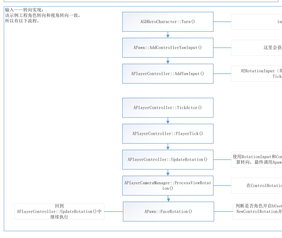
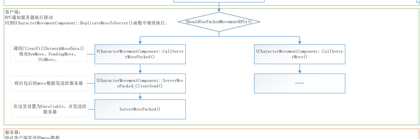
移动网络同步流程（普通移动）：
位置移动网络同步流程，使用的通信方式和流程，
使用的通信方式等级从高至低。

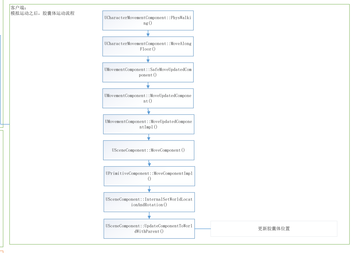


客户端：
执行本地移动

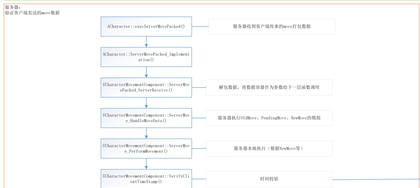
客户端：
时间
UCharacterMovementComponent::TickComponent()中继续执行。



客户端：
模拟运动之后，检查移动动画



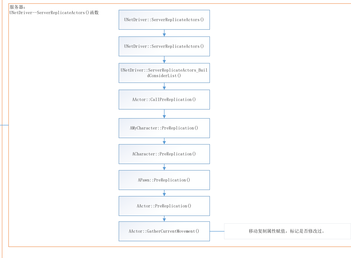
服务器：
处理客户端发送的Move数据



函数具体流程：
1、调用IsClientTimeStampValid()判断客户端TimeStamp是否有效，以防Cheat(ClientTimeStamp和LastReceivedClientTimeStamp进一步比较，如果客户端响应参数的Cmovemap之差超过一定时间，将设置PlayerMoveDetected为true。
2、如果上面的时间检测和位置信息UnrecognizedReceivedPylsLoc，将调用ProcessClientTimeStampOrTimeDiscrepancy()进行进一步的检测。

服务器：
UNetDriver

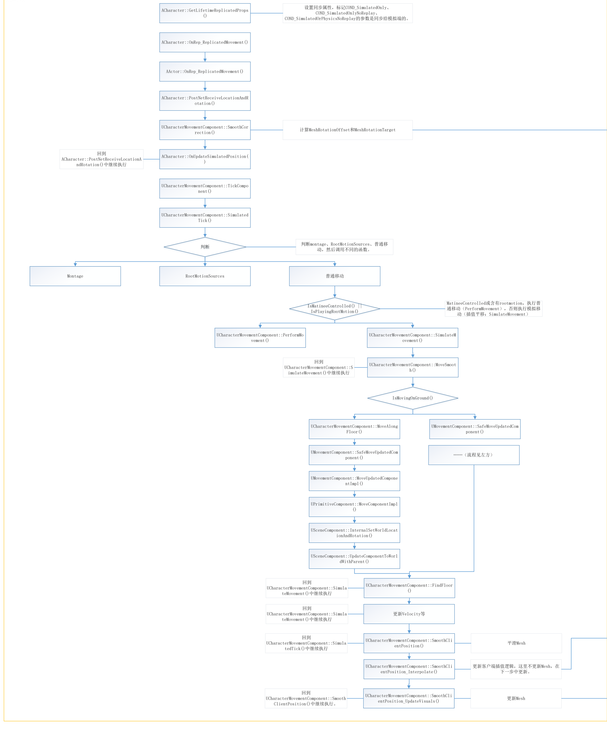
服务器：
UNetDriver::ServerReplicateActors()函数



客户端：
收到服务器move ack后的流程

函数具体流程：
1、调用AckMove()删除SavedMove数据中该Timestamp及之前的所有move。
2、更新Velocity，location，ServerMoveAckMove为服务器的数据。
3、设置UpdatedPositionNeeds，标记已经修正。
4、调用ClientUpdatePositionAfterServerUpdate()判断该值，如果该值为true，则调用UCharacterMovementComponent::ClientUpdatePositionAfterServerUpdate()进行重新演算，即最后SavedMove数据，调用ClientMoveAcknowledged()，但并不向服务器发送pos，这样move之前已经发送给服务器了，所以服务器会继续修正，但是由于本地重新演算，所以再次会被重修正。

模拟端流程：



函数具体流程：
1、当ClientSmoothingMode为Linear时，
（1）每次更新时间deltaseconds加上SmoothingClientTimeStamp。
（2）计算出SmoothingClientTimeStamp/SmoothingClientTimeLimit，（倒数，从0倒到1）。
（3）使用该比例计算出Offset，当Offset接近或超过PreOffset值（即1.0），时间到期，将之前LastPreOffset值赋为上一比例因子，将之前LastPreOffset设置为0。
（4）使用该比例计算出Offset，当LastPreOffset值不为0，则用TranslationOffset设置为0，因之调用Lerpable()计算出新的Offset。上面的两种判断都是以用TranslationOffset为基数，但RotationOffset使用RotationOrigin为基数，当LastPreOffset不为0之上面的情况时，直接使用前一次的RotationOrigin，而不是当前的RotationOrigin。
2、当ClientSmoothingMode为Exponential时，使用指数方式计算新的RootTranslationOffset，使用LerpFrom计算新的RootTranslationOffset。
3、当ClientSmoothingMode为Replay时，在空闲了解。

函数具体流程：
1、当ClientSmoothingMode为Linear时，计算出OldClientLocOffset，之后调用ClearRotationOffset()清RotationOffset。
2、当ClientSmoothingMode为Exponential时，根据上一步计算出的OldClientLocOffset和OldClientLocTarget，计算出新的OldClientLocOffset和OldClientLocTarget。
3、当ClientSmoothingMode为Replay时，调用ClearRotationOffset()清RotationOffset。