20180206 - Vulkan ASKs : Swapchain

This starts a series of posts outlining what I personally want added to Vulkan. Starting with some of the swapchain related issues ...

Prior posts 20180202 and 20180203 are good background.

For later reference: a single 32-bpp swapchain image for an 8K display (or desktop with four 4K displays) is a little under 128 MiB. Bumping up to something like scRGB (requires a 64-bpp swapchin) is thus double that, a little under 256 MiB, or a quarter of a GiB. All for one image. Hopefully this places the desire for minimizing swapchain memory waste in context.

Ability to Legally Skip VK_IMAGE_LAYOUT_PRESENT_SRC_KHR

Specifically ability to present from either **VK_IMAGE_LAYOUT_GENERAL** (written by CS) or **VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL** (written by FS). The API provides the swapchain images to the application. If those images are not in a form which is natively presentable, this is a serious design flaw IMO. *My current workaround is to just illegally skip the transition into/out-of PRESENT_SRC.*

VK_PRESENT_MODE_IMMEDIATE_1_DEEP_NO_COPY_ROUND_ROBIN_EXT

This mode represents the desired mode for "benchmarking" and "low-memory-usage immediate presentation" where the application is accepting on-screen tearing. The mode works similar to VK_PRESENT_MODE_IMMEDIATE_KHR, with improvements.

This mode allows both the app and compositor to access a swapchain image at the same time. This is a relatively new concept for those involved in modern compositors: *that the app need not wait until the compositor is done using the image, which is the key part for performance, no serial dependencies.* This can as a side effect possibly cause tearing in the application when windowed. The probability of tearing being a function of swapchain depth, and duration of the app's last pass which writes into the chain (which is fractionally small in modern applications).

The **NO_COPY** part is a guarantee that the presentation engine will **not** make an extra hidden copy (saving the execution time of a hidden copy, and the memory of a hidden image).

The **1_DEEP** part is a guarantee of support for 1-deep swap chains. This provides the most minimal memory overhead option to the application. The application is free to use larger than 1-deep swap chains to reduce the chance of seeing tearing as well. *Note a 1-deep swap-chain when full-screen is also the way to get front-buffer rendering.*

The combined effect of NO_COPY and 1_DEEP for windowed applications can be a massive amount of memory savings. Looking at a machine with a single 8K display running in scRGB windowed. The worst possible case would be an app attempting to run 3-image MAILBOX (since devs knows the compositor is mailbox) paired with a driver that is doing a hidden copy when windowed. This requires 6 swapchain images totalling to a little under 1.5 GiB. Running NO_COPY and 1_DEEP would be a little under a 1.25 GiB savings.

The **ROUND_ROBIN** part is a guarantee that if the swap chain is 2 or more deep, that image acquire ordering is round-robin. Specifically the 'pImageIndex' across a series of calls returns an increasing repeating sequence starting at 0, going to {swapchain depth - 1}. *Since image ordering is known statically, the vkAcquireNextImageKHR() call becomes explicitly non-blocking, which is key to performance in some cases.* Implementations are allowed some flexibility in supporting this extension. Specifically if the implementation needs to silently not display a frame during say ALT+TAB full-screen flip to composite switch on Windows, it is allowed to do so to maintain the round-robin ordering as seen by the application. But during steady state clearly the driver should not be silently dropping frames.

VK_PRESENT_MODE_FIFO_RELAXED_2_DEEP_NO_COPY_ROUND_ROBIN_EXT

This mode represents the desired mode for "vsync+fullscreen" in desktop games (mode desired when getting a flip instead of composited blit, minimal latency with no animation judder, and accepting poor frame rate if missing the relaxed v-sync). The **2_DEEP** part is a guarantee of support for 2-deep swap chains.

VK_PRESENT_MODE_VARIABLE_REFRESH_2_DEEP_NO_COPY_ROUND_ROBIN_EXT

This mode represents the desired mode for "freesync+fullscreen" in desktop games. Provides varaible refresh rate at the expense of possible visible animation judder. This would explicitly enable Freesync (or say G-Sync) when fullscreen.

Ability to Query V-Sync GPU Timestamp and Refresh Interval

This is the critical feature for making it easy to adapt rendering cost dynamically to maintain v-sync. See 20180203.

Update: looks like VK_GOOGLE_display_timing could solve this problem.