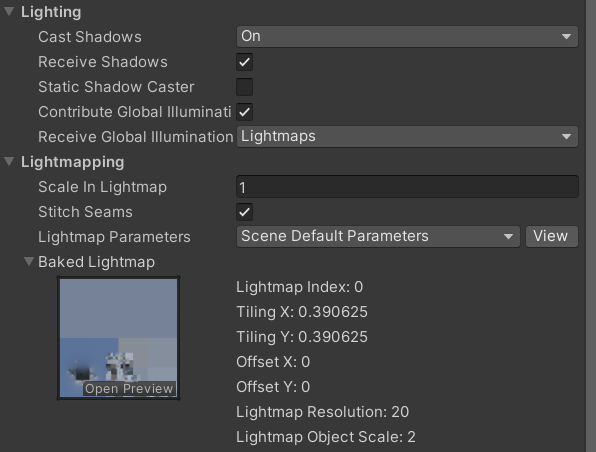
通过Window-Lighting-Rendering配置烘焙等光照选项。打开后，若缺少Lighting Settings则新建即可，一般和场景名相同。

在Mixed Lighting下启用全局光照；同时，对于场景中需要烘焙的光源，把它的Mode设置为Mixed。

对于物体而言，启用“Contribute Global Illumination”来将其纳入间接反射的贡献项。



对于动态物体，可以考虑使用光照探针技术，该部分之后再讨论。

这里的贴图是蓝色的，主要是因为天空盒占了很大一部分（作为间接光照中的环境光照存在）。

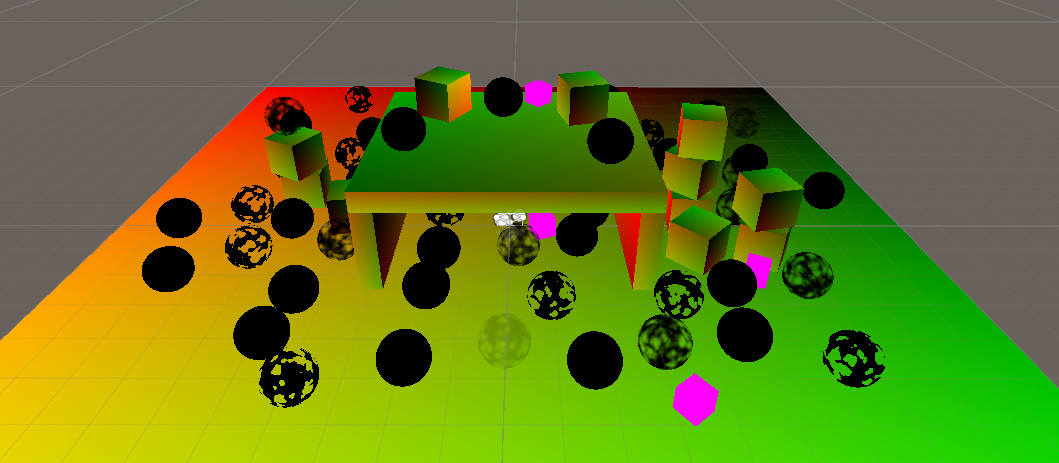
将前面的Light的Mode设置为Baked，可以把直接光也烘焙进贴图中，此时贴图更亮一些：



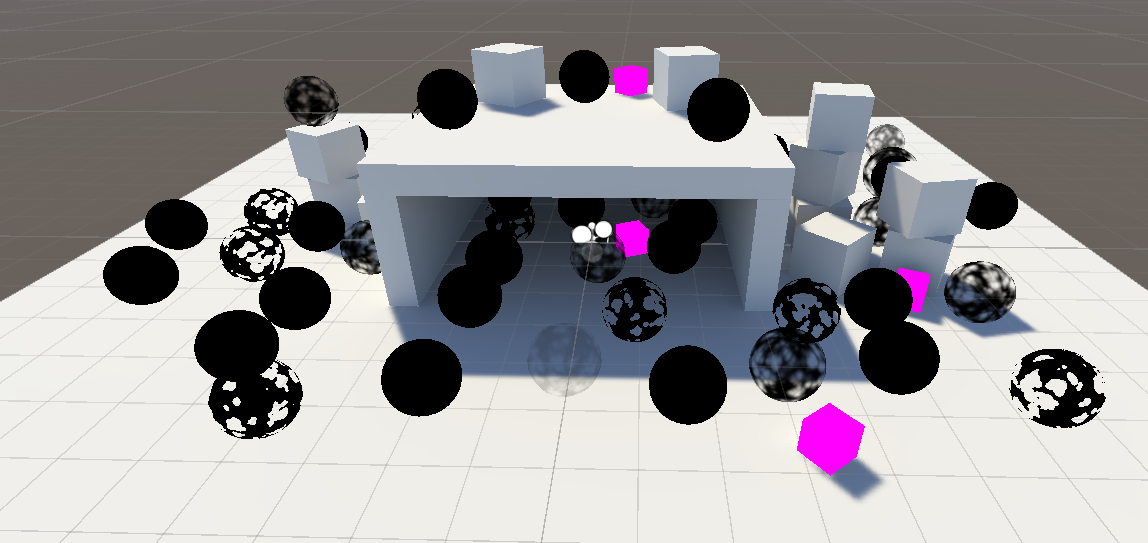
光照贴图一般影响diffuse项，而反射探针影响specular项。

若要采样光照贴图，我们要通过管线把对应的UV传给shader，对每个使用该方法的物体都要做此操作（在drawSettings里设置）。设置好后，shader会通过关键字识别是否lightmapped。然后在attribute里填入对应宏，以获取UV。

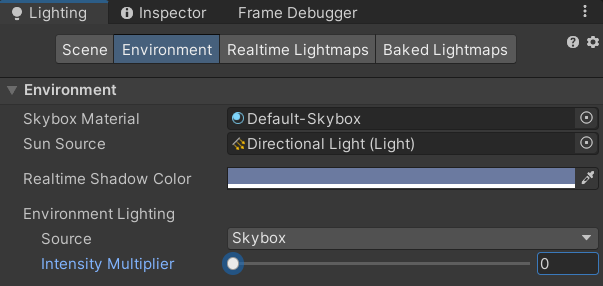
把UV作为结果输出，可以看到静态物体都有颜色了：



调用core rp里的接口采样烘焙的光照贴图，得到如下结果：

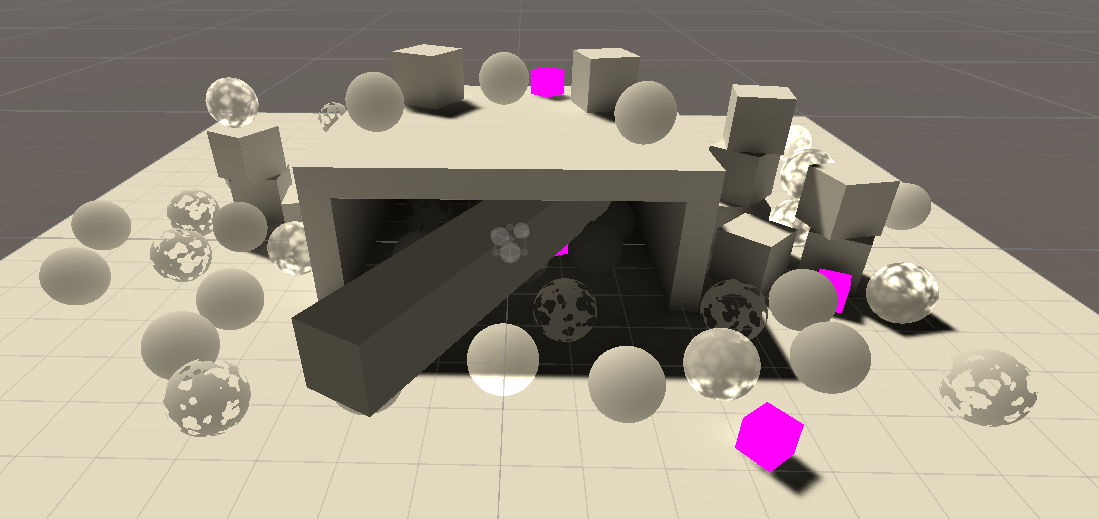


接下来，把天空盒的贡献减少到0来专注于方向光：



下面使用光照探针来影响动态物体的光照，光照探针是场景中的点，通过L2球谐来烘焙入射光。在Unity中，通过GameObject/Light/Light Probe Group添加。这些探针通过四面体连接，物体通过判断自身所处的四面体对顶点的探针进行插值，得到光照结果。（记得在DrawSettings里添加）

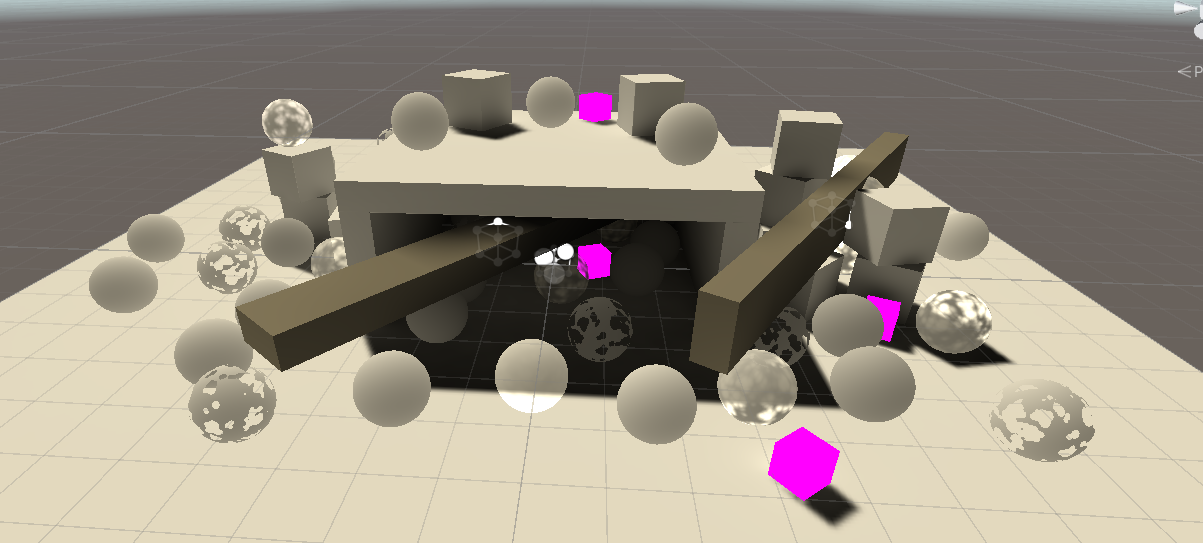
探针只适合物体比较小的情况，如果物体比较大，则会出现问题：



下面通过LPPV(Light Probe Proxy Volume)解决。

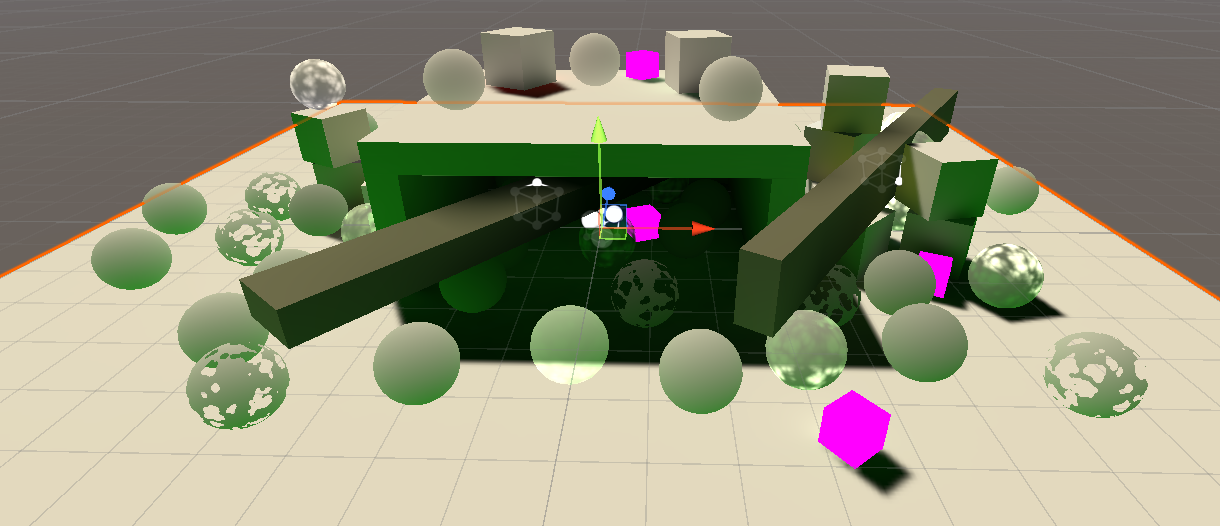
为这些物体添加LPPV组件，并把它们的Light Probes模式设置为Use Proxy Volume。

同样在DrawSettings，CBUFFER还有GetGI里添加对应调用。

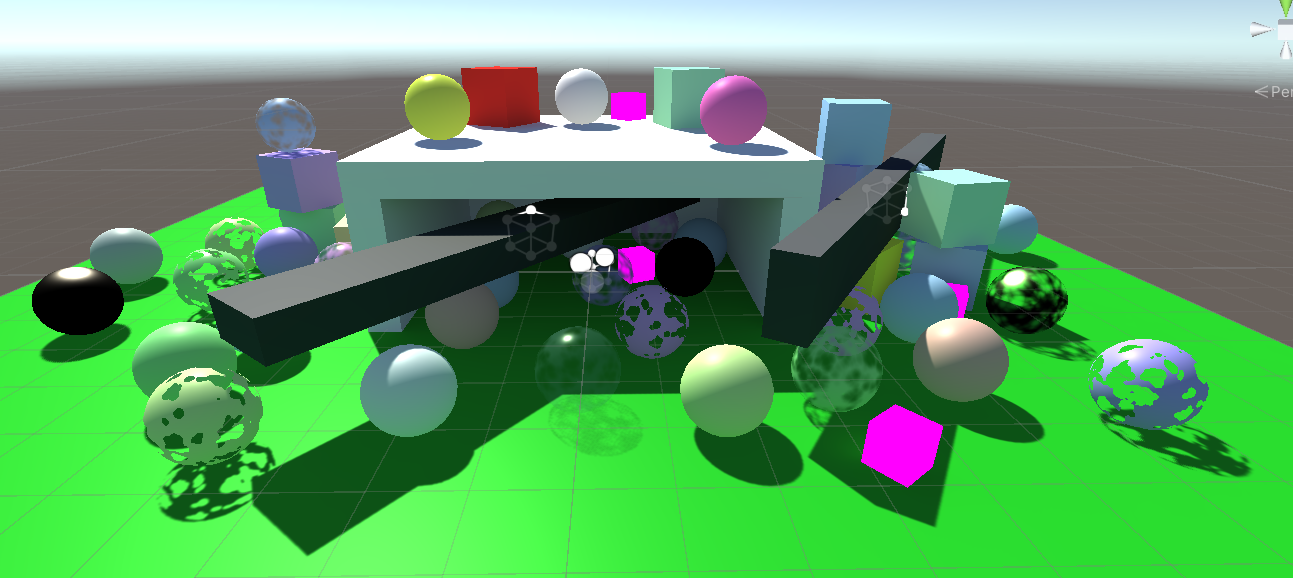


unity通过指定的meta pass确定烘焙时的反射光，默认为白色。

我们先将材质参数的接口提取出来，再为每个Shader添加MetaPass，在该Pass中，先计算得到物体的brdf数据，在根据unity的设定计算得到返回值meta，就可以正确显示间接光照了：



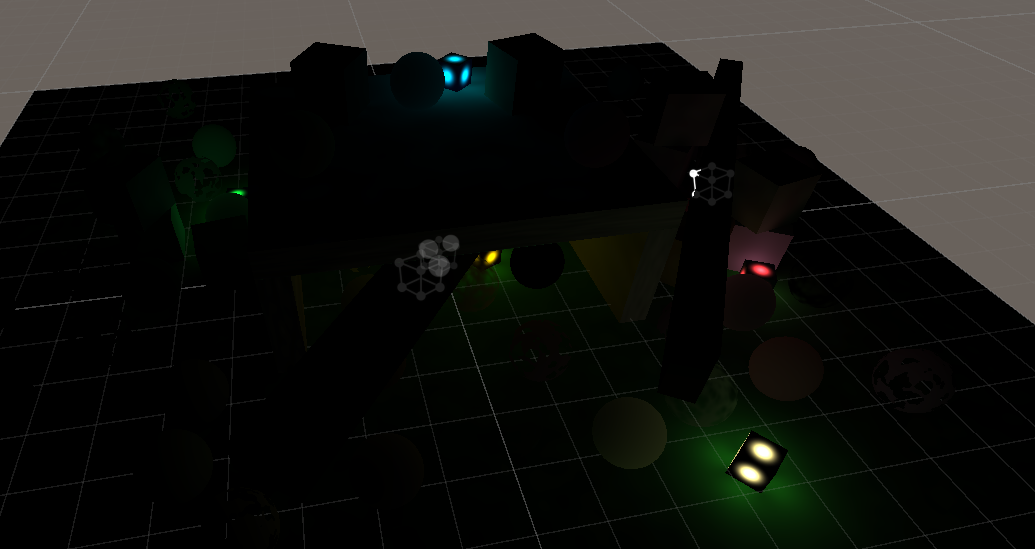
然后在计算全局光照的地方，把结果乘上brdf.diffuse。然后打开天空盒的影响，并且把Light的Mode设置回Mixed，得到结果：



下面，尝试写自发光材质，并让其贡献给烘焙光照。

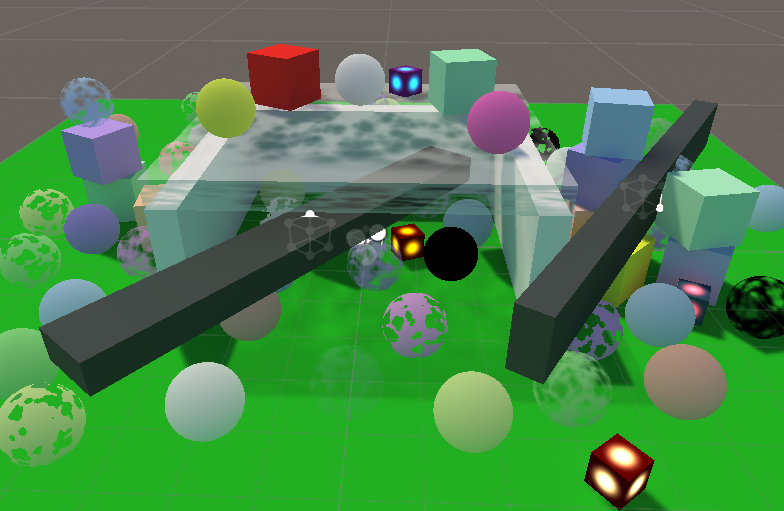
给LightPassFragment添加Emission结果；并给Meta Pass增加Emission贡献。

除此之外，给Shader的Editor部分增加Emission GI的接口，并在发生改变时禁用EmissiveIsBlack的选择。



下面，解决透明物体的烘焙。

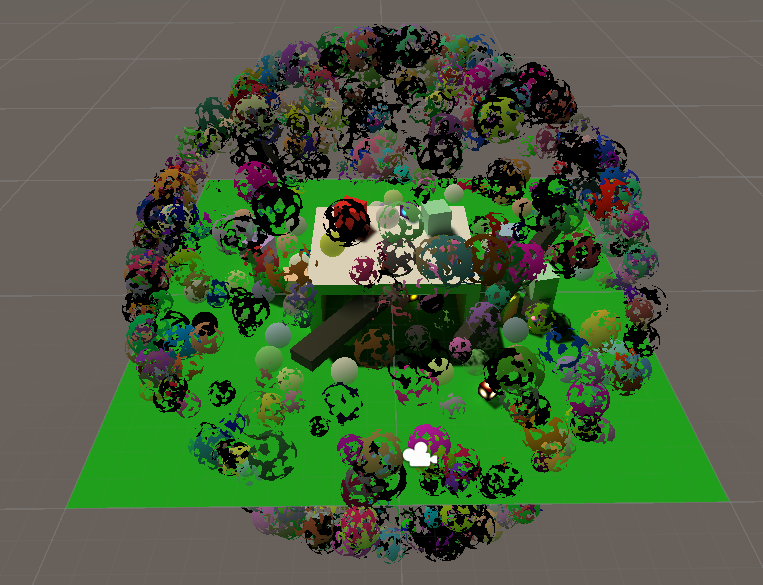
Unity需要MainTex、Color和Cutoff进行判断，前两者需要进行额外处理，还是在OnGUI那里把属性拷贝一下即可。



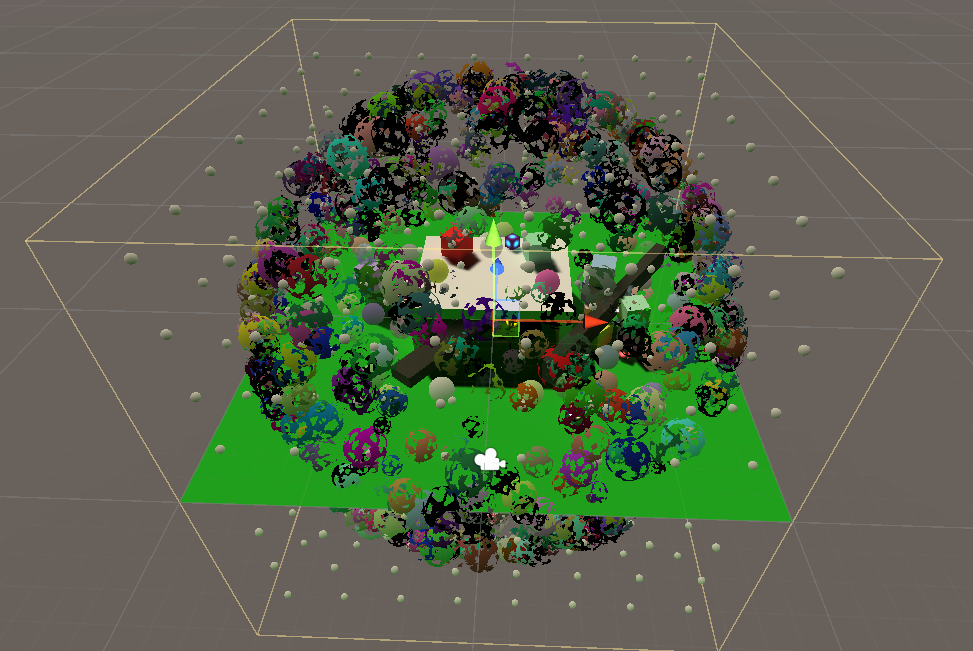
但是这种做法仅限于单个材质、颜色和Cutoff属性，并且该方法会忽略MaterialPerInstance的属性，而是只看该材质UI界面的属性。

最后是对运行时生成的实例添加支持的方法。

更改draw instance的调用，传入LightProbeUsage的设置。再手动生成光照探针，传入block：



另一种方法是使用LPPV。更改以下前面调用draw的设置即可。



最后的成果：

