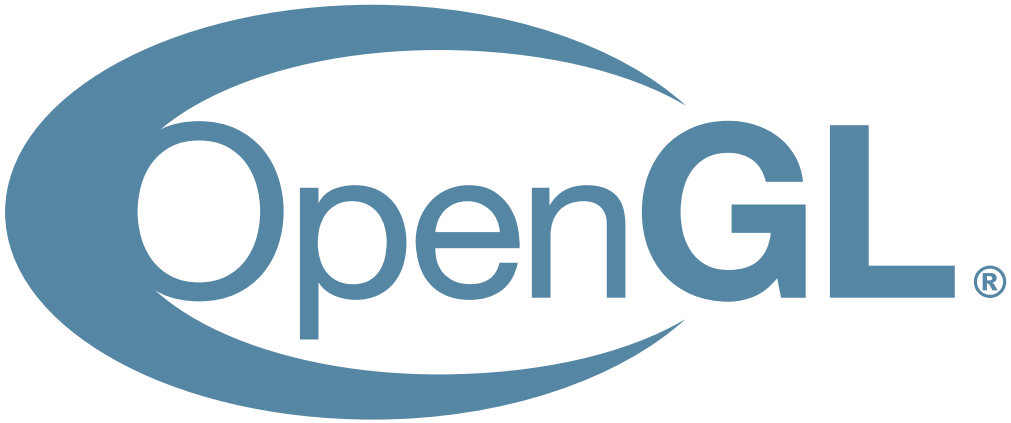
Zink: OpenGL on Vulkan



Erik Faye-Lund

XDC 2019

Open First

Why OpenGL on Vulkan

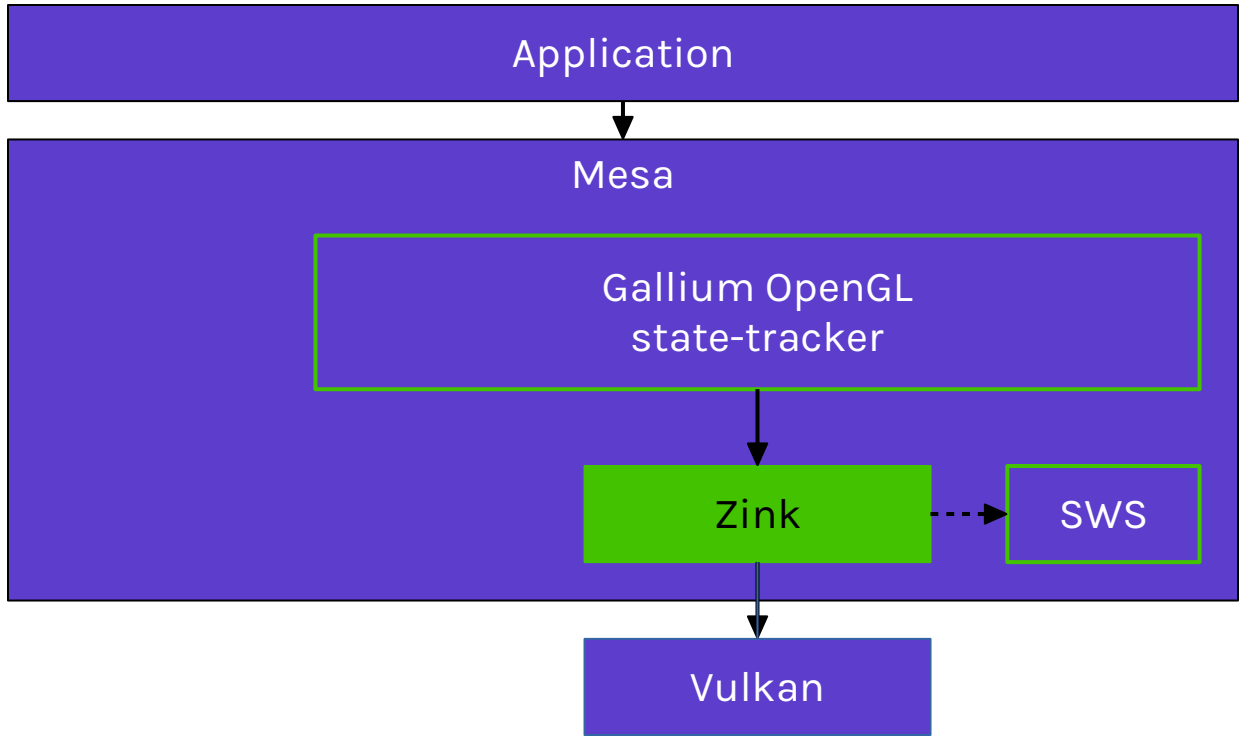
- OpenGL is slowly becoming obsolete
  - Vulkan is taking over on many platforms
  - But it's still required for a lot of applications
    - Some will probably never port over
- It's better for the community to focus on one API moving forward
  - Fewer GPU drivers to maintain
  - Vulkans driver are significantly easier to maintain than an OpenGL drivers
  - Less horizontal parts to deal with
- Support **full OpenGL** on platforms where this hasn't yet been available
  - Android, iOS, Fuchsia etc?

Zink: OpenGL on Vulkan!

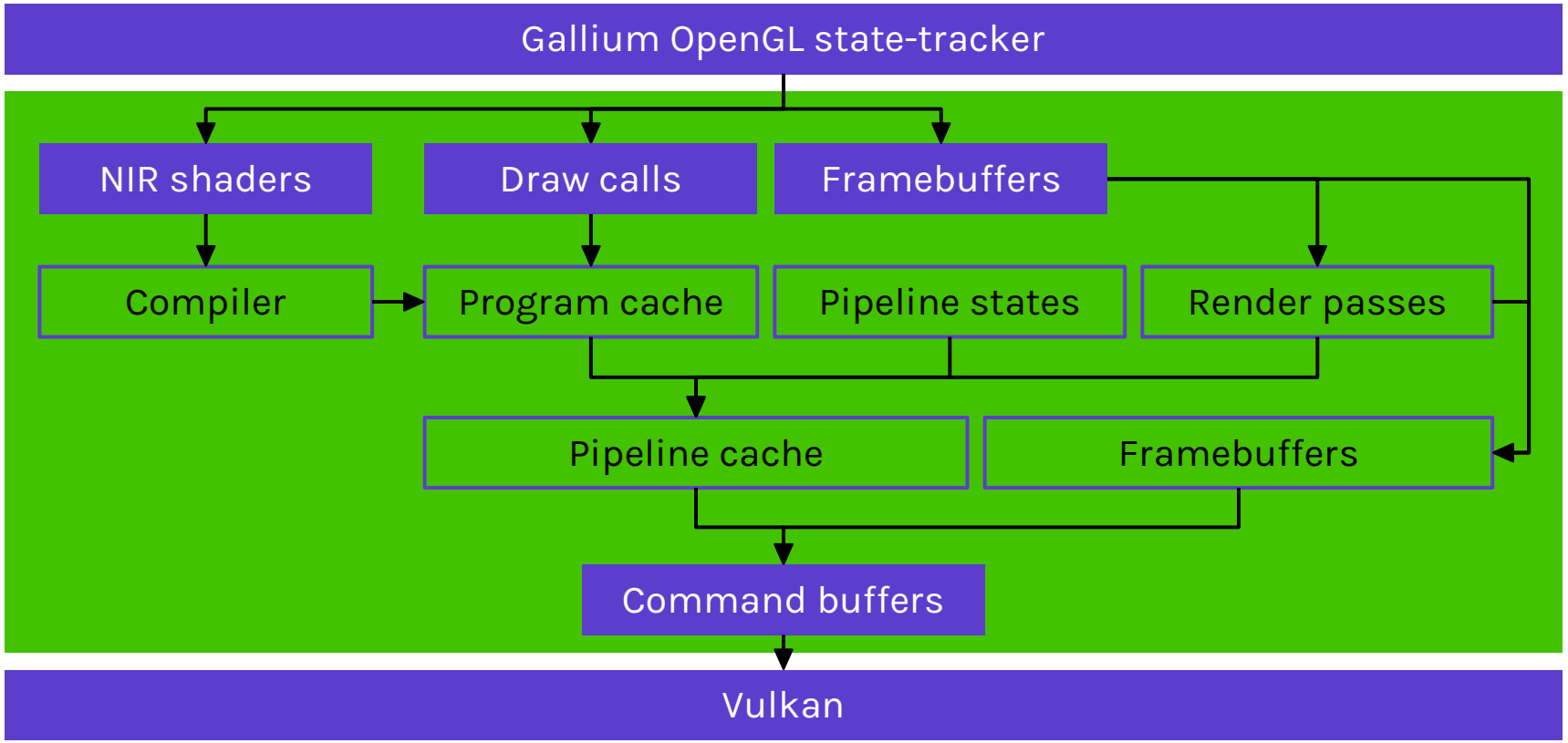
- Zink is a Gallium driver that translate gallium API calls into Vulkan calls
  - Treats the Vulkan API as if it was hardware
- Paired with the Gallium OpenGL state-tracker, we get a full OpenGL implementation
  - Currently supports OpenGL 2.1
    - (and OpenGL ES 2.0)
- But there's a few pitfalls

Time for some demos!

Architecture overview



Architecture details



Performance: Batching

- Several `zink_batch` -objects per context (currently 4)
  - Used in a round-robin fashion
  - Protected by a fence
  - Contains a `VkCommandBuffer` and a `VkDescriptorPool`
  - Keeps deleted `VkSampler` -objects alive until batch is reused
- Will flush if:
  - framebuffer is switched
  - Actual flush
  - `VkDescriptorPool` runs out of descriptors

Pitfalls

- Two-sided polygon mode
  - No good solution apart from faking it?
  - I'm open to ideas here!
- Line rasterization rules
  - Depend on recent extension
  - Can also implement diamond-exit rule in fragment shader
- Many more
  - See previous talks
  - See issue tracker: https://gitlab.freedesktop.org/kusma/mesa/issues

Upstreaming

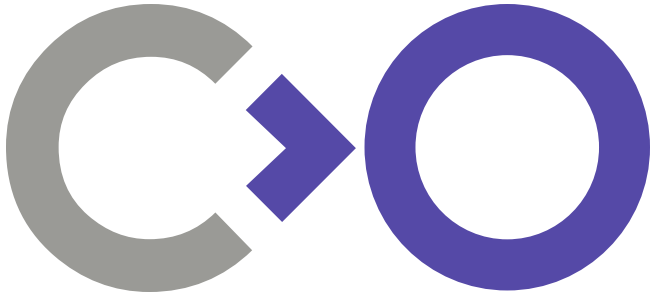
- Upstreaming in Mesa is the next goal
  - Some preparatory merge-requests sent this morning
  - Hopefully the rest will follow in the next few weeks
- Avoids reliance on one single person as a bottle-neck
  - There's several companies working on Zink right now
  - Avoids accidentally breaking things and not figuring out right away
- Gets wider scrutiny
  - Zink has modified the Gallium state-tracker a bit
- Allows building as part of distros

Changes to Gallium

- Add lowering of the following to the state-tracker
  - Flat-shading
  - Alpha testing
  - Point-size forwarding
  - Two-sided lighting
  - User-defined clip-planes
    - Lowers clip-distances to uniforms instead of driver-managed variable
- Prefer using R8 instead of A8 for glBitmap
- Will probably come first as a series, as they probaly require more review than the driver-speific bits.

Continous Integration

- After upstreaming, we should set up som CI
- Can use SwiftShader on GitLab CI for testing without hardware
  - SwiftShader contains a Vulkan 1.1 software rasterizer
- Completely TBD
  - If anyone wants to test Zink on SwiftShader, that'd be great!
  - Help wanted :)



Thank you!

Psst, slides here: https://gitlab.freedesktop.org/kusma/zink-xdc19