20180202 - Optimized Swapchain in Vulkan

Swapchain in Vulkan can be challenging, some thoughts on doing a good job ...

Recommended in-app display modes given what is supported in Vulkan as of Feb 2018.

> **Fullscreen Immediate** .... IMMEDIATE .... 2-deep commandchain ... 2-deep swapchain
> **Fullscreen V-Sync** .... FIFO_RELAXED (fallback to FIFO) ... 2-deep commandchain ... 2-deep swapchain
> **Fullscreen Mailbox** ...... MAILBOX (if supported) .... 2-deep commandchain ... 3-deep swapchain
> **Windowed** .... IMMEDIATE or MAILBOX (same) .... 2-deep commandchain ... 3-deep swapchain

2-deep commandchain means that the N command buffers used in a frame are all double buffered. So a different set on even and odd frames.

> **Borderless Fullscreen** ... IMMEDIATE or MAILBOX (same) .... 2-deep commandchain ... 3-deep swapchain

I personally don't see the point of "Borderless Fullscreen" but since others do, here is how to best get it in Vulkan. Fullscreen is automatic in Vuklan when the window is exactly the size of the display, exactly covering the display, and front-most. One can get Borderless Fullscreen by making either the height or width an extra pixel compared to Fullscreen.

Multi-Display Suggestions
Since the windowing compositor takes GPU cycles I'd suggest providing the user an option to grab all the displays in fullscreen mode, sending a black screen to the other unused displays. This would get rid of the windowing compositor overhead.

Optimal 2-Deep FIFO/FIFO_RELAXED For V-Sync
Desired setup for fullscreen for games which can hit v-sync, providing low-latency low-memory-overhead and judder-free animation.

> 2-deep swapchain with FIFO (or better FIFO_RELAXED) for v-sync
> 2-frame-deep buffering of a frame's command buffers
> Use at least one beginning-of-frame fence to stop begin/reset from happening during prior usage
> Fence is set to trigger after last command buffer of that frame is done
> Use at least 2 command buffers in the frame
> One for all work done before writing into the swapchain (pre-acquire) ... [gpuA] below
> A second for the pass which writes into the swapchain image (post-acquire) ... [A] below
> The post-acquire command buffer waits on the acquire semaphore
> The post-acquire command buffer signals the semaphore which present waits on

Running faster than v-sync.

```
                          |<- Command buffer [gpuA] is free to start after [B] (no semaphore)
                          |
                          |    >||<- Command buffer [A] writes into swap (waits on acquire semaphore)
                          |     ||
                   [__scanoutA__][__scanoutB__][__scanoutA__][__scanoutB__]
        [gpuA][A][gpuB][B][gpuA] [A][gpuB]     [B][gpuA]     [A][gpuB]
[cpuA_][cpuB_]   [cpuA_]  [cpuB_]   [cpuA_]       [cpuB_]       [cpuA_]
                ||
               >||<- prior A frame finish triggers CPU to start on frame A again
```

GPU-limited missing v-sync.

```
              |   A miss   |                            |   A miss   |                            |   A miss   |
[__scanoutB__][__rescan_B__][__scanoutA__][__scanoutB__][__rescan_B__][__scanoutA__][__scanoutB__][__rescan_B__]
___gpuA______][A][______gpuB______][B][______gpuA______][A][______gpuB______][B][______gpuA______][A][______gpuB
```

Running fully maxed but hitting v-sync.

```
                             [__scanoutA__][__scanoutB__][__scanoutA__][__scanoutB__]
               [___gpuA__][A][___gpuB__][B][___gpuA__][A][___gpuB__][B][___gpuA__][A][
[____cpuA____][____cpuB____] [____cpuA____][____cpuB____][____cpuA____][____cpuB____][
                            ||
                           >||<-- CPU work re-aligns at commannd buffer fence
```

This method can handle volatile run-times when hitting v-sync, because it maximizes the amount of the frame which can start early.

```
[__scanoutA__][__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
[B][___gpuA__][A][___gpuB____][B][_gpuA_] [A][___gpuB___][B][___gpuA__][A][_gpuB__] [B][___gpuA___
   [cpuB_]       [cpuA_]         [cpuB_]     [cpuA_]        [cpuB_]       [cpuA_]      [cpuB_]
```

Latency to a CRT given classic non-late-latch engine design.

```
                                             |<-- less than 3 frames of latency --->|
                                             |                                      |
[__scanoutA__][__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
[B][___gpuA__][A][___gpuB____][B][_gpuA_] [A][___gpuB___][B][___gpuA__][A][_gpuB__] [B][___gpuA___
   [cpuB_]       [cpuA_]         [cpuB_]     [cpuA_]        [cpuB_]       [cpuA_]      [cpuB_]
```

Latency to a CRT given late-latch (assuming worst case no view-independent work on GPU).

```
                              under 2 frames of latency -->|                        |<-
                                                           |                        |
[__scanoutA__][__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
[B][___gpuA__][A][___gpuB____][B][_gpuA_] [A][___gpuB__][B][___gpuA__][A][_gpuB__]  [B][___gpuA___
   [cpuB_]       [cpuA_]         [cpuB_]     [cpuA_]       [cpuB_]       [cpuA_]       [cpuB_]
```

Can tune by changing amount of work in pre-acquire and post-acquire command buffer split. Increase the work in the post-acquire command buffer decreases worst-case input latency.

```
                    ->|                   |<-- gets closer to one frame of latency
                      |                   |
[__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
        . . . [___B__][gpuA][___A__] . . .
```

But has a side effect of reducing the amount of frame volatility tolerance.

```
[__scanoutB__][__scanoutA__][__scanoutB__]
        . . . [B][__gpuA___] . . .
              |
[__scanoutB__][__scanoutA__][__scanoutB__]
              |    . . . [B][__gpuA___] . . .
              |          |
            ->|          |<-- volatility tolerance is size of pre-acquire work: [__gpuA___]

[__scanoutB__][__scanoutA__][__scanoutB__]
        . . . [___B__][gpuA] . . .
              |
[__scanoutB__][__scanoutA__][__scanoutB__]
              | . . [___B__][gpuA] . . .
              |     |
            ->|     |<-- volatility tolerance is size of pre-acquire work: [gpuA]
```

3-Deep FIFO Swapchain Is Bad For Input Latency

Can keep command buffers double buffered, even if swap is tripple buffered. Except this is FIFO so increasing swapchain depth by one slot increases input latency by one frame.

```
                    [__scanoutA0__][__scanoutB1__][__scanoutA2__][__scanoutB0__][__scanoutA1__][__scanoutB2__][__scanoutA0__]
        [gpuA0][A0][gpuB1][B1][gpuA2][A2][gpuB0][B0][gpuA1][A1][gpuB2][B2][gpuA0][A0][gpuB1]   [B1][gpuA2]    [A2][gpuB0]
[cpuA0][cpuB1]     [cpuA2]    [cpuB0]    [cpuA1]    [cpuB2]    [cpuA0]    [cpuB1]    [cpuA2]       [cpuB0]        [cpuA1]

. . . after steady state from above . . .

    |<--- almost 4 frames of latency for non-late latch --->|
    |                                                       |
[__scanoutA0__][__scanoutB1__][__scanoutA2__][__scanoutB0__][__scanoutA1__]
[A2][gpuB0]    [B0][gpuA1]    [A1]
    [cpuA1]
```

3-Deep Commandchain With 2-Deep FIFO Swapchain Can Also Be Bad

Another option. Which is ok for those who late-latch input. But poor for anyone (ie the majority of games) doing traditional non-late-latch input.

```
                                               late-latch latency under 2 frames -->|                         |<--
                                                                                    |                         |
                    [__scanoutA0__][__scanoutB1__][__scanoutC0__][__scanoutA1__][__scanoutB0__][__scanoutC1__][__scanoutA0__]
        [gpuA0][A0][gpuB1][B1][gpuC0][C0][gpuA1]  [A1][gpuB0]    [B0][gpuC1]    [C1][gpuA0]    [A0]           |
[cpuA0][cpuB1][cpuC0][cpuA1]  [cpuB0]    [cpuC1]      [cpuA0]                                                 |
                                                      |                                                       |
                                                      |<------ non-late-latch latency almost 4 frames ------->|
```

Using the Acquire Fence is Fail

Using the acquire fence to stall command buffer recording is a recipe for failure.

```
                   [__scanoutA__][__scanoutB__][____miss____]
        [__gpuA__][__gpuB__]     |      [__gpuA__] <- fail
[cpuA_][cpuB_]                   [cpuA_]
                                 |
                                 |<-- acquire fence
```

CPU-Stalling To Minimize Latency Does Not Work Well

This is a proxy for some VR practices. This example moves the pre-acquire command buffer to zero size to minimize latency. Which also decreases the tolerance to volatility and ability to fully utilize the GPU. Initially this might look like a good option from the latency perspective.

```
                                       |<-- latency excluding display -->|
                                       |                                 |
                 [__scanoutA__][__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
        [_gpuA_][_gpuB_]       [_gpuA_]      [_gpuB_]      [_gpuA_]      [_gpuB_]        . . .
[cpuA_][cpuB_]  [cpuA_] [cpuB_]        [cpuA_]       [cpuB_]       [cpuA_]       [cpuB_
                                       |
                                       |            |<---- latency ----->|
                                       |            |                    |
                               [__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
                               [_gpuA_]      [_gpuB_]      [_gpuA_]      [_gpuB_]        . . .
                                       |            [cpuA_]       [cpuB_]       [cpuA_]
                                       |            |
                                      >|            |<-- maximum amount of cpu idle possible
```

However this opens up the app to risk of missing frames due to "sleep" running over (OS sleep not good enough). So APP needs to busy spin which opens up risk of getting preempted by the OS (similar miss risk).

With a more realistic situation (less idle GPU) latency reduction is less.

```
                                           |<--------- latency -------->|
                                           |                            |
                              [__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
                      . . .   [____gpuA___] [____gpuB___] [____gpuA___] [____gpuB___]   . . .
                                           [cpuA_]       [cpuB_]       [cpuA_]
                                           |
                                           |       |<---- latency ----->|
                                           |       |                    |
                              [__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
                      . . .   [____gpuA___] [____gpuB___] [____gpuA___] [____gpuB___]   . . .
                                                   [cpuA_]       [cpuB_]       [cpuA_]
                                           |       |
                                          >|       |<-- maximum amount of cpu idle possible
```

Advantages reduced even more if CPU load is more realistic (using a larger amount of frame).

```
                                           |<--------- latency -------->|
                                           |                            |
                              [__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
                      . . .   [____gpuA___] [____gpuB___] [____gpuA___] [____gpuB___]   . . .
                                           [____cpuA___] [____cpuB___] [____cpuA___]
                                           |
                                           | |<------- latency -------->|
                                           | |                          |
                              [__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
                      . . .   [____gpuA___] [____gpuB___] [____gpuA___] [____gpuB___]   . . .
                                             [____cpuA___] [____cpuB___] [____cpuA___]
```

```
                                       | |
                                      >| |<-- maximum amount of cpu idle possible
```

Compared to late-latch.

```
                                                    >|            |<-- latency
                                                     |            |
                           [__scanoutB__][__scanoutA__][__scanoutB__][__scanoutA__]
                    . . .   [____gpuA___] [____gpuB___] [____gpuA___] [____gpuB___]   . . .
                                       [____cpuA___] [____cpuB___] [____cpuA___]
```

Windowed or MAILBOX

Windowed even when on IMMEDIATE display mode is effectively MAILBOX. 2-deep MAILBOX is effectively the same as FIFO. 3-deep swapchain required for MAILBOX to function as designed. 2-deep commandchain is optimal with 3-deep swapchain for MAILBOX. Showing MAILBOX's variable latency (non-late-latch).

```
                              |<------ latency ------>|        |<------- latency ------>|
                              |                       |        |                        |
                    [___scanoutA0___][___scanoutB1___][___scanoutB0___][___scanoutA1___][___scanoutA0___][___scanoutB1___]
        [gpuA0][A0][gpuB1][B1][gpuA2][A2][gpuB0][B0][gpuA1][A1][gpuB2][B2][gpuA0][A0][gpuB1][B1][gpuA2][A2][gpuB0][B0]
[cpuA0][cpuB1]     [cpuA2]    [cpuB0]    [cpuA1]    [cpuB2]    [cpuA0]    [cpuB1]    [cpuA2]    [cpuB0]    [cpuA1]
                                         |                             |  |                             |
                                         |<--------- latency --------->|  |<--------- latency ---------->|
```

And the associated animation judder.

```
|<--------21------->|         |<----------25--------->|        |<----------26---------->|       |<-----------27---------->|
|                   |         |                       |        |                        |       |                         |
|                   [___scanoutA0___][___scanoutB1___][___scanoutB0___][___scanoutA1___][___scanoutA0___][___scanoutB1___][
|       [gpuA0][A0][gpuB1][B1][gpuA2][A2][gpuB0][B0][gpuA1][A1][gpuB2][B2][gpuA0][A0][gpuB1][B1][gpuA2][A2][gpuB0][B0]
[cpuA0][cpuB1]     [cpuA2]    [cpuB0]    [cpuA1]    [cpuB2]    [cpuA0]    [cpuB1]    [cpuA2]    [cpuB0]    [cpuA1]
       |                             |   |                             |  |                             |
       |<-------------31------------>|   |<-------------31------------>|  |<-------------32------------->|
```