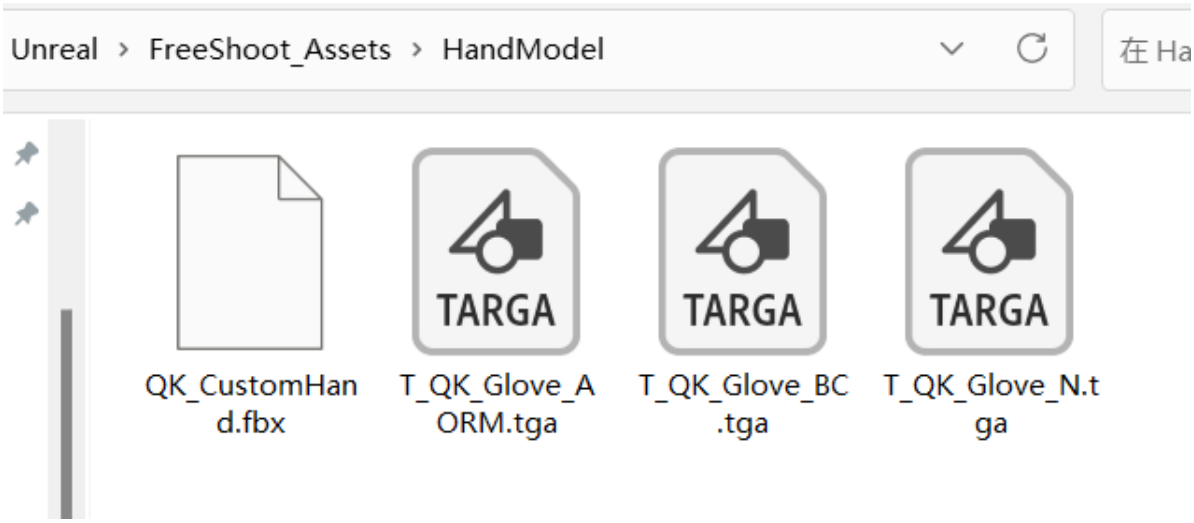
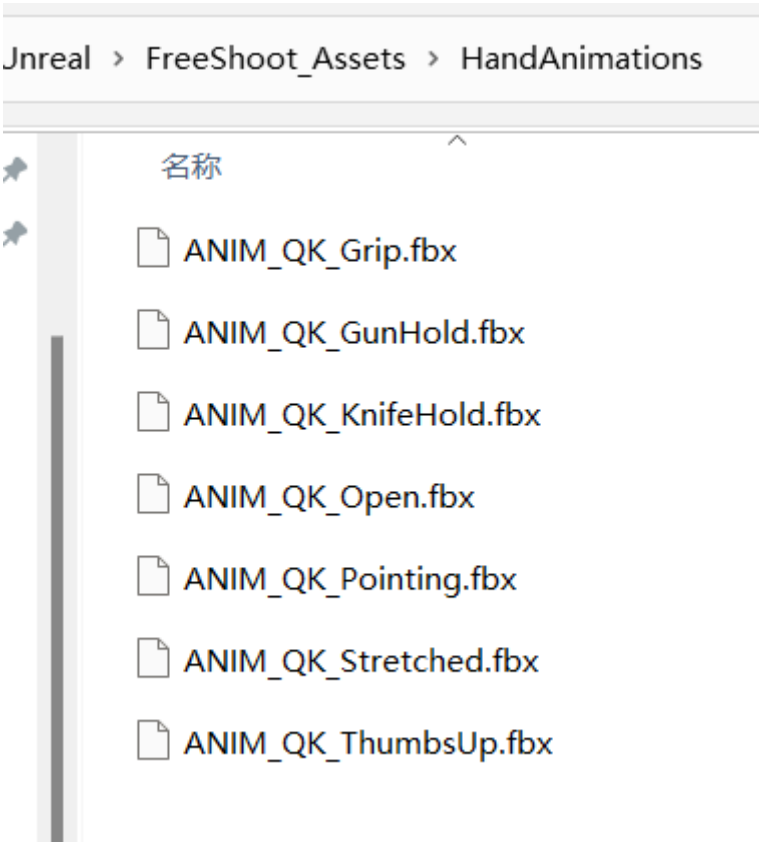
手部模型

导入模型

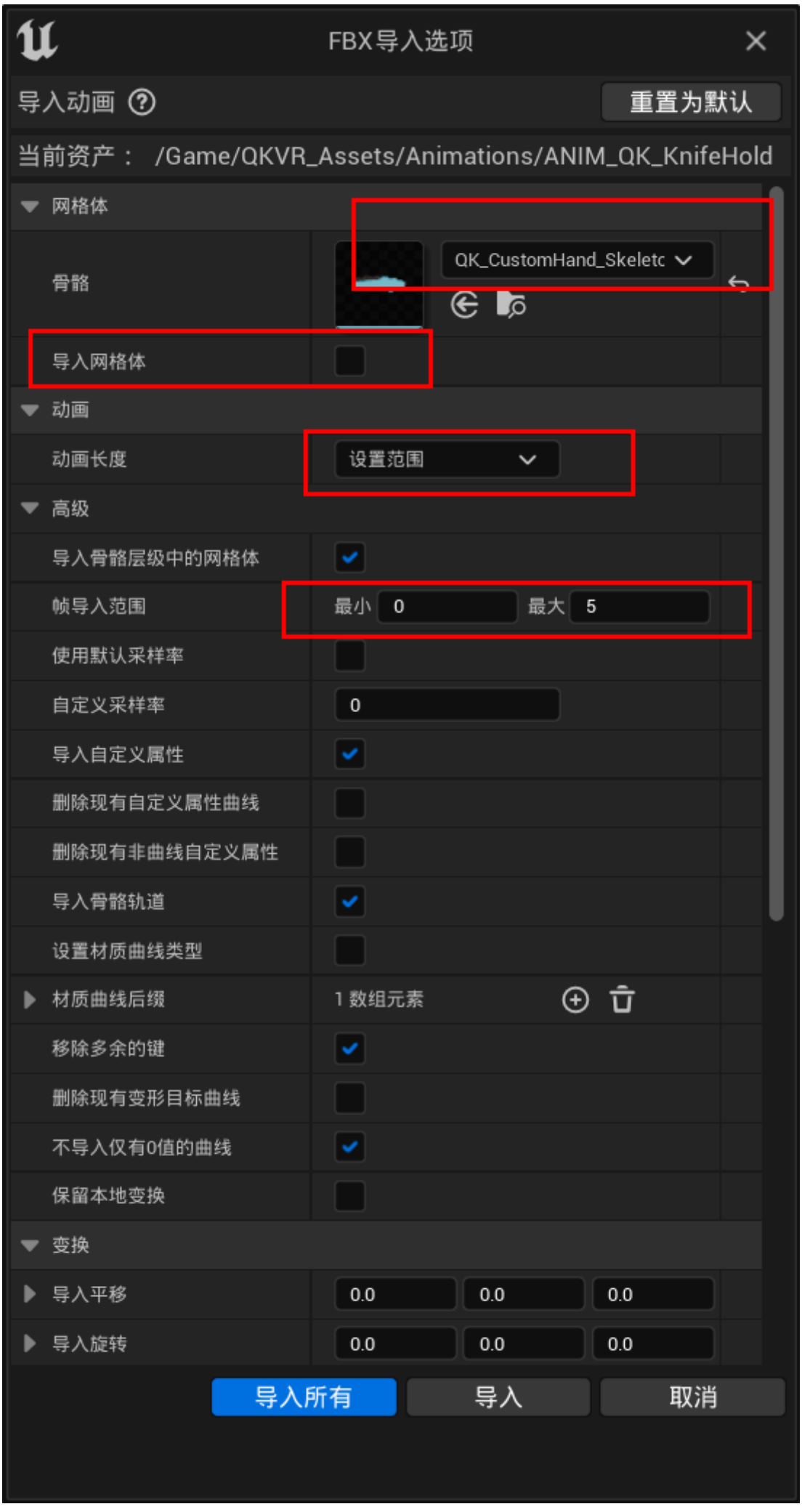
先创建一个Assets文件夹，导入手部模型，直接导入所有



再创建一个Animations文件夹，导入动画

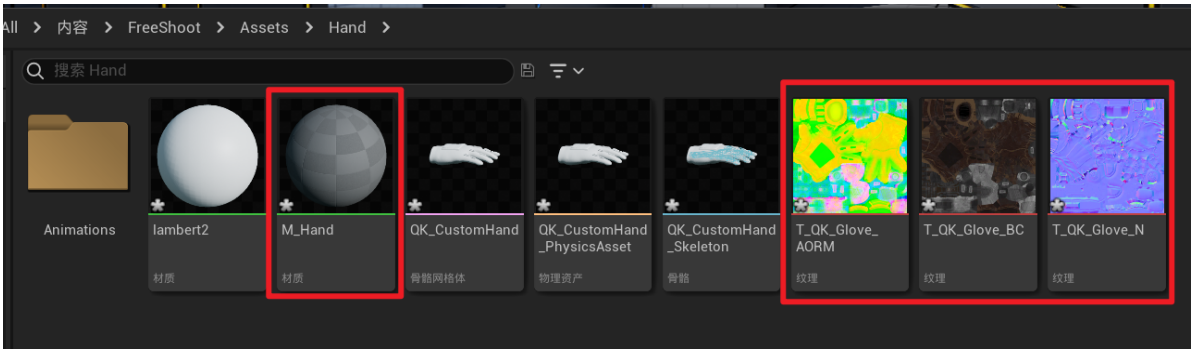


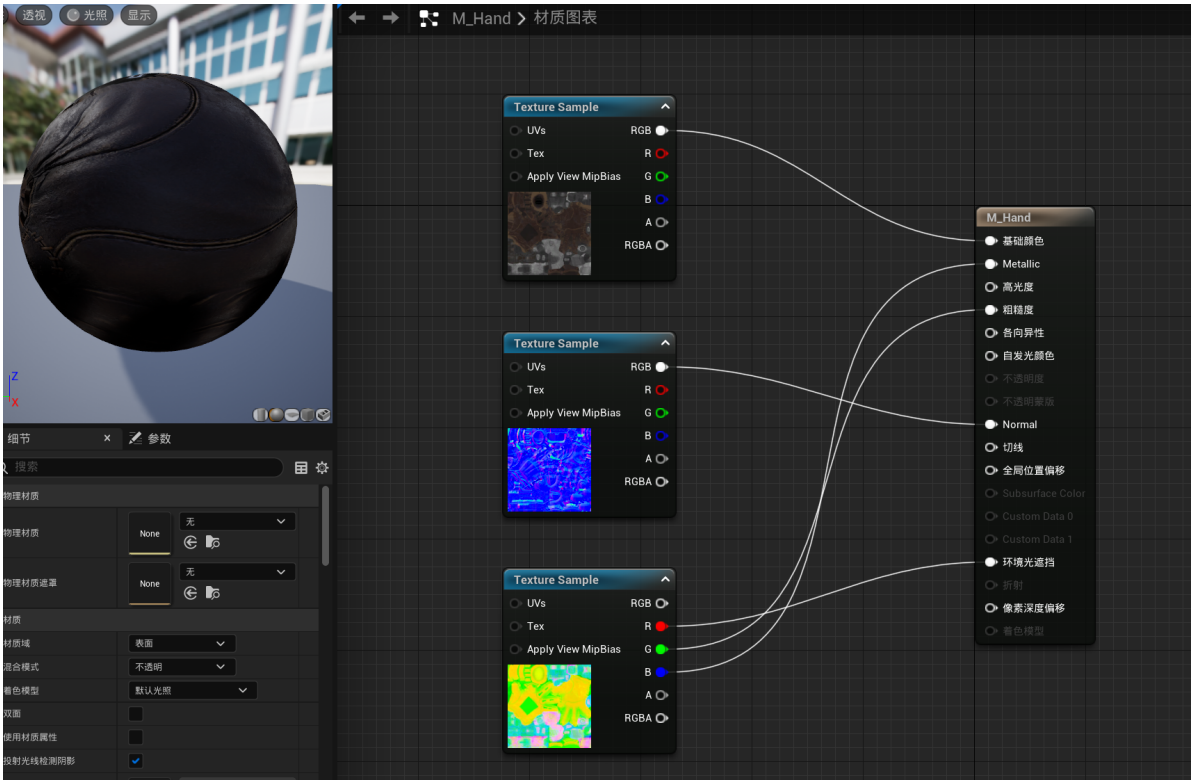
注意这里的导入选项


FBX导入选项
导入动画
重置为默认
当前资产： /Game/QKVR_Assets/Animations/ANIM_QK_KnifeHold
网格体
骨骼
QK_CustomHand_Skeletc
导入网格体
动画
动画长度
设置范围
高级
导入骨骼层级中的网格体
帧导入范围
最小 0
最大 5
使用默认采样率
自定义采样率
0
导入自定义属性
删除现有自定义属性曲线
删除现有非曲线自定义属性
导入骨骼轨道
设置材质曲线类型
材质曲线后缀
1 数组元素
移除多余的键
删除现有变形目标曲线
不导入仅有0值的曲线
保留本地变换
变换
导入平移
0.0
0.0
0.0
导入旋转
0.0
0.0
0.0
导入所有
导入
取消

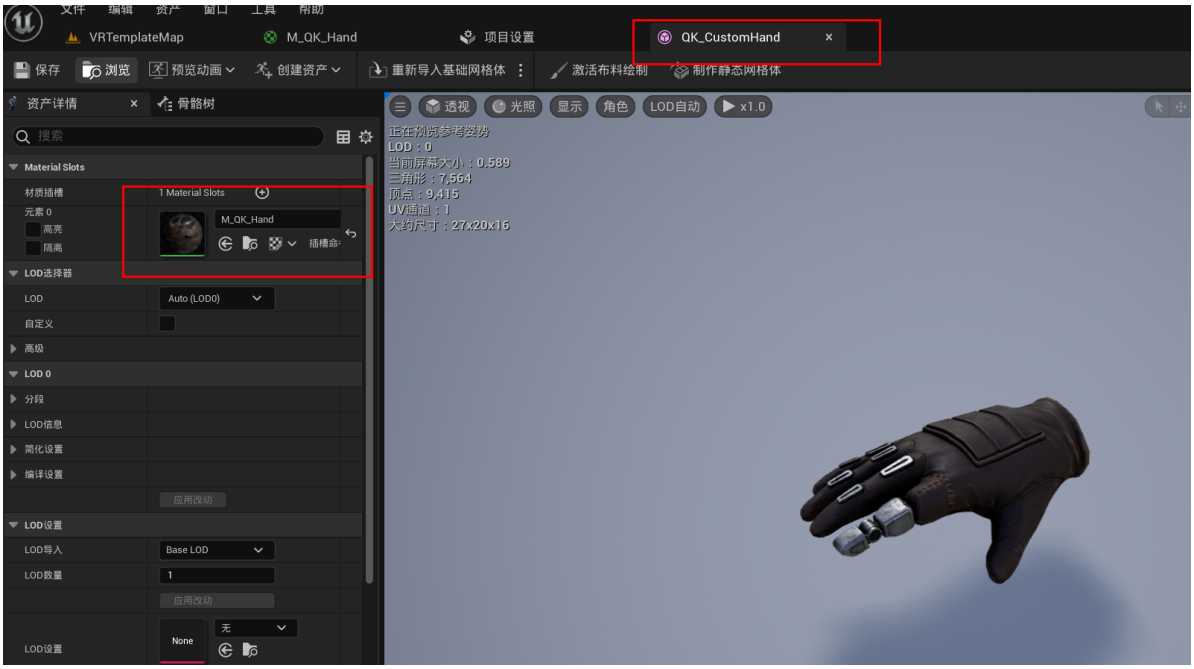
添加材质

创建一个材质M_Hand，导入的三个贴图放到材质里，并按照如下排布





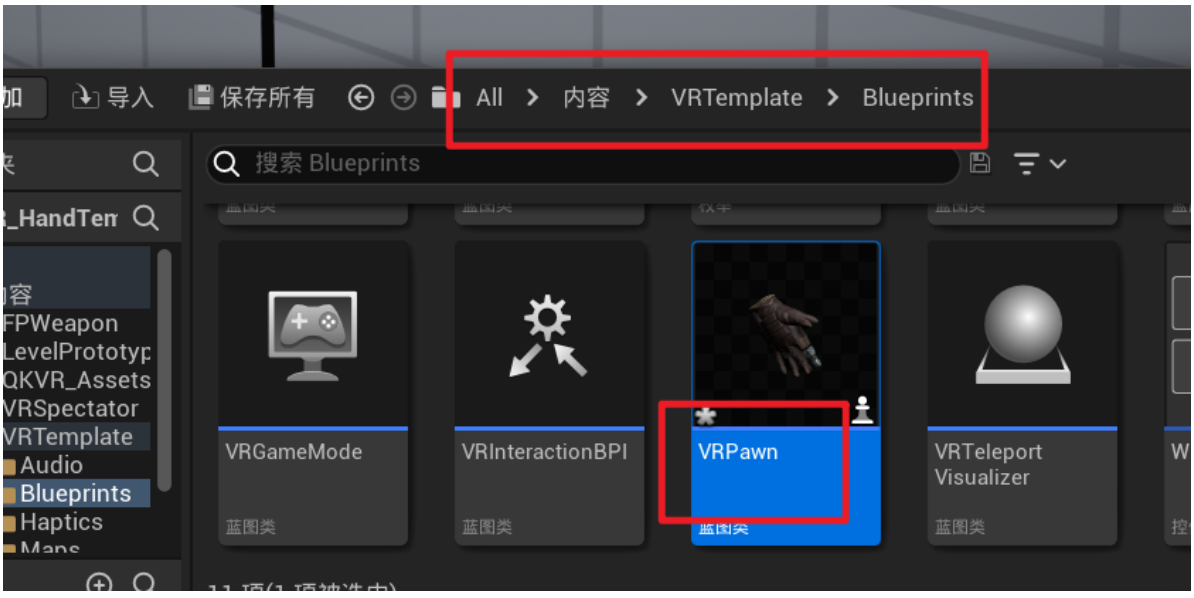
把这个材质应用到手上



更换模型

添加两个操作映射

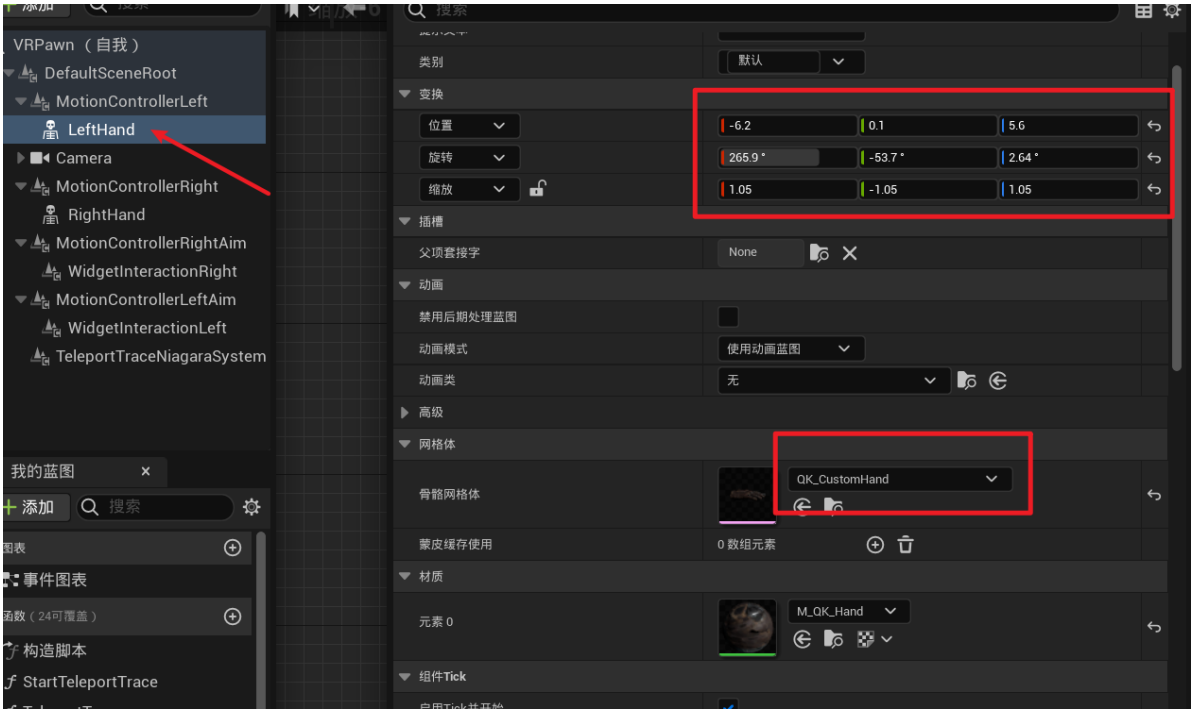
接下来开始调试手部模型，打开VRPawn



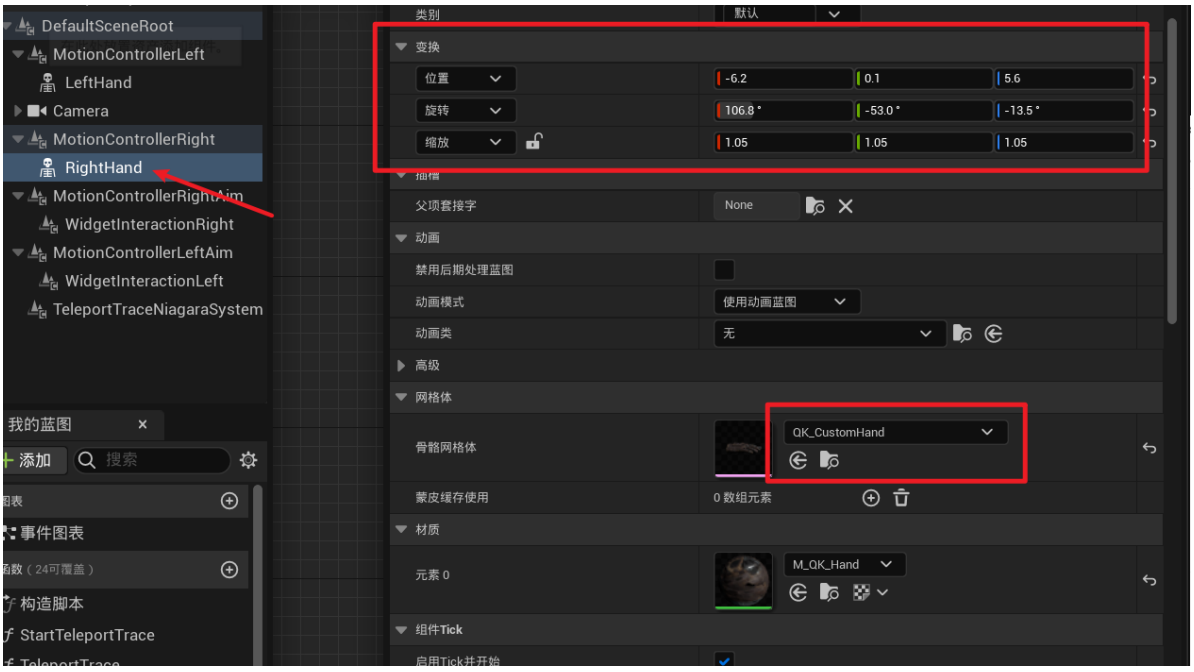
选中MotionControllerLeft和Right，添加骨骼网格体组件，并对其命名

分别对左右网格体进行如下设置

```
(X=-6.200000,Y=0.100000,Z=5.600000)
(Pitch=-53.700000,Yaw=2.640000,Roll=265.900000)
(X=1.050000,Y=-1.050000,Z=1.050000)
```



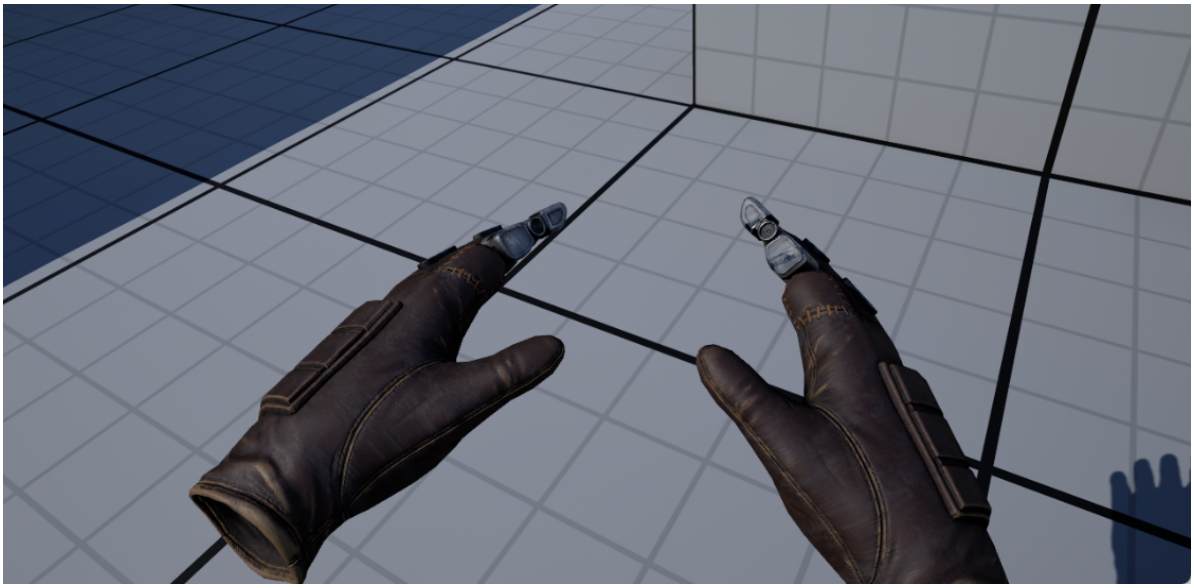

```
(X=-6.200000,Y=0.100000,Z=5.600000)
(Pitch=-53.000000,Yaw=-13.500000,Roll=106.800000)
(X=1.050000,Y=1.050000,Z=1.050000)
```



关闭HMD的可视

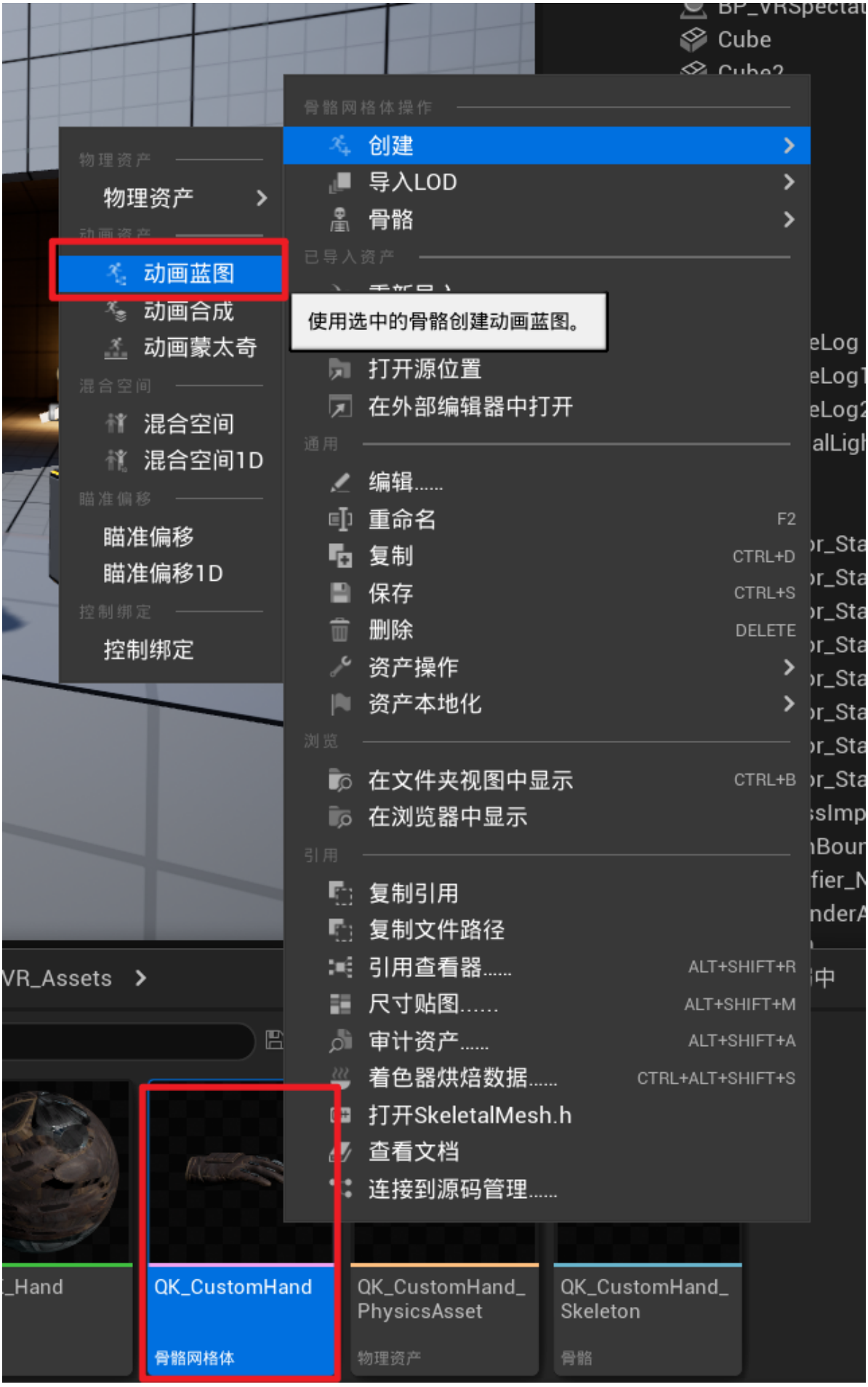
禁止MotionControllerLeft和Right显示设备模型

然后编译保存，你将会发现你的控制器变成了手部模型，效果如下

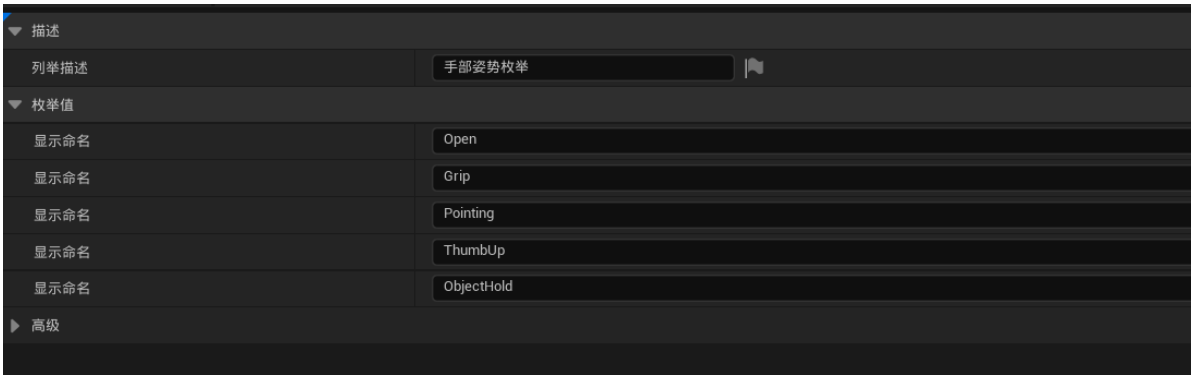


手部动画

接下来开始创建动画，创建动画蓝图，并命名为BP_HandAnim

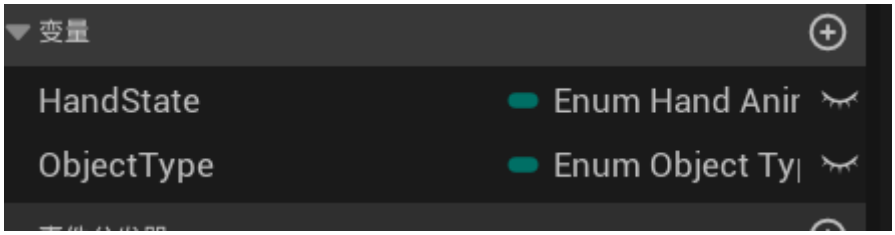


创建一个枚举，命名为Enum_HandAnim

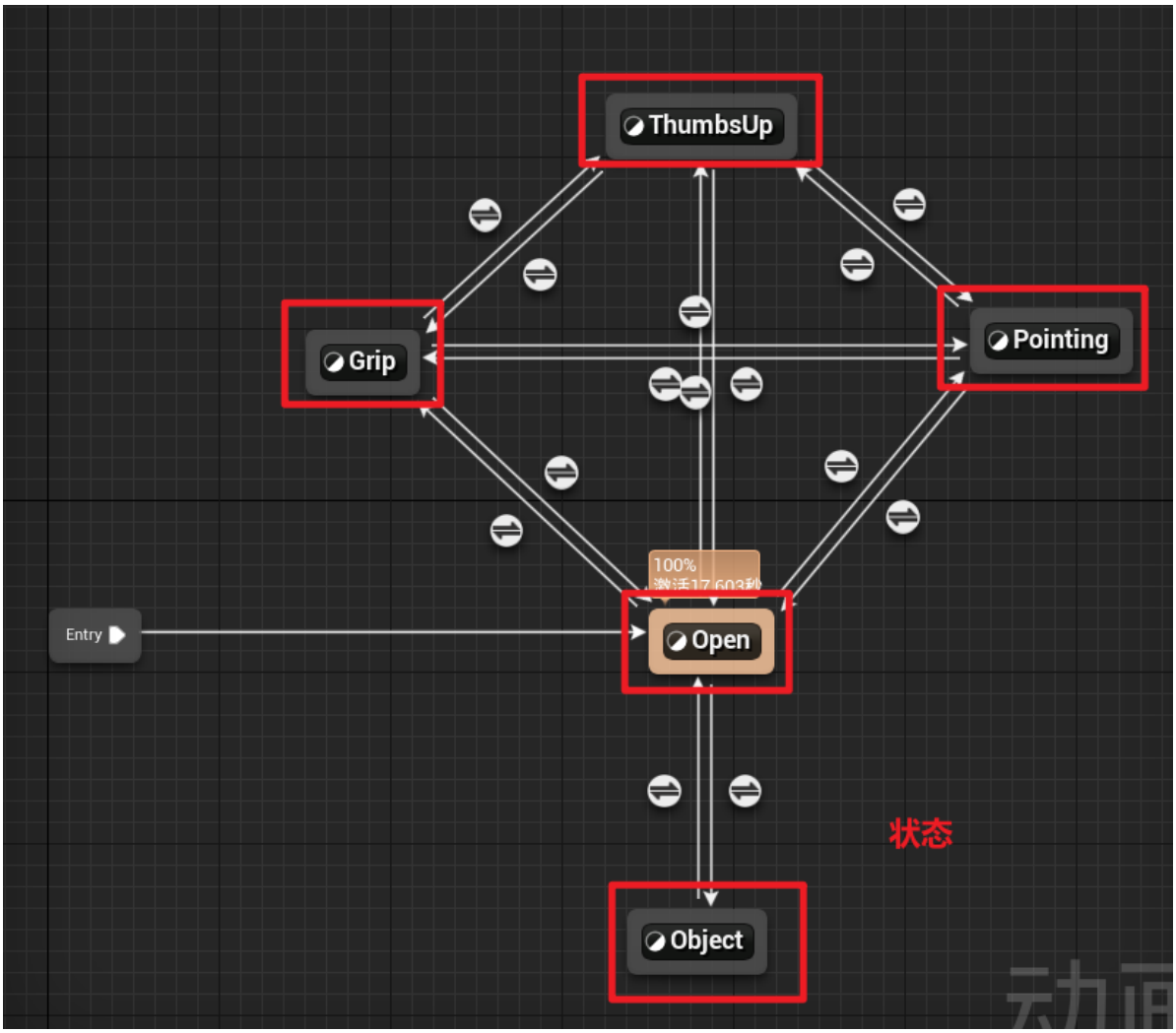


再创建一个枚举，命名为Enum_ObjectType

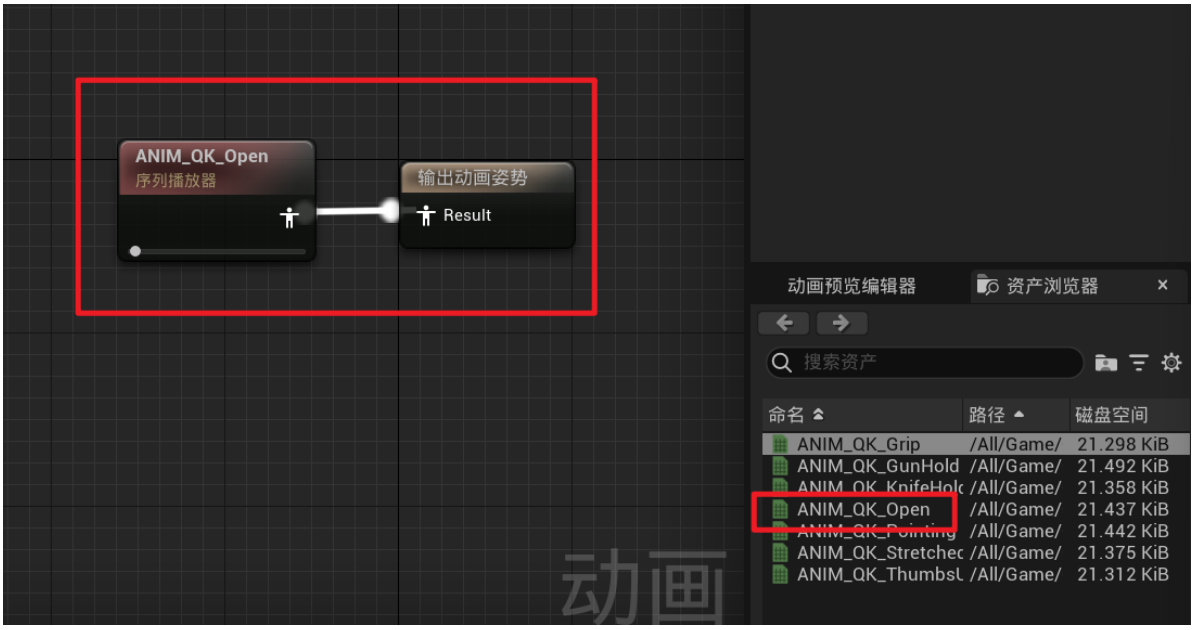
双击进入BP_HandAnim，创建两个变量，注意命名和变量类型



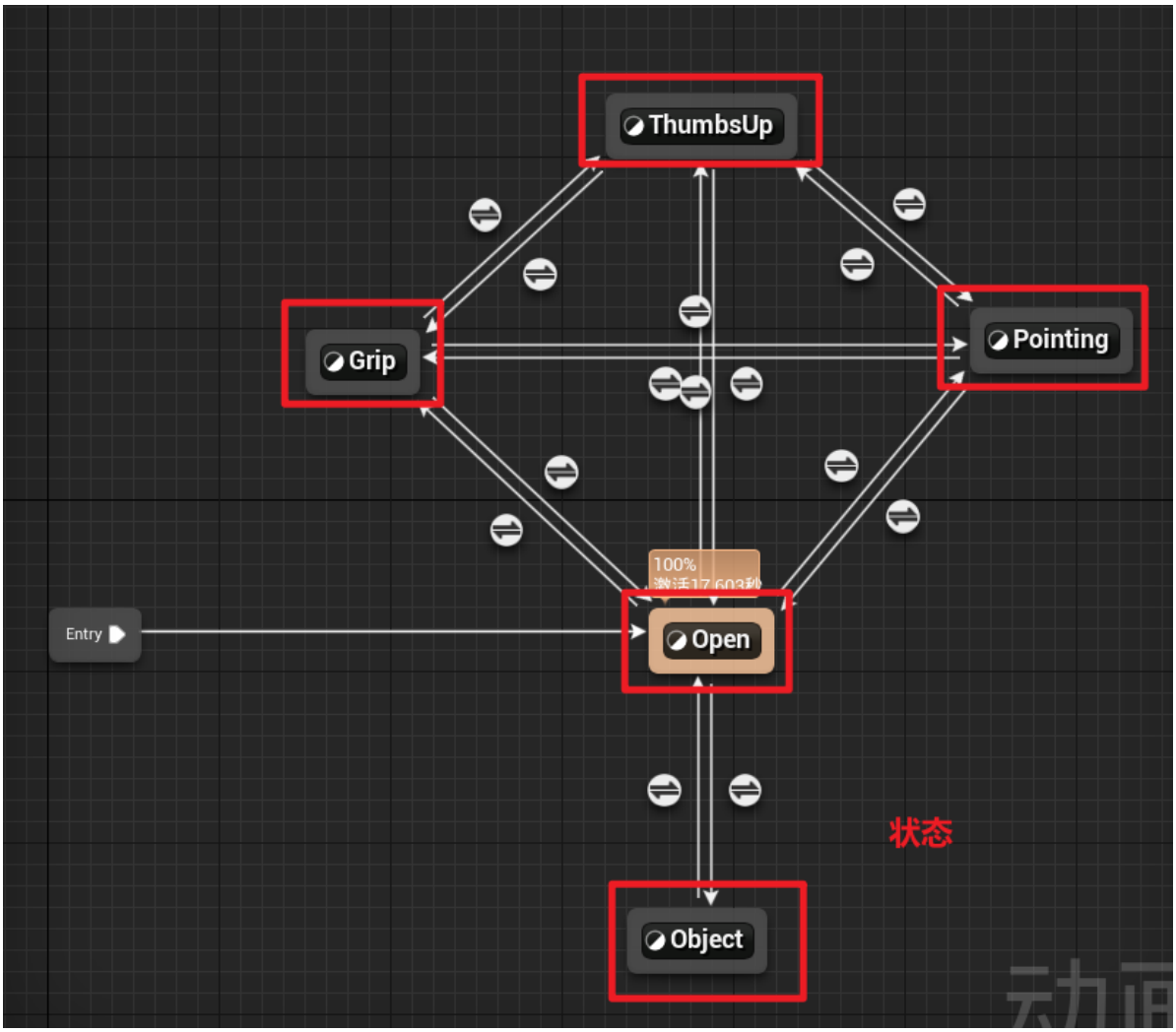
配置状态和状态转换规则

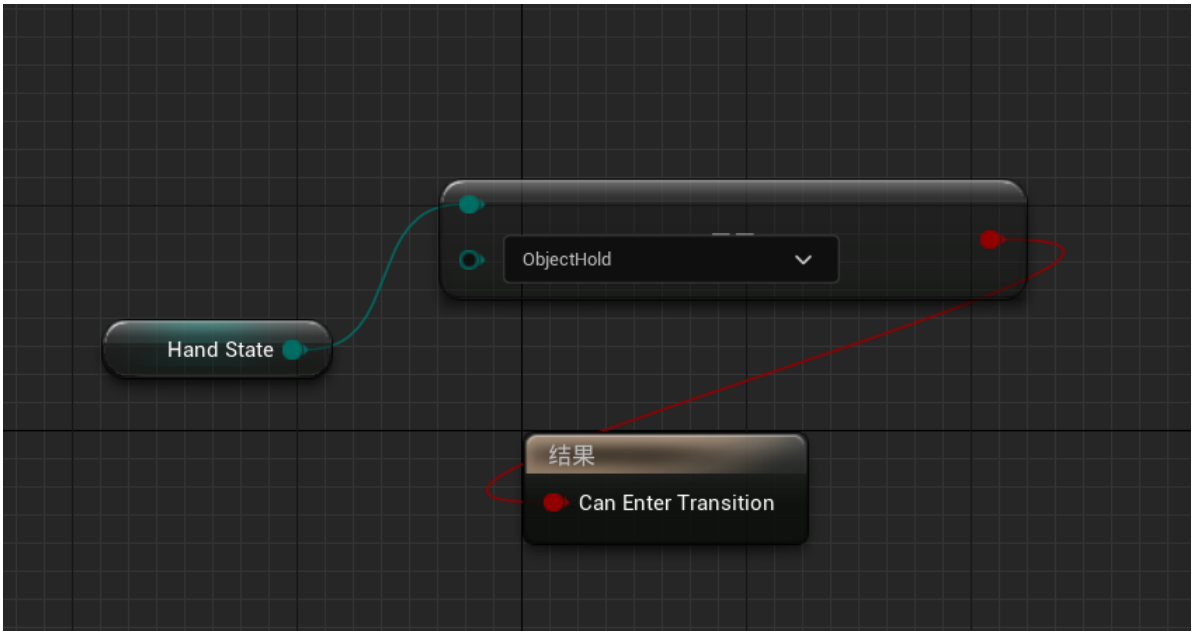


依次双击每个状态机，配置相应资产，以下以Open为例

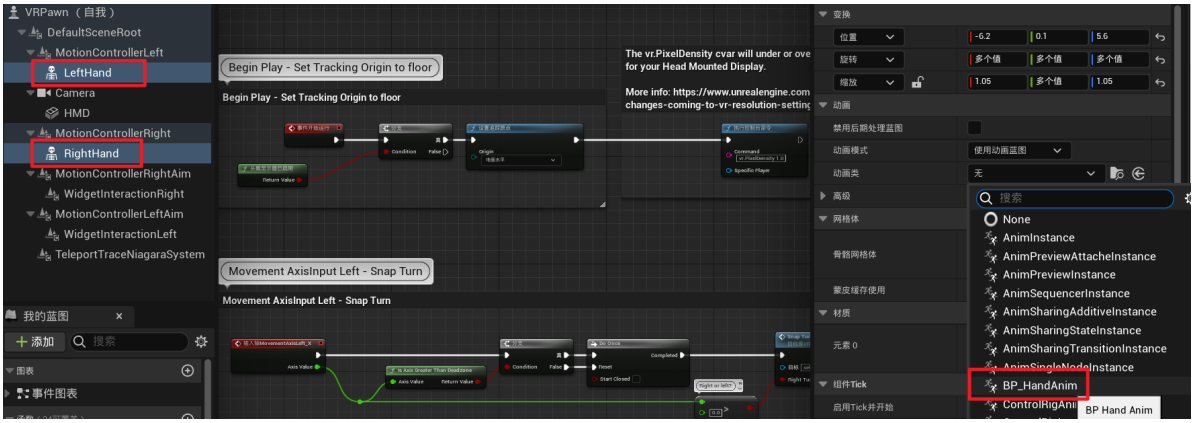


配置每个状态之间的过渡规则，这里以Open到Grip的状态过渡规则为例，双击那个双箭头的小图标，然后添加如下规则

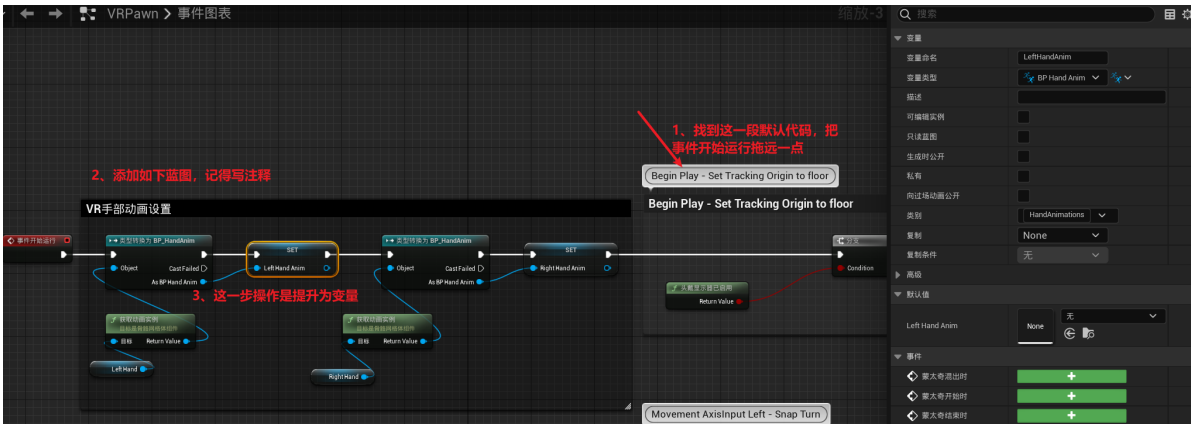




接下来双击VRPawn，设置左右手的动画类



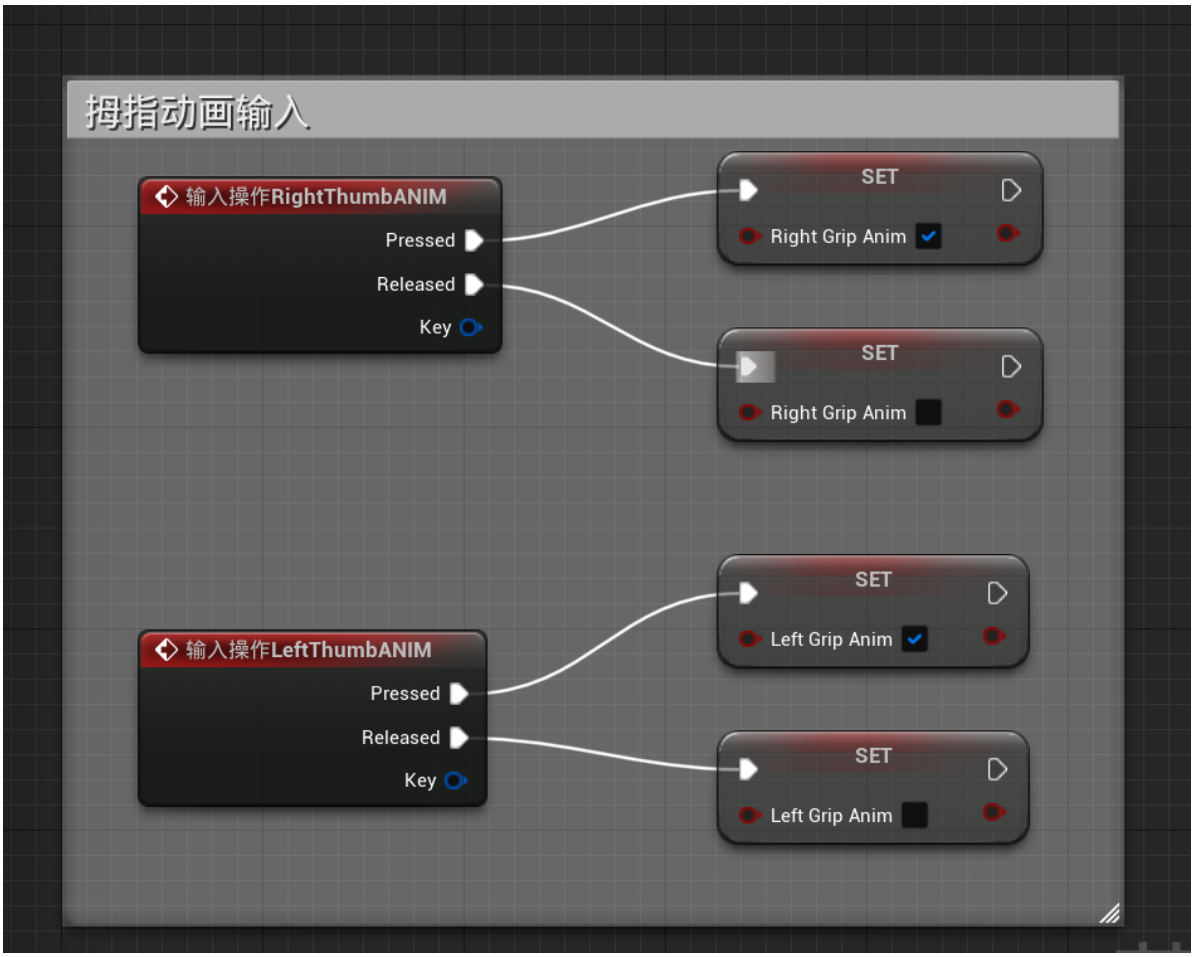
给VRPawn设置一些蓝图（注意变量提升之后要重命名和更改类别为HandAnimations，是类别不是变量类型）

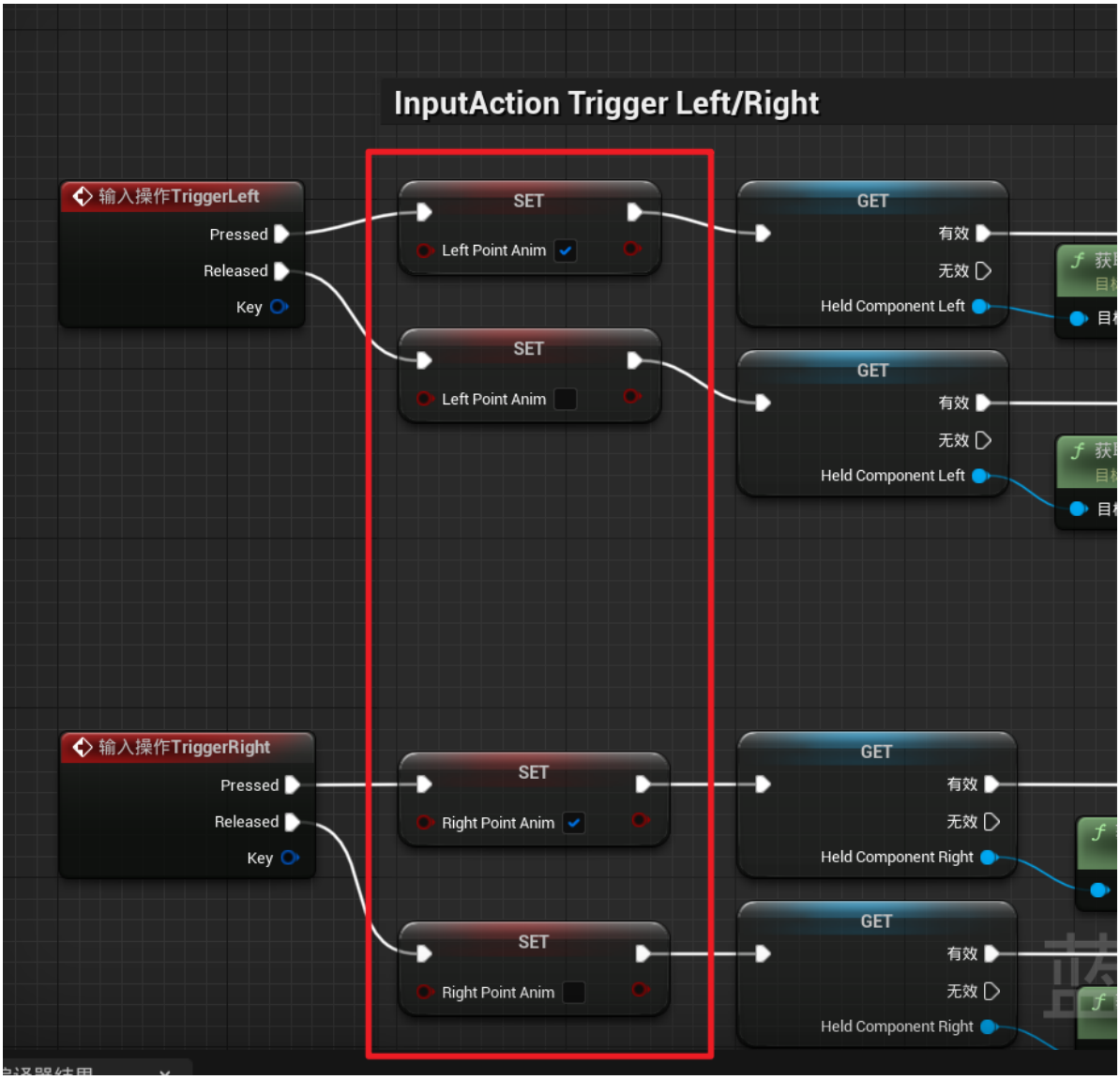


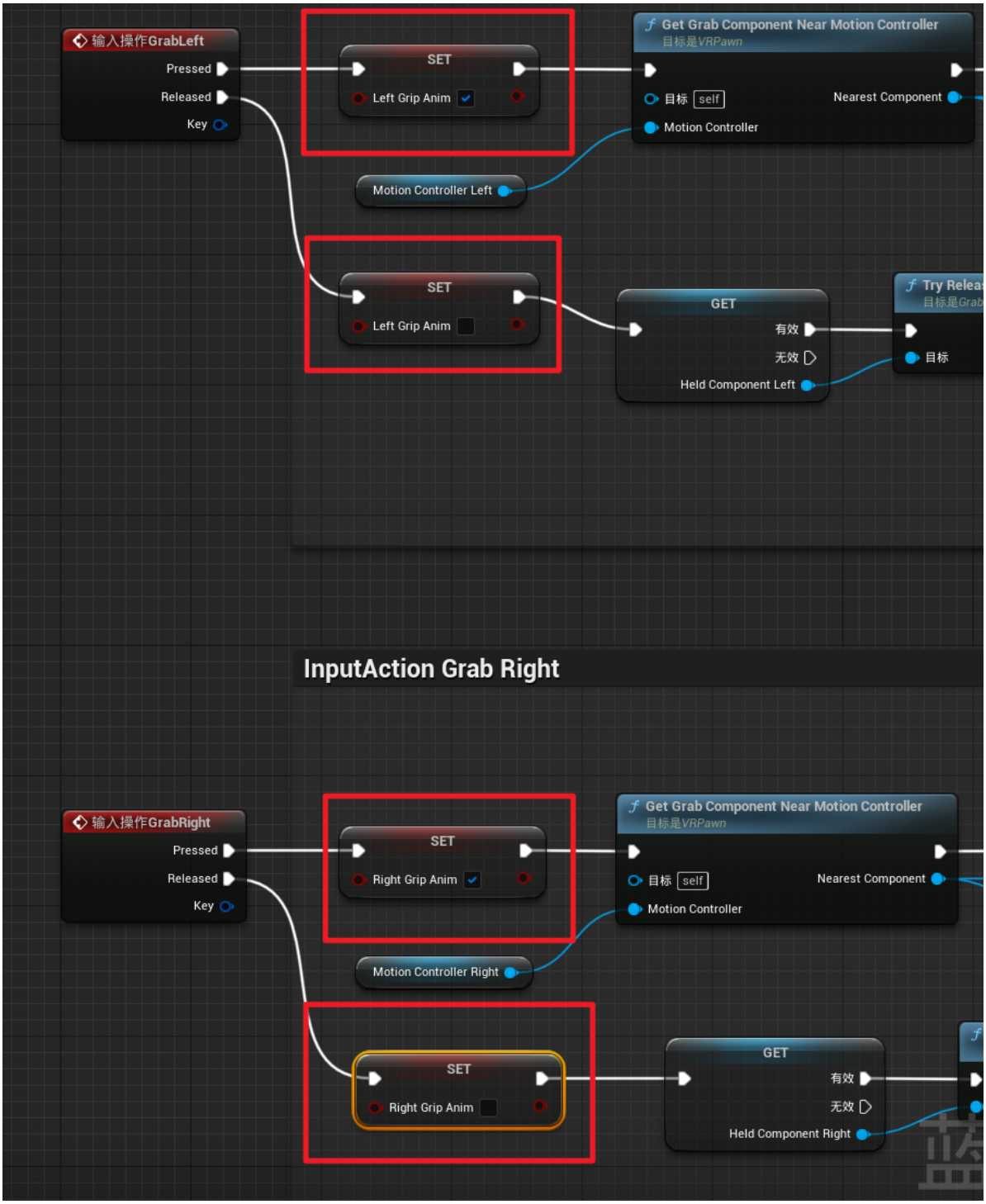
创建下面6个布尔变量


RightGripAnim 布尔
RightPointAnim 布尔
RightThumbAnim 布尔
LeftGripAnim 布尔
LeftPointAnim 布尔
LeftThumbAnim 布尔

添加如下蓝图

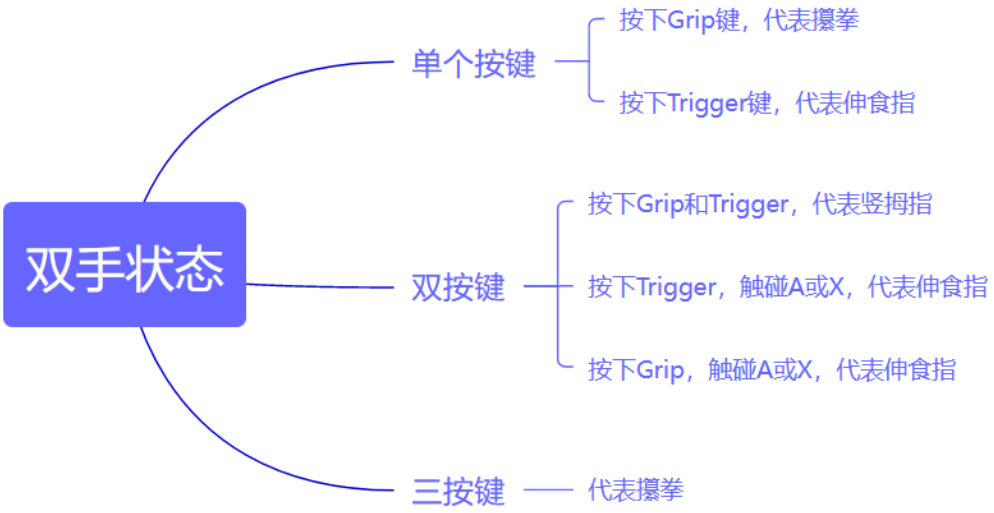

拇指动画输入
输入操作RightThumbANIM
Pressed
Released
Key
SET
Right Grip Anim
SET
Right Grip Anim
输入操作LeftThumbANIM
Pressed
Released
Key
SET
Left Grip Anim
SET
Left Grip Anim


InputAction Trigger Left/Right
输入操作TriggerLeft
Pressed
Released
Key
SET
Left Point Anim
SET
Left Point Anim
GET
有效
无效
Held Component Left
GET
有效
无效
Held Component Left
输入操作TriggerRight
Pressed
Released
Key
SET
Right Point Anim
SET
Right Point Anim
GET
有效
无效
Held Component Right
GET
有效
无效
Held Component Right

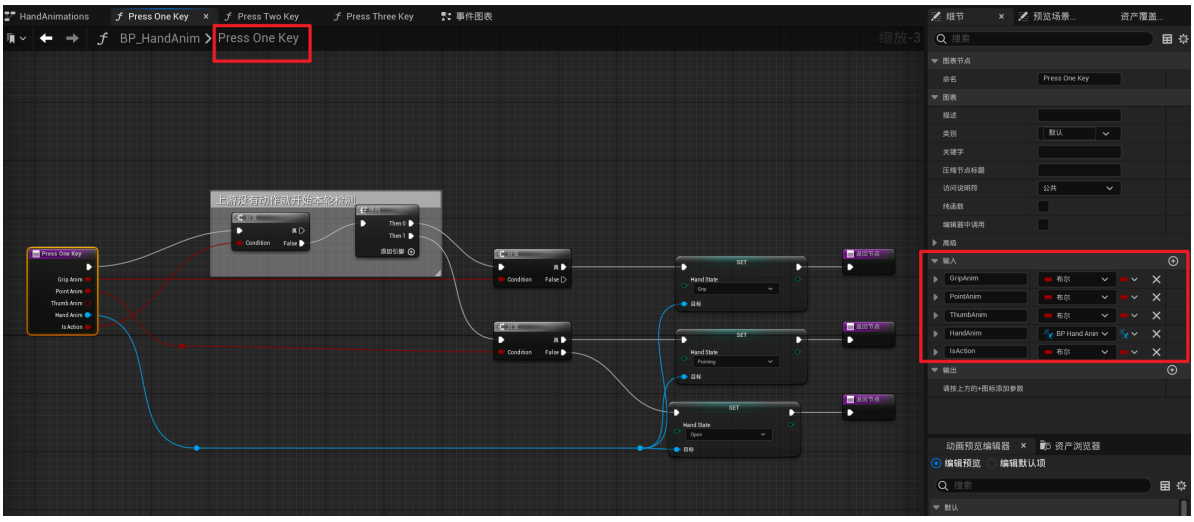


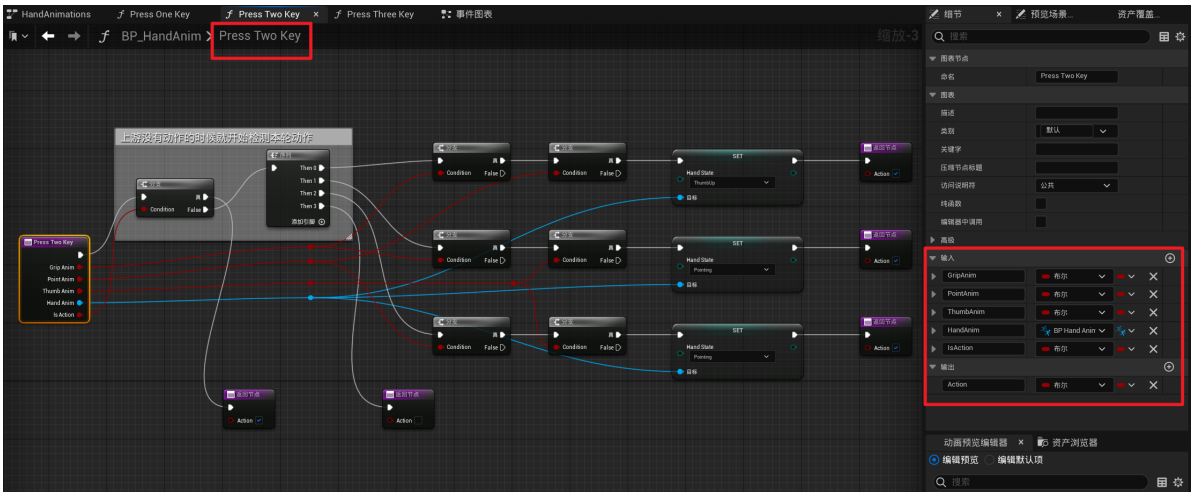
最后打开BP_HandAnim，我们来写一些状态转换的蓝图控制代码

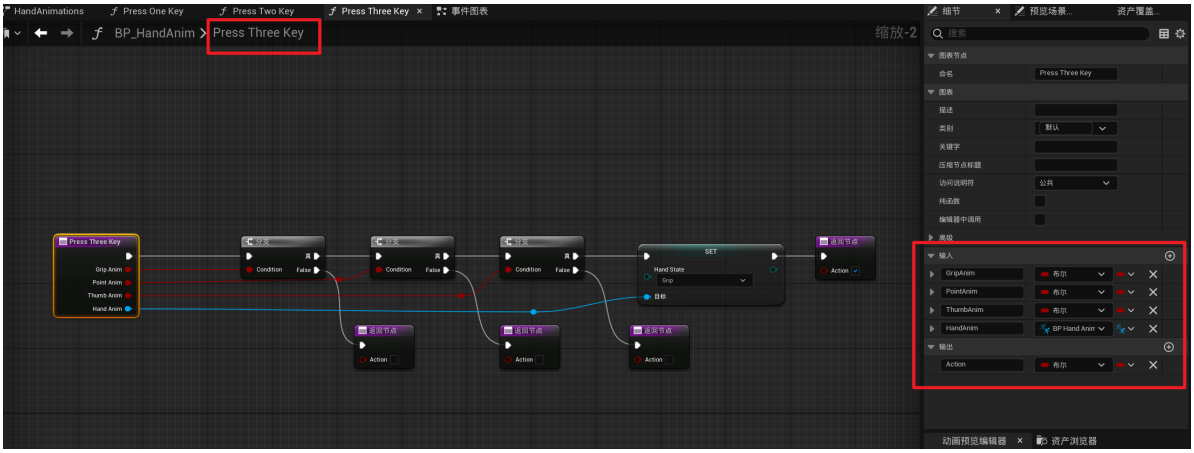
具体转换规则定义如下



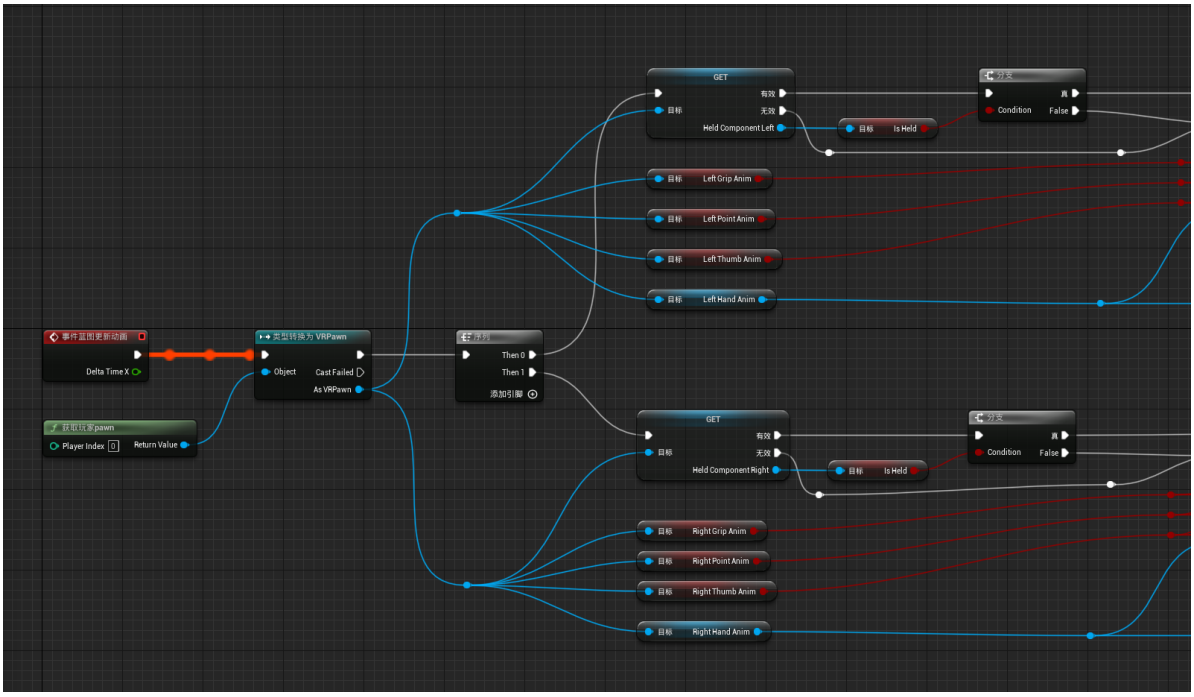
先转换类型，创建三个函数

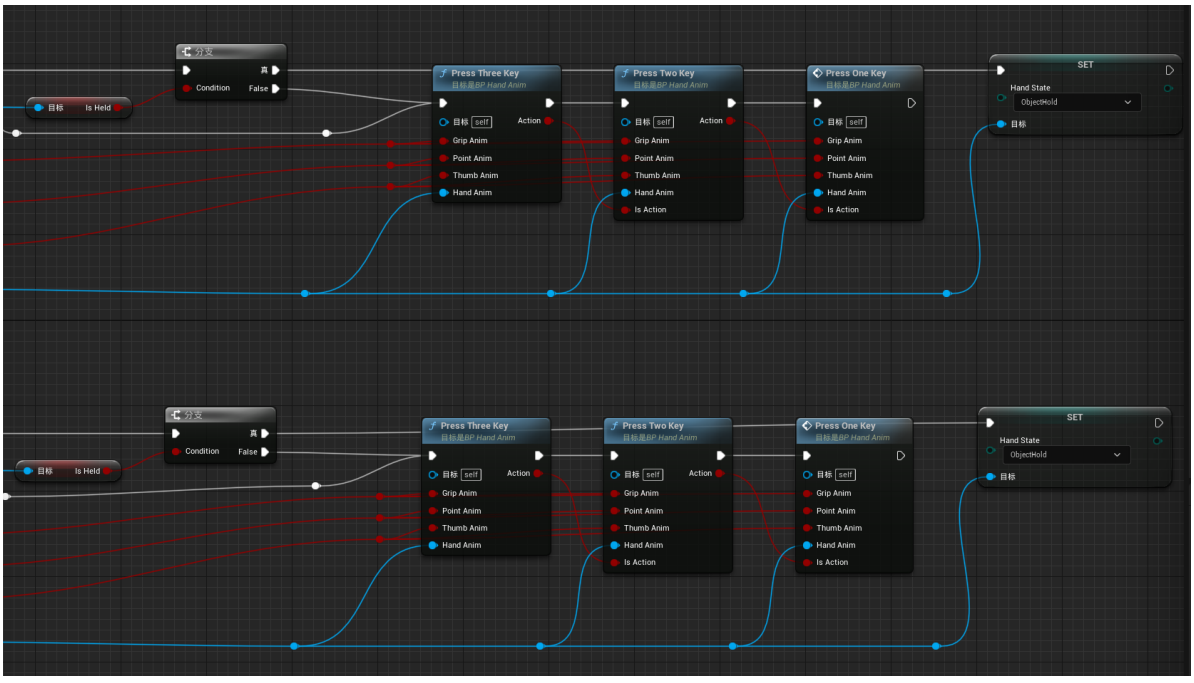






然后写事件图表（蓝图较长，各位请自行拼接）

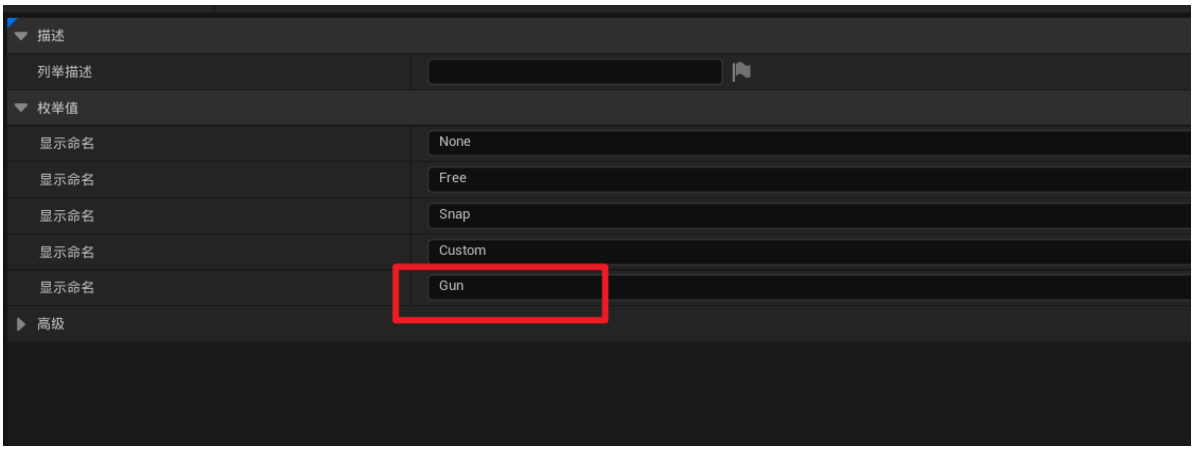




以上蓝图确认无误之后，你的双手就可以按照上面定义的规则来做出相应的动画了

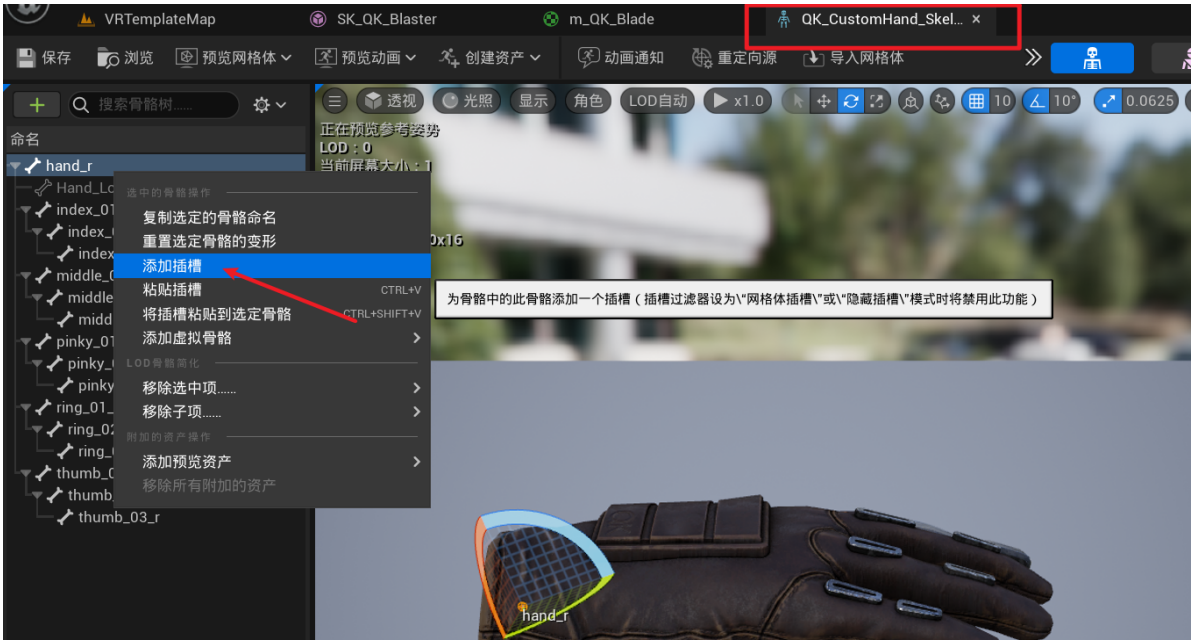
使用武器

新增GrabType枚举

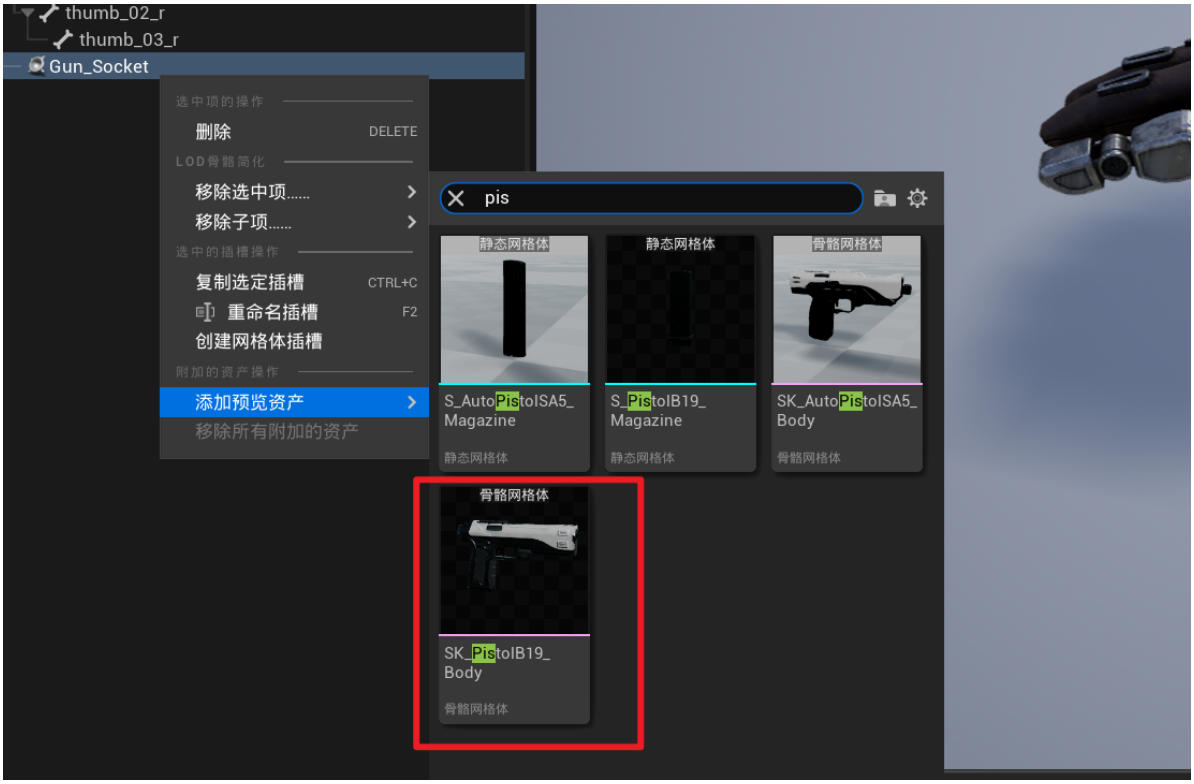


设置插槽

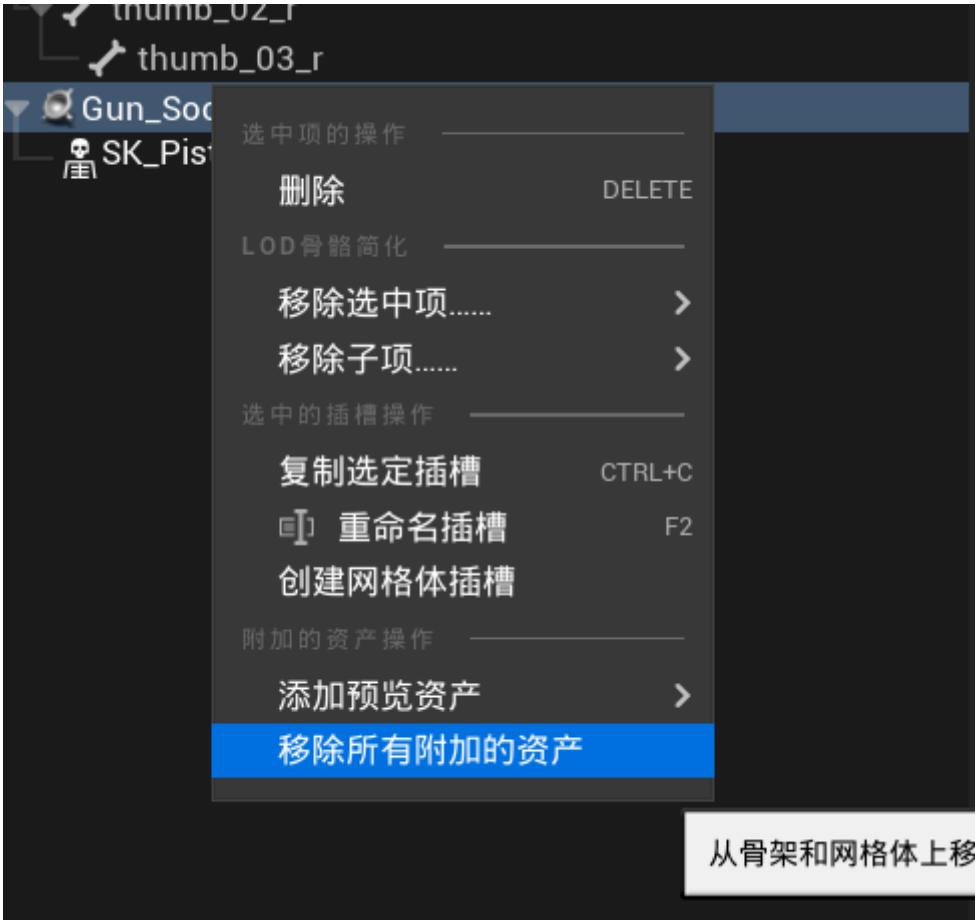
打开手部骨骼模型，为拿枪的动作添加一个插槽



把枪放进来


thumb_02_r
thumb_03_r
Gun_Socket
选中项的操作
删除
DELETE
LOD骨骼简化
移除选中项.....
移除子项.....
选中的插槽操作
复制选定插槽
CTRL+C
重命名插槽
F2
创建网格体插槽
附加的资产操作
添加预览资产
移除所有附加的资产
pis
静态网格体
S_AutoPistolSA5_Magazine
静态网格体
静态网格体
S_PistolB19_Magazine
静态网格体
骨骼网格体
SK_AutoPistolSA5_Body
骨骼网格体
骨骼网格体
SK_PistolB19_Body
骨骼网格体

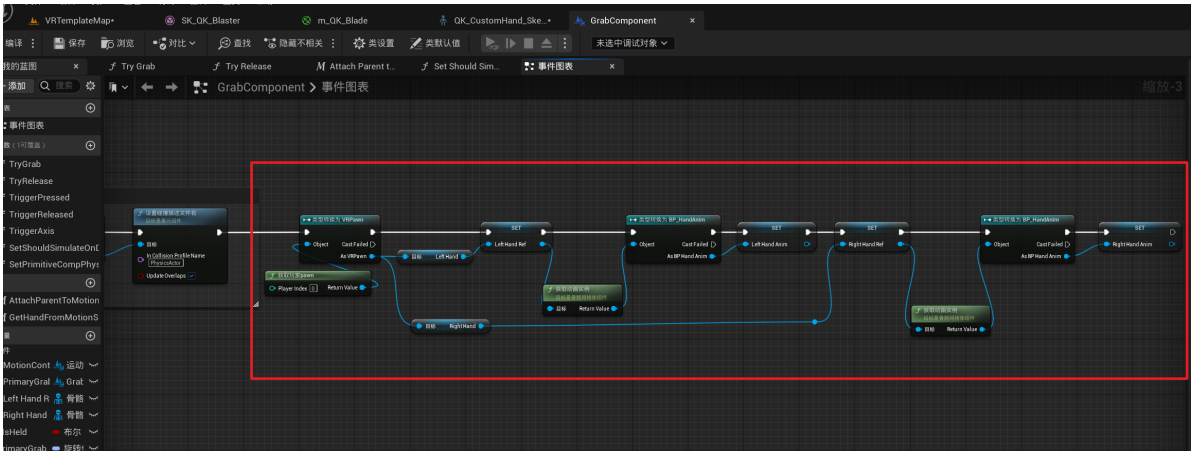
把枪移动到合适的位置



然后把资产移除，并把动画换成默认

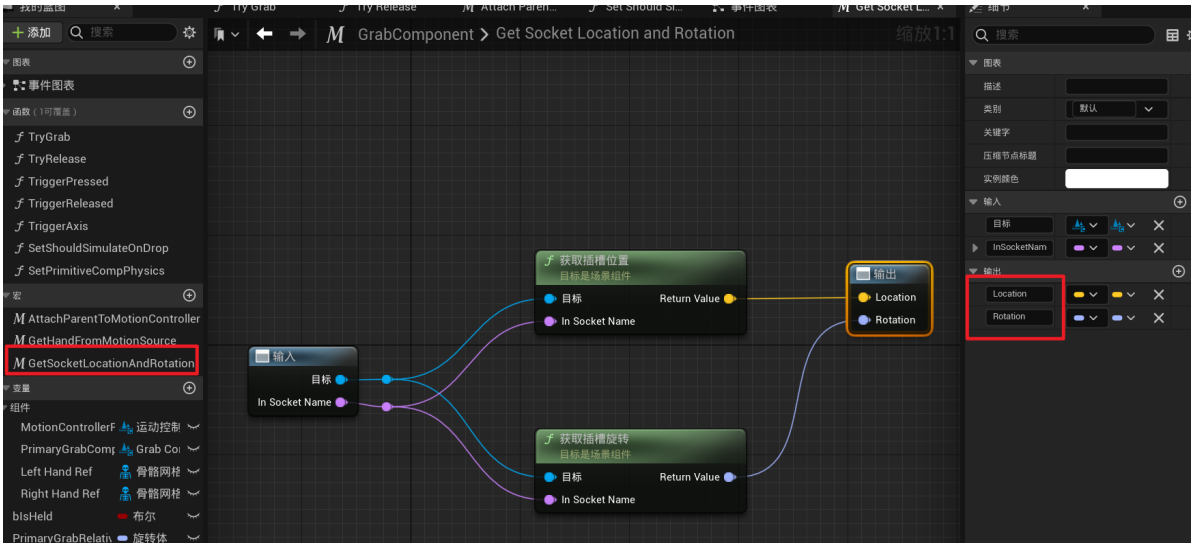
使用插槽

打开GrabComponent，编写如下蓝图

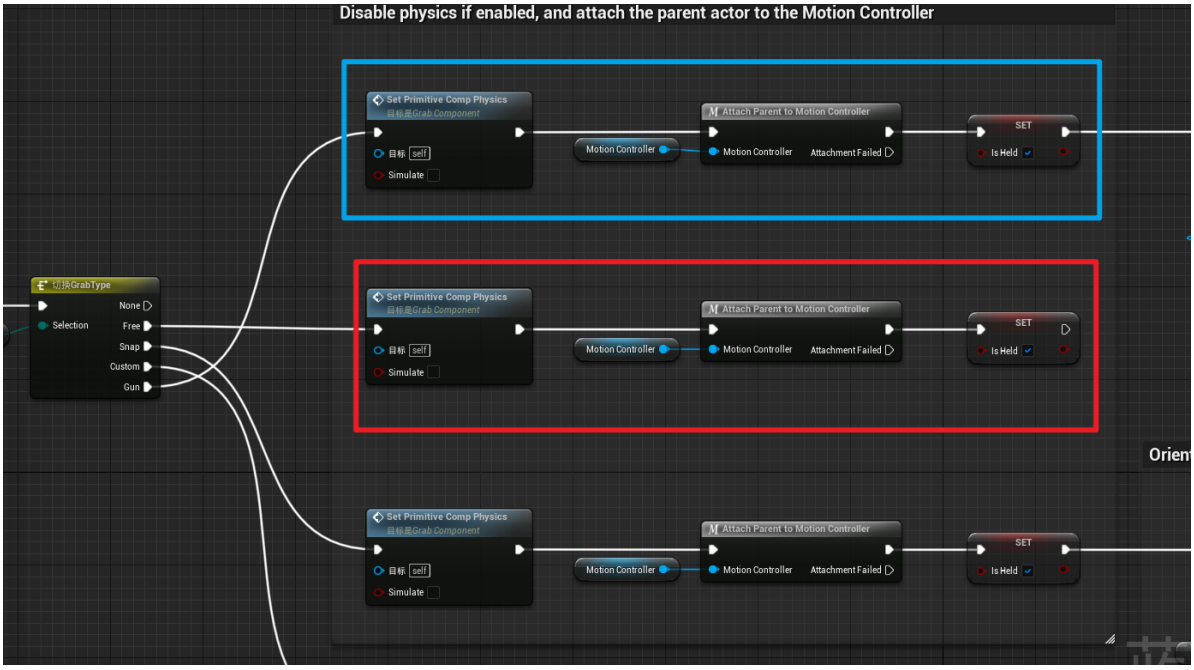


注意，上面的每一次SET都是一次变量提升，记得修改它的变量名，跟上图保持一致。这些蓝图的意义是让我们在每一次Grab的时候能够调用对这些变量的引用

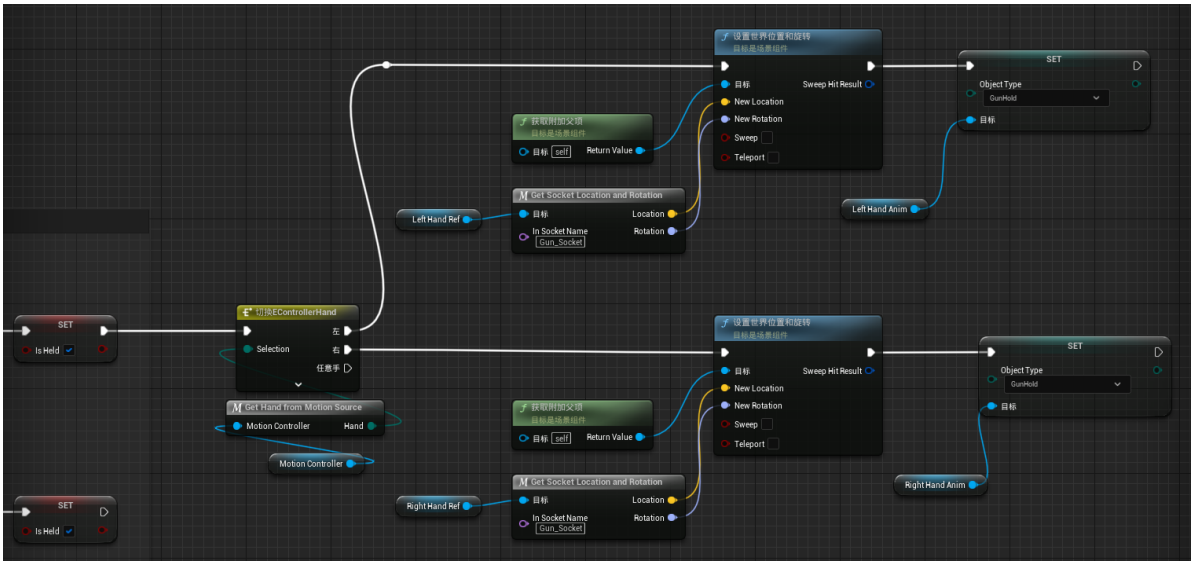
然后创建一个宏，方便获取插槽的位置和旋转，注意命名要保持一致



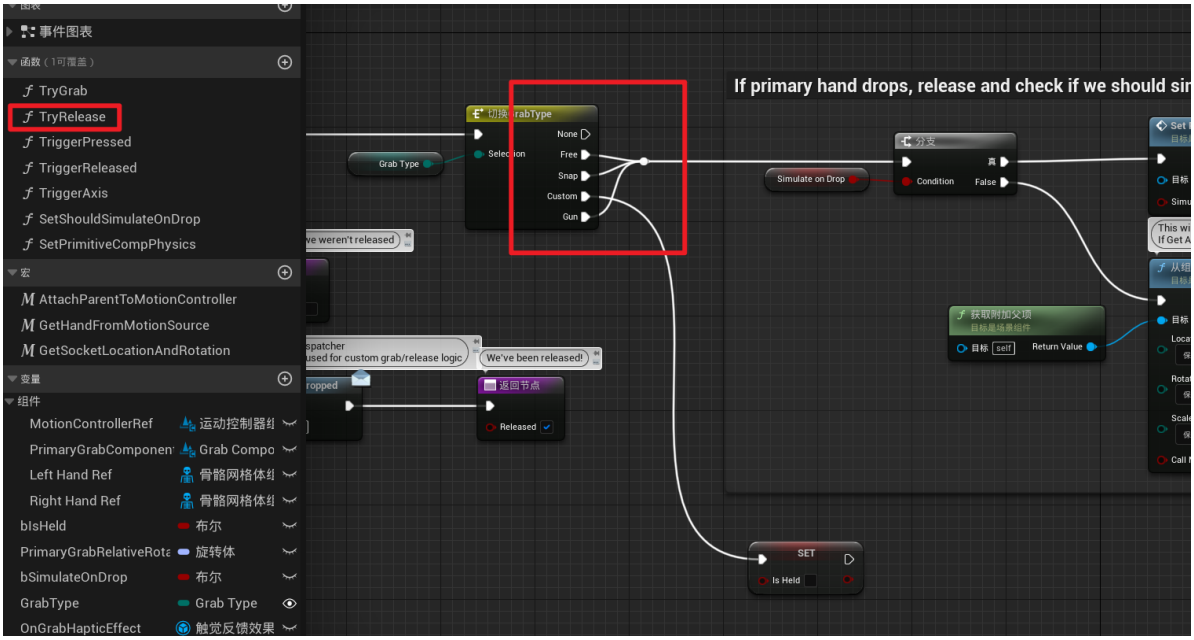
接下来编辑TryGrab函数，将红色框的蓝图复制一份到蓝色框的位置，并连接Gun

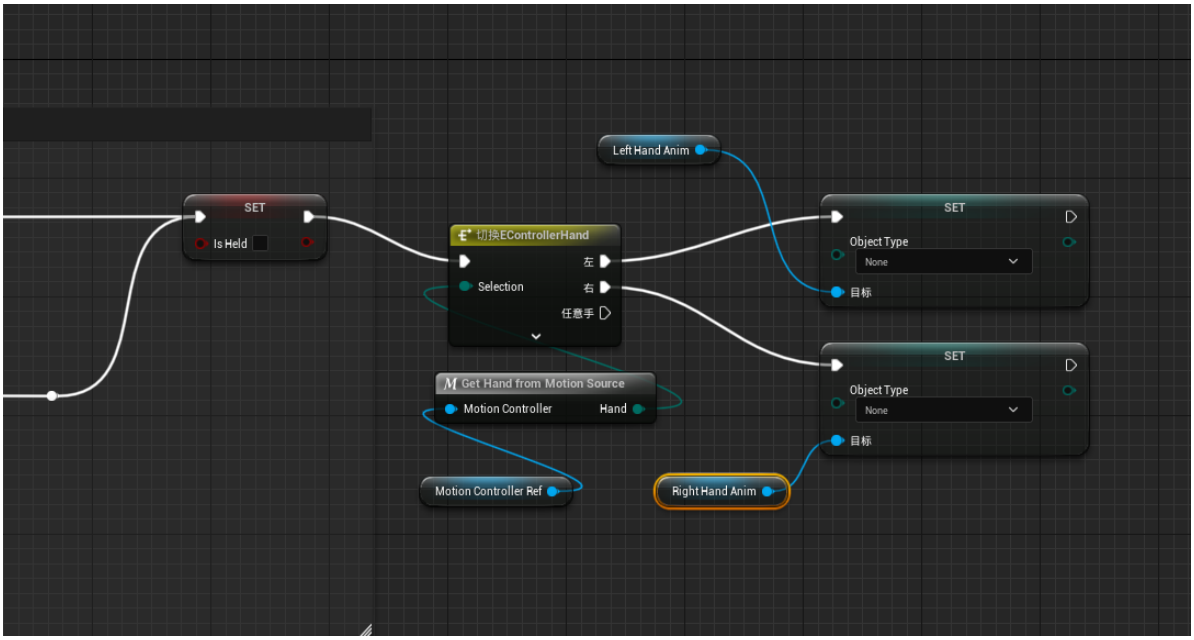


编写如下蓝图，它对应Gun这一块（注意SocketName，一定要和前面的插槽名字保持一致。观察蓝图，其实很多地方都是一样的，多用复制粘贴）

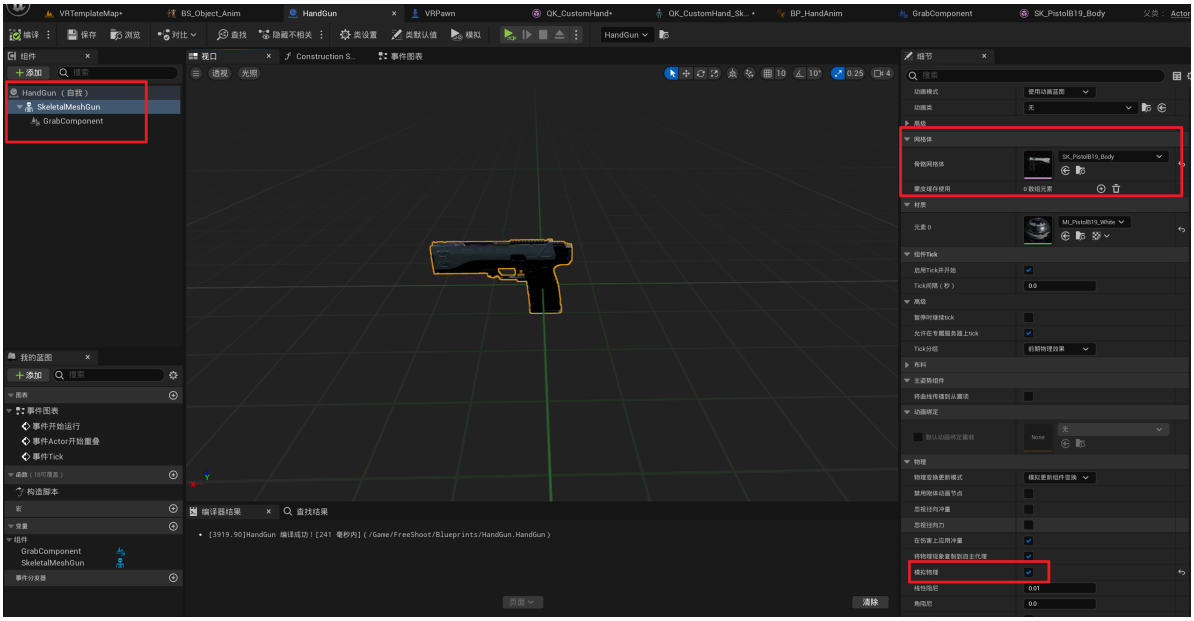


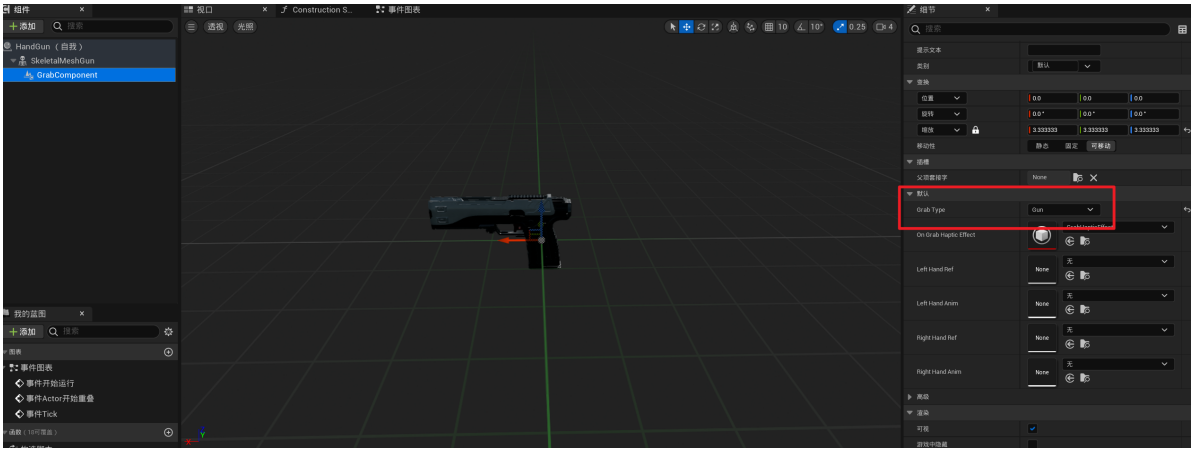
然后编辑TryRelease函数



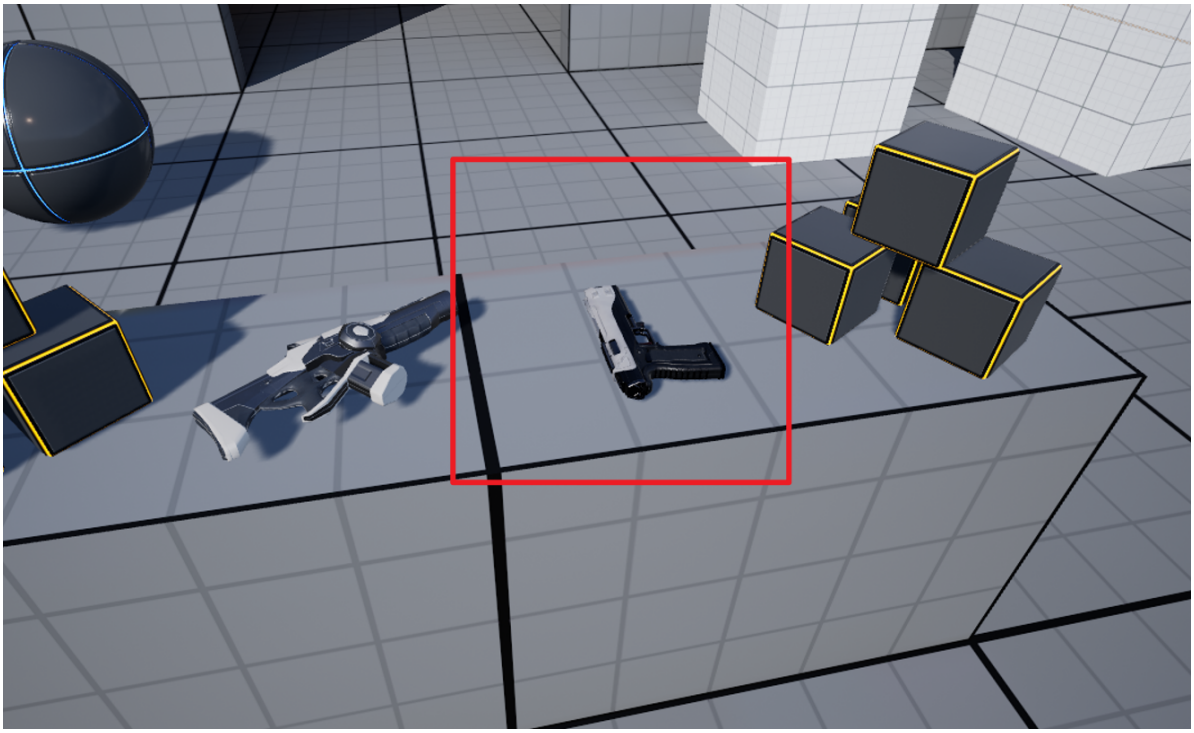


创建资产



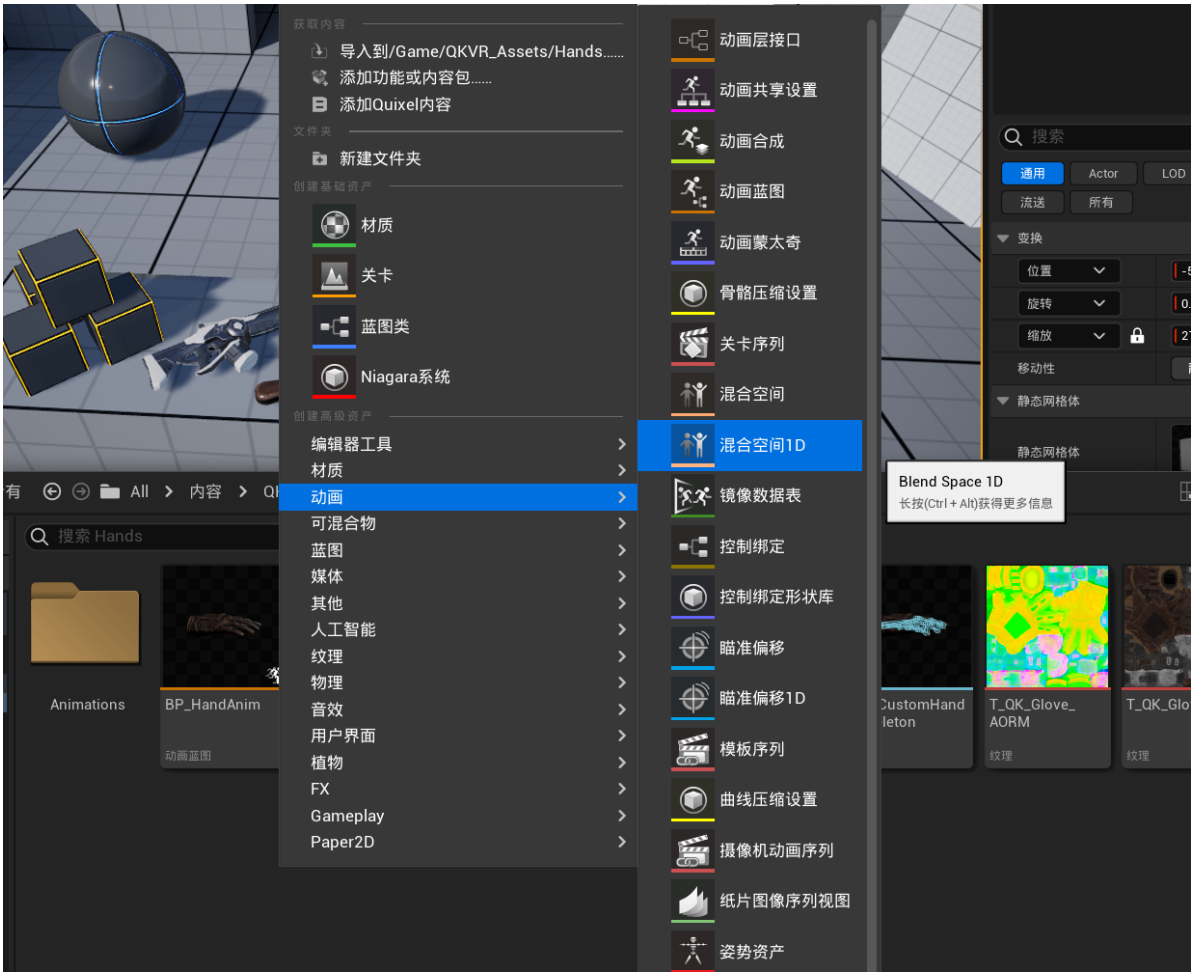


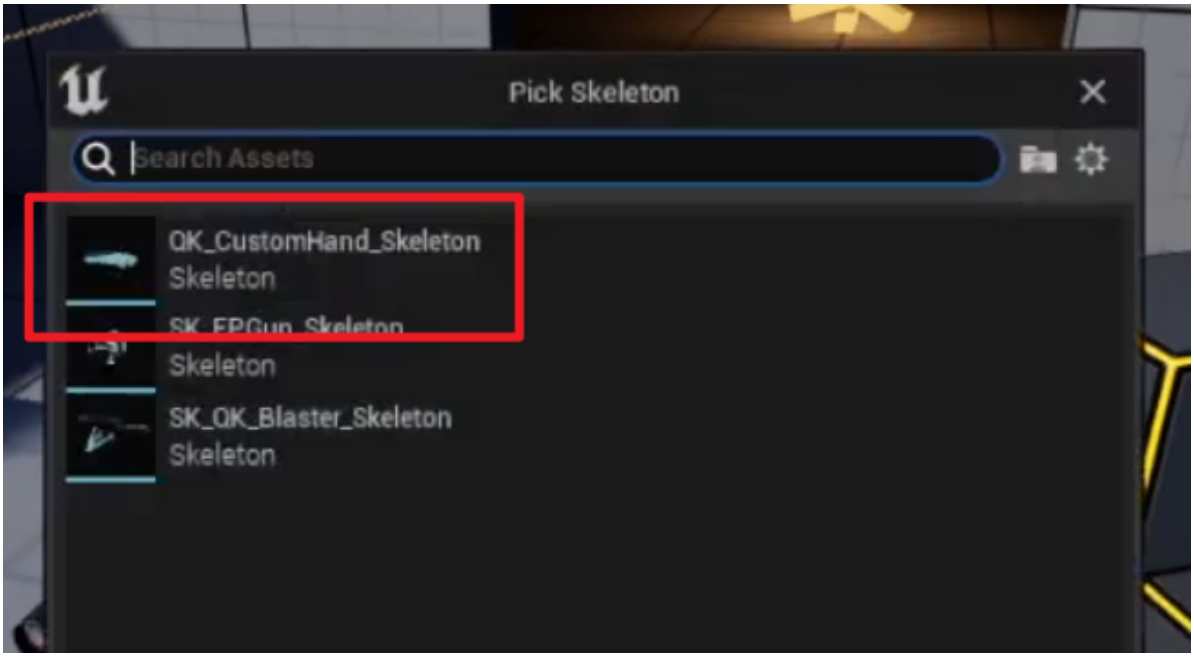
放进场景里



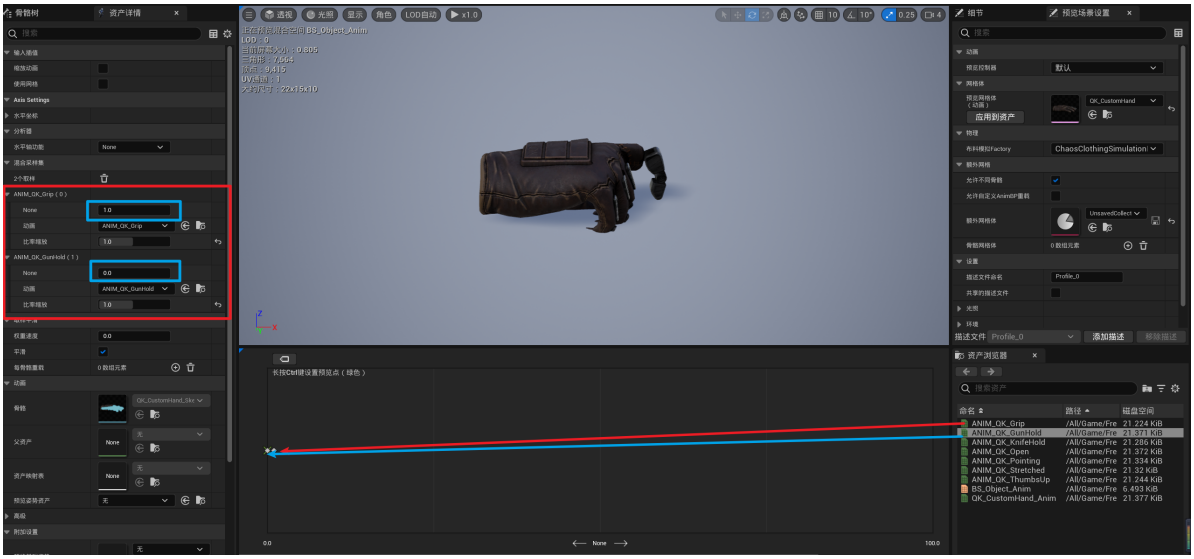
混合空间

还没完，我们还需要创建一个混合空间

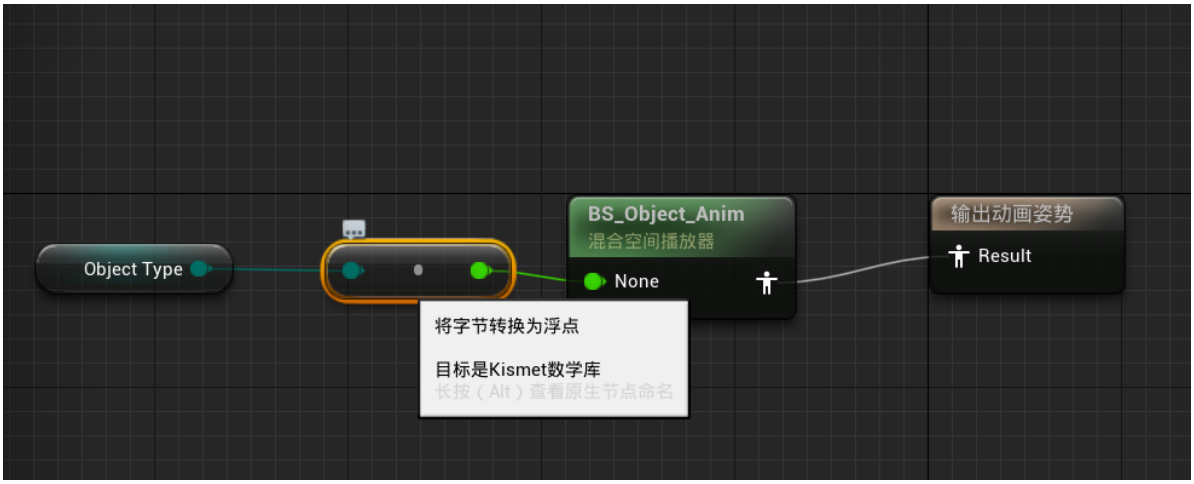




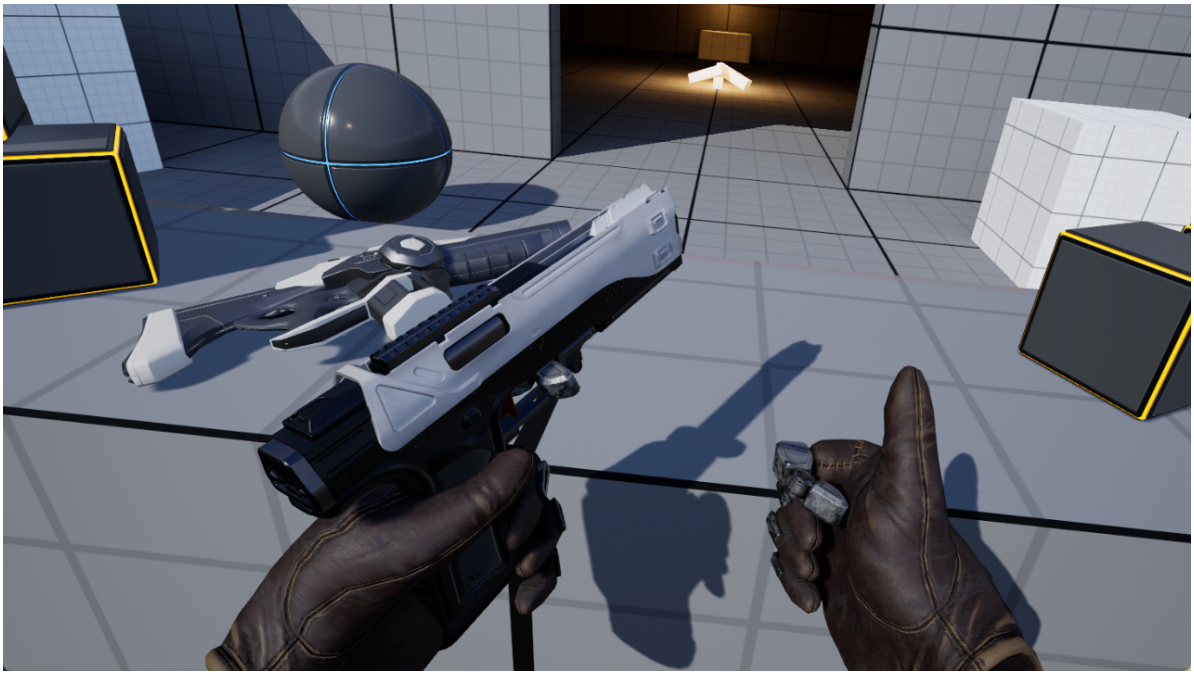
命名为BS_Object_Anim，然后打开它



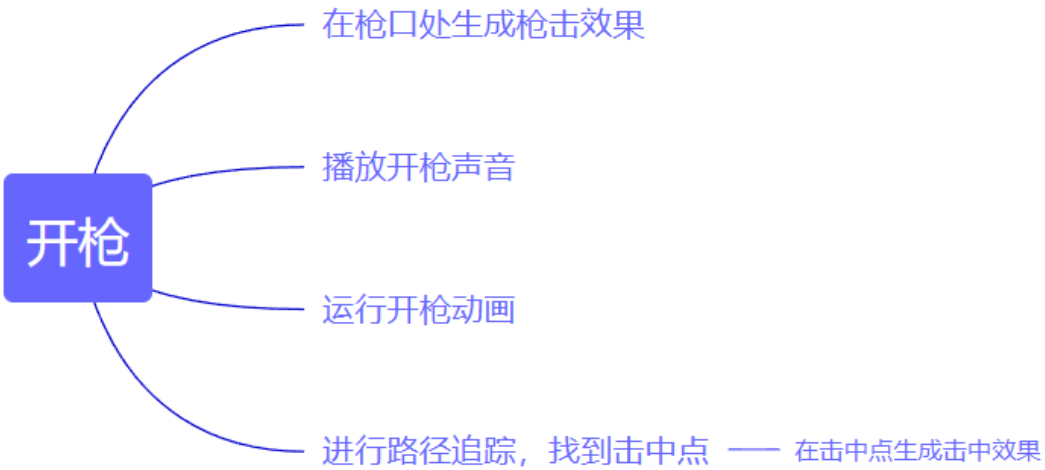
打开BP_HandAnim，进入Object状态，删掉原来的ANIM_OK_Grip序列播放器，编写如下蓝图



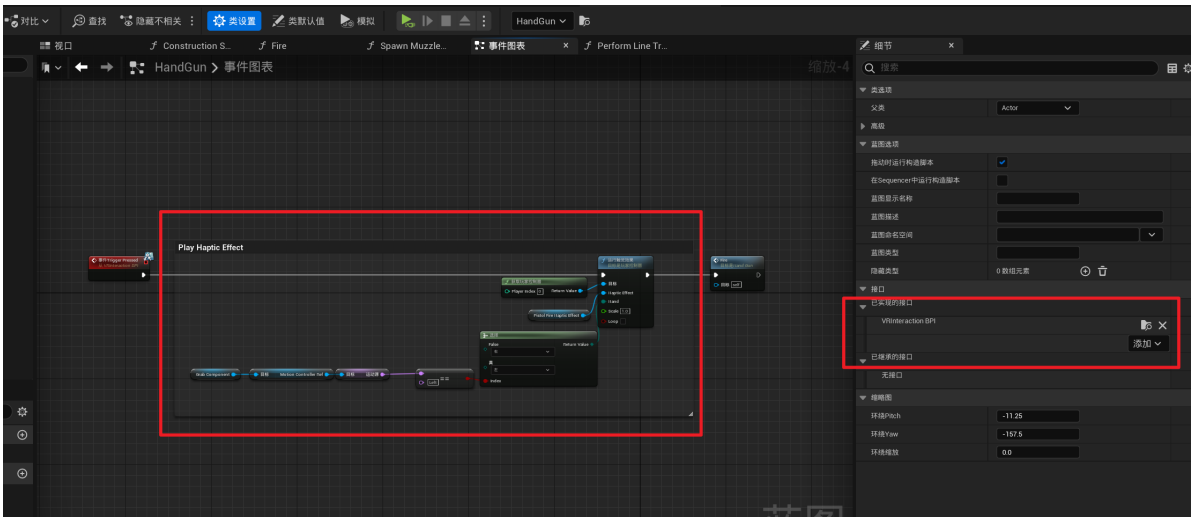
接着你可以再次运行查看效果，不出意外应该是一切正常（出意外了就好好检查一下过程吧，代码是不会骗人的，害）



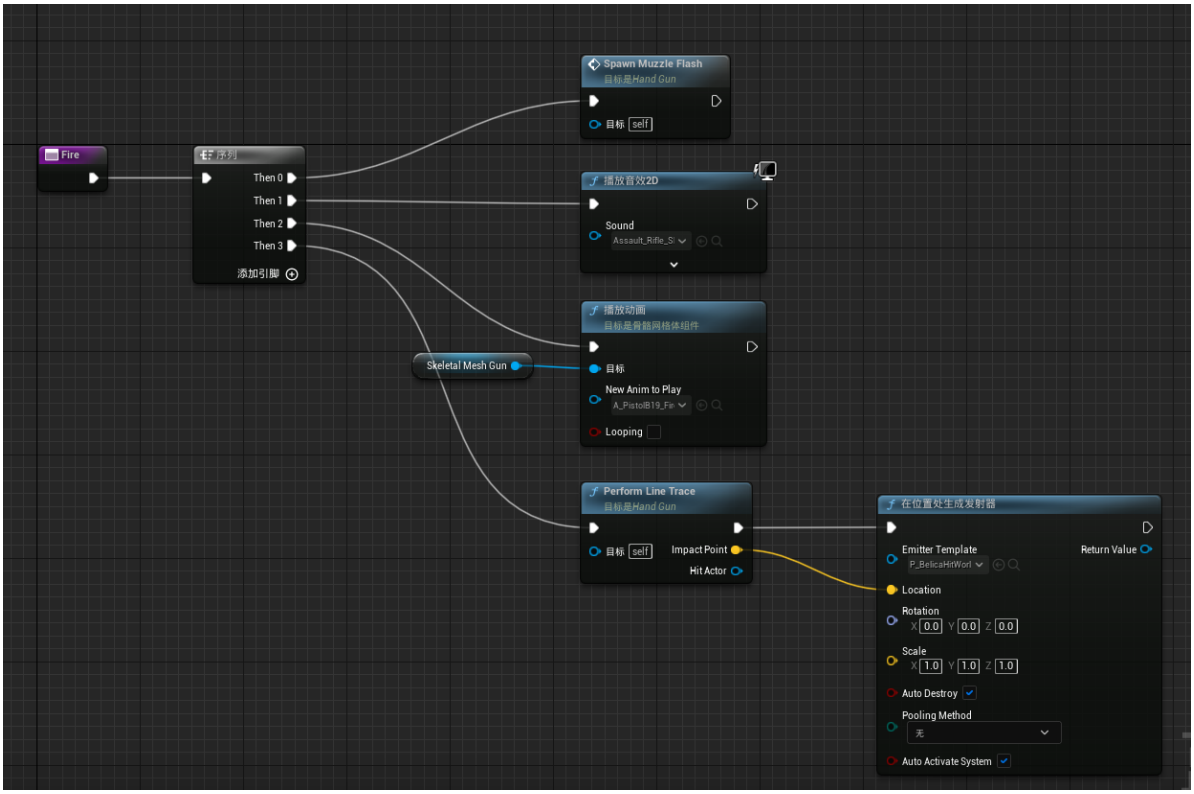
手枪



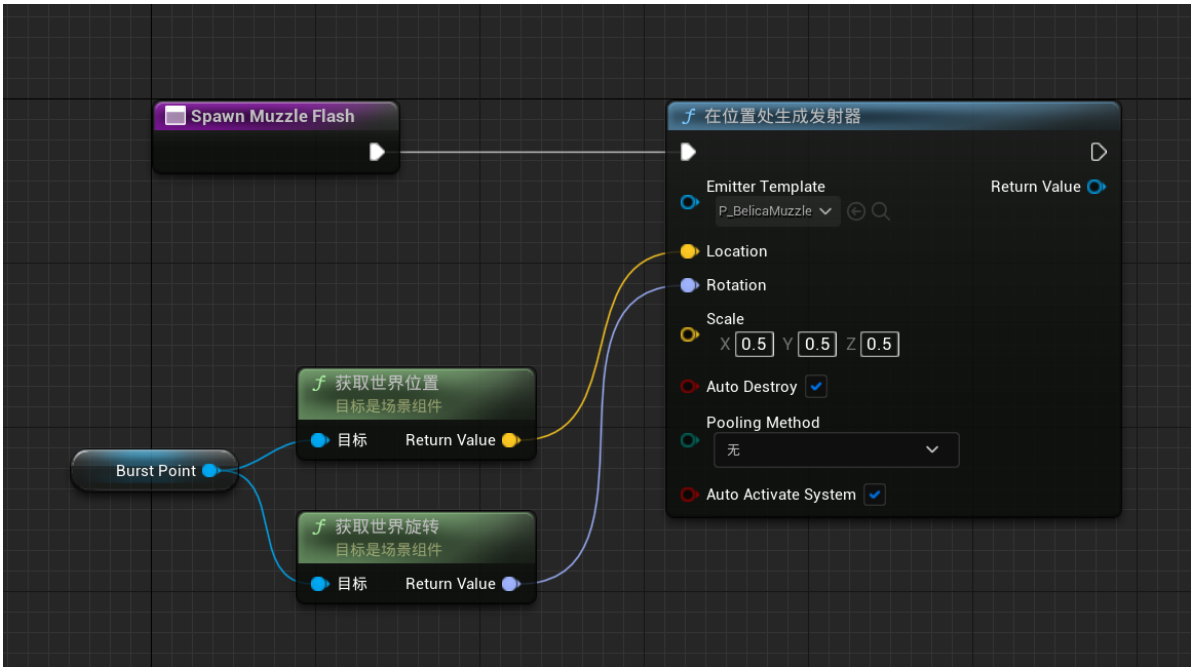
继承接口，从系统提供的样例中复制红色框的函数，然后创建一个Fire函数



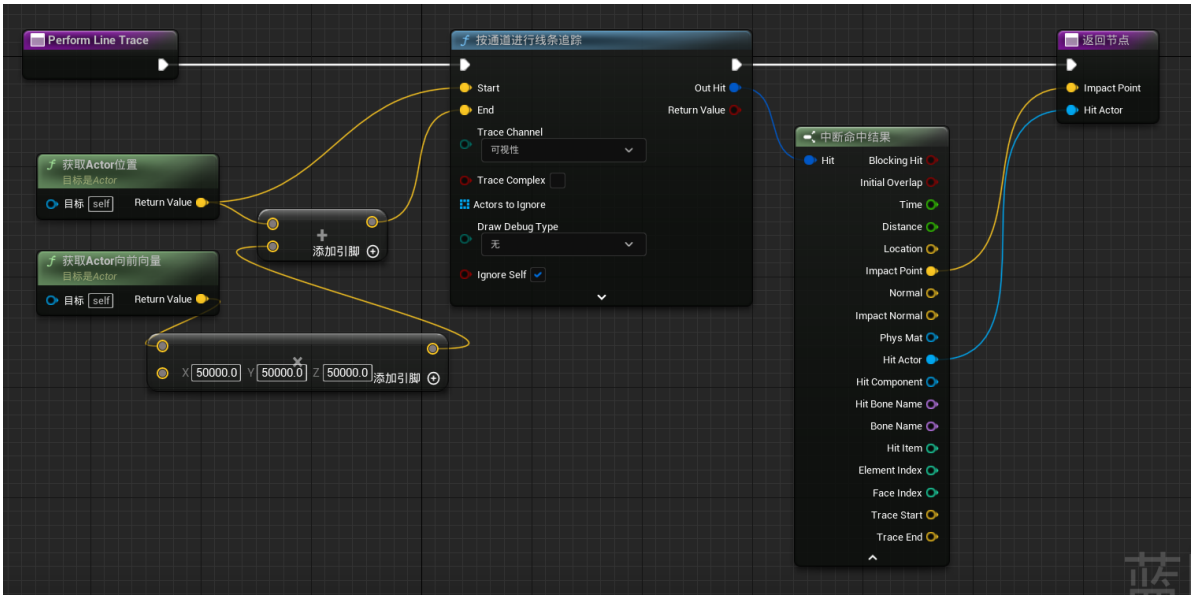
按照思维导图的思路，编写Fire函数蓝图



其中击中效果的具体实现如下

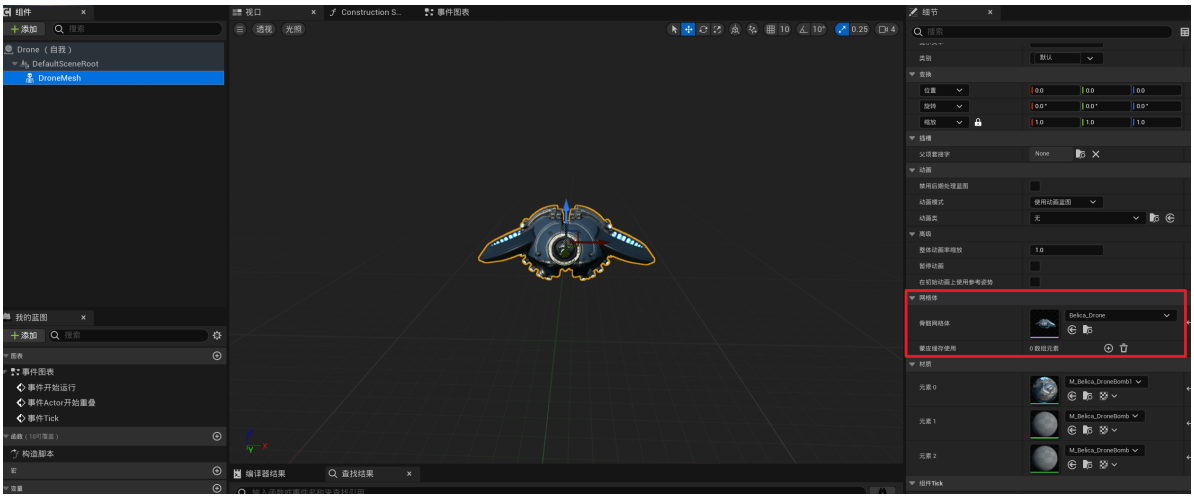


路径追踪的代码如下



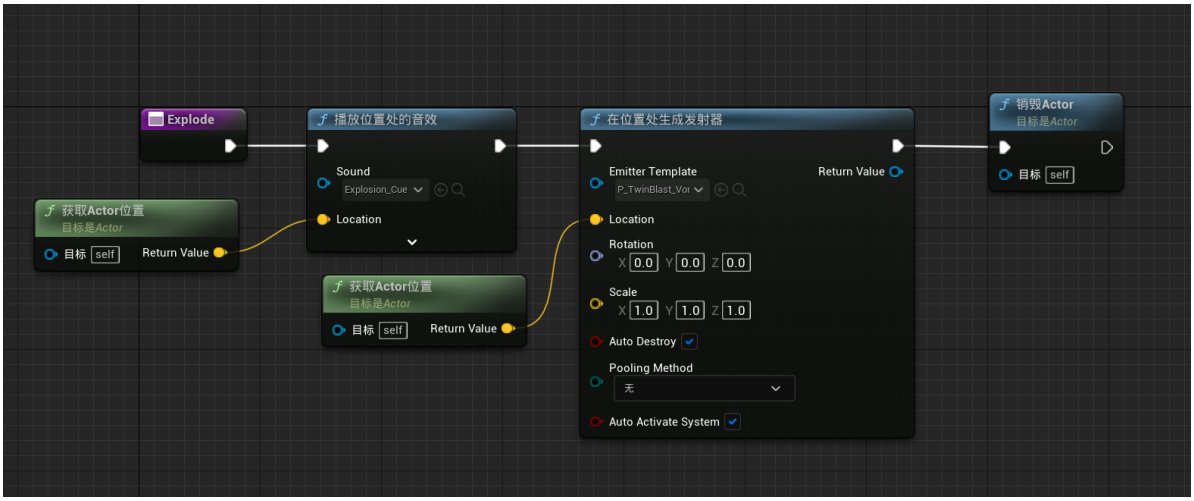
敌人

创建一个Drone蓝图（注意这是个Pawn，不是Actor）



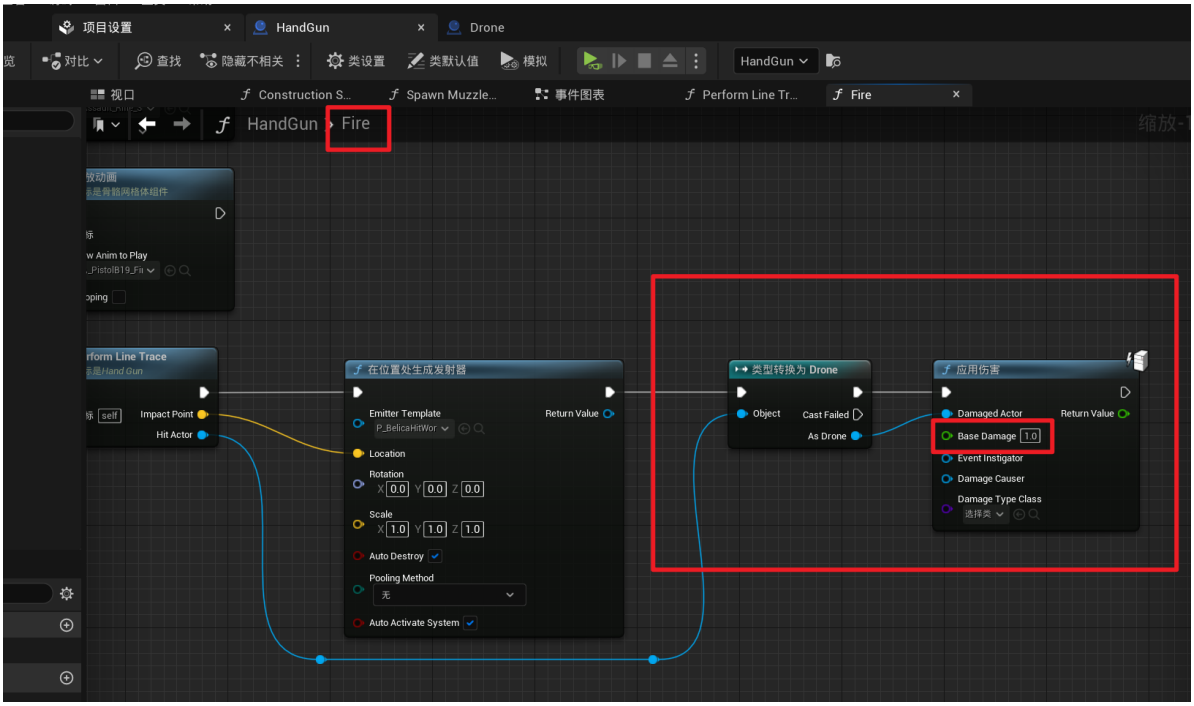
爆炸效果

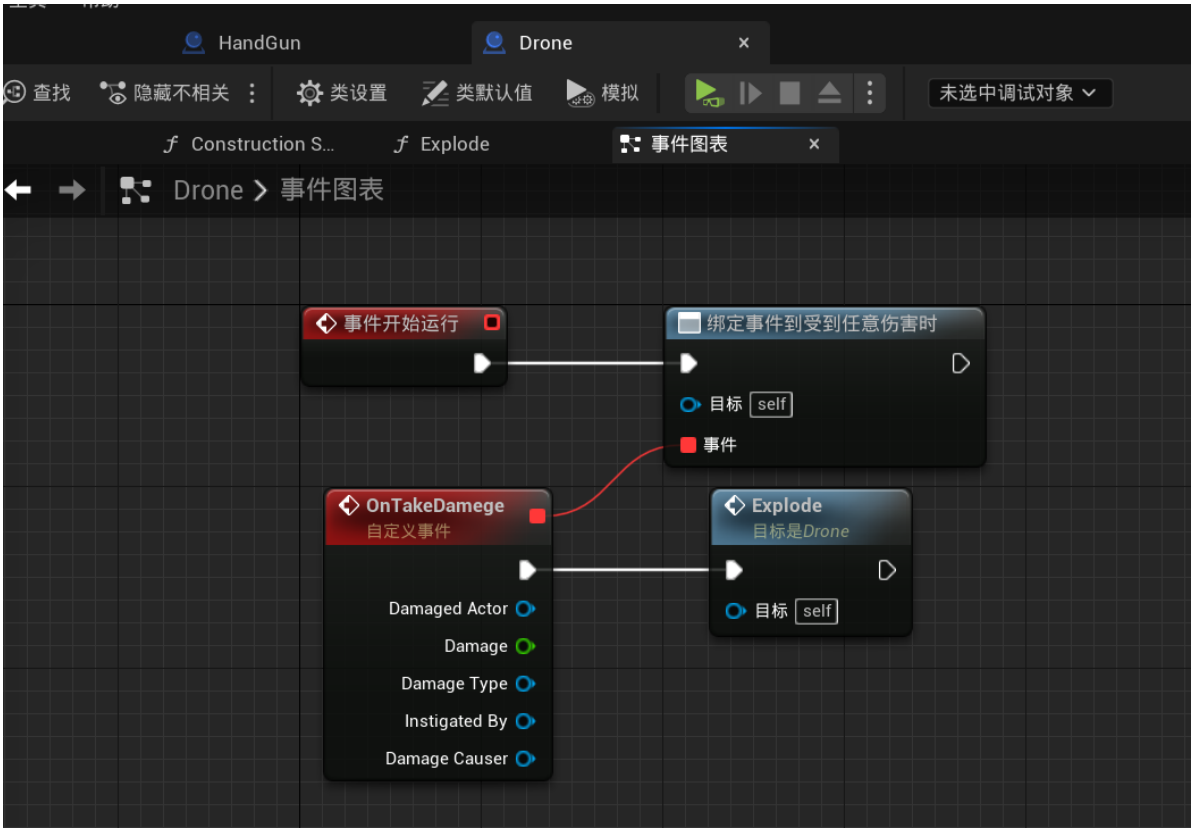
敌人被击中后需要播放爆炸效果，给DronePawn创建一个Explode函数



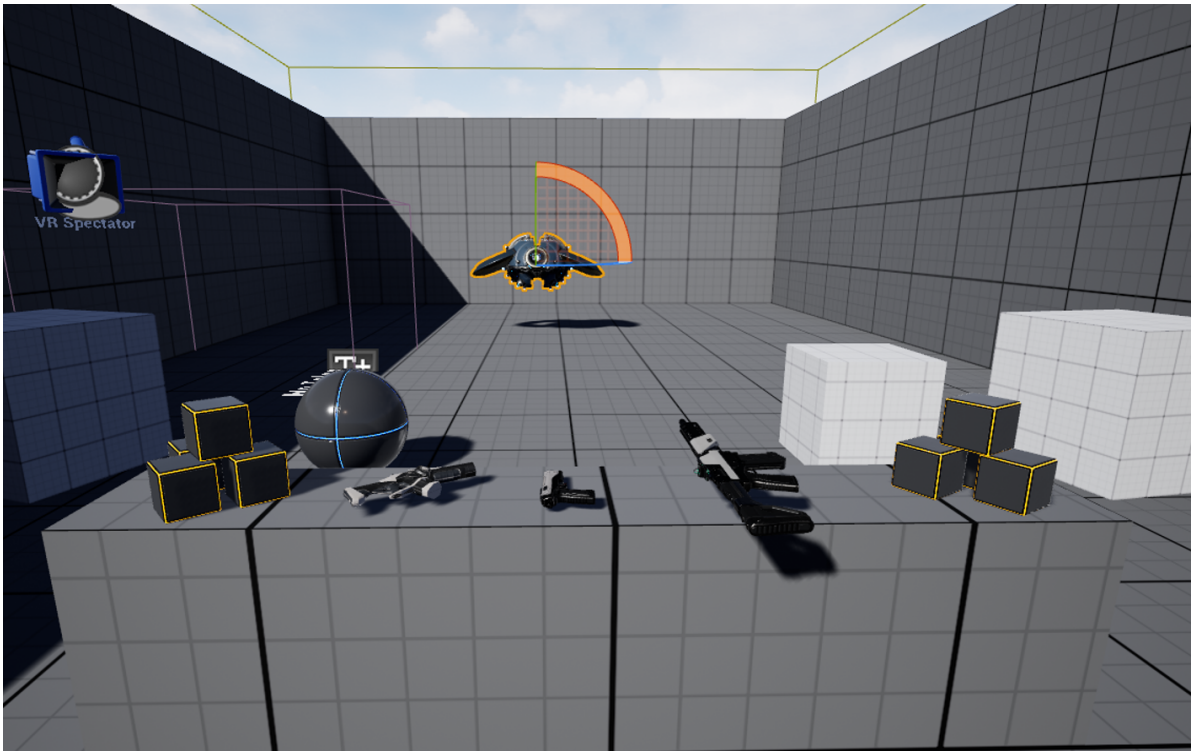
被击中

让枪支可以击中Drone，并运行击中效果



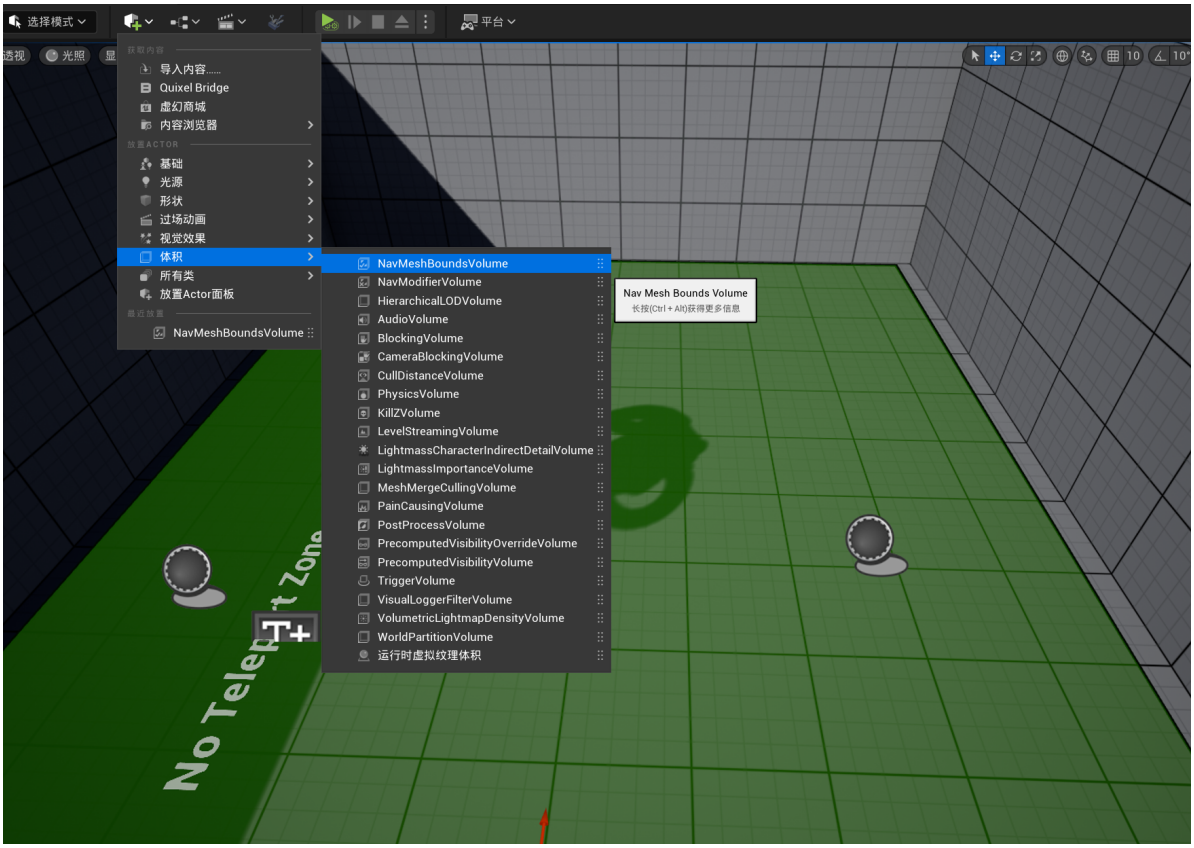


然后把Drone放到场景里，就可以用手枪击中了，步枪也是如此

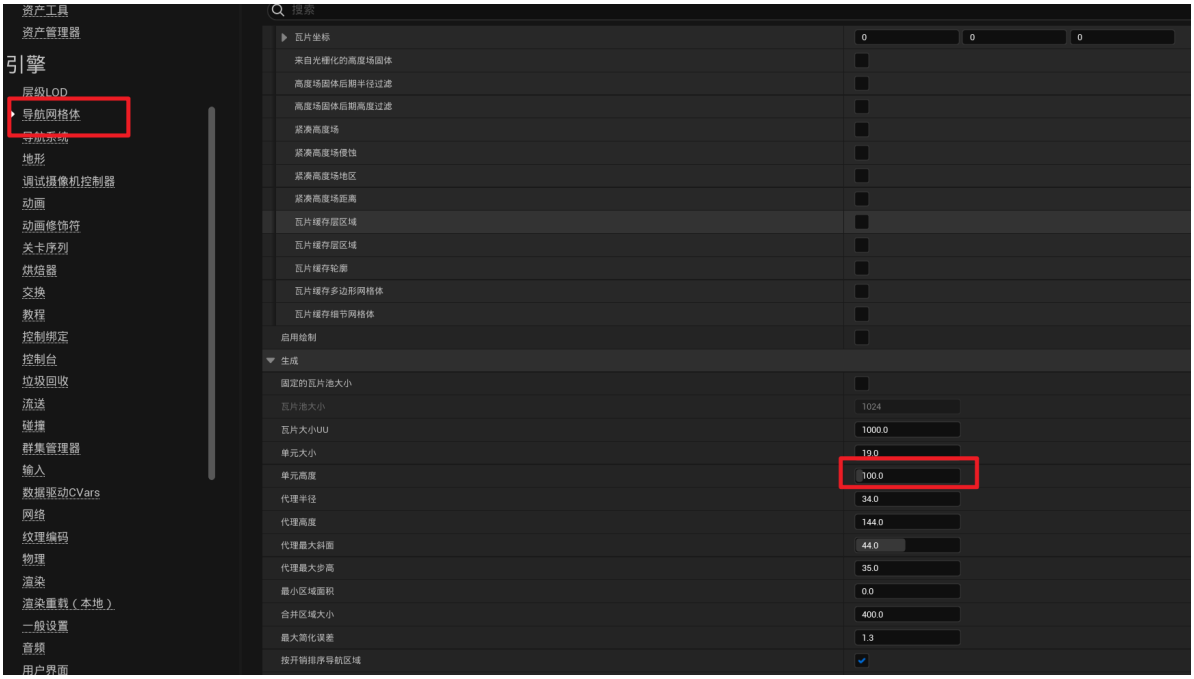


巡逻

添加导航组件，放到路面上，按p建，会发现与地面重合的部分变绿了，绿的部分就说明可以导航

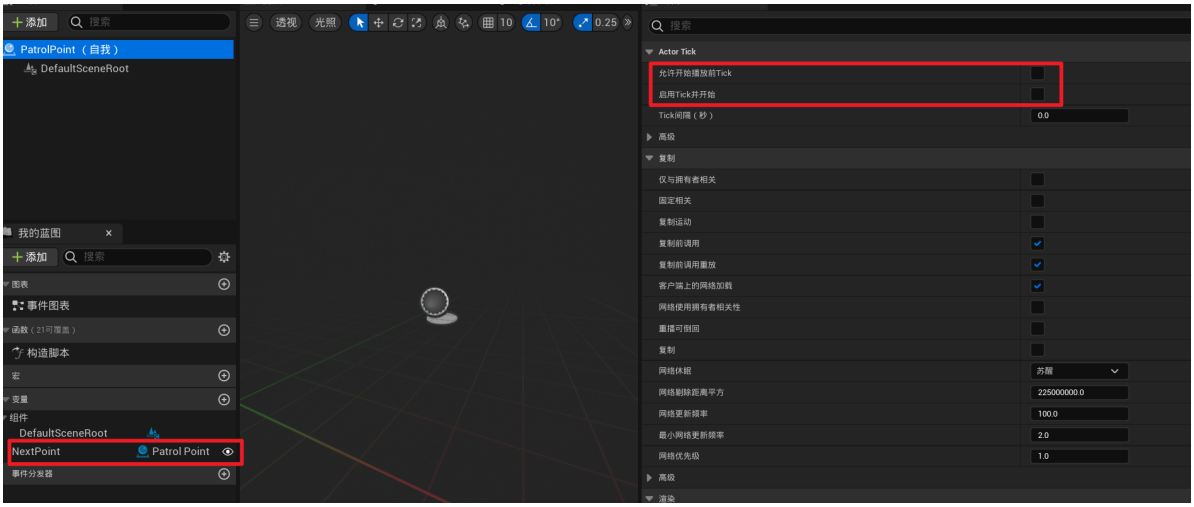


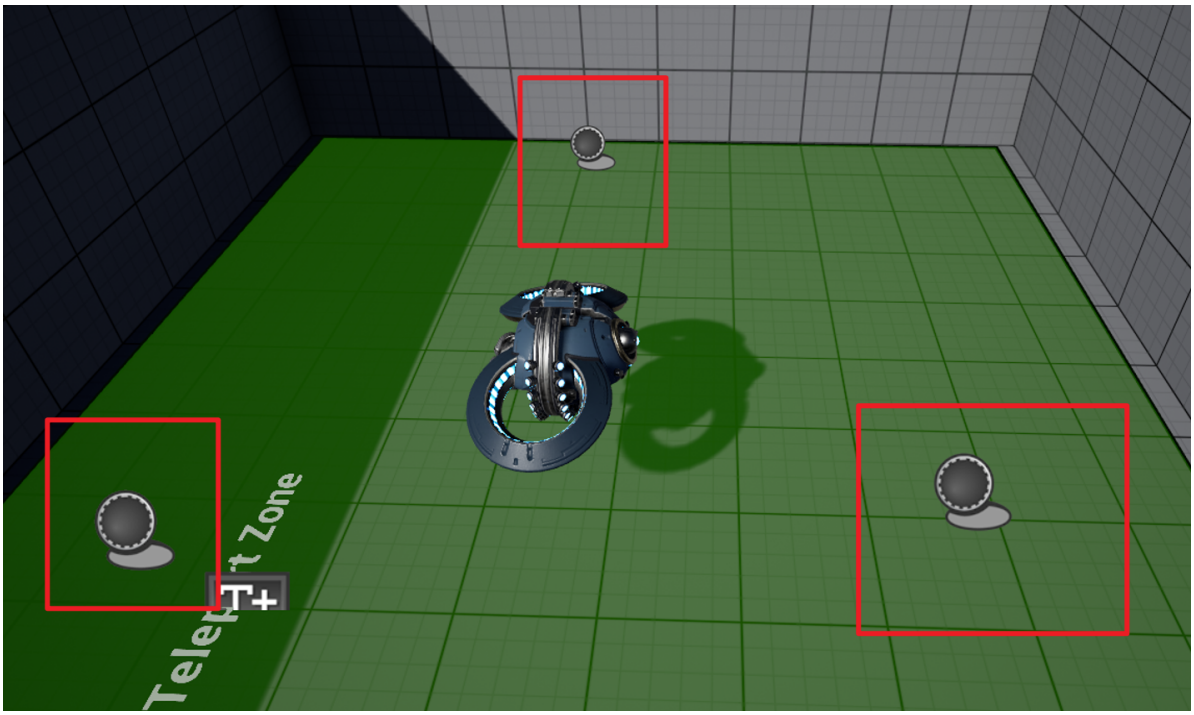
如果发现有路沿没办法被覆盖，可以调整一下项目设置



调整好之后再按一下P键就可以让绿色消失了

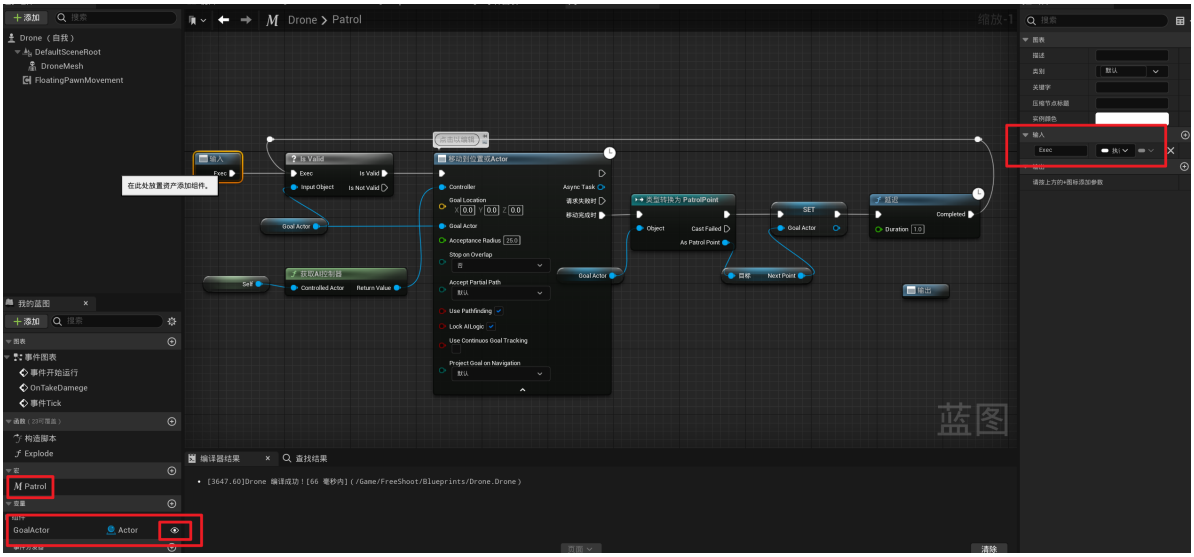
然后创建一个Actor，命名为PatrolPoint，也叫巡逻点，主要用于定位，不用写什么功能



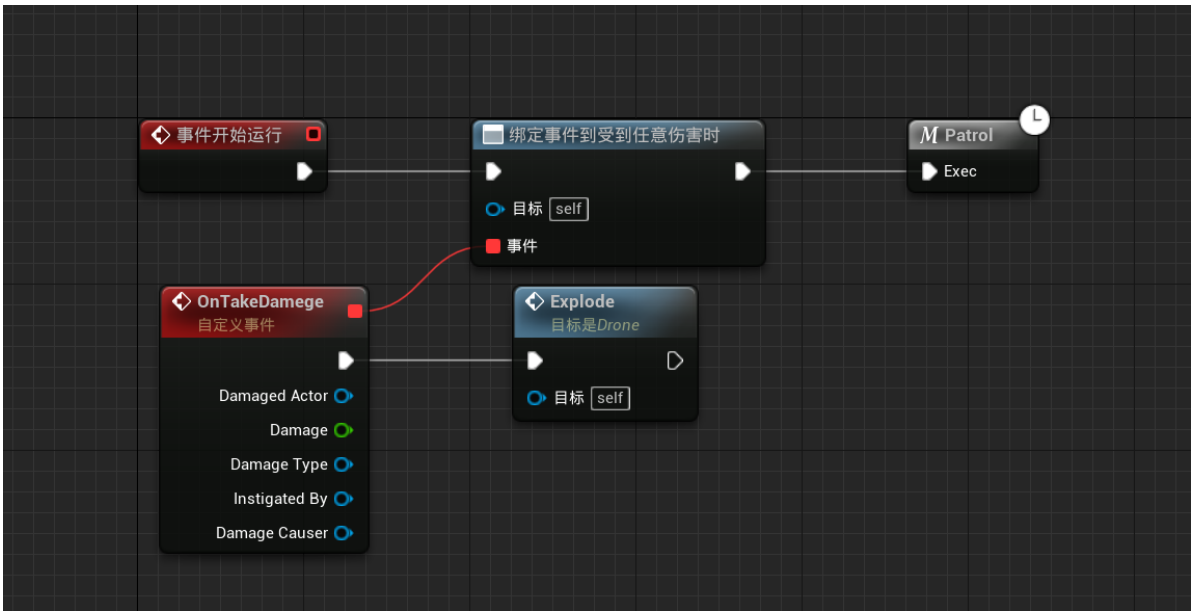


然后互相指定为对方的NextPoint

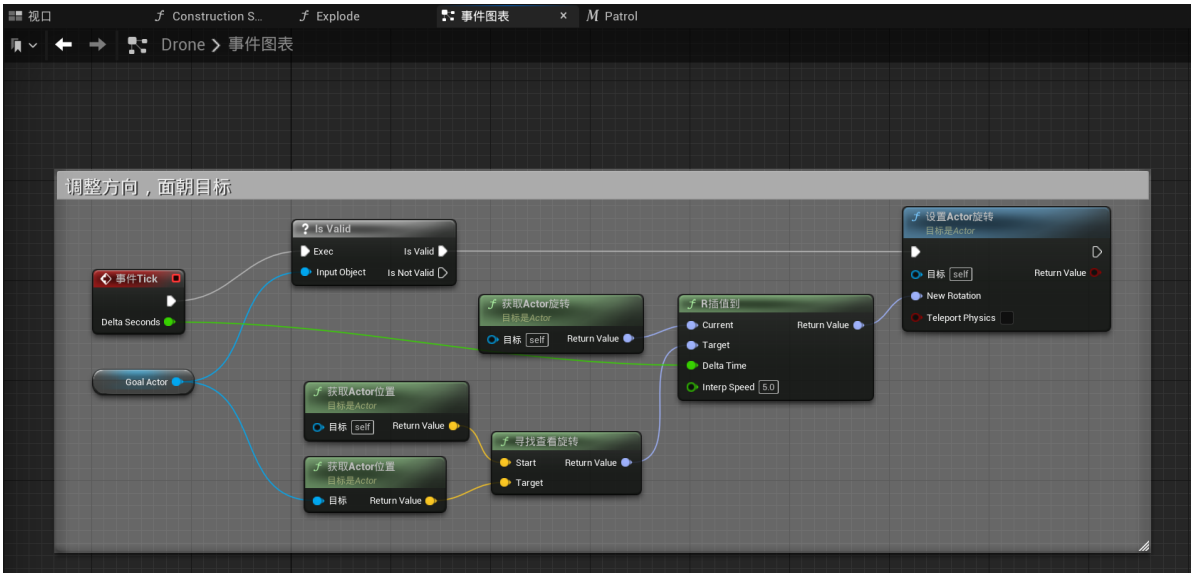
然后在DronePawn中，创建一个宏



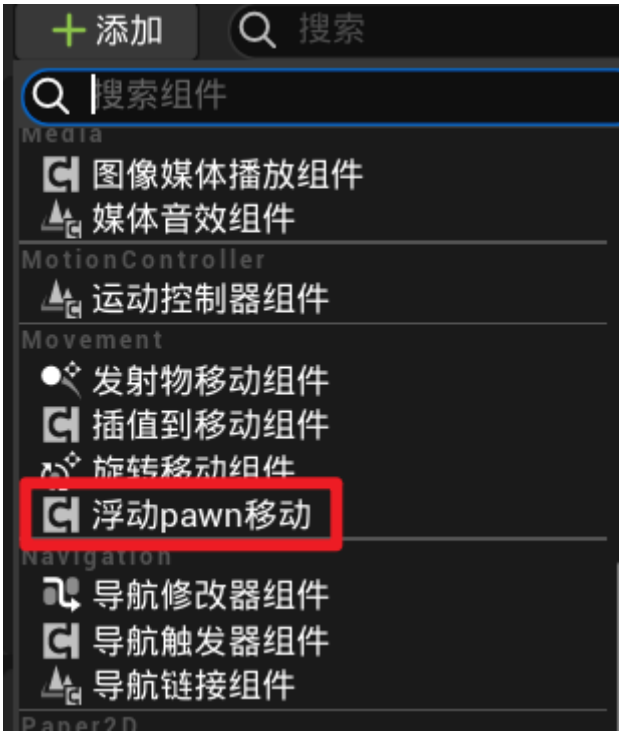
在事件中应用



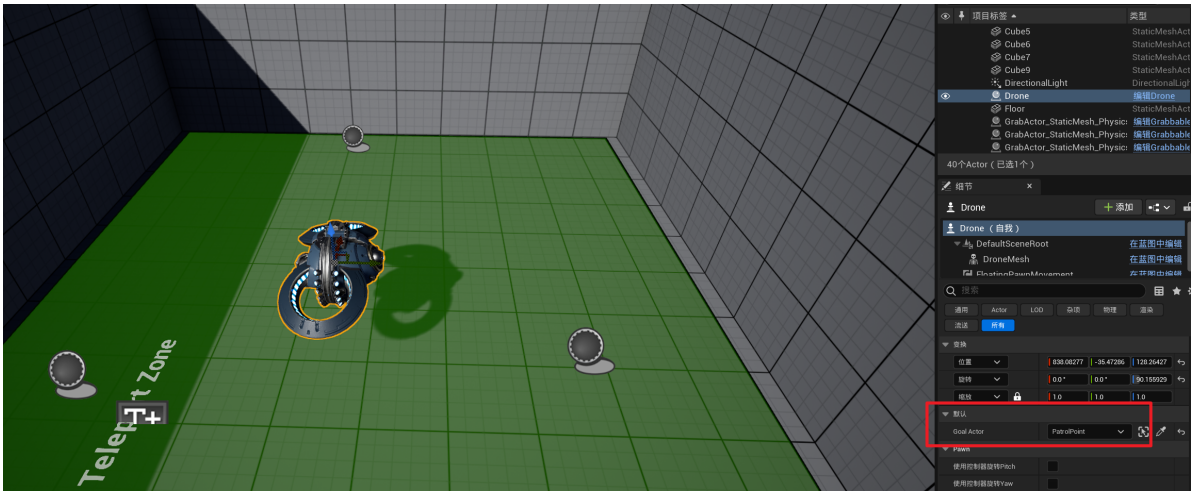
添加控制朝向的蓝图



接下来给Drone创建一个移动组件



指定Goal Actor

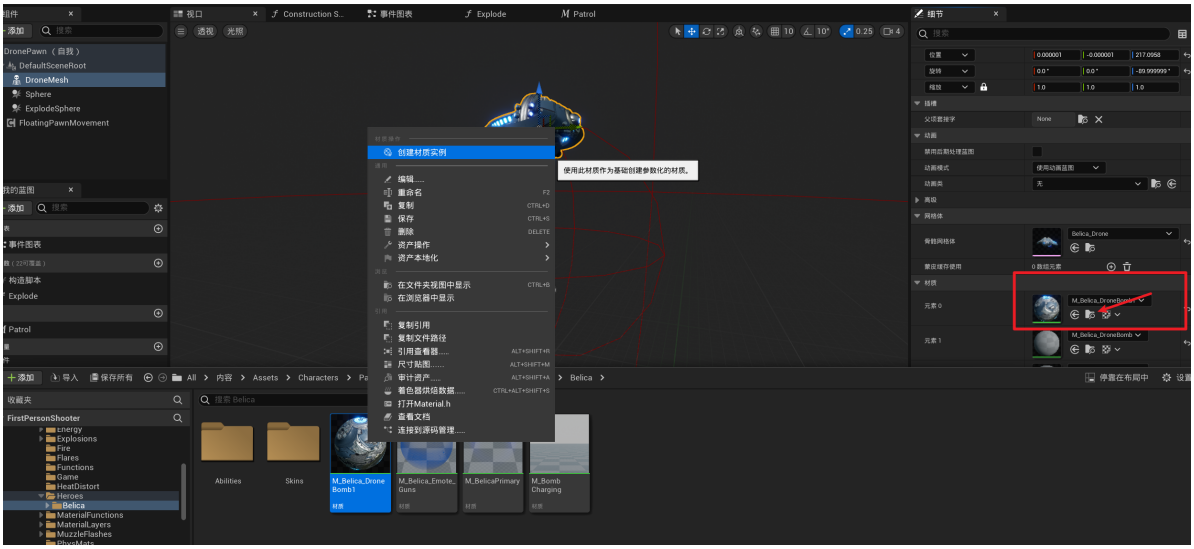


这些个操作的主要原理是：设定GoalActor为初始巡逻点，然后敌机就会在游戏启动的时候移动过去，移动到巡逻点之后，会将GoalActor设置为下一个巡逻点，然后继续移动，如此往复。

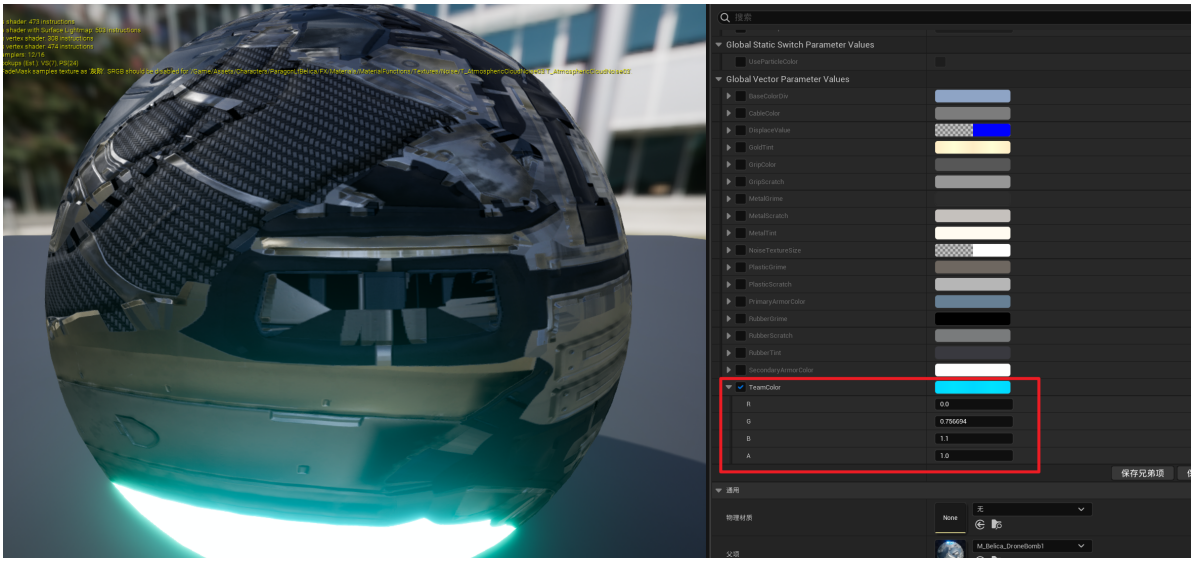
激怒

我们希望敌机被打了之后就变红，实现过程如下：

给敌机创建一个材质实例

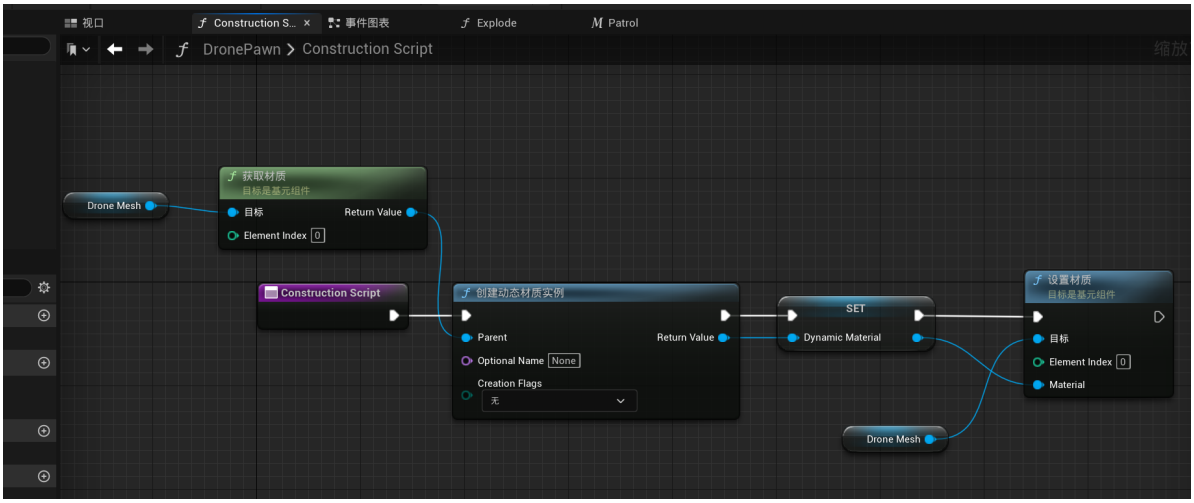


可以看到这里用于控制颜色的字段是TeamColor

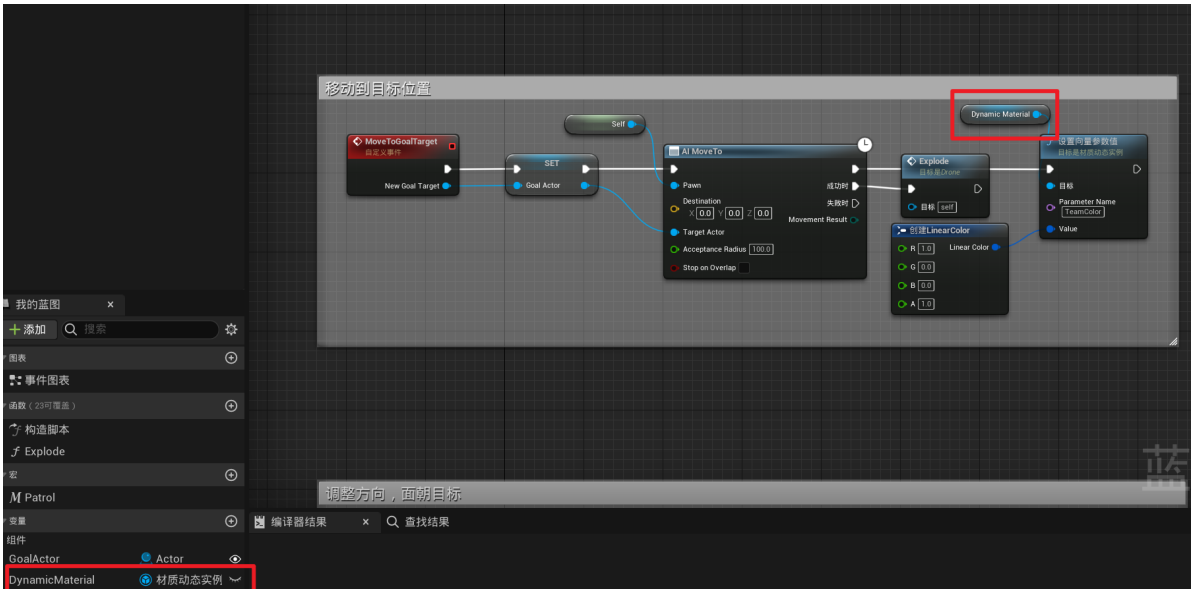


我们需要在游戏运行的时候给他创建动态的材质实例

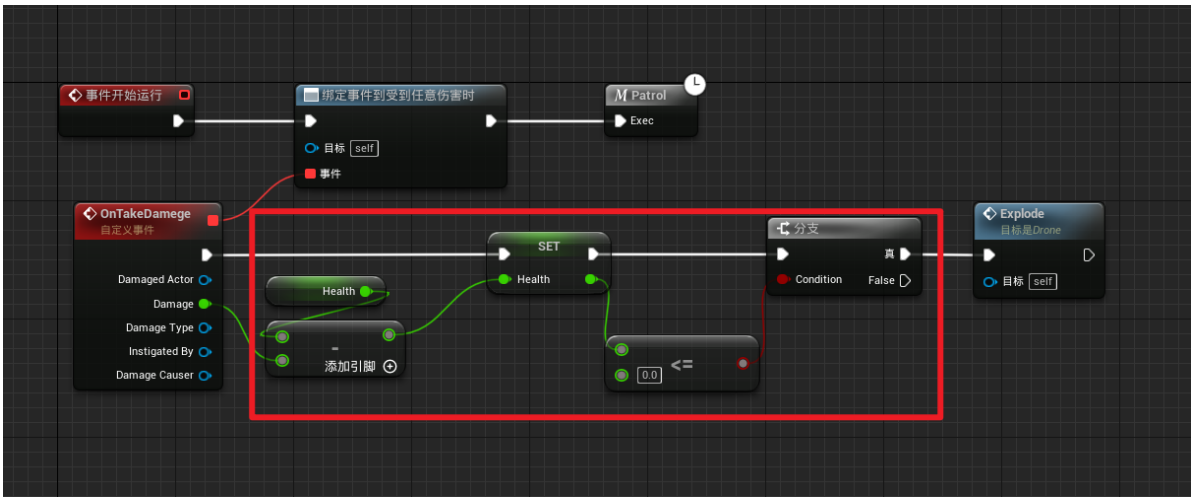
打开DronePawn的构造脚本，编写如下蓝图



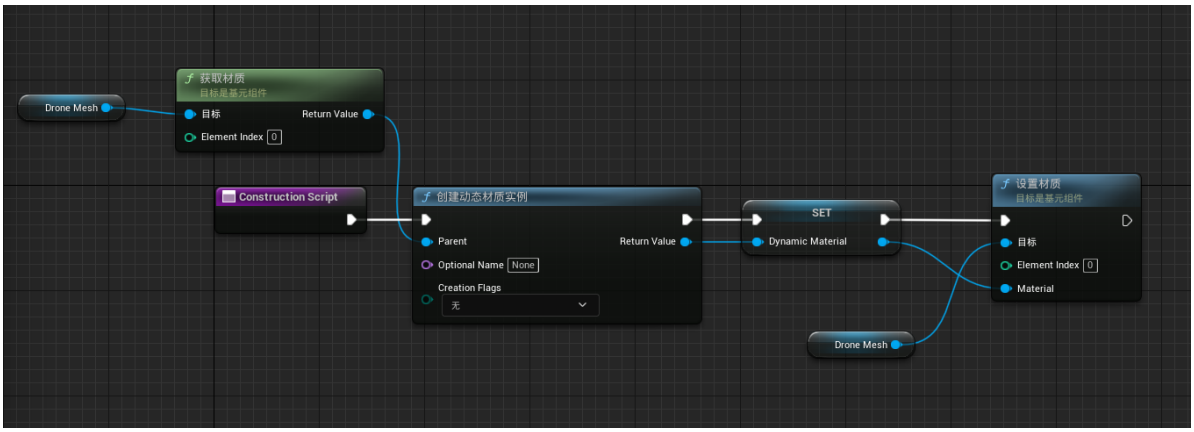
然后修改Drone代码，新增一个自定义事件



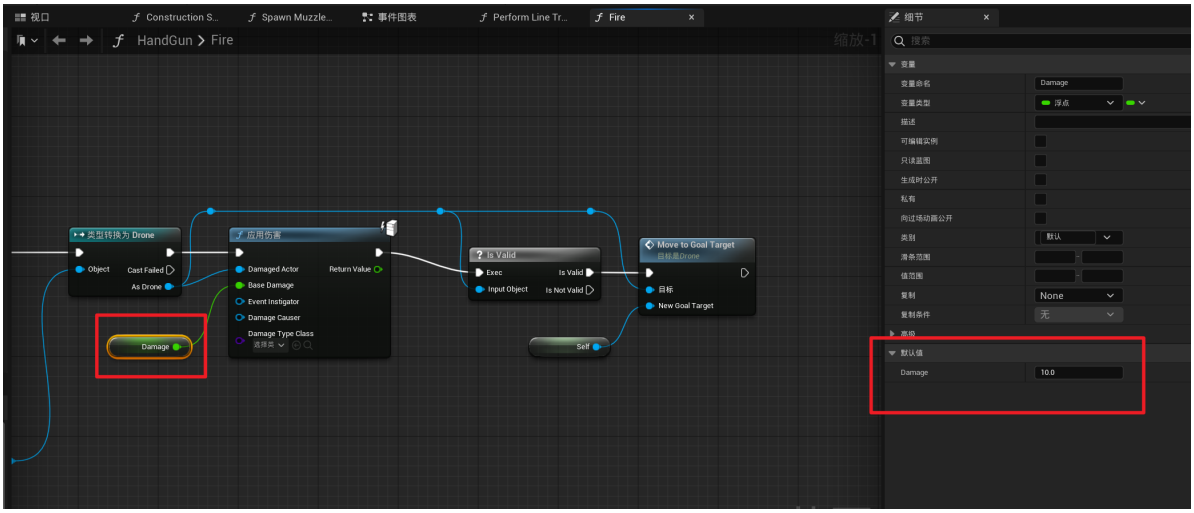
修改受伤函数



修改构造函数



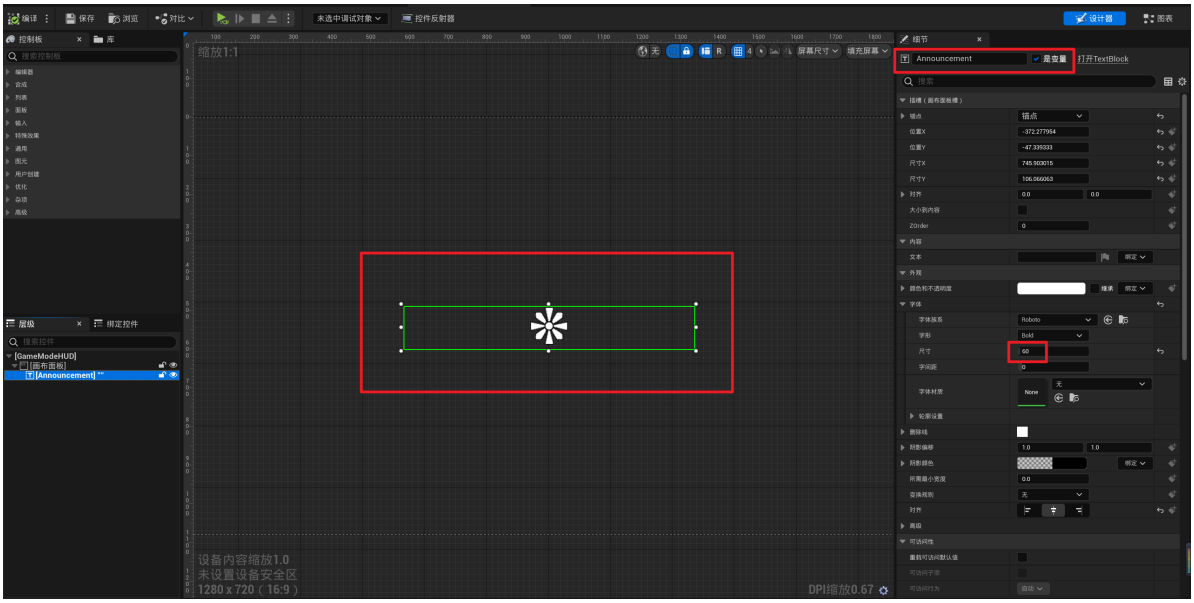
然后在Fire中应用一下



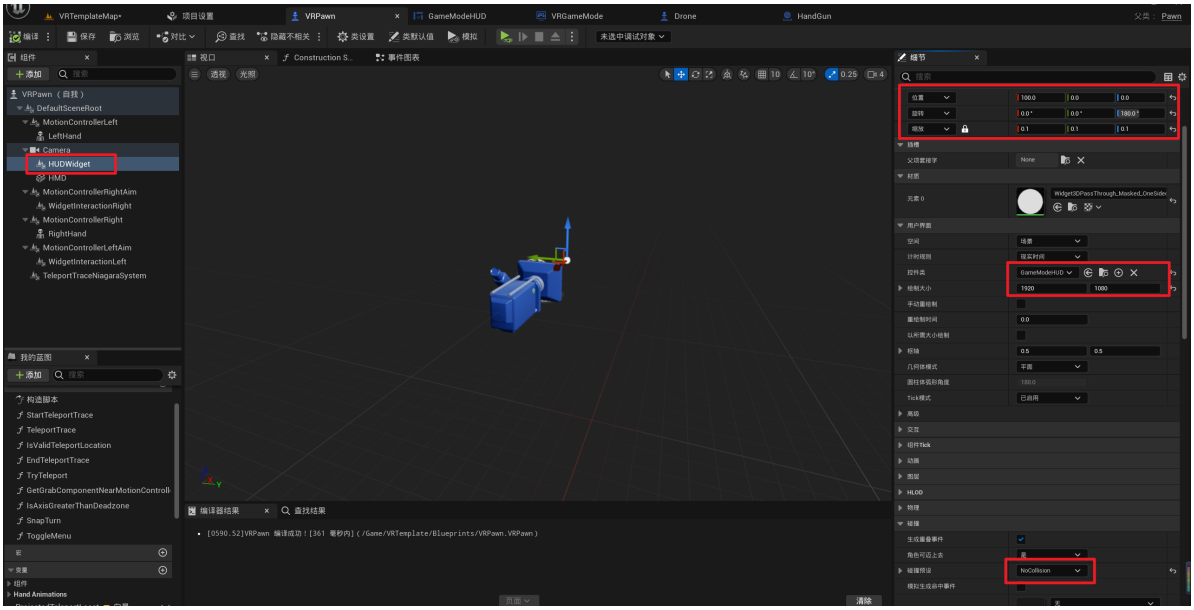
游戏模式

如果所有的敌机都被消灭了，且自身没有被一半的敌机击中，就算赢，否则就算输

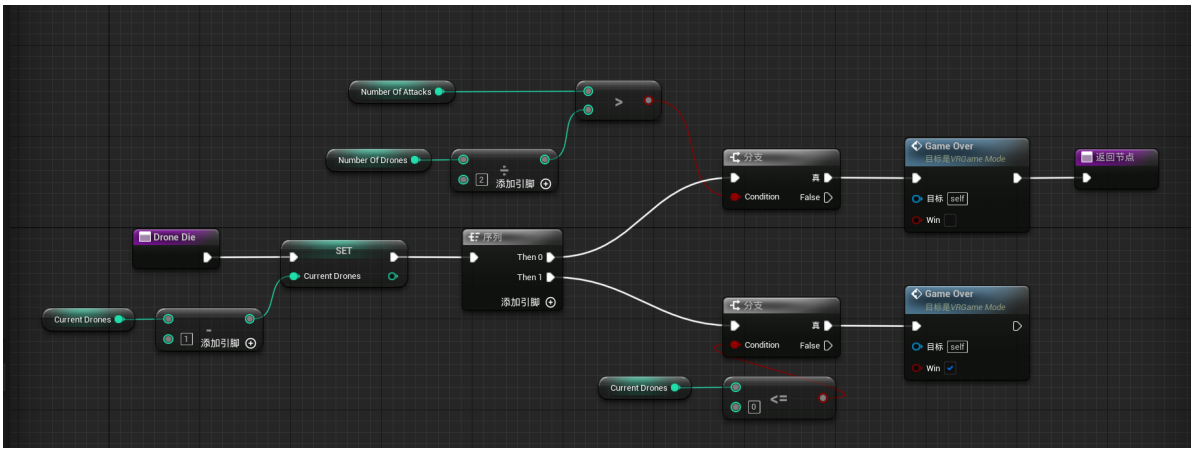
创建一个GameHUD

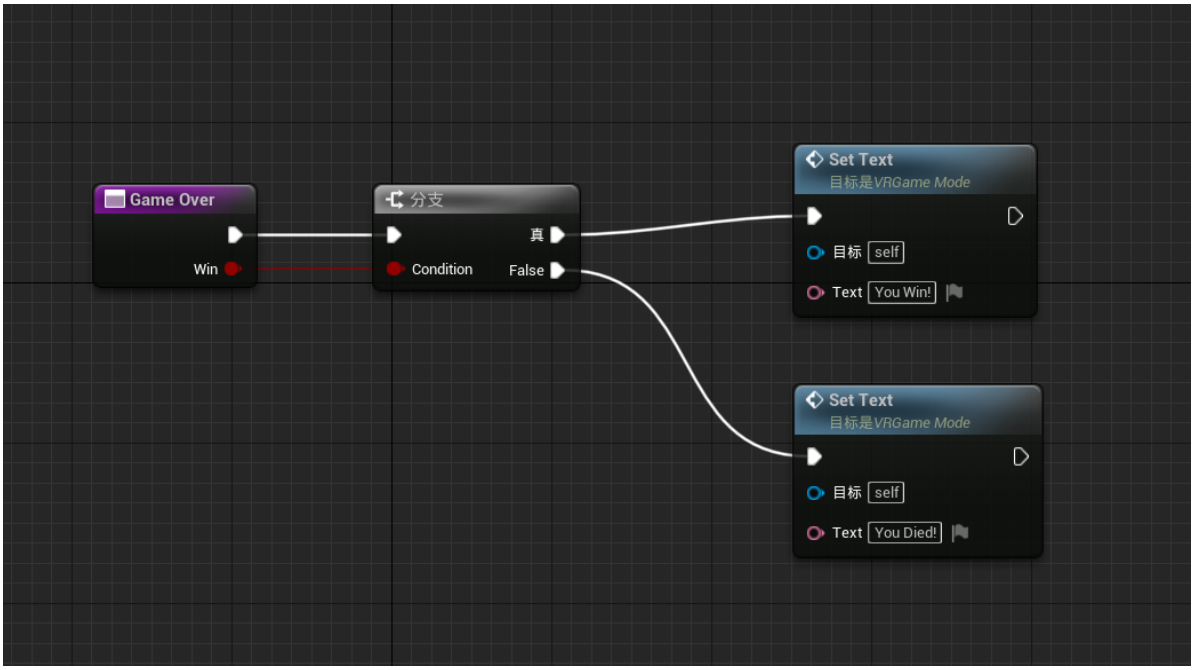


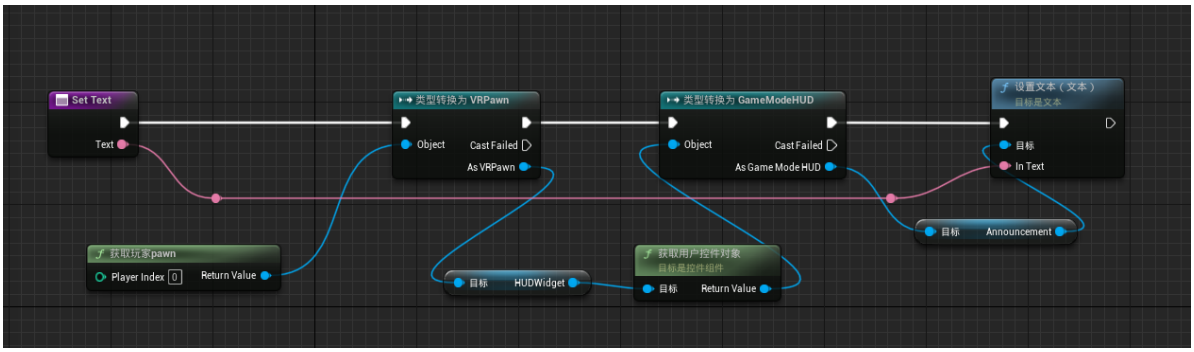
在VRPawn中创建一个控件组件，应用这个GameHUD



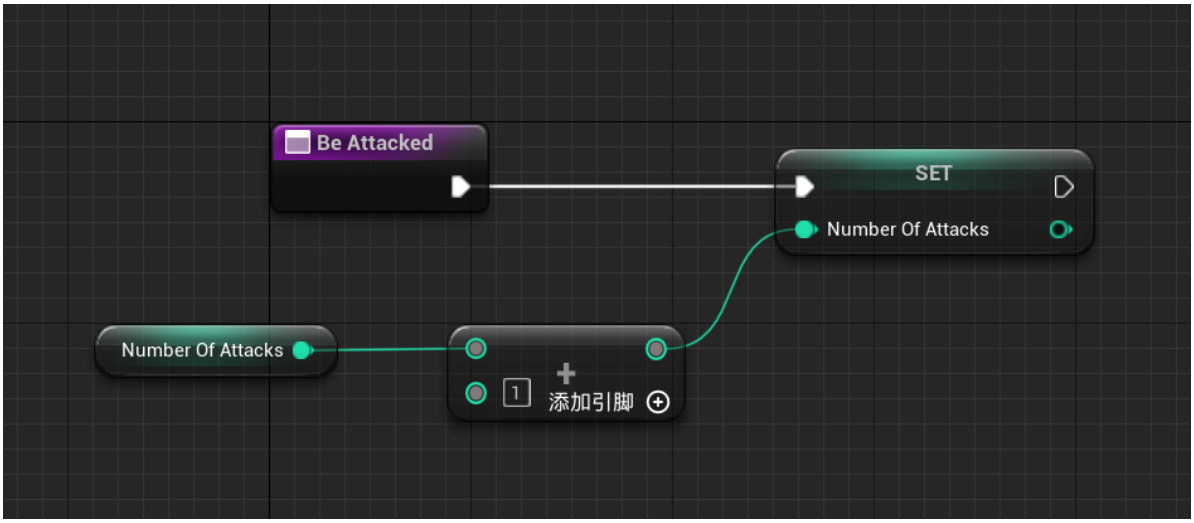
打开默认游戏模式VRGameMode，记录目前剩余的敌机数量和被击中的数量，满足条件时就结束游戏



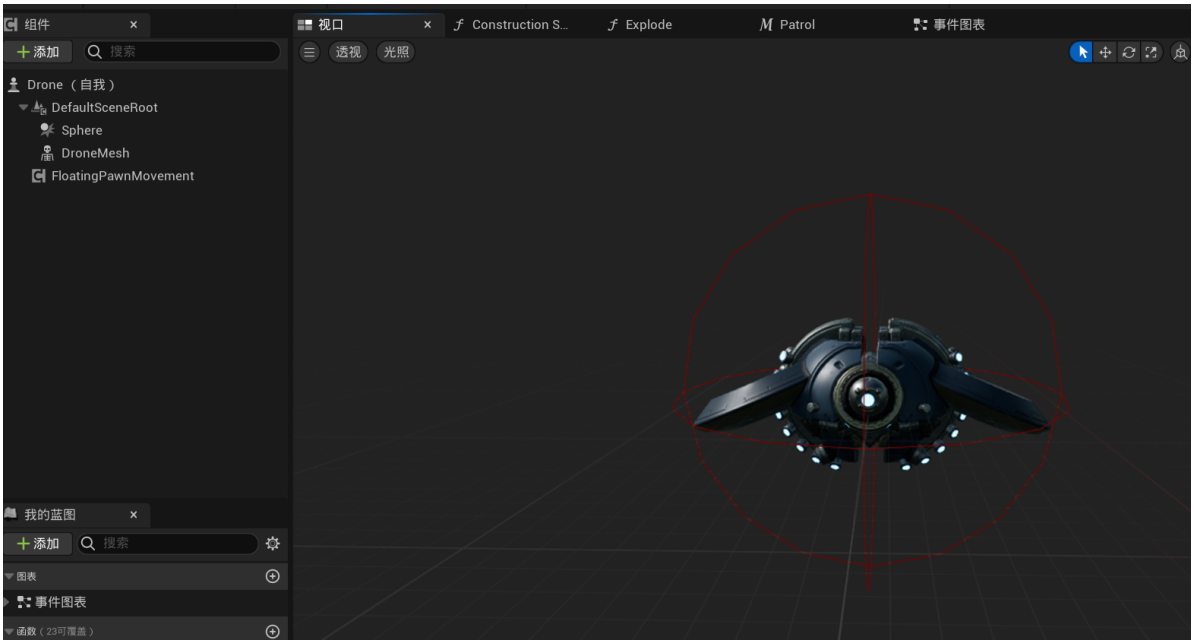


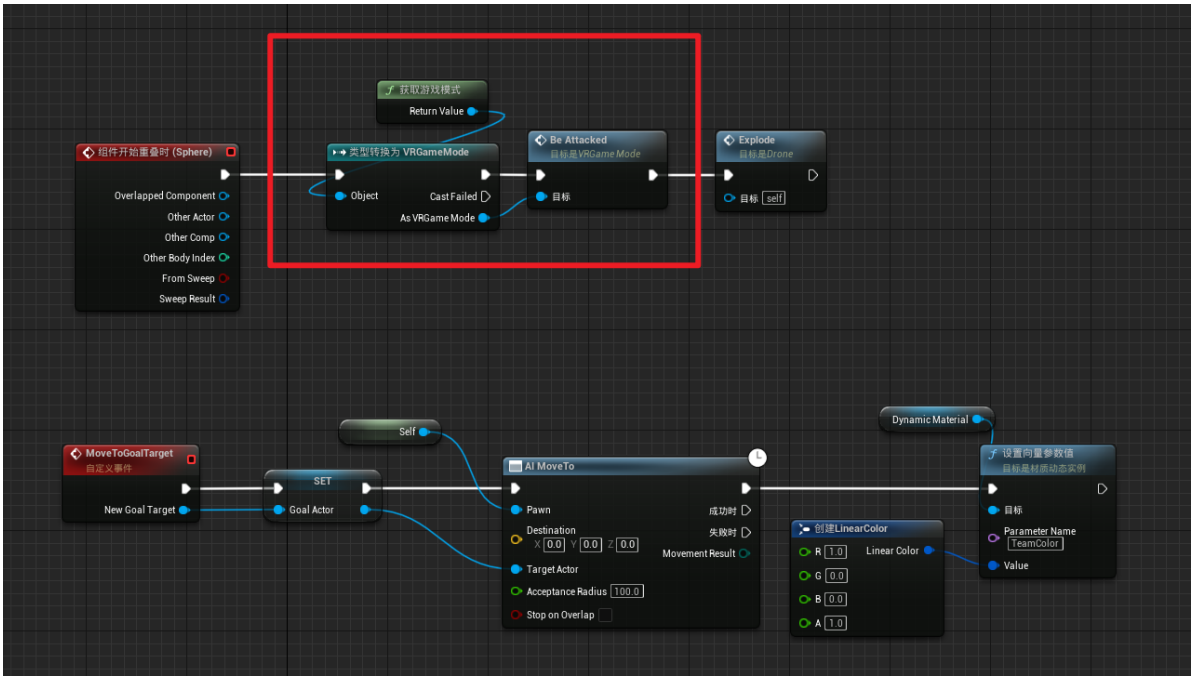


被击中时也需要记录



给Drone创建一个碰撞器

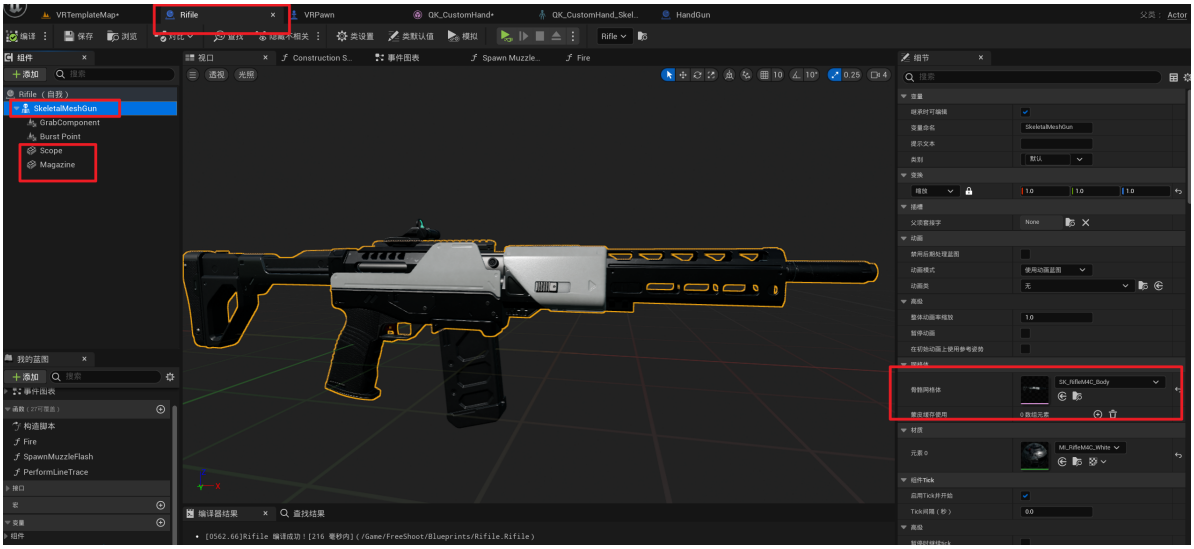




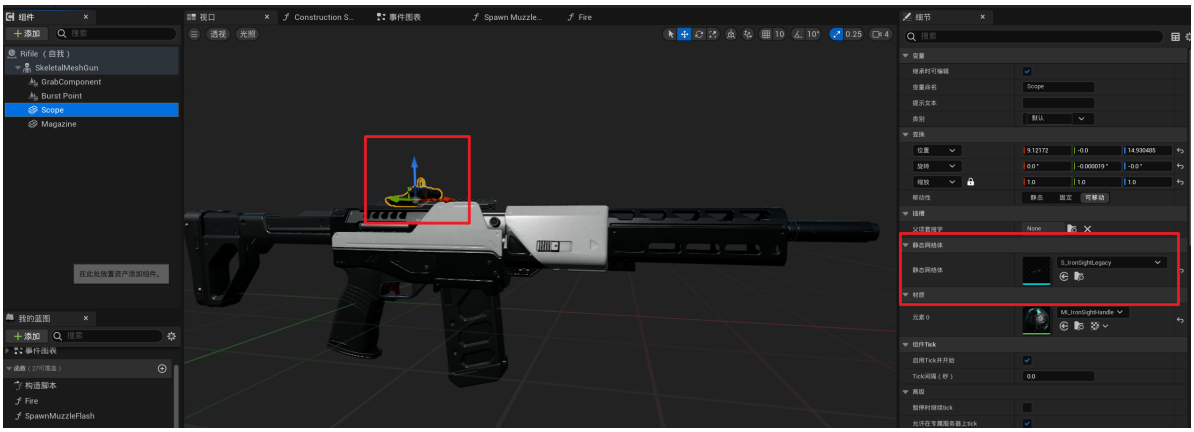
步枪

手枪是单发，步枪是连发的

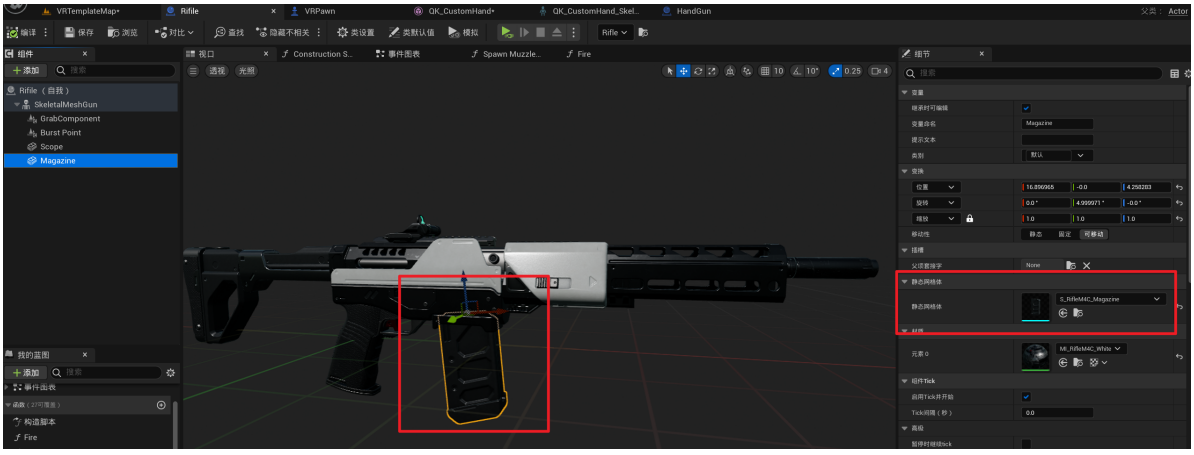
先复制手枪蓝图，配置如下步枪蓝图



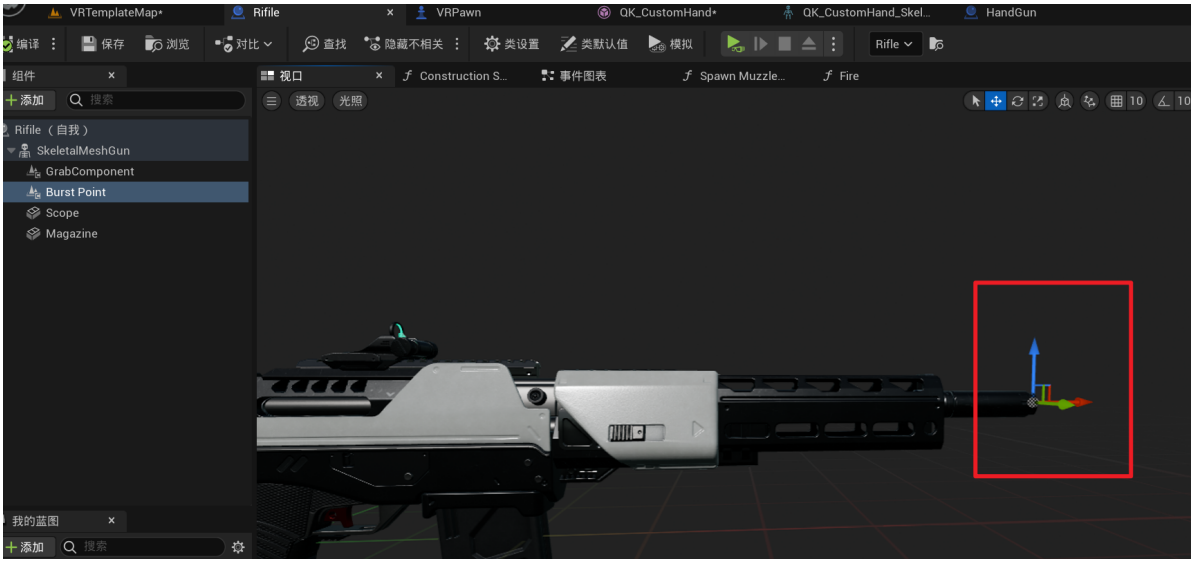
配置瞄准镜



加上弹夹



出射点往前放



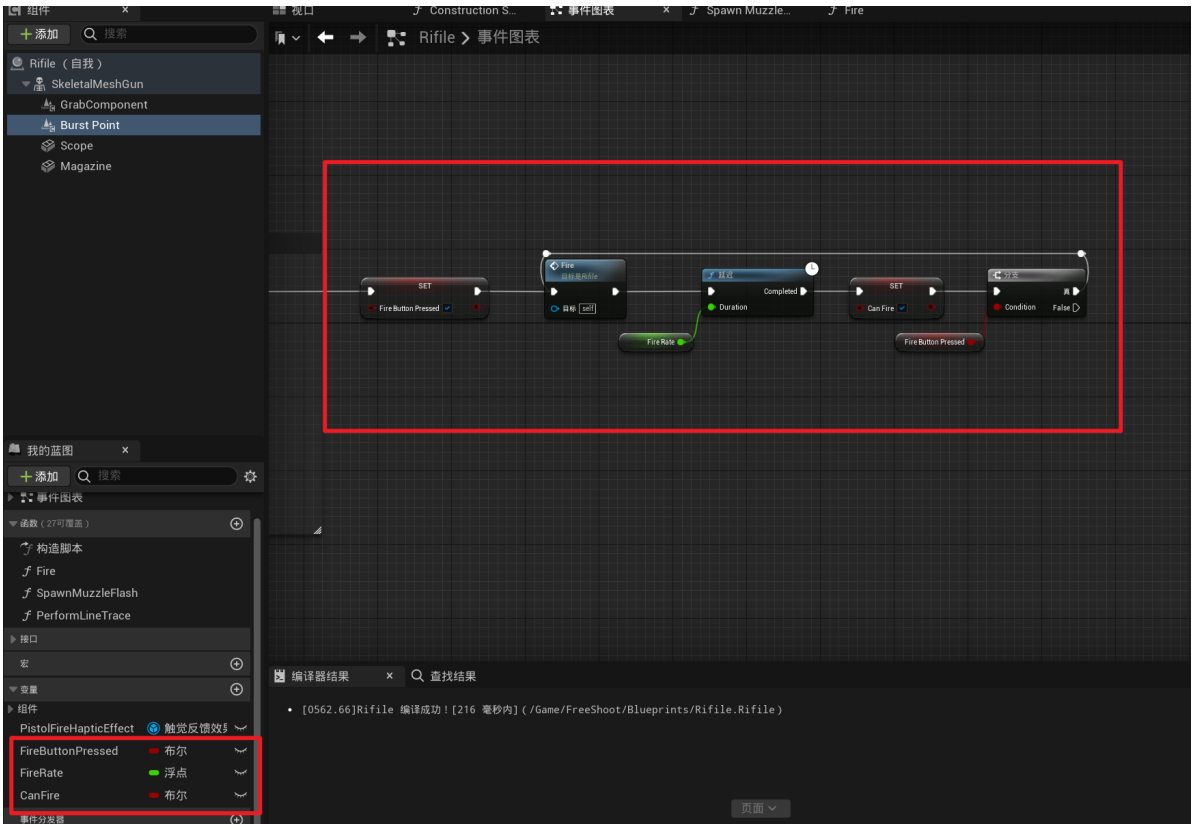
配置手部插槽

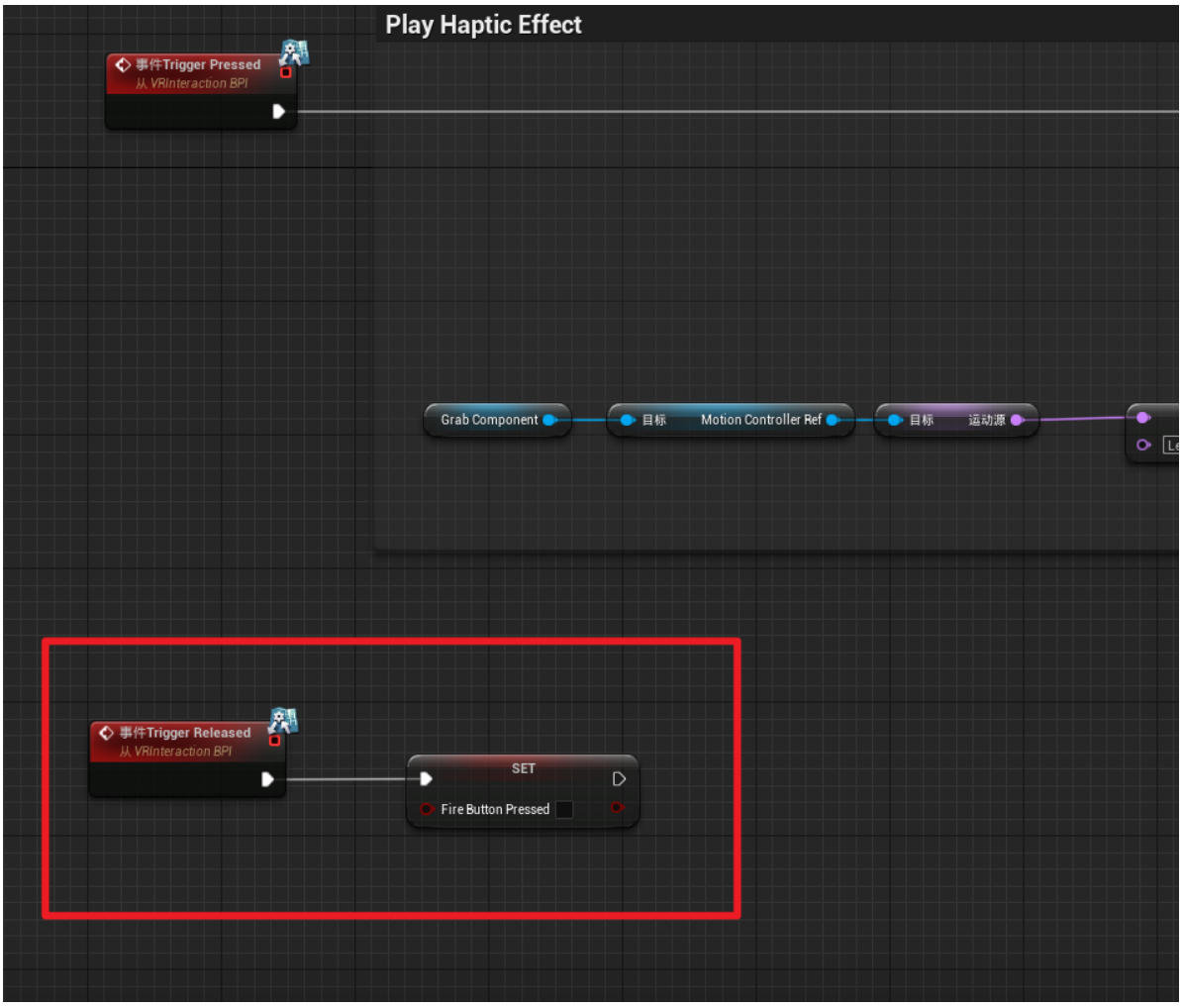
完善开枪逻辑，创建所需变量

其中FireButtonPressed默认为False

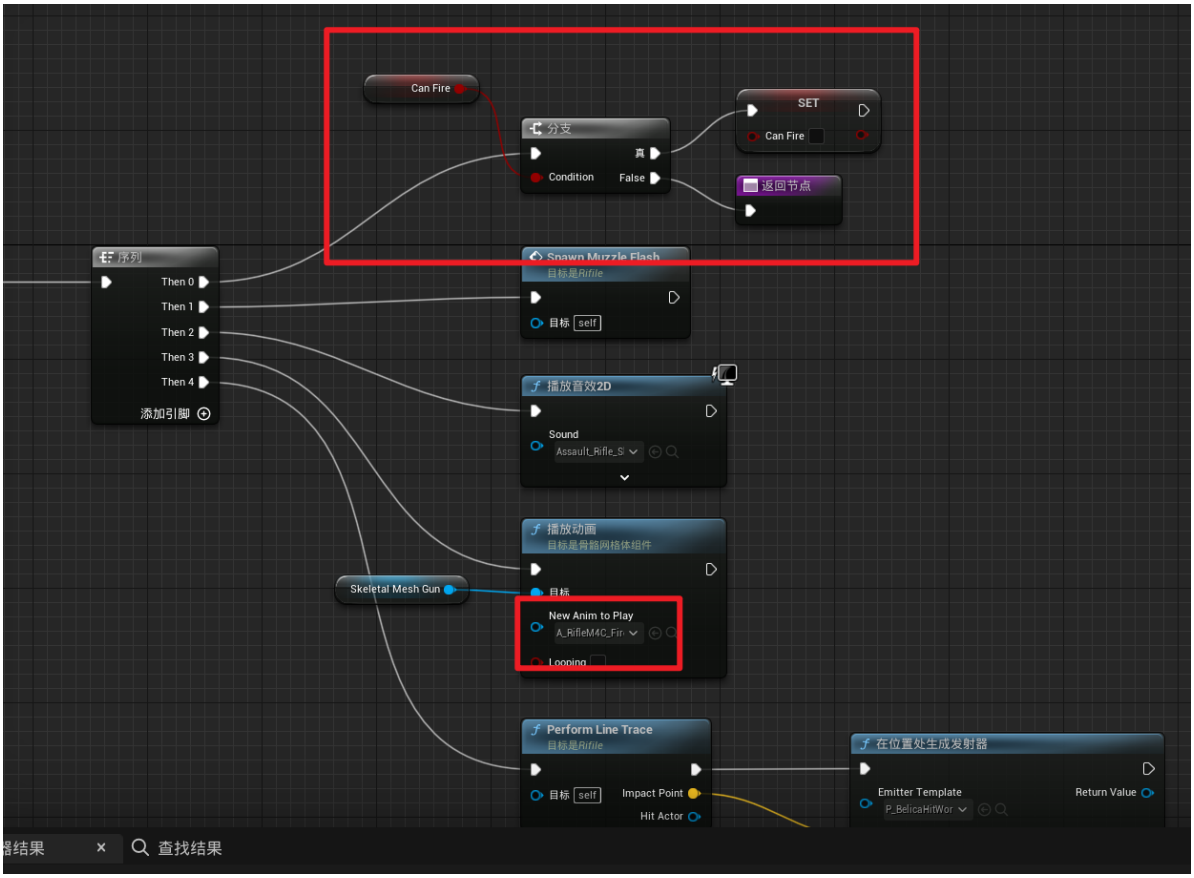
FireRate为0.1

CanFire为True

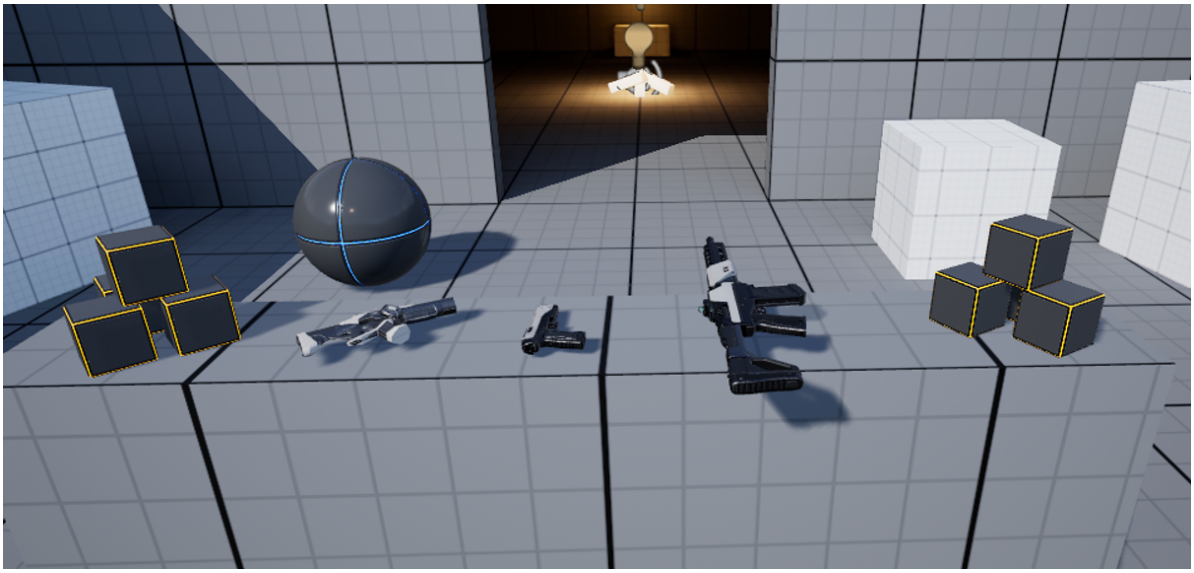

Rifile ▸ 事件图表
Rifile（自我）
SkeletalMeshGun
GrabComponent
Burst Point
Scope
Magazine
SET
Fire Button Pressed
Fire
目标 self
Completed
Duration
Fire Rate
SET
Can Fire
Fire Button Pressed
分支
Condition
False
我的蓝图
添加
搜索
事件图表
函数（27可覆盖）
构造脚本
Fire
SpawnMuzzleFlash
PerformLineTrace
接口
宏
变量
组件
PistolFireHapticEffect
触觉反馈效果
FireButtonPressed
布尔
FireRate
浮点
CanFire
布尔
事件分发器
编译器结果
查找结果
[0562.66]Rifile 编译成功！[216 毫秒内]（/Game/FreeShoot/Blueprints/Rifile.Rifile）
页面


Play Haptic Effect
事件Trigger Pressed
从 VRInteraction BPI
Grab Component
目标
Motion Controller Ref
目标
运动源
事件Trigger Released
从 VRInteraction BPI
SET
Fire Button Pressed

修改Fire函数

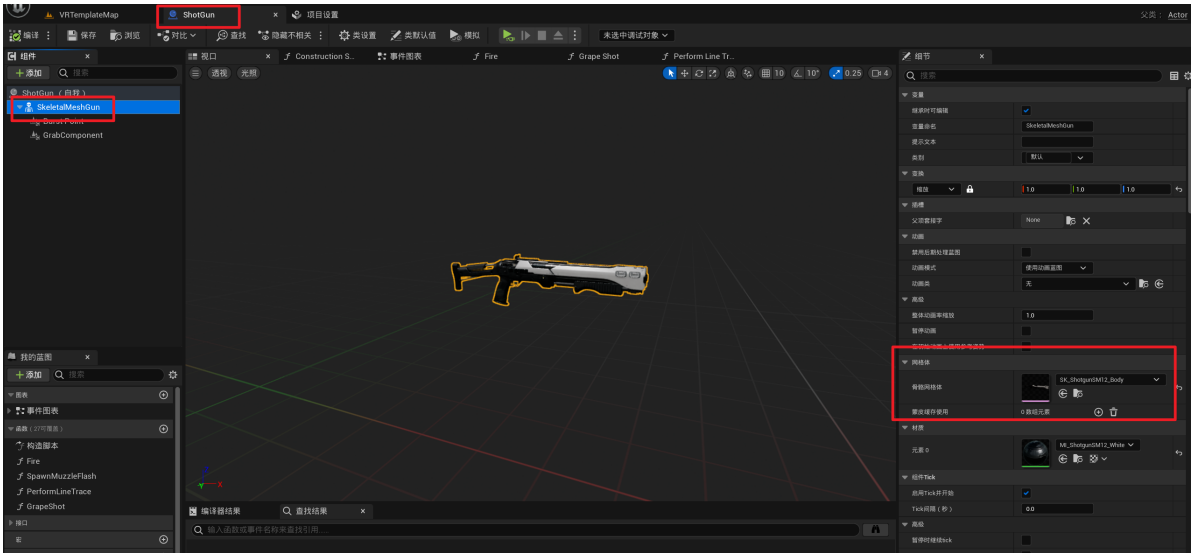


然后放在场景里就可以了

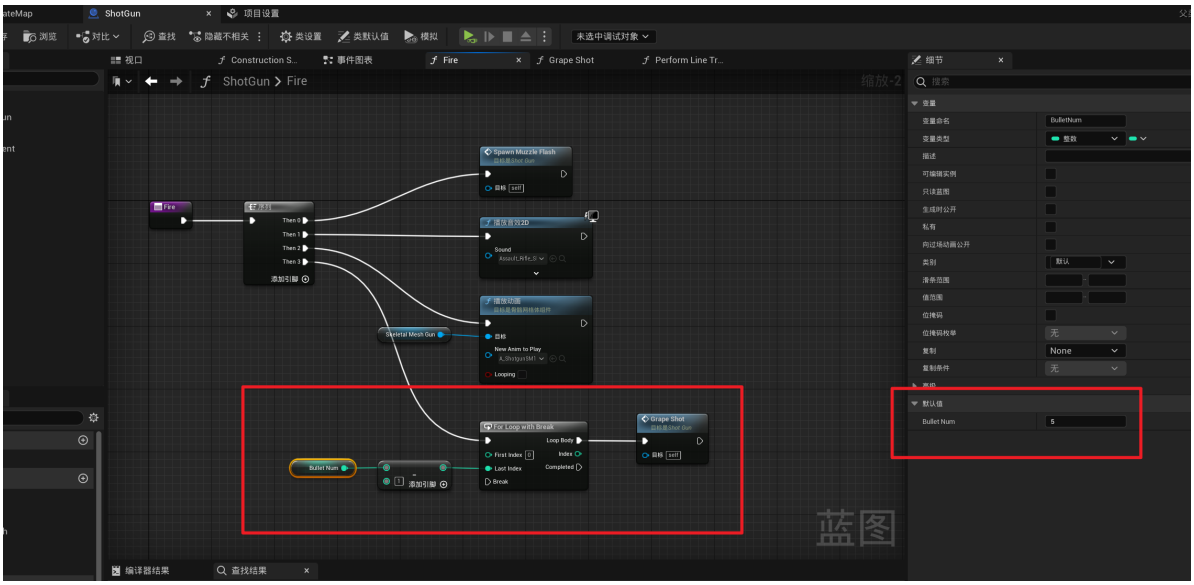


霰弹枪

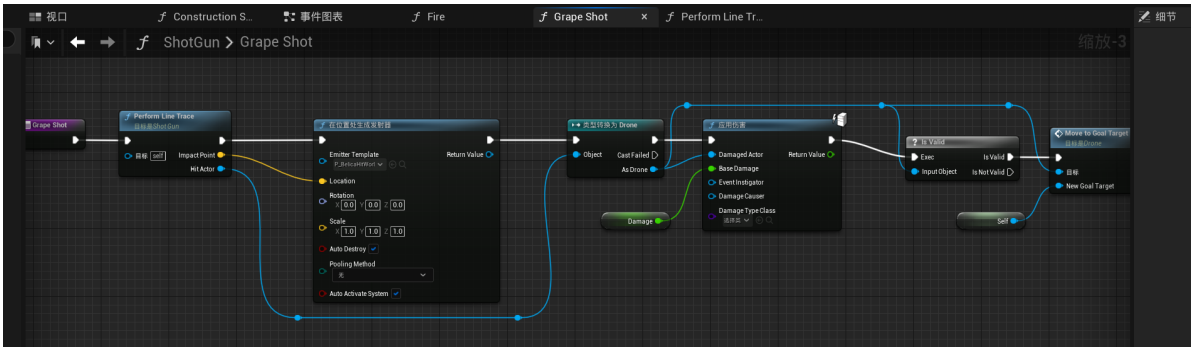
复制手枪蓝图，修改对应骨骼



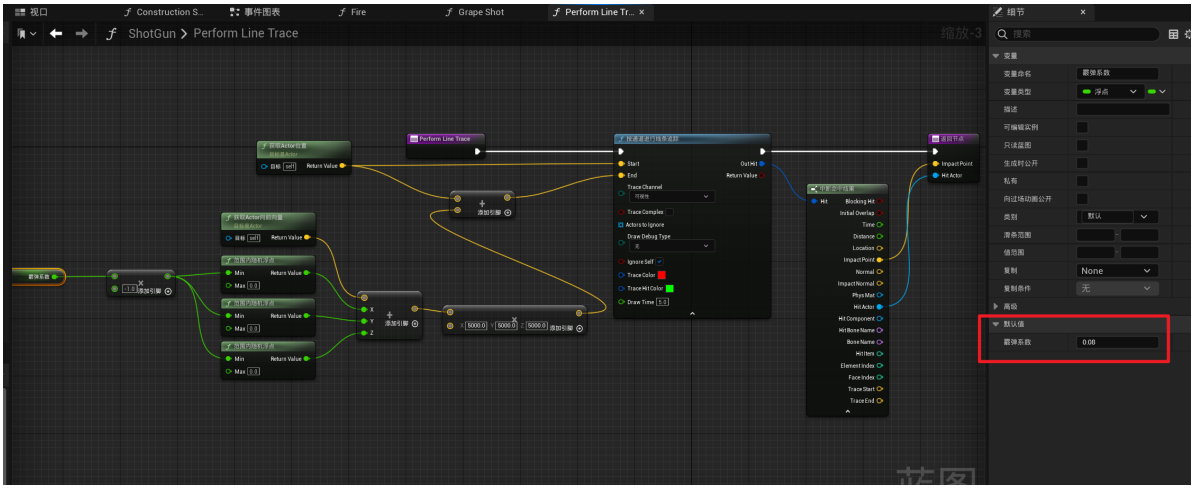
修改Fire函数，把下面的原有的代码都折叠成Grape Shot函数，然后添加循环发射代码



下面这一段基本不变

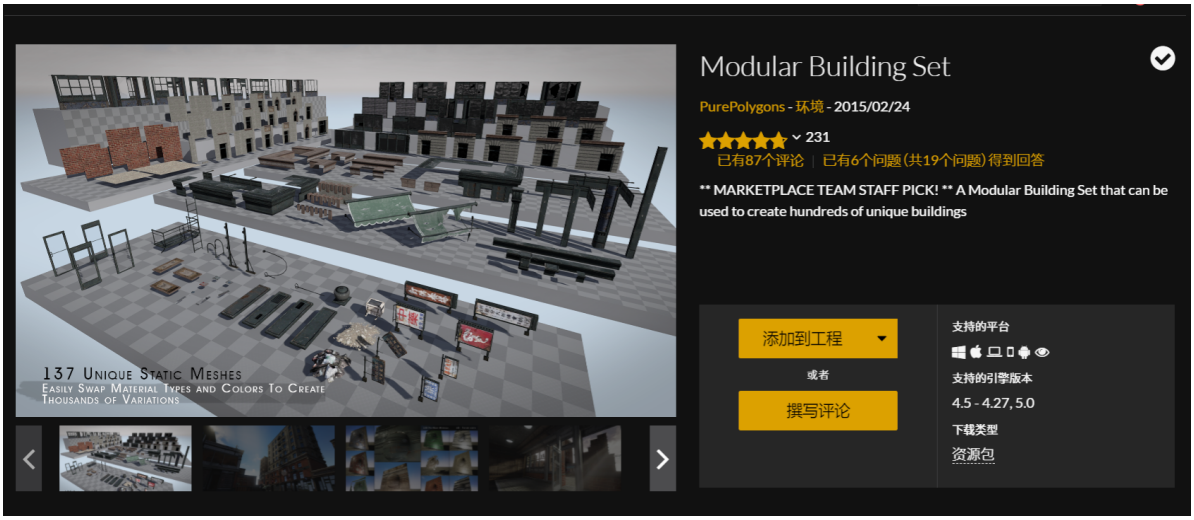


修改修改Perform Line Trace函数，生成随机子弹路径，其中霰弹系数代表子弹之间的稀疏程度，可自行调节

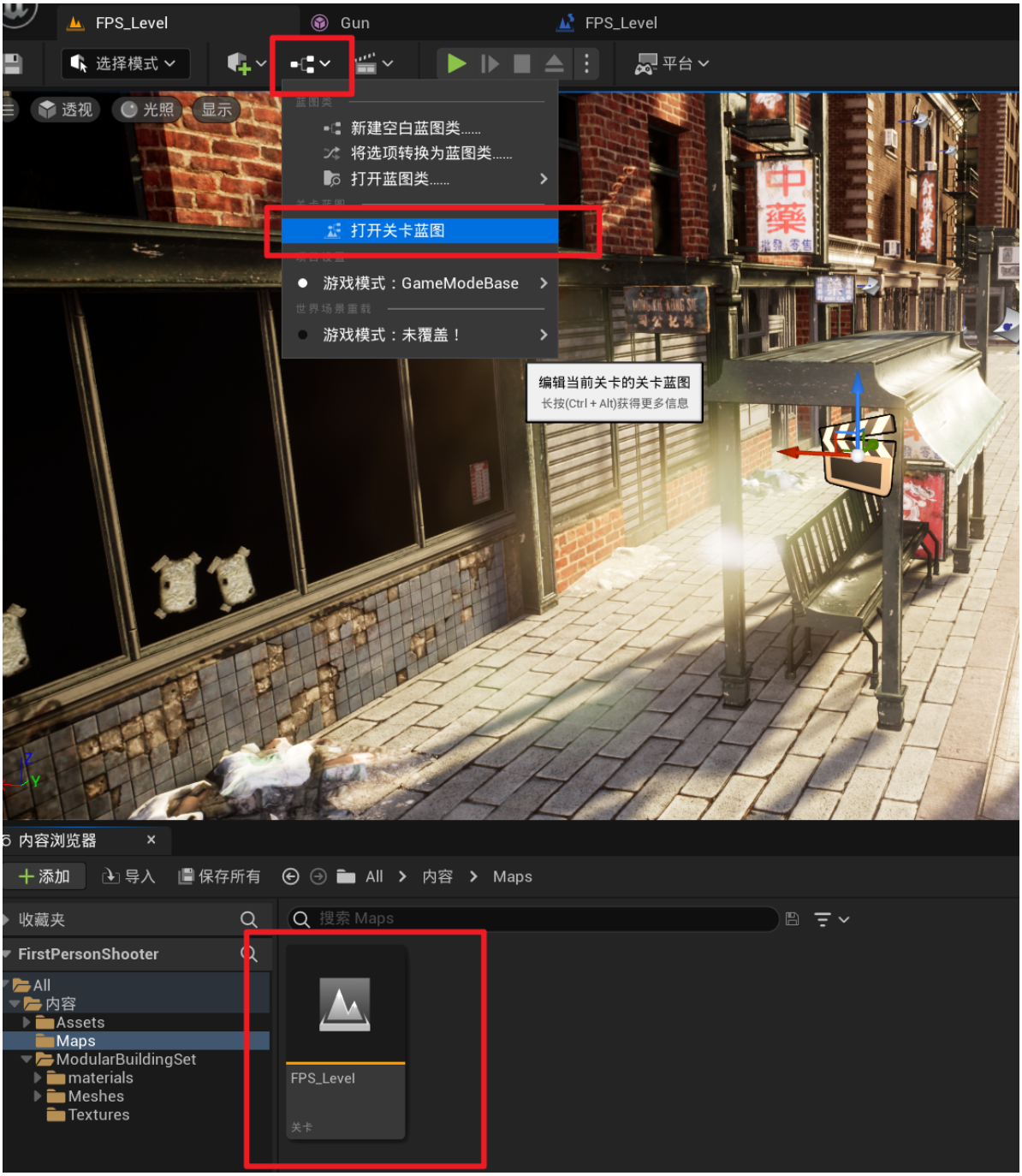


配置场景

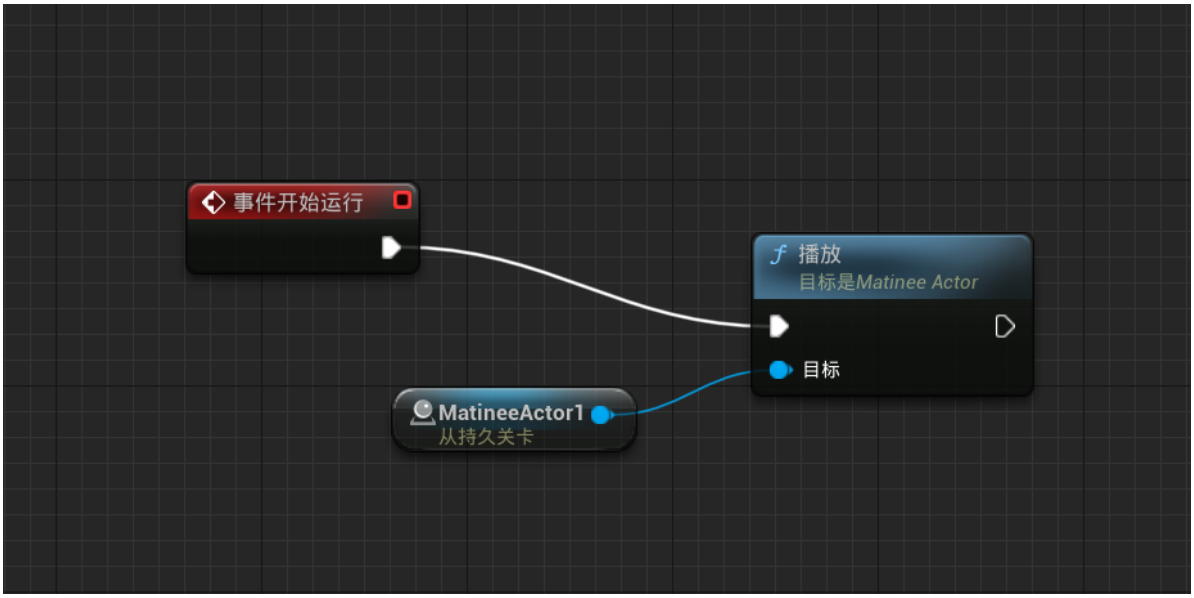
在虚幻市场中下载这个场景，并将之添加到初始项目里



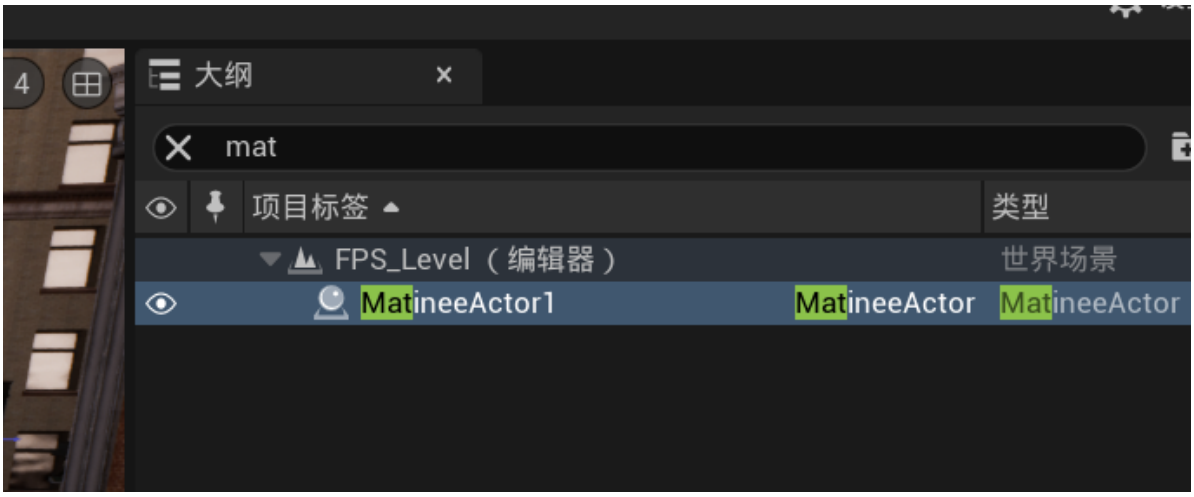
打开这个地图的关卡蓝图


FPS_Level
Gun
FPS_Level
选择模式
平台
透视
光照
显示
新建空白蓝图类.....
将选项转换为蓝图类.....
打开蓝图类.....
打开关卡蓝图
游戏模式：GameModeBase
游戏模式：未覆盖！
编辑当前关卡的关卡蓝图
长按(Ctrl + Alt)获得更多信息
内容浏览器
添加
导入
保存所有
All
内容
Maps
收藏夹
搜索 Maps
FirstPersonShooter
All
内容
Assets
Maps
ModularBuildingSet
materials
Meshes
Textures
FPS_Level
关卡

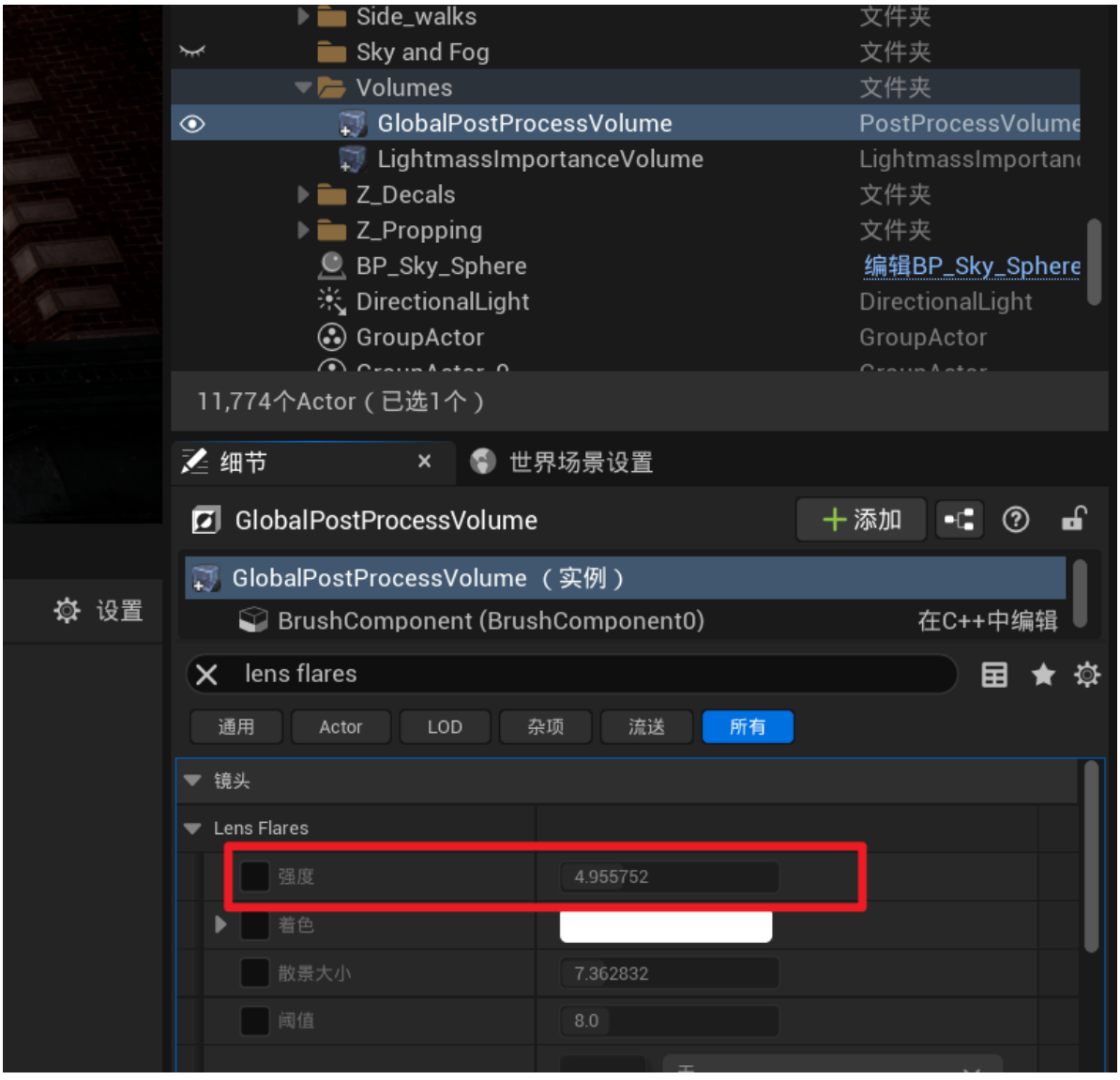
把下面这一段都删掉


事件开始运行
播放
目标是Matinee Actor
目标
MatineeActor1
从持久关卡

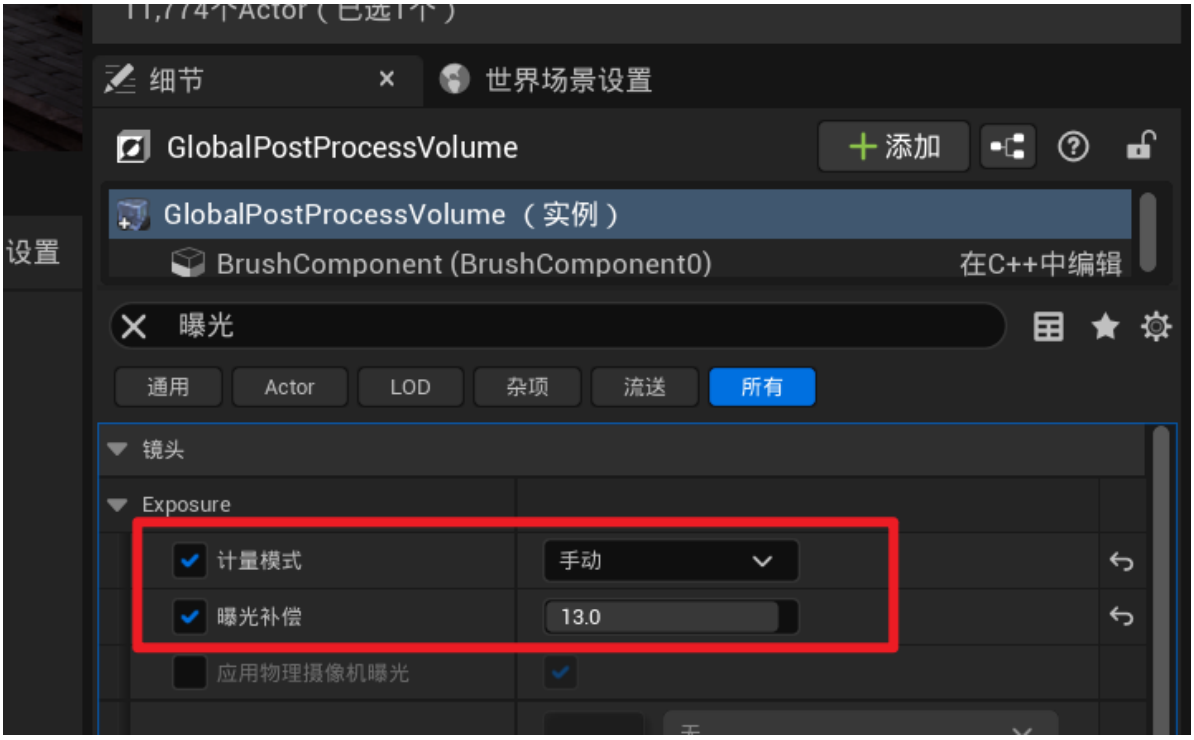
然后把这个电影序列也删掉



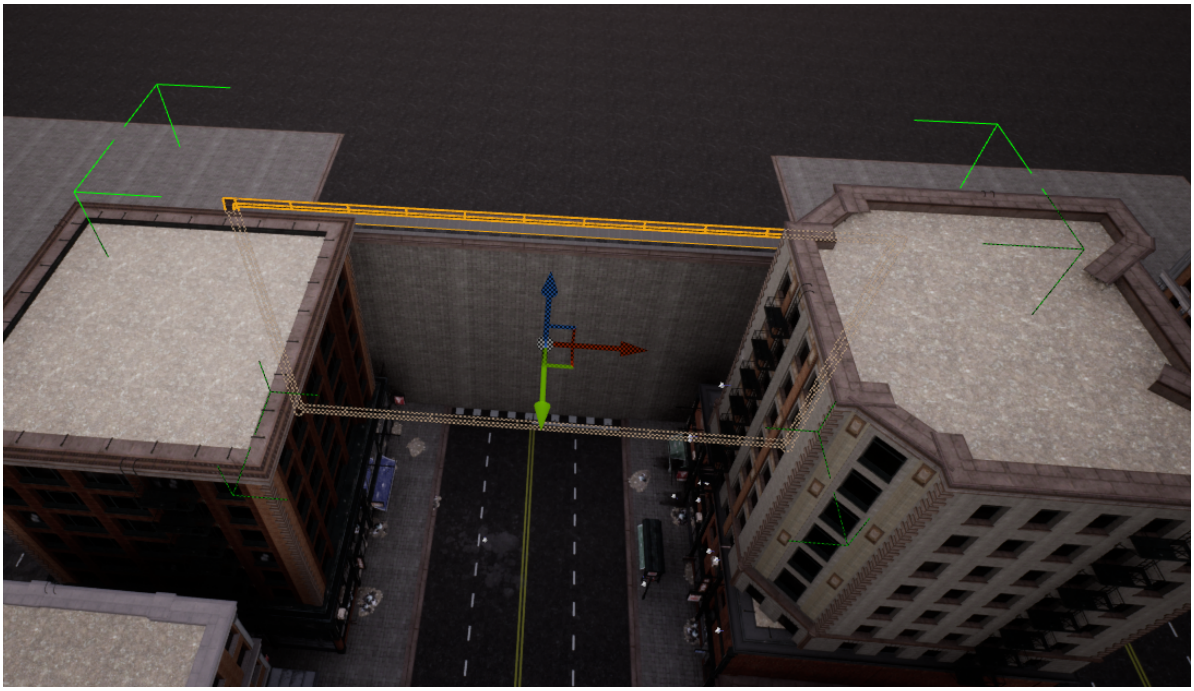
找到全局后期处理体积，把lens flares的强度取消，这样镜头前就不会出现光晕



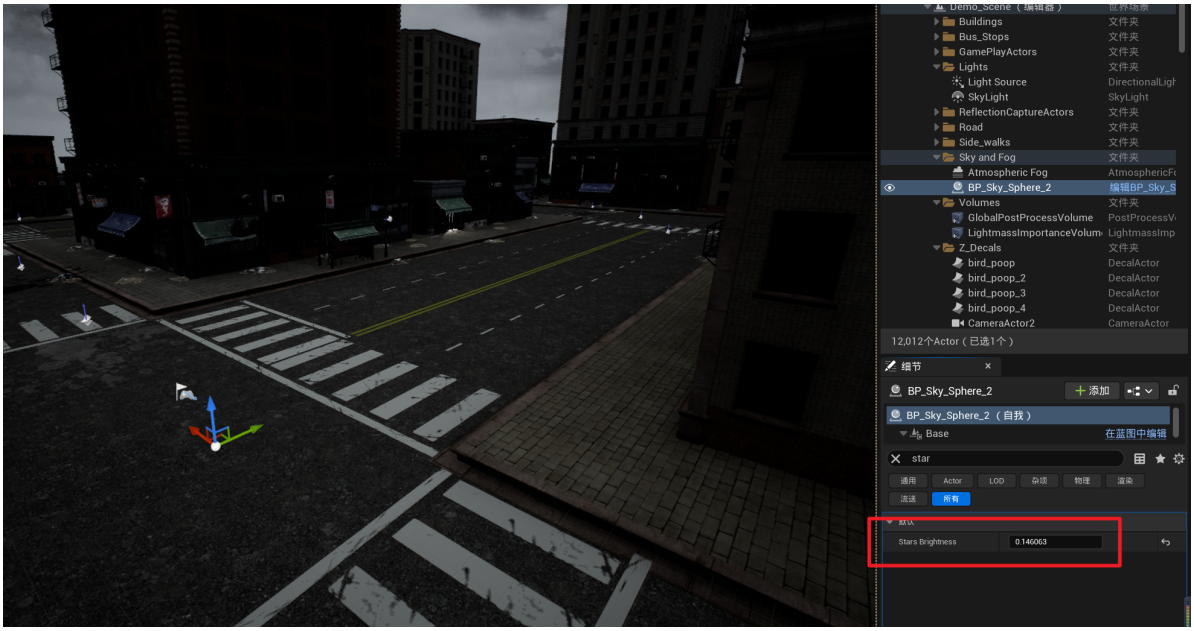
取消自动曝光



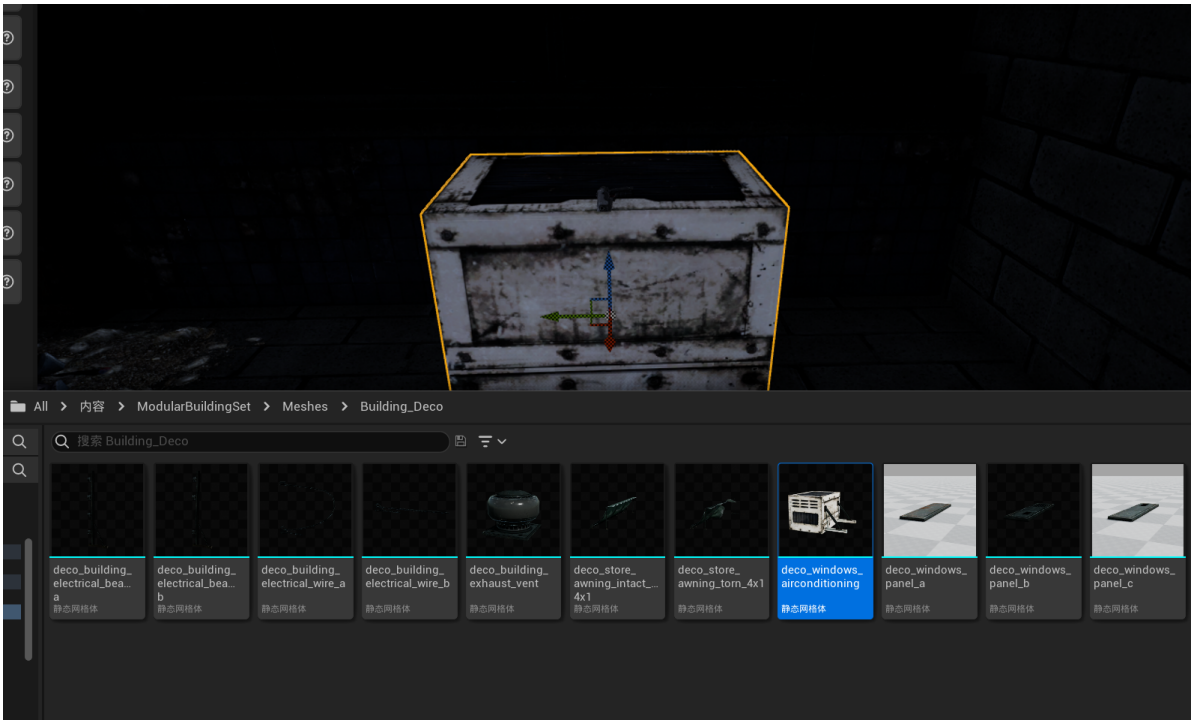
复制一下外面的地面，把胡同口堵住，形成一堵墙

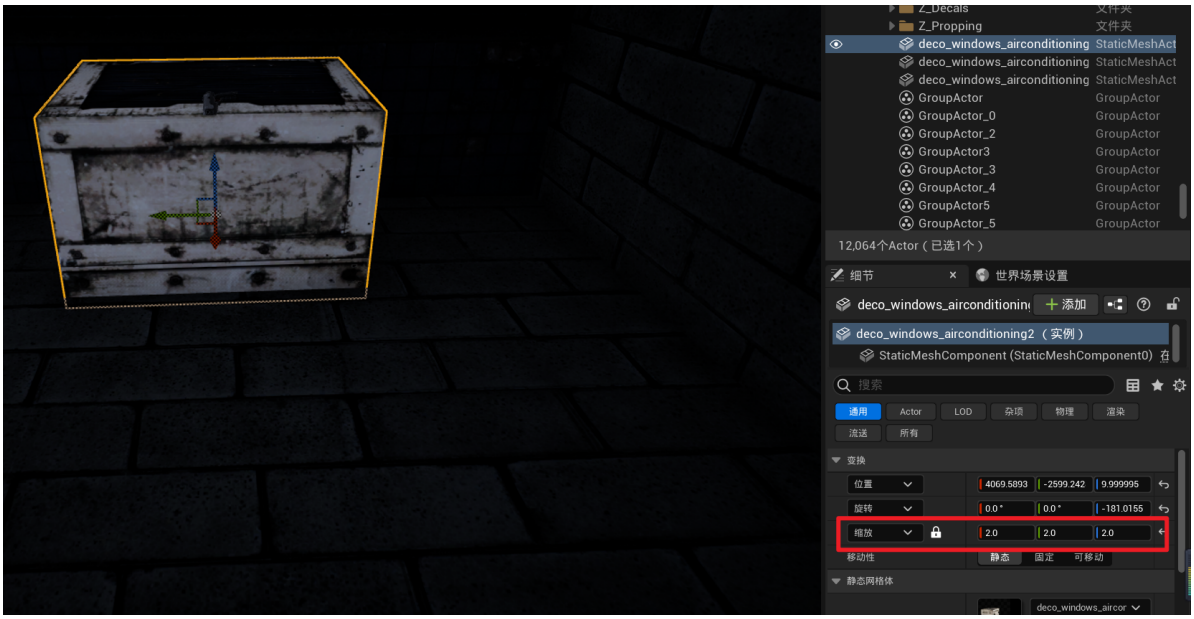


调节星星的亮度

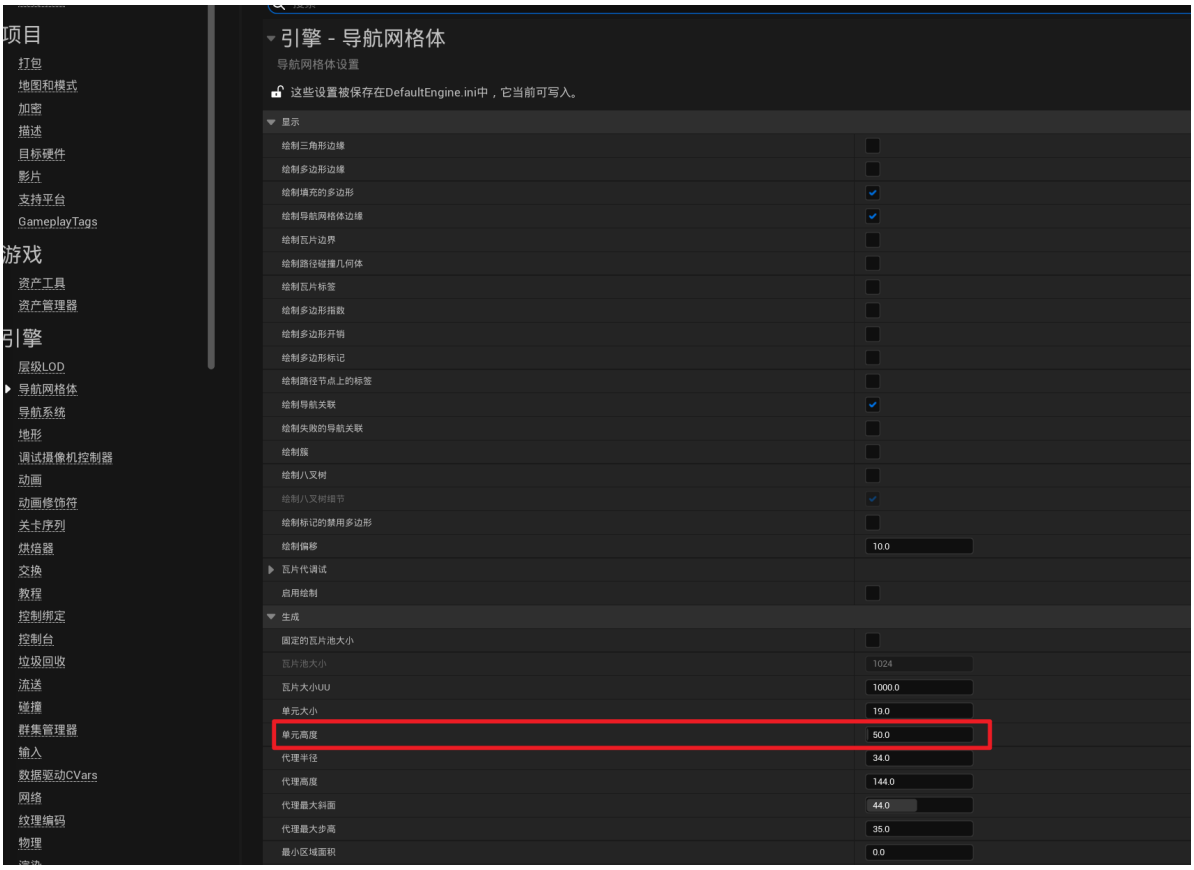


在场景中放三个箱子，在箱子上分别放置手枪、步枪和霰弹枪



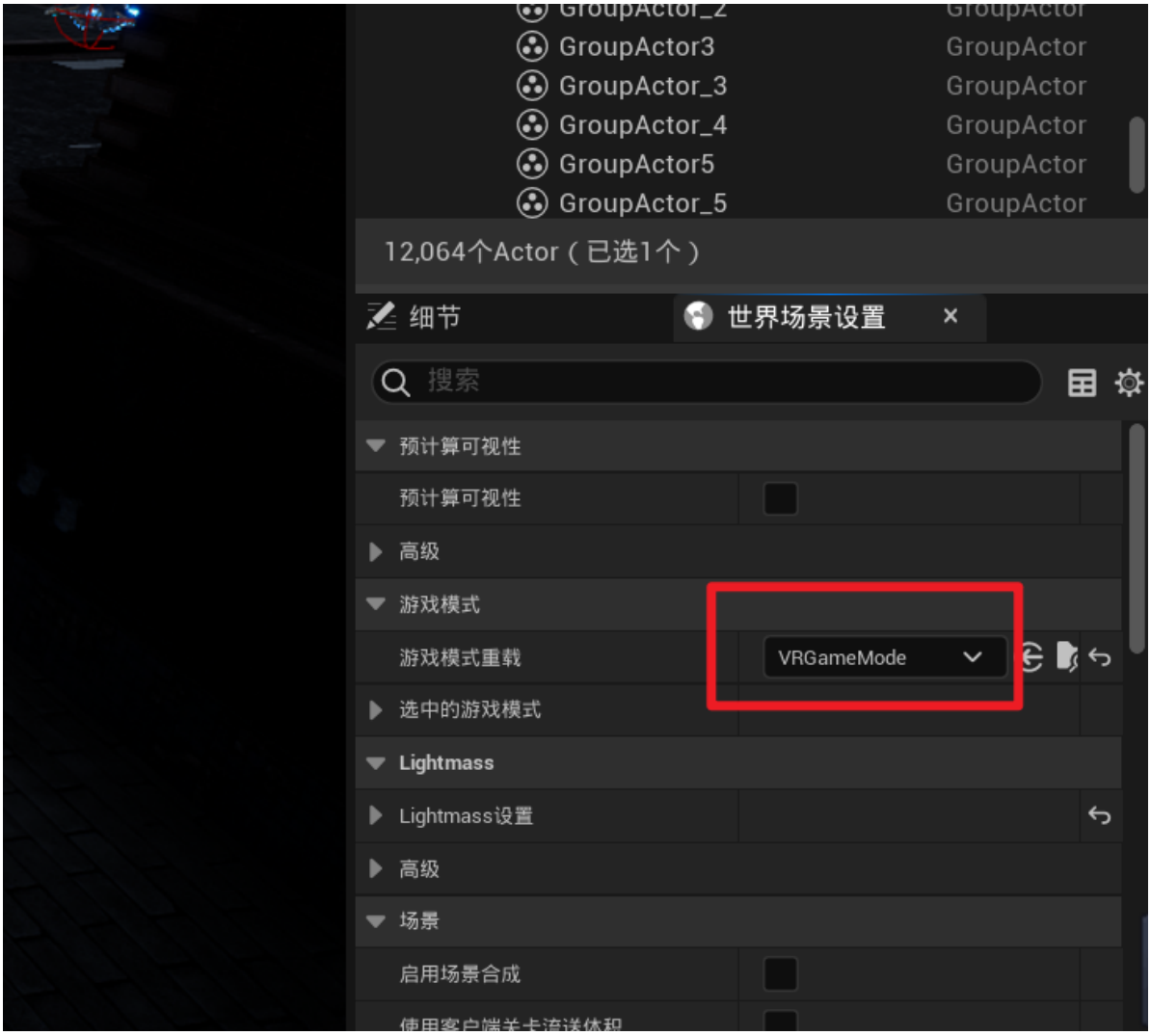


给场地上铺一层导航网格，设置单元高度为50



再给场景加上一些Drones

最后指定相应的游戏模式


GroupActor3
GroupActor
GroupActor_3
GroupActor
GroupActor_4
GroupActor
GroupActor5
GroupActor
GroupActor_5
GroupActor
12,064个Actor（已选1个）
细节
世界场景设置
搜索
预计算可视性
预计算可视性
高级
游戏模式
游戏模式重载
VRGameMode
选中的游戏模式
Lightmass
Lightmass设置
高级
场景
启用场景合成

整个游戏就大功告成了！