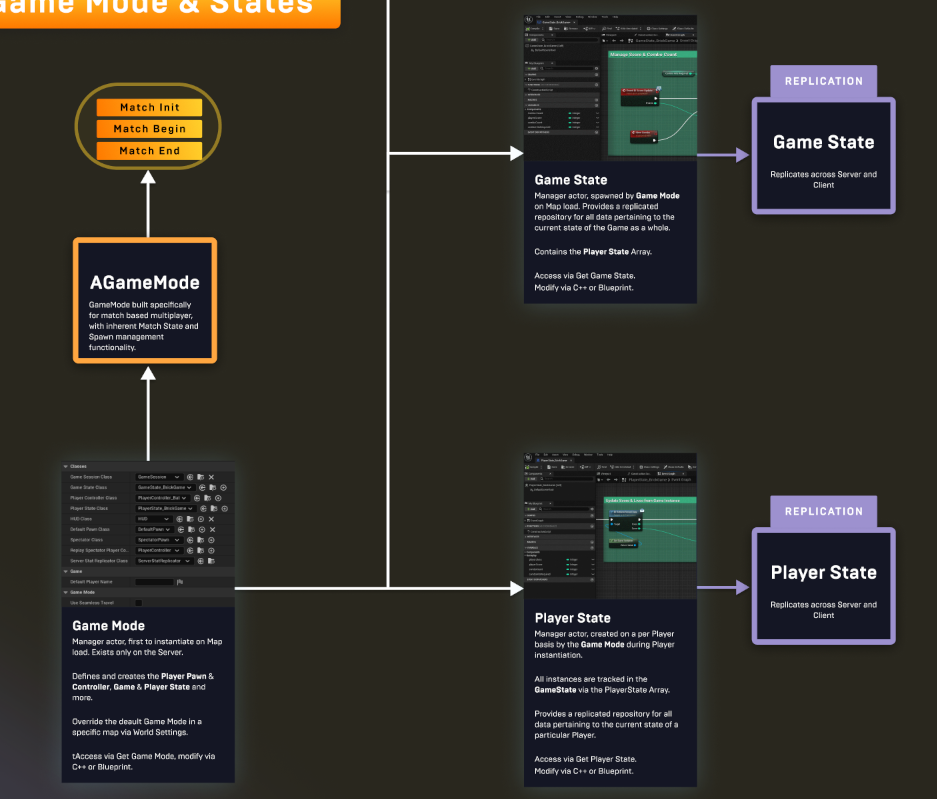


*游戏模式* 会在引擎加载你的关卡并创建世界后立即实例化。游戏模式是一种基于服务器的管理器类，继承自Actor类。由于这种类会在关卡加载时创建，它不会持续运载于所有关卡中。游戏模式是在关卡加载时第一个被实例化的Actor，可以按地图一一设置。游戏模式在Gameplay框架中处于核心地位，管理着一个Gameplay会话的所有规则和结构，并在创建时实例化剩余的框架Actor。前两个将是游戏状态和玩家状态。

*游戏状态* 和 *玩家状态* 是非实体Actor，分别用于追踪游戏和游戏内玩家的状态。在联网多人游戏中，这些类会在权威服务器和所有互联的客户端间复制其状态信息。游戏状态包含与游戏内所有玩家相关的数据和逻辑，例如队伍得分、目标以及所有玩家及其相关玩家状态的列表。而玩家状态处理其关联玩家的相关数据和逻辑，例如生命值、子弹数量和物品栏。游戏状态由游戏模式创建。玩家状态会在每个玩家加入游戏或进入关卡时分别创建。

游戏模式会在玩家加入游戏时生成玩家。一个玩家主要由一个 *控制器* 和一个 *Pawn* 组成。控制器类处理玩家在游戏世界内的操作逻辑。UE中有两种用途广泛的控制器类：*玩家控制器* 和 *AI控制器*。玩家控制器类是一种管理器类，可以处理来自人类的输入，显示抬头信息并处理游戏内的物理表现。AI控制器类也是管理器类，主要处理游戏内的物理表现，并在UE的人工智能帮助下显示其操作，包括：行为树、状态树、寻路等等。

作为一种非实体Actor类，控制器类及其派生的类在游戏世界内没有实体。*Pawn* 类由玩家在游戏世界内的物理实体组成。Pawn类对玩家的重要性不亚于控制器类。控制器处理Pawn并指使其在游戏执行操作。Pawn作为一种派生自Actor的类，由数种Actor组件构成，包括碰撞组件、静态网格体组件和移动组件。角色类是派生自Pawn的子类，由默认Pawn类以及多种功能丰富的组件构成，包括：角色移动组件、骨架网格体组件和胶囊体组件。



**直接通信**

直接Actor（Direct Actor）通信是关卡中各Actor间最常用的通信方法。

这种方法需要你引用目标Actor（target actor），以便当前Actor（working actor）获取目标Actor的信息。采用这类通信时，当前Actor和目标Actor之间是一对一的关系。

**类型转换**

类型转换是一种常用的通信方法，方法是获取Actor的引用，并尝试将其转换为其他类型。如果转换成功，则可以通过直接通信来访问其信息和功能。

此方法要求引用关卡中的Actor，以便使用 **Cast** 节点来将它转换成特定的类。采用此通信方法时，当前Actor和目标Actor之间是一对一关系。

**使用时机**

如果你已经引用了某个Actor，并且希望在访问其信息之前检查该Actor是否属于某个类型，请使用此通信方法。

**接口**

接口会定义一组常用行为或功能，这些行为或功能可以由不同的Actor类实现。此通信方法简化了在不同Actor类上实现同类型函数的过程。

此方法要求每个Actor都实现接口，以便访问它们共有的函数。此外，你还需要拥有Actor的引用，以便通过引用来调用接口函数。采用这种通信方法时，当前Actor和目标Actor之间是一对一关系。

**使用时机**

当你希望为不同类型的Actor创建通用功能时，请使用此方法。

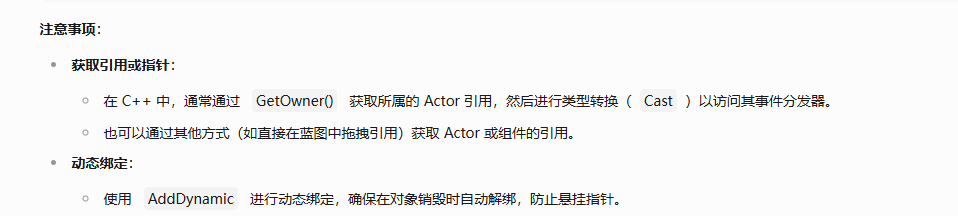
**示例**

* 创建一个交互系统，其中，每个与玩家交互的Actor都会做出不同响应。例如，当玩家与门Actor交互时，门Actor会打开；而当玩家交互光源Actor时，光源Actor会点亮。
* 将破坏应用至关卡中的不同Actor。每个Actor都会对发生的破坏做出不同的响应。例如，墙Actor在受到破坏时将会破裂，而门Actor在受到破坏时将会打开。

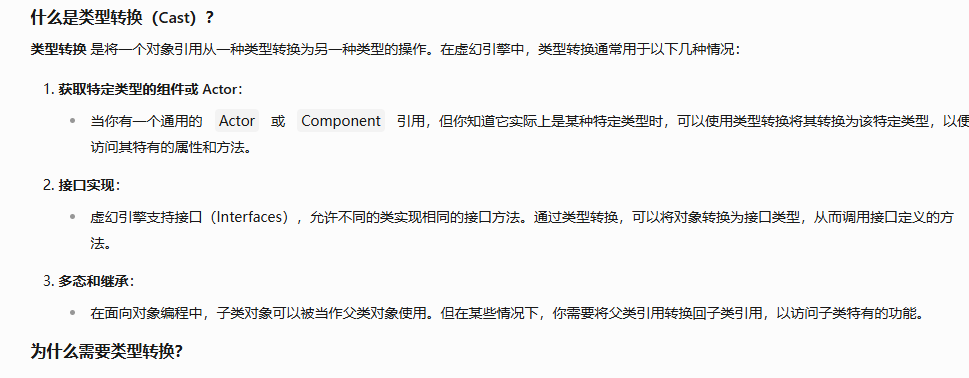


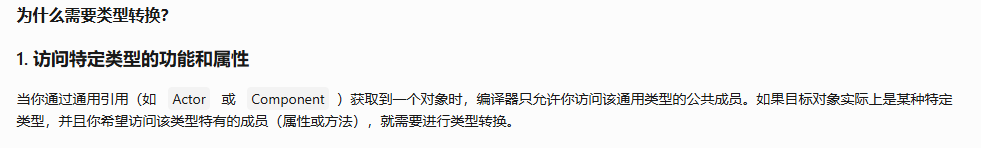


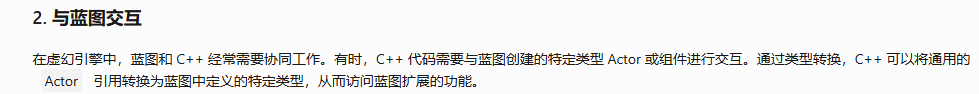


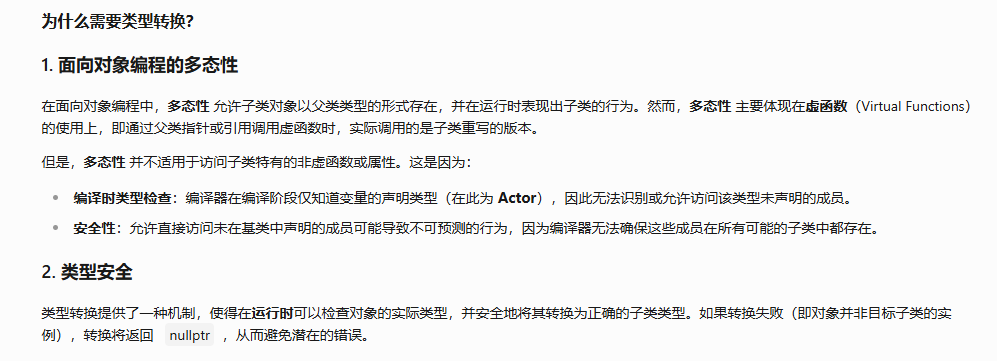


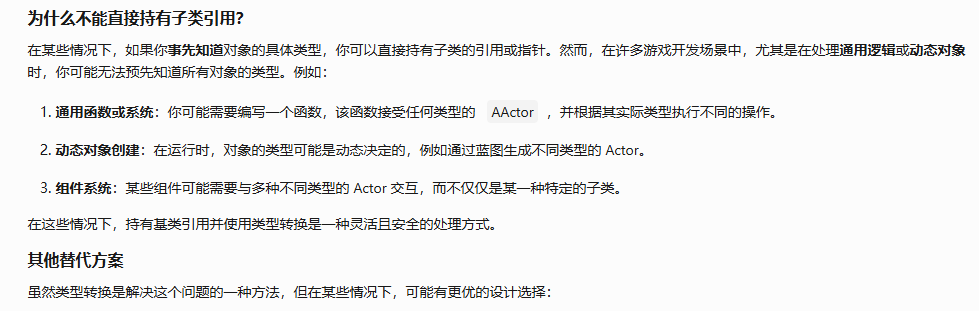


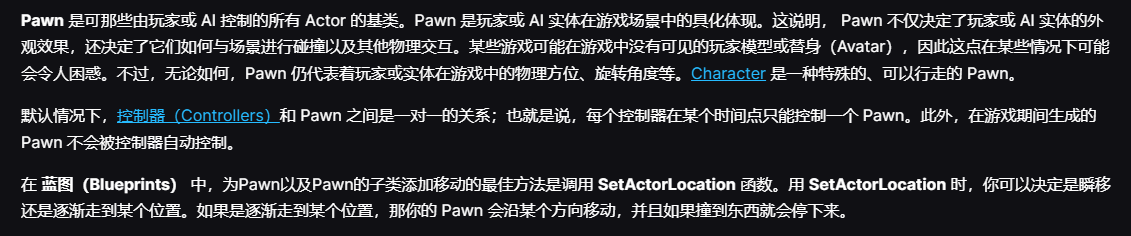












添加 CharacterMovementComponent、CapsuleComponent 和 SkeletalMeshComponent 后，Pawn类可延展为功能完善的 **角色** 类。 角色用于代表垂直站立的玩家，可以在场景中行走、跑动、跳跃、飞行和游泳。 此类也包含基础网络连接和输入模型的实现。

**骨架网格体组件**

与pawn不同的是，角色自带 SkeletalMeshComponent，可启用使用骨架的高级动画。可以将其他骨架网格体添加到角色派生的类，但这才是与角色相关的主骨架网格体。 如需了解骨架网格体的更多内容，请参见：

对于每个帧，ML布料模拟系统会取角色的骨骼姿势作为输入，并使用线性混合蒙皮对基础布料网格体变形，从而预测最终布料姿势。变形有两大组件。

第一个组件是类似于神经变形模型的 **低频率变形** 。

此组件使用多层感知机（MLP）网络建模，该网络会使用骨骼网格体姿势作为输入来预测一组系数。低频率增量的计算方式为：

low\_frequency\_deltas = mean\_delta + coeffs \* basis

复制完整片段

该基础信息可以在训练时习得，也可以通过主组件分析（PCA）预先计算得出。

要训练此组件，你应该准备随机姿势数据集并模拟要在每个姿势上稳定呈现的服装。对于此组件，我们推荐对人形角色采用5000个姿势。

第二个组件是使用 **最接近相邻值（Nearest Neighbor）** 搜索的 **高频率变形**。

此组件会从第一个组件获取系数，并在"最接近相邻值"数据集内搜索最接近相邻值，如：

high\_frequency\_deltas = NearestNeighborSearch(coeffs)

复制完整片段

对于此组件，应该准备一组较小而多样化的姿势，并模拟要在每个姿势上稳定呈现的服装。理想情况下，姿势应该取自游戏内动画的一些关键帧。为了获得最优效果，推荐对人形角色采用50到100个姿势。