Using GFXReconstruct for Capture & Replay with Vulkan and D3D12

Brad Grantham
LunarG, Inc.

GFXReconstruct - Agenda

- What is it
- Where to get it
- How to use it
- Pointers for advanced users
- Questions and answers

GFXReconstruct - Overview

GFXReconstruct lets a developer examine and replay a program's captured graphics commands after a program has been run.

- **Capture** an application's graphics live commands
- **Replay** those commands at any time without the application
- Allows inspection and some transformation of those commands

GFXReconstruct - Overview

This is useful for:

- Architecture simulation
- Silicon bringup
- Driver regression testing
- Bug reporting
- Developer investigation

Currently in use by several GPU, chipset, platform vendors

GFXReconstruct - Overview

Release packages are available on GitHub:

https://github.com/LunarG/gfxreconstruct/releases

Binaries (without Direct3D) also in the Vulkan SDK:

https://vulkan.lunarg.com/sdk/home

Source code:

https://github.com/LunarG/gfxreconstruct

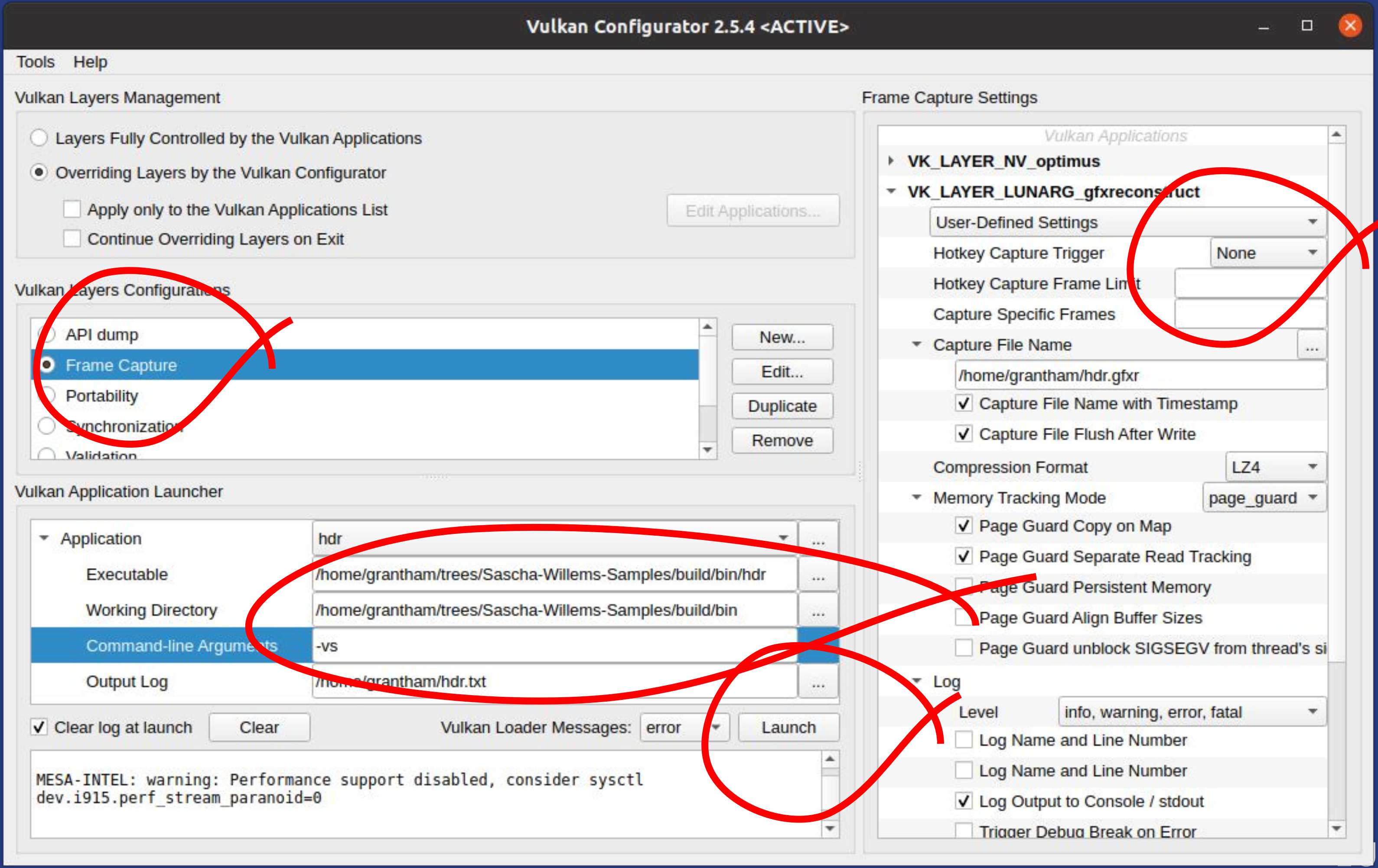
GFXReconstruct - Overview

- Capture graphics commands in a file (aka a “capture file” or “trace file”)
  - Function call inputs and return code and outputs if successful
  - Binary file for fast readback and replay
  - Commands are stored in compressed blocks
- Replay captures by issuing the same commands
- Handful of additional tools to operate on capture files
- C++ libraries, layers, and apps; some Python wrappers
- Linux, Android, Windows
- API-agnostic; Vulkan and Direct3D 12 so far!

GFXReconstruct - Capturing A Vulkan Application

Vulkan API layer “`libVkLayer_gfxreconstruct.so`” or `.dll`

- Use Vulkan Configurator (VkConfig)
  - or “`gfxrecon-capture-vulkan.py`”
  - or `VK_INSTANCE_LAYERS=VK_LAYER_LUNARG_gfxreconstruct`
- Records all core 1.3 Vulkan function calls and many extensions



Vulkan Configurator 2.5.4 <ACTIVE>

Tools Help

Vulkan Layers Management

- ( ) Layers Fully Controlled by the Vulkan Applications
- (●) Overriding Layers by the Vulkan Configurator
  - ☐ Apply only to the Vulkan Applications List
  - ☐ Continue Overriding Layers on Exit

Edit Applications...

Vulkan Layers Configurations

- ( ) API dump
- (●) Frame Capture
- ( ) Portability
- ( ) Synchronization
- ( ) Validation

New... Edit... Duplicate Remove

Vulkan Application Launcher

Application	hdr
Executable	/home/grantham/trees/Sascha-Willems-Samples/build/bin/hdr
Working Directory	/home/grantham/trees/Sascha-Willems-Samples/build/bin
Command-line Arguments	-vs
Output Log	/home/grantham/hdr.txt

✓ Clear log at launch Clear Vulkan Loader Messages: error Launch

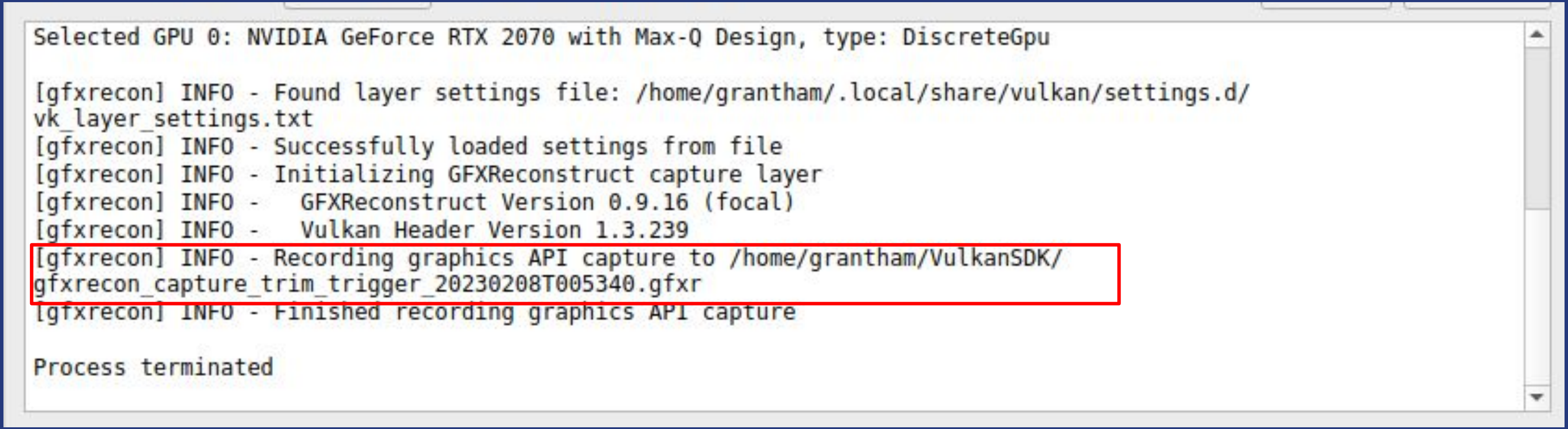
```
MESA-INTEL: warning: Performance support disabled, consider sysctl
dev.i915.perf_stream_paranoid=0
```

Frame Capture Settings

Vulkan Applications

- VK_LAYER_NV_optimus
- VK_LAYER_LUNARG_gfxreconstruct

User-Defined Settings	
Hotkey Capture Trigger	None
Hotkey Capture Frame Limit	
Capture Specific Frames	
Capture File Name	/home/grantham/hdr.gfxr
✓ Capture File Name with Timestamp	
✓ Capture File Flush After Write	
Compression Format	LZ4
Memory Tracking Mode	page_guard
✓ Page Guard Copy on Map	
✓ Page Guard Separate Read Tracking	
☐ Page Guard Persistent Memory	
☐ Page Guard Align Buffer Sizes	
☐ Page Guard unblock SIGSEGV from thread's si	
Log	
Level	info, warning, error, fatal
☐ Log Name and Line Number	
☐ Log Name and Line Number	
✓ Log Output to Console / stdout	
☐ Trigger Debug Break on Error	



```
Selected GPU 0: NVIDIA GeForce RTX 2070 with Max-Q Design, type: DiscreteGpu

[gfxrecon] INFO - Found layer settings file: /home/grantham/.local/share/vulkan/settings.d/
vk_layer_settings.txt
[gfxrecon] INFO - Successfully loaded settings from file
[gfxrecon] INFO - Initializing GFXReconstruct capture layer
[gfxrecon] INFO -   GFXReconstruct Version 0.9.16 (focal)
[gfxrecon] INFO -   Vulkan Header Version 1.3.239
[gfxrecon] INFO - Recording graphics API capture to /home/grantham/VulkanSDK/
gfxrecon_capture_trim_trigger_20230208T005340.gfxr
[gfxrecon] INFO - Finished recording graphics API capture

Process terminated
```

LUNARG

GFXReconstruct - Capturing Ranges of Frames

- Can also capture ranges of frames
  - By number e.g. `GFXRECON_CAPTURE_FRAMES=1,2,10-20`
  - Or (on desktop) using a hotkey (e.g. F3)
- All graphics **state** up to the range is “tracked”
  - Stored in the capture file as state setup
- Conservative
  - Writes all tracked objects to the file at beginning of range
  - Can’t know what future frames will reference

(Works for Vulkan and Direct3D 12)

GFXReconstruct - `gfxrecon.py info`

Display useful information about a capture

- Compression
- Frames
- App info
- Device info

```
$ gfxrecon.py info ~/gfxrecon_capture_20220412T075011.gfxr
File info:
        Compression format: LZ4
        Total frames: 50

Application info:
        Application name: vkcube
        Application version: 0
        Engine name: vkcube
        Engine version: 0
        Target API version: 4198400 (1.1.0)

Physical device info:
        Device name: AMD Radeon RX 6700 XT
        Device ID: 0x73df
        Vendor ID: 0x1002
        Driver version: 8388821 (0x8000d5)
        API version: 4206795 (1.3.203)

Device memory allocation info:
[...]
```

GFXReconstruct - `gfxrecon.py convert`

```
{
  "index": 1,
  "vkFunc": {
    "name": "vkCreateInstance",
    "return": "VK_SUCCESS",
    "args": {
      "pCreateInfo": {
        "sType": "VK_STRUCTURE_TYPE_INSTANCE_CREATE_INFO",
        "pNext": null,
        "flags": 1,
        "pApplicationInfo": {
          "sType": "VK_STRUCTURE_TYPE_APPLICATION_INFO",
          "pNext": null,
          "pApplicationName": "vkcube",
[...]
```

GFXReconstruct - `gfxrecon.py replay`

Plays graphics function call stream as close to the original as possible

```
gfxrecon.py replay your-capture-file.gfxr
```

GFXReconstruct - `gfxrecon.py replay`

Vulkan and Direct3D 12 are explicit APIs that expose low-level control of hardware

- GPU-specific memory alignment
- Hardware extensions
- Presentation modes (used to call it "swapbuffers")
- etc

But it *is* possible to replay on other drivers, GPUs, vendors (to varying degrees)

- Fix memory alignment and hesp types using "`-m`"
    - Most likely to succeed is "-m `rebind`" (completely redo all allocations)

- Can mask off extensions, ignore missing capabilities: "`--remove-unsupported`"

- Can attempt replay even on different platform with "--wsi"

GFXReconstruct - `gfxrecon.py replay`

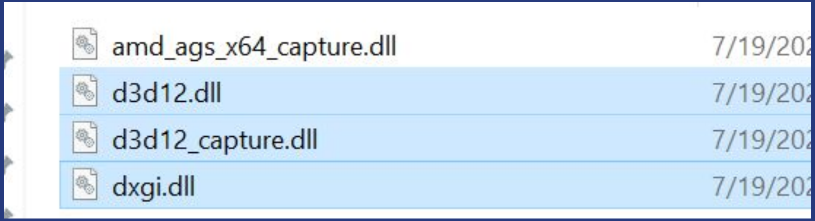
Some additional options:

- Can also skip allocations that failed in capture with “--sfa”
- Select one of multiple GPUs: “`--gpu`”
- On Android using Vulkan: “--surface-index”
  - Choose one of multiple captured surfaces to replay
- Save presented images: “--screenshots”, “--screenshot-all”

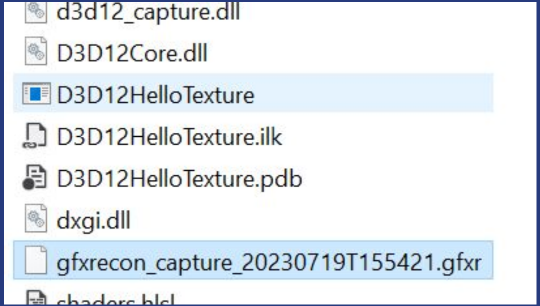
GFXReconstruct - Capturing A Direct3D 12 Application

D3D12 interception DLLs

- Included in GFXR Release Packages (on GitHub)
- Copy DLLs to your app directory
- Set optional environment variables
- Run app and exit normally
- GFXR records a ".gfxr" file



```
amd_ags_x64_capture.dll   7/19/202
d3d12.dll                 7/19/202
d3d12_capture.dll         7/19/202
dxgi.dll                  7/19/202
```



```
d3d12_capture.dll
D3D12Core.dll
D3D12HelloTexture
D3D12HelloTexture.ilk
D3D12HelloTexture.pdb
dxgi.dll
gfxrecon_capture_20230719T155421.gfxr
shaders.hlsl
```

Check out the page on GFXR and Direct3D 12 on GPUOpen.com!
https://gpuopen.com/learn/amd-lunarg-gfxreconstruct-dx12-dxr/

LUNARG

GFXReconstruct - Replay with Vulkan Validation

Useful to replay a capture to show Vulkan valid usage errors

`gfxrecon.py replay --validate`

```
$ gfxrecon.py replay --validate gfxrecon_capture_20230802T164029.gfxr
[gfxrecon] INFO - Replay has added the following required layers to VkInstanceCreateInfo
when calling vkCreateInstance:
[gfxrecon] INFO -        VK_LAYER_KHRONOS_validation
[...]
VUID-VkFenceCreateInfo-sType-sType(ERROR / SPEC): msgNum: 913590280 - Validation Error: [
VUID-VkFenceCreateInfo-sType-sType ] Object 0: handle = 0x55c0bb792ca0, type =
VK_OBJECT_TYPE_DEVICE; | MessageID = 0x36744808 | vkCreateFence: parameter
pCreateInfo->sType must be VK_STRUCTURE_TYPE_FENCE_CREATE_INFO. The Vulkan spec
states: sType must be VK_STRUCTURE_TYPE_FENCE_CREATE_INFO
(https://vulkan.lunarg.com/doc/view/1.3.250.1/linux/1.3-extensions/vkspec.html#VUID-VkFenceC
reateInfo-sType-sType)
[...]
```

GFXReconstruct - With Other Tools

`gfxrecon-replay` is just a program making graphics function calls

Capture files can be replayed inside other tools!

- RenderDoc
- NVIDIA NSight
- AMD Radeon Graphics Tools
- Etc

GFXReconstruct - Use with Android

Somewhat similar to desktop

- GFXR capture Vulkan layer is loaded within the target app
- `gfxrecon-replay` is an app that replays capture files

But Android's security model is more strict than desktop!

- Need "debuggable" app or rooted device
- Locations of loadable layers and for writing files change frequently

We provide a detailed guide to GFXR on Android

- `HOWTO_android.md` in the source tree

GFXReconstruct - `gfxrecon.py optimize`

Trimmed captures contain all objects created before the start of the trim range.

For Vulkan and Direct3D captures, `optimize`:

- Scans for unused resources
- Creates a new capture without unused resources
- Improves replay performance
- Reduces file size

GFXReconstruct - `gfxrecon.py optimize`

```
$ gfxrecon.py optimize trim.gfxr trim.opt.gfxr
Scanning F:/SaschaWillems-Vulkan-Samples/bin/trim.gfxr for unreferenced
resources.
[...]
Resource filtering complete.
        Original file size: 9588217 bytes
        Optimized file size: 6873678 bytes


$ gfxrecon.py replay trim.gfxr
[...] Replay FPS: 1514.922818 fps [...]


$ gfxrecon.py replay trim.opt.gfxr
[...] Replay FPS: 1794.189268 fps [...]
```

28% reduction

18% improvement

GFXReconstruct -- Other Tools in the Package

- `compress` - Change compression format or decompress
- `extract` - Extract shader binaries for inspection or replacement

GFXReconstruct - Practical Concerns

Prefer an SSD - file I/O is often the performance bottleneck

More RAM allows better trimmed capture performance

Some captures can't be replayed on other GPUs / platforms

- Unavailable extensions, features
- Use too much graphics memory

GFXReconstruct Architecture

Components

- `CaptureManager`, `VulkanCaptureManager`, `D3D12CaptureManager` - deal with API specifics, trimming, misc
- `Encoder` - serialize API call info and parameters
- `FileProcessor` - read blocks from a file, decompress and call `Decoders`
- `Decoder` - deserialize API call info, call `Consumers`
- `Consumer` - take API call info, do something with it
  - E.g. `VulkanReplayConsumer`
  - E.g. `Dx12StatsConsumer`

GFXReconstruct Source Code

Directory structure

- `framework/`
  - `generated/` - generators & generated code *is checked in*
  - `encode/` - capture manager, handwritten capture, state tracking
  - `decode/` - file processing, decoding, replay, and other consumers
  - `format/` - file format metacommand structs, API call IDs
  - `util/` - etc

LUNARG

GFXReconstruct Source Code

Directory structure - cont.

- `tools/` - settings, tool `main()`s, etc
- `layer/` - Vulkan API layer and D3D12 DLL code
- `scripts/` - `gfxrecon.py, build.py`

Thanks!

Release packages are available on GitHub:

https://github.com/LunarG/gfxreconstruct/releases

Binaries (without Direct3D) in the Vulkan SDK:

https://vulkan.lunarg.com/sdk/home

Source:

https://github.com/LunarG/GFXReconstruct

https://lunarg.com

info@lunarg.com

This Presentation

LUNARG

More LunarG at SIGGRAPH 2023 -

Vulkan Development in Apple Environments

Wed, Aug 9th, 9:00 - 10:30 am PDT   Room - LACC 518B

Presenters -

Bill Hollings, Brenwill Workshop

Richard Wright, LunarG Inc.

Vulkan, Forging Ahead

Wed, Aug 9th, 3:00 - 6:00 pm PDT   JW Marriott LA, Platinum Salon D

Includes a Presentation by -

Karen Ghavam, LunarG Inc. - Vulkan SDK & Ecosystem Tools

See our Vulkan demos at the LunarG table during the Networking Event!

LUNARG®