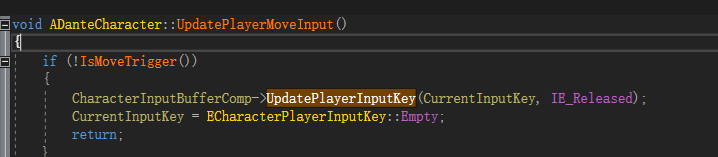
首先看下移动的问题：

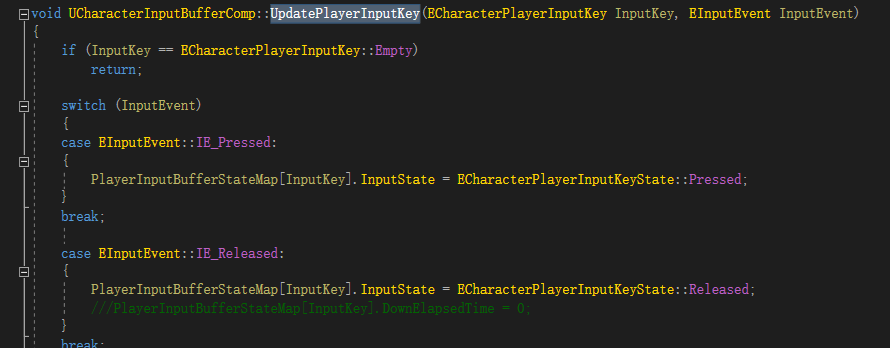
UpdatePlayerMoveInput



已经没有调用了



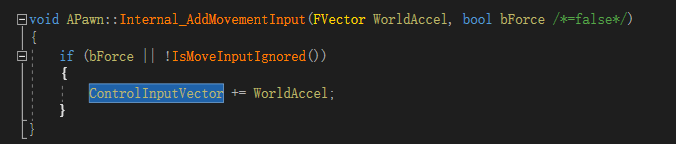
也已经弃用了。



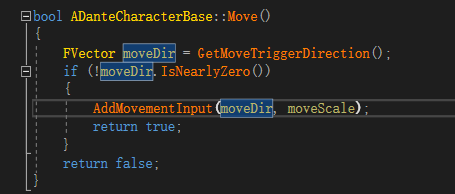
也弃用了。

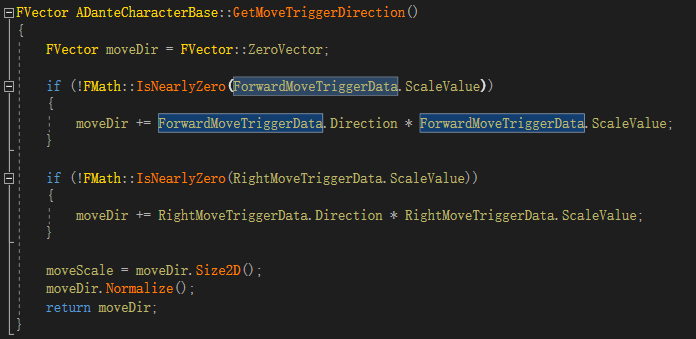
玩家移动问题探究：

CharacterStateMachinBase中，移动状态，调用了DanteCharacterBase的Move，最终调用到了APawn的

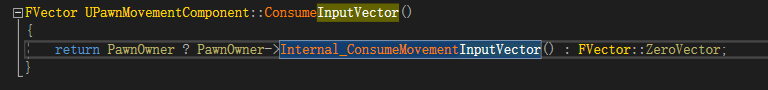


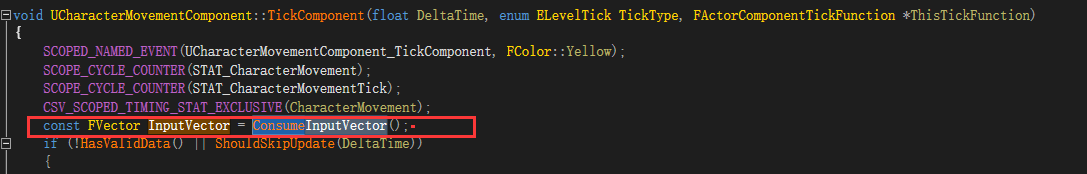
而这个WorldAccel，是这里来的：





所以只要记录了ForwardMoveTriggerData，理论上来说，移动方向是没问题的,看看这个是如何影响Movement的:



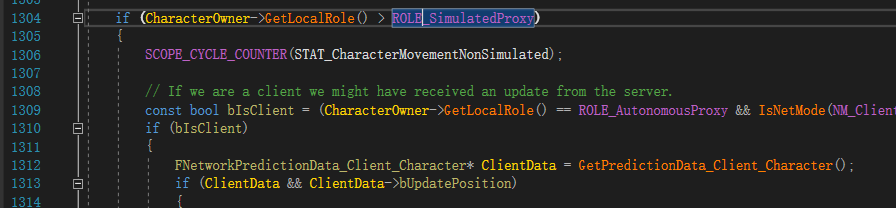


tickComponent中，会去取InputVector！

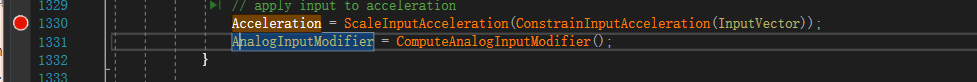


所以这个velocity是根据其余的算出来的，应该是保持其余参数正确，他就正确了。

那为什么在replay的时候，Acceleration是0呢，因为，在TickComponent中，



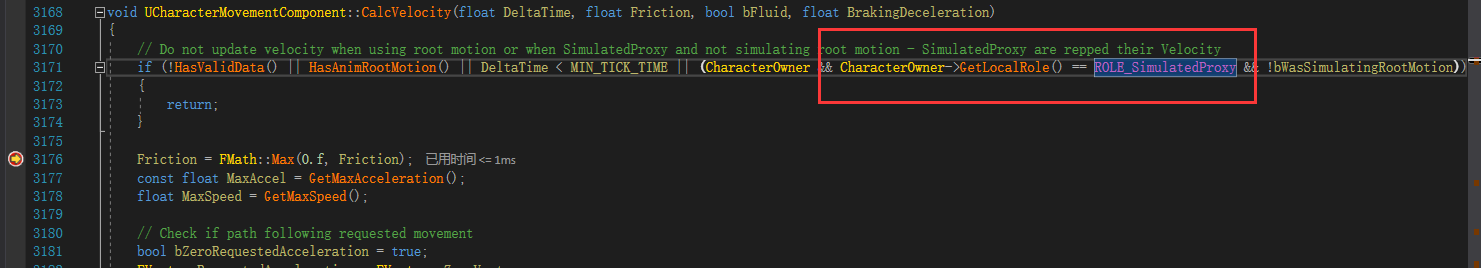
这一步直接return了，并没有执行到为Acceleration赋值的部分，同样的



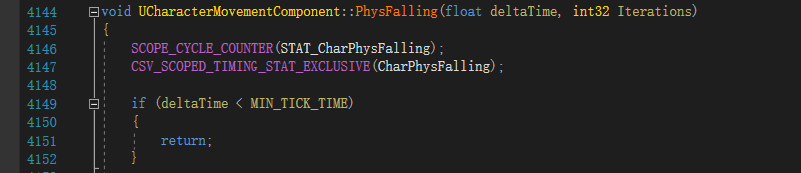
AnalogInputModifier也并没有执行！

哪怕是在replay中，也是执行的PerformMovement。

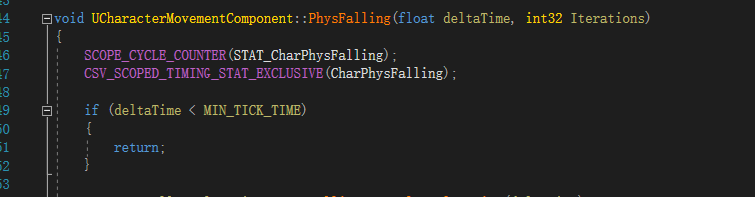
在replay中，CalcVelocity是不执行的！（这是不是就是关于速度为什么不能正确显示的原因？）



下降，在跳跃的时候执行，在replay中看看是否执行呢：

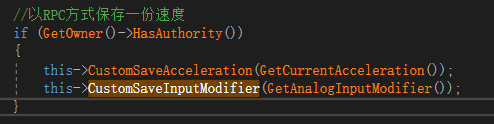


跳跃为什么现在是正常的呢，因为PhysFalling是正常运行的，（但是之前为什么，不正常？肯定是参数不对），所以能正确计算Velocity





所以，为了让玩家正常行走，



自定义的Acceleration与InputModifier是必须额外记录的！

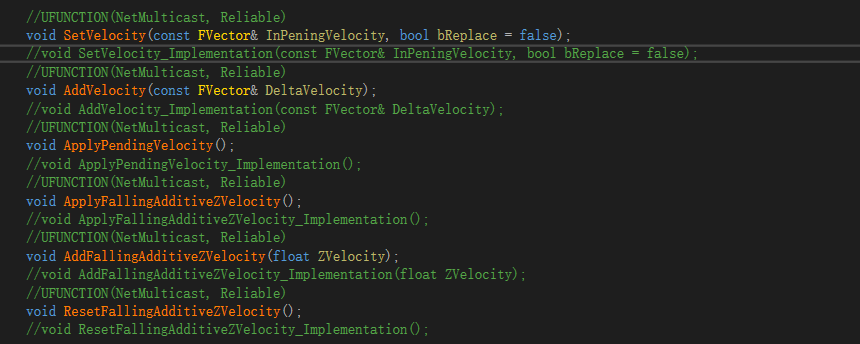
之所以要记录TriggerMove这两个函数，因为这两个，是方向移动的输入记录。





## 不使用RPC记录曲线移动，用过逻辑去调用曲线移动

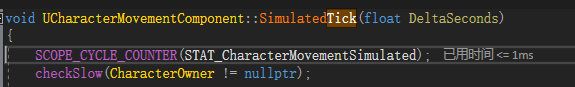
下一步看看：如果将这些记录为普通函数，会怎样呢~还挺正常的！也不意外啊，因为所有的函数，都是根据参数即时执行的！



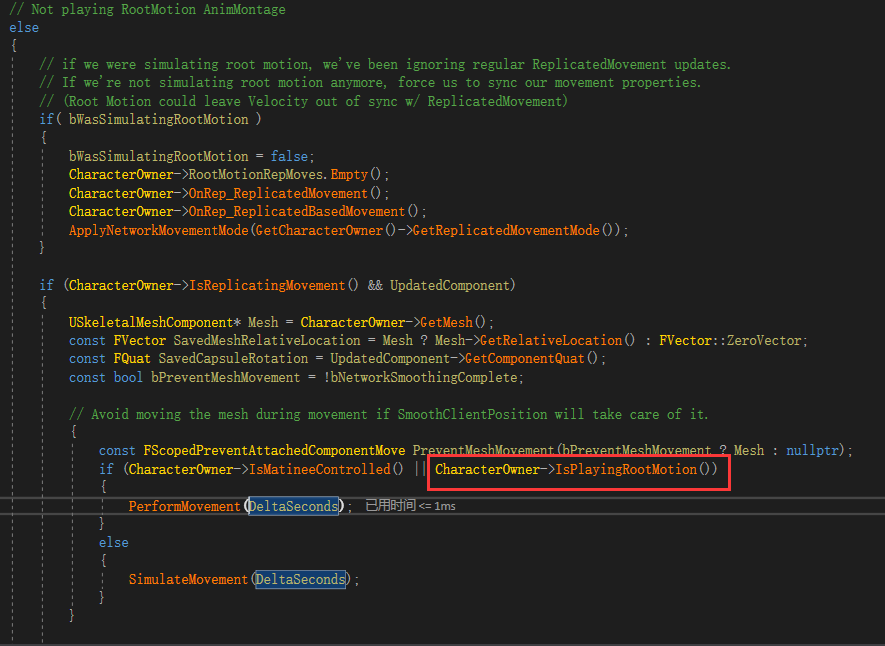
但是还有个疑问，如果没有速度了，那么是怎么移动的呢？

在此研究并记录Actor的位置同步：

TickComponent 首先replay中更新是走的



一直跳过直到：



应该判断IsPlayingRootMotion为true，还是调用的PerformMovement,而当前的速度

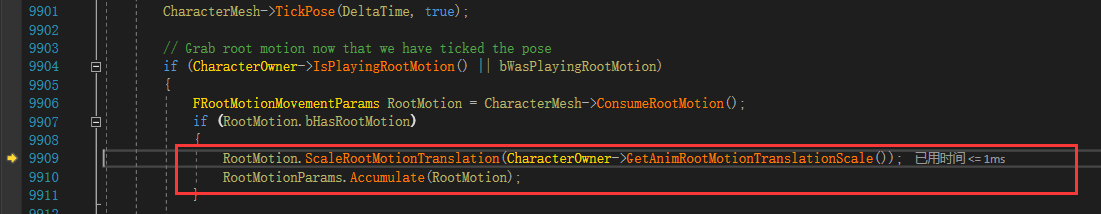


是有的！**注意：这时候调用的PerformMovement，是会先调用DanteCharacterMovementComp的PerformMovement！调用了**ApplyPendingVelocity！**如果此时需要用到曲线的话，就会自动去应用曲线！如果之后出问题，可以看看是不是需要记录**bReplacingVelocity。



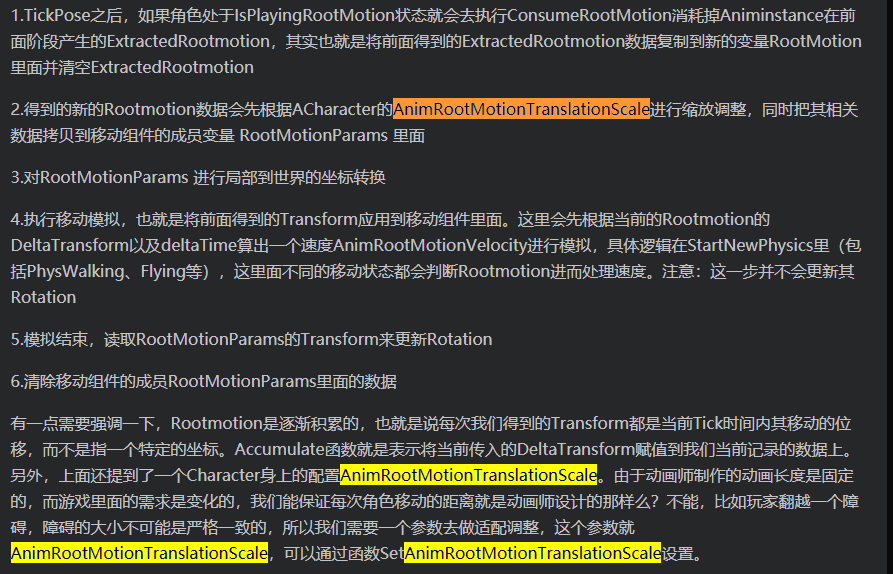
函数内的这里有叠加一下速度，

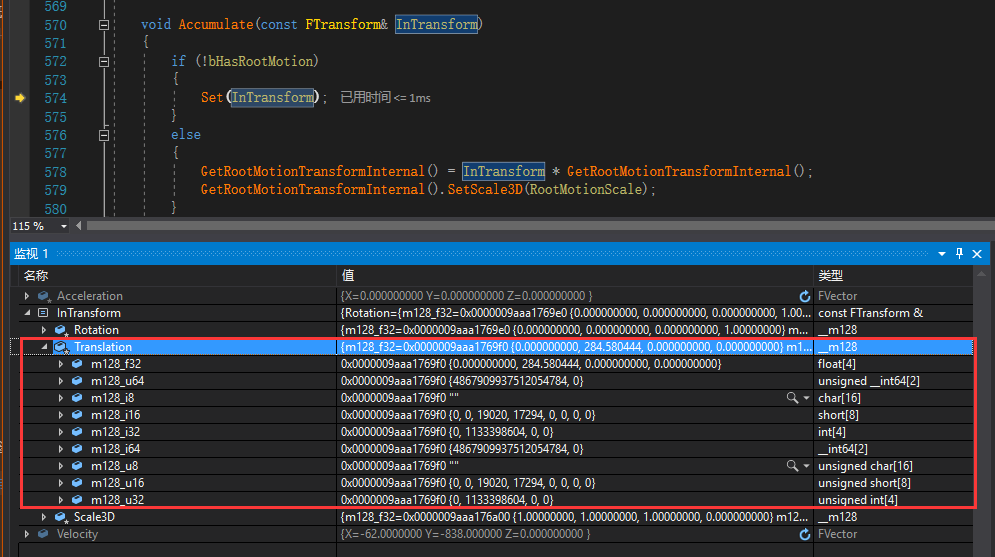
**在TickCharacterPose这个函数中，会去根据rootMotion计算位置移动！！！**



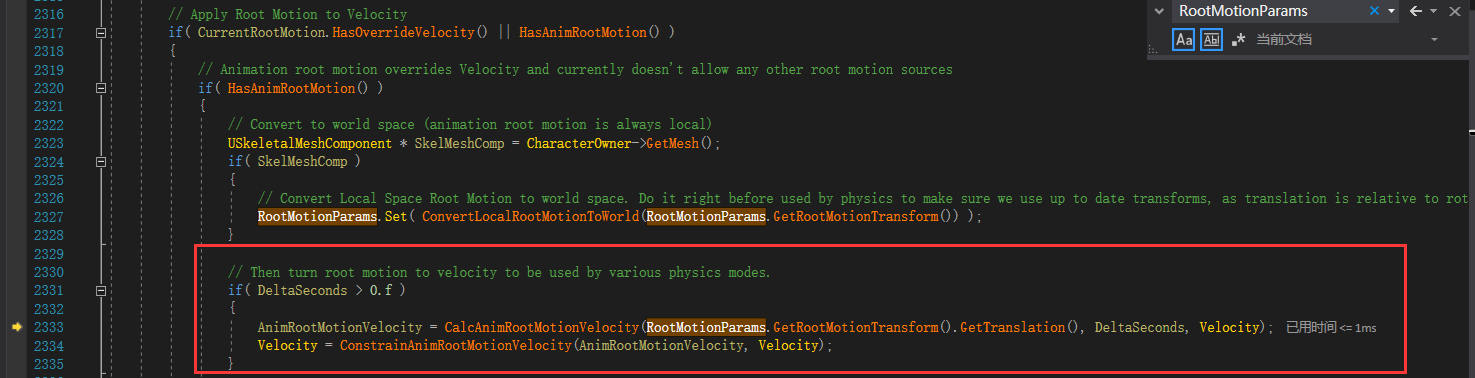
RootMotion.ScaleRootMotionTranslation

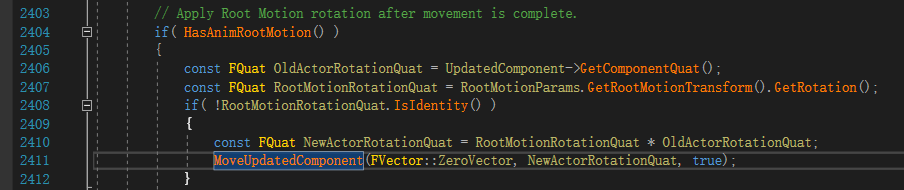
之后的Accumulate中，设置了一个RootMotionParams

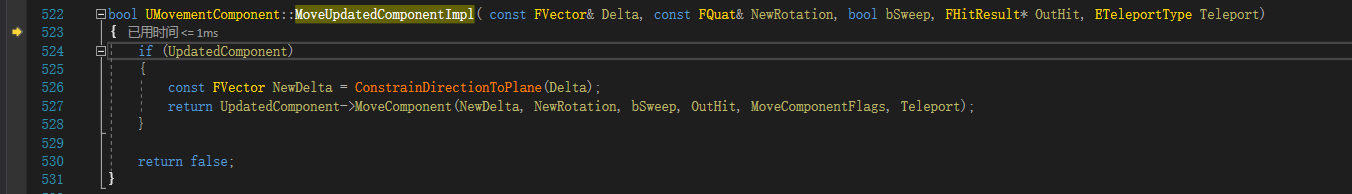




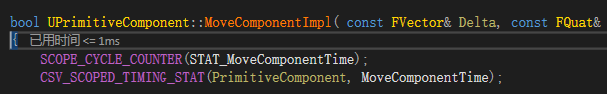
在上一步的TickCharacterPose计算完rootMotion的移动位置之后，

  
在下面就应用到Velocity上了！之后会将rootmotion应用到





中，进而调用，这里应该就是进行正式移动了。



之后会发送范围移动完毕的回调，再更新movemengcomponent的速度

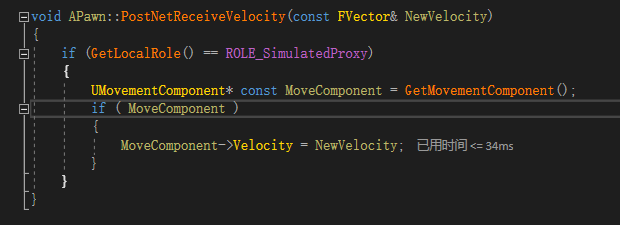




最后将本次的速度等记录下来，一次移动完毕。SimulatedTick

所以如果没有rootMotion，应该就不能动了？

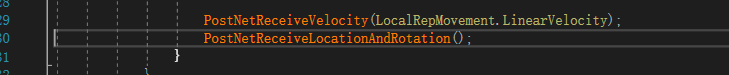
## 除此之外，还有一个地方影响了replay的速度！



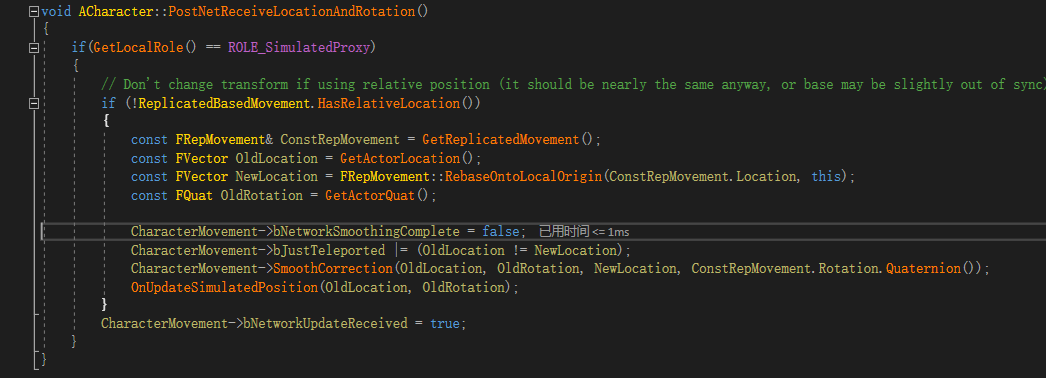
就是在收到网络包，ReceiveBunch,ProcessBunch之后，进入到



等函数中，进行Actor的property同步~其中



还同步了一下位置



由断点上可以看到，是先同步了位置与速度，然后才去执行了TickComponent，重新根据速度计算了一下移动等。

## 移动组件详解的文章

<https://blog.csdn.net/u012999985/article/details/78755939>

## 所以所以，总结一下：

Replay中是会同步位置与速度的，但是同步完毕之后，还是会走正式的计算流程，所以，哪怕在运行record的时候，并没有记录下当时的额外变化的速度，理论上还是能够通过正确的逻辑运行出来的，因为Input也是根据RPC进行的记录，所以应该也能很好的还原出，正确的情况。

也正是因为有同步位置，所以有可能出现闪烁，但是只要运行逻辑正确，理论上不会，这也是为什么就算敌人没有在replay中攻击到玩家，玩家也会发生位移的原因。

# 接下来处理下敌人不会攻击的问题