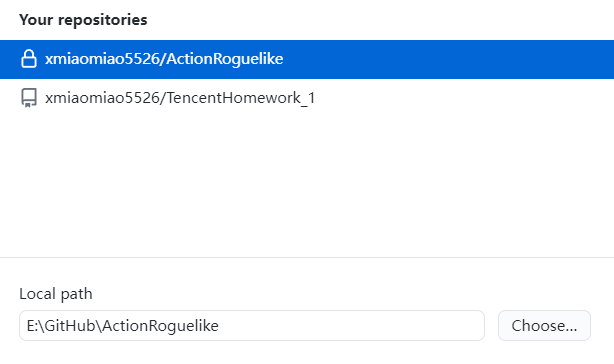
**ActionRoguelikeGame笔记**

# 新建项目与GitHub

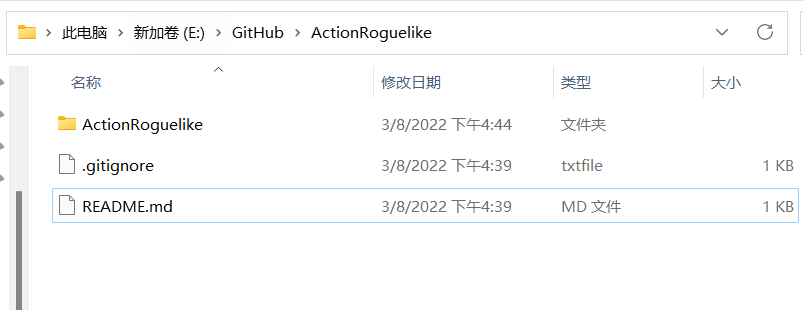
## 新建Github repository

使用Github Desktop，点击File->Clone reposit->建好的仓库->clone。



## 新建UE4项目

在Github的文件夹下新建UE4同名项目，此时文件夹内容：



将该文件夹下同名文件夹中的内容移到外面后，右键.uproject文件重新生成VS文件。

# 人物创建

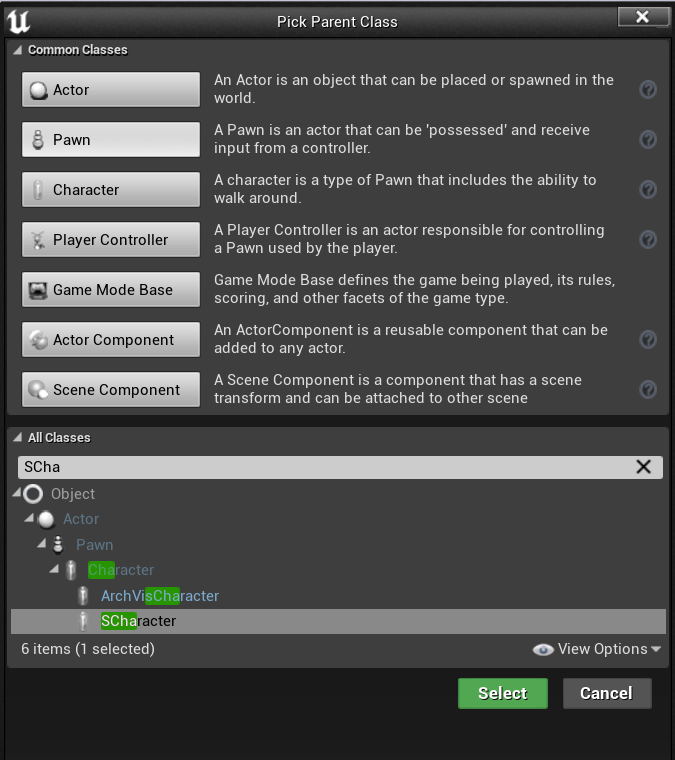
## 新建C++ Character类，名SCharacter

GENERATED\_BODY()宏应保证在类中第一行

CTRL+F5编译并唤醒UE编辑器（推荐使用，而不是热重载）

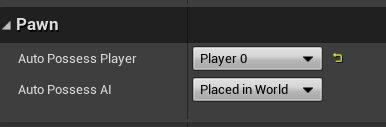
在Content文件夹下新建文件夹ActionRoguelike

## 创建蓝图类继承自SCharacter，命名为



拖拽到场景中，删掉场景中默认的Player Start

在Details面板中设置Auto Possess Player为Player 0

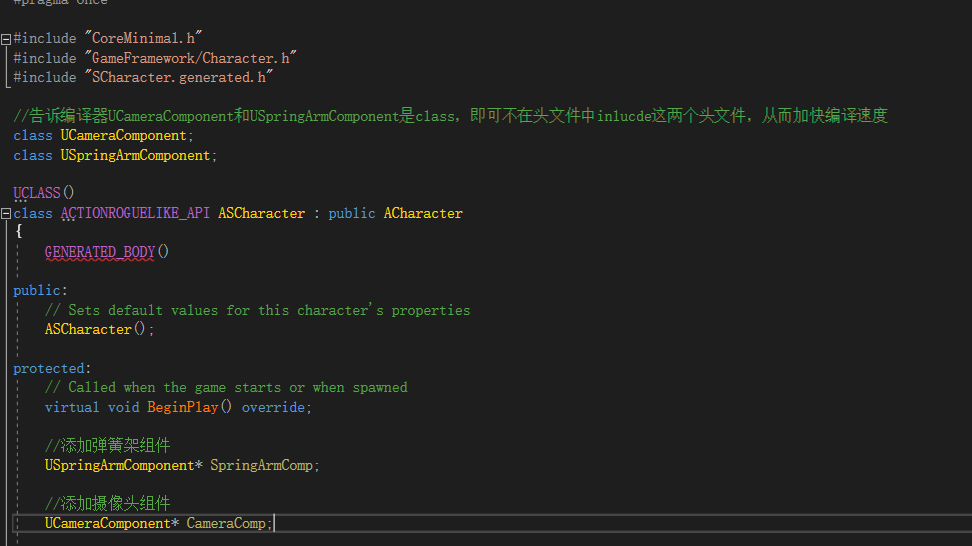


在Content->ActionRoguelike下新建文件夹map

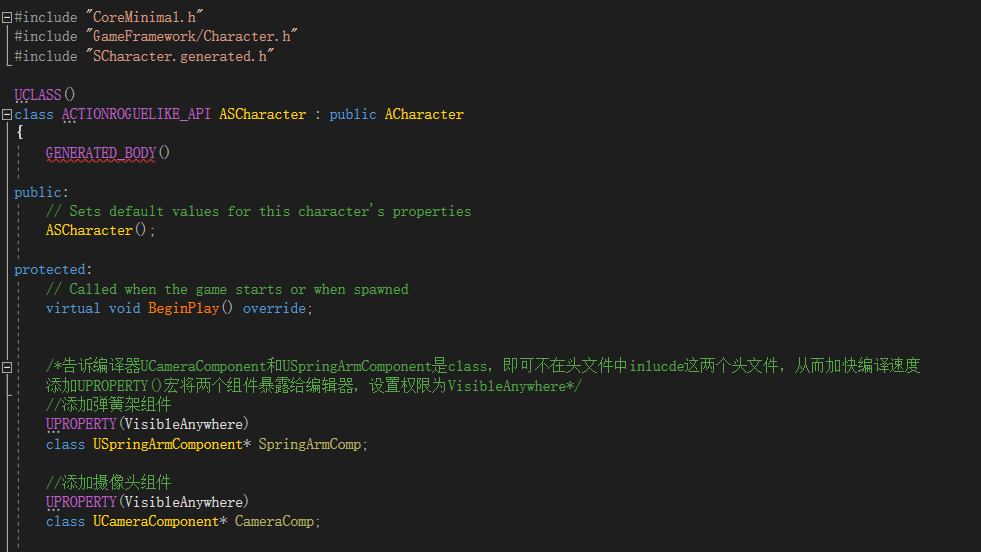
Save all，将level保存到map文件夹下，命名为TestLevel

2 添加摄像头组件

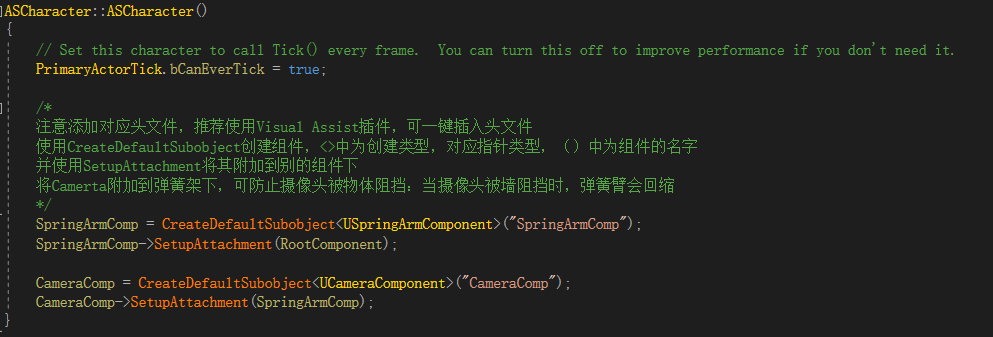
SCharacter.h



也可以把class在类名前面，不用在开头声明，并添加宏：



目前只是空指针，在.cpp文件中添加代码：



将如图所示三个模型文件夹放在Content下：

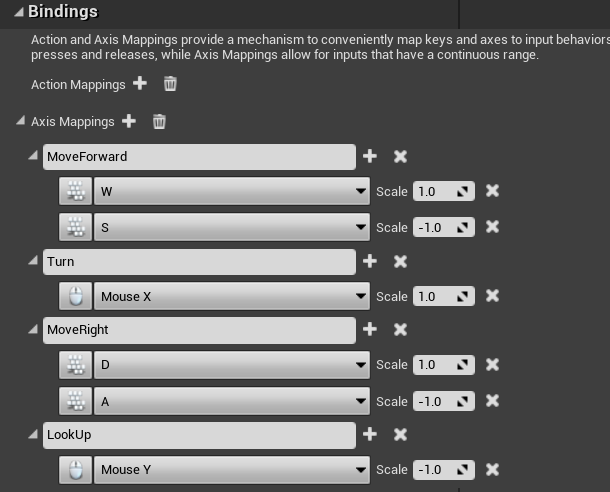


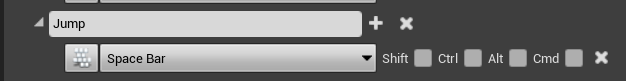
在人物的蓝图中设置Skeletal Mesh，并调整合适的高度与旋转

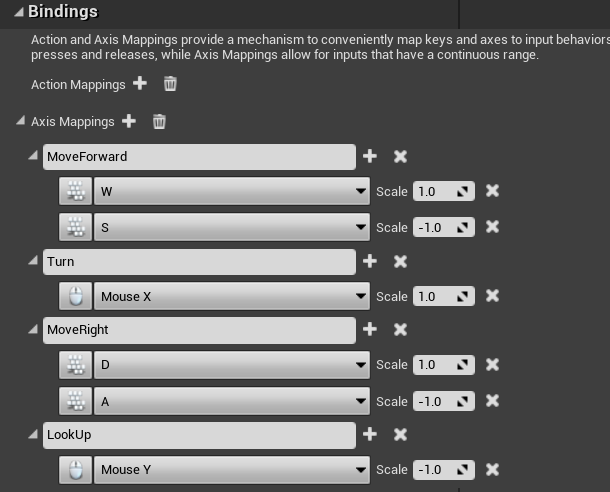
在人物的蓝图（Mesh）中设置动画

## 绑定运动输入

在编辑器中绑定轴Edit->Project Setting->左边一列input

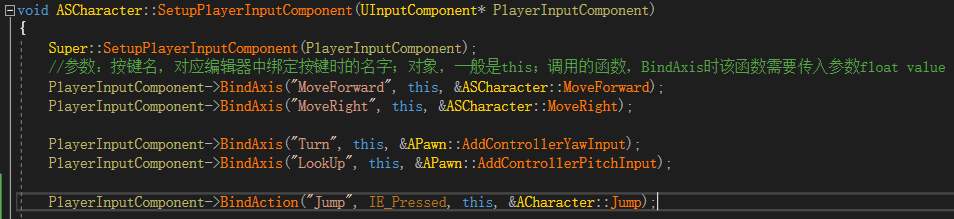




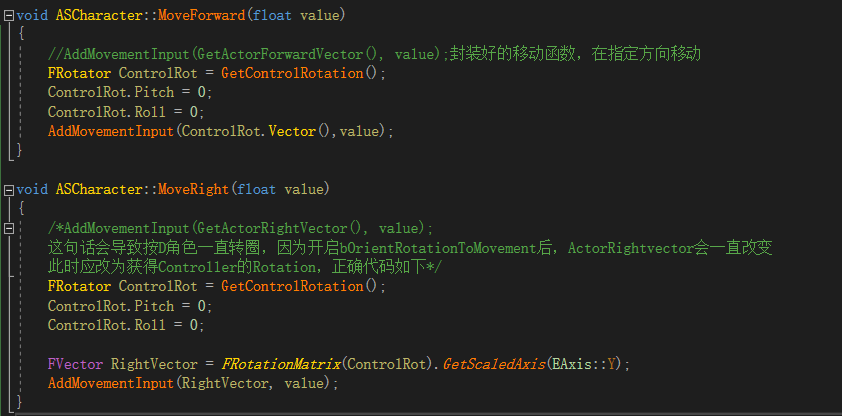


添加C++代码

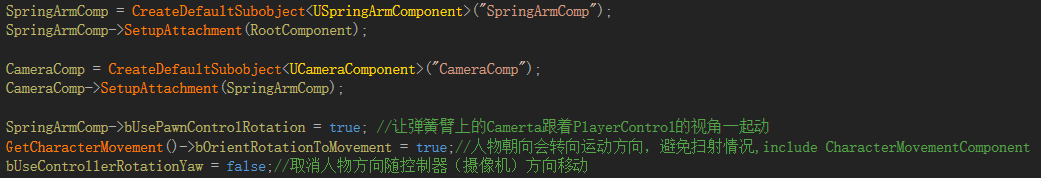
添加绑定



前后、左右运动函数



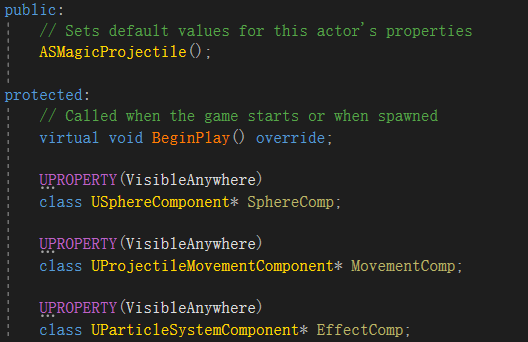
完善角色运动，在构造函数中修改弹簧臂和运动组件相关选项。以下布尔量可以在蓝图中找到并修改。

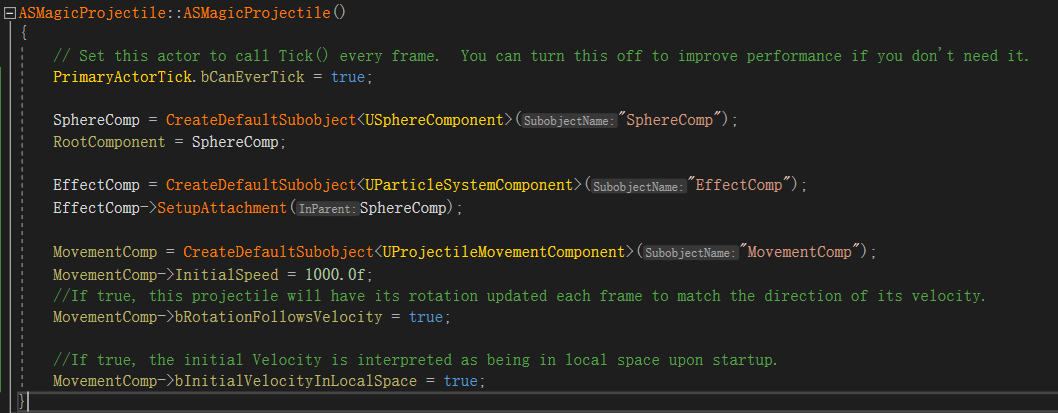


# 魔法攻击

## 创建类

创建如图所示AActor的子类，命名为SMagicProjectile，作为魔法攻击的发射物





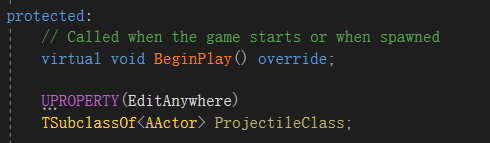
创建对应蓝图类，选择一个酷炫的粒子特效。

## 生成投射物（Spawn Actor）

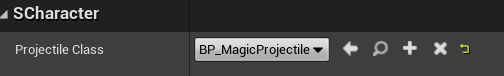
在SCharacter中绑定按键，在编辑器中将PrimaryAttack绑定到鼠标左键



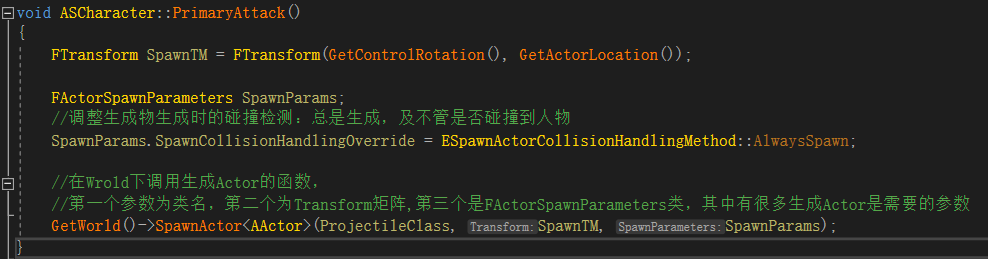
头文件中添加TSubClass



并在蓝图中指定ProjectileClass为魔法攻击粒子的蓝图类：

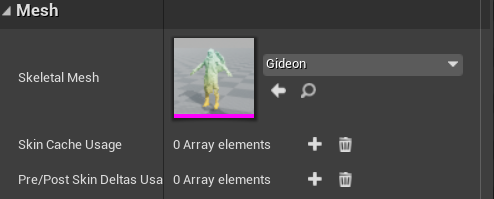


按键绑定触发的函数（绑定Action的函数不需要输入参数）：

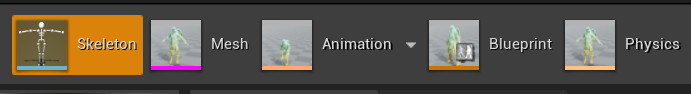


至此人物已经可以发射魔法飞弹了，但发射位置有点奇怪，我们希望从人物手的位置反射飞弹。

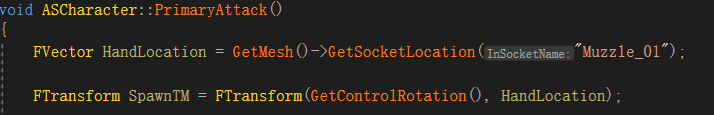
双击skeletal mesh中的图标可以看见骨骼的详情

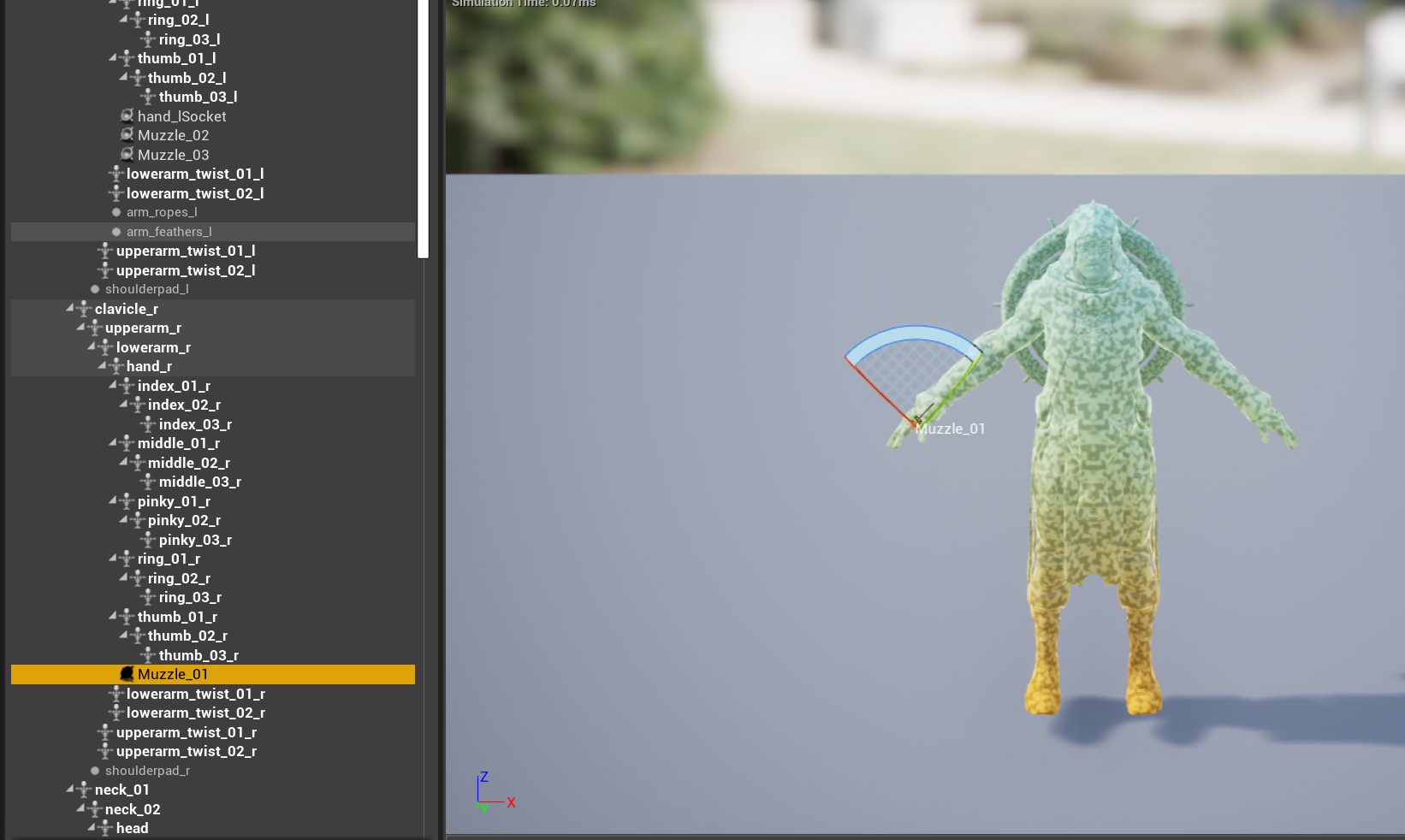


点击右上角Skeleton



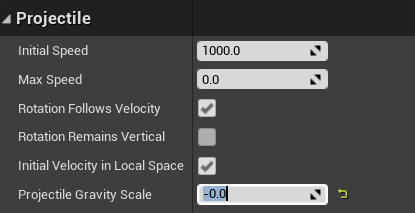
能够在左边一栏看见一堆东西，其中亮白色的是普通骨骼，而暗色的是插槽（Sockets），在UE编辑器中右键骨骼可以add Socket。找到Muzzle\_01，我们进入C++，指定飞弹从这个位置发射。修改部分代码：





## 碰撞设置

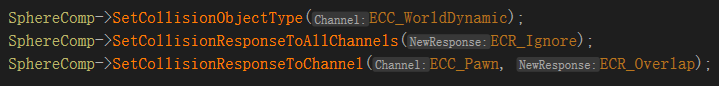
为方便显示，把投射物的重力设为0



摆放一个立方体在场景中，在Detail面板Collision栏中，设置Collision Presets为Custom可以看见碰撞检测相关属性，其全为block（左图）。同理，在投射物蓝图的SphereComp中可以看见全为Overlap（右图）。

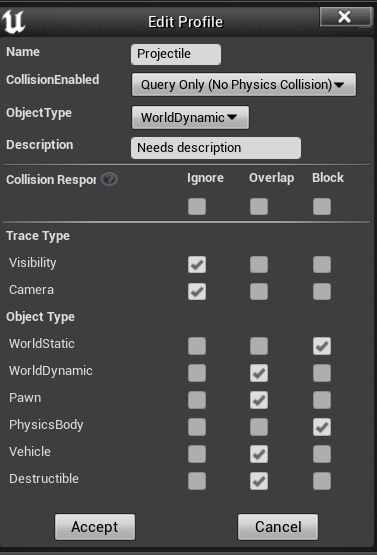
|  |  |
| --- | --- |
|  |  |

当两个物体在游戏场景中相撞时，会触发两个物体勾选的中反应最小的，即按照Ignore、Overlap、Block的顺序检测碰撞反应，例如：若俩物体其中一个为Ignore，则两物碰撞不会有任何反应。因此，投射物会透过Cube，若定义了Overlap事件，会触发该事件。将投射物的WorldStatic勾选为Block，此时投射物会被Cube阻挡。可以才C++中对上面的参数进行修改，避免重复操作：

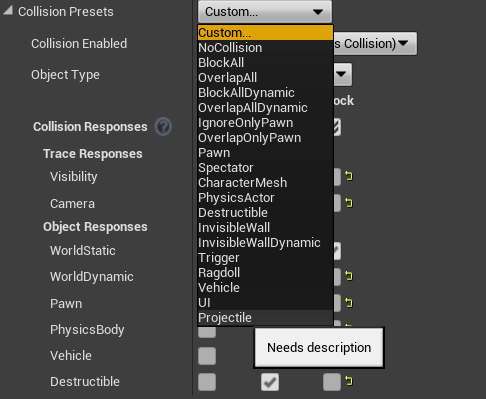


也可以在编辑器中新建配置文件：

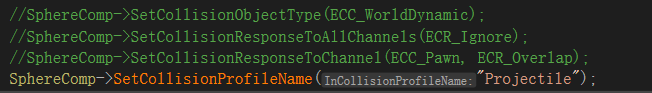
Edit->project Settings->Collision->Preset.New



点击Accept，随后即可在蓝图类中Collision Presets中找到Projectile



当然，在C++中修改更方便，可以把之前的注释掉：

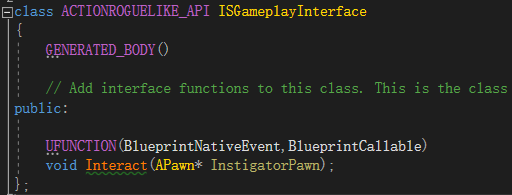


# 交互

## Unreal Interface类

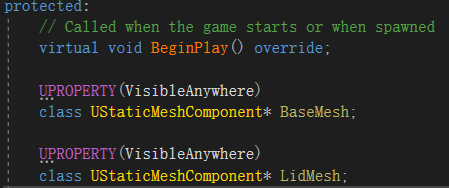
创建C++类，继承自Unreal Interface，命名为SGameplayInterface

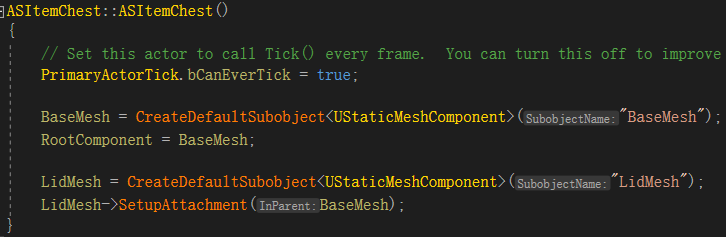
在头文件中声明如下函数，注意不要在对应cpp文件中创建定义。这里加上UFUNCTION宏将此方法暴露给蓝图，因为后续创建的可互动物品可能会是个纯蓝图类，没有C++基类。关于BlueprintNativeEvent和BlueprintImplementableEvent的区别详见：[UE4函数标记BlueprintImplementableEvent和BlueprintNativeEvent区别\_落寞根号七的博客-CSDN博客\_blueprintnativeevent](https://blog.csdn.net/luomogenhaoqi/article/details/89311695)



## 创建可交互物品（宝箱）

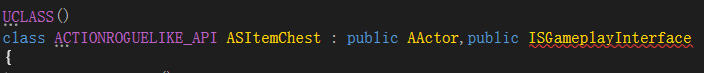
基本Mesh



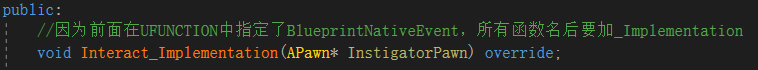


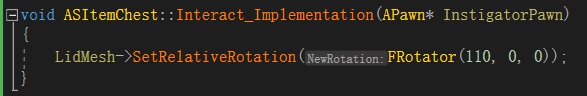
创建蓝图类，并添加具体网格

将交互类设为父类



重写Interact函数





这里稍稍把110封装下，并暴露给编辑器（EditAnywhere）



至此，宝箱就可以动了，但还差人物去触发。

## 碰撞查询

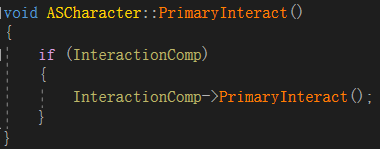
新建C++类继承自ActorComponent，命名为SinteractionComponent，并定义如下函数。



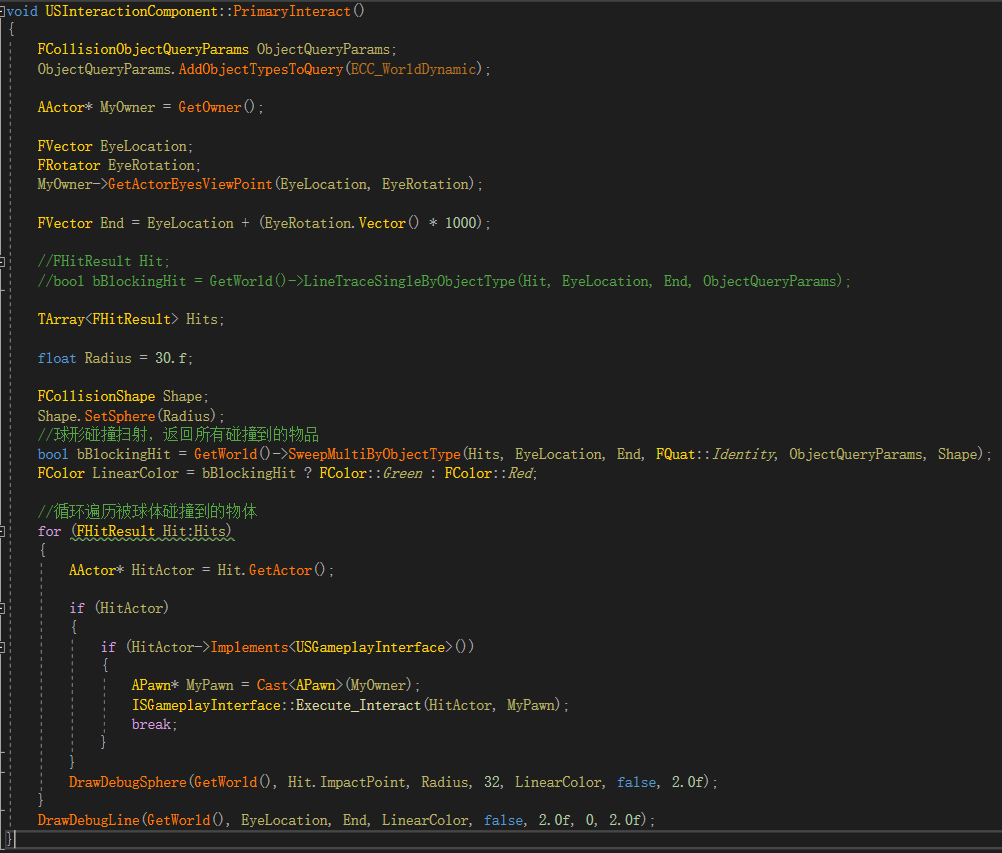
回到Scharacter，添加上述Component，并绑定按键





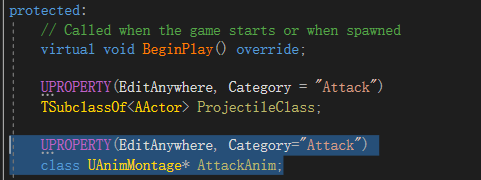


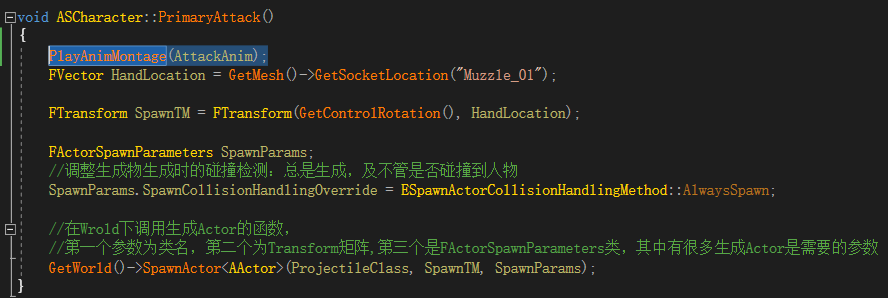
至此已经可以在游戏中实现按键交互了。但仍然有些小问题：查询碰撞的光线太细了，可能只有几个像素，这在之后与较小物品交互时会带来很大的问题，于是在USInteractionComponent::PrimaryInteract函数中做出如下修改：



## 改进攻击（动画和计时器）

为攻击插入动画。在SCharacter头文件和cpp文件中分别添加：



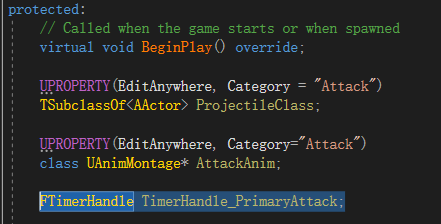


并在蓝图中赋予如下动画值

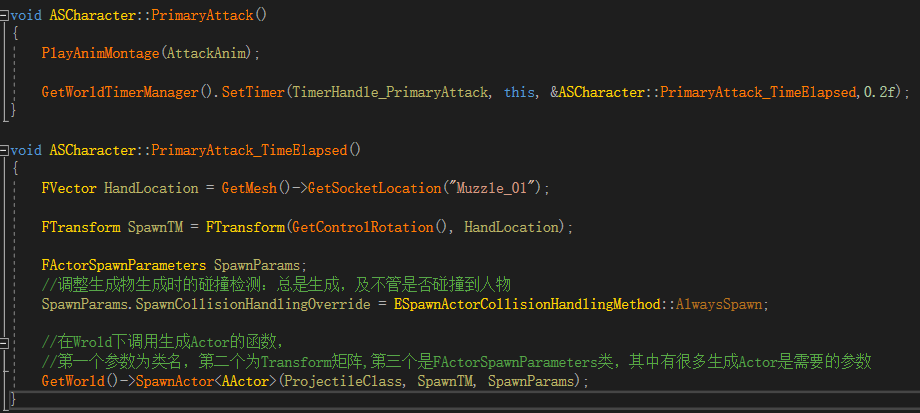


添加动画后攻击任然很奇怪，任然从动画播放前手的位置发射魔法攻击，目前可以为攻击添加一个计时器，当动画播放手抬起时，发射魔法攻击：

首先在头文件中声明一个FTimerHandle变量



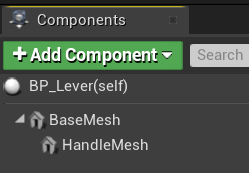
修改.cpp代码，需要重定义个函数，将动作主体放入其中，其中0.2f是延迟时间，可以暴露在蓝图中慢慢调整。



# 蓝图

## 蓝图基础（互动、与C++互通）

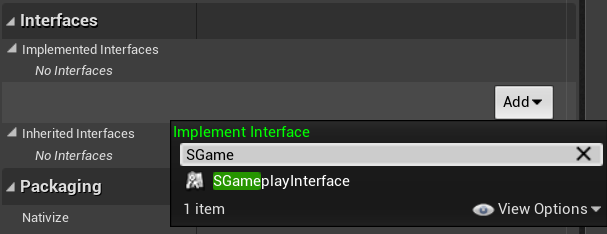
新建Actor蓝图类，进入编辑，左上角可以Add Component。添加两个StaticMesh，拖拽可以调整层级和覆盖初始默认的RootComponent。



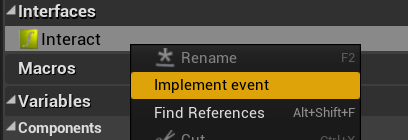
添加Content，并加一点点合适的旋转

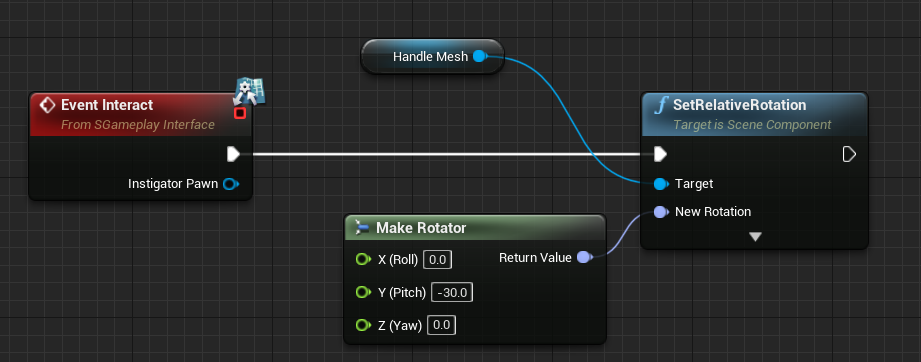
|  |  |
| --- | --- |
|  |  |

点击上方Class Setings，随后在右边detail面板中选中我们写的C++接口类



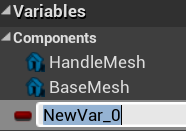
随即左下角的My Blueprint面板会出现Interface，右键点击，然后点Implement event



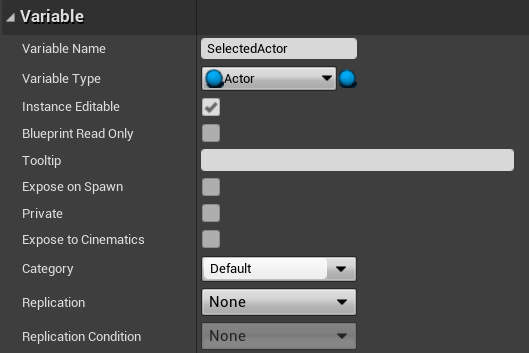


白色为执行线，作为蓝图执行时的顺序，其余颜色为数据线，作为输入和输出的数据传入。变量(Handle Mesh)可以直接从左边拖拽进场景，直接右键可以看到一堆节点选项，而从节点连线出去，只会出现相关联的节点选项。至此开关可以交互且可动了，接下里把开关绑定到宝箱上，我们希望按下开关可以打开宝箱。

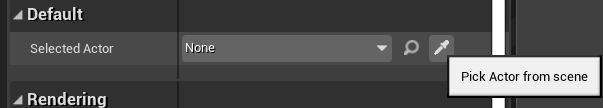
点击左下角面板Variables右边的加号新建一个引用。



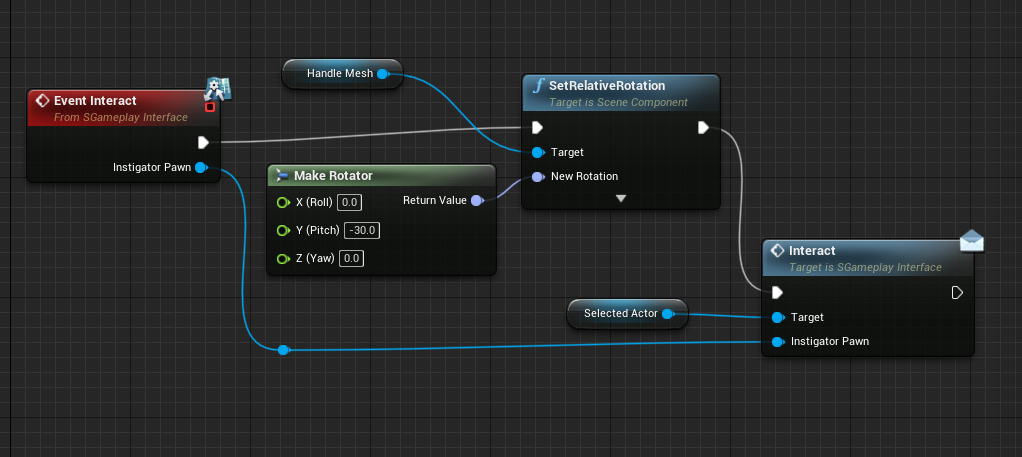
点击后，在右边的Detail面板，设置名字、类型、Instance Editable。



在场景中选中开关的示例，在Detail面板中点击如下按钮



随后点击场景中的一个宝箱即可完成引用变量的绑定，修改蓝图

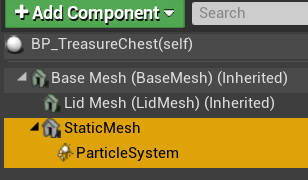


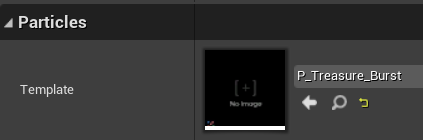
## 使用蓝图完善宝箱（timeline添加动画）

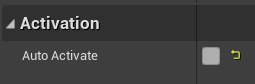
在C++中把宝箱的盖子暴露给蓝图



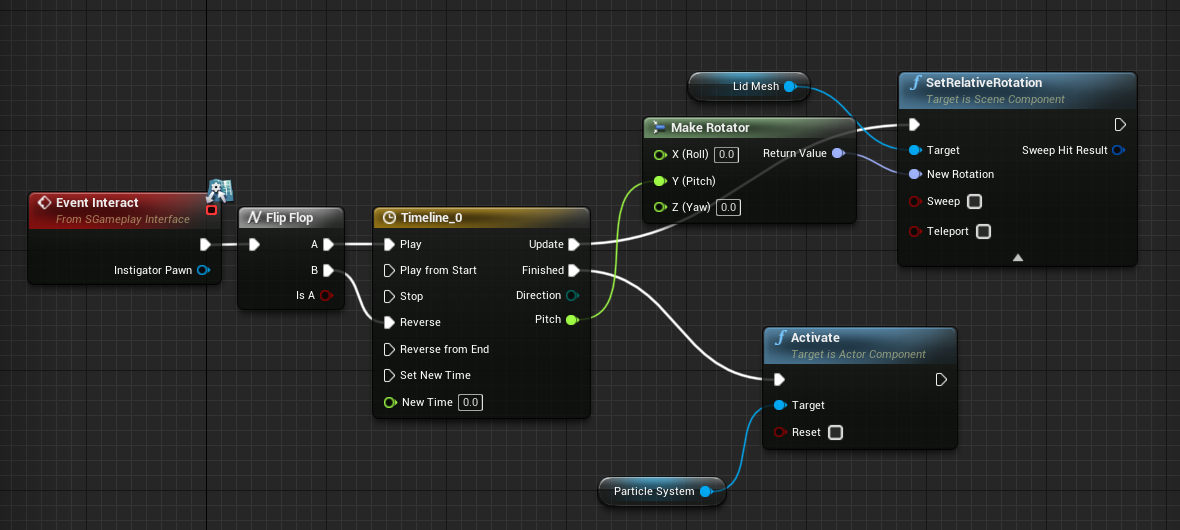
添加如图所示两个Component，并选择合适的Mesh与调整位置，取消勾选粒子系统中的自动激活，否则粒子会在刚开始游戏时就被激活。

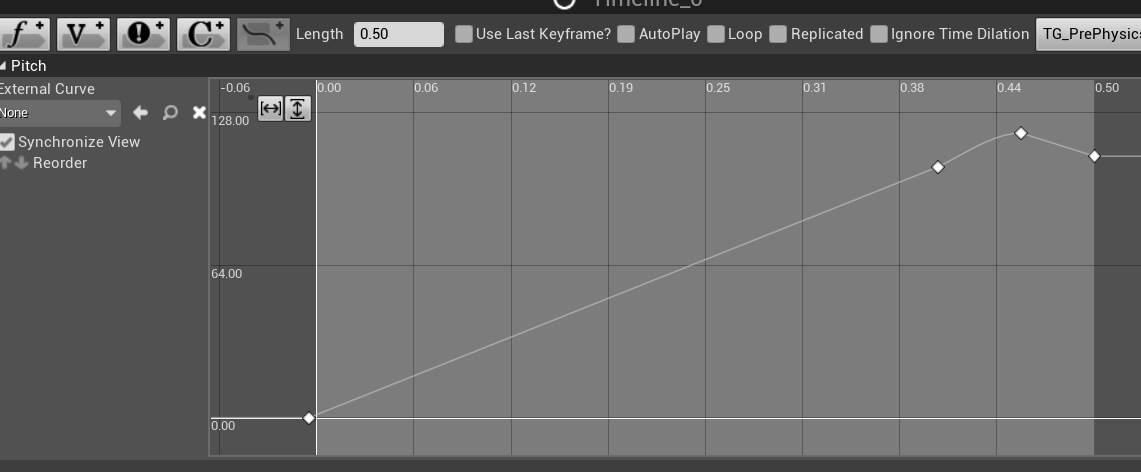




在宝箱的蓝图中，重写Interact：



其中timeline：



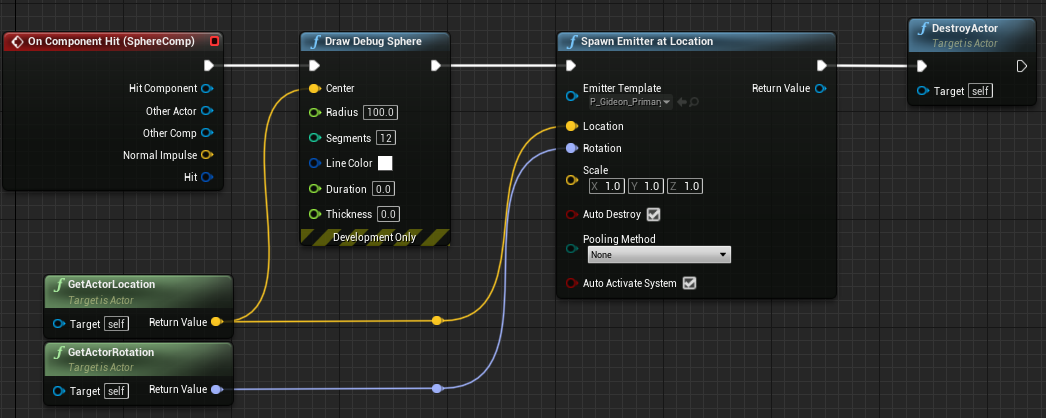
至此实现了可开可关的宝箱，并在打开时添加了粒子特效

## 使用蓝图完善攻击（碰撞爆炸）

修改Collision。

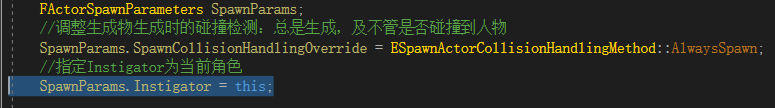


在蓝图中添加酷炫的爆炸效果：

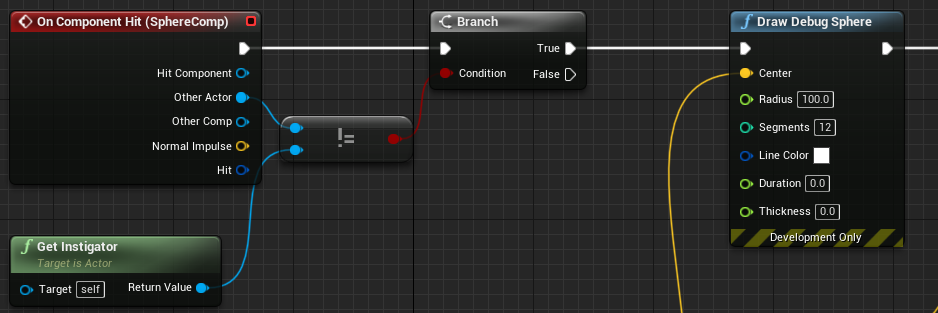


此时人物攻击时，飞弹会被自己的手阻挡并自爆，下面我们将解决该问题。

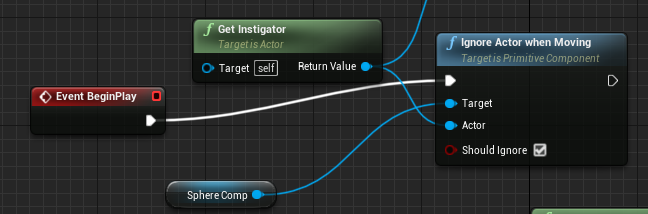
修改C++代码，指定Instigator，在角色Spawn Actor处修改参数：



并在蓝图中添加判断是否撞的是Instigator：

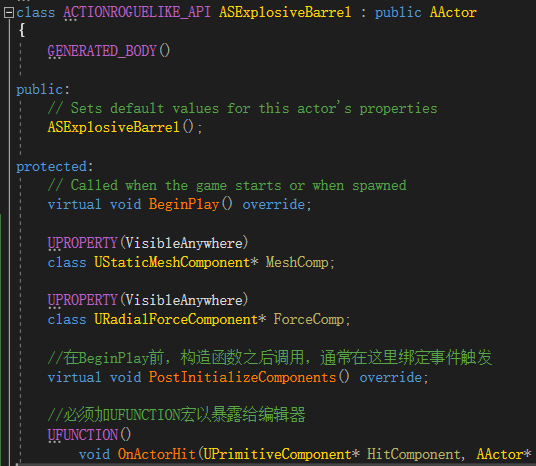


此时飞弹不会刚发射就自爆了，但依然会停下，且会阻挡角色，还需要添加“如果碰撞的是Instigator，则忽视”的逻辑。添加以下蓝图，使飞弹在运动时忽视Instigator，至此，以完善攻击特效



# 爆炸桶与Debug

## 爆炸桶实现

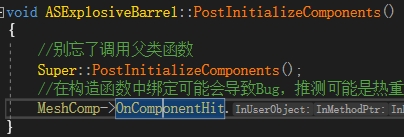




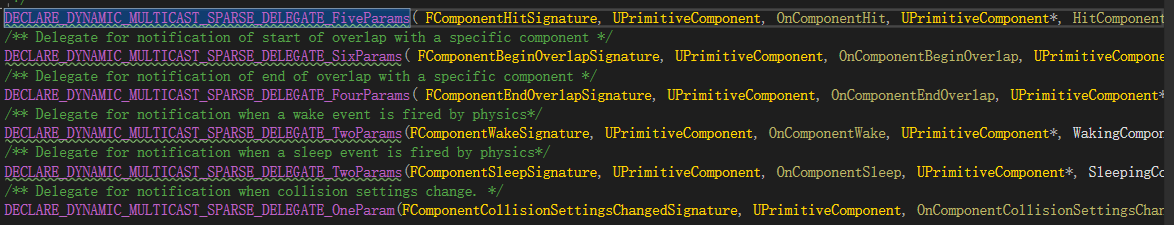


其中OnActorHit的一长串参数可以从此处获得：

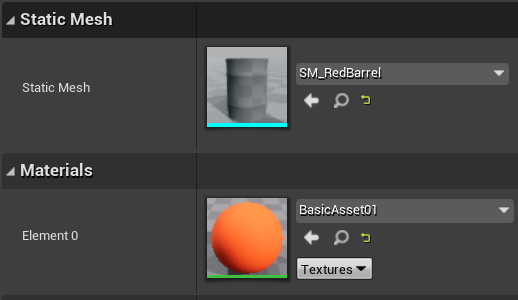
选中OnComponentHit按F12，再选中FComponentHitSignature按F12，即出现需要用到的参数。一般从该宏中的第三个开始复制，粘贴到OnActorHit的括号中，删去该删的逗号即可。





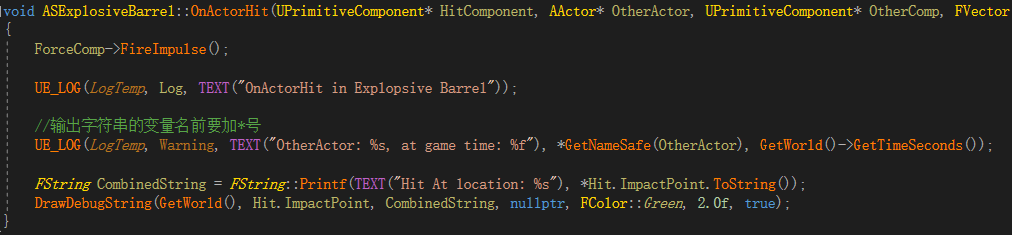


随后即可创建对应蓝图类，选择Mesh与材质：



## 调式

日志输出：

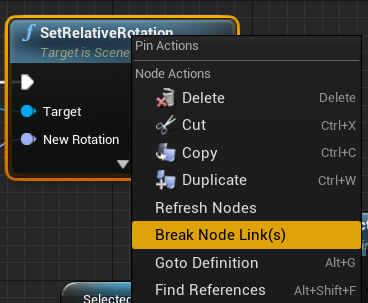


断点调试：

C++

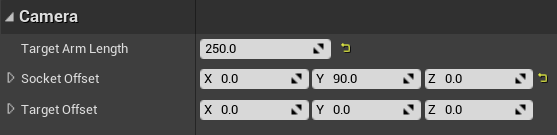
\

蓝图：右键可在节点中添加断点，或使用快捷键F9



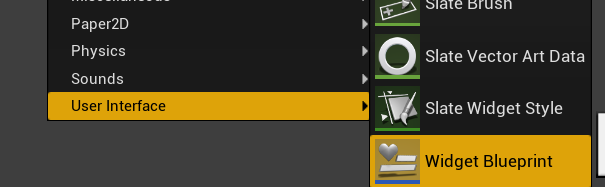
## 相机变换

想把人物往屏幕右边调整，首先想到是直接把摄像机左移，但会失去弹簧臂的一些功能，如不能再保证摄像头不穿墙。因此，应该在弹簧臂上调整摄像机在人物的蓝图类中选中弹簧臂，修改Detail面板：

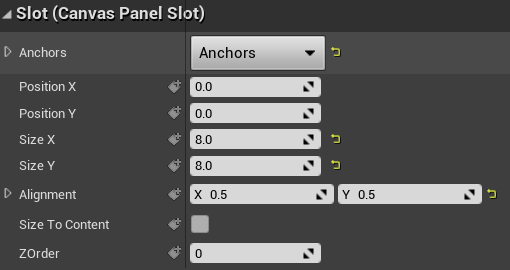


## 准星

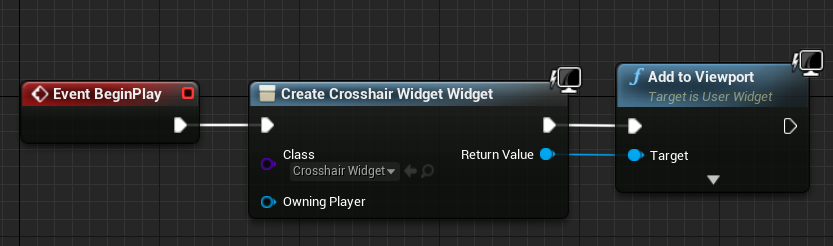
在Content->Action Rogue like目录下新建文件夹命名为UI，在文件夹下可新建Wiegit



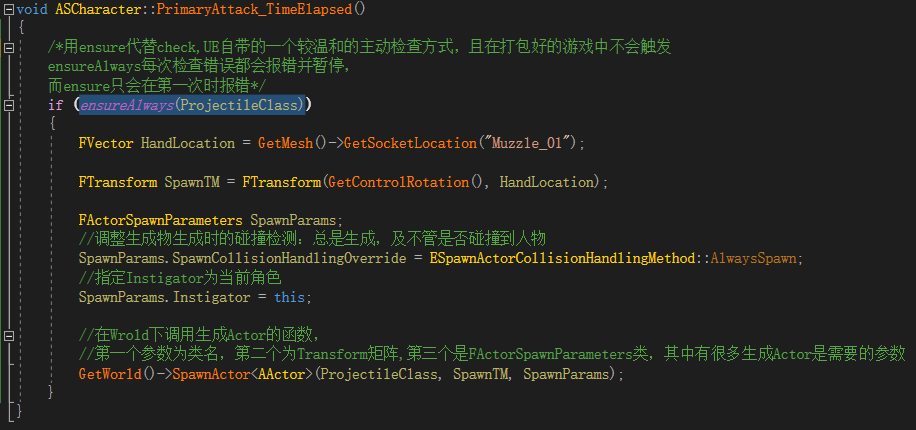
添加Image，在Detail面板中设置位置，其中锚点选中心位置



在人物蓝图中添加事件：开始游戏时创建刚定义的Widget并添加到屏幕上



## 主动类型检查



在VS编辑器中，可修改解决方案配置，当需要Debug代码时，可以将配置修改为DebugGame Editor，编译器会重写编译代码，且不会对代码做过多优化，方便从而添加断点等Debug手段。然后会降低编译速度，由此日常中选择Development Editor，当需要Debug时才切换到DebugGame Editor。

