HDR stack enablement in Gnome Mutter

Authors
Naveen Kumar <naveen1.kumar@intel.com>
Uma Shankar <uma.shankar@intel.com>
Agenda

- HDR: Introduction
- Types of HDR - HDR10 vs HDR10+
- HDR Metadata & Transfer functions
- Color Transformation & Tone mapping
- Functional Flow of HDR in Gnome Mutter
- Key design consideration
- Implementation plan
- Future of HDR on Gnome Mutter
- References
**HDR Basics**

- It is a technique to reproduce greater dynamic range of luminosity.
- Bright's are brighter; darks are darker.
- It increases the range and granularity of luminance i.e., from very dark values (0.00005 nits) to very bright values (10000 nits).
- Standard Dynamic Range (SDR) (pre-HDR displays): max ~100 nits.
- HDR defines up to max 10000 nits.
- 2 primary HDR standards are:
  - HDR10: Static HDR
  - HDR10+: Dynamic HDR
Static HDR vs Dynamic HDR

- Static HDR uses a single image descriptor in metadata that is a compromise that applies to every scene and every frame of the whole movie.
- Dynamic HDR enables a noticeable progression in overall video image quality from SDR to static, and now static HDR to dynamic HDR.
- Dynamic HDR ensures every moment of a video is displayed at its ideal value for depth, detail, brightness, contrast and wider color gamut's - on a scene by scene or even a frame-by-frame basis.
HDR Metadata & Transfer Functions

- HDR metadata
  - $x,y$ chromaticity coordinates for color primaries and white point (i.e., color gamut).
  - Maximum luminance (in cd/m$^2$).
  - Minimum luminance (in cd/m$^2$).

- Transfer functions
  - Electro-Optical Transfer Function (EOTF):
    - Defines how display should convert non-linear digital signal to linear light values.
    - sRGB is the defacto EOTF for SDR.
    - Two common HDR EOTFs:
      - SMPTE 2084: Perceptual Quantizer (PQ).
      - Hybrid-Log (HLG).
    - To create digital signal, GPU needs to do inverse of EOTF (aka “OETF”).
  - OETF is used by image sources (camera, graphics processor) where as EOTF is used by sink devices such as a display panel. The following equations describe OETF and EOTF mathematically. The figures compare OETF and EOTF with gamma encoding and decoding.

```plaintext
double m1 = 0.1593017578125;
double m2 = 78.84375;
double c1 = 0.8359375;
double c2 = 18.8515625;

// input within [0.0, 1.0] range.
double oetf_output = pow(((c1 + (c2 * pow(input, m1))) / (1 + (c3 * pow(input, m1)))), m2);

double eotf_output = pow(((max((pow(input, (1.0 / m2)) - c1), 0)) / (c2 - (c3 * pow(input, (1.0 / m2))))), (1.0 / m1)); // input within [0.0, 1.0] range.
```
Color Transformation & Tone mapping

- Color transformation:
  - It is to convert respective layers from one colorspace/format to a common colorspace/format for blending

- Tone Mapping:
  - It is a mechanism to map content luminance to panel luminance or, to one common luminance level for blending.

- Compositor needs to define a blending policy.

- Based on the hardware capabilities, either use
  - Display Engine
  - Media Engine
  - GPU

- Tone mapping decisions will be based on below usecases:
  - HDR to HDR
  - SDR to HDR
  - HDR to SDR
Functional Flow of HDR in Gnome Mutter
Key design consideration

- How client parses the HDR metadata from content
  - Client can directly create the metadata based on the input rendering content
  - It can parse the HDR metadata with the help of FFMPEG/Gstreamer media frame
- Wayland-protocols for HDR
  - To pass the hdr metadata to compositor
- Wayland Server side implementation for HDR in Gnome Mutter
  - This will receive the metadata from client and will help in making blending decisions
- Mutter/KMS Interface
  - Based on the blending policies, the final metadata is converted into DRM blob and passed to driver through KMS interface
- Media engine/Vaapi interface - [HDR vaapi/libva interface](#)
  - Tone mapping can also be done using Media engine. Compositor can choose based on hardware capability & related policies
- 3D GL/Vulkan shaders implementations
  - Tone mapping can also be done using GPU. Compositor can choose based on hardware capability & related policies
Client HDR metadata parsing

- FFMPEG/Gstreamer Media framework is used for content demuxing|decoding|metadata-parsing

- Reference implementation:
  clients/simple-hdr-video-gbm.c · test-hdr · Naveen Kumar / weston · GitLab
Wayland-protocols Interface

- Wayland-protocols specifies the communication between a display server and its clients
- Reference implementation:

```
unstable/hdr-metadata/hdr-metadata-unstable-v1.xml · test-hdr · Naveen Kumar / wayland-protocols · GitLab
```
Wayland Server Metadata handling

- Wayland Server side implementation for HDR in Gnome Mutter
- Reference implementation: src/wayland/meta-wayland-hdr.c

```c
/* Implements the protocol function set metadata */
static void
data_set_metadata(struct wl_client *client,
struct wl_resource *surface_resource,
uint32_t primary_l, uint32_t primary_r,
uint32_t primary_g, uint32_t primary_b,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32_t max하신cance,
uint32_t max_luminance, uint32 т maxGeneratedValue,
uint32_t max_luminance, uint32 t maxGeneratedValue,
uint32_t max_luminance, uint32 t maxGeneratedValue,
uint32_t max_luminance, uint32 t maxGeneratedValue,
uint32_t max_luminance, uint32 t maxGeneratedValue,
uint32_t max_luminance, uint32 t maxGeneratedValue,
uint32_t max_luminance, uint32 t maxGeneratedValue,
uint32_t max_luminance, uint32 t maxGeneratedValue,
uint32_t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
uint32 t max_luminance, uint32 t maxGeneratedValue,
Mutter-KMS interface

- Metadata handling & property interface with KMS
- Reference implementation:

  `src/backends/native/meta-kms-impl-device-atomic.c`
Implementation plan

- KMS HDR support – Latest upstream kernel supports HDR
  - It basically defines HDR metadata structures, property to pass content (after blending) metadata from user space compositors to driver.
  - DRM HDR Metadata property (hdr_metadata_property) - This is used to pass HDR metadata information from user space to driver. Driver will use this and create the DRM Infoframe packet and send to HDMI panel.
  - For details - HDR support in DRM

- Userspace/Compositors HDR support - Blending policies and metadata blob creation and passing to driver. [WIP] The implementation of HDR userspace development is planned in following steps:
  - clutter: Attach color state information to actors (I2443) - Merge requests - GNOME / mutter - GitLab
  - cogl/texture: Add higher bit depth offscreen framebuffers support (I2461) - Merge requests - GNOME / mutter - GitLab - Add 10bit (higher bit depth) pixel formats support
  - Draft: [WIP] Attach shaders to cogl pipelines when painting actors with color state (I2643) - Merge requests - GNOME / mutter - GitLab - Attach shaders to clutter actor using cogl pipeline
  - Colorspace conversion using shaders
  - Send HDR metadata using KMS properties
  - Tone mapping

- Wayland-protocols for HDR - Commits - test-hdr - Naveen Kumar / wayland-protocols - GitLab

- Wayland HDR client application - Commits - test-hdr - Naveen Kumar / weston - GitLab
A POC to implement [HDR on Gnome Mutter](#) is available. Details below

- This implements a full screen direct scanout without tone mapping or color conversion.
- It creates an E2E HDR pipeline and exercises DRM KMS interfaces
- It implements a reference wayland-protocols for handling HDR metadata across client & mutter
- It also parses EDID to get the sink’s HDR capability

**Steps To Test HDR Fullscreen Direct Scanout**
- Compile Weston for HDR video client app [Weston HDR video client app](#)
- Compile Wayland-protocols for HDR, CM [Wayland-protocols for HDR, CM](#)
- Compile Mutter with HDR metadata passthrough support: this MR ![2356](#)
- Test HDR video playback with following command line:
  - `$ ./weston-simple-hdr-video-gbm -i <path_video_file> -f 1 -w 3840 -h 2160 -p P010`
Future of HDR on Gnome Mutter

- HDR10+ (Dynamic HDR)
- Multiplane overlay support can be enabled
- Plane level color properties are proposed for compositors to display operations per surface. So the composition can directly happen on display hardware
- YUV support
References

- **Mutter**
  - Basic support for presenting HDR content from Wayland clients (#2134) · Issues · GNOME / mutter · GitLab
  - Draft: HDR support in Mutter (!2356) · Merge requests · GNOME / mutter · GitLab

- **Wayland-protocols for HDR reference only** - Commits · test-hdr · Naveen Kumar / wayland-protocols · GitLab

- **HDR client application** - Commits · test-hdr · Naveen Kumar / weston · GitLab

- **KMS** - Latest upstream kernel supports HDR

Thank You