FreeSync™, Adaptive Sync & VRR

Harry Wentland

AGENDA

Static and dynamic refresh rates

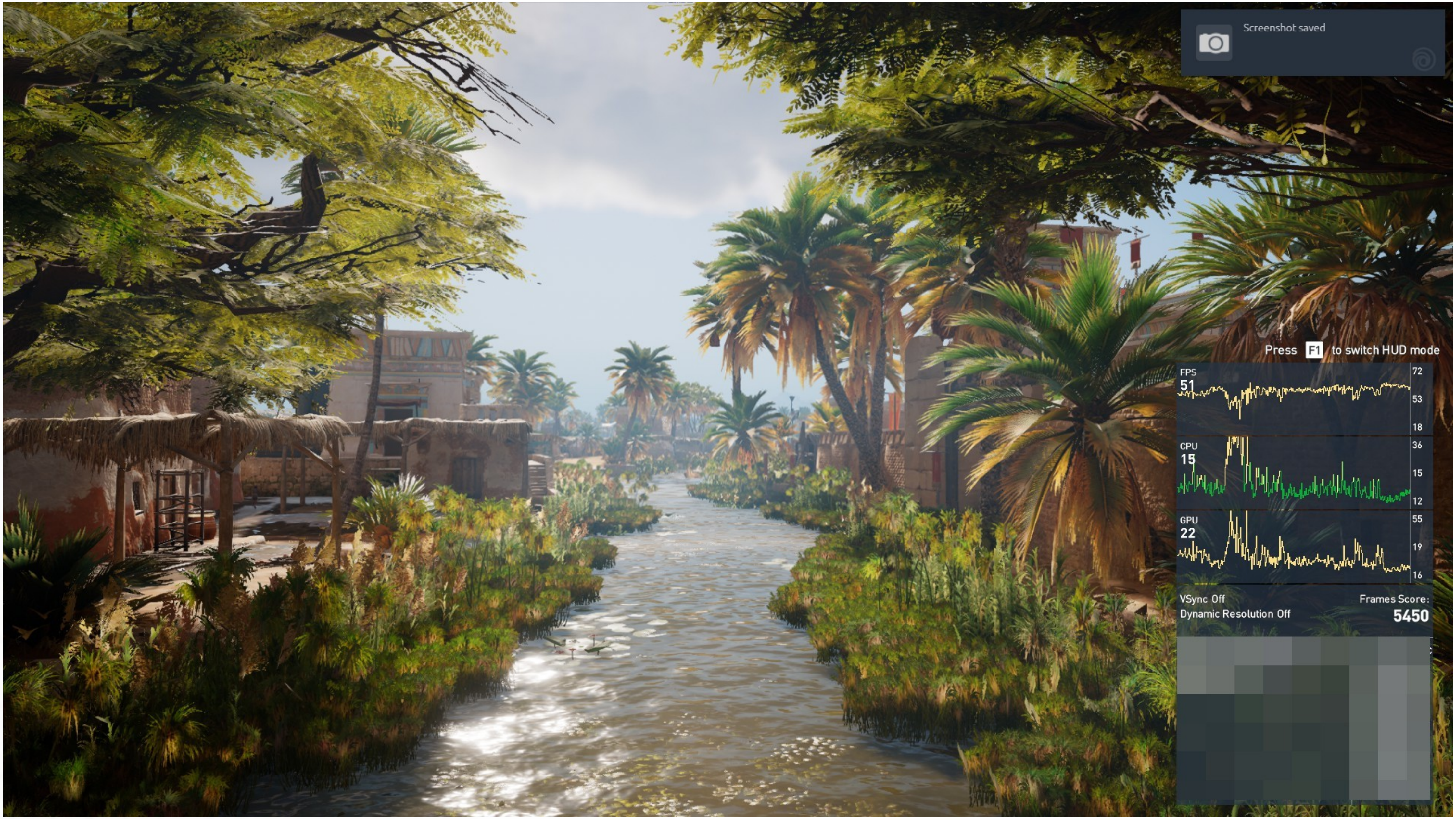
DP Adaptive Sync, HDMI™ VRR, FreeSync™

VRR in DRM & Mesa

Next Steps

Conclusion & Questions

Static and dynamic refresh rates

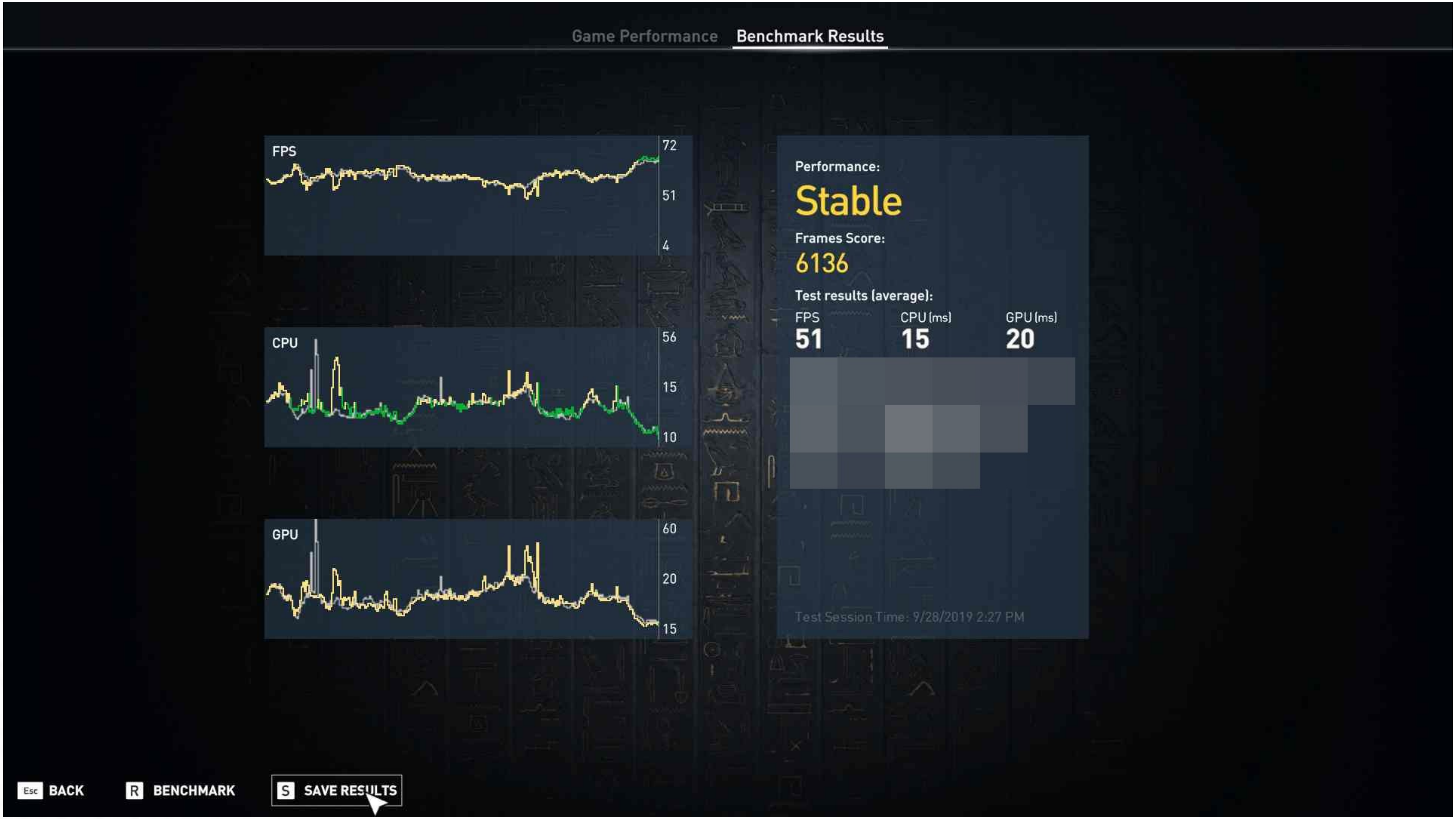


Screenshot saved

Press F1 to switch HUD mode

VSync Off
Dynamic Resolution Off

Frames Score:
5450



Performance:

Stable

Frames Score:

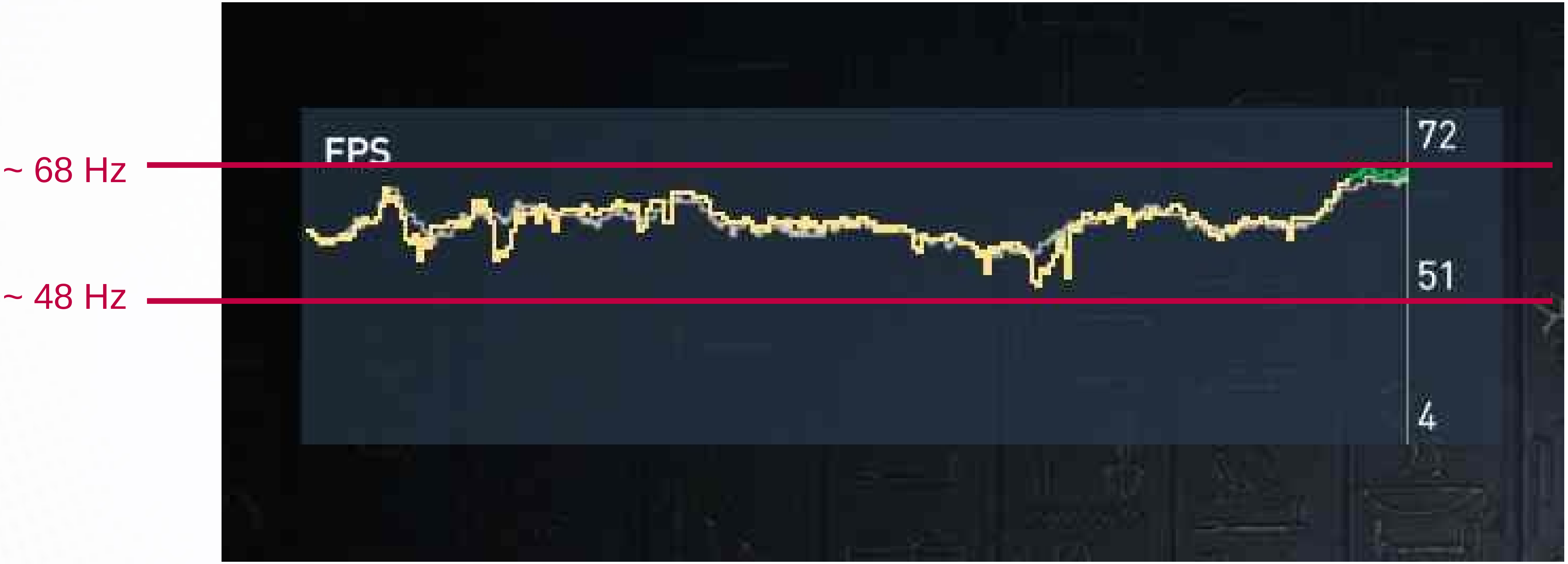
6136

Test results (average):

FPS	CPU (ms)	GPU (ms)
51	15	20

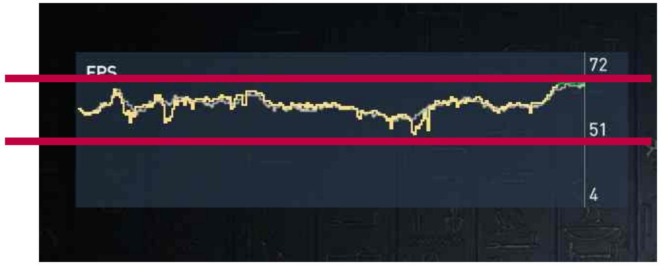
[redacted]

Test Session Time: 9/28/2019 2:27 PM


~ 68 Hz
~ 48 Hz
FPS
72
51
4

Dynamic refresh rates for gaming

- Render rate varies with content
- Latency between render and display matters

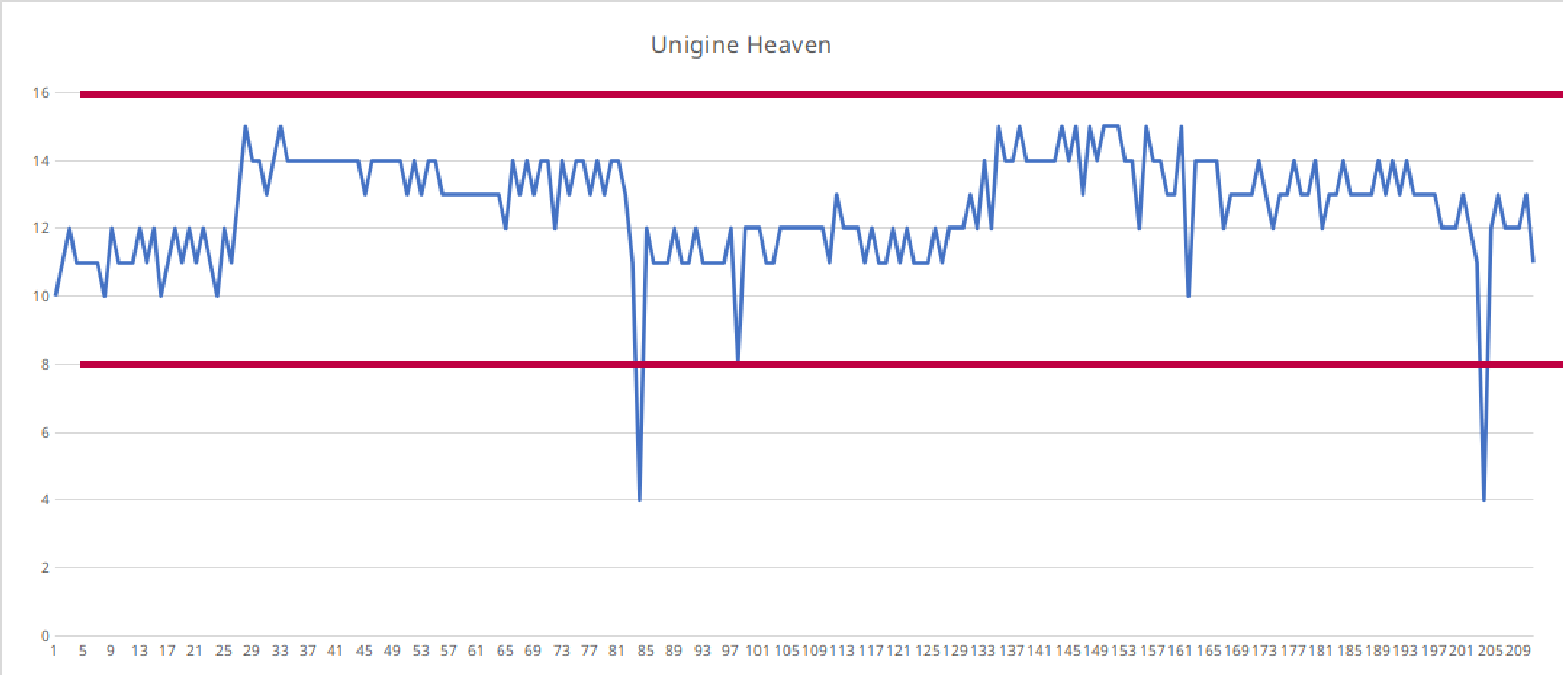


~ 68 Hz

~ 48 Hz

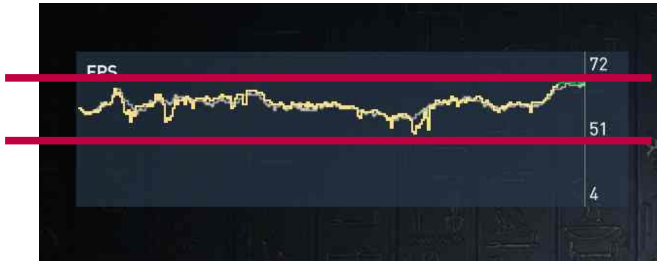
60 Hz

120 Hz



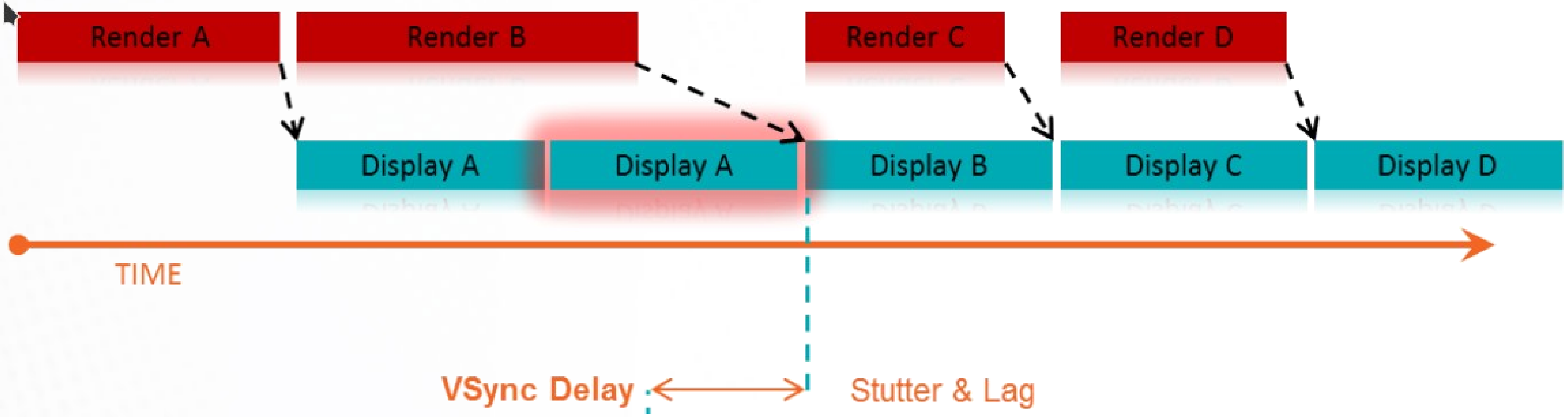
Dynamic refresh rates for gaming

- Mismatch between render rate & refresh rate leads to stutter & lag



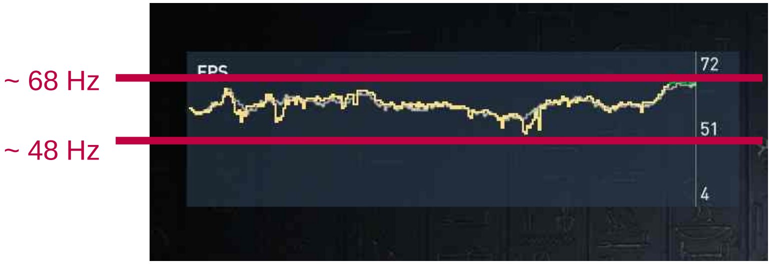
~ 68 Hz

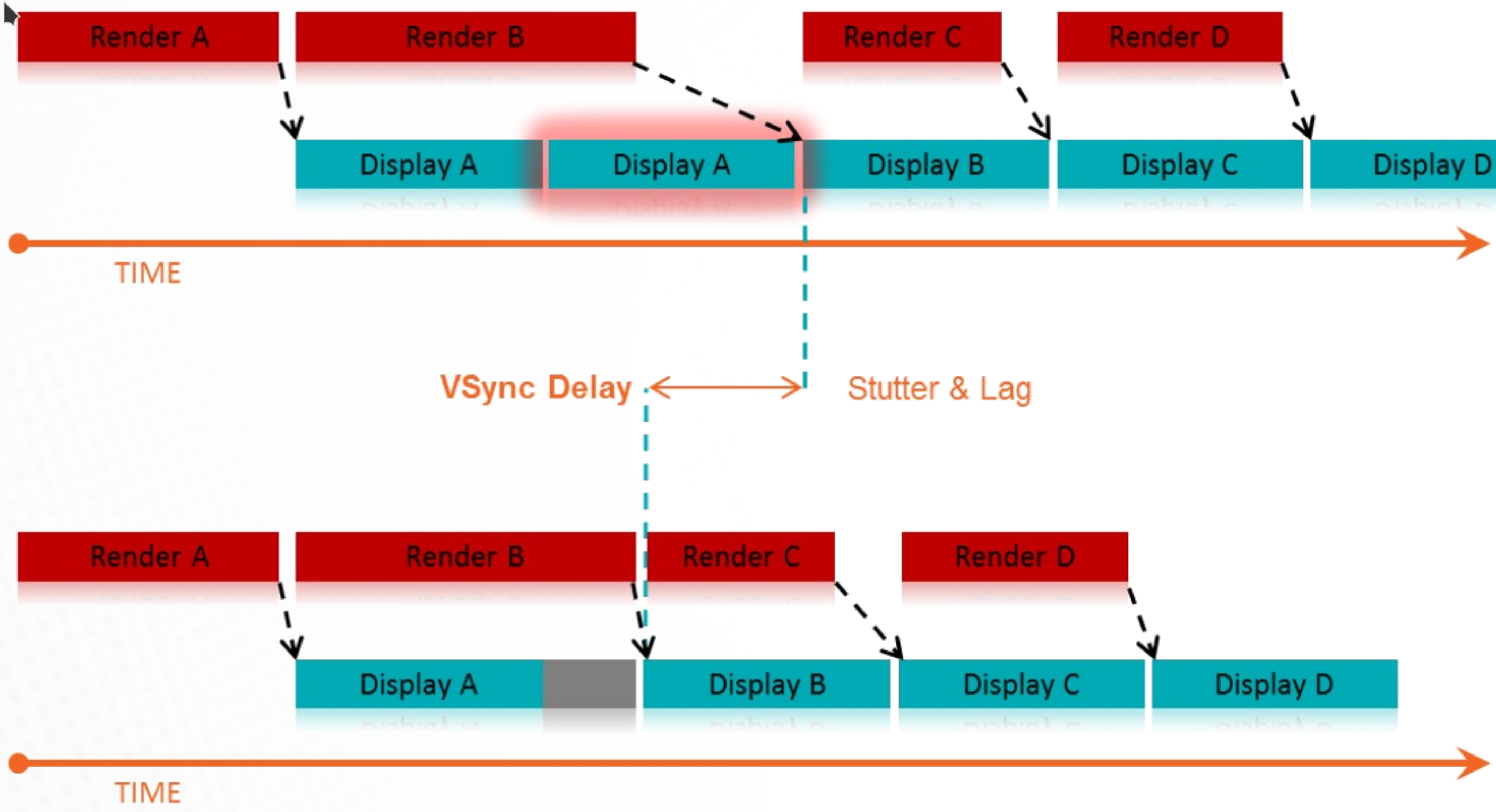
~ 48 Hz



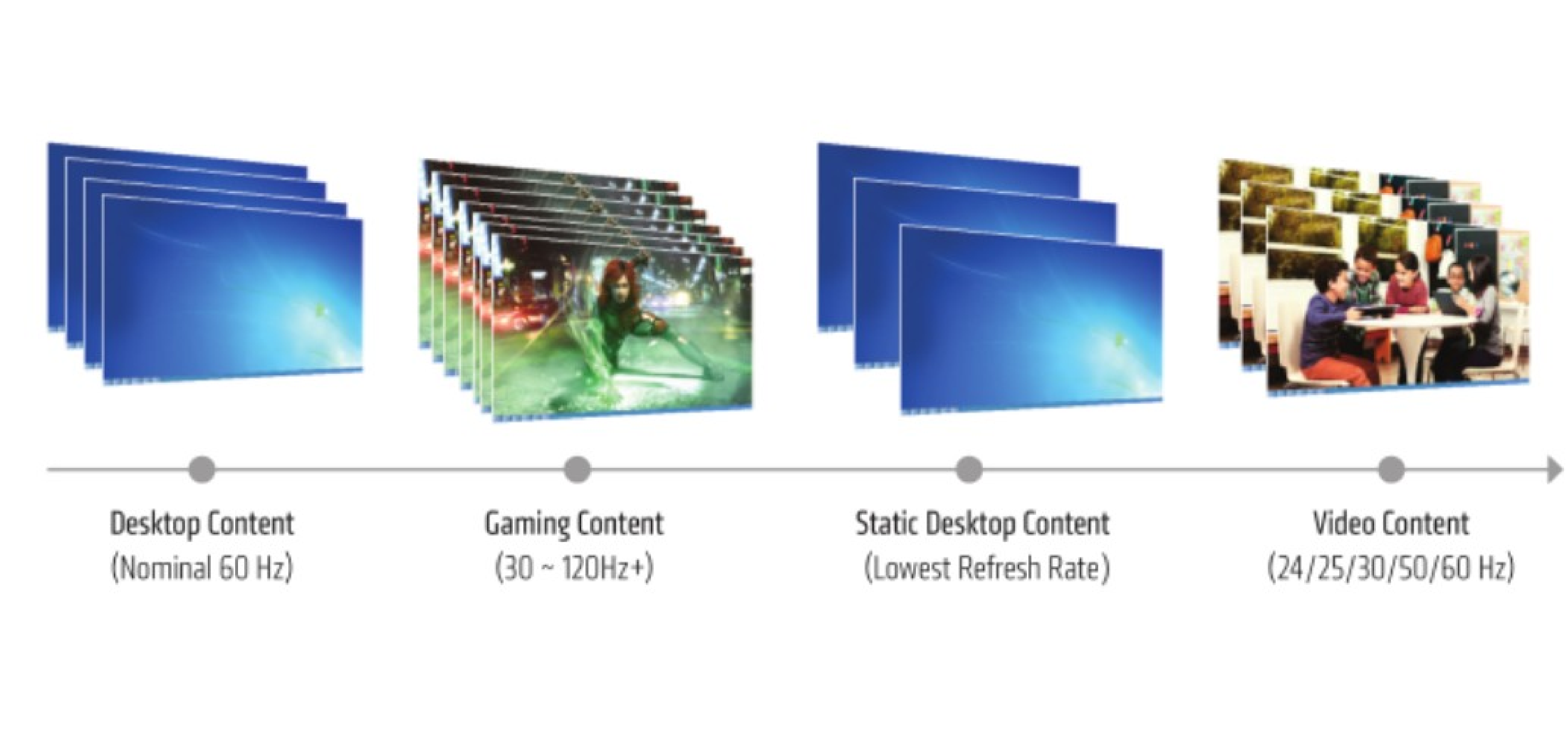
Dynamic refresh rates for gaming

- Syncing render & refresh rates reduces lag and eliminates stutter





Other dynamic refresh rate use cases

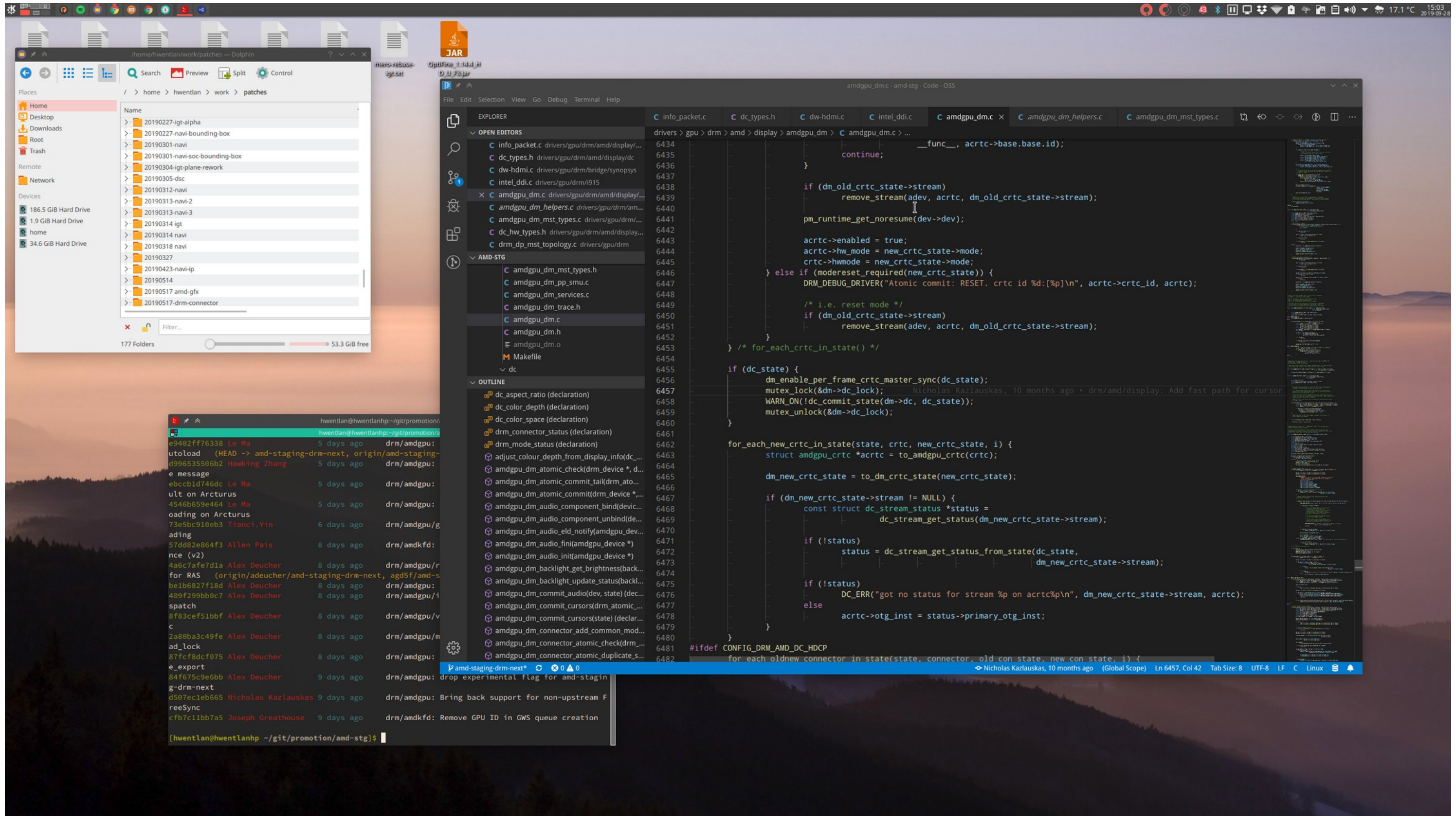


Benefits of Adaptive Sync - Video

- Video frame rates rarely match display refresh rate
- Common video frame rates: 24, 25
- Common display refresh rates: 60, 120
- Would be nice to switch to 24 Hz without a mode change
- With dynamic refresh rates we can seemlessly adjusted to video’s frame rate
- We can save power running at lower refresh rate

Benefits of Adaptive Sync – Power Saving

- Desktop content is mostly static in many cases
- Using adaptive sync system can switch to a lower refresh rate for static content
- Power Savings



/home/hwentlan/work/patches — Dolphin
amdgpu_dm.c - amd-stg - Code - OSS
info_packet.c
dc_types.h
dw-hdmi.c
intel_ddi.c
amdgpu_dm.c
amdgpu_dm_helpers.c
amdgpu_dm_mst_types.c
drivers > gpu > drm > amd > display > amdgpu_dm > amdgpu_dm.c > ...
OUTLINE
amd-staging-drm-next*

```c
                                        __func__, acrtc->base.base.id);
                                continue;
                        }

                        if (dm_old_crtc_state->stream)
                                remove_stream(adev, acrtc, dm_old_crtc_state->stream);

                        pm_runtime_get_noresume(dev->dev);

                        acrtc->enabled = true;
                        acrtc->hw_mode = new_crtc_state->mode;
                        crtc->hwmode = new_crtc_state->mode;
                } else if (modereset_required(new_crtc_state)) {
                        DRM_DEBUG_DRIVER("Atomic commit: RESET. crtc id %d:[%p]\n", acrtc->crtc_id, acrtc);

                        /* i.e. reset mode */
                        if (dm_old_crtc_state->stream)
                                remove_stream(adev, acrtc, dm_old_crtc_state->stream);
                }
        } /* for_each_crtc_in_state() */

        if (dc_state) {
                dm_enable_per_frame_crtc_master_sync(dc_state);
                mutex_lock(&dm->dc_lock);       Nicholas Kazlauskas, 10 months ago • drm/amd/display: Add fast path for cursor
                WARN_ON(!dc_commit_state(dm->dc, dc_state));
                mutex_unlock(&dm->dc_lock);
        }

        for_each_new_crtc_in_state(state, crtc, new_crtc_state, i) {
                struct amdgpu_crtc *acrtc = to_amdgpu_crtc(crtc);

                dm_new_crtc_state = to_dm_crtc_state(new_crtc_state);

                if (dm_new_crtc_state->stream != NULL) {
                        const struct dc_stream_status *status =
                                        dc_stream_get_status(dm_new_crtc_state->stream);

                        if (!status)
                                status = dc_stream_get_status_from_state(dc_state,
                                                                         dm_new_crtc_state->stream);

                        if (!status)
                                DC_ERR("got no status for stream %p on acrtc%p\n", dm_new_crtc_state->stream, acrtc);
                        else
                                acrtc->otg_inst = status->primary_otg_inst;
                }
        }
#ifdef CONFIG_DRM_AMD_DC_HDCP
        for_each_oldnew_connector_in_state(state, connector, old_con_state, new_con_state, i) {
```

```
Nicholas Kazlauskas, 10 months ago   (Global Scope)   Ln 6457, Col 42   Tab Size: 8   UTF-8   LF   C   Linux
```

```
e9402ff76338 Le Ma              5 days ago    drm/amdgpu:
utoload   (HEAD -> amd-staging-drm-next, origin/amd-staging-
d996535506b2 Hawking Zhang      5 days ago    drm/amdgpu:
e message
ebccb1d746dc Le Ma              5 days ago    drm/amdgpu:
ult on Arcturus
4546b659e464 Le Ma              5 days ago    drm/amdgpu:
oading on Arcturus
73e5bc910eb3 Tianci.Yin         6 days ago    drm/amdgpu/g
ading
57dd82e864f3 Allen Pais         8 days ago    drm/amdkfd:
nce (v2)
4a6c7afe7d1a Alex Deucher       8 days ago    drm/amdgpu/r
for RAS   (origin/adeucher/amd-staging-drm-next, agd5f/amd-s
be1b6827f18d Alex Deucher       8 days ago    drm/amdgpu:
409f299bb0c7 Alex Deucher       8 days ago    drm/amdgpu/i
spatch
8f83cef51bbf Alex Deucher       8 days ago    drm/amdgpu/v
c
2a80ba3c49fe Alex Deucher       8 days ago    drm/amdgpu/m
ad_lock
87fcf8dcf075 Alex Deucher       8 days ago    drm/amdgpu:
e_export
84f675c9e6bb Alex Deucher       9 days ago    drm/amdgpu: drop experimental flag for amd-stagin
g-drm-next
d507ec1eb665 Nicholas Kazlauskas 9 days ago   drm/amdgpu: Bring back support for non-upstream F
reeSync
cfb7c11bb7a5 Joseph Greathouse  9 days ago    drm/amdkfd: Remove GPU ID in GWS queue creation

[hwentlan@hwentlanhp ~/git/promotion/amd-stg]$
```

DP Adaptive Sync, HDMI™ VRR, FreeSync™

What is Adaptive Sync

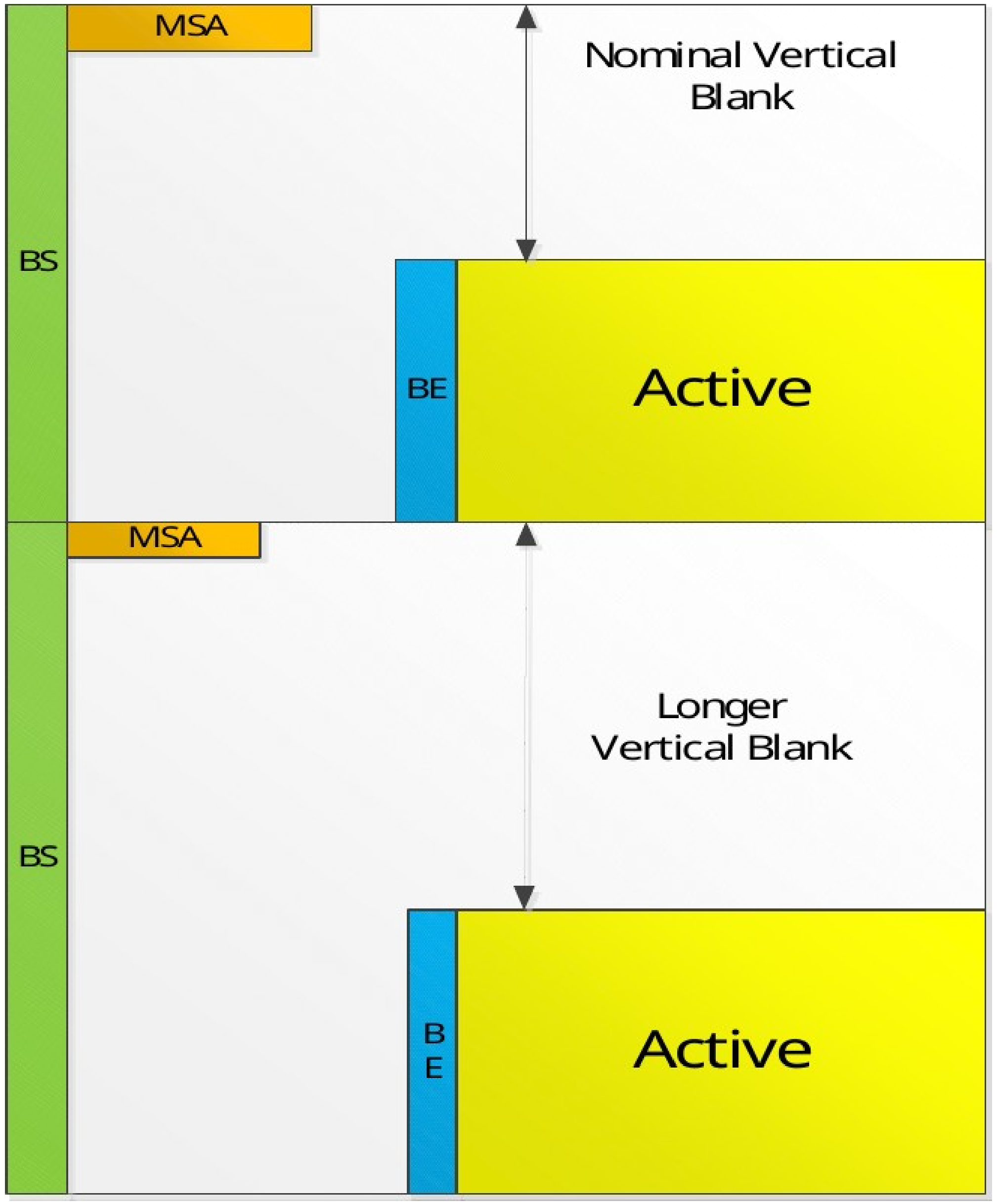
- VESA spec introduced variable refresh rate framework called “ignore MSA” with initial eDP spec
- Rolled out to DP and branded as ‘Adaptive-Sync’ with DP 1.2a in 2014
- Protocol to seamlessly vary framerate by changing blank duration and keeping pixel rate the same
- VESA press release addressed three main use cases
  - Seamless variable frame change for smooth gaming use case
  - Seamless change of frame rate to match video rate for judder free video playback
  - Reduce frame rate for power saving in battery

1) https://vesa.org/featured-articles/vesa-adds-adaptive-sync-to-popular-displayport-video-standard/

How does adaptive sync work?

- Tx (Transmitter) reads range limits from EDID
- When enabling the display
  - Tx writes ignore_msa bit in DPCD
- When user indicates content is suitable for adaptive sync
  - Set up Tx with range limits
  - Tx will extend vertical blank
  - For low latency use case (i.e., gaming) Tx shall end frame immediately once new frame is presented

Adaptive Sync DP Symbols



What is HDMI™ VRR

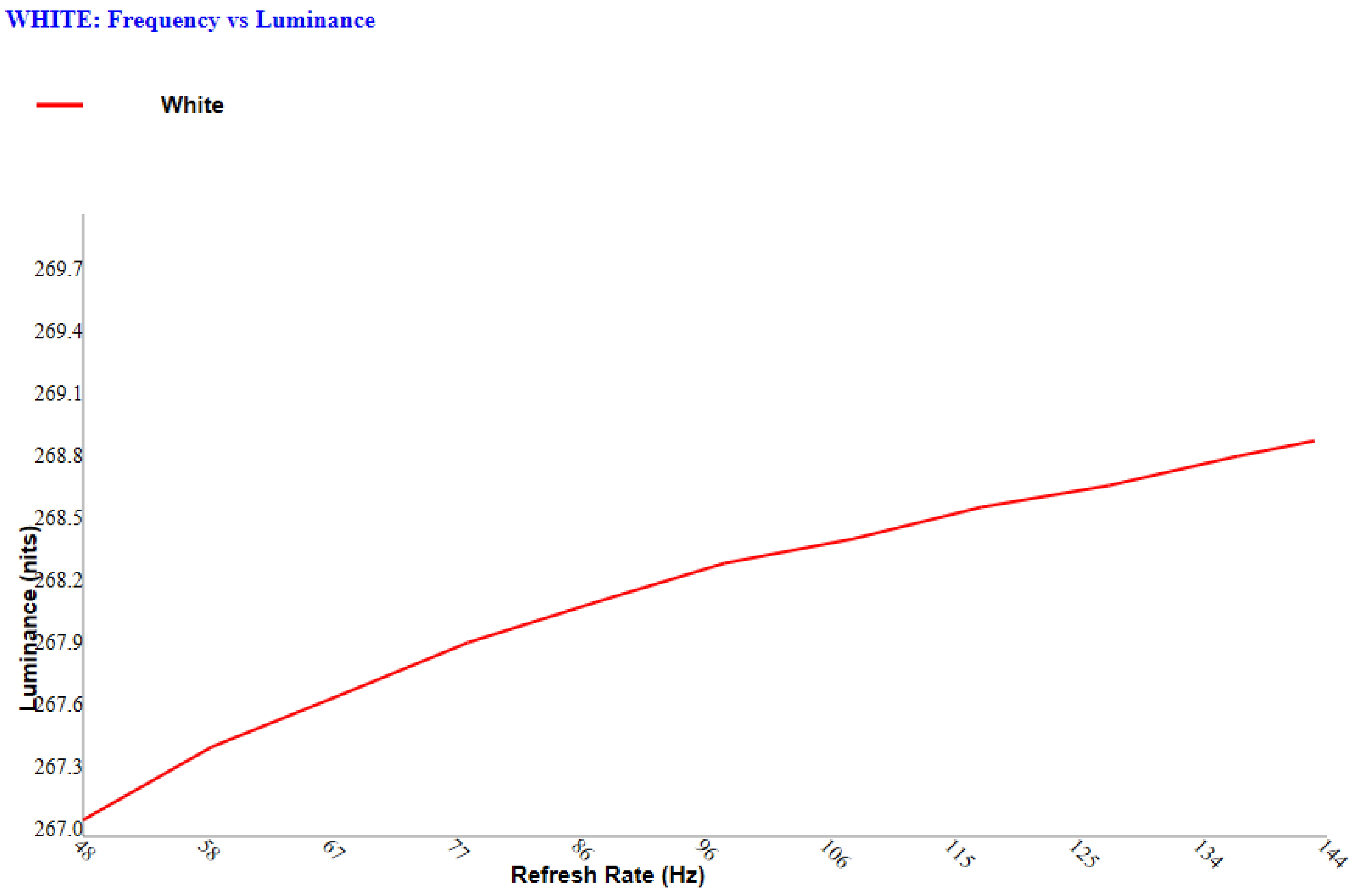
- Part of HDMI™ 2.1 spec
- AMD currently doesn’t enable HDMI™ VRR pending HDMI™ VRR CTS
- AMD enables FreeSync™ on HDMI™ via AMD proprietary protocol (Windows only)

What is FreeSync™

- AMD implementation of adaptive sync and VRR
- DP supported via
  - Adaptive Sync spec
  - Proprietary protocol
- HDMI™ supported via
  - Proprietary protocol
- FreeSync™ certification and logo
- FreeSync™ 2
  - Adaptive sync
  - HDR
  - Stricter certification requirements

Physical Limitations

- Static flicker
  - At very low refresh rates some displays will exhibit flicker due to luminance drop in between frames
- Dynamic flicker
  - When switching between short and long frame durations average brightness changes due to larger luminance drop for longer frames



VRR in DRM & Mesa

DRM/KMS VRR interface

CRTC Property

vrr_enabled

Indicates if variable refresh rate should be enabled for the CRTC. Support for the requested vrr state will depend on driver and hardware capabiltiy - lacking support is not treated as failure.

Connector Property

vrr_capable_property

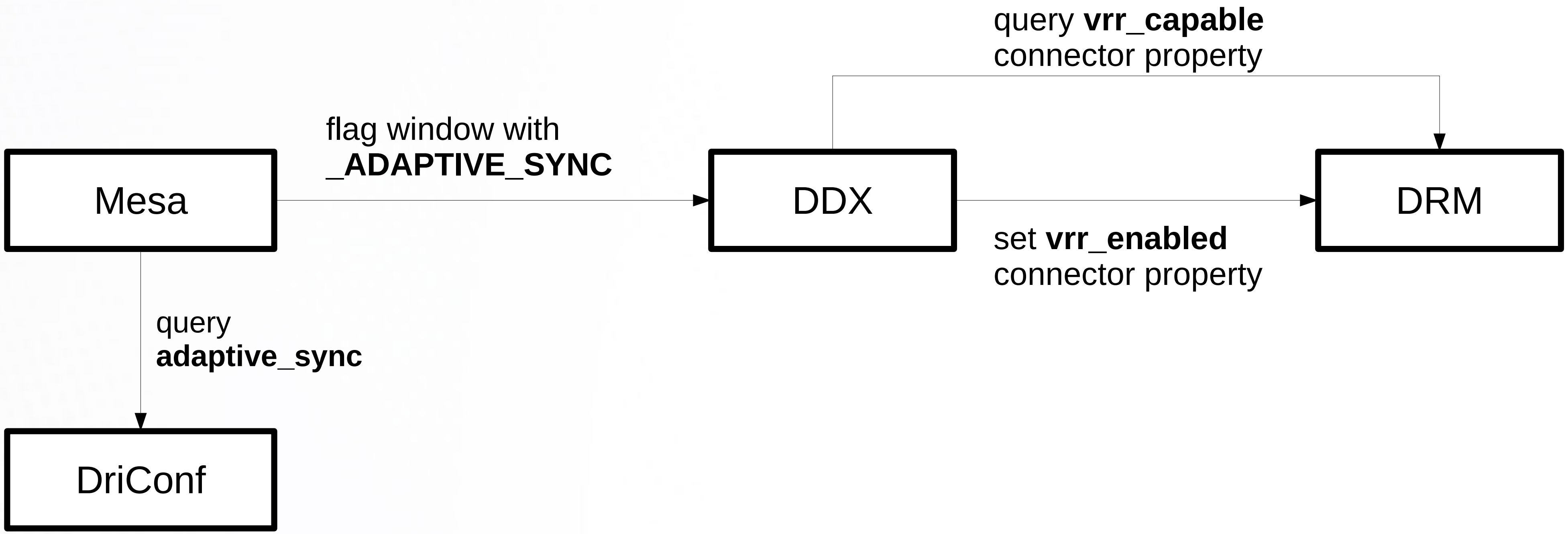
Optional property to help userspace query hardware support for variable refresh rate on a connector. connector. Drivers can add the property to a connector by calling drm_connector_attach_vrr_capable_property().

This should be updated only by calling drm_connector_set_vrr_capable_property().

VRR in Userland (X)

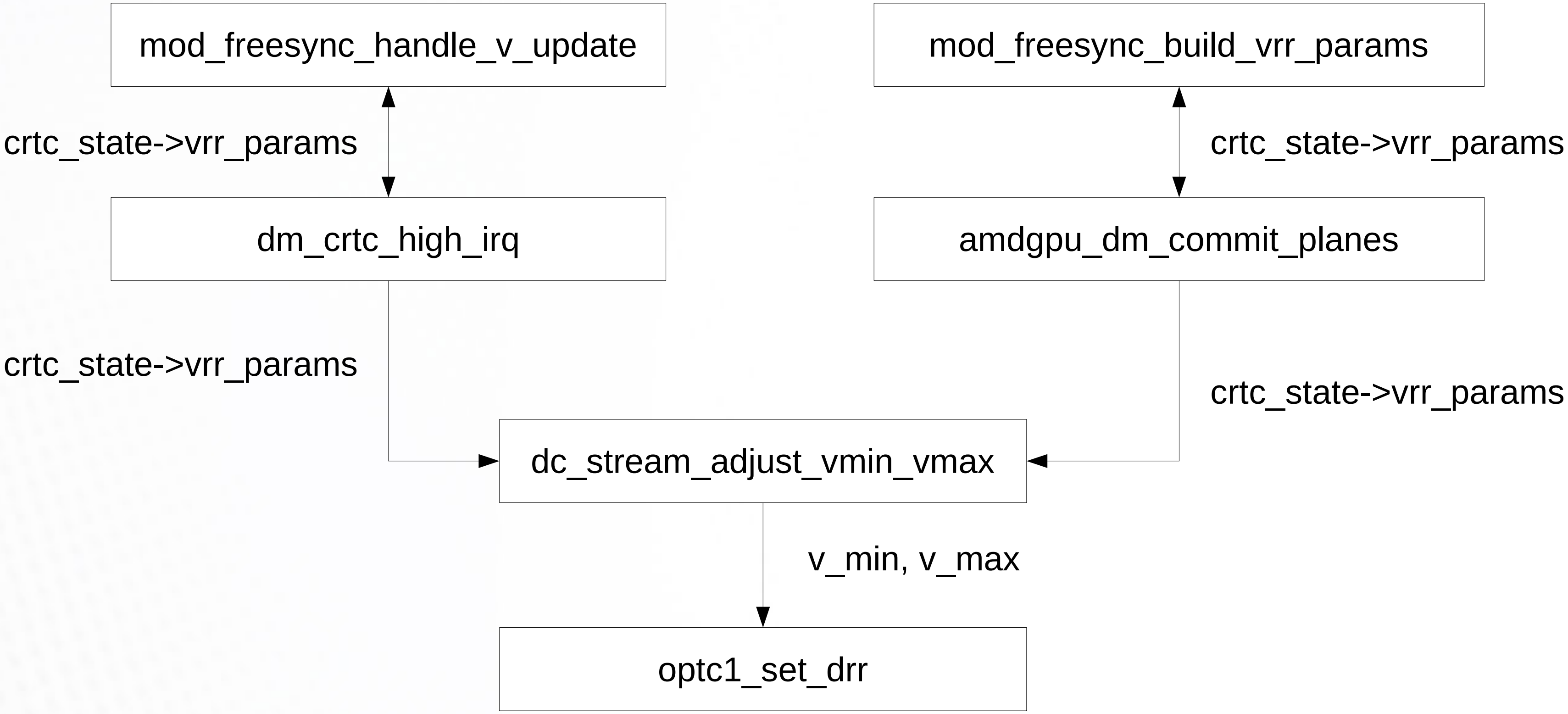
- VRR is supported by
  - radeonsi GL
  - orca GL (proprietary AMD GL driver)
  - radv Vulkan
- A free-running variable refresh rate is not suited for all rendered content, such as current implementations of web browsers, compositors, video players
- Mesa has a blacklist through DriConf for such applications: 00-mesa-defaults.conf
- VRR is enabled for GL/Vulkan rendered applications that use the Present extension and are not blacklisted
- Requires xf86-video-amdgpu
- X doesn’t support present flipping unless the application covers the entire X screen, which means VRR generally won’t enable on multi-monitor setups

VRR with Mesa in X



- Adaptive Sync Patches
  - **Mesa**: https://patchwork.freedesktop.org/series/51388/
  - **xf86-video-amdgpu**: https://gitlab.freedesktop.org/xorg/driver/xf86-video-amdgpu/merge_requests/5
  - **Kernel**: https://patchwork.freedesktop.org/series/49487/

FreeSync™ in DC



Next Steps

Enabling VRR beyond X, what is needed?

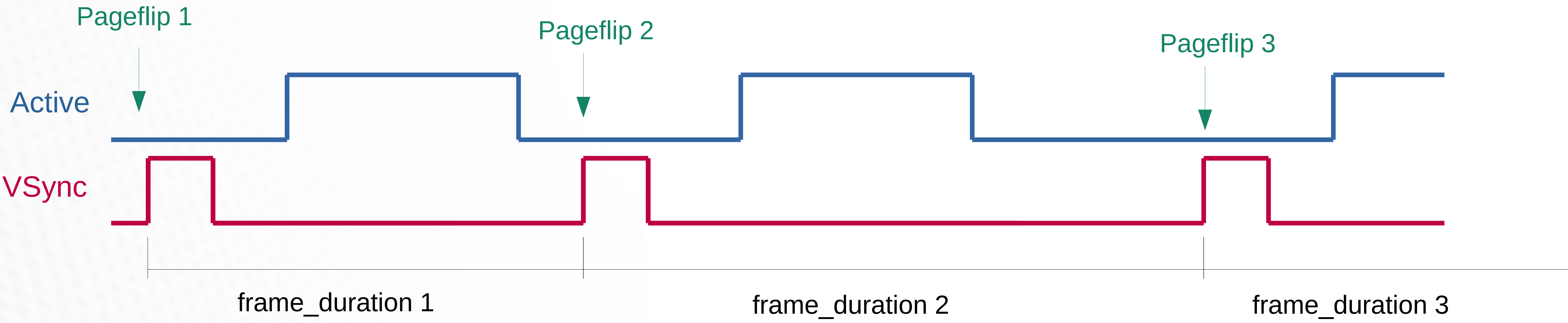
- Currently VRR is only enabled on X
- Good candidates to start adopting VRR would be
  - Wayland
    - Weston
    - Plasma
    - Gnome
  - ChromeOS
  - Etc.
- Looking for community engagement and input here

Enabling more use cases on Linux

- Current solution only covers gaming
- Smooth video playback requires refresh rate to match the content rate
  - With adaptive sync we can dynamically switch the refresh rate without requiring a mode set
- When the desktop is static there is no need to output at full refresh rate
  - Lowering the refresh rate can provide power savings
- Etc…. would love to see what other use cases the community comes up with

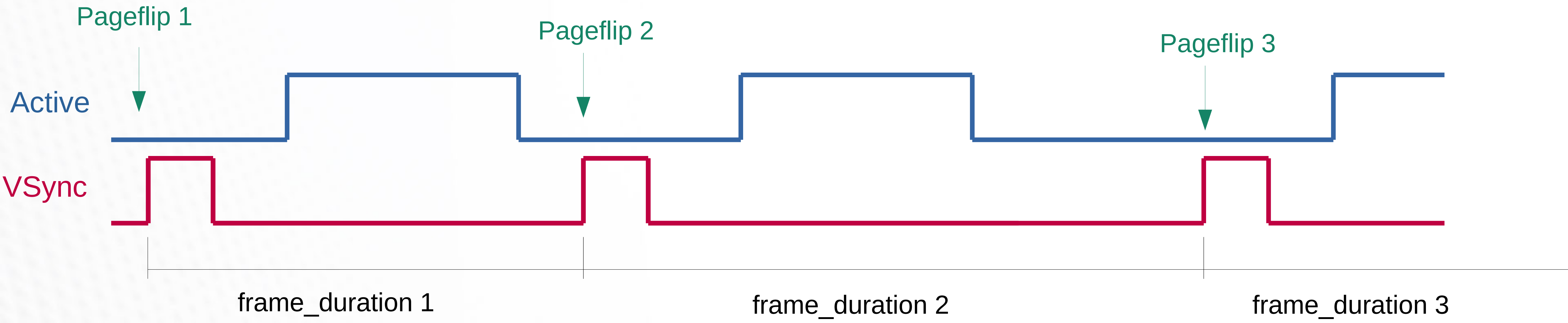
A frame duration time

- DRM/KMS exposes a frame duration time
- If userland provides it the kernel driver will program HW to refresh at rate calculated from frame_duration_time



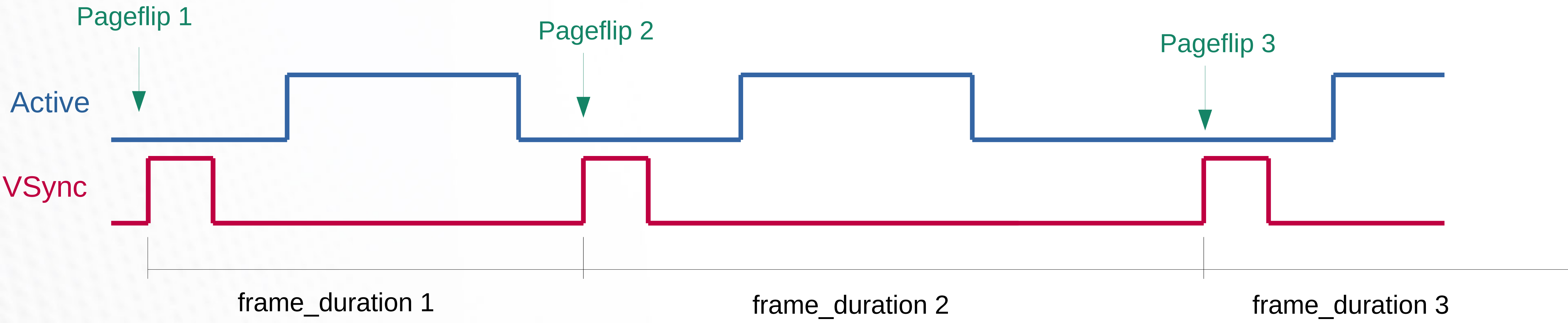
A frame duration time

- Video players can
  - Target the presentation duration, e.g. 1000 / 24 ms
  - Fudge the presentation duration up or down if audio playback drifts
- Compositors can
  - Target a larger presentation time on static screen



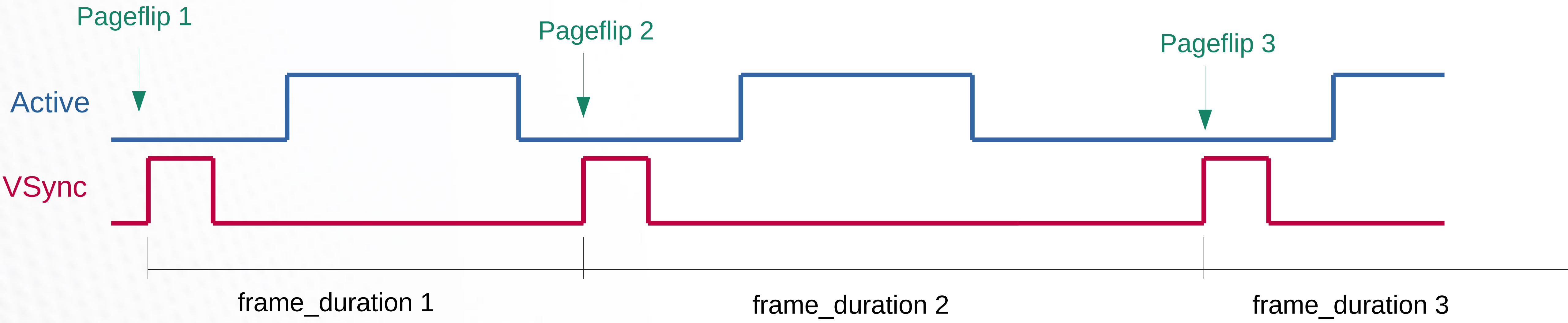
A frame duration time – pros and cons

- Pros
  - Programming length of frame we're submitting → can use HW to adjust frame
  - No need to recalculate frame time every frame for fixed rates
  - No need to calculate frame duration in driver



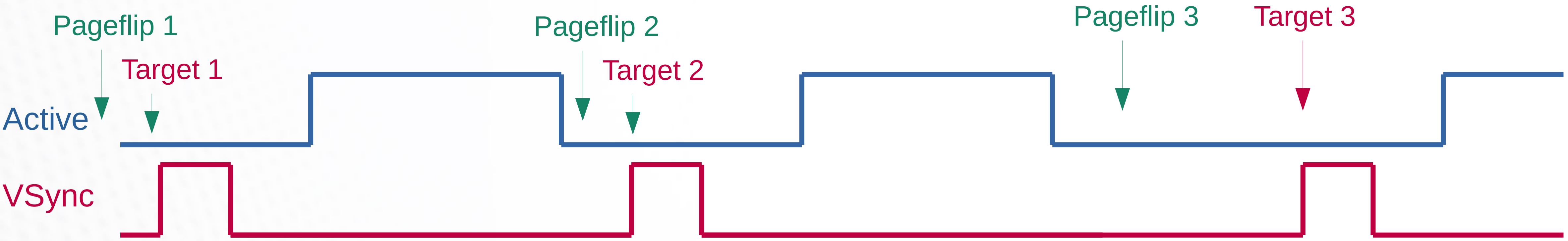
A frame duration time – pros and cons

- Cons
  - Potential for dynamic flicker
  - Userspace has to be aware of frame presentation time (vsync)
  - If flip is programmed too late results won't be as expected



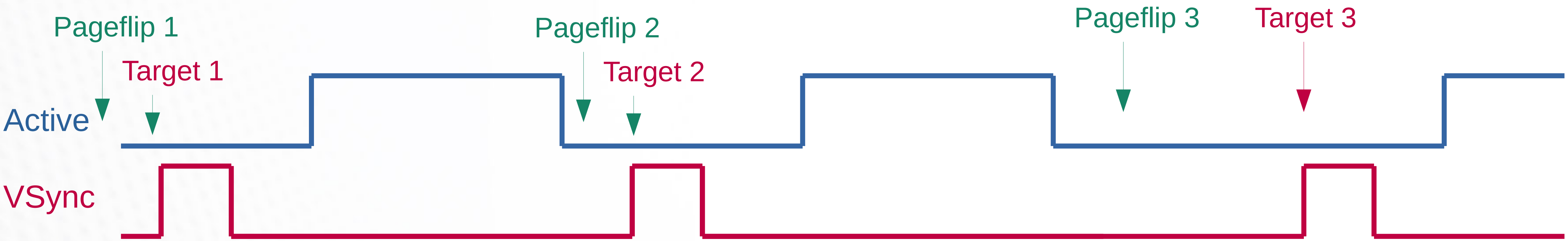
An absolute presentation target

- DRM/KMS exposes an presentation target timestamp
- If userland provides it the kernel driver will program HW in such a way that the start of scanout is no sooner than the timestamp



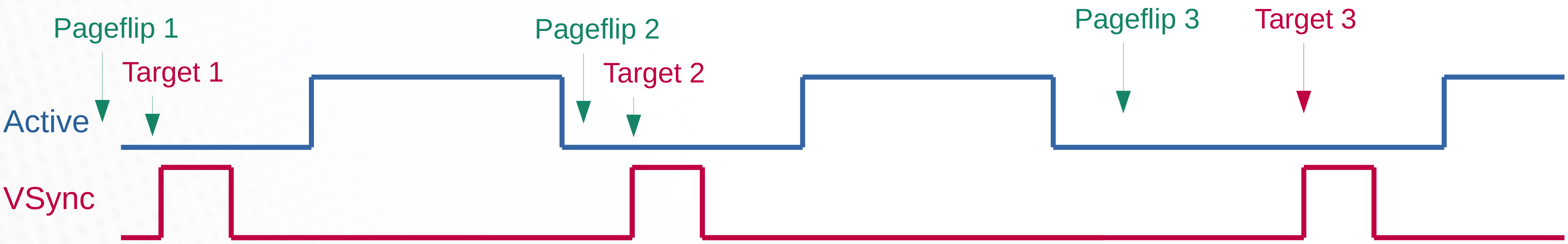
An absolute presentation target

- Video players can
  - Target the presentation time to be current_time_in_ms + (1000/24)
  - Fudge the presentation time up or down if audio playback drifts
- Compositors can
  - Target a larger presentation time on static screen



An absolute presentation target – pros & cons

- Pros
  - Aligns with existing vdpau interface[1] (earliest_presentation_time)
  - Aligns with existing vulkan extension[2] VK_GOOGLE_display_timing
  - Allows SW synchronization of all displays if they all support adaptive sync
  - Userspace doesn't need to be aware of range limits (vmin/vmax) or vsync
  - Might be useful for VR cases

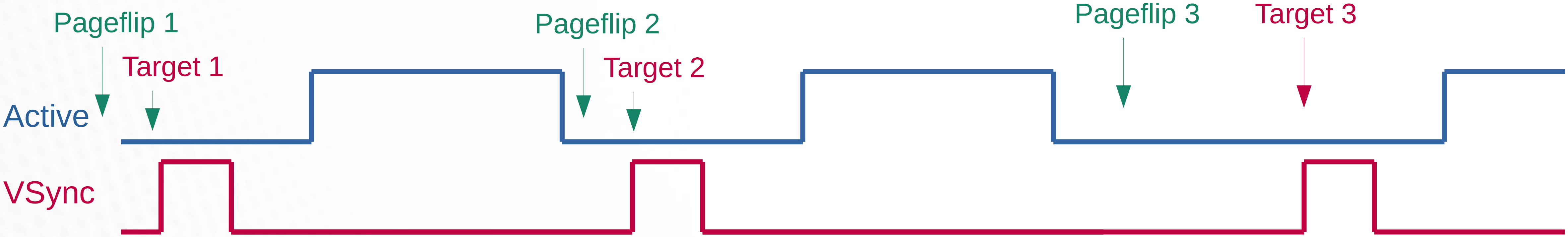


1) https://http.download.nvidia.com/XFree86/vdpau/doxygen/html/group__vdp_presentation_queue.html#ga5bd61ca8ef5d1bc54ca6921aa57f835a

2) https://github.com/KhronosGroup/Vulkan-Docs/blob/master/appendices/VK_GOOGLE_display_timing.txt

An absolute presentation target – pros & cons

- Cons
  - Potential for dynamic flicker
  - Userspace has to calculate new target presentation time with each flip
  - Display can't use HW to target presentation – has to schedule this in SW
  - Need to limit how far in the future presentation target can be



Conclusions

Conclusions

- Dynamic refresh rates greatly improve the gaming experience by reducing
  - Lag
  - Stutter
- There are many displays on the market that support dynamic refresh
- Dynamic/variable refresh rate support is availabe on X
- Wayland compositors still lack support
- A more explicit interface might be useful to enable other use cases

Questions

DISCLAIMER AND ATTRIBUTIONS

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD’s products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18