**Gamma Conversion**

On importance of Gamma Conversion you can read “Gamma Error in Picture Scaling.pdf” written by Eric Brasseur and also hosted on his website at <http://www.ericbrasseur.org/gamma.html>

While I have no permission to distribute his article, I saved it as PDF for archiving purposes if his website ever goes down.

**Color Spaces**

Color spaces are defined by 2 things:

1. How color is constructed i.e. color components
2. How components are represented numerically

This program will work with RGB color components. Image decoders will be asked to output RGB format image when an image is opened and will do conversion internally.

Numerical representation of RGB is defined by gamma formulas:

**Linear color space** **aka XYZ**

is the simplest. Values on a scale from 0 to 1 represent % of linear brightness. 0 is the black tone. 1 is white. 0.5 is gray. This color space should be used for all image processing. In this color space to calculate average of 2 brightness values you simply add them up and divide by 2.

**Simple Gamma color space**

Formula for translating from XYZ CS to this color space is where gamma usually is between 1.0 and 3.0 and most often is around 2.2.

Translating from corrected color space to linear is performed with

**sRGB color space**

A formula like simple gamma formula that is a bit tuned.

In those formula input and output values are normalized to be between 0 and 1:

The formula for transferring from sRGB color space to linear (CIE XYZ) brightness space is from Wikipedia article <https://en.wikipedia.org/wiki/SRGB>.

The formula for transferring from linear color space to sRGB is:

**Other Color Spaces**

Currently are not implemented

**Implementation**

**Class Structure**

Every color space conversion is performed with object associated with particular color space. Objects are created when first needed and then reused until the program stops.

Conversion is performed using pre-calculated conversion tables for speed. If we convert from 16 bit value we have to maintain a table of 65536 indices which is . Therefore is we have an image that has more than 65536 pixels, or dimensions larger than 256x256 pixels we will be doing more work if we convert for each pixel directly.

Base **class GammaConverter** provides interface for converters:

1. 8bit corrected to 16bit linear
2. 16bit corrected to 16bit linear
3. 16bit linear to 8bit corrected
4. 16bit linear to 16bit corrected

Concrete classes should implement methods that fill conversion tables of the base class and therefore define conversion formula.

Because converter object should be initialized once and re-used **class GammaDispatcher** manages creating and keeping pointers to existing converters and should be used to request a converter by the user.

**Calculation**

Calculations are performed in double and then rounded to nearest integer with std::lround().

**sRGB calculation**

The formula given above is for normalized real values between 0 and 1. In the program values are represented as integers between 0 and 255, 0 and 65535 for 8 and 16 bit respectively. sRGB values can be both 8bit and 16bit while linear values are only 16bit.

The function is split into two regions. If we calculate integer values corresponding to those regions (called thresholds) we get for 8 bit:

And for 16 bit:

Since they do not change between runs and sRGB is most used color space it would be wise to use math software and pre-calculate all those values to include them at the compile time. This is a TODO item. Check out this software <https://smath.com/en-US/view/SMathStudio/pricing>