# Nuke's Color Management Demystified

## *by Steve Wright*

I spent eight years at Kodak's Cinesite in Hollywood first as a Senior Compositor then as a 2D Technical Director watching the engineering department struggle with a very fundamental question - how do you composite linear CGI images with cineon log film scans? These were some very smart guys, but they could never get it quite right. Then, one day, I opened up Nuke and saw its color management system. The dark clouds parted, a choir sang "hallelujah," and golden rays of sunlight shone down upon my face. Nuke got it right! Nuke's three secret ingredients to getting it right are floating point data, high dynamic range images, and linear light space. In this article, we will see why this is so and how it all works.

## *Floating Point*


The need for floating point (or "float") is simply because using 8 bit integer data with its paltry 256 discreet code values, or even 16 bits with its somewhat more robust 65,535 code values quickly run out of precision and start degrading the quality of the images during processing. When you load any image into Nuke, regardless of the image type, it is automatically promoted to 32 bit float - no options. The amazing thing about Nuke's 32 bit float implementation is its blazing speed. Somehow it does not suffer the usual speed penalty for working in float.

## *HDR Images*

The second reason for working in float is that HDR (High Dynamic Range) images can have code values that go far above 1.0, and float is the best way to represent them. Nuke was designed from the ground up to create the highest possible quality feature film visual effects, which meant that they were going to be compositing a lot of 10 bit Cineon and DPX log film scans. What may come as a surprise to many is that these log files are in fact HDR images. When converted to linear they produce float code values up to 13.5. OpenEXR files can go even higher.

## *Linear Light Space*

Nuke also converts every image to "linear light space." Most image types come with baked-in colorspace corrections that make them non-linear. If an image is truly linear, it means that any change in the code value results in an equal change in the image brightness. In a linear system, if you double a pixel's code value it will double the brightness it represents. This is not true of non-linear images.

*I used the term "brightness" here, but that is not technically correct. The correct term is "intensity," the amount and spectrum of light energy that is reflected by a scene. Brightness is actually the human perceptual response to intensity, and is decidedly not linear. Our visual perception is so non-linear that doubling the intensity of a scene only increases the perceived brightness by 18%. However, I am going to continue to use the term "brightness" even though it will offend color scientists because it is intuitively meaningful to digital artists.*

There are two very big reasons for converting everything to linear light space. The first is that the real world works in linear light space. And since CGI is a simulation of the real world, it uses it also for its render calculations. Inside Maya, Mental Ray, or RenderMan all of the computations for ambient and diffuse lighting, reflections, specular highlights, global illumination models, etc. are all calculated in linear light space. Nuke follows this paradigm to not only maintain consistency with CGI but also for proper image processing math. The second reason is to allow multiple image types to be combined in compositing, regardless of origin. Everything is converted to linear, so everybody plays nice together.

The problem with CGI images is that they do not stay in linear light space when they are written out to disk. They are given a large gamma correction in order to look right on a monitor. Worse, this gamma correction is baked into the image data, so it becomes permanently disfigured.

## Gamma Corrected Images

The reason a gamma correction is baked into every image is because of the history of CRT monitors. Due to their physical nature, CRT monitors (which includes TV sets) darken an image - by quite a bit. We have to pre-correct the image data if we want it to look right on a monitor, and that is done with a gamma correction. This gamma correction pre-brightens the image data by the same amount that the monitor darkens it. The result is a natural looking image on the screen, but unnatural data describing it.

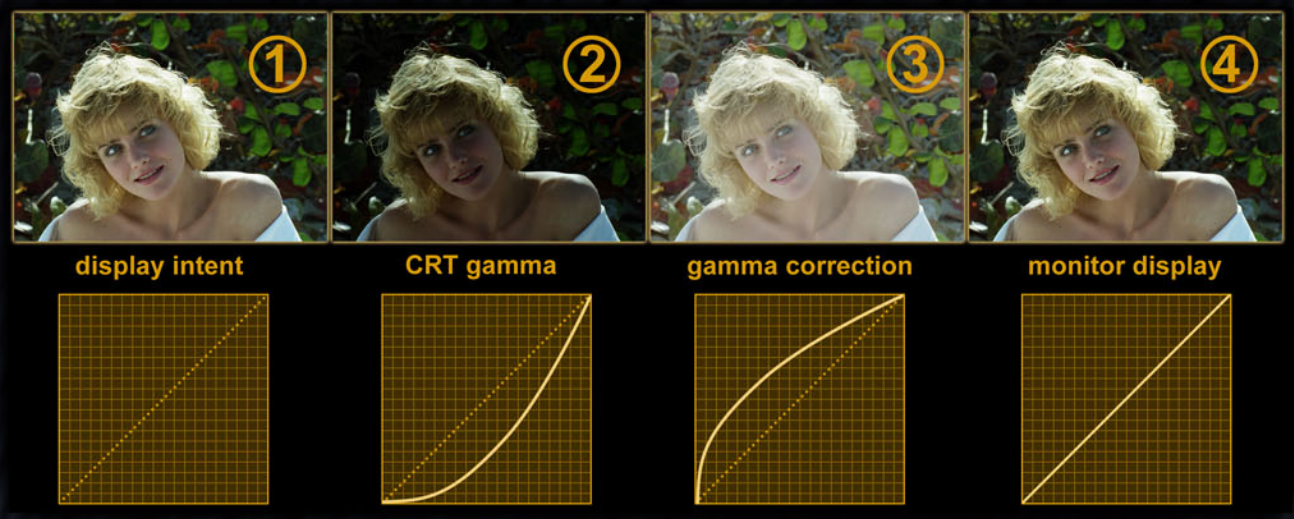



*Figure 1 - why gamma correction is baked into an image*

The sequence of events is illustrated in Figure 1. Image (1) display intent shows how the picture is "supposed" to look. Image (2) CRT gamma shows how the gamma of a CRT seriously darkens it, depicted by the gamma curve below. Image (3) gamma correction shows how the image is dramatically pre-brightened using the gamma correction curve below. Finally, image (4) monitor display shows that the gamma correction cancels out the CRT gamma to display the image correctly.

Modern flat panel displays do not have the same physics as CRT monitors so they don't have the built-in darkening gamma. However, to avoid having to make multiple versions of images in order to look right on all possible displays, it was universally agreed that flat panel displays will have an artificial gamma LUT built into them so that they replicate the appearance of CRT monitors.

## Nuke's Color Management Workflow

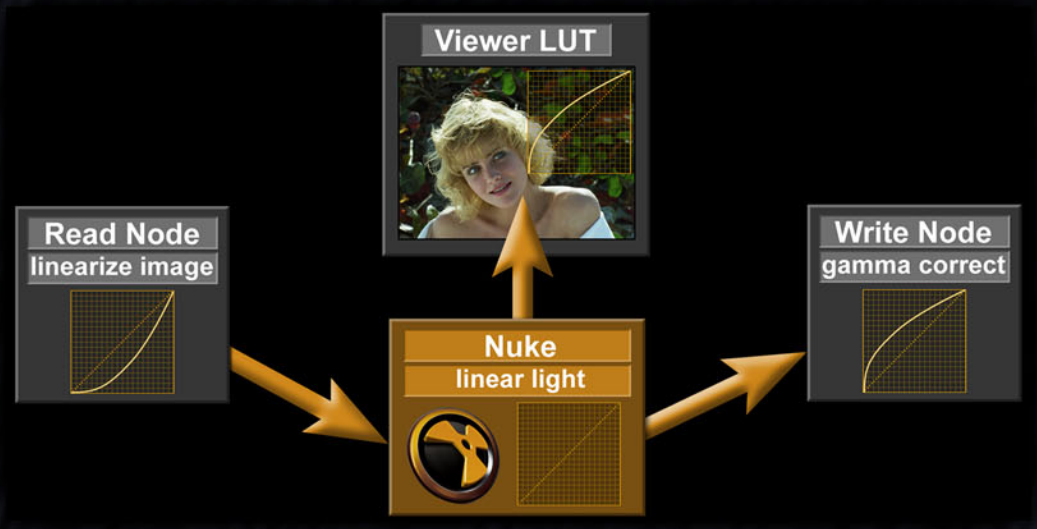



*Figure 2 - Nuke's color management workflow*

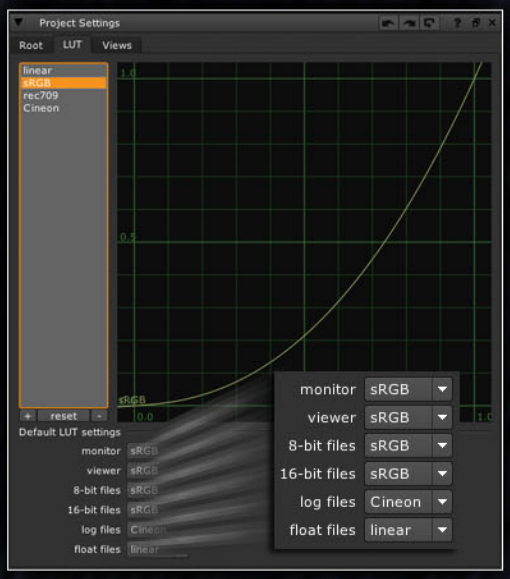



*Figure 3 - Project Settings LUTs*

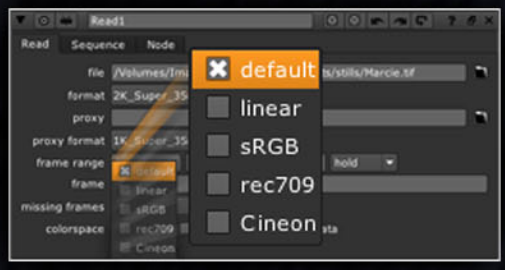
Now that we've seen why most images are non-linear, let's take a look at Nuke's color management workflow, diagrammed in Figure 2. When an image is read into Nuke the Read node immediately selects a LUT to linearize the image. For some file formats all that is needed is to back out a baked in gamma correction, but log images need to be converted from log to linear. All internal computations on the image are then done in linear light space. If the raw linear data were displayed in the viewer it would look way too dark, like image (2) in Figure 1. So Nuke's Viewer LUT gamma corrects the linear image data so that it looks right in the viewer. Keep in mind this is only applied in the viewer and not to the original image, which stays linear. Finally, when an image is rendered out to disk, the Write node applies a gamma correction LUT so the saved image will display correctly on a monitor.

The master repository of Nuke's LUTs is the LUT tab in Project Settings. This is where all of Nuke's LUTs are defined and where you can create your own custom LUTs. This is also the master control where Nuke looks to see which LUT to use under what circumstances. As you can see in Figure 3, the Default LUT choice for the viewer is sRGB and for log files Nuke will use the Cineon LUT. You may also change the default LUT that Nuke chooses for the various file types by selecting a new one from the popup menu.

There is one idiosyncrasy about this list that you should be aware of. Most of the LUTs listed there are designed to convert an image from its current colorspace to linear. The exception is the LUT named linear. It does not convert anything to linear. This LUT does no conversion at all, and is what some call an identity LUT or unity LUT, because what goes in comes out unchanged. This is specifically designed for data channel files such as motion vectors, depth, and especially alpha channels, as they do not represent light and need to be left unmolested.

## The Read Node

As we saw in Figure 2, the Read node is the first checkpoint in Nuke's color management where the incoming image is converted from whatever colorspace it is in to linear. The question is, how does Nuke know which LUT to use?




Read node LUTs

Remember the Project Settings Default LUT list in Figure 3? It shows, for example, that for an 8-bit file Nuke will use the sRGB LUT. So let's say you read in an 8 bit tiff file and the Read node is set for default. That means Nuke will go to the Default LUT list in Project Settings and look up "8-bit files"and use what is set there - sRGB in this case. If you don't want to use the default LUT then you can use the colorspace popup menu to force a LUT of your choice.

One very important point about the Read node and its colorspace conversions is the premultiplied toggle right next to the colorspace LUT. This must be turned ON if the input file is a 4 channel premultiplied image (CGI). This tells Nuke to unpremultiply the image before applying the colorspace LUT. If it is not turned on the output CGI image will composite with darkened edges.

## Viewer LUTs

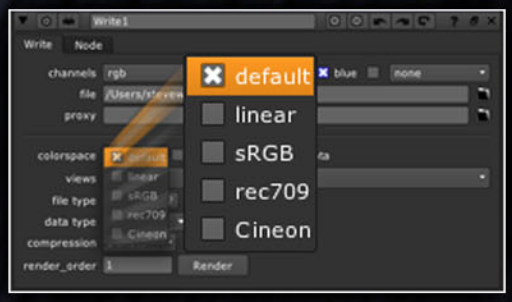
As we saw above, the raw linear image data gets a LUT at the viewer to correct it to look right. All of the LUTs defined in the Project Settings show up here, including any custom LUTs, and are automatically inverted for use by the viewer. Very nice!


Viewer node LUTs

The default Viewer LUT is sRGB because that is the normal colorspace of workstation monitors, CRT or flat panel. If you had a broadcast monitor hooked up to your workstation you would use the rec709 LUT instead. If you ever want to see what uncorrected linear data looks like, just set the Viewer LUT to linear. Be sure to put it back when you are done, however.

# The Write Node

The third and last step in Nuke's color management occurs at the Write node when the image is written to disk. Just like CGI renders, Nuke applies a gamma correction when writing the image out to disk. And for the same reason - to convert the linear data for proper display on a monitor or TV.




Write node LUTs

Similar to the Read node, when set to **default** the Write node automatically selects which colorspace LUT to use based on the type of file you write to disk. If you render an 8 or 16 bit tiff it will look it up in the Project Settings and select **sRGB**. For an OpenEXR imageNuke will use the linear LUT. If you render a **Cineon** file it will use the Cineon LUT to convert it to log colorspace for the film recorder. If your images are destined for television then the **rec709** LUT is for you.

These LUTs are also automatically generated from the LUTs in the Project Settings, so any custom LUTs show up in the Write node as well. Again, if you want a LUT other than what Nuke selects by default you can manually choose one from the colorspace popup menu. You will also find a **premultiplied** toggle here just like in the Read node. You will want this **ON** if you are rendering a four channel premultiplied image to disk that will be composited later.

# Conclusion

So there you have it. The complete story of Nuke's marvelous color management. Regardless of what colorspace the incoming image is in, it is converted to linear for proper processing. It now does not matter whether the original images were CGI, log, or whatever. They are now all perfectly compatible which allows them to be combined in the mathematically correct way. At the end of the compositing tree the final composited linear image is converted to whatever destination colorspace is desired in the Write node when it is saved to disk.

In defense of the frustrated engineers at Cinesite, it is now clear why they did not get it right. Indeed, why they could not get it right back then. As we have seen, to get it right you must be working in float, and in those days float was hopelessly slow, so 16 bit integer was the best we could do. And it was never going to work in 16 bit integer. We were doomed.

# Bio

Steve Wright is a 20 year visual effects compositing veteran with over 60 feature film credits. He has published two popular books on compositing (links below) and is now a trainer for vfx compositing, Shake and Nuke. He travels around the world conducting staff training at VFX studios and conducts location-based and online workshops.

visit Steve's training website at
www.vfxIO.com

Steve's film credits on IMDb

Digital Compositing for Film and Video

Compositing Visual Effects: Essentials for the Aspiring Artist