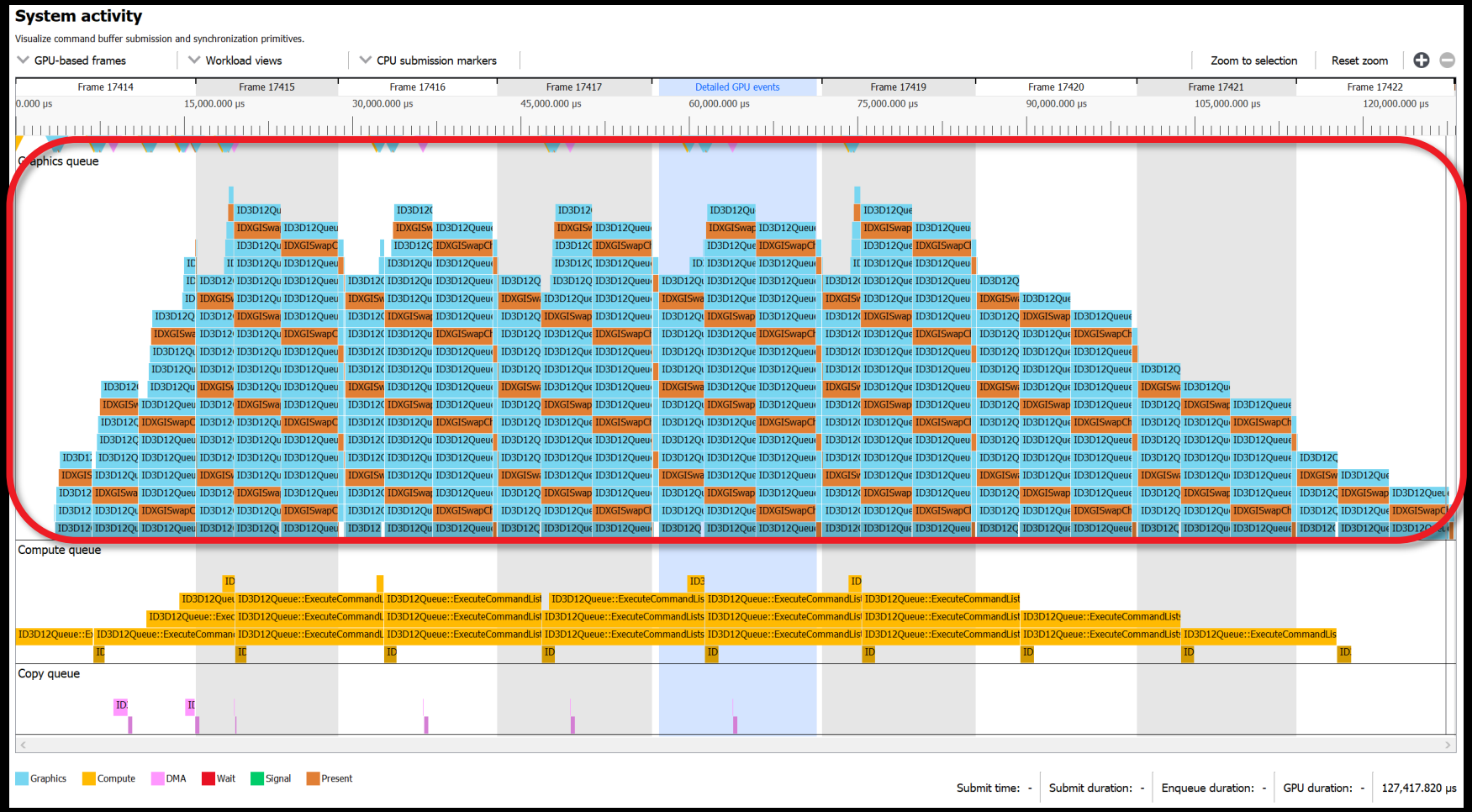


RGP 1.2 - Frame summary



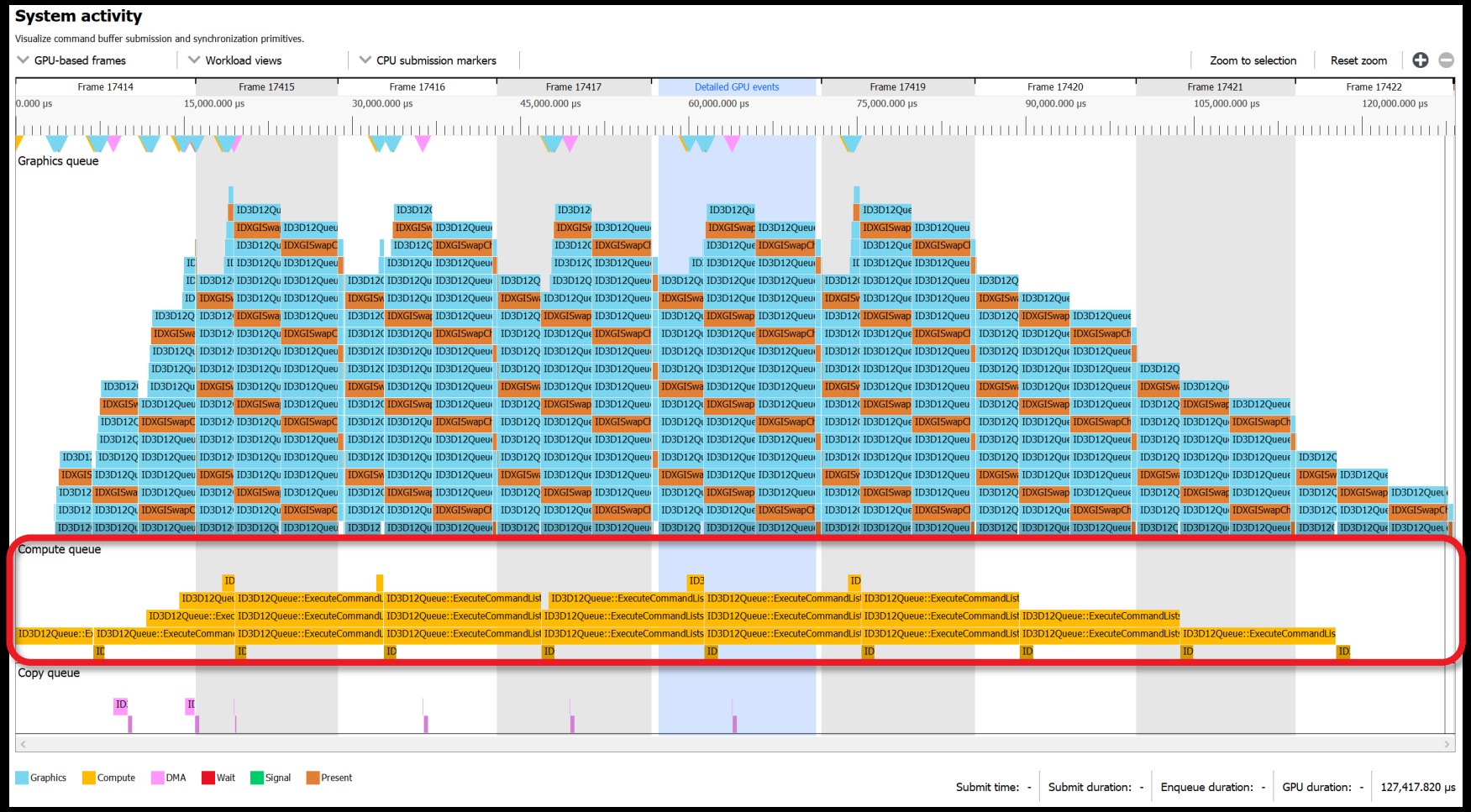
Visualize command buffer submission, queue parallelism, and synchronization

RGP 1.2 - Frame summary



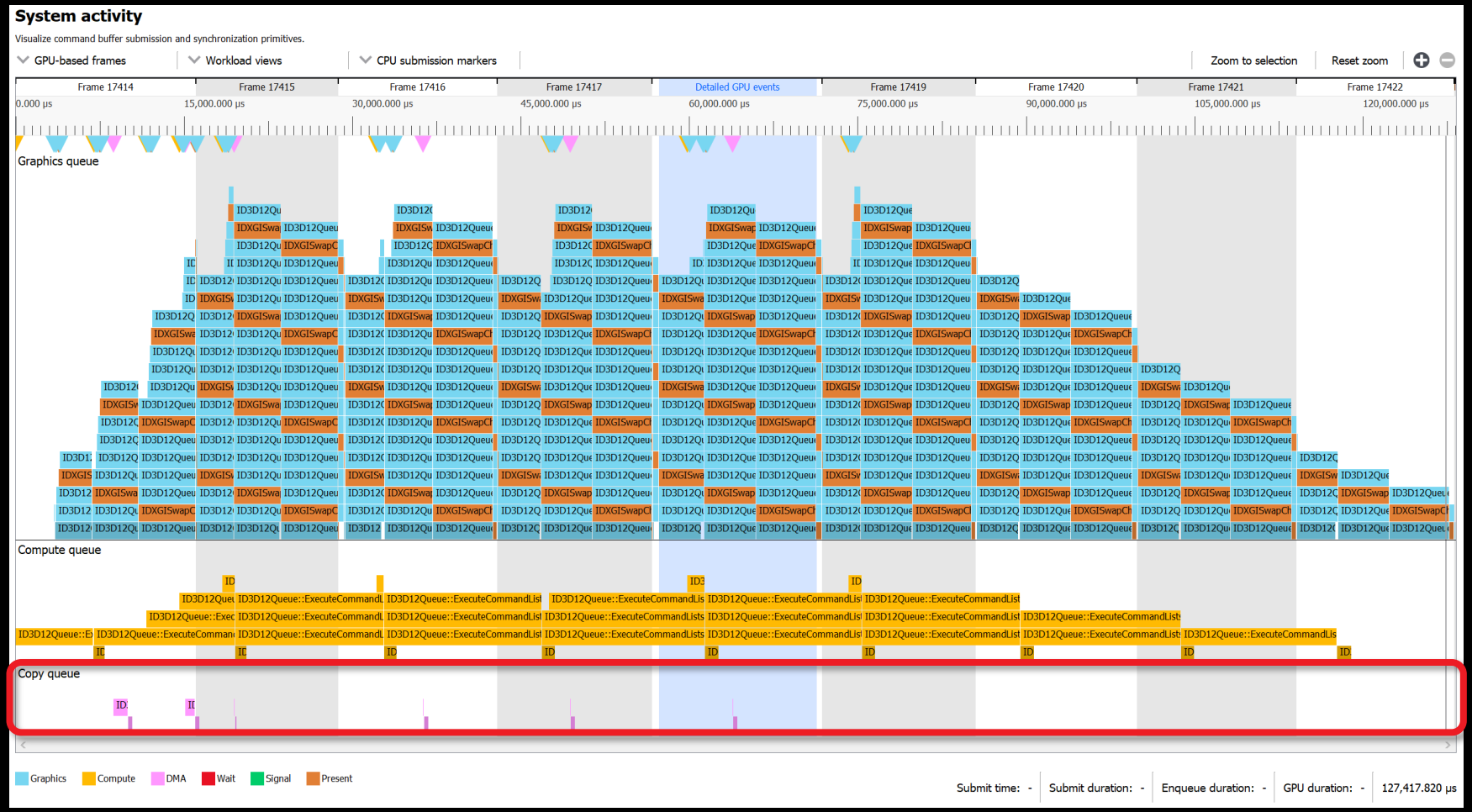
Visualize command buffer submission, queue parallelism, and synchronization

RGP 1.2 - Frame summary



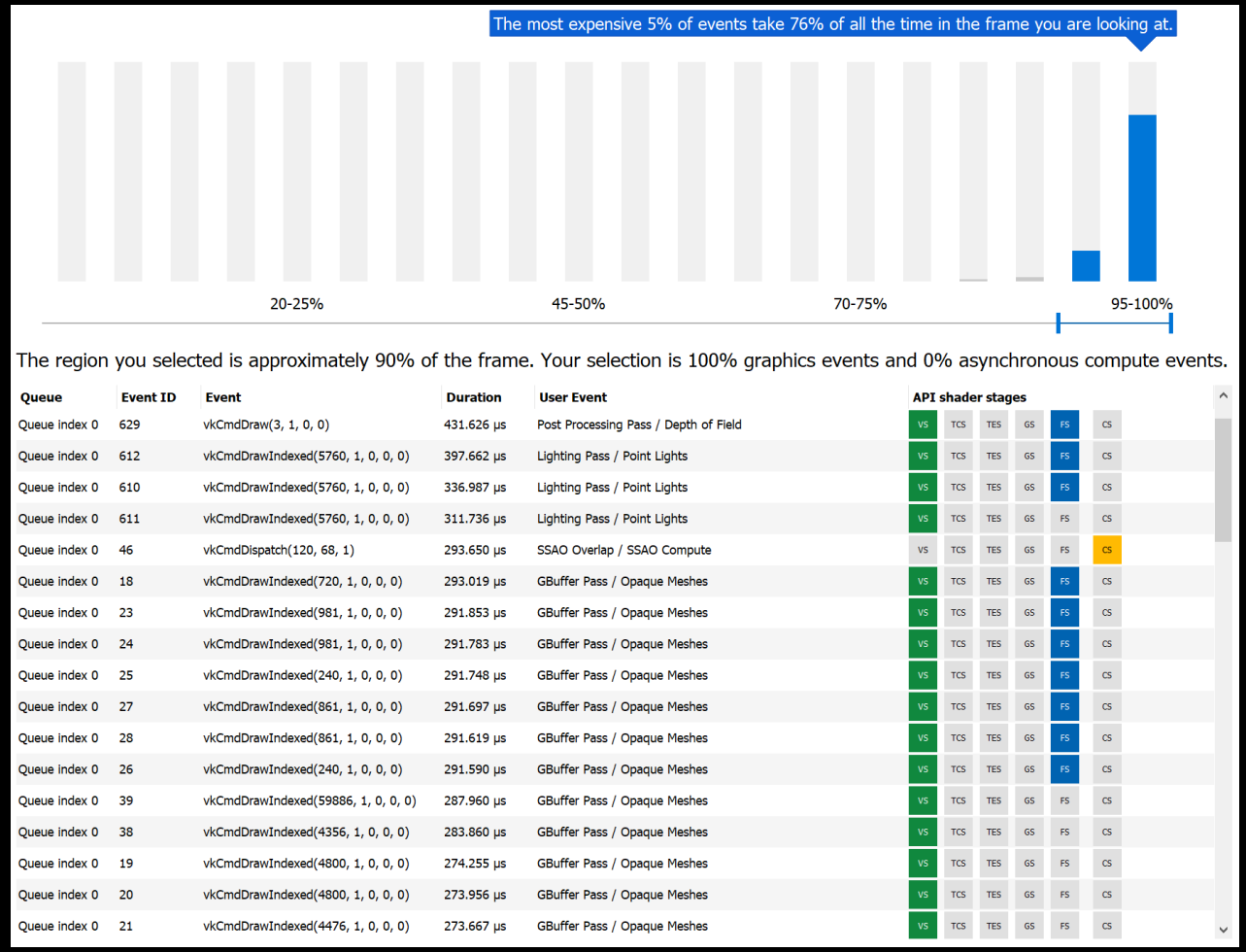
Visualize command buffer submission, queue parallelism, and synchronization

RGP 1.2 - Frame summary



Visualize command buffer submission, queue parallelism, and synchronization

RGP 1.2 - Most expensive events

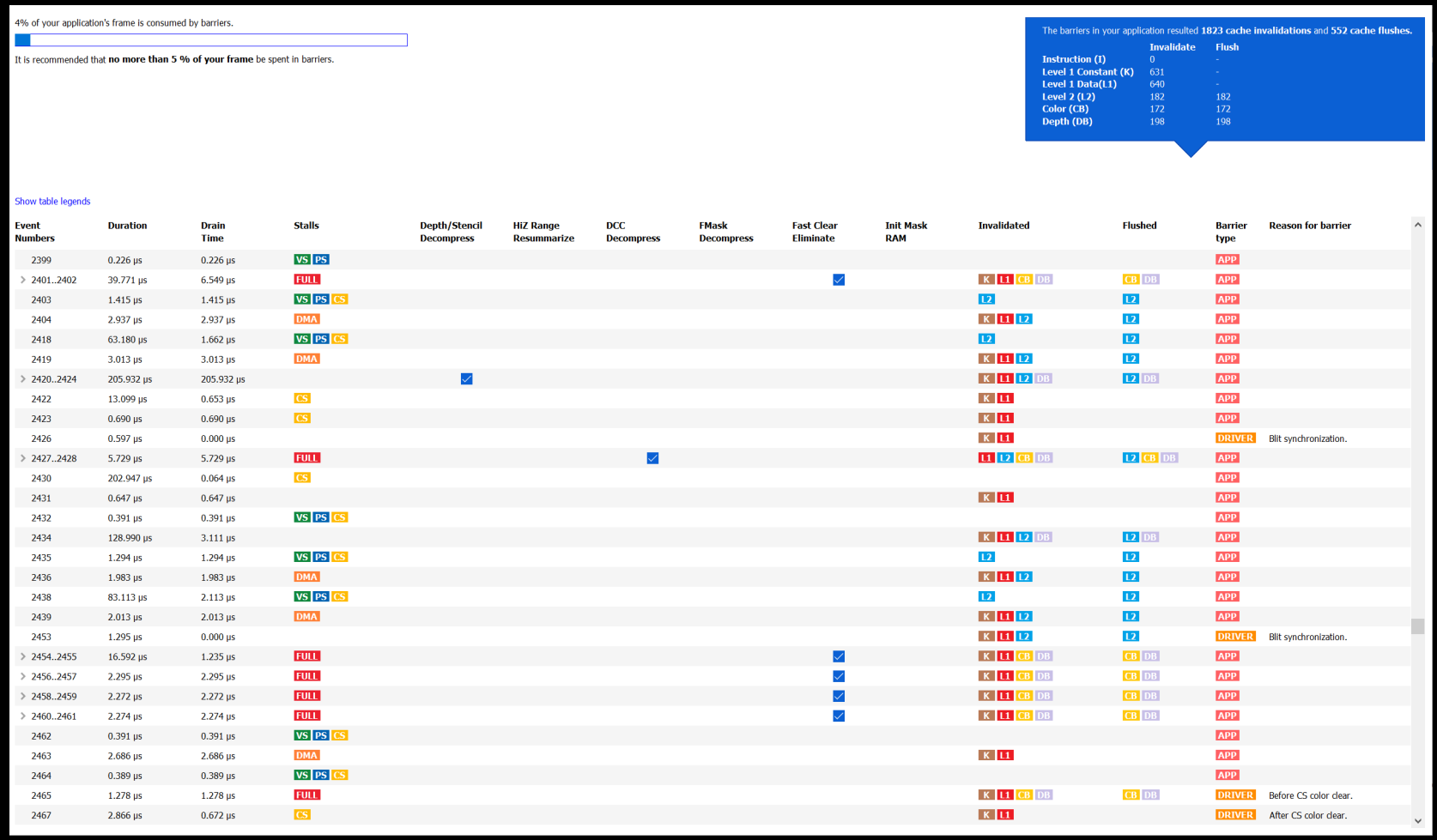


The region you selected is approximately 90% of the frame. Your selection is 100% graphics events and 0% asynchronous compute events.

Queue	Event ID	Event	Duration	User Event	API shader stages
Queue index 0	629	vkCmdDraw(3, 1, 0, 0)	431.626 µs	Post Processing Pass / Depth of Field	VS TCS TES GS FS CS
Queue index 0	612	vkCmdDrawIndexed(5760, 1, 0, 0, 0)	397.662 µs	Lighting Pass / Point Lights	VS TCS TES GS FS CS
Queue index 0	610	vkCmdDrawIndexed(5760, 1, 0, 0, 0)	336.987 µs	Lighting Pass / Point Lights	VS TCS TES GS FS CS
Queue index 0	611	vkCmdDrawIndexed(5760, 1, 0, 0, 0)	311.736 µs	Lighting Pass / Point Lights	VS TCS TES GS FS CS
Queue index 0	46	vkCmdDispatch(120, 68, 1)	293.650 µs	SSAO Overlap / SSAO Compute	VS TCS TES GS FS CS
Queue index 0	18	vkCmdDrawIndexed(720, 1, 0, 0, 0)	293.019 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	23	vkCmdDrawIndexed(981, 1, 0, 0, 0)	291.853 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	24	vkCmdDrawIndexed(981, 1, 0, 0, 0)	291.783 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	25	vkCmdDrawIndexed(240, 1, 0, 0, 0)	291.748 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	27	vkCmdDrawIndexed(861, 1, 0, 0, 0)	291.697 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	28	vkCmdDrawIndexed(861, 1, 0, 0, 0)	291.619 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	26	vkCmdDrawIndexed(240, 1, 0, 0, 0)	291.590 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	39	vkCmdDrawIndexed(59886, 1, 0, 0, 0)	287.960 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	38	vkCmdDrawIndexed(4356, 1, 0, 0, 0)	283.860 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	19	vkCmdDrawIndexed(4800, 1, 0, 0, 0)	274.255 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	20	vkCmdDrawIndexed(4800, 1, 0, 0, 0)	273.956 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS
Queue index 0	21	vkCmdDrawIndexed(4476, 1, 0, 0, 0)	273.667 µs	GBuffer Pass / Opaque Meshes	VS TCS TES GS FS CS

Pinpoint optimization candidates

RGP 1.2 - Barriers



4% of your application's frame is consumed by barriers.

It is recommended that **no more than 5 % of your frame** be spent in barriers.

The barriers in your application resulted **1823 cache invalidations** and **552 cache flushes.**

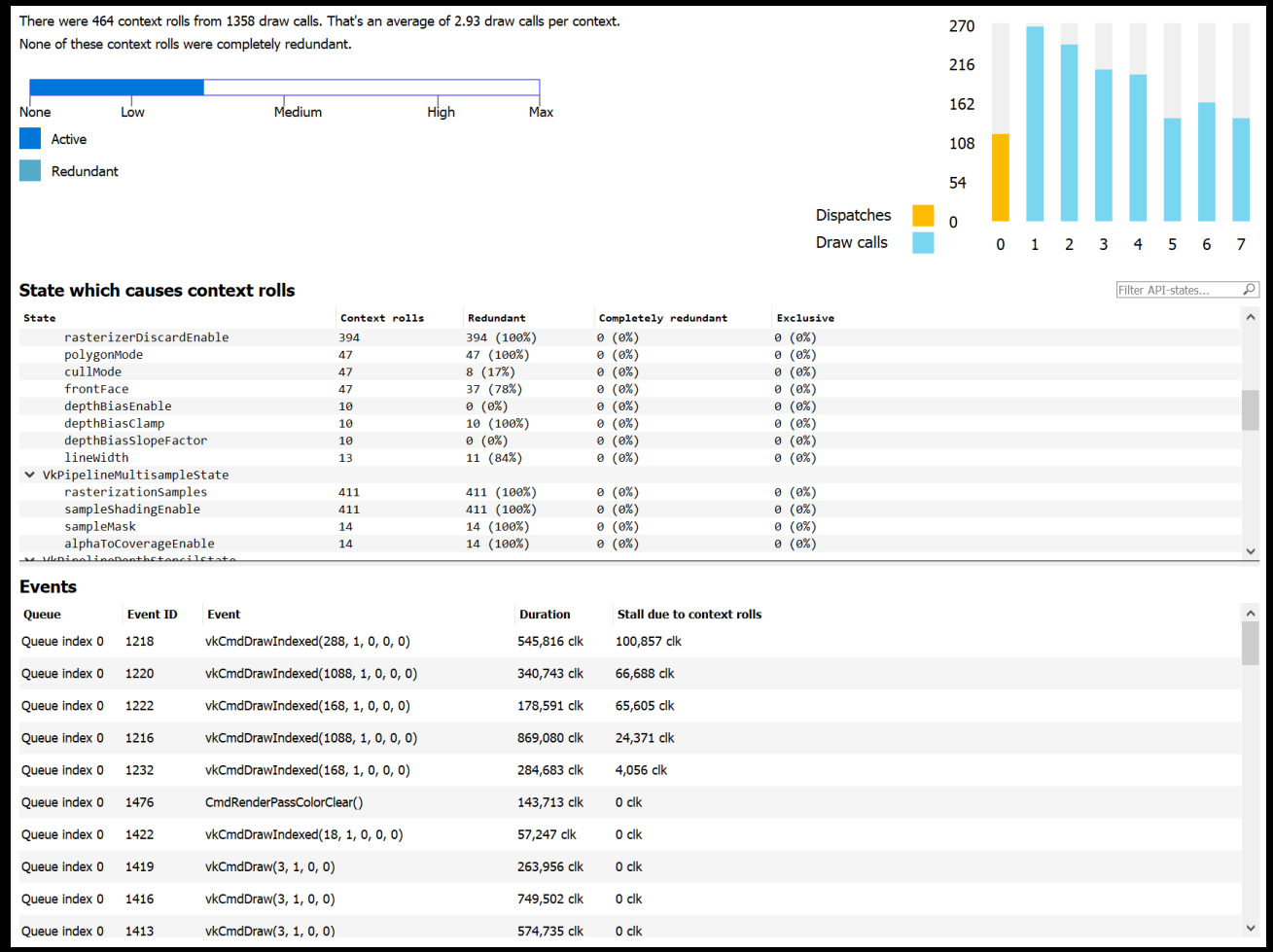
	Invalidate	Flush
Instruction (I)	0	-
Level 1 Constant (K)	631	-
Level 1 Data(L1)	640	-
Level 2 (L2)	182	182
Color (CB)	172	172
Depth (DB)	198	198

Show table legends

Event Numbers	Duration	Drain Time	Stalls	Depth/Stencil Decompress	HiZ Range Resummarize	DCC Decompress	FMask Decompress	Fast Clear Eliminate	Init Mask RAM	Invalidated	Flushed	Barrier type	Reason for barrier
2399	0.226 µs	0.226 µs	VS PS									APP	
2401..2402	39.771 µs	6.549 µs	FULL					✓		K L1 CB DB	CB DB	APP	
2403	1.415 µs	1.415 µs	VS PS CS							L2	L2	APP	
2404	2.937 µs	2.937 µs	DMA							K L1 L2	L2	APP	
2418	63.180 µs	1.662 µs	VS PS CS							L2	L2	APP	
2419	3.013 µs	3.013 µs	DMA							K L1 L2	L2	APP	
2420..2424	205.932 µs	205.932 µs		✓						K L1 L2 DB	L2 DB	APP	
2422	13.099 µs	0.653 µs	CS							K L1		APP	
2423	0.690 µs	0.690 µs	CS							K L1		APP	
2426	0.597 µs	0.000 µs								K L1		DRIVER	Blit synchronization.
2427..2428	5.729 µs	5.729 µs	FULL			✓				L1 L2 CB DB	L2 CB DB	APP	
2430	202.947 µs	0.064 µs	CS									APP	
2431	0.647 µs	0.647 µs								K L1		APP	
2432	0.391 µs	0.391 µs	VS PS CS									APP	
2434	128.990 µs	3.111 µs								K L1 L2 DB	L2 DB	APP	
2435	1.294 µs	1.294 µs	VS PS CS							L2	L2	APP	
2436	1.983 µs	1.983 µs	DMA							K L1 L2	L2	APP	
2438	83.113 µs	2.113 µs	VS PS CS							L2	L2	APP	
2439	2.013 µs	2.013 µs	DMA							K L1 L2	L2	APP	
2453	1.295 µs	0.000 µs								K L1 L2	L2	DRIVER	Blit synchronization.
2454..2455	16.592 µs	1.235 µs	FULL					✓		K L1 CB DB	CB DB	APP	
2456..2457	2.295 µs	2.295 µs	FULL					✓		K L1 CB DB	CB DB	APP	
2458..2459	2.272 µs	2.272 µs	FULL					✓		K L1 CB DB	CB DB	APP	
2460..2461	2.274 µs	2.274 µs	FULL					✓		K L1 CB DB	CB DB	APP	
2462	0.391 µs	0.391 µs	VS PS CS									APP	
2463	2.686 µs	2.686 µs	DMA							K L1		APP	
2464	0.389 µs	0.389 µs	VS PS CS									APP	
2465	1.278 µs	1.278 µs	FULL							K L1 CB DB	CB DB	DRIVER	Before CS color clear.
2467	2.866 µs	0.672 µs	CS							K L1		DRIVER	After CS color clear.

Spot potential pipeline bubbles

RGP 1.2 - Context rolls



Find redundant state changes

RGP 1.2 - Wavefront occupancy

See how busy the GPU is

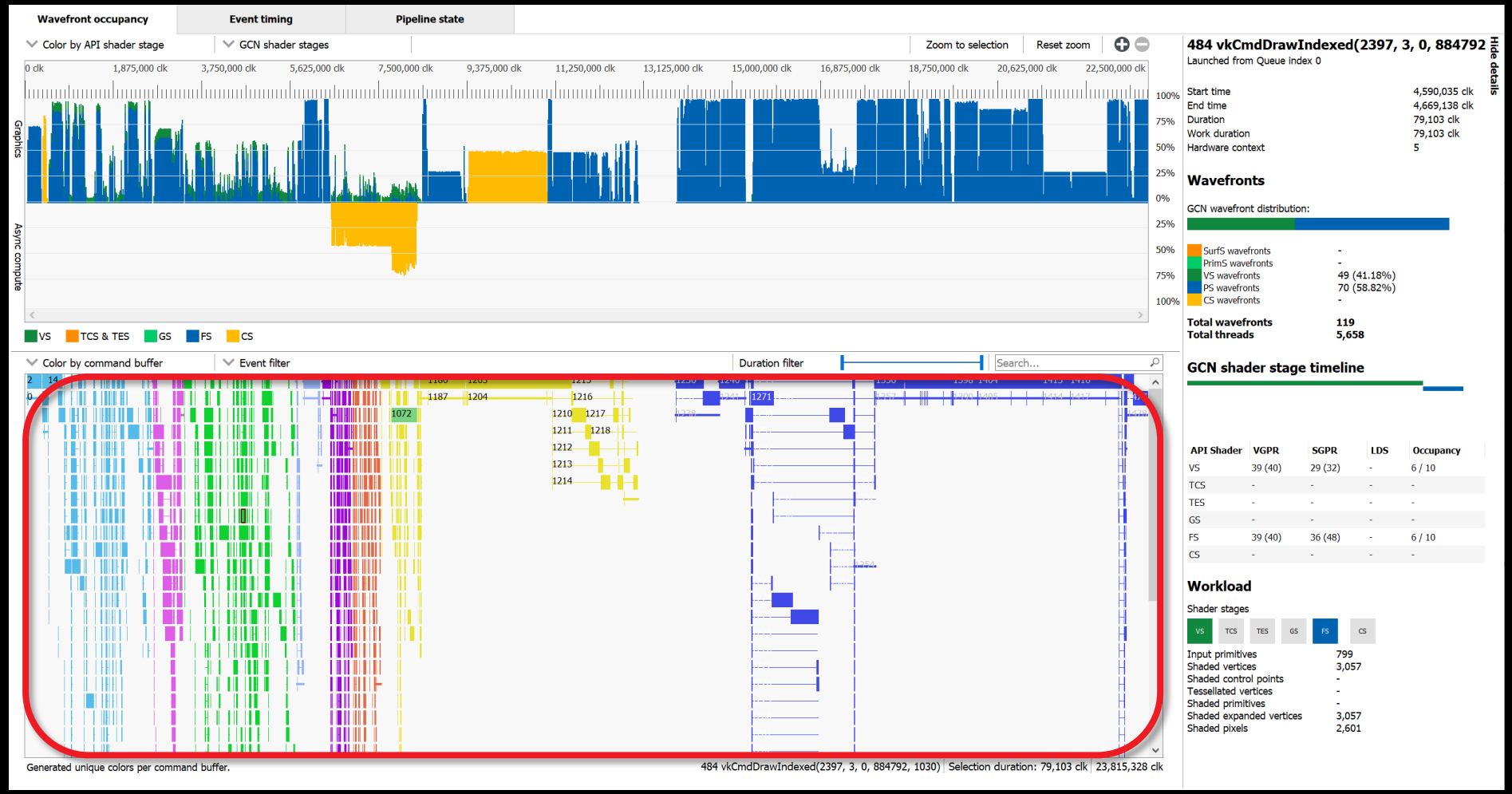
RGP 1.2 - Wavefront occupancy

See how busy the GPU is

RGP 1.2 - Wavefront occupancy

See how busy the GPU is

RGP 1.2 - Wavefront occupancy

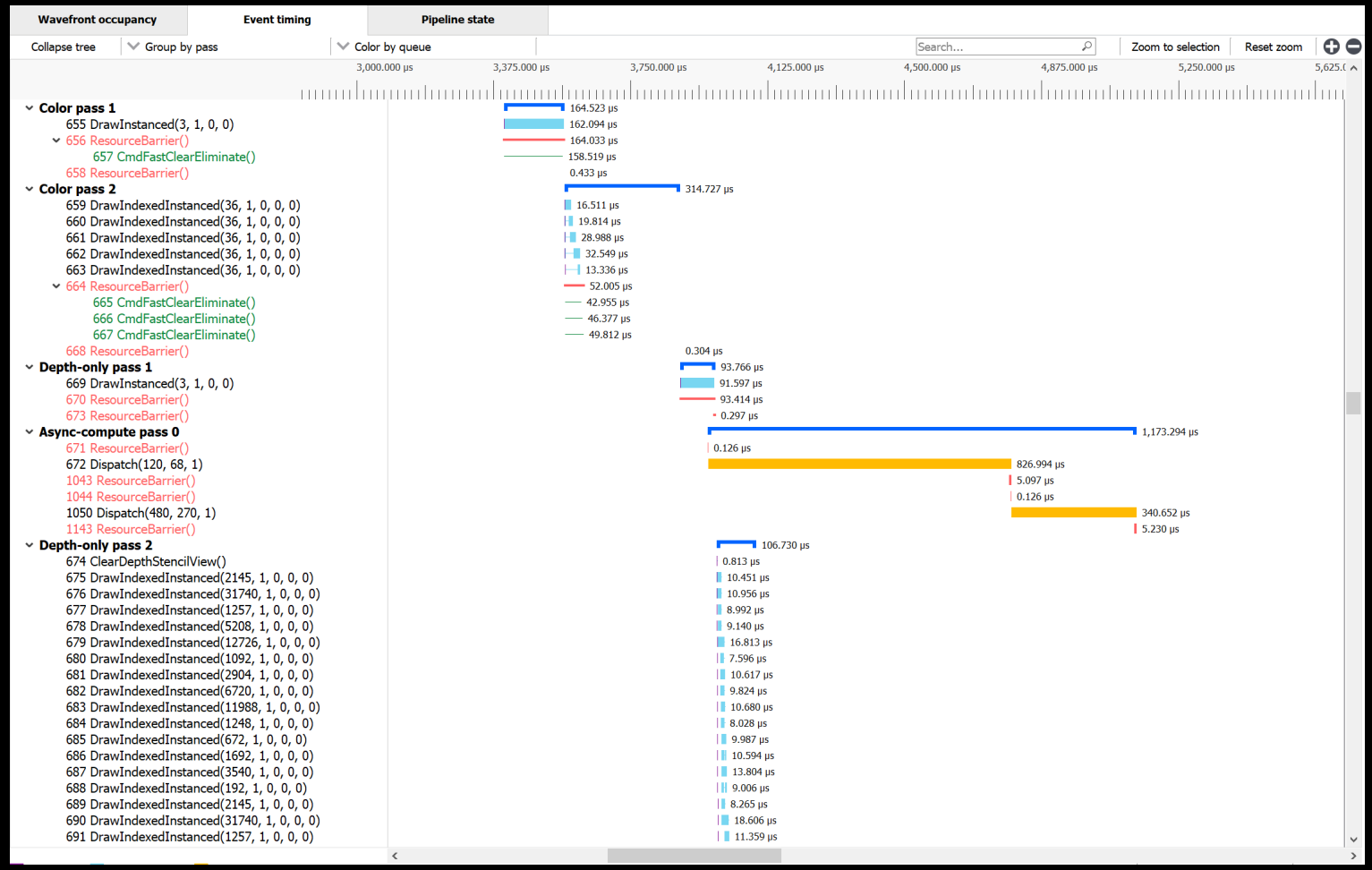


See how busy the GPU is

RGP 1.2 - Wavefront occupancy

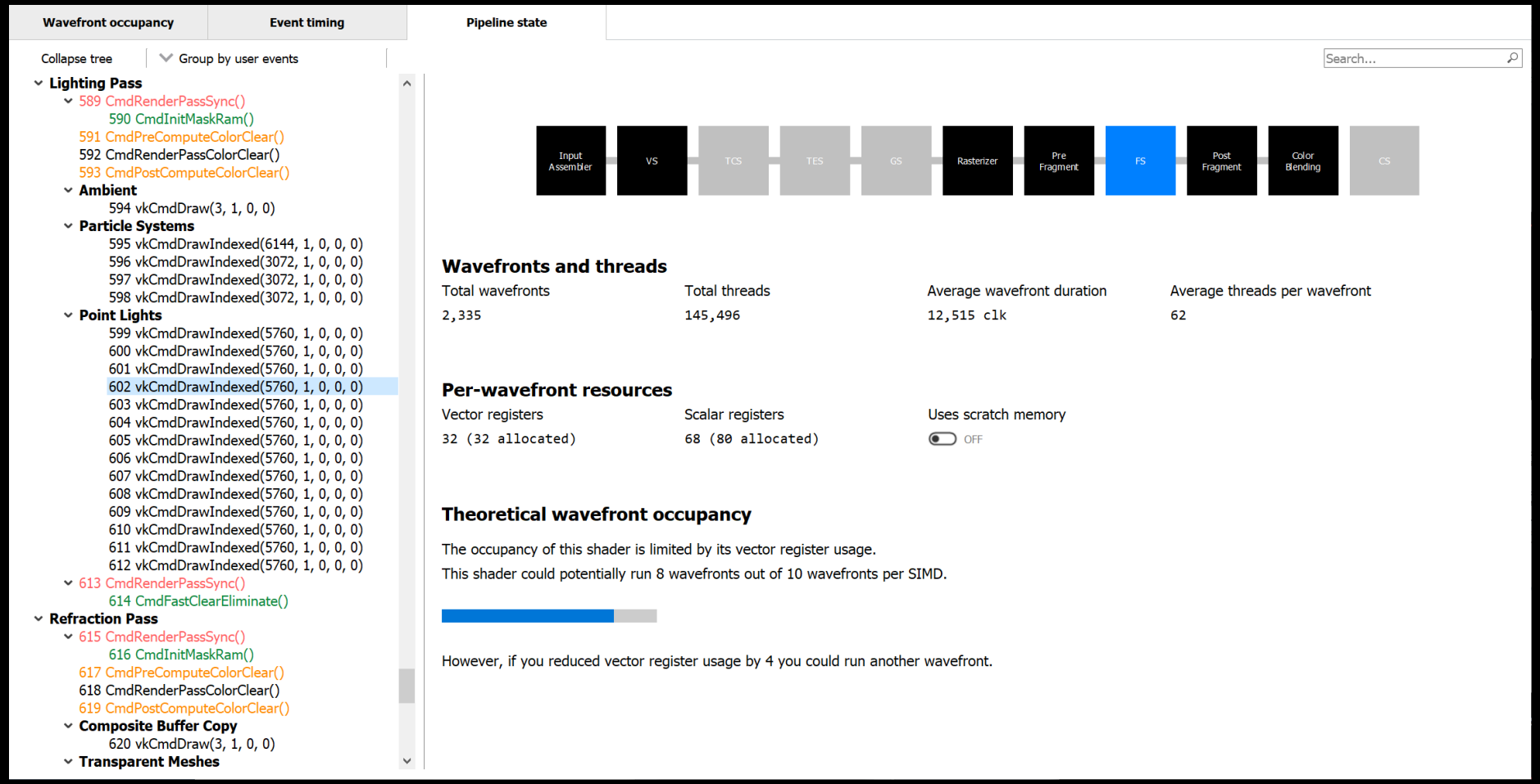
See how busy the GPU is

RGP 1.2 - Event timing



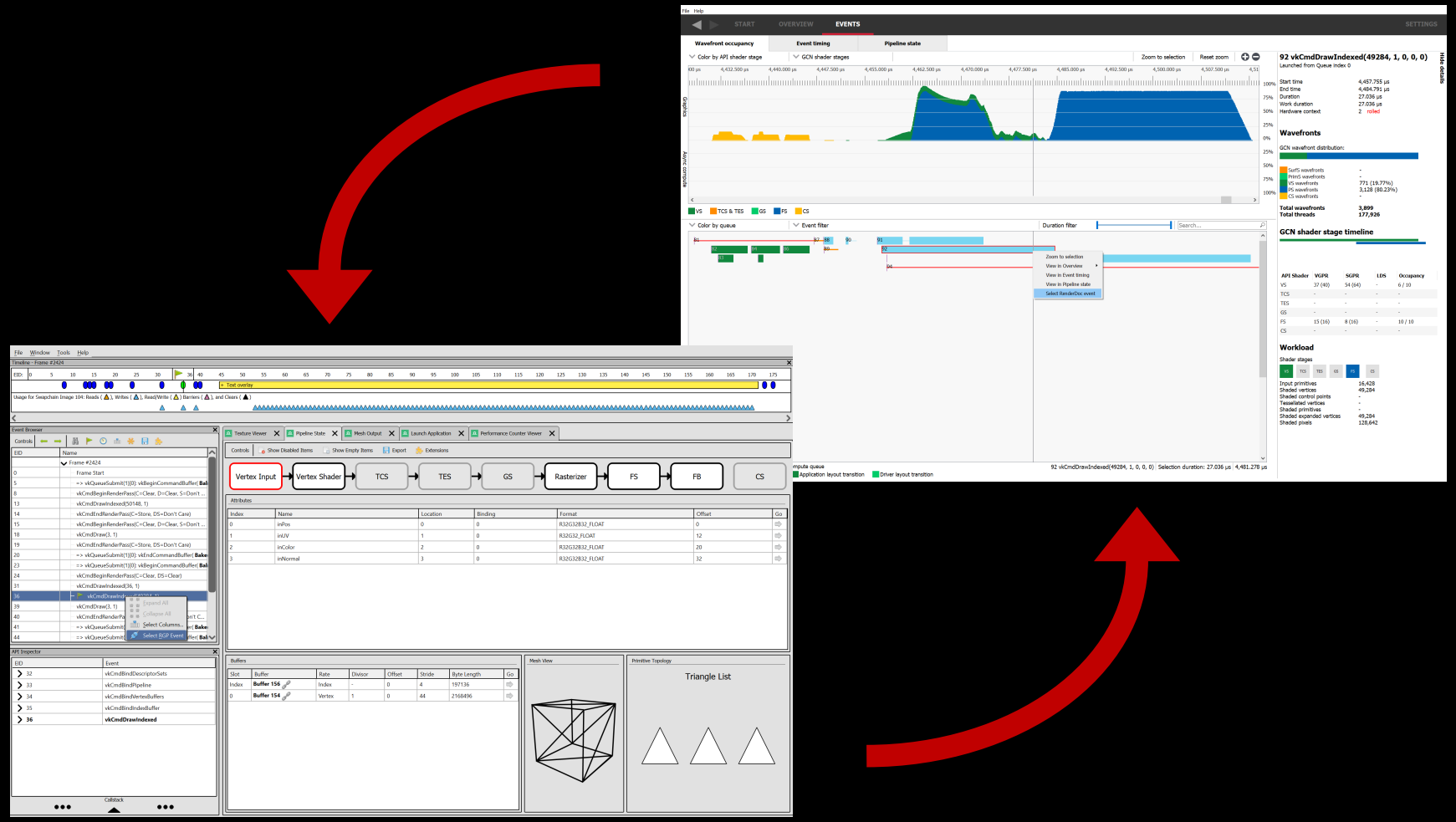
Understand the cost of GPU work in your frame

RGP 1.2 - Pipeline state



Expose pipeline state and GPR consumption

RGP 1.2 - RenderDoc ←→ RGP interop



Understand correlation between rendered scene ←→ GPU work

Cool, but I want more!

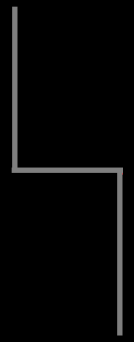
What about

- Wavefront shader execution?
- How do my shaders translate to GCN ISA?
- Can you give me information about my render targets?
- Enhanced user marker visualization?
- What if I'm a compute-only developer?

?

So what's new?

GDC 2018



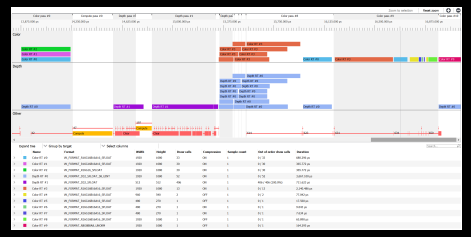
RGP 1.2
RenderDoc interop

So what's new?

GDC 2018

RGP 1.2
RenderDoc interop

RGP 1.3



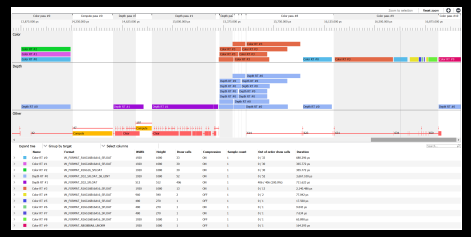
Render & depth targets

So what's new?

GDC 2018

RGP 1.2
RenderDoc interop

RGP 1.3



Render & depth targets

RGP 1.4

OpenCL™ support

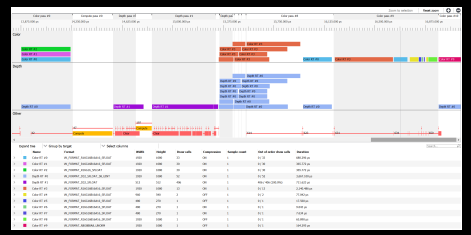
So what's new?

GDC 2018

GDC 2019

RGP 1.2
RenderDoc interop

RGP 1.3

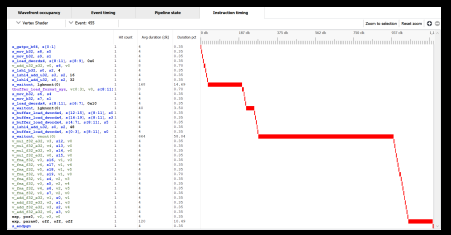


Render & depth targets

RGP 1.4

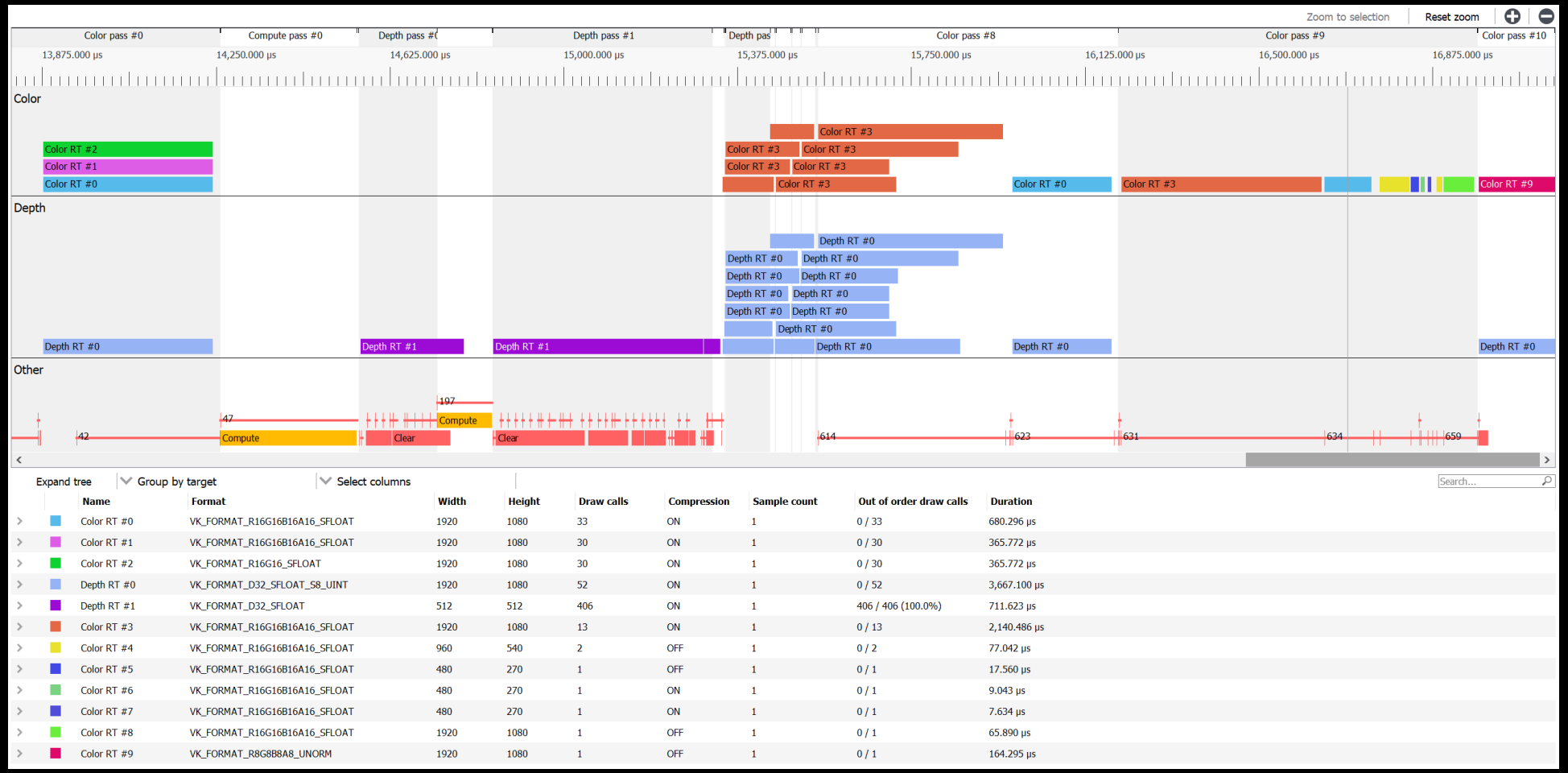
OpenCL™ support

RGP 1.5.x



Instruction timing
Shader ISA
User marker display

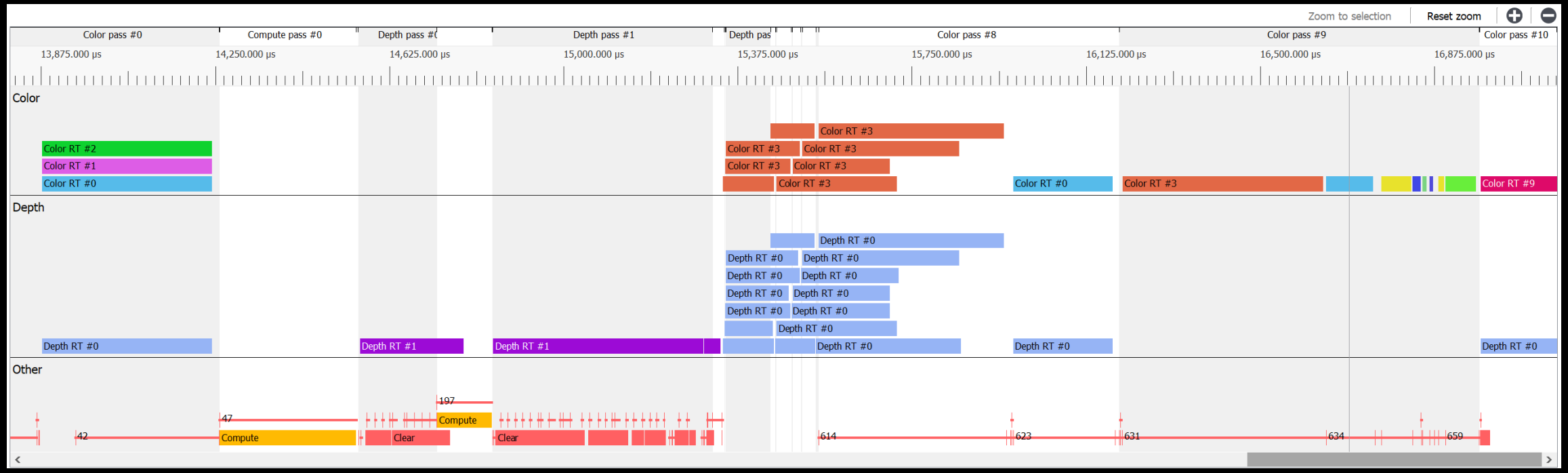
RGP 1.3 - Render & depth targets



	Name	Format	Width	Height	Draw calls	Compression	Sample count	Out of order draw calls	Duration
	Color RT #0	VK_FORMAT_R16G16B16A16_SFLOAT	1920	1080	33	ON	1	0 / 33	680.296 µs
	Color RT #1	VK_FORMAT_R16G16B16A16_SFLOAT	1920	1080	30	ON	1	0 / 30	365.772 µs
	Color RT #2	VK_FORMAT_R16G16_SFLOAT	1920	1080	30	ON	1	0 / 30	365.772 µs
	Depth RT #0	VK_FORMAT_D32_SFLOAT_S8_UINT	1920	1080	52	ON	1	0 / 52	3,667.100 µs
	Depth RT #1	VK_FORMAT_D32_SFLOAT	512	512	406	ON	1	406 / 406 (100.0%)	711.623 µs
	Color RT #3	VK_FORMAT_R16G16B16A16_SFLOAT	1920	1080	13	ON	1	0 / 13	2,140.486 µs
	Color RT #4	VK_FORMAT_R16G16B16A16_SFLOAT	960	540	2	OFF	1	0 / 2	77.042 µs
	Color RT #5	VK_FORMAT_R16G16B16A16_SFLOAT	480	270	1	OFF	1	0 / 1	17.560 µs
	Color RT #6	VK_FORMAT_R16G16B16A16_SFLOAT	480	270	1	ON	1	0 / 1	9.043 µs
	Color RT #7	VK_FORMAT_R16G16B16A16_SFLOAT	480	270	1	ON	1	0 / 1	7.634 µs
	Color RT #8	VK_FORMAT_R16G16B16A16_SFLOAT	1920	1080	1	OFF	1	0 / 1	65.890 µs
	Color RT #9	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	1	OFF	1	0 / 1	164.295 µs

Understand all your render targets

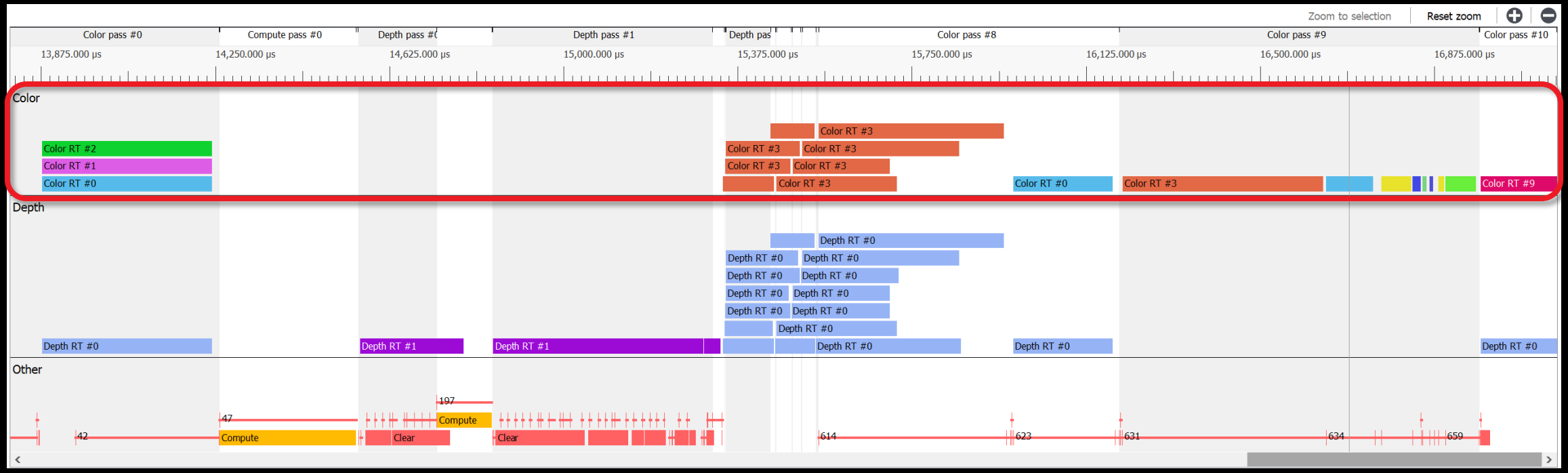
RGP 1.3 - Render & depth targets



Interactive timeline

- Three sections - color, depth, other
- Displays when gfx/compute passes begin/end
- Hooked up to other RGP panes via context menu

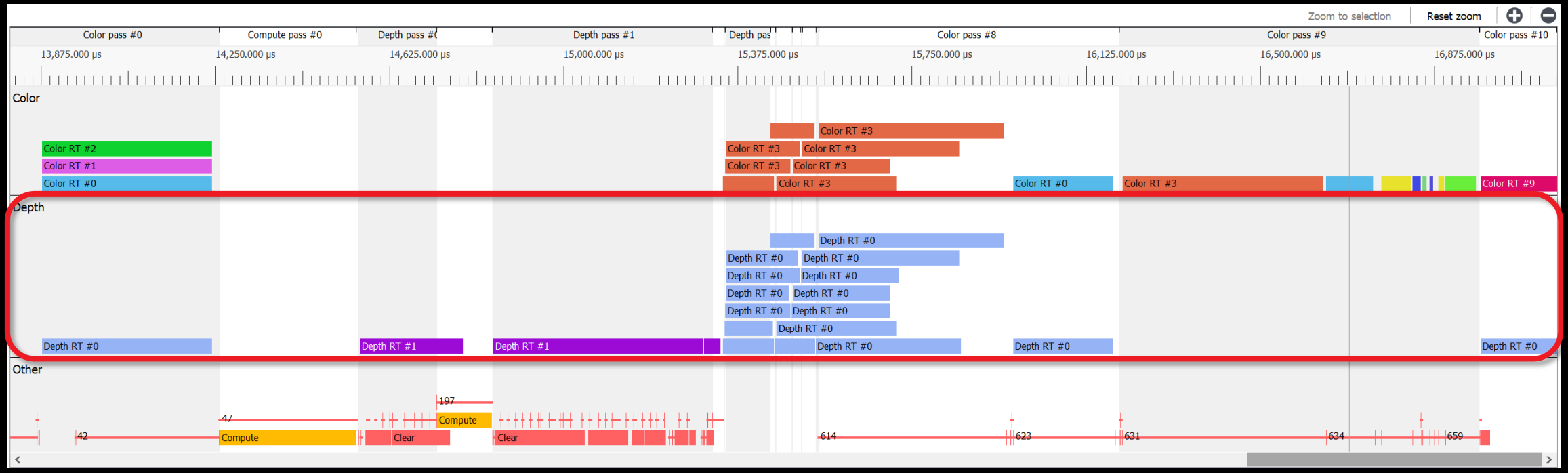
RGP 1.3 - Render & depth targets



Interactive timeline

- Three sections - color, depth, other
- Displays when gfx/compute passes begin/end
- Hooked up to other RGP panes via context menu

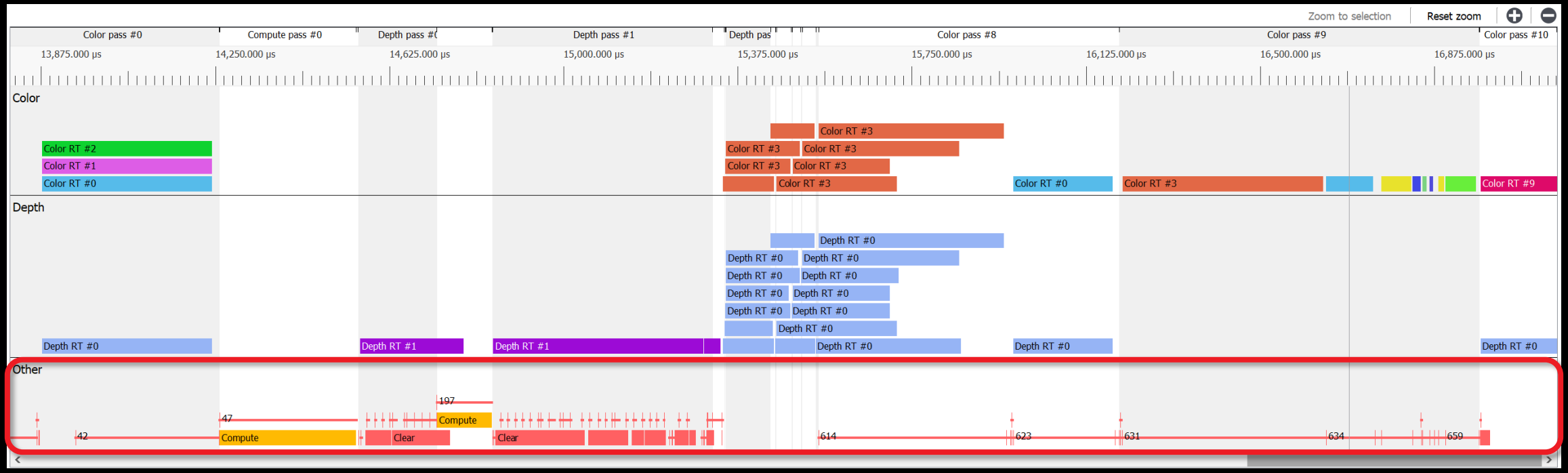
RGP 1.3 - Render & depth targets



Interactive timeline

- Three sections - color, depth, other
- Displays when gfx/compute passes begin/end
- Hooked up to other RGP panes via context menu

RGP 1.3 - Render & depth targets



Interactive timeline

- Three sections - color, depth, other
- Displays when gfx/compute passes begin/end
- Hooked up to other RGP panes via context menu

RGP 1.3 - Render & depth targets

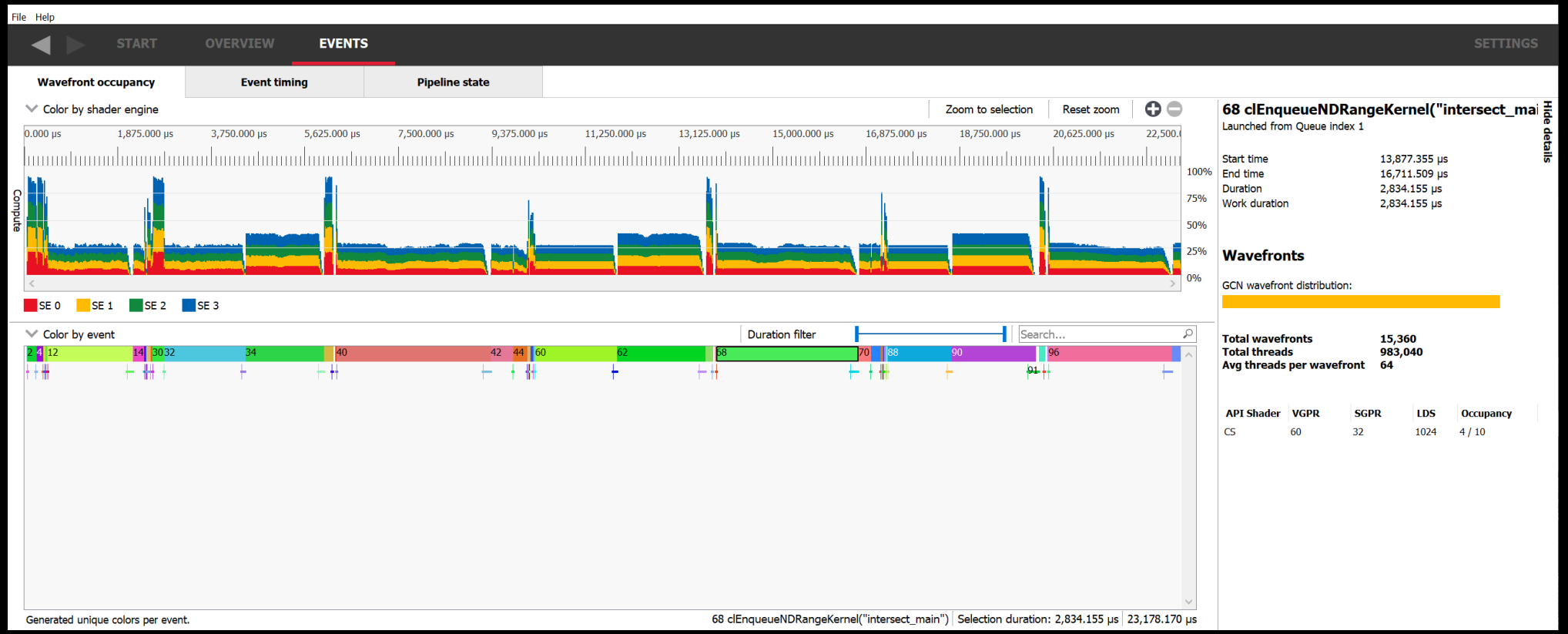
Expand tree | Group by target | Select columns

Name	Format	Width	Height	Draw calls	Compression	Sample count	Out of order draw calls	Duration
Color RT #0	VK_FORMAT_R16G16B16A16_SFLOAT	1920	1080	33	ON	1	0 / 33	680.296 µs
Color RT #1	VK_FORMAT_R16G16B16A16_SFLOAT	1920	1080	30	ON	1	0 / 30	365.772 µs
Color RT #2	VK_FORMAT_R16G16_SFLOAT	1920	1080	30	ON	1	0 / 30	365.772 µs
Depth RT #0	VK_FORMAT_D32_SFLOAT_S8_UINT	1920	1080	52	ON	1	0 / 52	3,667.100 µs
Depth RT #1	VK_FORMAT_D32_SFLOAT	512	512	406	ON	1	406 / 406 (100.0%)	711.623 µs
Color RT #3	VK_FORMAT_R16G16B16A16_SFLOAT	1920	1080	13	ON	1	0 / 13	2,140.486 µs
Color RT #4	VK_FORMAT_R16G16B16A16_SFLOAT	960	540	2	OFF	1	0 / 2	77.042 µs
Color RT #5	VK_FORMAT_R16G16B16A16_SFLOAT	480	270	1	OFF	1	0 / 1	17.560 µs
Color RT #6	VK_FORMAT_R16G16B16A16_SFLOAT	480	270	1	ON	1	0 / 1	9.043 µs
Color RT #7	VK_FORMAT_R16G16B16A16_SFLOAT	480	270	1	ON	1	0 / 1	7.634 µs
Color RT #8	VK_FORMAT_R16G16B16A16_SFLOAT	1920	1080	1	OFF	1	0 / 1	65.890 µs
Color RT #9	VK_FORMAT_R8G8B8A8_UNORM	1920	1080	1	OFF	1	0 / 1	164.295 µs

Properties table

- Dimensions and format
- Draw and sample counts
- Duration
- Optimizations

RGP 1.4 - OpenCL™ support



What is it?

- RGP minus the graphics-only stuff

How do I use it?

- Developer panel now detects OpenCL™ applications
- Set number of kernels, and capture!

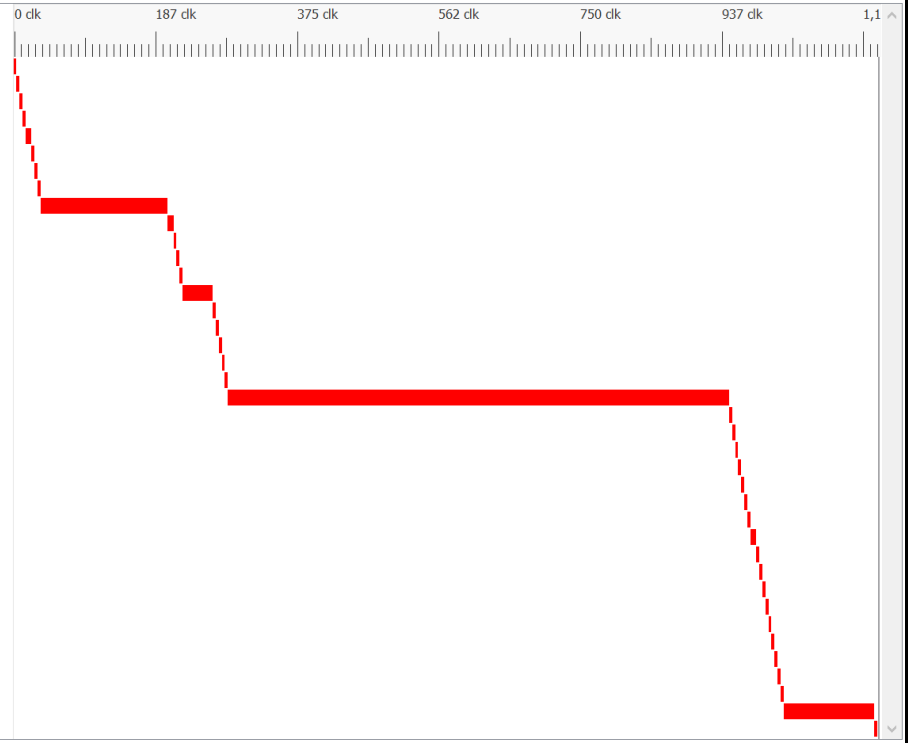
Planned for RGP 1.5.x

Instruction timing

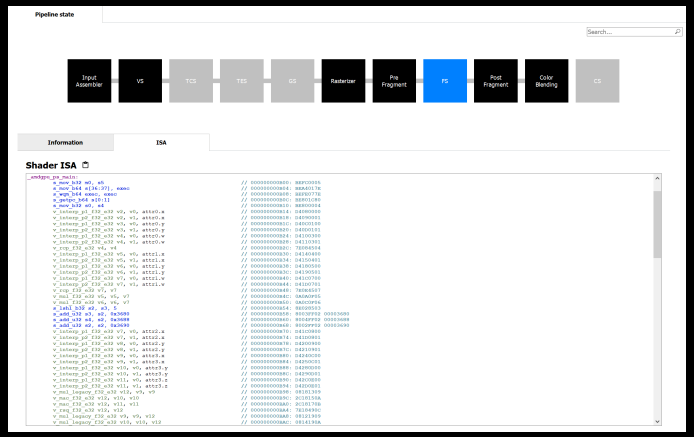
Wavefront occupancy | Event timing | Pipeline state | Instruction timing

Vertex Shader | Event: 455 | Zoom to selection | Reset zoom

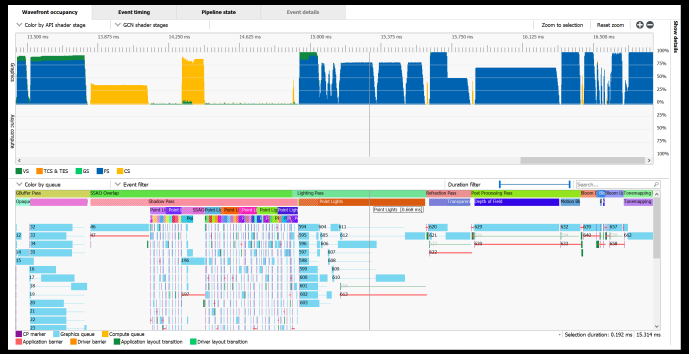
	Hit count	Avg duration (clk)	Duration pct
s_getpc_b64, s[0:1]	1	4	0.35
s_mov_b32, s8, s5	1	4	0.35
s_mov_b32, s9, s1	1	4	0.35
s_load_dwordx4, s[8:11], s[8:9], 0x0	1	4	0.35
v_add_u32_e32, v0, s6, v0	1	8	0.70
s_lshl_b32, s0, s2, 4	1	4	0.35
s_lshl4_add_u32, s3, s2, 16	1	4	0.35
s_lshl4_add_u32, s5, s2, 32	1	4	0.35
s_waitcnt, lgkmcnt(0)	1	168	14.69
tbuffer_load_format_xyz, v[0:3], v0, s[8:11]	1	8	0.70
s_mov_b32, s6, s4	1	4	0.35
s_mov_b32, s7, s1	1	4	0.35
s_load_dwordx4, s[8:11], s[6:7], 0x10	1	4	0.35
s_waitcnt, lgkmcnt(0)	1	40	3.50
s_buffer_load_dwordx4, s[12:15], s[8:11], s0	1	4	0.35
s_buffer_load_dwordx4, s[16:19], s[8:11], s3	1	4	0.35
s_buffer_load_dwordx4, s[4:7], s[8:11], s5	1	4	0.35
s_lshl4_add_u32, s0, s2, 48	1	4	0.35
s_buffer_load_dwordx4, s[0:3], s[8:11], s0	1	4	0.35
s_waitcnt, vmcnt(0)	1	664	58.04
v_mul_f32_e32, v3, s12, v0	1	4	0.35
v_mul_f32_e32, v4, s13, v0	1	4	0.35
v_mul_f32_e32, v5, s14, v0	1	4	0.35
v_mul_f32_e32, v0, s15, v0	1	4	0.35
v_fma_f32, v3, s16, v1, v3	1	4	0.35
v_fma_f32, v4, s17, v1, v4	1	4	0.35
v_fma_f32, v5, s18, v1, v5	1	4	0.35
v_fma_f32, v0, s19, v1, v0	1	8	0.70
v_fma_f32, v1, s4, v2, v3	1	4	0.35
v_fma_f32, v3, s5, v2, v4	1	4	0.35
v_fma_f32, v4, s6, v2, v5	1	4	0.35
v_fma_f32, v0, s7, v2, v0	1	4	0.35
v_add_f32_e32, v1, s0, v1	1	4	0.35
v_add_f32_e32, v2, s1, v3	1	4	0.35
v_add_f32_e32, v3, s2, v4	1	4	0.35
v_add_f32_e32, v0, s3, v0	1	4	0.35
exp, pos0, v2, v3, v0	1	4	0.35
exp, param0, off, off, off	1	120	10.49
s_endpgm	1	4	0.35



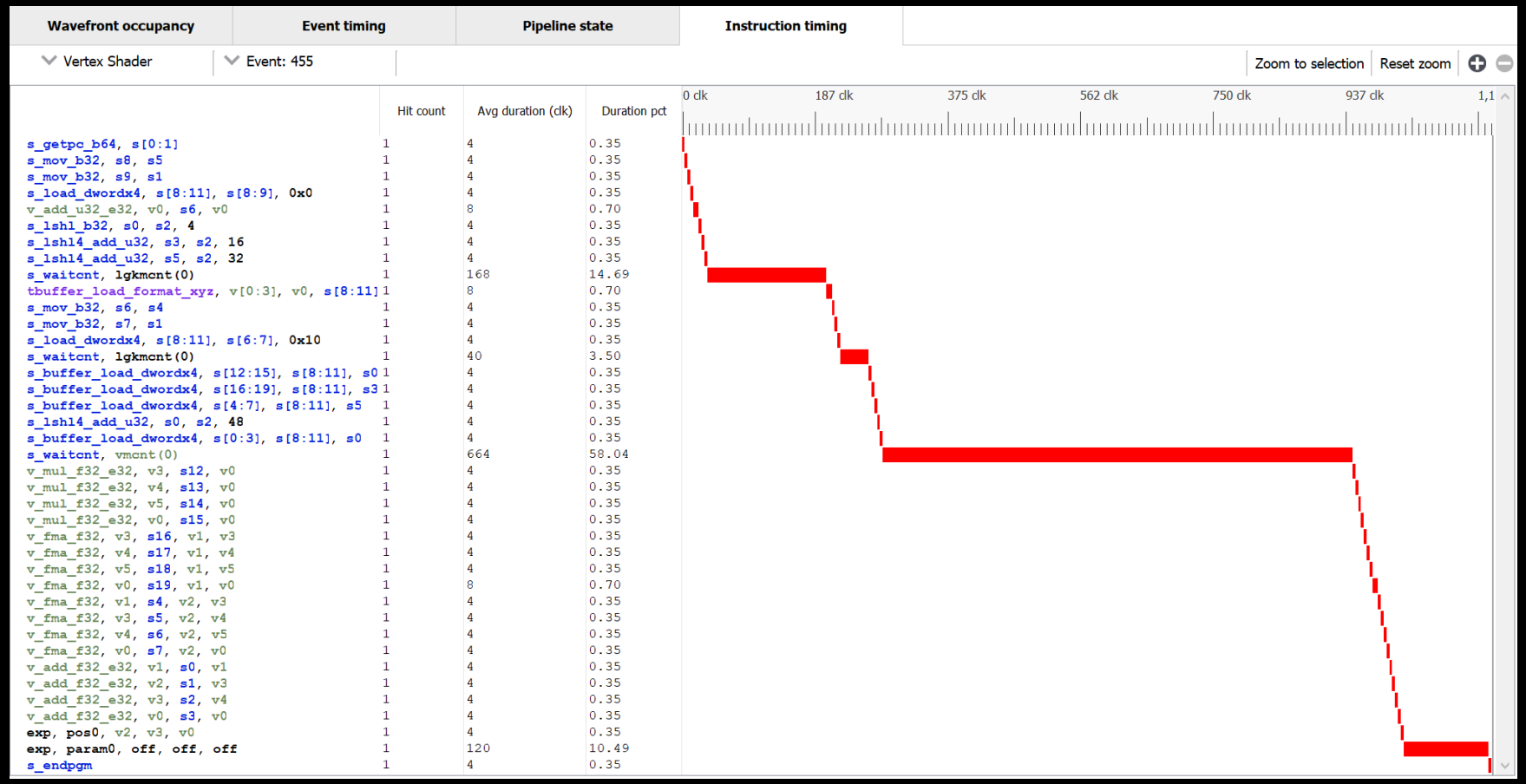
Shader ISA



User marker display



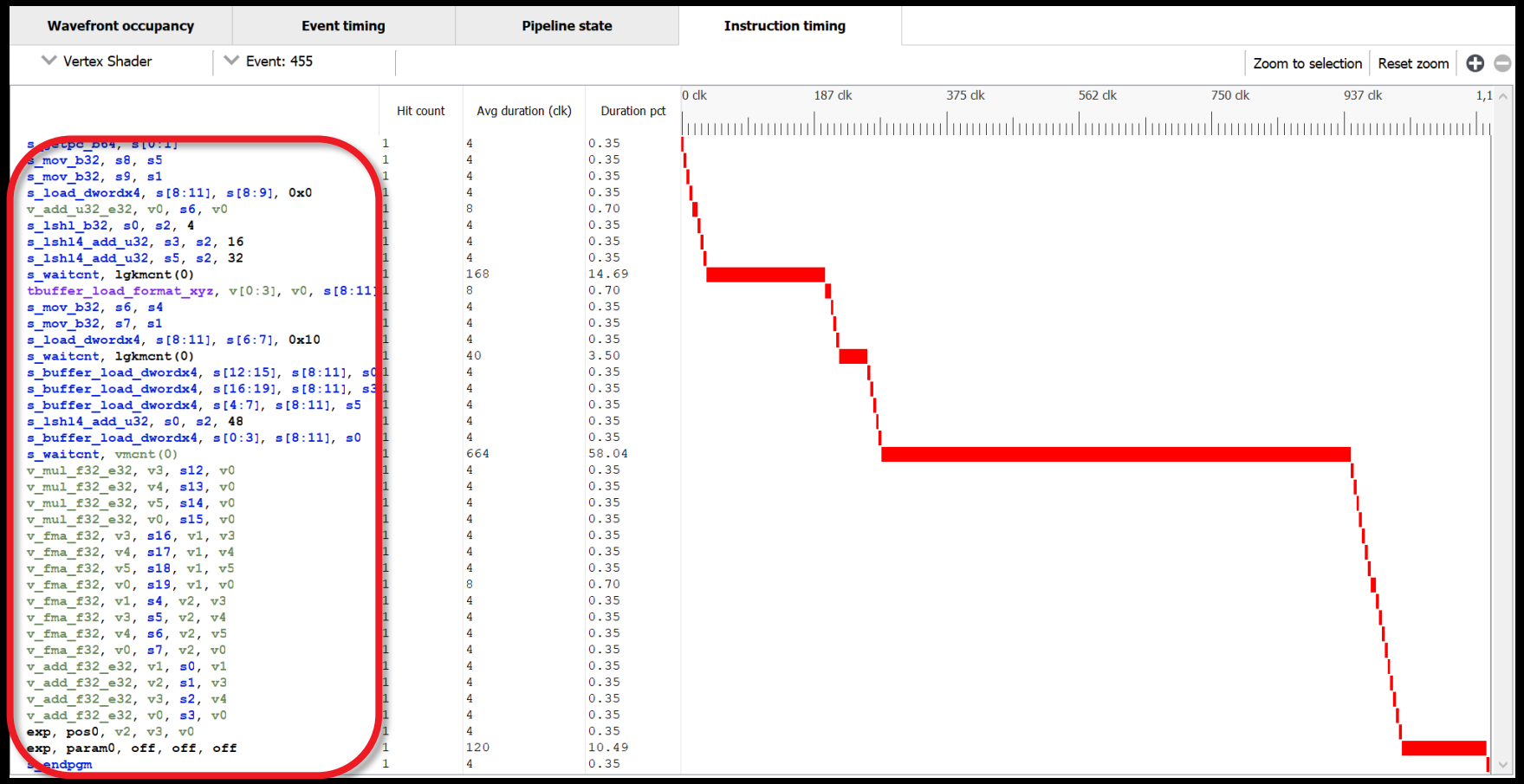
Planned for RGP 1.5.x - Instruction timing



	Hit count	Avg duration (clk)	Duration pct
s_getpc_b64, s[0:1]	1	4	0.35
s_mov_b32, s8, s5	1	4	0.35
s_mov_b32, s9, s1	1	4	0.35
s_load_dwordx4, s[8:11], s[8:9], 0x0	1	4	0.35
v_add_u32_e32, v0, s6, v0	1	8	0.70
s_lshl_b32, s0, s2, 4	1	4	0.35
s_lshl4_add_u32, s3, s2, 16	1	4	0.35
s_lshl4_add_u32, s5, s2, 32	1	4	0.35
s_waitcnt, lgkmcnt(0)	1	168	14.69
tbuffer_load_format_xyz, v[0:3], v0, s[8:11]	1	8	0.70
s_mov_b32, s6, s4	1	4	0.35
s_mov_b32, s7, s1	1	4	0.35
s_load_dwordx4, s[8:11], s[6:7], 0x10	1	4	0.35
s_waitcnt, lgkmcnt(0)	1	40	3.50
s_buffer_load_dwordx4, s[12:15], s[8:11], s0	1	4	0.35
s_buffer_load_dwordx4, s[16:19], s[8:11], s3	1	4	0.35
s_buffer_load_dwordx4, s[4:7], s[8:11], s5	1	4	0.35
s_lshl4_add_u32, s0, s2, 48	1	4	0.35
s_buffer_load_dwordx4, s[0:3], s[8:11], s0	1	4	0.35
s_waitcnt, vmcnt(0)	1	664	58.04
v_mul_f32_e32, v3, s12, v0	1	4	0.35
v_mul_f32_e32, v4, s13, v0	1	4	0.35
v_mul_f32_e32, v5, s14, v0	1	4	0.35
v_mul_f32_e32, v0, s15, v0	1	4	0.35
v_fma_f32, v3, s16, v1, v3	1	4	0.35
v_fma_f32, v4, s17, v1, v4	1	4	0.35
v_fma_f32, v5, s18, v1, v5	1	4	0.35
v_fma_f32, v0, s19, v1, v0	1	8	0.70
v_fma_f32, v1, s4, v2, v3	1	4	0.35
v_fma_f32, v3, s5, v2, v4	1	4	0.35
v_fma_f32, v4, s6, v2, v5	1	4	0.35
v_fma_f32, v0, s7, v2, v0	1	4	0.35
v_add_f32_e32, v1, s0, v1	1	4	0.35
v_add_f32_e32, v2, s1, v3	1	4	0.35
v_add_f32_e32, v3, s2, v4	1	4	0.35
v_add_f32_e32, v0, s3, v0	1	4	0.35
exp, pos0, v2, v3, v0	1	4	0.35
exp, param0, off, off, off	1	120	10.49
s_endpgm	1	4	0.35

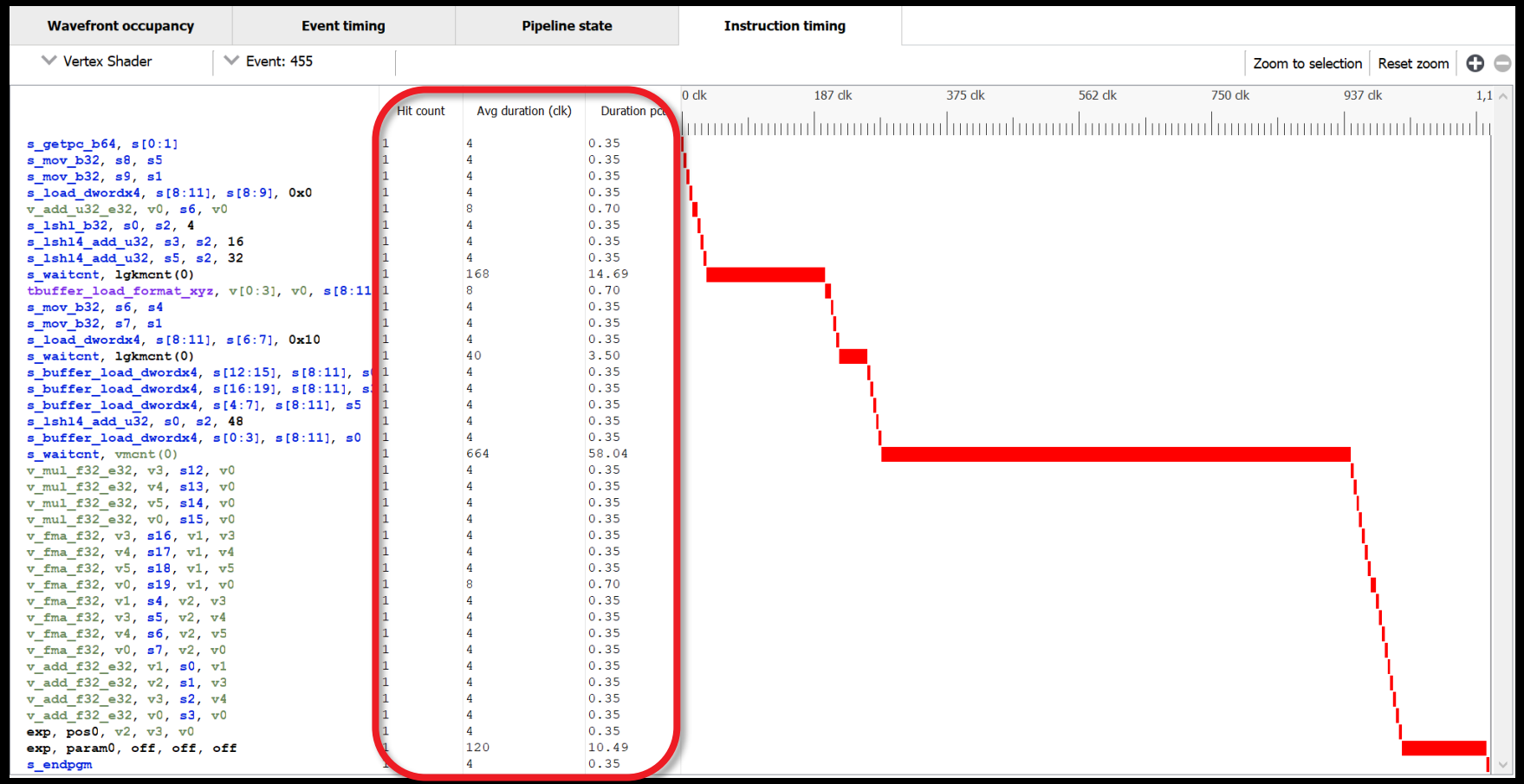
- Top-down program execution
- Find which part of your program is hot
- Available for all shader stages
- See instruction durations
- Functional unit utilization (VALU, SALU, LDS)
- Does not require shader modification

Planned for RGP 1.5.x - Instruction timing



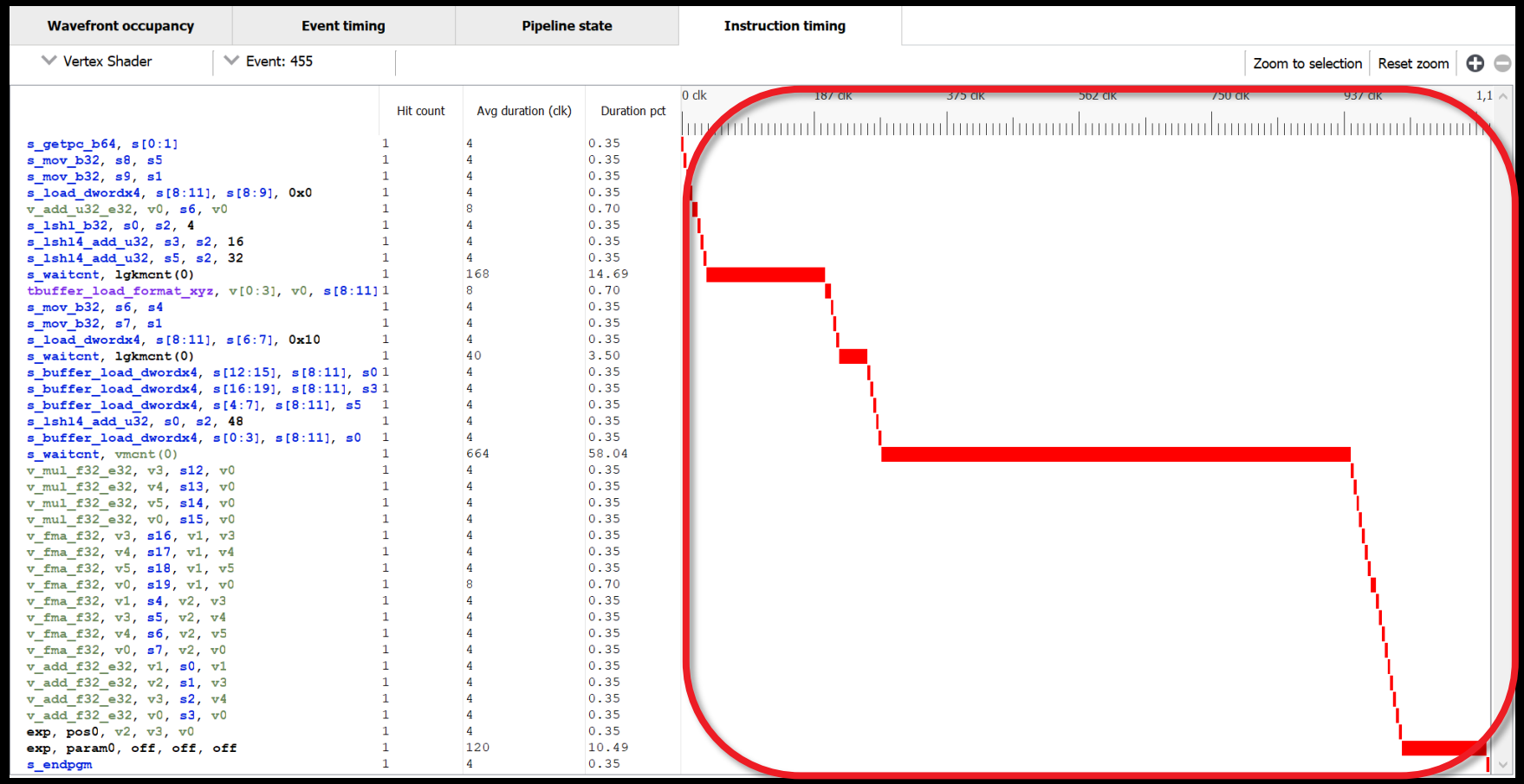
- Top-down program execution
- Find which part of your program is hot
- Available for all shader stages
- See instruction durations
- Functional unit utilization (VALU, SALU, LDS)
- Does not require shader modification

Planned for RGP 1.5.x - Instruction timing



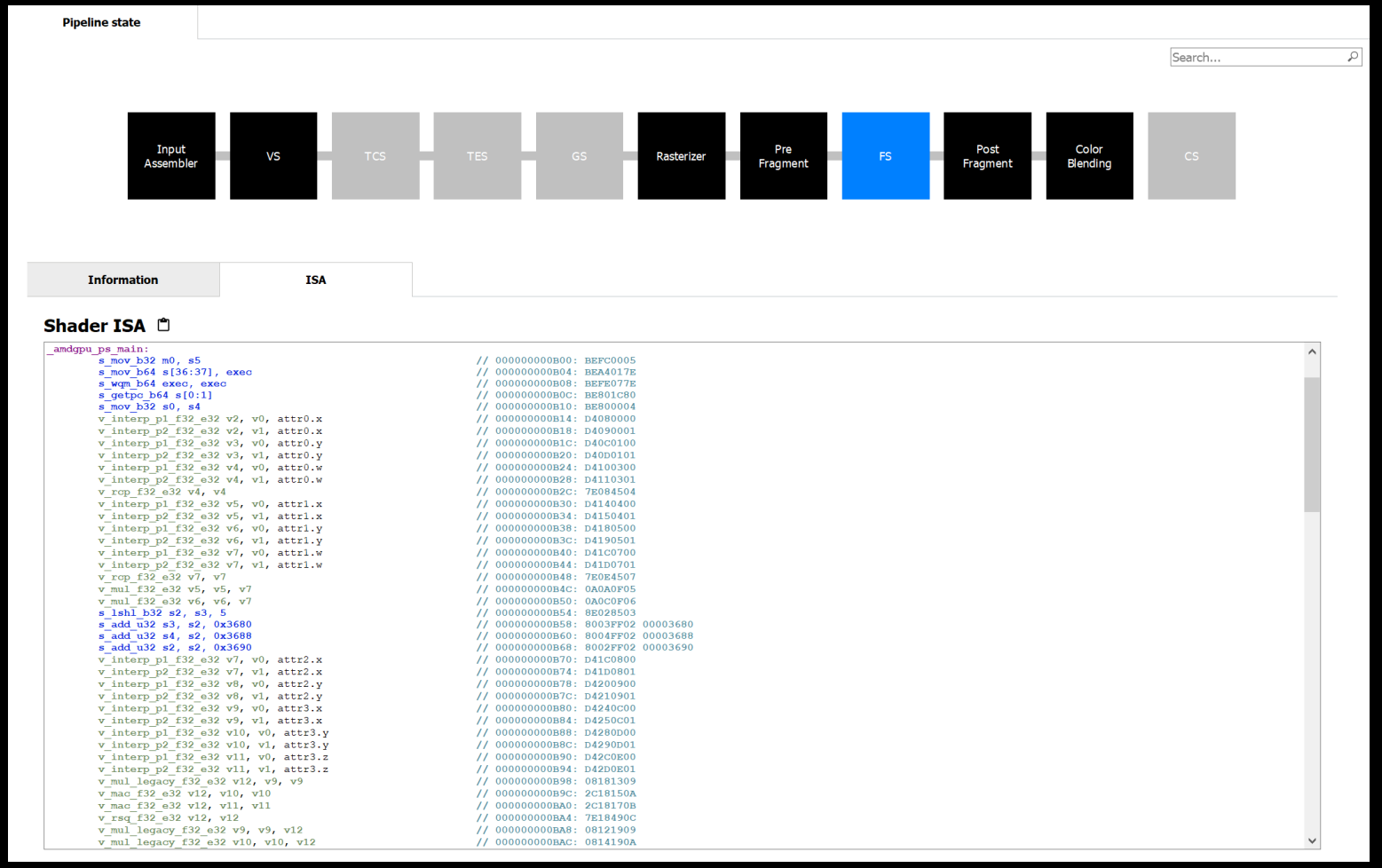
- Top-down program execution
- Find which part of your program is hot
- Available for all shader stages
- See instruction durations
- Functional unit utilization (VALU, SALU, LDS)
- Does not require shader modification

Planned for RGP 1.5.x - Instruction timing



- Top-down program execution
- Find which part of your program is hot
- Available for all shader stages
- See instruction durations
- Functional unit utilization (VALU, SALU, LDS)
- Does not require shader modification

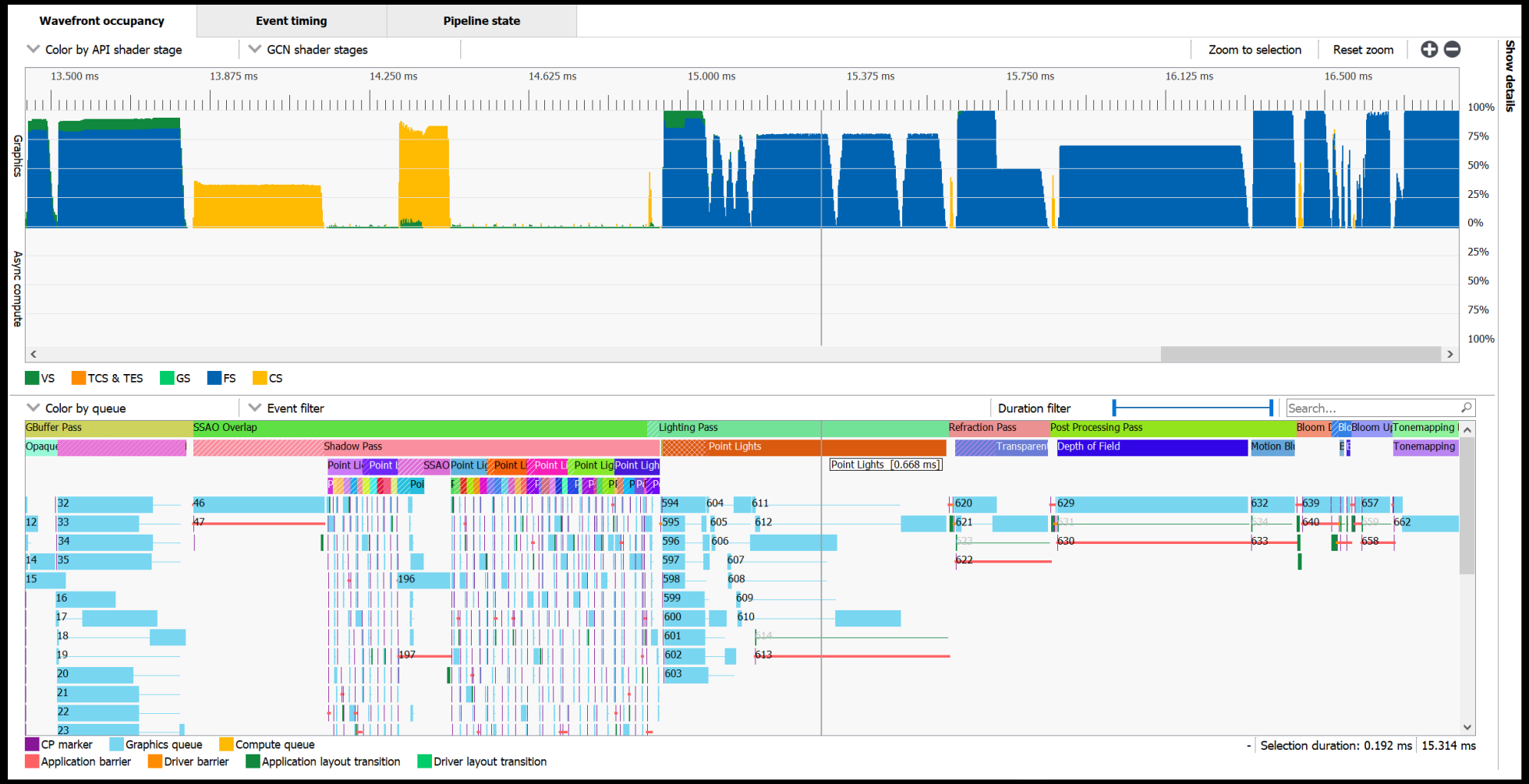
Planned for RGP 1.5.x - Shader ISA




```
_amdgpu_ps_main:
        s_mov_b32 m0, s5                                    // 000000000B00: BEFC0005
        s_mov_b64 s[36:37], exec                            // 000000000B04: BEA4017E
        s_wqm_b64 exec, exec                                // 000000000B08: BEFE077E
        s_getpc_b64 s[0:1]                                  // 000000000B0C: BE801C80
        s_mov_b32 s0, s4                                    // 000000000B10: BE800004
        v_interp_p1_f32_e32 v2, v0, attr0.x                 // 000000000B14: D4080000
        v_interp_p2_f32_e32 v2, v1, attr0.x                 // 000000000B18: D4090001
        v_interp_p1_f32_e32 v3, v0, attr0.y                 // 000000000B1C: D40C0100
        v_interp_p2_f32_e32 v3, v1, attr0.y                 // 000000000B20: D40D0101
        v_interp_p1_f32_e32 v4, v0, attr0.w                 // 000000000B24: D4100300
        v_interp_p2_f32_e32 v4, v1, attr0.w                 // 000000000B28: D4110301
        v_rcp_f32_e32 v4, v4                                // 000000000B2C: 7E084504
        v_interp_p1_f32_e32 v5, v0, attr1.x                 // 000000000B30: D4140400
        v_interp_p2_f32_e32 v5, v1, attr1.x                 // 000000000B34: D4150401
        v_interp_p1_f32_e32 v6, v0, attr1.y                 // 000000000B38: D4180500
        v_interp_p2_f32_e32 v6, v1, attr1.y                 // 000000000B3C: D4190501
        v_interp_p1_f32_e32 v7, v0, attr1.w                 // 000000000B40: D41C0700
        v_interp_p2_f32_e32 v7, v1, attr1.w                 // 000000000B44: D41D0701
        v_rcp_f32_e32 v7, v7                                // 000000000B48: 7E0E4507
        v_mul_f32_e32 v5, v5, v7                            // 000000000B4C: 0A0A0F05
        v_mul_f32_e32 v6, v6, v7                            // 000000000B50: 0A0C0F06
        s_lshl_b32 s2, s3, 5                                // 000000000B54: 8E028503
        s_add_u32 s3, s2, 0x3680                            // 000000000B58: 8003FF02 00003680
        s_add_u32 s4, s2, 0x3688                            // 000000000B60: 8004FF02 00003688
        s_add_u32 s2, s2, 0x3690                            // 000000000B68: 8002FF02 00003690
        v_interp_p1_f32_e32 v7, v0, attr2.x                 // 000000000B70: D41C0800
        v_interp_p2_f32_e32 v7, v1, attr2.x                 // 000000000B74: D41D0801
        v_interp_p1_f32_e32 v8, v0, attr2.y                 // 000000000B78: D4200900
        v_interp_p2_f32_e32 v8, v1, attr2.y                 // 000000000B7C: D4210901
        v_interp_p1_f32_e32 v9, v0, attr3.x                 // 000000000B80: D4240C00
        v_interp_p2_f32_e32 v9, v1, attr3.x                 // 000000000B84: D4250C01
        v_interp_p1_f32_e32 v10, v0, attr3.y                // 000000000B88: D4280D00
        v_interp_p2_f32_e32 v10, v1, attr3.y                // 000000000B8C: D4290D01
        v_interp_p1_f32_e32 v11, v0, attr3.z                // 000000000B90: D42C0E00
        v_interp_p2_f32_e32 v11, v1, attr3.z                // 000000000B94: D42D0E01
        v_mul_legacy_f32_e32 v12, v9, v9                    // 000000000B98: 08181309
        v_mac_f32_e32 v12, v10, v10                         // 000000000B9C: 2C18150A
        v_mac_f32_e32 v12, v11, v11                         // 000000000BA0: 2C18170B
        v_rsq_f32_e32 v12, v12                              // 000000000BA4: 7E18490C
        v_mul_legacy_f32_e32 v9, v9, v12                    // 000000000BA8: 08121909
        v_mul_legacy_f32_e32 v10, v10, v12                  // 000000000BAC: 0814190A
```

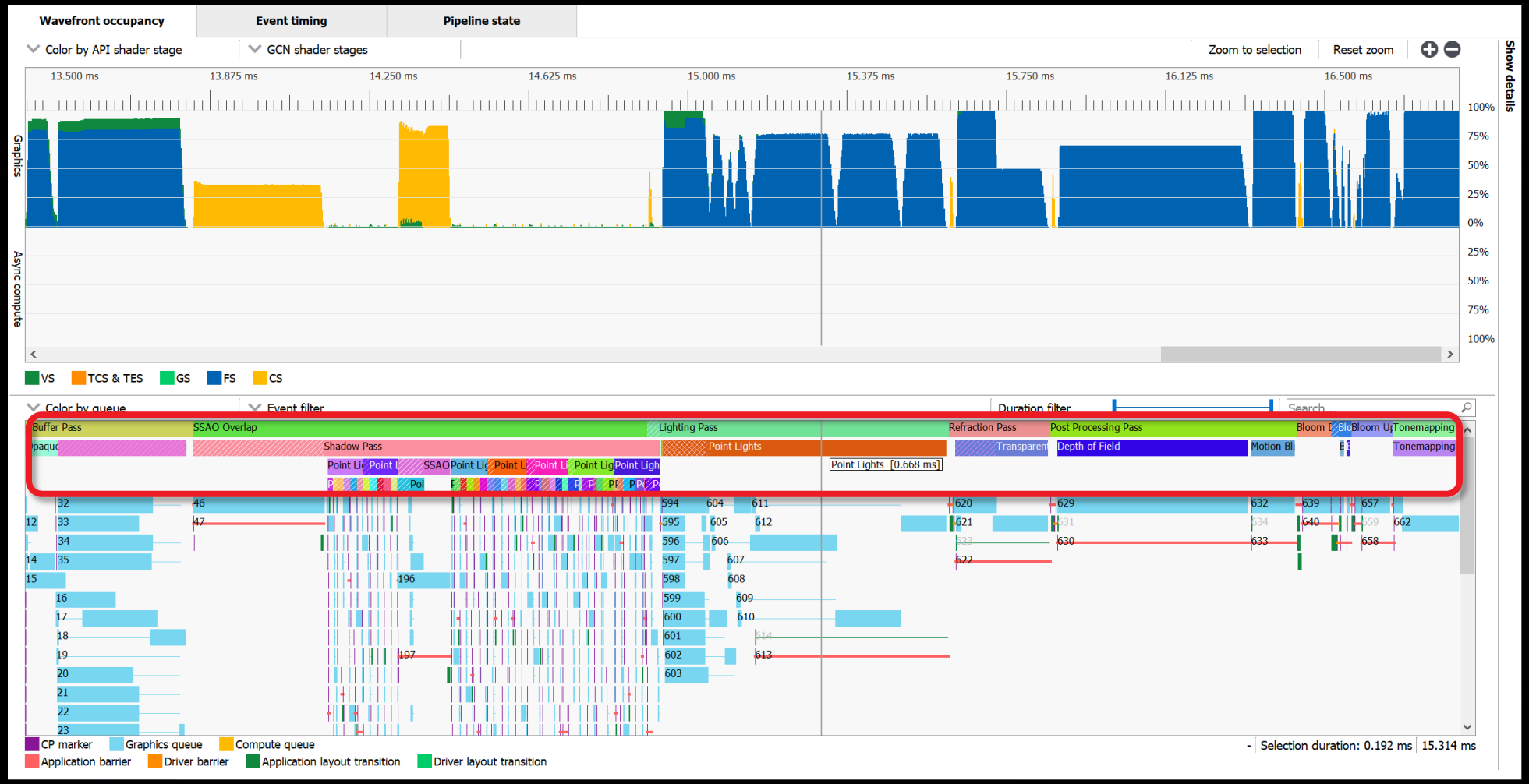
See shader code in pipeline state

Planned for RGP 1.5.x - User marker display



Know what the GPU is working on

Planned for RGP 1.5.x - User marker display



Know what the GPU is working on

AMD

AGENDA

Microsoft® PIX for Windows

Rodrigo Urra

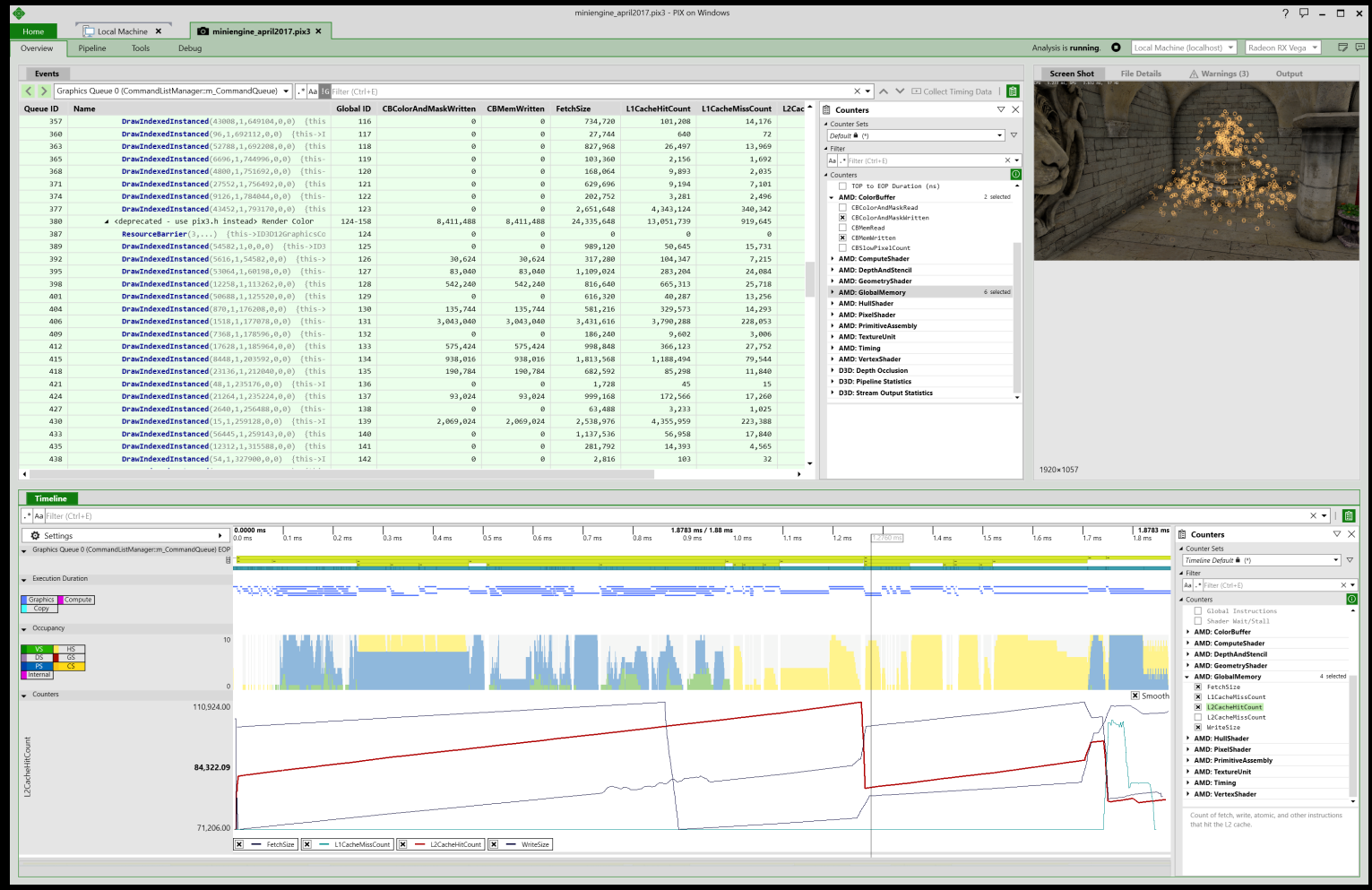
Introduction

AMD-specific GPU data

What is Microsoft® PIX for Windows?

Graphics debugging and perf tuning tool

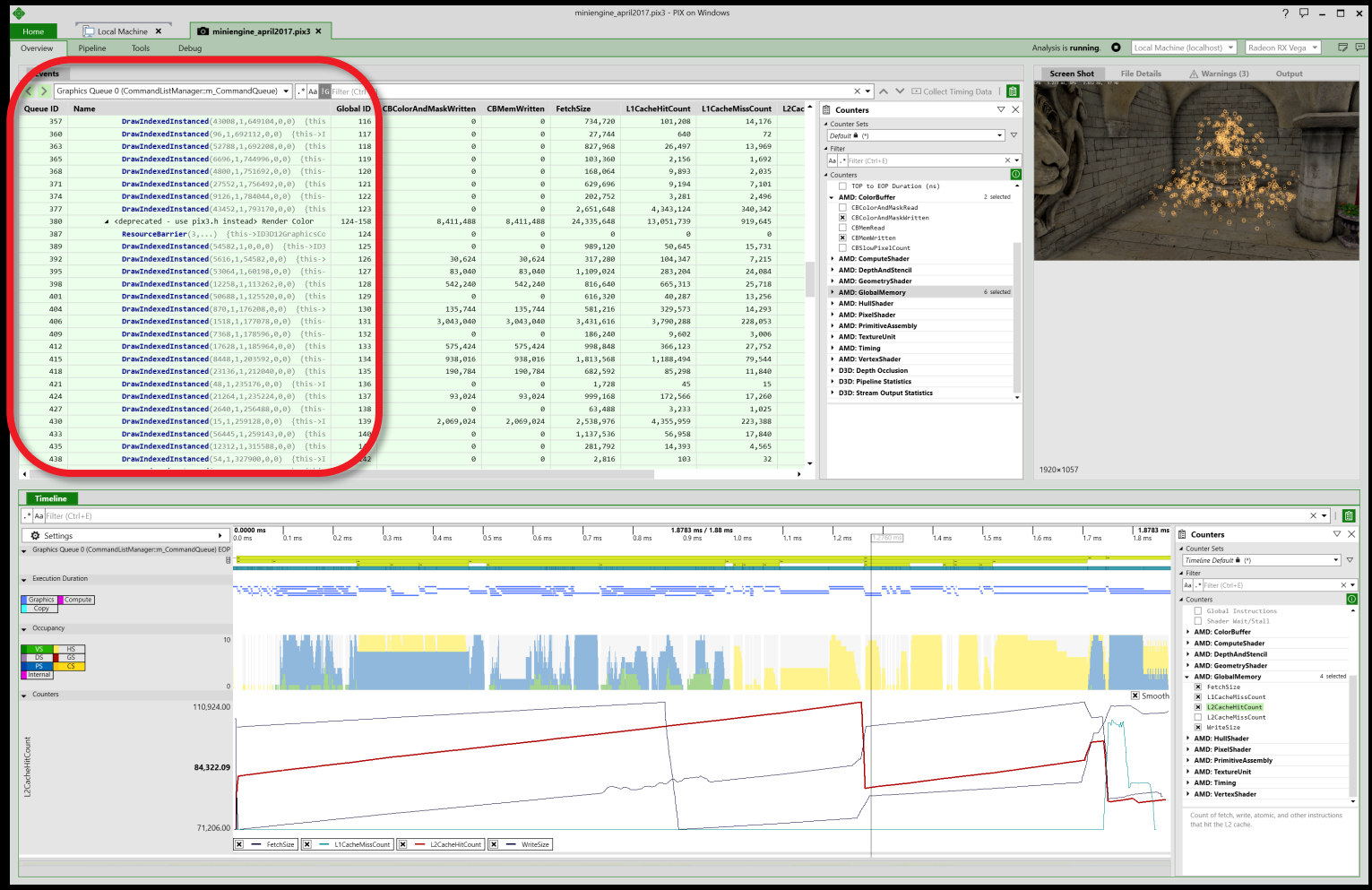
- DirectX® 12 on Windows 10



What is Microsoft® PIX for Windows?

Graphics debugging and perf tuning tool

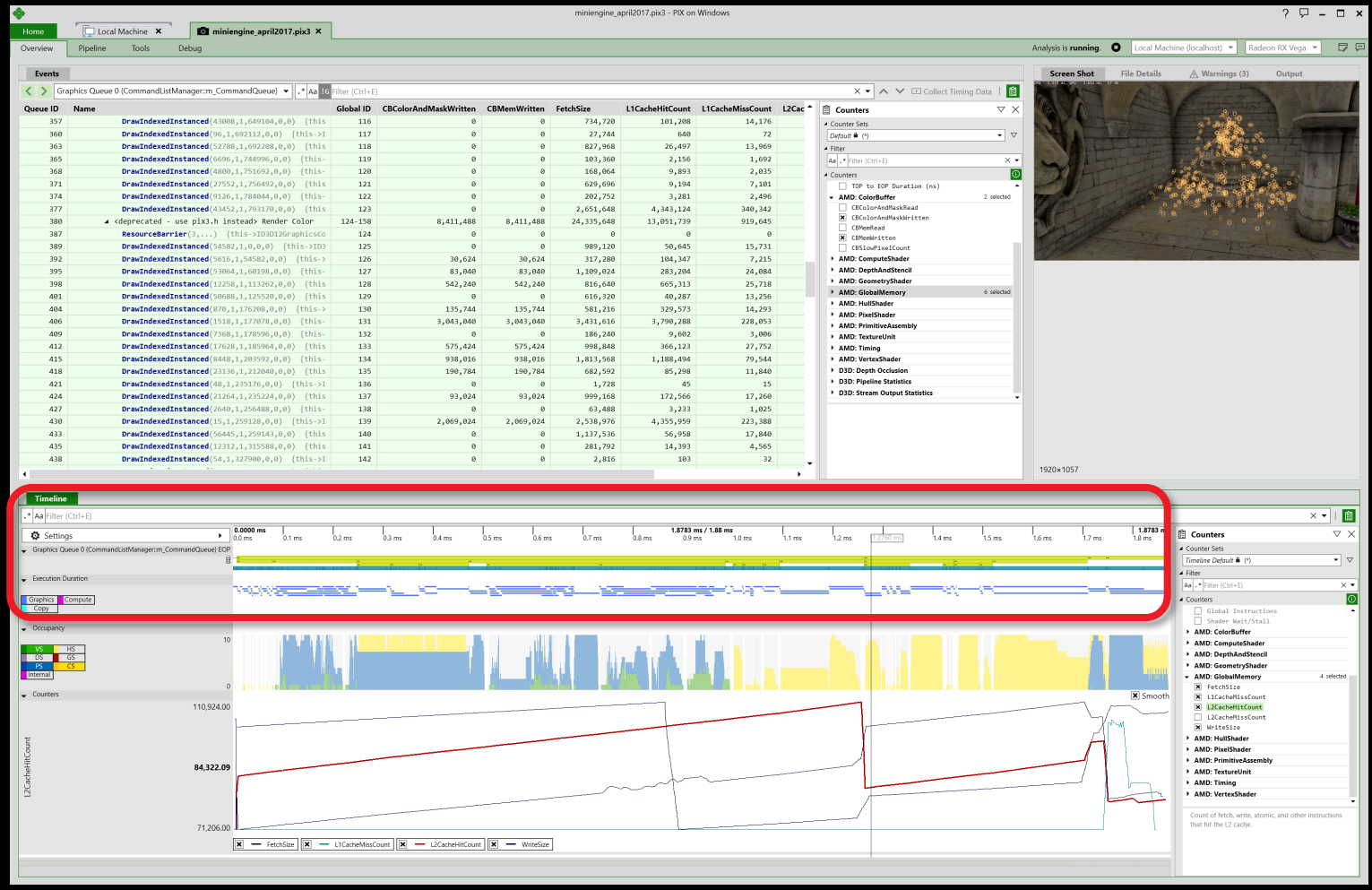
- DirectX® 12 on Windows 10



What is Microsoft® PIX for Windows?

Graphics debugging and perf tuning tool

- DirectX® 12 on Windows 10



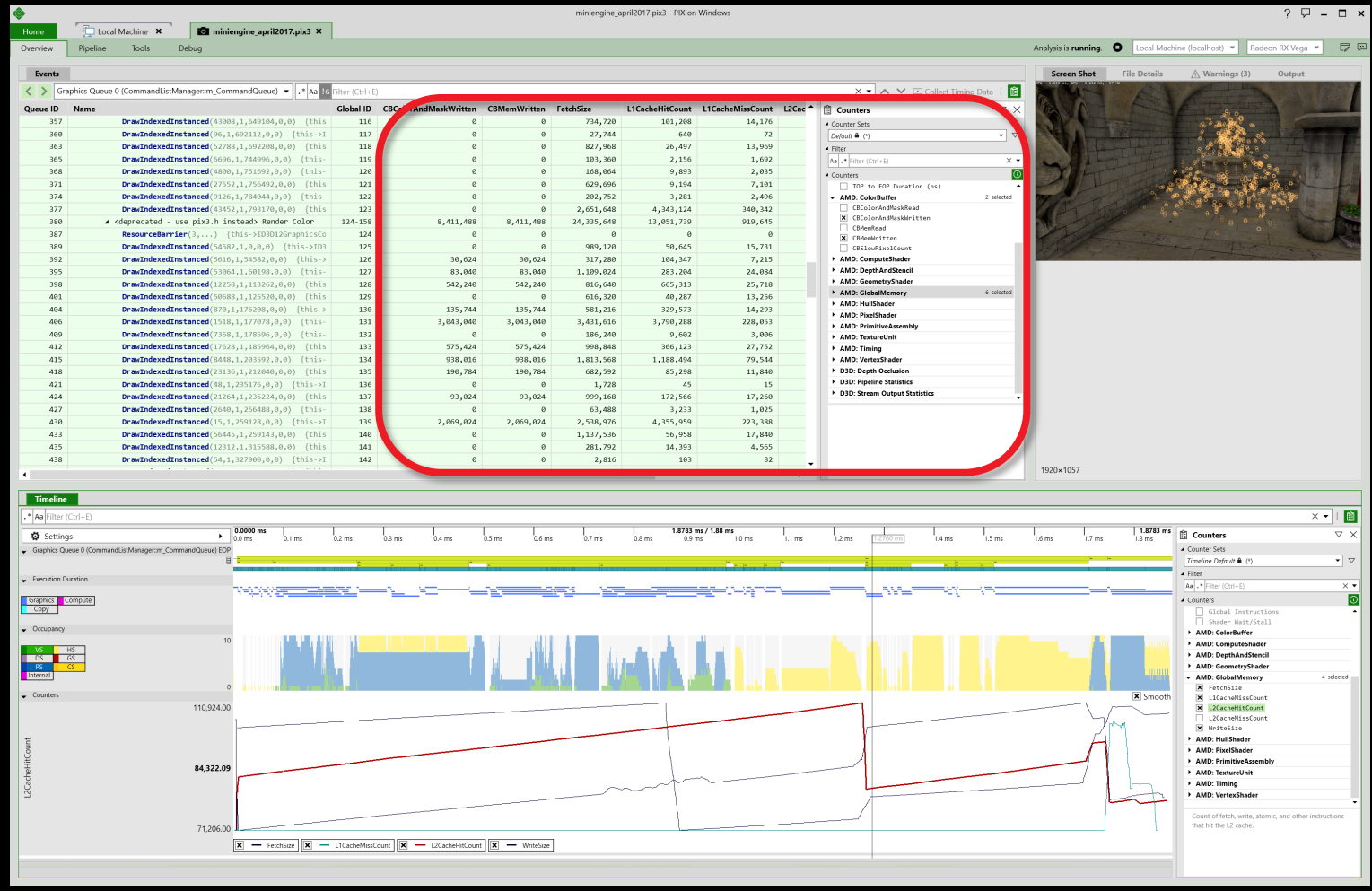
Microsoft® PIX for Windows – AMD support

Graphics debugging and perf tuning tool

- DirectX® 12 on Windows 10

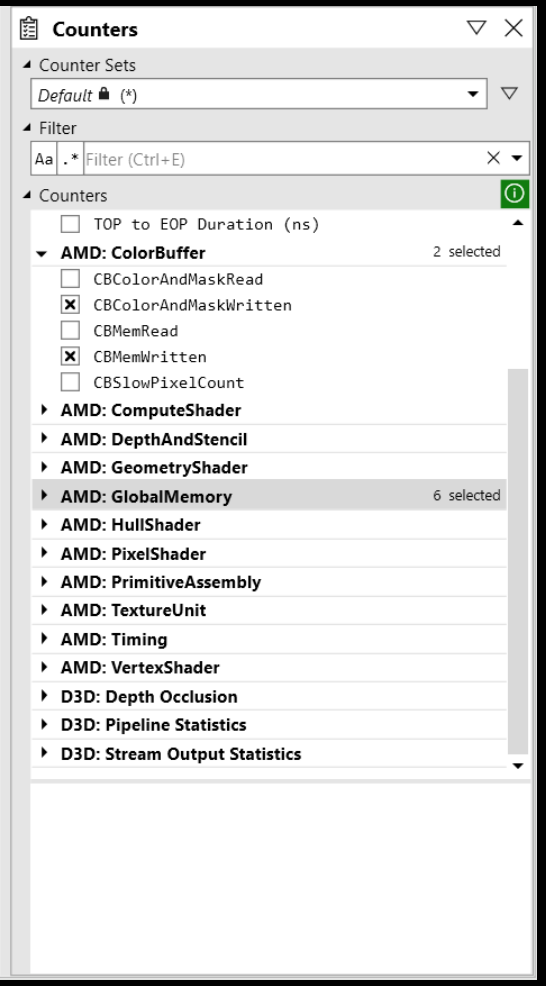
AMD-specific GPU data

- Event list counters



Event list counters: Command-specific counters

CBColorAndMaskWritten	CBMemWritten	FetchSize	L1CacheHitCount	L1CacheMissCount	L2Cac
0	0	734,720	101,208	14,176	
0	0	27,744	640	72	
0	0	827,968	26,497	13,969	
0	0	103,360	2,156	1,692	
0	0	168,064	9,893	2,035	
0	0	629,696	9,194	7,101	
0	0	202,752	3,281	2,496	
0	0	2,651,648	4,343,124	340,342	
8,411,488	8,411,488	24,335,648	13,051,739	919,645	
0	0	0	0	0	
0	0	989,120	50,645	15,731	
30,624	30,624	317,280	104,347	7,215	
83,040	83,040	1,109,024	283,204	24,084	
542,240	542,240	816,640	665,313	25,718	
0	0	616,320	40,287	13,256	
135,744	135,744	581,216	329,573	14,293	
3,043,040	3,043,040	3,431,616	3,790,288	228,053	
0	0	186,240	9,602	3,006	
575,424	575,424	998,848	366,123	27,752	
938,016	938,016	1,813,568	1,188,494	79,544	
190,784	190,784	682,592	85,298	11,840	
0	0	1,728	45	15	
93,024	93,024	999,168	172,566	17,260	
0	0	63,488	3,233	1,025	
2,069,024	2,069,024	2,538,976	4,355,959	223,388	
0	0	1,137,536	56,958	17,840	
0	0	281,792	14,393	4,565	
0	0	2,816	103	32	



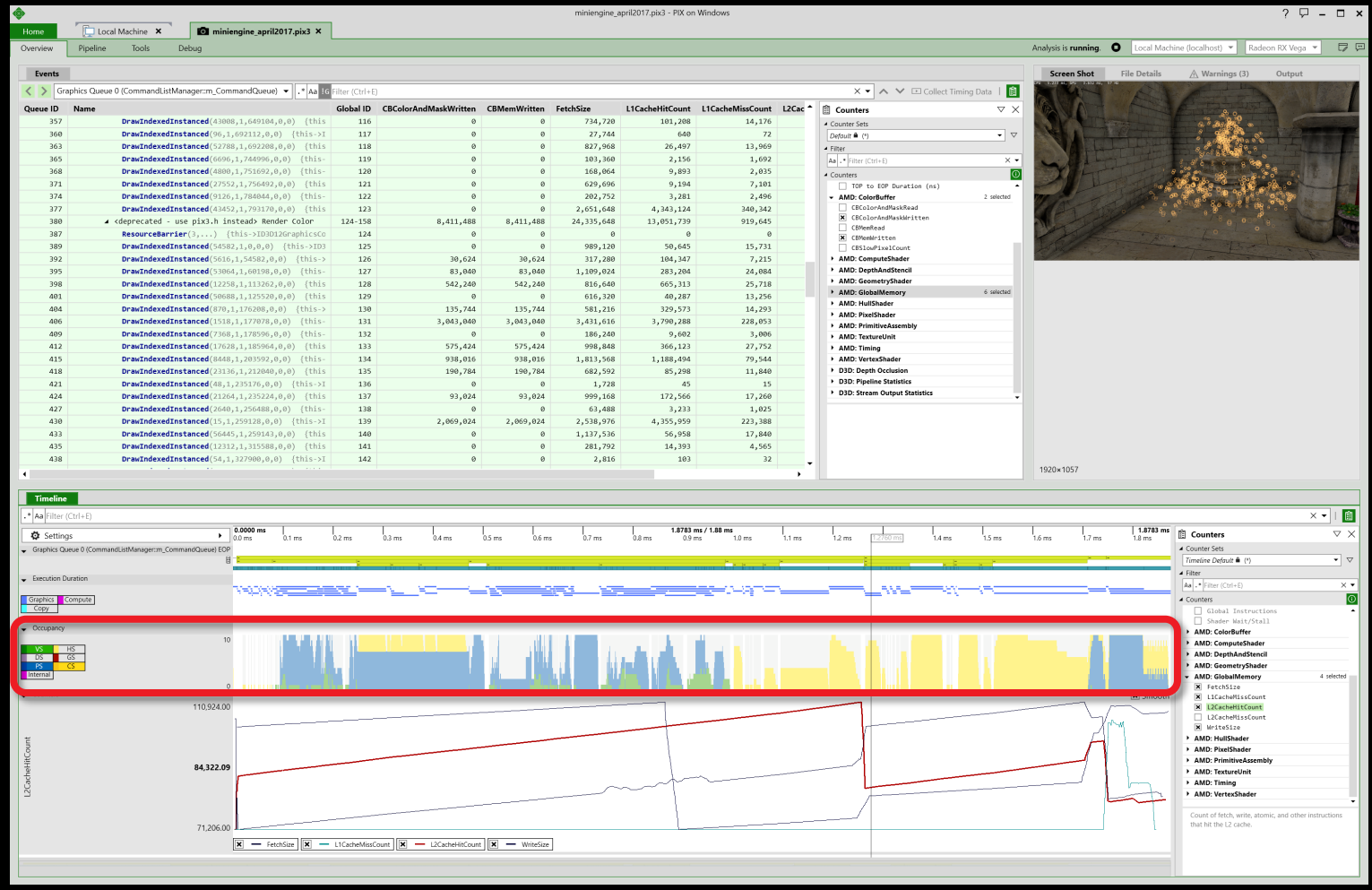
Microsoft® PIX for Windows – AMD support

Graphics debugging and perf tuning tool

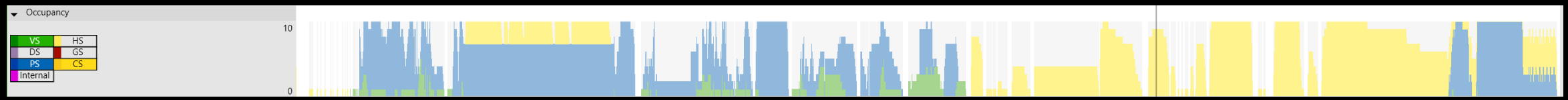
- DirectX® 12 on Windows 10

AMD-specific GPU data

- Event list counters
- Wavefront occupancy (SQTT)



Wavefront occupancy - Execution stage utilization



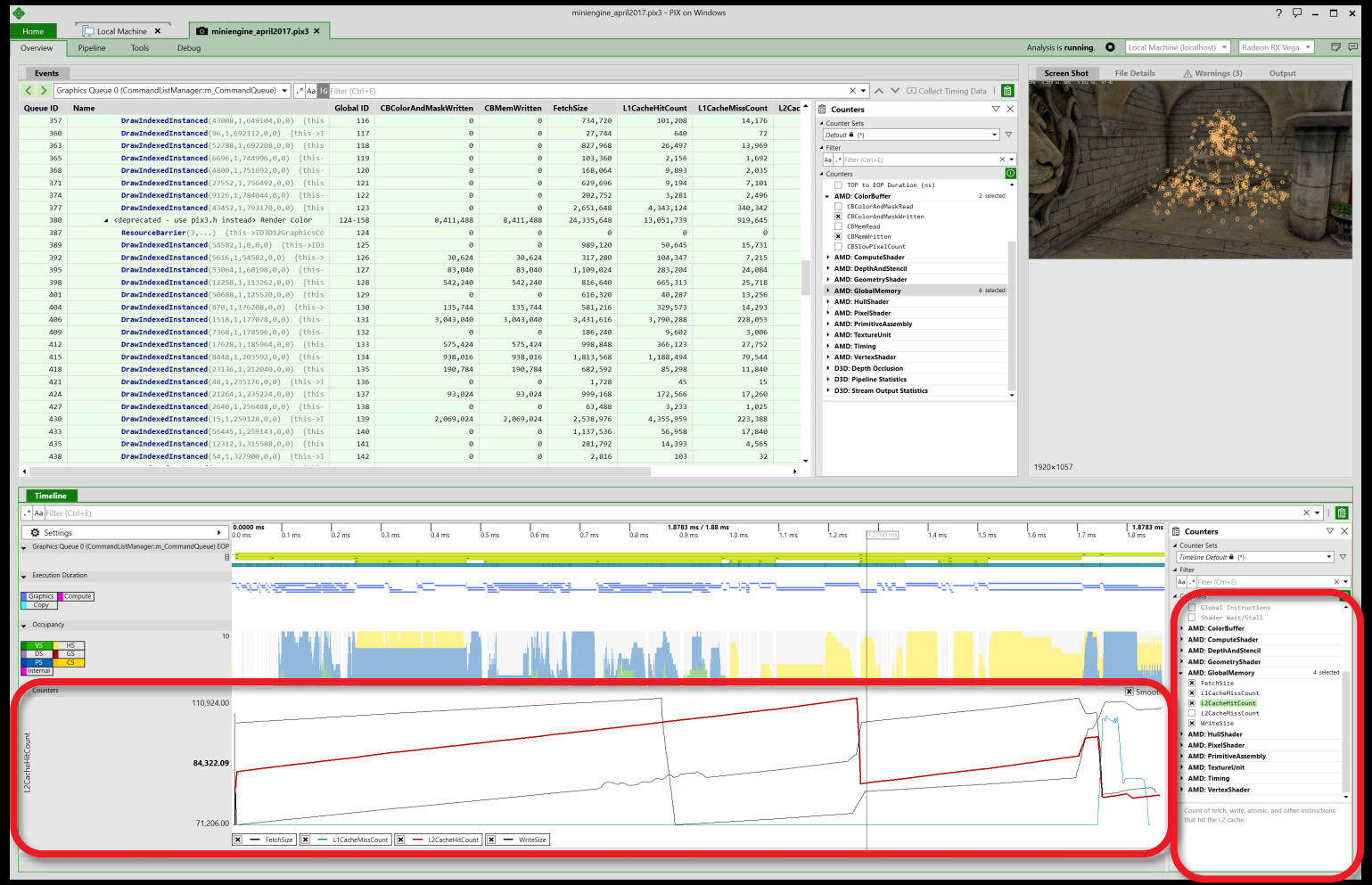
Microsoft® PIX for Windows – AMD support

Graphics debugging and perf tuning tool

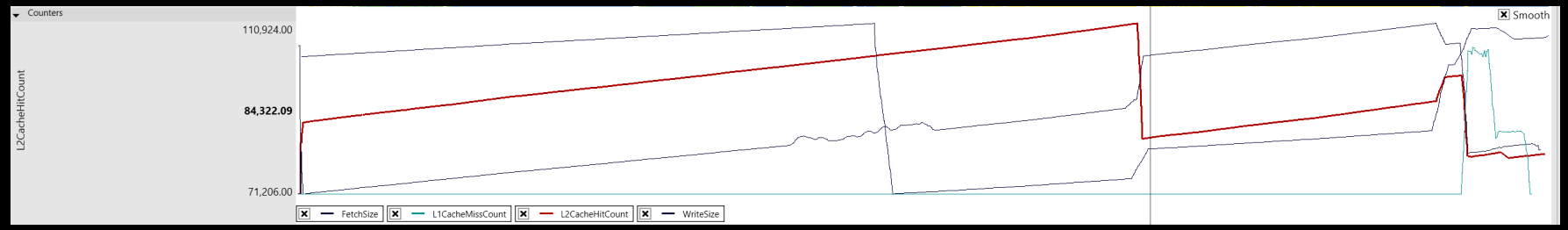
- DirectX® 12 on Windows 10

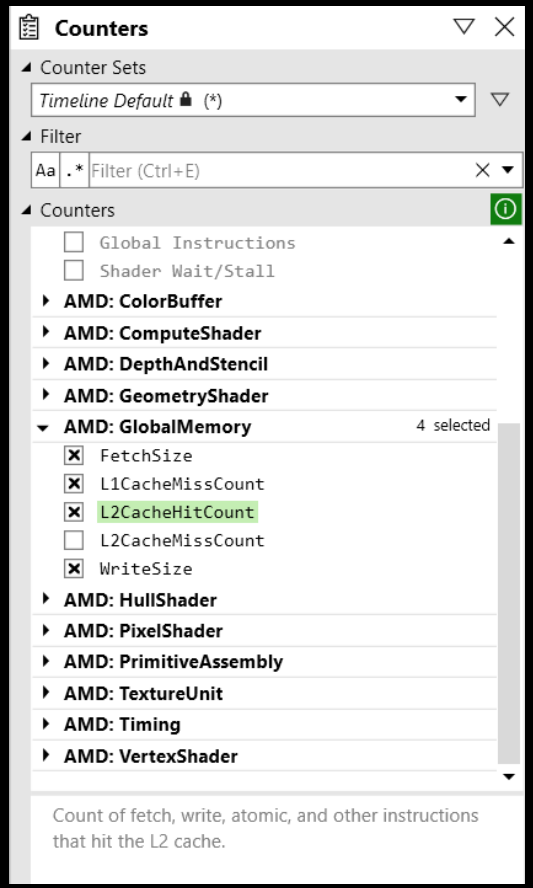
AMD-specific GPU data

- Event list counters
- Wavefront occupancy (SQTT)
- High frequency counters (SPM)



High frequency counters (HFC): ns execution sampling





Microsoft® PIX for Windows – AMD support

Graphics debugging and perf tuning tool

- DirectX® 12 on Windows 10

AMD-specific GPU data

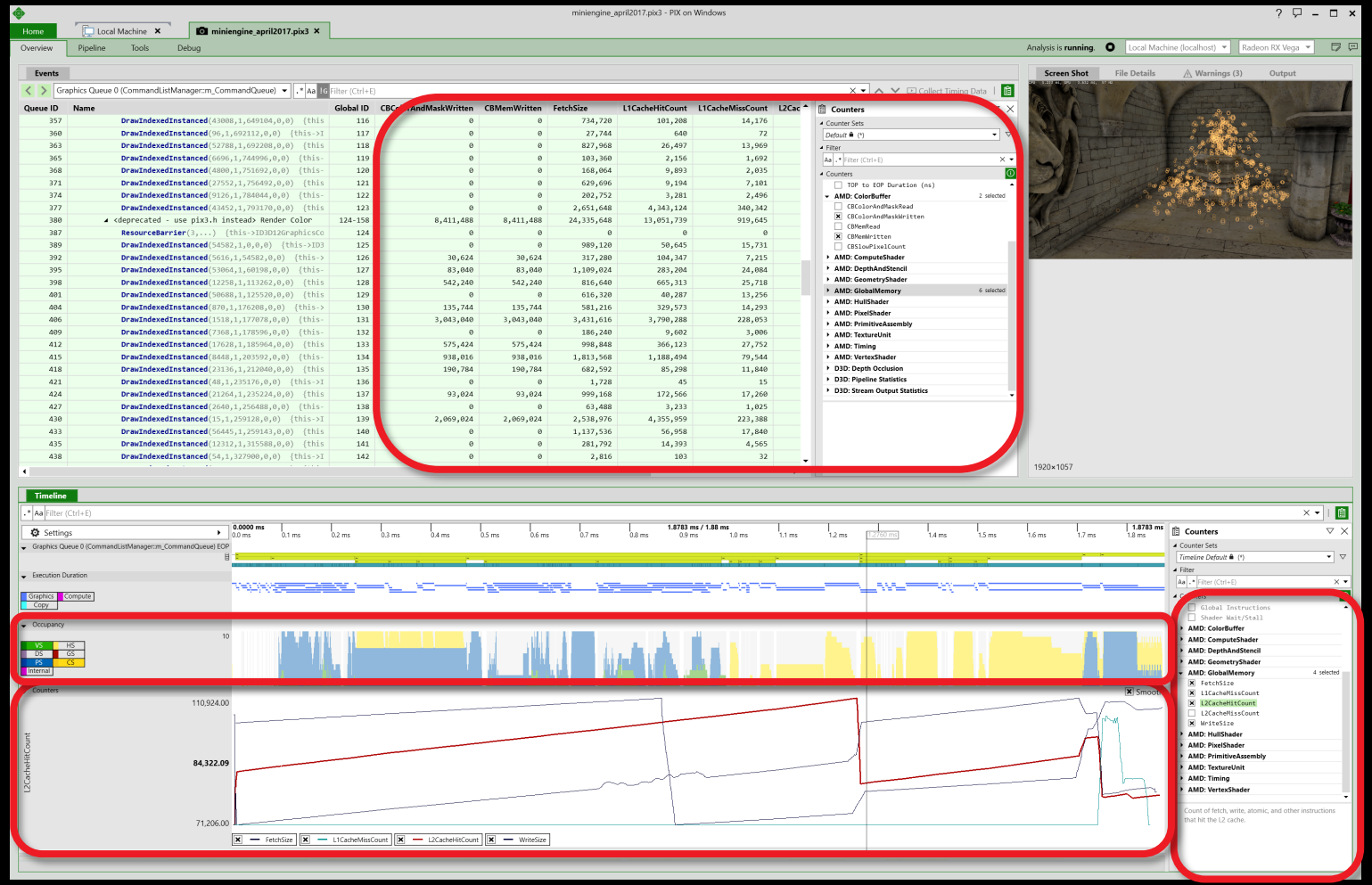
- Event list counters
- Wavefront occupancy (SQTT)
- High frequency counters (SPM)

Displayed data

- Includes driver + GPU work

Supported GPUs

Radeon™ VII, Radeon™ Vega, Radeon™ RX400 and RX500, and Intel Core i7 Radeon™ RX Vega M



AGENDA

Radeon™ GPU Analyzer

Amit Ben-Moshe

Introduction

RGA for Vulkan™

A look at the UI

The RGA layer

What is Radeon GPU Analyzer

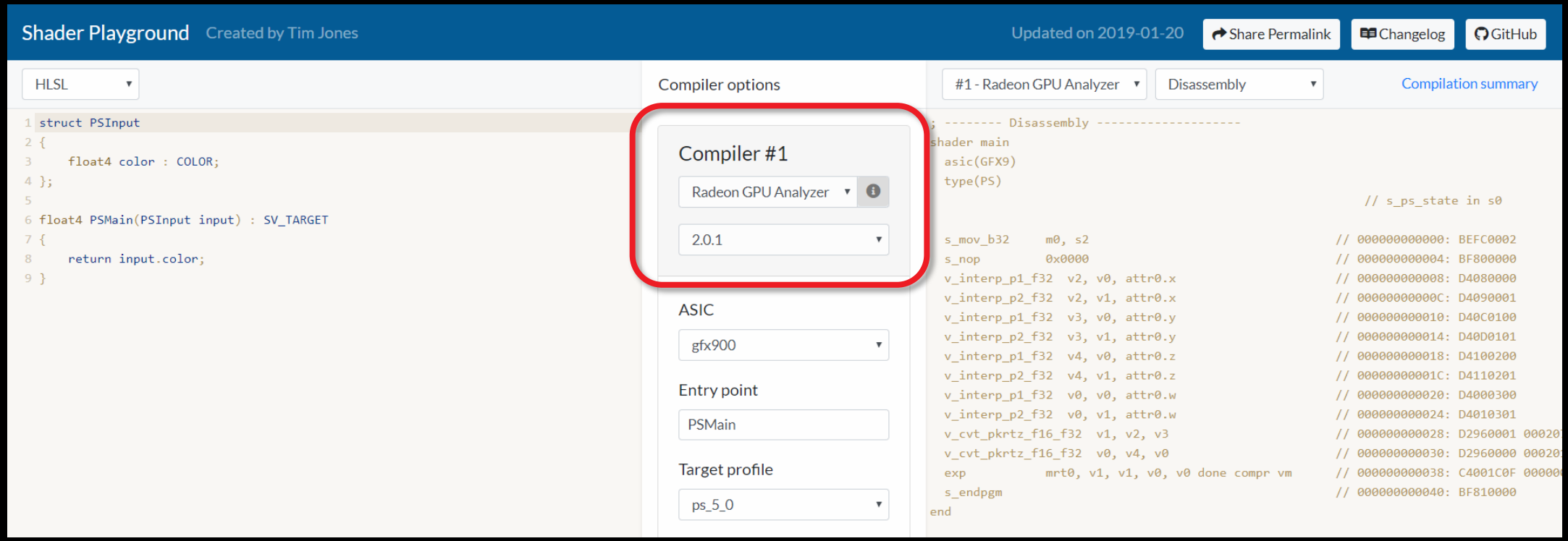
Compiler and code analysis tool

- Generate GCN ISA and IL disassembly
- Retrieve shader HW resource usage
- Static analysis
- Independent of the installed HW
- Designed to support both Linux® and Windows®
- Designed to support multiple APIs: Vulkan™, DirectX®11, OpenGL™, OpenCL™

Integration with other tools

Integrated into tools you love

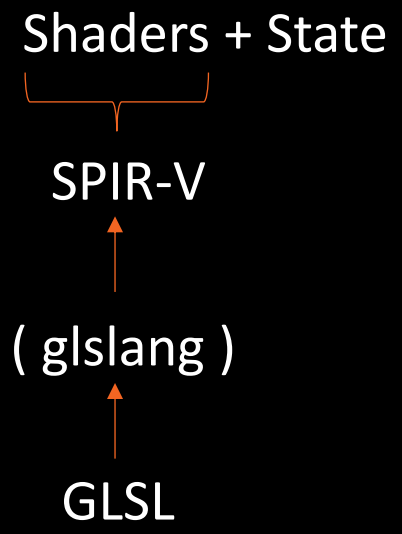
- RenderDoc, Shader Playground, CodeXL, Pyramid



RGA 2.1 for Vulkan™

New approach for shader optimization

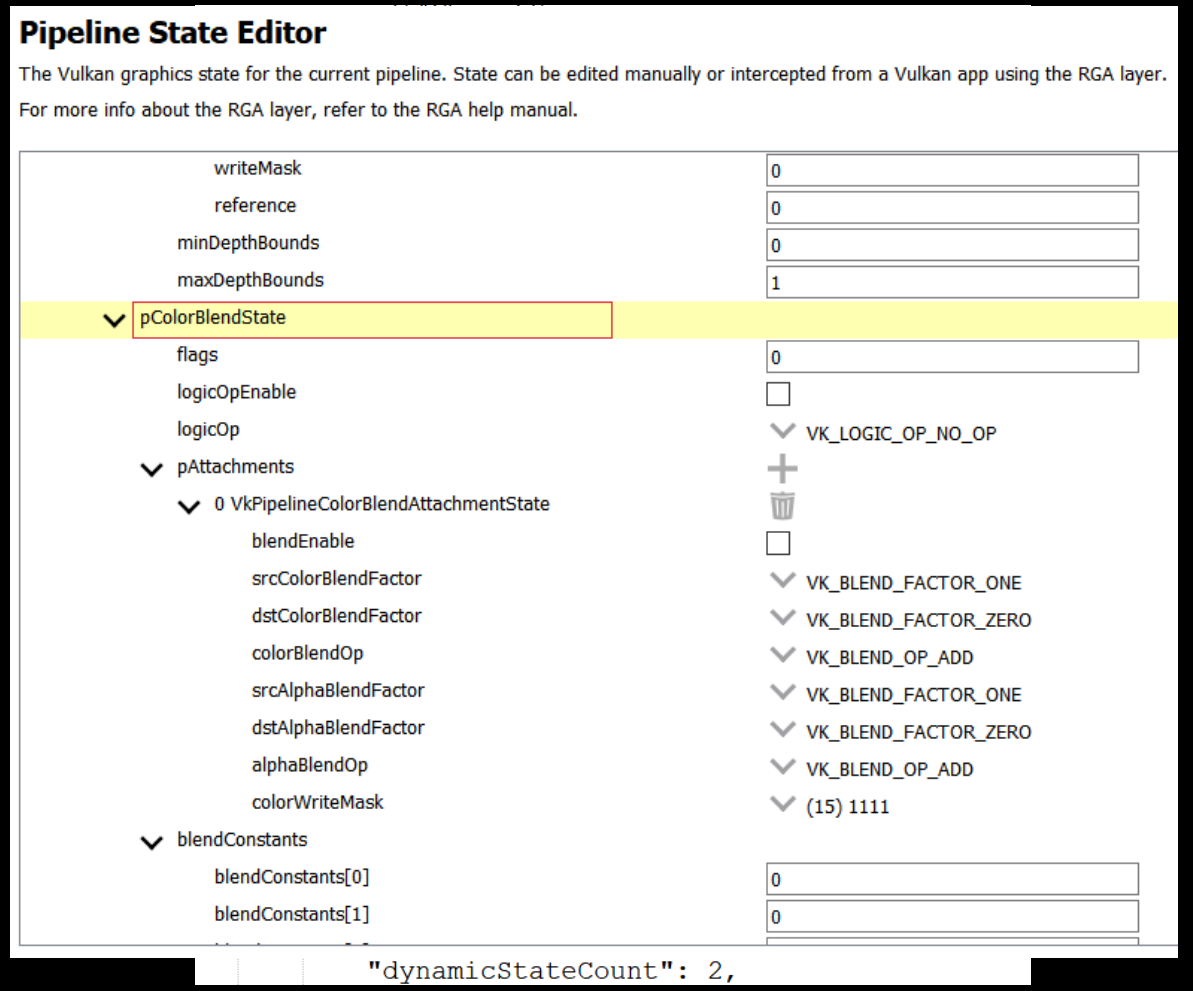
- Uses the live AMD driver compiler rather than offline compiler:
  - What you see is what you get
  - Compiler updates as you update the driver
- Vulkan™ app creates and compiles **Vulkan™ pipelines**



RGA for Vulkan™

Pipeline state

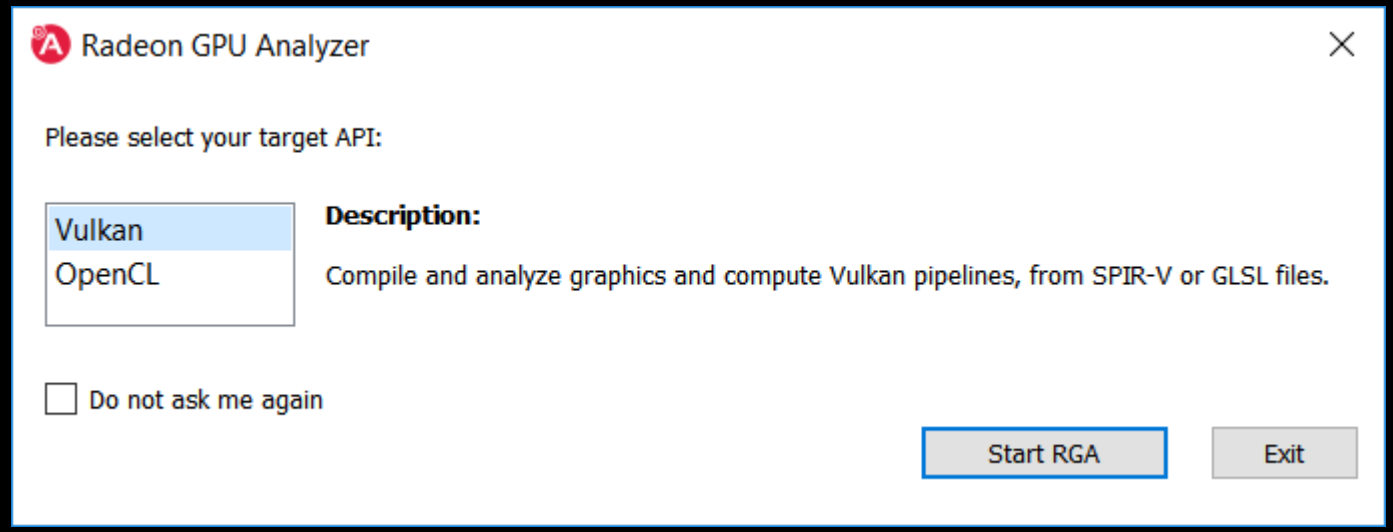
- JSON representation
  - VkGraphicsPipelineCreateInfo for graphics (.gpso)
  - VkComputePipelineCreateInfo for compute (.cpso)
- Edit/Compose Vulkan™ state in UI



The new RGA UI

Startup window

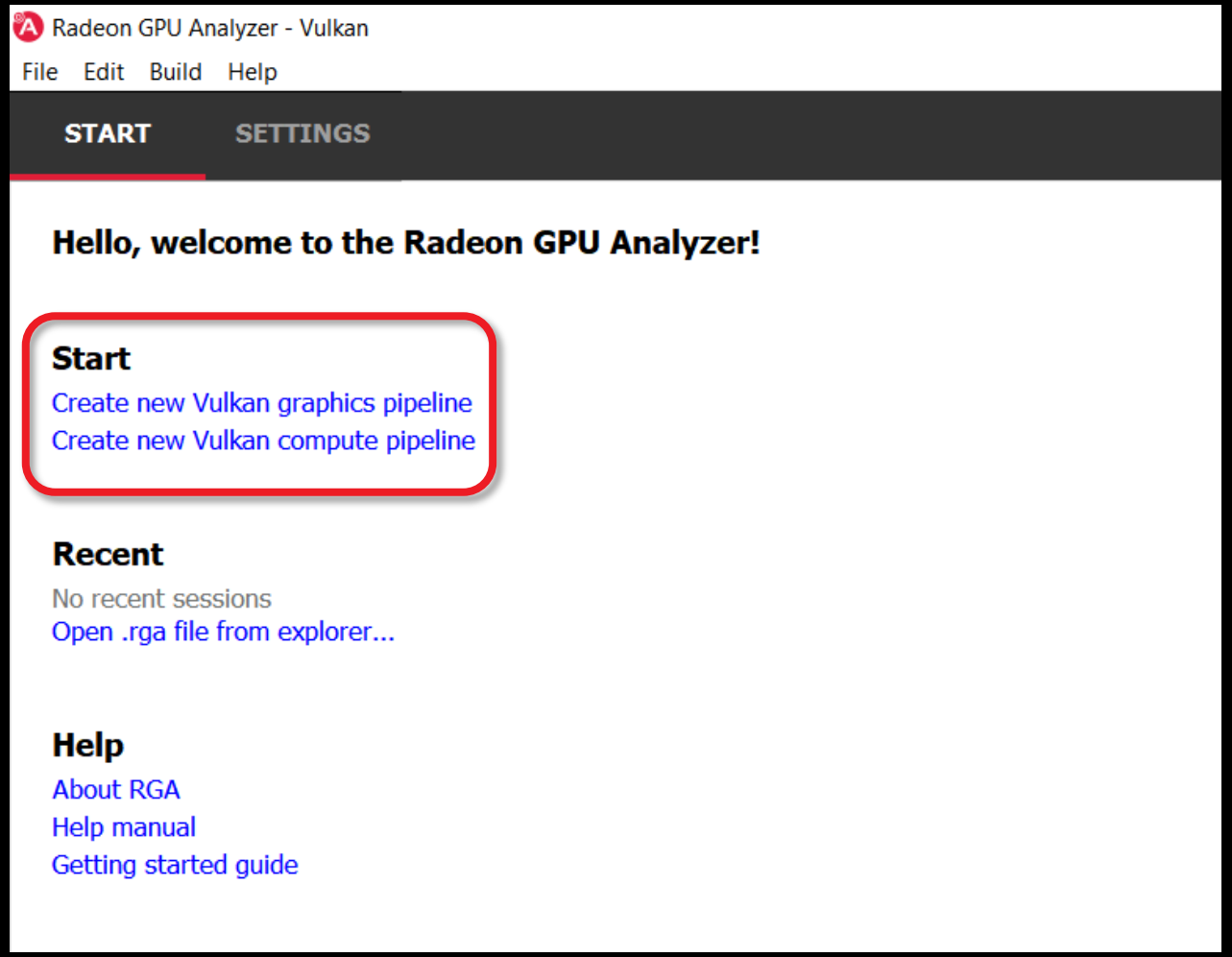
- Select your API of choice



Create new graphics/compute pipeline

Home page

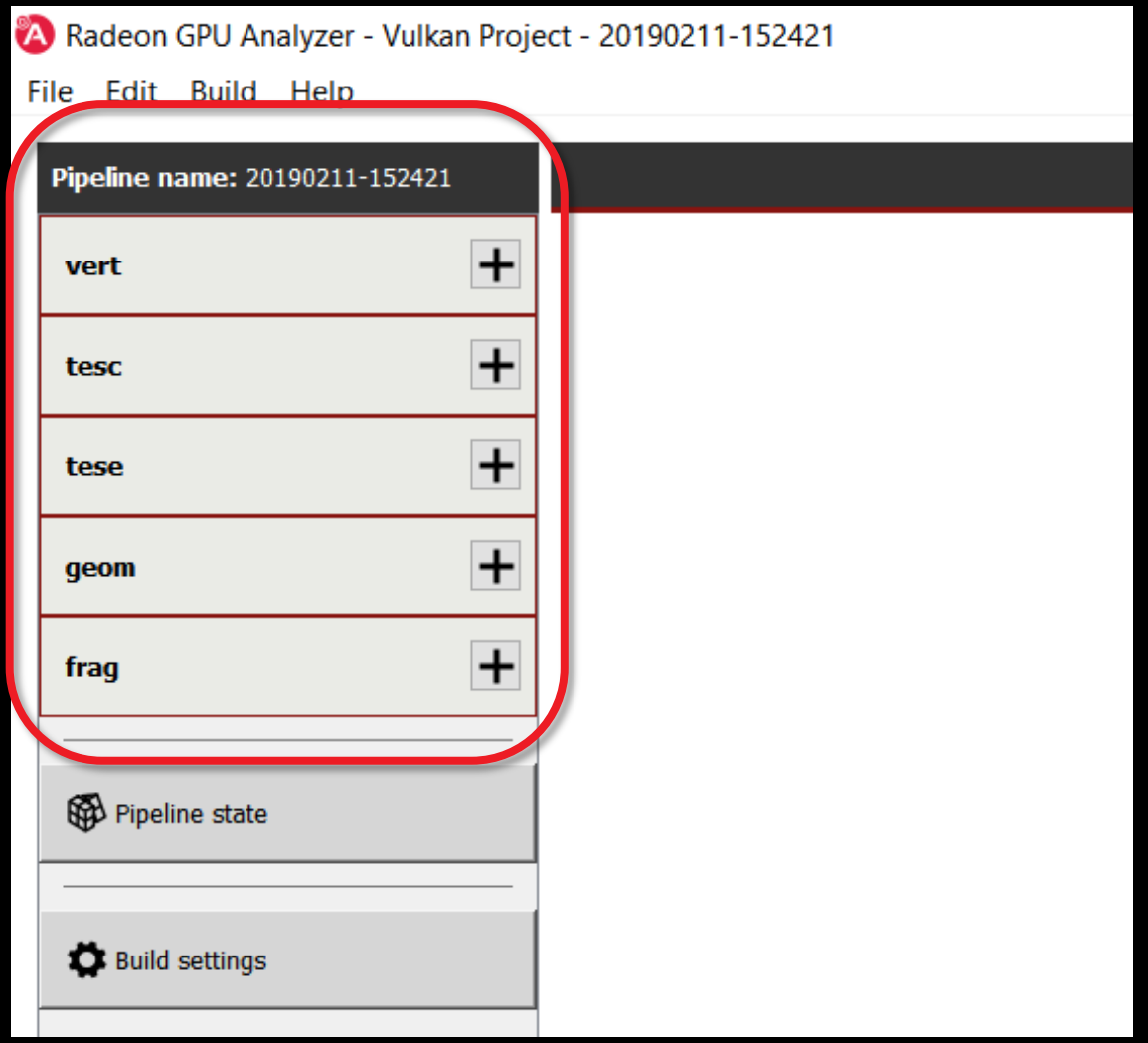
- Start by creating a graphics or compute pipeline



Add shaders

Two choices

- Create template GLSL shader using RGA
- Load or drag & drop existing shader file (GLSL / SPIR-V)



Edit GLSL / SPIR-V code

- Syntax highlighting
- Find text
- Jump to line
- Open header files (GLSL)

```
1  /* Auto-generated with Radeon GPU Analyzer (RGA).*/
2  #version 450
3
4  #extension GL_GOOGLE_include_directive : require
5  #extension GL_ARB_separate_shader_objects : enable
6
7  out gl_PerVertex
8  {
9      vec4 gl_Position;
10 };
11
12 layout(location = 0) out vec3 fragColor;
13
14 void main()
15 {
16     float x = -1.0 + float((gl_VertexIndex & 1) << 2);
17     float y = -1.0 + float((gl_VertexIndex & 2) << 1);
18     gl_Position = vec4(x, y, 0, 1);
19
20     fragColor = vec3(1.0f, 0.0f, 0.0f);
21 }
22
```

SPIR-V binary editing

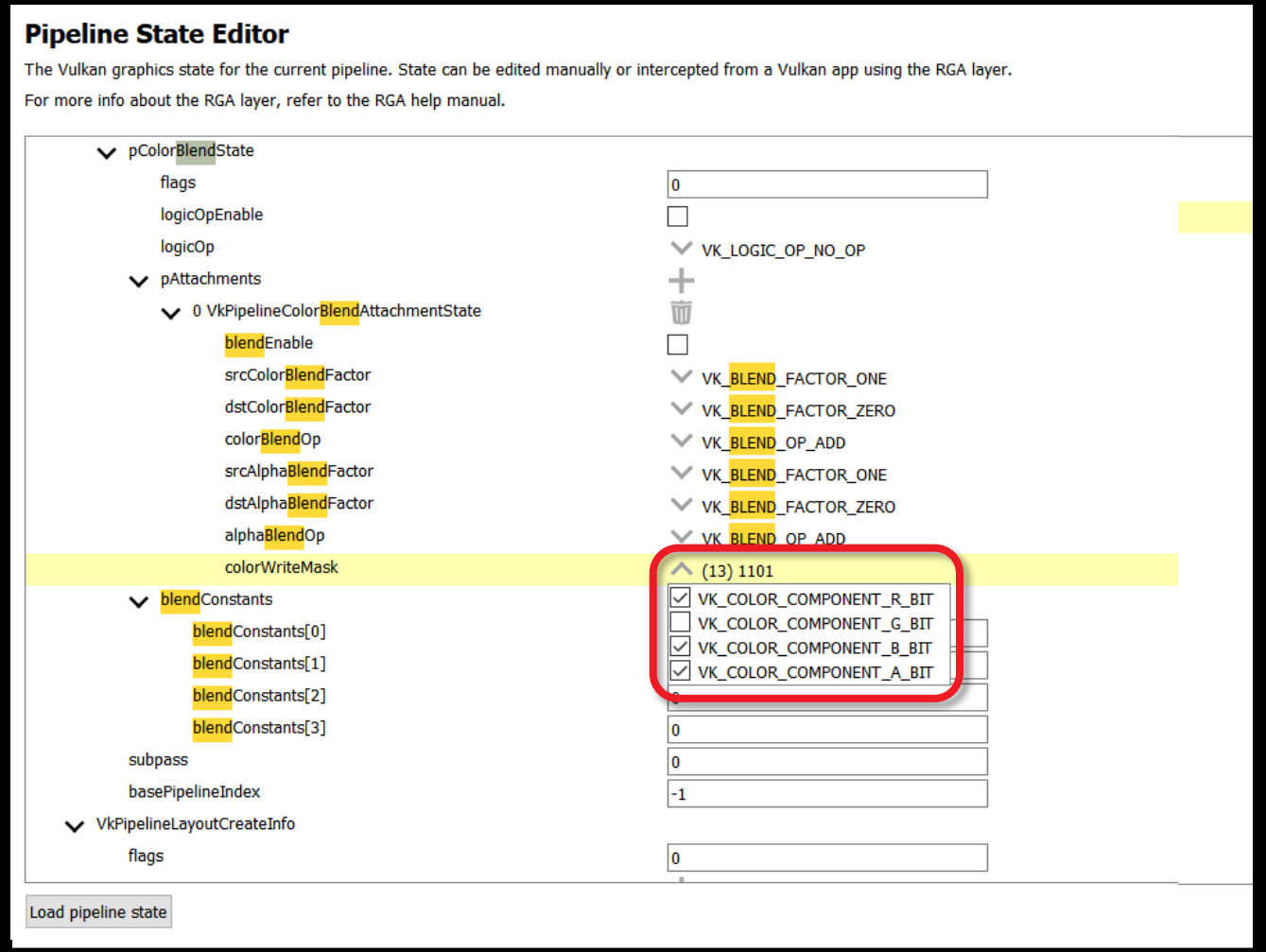
Drop SPIR-V binary in RGA

- Binary disassembled
- Syntax highlighted
- Edit and reassemble
- Revert to original

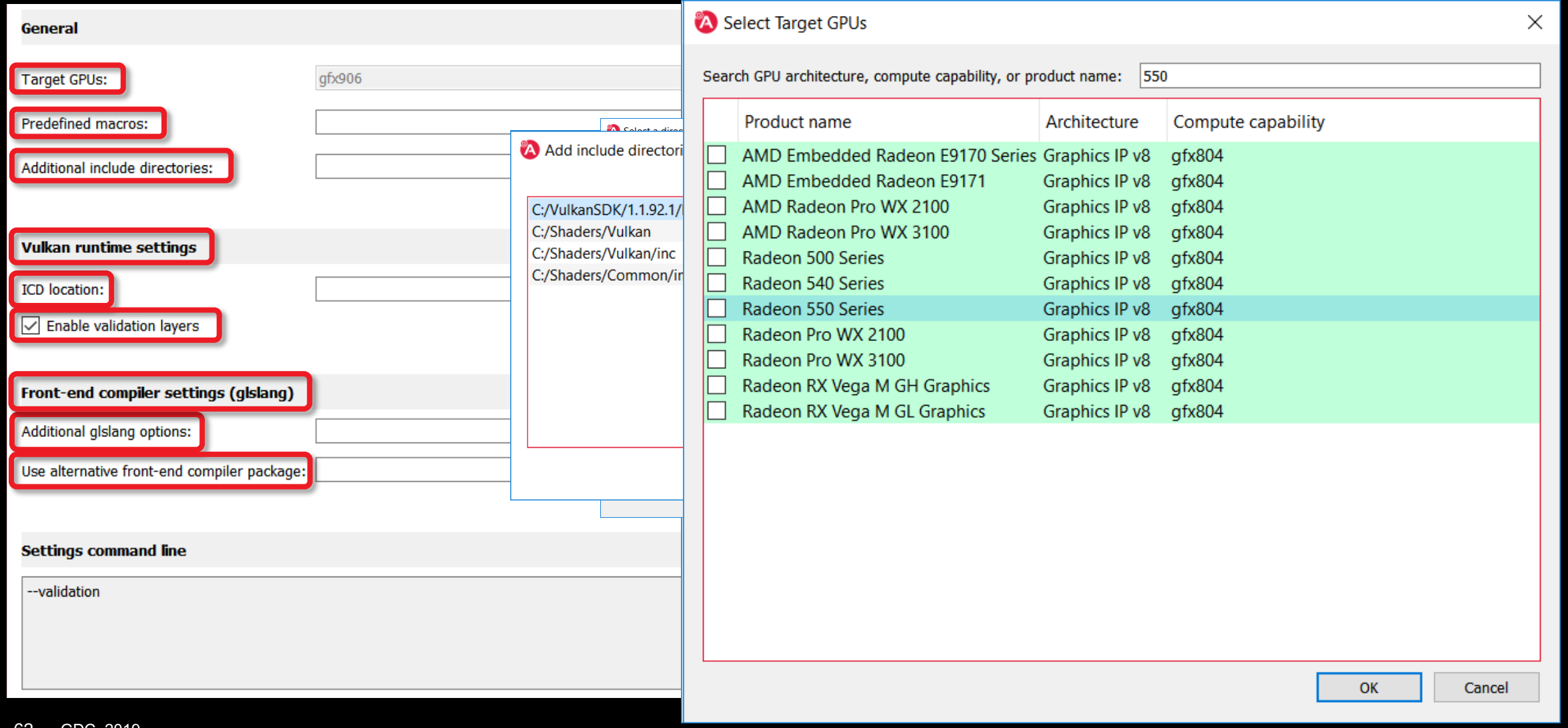
```
; SPIR-V
; Version: 1.0
; Generator: Khronos Glslang Reference Front End; 1
; Bound: 172
; Schema: 0
                OpCapability Shader
                OpCapability ImageQuery
         %1 = OpExtInstImport "GLSL.std.450"
                OpMemoryModel Logical GLSL450
                OpEntryPoint Fragment %main "main" %inUV %outFragColor
                OpExecutionMode %main OriginUpperLeft
                OpSource GLSL 450
                OpSourceExtension "GL_ARB_separate_shader_objects"
                OpSourceExtension "GL_ARB_shading_language_420pack"
                OpName %main "main"
                OpName %weight "weight"
                OpName %tex_offset "tex_offset"
                OpName %samplerColor "samplerColor"
                OpName %UBO "UBO"
                OpMemberName %UBO 0 "blurScale"
                OpMemberName %UBO 1 "blurStrength"
                OpName %ubo "ubo"
                OpName %result "result"
                OpName %inUV "inUV"
                OpName %i "i"
                OpName %outFragColor "outFragColor"
                OpDecorate %samplerColor DescriptorSet 0
                OpDecorate %samplerColor Binding 1
                OpMemberDecorate %UBO 0 Offset 0
                OpMemberDecorate %UBO 1 Offset 4
                OpDecorate %UBO Block
                OpDecorate %ubo DescriptorSet 0
                OpDecorate %ubo Binding 0
                OpDecorate %inUV Location 0
```

Configure the pipeline state

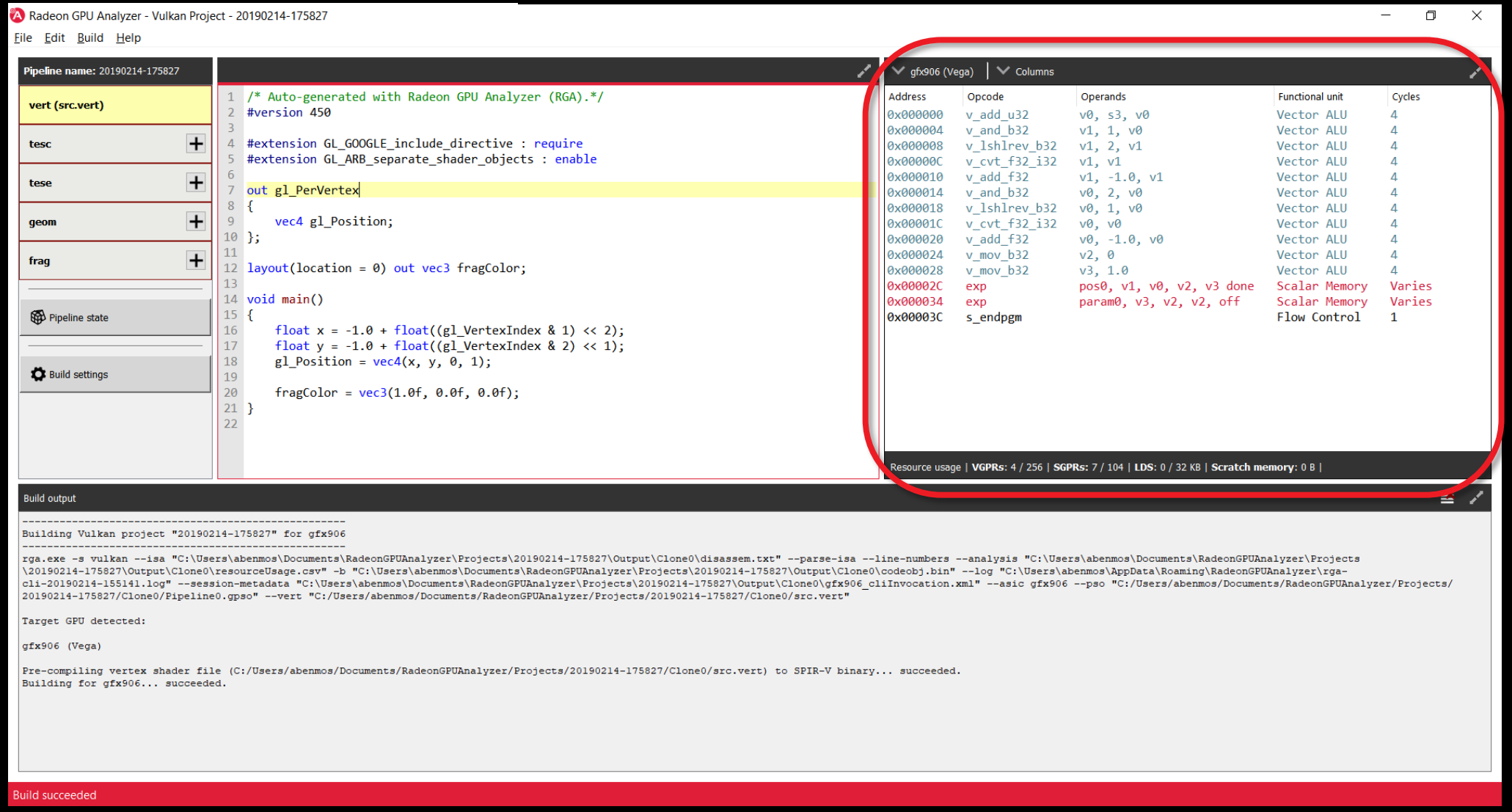
- Graphical representation of the Vulkan™ pipeline state as a tree
- Easy search and navigation
- Filter the state
- Edit the state



Configure the build settings

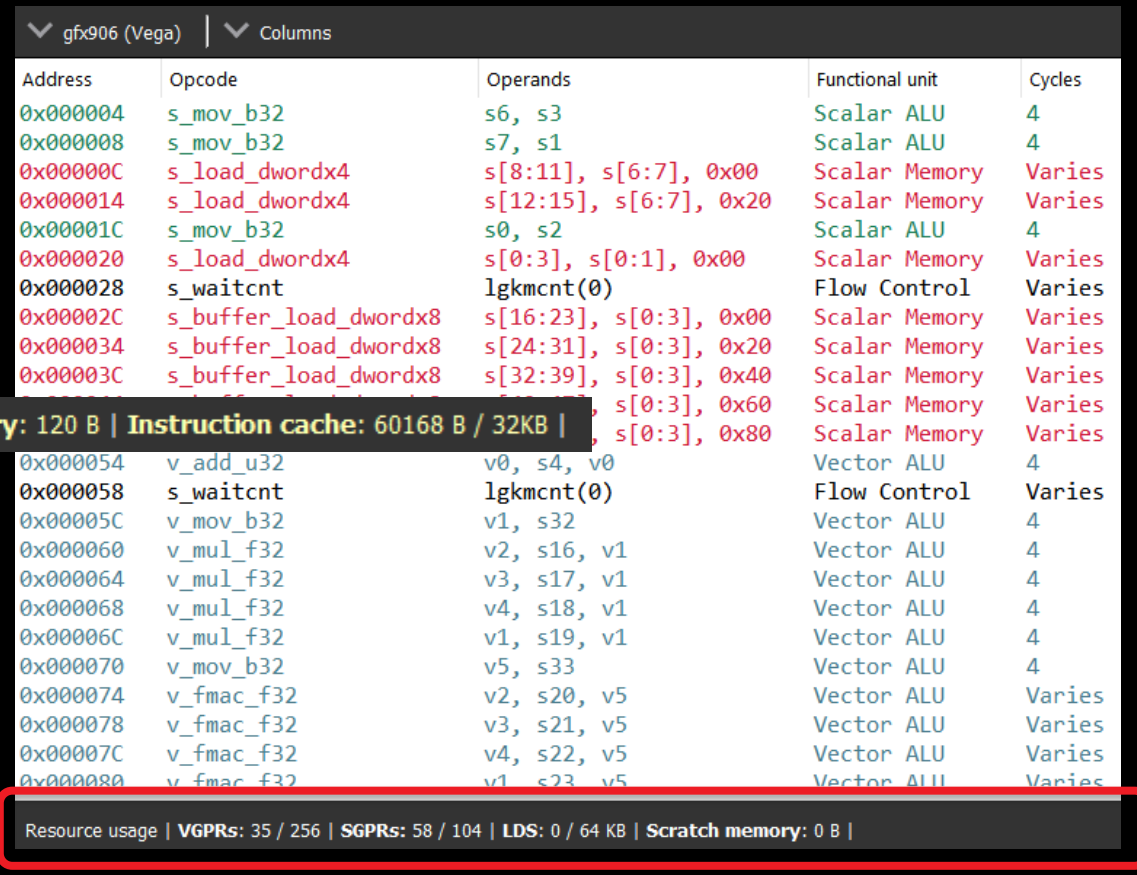


Build the pipeline



Examine the disassembly and resource usage

- Disassembly and resource usage in the same view
- Breaks down each instruction into its components
- Color coding by instruction class
- Resource usage hazards highlighted as warnings



How can I get my app's shaders into RGA?

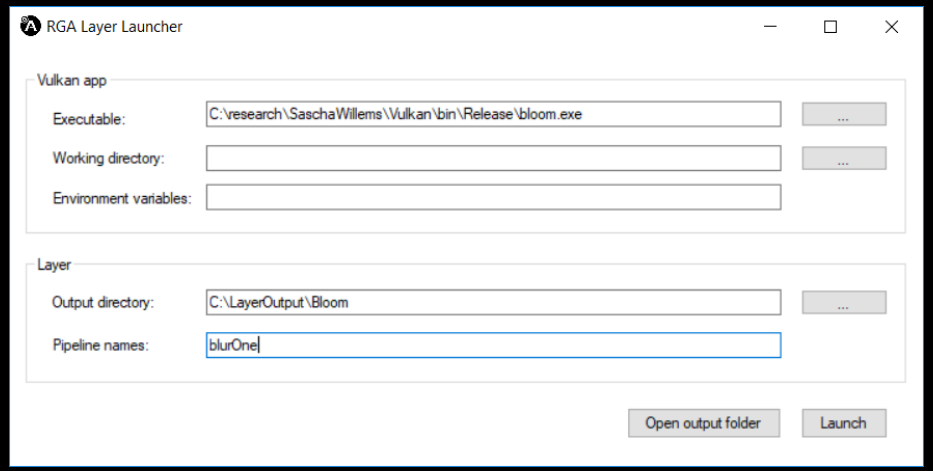
Use the RGA layer – beta feature

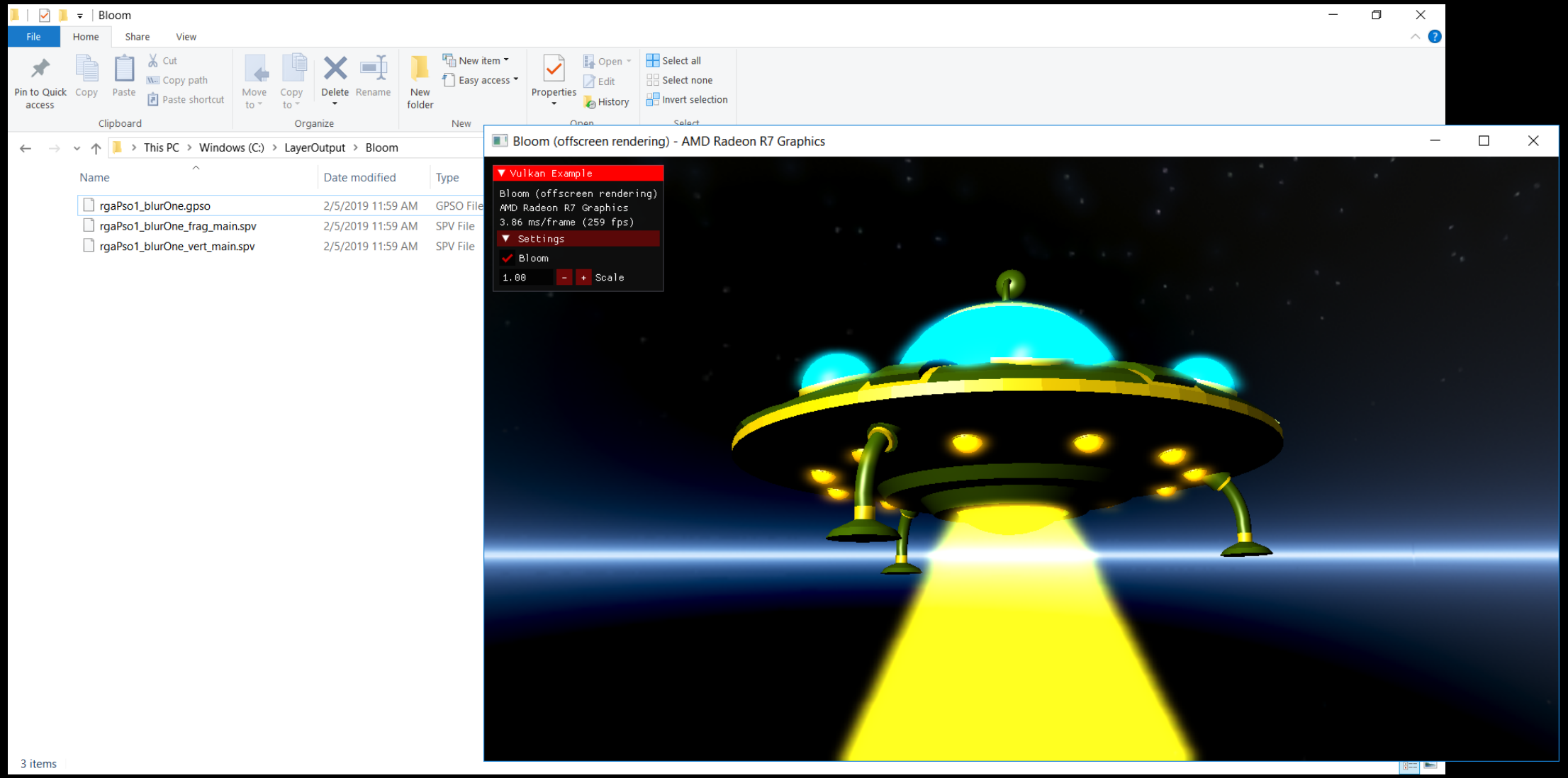
- Implicit Vulkan™ layer
- Serializes state and shaders for all pipelines
- Can intercept specific pipelines if VK_EXT_DEBUG_MARKER is used

Using the RGA Layer

Suggested RGA workflow

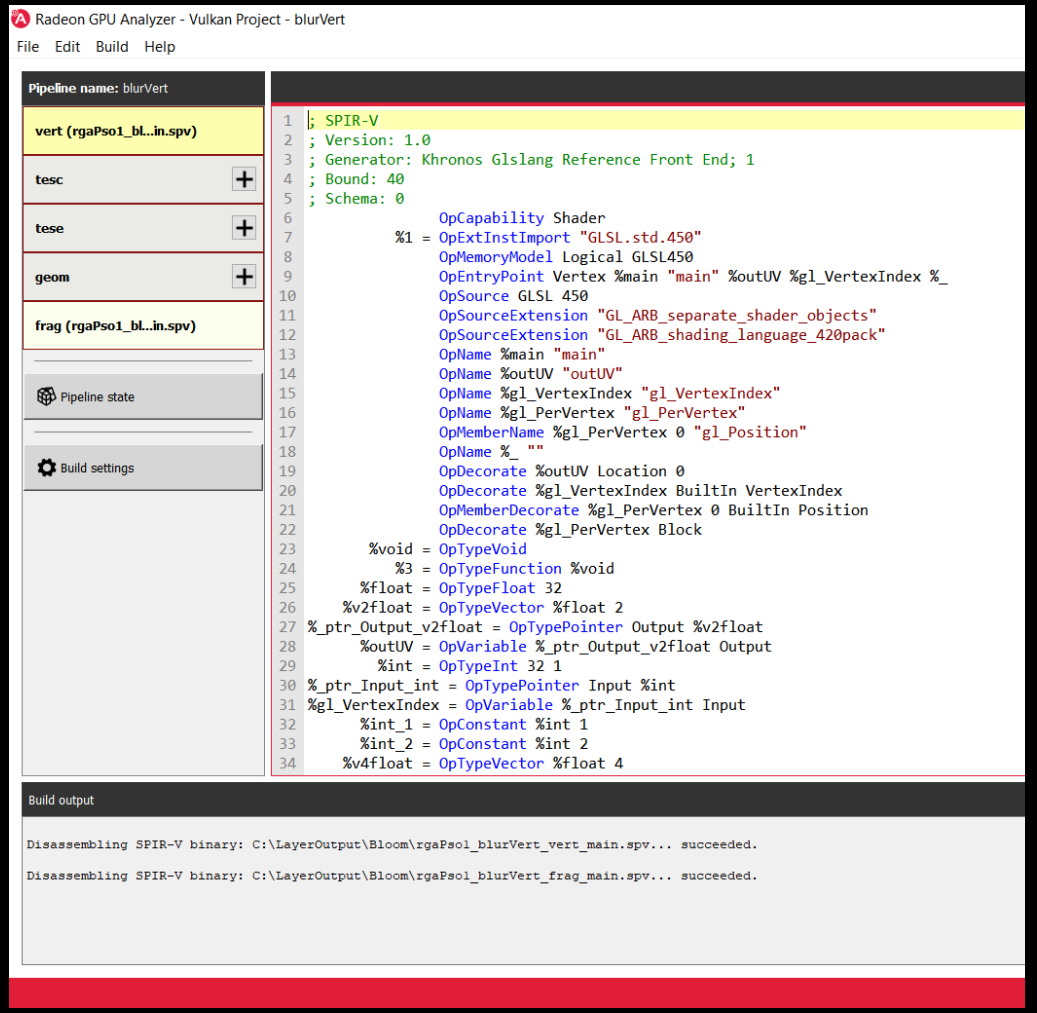
- Good idea to use VK_EXT_DEBUG_MARKER to name your objects
- Identify your bottleneck pipeline using a profiler or debugger like RenderDoc
- Suppose pipeline “blurOne” was identified as a bottleneck
- Can you locate your shaders? Just drop them in RGA
- Can’t locate shaders or don’t feel like manually editing the state?
  - Use the RGA layer



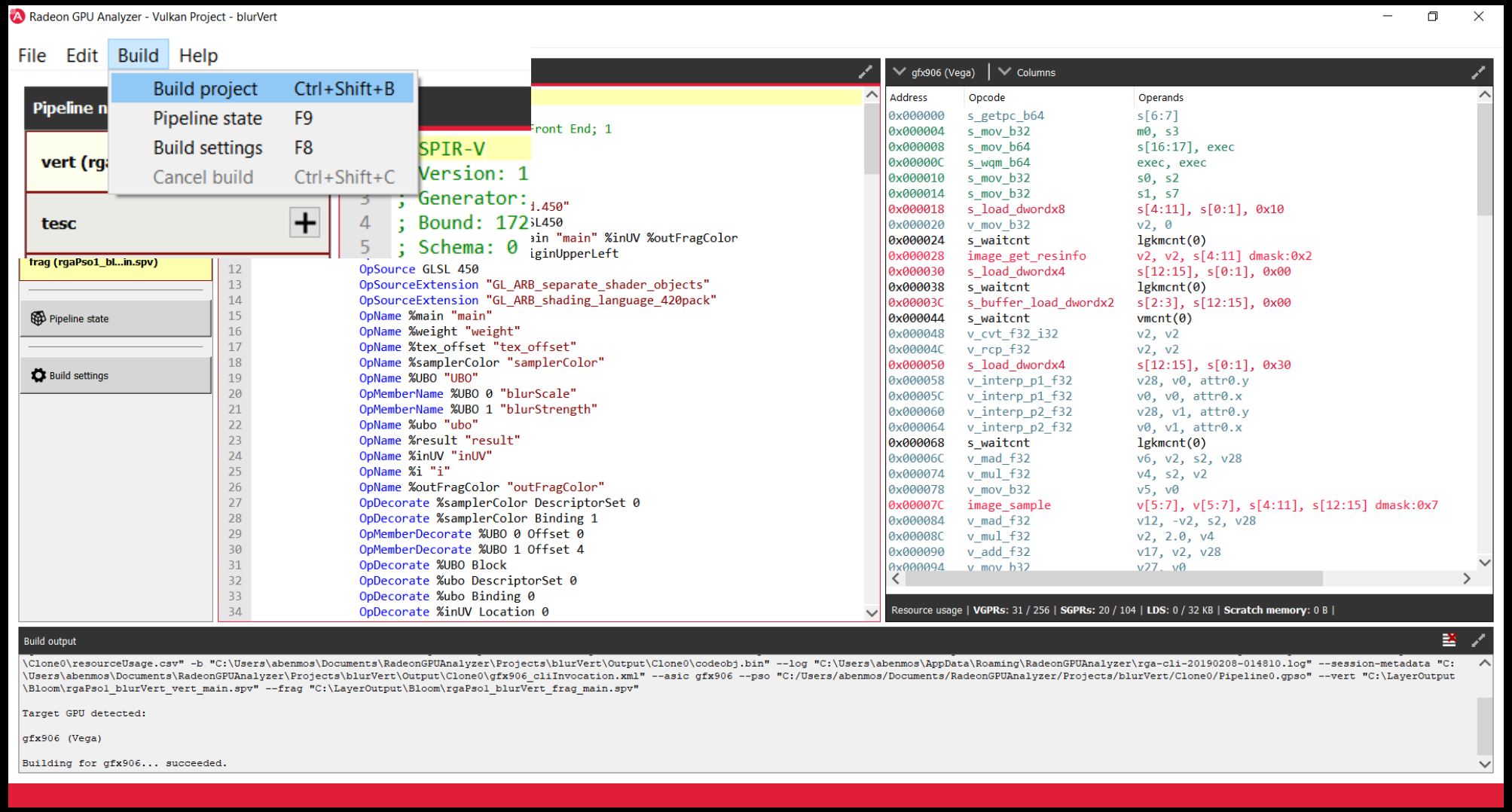

Bloom
File
Home
Share
View
Pin to Quick access
Copy
Paste
Cut
Copy path
Paste shortcut
Clipboard
Move to
Copy to
Delete
Rename
Organize
New folder
New item
Easy access
New
Properties
Open
Edit
History
Select all
Select none
Invert selection
This PC
Windows (C:)
LayerOutput
Bloom
Name
Date modified
Type
rgaPso1_blurOne.gpso
2/5/2019 11:59 AM
GPSO File
rgaPso1_blurOne_frag_main.spv
2/5/2019 11:59 AM
SPV File
rgaPso1_blurOne_vert_main.spv
2/5/2019 11:59 AM
SPV File
3 items
Bloom (offscreen rendering) - AMD Radeon R7 Graphics
Vulkan Example
Bloom (offscreen rendering)
AMD Radeon R7 Graphics
3.86 ms/frame (259 fps)
Settings
Bloom
1.00
Scale

Drop serialized files in RGA

Name	Date modified	Type
rgaPso1_blurOne.gpso	2/5/2019 11:59 AM	GPSO File
rgaPso1_blurOne_frag_main.spv	2/5/2019 11:59 AM	SPV File
rgaPso1_blurOne_vert_main.spv	2/5/2019 11:59 AM	SPV File



Build, tweak GLSL / SPIR-V, tweak pipeline state, repeat!



RGA for Vulkan – key takeaways

- Compile through the live-driver
- Edit GLSL source code or SPIR-V binaries
- Edit Vulkan pipeline state with graphical editor
- Extract state/shaders from apps with the layer – beta feature
- Highlights VMEM, SMEM instructions, and excessive HW resource usage
- Runs on non-AMD configurations as well

Thank you! Questions?

Information

GPUOpen: https://gpuopen.com/

RGP: https://gpuopen.com/gaming-product/radeon-gpu-profiler-rgp/

RGA: https://gpuopen.com/gaming-product/radeon-gpu-analyzer-rga/

RenderDoc: https://renderdoc.org

Microsoft® PIX: https://blogs.msdn.microsoft.com/pix

Downloads

RGP: https://github.com/GPUOpen-Tools/RGP/releases

RGA: https://github.com/GPUOpen-Tools/RGA/releases

Acknowledgements

Baldur Karlsson for assistance with RenderDoc-RGP and RGA interop

Gregory Mitrano for RGP content

Contact

Rodrigo.Urra@amd.com

Amit.Ben-Moshe@amd.com

Disclaimer and Attribution