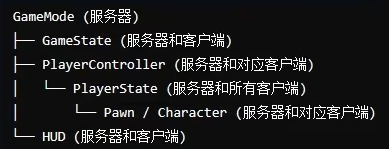
对于非本地多人游戏，

对于一个游戏实例，它有Gamemode、GameState、PlayerController、PlayerState、Pawn等。他们的生命周期和存在位置需要进行关注，以及他们能否复制。



GameInstance可以跨关卡存储数据。

列举出一场游戏的上述各个内容的层次关系。

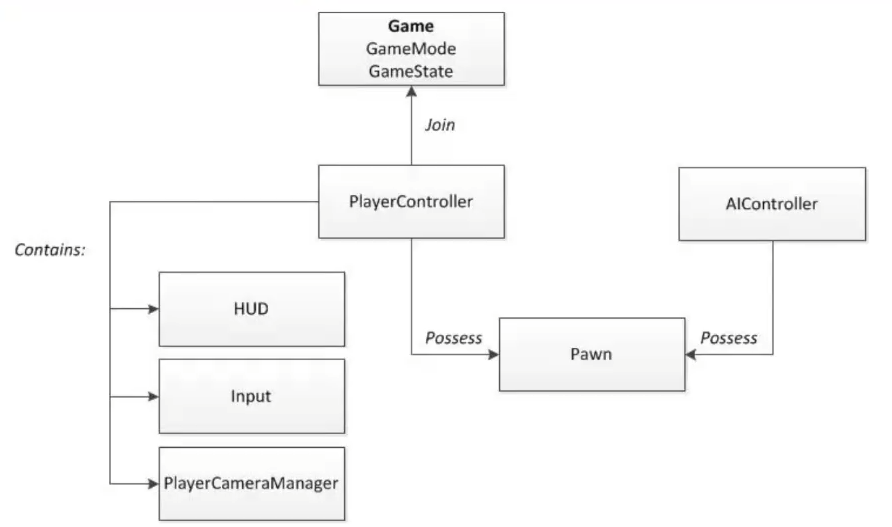


## GamePlay

Gameplay用于设置游戏规则，表示世界场景中的玩家、好友和敌人，使用玩家输入或AI逻辑控制场景中的角色，追踪游戏和玩家的进展情况，并向玩家返回相关信息

可以在世界场景设置中设置自定义的游戏模式，和游戏模式下的各种类。

下图是Gameplay框架的关系图



## Pawn与Character

## PlayerController

## HUD

## GameState与PlayerState

## GameMode

## UObject

## AActor

## Component

## GameInstance

## Level

## LevelBlueprint

## UWorld

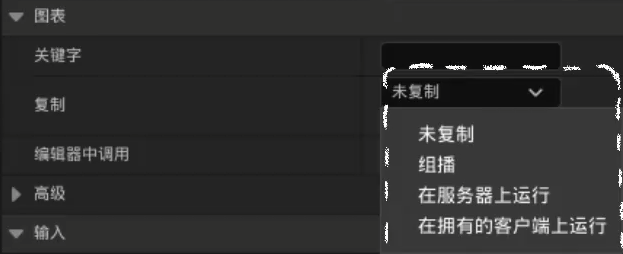
## Subsystem

## 网络框架

## Actor网络同步

## 行为同步与属性同步

行为同步的主要目的是为了解决在调用函数时，将调用与执行位置进行拆分，即调用位置和执行位置不一定是相同的。



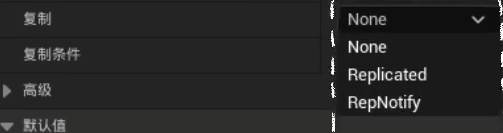
例如在服务器调用一个函数，但执行就是在客户端执行。在客户端调用一个函数，但执行是在服务器中执行。

为了实现所属机制，可以使用虚幻给我们提供的RPC远程调用。

至于属性同步

属性更新只会来自于服务器，客户端永远不会向服务器或其他客户端发送属性更新。这是出于数据安全性考虑。

属性同步有两种方式：Replicated和RepNotify



Replicated，同步数据，但没有通知，无法直接通过参数修改驱动逻辑

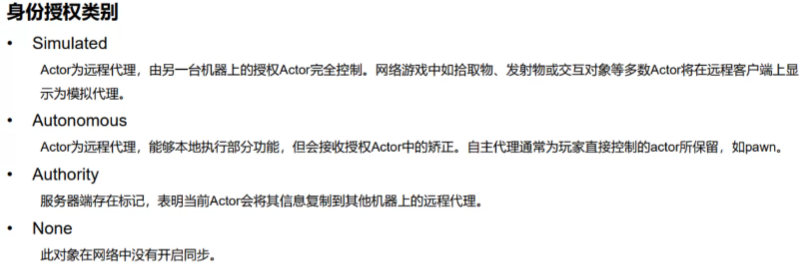
RepNotify，同步数据，并生成通知函数，进行更新通知

两者的区别就是在属性更新时是否执行回调逻辑。在大多数情况下，属性同步是可以使用行为同步代替的，但并不代表属性同步没有作用。

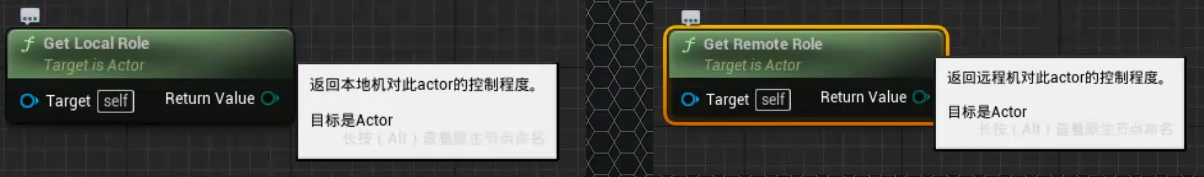
例如在设计发射子弹播放特效时，我们可以同步子弹数量，当发生变化时直接播放特效，免去了单独编写函数的目的。

## Actor的网络身份

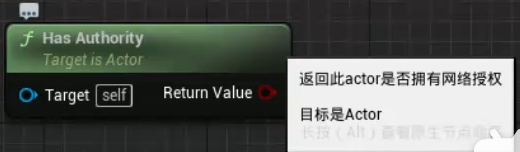
在UE中，当对象开启了网络同步，就会获得网络身份授权



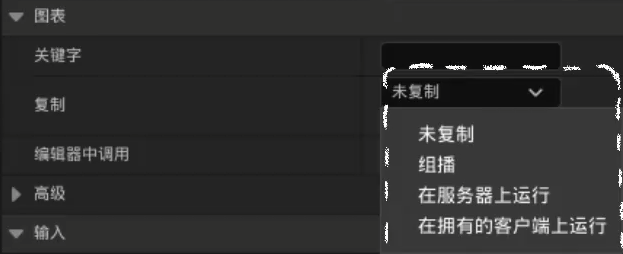
可以通过如下两个节点查看当前Actor在当前终端和远端的身份



同时Actor上也封装了函数HasAuthority，用来判断对象是在服务端还是在客户端。



在执行行为同步时，我们必须知道当前对象执行行为时所在的终端类型，以便我们选择合适的同步方式处理动作



例如在开枪动作产生在客户端时，我们需要让服务器去检查玩家是否具备开枪能力。开枪动作虽然在客户端产生，但是执行结果是在服务器。这时就需要我们知道动作是在哪里产生的。