**向心力**

*TArray*<*UPrimitiveComponent*\*> OverlappingComps; 创建数组将范围内物体存入

OutSphereCompont->*GetOverlappingComponents*(OverlappingComps);

for (*int32* i = 0; i < OverlappingComps.*Num*(); i++)

{

*UPrimitiveComponent*\* PrimComp = OverlappingComps[i]; 逐一赋值

if (PrimComp && PrimComp->*IsSimulatingPhysics*())

{

const float SphereRadius = OutSphereCompont->*GetScaledSphereRadius*();

const float ForceStrength = -8000; 负值使它向原点拉而不是推开

PrimComp->*AddRadialForce*(*GetActorLocation*(), SphereRadius, ForceStrength, *ERadialImpulseFalloff*::*RIF\_Constant*, true);

}

}

**碰撞设置**

MeshComp->*SetCollisionEnabled*(*ECollisionEnabled*::*NoCollision*); 无碰撞

OverlapComp->*SetCollisionEnabled*(*ECollisionEnabled*::*QueryOnly*); 仅查询 包括追踪轨迹线或重叠等

OverlapComp->*SetCollisionResponseToAllChannels*(*ECR\_Ignore*); 全为忽略

OverlapComp->*SetCollisionResponseToChannel*(*ECC\_Pawn*, *ECR\_Overlap*); 只对Pawn 设置重叠

**碰撞方法**

1

.h

*UFUNCTION*() 绑定函数时，需将其标记，以便让虚幻后端明白该函函数的含义及如何将其与事件绑定

void HandLeOverlap( *UPrimitiveComponent*\* OverlappedComponent, *AActor*\* OtherActor, *UPrimitiveComponent*\* OtherComp, *int32* OtherBodyIndex,

bool bFromSweep, const *FHitResult* & SweepResult);

cpp

….

OverlapComp->*OnComponentBeginOverlap*.*AddDynamic*(this, &AClearanceActor::HandLeOverlap); 绑定

….

void AClearanceActor::HandLeOverlap(*UPrimitiveComponent* \* OverlappedComponent, *AActor* \* OtherActor, *UPrimitiveComponent* \* OtherComp,

*int32* OtherBodyIndex, bool bFromSweep, const *FHitResult* & SweepResult)

{

*UE\_LOG*(LogTemp, *Log*, *TEXT*("overloap")); 输出到日志

}

2

H

virtual void NotifyActorBeginOverlap(*AActor*\* OtherActor)override; 重载

cpp

void AMyActor::NotifyActorBeginOverlap(*AActor*\* OtherActor)

{

Super::NotifyActorBeginOverlap(OtherActor);

PlayEffects();

Abiu5Character\* character = *Cast*<Abiu5Character>(OtherActor); 类型转换

if (character)

{

character->bIsCarryingObjective = true;

*Destroy*();

}

}