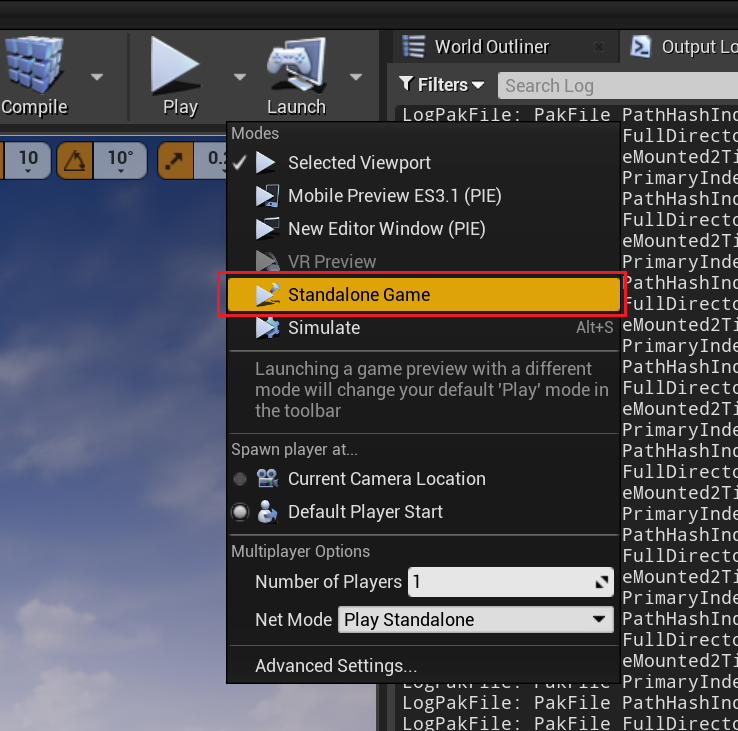
Demo 解释：

在 /Game 目录下有两个Map, 分别为MainMap 和 ChlidLevel，游戏在MainMap，

StandAlone 模式下启动。

游戏启动会创建一个初始UI（WidgetBlueprint'/Game/UMGTest/UMG\_Panel.UMG\_Panel'），里面的按钮主要负责创建UI（UMG\_Main<嵌套了UMG> 和 UMG\_Alone<未嵌套UMG>）和切换关卡。

正确表现：在MainMap下创建的的 UMG 和 Actor 在切换场景后，主动调用GC，会被GC掉。

验证方式：在standAlone窗口下，使用 MemReport full指令查看内存对象，前一关卡下所创建的UMG均被GC释放。

错误表现：

切换关卡后，用MemReport full指令查看，前一关卡所创建的对象均存在。

在

puerts\_unreal\_demo\Saved\Profiling

文件夹下查看MemReport的报告文件。

如果发生内存泄漏，可以通过 obj refs name= ClassName(UMG\_Panel\_C) shortest，查看对象引用链。Preview模式下结果输出在output log 窗口下，stanalone模式下，输出在Saved\Logs目录下的日志文件中

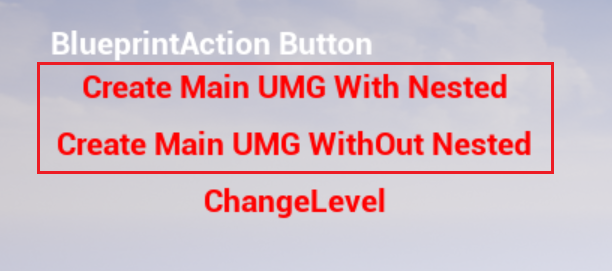
-------------------------------------------------------------------------------

UMGNestedTest.ts是demo启动文件

操作步骤一共分三块来验证内存泄漏：

Part1: 使用蓝图创建UMG

🡺

点击两个按钮创建创建UI，

🡺

点击ChangeLevel切换关卡

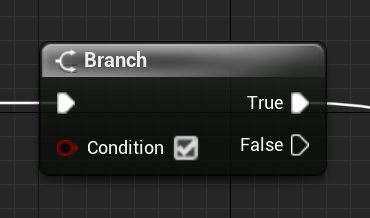
🡺

切换关卡后点击Garbage Collect按钮，

🡺

使用MemReport full查看内存，发现上一关卡创建的UMG已经被回收。

Part2: 使用TS创建UMG。

打开MainMap 的 Level Blueprint, ，将condition改为false.

🡺

取消UMGNestedTest.ts 文件中的代码注释，在TS代码中创建UMG

🡺

与part1相同的步骤，不过按钮点击选用

使用MemReport full查看内存，发现上一关卡创建的UMG未被回收，

但是ts内创建的Actor被回收了

使用obj refs name= UMG\_Panel\_C查看，

有引用链：

[2023.02.02-08.53.42:724][537]LogReferenceChain: (root) GCObjectReferencer /Engine/Transient.GCObjectReferencer\_0::AddReferencedObjects(): Puerts\_UserObjectRetainer

[2023.02.02-08.53.42:724][537]LogReferenceChain:  WidgetBlueprintGeneratedClass /Game/UMGTest/UMG\_Panel.UMG\_Panel\_C

[2023.02.02-08.53.42:724][537]LogReferenceChain:

[2023.02.02-08.53.42:724][537]LogReferenceChain: (root) GCObjectReferencer /Engine/Transient.GCObjectReferencer\_0::AddReferencedObjects(): Puerts\_UserObjectRetainer

[2023.02.02-08.53.42:724][537]LogReferenceChain:  UMG\_Panel\_C /Engine/Transient.GameEngine\_0:TsGameInstance\_0.UMG\_Panel\_C\_0->Class

[2023.02.02-08.53.42:724][537]LogReferenceChain:   WidgetBlueprintGeneratedClass /Game/UMGTest/UMG\_Panel.UMG\_Panel\_C

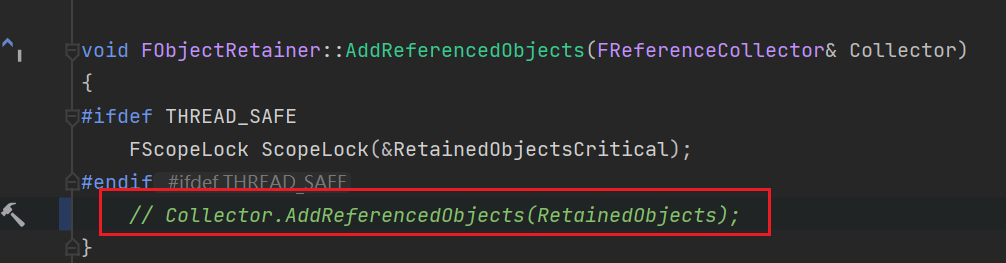
[2023.02.02-08.53.42:724][537]LogReferenceChain:

发现，puerts 中ObjectRetainer.cpp 文件，函数

void FObjectRetainer::AddReferencedObjects(FReferenceCollector& Collector) 中的 Collector.AddReferencedObjects(RetainedObjects); 保存了引用，未被释放。

Part3: 自己的简单修复(不正确，该项目可以正常跑，但会导致其他项目crash)

注释掉上述函数的此行，并进行与part2相同的操作步骤，

，发现前一关卡创建的UMG被成功GC。

在demo中注释上一行能跑，但是我在项目中注释这一行会导致crash，所以因该不是修改这里。

总结：个人认为，UMG被释放的时候，Collector中未被释放，保留了引用，导致无法被GC。