本部分将实现后处理。

后处理本质上是一系列特效的叠加堆栈。我们先实现这样的堆栈，并实现泛光效果。

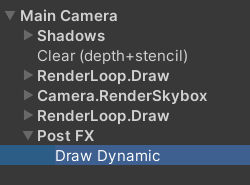
在RP Asset里配置PostFXSettings（继承自Scriptable Object），然后传给RP的构造并记录下来，在调用CameraRenderer.Render时传入。



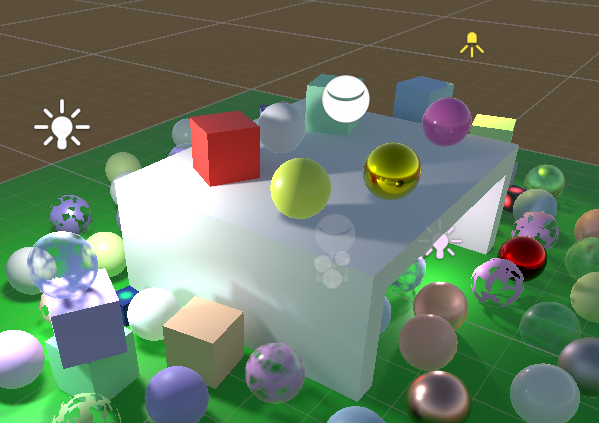
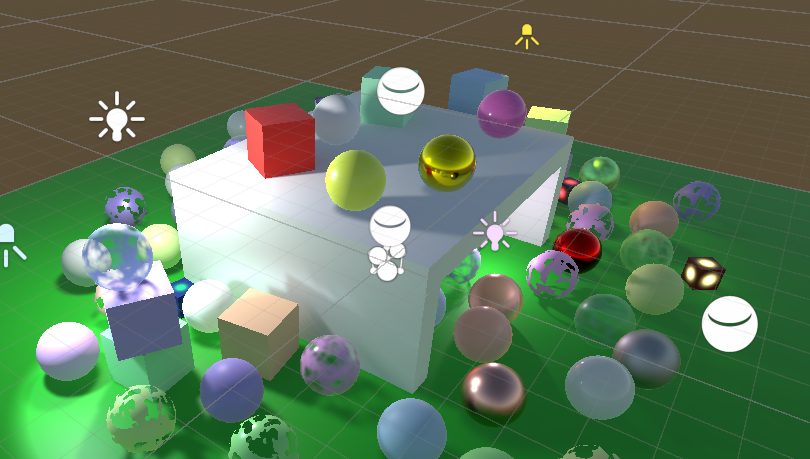
仿照Lighting和Shadows，我们构建PostFXStack，用于记录buffer，并进行一系列设置。

在CameraRenderer里Setup该对象。

如果设置了PostFX，那么在最开始构建贴图并设置为RenderTarget，然后在后处理的Render中将该贴图blit到相机的渲染缓冲区。此时渲染结果无变化，但是可以在FramDebug里看到blit操作：



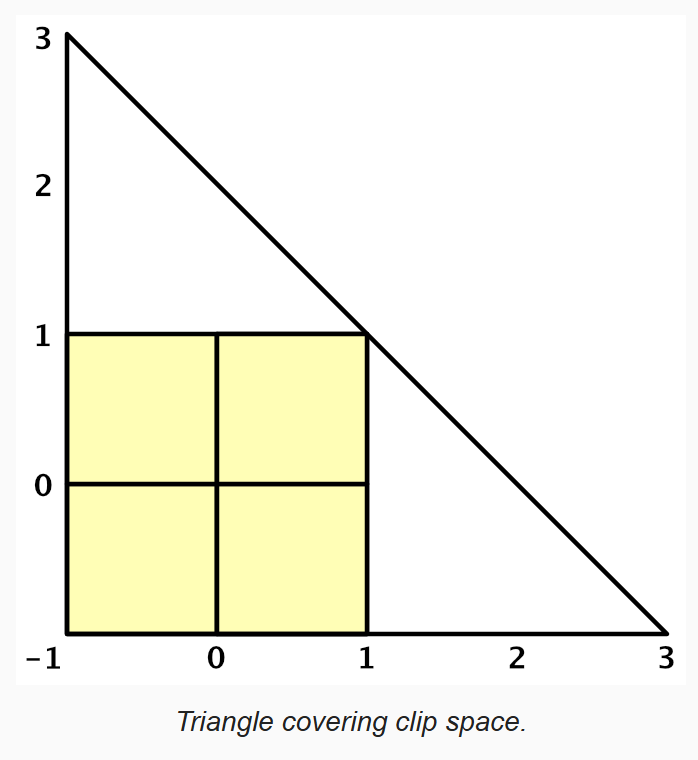
我们需要把gizmo分为FX之前和之后两个部分，在正确的地方调用。

  后者是添加了FX的

注意到，因为3D Gizmo依赖原来的帧缓冲进行遮挡，所以加了FX后，遮挡失效了，我们会在之后处理它们。

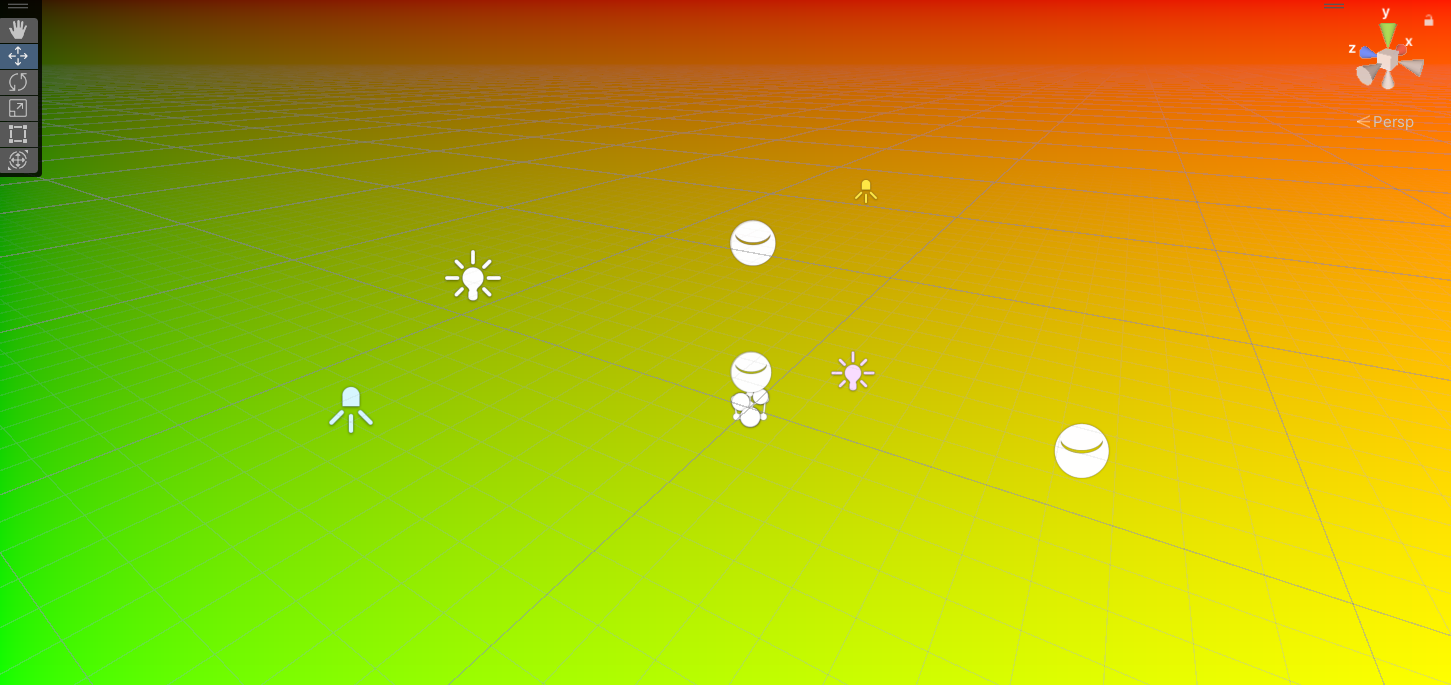
下面，因为Unity默认的Blit是绘制两个三角形组成一个矩形，这会使得在对角线的像素被绘制两次，所以我们可以试着只绘制一个三角形替代该做法。

在自定义的hlsl中，输入vertexID，根据ID指定坐标，该三角形覆盖原矩形：

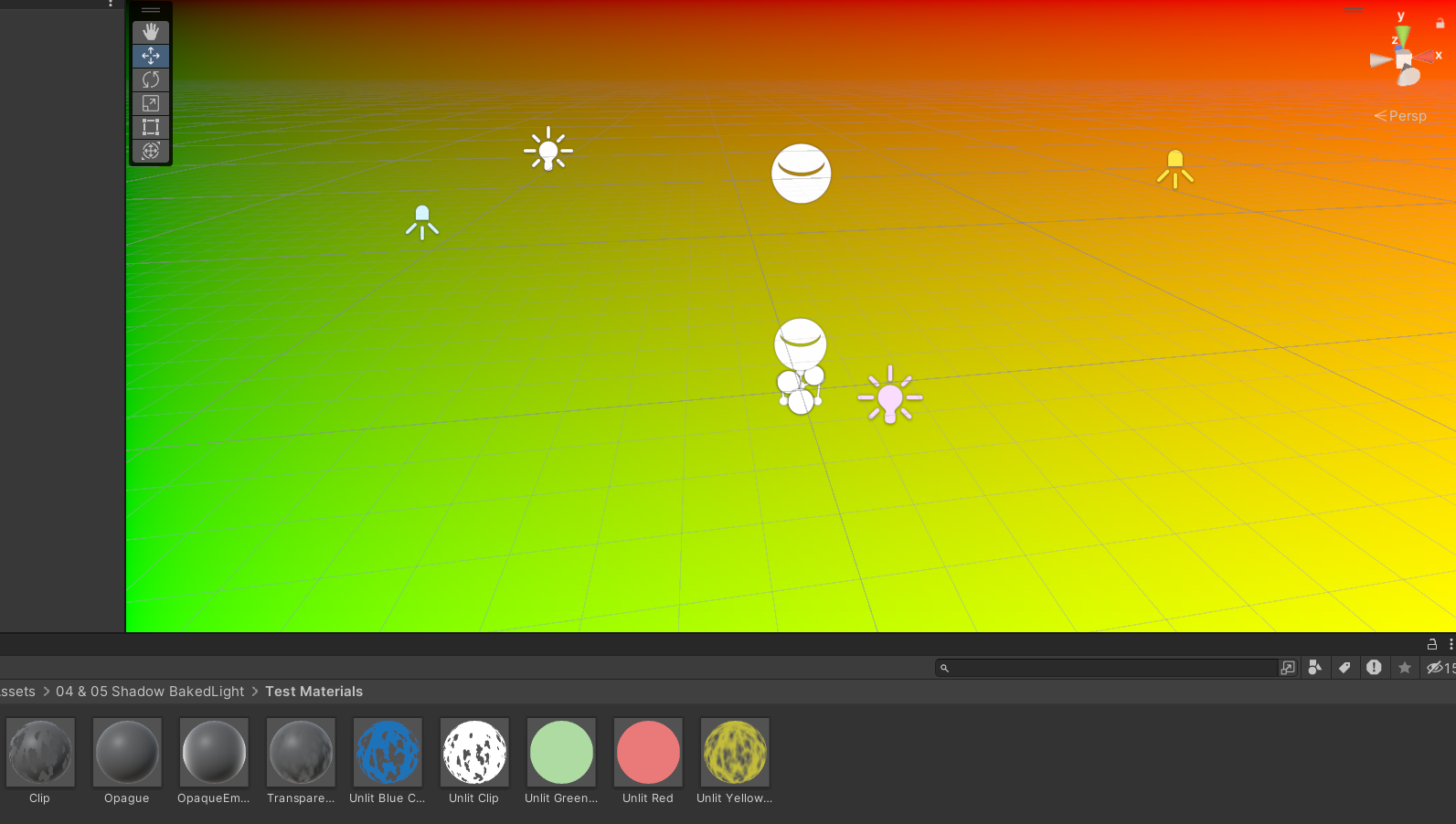


在Settings里配置shader，并且定义Material变量，在外界访问时使用该shader创建材质。

在FXStack中通过DrawProcedural进行绘制，先将UV值作为Debug输出：

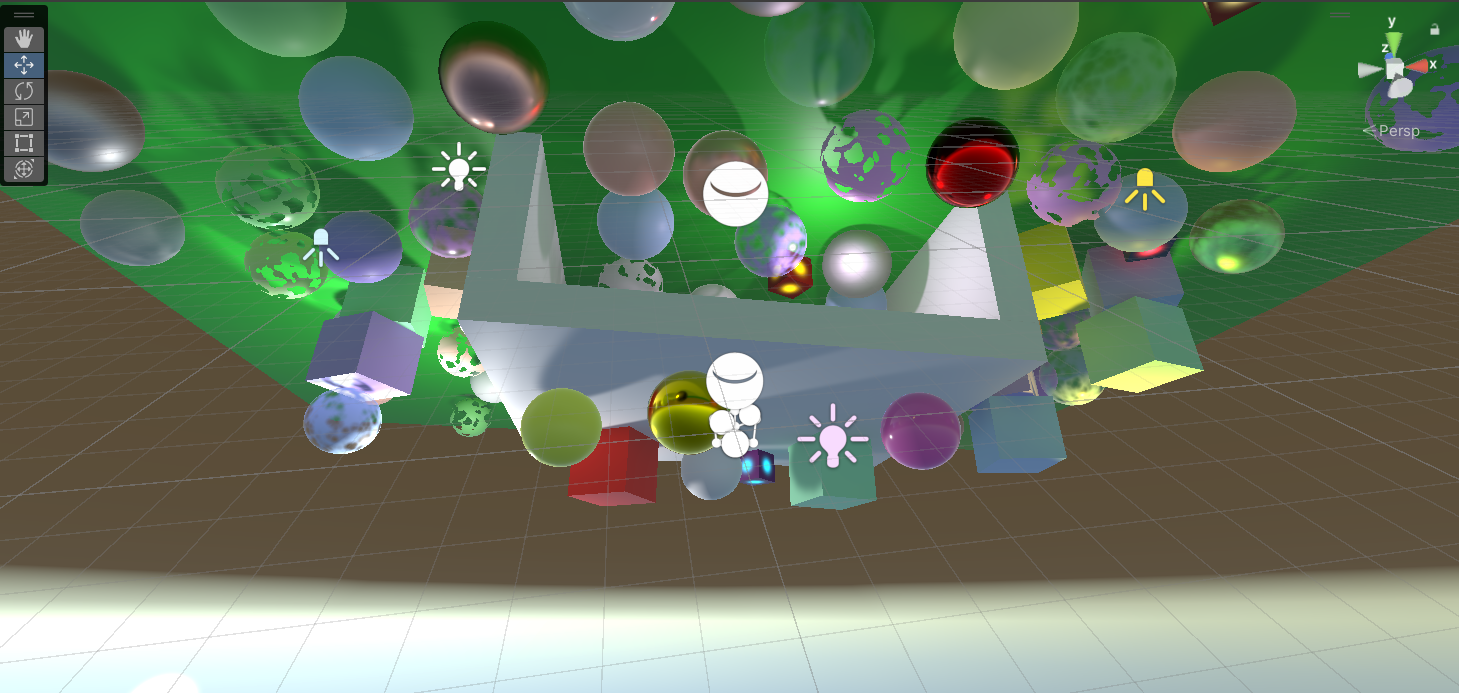


但是，我们并不想对所有的渲染都应用后处理，特别是材质预览、反射探针等场景。在Setup时检查相机类型，只在Game和SceneView下应用。

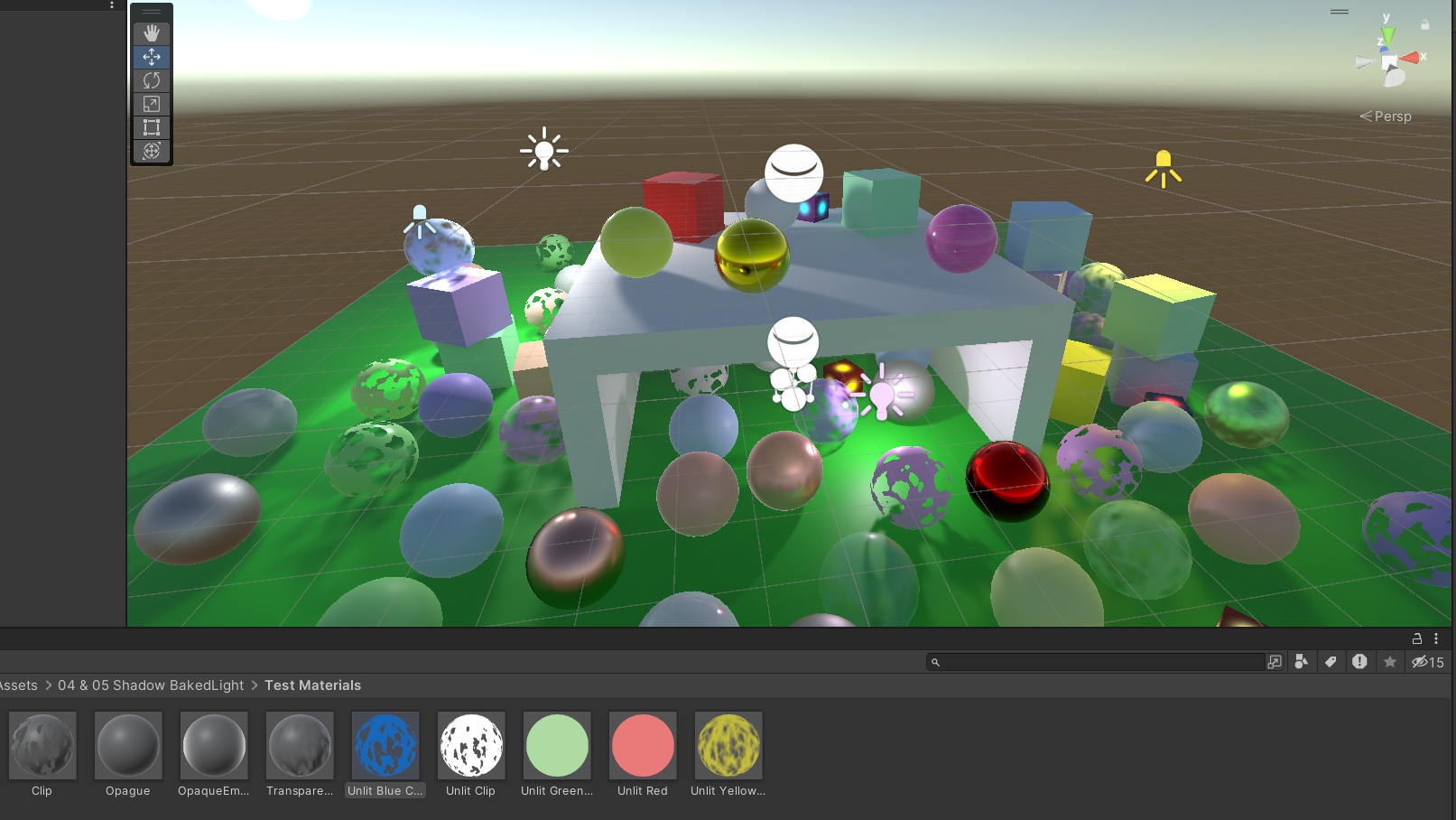


此外，在Editor模式下，因为有时候会希望对比开启和关闭后处理的情况，所以在Setup时需要检查sceneViewState，决定是否关闭后处理。

接下来，如果我们直接采样贴图，可以看到是翻转的：



这是因为一些纹理的v分量从顶部开始，我们通过\_ProjectionParams判断该点，以修复问题：



接下来实现LDR的bloom效果。

泛光效果原理是下采样到分辨率更低的贴图，以实现模糊效果：

