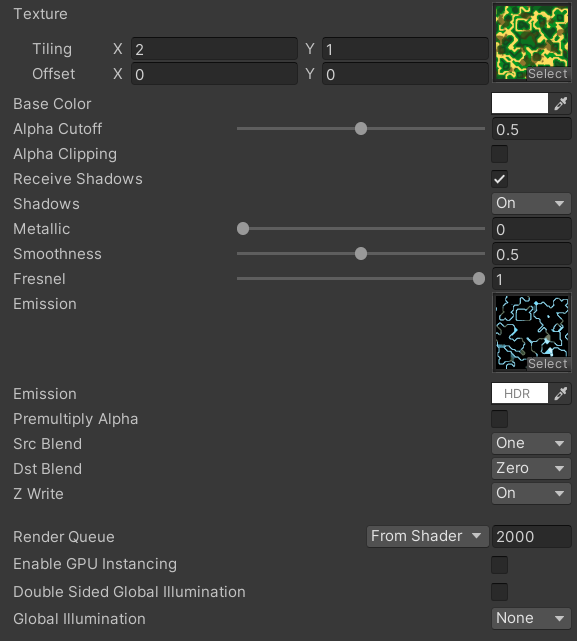
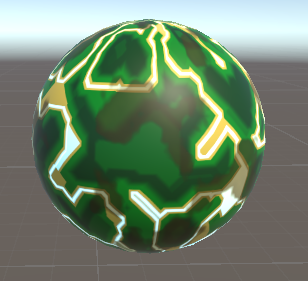
该部分将支持更多种类的材质贴图。

首先，我们已经支持了Albedo和Emission，添加一个材质，带有这两种贴图：



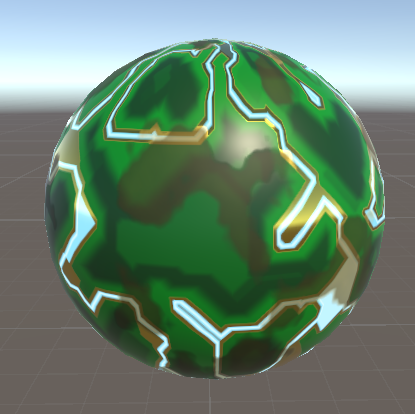


Unity的HDRP有一种MODS贴图，四个通道分别代表Metallic（金属）、Occlusion（遮挡）、Detail（细节）和 Smoothness（平滑度），我们来实现一下。

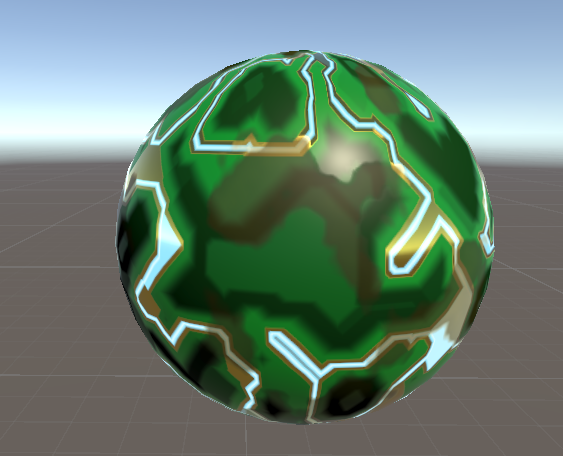
目前先只使用Metallic和Smoothness。这种贴图使用时需要关闭sRGB选项。

通过贴图传入MODS Map，然后在对应属性取值时乘上该贴图的对应通道值。

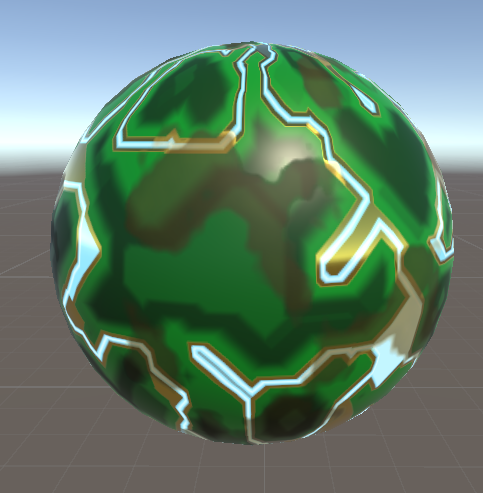
可以看到，金色部分是偏金属和平滑的，绿色部分相反：



对于Occlusion属性，它表示物体的这个区域接收间接光的程度，也就是物体表面的特点导致其间接光强度的变化，为Surface添加对应属性并实现：



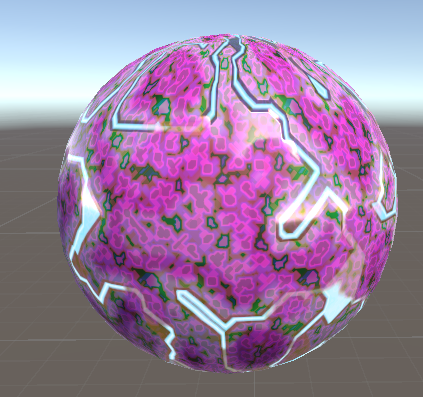
可以单独添加一个属性控制该项的强度，下图为Occlusion设置为0.5的情况：



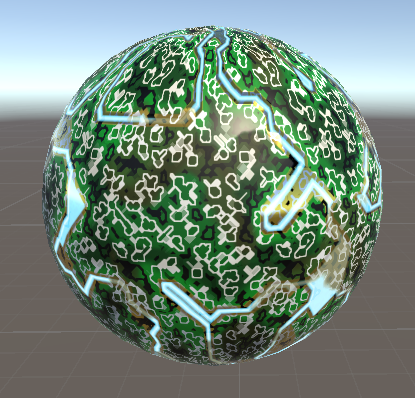
下面实现Detail Map，HDRP使用ANySNx格式，对albedo和smoothness的调整在R通道和B通道中，细节法线的XY分量对应AG通道。这里我们只实现RB通道。

这张图的默认值为vec4(0.5)，表示没有变化。

先直接把采样结果加到albedo上看看效果：



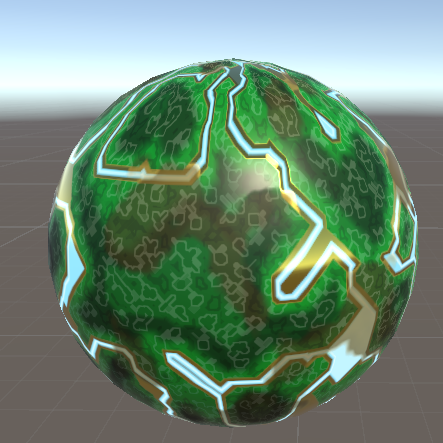
我们把采样结果映射到-1到1，然后对baseMap进行调整，负数会将其变黑，正数会将其变白，插值仅对rgb操作，并且在gamma空间插值：



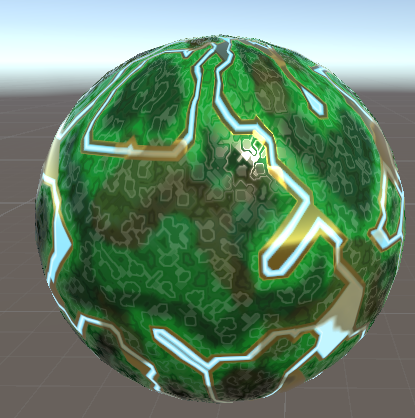
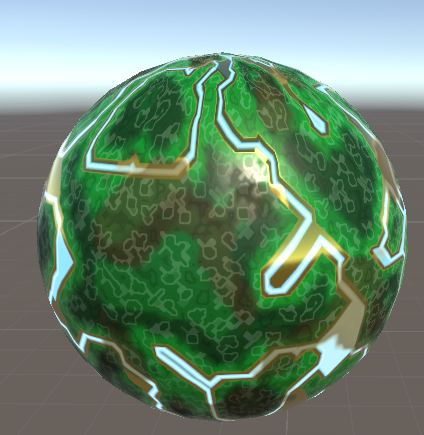
再把之前MaskMap的b通道，作为细节的遮罩：



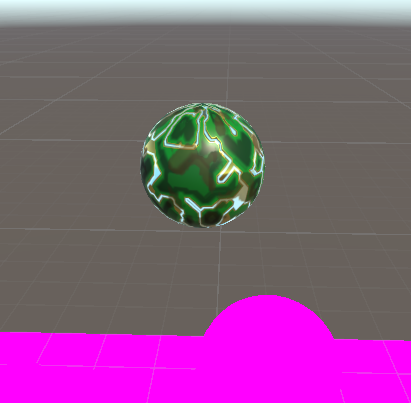
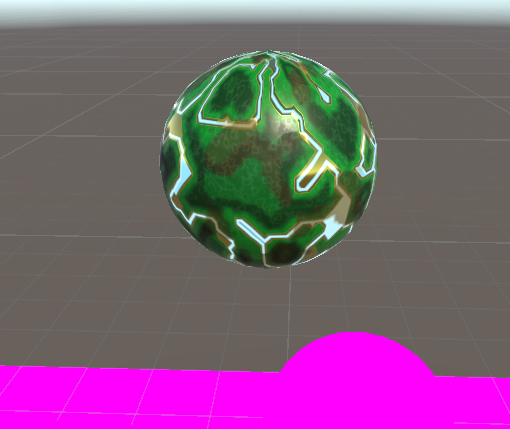
再引入一个可配置的参数控制该细节的整体强度：



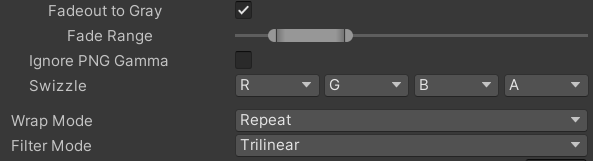
一样的步骤，可以加入Detail Smoothness：

为贴图施加MipMap和Fade out to gray，可以在细节过小时，不应用它们，进而避免噪声：



Fade Range控制淡化开始和结束的mip级别：



注意到贴图用了三线性插值，与双线性比，多了mipmap的插值。

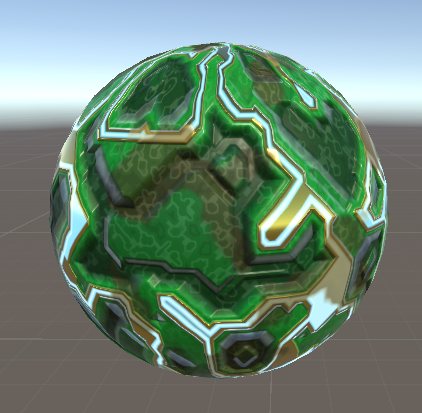
下面开始实现法线贴图：

对于贴图资源，导入后要设置类型为Normal Map。

法线贴图的默认值为“bump”

法线贴图有时会被打包存储，因为如果法线一直朝上的话，只需两个分量即可推出第三个分量，因此可以修改一些通道的位数实现打包。在Common.hlsl里的DecodeNormal有具体解包的调用。

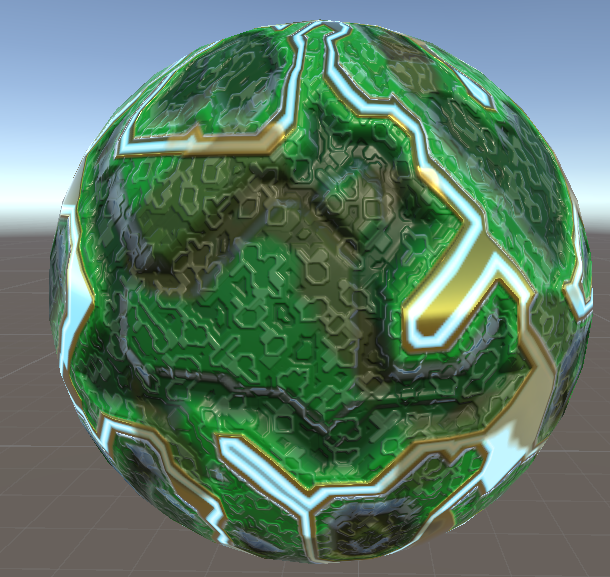
因为法线贴图存储的是切向空间的法线（所以默认值是up axis），所以我们需要先使用顶点存储的切向向量和法线向量计算出切向空间到世界空间的变换矩阵，再用该矩阵变化法线贴图的采样结果：（顶点存储的tangent是vec4，第四个分量控制切向空间的z轴朝向，即-1或1）



注意到，在执行阴影计算时，我们需要使用原来的Normal来进行bias偏移，为此，我们需要在Surface里记录原来的法线，并在GetCascadedShadow中使用该向量。

与HDRP不同的是，我们使用单独的贴图实现Detail Normal。因为对于Mipmap处理而言，NormalMap与其他种类的贴图是不一样的。所以需要自己额外写这部分代码（并且fadeOut和Unpack也要自己写）。因此，就用额外一张贴图来简化操作。

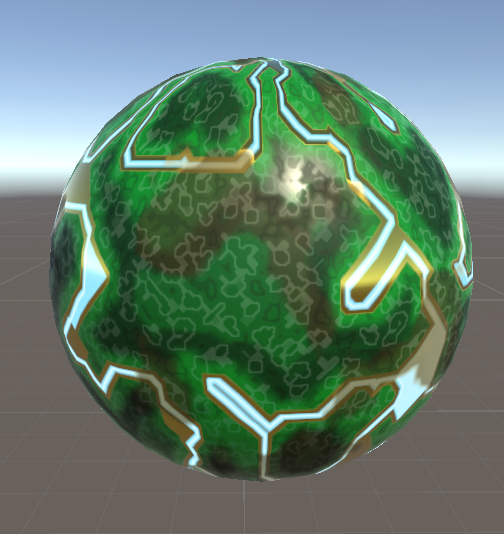
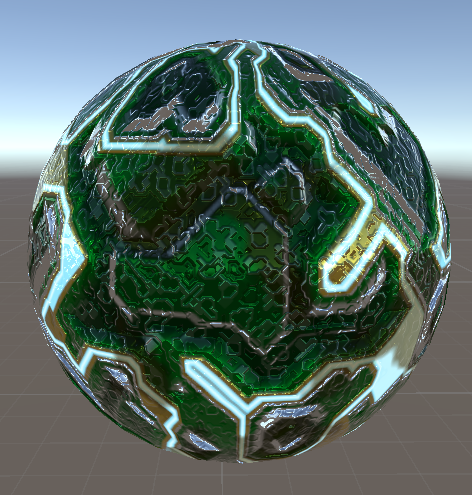
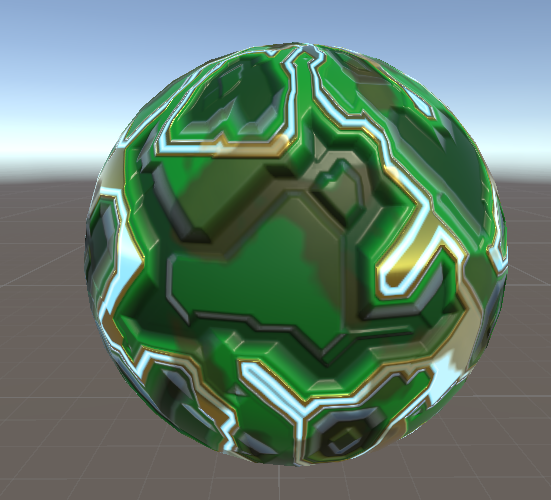
与颜色用的lerp不同，法线需要使用BlendNormalRNM来实现细节法线的应用：



下面，尽管Unity会根据贴图是否分配来决定是否进行贴图采样，但是我们可以显式实现这一控制（通过Toggle）。

再次Callback：1、multi\_compile —— 无论如何都会被编译的变体 2、shader\_feature —— 通过材质的使用情况来决定是否编译的变体。

实现Normal、Mask和Detail的开关，主要是一些采样和变量的代码分支。

最终成果：

