虚幻引擎网络系统（一）

虚幻引擎高级程序开发工程师班

PART 01

单机游戏&网络游戏

虚幻引擎高级程序开发工程师班

单机游戏

单机游戏（Single-Player Game），也称单人游戏，是相对于网络游戏而言的。一般指游戏的主要玩法只需要一台电脑就能完成的电子游戏，不能进行互联网对战。

分类：主机游戏，电脑游戏，掌机游戏，街机游戏，手机游戏

狭义上的单机游戏

早期：指完全没有线上游戏功能者，**只能在一台或多台相邻的主机上执行**，如：上古卷轴。

目前：无须互联网即可游玩单人模式(战役、剧情等)，或仅需连上互联网更新、验证身份即可游玩，无须持续与服务器连线者，如绝对武力、战地风云、决胜时刻、末日之战。

广义上的单机游戏

有单人模式，但存取游戏或使用主要功能时，**必须连上游戏服务器执行**，如：暗黑破坏神III、模拟城市5

联网游戏

网络游戏，英文名称为Online Game，又称 “在线游戏”，简称“网游”。指以**互联网为传输媒介**，以游戏**运营商服务器和用户计算机为处理终端**，以游戏客户端软件为信息交互窗口的旨在实现娱乐、休闲、交流和取得虚拟成就的具有可持续性的个体性多人在线游戏。

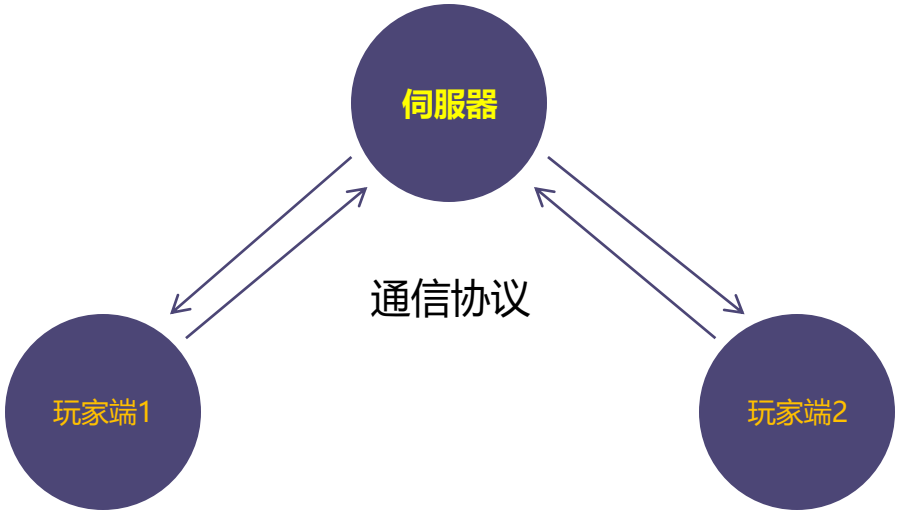
定义：借助广域网，连接更多的玩家进行游戏。

意义：早期受限于硬件制约，广域网的游戏没有兴起。随着硬件的发展，网络环境的变革，促使网络游戏快速兴起！网络游戏使得人类**生活更加丰富**，促使社会的进步。**丰富了人类的精神世界和物质世界，让人类的生活更快乐**。

网络游戏构成必要条件

1. 终端用户
2. 联网硬件环境
3. 伺服器
4. 数据传输通信协议

网络游戏结构



借助**伺服器和通信协议**，可以构建玩家之间的交互行为，丰富游戏中的玩法乐趣

伺服器

是提供计算服务的设备。由于服务器需要响应服务请求，并进行处理，因此一般来说服务器应具备承担服务并且保障服务的能力。

游戏服务器：

更加复杂的计算逻辑，响应要求快，并发连接数更大，设计难度高。并且游戏服务器要求更加广泛的适应性，数据处理及时性要求严格。

客户端

客户端（Client）或称为用户端，是指与服务器相对应，为客户提供本地服务的程序。

游戏客户端一般只用来做表现操作，将玩家的操作进行**收集**，将重要逻辑操作**反馈**给伺服器。伺服器将用户的反馈进行整理，根据设定进行判定，再做下发广播，客户端根据伺服器的反馈进行**操作展示**。游戏客户端设计的繁琐度高，制作难度大。

与单机游戏区别

1. 用户数据交互方式不同，单机游戏仅限小范围内数据交互操作或是无数据交互的单用户操作。网络游戏交互范围更加宽泛
2. 单机游戏不约束用户自主作弊方式提高游戏乐趣，网游则需要保证所有用户的一致公平性
3. 单机游戏无运营商，伺服器（广域网），网络游戏则有运营商，伺服器
4. 单机游戏个性更加突出，玩法更加精良，网络游戏则需要突出大众化的特点，适应更多用户
5. 单机游戏的硬件平台更加宽泛，网络游戏则由于通信限制对于设备硬件有要求

PART 02

虚幻网络框架

虚幻引擎高级程序开发工程师班

虚幻网络特点

虚幻引擎的网络系统与其他网络游戏系统设计是有本质区别的。首先虚幻引擎的网络系统并**没有提供数据库存储解决方案**（如果需要状态存储需要使用者自行设计）。所以，相比较状态类型联网游戏（例如魔兽世界，暗黑破坏神）而言，虚幻引擎更适合设计为以“**局**”为单位的游戏（例如绝地求生，英雄联盟），虚幻引擎中你可以将服务器理解为“主机”（客户端-服务器模型）。

虚幻引擎提供完整的**产品层（非引擎层）交互解决方案**，通过简单的设计编码即可完成数据同步。并且虚幻清晰的划分了同步的种类，以语言本身的特点而言，提供**“数据同步”和“行为同步”**。需要我们注意的是，**虚幻引擎中不需要我们处理网络链接逻辑，制定通信协议，寻找数据压缩策略，协议序列化策略，包括硬件通信协议选择等**。

服务器

虚幻引擎构建服务器的方式分为两种：**监听服务器和专用服务器**

启动服务器指令

类型	启动命令
监听服务器	UE4Editor.exe ProjectName MapName?Listen -game
专用服务器	UE4Editor.exe ProjectName MapName -server -game -log
客户端	UE4Editor.exe ProjectName ServerIP -game

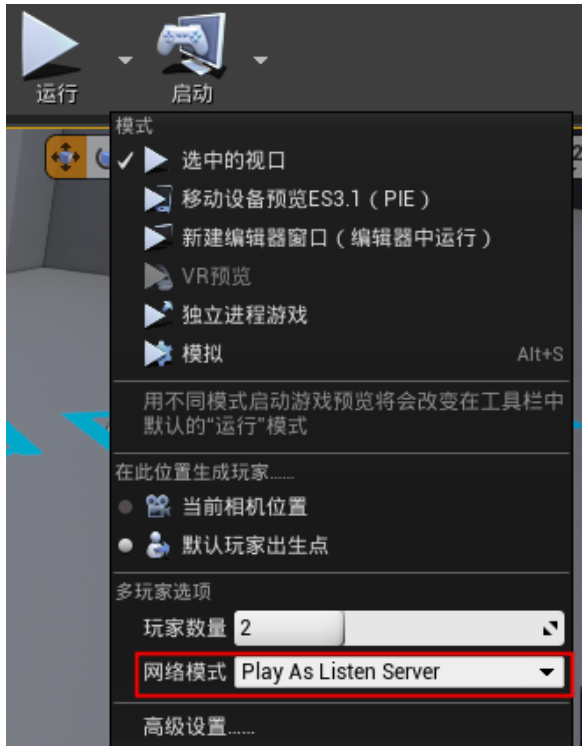
注意：启动指令需要在控制台使用，并且将虚幻编辑器路径设置在用户环境变量中。专用服务器在默认情况下并不会显示窗口。如果不使用 -log，您将不会看到任何呈现专用服务器的窗口。

服务器是虚幻引擎多人游戏的一个重要部分。服务器的作用包括：做出所有重要决定，包含所有的主控状态，处理客户端连接，转移到新的地图以及处理比赛开始/结束时的总体游戏流程等。

监听服务器

监听服务器模型（**Listen-Server**）中会有一个**客户端担当游戏状态的主控者（服务器）**，而连接的客户端将保持近似复本。

监听服务器中，一名玩家终端会被做为服务器引用，在虚幻编辑器中开启监听服务器的方法是运行时，通过启动设置，选择网络模式为“Play As Listen Server”



专用服务器模型

虚幻引擎允许用户设计专用服务器，与设计监听服务器模型一样，专用服务器即去掉客户端特性。在编辑器中，可以通过启动设置，网络模式选择Play As Client完成。

专属服务器本质是只起到服务器逻辑作用，去掉了渲染功能

《绝地求生》游戏使用的就是专属服务器。

专属服务器的好处，可以隔绝用户触及游戏数据，保证游戏的公平，并且提升游戏的稳定性（客户机运行环境复杂如果出现崩溃将导致所有游戏终止）

通过C++源码可以编译独立服务器。

服务器是权威的

由于虚幻引擎的设计特点，使得我们在设计网络游戏时无需独立设计游戏服务器，只从产品设计合理性出发，即可完成网络游戏的设计。

前端设计人员遵循“**服务器是权威**”游戏设计规则，可以将动作校验逻辑放在“主机端”执行，保证“主机端”动作是合理的，客户端即可真正执行动作。

举例：

当您移动您的角色，作为客户端，在多人游戏中，您实际上并没有移动您的角色自己，但告诉服务器你想要移动它。

然后，服务器会为所有其他人（包括您）更新角色的位置。注意：为了防止本地客户端的“滞后”感觉，另外还要让这个玩家直接在本地控制他们的角色，尽管服务器仍然可以覆盖角色的位置当客户端开始作弊！这意味着，客户将（几乎）不会直接与其他客户端有交互。

另一个例子：

当向另一个客户端发送聊天消息时，您实际上将其发送到服务器，然后服务器

将其传递给您想要访问的客户端。这也可能是一个队伍，公会，组等。

总结

- 虚幻引擎网络系统无需设计独立的服务器，也就是无需编写独立的服务器
- 虚幻引擎网络系统更适合设计无状态联网游戏，例如绝地求生，英雄联盟等，所以使用虚幻引擎网络模块设计网络游戏需要考虑游戏特点（你也可以通过自己的修改，设计带有数据库的服务器）
- 虚幻引擎设计应保证服务器是权威的，动作校验需要在服务器完成
- 虚幻网络系统具有高效开发特性，前端开发人员可以独立完成联网游戏设计
- 虚幻网络系统中的核心设计点是同步数据和同步行为
- 使用虚幻引擎网络系统无需管理链接通信问题

做个测试

虚幻引擎框架中的角色，我们需要先搞清楚在网络系统中，他们都在哪里（主机端还是客户机端）？

- GameMode
- GameState
- PlayerController
- PlayerState
- Pawn
- HUD

通过开启联网模式，我们可以测试到结果，需要考虑对象在网络中存在几份。

测试结果

虚幻引擎框架中的角色，我们需要先搞清楚在网络系统中，他们都在哪里（主机端还是客户机端）？

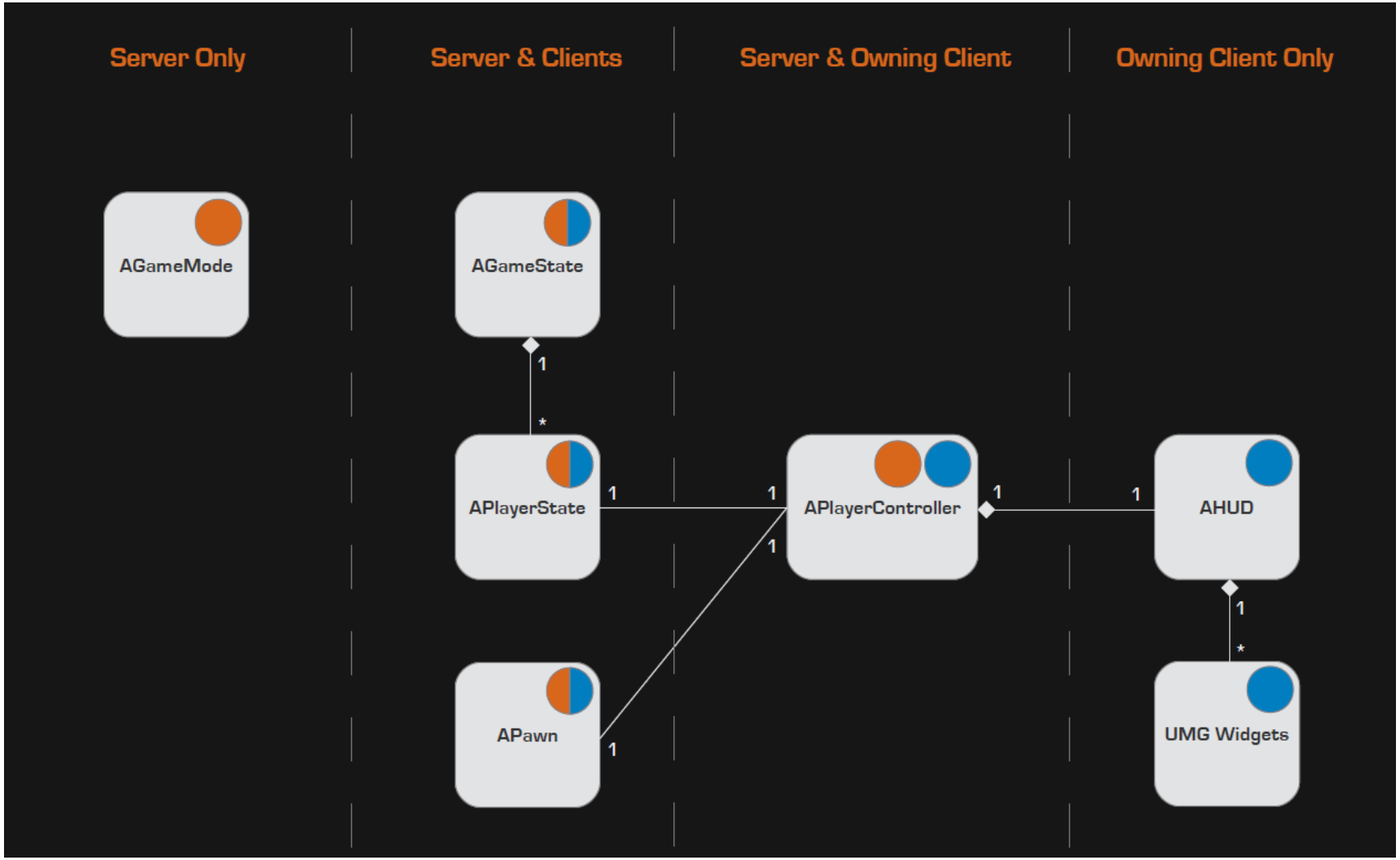
- GameMode（服务器有，客户端无）
- GameState（服务器和客户端）
- PlayerController（服务器有所有人，客户端有自己）
- PlayerState（服务器客户端所有人）
- Pawn（服务器客户端有所有）
- HUD（自己终端）

所在位置分类

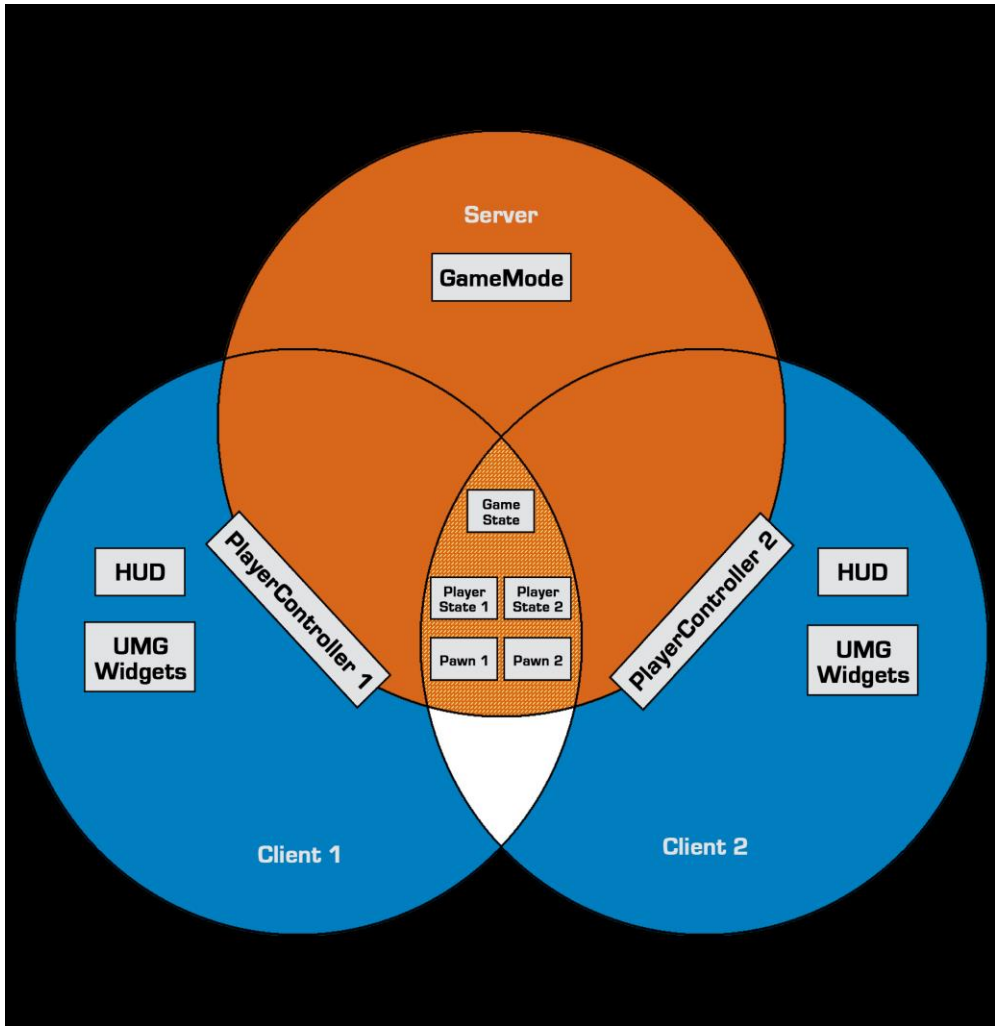
通过测试，我们大致将框架中角色所在位置分为以下计中

- Server Only - 这些对象仅存在于服务器上
- Server & Clients - 这些对象存在于服务器和所有客户端上
- Server & Owning Client - 这些对象仅存在于服务器和自身客户端上
- Owning Client Only - 这些对象仅存在于自己客户端

图表分析


Server Only
Server & Clients
Server & Owning Client
Owning Client Only
AGameMode
AGameState
APlayerState
APlayerController
AHUD
APawn
UMG Widgets
1
*
1
1
1
1
1
1
*

交叉图分析


Server
GameMode
Game State
Player State 1
Player State 2
Pawn 1
Pawn 2
PlayerController 1
PlayerController 2
HUD
UMG Widgets
HUD
UMG Widgets
Client 1
Client 2

PART 03

网络对象同步

虚幻引擎高级程序开发工程师班

思考

- 什么是同步?
- 网络游戏中，什么样的内容需要同步？是不是所有的内容都需要同步?
- 虚幻中的同步单位是什么?

针对这些问题，我们需要进行一次讨论!

思考

是否场景中摆放的物体都，启动时都会同步到所有终端？

场景中摆放的Actor，默认是会同步到所有终端，但是他们没有做网络关联！

Net Load On Client选项默认是勾选的，也就是场景中的物体默认都会在所有终端显示，但是**并不代表所有物体对象都需要进行网络同步（因为所有内容全部同步的成本是非常高的）**。

思考

动态生成Actor是否会同步到所有终端？（需要考虑在哪里生成）

Replicates

当场景中摆放的Actor需要进行网络同步时，需要在细节面板中勾选“Replicates”，需要注意，此选项默认是关闭的。需要设计者根据需求开启此选项。

生成和销毁

- 在服务器生成Actor时，如果生成类开启了“Replicates”则会同步到所有终端
- 当在服务器销毁Actor时，如果Actor开启了“Replicates”则所有终端中的关联对象均会被销毁

思考

当对象开启了Replicate是否就代表对象已经在网络中完成同步了？例如移动对象的位置是否会同步到所有终端（此处我们需要思考移动应该产生在哪里？）

位置同步

当对象开启了“Replicates”只表明了对象在网络中具有同步的能力，但是具体要同步的内容是什么，还需要设计者去思考。同步规则引擎是无法制定的（不同的游戏产品在设计的时候有不相同的需求），设计者根据设计场景的变化，提供对应的同步逻辑可。虚幻对于同步操作，提供了两种方式，**行为同步和属性同步**。

PART 04

行为同步（RPC）

虚幻引擎高级程序开发工程师班

行为同步

“行为同步”指对象调用函数时，可以在不同终端完成相同的（行为）执行结果。例如在服务器调用一个函数，但是执行却是在客户端执行，在客户端调用一个函数，但是执行却在服务器。

调用和执行拆分的前提是调用**成员函数**，成员函数存在于类内，也就是说“行为同步”我们同步的是成员函数，依据的是对象。在虚幻网络中，对象和对象在不同终端之间的关联是需要开启“Replicates”，开启了复制，不同终端中的“相同”对象才会有关系。

总结

- 行为同步，主要描述的是**成员函数**
- 如果对象需要具备行为同步，需要开启网络复制
- 是将成员函数的调用位置和执行位置进行拆分，拆分针对的是不同终端，例如在服务器端的一个对象身上调用一个函数，在客户端上**与之关联的对象**身上执行了结果，这就是同步行为

网络游戏设计过程中，我们需要思考的问题是，如果将动作完成同步，例如丢置一颗手雷，如果让所有终端都可以看到手雷的丢置，这其实就是丢置动作的同步。行为同步是设计网络游戏中的重要实现手段

在了解行为同步之前，我们需要先了解下如何判定执行逻辑在客户端还是服务端

4-1.Actor网络身份（区分客户端OR服务端）

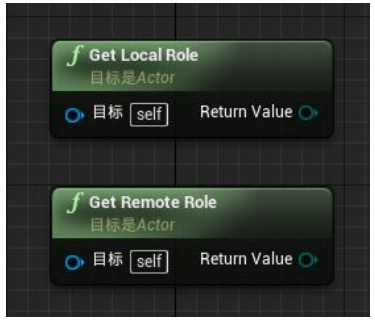
角色网络身份

当**对象开启了网络同步，那么将会获得网络身份授权。终端只分两种，客户端or服务端**，网络身份分为**本地身份和远端身份**（本地即当前终端远端即对立端，例如在客户端远端即服务端，反之亦然）。所有的Actor只要开启了网络复制，均可以获得身份。下图是蓝图中身份的获取方式。

LocalRole，用来裁定当前Actor在当前终端（服务器，客户端）中的身份。

RemoteRole，用来裁定当前Actor在远端（客户端，服务器端）中的身份

在C++中是以成员变量方式存在



身份的用途

网络中的身份可以帮助我们更好的甄别**当前对象**和**当前所在终端**的关系，并且可以帮助我们判断当前对象所在的终端类型（客户端or服务端）。

身份授权类别

- Simulated

  Actor为远程代理，由另一台机器上的授权Actor完全控制。网络游戏中如拾取物、发射物或交互对象等多数Actor将在远程客户端上显示为模拟代理。

- Autonomous

  Actor为远程代理，能够本地执行部分功能，但会接收授权Actor中的矫正。自主代理通常为玩家直接控制的actor所保留，如pawn。

- Authority

  服务器端存在标记，表明当前Actor会将其信息复制到其他机器上的远程代理。

- None

  此对象在网络中没有开启同步。

Local Role身份说明

	服务器	客户端
Simulate	不存在	服务器在当前引擎实例中模拟的玩家角色
Autonomous	不存在	当前引擎实例中由真人操控的角色
Authority	服务器拥有复制权限的权威性	不存在

通过图表可以清晰获得Local Role在不同终端的标记情况。**通过Local Role的终端标记，我们可以很清晰的判定当前对象处于哪个终端中**。如果获取当前对象的LocalRole，结果是Authority，则表明当前对象处于服务端。这是我们最常用的判定对象所在位置的关键依据。

注意：在Actor上，封装了函数HasAuthority，用来判定对象是在服务端还是客户端。

Remote Role身份说明

	服务器	客户端
Simulate	当前Actor在远端是模拟的身份	不存在
Autonomous	当前Actor自主权操控权在远端	不存在
Authority	不存在	拥有当前Actor复制权限的权威性

用来描述当前对象在远端（客户端or服务端）的身份。可以用来**在服务器检查角色是否由远端主控**，当对象的Remote Role类型是Autonomous，则表明当前对象的操控权在客户端。或检查**当前对象是否在客户端**，只要远端身份为Authority则说明对象在客户端。

划分终端所在用途?

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，当开枪动作产生在客户端时，我们需要到服务器去检查玩家是否具备开枪能力。开枪的动作在客户端产生，但是执行结果是在服务端，这时就需要我们知道动作是产生在哪里的。

虚幻引擎框架中，有着主机端的概念，**即玩家操控端和主机端重叠**（参考CS，魔兽争霸等游戏房主概念，房主本身也是游戏的参与者）。主机端也会产生玩家逻辑，在执行动作时，我们必须要清楚的拆分出动作所在终端。

4-2.RPC

RPC（全称Remote Procedure Call），远端调用，指在本机上调用函数，但在其他机器上远程执行的函数。RPC函数可以**允许客户端或服务器通过网络连接相互发送消息**。RPC既是我们说的行为同步，主要作用是将调用和执行进行了拆分。

RPC执行分三种形式

- Server　在**持有当前Actor客户端**调用，在服务器端执行（**Server标记产生在客户端是合理的)**
- Client　在服务器端调用，在**持有对象的客户端**执行（**Client标记在服务端调用是合理的）**
- Multicast　在服务器端调用，在所有终端执行（ **Multicast标记在服务端调用是合理的)**

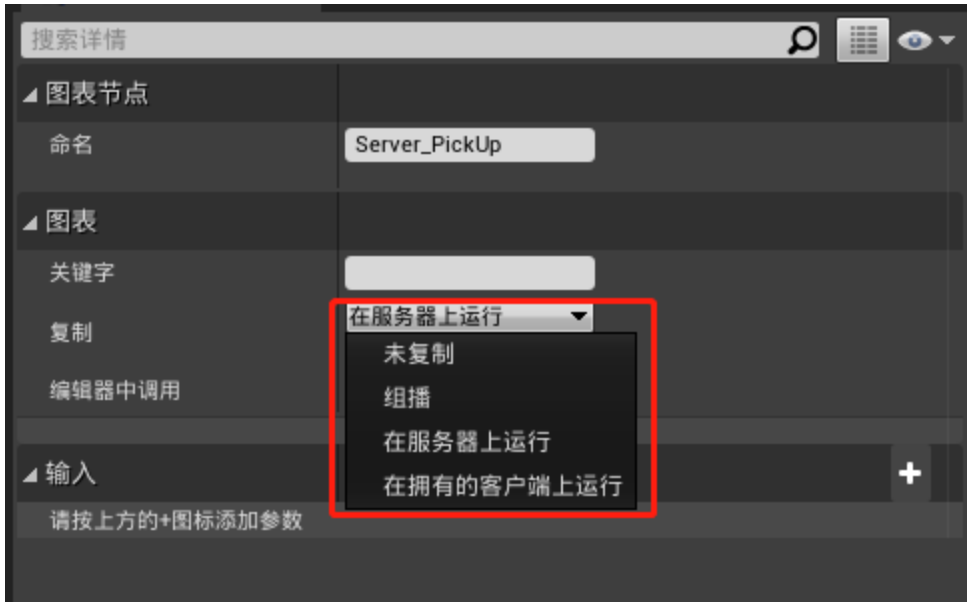
蓝图RPC

在蓝图中RPC的实现是通过事件完成，当Actor对象开启网络同步，构建的事件则可以选择复制方式，参照截图。

组播（Multicast）

在服务器上运行（Server）

在拥有的客户端上运行（Client）



4-3.在客户端是否所有对象都可以执行RPC

拥有权

执行RPC的前提是此Actor需要在网络上被复制（勾选Replicates），当在服务端执行逻辑时，由于服务端是权威的所以RPC的三种通信方式在服务器端执行都可可以的。但是在客户端执行时需要注意，只有当此Actor属于当前终端时（泛指客户端），才可以在Actor身上执行Server调用，并在服务器端执行结果。

在客户端中，**当前终端的PlayerController，Pawn，PlayerState**的所有权均属于当前终端。所以在以上三个对象中使用RPC的Server调用，是存在有效执行结果的

设置拥有权

Actor的所有权是可以修改的，**修改动作必须发生在服务器**，调用Actor的成员函数SetOwner即可。需要提供将所有权分配给哪一个**终端的PlayerController**。分配完成后，此Actor对象在指定终端中的关联对象即可使用RPC的Server标记，调用函数，并且可以完成有效执行结果。

对于操控的Pawn可以通过使用C++中的 IsLocallyControlled 函数，或蓝图中的 Is Locally Controlled 节点，以决定Pawn是否在其拥有客户端上（原理获取操控Controller检查是否于本机PlayerController相同）。

例如：玩家拾起地上的枪械，在没有拾取时，枪械没有归属关系，当拾取后，枪械所有权要修改给拾取玩家所在终端的PlayerController。这样做主要不是为了在枪械对象上向服务器发送消息，而是为了更好的处理带宽优化，整理对象相关性。

RPC调用情况分析

从服务器上调用RPC

Actor所有权	未复制	NetMulticast	Server	Client
Client-Owned Actor	在服务器上运行	在服务器和所有客户端上运行	在服务器上运行	在 actor 的所属客户端上运行
Server-owned actor	在服务器上运行	在服务器和所有客户端上运行	在服务器上运行	在服务器上运行
Unowned actor	在服务器上运行	在服务器和所有客户端上运行	在服务器上运行	在服务器上运行

从客户端上调用RPC

Actor所有权	未复制	NetMulticast	Server	Client
Owned by invoking client（当前终端所有权）	在执行调用的客户端上运行	在执行调用的客户端上运行	在服务器上运行	在执行调用的客户端上运行
Owned by a different client（其他终端所有权）	在执行调用的客户端上运行	在执行调用的客户端上运行	丢弃	在执行调用的客户端上运行
Server-owned actor	在执行调用的客户端上运行	在执行调用的客户端上运行	丢弃	在执行调用的客户端上运行
Unowned actor	在执行调用的客户端上运行	在执行调用的客户端上运行	丢弃	在执行调用的客户端上运行

总结

- RPC需要从Actor上完成
- RPC操作在蓝图中是通过“事件”来完成的
- 只有当Actor对象在网络中开启了Replicates，才可以完成RPC动作
- RPC操作中，一般来讲Client，Multicast需要在服务器调用。Server需要在客户端使用，并且使用的对象归调用客户端所有
- 用来区分当前对象所在终端的方法是通过网络身份来判定，网络身份只有当对象开启了网络同步才可以获得
- 在客户端，只有执行RPC的对象归属于当前终端，才可以完成向服务器发消息
- 在客户端，默认只有PC，PS，Pawn可以向服务器发消息
- 修改对象所有权需要在服务器完成，并且将所有者设置到指定终端的PlayerControlIer上

PART 05

属性同步

虚幻引擎高级程序开发工程师班

属性同步

每个Actor维护一个全属性列表，其中包含Replicated 说明符 。每当复制的属性值发生变化时，服务器会向所有客户端发送更新。客户端会将其应用到Actor的本地版本上。这些更新只会来自服务器，客户端永远不会向服务器或其他客户端发送属性更新。

Actor属性复制可靠。这意味着，Actor的客户端版本的属性最终将反映服务器上的值，但客户端不必接受服务器上某个属性的每一个单独变更。例如，如果一个整数属性的值快速从100变成200，然后又变成了300，客户端将最终接受一个值为300的变更，但客户端不一定会知道这个值曾经变成过200。

注意：我们不推荐在客户端上更改复制的变量值。该值将始终与服务器端的值不一致，直到服务器下一次侦测到变更并发送更新为止。如果服务器版本的属性不是经常更新，那客户端就需要等待很长时间才能被纠正。

属性同步

首先Actor必须满足在**网络复制（勾选Replicates）**，设置的参数需要需要开启复制，属性的修正**必须在服务器端**修改，才可以在网络上同步。

蓝图参数同步有两种方式**Replicated**，**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（向所有终端通知，满足相关性）

注意：在大多数情况下，属性同步可以使用行为同步替代，但是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。

复制对象引用

一些情景中，我们需要将UObject对象进行同步，一般而言，对象引用会在 UE4 多人游戏架构中自动处理。这就是说，如果您有一个已经**复制的 UObject** 属性，则对该对象的引用将作为服务器分配的专门 ID 通过网络进行发送。这个专门id 是一个 FNetworkGUID。服务器将负责分配此 id，然后向所有已连接的客户端告知这一分配。通过ID，我们可以在不同终端找到关联的对象。

如果对象没有被复制（勾选复制选项），但是设置给属性进行同步，需要遵循此对象具有可靠命名，即对象不能是动态生成，而是直接通过资源包摆放在场景中的。

通常可以按照以下原则来确定是否可以通过网络引用一个对象：

- 任何复制的 actor 都可以复制为一个引用
- 任何未复制的 actor 都必须有可靠命名（直接从数据包加载）
- 任何复制的组件都可以复制为一个引用
- 任何未复制的组件都必须有可靠命名。
- 其他所有 UObject（非 actor 或组件）必须由加载的数据包直接提供

拥有可靠命名对象

拥有可靠命名的对象指的是存在于服务器和客户端上的同名对象（**例如直接在编辑模式下摆放在地图中的对象**）。

如果 Actor 是从数据包直接加载（并非在游戏期间生成），它们就被认为是拥有可靠命名。

其实属性同步中，当同步的变量的数据类型是Actor类型时，我们需要考虑在不同终端间如何构建对象的关联关系。如果对象是开启网络同步的，那么由引擎负责构建关系，通过NetID关联不同终端的相同对象。当对象不是勾选的网络同步，则引擎不会负责构建关系，那么如何保证在每个终端上找到的对象都相同呢？那么可靠命名就是最好的方案了！在A终端上存在一个名字是N的对象Actor，在B终端上也能找到N的对象Actor，则通过名字我们就构建了对象关系。但是对象不具备网络同步的能力。（在世界大纲中，不可能存在名字相同的对象）。

为什么动态生成的不可以？首先终端生成对象时，会根据终端当前命名序列给予命名，当不同的终端随着运行时间增加，差异也会越来越大，所以不能使用动态生成。

总结

- 对象Actor必须开启网络复制，其属性才能够完成同步
- 同步中，如果属性变化需要得到通知，尽量不要使用轮询方式进行更新，应使用通知方式
- 同步属性应该在服务器完成（给属性赋值动作），禁止在客户端操作（操作有效，但是不会同步）
- 通常同步数据的类型需要使用UE的Gameplay为基础的变量
- 当属性类型为Actor，注意Actor复制引用的相关操作

感谢观看