我们的逻辑设计是：

ACharacter->ABaseCharacter->ABaseLegend->BP\_Legend\_CHR\_01

ACharacter->ABaseCharacter->ABaseMonster->BP\_Monster\_CHR\_01

ABaseLegend比ABaseMonster多的是弹簧臂和摄像机

在BaseLegend中

class USpringArmComponent;

class UCameraComponent;

// #include "GameFramework/SpringArmComponent.h"

UPROPERTY(VisibleAnywhere,BlueprintReadWrite,Category="Components")

USpringArmComponent\* SpringArm;

// #include "Camera/CameraComponent.h"

UPROPERTY(VisibleAnywhere,BlueprintReadWrite,Category="Components")

UCameraComponent\* Camera;

然后在构造函数中实现初始化

ABaseLegend::ABaseLegend()

{

    // 初始化弹簧臂和摄像机

    SpringArm=CreateDefaultSubobject<USpringArmComponent>(TEXT("SpringArm"));

    SpringArm->SetupAttachment(GetRootComponent());

    SpringArm->TargetArmLength=1200.0f;

    SpringArm->SetRelativeRotation(FRotator(-45,0,0));

    SpringArm->bDoCollisionTest=false;

    Camera = CreateDefaultSubobject<UCameraComponent>(TEXT("CameraComp"));

    Camera->SetupAttachment(SpringArm,USpringArmComponent::SocketName);

    Camera->bUsePawnControlRotation=false;

    // 确保玩家控制为上帝视角wasd视角

    GetCharacterMovement()->bOrientRotationToMovement=true;

    bUseControllerRotationYaw = false;

    SpringArm->bInheritYaw=false;

}

BaseCharacter需要一个Actor组件来实现元素战斗功能

创建ActorComponent类ElementalCombatSystem，然后在ABaseCharacter中添加这个组件

// add sentence:class UElementalCombatSystem;

UPROPERTY(VisibleAnywhere,BlueprintReadWrite,Category="Components")

UElementalCombatSystem\* ElementalCombatSystem;

BaseCharacter也需要一个Actor组件来实现装备功能。

装备组件需要基于ElementalCombatSystem来给角色提供属性数值或额外技能。

在后期，我们会给角色的UI进行初始化

这里我们需要一个结构体来存储相应信息，并在蓝图中赋值。

出于“角色可能自身就拥有元素”的考虑，我们将这些信息封装在ElementalCombatSystem中

后期我们会给角色创建技能。Legends类有6个技能，Monster类不一定。为了方便管理，在BaseCharacter中声明这些技能。当我们创建蓝图子类的时候，需要实现这些函数

因为所有的Legend都是这样的功能，所以适合使用接口来实现

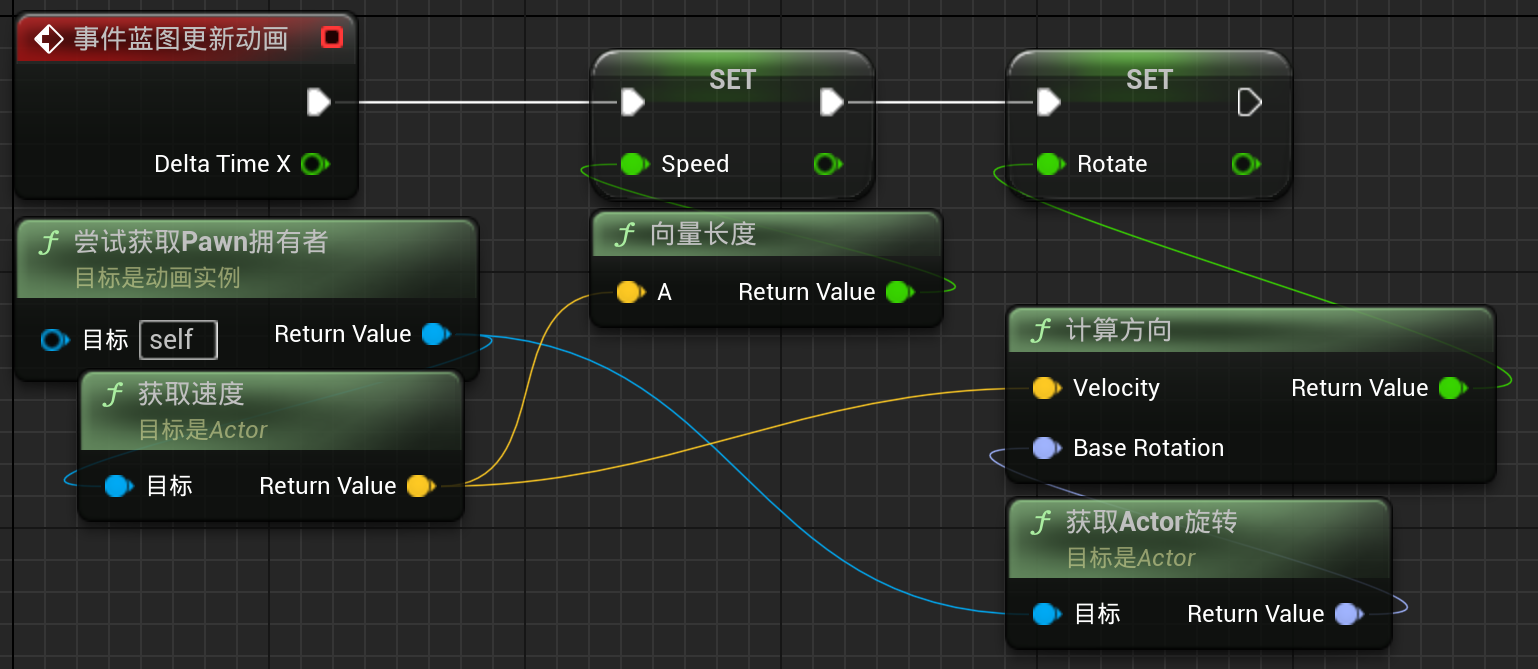
创建

动画系统，统一使用蓝图实现，而不用C++

创建模板动画蓝图，命名为ABP\_Template

创建Speed和Rotate变量，其他的基本不用弄了

但是也可以加一点东西，比如在事件图表中给两个变量赋值.



如果想通过混合空间播放器将动画播放给输出端，可能需要用到LayeredBlendPerBone函数，这个函数使用的骨骼名称可能不一样，所以这里操作实际上用处不大