Make Your Game Friendly for Graphics Debugging and Optimization

Adam Sawicki
Developer Technology Engineer
AMD

AGENDA

- Introduction
- Basics
- Advanced
- Conclusion

Introduction

Iteration time is critically important

Good engine is not about awesome renderer.
It's about tools – like convenient editor.

- Shorter iteration time
- Good tools

Everyone benefits

Easier to search for bugs and performance optimizations – benefit for:

- developers
- QA
- external partners

Part 1

Basics

Options (1)

Provide configuration options:

- resolution
- display mode: Windowed / Borderless / Exclusive fullscreen
- V-sync On/Off

Options (2)

- texture resolution: important for GPUs with little memory
- texture filtering quality (linear vs anisotropic): impacts performance
- MSAA Disabled/2x/4x/…

- (extra) resolution scaling

Ways to provide options

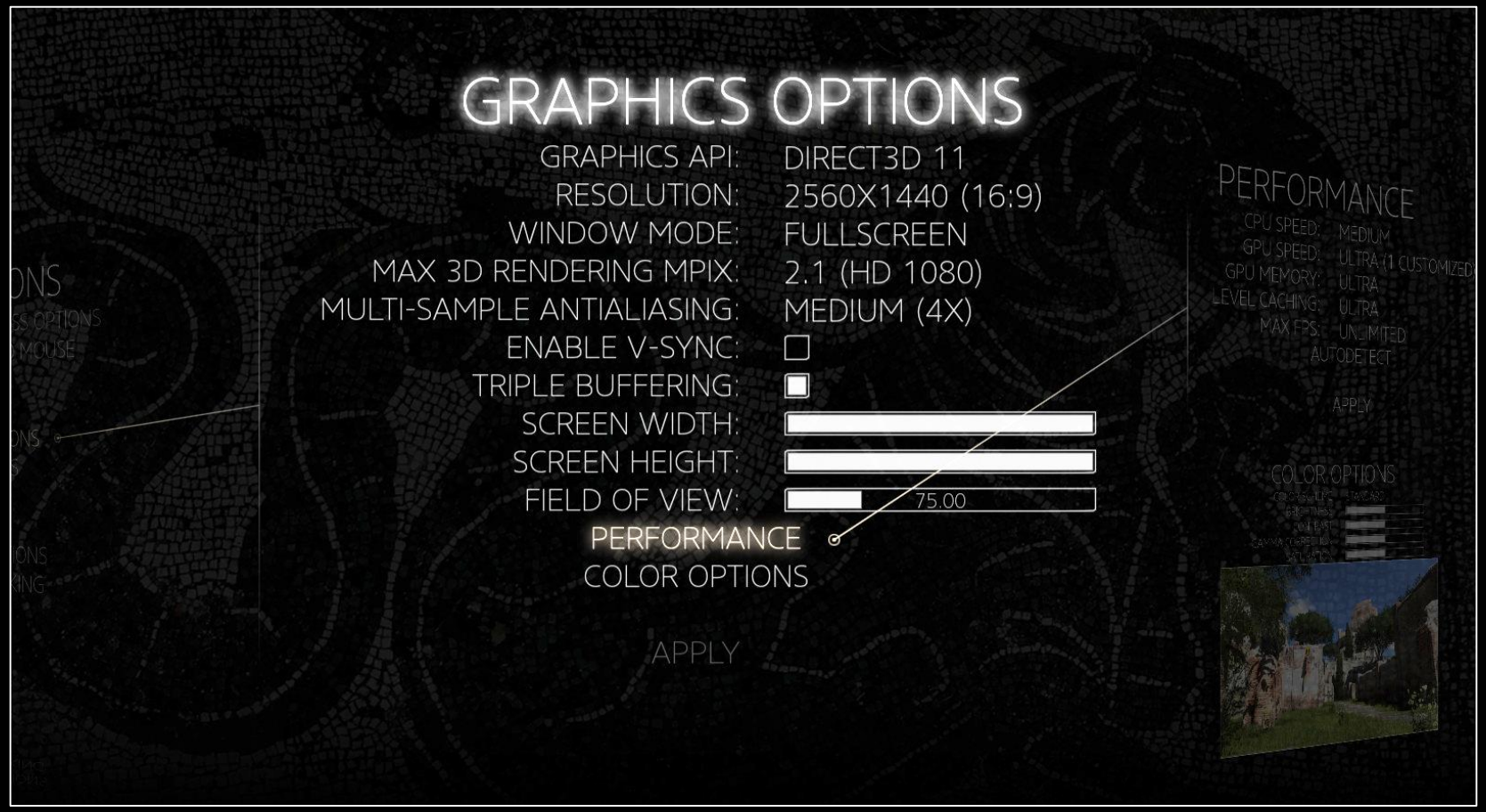
Some may be visible to end users.

- in-game menu
- launcher window (e.g. Unity) or separate app

Some may be hidden.

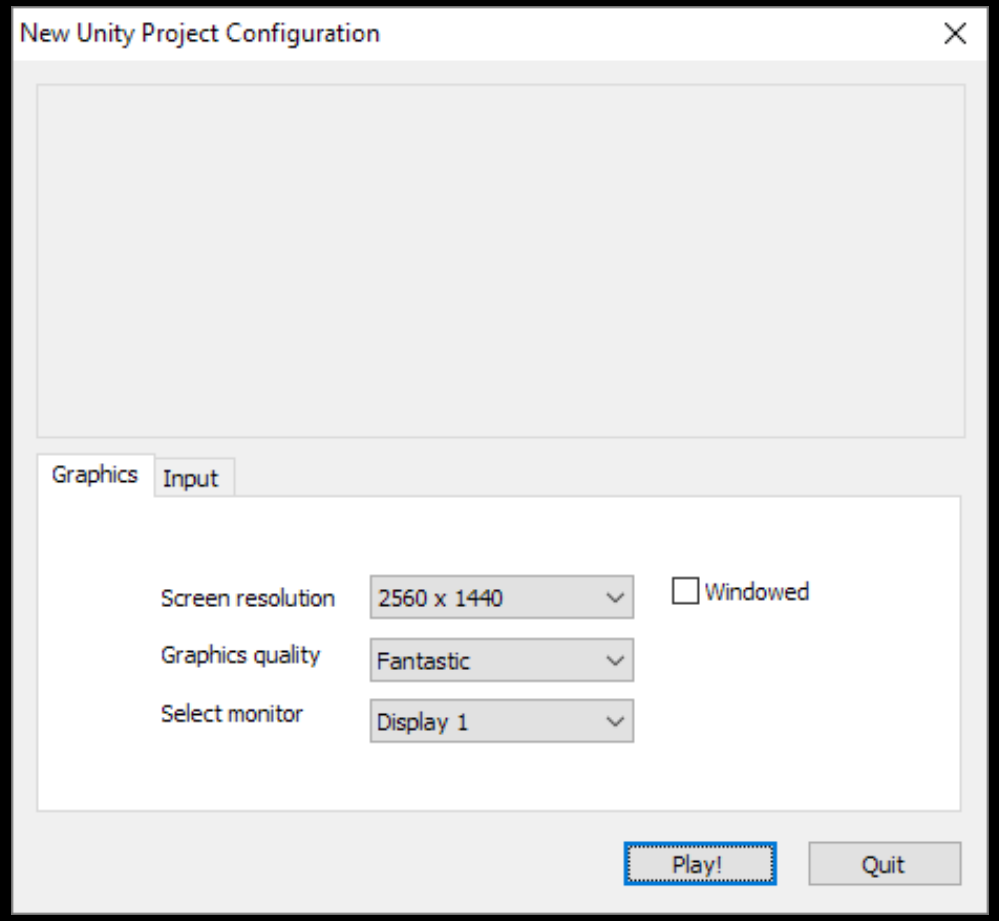
- in-game developer/cheat panel
- in-game console (e.g. Unreal Engine)
- command line options
- configuration file, Windows registry

Options



The Talos Principle – in-game options

Options

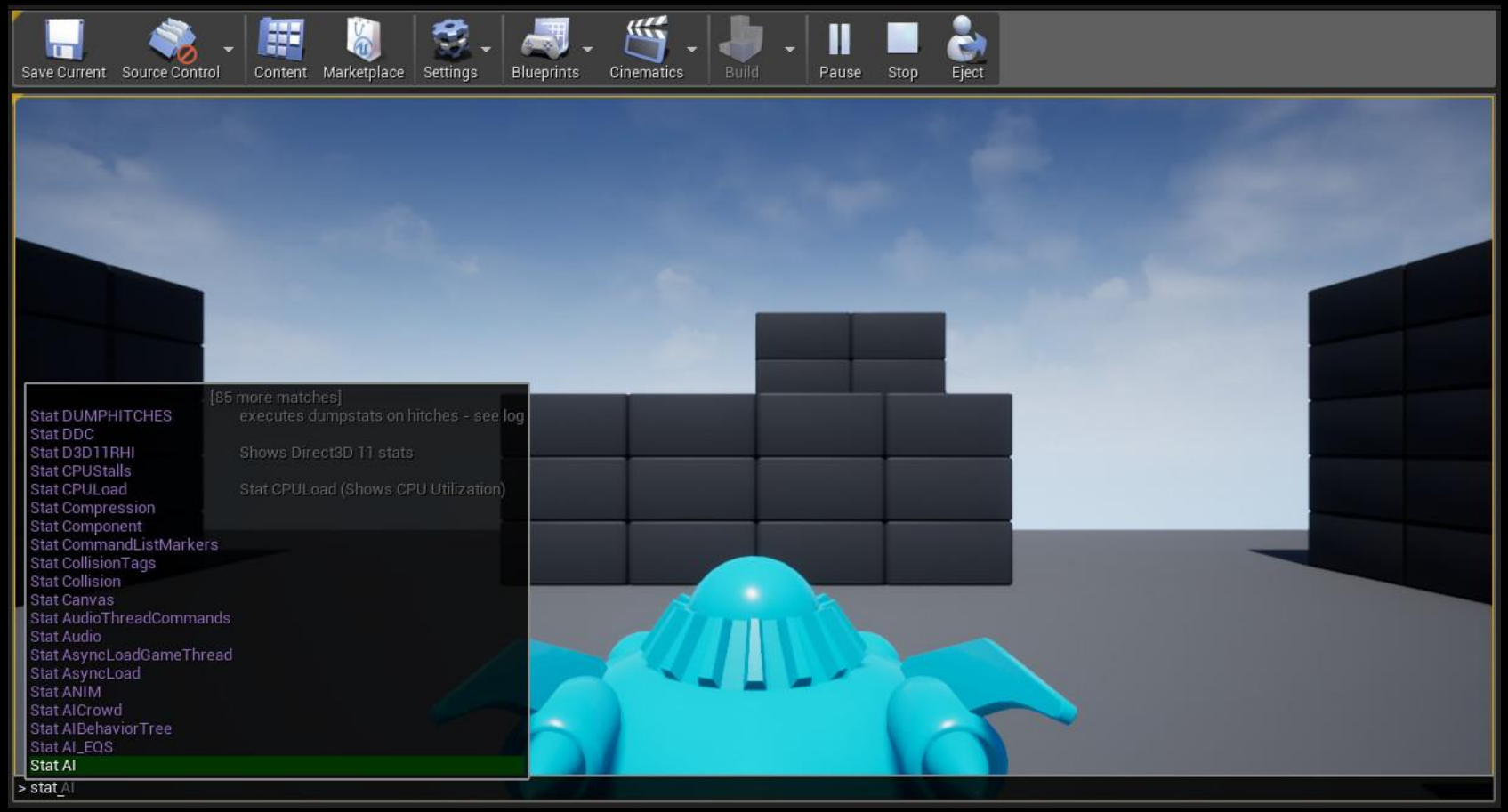


Unity launcher window

Options

War Thunder – custom launcher app

Options



Unreal Engine – in-game console

Options (3)

Effects On/Off/quality level:

- user-facing
  - number of objects (e.g. grass distance)
  - shader complexity (e.g. water quality)

Options (4)

Effects On/Off/quality level:

- internal
  - CPU workloads
  - Graphics
  - GPU compute

Optimizations On/Off

Correctness/stability

“Safe mode”

- single threaded
- D3D12, Vulkan®: additional synchronization
- useful for debugging visual corruptions and GPU crashes (TDR)

Developer Game Build

- works as standalone EXE (without Steam, no UWP)
- works offline, no connection to server required
  - Server often down or incompatible during development.
  - Users may be behind restrictive firewall/VPN.
- no DRM/anti-cheat/anti-piracy protection (like Denuvo)

- Provide documentation of available settings, cheats, god mode.
- Provide saves for various locations.

Loading Time

Short loading time is critical.

- simple test scene with just few objects
- loading production scene should still be fast
- performance of debug build also important

Benchmark Mode

- automated testing (continuous integration)
  - correctness
  - performance
- manual testing
  - your developers
  - QA

[van Valburg]

Benchmark Mode

- launched with special command line parameter
- non-interactive – scripted camera flythrough
- representative scene
  - graphics
  - gameplay logic?
- deterministic
  - every run calculates and shows the same

Extra:

- ends automatically
- measures performance, writes results to text file

```
minfps=27.7
avgfps=30.8
score=1882
```

Part 2

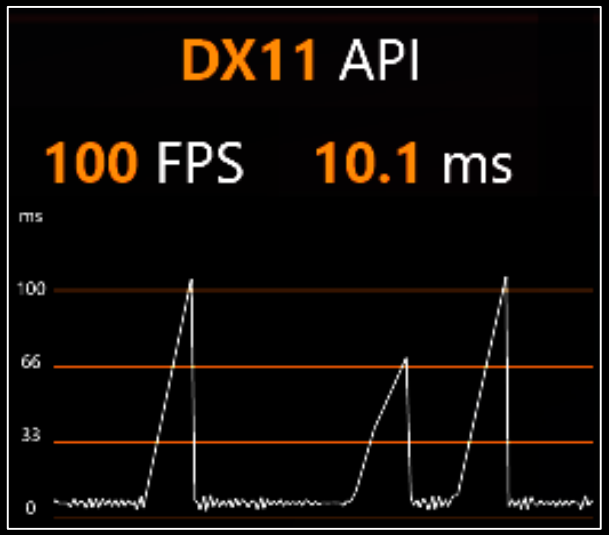
Advanced

Beware! Code ahead.

In-Game Profiler

- FPS and average frame time
- detailed counters – time of render passes

If not available, you can measure frame times with OCAT. [OCAT]

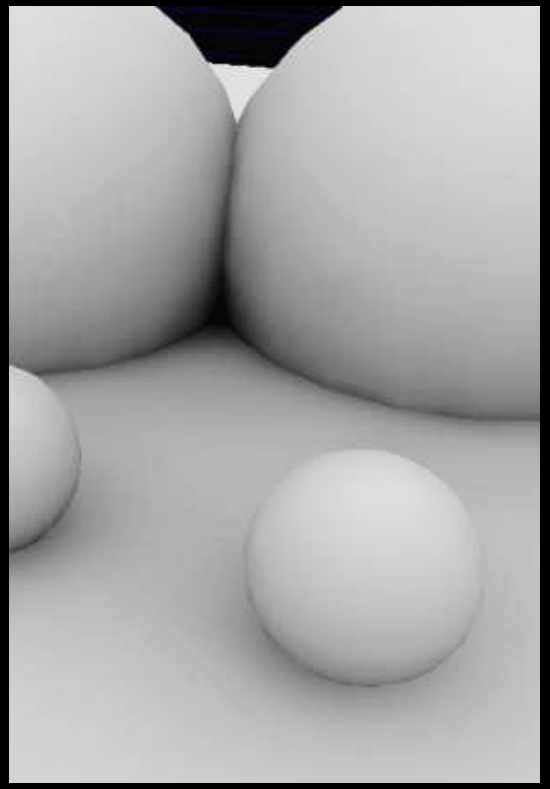


FPS and beyond

- Statistical measures – avg, min, max, percentile (1%, 99%)
- Catch spikes
- Draw frame histogram

In-Game Tools

Debug modes for visualization of intermediate data e.g. ambient occlusion only.



Source: docs.unrealengine.com

- Pixel inspector window
- Way to distinguish NaN, INF from 0, 1

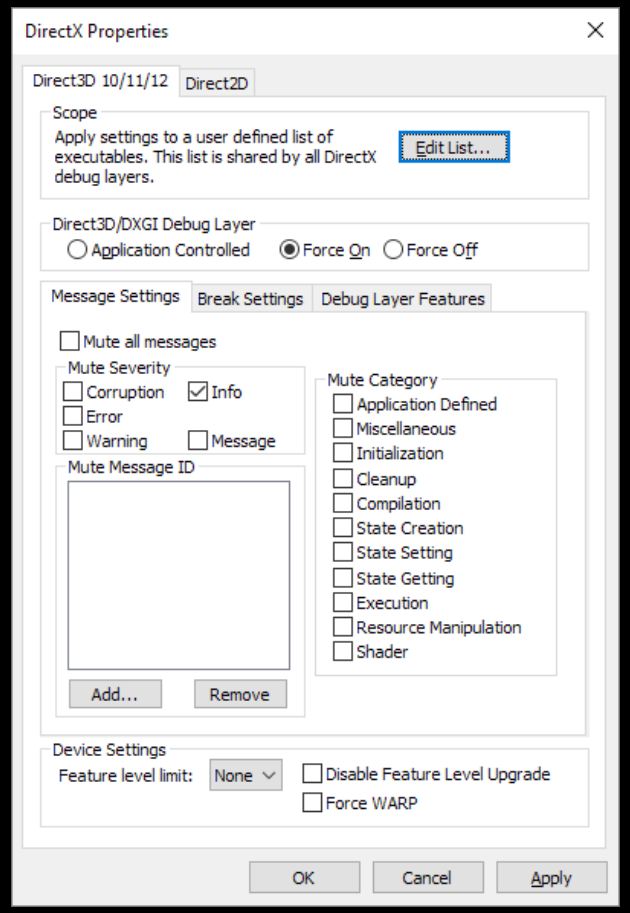
Debug/Validation Layers

- Old APIs (Direct3D 9, OpenGL®): any function can return error.

- New APIs (Direct3D 11, 12, Vulkan®): no error checking.
- Debug/validation layer can be explicitly enabled.

- Use them in regular testing.
- Require to pass on all APIs.

Debug Layer – Direct3D 11, 12

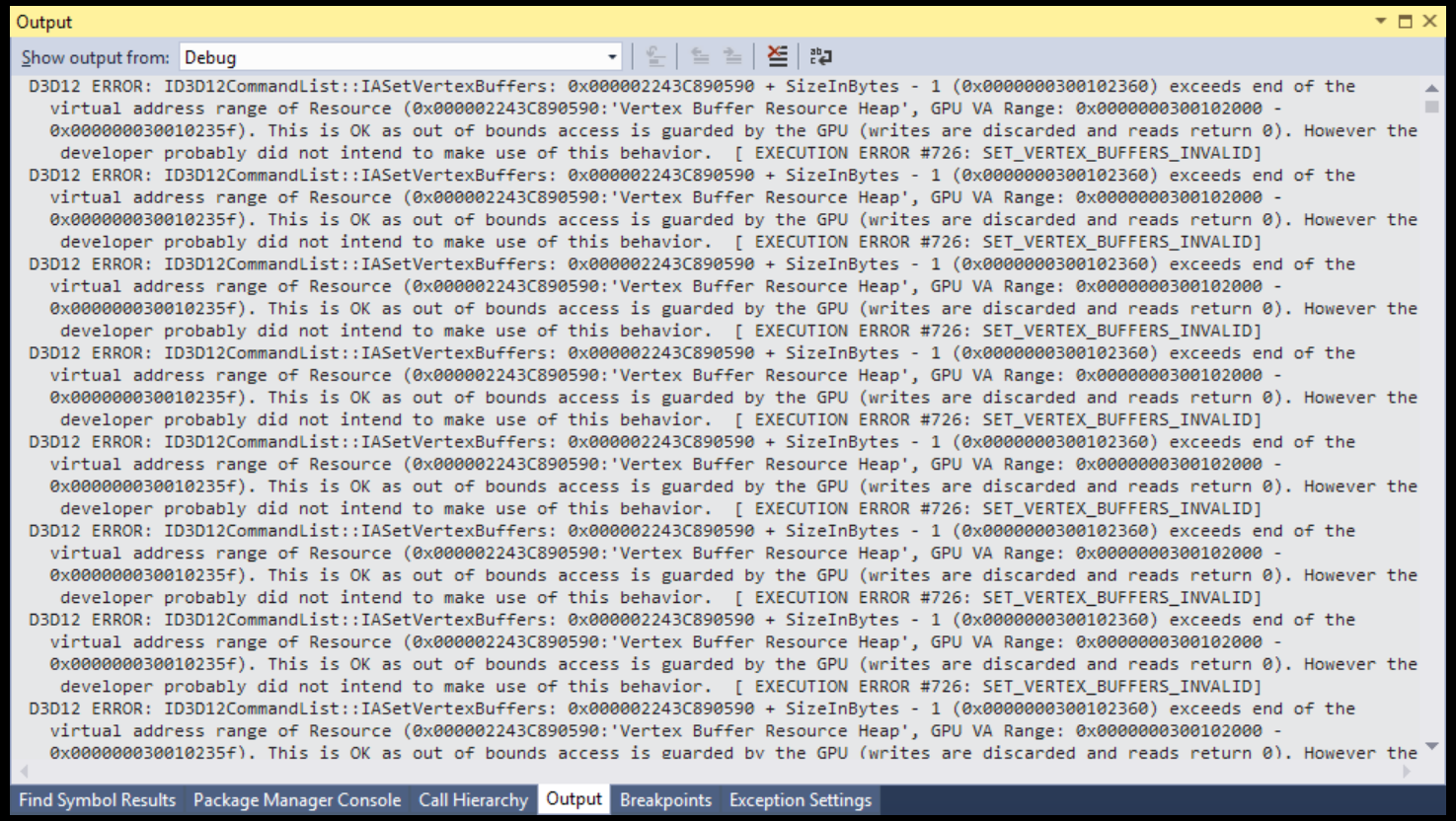


dxcpl

```
D3D11_CREATE_DEVICE_DEBUG
```

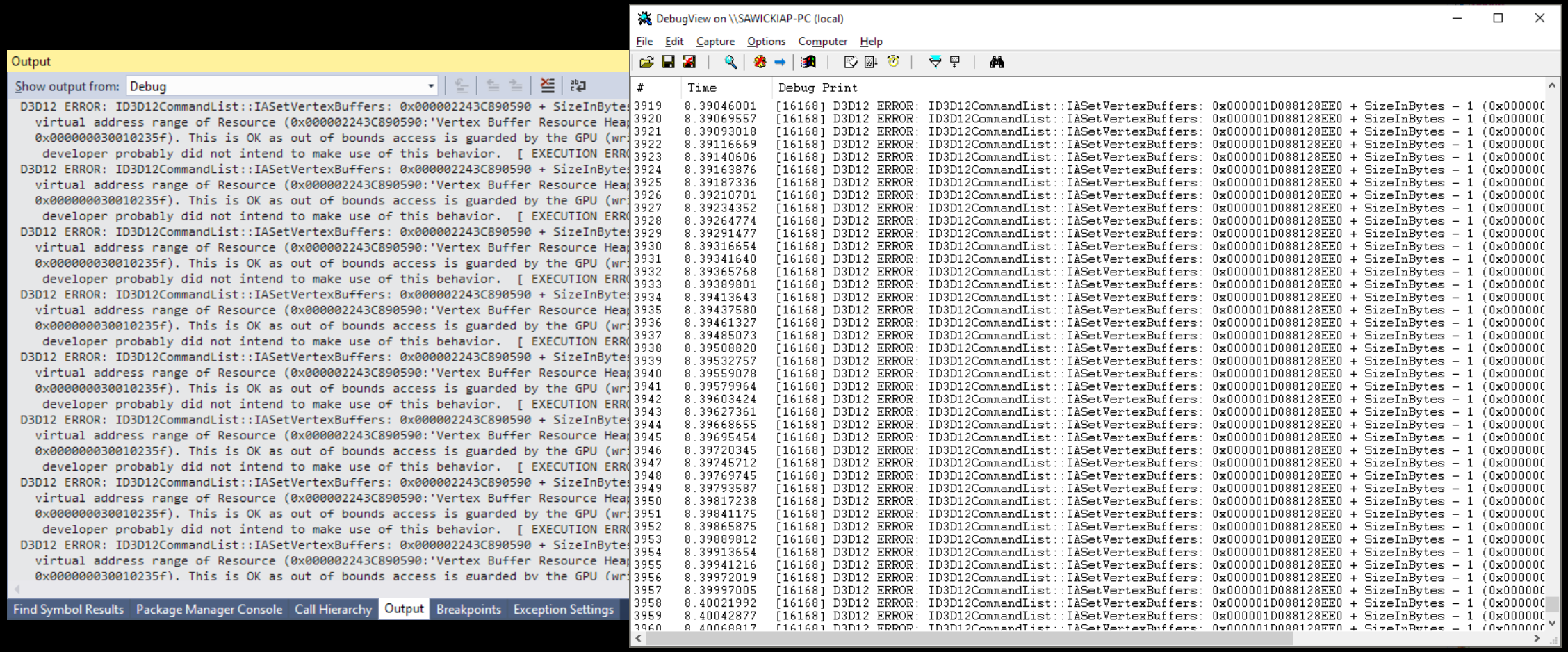
```
CComPtr<ID3D12Debug> debug;
if(SUCCEEDED(D3D12GetDebugInterface(
    IID_PPV_ARGS(&debug))))
    debug->EnableDebugLayer();
```

Debug Layer – Direct3D



Visual Studio – Output

Debug Layer – Direct3D



DebugView from SysInternals

Validation Layers – Vulkan®

VK_LAYER_LUNARG_standard_validation

- enable programmatically or
- `set VK_INSTANCE_LAYERS=VK_LAYER_LUNARG_standard_validation`

- messages delivered as callback

Driver Bug?

OGL, D3D11:

- driver handling complex logic
- crashes not expected – if crashed, driver bug very likely

Vulkan®, D3D12:

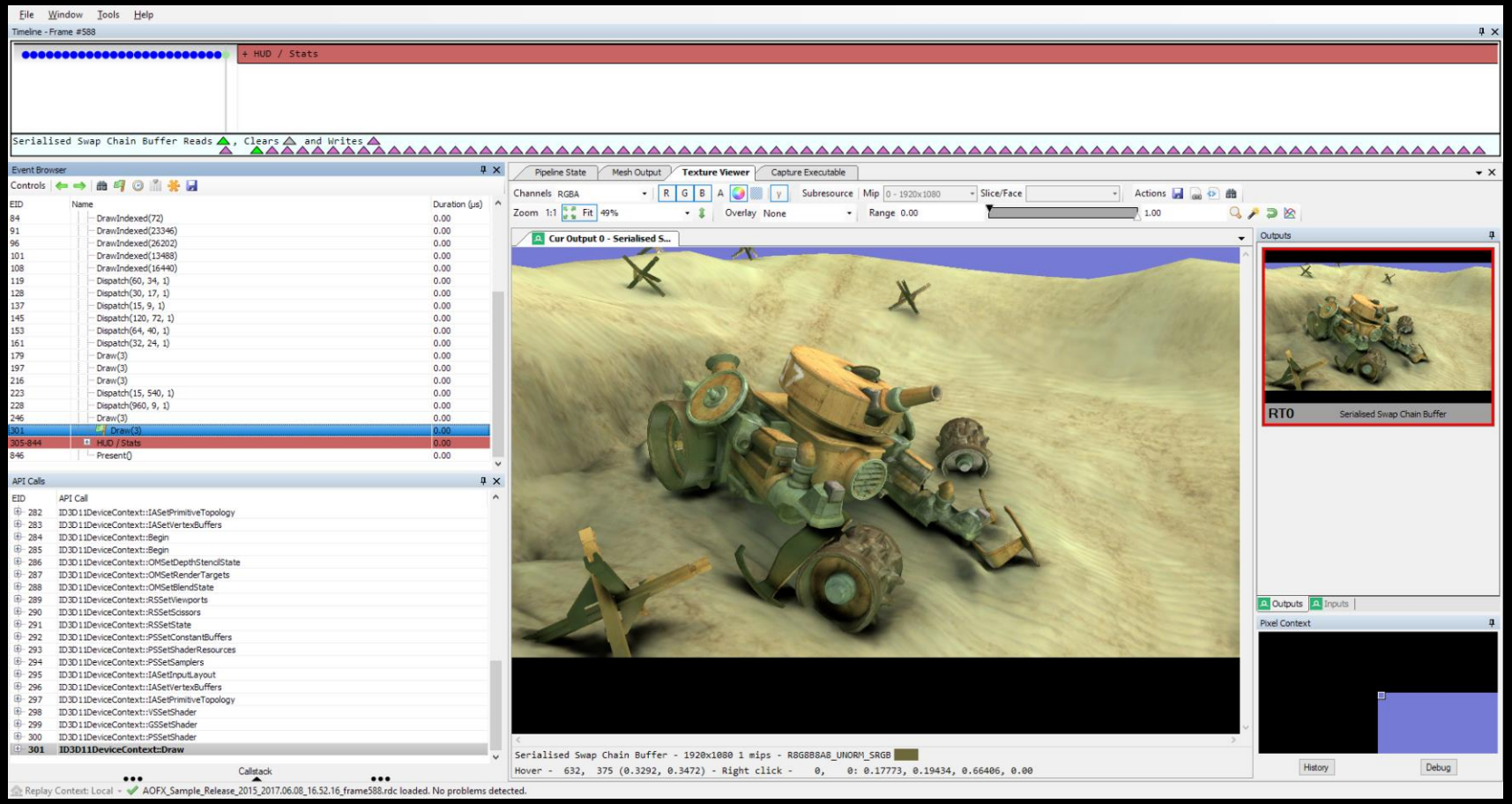
- game/engine responsible for most logic
- API is low level, driver is thin
- if crashing, most likely your bug

GPU-Assisted Validation

- Available in Vulkan (“GPU-assisted validation”) and Direct3D 12 (“GPU-based validation”).
- Enabled programmatically or externally.
- Injects additional code to shaders.

- Finds bugs in dynamic resource indexing - useful for “bindless”.

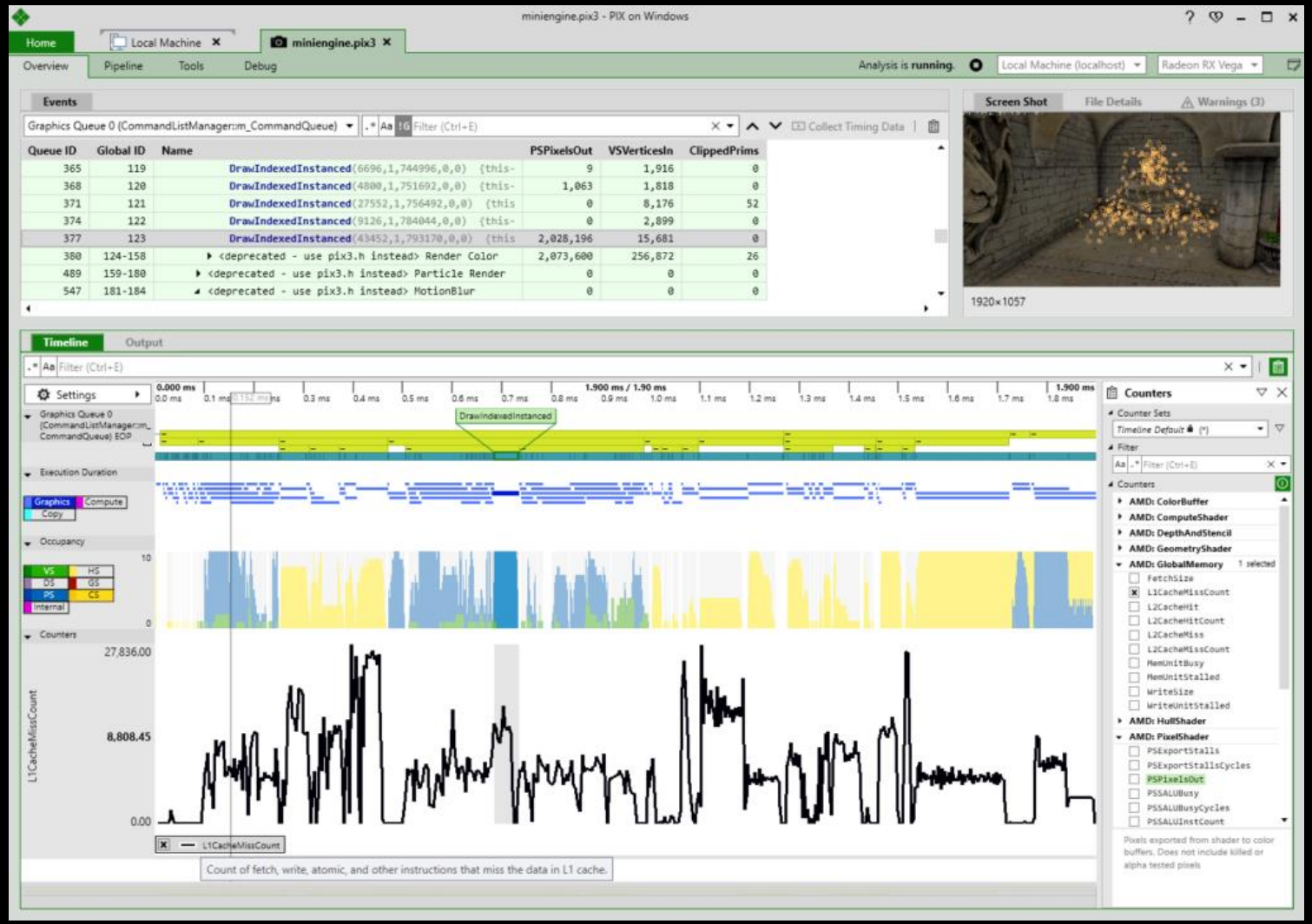
Tools – RenderDoc



[RenderDoc]

Alternatives: Nvidia® Nsight™, Intel® Graphics Performance Analyzers (GPA)

Tools – PIX

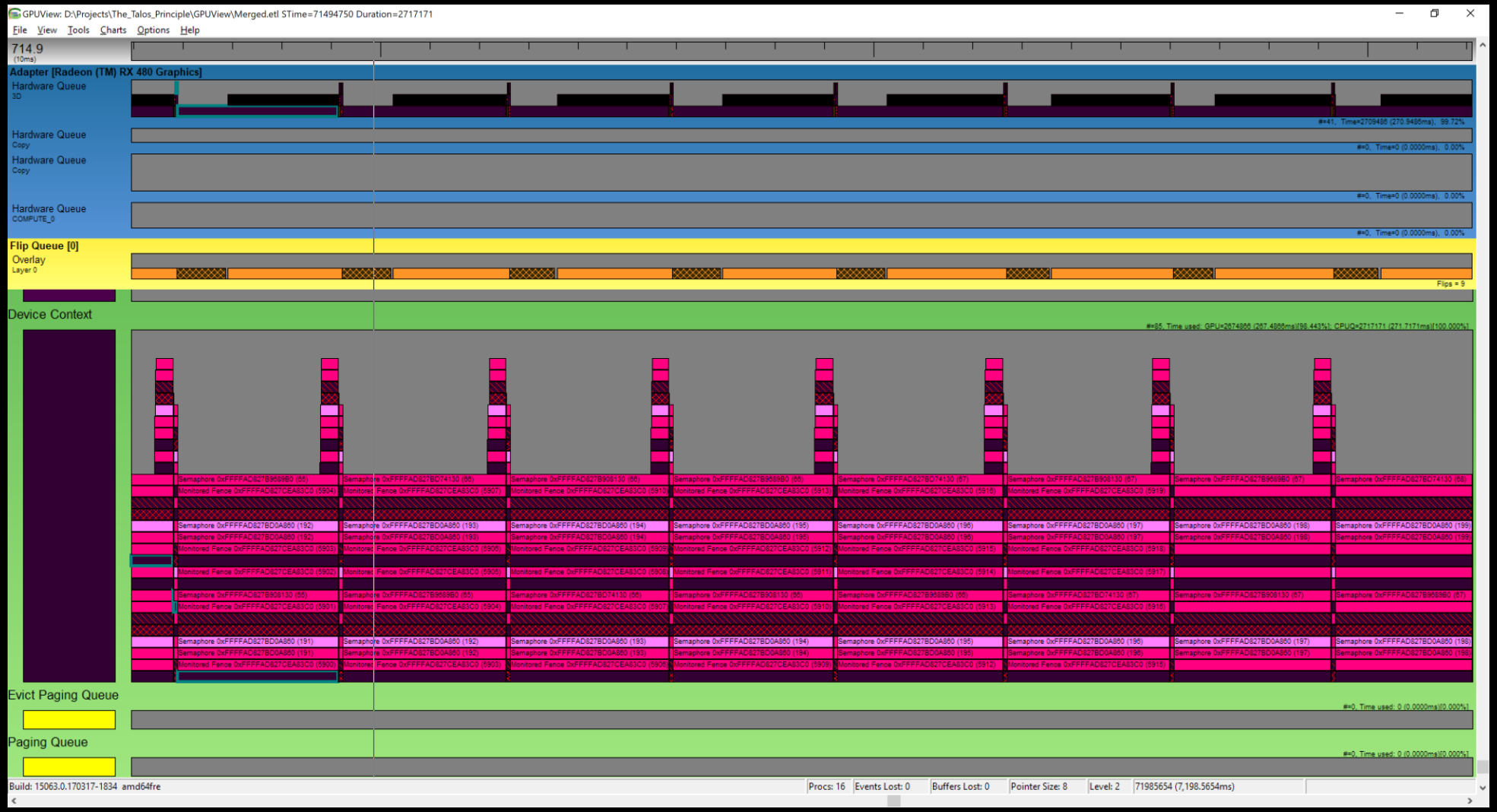


[PIX]

Tools – Radeon GPU Profiler

[RGP]

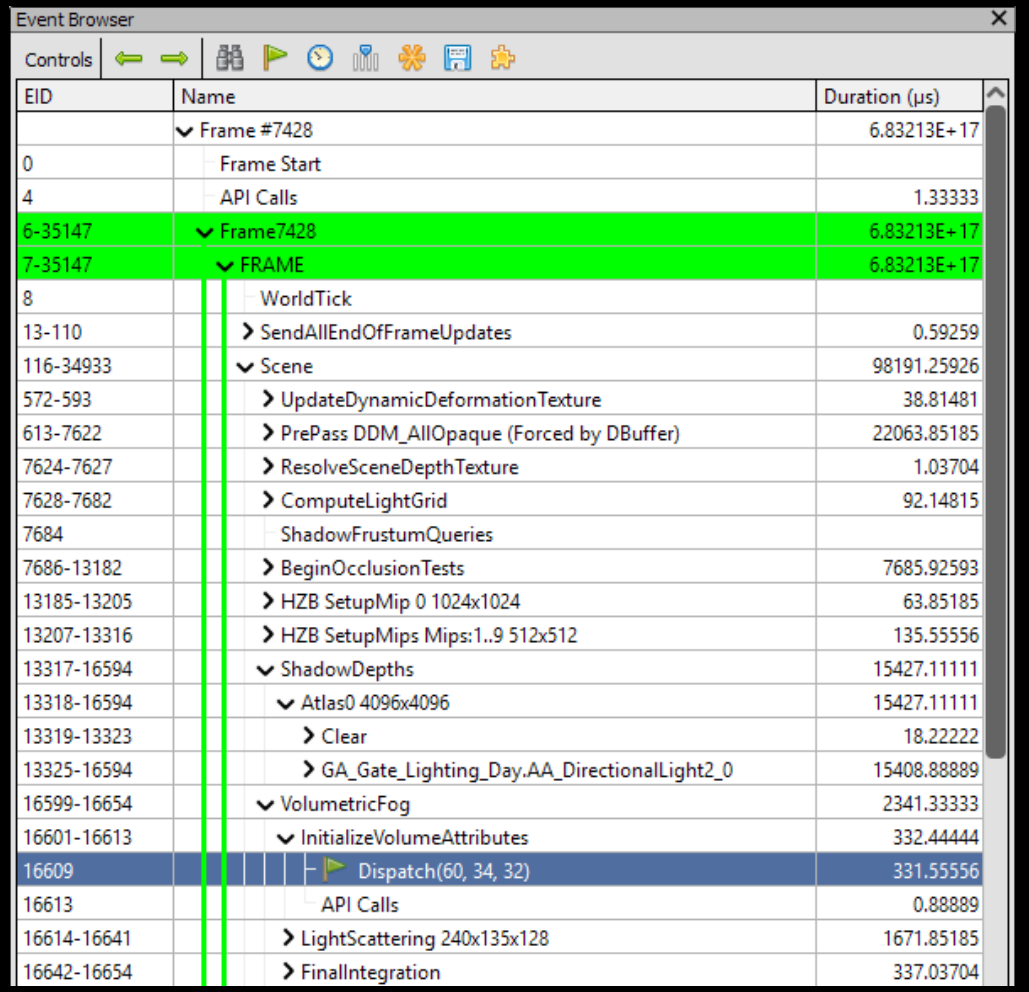
Tools – GPUView



[GPUView]

Debug Markers

- Aka “labels”, “annotations”
- Wrap passes with Begin…End markers with custom names.
- Give names to resources.
- Supported by many tools.



Debug Markers

- D3D9+: D3DPERF_BeginEvent(), D3DPERF_EndEvent()
- D3D11.1+: ID3DUserDefinedAnnotation::BeginEvent(), EndEvent()
- Vulkan®: VK_EXT_debug_utils
  - previously: VK_EXT_debug_marker

Debugging GPU crash/hang

Crash/hang of driver/GPU doesn’t crash whole system.

- Handled by Timeout Detection & Recovery (TDR).

`VK_ERROR_DEVICE_LOST`

`DXGI_ERROR_DEVICE_REMOVED`

Difficult to debug.

Debugging GPU crash/hang

- Solution: markers written on GPU between draw calls.
- After crash: inspect last written value, deduce culprit draw call.

- Vulkan®: vkCmdFillBuffer()
- Vulkan® + AMD: VK_AMD_buffer_marker
- D3D12: ID3D12GraphicsCommandList2:: WriteBufferImmediate()
- D3D12 + AMD: breadcrumb markers in AGS [AGS]
- D3D11/D3D12 + NVIDIA: NVIDIA Aftermath [Aftermath]

Debugging GPU crash/hang

Heavyweight solution: record each draw call state to a blob.

- Support serialized states and draw calls.
- Recreate state, reproduce the crash.
- Even better: record command buffers.

Conclusion

Good Practices

- first stability, then correctness, then performance

- test early, test often
- test on various GPUs
  - AMD, NVIDIA, Intel
  - low-end, high-end

- track regressions

Conclusion

- Good tools and short iterations time is important.
- Ensure them with:
  - development practices
  - your code
  - external tools
- Everyone benefits ☺

References

- [van Valburg] Automated testing for Call of Duty, Jan van Valburg (Activision), Digital Dragons 2018
  - https://www.youtube.com/watch?v=6OVFbLnIFR4
- [OCAT] The Open Capture and Analytics Tool
  - https://github.com/GPUOpen-Tools/OCAT
- [AGS] AMD GPU Services (AGS)
  - https://gpuopen.com/gaming-product/amd-gpu-services-ags-library/
- [Aftermath] NVIDIA Aftermath
  - https://developer.nvidia.com/nvidia-aftermath
- [RenderDoc] RenderDoc
  - http://renderdoc.org/
- [PIX] PIX on Windows
  - https://devblogs.microsoft.com/pix/
- [RGP] Radeon GPU Profiler
  - https://gpuopen.com/gaming-product/radeon-gpu-profiler-rgp/
- [GPUView] GPUView
  - https://graphics.stanford.edu/~mdfisher/GPUView.html

QUESTIONS?

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


AMD