Linux and High Dynamic Range (HDR) Display

XDC 2016

Andy Ritger

NVIDIA

Linux and HDR Display

Disclaimer

- I am not a color expert.
- HDR is a broad topic.
- NVIDIA has not yet implemented HDR support in our Linux driver, though we have on Windows and Android.
- Goal today is to:
  - Give an overview of the building blocks of HDR from a window system perspective.
  - Raise awareness of some of the areas of the Linux ecosystem that may require updating for HDR.

Linux and HDR Display

Ultra High Definition (UHD) Displays

Next generation displays aim to produce more realistic images.

- Higher pixel resolution ("4k" and "8k").
- Wider color gamut: express a wider range of colors than today.
- High Dynamic Range (HDR): express a wider range of luminance than today.

Linux and HDR Display

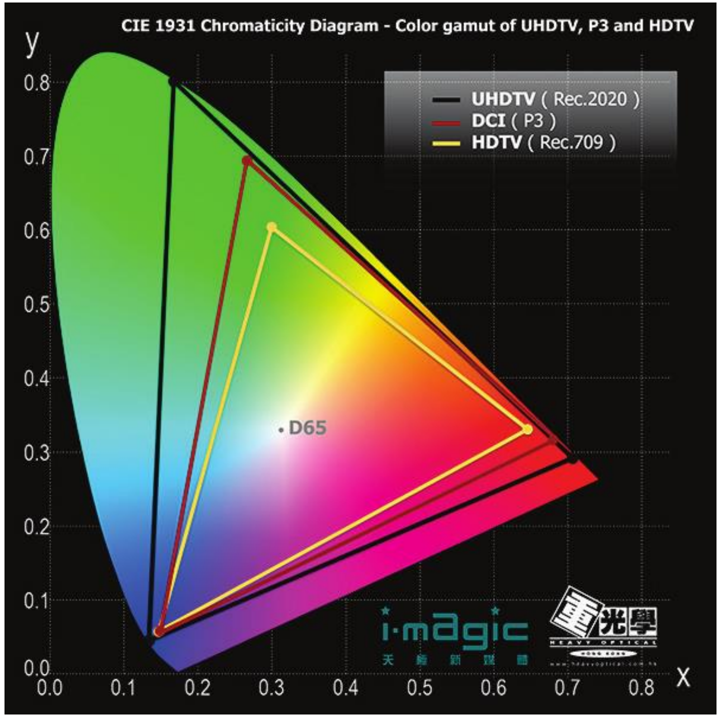
Background: ITU-R BT.2020

- ITU-R BT.2020 (aka "Rec. 2020" or "BT.2020") recommends UHD parameters.
- This includes recommendations for:
    - Resolutions.
    - Refresh rates.
    - Chromaticity.
    - Signal formats (RGB 4:4:4, YCbCr 4:2:0, etc).
    - Digital representation (10- or 12-bit).
    - Transfer function.
- Often, when people say "BT.2020", they mean the BT.2020 color gamut.
- BT.2020’s color gamut is the goal; very few displays get close to that today.
- Current generation UHD displays are much closer to the DCI-P3 color gamut.

Linux and HDR Display

Background: Chromaticity

- Different color spaces represent different sets of colors (color gamuts).
- CIE XYZ color space:
  - Color primaries X, Y, Z.
  - Y == luminance.
- CIE xy chromaticity diagram: 2D projection of 3D CIE XYZ color space.
- Other color spaces described by x,y coordinates within CIE xy chromaticity diagram:
  - x,y location of the red, green, and blue primaries.
  - x,y location of the “white point”.



Linux and HDR Display

Background: Linear versus non-Linear Color Spaces

- Linear color space is a color representation where:
  - There is linear relationship between numbers stored and intensities they represent.
  - E.g., doubling the stored number doubles the represented intensity.
  - Always do graphics operations (blending, scaling, etc) in linear color space.
- However, human perception is not linear: more sensitive to darks than lights.
- Given finite discrete steps (e.g., 0-255), linear isn't great:
  - Insufficient granularity in the darks; wasted precision in the lights.
  - So, generally recommended to store colors in non-linear color space.
  - The most ubiquitous non-linear color space is sRGB.
- Most (pre-HDR) monitors expect their input to be sRGB.

Linux and HDR Display

What is HDR?

Informally:

- Brights are brighter; darks are darker.
- Details are more perceivable in both darks and brights.

More formally:

- HDR increases the range and granularity of luminance.
- Luminance is a measurement of intensity over area.
    - Unit is: candela per square meter (cd/m^2), aka "nit".
    - Standard Dynamic Range (SDR) (i.e., pre-HDR displays): max ~100 nits.
    - First generation HDR displays max ~1000 nits; HDR defines up to max 10000 nits.

HDR is about allowing highlights to be brighter, not making entire scene brighter.

Linux and HDR Display

HDR *Rendering* is Not New

Many 3D graphics applications already use HDR for rendering.

- Create an FP16 (aka half-float, 16-bits per component) buffer.
- Render in FP16.
- At the end of the pipeline, “tone-map” from FP16 to lower-precision, lower luminance, e.g., RGBA8 SDR, for display.
- Now that we have more capable displays, we want to:
  - Give applications the information they need to tone map for the target HDR display.
  - Be able to pass the application’s higher-precision content through to the display.

Linux and HDR Display

HDR Basic Flow

The basic flow for 3D applications to render and display HDR:

Application

- Create FP16 buffer for rendering.
- Render to FP16, using scRGB color space.
- Tone map the scRGB FP16 content, for the target monitor's capabilities.
- Provide metadata to be sent to monitor.

Driver (or composite manager)

- Composite with SDR content.

Driver/GPU

- Receive scRGB FP16 image.
- Perform inverse EOTF to encode FP16 in display signal.
- Send encoded display signal and HDR InfoFrame to monitor.

Monitor

- Receive HDR InfoFrame.
- Perform EOTF to decode digital signal into HDR content.

Linux and HDR Display

scRGB

- Aka *Canonical Compositing Color Space (CCCS)*.
- Has the same chromaticity coordinates as sRGB.
- Has a linear, FP16 encoding.
- (0.0, 1.0) corresponds to the traditional sRGB colors.
- Can have values outside of (0.0, 1.0).
- Values above or below (0.0, 1.0) extend the color range.
- Is absolute, rather than relative:
  - Relative: 0.0==darkest the monitor can display, 1.0==brightest the monitor can display.
  - Absolute: 1.0==80nits, 12.5==1000nits.

These properties make scRGB good for representing HDR content, and also good for compositing HDR content with SDR content.

Linux and HDR Display

HDR Metadata: SMPTE 2086

SMPTE 2086 defines HDR-related metadata passed between GPU and monitor:

- 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).

The GPU needs this metadata from monitor, to know how to render image.

The monitor needs this metadata from GPU, to know how to interpret signal.

CEA-861-3 defines how HDR metadata is transferred:

- Encoded in EDID (Display => GPU).
- Encoded in InfoFrame (GPU => Display).

Linux and HDR Display

HDR Transfer Functions

Electro-Optical Transfer Function (EOTF):

- Defines how display should convert non-linear digital signal to linear light values.
- Optimized for bandwidth: compress signal into as few bits as possible, sacrificing precision where it won't be missed.
- 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").

Linux and HDR Display

HDR Basic Flow (again)

The basic flow for 3D applications to render and display HDR:

Application	• Create FP16 buffer for rendering. • Render to FP16, using scRGB color space. • Tone map the scRGB FP16 content, for the target monitor's capabilities. • Provide metadata to be sent to monitor.
Driver (or composite manager)	• Composite with SDR content.
Driver/GPU	• Receive scRGB FP16 image. • Perform inverse EOTF to encode FP16 in display signal. • Send encoded display signal and HDR InfoFrame to monitor.
Monitor	• Receive HDR InfoFrame. • Perform EOTF to decode digital signal into HDR content.

Linux and HDR Display

Missing Pieces on Linux: HDR Metadata

- An API for getting HDR metadata from display.
  - Maybe this should just be client-side library that parses the EDID.
  - Maybe this should be reported by drivers… depends on whether drivers will need to influence the metadata.
  - Potentially composite managers would need to influence metadata, too?

- An API for applications to express the HDR metadata for the buffers they present.
  - Consider arbitration: multiple applications could provide input that influences metadata.
  - Perhaps composite managers would be involved in metadata arbitration?

Linux and HDR Display

Missing Pieces on Linux: Displaying FP16

- Submitting FP16 to display engine.
    - Eventually display hardware will do FP16 => digital signal (using inverse EOTF).
    - Until then, use shader.

- Would be nice if current SDR RGBA8 windows can coexist in the same desktop as FP16 HDR windows.
    - scRGB defines a good way for the two to be composited together.

Linux and HDR Display

Missing Pieces on Linux: Displaying FP16 In Wayland

- Wayland compositors will need to be FP16-aware.
- Be able to accept FP16 buffers from clients.
- Be able to composite SDR RGBA8 with HDR FP16 into an FP16 buffer.
- Be able to flip to FP16 buffer.

Linux and HDR Display

Missing Pieces on Linux: Displaying FP16 In X11

- Primary producers of HDR content will be:
    - 3D APIs (OpenGL, Vulkan).
    - Video APIs (VDPAU, VAAPI).
- Don't *need* X rendering to FP16, but easier to prohibit or allow?
- Or make FP16 look like SDR RGBA8 to X rendering (hiding RGBA8/FP16 conversions in drivers)?
- Should we allow the root window to be FP16?
- Or add an FP16 overlay visual (like traditional 8-bit color index overlay)?
    - When FP16 window is unredirected, driver's job to composite HDR FP16 with SDR RGBA8.
    - When FP16 window is redirected, composite manager's job to composite HDR FP16 with SDR RGBA8.
    - Composite managers create FP16 child window of Composite Overlay Window to display HDR FP16.

Linux and HDR Display

Conclusion

- Hopefully these slides give context to help understand HDR documentation.
- There are several open design questions for enabling HDR on Linux.
- Over coming months, we'll try to make more concrete strawman proposals.
- Questions or feedback: aritger 'at' nvidia.com

Linux and HDR Display

Links, Further Reading

HDR:

- http://files.spectracal.com/Documents/White%20Papers/HDR_Demystified.pdf
- https://developer.nvidia.com/sites/default/files/akamai/gameworks/hdr/UHDColorForGames.pdf
- https://developer.nvidia.com/high-dynamic-range-display-development

Linear RGB vs sRGB:

- http://stackoverflow.com/questions/12524623/what-are-the-practical-differences-when-working-with-colors-in-a-linear-vs-a-no
- http://www.4p8.com/eric.brasseur/gamma.html
- http://http.developer.nvidia.com/GPUGems3/gpugems3_ch24.html

Linux and HDR Display

Specifications

- ITU-R BT.2020
  - Defines numerous UHD parameters.
  - http://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.2020-2-201510-I!!PDF-E.pdf
- SMPTE 2086
  - Defines HDR metadata.
  - http://ieeexplore.ieee.org/document/7291707/
- SMPTE 2084
  - Defines the Perceptual Quantizer (PQ), a particular EOTF that is particularly efficient at encoding HDR content.
  - http://ieeexplore.ieee.org/document/7291452/
- CEA-861-3
  - Defines how SMPTE 2086-defined metadata should be encoded in EDID and InfoFrame.
  - http://shop.standards.ie/nsai/details.aspx?ProductID=1792152