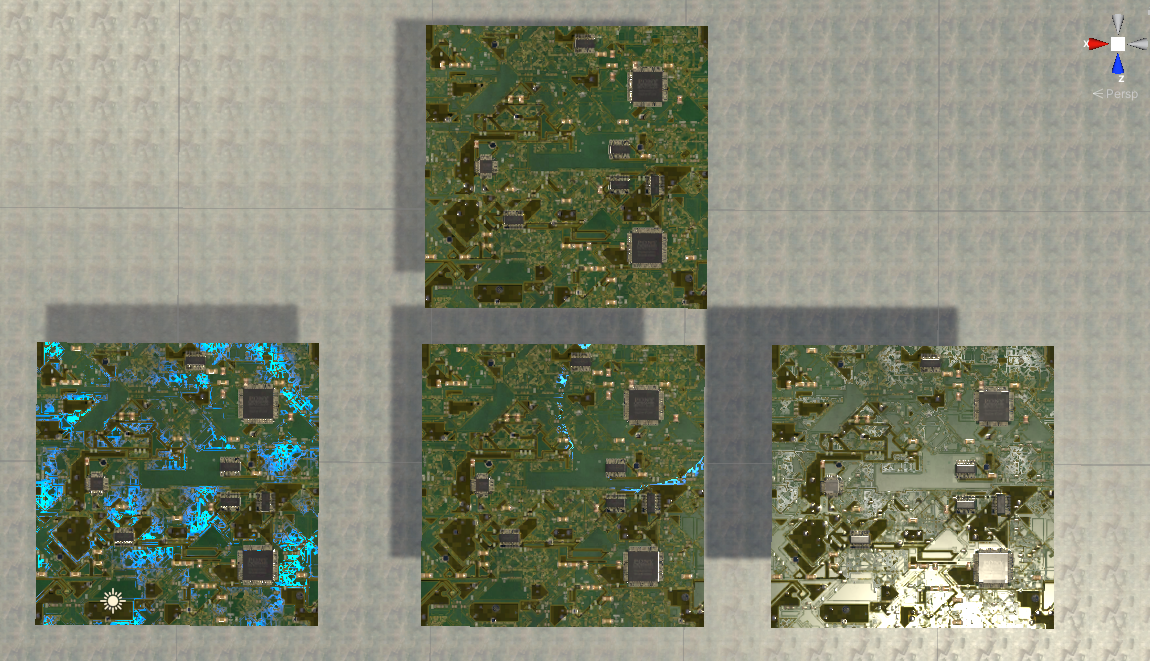
# 游戏引擎原理与实践

深互动媒体硕20 鄢磊 2020214426

如图所示，PBR的材质实践效果如图1所示。可执行文件在Build目录中。

自定义材质实现的电路流动效果分为三类，第一类是类voronoi图的随机流动方式（下左），第二类是电流流向右上角芯片中（下中），第三类是电流从右向左流动。

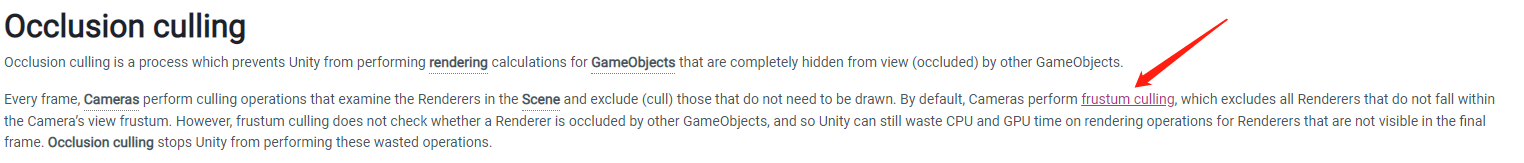


图一 材质系统实践效果

**遮挡剔除与视锥剔除：**

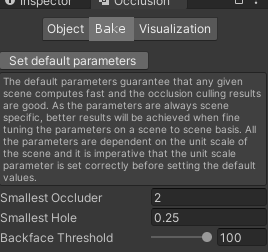
1. **视锥剔除**

所使用的unity版本为2020.1.5f1c1，查阅unity manual可知，unity在渲染中默认开启了视锥剔除.如下图所示。

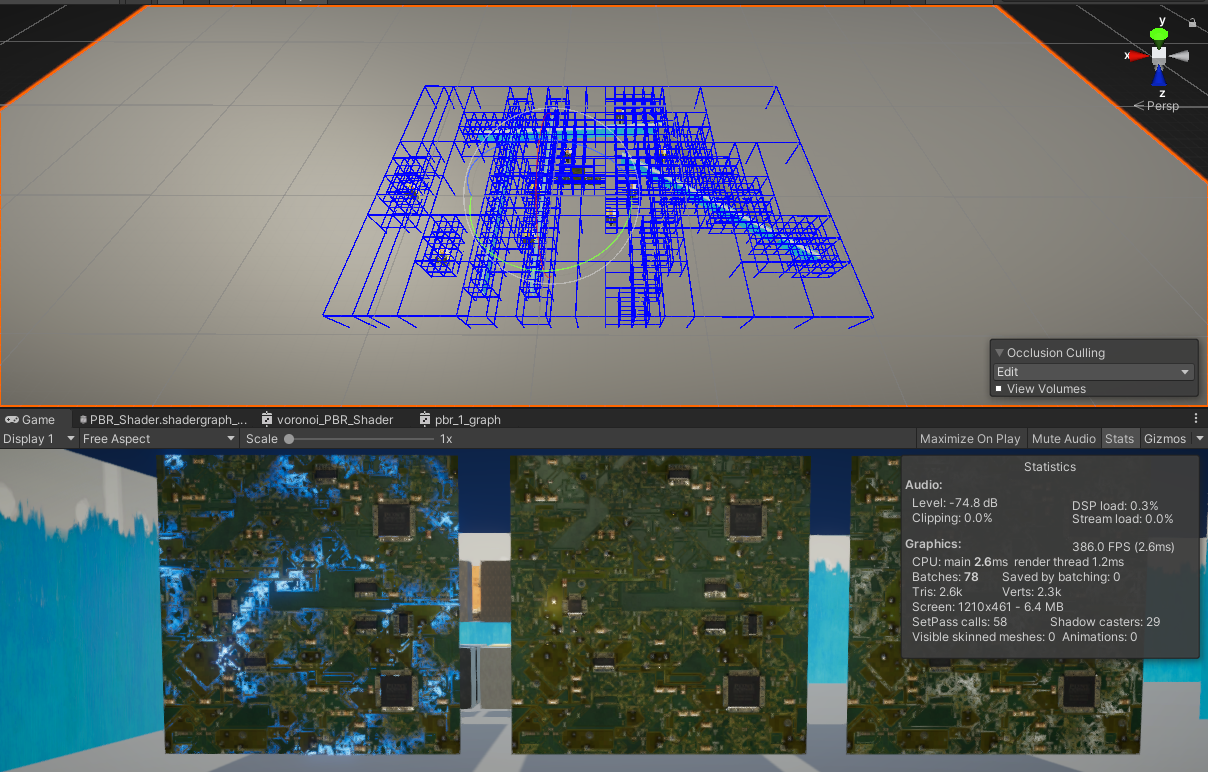


1. **遮挡剔除**

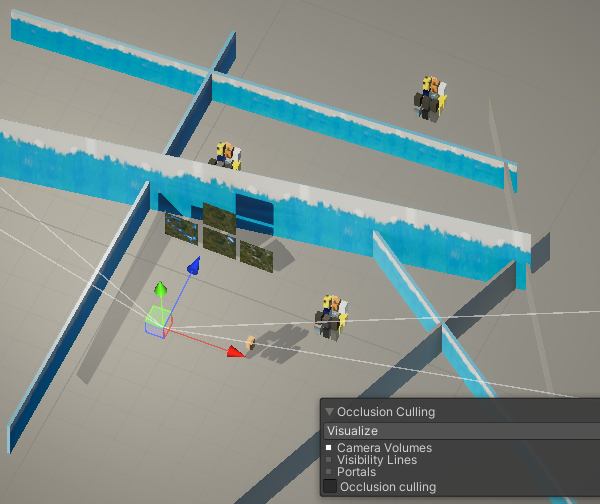
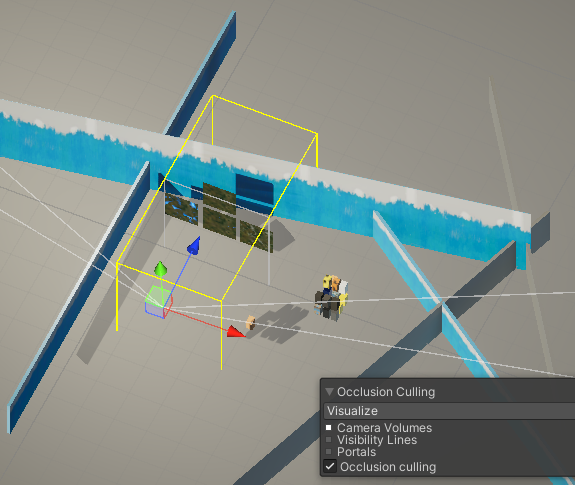
为了开启遮挡剔除，将场景中不动物体（即所有物体）标记为occlusion static后，采用了如下参数bake:



可以得到场景中当前的occluder与stats如图：



开启遮挡剔除前后的对比图：

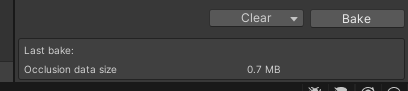
 

可以看出，在开启遮挡剔除后，后方的物体不再被渲染。

此处已经去掉了默认的天空盒，以减少verts数量。

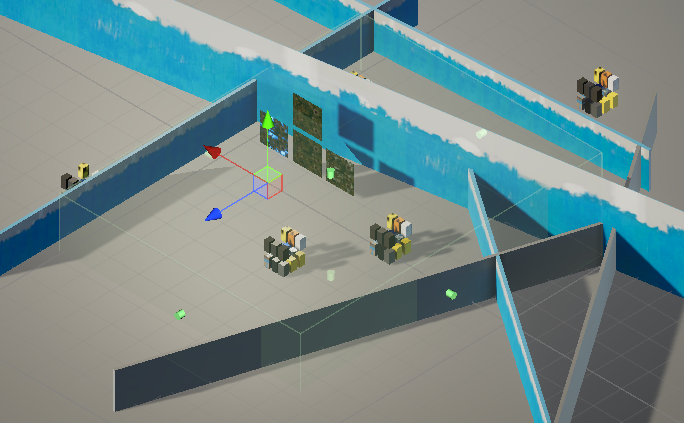
1. **Occlusion Area**

此时尚未采用Occlusion Area,因此Bake出的数据较大，也更花时间，如图所示。

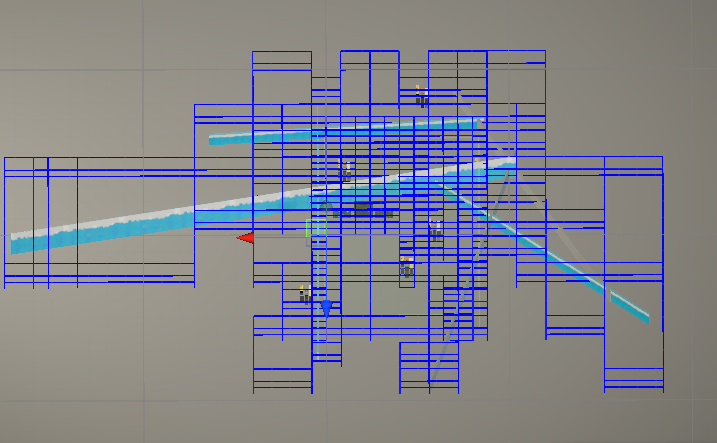


Bake后数据量 0.7MB

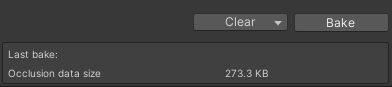
此后，在场景中可移动区域设置了Occlusion Area,进行Bake,如图所示：



场景中的Occlusion Area

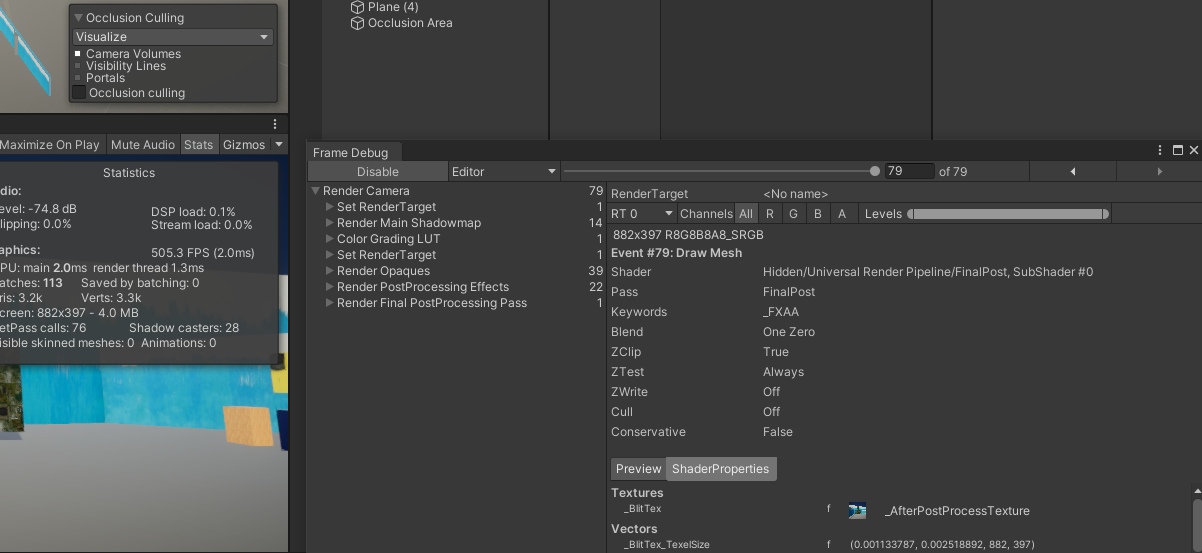


Bake后的Occluder 盒



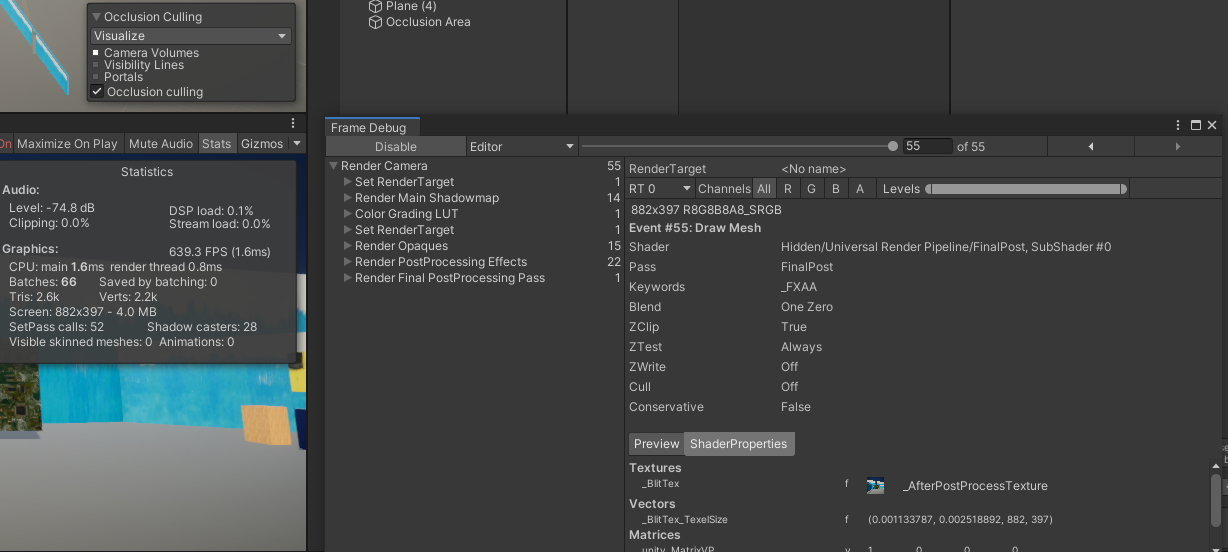
设置Occlusion Area后的data size:273.3KB,缩小3倍

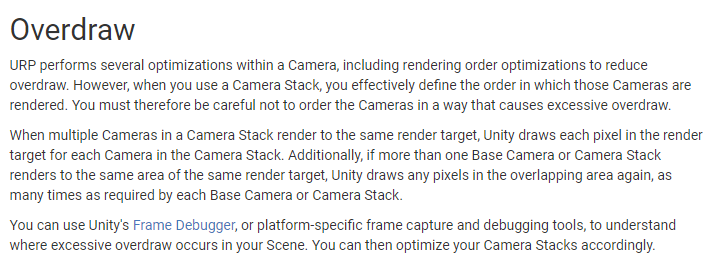
首先查看场景中只有一个立方体，一个平面的信息，由于这两个物体的scale均经过调整，在unity中batch数量与物体数量并不一致。立方体之后挡住了15个立方体，即其为不可见。现在还将刚才启用的Occuler进行Clear，开启frameDebugger与stats窗口，可以查看到当前的渲染信息：



可以看到当前的batchs为113，setPass calls为73，而渲染事件包含79个（包含后处理等），由于项目整体采用的是URP的渲染管线，并不能从Scene视图中查看到overdraw的叠加图，只能从FrameDebugger中查看信息，可以看到其中Render Opaques部分数量明显远大于场景中可见物体数量。

在保持场景物体位置不变的情况下，对遮挡进遮挡剔除，可以看到当前的batchs降到了66，setPass calls为52，而渲染事件包含55个，其中render opaques的数量明显下降，可见其overdraw进行了较大的优化。并且可以看到场景中Verts与Tris的数量都有所降低，可见确实少处理了一些物体。





**总结：**

由于采用了URP渲染管线（URP在管线中对单个相机会进行渲染顺序的优化从而降低overdraw，见上图），自定义了多Pass的渲染方式，且改变了物体的材质，导致不易定量计算当前视图中SetPassCall,Batch数量，overdraw与物体数量的相对关系。但可以明显看到，随着开启遮挡剔除，各项数值均有明显下降，可证明遮挡剔除对于渲染优化的重要意义。