**LUT**

**什么是LUT？**

执行每个像素的所有颜色分级步骤是一个很大的工作量，我们虽然可以添加一些只执行某些步骤的变体，但也需要声明大量的关键字。相反我们可以将颜色分级烘焙到查找表中（简称LUT，Look-Up-Table），通过对其采样来转换颜色。LUT是一种3D纹理，通常为32\*32\*32，用烘焙填充该纹理并稍后进行采样比直接对整个图像执行颜色分级要少很多工作量，URP和HDRP也使用这个方法。

**渲染到2D LUT纹理**

1. LUT是3D纹理，但常规着色器无法渲染到3D纹理。因此我们将2D切片连续放置一排，组合成一个宽的2D纹理来模拟3D纹理。这样LUT纹理的高度等于配置的分辨率，其宽度等于分辨率的平方

2. 从现在开始，我们要将颜色分级和色调映射都渲染到LUT纹理中，重命名现有的色调映射通道，因此ToneMappingNone将变为ColorGradingNone等等，然后将源纹理渲染到LUT纹理中而不是相机目标中，最后再将源纹理数据拷贝到相机目标，以获取未经调整的图像作为最终结果，并释放LUT。现在我们绕过了颜色分级和色调映射，但帧调试器显示我们在最终拷贝之前绘制了扁平化的图像。

**LUT颜色矩阵**

1. 要创建合适的LUT纹理，我们需要用颜色转换矩阵填充它。可以通过调整颜色分级的Pass方法来使用从UV坐标提取的颜色，而不是对源纹理进行采样。定义GetColorGradedLUT方法得到颜色值然后立即进行颜色分级，颜色分级的Pass各个片元函数只需要调用色调映射。

2. 我们可以通过调用GetLutStripValue方法获取LUT的输入颜色，它需要UV坐标和从CPU发送来的颜色分级LUT参数向量。

3. 回到PostFXStack脚本的DoColorGradingAndToneMapping方法中，将LUT相关参数发送到GPU。四个参数分别是LUT高度、0.5除以LUT宽度、0.5除以LUT高度和高度除以高度减一。

Log C LUT

1. 我们获得的LUT矩阵位于线性颜色空间中，仅涵盖 0-1的范围。为了支持HDR，我们必须扩大该范围。可以通过将输入颜色转换到Log C空间来实现，这将范围扩大到略低于59。

2. 与线性空间相比，Log C为最黑暗的值增加了一点分辨率，它超过了大约0.5的线性值。之后强度迅速上升，矩阵分辨率降低很多。这时需要覆盖HDR值，如果我们不需要这些值则最好保留在线性空间，否则几乎一半的分辨率被浪费。在着色器中添加一个bool值来控制。

3. 当在使用HDR且应用了色调映射的情况下启用Log C模式。

4. 因为我们不再依赖渲染的图像，所以不再需要将颜色范围限制在60。它已经受到LUT范围的限制。

应用LUT

1. 要应用LUT，我们定义一个Final Pass和相关片元函数，它需要做的是获得源纹理颜色，并将其应用到颜色分级LUT中。在单独的ApplyColorGradingLUT方法中进行处理。

2. 可以通过ApplyLut2D方法应用LUT，该功能可将2D LUT切片解释为3D纹理。该方法需要LUT纹理和采样器作为参数，然后是输入颜色，无论是线性还是Log C色彩空间，最后是参数向量的XYZ分量。

3. 将\_ColorGradingLUTParameters参数传到GPU，3个分量分别是1除以LUT宽度、1除以LUT高度和LUT高度减1，并使用Final Pass进行最终绘制。

最后，我们是否需要在每帧重新创建LUT纹理呢？

只对LUT纹理进行颜色分级和色调映射比单独渲染图像的所有像素要进行的工作少很多，进一步的优化就是缓存LUT。但是确定是否需要刷新LUT可能会变得复杂，尤其是在支持多相机的不同设置或混合设置时。因此我们坚持每次渲染摄像机时都重新创建LUT，URP和HDRP也是这样做的。

LUT条状带

虽然现在我们使用LUT进行颜色分级和色调映射，但渲染结果应该与以前一样。由于LUT的分辨率有限，我们要使用双线性插值进行采样，因此它将原本平滑的颜色转换为线性带。对于分辨率为32的LUT来说，这通常不明显，但在具有极端HDR颜色变化的区域中，条状带将会变得可见。