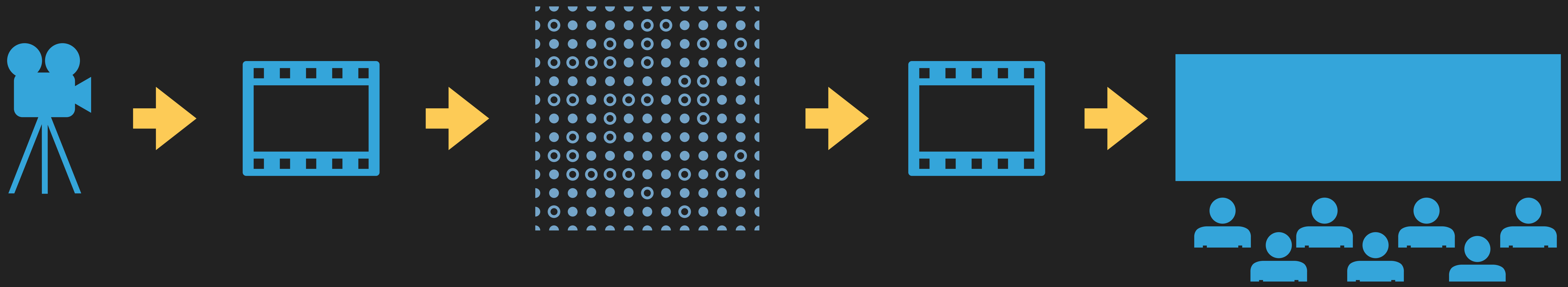
PIXAR
ANIMATION STUDIOS

OpenColorIO USD MATERIALX

NanoColor

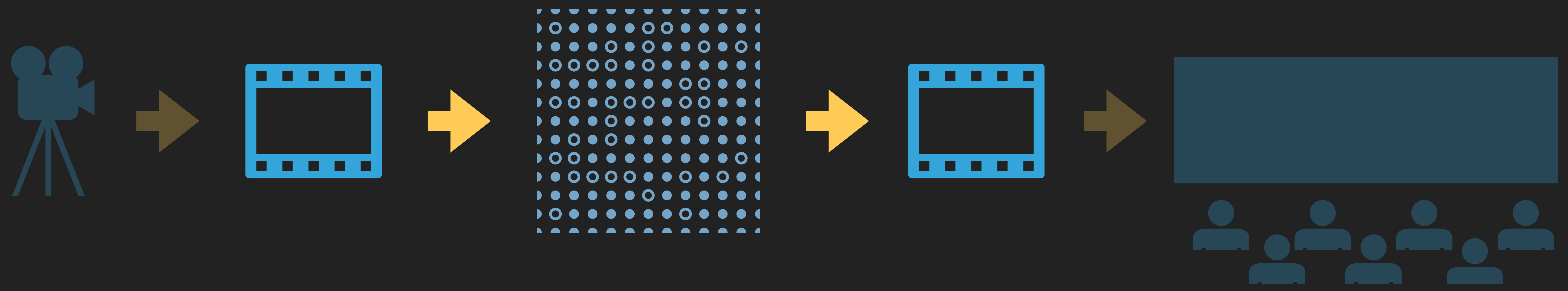
A VERY SMALL COLOR TRANSFORM LIBRARY

Nick Porcino & Rick Sayre



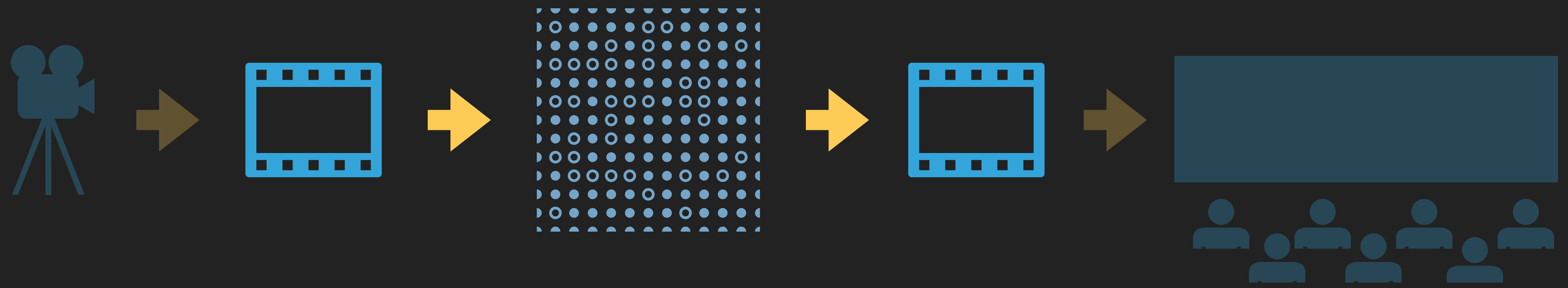
OpenColorIO

Give artists the power to manipulate and guide the transformation of color from capture to presentation



A renderer's input working space to output working space is an interesting subset

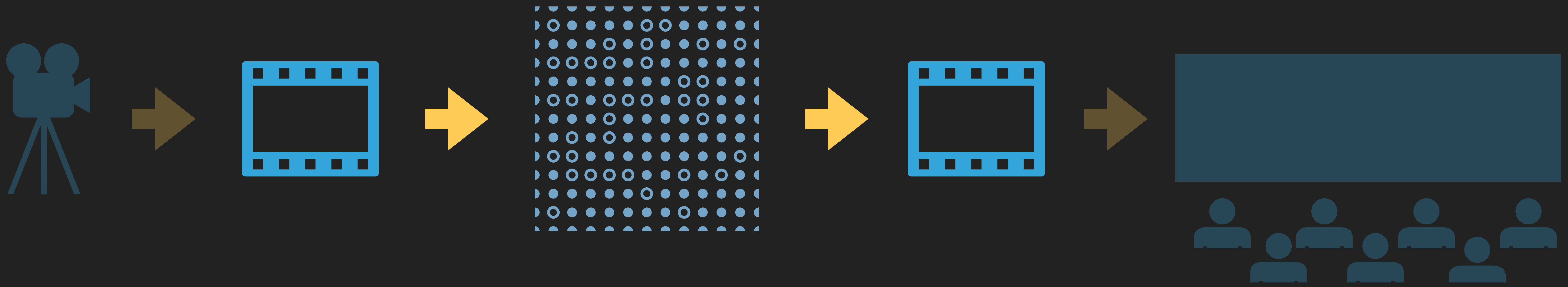
COLOR



MATERIALX

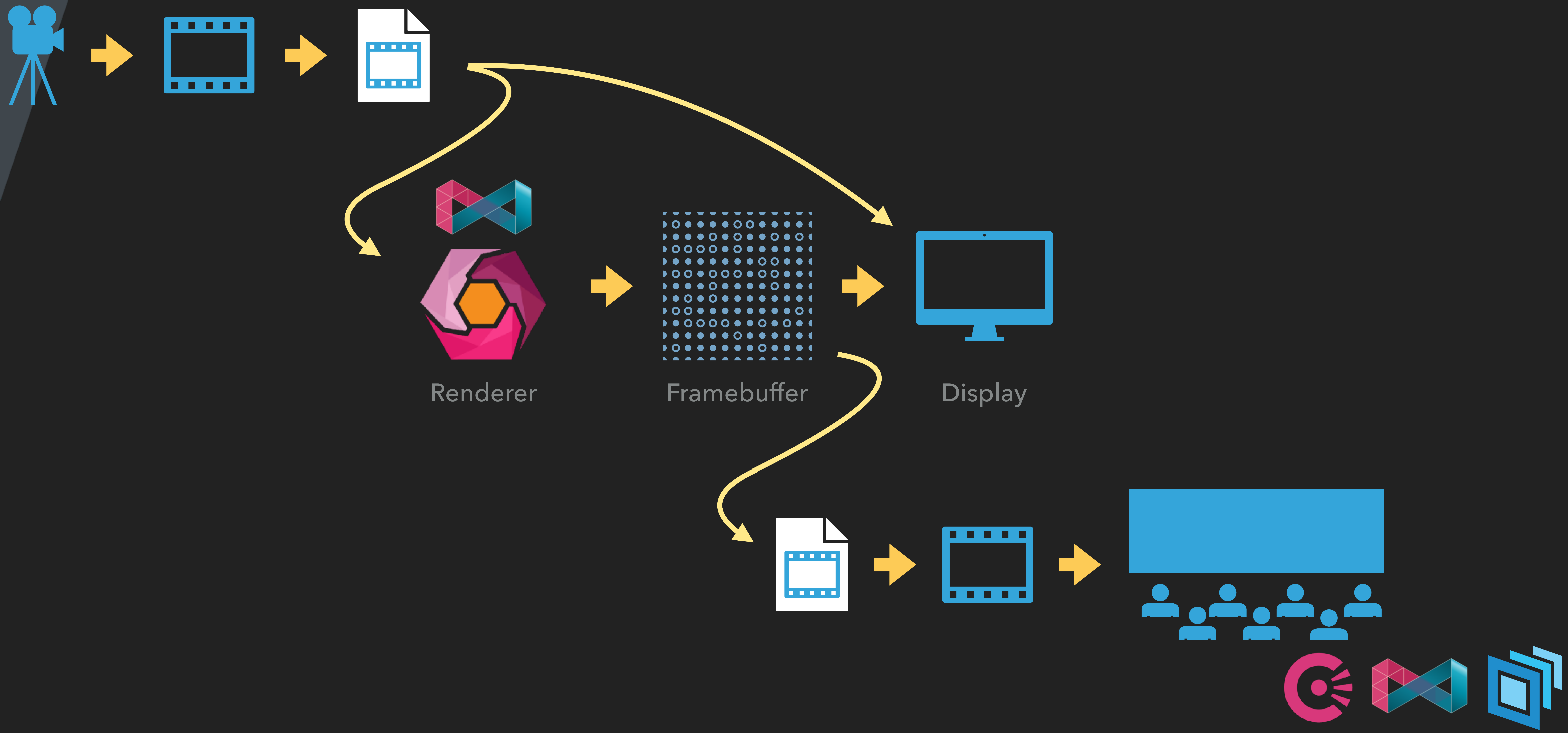
MaterialX describes surface appearances through a network of computational nodes, color appearance is a critical aspect of that

COLOR

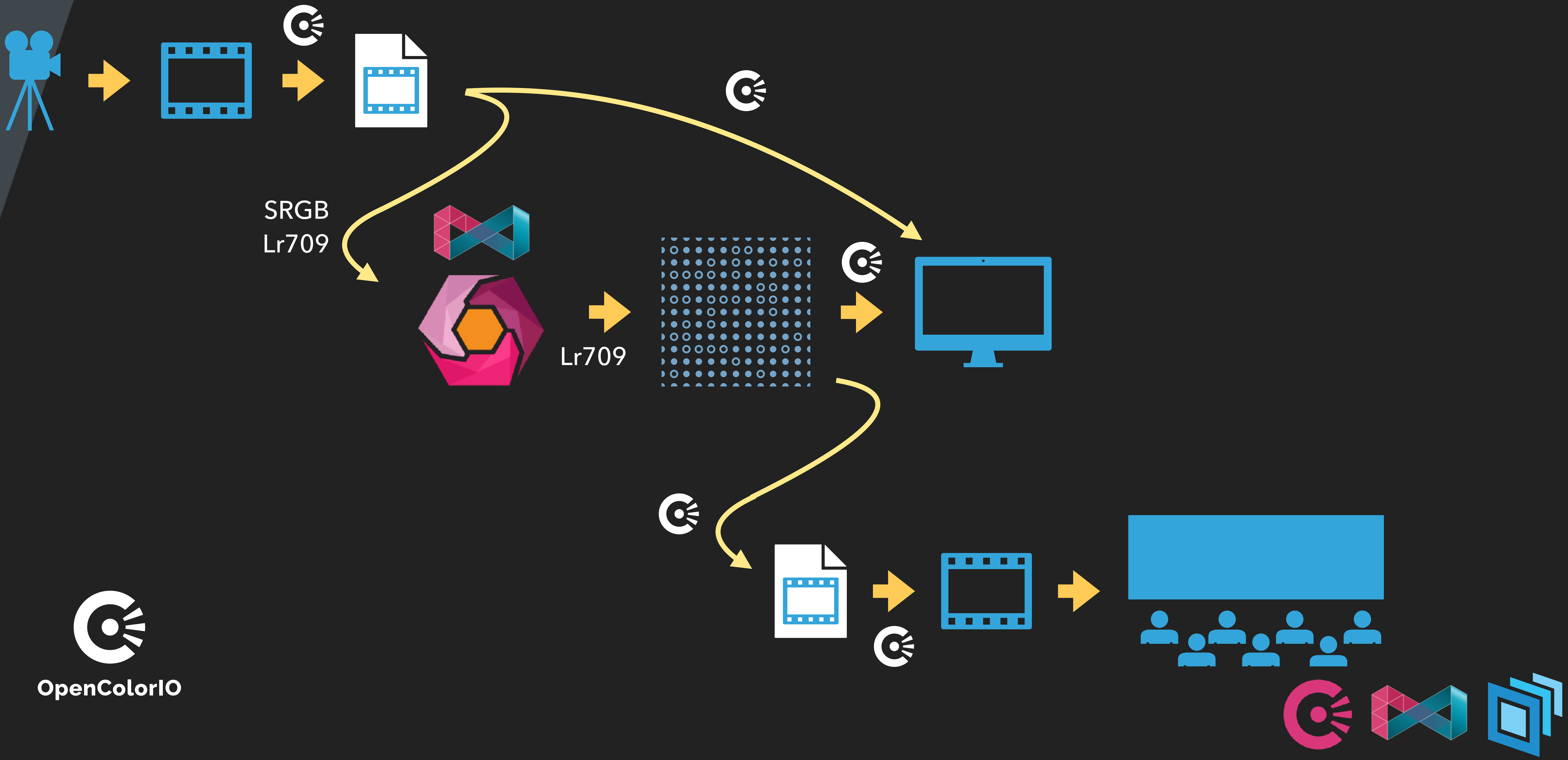


OpenUSD is the industry standard format for composing and transporting information about scenes, including color specifications of objects in those scenes

WHERE ARE THE COLOR TRANSFORMS?

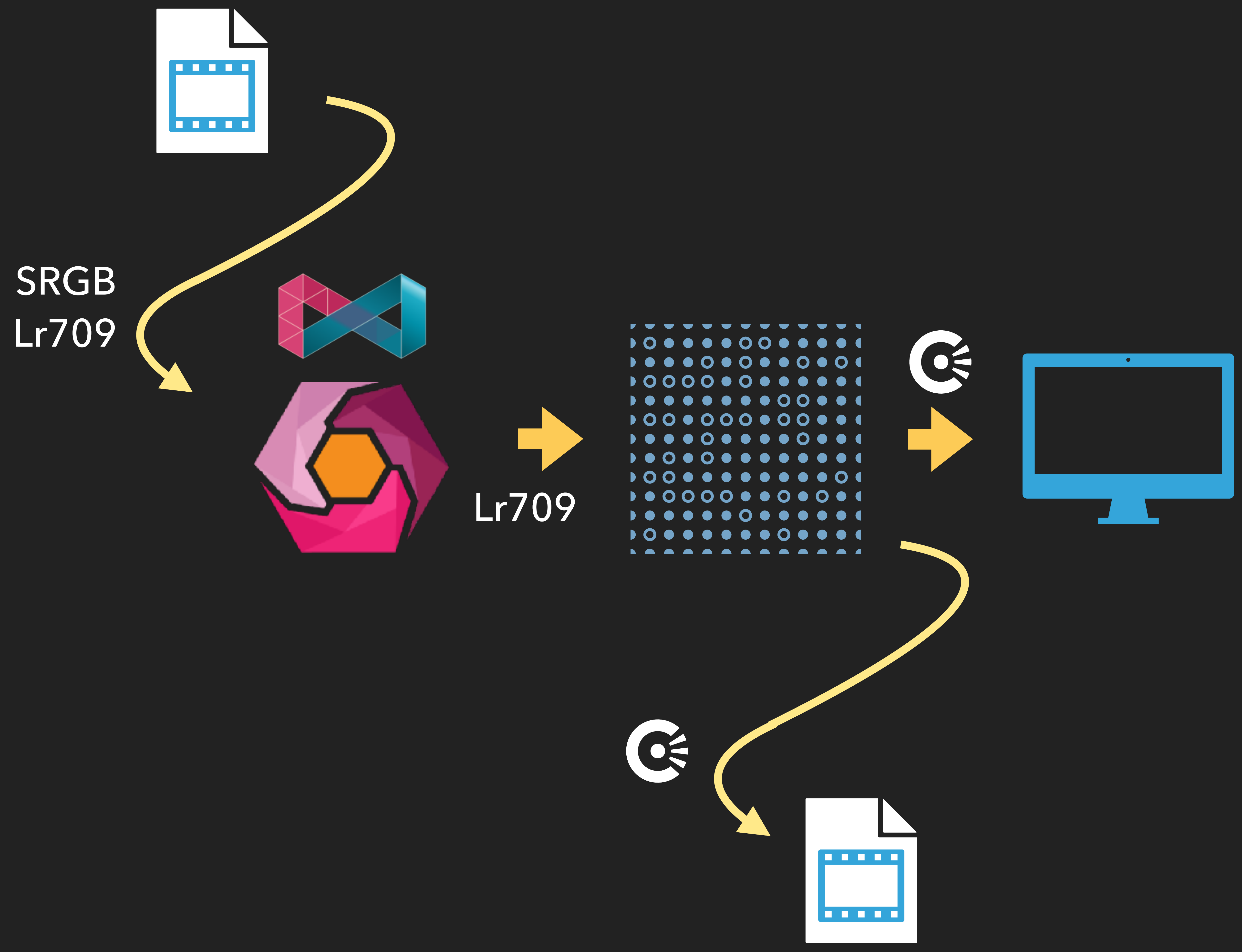


HOW ARE THEY MANAGED?

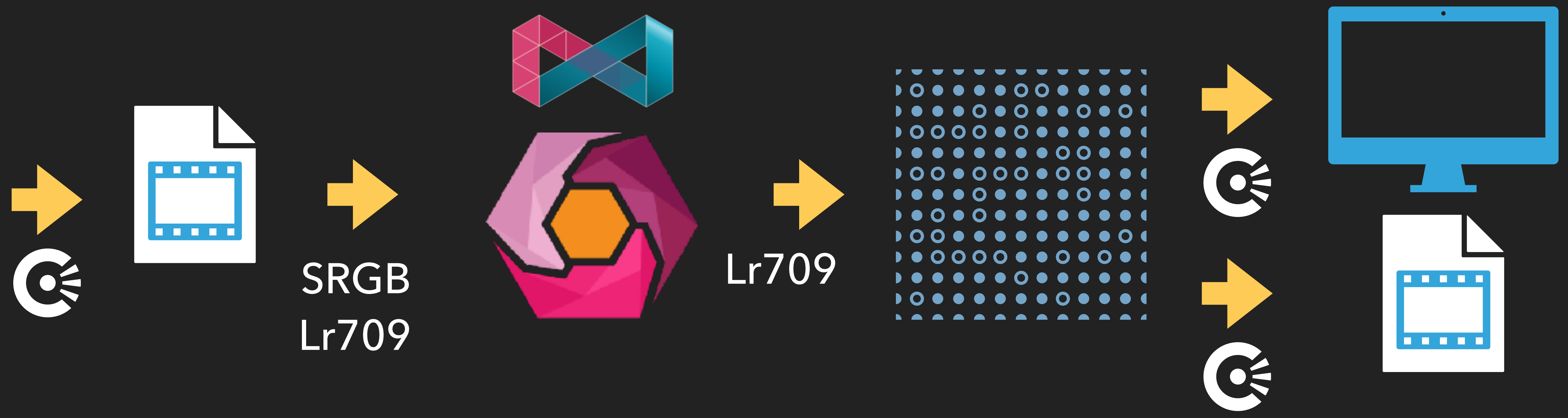


OpenColorIO

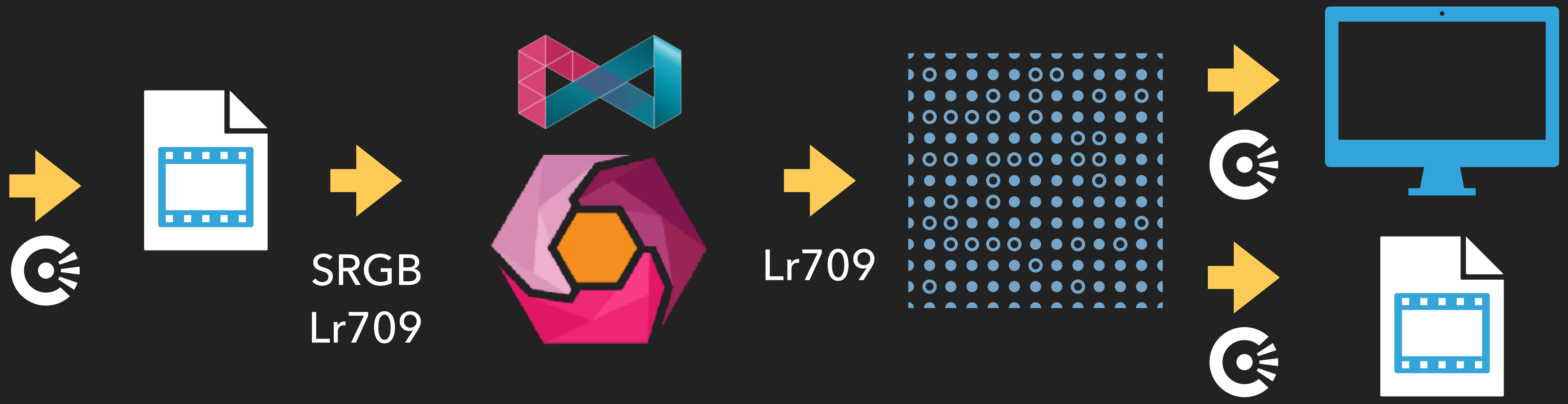
WHICH PARTS DO USD & HYDRA CARE ABOUT?



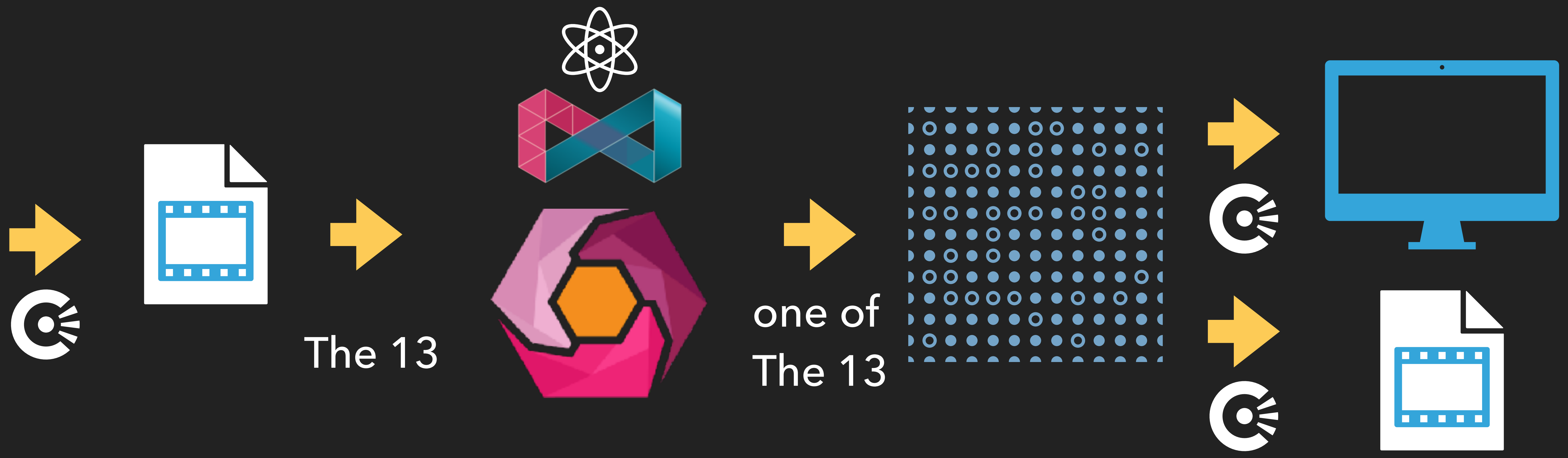
STRAIGHTEN THAT OUT A BIT



STRAIGHTEN THAT OUT A BIT

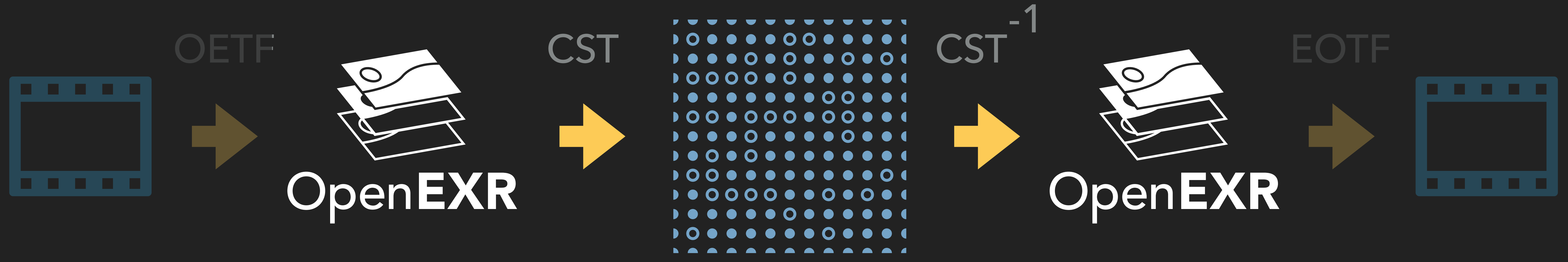


THIS IS WHERE NANOCOLOR COMES IN



nanocolor

COLOR



OETF

OpenEXR

CST

CST^{-1}

OpenEXR

EOTF

Chromaticities

Whitepoint

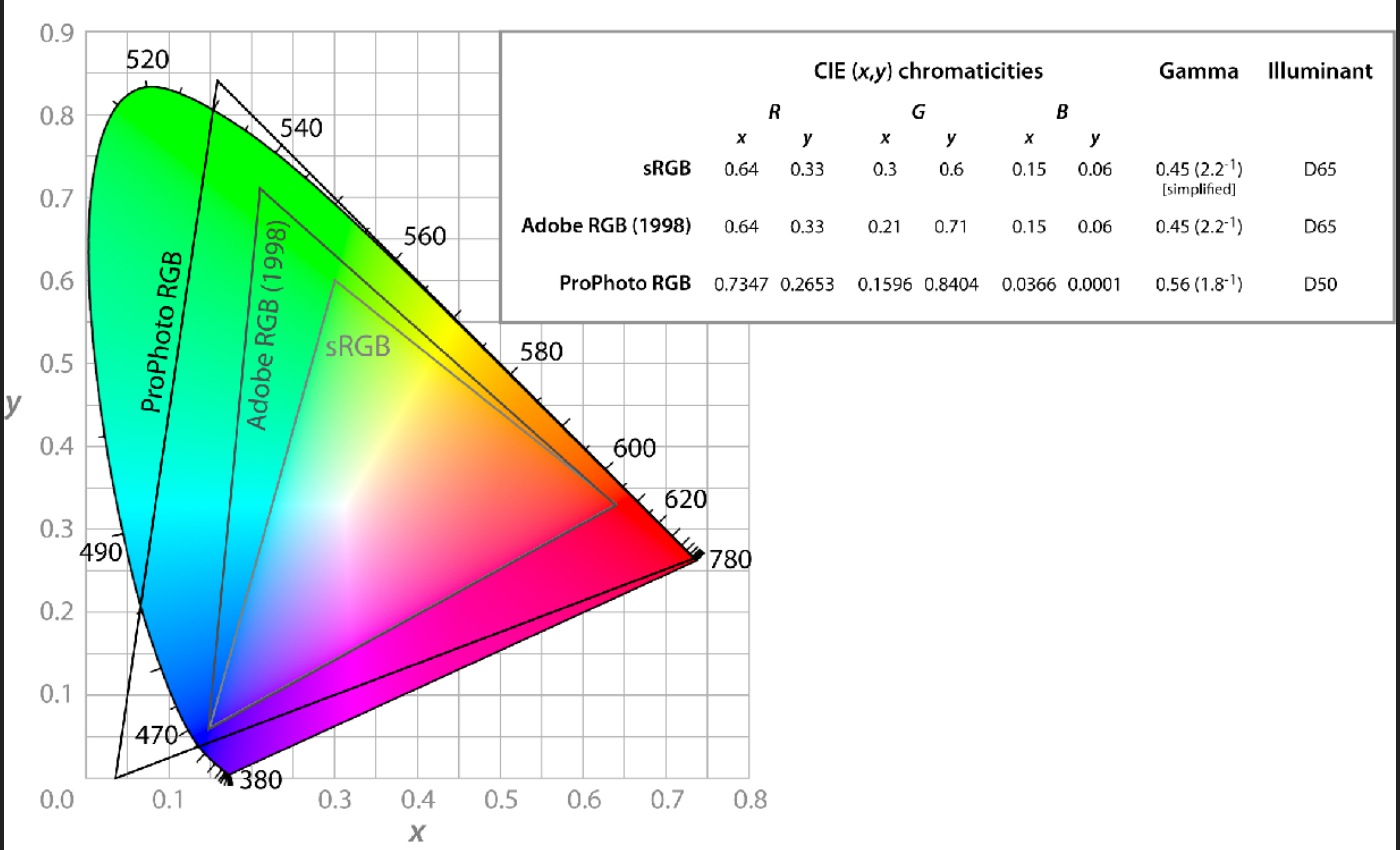
Working Color Space

Chromaticities

Whitepoint

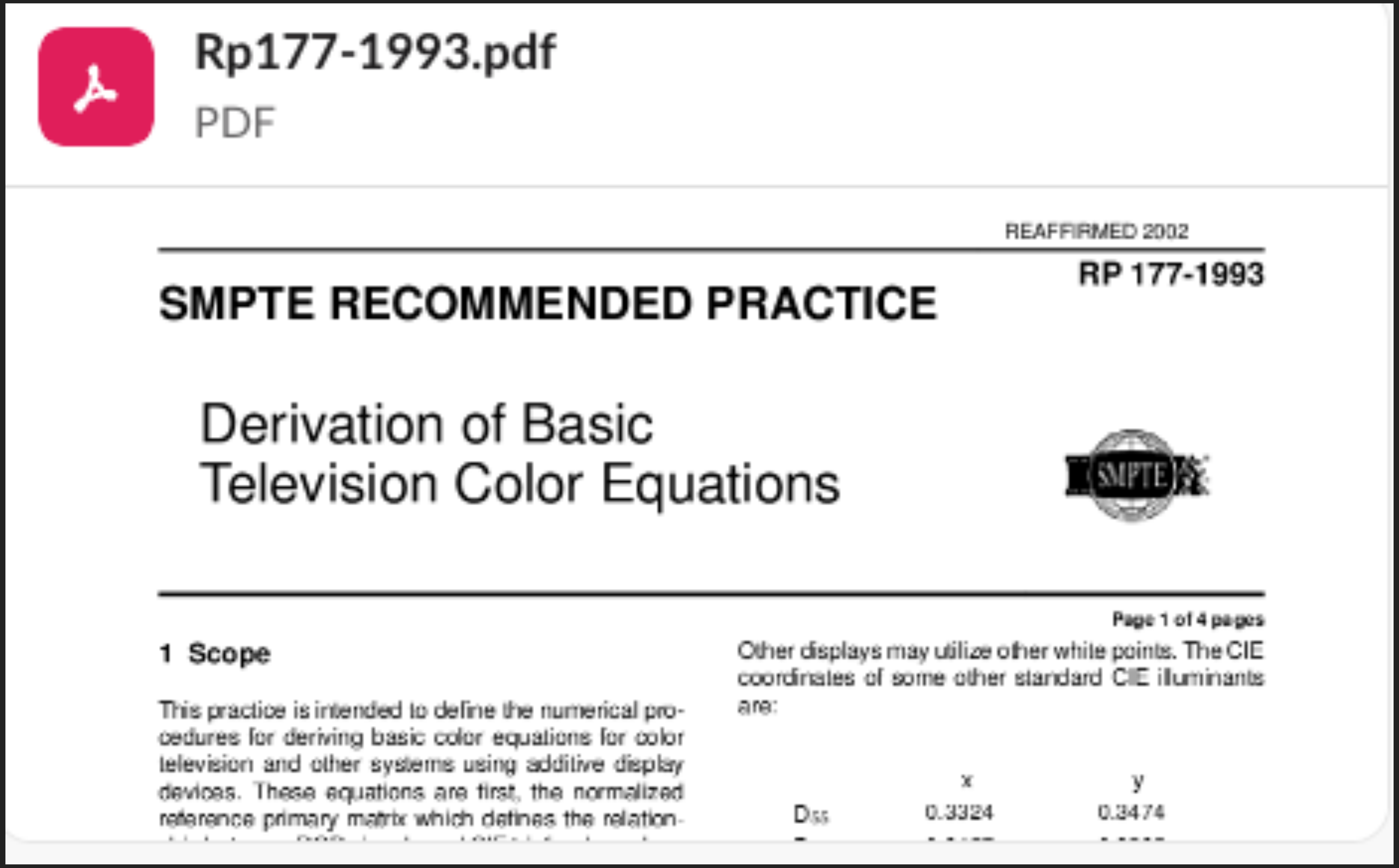
OpenEXR takes a restricted, yet interesting, view of it

COLOR



	CIE (x,y) chromaticities						Gamma	Illuminant
	R		G		B			
	x	y	x	y	x	y		
sRGB	0.64	0.33	0.3	0.6	0.15	0.06	0.45 (2.2^{-1}) [simplified]	D65
Adobe RGB (1998)	0.64	0.33	0.21	0.71	0.15	0.06	0.45 (2.2^{-1})	D65
ProPhoto RGB	0.7347	0.2653	0.1596	0.8404	0.0366	0.0001	0.56 (1.8^{-1})	D50

Chromaticities and white point let us map to the CIEXYZ space



Rp177-1993.pdf

PDF

REAFFIRMED 2002

RP 177-1993

SMPTE RECOMMENDED PRACTICE

Derivation of Basic Television Color Equations

Page 1 of 4 pages

1 Scope

This practice is intended to define the numerical procedures for deriving basic color equations for color television and other systems using additive display devices. These equations are first, the normalized reference primary matrix which defines the relation-

Other displays may utilize other white points. The CIE coordinates of some other standard CIE illuminants are:

	x	y
D_{55}	0.3324	0.3474

via these equations

NanoColor

```
ComputeRGB2XYZMatrixFromPrimaries <- function(redp, greenp, bluep, whitep) {
    # To be consistent, use SMPTE RP 177-1993
    # compute xyz [little xyz]
    red   <- c(redp,   1-sum(redp))
    green <- c(greenp, 1-sum(greenp))
    blue  <- c(bluep,  1-sum(bluep))
    white <- c(whitep, 1-sum(whitep))

    # Build the P matrix
    P <- cbind(red, green, blue)

    # and W
    W <- white / white[2]  # white has luminance factor of 1.0, ie Y = 1

    C <- solve(P) %*% W    #  coefficients to scale primaries

    P %*% diag(as.vector(C))
}
```

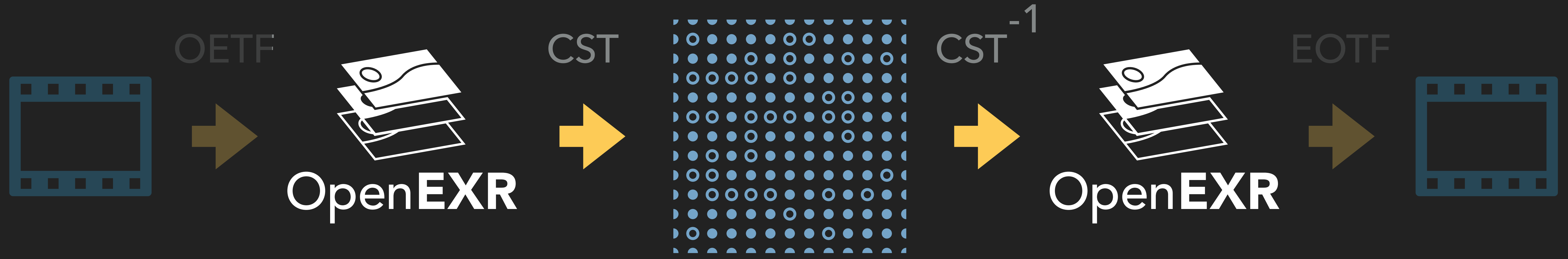
NanoColor

```
# ITU-R BT 2020
#       Primaries are defined as wavelengths:
#       Red = 630nm, Green=532 nm, Blue=467nm
# This is also Fuji F-Log-Gamut

Rec2020 <- ComputeRGB2XYZMatrixFromPrimaries(
                red =   c(.708, .292),
                green = c(.170, .797),
                blue =  c(.131, .046),
                white = c(.3127, .3290))
```

NanoColor

```
# ITU-R BT 2020
#       Primaries are defined as wavelengths:
#       Red = 630nm, Green=532 nm, Blue=467nm
# This is also Fuji F-Log-Gamut

Rec2020 <- ComputeRGB2XYZMatrixFromPrimaries(
                red =   c(.708, .292),
                green = c(.170, .797),
                blue =  c(.131, .046),
                white = c(.3127, .3290))
```

COLOR



however interesting colors don't come from just OpenEXR

COLOR

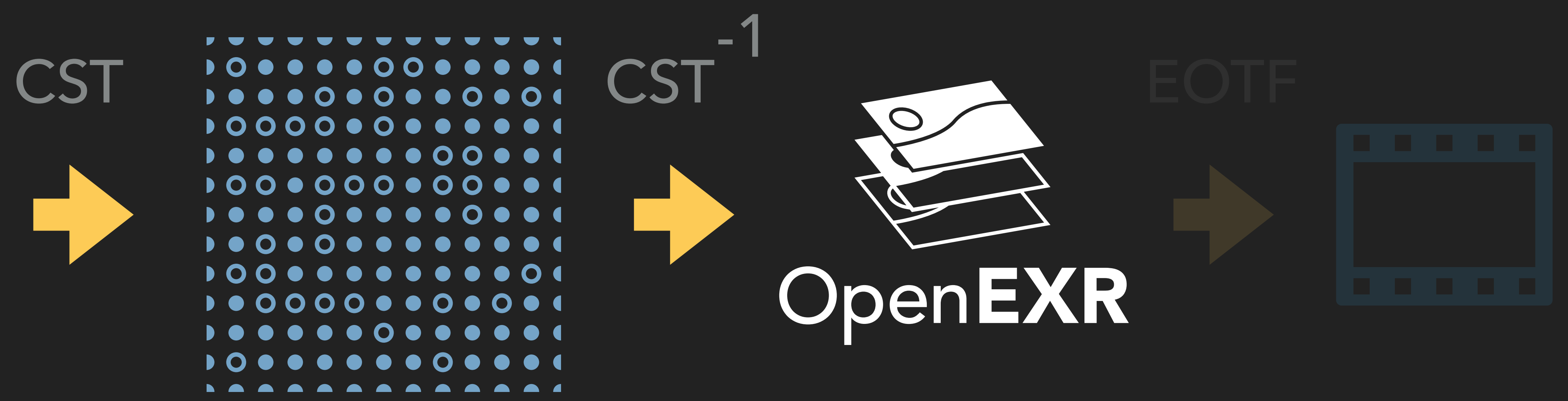
OpenEXR

MATERIALX

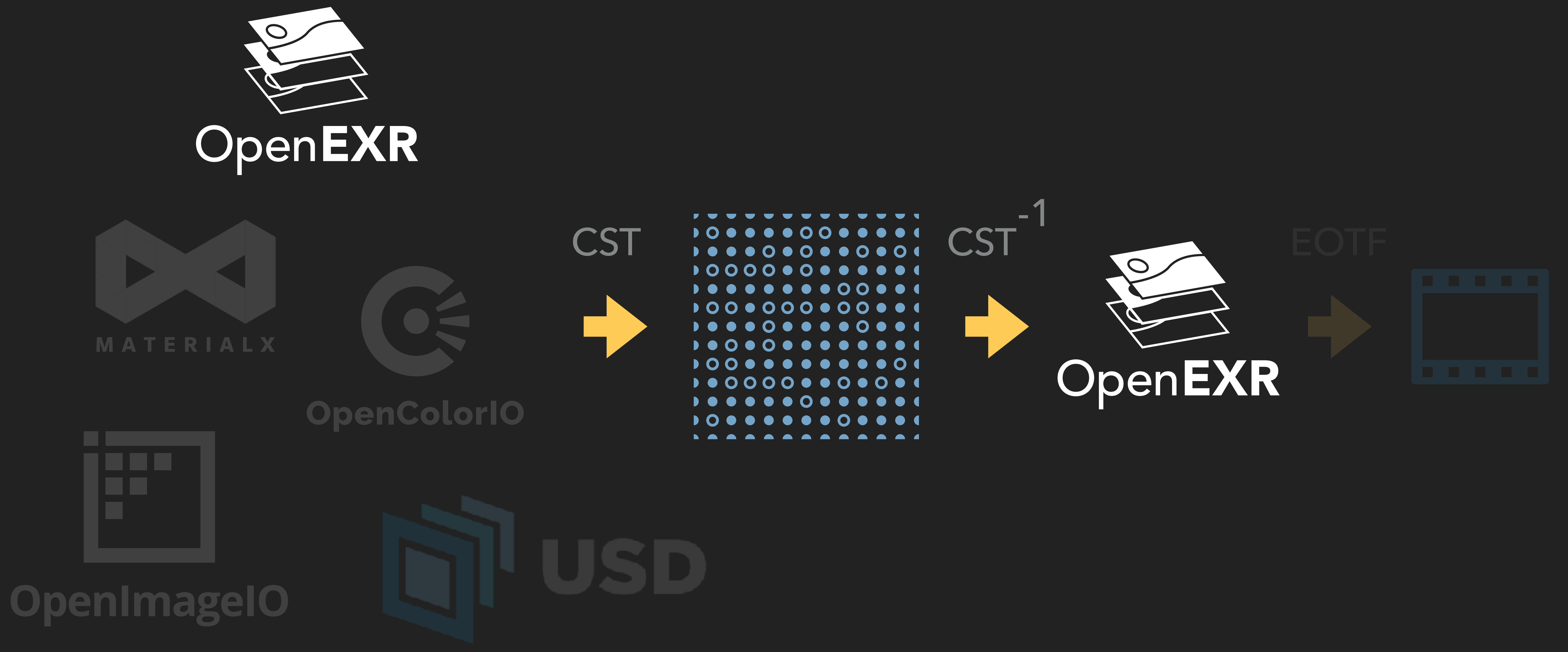
OpenColorIO

OpenImageIO

USD

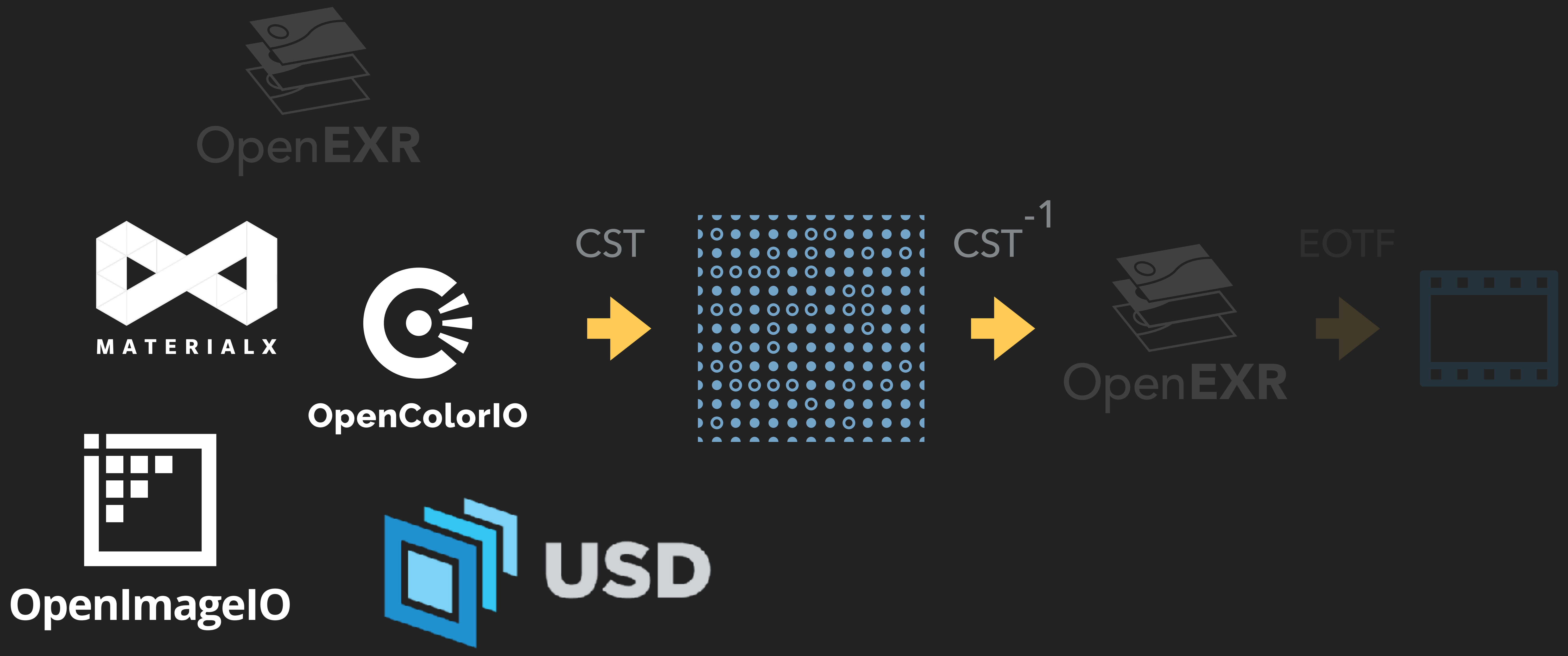


COLOR



Only OpenEXR specifies color in terms of chromaticities and whitepoint.

COLOR



These formats name a color space with a string that corresponds to a recipe.

COLOR

OpenEXR

MATERIALX

OpenColorIO

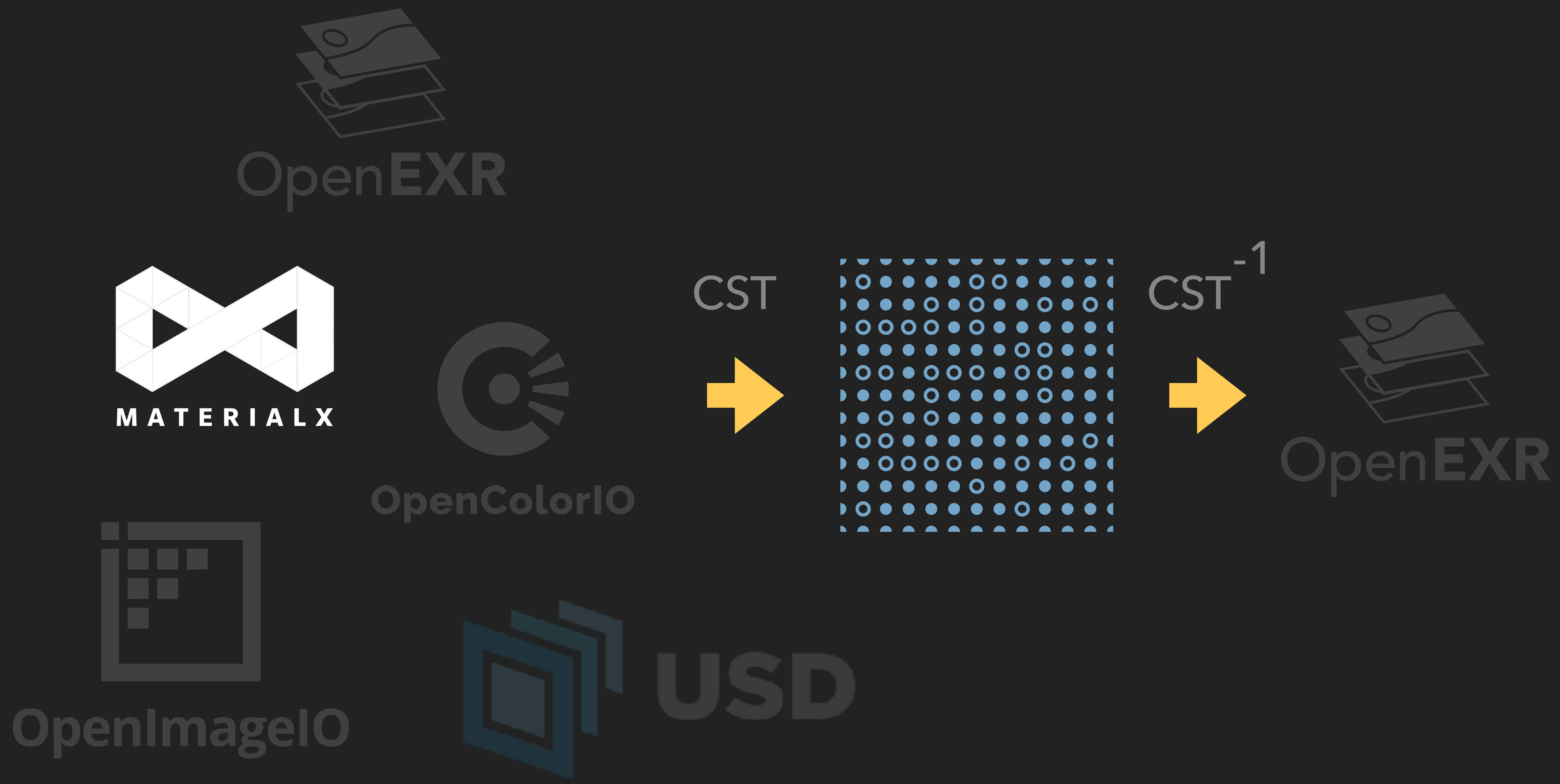
CST
CST^{-1}
OpenEXR

OpenImageIO

USD

Names and recipes vary from site to site.

COLOR



This format has a restricted set of names described normatively

MaterialX's interesting, named, normative, color spaces

```
* `srgb_texture`
* `lin_rec709`
* `g22_rec709`
* `g18_rec709`
* `acescg`
* `lin_ap1 (alias for "acescg")`
* `g22_ap1`
* `g18_ap1`
* `lin_srgb`
* `adobergb`
* `lin_adobergb`
* `srgb_displayp3`
* `lin_displayp3`
```

MaterialX's interesting, named, normative, color spaces

```
* `srgb_texture`
* `lin_rec709`
* `g22_rec709`
* `g18_rec709`
* `acescg`
* `lin_ap1 (alias for "acescg")`
* `g22_ap1`
* `g18_ap1`
* `lin_srgb`
* `adobergb`
* `lin_adobergb`
* `srgb_displayp3`
* `lin_displayp3`
```

```
* `lin_rec2020`
```

This one is also interesting!

Specification through specification data

```
* `srgb_texture`
* `lin_rec709`
* `g22_rec709`
* `g18_rec709`
* `acescg`
* `lin_ap1 (alias for "acescg")`
* `g22_ap1`
* `g18_ap1`
* `lin_srgb`
* `adobergb`
* `lin_adobergb`
* `srgb_displayp3`
* `lin_displayp3`
* `lin_rec2020`
```

These can be specified via

- input transform removal operator
- chromaticities
- whitepoint

Then, the RP 177-1993 equations take us in and out of CIEXYZ, and can be followed with an

- output transform operator

SMPTE

RP177-1993

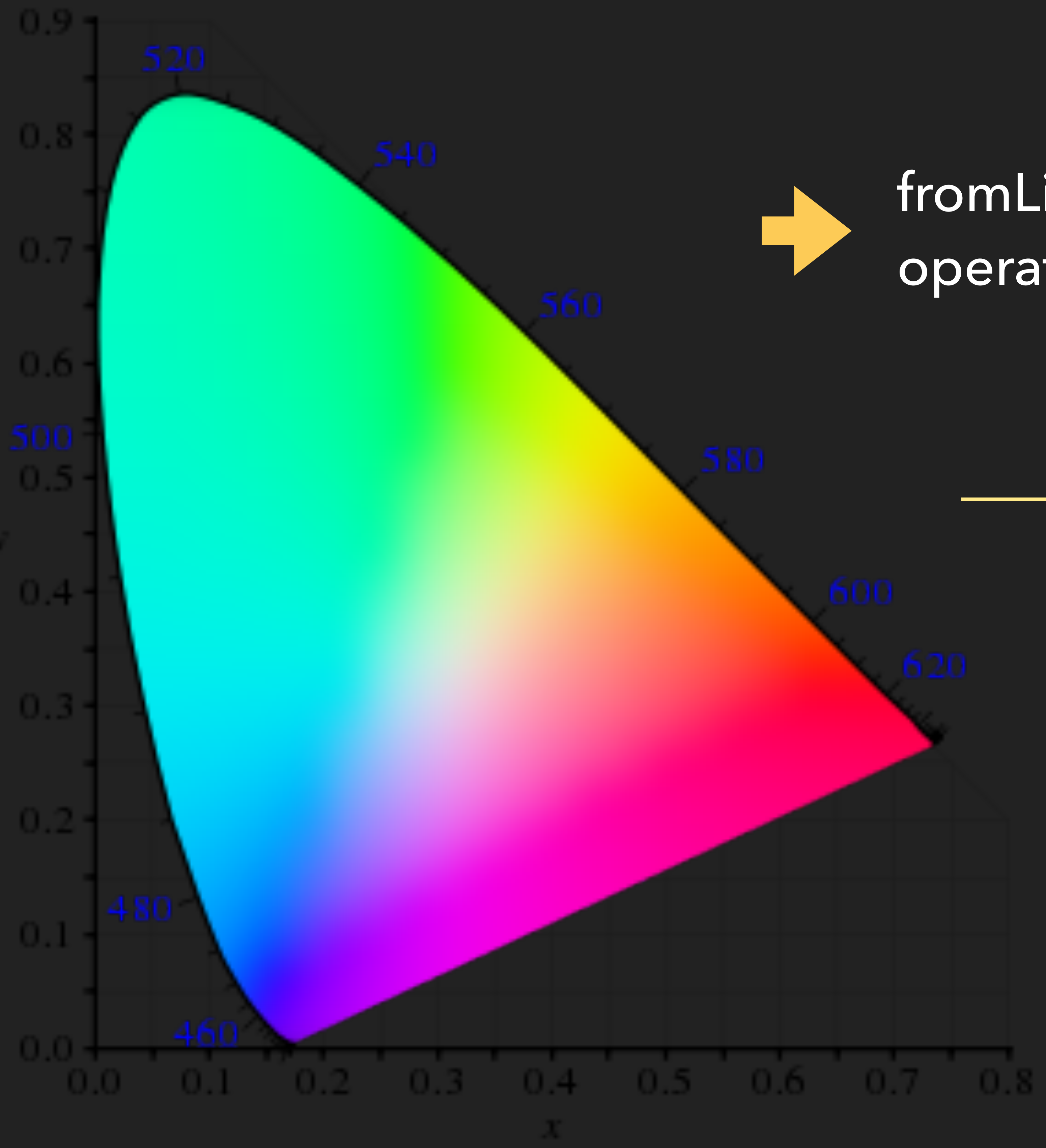
We can get to any working color space via CIEXYZ

* `srgb_texture`
* `g22_rec709`
* `g18_rec709`
* `g22_ap1`
* `g18_ap1`
* `srgb_displayp3`

→ toLinear operator →

* `lin_rec709`
* `acescg`
* `lin_ap1`
* `lin_srgb`
* `adobergb`
* `lin_adobergb`
* `lin_displayp3`
* `lin_rec2020`

→



→ fromLinear operator →

* `srgb_texture`
* `g22_rec709`
* `g18_rec709`
* `g22_ap1`
* `g18_ap1`
* `srgb_displayp3`

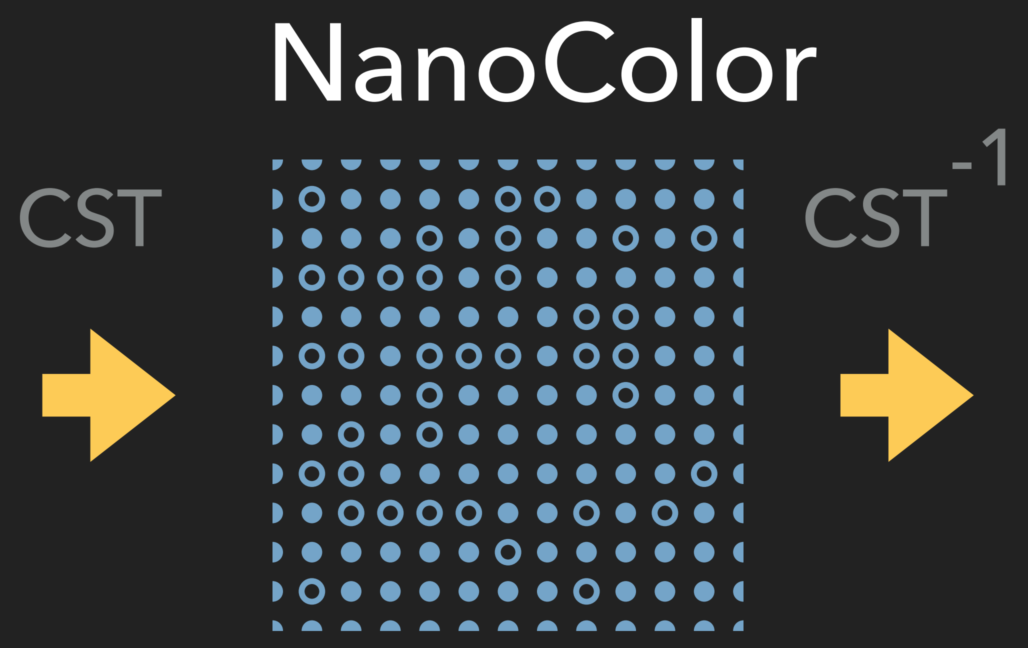
→

* `lin_rec709`
* `acescg`
* `lin_ap1`
* `lin_srgb`
* `adobergb`
* `lin_adobergb`
* `lin_displayp3`
* `lin_rec2020`

USD & MaterialX & OCIO

- can share some named "working" color spaces
- can provide first principles math functions
- can declare the spaces in terms of a parameterized color space node
- based on a small set of operators, themselves MaterialX nodes

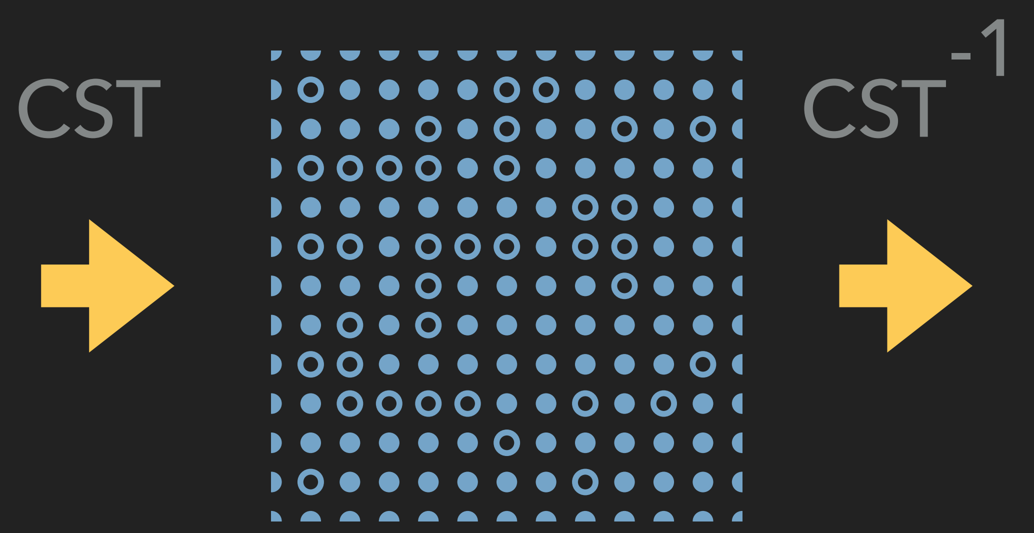
Contribution to OCIO



- A constrained problem domain
- A renderer's input working space to output working space
- Can be encapsulated in a small first principles library
- For use by all renderers whether offline or realtime

OpenColorIO

NanoColor

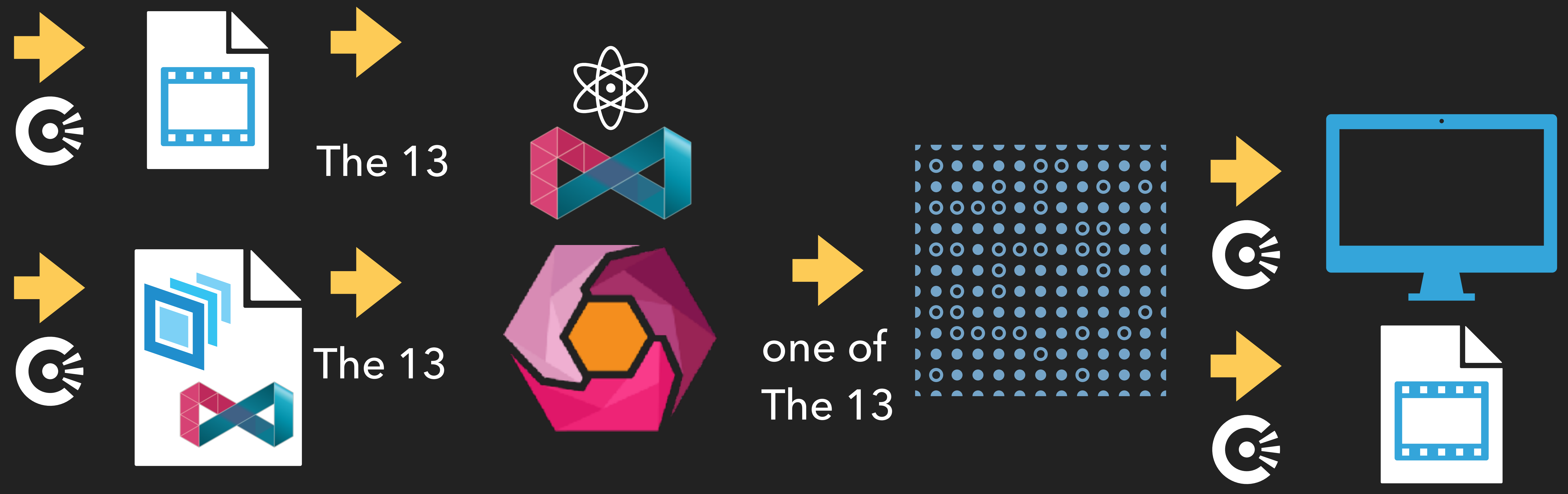


```
typedef struct {
    float x;
    float y;
} ncCIEXY;

typedef enum {
    linear, sRGB, Rec709, Rec2020
} ncLinearOp;

typedef struct {
    const char* name;
    CIEXY chromaticities;
    CIEXY white;
    linearOp op;
} ncColorSpace;

typedef struct {
    float m[9];
} ncMatrix3x3;
```

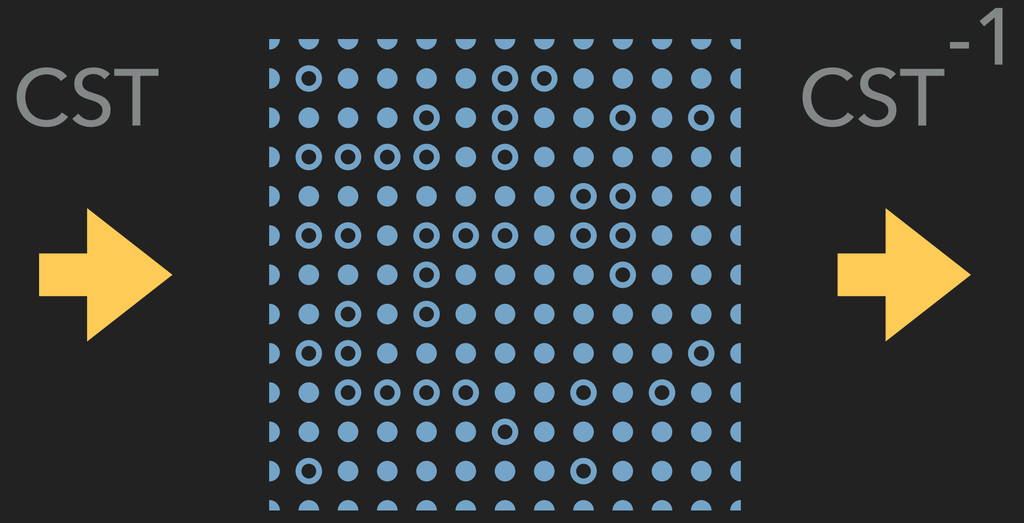
```
ncMatrix3x3 ncGetRGBtoCIEXYZMatrix(const ncColorSpace* cs) {
    ncMatrix3x3 m;
    // compute according to RP177-1993
    // ...
    return m;
}

ncMatrix3x3 ncGetCIEXYZtoRGBMatrix(const ncColorSpace* cs) {
    return ncInv3x3(ncGetRGBtoCIEXYZMatrix(cs));
}

typedef struct {
    ncLinearOp  toLinear;
    ncLinearOp  fromLinear;
    ncMatrix3x3 transform;
} ncColorTransform;

ncColorTransform ncGetRGBtoRGBMatrix(const ncColorSpace* src,
                                     const ncColorSpace* dst) {
    ncColorTransform t;
    t.transform = ncMul3x3(ncGetRGBtoCIEXYZMatrix(dst),
                           ncGetCIEXYZtoRGBMatrix(src));
    t.toLinear = dst->op;
    t.fromLinear = src->op;
    return t;
}

ncColorSpace ncGetColorSpaceFromName(const char* name) {
    ncColorSpace cs;
    // ...
    return cs;
}
```

THE 13

Remember: The 13 means the "canonical set" plus user defined (eg PhysCam) plus OCIO confs where the user definitions and confs are restricted to spaces that can be represented by a closed form linearization equation, and a 3x3 matrix.



NanoColor



CST

CST^{-1}

As simple as possible, but no simpler