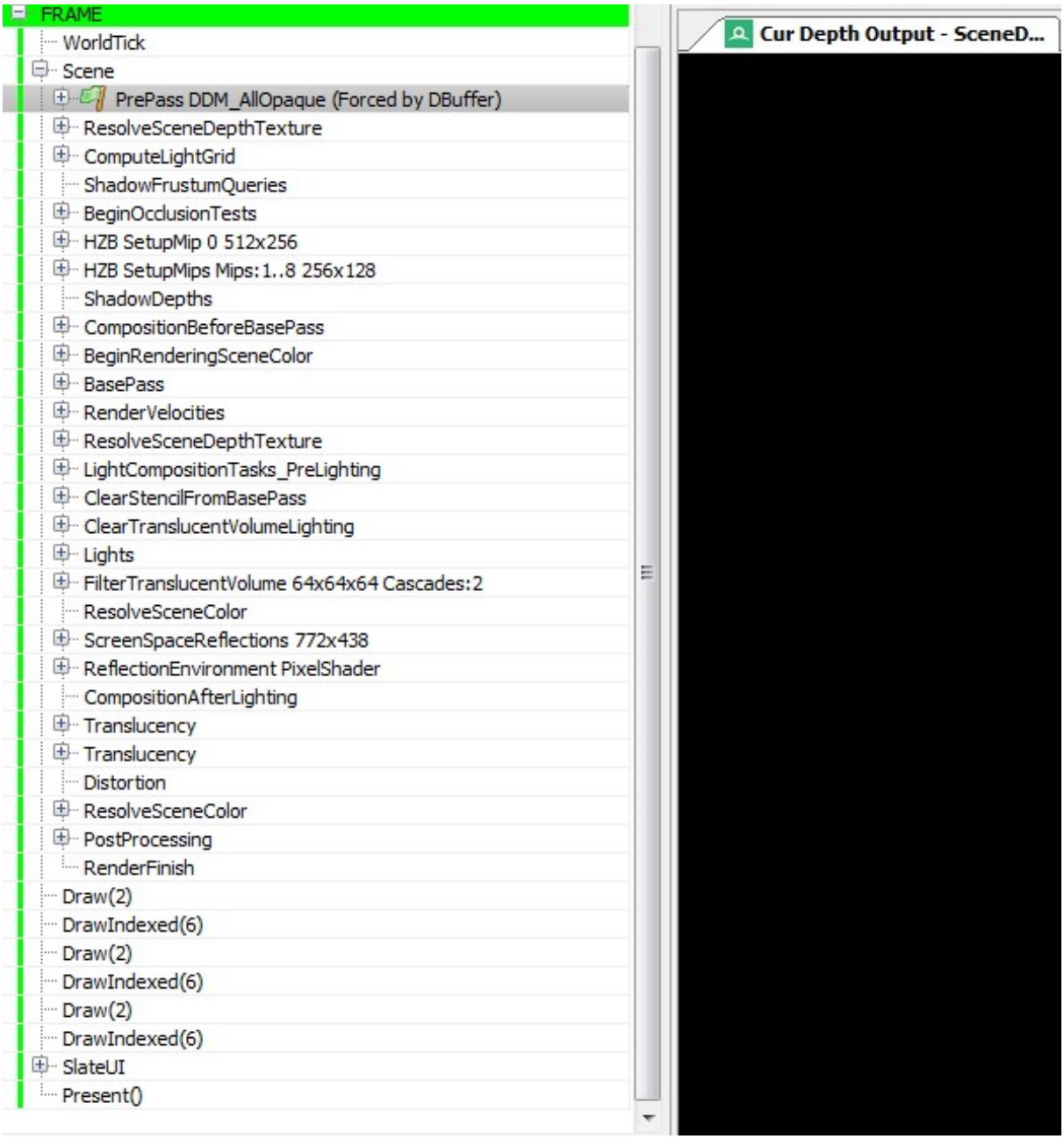
虚幻引擎 4 渲染流程分析

2018-01-24 王文涛 虚幻引擎

今天为大家带来的是转载自虚幻引擎爱好者王文涛的博客文章《虚幻引擎 4 渲染流程分析》，这篇文章通过 Renderdoc 的分析通俗地介绍虚幻引擎 4 的渲染过程。 点击阅读原文跳转到作者博客。

UE4作为当今商业引擎界的大佬，渲染和图形质量一直是首屈一指的水准，但是相对于unity来说UE4基本上是一套完整方案提供，不通过源码修改对渲染进行定制的可能性比较小，而且同时UE4这方面的文档很少，因此这篇文章就是想通过分析UE4的渲染过程，来给大家针对自己使用ue4开发的游戏的内容特点做出优化带来启发。

我们使用Renderdoc对UE4(PC，DX11）截帧，UE4的版本为4.18. 可以看到UE4一帧画面的渲染过程如下



可以看到的是整个渲染流程还是很清晰明了的，接下来就会逐步分析每个过程。

1.Z-Prepass

UE4在deferred shading 过程之前这个，会有一系列的culling过程剔除掉不需要的像素或者几何体，基本上可以猜测是UE4是为了减轻后期在deferred shadinggbuffer 生成中的庞大计算量，

第一遍的zpass会先渲染一遍场景中的几何，用于生产SceneDepthZ以及HZB buffer，格式为R24G8TYPELESS

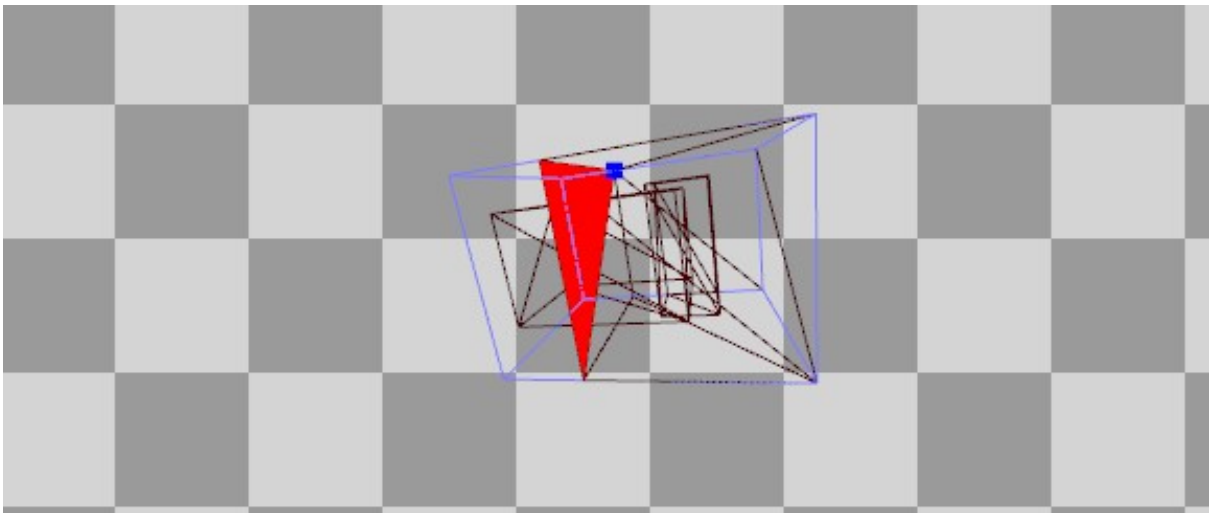
2.Compute light grid

在Pre-Z之后UE4会把场景中的灯光按照屏幕空间分成相应的grid，类似于cluster shading的方法，注意这里的grid只考虑点光源，聚光灯，以及reflection captures，UE4这一步是通过compute shader实现的，所以只在sm>5.0的平台上有。具体shader代码在LightGridInjection.usf，阅读代码之后我们可以发现 UE4的灯光空间grid的划分是按照指数增长的。也就是每个grid的z随着距离会增长。

在真正计算光照时，我们可以用GridIndex来快速决定某点是否受到灯光影响。Lightculling的方法在forward下对于提高灯光的渲染效率是十分有用的，但UE4在DS下仍然保有了这一个过程。其效果存疑，初步推测是为了和UE4新加的Forward renderer统一。

3.Occlusion query

这一步在light culling之后，和Pre-Z pass 不同的是，Occlusion query 主要在物体级别做culling。ue4同样使用的是hardware occlusion queries（GPU query）的技术。在这一个pass中，所有的不透明物体会被渲染为一个occluder（包围盒）：



在根据深度计算query之后，query的数据会传回cpu,我们就可以计算每个物体有多少像素可见。这样我们就能知道物体最终是否会被渲染。

在不透明物体的query pass之后。Unreal 还有一些其他的query pass，例如灯光（点光源）会有一个ShadowFrustumQUeries（一般是画一个球体）反射则有 PlanarReflection queries（一般是画一个 Cube)

4.HZB generation

接下来UE4会生成场景的Hi-z（Hierarchical Z），R16_Float 格式，这一步也就是对之前的zbuffer做连续的downsample。HZB buffer会在之后的计算中起到很多作用，特别是Image based 的lighting技术，例如SSR等等。

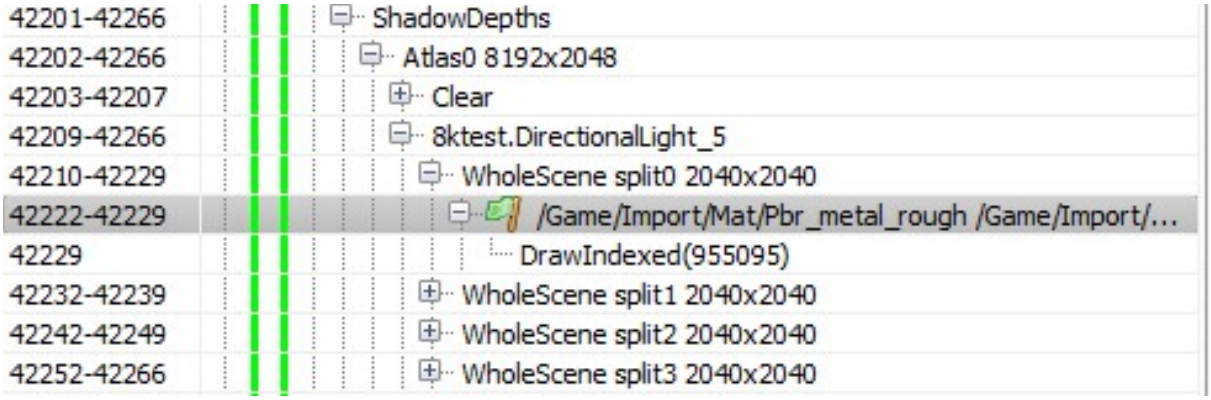
5.ShadowMap 渲染

接下来的一步就是渲染shadowmap（shadowDepth,注意，这里指的是实时阴影的计算。根据UE4中灯光类型的不同，实时阴影

的计算也有一定的差别。
UE4中的灯光类 型分为stationary， static，moveable三种，相应的每种灯光cast realtime shadow的方式也不同。

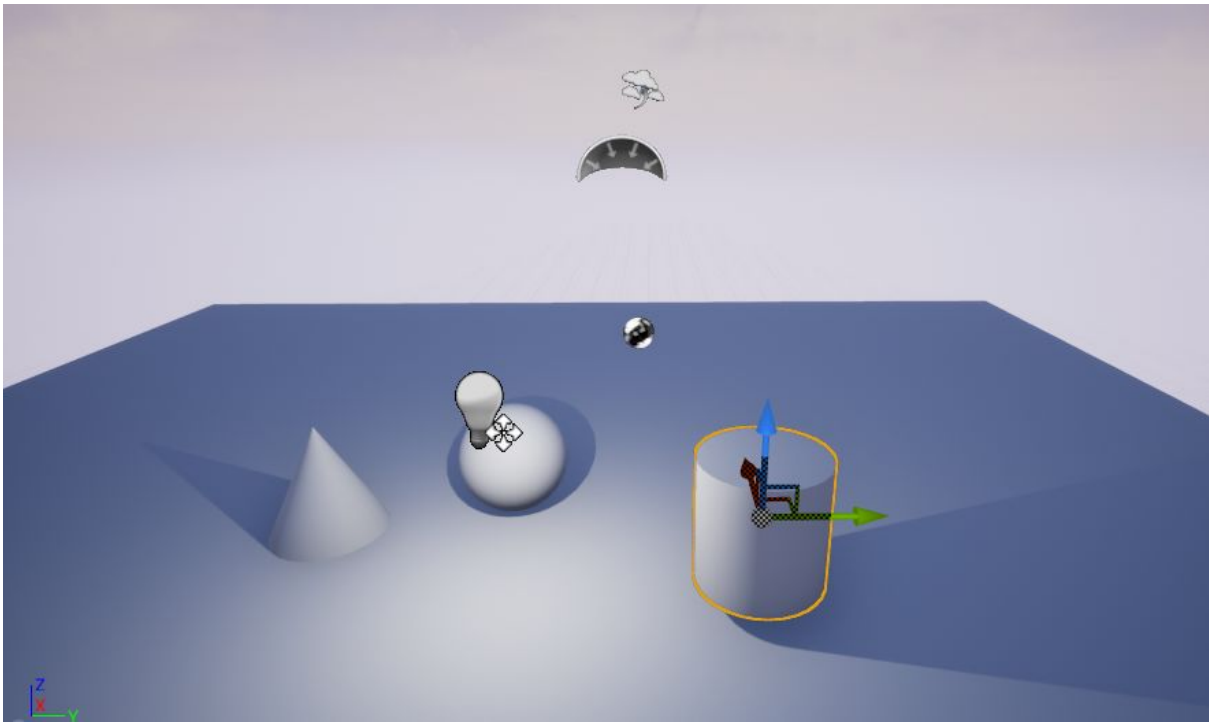
对于stationary light，静态物体的阴影会bake到static shadowmap，shadowmap只计算标记为动态物体的阴影，而对于dynamic light 会对所有物体投射阴影，而静态灯光不会产生实时阴影。

ue4首先会渲染方向光的阴影，一般会渲染3split的cascade shadow ，所以我们能在截帧信息看到split0， split1和split2，注意cascade split数目在ue4中也是可以在方向光参数中设置的变量。

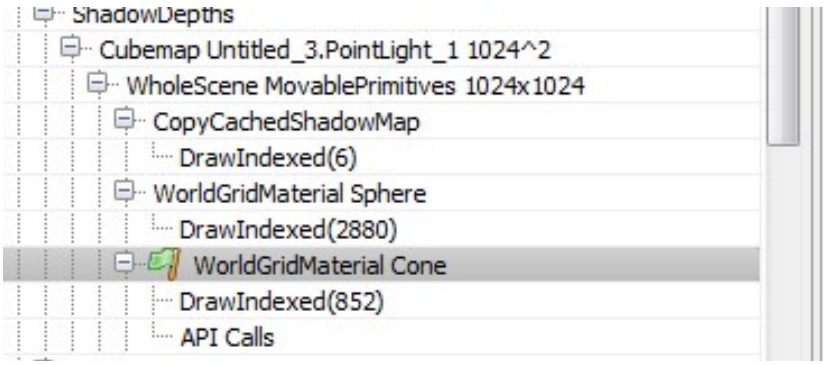


之后是stationary light的shadow渲染，注意这里只针对场景中的moveable的物体。
最后是对于movable light的渲染，对于movable的方向光，ue4仍然是cascade shadow map计算阴影，需要注意的是对于movable的点光源，ue4使用了cubemap shadowmap，在cubeshadowmap的第一个pass CopyCachedShadowMap中，

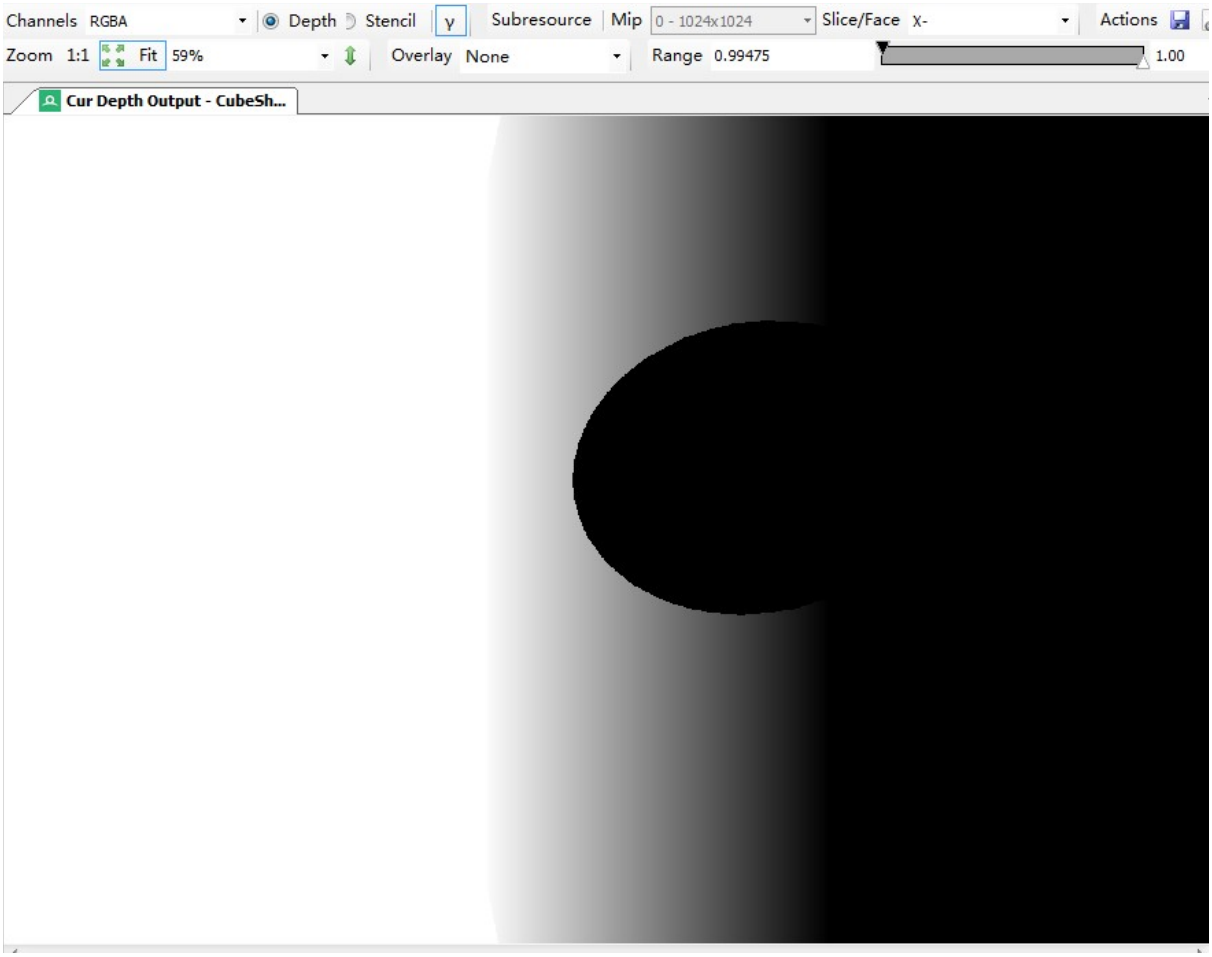
ue4会直接cachecopy static物体的shadowmap，例如这个场景中

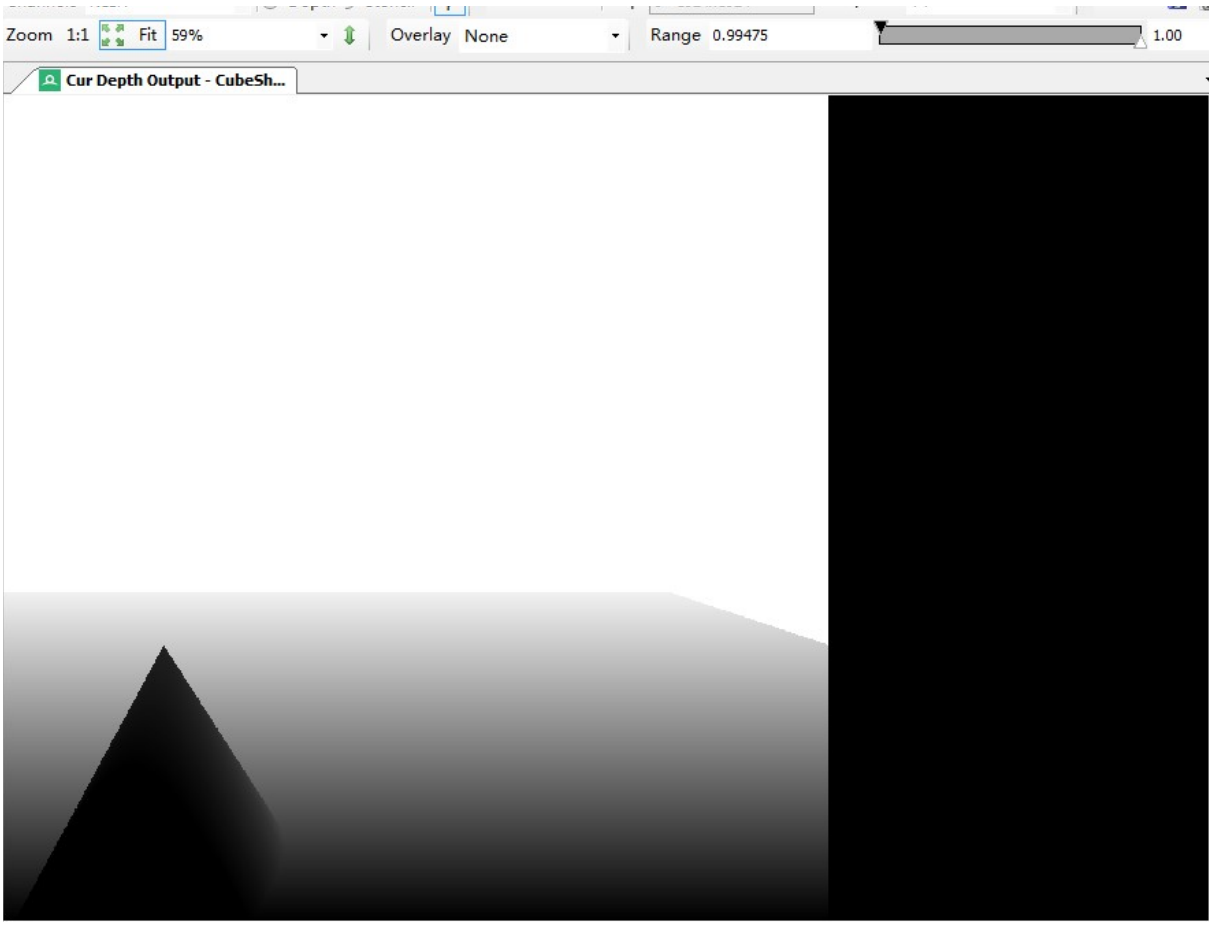


圆柱体为static，其他两个物体设为movable，因此最后我们能看到Shadow 只画了两个几何体。



最后动态物体的shadow会添加在这个cubemap上面。


Channels RGBA
Depth
Stencil
Subresource
Mip
0 - 1024x1024
Slice/Face
Actions
Zoom 1:1
Fit
59%
Overlay None
Range 0.99475
1.00
Cur Depth Output - CubeSh...

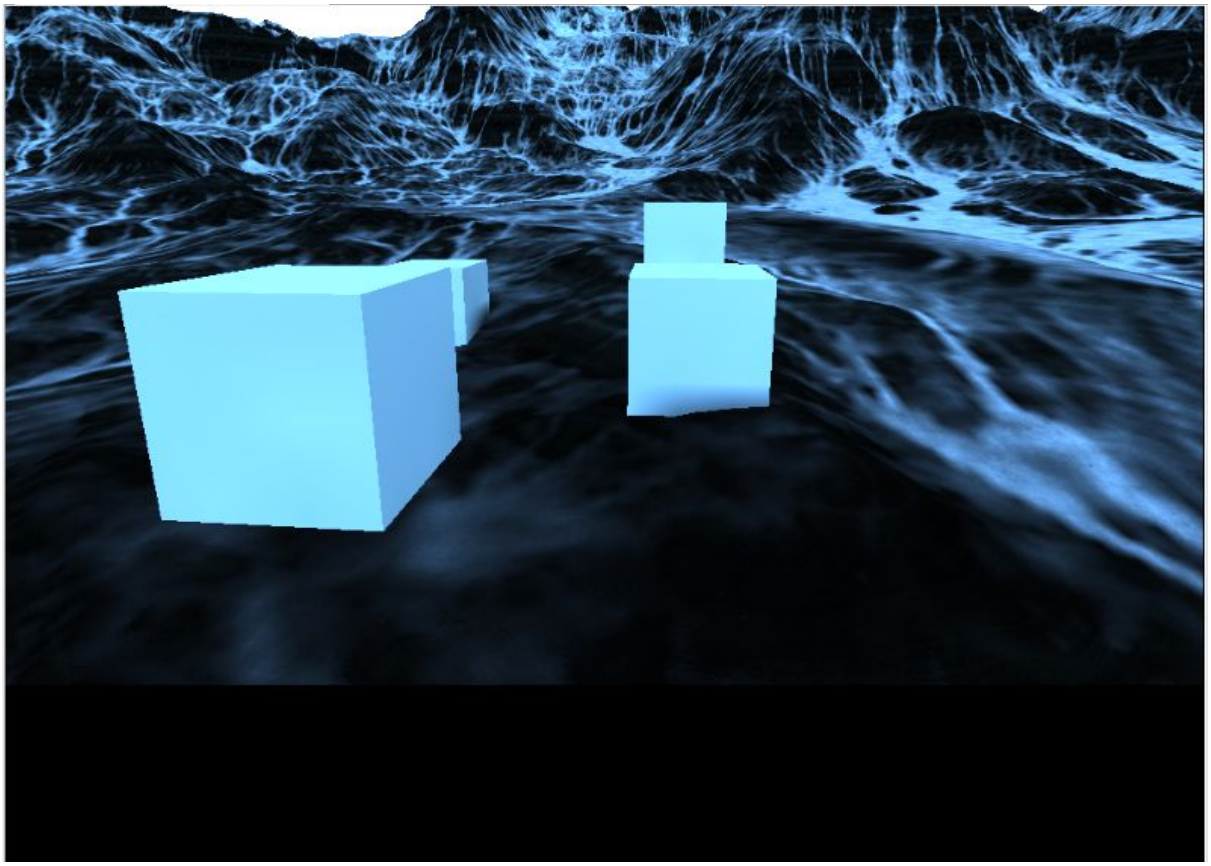

Zoom 1:1
Fit
59%
Overlay None
Range 0.99475
1.00
Cur Depth Output - CubeSh...

注意 shadowcubemap使用了geometry shader来选择画在那个面上。

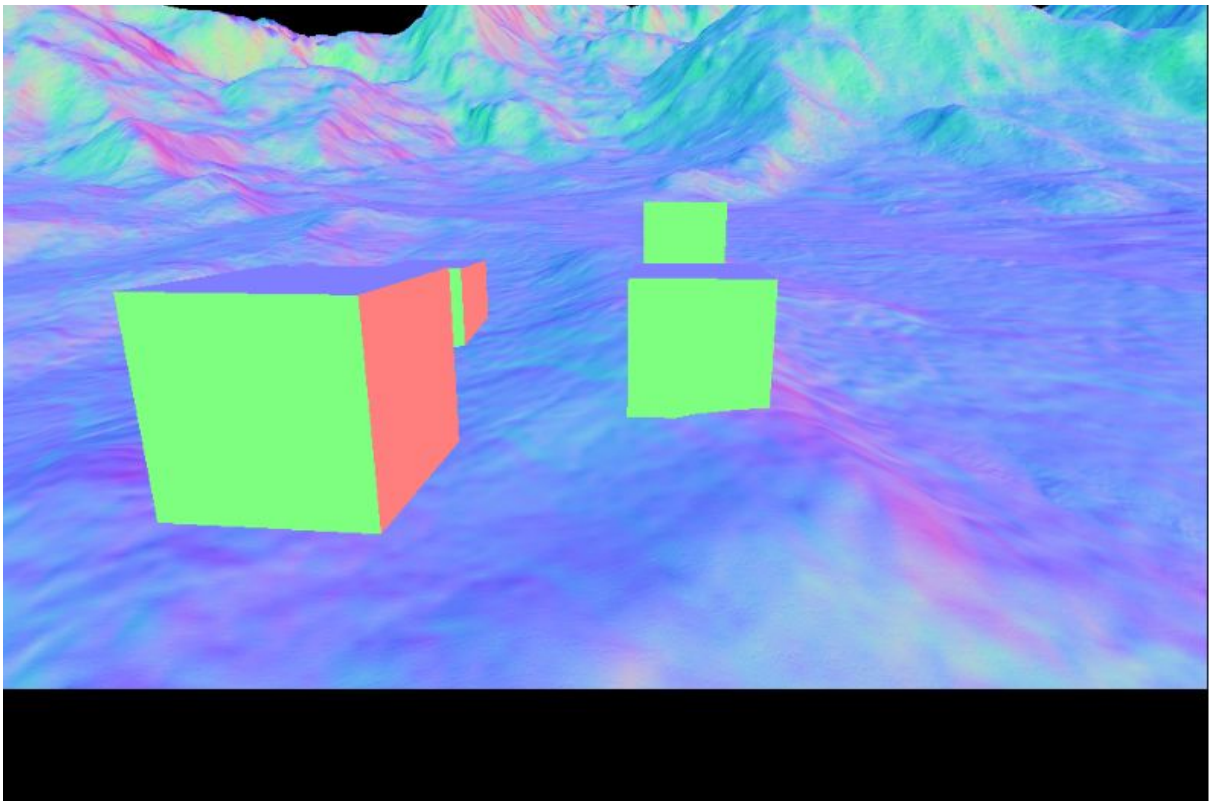
6.G-prepass

其实在g-prepass之前还会渲染一个volumtric fog（如果场景中有的话）这里我们先跳过，

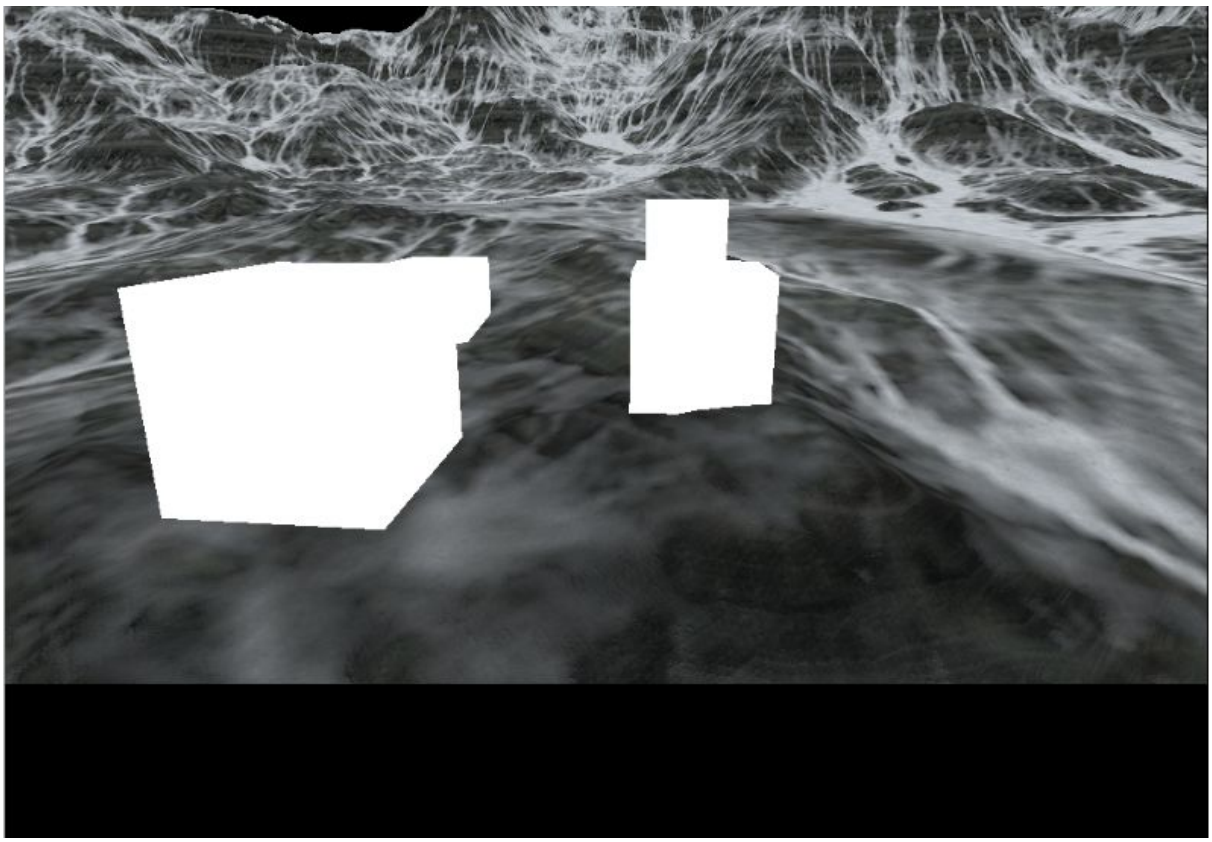
G-prepass就是ue4中常说的basspass，这个bass会真正的渲染场景并产生我们在deferred shading 中需要的G-buffers:



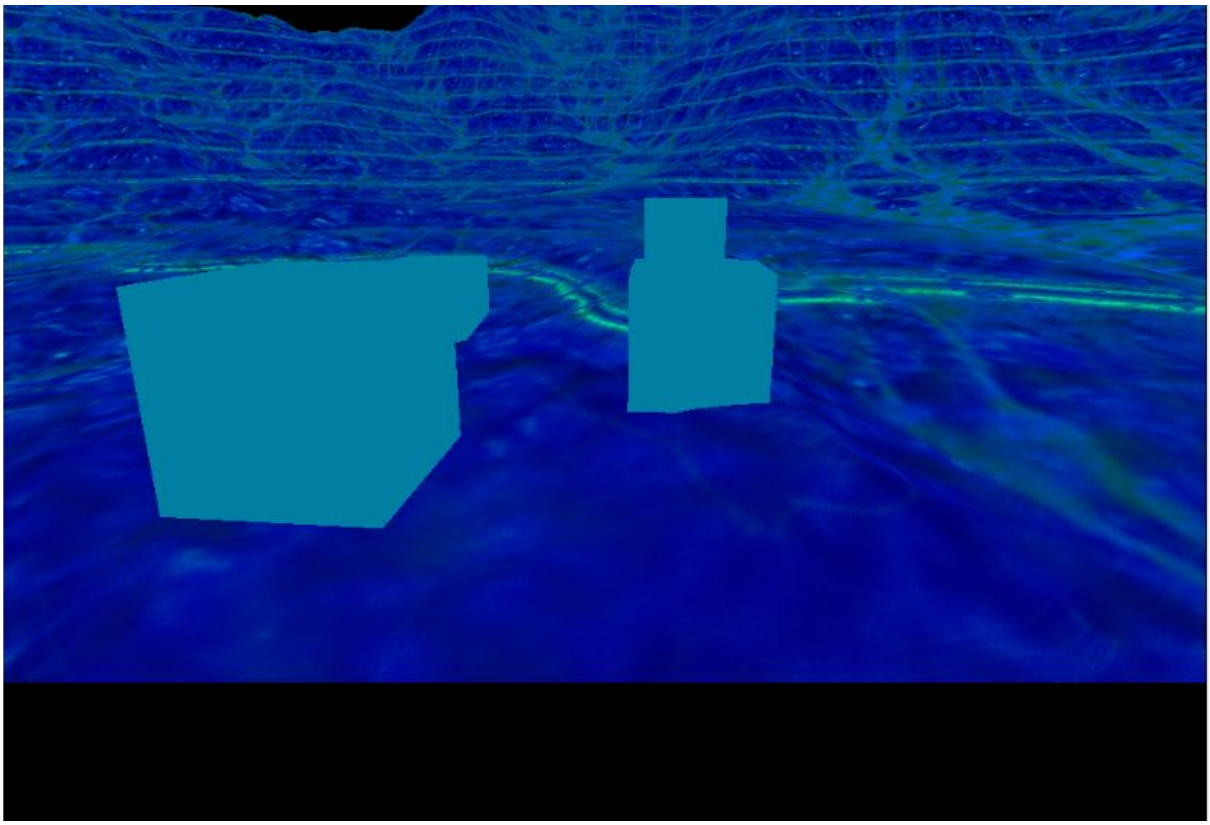
SceneColorDefferd:包含了间接光照信息，例如lightmap和lightprobe（ue4叫ILC）



场景Normal

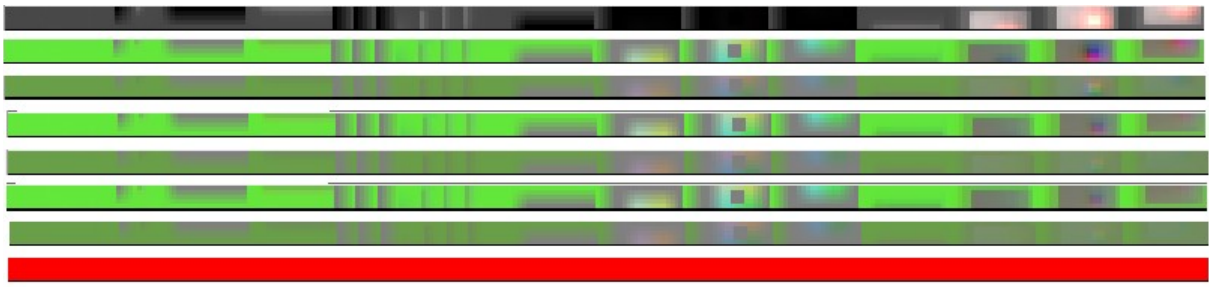


场景Albedo颜色



PBR Specular信息

除此之外还有针对特殊shading模型的特殊Custom Data RT（例如 头发的tangent sss等）和Pre baked shadow factors RT，一般情况下UE4的渲染需要5-6个RT输出，除了产生GBuffer的计算之外，在这一步UE4还会计算完间接光照的信息，主要包括采集lightmap信息（静态物体）和球谐函数信息（Indirect lighting cache或者Volumetric light map）
UE4在4.18引入了新的半动态光照技术也就是Volumetric lightmap，相比于之前的 ILC机制，Volumetric lightmap能够更加细致的根据物体的空间位置对球谐函数probe做插值（ILC是每个物体做插值）



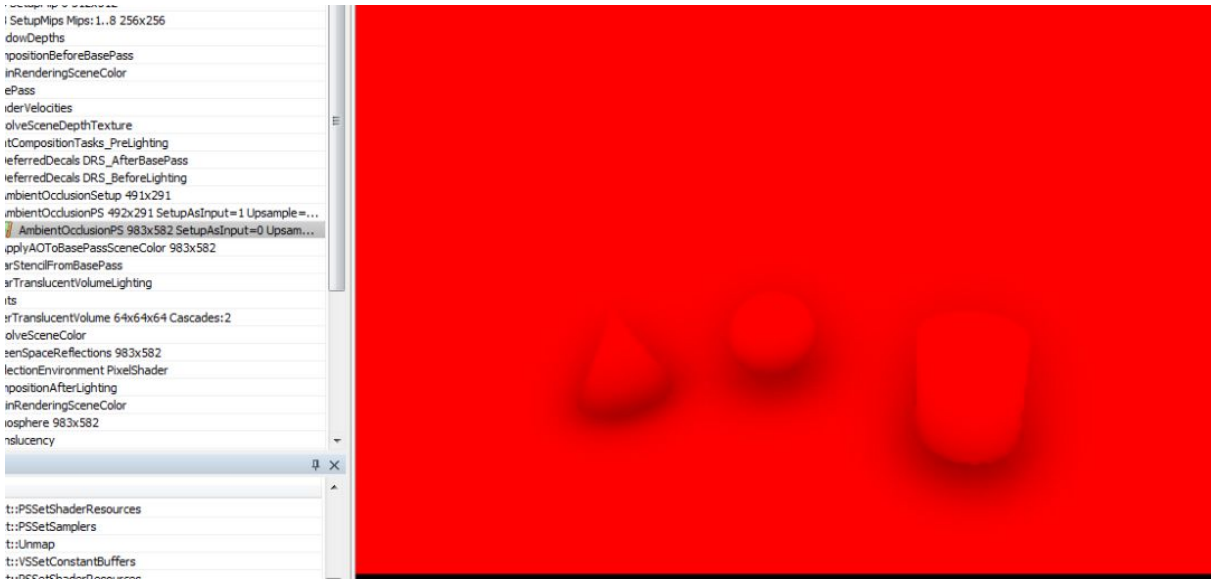
Volumetric lightmap的Texture3d

7.Velocity rendering

在basepass之后是Velocity rendering，Velocity buffer会渲染为一张R16B16UNorm，主要用于motion blur和TAA

8.Pre-Lighting

UE4在这一部分会计算DeferredDecal（屏幕空间贴花），和AmbientOcclusion， UE4的屏幕空间AO考虑了深度和Normal信息，UE4的SSAO分为两个Pass，第一个pass会计算一个四分之一分辨率的RT，使用的是四分之一分辨率的normal和depth，注意这里就用了之前生成的HZB buffer，第二个pass会渲染一个全分辨率RT并与第一个combine.注意最后计算的结果会乘到SceneColorDeferred这个RT上.

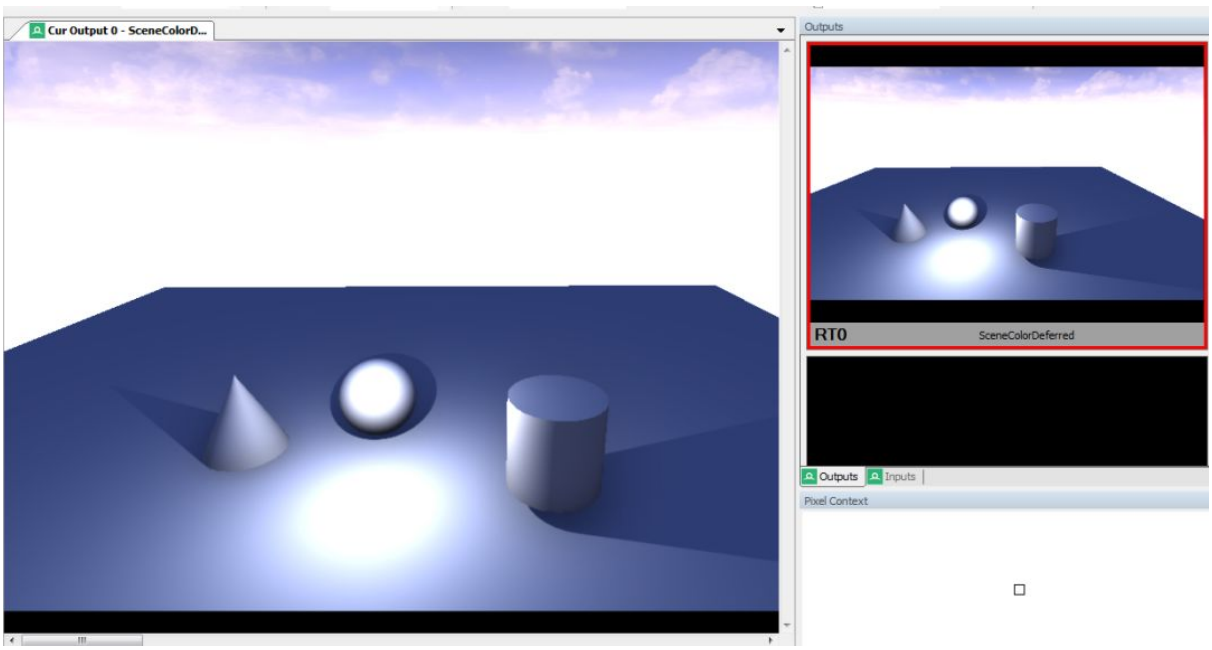


9.Lighting

接下来就是光照的渲染部分，UE4在渲染灯光光照时会先处理translucent 物体的照明。

在这之后会分别计算阴影灯光和非阴影灯光的standarddeferredlighting，

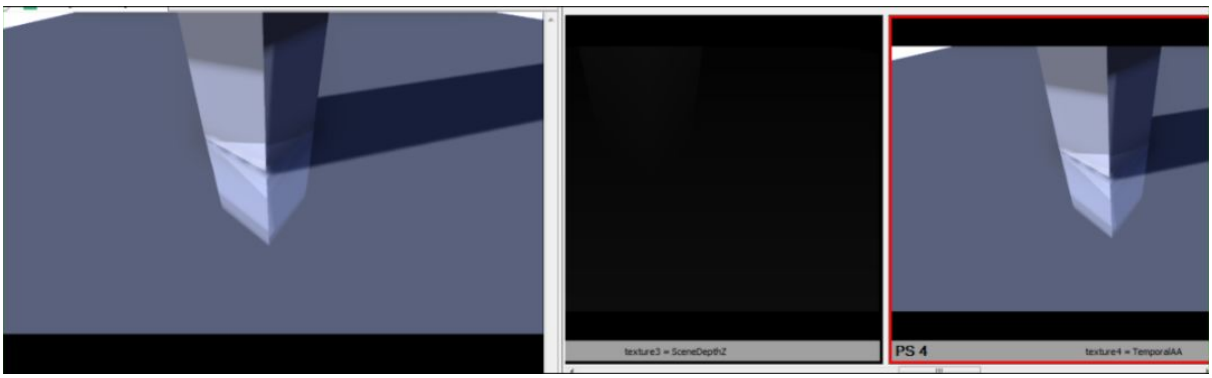
在这个pass之后，SceneColorDeferred RT就会包含最后直接光照的结果



10.ImageBased lighting

接下来UE4会渲染屏幕空间的一些光照效果，例如SSR（屏幕空间反射）还有ReflectionProbe等等

SSR会用到我们之前生成的HZB，在屏幕空间做Zbuffer的raymarching ，同时ue4的SSR会每帧jitter和TAA结合来提高质量。当击中时SSR的shader会采样上一帧的RT来获得颜色



在SSR之后是ReflectionEnvironment Pass。这一步会结合场景中的反射球和之前的SSR结果会叠加到SceneColorDeferred这个RT中。

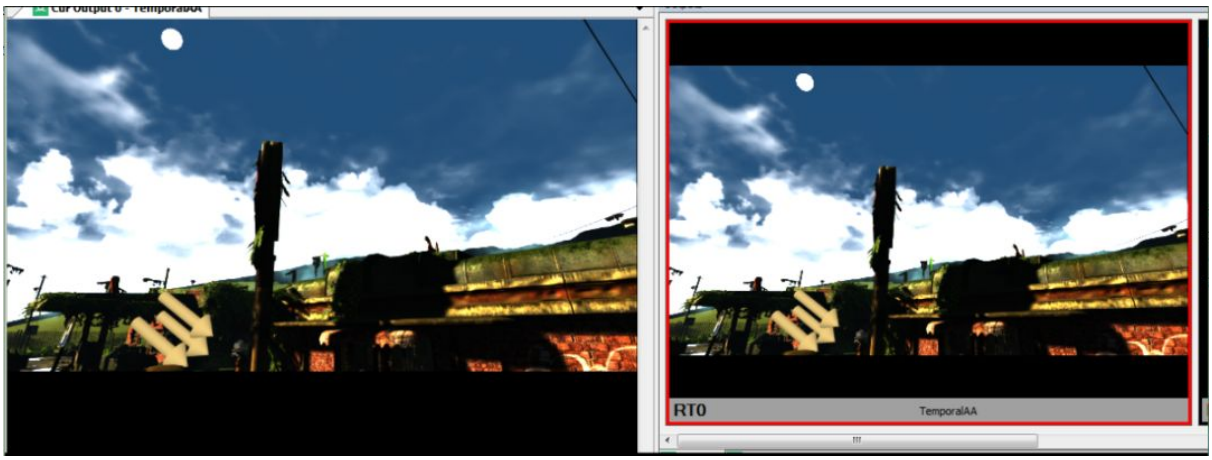
11. Post Processing

最后一步是UE4的Postprocessing，主要包括Temporal AA；Bloom；EyeAdaption等等这些可以自定义的内容。

UE4 的TAA会经历两个pass，第一个pass会处理没被stencil的像素（例如有粒子特效的时候），会用到MainRT和velocity buffer，第二个pass会处理例如粒子这样stencilled的pixel。两

个pass的区别在于混合当前RT和History buffer的blendfactor的不同，第一个pass的blendfactor会根据pixel的亮度距离等等变化，而第二个pass的blendfactor啧固定为0.25，也就是说第二个pass的像素会更多考虑当前像素，很可能这是为了减少TAA中很常见的ghosting effect

注意：TAA的处理只包含动态的光照部分，也就是纯动态光照。



以上的分析还是省略了很多的细节，例如半透照明这种还没涉及。从整体的流程分析来看，UE4在设计渲染方案的时候还是最大限度的考虑了功能的最大化，UE4的DS renderer包含了非常齐全的光照特效，包括静态的lightmap，动态的lightprobe，屏幕空间的lighting，以及一些影视级别的渲染技术，例如头发的渲染模型等等，但同时为了功能UE4的计算任务是很繁重的，因此也就不难理解为什么UE4需要Pre-Z和Occlusion Culling去剔除掉不用的像素。当然，对于使用UE4制作游戏的团队来说，根据游戏内容特点，画面的艺术风格，渲染管线都没必要一成不变，例如对于一款开放世界的野外生存游戏。我们可以考虑省掉Pre-Z的过程，或者，只用地表去画Pre-Z，又或者对于NPR画面的游戏我们完全可以不需要6-7个RT去做Deferred shading。UE4应对这

些定制化开发的需求的方法就是：开源。代码就在DeferredShadingRenderer.cpp里。

如果你也有关于虚幻引擎4 的文章或者开发经验分享也可以在微信后台与我们联系，我们会不定期挑选出优秀的技术文章与所有开发者共同分享。

阅读原文