颜色

光和颜色

什么是光

> 光通常指的是人类眼睛可以见的电磁波（可见光），视知觉就是对于可见光的知觉。可见光只是电磁波谱上的某一段频谱，一般是定义为波长介于400至700奈（纳）米（nm）之间的电磁波，也就是波长比紫外线长，比红外线短的电磁波。有些资料来源定义的可见光的波长范围也有不同，较窄的有介于420至680nm，较宽的有介于380至800nm。……光既是一种高频的电磁波，又是一种由称为光子的基本粒子组成的粒子流。因此光同时具有粒子性与波动性，或者说光具有"波粒二象性"。
>
> ——From wiki

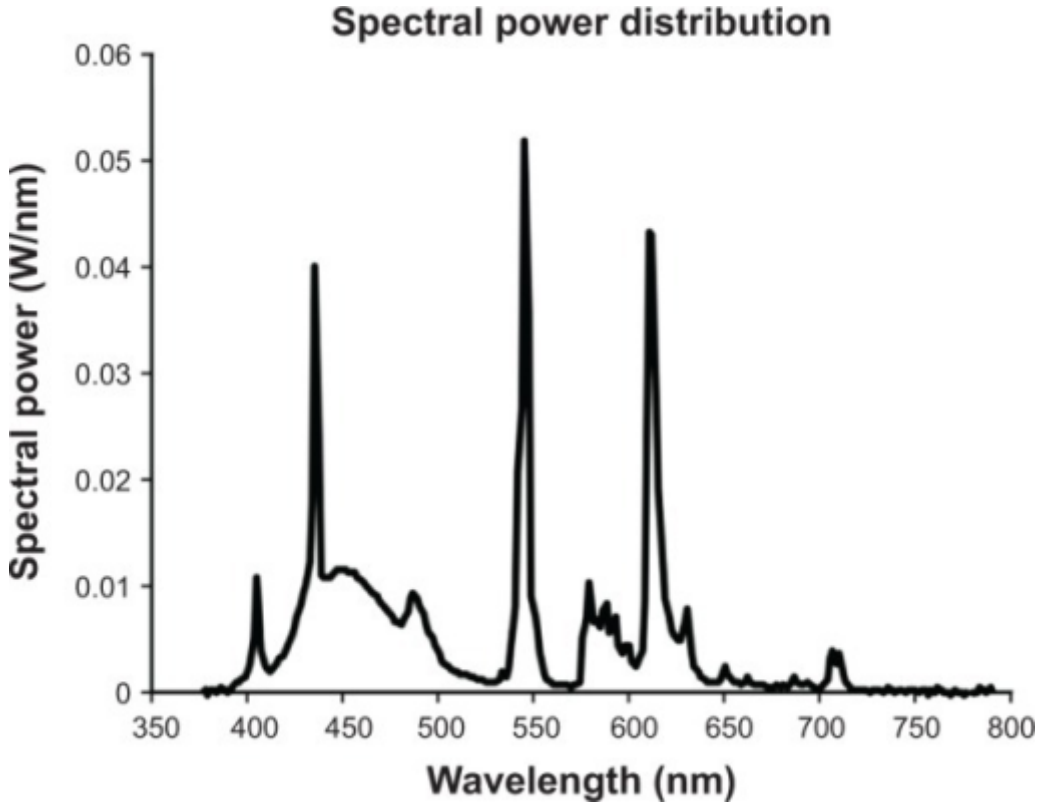
简洁来说，光是一种能量，可以以电磁波的形式传递。将复色光通过棱镜等色散系统分光后，按照光的波长或者频率大小依次排列成图案，就得到了光谱。

光谱中只有一部分的光是可见的，波长大概在400nm到750nm之间，这部分为成为可见光。

那什么是颜色呢?

> 颜色是眼、脑和我们的生活经验对光的颜色类别描述的视觉感知特征，其名称有红色、橙色、黄色、绿色、蓝色或紫色。这种对颜色的感知来自可见光谱中的电磁辐射对人眼视锥细胞的刺激。颜色的种类和颜色的物理规格是通过反射光的波长与物体相联系的。这种反射是由物体的物理性质决定的，如光的吸收、发射光谱等。但人对颜色的感觉不仅仅由光的物理性质所决定，还包含心理等许多因素，比如人类对颜色的感觉往往受到周围颜色的影响。
> —— From wiki

一种表示光的方式为——光谱能量分布图。横坐标为波长，纵坐标为波长的单位能量



所以姑且可以认为，不同波长的光就是我们看到的颜色

人眼怎么看到颜色

简单来说，人眼在视网膜上感知光的细胞有两种

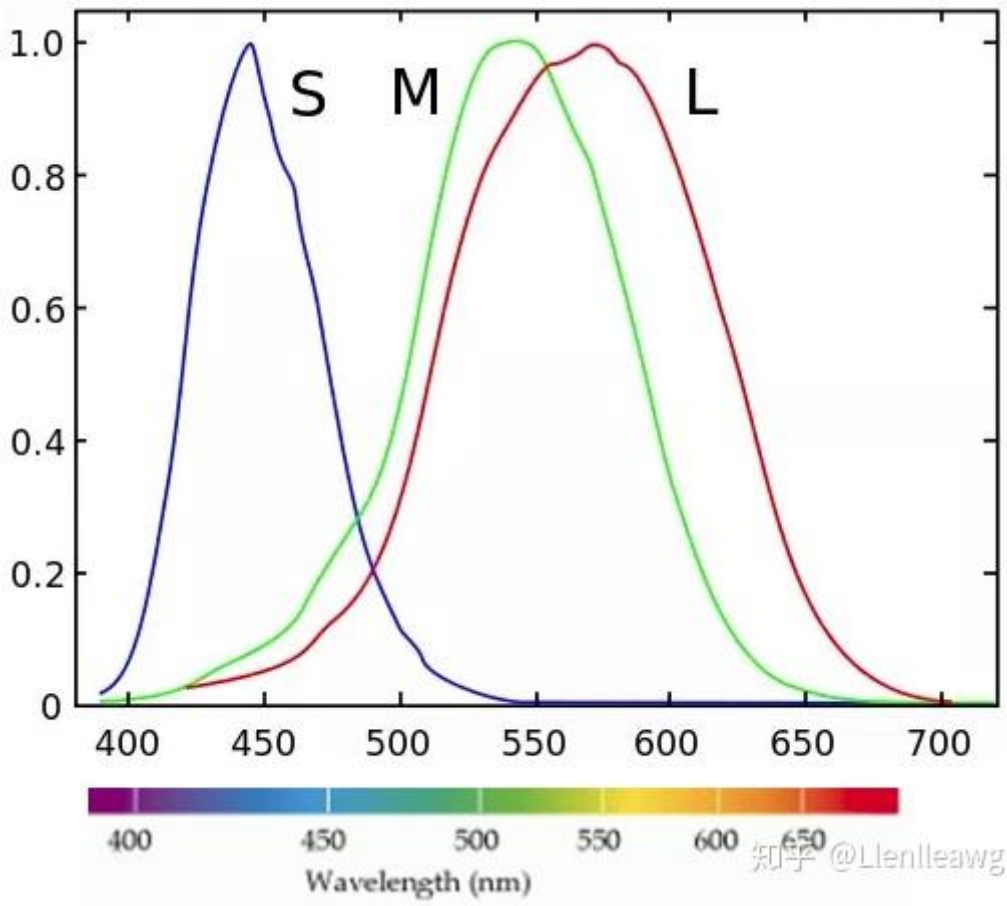
——视杆细胞，用来感知亮度。

——视锥细胞，用来感知颜色。

其中，视锥细胞有三个分类，分别为**S-视锥细胞**，**M-视锥细胞**，**L-视锥细胞。**分别对应着波长的短中长。

一般来说，S-视锥细胞对波长420nm的光线最为敏感，M-视锥细胞530nm的波长最为敏感，而L-视锥细胞对于560nm的波长最为敏感。

下图是三种细胞对不同波长的敏感程度，横坐标为波长，纵坐标为敏感度

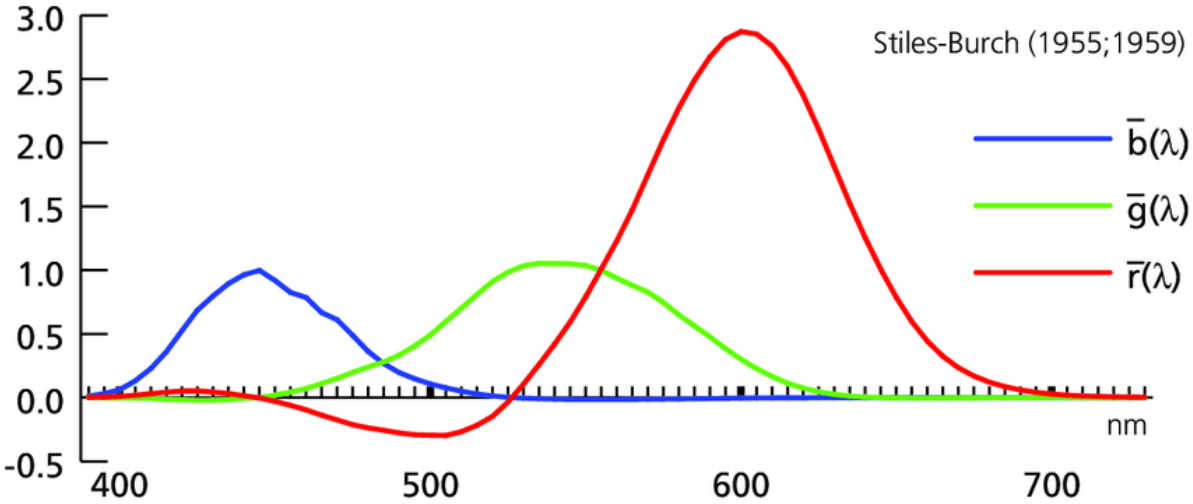


也就是说，光的光谱无论多么复杂，人眼只对光谱中的特定三个波长的敏感，而”颜色“也是这三种波长复合而来。由此可得，即便有两个光谱能量分布完全不一样的复色光，在我们人眼看来它们可能是同样的“颜色”，只要它们对视锥细胞产生的刺激是一致的。

量化颜色：Color Matching Functions

用光谱能量分布图表示颜色，需要记录的信息太多，于是受人眼启发，人们选取了三个单波长光，来表示可见光。

这就是**等色匹配实验（color matching experiment**选取三个单波长光，把它们称为primaries（三元光）。对于可见光谱中的每一个单波长光，找出三个参数分别对应每个三元光的强度，使得人眼对它们的颜色感知是一致的。这样对可见光谱中的所有单波长光进行测量，我们就可以得到一组函数r(λ)、g(λ)、b(λ)，把它们画到 一张表上就是：

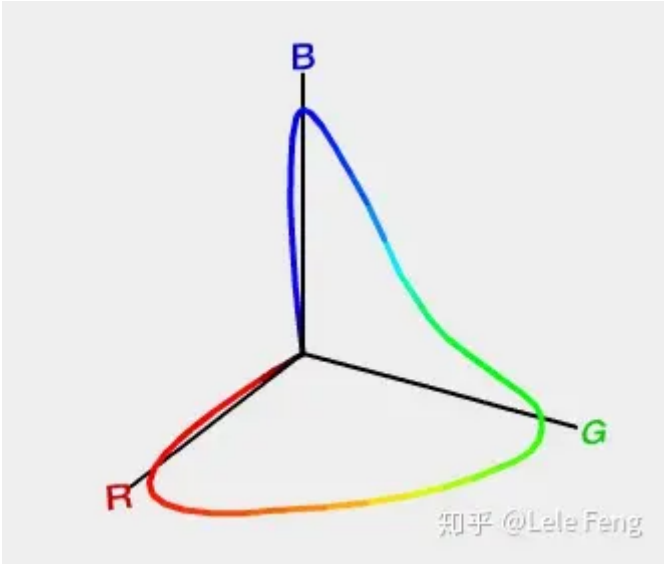


> 如：450nm的单色光约等于**1.0*B*(λ)+*0*G(λ)+0*R(λ)**

历史上做了很多这样的等色匹配实验，其中最著名也是应用最广泛（之一）的是1931年的CIE 1931 RGB Color Matching Functions。CIE（International Commission on Illumination，那为啥缩写不是COI而是CIE呢，因为缩写来源于它的法语名2333）是一个1913年成立的组织，致力于制定国际通用的和光与颜色相关的标准。CIE 1931 RGB使用的三元光波长分别是700nm（R）、546.1nm（G）、435.8nm（B）。

> 可以注意到，700nm（R）的波长会取到负值，这是由于，在实验室有些波长不可以通过三元光*正系数得到。*
> *如 待测试的波长为A， 无论怎么调整RGB的系数都得不到A，这时就会通过让A加上一个波长B即 A+B = xr(λ)+yg(λ)+zb(λ);*随后把B移过去，有时就会产生负值。

虽然color matching functions表示的是对单一波长的可见光源的测量，但由于光的叠加性，我们可以对任意频谱的光源做这样的一个匹配函数。如果把RGB分别作为三维空间的三个坐标轴，再归一化后，把CIE 1931 color matching functions的三个函数依次画到这个三维空间里，我们可以得到一条三维空间里的彩色线条，这条曲线就被称为光谱轨迹（spectral locus）：

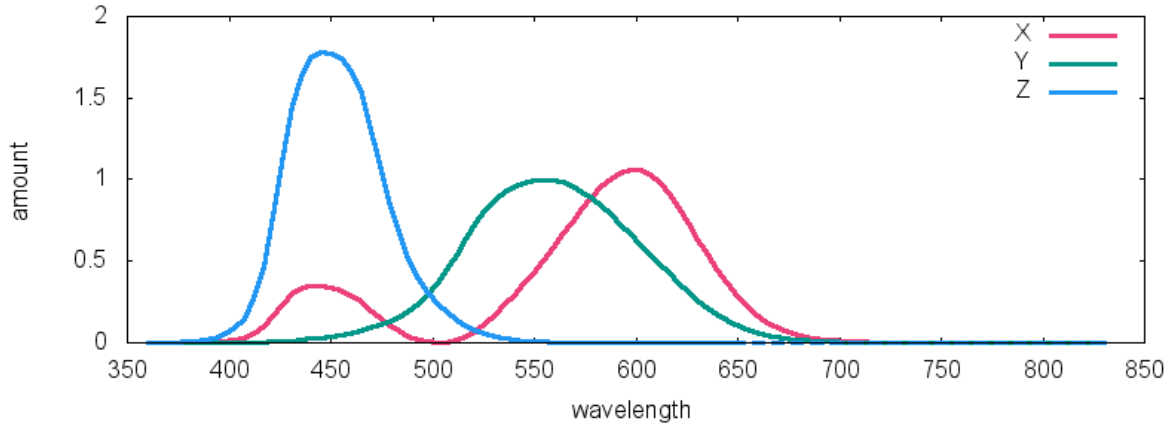


通过将这条曲线朝着接近或者远离原点的方向进行缩放，就可以表示不同亮度的单色光在这个空间下的表示。

取消负值

由于RGB color space 中有负值，于是有人便转换到了一个XYZ颜色空间，使用一个3x3的空间变换矩阵在两个颜色空间做转换。

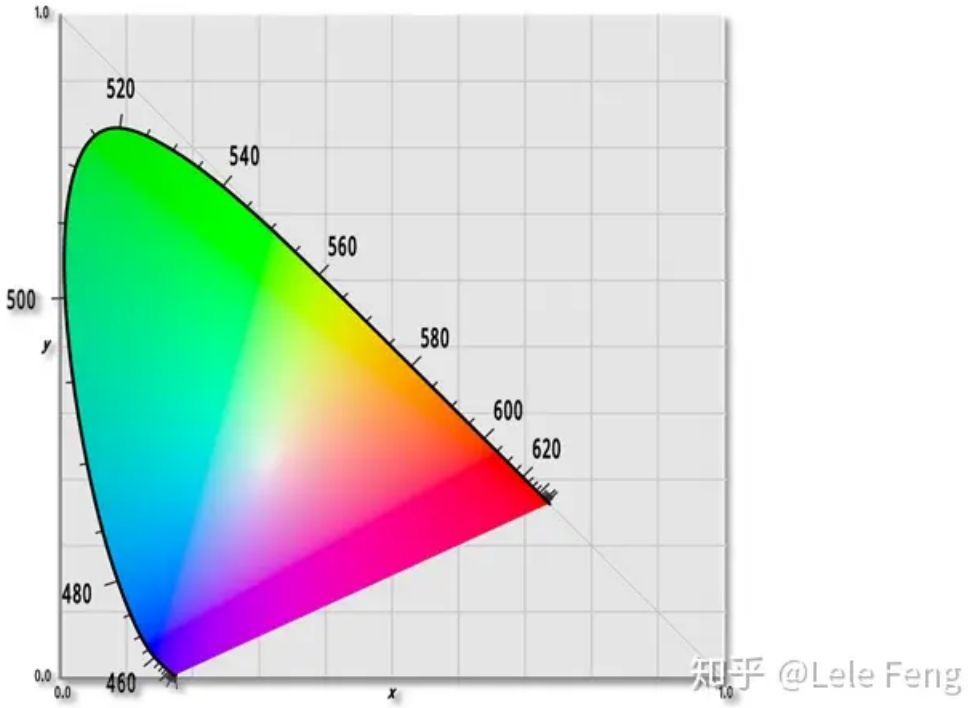
XYZ color space有很多好处，最明显的特点就是XYZ color matching functions全是正值，另一个特点是它的Y轴是刻意挑选出来的，它的走向和亮度函数是匹配的，也就是说Y值越大亮度也会越大。



再次降维

XYZ color space虽然很好，但要在这样一个三维空间里讨论一个颜色还是有点困难。很多时候我们其实只关心颜色的色度和饱和度，不关心它的亮度，这意味着我们可以把颜色表示从三维降维到二维表示。通常来说，我们可以把一个颜色从XYZ color space投影到X + Y + Z = 1的平面上。这个二维色度空间（chromaticity color space）就被称为CIE 1931 xyz color space。有了x + y + z = 1这个约束，实际上我们只需要两个值就可以指定一个色度（chromaticity）。通常，我们选择留下xy丢掉z，即把上述x + y + z = 1的平面再投影到XY平面上，这样我们就可以只使用xy值来指定一个色度了。

降维以后的xy色度图



> 色度图的边缘线代表各个波长的单色光对应的颜色，即光谱轨迹（spectral locus）。
> 光谱轨迹线所围成的区域代表了通过混合各种不同波长单色光我们能感知到的颜色。
> 颜色越纯，指的便是靠近色度图边缘。

为什么Y可以代表亮度?

因为人眼对绿光波长最敏感，RGB系统三原色的相对亮度（不是数量的多少）比为：
$L_r : L_g : L_b = 1.000 : 4.5907 : 0.0601$ （这意味着蓝色给人的感觉非常暗,绿色给人感觉最亮）。因此这里设置原色Y代表亮度和绿色的在颜色中的比例也比较也合理（颜色Y值越大，人眼感觉越亮，也代表颜色中绿色的比例越高）。

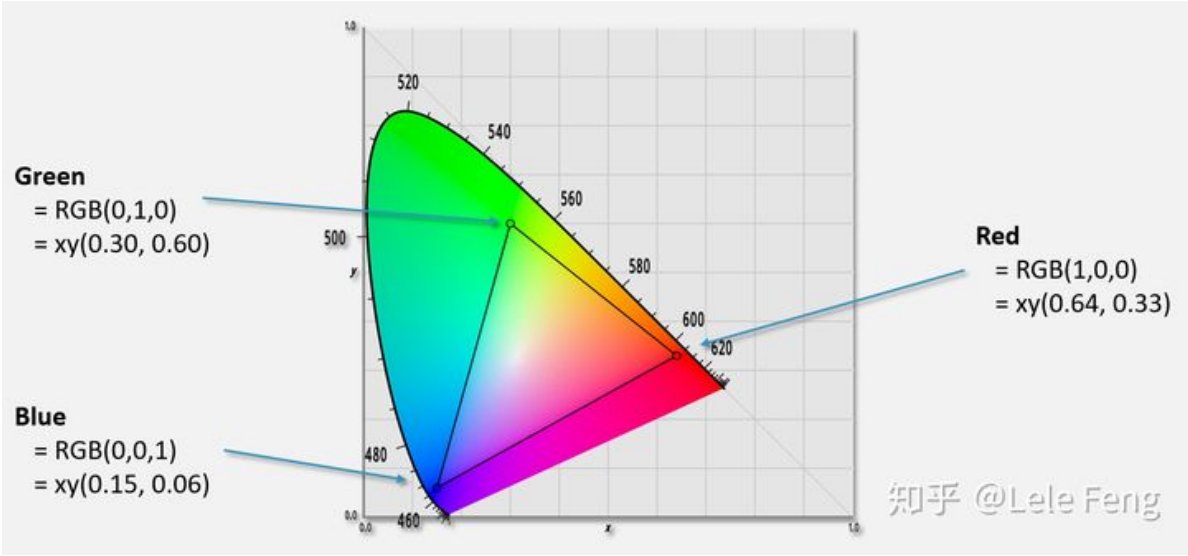
另外由于颜色匹配也包括其亮度的匹配，因此方程(1)可以将三原色相对亮度代入可得到亮度方程
$L(C) = r + 4.5907g + 0.0601b$，如果颜色在无亮度$L(C) = 0$线上，则
$r + 4.5907g + 0.0601b = 0$，代入$b = 1 - r - g$，整理后可得 XZ 线的方程。

颜色空间

上面说到，我们定义了一个**CIE 1931 xyz color space**,这个颜色空间基本包含了我们能感知到的最大色域，所以**CIE 1931 xyz color space**成为我们定义其他色域的标准空间。

通常，我们工作的色域范围并不是**CIE 1931 xyz color space**所涵盖的范围，而是在此基础上的一个RGB颜色空间。

上图右侧是一个三维的RGB颜色空间。为了方便可视化们通常会在CIE 1931 xy色度图里来表示它的色度范围（右下图），上面的黑点表示右上图的所有像素在这个RGB颜色空间的位置。



如何定义一个RGB颜色空间呢

- 三原色（Primaries）
- 白点（Whitepoint）
- 传递函数（Transfer Functions）

三原色（Primaries）

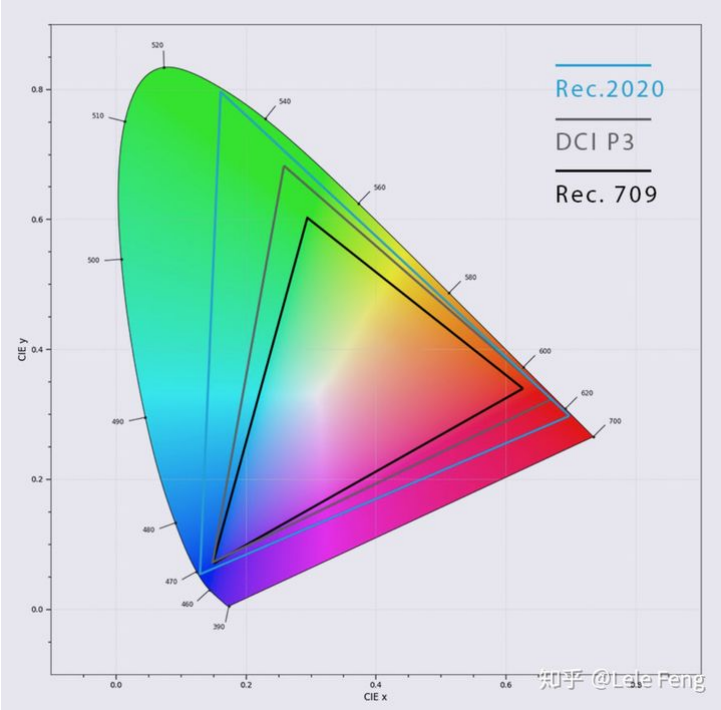
此处的三原色和文章最开头的三原色并不是一个定义，此处的三原色指的是构成我们工作的色域范围的三原色在 **CIE 1931 xyz color space**中的坐标。例如我们熟悉的sRGB颜色空间三原色的xy坐标如下：

可以看到，三原色RGB指的是红色R（1，0，0），绿色G（0，1，0），蓝色B（0，0，1）在CIE 1931 xy标准色度图中的xy坐标。三原色构成的三角形涵盖的色域图是我们工作范围的色域图，不同的RGB空间的标准不同，其三原色在CIE 1931 xy色度图中的xy坐标也不同。

以下给出几种常见的颜色空间以及它们的三原色和白点（后面会详细讲白点）的位置

Color Space	Gamut	White Point	Primaries
Rec. 709 / sRGB	CRT	D65	R (0.64, 0.33), G (0.30, 0.60), B (0.15, 0.06)
DCI-P3 (Display)	Wide	D65	R (0.680, 0.320), G (0.265, 0.690), B (0.150, 0.060)
Rec. 2020 / BT2020	Wide	D65	R (0.708, 0.292), G (0.170, 0.797), B (0.131, 0.046)
CIE 1931 RGB	Wide	E	R (0.7347, 0.2653), G (0.2738, 0.7174), B (0.1666, 0.0089)
CIE 1931 XYZ	Unlimited	E	R (1, 0), G (0, 1), B (0, 0)

色度图中的这些颜色空间三角形给了我们一个比较不同颜色空间范围的可视化方式，范围越大的颜色空间可以访问到更鲜亮的颜色，这在画面颜色体验上是非常不同的感受（符华的MMD可以下载HDR版本，在iPhone手机用nPlayer看HDR版本和在电脑上看普通版本是完全不一样的体验）。以下在色度图里比较各种颜色空间的色域范围：

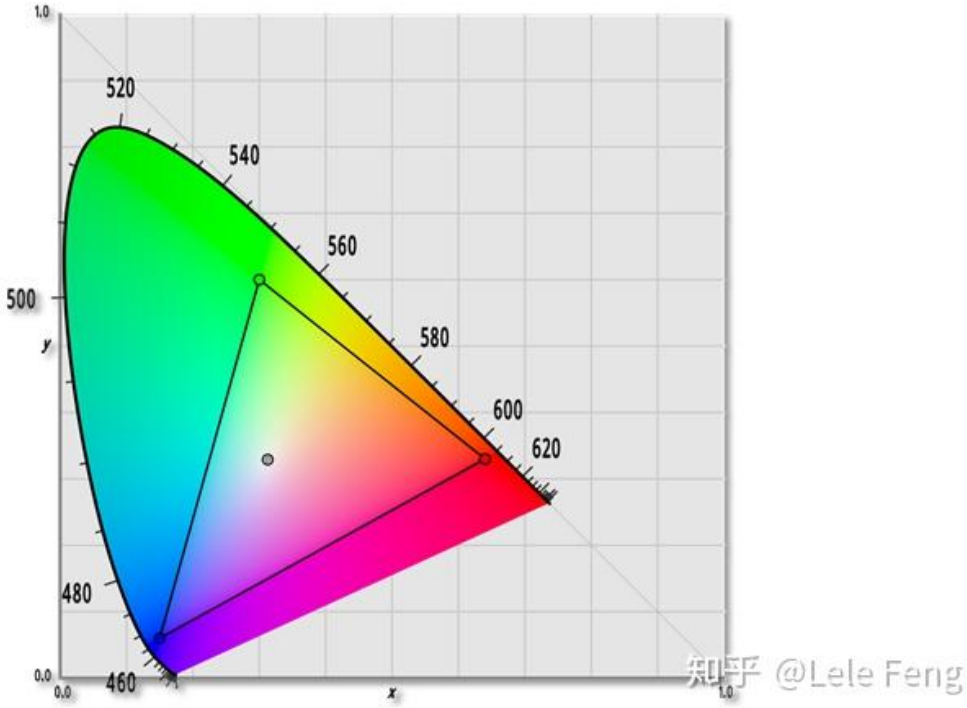


白点（Whitepoint）

我们之前说过，CIE 1931 xy色度图已经是经过降维后的二维空间了，这个空间只能表示色度信息而丢失了亮度信息，而我们实际需要使用的颜色空间仍然是三维空间。要想得到完整的颜色信息，我们需要定义一个白点（White Point）：

> The whitepoint defines the white color for a given RGB color space. Any set of colors lying on the neutral axis passing through the whitepoint, no matter their luminance, will be neutral to that RGB colorspace.
> —— From this article

例如，我们之前提到的sRGB颜色空间的白点D65的位置如下（xy坐标为(0.31271, 0.32902)）：



白点定义了RGB空间下，纯白色（1，1，1）在CIE 1931 xy标准色度图上的位置。通过白点我们可以得到颜色空间之间转换的矩阵

```
// 我们需要求解新的RGB颜色空间（sRGB）的三原色在XYZ颜色空间的索引值：
Rxyz = (Rx, Ry, Rz)
Gxyz = (Gx, Gy, Gz)
Bxyz = (Bx, By, Bz)

// 通过色度图中的三原色和白点坐标，我们可知：
Rxyz = (0.64, 0.33, 0.03) * r
Gxyz = (0.30, 0.60, 0.10) * g
Bxyz = (0.15, 0.06, 0.79) * b

// 白点在XYZ颜色空间的索引值为
Wxyz = (Wx, 1, Wz)
     = (0.31271, 0.32902, 0.35827) * w
     = |Rx Gx Bx| |1|
       |Ry Gy By| |1|
       |Rz Gz Bz| |1|

// 那么有：
w = 3.03933
Wxyz = (0.95043, 1, 1.08890)

// 结合两者有：
0.64r + 0.30g + 0.15b = 0.95043
0.33r + 0.60g + 0.06b = 1
0.03r + 0.10g + 0.79b = 1.08890

// 求解可得：
r = 0.644463125
g = 1.191920333
b = 1.202916667
```

```
// 那么sRGB颜色空间的三原色在XYZ颜色空间的索引值为：
Rxyz = (0.4124564, 0.2126729, 0.0193339)
Gxyz = (0.3575761, 0.7151522, 0.1191920)
Bxyz = (0.1804375, 0.0721750, 0.9503041)
```

由于XYZ空间的Y轴对应了亮度值，所以在新的颜色空间下计算亮度时，可以用矩阵的第二行作为系数。

色温

不同的色温便是在同样的三原色但是白点不同得到的颜色变换矩阵。

传递函数

定义 :由线性三色值到非线性视频信号的转换就是传递函数。使用传递函数有以下的优点

1. 优化带宽和存储空间
2. 适配人眼对暗部敏感的特性

有两种传递函数

- OETF 光转电传递函数（opto-electronic transfer function），负责把场景线性光（relative scene linear light）转换到非线性视频信号值（non-linear signal value）。例如，当我们用摄像机拍摄时，现实场景的光会经过一次OETF转换成摄像机的视频信号
- EOTF：电转光传递函数（electro-optical transfer function），负责把非线性视频信号值（non-linear signal value）转换成显示光亮度（display light）。例如，当我们把一个视频信号显示到屏幕上时，会经过一次EOTF

这些传递函数就是伽马矫正里使用的函数。伽马矫正（简称伽马）是指对线性三色值和非线性视频信号之间进行编码和解码的操作。在最简化版本的伽马矫正里，我们会使用一个γ < 1的值来作为编码伽马值（encoding gamma），使用这样一个伽马函数作为OETF（encoding function）来把线性颜色值转换成非线性信号值，这个过程被称为是伽马压缩（gamma compression）；相反的，我们会使用一个γ > 1的值作为解码伽马值（decoding gamma），使用这样一个伽马函数作为EOTF（decoding function）来把非线性信号值转换回线性颜色值，这个过程被称为是伽马展开（gamma expansion）。

我们经常听到人们说，sRGB的解码伽马值是2.2，但实际上sRGB的伽马展开转换函数使用的指数值是2.4。

现在的显示器和GPU里的一些做法

显示器

现代的显示器会应用个EOTF即gamama2.2，将存储在文件中的颜色值拉低。

主要原因是有两个

- 人眼看到的中灰在物理上是0.218左右而不是0.5。文件中的0.5是存储时的中灰，要通过gamama2.2将其变成显示的中灰
- 制定显示标准时，统计了当时市面上的大量显示设备得到了**gamama2.2**这个简洁的曲线。

我们在存储颜色值时，要更多的存储暗部细节，所以会应用一个**gamama1/2.2**，这样做的好处就是，暗部区域**0-0.2**能分配到128个色阶，不然只有50个色阶。

有关存储和显示对gamama的需求，我个人在思考的时候，分开看待更容易理解。

GPU

- SRGB缓冲

  以OpenGl为例子，我们可以使用**GL_FRAMEBUFFER_SRGB**命令让后续计算完的颜色值在存储到颜色缓冲前进行一个gamama矫正

  ```
  glEnable(GL_FRAMEBUFFER_SRGB)
  ```

  这会让存储在帧缓冲中的值都是被进行过gamama矫正的值，如果是8bit位数的缓冲，会丢失一部分亮度区域的信息。

- SRGB纹理

  如果不经过设置，那从PS中导出的图都是经过gamama矫正的，传给shader进行采样时，采出来的值自然也是非线性的。

  我们当然可以在采样结束后，对值进行一个重计算

  ```
  float gamma = 2.2;
  vec3 diffuseColor = pow(texture(diffuse, texCoords).rgb, vec3(gamma));
  ```

  但是每次采样都要这么做未免太麻烦，现代API都提供了一些设置来帮助我们在采样时直接进行一个gamamma映射

  申请一个纹理缓冲时，将其设置为SRGB或者SRGBA即可。

Unity

我们先从PS中导出一张没有经过gamama变换的线性图，视觉中灰，其物理值为0.2。

其显示为深灰色，由于是线性图，PS导出时，没有经过处理，显示的时候屏幕做了gamama变换，实际显示比真实要黑。

我们在Unity中设置颜色空间为**Liner**，将贴图设置成非SRGB贴图。采样后显示为

我们用拾色器进行拾色，发现值为0.5左右，说明在线性空间下采样一张线性图后会，写入颜色缓冲时会将值进行一下gamama矫正。

而当将图片设置为SRGB时，显示的值又为0.2。这是因为，SRGB的图在采样时会进行gamama变换，采样出来的值就是0.03左右，再写入颜色缓冲时，会再进行一次伽马矫正，变成0.2传输给屏幕。

而将Unity的颜色空间设置为Gamama，无论将贴图设置成SRGB还是非SRGB都不会在读取时进行伽马变换，写入颜色缓冲时也不会进行gamama矫正。

HDR和显示

HDR

PBR渲染是基于物理的渲染，意味着我们进行光照计算时，所用到的一些值都是真实的物理值。例如，阳光的强度在现实世界可以达到120K Lux，而计算出来的颜色的亮度值也会大于1。这意味着我们使用PBR模型进行光照计算时，会得到一个高范围的值，这个范围就称为HDR。

我们常用的显示设配是SDR显示器，SDR显示最常见的两个颜色空间标准是sRGB和Rec.709，这两个颜色空间并不能完全显示HDR的值

> 我的理解是在sRGB空间亮度值超过一，即超过了这个颜色空间所能显示的颜色范围，但是超过的值仍然可以在XYZ颜色空间找到。

SRgb和Rec.709

这两个颜色空间的三原色和白点的值是一样的，只不过传递函数不同。这么做是因为我们的显示环境 **Viewing Environment** 不同，Rec.709的标准的观察环境是用于家庭影院这种更为昏暗的照明环境的。因此，它们的传递函数被选择成能够更好地在这两种环境中表现图像细节的函数曲线。

由HDR到SDR

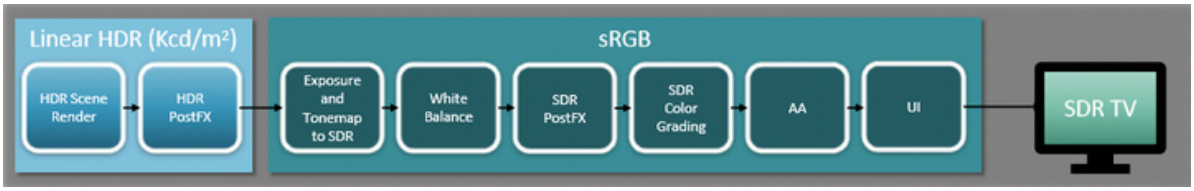
由于帧缓冲保存的值是0-1范围内，HDR计算出来的超过1的值，在进入帧缓冲后就会被丢弃，为了解决这个问题，我们需要把高动态的值重新映射到0-1的范围内。

这个技术也叫做**tonemapping**，参考tonemapping的进化史,不同的tonemapping曲线的对比效果参考

曝光

除了Tonemapping，和HDR渲染另一个紧密相关的计算是曝光（Exposure）。简单来说，曝光就是一个亮度缩放系数，例如对于使用天光照明的场景来说，我们可以参考摄影领域著名的阳光16法这部分内容可以参考李文磊写的真实物理属性的灯光环境在UE4中的应用探讨一文和聊聊自动曝光直播课程。

这里附上一张早期SDR Color Pipeline



图中各个步骤过程如下：

- 蓝色框内的步骤表示Scene Referred空间下的操作，该空间使用了以sRGB三原色表示的线性颜色值，渲染出来的场景亮度值可达到上千尼特。在这个HDR Scene Referred颜色空间（sRGB linear space）下，还会应用一些后处理计算
- 使用曝光和Tonemapping曲线把场景亮度重新映射到一个可供显示的区间范围
- 之后，使用sRGB OETF传递函数对数据进行编码，到这一步我们就转换到了Display Referred颜色空间即sRGB gamma space下了（图中绿色框）
- 继续在sRGB gamma空间里进行后续SDR后处理计算、Color Grading、AA和UI渲染等操作
- 最后将信号发送给SDR显示器，由显示器应用EOTF来编码后的电信号再次转换成真正的显示亮度

ACES一个统一的颜色管理标准

SDR显示在色彩管理和显示上有天然的缺陷，随着对色彩显示和存储的更高要求，我们迎来了HDR显示时代。

SDR的缺陷

- SDR颜色空间覆盖的色域范围太小，只有CIE 1931XYZ颜色空间的35.9%左右。无法表现一些饱和度高的颜色，影响显示的丰富度。
- 在低色域下计算光照和渲染会导致颜色值被CLIP掉，影响色彩的准确性。
- 从SDR到HDR显示器和从SDR到SDR显示器需要不同的标准，否则表现不一致，导致色彩管理的复杂度增加。