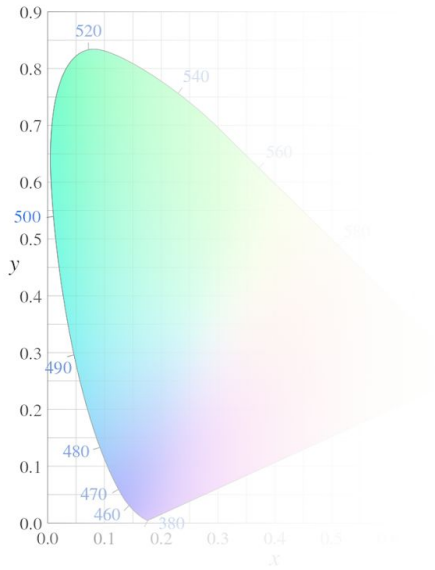
in Visual Effects

Welcome to this ACES explainer! I know you are anxious to find out all about ACES, but in order to explain ACES I need to explain more about color spaces in general. ACES is more than ‘just a color space’, but color spaces are a big part of what it’s about. And because color spaces is a very confusing subject, I present to you:

Color SpACES

in Visual Effects

Color Spaces



sRGB IEC61966-2.1
ITU-R BT.709
DCI-P3
Adobe RGB (1998)
CIE-xyY
AlexaLogC
Cineon
Aces2065-1
YCbCr

And many more...

Let’s talk about color spaces. First, they have confusing names. You shouldn't be put off by these names, they may look intimidating, but the intimidating part usually refers to the organization that defined the color space as a standard.

Then, every software package has a different way to implement them, using different terms to talk about aspects of color. Some information is derived from photography, some from tv, or early computer graphics. Some commonly used terms are just wrong. And then you ask someone if they can tell you what it’s all about and you get this very confusing lecture that ends in ‘just click here and check that box and then it should be ok… I think’.

I hope to lift the veil somewhat and give you a better understanding of what’s going on with color inside our computers.

So here's some of the stuff that’ll come up, besides ACES. We’ll do some theory first about color spaces, obviously, but also about white point, gamma, bit depth and LUTs. Then we’ll look at ACES and finally I’ll talk about working with this stuff inside your software through OpenColorIO - even if your software does not support OpenColorIO directly.

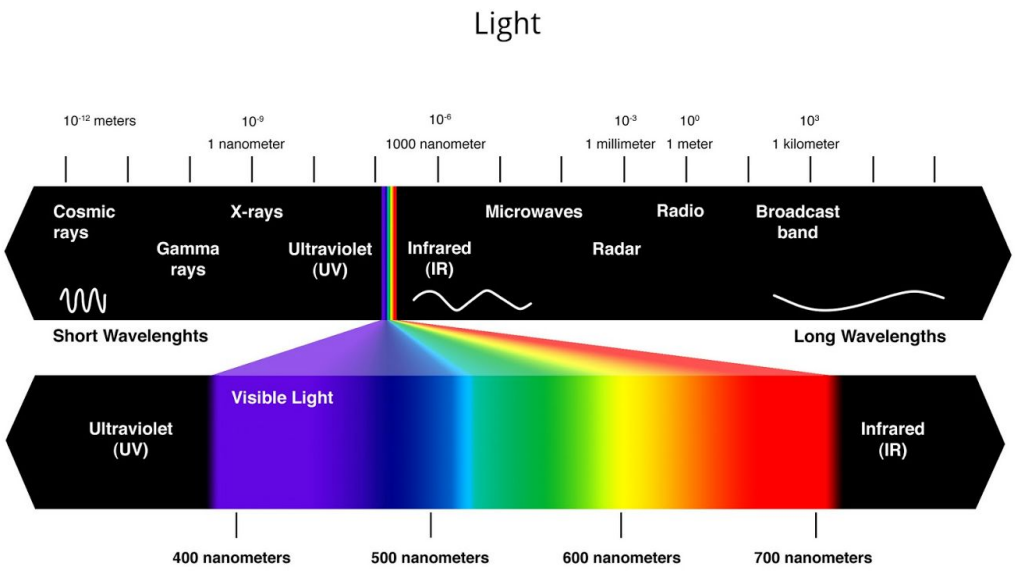
Disclaimer: I will not contend that everything I will explain is the complete truth, or even that I know the whole story. For reasons of clarity I have chosen to simplify some complex issues. I aim to give information that will help you make better decisions when working with color. Going too deep would cloud the issues at hand and make for a long and boring document. On the other hand, this article is intended for VFX artists, so a basic level of technical understanding is assumed. And be warned, things will get a bit technical…

Anyway, on to the theory!

Example Color Spaces

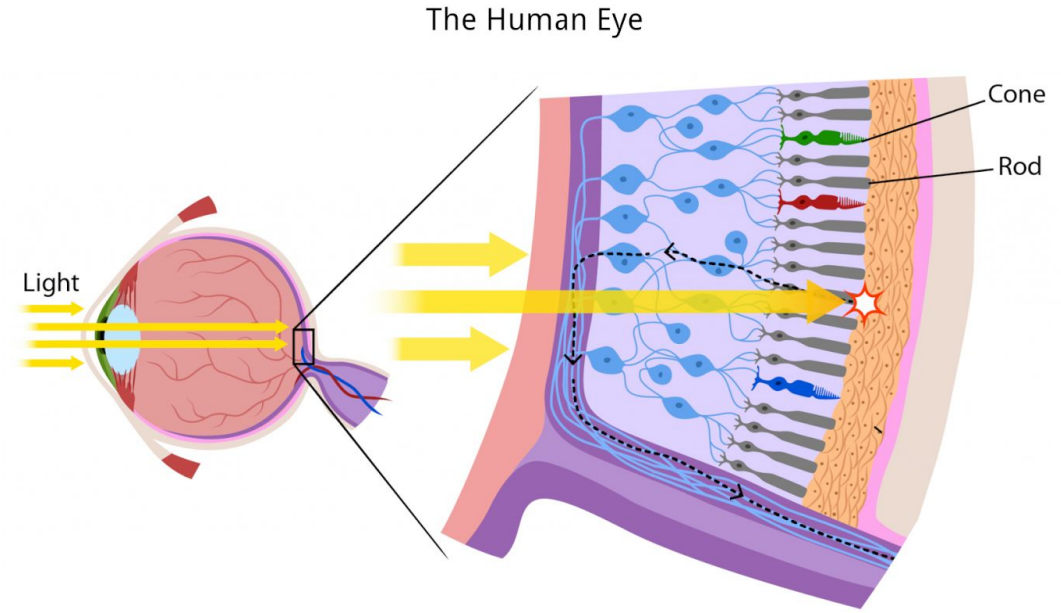
AlexaLogC sRGB DCI-P3

Lineair ACES2065-1 CIE XYZ



http://www.infohow.org/science/physics/electromagnetic-spectrum

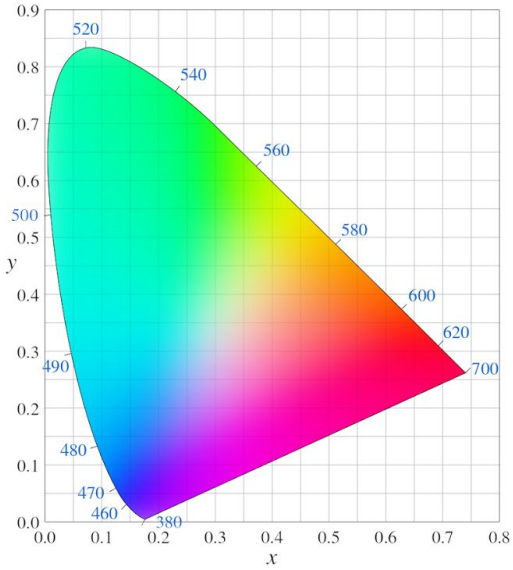
Everything that's related to the display of images is related to the human eye and the way we perceive the world. Our eyes are sensitive to light. Light is electromagnetic radiation. Radiation has a wavelength and a small part of all possible wavelengths are what we refer to as Visible Light.



https://askabiologist.asu.edu/rods-and-cones

The human eye perceives the world through special cells called rods and cones. The rods record luminance and the cones record color. There are three types of cones and they respond to different wavelengths of light. We have cones for red, green and blue light: RGB.

Gamut of Human Vision



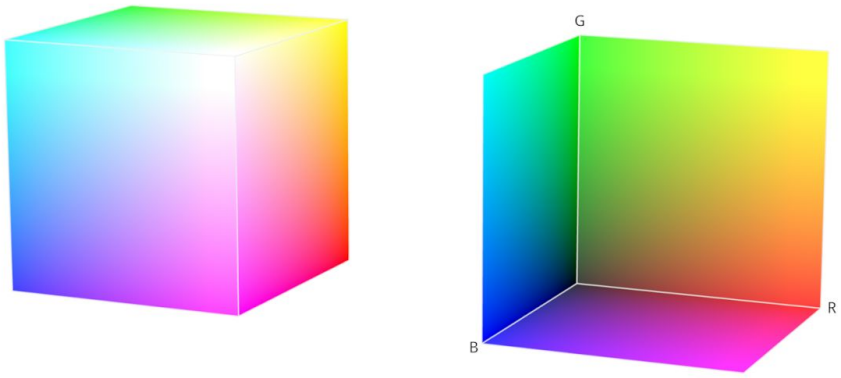
CIE 1931

In 1931 a bunch of scientists did some experiments to determine what's now called the "Gamut of human vision", or the range of colors the average person can see. This resulted in this graph.

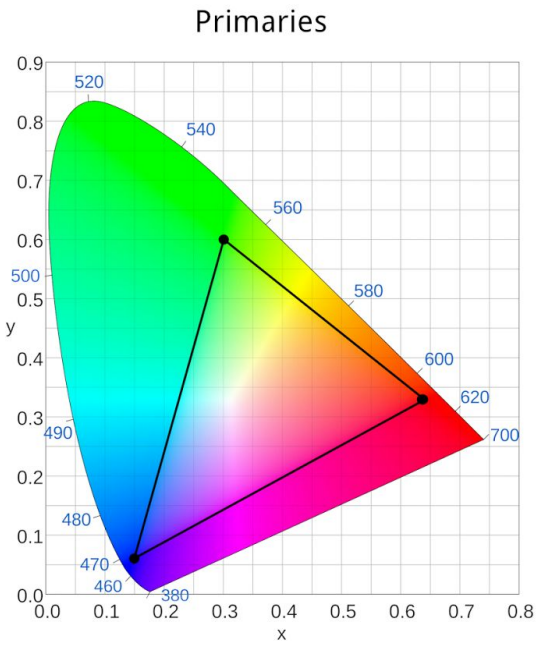
This weird horse-shoe shape is called the CIE 1931 chromaticity diagram and it's also referred to as the 'spectral locus'. It is widely used to indicate the gamut of a color space. In this graph we see the gamut of our own color space, expressed in chromaticity (x,y). Along the curved line you see the wavelengths of colored light.

In order to understand color as a space you have to realise this graph only shows chromaticity (colorfulness) and it does not take into account the luminance of the colors. You can imagine a third 'z' axis pointing out of the screen that represents luminance. One color space that makes use of this is the CIE XYZ color space - it was directly derived from CIE 1931. It was expressly chosen so that the Y axis would line up with the luminance. The luminance axis in CIE is therefore also labeled Y. This means we have three axes to define a color space in relation to human vision: (x, y, Y).

Color as a Space I

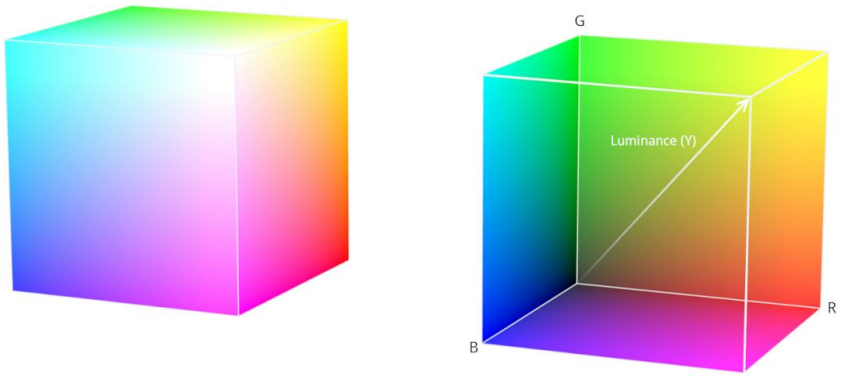


The term color space is not just some fancy name to describe what colors are possible, it actually refers to a space, with three dimensions. We've already seen three dimensions when we were looking at the CIE diagram. Now instead of expressing color in x, y and Y, or X, Y and Z you can also express color in values of R, G and B. And of course, a point in space is defined by three values and so you can plot a color space in values of RGB, resulting in this color cube. By looking up a point inside this cube any color within this color space can be created. Looking at the color cube you can see the R, G and B axes. The corners of the cube where R, G and B are at their max are called the primaries, the yellow, cyan and magenta corners are the secondaries.



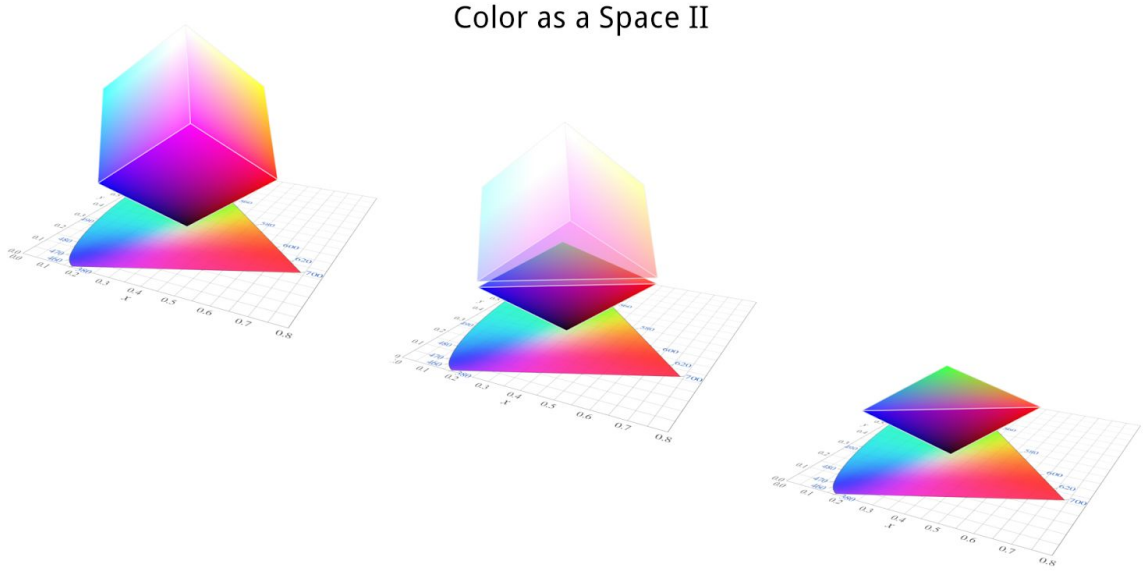
Let’s have a look at a common color space, sRGB. It’s widely used for computer monitors and it defines the gamut monitors can display. The gamut of a color space is represented by the area contained within a triangle. The corners of this triangle are the primaries: the reddest, greenest and bluest color the color space allows. Here we see the sRGB gamut and the primaries in the CIE diagram. It defines RGB values in [x,y] coordinates. Remember that the diagram doesn't show the luminance coordinate, but it is defined! For sRGB pure red is defined as RGB[1,0,0] = x,y,Y[0.64,0.33,0.2126].

Color as a Space II



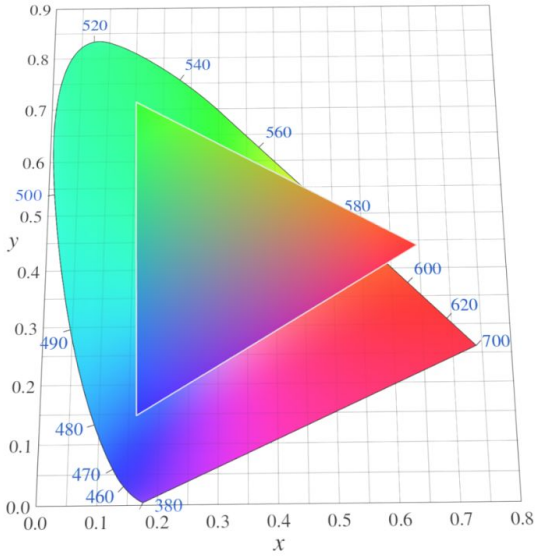
Now we just saw color as a cube and now we're looking at a triangle. Let's see where that triangle comes from. Remember the RGB cube. Its inner diagonal represents the luminance axis and the R, G and B corners are the primaries.

Color as a Space II

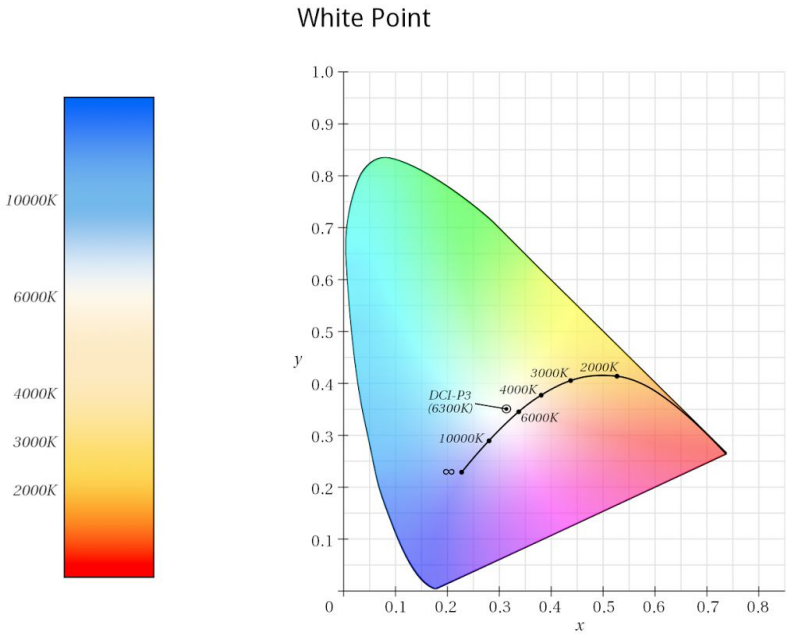


Now, let’s rotate it so its inner diagonal points straight up, like in the CIE system. When we cut the cube exactly through the corners with the primaries and remove the top half, we’re left with an inverted triangular pyramid.

Color as a Space II



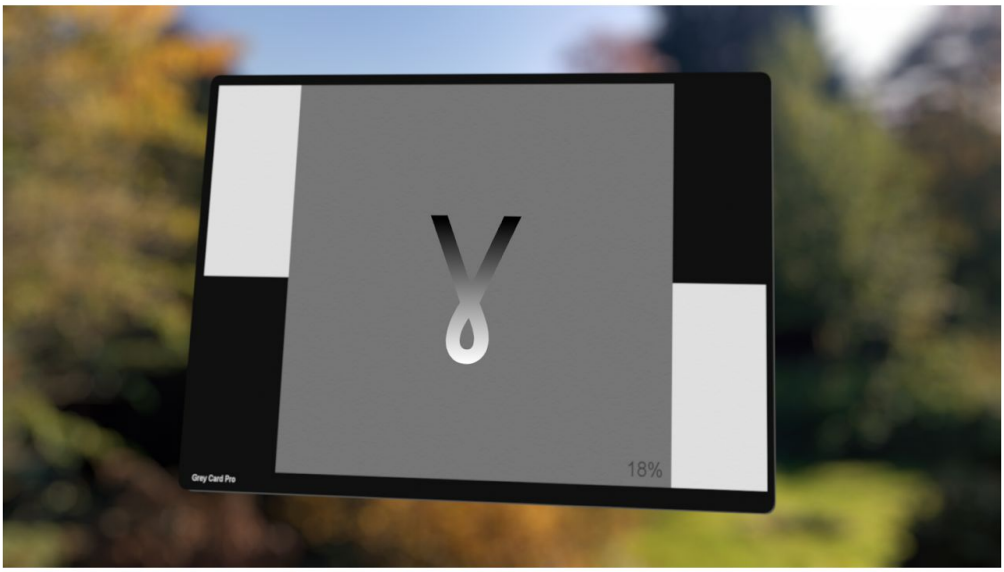
Looking at a top view of this pyramid over the CIE diagram, you can now see where the triangular shape of the gamut comes from. The triangle does not show you all possible colors in a color space, but together with the Y coordinates and the white point it is enough to define the gamut. This is what it means when we say the triangle in the diagram represents the gamut of a color space.



Besides the primaries a color space defines a white point, or illuminant: the [x,y,Y] coordinates that correspond to RGB[1,1,1]. Because the Y coordinate is often 1 (the white point is the brightest point) it is usually enough to define only the [x,y] coordinates. You have to realise that this is different from the white point when editing photographs for example. The white point of a color space defines the physical color this color space calls ‘white’. It can not be used to edit an image, it makes sure that the white in your image appears white on your display medium.

Note: Though technically not correct the white point is often referred to in terms of color temperature. The sRGB white point is referred to as D65, or Daylight 6500K. Note that the white point does not *have to be on this line, it can be greener or more pink. For example, the DCI-P3 white point is referred to as 6300K, but it is in fact more towards the green. The only way to accurately talk about white point is in [x,y] coordinates.*

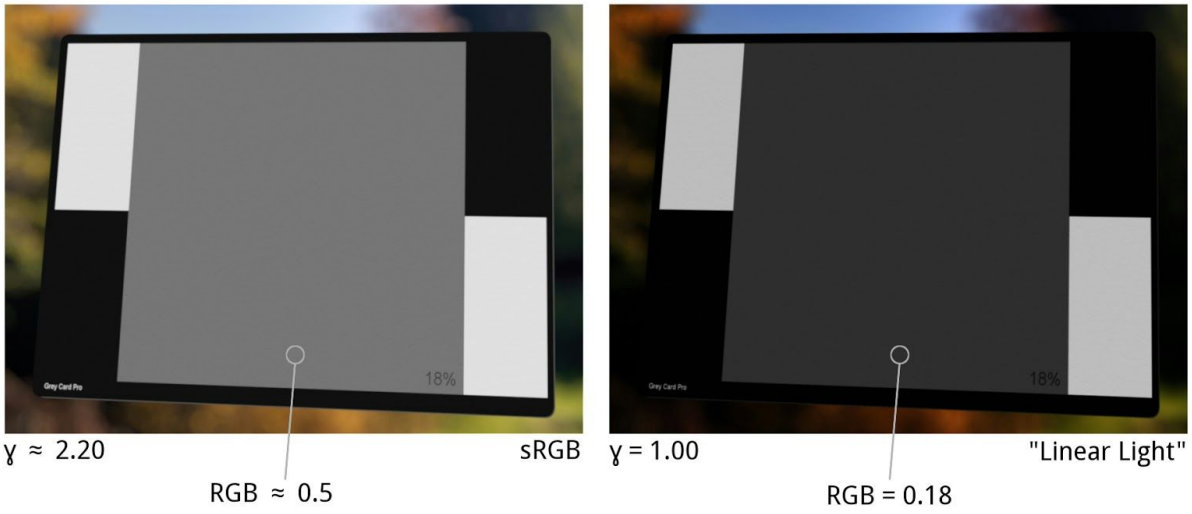
Gamma



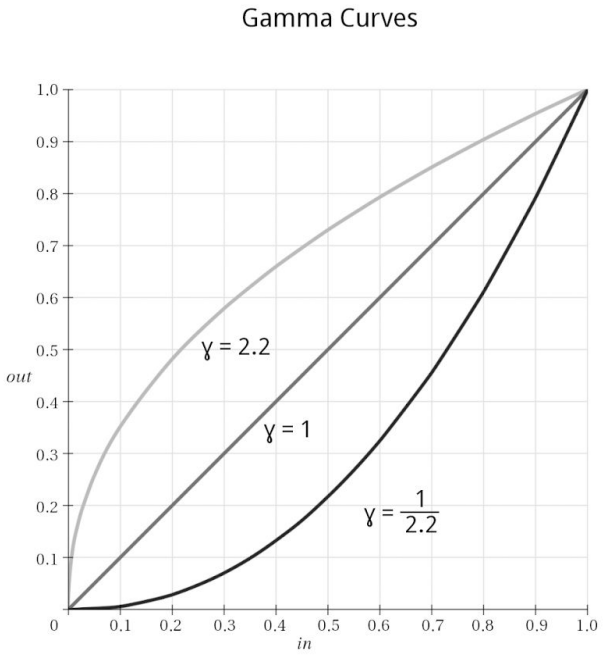
One important parameter to define a color space is the gamma. Gamma is an important concept when you're working with images and especially in VFX, where we often try to mimic the real world in great detail.

Gamma, in short, describes the relationship of the brightness of a pixel to the brightness of the light value that that pixel represents. Now we need to go back to the physical properties of light. In the real world when you change a 20W bulb for a 40W bulb it emits twice as much light. This is what it means when we say that light is ‘linear’: double the energy, double the brightness. Our eyes, however, do not work like this. We are much more sensitive to changes in the dark than we are to changes in bright light. Consider our light bulb: add 20W to a 20W bulb and it will seem a lot brighter. Add 20W to a 100W bulb and the increase in perceived brightness is a lot less.

Gamma Corrected Images



Now instead of light bulbs we're looking at pixels on a monitor. And to make an image look good to our eyes the monitor needs to replicate this perceived brightness. The way it does this is through gamma. Applying gamma to a picture transforms the brightness of the pixels in a non-linear way. That means that dark pixels are affected differently from bright pixels. Gamma only changes values that are not 0 or 1, so black and white (in a grey scale image) are unaffected. When you raise the gamma of an image it will look brighter, because the darker the original pixel was the more it will get lifted. Bright pixels get lifted too, but a lot less.



For completeness here are the three most commonly used gamma curves.
γ = 1 represents linear gamma. This has no effect.
γ = 2.2 represents the new values after applying a gamma of 2.2 to a linear image.
γ = 1/2.2 represents the inverse operation, used to 'undo' a gamma of 2.2 to create a linear light image.

As you can see, 0 and 1 are unaffected and darker values are affected more than bright values.

Be aware though, that 2.2 is by no means the only gamma value in play. DCI-P3, a color space designed for projectors, has a gamma of 2.6. The gamma of the human eye is rated at 2.5. To calculate the result of a gamma curve yourself you can use the following formula:

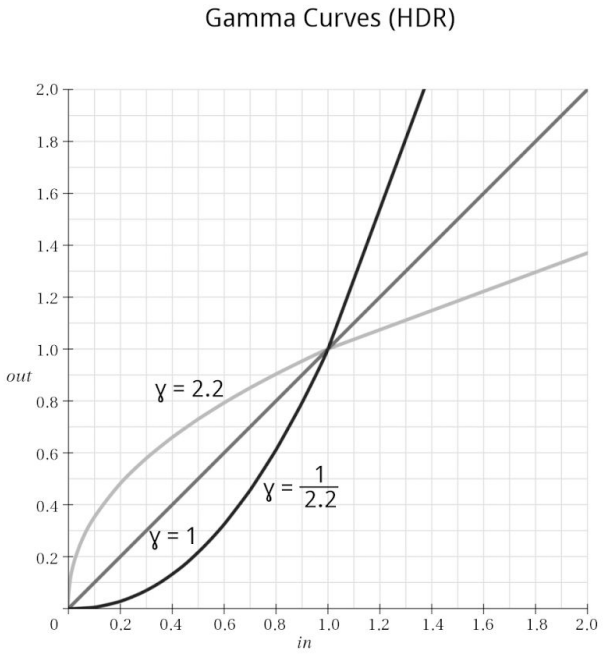
$$input\_value^{(1/gamma)} = output\_value$$

So for a properly exposed 18% grey card, viewed by a human eye:

$$0.18^{(1/2.5)} = 0.5$$

Which is exactly mid grey. Now you know why a grey card is called mid grey, even though it is only 18% grey!

Note: whether you're supposed to use gamma or the inverse gamma (1/gamma) really depends on the software implementation. Often a higher value of gamma will result in a brighter image in which case the above formula is implemented. However sometimes this is not the case and the formula $input\_value^{gamma} = output\_value$ *is used, so that means a higher gamma value results in a darker image. What you need to keep in mind is that if your software applies the inverse gamma you need to enter the inverse gamma value. So for a gamma of 2.2 this would be 0.4545.*

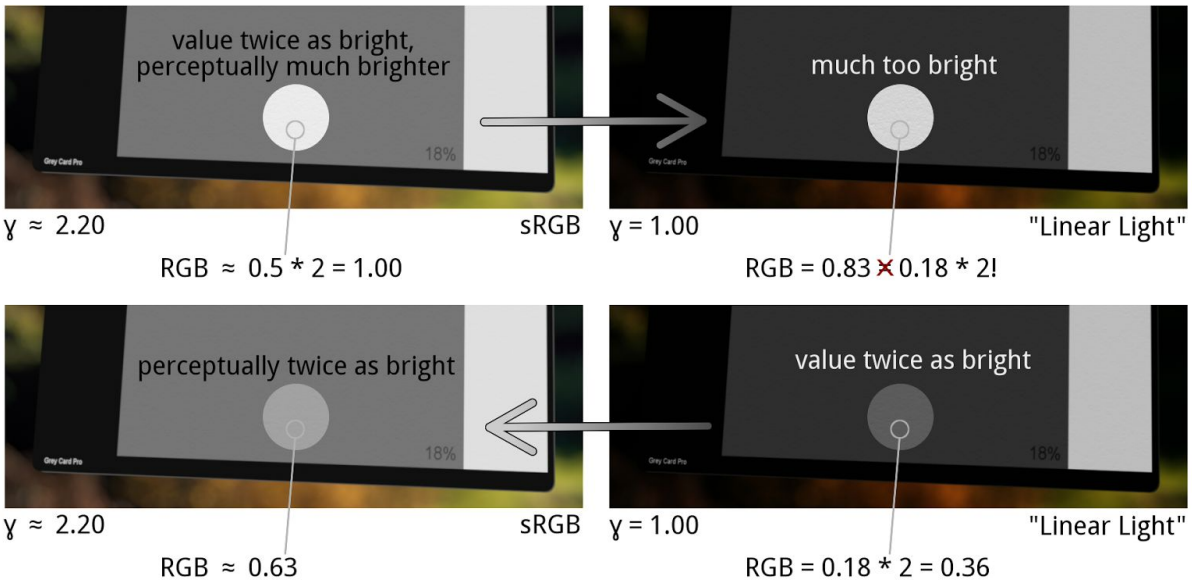


Now it is time for a word of warning: when you are working with HDR images the need for working with linear gamma images is extra important. The gamma operation works on values that are not 0 or 1, but HDR images can hold values well above 1. Consider what happens to a bright specular highlight with a linear value of 4 when the gamma formula is applied.

$4^{(1/2.2)} = 1.88$

While applying a gamma of 2.2 lifts values between 0 and 1, values of more than 1 are lowered! So your superbright specular is now a lot darker (more than half!) due to the gamma change. You can imagine how this can wreak havoc on your composite… Let's have a more detailed look at the influence of gamma on compositing operations.

Gamma Corrected Math



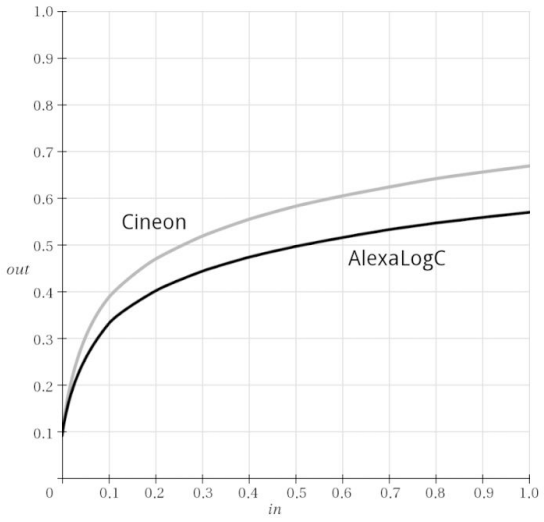
When we are compositing or rendering out cg the computer assumes it's working on linear images, meaning images in which the pixel values represent light values. This is why we need to convert our images to linear gamma. If we didn't we would make all kinds of mathematical errors in our composites and when rendering out cg.

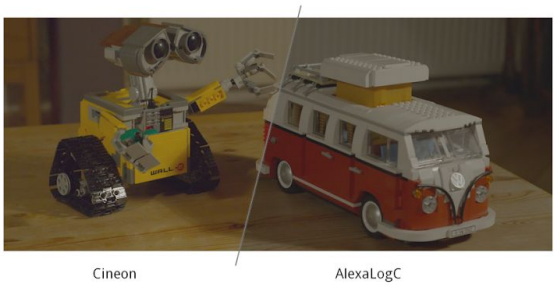
Consider this example: we have this image of our trusty grey card and we want to make the image twice as bright, perhaps we want to simulate a lamp shining on it that's twice as powerful. In the top left image we see the gamma corrected grey card. The middle grey has a value of approximately 0.5. To make it twice as bright you would, logically, double this value to about 1. But this looks a lot brighter than you would expect. Let's see what happens when we convert this image to linear gamma (top right). The original value of the grey card, 0.18, hasn't been doubled - it has been more than *quadrupled*. This is a typical error due to applying linear math on a gamma corrected image. Now let's have a look at linear math on a proper, linear image. In the lower right we see our linear image, which is now our starting point. After doubling the grey value in linear gamma and applying a gamma of 2.2 we can now see the expected result: the patch looks twice as bright. The value however, has only increased by 0.13.

Ok, now that that's clear, you might wonder why I didn't just add 0.13 in the first place, to perceptually twice the brightness. Well, doubling the perceptual brightness of a source value of 0.5 may be achieved by adding 0.13, but how much should we add for 0.6? Or 0.4? The answer, of course, is not 0.13, because gamma corrections are non-linear in nature. You would have to calculate the proper value to add for each pixel separately, which is exactly what we achieve by working on a linear image!

A note about gamma and the linearity of light: even though a linear gamma ensures the pixels accurately represent the light value in the scene (assuming no color grading operations were done in non-linear gamma) they do so in a relative way. The computer has no way of knowing the f-stop of the lens or how many ND filters were used or how bright the studio lights were that were supposed to represent the sun. This means that you can't assume the light values will match from shot to shot or between your foreground and background plate - you will still need to match them! Just try to match them using gain and lift controls and not the gamma ;-)

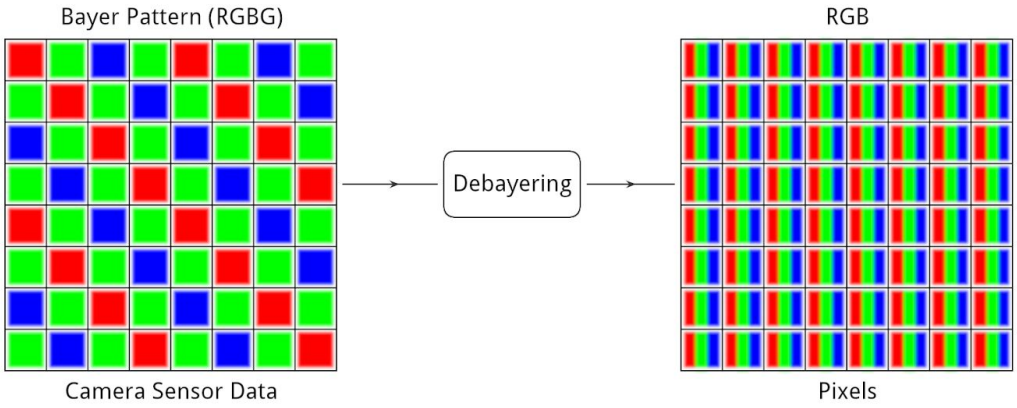
Log Gamma





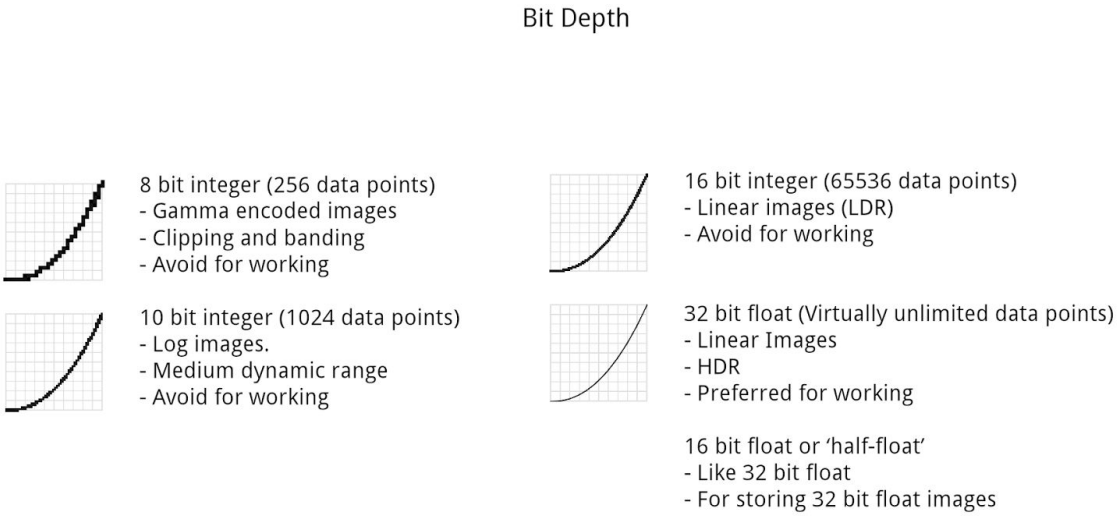
Another important gamma is log gamma. The use log gamma to store digital images was originally invented by Kodak in the Cineon image standard (.cin or .dpx files). Designed for storing digital scans of optical film, it is essentially a way to compress the image by using a gamma curve. This was needed because in the early days of film digitization computers were not as powerful as they are today and storage space was limited. Compressing the image data allowed for efficient storage of image data at a minimal loss. The gamma curve Kodak chose is *log*arithmic - if you remember your math classes you'll know the logarithm is the inverse of exponentiation. The choice for a log gamma is actually very logical. Film exposure is measured in 'light stops', where each stop represents a doubling in in the amount of light, which means it is an exponential function. If you store data linearly each stop of light requires double the amount of bits from the previous stop. That means that highlight data requires a lot of bits to be stored linearly, but as we've seen in the gamma discussion before, our eyes are relatively insensitive to highlight detail and *very* sensitive to shadow and midtone detail. By applying a log gamma to an image the exponential character of linear light values is in effect reversed so each stop of light gets the same amount of bits assigned. If you were to store 10 stops of light in a 10 bit file with log gamma applied each stop of light would have 1 bit assigned. This results in the typical 'washed out' look of log images, as you can see above, but it is much more efficient to store image data this way. Of course, some fine highlight detail is lost. But because our eyes and brain do not register this detail in real life anyway this is quite acceptable in practice. Log images can be stored in higher bit depths to allow for storage of more stops of light, 12 or 16bit log files are not uncommon - but more on bit depth later. It's worth noting that though a 10bit log file is not HDR it does contain HDR data, after linearization the image can contain data with values roughly between 0 and 13!

RAW files



When dealing with raw files, either from a still camera in the form of NEF or CR2 files, or from a digital film camera (ARI, R3D) you have to realize that these files are not yet decoded into a particular color space. Raw files do not store image info in RGB values, they store the raw data that was recorded by the camera sensor. To understand why sensor data is not in RGB let's look at a close up of a camera sensor.

A camera sensor is made up of photosites, small dots that give off an electric pulse when they are hit by light. Since each photosite can only give off one pulse the light is filtered so that only red, green or blue light can reach a photosite. Instead of recording equal amounts of red, green and blue light photosites often contain more green samples, because to the human eye green contains the most brightness information. A typical layout of a sensor's filter would be RGBG, which is called a Bayer filter. In order to properly display and work with these images they need to be converted to pixels with RGB values. This process is called debayering. When the image is debayered it is fit into a color space. It depends on the camera make and model which color space best fits the debayered image, but it is best to err on the side of safety and choose a wide gamut color space, you can always decide to convert to a smaller gamut later if needs be. A word of warning: debayering gives you many options to manipulate the image - think white balance, denoise, highlight retention, etc - and some of these cannot be altered after debayering. This means you have to take care when debayering. Try to keep as much information as possible and debayer to a scene referred color space. When you're doing visual effects on shots from a scene shot on R3D, for example, be aware that should the colorist decide to change the debayering settings of any non-vfx shots in the sequence you will need newly debayered images, which will cause you to have to redo all your vfx! If at all possible try to decide on final debayering settings before starting vfx work and make sure the colorist understands the implications of changing the settings later on!



One of the reasons gamma encoding is used is to make efficient use of limited bit depth. An 8 bit image, like a jpg file, has a limited amount of data points. For each color only 256 color levels are available. It would be a shame to use this limited amount to store a lot of highlight data that will be ignored by the viewer anyway, while at the same time losing precious detail in the midtone and shadow areas. By applying gamma encoding to an image the highlight data becomes 'compressed' into fewer bits and the midtone and shadow data is 'stretched out' over more bits, allowing for more visible detail.

For efficient storage of small images this is not a bad idea, but when we are working with images we need to make sure we are not limiting ourselves to using only 8 bits as this would cause all sorts of artifacts like clipping and banding. So here's a short overview of commonly used bit depths, what their properties are and when to use them:

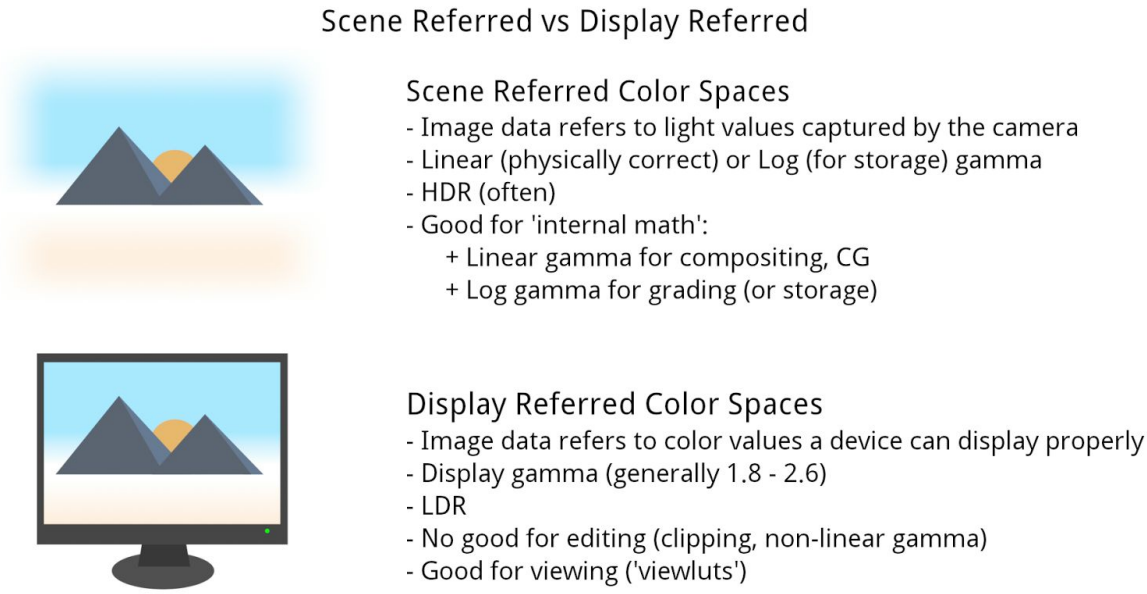
8 bit integer: 256 data points per channel.
Use for storing gamma encoded images and displaying on consumer quality hardware.
Watch out for clipping and banding. Do not work in this bit depth unless you know what you're doing.

10 bit integer: 1024 data points per channel.
Commonly used for storing log images. Is useful for storing medium dynamic range data in a limited amount of bits at the expense of highlight detail due to the log compression.

16 bit integer: 65536 data points per channel.
Can be used for storing linear images but preferably not for high dynamic range images, because the amount of data points is still limited. If possible avoid working in this bit depth.

32 bit float: Virtually unlimited amount of data points per channel.
Preferred bit depth to work in. Ideal for linear gamma, high dynamic range. No risk of clipping. Use it if you got it!

16 bit float: Also referred to as 'half-float'. Virtually unlimited amount of data points per channel, but smaller file size.
A very tiny bit of data is lost compared to 32 bit float, but this does not cause problems. It is the recommended bit depth for storing linear gamma, high dynamic range images and is used by all the major VFX houses.

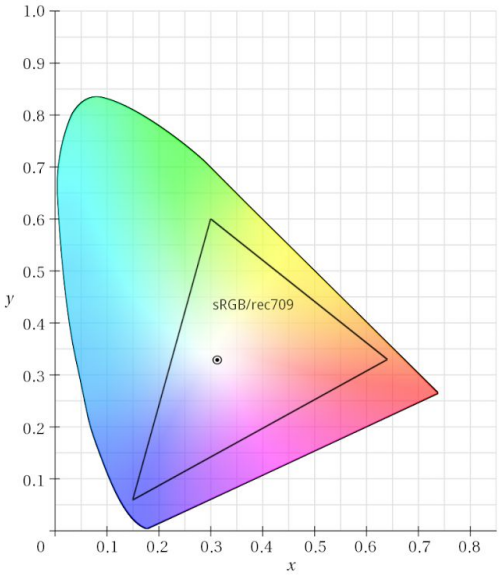


Most of the color spaces we've talked about up to now are Display Referred. This means that they are designed to make your image look properly on a given output device like a monitor or a projector. sRGB and DCI-P3 are examples. Other color spaces are Scene Referred. This means that the way the data is stored relates to the scene that was filmed, or more generally the data represents the light values as they hit the camera sensor, be that a physical camera like the Arri Alexa or an imaginary camera in a 3D package. Scene referred color spaces come in two types: working spaces and storage spaces. Storage spaces are not intended to work in, they merely try to store all the data from the camera. They focus on efficiency and fidelity. Often storage spaces are in log format (which is effectively a really strong gamma curve), so when you view it without converting to a display referred color space they look really flat and greyish. Examples include AlexaLogC, Slog or ACES (which is linear). Working spaces are intended to work in, so they have a linear gamma and look really dark without converting. Examples here are Linear sRGB or ACEScg.

Note: even though storage spaces are not intended to work in there are software packages that work natively with log files, most notably grading programs. This just means that the software does an internal log-to-lin conversion before applying the math and the tools in the software take this into account. Feeding such a package with a linear image will result in a very different behavior. Today most packages allow you to use a form of color management to make sure the tools get the kind of data they expect.

So a display referred color space ensures that your image doesn't look funky when you try to watch it on your phone, while a scene referred color space makes sure that you can store or work with the image without mucking it up.

Comparing Color Spaces

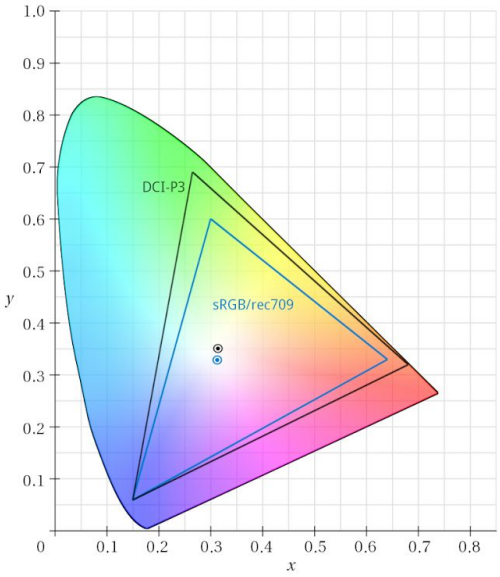


sRGB

Let's examine a few color spaces in light of what we've just seen.

First we have sRGB. Its full name is sRGB IEC61966-2.1. As you can see it has a relatively small gamut. It's white point is defined as x = 0.3127, y = 0.3290, which is 6500K, or D65. It has a gamma curve that is approximately 2.2, though it is in fact defined a little differently - it has a linear part near 0 and curves slightly off of 2.2, but for all intents and purposes 2.2 will do just fine. It's interesting to note that sRGB, rec709 (ITU-R BT.709) and "Linear Light" share the same primaries and white point, they just have different gamma values (~2.2, ~1.95 and 1 respectively). It is therefore better to refer to Linear sRGB or Linear rec709. sRGB is intended for computer monitors in a indoor daytime setting and it is used widely on the web. Rec709 is intended for television viewing and assumes a dimly lit setting. Linear sRGB is not intended for viewing but as a working color space for the famous 'linear workflow'.

Comparing Color Spaces

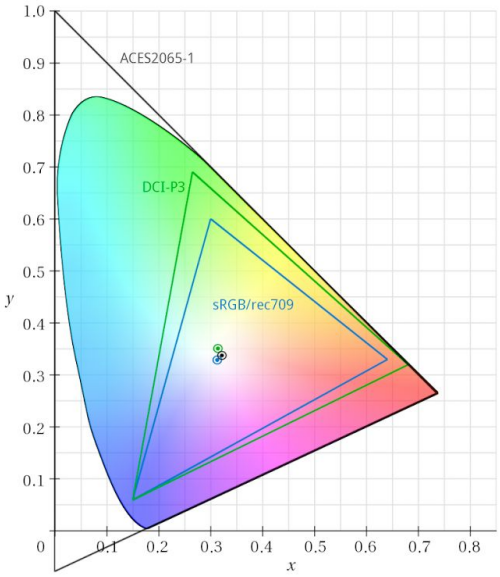


sRGB

DCI-P3

Next up: DCI-P3. You can see it has a much larger gamut, so it can display a much wider variety of colors. It has a white point of x = 0.314 and y = 0.351, which is referred to as 6300K, though we've seen it is much greener than that. It has a gamma of 2.6 and is the native color space of many professional projectors, often used in grading for cinema.

Comparing Color Spaces

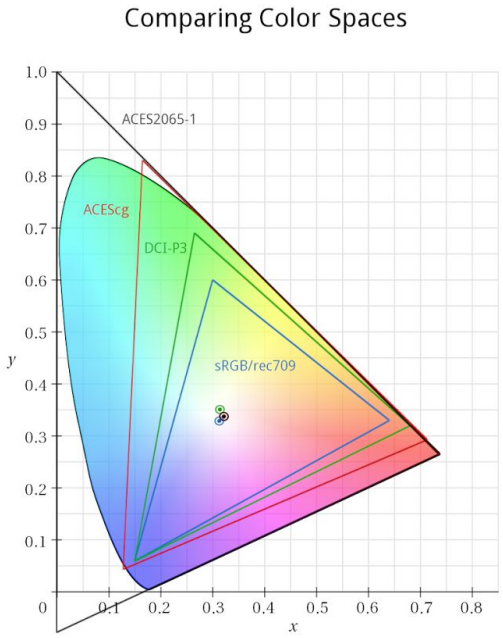


sRGB

DCI-P3

ACES2065-1

Introducing ACES2065-1! The color space that inspired this document. It has a huge gamut, with primaries that lie outside of the visible spectrum. This was done so that it includes the entire visible spectrum. It is intended for the storage of image data. The Aces color space has a linear gamma and uses x = 0.32168 and y = 0.33767 as the white point, which is the CIE D60 illuminant. This white point was chosen, because in the setting of a dark cinema D65 was found to appear too blue. ACES2065-1 is also referred to as AP0, or “ACES Primaries 0”. Whenever I refer to ACES as a color space this is the one I’m talking about.

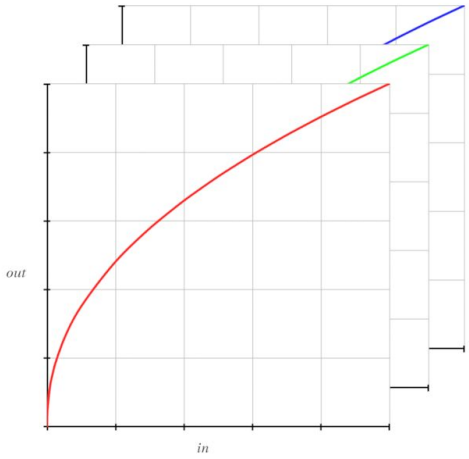


And finally we see ACEScg. This is the Aces color space that is intended to be a working space for compositing and cg. It features a large gamut (it encompasses rec2020, the new UHD standard color space) and also has a linear gamma. It uses the same D60 white point as ACES. In some software ACEScg is also referred to as AP1, which stands for “ACES Primaries 1”. There is another ACES color space that uses these primaries, called ACEScc. ACEScc has a log gamma, but is otherwise identical to ACEScg. It is intended as a working space for grading applications.

LUTs

Now that we have seen all these color spaces let's look at a practical issue: how to convert image data from one color space to another. The way software does this is through LUT files or LUTs. LUT stands for Look Up Table and that's exactly what it is. The computer uses a table to look up the appropriate output value that corresponds to an input value. Most LUTs are plain text files which you can open in any text editor and read yourself. The two types of LUTs that are commonly used are 1D LUTs and 3D LUTs. I'll tell you a bit about both, but really all you as an artist need to know is that they're files and you need to load the right ones in your software.

1D LUTs



in		*out*
0.0	⟷	*0.00*
0.1	⟷	*0.35*
0.2	⟷	*0.48*
0.3	⟷	*0.58*
0.4	⟷	*0.66*
0.5	⟷	*0.72*
0.6	⟷	*0.79*
0.7	⟷	*0.85*
0.8	⟷	*0.90*
0.9	⟷	*0.95*
1.0	⟷	*1.0*

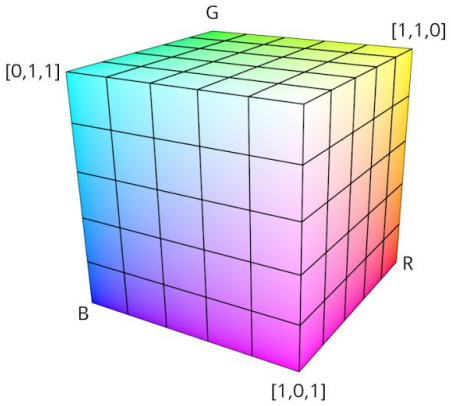
Properties:
- Typical resolution of 1024
- All channels treated equal
- Simple color changes
- Reversible

Application:
- Gamma conversion
- Log / Lin conversion

The simplest LUT is a ‘1D’ LUT. A 1D LUT simply maps an input value to an output value. It treats all channels equally, so 1 input value has 1 output value (hence 1D). A simple 1D lut for gamma conversion from linear gamma to a gamma of 2.2 could look like the table above.

This 1D LUT has 11 points, which is not very accurate. Usually a 1D LUT has 1024 entries, or a resolution of 1024. An important aspect of the 1D LUT is that it is *reversible*, it can be used both ways because you can just swap the two columns.

3D LUTs



in	*out*		
R,G,B	*R*	*G*	*B*
0,0,0 →	*0.01*	*0.02*	*0.01*
0,0,1 →	*0.01*	*0.03*	*0.98*
0,1,0 →	*0.01*	*0.98*	*0.01*
0,1,1 →	*0.01*	*0.97*	*0.96*
1,0,0 →	*0.99*	*0.02*	*0.01*
1,0,1 →	*0.99*	*0.03*	*0.98*
1,1,1 →	*0.99*	*0.97*	*0.98*

Properties:
- Typical resolution of 17-65
- Each value mapped
- Complex color changes
- Irreversible

Application:
- Colorspace conversion
- Applying grades

1D LUTs are too simple to convert between color spaces with different primaries or white points, which is why 3D LUTs were created. A 3D LUT is still a table, but it has separate values for R, G and B. This means that a 3D LUT can perform much more complex color changes. A simple 3D LUT looks like this table.

This 3D LUT has a resolution of 2: for each color the input values of 0 and 1 are mapped to new values. The computer will interpolate the rest. Of course a 3D LUT with a resolution 2 will not prove very useful. In practice 3D LUTs have resolutions between 17 and 65.

Note that 3D LUTs are not reversible. If you need to invert a 3D LUT you will need to find the inverse LUT file. It is possible to reverse engineer a 3D LUT to get an approximate inverse, but this is not very accurate and should only be used when you have no other option.

How to use a LUT

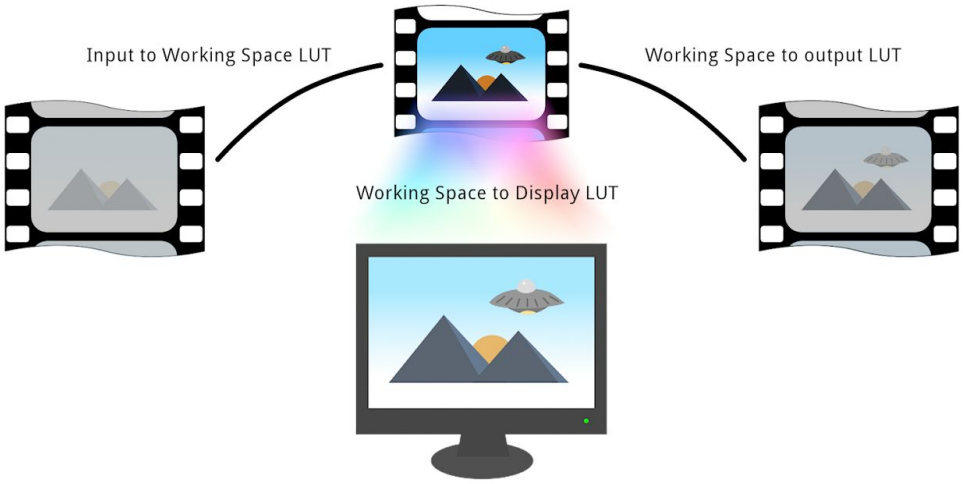
- LUT files
  (3dl, icc, cube, lut, cc, csp, hdl, etc.)
- Color space nodes
- Lossless conversion in 32 bit float
- Always 'from' and 'to' a color space

Now that we know what a LUT is, how do you use them? Luckily that's relatively simple: once you've figured out what your input color space is and you know to which color space you want to transform it, simply find the proper LUT file and apply it to your image in your software package.
Instead of applying a LUT file your software might have a color space conversion tool or plugin that allows you to select the proper in- and output color spaces. As long as the color spaces you need are available feel free to use those in stead.

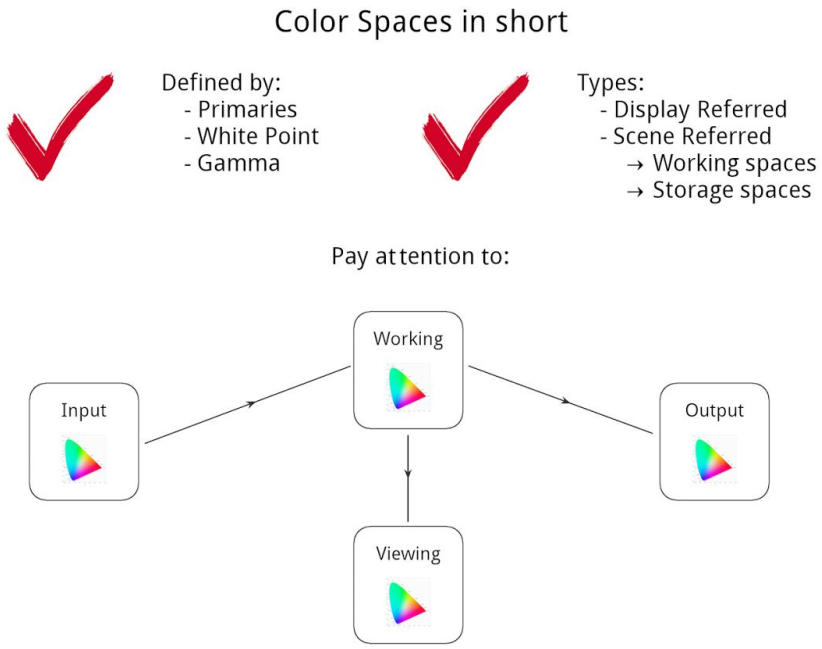
What is very important about color space conversions is that they are essentially lossless as long as you are performing them in a floating point bit depth. You can go from ACEScg to sRGB and back without loss of data in 32bit float, but be very careful when you render images to anything other than floating point EXRs! Floating point allows for negative values when you're converting to a smaller gamut, but even 12bit dpx files, which can hold a significant dynamic range, will clip those negative values.

Sometimes you run into a LUT that doesn’t specify both the input and the output colorspace. Remember that an image is always in a color space. So there's no such thing as a LUT with one color space. For example, sometimes people will talk about a rec709 LUT, or a P3 LUT. This is wrong and confusing. A LUT always converts from one color space to another. If a LUT does not mention two color spaces it often means the ‘from’ color space is implicit and it will most likely be a standard color space like sRGB. ICC profiles tend to be like this (be careful Adobe people!). Just be aware of this and test it out if you need to use a LUT like this.

View LUTs



So how are you supposed to work in one color space but view your work in another? This is where the View LUT comes in. Basically it goes like this: you get your material, probably in a storage space like AlexaLogC, import it in your compositing package and convert it to a working space like Linear sRGB. Now you can safely work on it, but you can't judge what you're seeing because it's way to dark and contrasty. So now you tell your software to load a View LUT: Linear sRGB to sRGB (when you're viewing your work on an lcd monitor). This converts the image to sRGB before it is sent to the monitor. It doesn't influence the compositing, it doesn't get rendered to a file, it only transforms your image so it looks good on your monitor. This is really the only way to work with images in a consistent way.



Well, that ended up being quite technical! We're now at the end of the color spaces part, but let me sum up the essentials for you.

Color spaces are defined through primaries (gamut) and white point in CIE-xyY coordinates, relating them to the gamut of human vision, and they have a gamma definition, describing the relation between brightness values.

Color spaces come in two flavours, Display Referred and Scene Referred, and you can split Scene Referred color spaces into Working Spaces and Storage Spaces.

We've looked at LUT files and we've seen how to use a View LUT to set up a proper linear workflow.

What you, as an artist, need to keep track of:

- What's the input color space?
- What color space am I looking at?
- What color space am I working in?
- What's the output color space?
- Do I have the LUTs to convert between them?

So, if you keep all of this in mind, not much can go wrong, right? So why ACES then? Well, there is one major problem with color spaces and images: If you don't know the color space of an image you can only tell which color space it's in if it looks as intended. And there's the catch: 'as intended'.

Because camera manufacturers like to decide how their footage was intended. So when they convert your footage to something you know, like rec709, for editing for example, they make it look ‘nice’. After all, someone paid big money for this camera, so it shouldn't look like crap right? And then the director and the editor get used to this look and then you deliver the vfx shots and they're totally different, because you worked of of the source and tried to keep everything as close to the source as possible. And you don't know which LUT was used to render the offline material, or worse, the LUT was only available in some odd software that you don't have access to. And they shot this thing on two cameras so things aren’t matching up anyway. Then, when they're going into the grading suite, everything looks totally different again, because, hey, they’re grading in P3 and the projector’s on spec, but there's no way to recreate the look that was used for the edit. And of course this is when the grader decides he’s more comfortable grading in redcolor3 or whatever so now all the vfx need to be rerendered…

Finally!

Enter ACES. ACES was originally created as a future proof way to archive digital footage, but along the way it also became a solution to specifically address the problems with color spaces in post production. To see that you must realise ACES is not just one color space, it's a system. It is a system of color spaces, mostly, though it also includes file format definitions, with definitions for metadata as well. I'll focus on the color spaces here, but suffice to say that ACES should be stored in 16 bit float OpenEXR files. Apart from archiving, ACES aims to standardize the color workflow.

This graphic represents the simplified ACES process. As you can see the Academy really likes three letter acronyms, so I'll walk you through it.

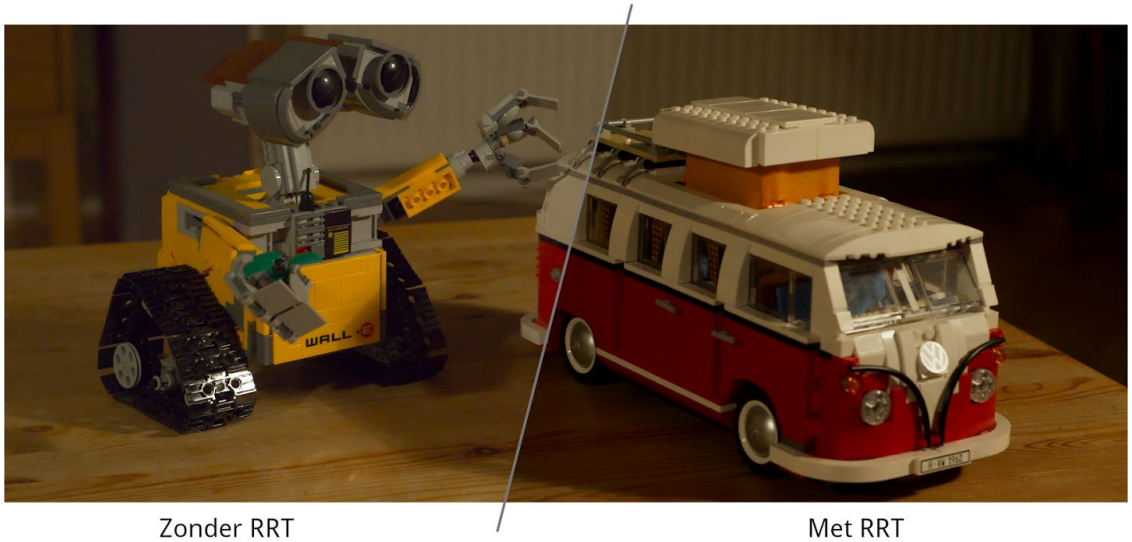
First we have a scene, can be anything, really, that's shot with a camera. The data from the camera is then converted to ACES using the IDT: the Input Device Transform. Basically that a LUT transforming the data from camera color space to the ACES storage color space. A lot of the magic happens in the IDT and I'll explain in a moment. So now we have the rushes in ACES, ready for archiving and distributing to the various departments. The data can be exported to any output device using the RRT and ODT. The ODT is the Output Device Transform, another LUT to convert the data for output devices like monitors and projectors. But before doing this technical conversion the RRT is applied. The Reference Rendering Transform is another LUT, but this one's not technical per se. Essentially the RRT applies a 'look' to the footage, but it's a carefully crafted look. I'll dig into that as well, but first let's look at the IDT.

IDT

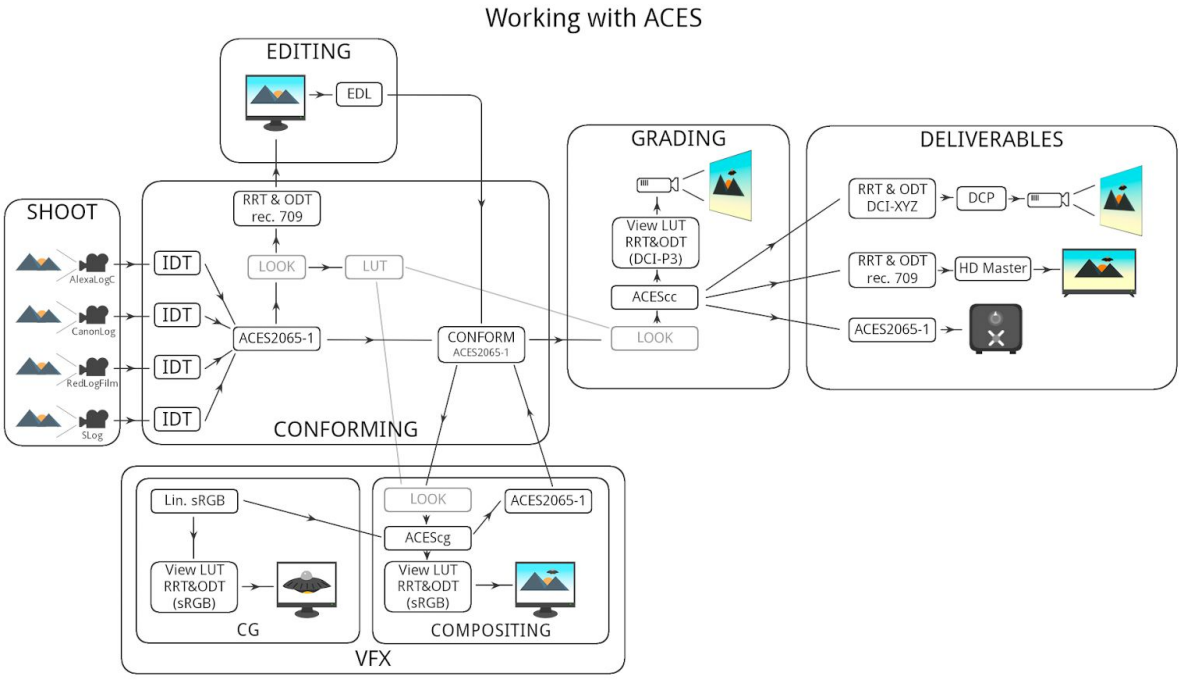
via Dado Valentic / www.mytherapy.tv

What makes the IDT special is that they are created by the camera manufacturers, but according to specifications designed by the academy. It's like the Academy set up a scene with a color card, had all the manufacturers shoot it and the told them each to make a LUT for their camera that transforms all the patches from the color card to predefined values in the ACES color space. That way the manufacturers don’t have to share their secrets with the world, and we get images from different cameras in roughly the same space. Of course differences remain between cameras as they are built differently, but this takes the guesswork out of it. You can now start to see how ACES standardizes workflows; the IDT is the first piece of the puzzle. As a VFX Artist you really shouldn’t have a lot to do with this, but it’s important to know what's going on, so you can properly decide how to work with the data you're given.

RRT



Now the RRT. As I said this is a ‘look’ that is applied before transforming to the output device. The look aims to simulate, to some extent, the effect of transferring to print film stock. What the Academy did was to interview many film professionals, DOPs, directors, etc., about what they would *expect* an (ungraded) image to look like and what they would *want* an image to look like. All of that feedback was turned into a sort of ‘generic look’ that is the RRT. Though this might sound a bit arbitrary, this is the other piece to the puzzle of standardization, because the RRT makes sure that what you see on your sRGB monitor looks about the same as what you see on the P3 projector.



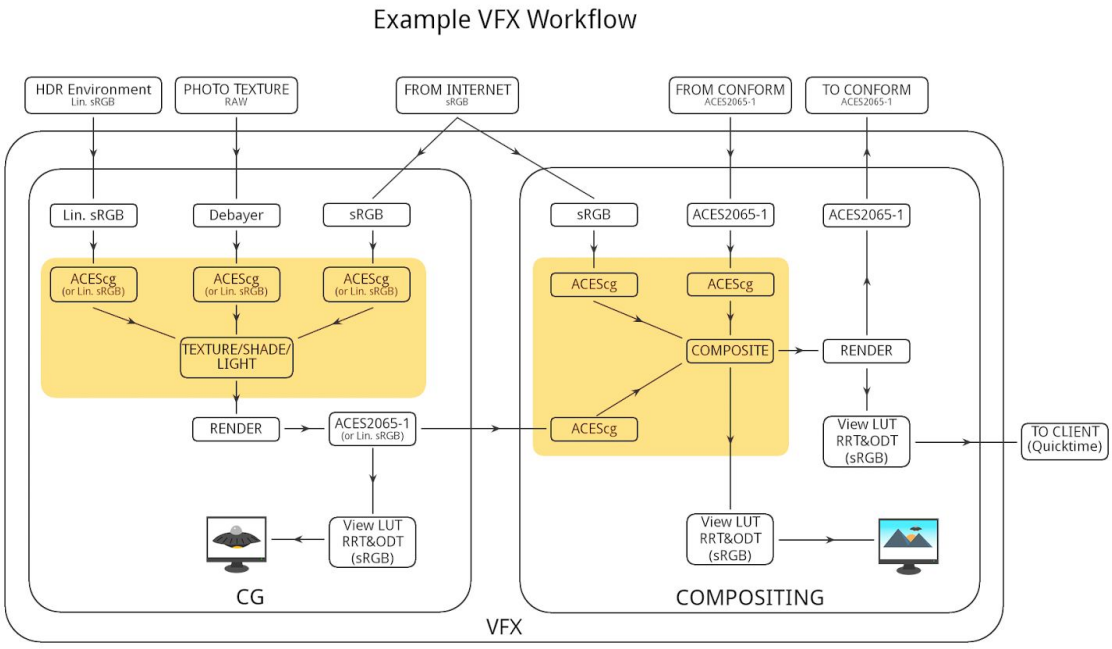
Here I've expanded the ACES graphic to explain a bit more about the way the workflow is applied in practice. We start with the shoot, so we have the scene again, but now shot with different cameras. Or maybe it's multiple scenes, or green screens and plates, you name it. Each camera converts the light in the scene to image data in a Scene Referred color space, a Storage Space. Now ideally all the rushes would get converted to ACES using the IDTs and stored for archiving, but in the real world most likely the data is stored in its original storage space.

So, next up is the editing department. The rushes get converted to ACES as an intermediate step and perhaps a look is applied. The rushes are then rendered to a Display Referred space suitable for offline editing, and the look is saved in a LUT file. What's important is that this look is saved without the RRT/ODT embedded, so anyone else can apply the look as well!

After the edit is done the data that made the cut is conformed. From the conforming station visual effects shots are exported in ACES. The VFX department imports the shots and converts them to ACEScg internally to work on. Because all the data was converted to ACES using the correct IDTs the amount of work lost in trying to match different cameras is minimised. Any looks that were applied are sent to VFX, in the form of the LUT that was saved before. Any CG elements are rendered out and saved in ACES or ACEScg if possible, or otherwise in Linear sRGB, and the CG is composited. As you can see both CG artists and compositors view their work through an RRT/ODT, sRGB (D60 sim.) assuming they are working on sRGB monitors. It is very important the same View LUTs are used, otherwise the CG will look completely different to the compositor than it did to the CG artist. Any previews that are needed in the edit can be rendered out with the look LUT applied. And finally completed shots are rendered out to ACES again and are sent back to the conforming station.

In the meantime the grading department will probably have started grading. It's likely the look that was used for editing will be a starting point for the grade. Graders should be working in a scene referred color space like ACEScc and will be looking through the appropriate RRT/ODT View LUT, like DCI-P3 or, if the projector is on target, just the RRT. Because VFX was using the same RRT with the ODT for their display the difference between what the artist saw and what the grader sees should be minimal. And again, any mismatch between different cameras should lead to a minimum of extra work thanks to the IDTs. When the grading is done the graded master should be rendered out to ACES so it can be used to make DCPs, TV versions and even film prints.

Of course, keeping track of the 'color space journey' requires a lot of attention and it's easy to get sidetracked with all the different departments involved, but I hope you can see why using ACES in your workflow can eliminate a lot of confusion and frustration!



Here I've made a close up of the VFX part of the previous flowchart.

Important things to note:

Compositing
- Aces2065-1 sources from conform
- AcesCG working space
- Convert all input to working space
- sRGB RRT/ODT viewlut
- Aces2065-1 output

3D CGI
- ACEScg working space if available, otherwise Linear sRGB
- Convert Textures to working space
- sRGB RRT/ODT viewlut
- Aces2065-1 output

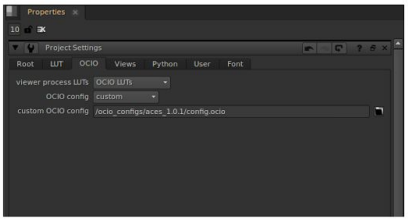
In de Praktijk

OpencolorIO (OCIO)
- Open Source Color Management
- Volledige ACES support
- Steeds meer ondersteunde software, waaronder:
  + Nuke
  + Fusion
  + After Effects (middels plugin)
  + Maya 2016
  + Modo
  + Blender
  + ...

- Python ondersteuning

- Laad de ACES OCIO Config file in de software
- Krijg toegang tot alle LUTs
- Verschillende programma's, zelfde LUTs

http://www.opencolorio.org
https://github.com/hpd/OpenColorIO-Configs (ACES config)



All this talk about ACES is nice of course, but how are you supposed to use all this? The answer comes in the form of OpenColorIO. OpenColorIO is an open source color management system developed by Sony Pictures Imageworks for their pipeline and it's being implemented in more and more software. You can even install it to manage your color on the OS level. Most major Visual Effects packages now support OpenColorIO, or OCIO, including Nuke, Fusion, Maya, Vray and more. For After Effects there's a free plugin that allows you to to work with OCIO.

The way it works is through OCIO config files. Software packages will often include one or more OCIO config files, but in my experience the one we need is not supplied with the software. The latest ACES config is available from https://github.com/hpd/OpenColorIO-Configs. This is the official location for the ACES OCIO config. Once you've downloaded it you simply point your software to use the custom config and you'll have all the IDTs, RRT/ODTs and ACES color spaces available to you, allowing you to convert images back and forth.

If you are using software that is not yet able to use OpenColorIO directly this doesn't mean you cannot work in ACES! As long as your software supports a common LUT format you can create the LUTs you need directly from the config. For this you'll need to install OpenColorIO on your system. When you do you'll get a command line tool called 'ociobakelut' that allows you to generate the LUT files from the config. How this works exactly is beyond the scope of this document, but head to http://www.opencolorio.org and you'll figure it out.

THANKS!

Willem Zwarthoed

willem@filmmore.nl

And so we've reached the end of this color space adventure. I hope I've been able to help you better grasp the concept of color spaces and that you are curious and confident enough to give ACES a try. I also hope you will be able to help spread better knowledge of color spaces. There are a lot of misconceptions about the subject and if we can educate each other we can put more energy in making Visual Effects instead of solving color space issues. Of course the workflow I've presented here is a suggestion and though it ensures proper color management it is by no means the only way to work. Whatever you're doing, if it works for you and you and your clients are happy with your end result - by all means continue. But it's always better to break the rules when you know the rules.

Thank you for reading,

Willem Zwarthoed

Februari 2016

Appendix A - Enabling the ACES config in Nuke and After Effects

First download the ACES OCIO config from https://github.com/hpd/OpenColorIO-Configs and extract the *aces_1.0.1* folder.

Nuke Instructions:

Open your project settings [press s] and click the OCIO tab. Set the following:

viewer process LUTs	→	*OCIO LUTs*
OCIO config	→	*custom*
custom OCIO config	→	*/your/path/to/aces_1.0.1/config.ocio*

To set all your new projects to use ACES by default, open the preferences and navigate to *Project Defaults > Color Management*. Under *OpenColorIO config* select *custom* from the dropdown and enter the path or browse to the aces_1.0.1/config.ocio file.

You will see your ViewLUTs have changed and you can now use the OCIO_Colorspace node to do color space conversions. Remember you will have to set your Read and Write nodes to 'raw' to disable Nuke's internal color space conversions. Use OCIO nodes to convert correctly.

Note: Using a dedicated color space conversion node in Nuke is a good idea anyway as the color space conversions in Read and Write nodes are gamma only - primaries and white point are not adjusted in Read and Write nodes!

After Effects Instructions:

Download and install the After Effects OCIO plugin from http://fnordware.blogspot.nl/2012/05/opencolorio-for-after-effects.html. Open After Effects and set your settings as follows:

Project Settings	>	*Color Settings*	>	*Depth*	>	*32 bits per channel (floating point)*		
				Working Space	>	*None*		
Interpret Footage	>	*Main*	>	*Color Management*	>	*Preserve RGB*	>	*on*
Render	>	*Output Module*	>	*Color Management*	>	*Preserve RGB*	>	*on*

In order to work with ACES put your footage in a comp and add the OCIO plugin effect. In the effect settings browse to aces_1.0.1/config.ocio to load the luts. You can now set your input color space to match the footage; set the output color space to your working space, ACEScg.
Add an adjustment layer as the topmost layer and add the OCIO plugin. Set the input color space to match the working space set in the footage layer (ACEScg) and the output color space to match your output (for example AlexaLogC). This layer needs to be on top of any compositing operations.
To enable a ViewLUT add another adjustment layer on top of the last one and add the OCIO plugin again. Set the input to the output space (AlexaLogC in this example) and the output to match your display device. *Remember to disable this layer before your final render!* Tip: you can also make this layer a guide layer.

Appendix B - Some ACES OCIO LUTs Explained

ACES - ACES proxy

Log color space that uses ACES PRIMARIES 1 and the D60 whitepoint. Designed for on set color management and transmission over HD-SDI.

Output - sRGB (D60 sim.)

RRT+ sRGB ODT, shifted to 6000K to match typical cinema environment.

All the color spaces in the ‘Output’ category have the RRT applied.

ADX

Academy standard for film scans. Alternative to Cineon, although it doesn’t match exactly.

Utility LUTs

Utility - Rec709 - Camera:	rec709 color space with a gamma of ~1.95	(in Nuke: rec709)
Utility - Linear - sRGB:	Standard Linear Light working space	(in Nuke: Linear)
Utility - sRGB - Texture:	Standard sRGB color space	(in Nuke: sRGB)
Utility - RAW:	No color space conversion / disable ViewLUT	(in Nuke: None)

Appendix C - References and Further Reading

Academy Aces:	http://www.oscars.org/science-technology/sci-tech-projects/aces
OpenColorIO:	http://opencolorio.org/
ACES OCIO config:	https://github.com/hpd/OpenColorIO-Configs
After Effects OCIO plugin:	http://fnordware.blogspot.nl/2012/05/opencolorio-for-after-effects.html
fxguide, The Art of Digital Color:	https://www.fxguide.com/featured/the-art-of-digital-color/
VES, Cinematic Color:	http://cinematiccolor.com/
Haarm Pieter Duiker:	http://duikerresearch.com/ http://www.slideshare.net/hpduiker/
Alex Fry - ACES on LEGO movie:	https://www.youtube.com/watch?v=vKtF2S7WEv0
Visualizing the XYZ color space:	https://youtu.be/x0-qoXOCOow
Programmer's Guide to XYZ:	http://ninedegreesbelow.com/photography/xyz-rgb.html
LUTs - Light Illusion:	http://www.lightillusion.com/luts.html
Gamma:	http://www.cambridgeincolour.com/tutorials/gamma-correction.htm http://www.poynton.com/notes/colour_and_gamma/GammaFAQ.html
Log - prolost, Raw is not Magic:	http://prolost.com/blog/rawvslog
Understanding RedLogFilm:	http://www.red.com/learn/red-101/redlogfilm-redgamma

Wikipedia Articles

- ACES	https://en.wikipedia.org/wiki/Academy_Color_Encoding_System
- CIE 1931	https://en.wikipedia.org/wiki/CIE_1931_color_space
- sRGB	https://en.wikipedia.org/wiki/SRGB
- rec709	https://en.wikipedia.org/wiki/Rec._709
- Gamma	https://en.wikipedia.org/wiki/Gamma_correction