Linear Workflow
in LightWave

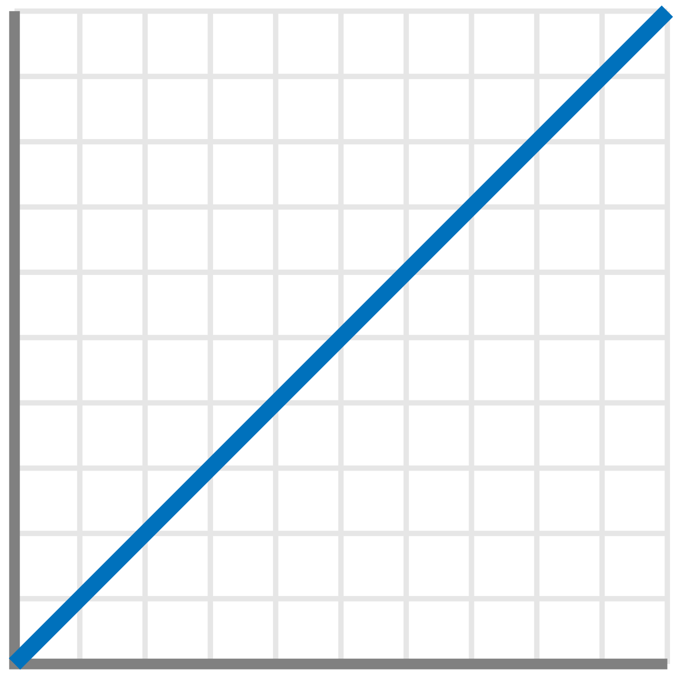
What is it and why you should care

Matt Gorner | www.pixsim.co.uk

In a nut shell ...

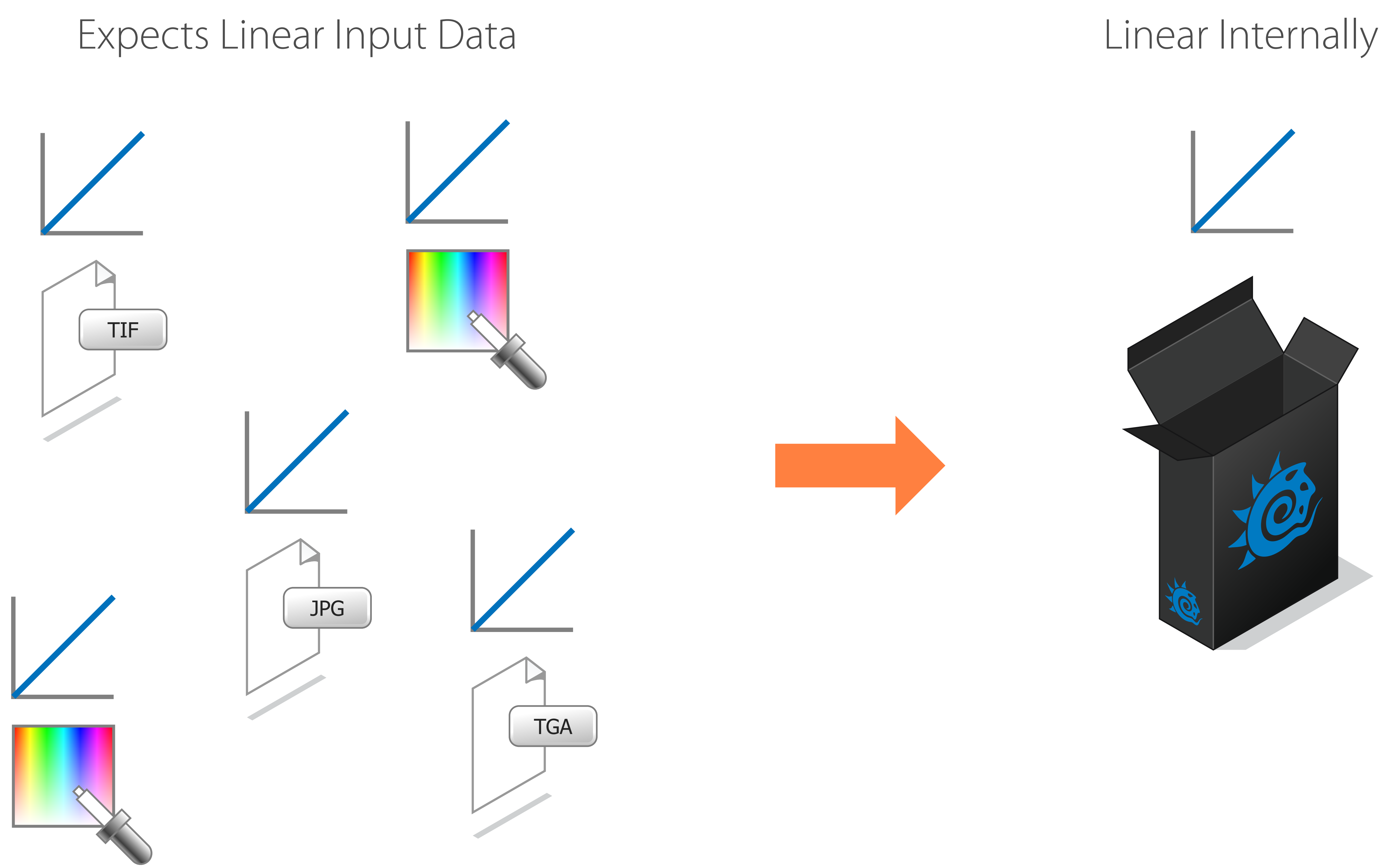
Linear Workflow

LightWave's render engine works in linear colour space internally



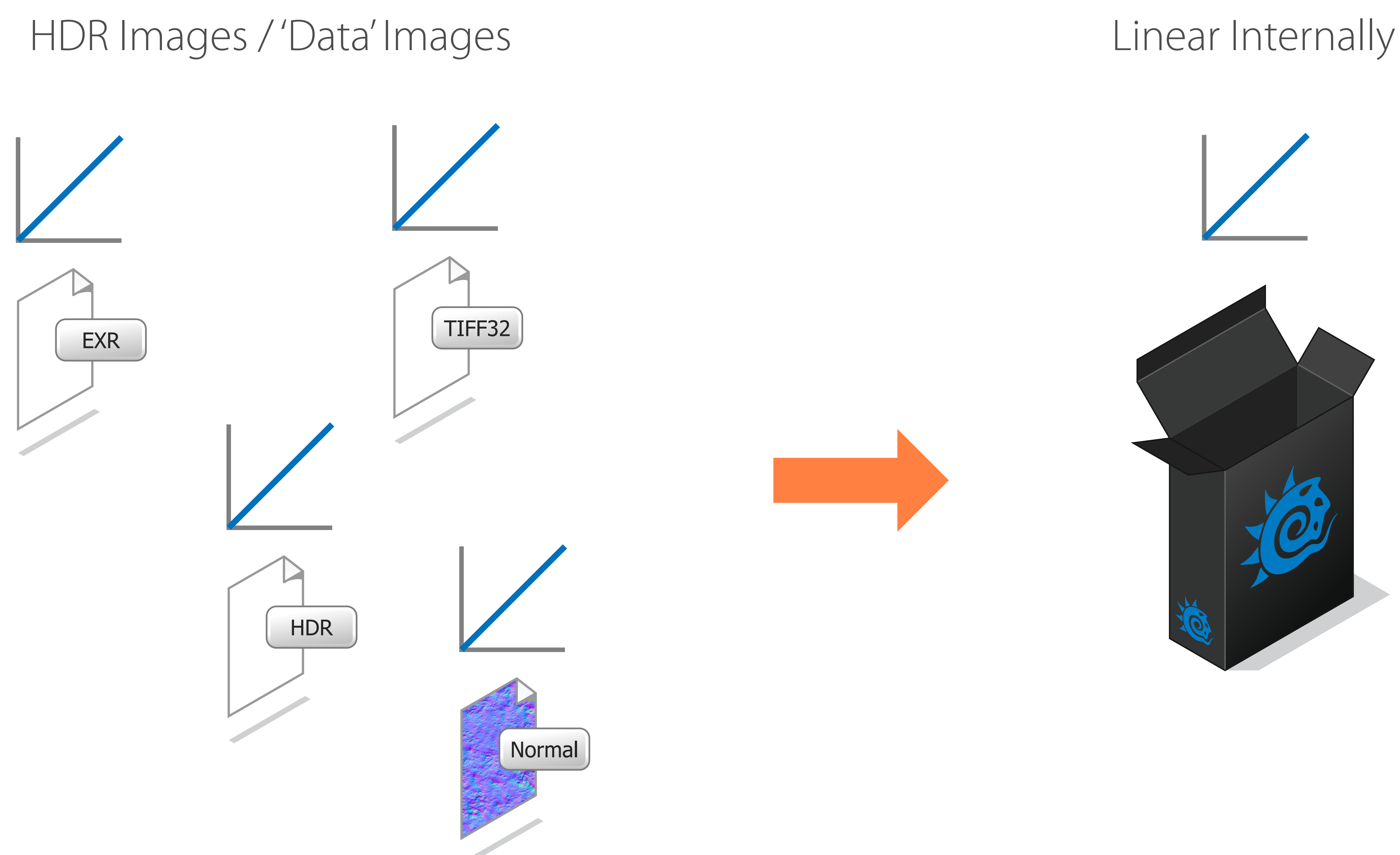
Linear Workflow

For the rendering calculations to be correct, LightWave expects colours and textures to be linear too



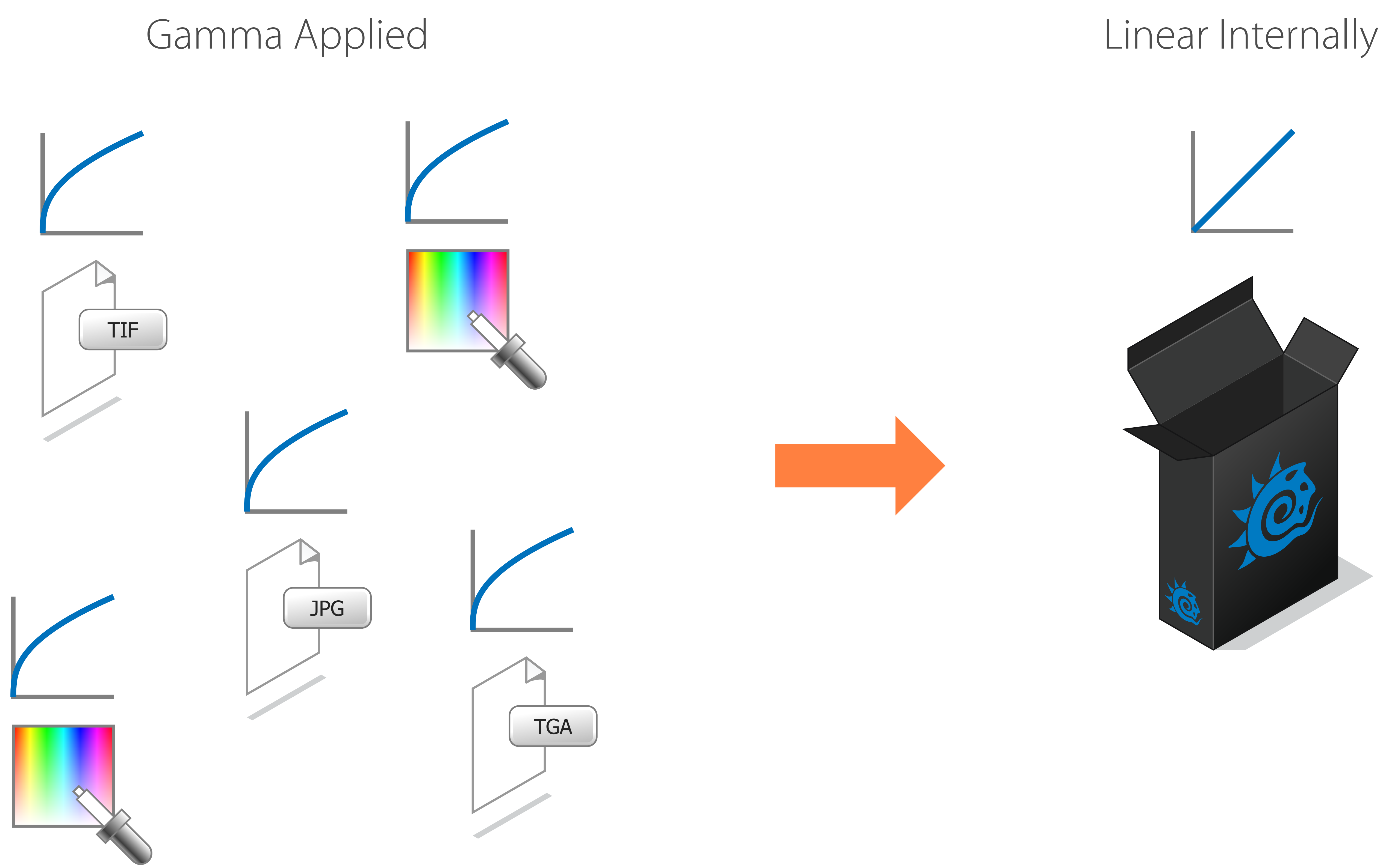
Linear Workflow

Except High Dynamic Range and 'data' images such as normal maps



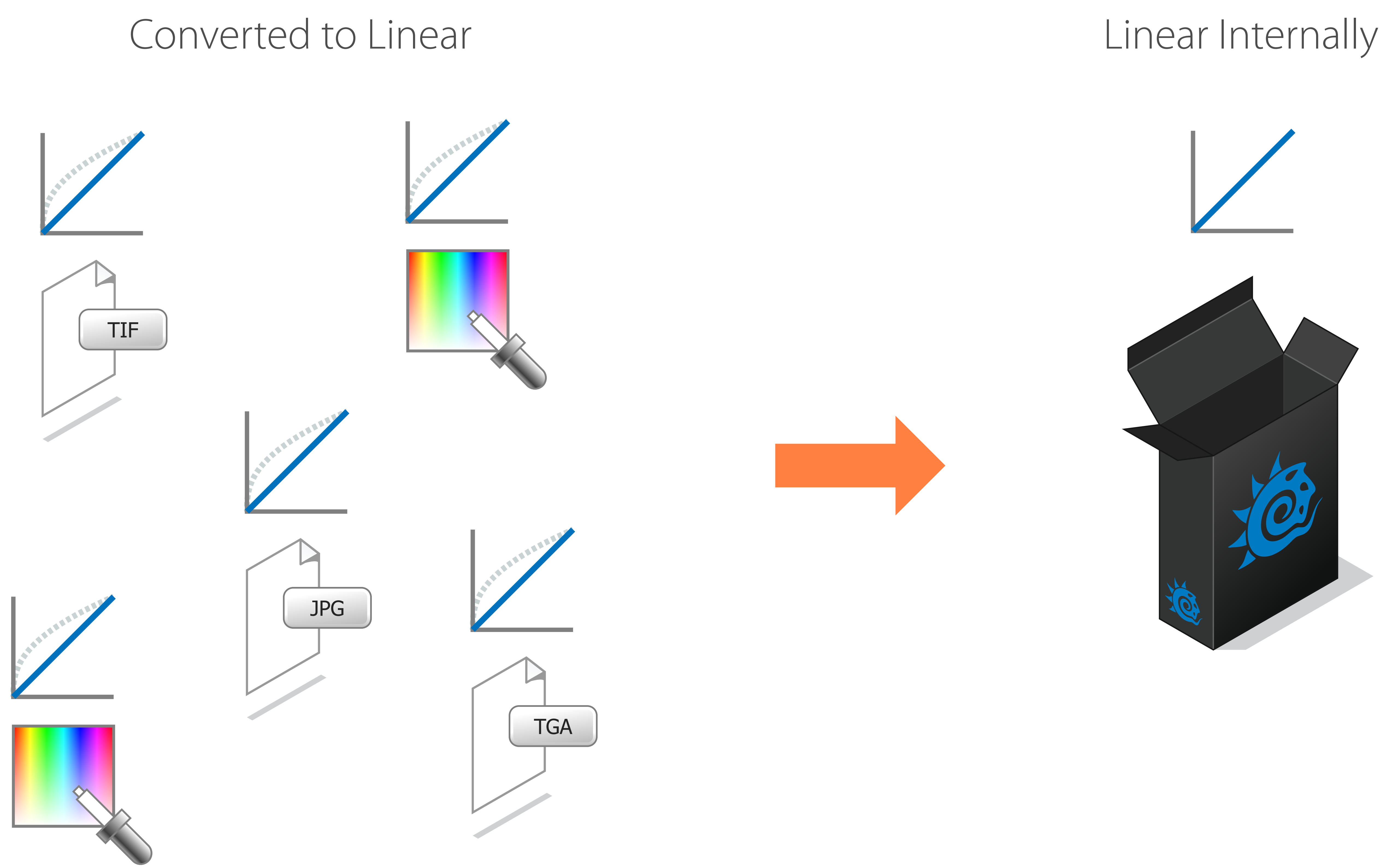
Linear Workflow

But often colours and texture maps are 'Gamma' encoded



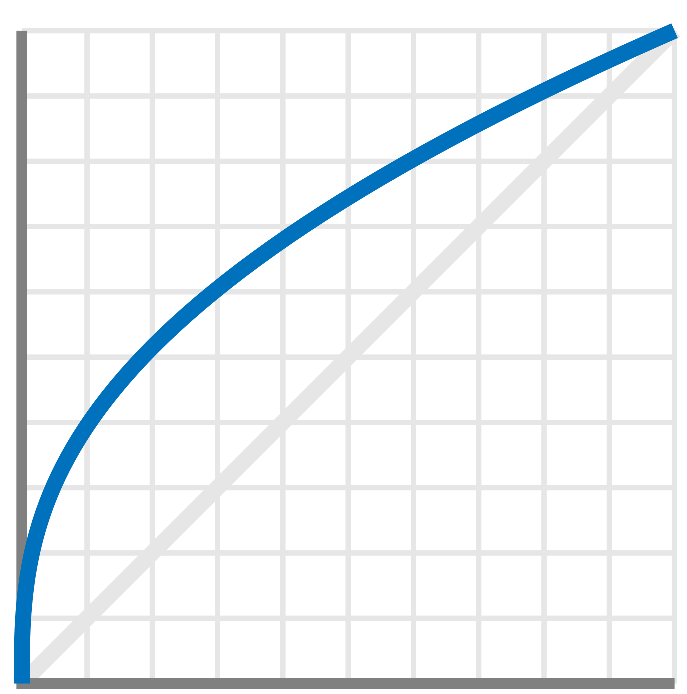
Linear Workflow

So textures and colours need to be converted to linear colour space

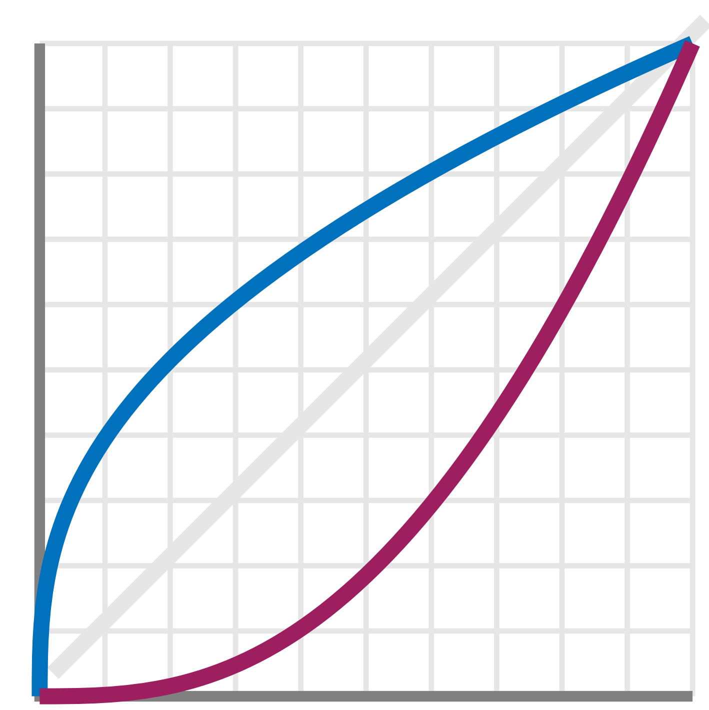


Linear Workflow

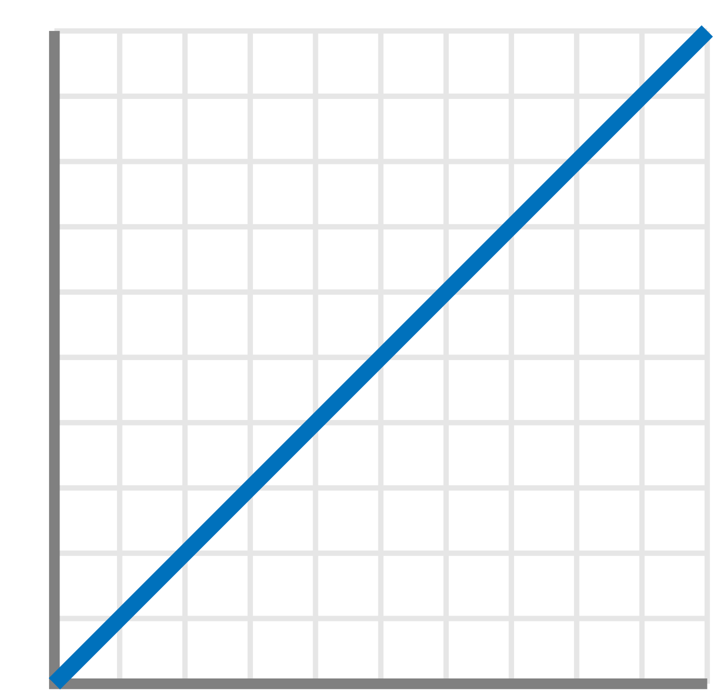
This is done by applying the inverse of the colour space applied to the texture or colour



sRGB
(2.2 Gamma)



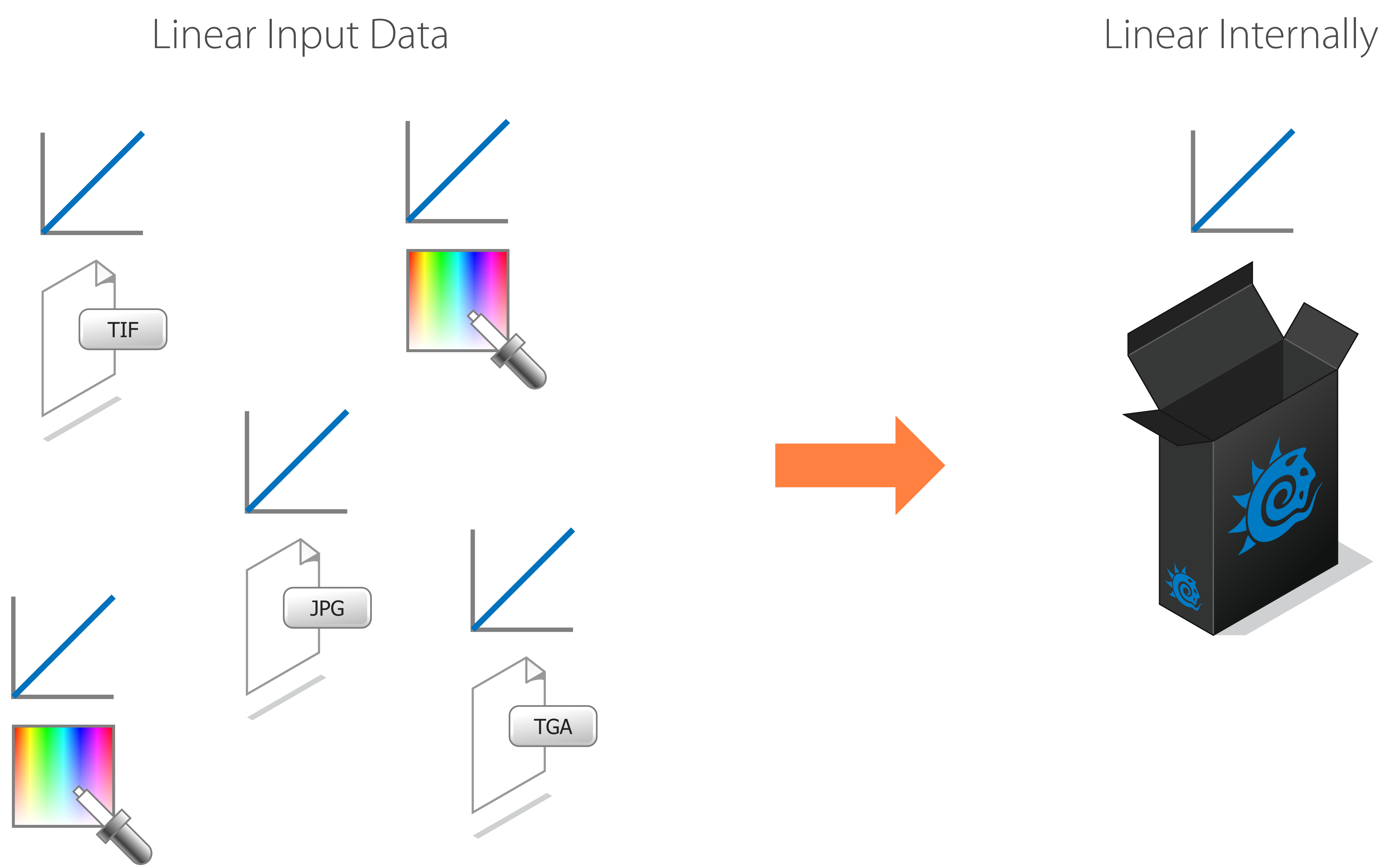
Apply Inverse
(1.0 / 2.2)



Linear
(1.0 Gamma)

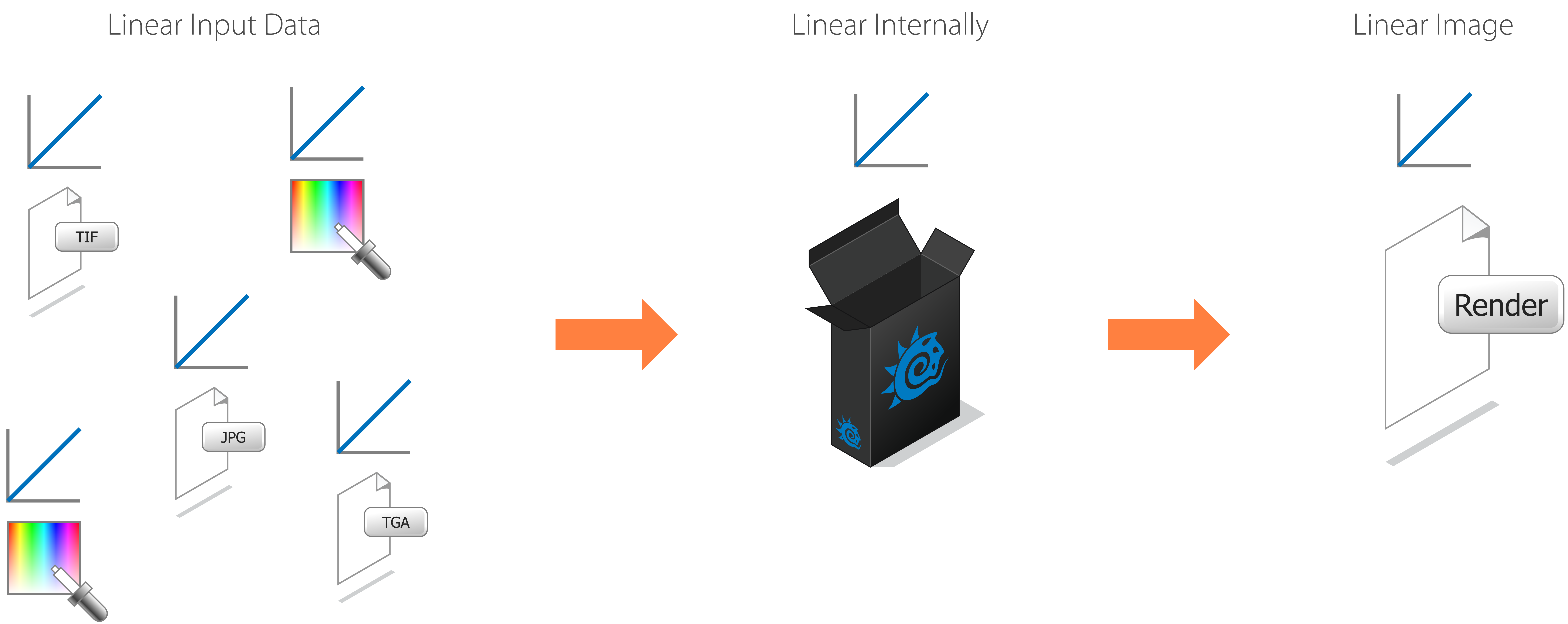
Linear Workflow

So now, all the data fed into the rendering algorithm is in the same space



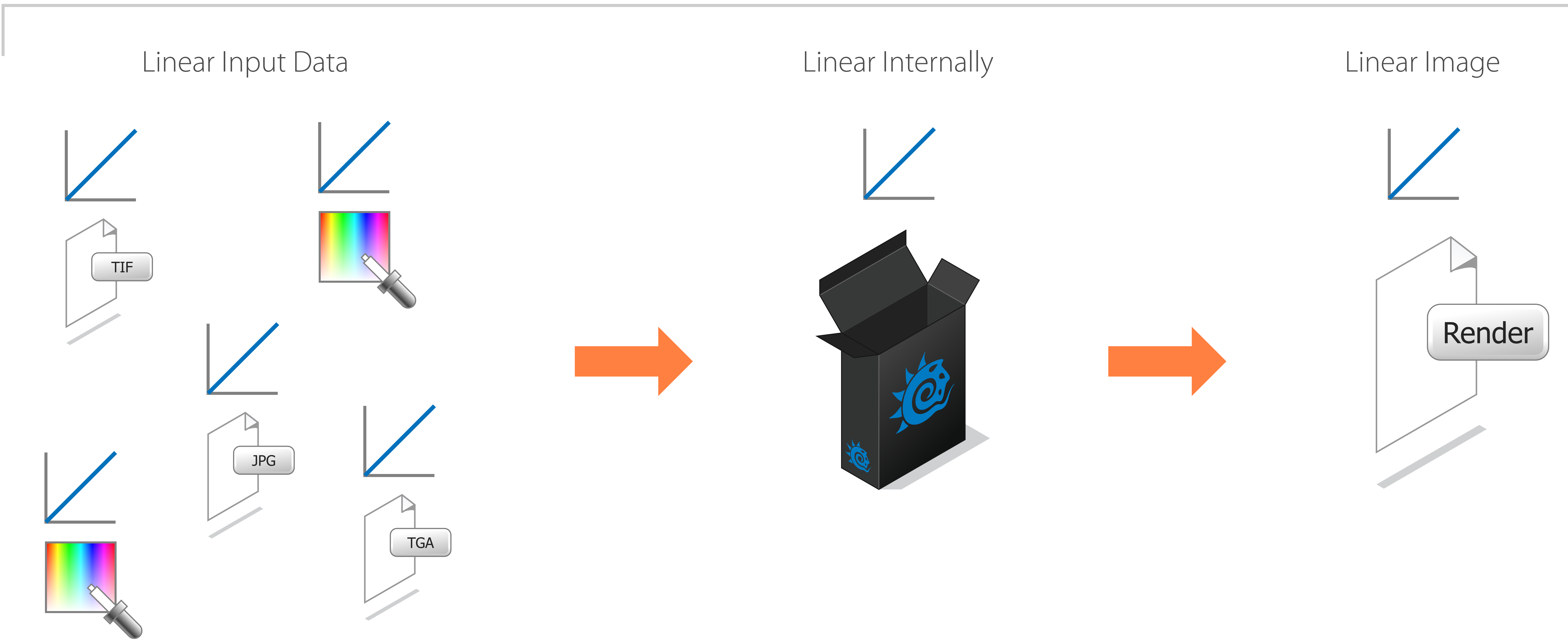
Linear Workflow

Once rendering is complete, the final image will also be in linear colour space



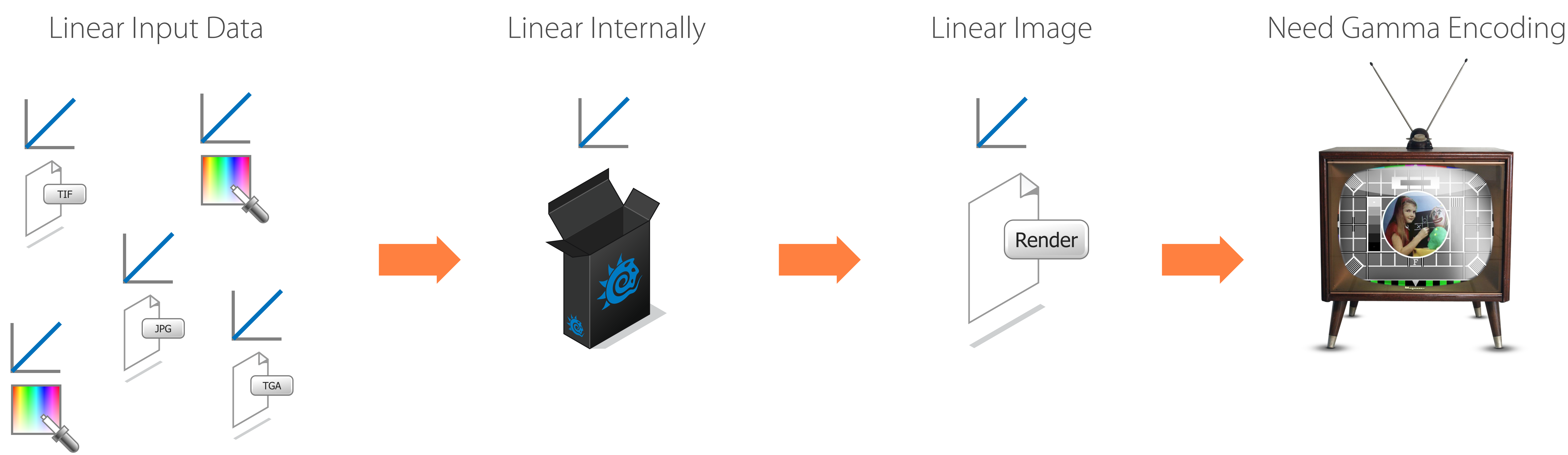
Linear Workflow

Linear Workflow - The process of keeping all data linear through the rendering process



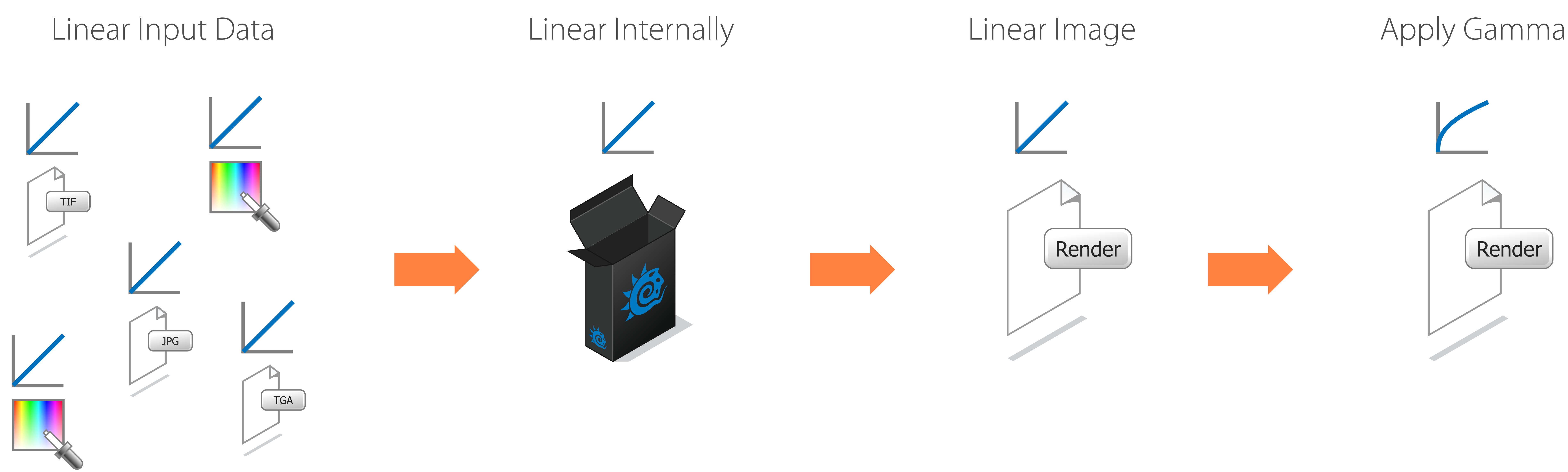
Linear Workflow

However, linear images cannot be displayed properly on conventional monitors



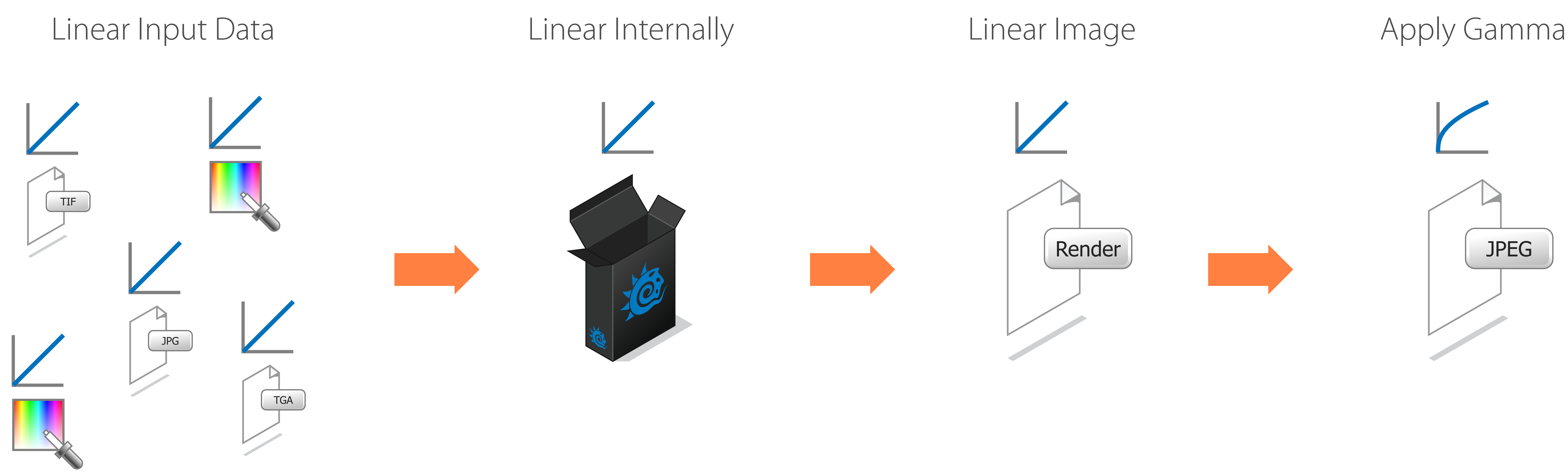
Linear Workflow

They need to be Gamma encoded to look correct, this can be done in LightWave



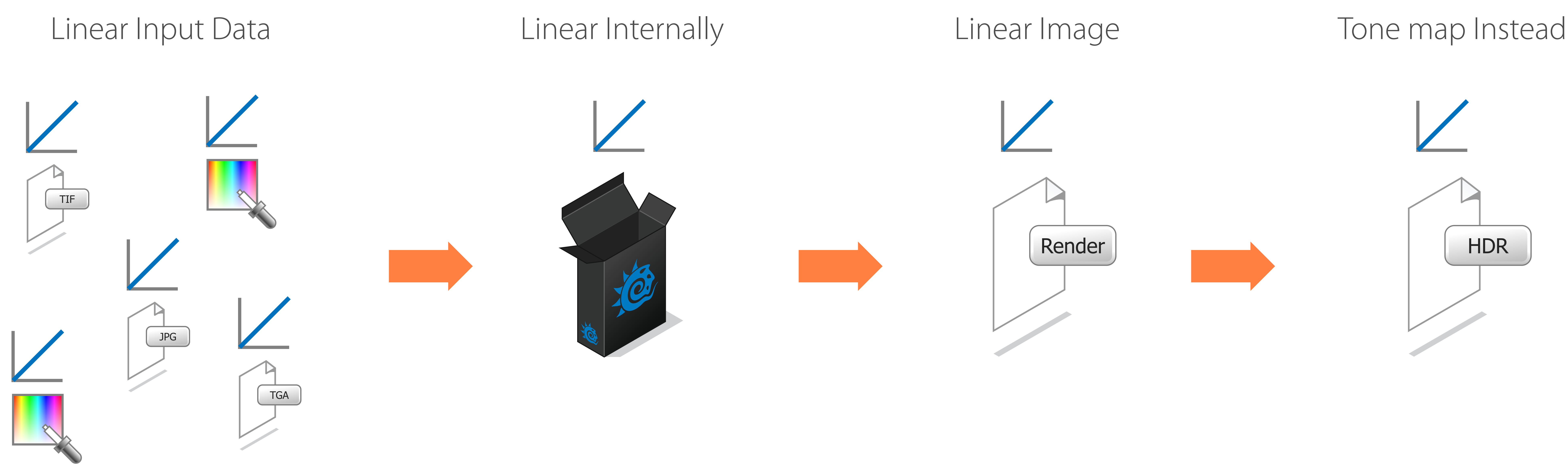
Linear Workflow

You also need to Gamma encode if saving to an 8-bit image format (e.g. JPEG)



Linear Workflow

Or the render can be left linear and saved as an HDR image to be tone mapped in another application

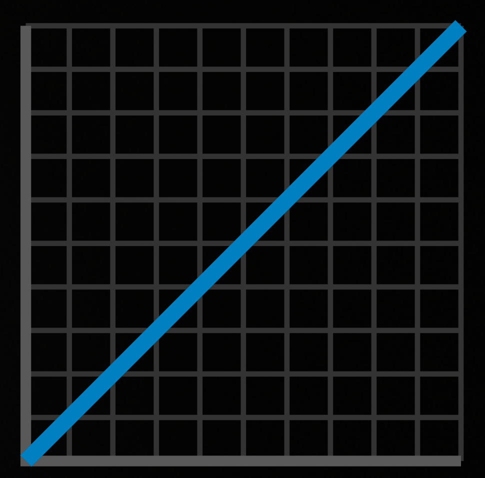


What Does This Do
For Me Exactly?


Images are displayed correctly
Linear (1.0 Gamma)

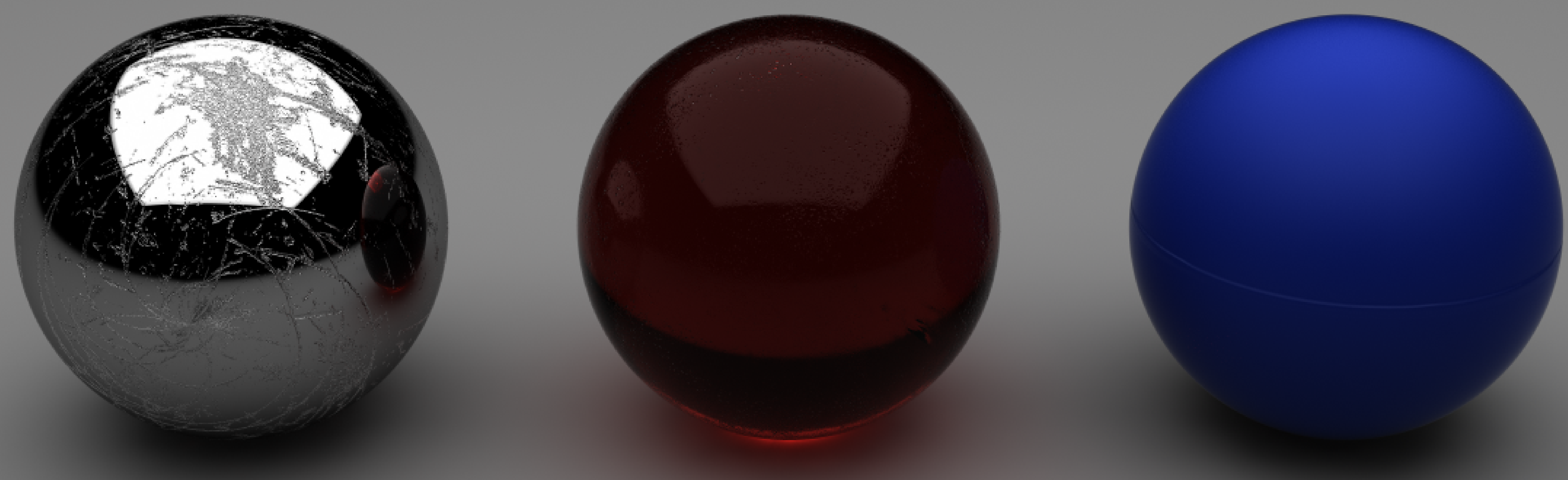

Images are displayed correctly
sRGB (2.2 Gamma)

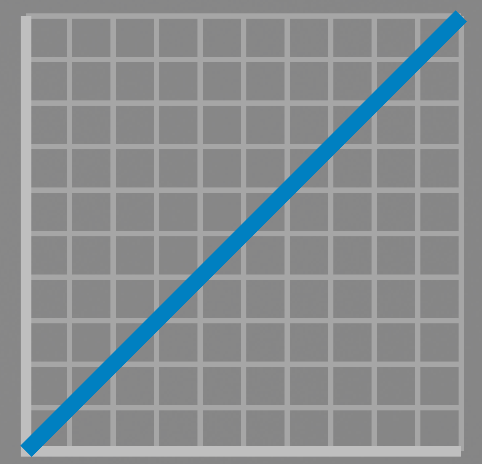
Lights with Inv^2 falloff behave correctly



Lights with Inv^2 falloff behave correctly

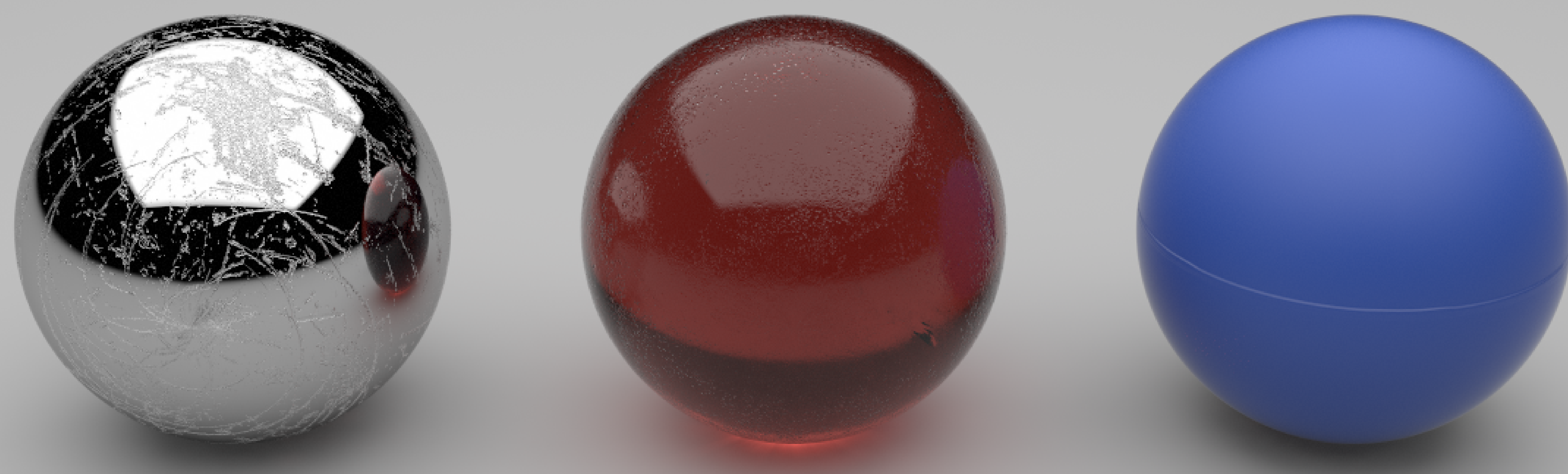
Materials respond correctly

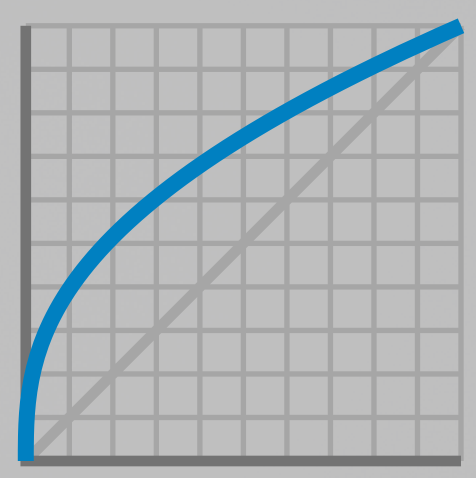




Linear (1.0 Gamma)

Materials respond correctly



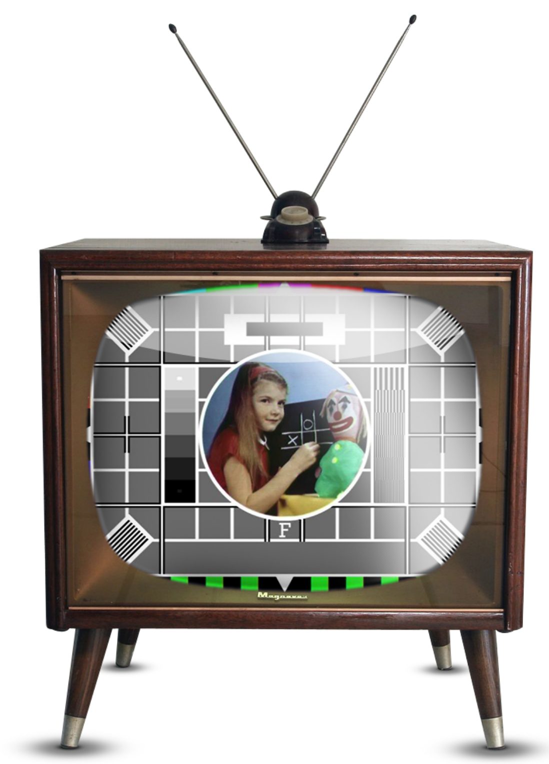


sRGB (2.2 Gamma)

Why do I have to
do all this anyway?

What you need to know

There are four main topics that help in understanding why Linear Workflow is important



How Devices Display Images

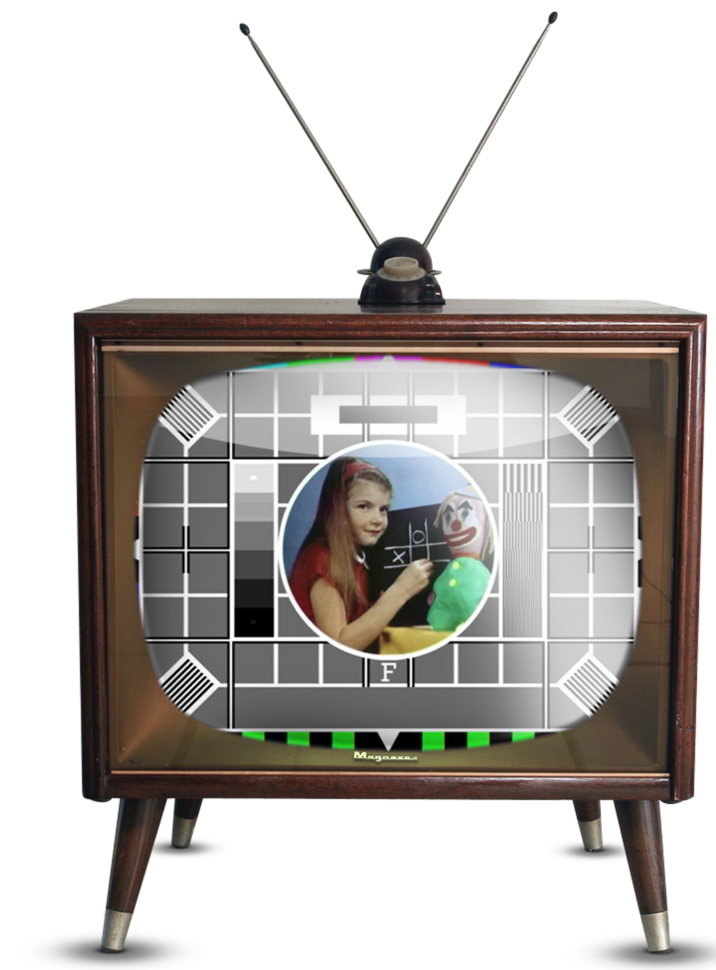
γ

What Gamma is

How Devices Capture Images
(Cameras, Scanner, Etc)

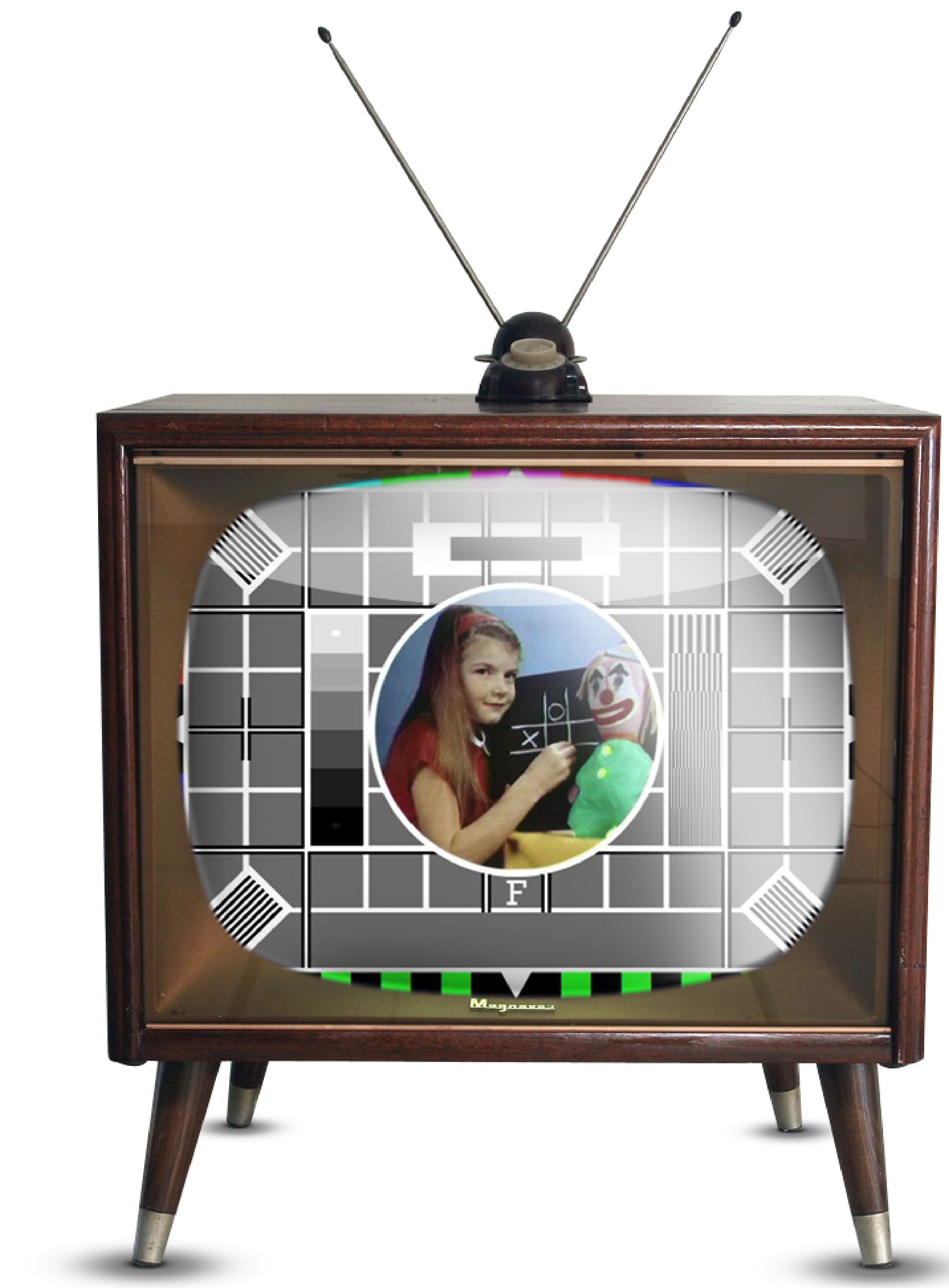
How Our Eyes See

How Devices Display Images



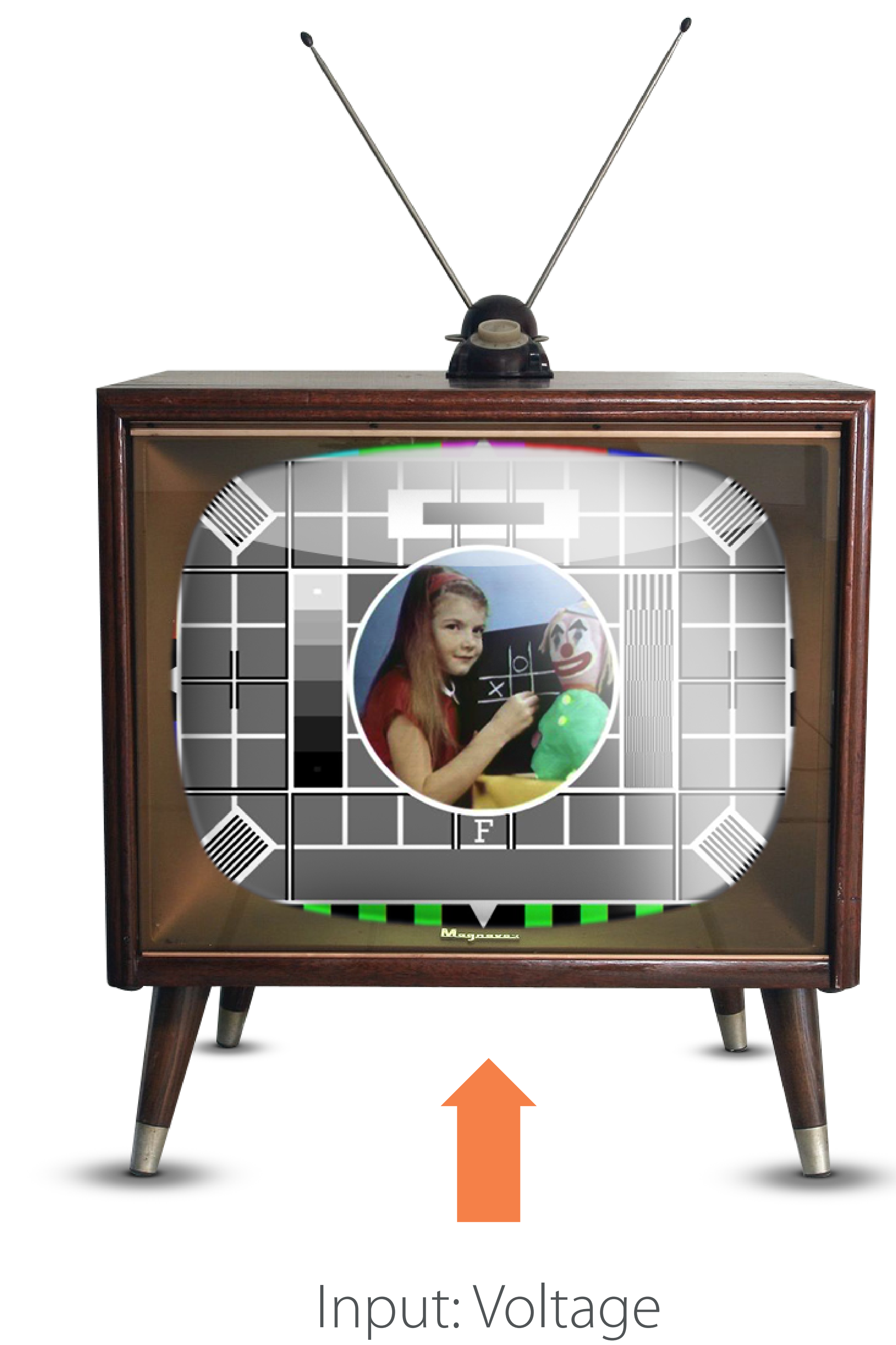
How Devices Display Images

We need to go back to how CRT displays work ...



How Devices Display Images

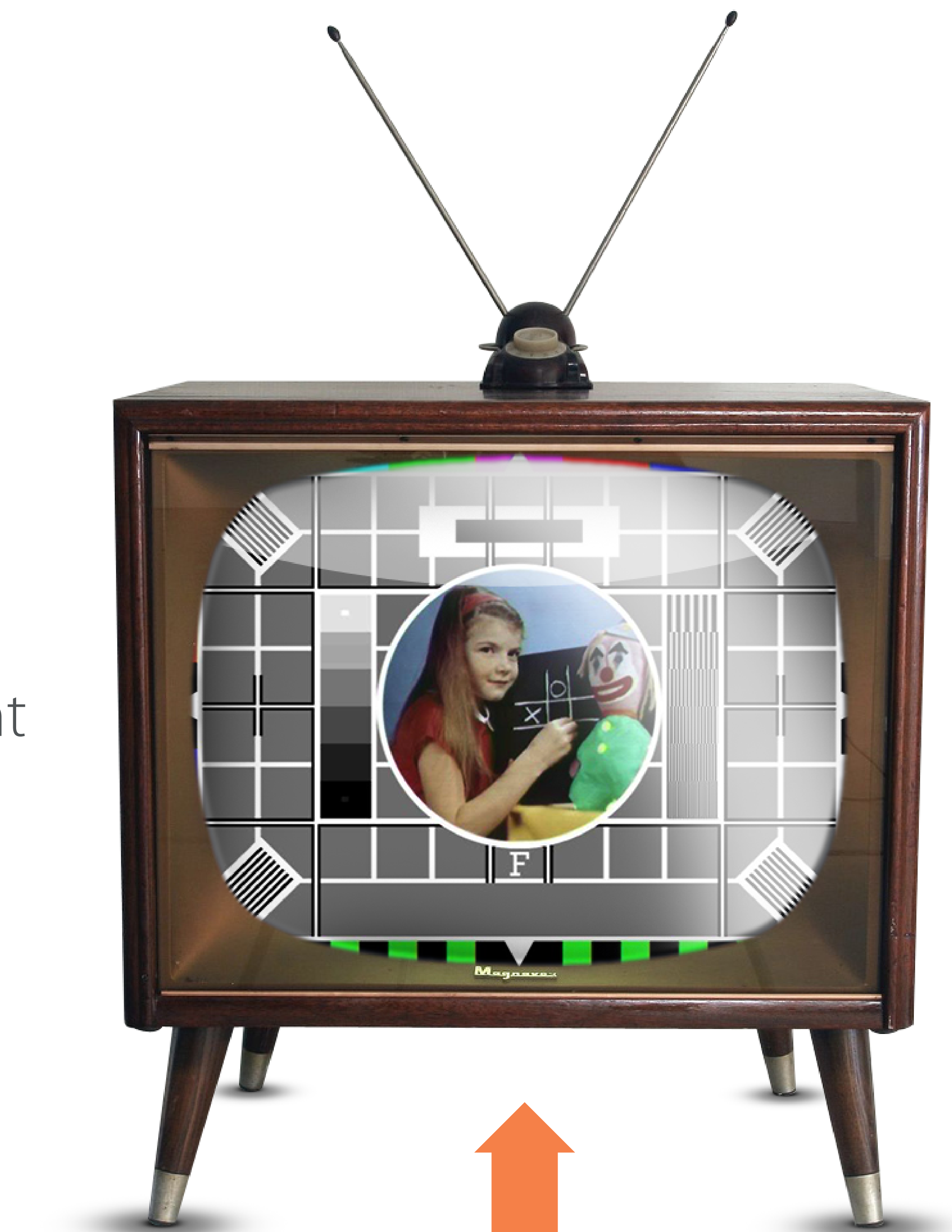
CRT displays work by taking an input voltage ...



How Devices Display Images

Which the electron gun inside the CRT uses to fire electron beams at a phosphor coated screen.
The phosphor reacts by emitting light when it is hit by a beam (output light)

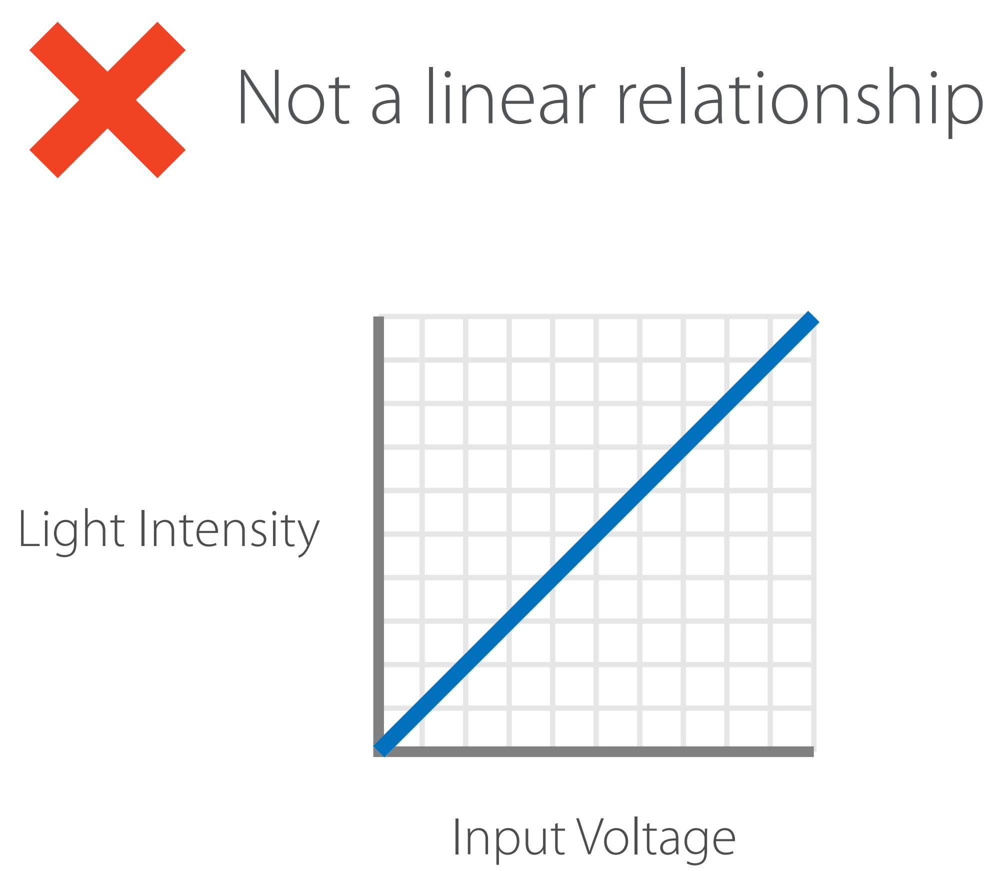
Output: Light

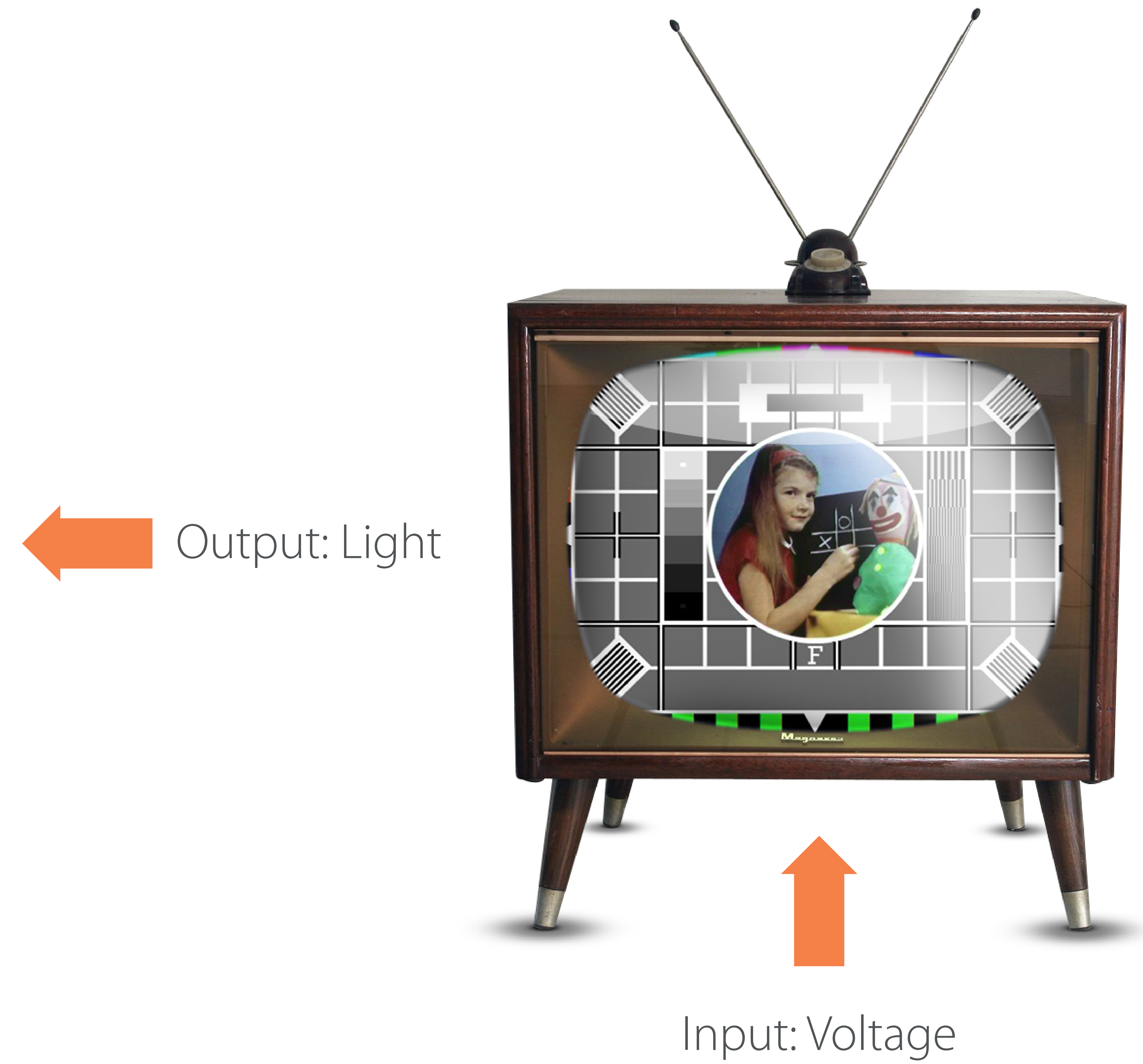


Input: Voltage

How Devices Display Images

However, due to electrostatic effects inside the CRT electron gun, the output response to the input voltage is not linear



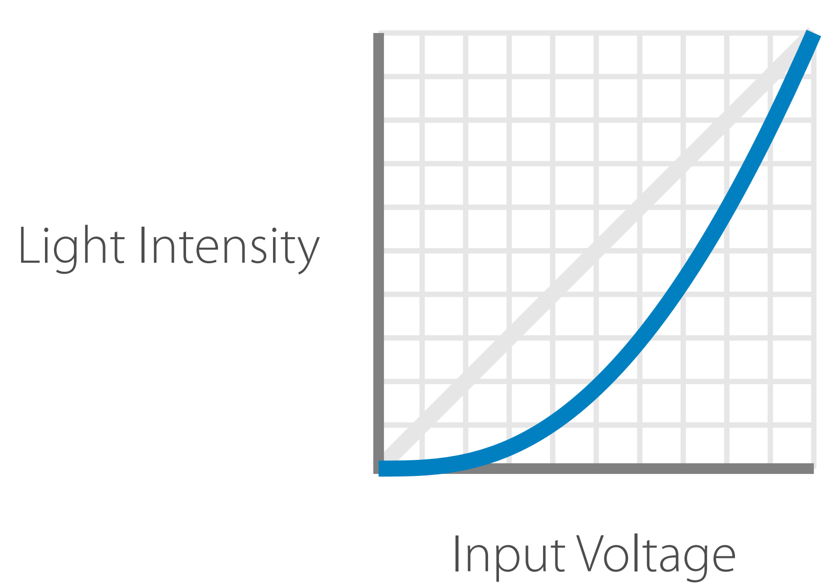


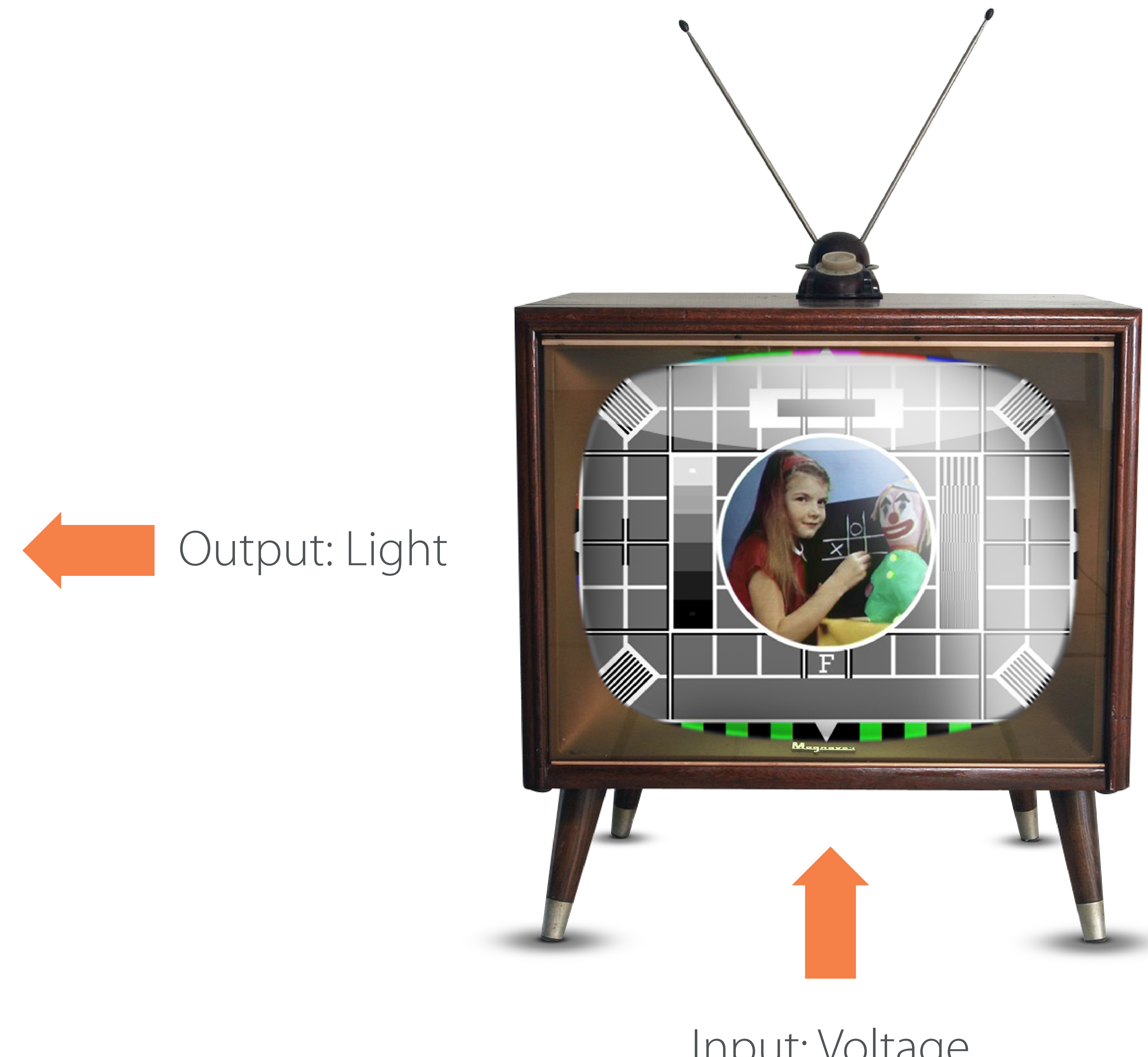
How Devices Display Images

On a CRT display, light intensity is approximate to the voltage raised to the power close to 2.5, which can be expressed using:

$$L = (V' + \varepsilon)^{\gamma}$$

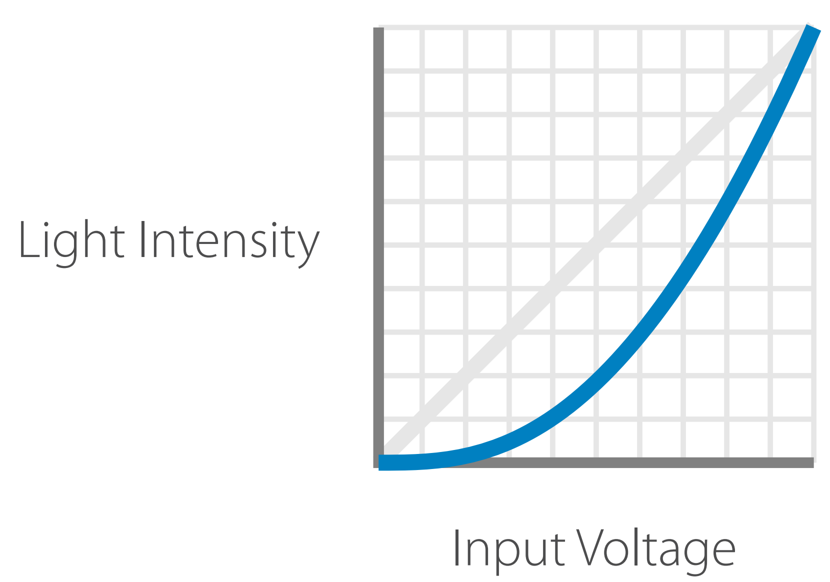
The numerical value of this power law is known as "Gamma"

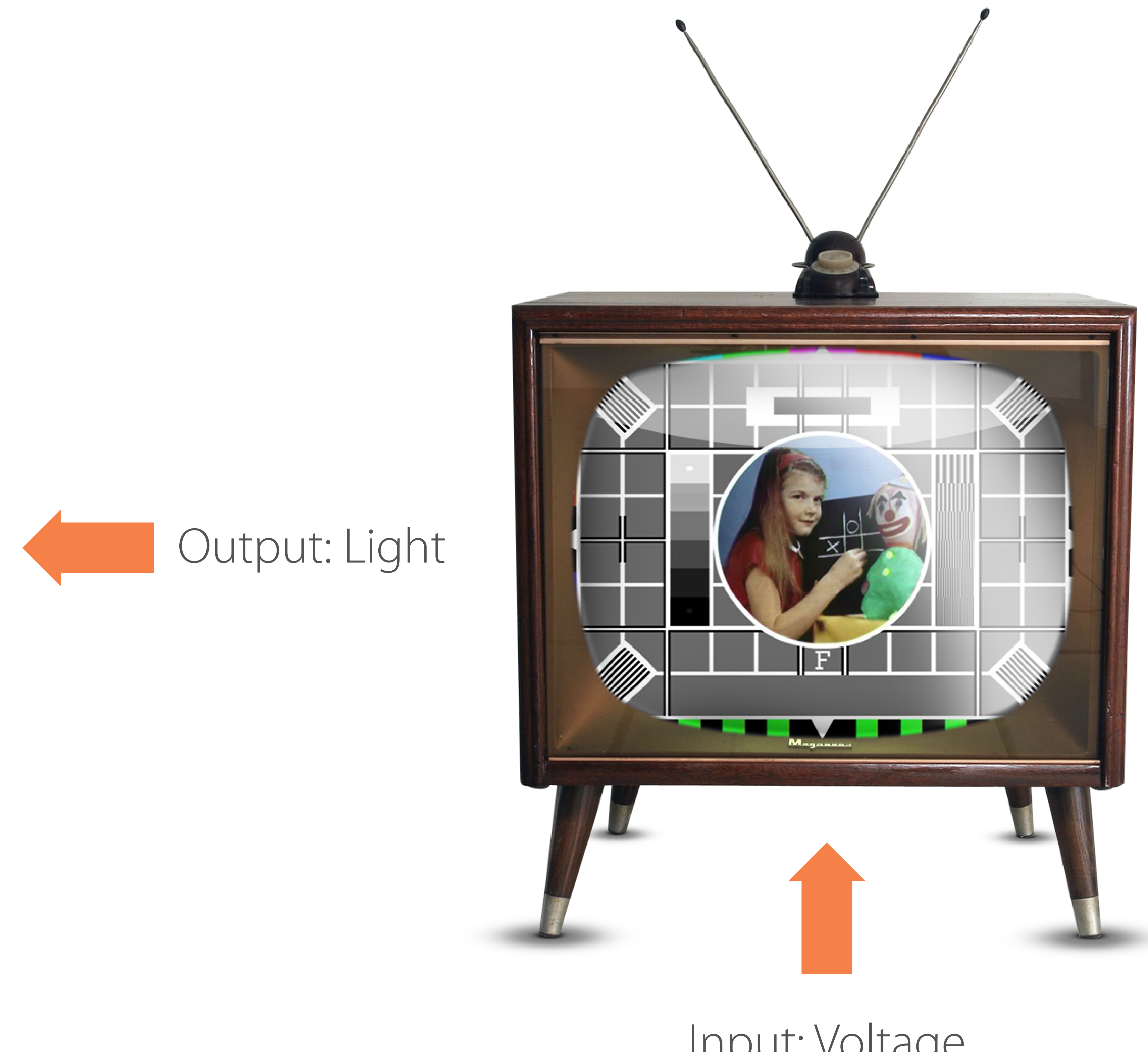




How Devices Display Images

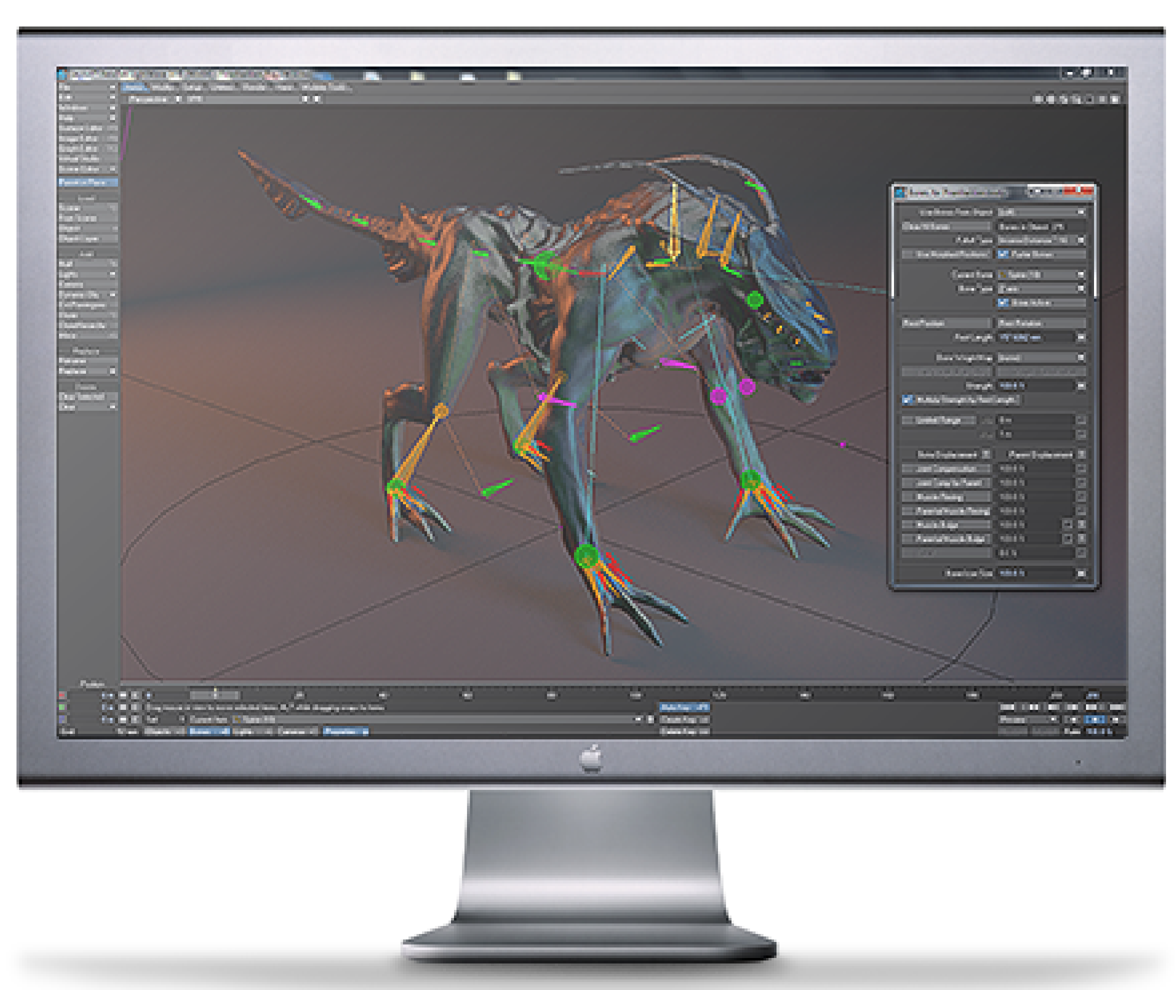
Or, to put it another way, on a CRT display, the relationship of input voltage to light output intensity is curved.
The mathematical description of this curve is called "Gamma"

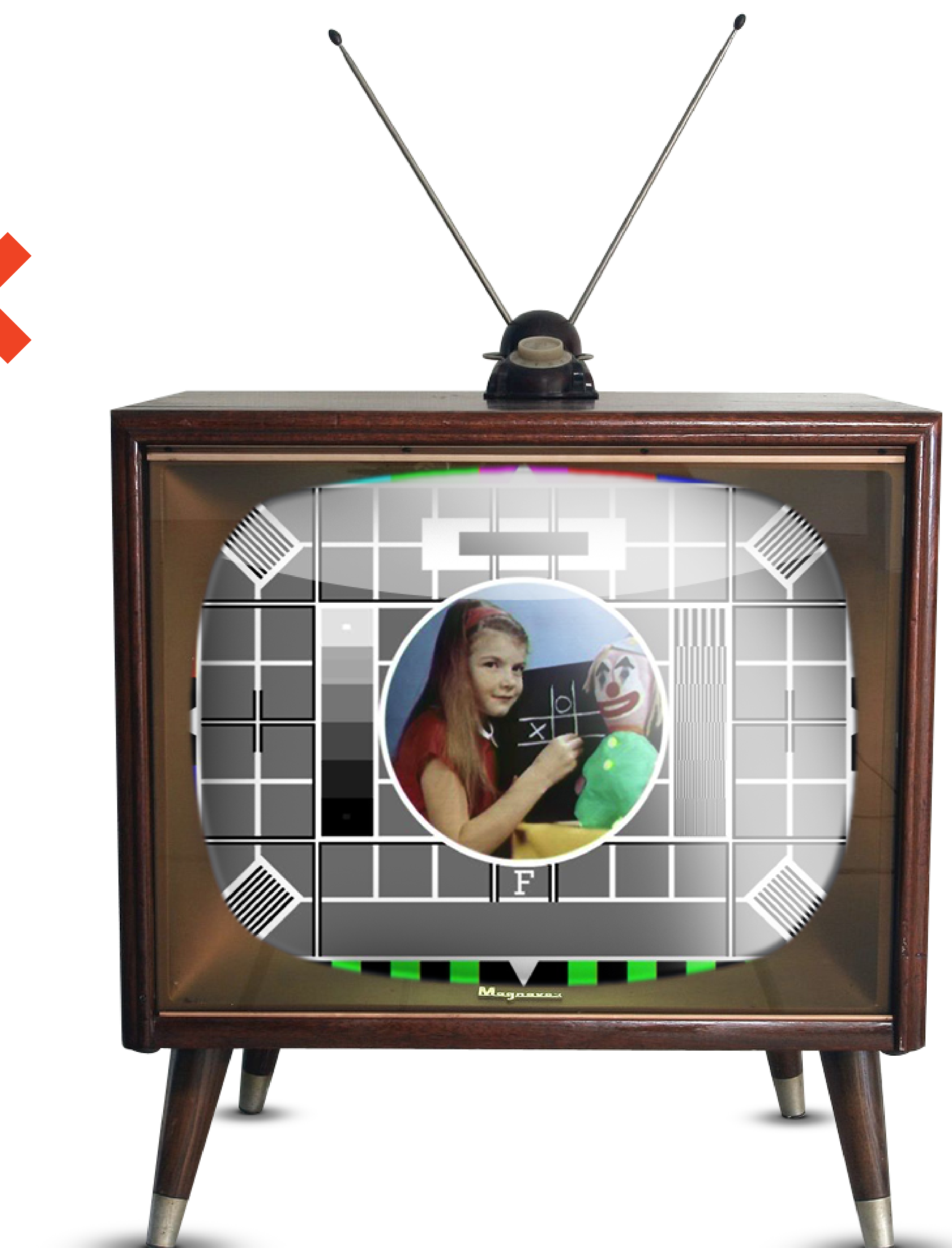




How Devices Display Images

Okay, so a CRT has a nonlinear Gamma curve, so what! Everyone uses TFT displays these days?





How Devices Capture Images

How Devices Capture Images

Electronic devices capture light and convert it to an electrical signal ...

Charge Coupled Device
Complementary Metal Oxide Semiconductor

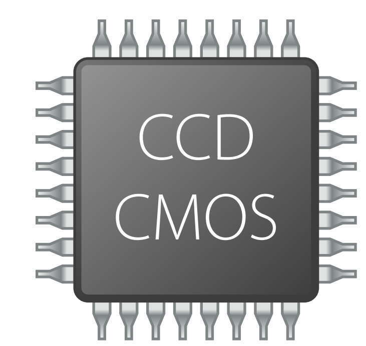
Input: Light

Converted to Electrical Signal

How Devices Capture Images

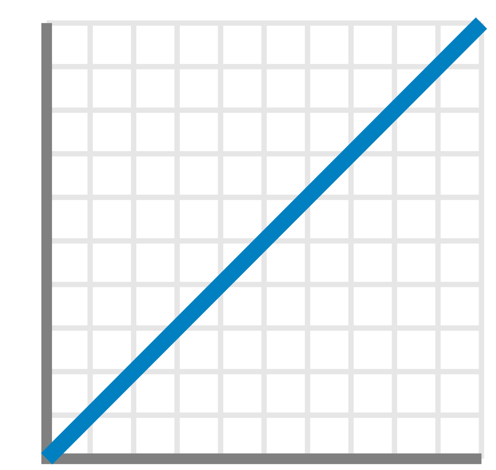
Cameras capture light intensity proportionally to the amount of light coming into the lens, or, they capture and store the light linearly.

Input: Light



Charge Coupled Device
Complementary Metal Oxide Semiconductor

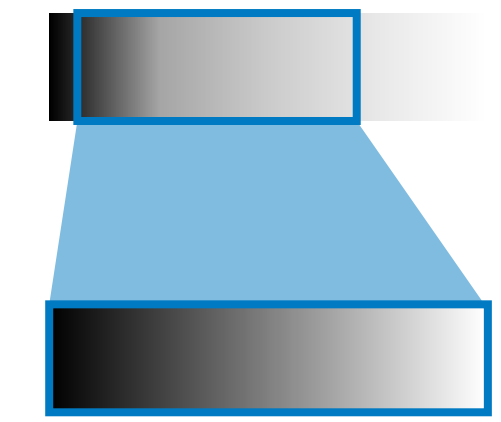
Converted to Electrical Signal



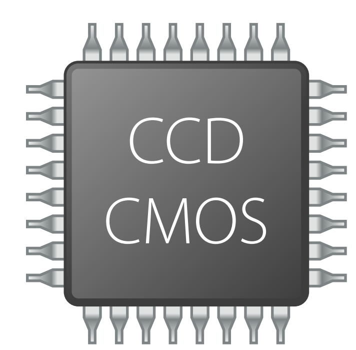
Light Intensity

How Devices Capture Images

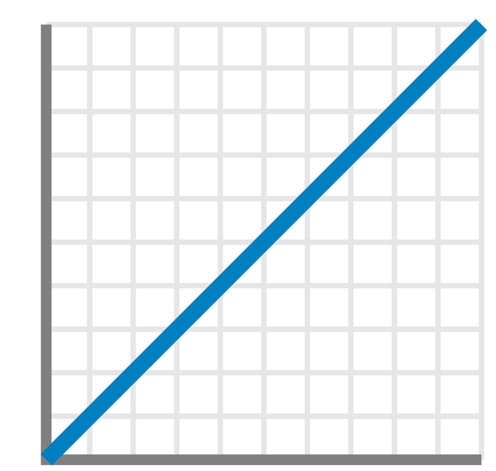
However, the electrical signal is Gamma encoded before it is saved or broadcast



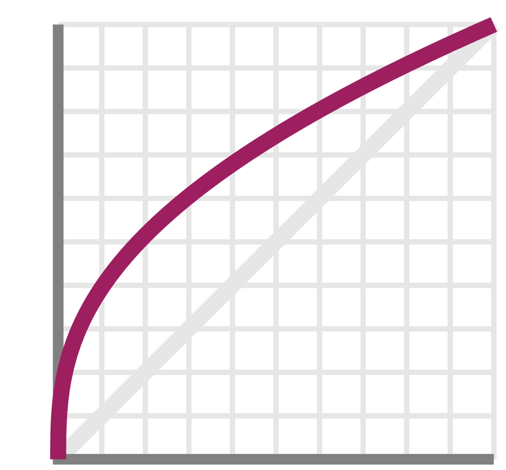
Gamma encoding takes advantage of the range our eyes can see, and re-maps linear data to 8-bit. This make values appear visually equal in intensity change, as well as making the data more efficient



Charge Coupled Device
Complementary Metal Oxide Semiconductor



Light Intensity



Gamma Encoded

Input: Light

Converted to Electrical Signal

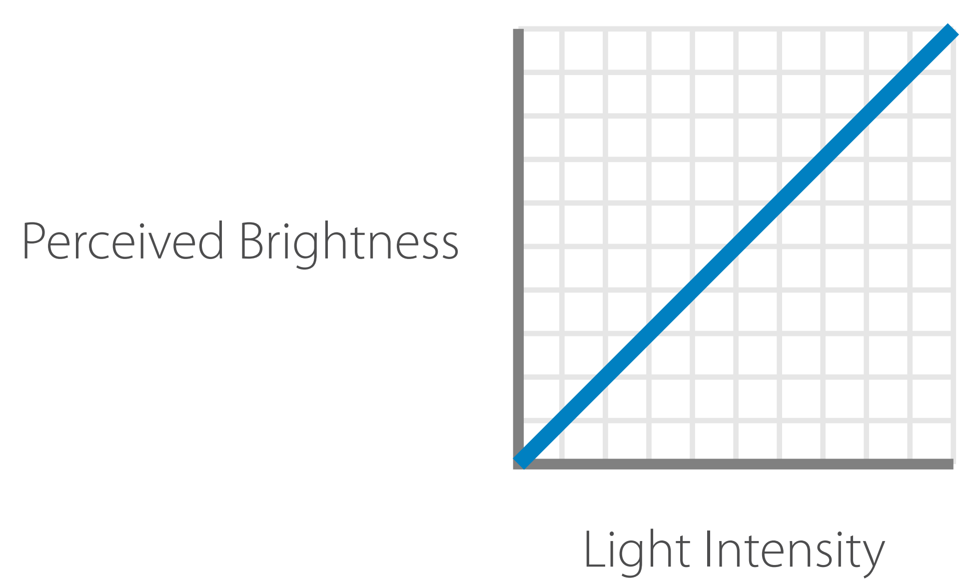
Output: Electrical Signal

How Our Eyes See

How Our Eyes See

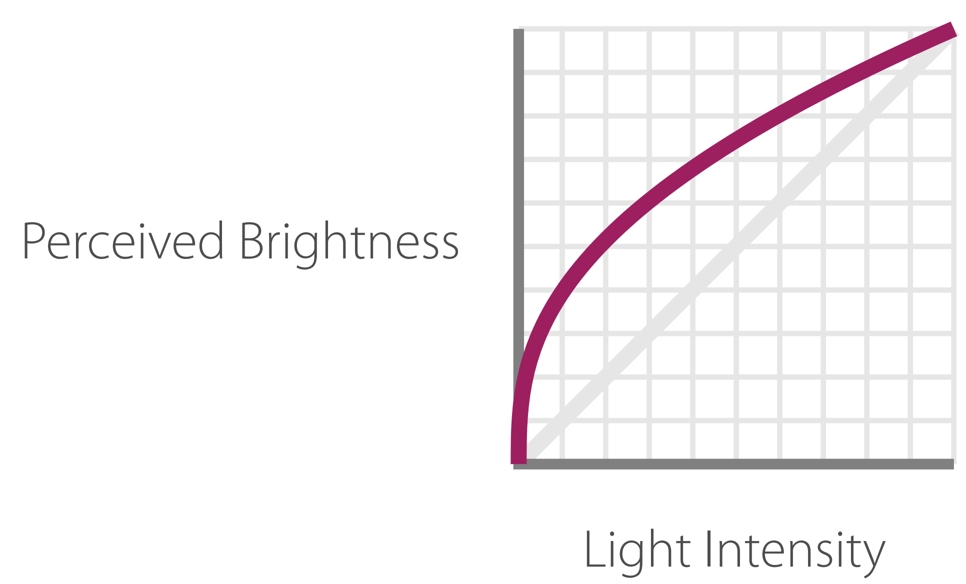
Our eyes do not perceive light intensity in a linear manner

Not a linear relationship



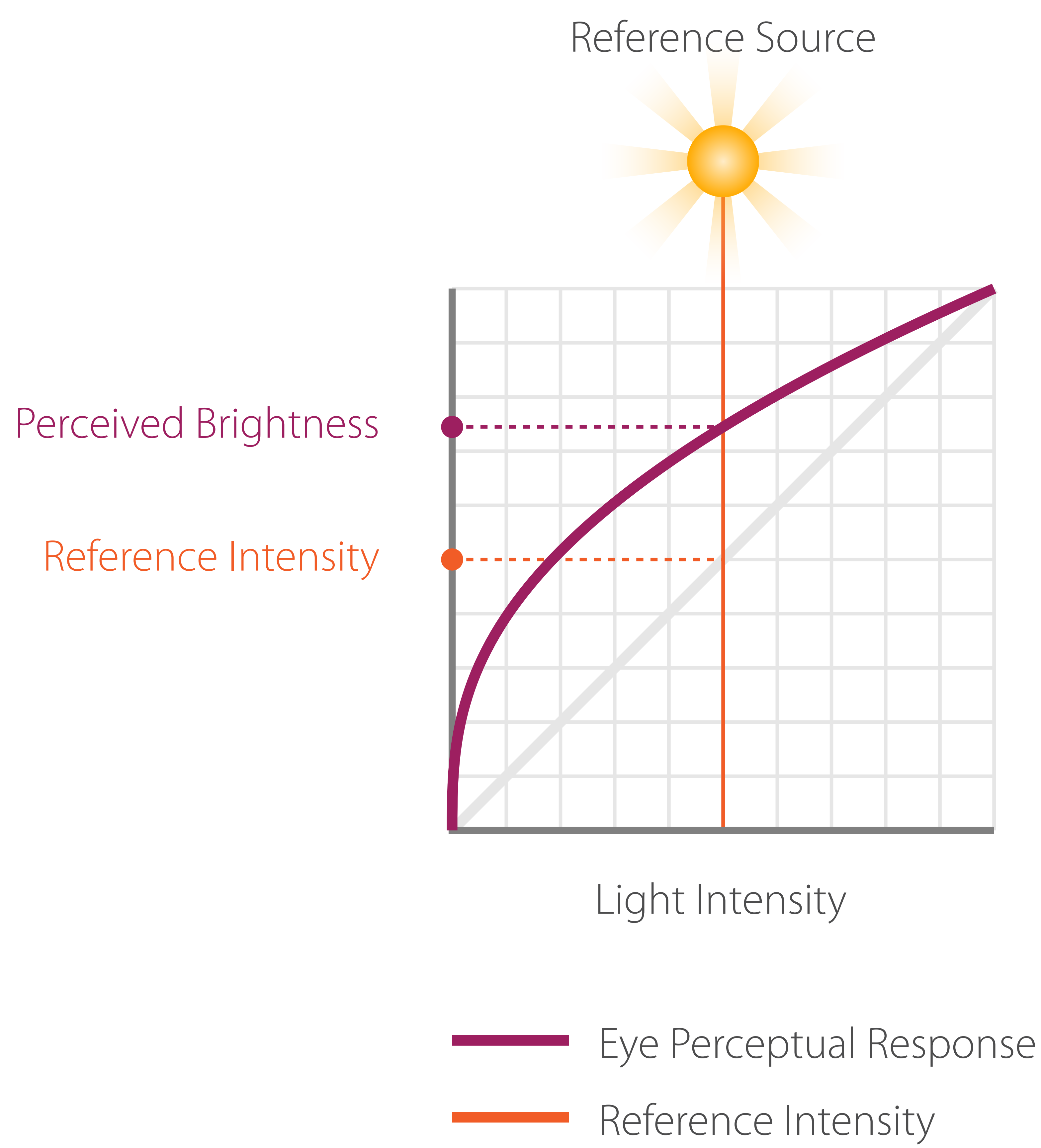
How Our Eyes See

Human vision has a nonlinear perceptual response to brightness



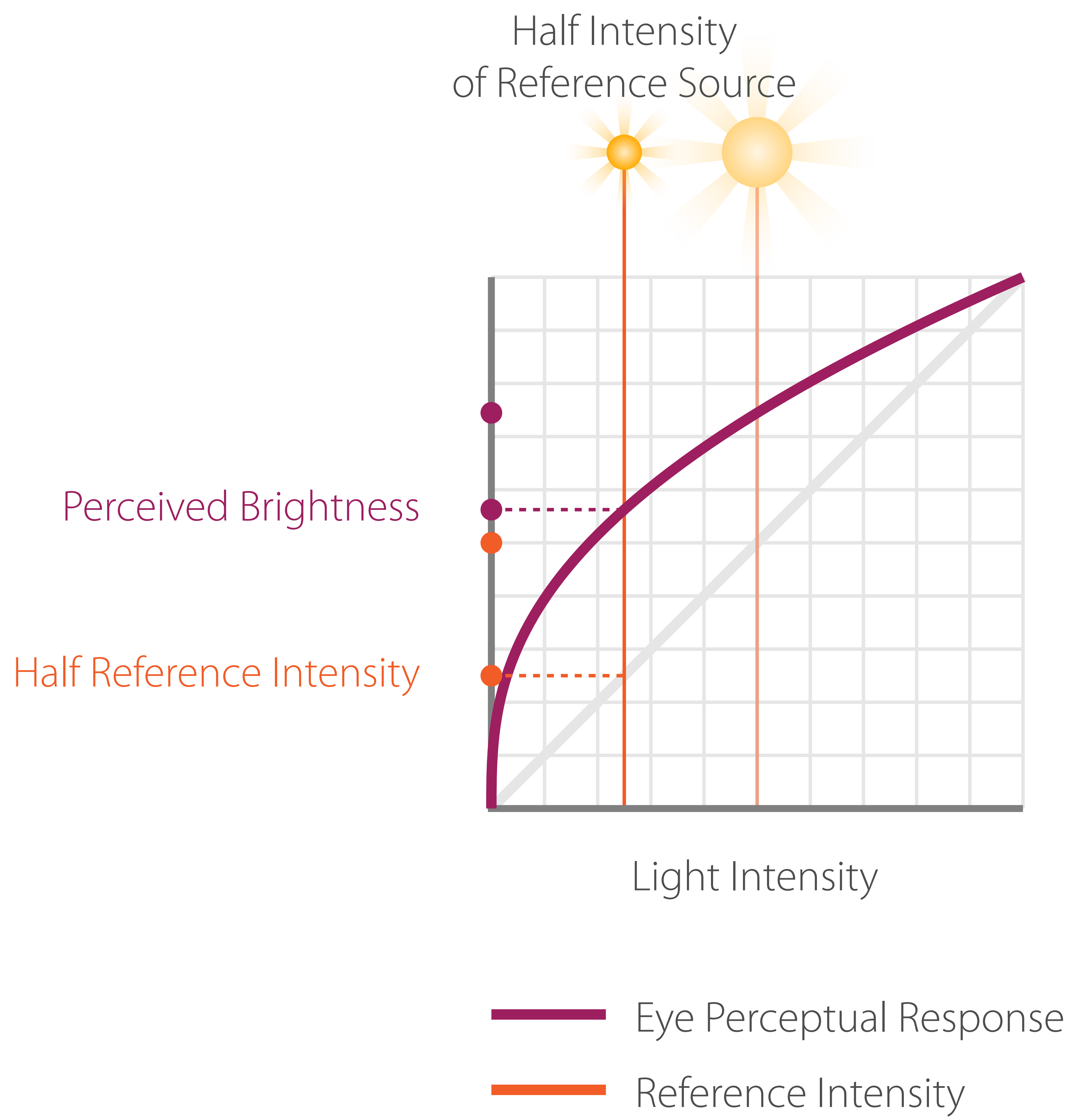
How Our Eyes See

If we were to look at source of light intensity (could be anything, not just an actual 'light')



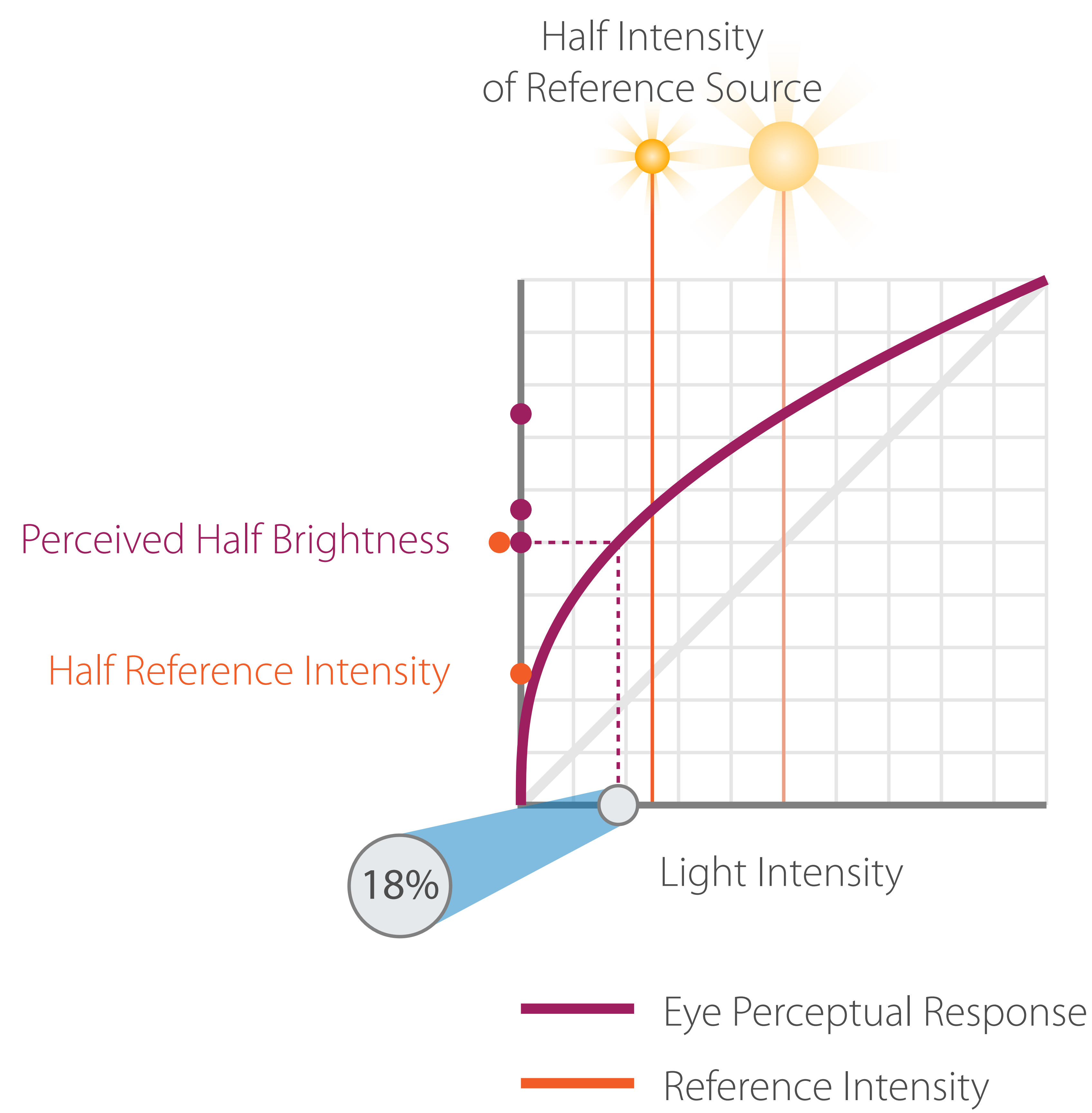
How Our Eyes See

Then look a source half as intense, we do not perceive it as half the brightness



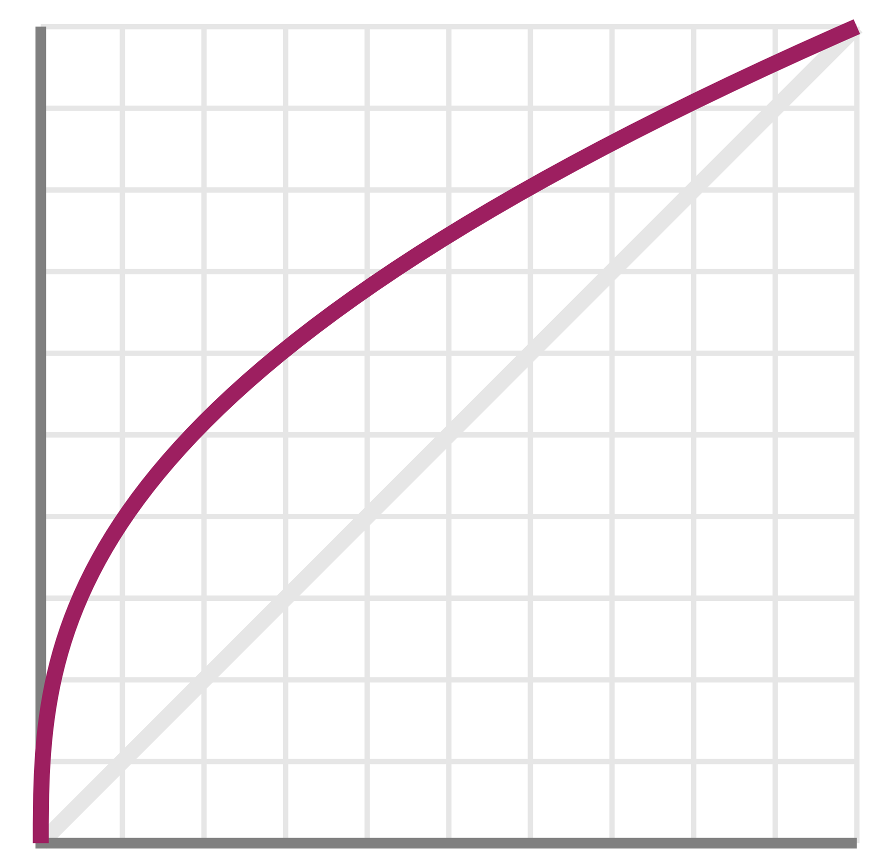
How Our Eyes See

For our eyes to *perceive* the source as half as bright, the reference intensity would need to be around 18%



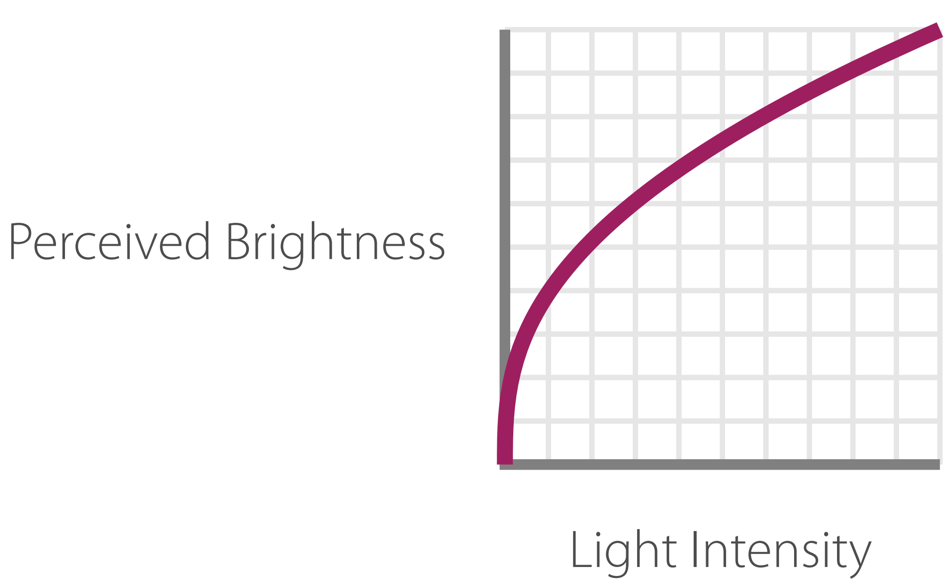
How Our Eyes See

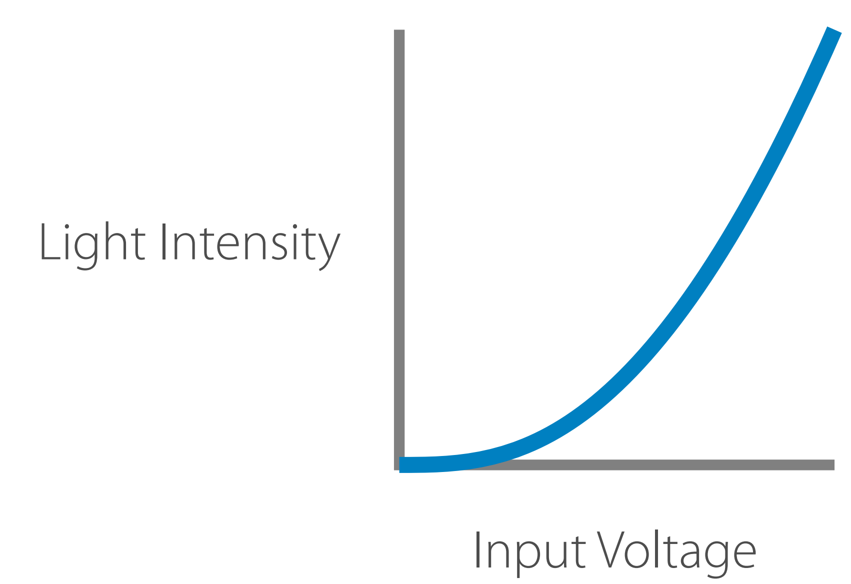
This curve looks familiar doesn't it!



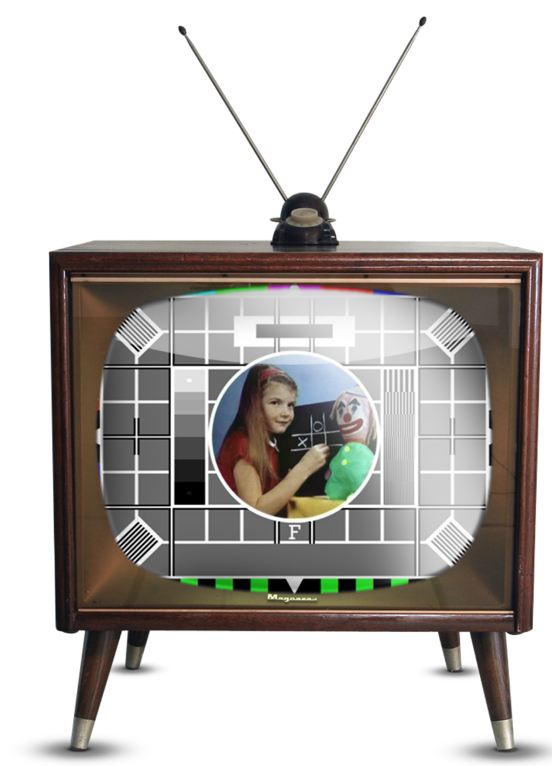
How Our Eyes See

In an amazing coincidence, the human eye perceives light intensity at almost the exact inverse of how CRT displays respond to input voltage



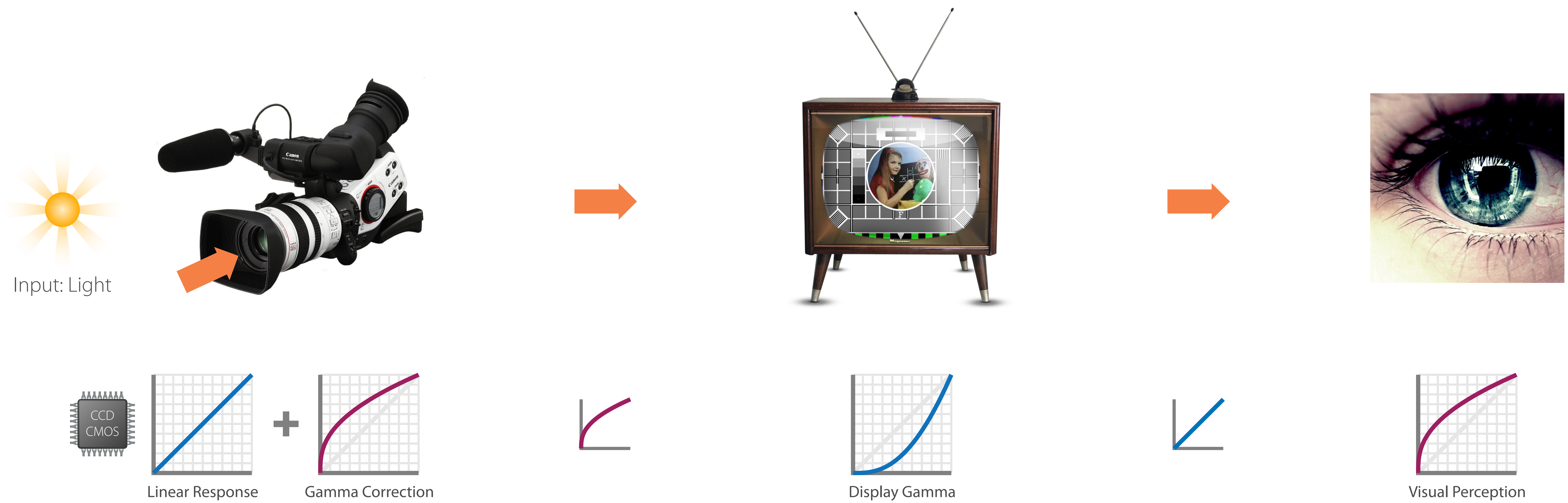


How This All Fits Together



How This All Fits Together

All of this is so that light intensity is compensated for how we see, and allows predictable intensity changes



Linear Workflow

This is same process in LightWave - to ensure predictable results