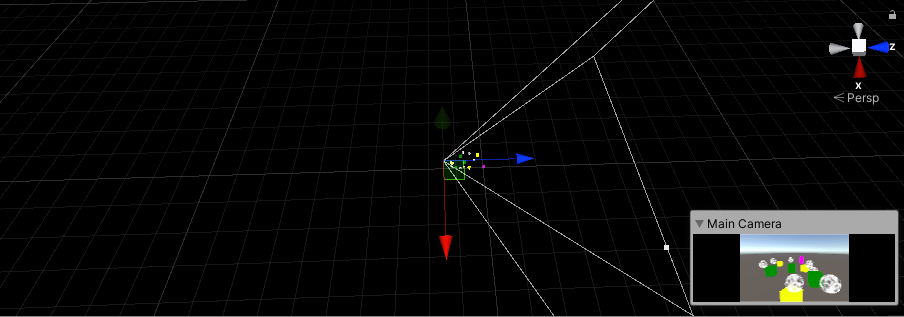
3.4:画Gimos

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

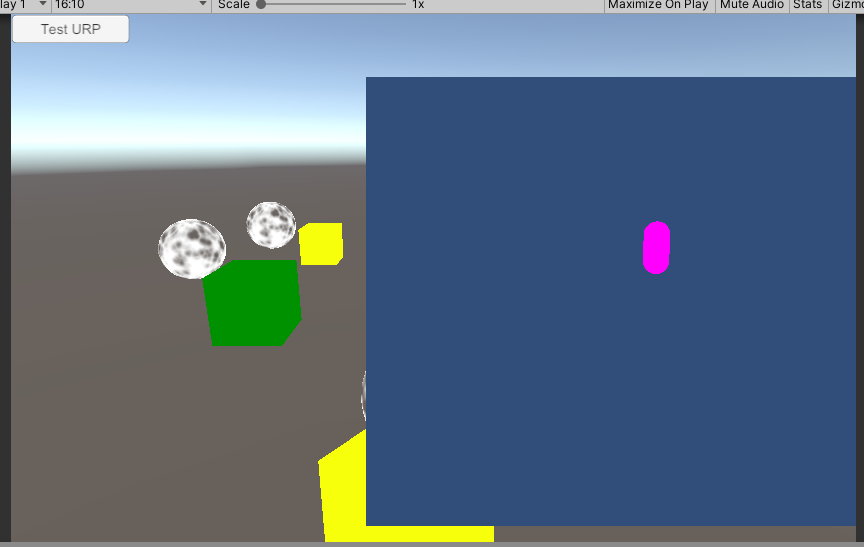
4.1 两个摄像机

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

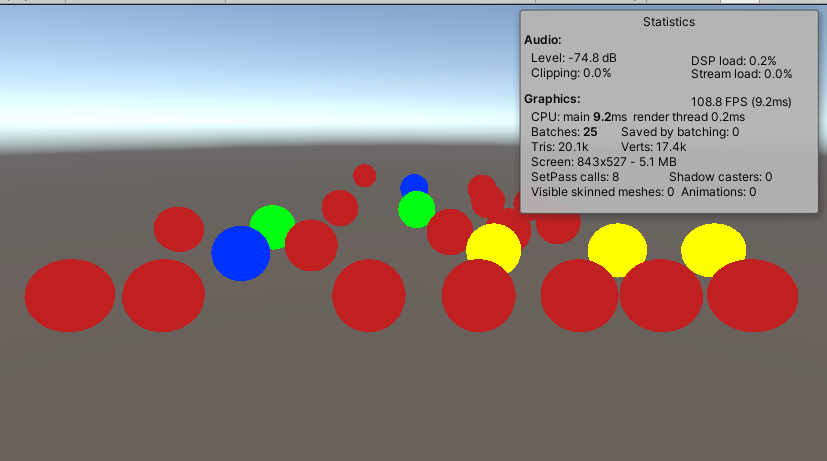
* 1. 空间转换
  2. 核心库 Common.HLSL SpaceTransform.HLSL
  3. 颜色

每一个材质球配置颜色\_BaseColor

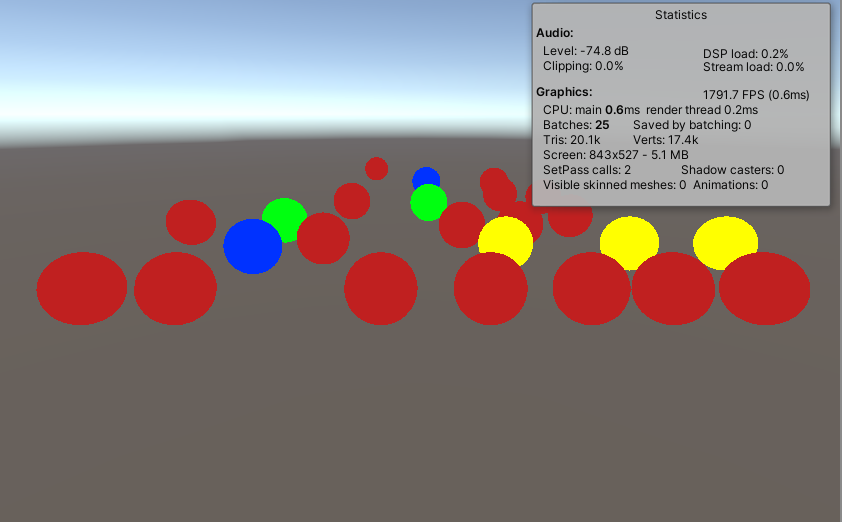
1. 合批
   1. SRP Batch

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

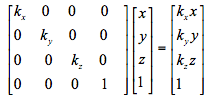
‘ 创建自定义GUI 预先设置

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

* 1. 进来的灯光
  2. 输出的光线 光线反射和光线的散射

分散的高光反射、

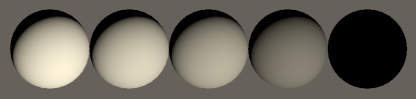
3.3表面属性

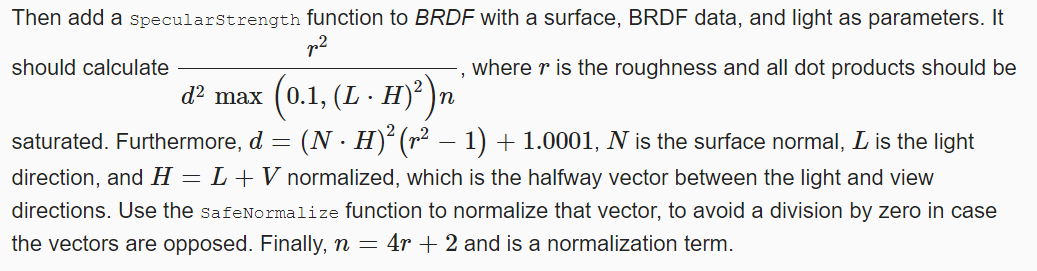
金属度和光滑度

* 1. BRDF属性

3.5反射率

金属度越强 漫反射越弱 能量守恒



* 1. 高光
  2. 粗糙度
  3. 视野方向
  4. 高光强度 计算高光系数
  5. 

增加金属度 漫反射颜色逐渐变黑 高光反射金属的颜色

Mesh Instance

4半透明

4.1预乘Alpha

预乘alpha和漫射有效地将物体变成玻璃

可以把漫反射影响也变成透明 的颜色完全模拟玻璃效果

5 Shader GUI

5.2设置属性和关键字

5.3 预设置按钮

第四章 方向光阴影

渲染和采样阴影RT

支持多个方向光阴影

用 Cascaded 阴影图 –>会增加Draw Call 软阴影技术

混合 淡入淡出 和 过滤阴影

1渲染阴影

投射阴影物体和接收阴影物体

1.1阴影设置

渲染阴影的距离和阴影图的大小,阴影生成的最大距离

1.2 通过设置

1.3 Shadow 类

虽然从逻辑上来说阴影是照明的一部分，但它们相当复杂。

1.4 灯光和阴影

GetShadowCasterBounds 当前灯光没有照射任何物体，但是依然进行shadowmap的绘制 看是否绘制ShadowMap

1.5 创建阴影图集

创建阴影贴图是通过在纹理上绘制阴影投射对象来完成的，格式为shadowMap格式

我们只能在首次声明一个纹理时才释放它，目前我们只在有方向阴影要渲染时才这么做。最明显的解决方案是只有当我们有阴影时才释放纹理。但是，不声明纹理会给WebGL 2.0带来问题，因为它将纹理和采样器绑定在一起。当一个材质和我们的着色器加载而纹理缺失时，它将失败，因为它将得到一个默认的纹理，这将与阴影采样器不兼容。我们可以通过引入一个着色器关键字来生成省略阴影采样的着色器变体来避免这一点,少引入shader变体

1.6 影子第一步

1.7 渲染

渲染多少盏平行光阴影是分Tile的, 阴影贴图的想法是我们从光的角度渲染场景，只存储深度信息，这个结果告诉我们光在击中物体之前传播了多远。

每增加一盏方向光,物体画阴影多一倍的DrawCall 在灯光位置画多次物体 全部叠加到一张ShadowMap 分Tile

1.8 阴影投射 Pass

1.9 多个灯光

2 采样阴影

2.1阴影矩阵

构建阴影矩阵,具体从哪一个Tile里面采样阴影图

近裁剪平面和远裁剪平面 决定摄像机看到的深度范围

2.2 储存阴影数据每一盏光

2.3 Shadow HLSL

双线性对常规的ShadowMap是没有意义的。

2.4 采样ShadowMap ->转换到阴影空间的矩阵直接是Unity进行提供的。

计算阴影的衰减, 然后添加一个GetDirectionalShadowAttenuation函数，返回shadow attenuation，给定的方向阴影数据和一个surface，这个surface应该是在世界空间定义的。它使用平铺偏移量来检索正确的矩阵，将表面位置转换为阴影平铺空间，然后对图集进行采样。块的片段是并行进行着色的

2.5 灯光衰减 动态分支和静态分支

问题出现摩尔纹和黑方块, 摩尔纹是自阴影导致的,就是阴影ShadowMap的分辨率有限patterns 提高分辨率并不能消除 ShadowMap对齐的方向是灯光的方向而不是摄像机的方向,一个光使用另外一个光的阴影。现在这些无效的阴影保持可见.黑方块跟阴影距离有关系。

3 Cascaded 级联的阴影图

比率增大ShadowMap分辨率 更近的地方用更高的分辨率,需要一个可变的ShadowMap分辨率,基于阴影接收器的视野距离。级联阴影映射是这个问题的解决方案。这个想法是阴影施法器被多次渲染，所以每个光在图集中得到多个瓷砖，称为瀑布。第一个级联只覆盖靠近相机的一个小区域，连续的级联缩小，以相同数量的texel覆盖一个越来越大的区域。着色器然后为每个片段采样最好的级联。

3.1设置

3.2 渲染Cascades

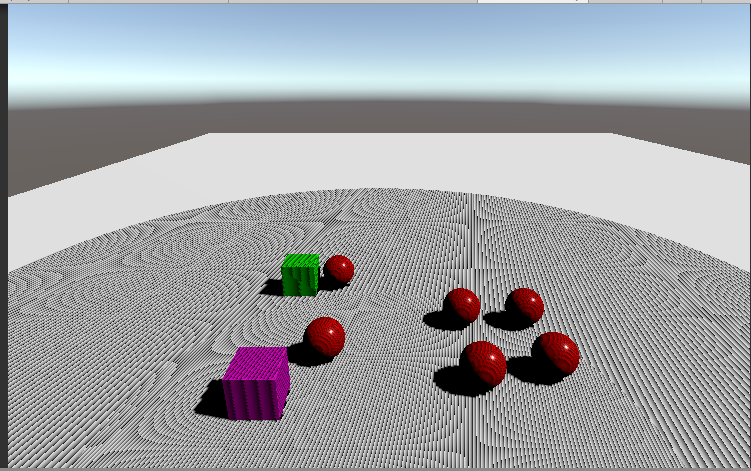
每多一个Cascade 就会多一倍的DrawCall 会在画一遍,多一盏光也会多一倍的Draw Call。开四倍 地板Plane画了四次。

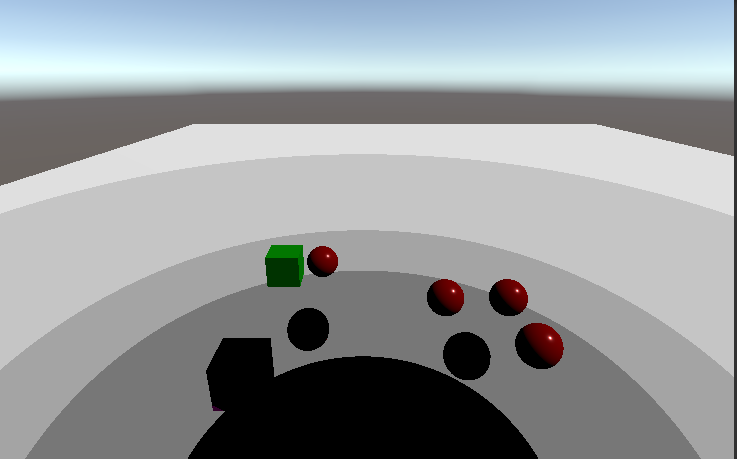
3.3 剔除球 优化

检查片段是否在这个球里面,球是确定片段是否在哪一级的Cascade里面

3.4 采样Cascade

世界空间表面位置 剔除球决定片段在哪一级的Cascade里面。根据片段世界位置到球体中心的距离和球体半径进行比较。可视化阴影Caca





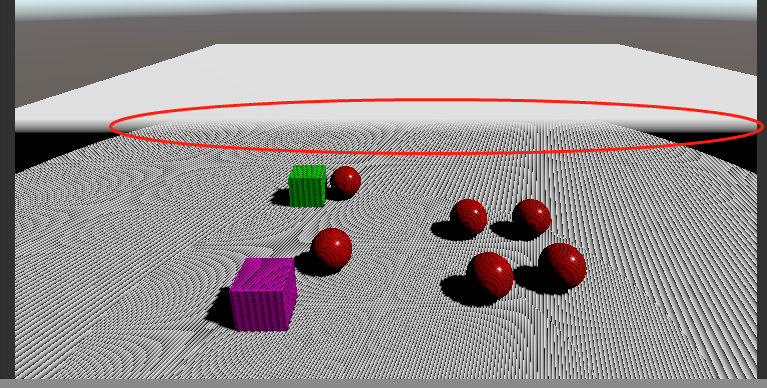
* 1. 剔除阴影采样

3.6最大距离

最大距离是基于视野空间深度,而不是相机的位置距离,

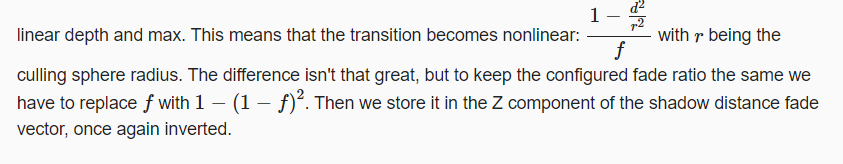
3.7阴影衰减

增加一个衰减项产生一个过渡,边界处在阴影消失的边界的地方~~



3.8 衰减 Cascades 级联贴图

两个Cascades之间做一个过渡。利用一个非线性的衰减



4:阴影质量

问题:acne这是由表面的不正确的自阴影造成的，没有与光的方向完全对齐 简单的增加分辨率是不能解决acne

4.1：深度Bias

增加恒定的偏移,问题Peter平移。

4.2 Cascade 数据

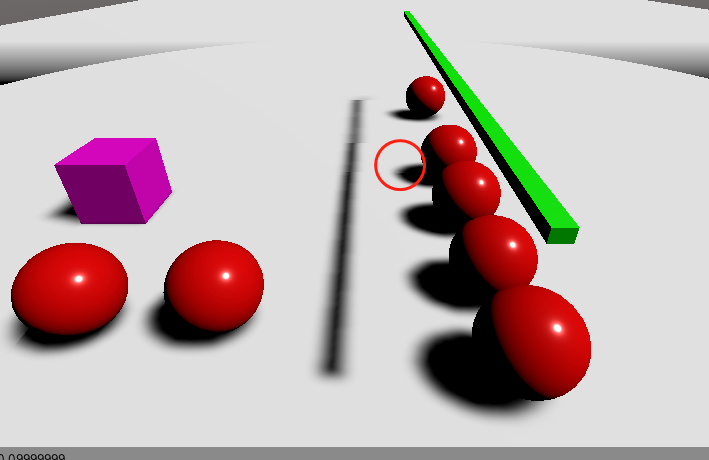
acne 取决于世界空间纹素的大小

4.3 Normal 偏移

不正确的自阴影发生是因为投射阴影者的深度texel覆盖了多个碎片,

4.4配置Bias

4.5 阴影平坠 ？

* 4.6 软阴影PCF过滤 百分比渐进过滤
* 4.7 混合Cascade
* 

4.8 抖动过渡

抖动 更省性能模式

4.9 剔除偏差 网易P8相当腾讯3-3 腾讯3-2

5半透明

5.1 阴影模式

5.2裁剪阴影

半透明 需要ShadowCaster Pass里面用clip 裁剪掉投射的图形。

5.3 抖动阴影

5.4 没有阴影

5.5 Unlit 阴影投射

第五课时 烘培灯光

灯光图和反射球

烘培静态全局光,采样Light Maps和反射球 和LPPVs

支持自发光表面

1 烘培静态光

提前存在LightMap里面和Probe，后者是全局光照的一部分,静态的不能在运行时进行改变,增加构建的时间的内存。

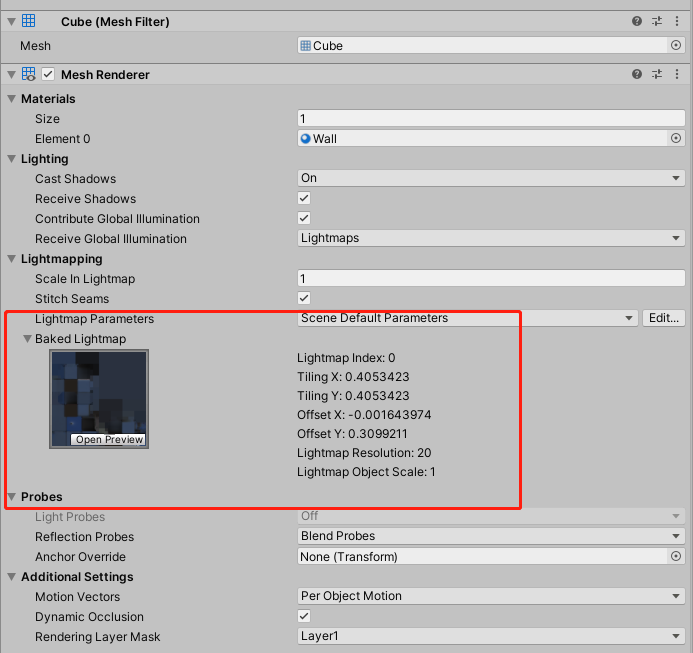
1.1场景灯光的设置

方向光模式跟NormalMap有关系？ 现在选择No-Directional

1.2 静态物体

灯光模式Mix，告诉Unity要对这盏光烘培间接光,除此之外这盏光还是一盏实时光。灯光设置勾选自动生成LightMap

勾选LightMap 使用物体会变成静态



* 1. 完全的Bake灯光

Bake 场景完全黑掉

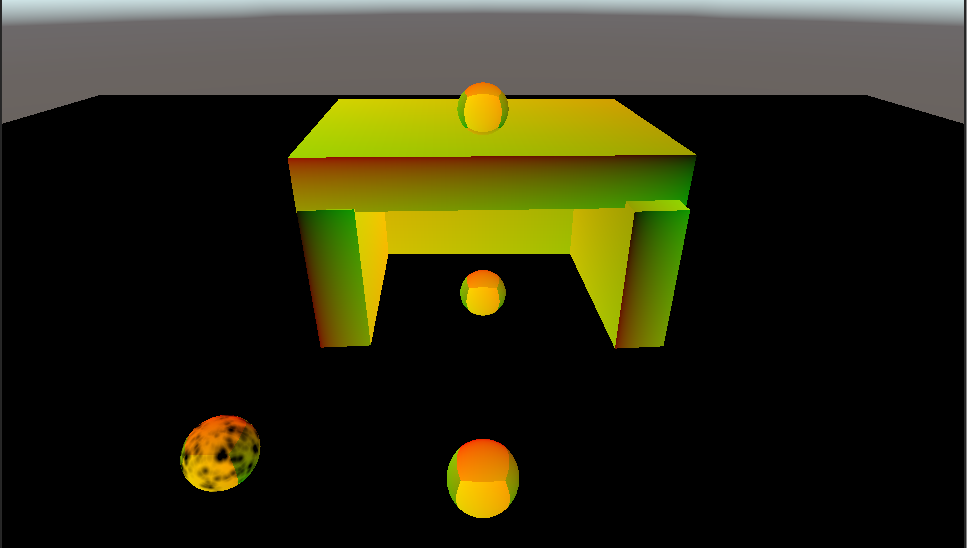
1. 采样 烘培灯光

2.1全局照明

全局灯光的高光是反射球是SSR Screen-space reflections 屏幕空间反射

2.2 LightMap 坐标

设置每个物体逐物体数据LightMaps灯光贴图DrawingSettings 里面进行设置



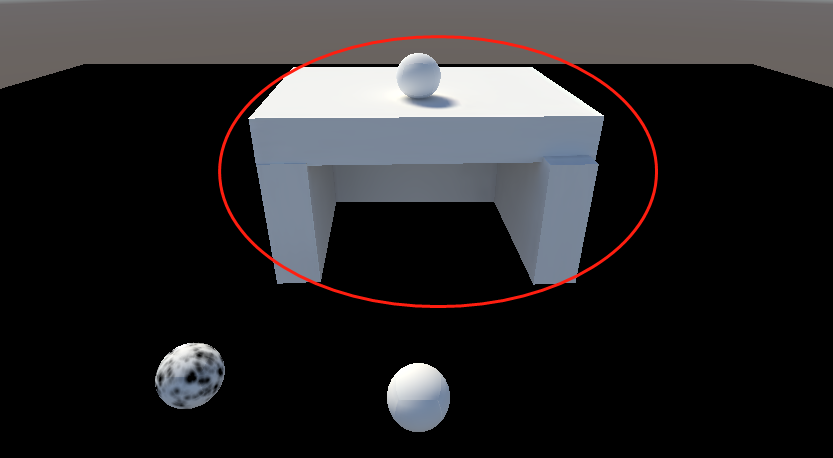
静态的物体会显示出UV 动态的物体 是黑色的没有UV信息

2.3 转换Light Map 坐标

Light Map UV坐标是每一个Mesh自动生成的或者导入网格数据的一部分。增加unityDynamicLightmapST 否则SRP Batch兼容性是会打断的。UnityPerDraw是在被需要的时候进行实例化。

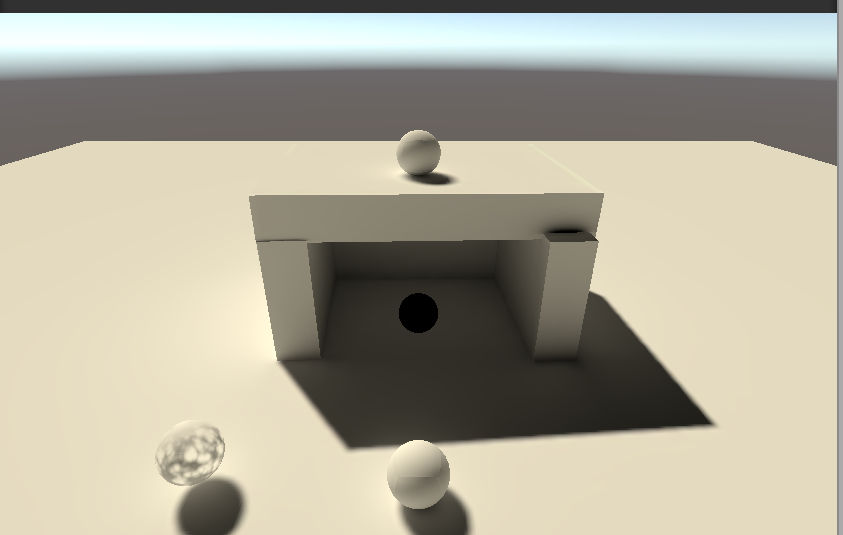
2.4 采样LightMap

采样成功LightMap

HDR不压缩 每次改变静态非静态重新Bake

2.5 禁用掉环境光

环境光强度设置为零 Bake出来就不包含天空球的间接光部分



1. 光探针

球谐光,L2球面谐波

* 1. Light 探针组

Bake之后才能看到动态物体受哪些光探针影响选中就可以,自动合并两个LightProbe区域。

* 1. 采样Probes

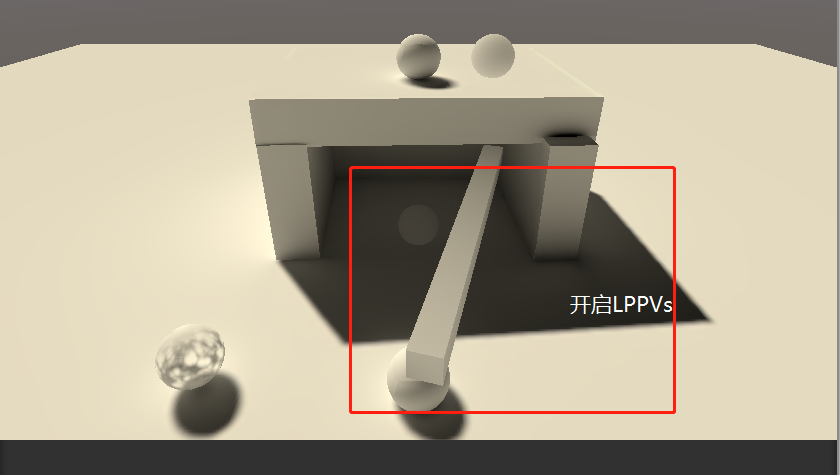
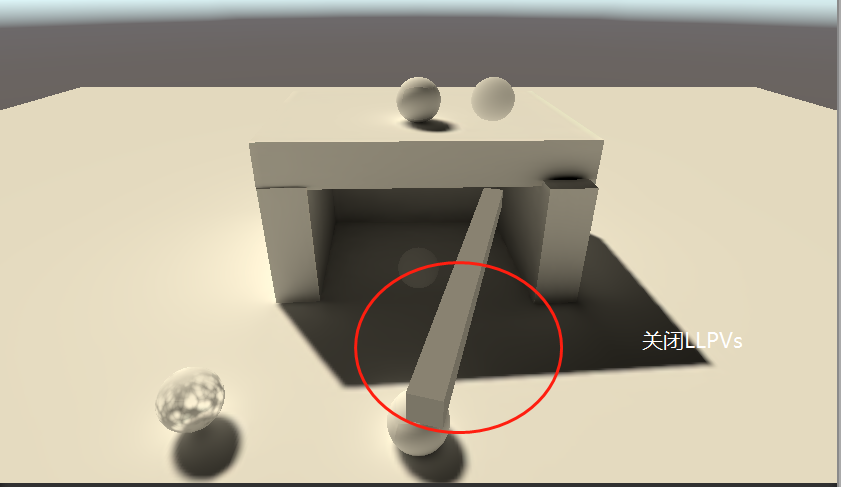
需要一个方向

* 1. LightProbe 体积

对大物体的支持，均匀黑暗是不正确的？在阴影的地方黑在外面应该间接光更亮一点才对。

* 1. 采样LPPVs

unity\_ProbeVolumeParams.x 进行判断的 世界坐标和法线信息进行采样



1. Meta Pass 干什么用的？ 是根据物体反射率反射到周围的颜色是怎么样的

Unity 烘培光的间接反射部分

4.1统一的输入

HLSLINCLUDE

4.2 Unlit

4.3 Meta 灯光模式 ？？？

4.4 lightMap 坐标

4.5 漫反射反射率

Mix灯光 实时光+间接光部分 细节更好

成功

5 自发光表面

一些表面他们自己会发光,然而这个影响会对Bake 灯光有贡献。

5.1 发光的光源

HDR配置

5.2烘焙自发光

自发光影响周围环境而且是静态的。

6 Bake 半透明

6.1 硬编码属性

6.2 复制属性

7 Mesh Ball

7.1 光探针

7.2 LPPV

动态物体的间接光部分 静态是LightMap 自发光也会成为一部分间接光部分

<https://blog.csdn.net/weixin_30412167/article/details/95799502> 解释MetaPass为全局光照提供反射率和自发光项

1. 静态阴影

Bake直接遮挡

Bake 静态阴影

结合实时光 Bake 阴影

Mix 实时光和Bake 阴影

支持四盏光的ShadowMask 光 支持更远的阴影

1 烘培阴影

实时阴影+ShadowMask 远距离阴影

* 1. 距离ShadowMask

实时+静态 ShadowMask是静态

ShadowMask 包含阴影的衰减 静态物体投射的阴影存在红色通道里面。光是不能够旋转的 不然阴影将变得无效

Indirect Multiplier 灯光设置 影响间接光的Bake  
 1.2 检测一个ShadowMask

1.3ShadowMask数据

1.4 遮挡(Occlusion Probes)

Unity 烘培Shadow Mask 到LightProbe里面 得到ShadowMask是足够的,但是打断了GPU Instance Unity Instancing 只有定义 SHADOWS\_SHADOWMASK才会自动进行做

1.5 LPPVs

1.6 Mesh Ball Instance

2 混合Shadows

在超出实时阴影距离的时候使用ShadowMask

2.1 当可用的时候用Baked数据

2.2 过渡到Baked 阴影

要根据深度从实时阴影过渡到烘焙阴影，我们必须根据全局阴影强度在它们之间进行插值.然而另外我们也要应用光的阴影强度。

2.3 仅仅用Baked 阴影

现在实时阴影和Bake阴影进行绑架,没有实时阴影 Bake阴影也会消失支持只有Bake阴影的情况。

2.4 总是用ShadowMask

ShadowMask 是静态物体烘培 静态 接受动态物体获取实时阴影 但在阴影距离内静态物体不是实时进行计算的。

3 多种灯光

因为ShadowMask 拥有四个通道它可以支持四个混合灯光.超过四个混合灯光会发生什么？其他灯光有限制区域。两个灯光共享相同的Bake 阴影

3.1 ShadowMask 通道

LightBakingOutput lightBaking.occlusionMaskChannel

3.2 选择合适的通道 - https://zhuanlan.zhihu.com/p/126362480

Isn't a dot product better than indexing a channel?

什么是Subtractive 模式？

减法照明是一种替代的方式来结合烘烤的照明和阴影，只使用一个单一的光地图。这个想法是，你完全烘烤一盏灯，但也使用它来实时照明。然后计算该光的实时漫射光，采样实时阴影，并使用它来确定有多少漫射光被阴影，这可以从漫射GI中减去。

第七课时 LOD和 Reflections

使用LOD Groups

在LOD 之间使用渐变 Cross-Fade

反射环境通过采样 反射球

支持可选择的 菲涅尔反射

1 LOD Group组

* 1. LOD 组 组件

根据占据视口的尺寸决定使用LOD,LOD Bias

* 1. 添加LOD Group
  2. LOD 过渡

Cross Fade 淡入淡出

它使用自己的LOD系统来折叠树和在3D模型和广告牌之间的转换。我们不会用它LOD 使用广告牌

* 1. 混色过渡

InterleavedGradientNoise 很像过滤网门

* 1. 动画过渡

1. 反射
   1. 间接BRDF

粗糙度会散射这些反射

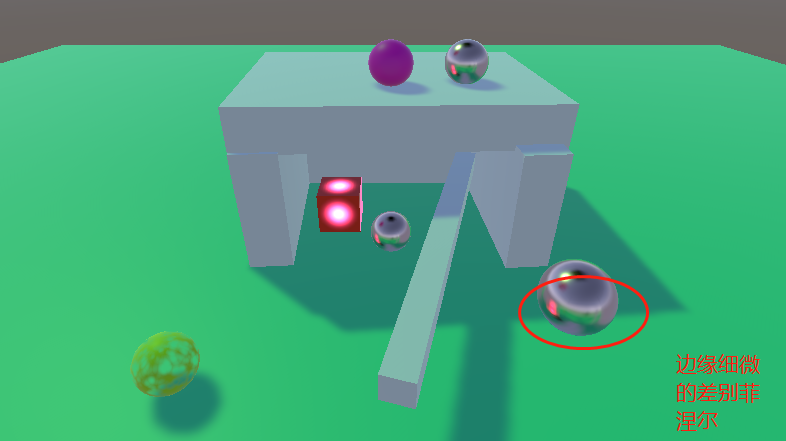
* 1. 采样环境

现在采样的天空球

* 1. 粗糙度反射
  2. 菲涅尔反射

当时掠射角观察物体时,物体体现出全美的镜面效果。因为光线在他们之间反射完全不受影响,我们近似模拟菲涅尔

* 1. 菲涅尔滑动条 调节菲涅尔反射
  2. 反射探针



Instance 不能使用反射球 总是使用天空球 混合是SRP Bathch不兼容

* 1. 解码反射球

第八课时 复合贴图

创建一个环绕材质 支持MODS Mask 图 引进第二个细节图 执行一个切线空间法线贴图

1 环路材质

* 1. 反射率纹理
  2. 自发光纹理

2 遮罩纹理

2.1 MODS

2.2 遮罩纹理输入

2.3 金属度

2.4 粗糙度

2.5 遮挡

描述缝隙和洞的遮挡 阴影 暗部区域

遮挡只用间接环境照明部分

3细节纹理

3.1 细节UV 坐标

3.2 细节反射率

3.3 细节粗糙度

3.4 过渡细节

4 法线纹理

* 1. 采样Normal 纹理

What does DXT5nm mean？

* 1. 切线空间
  2. 给阴影Bias插值 法线
  3. 细节法线纹理

1. 可选择纹理
   1. 法线纹理
   2. 输入配置
   3. 可选择的遮罩纹理
   4. 可选择细节纹理

第九课时 点光源和斑点照射光源 光的限制影响

支持更多的光源类型比单一的方向光

包括实时点光源和照射光源

烘培光源和阴影对于点光源和照射光源

限制渲染最多8盏其他光源对于每一个物体

1: 点光源

* 1. 其他光源数据
  2. 点光源设置
  3. 着色
  4. 距离衰减
  5. 灯光范围

添加范围之后.灯光包含在一个边界球内。明确他们的位置和范围加上范围衰减 可看到超出范围看不到灯光的效果

1. 聚光灯

受圆锥体的限制

* 1. 方向
  2. 聚光灯角度

聚光灯有一个角度控制光椎的宽度,还有一个内角控制光线什么时候开始衰减,内角和外角以及弧度

* 1. 配置内角

CustomEditorForRenderPipeline属性 暴漏内角属性

1. 烘培灯光和阴影
   1. 完全烘培

使用啦错误的灯光衰减

* 1. 灯光代理

我们可以告诉Unity使用不同的灯光衰减

* 1. ShadowMask

1. 每物体灯光

如果灯光只影响一个大物体的一小部分,但是大物体所有的片段都会执行一遍,造成灯光计算的浪费。

* 1. 每个物体灯光
  2. 灯光索引处理
  3. 使用灯光索引

注意，如果启用了每个对象的光照，GPU实例化的效率就会降低，因为只有光照计数和索引列表匹配的对象才会被分组。SRP批处理程序不受影响，因为每个对象仍然得到自己的优化draw调用。

第10课时 点光源和聚光灯 阴影

混合烘培 和实时阴影对于点光源和聚光灯

增加第二个阴影图集

渲染和采样阴影在透视投影下

用自定义立方体纹理

1:聚光灯 阴影

* 1. 混合灯光

超过阴影距离的最大Cascade sphere, Cascade 仅仅适用于方向光

* 1. 其他实时光阴影

支持16盏 其他光源阴影

* 1. 两个图集
  2. 渲染聚光灯阴影
  3. 没有 Pancaking
  4. 采样聚光灯阴影
  5. Normal 偏差

Cot =tan分之一

* 1. 钳制采样

问题 Normal bias会使采到锥形边界的地方。

1. 点光源阴影 点光源是向四面八方进行扩散的 所以点光源阴影渲染到6个面中

需要渲染阴影到一个立方体内,一个点光源占据6个Tile 没有正确渲染 todo

* 1. 6个块对于一个灯光
  2. 渲染点光源阴影
  3. 采样点光源
  4. 画正确的面
  5. FOV偏差

第11课时 后期处理Bloom

创建一个简单的Post 后期栈

选择渲染的图片 执行后期当需要的时候

制作一个Bloom效果

1 后期栈

* 1. 设置资源
  2. 栈物体
  3. 用Stack
  4. 强制清除
  5. 线框
  6. 定制绘画

现在我们用的是一个四边形 覆盖全部的屏幕空间,多余的Block渲染,因为GPU渲染片段是并行在小的Blocks。可以有益对一致的本地缓存

裁剪空间位置和屏幕空间UV 信息

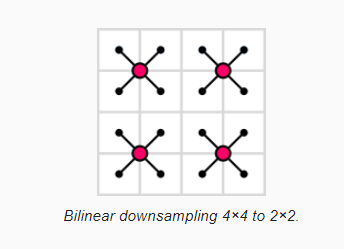
裁剪空间不是NDC空间

* 1. 不总是使用后处理
  2. 复制

1. Bloom

Bloom经常使物体发光这个是最基本的物理学的,现在是LDR 的Bloom 更真实的是HDR Bloom.

3.1 Bloom金字塔



双线性降采样 4x4 到2x2

Bloom代表性的是分散颜色,明亮的像素会渗透到附近暗的像素,因此出现发光,最快最简单的方式模糊一个纹理通过复制他到另外一个有一半高和宽的纹理当中。

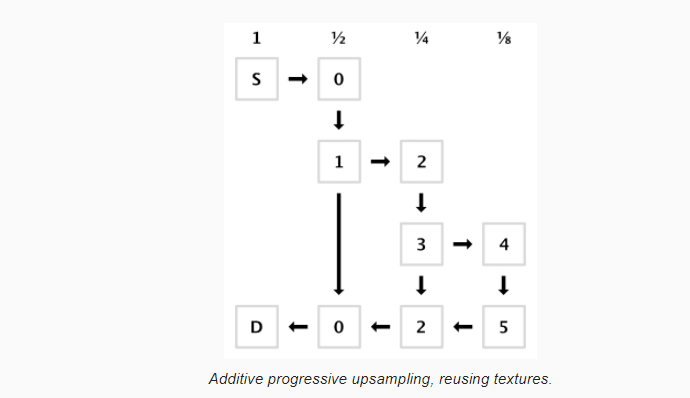
复制过程中的每个样本最终都采样在四个源像素之间进行采样。用双线性过滤2x2的像素块

2.2 配置Bloom

2.3 高斯过滤

用一个小的2×2滤波器向下采样会产生非常块状的结果,双线性+9x9 双线性是硬件 在加上滤波盒变为18x18 虽然这个操作混合了81个样本，但它是可分离的 可以分为两个Pass一个水平的Pass和一个垂直的Pass。因此我们需要18次 水平九次垂直9次,每个迭代有两次的绘制过程。具有相对权重的可分离3×3滤波器。权重是由帕斯卡三角形推导出来的。对于一个合适的9×9高斯滤波器，我们选择三角形的第九行，即1 8 28 56 70 56 28 81 1。但是这使得滤光器边缘的样本贡献太弱而无法被注意到，所以我们移到第十三行并剪掉它的边缘，到达66 220 495 792 924 792 495 220 66。这些数的和是4070，所以每个数除以它得到最终的权重。

2.4 额外的模糊



叠加的模式 4跟5叠加 4和5指的是RT

2.5三线性升采样

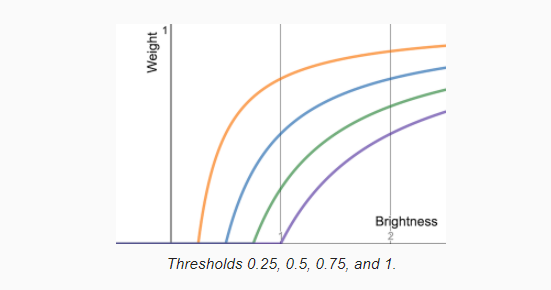
双线性会使Bloom显示块状的感觉,双线性过滤是硬件支持的功能

2.6 半分辨率

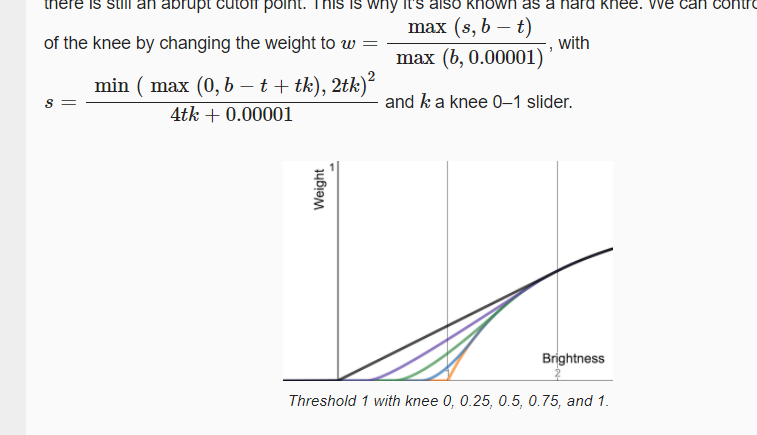
减少消耗用一半的分辨率

2.7阈值





Knee 曲线



LDR 颜色值不会大于1

* 1. 强度

第12课时 HDR散射和色调映射(Tone Mapping)

渲染到HDR纹理 减少Bloom亮点 增加散射Bloom 支持多重的色调映射模式

1 高动态范围

Is (1,1,1) really white?

介绍HDR DisplayB8G8R8A8\_SRGB 每个像素有32bit一旦渲染完成，缓冲区被发送到显示，它解释为sRGB颜色数据

Unity目前不支持HDR显示。所有显示器都假定为LDR sRGB

渲染到HDR Buffer中去

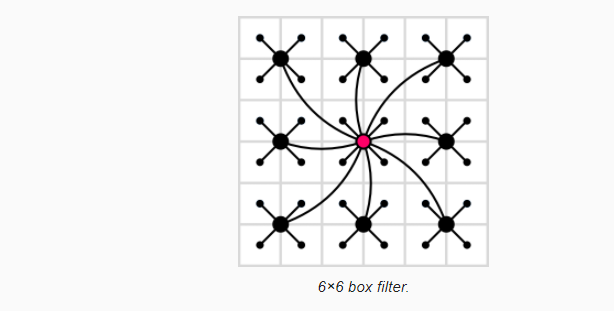
* 1. HDR 反射球
  2. HDR 摄像机
  3. HDR RT格式

  R16G16B16A16\_SFloat纹理格式每个像素64bit

* 1. HDR 的后期处理

例如，由于向下采样，取0、0、0和1的一个2×2块的平均值时，结果为0.25。但是如果HDR版本的平均值是0、0、0和10，那么结果就是2.5。与LDR相比，似乎0.25的结果被提升到了1

* 1. 完美的银光灯效果



权重平均代替,基于颜色的发光性(Luminance)

A sample's weight is 1l+11l+1 with ll its luminance. Thus for luminance 0 the weight is 1, for luminance 1 the weight is ½, for 3 it's ¼, for 7 it's ⅛, and so on.

1. 散射Bloom
   1. Bloom模式

叠加模式和节能的散射模式

* 1. 扩散限制
  2. 阈值

不会模糊暗部,会使图像变得更清晰一些。

1. 色调映射
   1. 额外得后处理设置

在Bloom之后进行ToneMapping

现在是分开得 其他是Uber Shader

* 1. ToneMapping模式
  2. 莱因哈德

精度问题 需要限制值

* 1. 中性得
  2. ACES

ACES减少暗部，提高了对比度

第13章 ColorGrading

调整颜色氛围

执行颜色分级 复制多个URP/HDRP 颜色分级工具

用一个颜色查找表

1颜色调整

* 1. 在ToneMapping之前进行颜色矫正
  2. 设置
  3. 曝光值
  4. 对比度
  5. 滤镜
  6. 色调偏移
  7. 饱和度

1. ：更多得控制
   1. 白色平衡

图像温度得感知

* 1. 色调分离
  2. 通道混合
  3. 阴影中间调亮点
  4. ACES颜色空间

1. 查找表
   1. 查找表分辨率
   2. 渲染一张2D LUT查找表
   3. LUT 颜色矩阵
   4. Log C 查找表
   5. Final Pass
   6. LUT 条带

第14章 多摄像机 摄像机混合和渲染层

渲染多个摄像机用不同得后期 层摄像机用自定义混合

支持渲染Layer Mask

遮罩灯光每个摄像机

1：联合摄像机

* 1. 分区屏幕
  2. 分层摄像机
  3. 分层透明度
  4. 自定义混合

混合之前的摄像机层是有意义的对于重叠的摄像机

1.5 RT

1.6 Unity UI

1.7 后期设置给每个摄像机

2 渲染层级

* 1. 剔除遮罩
  2. 调整渲染层级遮罩
  3. 发送遮罩到GPU
  4. 重新解释Int 作为 float
  5. 摄像机渲染层级遮罩
  6. 遮罩灯光对于每一个摄像机

实时灯光可以被Mask 但是像LightMap Bake的是不可以被Mask

第15章粒子 颜色和深度纹理

支持flipbook 近过渡 软粒子和扰动粒子

确定片段深度给正交和透视摄像机投影

复制和采样颜色深度缓存区

创建基于深度的衰减和扰动粒子 依靠一个颜色和深度纹理

1 不发光的粒子

* 1. 粒子系统
  2. 不发光粒子Shader
  3. 顶点颜色
  4. 翻页书

公告牌粒子

* 1. 翻页混合

1. 过渡摄像机的近处

处理接近近平面的粒子的另一种方法是根据它们的片段深度使它们淡出

* 1. 片段数据

SV\_POSITION 代表着裁剪空间的位置 顶点的

* 1. 片段深度
  2. 正交深度
  3. 基于距离的渐变

1. 软粒子 是什么？

粒子和几何交互时有效果问题 所以需要软粒子 为了实现这一工作，粒子的碎片深度必须与之前在相机缓冲器中相同位置绘制的深度进行比较,这要求我们必须采样深度Buffer.

* 1. 分离深度Buffer
  2. 拷贝深度图

我们不能在深度缓冲区被用于渲染的同时对它进行采样这意味着深度纹理只在渲染透明对象时可用

* 1. 没有后处理拷贝深度图
  2. 重建视野空间深度

画重点 面试经常问到

屏幕空间\_ScreenParams

x is current render target width in pixels x是当前渲染目标在像素值中宽度  
y is current render target height in pixels y是当前渲染目标在像素值中的高度  
z is 1.0 + 1.0/width z是1.0+1.0/宽度   
w is 1.0 + 1.0/height w是1.0+1.0/高度

SAMPLE\_DEPTH\_TEXTURE  仅仅返回R通道

 uniform float4 \_ZBufferParams;  
用于线性化z buffer  
x = 1-far/near  
y = far/near  
z = x/far  
w = y/far

* 1. 可选择的深度纹理
  2. 漏掉的纹理
  3. 背景附近渐变粒子
  4. 不拷贝纹理支持
  5. 线框和深度

1. 扰动
   1. 颜色拷贝纹理
   2. 采样颜色Buffer
   3. 扰动向量
   4. 扰动融合