1. 类创建特效

.h

Class UParticleSystem;

UPROPERTY(EditDefaultsOnly,Category = "Effects") //公开声明

UParticleSystem\* PickupFX;

.cpp

UGameplayStatics::SpawnEmitterAtLocation(this, PickupFX, GetActorLocation());

在原处生成 特效 获取物体坐标

1. 创建Actor类

.h

UPROPERTY(VisibleAnywhere,Category=”Components”)

UStaticMeshComponent\* MeshComp;

UPROPERTY(VisibleAnywhere,Category=”Components”)

USphereComponent\* SphereComp

.cpp

MeshComp = CreateDefaultSubobject<UStaticMeshComponent>(TEXT(“MeshComp”));

//创建名为“MeshComp”的静态网格体

RootComponent = MeshComp;

//MeshComp 设为根组件

SphereComp = CreateDefaultSuboboject<USphereComponent>(TEXT(“SphereComp”));

//创建名为“SphereComp“的静态网格体

SphereComp->SetupAttachment(MeshComp);

//SphereComp 设为MeshComp的子组件

1. 设置碰撞 碰撞响应
2. .h文件

UFUNCTION()

void HandleOverlap(UPrimitiveComponent\* OverlappedComponent, AActor\* OtherActor, UPrimitiveComponent\* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult & SweepResult);

.cpp文件

OverlapComp->OnComponentBeginOverlap.AddDynamic(this,

&AFPSExtractionZone::HandleOverlap);

void AFPSExtractionZone::HandleOverlap(UPrimitiveComponent\* OverlappedComponent, AActor\* OtherActor, UPrimitiveComponent\* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult & SweepResult)

1. .h文件

virtual void NotifyActorBeginOverlap(AActor\* OtherActor) override;

.cpp文件

SphereComp->SetCollisionEnabled(ECollisionEnabled::QueryOnly);

启用碰撞 仅查询

SphereComp->SetCollisionResponseToAllChannels(ECR\_Ignore);

开启碰撞响应 无视

SphereComp->SetCollisionResponseToChannel(ECC\_Pawn, ECR\_Overlap);

设置碰撞响应通道 仅Pawn类 重叠

void AFPSObjectiveActor::NotifyActorBeginOverlap(AActor\* OtherActor)

{

Super::NotifyActorBeginOverlap(OtherActor);

}

1. 转化类型

AFPSCjaracter\* MyCharacter = Cast<AFPSCharacter>(OtherActor);

1. 设置半径

InnerSphereComponet->SetSphereRadius(100);

1. 创建数组

TArray<UPrimitiveComponent\*> OverlappingComps;

数组 原始组件 组名

1. 施加作用力

PrimComp->AddRadialForce(GetActorLocation(),SphereRadius,ForceStength,

半径 力的大小

ERadialImpulseFalloff::RIF\_Constant,ture)

恒速 加速变化

1. 日志

UE\_LOG(LogTemp, Log, TEXT("Overlapped with extraction zone!"));

九、设置响应范围

OverlapComp->SetBoxExtent(Fvector(200.0f))

x,y,z=200个单位

1. 显示隐藏项

OverLapComp->SetHiddenInGame(false);

1. 贴花组件

.h

UPROPERTY(VisibleAnywhere, Category = "Components")

UDecalComponent\* DacalComp;

.cpp

DecalComp = CreateDefaultSubobject<UDecalComponent>()

创建默认子对象 贴花组件

十一、禁用输入

InstigatorPawn->DisableInput(nullptr)

专门禁用对玩家控制器控制着的Pawn 即游戏角色的输入

十二、公开声明函数

.h

UFUNCTION(BlueprintImplementableEvent,Category="GameMode")

设置为蓝图可实现事件 类目 游戏模式

void OnMissionCompleted(APawn\* InstigatorPawn);

.cpp

OnMissionCompleted(InstigatorPawn);

十三、获取游戏模式

GetWorld()->GetAuthGameMode()

客户端多人设置时无效

十四、声音

.h

UPROPERTY(EditDefaultsOnly,Category="Sounds")

USoundBase\* ObjectiveMissingSound;

音基

.cpp

UGameplayStatics::PlaySound2D(this, ObjectiveMissingSound);

十五、设置视图

.h

UPROPERTY(EditDefaultsOnly, Category = "Spectating")

TSubclassOf<AActor> SpectatingViewpointClass;

.cpp

TArray<AActor\*> ReturnnedActors;

UGameplayStatics::GetAllActorsOfClass(this, SpectatingViewpointClass,

ReturnnedActors);

if (ReturnnedActors.Num()>0)

{

AActor\* NewViewTarget = ReturnnedActors[0];

APlayerController\* PC =

Cast<APlayerController>(InstigatorPawn->GetController());

if (PC)

{

PC->SetViewTargetWithBlend(NewViewTarget, 0.5f, EViewTargetBlendFunction::VTBlend\_Cubic);

}

}

十六、创建ACTOR

AActor\* NewViewTarget;

十七、创建数组(actor)

TArray<AActor\*> Actors;

十八、人体感应

.h

UPROPERTY(VisibleAnywhere,Category="Components")

class UPawnSensingComponent\* PawnSensingComp;

UFUNCTION()

void OnPawnSeen(APawn\* Seenpawn); //设置看到函数

.cpp

PawnSensingComp = CreateDefaultSubobject<UPawnSensingComponent>(TEXT("PawnSensingComp"));

//人体感应组件

PawnSensingComp->OnSeePawn.AddDynamic(this, &AFPSAIGuard::OnPawnSeen);

void AFPSAIGuard::OnPawnSeen(APawn\* Seenpawn)

{

if (Seenpawn == nullptr)

{

return;

}

DrawDebugSphere(GetWorld(), Seenpawn->GetActorLocation(), 32.0f, 12, FColor::Yellow, false, 10.0f);//看到后产生小球

}

十九、添加AI组件

using UnrealBuildTool;

public class FPSGame : ModuleRules

{

public FPSGame(ReadOnlyTargetRules Target) : base(Target)

{

PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Engine", "InputCore" ,"AIModule"});

}

}

二十、弹射

.h

UFUNCTION()

void OverlapLaunchPad(UPrimitiveComponent\* OverlappedComponent, AActor\* OtherActor, UPrimitiveComponent\* OtherComp, int32 OtherBodyIndex, bool bFromSweep, const FHitResult& SweepResult)

UPROPERTY(EditInstanceOnly, Category = "LaunchPad")

float LaunchStrength; //设置力度

UPROPERTY(EditInstanceOnly, Category = "LaunchPad")

float LaunchPitchAngle;//设置角度

.cpp

OverlapComp->OnComponentBeginOverlap.AddDynamic(this,&AFPSLaunchPad::OverlapLaunchPad);//重叠触发函数

LaunchStrength = 1500;

LaunchPitchAngle = 35.0f;

FRotator LaunchDirection = GetActorRotation();

LaunchDirection.Pitch += LaunchPitchAngle;

FVector LaunchVelocity = LaunchDirection.Vector() \* LaunchStrength;

ACharacter\* OtherCharacter = Cast<ACharacter>(OtherActor);

//其他物体重叠转换成Character

if (OtherCharacter)

{

OtherCharacter->LaunchCharacter(LaunchVelocity, true, true);

}

else if (OtherComp && OtherComp->IsSimulatingPhysics())

{

OtherComp->AddImpulse(LaunchVelocity, NAME\_None, true);

}

二十一、 定时器

GetWorldTimerManager().ClearTimer (TimerHandle\_LifeSpanExpired);

清除

GetWorldTimerManager().SetTimer(TimerHandle\_LifeSpanExpired, this, &AFPSAIGuard::ResetOrientation,3.0f);

二十二、移动函数

UNavigationSystem::SimpleMoveToActor(GetController(), CurrentPatrolPoint);

二十三、创建多人游戏

1.发射物

.cpp

SetReplicates(true);

SetReplicateMovement(true);

void AFPSProjectile::OnHit

{

if (Role == ROLE\_Authority) //角色被授权

{

MakeNoise(1.0f, Instigator);

Destroy();

}

}

2.角色

.h

void Fire();

UFUNCTION(Server, Reliable, WithValidation)

void ServerFire();

.cpp

void AFPSCharacter::Fire()

{

ServerFire();

}

void AFPSCharacter::ServerFire\_Implementation()

{

if (ProjectileClass)

{

FVector MuzzleLocation = GunMeshComponent->GetSocketLocation("Muzzle");

FRotator MuzzleRotation = GunMeshComponent->GetSocketRotation("Muzzle");

//Set Spawn Collision Handling Override

FActorSpawnParameters ActorSpawnParams;

ActorSpawnParams.SpawnCollisionHandlingOverride = ESpawnActorCollisionHandlingMethod::AdjustIfPossibleButDontSpawnIfColliding;

ActorSpawnParams.Instigator = this;

// spawn the projectile at the muzzle

GetWorld()->SpawnActor<AFPSProjectile>(ProjectileClass, MuzzleLocation, MuzzleRotation, ActorSpawnParams);

}

}

bool AFPSCharacter::ServerFire\_Validate()

{

return true;

}

二十四、同步射击角度

.h

virtual void Tick(float DeltaTime) override;

.cpp

void AFPSCharacter::Tick(float DeltaTime)

{

Super::Tick(DeltaTime);

if (!IsLocallyControlled())

{

FRotator NewRot = CameraComponent->RelativeRotation;

NewRot.Pitch = RemoteViewPitch \* 360.0f / 255.0f; //系统压缩 进行解压

CameraComponent->SetRelativeRotation(NewRot);

}

}