Trigger 触发条件

**刘连友**

**2021/8/30**

1. **概述**

Trigger条件是触发器普遍使用的一个非常有效的工具，和Trigger Task组合使用可以实现非常具有自定义的个性化的内容。

Trigger条件的使用能够让代码的重复使用大大的提升，能够让内容的生产更加规范化，流程化，通用化。在大数据，多任务，独特性有着非常有力的支撑。

所谓的Trigger条件， 和我们一般上理解的条件一样，都是对一个特定事件的触发的描述。它和Trigger Task是两个系统。Trigger条件是用来修饰Trigger Task的，它是可有可无的，没有条件，Trigger Task照样是可以执行下去的，而条件只是说对Trigger Task的执行时机做了一下限定。

就比如我们吃饭，这个就可能需要非常多的条件，第一个是要朋友，第二个是要有时间，第三个是要在宇宙第一饭店吃，等等条件，这些条件都满足了，那我们就可以好好吃饭了的。在这里条件是（要有朋友，要有时间，在宇宙第一饭店），Trigger Task是吃饭。吃饭是固定的，就算没有朋友，没有时间，没有宇宙第一饭店，我们也能吃。至于要怎么吃，吃什么，吃完后要做什么这些都是在吃饭这个Trigger Task里面定义的。

我把条件和Trigger Task分开定义使用的基本想法也是源于上面，吃饭这个任务大多数都是相同的，而条件会千变万化，不能统一而论，因此为了让使用简单，维护简单，能够让我们的场景事件编辑更加流程化，我就将触发条件单独分裂出来

在上面这个例子的基础上我对Trigger条件的使用范围做了一些扩充，因为上面示例只是用来描述一个Trigger Task的开始条件，或者说前置条件**StartCondition**，而在我们的Trigger Event System里面有很多的状态，这些状态的自动切换就又可以用各种条件来进行修饰了，比如有

**FinishCondition,**

**SkipCondition,**

**ResetCondition,**

**PauseCondition,**

**ResumeConditon,**

**RepeatCondition,**

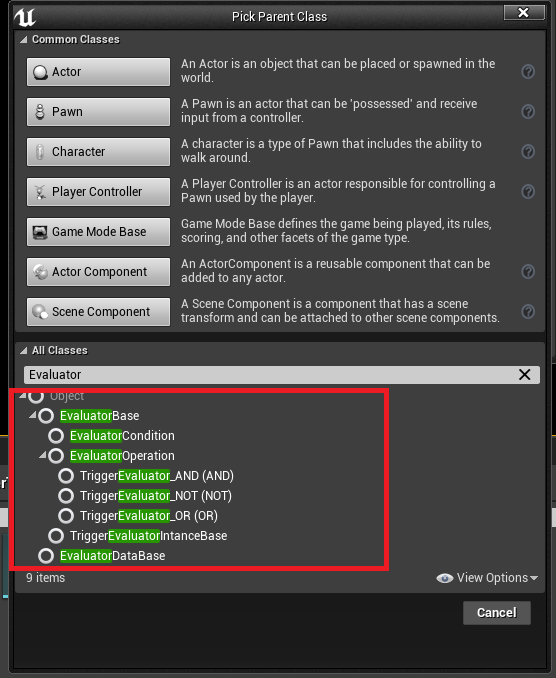
**StopConditon,**

**SleepCondition.**

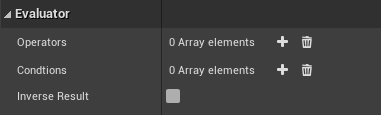
这些条件都会使用到Trigger条件。

1. **基本结构**

Trigger条件分为2大类，一个是操作符，一个是条件内容。



而所有的Trigger条件都是可以通过勾选InverseResult来让结果最终结果反过来。这个和下面提到的OR是一个意思，只是这里更加方便快捷的。

1. 操作符EvaluatorOperation是用用来组织使用那些条件，能够让多个条件按照一定的组织后，使用他们组合后的结果

操作符里面是可以包含额外的操作符的。

操作符里面的Conditions就是下面的条件内容

1. 条件内容EvaluatorCondition是用来描述具体的这个条件是如何描述的。
2. ExternalData:这个目前是当这个条件不满足的的时候来做额外的再次判断，这个是为了能够在不改变已有的配置能够支持新的功能要求。目前还没怎么用到。

还是以最开的例子来组合条件就是AND(朋友，时间，宇宙第一宇宙)，只有当这里都满足的时候我才愿意吃饭的。

1. **条件操作符Operator**
2. AND

他里面的所有的条件和操作符都要返回TRUE才会最终返回TRUE

1. OR

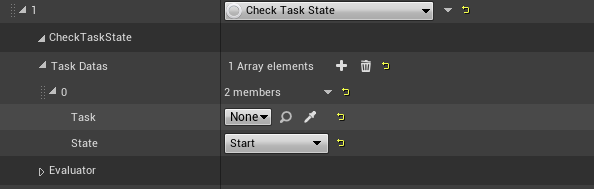
只要他里面的所有条件和操作符有一个返回TRUE就返回的是TRUE

1. NOT

将最终结果翻转，和InvertResult是一个意思。

1. **条件内容 Condition**
2. **CheckTaskState**

这个是用来监视场景中哪一个Task是否到过某一个状态的。



在这个里面需要配置TaskDatas，这个是一个数组，我们可以添加多个需要监听的Task,这个条件只有当多有的Task都到了指定状态才能返回TRUE.

1. **UCheckGameOverFailed\_AllPlayerDead**

当所有的玩家都死亡了的，那么这个条件就是真。

1. **CheckGameOver\_Exist**

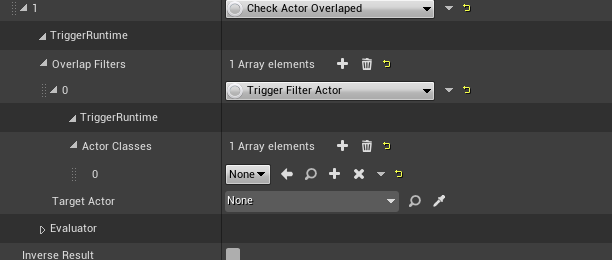
当有玩家主动离开游戏了这个条件就是真

1. **CheckGameOver\_Disconnect**

当有玩家和DS断开连接的时候，这个额条件为真

1. **CheckActorOverlaped**

用来监视是那个Actor碰到这trigger的。

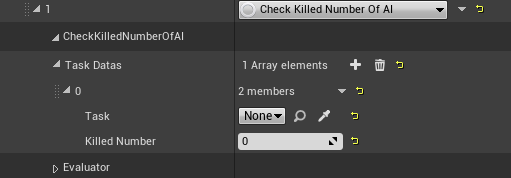


这个里面需要配置的是两个

* TargetActor：这个是用来确定是哪个Actor需要被碰到，如果这个值是空的，那么就会拿这个条件所在的Actor。
* OverlapFilters：这个是用来过滤掉哪些碰撞是不要考虑的。

1. **CheckKilledCountOfAI**

这个是用来监视指定的Task里面有多少个AI被杀死的

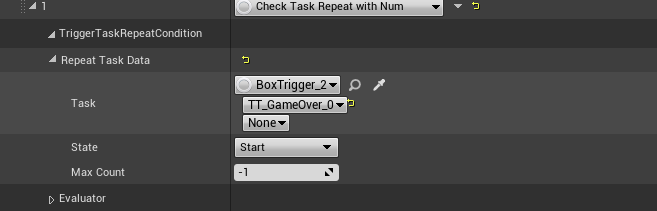


这里需要配置两个：

* KilledNumber:被杀死的AI的数量
* Task:需要监听的是哪一个Task里面生成的AI

1. **CheckTaskRepeatWithNum**

这个是用来监视指定的Task的循环次数的

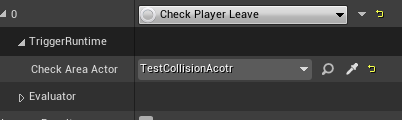


这里的MaxCount如果是-1就表示是无限次数的。

1. **CheckPlayerLeave**

这个条件是用来检测玩家是否离开了某指定区域，这里的玩家并没有限定为本地玩家，同一局内的任一玩家离开指定区域后，这个条件都会被判断为满足。

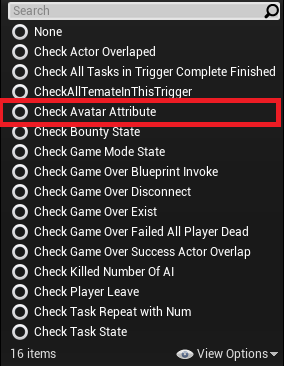
具体如下：



* Check Area Actor：一个Actor，用来与玩家进行Overlap检测，即指定监视的区域

1. **CheckAvatarAttribute**

这个也是可以用来监视玩家或者NPC的属性，后期也可以包含各种行为（这两个在概念上是不同的，但是在实现上可以统一来做，可以看成行为是对属性的扩展）

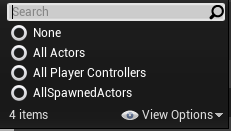


在任何一个可以添加Condition的地方都可以放置这个条件。

这个Condition包含两个主要部分：



* Avatar Access Rule: 这个是用来确定如何获得对象Avatar的方式。在目前只有3个：



* All Actors: 获得地图中当前的所有的Actors.
* All Player Controllers:获得当前地图中所有的玩家
* All Spawn Actors: 获得所有在这个Condition生效之后新生成的Actor

在所有的方式里面都有一个基本的配置Filter：

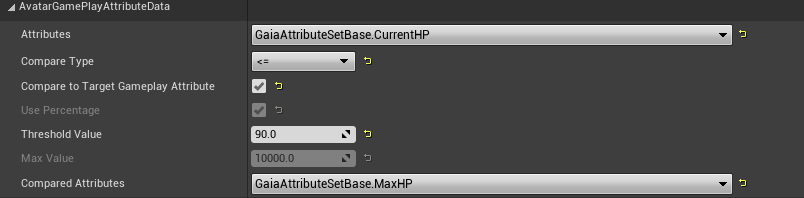


这个Filter是用来过滤哪些Actor是有效的，目前有效的就是TriggerFilterActor

* InspectedDatas:这个是用来确定监视目标的数据。



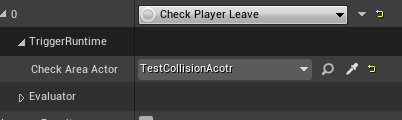
* **Avatar Game Play Attribute Data**:这个是用来监视玩家/NPC的属性是否达到预期



这里里面配置的是：

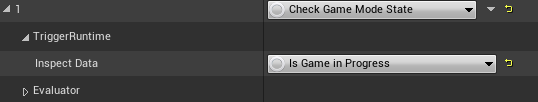
* **Attributes**:需要监视的玩家的当前属性
* **Compare Type:** 这个是比较的方式，有>(大于),<(小于),>=(大于等于),<=(小于等于)，=(等于)，
* **Compare To Target Game Play Attribute:**这个是用来确定是否让Attribute和Gameplay Attribute 进行比较
* **Use Percentage:** 是否让Threshold Value是表示的百分比。
* **Max Value：**这个只有在Compare To Target Game Play Attribute没有勾选，Use Percentage:勾选后才会生效，当不和Attribute进行比较又想用百分比，那么就说明得需要一个最大值来确定比较的值。
* **Compared Attribute:**这个是用来确定需要对标的属性
* **Avatar in Target Area**：用来检测Avatar是否在目标区域内的https://internal-api.feishu.cn/space/api/box/stream/download/wps/auth_code/?code=212d8d5ce3128e6a9f4d5c60f4d9a689_e0da7e6ca19b3f2a_Q97FN6B3DO_B5D9P6BPOP21UPDMG52T4JNRJ4
* **Target Area**: 这个是用来表示目标区域的范围

1. **CheckBountyState**

该条件用来判断玩家在局内是否击杀了特定数量的某种Npc，直接在BountyTargets中配置相应的NpcId和达到条件所需数量即可，当BountyTargets中配置的所有Npc的击 杀数都达到了配置的数量，该Task就会被触发（前提是勾选了Execute Auto）

1. **CheckGameModeState**

该条件是用来监控GameMode和GameState里面的一些属性或者行为。比如说是否游戏开始，是否地图已经载入的等等关于游戏模式里的信息都可以通过这个来监视

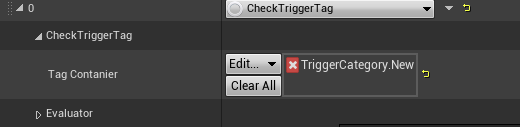


这个里面的InspectData有下面的：

* IsGameInProgress:游戏是否已经开始，此时玩家都已经进入游戏中

1. **CheckTaskTag**

这个条件是用来监控整个TriggerEventSystem 里面是否有指定的TriggerTaskTag的。



* TagContanier:指定的TriggerTags