## GA、GE--技能--普攻

创建一个平A以及CD

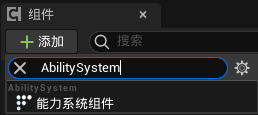
添加AbilitySystemComponent，添加Gameplay基类

GAS结合gameplay的做法

平A检测打到谁了

添加AbilitySystemComponent

在BP\_BaseCharacter中添加组件

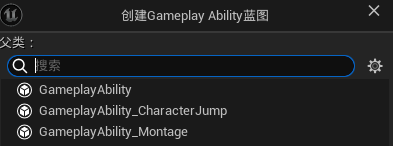


添加GameplayAbility基类

在GaneAsset文件夹下创建文件夹，命名为Ability

在文件夹中创建





选择继承自GameplayAbility，命名为GA\_BaseAbility



他应该是继承自一个自己写的C++类，我们当前用不到，所以就先不管。

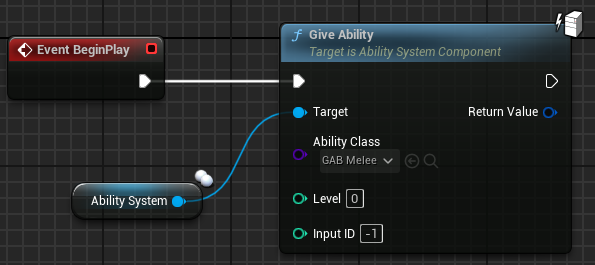
然后做第0个技能：普攻

在文件夹下创建文件夹，命名为00Melee

在里面创建继承自GA\_BaseAbility的技能蓝图，命名为GAB\_Melee

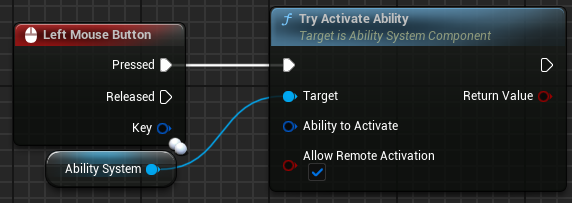
然后进入普攻的技能蓝图，在EventActivateAbility时Print看情况。

然后我们需要能释放技能。进入BP\_BaseCharacter，创建AbilitySystemComponent的节点，通过GiveAbility将技能蓝图给予角色



这样就习得了技能

接下来使用技能，通过AbilitySystem的TryActiveAbility节点



TryActiveAbility节点的AbilityToActivate

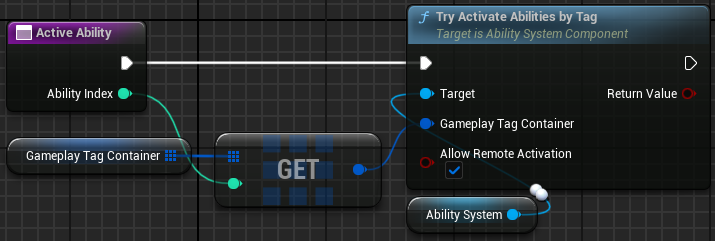
我们也可以用TryActivateAbilityByTag来实现技能使用。



我们会有很多技能，一个一个写会很麻烦。所以我们用函数来帮助快速实现

创建一个函数，命名为“”

首先我们根据TryByTag的GameplayTagContainer进行Promote To Variable。注意，不是局部变量。因为我们要用这个变量存储角色的技能。这个变量就用默认的名称GameplayTagContainer就可以，但是要是数组型变量。我们通过角色技能的索引来Get（a copy）容器中的具体技能。



然后点击GameplayTagContainer变量，在细节中给变量的默认值进行设置



在选择添加时，会有如下的界面



然后我们需要回到GAB\_Melee蓝图，在类默认值的细节中找到标签



给能力标签加个标记



点绿色的加号

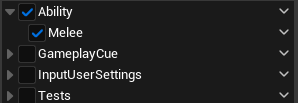


标签是树形结构，命名需要用 . 来表明标签结构与层次

同时，源我们也要进行选择

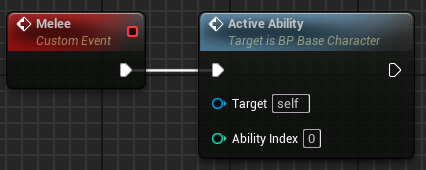


创建好后进行勾选

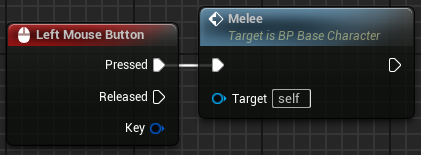


然后回到BP\_BaseCharacter的ActiveAbility函数中，给GameplayTagContainer的默认值的索引0进行一个赋值。

回到BaseCharacter的事件图表中，CustomEvent，命名为Melee，当Melee触发的时候，执行ActiveAbility函数，索引为0，表示释放普通攻击。



然后进入BP\_Player，当LeftMouseButton时，调用Melee事件

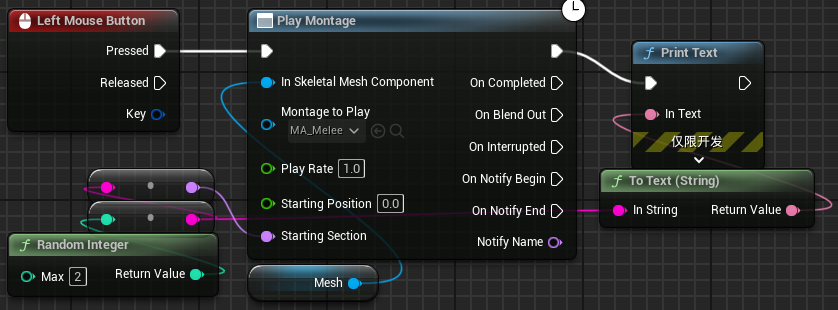


这样就可以释放普通攻击。

（我们把普通攻击放在基类里，因为Player和Enemy都可以释放普通攻击。其他的技能，一般来说，Enemy没有，但是Player有。所以后续技能不在BaseCharacter中制作，在Player中制作）

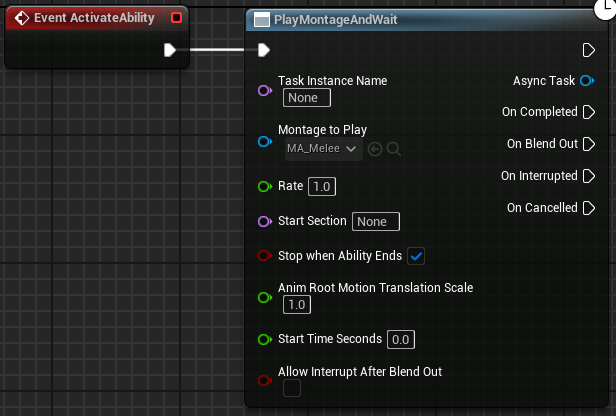
（还需要进入BP\_Player。我们需要确保EventBeginPlay时继承父类的BeginPlay。 ）

目前的技能释放是在BP\_Player中



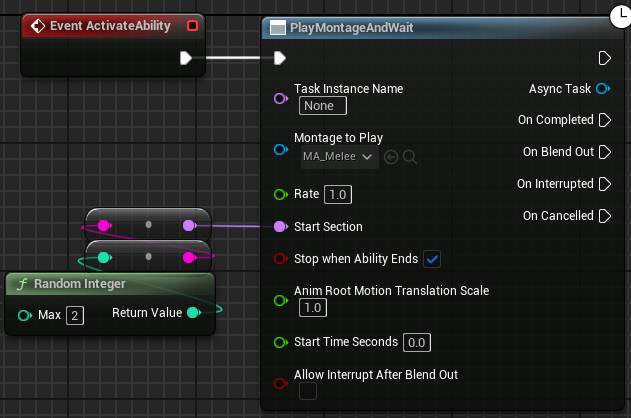
接下来我们来真正的实现一下普通攻击。进入GAB\_Melee，创建节点PlayMontageAndWait





它不需要连Mesh组件的节点，因为会自动匹配。如果匹配不上会自动失败。

同样的，因为MA\_Melee有两段，所以我们需要随机给一个值来决定StartSelection。



现在的情况是，当我们快速点击鼠标左键的时候，角色会抽搐。

解决方法是我们给普攻一个CD。

在GAB\_Melee的类默认值的细节中，可以直接看到有冷却部分



直接点加号，创建Melee的技能效果，命名为GE\_MeleeCD

GE基本上只存储数据和处理数据，从来不写逻辑。不需要写任何蓝图。

进入GE\_MeleeCD，在类默认值的细节中，找到“持续时间”

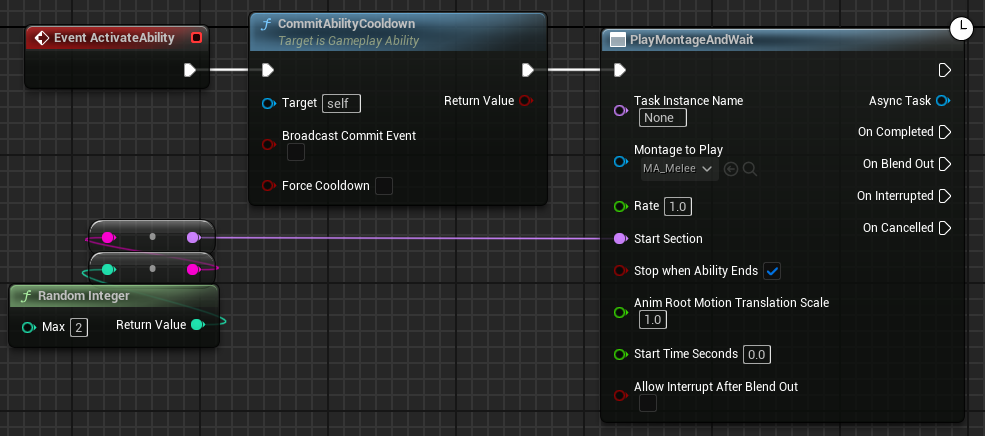


我们给可扩展浮点幅度赋值0.5，表示冷却时间为0.5，并给“Gameplay效果--组件”赋值一个“用标签阻止能力”，并将标签赋值





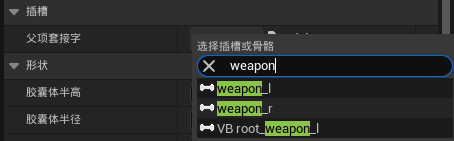
然后我们需要提交CD。看什么时候计时。有的技能是刚开始就计时，有的技能是结束后计时。普攻我们按照刚开始就计时。提交CD很简单，在GA中CommitAbilityCooldown函数即可。



接下来检测普攻打到了谁。这里我们就用Collision去做。这也是常规做法。

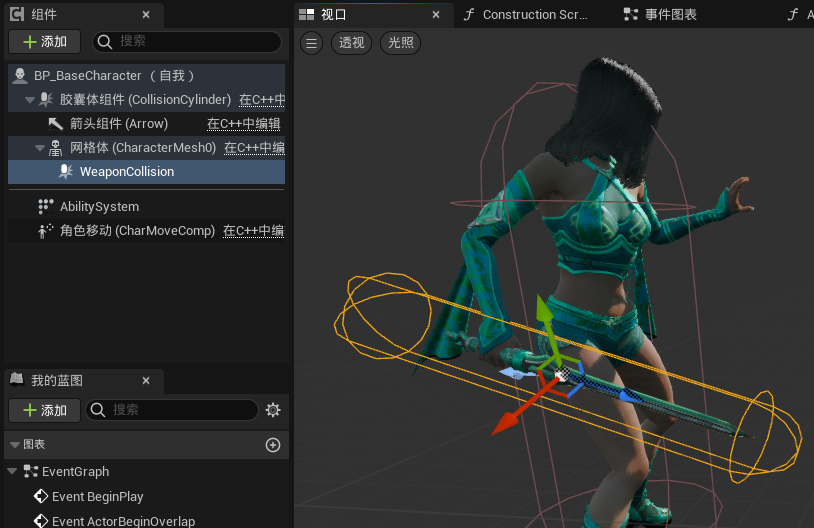
在BP\_BaseCharacter中创建胶囊体碰撞组件（现在BP\_BaseCharacter的Mesh的赋值情况和BP\_Player几乎一致）

碰撞组件需要放在Mesh上面，将碰撞组件通过插槽绑定在右手上（weapon\_r）



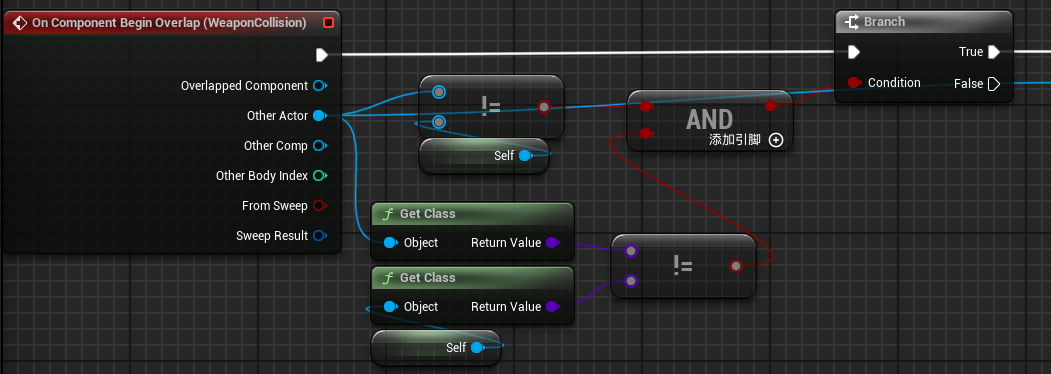
然后通过调整碰撞组件的角度、位置、大小、长度来让碰撞组件和剑尽可能地重叠。

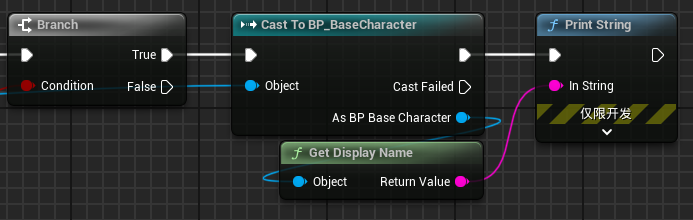
然后修改碰撞体组件的名称，命名为WeaponCollision



然后修改碰撞组件的“碰撞--碰撞预设”，只让它检测有没有碰到人（OverlapOnlyPawn）。并创建OnComponentBeginOverlap。

首先，OtherActor不能等于自己，且类型也不能等于自己的类型



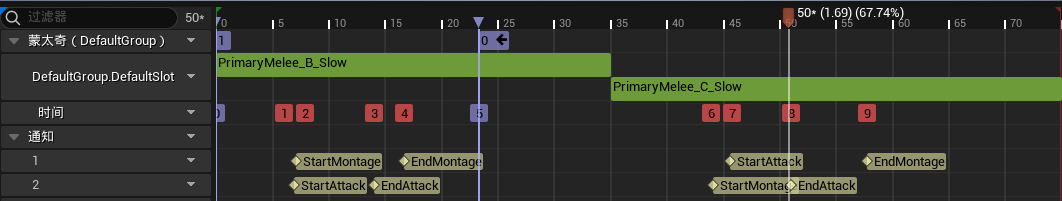


GetClass是工具集中的GetClass

这时只是碰撞发生的时候就触发，也就是说，即使没有挥剑也会触发。

在动画蒙太奇中，通过在攻击片段中添加两个通知，判断OnComponent是否在这两个通知之间，之后再处理后面的东西。

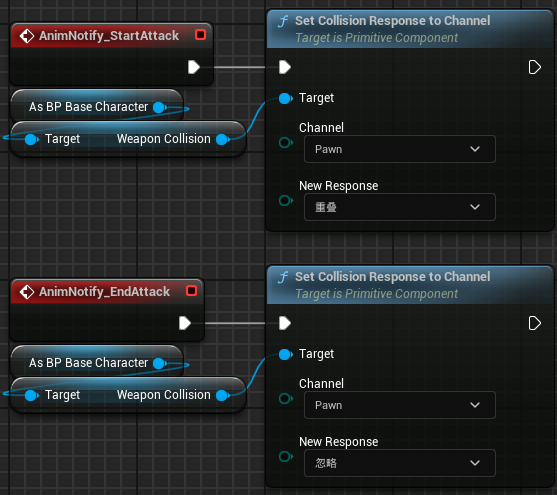
两个通知命名为StartAttack和EndAttack



然后进入到动画蓝图中

在事件图表中，添加动画通知节点





然后在BaseCharacter中，给WeaponCollision的碰撞预设设置为Custom



当然，这里最好用状态通知。如果这个动画中途被打断了，attack结束的时候的通知是无法触发的，导致永远无法命中了。