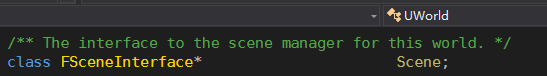
UE4分析

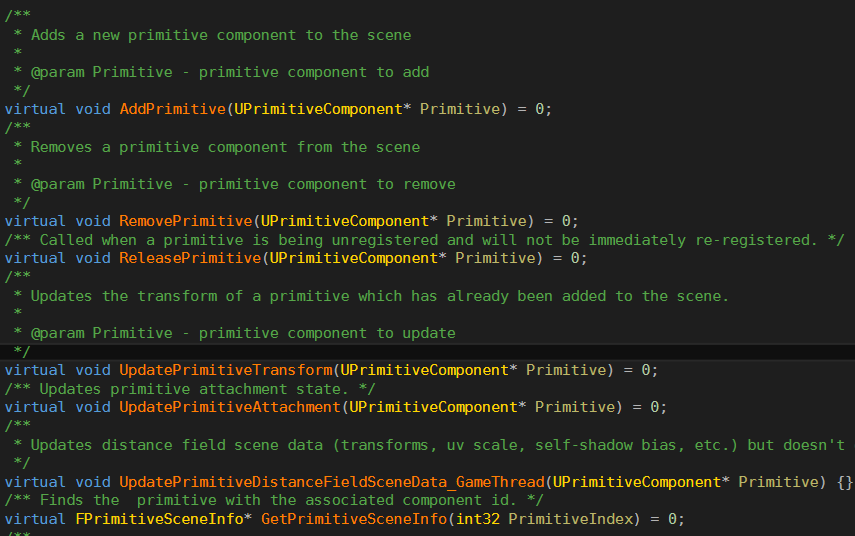
# 渲染

位于引擎中Engine\Source\Runtime\Engine\Public\SceneInterface.h的FSceneInterface接口类是一个UWorld对应的渲染接口：

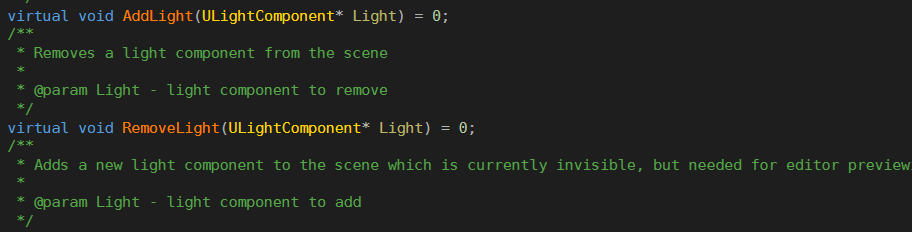


它主要有以下接口函数：

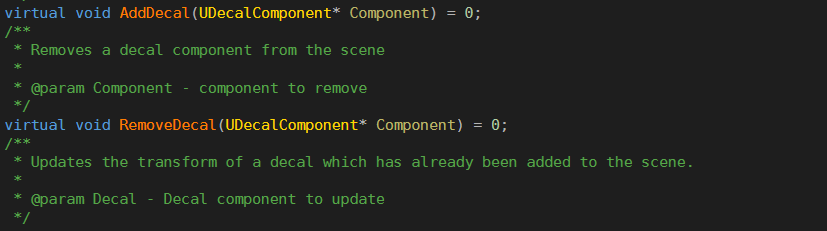
-相关Primitive组件：



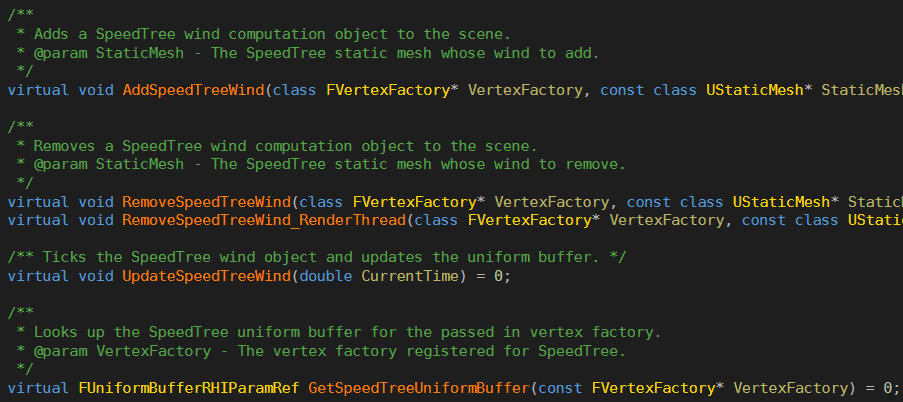
-相关Light组件：



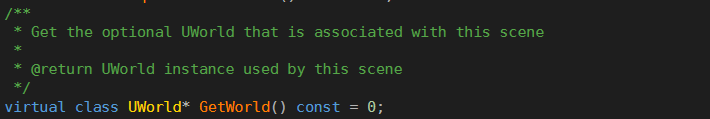
-相关Decal组件：



-等等其他的可渲染组件，其中也有SpeedTree的存在(SpeedTree是一款树的建模软件，有专门的UE4版本)

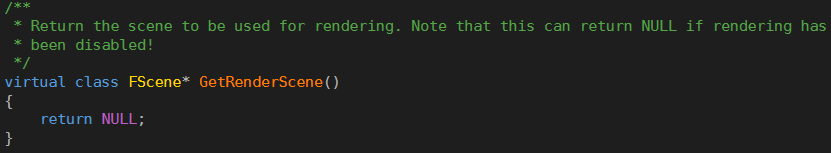


这个类是和UWorld相关联的

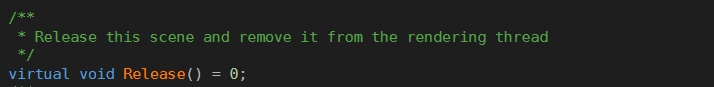




并且有相应的Scene



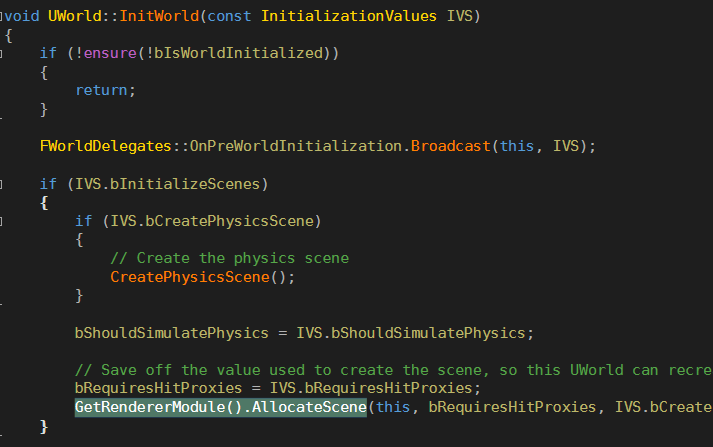
当然也和渲染线程相关（~只从接口类暂时不清楚如何关联~）



这个FSceneInterface接口类是一个实现了FSceneInterface的FScene的接口，而FScene通过GetRendererModule().AllocateScene构建；

其中找到的一个调用是在World.cpp中的InitWorld()初始化World时进行的：

从具体的实现中能够发现这个世界渲染与否和World的初始化参数中的bInitializeScenes参数相关。而Scene这个单词可以理解为是World在渲染模块中的概念

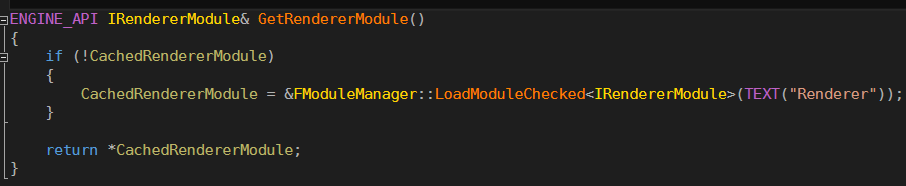


其中这个GetRendererModule()是UE4中的“ENGINE\_API”

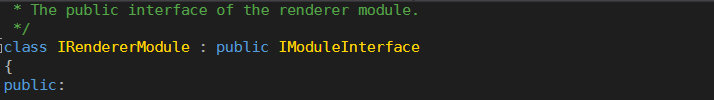
可能是UE4提供的引擎API

它是位于Engine\Source\Runtime\Engine\Public\EngineGlobals.h中

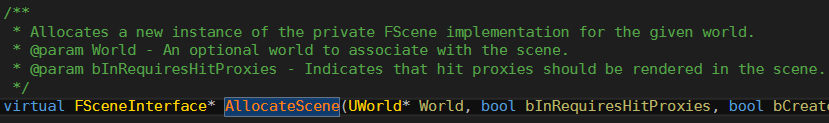
其功能一目了然：返回或加载返回渲染模块<IRendererModule>



IRendererModule是声明在Engine\Source\Runtime\RenderCore\Public\RendererInterface.h中的，继承自IModuleInterface（所有游戏引擎用到的模块都应当继承于此，它只有一些加载，卸载，等等的基函数）（注意: IRendererModule依然是一个接口类）



AllocateScene的定义



到此一个渲染模块已经存在，并且通过渲染模块的AllocateScene分配了FScene的实例到UWorld上，也就是 < FSceneInterface \*>Scene，从World中的RecreateScene()中可以看到World中的< FSceneInterface \*>Scene与渲染模块IRendererModule是紧密相关的，而且RecreateScene的操作只可能在AllocateScene中进行，因此推断AllocateScene生成了相关这个World的FScene对象，而且为这个World中的成员变量< FSceneInterface \*>Scene赋予了初始值即生成的FScene对象



这里小结一下：

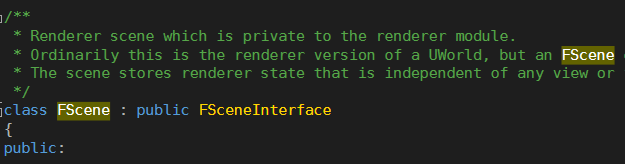
UWorld中存在一个成员变量<FSceneInterface\*>Scene【UWorld中主要UWorld::CreateFXSystem(),

UWorld::UpdateLevelStreamingInner(),

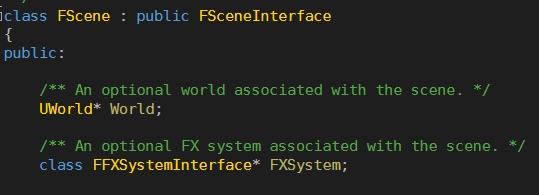
UWorld::UpdateParameterCollectionInstances()存在相关的调用】

而在UWorld的InitWorld()中，通过调用GetRendererModule().AllocateScene()生成了相关这个UWorld的FScene对象，且为这个UWorld中的成员变量< FSceneInterface \*>Scene赋予了这个初始值【注意：FScene是继承于FSceneInterface的，所以UWorld中的FSceneInterface\*变量命名是Scene】

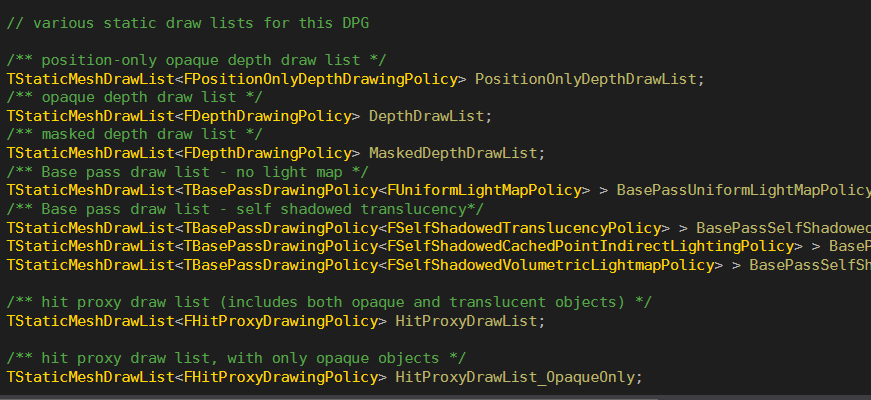
FScene声明于Engine\Source\Runtime\Renderer\Private\ScenePrivate.h中，继承自FSceneInterface



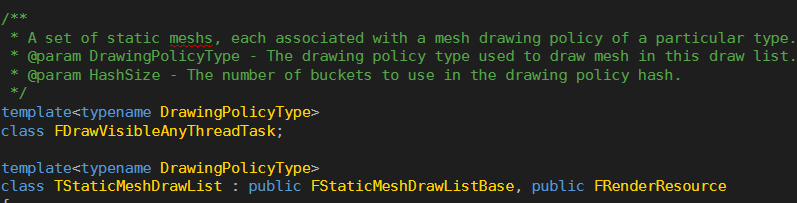
FScene中有当前相关世界的索引和FXSystem接口（FX在UE4中指一些特效）



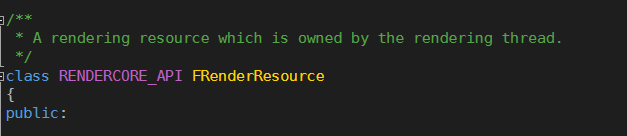
最重要的，它有各种DrawList，概念类似于渲染队列？



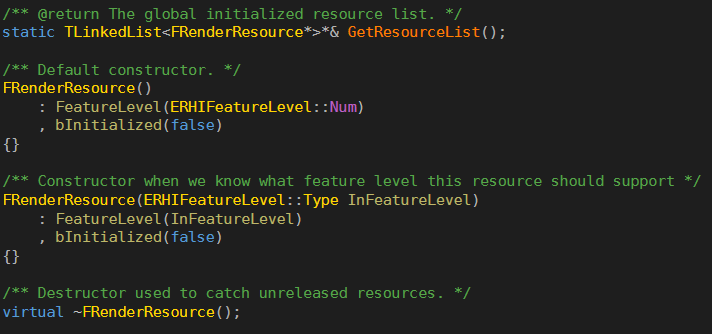
其注释如下：

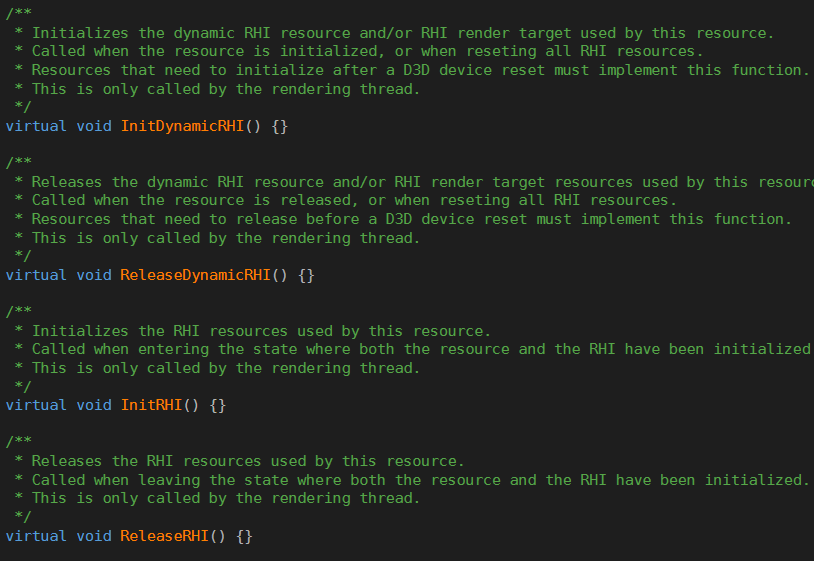


其中RenderResource位于Engine\Source\Runtime\RenderCore\Public\RenderResource.h，对应的注释如下：

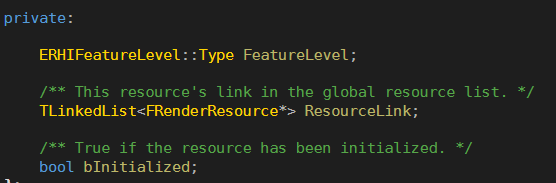


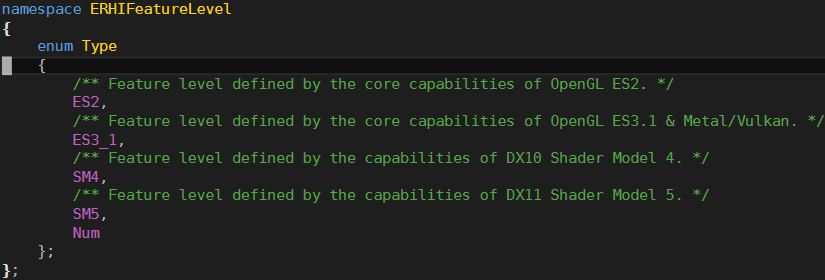
相关的函数：





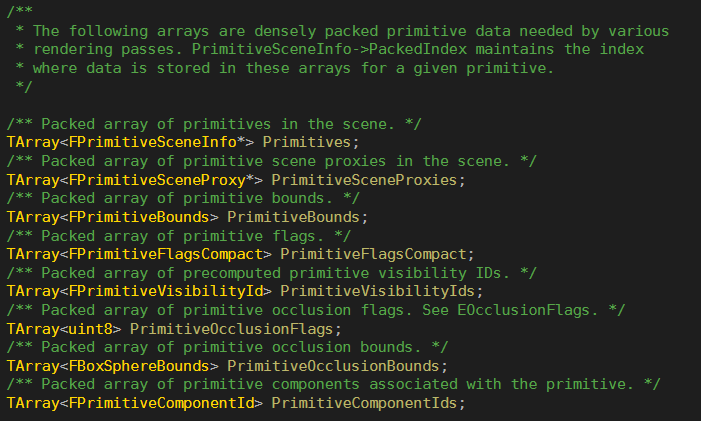
成员变量：



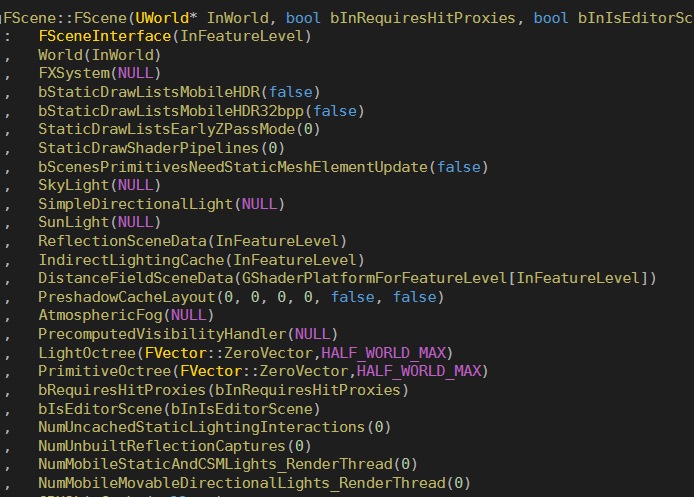


RHI: Render hardware interface 渲染硬件层接口，是虚幻4建立的一套与底层图形硬件平台无关的图形渲染API.

如注释所说的：FScene中存放有进行渲染需要的数据



FScene的构造函数，有一堆的初始化列表项



关键的过程：

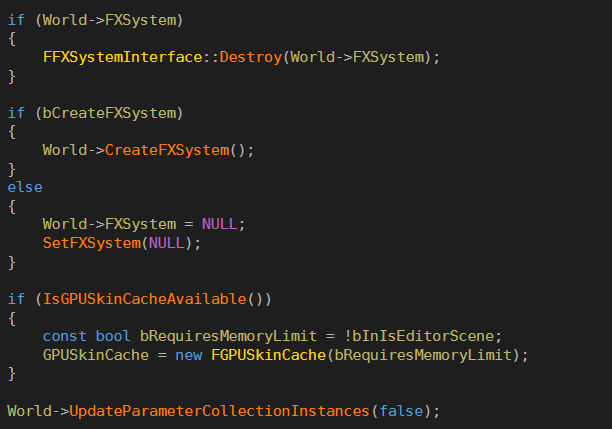
1.check World成员（初始化列表中初始过）并设置World中的接口引用自身



2.FScene中的FeatureLevel的赋值

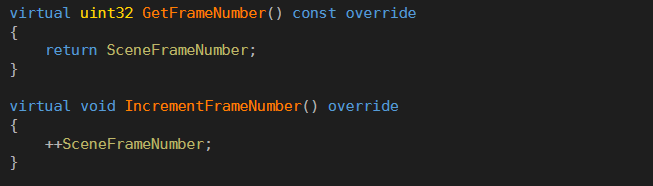


3.对World的设置（FXSystem相关等）



……

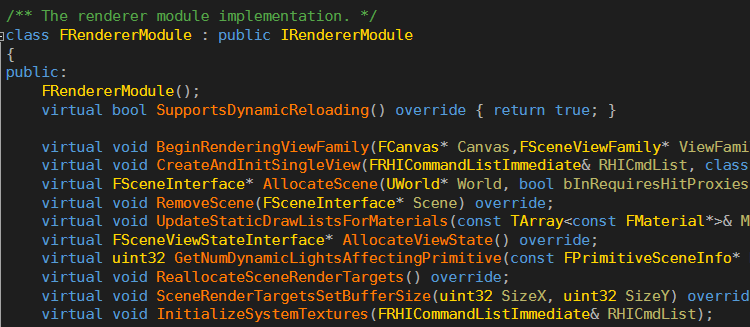
FScene中的Frame，通过这个可以大概知道FScene是一个什么样的地方，即FScene是UWorld在渲染中的表现，之前也说了FScene中有渲染所需的各种信息集，例如：TArray<FPrimitiveSceneProxy\*> PrimitiveSceneProxies; 而PrimitiveSceneProxies也是UPrimitiveComponent组件在渲染中的表现，和GamePlay游戏框架有相似之处



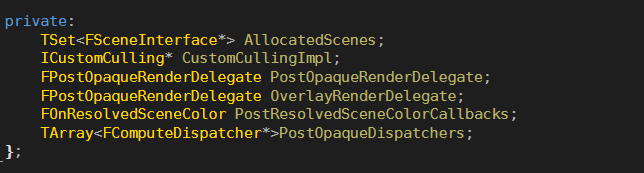


FRendererModule声明于Engine\Source\Runtime\Renderer\Private\RendererModule.h

继承于IRendererModule接口

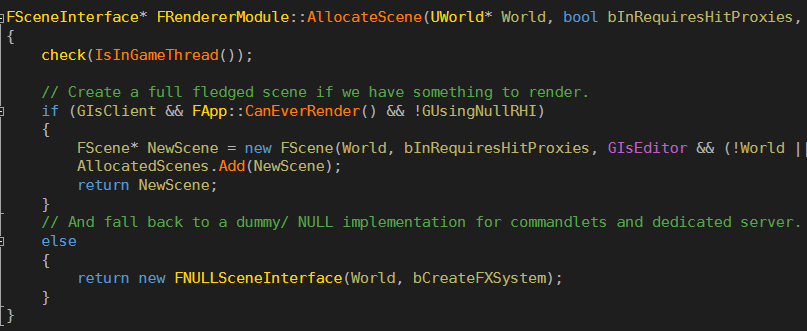


关键的成员变量：

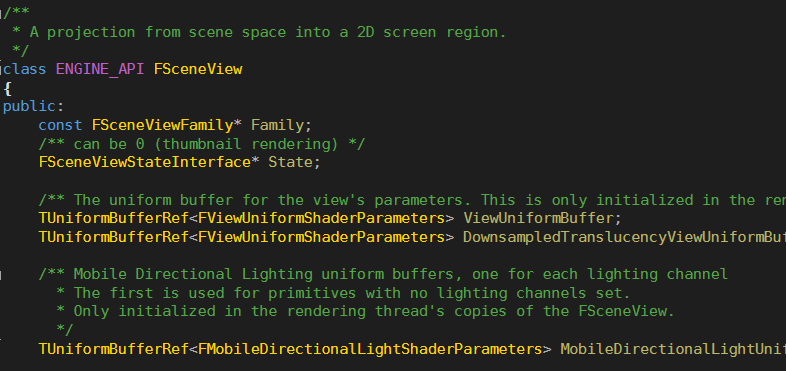


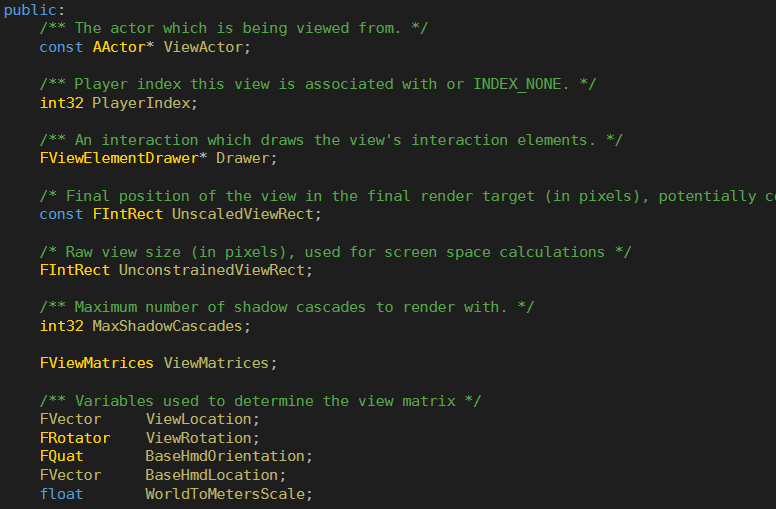
AllocateScene的实现：

FScene与UWorld之间的过程主要由FScene的构造处理，AllocateScene中主要过程是FScene与自身的关系：AllocatedScenes.Add(NewScene);因此可以看出FRendererModule是相当于一个管理器（管理FScene）的存在，（~但独立的游戏只有一个World，只有一个FScene，那么这个FRendererModule中只有一个元素~）



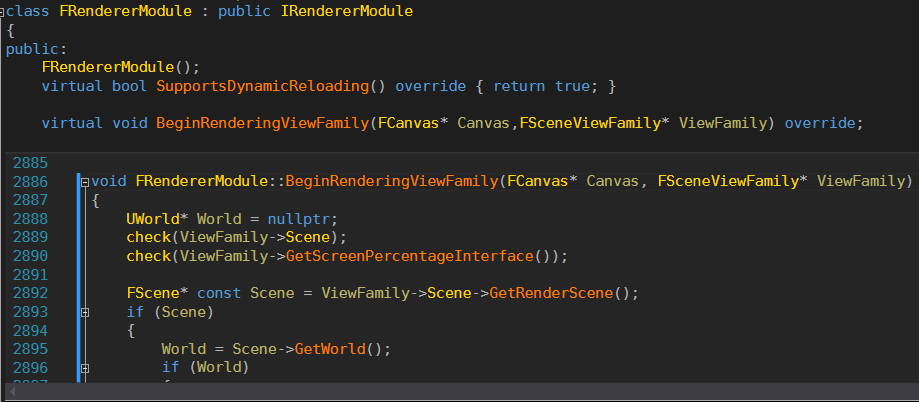
SceneView位于Engine\Source\Runtime\Engine\Public\SceneView.h

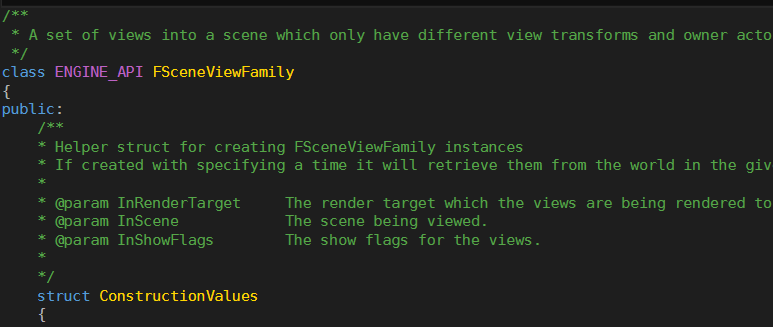




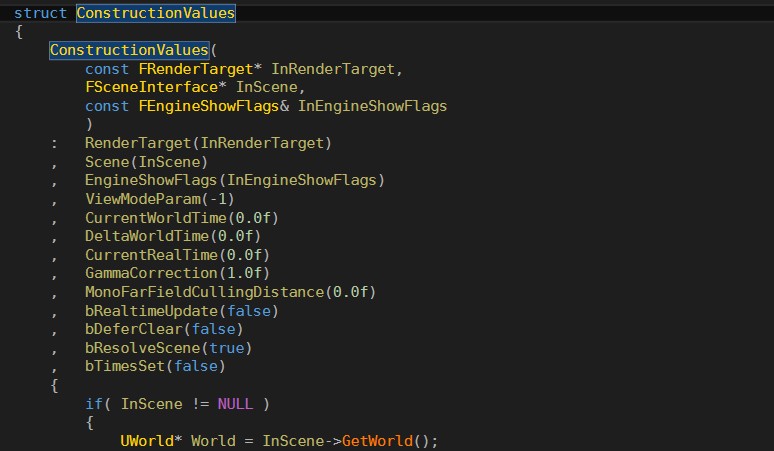
FSceneViewFamily也位于Engine\Source\Runtime\Engine\Public\SceneView.h

它在FRendererModule的BeginRenderingViewFamily()中作为参数被调用，本身没有存储相关的引用

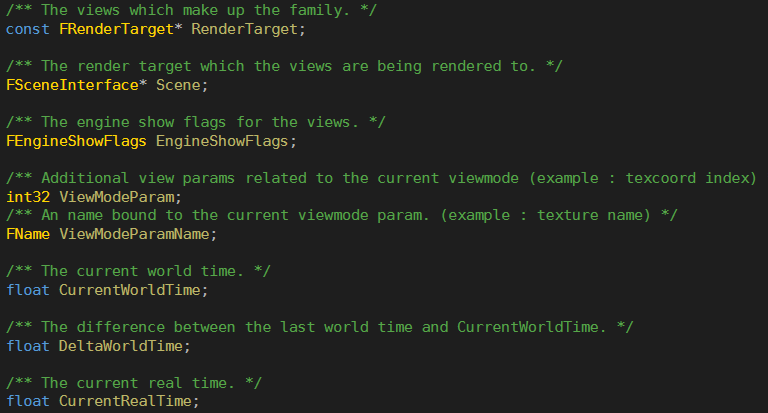




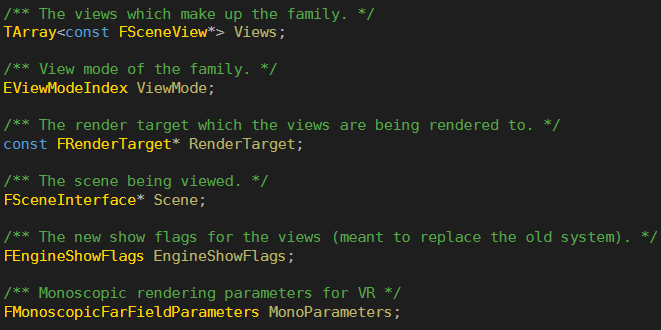
结构体ConstructionValues：



ConstructionValues中存在有以下成员：

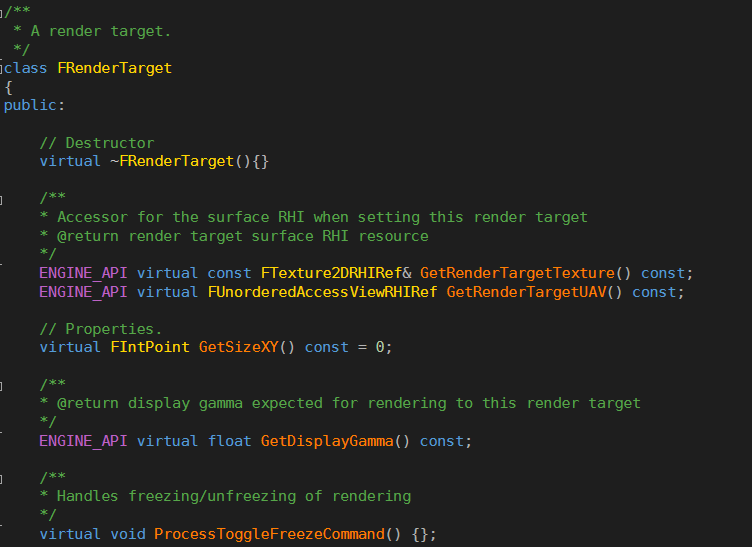


FSceneViewFamily也是由FSceneView构成的，以下是FSceneViewFamily的部分成员变量



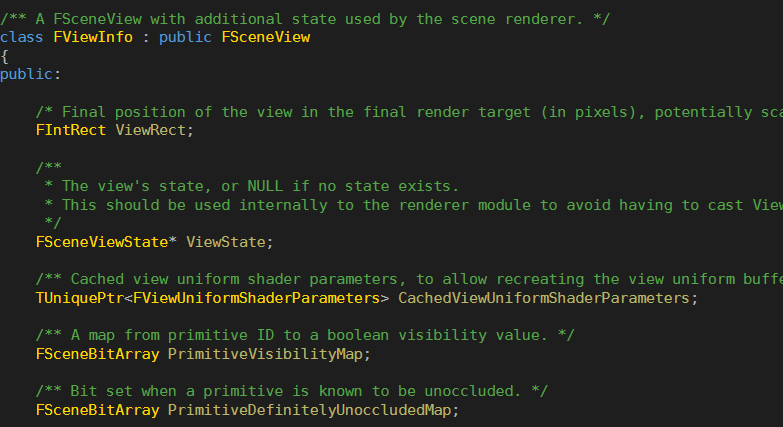
FRenderTarget位于Engine\Source\Runtime\Engine\Public\UnrealClient.h

Render Target在引擎中是一个对象，可用于做实时贴图，位于材质&贴图->创建Render Target对象，同时从所属模块来看也不是渲染实现相关的类

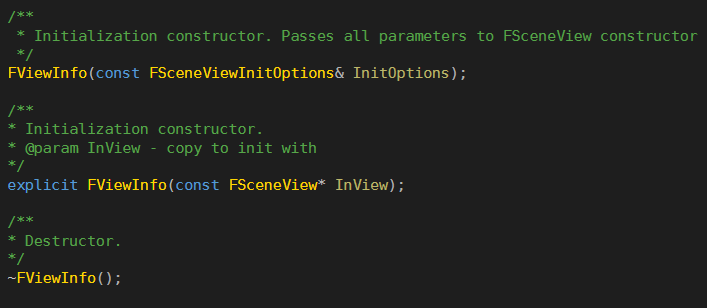


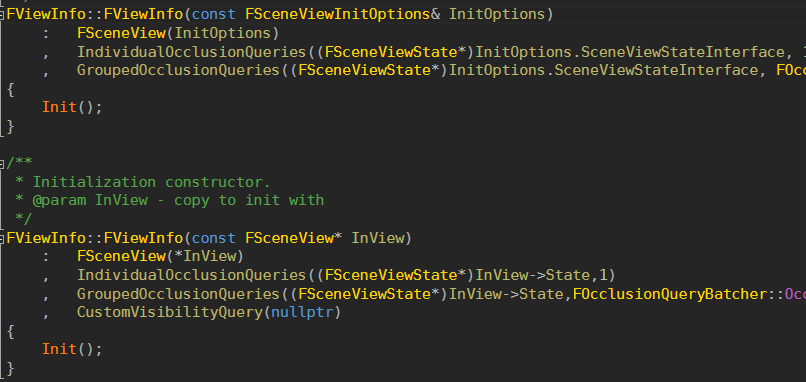
FViewInfo位于Engine\Source\Runtime\Renderer\Private\SceneRendering.h

注释中说FViewInfo是FSceneView和一些其他的状态信息组成的类，而且也是继承于FSceneView的，而FSceneView是Scene到2D的投影，但与其不同的是FViewInfo位于渲染模块下

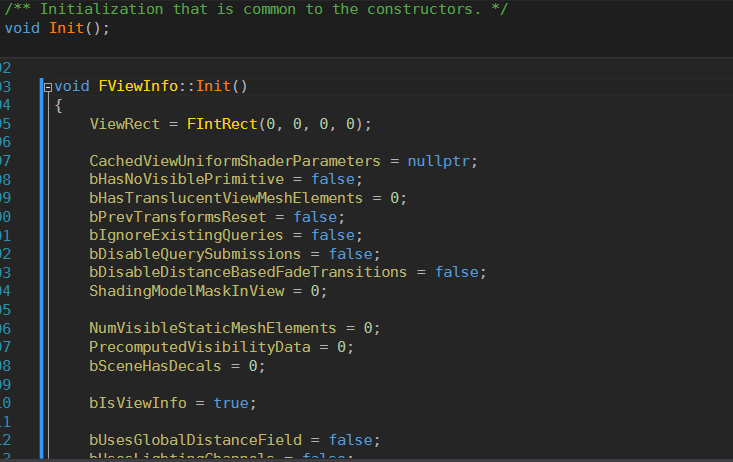


FViewInfo的构造函数：

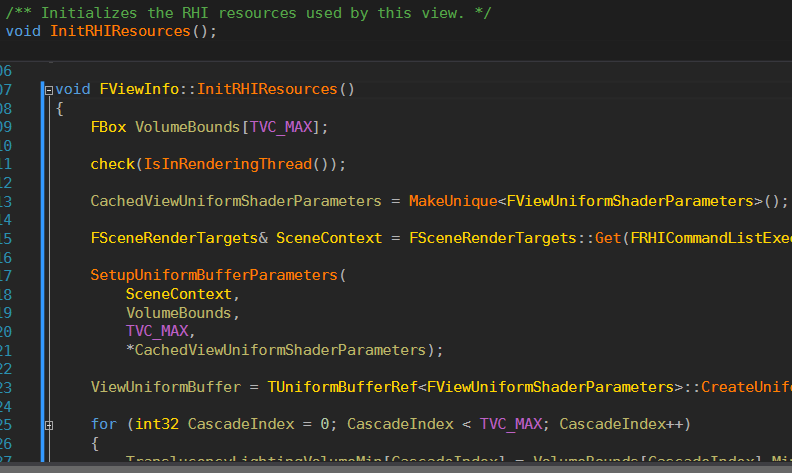




FViewInfo的Init()函数：进行大量的变量初始化

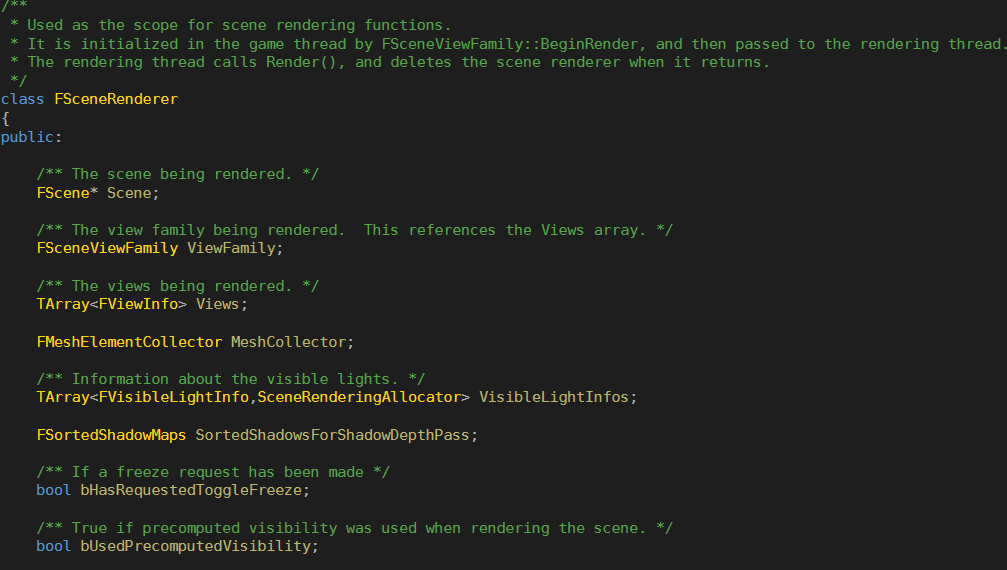


FviewInfo的其他函数

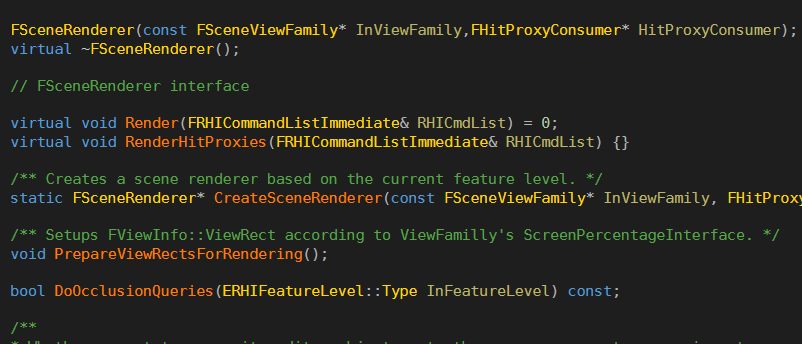


…

FSceneRenderer位于Engine\Source\Runtime\Renderer\Private\SceneRendering.h

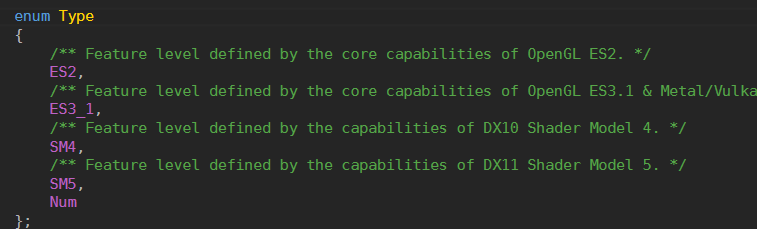


FSceneRenderer的重要函数：



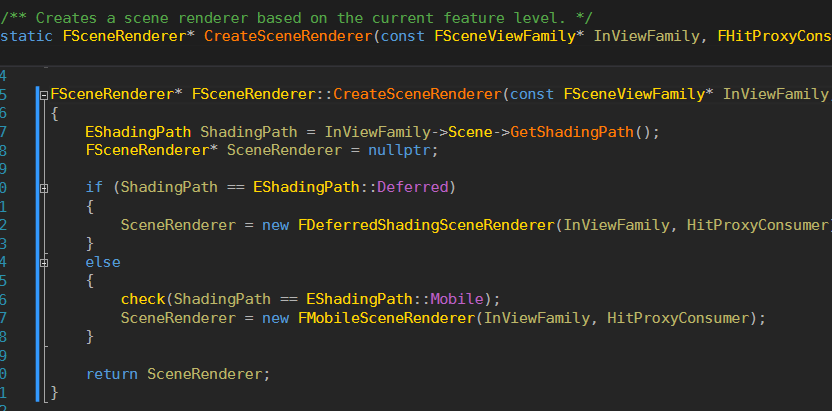
注意FeatureLevel是一个枚举类型：



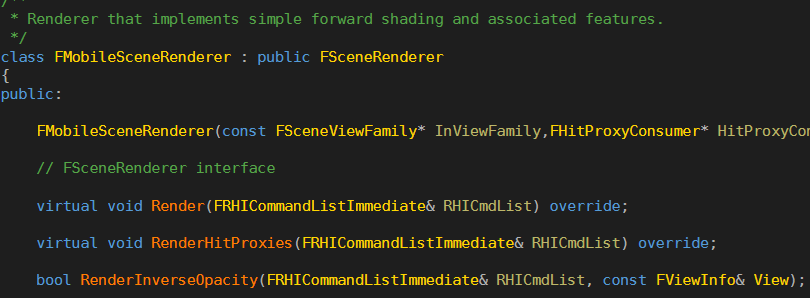


FSceneRenderer:: CreateSceneRenderer ():

根据ShadingPath的不同，选择使用延迟渲染和面向移动端渲染（前向渲染）

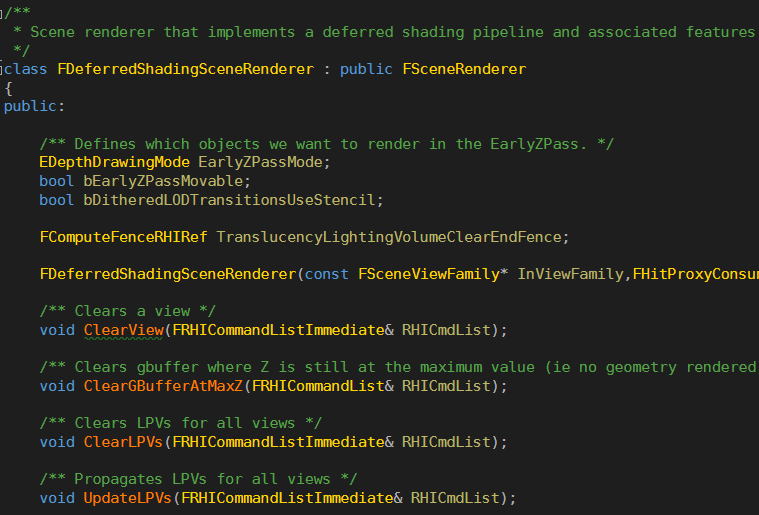


FMobileSceneRenderer位于Engine\Source\Runtime\Renderer\Private\SceneRendering.h（与FSceneRenderer同一下目录下）



FDeferredShadingSceneRenderer位于

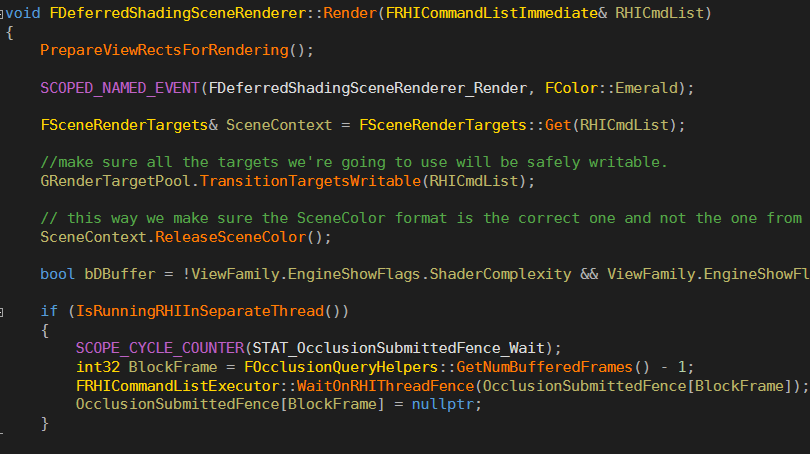
Engine\Source\Runtime\Renderer\Private\DeferredShadingRenderer.h下



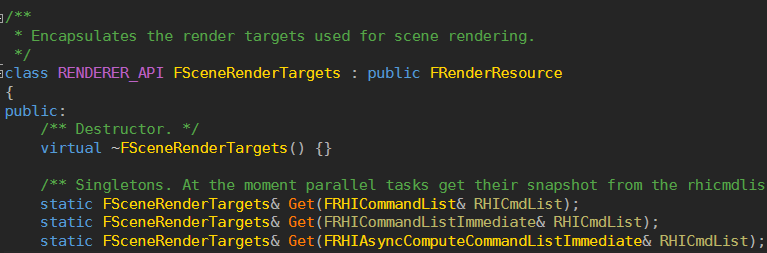
主要函数：

FDeferredShadingSceneRenderer::Render(FRHICommandListImmediate& RHICmdList)

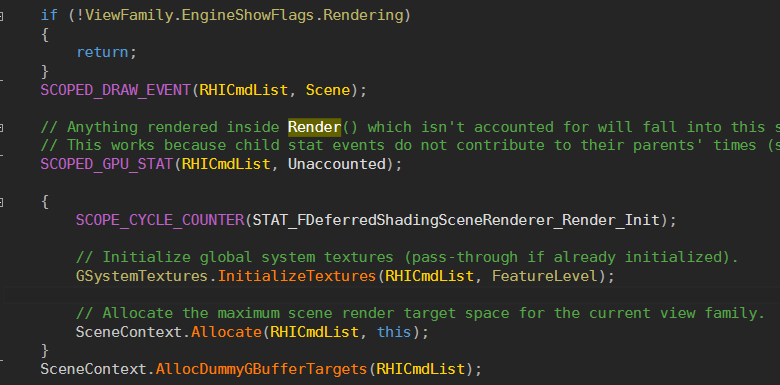
1.首先调用了PrepareViewRectsForRendering()，相关游戏的显示窗的百分比，接着命名事件，创建FSceneRenderTargets……



其中的FSceneRenderTargets的定义如下

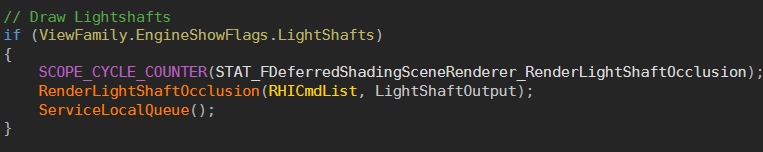


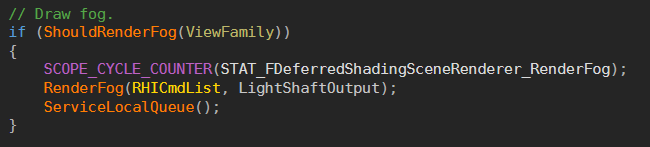
2根据ViewFamily的标记位决定是否继续渲染



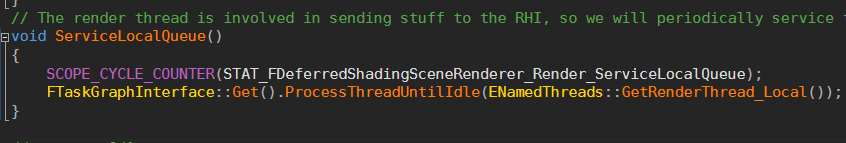
3后面实现根据各种标记位判断是否渲染相关（比如：Light，）……

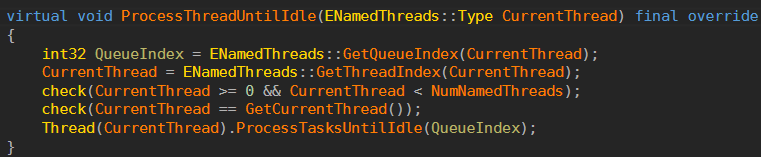
比如Lightshafts，fog，这些渲染实现相似，直接先调用相应的本类函数接着调用SeriveLocalQueue()

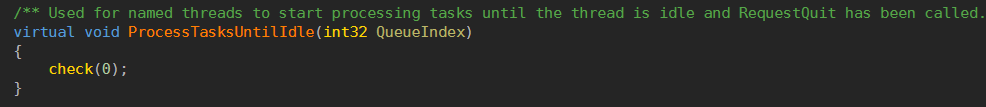




ServiceLocalQueue()实现如下：







……

FPrimitiveSceneProxy:

FPrimitiveSceneInfo:

小结：从FSceneRenderer中理解ViewFamily是一个相机(GamePlay框架中)的List，而Scene是这些相机所属的World(GamePlay框架中), Views则是所有FSceneView的对象再加上额外的渲染信息，两者所属模块不一样，前者所属Renderer，后者所属Engine，FSceneRenderer中的Render()默认UE4使用延迟渲染，否则使用前向渲染（移动端）。两种FDeferredShadingSceneRenderer和FMobileSceneRenderer都继承于FSceneRenderer用于两种渲染方式，并重写相应的Render()实现

官方文档中如下介绍：

**FSceneView** : 单个视图到一个 FScene 的引擎代表。视图可以通过对 FSceneRenderer::Render 的不同调用的不同视图来渲染（多编辑器视口）或通过对 FSceneRenderer::Render 的同一调用中的多个视图来渲染（分屏游戏）。为每个帧构建新视图

**FViewInfo** : 渲染线程中描述View信息的对象，可以看成是view在渲染线程中的副本

**FSceneViewState** ： ViewState 存储有关在多个帧中需要的某个视图的私有渲染信息。在游戏中，每个 ULocalPlayer 都有一个视图状态

**FSceneRenderer** ： 每个帧都会被创建的类，用于封装跨帧的临时对象

# 物理