Gamma Correction

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Note: If color tones change when brightened / darkened, then most likely non-linear lighting is taking place and gamma correction is needed.
• Eyes perceive non-linear brightness

The *physical* brightness (photon count) is not equal to the *perceptual* brightness

– E.g.: A light at 22 % brightness appears half as bright!

• Monitors are not linear in brightness output

– CRT: Twice the input voltage ≠ twice the brightness
  – Gamma correction still valid with modern displays
Image Encoding

- Since perception is non-linear
- Images stored non-linear as well
  - sRGB colour space uses ~ 2.2 gamma
Gamma Transfer Function

• Conversion between
  – Linear space
  – Gamma space

• To compress linear values: \( V_{encoded} = \frac{1}{V_{linear}} \)

• To expand encoded values: \( V_{linear} = V_{encoded}^\gamma \)

Note: Values between 0 and 1
\( \gamma = 1/2.2 \)  
\( \gamma = 1.0 \)  
\( \gamma = 2.2 \)  

*lighter & less contrast*  
*darker & more contrast*
Gamma Correction & Rendering

- Textures typically stored in sRGB
  - Already gamma corrected!
- Monitor output in sRGB
  - Matches perfectly with the textures
- But: When doing calculations
  - E.g. Lighting, Shading, Blending
  - We must work in linear space!
Two ways of doing gamma correction

- Manually
  - Advantage: Full control over conversion
  - Disadvantage: Shader code needs to change
  - Disadvantage: Unnecessarily complicated and costly

- Let the GPU hardware do it
  - Advantage: No change in shader
  - Advantage: Hardware practically does it for free
  - Disadvantage: Less control
Manual Gamma Correction

• When sampling an sRGB texture
  \[
  \text{vec3 color} = \text{pow(texture2D}(\text{theTexture, uv}).\text{rgb}, \text{vec3}(2.2));
  \]

• For the final stage output
  \[
  \text{out vec4 FragColor;}
  \]
  \[
  [...]
  \]
  \[
  \text{FragColor} = \text{vec4}(\text{pow}(\text{finalColor}, \text{vec3}(1.0 / 2.2)), \text{alpha});
  \]

**Note:** Do not gamma correct in intermediate stages (e.g. when writing to the GBuffer). Only the final writing to the screen or to the final framebuffer needs to be gamma corrected.
Hardware Gamma Correction

• When loading an sRGB texture

```c
GLuint texId;
glGenTextures(1, &texId);
glBindTexture(GL_TEXTURE_2D, texId);
glTexImage2D(GL_TEXTURE_2D, 0, GL_SRGB, ..., data);
```

• Before writing to screen or final FBO

```c
glEnable(GL_FRAMEBUFFER_SRGB);
render();
glDisable(GL_FRAMEBUFFER_SRGB);
```

**Note**: You might need to call `glfwWindowHint(GLFW_SRGB_CAPABLE, true);` before opening your window.
Common mistakes

• Normal maps, displacement maps, etc.
  – The file should be saved in linear space (not sRGB)
  – Should **not** be loaded with GL_SRGB
  – Don’t need to be gamma corrected
  – They have nothing to do with perception!

• Be aware of the space (linear vs. gamma)
  – Don’t do calculations in gamma space
References

• [http://blog.johnnovak.net/2016/09/21/what-every-coder-should-know-about-gamma/](http://blog.johnnovak.net/2016/09/21/what-every-coder-should-know-about-gamma/)

• [http://learnopengl.com/#!Advanced-Lighting/Gamma-Correction](http://learnopengl.com/#!Advanced-Lighting/Gamma-Correction)