UE5.4基础知识学习

1.1 创建项目、修复编译错误以及推荐配置和IDE的选择

1.2 如何在虚幻中Debug我们的C++代码

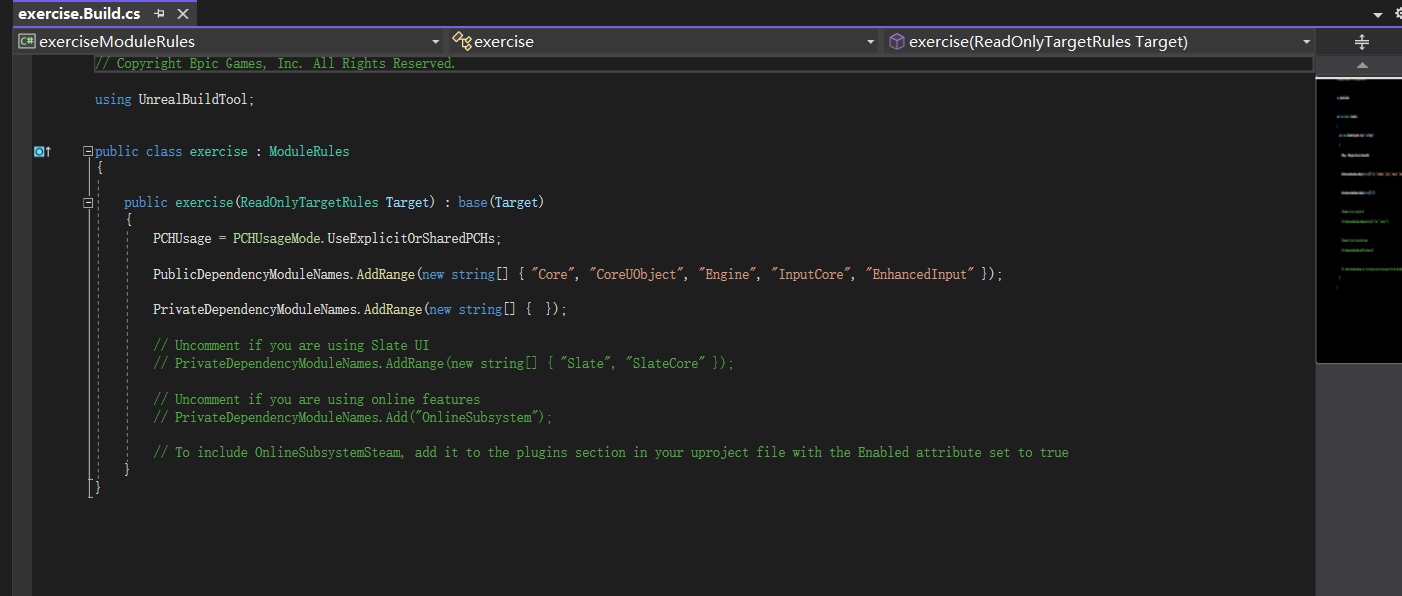
1.3 虚幻中的C++与蓝图的关系

1.设置 ‘Asset Editor Open Location’为‘Main Window’

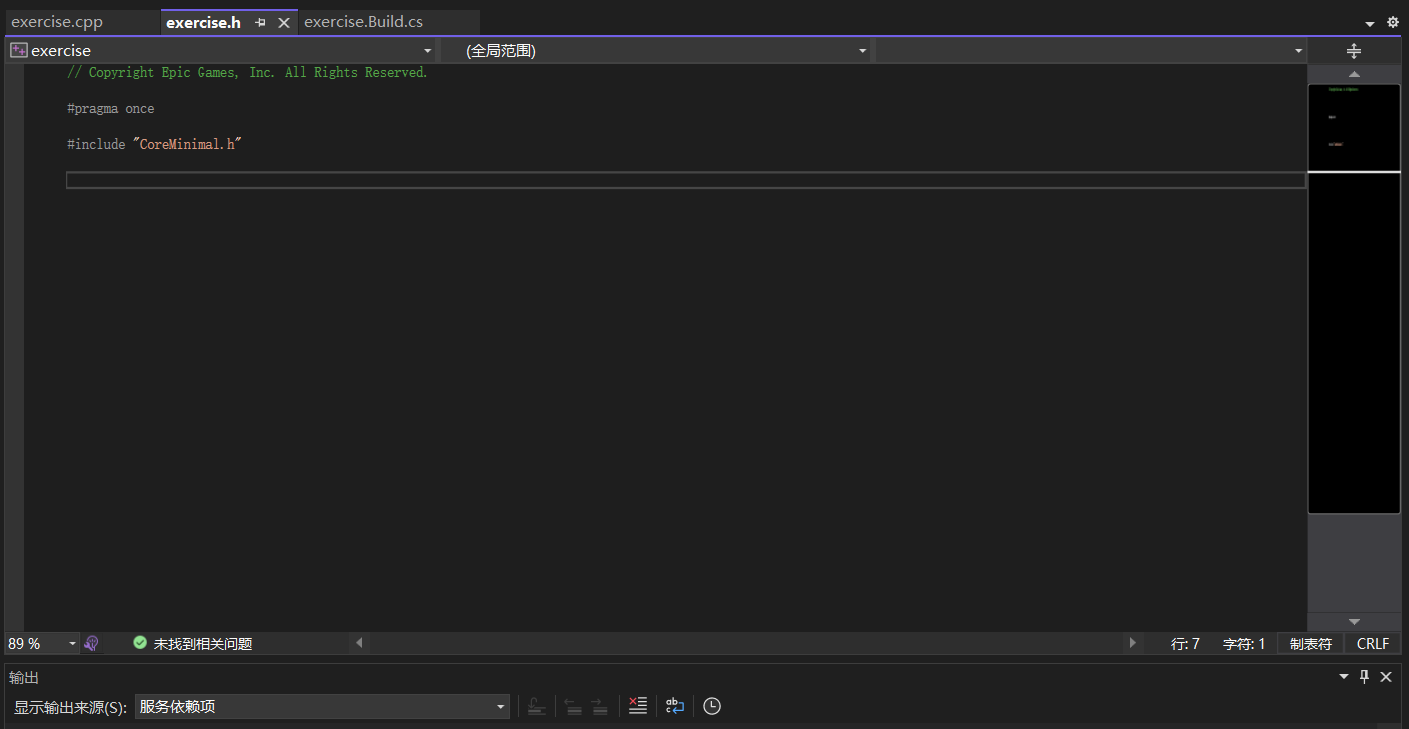
2.设置 Live Coding 中的 Enable Live Coding 为 false

3.如果是通过 ‘First Person’或是 ‘Third Person’‘Top Down’等有默认的代码生成的模板生成的项目，将头文件中的属性的指针(\*)改为TObjectPtr<T>





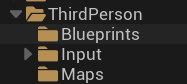
添加模块的c#文件，定义了项目的模块，可以在该文件中配置编译选项和依赖关系



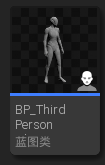
项目头文件：1.声明项目主要类 2.引用其他头文件，库，模块 3.自定义接口

创建两个项目，一个为空一个为第三人称项目进行对比：

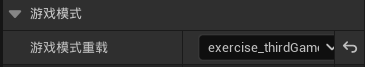
Third项目多出了以下



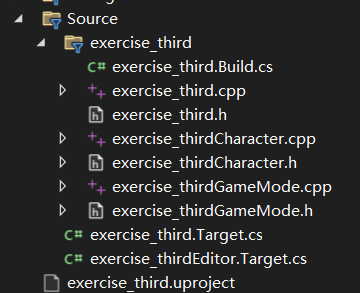
Bluepringts：蓝图类 input：输入之后的动作 maps：？地图关卡

关键点：



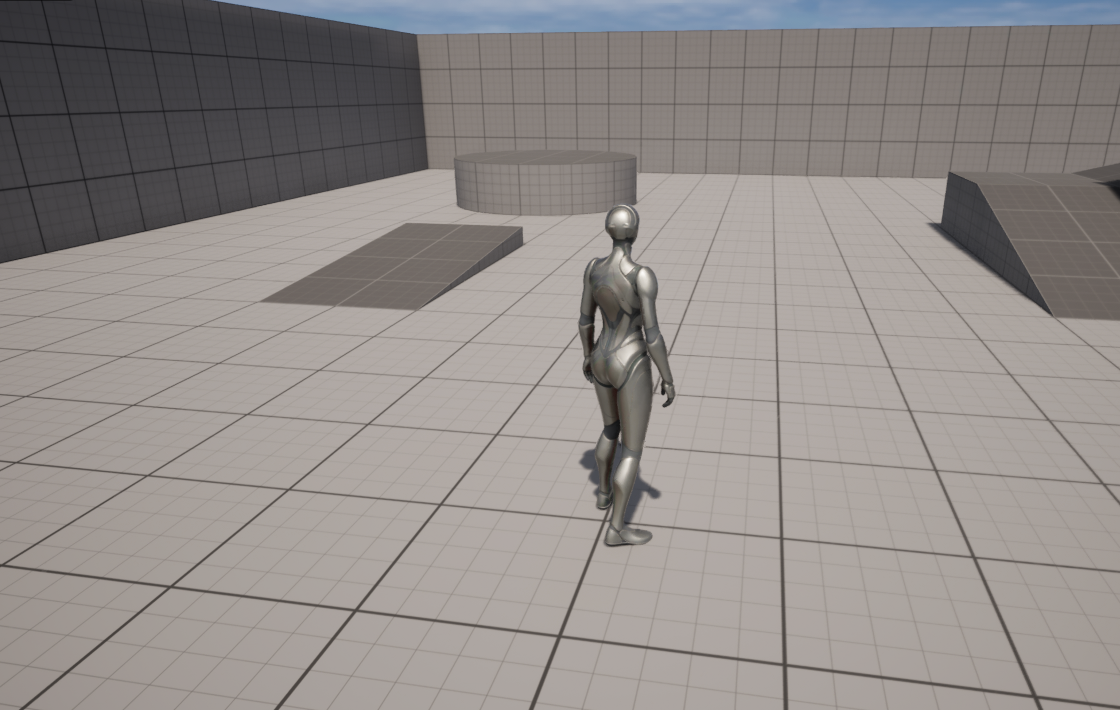
游戏模式中为thirdGamemod，用于控制游戏输入输入之类。

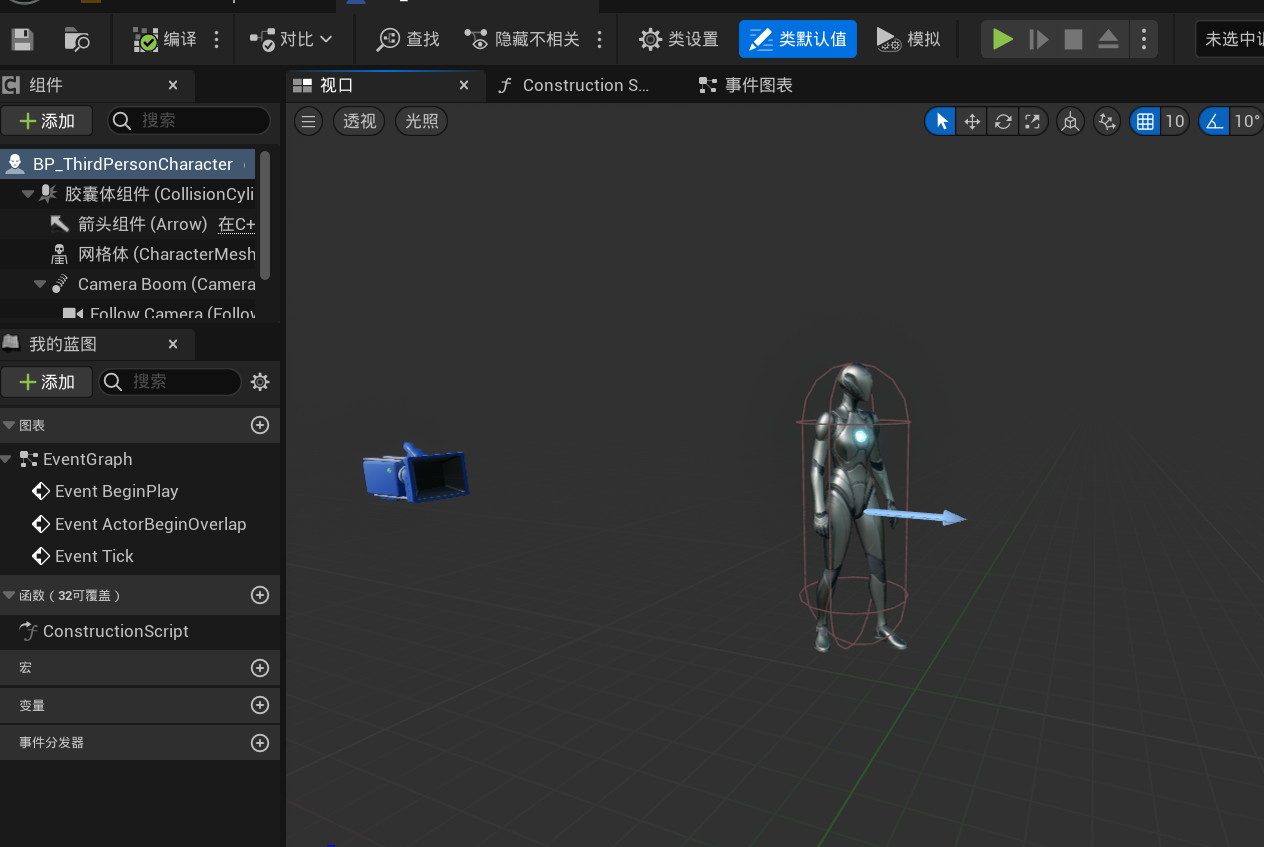


与之对应的，在源码部分会多出thirdGamemode的.cpp，

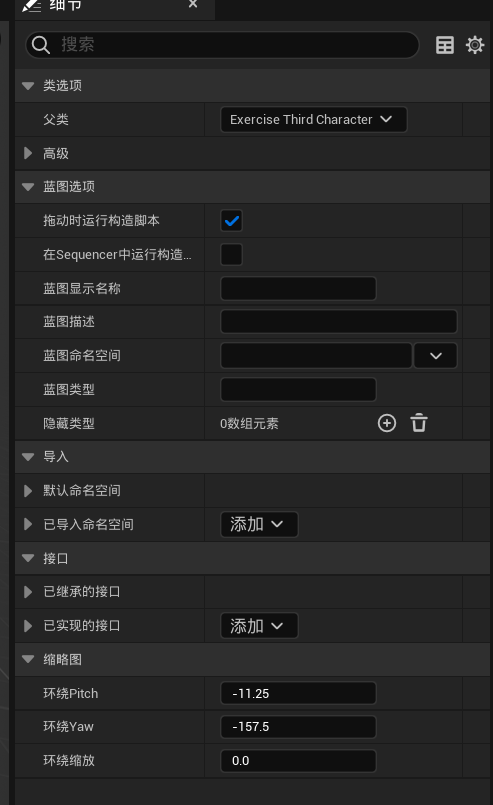
同时可以看到在该cpp文件中，PlayerPawnBPClass 会有明确的一个文件路径，这个PlayerPawnBPClass是项目第三人称的我们所控制的角色类，查看文件路径是指向上面所提到的蓝图文件下的所控制的游戏角色，该蓝图类的父类是一个c++类



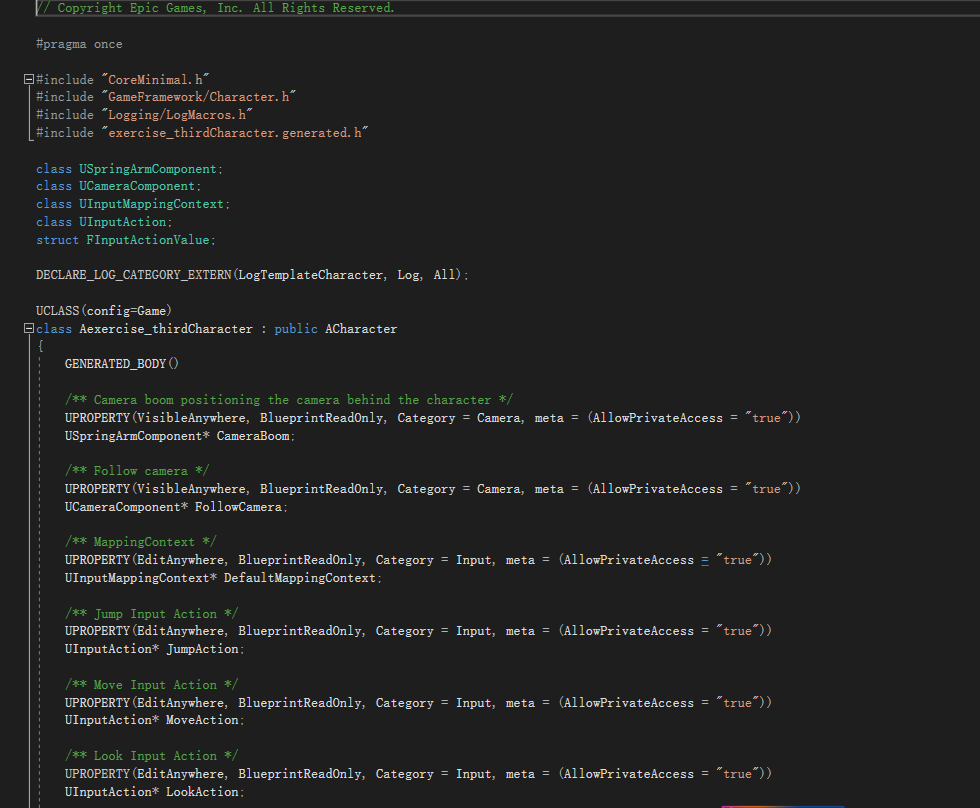




可以通过类设置对其进行修改

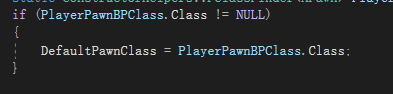


可以查看到该gamemode的具体代码：



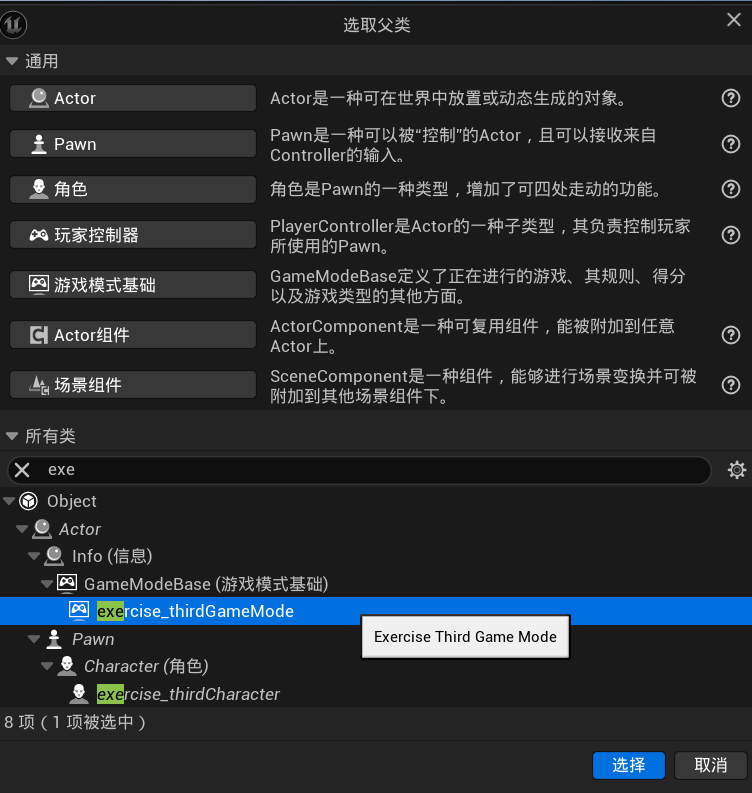
Thirdgamemode缺点：该对象类文件路径在样例中写死，并不适合具体在游戏开发中的情况。同时在具体对控制对象进行修改时，由于样例给出的gamemode直接是cpp，无法直接进行修改。

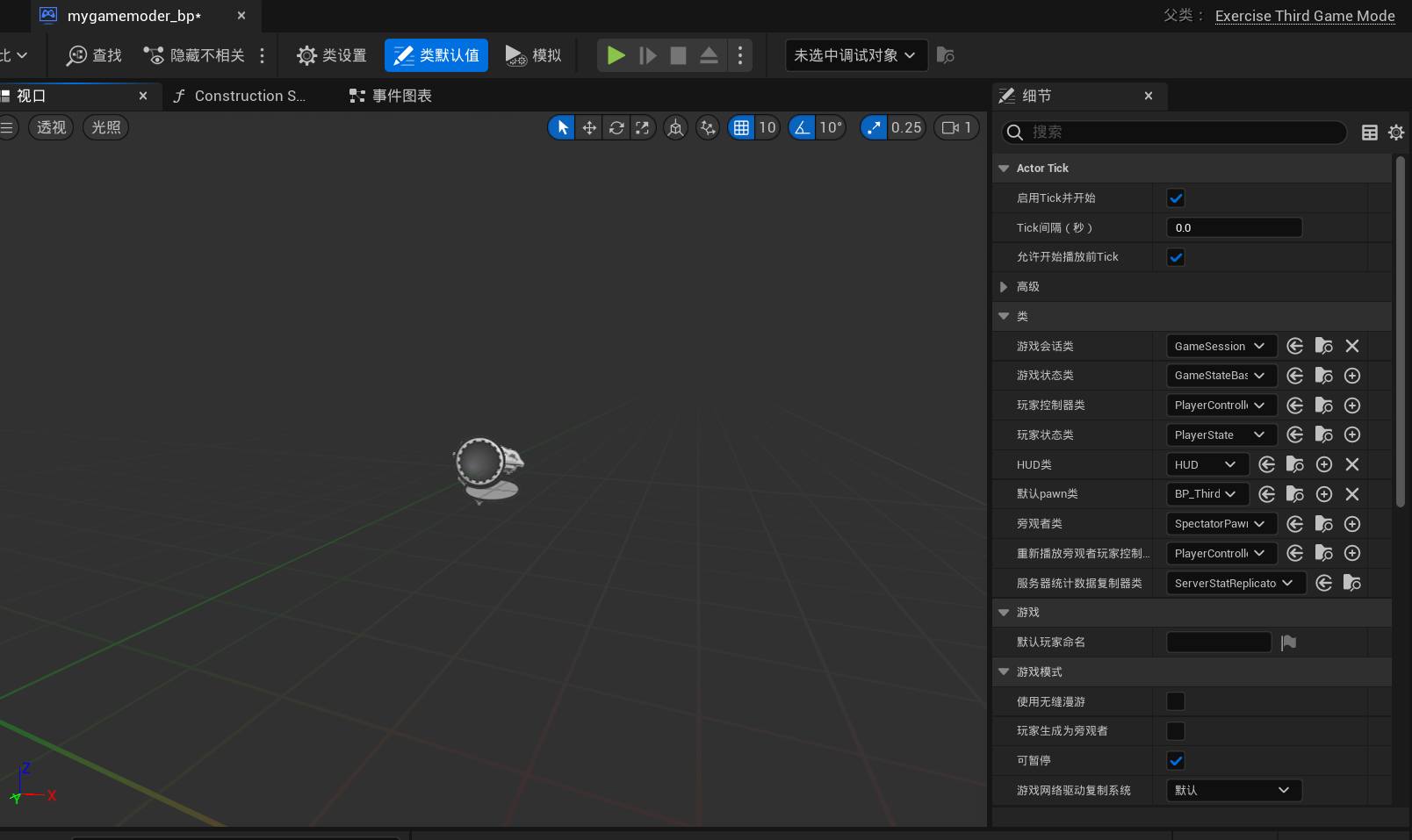
解决办法：（1）创建一个自己的控制角色的文件夹（2）并创建对应的BP文件，需要将该BP类继承我们的cpp中所设置的这个PlayerPawnBPClass）（3）然后再将gamemode设置为当前我们创立的这个BP

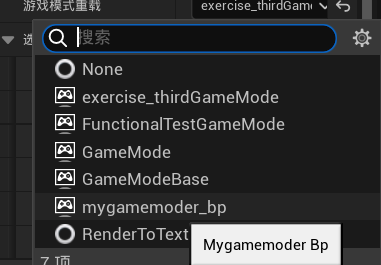


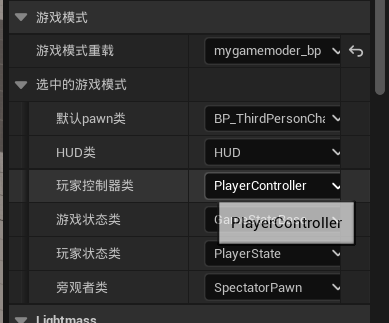
具体图例：





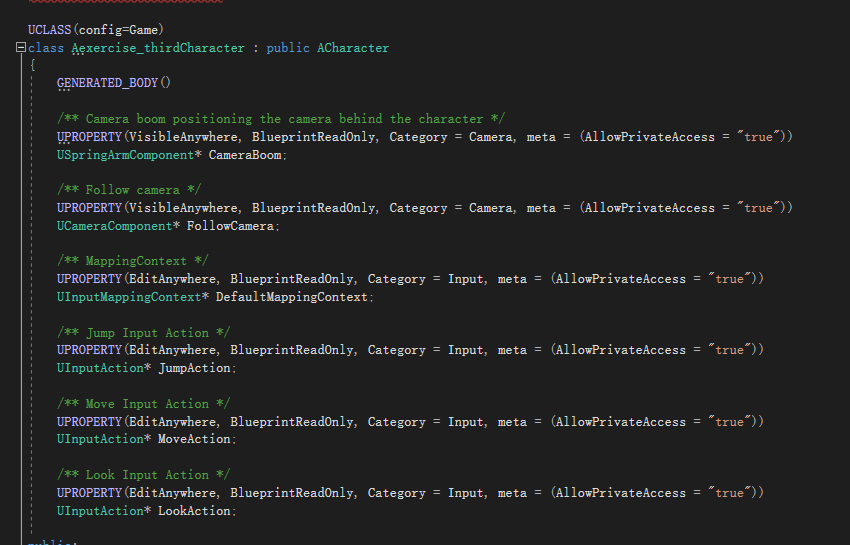






另外注意点：项目默认的这个控制对象继承的cpp中使用了大量原始指针，不适合开发的调试维护。

初始样例：



修改后：

