Stop double correcting for gamma

Akash Garg

Version 1.0 - July 14, 2024

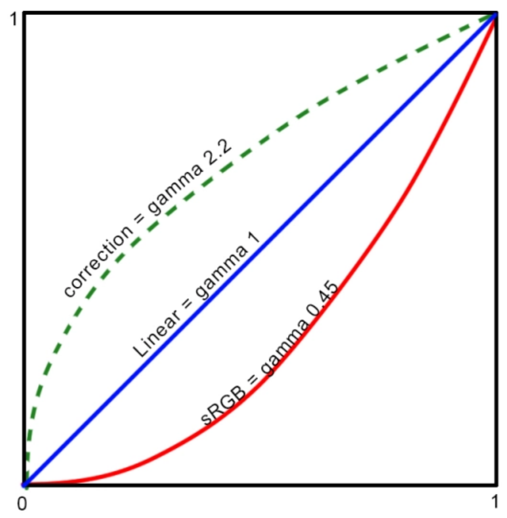

Figure 1: Gamma Transfer Function

There has been a lot written about gamma correction and sRGB, I won't rehash all that here (See [5, 1, 4, 3]). The main thing to note is that our perception is non-linear and displays accomodate for that. The sRGB curve is exponential, approximated by `pow(rgb, 2.2)`, but a more exact formulation can be found on wikipedia [2].

The quick summary:

sRGB images are saved (gamma corrected), meaning they are saved as the dotted green curve above. When loading sRGB images, one need to convert them to linear space using:

$$C_o = C_i^{\gamma} \approx C_i^{2.2}$$

Note that the previous range $[0, 0.5]$ on the linear curve is now mapped to approximately $[0, 0.73]$; meaning that we have about 75% of the bits to use for the darker regions (¡ 50%) of the color values. This is important because human perception is more attuned to darker tones than lighter tones.

Once we need to send the output from our renderer to the screen, note that the display will transform the images to sRGB space (the red curve). Thus we need to do *gamma correct*, shift our values back to the dotted green curve so that when the display performs it's own sRGB conversion the color values will be in linear space as intended.

$$C_o = C_i^{\frac{1}{\gamma}} \approx C_i^{\frac{1}{2.2}}$$

Usually we don't have to do all this manually. When loading sRGB images, you can denote the texture image is in sRGB format and they will be converted for you in hardware available in linear space when you access values in your shaders. When using a standard image loading library however, this conversion is not done for you. If you need to get linear values you need to apply the inverse gamma function to move images back to linear space.

Gamma correction before the rendering is also mostly done for you hwen you denote your render target output encoding has sRGBEncoding.

If your renders are too bright, you are applying gamma correction step `pow(rgb, vec3(1/2.2))` too many times. If your renders are too dark you are probably missing the linear conversion `pow(rgb, vec3(2.2))`.

References

[1] John Novak. *What every coder should know about gamma*. URL: https://blog.johnnovak.net/2016/09/21/what-every-coder-should-know-about-gamma/.

[2] *SRGB*. URL: https://en.wikipedia.org/wiki/SRGB.

[3] *Tone Mapping vs. Gamma Correction*. URL: https://computergraphics.stackexchange.com/questions/10315/tone-mapping-vs-gamma-correction.

[4] *Understanding Gamma Correction*. URL: https://www.cambridgeincolour.com/tutorials/gamma-correction.htm.

[5] Joey de Vries. *Gamma Correction*. URL: https://learnopengl.com/Advanced-Lighting/Gamma-Correction.

Revision	**Date**	**Description**
1.0	July 14, 2024	Initial draft