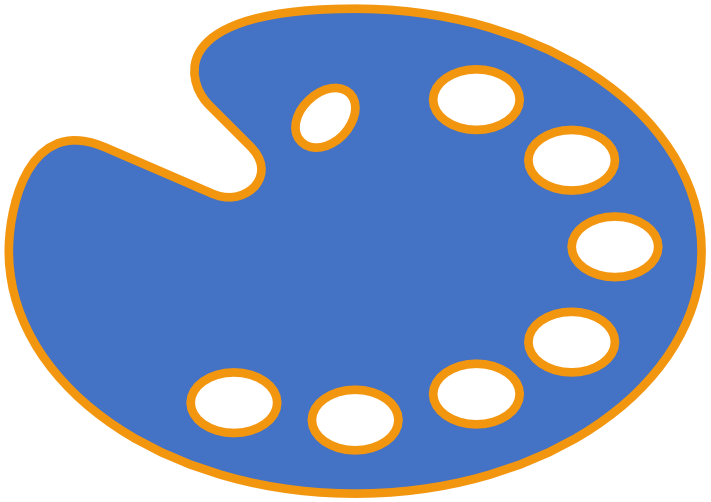
About

- Principal Coding Manager, Creative Assembly
- Working on Total War for 20 years
- Regular participant in WG21 meetings and telecons
  - "On the committee"
- Graphics proposal co-author P0267
- Linear algebra proposal co-author P1385
- Audio proposal co-author P1386
- Co-founder of #include <C++>



Everything you know about colour is wrong

J Guy Davidson

#include<C++>

includecpp.org

Agenda

- Identify colours
- Apprehend intensity and colour
- CIE1931 linear colour space
- SRGB non-linear colour space
- Transfer function
- Misapplication of colour management
- Linear algebra
- Uses of colour for C++
- Proposed API

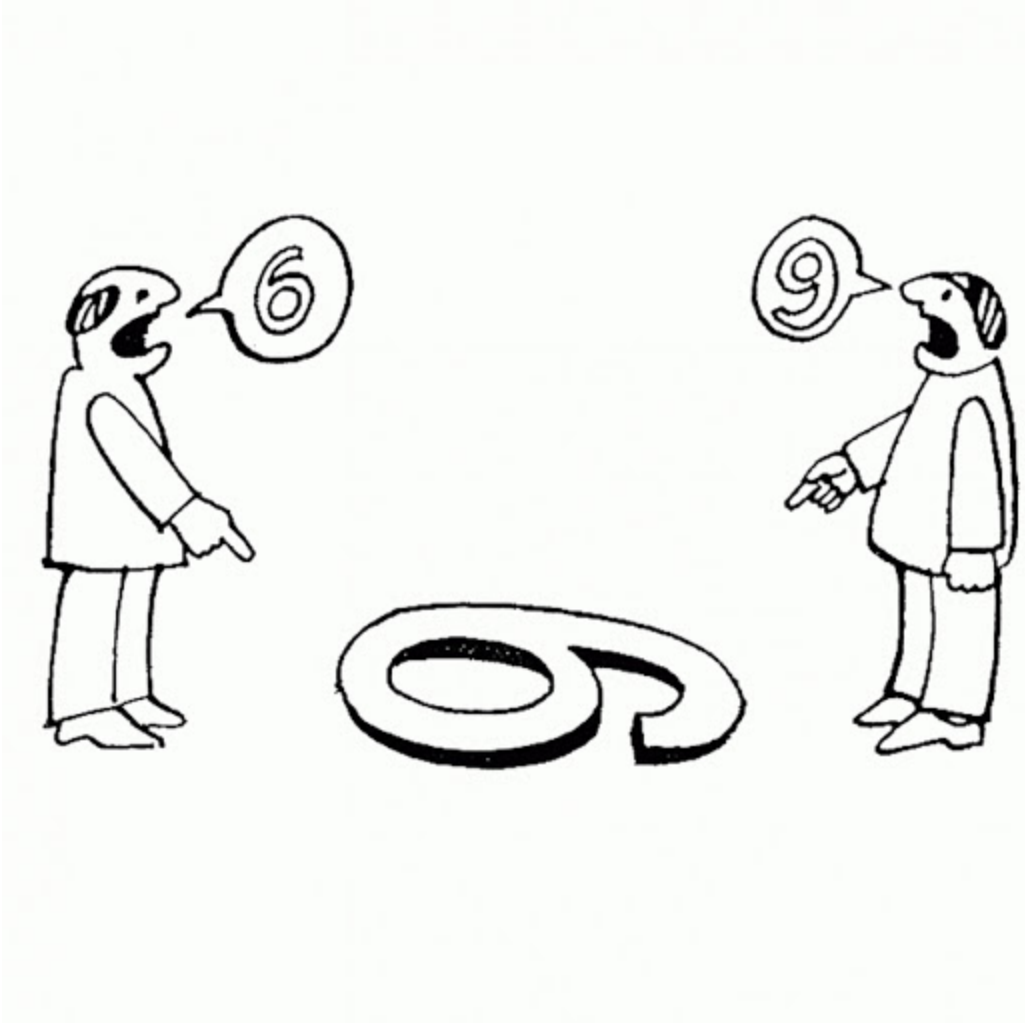
Quiz time

Answer in the chat

Part 1

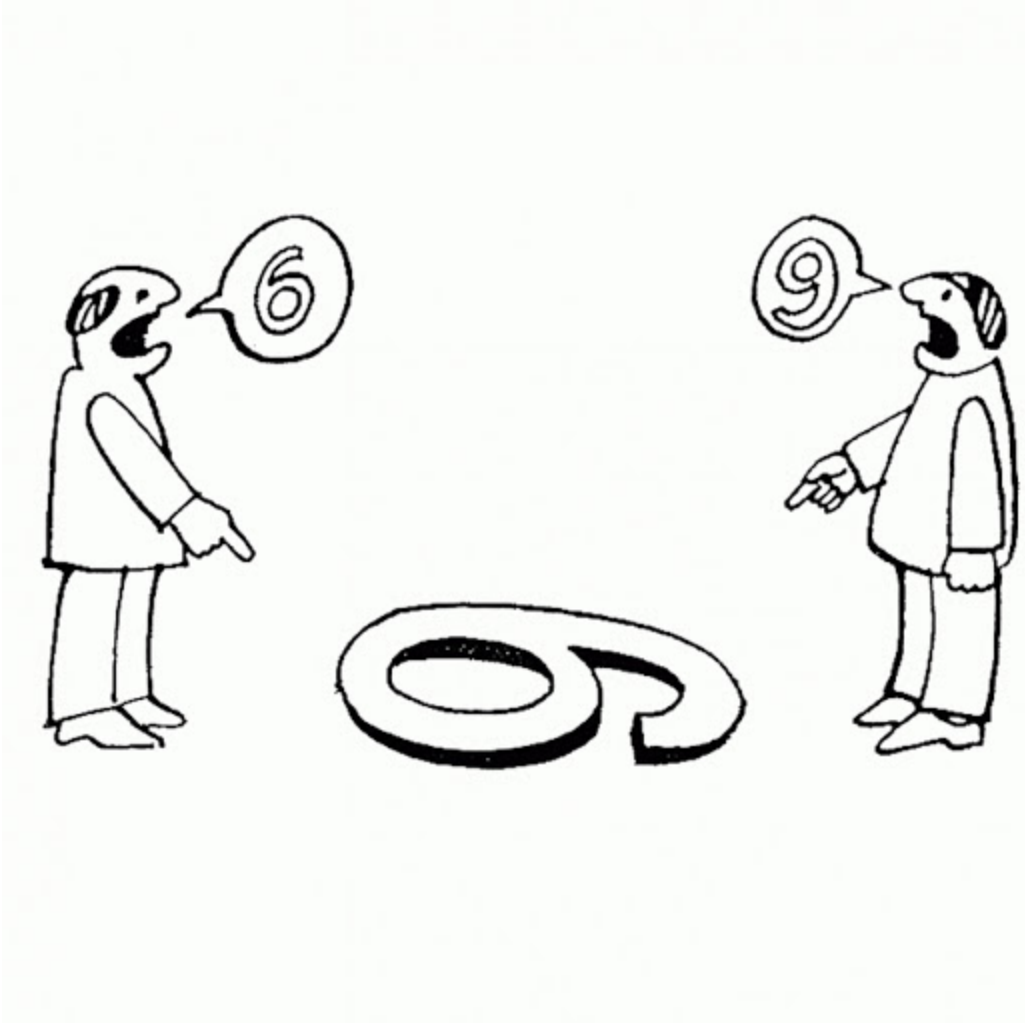
Colorimetry

Subjectivity



Subjectivity

(and context)



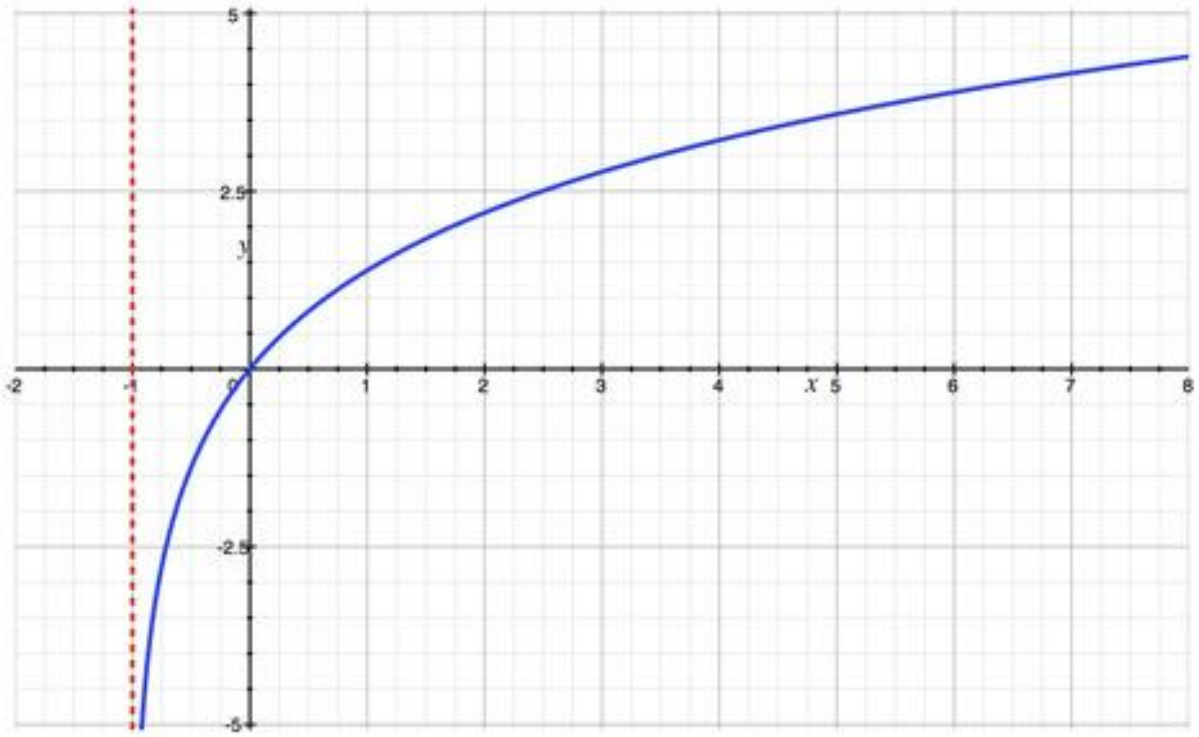
How do we eliminate subjectivity?

How do we eliminate subjectivity?

Consider only objective criteria

Measuring human vision

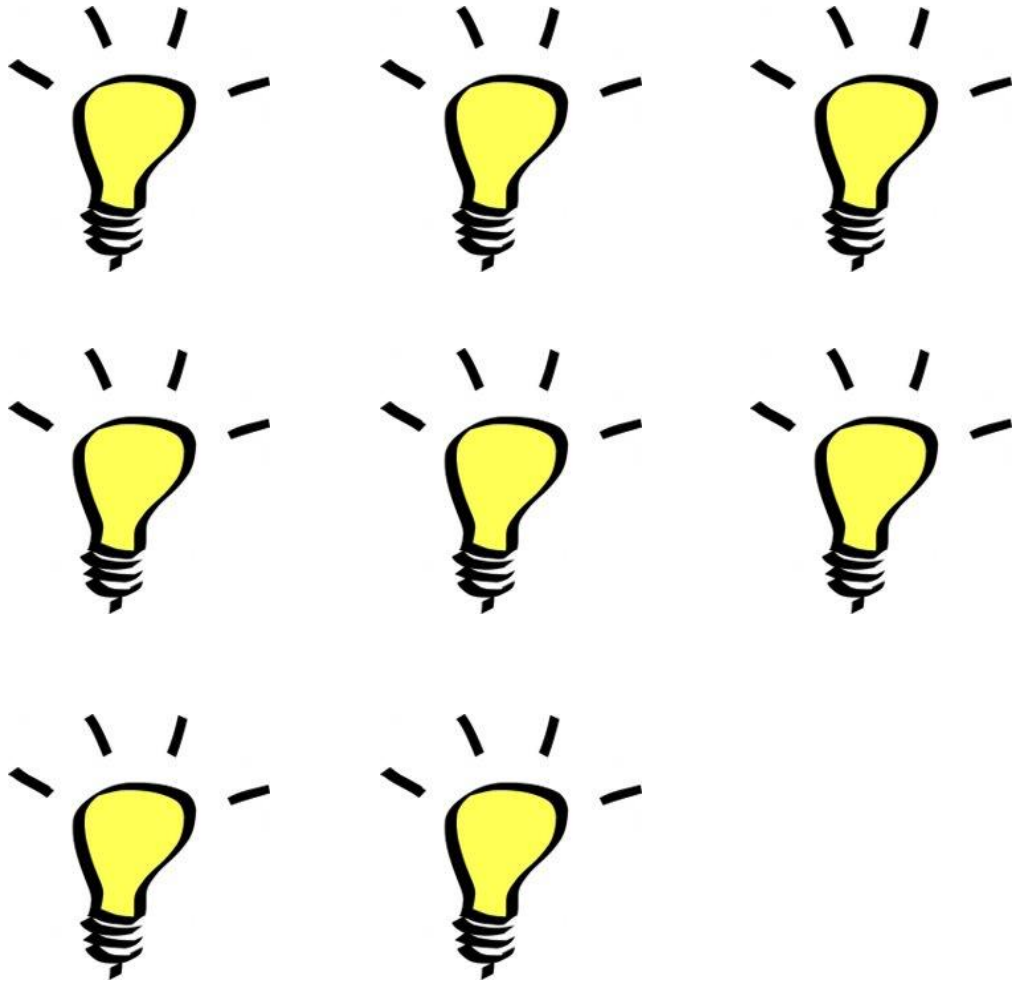
Human vision is logarithmic



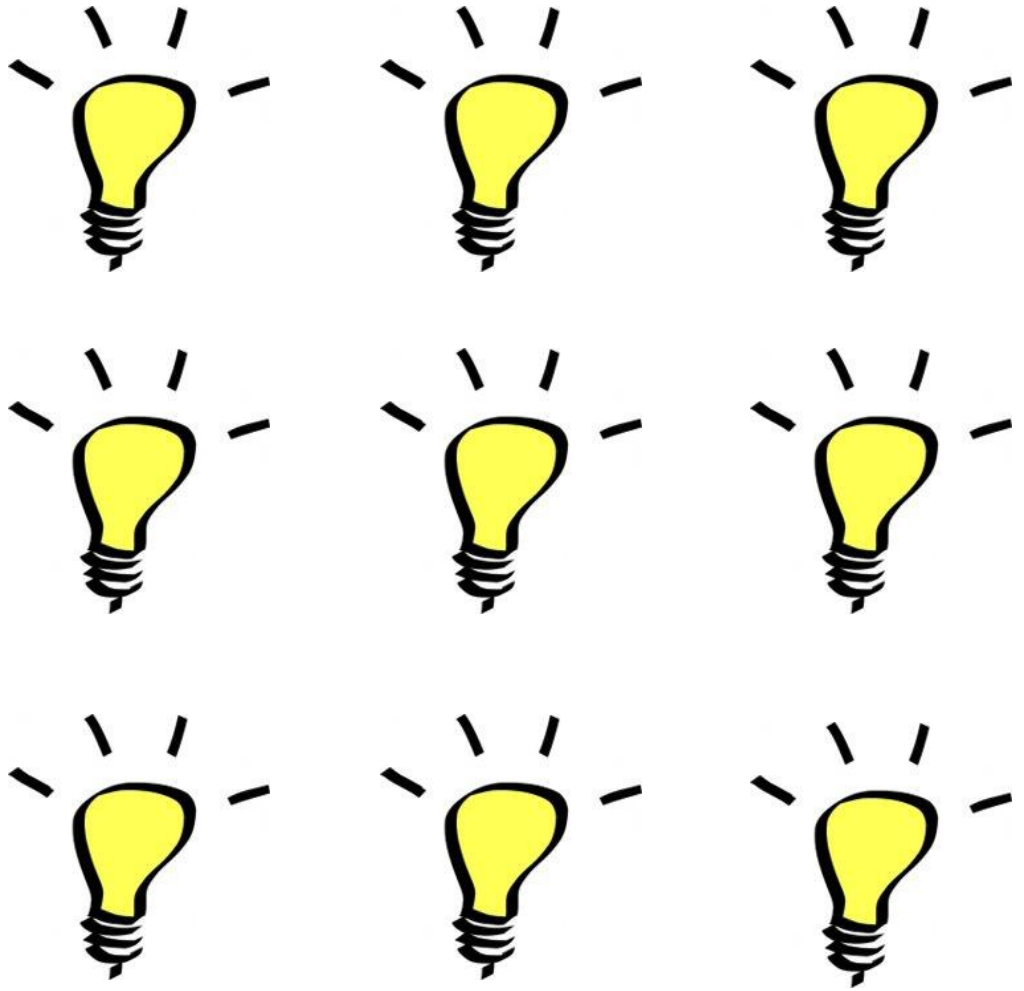
Human vision is logarithmic

Human vision is logarithmic

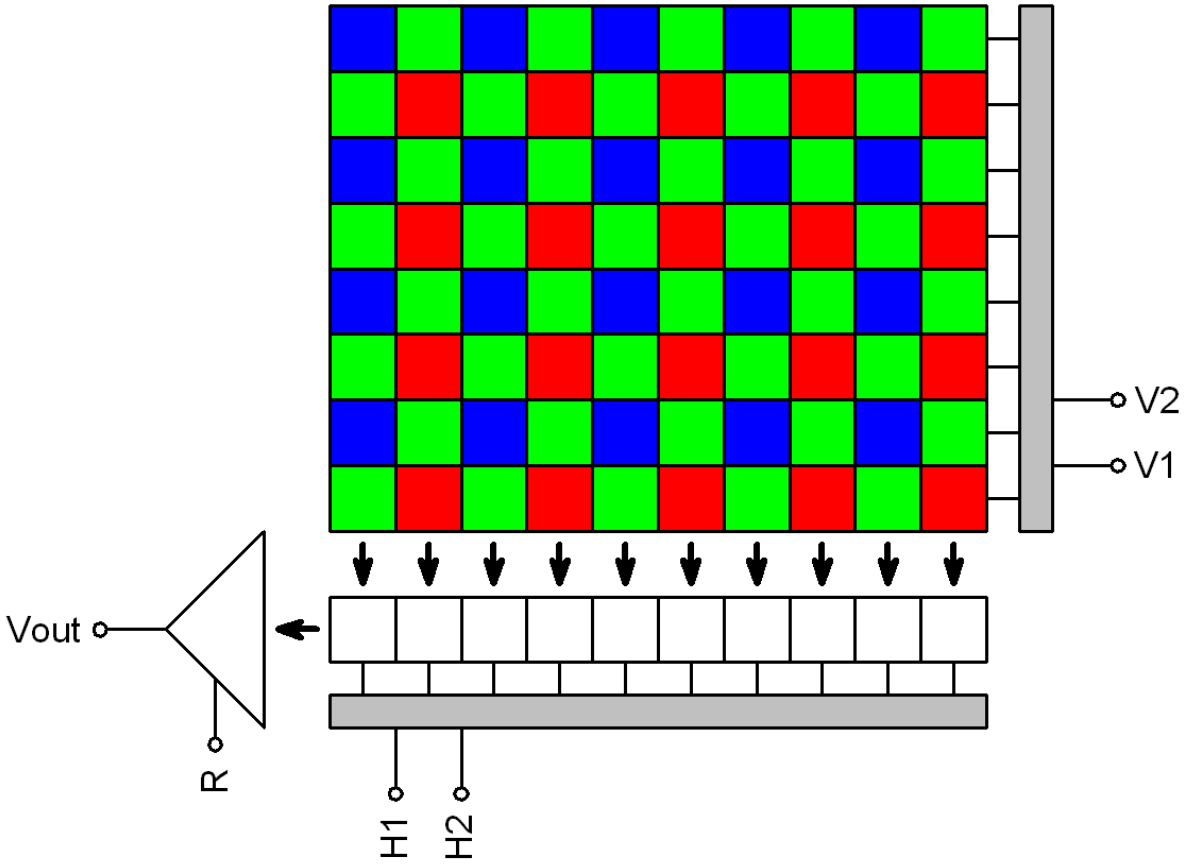
Human vision is logarithmic



Human vision is logarithmic



Mechanical "vision" is linear



The useful stuff is at the bottom

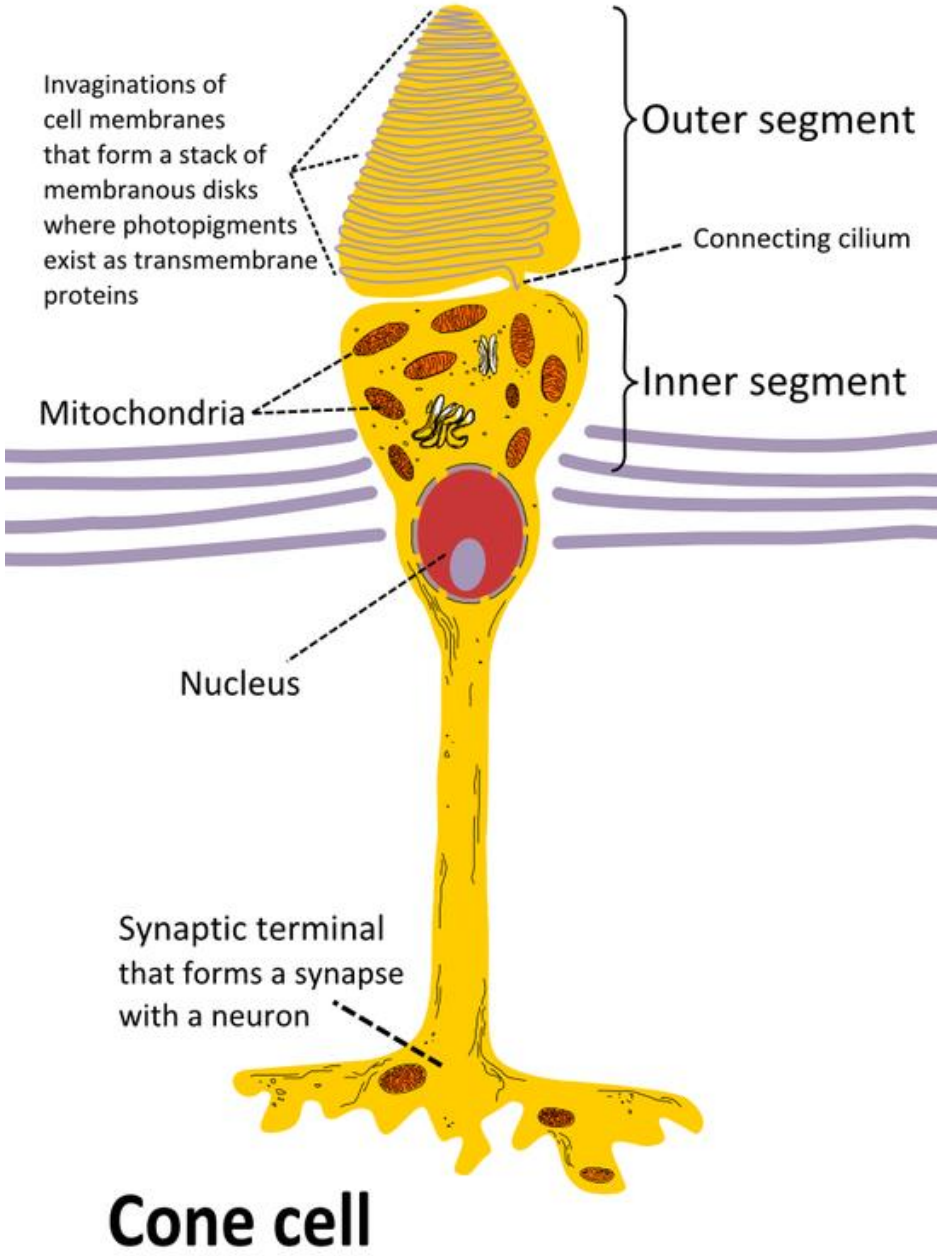
$2.0\sqrt{x}$ $2.0\sqrt{x}$ $2.0\sqrt{x}$ $2.0\sqrt{x}$ $2.0\sqrt{x}$ $2.0\sqrt{x}$

The useful stuff is at the bottom

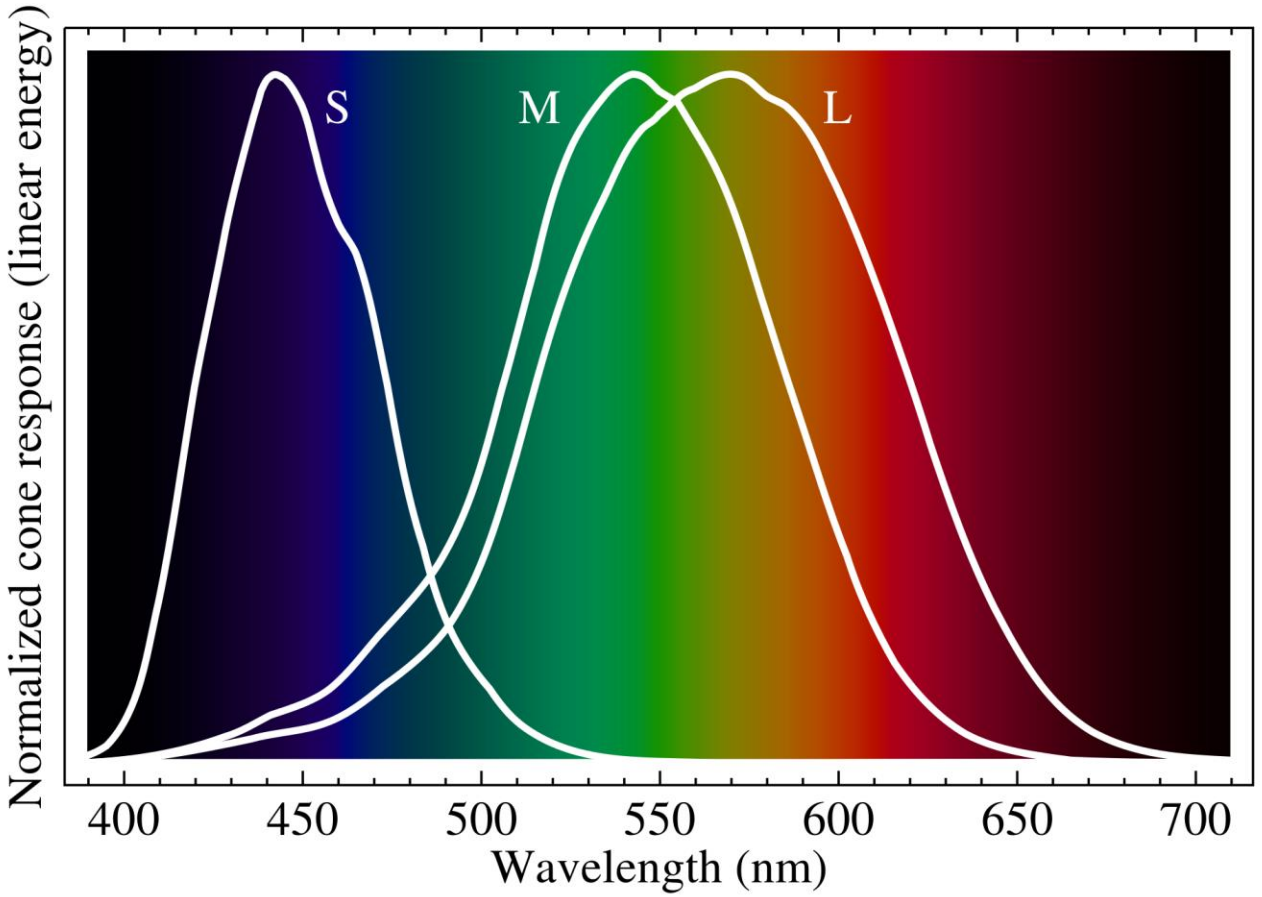
$2.2\sqrt{x}$ $2.2\sqrt{x}$ $2.2\sqrt{x}$ $2.2\sqrt{x}$ $2.2\sqrt{x}$ $2.2\sqrt{x}$

$1.8\sqrt{x}$ $1.8\sqrt{x}$ $1.8\sqrt{x}$ $1.8\sqrt{x}$ $1.8\sqrt{x}$ $1.8\sqrt{x}$

But what about colour?



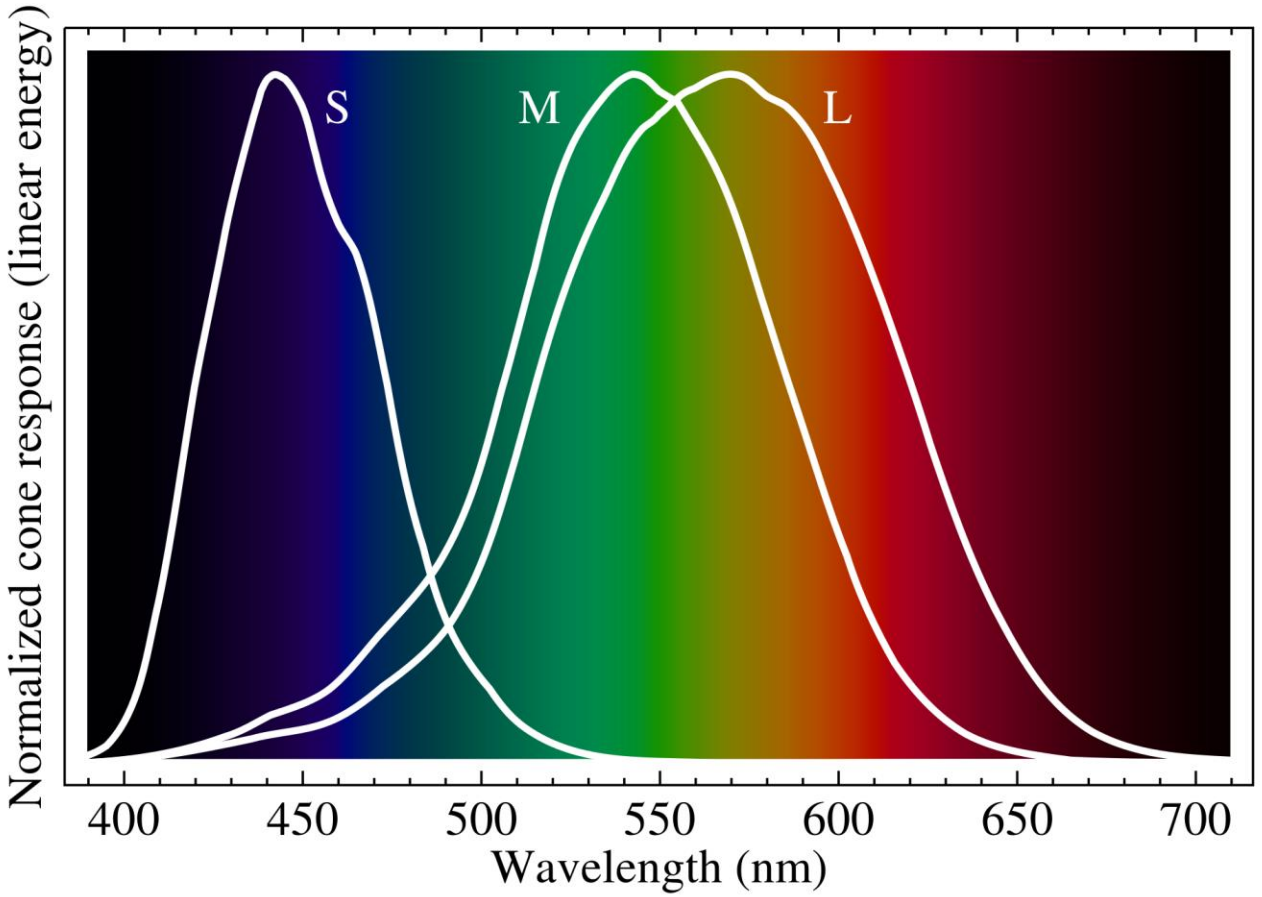
But what about colour?



The CIE 1931 colour spaces

- Take a standard human
- Put them in a standard environment
- Measure how they perceive electromagnetic waves, via matching the colours of lights (mixes of primaries)
- Build a function that maps electromagnetic wavelengths to human perception, giving 3 values (X, Y, Z)
- Add some mathematical constraints (values > 0, Y = relative luminance from 0 to 100)

But what about colour?



Chromaticity

- Humans separate colour from brightness
- Normalise:

  $x = X / (X + Y + Z)$

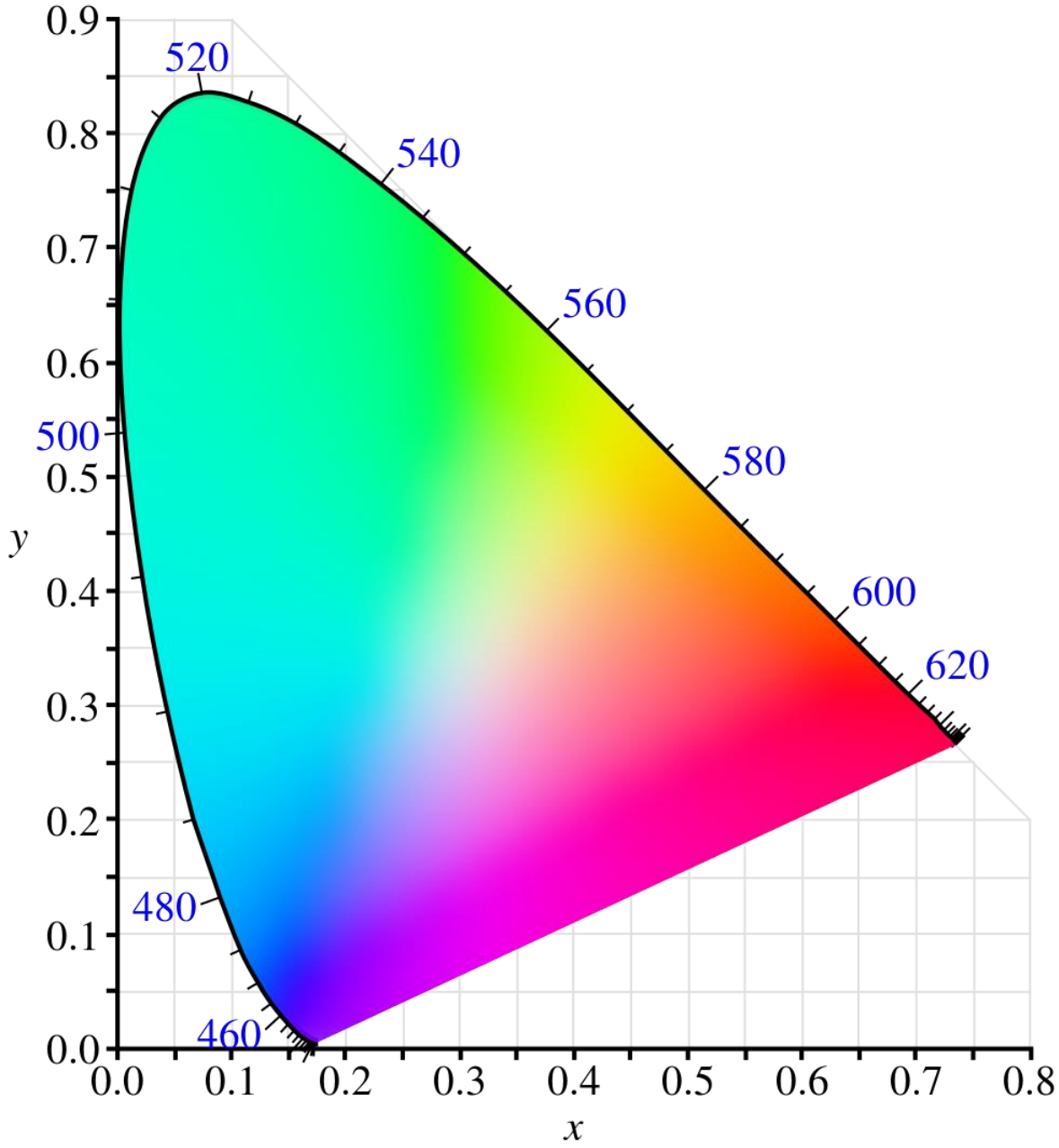
  $y = Y / (X + Y + Z)$

  $z = Z / (X + Y + Z) = (1 - x - y)$
- xyY colour space

  x and y are colour

  Y is relative luminance

The CIE 1931 colour space chromaticity diagram.



Perceptual uniformity

- Small change in a value has the same effect in perceived colour
- XYZ values are not perceptually uniform
- Inefficient, like storing sound volume in raw values rather than in dB. 100dB = 10^100

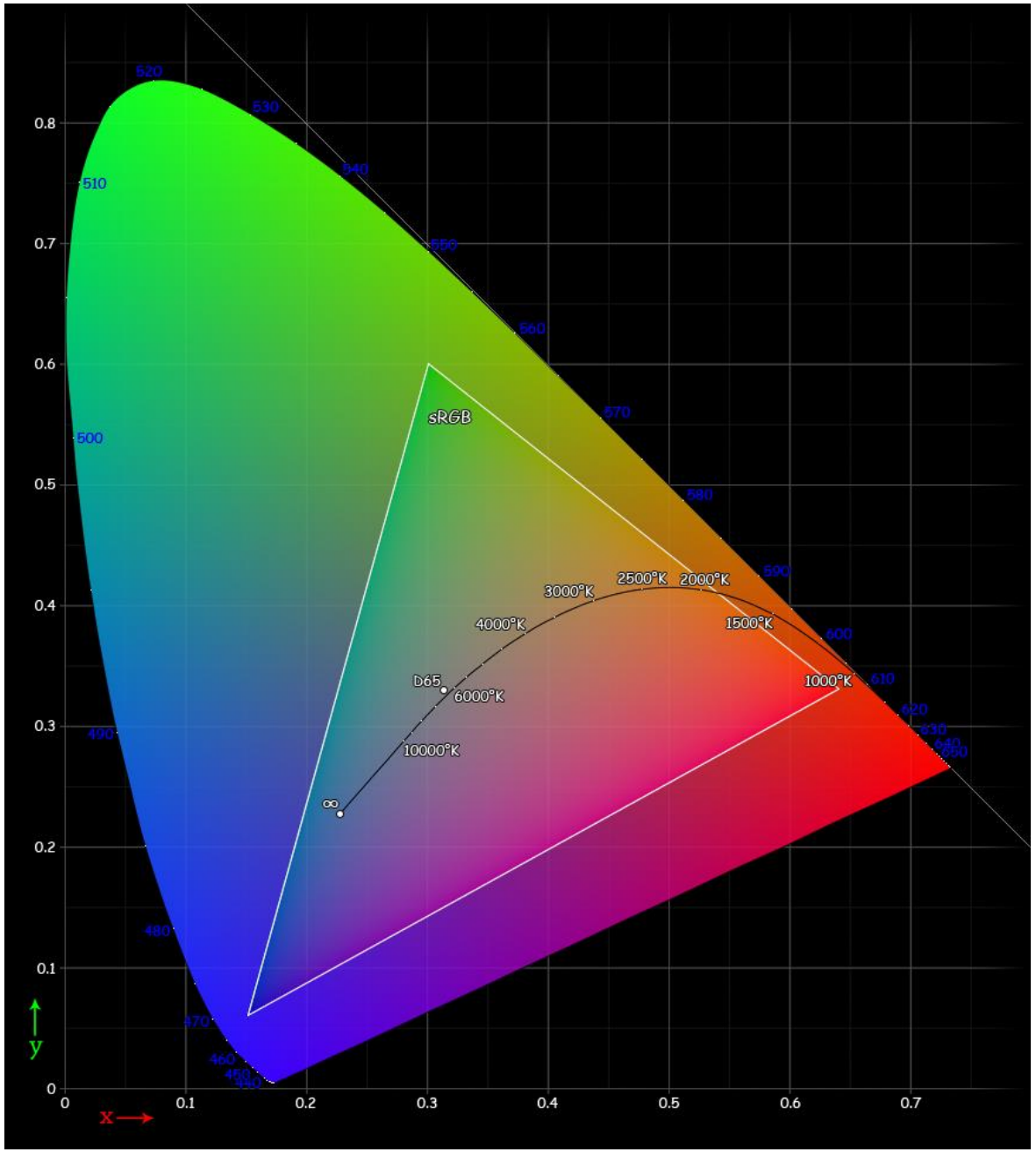
sRGB

- 1996: Microsoft + HP
- IEC 61966-2-1:1999
- Default colour space where NO COLOUR SPACE INFORMATION is provided

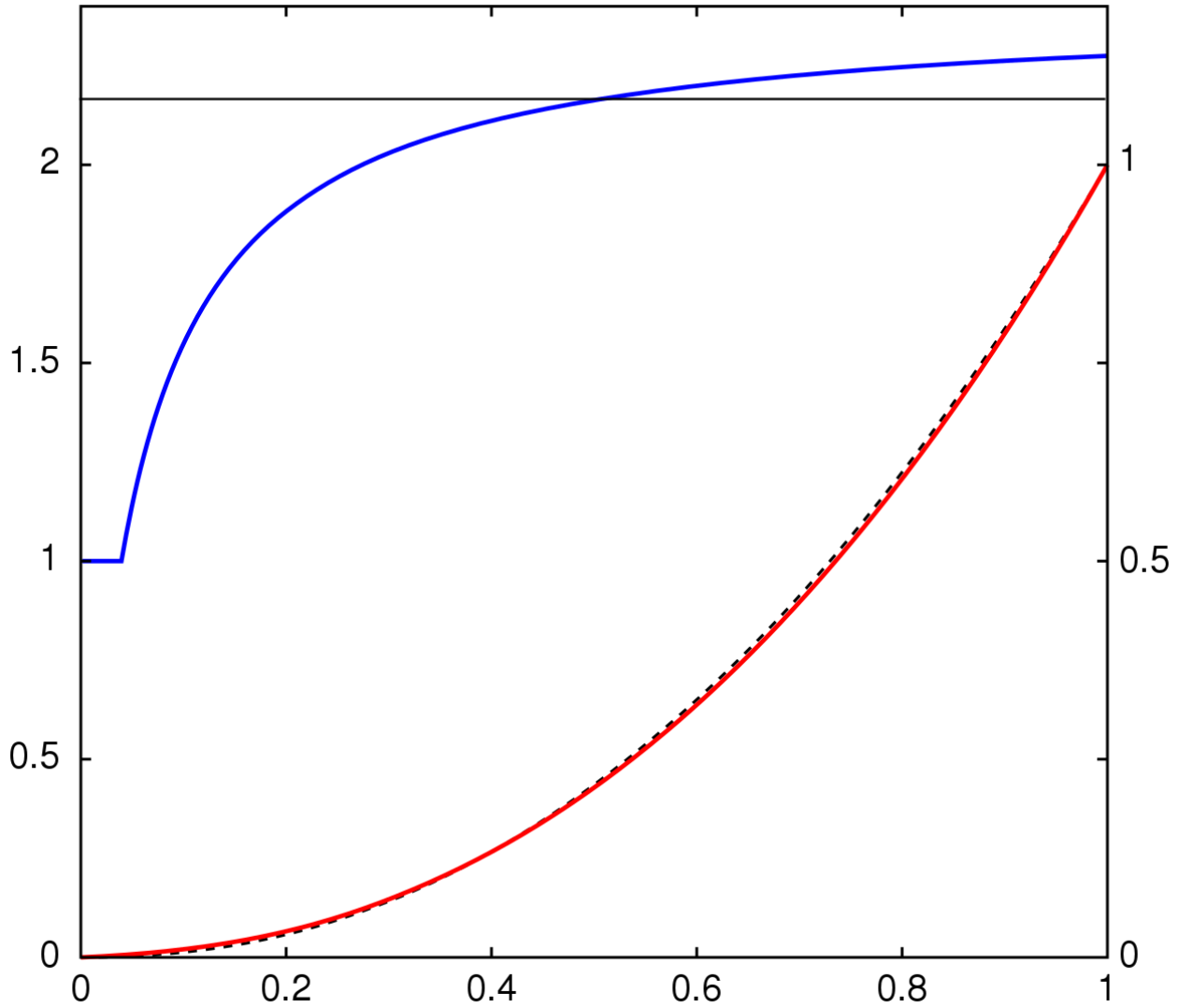
sRGB

Chromaticity	Red	Green	Blue	White point
x	0.6400	0.3000	0.1500	0.3127
y	0.3300	0.6000	0.0600	0.3290
Y	0.2126	0.7152	0.0722	1.0000

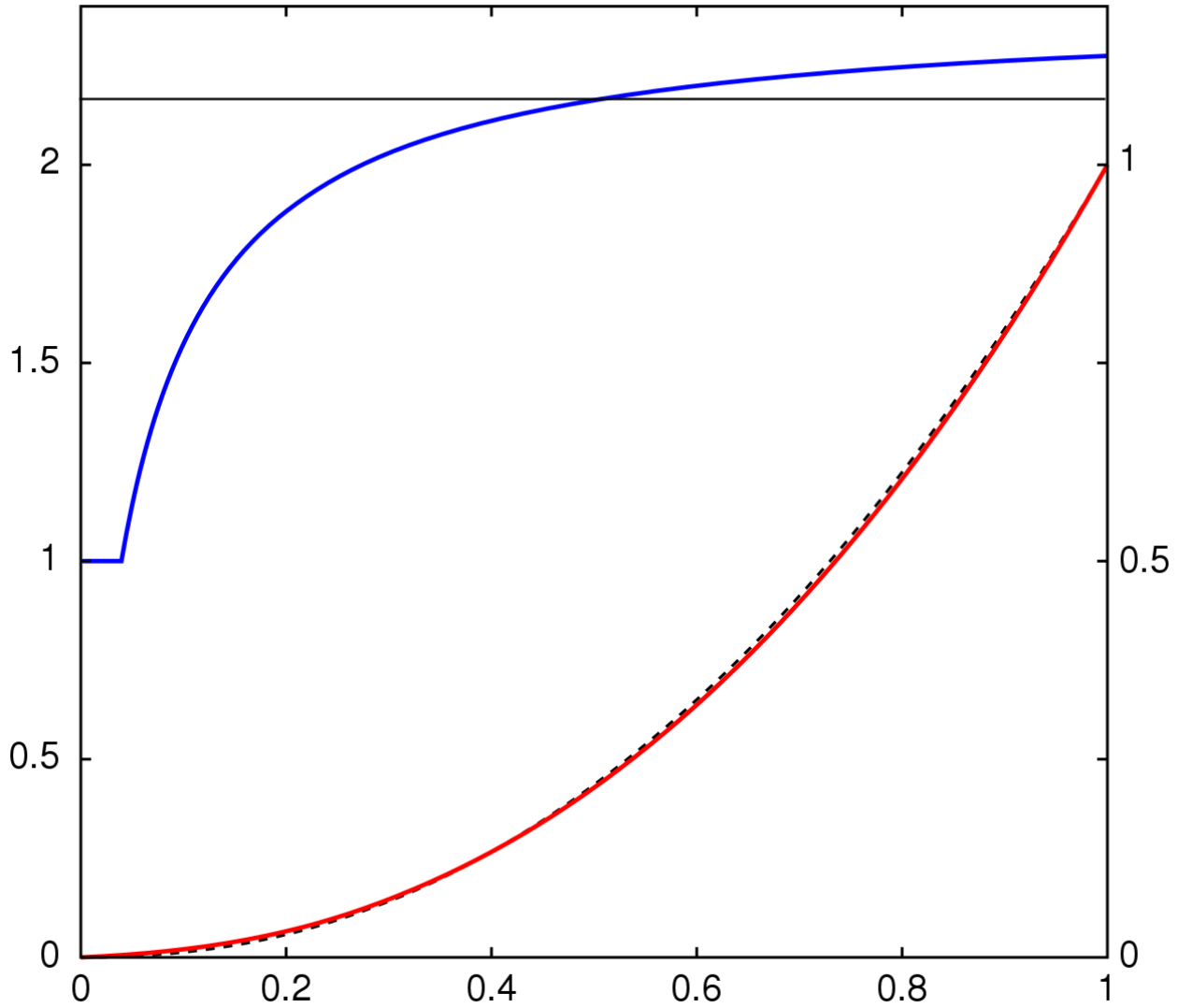
sRGB


sRGB
D65
1000°K
1500°K
2000°K
2500°K
3000°K
4000°K
6000°K
10000°K
∞
x
y

Transfer Function



Transfer Function



Transfer Function

$$\begin{bmatrix} R_{\text{linear}} \\ G_{\text{linear}} \\ B_{\text{linear}} \end{bmatrix} = \begin{bmatrix} +3.24096994 & -1.53738318 & -0.49861076 \\ -0.96924364 & +1.8759675 & +0.04155506 \\ +0.05563008 & -0.20397696 & +1.05697151 \end{bmatrix} \begin{bmatrix} X_{D65} \\ Y_{D65} \\ Z_{D65} \end{bmatrix}$$

$$\gamma(u) = \begin{cases} 12.92u & = \frac{323u}{25} & u \leq 0.0031308 \\ 1.055u^{1/2.4} - 0.055 & = \frac{211u^{\frac{5}{12}} - 11}{200} & \text{otherwise} \end{cases}$$

$$\gamma^{-1}(u) = \begin{cases} \frac{u}{12.92} & = \frac{25u}{323} & u \leq 0.04045 \\ \left(\frac{u+0.055}{1.055}\right)^{2.4} & = \left(\frac{200u+11}{211}\right)^{\frac{12}{5}} & \text{otherwise} \end{cases}$$

$$\begin{bmatrix} X_{D65} \\ Y_{D65} \\ Z_{D65} \end{bmatrix} = \begin{bmatrix} 0.41239080 & 0.35758434 & 0.18048079 \\ 0.21263901 & 0.71516868 & 0.07219232 \\ 0.01933082 & 0.11919478 & 0.95053215 \end{bmatrix} \begin{bmatrix} R_{\text{linear}} \\ G_{\text{linear}} \\ B_{\text{linear}} \end{bmatrix}$$

Implementation

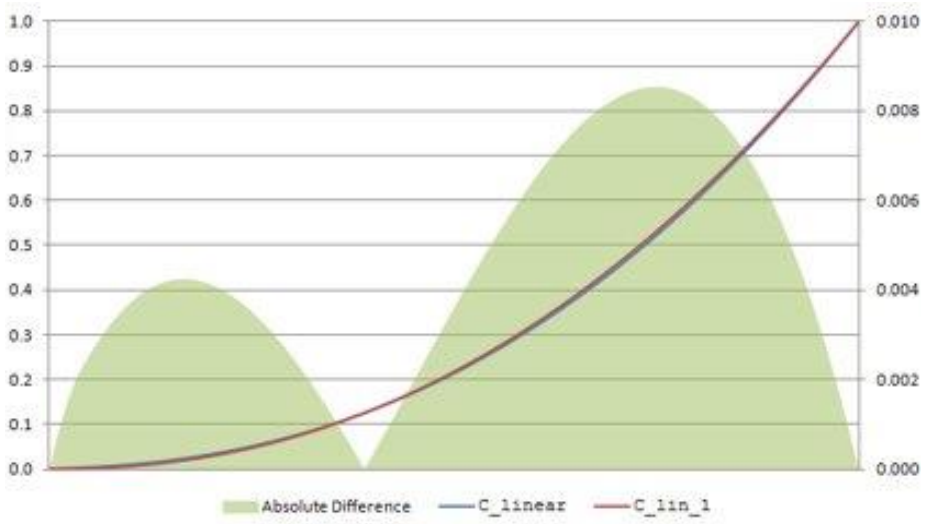
$$\gamma^{-1}(u) = \begin{cases} \frac{u}{12.92} & = \frac{25u}{323} & u \leq 0.04045 \\ \left(\frac{u+0.055}{1.055}\right)^{2.4} & = \left(\frac{200u+11}{211}\right)^{\frac{12}{5}} & \text{otherwise} \end{cases}$$

- if (C_srgb <= 0.04045)
  C_lin = C_srgb / 12.92;
  else
  C_lin = pow((C_srgb + 0.055) / 1.055, 2.4);

Implementation

$$\gamma^{-1}(u) = \begin{cases} \frac{u}{12.92} & = \frac{25u}{323} & u \leq 0.04045 \\ \left(\frac{u+0.055}{1.055}\right)^{2.4} & = \left(\frac{200u+11}{211}\right)^{\frac{12}{5}} & \text{otherwise} \end{cases}$$

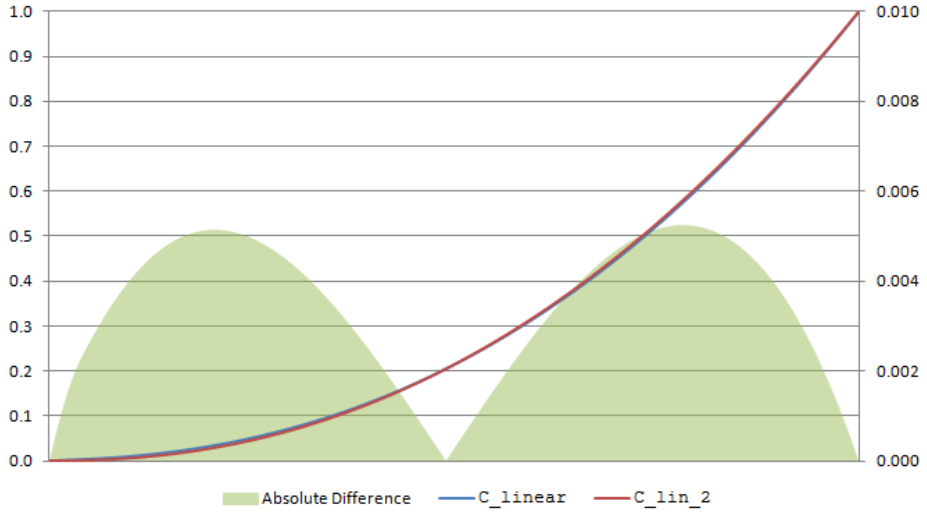
- C_lin_1 = pow(C_srgb, 2.2);



Implementation

$$\gamma^{-1}(u) = \begin{cases} \frac{u}{12.92} & = \frac{25u}{323} & u \leq 0.04045 \\ \left(\frac{u+0.055}{1.055}\right)^{2.4} & = \left(\frac{200u+11}{211}\right)^{\frac{12}{5}} & \text{otherwise} \end{cases}$$

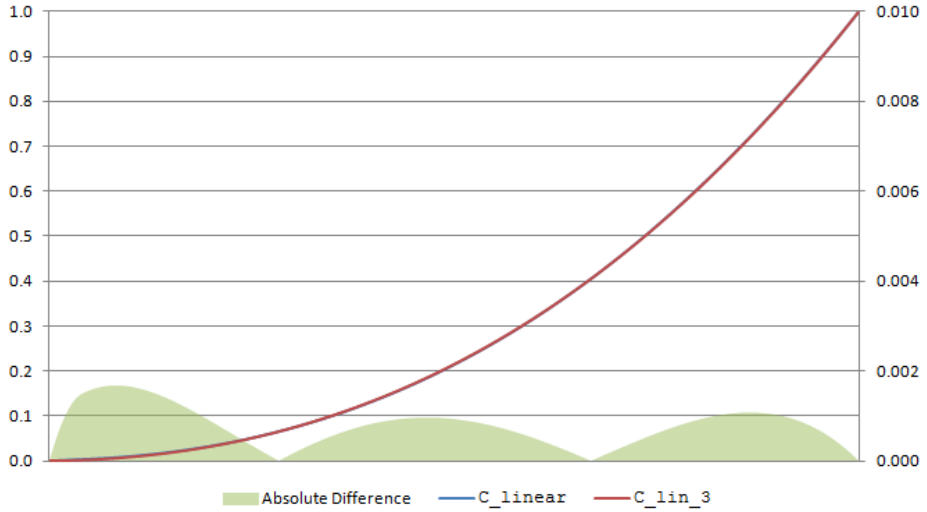
- C_lin_2 = pow(C_srgb, 2.233333333);



Implementation

$$\gamma^{-1}(u) = \begin{cases} \frac{u}{12.92} & = \frac{25u}{323} & u \leq 0.04045 \\ \left(\frac{u+0.055}{1.055}\right)^{2.4} & = \left(\frac{200u+11}{211}\right)^{\frac{12}{5}} & \text{otherwise} \end{cases}$$

- C_lin_3 = 0.012522878 * C_srgb +
  0.682171111 * C_srgb * C_srgb +
  0.305306011 * C_srgb * C_srgb * C_srgb;



Implementation

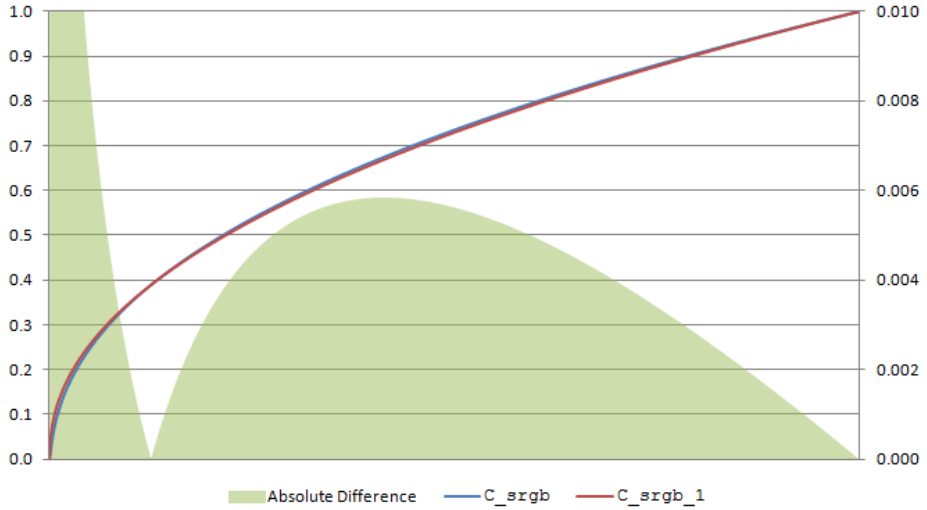
$$\gamma(u) = \begin{cases} 12.92u & = \frac{323u}{25} & u \leq 0.0031308 \\ 1.055u^{1/2.4} - 0.055 & = \frac{211u^{\frac{5}{12}} - 11}{200} & \text{otherwise} \end{cases}$$

- if (C_lin <= 0.0031308)
  C_srgb = C_lin * 12.92;
  else
  C_srgb = 1.055 * pow(C_lin, 1.0 / 2.4) - 0.055;

Implementation

$$\gamma^{-1}(u) = \begin{cases} \frac{u}{12.92} & = \frac{25u}{323} & u \leq 0.04045 \\ \left(\frac{u+0.055}{1.055}\right)^{2.4} & = \left(\frac{200u+11}{211}\right)^{\frac{12}{5}} & \text{otherwise} \end{cases}$$

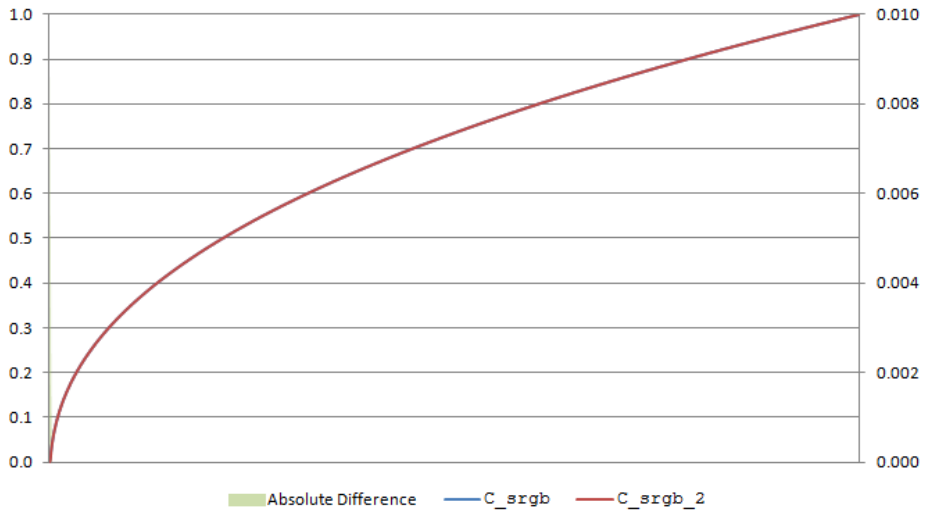
- C_srgb_1 = pow(C_lin, 0.4545454545);



Implementation

$$\gamma^{-1}(u) = \begin{cases} \frac{u}{12.92} & = \frac{25u}{323} & u \leq 0.04045 \\ \left(\frac{u+0.055}{1.055}\right)^{2.4} & = \left(\frac{200u+11}{211}\right)^{\frac{12}{5}} & \text{otherwise} \end{cases}$$

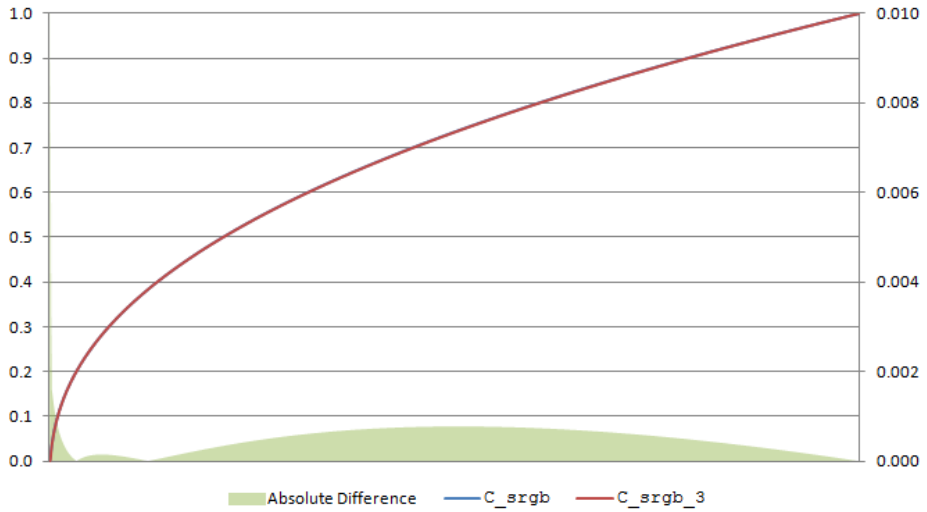
- C_srgb_2 = max(1.055 * pow(C_lin, 0.416666667) - 0.055, 0);



Implementation

$$\gamma^{-1}(u) = \begin{cases} \frac{u}{12.92} & = \frac{25u}{323} & u \leq 0.04045 \\ \left(\frac{u+0.055}{1.055}\right)^{2.4} & = \left(\frac{200u+11}{211}\right)^{\frac{12}{5}} & \text{otherwise} \end{cases}$$

- C_srgb_3 = 0.585122381 * sqrt(C_lin) +
  0.783140355 * sqrt(sqrt(C_lin)) -
  0.368262736 * sqrt(sqrt(sqrt(C_lin)));



Summary

- Brightness perception is logarithmic
- XYZ defines absolute perceptual colours
- The xyY colourspace is linear
- Linear interpolation is valid on linear colourspaces
- sRGB is defined relative to xyY
- The transfer function is non-linear and expensive
- sRGB is non-linear
- Linear interpolation is invalid on sRGB

Part 2

Getting it wrong

Interpolation

- (x + y) / 2
- (√x + √y) / 2 < √((x + y) / 2)
- x = 9, y = 16
- (√9 + √16) / 2 = 3.5
- √((9 + 16) / 2) = 3.535
- template <class T> constexpr std::midpoint(T a, T b) noexcept;
- constexpr float std::lerp(float a, float b, float t) noexcept;

Interpolation

```
0 0 0 0 0 0 0 13 13 13 13 13 13 13 13 13 13 13 22 22 22 22 22 22 22 22 28 28 28 28 28 28 34 34 34 34 34 38 38 38 38 42 4
2 42 42 46 46 46 50 50 50 50 53 53 53 56 56 56 59 59 61 61 61 64 64 64 66 66 69 69 71 71 73 73 73 75 75 77 77 79 79 81 8
3 83 85 85 86 86 88 88 90 92 92 93 95 95 96 96 98 99 99 101 102 104 104 105 106 106 108 109 110 112 112 113 114 115 117
117 118 119 120 121 122 124 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146
147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176
177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206
207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236
237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255
Max error 6
Total error 127
```

Posterisation

SDL

- https://www.libsdl.org
- 8-bit colour type
- sRGB mode
- No documentation

SFML

- https://www.sfml-dev.org
- 8-bit colour type
- sRGB mode
- Operator overloading on colours
- No documentation

Dear ImGui

- https://github.com/ocornut/imgui
- Alpha blending is wrong by default
- sRGB and linear colour are interchangeable internally
- 32-bit colour API
- No documentation

Flash

- No mention of gamma correction
- 8-bit sRGB used like linear colours
- No documentation

Unity

- https://unity.com
- No documentation on the colour space
- If sRGB, lerp and operators are incorrect
- If linear, constants and HSVtoRGB are incorrect
- If both, toggle is a runtime choice, default unspecified

Godot

- https://godotengine.org
- No documentation on the colour space
- If sRGB, lerp, blending and grey are incorrect
- If linear, constructors and converters are incorrect
- Grey is incorrect
- Lighten/darken imply something different to what is being done

OGRE

- https://www.ogre3d.org
- No documentation on the colour space
- If sRGB, operators are incorrect
- If linear, converters are incorrect

CRYENGINE

- https://www.cryengine.com
- Color_tpl<T> (T = float or T = std::uint8_t)
- float*, std::uint8_t*
- vec3/vec4
- float r, float g, float b, float a

CRYENGINE

- Color_tpl<T> has arithmetic operators which can never be correct for T=std::uint8_t
- Other operators will quantise when T=std::uint8_t
- Color_tpl<std::uint8_t> can never store linear colours
- No documentation

CRYENGINE

- If sRGB:
- Arithmetic operators are incorrect
- srgb2rgb is lossy for Color_tpl<std::uint8_t>
- ScaleCol is probably incorrect
- lerpFloat will give unexpected results
- Grey is probably incorrect

CRYENGINE

- If linear:
- Color_tpl<T> provides incorrect packing operators
- To/fromHSV is incorrect

CRYENGINE

- Luminance calculation is incorrect one way or the other
- No canonical representation
- Some documentation is wrong rather than absent

Qt

- https://www.qt.io
- QColor
- qcolorspace
- qrgb

MatLab

- https://uk.mathworks.com/products/matlab.html
- Luminance calculation is incorrect (assumes NTSC)

OpenCV

- https://opencv.org
- Luminance calculation is incorrect (assumes NTSC)

SVG and CSS

- https://observablehq.com/@mootari/color-blending-is-broken

Alacritty

- https://github.com/alacritty/alacritty
- Subpixel fonts are rendered incorrectly

MSYS2

- https://www.msys2.org
- Subpixel fonts are rendered incorrectly

Windows console

- **Subpixel fonts are rendered incorrectly**

Microsoft terminal

- **Works correctly in cleartype mode!**

UE4

- https://www.unrealengine.com/en-US
- Linear by default
- FColor
- FLinearColor
- Appropriate operators
- WITH DOCUMENTATION!!!

Miscellaneous

- Linux and X-Windows
- OpenGL
- Linux Nvidia bug

P1385

- https://wg21.link/P1385

Goals

- Provide linear algebra vocabulary types
- Parameterise orthogonal aspects of implementation
- Defaults for the 90%, customisable for power users
- Element access, matrix arithmetic, fundamental operations
- Mixed precision and mixed representation expressions

Linear Algebra for beginners

- "The branch of mathematics concerning linear equations and linear functions, and their representation through matrices and vector spaces"
- $a_1x_1 + a_2x_2 + \ldots + a_nx_n = b$
- Geometry
- Linear regression
- Simultaneous equations

Vector

- $[a_1, a_2, a_3 \dots a_n]$
- $[a_1, a_2, a_3 \dots a_n] + [b_1, b_2, b_3 \dots b_n]$
  $= [a_1+b_1, a_2+b_2, a_3+b_3 \dots a_n+b_n]$
- $b * [a_1, a_2, a_3 \dots a_n]$
  $= [ba_1, ba_2, ba_3 \dots ba_n]$
- $[a_1, a_2, a_3] . [b_1, b_2, b_3]$
  $= a_1b_1+a_2b_2+a_3b_3$

Matrix

- $[a_{11}, a_{12}, a_{13} \dots a_{1n}]$
  $[a_{21}, a_{22}, a_{23} \dots a_{2n}]$
  $[a_{31}, a_{32}, a_{33} \dots a_{3n}]$
- $[a_{11}, a_{12}, a_{13} \dots a_{1n}] \quad [b_{11}, b_{12}, b_{13} \dots b_{1n}]$
  $[a_{21}, a_{22}, a_{23} \dots a_{2n}] + [b_{21}, b_{22}, b_{23} \dots b_{2n}]$
  $[a_{31}, a_{32}, a_{33} \dots a_{3n}] \quad [b_{31}, b_{32}, b_{33} \dots b_{3n}]$

$$[a_{11}+b_{11}, a_{12}+b_{12}, a_{13}+b_{13} \dots a_{1n}+b_{1n}]$$
$$= [a_{21}+b_{21}, a_{22}+b_{22}, a_{23}+b_{23} \dots a_{2n}+b_{2n}]$$
$$[a_{31}+b_{31}, a_{32}+b_{32}, a_{33}+b_{33} \dots a_{3n}+b_{3n}]$$

Matrix

- $b * [a_{11}, a_{12}, a_{13} \dots a_{1n}] \quad [ba_{11}, ba_{12}, ba_{13} \dots ba_{1n}]$
  $[a_{21}, a_{22}, a_{23} \dots a_{2n}] = [ba_{21}, ba_{22}, ba_{23} \dots ba_{2n}]$
  $[a_{31}, a_{32}, a_{33} \dots a_{3n}] \quad [ba_{31}, ba_{32}, ba_{33} \dots ba_{3n}]$

BLAS

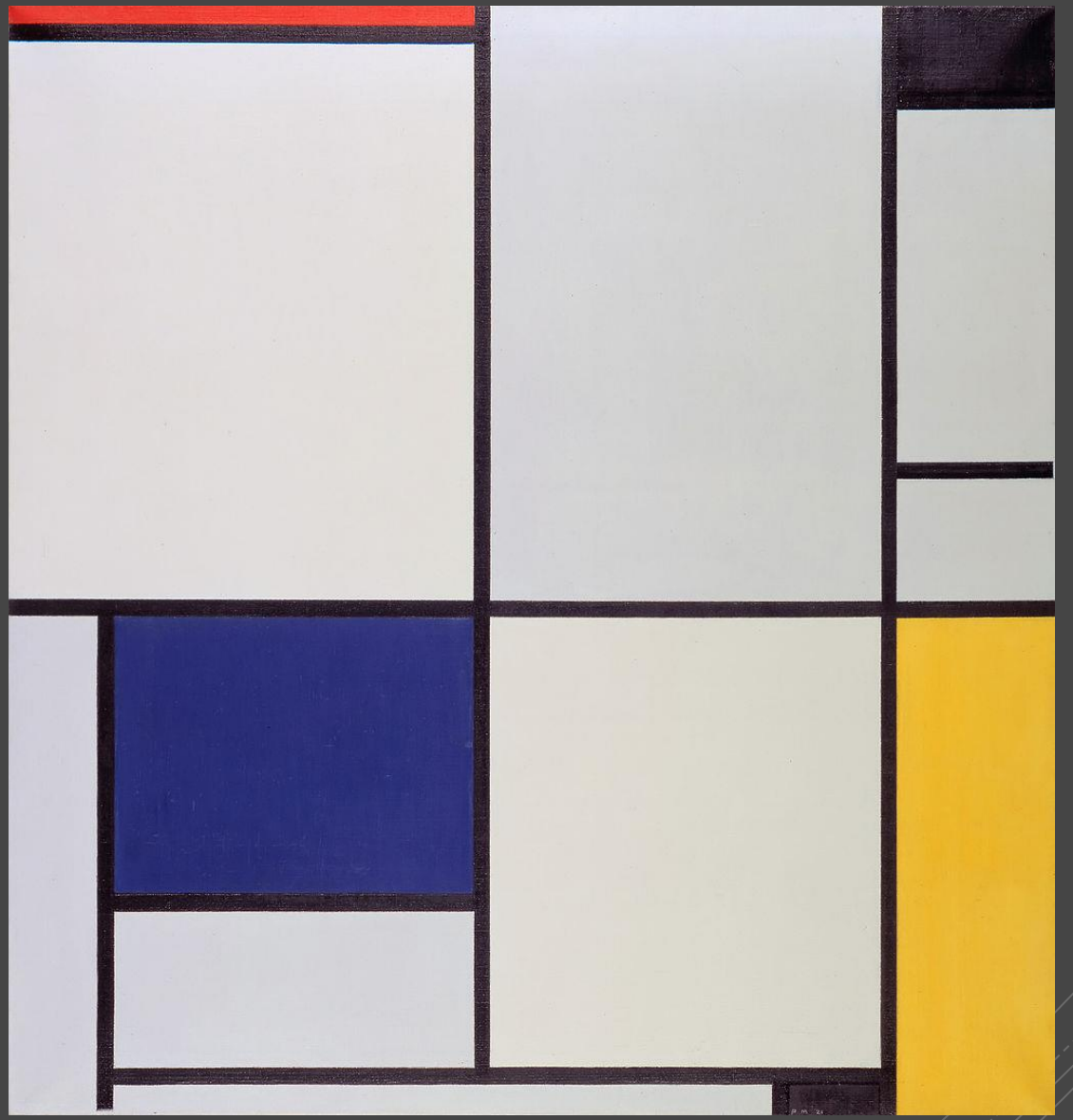
- https://wg21.link/P1673
- Add BLAS to the standard library as a C++ API
- First introduced in 1979
- C++23

Mixed representation

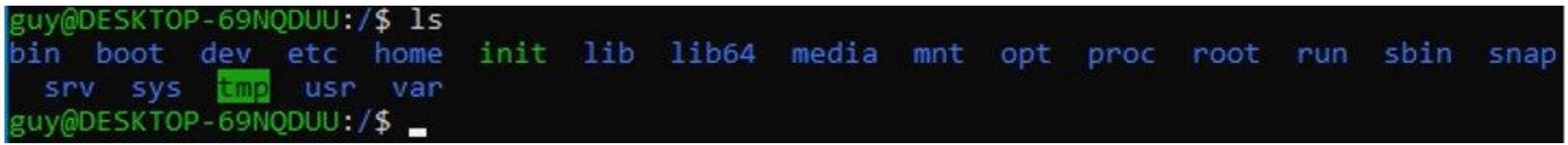
- 0.25 + 0.125f = 0.375
- complex<float>{0.1, 0.1} + complex<double>{0.2, 0.2}
- vector<double, 3> * matrix<float, 3, 3>

Alternative algorithms

- **Multiplication**
- **$O(n^3)$**
- **Strassen – $O(n^{2.807})$**



Why?


guy@DESKTOP-69NQDUU:/$ ls
bin boot dev etc home init lib lib64 media mnt opt proc root run sbin snap
srv sys tmp usr var
guy@DESKTOP-69NQDUU:/$

Wrong

Why teach colour to C++?

Use cases

- std::fmt
- Drawing software
- Graphics proposal

Requirements

- XYZ support
- Conversion between XYZ-derived colour spaces
- Compile-time user-defined colour spaces
- Runtime data-defined colour spaces
- Allow ICC profile implementation
- Strongly typed
- High performance without storage or speed overhead

Current status

- struct basic_color_space
- struct generic_RGB_space : basic_color_space
- struct XYZ_space : basic_color_space

Current status

- struct basic_color_model
- struct XYZ_model : basic_color_model

```
{
  float X = 0.f;
  float Y = 0.f;
  float Z = 0.f;
};
```

Current status

```
template<typename V1, typename V2, typename V3>
struct RGB_model : basic_color_model
{
  typename V1::type r = typename V1::type();
  typename V2::type g = typename V2::type();
  typename V3::type b = typename V3::type();

  using R_value = V1;
  using G_value = V2;
  using B_value = V3;
...
};
```

Current status

```
template<typename VA>
struct alpha_model
{
   typename VA::type a = typename VA::type();

   using A_value = VA;
...
};
```

Current status

```
template<typename cspace,
         typename cmodel,
         typename calpha>
struct basic_color : cmodel, calpha
{
    using space_type = cspace;
    using model_type = cmodel;
    using alpha_type = calpha;
    using cmodel::cmodel;
    using calpha::calpha;
};
```

Current status

- struct sRGB_uint8 :
  basic_color<sRGB_space,
  RGB_uint8_model,
  no_alpha>

Current status

- struct sRGB_float :
  basic_color<sRGB_space,
  RGB_float_model,
  no_alpha>

Current status

- struct sRGBA_uint8 :
  basic_color<sRGB_space,
  RGB_uint8_model,
  uint8_alpha>

Current status

- struct sRGBA_float :
  basic_color<sRGB_space,
  RGB_float_model,
  float_alpha>

Current status

- struct linear_sRGB_float :
  basic_color<linear_sRGB_space,
  RGB_float_model,
  no_alpha>

Current status

- struct linear_sRGBA_float :
  basic_color<linear_sRGB_space,
  RGB_float_model,
  float_alpha>

Current status

- struct XYZ :
  basic_color<XYZ_space,
  XYZ_model,
  no_alpha>

Current status

- template <typename T1, typename T2>
  constexpr void
  alpha_convert(const alpha_model<T1>& in,
  alpha_model<T2>& out);

Current status

- template <typename T1, typename U1, typename V1,
  typename T2, typename U2, typename V2>
  constexpr void
  model_convert(const RGB_model<T1, U1, V1>& in,
  RGB_model<T2, U2, V2>& out);

Current status

```
template <typename space_1, typename model_1,
          typename gamma_1, typename alpha_1,
          typename space_2, typename model_2,
          typename gamma_2, typename alpha_2>
constexpr void
color_convert(const basic_color<
          generic_RGB_space<space_1, gamma_1>,
          model_1, alpha_1>& in,
           basic_color<
          generic_RGB_space<space_2, gamma_2>,
          model_2, alpha_2>& out);
```

Current status

```
template <typename space, typename model,
          typename trans,
          typename alpha_1, typename alpha_2>
constexpr void
color_convert(const basic_color<
          generic_RGB_space<space, trans>,
          model, alpha_1>& in,
           basic_color<
          XYZ_space,
          XYZ_model, alpha_2>& out);
```

Current status

- template <typename space, typename model,
  typename trans,
  typename alpha_1, typename alpha_2>
  constexpr void
  color_convert(const basic_color<
  XYZ_space,
  XYZ_model, alpha_1>& in,
  basic_color<
  generic_RGB_space<space, trans>,
  model, alpha_2>& out);

Current status

- template <typename destination,
  typename source,
  typename...T>
  constexpr destination
  convert(const source& in,T&&...args);

Summary

- Identify colours
- Apprehend intensity and colour
- CIE1931 linear colour space
- SRGB non-linear colour space
- Transfer function
- Misapplication of colour management
- Linear algebra
- Uses of colour for C++
- Proposed API

Everything you knew about colour was wrong

J Guy Davidson