LET YOUR GAME SHINE – OPTIMIZING DIRECTX 12® AND VULKAN™ PERFORMANCE WITH AMD CODE XL

DORON OFEK

SENIOR MANAGER
AMD DEVELOPER TOOLS

AMD

*Enabling today.
Inspiring tomorrow.*

AGENDA

- What is CodeXL?
- New! Game Development features
- There’s so much more to CodeXL
- CodeXL on GPUOpen

WHAT IS CODEXL?

CODEXL

- CodeXL is a tool suite which helps SW developers get the best performance on AMD CPUs and GPUs
- Debug, Profile and Analyze applications
  - On local and remote hosts
- Multiple platforms and Operating Systems
  - Standalone application for Windows® and Linux®
  - Integrated into Microsoft® Visual Studio®
  - Linux and Windows feature parity
- Free to download and use

Debug

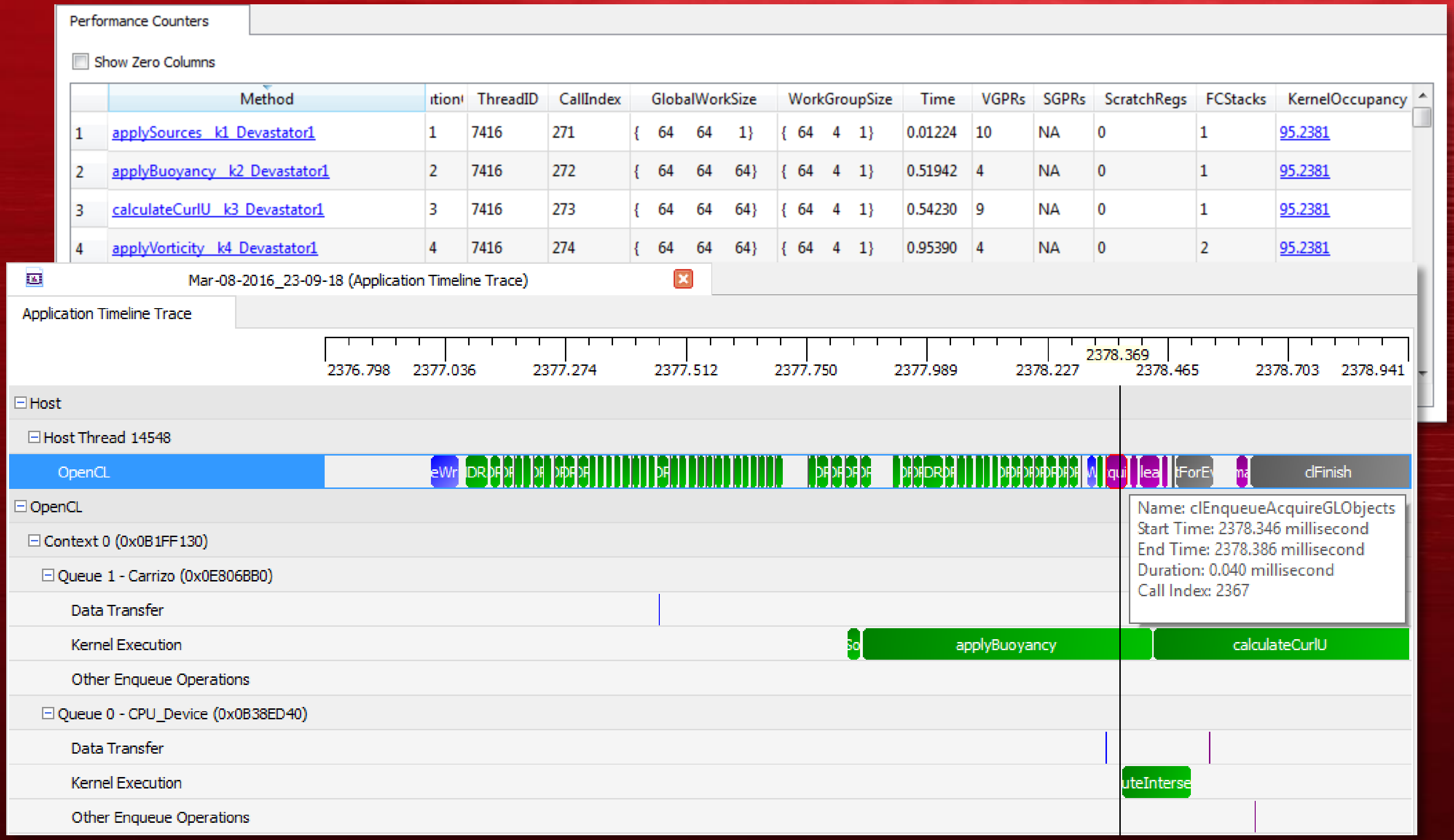
Profile

Frame Analysis

Build & Analyze

GPU COMPUTE PROFILING

- Profile OpenCL™ 1.2 and 2.0 kernels, and DirectCompute shaders
- Collect Application Trace
- Collect GPU Performance Counters
- OpenCL Timeline Visualization
- OpenCL Application Summary pages
- IL and ISA Display
- OpenCL Kernel Occupancy Viewer



Performance Counters

□ Show Zero Columns

	Method	ition	ThreadID	CallIndex	GlobalWorkSize	WorkGroupSize	Time	VGPRs	SGPRs	ScratchRegs	FCStacks	KernelOccupancy
1	applySources__k1_Devastator1	1	7416	271	{ 64 64 1}	{ 64 4 1}	0.01224	10	NA	0	1	95.2381
2	applyBuoyancy__k2_Devastator1	2	7416	272	{ 64 64 64}	{ 64 4 1}	0.51942	4	NA	0	1	95.2381
3	calculateCurlU__k3_Devastator1	3	7416	273	{ 64 64 64}	{ 64 4 1}	0.54230	9	NA	0	1	95.2381
4	applyVorticity__k4_Devastator1	4	7416	274	{ 64 64 64}	{ 64 4 1}	0.95390	4	NA	0	2	95.2381

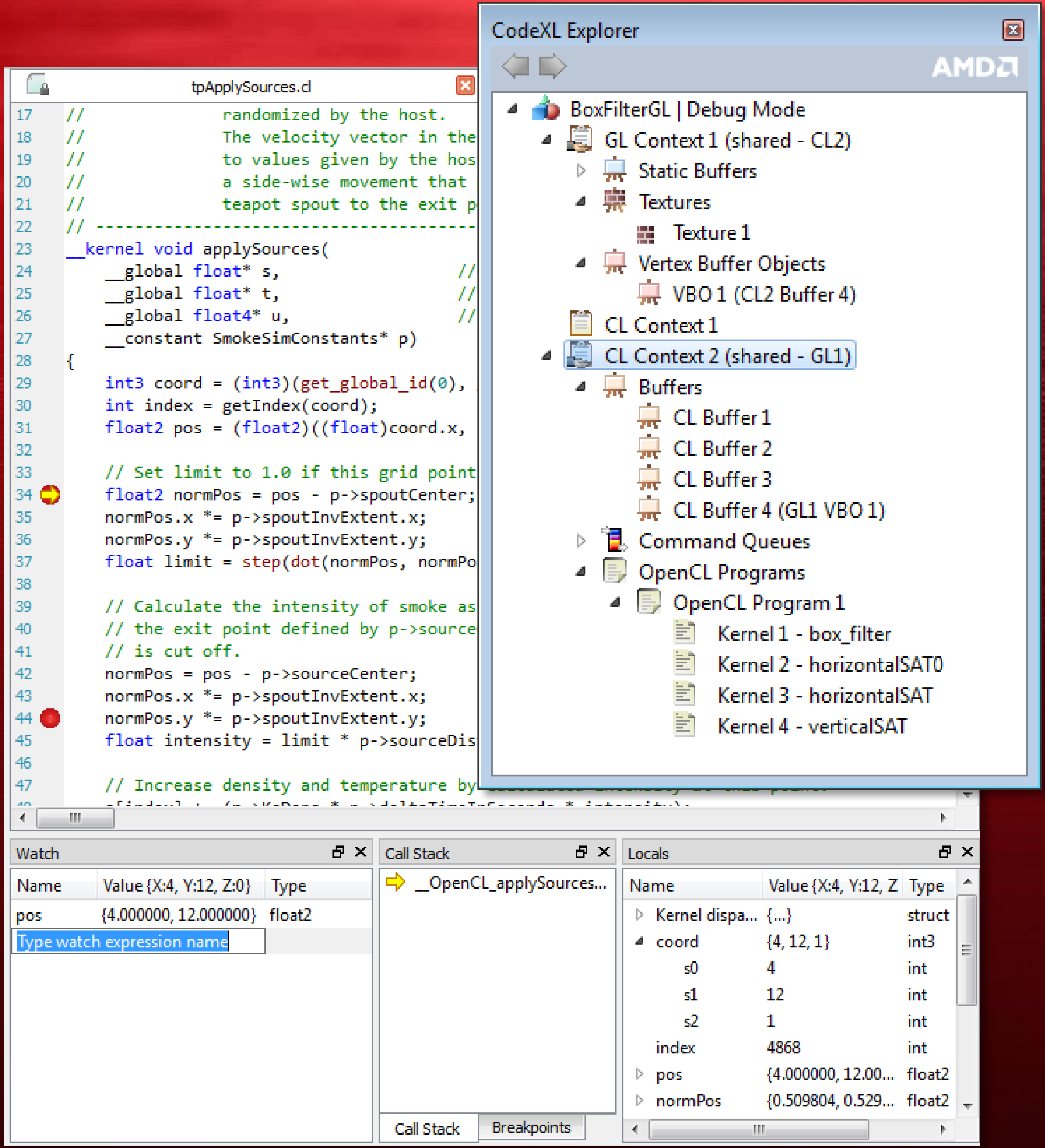
Mar-08-2016_23-09-18 (Application Timeline Trace)

Application Timeline Trace

Name: clEnqueueAcquireGLObjects
Start Time: 2378.346 millisecond
End Time: 2378.386 millisecond
Duration: 0.040 millisecond
Call Index: 2367

HOST AND GPU DEBUGGING

- Out-of-the-Box Debugging
- Debugging of C/C++host code and OpenCL 1.2 kernels
- OpenCL and OpenGL API-level Debugging
- API Statistics
- Object Visualization



tpApplySources.cl

```
//                randomized by the host.
//                The velocity vector in the
//                to values given by the hos
//                a side-wise movement that
//                teapot spout to the exit p
// -----------------------------------------
__kernel void applySources(
    __global float* s,                    //
    __global float* t,                    //
    __global float4* u,                   //
    __constant SmokeSimConstants* p)
{
    int3 coord = (int3)(get_global_id(0),
    int index = getIndex(coord);
    float2 pos = (float2)((float)coord.x,

    // Set limit to 1.0 if this grid point
    float2 normPos = pos - p->spoutCenter;
    normPos.x *= p->spoutInvExtent.x;
    normPos.y *= p->spoutInvExtent.y;
    float limit = step(dot(normPos, normPo

    // Calculate the intensity of smoke as
    // the exit point defined by p->source
    // is cut off.
    normPos = pos - p->sourceCenter;
    normPos.x *= p->spoutInvExtent.x;
    normPos.y *= p->spoutInvExtent.y;
    float intensity = limit * p->sourceDis

    // Increase density and temperature by
```

CodeXL Explorer

- BoxFilterGL | Debug Mode
  - GL Context 1 (shared - CL2)
    - Static Buffers
    - Textures
      - Texture 1
    - Vertex Buffer Objects
      - VBO 1 (CL2 Buffer 4)
  - CL Context 1
  - CL Context 2 (shared - GL1)
    - Buffers
      - CL Buffer 1
      - CL Buffer 2
      - CL Buffer 3
      - CL Buffer 4 (GL1 VBO 1)
    - Command Queues
    - OpenCL Programs
      - OpenCL Program 1
        - Kernel 1 - box_filter
        - Kernel 2 - horizontalSAT0
        - Kernel 3 - horizontalSAT
        - Kernel 4 - verticalSAT

Watch

Name	Value {X:4, Y:12, Z:0}	Type
pos	{4.000000, 12.000000}	float2
Type watch expression name		

Call Stack

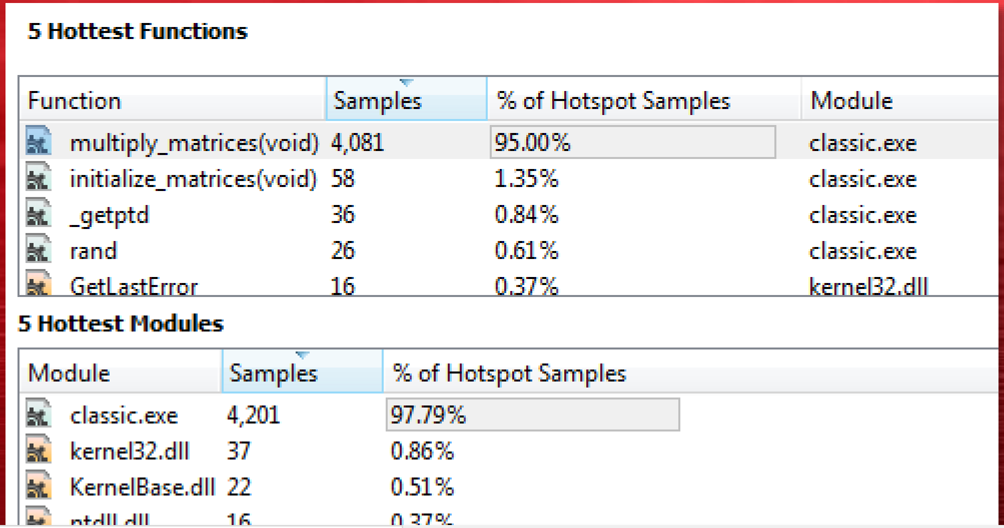
_OpenCL_applySources...

Locals

Name	Value {X:4, Y:12, Z	Type
Kernel dispa...	{...}	struct
coord	{4, 12, 1}	int3
s0	4	int
s1	12	int
s2	1	int
index	4868	int
pos	{4.000000, 12.00...	float2
normPos	{0.509804, 0.529...	float2

CPU PROFILING

- Detect Hotspots on any x86 platform
- Profile single application or whole system
- Profile applications and drivers
- Profile native code, Java and .NET applications
- Advanced Diagnosis of Performance Issues on AMD Platforms
- Analyze Call Chain Relationships
- Drill Down to the Source Code Line and ISA Instruction Level
- CPU Events Monitoring
- Instruction Based Sampling

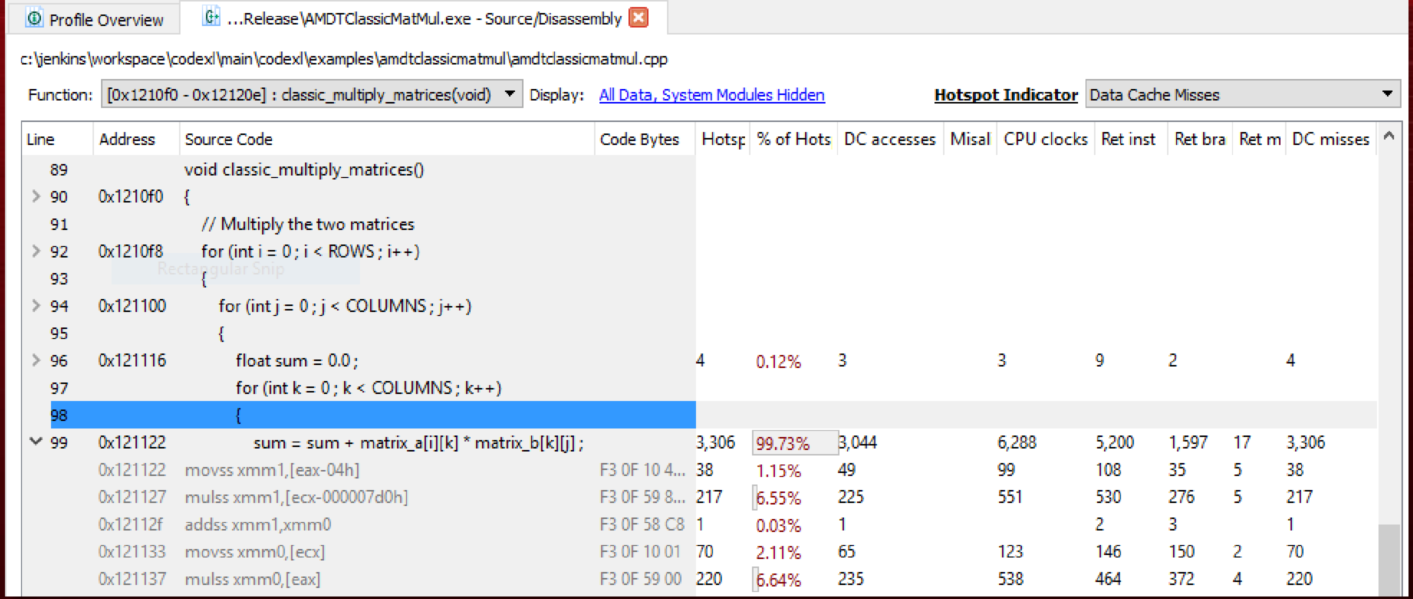


5 Hottest Functions

Function	Samples	% of Hotspot Samples	Module
multiply_matrices(void)	4,081	95.00%	classic.exe
initialize_matrices(void)	58	1.35%	classic.exe
_getptd	36	0.84%	classic.exe
rand	26	0.61%	classic.exe
GetLastError	16	0.37%	kernel32.dll

5 Hottest Modules

Module	Samples	% of Hotspot Samples
classic.exe	4,201	97.79%
kernel32.dll	37	0.86%
KernelBase.dll	22	0.51%
ntdll.dll	16	0.37%



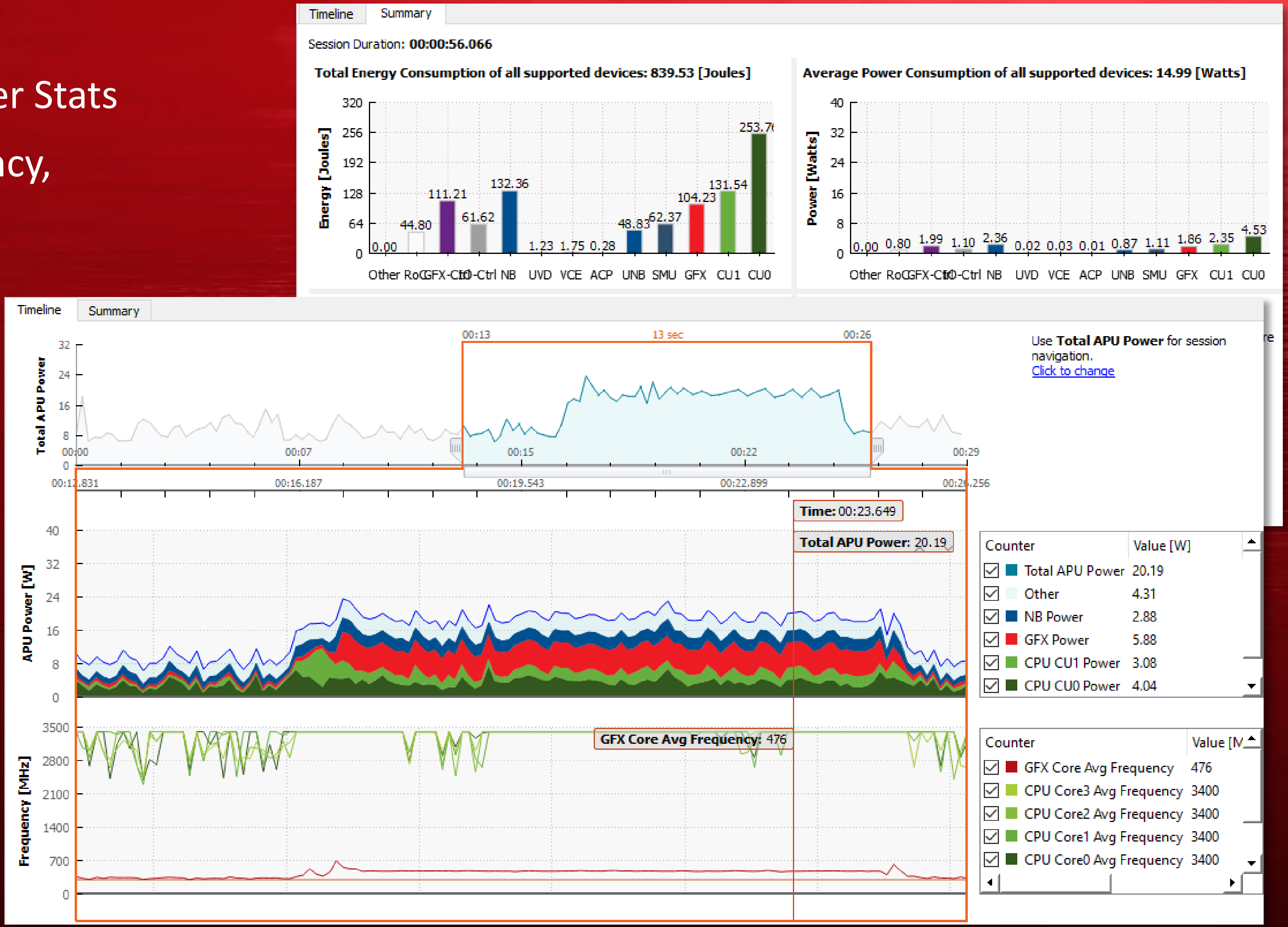
c:\jenkins\workspace\codexl\main\codexl\examples\amdtclassicmatmul\amdtclassicmatmul.cpp

Function: [0x1210f0 - 0x12120e] : classic_multiply_matrices(void) Display: All Data, System Modules Hidden **Hotspot Indicator** Data Cache Misses

Line	Address	Source Code	Code Bytes	Hotsp	% of Hots	DC accesses	Misal	CPU clocks	Ret inst	Ret bra	Ret m	DC misses
89		void classic_multiply_matrices()										
90	0x1210f0	{										
91		// Multiply the two matrices										
92	0x1210f8	for (int i = 0 ; i < ROWS ; i++)										
93		{										
94	0x121100	for (int j = 0 ; j < COLUMNS ; j++)										
95		{										
96	0x121116	float sum = 0.0 ;		4	0.12%	3		3	9	2		4
97		for (int k = 0 ; k < COLUMNS ; k++)										
98		{										
99	0x121122	sum = sum + matrix_a[i][k] * matrix_b[k][j] ;		3,306	99.73%	3,044		6,288	5,200	1,597	17	3,306
	0x121122	movss xmm1,[eax-04h]	F3 0F 10 4...	38	1.15%	49		99	108	35	5	38
	0x121127	mulss xmm1,[ecx-000007d0h]	F3 0F 59 8...	217	6.55%	225		551	530	276	5	217
	0x12112f	addss xmm1,xmm0	F3 0F 58 C8	1	0.03%	1			2	3		1
	0x121133	movss xmm0,[ecx]	F3 0F 10 01	70	2.11%	65		123	146	150	2	70
	0x121137	mulss xmm0,[eax]	F3 0F 59 00	220	6.64%	235		538	464	372	4	220

POWER PROFILING

- Online Capture and Display of Power Stats
- Real-Time Monitor: Power, Frequency, Temperature and Core state
- Cumulative and Average Power Consumption Histograms
- Profile CPUs, APUs and dGPUs
- Command-Line Tool and Graphic Application



Session Duration: **00:00:56.066**

Total Energy Consumption of all supported devices: 839.53 [Joules]

Average Power Consumption of all supported devices: 14.99 [Watts]

Use **Total APU Power** for session navigation.
Click to change

Time: 00:23.649

Total APU Power: 20.19

GFX Core Avg Frequency: 476

	Counter	Value [W]
☑	🟦 Total APU Power	20.19
☑	⬜ Other	4.31
☑	🟦 NB Power	2.88
☑	🟥 GFX Power	5.88
☑	🟩 CPU CU1 Power	3.08
☑	🟩 CPU CU0 Power	4.04

	Counter	Value [M
☑	🟥 GFX Core Avg Frequency	476
☑	🟩 CPU Core3 Avg Frequency	3400
☑	🟩 CPU Core2 Avg Frequency	3400
☑	🟩 CPU Core1 Avg Frequency	3400
☑	🟩 CPU Core0 Avg Frequency	3400

Screenshot of a power profiling session with Microsoft SDK D3D12Multithreading on a Carrizo A12 laptop

GAME DEVELOPMENT WITH CODEXL

AMD

Enabling today.
Inspiring tomorrow.

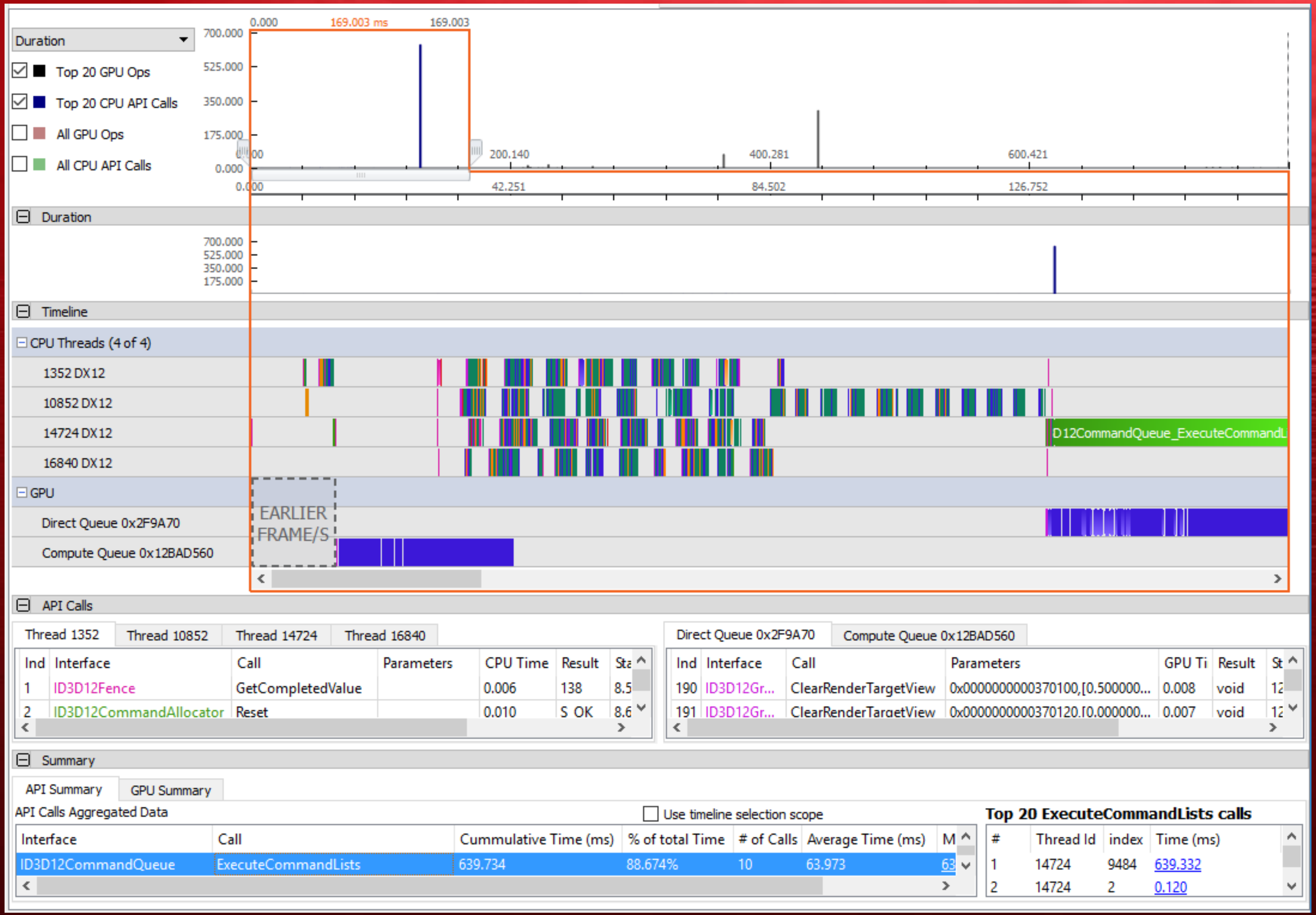
CODEXL 2.0 GAME DEVELOPMENT INITIAL RELEASE

BETA

- DirectX 12 Frame Analysis with Timeline Trace
- Static Analysis of Vulkan programs

DIRECTX 12 TIMELINE TRACE

- Capture individual frames from Microsoft DirectX12 games/applications
- Run the game locally or on a remote host
- Linked trace of API calls and GPU ops, co-referenced
- API Statistics
- Timeline visualization
- Easy navigation
- Immediate focus on longest operations



Ind	Interface	Call	Parameters	CPU Time	Result	Sta
1	ID3D12Fence	GetCompletedValue		0.006	138	8.5
2	ID3D12CommandAllocator	Reset		0.010	S OK	8.6

Ind	Interface	Call	Parameters	GPU Ti	Result	St
190	ID3D12Gr...	ClearRenderTargetView	0x0000000000370100,[0.500000...	0.008	void	12
191	ID3D12Gr...	ClearRenderTargetView	0x0000000000370120,[0.000000...	0.007	void	12

API Calls Aggregated Data

□ Use timeline selection scope

Interface	Call	Cummulative Time (ms)	% of total Time	# of Calls	Average Time (ms)	M
ID3D12CommandQueue	ExecuteCommandLists	639.734	88.674%	10	63.973	63

Top 20 ExecuteCommandLists calls

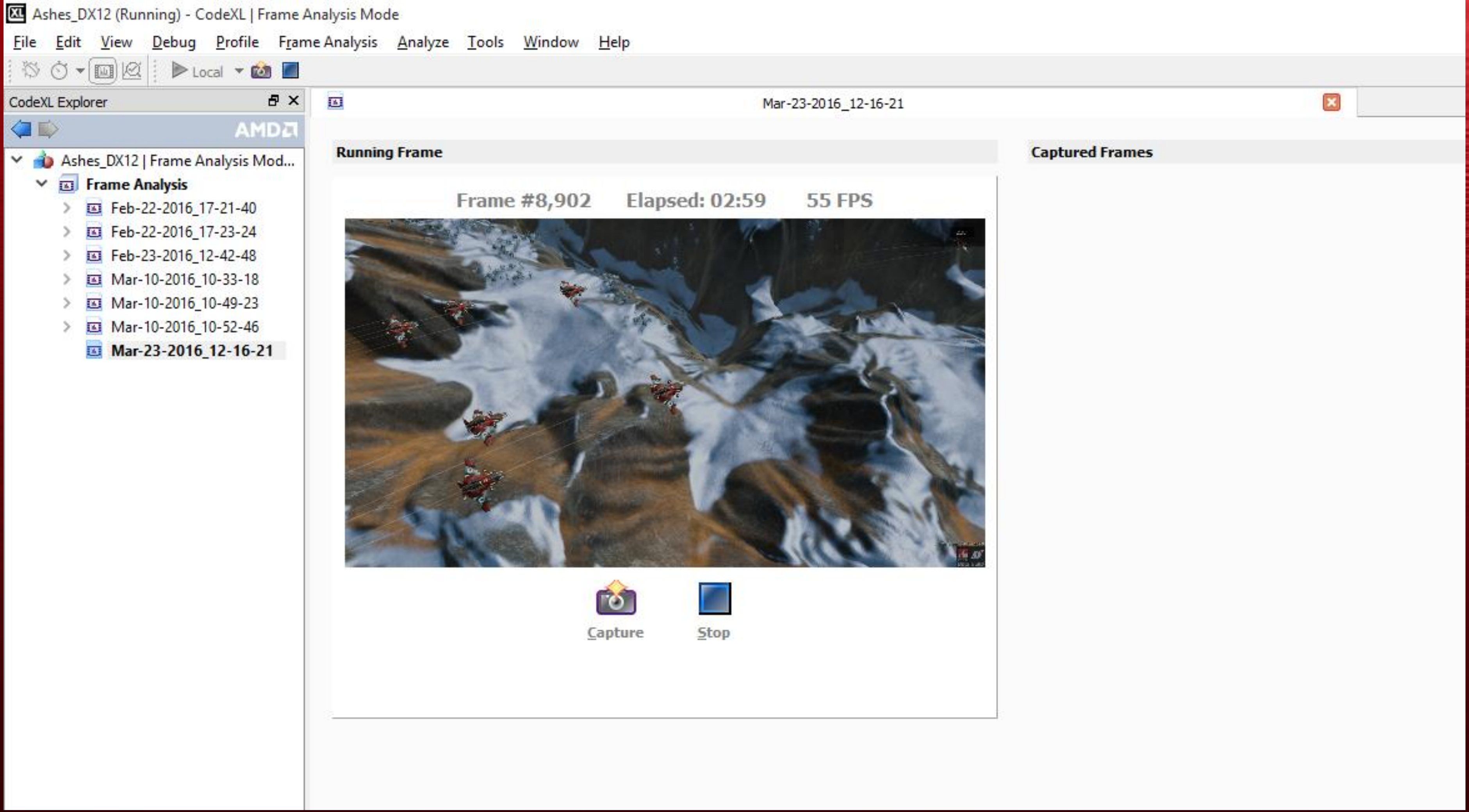
#	Thread Id	index	Time (ms)
1	14724	9484	639.332
2	14724	2	0.120

CODEXL DIRECTX 12 FRAME ANALYSIS DEMO

Many thanks to Dan Baker for approving the use of ‘Ashes of the Singularity’ in this demo

DEMO – FRAME ANALYSIS (1)

- I set the path to the game executable and click the Play button to begin a frame analysis session.
- CodeXL launches the game and continuously displays the current frame image, elapsed time, frame number and FPS rate.

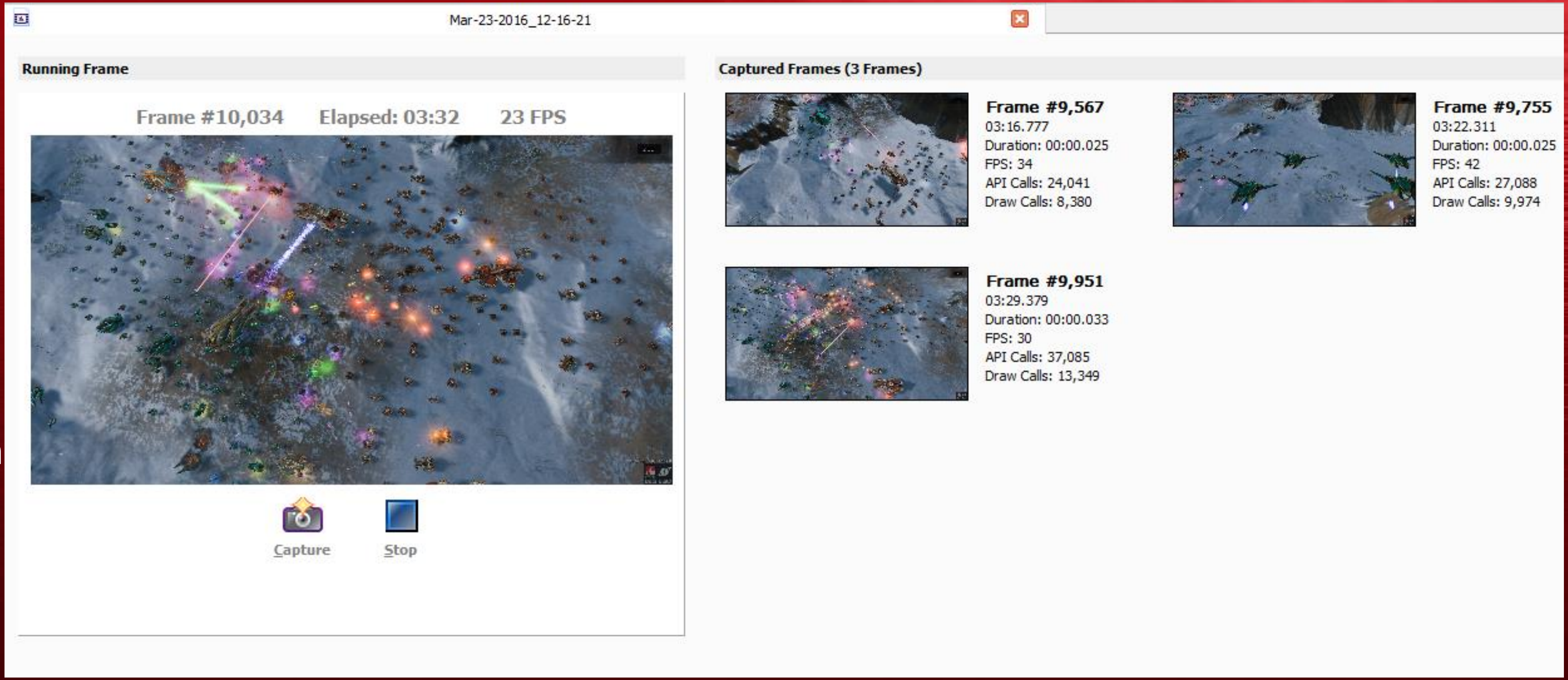


Ashes_DX12 (Running) - CodeXL | Frame Analysis Mode

Frame #8,902 Elapsed: 02:59 55 FPS

DEMO – FRAME ANALYSIS (2)

- I click the Capture button 3 times to capture 3 frame traces.
- The thumbnail of each captured frame is displayed in the Captured Frames section, together with the frame duration, FPS rate and number of API calls.



Mar-23-2016_12-16-21

Running Frame

Frame #10,034 Elapsed: 03:32 23 FPS

Captured Frames (3 Frames)

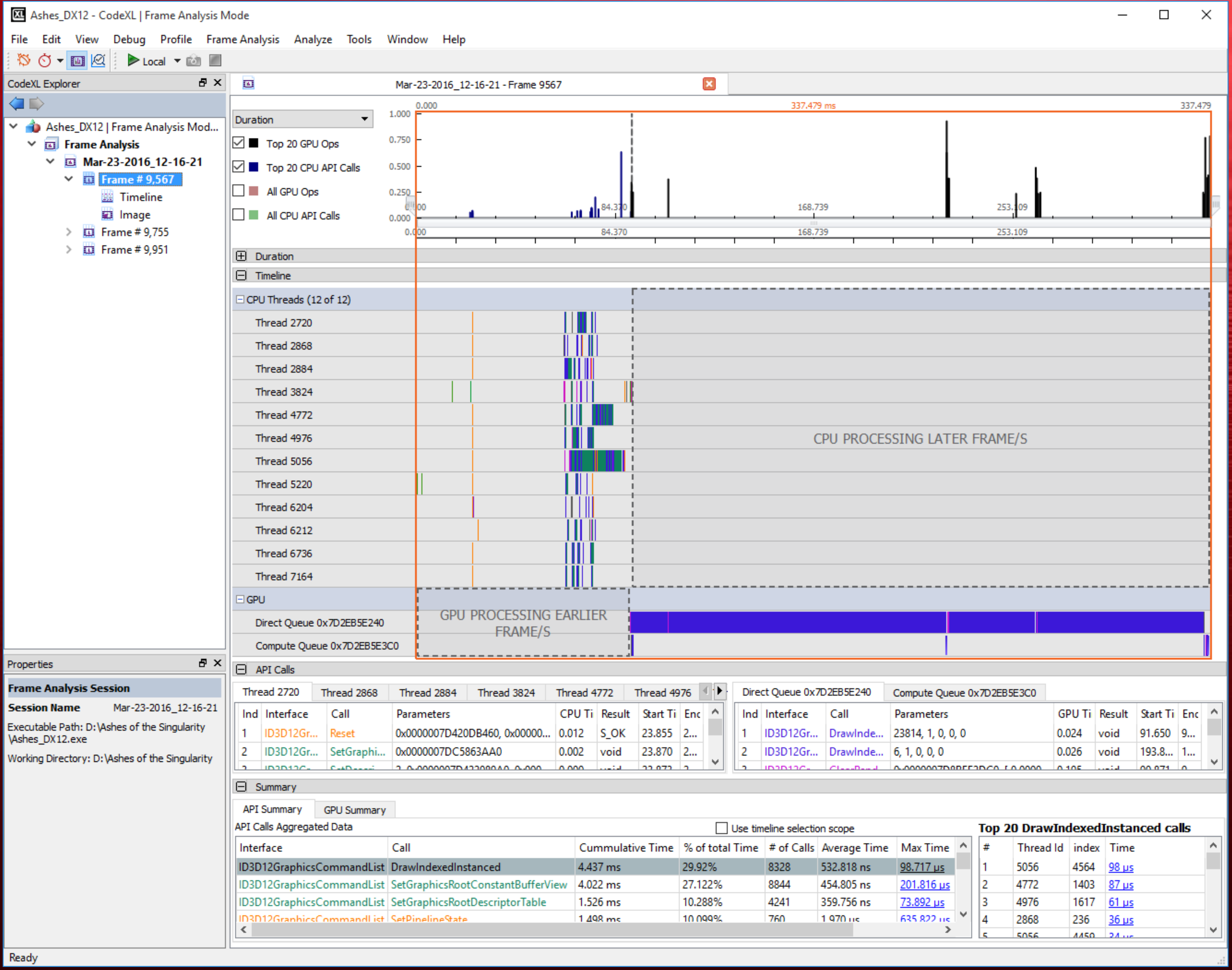
Frame #9,567
03:16.777
Duration: 00:00.025
FPS: 34
API Calls: 24,041
Draw Calls: 8,380

Frame #9,755
03:22.311
Duration: 00:00.025
FPS: 42
API Calls: 27,088
Draw Calls: 9,974

Frame #9,951
03:29.379
Duration: 00:00.033
FPS: 30
API Calls: 37,085
Draw Calls: 13,349

DEMO – FRAME ANALYSIS (3)

- Double-clicking a frame opens the frame timeline view.
- The navigation ribbon on top presents the complete frame time scope, and the top 20 CPU and GPU API calls.
- The timeline chart displays all of the calls executed by each CPU thread and each GPU Queue.
- The API Calls ribbon displays tables with the details of each API call, complete with timing and parameter values.
- Clicking a GPU call highlights the corresponding CPU call.



Frame Analysis Session

Session Name	Mar-23-2016_12-16-21

Executable Path: D:\Ashes of the Singularity\Ashes_DX12.exe

Working Directory: D:\Ashes of the Singularity

Thread 2720

Ind	Interface	Call	Parameters	CPU Ti	Result	Start Ti	Enc
1	ID3D12Gr...	Reset	0x0000007D420DB460, 0x00000...	0.012	S_OK	23.855	2...
2	ID3D12Gr...	SetGraphi...	0x0000007DC5863AA0	0.002	void	23.870	2...

Direct Queue 0x7D2EB5E240

Ind	Interface	Call	Parameters	GPU Ti	Result	Start Ti	Enc
1	ID3D12Gr...	DrawInde...	23814, 1, 0, 0, 0	0.024	void	91.650	9...
2	ID3D12Gr...	DrawInde...	6, 1, 0, 0, 0	0.026	void	193.8...	1...

API Calls Aggregated Data

□ Use timeline selection scope

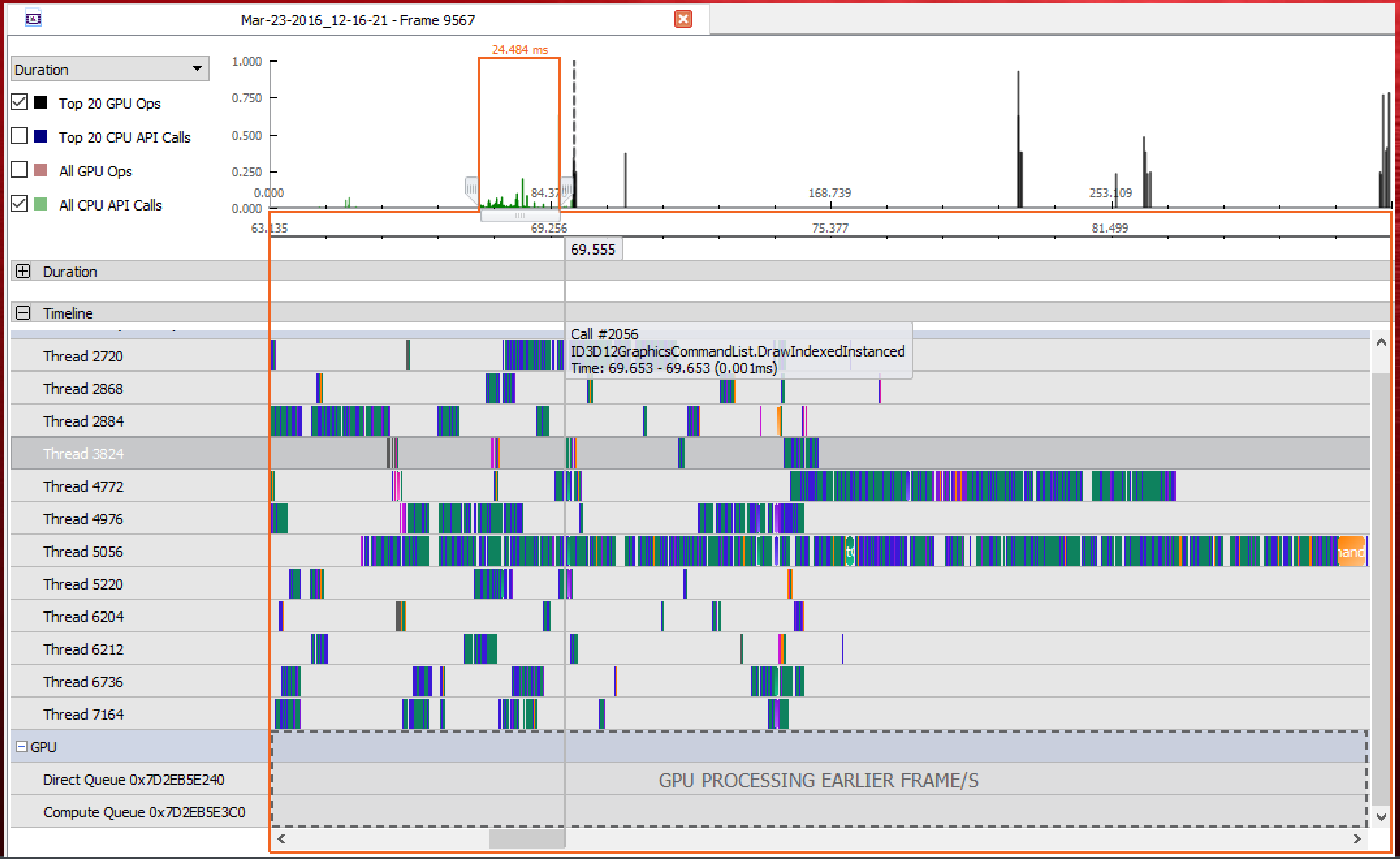
Interface	Call	Cummulative Time	% of total Time	# of Calls	Average Time	Max Time
ID3D12GraphicsCommandList	DrawIndexedInstanced	4.437 ms	29.92%	8328	532.818 ns	98.717 µs
ID3D12GraphicsCommandList	SetGraphicsRootConstantBufferView	4.022 ms	27.122%	8844	454.805 ns	201.816 µs
ID3D12GraphicsCommandList	SetGraphicsRootDescriptorTable	1.526 ms	10.288%	4241	359.756 ns	73.892 µs
ID3D12GraphicsCommandList	SetPipelineState	1.498 ms	10.099%	760	1.970 µs	635.822 µs

Top 20 DrawIndexedInstanced calls

#	Thread Id	index	Time
1	5056	4564	98 µs
2	4772	1403	87 µs
3	4976	1617	61 µs
4	2868	236	36 µs
5	5056	4450	34 µs

DEMO – FRAME ANALYSIS (4)

- The handles on the navigation ribbon can be used to focus on a specific part of the frame timeline.
- The timeline ribbon is synchronized with the navigation ribbon and displays the same selected part of the frame timeline



Mar-23-2016_12-16-21 - Frame 9567

Duration

☑ Top 20 GPU Ops

☐ Top 20 CPU API Calls

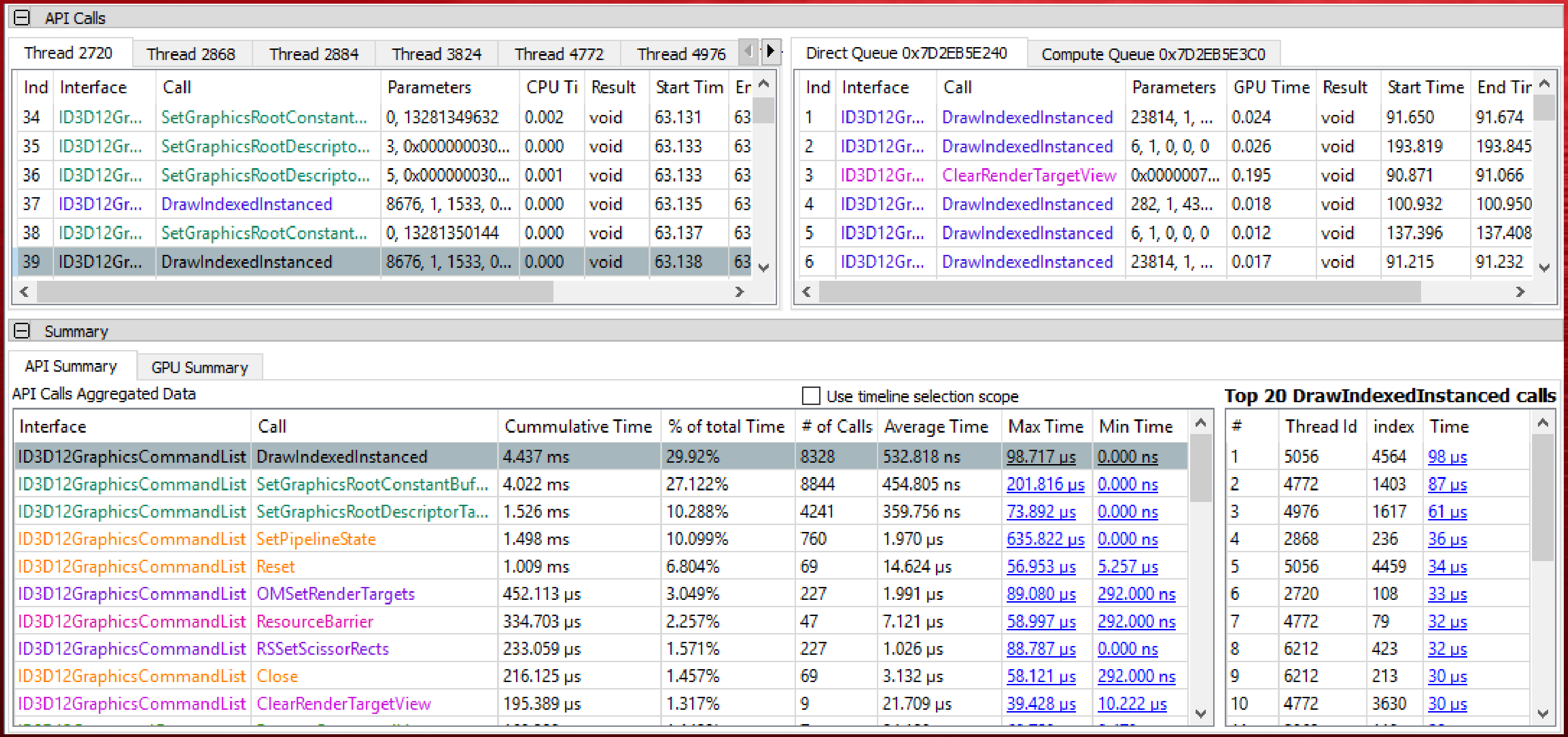
☐ All GPU Ops

☑ All CPU API Calls

Call #2056
ID3D12GraphicsCommandList.DrawIndexedInstanced
Time: 69.653 - 69.653 (0.001ms)

DEMO – FRAME ANALYSIS (5)

- The API Calls tables are synchronized – clicking a CPU call automatically highlights the corresponding GPU call and vice versa.
- The Summary ribbon displays aggregated data for each type of API call – the cumulative time of all the calls of that particular API as well as the longest and shortest call and additional statistics.
- The top 20 calls of the selected API are displayed in the 'Top 20' table on the right.



Ind	Interface	Call	Parameters	CPU Ti	Result	Start Tim	Er
34	ID3D12Gr...	SetGraphicsRootConstant...	0, 13281349632	0.002	void	63.131	63
35	ID3D12Gr...	SetGraphicsRootDescripto...	3, 0x000000030...	0.000	void	63.133	63
36	ID3D12Gr...	SetGraphicsRootDescripto...	5, 0x000000030...	0.001	void	63.133	63
37	ID3D12Gr...	DrawIndexedInstanced	8676, 1, 1533, 0...	0.000	void	63.135	63
38	ID3D12Gr...	SetGraphicsRootConstant...	0, 13281350144	0.000	void	63.137	63
39	ID3D12Gr...	DrawIndexedInstanced	8676, 1, 1533, 0...	0.000	void	63.138	63

Ind	Interface	Call	Parameters	GPU Time	Result	Start Time	End Tir
1	ID3D12Gr...	DrawIndexedInstanced	23814, 1, ...	0.024	void	91.650	91.674
2	ID3D12Gr...	DrawIndexedInstanced	6, 1, 0, 0, 0	0.026	void	193.819	193.845
3	ID3D12Gr...	ClearRenderTargetView	0x0000007...	0.195	void	90.871	91.066
4	ID3D12Gr...	DrawIndexedInstanced	282, 1, 43...	0.018	void	100.932	100.950
5	ID3D12Gr...	DrawIndexedInstanced	6, 1, 0, 0, 0	0.012	void	137.396	137.408
6	ID3D12Gr...	DrawIndexedInstanced	23814, 1, ...	0.017	void	91.215	91.232

API Calls Aggregated Data

☐ Use timeline selection scope

Interface	Call	Cummulative Time	% of total Time	# of Calls	Average Time	Max Time	Min Time
ID3D12GraphicsCommandList	DrawIndexedInstanced	4.437 ms	29.92%	8328	532.818 ns	98.717 µs	0.000 ns
ID3D12GraphicsCommandList	SetGraphicsRootConstantBuf...	4.022 ms	27.122%	8844	454.805 ns	201.816 µs	0.000 ns
ID3D12GraphicsCommandList	SetGraphicsRootDescriptorTa...	1.526 ms	10.288%	4241	359.756 ns	73.892 µs	0.000 ns
ID3D12GraphicsCommandList	SetPipelineState	1.498 ms	10.099%	760	1.970 µs	635.822 µs	0.000 ns
ID3D12GraphicsCommandList	Reset	1.009 ms	6.804%	69	14.624 µs	56.953 µs	5.257 µs
ID3D12GraphicsCommandList	OMSetRenderTargets	452.113 µs	3.049%	227	1.991 µs	89.080 µs	292.000 ns
ID3D12GraphicsCommandList	ResourceBarrier	334.703 µs	2.257%	47	7.121 µs	58.997 µs	292.000 ns
ID3D12GraphicsCommandList	RSSetScissorRects	233.059 µs	1.571%	227	1.026 µs	88.787 µs	0.000 ns
ID3D12GraphicsCommandList	Close	216.125 µs	1.457%	69	3.132 µs	58.121 µs	292.000 ns
ID3D12GraphicsCommandList	ClearRenderTargetView	195.389 µs	1.317%	9	21.709 µs	39.428 µs	10.222 µs

Top 20 DrawIndexedInstanced calls

#	Thread Id	index	Time
1	5056	4564	98 µs
2	4772	1403	87 µs
3	4976	1617	61 µs
4	2868	236	36 µs
5	5056	4459	34 µs
6	2720	108	33 µs
7	4772	79	32 µs
8	6212	423	32 µs
9	6212	213	30 µs
10	4772	3630	30 µs

END OF FRAME ANALYSIS DEMO

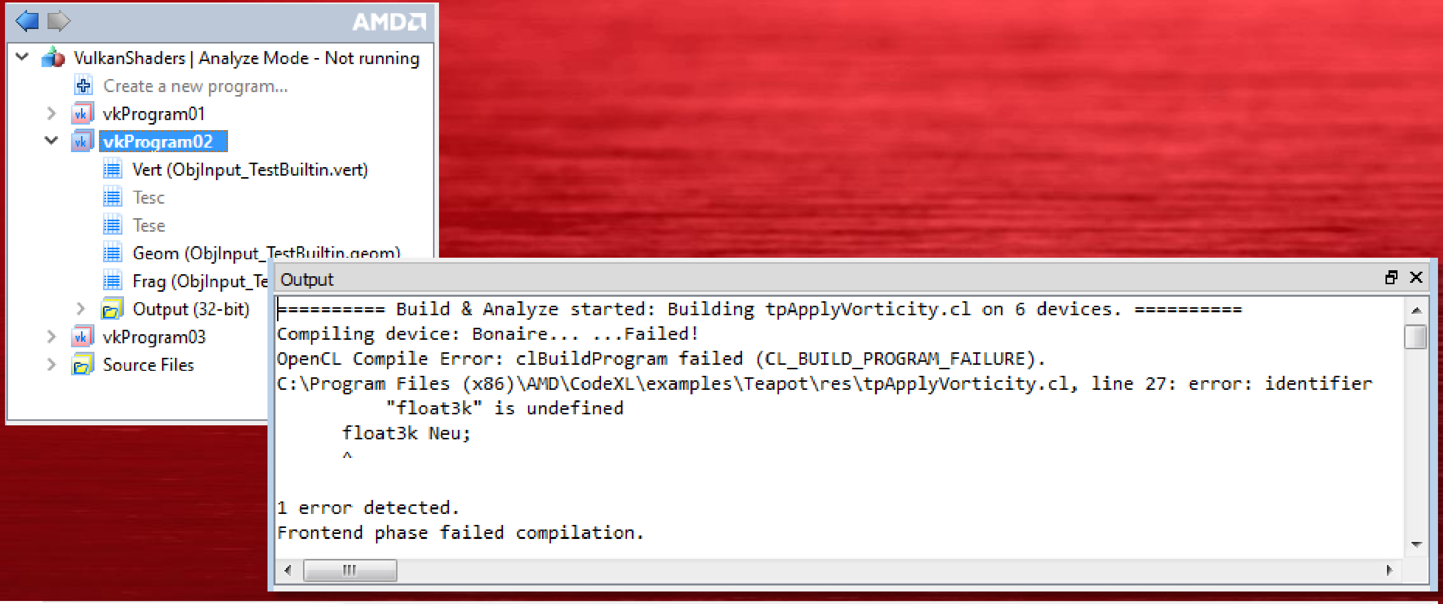
Many thanks to Dan Baker for approving the use of ‘Ashes of the Singularity’ in this demo

CODEXL FRAME ANALYSIS

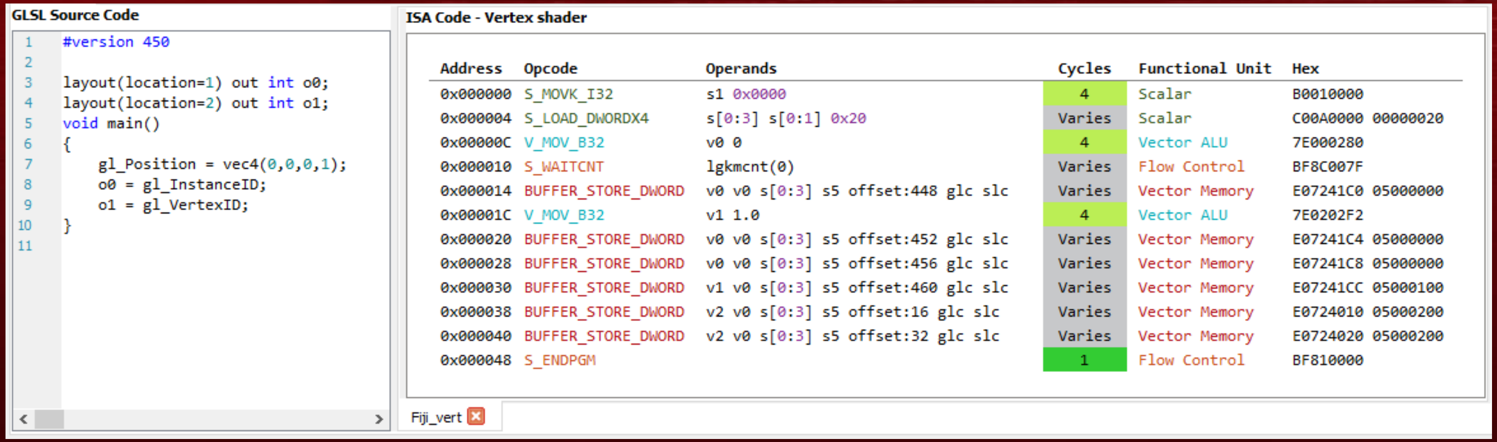
- CodeXL 2.0 is the initial release for game development features
  - Beta
- Many exciting features are awaiting in future releases:
  - Vulkan timeline trace
  - Frame Capture and Replay
  - Full set of performance counters
  - New version of GPU Performance API (GPA)
  - Frame Debugger
  - Hardware-based shader debugging

KERNEL/SHADER STATIC ANALYSIS

- Offline Build OpenCL kernels, DirectX, OpenGL* and Vulkan shaders and programs
- Multiple GPU device targets
- View Compiler Output, Resource Usage Statistics and Generated IL/ISA
- Help Detect Bottlenecks
- Performance Advisory Dashboard
- Easy Project Navigation
- Generate Binaries for Use In Your Application



```
========== Build & Analyze started: Building tpApplyVorticity.cl on 6 devices. ==========
Compiling device: Bonaire... ...Failed!
OpenCL Compile Error: clBuildProgram failed (CL_BUILD_PROGRAM_FAILURE).
C:\Program Files (x86)\AMD\CodeXL\examples\Teapot\res\tpApplyVorticity.cl, line 27: error: identifier
          "float3k" is undefined
      float3k Neu;
      ^

1 error detected.
Frontend phase failed compilation.
```



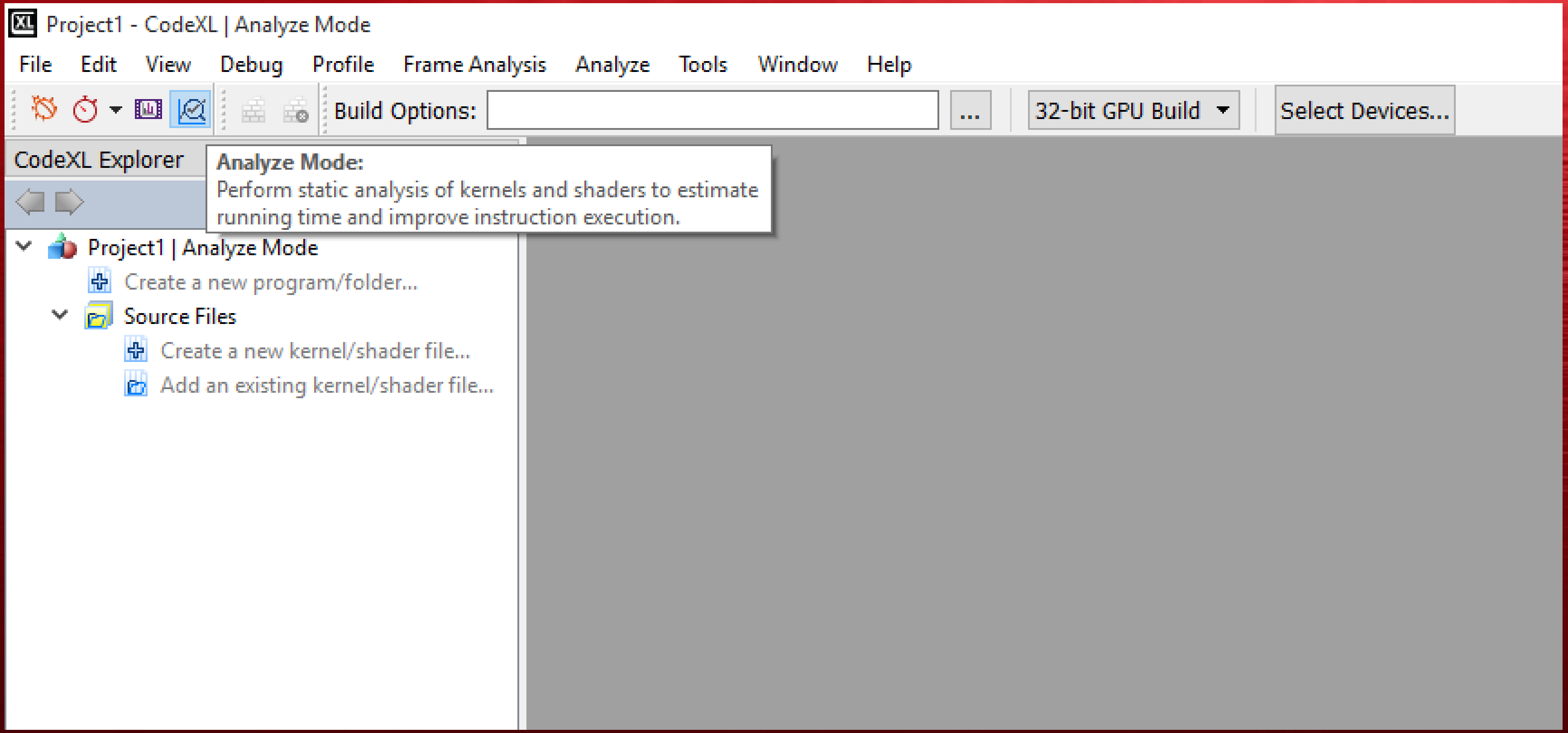
```
#version 450

layout(location=1) out int o0;
layout(location=2) out int o1;
void main()
{
    gl_Position = vec4(0,0,0,1);
    o0 = gl_InstanceID;
    o1 = gl_VertexID;
}
```

Address	Opcode	Operands	Cycles	Functional Unit	Hex
0x000000	S_MOVK_I32	s1 0x0000	4	Scalar	B0010000
0x000004	S_LOAD_DWORDX4	s[0:3] s[0:1] 0x20	Varies	Scalar	C00A0000 00000020
0x00000C	V_MOV_B32	v0 0	4	Vector ALU	7E000280
0x000010	S_WAITCNT	lgkmcnt(0)	Varies	Flow Control	BF8C007F
0x000014	BUFFER_STORE_DWORD	v0 v0 s[0:3] s5 offset:448 glc slc	Varies	Vector Memory	E07241C0 05000000
0x00001C	V_MOV_B32	v1 1.0	4	Vector ALU	7E0202F2
0x000020	BUFFER_STORE_DWORD	v0 v0 s[0:3] s5 offset:452 glc slc	Varies	Vector Memory	E07241C4 05000000
0x000028	BUFFER_STORE_DWORD	v0 v0 s[0:3] s5 offset:456 glc slc	Varies	Vector Memory	E07241C8 05000000
0x000030	BUFFER_STORE_DWORD	v1 v0 s[0:3] s5 offset:460 glc slc	Varies	Vector Memory	E07241CC 05000100
0x000038	BUFFER_STORE_DWORD	v2 v0 s[0:3] s5 offset:16 glc slc	Varies	Vector Memory	E0724010 05000200
0x000040	BUFFER_STORE_DWORD	v2 v0 s[0:3] s5 offset:32 glc slc	Varies	Vector Memory	E0724020 05000200
0x000048	S_ENDPGM		1	Flow Control	BF810000

CODEXL SHADER ANALYZER
DEMO

DEMO – SHADER ANALYSIS (1)

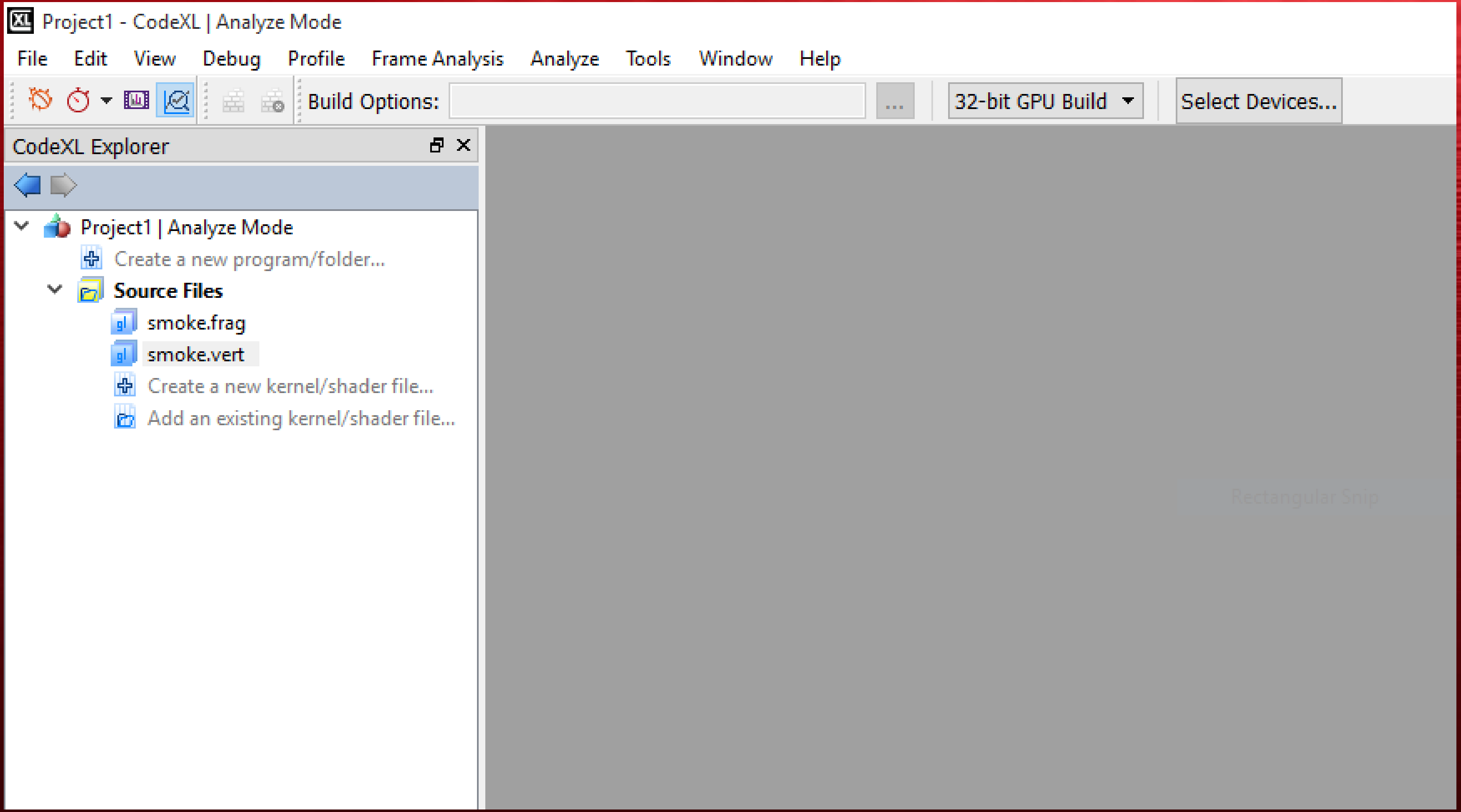
- I create a new project and switch to Analyze mode by clicking the Analyze Mode button.



Analyze Mode:
Perform static analysis of kernels and shaders to estimate running time and improve instruction execution.

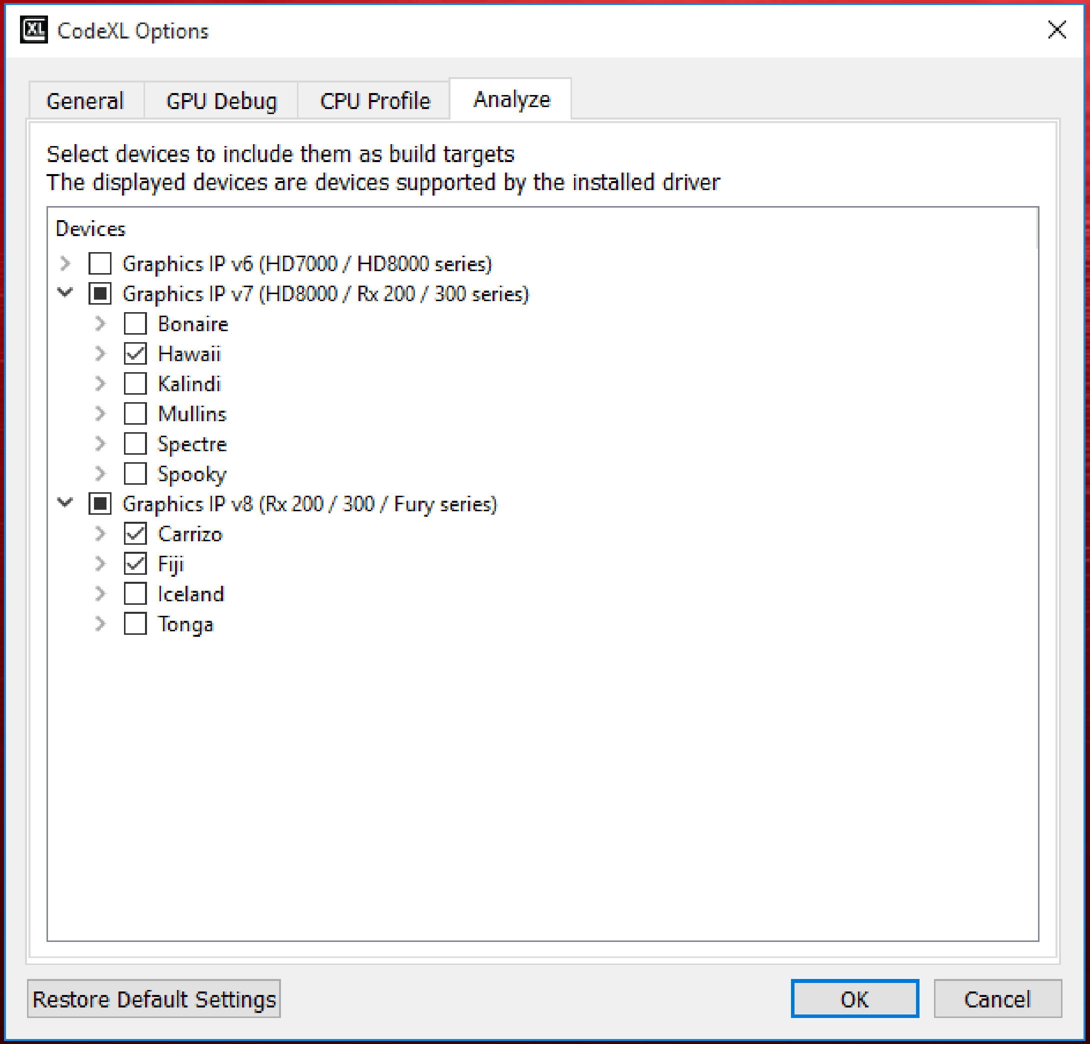
DEMO – SHADER ANALYSIS (2)

- I drag Vulkan shader files from the windows explorer to the CodeXL tree to add them to the project.



DEMO – SHADER ANALYSIS (3)

- I click the Select Devices button to select the target devices the compiler will generate ISA for.
- The target devices can be any AMD GPU or APU.
- I can choose any target device regardless of the actual AMD hardware installed on my workstation.



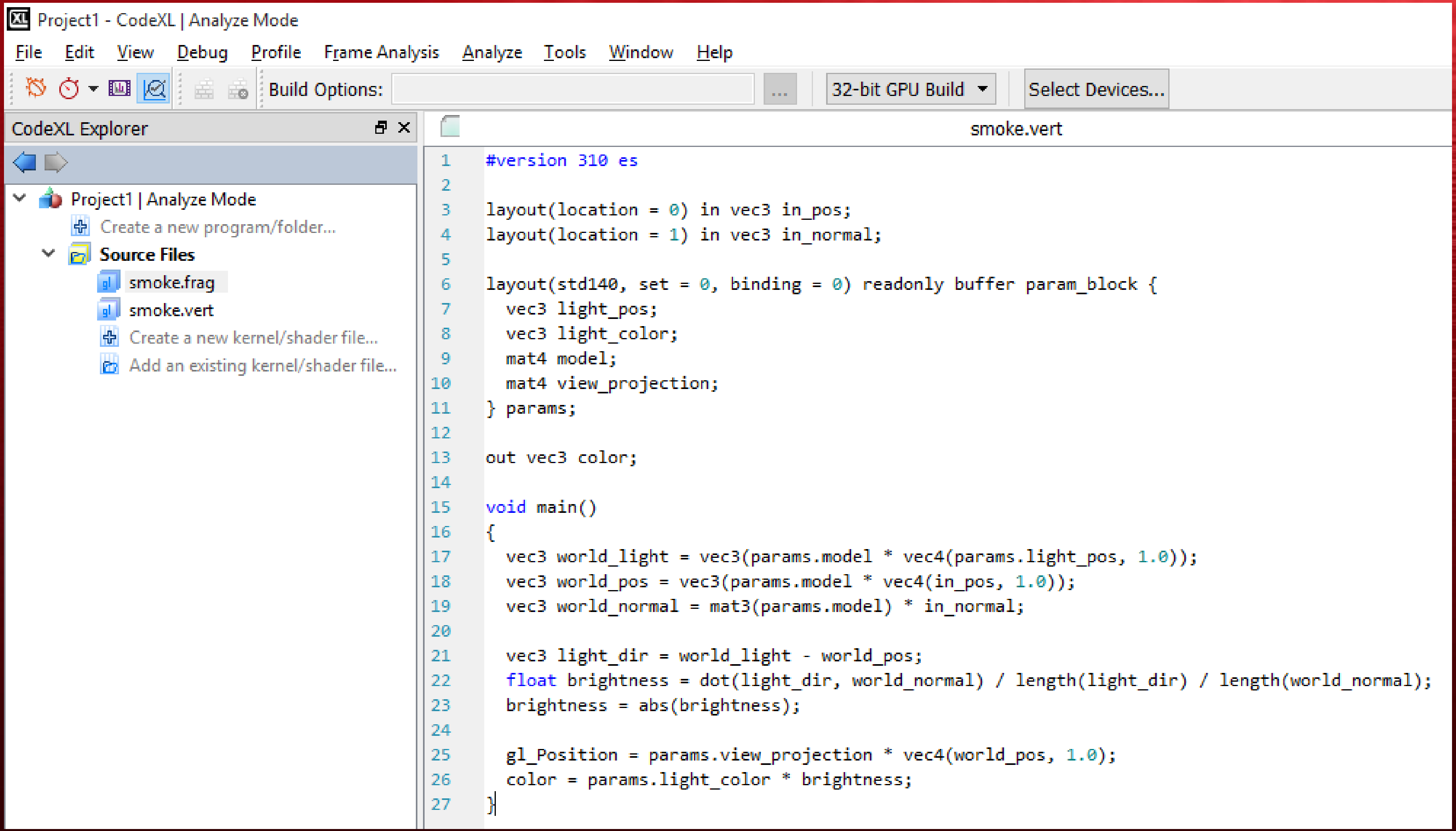
Select devices to include them as build targets
The displayed devices are devices supported by the installed driver

Devices

- □ Graphics IP v6 (HD7000 / HD8000 series)
- ■ Graphics IP v7 (HD8000 / Rx 200 / 300 series)
  - □ Bonaire
  - ☑ Hawaii
  - □ Kalindi
  - □ Mullins
  - □ Spectre
  - □ Spooky
- ■ Graphics IP v8 (Rx 200 / 300 / Fury series)
  - ☑ Carrizo
  - ☑ Fiji
  - □ Iceland
  - □ Tonga

DEMO – SHADER ANALYSIS (4)

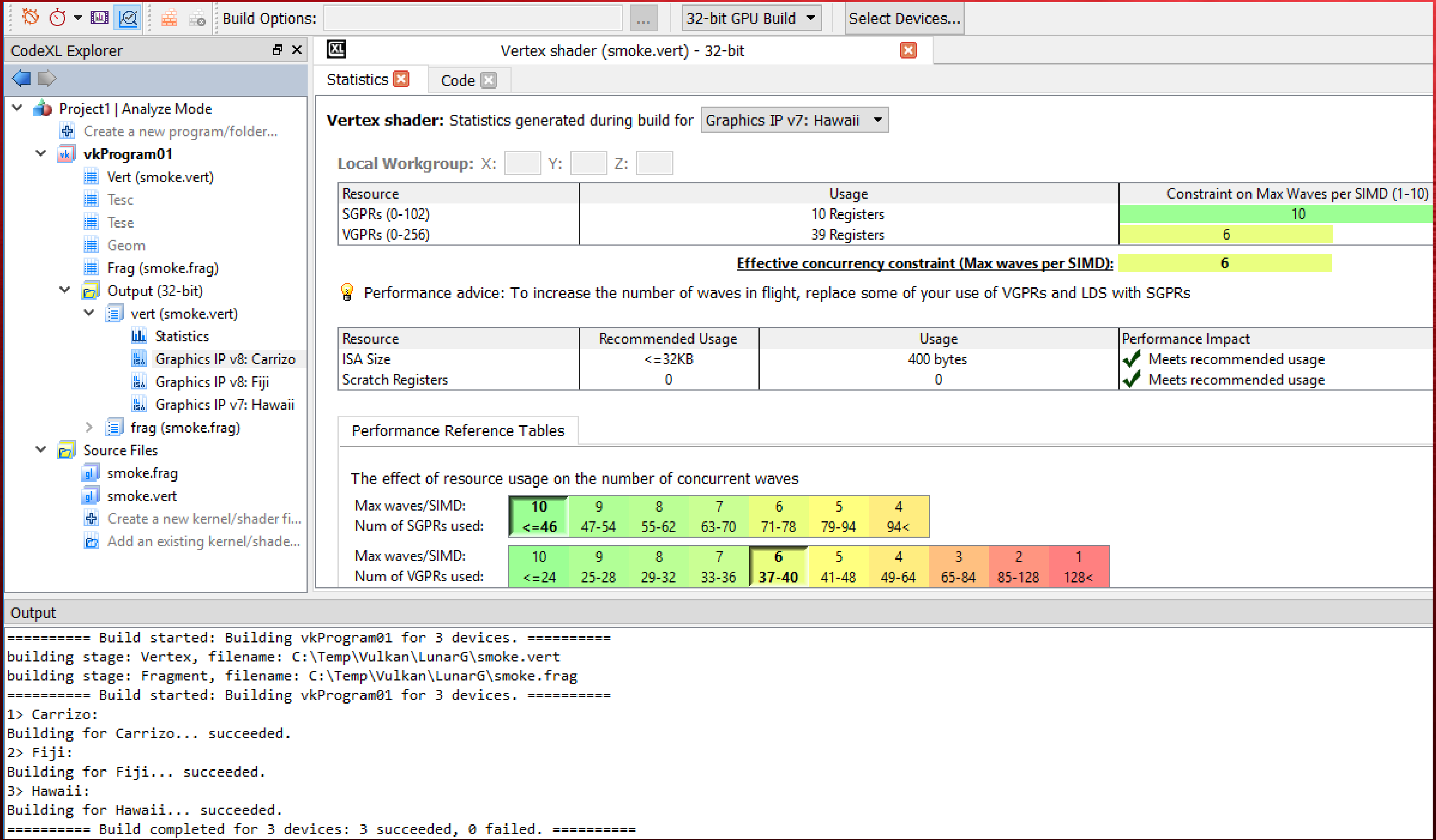
- I can edit the shader source code.



```glsl
#version 310 es

layout(location = 0) in vec3 in_pos;
layout(location = 1) in vec3 in_normal;

layout(std140, set = 0, binding = 0) readonly buffer param_block {
  vec3 light_pos;
  vec3 light_color;
  mat4 model;
  mat4 view_projection;
} params;

out vec3 color;

void main()
{
  vec3 world_light = vec3(params.model * vec4(params.light_pos, 1.0));
  vec3 world_pos = vec3(params.model * vec4(in_pos, 1.0));
  vec3 world_normal = mat3(params.model) * in_normal;

  vec3 light_dir = world_light - world_pos;
  float brightness = dot(light_dir, world_normal) / length(light_dir) / length(world_normal);
  brightness = abs(brightness);

  gl_Position = params.view_projection * vec4(world_pos, 1.0);
  color = params.light_color * brightness;
}
```

DEMO – SHADER ANALYSIS (5)

- I create a Vulkan rendering program and drag each shader to the appropriate pipeline stages.
- I click F7 to build the program.
- CodeXL displays the compiler output and a resource usage report.
- The 32-bit Output sub-tree is populated In the CodeXL explorer tree.



Vertex shader: Statistics generated during build for Graphics IP v7: Hawaii

Local Workgroup: X: Y: Z:

Resource	Usage	Constraint on Max Waves per SIMD (1-10)
SGPRs (0-102)	10 Registers	10
VGPRs (0-256)	39 Registers	6

Effective concurrency constraint (Max waves per SIMD): 6

Performance advice: To increase the number of waves in flight, replace some of your use of VGPRs and LDS with SGPRs

Resource	Recommended Usage	Usage	Performance Impact
ISA Size	<=32KB	400 bytes	✓ Meets recommended usage
Scratch Registers	0	0	✓ Meets recommended usage

Performance Reference Tables

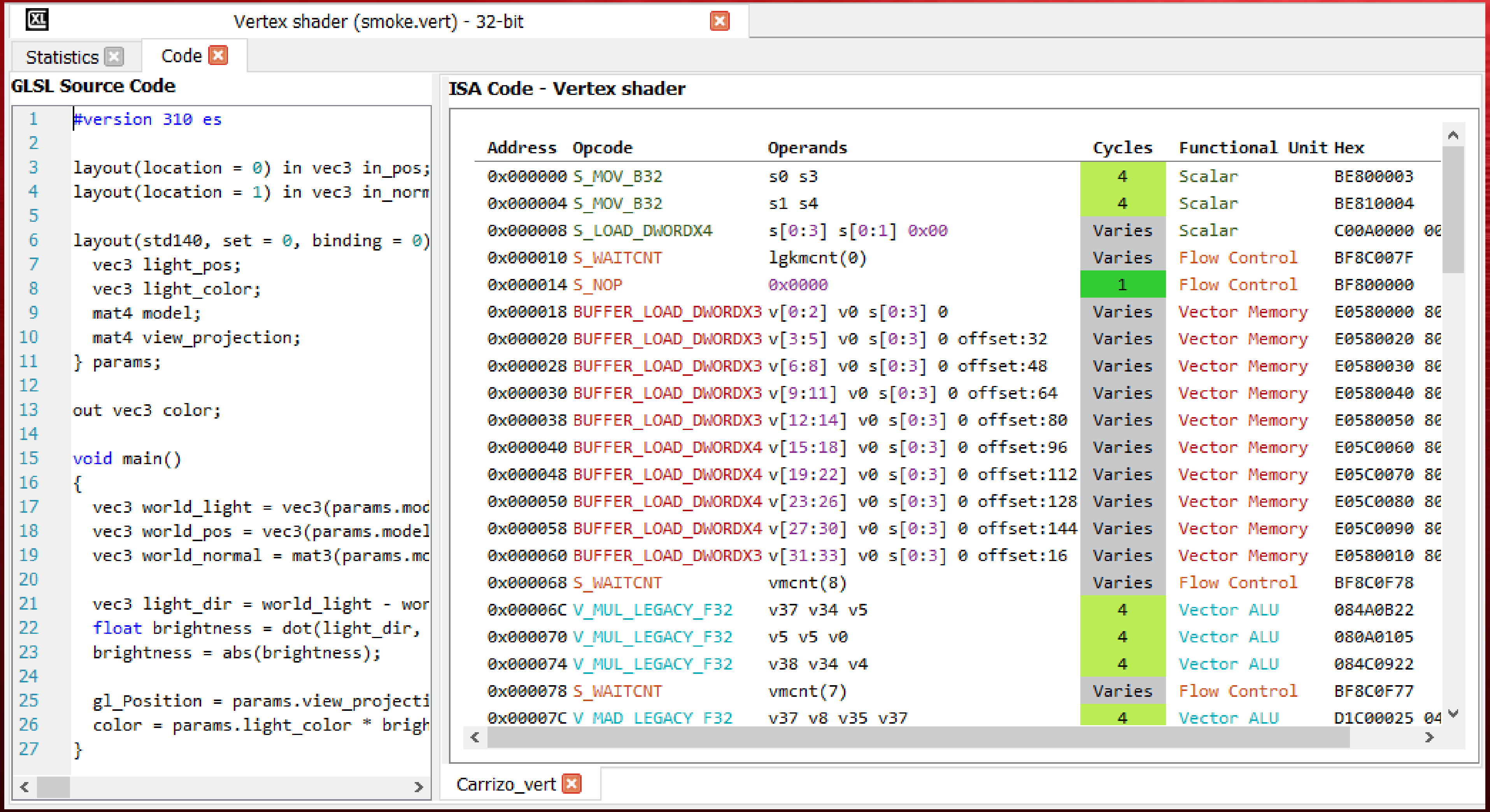
The effect of resource usage on the number of concurrent waves

Max waves/SIMD:	**10**	9	8	7	6	5	4			
Num of SGPRs used:	**<=46**	47-54	55-62	63-70	71-78	79-94	94<			
Max waves/SIMD:	10	9	8	7	**6**	5	4	3	2	1
Num of VGPRs used:	<=24	25-28	29-32	33-36	**37-40**	41-48	49-64	65-84	85-128	128<

```
========== Build started: Building vkProgram01 for 3 devices. ==========
building stage: Vertex, filename: C:\Temp\Vulkan\LunarG\smoke.vert
building stage: Fragment, filename: C:\Temp\Vulkan\LunarG\smoke.frag
========== Build started: Building vkProgram01 for 3 devices. ==========
1> Carrizo:
Building for Carrizo... succeeded.
2> Fiji:
Building for Fiji... succeeded.
3> Hawaii:
Building for Hawaii... succeeded.
========== Build completed for 3 devices: 3 succeeded, 0 failed. ==========
```

DEMO – SHADER ANALYSIS (6)

- Clicking a target device node displays the ISA generated for that device next to the high level GLSL source code.
- ISA is color coded by the type of instruction and theoretical number of cycles required to execute it.



GLSL Source Code

```
#version 310 es

layout(location = 0) in vec3 in_pos;
layout(location = 1) in vec3 in_norm

layout(std140, set = 0, binding = 0)
  vec3 light_pos;
  vec3 light_color;
  mat4 model;
  mat4 view_projection;
} params;

out vec3 color;

void main()
{
  vec3 world_light = vec3(params.moc
  vec3 world_pos = vec3(params.model
  vec3 world_normal = mat3(params.mc

  vec3 light_dir = world_light - wor
  float brightness = dot(light_dir,
  brightness = abs(brightness);

  gl_Position = params.view_projecti
  color = params.light_color * brigh
}
```

ISA Code - Vertex shader

Address	Opcode	Operands	Cycles	Functional Unit	Hex
0x000000	S_MOV_B32	s0 s3	4	Scalar	BE800003
0x000004	S_MOV_B32	s1 s4	4	Scalar	BE810004
0x000008	S_LOAD_DWORDX4	s[0:3] s[0:1] 0x00	Varies	Scalar	C00A0000 00
0x000010	S_WAITCNT	lgkmcnt(0)	Varies	Flow Control	BF8C007F
0x000014	S_NOP	0x0000	1	Flow Control	BF800000
0x000018	BUFFER_LOAD_DWORDX3	v[0:2] v0 s[0:3] 0	Varies	Vector Memory	E0580000 80
0x000020	BUFFER_LOAD_DWORDX3	v[3:5] v0 s[0:3] 0 offset:32	Varies	Vector Memory	E0580020 80
0x000028	BUFFER_LOAD_DWORDX3	v[6:8] v0 s[0:3] 0 offset:48	Varies	Vector Memory	E0580030 80
0x000030	BUFFER_LOAD_DWORDX3	v[9:11] v0 s[0:3] 0 offset:64	Varies	Vector Memory	E0580040 80
0x000038	BUFFER_LOAD_DWORDX3	v[12:14] v0 s[0:3] 0 offset:80	Varies	Vector Memory	E0580050 80
0x000040	BUFFER_LOAD_DWORDX4	v[15:18] v0 s[0:3] 0 offset:96	Varies	Vector Memory	E05C0060 80
0x000048	BUFFER_LOAD_DWORDX4	v[19:22] v0 s[0:3] 0 offset:112	Varies	Vector Memory	E05C0070 80
0x000050	BUFFER_LOAD_DWORDX4	v[23:26] v0 s[0:3] 0 offset:128	Varies	Vector Memory	E05C0080 80
0x000058	BUFFER_LOAD_DWORDX4	v[27:30] v0 s[0:3] 0 offset:144	Varies	Vector Memory	E05C0090 80
0x000060	BUFFER_LOAD_DWORDX3	v[31:33] v0 s[0:3] 0 offset:16	Varies	Vector Memory	E0580010 80
0x000068	S_WAITCNT	vmcnt(8)	Varies	Flow Control	BF8C0F78
0x00006C	V_MUL_LEGACY_F32	v37 v34 v5	4	Vector ALU	084A0B22
0x000070	V_MUL_LEGACY_F32	v5 v5 v0	4	Vector ALU	080A0105
0x000074	V_MUL_LEGACY_F32	v38 v34 v4	4	Vector ALU	084C0922
0x000078	S_WAITCNT	vmcnt(7)	Varies	Flow Control	BF8C0F77
0x00007C	V_MAD_LEGACY_F32	v37 v8 v35 v37	4	Vector ALU	D1C00025 04

END OF SHADER ANALYZER DEMO

THERE'S SO MUCH MORE TO CODEXL!

MORE CODEXL 2.0 NEW FEATURES

- Static Analysis of OpenGL shader programs
- Integration with Visual Studio 2015
- Combined host debugging + GPU debugging of OpenCL 1.2 kernels
- Process-specific power profiling

AMD

Enabling today.
Inspiring tomorrow.

CODEXL ON GPUOPEN

CODEXL IS MOVING TO OPEN-SOURCE!

- All of CodeXL is open-source and hosted on GitHub as part of GPUOpen.com (anticipated Apr-2016)
- MIT license
- Everyone is welcome to join the development
- AMD Developer Tools group will continue developing new features and expanding CodeXL
- CodeXL is continuously tested and verified on:
  - Windows® 7 64-bit, 8.1 64-bit and 10 64-bit
  - Ubuntu 64-bit
  - Red-Hat 64-bit
- CodeXL User Forum: http://devgurus.amd.com/community/codexl

GPUOpen

Q&A

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION

LunarG Vulkan sample program ‘Hologram’ source code used in Shader Analyzer demo is copyright LunarG. See complete copyright notice here:
https://github.com/LunarG/VulkanSamples/blob/master/LICENSE.txt