



# Mesa for Mapping Layers

# The 5 Ws

---

Who

---

What

---

When

---

Where

---

Why

---

How

# Who

- 
- Microsoft
    - Me (Jesse Natalie)
    - Bill Kristiansen
    - Angela Jiang
  - Collabora
    - Erik Faye-Lund
    - Daniel Stone
    - Gert Wollny
    - Louis-Francis Ratté-Boulianne
    - Boris Brezillon
    - Elie Tournier

# What

## Two mapping layers

- OpenGL on D3D12
- OpenCL on D3D12

## OpenGLOn12

- Gallium driver in Mesa

## OpenCLOn12

- Compiler stack built out of Mesa
  - Leveraging and extending work done for Clover
- Runtime separate from Mesa
  - Based on D3D12TranslationLayer, core of other D3D12 mapping layers

# When



- Project underway for ~1 year
- Was the reason for Microsoft's presence at XDC last year

# Where



- Development started in secret
  - Private GitLab project hosted by Collabora
- Shifted to full open-source
  - Erik's FreeDesktop.org GitLab Mesa fork
- Upstreaming to Mesa's mainline

# Why

---

- Windows
  - Devices with no Windows drivers for these APIs
    - Qualcomm
    - WARP – Virtual Machines
    - New devices?
  - Enable partners to transition to mapping layers instead of native
    - Not a requirement
    - If desired, they can reduce development costs
- Interop
  - Possibility for simpler / more efficient cross-API interop via mapping layers than across driver stacks
- Debugging
  - Access to D3D12 debugging tools

# Why - continued

---

- WSL
  - Only one vendor usermode driver needed in WSL
- Why Mesa
  - We considered it the only way to have a viable OpenGL implementation
  - Collabora's suggestion to use it for OpenCL as well
    - Definitely the right call



# How

---

- 4 primary components

- NIR → DXIL translator

- DXIL is LLVM-based, but old LLVM – would conflict with other LLVMs
    - Built a custom LLVM bitcode emitter

- Mesa Gallium driver

- Implements Gallium interface, translates to D3D12 APIs
    - Built on the shoulders of Zink

- Windows DXGI/D3D12 WinSys

- Enables more efficient Present than software path

- CLC → DXIL compiler

- OpenCL C → Clang → SPIR → SPIRV-LLVM-Translator → SPIR-V → Mesa SPIRV-to-NIR  
NIR → NIR-to-DXIL → DXIL

# NIR DXIL

## Why not use LLVM?

- DXIL is LLVM 3.7
- Already wanted to use Clang/LLVM for OpenCL, but not 3.7

## LLVM bitcode difficult to deal with

- Variable-sized fields, unaligned data
- DXIL validator imposes additional constraints, such as metadata ordering

## Two-pass approach

- First: Translate NIR to data structures which match DXIL semantics
- Second: Walk data structures and deal with LLVM bitcode emission

## LLVM IR wrapped in container

- Additional chunks of data for validation and quick analysis

# D3D12 Gallium Driver

- 
- More or less self-contained
  - Deals with tracking state, allocating resources, managing command lists
  - Has to emulate several features... examples:
    - Wide points – D3D10+ don't support these
    - Interleaved depth-stencil – D3D12 treats depth-stencil as planar
    - Missing vertex formats – R10G10B10A2 vertex input
    - 8bit index buffer formats
    - Two-sided polygon mode
    - Provoking vertex
    - Combined image+sampler – D3D has these separated
    - Point-sampling of integer textures – D3D only allows loads
      - Needs to deal with normalized coords and border behavior

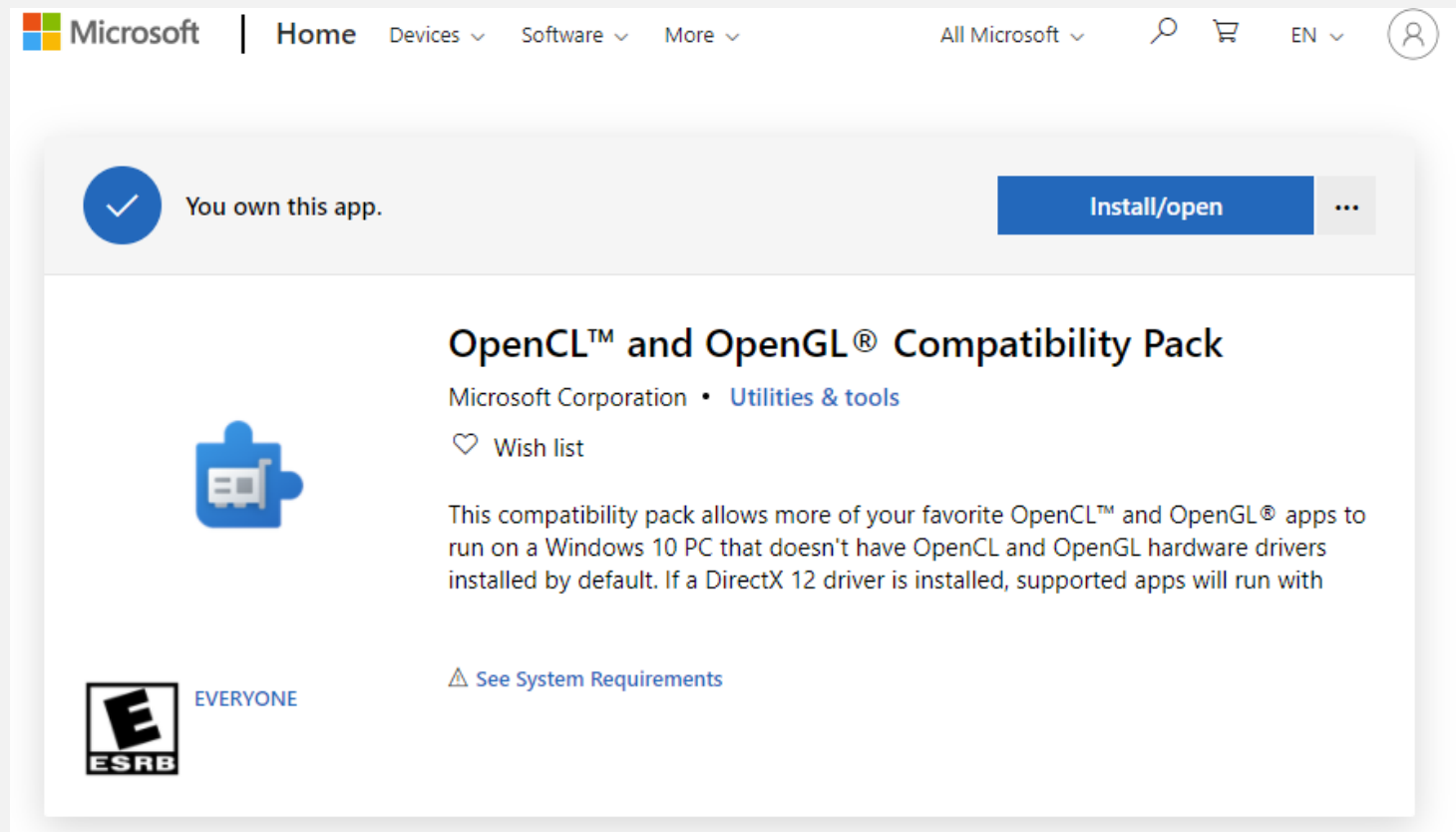
# DXGI/D3D12 WinSys

- 
- On Windows, Present is complicated
  - OpenGL32.dll provides some callbacks to help, but requires being a Windows driver – a mapping layer can't really use them
  - Only other option in Mesa was to use GDI
    - Requires copying from GPU → CPU and waiting, then another CPU → CPU copy via GDI. Really slow
  - DXGI allows queued, efficient Present
  - But... Mesa state tracker likes to own framebuffer allocations. DXGI needs to own its resources.
    - Somewhat messy interface to winsys, but it works and is efficient

# OpenCL Compiler

- 
- Complex compilation pipeline
    - Fortunately, Clover pioneered it
  - Clover NIR support at sub-1.0. Lots of work to get things up to 1.2
    - Crazy float $\leftrightarrow$ integer conversions with all kinds of rounding
    - Support for work item offsets
    - Images – not technically required, but required in practice
    - Complex math – importing from LLVM's libclc
    - More pointer types
  - DXIL-specific shortcomings
    - No pointers... eventually able to use NIR to convert OpenCL pointers into (index, offset) pairs
  - Upstream contributions accelerating Clover support

# How... will customers get it?



The screenshot shows the Microsoft Store interface for the 'OpenCL™ and OpenGL® Compatibility Pack'. At the top, the Microsoft logo is on the left, and navigation links for 'Home', 'Devices', 'Software', and 'More' are in the center. On the right, there are links for 'All Microsoft', a search icon, a shopping cart icon, 'EN', and a user profile icon. Below the navigation bar, a status bar indicates 'You own this app.' with a checkmark icon and an 'Install/open' button. The main content area features the app's icon (a blue puzzle piece with a white document), the title 'OpenCL™ and OpenGL® Compatibility Pack', and the publisher 'Microsoft Corporation • Utilities & tools'. A 'Wish list' link is also present. The description states: 'This compatibility pack allows more of your favorite OpenCL™ and OpenGL® apps to run on a Windows 10 PC that doesn't have OpenCL and OpenGL hardware drivers installed by default. If a DirectX 12 driver is installed, supported apps will run with'. An ESRB rating of 'E' (Everyone) is shown in the bottom left, and a 'See System Requirements' link is in the bottom right.

Microsoft | Home Devices Software More All Microsoft EN

You own this app. Install/open

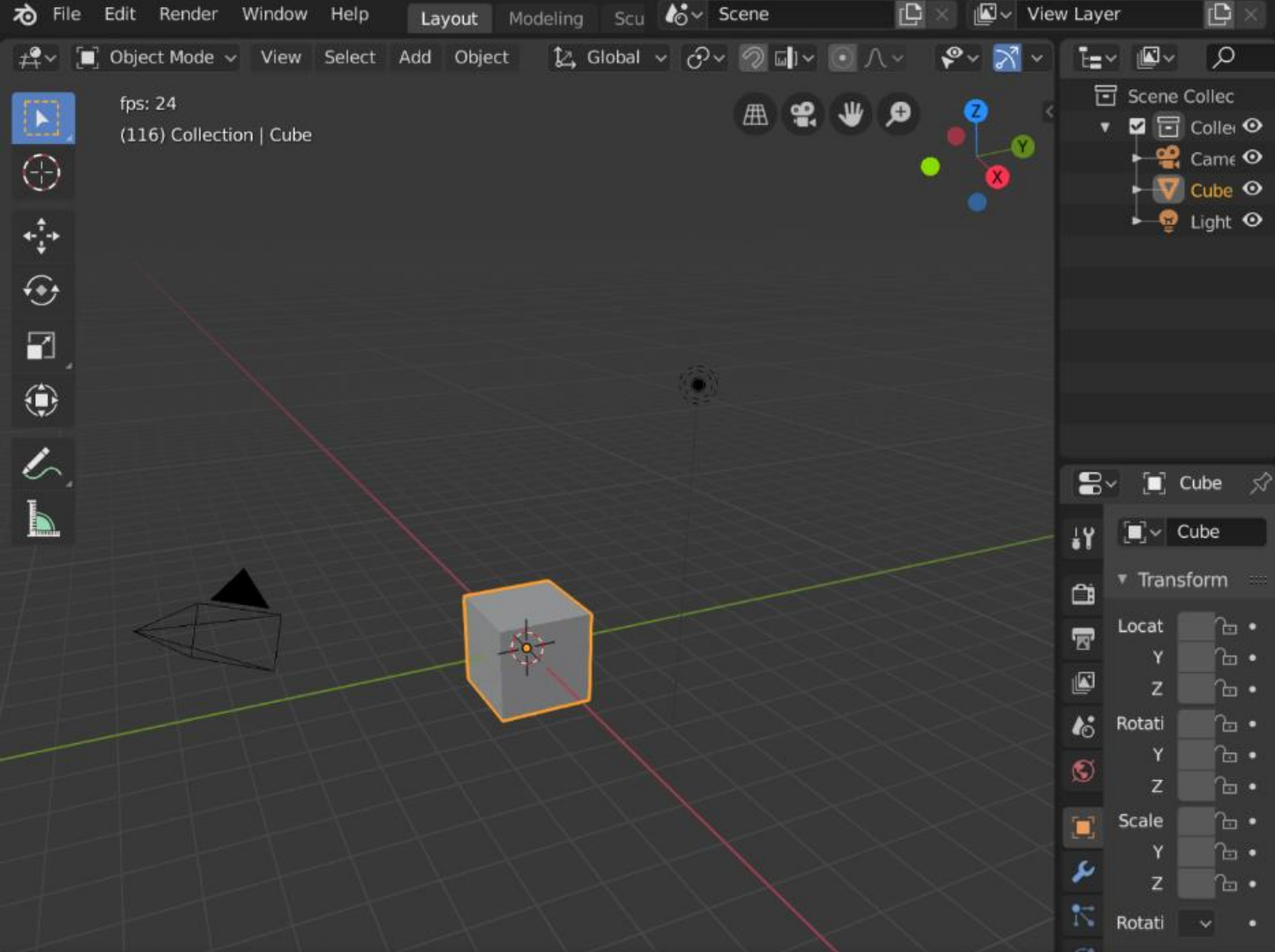
## OpenCL™ and OpenGL® Compatibility Pack

Microsoft Corporation • Utilities & tools

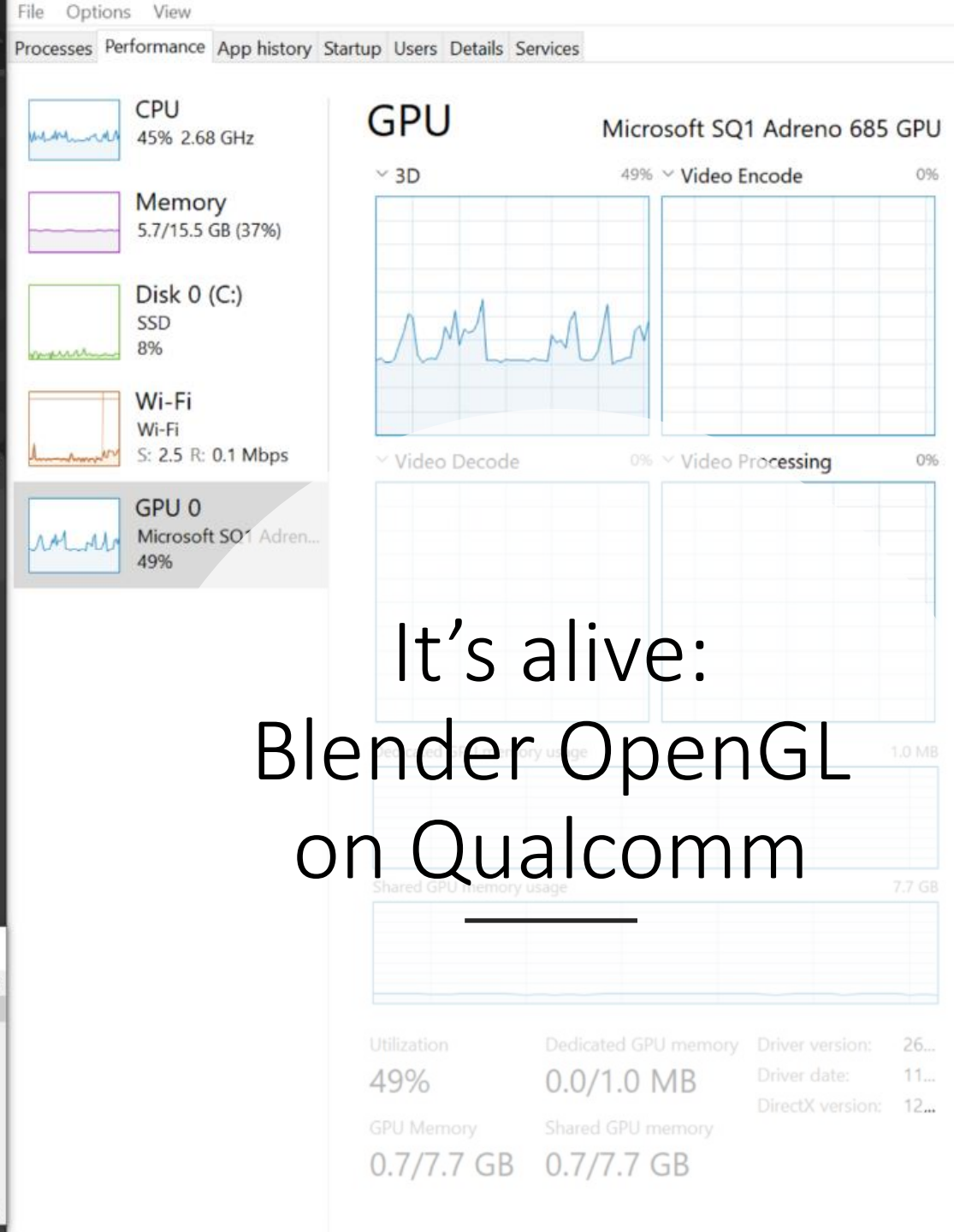
Wish list

This compatibility pack allows more of your favorite OpenCL™ and OpenGL® apps to run on a Windows 10 PC that doesn't have OpenCL and OpenGL hardware drivers installed by default. If a DirectX 12 driver is installed, supported apps will run with

**E** EVERYONE [See System Requirements](#)



```
C:\Users\Admin\Desktop>tlist.exe -m openglon12.dll
C:\Program Files\WindowsApps\Microsoft.D3DMappingLayers_2.2007.7.0_arm64__8wekyb3d8bbwe\x86\OpenGLon12.dll
- 10568 blender.exe      Blender
C:\Users\Admin\Desktop>
```



It's alive:  
Blender OpenGL  
on Qualcomm