Be gamma correct!

© 2009 Martin Breidt (http://scripts.breidt.net)
v 1.3 - 17-Jun-09

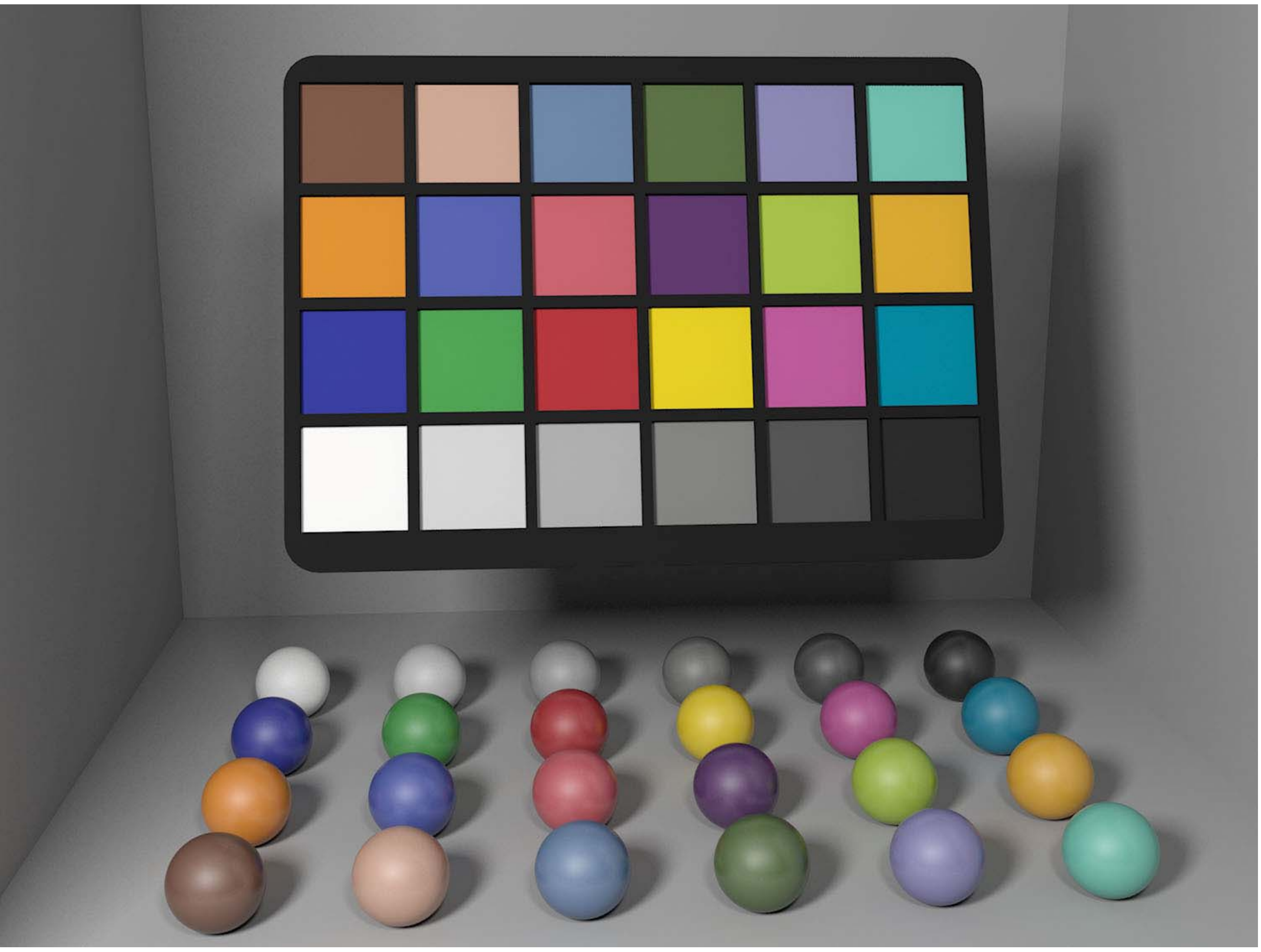


Figure 1 – A 3D simulation of the well known Gretag Macbeth color checker with accurate rendering of color and brightness thanks to gamma correction

The terms "gamma" and "gamma correction" are well-known to most graphics practitioners. They describe a fundamentally simple but sometimes confusing method used in print, imaging and video technology, which also plays an important role in 3D computer graphics and compositing. Even though most have heard the terms before, there are many misconceptions about gamma correction, especially in the field of computer graphics (for example: "By calibrating my monitor, I remove its gamma curve"). This article, intended for any slightly technically inclined 3D practitioner, tries to clear things up by giving some background on what gamma correction is, why it is needed, and what happens if you ignore it. Additionally, the article explains why it is important to use gamma correction in any image processing and how to set up proper gamma correction in 3ds Max as an example.

01. Basics

The main goal of gamma correction is to reproduce brightness levels in a picture shown on an output device in the same way as they were originally observed (e.g., by a camera, scanner, or also by the user on the screen).

The notion of a "gamma curve" was originally introduced in video technology to characterize the non-linear relation between the video signal voltage and the resulting brightness on a cathode ray tube monitor (CRT), caused by the properties of the electron gun used in such a monitor.

Let I be the brightness and U be the image signal on a normalized scale from 0-1. Their relationship on a CRT can be described by the power function $I = kU^{\gamma}$ with k being a constant describing the maximum possible brightness (while neglecting the minimum residual brightness, also called "black level"). The exponent γ used here is called the "gamma" of the monitor and normally ranges between 2.3 and 2.6 for CRTs, but other parts of the graphics pipeline can also influence this value to produce a net effect of anything between 1.5 and 2.5 (Figure 2 left).

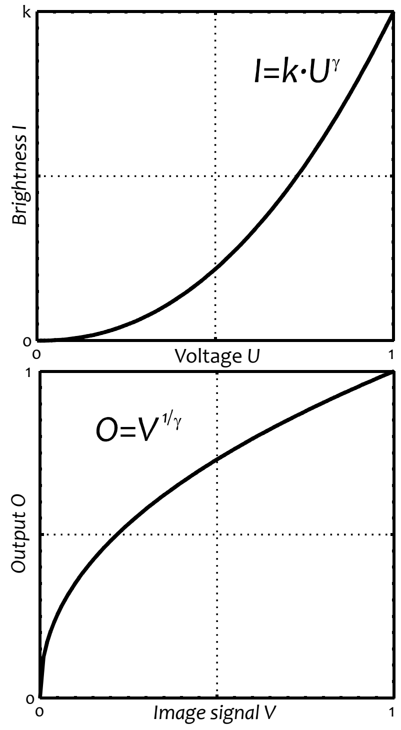


Figure 2 – Top: A typical gamma curve of a monitor (γ=2.2, Black Level ignored). Bottom: The matching gamma correction curve.

In order for the monitor to produce brightness levels identical to what the camera saw initially, the video signal has to be modified to compensate for the non-linear response curve of the monitor. Thus, the video signal is raised to the power of $1/\gamma$ (Figure 2 right) such that the two power functions eventually cancel each other out and produce a linear brightness reproduction. This intentional distortion of the image data to compensate for a non-linear display is called "gamma correction"[1].

[1] Note that while the monitor's ray gun behaves nonlinearly when controlled by different voltages, the light/photons emitted from the screen are of course linear (in fact there is no such thing as nonlinear light).

02. Non-linear perception

In order to produce accurate images for the human eye, one has to take into account the effects of human perception. While the precise details are remarkably complicated and depend a lot on the context, it turns out that humans perceive changes in physical size in many domains (e.g. brightness, volume, weight) differently depending on where you start. In other words: we perceive relative changes, not absolute ones. Roughly speaking this means that our perception often uses a logarithmic scale. Hence, the brightness of stars is measured on a logarithmic scale, as is audio volume (in decibel) or the density of photographic film.

If, for example, we hold a weight in our hand, then the smallest weight difference we can perceive depends on how big the original weight was. With a small weight we will notice when adding just a few grams; with a big weight, however, we do not perceive the addition of the same few grams at all. This principle is known as Weber-Fechner law and also applies to our brightness perception[2].

So, by lucky coincidence, the human eye reacts in a quite similar non-linear way to signals as a CRT does (and quite differently than a light meter). More on this later, but let's take a closer look first at human perception of brightness values.

03. Non-linear encoding

The human eye, when adapted to the current light situation, can normally perceive a dynamic range of about 100:1 at once and is able to recognize brightness differences of about 1%. This means, for example, that we will clearly see the difference between two 10W and 11W light bulbs, but we will not see the same difference between light bulbs of 150W and 151W power. This relative sensitivity is why it would be inefficient to save the brightness values of a digital image with the same linear accuracy across the entire dynamic range: Stepping from 1 to 100 with a constant step size of 0.01 will require 9900 steps and therefore a binary precision of $\log_2 9900 \approx 14$ bit. With that, there will be no visible banding in the dark regions anymore, but at the same time the bright areas would be unnecessarily detailed. But if you use a non-linear encoding based on a

[2] As mentioned before, the actual details of human brightness perception are a lot more complicated and can be affected by surround brightness, eye adaptation, image frequency etc. The simplification we make here is sufficient for practical purposes of observers under normal viewing conditions.

gamma curve, for example, you get higher precision in the dark regions and less precision in the bright regions while only using 9 bits in total. This is fairly close to the usual 24 bit RGB images that provide 8 bit per color channel. Therefore, all 8 bit images intended for human observation should be saved in a non-linear fashion – sometimes also called "gamma coded" – to avoid visible artifacts.

A quick comment on the actual pixel values: typically, 8 bit values in a computer are stored in a Byte, ranging from values 0 to 255, with 0 being the darkest black and 255 being the brightest white[3] that can be handled by the device. When looking at luminance in floating point precision, most people treat the same dynamic range (previously stored as 0…255) as values 0.0 to 1.0, and any value above 1.0 as "superbrights".

04. Not a bug, a feature

So why is it a lucky coincidence that a CRT and our eye both deal with the image signal in a very similar, non-linear way? Because a compact, gamma-coded 8-bit image will then be transformed back simply by displaying it on a monitor and thus produce correct, linear brightness values as originally observed by the camera without the need for any additional computations.

In addition, the gamma curve of the monitor will work nicely with the brightness perception curve of the human eye: a simple linear gradient between 0…255 without gamma correction will produce a downwards bent intensity profile on the monitor (when measured with a light meter) but this in turn is perceived as a linear increase in brightness by the eye, as intended. In other words: equal steps in framebuffer pixel values (or signal voltage) will produce equal steps in perceived brightness thanks to the monitor gamma curve.

So gamma is indeed a desired feature that allows minimizing noise and banding when using a reduced data bandwidth. It is not an artifact that needs to be removed but is an effect matching the human eye.

Directly displaying linear image data on a monitor that has a gamma curve will lead to noticeably darkened midtones. Accidentally applying gamma correction twice will result in midtones which are much too bright.

[3] In reality, color is coded as red, green and blue values, but in this article, we only discuss greyscale images for simplicity. The concepts can be directly transferred to color images, though.

Sometimes the gamma correction embedded in the image data and the gamma curve of the monitor do not cancel out each other completely, leading to an increase in image contrast - an effect that can be desired in order to prepare image material for viewing in dim surroundings, typical for TV or cinema viewers.

05. Flat screens

While the brightness response of CRTs can be well described by a gamma curve, the behavior of flat screens (LCD, TFT) is fundamentally different and often highly non-linear. Luckily, most flat screens contain special circuitry to simulate a gamma curve similar to CRTs. Therefore, for practical purposes a gamma correction can be performed for these monitor types, too.

06. The problem, part one

But let's return to our digital images that use 8 bits per color channel to encode red, green and blue intensities. These images are used by many on a daily basis for texturing 3D objects, as layers in Photoshop, or as medium to store 3D renderings. As described before, these images should be using a non-linear encoding in order to produce an optimal result for the human eye.

The problem arises when feeding this (non-linear) image data directly to algorithms that strictly assume that brightness is encoded linearly. This assumption is made in almost any piece of graphics software since linear illumination formulas are faster to develop and to compute. Unfortunately, it is often ignored that the image data is already stored on disk non-linearly since the image format only uses 8 bit per color channel.

07. The problem, part two

Even without using external image data, it is often forgotten when computing image values that the brightness values on screen do not depend linearly on the pure RGB pixel values, thanks to the screen gamma curve. Therefore, even the most simplistic addition or multiplication of RGB values can produce incorrect results on screen.

Here is a simple example to illustrate this: Let's assume that a white image (RGB values = 255/255/255) produces a maximum brightness of 1.0 units on screen. Now the image is to be darkened down by half. If you now simply half all pixel values and send them as 128/128/128 to the screen, the screen's gamma curve will render this into a effective brightness of $(128/255)^{2,2} = 0.22$ units! In reverse, this also means that a naïve doubling of RGB value 128/128/128 will increase the brightness from 0.22 to 1.0. Suddenly, our math is 0.22 * 2 = 1.0.

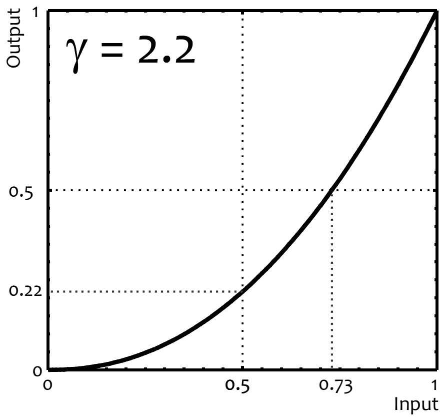


Figure 3 – Gamma curve for γ=2.2, closely matching the sRGB standard. An input value of 0.5 (pixel value=128) produces an output value (brightness) of only 0.22. Conversely, an output value of 0.5 requires an input value of 0.73 (=186)

In other words: The software treats image data as “50% brightness” which in reality only produces 22% brightness. The correct value for half maximum brightness would have been 186/186/186 (Figure 3).

This drastic mistake also occurs, if you do not correct for gamma and…

- …compute transparent objects,
- …compute anti-aliasing or motion blur,
- …mix RGB textures with physically correct lighting (Global Illumination),
- …add up the brightness contributions of multiple light sources,
- …scale and filter textures (e.g. for mip mapping),
- …blur images.

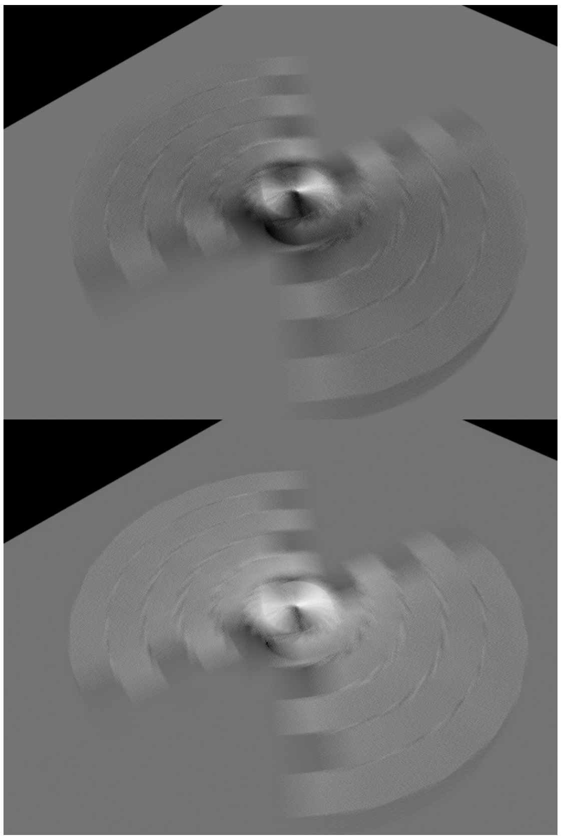


Figure 4 – Simple motion blur example using a black and white texture on a rotating box. Top: too dark without gamma correction. Bottom: With gamma correction.

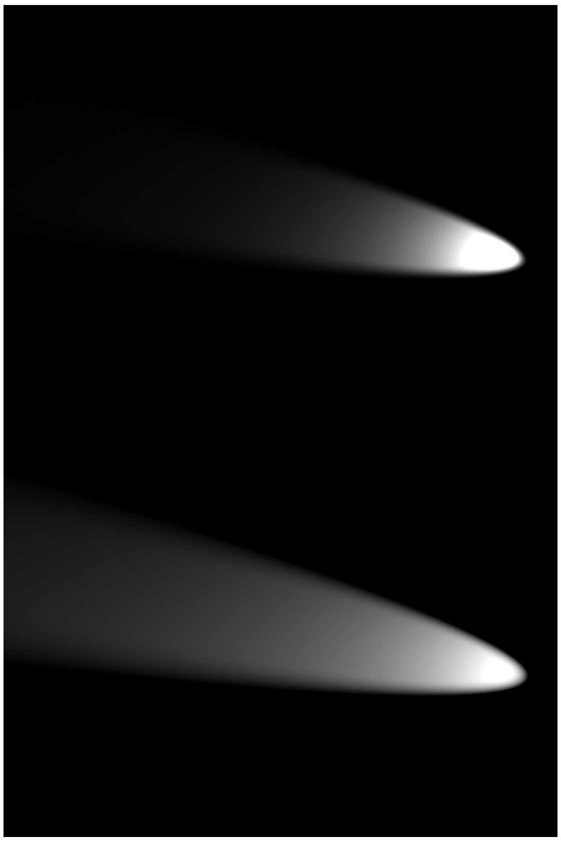


Figure 5 – A spot light illuminates a ground plane with quadratic falloff. Top: Without gamma correction, we get a too strong falloff and overexposed hotspot. Bottom: Proper result with gamma correction

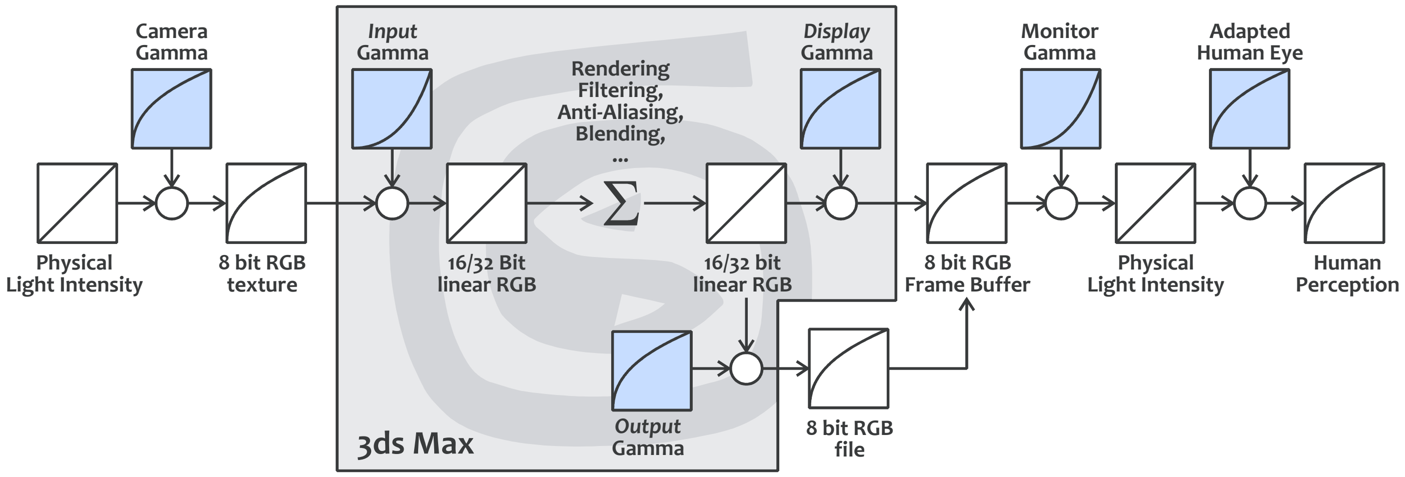


Figure 6 – The complete gamma/correction pipeline, starting with the texture acquisition using a camera (left) up to the display of a 3D rendering on the screen (right) with intermediate storage as 8 bit image file (center). In order to be able to correctly calculate within the 3D software, the image data has to be linearized initially (using the *Input* Gamma) and to be stored non-linearly at the end (applying *Display*/*Output* Gamma). Finally, the correct light intensities are produced on the screen due to the intrinsic gamma curve of the hardware (Monitor Gamma), which then are perceived non-linearly by our eye, just as the original intensities would be.

08. Any questions?

At this point the question might arise how it is possible that such dramatic errors are made but we all have had good results for years without gamma correction? There are several reasons for this: First, we have all learned to work around the incorrect results, by adding more fill lights, or by brightening up downsized images, for example. Secondly, many algorithms (e.g. lens flares) have special code to compensate for problems that originally arise due to missing gamma correction, in order to produce "pretty" results. These results have no physical basis anymore and have nothing to do with realistic behavior of light. And finally, we all have learned to accept certain artifacts caused by lack of gamma correction, for example when computing anti-aliasing and having to increase sampling rates while the actual problem is too dark intermediate pixel values caused by gamma ignorance.

For those that still doubt the 0.22 * 2 = 1.0 statement: you can simply use the light meter built into your camera to measure for yourself that a grey level value of 128 will only produce 22% of the maximum brightness on screen if no gamma correction occurs: For this, you should simply put your camera to the highest ISO sensitivity setting, point your camera at your screen and read out the exposure values for the grey levels 255 and 128 respectively. Half the brightness should lead to twice the exposure time in your camera – but for grey level 128 your camera will report a quadrupled time, indicating that the actual brightness is approximately 25%.

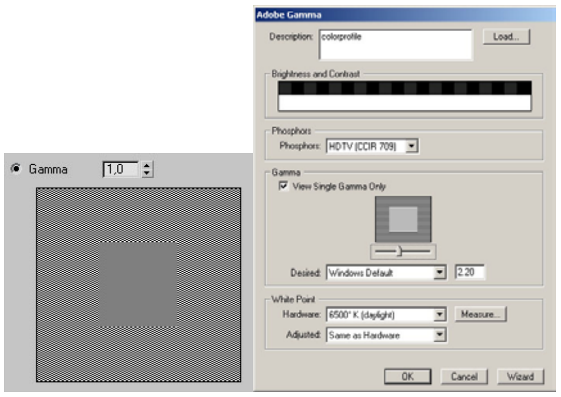
Gamma correction always has to be applied when the display device behaves non-linearly or when the input data is stored non-linearly and needs to be processed. This is the case for images taken with a still camera, for example. The RAW format of most digital cameras is initially storing linear values but as soon as it is converted to JPEG or TIFF, a non-linearity is introduced that needs to be corrected before using the data for calculations. Also data from scanners and video cameras is stored non-linearly most of the time and has to be gamma corrected.

09. sRGB

So how can we determine what kind of nonlinearity is contained in our images and output devices?

Several companies sell special measuring devices which can profile and/or calibrate screens, projectors, printers or cameras. Spyder from Datacolor, EyeOne from x-rite or huey from Pantone are some examples. These devices measure the exact properties of the imaging hard- and software and store the results in a standardized form as color profile to ensure consistent color rendering on multiple output devices. Sometimes, correction tables can be generated additionally making sure that the measured device is meeting exact standards.

Alternatively there is a multitude of image plates and tools available on the Internet designed to allow determining a display gamma value by simply looking at intersecting grayscale gradients. Most of the time, these tools take advantage of the fact that a fine pattern of black and white pixels is supposed to look exactly like a 50% gray value in combination with the right gamma value. Adobe Gamma is one of the tools using this technique, as is 3ds Max (Figure 7).


Gamma 1,0

Adobe Gamma
Description: colorprofile
Brightness and Contrast
Phosphors
Phosphors: HDTV (CCIR 709)
Gamma
☑ View Single Gamma Only
Desired: Windows Default 2.20
White Point
Hardware: 6500° K (daylight)
Adjusted: Same as Hardware

Figure 7 – Visual estimation of gamma by comparing a black and white pattern next to a patch of gray, in 3ds Max (left) and Adobe Gamma (right)

Typically, these estimates can vary easily by +/- 0,2 units for the gamma value[4], so nowadays, with a modern flatscreen, you might do just as well by simply assuming a gamma of 2.2. This value is derived from the so called sRGB color space which has become a very widely used standard for color rendering. If you do not have the highest demands for color reproduction accuracy you should be well served with this assumption since most screens and cameras have at least a sRGB preset[5]. More precisely, sRGB does not use a single gamma curve but replaces the lowest part of a gamma=2.2 curve with a straight line, but gamma=2.2 is a pretty good approximation in most cases.

Unfortunately, the situation for image files properties is a more complicated. Some image formats (e.g. JPEG, PNG, PSD, TIFF) allow metadata for saving information about which color space (and therefore using which gamma) the pixel values should be interpreted in. For other formats, or when metadata is missing, a rule of thumb is: if the image looks “good” on a monitor, the gamma of the image matches that of the monitor (which is often sRGB). An exception to that rule is images that were not made to be viewed directly but encode special object properties, such as surface normals, displacement height, or other general influence maps. This data is normally computed without gamma and therefore should not be corrected. The same applies to HDR images that normally should be saved without gamma. Hand-painted textures, on the other hand, that were judged by an artist on his/her screen should be saved in that screen’s color space and gamma.

Here is a warning in case someone might now have the idea to generally pass all 8 bit images through a gamma correction and then save them as linear data for later use. As explained before, 8 bits are not enough to encode an image without visible artifacts.

And for those who might want to know if a particular piece of software is handling gamma correctly, they can simply have the software scale down an image consisting of alternating one-pixel lines of black and white color down to 50%. If you observe the reduced result and the original image from some distance such that you cannot see the individual lines in the original any more (squinting might help), both should appear of equal brightness if gamma correction is performed. Alternatively, you can output a 50% pixel value and use your camera light meter as described in section 08 to measure whether this really produces 50% luminance.

10. *Hands on*

Using 3ds Max 2009[6] as a case study, we now will explain in detail what needs to be done to establish a correct gamma workflow in practice. Similar procedures exist in other 3D packages, but might require the use of individual shaders to perform the gamma correction.

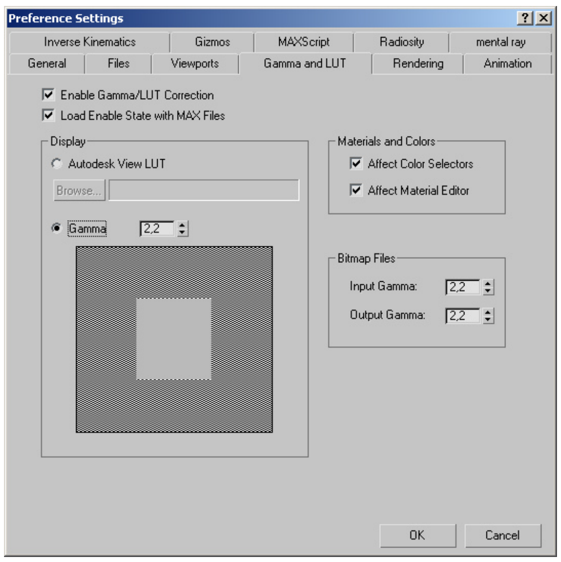
3ds Max has the ability to perform a general gamma correction when loading textures, displaying images on screen as well as when saving images. For this, you open the tab *Gamma and LUT* in the *Preferences* dialog and select the following settings for the normal case of an sRGB workflow (Figure 8):

Enable Gamma/LUT Correction = on
Display > Gamma, value = 2.2
Affect Color Selectors = on
Affect Material Editor = on
Input Gamma = 2.2
Output Gamma = 2.2

[4] Also, a one-pixel checker pattern can produce display artifacts due to interference of neighboring pixels and thus distort the gamma estimate.

[5] Within the print design community, sRGB does not have a very good reputation due to it’s limited color gamut and it’s intended use in RGB display devices, which makes it only second choice for images designed to be printed based on CMYK color separation. Many prefer the Adobe RGB color space, which also uses a gamma of 2.2.

[6] Changes introduced with 3ds Max 2010 are added as comments to each section



[✓] Enable Gamma/LUT Correction
[✓] Load Enable State with MAX Files

Display
○ Autodesk View LUT
● Gamma 2,2

Materials and Colors
[✓] Affect Color Selectors
[✓] Affect Material Editor

Bitmap Files
Input Gamma: 2,2
Output Gamma: 2,2

Figure 8 – The Gamma parameters in 3ds Max with the recommended settings for an sRGB workflow

It is recommended to activate the option *Load Enabled State with MAX Files* to ensure that the gamma correction parameters will be used later on when opening the scene file on some other system. Unfortunately, this also needs to be set manually for each machine participating in Backburner network rendering. 3ds Max 2010 has changed both behaviors by asking in the *Gamma and LUT Settings Mismatch Dialog* what to do when it encounters differences in gamma between the loaded file and the default 3ds Max settings.

The parameters *Input* and *Output Gamma* determine that a gamma of 2.2 will be used by default when opening image files. This value can be changed later on for each image file separately while opening the file in the *Bitmap File Dialog*. There you can select whether the gamma value saved in the header of the image file should be used (*Use image's own gamma*), whether the default value from the preferences should be used (*Use system default gamma*), or whether a specific value just for this file should be used instead (*Override*). For bump, normal or displacement maps, the latter option with a value of 1.0 should be used, as these image files are pure data channels and should not be gamma corrected.

In principle, all settings are now made for proper gamma correction in 3ds Max. Now, a grey value of 128 from the 3ds Max color picker will indeed produce 50% brightness on screen.

Unfortunately, there are still some small glitches in 3ds Max when dealing with gamma: The preview thumbnails in the File Open dialog will ignore the gamma settings, and so does the Exposure Control preview image. (Note: 3ds Max 2010 fixes both). Prior to 3ds Max 2010, you have to restart 3ds Max after changing gamma parameters (or at least select *File* > *Reset*). And some parts of the software still do not work well with gamma correction, such as lens flares, the mental ray *Glare* shader or the *Gradient* map (which can be corrected using *Output Curves*).

In the best tradition, a simple teapot wireframe model will serve as our first test object: In a first step, render the image without gamma correction and save it. Then enable gamma correction, save the scene temporarily using Edit > Hold, restart 3ds Max using File > Reset, and finally reload the saved scene file using Edit > Fetch, now with gamma correction enabled. (Note: for 3ds Max 2010, you simply enable gamma and continue). Now render the same image once more (with otherwise identical render settings), save it and compare it to the first version. You will notice a dramatic difference between the two thanks to the error made when rendering anti-aliasing without gamma correction (Figure 9). Similar differences can be observed in anti-aliasing/filtering of textures close to the horizon, or with motion blur effects (Figure 4).

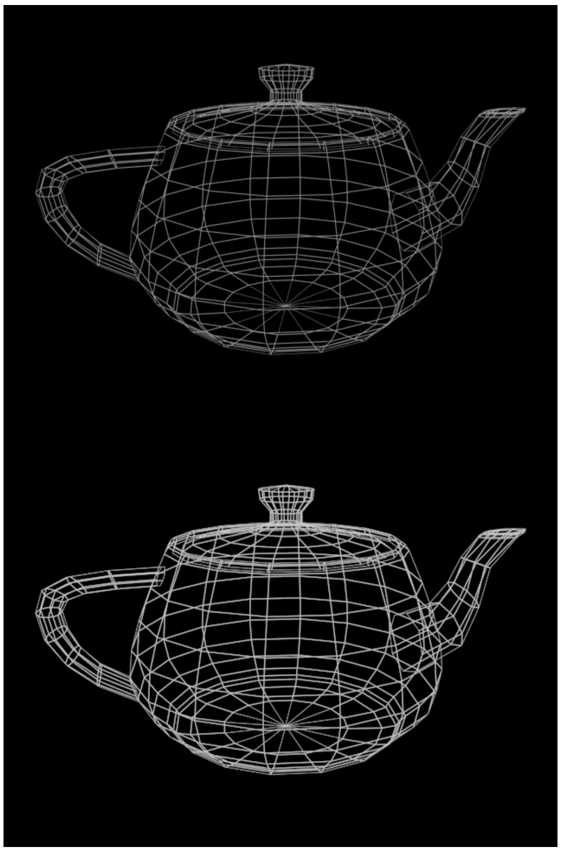


Figure 9 – A wireframe rendering, top without gamma correction and bad anti-aliasing, bottom with identical render settings but proper gamma correction and much better results

Note that in most cases it will not work to simply activate gamma correction for old scene files and re-render since these scenes were optimized to produce pretty pictures despite using wrong math. In order to preserve all the colors used in old scenes after activating gamma correction, all

RGB values need to be converted according to this formula: $new = 255 \cdot (old/255)^{2.2}$, otherwise you will get washed out colors. Also, light levels, bounce lights, amount of reflections etc. might need to be adjusted.

When working with Global Illumination or other physically correct light sources, proper gamma correction is absolutely mandatory. One the one hand for linearizing 8-bit image data and thus making it compatible to the physically correct algorithms used, one the other hand for preparing the linear image data computed by the render for an accurate display on screen. Gamma correction is so important in this case that 3ds Max will automatically do a gamma correction in the *mr Photographic Exposure Control* if it has not been activated in the preferences yet.

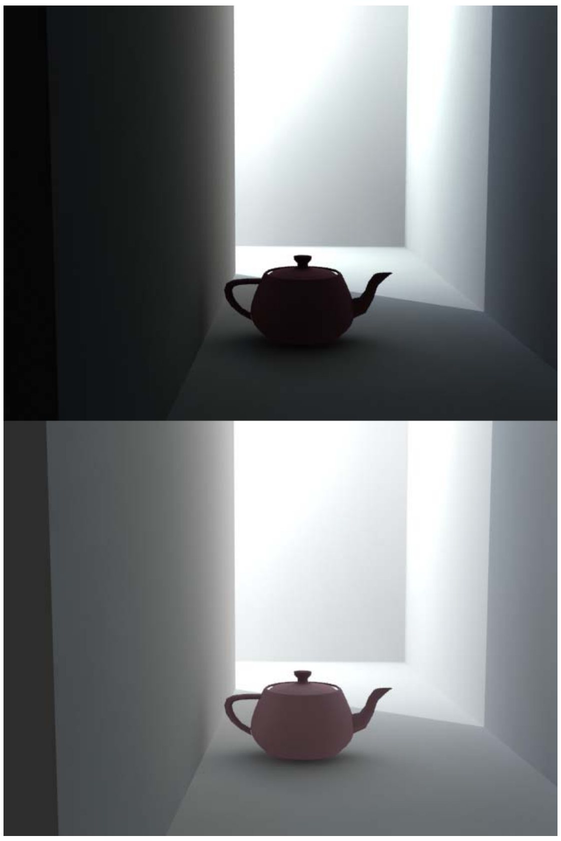


Figure 10 – Example for the influence of gamma correction on global illumination rendering. Top: without, bottom: with gamma correction. Light intensity for the top image was adjusted so the overall brightness of the rear wall roughly matches the one in the bottom image.

11. Floating Point Color and HDR

The problem of errors due to missing gamma correction is largely moot as soon as all steps are performed with floating point precision. Then all image data can be saved and processed directly as linear color without further transformations. This is especially useful when employing global illumination models and High Dynamic Range (HDR) images and the results need to postprocessed in compositing. This approach has recently become popular under the name "Linear Workflow". In particular, the OpenEXR file format developed and freely released by ILM is often used in this workflow as it allows very efficient storage of floating point data.

Only when displaying images on screen and when using "classic" 8-bit low dynamic range textures you still need to perform gamma correction, but not when saving images to floating point format.

12. Data channels

Some image formats allow storage of additional pixel data besides RGB color. Similar to the situation where you store computational data in the RGB channels, this is a special situation that needs to be considered when performing gamma correction on image files. Typical examples for additional image data are alpha channel, matte pass, depth buffer, bump map, normal map, displacement map etc.

Data channels are typically not made to be directly shown to a human but serve as input for some algorithm, and therefore do not have to deal with the nonlinearities of a display or the human eye. Subsequently, this type of data should not be gamma-encoded, but saved and loaded as linear data, much like HDR images. Special care needs to be taken when manually painting such data channels (e.g bump maps) in image editors such as Photoshop.

The alpha channel is a special case, since it directly encodes the transparency of an object and therefore is affected by the non-linear sensitivity of the human eye. Therefore, alpha should be gamma-encoded when saving to 8 bit.

In general it is important to understand how the individual software packages encode their data and to make sure that any non-linearity is undone before data is processed by the renderer.

13. Et tu, Brute?

It should be mentioned at this point that even in 2009, only a few image processing software packages are handling gamma correctly. Photoshop, for example, the work horse of most visual media productions, is still ignoring gamma when scaling down images or blending layers[7]. Re-

[7] A very simple test to illustrate this: Create a 2*2 pixel checker pattern using pure black and white, then fill the entire image with that pattern. Now scale it down to 50% size (which should produce a 1:1 mix of black and white, resulting in 50% brightness), and up again by 200% to get the original image size – the image will now be much darker. Do the same steps, but now in

cently, Photoshop has received a preference option *Blend RGB colors using gamma=1.0* which fixes the error when blending layers but scaling images is still broken[8]. In contrast, film compositing software such as Nuke or Digital Fusion allow you to assign color spaces to images and appropriately apply gamma correction before running any image operations.

It is interesting to note that recent versions of OpenGL and DirectX added support for a gamma-enabled workflow (by supporting gamma-encoded sRGB textures and performing linearization in realtime), and a range of current game and engine developers (e.g. Half Life 2) have implemented this.

14. *Final words*

Gamma correction is a rather simple tool in the quest of correct reproduction of images on different systems, summarizing the system characteristics with a single exponential transfer curve[9]. It would be better to properly handle the entire color space in which images are recorded and displayed. While this is a standard technique in print (and also often in compositing) it is largely unsupported in 3D computer graphics, even if, for example, mental ray provides support for color profiles since version 3.4. Hopefully this will change in the future with increased demand for correct color management.

I hope I was able to show that gamma correction is a simple but important step in all areas of image processing and in particular for 3D computer graphics. The correct application of gamma curves allows the reduction of a whole range of well-known image artifacts with a few simple steps and will, if used consistently, lead to better looking imagery. Of course gamma correction is no silver bullet, but ignoring it causes a whole range of unnecessary problems that are easily avoidable.

32Bits/channel color mode to see this being done correctly; also note the actual pixel values in both cases.

[8] Other examples of software not using gamma correction when scaling images are Gimp, XnView, Matlab, Word, Firefox

[9] Typically, three separate gamma curves for red, green, and blue are used but nevertheless this is an oversimplification of actual display characteristics; gamma lookup tables improve on this but still lack the accurate representation of color.

15. *Take Home Message*

Here are some (slightly oversimplified) facts about gamma that basically capture the entire story:

1. 3D rendering and most other image processing algorithms assume linear light encoding, meaning that a pixel value of 128 encodes half the maximum brightness.
2. 8 bit is not enough to encode images without artifacts, so you need to use gamma encoding; which means any 8 bit texture/photo/drawing (from cameras, scanners, or from painting on screen) needs to be linearized before feeding it into a graphics algorithm; also when saving the final output of such an algorithm in 8 bit, you need to apply gamma.
3. A computer screen has a gamma curve, so linear data needs to be corrected before displaying it on screen in order for it to produce correct luminance levels.
4. sRGB is the most commonly used color space for image recording and display devices, and it has a transfer curve very similar to gamma=2.2.

16. *Further reading*

http://www.poynton.com/notes/colour_and_gamma/GammaFAQ.html
Charles Poynton, specialist for digital color imaging systems, has some excellent and very detailed information on gamma correction and color reproduction.
http://www.4p8.com/eric.brasseur/gamma.html
This interesting web site has several nice test images that dramatically show how almost any image processing software is completely ignoring gamma correction when scaling images.

http://www.teamten.com/lawrence/graphics/gamma/
This web page by Lawrence Kesteloot, a former employee of PDI, describes what is wrong with processing images without gamma correction.

http://mysite.verizon.net/spitzak/conversion/composite.html
Bill Spitzak of Digital Domain shows a range of nice, interactive examples on why it is important to use linear floating point precision for compositing and 3D, with gamma correction as a necessary part of that.

http://mymentalray.com/wiki/index.php/Gamma
Entry in the mental ray Wiki on the topic of gamma with practical tips and explanations.

17. Additional comments

This section contains some late comments and other information related to gamma correction that did not fit into the general article flow:

- Some HDR images available on the web have gamma applied
- Some users argue that using a default input gamma of 1.0 in 3ds Max makes for a better workflow as this ensures that all non-color image data (bump, normal maps etc.) are loaded correctly. Uncorrected color images are typically easy to spot and can be adjusted using the 3ds Max tools.

18. Acknowledgements

Håkan "Zap" Anderson of mental images provided the foundation for much of the information contained in this article and never got tired of reiterating why gamma correction is important.

Thanks to David Baker, for his useful comments and corrections, as well as to Gavin Greenwalt and Joep van der Steen for their comments.