Overview

- Why would you want to have a wrapper for a graphics library?
- X.Org
- WINE
- Purpose
- Performance
- Alternatives?
Why?

- DirectX is unique to Windows
- Metal is also unique to MacOS
  - Even worse because of the extremely closed nature of MacOS
  - If you want Vulkan support you have to go with another wrapper library - MoltenVK!
- Why do they do this?!
  - Platform lock users, prevent them freedom of OS choice because their software is MacOS only

X.Org Server

- Previous to a graphics environment, everything UNIX-based had a simple bitmap console.
  - The screen is represented by a grid of characters, that each have a constant pixel dimension and control which pixels in their space are lit up
  - This is why Linux's terminal graphics drivers run on EVERYTHING - it's extremely basic and largely the same code as it was decades ago
- Then the light was introduced with graphics environments and changed the game forever.
  - Somewhere along the way X.Org was established to create a universal set of standards for graphics on Linux, and is still largely used today.
  - Hatred of X.Org grew into Wayland, but that's beyond the scope of this basic overview
How does a GPU fit into this pipeline?

1. Windowed applications: OS < -- > GPU < -- > X.Org < -- > Compositor < -- > Application
2. Windowed GPU-accelerated applications: OS < -- > GPU < -- > X.Org < -- > Compositor < -- > GPU < -- > Application
3. Fullscreen applications: OS < -- > GPU < -- > X.Org < -- > Application

What is WINE?

- **Wine Is Not an Emulator** - Recursive definition
- Compatibility layer to support the various W32 API's
- It DOES NOT EMULATE support!
- Each library has been written to add identical(roughly…) support for every required function - A painstakingly difficult task
- Does not guarantee an application will actually work…
  - [https://appdb.winehq.org/appbrowse.php?CatId=2](https://appdb.winehq.org/appbrowse.php?CatId=2)
Purpose of DXVK

- Largely to add another backend graphics library to WINE
- Lower level access and reduced overhead results in better performance
- Support the DirectX 11_0 featureset
- Run applications locked to Windows due to W32 and DirectX platform locks

Performance

- Fallout 4: https://www.youtube.com/watch?v=NWggyxy4Fno
- Grand Theft Auto V: https://www.youtube.com/watch?v=YMhB9gz8RnI
- The Witcher 3: https://www.youtube.com/watch?v= fZO4BJ87Bw
- Tomb Raider: https://youtu.be/9NhsTdQTI3U?t=124
- Shockingly better than existing WINE and DirectX 11 for a large amount of applications
- Still not perfect and has missing or unimplemented features
Missing Features

- **Missing**
  - Predication - Culling on the GPU instead of CPU
  - Stream output - Takes data from a geometry shader and forwards access
- **Incomplete**
  - Tessellation - Creates additional surfaces on geometric surfaces
  - Geometry shaders - Modify vertex information
  - Unordered Access View operations - Similar to render targets

How does it work?

- Same way WINE does, but due to the low level nature of Vulkan it works exceptionally well
- Each DirectX function is written to use the respective Vulkan calls
  - Not always as easy and mapping one function to another!
  - DirectX is similar to OpenGL in that it is NOT stateless
- DirectX compiled shaders are translated bytecode into SPIR-V
  - This is the majority of the "secret sauce" that makes this work so well
- Launch the application with a custom compiled WINE and you’re good to go!
Alternatives

- Run Windows…
- Virtualize Windows with GPU Passthrough
- Ignore and boycott applications that do not have Linux support

- All completely logical things to do...

Nier: Automata as of version 0.20, on an AMD RX 480
Source: [https://github.com/doitsujin/dxvk/wiki](https://github.com/doitsujin/dxvk/wiki)
Unigine Heaven as of version 0.31, on an AMD RX 480
Source: https://github.com/doitsujin/dxvk/wiki

Sources
- MSDN for DirectX
- DXVK Github: https://github.com/doitsujin/dxvk