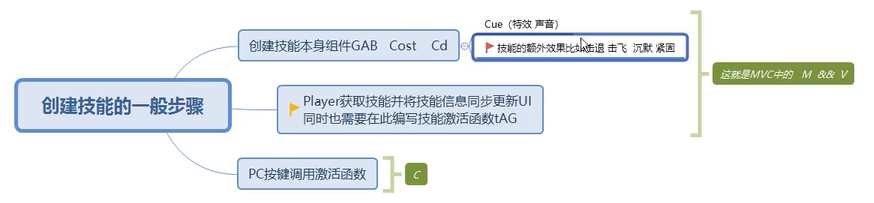
1. GA、GE、Montage
2. 技能撞到哪些敌人

应用伤害的第二种方式

GAS发消息传到GAB中

击退效果



创建动画蒙太奇

因为这个蒙太奇自己就有特效，所以不用特意创建Cue

创建技能02Dash

此技能是一次性耗蓝，所以可以直接提交。

添加技能Tag

PlayMontage

添加GE\_Cost/CD/Damage

CD设置：持续时间策略和冷却时间，以及Gameplay效果用标签阻止能力，并且加上Tag。

Cost设置：设置Gameplay效果耗蓝即可。

Damage设置：设置Gameplay耗血即可。

在PlayMontage之后EndAbility。

进入BP\_Player学习技能

在变量中给GameplayTagContainer类型变量和BaseGameplayAbility类型变量加上数组元素。

现在进入游戏，就可以使用技能了。

下面来实现技能效果。

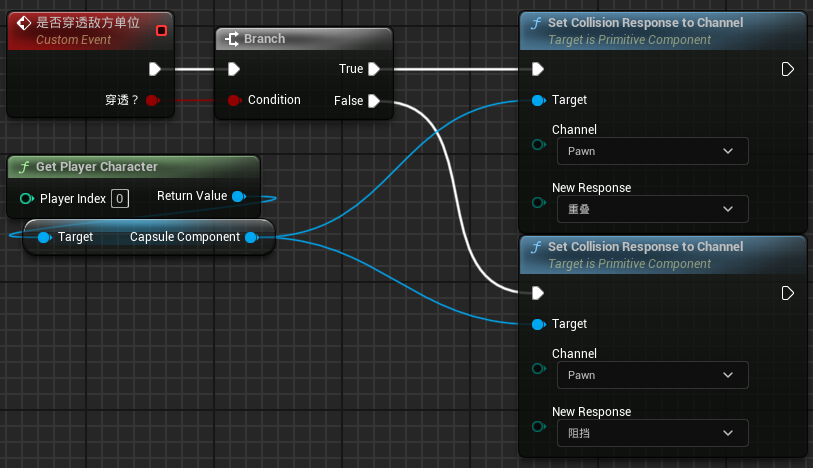
1. 角色移动：
2. 命中敌方单位
3. 击退敌方单位
4. 冲刺方法：

给角色一个力

在GA中实现一个自定义事件

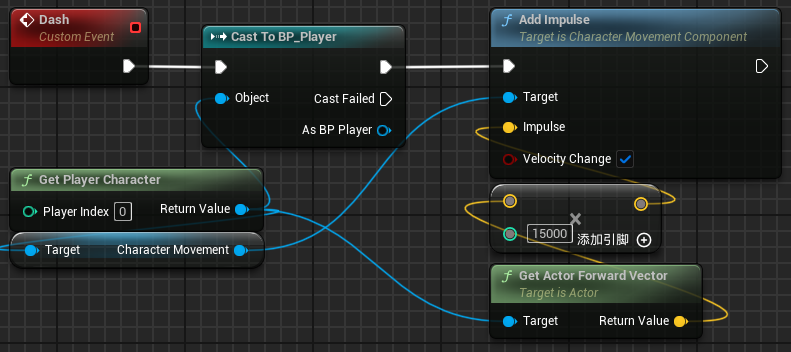
在GA中

创建自定义事件，实现技能位移穿透敌方单位

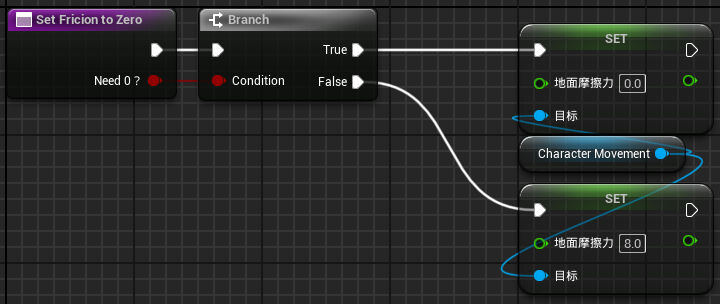


在EventActivateAbility中调用

创建自定义事件，实现冲刺功能。



这里向量乘了一个非常大的数，因为地上有摩擦力。我们实现一个设置摩擦力的函数。在BP\_Player中



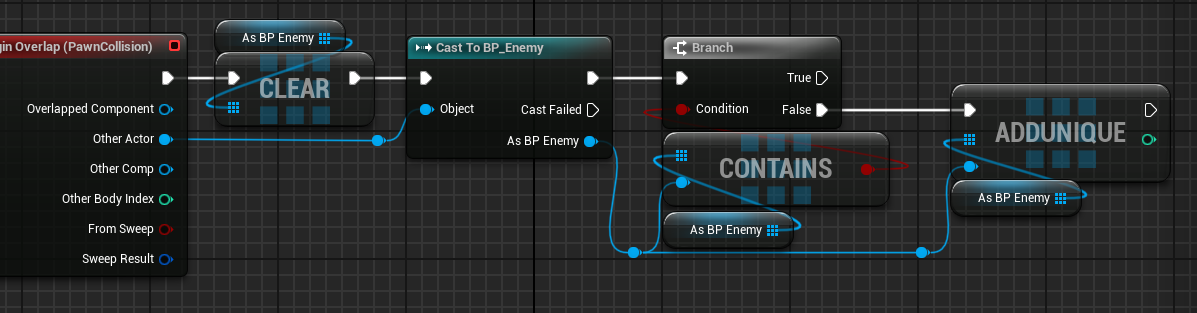
## 然后实现判定技能撞到了哪些人

在BP\_Player中添加一个SphereCollision，命名为SpawnCollision

体积稍微大一点，碰撞类型为OverlapOnlyPawn

撞到了多少敌人肯定不止一个，所以创建一个BP\_Enemy对象类型的数组

对这个HitBox进行OnComponentBeginOverlap。



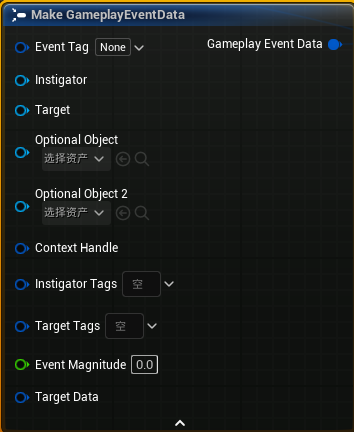
这里就要谈到GA传递数据应用伤害的第二种方式，也是官方推荐的做法：发消息

发的消息就是一个Tag。也就是Send Gameplay Event To Actor节点



由自己发消息。

Payload叫负载，由payload进行make一个结构，会得到结构的成员



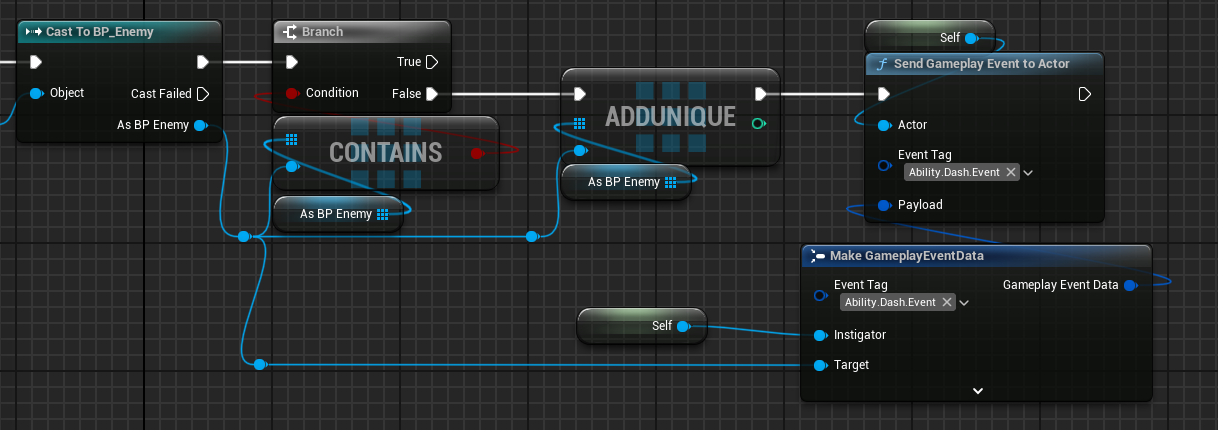
Instigator就是教唆者，也就是BP本身，是指由谁去make

Target就是我们要传递过去的数据，也就是CastTo的返回值

然后要给SendGameplay函数的EventTag参数进行赋值。这个标签是一个事件标签，等会到GA中去接受这个标签。

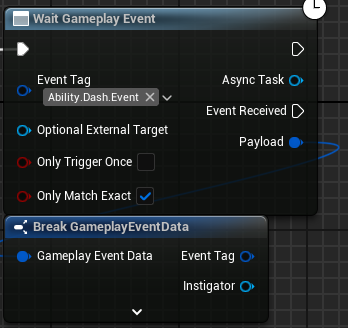
这个事件标签的路径是Ability.Dash.Event

这个Tag也可以在make节点中给第一个参数赋值。

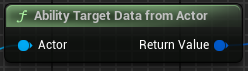


在BP\_Player中是发事件，我们要回到GA中接收事件。

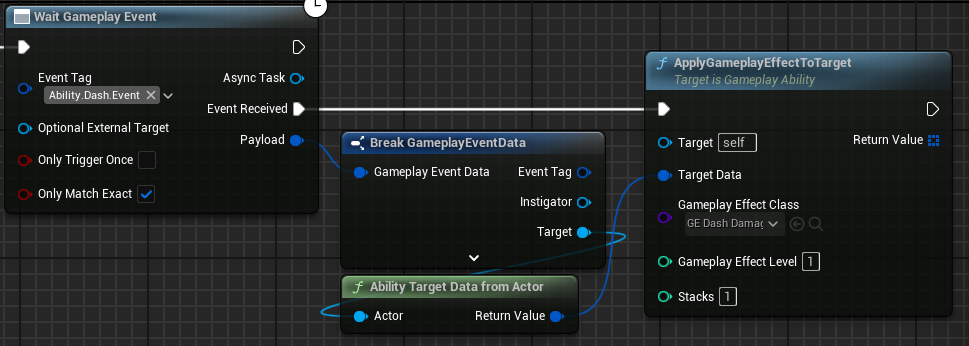
也就是Wait Gameplay Event函数。



从负载里面拿到参数，拿到负载的Target，通过AbilityTargetDataFromActor来获取Actor的能力组件



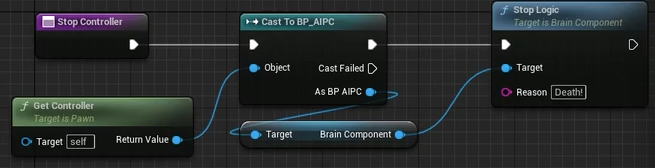
然后就可以直接ApplyGEToActor

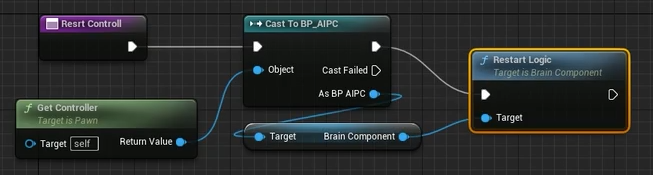


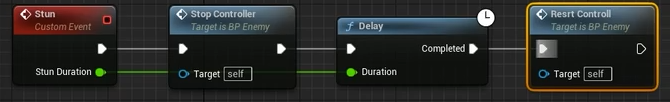
这样就实现了伤害

## 下面实现击晕和击退。

在BP\_BaseCharacter中自定义事件Stun实现击晕。击晕就是停止控制，在Delay控制时长之后再重新拿起控制器。







还有击退。击退与冲刺类似同理。在BP\_BaseCharacter中实现。

然后在GA中调用这两个函数。

## 冲刺期间禁用其他技能

在GA中，细节中的“激活已拥有标签”，加上标签Ability.Dash.Active。

进入其他技能的GA，在细节中“激活阻止标签（ActivationBlockedTags）”选择这个标签。

这些标签会随着技能的结束自动消失，所以要用EndAbility函数。