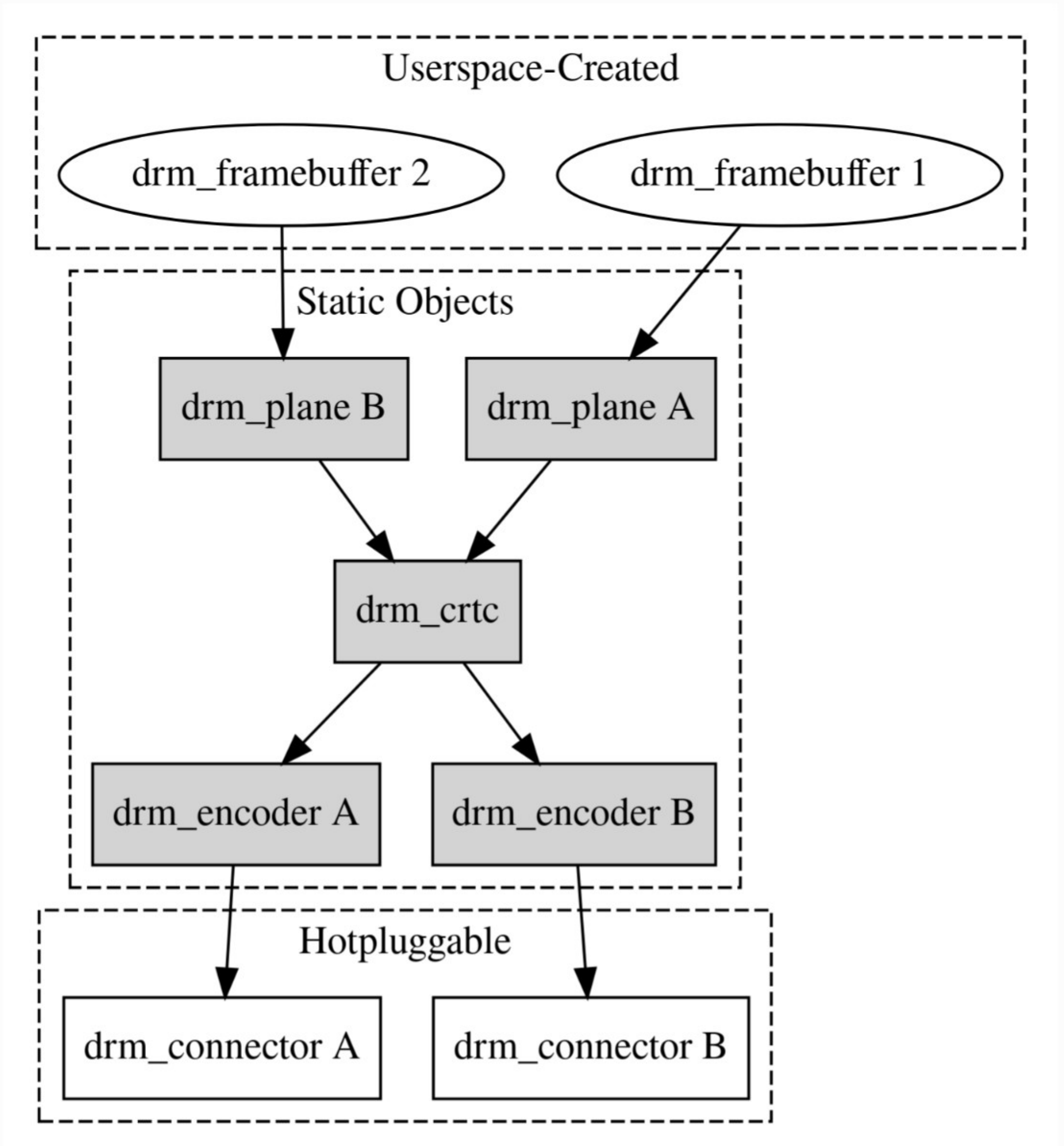
KMS offloading in KWin

Who am I?

- Xaver Hugl
- Work at Techpaladin LLC
- Work on
  - KDE, specifically KWin
  - Wayland Protocols
  - Xwayland, Mesa, Linux, wherever is needed for KWin stuff

KMS: important bits

- Framebuffers
- Planes
  - Primary
  - Cursor
  - Overlay
- Crtcs
- Connectors



Legacy Modesetting

- Individual ioctl for everything!
  - Setting a mode on a CRTC
  - Setting a buffer on a CRTC
  - Setting the cursor image
  - Setting the cursor position
  - Setting a 1D “gamma” LUT on a CRTC

Atomic Modesetting

- Exposes planes, crtcs and connectors
- Each object has a list of properties
- Properties are changed with the ATOMIC ioctl
  - List of properties with the values you’re setting
- TEST_ONLY flag!

KWin in 2020

- Legacy and atomic modesetting support
- Only used the functionality of legacy modesetting
  - Set a buffer on the primary plane
  - Set the cursor using the legacy ioctl!
- KMS offloading: Night light with the “gamma” LUT

Direct scanout

- Works with unobstructed fullscreen windows
- Instead of compositing on the GPU, directly use application buffers for scanout!
- Complication: Buffers need to be suitable for scanout
  - Wayland protocol addition: dmabuf feedback

Hardware rotation

- Rotation in KWin at that point: render into offscreen buffer, then rotate it with a second shader pass
- Plane “rotation” property
- Kernel bug!
- Legacy cursor ioctl worked, but caused atomic commit to fail! → Port cursor to atomic + software fallback
- Later on: Fixed the scene to rotate during compositing

Rotation and scaling for direct scanout

- Game or video with non-native resolution or mismatched rotation
- Plane properties:
  - rotation
  - SRC_X, SRC_Y, SRC_W, SRC_H
  - CRTC_X, CRTC_Y, CRTC_W, CRTC_H

Direct scanout for video

- Needed to change renderer to import YUV buffers
- COLOR_ENCODING and COLOR_RANGE properties to describe YUV properties
- NV12 and P010 supported right now, still need to expand support to other formats
- Special casing for black background

HDR and color management

- Initial implementation in 2023: Linear blending for “correctness”
  - Done with an ICC profile or HDR enabled
  - Rendering in a 16 bits per color float shadow buffer
  - Fullscreen copy to the output swapchain with a second shader pass
- Performance and efficiency suffered
- Solution:
  - Instead of linear, use gamma 2.2 → 10 bits per color is enough
  - With HDR, use KMS gamma LUT to convert gamma 2.2 to PQ

ICC profiles

- Contain nearly arbitrary transformations
- Too complicated for KMS... unless you dumb it down
- Turn every profile into matrix+shaper
  - Apply matrix in normal compositing pass
  - Apply shaper in KMS “gamma” LUT
- Setting: “prefer efficiency” vs. “prefer accuracy”

Direct scanout again: Color offloading

- CRTC provides up to three color operations:
  - DEGAMMA_LUT (1D LUT)
  - CTM (3x3 matrix)
  - GAMMA_LUT (1D LUT)
- Can be used for direct scanout…

Direct scanout again: Color offloading

- CRTC provides up to three color operations:
  - DEGAMMA_LUT (1D LUT)
  - CTM (3x3 matrix)
  - GAMMA_LUT (1D LUT)
- Can be used for direct scanout… or so I thought
  - Nvidia: degamma doesn’t apply to the cursor
  - Intel: degamma sometimes has terrible resolution
  - AMD: glitches when changing the LUTs

More than fullscreen direct scanout

- KMS supports multiple plane types per CRTC, we only use two:
  - Primary
  - Cursor
- Preparations for overlay planes:
  - Backend API to expose the overlay planes
  - Repaint scheduling
  - Scene refactors
  - Treat the cursor like an overlay too

Overlay plane strategy

- Drm backend provides overlays (only on single display setups)
- Scene picks up to n items:
  - Not occluded
  - Update quickly
  - If there’s too many, no overlays get chosen!
- Compositor assigns them to planes
- ***One*** atomic test for primary+cursor+overlays
  - If that fails, ***one*** atomic test for primary+cursor
  - If even that fails, fall back to only the primary plane

Problems with overlays

- amdgpu sometimes takes ages for atomic tests
  - With CPU load, I saw it taking 130ms once!
- amdgpu has pageflip timeouts on my laptop
- nvidia-drm causes system freezes
- i915 seems to work fine
- For now, only enabled in development versions of Kwin
- Set KWIN_USE_OVERLAYS=1 environment variable to enable overlay usage in Plasma 6.5

Underlay strategy

- Overlays go above the composited scene, underlays go below it
- Big advantages:
  - Underlays can have other things on top of them, like subtitles!
  - Underlays can also have compositor-side rounded corners
- Implementation:
  - Compositor paints a transparent hole into the scene, where the item would normally be
  - Only works in Plasma 6.5 if overlay plane supports zpos < primary zpos
  - Putting scene on overlay plane instead will be in Plasma 6.6

Overlays with colors

- Lots of things prevent overlay usage, like night light, color profiles, HDR, tonemapping
- New COLOR_PIPELINE API will fix it, allows mapping KWin’s color operations to hardware capabilities
- KWin implementation is waiting for the kernel side, please merge it already!

What’s next?

- Make use of this in applications!
  - Pass hardware decoded video buffers to compositor directly
  - Put stuff on Wayland subsurfaces
- Futher improve power usage with this
  - Video playback on my laptop: 7.3W vs. 8.4W
  - Optimize hardware decoding in applications and drivers
  - Change the CPU power profile for video playback?
- Fix all the driver bugs pls.