一、GameplayEffect

一般的用途：

\* 修改 AttributeSet 内的属性值

\* 给目标添加 GameplayTags 标记

\* 触发 GameplayCue，播放各种特效，音频等

\* 赋予目标新的技能（调用 GiveAbility）

**FGameplayEffectSpec**

给目标施加 GE，除了 GE 本身配置还需要一些其它参数（比如：GE 等级、什么时候施加 GE、谁施加的 GE 等、GE 的目标是谁等） 因此施加 GE 需要包装一层，而这就是 FGameplayEffectSpec，且GameplayEffect会存在GamePlayEffectSpec中，不可以进行写，因此CDO不能随意进行更改！

**FGameplayEffectSpecHandle**

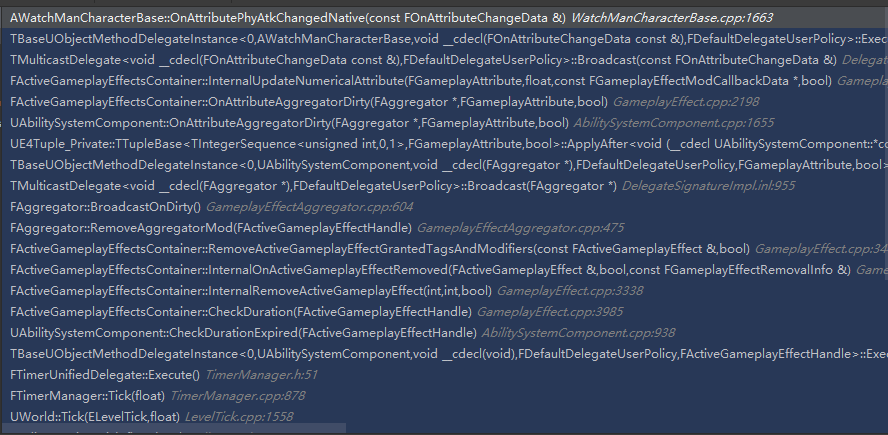
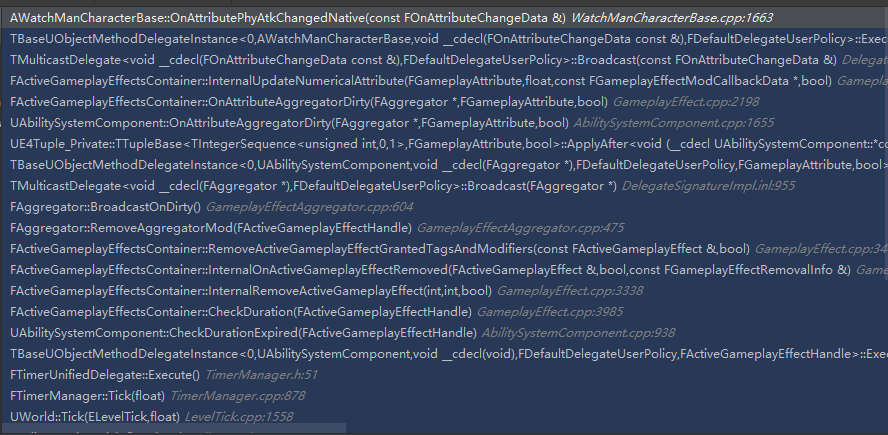
采用TSharedPtr<FGameplayEffectSpec>的形式包装FGameplayEffectSpec，来保证FGameplayEffectSpec的自动delete，而Handle的管理则会随着作用域的结束自动结束生命周期。

**FGameplayEffectContext**

FGameplayEffectContext 主要是用来传递内容？？ 以传递给 UGameplayEffectExecutionCalculation，UGameplayCue使用

**FAggregator**

**GE随Duration的移除过程**

****

**ExecutionClac**

只能用于InstantGE或者是Duration

如何在Execution中去拿到Modifer的值？？

1，通过SetByCaller();

2，通过一个计算接口可以拿到！！

小技巧：如何想要在能预览Modifiers的情况下也能使用Calculation：Modifiers随便填一个属性即可！

特殊用法

CoolDownGE

其实说白了就是一个HasDuration的GE

比较官方的处理方法就是通过SetByCaller这种方法来动态指定CoolDownGE的Duration和Tag，来保证2者的可配置化，CoolDownTag和冷却时间都可以定义在GA上以做相应处理。

并且也可以监听冷却时间的开始与结束以保证一些状态的同步（实现异步任务 AsyncTaskEffectCooldownChanged）

DurationGE的修改方式（待验证）

**bool** UPAAbilitySystemComponent**::**SetGameplayEffectDurationHandle(FActiveGameplayEffectHandle Handle, **float** NewDuration)

{

**if** (**!**Handle.IsValid())

{

**return** false;

}

**const** FActiveGameplayEffect**\*** ActiveGameplayEffect **=** GetActiveGameplayEffect(Handle);

**if** (**!**ActiveGameplayEffect)

{

**return** false;

}

FActiveGameplayEffect**\*** AGE **=** **const\_cast<**FActiveGameplayEffect**\*>**(ActiveGameplayEffect);

**if** (NewDuration **>** 0)

{

AGE**->**Spec.Duration **=** NewDuration;

}

**else**

{

AGE**->**Spec.Duration **=** 0.01f;

}

AGE**->**StartServerWorldTime **=** ActiveGameplayEffects.GetServerWorldTime();

AGE**->**CachedStartServerWorldTime **=** AGE**->**StartServerWorldTime;

AGE**->**StartWorldTime **=** ActiveGameplayEffects.GetWorldTime();

ActiveGameplayEffects.MarkItemDirty(**\***AGE);

ActiveGameplayEffects.CheckDuration(Handle);

AGE**->**EventSet.OnTimeChanged.Broadcast(AGE**->**Handle, AGE**->**StartWorldTime, AGE**->**GetDuration());

OnGameplayEffectDurationChange(**\***AGE);

**return** true;

}

CostGE

各种TagContainer分析

Immunity类

二、关于GAS的预测

1，为什么要做预测？

针对客户端的操作，无论是帧同步还是状态同步，通过Client->Server->Client就一定会让客户端的表现出现延迟，由此引入预测来降低这个延迟

2，GAS怎么实现预测

FPredictionKey：预测密钥

FPredictionKey 是在 GameplayAbility 系统中支持客户端预测的通用方式。

客户端每生成一个Key时候在一个16位的全局整形变量加一。生成Key之后客户端可以绑定预测操作或副作用。

在接下来要的Ability激活预测、Montage播放等地方都大量用到了副作用绑定，作用是为了做“回滚”操作，如果预测结果失败，客户端通过这个Key找到关联的副作用。

接下来客户端会把这个预测密钥RPC到服务器上，服务器会通过这个Key相应预测密钥，如果服务器操作失败，则需要告诉客户端做回滚操作，客户端就可以通过这个Key找到相关的副作用并执行，PlayMontage的预测就是这种方式进行回滚的！

①关于GA的预测

InstancingPolicy:

1，InstancedPerExecution:每次执行实例化，不必考虑重置变量问题

2，InstancedPerActor:会重复使用，并且每次执行这个技能后需要清理这个技能的变量和状态

3，Non-Instanced

NetExecutionPolicy:

1，LocalPredicated:

2，LocalOnly:

3，ServerInitiated:

4，ServerOnly:

TryActivateAbility()->

InternalActivateAbility()->

ABILITYLIST\_SCOPE\_LOCK() 防止技能激活状态中被销毁->

CanActivateAbility() 验证冷却,Cost等状态->

标记Spec OwnerActor激活次数，可用于判断技能是否处于激活状态->

FScopedWi