图示

AI 生成的内容可能不正确。

藉着UObject提供的元数据、反射生成、GC垃圾回收、序列化、编辑器可见，Class Default Object等，UE可以构建一个Object运行的世界。

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

文本, 电子邮件

AI 生成的内容可能不正确。

图片包含 应用程序

AI 生成的内容可能不正确。

脱胎自Object的Actor也多了一些本事：Replication（网络复制）,Spawn（生生死死），Tick(有了心跳)。  
Actor无疑是UE中最重要的角色之一，组织庞大，最常见的有StaticMeshActor, CameraActor和 PlayerStartActor等。Actor之间还可以互相“嵌套”，拥有相对的“父子”关系。

下定决心，让Actor们轻装上阵，只提供一些通用的基本生存能力，而把众多的“技能”抽象成了一个个“Component”并提供组装的接口，让Actor随用随组装，把自己武装成一个个专业能手。

看见[UActorComponent](https://zhida.zhihu.com/search?content_id=1318397&content_type=Article&match_order=1&q=UActorComponent&zhida_source=entity)的U前缀，是不是想起了什么？没错，UActorComponent也是基础于UObject的一个子类，这意味着其实Component也是有UObject的那些通用功能的。（关于Actor和Component之间Tick的传递后续再细讨论）

TSet<UActorComponent\*> OwnedComponents 保存着这个Actor所拥有的所有Component,一般其中会有一个SceneComponent作为RootComponent。  
TArray<UActorComponent\*> InstanceComponents 保存着实例化的Components。实例化是个什么意思呢，就是你在蓝图里Details定义的Component,当这个Actor被实例化的时候，这些附属的Component也会被实例化。

一个Actor若想可以被放进Level里，就必须实例化USceneComponent\* RootComponent。

图示

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图形用户界面, 文本

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

而从游戏逻辑的实现来说，UE也是不推荐把游戏逻辑写在Component里面，所以你其实也没什么机会去写一个很复杂的Component.

**思考：Actor的SceneComponent哲学**  
很多其他游戏引擎，还有一种设计思路是“万物皆Node”。Node都带变换。比如说你要设计一辆汽车，一种方式是车身作为一个Node,4个轮子各为车身的子Node，然后移动父Node来前进。而在UE里，一种很可能的方式就变成，汽车是一个Actor，车身作为RootComponent，4个轮子都作为RootComponent的子SceneComponent。请读者们细细体会这二者的区别。两种方式都可以实现出优秀的游戏引擎，只是有些理念和侧重点不同。

图形用户界面

AI 生成的内容可能不正确。

文本, 应用程序

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

文本, 应用程序

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

图形用户界面, 文本

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

手机屏幕截图

AI 生成的内容可能不正确。

文本, 信件

AI 生成的内容可能不正确。

。Pawn表达的最关键点是可被玩家操纵的能力。

图形用户界面, 文本

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图形用户界面, 文本, 应用程序

AI 生成的内容可能不正确。

图片包含 表格

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

文本

AI 生成的内容可能不正确。

图片包含 表格

AI 生成的内容可能不正确。

Pawn和controller是一对一的关系。

图形用户界面, 文本, 应用程序, 电子邮件

AI 生成的内容可能不正确。

**为何Controller不能像Actor层级嵌套？**

再来说位置，如果Controller有自己的位置，这样在Respawn重新生成Pawn的时候，你就可以选择在当前位置创建。因此为了自动更新Controller的位置，UE还提供了一个bAttachToPawn的开关选项，默认是关闭的，UE不会自动的更新Controller的位置信息；而如果打开，就会把Controller附加到Pawn的子节点里面去，让Controller跟随Pawn来移动。你可以把这两种模式想象成一种是上帝视角在千里之外心电感应控制Pawn，另一种是骑在Pawn肩上来指挥。

文本

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

一个游戏中只有一个PlayerController，在不同的关卡中你可以使用不同的PlayerController，但是同一时刻响应的只能是一个PlayerController。

因此，为了支持这种情况，我们可以开始允许游戏中同时出现多个PlayerController，每个PlayerController甚至都可以拥有自己的Viewport（分屏或者不同窗口），这样我们通过配置，可以精确的路由手柄1的输入给玩家1，各自的逻辑也很好的区分和复用。

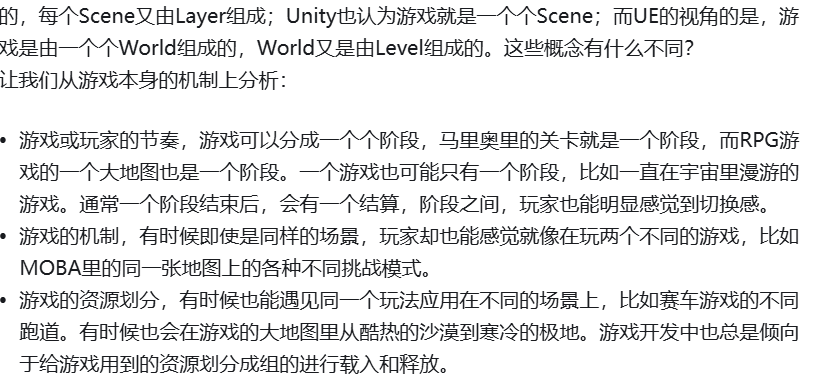
我们就又得扩展一下PlayerController的概念，PlayerController不仅能控制本地的Pawn，而且还能“控制”远程的Pawn（实际上是通过Server上的PlayerController控制Server上的Pawn，然后再复制到远程机器上的Pawn实现的）。

PlayerController作为玩家直接控制的实体，很多的跟玩家直接相关的操作也都得委托它来完成。目前来说PlayerController里旗下的100+的函数也大概可以分为以上几大模块，也根据需要重载了Controller里的一些其他函数。  
UE的思想是具象化一个“玩家实体”，并把所有的跟该玩家相关的操作和接口都交给它完成。一般其他的游戏引擎只是个“功能引擎”，提供了一些图形渲染UI系统等组件，但是在GamePlay这个层次就都非常欠缺了，一般都需要开发者自己搭建一套。而回想你写过的游戏，是不是也往往有一个Player类（一般是单件或者全局变量）？里面几乎是放着所有跟该玩家相关的业务逻辑代码。

图示

AI 生成的内容可能不正确。

那么本篇，我们继续爬升，对于由Actors组成的Level这一层次，UE又是怎么控制的呢？  
对Level记不太清楚的朋友，可以翻回去查看“GamePlay架构（二）Level和World”的讲述，简单概括就是World是由一个PersistentLevel和一些subLevels组成的，PersistentLevel切换了，相应的World也会切换。



玩法就是“逻辑”，场景就是“表示”。所以我们如果以逻辑来划分游戏，得到的就是一个个World的概念；如果以表示来划分，得到就是一个个Level。

笼统的讲，一个World就是一个Game，把玩法叫做Mode，我们应该也能接受吧。

图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

图表

AI 生成的内容可能不正确。

图片包含 图示

AI 生成的内容可能不正确。

图示

AI 生成的内容可能不正确。

UObject

└── UEngine

├── UGameEngine (游戏运行时引擎)

└── UEditorEngine (编辑器环境下的引擎核心)

├── UUnrealEdEngine (编辑器主逻辑管理)

├── UAssetEditorEngine (资产管理编辑器基类)

│ ├── UMaterialEditorEngine (材质编辑器)

│ ├── UBlueprintEditorEngine (蓝图编辑器)

│ └── UAnimationEditorEngine (动画编辑器)

├── ULevelEditorSubsystem (关卡编辑器子系统)

├── UContentBrowserSubsystem (资源浏览器子系统)

├── UEditorUtilitySubsystem (编辑器工具与脚本扩展)

└── UPluginManager (插件管理系统)