Contribution ID: **37** Type: **not specified**

A case study on frame presentation from user space via KMS

Thursday, 3 October 2019 11:40 (45 minutes)

Traditionally, an application had very little control about when a rendered frame is actually going to be displayed. For games, this uncertainty can cause animation stuttering [0]. A Vulkan prototype extension was added to address this problem [1].

XR (AR/VR) applications similarly need accurate knowledge of presentation timestamps in order to predict the head-pose for the time a frame will be displayed. Here, inaccuracies lead to registration errors (i.e. mismatch between virtual and real head pose), causing users to get motion sickness or to experience *swimming* of virtual content.

XR compositors also optimize for latency. An already-rendered frame is corrected for the most recent head-pose, right before its scan-out to display. The time between the correction of a frame and its presentation determines the resulting latency. In order to keep this value as low as possible, a compositor needs to control how late a frame can be scheduled in order to make the desired presentation time.

The *Atomic KMS API* is the lowest-level cross-driver API for programming display controllers on Linux. With KMS, buffers can be submitted directly from user space for display, circumventing traditional presentation layers of graphics APIs (e.g. EGL surfaces or Vulkan swapchains). This way, applications gain exclusive access to the display engine for maximum control. Collabora and DAQRI recently published the *kms-quads* sample project to demonstrate this technique [2]. While working on this, we identified several issues of the KMS API that make it challenging to implement tightly scheduled buffer presentations as required by the use cases mentioned above. For instance, which part of the scan-out signal timestamps provided by KMS refer to is not well defined. Furthermore, it is unclear what the latest point in time is that a buffer can be submitted to make a specific presentation deadline (see [3] for related discussion). The advent of adaptive-sync support in KMS makes this topic even more complex.

This talk should serve as an introduction and summary to user-driven presentation timing via KMS, based on last year's experience of implementing a KMS-based AR compositor at DAQRI. We will discuss the use-case, its implementation and demonstrate open problems of this topic, hopefully leading to further discussion at the venue.

[0] https://medium.com/@alen.ladavac/the-elusive-frame-timing-168f899aec92
[1] https://lists.freedesktop.org/archives/dri-devel/2018-February/165319.html
[2] https://gitlab.freedesktop.org/daniels/kms-quads
[3] https://github.com/mikesart/gpuvis/issues/30

Code of Conduct

Yes

GSoC, EVoC or Outreachy

No

Presenter: FINK, Heinrich (DAQRI)

Session Classification: Main Track

Track Classification: Talk (full slot) (closed)