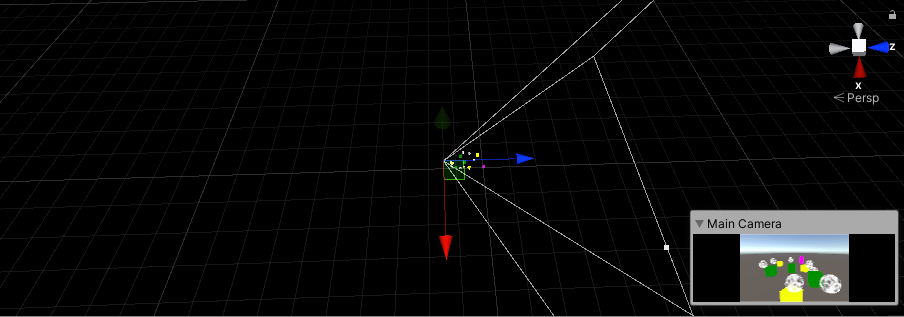
3.4:画Gimos

3.5 画Unity UI

游戏窗口有但是Editor窗口没有UI 显示

4:多摄像机

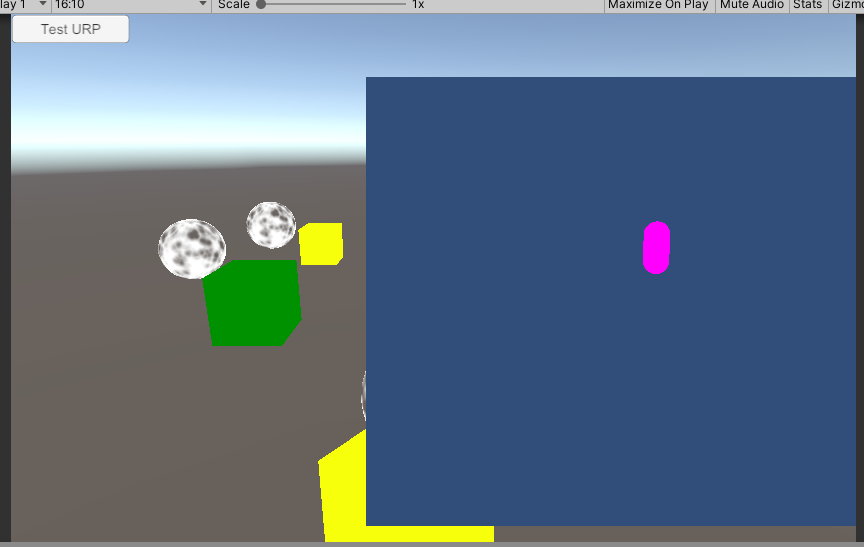
4.1 两个摄像机

两个摄像机同一个范围会合并到一起,要有不同的范围进行渲染。

4.2 处理改变的BufferName

4.3 图层

4.4 清除标志 Cull的层和摄像机的清除标志

课程2 Draw Call

目标：写一个HLSL 着色器

支持SRP Batch ,GPU Instancing 和动态合批

配置 材质属性对于每一个物体和随意的画很多

创建半透明材质和 镂空材质

1：着色器

* 1. 不发光的Shader
  2. HLSL程序
  3. Include 保护
  4. 着色器功能

我们是应该使用float还是half

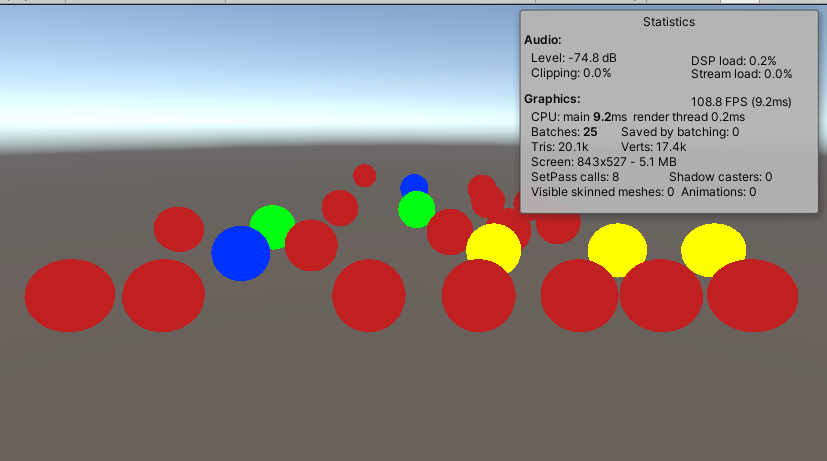
* 1. 空间转换
  2. 核心库 Common.HLSL SpaceTransform.HLSL
  3. 颜色

每一个材质球配置颜色\_BaseColor

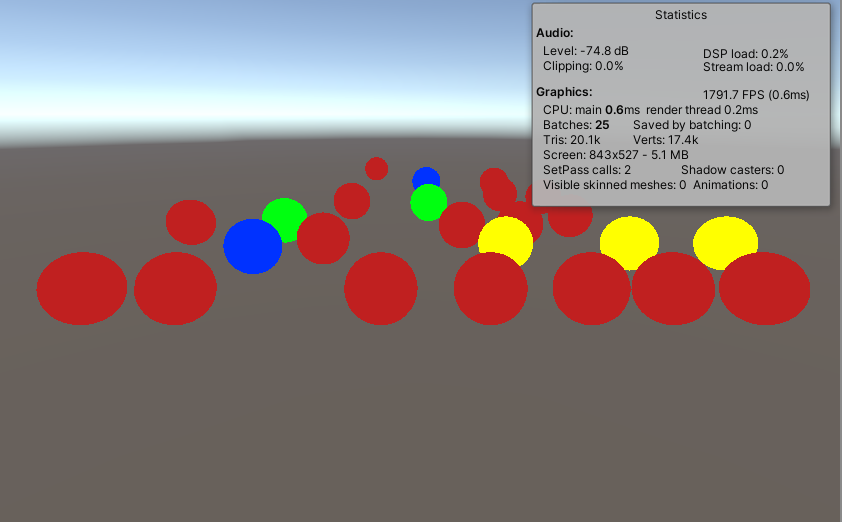
1. 合批
   1. SRP Batch

SRP Batch并不会减少Draw Call的数量而是Shader更精简,缓存材质属性在GPU 并不必发送给每一个Draw CallSRP批不是减少draw调用的数量，而是使它们更精简。它在GPU上缓存材质属性，所以它们不必在每次draw调用时都被发送。这既减少了必须通信的数据量，也减少了CPU在每次draw调用中必须做的工作。但这只有在着色器遵循统一数据的严格结构时才有效。

开启之前



开启之后



1. Save by batching Set Pass Call减少到2的数量
   1. 许多颜色

Unity简单得会把相同变体得Shader进行合批。OnValidate()当组件被加载和改变得时候进行触发,SRP Batch并不能处理逐物体得材质属性

* 1. GPU Instancing

#include "Packages/com.unity.renderpipelines.core/ShaderLibrary/UnityInstancing.hlsl"

Instancing Buffer

GPU Instancing 必须使用相同的材质 SRP Batch是使用相同的Shader变体 材质属性单独改变,批处理大小是会有限制的。

* 1. 画许多实例化网格

使用Instancing API的时候 必须材质开启Enable GPU Instancing 多少Draw Call是取决与平台的,每个Draw对应的最大Buffer是不同的。

* 1. 动态合批

会有顶点限制和物体尺寸限制 静态合批需要标记和会增大内存

* 1. 配置合批

1. 半透明

3.1混合方式

Src Alpha 是要画的图像

* 1. 不写入深度
  2. Alpha纹理
  3. Alpha 裁剪

0或者更小的值 片段会被裁剪掉

Alpha Test 2450 在不透明渲染之后进行渲染,丢弃片段让GPU 优化变得不可能,

3.5 Shader 特征

**shader\_feature 根据配置勾选材质生成变体 Multi都生成**

3.6Cutoff 每一个物体

3.7生成更多的Alpha Test物体

第三课时 方向光

用法向量计算光照

支持4盏方向光

应用BRDF

制作点光源的半透明材质

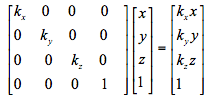
‘ 创建自定义GUI 预先设置

1 灯光

模拟灯光和物体表面相交

* 1. 灯光Shader
  2. 法线向量

世界空间法线向量,取决与灯光和表面的角度 表面法线



如果三个缩放系数k都相等，我们称为统一缩放，否则是非统一缩放，因为非统一缩放会拉伸或挤压模型，所以会改变与模型相关的角度和比例，这在法线变换时很重要，如果非统一变换，直接使用变换顶点的变换矩阵就会出错

1.3内插法线

1.4 表面属性

1.5计算灯光

2 灯光

2.1灯光的结构

光的放行指明的是光是从哪个方向来

2.2 灯光功能

2.3 发送灯光数据到GPU

2.4 可见灯光

2.5 多个方向光

2.6 Shader 循环

2.7 Shader 目标

3 BRDF

3.1 进来的灯光