移动组件跳跃相关学习

额外知识点：Actor创建完成会调用PostSpawnInitialize，在PostSpawnInitialize方法中执行

USceneComponent\* const SceneRootComponent = FixupNativeActorComponents(this);

设置Root组件（SceneComponent提供了两大能力：一是Transform，二是SceneComponent的互相嵌套）。所以Actor父子之间的“关系”其实隐含了许多数据，而这些数据都是在Component上提供的。Actor其实更像是一个容器，只提供了基本的创建销毁，网络复制，事件触发等一些逻辑性的功能，而把父子的关系维护都交给了具体的Component，所以更准确的说，其实是不同Actor的SceneComponent之间有父子关系，而Actor本身其实并不太关心。

PrimitiveComponents是包含或生成某种几何图形的场景组件，通常被渲染或用作碰撞数据。

对于各种类型的几何图形，有几个子类，但到目前为止最常见的是形状组件（胶囊、球体、长方体）、静态网格组件和骨架组件。

\*shapecomponents生成用于碰撞检测但未渲染的几何体，而staticmeshcomponents和skeletalmeshcomponents包含已渲染但也可用于碰撞检测的预构建几何体。

* 移动组件与RootComponent的关系：

玩家ACharacter构造初始化的时候会创建一个UCharacterMovementComponent组件赋值给CharacterMovement成员变量，并把移动组件的UpdatedComponent设置成玩家的胶囊体组件CapsuleComponent

* ACharacter相关成员变量及方法注释：

CapsuleComponent：character初始化的时候会构建个胶囊体组件作为这个character的RootComponent

CharacterMovement：character初始化的时候会构建移动组件，并把移动组件的UpdatedComponent设置成这个胶囊体组件。

JumpCurrentCount：当前的跳跃次数。这个变量是用来跟踪当前执行的跳跃次数，这在检查跳转输入中递增，是被用在CanJump\_Implementation方法里面，在 OnMovementModeChanged.改变后重置为0

JumpMaxCount：character能执行的最大跳跃次数（可用来做多段跳）。请注意, 如果 JumpMaxHoldTime 为非零, 并且不调用停止跳转, 则玩家可以执行和无限数量的跳跃。因此, 它通常是最好在跳转输入停止 (如按钮向上事件) 时调用停止跳转。

JumpKeyHoldTime：跳跃键保持时间。\*这是玩家按住跳跃键的时间，以秒为单位。

JumpMaxHoldTime：可以按住跳转键的最长时间。请注意，如果在达到最大调阅保持时间之前未调用StopJumping（），然后character将继续接收垂直速度。因此通常当跳转输入停止时（如按钮向上事件），最好调用StopJumping（）

bPressedJump：按下跳跃，当玩家想要跳跃的时候，比如按了空格键，调Jump会把这个变量设为True，StopJumping的时候设为false

CheckJumpInput(float DeltaTime)：检测跳跃输入，如果跳跃按钮按下的时候触发跳跃，可重写

ClearJumpInput(float DeltaTime)：清除跳跃输入，可重写。在移到组件PerformMovement执行移动之后会调这个清除跳跃输入, 允许移动事件触发它进行下一次更新。注意这个和ResetJumpState不一样，这个清除只是计算JumpKeyHoldTime +1，判断是否达到角色最大跳跃时间，达到了才执行bPressedJump = false

Jump：使玩家在下一次更新时跳跃，调这个方法时会设置bPressedJump为true，玩家在移动组件TickComponent下一个tick检测CheckJumpInput若bPressedJump为true时会执行跳跃。比如按键按下的时候需手动调用该函数

StopJumping：使玩家在下一次更新时停止跳跃。比如输入事件按键弹起的时候停止跳跃。

CanJump：检测玩家在当前状态是否能跳跃。可以通过在蓝图中实现自定义CanJump事件来覆盖或扩展默认实现。

OnJumped：当角色已经开始跳跃时触发事件。可重写OnJumped\_Implementation

Falling：移动组件移动模式改变时，如果是掉落模式则通知玩家执行Falling，可重写

OnMovementModeChanged：移动模式改变通知，由移动组件通知，可重写，蓝图重写K2\_OnMovementModeChanged在蓝图那边命名也叫OnMovementModeChanged

ResetJumpState：重置跳跃状态，移动模式改变的时候通知如果不是跳跃模式则要重置。重置会设置bPressedJump为false及JumpCurrentCount为0

NotifyJumpApex：当角色的跳跃达到顶点时调用。需要charactermovement->bnotifyapex=true

* 玩家移动组件UCharacterMovementComponent相关成员变量及方法注释：：

JumpZVelocity：跳跃时的初始速度（瞬时垂直加速度）。

DoJump：真正跳跃的执行入口Perform jump. Called by Character when a jump has been detected because Character->bPressedJump was true. Checks CanJump().

JumpOff：

PhysFalling：处理下落运动，真正下落过程是在这里面写的

SimulateMovement：

NotifyJumpApex：若bnotifyapex为True，当角色的跳跃达到顶点时调用通知，默认bnotifyapex是false

* 玩家跳跃状态检测流程：

1. 绑定触发响应事件

绑定按键事件，按键按下的时候触发Jump，弹起的时候触发StopJumping，Acharacter::Jump的时候会设置bPressedJump为True，StopJumping会设置bPressedJump为false及ResetJumpState重置状态

void APlayerCharacter::SetupPlayerInputComponent(class UInputComponent\* PlayerInputComponent)

{

// Set up gameplay key bindings

check(PlayerInputComponent);

PlayerInputComponent->BindAction("Jump", IE\_Pressed, this, &ACharacter::Jump);

PlayerInputComponent->BindAction("Jump", IE\_Released, this, &ACharacter::StopJumping);

}

1. 移动组件下一个Tick检测跳跃输入

UCharacterMovementComponent::TickComponent-》如果玩家是本地控制的角色则执行ACharacter::CheckJumpInput

1. ACharacter检测跳跃输入

ACharacter:: CheckJumpInput -》如果bPressedJump为true，如果JumpCurrentCount=0且玩家移动模式是掉落状态则JumpCurrentCount+1，-》 检查能不能跳CanJump（根据JumpCurrentCount是否<JumpMaxCount检查），蓝图可重写，比如一些状态下不能跳跃 -》通知移动组件执行跳跃DoJump -》跳跃成功之后JumpCurrentCount+1并通知OnJumped

1. UCharacterMovementComponent::DoJump(bool bReplayingMoves)

判断CharacterOwner->CanJump()及是否能上下移动，True的话设置移动模式为掉落模式SetMovementMode(MOVE\_Falling);

1. 移动模式改变

移动组件移动模式改变会通知CharacterOwner->OnMovementModeChanged ，如果bPressedJump为false及玩家不是掉落模式则重置跳跃状态ResetJumpState();

* 单机或者ListenServer服务器执行的跳跃流程

UCharacterMovementComponent::TickComponent

* DedicateServer自治(Autonomous)权利的角色的跳跃流程
* DedicateServer模拟(Simulate)角色的跳跃流程