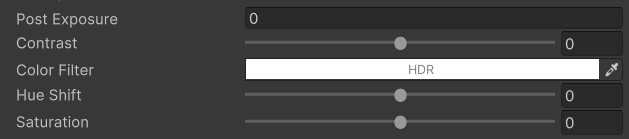
颜色调整有三步：1.色彩矫正，让图像与观察的一致；2.颜色分级，定制颜色；3.色调映射，将HDR颜色映射到显示范围。

比如，我们12中将颜色限制为60以下的操作就可以算作颜色分级，将其分离出来。并且让色调映射的None模式也使用一个Pass执行该操作。

添加ColorAdjustmentsSettings，可配置后处理曝光，对比度，颜色过滤，色相转变和饱和度。

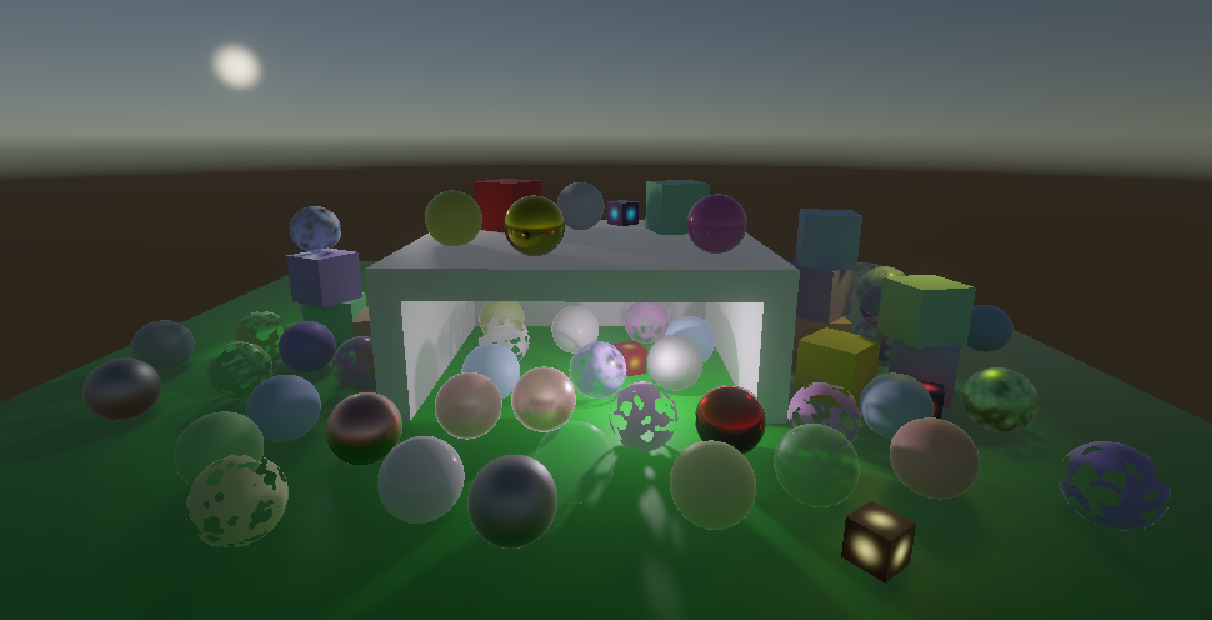
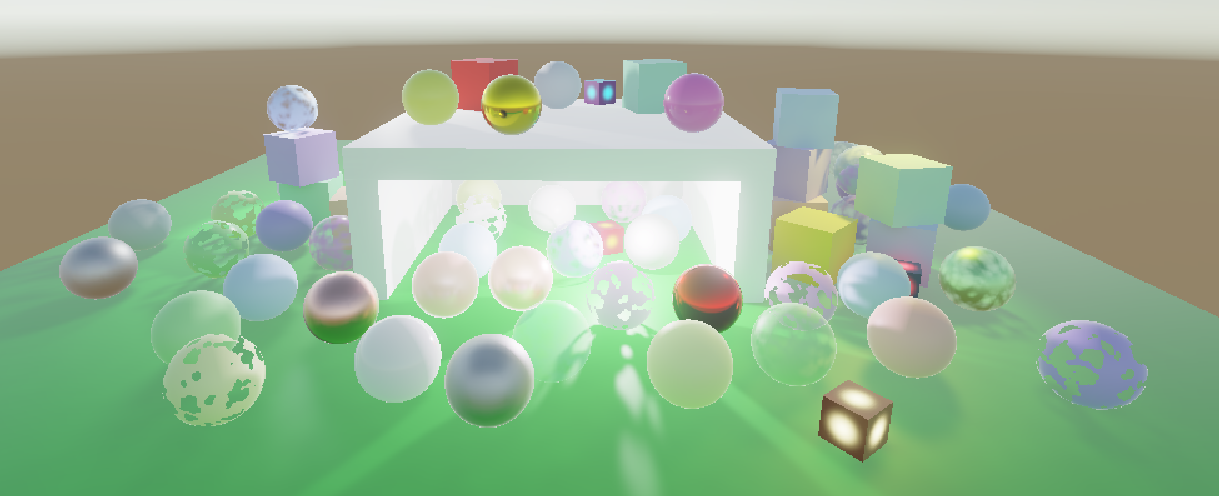


知识点，using一个类名时，要加static：

using static PostFXSettings;

在tonemapping之前，把颜色调整的参数设置为uniform。

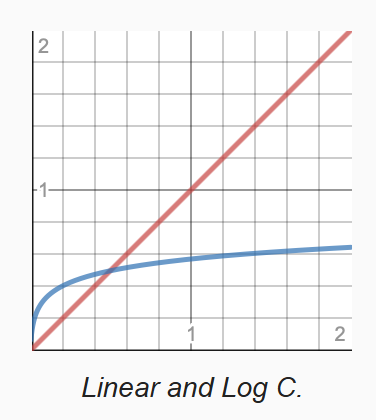
曝光值为w，则将颜色乘上2的w次幂（下图为w=-2和w=2的情况）：

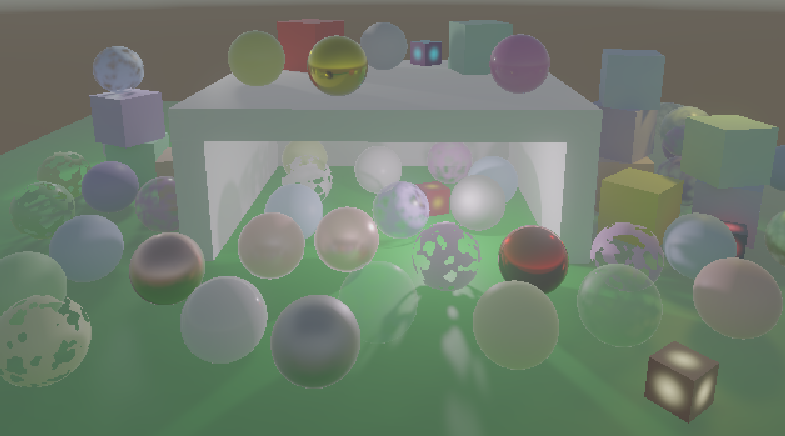
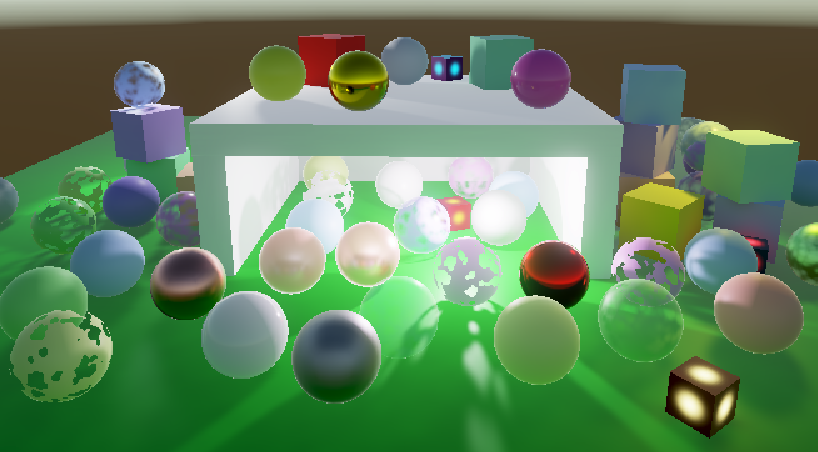
 

它在所有后处理效果后，但在所有其他颜色分级之前执行。

对比度是将颜色减去中间灰色值，再通过对比度向两端缩放，最后加回灰色值实现的.

该部分在LogC颜色空间实现，而不是线性空间。



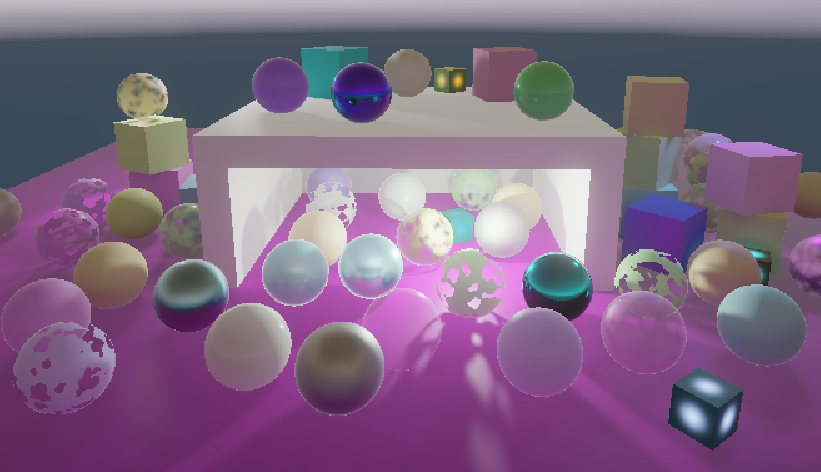
 

颜色过滤就是将配置的颜色直接相乘即可：



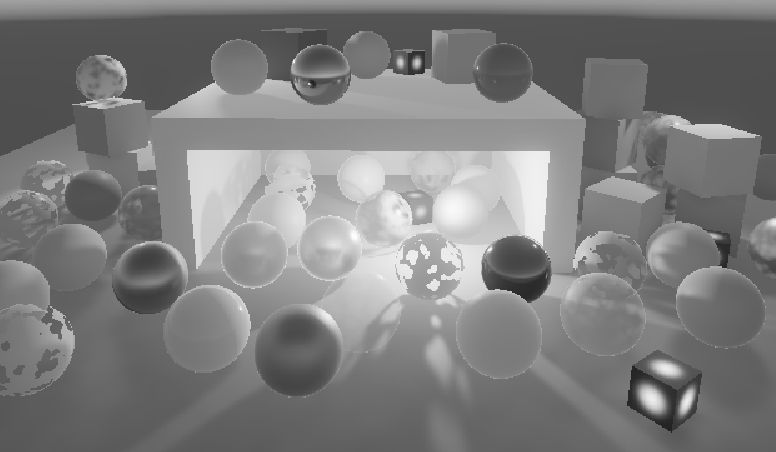
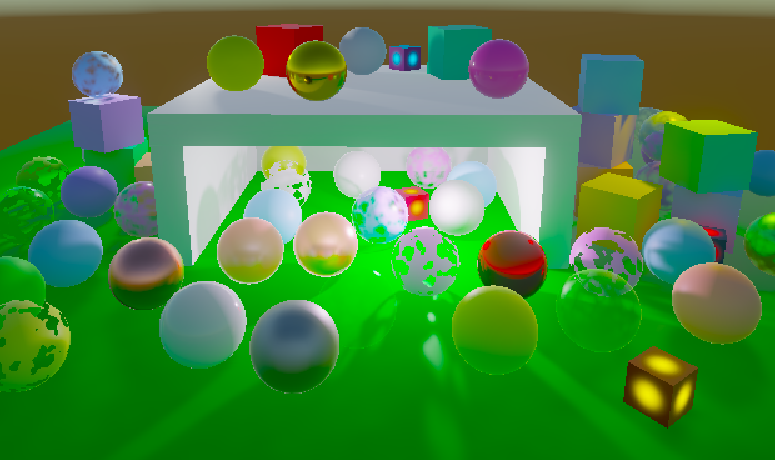
处理完要删去负数颜色。

将颜色从RGB表示转化为HSV表示，色相偏移应用在H分量上，再转化为RGB值。



饱和度和对比度应用类似，只是不算减去中灰度，而是减去亮度，而且在线性空间处理。

处理完要删去负数颜色。

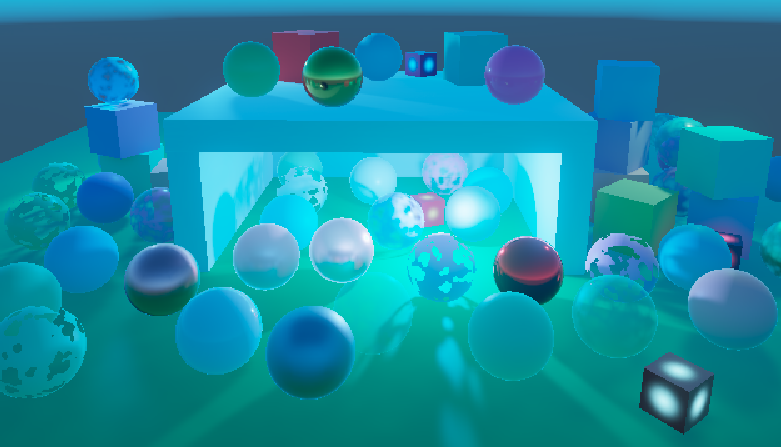
 

接下来，实现一些其他设置。

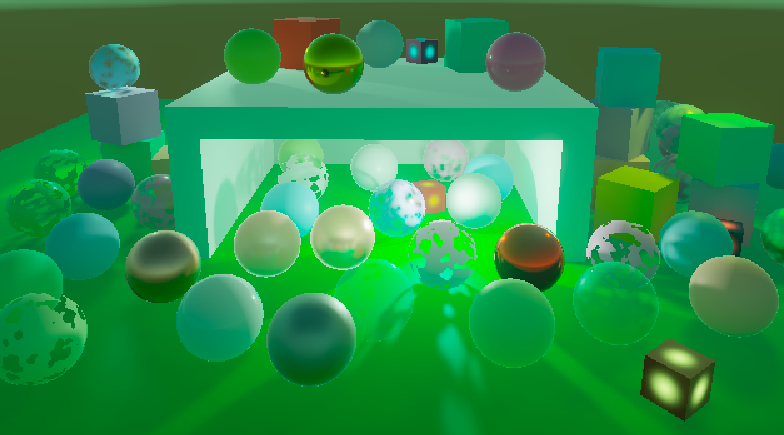
白平衡设置，一个变量控制色温，另一个变量控制色调，调用**[ColorUtils](https://docs.unity3d.com/Packages/com.unity.render-pipelines.core@14.0/api/UnityEngine.Rendering.ColorUtils.html)**.ColorBalanceTo LMSCoeffs将参数转化为颜色，然后在LMS空间应用上去。

该步骤在应用曝光之后执行。

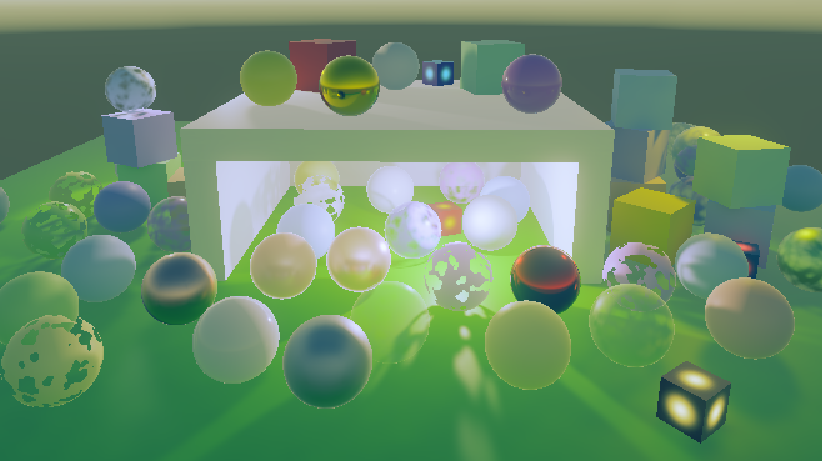
Temperature:

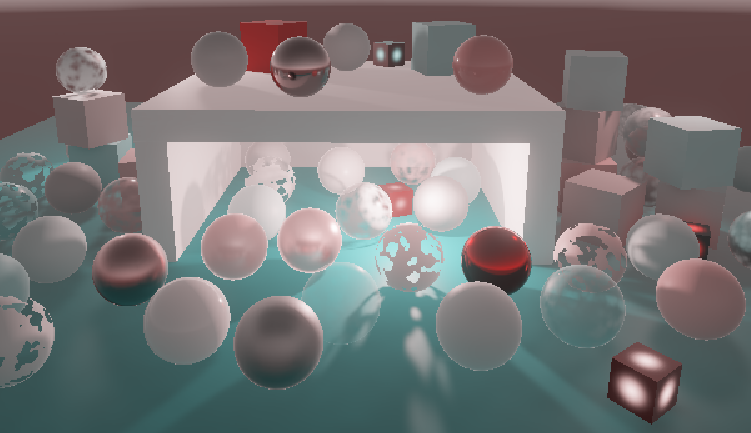
Tint:

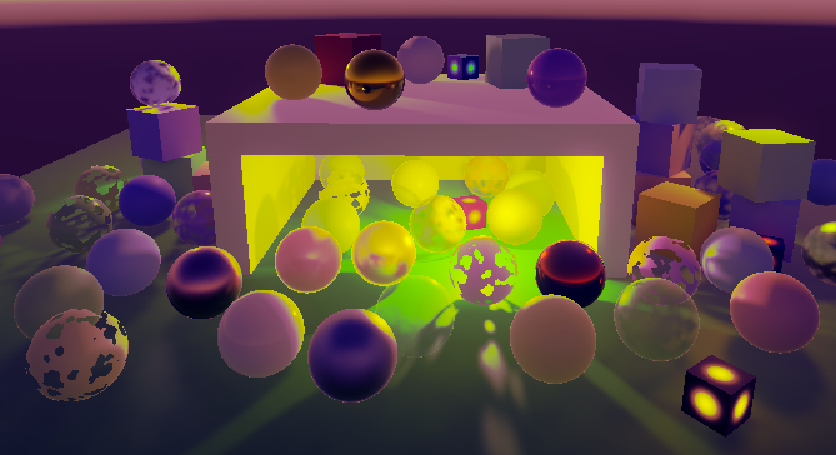
分体调可以单独配置阴影和高光的颜色。原理是先根据亮度调整配置的阴影和高光的颜色，再进行柔光混合。该步骤在颜色过滤之后。青橙色调：



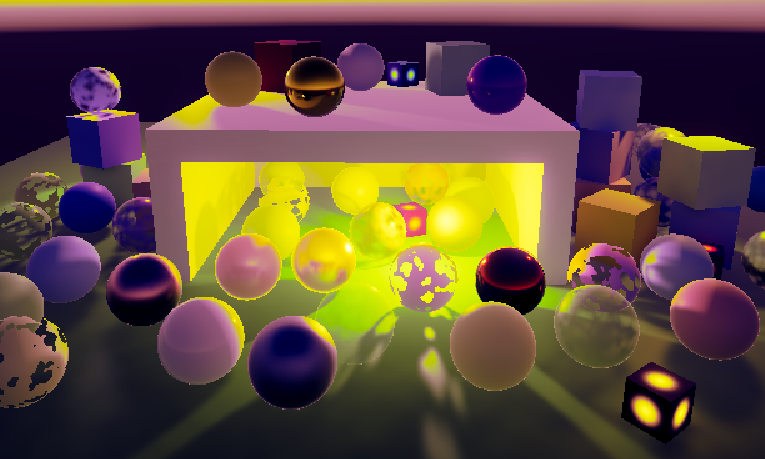
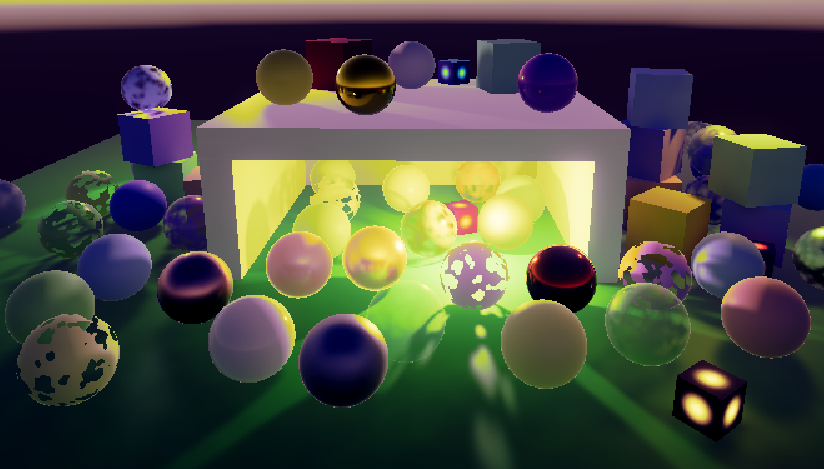
通道混合器，可以将RGB通道当作输入变量，通过变换转化为新的RGB值，本质上是执行3x3的矩阵变换。该步骤在分体调之后。让绿色平均到GB通道，而蓝色平均到RGB通道：



阴影、中间调、高光调整，用户定义各部分的分界点，然后根据亮度的范围，阴影的权重会从1变为0，高光的权重会从0变为1，中间调的权重为1减去前两者之和。该步骤发生在通道混合之后。



当使用ACES色调映射时，上述大部分操作应该在ACES色彩空间进行，我们进行调整。然后在使用ACES色调映射时开启选项。

上图左是在ACES空间处理的情况。

接下来，我们可以把各种配置烘焙进一个lookup table，称之为LUT。它是一个32x32x32的3D贴图。对它采样比对图像执行操作的工作量要少很多。

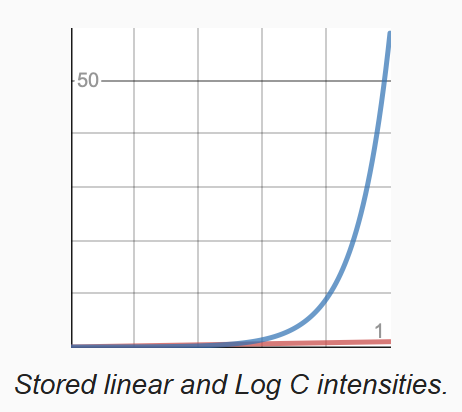
我们使用排成一排的2D纹理来替代3D的。高度为x，则宽度为x\*x。

调整渲染目标为LUT，并通过GetLutStripValue获取输入颜色：



输入颜色改为Log C空间（仅当处理HDR时这么做）：

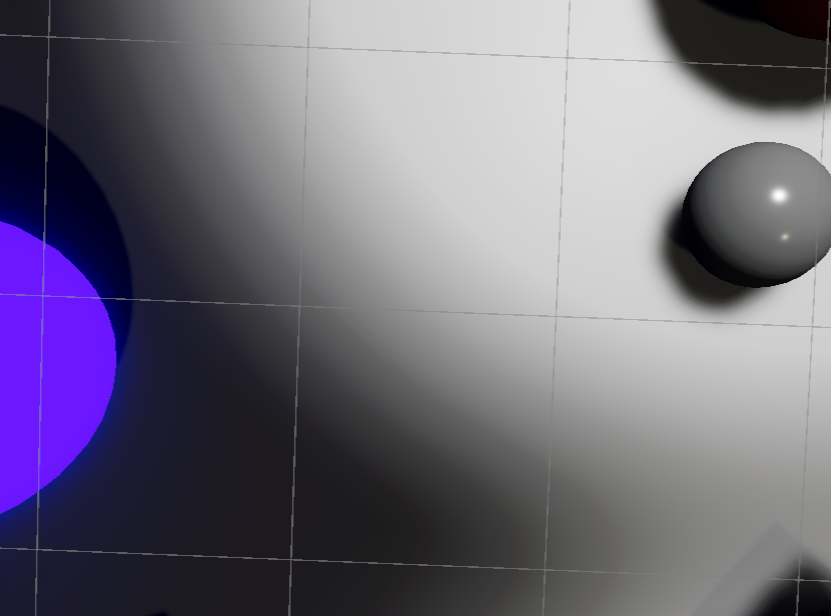
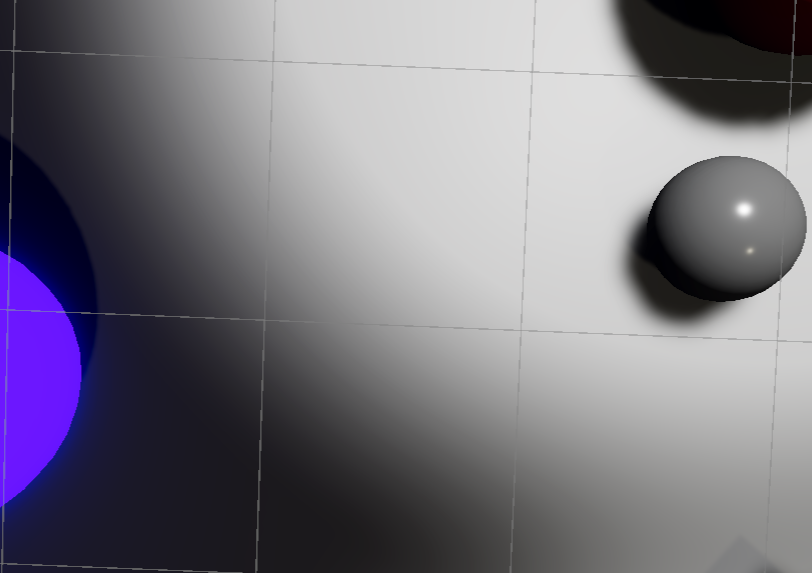




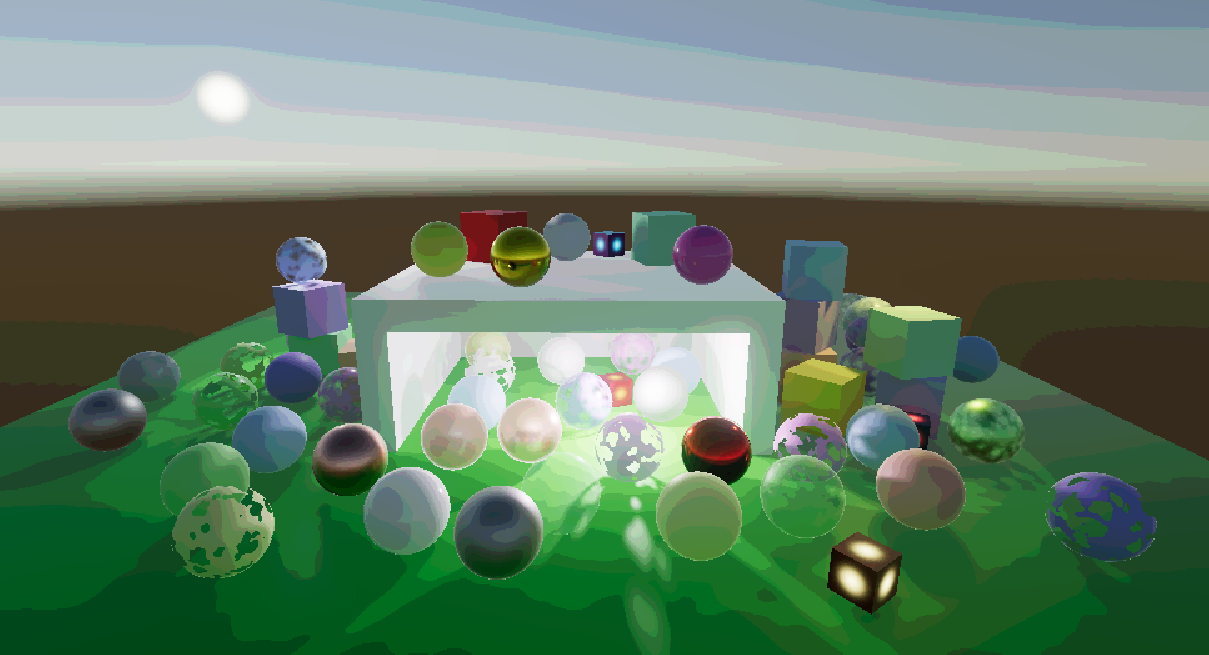
我们通过ApplyLut2D在最终Pass里读取2DLUT贴图并应用到当前颜色上。

因为生成LUT是很快的，而决定什么时候重新生成LUT很复杂，所以我们直接每帧都生成即可，URP和HDRP也是这么做的。

因为我们使用低分辨率的LUT，并且使用双线性采样，所以在具有极端HDR颜色渐变的区域，会有色彩断层，提高LUT分辨率可以缓解：

使用sampler\_point\_clamp可以关闭2D贴图内的线性插值：



不过，实践中一些条带的产生原因是8-bit帧缓冲区带来的限制，可以通过抖动解决。

最终结果：

