在了解UE4的移动逻辑之前，我们先熟悉下碰撞的基础接口

UE4移动中碰撞检测主要使用PhysX的Geometry Queries（几何查询）功能，包括射线检测RayCasts，重叠检测Overlaps，渗透深度计算Penetration Depth，Sweeps检测，InitialOverlaps检测。

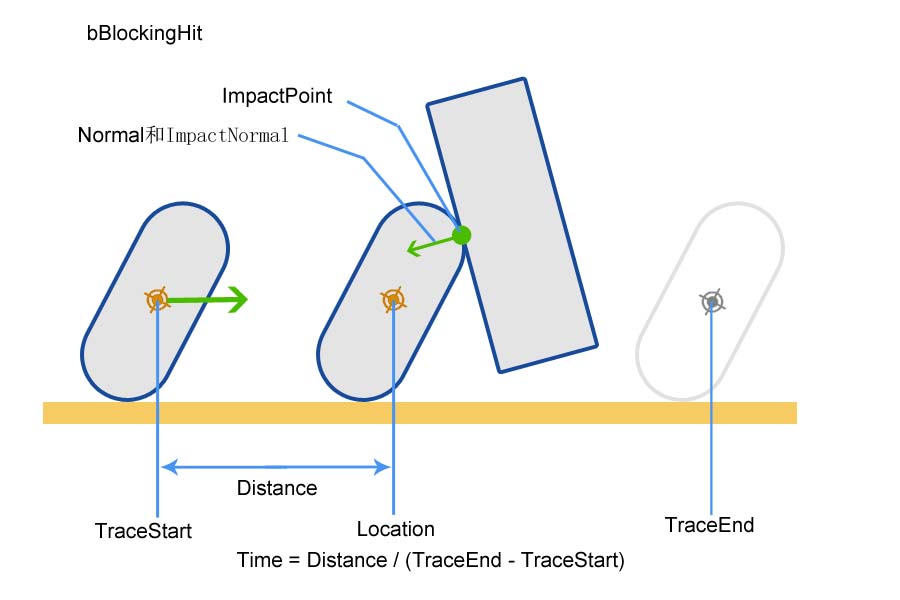
UE4把查询后返回的hit封装成了FhitResult，FhitResult的结构如下

### FHitResult

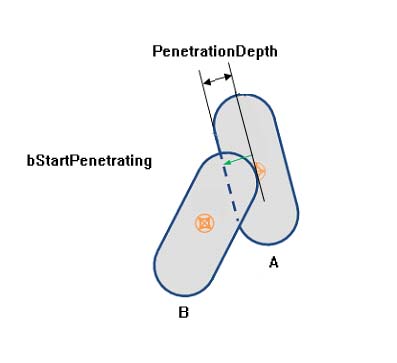
|  |  |
| --- | --- |
| bBlockingHit | 是否发生碰撞 |
| bStartPenetrating | 是否在检测开始就有渗透情况 |
| Time | 碰撞后实际移动距离除以检测移动距离 |
| Distance | 碰撞后实际移动距离 |
| Location | 碰撞后最终位置 |
| ImpactPoint | 碰撞接触点 |
| Normal | 碰撞切面法向量 |
| ImpactNormal | 碰撞切面法向量（非胶囊体和球体检测与Normal不同） |
| TraceStart | 检测开始位置 |
| TraceEnd | 检测结束位置 |
| PenetrationDepth | 渗透深度 |

我们可以借助以下两种移动中常见的情况熟悉一下这些参数，

第一种是常见的胶囊体Sweep查询，查询开始结束分别是TraceStart和TraceEnd两个位置，如果碰到了障碍，bBlockingHit就是true，胶囊体最终会停在Location位置，它移动的距离是Distance，Time是一个0到1的值，表示实际移动距离比查询距离。还有一些可能会用的参数，比如碰撞接触点ImpacePoint，碰撞切面法向量Normal和ImpactNormal



第二种常见的情况通常是InitialOverlaps，也就是移动查询的开始检测到了重叠，这时候bStartPenetrating是true，通过渗透深度计算可以获得PenetrationDepth，这个参数对于处理移动中穿透的情况非常重要



将PhysX的渗透深度计算Penetration Depth，Sweeps检测，InitialOverlaps检测的查询结果封装成一个FhitResult，这是UE4的第一层封装。

接下来我们再来看第二层封装，SafeMoveUpdatedComponent，它是UE4移动最关键的函数，几乎所有的移动都要靠它来完成。它的主要功能有以下几点

1. 筛选Hit
2. SetLocation并递归更新子组件
3. UpdateOverlap，Overlap检测
4. 解决渗透的情况，bStartPenetration
5. 返回检测结果Hit

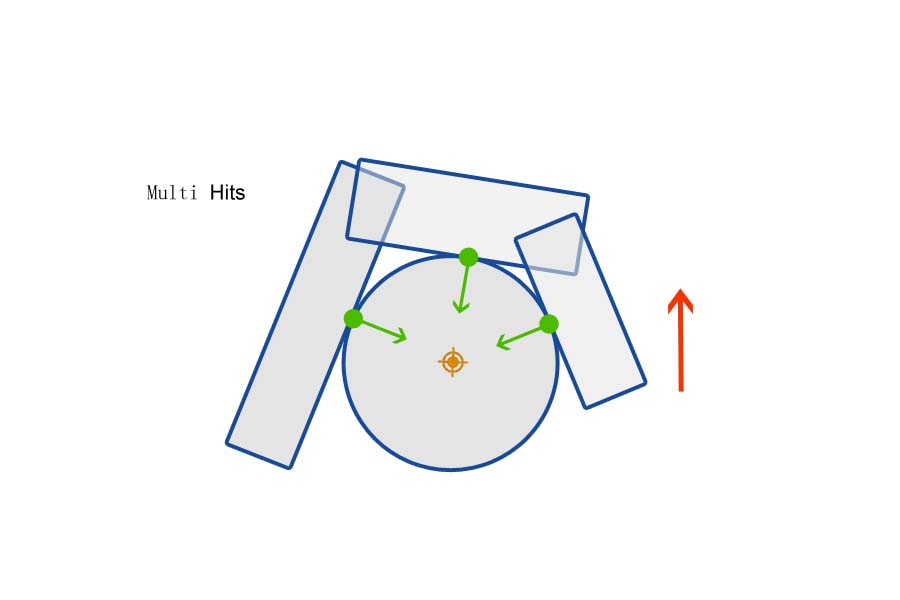
下面我们分别说明下这些功能

## ▽▽▽▽▽▽ SafeMoveUpdatedComponent ▽▽▽▽▽▽

#### ▽▽▽▽ UPrimitiveComponent::MoveComponentImpl ▽▽▽▽

调用SweepMulti得到的所有Hit需要拉回微小的距离(缩小hit.time)，避免因为计算精度的问题导致跟碰撞物重叠

如果检测到多个block hit，优先选择不是在初始位置就检测到block的hit，否则的话选取跟运动方向最相反的hit



根据hit.time计算移动到的最终位置，并setWorldLocationAndRotation，以及更新ComponentToWorld Transform矩阵，连带更新父组件和递归更新子组件，以及导航网格数据，Bounds，RenderTransform设置dirty以及必要的话更新 PhysicsTransform

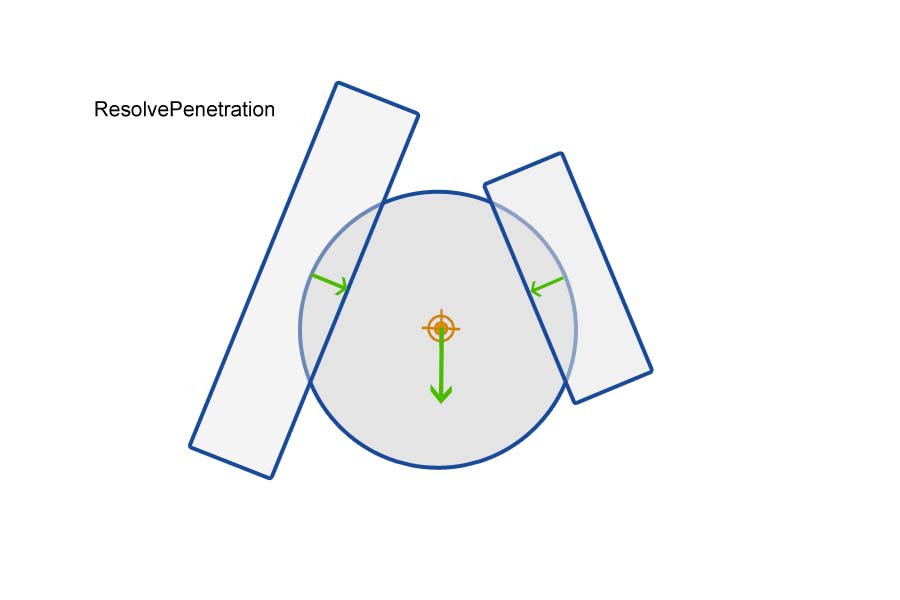
然后需要UpdateOverlaps，调用OverlapMulti，如果检测到overlap，比如进入水里，只有发生移动了才会更新Overlap Components列表，删除不再Overlap的Component，新增新的Component。并更新子Component的 Overlap Components列表以及PhysicsVolume

#### △△△△ UPrimitiveComponent::MoveComponentImpl △△△△

MoveComponentImpl返回的hit.bStartPenetrating如果是true，即在初始位置检测到了overlap，需要

#### ▽▽▽▽▽▽ ResolvePenetration ▽▽▽▽▽▽

简单来讲就是将物体朝hit的normal方向拉回PenetrationDepth远的距离，获得调整向量Adjustment，如果在拉回过程中又发生了overlap，加上这次overlap计算的调整向量再做调整移动，仍然无法移动，改为加上运动向量再做调整移动，如果最终调整移动成功了，就再次尝试最开始的移动。



#### △△△△△△ ResolvePenetration △△△△△△

## △△△△△△ SafeMoveUpdatedComponent △△△△△△

介绍完了这些基础的移动接口，就可以进一步了解UE4的移动逻辑了，我们按照函数的调用顺序来梳理一遍，看看UE4在一帧移动究竟做了什么。

## ▽▽▽▽▽▽ PerformMovement ▽▽▽▽▽▽

1.根据输入向量InputVector计算加速度向量Acceleration

2.随着被骑乘物MovementBase（比如电梯，载具）移动

3.将冲力Impulse和推力Force作用于速度Velocity，一般用于击退和径向运动

4.根据不同的运动状态运动

MOVE\_None (不做运动)

MOVE\_Walking （踩地面上运动）

MOVE\_NavWalking （踩导航网格上运动）

MOVE\_Falling （在空中受重力加速度）

MOVE\_Flying （不受重力加速度的运动）

MOVE\_Swiming （在水中运动）

MOVE\_Custom （自定义运动，比如插值运动）

先看下MOVE\_Walking

### ▽▽▽▽▽▽ PhysWalking ▽▽▽▽▽▽

首先将速度和加速度的垂直方向分量设为0，方向始终保持在水平面上

#### ▽▽▽▽▽▽ CalcVelocity ▽▽▽▽▽▽

计算速度，先设置为RequestedVelocity（寻路组件PathFollowingComponent根据路径不断设置该速度）

加速度是0的时候，将受到减速度BrakingDeceleration和摩擦力的影响而减速

加速度不是0的时候，摩擦力将会影响速度方向改变快慢

Velocity = Acceleration \* DeltaTime

最后，如果支持RVOAvoidance，将会根据RVO重新计算速度，避免跟其他角色重叠在一起，效果就像被弹回来。

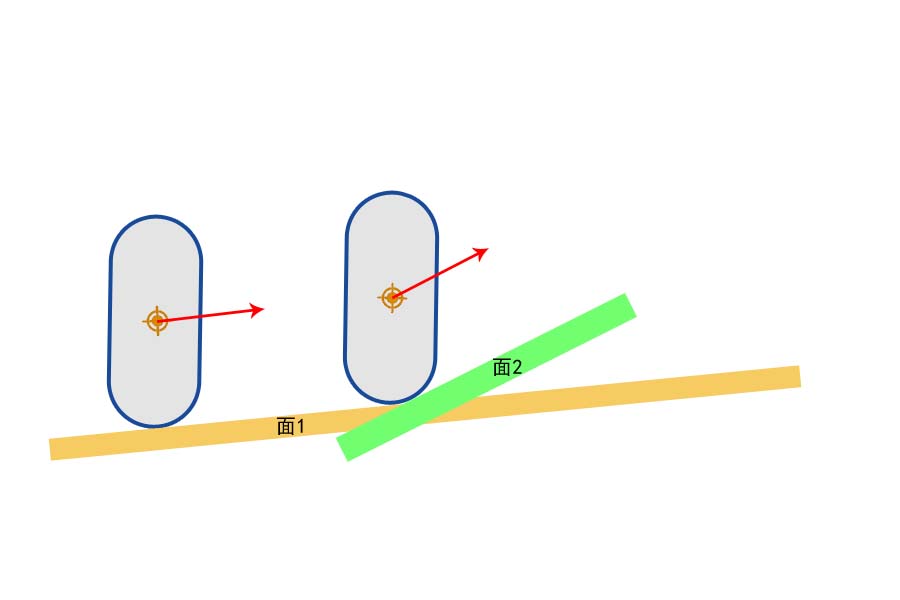
#### △△△△△△ CalcVelocity △△△△△△

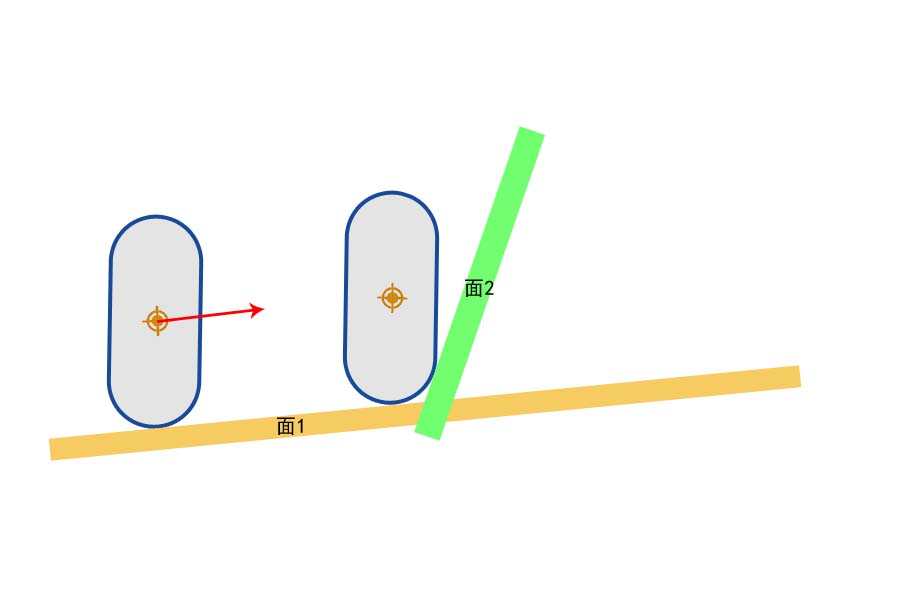
#### ▽▽▽▽▽▽ MoveAlongFloor ▽▽▽▽▽▽

计算移动向量Delta = Velocity \* DeltaTime

根据地面坡度调整移动向量方向，第一次调用SafeMoveUpdatedComponent

如果返回Hit结果是block，并且检测到碰到的斜面坡度较缓，可以调整移动向量沿着新的斜面运动，再次调用SafeMoveUpdatedComponent





如果返回的Hit结果还是block或者一开始的斜面就非常陡峭，可以开始尝试调用StepUp上楼

##### ▽▽▽▽▽▽ StepUp ▽▽▽▽▽▽

如果碰撞到胶囊体的上半部分，移动失败，直接返回。

首先向上移动抬脚高度，如果有bStartPenetratin则返回、

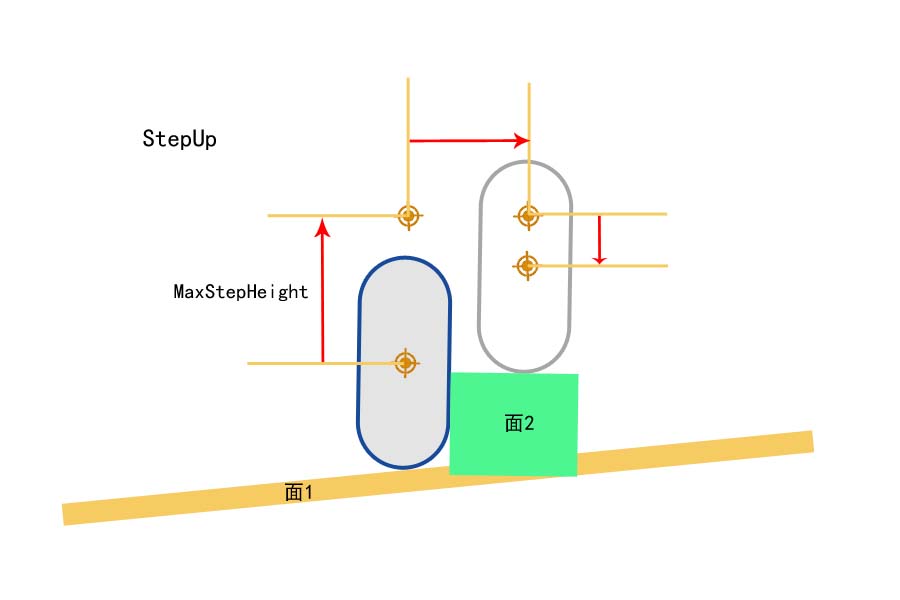
然后向前移动，检测到bBlockHit，然后调用SlideAlongSurface，如果仍然检测到没有移动，则直接返回。

最后向下移动，首先检测最终移动的垂直高度不能大于最大步高。如果hit不可行走，检测hit的法向量不能和运动方向相反，然后如果碰撞点比原来位置高，直接返回false，比原来位置低的话则允许滑下去。

剔除掉边缘检测到的hit

剔除掉垂直位移大于0并且最终hit不能行走的情况

最后向下移动的hit还可以用来检测Floor。



##### △△△△△△ StepUp △△△△△△

返回MoveAlongFloor，如果不能StepUp或者调用失败，则调用HandleImpact和SlideAlongSurface沿着斜面走。

##### ▽▽▽▽▽▽ HandleImpact ▽▽▽▽▽▽

发送MoveBlockedBy事件，如果开启bEnablePhysicsInteraction，可以通过给与刚体一个反推力来解决穿透问题

##### △△△△△△ HandleImpact △△△△△△

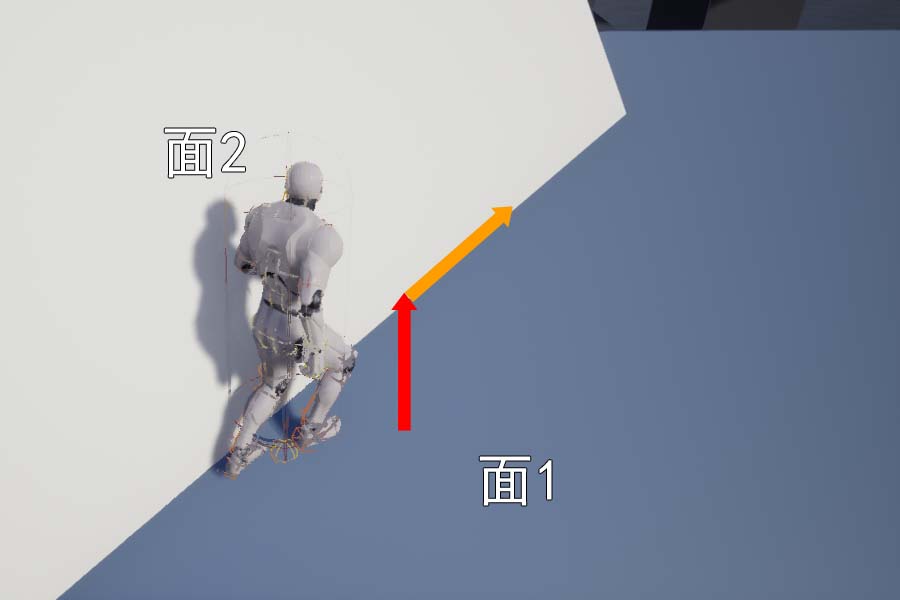
##### ▽▽▽▽▽▽ SlideAlongSurface ▽▽▽▽▽▽

如果是在地面移动，首先要调整一下面的法向量

如果法向量朝向朝上并且倾斜角度过于陡峭，则将法向量的垂直分量z舍弃，也就是说直接将斜面当做垂直面来处理。

如果法向量朝下，说明面挡住了胶囊体上半部分，并且可能离地面高度已经小于最小临界值，说明物体有挤入地面的风险，这种情况如果运动方向与地面法向量大致相反（向量点乘小于0，夹角大于90度），直接将法向量改为地面的法向量（宁可挤入斜面也不能挤入地面），如果不相反，则将斜面当做垂直面处理。

调整好法向量后调用ComputeSlideVector将移动向量投影到平面上，按照新的移动向量SafeMoveUpdatedComponent，如果返回hit结果block，说明又撞到了另一个面，先利用HandleImpact处理。然后调用TwoWallAdjust对移动向量SlideDelta做特殊处理



###### ▽▽▽▽▽▽ TwoWallAdjust ▽▽▽▽▽▽

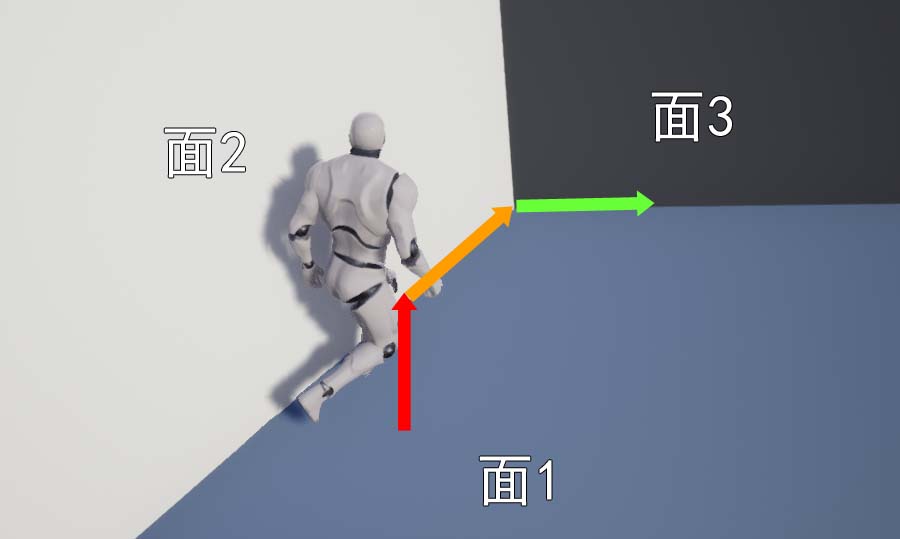
利用两个面法向量检测两个面的夹角，

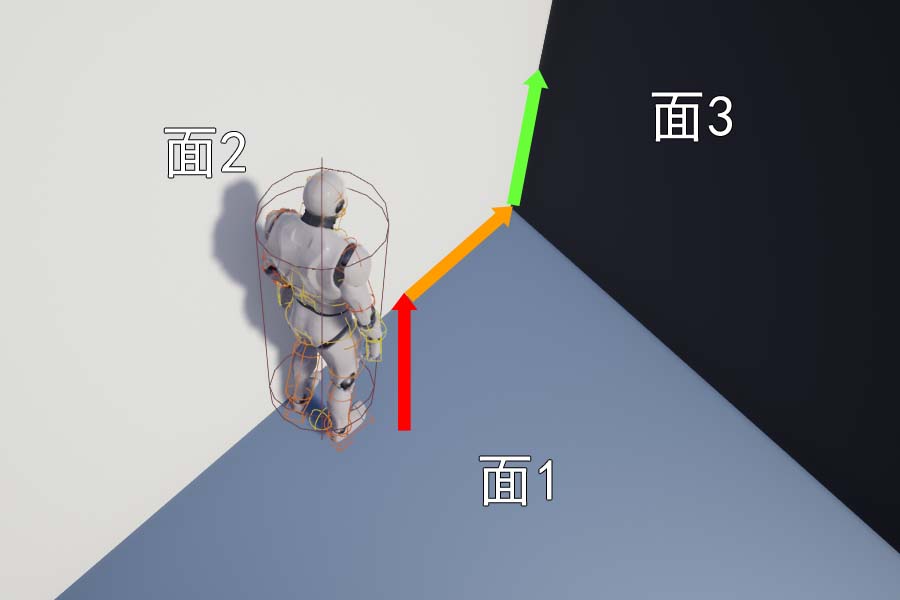
如果小于90度，计算两个法向量的叉乘向量，也就是同时垂直两个法向量的垂直向量。然后将前两次SafeMove移动后剩余的移动向量投影到这个垂直向量上，作为新的移动向量。

如果夹角大于90度，利用ComputeSlideVector将移动向量投影到新平面上，如果计算新的移动向量跟原来的移动向量相反，直接将移动向量置0，不做移动。

如果在地面移动，移动向量朝上并且最后碰到的斜面坡度平缓，则将移动向量改为沿坡移动。否则，直接将Z轴移动分量设为0，避免走上陡坡或者挤入地面。

调整后的SlideDelta不为0并且与最初移动向量不相反，则开始做SafeMoveUpdatedComponent，如果检测到block，调用HanleImpact做处理。





###### △△△△△△ TwoWallAdjust △△△△△△

##### △△△△△△ SlideAlongSurface △△△△△△

#### △△△△△△ MoveAlongFloor △△△△△△

到这里MoveAlongFloor就执行完了，返回PhysWalking

如果检测到IsFalling或者IsSwimming则利用剩余的时间开始新的运动 StartNewPhysics。

如果StepUp没有检测Floor的话，需要调用FindFloor，检测地面

#### ▽▽▽▽▽▽ FindFloor ▽▽▽▽▽▽

首先必须保证支持 QueryCollision

注意如果MovementBase即骑乘物不支持碰撞或者正在被清除，则强制检测地面，否则则不需要验证ValidateFloor

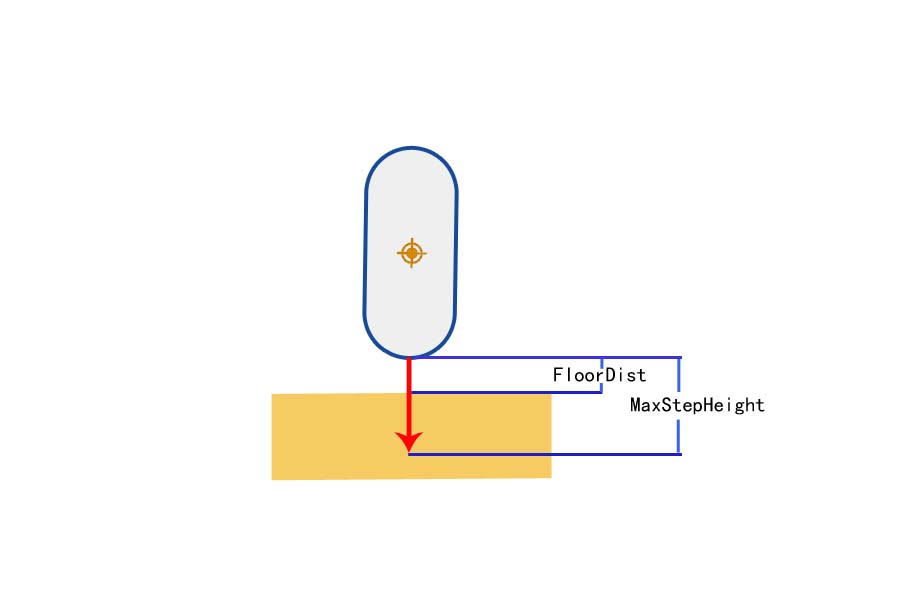
FFindFloorResult

|  |  |
| --- | --- |
| bBlockingHit | 跟地面有碰撞 |
| bWalkableFloor | 可以行走的地面 |
| bLineTrace | 是否是通过line trace检测出来的结果 |
| FloorDist | Sweep查询到地面的距离 |
| LineDist | LineTrace查询到地面的距离 |
| HitResult | 跟地面的FHitResult |

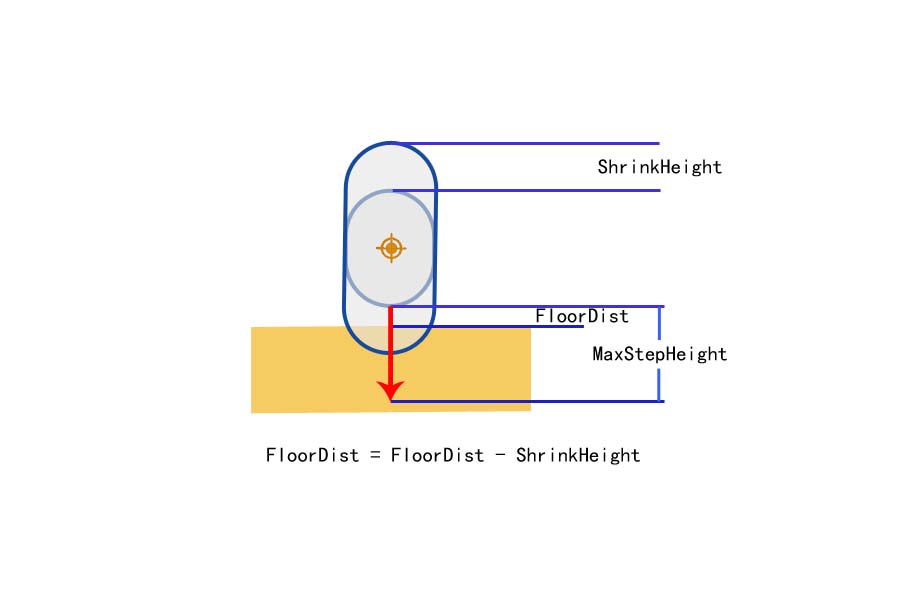
##### ▽▽▽▽▽▽ ComputeFloorDist ▽▽▽▽▽▽

如果之前StepUp已经计算出了sweep hit，需要保证是垂直向下检测，而且hit在胶囊体下半球

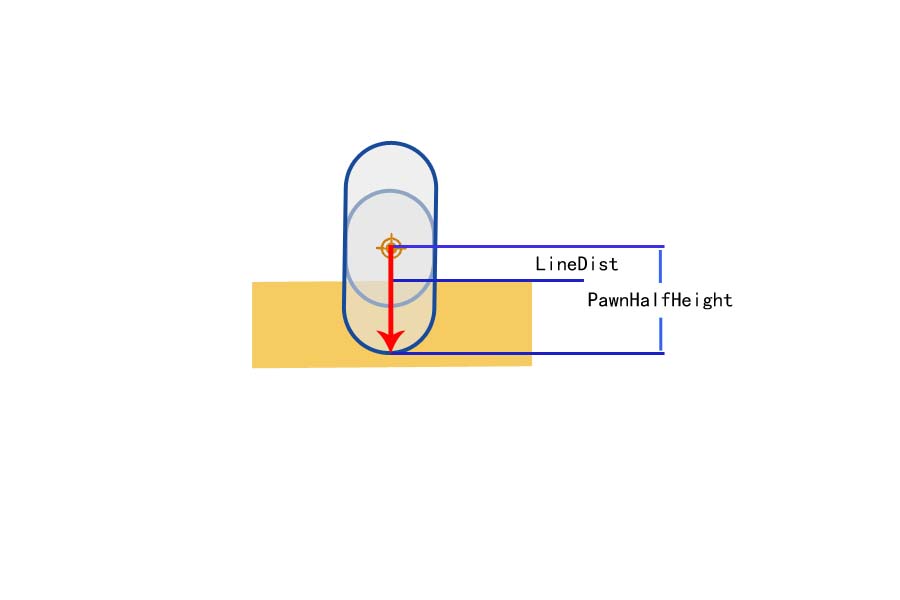
如果没有sweep hit则调用FloorSweepTest，垂直向下Sweep查询



如果返回的的Hit是block并且有渗透的话则需要用一个缩小的胶囊体来重新FloorSweepTest，如果这次没有穿透，可以返回FloorResult。



如果还是有穿透，这时候需要改用line trace，从胶囊体的中心向下trace胶囊体的半个身高，如果检测到了hit，则可以计算出穿透到地面以下的高度



注意无论是sweep还是line trace设置胶囊体向上抬的调整高度MaxPenetrationAdjust最大只能是胶囊体的半径，如果陷入地下的深度大于调整高度，一次调整是无法将胶囊体从地面抬出来的，往往需要多帧处理。

##### △△△△△△ ComputeFloorDist △△△△△△

#### △△△△△△ FindFloor △△△△△△

如果需要检测边缘，而且设置不能走下边缘并且当前所踩得地面不可走的话，

尝试跟当前位移方向垂直的两个方向尝试移动。仍然无法移动的话尝试Jump，CheckFall，开始下落。无法下落则撤销当前运动，让角色无法离开边缘

如果不需要检测边缘，则需要验证floor，

如果当前floor可走，调整离地面的高度AdjustFloorHeight，根据之前计算的FloorDist做SafeMoveUpdatedComponent，然后根据返回的Hit重新调整FloorDist

不可走的话如果还有穿透而且没有移动时间的情况，需要ResolvePenetration。

如果当前floor不可走，并且没有穿透，需要检测CheckFall

最后PhysWalking需要重新计算速度，根据实际的位移和tick时间来计算速度。如果是在地面上行走，还要将速度的Z轴分量置为0

至此PhysWalking的全部过程结束

### △△△△△△ PhysWalking △△△△△△

### ▽▽▽▽▽▽ PhysFalling ▽▽▽▽▽▽

首先计算水平面上的加速度，GetFallingLateralAcceleration，需要计算空气阻力的影响，GetAirControl会计算出一个0到1的数值，并乘以Acceleration。如果没有AirControl，FallAcceleration将会为0。

然后计算VelocityNoAirControl和Velocity，区别就是在CalcVelocity需不需要计算Acceleration

然后根据Gravity计算NewFallVelocity，需要注意不能超过PhysicsVolume物理空间的最大速度TerminalVelocity。

位移向量Adjusted是OldVelocity和Velocity的平均值。然后调用SafeMoveUpdatedComponent

根据得到的Hit调用IsValidLandingSpot，只保留下半球的hit，然后调用FindFloor，得到FloorResult

如果没有找到合理的Floor，尝试HandleImpact

如果支持AirControl，根据HitResult来LimitAirControl，如果HitResult是block而且坡度陡峭，地面不可落，则将FallAcceleration投射到垂直平面上，避免沿着坡上升。

如果HitResult存在StartPenetrating，并且坡朝下，则将FallAcceleration设置0

把计算出来的调整位移Adjusted变成沿着斜面，ComputeSlideVector，计算出Delta

再次SafeMoveUpdatedComponent，如果hit结果block，撞到了第二面墙，验证这个面是否可落。如果不可落，HandleImpact，如果支持AirControl，这次Move将不再考虑。直接计算沿斜面运动的Delta。然后更加两次Move返回的Hit调用TwoWallAdjust

AirControl只在远离第一个面时生效。

根据Delta重新计算Velocity，然后SafeMoveUpdatedComponent，如果没有移动，则尝试沿着面位移SideDelta。

如果两个面成沟状或者是可以落得地面，又或者再也无法移动了，则默认是踩到地面了。



### △△△△△△ PhysFalling △△△△△△