

客户端播放动画：

UAnimInstance::Montage\_Play()

1、此函数以秒为单位返回蒙太奇的长度，如果蒙太奇长度为0，则表示播放失败。  
2、调用HasRootMotion()判断蒙太奇是否有rootmotion，如果存在，则把刚刚创建并初始化的FAnimMontageInstance对象赋值给RootMotionMontageInstance（激活的蒙太奇实例），后面会用到。

客户端：  
移动组件tick  
主要是PerformMovement()内的流程

UCharacterMovementComponent::TickComponent()

移动组件tick，此函数计算加速度等必要数据。

UCharacterMovementComponent::ReplicateMoveToServer()

判断是否是重要的move数据、是否要组合发送等。

UCharacterMovementComponent::PerformMovement()

执行本地移动

UCharacterMovementComponent::HasRootMotionSources()

判断是否有RootMotionSources，在该函数中，如果CurrentRootMotion没有ActiveRootMotionSources，则调用IsPlayingRootMotion()判断，而该函数之后会调到IsPlayingNetworkedRootMotionMontage()，最后获得在初始化流程中初始化的蒙太奇实例，然后返回!RootMotion是否被禁用。

回到  
UCharacterMovementComponent::PerformMovement()函数中继续执行。

FRootMotionSourceGroup::CleanUpInvalidRootMotion()

如果有RootMotionSources，则清理无效的RootMotion

回到  
UCharacterMovementComponent::PerformMovement()函数中继续执行。

UCharacterMovementComponent::ApplyAccumulatedForces()

累加力

回到  
UCharacterMovementComponent::PerformMovement()函数中继续执行。

UCharacterMovementComponent::UpdateCharacterStateBeforeMovement()

更新角色状态

CharacterOwner->IsPlayingRootMotion()

最终使用到初始化流程中的RootMotionMontageInstance数据，会判断是否是RootMotionFromEverything或RootMotionFromMontagesOnly

回到  
UCharacterMovementComponent::PerformMovement()函数中继续执行。

UCharacterMovementComponent::TickCharacterPose()

角色pose tick

USkeletalMeshComponent::TickPose()

虽然mesh tick中会调用该函数，但是在这里我们需要强制执行一次

.....

请参考《mesh tick流程》

回到  
UCharacterMovementComponent::TickCharacterPose()函数中继续执行。

USkeletalMeshComponent::ConsumeRootMotion()

会获取到我们在动画tick时获得的ExtractedRootMotion并清空动画tick中的ExtractedRootMotion。

回到  
UCharacterMovementComponent::TickCharacterPose()函数中继续执行。

ScaleRootMotionTranslation()

压缩调整rootmotion

回到  
UCharacterMovementComponent::TickCharacterPose()函数中继续执行。

RootMotionParams.Accumulate(RootMotion);

进行缩放调整，然后把RootMotion存放到RootMotionParams中

回到  
UCharacterMovementComponent::PerformMovement()函数中继续执行。

CharacterOwner->ClientRootMotionParams = RootMotionParams;

赋值，方便在网络同步时调用

回到  
UCharacterMovementComponent::PerformMovement()函数中继续执行。

UCharacterMovementComponent::ConvertLocalRootMotionToWorld()

转换成世界坐标

回到  
UCharacterMovementComponent::PerformMovement()函数中继续执行。

UCharacterMovementComponent::CalcAnimRootMotionVelocity()

根据RootMotion计算AnimRootMotionVelocity，约束RootMotion分量，对Z坐标进行判断，并返回Velocity

回到  
UCharacterMovementComponent::PerformMovement()函数中继续执行。

UCharacterMovementComponent::ConstrainAnimRootMotionVelocity()

约束RootMotion分量，对Z坐标进行判断，并返回Velocity（所以，RootMotion中的位移信息最终转换成了Velocity）

回到  
UCharacterMovementComponent::PerformMovement()函数中继续执行。

.....

移动模拟，请参考《移动流程》（移动流程，进行模拟的时候，会使用到我们上一步计算的Velocity）

客户端：  
Mesh tick

USkeletalMeshComponent::TickComponent()

骨骼 mesh component tick

USkinnedMeshComponent::TickComponent()

USkeletalMeshComponent::TickPose()

骨骼 mesh pose tick

USkeletalMeshComponent::TickAnimation()

动画系统 tick，子动画更新

UAnimInstance::UpdateAnimation()

动画更新

回到UAnimInstance::UpdateAnimation()函数中继续执行。

UAnimInstance::PreUpdateAnimation()

如果设置了需要在主线程更新，则调用

FAnimInstanceProxy::PreUpdate()

调用多线程初始化代理更新，如：组件位移等

回到UAnimInstance::UpdateAnimation()函数中继续执行。

UAnimInstance::UpdateMontage()

蒙太奇更新，更新蒙太奇权重和RootMontion等。

UAnimInstance::Montage\_Advance()

主要处理root motion逻辑。遍历蒙太奇数组，如果根运动可以被提取（与RootMotionMode有关），则提取上一次更新的FRootMotionMovementParams数据（即RootMotionParams）。

FAnimMontageInstance::Advance()

更新root motion移动信息

UAnimMontage::ExtractRootMotionFromTrackRange()

从时间片中提取RootMotion，并返回FTransform RootMotion（位移信息）。把这一帧动画播放前的播放位置和当前执行的位置作为参数传递。

UAnimCompositeBase::ExtractRootMotionFromTrack()

从FAnimTrack中提取RootMotion

FAnimTrack::GetRootMotionExtractionStepsForTrackRange()

FAnimSegment::GetRootMotionExtractionStepsForTrackRange()

回到  
UAnimCompositeBase::ExtractRootMotionFromTrack()函数中继续执行。

UAnimSequence::ExtractRootMotionFromRange()

更新动画状态，发送通知

回到  
UAnimCompositeBase::ExtractRootMotionFromTrack()函数中继续执行。

RootMotion.Accumulate(DeltaTransform);

更新RootMotion

回到FAnimMontageInstance::Advance()函数中继续执行。

OutRootMotionParams->Accumulate(RootMotion);

拿到之前获得的RootMotion位移信息，作为参数传递给Accumulate()函数，以更新OutRootMotionParams。OutRootMotionParams其实就是在Montage\_Advance()中把ExtractedRootMotion作为参数传递给Advance()。（也就是说该表达式在更新ExtractedRootMotion）

回到FAnimMontageInstance::Advance()函数中继续执行。

FAnimMontageInstance::HandleEvents()

处理动画信息

回到UAnimInstance::UpdateAnimation()函数中继续执行。

UAnimInstance::PostUpdateAnimation()

通过proxy，提取最终的ExtractedRootMotion

服务器流程请参考上篇

客户端收到服务器move ack后的流程：调用UCharacterMovementComponent::ClientAdjustRootMotionPosition\_Implementation()。在这个函数中，如果客户端与服务器的montage不一样，则调用SetPosition()将本地的montage的位置设置成服务器的位置。其他同上篇文章，这里不再介绍。

模拟端流程：

ACharacter::OnRep\_RootMotion()

服务器在ACharacter::PreReplication()中对RepRootMotion进行赋值，模拟端在该函数中更新RootMotionRepMoves。

UCharacterMovementComponent::SimulatedTick()

UCharacterMovementComponent::TickCharacterPose()

动画tick，同上

回到  
UCharacterMovementComponent::SimulatedTick()函数中继续执行。

UCharacterMovementComponent::SimulateRootMotion()

与客户端的内容差不多，只不过这里在最后：如果RootMotionRotation不为默认值，则调用MoveUpdatedComponent()进行修正。（其他的数据同步请参考上篇套一下即可。）

回到  
UCharacterMovementComponent::SimulatedTick()函数中继续执行。

ACharacter::SimulatedRootMotionPositionFixup()

计算RootMotionRepMove.RootMotion.Position和本地的Position之差，如果大于KINDA\_SMALL\_NUMBER，则调用SmoothCorrection()进行修正。