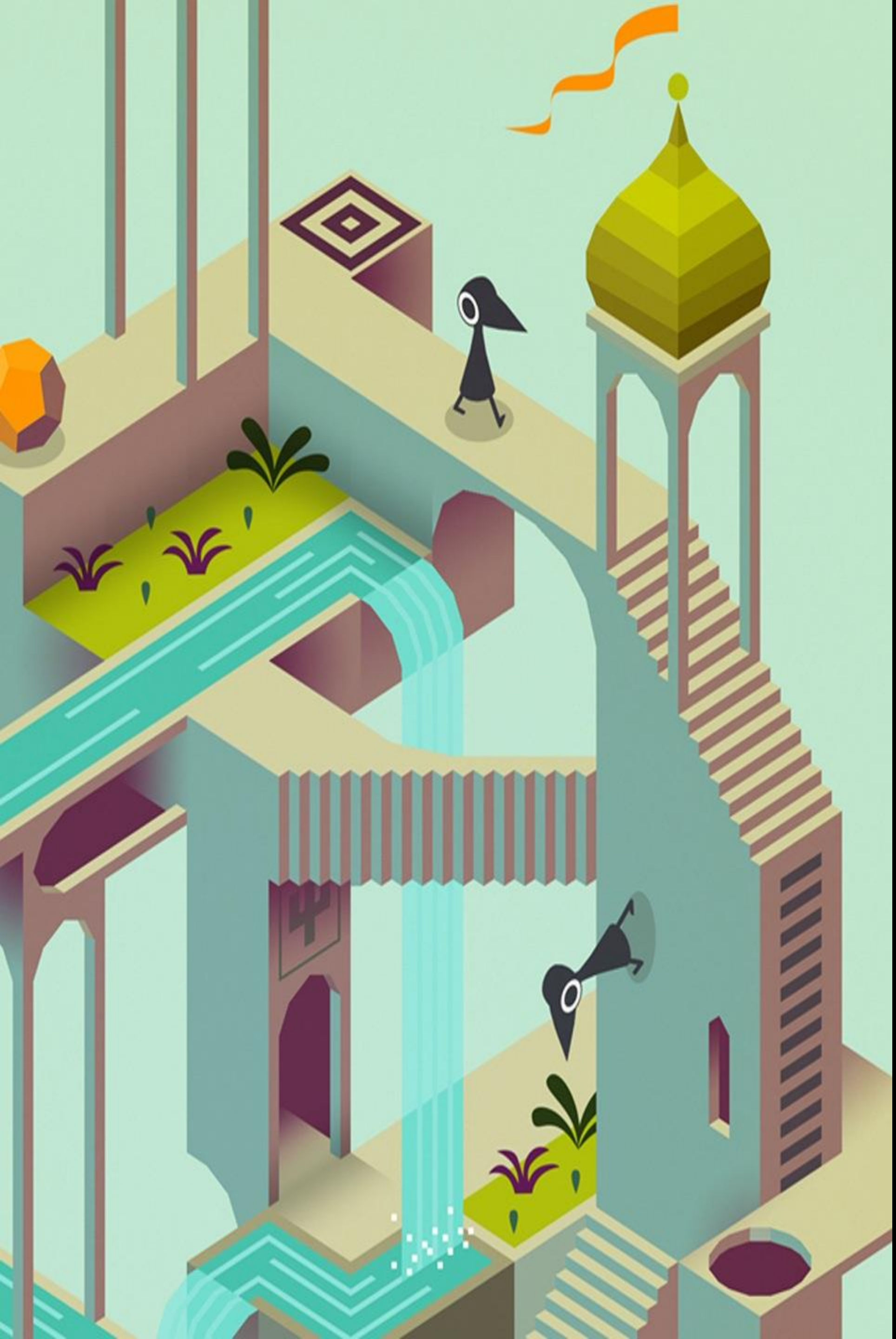
Prospects for a more robust, simpler and more efficient shader cross-compilation pipeline in Unity with SPIR-V

2015/04/14 - Christophe Riccio, OpenGL

Democratizing games development



Monument Valley by ustwo

- iOS / Android
- Puzzle game
- Multidisciplinary design agency

Unity benefits:

- Ease-of-use for the whole team
- Easy cross-platform production
- Custom editor scripts
- Unity Profiler

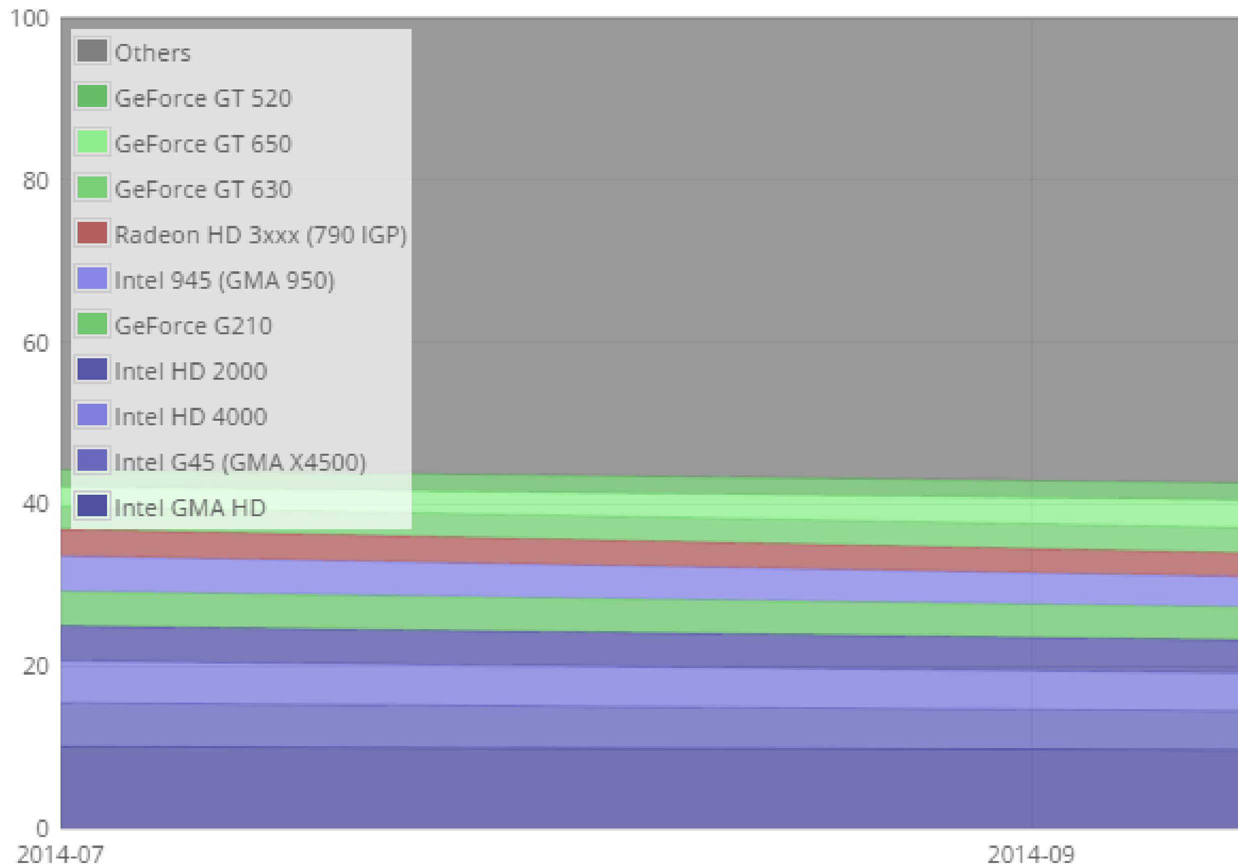
unity

Build once deploy anywhere

Standalone player GPU stats

Top on 2014-12:

- Intel GMA HD: **9.6%**
- Intel G45 (GMA X4500): **4.7%**
- Intel HD 4000: **4.6%**
- Intel HD 2000: **4.1%**
- GeForce G210: **4.0%**
- Intel 945 (GMA 950): **3.6%**
- GeForce GT 650: **3.6%**
- GeForce GT 630: **3.2%**
- Radeon HD 3xxx (790 IGP): **2.8%**
- GeForce GTX 750: **2.5%**
- Others (click to show): 57.3%
  - GRID K520: **2.2%**
  - GeForce GT 520: **2.1%**
  - Intel HD 3000: **1.9%**
  - Radeon HD 5400: **1.8%**
  - GeForce GTS 450: **1.6%**
  - GeForce 7000/7100: **1.5%**
  - GeForce GTX 550: **1.5%**
  - GeForce GTX 660: **1.4%**



Graphic APIs

Platforms have different APIs with different shading language:

- OpenGL / OpenGL ES / WebGL (GLSL)
- Direct3D 9 / 11 / 12 (HLSL)
- Metal (Metal SL)
- Basically an API per consoles (Binary and things shaders)

No one wants to write their shaders even twice => cross compilation

Drivers fragmentation

April 2015:

- AMD expose OpenGL 4.4 support on all GPUs since Evergreen (2009)
- NVIDIA expose OpenGL 4.5 support on all GPUs since Fermi (2010)
- Intel expose OpenGL 4.3 support on Haswell (2013)
- Intel expose OpenGL 4.2 support but image load store on Ivy Bridge (2012)
- Intel expose OpenGL 3.1 support on Sandy Bridge (2011)
- Apple expose OpenGL 4.1 support for all

GLSL case of complexity

- With current GLSL:
  - 4 ES versions !
  - 8 core profile versions !!
  - 12 compatibility profile versions !!!

⇒ Cross compilation performance problems: Too many shaders variations, too many shader outputs

Performance issue

eg: Unity graphics tests:

- ~15 minutes to build Unity
- ~30 minutes to build the shaders
- ~2 minutes to run the tests

And we are building a strict minimum amount of shaders, a lot less shader variants that we would actually want.

Scale of the problem to solve

- Lot of different platforms
- Lot of different APIs including multiple versions per API
- Lot of hardware and drivers to support

=> Cross platform support is a very complex problem!

Shader cross compiling in Unity

Unity developers use:

- The metaHLSL language is cross compiled to all platforms
  1 language -> N platforms
- GLSL snippets: Only for [Open/Web]GL (ES) platforms
  Because cross compilation is too hard already!

unity

Cross compilation approaches

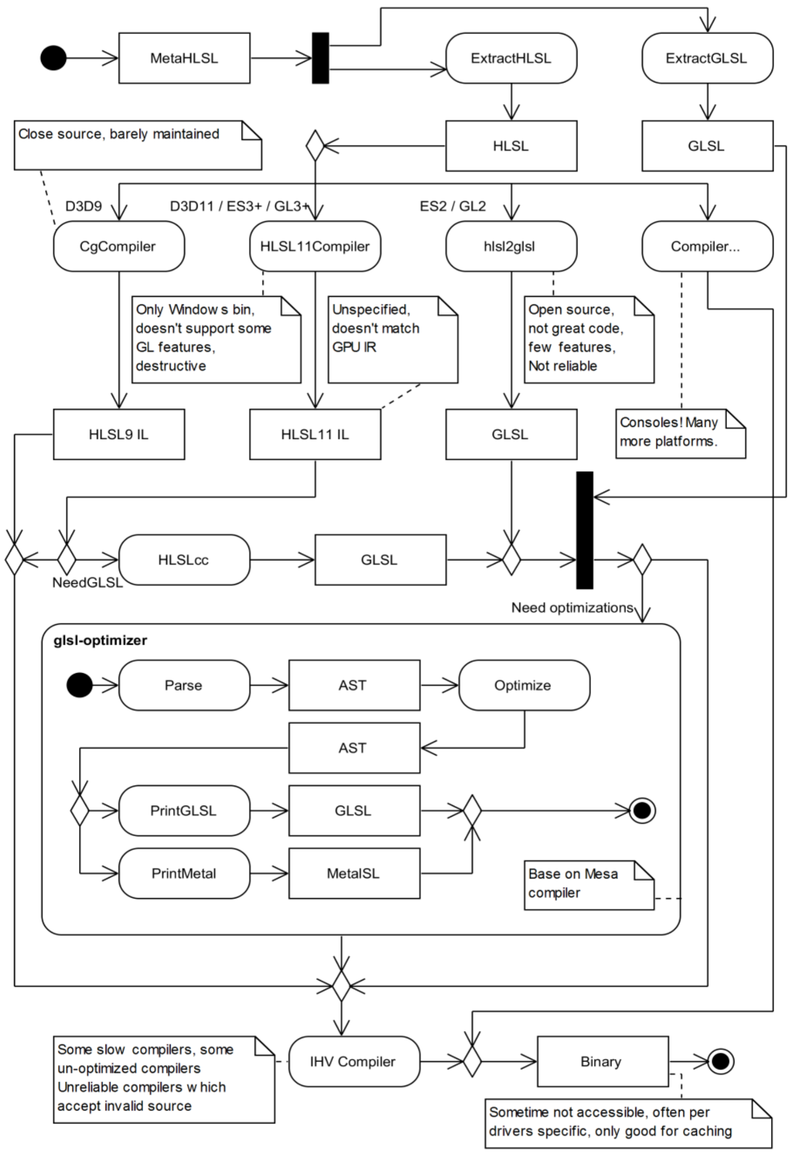
- Macros: Wrap shader languages differences into macros
- Creating our own language with N back-ends (visual editor)
- Using a meta HLSL with source level translation
- Using a meta HLSL with IL level translation

Meta-HLSL with IL level translation

Based on HLSL IL using HLSLcc:

- Close source D3D11 compiler: Only Windows executable! %^*)&%
- Unspecified IL
- Perform excessive optimization losing information
- Legacy vec4 design: over allocation of GPU registers
- Bound to D3D11 features (gl_DrawID, pixel local storage, fb fetch, etc, all missing)

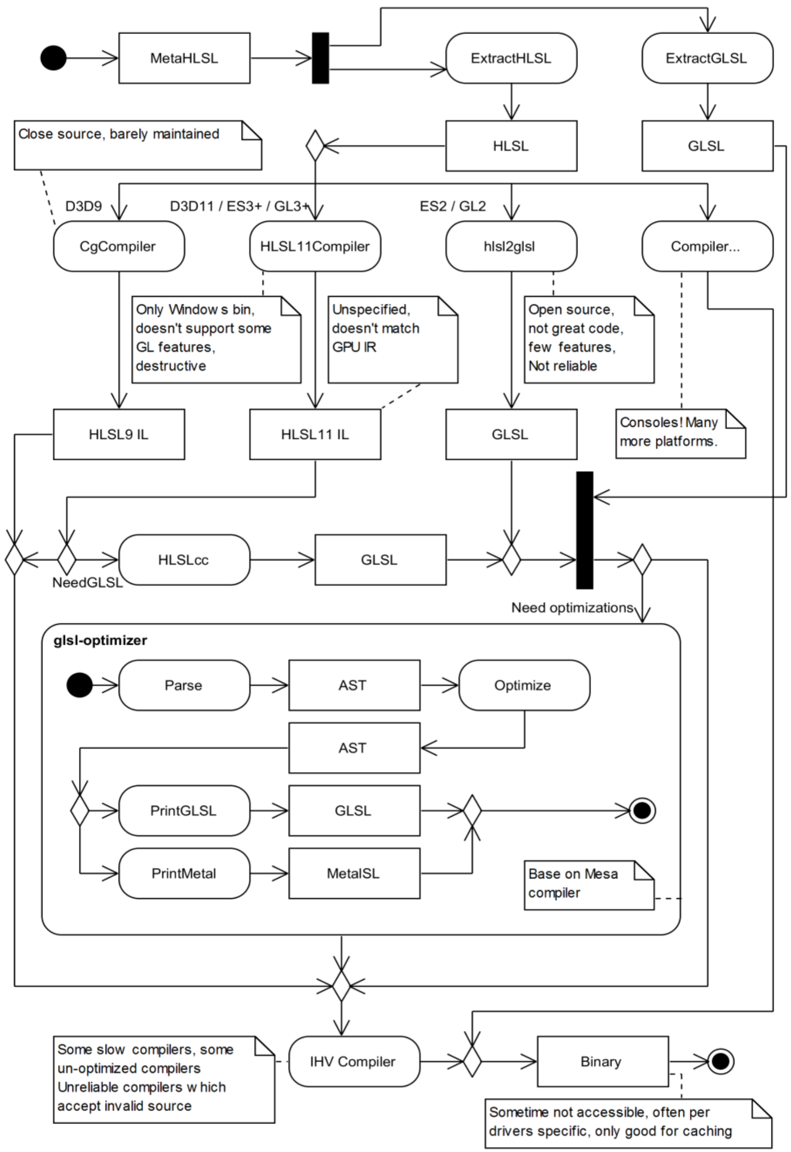
Unity 5.x shader pipeline



- D3D9: Cg compiler
- D3D11: HLSL11 compiler
- GL2/ES2: hlsl2glsl
- GL4/ES3: HLSL11 compiler + HLSLcc
- Metal: HLSL => GLSL => Metal
- Consoles: Their own things

unity

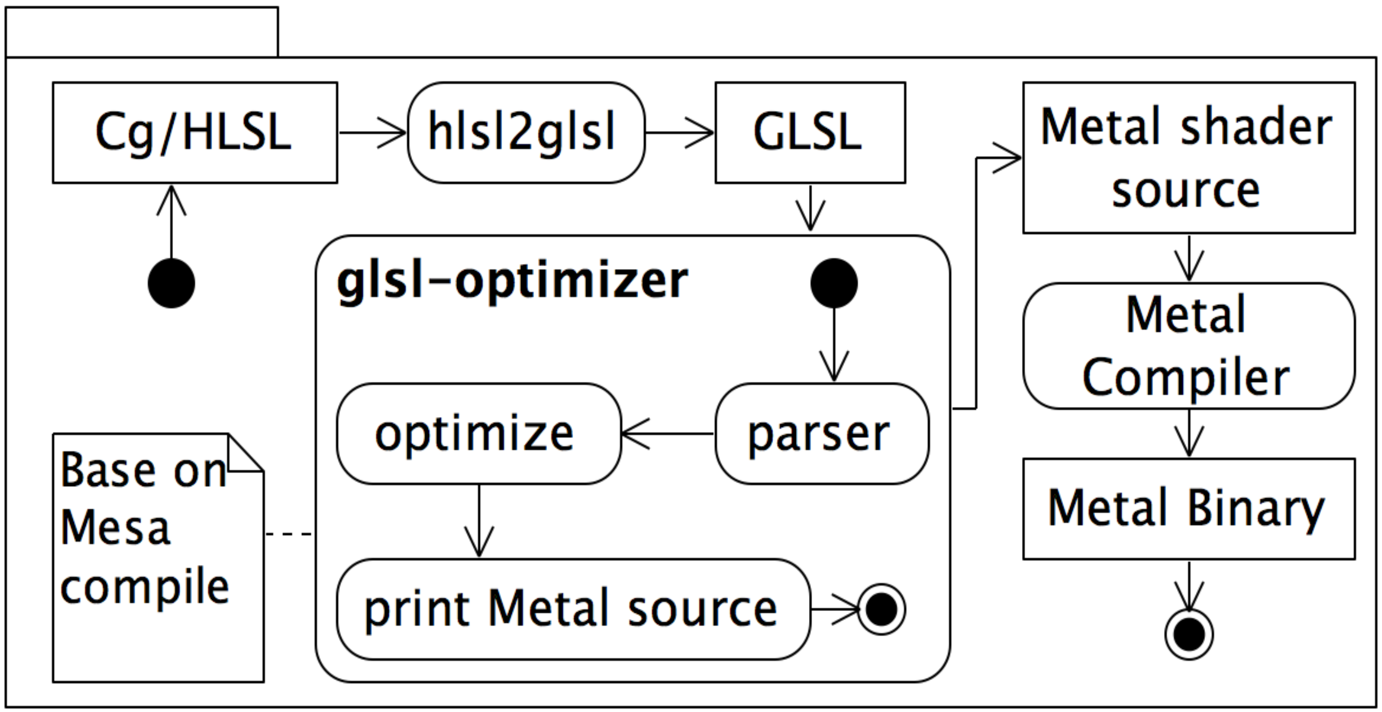
Unity 5.x shader pipeline



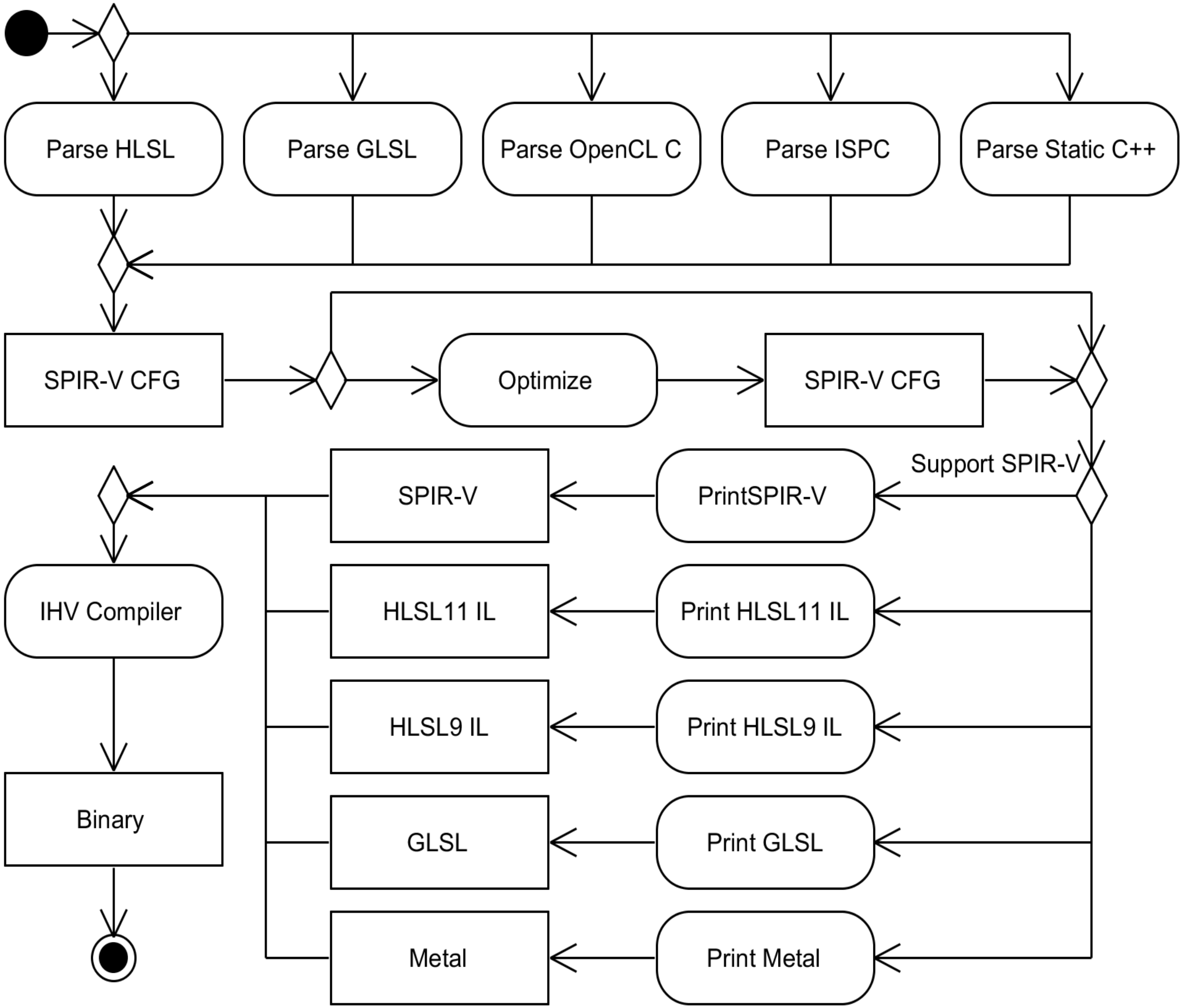
- D3D9: Cg compiler
- D3D11: HLSL11 compiler
- GL2/ES2: hlsl2glsl
- GL4/ES3: HLSL11 compiler + HLSLcc
- Metal: HLSL => GLSL => Metal
- Consoles: Their own things

AAAAAAAAAAAAAAAAAAAAhhhhh!

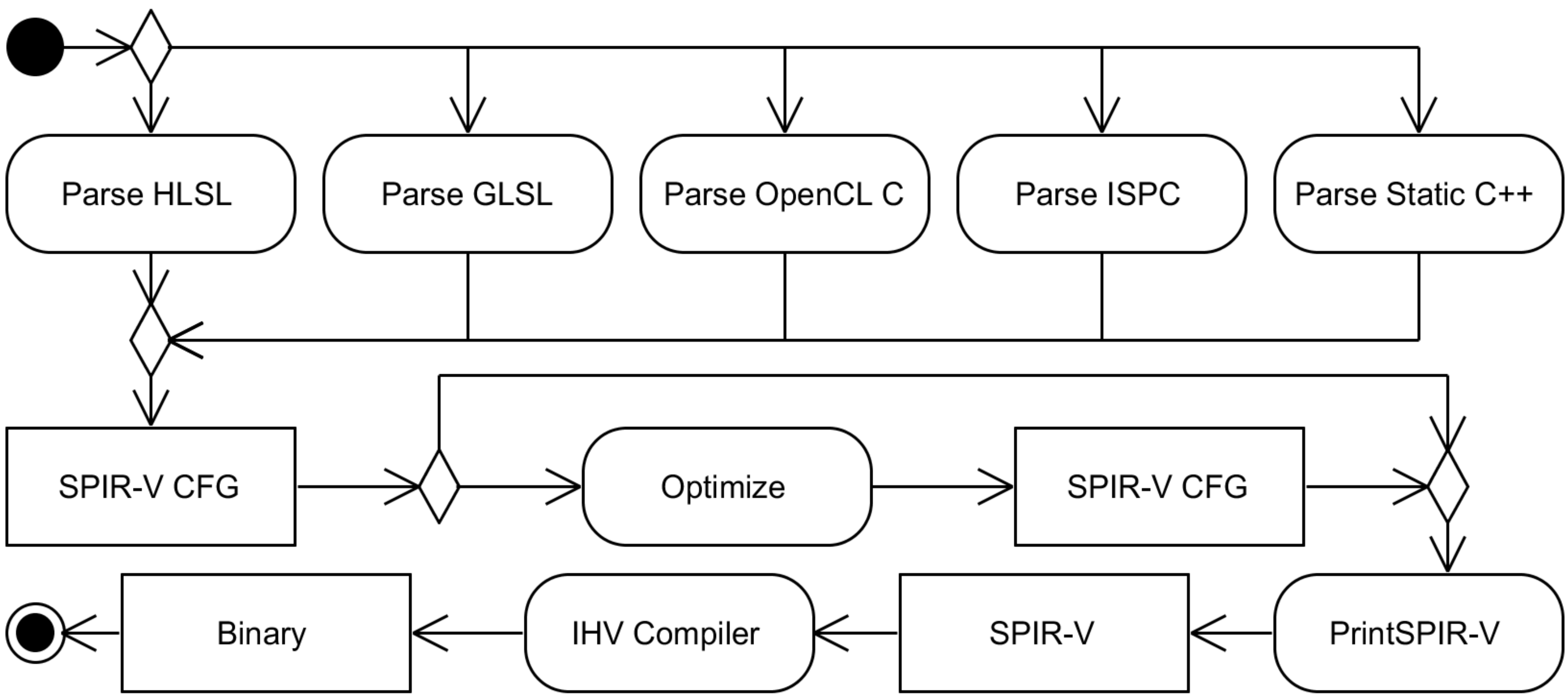
Example of a current shader pipeline



SPIR-V at the center of the pipeline



SPIR-V with a large adoption



Current SPIR-V limitations

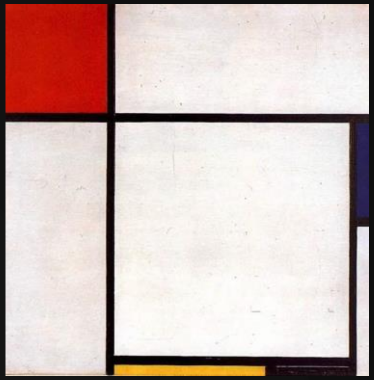
- OpenCL 1.2 / 2.0 / 2.1 – OpenGL 4.5 profiles? What?!
- We need to be able to factorize shader variations.

  Eg: Create a SPIR-V module that works for both OpenGL 4.5 compute shader and OpenCL 1.2 kernel.
- We need some level of runtime decisions. SPIR-V preprocessor?

  Eg: Use code path A if an extension is supported or else code path B.
- We need to be able to build our own profile with a clear set of features to match the market reality at any time.

  OpenGL extensions mechanism is extremely powerful

Conclusions



Composition 1929
Piet Mondrian

- SPIR-V success will depend on platform vendors adoption
- SPIR-V needs to target all existing graphics APIs and beyond
- SPIR-V requires a strong tooling ecosystem (wink wink)
- SPIR-V has potential for shading languages innovations
- SPIR-V needs to become for languages what sRGB is to color

References

- Redefining the shading languages ecosystem with SPIR-V
- Cross Platform Shaders in 2014
- Cross Platform Shaders in 2012
- Compiling HLSL into GLSL in 2010