GStreamer Vulkan

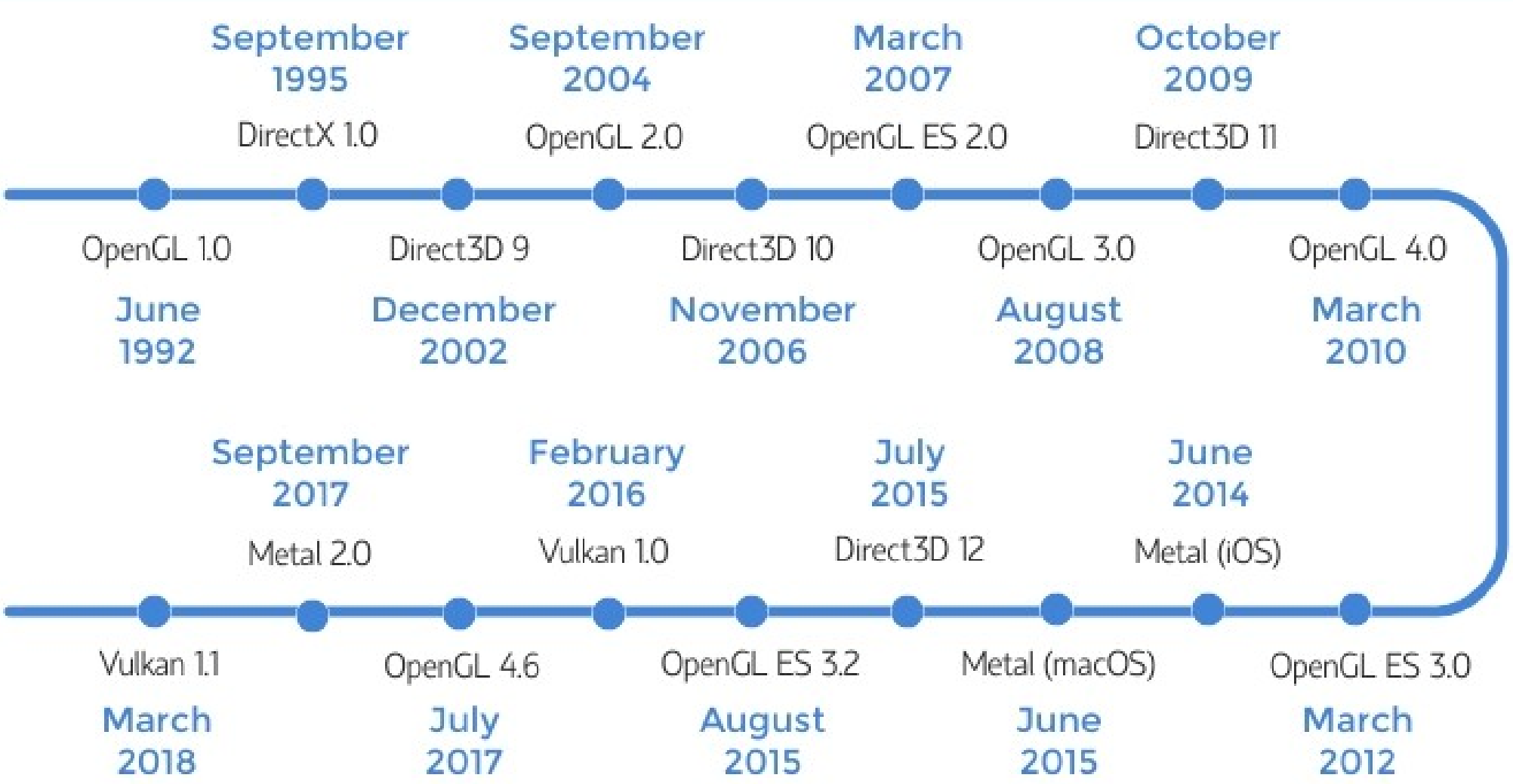
Matthew Waters (ystreet00)
GStreamer Conference 2019
1st November 2019

Centricular

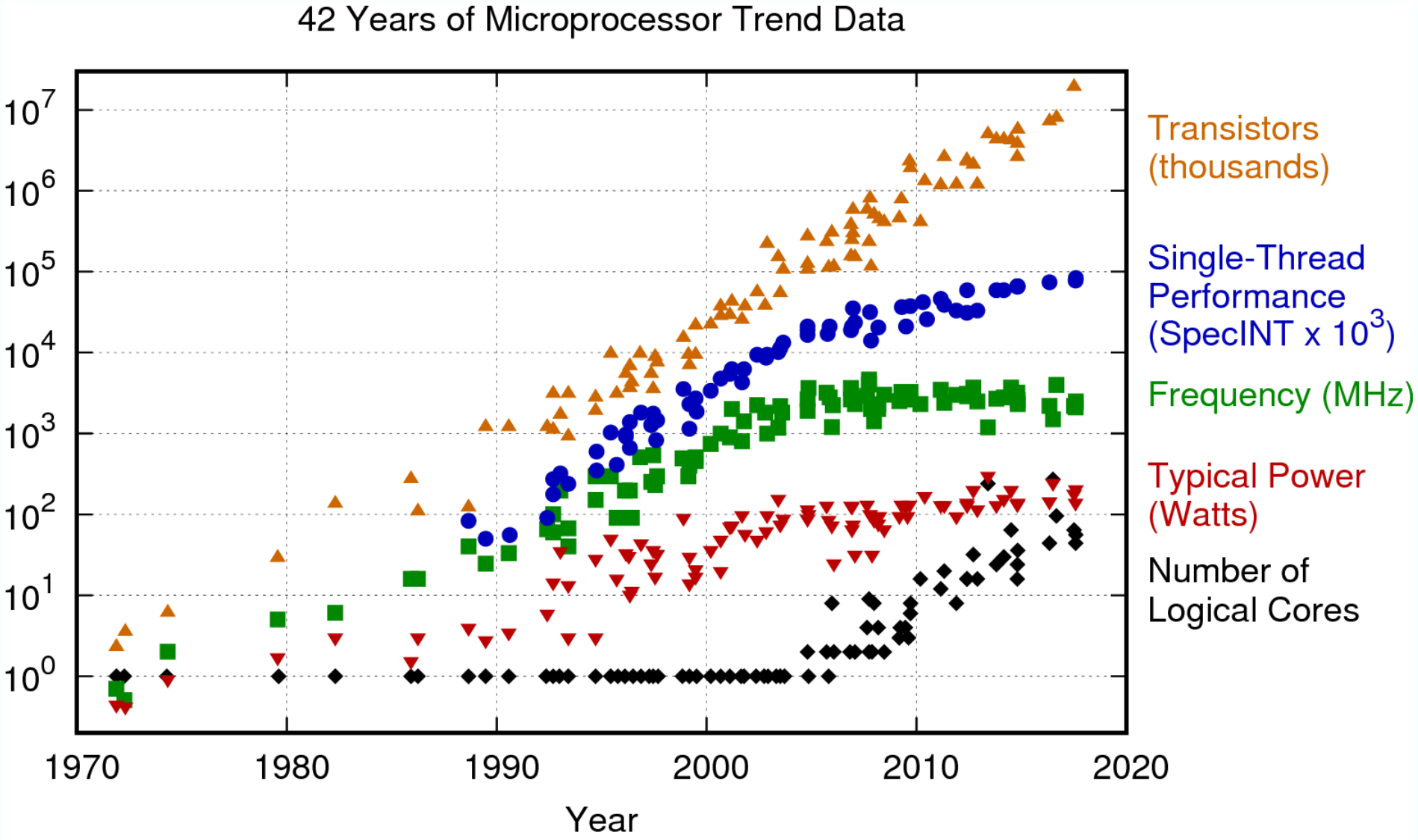
Who Am I

- Australian
- Centricular
- Multimedia
- Graphics – OpenGL, Vulkan
- WebRTC

History



Why Vulkan/Metal/Direct3D 12?



https://www.karlrupp.net/2018/02/42-years-of-microprocessor-trend-data/ - CC BY 4.0

Centricular

Why Vulkan/Metal/Direct3D 12?

- Multi-core processing
- Lower GPU driver overhead
- Closer to current state of GPU hardware
- API ‘reset’

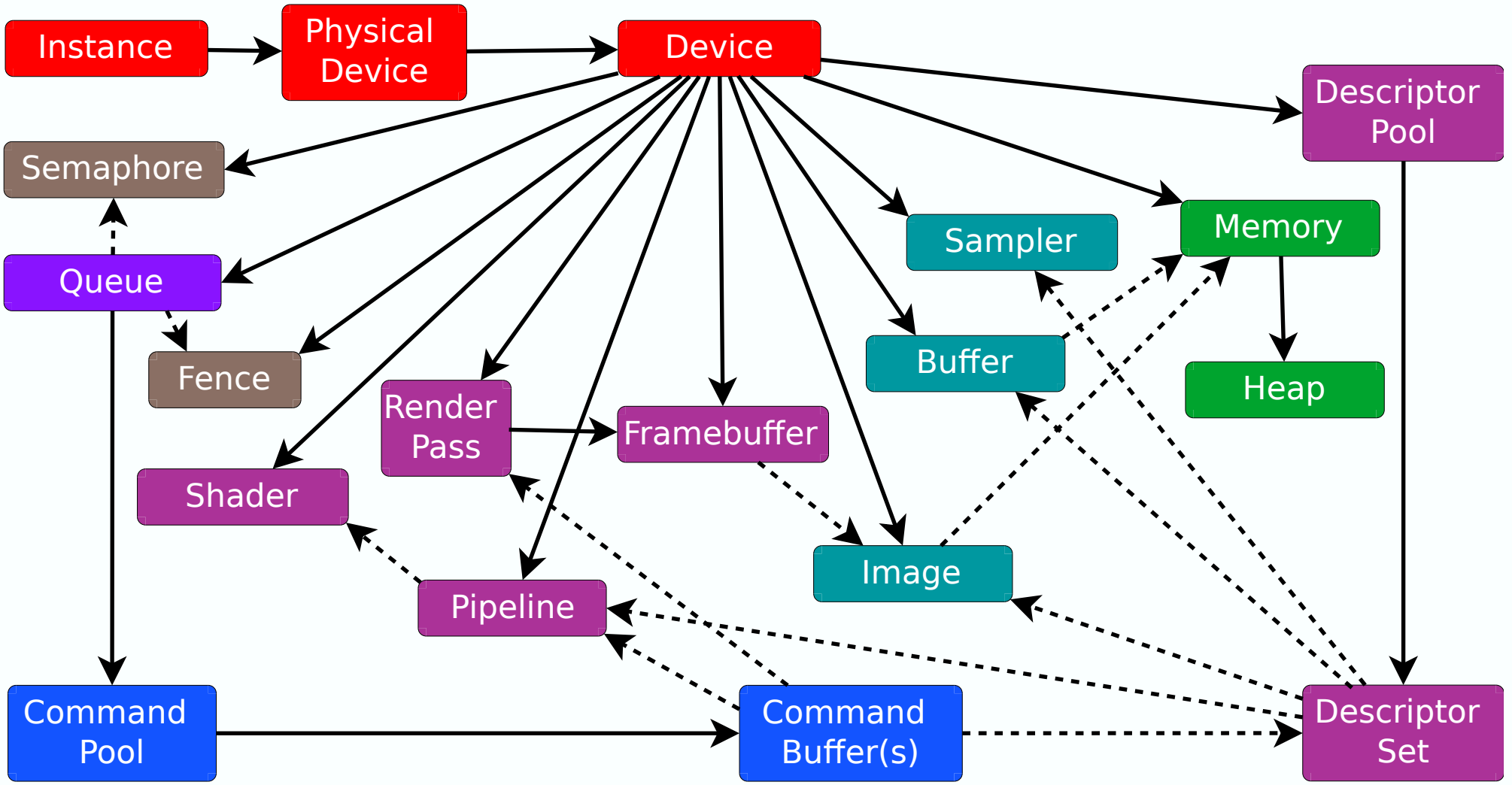
Hello Triangle

https://twitter.com/McNopper/status/1179774428357505024

Why Vulkan?

- Open cross-platform low-level 3D API
- X11, Wayland, Android, Windows, Embedded Linux
- MacOS/iOS with MoltenVK
  - OpenGL deprecated on MacOS/iOS
- Extensible
- Lots of control

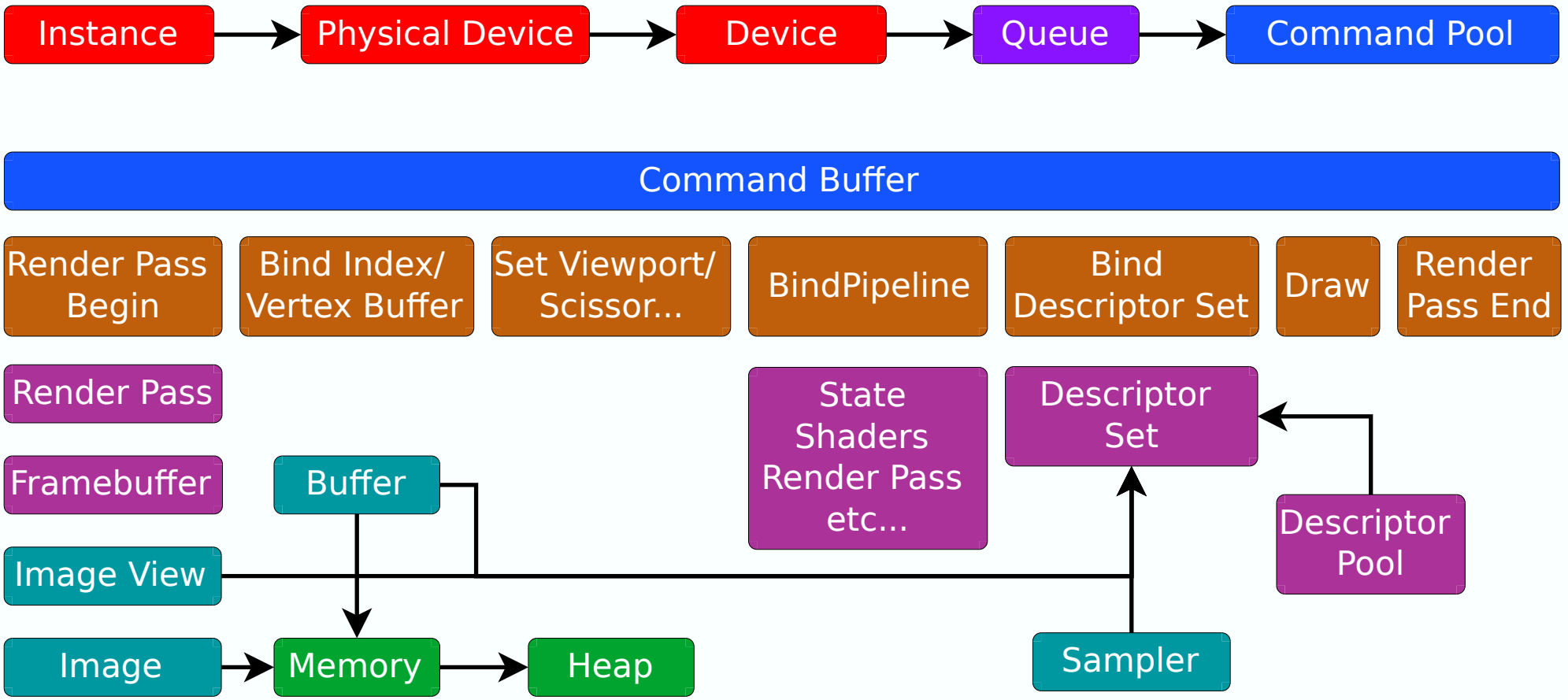
Vulkan Overview



Vulkan Extensions

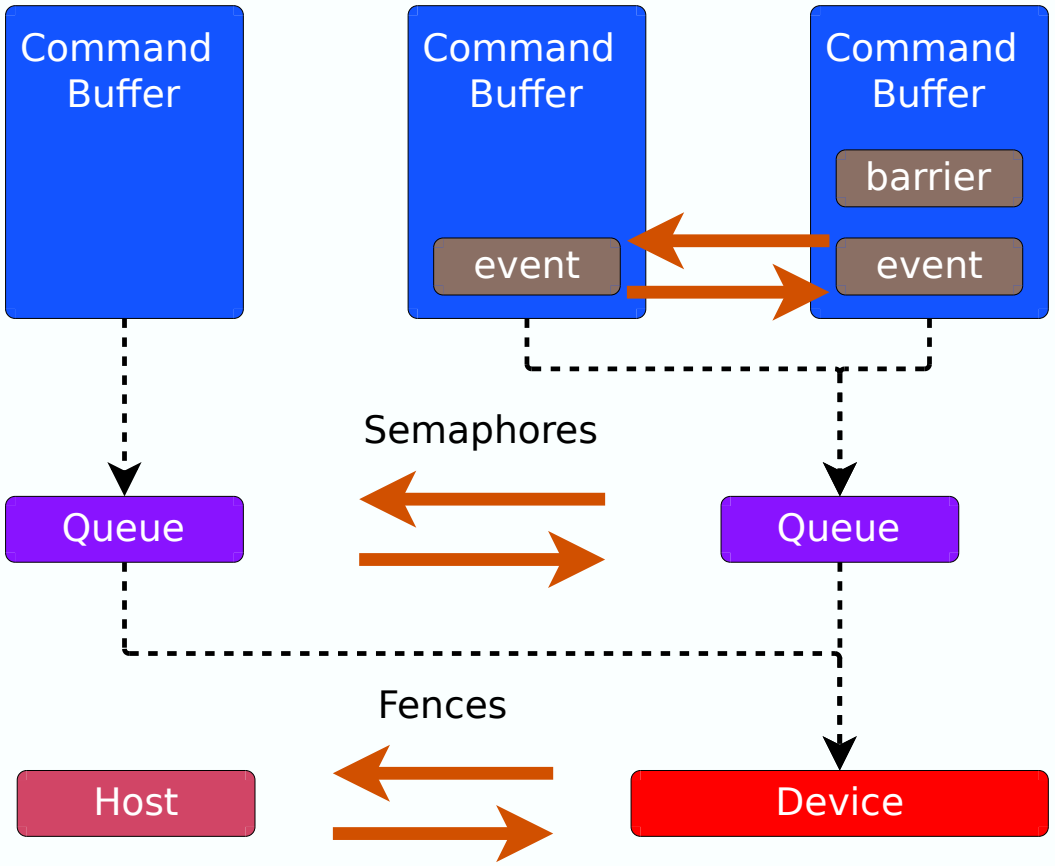
- Chosen by application at instance/device creation
  - Explicitly opt-in
- Rendering output is an extension
  - VK_KHR_SURFACE + platform-specific extension

Vulkan Command Buffers



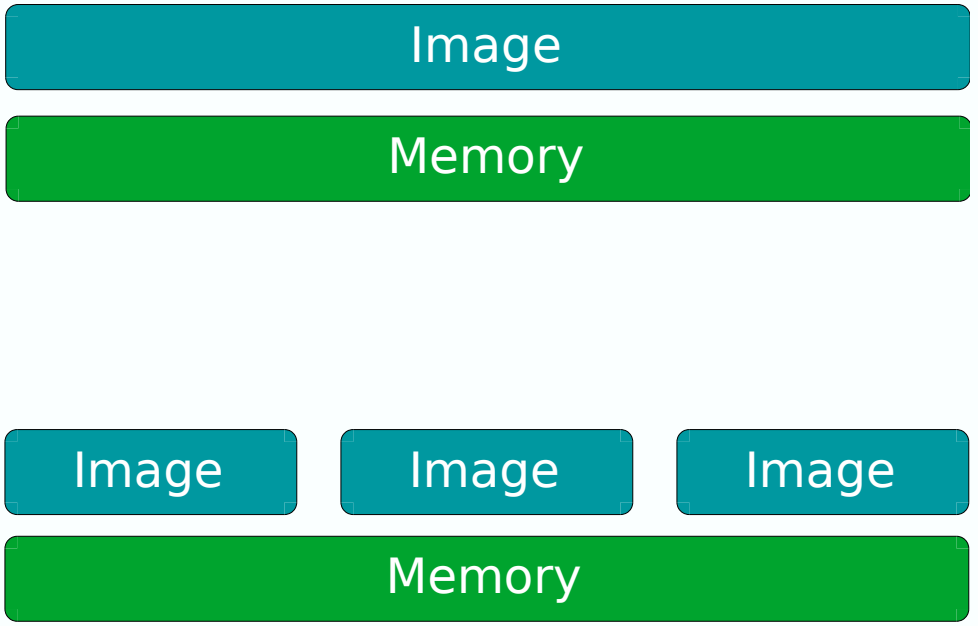
Vulkan Synchronisation

- Semaphores – device→device inter-queue or coarse same-queue
- Barriers and Events – device→device (fine-grained)
- Fences – host→device
- Timeline semaphores – future development



Vulkan Memory Allocation

- Vulkan-user controls memory allocation
- Some devices/drivers have small (4k) allocation count limits
- MoltenVK prefers dedicated due to Metal API
- Aliasing is possible



Vulkan Memory

- `VkImage`/`VkBuffer` is memory + layout description
- `VkImageView` is image + 'how to interpret'
- Images can have optimal or linear tiling
- Images can have stage-specific layouts (present, transfer, color attachment)
- Image/buffer requirements retrieved with `vkGet*MemoryRequirements()`
- `VkDeviceMemory` bound to image/buffer/etc with `vkBind*Memory()`

Shaders

- Core Vulkan only accepts SPIR-V shaders
- Reduces driver variations in shader parsing
- GLSL/HLSL→SPIR-V compiler exist

GStreamer Vulkan Library

- Very similar to GStreamer OpenGL
  - `GstVulkanDisplay` ~= `GstGLDisplay`
  - `GstVulkanWindow` ~= `GstGLWindow`
  - `GstVulkanDevice` + `GstVulkanQueue` ~= `GstGLContext`
  - `GstVulkanInstance`/`GstVulkanPhysicalDevice` has no analogue
  - `GstVulkanImageMemory` ~= `GstGLMemory`
  - `GstVulkanBufferMemory` ~= `GstGLBuffer`
  - `GstVulkanBufferPool` ~= `GstGLBufferPool`

GStreamer Vulkan Library 2

- GstVulkanCommandPool
  - Parent of GstVulkanCommandBuffer
- GstVulkanFence
- GstVulkanTrash
- GstVulkanSwapper
  - Renders to a VkSurface

Sharing Vulkan resources

- `GstVulkanDevice/Instance`
- `GstVulkanImage/BufferMemory`
- Synchronisation information
  - Fence
  - Event
  - Semaphore
- “external”

GStreamer Vulkan Elements

- `vulkanupload` – upload data to Vulkan memory
- `vulkancolorconvert` – convert between colour spaces
- `vulkanviewconvert` – convert between stereo view modes
- `vulkanimageidentity` – image copying
- `vulkansink` – output to screen

GStreamer Vulkan Issues

- `VkMemory` download
  - `VkImage` layout transition requirements and barriers
- Descriptor sets
- RGB→YUV shader requires separate downsampling pass
  - Imageless framebuffers?

GStreamer Vulkan Future

- Command buffer reuse
- Primary/secondary command buffers
- Render subpasses
- Reusing memory within frame
- dma-buf
- Application integration example
- Compute
- Vulkan encoders/decoders – hopefully Q1 2020

Thanks!

- ystreet00 in #gstreamer on freenode
- @ystreet00 on twitter