GAS大套餐

选取目标函数

## GAS大套餐

创建04GroundBlast

创建GA、GE、

添加GA的Tag

回到BP\_Player添加技能

他的动作有两个：选取动作、释放动作。

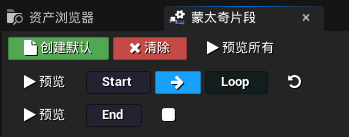
创建MA\_Select/MA\_

MA\_Select中

添加动画资产Ability\_AttackWolves\_StartTarget\_R

添加蒙太奇片段Start、Loop、End

设置蒙太奇片段



## 编写选取目标函数

创建C++类，父类为GameplayAbilityTargetActor

命名为GroundSelectTargetActor

在头文件中重载两个函数  
// 这个函数类似EventBeginPlay

virtual void StartTargeting(UGameplayAbility\* Ability) override;

virtual void ConfirmTargetingAndContinue() override;

// 用于选取的圆的半径有多大

// meta表示SpawnActor的时候这个参数会直接暴露在SpawnActor函数中（生成时公开）

UPROPERTY(EditAnywhere,BlueprintReadWrite,meta=(ExposeOnSpawn=true),Category="GroundSelect")

float SelectRadius;

// 这个函数是需要自定义自己实现，在父类中没有这个函数

// 函数是得到玩家看向地板的点的位置

UFUNCTION(BlueprintCallable,Category="GroundSelect")

bool GetPlayerLookAtPoint(FVector& Out\_LookPoint);

实现这些函数

头文件中加上头文件#include "Abilities/GameplayAbility.h""

void AGroundSelectTargetActor::StartTargeting(UGameplayAbility\* Ability)

{

    Super::StartTargeting(Ability);

    PrimaryPC=Cast<APlayerController>(Ability->GetOwningActorFromActorInfo()->GetInstigatorController());

}

bool AGroundSelectTargetActor::GetPlayerLookAtPoint(FVector& Out\_LookPoint)

{

    FVector ViewLoc;

    FRotator ViewRot;

    PrimaryPC->GetPlayerViewPoint(ViewLoc, ViewRot);

    // 射线检测返回的数值类型

    FHitResult HitResult;

    // 射线检测会检测到底是哪一个射线参数需要被检测到

    FCollisionQueryParams QueryParams;

    // 复杂检测，意味着高速移动状态下依然会检测到一些细小的物体

    // 当然会加大计算负担

    QueryParams.bTraceComplex = true;

    APawn\* SelfPawn=PrimaryPC->GetPawn();

    if (SelfPawn)

    {

        QueryParams.AddIgnoredActor(SelfPawn);

    }

    // 实现射线

    bool TraceResult = GetWorld()->LineTraceSingleByChannel(

        HitResult,

        ViewLoc,

        ViewLoc+ViewRot.Vector()\*5000.0f,

        ECollisionChannel::ECC\_Visibility,

        QueryParams);

    if(TraceResult)

    {

        Out\_LookPoint=HitResult.ImpactPoint;

    }

    return TraceResult;

}

void AGroundSelectTargetActor::ConfirmTargetingAndContinue()

{

    // Super::ConfirmTargetingAndContinue();

    FVector LookPoint;

    GetPlayerLookAtPoint(LookPoint);

    TArray<FOverlapResult> OverlapResults;

    TArray<TWeakObjectPtr<AActor>> OverlapActors;

    // 第一次选择是GetPlayerLookAtPoint函数，选择的是地面

    // 这里是第二次选择，选择的是敌人

    FCollisionQueryParams QueryParams;

    // 可以不写，默认为False

    QueryParams.bTraceComplex = false;

    // 物理材质

    // 一般用在走路时走在地面上和水坑上发出的声音不一样

    QueryParams.bReturnPhysicalMaterial = false;

    APawn\* SelfPawn = PrimaryPC->GetPawn();

    if (SelfPawn)

    {

        QueryParams.AddIgnoredActor(SelfPawn);

    }

    // 射线检测，可以检测到多个对象

    bool QueryResult=GetWorld()->OverlapMultiByObjectType(

        OverlapResults,

        LookPoint,

        FQuat::Identity,

        FCollisionObjectQueryParams(ECollisionChannel::ECC\_Pawn),

        FCollisionShape::MakeSphere(SelectRadius),

        QueryParams);

    if(QueryResult)

    {

        for(int i=0;i<OverlapResults.Num();i++)

        {

            APawn\* Enemy=Cast<APawn>(OverlapResults[i].GetActor());

            if(Enemy && !OverlapActors.Contains(Enemy))

            {

                OverlapActors.Add(Enemy);

            }

        }

    }

    FGameplayAbilityTargetDataHandle TargetDataHandle;

    FGameplayAbilityTargetData\_LocationInfo\* CenterLoc = new FGameplayAbilityTargetData\_LocationInfo();

    CenterLoc->TargetLocation.LiteralTransform = FTransform(LookPoint);

    // 定义负载的类型

    CenterLoc->TargetLocation.LocationType =EGameplayAbilityTargetingLocationType::LiteralTransform;

    // 负载进去

    // 0号负载，把看向的地方负载进去

    TargetDataHandle.Add(CenterLoc);

    // 1号负载，负载抓住了哪些敌人

    if(OverlapActors.Num()>0)

    {

        FGameplayAbilityTargetData\_ActorArray\* ActorArray = new FGameplayAbilityTargetData\_ActorArray();

        ActorArray->SetActors(OverlapActors);

        // 1号负载

        TargetDataHandle.Add(ActorArray);

    }

}

文档终止在这里，无法往下进行。问题在于ConfirmTargetingAndContinue函数

我们在函数中创建好了

TArray<FOverlapResult> OverlapResults;

使用他的时候

APawn\* Enemy=Cast<APawn>(OverlapResults[i].GetActor());

这里会报错

