Custom Render Pipeline（RP）

Unity 2018：the Lightweight RP and the High Definition RP.

Unity 2019：the Universal RP and the High Definition RP.

本部分将实现一个绘制Unlit材质的自定义管线，并将其他材质绘制成“错误”材质

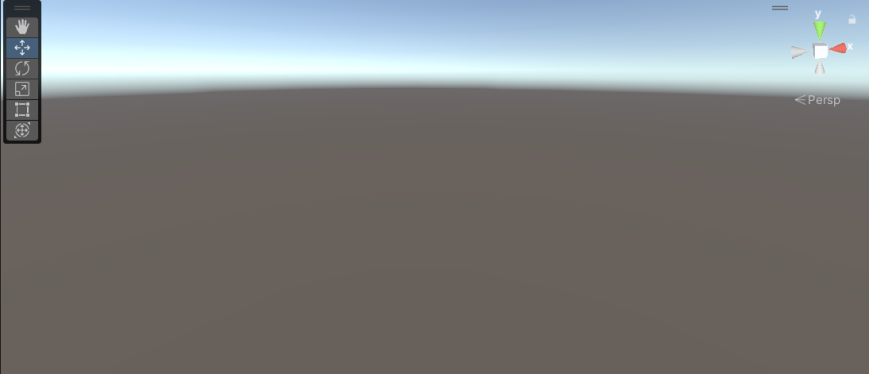
创建自定义的RP，首先需要一个CustomRenderPipelineAsset类和CustomRenderPipeline类。

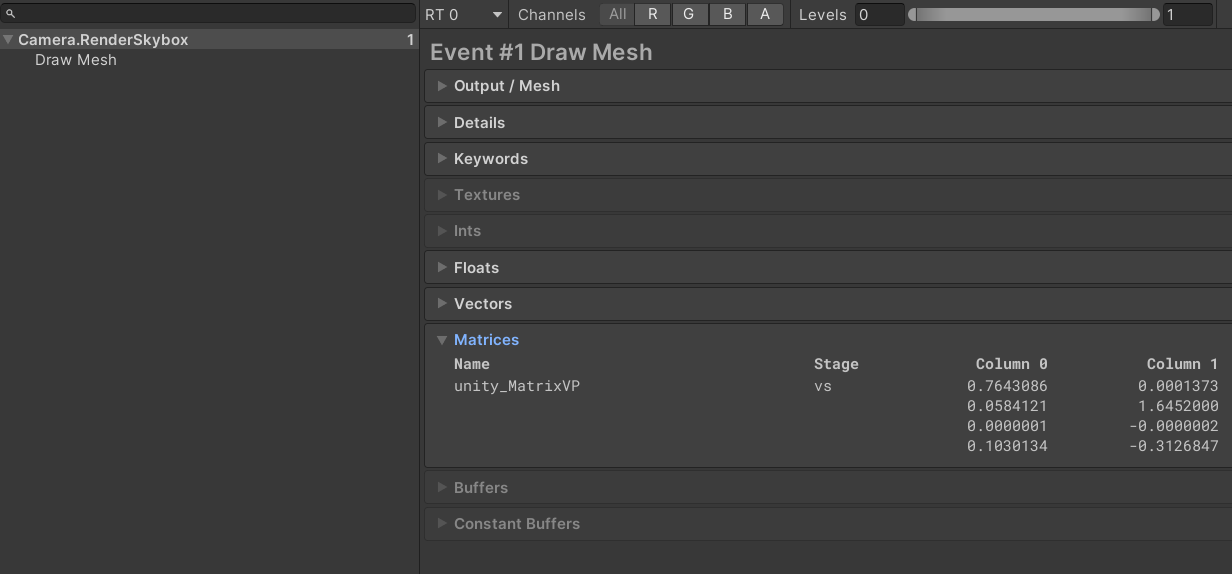
前者重写CreatePipeline方法，返回后者的一个实例，后者通过Render方法作为自定义SRP的入口。

注册自定义的CameraRenderer类，包含ScriptableRenderContext和Camera两个成员，在CustomRenderPipeline类的Render方法中，遍历所有相机并调用该Renderer类的绘制接口。

首先尝试绘制天空盒，分成三步：1.向context注册相机属性；2.调用DrawSkybox；3.提交context中的命令。

得到：





注：可以通过 Window / Analysis / Frame Debugger 打开帧调试器查看绘制步骤。

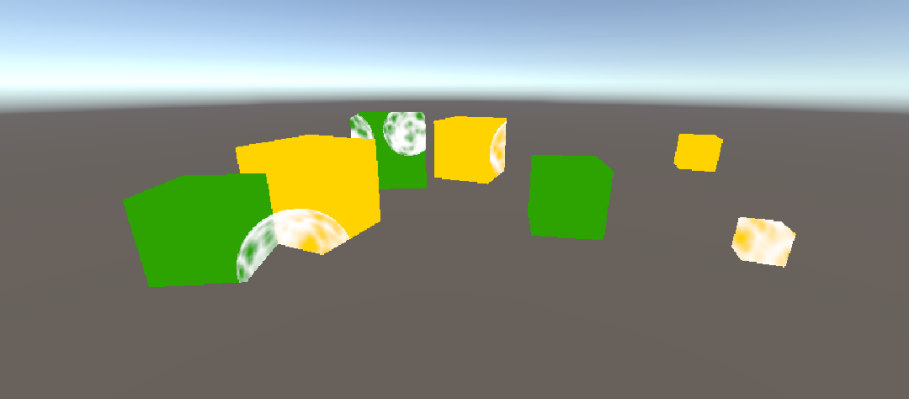
为了绘制其他物体，需要创建命令缓冲区，作为renderer类的一个成员变量。

在设置上下文时，清空缓冲区并开始采样；在提交上下文前，停止采样，并向上下文内写入命令缓冲区的命令。

为了获取需要绘制的物体，在每帧的开始，将相机的视锥体信息提交给context，获取视锥体剔除后的物体信息。

然后，在DrawVisibleGeometry中，通过相机设置物体排序方式（sortingSettings），通过shaderTagId和排序方式设置绘制方式（drawingSettings），通过指定RenderQueueRange设置过滤方式（filterSettings），再把它们和剔除结果一起传给context的DrawRenderer方法：

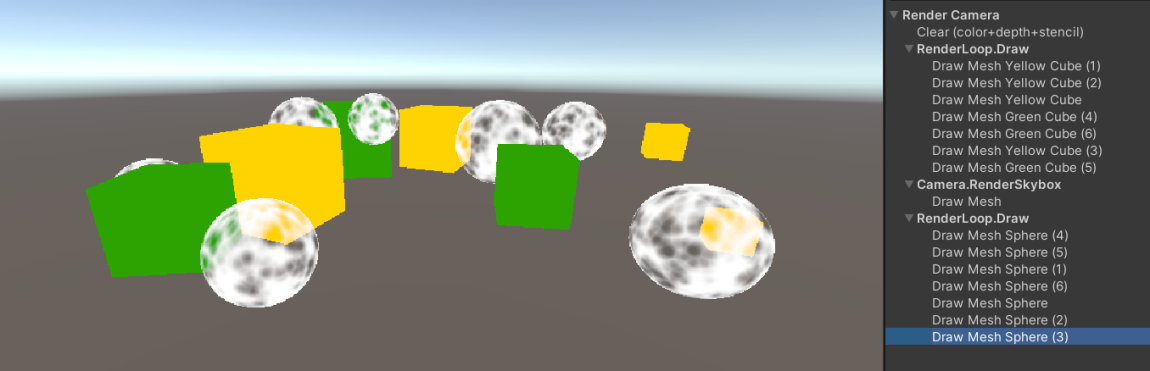
shaderTagId = SRPDefaultUnlit



因为先绘制透明物体，再绘制天空盒，且前者未写入深度缓冲区，所以显示结果如上所示。

因此，在第一次调用DrawRenderer时，限制过滤范围为不透明物体；在绘制天空盒后，再调用DrawRenderer，限制过滤范围为透明物体。

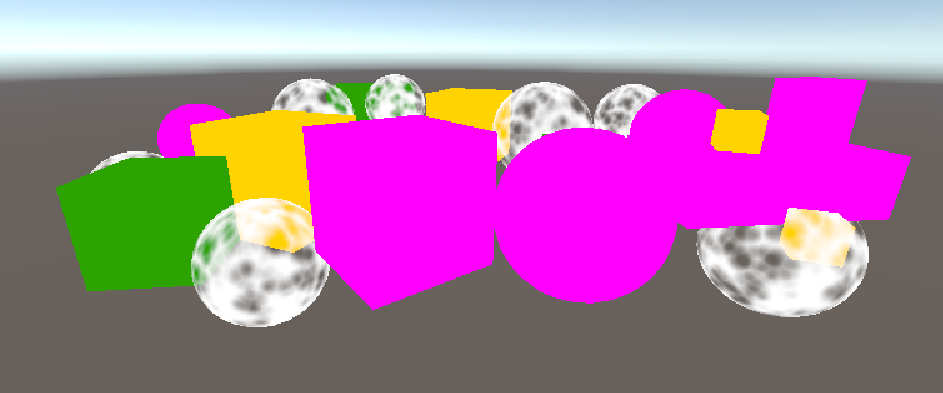
得到正确的结果：



（透明物体按实例从后向前绘制，但这并不是完全正确的）

下一步，准备拓展绘制的对象范围。

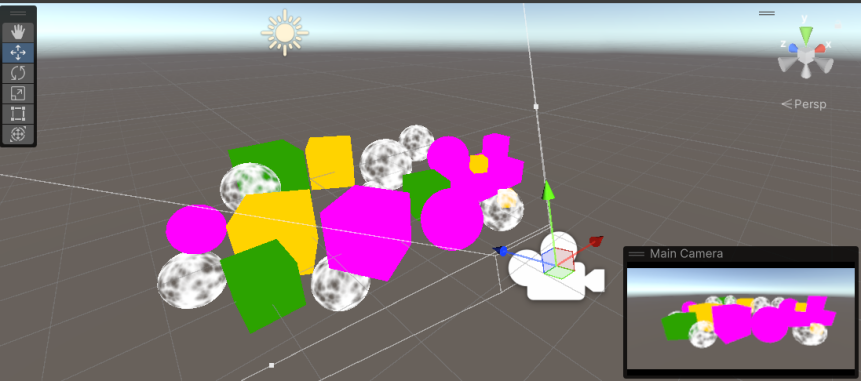
记录所有合法的ShaderTagId，依次注册给drawingSettings里的shaderPassName，再用“错误”材质替代原有材质，并使用默认的filterSetting绘制：



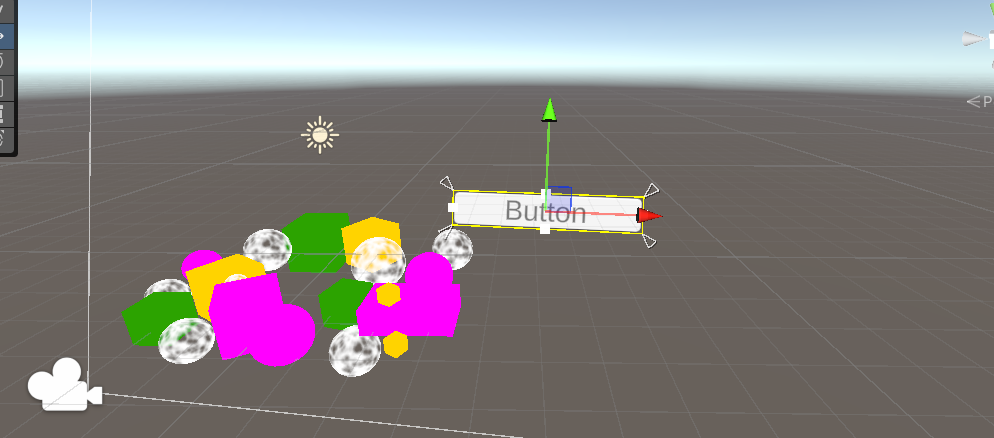
进一步的，通过partial关键字分离代码，并通过#if UNITY\_EDITOR控制“错误”材质的绘制。

下一步，准备绘制Gizmo。该步骤在所有绘制的最后执行。

通过Handles.ShouldRenderGizmos判断是否绘制Gizmo，然后直接调用context的对应接口即可，要对PreImage的部分和PostImage的部分都进行调用。



为了在编辑界面绘制游戏内UI，先把主相机传给Canvas，再在视锥剔除之前通过ScriptableRenderContext.EmitWorldGeometryForSceneView注册对应的几何体。



接下来，处理多个相机的情况，场景中存在多个相机时，会按照它们的"Depth"属性递增进行绘制。ClearFlags从1到4依次是 Skybox、Color、Depth 和 Nothing。因此在进行最开始的Clear时，可以根据相机的Flag调整设置。

调整第二个相机的ClearFlags和Viewport Rect属性，可以得到有趣的结果：



注：还可以通过Layer修改每个相机绘制的内容。