`vkd3d-shader` and the HLSL compiler

Giovanni Mascellani
gmascellani@codeweavers.com

CodeWeavers, Debian

6 October 2022

WineConf 2022, Minneapolis

Who am I?

- Mathematician in a previous life (but I still feel like one), always passionate about computers and programming
- Love free software (contributing to Debian since 2008, a bit less lately)
- Love making things work in “unexpected” ways: Wine, emulators, …
- Met people from Wine at FOSDEM 2022
- Working with CodeWeavers since 2020
- Spent some time on the HLSL compiler
- You probably know what I’ll be talking about better than me! ;-)

vkd3d

- Implementation of Direct3D 12 on top of Vulkan
- Developed (mostly) by CodeWeavers for Wine
- Wine's `d3d12.dll` is a thin layer on top of `vkd3d` (`dxgi.dll` and `d3dcompiler.dll` also pick from `vkd3d{,-shader}`)
- But it's independent of Wine: you can use it for porting your application (demo applications provided)
- LGPL-2.1+

vkd3d components

- vkd3d-shader: compiles shaders between different formats
- vkd3d: translates D3D12 calls to Vulkan
- vkd3d-utils: source porting utilities (not used by Wine)
- Each of those depends on the previous ones

vkd3d-shader

- SM1: Microsoft bytecode format, shader model 1-3
- SM4: Microsoft TPF format in a DXBC container, shader model 4-5
- SM6: Microsoft DXIL format in a DXBC container, shader model 6
- HLSL: Microsoft source format
- SPIR-V: Vulkan binary format
- GLSL: OpenGL/Vulkan source format
- (MSL: Apple source format)

- SM1, SM4 → text
- SM4 → SPIR-V
- HLSL → SM1, SM4
- (ongoing: SM1 → SPIR-V)
- (ongoing: SM6 → SPIR-V)
- (Wine: SM1, SM4 → GLSL)
- (Wine: text → SM1)

vkd3d-compiler

```
$ ./vkd3d-compiler -x hlsl -b dxbc-tpf -p vs_5_0 -e vs_main -o gears_vs.dxbc gears.hlsl
$ ./vkd3d-compiler -x dxbc-tpf -b d3d-asm gears_vs.dxbc
$ ./vkd3d-compiler -x dxbc-tpf -b spirv-binary -o gears_vs.spirv gears_vs.dxbc
$ spirv-dis gears_vs.spirv
```

`VKD3D_SHADER_DEBUG=trace` to see some internals!

vkd3d_shader_compile()

```
static const char triangle_hlsl[] = "...";

struct vkd3d_shader_hlsl_source_info hlsl_info = { 0 };
struct vkd3d_shader_compile_info info = { 0 };
struct vkd3d_shader_code vs;
char *messages;
int res;

hlsl_info.type = VKD3D_SHADER_STRUCTURE_TYPE_HLSL_SOURCE_INFO;
hlsl_info.entry_point = "vs_main";
hlsl_info.profile = "vs_5_0";
```

vkd3d_shader_compile()

```
info.type = VKD3D_SHADER_STRUCTURE_TYPE_COMPILE_INFO;
info.next = &hlsl_info;
info.source.code = triangle_hlsl;
info.source.size = strlen(triangle_hlsl);
info.source_type = VKD3D_SHADER_SOURCE_HLSL;
info.target_type = VKD3D_SHADER_TARGET_DXBC_TPF;

res = vkd3d_shader_compile(&info, &vs, &messages);
assert(res == VKD3D_OK);
```

Using vkd3d-utils

```
static const char triangle_hlsl[] = "...";


ID3DBlob *vs;
HRESULT hr;


hr = D3DCompile(triangle_hlsl, strlen(triangle_hlsl),
NULL, NULL, NULL, "vs_main", "vs_5_0", 0, 0, &vs, NULL);
assert(SUCCEEDED(hr));
```

HLSL: what’s (mostly) there?

- Parsing and SM4 code generation (SM1 is weaker)
- Fundamental types (scalars, vectors and matrices), arrays, structs
- Basic control flow (`if`, `for`, `while`)
- Most arithmetic and logic operators, type conversions, vector swizzling
- Complex initializers
- A fair number of intrinsics, including matrix multiplication
- Textures and samplers, some related methods
- Preprocessor
- Dead code elimination, copy propagation, constant folding

HLSL: what's missing?

- Function calls (i.e., function inlining)
- "Advanced" flow control (`break`, `continue`, `return`)
- Many other intrinsics and methods
- Function overloading
- A lot of missing details and corner cases
- Vectorization
- Some of these are already in the Proton branch
- Usually if it succeeds, then it is correct

Code overview

- `vkd3d_shader_main.c`: entry points, general helpers
- `preproc.l`, `preproc.y`: preprocessor
- `hlsl.l`, `hlsl.y`: HLSL parser, generates a program in a custom internal IR
- `hlsl_codegen.c`: runs a number of lowering and optimization passes (`hlsl_emit_bytecode()`)
- `hlsl_sm{1,4}.c`: code generation (`hlsl_sm{1,4}_write()`)
- `hlsl.c`: HLSL entry point (`hlsl_compile_shader()`) and helpers (mostly for handling types and IR)

Sources

- Upstream: https://gitlab.winehq.org/wine/vkd3d/
- Proton branch with more HLSL features: https://github.com/ValveSoftware/vkd3d/tree/proton_7.0
- Preliminary DXIL → SPIR-V patches: https://gitlab.winehq.org/cmccarthy/vkd3d/-/tree/sm6

`vkd3d-shader` and the HLSL compiler

Giovanni Mascellani
gmascellani@codeweavers.com

CodeWeavers, Debian

6 October 2022

WineConf 2022, Minneapolis