KosmicKrisp

A Vulkan to Metal Mesa driver

Aitor Camacho Larrondo

About me

- Ubisoft Berlin 2019–2020
  - Optimizing games for consoles, mainly Nintendo Switch
- Arm Norway AS 2021
  - Arm Mali GPU driver optimization
- LunarG 2022–Present
  - Vulkan Validation Layers
  - Vulkan Conformance Test Suite (CTS)
  - MoltenVK
  - KosmicKrisp

Passing 1.3 CTS!

LUNARG
POWER YOUR SUCCESS

What is KosmicKrisp?

- A Vulkan driver that translates Vulkan to Metal
- Built using Mesa's Vulkan driver framework
- Targets Apple Silicon
  - macOS: M1-series onwards
  - iOS: Not supported but designed for A14 Bionic onwards
- Baseline:
  - Vulkan 1.2 for macOS 13+
  - Vulkan 1.3 for macOS 15+
- Vulkan Conformance Test Suite for 1.3 submitted
  - Verification passed!!!

Passing 1.3 CTS!

LUNARG
POWER YOUR SUCCESS

Why do KosmicKrisp?

Passing 1.3 CTS!

- Achieve Vulkan Conformance
  - Eliminating need for Vulkan® Portability™
- Mesa provides a fast driver development environment:
  - 10 months from a blank state to passing Vulkan CTS 1.3!!!
  - NIR, removing the need for SPIRV-Cross
    - SPIRV-cross -> MSL (used by MoltenVK) isn't a compiler stack
    - Cleaner to add instructions, rewrite input/output layouts, ...
    - Remove dependency upon external project
  - Vulkan driver framework and utilities for emulation when needed
  - Broad community
  - Everything is in-tree and MIT licensed
- Fresh start allowing us to:
  - Design with new Metal features in mind
  - Drop older hardware which would difficult development

LUNARG
POWER YOUR SUCCESS

Timeline

- Jan ’24 LunarG starts work on MoltenVK
  - Goal: Vulkan conformant solution for Apple platforms
- Jun ’24 Alyssa Anne Rosenzweig releases HoneyKrisp
  - Demonstrates Vulkan 1.3 conformance on Apple hardware
  - Sparks preliminary investigation on a Mesa Vulkan® -> Metal® driver
- Oct ’24 LunarG Action plan for a Mesa Vulkan® –> Metal® driver
  - Goal: Vulkan 1.3 using Metal 3.1 as baseline, target macOS 13+
  - Need to wrap up work on MoltenVK, and ramp up on Mesa
  - Probe receptiveness
    https://gitlab.freedesktop.org/mesa/mesa/-/issues/11990
- Nov ’24 Funding secured, project starts

Passing 1.3 CTS!

LUNARG
POWER YOUR SUCCESS

Timeline (cont.)

- Nov ’24 Funding secured, project starts
- Feb ’25 First compute tests passing
  - Resources, queue submission, synchronization…
- Mar ’25 First triangle
- Apr ’25 vkcube working
- May ’25 First full CTS run completed
  - 95,086 pass; 114,168 failures; 2,538,080 not supported
- Aug ’25 SigGraph announcement and demo
- Sep ’25 No failures running Vulkan CTS for 1.3
  - Raise baseline to Metal 3.2 (macOS 15) for Vulkan 1.3
  - Keep Metal 3.1 (macOS 13) for Vulkan 1.2

Passing 1.3 CTS!

LUNAR G
POWER YOUR SUCCESS

KosmicKrisp architecture

- Project started with Metal 3.1 as baseline
  - Cover older OS to a reasonable degree
  - Metal 3.1 introduces image atomics, hard requirement
- NVK as baseline (following HoneyKrisp steps)
  - Stripped backend from NVK and started adding Metal
- Command buffers recorded at queue submission
  - MTLCommandQueue has limited command buffer count
  - Two command queues per queue: main and helper
  - Helper used only when inside a render pass that requires some translation done to the inputs, e.g. triangle fan emulation
  - Design allows to remove record at queue submission if MTLCommandQueue limitation is lifted

Passing 1.3 CTS!

KosmicKrisp architecture (cont.)

Passing 1.3 CTS!

- Considerable texture functionality through emulation
  - 1D textures unusable, emulated as 2D textures
  - No atomics for 2D cube textures, emulated as 2D texture arrays
  - Sampler bias is a function argument, not part of the sampler in Metal
- Visibility queries (same as HoneyKrisp)
  - Dedicated buffer with max queries always bound
  - Allocation/deallocation provides index to buffer
  - Reads get propagated after render encoder
- Dynamic rendering and bindless from the very beginning
  - 2 argument buffers bound at all times
    - Root argument buffer containing all required data (vbos, descriptors, etc)
    - Sampler argument buffer

LUNARG
POWER YOUR SUCCESS

KosmicKrisp architecture (cont.)

Passing 1.3 CTS!

- vk_meta to cover Metal gaps
  - Blit, fill buffer, resolve…
- NIR helped fill Metal gaps
  - Blending (Metal does not support blending on all formats)
  - All lowering for 1D textures
  - Gather operations with offsets
  - Cube map sampling with gradient
  - …
- Custom border color (same as HoneyKrisp)
  - Not exposed yet due to performance cost
- Vulkan 1.3 requires Vulkan Memory Model
  - This can only be accomplished with Metal 3.2
  - Metal Fence Functions required, namely “atomic_thread_fence”

Our biggest issues while developing

Passing 1.3 CTS!

- Metal
  - No specification, samples as the only reference
  - Invalid Metal usage can only be found through runtime validation
  - Crashes and bugs, a considerable amount. Worst one requiring reboot
- Buggy MSL compiler, e.g.
  - Crash due to uninitialized local array variable
  - Crash due to constant break condition in infinite loop
  - And many more. Workarounds will be documented in a file
- Xcode GPU debugger
  - Disparities between capture and actual result
  - Disparities between capture and shader debugger
- Documentation about how to in NIR would be beneficial

LUNARG
POWER YOUR SUCCESS

What's next?

- Merge upstream https://gitlab.freedesktop.org/mesa/mesa/-/merge_requests/37522
- 0 failures running CTS does not mean fool proof
  - Want to spend some time testing more real workloads
- Add KosmicKrisp to LunarG CI (builds, CTS, workloads)
- Add to Vulkan macOS SDK along with MoltenVK for users to choose
- Vulkan 1.4
- My bucket list in no particular order:
  - Sparse memory
  - Tessellation and Geometry shaders
  - Mesh shaders
  - Ray tracing???
  - Shader objects???

Passing 1.3 CTS!

LUNARG
POWER YOUR SUCCESS

Sponsored by Google

Passing 1.3 CTS!

"Google is committed to providing Android developers with exceptional tools across all platforms, including strong support for Vulkan. We're excited about how KosmicKrisp will enhance the Vulkan developer experience and make the Android Emulator even more powerful."

– Serdar Kocdemir – Senior Software Engineer at Google | Tech Lead for Graphics on the Android Emulator

LUNARG
POWER YOUR SUCCESS

The KosmicKrisp Team

- Aitor Camacho Larrondo
  - Overall driver architect and lead developer
- Arcady Goldmints-Orlov
  - NIR->MSL compiler
- Alyssa Anne Rosenzweig
  - Mentorship!

Passing 1.3 CTS!

Demo will showcase the Android emulator running through KosmicKrisp

LUNARG
POWER YOUR SUCCESS

Questions?

LUNARG®

POWER YOUR SUCCESS