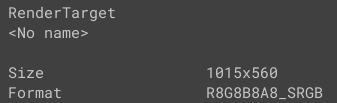
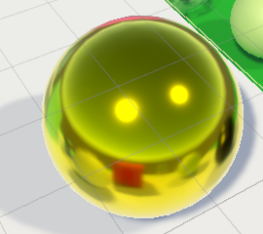
本部分将处理HDR绘制、基于散射的bloom和色调映射的支持。

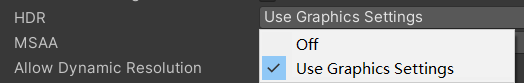
在LDR范围中，颜色被限制在0-1的范围。



HDR绘制需要HDR的render target。比如反射探针就可以配置HDR输出，下面的左图是HDR输出，可以包含高强度颜色：

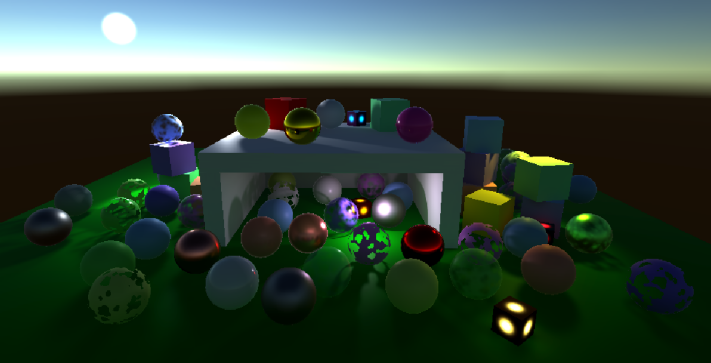
Camera也有HDR选项，该选项是否影响HDR取决于我们的自定义RP配置：



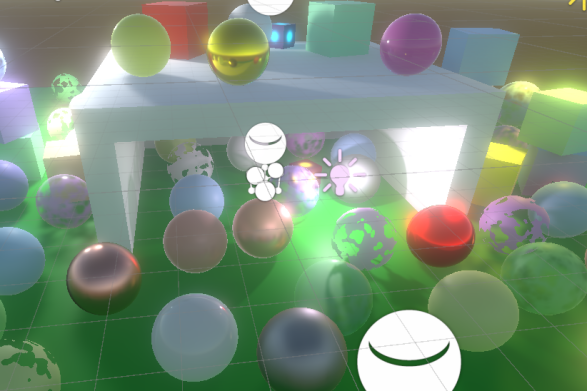
我们在RP asset里允许用户开启该配置，然后把该选择通过RP传给Camera Renderer，当相机开启HDR且Asset允许HDR时，Camera Renderer的HDR布尔值启用。



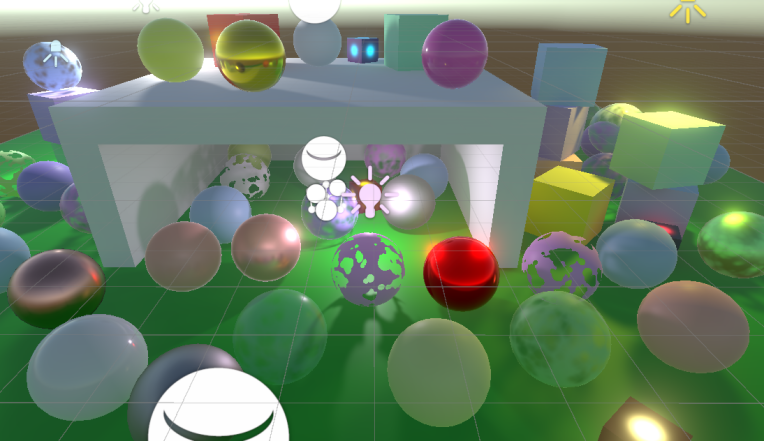
在开启后处理时，我们把自定义的帧缓冲区贴图改为HDR格式（会发现RT变为了R16G16B16A16\_SFloat ），进入后处理前的颜色变暗，这是因为它仍然按线性颜色显示：



把Bloom中创建的贴图也按需改为HDR，查看结果：

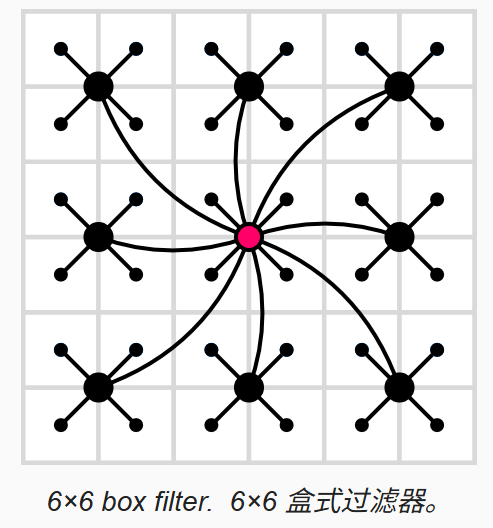
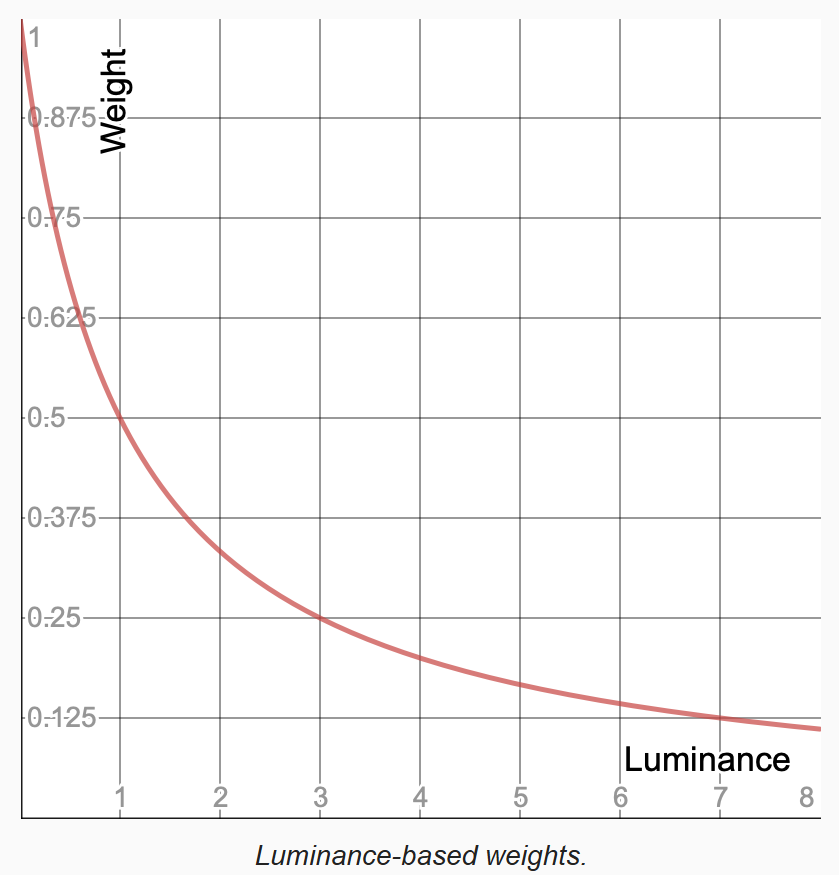


对于HDR Bloom，一般会把threshold设置为1而knee设置为0，这样只有HDR的颜色会影响它：

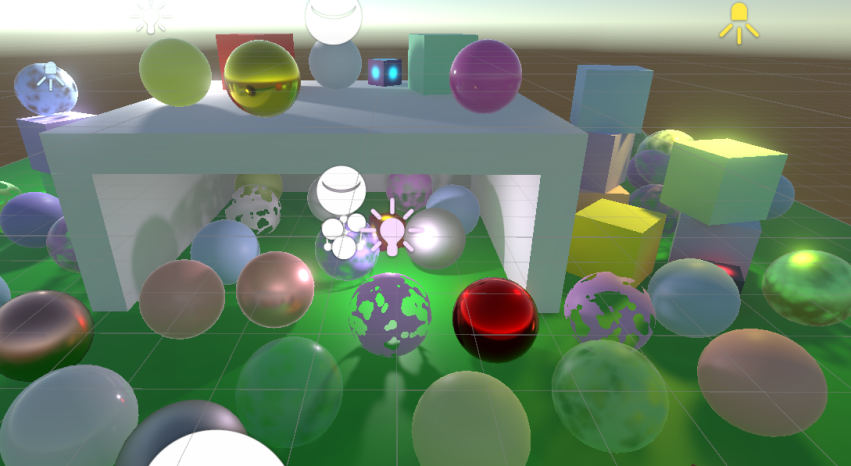


因为HDR会存储大于1的颜色，所以会导致一些区域过亮，并且在改变视角时大小变化过快，进而在Bloom的影响下产生闪烁。

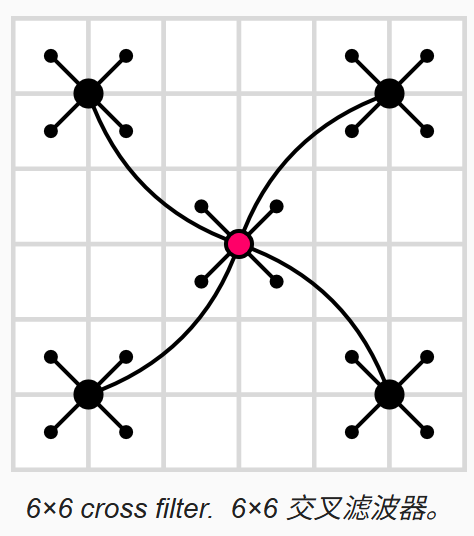
方法是在最开始下采样的时候施加一个3x3的平均过滤器，在取平均之前ApplyThreshold，在取平均时，使用的是基于感知亮度的加权平均：

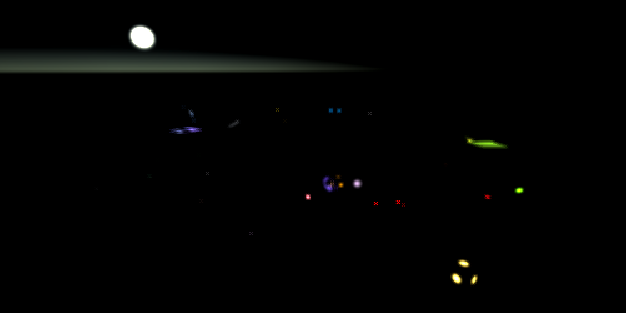
得到结果：

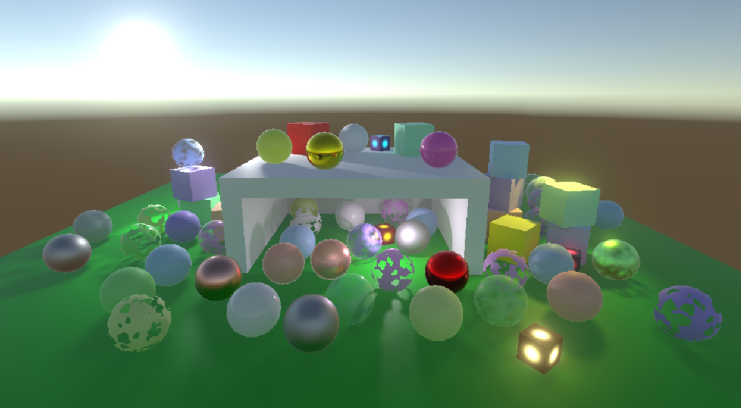


考虑到我们会后续对该结果进行高斯模糊，所以可以降低采样点，变为：



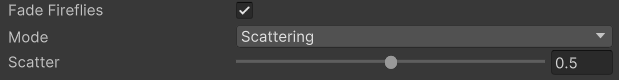
中间结果和最终结果为：



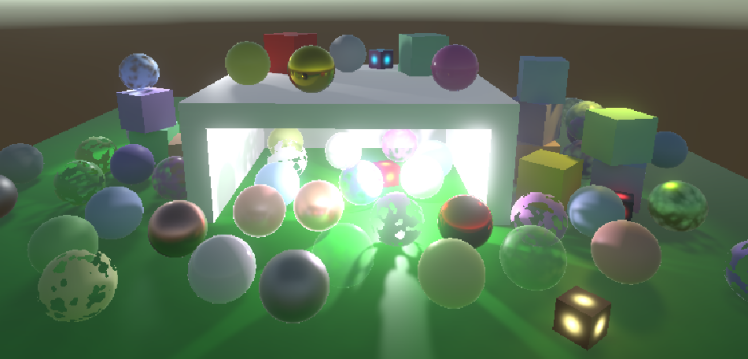


虽然这不能完全消除频闪，但是可以大大降低过亮的点的强度。

下面我们将支持其他的Bloom模式，之前的是加法上采样得到的，下面尝试支持散射Bloom，后者是Energy-Conserving的。



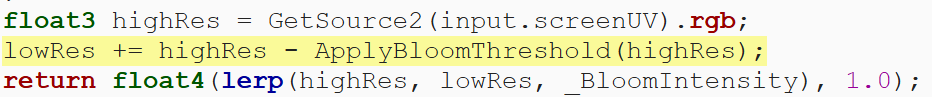
在结合时，使用lerp而不是加法，scatter为0表示仅使用更低层级的金字塔，为1表示仅使用更高层级的。

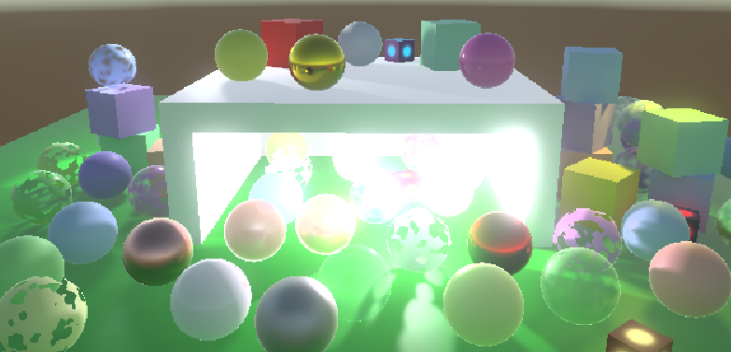


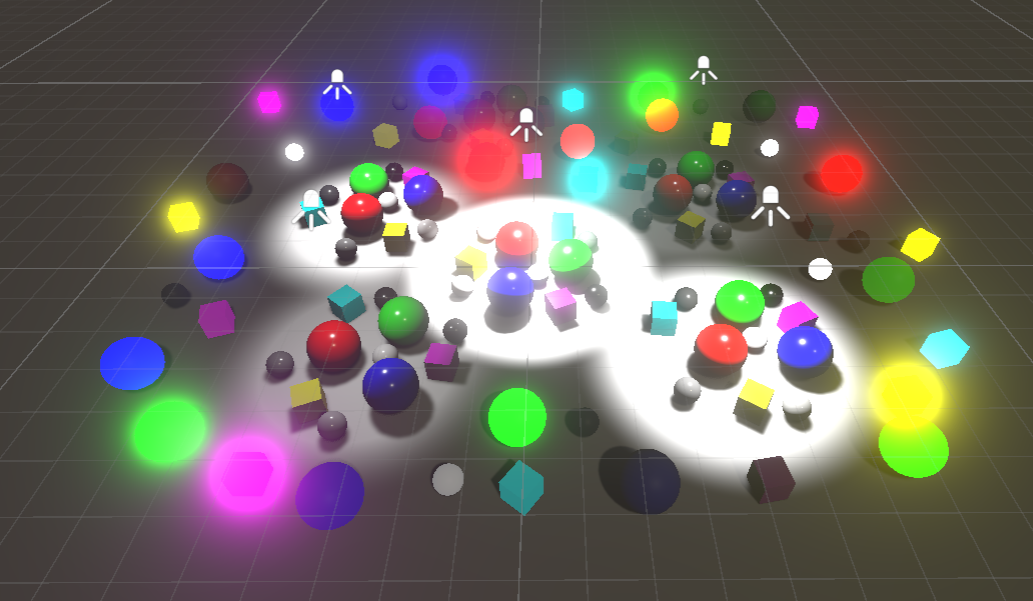
使用较强的强度时，可能会使得整体变暗：



因此，对于最后一步的Combine，要单独处理，对于模糊后的图像，把原来的光线加回来：



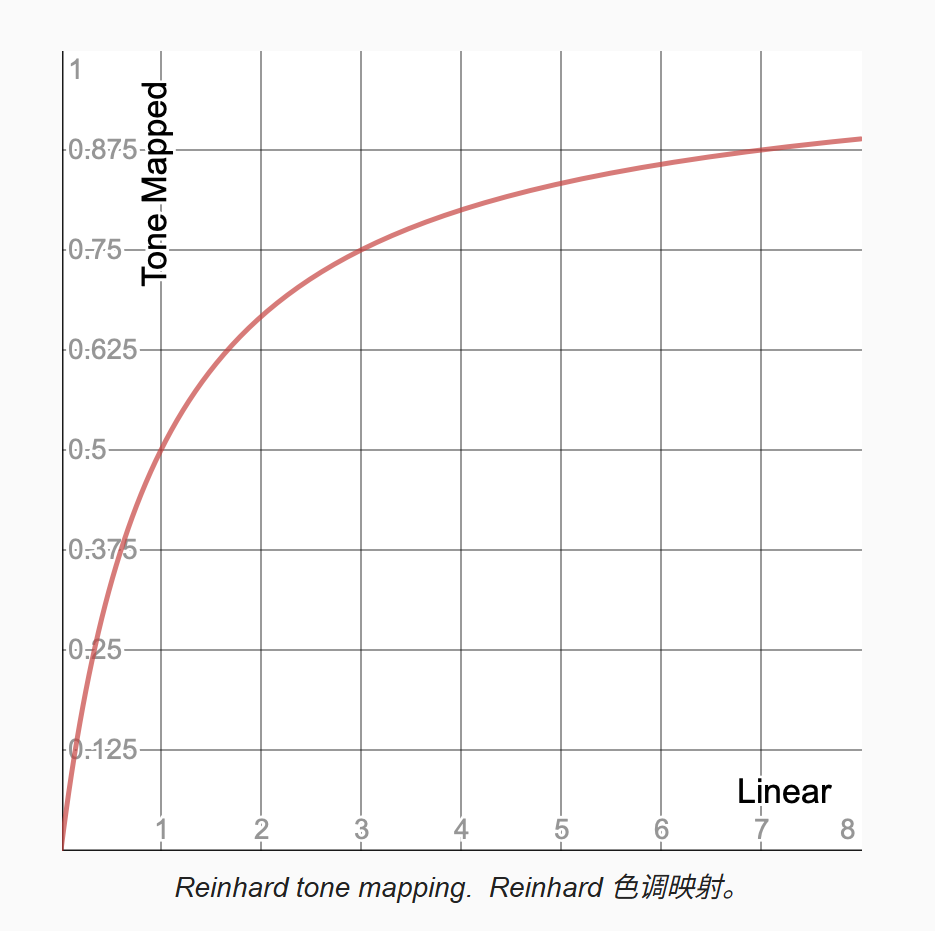




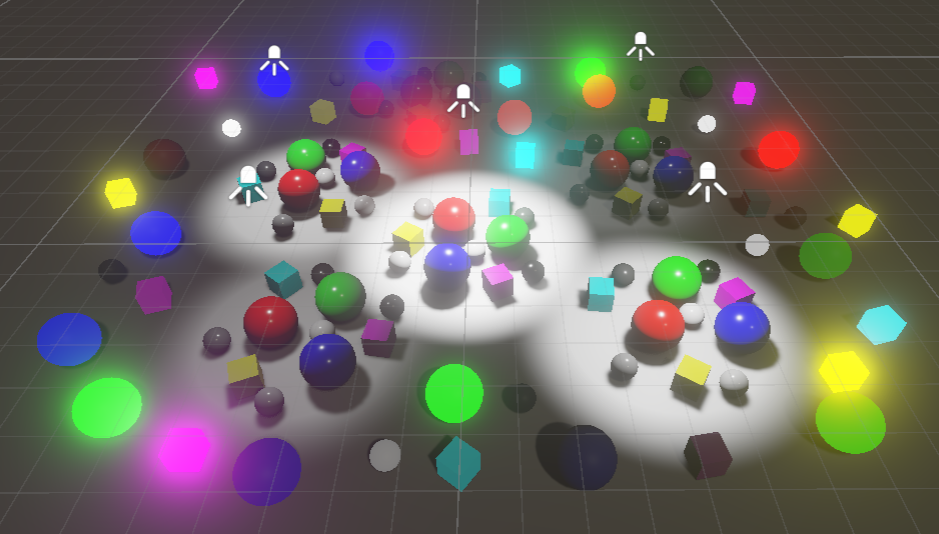
尽管我们中间结果是HDR的，但是最终输出还是LDR的，存在截断。因此，我们需要进行非均匀地颜色调整策略，对非常亮的颜色调整更多。这种从HDR到LDR的映射称为色调映射（tone mapping）。我们把它作为新的FX加入（DoToneMapping）。

URP和HDRP把tone mapping结合到了Uber pass里，不过我们的实现将其分离。

莱因哈特（Reinhard）色调映射函数如下：

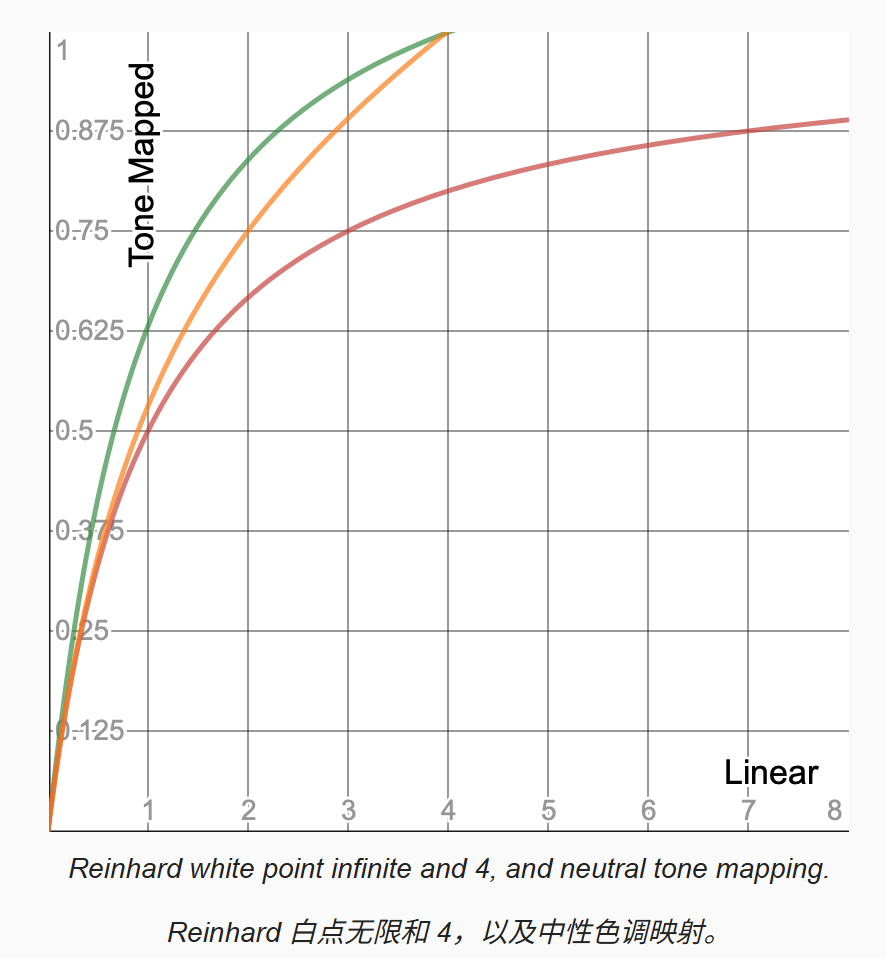


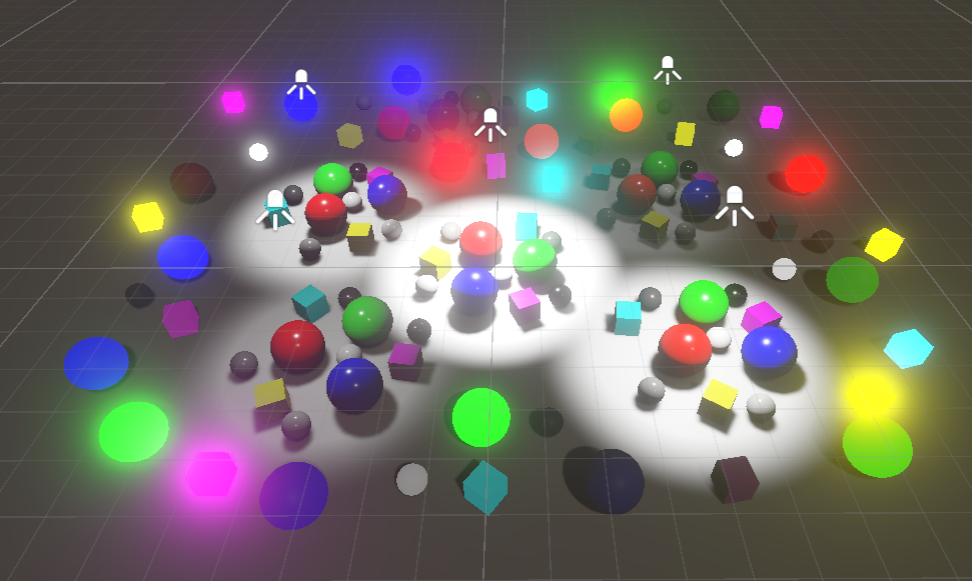
我们将该函数应用到颜色的每个通道，结果如下，高亮部分的细节被保留：



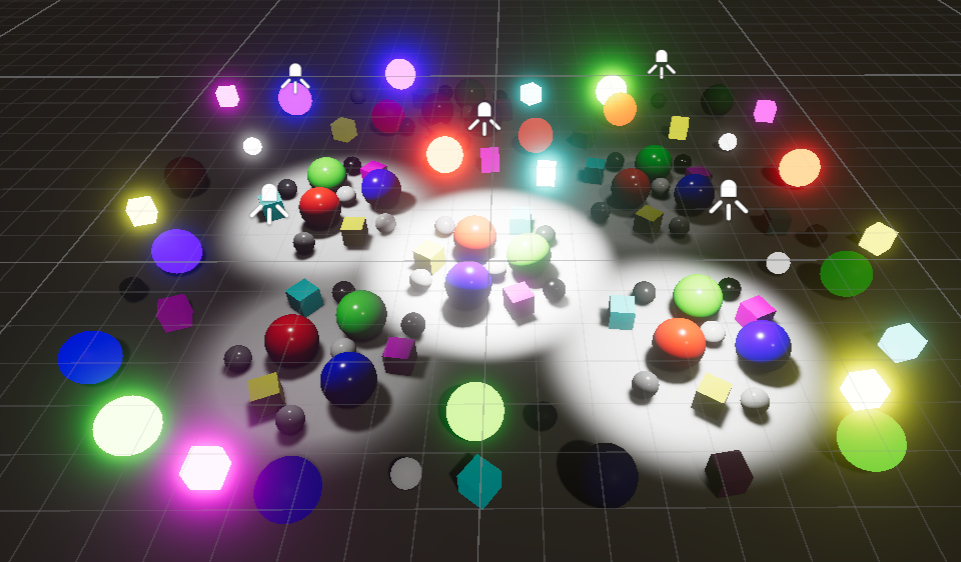
考虑到浮点数精度，我们在应用函数前把color的最大值限制为60。

还有一种Neutral映射，让白色点对应4.035，可以从库中调用该函数：



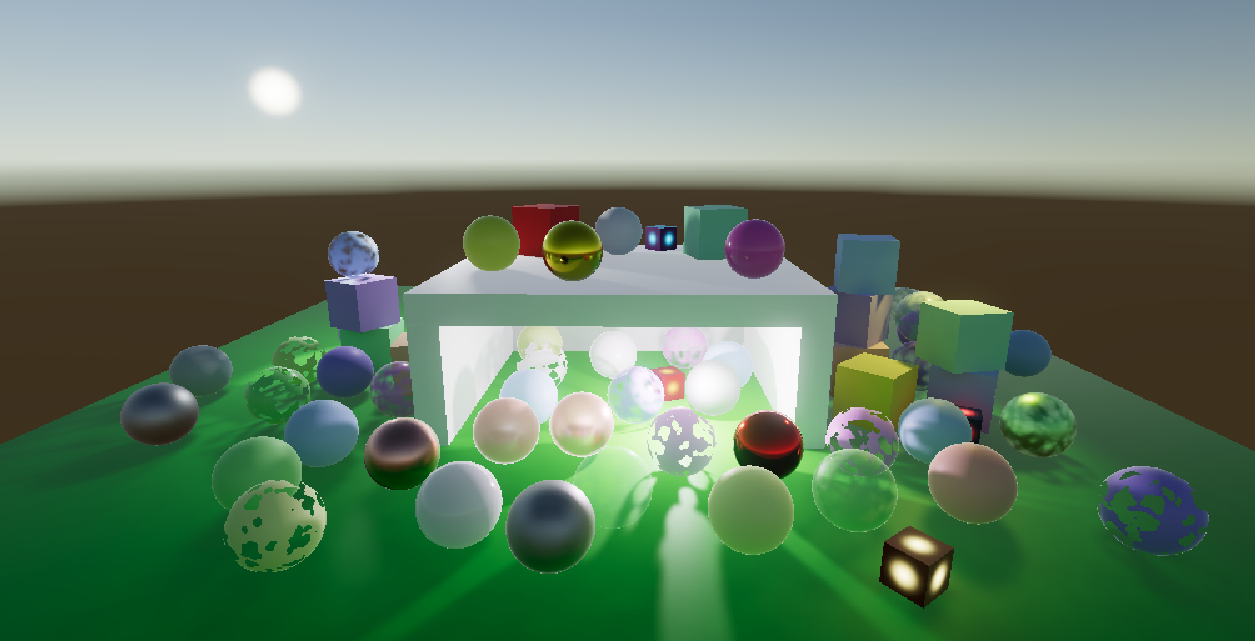


最后一个模式是ACES(Academy Color Encoding System)。也是可以直接调库。



可以看到，对于过亮的颜色，它应用了色相偏移。

最终结果（Scatter Bloom + HDR + Tone Mapping）：



注意到，对于镂空物体的Mesh Renderer，我们需要将其Lighting的设置改为TwoSided，防止背面不被绘制：

