An introduction to Wine Direct3D architecture

Henri Verbeet

WineConf 2018

About me

- Software developer working on Wine's Direct3D implementation for CodeWeavers
- First learned about Wine in 1997
- Wine contributor since late 2005
- Joined CodeWeavers in late 2008

Introduction

- Fairly brief, inaccurate
- Idea is to provide context rather than details
- I'll assume some familiarity with 3D graphics
  e.g. won't be explaining what a texture or a shader is

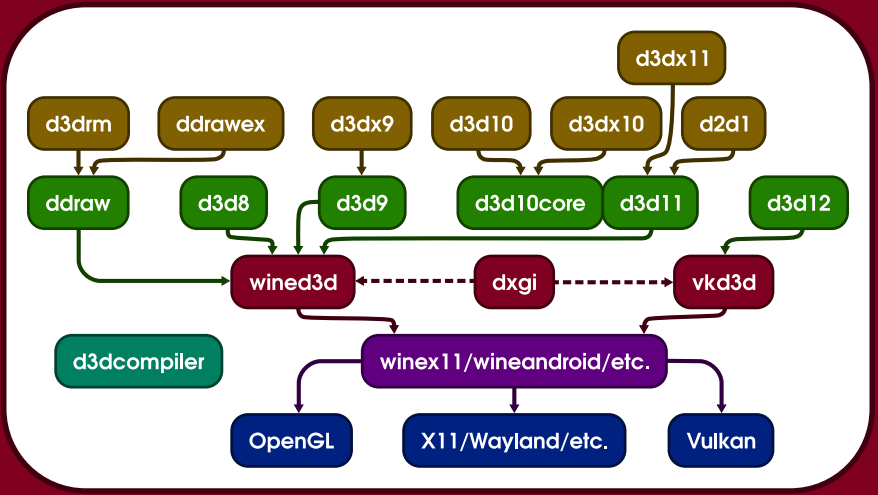
History

- Initial DirectDraw implementation by Marcus Meissner in late 1997
- TransGaming WineX/Cedega fork in 2001
- Switch to the GNU LGPL in early 2002
- Initial wined3d commit in late 2003
- DirectDraw and Direct3D 8 on top of wined3d in 2006

Key points

- We've been around for a while
  - And we'd like to still be here in a few years, so we tend to take the long view
  - Can be frustrating for less experienced developers
- We have experience with both lax and copyleft licenses
  - And there's a clear winner
- We support a broad range of GPUs
  - R300 and GeForce 4 are getting pretty rare
  - R600 and GeForce 8 not so much

How things fit together



Wined3d components

- State collection and application
- Shader compiler
- Blitter
- Command stream
- Screen/window handling
- Adapter enumeration and capability reporting

State collection and application

- Direct3D state is similar, but not quite the same as OpenGL state
  - E.g. GL_MODELVIEW -> D3DTS_VIEW + D3DTS_WORLD
- Different implementations based on GL extensions.
  - E.g. ARB_clip_control for pixel origin
- `struct wined3d_state`
- `struct wined3d_context.state_table`
- `wined3d_device_invalidate_state()`
- `context_apply_draw_state()`

Shader compiler

- Not the HLSL compiler; that one lives in d3dcompiler
- Compile D3D bytecode to:
  - GLSL; GLSL as a language has its issues, but is fairly easy to understand and get started with
  - SPIR-V; All the advantages of a binary format, and all the disadvantages
  - ARB fragment/vertex programs
- Forked for libvkd3d-shader
- `struct wined3d_shader_frontend`, `shader_sm1.c`, `shader_sm4.c`
- `struct wined3d_shader_backend_ops`, `shader_glsl_select()`

Blitter

- Why does this even exist?
  - FBO blits would perhaps be the obvious choice, but didn't exist for most of Wine's history
  - FBO blits can't do everything. E.g. raw/typeless blits, P8 blits
  - We want to do blits between CPU resources on the CPU
- Blitters are tried in order
- Perhaps not ideal to have a single blitter interface that can do everything; Inherited from DirectDraw
- `struct wined3d_blitter_ops`, `blitter_blit()`, `blitter_clear()`

Command stream

- Applications can call Direct3D from multiple threads, and expect deterministic behaviour
- Multi-threaded OpenGL is a bit of a pain;
  E.g. occlusion queries aren't shared between contexts.
- We used to have `StrictDrawOrdering`; Very inefficient
- `csmt` serialises Direct3D operations into a single thread
- As a happy coincidence, there are performance advantages as well
- `wined3d_cs_emit_*()`
- `wined3d_cs_exec_*()`

Drivers

- Linux NVIDIA
  - One of the first vendors to support accelerated OpenGL on Linux
  - For a long time the only realistic option
  - These days the only desktop vendor that doesn't provide Free Software drivers
  - Nouveau is incredible, but faces challenges
- Linux Intel
  - 'Official' Free Software driver since around 2007
  - Initial 3D hardware fairly weak
  - Current 3D hardware much more powerful
  - Generally well supported

Drivers

- Linux AMD/ATI
  - Traditionally supported by fglrx/Catalyst
  - Since around 2009 active support for the Free Software driver
  - These days generally well supported
- Android
  - Lots of proprietary drivers
  - Generally more challenging than regular Linux

Drivers

- MacOS
  - Apple is pretty much just hostile to Free Software
  - Generally poor OpenGL support for many years; now simply deprecated
  - Vulkan only through MoltenVK
  - Metal is a proprietary 64-bit only API, and only available to Objective-C and Swift

Direct3D 12

- Implemented on top of vkd3d
- Somewhat of a testing ground for new ideas
- May end up growing support for Direct3D 11 and before
- But Vulkan isn't necessarily a great fit for Direct3D 11 and before
  - Pipeline object creation is expensive
  - Only supported on fairly recent GPUs
  - If Direct3D 12 and Vulkan are really the future, does it really make sense to invest effort in Direct3D 11 on top of Vulkan?

Questions?