Vulkanised 2025

The 7th Vulkan Developer Conference
Cambridge, UK | February 11-13, 2025

Color Volume Transformations in Vulkan Shaders

Richard Everheart, AAD

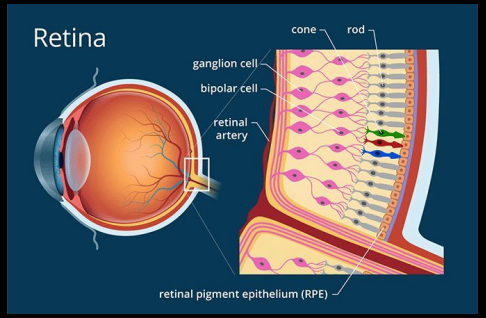
OVERVIEW

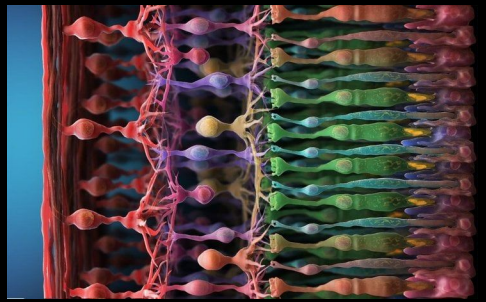
- Preliminaries
  - XYZ colorspace
- What is a color volume?
  - How color volumes differ from colorspaces
- What is tone mapping?
  - How to handle HDR metadata
  - How to get LDR source content displayed correctly on HDR sinks, and vise-versa
- Our approach on Linux
  - Gaining advanced color capabilities on Linux (X11, Wayland), without driver modification

PHYSIOLOGY

ADVANCED APPLICATION DESIGN, INC.

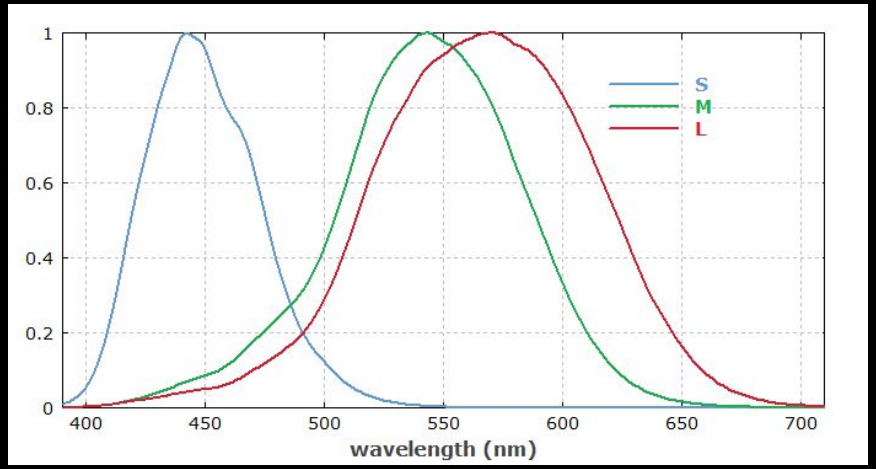
- The human eye contains cones which activate under certain wavelengths of light
  - They also contain rods, but we don't care about these
- Cones for Long/Red (L), Medium/Green (M), and Short/Blue (S) wavelengths
- Cone cells are analog, and do not give an "all or nothing" response
  - Tonically active, and continually release neurotransmitters
  - Wavelength and light intensity information is confounded in their output
- Completely different waveforms of light may look the same (metameric)
  - Completely identical waveforms may look different





LMS COLORSPACE

- A *colorspace* mathematically relates observed color phenomenon to physical color phenomenon



- The activation of L, M, and S cones for waveforms of light form a colorspace
  - A waveform of light with wavelength of 420nm results in a normalized cone response of 0.5

CIE 1931 COLORSPACE

- Spectral sensitivities of cones were not available in 1931
  - Not available until 1983! (H. J. A. Dartnall, J. K. Bowmaker and J. D. Mollon, 1983)

- CIE 1931 colorspace was created from experimental data instead, involving exclusively male, British human observers
  - William David Wright: 10 observers
  - John Guild: 7 observers

WRIGHT/GUILD COLOR EXPERIMENTS

Advanced Application Design, Inc.

- Two boxes enclosing light sources, with slits for looking inside
- **Left box:** Three light sources
  - Emitted light at constant wavelengths *700*nm, *546.1*nm, *435.8*nm, known as color primaries
  - Chosen because they could be accurately emitted using a mercury-arc lamp
- **Right box:** Target light source
  - Emitted light at a constant wavelength, and constant energy level (brightness)
- **Goal:** Change the brightness of the three sources on the left to match the light source on the right

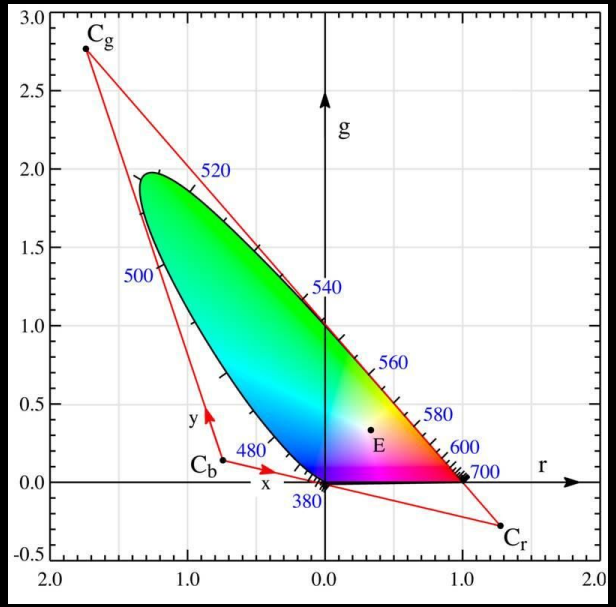
CIE 1931 RGB COLORSPACE

ADVANCED APPLICATION DESIGN, INC.

- Experimental data measured how color primaries could be added together to form unique colors

- Three color matching functions (CMFs) were chosen to represent experimental data
  - $r(\lambda), g(\lambda), b(\lambda)$
  - C_r, C_g, C_b are not actually visible

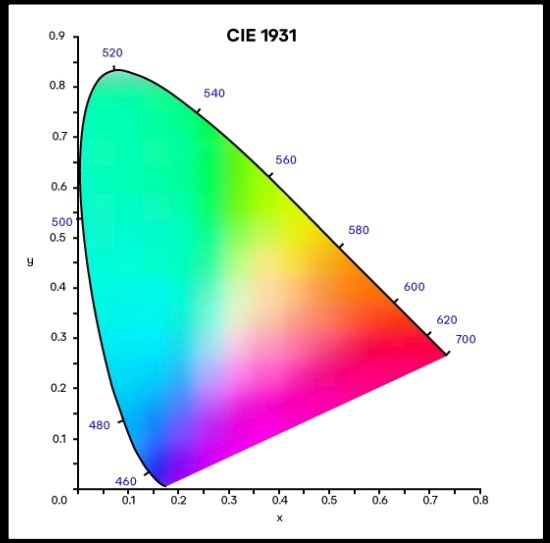
- If we project the colorspace onto the RG plane, we get the CIE RG colorspace



CIE 1931 XYZ COLORSPACE

Advanced Application Design, Inc.

- **Problem:** Red is not strictly additive, so the resultant functions were not non-negative
  - The concern was this could mess up calculations
  - Computers were not invented yet
- **Solution:** Map CIE 1931 RGB CMFs to a strictly positive domain
- This is CIE XYZ colorspace
  - X is chroma (red, green, blue), Y is luminance, Z is quasi-blue
  - Normalized, we get CIE xyz colorspace, with $z = 1 - x - y$
- If we project to the xz plane, we get the CIE xy colorspace
  - If we reintroduce the Y component we get CIE xyY colorspace
  - Everything in the triangle $(0, 0), (1, 0), (0. 1)$
- **Note:** CIE XYZ, CIE xyz, CIE xyY are all used interchangeably to mean CIE xyz

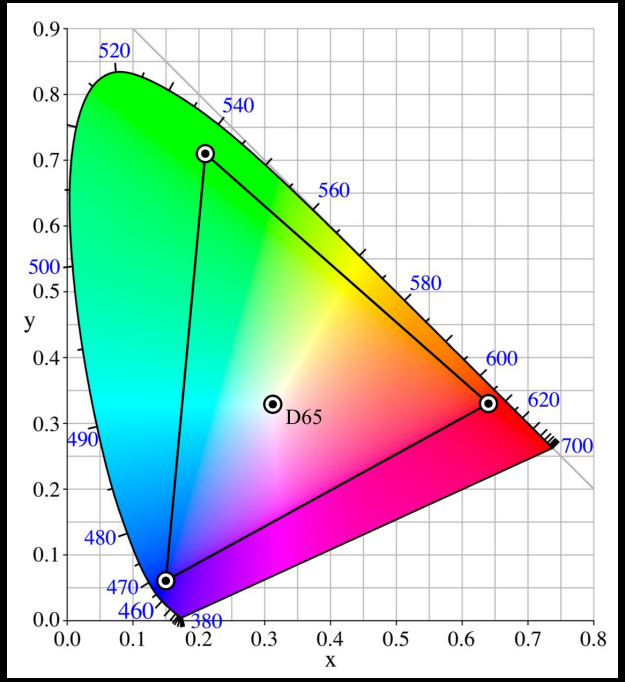


WHAT IS A COLORSPACE?

- Most colorspaces are derived from the CIE xyz colorspace

- A colorspace includes
  - *Color primaries* (triangle vertices), defining a gamut
  - *White point*

- **Example:** SRGB
  - Everything in the triangle,
  - Whitepoint of D65 (6500K)

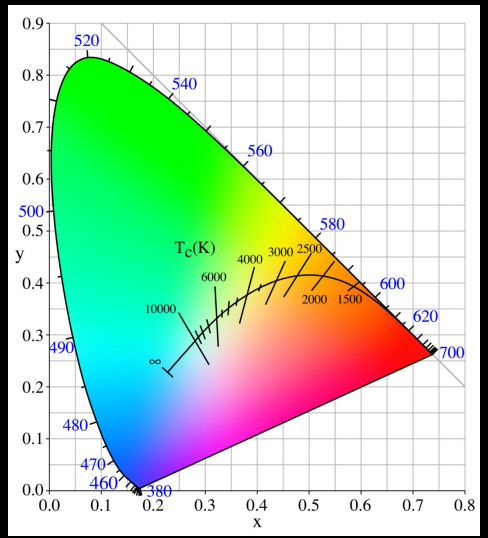


WHITE POINT

- The location of the maximum color in the color space
  - i.e. (255, 255, 255)

- **Plankian locus**: The path the color of an incandescent black body would take, when heated, in a particular colorspace
  - D65: Color of an incandescent black body at 6500K

- A change in white point is rotates the colorspace along the Plankian locus



WHAT IS A COLOR VOLUME?

- Colorspaces do not specify how luminosity (Y) is interpreted
  - What luminosity should a value of rgb(200, 200, 200) be displayed at on a monitor?
  - (200.0f / 255.0f) = 78% maximum display luminosity? Something else?

- A color volume includes an EOTF (gamma), and OETF (degamma) for mapping luminosity
  - EOTF: Electric Optical Transfer Function (OETF^{-1})
  - OETF: Optical Electric Transfer Function (EOTF^{-1})

- An EOTF takes an input a normalized color, and outputs a brightness value
  - This brightness value should be normalized to the brightness range of your monitor

LINEAR VS NON-LINEAR COLOR VOLUMES

- A color volume is *linear* if it uses a linear EOTF/OETF
  - SRGB is non-linear, with low dynamic range (LDR)
  - Rec. 2100 is non-linear, with high dynamic range (HDR)

- A linear color volume is required for
  - Changing color primaries
  - Whitepoint conversion
  - Texture filtering
  - Blending

- A color volume can be made linear by applying its EOTF^{-1}
  - Known as “degamma”

DYNAMIC RANGE

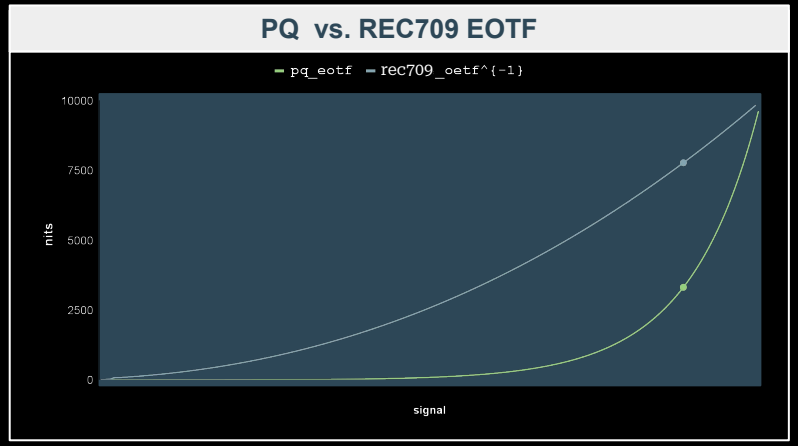
- The ratio between min and max luminance
  - HDR10/HDR10+ allows for 10,000 nits of luminance
    - Practically, OLED HDR monitors have max luminance between 600 and 1600
  - LDR (SDR) allows for 100 nits of luminance

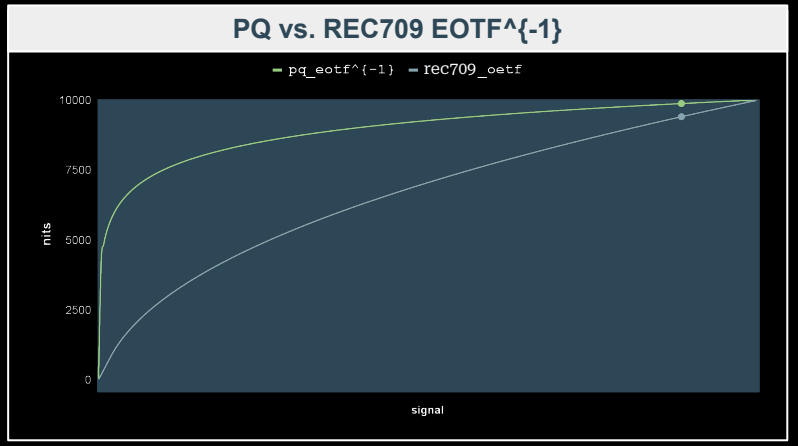
- HDR images use either a *PQ* (great) or *HLG* (not nearly as great) EOTF
  - **PQ:** Perceptual Quantizer
  - **HLG:** Hybrid Log Gamma
    - Most HDR applications use HLG, which is why no one is that impressed with HDR yet

$$F_{\mathrm{PQ}}(L) = 10000\left(\frac{\max\left[\left(L^{1/m_2} - \frac{107}{128}\right), 0\right]}{\frac{2413}{128} - \frac{2392}{128}L^{1/m_2}}\right)^{1/m_2} \qquad m_1 = \frac{1305}{8192} \qquad m_2 = \frac{2523}{32}$$

EOTF COMPARISON

The EOTFs for LDR and HDR data are different.



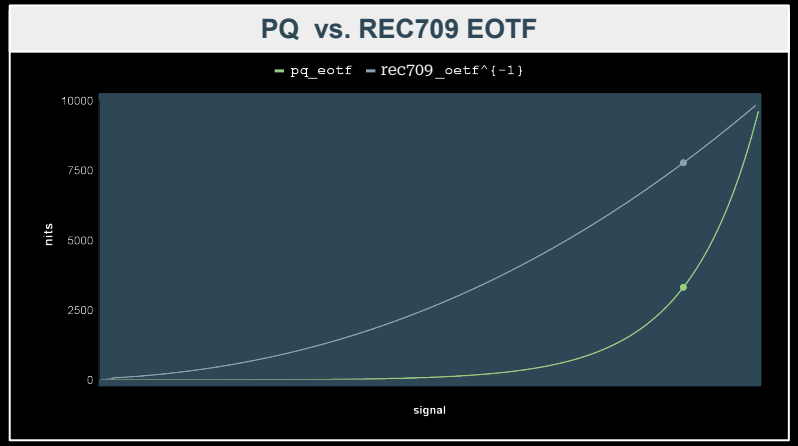


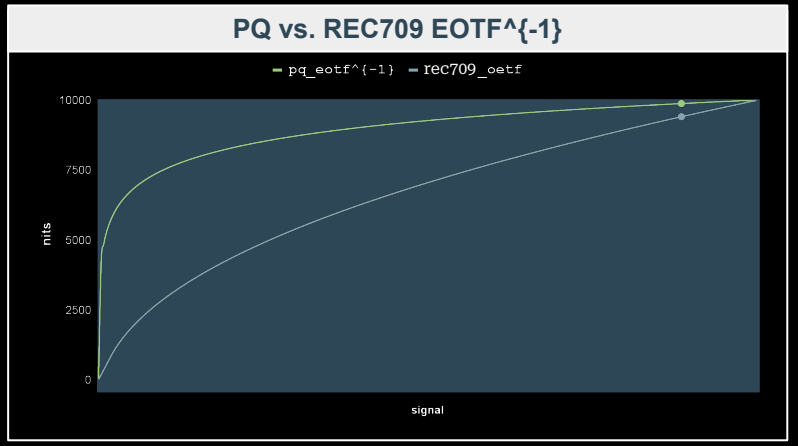
A value of `{230,230,230}` of LDR content will have 75% of the brightness of our monitor.

- Assuming our LDR content is in Rec. 709

EOTF COMPARISON

The EOTFs for LDR and HDR data are different.





A value of `{230,230,230}` of HDR content will have 33% of the brightness of our monitor.

- Assuming our HDR content is in PQ (Rec. 2100)

EXAMPLES OF COLOR VOLUMES

Advanced Application Design, Inc.

	(x_w, y_w)	(x_r, y_r)	(x_g, y_g)	(x_b, y_b)	VkFormat, VkColorspaceKHR
SRGB	$(0.3127, 0.3290)$	$(0.64, 0.33)$	$(0.33, 0.30)$	$(0.15, 0.06)$	VK_FORMAT_X_N_SRGB VK_COLOR_SPACE_SRGB_NONLINEAR_KHR
Rec. 709	$(0.3127, 0.3290)$	$(0.64, 0.33)$	$(0.33, 0.30)$	$(0.15, 0.06)$	N/A VK_COLOR_SPACE_BT709_NONLINEAR_EXT
Rec. 2100	$(0.3127, 0.3290)$	$(0.708, 0.292)$	$(0.170, 0.797)$	$(0.131, 0.046)$	N/A VK_COLOR_SPACE_HDR10_ST2084_EXT

EXAMPLES OF COLOR VOLUMES

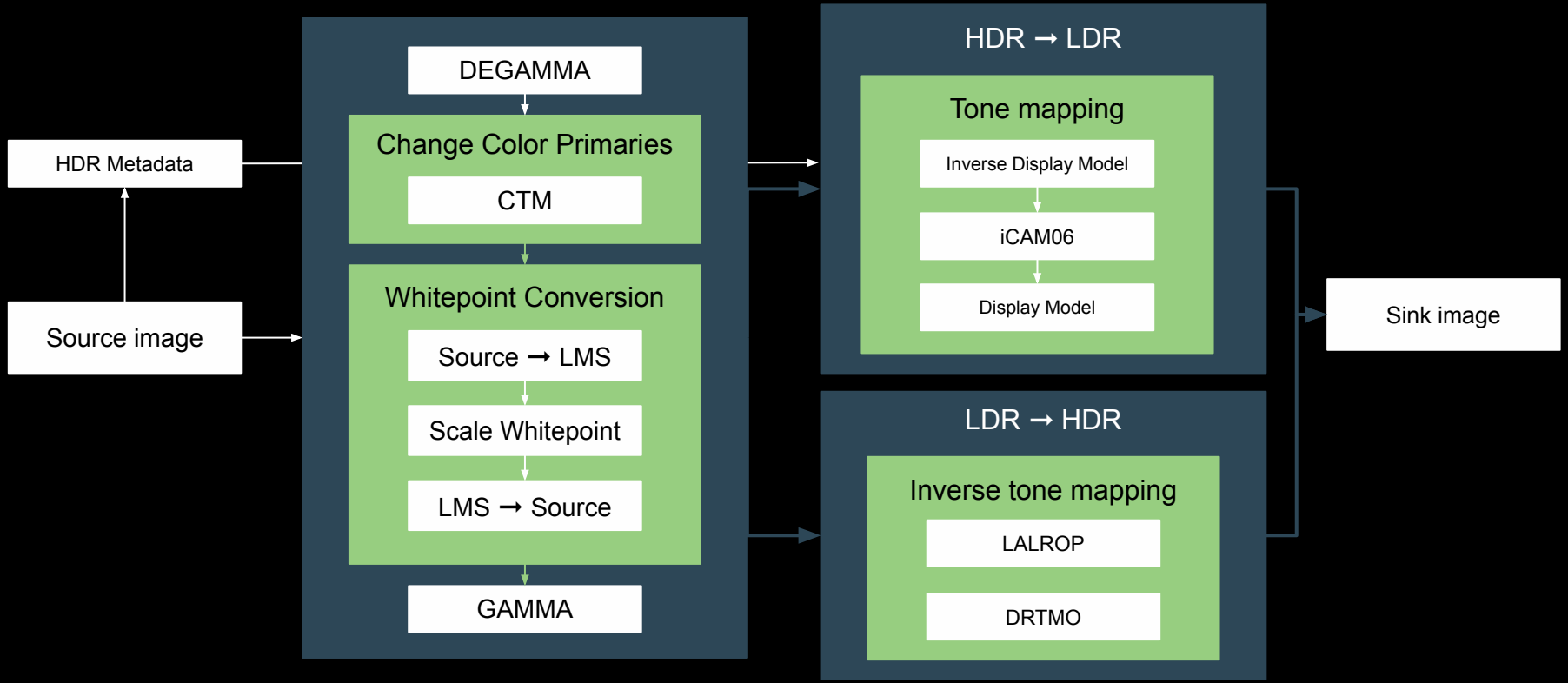
ADVANCED APPLICATION DESIGN, INC.

	EOTF	Inverse EOTF (OETF)
SRGB	$F_{\text{srgb}}(L) = \begin{cases} (12.92)L & \text{if } L \leq 0.0031308 \\ \left(1 + 0.55\right)L^{1/\gamma} - 0.55 & \text{otherwise} \end{cases}$	$F^{-1}_{\text{srgb}}(L) = \begin{cases} \frac{L}{12.92} & \text{if } L \leq 0.04045 \\ \left(\frac{L+0.55}{1+C}\right)^{\gamma} & \text{otherwise} \end{cases}$
Rec. 709	$F_{\text{rec709}}(L) = \begin{cases} (4.5)L & \text{if } L < 0.018 \\ 1.099L^{0.45} - 0.099 & \text{otherwise} \end{cases}$	$F^{-1}_{\text{rec709}}(L) = \begin{cases} \frac{V}{4.5} & \text{if } L < 0.081 \\ \left(\frac{V+0.099}{1.099}\right)^{1/0.45} & \text{otherwise} \end{cases}$
Rec. 2100	$F_{\text{PQ}}(L) = 10000\left(\frac{\max\left[\left(L^{1/m_2} - \frac{107}{128}\right), 0\right]}{\frac{2413}{128} - \frac{2392}{128}L^{1/m_2}}\right)^{1/m_2}$ $m_1 = \frac{1305}{8192} \quad m_2 = \frac{2523}{32}$	$F^{-1}_{\text{PQ}}(L) = \left(\frac{\frac{107}{128} + \frac{2413}{128}\left(\frac{L}{10000}\right)^{m_1}}{1 + \frac{2392}{128}\left(\frac{L}{10000}\right)^{m_1}}\right)^{m_2}$

COLOR VOLUME CONVERSION

Advanced Application Design, Inc.

Map colors from a source color volume to a sink color volume such that they look the same, or better



CHANGE COLOR PRIMARIES

- Generate a Color Transformation Matrix (CTM)
- Find a matrix $[M_{\text{source}}]$ for converting source colorspace to xyz colorspace
- Find a matrix $[M_{\text{sink}}]$ for converting sink colorspace to xyz colorspace
- $M_{\text{ctm}} = [M_{\text{source}}]\,[M_{\text{sink}}]^{-1}$

CHANGE COLOR PRIMARIES

Let $(x_r, y_r), (x_g, y_g), (x_b, y_b)$ be the color primaries of our RGB colorspace

- Calculate contribution of each color primary on X

$$X_b = \frac{x_b}{y_b} \quad X_r = \frac{x_r}{y_r} \quad X_g = \frac{x_g}{y_g}$$

- The contribution of each color primary on Y (luma) is 1

$$Y_r, Y_g, Y_b = 1$$

- Calculate contribution of each color primary on Z

$$Z_r = \frac{1 - x_r - y_r}{y_r} \quad Z_g = \frac{1 - x_g - y_g}{y_g} \quad Z_b = \frac{1 - x_b - y_b}{y_b}$$

CHANGE COLOR PRIMARIES

Let (x_w, y_w, z_w) be the white point of our RGB colorspace

- Convert white point separately, then create RGB to XYZ matrix

$$\begin{bmatrix} W_r \\ W_g \\ W_b \end{bmatrix} = \begin{bmatrix} X_r & X_g & X_b \\ Y_r & Y_g & Y_b \\ Z_r & Z_g & Z_b \end{bmatrix}^{-1} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} \qquad [M] = \begin{bmatrix} W_r X_r & W_g X_g & W_b X_b \\ W_r Y_r & W_g Y_g & W_b Y_b \\ W_r X_r & W_g Z_g & W_b Z_b \end{bmatrix}$$

- Create CTM from source and sink RGB to XYZ matrices

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = [M] \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad M_{\text{ctm}} = [M_{\text{source}}] \, [M_{\text{sink}}]^{-1}$$

WHITE POINT CONVERSION

Generate a *Chromatic Adaptation Matrix* $[M]$

- Converts a source color in some XYZ colorspace with some whitepoint to a sink color in XYZ colorspace with a new whitepoint

$$\begin{bmatrix} X_D \\ Y_D \\ Z_D \end{bmatrix} = [M] \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix}$$

- Let (X_S, Y_S, Z_S) be a source color
- Let (X_D, Y_D, Z_D) be a sink color

WHITE POINT CONVERSION

- Convert a source color (X_S, Y_S, Z_S) to a sink color (X_D, Y_D, Z_D)
- Let $(X_{W,S}, Y_{W,S}, Z_{W,S})$ be the source white point
- Let $(X_{W,D}, Y_{W,D}, Z_{W,D})$ be the destination white point

$$\begin{bmatrix} L_S \\ M_S \\ S_S \end{bmatrix} = [M_A] \begin{bmatrix} X_{W,S} \\ Y_{W,S} \\ Z_{W,S} \end{bmatrix} \qquad \begin{bmatrix} L_D \\ M_D \\ S_D \end{bmatrix} = [M_A] \begin{bmatrix} X_{W,D} \\ Y_{W,D} \\ Z_{W,D} \end{bmatrix}$$

- Here $[M_A]$ is the Bradford, Von Kries, or Hunt-Pointer-Estevez matrix

WHITE POINT CONVERSION

- Let $(X_{W,S}, Y_{W,S}, Z_{W,S})$ be the source white point
- Let $(X_{W,D}, Y_{W,D}, Z_{W,D})$ be the destination white point

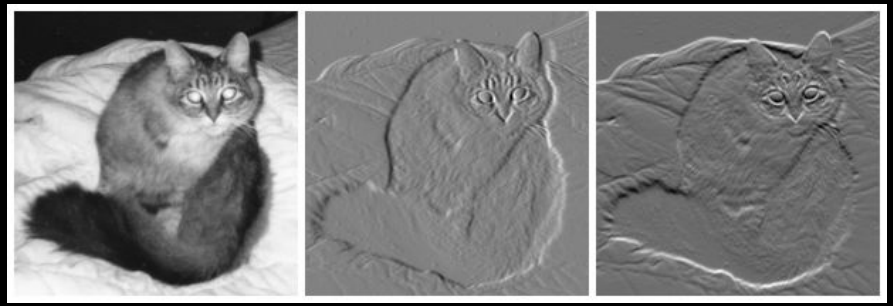
$$[M] = [M_A]^{-1} \begin{bmatrix} \frac{L_D}{L_S} & 0 & 0 \\ 0 & \frac{M_D}{M_S} & 0 \\ 0 & 0 & \frac{S_D}{S_S} \end{bmatrix} [M_A] \qquad \begin{bmatrix} X_D \\ Y_D \\ Z_D \end{bmatrix} = [M] \begin{bmatrix} X_S \\ Y_S \\ Z_S \end{bmatrix}$$

- Here $[M_A]$ is the Bradford, Von Kries, or Hunt-Pointer-Estevez matrix

TONE MAPPING (HDR ➡ LDR)

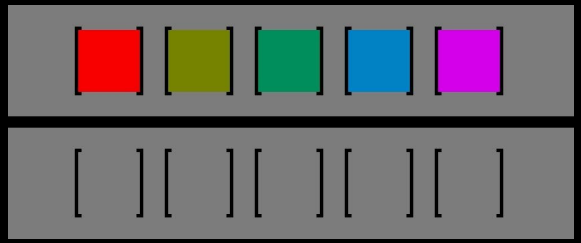
- **Tone:** Gradient of luminosity
- **Tone Map:** Transformation of higher dynamic luminosity gradient to lower dynamic luminosity gradient
  - f is our source image gradient
  - φ is our mapped gradient
  - $\nabla^2 = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)$ is the divergence of our source and sink gradients

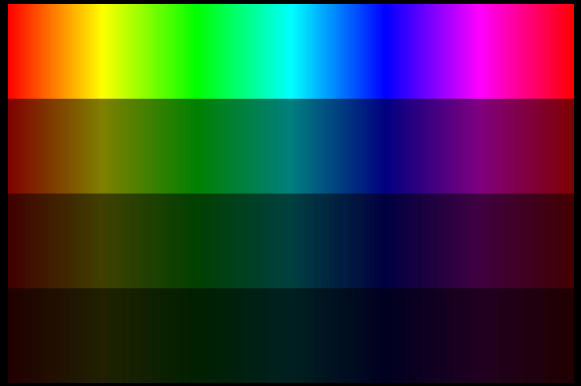
$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\varphi(x, y) = f(x, y)$$



TONE MAPPING (HDR ➡ LDR)

- We won’t have a detailed destination gradient
- The human brain confabulates information
  - Helmholtz–Kohlrausch effect
    - Hues with constant luma are not perceived with constant luma
  - Stevens effect
    - Perceived contrast increases with luma
  - Hunt effect
    - Perceived chroma increases with luma





- iCAM06 is the standard for tone mapping
  - Though, there are others
  - Histogram equalization, sigmoid matching, local average, etc.

SMPTE 2084

Advanced Application Design, Inc.

- Tone mapping requires accounting for display discrepancies
  - *Mastering display* is the display content was created on
  - *Target display* is the user's display

- Display/content properties provided as SMPTE 2084 metadata
  - Standard for HDR10/HDR10+ metadata
  - EOTF: SMPTE ST 2084 (PQ, HLG)
  - Min, max resolution
  - Bit Depth in the range `[10, 16]`
  - Color primaries: ITU-R BT.2020
  - Maximum luminosity of 10,000 cd/m2
  - **Metadata:** Mastering Display Color Volume Metadata
  - **Metadata:** MaxCLL, MaxFALL

INVERSE DISPLAY MODEL

ADVANCED APPLICATION DESIGN, INC.

- Mastering Display Color Volume Metadata
  - Chromaticity of whitepoint (CIE xyz)
  - Color primaries (CIE xyz)
  - Max/Min luminance in cd/m2

- **MaxCLL :** Maximum Content Light Level
  - E_{amb} is ambience in lux (20,000 for midday)
  - k is reflectance factor (0.01)
  - L_{peak} is MaxCLL of the mastering display

$$L_{\text{refl}} = \frac{k}{\pi} E_{\text{amb}}$$

$$V = F_{\text{eotf}}^{-1}\left(\frac{L - L_{\text{black}} - L_{\text{refl}}}{L_{\text{peak}} - L_{\text{black}}}\right)$$

- **MaxFLL :** Maximum Frame Average Light Level
  - May be used in tone mapping algorithms

FORWARD DISPLAY MODEL

ADVANCED APPLICATION DESIGN, INC.

- Target Display Color Volume Metadata
  - Chromaticity of whitepoint (CIE xyz)
  - Color primaries (CIE xyz)
  - Max/Min luminance in cd/m2

- **MaxCLL :** Maximum Content Light Level
  - E_{amb} is ambience in lux (20,000 for midday)
  - k is reflectance factor (0.01)
  - L_{peak} is MaxCLL of the target display

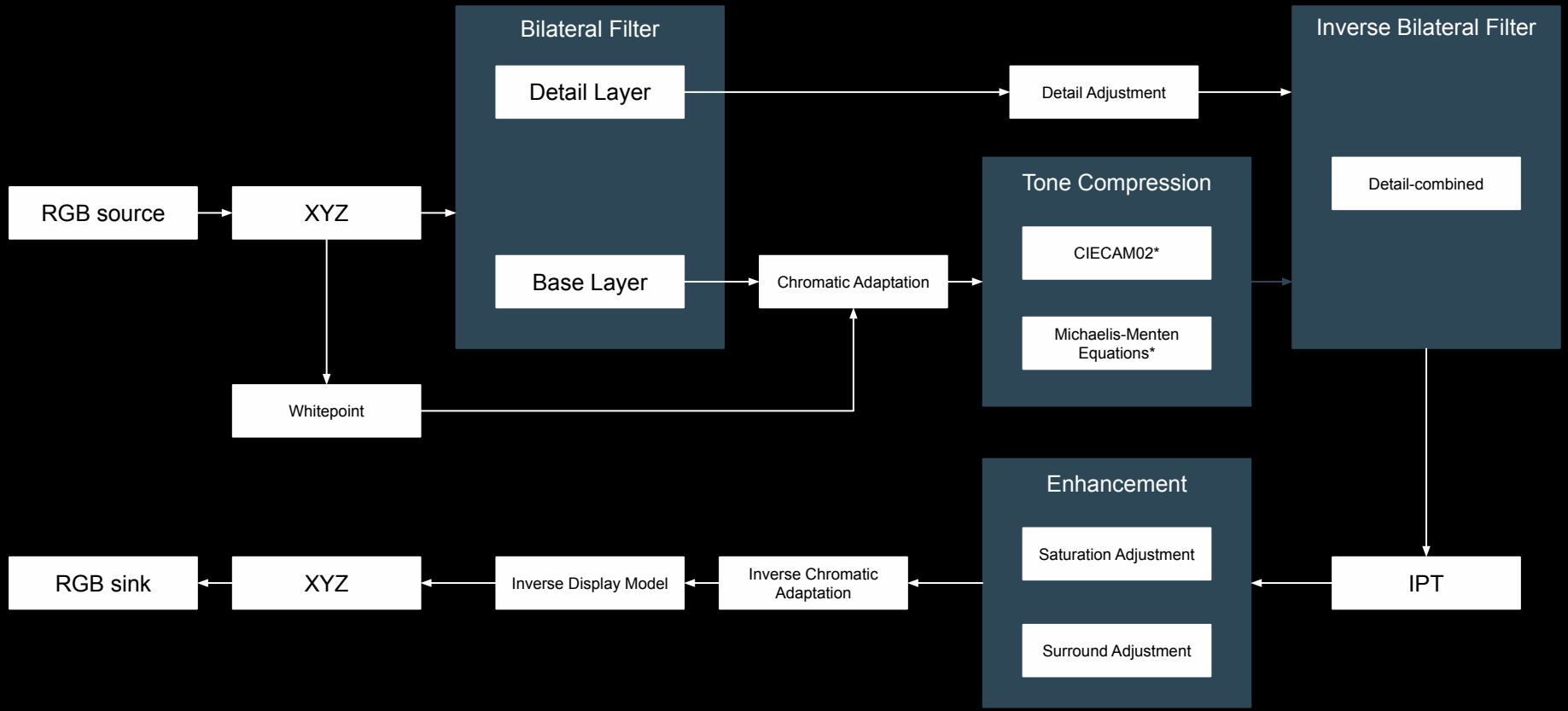
$$L_{\text{refl}} = \frac{k}{\pi} E_{\text{amb}}$$

$$L = (L_{\text{peak}} - L_{black}) F_{\text{eotf}}(V) + L_{\text{black}} + L_{\text{refl}}$$

- **MaxFLL :** Maximum Frame Average Light Level
  - May be used in tone mapping algorithms

1. iCAM06

ADVANCED APPLICATION DESIGN, INC.



INVERSE TONE MAPPING (LDR ➡ HDR)

- **Inverse Tone Map:** Transformation of lower dynamic luminosity gradient to higher dynamic luminosity gradient
  - Requires generating new information (i.e. upsampling image luminosity gradient)

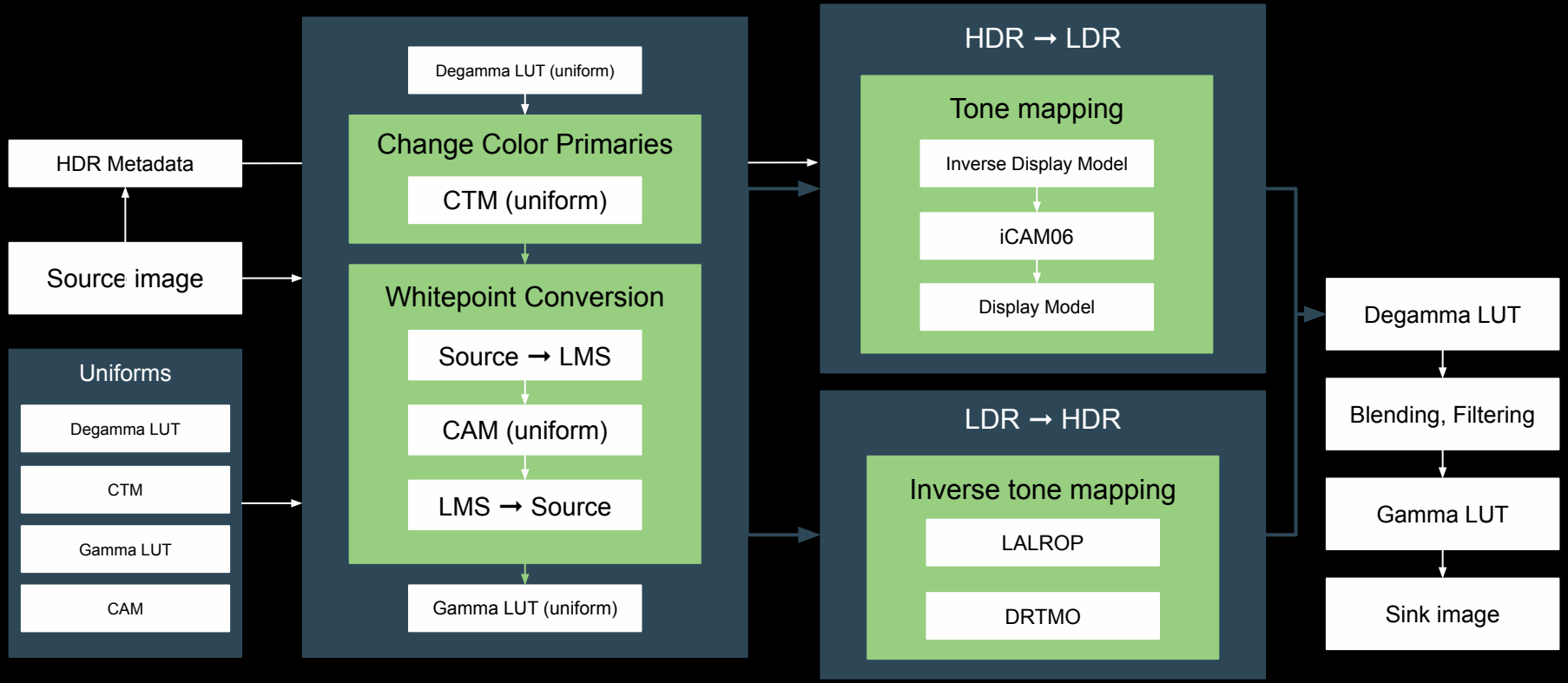
- AI methodologies are effective, but not necessarily performant
  - DRTMO (CNN implementation in Cuda exceeds 1gb in size), DREIS, etc.
    - Paucity of training data, so clever supplementations (camera exposures) are used

- Realtime methods are not as effective, but are performant
  - LALROP, ITMLUTs, etc.

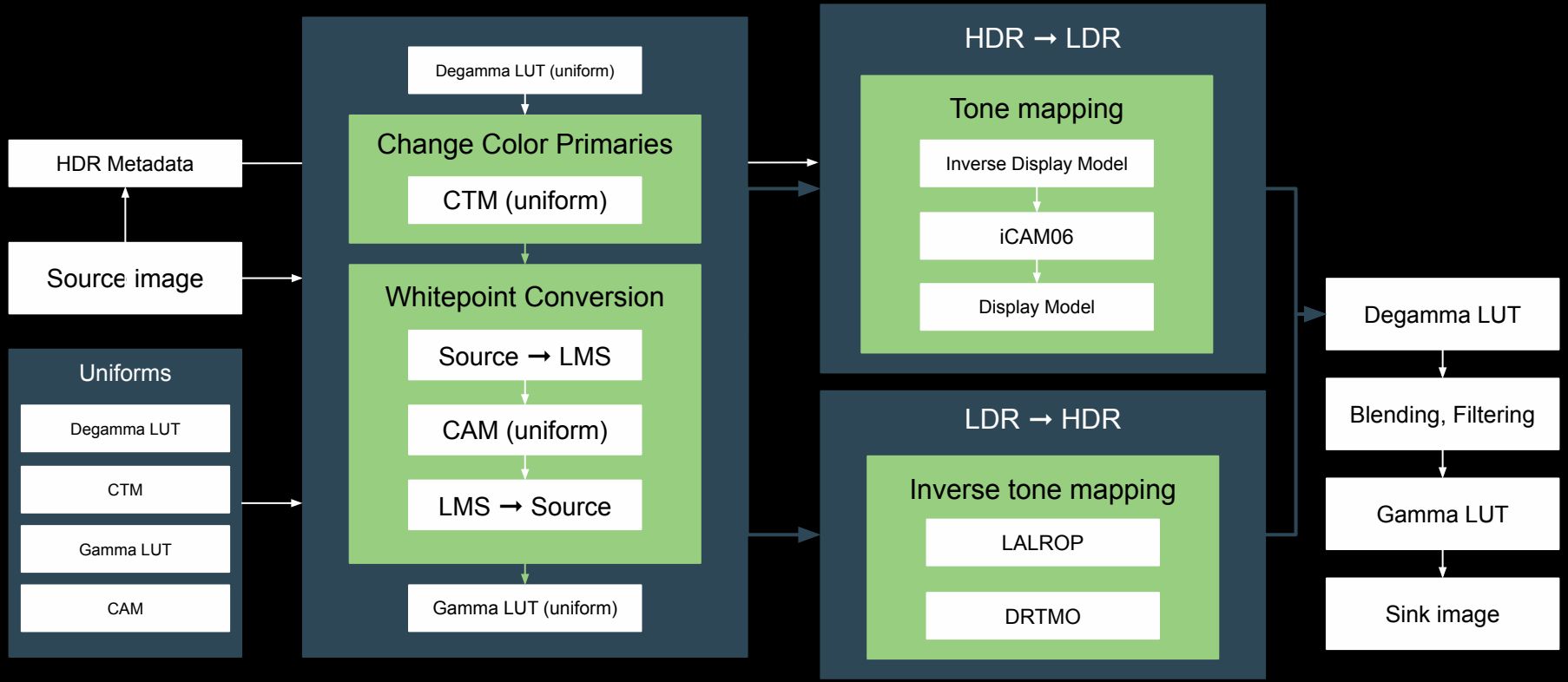
- `VK_FORMAT_X_N_UNORM`
  - Don’t let the driver apply an incorrect EOTF/OETF to your color volume
- `VkPipelineColorBlendAttachmentState::blendEnable = VK_FALSE`
  - You must manually implement color blending in your shader
- `VK_FILTER_NEAREST`
  - You must manually implement (bilinear) filtering in your shader

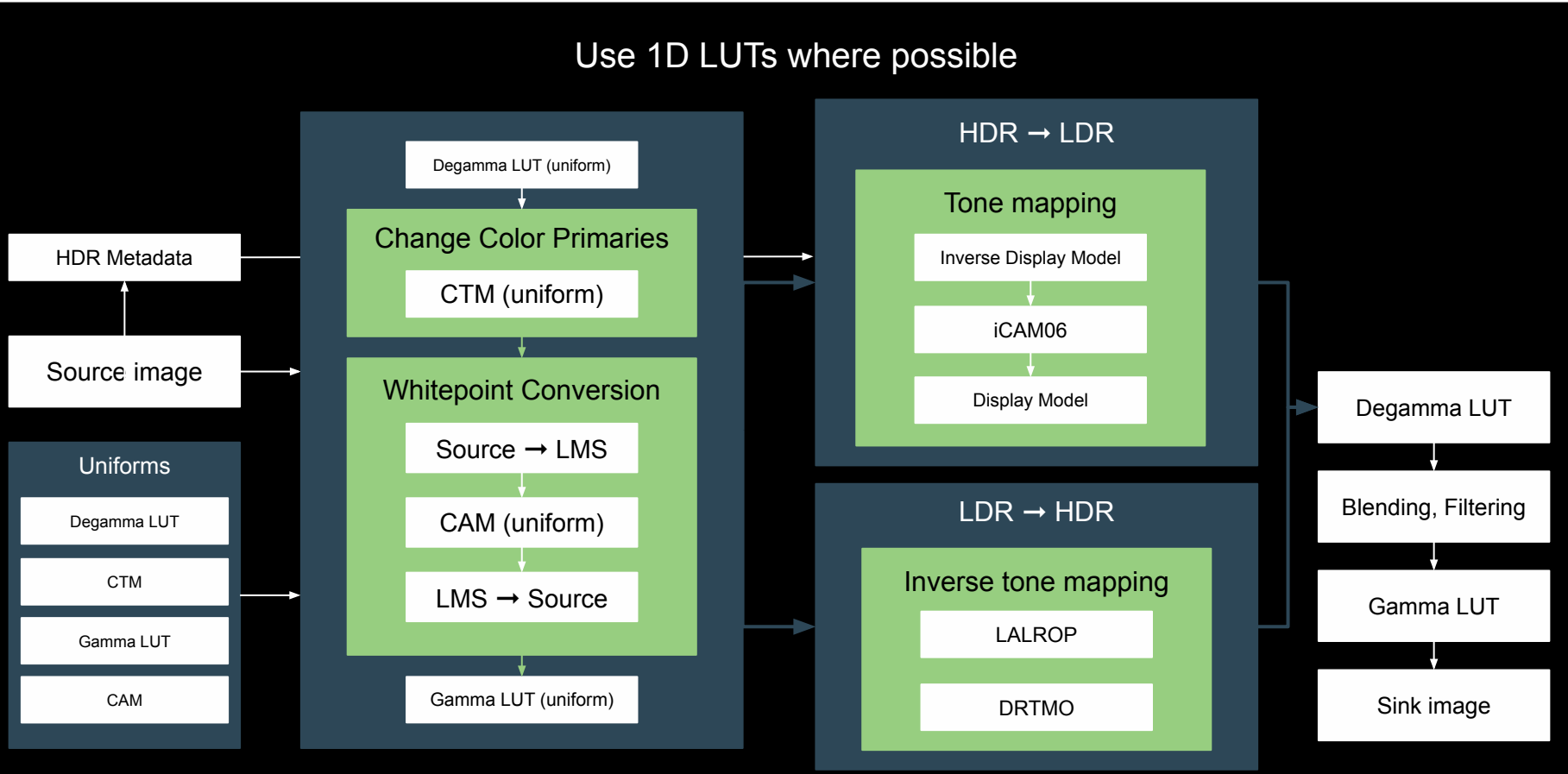
VULKAN

If your Vulkan application is not a direct renderer, then sink color volume is set by your WM or compositor



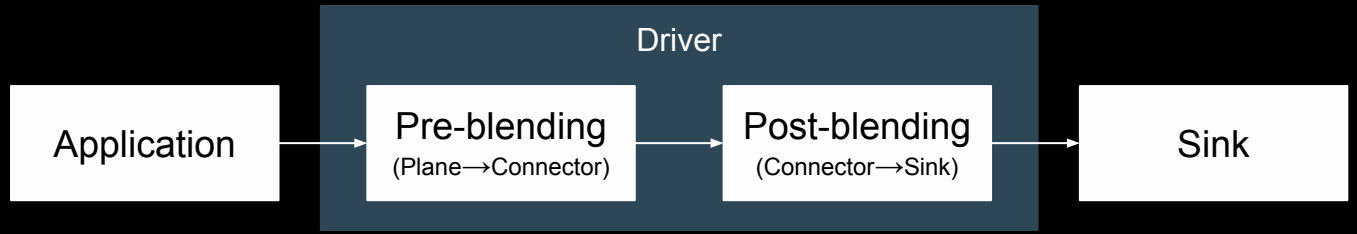
Convert to sink (`VkSwapchainCreateInfoKHR::imageColorSpace`) colorspace




Use 1D LUTs where possible
HDR Metadata
Source image
Uniforms
Degamma LUT
CTM
Gamma LUT
CAM
Degamma LUT (uniform)
Change Color Primaries
CTM (uniform)
Whitepoint Conversion
Source → LMS
CAM (uniform)
LMS → Source
Gamma LUT (uniform)
HDR → LDR
Tone mapping
Inverse Display Model
iCAM06
Display Model
LDR → HDR
Inverse tone mapping
LALROP
DRTMO
Degamma LUT
Blending, Filtering
Gamma LUT
Sink image

DIRECT RENDERERS (LINUX)

Where do we handle color management?

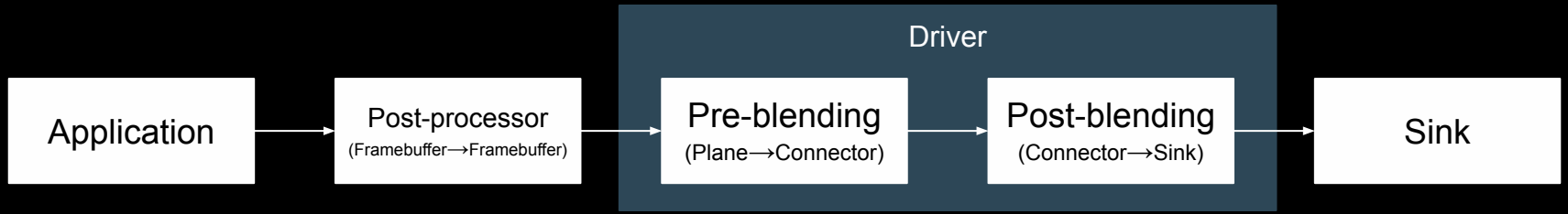


The only apparent option is in the pre-blending phase.

- Modify the colorspace before blending
- Send info frames over wire describing your colorspace

DIRECT RENDERERS (LINUX)

There is another option.

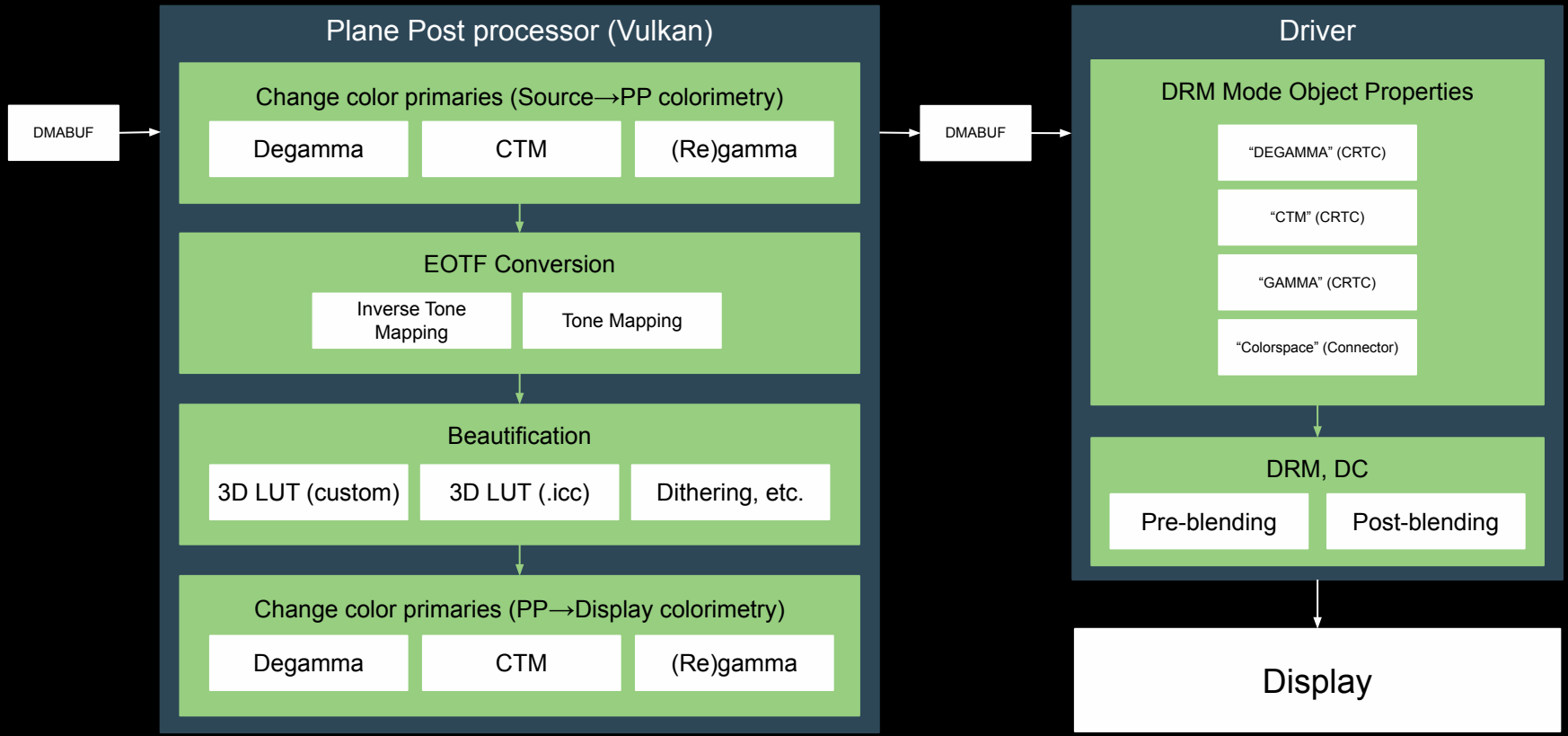


Add a “just in time” post-processing layer.

- Leverage driver features where available
- Otherwise, intercept the DMABUF as if we’re a zero copy video player
- Modify color volumes here, but remember to send info frames later

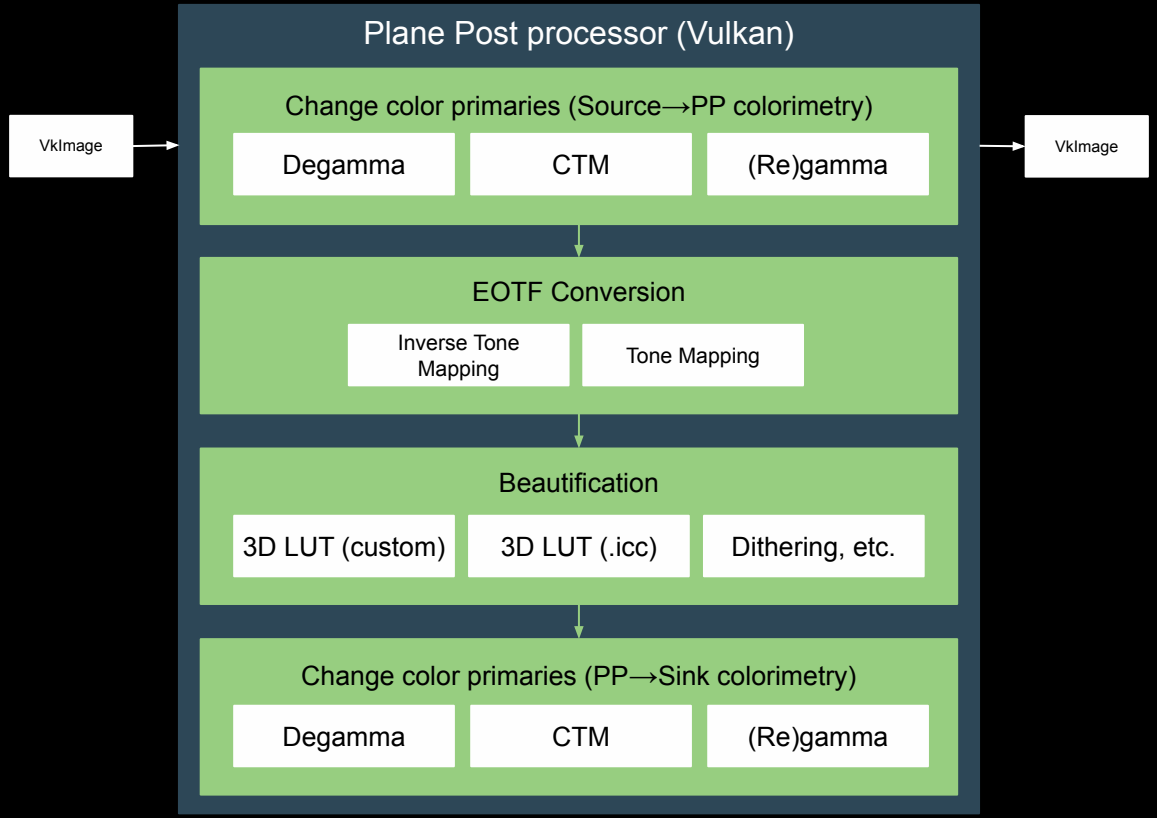
METHODOLOGY

ADVANCED APPLICATION DESIGN, INC.



METHODOLOGY

ADVANCED APPLICATION DESIGN, INC.



METHODOLOGY (COMPUTE SHADER)

- `VK_EXT_external_memory_dma_buf` is used to import DMABUFs
  - **NOTE:** Linux has no userland API for DMABUF attachment synchronization on the same device

- DRM format modifiers *must* be respected
  - Y-tiling, CCS planes, etc.
  - DRM format modifier used for scanout buffers is (usually) optimal for compute shaders
  - Mesa's libraries for handling these are not userland facing

- A GLSL compute shader is used to do our post processing
  - Ping pong rendering in a fragment shader is doable, but slower
  - Storage buffers are used, not storage images
    - `imageLoad()` with DRM format modifiers is buggy

- Convert from 8bit to 10bit, or 12bit color space, as needed
  - HDR requires *at least* 10 bit color

METHODOLOGY (DRM / KMS)

ADVANCED APPLICATION DESIGN, INC.

- The minimally necessary DRM mode object properties are `DEGAMMA`, `GAMMA`, `CTM`
  - The driver should respect them for plane, CRTC blending
  - Your monitor will not know it's in HDR, even though it is
  - `CTM` does not use ieee754, nor two's complement
  - `DEGAMMA`/`GAMMA` scaled to [0, 65536]

- Other (driver specific) DRM mode object properties are available
  - `NV_HDR_STATIC_METADATA`, `NV_INPUT_COLORSPACE` (Nvidia)
  - `HDR_OUTPUT_METADATA`, `HDR_SOURCE_METADATA` (Rockchip)
  - All of the rainbow color map (AMD)

- If your imported/exported Vulkan buffer has incorrect memory size, padding, or offset then:
  - Modesetting causes a kernel panic on AMD Vega GPUs
  - Modesetting causes the system to hang indefinitely on NVIDIA RTX 30 series (and up) GPUs
    - NVIDIA requires buffers to be aligned to 1024, but doesn't always return EINVAL

LIBDRM INTEGRATION

- Integrated into a meta-implementation of libdrm

- Libdrm is effectively augmented with post processing capabilities
  - “`drmModeAtomicCommit`” performs our post processing, and its original functionality

```
int drmModeAtomicCommit(int fd, drmModeAtomicReqPtr req, uint32_t flags, void* user_data) {
        for (int i = 0; i < req->plane_list_size; ++i) {
                uint32_t plane_id = req->plane_list[i];
                drmModePlanePtr plane = drmModeGetPlane(..., plane_id);
                post_process(..., plane->fb_id);
        }
        return original_drmModeAtomicCommit(fd, req, flags, user_data);
}
```

LIBDRM INTEGRATION

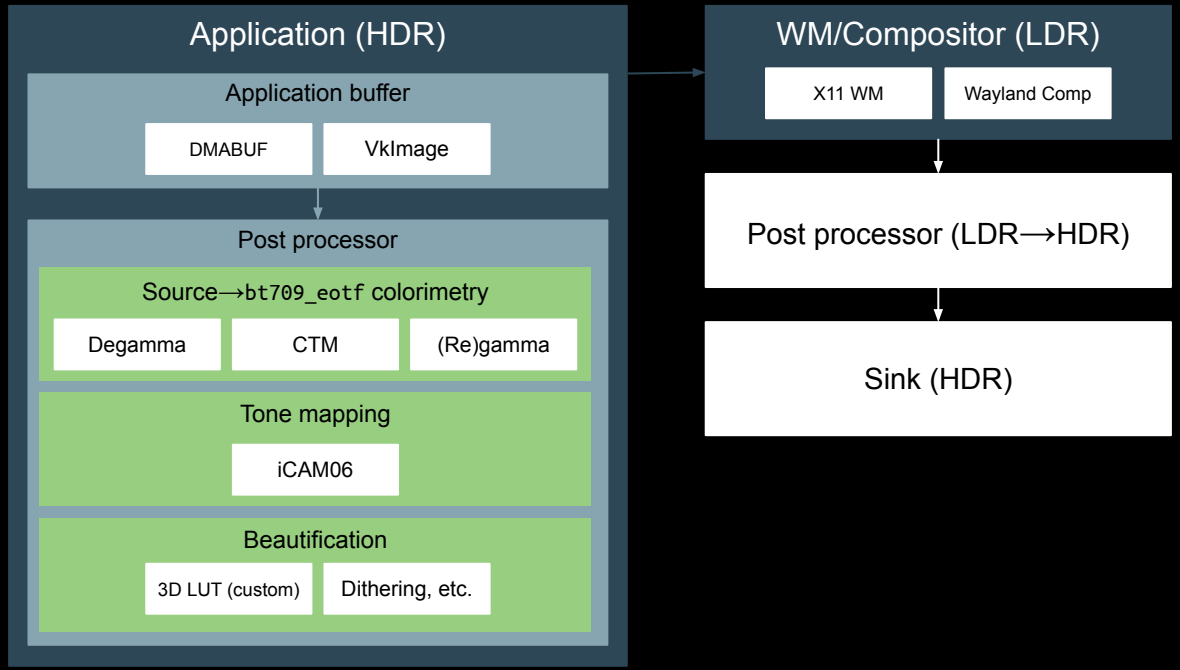
- Now any application that uses libdrm has advanced color capabilities!
  - LDR content (everything) can be inverse tone mapped to HDR

- We have a unix domain socket for advanced configuration
  - **`*_PLANE_SOURCE/SINK_COLOR_VOLUME`**: Specifying hardware plane source color volume
  - **`*_SOURCE/SINK_COLOR_VOLUME`**: Specifying global sink color volume
  - **`*_PLANE_3D_LUT`**: What 3D LUTs are applied to what hardware planes
  - **`*_INVTM_KIND`**: Specify inverse tone mapping algorithm

- Ideally, this should be handled internally by our window manager/compositor

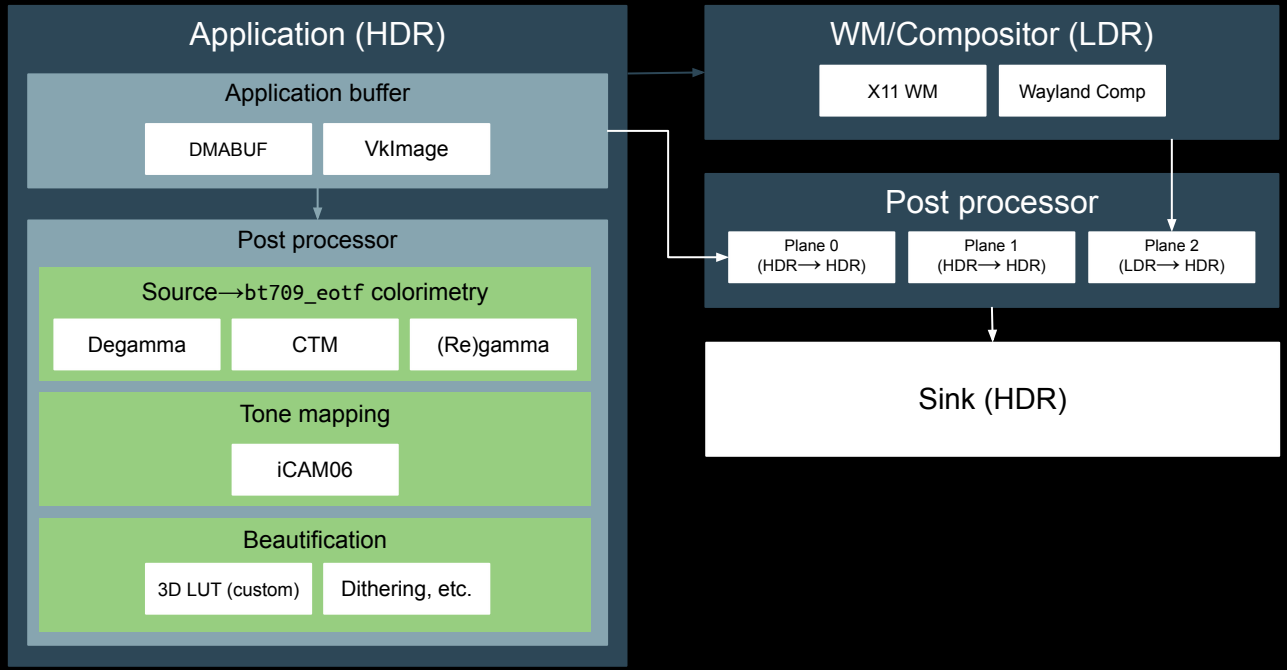
HDR CONTENT ON SDR WM/COMPOSITOR

Applications rendering HDR content can do so on SDR WMs and compositors.



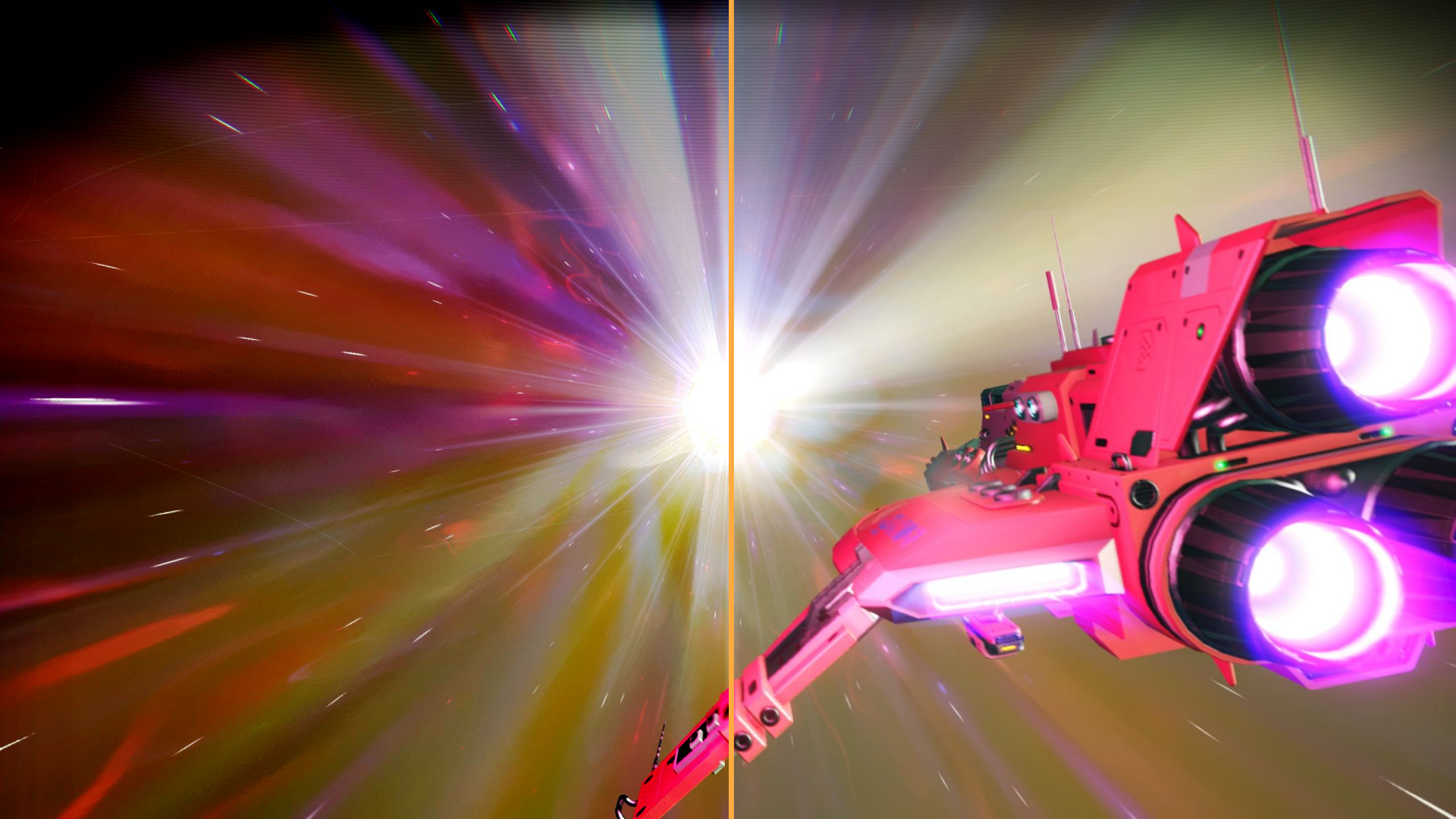
HDR CONTENT ON SDR WM/COMPOSITOR

Alternatively, duplicate our original buffer on a dedicated plane.



ADVANTAGES

- Surprisingly efficient after optimization
  - < 1ms to do all of our post processing, modesetting on i915, AMD, NVIDIA
- Cross-platform
  - Not limited to driver specific features (i.e. driver specific 3D LUT size)
  - Anything that uses `libdrm`
- Can act as a post processor on any buffer, not just a buffer intended for direct rendering
  - i.e. a Wayland client can use it
  - Anything that uses a `VKImage` with the correct image usage
- Even if your application content is LDR, we can make it look like it was made as HDR



LSP Error List

```
647 ∀ A : BoolCondAnf, (isAltResolvent Δ R A) → ¬(BoolCond.isTrivial A) := by
648     induction R
649     case empty =>
650         intro A H_A_altres
651         unfold isAltResolvent at H_A_altres
652         if H_A_clauses_in_Δ : BoolAnf.containsClauses Δ A then
653             apply if_A_clauses_in_Δ_and_Δ_nontriv_then_A_nontriv H_Δ_nontriv H_A_clauses_in_Δ
654         else
655             simp [H_A_clauses_in_Δ] at H_A_altres
656             if H_A_clauses_in_R : containsClauses empty A then
657                 unfold containsClauses at H_A_clauses_in_R
658                 contradiction
659             else
660                 simp [H_A_clauses_in_R] at H_A_altres
661                 if H_A_subs : isSubsidation Δ empty A then
662                     apply subs_A_not_trivial H_getAnf_Δ_R_SAT H_A_subs
663                 else
664                     simp [H_A_subs] at H_A_altres
665                     if H_A_recp : isReciprocal Δ empty A then
666                         unfold isReciprocal contains at H_A_recp; simp at H_A_recp
667                         match H_A_recp with
668                         | ⟨B, _, H_B_in_Δ, H_some_i⟩ =>
669                             match H_some_i with
670                             | ⟨i, _, _, j, H_recp_B_i_j_eq_A⟩ =>
671                                 have H_B_nontriv : ¬(BoolCond.isTrivial B) := by
672                                     apply BoolAnf.if_Δ_non_trivial_and_δ_in_Δ_then_δ_non_trivial
673                                         H_A_nontriv H_B_in_Δ
674                                 rw [←H_recp_B_i_j_eq_A]
675                                 apply BoolCond.if_X_nontriv_then_recp_X_nontriv H_B_nontriv
676                     else
677                         simp [H_A_recp] at H_A_altres
678                         unfold isAltResRuleResolvent at H_A_altres
679                         unfold contains at H_A_altres; simp at H_A_altres
680                         match H_A_altres with
681                         | ⟨U, H_U_in_Δ, H_some_V⟩ =>
683                             | ⟨V, _, _, H_altres_U_V⟩ =>
684                                 have H_U_nontriv : ¬(BoolCond.isTrivial U) := by
685                                     apply BoolAnf.if_Δ_non_trivial_and_δ_in_Δ_then_δ_non_trivial
686                                     repeat { assumption }
687                                 rw [←H_altres_U_V]
688                                 apply BoolCond.if_X_nontriv_then_altres_X_Y_nontriv H_U_nontriv
689     case next R' A' IH =>
690         have H_R'_altres : isAltResolutionDer Δ R' := by
691             apply if_next_R_A_altres_der_then_R_altres_der H_R_altres
692         have H_getAnf_Δ_R'_SAT : BoolConj.isSatisfiable (getAnf Δ R') := by
693             rw [getAnf_Δ_next_R_X_eq_AND_getAnf_Δ_R_X] at H_getAnf_Δ_R_SAT
694             unfold BoolConj.isSatisfiable at H_getAnf_Δ_R_SAT
695             match H_getAnf_Δ_R_SAT with
696             | ⟨S, H_S_int, H_AND_getAnf_Δ_R'_A'_sat⟩ =>
697                 unfold BoolConj.isSatisfiable
698                 exists S; simp [H_S_int]
699                 apply BoolConj.if_AND_X_x_sat_then_X_sat H_AND_getAnf_Δ_R'_A'_sat
700         simp [H_R'_altres, H_getAnf_Δ_R'_SAT] at IH
701         intro A H_A_altres
702         unfold isAltResolvent at H_A_altres
703         if H_A_clauses_in_Δ : BoolAnf.containsClauses Δ A then
704             apply if_A_clauses_in_Δ_and_Δ_nontriv_then_A_nontriv H_Δ_nontriv H_A_clauses_in_Δ
705         else
706             simp [H_A_clauses_in_Δ] at H_A_altres
707             if H_A_clauses_in_R : containsClauses (next R' A') A then
708                 unfold containsClauses at H_A_clauses_in_R
709                 if H_A_clauses_in_A' : BoolCond.containsClauses A' A then
710                     unfold isAltResolutionDer at H_R_altres
711                     specialize IH A'; simp [H_R_altres] at IH
712                     rw [←Bool.eq_false_eq_not_eq_true] at IH
713                     apply BoolCond.if_X_nontriv_then_X_non_empty at IH
714                     apply BoolCond.if_X_non_empty_and_A_clauses_in_X_then_A_nontriv IH H_A_clauses_in_A'
715                 else
716                     simp [H_A_clauses_in_A'] at H_A_clauses_in_R
717                     specialize IH A; unfold isAltResolvent at IH; simp [H_A_clauses_in_R] at IH
```

- proof 1 /home/rpe/projects/altsat
  - BoolProof.lean 1 altsat/Altsat/
    - ⓘ [Lean 4] declaration uses 'sorry' (2994:8)
- altsat 1 /home/rpe/projects/altsat/proof
- altsat 1 /home/rpe/projects

~/p/a/p/altsat/A/BoolProof.lean 672:0 17% LF UTF-8 Lean 4 ⊘ 1 Errors List

CONCLUSION

Color is complicated!

- Vulkan doesn’t support *rendering* images with PQ color volumes
- Vulkan does support *presenting* images with PQ color volumes
- **Ideal solution:** Inverse tone map as a post processing effect

Doing post processing immediately before scanout provides plenty of perks!

- All content automatically looks like it was made in HDR
- “Bad” HDR implementations using HLG are corrected to use PQ
- Correcting VRR image artifacts on OLED monitors
- Color correction and beautification of media (custom 3D luts)
- Stereoscopic rendering
- Xrandr for Wayland

Thank you! Cheers.