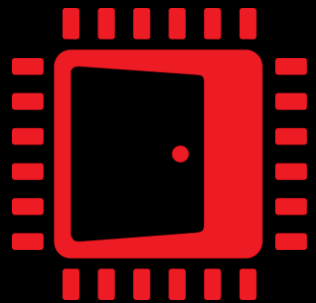


GDC



AMD 
GPUOpen

GAME OPTIMIZATION WITH THE AMD RADEON™ DEVELOPER TOOL SUITE

CHRIS HESIK (AMD)

AMIT MULAY (AMD)

AMD 
together we advance_

AMD

RADEON

Developer Tool Suite

Agenda

- Overview of the tools and new features
- Introduction to GPU Reshape

AMD RADEON Developer Panel

Local - Radeon Developer Panel v2.12.0.20

CONNECTION SYSTEM APPLICATIONS

My_Game_DX12.exe
HybridShadows_DX12d....
Dropbox.exe
D3D12nBodyGravity.exe
D3D12Multithreading.exe
computenbody.exe

Global workflow: Profiling

Profiling Device Clocks

atus: Offli

Capture profile

Hotkey Ctrl+Alt+C

☒ Enable instruction tracing (can affect performance)
☒ Collect counters
☐ Delay capture

Recently collected profiles

Path: C:\User\your_name\Documents\rgp_profiles

Name	Size	Date Modified
My_Game_DX12-20240202-184353.rgp	88.072 MB	2/2/2024 6:43 PM
D3D12nBodyGravity-20240202-184237.rgp	81.794 MB	2/2/2024 6:42 PM
computenbody-20240202-183945.rgp	4.728 MB	2/2/2024 6:39 PM
My_Game_DX12-20240202-183410.rgp	301.190 MB	2/2/2024 6:34 PM
D3D12Multithreading-20240202-185151.rgp	7.409 MB	12/1/2020 6:51 PM
D3D12Multithreading-20240202-185043.rgp	6.869 MB	12/1/2020 6:50 PM

Open

AMD RADEON GPU Profiler



AMD RADEON GPU Analyzer

Radeon GPU Analyzer - OpenCL Project - BinarySearch

File Edit Build Help

Project name: BinarySearch

BinarySearch_Kernels.cl

binarySearch
binarySearch_mulkeys
binarySearch...sConcurrent

+ Add file
Create file
Build settings

```

16  /**
17  * One instance of this kernel call is a thread.
18  * Each thread finds out the segment in which it should look for the element.
19  * After that, it checks if the element is between the lower bound and upper bound
20  * of its segment. If yes, then this segment becomes the total searchspace for the next pass.
21  *
22  * To achieve this, it writes the lower bound and upper bound to the output array.
23  * In case the element at the left end (lower bound) matches the element we are looking for,
24  * That is marked in the output and we no longer need to look any further.
25  */
26  kernel void
27  binarySearch(
28      __global uint4 * outputArray,
29      __const __global uint2 * sortedArray,
30      __const unsigned int findMe)
31  {
32      unsigned int tid = get_global_id(0);
33
34      /* Then we find the elements for this thread */
35      uint2 element = sortedArray[tid];
36
37      /* If the element to be found does not lie between them, then nothing left to do in this thread
38      if( (element.x > findMe) || (element.y < findMe))
39      {
40          return;
41      }
42      else
43      {
44          /* However, if the element does lie between the lower and upper bounds of this thread's see
45          * we need to narrow down the search further in this search space
46          */
47
48          /* The search space for this thread is marked in the output as being the total search space
49          outputArray[tid].x = findMe;
50          outputArray[tid].y = 1;
51      }
52  }
53  }
54
55  kernel void
56  
```

Address	Opcode	Operands	VGPR pressure (used/8/256)
0x002500	s_clause	0x3	2
0x002504	s_load_b32	s3, s[0:1], 0x24	2
0x00250C	s_load_b64	s[0:9], s[0:1], 0x40	2
0x002514	s_load_b128	s[4:7], s[0:1], null	2
0x00251C	s_load_b32	s0, s[0:1], 0x10	2
0x002524	s_mov_b32	s33, 0	2
0x002528	s_waitcnt	lgkcnt(0)	2
0x00252C	s_and_b32	s3, s3, 0xffff	2
0x002534	s_delay_alu	instid0(SALU_CYCLE_1) instskip(SKIP_1) instid1(VALU_DEP_2)	2
0x002538	v_mov_u64_u32	v[2:3], null, s2, s3, v[0:1]	3
0x002540	v_mov_b32_e32	v1, 0	2
0x002544	v_add_co_u32	v0, null, s0, v2	2
0x00254C	s_delay_alu	instid0(VALU_DEP_1) instskip(NEXT) instid1(VALU_DEP_1)	2
0x002550	v_lshrev_b64	v[2:3], s, v[0:1]	4
0x002558	v_add_co_u32	v2, vcc_lo, s6, v2	4
0x002560	s_delay_alu	instid0(VALU_DEP_2) instskip(SKIP_4) instid1(VALU_DEP_1)	4
0x002564	v_add_co_ci_u32_e32	v3, vcc_lo, s7, v3, vcc_lo	4
0x002568	global_load_b64	v[2:3], v[2:3], off	4
0x002570	s_waitcnt	vmcnt(0)	4
0x002574	v_cmp_ge_u32_e32	vcc_lo, s0, v2	4
0x002578	v_cmp_le_u32_e64	s0, s0, v3	3
0x002580	s_and_b32	s0, vcc_lo, s0	2
0x002584	s_delay_alu	instid0(SALU_CYCLE_1)	2
0x002588	s_and_saveexec_b32	s1, s0	2
0x00258C	s_branch_execz	10	2
0x002590	s_load_b128	s[0:3], s[4:5], null	2
0x002598	s_mov_b32	s6, 1	2
0x00259C	s_waitcnt	lgkcnt(0)	2
0x0025A0	v_dual_mov_b32	v5, s3 :: v_dual_mov_b32 v4, s2	3
0x0025A8	v_dual_mov_b32	v3, s1 :: v_dual_mov_b32 v2, s0	3
0x0025B0	v_dual_mov_b32	v2, v0 :: v_dual_mov_b32 v5, s6	6
0x0025B8	global_store_b128	v1, v[2:5], s[4:5]	5
0x0025C0	s_nop	0	0
0x0025C4	s_sendmsg	sendmsg(MSG_DEALLOC_VGPRS)	0
0x0025C8	s_endpgm		0

Resource usage | VGPRs: 6 / 256 | SGPRs: 36 / 106 | LDS: 0 / 64 KB | Scratch memory: 0 B

Build output

```

Building OpenCL project "240202-145613" for gfx1103
-----
rga.exe -s opencl --isa "C:\Users\apumodak\Documents\RadeonGPUAnalyzer\Projects\240202-145613\Output\Clone0\disassem.txt" --parse-isa --livevereg "C:\Users\apumodak\Documents\RadeonGPUAnalyzer\Projects\240202-145613\Output\Clone0\livevereg.txt" --line-numbers --analysis "C:\Users\apumodak\Documents\RadeonGPUAnalyzer\Projects\240202-145613\Output\Clone0\resourceUsage.csv" -b "C:\Users\apumodak\Documents\RadeonGPUAnalyzer\Projects\240202-145613\Output\Clone0\codeobj.bin" --log "C:\Users\apumodak\AppData\Roaming\RadeonGPUAnalyzer\rga-cli-20240202-145643.log" --session-metadata "C:\Users\apumodak\Documents\RadeonGPUAnalyzer\Projects\240202-145613\Output\Clone0\gfx1103_cliInvocation.xml" --asic gfx1103 "C:/amd/BinarySearch_Kernels.cl"

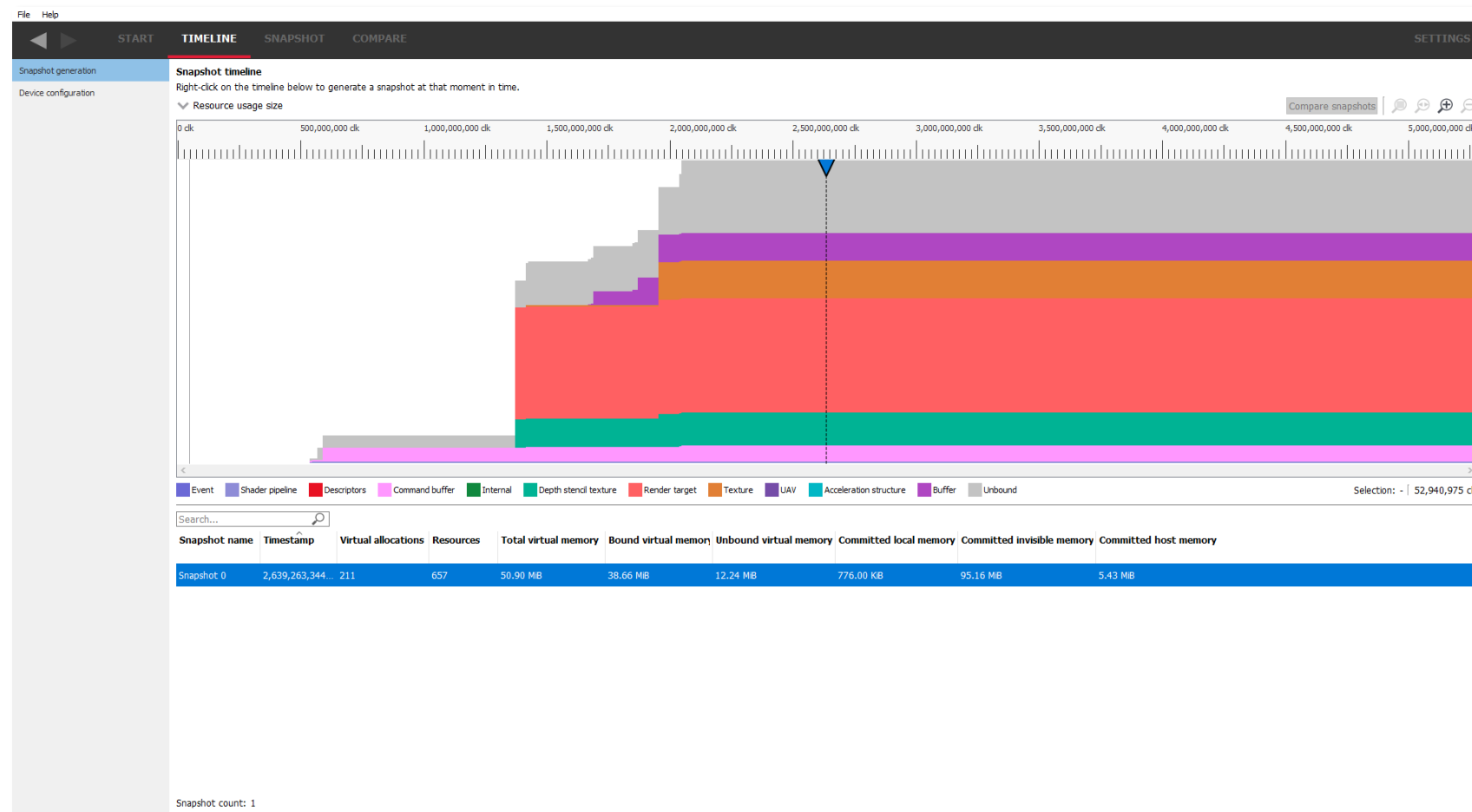
Target GPU detected:
gfx1103 (RDNA3)
AMD Radeon 740M Graphics
AMD Radeon 760M Graphics
AMD Radeon 780M Graphics

Building for gfx1103... succeeded.
Extracting ISA for gfx1103... succeeded.
Performing live vgpr analysis for gfx1103 for kernel binarySearch... succeeded.
Performing live vgpr analysis for gfx1103 for kernel binarySearch_mulkeys... succeeded.
Performing live vgpr analysis for gfx1103 for kernel binarySearch...sConcurrent... succeeded.
Extracting statistics... succeeded.

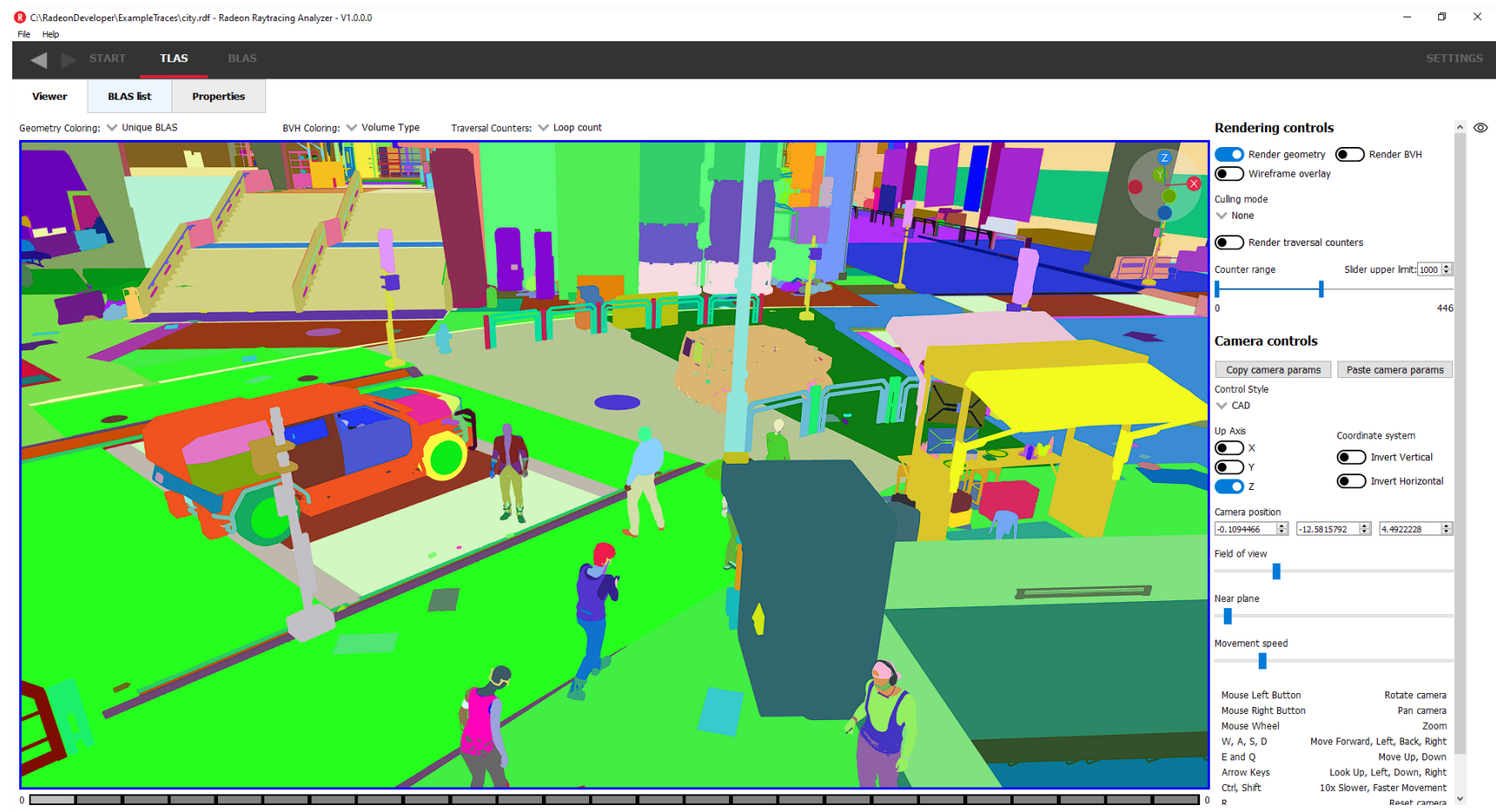
Build succeeded

```

AMD RADEON Memory Visualizer



AMD RADEON Raytracing Analyzer



AMD RADEON GPU Detective

```

My_Game_DX12-20231116-170357-case22.txt - Notepad
File Edit Format View Help

=====
PAGE FAULT SUMMARY
=====

Offending VA: 0x237bb2000

Resource timeline
=====

Legend
=====
<> denotes paired resources: in certain cases, a resource (heap or buffer) is created implicitly with another resource by the runtime or the driver. When the tool detects this situation, th
Timestamp format: HH:MM:SS.clk_cycles (clk_cycles per second: 10,000,000)

Timestamp      Event type      Resource type      Resource identifier      Resource size      Resource name
-----
00:00:00.3670688 Create      <Heap, Buffer>      <0x7c62d7b600000151, 0x285183e100000152> 671088640 (640.00 MB) StaticBufferPoolDX12::m_pSysMemBuffer
00:00:00.3698656 Bind        <Heap, Buffer>      <0x7c62d7b600000151, 0x285183e100000152> 671088640 (640.00 MB) StaticBufferPoolDX12::m_pSysMemBuffer
00:00:00.3698656 Make Resident <Heap, Buffer>      <0x7c62d7b600000151, 0x285183e100000152> 671088640 (640.00 MB) StaticBufferPoolDX12::m_pSysMemBuffer
00:00:00.8157920 Destroy     <Heap, Buffer>      <0x7c62d7b600000151, 0x285183e100000152> 671088640 (640.00 MB) StaticBufferPoolDX12::m_pSysMemBuffer
00:00:02.2023168 Create      <Heap, Image>      <0xf13ab31800000a7e, 0x326ad6f600000a7f> 16810352 (16.03 MB) Temporary texture released before use
00:00:02.2026624 Bind        <Heap, Image>      <0xf13ab31800000a7e, 0x326ad6f600000a7f> 16810352 (16.03 MB) Temporary texture released before use
00:00:02.2026624 Make Resident <Heap, Image>      <0xf13ab31800000a7e, 0x326ad6f600000a7f> 16810352 (16.03 MB) Temporary texture released before use
00:00:02.2044448 Destroy     <Heap, Image>      <0xf13ab31800000a7e, 0x326ad6f600000a7f> 16810352 (16.03 MB) Temporary texture released before use

Associated resources
=====
Resource id: <0x7c62d7b600000151, 0x285183e100000152>
  Type: <Heap, Buffer>
  Name: StaticBufferPoolDX12::m_pSysMemBuffer
  Virtual address:
    0x236bc0000 [size: 671088640 (640.00 MB), parent address + offset: 0x236bc0000 + 0x0, preferred heap: Host (CPU memory)]
  Commit type: COMMITTED
  Attributes (Buffer):
    Create flags: None
    Usage flags: None
  Resource timeline:
    00:00:00.3670688 : Create
    00:00:00.3698656 : Bind into 0x236bc0000
    00:00:00.3698656 : Make Resident into 0x236bc0000
    00:00:00.8157920 : Destroy

Resource id: <0xf13ab31800000a7e, 0x326ad6f600000a7f>
<

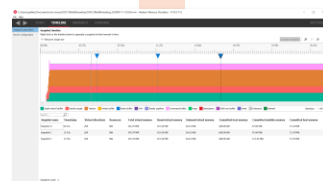
```


WHAT'S NEW?



RRA 1.2

UI Improvements to traversal view er, BLAS tab, TLAS view er pane, and traversal mode view er



RMV 1.6

Improved history resource table



RGV 1.15

Redesigned ISA disassembly view
Initial work graph support



RGV 1.16

ISA disassembly view improvements
Quality of life improvements



RRA 1.3

Ray Visualization feature
Persistent UI state



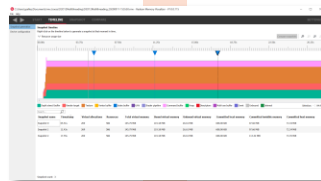
RGD 1.1

Vulkan® Support



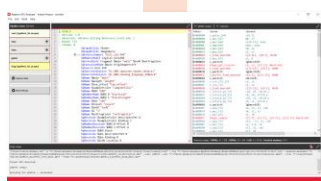
RRA 1.4

Ray Direction Visualization



RMV 1.7

Aliased Resource improvements
Loading of RGD crash dump files
Resource name and implicit buffer fixes



RGA 2.8

Support for AMD Radeon™ RX 7800/7700 XT
VGPR pressure GUI
AMD offline LLVM pipeline compiler



RGV 2.0

Redesigned Wavefront occupancy UI
Dark mode support
Raytracing pipeline thread divergence



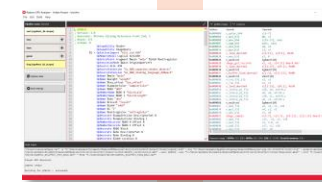
RMV 1.8

Improvements to Resource Usage size timeline



RGD 1.0

DirectX® 12 Support



RGA 2.9

Analyze pre-compiled Code Object binaries.
Analyze pre-compiled HIP binaries for the MI-200 architecture

GDC 2024

GDC 2023

March

April

May

June

July

Aug

Sept

Oct

Nov

Dec

AMD

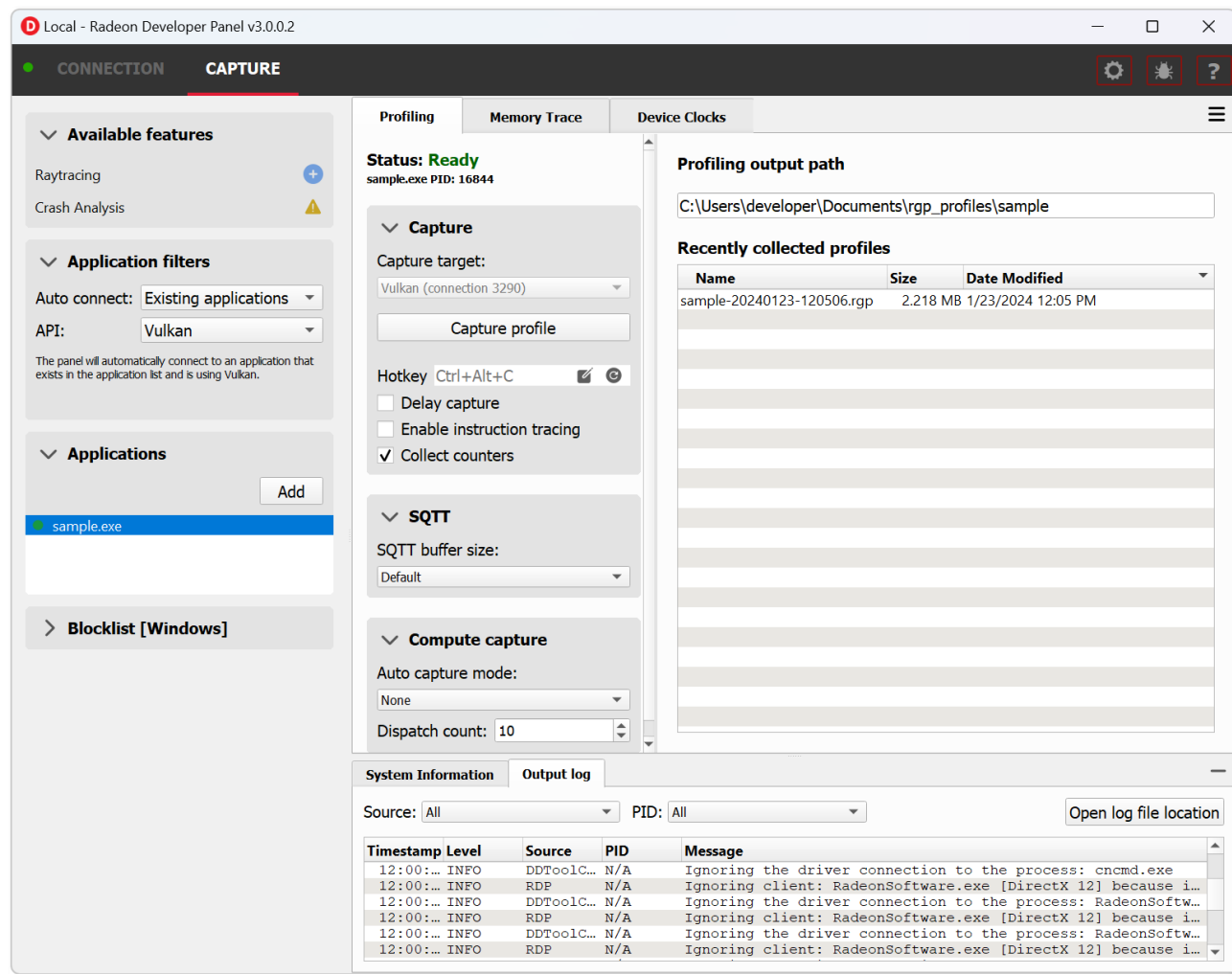
RADEON

Developer Panel

PREVIEW: AMD RADEON DEVELOPER PANEL 3.0

Redesigned user interface offers

- Improved experience for new users
- Simplified user workflows for setting capture options
- Persistence of all settings across invocations of the panel
- Support for new features without increasing complexity



AMD

RADEON

GPU Profiler

RGP 1.16 WAVEFRONT OCCUPANCY LAYOUT

Previous layout

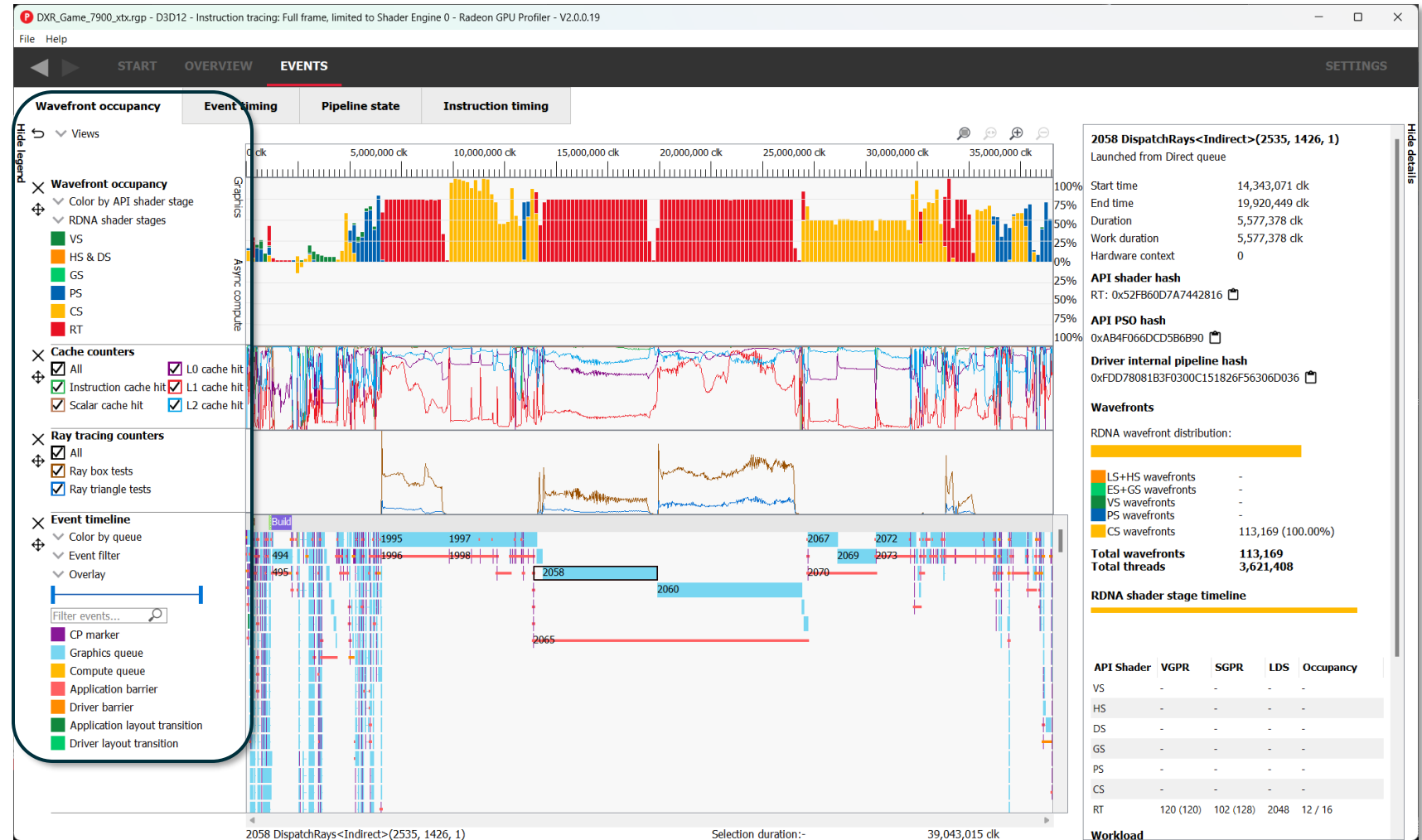
- UI controls above each row
- Legends below each row



RGP 2.0 WAVEFRONT OCCUPANCY LAYOUT

New layout

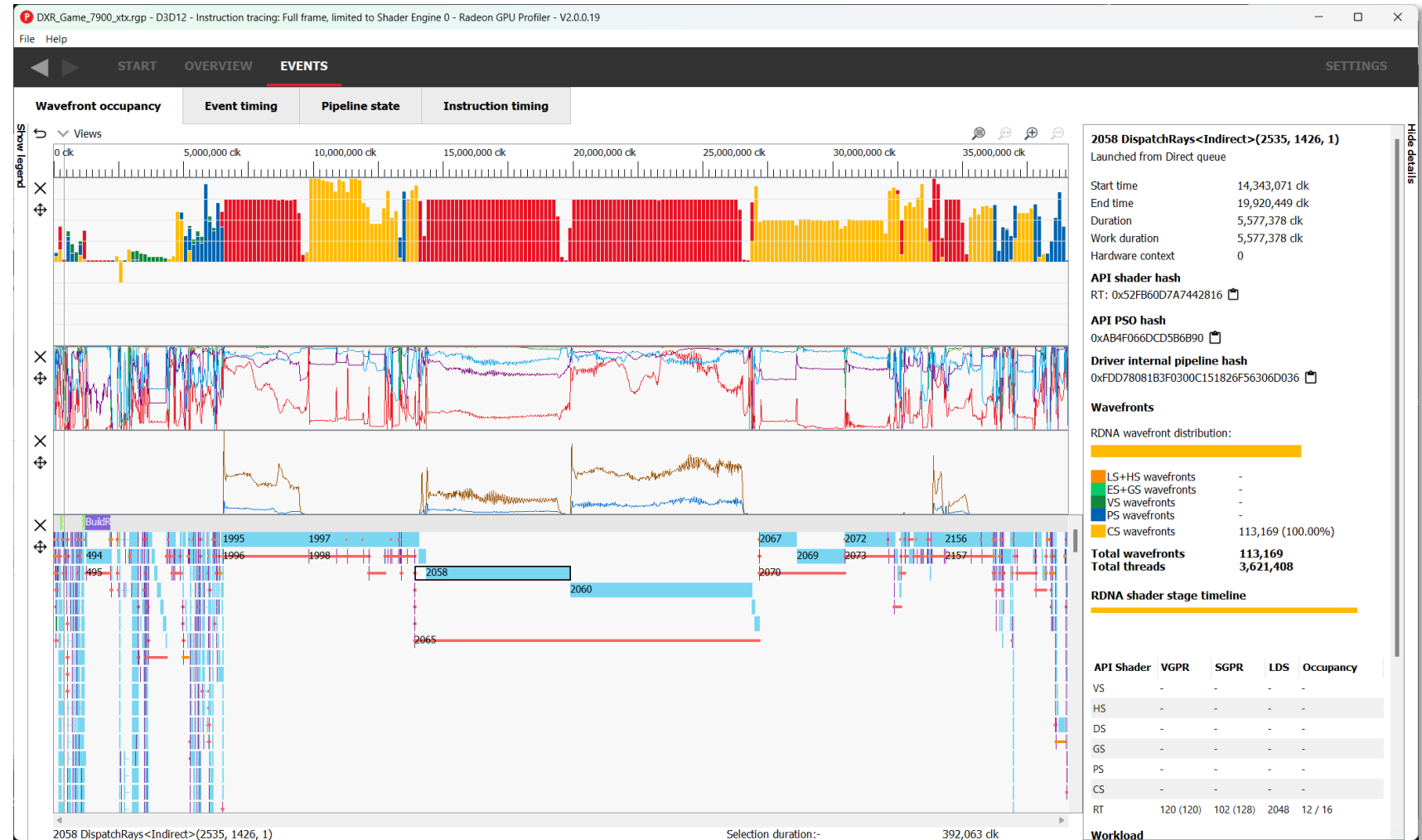
- UI controls and Legends are moved to the left of each row
- More vertical screen real estate allocated to the data views



RGP 2.0 WAVEFRONT OCCUPANCY LAYOUT

New layout

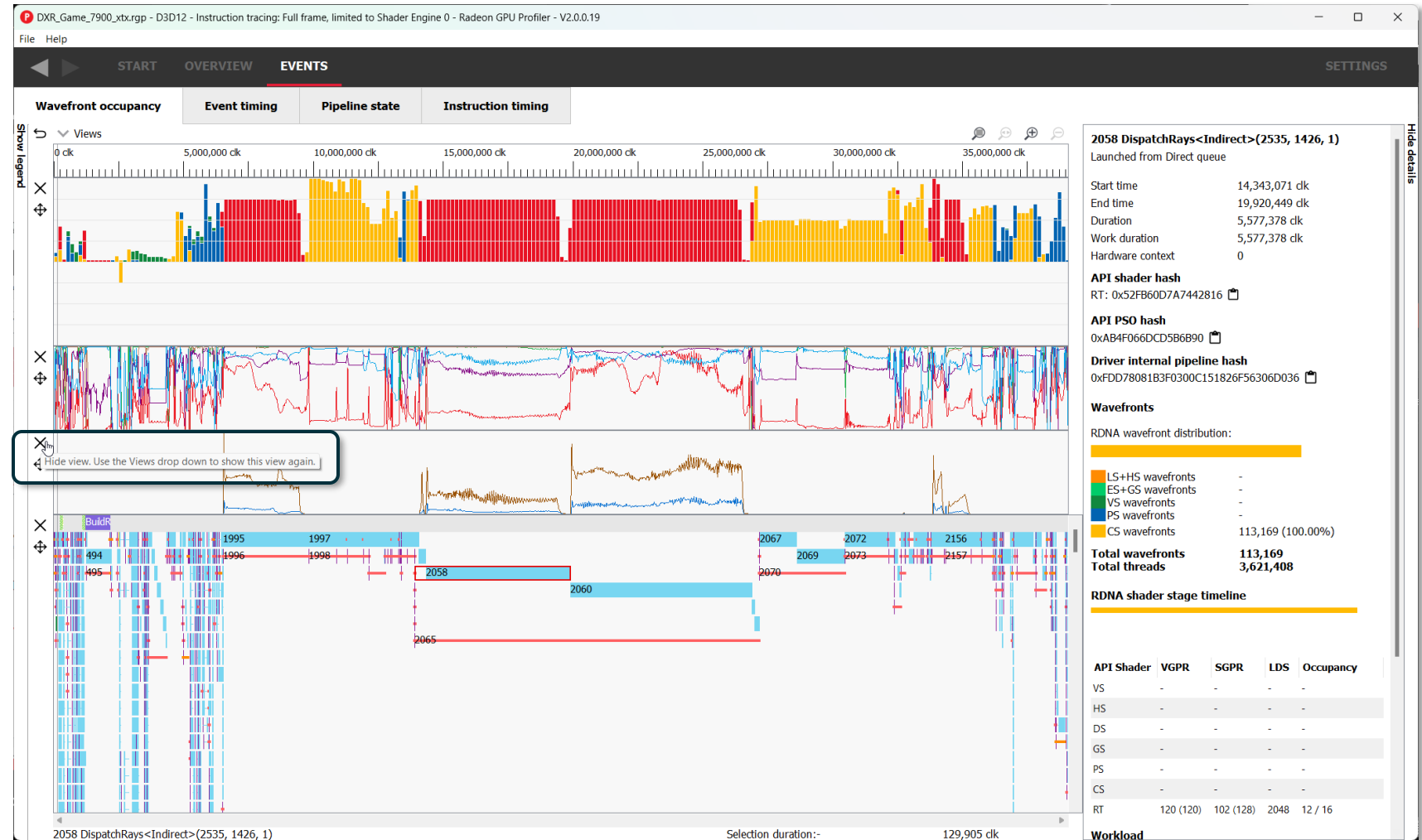
- The new left hand side panel can be hidden



RGP – CUSTOMIZE WAVEFRONT OCCUPANCY LAYOUT

New layout

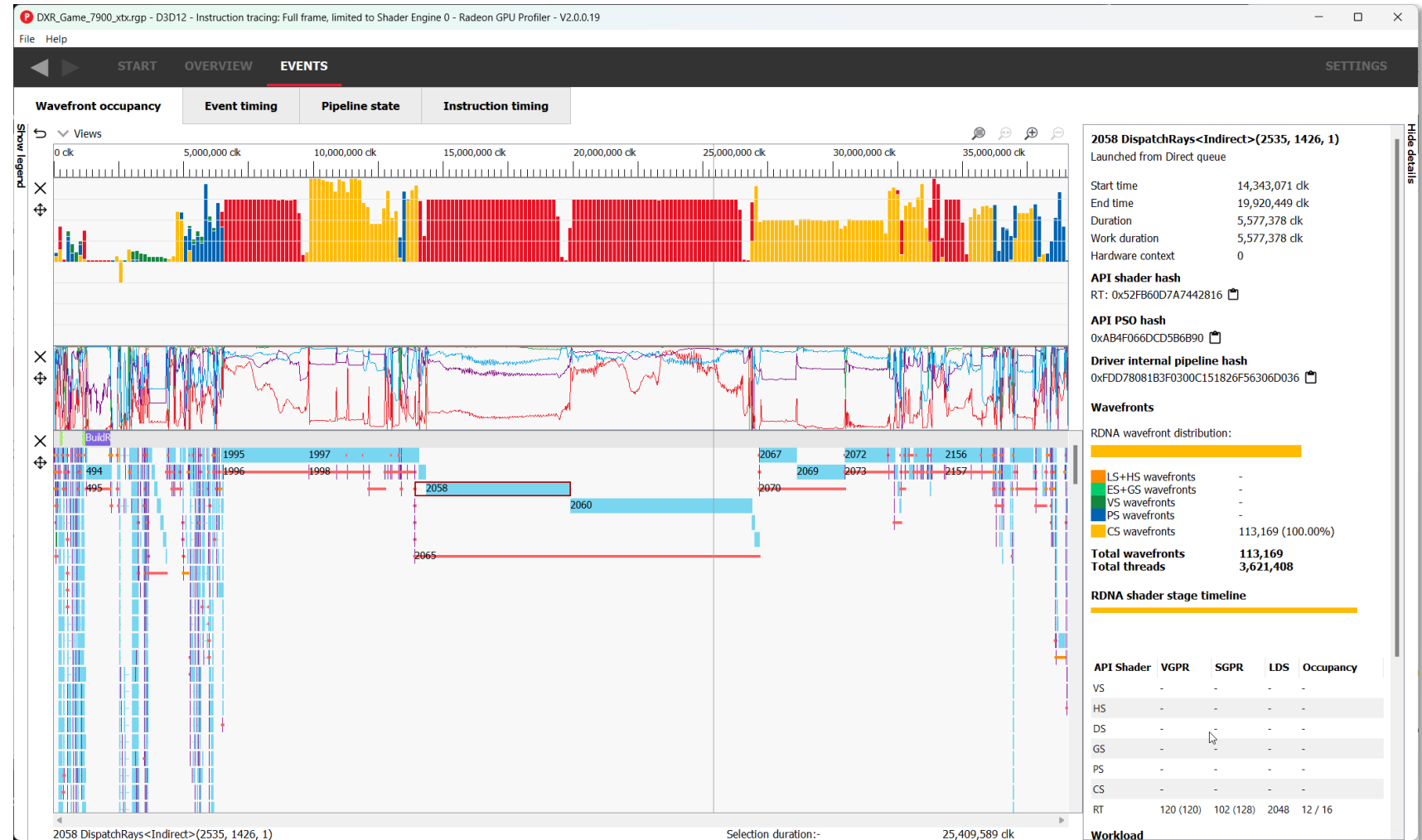
- Individual rows can be hidden



RGP – CUSTOMIZE WAVEFRONT OCCUPANCY LAYOUT

New layout

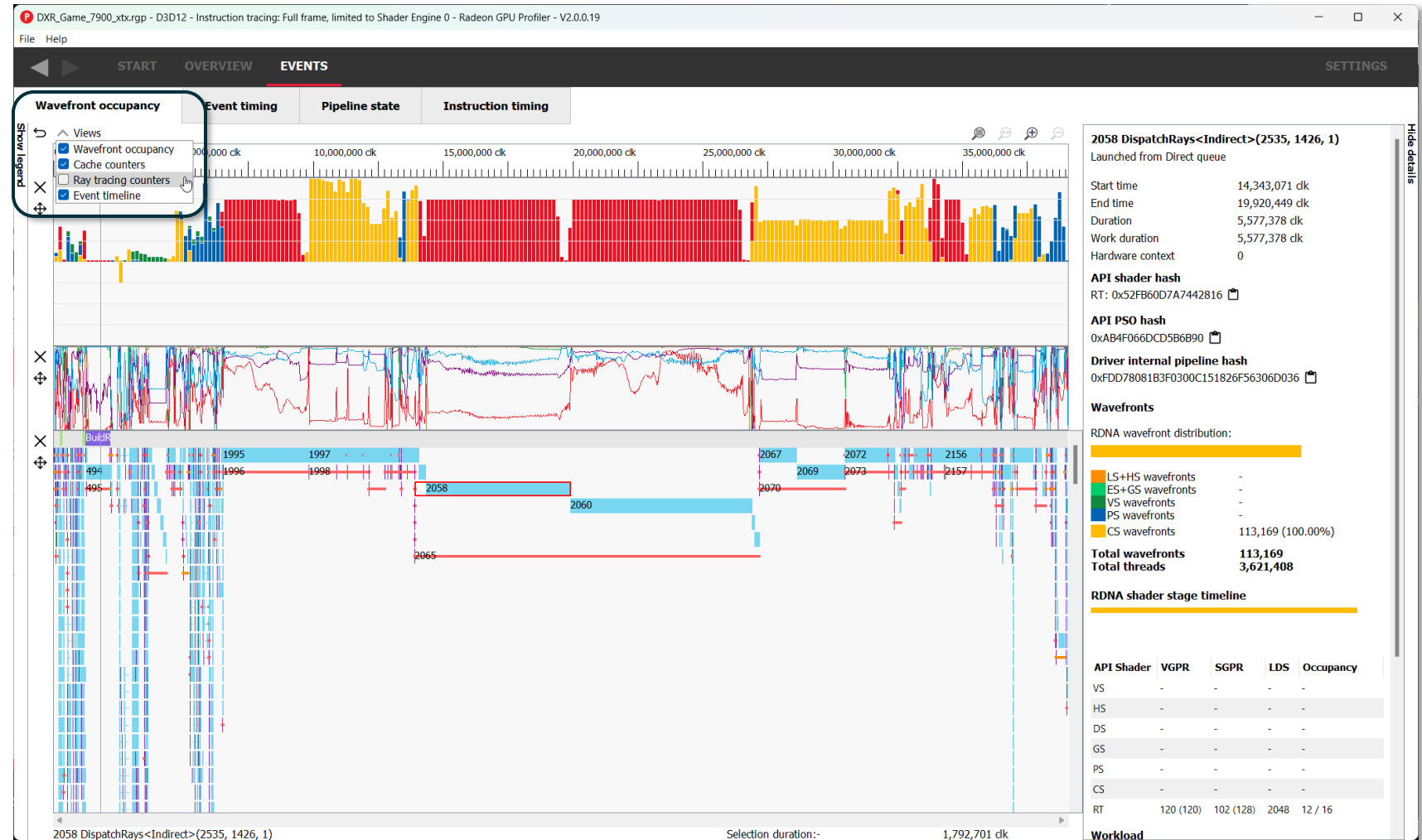
- Individual rows can be hidden
- Raytracing counters hidden



RGP – CUSTOMIZE WAVEFRONT OCCUPANCY LAYOUT

New layout

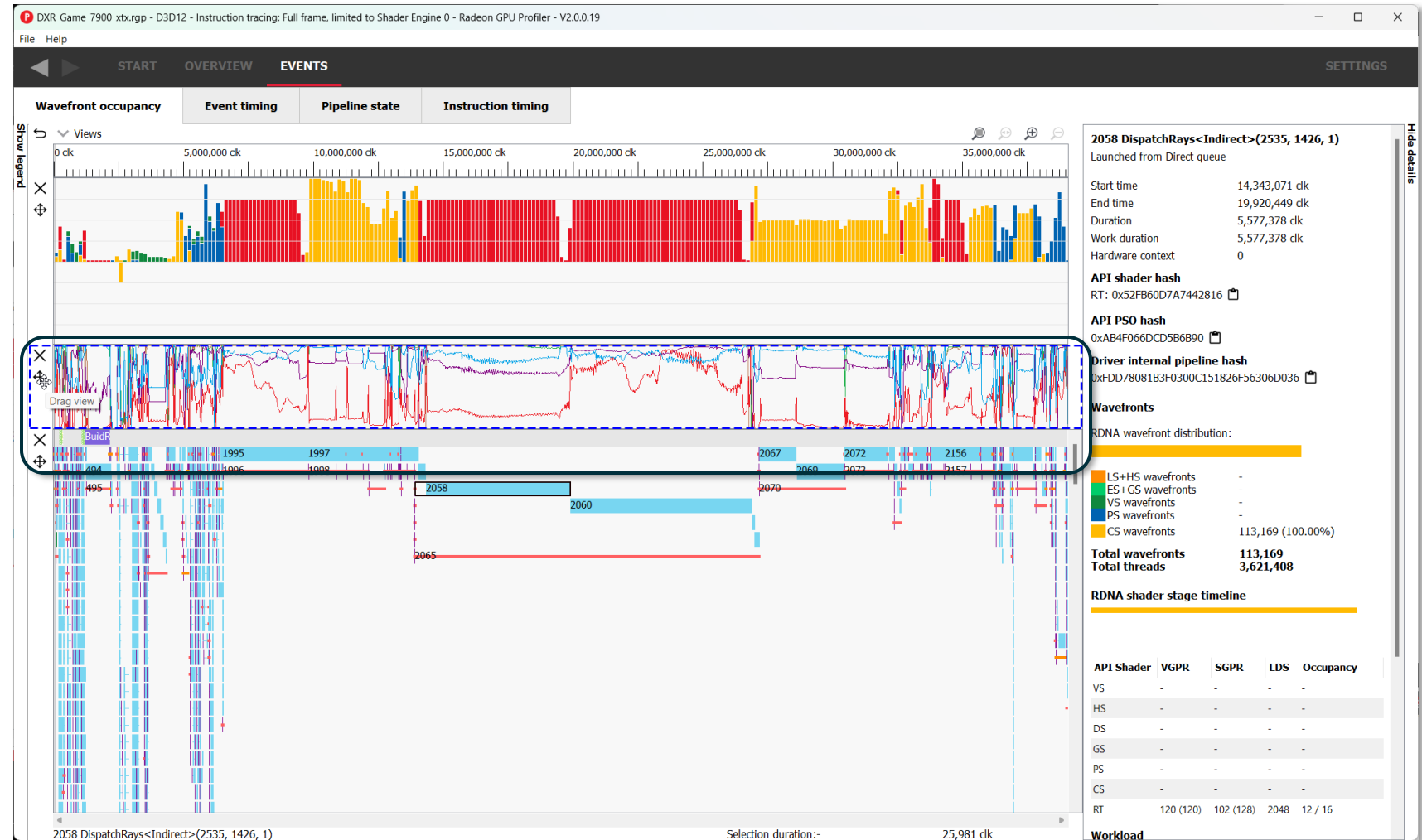
- Individual rows can be hidden
- Raytracing counters hidden
- Hidden rows can be shown again



RGP – CUSTOMIZE WAVEFRONT OCCUPANCY LAYOUT

New layout

- The position of individual rows can be changed



RGP – CUSTOMIZE WAVEFRONT OCCUPANCY LAYOUT

New layout

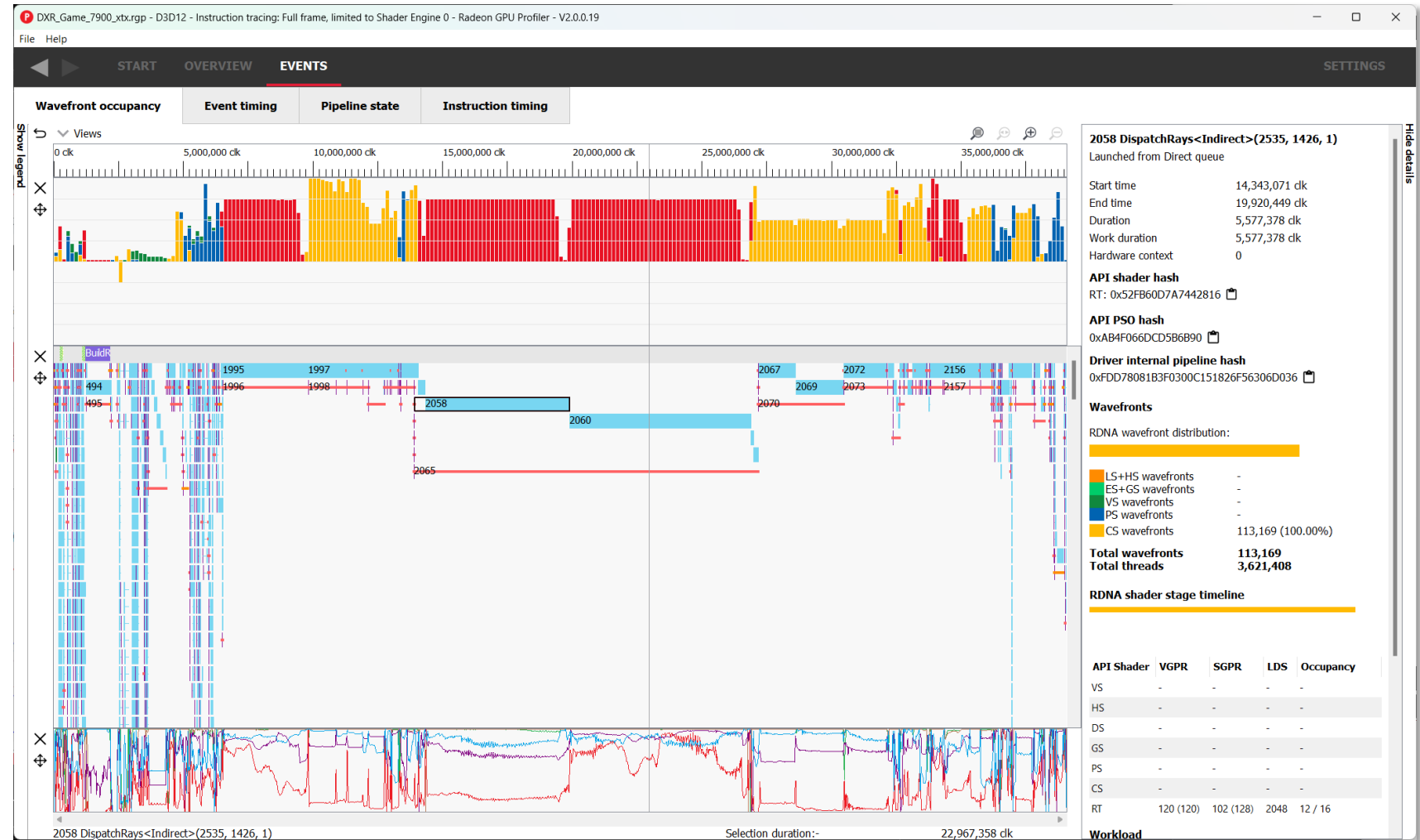
- The position of individual rows can be changed
- Click and drag a view to reposition it



RGP – CUSTOMIZE WAVEFRONT OCCUPANCY LAYOUT

New layout

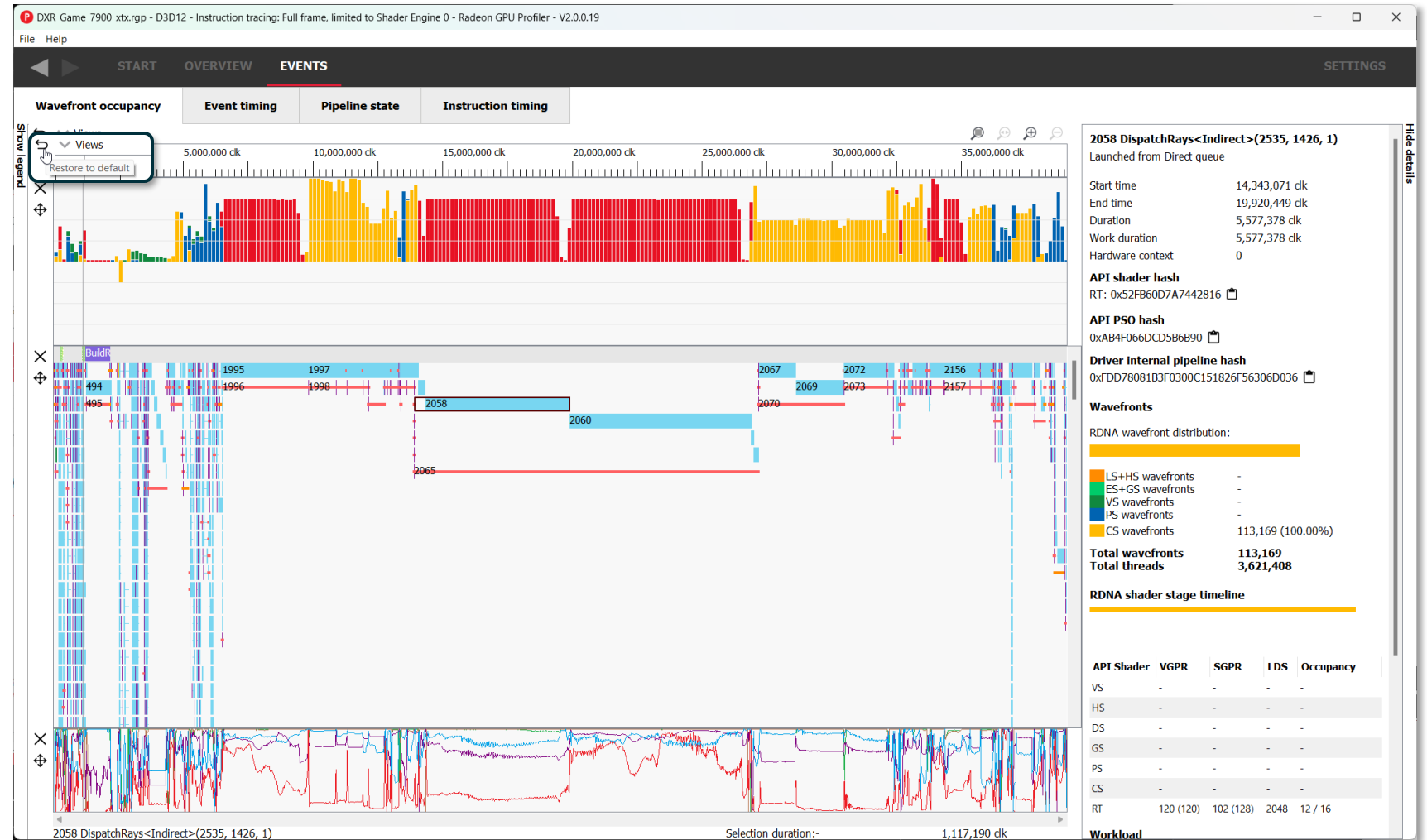
- The position of individual rows can be changed
- Click and drag a view to reposition it
- Drop it in the new position



RGP – CUSTOMIZE WAVEFRONT OCCUPANCY LAYOUT

New layout

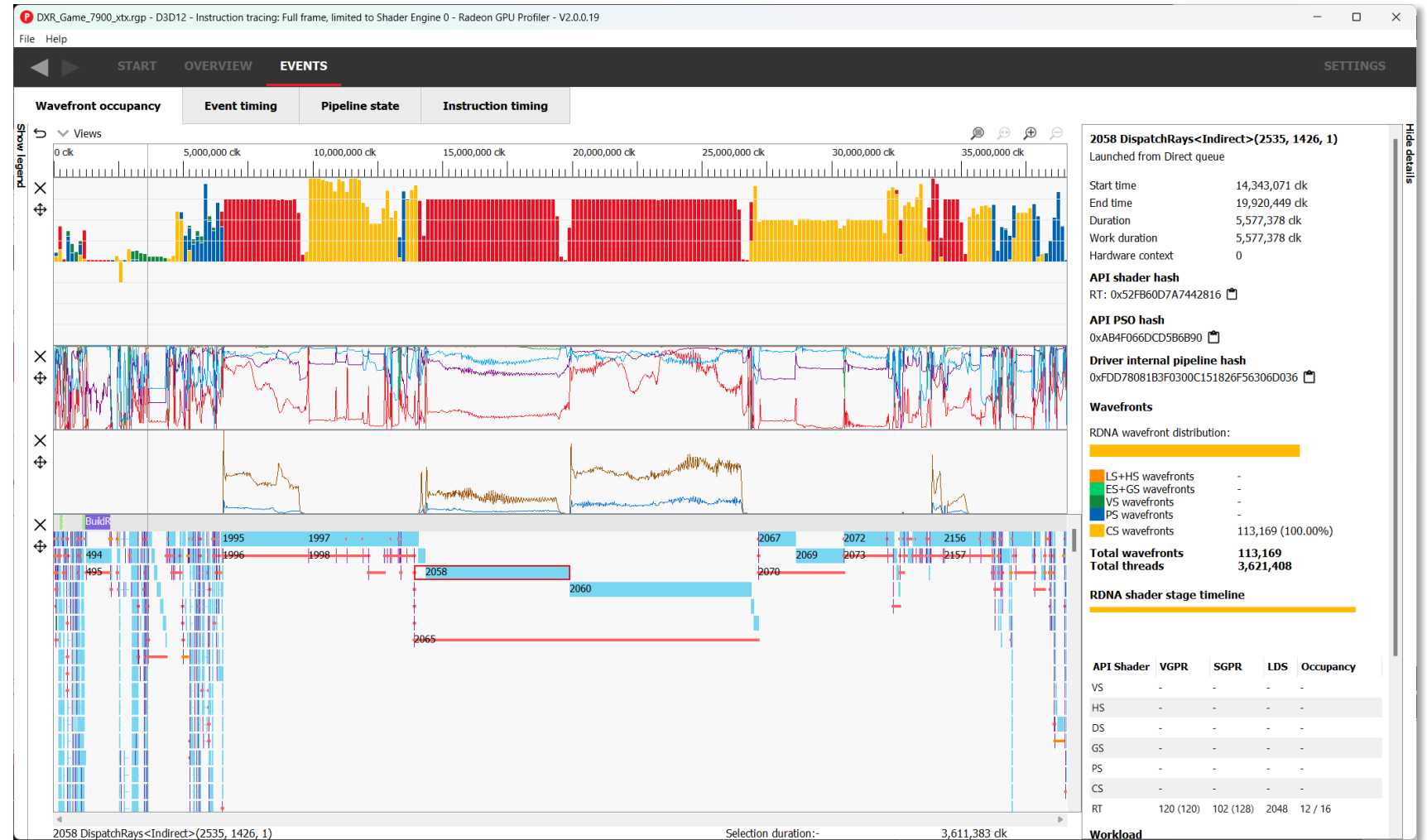
- The default view can be restored



RGP – CUSTOMIZE WAVEFRONT OCCUPANCY LAYOUT

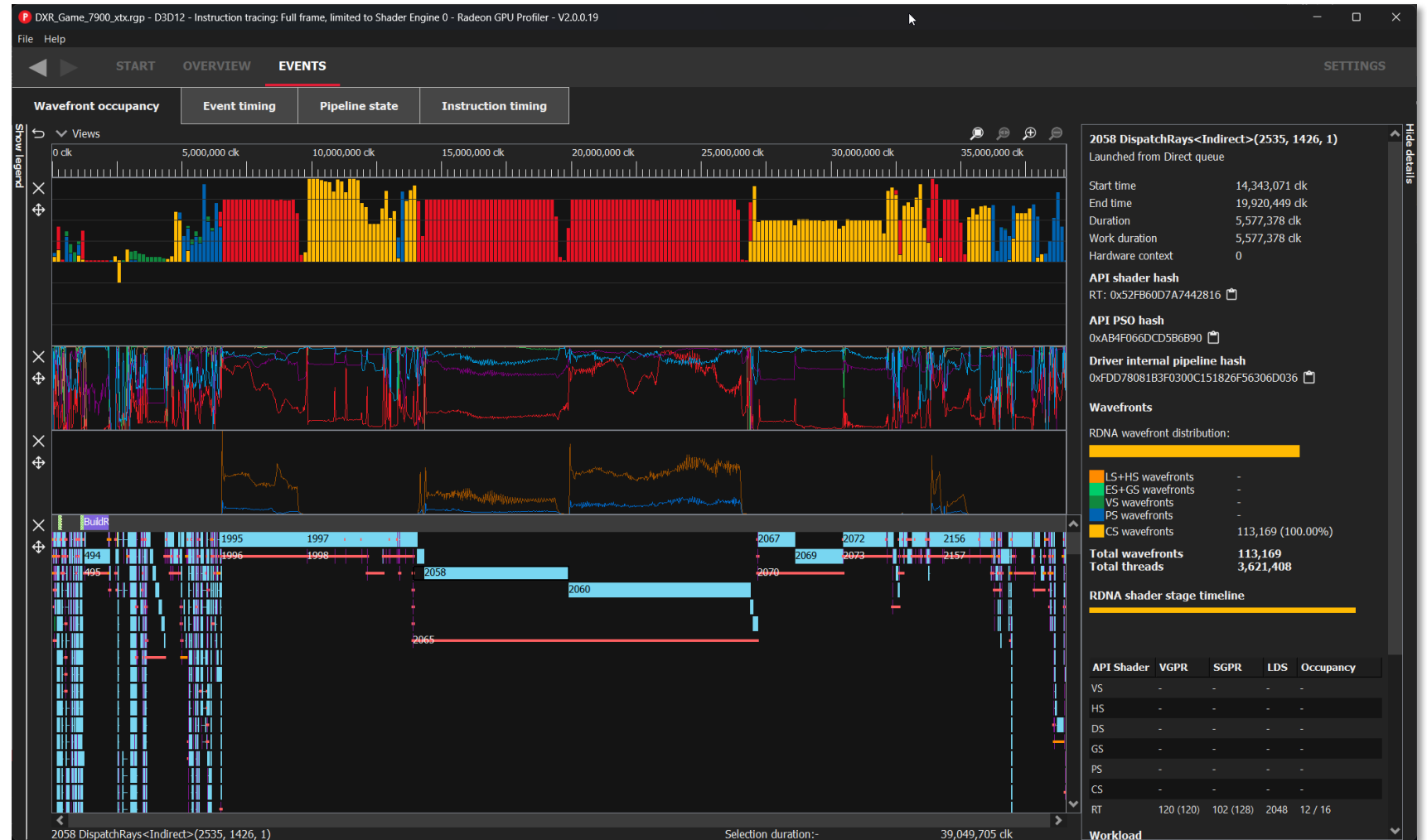
New layout

- The default view can be restored



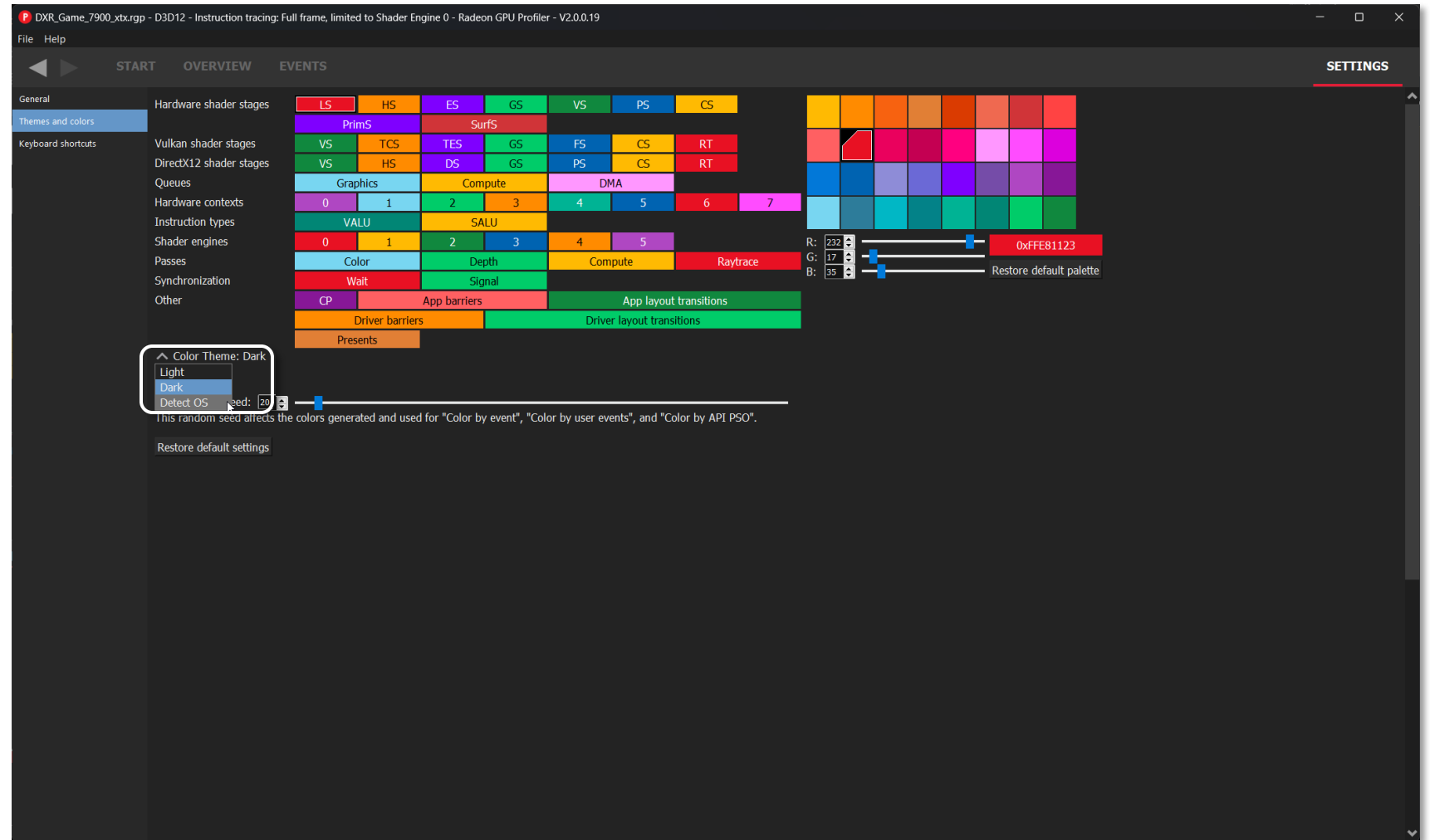
RGP – DARK MODE

- RGP can be set to use Dark Mode or Light mode
- Or it can be set to follow the host operating system



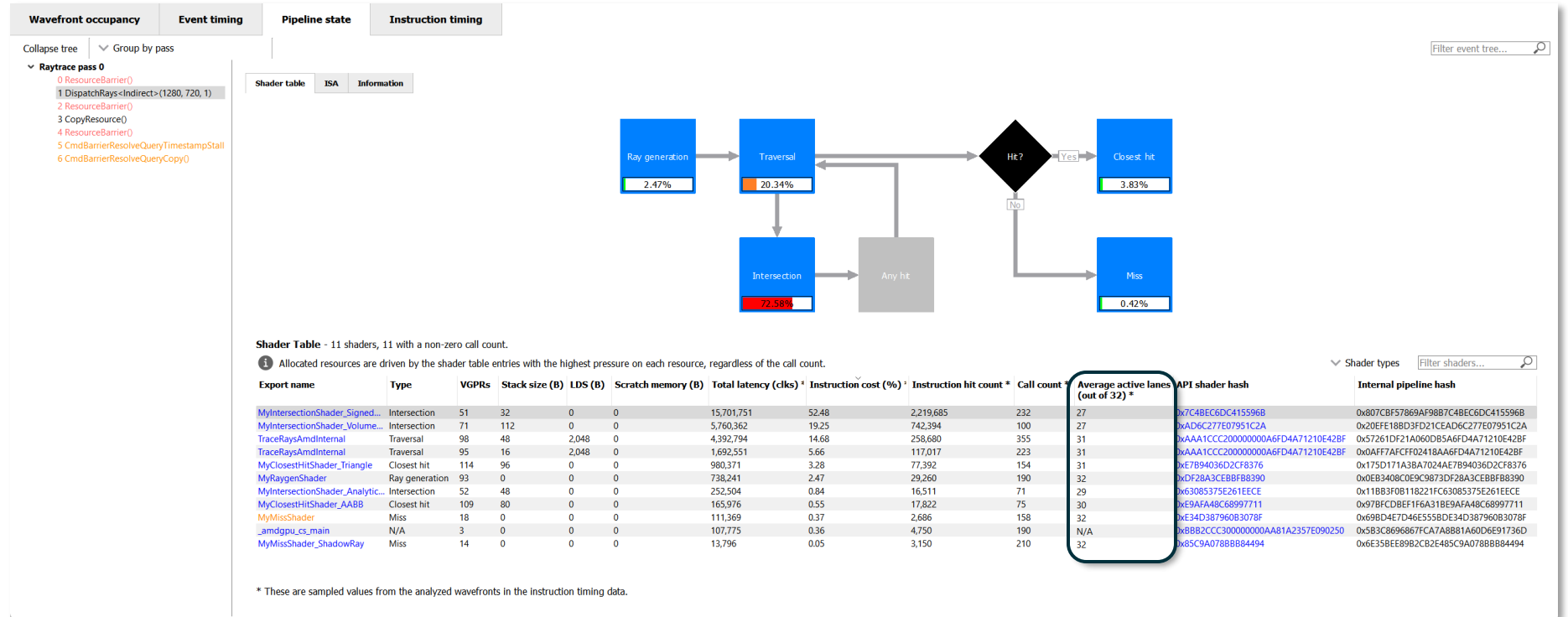
RGP – DARK MODE

- RGP can be set to use Dark Mode or Light mode
- Or it can be set to follow the host operating system
- Use the new “Color Theme” setting on the “Themes and Colors” page



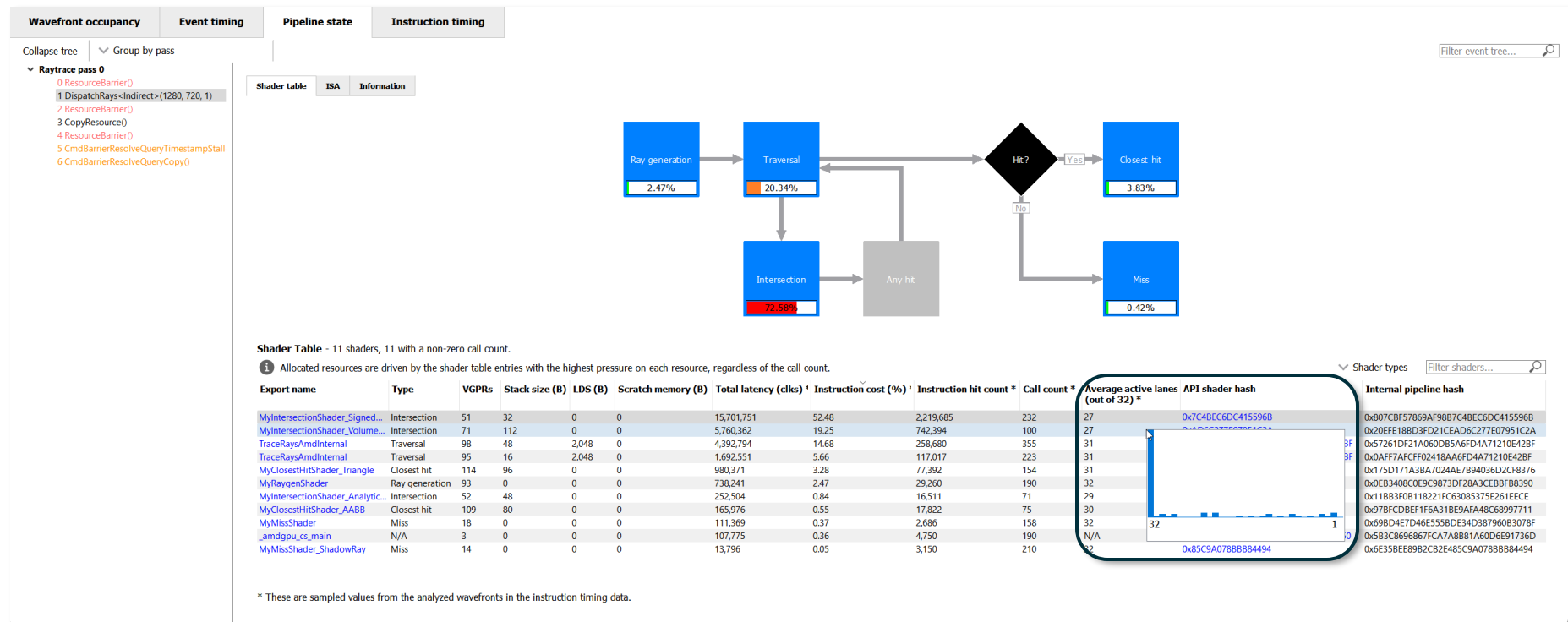
RGP: RAY TRACING THREAD DIVERGENCE

- Learn about thread divergence in your ray tracing pipelines
- RGP reports the average number of active lanes upon entry of each function in the ray tracing pipeline



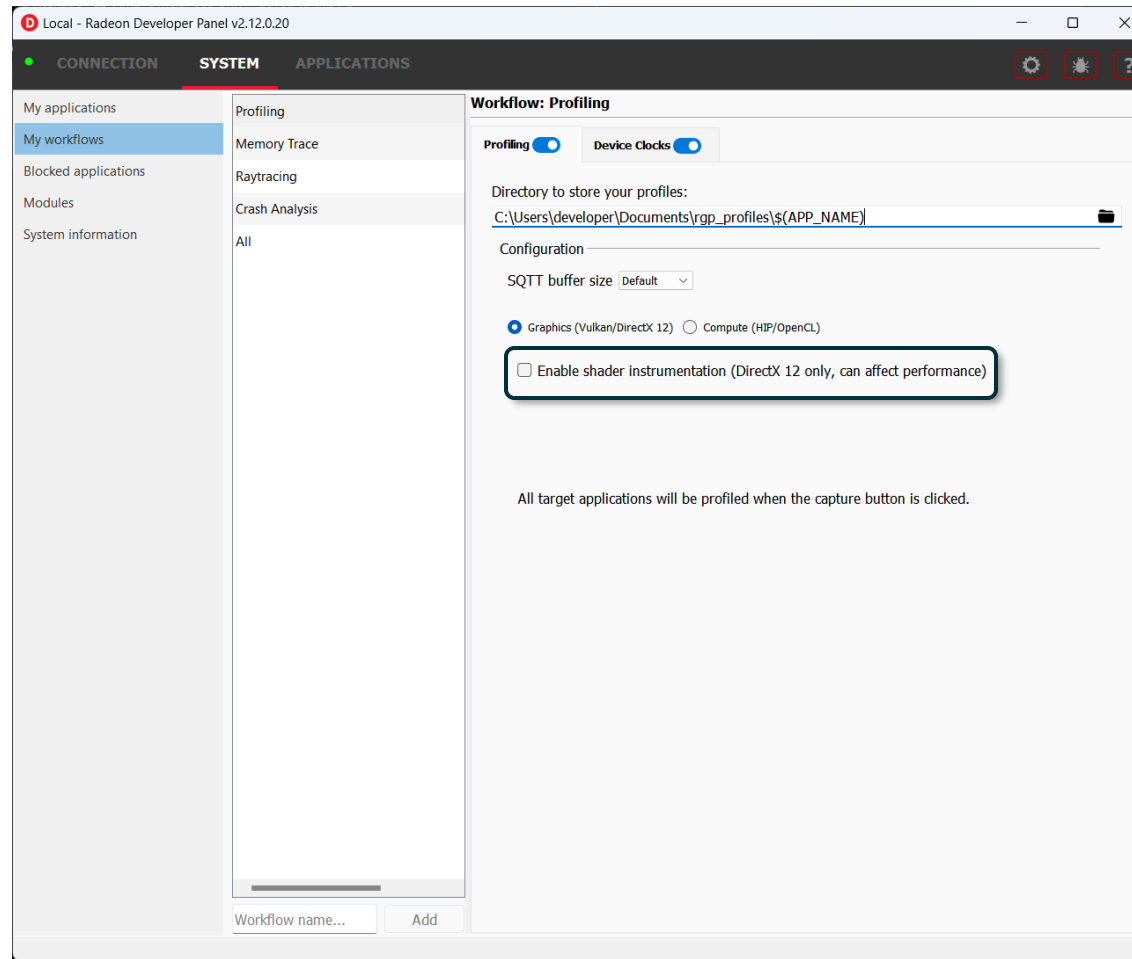
RGP: RAY TRACING THREAD DIVERGENCE

- Learn about thread divergence in your ray tracing pipelines
- New column reports the average number of active lanes upon entry of each function in the ray tracing pipeline
- Histogram shows the distribution across all invocations



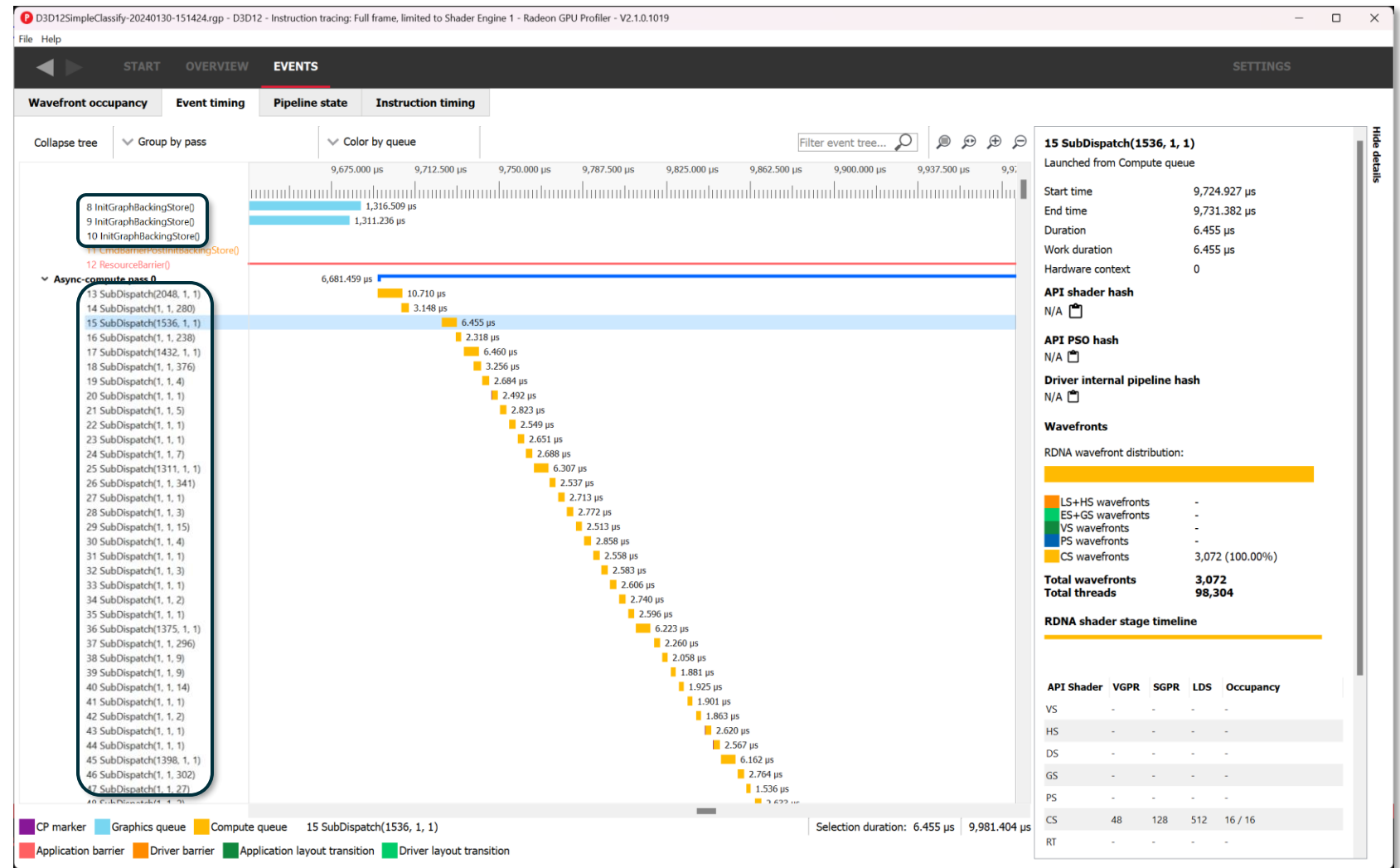
RGP: RAY TRACING THREAD DIVERGENCE – RDP SUPPORT

- “Enable shader instrumentation” checkbox in RDP
- May add extra overhead which can affect runtime performance



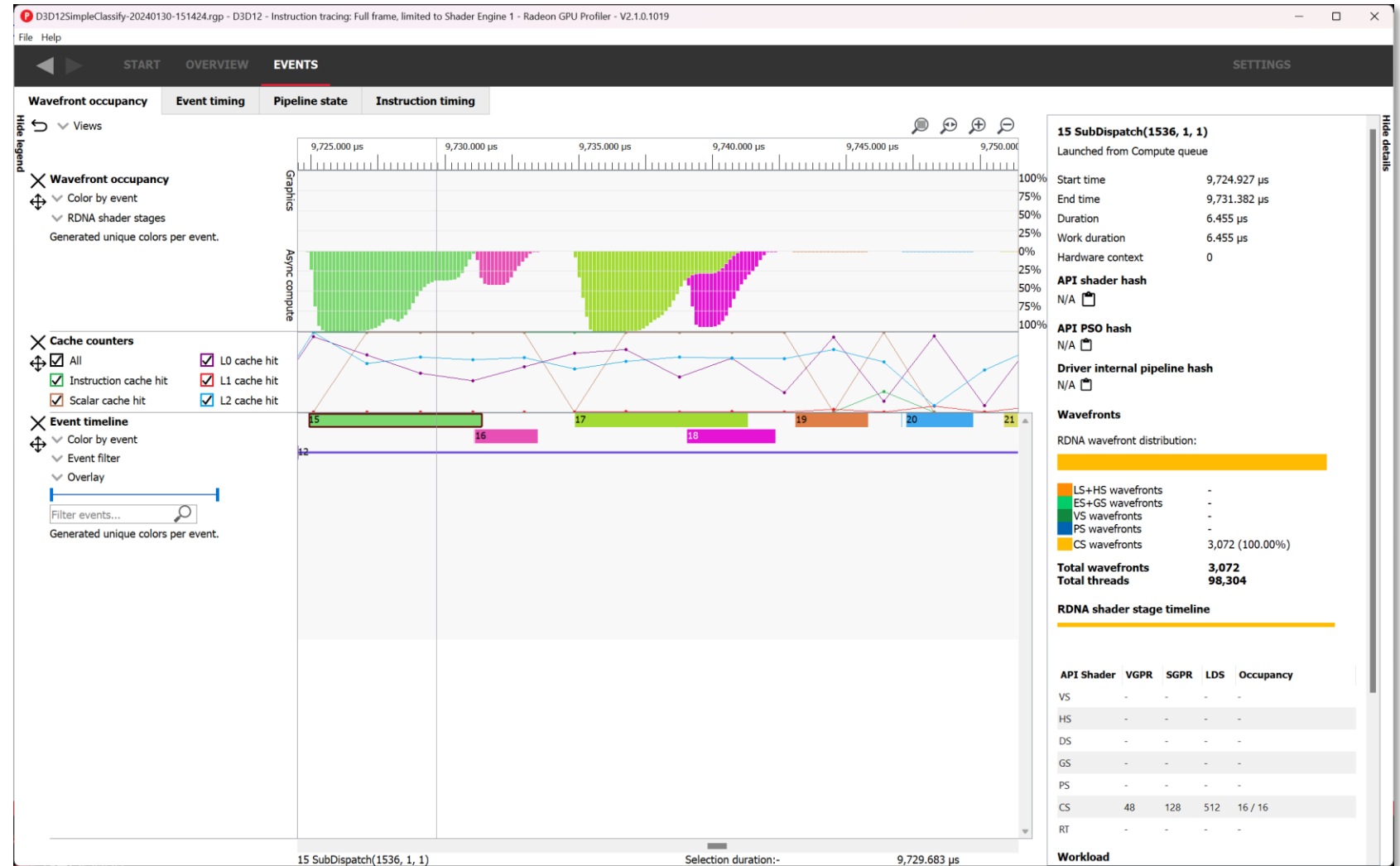
RGP: WORK GRAPHS

- RGP's event lists show individual sub-dispatches
- Shows how the work is broken down during graph execution



RGP: WORK GRAPHS

- RGP's event lists show individual sub-dispatches
- Shows how the work is broken down during graph execution
- Coloring by event shows which waves come from which graph sub-dispatches
- More info on GPUOpen: <https://gpuopen.com/learn/rp-work-graphs/>



AMD

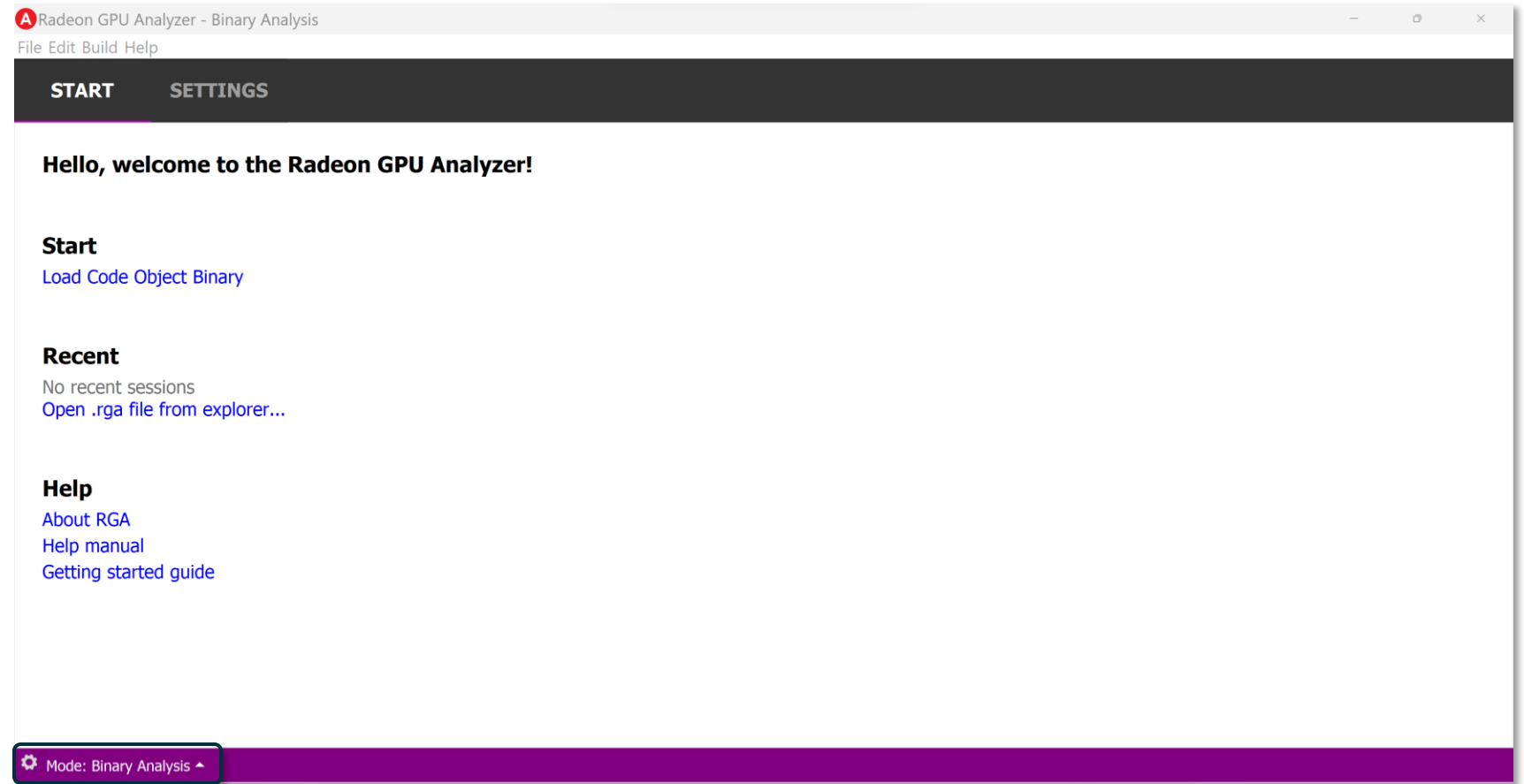
RADEON

GPU Analyzer

AMD RADEON™ GPU ANALYZER

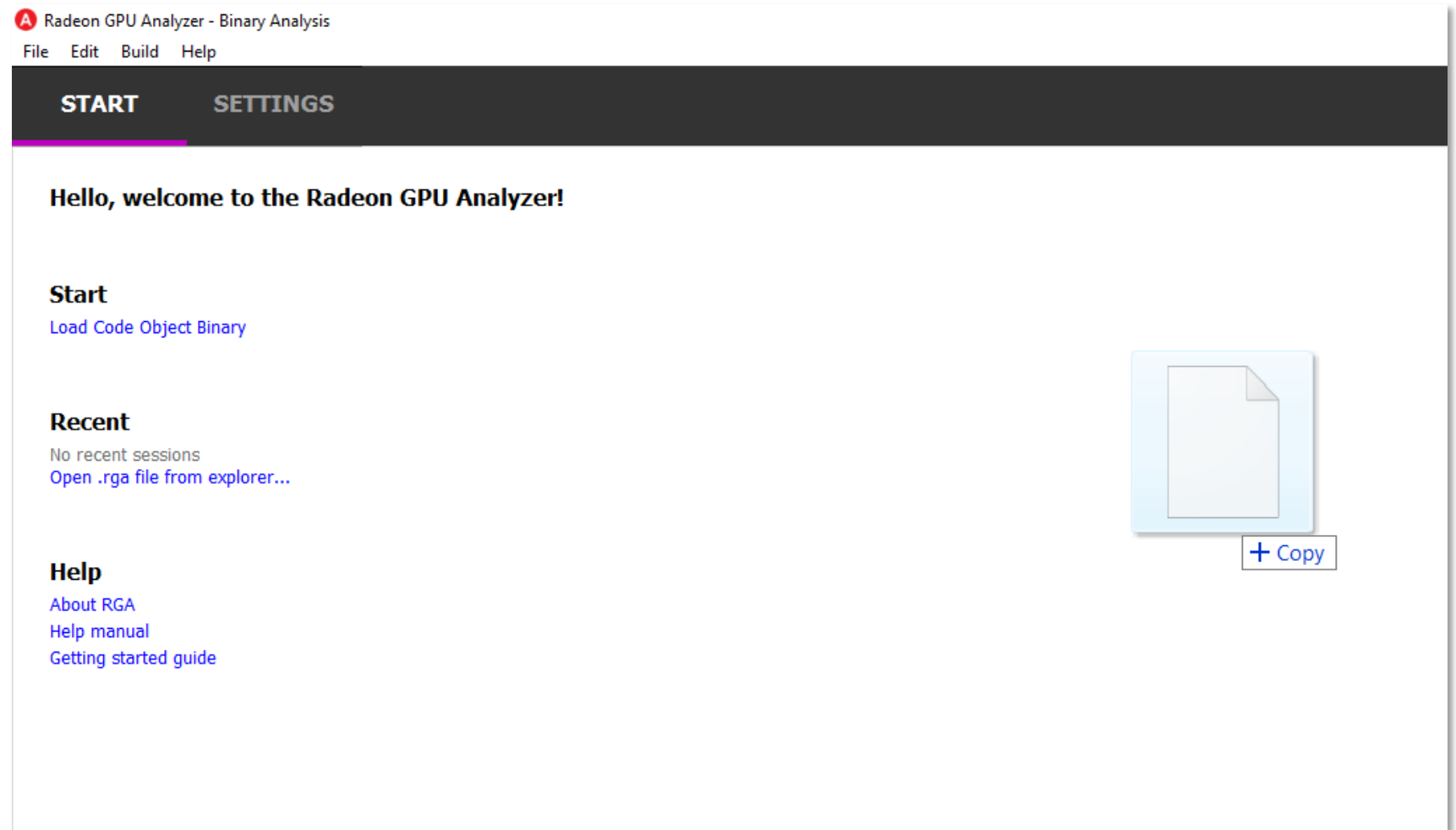
New mode

- Binary Analysis



RGA BINARY ANALYSIS MODE

- Drag & drop any **pre-compiled** AMD GPU Code Object binary
- Unlike other RGA modes, here we start with a **pre-compiled Code Object binary file**



RTX BINARY ANALYSIS MODE

- Shaders and kernels appear on the left pane

Radeon GPU Analyzer - Binary Analysis Project - 240205-135211

File Edit Build Help

Project name: 240205-135211

gfx1103 (RDNA3) Columns

gfx1103_Pa...raw_Binary.bin	Address	Opcode	Operands	VGPR pressure (used:24, allocated:32/256)
DX12_Vertex	0x000000	s_version	0x2006	4
DX12_Geometry	0x000004	s_bfe_u32	s4, s3, 0x80000	4
DX12_Pixel	0x00000C	s_delay_alu	instid0(SALU_CYCLE_1) instskip(SKIP_1) instid1(SALU_CYCLE_1)	4
	0x000010	s_bfm_b64	exec, s4, 0	4
	0x000014	s_cmp_eq_u32	s4, 64	4
	0x000018	s_cselect_b64	exec, -1, exec	4
	0x00001C	s_getpc_b64	s[6:7]	4
	0x000020	s_mov_b32	s22, s8	4
	0x000024	s_mov_b32	s23, s7	4
	0x000028	s_set_inst_prefetch_distance	0x3	4
	0x00002C	s_cmp_eq_u32	s4, 0	4
	0x000030	s_cbranch_scc1	L0	4
	0x000034	s_mov_b32	s22, s20	4
	0x000038	s_mov_b32	s23, s7	4
	0x00003C	s_mov_b32	s6, s12	4
	0x000040	s_load_b128	s[24:27], s[22:23], null	4
	0x000048	s_load_b128	s[28:31], s[6:7], null	4
	0x000050	v_add_nc_u32_e32	v6, s13, v5	5
	0x000054	s_waitcnt	lgkmcnt(0)	5
	0x000058	tbuffer_load_format_xyzw	v[8:11], v6, s[24:27], 0 format:[BUF_FMT_32_32_32_32_FLOAT] idxen	9
	0x000060	s_clause	0x1	8
	0x000064	tbuffer_load_format_x	v6, v5, s[28:31], 0 format:[BUF_FMT_32_FLOAT] idxen offset:28	9
	0x00006C	tbuffer_load_format_xyz	v[12:14], v5, s[28:31], 0 format:[BUF_FMT_32_32_32_FLOAT] idxen	12
	0x000074	s_bfe_u32	s4, s3, 0x40018	11
	0x00007C	s_delay_alu	instid0(SALU_CYCLE_1) instskip(SKIP_1) instid1(VALU_DEP_1)	11
	0x000080	s_mulk_i32	s4, 0x900	11
	0x000084	v_mbcnt_lo_u32_b32	v7, -1, 0	12
	0x00008C	v_mbcnt_hi_u32_b32	v5, -1, v7	13
	0x000094	s_delay_alu	instid0(VALU_DEP_1)	12
	0x000098	v_mad_u32_u24	v7, v5, 36, s4	13
	0x0000A0	s_waitcnt	vmcnt(2)	12
	0x0000A4	v_add_f32_e32	v5, -1.0, v8	13
	0x0000A8	v_add_f32_e32	v8, 0xbdcccccd, v9	13
	0x0000B0	s_waitcnt	vmcnt(1)	12
	0x0000B4	v_mul_f32_e32	v6, 0x3de38e39, v6	12
	0x0000BC	v_add_f32_e32	v9, 0xbdcccccd, v10	13
	0x0000C4	v_add_f32_e32	v10, -1.0, v11	13
	0x0000C8	s_delay_alu	instid0(VALU_DEP_3) instskip(SKIP_1) instid1(VALU_DEP_4)	12
	0x0000CC	v_mul_f32_e32	v5, v6, v5	12
	0x0000D0	v_mul_f32_e32	v8, v6, v8	12
	0x0000D4	v_mul_f32_e32	v9, v6, v9	12
	0x0000D8	s_delay_alu	instid0(VALU_DEP_4) instskip(NEXT) instid1(VALU_DEP_4)	12
	0x0000DC	v_mul_f32_e32	v6, v6, v10	12

RGa BINARY ANALYSIS MODE

- Shaders and kernels appear on the left pane
- Binary gets disassembled

A Radeon GPU Analyzer - Binary Analysis Project - 240205-135211
File Edit Build Help

Project name: 240205-135211	gfx1103 (RDNA3) Columns			VGPR pressure (used:24, allocated:32/256)
gfx1103_Pa...raw_Binary.bin	Address	Opcode	Operands	
Dx12_Vertex	0x000000	s_version	0x2006	4
Dx12_Geometry	0x000004	s_bfe_u32	s4, s3, 0x80000	4
Dx12_Pixel	0x00000C	s_delay_alu	instid0(SALU_CYCLE_1) instskip(SKIP_1) instid1(SALU_CYCLE_1)	4
	0x000010	s_bfm_b64	exec, s4, 0	4
	0x000014	s_cmp_eq_u32	s4, 64	4
	0x000018	s_cselect_b64	exec, -1, exec	4
	0x00001C	s_getpc_b64	s[6:7]	4
	0x000020	s_mov_b32	s22, s8	4
	0x000024	s_mov_b32	s23, s7	4
	0x000028	s_set_inst_prefetch_distance	0x3	4
	0x00002C	s_cmp_eq_u32	s4, 0	4
	0x000030	s_cbranch_scc1	_l0	4
	0x000034	s_mov_b32	s22, s20	4
	0x000038	s_mov_b32	s23, s7	4
	0x00003C	s_mov_b32	s6, s12	4
	0x000040	s_load_b128	s[24:27], s[22:23], null	4
	0x000048	s_load_b128	s[28:31], s[6:7], null	4
	0x000050	v_add_nc_u32_e32	v6, s13, v5	5
	0x000054	s_waitcnt	lgkmcnt(0)	5
	0x000058	tbuffer_load_format_xyzw	v[8:11], v6, s[24:27], 0 format:[BUF_FMT_32_32_32_FLOAT] idxen	9
	0x000060	s_clause	0x1	8
	0x000064	tbuffer_load_format_x	v6, v5, s[28:31], 0 format:[BUF_FMT_32_FLOAT] idxen offset:28	9
	0x00006C	tbuffer_load_format_xyz	v[12:14], v5, s[28:31], 0 format:[BUF_FMT_32_32_FLOAT] idxen	12
	0x000074	s_bfe_u32	s4, s3, 0x40018	11
	0x00007C	s_delay_alu	instid0(SALU_CYCLE_1) instskip(SKIP_1) instid1(VALU_DEP_1)	11
	0x000080	s_mulk_i32	s4, 0x900	11
	0x000084	v_mbcnt_lo_u32_b32	v7, -1, 0	12
	0x00008C	v_mbcnt_hi_u32_b32	v5, -1, v7	13
	0x000094	s_delay_alu	instid0(VALU_DEP_1)	12
	0x000098	v_mad_u32_u24	v7, v5, 36, s4	13
	0x0000A0	s_waitcnt	vmcnt(2)	12
	0x0000A4	v_add_f32_e32	v5, -1.0, v8	13
	0x0000A8	v_add_f32_e32	v8, 0xbdcccccd, v9	13
	0x0000B0	s_waitcnt	vmcnt(1)	12
	0x0000B4	v_mul_f32_e32	v6, 0x3de38e39, v6	12
	0x0000BC	v_add_f32_e32	v9, 0xbdcccccd, v10	13
	0x0000C4	v_add_f32_e32	v10, -1.0, v11	13
	0x0000C8	s_delay_alu	instid0(VALU_DEP_3) instskip(SKIP_1) instid1(VALU_DEP_4)	12
	0x0000CC	v_mul_f32_e32	v5, v6, v5	12
	0x0000D0	v_mul_f32_e32	v8, v6, v8	12
	0x0000D4	v_mul_f32_e32	v9, v6, v9	12
	0x0000D8	s_delay_alu	instid0(VALU_DEP_4) instskip(NEXT) instid1(VALU_DEP_4)	12
	0x0000DC	v_mul_f32_e32	v6, v6, v10	12

RGAs BINARY ANALYSIS MODE

- Shaders and kernels appear on the left pane
- Binary gets disassembled
- VGPR pressure visualized

Radeon GPU Analyzer - Binary Analysis Project - 240205-135211

File Edit Build Help

Project name: 240205-135211

gfx1103 (RDNA3) Columns

gfx1103_Pa...raw_Binary.bin

DX12_Vertex
DX12_Geometry
DX12_Pixel

Address	Opcode	Operands	VGPR pressure (used:24, allocated:32/256)
0x000000	s_version	0x2006	4
0x000004	s_bfe_u32	s4, s3, 0x80000	4
0x00000C	s_delay_alu	instid0(SALU_CYCLE_1) instskip(SKIP_1) instid1(SALU_CYCLE_1)	4
0x000010	s_bfm_b64	exec, s4, 0	4
0x000014	s_cmp_eq_u32	s4, 64	4
0x000018	s_cselect_b64	exec, -1, exec	4
0x00001C	s_getpc_b64	s[6:7]	4
0x000020	s_mov_b32	s22, s8	4
0x000024	s_mov_b32	s23, s7	4
0x000028	s_set_inst_prefetch_distance	0x3	4
0x00002C	s_cmp_eq_u32	s4, 0	4
0x000030	s_cbranch_scc1	_L0	4
0x000034	s_mov_b32	s22, s20	4
0x000038	s_mov_b32	s23, s7	4
0x00003C	s_mov_b32	s6, s12	4
0x000040	s_load_b128	s[24:27], s[22:23], null	4
0x000048	s_load_b128	s[28:31], s[6:7], null	4
0x000050	v_add_nc_u32_e32	v6, s13, v5	5
0x000054	s_waitcnt	lgkmcnt(0)	5
0x000058	tbuffer_load_format_xyzw	v[8:11], v6, s[24:27], 0 format:[BUF_FMT_32_32_32_32_FLOAT] idxen	9
0x000060	s_clause	0x1	8
0x000064	tbuffer_load_format_x	v6, v5, s[28:31], 0 format:[BUF_FMT_32_FLOAT] idxen offset:28	9
0x00006C	tbuffer_load_format_xyz	v[12:14], v5, s[28:31], 0 format:[BUF_FMT_32_32_32_FLOAT] idxen	12
0x000074	s_bfe_u32	s4, s3, 0x40018	11
0x00007C	s_delay_alu	instid0(SALU_CYCLE_1) instskip(SKIP_1) instid1(VALU_DEP_1)	11
0x000080	s_mulk_i32	s4, 0x900	11
0x000084	v_mbcnt_lo_u32_b32	v7, -1, 0	12
0x00008C	v_mbcnt_hi_u32_b32	v5, -1, v7	13
0x000094	s_delay_alu	instid0(VALU_DEP_1)	12
0x000098	v_mad_u32_u24	v7, v5, 36, s4	13
0x0000A0	s_waitcnt	vmcnt(2)	12
0x0000A4	v_add_f32_e32	v5, -1.0, v8	13
0x0000A8	v_add_f32_e32	v8, 0xbdcccccd, v9	13
0x0000B0	s_waitcnt	vmcnt(1)	12
0x0000B4	v_mul_f32_e32	v6, 0x3de38e39, v6	12
0x0000BC	v_add_f32_e32	v9, 0xbdcccccd, v10	13
0x0000C4	v_add_f32_e32	v10, -1.0, v11	13
0x0000C8	s_delay_alu	instid0(VALU_DEP_3) instskip(SKIP_1) instid1(VALU_DEP_4)	12
0x0000CC	v_mul_f32_e32	v5, v6, v5	12
0x0000D0	v_mul_f32_e32	v8, v6, v8	12
0x0000D4	v_mul_f32_e32	v9, v6, v9	12
0x0000D8	s_delay_alu	instid0(VALU_DEP_4) instskip(NEXT) instid1(VALU_DEP_4)	12
0x0000DC	v_mul_f32_e32	v6, v6, v10	12

RGa BINARY ANALYSIS MODE

- RGA will detect modified binaries and reload the contents automatically
- You can continue to use your normal workflows to edit and compile binaries – RGA will always show the latest updates

Radeon GPU Analyzer - Binary Analysis Project - 240301-095051

File Edit Build Help

Project name: 240301-095051

gfx11_matr...ult_kernel.bin

_Z9mmmKern...4EES1_S1_jj

gfx11100 (RDNA3) Columns

Address	Opcode	Operands
0x001900	s_clause	0x1
0x001904	s_load_b32	s8, s[0:1], 0x2c
0x00190C	s_load_b256	s[0:7], s[0:1], null
0x001914	v_bfe_u32	v1, v0, 10, 10
0x00191C	v_and_b32_e32	v0, 0x3ff, v0
0x001924	s_waitcnt	lgkmcnt(0)
0x001928	s_lshr_b32	s9, s8, 16
0x00192C	s_and_b32	s8, s8, 0xffff
0x001934	s_mul_i32	s15, s15, s9
0x001938	v_mad_u64_u32	v[40:41], null, s14, s8, v[0:1]
0x001940	v_add_lshl_u32	v51, s15, v1, 2
0x001948	s_lshr_b32	s7, s7, 2
0x00194C	s_cmp_lg_u32	s6, 0
0x001950	s_delay_alu	instid0(VALU_DEP_1)
0x001954	v_or_b32_e32	v55, 1, v51
0x001958	v_or_b32_e32	v56, 2, v51
0x00195C	v_or_b32_e32	v57, 3, v51
0x001960	s_cbranch	instid1(VALU_DEP_4)
0x001964	s_lshr_b32	s9, s9, 16
0x001968	v_dual_mov	v47, s9, v1
0x001970	v_mul_lo_u32	v52, 3, v51
0x001978	v_dual_mov	v4, v42 :: v_dual_add_nc_u32 v45, s9, v41
0x001980	s_delay_alu	v5, v42
0x001984	v_mul_lo_u32	v47, s9, v1
0x00198C	v_or_b32_e32	v52, 3, v51
0x001990	v_or_b32_e32	v52, 3, v51
0x001994	v_mul_lo_u32	v47, s9, v1
0x00199C	v_or_b32_e32	v52, 3, v51
0x0019A0	v_dual_mov_b32	v4, v42 :: v_dual_add_nc_u32 v45, s9, v41
0x0019A8	v_mov_b32_e32	v5, v42

VGPR pressure (used:68, allocated: 256)

Resource usage | VGPRs: 71 / 256 | SGPRs: 18 / 106 | LDS: 0 / 64 KB | Scratch memory: 0 B

PREVIEW: RGP/RGA INTEROP

DXR_Game_7900_xtx.rgp - D3D12 - Instruction tracing: Full frame, limited to Shader Engine 0 - Radeon GPU Profiler - V2.0.0.19

File Help

START OVERVIEW **EVENTS** SETTINGS

Wavefront occupancy Event timing Pipeline state Instruction timing

Collapse tree Group by pass

2043 ResourceBarrier() 2044 ExecuteIndirect() 2045 ResourceBarrier() 2046 DrawIndexedInstanced(3, 1, 0, 0, 0) 2047 ResourceBarrier() 2048 ResourceBarrier() 2049 ExecuteIndirect() 2050 ResourceBarrier() 2051 ExecuteIndirect() 2052 ResourceBarrier() 2053 ExecuteIndirect() 2054 ResourceBarrier() 2055 ResourceBarrier() 2056 DispatchRays<Indirect>(1011, 1187, 2057 ResourceBarrier() 2058 DispatchRays<Indirect>(2535, 1426, 2059 ResourceBarrier() 2060 DispatchRays<Indirect>(2535, 1426, 2061 ResourceBarrier() 2062 DispatchRays<Indirect>(812, 781, 1) 2063 ResourceBarrier() 2064 Dispatch(317, 179, 1) 2065 ResourceBarrier() 2066 Dispatch(317, 179, 1) 2067 Dispatch(317, 179, 1) 2068 Dispatch(317, 179, 1) 2069 Dispatch(317, 179, 1) 2070 ResourceBarrier() 2071 ResourceBarrier() 2072 Dispatch(317, 179, 1) 2073 ResourceBarrier() 2074 ResourceBarrier() 2075 Dispatch(317, 179, 1)

Information ISA

Shader ISA

Viewing Options

Go to line... Search... No results

OpCode	Operands
0 // API shader hash:	0x349D8BC35D80AD858C8FB18784AE349C
1 // API PSO hash:	0x328030C09830B5B1
2 // Driver internal pipeline hash:	0x4D6A052012B580632E030C09830B5B1
3 //	
4 // Vector registers:	41 (48 allocated)
5 // Scalar registers:	80 (128 allocated)
6 _endgpu_ps_main	
7 _s_version	0x2006
8 _s_set_inst_prefetch_distance	0x3
9 _s_getpc_b64	s[0:1]
10 _s_pack_11_b32_b16	s0, s6, 16
11 _s_mov_b32	s12, s5
12 _s_mov_b32	s13, s0
13 _s_mov_b32	s14, 0x1000
14 _s_mov_b32	s15, 0x2003dfac
15 _s_buffer_load_b64	s[4:5], s[12:15], 0x14
16 _s_waitont	lgkmon(0)
17 v_cvt_f32_u32_e32	v0, s5
18 _s_delay_alu	inatl40 (VALU_DRP_1) inatkip (SKIP_4) inatidi (VALU_DRP_2)
19 v_cvt_f32_e32	v0, v0
20 _s_and_b32	s0, s4, 1
21 v_mul_f32_e32	v1, 2.0, v2
22 _s_bfe_u32	s4, s4, 0x10001
23 v_mul_f32_e32	v2, 2.0, v3
24 v_cvt_u32_f32_e32	v1, v1

01011010
10110100
11001001

Radeon GPU Analyzer - Binary Analysis Project - 240130-112037

File Edit Build Help

Project name: 240130-112037

DXR_Game_7...C09830B5B1.eb

Address	OpCode	Operands	VGPR pressure
0x00001C	v_fma_f32	v30, v6, v29, v13	32
0x000024	v_fma_f32	v32, v21, v29, v7	33
0x00002C	v_fma_f32	v33, v6, v31, v13	34
0x000034	v_fma_f32	v34, v21, v31, v7	35
0x00003C	v_fma_f32	v36, v6, v35, v13	36
0x000044	v_fma_f32	v38, v21, v35, v7	37
0x00004C	v_fma_f32	v37, v6, v9, v13	38
0x000054	v_fma_f32	v39, v21, v9, v7	39
0x00005C	s_clause	0x8	39
0x000060	image_sample_lz	v40, [v13, v7], s[16:23], s[0:3] dmask:0x1 dim:SQ_RSRC_IMG_...	40
0x00006C	image_sample_lz	v15, [v15, v20], s[16:23], s[0:3] dmask:0x1 dim:SQ_RSRC_IMG_...	40
0x000078	image_sample_lz	v19, [v19, v24], s[16:23], s[0:3] dmask:0x1 dim:SQ_RSRC_IMG_...	39
0x000084	image_sample_lz	v23, [v23, v25], s[16:23], s[0:3] dmask:0x1 dim:SQ_RSRC_IMG_...	38
0x000090	image_sample_lz	v27, [v27:28], s[16:23], s[0:3] dmask:0x1 dim:SQ_RSRC_IMG_2D	37
0x000098	image_sample_lz	v30, [v30, v32], s[16:23], s[0:3] dmask:0x1 dim:SQ_RSRC_IMG_...	36
0x0000A4	image_sample_lz	v34, [v33:34], s[16:23], s[0:3] dmask:0x1 dim:SQ_RSRC_IMG_2D	35
0x0000AC	image_sample_lz	v38, [v36, v38], s[16:23], s[0:3] dmask:0x1 dim:SQ_RSRC_IMG_...	34
0x0000B8	image_sample_lz	v39, [v37, v39], s[16:23], s[0:3] dmask:0x1 dim:SQ_RSRC_IMG_...	33
0x0000C4	v_fma_f32	v4, v16, s66, s74	33
0x0000C6	v_fma_f32_e32	v3, s0, v30	33

Resource usage | VGPRs: 41 / 256 | SGPRs: 80 / 106 | LDS: 0 / 64 KB | Scratch memory: 0 B |

Build output

Target GPU detected:

gfx1100 (RDNA3)

AMD Radeon PRO W7800

AMD Radeon PRO W7900

AMD Radeon RX 7900 XT

AMD Radeon RX 7900 XTX

AMD Radeon (TM) Graphics

Extracting ISA for gfx1100... succeeded.

Performing live vgpr analysis for gfx1100... succeeded.

PREVIEW: RGP/RGA INTEROP

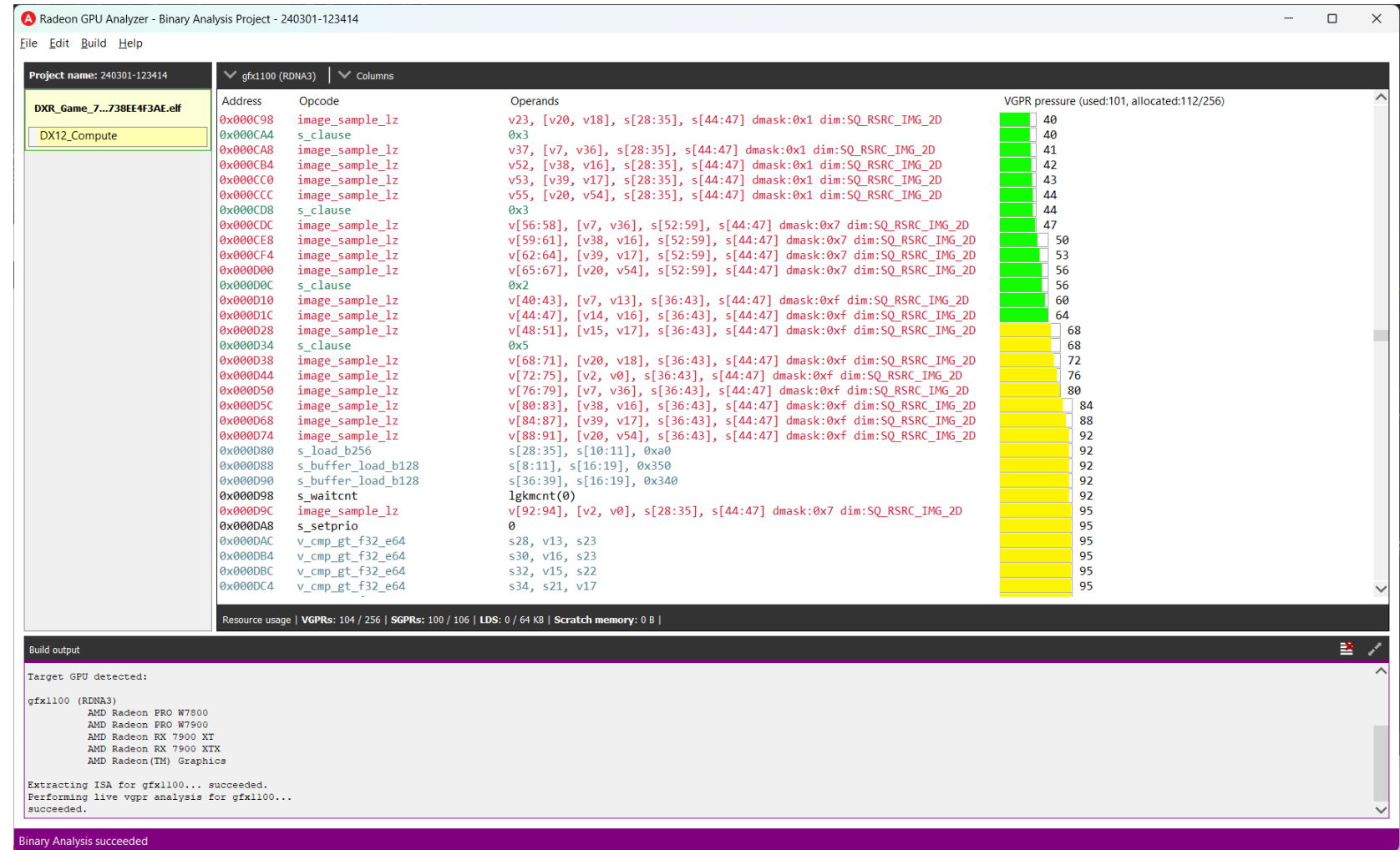
- Any pipeline available in RGP can be automatically extracted and analyzed in RGA
- Use the new “Analyze pipeline in Radeon GPU Analyzer” menu item

The screenshot shows the Radeon GPU Profiler (RGP) interface with the 'EVENTS' tab selected. The top navigation bar includes 'START', 'OVERVIEW', 'EVENTS', and 'SETTINGS'. Below this, there are tabs for 'Wavefront occupancy', 'Event timing', 'Pipeline state', and 'Instruction timing'. The 'Pipeline state' tab is active, showing a sequence of pipeline stages: IA, VS, HS, DS, GS, RS, PS, OM, and CS. The 'Event timing' tab is also visible, showing a list of events. A context menu is open over the event list, highlighting the 'Analyze pipeline in Radeon GPU Analyzer' option. The bottom panel shows the details of the selected event (2007 Dispatch), including the PC address, Opcode, and Operands.

PC address	Opcode	Operands
604 00000000003C	BBF0_4	
	s_clause	0x3
	image_sample_lz	v[24:26], [v7, v13], s[52:59], s[44:47] dmask:0x7 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v[27:29], [v14, v16], s[52:59], s[44:47] dmask:0x7 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v[30:32], [v15, v17], s[52:59], s[44:47] dmask:0x7 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v[33:35], [v20, v18], s[52:59], s[44:47] dmask:0x7 dim:SQ_RSRC_IMG_2D
	s_clause	0x3
	image_sample_lz	v19, [v7, v13], s[28:35], s[44:47] dmask:0x1 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v21, [v14, v16], s[28:35], s[44:47] dmask:0x1 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v22, [v15, v17], s[28:35], s[44:47] dmask:0x1 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v23, [v20, v18], s[28:35], s[44:47] dmask:0x1 dim:SQ_RSRC_IMG_2D
	s_clause	0x3
	image_sample_lz	v37, [v7, v13], s[28:35], s[44:47] dmask:0x1 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v52, [v38, v16], s[28:35], s[44:47] dmask:0x1 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v53, [v39, v17], s[28:35], s[44:47] dmask:0x1 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v55, [v20, v54], s[28:35], s[44:47] dmask:0x1 dim:SQ_RSRC_IMG_2D
	s_clause	0x3
	image_sample_lz	v[56:58], [v7, v36], s[52:59], s[44:47] dmask:0x7 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v[59:61], [v38, v16], s[52:59], s[44:47] dmask:0x7 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v[62:64], [v39, v17], s[52:59], s[44:47] dmask:0x7 dim:SQ_RSRC_IMG_2D
	image_sample_lz	v[65:67], [v20, v54], s[52:59], s[44:47] dmask:0x7 dim:SQ_RSRC_IMG_2D
	s_clause	0x2
	image_sample_lz	v[40:43], [v7, v13], s[36:43], s[44:47] dmask:0xf dim:SQ_RSRC_IMG_2D
	image_sample_lz	v[44:47], [v14, v16], s[36:43], s[44:47] dmask:0xf dim:SQ_RSRC_IMG_2D

PREVIEW: RGP/RGA INTEROP

- The selected pipeline will be extracted
- RGA will be launched, and the extracted pipeline will be loaded and analyzed

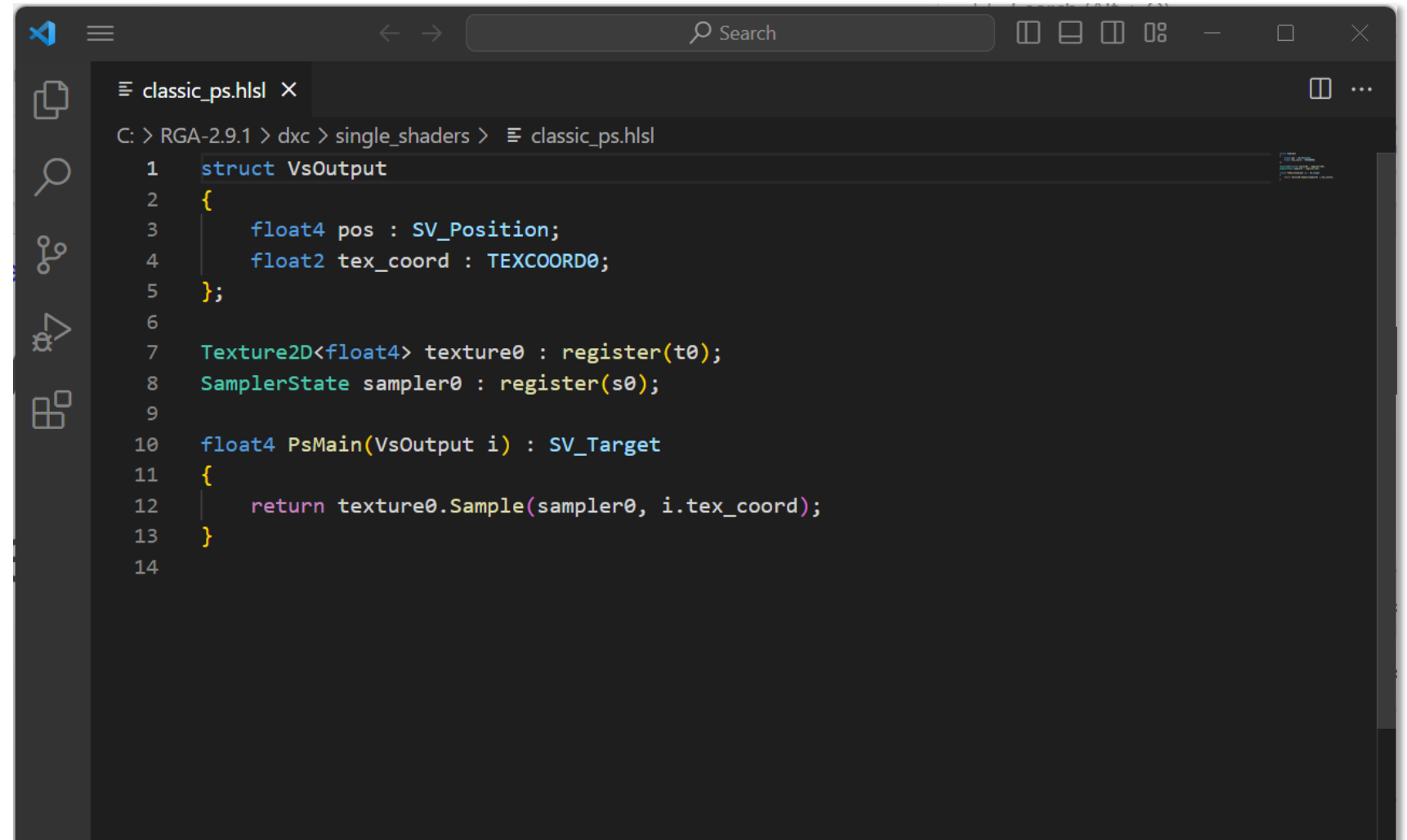


PREVIEW: RGA DX12 SINGLE SHADER COMPILATION

- Currently RGA works with complete pipelines
 - For graphics, that means you would need at least the following:
 - Vertex shader
 - Pixel shader
 - Graphics pipeline state object
 - Root signature
- What if you don't have a complete pipeline?
 - You are missing one of the shaders
 - Missing the graphics pipeline state object
 - Missing the root signature
- RGA will have a new mode that can autogenerate the missing pieces of the pipeline!

PREVIEW: RGA DX12 SINGLE SHADER COMPILATION

- Let's consider an example:
- You have a single pixel shader, but do not have a vertex shader, a root signature or a graphics pipeline state object

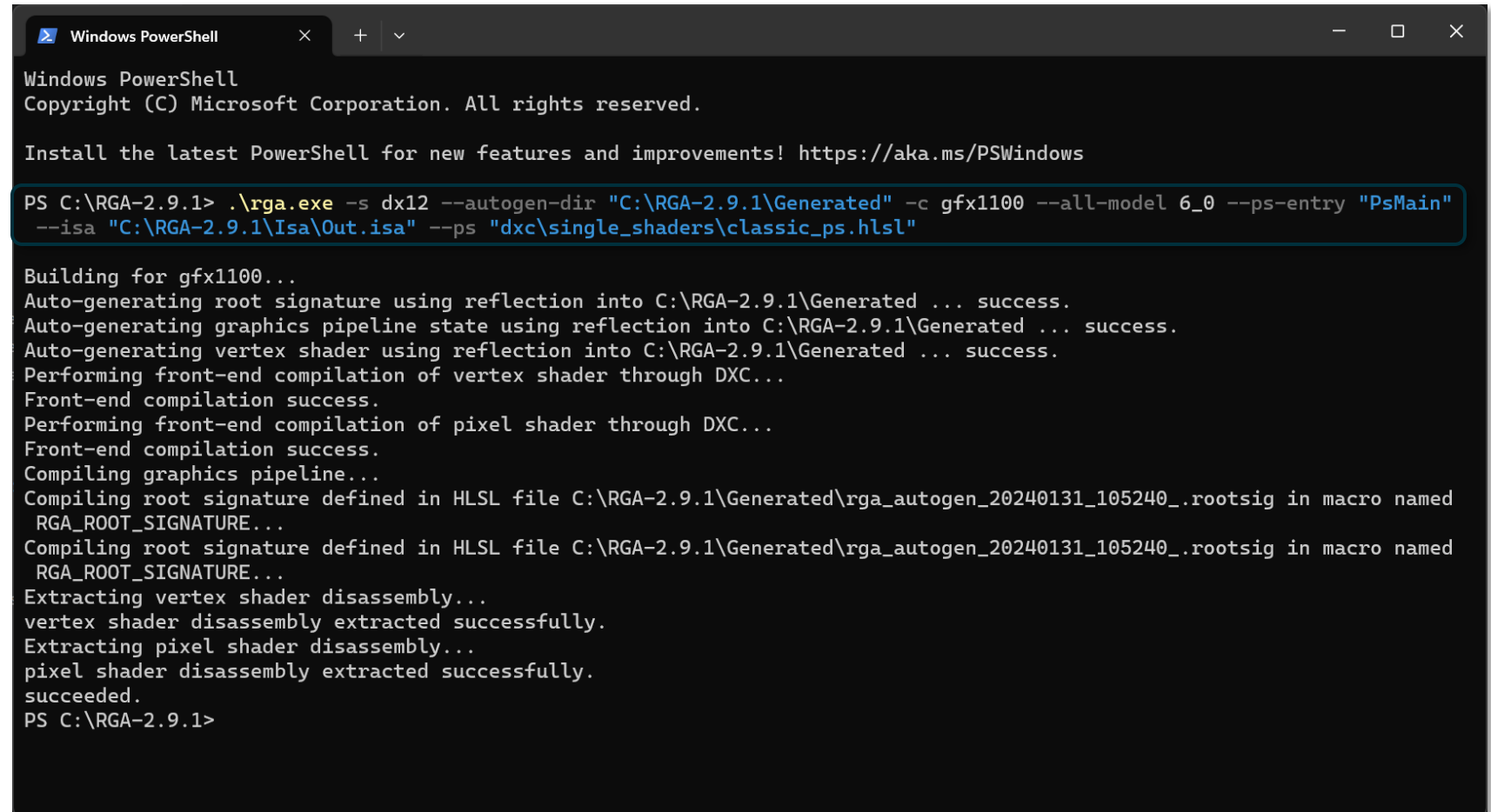


```
classic_ps.hlsl X
C: > RGA-2.9.1 > dxc > single_shaders > classic_ps.hlsl

1 struct VsOutput
2 {
3     float4 pos : SV_Position;
4     float2 tex_coord : TEXCOORD0;
5 };
6
7 Texture2D<float4> texture0 : register(t0);
8 SamplerState sampler0 : register(s0);
9
10 float4 PsMain(VsOutput i) : SV_Target
11 {
12     return texture0.Sample(sampler0, i.tex_coord);
13 }
14
```

PREVIEW: RGA DX12 SINGLE SHADER COMPILATION

- Compiling as usual in DX12 mode, providing only the pixel shader as an input



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\RGA-2.9.1> .\rga.exe -s dx12 --autogen-dir "C:\RGA-2.9.1\Generated" -c gfx1100 --all-model 6_0 --ps-entry "PsMain"
--isa "C:\RGA-2.9.1\Isa\Out.isa" --ps "dxc\single_shaders\classic_ps.hlsl"

Building for gfx1100...
Auto-generating root signature using reflection into C:\RGA-2.9.1\Generated ... success.
Auto-generating graphics pipeline state using reflection into C:\RGA-2.9.1\Generated ... success.
Auto-generating vertex shader using reflection into C:\RGA-2.9.1\Generated ... success.
Performing front-end compilation of vertex shader through DXC...
Front-end compilation success.
Performing front-end compilation of pixel shader through DXC...
Front-end compilation success.
Compiling graphics pipeline...
Compiling root signature defined in HLSL file C:\RGA-2.9.1\Generated\rga_autogen_20240131_105240_.root sig in macro named
RGA_ROOT_SIGNATURE...
Compiling root signature defined in HLSL file C:\RGA-2.9.1\Generated\rga_autogen_20240131_105240_.root sig in macro named
RGA_ROOT_SIGNATURE...
Extracting vertex shader disassembly...
vertex shader disassembly extracted successfully.
Extracting pixel shader disassembly...
pixel shader disassembly extracted successfully.
succeeded.
PS C:\RGA-2.9.1>
```

PREVIEW: RGA DX12 SINGLE SHADER COMPILATION

- The vertex shader is auto generated

```
ga_autogen_20240305_111334_vs.hlsl X
ga_autogen_20240305_111334_vs.hlsl
// Auto-generated with Radeon GPU Analyzer (RGA).

struct VsInput
{
    float4 attribute0: POSITION0;
};

struct VsOutput
{
    float4 attribute0: SV_POSITION;
    float2 attribute1: TEXCOORD0;
};

void main(VsInput input, out VsOutput output)
{
    float4 result = float4(0.0, 0.0, 0.0, 0.0);
    result += float4(input.attribute0.xyzw);
    output.attribute0 = float4(result.xyzw);
    output.attribute1 = float2(result.xy);
}
```

PREVIEW: RGA DX12 SINGLE SHADER COMPILATION

- The root signature is auto generated

```
rga_autogen_20240131_105240.rootsig X
rga_autogen_20240131_105240.rootsig
1 // Auto-generated with Radeon GPU Analyzer (RGA).
2
3 #define RGA_ROOT_SIGNATURE \
4     "RootFlags( ALLOW_INPUT_ASSEMBLER_INPUT_LAYOUT | DENY_HULL_SHADER_ROOT_ACCESS " \
5     "| DENY_DOMAIN_SHADER_ROOT_ACCESS | DENY_GEOMETRY_SHADER_ROOT_ACCESS ), " \
6     "DescriptorTable(Sampler(s0), visibility=SHADER_VISIBILITY_PIXEL), " \
7     "DescriptorTable(SRV(t0), visibility=SHADER_VISIBILITY_PIXEL)"
8
9
```

PREVIEW: RGA DX12 SINGLE SHADER COMPILATION

```
rga_autogen_20240305_111334_gpso X
rga_autogen_20240305_111334_gpso
1  # Auto-generated with Radeon GPU Analyzer (RGA).
2
3  # schemaVersion
4  1.0
5
6  # InputLayoutNumElements (the number of D3D12_INPUT_ELEMENT_DESC elements in the D3D12_INPUT_LAYOUT_DESC structure - must match the following "InputLayout" section)
7  1
8
9  # InputLayout ( {SemanticName, SemanticIndex, Format, InputSlot, AlignedByteOffset, InputSlotClass, InstanceDataStepRate } )
10 { "POSITION", 0, DXGI_FORMAT_R32G32B32A32_FLOAT, 0, 0, D3D12_INPUT_CLASSIFICATION_PER_VERTEX_DATA, 0 }
11
12 # PrimitiveTopologyType (the D3D12_PRIMITIVE_TOPOLOGY_TYPE value to be used when creating the PSO)
13 D3D12_PRIMITIVE_TOPOLOGY_TYPE_TRIANGLE
14
15 # NumRenderTargets (the number of formats in the upcoming RTVFormats section)
16 1
17
18 # RTVFormats (an array of DXGI_FORMAT-typed values for the render target formats - the number of items in the array should match the above NumRenderTargets section)
19 { DXGI_FORMAT_R8G8B8A8_UNORM }
20
21
```

- The graphics pipeline state object is auto generated

PREVIEW: RGA DX12 SINGLE SHADER COMPILATION

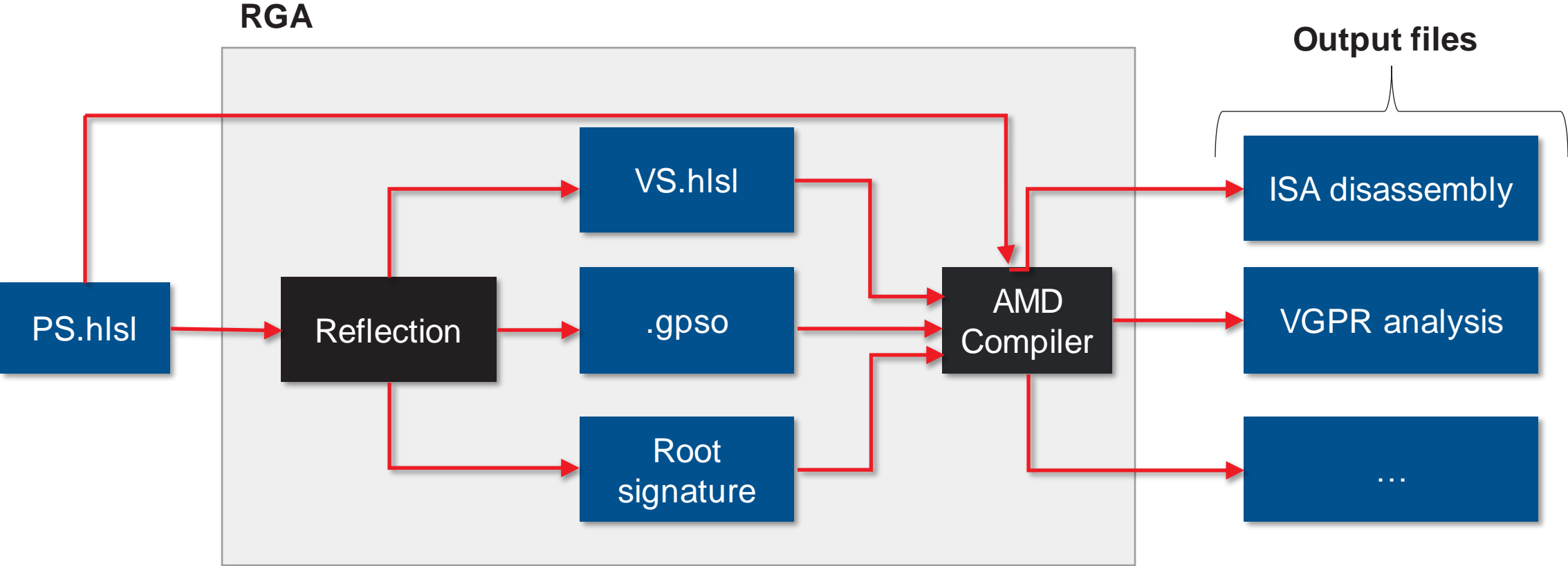
- RGA will output the pixel shader's disassembly

```

1 ; D3D12 Shader Hash 0xa63e3e834124e93ee28366807a0926af
2 ; API PSO Hash 0xdd7f0f8f79875091
3 ; Driver Internal Pipeline Hash 0x2774d5f37b5943a0
4 ; ----- Disassembly -----
5 shader main
6     asic(GFX11)
7     type(PS)
8     sgpr_count(14)
9     vgpr_count(8)
10    wave_size(64)
11    | | | | | | | | | | | | | | | | | | // s_ps_state in s0
12
13    s_version      UC_VERSION_GFX11 | UC_VERSION_W64_BIT // 000000000000: B0802006
14    s_set_inst_prefetch_distance 0x0003 // 000000000004: BF840003
15    s_mov_b32      m0, s4 // 000000000008: BEFD0004
16    s_mov_b64      s[12:13], exec // 00000000000C: BE8C017E
17    s_wqm_b64      exec, exec // 000000000010: BEFE1D7E
18    s_getpc_b64     s[0:1] // 000000000014: BE804780
19    lds_param_load v2, attr0.x wait_vdst:0 // 000000000018: CE000002
20    lds_param_load v3, attr0.y wait_vdst:0 // 00000000001C: CE000103
21    s_mov_b32      s4, s3 // 000000000020: BE840003
22    v_interp_p10_f32 v4, v2, v0, v2 wait_exp:1 // 000000000024: CD000104 040A0102
23    v_interp_p10_f32 v0, v3, v0, v3 wait_exp:0 // 00000000002C: CD000000 040E0103
24    s_mov_b32      s5, s1 // 000000000034: BE850001
25    s_mov_b32      s0, s2 // 000000000038: BE800002
26    | | | | | | | | | | | | | | | | s_delay_alu instid0(VALU_DEP_2) | instskip(SKIP_1) | instid1(VALU_DEP_2) // 00000000003C: BF870122
27    v_interp_p2_f32 v2, v2, v1, v4 wait_exp:7 // 000000000040: CD010702 04120302
28    s_load_b256     s[4:11], s[4:5], null // 000000000048: F40C0102 F8000000
29    v_interp_p2_f32 v0, v3, v1, v0 wait_exp:7 // 000000000050: CD010700 04020303
30    s_load_b128     s[0:3], s[0:1], null // 000000000058: F4080000 F8000000
31    s_and_b64       exec, exec, s[12:13] // 000000000060: 8BFEOC7E
32    s_waitcnt       lgkmcnt(0) // 000000000064: BF89FC07
33    image_sample    v[0:3], [v2,v0], s[4:11], s[0:3] dmask:0xf dim:SQ_RSRC_IMG_2D // 000000000068: F06C0F05 00010002 00000000
34    s_waitcnt       vmcnt(0) // 000000000074: BF8903F7
35    v_cvt_pk_rtz_f16_f32 v0, v0, v1 // 000000000078: 5E000300
36    v_cvt_pk_rtz_f16_f32 v2, v2, v3 // 00000000007C: 5E040702
37    s_mov_b64       exec, s[12:13] // 000000000080: BEFE010C
38    exp            mrt0, v0, v2, off, off done // 000000000084: F8000803 00000200
39    s_endpgm        // 00000000008C: BEFA0000

```

PREVIEW: RGA DX12 SINGLE SHADER COMPILATION



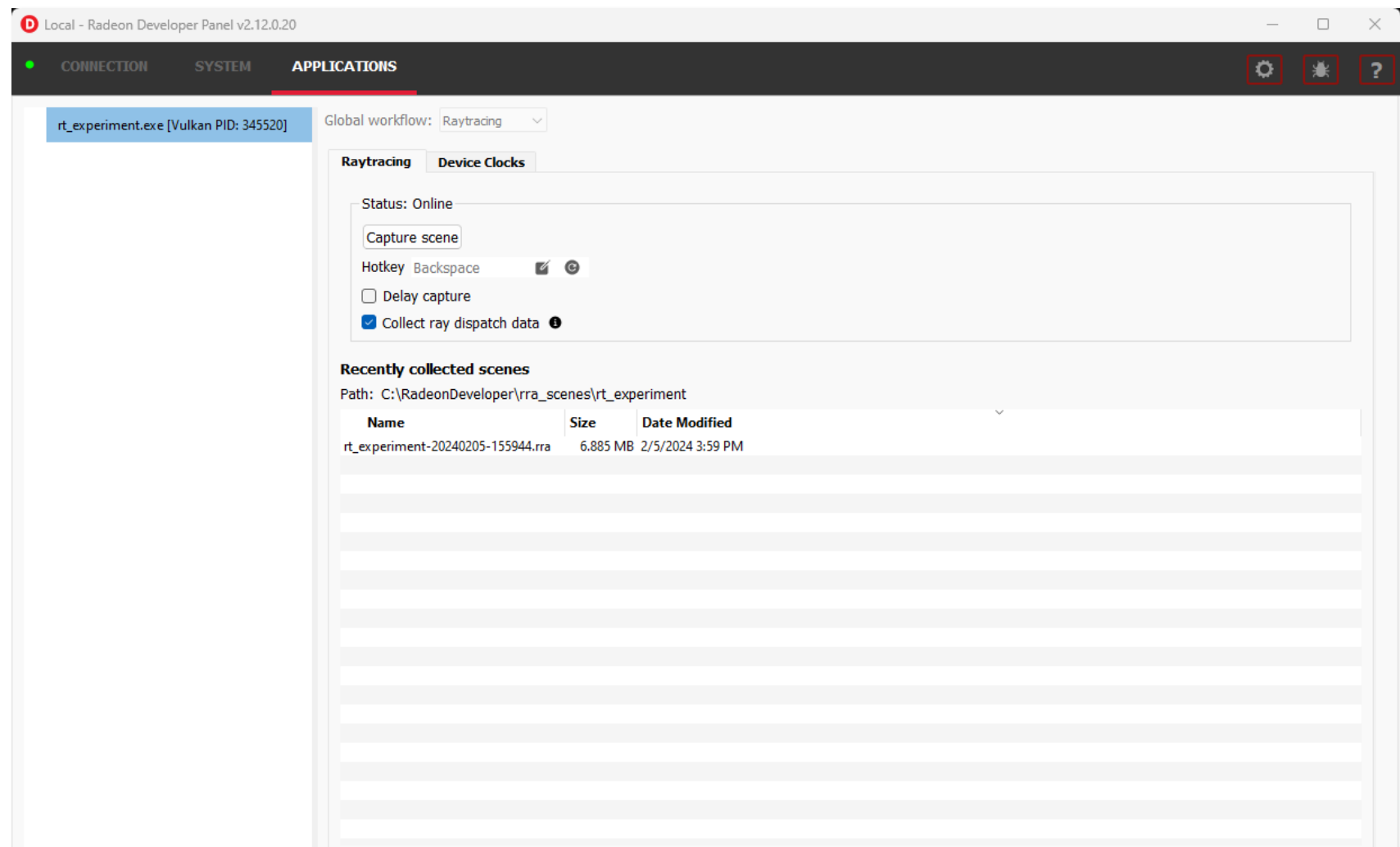
AMD

RADEON

Raytracing Analyzer

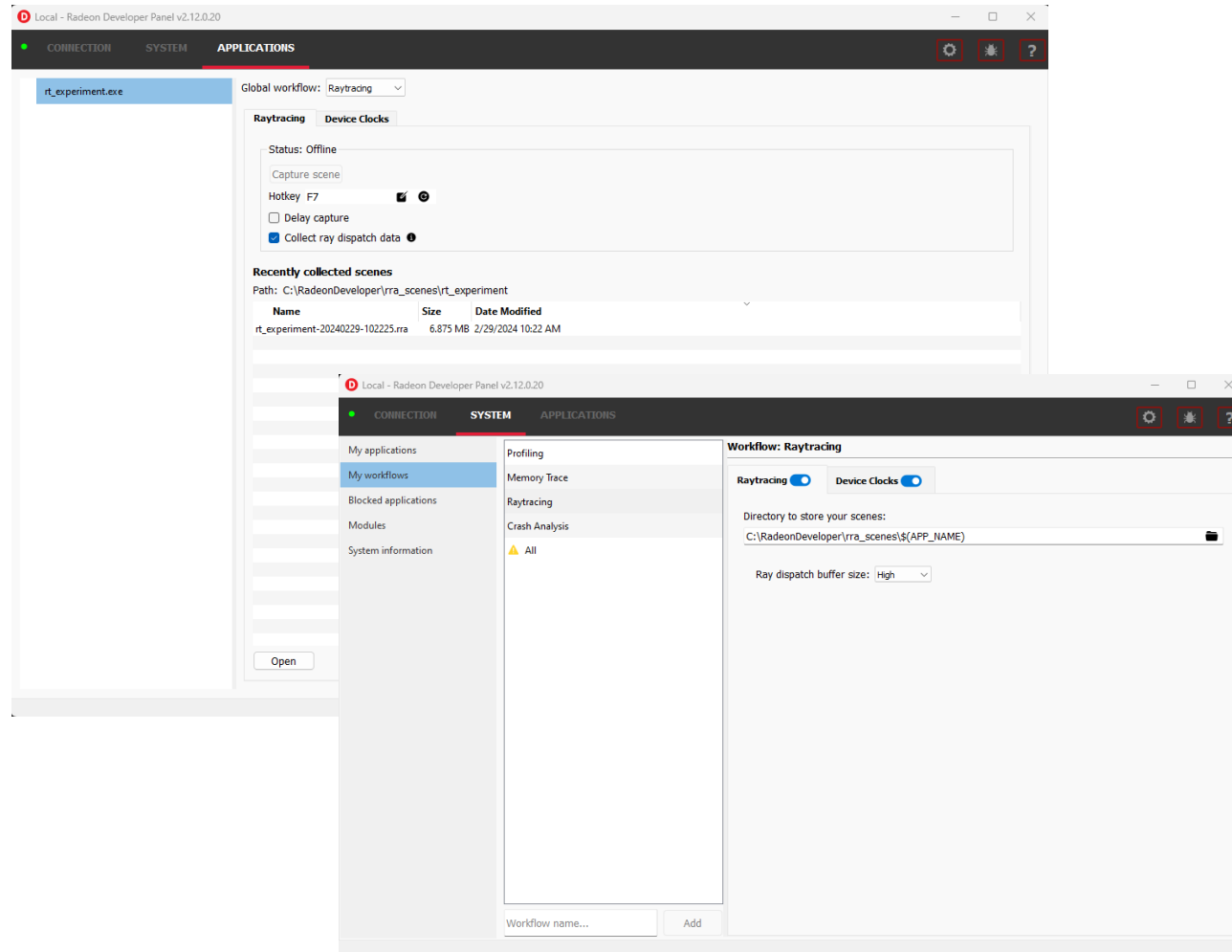
RRA 1.3 - NEW RAY FEATURES

- New ability to capture raytracing dispatches
- Captured through RDP by enabling "Collect ray dispatch data" option
- A buffer size option is needed to ensure all the data can be captured



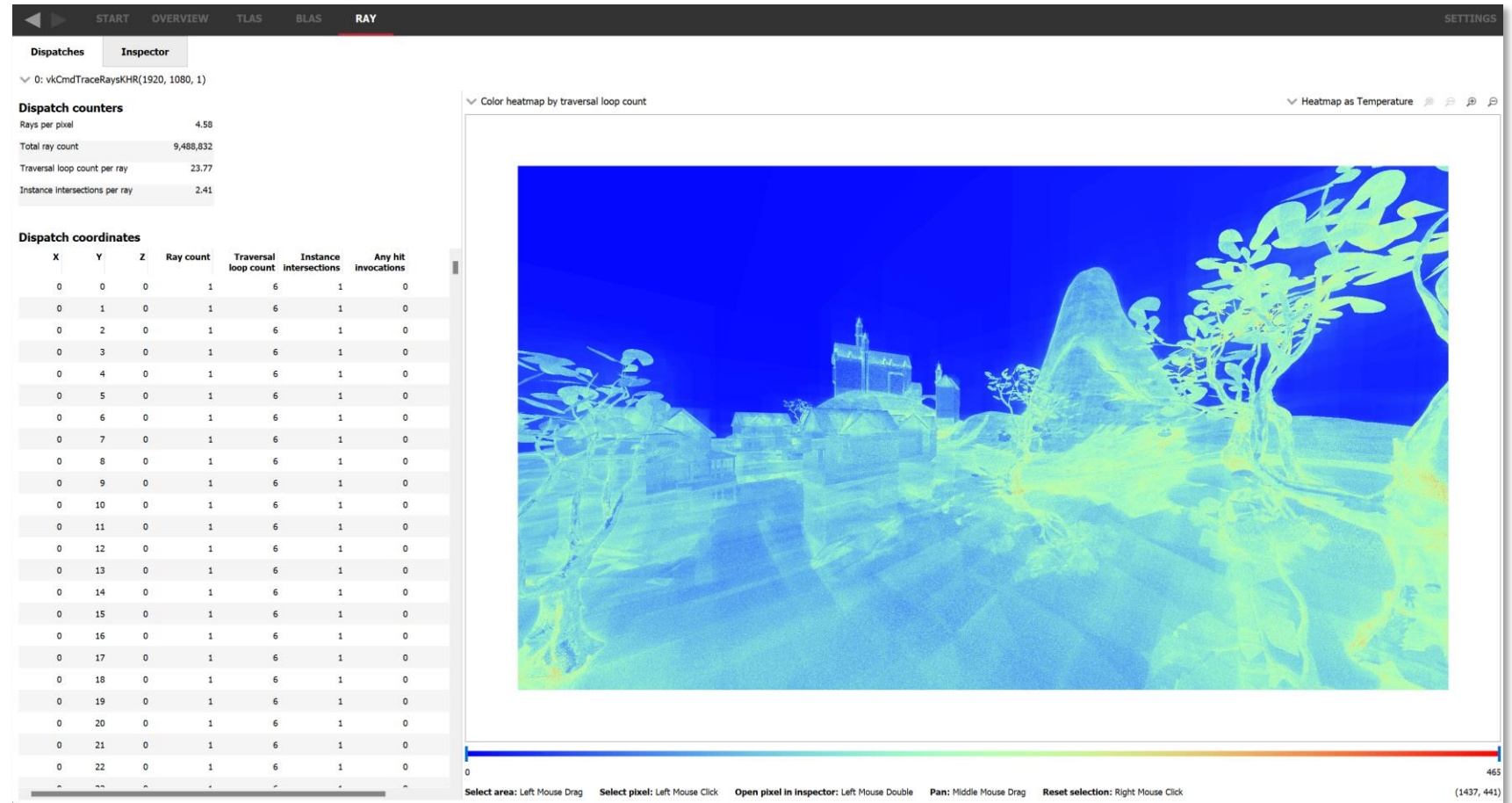
RRA 1.3 - NEW RAY FEATURES

- New ability to capture raytracing dispatches
- Captured through RDP by enabling "Collect ray dispatch data" option
- A buffer size option is needed to ensure all the data can be captured



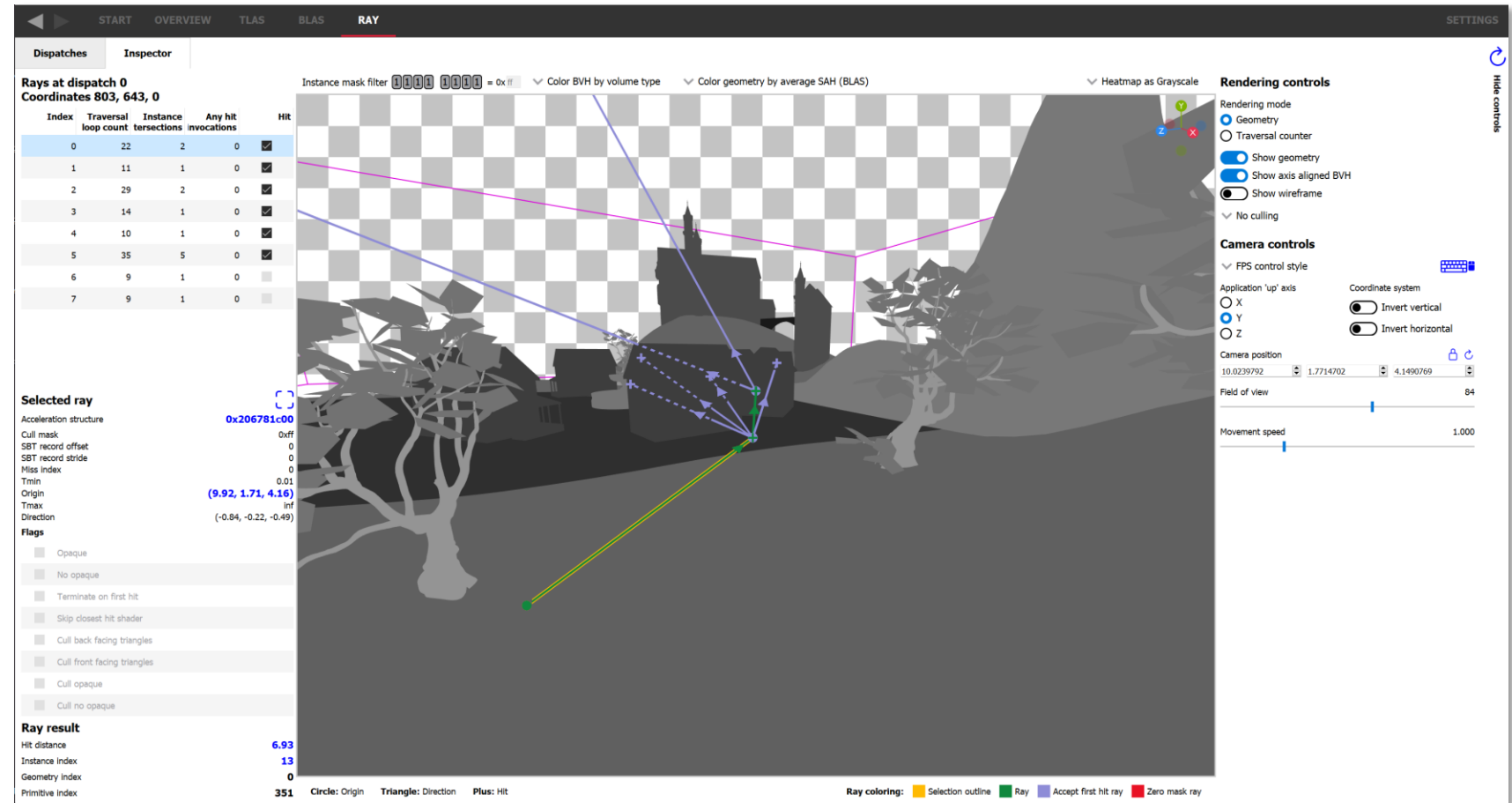
RRA 1.3 - DISPATCH

- Dispatch can be visualized into heatmap of various types if the dispatch shape is 2D or 3D
- Can also be mapped from 1D dispatches, must not be sparse
- Allows for exact traversal cost analysis since all the shadow and reflection rays will be included
- Select a "pixel" to inspect it



RRA 1.3 - RAY INSPECTOR

- View each ray for a given dispatch index
- See the cost and arguments for each ray



RRA 1.4

- New ray directions feature on the ray dispatches
- Quickly find and identify shadow and reflection areas
- Bugfixes and quality of life updates

Dispatches | **Inspector**

0: vkCmdTraceRaysKHR(1920, 1080, 1)

Dispatch counters

Rays per pixel	4.58
Total ray count	9,488,832
Traversal loop count per ray	23.77
Instance intersections per ray	2.41

Dispatch coordinates

X	Y	Z	Ray count	Traversal loop count	Instance intersections	Any hit invocations
803	643	0	8	139	14	0
803	644	0	8	146	17	0
803	645	0	8	140	15	0
803	646	0	8	170	17	0
803	647	0	8	144	14	0
803	648	0	8	144	15	0
803	649	0	8	145	14	0
803	650	0	8	149	13	0
803	651	0	7	156	15	0
803	652	0	7	159	16	0
803	653	0	8	160	13	0
803	654	0	8	183	15	0
803	655	0	8	204	21	0
803	656	0	8	187	15	0
803	657	0	8	173	14	0
803	658	0	8	174	13	0
803	659	0	8	197	15	0
803	660	0	8	185	17	0
803	661	0	8	172	15	0
803	662	0	8	189	14	0
803	663	0	8	168	14	0
803	664	0	8	198	16	0
803	665	0	8	184	16	0

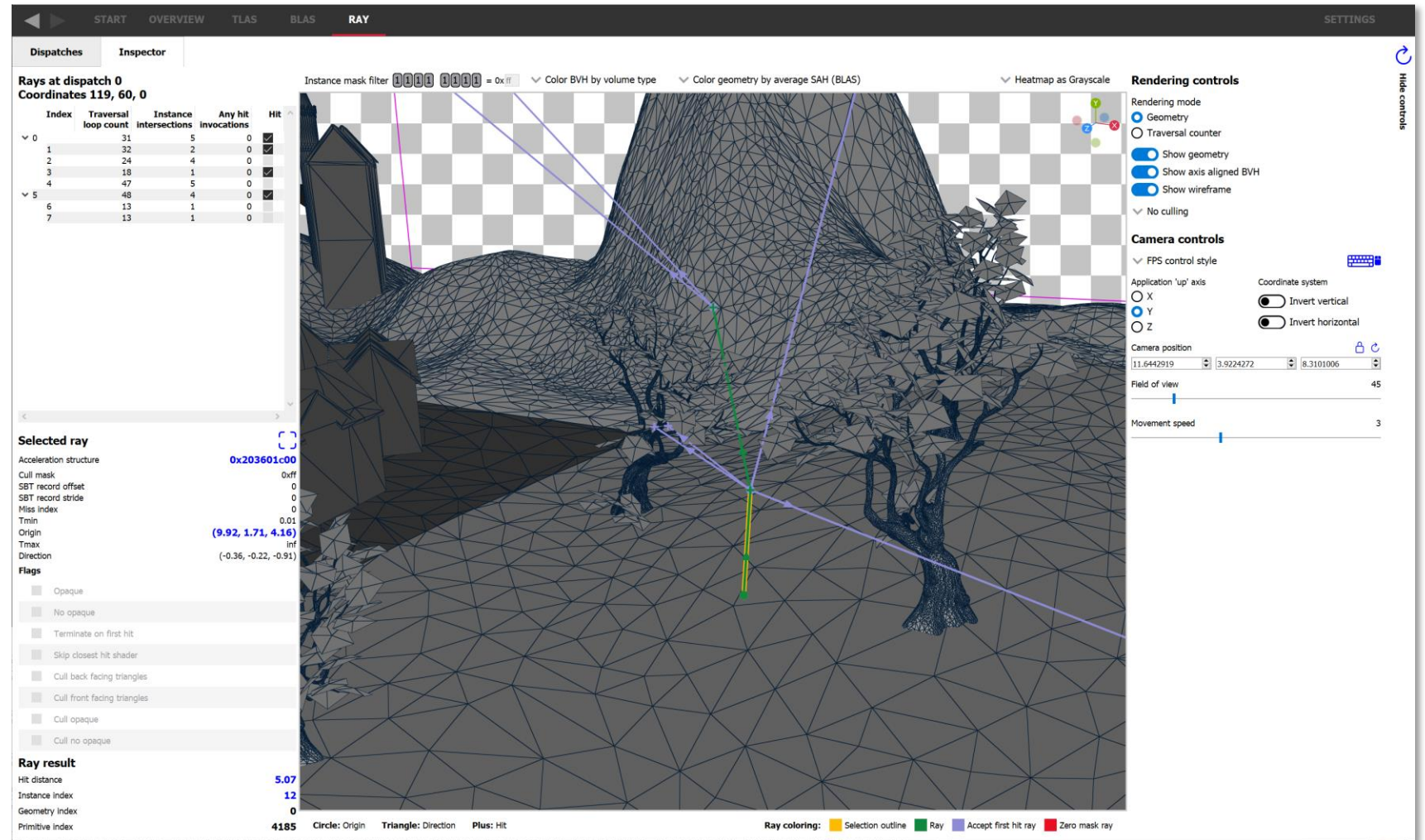
Color by ray direction

Select area: Left Mouse Drag | Select pixel: Left Mouse Click | Open pixel in inspector: Left Mouse Double | Pan: Middle Mouse Drag | Reset selection: Right Mouse Click

(1070, 500)

PREVIEW: RRA 1.5

- New ray hierarchy in Ray Inspector
- Quickly verify sampling rate and recursive calls



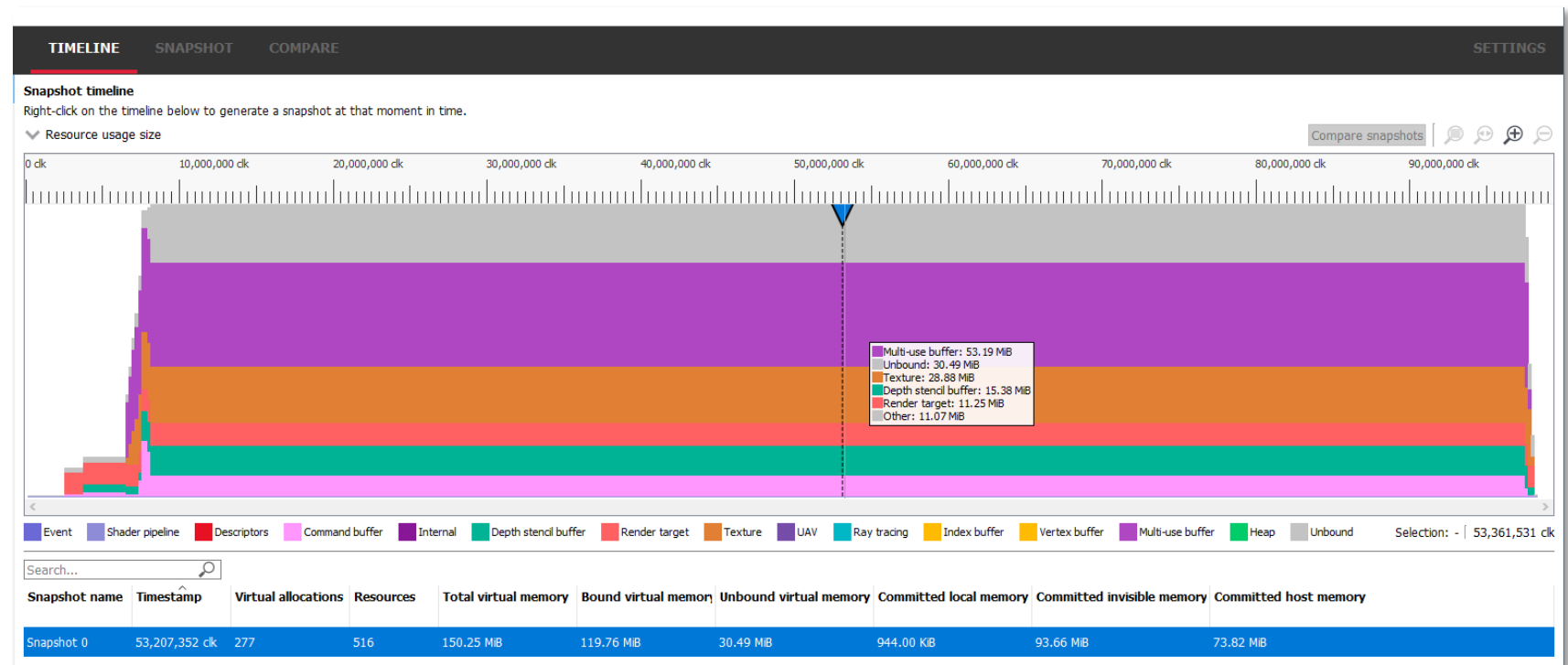
AMD

RADEON

Memory Visualizer

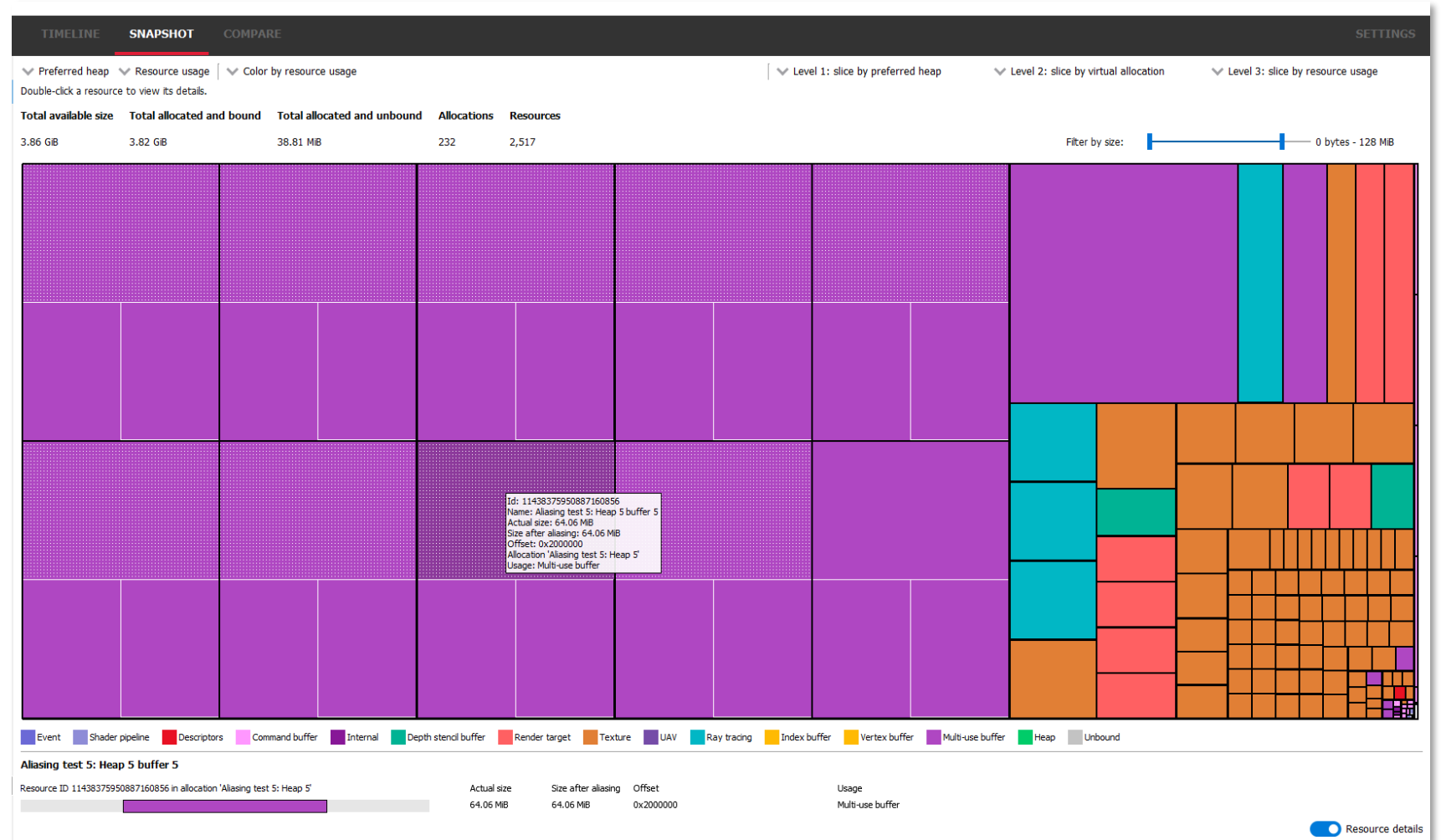
RMV TIMELINE IMPROVEMENTS

- Better memory visualization
- Unbound memory is now shown on timeline
- Usage sizes for aliased resources properly calculated
- Implicit resources filtered from calculations



RMV RESOURCE OVERVIEW IMPROVEMENTS

- Usage sizes for aliased resources properly calculated and shown in tooltip
- Named allocations
- Improved size range filter



OTHER RMV IMPROVEMENTS

- Heap overview pane contains additional information
- Device configuration pane shows system memory and driver info
- Support added for file format which supports compression
- New time unit format
- Expanded history resource columns

AMD

RADEON

GPU Detective

AMD RADEON™ GPU DETECTIVE (RGD)

- Newest member of AMD Radeon™ Developer Tool Suite (<https://gpuopen.com/tools/>)

Overview:

- Tool for post-mortem analysis of GPU crashes
- Sets driver to Crash Analysis mode before reproducing crash
- Developers capture AMD GPU Crash Dump files upon crash
- Produces concise crash analysis report in Text/JSON formats
- Report helps narrow down the search for the crash root cause

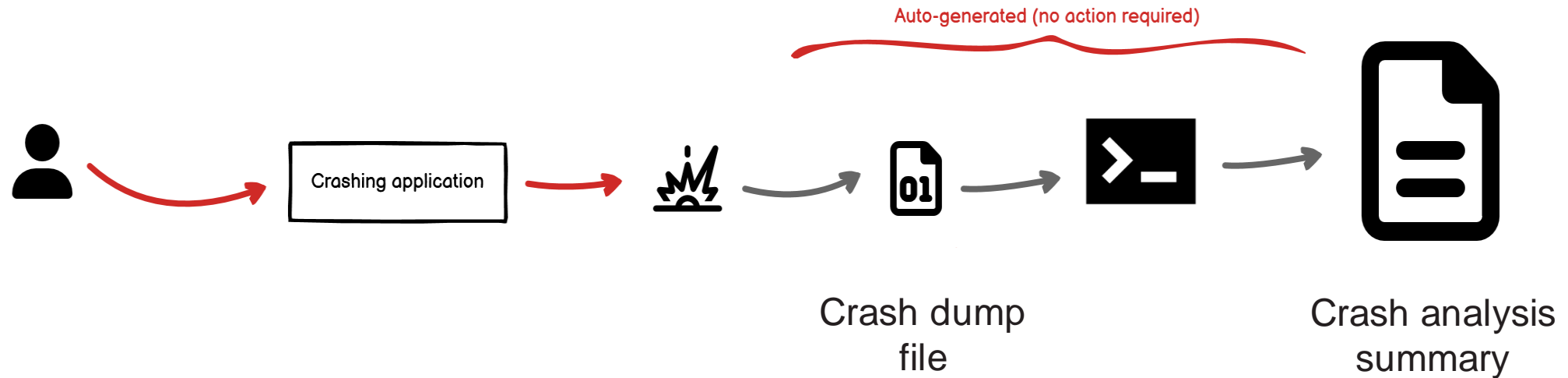
AMD RADEON™ GPU DETECTIVE (RGD)

- Newest member of AMD Radeon™ Developer Tool Suite (<https://gpuopen.com/tools/>)

Requirements:

- OS: Windows 10 or 11
- GPU: AMD Radeon™ RX 7000 Series (RDNA™ 3) or AMD Radeon™ RX 6000 Series (RDNA™ 2)
- Driver: AMD Software: Adrenalin Edition 23.12.1 or newer
- Graphics API used by the crashing application: **DirectX 12** or **Vulkan**

UNVEILING THE TOOL: WORKFLOW AND FUNCTIONAL INSIGHTS



UNVEILING THE TOOL: WORKFLOW AND FUNCTIONAL INSIGHTS

- Curious about the newest addition to our tool suite?
- Stay tuned for the next presentation

The slide features a large black triangle on the left side, separated from the white background by a red diagonal line. The AMD logo and tagline 'together we advance_' are in the top left. The GDC logo is in the top right. The title 'POST-MORTEM GPU CRASH ANALYSIS WITH AMD RADEON™ GPU DETECTIVE (RGD)' is centered. The speakers' names are listed on the right. The AMD GPUOpen logo is in the bottom right.

AMD
together we advance_

GDC

**POST-MORTEM GPU CRASH ANALYSIS
WITH
AMD RADEON™ GPU DETECTIVE (RGD)**

ADAM SAWICKI (AMD)
AMIT MULAY (AMD)
MARCO BOUTERSE (NIXXES SOFTWARE)

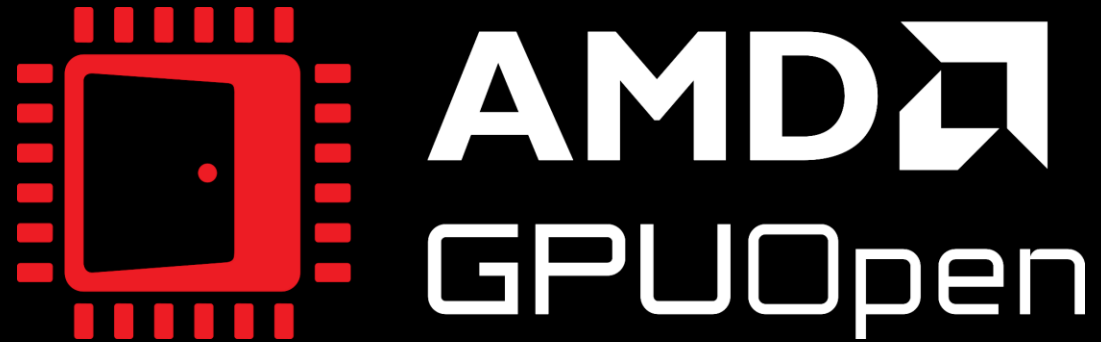
AMD
GPUOpen

DISCLAIMER

GENERAL DISCLAIMER

The information contained herein is for informational purposes only and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

© 2024 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, Radeon and combinations thereof are trademarks of Advanced Micro Devices, Inc. DirectX is a registered trademark of Microsoft Corporation in the US and other jurisdictions. Linux is a registered trademark of Linus Torvalds. OpenCL is a trademark of Apple, Inc. used by permission from The Khronos Group. LLVM is a trademark of LLVM Foundation. SPIR, SPIR-V and the SPIR logo are trademarks of the Khronos Group Inc. Vulkan and the Vulkan logo are registered trademarks of the Khronos Group Inc. Windows is a registered trademark of Microsoft Corporation in the US and other jurisdictions. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.



AMD 
together we advance_

AMD 
EPYC

AMD 
RYZEN

AMD 
RADEON