Vulkanised 2026

The 8th Vulkan Developer Conference
San Diego, USA | February 9-11, 2026

KosmicKrisp: Conformant Vulkan for Apple Hardware

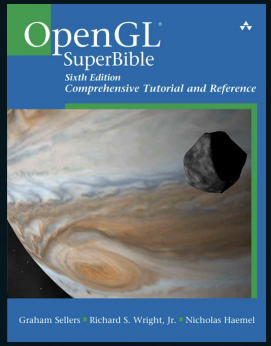
Richard Wright, LunarG

Some quick information about me

- 40+ Years a Software Developer
  - DOS -> 16-bit/32-bit Windows (all database stuff)

Some quick information about me

- 40+ Years a Software Developer
  - DOS -> 16-bit/32-bit Windows (all database stuff)
- 30 Years 3D Graphics based
  - All OpenGL until 2018



LUNARG

Some quick information about me

- 40+ Years a Software Developer
  - DOS -> 16-bit/32-bit Windows (all database stuff)
- 30 Years 3D Graphics based
  - All OpenGL until ~2018
- Application Developer at heart
  - Astronomy/Vis-Sim/Imaging
  - Discovered Apple at Starry Night
  - Windows/macOS/Linux/iOS/Android

LUNARG

Some quick information about me

- 40+ Years a Software Developer
  - DOS -> 16-bit/32-bit Windows (all database stuff)
- 30 Years 3D Graphics based
  - All OpenGL until 2018
- Application Developer at heart
  - Astronomy/Vis-Sim/Imaging
  - Discovered Apple at Starry Night
  - Windows/macOS/Linux/iOS/Android
- Joined LunarG 6+ years ago
  - Part of the Vulkan SDK team at LunarG
  - "Apple Champion" at LunarG
  - KosmicKrisp Cheerleader

LUNARG

My Agenda is to show you:

- Why you should care about the Apple market
- That Vulkan on Apple hardware works great
- Why KosmicKrisp may work for you
- How to get & use KosmicKrisp
- How to support the widest range of Apple hardware

Don’t ignore the Apple market!

- There are 1.56 billion people actively using an iPhone and about 700 million with iPads.
  (27%–28% vs 71%–73%)
- There are over 500 million people on Mac laptops/Desktops
  (15% vs 70%–22%)
- 2.35 Billion+ Active devices

LUNARG

Don’t ignore the Apple market!

- There are 1.56 billion people actively using an iPhone and about 700 million with iPads.
  (27%–28% vs 71%–73%)
- There are over 500 million people on Mac laptops/Desktops
  (15% vs 70%–22%)
- 2.35 Billion+ Active devices
- The Mac Vulkan SDK downloads Rivals/exceeds Linux

Writing Cross Platform code has challenges

- Tooling

Writing Cross Platform code has challenges

- Tooling
- Programing languages

Writing Cross Platform code has challenges

- Tooling
- Programing languages
- All sorts of OS/Device idiosyncrasies

Writing Cross Platform code has challenges

- Tooling
- Programing languages
- All sorts of OS/Device idiosyncrasies
- Wouldn’t it be nice if you didn’t have to port your rendering code?

Writing Cross Platform code has challenges

- Tooling
- Programing languages
- All sorts of OS/Device idiosyncrasies
- Wouldn’t it be nice if you didn’t have to port your rendering code?
- TADA: That’s kind of the whole point of Vulkan?

The State of Vulkan on Apple devices

- No “native” Vulkan driver on macOS?

The State of Vulkan on Apple devices

- No “native” Vulkan driver on macOS?
- We layer Vulkan over Metal instead
  - KosmicKrisp
  - MoltenVK

LUNARG

The State of Vulkan on Apple devices

- No “native” Vulkan driver on macOS?
- We layer Vulkan over Metal instead
  - KosmicKrisp
  - MoltenVK
- This “driver” is included in your app bundle

LUNARG

Hot Tips

- Same performance caveats as Metal

Hot Tips

- Same performance caveats as Metal
- Tile Based Deferred Rendering (TBDR)
  - Similar to Qualcomm ARM laptops
  - Similar to most Android Devices
  - Pretty much exactly like iOS/iPad

Hot Tips

- Same performance caveats as Metal
- Tile Based Deferred Rendering (TBDR)
  - Similar to Qualcomm ARM laptops
  - Similar to most Android Devices
  - Pretty much exactly like iOS/iPad
- Don’t interleave compute and graphics
  - Switching Metal Encoders is expensive
  - See if you can’t accomplish what you want in the vertex/fragment shaders
  - Don’t assume compute is always a win…

Hot Tips

- Same performance caveats as Metal
- Tile Based Deferred Rendering (TBDR)
  - Similar to Qualcomm ARM laptops
  - Similar to most Android Devices
  - Pretty much exactly like iOS/iPad
- Don’t interleave compute and graphics
  - Switching Metal Encoders is expensive
  - See if you can’t accomplish what you want in the vertex/fragment shaders
  - Don’t assume compute is always a win…
- Screen Scaling may be a win – you do not have to render at retina resolutions

Why KosmicKrisp?

- Mesa is a much larger community
  - Already getting 3rd party PR's
  - RenderDoc requires Vulkan conformant implementation
  - SO MUCH emulation work is already done

Why KosmicKrisp?

- Mesa is a much larger community
  - Already getting 3rd party PR's
  - RenderDoc requires Vulkan conformant implementation
  - SO MUCH emulation work is already done
- NIR compiler stack – easier to massage shaders
  - Apple hardware consumes MSL
  - SPIRV -> MSL becomes much more sane
  - NIR is a first party compiler stack – less reliance on 3rd party tools (SPRIV-CROSS)

Why KosmicKrisp?

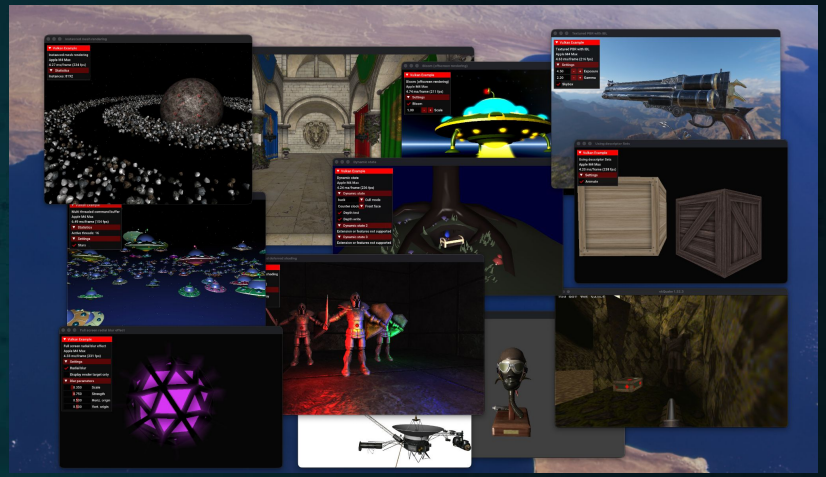
- Mesa is a much larger community
  - Already getting 3rd party PR's
  - RenderDoc requires Vulkan conformant implementation
  - SO MUCH emulation work is already done
- NIR compiler stack – easier to massage shaders
  - Apple hardware consumes MSL
  - SPIRV -> MSL becomes much more sane
  - NIR is a first party compiler stack – less reliance on 3rd party tools (SPRIV-CROSS)
- Bindless design from the outset

Why KosmicKrisp?

- Mesa is a much larger community
  - Already getting 3rd party PR's
  - RenderDoc requires Vulkan conformant implementation
  - SO MUCH emulation work is already done
- NIR compiler stack – easier to massage shaders
  - Apple hardware consumes MSL
  - SPIRV -> MSL becomes much more sane
  - NIR is a first party compiler stack – less reliance on 3rd party tools (SPRIV-CROSS)
- Bindless design from the outset
- No more Portability extension...

KosmicKrisp Roadmap

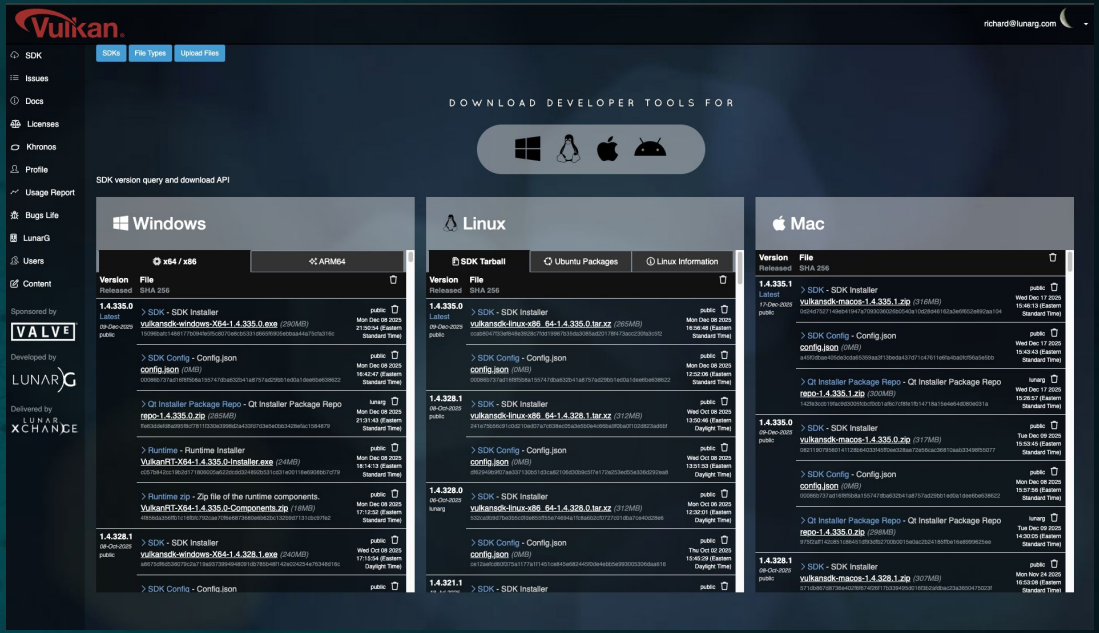
- 1.3 conformant now
- Part of Mesa 3D – so moving forward quickly
- Apple Silicon Only
- macOS 26+
- Metal 4
- 3–6 month priority list
  - Tessellation/Geometry
  - Performance
  - 1.4 Conformance
  - Shader objects and descriptor heap
- Debug features (RenderDoc!)
- iOS support



LUNARG

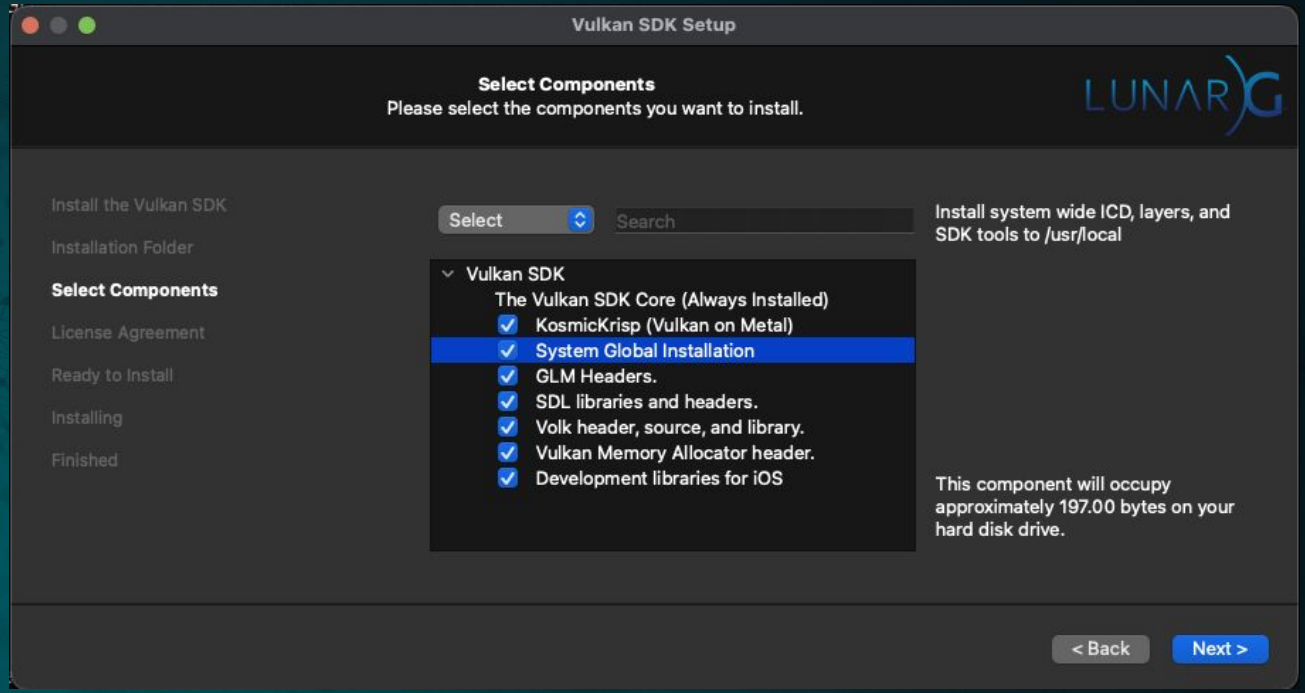
Where do we get these “drivers”?

Included in the Vulkan SDK available free at: vulkan.lunarg.org



LUNARG

System Wide Loader/ICD*

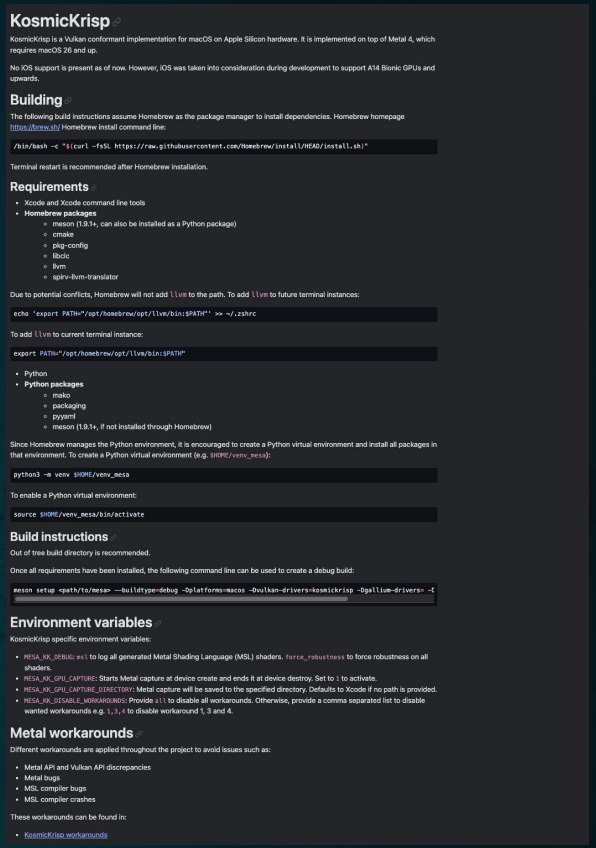


*For developers only – not end users!

LUNARG

Build it yourself!

https://docs.mesa3d.org/drivers/kosmickrisp.html



KosmicKrisp

KosmicKrisp is a Vulkan conformant implementation for macOS on Apple Silicon hardware. It is implemented on top of Metal 4, which requires macOS 26 and up.

No iOS support is present as of now. However, iOS was taken into consideration during development to support A14 Bionic GPUs and upwards.

Building

The following build instructions assume Homebrew as the package manager to install dependencies. Homebrew homepage https://brew.sh/ Homebrew install command line:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Terminal restart is recommended after Homebrew installation.

Requirements

- Xcode and Xcode command line tools
- **Homebrew packages**
  - meson (1.9.1+, can also be installed as a Python package)
  - cmake
  - pkg-config
  - libclc
  - llvm
  - spirv-llvm-translator

Due to potential conflicts, Homebrew will not add `llvm` to the path. To add `llvm` to future terminal instances:

```
echo 'export PATH="/opt/homebrew/opt/llvm/bin:$PATH"' >> ~/.zshrc
```

To add `llvm` to current terminal instance:

```
export PATH="/opt/homebrew/opt/llvm/bin:$PATH"
```

- Python
- **Python packages**
  - mako
  - packaging
  - pyyaml
  - meson (1.9.1+, if not installed through Homebrew)

Since Homebrew manages the Python environment, it is encouraged to create a Python virtual environment and install all packages in that environment. To create a Python virtual environment (e.g. `$HOME/venv_mesa`):

```
python3 -m venv $HOME/venv_mesa
```

To enable a Python virtual environment:

```
source $HOME/venv_mesa/bin/activate
```

Build instructions

Out of tree build directory is recommended.

Once all requirements have been installed, the following command line can be used to create a debug build:

```
meson setup <path/to/mesa> --buildtype=debug -Dplatforms=macos -Dvulkan-drivers=kosmickrisp -Dgallium-drivers= -D
```

Environment variables

KosmicKrisp specific environment variables:

- `MESA_KK_DEBUG`: `msl` to log all generated Metal Shading Language (MSL) shaders. `force_robustness` to force robustness on all shaders.
- `MESA_KK_GPU_CAPTURE`: Starts Metal capture at device create and ends it at device destroy. Set to `1` to activate.
- `MESA_KK_GPU_CAPTURE_DIRECTORY`: Metal capture will be saved to the specified directory. Defaults to Xcode if no path is provided.
- `MESA_KK_DISABLE_WORKAROUNDS`: Provide `all` to disable all workarounds. Otherwise, provide a comma separated list to disable wanted workarounds e.g. `1,3,4` to disable workaround 1, 3 and 4.

Metal workarounds

Different workarounds are applied throughout the project to avoid issues such as:

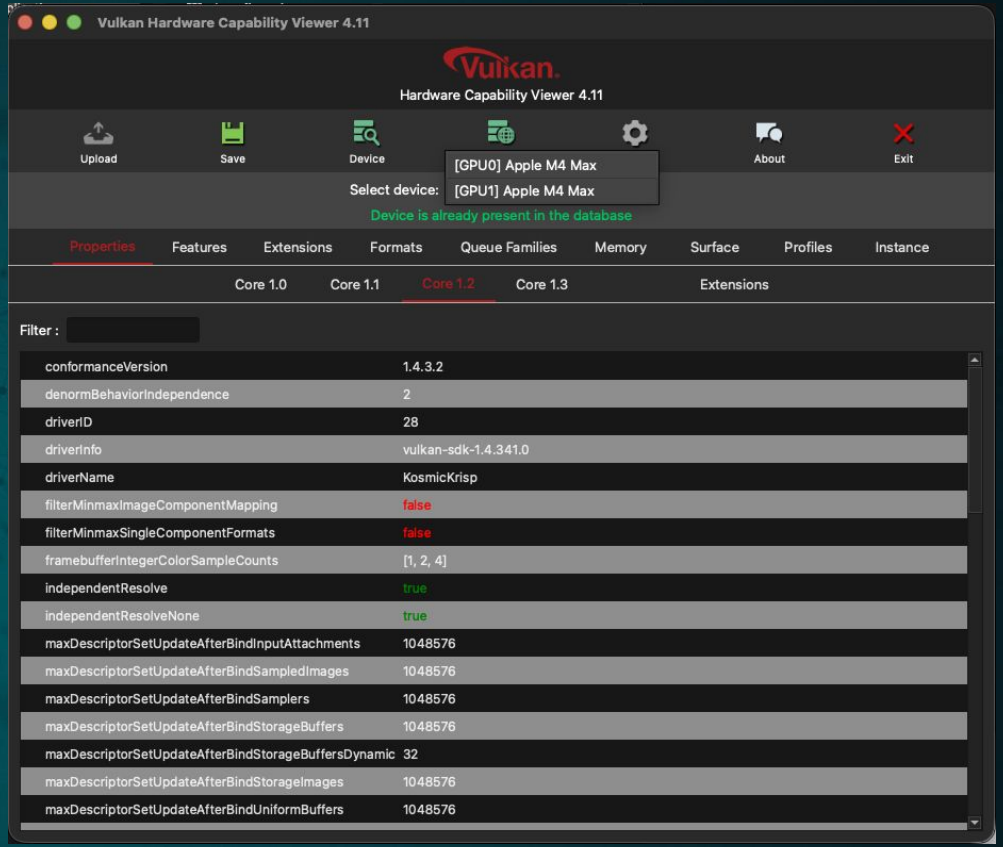
- Metal API and Vulkan API discrepancies
- Metal bugs
- MSL compiler bugs
- MSL compiler crashes

These workarounds can be found in:

- KosmicKrisp workarounds

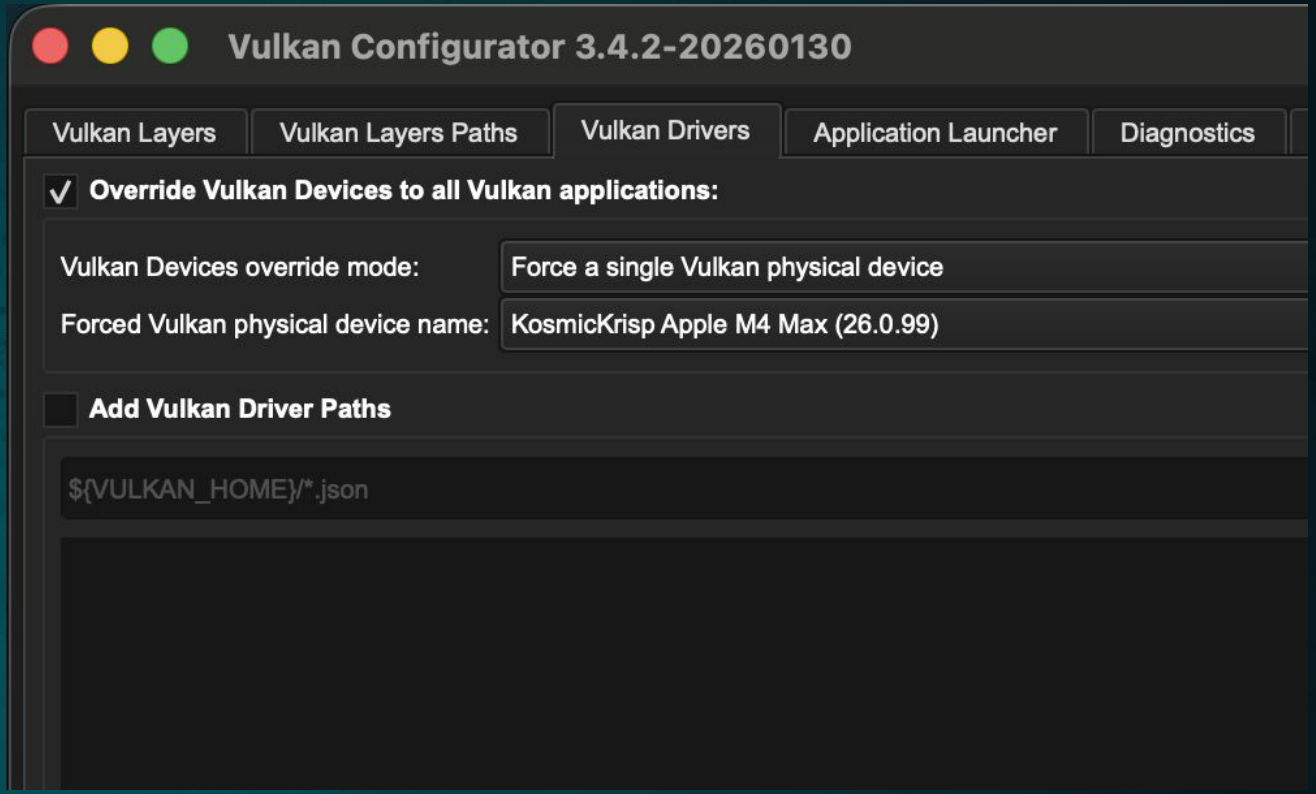
LUNARG

Hardware Caps Viewer



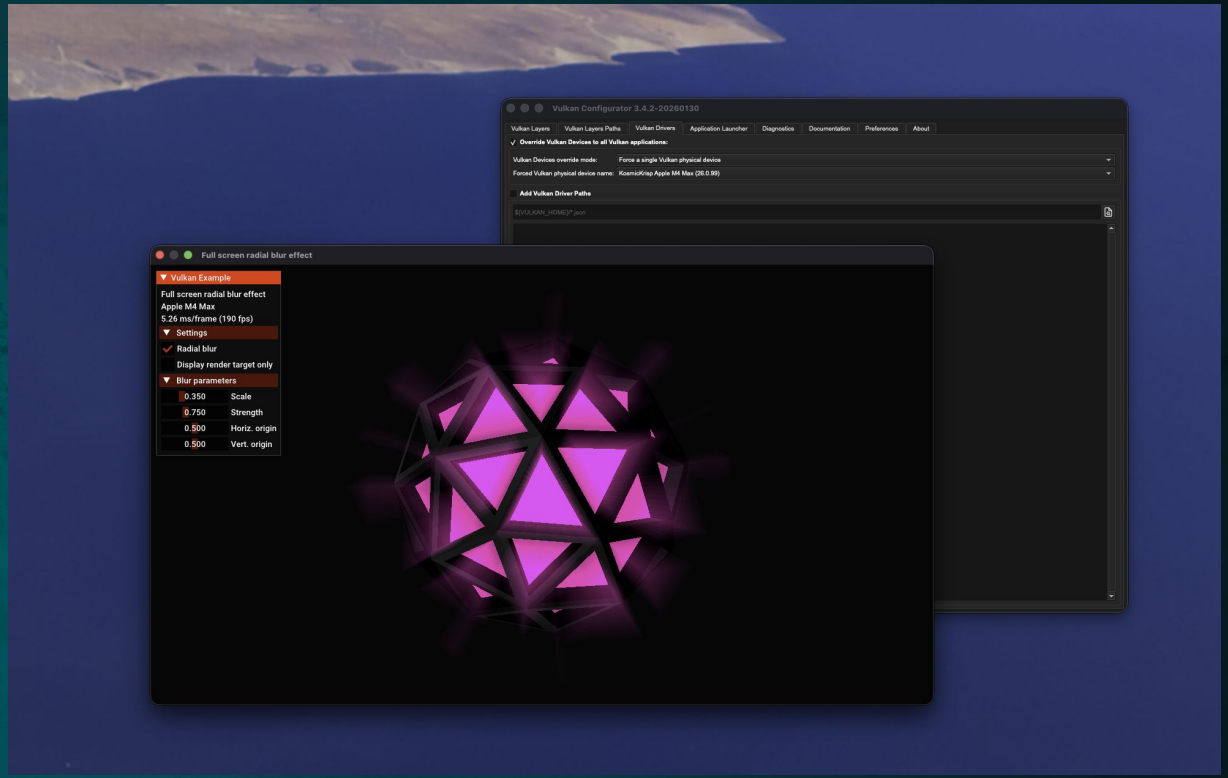
LUNARG

Vulkan Configurator



LUNARG

Vulkan Configurator



LUNARG

Android Emulator*

- Primary funding is from Google

*Not public yet

LUNARG

Android Emulator*

- Primary funding is from Google
- Motivation? – Android Emulator on macOS

*Not public yet

LUNARG

Android Emulator*

- Primary funding is from Google
- Motivation? – Android Emulator on macOS
  - Vulkan is the deFacto graphics stack now on Android
  - Via Angle -> GUI and all OpenGL ES games are now basically Vulkan
  - For the emulator on macOS – Vulkan can be implemented by KosmicKrisp

*Not public yet

LUNARG

Shipping Vulkan on Apple

```
VulkanRocks.app
    /Contents
        /Frameworks
            libvulkan_kosmickrisp.dylib
            libvulkan.1.[version number].dylib
            libvulkan.1.dylib -> libvulkan.1.[version number].dylib
        /MacOS
            VulkanRocks
        /Resources
            /vulkan
                /icd.d
                    libkosmickrisp_icd.json
```


LUNARG

Shipping Vulkan(s) on Apple

```
VulkanRocks.app
    /Contents
        /Frameworks
            libvulkan_kosmickrisp.dylib
            libMoltenVK.dylib
            libvulkan.1.[version number].dylib
            libvulkan.1.dylib -> libvulkan.1.[version number].dylib
        /MacOS
            VulkanRocks
        /Resources
            /vulkan
                /icd.d
                    libkosmickrisp_icd.json
                    libMoltenVK_icd.json
```


LUNARG

Why not both?

- You could target Intel HW with MoltenVK
- Target Apple Silicon with KosmicKrisp

Why not both?

- You could target Intel HW with MoltenVK
- Target Apple Silicon with KosmicKrisp
- iOS support is coming…

Why not both?

- You could target Intel HW with MoltenVK
- Target Apple Silicon with KosmicKrisp
- iOS support is coming…
- Watch Portability enumeration & Portability Subset
  - KosmicKrisp implements neither of these
  - It doesn't need them

LUNARG

No more of THIS!

```
VkInstanceCreateInfo inst_info = {};
std::vector<const char*> m_requiredExtensions;
        . . .
#ifdef __APPLE__

m_requiredExtensions.push_back(VK_KHR_PORTABILITY_ENUMERATION_EXTENSION_NAME);
inst_info.flags |= VK_INSTANCE_CREATE_ENUMERATE_PORTABILITY_BIT_KHR;

#endif
        . . .
        . . .
```

Why not both?

VK_KHR_portability_enumeration

VK_KHR_portability_subset

If you do not include these extensions, the loader will only see KosmicKrisp. This is normal behavior outside macOS, but MoltenVK is a “portability implementation”. Enable this only if present and no more #ifdef __APPLE__

Kosmic Conclusion

- Don’t ignore Apple – there are a **LOT** of devices out there

Kosmic Conclusion

- Don’t ignore Apple – there are a **LOT** of devices out there
- There are TWO good Vulkan implementations for Apple

Kosmic Conclusion

- Don’t ignore Apple – there are a **LOT** of devices out there
- There are TWO good Vulkan implementations for Apple
- KosmicKrisp is likely to be the defacto standard soon for Apple Silicon on all devices

Kosmic Conclusion

- Don’t ignore Apple – there are a **LOT** of devices out there
- There are TWO good Vulkan implementations for Apple
- KosmicKrisp is likely to be the defacto standard soon for Apple Silicon on all devices
- MoltenVK will remain relevant for quite some time – and you can eat your cake and have it too – ship both of them!

Worms in the Apple?

Report issues to Mesa3D repository

Prefix “kk: title” so it stands out

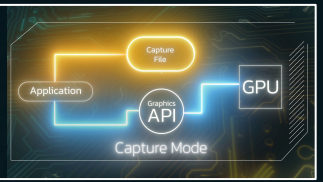
LUNARG
POWER YOUR SUCCESS

Thank you –
KHRONOS GROUP

KosmicKrisp

Come to the LunarG Table!

See KosmicKrisp & GFXReconstruct



Take the 2026 Vulkan Ecosystem Survey!

LunarG Presentations

Vulkanised 2026

LunarG Presentations

Shading Languages Symposium 2026


LUNARG
POWER YOUR SUCCESS