**CHECK POINT 代码文档**

**作者：刘连友**

**时间：2021/4/26**

**版本：1.0**

1. **概述**

CheckPoint的是用来团队复活的任务，能够让小队在一个CheckPoint定义的复活点全体复活。

他是依赖RespawnTask，RespawnTask是用来单人复活。因为都是有复活的功能，因此我将CheckPoint继承与RespawnTask，这样就可以重复使用非常多得功能

1. **代码结构**
2. 继承关系：UTT\_RespawnTask.
3. 依赖关系：UTriggerRecordBase。
4. **代码实现：**
5. 在这个类里面主要是重载了基类UTT\_RespawnTask 的4个函数：

* virtual void GeneratePotentialRespawnedPawn(TArray<APawn\*>& OutPawns, APawn\* ToggledPawn);

在这个函数里面主要是用来收集所有的潜在的需要在这个复活的玩家，因为这个Task是用来团队复活的，所以当有一个玩家触发了这个那么他所在的所有的小队都会注册到这个task里面，这里需要注意的是目前定义的小队，是这个地图里面的玩家都算是一个队伍。

* virtual void StopOtherRespawnTasks(const TArray<APawn\*>& NewToggledPawns);

这个函数主要是用来做这类型Task互斥。因为目前RespawnTask之间是有互斥的。在任何时间段里只有一个ResapwnTask能够有效。因此在这个函数里面就是定义如果停止其他的RespawnTask.需要注意的是这里的Stop不是TriggerTaskBase里的Stop State, 这里的Stop只是说让其他的CheckPointTask 失效，不是停止

* virtual void TryToRespawnNewPawn(APawn\* Pawn);

这个函数是用来真正用来复活新玩家，正如他的名字，这个只是说尝试，不是一定，具体的需要在这个Task里面定义，在这个Task里面是会检测是否队伍里面的玩家都死亡了的，那么这些玩家就能复活，具体真正的复活时间是在Active里面的。

* virtual void RespawnNewPawn(APawn\* OldPawn, APawn\* NewPawn, ERespawnType RespawnType);

这个是新角色创建的事件，在基类RespawnTask里面可以看到非常多的回调事件，这个函数只是用来给子类用的。

1. 数据：
   * int Level;

所有的CheckPoint都有群组和等级的概念的，比如说当玩家激活的CheckPoint要比之前的CheckPoint要远，那么之前的CheckPoint 就会失效用当前的，如果当前的CheckPoint 的level要比之前的等级低，那么就不会生效，

这个是在函数FPotentialRespawnedPawnGroup：：IsVaildCheckPointToRespawn函数里面来确定的。

* + UTriggerRecordBase\* RecordTemplate = nullptr;

这个成员变量是用来记录当这个CheckPoint激活后，后面所有的激活task这个成员都会记录，这样就可以在玩家重生后能够重置那么已经激活的task.

在这个里面有3个函数需要注意的的：

* + - void Execute(FRecordInstigatorInfo& InstigatorInfo);

这个是用来执行这个记录的，就是将StartRescord之后收集的信息进行处理的。

* + - void StartRecord(FRecordInstigatorInfo& InstigatorInfo);

在执行这个函数后，那么这个对象就会开始记录它所关心的数据，这些数据最终会是在Execute里面使用。

* + - void EndRecord(FRecordInstigatorInfo& InstigatorInfo, EEndRecordType EndType);

这个是用来结束记录的，正如所有的开关都一个开，和关的对应，这里也是的。

需要注意的当这个函数执行的时候不代表这个对象开始处理记录的信息，所有需要记录的信息处理都是在Execute函数里面的。

* + static FPotentialRespawnGroups GroupInfo;

这个数据是静态的，之所以要用静态的是因为，所有CheckPoint都要共用一份玩家数据，因为他们是互斥的，所以就没必要每个checkpoint都保存一份玩家数据，这样数据量多，而且维护比较麻烦因为每次改动都需要将全局的数据改动，这里面就是缓存了所有角色和CheckPoint之间的关系