Gamma Correction

Galina Paskaleva

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Radiance in the direction $\theta$:

$$R(\theta) = \frac{d^2\Phi}{dA d\Omega \cos \theta} \text{[W/m}^2\text{sr]}$$

where:

$\Phi$ [W]... the total power of electro-magnetic radiation in all frequencies emitted by the surface with area $A$;

$\theta$ [1] ... the angle between the surface normal $\mathbf{N}$ and the viewing vector $\mathbf{V}$;

$\Omega$ [sr] ... the solid angle obtained by projecting the observed surface onto a unit sphere with center at the point from which the measurement is taken.
Radiance:

\[
R = \int_{\phi = 0}^{2\pi} \int_{\theta = 0}^{\pi/2} R(\theta) \cos \theta \sin \theta \, d\theta \, d\phi \quad [\text{W}/\text{m}^2]\n\]

Spectral Radiance in the direction \(\theta\):

\[
R(\lambda, \theta) = \frac{d^2\Phi(\lambda)}{dA d\Omega \cos \theta} \quad [\text{W}/\text{m}^2\text{sr}]
\]

Spectral Radiance:

\[
R(\lambda) = \int_{\phi = 0}^{2\pi} \int_{\theta = 0}^{\pi/2} R(\lambda, \theta) \cos \theta \sin \theta \, d\theta \, d\phi \quad [\text{W}/\text{m}^2]
\]
Psychophysics

Luminosity for monochromatic light:

\[ L(\lambda) = 683 \ f(\lambda) \ R(\lambda) \ \text{[cd/m}^2\text{]} \]

where:

\[ f(\lambda) \ldots \text{the standard luminosity function, describing the average sensitivity of the human eye} \]

\[ L = 683 \int_{\lambda=0}^{\infty} f(\lambda) \ R(\lambda) \ d\lambda \ \text{[cd/m}^2\text{]} \]
Steven’s Law

- describes the relation between
  - Radiance (R) ~ Intensity (I) and
  - Luminosity (L) ~ Brightness (B)
- by approximating the integral

\[
L = 683 \int_{\lambda = 0}^{\infty} f(\lambda) R(\lambda) \, d\lambda \quad [\text{cd/m}^2]
\]

as

\[
d\frac{B}{B} = \gamma \cdot \frac{dI}{I} \quad \rightarrow \quad B = c \cdot I^\gamma
\]

- in a PC \quad \gamma = 0.45
- in a Mac \quad \gamma = 0.55
CRT Display

- CRT has a non-linear response to the input signal, described as the relation between
  - Voltage (V) and
  - Intensity (I)

\[ I = a \cdot V^\gamma + b \]

where \( \gamma' = 2.5 \)

- incidentally \( 0.45 \times 2.5 \approx 1.0 \)
Transfer Functions

- storage linear in intensity (practically unused)

- storage linear in brightness (most common)
Storage Types

linear in Brightness

linear in Intensity

0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
If an image looks good in a web browser, it has, in all probability, already been gamma-corrected.

Alpha-channels, Height or Displacement Maps, on the other hand, are not.

JPEG files are pre-corrected for a $\gamma = 2.2$,
- i.e. they are linear in brightness but non-linear in intensity.
Linearity

- a linear transformation is characterized by:
  - $f(x + y) = f(x) + f(y)$ and
  - $f(\lambda x) = \lambda f(x)$

- the transfer of intensity to brightness $g$ is not linear:
  - $g^{-1}(g(I_1)) = I_1$ but
  - $g(I_1 + I_2) \neq g(I_1) + g(I_2)$ and
  - $g(\lambda I_1) \neq \lambda g(I_1)$

- do we perform lighting and filtering with the intensity $I$ or with the brightness $B = g(I)$?
Interpolation w/o Gamma Correction

- **Intensity**:
  - 0.0
  - 0.5
  - 1.0

- **Encoding Transfer Function**:
  - $\gamma = 0.45$
  - 0.5

- **Perception / Storage**:
  - 0.0
  - 0.73
  - 1.0

- **Display Transfer Function**:
  - $\gamma = 2.50$
  - 0.5

- **Reference Values**:
  - 0.0
  - 0.5
  - 1.0

- **Interpolation**:
  - 0.0
  - 0.5
  - 1.0

- **Interpolated Values**:
  - 0.0
  - 0.22
  - 1.0

- **Perception of Reference Values**:
  - 0.0
  - 0.73
  - 1.0

- **Perception of Interpolated Values**:
  - 0.0
  - 0.5
  - 1.0
“Roping” Artefacts

“Bending” of anti-aliased edges due to the non-linearity of brightness space

- interpolated in intensity space

- interpolated in brightness space
Lighting w/o Gamma Correction
Incorrect Lighting Calculation

- gamma-corrected / not gamma-corrected
Store and Calculate in different spaces

natural scene/computer graphics

INTENSITY

encoding transfer function

$\gamma = 0.45$

linear in $I$

BRIGHTNESS SPACE

STORAGE

linear in $B$

$\gamma = 2.50$

de-gamma

INTENSITY SPACE
sRGB Texture

- sRGB texture format with $\gamma = 2.2$
  - proper gamma is applied by the HW before the results of texture fetches are used for shading
    - can be done manually
  - all samples used in a texture filtering operation are linearized by the HW before filtering
    - cannot be done manually
- SRGB8 and SRGB8_ALPHA8 are stored as 8-bit unsigned fixed-point values
sRGB Frame Buffers

- final re-application of the gamma correction by the \textbf{HW} before display on the monitor
- any value returned in the Shader is gamma-corrected by the \textbf{HW} before storage in the frame buffer (or render-to-texture buffer)
- blending is automatically performed in linear intensity space and the result is also automatically gamma-corrected
  - Alpha values are not affected
- use sRGB intermediate color buffers
Visible in the Application:

```cpp
glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_ARB_TEXTURE_SRGB, texture);

GLenum sRGBFramebuffer = GL_FRAMEBUFFER_SRGB;

glEnable(GL_ARB_FRAMEBUFFER_SRGB);
glBindFramebuffer(GL_DRAW_FRAMEBUFFER, fbuffer);
```

No influence in the Shader:

```cpp
in vec2 ex_texCoord;
uniform sampler2D un_texture;
vec4 color = texture2D(un_texture, texCoord);
```
visible in the Shader:

- **on fetch:**
  ```
  vec4 color = pow(texture2D(un_texture, texCoord), 2.2);
  ```

- **on last-stage output:**
  ```
  return float4(pow(finalCol, 1.0 / 2.2), pixelAlpha);
  ```

- **filtering** and **mipmapping** is performed in **non-linear** space!
DirectX 9 incorrectly converts into non-linear space before blending (not problematic for opaque surfaces)

DirectX 10 performs blending in linear space

