ChapterOne

Alt+G 跳转

Alt+O 跳转头文件

Destroy 毁坏

.h

class UparticleSystem; //向前声明

UPROPERTY(EditDefaultsOnly, Category = "FX")

（公开） （仅编辑默认） （类目）

UParticleSystem\* ExplosionEffect;

（粒子系统）

.cpp

//生成粒子系统

UGameplayStatics::SpawnEmitterAtLocation(GetWorld(),ExplosionEffect,GetActorLocation());

（在原处生成发射器） （爆炸效应）

ChapterTwo

Shift+Alt+O查找

QueryOnly 只发出询问

BlueprintReadOnly 仅阅读蓝图

EditAnyWhere: 此成员变量在蓝图编辑器中和关卡细节面板中都会被暴露出来

EditDefaultsOnly: 此成员变量只会在蓝图编辑器中被暴露出来

EditInstanceOnly: 此成员变量只会在关卡细节面板中被暴露出来

Virtual: 在某基类中声明为 virtual 并在一个或多个派生类中被重新定义的成员函数，用法格式为：virtual 函数返回类型 函数名（参数表） {函数体}；实现多态性，通过指向派生类的基类指针或引用，访问派生类中同名覆盖成员函数。

.h

公开碰撞组建在蓝图中显示,定义网格体

eg:

UPROPERTY(VisibleAnywhere, Category = "Components")

（随处可见）

UStaticMeshComponent\* MeshComp;

UPROPERTY(VisibleAnywhere, Category = "Components")

USphereComponent\* SphereComp;

UPROPERTY(EditDefaultsOnly,Category="Effects")

（仅在蓝图编辑器中显示） （特效）

UParticleSystem\* PickupFX;

函数重载重写

eg:

virtual void NotifyActorBeginOverlap(AActor\* OtherActor) override;

（虚函数）重写 （在发生重叠时通知Actor） （覆盖）

.cpp

MeshComp = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("MeshComp"));

(创建默认子对象) （实例化类型） （名称）

MeshComp->SetCollisionEnabled(ECollisionEnabled::NoCollision);

（设置碰撞启用）

RootComponent = MeshComp;

（设置根组件）

SphereComp->SetCollisionResponseToAllChannels(ECR\_Ignore);

（设置碰撞为响应所有通道） (忽略)

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

(设置碰撞为响应指定通道) (人形体) (忽略)

SphereComp->SetupAttachment(MeshComp);

（设置附加）

Void （文件名）::自定义事件名（）

{

功能

}

eg：

void AFBSObjectiveActor::PlayEffects()

{

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

创建插槽(名字)

}

void AFBSObjectiveActor::NotifyActorBeginOverlap(AActor\* OtherActor)

{

Super::NotifyActorBeginOverlap(OtherActor);

(特级的)

PlayEffects();

类型转换

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

if (MyCharacter)

{

MyCharacter->bIsCarryingObjective = true;

Destroy();

}

}

BlackHole

AddDynamic 添加动态

Generate Overlap(生成重叠事件)

//称每一帧

void AFPSBlackHold::Tick(float DeltaTime)

{

Super::Tick(DeltaTime);

//找到所有可能发生碰撞的重叠组件，并可能进行物理模拟

TArray<UPrimitiveComponent\*>OverlappingComps;//ZXprimitive元组件

//物理作用力的对象

OuterSphereComponent->GetOverlappingComponents(OverlappingComps);

//循环

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

{

UPrimitiveComponent\* PrimComp = OverlappingComps[i];

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

{

//我们正在寻找的组件！它需要模拟才能施加外力。

const float SphereRadius = OuterSphereComponent->GetScaledSphereRadius();

//球体半径

const float ForceStrength = -2000;//负的价值使它向原点而不是推开

//作用力强度

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

//施加作用力（径向力），返回元组件

}

}

}

ChapterThree

EditAnyWhere: 此成员变量在蓝图编辑器中和关卡细节面板中都会被暴露出来

EditDefaultsOnly: 此成员变量只会在蓝图编辑器中被暴露出来

EditInstanceOnly: 此成员变量只会在关卡细节面板中被暴露出来

Intellisense智能编码

Close All But This 关闭除此之外的所用文件

DeveLoper Tools开发者工具

Output Log输出日志

.cpp

//实例化 Ctrl+F 搜索“Size”尺寸

DecalComp = CreateDefaultSubobject<UDecalComponent>(TEXT("DecalComp"));

DecalComp->DecalSize = FVector(200.0f,200.0f,200.0f);

DecalComp->SetupAttachment(RootComponent);

.h

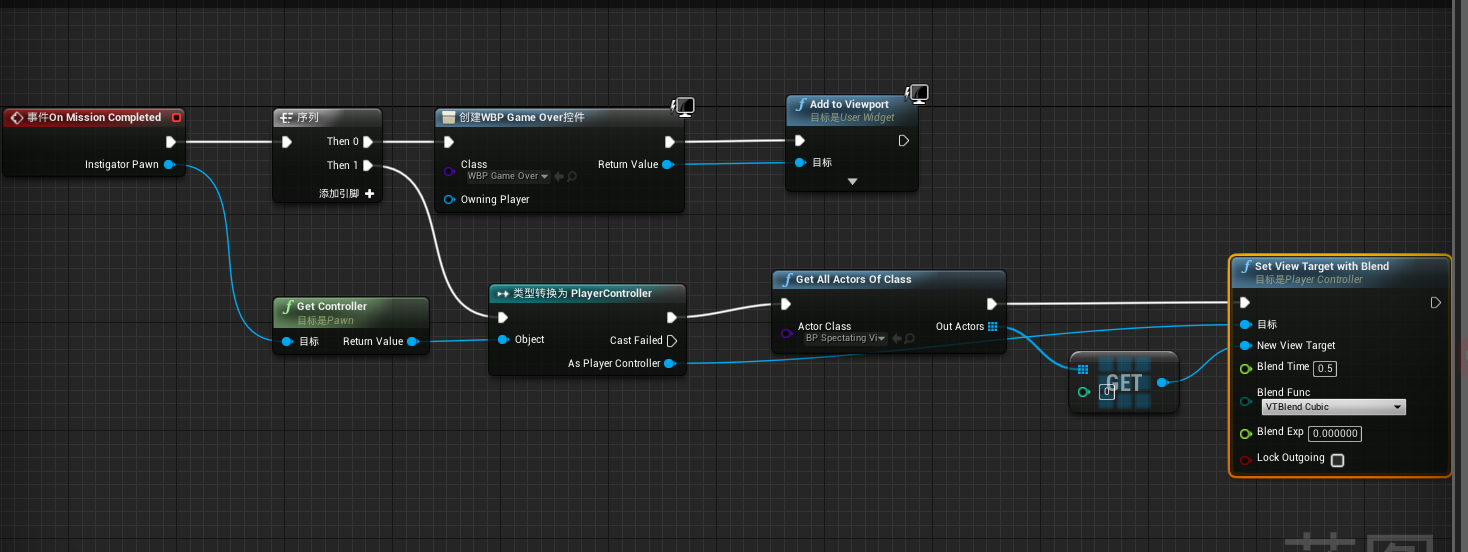
UPROPERTY(VisibleAnywhere, Category = "Components")

UDecalComponent\* DecalComp;

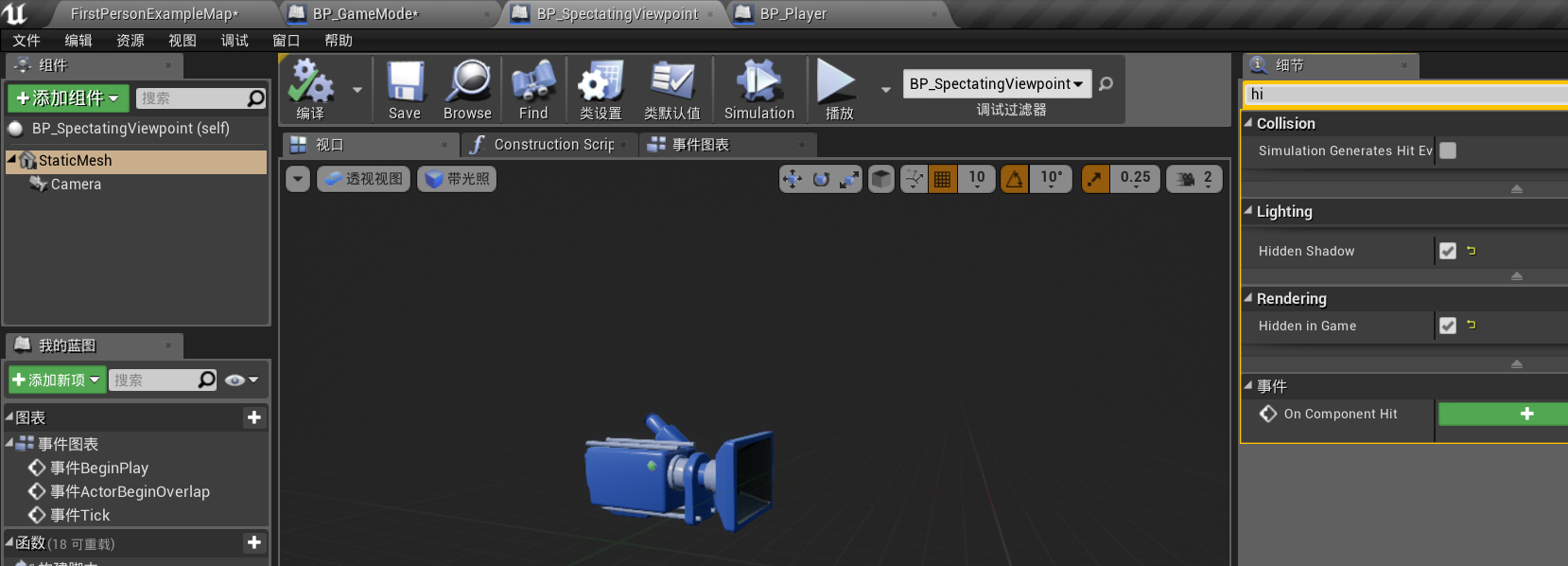
//ZC贴花组件

蓝图第一人称转第三人称视角编辑：

打开BP\_GameMode



新建蓝图添加StaticMesh(静态网格体) 固定在根目录下, 添加Camera（相机）,设置Hidden in Game保证它再游戏中隐藏显示。



.h

UPROPERTY(VisibleAnywhere,Category="Components")

ChapterFour

Use Animation Asset(使用动画资源)

hearing Threshold 听阈；听阀；听力障碍；听觉阈限

LOSHearing Threshold 视距 （指朝发生源望去，其间无遮挡的直线距离）

Peripheral Vision Angle 视野范围

Sight Radius 视野半径

Delegate代理

nullptr 空指针;指针

Create Implementation(建立执行)

Current Project(当前项目)

Expose On Spawn(在生成时显示)

Instance Editable(实力可编辑)

Refresh Node(更新节点）

Guard State(守卫状态)

Draw at Desired Size(自定义尺寸)

OnStateChanged(状态更新时)

CHALLENGE CODE 挑战模式

Nav Mesh Bound Volume(寻路网格体包围体)（p显示）

Select AllMatching Class(选中所有匹配类)

UpawnSensingComponent\* PawnSensingComp;

(人形体感应组建)

PawnSensingComponent是涉及AI编程时候的常用类，实现了很多强大的功能，对于AI感知帮助很大。

1.AI的视觉，是一个锥形感知体。

2.AI的听觉，主要与距离和声音大小有关，与遮挡物也有关。

1.CouldSeePawn仅仅简单地判断你给出的Pawn是否在锥形内。（这个考虑了锥形的覆盖，但是没有考虑遮蔽）

2.HasLineOfSightTo仅仅简单地判断你给出的Pawn是否内被这个PawnSensingComponent看见（这个又没有考虑锥形的覆盖，只考虑了遮挡)

.cpp

视觉绑定，添加动态函数

PawnSensingComp->OnSeePawn(Alt+G).AddDynamic(this, &AEPSAIGuard::OnPawnSeen);

{

if(SeenPawn==nullptr)

{

return;

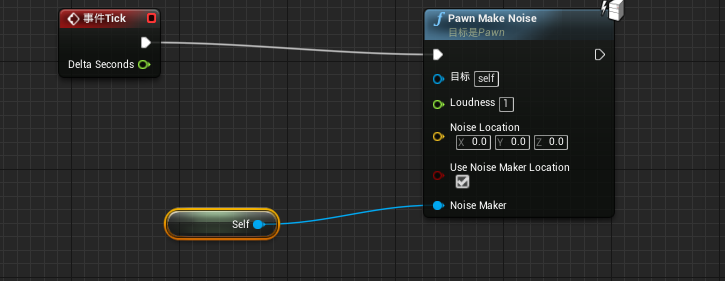
}

DrawDebugSphere(GetWorld(),SeenPawn->GetActorLocation(),32.0f,12,FColor::Red,false, 10.0f);

//设置半径，分段数 ，颜色 , PersistentLine(持久谱线)，持续时长

}

制造声音



LookAtRotation(朝向旋转)

FRotationMatrix(旋转矩阵)

Direction(方向)

GetWorldTimerManager(获取世界场景时钟管理器)

TimeHandle(定时器句柄)

ResetOrientation(重置方向)

ClearTimer(清除定时器)

Pitch上下转动

Yaw左右转动

Roll倾斜转动

OriginalRotation初始方位

.cpp

//在巡逻时停止移动

AController\* Controller = GetController();

if (Controller)

{

Controller->StopMovement();

}

}

//转动调整

NewLookAt.Pitch = 0.0f;

NewLookAt.Roll = 0.0f;

SetActorRotation(NewLookAt);

//ZX用该句柄停止原先的定时器，并重设一个新的定时器，这样就不会同时运行多个定时器了

GetWorldTimerManager().ClearTimer(TimerHandle\_ResetOrientation);

//ZX设置定时器

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

// 称每一帧

void AFPSAIGuard::Tick(float DeltaTime)

{

Super::Tick(DeltaTime);

// 巡逻检查目标

if (CurrentPatrolPoint)

{

FVector Delta = GetActorLocation() - CurrentPatrolPoint->GetActorLocation();

float DistanceToGoal = Delta.Size();

// 距离小于50个单位长度，则寻找下一个巡逻点,ZX实现守卫往返两点之间

if (DistanceToGoal < 100)

{

MoveToNextPatrolPoint();

}

}

}

void AFPSAIGuard::MoveToNextPatrolPoint()

{

//ZX分配下一个巡逻点

// ZX检查守卫是否不在巡逻点上，当游戏开始时，条件肯定为真，或用于检查当前的巡逻点是否为第二个巡逻点

if (CurrentPatrolPoint == nullptr || CurrentPatrolPoint == SecondPatrolPoint)

{

//ZX为真靠近第一个巡逻点

CurrentPatrolPoint = FirstPatrolPoint;

}

else

{

//ZX为假靠近第二个巡逻点

CurrentPatrolPoint = SecondPatrolPoint;

}

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

//ZX当前巡逻点

}

UENUM(BlueprintType)

enum class EAIState : uint8

{

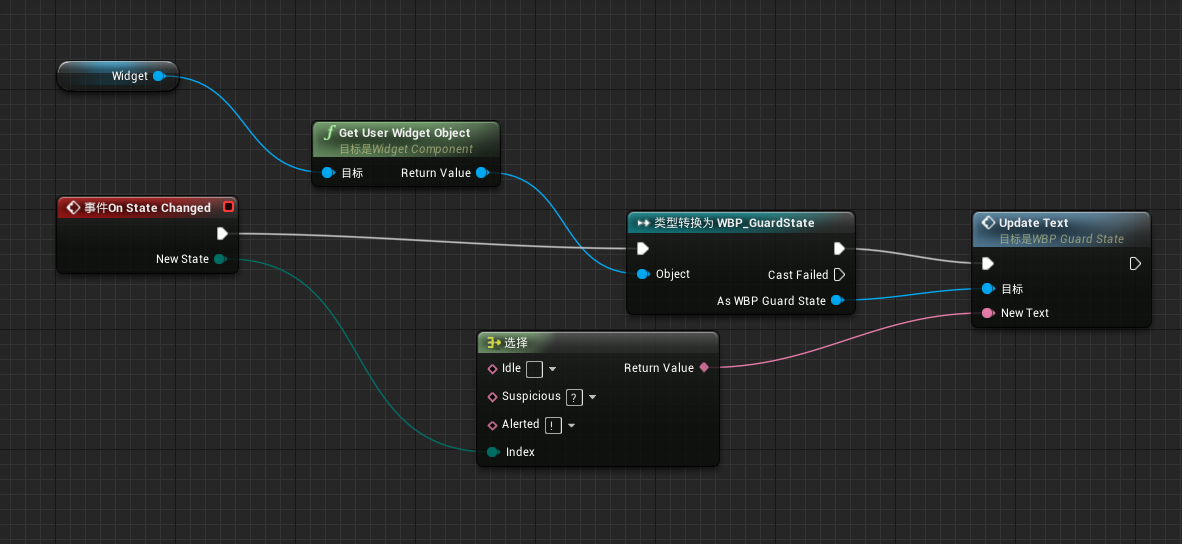
Idle,//ZX闲散

Suspicious,//ZX怀疑

Alerted//ZX警觉

};

状态设置



protected:

//ZX挑战模式

UPROPERTY(EditInstanceOnly, Category = "AI")

bool bPatrol;

//ZX两个巡逻点的第一个巡逻点

UPROPERTY(EditInstanceOnly, Category = "AI", meta = (EditCondition = "bPatrol"))

AActor\* FirstPatrolPoint;

//ZX两个巡逻点的第二个巡逻点

UPROPERTY(EditInstanceOnly, Category = "AI", meta = (EditCondition = "bPatrol"))

AActor\* SecondPatrolPoint;

//ZX目前的观点是，演员要么移动要么站在

AActor\* CurrentPatrolPoint;

//ZX移动至下一个巡逻点

void MoveToNextPatrolPoint();