# Map D3D9 to D3D10

当前可以将D3D9支持格式映射到D3D10格式，不过会存在一些名称的变化和格式的变化，主要从D3D9映射过来的一些格式可以将

# Source Types

# Data Conversion Rules

## Data Type Terminology

SNORM：有符号归一化整形

UNORM：无符号归一化整形，对n位数字，0表示0.0f，1表示1.0f，表示从0.0-1.0f的均匀分布的浮点序列

SINT：有符号整型

UINT：无符号整型

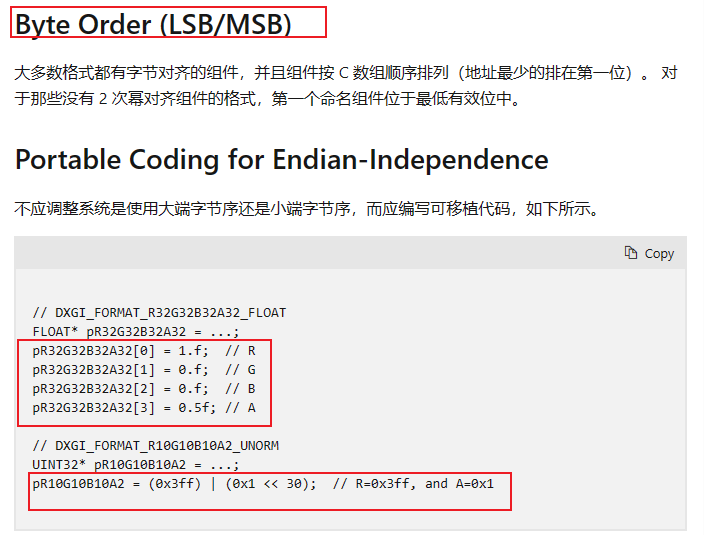
FLOAT：浮点

SRGB：和UNORM类似，不同的是编码序列解释为非线性级数并显示为一些列RGB 颜色

FOURCC 代码是通过串联四个 ASCII 字符创建的 32 位无符号整数。 例如，YUY2 视频的 FOURCC 代码为“YUY2”。 对于压缩的视频格式和非 RGB 视频格式 (（如 YUV) ），应将 [**BITMAPINFOHEADER**](https://learn.microsoft.com/zh-cn/windows/win32/api/wingdi/ns-wingdi-bitmapinfoheader) 结构的 **biCompression** 成员设置为 FOURCC 代码。

## FLOAT Point Conversion

从较高范围向较低范围表示形式转换



DXGI\_FORMAT\_NV12  
值：103  
最常见的 YUV 4：2：0 视频资源格式。**此视频资源格式的有效亮度数据视图格式DXGI\_FORMAT\_R8\_UNORM和DXGI\_FORMAT\_R8\_UINT。** (宽度和高度的有效色度数据视图格式是此视频资源格式的 1/2 亮度视图) DXGI\_FORMAT\_R8G8\_UNORM和DXGI\_FORMAT\_R8G8\_UINT。支持的视图类型包括 SRV、RTV 和 UAV。对于亮度数据视图，到视图通道的映射为 Y->R8。对于色度数据视图，到视图通道的映射为 U->R8 和  
V->G8。  
  
有关用于视频呈现的 YUV 格式的详细信息，请参阅[建议用于视频呈现的 8 位 YUV 格式](https://learn.microsoft.com/en-us/windows/desktop/medfound/recommended-8-bit-yuv-formats-for-video-rendering)。  
  
宽度和高度必须均匀。此格式的 Direct3D 11 暂存资源和 initData 参数使用 （rowPitch \* （height + （height / 2））） 字节。第一个（SysMemPitch \* height）字节是Y平面，其余（SysMemPitch \*（高度/ 2））字节是UV平面。

使用 YUY 4：2：0 格式的应用必须将亮度 （Y） 平面与色度 （UV） 平面分开映射。开发人员通过为相同的纹理调用 [ID3D12Device：：CreateShaderResourceView](https://learn.microsoft.com/en-us/windows/desktop/api/d3d12/nf-d3d12-id3d12device-createshaderresourceview) 两次并以 1 通道和 2 通道格式传递来做到这一点。以与 Y 平面兼容的 1 通道格式传递仅映射 Y 平面。以与 UV 平面兼容的 2 通道格式（一起）传递，仅将 U 平面和 V 平面映射为单个资源视图。

# YUV格式后的数字的意义

[YUV](https://link.zhihu.com/?target=https%3A//en.wikipedia.org/wiki/YUV) 是一种彩色编码系统，主要用在视频、图形处理流水线中(pipeline)。相对于 RGB 颜色空间，设计 YUV 的目的就是为了编码、传输的方便，减少带宽占用和信息出错。

人眼的视觉特点是对亮度更铭感，对位置、色彩相对来说不铭感。在视频编码系统中为了降低带宽，可以保存更多的亮度信息(luma)，保存较少的色差信息(chroma)。

Y’UV、YUV、YCbCr、YPbPr 几个概念其实是一回事儿。由于历史关系，Y’UV、YUV 主要是用在彩色电视中，用于模拟信号表示。YCbCr 是用在数字视频、图像的压缩和传输，如 MPEG、JPEG。今天大家所讲的 YUV 其实就是指 YCbCr。Y 表示亮度（luma），CbCr 表示色度（chroma）。

luminance 亮度，luma 是在视频编码系统中指**亮度值**；chrominance 色度，chroma 是在视频编码系统中指**色度值**。

Y’UV 设计的初衷是为了使彩色电视能够兼容黑白电视。对于黑白电视信号，没有色度信息也就是(UV)，那么在彩色电视显示的时候指显示亮度信息。

Y’UV 不是 Absolute Color Space，只是一种 RGB 的信息编码，实际的显示还是通过 RGB 来显示。Y’，U，V 叫做不同的 component 。

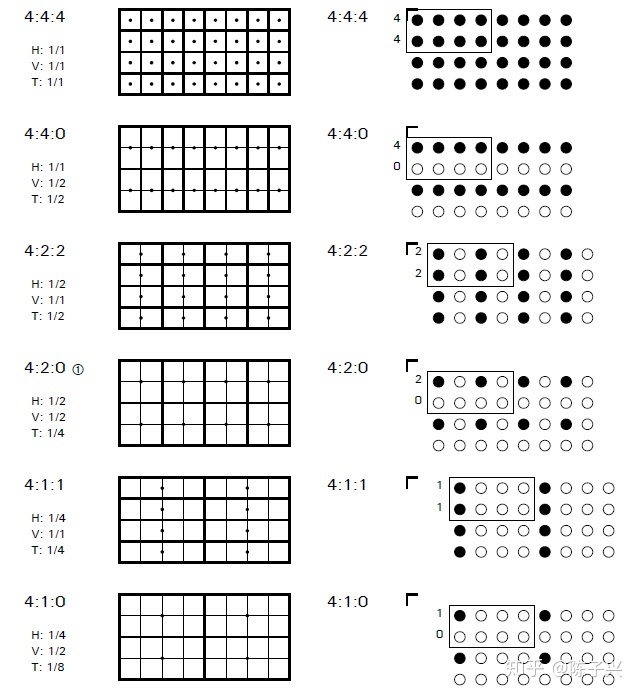
如图中所示，左侧一列，每**一个小矩形是图形像素**表示，**黑框矩形是色度像素**表示，**小黑点是表示色度像素值**(Cb+Cr)，**表示图形像素和色度像素在水平和垂直方向的比例**关系。比如，

右侧一列是**二次采样模式记号**表示, 是 **J：a：b （YUV后的数字）**模式，实心黑色圆圈表示包含色度像素(Cb+Cr），空心圆圈表示不包含色度像素。对于 J:a:b 模式，主要是围绕**参考块**的概念定义的，这个**参考块**是一个 J x 2 的矩形，J 通常是 4。这样，此**参考块**就是宽度有 4 个像素、高度有 2 个像素的矩形。a 表示**参考块**的第一行包含的色度像素样本数，b 表示在**参考块**的第二行包含的色度像素样本数

4：4：0 参考块第一行包含四个色度样本，第二行没有包含色度样本。

4：2：2 参考块第一行包含两个色度样本，第二行也包含两个色度样本，他们是交替出现。

4：2：0 参考块第一行包含两个色度样本，第二行没有包含色度样本。



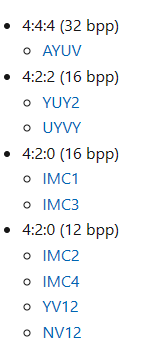
我们可以说YUV444代表4个像素块中每一个都有亮度亮度值，并且在这四个像素中CB和CR都站四个像素，但是当0的时候呢

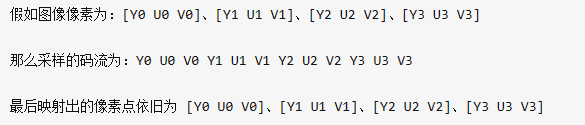
现在知道yuv444，yuv422，yuv420 yuv 等像素格式的本质是：每个图形像素都会包含亮度值，但是某几个图形像素会共用一个色度值，这个比例关系就是通过 4 x 2 的矩形**参考块**来定的。这样很容易理解类似 yuv440，yuv420 这样的格式了

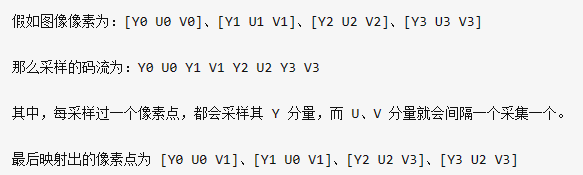
色度通道的采样率可能低于亮度通道，而不会显著损失感知质量。称为“A：B：C”表示法的表示法用于描述相对于 Y 对 U 和 V 进行采样的频率：

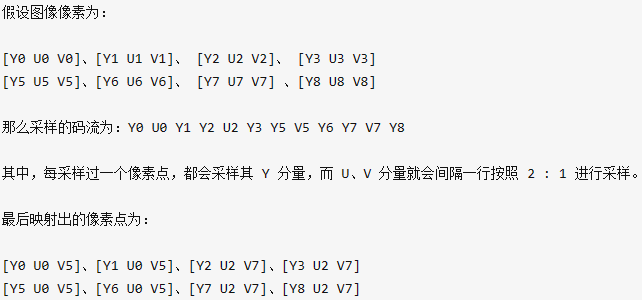
* 4：4：4 表示色度通道没有缩减采样。
* 4：2：2 表示 2：1 水平下采样，没有垂直下采样。每两条 U 或 V 样本每条扫描线包含四个 Y 样本。
* 4：2：0 表示 2：1 水平下采样，2：1 垂直下采样。
* 4：1：1 表示 4：1 水平下采样，没有垂直下采样。每个扫描线包含每个 U 和 V 样本的四个 Y 样本。

每个YUV格式都有FOURCC代码，







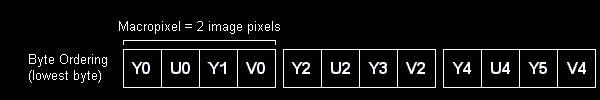


怎么理解采样的码流

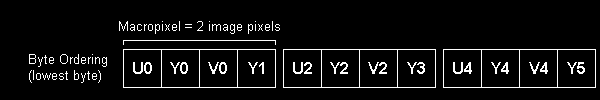
### YUYV和UYUV格式 基于 YUV 4:2:2 采样的格式

YUV 4:2:2 采样规定了 Y 和 UV 分量按照 2: 1 的比例采样，两个 Y 分量公用一组 UV 分量。

YUYV 格式是采用打包格式进行存储的，指每个像素点都采用 Y 分量，但是每隔一个像素采样它的 UV 分量，排列顺序如下：



UYVY 格式也是采用打包格式进行存储，它的顺序和 YUYV 相反，先采用 U 分量再采样 Y 分量，排列顺序如下：

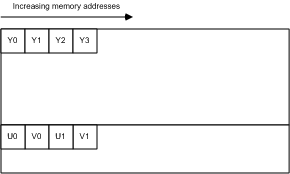


### NV12 和 NV21 格式 基于YUV420的采样格式

NV12 和 NV21 格式都属于 YUV420SP 类型。它也是先存储了 Y 分量，但接下来并不是再存储所有的 U 或者 V 分量，而是把 UV 分量交替连续存储。

NV12 是 IOS 中有的模式，它的存储顺序是先存 Y 分量，再 UV 进行交替存储

所有 Y 示例首先在内存中显示为具有偶数行的无符号**字符**值数组。Y 平面后面紧跟一个无符号**字符**值数组，其中包含打包的 U （Cb） 和 V （Cr） 样本。当组合的 U-V 数组被寻址为小端 **WORD** 值数组时，LSB 包含 U 值，MSB 包含 V 值。NV12 是 DirectX VA 的首选 4：2：0 像素格式。预计这将是支持 4：2：0 视频的 DirectX VA 加速器的中期要求。下图显示了 Y 平面和包含打包的 U 和 V 样本的数组。



YUV 420P 和 YUV 420SP 都是基于 Planar 平面模式 进行存储的，先存储所有的 Y 分量后， YUV420P 类型就会先存储所有的 U 分量或者 V 分量，而 YUV420SP 则是按照 UV 或者 VU 的交替顺序进行存储了，具体查看看下图：

YUV420SP 的格式：



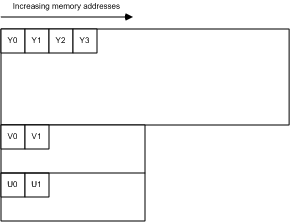
YUV420P 的格式：



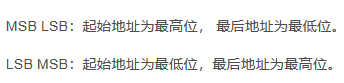
### YU12 和 YV12 格式

YU12 和 YV12 格式都属于 YUV 420P 类型，即先存储 Y 分量，再存储 U、V 分量，区别在于：YU12 是先 Y 再 U 后 V，而 YV12 是先 Y 再 V 后 U 。

YV 12 的存储格式如下图所示：



排布顺序与LSB MSB有关

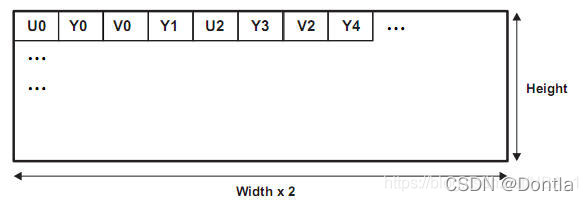
****

打包(packed)格式（紧缩格式）和平面(planar)格式。前者将 YUV 分量存放在同一个数组中,通常是几个相邻的像素组成一个宏像素(macro-pixel);而后者使用三个数组分开存放 YUV 三个分量,就像是一个三维平面一样。

# 压缩格式Packed Formats

压缩格式是指用一个数组表示 YCbCr，每一个 component 是交替出现的。

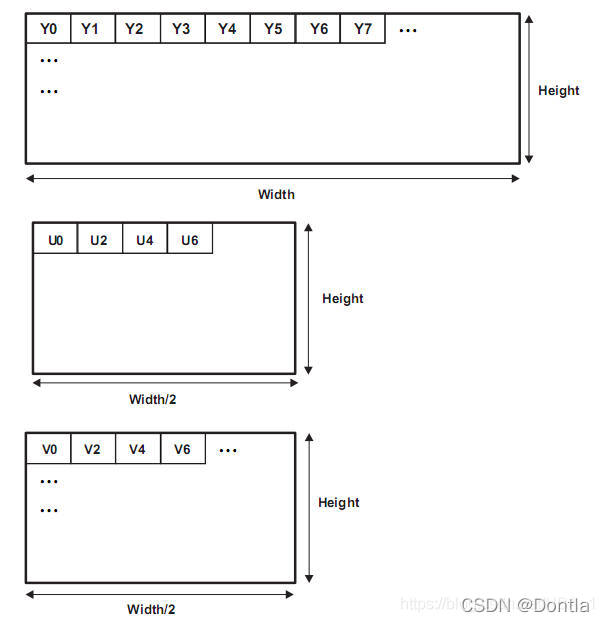
指先连续存储所有像素点的 Y 分量，然后存储 U 分量，最后是 V 分量。



# 平面格式Planar Format

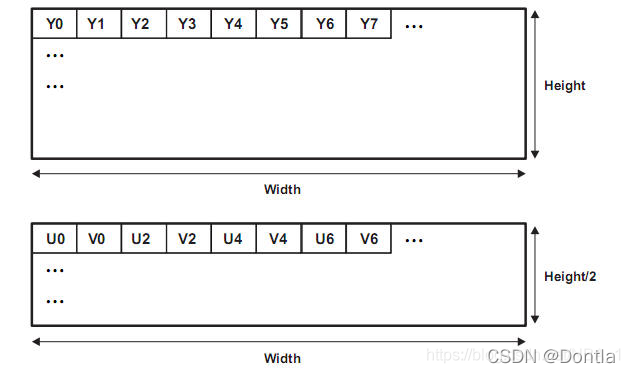
平面格式是用三个不同的数组来表示YUV（YCbCr）的三个Component，每一个 Component 都是通过不同的平面表示。为此，每一个 Component 会对应一个 plane。

指每个像素点的 Y、U、V 分量是连续交替存储的



### YUV420sp（Semi-Planar）（这个是我们hikflow\_demo里用到的格式，NV21）★

[参考文章：关于yuv 的pack(紧缩格式)和planner(平面格式)格式](https://blog.csdn.net/NBA_1/article/details/102569907)

这个格式的数据量跟YUV420 Planar的一样，但是U、V是交叉存放的，如下图。  


[D3D11 texture 创建 NV12 YV12 YUV420 cpu 访问 内存分布 拷贝 map updateresource copyresource\_d3d11 copyresource\_丘上人的博客-CSDN博客](https://blog.csdn.net/qiushangren/article/details/89244745)

RGB YUV之间是可以相互转化的，YUV自己也可以转化