漫谈颜色和颜色空间

这是一个新增的章节，非计划内。

本身，我是没有打算写这个章节的。无他，对颜色和颜色空间的理解，除了需要技术层面，一般还需要美术层面。例如，你如果PS用得666，好歹对颜色的理解，会比一般人更容易深入一些，而我是个脸盲兼色盲，满肚子理论，奈何眼睛不给力。

让我坚定写这个章节的原因，是某天群里在讨论gamma矫正，sRGB等热门话题，我发现我虽然很菜，但是绝大多数人居然比我还要菜，我觉得我有必要科普一下。当然了，也可能大伙是对的，我是错的，讲得不好或者不对的，求喷。

**注意，前方高能！写文章之前，我有查过资料的，包括英文的，中文的，百度的，google的，知乎的。但是，我发现这一块资料逻辑混乱，前后矛盾，一知半解，无出其右。不管是百度的，知乎的，google的，中英文的，奇奇怪怪的资料一大堆，不乏各种逻辑混乱的，前后矛盾的。那么，如何能保证我的就是对的，别人就是错的呢？不能！但是，我觉得我的是对的，原因在于，我的资料，整个逻辑是前后通顺的，说得通的。此外，我所有引用的资料，都不是街边小资料，而一般都是MS这类官网上的资料。很多东西，我亲自写代码验证过的，并且不是简单的用个PS弄弄，用个unity写写代码，而是直接用的DX11，从底层，从根源上来理解。此外，觉得我写得不对的，求喷。请有理有据的喷，而不是为喷而喷。**

让我们从基础讲起，慢慢深入，然后浅出。

首先讲一些基本的概念。讲到颜色，绝对脱离不了RGB，这是基础的色光三原色。除了RGB，对颜色的描述，还有其他很多种类，例如典型的HSL/HSV，打印机的CMYK……这就是所谓的颜色空间，英文原文叫做color space。高中教科书应该讲过，色光三原色是RGB，各种书籍里面，都有大量关于RGB的描述，容易给人造成一个错觉：RGB是唯一的，RGB是权威的。然而，RGB确实是最常见的颜色描述，但是远远谈不上“唯一”。在很多领域，RGB更加连“最常用”都算不上。典型的是视频编码，YUV才是视频编码领域里面最常用的颜色空间（我应该有在其他章节详细讲过YUV、NV12之类的东东）。即使在RGB领域，还有其他很多的RGB，例如Adobe RGB，sRGB，Wide RGB，CIERGB……。打开PhotoShop，一眼看下去，几十种颜色空间，眼花缭乱。

如何理解颜色空间？没记错的话，我在其他章节讲过颜色的量化。理解了颜色的量化，其实很好理解颜色空间。颜色空间就是：制定一套颜色标准，用以量化并且描述颜色。例如，颜色千变万化，我们可以用RGB来混合出各种颜色，我们同样可以用HSL/HSV来混合出各种颜色。同样是RGB，人眼对明暗的感知不同，所以推出了sRGB（standard RGB）对标普通线性RGB。作为色彩软件届的大佬，Adobe公司怎甘人后，推出了自己的Adobe RGB……。

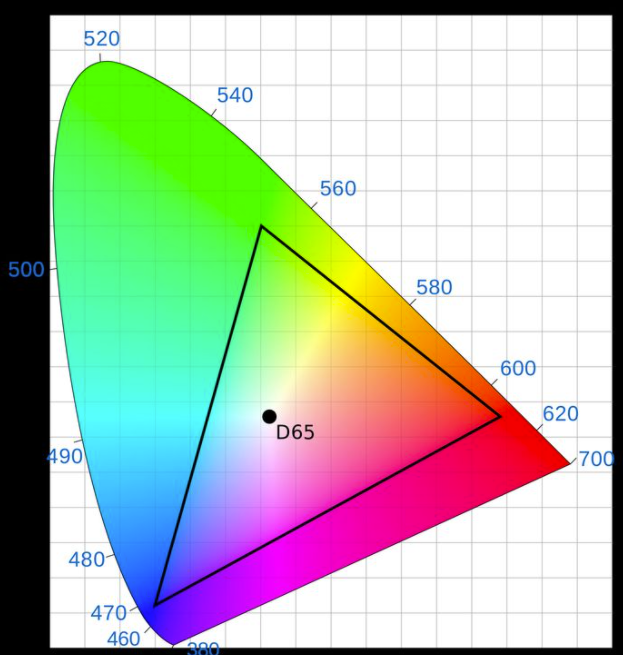
而为什么有不同的颜色空间？我们为什么需要那么多的颜色空间？

你可以这样理解，同样是旋转，你可以用四元数来描述，你可以用矩阵来描述，你可以用欧拉角来描述。再如坐标系，定义坐标系，你可以用平面直角坐标系，也可以用极坐标系。而颜色空间，不同的颜色空间有不同的用处或者需求，或者是历史原因。例如：RGB和CMYK，一个是色光三原色，一个是打印四原色，为什么需要两个体系？原因在于，色光三原色，是通过发光产生的颜色，而打印，是通过反射光的来表述出来颜色。有反射就有吸收，显示器要发出红光，直接一束红光，颜料要产生红光，需要吸收掉绿光和蓝光，这就是区别。如果同样用RGB作为原料，据说产生不了明亮的颜色。

人眼为什么能看见颜色？众所周知，光其实是电磁波，所以颜色本质上是不存在的，不同的颜色，是人眼的神经细胞对不同波长电磁波的反映。而不同的颜色，其实是波长不同的电磁波进入了人眼。我看过的资料，是人眼里面的神经细胞，对不同波长的电磁波产生不同的感应，所以产生了颜色。

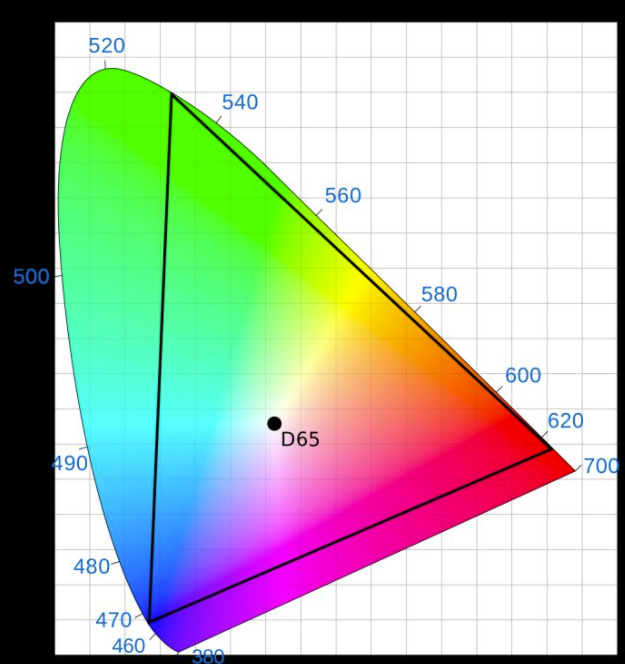
**那么有一个有点意思的问题：人眼到底能识别多少颜色？**

这个，有教科书答案，看图：



看图，马蹄铁形状的颜色，是人类眼睛能识别到的所有颜色集合，这个，叫做CIE1931。关于这个的资料，随便百度一下就有，我就不搬运了。大意是：有大佬在192x年代就专门研究过这个问题，并且建立了一个坐标系来描述这个问题。我仔细研究过这个坐标系的建立和旋转，有兴趣也也可以自己推算一下。

上图三角形内的颜色数，叫REC709，是1990年的颜色标准，也是目前主流显示器，电视机等等的颜色标准。就图来看，目前主流的显示器，距离人眼能识别到的颜色，还有较大差距。

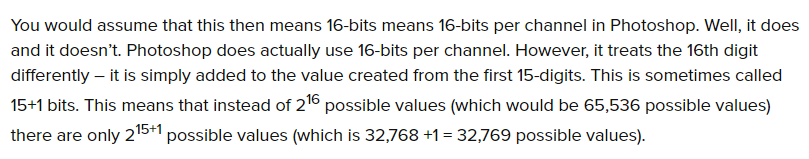


再看看这图。这图三角形部分描述的颜色，叫REC2020。这里，千万不要顾名思义，认为这是2020年指定的标准，其实不是。我没有仔细探究过这个命名的由来，但是我知道这个标准最早发布于2012年，也是未来显示器要追求的标准。

那么，人眼识别的颜色数实际是多少？这个问题，众说纷纭，比较主流的看法是：人类的眼睛，不借助其他工具，单通道能识别到16位已经是极限。目前的显示器，主流的显示器，单通道是8位，RGB是24位，这里，简称为RGB24。就我个人而言，我基本上很难分辨出RGB24比RGB16更好了。而目前一般的游戏贴图，都是RGB16，不要说手游，就算是端游，Nvidia一力主推的DXT格式，基本都是RGB16，这个我在其他章节应该讲过，有兴趣的可以自己去MS官网看，有详细的DXT压缩文档介绍。

目前，我所知道的，号称更新一代的显示器，貌似很贵的HDR显示器，号称是R10G10B10A2。这么高档的显示器，我本人没用过，只是听说。有用过的大佬可以说说感受。

实际计算中，普通的PC做颜色计算，绝大部分普通的windows程序，都是用的RGB24，每个通道8位，然后用uchar进行计算，也就是单通道都是0-255。这样的计算，一般是没问题的，但是做图像处理，显然是不行的。例如，你用颜色100，跟另外一个101混合，得到的颜色还是（100 + 101） / 2 = 100，那个1，就丢失掉了。在GPU里，都是用的0-1之间的浮点数，精度就高不少，效果要更好。而我网上看到的资料，说的是PhotoShop，这块计算用的是16位单通道，但也不是简单的16位，而是15位+1，也就是32769种颜色一个通道。是不是真的我不知道，但是可以肯定，专业的图像处理软件，用浮点数，或者更高精度的颜色进行计算，尽量精确，减少损失，是必要的。原文如下图：



以上这个答案足够权威吗？我没法回答。这是我觉得公开资料里面比较权威的解释。我们看PhotoShop，可以看到他们支持最高单通道16位色，可以看到他们支持单通道32位浮点数颜色，但是很难找到细节这个16位是不是标准的0-65535的16，还是32768 +1。谁有更多的资料，可以告诉我。

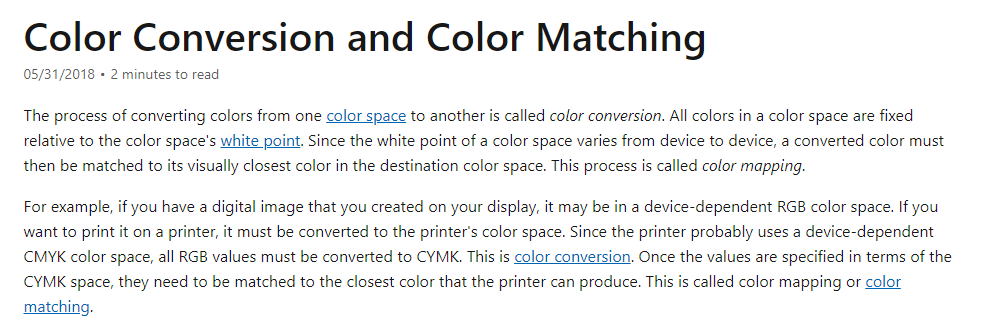
关于颜色，在图象基础里应该已经讲过很多，这里不打算重复了，下面着重讲讲颜色空间。

既然有众多的颜色空间，自然就涉及到转换。这就好比旋转矩阵转四元数，极坐标系转平面直角坐标系。本质上，你可以把RGB理解为一个RGB的坐标系，HSV理解为一个HSV坐标系。这里面有几点细节：

1. 转换过程都是可逆的。
2. 转换过程会有损耗，只能找最相近的颜色。
3. 转换公式有标准，但是不是绝对的。例如RGB转灰度图，我就见过不同的公式。标准之所以成为标准，首先是效果不能太差。效果差不多的时候怎么办？当然是谁的拳头硬谁说了算。

上面第三点大家估计有疑问：科学的东西，还能用拳头解决？当然能。历史上，产生过很多很多的图片格式，现在通用的，常用的，也就那么三两个，其他的，基本淘汰掉了。当年火遍全网的Real Player，现在已经见不到了。这些，不是不好，而是拳头不够硬。这样的例子，比比皆是。

颜色转换，英文叫Color Conversion，而颜色从一个颜色空间变换到另外一个颜色空间，这个英文叫做Color mapping或者Color matching。记一下这些关键字，有助于自己google查资料的时候用。这里，贴一段MS的英文原文解释，这个比较权威。



好了，开局讲完了，回到正题。写这篇文章的初衷，是群里对于Gama，sRGB的讨论，我们回到这个讨论。

**为什么Gamma？我的理解，有两个原因，第一个，是历史原因：**

我们知道，颜色的定义，我们一般是0-255之间，0是黑色，255是白色，这里，纯是一个线性关系。0、1、2、3……253、254、255.这个线性关系，好理解吧？但是，早期的CRT显示器，电压跟现实颜色的亮度，并非线性关系。理想的线性关系，是电压强度跟颜色一一对应，例如电压+1单位，颜色+1，电压+100，颜色+100。但是显示器不是理想状态，假设电压+100.，颜色+80。你不纠正，不就错了吗？所以需要校正，这个，就是历史原因造成的gamma校正。

典型的gamma值，有1.0，1.8，2.0，2.2等，标准的校正值，是2.2。这个的意思是，原本的RGB，做一个pow(RGB, 2.2)。这里为什么是2.2？因为颜色显示的时候，并非0-255，也是浮点显示，0-1之间。写过pixel shader的都知道，最终颜色值，就是0-1之间。

以上，是历史原因导致的gamma校正。

在液晶显示年代，这个校正，我所知道的，windows跟mac是不一样的。Windows定义的是2.2，而mac之前版本定义的是1.8，现在改成了2.2。我查了一下资料，很多显示器可以自己手动修改这个值，然后我尝试了一下我的DELL显示器，这个只有两个选项：PC跟MAC，不能自主修改。

这个，随便google一下就有答案，看图：



**第二个原因，是人眼对颜色的感受，本身不是线性的，而感光元件，对光的感受，是线性的。**

不好理解？我们来说说感光元件。例如最早期的相机，拍照的时候，就是感光元件，对光的感应，形成图象。看图：



这些，小时候冲洗照片的底片，印象深刻。

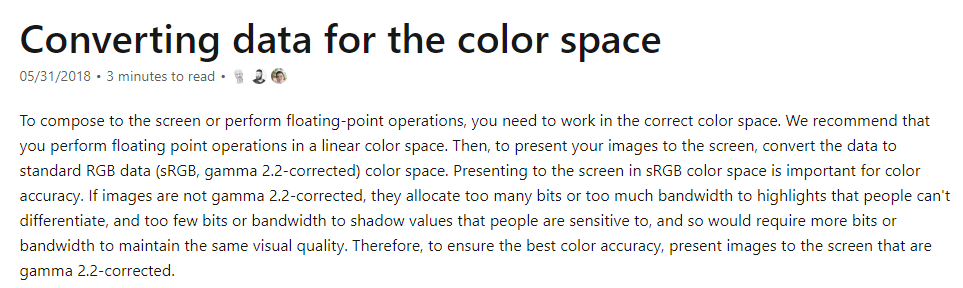
感光元件，对光的感应，是生硬的，冷冰冰的，你多亮，我就多亮。基本上是线性无疑。而人眼不一样。这么想一下就明白了，假设你在一个绝对的黑暗中，突然之间，进来了一丝光亮，其实这丝光亮非常小，亮度只有5，但是，你会觉得，整个世界明亮了。但是，如果你在一个非常明亮的房间，本身亮度是200，现在增加5，增加到205，估计你毫无感觉。

一句话，无论是感光元件，还是数学的逻辑光照计算，都是线性的，慎密的，而人眼，却不是线性的感光。所以，我们需要做gamma调整，让人眼感光效果更好。

这个，是现在图形学渲染后，图片做一个gamma校正，或者说，现在无处不在的sRGB的原因所在，这个，应该叫做standard RGB。

**这个校正，也是pow(RGB, 2.2)。这里，有一个奇妙的巧合，CRT显示器的校正，刚好跟人眼的校正，保持了基本一致！所以我认为，网上一切仅仅认为是历史原因，CRT显示器导致的Gamma校正，或者人眼对光的非线性感知导致的Gamma校正，都是片面的，不完整的，有问题的。这个Gamma值，其实只是大体相似，而不是绝对相似。据说CRT是2.2-2.5之间，而人眼是1.8-2.2之间。**

那么，问题来了，什么时候需要校正，什么时候不需要校正？怎么用才是合理的？这里，我引用MS大神的原话，按道理这个是比较权威的说法，看图：



To compose to the screen or perform floating-point operations, you need to work in the correct color space. We recommend that you perform floating point operations in a linear color space. Then, to present your images to the screen, convert the data to standard RGB data (sRGB, gamma 2.2-corrected) color space. Presenting to the screen in sRGB color space is important for color accuracy. If images are not gamma 2.2-corrected, they allocate too many bits or too much bandwidth to highlights that people can't differentiate, and too few bits or bandwidth to shadow values that people are sensitive to, and so would require more bits or bandwidth to maintain the same visual quality. Therefore, to ensure the best color accuracy, present images to the screen that are gamma 2.2-corrected.

这个翻译的意思就是：计算的时候，用线性，最终显示的时候，做gamma。为什么？后面说得很清楚了，如果不这么做，图片花了大量的bit或者bandwidth到人眼感光不明显的高亮部分，而人眼感官敏感的阴暗部分，却只有少量的bit或者bandwidth。

上面的图片链接：<https://docs.microsoft.com/en-us/windows/win32/direct3ddxgi/converting-data-color-space>

我查了一下网上的资料，中文资料对于gamma的解释，惨不忍睹（当然了，如果不是因为惨不忍睹，我也就不用写这个文章了），各种乱，各种互相矛盾。我觉得，要理解一样东西，首先要有一个确定的前提，这也就是我们科学上的“公理”。我先把MS这个解释，简称为“公理MS”，一切都以这个为准，不以这个为准的，矛盾的，都是错的，后续再展开推论。为什么以MS的解释作为公理？因为这个sRGB就是HP跟MS搞的啊，他们搞的东西，他们的解释，不应该是最权威的吗？当然，各位也可以找出来更加权威专业的解释或者定义，然后我后续跟这个相关的推论估计都被打脸了。有的话，来喷我。

上面一段话的后半句，我本人一直觉得是非常好理解的，但是我发现，我跟别人讨论的时候，别人却理解不到位。这个，其实跟那个深度值的计算是一样的。线性计算，跟log计算，跟pow计算，侧重点以及所占用的“bit”是完全不同的，我用实际的数字，来尝试解释一下。我现在以0-255为例。

0-255，是8位，一起是256个颜色数据，这点首先要能理解。理解不了的，没必要往下看了，你学不会。

那么，这里一共是256个颜色，一个颜色一个坑。我假设200以上的，属于高亮的颜色，那么有56个。但是，如果人眼对高亮颜色不敏感，仅仅只能识别出30个呢？那不就是有26个坑是没用的，给浪费掉了？反之，假设100以下属于偏暗部分，人眼能识别得更多，例如能识别到120个，这里，不就差了20个吗？如果我们设计的时候，就考虑到这个原则，高亮部分位数减少，偏暗部分位数增加，不就很好的解决了这个问题？这就是典型的线性的数据储存和非线性的人眼识别之间的矛盾。

好了，解释完了，能明白就明白，不能明白的也没办法了。回到上面的sRGB的解释，这个其实很好理解，因为本身图片无论是获取（拍照，录视频），制作（photoshop等各种软件），光照算法……诸如此类，涉及到计算的，宇宙的规则，这就是线性的，用线性来计算，没问题。而最终显示，为了让人眼得到更好的效果，做一个gamma就好了。

以上解释，只是大方向。但是，估计不大好理解。这里，我自己写程序测试一遍，然后把代码，结果都贴出来，便于理解。第一个，假设我代码写入一张图片，亮度从0-255，线性。假设LCD显示器没有做gamma校正，会看到什么？看图：

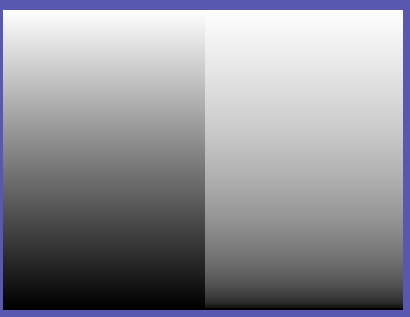


图1

这图是我用DX写的，图片代码生成，左边是线性图片，右边是做了个pow(rgb, 0.45)的。我很少用unity做这类测试，因为我并没有unity源码，不知道底层做了什么东东。图片是线性的，显示器上看到也是线性的。

我的理解是：如果显示器并没有做pow(rgb, 2.2)的校正，那么，你本身应该看起来是左边的线性的效果，人眼会看成了右边的！

不信的话，自己看看自己的显示器，可不可以调整gamma。如果能调整，调整成1试试看。

所以我认为，自然界的光，本身是线性的。但是人眼是非线性的，这导致了看显示器的时候，造成了颜色偏差。Gamma校正，就是为了调整这个偏差的。那么，更早期的CRT显示器，为什么没有这个调整？只有把gamma调整成简单的线性？我认为，那是显示器发展的早期阶段。要知道，sRGB标准，是1996年才提出的，等到各大厂商都支持了，至少2000年以后的事情了。而早期的CRT显示器，不同厂商的显示器，颜色偏差估计都比这个gamma偏差还要大。我记得我第一次见电脑，那是1996年。学校机房的各种286.386，486，很多还是纯黑白的，颜色偏差压根没人注意。等到2000年左右，去网吧，小县城的网吧，电脑显示器五花八门，颜色偏色比比皆是。只能说，现在要求高了，有标准了，才有现在严格的Gamma2.2罢了。

那么，假设我这个线性的数据，写入SRGB的图片，会出来什么样的效果？看图：

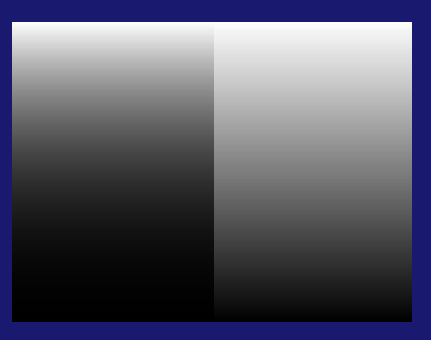
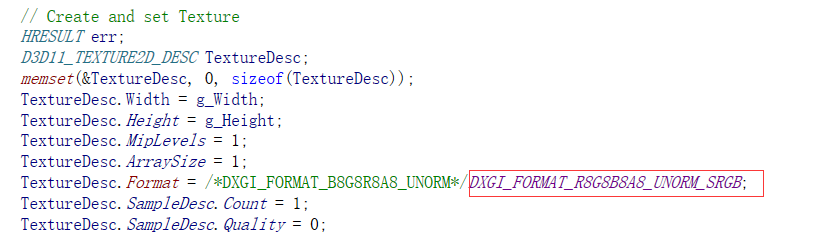
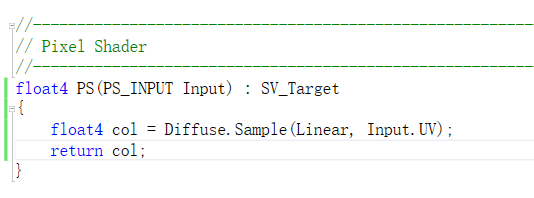


图2



为什么变暗了？很简单啊，因为线性RGB数据中的0.5跟sRGB中的0.5，不是一个数据啊。因为你创建的是sRGB的图，传进来个0.5，电脑就会认为你传进来的是sRGB中的0.5。这就好比你建立了一个坐标系，传入了数据，别人当然认为你传进来的是当前坐标系下的数据，真实使用的时候，会变换成线性坐标系。

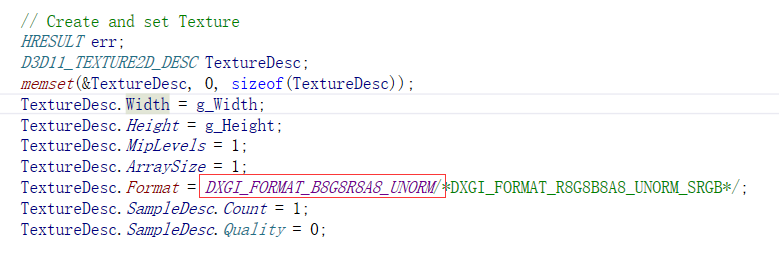
看看这里：

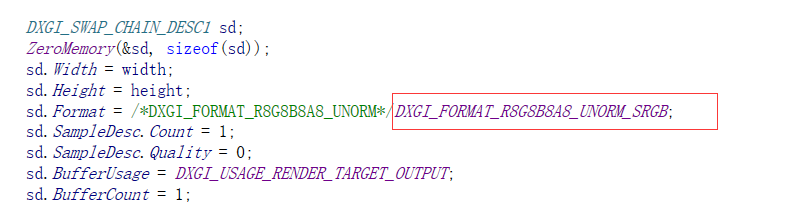


这个是一个极其简单的pixel shader。问题来来了，这个代码你是已知的，而且图片里的像素值你也是已知的，那么，为什么sRGB变暗了？我们可以轻易推出，这个sample，也就是说这个纹理采样，内部是做了pow(rgb,2.2)的。意思是：假设你是创建一张linear的图片，那么采样的时候，你本身是0.5就是0.5。但是假设你这张图是sRGB的，那么你sample的时候，GPU内部会给你做一个pow(rgb,2.2)。这个，圆满的呼应了之前“MS公理”：显示的时候，做gamma校正，计算的时候，换算成linear。显卡公司，无论是NV还是ATI，跟MS一条裤子的，估计都是互相制定了这些标准。

当然了，以上所有的一切，我没有找到任何资料，只是根据计算过程的推测。如果有知道资料的，或者是NV内部的，MS内部的，可以出来现身说法，解释一下。

下面，我们再来做一个测试，我把创建图片改成传统的RGB，而swapchain改成sRGB，看图：





那么，结果是怎么样的呢？看图：

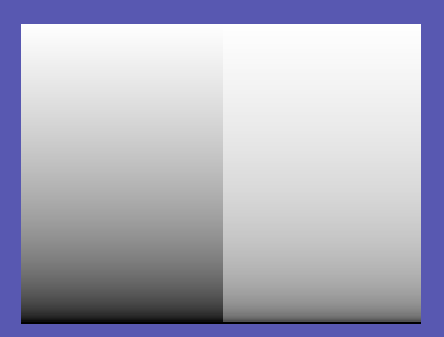


图3

根据上图，我做了以下总结：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Texture格式： |  | SwapChain格式： |  | 结果： |
| RGB |  | RGB |  | 渐变正常（图1） |
| sRGB |  | sRGB |  | 渐变正常（图1） |
| RGB |  | sRGB |  | 变亮（图3） |
| sRGB |  | RGB |  | 变暗（图2） |

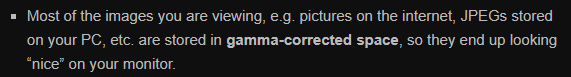
从上述表格，以及我们所知道的中间过程，基本可以推出：

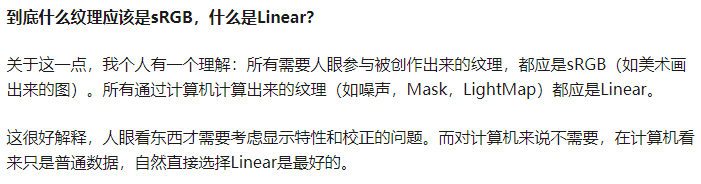
1. **使用同样的颜色空间的时候，颜色正常。**
2. **GPU里，当RGB颜色转换成sRGB颜色的时候，会有一个pow(x, 1/2.2)的转换。**
3. **GPU里，当sRGB颜色转换成RGB颜色的时候，会有一个pow(x, 2.2)的转换。**

以上三点，基本推测出了GPU对这个sRGB转换的一些基本准则，使用的时候需要注意。

一个非常有意思的，网上各种前后矛盾的问题：我们平常保存的图片，例如BMP、PNG、JPG，到底是什么颜色空间的？默认是不是已经做了Gamma校正的，sRGB颜色空间的图片？

我网上看了，英文的，中文的，很多都说，图片已经做了gamma校正的，因为做了的话，颜色更好。我随便截一下图：

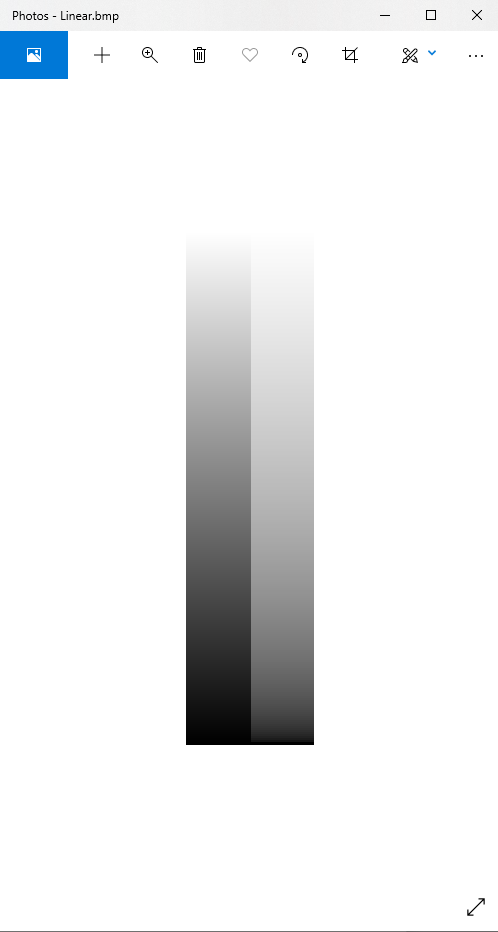




上图来自知乎文章。

中英文我分别来一篇。然而，我的意见却是不同的。我不同意见的理由在于：

1. 图片格式只是一个二进制文件，例如BMP文件，文件非常简单，压根就没有“颜色空间”选项，所以，通过文件格式识别这是不是sRGB颜色空间，完全不可能！不信的大佬，可以去搜一下BMP文件格式，PNG文件格式，JPG文件格式，DXT文件格式，哪个格式文件头有一个标记，表示这个是什么颜色空间了吗？
2. sRGB仅仅是一个颜色空间，是几十个颜色空间里面的一个，我们知道的颜色空间，光是RGB颜色空间都有一大堆，sRGB之所以被广泛认知，一个原因是MS强势，另外一个原因是比较符合人眼对光的感知。但是，这个宇宙的光的规则，本身就是线性的，因此，线性的颜色空间，才是最合理的。
3. 不存在说保存成sRGB颜色空间，图片看起来效果更好，更符合人眼。我专门请教过公司的美术，不止一个，包括专门搞摄影的，搞修图的。他们的说法我总结如下：平时用PS，压根不关注用什么颜色空间，都是默认的。画图，改图，保存，都是直接来的。好几个居然都不知道sRGB是个什么东东，根本谈不上改图、修图的时候专门选择sRGB了。搞摄影的，拍摄的，倒是对这个有比较深入的了解。摄影器材，很多是可以选择保存的颜色空间的，例如你确实可以保存成sRGB的，也可以保存成其他的。不同的摄影器材，能保存的颜色空间非常多。平常来看，不好看出来有什么太大的区别。正常的处理，如果你摄影的时候选择了sRGB，那么你PS操作的时候，同样选择sRGB即可。我没玩过摄影器材，有玩过的其实也可以出来现身说法。

所以我的看法，除非特别的器材获取的图片，视频，规定了颜色空间，否则，一般的图片处理，都是线性颜色空间。我曾经程序直接写入一张BMP图片，线性的，然后显示出来的效果同样是线性的。看图：

如果说普通的图片保存成的不是线性空间，而是sRGB空间的话，那么，显示器显示的时候，如何知道这是线性还是sRGB？这不乱套了吗？因此，是什么颜色空间，纯靠人指定，所以才会导致Unity导入的时候有一个是否sRGB的选项。而显示器显示的时候，就没有这个选项了。

当然了，以上全是我个人理解，不能保证绝对正确，不对的大家可以喷我。

好了，关于颜色和颜色空间，暂时讲到这里。