



Vulkan in KWin

Xaver Hugl



About me

- Graphics programming for a decade
- Working on KWin for about 6 years, and everything that comes with it
- Employed for Techpaladin Software
- Blog: zamundaaa.github.io

Xaver Hugl



What even is Vulkan?



What even is Vulkan?

API to do stuff with graphics cards

- Copy
- Render
- Generic compute



OpenGL

- Also does stuff with graphics cards
- Released in 1992
- OpenGL 1.0: fixed function approach
- KWin's baseline: OpenGL 2.0 (2004)



OpenGL

- High level API
- State based API
 - create an object: *glGenTextures(...)*
 - bind it: *glBindTexture(GL_TEXTURE_2D, ...)*
 - upload data: *glTexImage2D(GL_TEXTURE_2D, ...)*
- All state is part of an OpenGL context



OpenGL contexts

- Each thread can have one context current
- Creating contexts is expensive
- Switching between contexts can be expensive
- Dealing with multiple contexts is a pain
 - One for rendering
 - One for QtQuick effects
 - One per graphics card



OpenGL Driver Magic

“upload data” / *glTexImage2D* means:

- Allocate a staging buffer on the CPU
- copy the data to that staging buffer with the CPU
- allocate the actual texture on the GPU
- schedule a copy with the GPU from the staging buffer to the texture
- once that copy is done, delete the staging buffer
- ...or keep it around for future uploads, or re-use it somewhere else, or...



Implicit synchronization

- Reads and writes to graphics resources are automatically synchronized
- When deleting some resource, the driver keeps it around until it's actually no longer used
- Over-synchronization can reduce performance and efficiency



More limitations in the API

- some formats can only be read from, but not written to
 - YCbCr (for video)
 - Linear buffers with proprietary Nvidia driver
- can't use realtime compute
- for most of the API, the driver must ensure application behavior is correct



How is Vulkan better?

- Low level API, everything is explicit
- Driver does basically exactly what you tell it to
- Synchronization is explicit
- No context, very little state
- Better compatibility with "weird" hardware and formats
- Validation layers catch a lot of problems



How is Vulkan worse?

- Low level API, everything is explicit
- Driver does basically exactly what you tell it to
- Synchronization is explicit
- GPU resets are much easier to trigger
- Descriptors



Vulkan Descriptors

- Descriptors are blobs of data shaders can access: uniforms, samplers, textures
- Every shader has a different data layout for descriptors
- New extensions to improve things:
 - *VK_EXT_descriptor_buffer*
 - *VK_EXT_descriptor_heap*



VK_EXT_descriptor_heap

- Buffer for resource descriptors
- Buffer for image/sampler descriptors
- No dependency on a specific shader
- Doesn't support YCbCr
- Only *very* recently supported in drivers



Multi-GPU copies

- Multi-GPU copies with OpenGL requires context switches
- With Vulkan
 - *vkCmdBlitImage*, no rendering or shaders needed
 - 10% performance improvement with Cyberpunk 2077 on my eGPU, just with *vkCmdCopyImage*
 - Allowed enabling version 6 linux-dmabuf protocol
 - Allowed working around some driver bugs



Plan: Vulkan renderer

- Performance: likely no big jumps, but hopefully some reduced CPU usage
- Asynchronous compute: lower and more predictable latency, fewer frame drops
- Hardware enablement: writing a Vulkan driver is faster than OpenGL



Vulkan renderer prototype

- Lots of hacks
- No support for effects
- Definitely much slower than OpenGL



Effects API

- Currently basically assumes OpenGL
- for all effects: *prePaintScreen*
 - for all windows, for all effects: *prePaintWindow*
- for all effects: *paintScreen*
 - for all windows, for all effects: *paintWindow*
 - for all windows, for all effects: *drawWindow*
- for all effects: *postPaintScreen*



New Effects API

- Contain most rendering in Kwin itself
- *prePaintScreen, postPaintScreen*
- Everything is an *Item*: *WindowItem, SurfaceItem, DecorationItem* etc.
- Effects can already add their own *Items*
- Also allow modifying existing ones: move, scale, clip, change opacity, change brightness, duplicate window, hide windows, replace windows
- Item for background effects
- Prototype looks promising



Two Vulkan renderers?

- Desktop GPUs: asynchronous compute
- Mobile GPUs: renderpass, fragment shaders