AMD

Is HDR Harder?

Harry Wentland

Agenda

- Why do we want HDR?
- HDR Concepts
- HDR on Linux
- Conclusion

Disclaimer

- I'm not a color expert – but I've learned a lot during the last couple of years

- This presentation will:
  - Give overview of key HDR concepts
  - Touch on the state of HDR on Wayland, Weston
  - Talk about a DRM/KMS API proposal

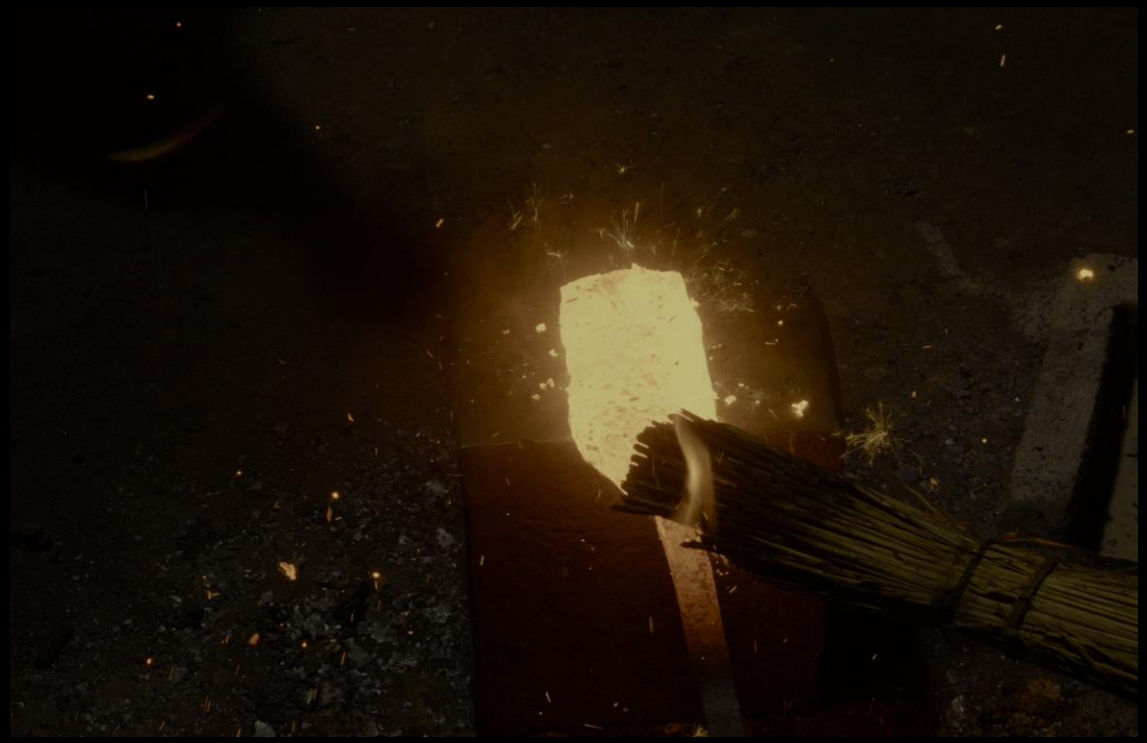
- This presentation will not:
  - Go into details on HDR composition
  - Go into details on previous DRM/KMS API proposals

AMD

Why do we want HDR?

... and what do we want?

What if we could have ...



these details **with** this color saturation **?!**

What if we could have ...

these details **with** this color saturation **?!**

What if we could have ...

these details **with** this color saturation **?!**

Where dark areas aren't lost

While brights aren't washed out

As well as highlights that aren't dull

Why do we want HDR?

- Ability to display content with
  - Larger dynamic range
  - Wider color gamuts
- Ability to combine content with different dynamic ranges or color gamuts

- HDR does NOT mean
  - Everything is brighter
  - Content always preserves detail in darks and brights
- HDR allows content creators to make those creative decisions, but
- HDR gives content creators a larger palette to work with

AMD

HDR Concepts

Color Space – Primaries & Whitepoint

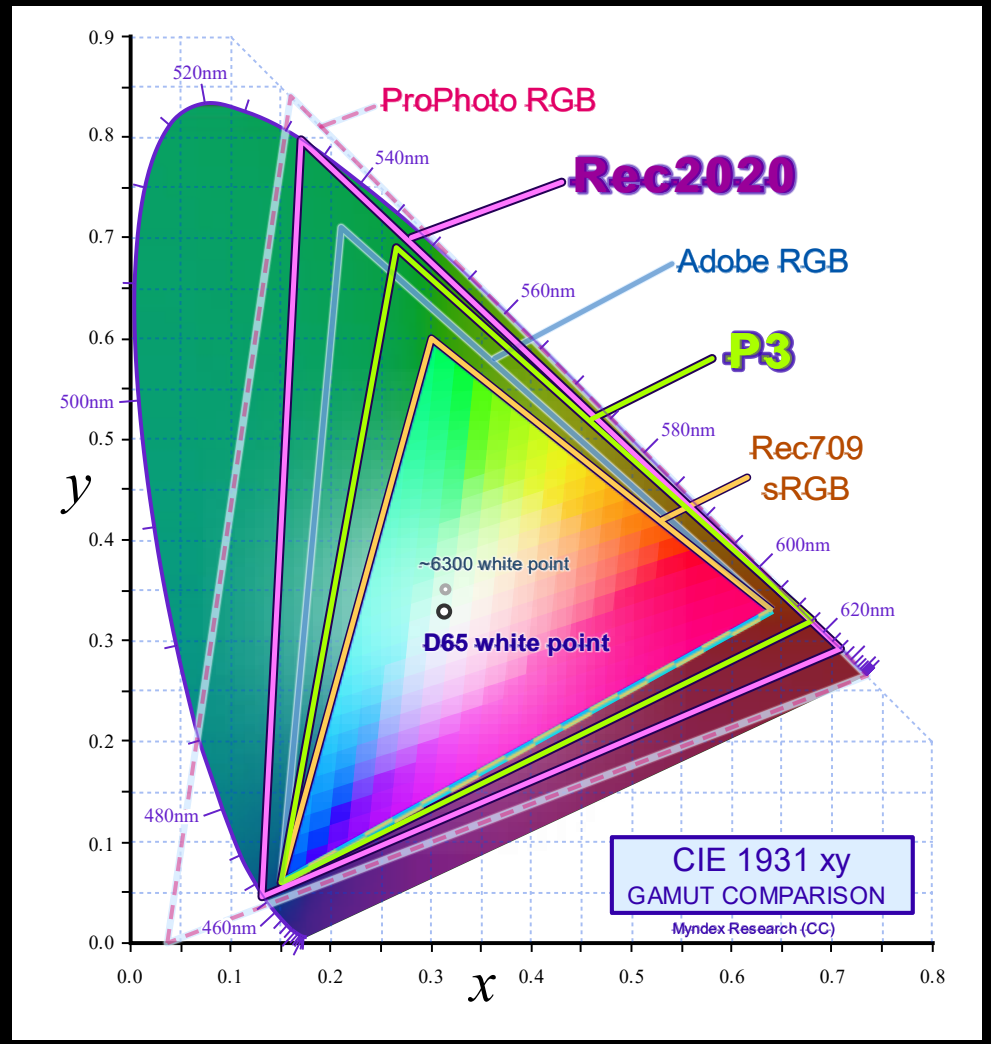
- Color space defines the extent of colors that can be represented
- Primaries define the RGB colors
  - e.g., the red primaries defines what RGB (1.0, 0.0, 0.0) means
- White point defines what point is considered white,
  - I.e., what RGB (1.0, 1.0, 1.0) refers to

- SDR content generally uses
  - sRGB / BT.709
  - BT.601
- HDR content often uses
  - BT.2020
  - DCI-P3
  - sRGB (for gaming)

RGB color space parameters[7]

Color space	White point		Primaries					
	x_W	y_W	x_R	y_R	x_G	y_G	x_B	y_B
ITU-R BT.709	0.3127	0.3290	0.64	0.33	0.30	0.60	0.15	0.06

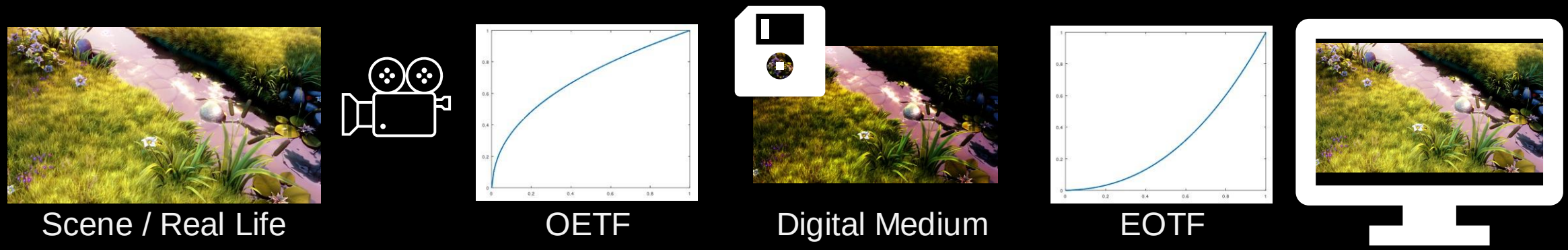
https://en.wikipedia.org/wiki/Rec._709



CIE 1931 xy GAMUT COMPARISON by Myndex / CC BY-SA 4.0

Transfer Functions

- OETF - opto-electronic transfer function
  - Converts scene light to an electric signal
- EOTF – electro-optical transfer function
  - Converts an electrical signal into the linear light output of a display
- OOTF – opto-optical transfer function
  - Used to convert scene light to displayed light
  - OETF + EOTF
- $EOTF^{-1}$ – the inverse of the electro-optical transfer function
  - Converts from a representation of displayed light back to its electrical signal



Transfer Functions

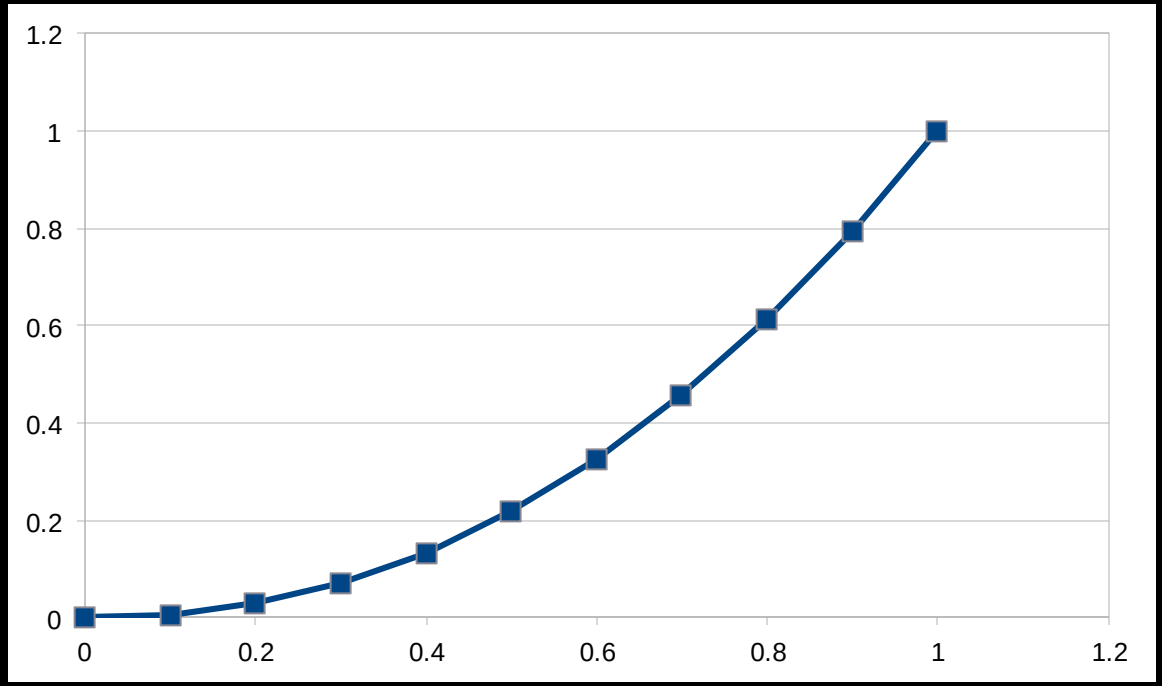
- Luminance is measured in nits, also known as candela per square meters (cd/m^2)
- Human's perceive difference in change between different stimuli
- JND = just-noticeable difference
- It's more efficient to encode in a roughly logarithmic fashion
- SDR Transfer Functions
  - sRGB, BT.709, BT.601
- HDR Transfer Functions
  - PQ (Perceptual Quantizer), HLG (Hybrid Log-Gamma)

Object	Luminance in nits (cd/m^2)
Sun	1.6 million
Fluorescent light	10,000
Highlights	1,000 - sunlight
White Objects	250 – 1,000
Typical Objects	1 - 250
Shadows	0.01 - 1
Ultra-Blacks	0 – 0.0005

Display	Luminance in nits (cd/m^2)
Typical PC display	0.3 - 200
Excellent LCD HDTV	0.3 - 400
HDR LCD w/ local dimming	0.05 - 1,500

Representing EOTF with a LUT

- LUT = lookup table
- LUTs are provided as mapping of uniformly spaced input values to outputs
- This is how current drm_crtc GAMMA and DEGAMMA LUTs work
- 256 entries for 8-bit inputs
- 1024 entries for 10-bit inputs
- 4096 entries for 12-bit inputs
- Display pipeline intermediate formats are generally much higher precision and using floating point formats to allow representation of content in linear luminance, in particular in the dark regions

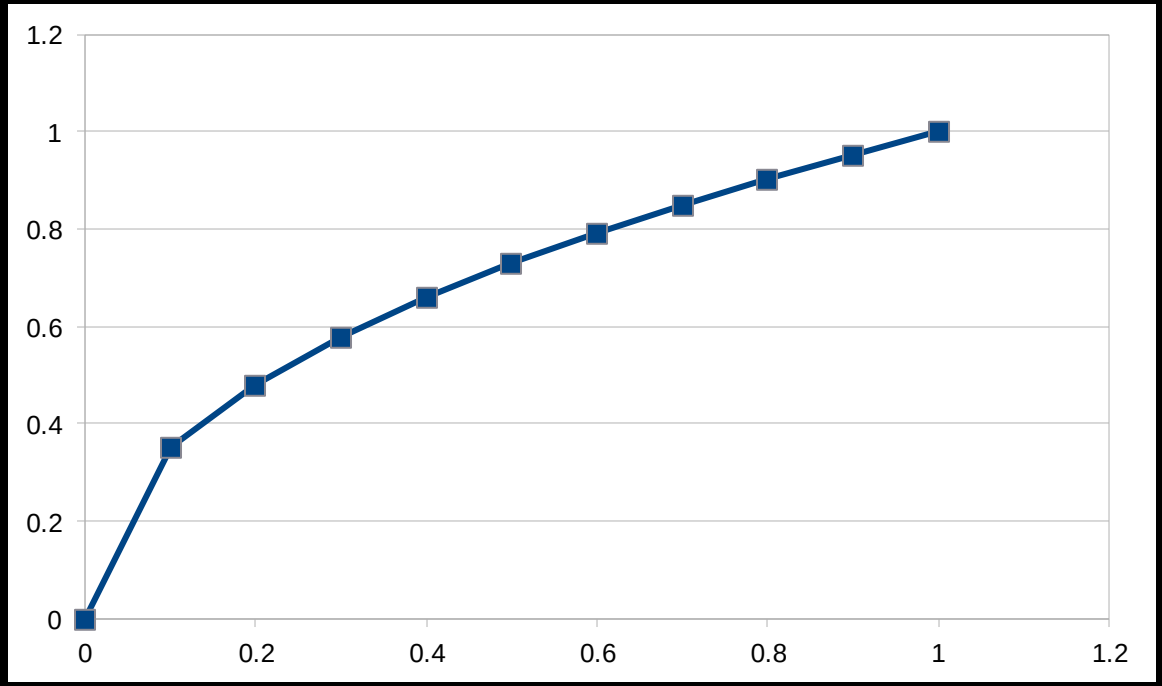


10-entry Gamma 2.2 LUT
to convert from non-linear to linear,
i.e., an EOTF

The Problem with EOTF^{-1} LUTs

- While our LUT seemed to approximate the Gamma LUT well it doesn't do so for the Inverse Gamma
- The graph on the right should look like a curve
- Instead it looks like a bunch of linear segments
- We don't have enough entries to approximate the dark regions well enough

- I see three ways to solve this problem



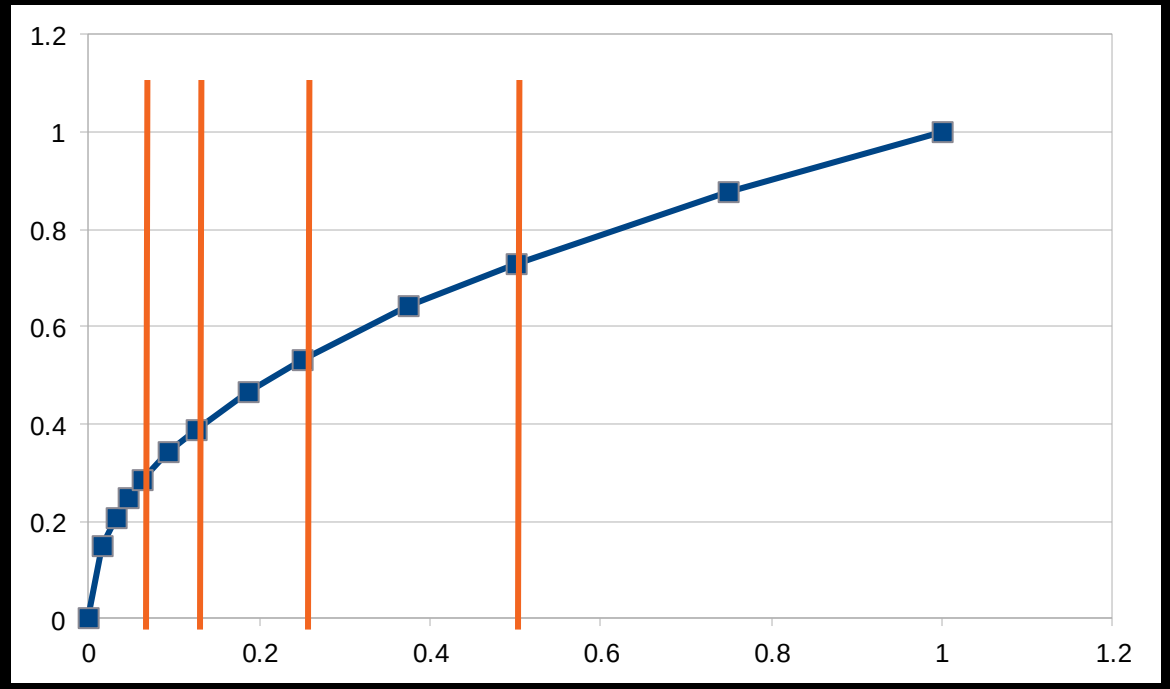
10-entry Inverse Gamma 2.2 LUT
to convert from linear to non-linear,
i.e., an EOTF^{-1}

Segmented LUTs

- Segment the LUT and use more points to describe lower regions

- Fairly complex API to describe segmented LUTs
- Different transfer functions require different segmentation
- We're at risk of defining an API based on one type of HW in a way that won't suit other HW in the future

- Proposed by Uma Shankar https://patchwork.freedesktop.org/series/90826/



12-entry PWL LUT,
i.e., an $EOTF^{-1}$

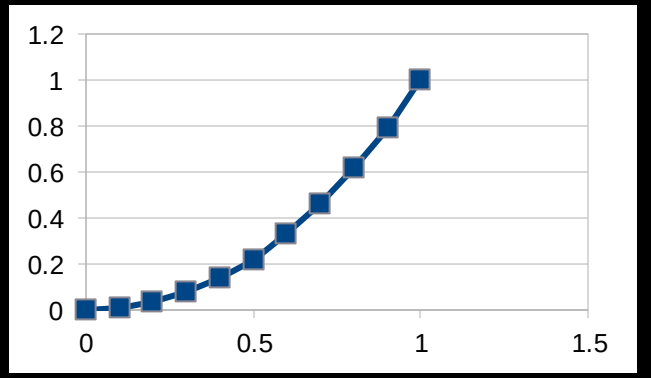
Inverse LUTs

- Define EOTF with inverse flag

- We can define the EOTF without problems
- Describe an EOTF with a regular LUT
- Pass a flag to use the LUT in the inverse
- Driver can invert the LUT

- I.e., define the output-to-input mapping

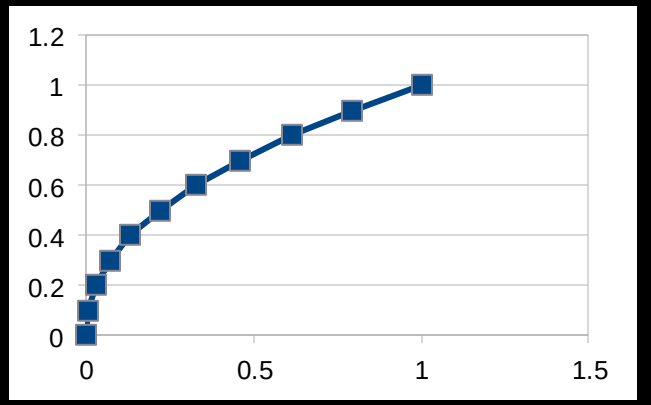
- This keeps the API definition simple
- Inverse of LUT (using segmented LUT) can be computed using integer math



1) Define the EOTF

LUT_INVERTED

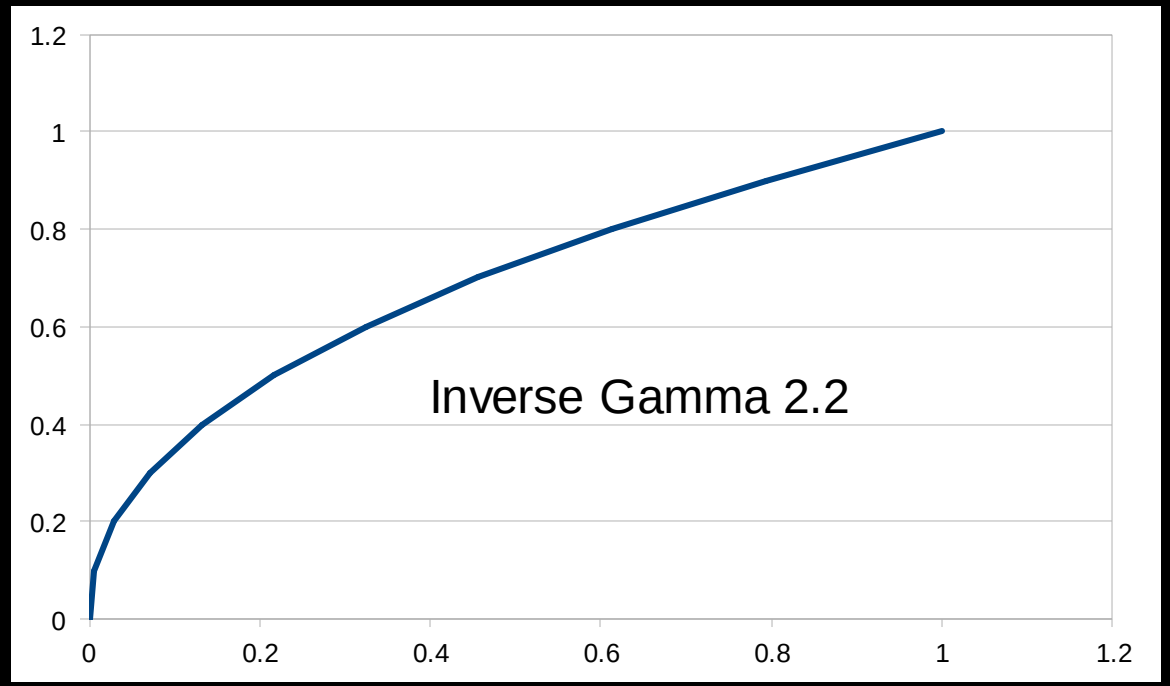
2) Pass Inverted Flag



3) Use Inverse of LUT

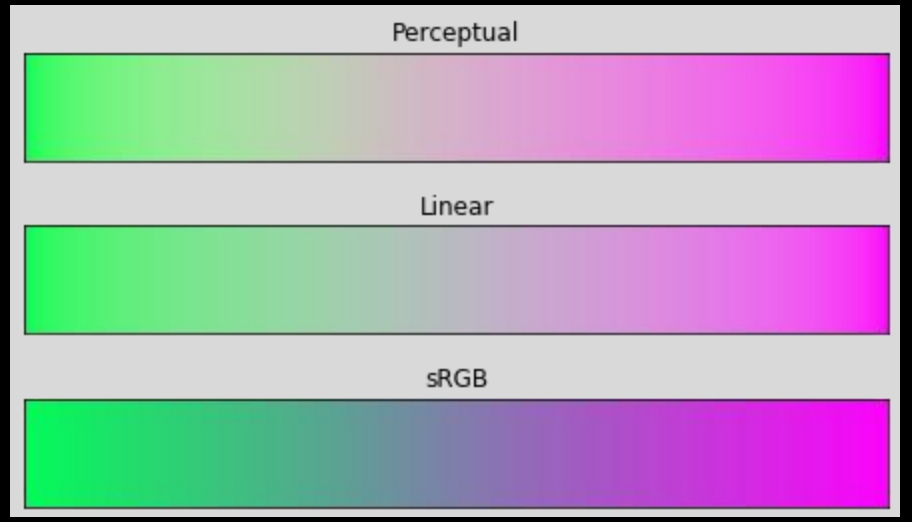
Named LUTs

- Define an enumeration of known $EOTF^{-1}$s
- Driver can use HW ROMs that support $EOTF^{-1}$
- If HW only supports custom LUT a driver can use hard-coded tables or compute the LUT



Blending & Scaling

- **Blending** in linear space is much closer to real-world physics
- People have strong opinions about (and reasons for) blending in linear or nonlinear representations
- Traditionally sRGB blending is often done using the nonlinear representation. It can be desirable to continue this to avoid breaking familiar behavior.
- Rendering Intent is important
- **Scaling** blends neighbouring pixel values and shows similar characteristics
- Read https://bottosson.github.io/posts/colorwrong/ for more info

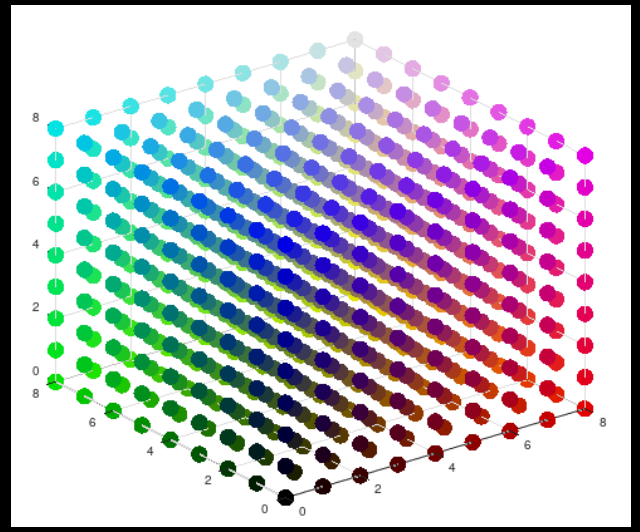


How Software gets color wrong by Björn Ottosson

What is a 3D LUT?

- 3D LUTs describe a transformation from an (R,G,B) triplet to another (R,G,B) triplet
- Uses limited entries in each dimension and interpolate in-between values
- Can use fewer entries when performed in non-linear space instead of linear space

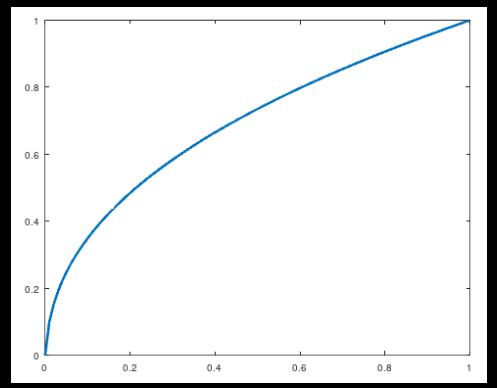
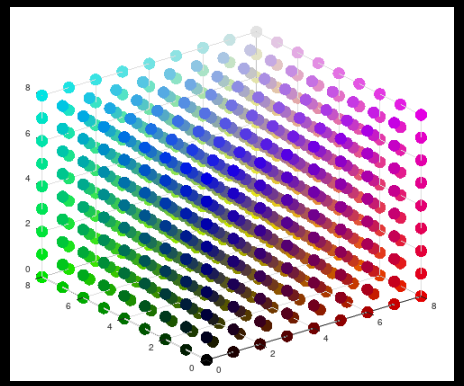
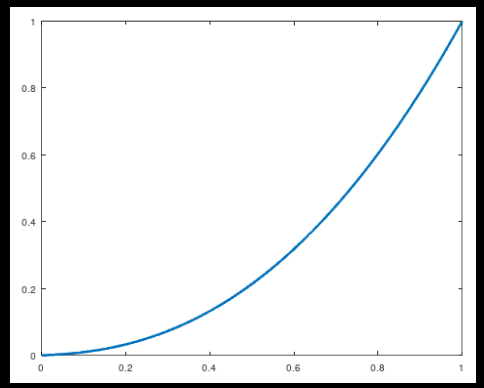
- Can express transformations well that:
  - affect all colors equally
  - affect only specific colors at specific brightness
  - desaturate or saturate highlights, dark, or midtones
- Can essentially be used to describe any complex or simple color transformation
- https://www.youtube.com/watch?v=xxlBTiNvYzE



3D LUT size	Entries	Bytes (assuming 64-bit entries)
9^3	729	5,832
17^3	4,913	39,304
33^3	35,937	287,496
65^3	274,625	2,197,000

3D LUT and Shaper LUTs

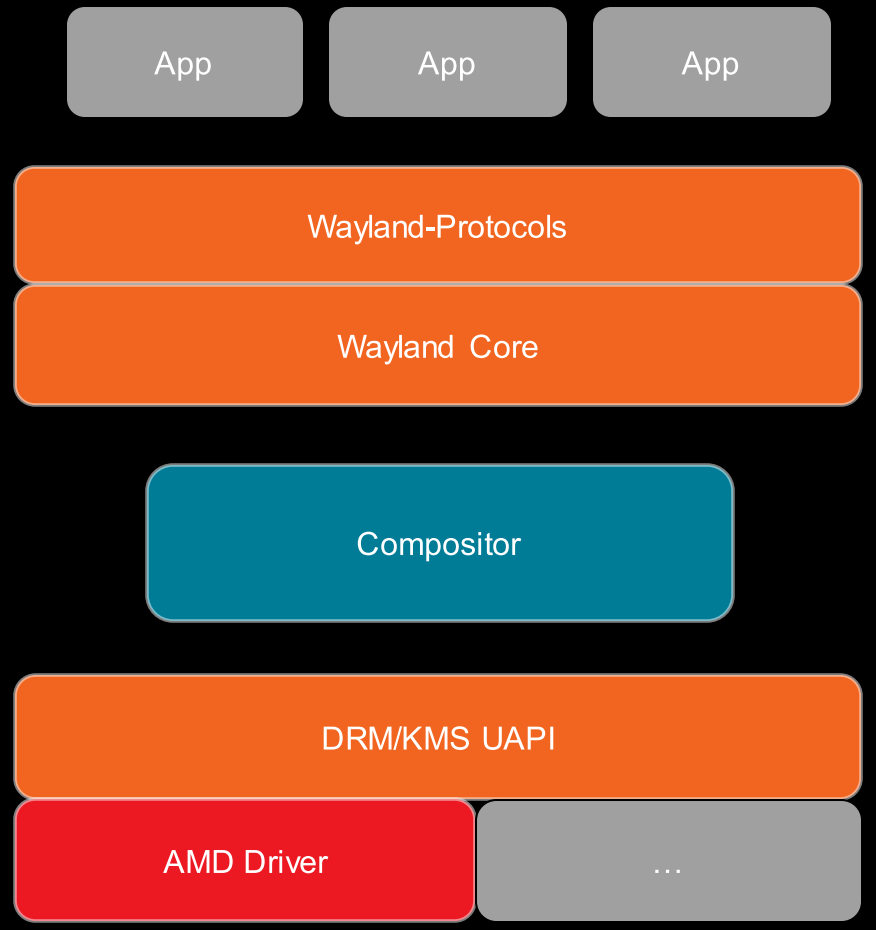
- 3D LUTs can apply on linear or non-linear content (optical or electric)
- 3D LUTs are limited in size because they contain N^3 entries, where N is the size of the LUT
- For the same reason that we encode content most efficiently in non-linear (sRGB, PQ, etc.) spaces a 3D LUT becomes much more effective when applied on non-linear content
- This can be facilitated by sandwiching the 3D LUT with Shaper 1D LUTs

AMD

HDR on Linux

Wayland & DRM/KMS



What do we want?

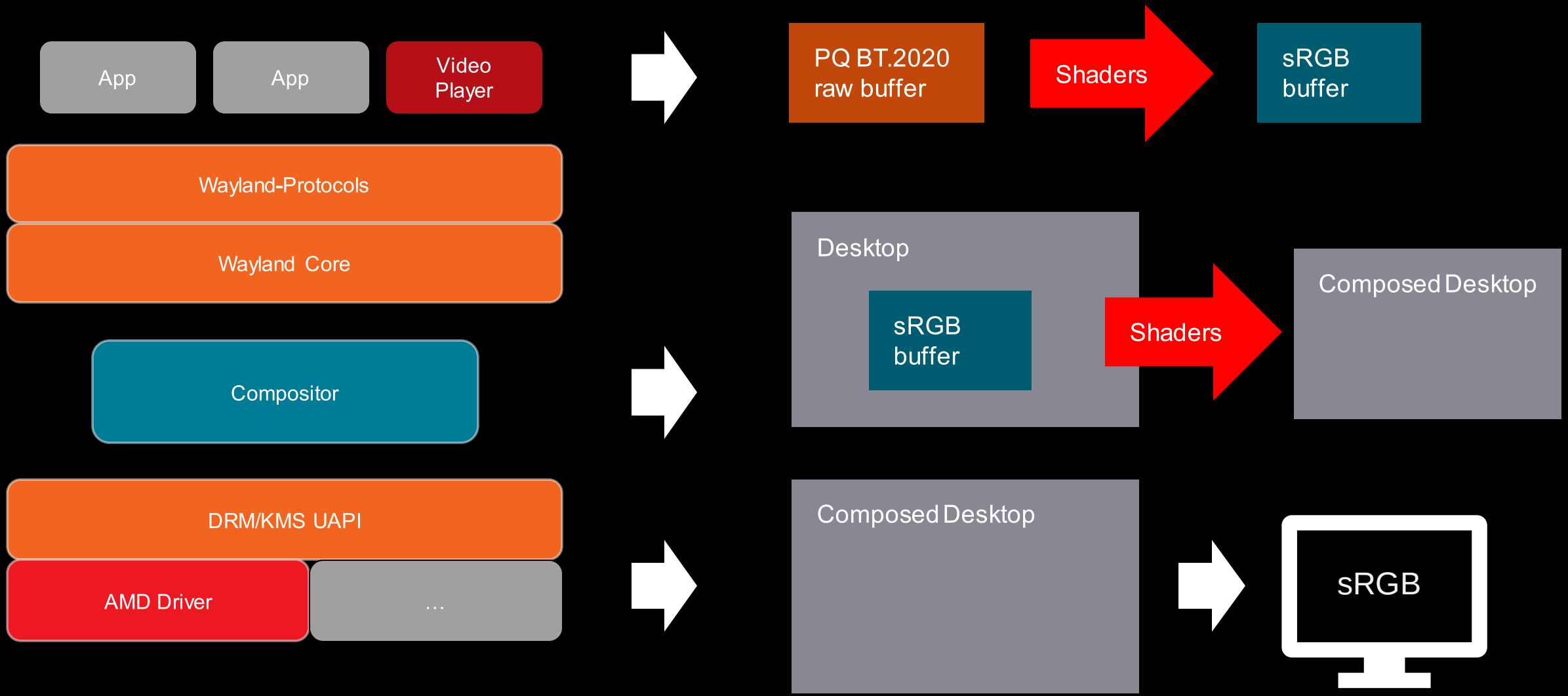
- Display HDR content (videos or games) in HDR on HDR displays
- Compose HDR content with SDR content on the desktop
- Direct scanout of HDR content

Why direct scanout?

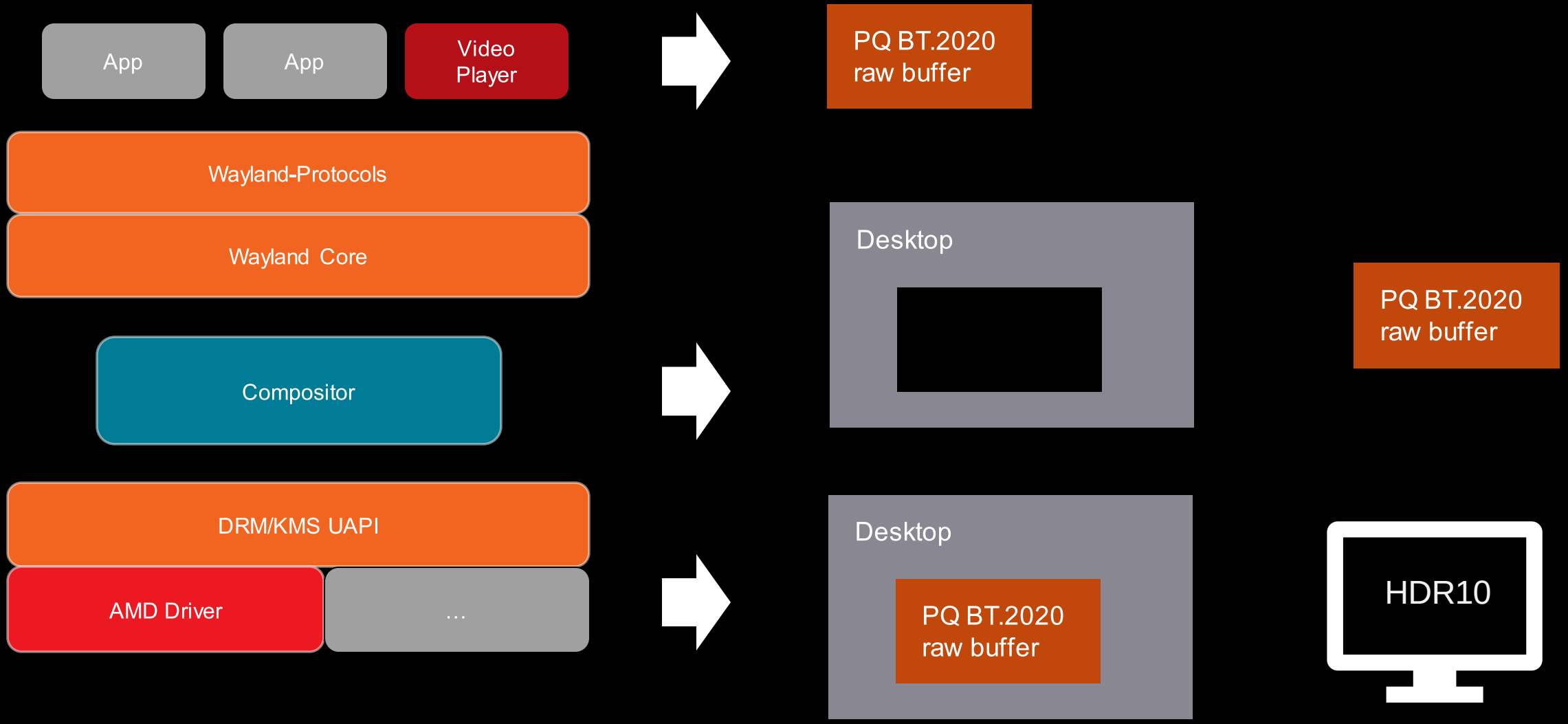
- When directly scanning out surfaces we can avoid shader composition
- This saves bandwidth
- It also saves GPU resources
- On AMD HW the GPU can go into a very low power state if it is entirely unused for stretches of time

- Primary use case:
  - Direct scanout of video planes (as underlay plane)

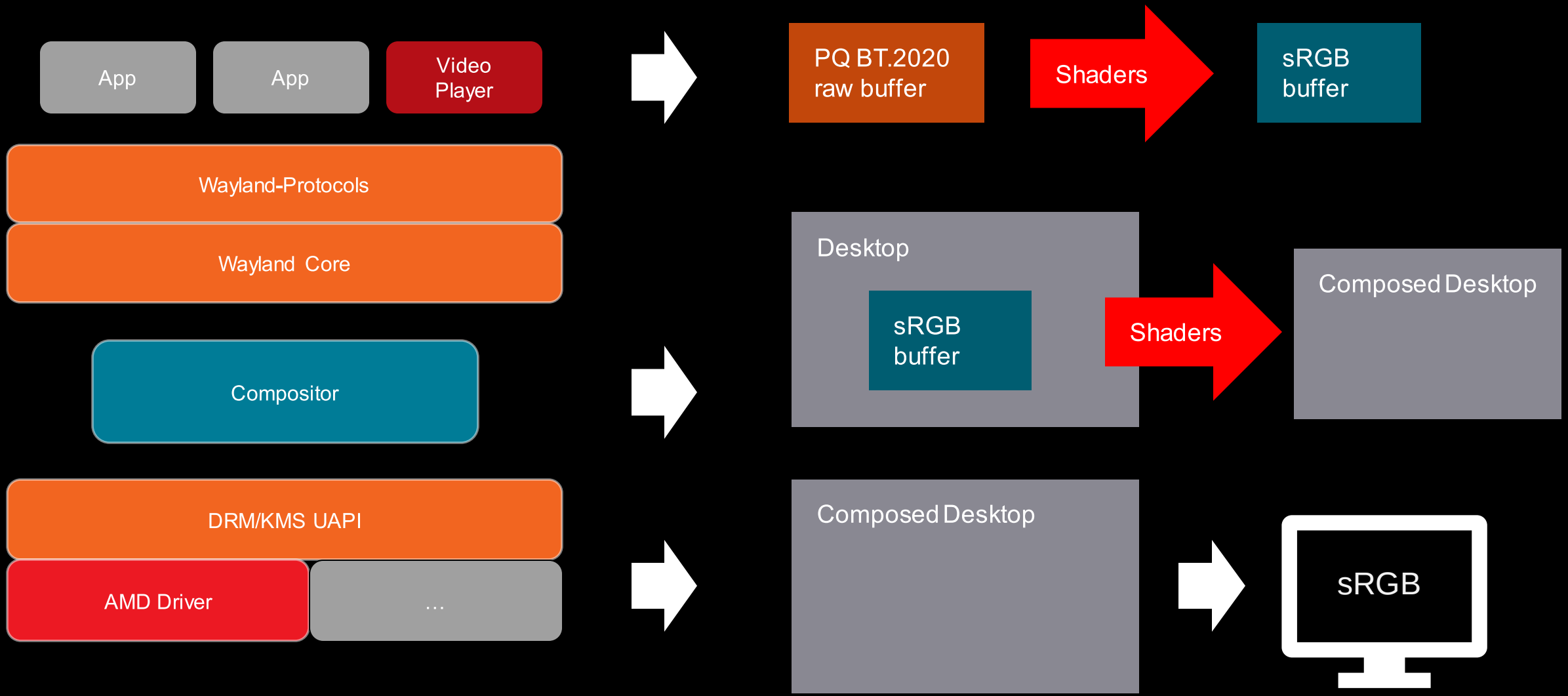
HDR Video – Current State



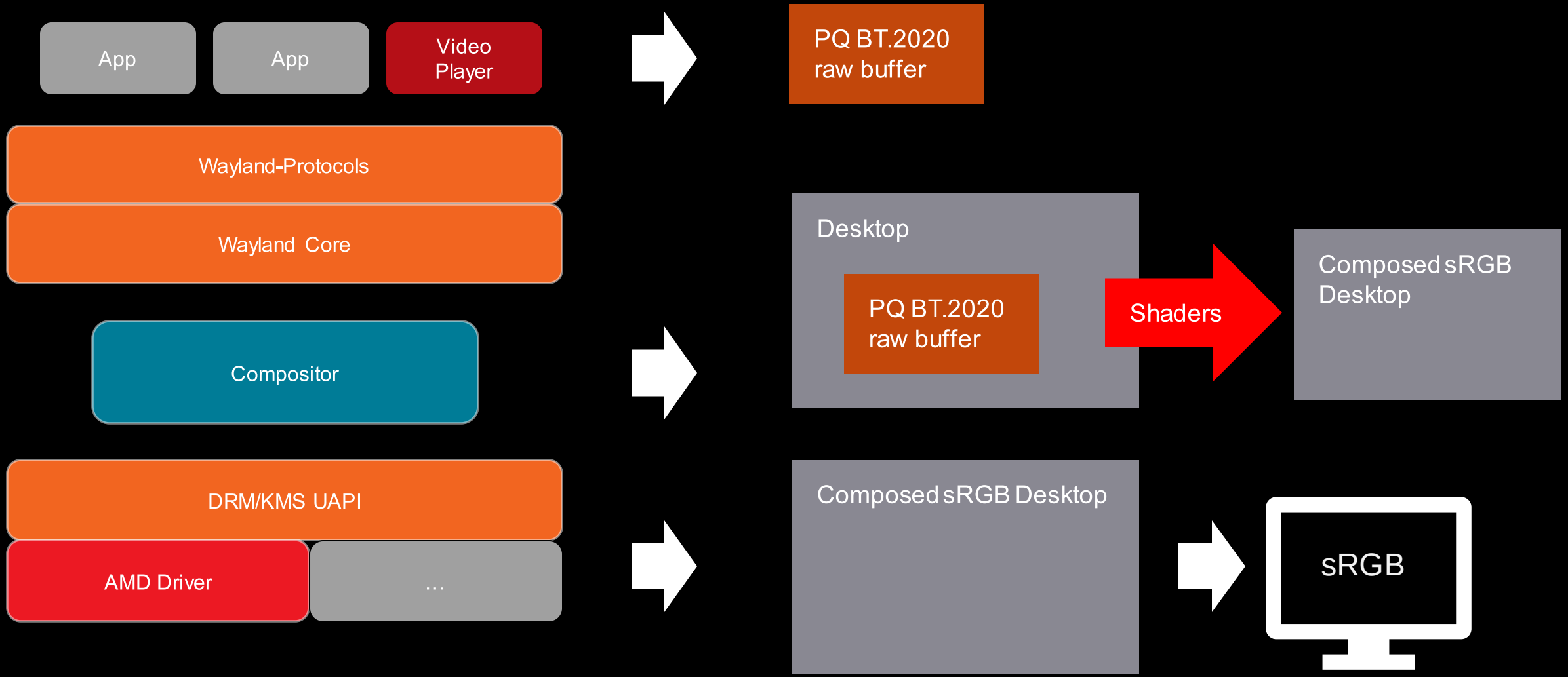
HDR Video – Desired State



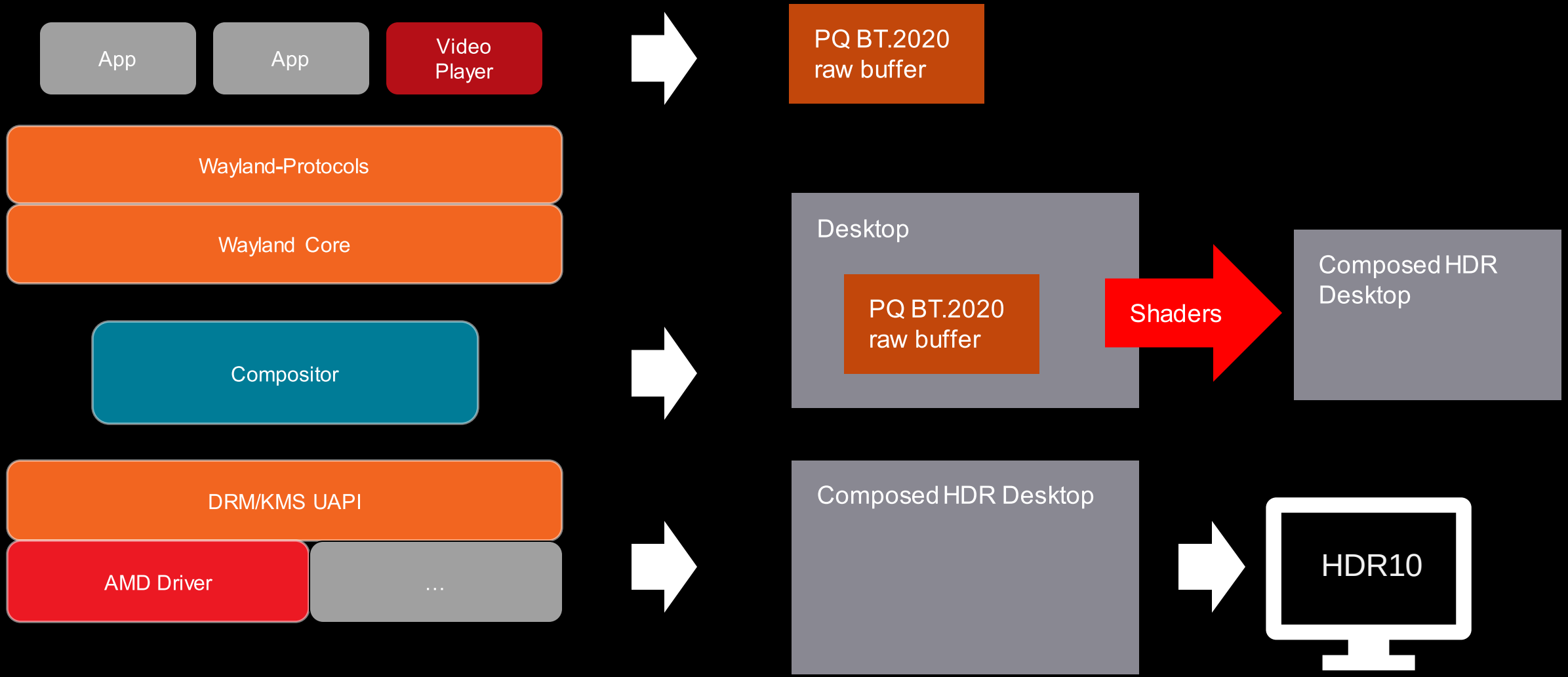
HDR Video – Current State



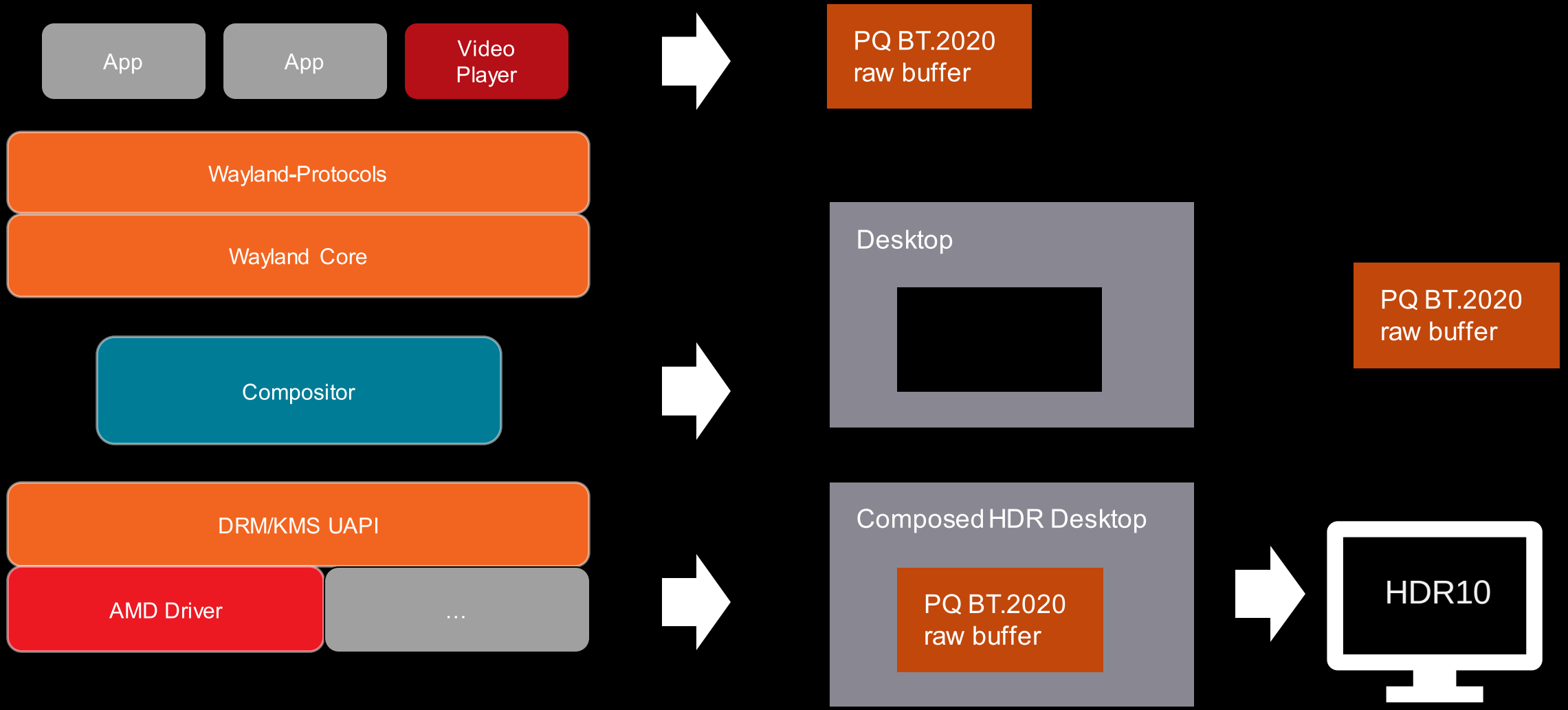
HDR Video – Step 1 – Desktop Composition



HDR Video – Step 2 – HDR Output



HDR Video – Step 3 – DRM/KMS Offloading

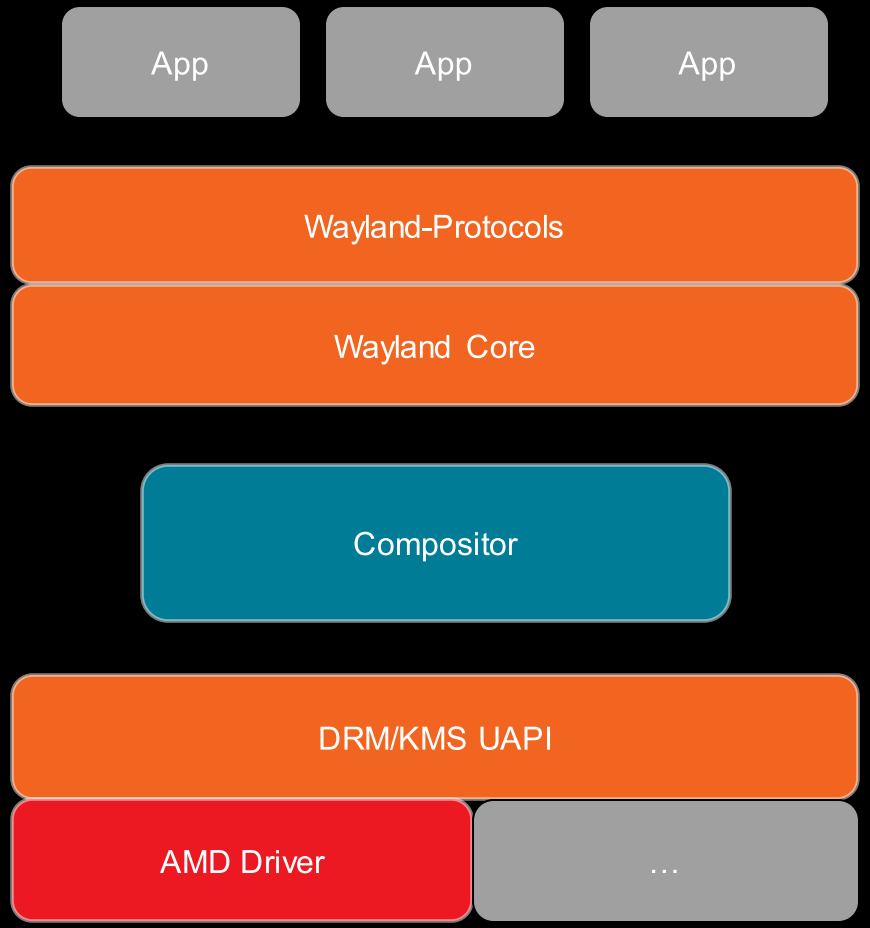


How do we get there?

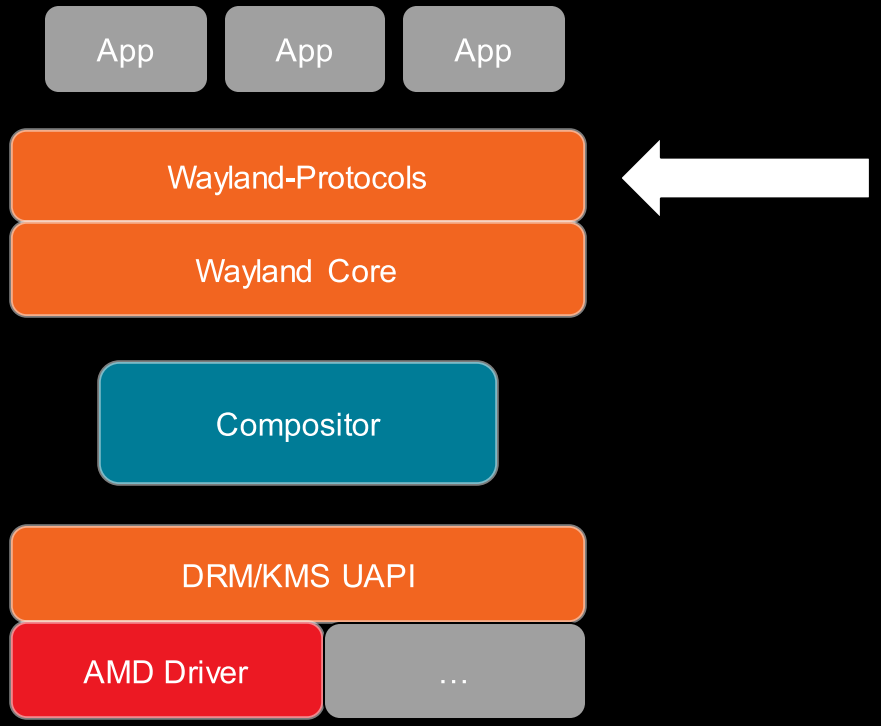
- We need a way for an application to communicate the surface color space(1) with the compositor
- We need to teach the compositor how to compose content with different color spaces
- We need to teach the compositor to render to an HDR framebuffer
- We need to provide the ability to offload a compositor's color pipeline to DRM/KMS

1) I'm using the term color space very loosely here to mean transfer function, primaries, etc.

The Big Picture



Wayland Protocol API



Existing

- Subcompositor API
  - Allows composition for surfaces within application window

- Viewporter API
  - Defines viewport of surface
  - Required for scaling MPO

Proposed

- Color Representation API
  - Chroma siting
  - Full vs Limited/Studio
  - YCbCr matrix

- Color Management API
  - TF or Gamma
  - Color space or Primaries+Whitepoint
  - YCbCr Matrix
  - Or (mutually exclusive) ICC profile

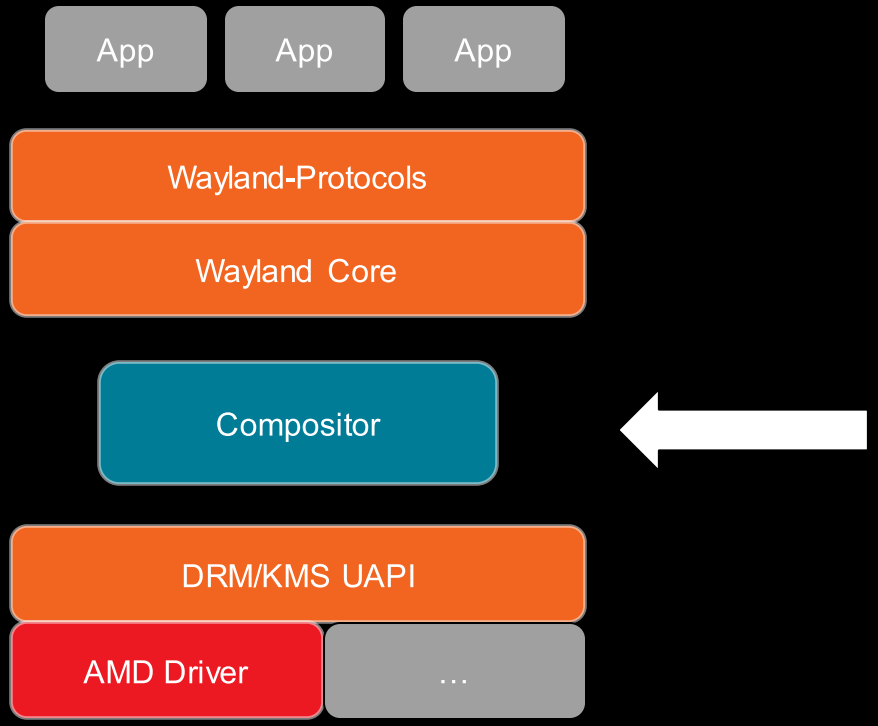
ToDo

- HDR Metadata API
  - Mastering Display
    - Primaries
    - Whitepoint
    - Luminance
    - MaxFALL
    - MaxCLL

- Scaling API
  - Define scaling filter
  - Define preference for scaling in linear vs non-linear space

- Direct 3D LUT

Weston Compositor



Existing work in HDR Branch

- Color-management awareness
- Blending in linear
- Assuming sRGB

Required work

- Use color representation API
- Use color management API (TF & Primaries) instead of assuming sRGB

- Offload post-blending EOTF^{-1} to DRM/KMS

- Offload pre-blending EOTF to DRM/KMS

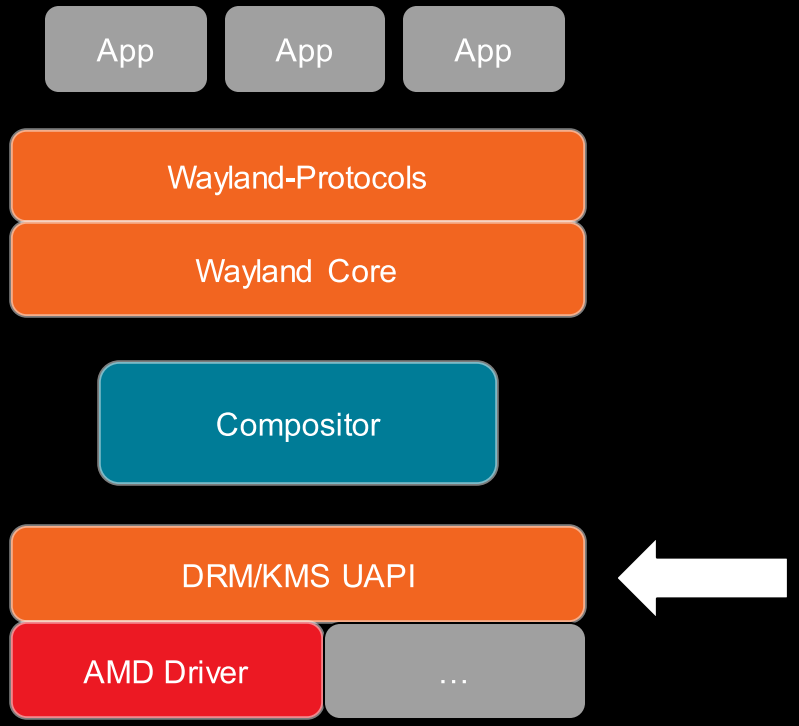
- Implement 3DLUT-based gamut mapping

- Offload pre-blend gamut mapping to DRM/KMS

- Implement 3DLUT-based display color correction

- Offload display color correction to DRM/KMS

DRM/KMS API – Current State



Existing

- drm_framebuffer
  - format
- drm_plane
  - color_encoding
  - color_range
- drm_crtc
  - Blender
  - Degamma
  - CTM
  - Gamma
- drm_connector
  - HDR Output Metadata

Missing

- Pre-blending Color Management
  - CTM
  - 1D LUT
  - 3D LUT
  - Shaper LUTs
- Post-blending Color Management
  - 3D LUT
  - Shaper LUTs
- 1D LUTs
  - Ability to specify pre-defined TF
  - Ability to specify $EOTF^{-1}$
- Ordering of operations
  - Order should support HDR color pipelines
  - ***Define CM in a way that allows drivers to map onto unique HW ordering***
- Predictability
  - Configuration of CM can fail
  - Programming of CM should never fail
  - Switching to bypass should never fail

DRM/KMS API – High-level design goals

- DRM/KMS API exists for power and bandwidth savings
- We need API that can transition seamlessly between DRM/KMS and shaders
- We need flexibility to accommodate the fact that
  - Fixed-function HW usually won't match shader precision
  - Fixed-function HW usually won't match shader flexibility
- We want to avoid leaving optimizations to the compositor
  - Optimizations can save hours of battery life
  - HW-specific optimizations are (rightfully) not a priority for compositor developers
  - When new HW allows for new optimizations, we want to enable them at product launch
  - We need an API and architecture that allows HW vendors control over release

AMD

Conclusion

Is HDR Harder?

Yes and No

Conclusion

HDR adds complexities but

- We have solved these problems on other platforms
- HDR and overall color management is making progress on Linux

We need a DRM/KMS API that provides flexibility for the diverse ecosystem on Linux

We need an API that is flexible enough to support evolving display HW and evolving understanding of HDR

We want an API that works for everyone

Join us in the HDR KMS API workshop tomorrow to discuss

Special Thanks

- AMD Folks
  - Syed Hussein
  - Keith Lee
  - Aric Cyr
  - Krunoslav Kovac
  - Bhawanpreet Lakha
  - Alex Hung
  - Alexander Deucher
  - Shashank Sharma
  - Vitaly Prosyak
- Others
  - Pekka Paalanen
  - Sebastian Wick
  - Timo Kunkel
  - Björn Ottosson
  - Uma Shankar
  - Ville Syrjälä
  - Jim Shargo
  - Chris Cameron
  - Miguel Casas
  - Sean Paul
  - Simon Ser

… and many others.

Further Reading

- Weston CM & HDR
- Color management and HDR documentation for FOSS graphics
- HDR in Linux: Part 1
- HDR in Linux: Part 2
- WIP: unstable: add color management protocol

- How Software gets color wrong
- sRGB gamut clipping
- Dolby Vision Overview of Key Features & Workflows
- The Perceptual Quantizer Design Considerations and Applications


AMD

Copyright and disclaimer

- The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate releases, for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

- THIS INFORMATION IS PROVIDED 'AS IS." AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION

DRM/KMS API – Current State

- There have been several proposals from different parties
- None address everything
- None are backed by compositor patches
- Weston is ready for offloading pre-blend EOTF and post-blend $EOTF^{-1}$ for sRGB
- We need more than EOTF and $EOTF^{-1}$ for sRGB to prove an API

Why underlay?

- Overlay = Desktop plane is at bottom; other planes are above
- Underlay = Desktop plane is at top, other planes are below
  - The main desktop plane will include a transparent cutout to show underlay plane beneath
  - The underlay plane is then positioned behind

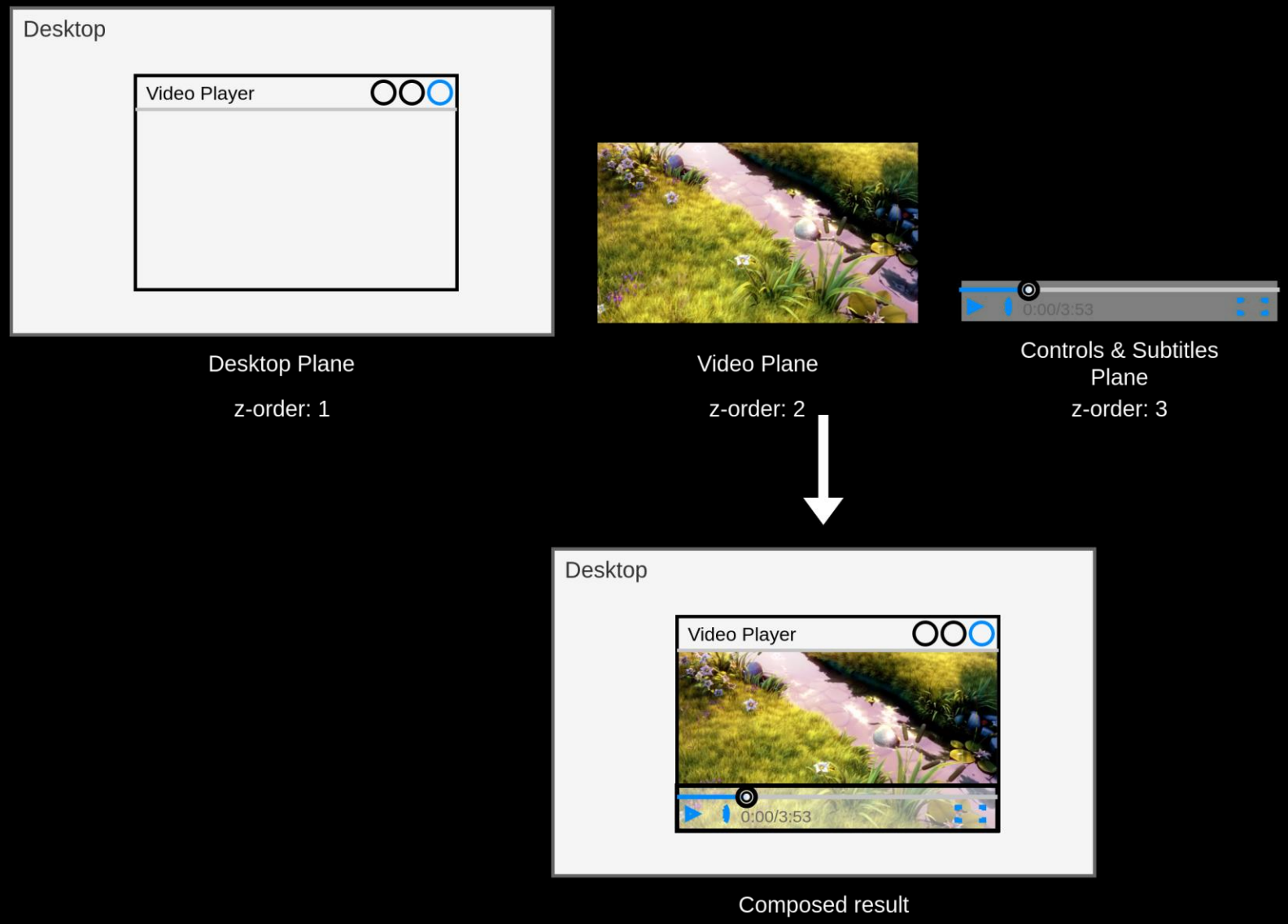
- Overlay planes can only be offloaded when they're entirely unoccluded
- Underlay planes can be occluded

- This allows rendering of subtitles and video controls on main desktop plane
- Allows composition of subtitles, controls, video and main desktop plane with 2 planes

- If rest of desktop, controls, and subtitles update infrequently the GPU can remain off during render of most video frames

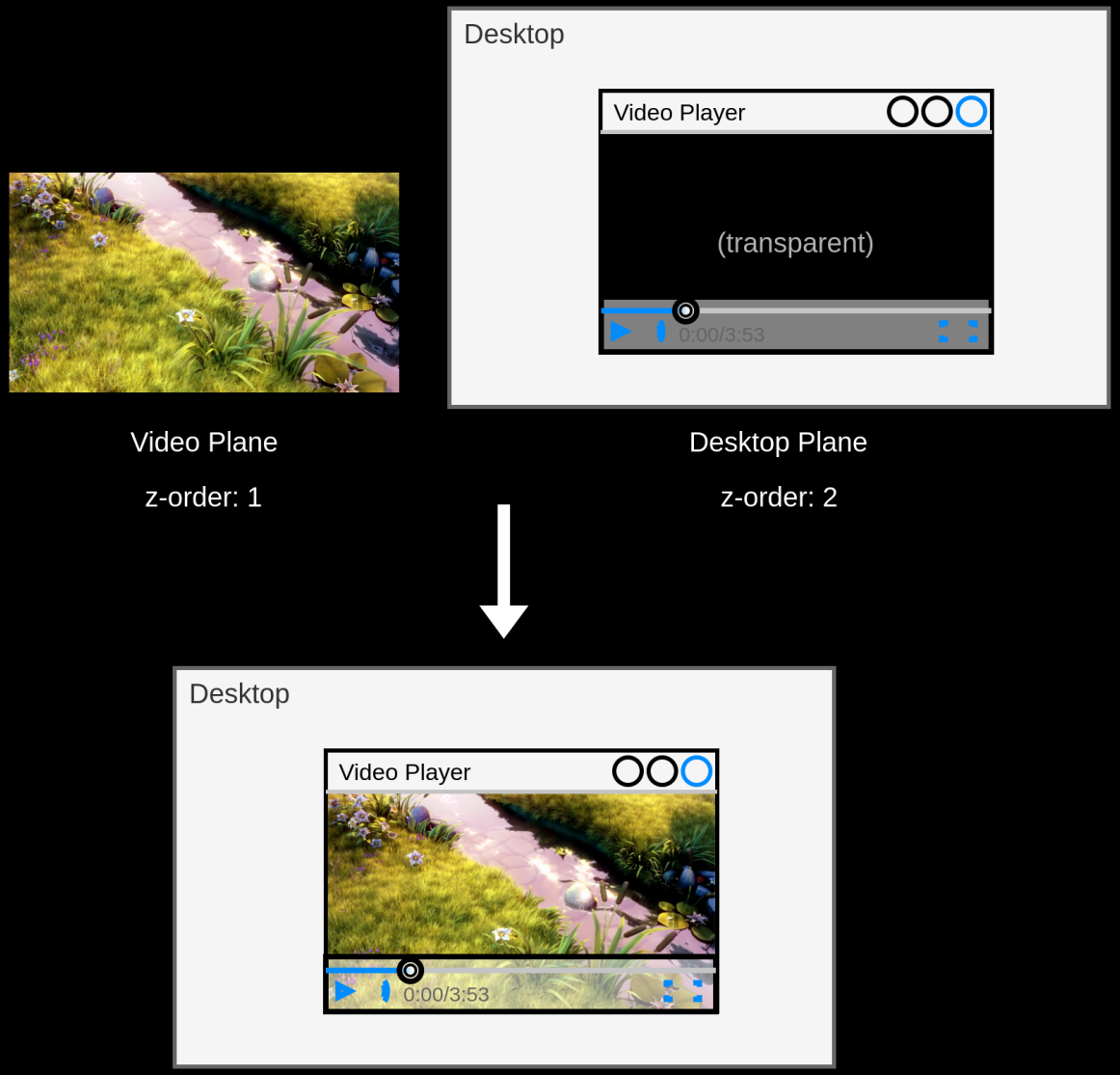
Why underlay?

Overlay composition:



Why underlay?

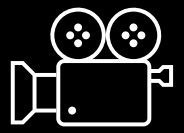
Underlay composition:

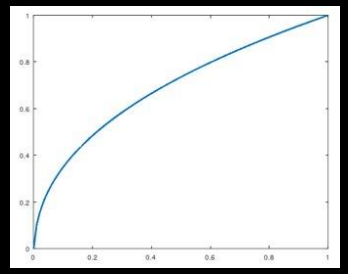


Transfer Functions - Gamma

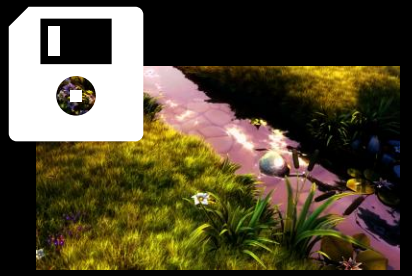
- Old analog displays and TVs CRTs convert signals into light in a non-linear fashion, roughly following a gamma function
- This worked nicely with logarithmically encoded video data
- Modern LCD/LED displays will use an explicit electro-optical transfer function (EOTF) to convert from the logarithmically encoded signal to drive the LCDs or LEDs
- SDR content is generally encoded via one of these opto-electronic transfer functions
  - sRGB
  - BT.601
  - BT.709

Real Life

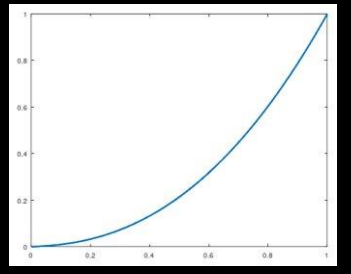




OETF



Digital Medium



EOTF

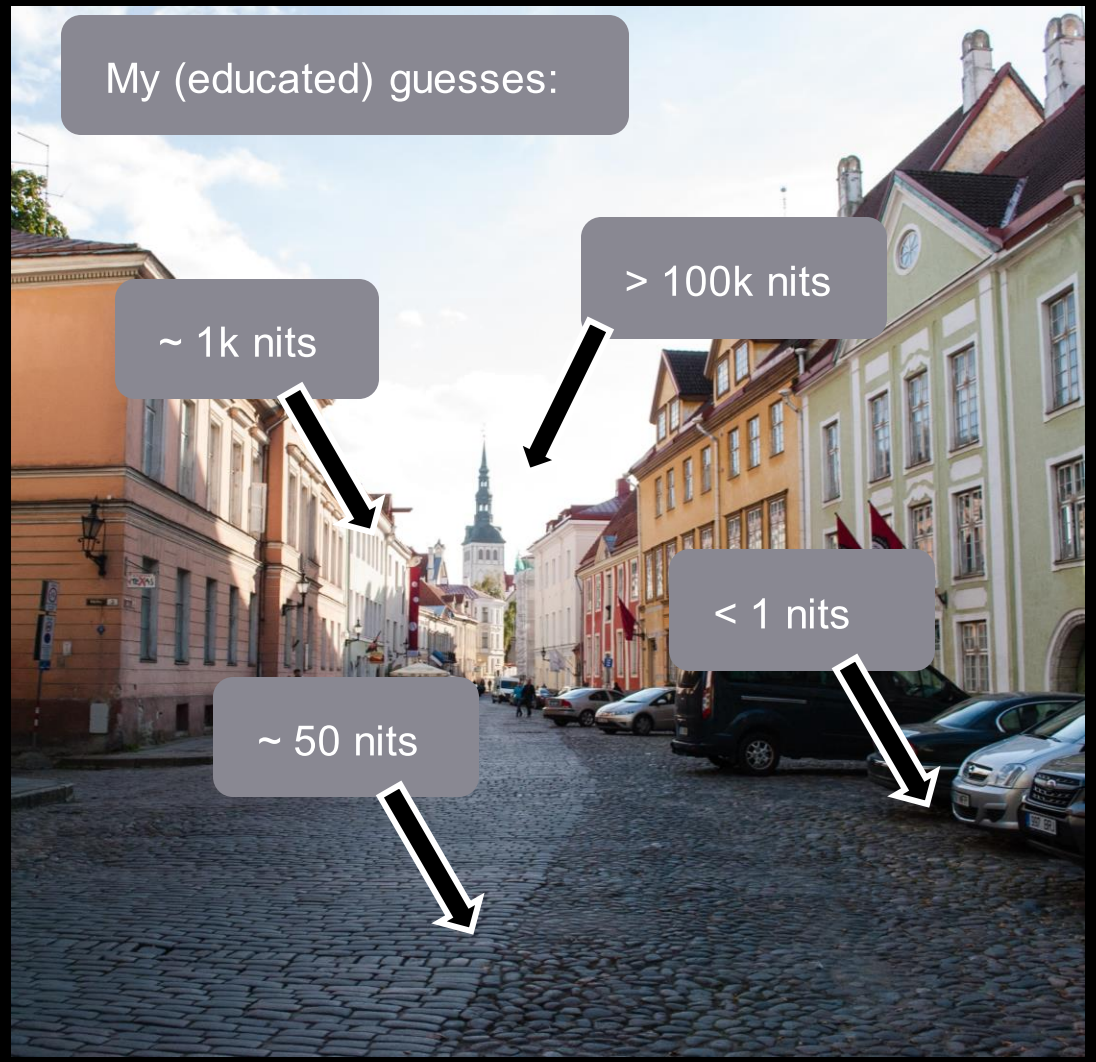
Transfer Functions – PQ and HLG

- Dolby defined PQ – Perceptual Quantizer
  - Designed to approximate human perception of luminance
  - Maps signal values to cd/m^2 (nits) in range of 0.0001 to 10,000
  - Used by HDR10
  - Originated in the movie industry
  - https://www.color.org/hdr/04-Timo_Kunkel.pdf
- BBC defined HLG – Hybrid Log-Gamma
  - Compatible with standard SDR content
  - Does not map to nits
  - HLG does not assume a standard viewing environment and is intended to work fine with a wide range of viewing environments and display brightness
  - Developed for (live) TV broadcast
  - https://www.color.org/hdr/03-Simon_Thompson.pdf

Reference Luminance

Object	Luminance in nits (cd/m²)
Sun	1.6 million
Fluorescent light	10,000
Highlights	1,000 - sunlight
White Objects	250 – 1,000
Typical Objects	1 - 250
Shadows	0.01 - 1
Ultra-Blacks	0 – 0.0005

Display	Luminance in nits (cd/m²)
Typical PC display	0.3 - 200
Excellent LCD HDTV	0.3 - 400
HDR LCD w/ local dimming	0.05 - 1,500



Tone Mapping

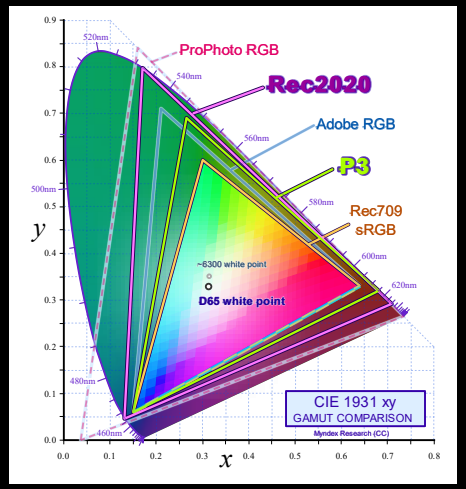
- Reality: 1+ million nits
- Mastering environment: Dark room, 4,000 nits panel
- Viewing environments:
  - Dark living room, 1,000 nits panel
  - Bedroom, 400 nits panel
  - Sunlit patio, 500 nits panel
- What do you do to go from mastered content to a panel in a given viewing environment?
  - Do nothing, cap at output nits
  - Apply curve to gently bring max nits down to panel caps
  - #2 + adjust dark regions to viewing environment
  - plus, a number of other approaches
- In essence we need to apply a curve. Though once we deal with colors and operate at the limits of our representations a 1D curve (one curve per color component) might not be enough.

Static & Dynamic Metadata

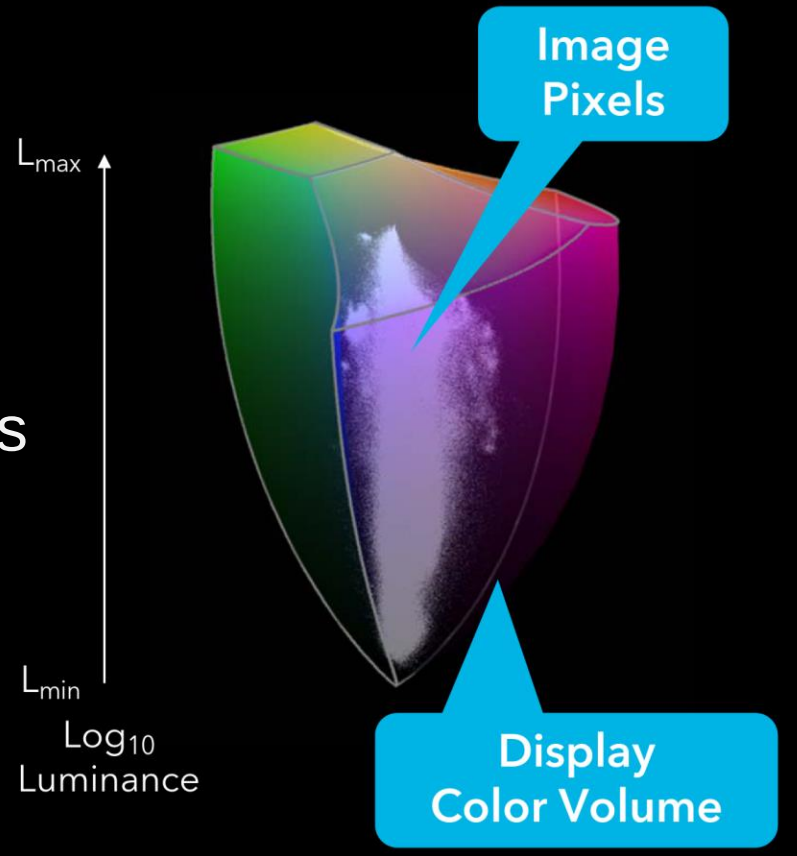
- HDR Video Content carries metadata info
- Metadata provides extra information about content to help with tone and gamut mapping
- Metadata can be static, i.e., not change for the duration of the video
- Or dynamic, i.e., change frame by frame

Color Volume

- The CIE 1931 diagram shows color gamut at a fixed luminance
- Displays' ability to represent colors changes with luminance
- Color Volume diagram can show full extent of representable color volume, as well as the full volume of image pixels



CIE 1931 xy GAMUT COMPARISON by Myndex / CC BY-SA 4.0

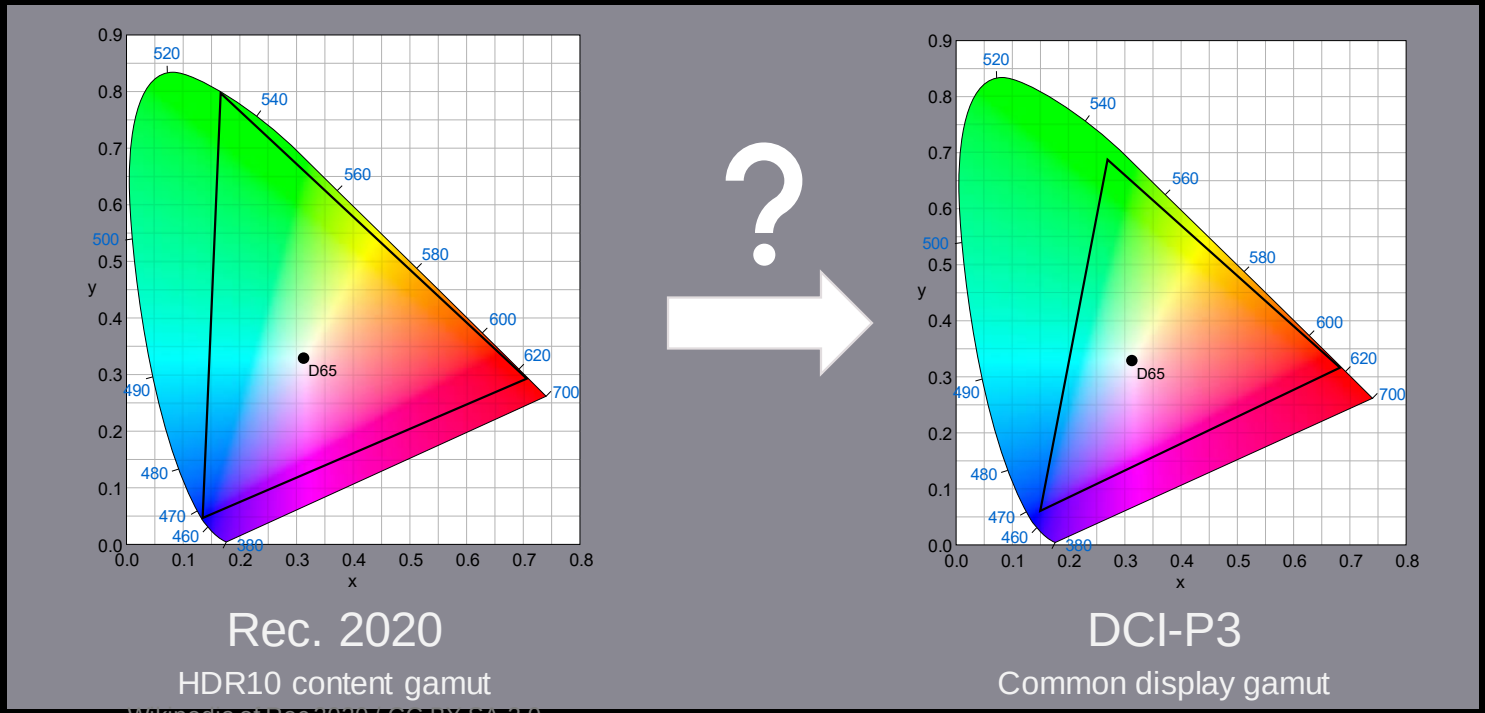


Display Color Volume by Timo Kunkel

Gamut mapping

What problem are we solving?

Mapping a larger gamut, such as Rec. 2020 onto a smaller gamut, such as DCI-P3, or Rec. 709



Wikipedia of Rec 2020 / CC BY-SA 3.0

Wikimedia of DCI-P3 / CC BY-SA 3.0

Matrix vs 3DLUT Gamut mapping

Unprocessed
(values interpreted in Destination color space)

Clipped
(color transformation matrix)

Gamut Mapped
(3D LUT)

sRGB Gamut Clipping by Björn Ottosson

Gamut Mapping algorithm

- Gamut Mapping (and most other color transformations) can be supported by a 3D LUT
- In order to use 3D LUT for gamut mapping we need an algorithm to compute the 3D LUT
- Most companies guard 3D LUT algorithms closely
- We might be able to look to video players like mpv or VLC for open-source 3D LUT algorithms (or libplacebo)
- Gamut clipping still seems to be a common default for gamut mapping operations


AMD

Copyright and disclaimer