UE4 性能 - (五) 性能分析：Render Passes(2)

小能猫吃牙膏
The only way out is through

25 人赞同了该文章

前言

关于 UE 中的 Render Pass，上一篇 UE4 性能 - (四) 性能分析：Render Passes(1) 已简单提

相关的 Pass。由于使用的示例工程可能缺乏普遍性，未必能够囊括所有类型的 Pass，我也会持续跟进，尽量补充完善。

赞同 2
分享

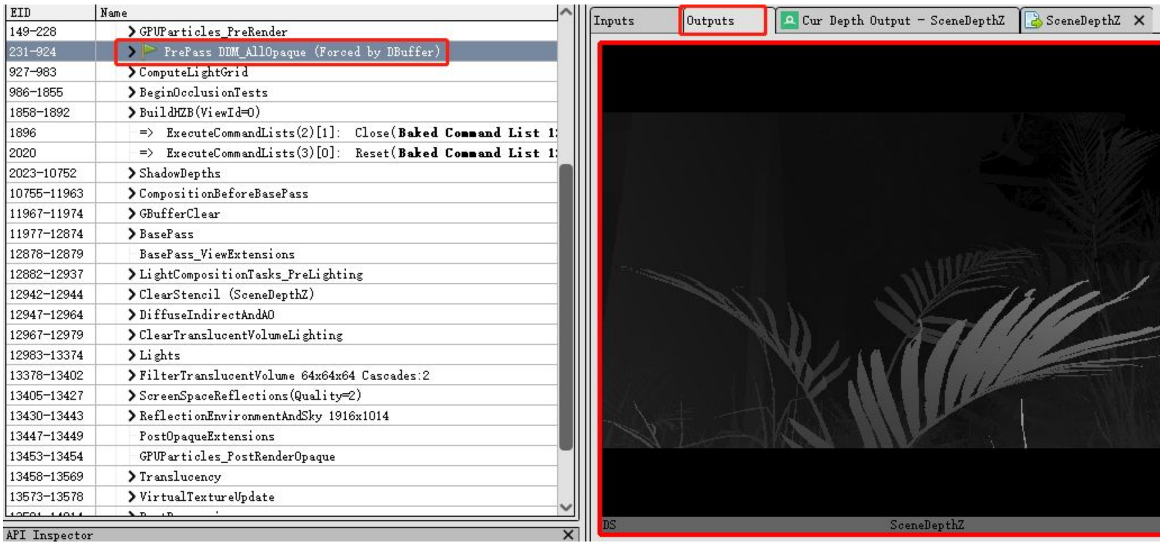
PrePass

Overview

- PrePass 有很多别名，比如 Z-PrePass, Depth Only Pass, Early-Z Pass 等，本质上是一回
- 之前介绍过 Z-Buffer 的概念，它存储着当前帧各像素的深度值，UE 中填充 Z-Buffer 的阶称作 PrePass

Notes

- 基于上一篇给出的示例，PrePass 的输出结果如下

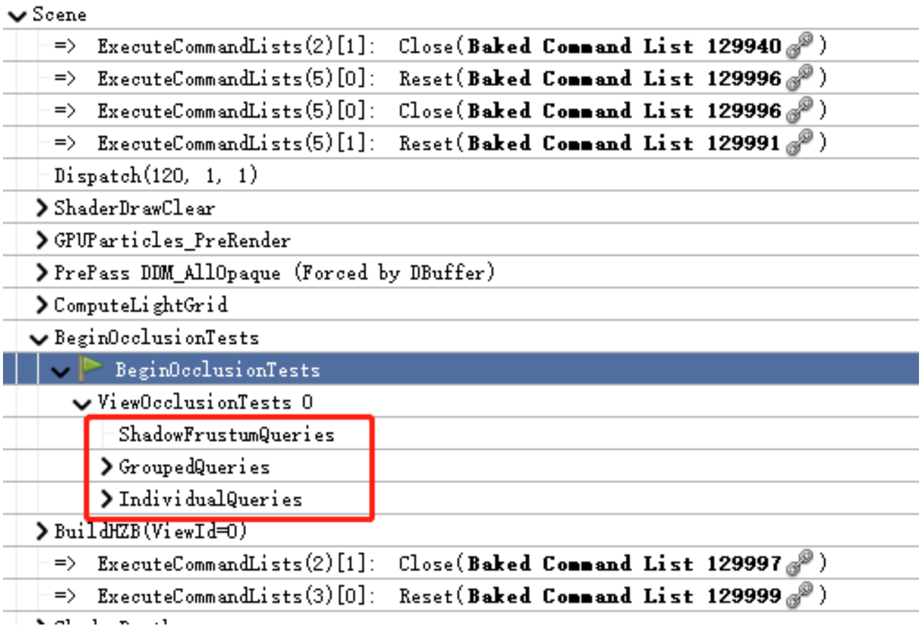


```
EID           Name
149-228       > GPUParticles_PreRender
231-924       > PrePass DDM_AllOpaque (Forced by DBuffer)
927-983       > ComputeLightGrid
986-1855      > BeginOcclusionTests
1858-1892     > BuildHZB(ViewId=0)
1896             => ExecuteCommandLists(2)[1]:  Close(Baked Command List 1:
2020             => ExecuteCommandLists(3)[0]:  Reset(Baked Command List 1:
2023-10752    > ShadowDepths
10755-11963   > CompositionBeforeBasePass
11967-11974   > GBufferClear
11977-12874   > BasePass
12878-12879     BasePass_ViewExtensions
12882-12937   > LightCompositionTasks_PreLighting
12942-12944   > ClearStencil (SceneDepthZ)
12947-12964   > DiffuseIndirectAndAO
12967-12979   > ClearTranslucentVolumeLighting
12983-13374   > Lights
13378-13402   > FilterTranslucentVolume 64x64x64 Cascades:2
13405-13427   > ScreenSpaceReflections(Quality=2)
13430-13443   > ReflectionEnvironmentAndSky 1916x1014
13447-13449     PostOpaqueExtensions
13453-13454     GPUParticles_PostRenderOpaque
13458-13569   > Translucency
13573-13578   > VirtualTextureUpdate
```

- 截图所示的灰度图即 Z-Buffer
- 结合之前介绍过的 Reverse-Z，可以看到离相机较近的部分颜色较浅(趋近于1)，较远的部

Overview

- BeginOcclusionTests 负责一帧内的遮挡性查询(Occlusion Queries)，即在运行时动态查中的物体是否被其他物体所遮挡
- UE 中支持动态遮挡性查询(Dynamic Occlusion) 的方法有多种，默认采用的是 **Hardware Occlusion Queries**
- RenderDoc 中可以看到，BeginOcclusionTests 过程一般包括几个子项：IndividualQuerie GroupedQueries 和 ShadowFrustumQueries



```
Scene
   => ExecuteCommandLists(2)[1]:  Close(Baked Command List 129940 )
   => ExecuteCommandLists(5)[0]:  Reset(Baked Command List 129996 )
   => ExecuteCommandLists(5)[0]:  Close(Baked Command List 129996 )
   => ExecuteCommandLists(5)[1]:  Reset(Baked Command List 129991 )
   Dispatch(120, 1, 1)
 > ShaderDrawClear
 > GPUParticles_PreRender
 > PrePass DDM_AllOpaque (Forced by DBuffer)
 > ComputeLightGrid
 v BeginOcclusionTests
    v BeginOcclusionTests
      v ViewOcclusionTests 0
          ShadowFrustumQueries
        > GroupedQueries
        > IndividualQueries
 > BuildHZB(ViewId=0)
   => ExecuteCommandLists(2)[1]:  Close(Baked Command List 129997 )
   => ExecuteCommandLists(3)[0]:  Reset(Baked Command List 129999 )
```

Hardware Occlusion Queries

每一帧 CPU 都会向 GPU 发送请求，查询场景中 **每个 Actor** 是否可见，GPU 会在执行完这询(此过程中会用到 Z-Buffer)的 **下一帧** 将结果回读(read back)给 CPU。其中主要包含两个风险：

- GPU 如果没能及时处理完某一次查询，CPU 则会持续等待，直到 GPU 完成查询的下一判断眼前的物体是否可见。而在相机快速移动的情况下，这样的延迟可能导致场景中某不可见的物体 “意外” 地突然出现
- 由于查询粒度是 **per-frame per-Actor**，CPU 通知 GPU 执行指令的次数过多，即 **Draw 数量太大**，会带来显著的性能开销

IndividualQueries & GroupedQueries

为了尽可能降低 Draw Call 的数量，UE 采取了一种类似于 “合批” 的机制控制查询的粒度：

- 细粒度：首先还是基于 **per-frame per-Actor** 的粒度进行查询，如果结果是此帧此物体 visible，表明此物体接下来需要被绘制，这种情况下的查询就称作 **IndividualQueries**
- 粗粒度：而如果结果是 invisible，不需要被绘制，就把这个 Actor 添加到一个 Group，针对这个 Group 进行查询(粒度变大)；直到整个 Group 都可见，再把它打散清零，粒度

Overview

- **HZB = Hierarchical Z-Buffer**，实质上就是带 Mipmap 的 Z-Buffer
- BuildHZB 作为一个 Render Pass，就是基于 PrePass 输出的 Z-Buffer，生成它的一系列 Mipmap
- 如图所示，示例中的 BuildHZB 分三次生成了 0-9 共 10 级 Mipmap

Notes

- HZB 主要用于另一种叫作 **Hierarchical Z-Buffer Occlusion** 的动态剔除(Dynamic Occlu
  方法，以及 **SSR**(ScreenSpaceReflections) 和 **SSAO**(ScreenSpaceAmbientOcclusion)
- Hierarchical Z-Buffer Occlusion 的工作原理与上面提到的 Hardware Occlusion Queries
  似，但在对每个 Actor 做可见性查询时，会先根据 Actor 的包围盒大小选择适当的 Mipm
  再从这张 Mipmap 中进行采样，得到查询结果。这种方法相对于 Hardware Occlusion Qu
  更加保守，**查询结果精确度更低，实际剔除的物体会更少**；但好处是**减少了对纹理的采**
  数，降低了读写带宽的消耗
- BuildHZB 只负责生成 Z-Buffer 的各级 Mipmap，直到分辨率的长或宽为 1 为止，因此它
  销主要取决于 Z-Buffer 本身的大小，即渲染分辨率的大小

ParticleSimulation

Overview

- 计算 **GPU Sprites** 类型的粒子的运动情况，并记录到两张纹理上
- ParticleSimulation 是在 Unreal 渲染一帧的过程中 **最先执行** 的 Pass
- 如图，该 Pass 的输出是两张纹理，分别记录着相应粒子的世界坐标和运动速度

Overview

- 针对由 GPU 驱动的、处于运动状态的物体，保存在当前帧物体各顶点的运动速度
- 主要用于 Motion Blur 和 TAA(Temporal Anti-Aliasing)
- 性能开销主要源于 **运动物体的数量，及其这些物体的多边形(polygon)数量**

References

Unreal's Rendering Passes

How Unreal Renders a Frame

Visibility and Occlusion Culling

UE4 性能 - (四) 性能分析：Render Passes(1)

编辑于 2022-04-11 10:37

虚幻4引擎 渲染 性能分析

写下你的评论...

个叫 Early Z-pass 的属性，其实针对的也就是 Prepass，所以这俩在 UE 里是一回
Unity 里似乎叫作 Depth Only Pass。这同一个概念在不同的地方可能以不同的名称
现，具有一定迷惑性，所以特意指出了

2022-01-24

TaikonAut

持续关注大佬分享

2022-01-16

小能猫吃牙膏 作者

谢谢关注

2022-01-18

Michael-米歇尔

大佬厉害

2022-01-12

小能猫吃牙膏 作者

谢谢支持 共同进步

2022-01-12

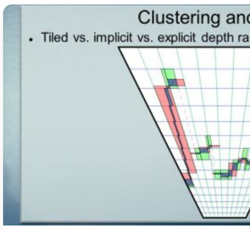
推荐阅读

UE4 性能 - (四) 性能分析：Render Passes(1)

小能猫吃牙膏

Houdini 结合 ue4 概念场景设计（二）UE4篇

蓝墟



**UE4 Forward+ Cl
Render计算分析**

will ...　　发表