

Vulkanised 2024

The 6th Vulkan Developer Conference
Sunnyvale, California | February 5-7, 2024

8 years of open drivers

Faith Ekstrand, Collabora
Iago Toral, Igalia



About Iago

- Graphics engineer at Igalia
- Contributor to Mesa since 2014
- Initial contributions in Mesa to Intel drivers for OpenGL and Vulkan
- Currently leading the Raspberry Pi 3D driver team at Igalia, mostly focused on the Vulkan stack



About Faith

- Active member of the Vulkan working group since before 1.0
- Led Linux Vulkan driver development at Intel for 6 years
- Now at Collabora working on Linux graphics over-all
- Mesa Maintainer, focused on:
 - NIR optimizing shader compiler
 - SPIR-V front-end
 - Common Vulkan runtime
 - NVK, the new open-source driver for NVIDIA hardware



History of Vulkan in Mesa

- Mesa Vulkan work started May 8, 2015 (that's before the 1.0 release 😊)
- Initial Intel driver was released on February 16, 2016
 - Released in tandem with the Vulkan 1.0 spec
 - One of the first 4 conformant implementations
 - Merged into mesa/main about 2 months later
- The RADV driver for AMD hardware landed October 7, 2016
 - Initially not conformant, but some apps ran
- Since then, we've gained drivers for nearly every major HW vendor:
 - Qualcomm (Turnip), Arm (Panvk), Broadcom (V3DV), Imagination (PVR), and NVIDIA (NVK)
- Lavapipe, utilizing the LLVMpipe CPU rasterization core



History of Vulkan in Mesa

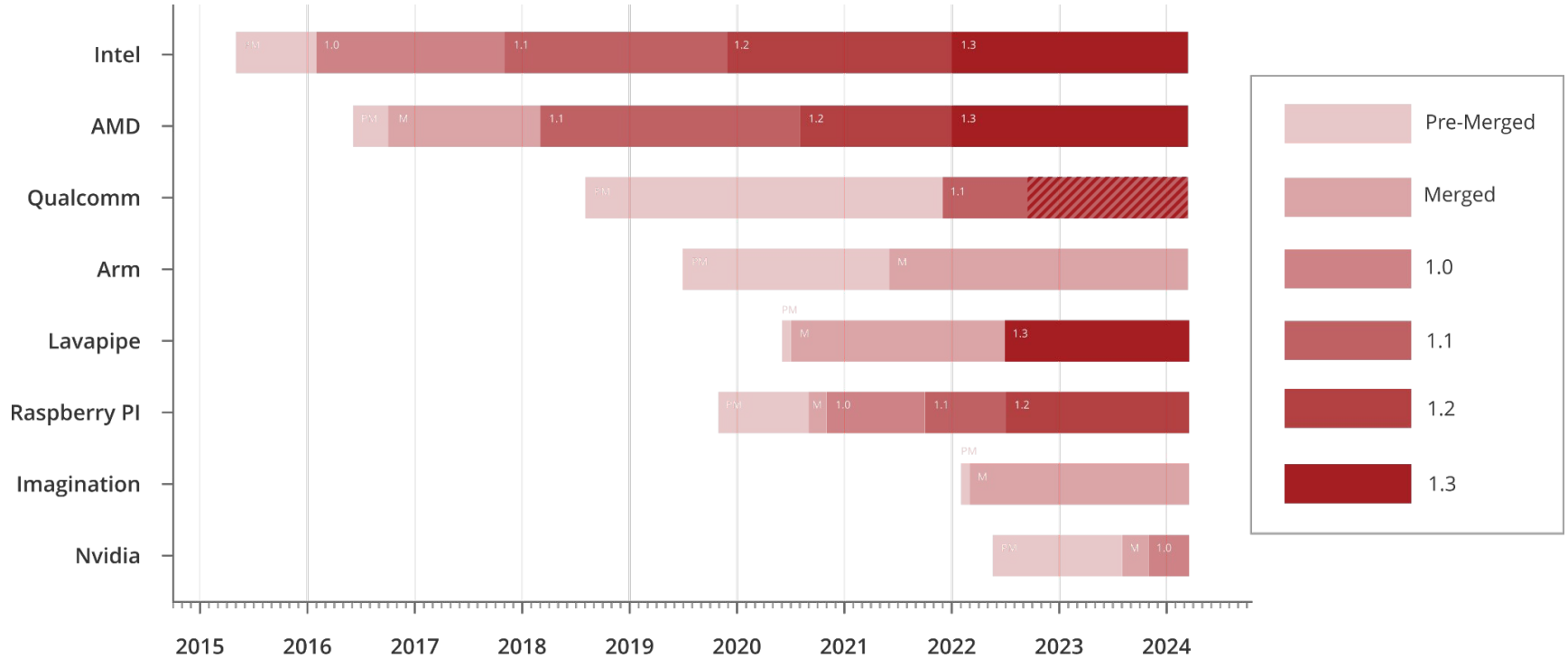
Mesa also has a common runtime, shared across HW drivers

- Some of this was built as part of the initial Intel efforts:
 - NIR optimizing shader compiler core
 - SPIR-V front-end for NIR
 - Window-system code for X11 and Wayland
- Other pieces have been added later:
 - Dispatch and `vkGet*ProcAddr()` handling
 - `vkQueueSubmit()` threading and timeline semaphores
 - Render passes in terms of `VK_KHR_dynamic_rendering`
 - Pipelines in terms of `VK_EXT_shader_object` (still WIP)



A timeline of Vulkan in Mesa

As of February 1, 2024



The Mesa Matrix (<https://mesamatrix.net>)

Vulkan 1.3

Extension	AMD		Arm	Broadcom	Intel		Nvidia	PowerVR	Qualcomm	Software	Translation	
	mesa	radv	panvk	v3dv	anv	hasvk	nvk	pvr	tu	lvp	dzn	vn
	100.0%	100.0%	8.7%	78.3%	100.0%	91.3%	95.7%	21.7%	100.0%	100.0%	13.0%	100.0%
VK_KHR_copy_commands2						2023-07-06	2023-08-04	2023-08-10				
VK_KHR_dynamic_rendering						2023-07-06	2023-08-04					
VK_KHR_format_feature_flags2						2023-07-06	2023-08-04	2023-09-05				
VK_KHR_maintenance4						2023-07-06	2023-08-04					
VK_KHR_shader_integer_dot_product						2023-07-06	2023-12-06				2023-05-15	
VK_KHR_shader_non_semantic_info						2023-07-06	2023-08-04					
VK_KHR_shader_terminate_invocation				2023-12-15		2023-07-06	2023-11-21					
VK_KHR_synchronization2						2023-07-06	2023-11-17				2023-05-15	
VK_KHR_zero_initialize_workgroup_memory						2023-07-06	2024-01-14					
VK_EXT_4444_formats						2023-07-06	2023-08-04					
VK_EXT_extended_dynamic_state						2023-07-06	2023-08-04					
VK_EXT_extended_dynamic_state2						2023-07-06	2023-08-04					
VK_EXT_inline_uniform_block						2023-07-06	2023-08-04					
VK_EXT_pipeline_creation_cache_control						2023-07-06	2024-01-09					
VK_EXT_pipeline_creation_feedback						2023-07-06	2024-01-09					
VK_EXT_private_data						2023-07-06	2023-08-04					
VK_EXT_image_robustness						2023-07-06	2023-08-04					
VK_EXT_shader_demote_to_helper_invocation				2024-01-09		2023-07-06	2023-11-21					
VK_EXT_subgroup_size_control				2024-01-11		2023-07-06	2023-12-05					
VK_EXT_texel_buffer_alignment						2023-07-06	2023-12-14	2023-10-20				
VK_EXT_texture_compression_astc_hdr												



Layered drivers

- Venus:
 - VirtIO-GPU Vulkan driver.
 - Allows guest OS to use the host's native Vulkan implementation.
 - Merged upstream in April 2021.
- Dozen:
 - Vulkan over Direct3D 12
 - Vulkan support on Microsoft platforms where native Vulkan is not available.
 - Merged upstream in March 2022.
- Zink:
 - OpenGL over Vulkan.
 - OpenGL support on platforms where native OpenGL is not available.
 - Merged upstream in October 2019.



Useful environment variables

- MESA_VK_TRACE=rmv,rgp,rra
 - Comma-separated list of trace types to dump for offline analysis.
 - All currently supported trace types are for AMD Radeon tools:
 - Radeon Memory Visualizer: rmv (RADV, ANV)
 - Radeon GPU profiler: rgp (RADV)
 - Radeon Raytracing Analyzer: rra (RADV)
- MESA_VK_TRACE_FRAME
 - Specifies a frame index to capture.
- MESA_VK_TRACE_TRIGGER
 - Triggers frame capture when specified trigger file is created.



Useful environment variables

- `MESA_VK_WSI_PRESENT_MODE=fifo|relaxed|mailbox|immediate`
 - Overrides present mode in `VkSwapchainCreateInfoKHR::presentMode`.
- `MESA_VK_HEADLESS_SWAPCHAIN`
 - Forces all swapchains to be headless, as if `VK_EXT_headless_surface` was used.
- `MESA_VK_ABORT_ON_DEVICE_LOSS (*)`
 - Aborts execution upon seeing `VK_ERROR_DEVICE_LOST`.
- `MESA_VK_ENABLE_SUBMIT_THREAD`
 - Forces use of a submit thread on all queue submissions.



Useful environment variables

- Per-driver variables
 - These expose many driver-specific options.
 - Some accept 'help' as value to dump the list of accepted options.
 - INTEL_DEBUG (Anvil)
 - RADV_DEBUG, ACO_DEBUG (Radv)
 - TU_DEBUG (Turnip)
 - V3D_DEBUG (V3DV)
 - NVK_DEBUG (NVK)
 - PANVK_DEBUG (Panvk)



Useful Vulkan layers

- `VK_LAYER_MESA_device_select`
 - Forces device selection on platforms with multiple Vulkan implementations available.
 - Build with `-Dvulkan-layers=device-select`

```
$ MESA_VK_DEVICE_SELECT=list vkcube
selectable devices:
GPU 0: 8086:46a6 "Intel(R) Graphics (ADL GT2)" integrated GPU
GPU 1: 14e4:46a6 "V3D 4.2.14" integrated GPU

$ MESA_VK_DEVICE_SELECT=8086:46a6 vkcube
Selected GPU 0: Intel(R) Graphics (ADL GT2), type: IntegratedGpu
```



Useful Vulkan layers

- `VK_LAYER_MESA_overlay`
 - Renders an overlay with selected statistics.
 - Build with `-Dvulkan-layers=overlay`.
 - `VK_LAYER_MESA_OVERLAY_CONFIG=help` for available stats.



Benefits of open drivers

- Linux OS integration
 - You already have a Mesa driver 😊
 - Packaged by all the Linux and BSD distros
 - No binary downloads from independent 3rd parties.



Benefits of open drivers

- Linux OS integration
- Application debugging
 - We aren't hiding anything! (You can look at the code)
 - All the Driver developer tools are available to you
 - Environment variables
 - Command buffer streams
 - Shader compiler dumps
 - Most drivers support `VK_pipeline_executable_properties` integration for RenderDoc.



Benefits of open drivers

The screenshot displays a graphics driver development tool interface. At the top, a timeline shows the execution of various passes, including 'Depth-only Pass #1' and 'Colour Pass #1 (3 Targets + Depth)'. Below the timeline, the 'Event Browser' shows a list of events, with 'vkEndCommandBuffer(Command Buffer 21840)' selected at EID 1252. The main window shows the disassembly of the shader module, with the following code visible:

```
656 ---- GEN Assembly ----
657
658 ; Final GEN assembly for the generated shader binary
659
660
661 mov(8) g72<1>F g2<8, 8, 1>F { align1 WE_all 10 compacted };
662 mov(8) g73<1>F g4<8, 8, 1>F { align1 WE_all 10 compacted };
663 mov(8) g63<1>F g3<8, 8, 1>F { align1 WE_all 10 compacted };
664 mov(8) g64<1>F g5<8, 8, 1>F { align1 WE_all 10 compacted };
665 mul(16) g24<1>D g8<3<0, 1, 0>D 80W { align1 IH compacted };
666 sync nop(1) null<0, 1, 0>UB { align1 WE_all IN @2 };
667 mad(16) g65<1>F g10<3<0, 1, 0>F g18<1<0, 1, 0>F g63<1, 1, 1>F { align1 IH };
668 mad(16) g67<1>F g10<7<0, 1, 0>F g10<5<0, 1, 0>F g63<1, 1, 1>F { align1 IH };
669 mad(16) g69<1>F g11<3<0, 1, 0>F g11<1<0, 1, 0>F g63<1, 1, 1>F { align1 IH };
670 send(16) g16UD nullUD 0x4805000 0x00000000
671 dp data 1 MsgDesc: (untyped surface read, Surface = 8, SIMD16, Mask = 0x0) mlen 2 ex_mlen 0 rlen 8 { align1 IH @ $0 };
672 add(16) g32<1>D g24<8, 8, 1>D 32D { align1 IH compacted };
673 add(16) g40<1>D g24<8, 8, 1>D 48D { align1 IH compacted };
674 add(16) g54<1>D g24<8, 8, 1>D 68D { align1 IH compacted };
675 sync nop(1) null<0, 1, 0>UB { align1 WE_all IN @6 };
676 mad(16) g48<1>F g65<8, 8, 1>F g10<8<0, 1, 0>F g72<1, 1, 1>F { align1 IH compacted };
677 mad(16) g67<8, 8, 1>F g10<4<0, 1, 0>F g72<1, 1, 1>F { align1 IH @6 compacted };
678 mad(16) g52<1>F g69<8, 8, 1>F g11<8<0, 1, 0>F g72<1, 1, 1>F { align1 IH @6 compacted };
679 send(16) g24UD nullUD 0x4805000 0x00000000
680 dp data 1 MsgDesc: (untyped surface read, Surface = 8, SIMD16, Mask = 0x0) mlen 2 ex_mlen 0 rlen 8 { align1 IH @ $1 };
681 sync nop(1) null<0, 1, 0>UB { align1 WE_all IN $1.src };
682 send(16) g32UD nullUD 0x4805000 0x00000000
683 dp data 1 MsgDesc: (untyped surface read, Surface = 8, SIMD16, Mask = 0x0) mlen 2 ex_mlen 0 rlen 8 { align1 WE @ $2 };
684 sync nop(1) null<0, 1, 0>UB { align1 WE_all IN $2.src };
685 send(16) g40UD g54UD nullUD 0x4805000 0x00000000
686 dp data 1 MsgDesc: (untyped surface read, Surface = 8, SIMD16, Mask = 0x0) mlen 2 ex_mlen 0 rlen 8 { align1 IH @ $3 };
687 cmp.g.f0.0(16) null<1>UD g36<8, 8, 1>UD 0x00000000UD { align1 IH $2.dst };
688 (+f0.0) if(16) JIP: LABEL0 UIP: LABEL0 { align1 IH };
689 mad(16) g70<1>F g12<3<0, 1, 0>F g12<1<0, 1, 0>F g63<1, 1, 1>F { align1 IH };
690 mad(16) g74<1>F g12<7<0, 1, 0>F g12<5<0, 1, 0>F g63<1, 1, 1>F { align1 IH };
691 mad(16) g76<1>F g13<3<0, 1, 0>F g13<1<0, 1, 0>F g63<1, 1, 1>F { align1 IH };
692 mad(16) g82<1>F g14<3<0, 1, 0>F g14<1<0, 1, 0>F g63<1, 1, 1>F { align1 IH };
693 mad(16) g88<1>F g14<7<0, 1, 0>F g14<5<0, 1, 0>F g63<1, 1, 1>F { align1 IH };
```

At the bottom, the 'Input Signature' and 'Output Signature' tables are visible:

Name	Index	Reg	Type	SysValue	Mask	Used
in_uv	0	float2	Undefined	RG	RG	
in_material_idx	1	uint	Undefined	R	R	
in_eye_normal	2	float3	Undefined	RGB	RGB	

Name	Index	Reg	Type	SysValue	Mask	Used
out_eye_normal	-	float4	Color Output	RGBA	RGBA	
out_diffuse	-	float4	Color Output	RGBA	RGBA	
out_specular	-	float4	Color Output	RGBA	RGBA	



Benefits of open drivers

- Linux OS integration
- Application debugging
- Source code access
 - You can look under the hood
 - See how software drives the actual hardware
 - Build in debug mode and set breakpoints inside the driver
 - Maybe you can even try to fix your issue yourself!



Benefits of open drivers

- Linux OS integration
- Application debugging
- Source code access
- Better access to driver developers
 - We respond directly to issues
 - <https://gitlab.freedesktop.org/mesa/mesa/-/issues/>
 - You can join our IRC channels and ask questions



Benefits of open drivers



Julien Barnoin

@julienbarnoin@mastodon.gamedev.place

Sometimes open source really delivers 😍

Working on my Vulkan-based engine I've been hitting this obscure shader bug for a few months, and on sunday I finally created and shared a repro program and submitted a bug report to Mesa. @pixelcluster immediately investigated and delivered a fix, it was merged into main the next day, and then released on wednesday in Mesa 23.3.4 .

A 3 day turnaround time to release is pretty incredible. Thanks to all who contributed !

gitlab.freedesktop.org/mesa/me...



Benefits of open drivers

Relaxed atomic loads in while loops being optimized away

Edit



Closed Issue created 2 years ago by

Describe the issue

It appears that while loops that include a relaxed atomic memory load are being optimized away. Here are a couple amber tests that show the problem: <https://gist.github.com/reeselevine/935febcc7a8c4c192c234c54522f0cb0>

The infinite loop test should hang, since the flag is never updated, and on an Nvidia Quadro RTX 4000 it does, as expected. However, on the Intel GPU, the test passes immediately.

To confirm this is not due to an optimization that is simply removing an infinite loop, I wrote a second example that mimics message passing, where one thread updates data and sets a flag, while the other spins on the flag and then loads the data. This test should always pass, however on the Intel GPU the amber test fails.

Interestingly, if the commented out fence in the message-passing amber test (line 98) is uncommented, the test passes. It also happens if the load on line 96 is strengthened to an acquire. Therefore, this issue seems only to affect relaxed memory accesses.





[Redacted] · 2 years ago

Author    

Hey [Redacted], [Redacted] and I talked more about this today, and came to a better understanding of what's going on. The issue is that while it's true we can't rely on relaxed atomics for synchronization, the relaxed spin loop along with the following `OpMemoryBarrier` are enough to induce the ordering [Redacted] mentioned above, so we should expect the test to pass. So from the perspective of the memory model, it's not true that the loop has no side effects.

I went ahead and modified the amber test I have above here: <https://gist.github.com/reeselevine/935febcc7a8c4c192c234c54522f0cb0#file-spooky-message-passing-amber>

As written, the test fails, but if store on line 104 is uncommented, the test passes. It does appear to be a little spooky, in that observing the result of the loop completely changes the semantics of the program.



[Redacted] · 2 years ago

Developer    

[Redacted] I agree with the argument that practically speaking, would be nice if the loop did not get removed, so I'll try to have an MR to handle that and we can test it. On top of that, I think other atomic operations (e.g. max) would've prevented the loop to be removed, so making the atomic load behave the same is consistent.

However, I'm still a bit puzzled with the expectations about relaxed memory ordering. Still thinking about the loop example: Is there a guarantee that the relaxed atomic store will be visible by a relaxed atomic load in another thread? Or an infinite loop could be a valid result?



Benefits of open drivers

- Linux OS integration
- Application debugging
- Source code access
- Better access to driver developers
- Better cross-company collaboration
 - All the Mesa developers talk to each other
 - We all work on the common runtime and shader compiler
 - Most of us are also Khronos members
 - Pre-release extension work happens in Khronos gitlab



Want to help out?

Mesa lives at <https://gitlab.freedesktop.org/mesa/mesa/>

Mesa / mesa



🔔 ▾ ☆ Star 447 🍴 Fork 1054 ⋮

↩ 183,646 Commits 🌐 56 Branches 🏷 818 Tags 📦 12.8 TiB Project Storage

Mesa 3D graphics library

coverity passed 21 new defects



nir/lower_shader_calls: remove CF before nir_opt_if ⋮

e465ac25



authored 4 days ago

main ▾ mesa / + ▾

History Find file Edit ▾ Code ▾

📖 README

🔗 CI/CD configuration

📄 Add LICENSE

📄 Add CHANGELOG

📄 Add CONTRIBUTING

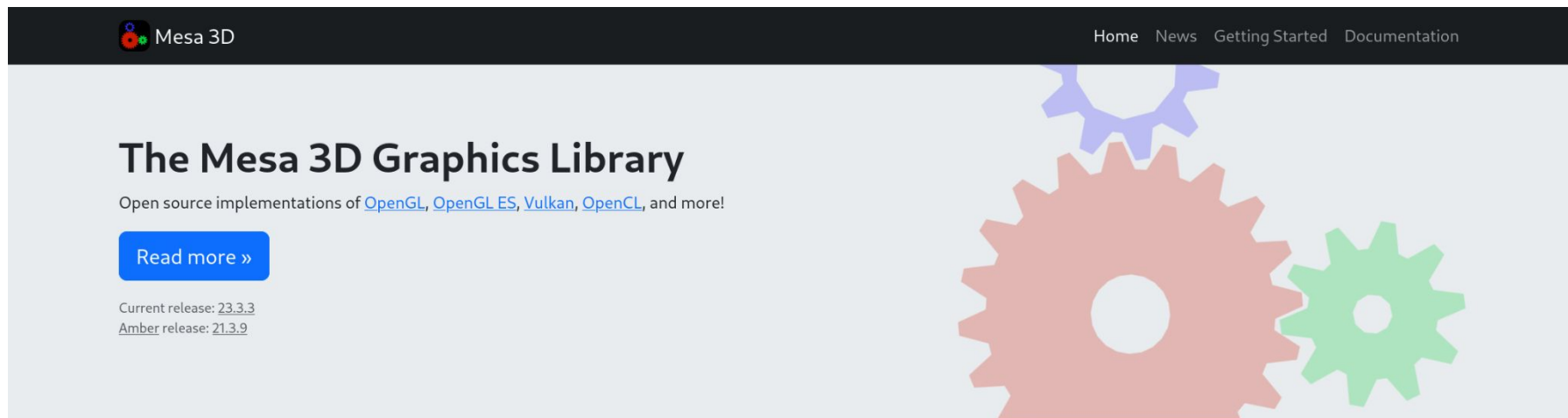
⚙ Configure Integrations



Want to help out?

Mesa lives at <https://gitlab.freedesktop.org/mesa/mesa/>

And also <https://mesa3d.org>

A screenshot of the Mesa 3D website homepage. The page has a dark grey header with the Mesa 3D logo and navigation links: Home, News, Getting Started, and Documentation. The main content area is light grey and features the title "The Mesa 3D Graphics Library" in a large, bold font. Below the title is a subtitle: "Open source implementations of OpenGL, OpenGL ES, Vulkan, OpenCL, and more!". A blue button with the text "Read more »" is positioned below the subtitle. At the bottom left of the main content area, it lists the "Current release: 23.3.3" and the "Amber release: 21.3.9". On the right side of the page, there is a decorative graphic of three interlocking gears in purple, red, and green.

Mesa 3D

Home News Getting Started Documentation

The Mesa 3D Graphics Library

Open source implementations of [OpenGL](#), [OpenGL ES](#), [Vulkan](#), [OpenCL](#), and more!

[Read more »](#)

Current release: [23.3.3](#)
Amber release: [21.3.9](#)



Want to help out?

Mesa lives at <https://gitlab.freedesktop.org/mesa/mesa/>

And also <https://mesa3d.org>

We're also in the #dri-devel channel on the OFTC IRC server



Any Questions?



COLLABORA



igalia

Open Source Consultancy

Thanks!



COLLABORA



igalia

Open Source Consultancy