Vulkan Portability and MoltenVK

Layering Vulkan Over Metal

Bill Hollings
The Brenwill Workshop Ltd.
October 2021
MoltenVK
Vulkan over Metal
# A Brief History of MoltenVK

<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>2011</td>
<td>Open-source Objective-C framework expansion of Cocos2D for building 3D games on iOS &amp; macOS.</td>
</tr>
<tr>
<td>2015</td>
<td>Proprietary Objective-C implementation of the OpenGL ES 2.0 API layered on top of Metal. Originally developed to support Cocos3D/2D on Metal.</td>
</tr>
<tr>
<td>2016</td>
<td>Originally proprietary C++ implementation of the Vulkan API layered on top of Metal.</td>
</tr>
<tr>
<td>2017</td>
<td>Valve early investment in MoltenVK development, followed by sponsoring the donation of MoltenVK to the Vulkan ecosystem as a Khronos open-source project. Valve continues ongoing funding of MoltenVK development.</td>
</tr>
<tr>
<td>2018</td>
<td>Significant ongoing contributor to MoltenVK development.</td>
</tr>
</tbody>
</table>
MoltenVK Current Status (Oct 2021)

- Support for Vulkan 1.1 feature set and numerous extensions
- Supports macOS, iOS, tvOS, and Mac Catalyst
- Supports Apple and non-Apple GPUs
- Embedded in the macOS Vulkan SDK
- Aggressively iterating through 800,000 Vulkan CTS tests for Vulkan 1.0 core feature set for Vulkan Portability conformance:
  - Passing 167,781
  - Failing 818 (0.49%)
- To accomplish 0 failed CTS tests:
  - Iterative fixes to MoltenVK and SPIRV-Cross.
  - Numerous CTS tests need to be modified to accommodate the feature differences of Vulkan Portability

![Commercial Vulkan Developers](From LunarG 2020 Developer Survey)
Vulkan Apps on macOS and iOS using MoltenVK

“I was initially skeptical about the use of a Vulkan translation layer to Metal, but it turns out MoltenVK enables applications using the real-time NAP Framework to run as well on macOS as they do on Linux and Windows. This is something I’d like to highlight and stress the importance of - without the portability of Vulkan we would have been forced to implement a special Metal rendering backend just for macOS users, something not doable for a relatively small company such as ours. Plus switching to Vulkan dramatically improved render-times for Mac users.”
Coen Klosters, Lead Developer of NAP Framework and Technical Director at Naivi

Production Dota 2 on macOS with up to 50% more perf than Apple’s OpenGL
First iOS Apps using MoltenVK ship through app store
Multiple iOS and macOS apps shipping e.g., Forsaken Remastered
Google Filament PBR Renderer on Mac
RPCS3 PlayStation 3 Emulator on macOS
CodeWeavers and Square Enix ships Final Fantasy XIV on macOS
Crossover Windows compatibility layer

June 2018
September 2018
November 2018
January 2019
June 2019
January 2021
July 2021
Vulkan Portability Initiative
Open-Source Vulkan Implementations
Layering Vulkan on native platform API’s

- MoltenVK
- gfx-portability
- Ashes
- ...

- Metal
- Metal DX11-12
- OpenGL DX11
- ...

This work is licensed under a Creative Commons Attribution 4.0 International License
Open-Source Vulkan Targets

Layering other API's on Vulkan

- DX12
  - vkd3d-proton
  - vkd3d
- DX9-11
  - DXVK
  - WineD3D
- OpenGL
  - Zink
- OpenGL ES
  - Angle
  - GLOVE
- OpenCL
  - clspv
  - clvk
Potential API Layering Stacks

Vulkan as intermediary between game API’s and platform API’s.

1. DX9-11
2. DXVK
3. WineD3D
4. Vulkan
5. MoltenVK
6. Metal
## Open-Source Layering Projects

Fighting Platform Fragmentation

<table>
<thead>
<tr>
<th>Layers Over</th>
<th>Vulkan</th>
<th>OpenGL</th>
<th>OpenCL</th>
<th>OpenGL ES</th>
<th>DX12</th>
<th>DX9-11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vulkan</td>
<td>Zink</td>
<td>clspv</td>
<td>clvk</td>
<td>Angle</td>
<td>vkd3d-Proton</td>
<td>DXVK</td>
</tr>
<tr>
<td>OpenGL</td>
<td>gfx-portability</td>
<td>Ashes</td>
<td>Angle</td>
<td>vkd3d-Proton</td>
<td>DXVK</td>
<td>WineD3D</td>
</tr>
<tr>
<td>DX12</td>
<td>gfx-portability</td>
<td>Microsoft ‘GLOn12’</td>
<td>Microsoft ‘CLOn12’</td>
<td>Microsoft D3D11On12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DX11</td>
<td>gfx-portability</td>
<td>Ashes</td>
<td>Angle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metal</td>
<td>MoltenVK</td>
<td>gfx-portability</td>
<td>clspv + SPIRV-Cross?</td>
<td>MoltenGL</td>
<td>Angle</td>
<td></td>
</tr>
</tbody>
</table>

**Vulkan as porting target for multiple APIs**

**ROWS:** Bring more APIs to Platforms

**COLUMNS:** Making APIs available across platforms

Vulkan functionality available everywhere

This work is licensed under a Creative Commons Attribution 4.0 International License
Vulkan Portability

Multiple Implementations of Layered Vulkan Functionality
Enable Vulkan applications on diverse platforms and APIs

Khronos Portability Extension
Portable queries for what functionality is available through Vulkan layering

Extended Vulkan Conformance Tests
Gracefully handle partial functionality to enable Vulkan Portability conformance - what is present must work!

Enhanced Vulkan Layers and Tools
DevSim/Validation Layers flag or simulate queries for features that are not present

Fighting fragmentation by enabling Vulkan applications on platforms without native drivers through layered implementations of Vulkan functionality over Metal, DX12 and other APIs

https://github.com/KhronosGroup/Vulkan-Portability

© The Khronos® Group Inc. 2021 - Page 11

This work is licensed under a Creative Commons Attribution 4.0 International License
Vulkan Portability Timelines

- **MoltenVK released in open source for macOS and iOS**
- **Vulkan SDK on macOS**
- **Vulkan CTS ported to macOS**
- **gfx-rs DX11 backend in alpha - joins DX12 and Metal backends**
- **Vulkan Portability Provisional Extension released and shipped in MoltenVK and gfx-rs**
- **MoltenVK 1.1 ships with Vulkan 1.1 features, 48 Vulkan extensions, Metal 3.0, Apple Silicon, Mac Catalyst and tvOS**
- **Vulkan SDK Validation Layers support Portability Extension. Vulkan Configurator manages DevSim and config files**
- **Hardware capability viewer feeds data to gpuinfo.org ported to iOS and made available on Apple Store. DevSim layer ported to macOS.**
- **Vulkan Portability Final Extension release target**
- **MoltenVK conformant with Vulkan Portability**

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDC</td>
<td>2018</td>
</tr>
<tr>
<td>August</td>
<td>2018</td>
</tr>
<tr>
<td>September</td>
<td>2020</td>
</tr>
<tr>
<td>Fall</td>
<td>2020</td>
</tr>
<tr>
<td>January</td>
<td>2021</td>
</tr>
<tr>
<td>February</td>
<td>2021</td>
</tr>
</tbody>
</table>

*This work is licensed under a Creative Commons Attribution 4.0 International License*
VK_KHR_portability_subset Extension

- Identifies differences between a non-conformant layered Vulkan implementation and a fully-conformant native Vulkan implementation.

- Provides the ability for implementations to mark otherwise-required capabilities as unsupported.

- Establishes additional properties and limits that the application should adhere to in order to guarantee portable behaviour.

typedef struct VkPhysicalDevicePortabilitySubsetFeaturesKHR {
    VkStructureType    sType;
    void*              pNext;
    VkBool32           constantAlphaColorBlendFactors;
    VkBool32           events;
    VkBool32           imageViewFormatReinterpretation;
    VkBool32           imageViewFormatSwizzle;
    VkBool32           imageView2DOn3DImage;
    VkBool32           multisampleArrayImage;
    VkBool32           mutableComparisonSamplers;
    VkBool32           pointPolygons;
    VkBool32           samplerMipLodBias;
    VkBool32           separateStencilMaskRef;
    VkBool32           shaderSampleRateInterpolationFunctions;
    VkBool32           tessellationIsolines;
    VkBool32           tessellationPointMode;
    VkBool32           triangleFans;
    VkBool32           vertexAttributeAccessBeyondStride;
} VkPhysicalDevicePortabilitySubsetFeaturesKHR;
The Device Simulation Layer and the VK_KHR_portability_subset Extension

• **VK_KHR_portability_subset Extension**
  - Provides new queryable structures for the pNext changes of:
    - `VkPhysicalDeviceProperties2 struct`
    - `VkPhysicalDeviceFeatures2 struct`
  - New structures inform application what features are not supported by the portability subset implementation.

• **Device Simulation Layer**
  - Devsim modifies query results to enable a test environment for applications
  - One Devsim feature allows emulation of the `VK_KHR_portability_subset` extension on fully-conformant implementation.
    - This allows developers to test applications for portability subset compliance on fully-conformant implementations.
  - Included in the Vulkan SDK (Windows, Linux, macOS)
Enabling Devsim and Validation from vkconfig
Vulkan Portability Development on Apple

- SPIRV-Cross: Open-source disassembler, converts SPIR-V shaders to Metal Shading Language
- Vulkan SDK
  - Running on macOS
    - Vulkan Configurator
    - DevSim config file selection
    - DevSim Layer: simulates queries for selected target system
    - Validation Layer: flags incorrect usage of available functionality
- DevSim JSON Config files
  - Defines available functionality on a target device
  - Pre-populated subset in SDK
- Vulkan to Metal API Mapping Layer
  - Vulkan Portability Extension
  - Portable run-time queries for available functionality on the current system
- macOS / iOS / tvOS
- Hardware Capability Viewer
  - (for macOS and iOS on app store)
  - Run-time application now includes interrogation of Vulkan Portability Extension to discover available features of layered implementations
- Enhanced Vulkan SDK, Layers and Tools
  - Streamlines application development for Apple devices. Universal Binaries supported
- Database Export
  - gpuinfo.org
  - Public Database
  - Available Vulkan functionality per GPU/Driver/Layer/OS
  - List of entries containing Vulkan Portability data
- Database Upload

This work is licensed under a Creative Commons Attribution 4.0 International License