移动玩家到指定位置

在指定位置生成例子特效  
 *// We look for the location in the world where the player has pressed the input* FVector HitLocation = FVector::ZeroVector;  
 FHitResult Hit;  
 GetHitResultUnderCursor(ECC\_Visibility, true, Hit);  
 HitLocation = Hit.Location;  
  
 *// We move there and spawn some particles* UAIBlueprintHelperLibrary::SimpleMoveToLocation(this, HitLocation);  
 UNiagaraFunctionLibrary::SpawnSystemAtLocation(this, FXCursor, HitLocation, FRotator::ZeroRotator, FVector(1.f, 1.f, 1.f), true, true, ENCPoolMethod::None, true);

绑定输入事件 action

InputComponent->BindAction("SetDestination", IE\_Pressed, this, &ATopDownPlayerController::OnSetDestinationPressed);  
InputComponent->BindAction("SetDestination", IE\_Released, this, &ATopDownPlayerController::OnSetDestinationReleased);

绑定输入事件 touch

*// support touch devices*InputComponent->BindTouch(EInputEvent::IE\_Pressed, this, &ATopDownPlayerController::OnTouchPressed);  
InputComponent->BindTouch(EInputEvent::IE\_Released, this, &ATopDownPlayerController::OnTouchReleased);

*// Look for the touch location*FVector HitLocation = FVector::ZeroVector;  
FHitResult Hit;  
if(bIsTouch)  
{

获取手指点击信息  
 GetHitResultUnderFinger(ETouchIndex::Touch1, ECC\_Visibility, true, Hit);  
}  
else  
{

获取鼠标点击信息  
 GetHitResultUnderCursor(ECC\_Visibility, true, Hit);  
}  
HitLocation = Hit.Location;

指挥兵去指定方向

*// Direct the Pawn towards that location*APawn\* const MyPawn = GetPawn();  
if(MyPawn)  
{  
 FVector WorldDirection = (HitLocation - MyPawn->GetActorLocation()).GetSafeNormal();  
 MyPawn->AddMovementInput(WorldDirection, 1.f, false);  
}

是否显示鼠标 鼠标样式

bShowMouseCursor = true;  
DefaultMouseCursor = EMouseCursor::Default;

默认构造 从content文件夹

*// set default pawn class to our Blueprinted character*static ConstructorHelpers::FClassFinder<APawn> PlayerPawnBPClass(TEXT("/Game/TopDown/Blueprints/BP\_TopDownCharacter"));  
if (PlayerPawnBPClass.Class != nullptr)  
{  
 DefaultPawnClass = PlayerPawnBPClass.Class;  
}  
  
*// set default controller to our Blueprinted controller*static ConstructorHelpers::FClassFinder<APlayerController> PlayerControllerBPClass(TEXT("/Game/TopDown/Blueprints/BP\_TopDownPlayerController"));  
if(PlayerControllerBPClass.Class != NULL)  
{  
 PlayerControllerClass = PlayerControllerBPClass.Class;  
}

设置胶囊体size

*// Set size for player capsule*GetCapsuleComponent()->InitCapsuleSize(42.f, 96.0f);

是否控制旋转

*// Don't rotate character to camera direction*bUseControllerRotationPitch = false;  
bUseControllerRotationYaw = false;  
bUseControllerRotationRoll = false;

配置移动组件

*// Configure character movement*GetCharacterMovement()->bOrientRotationToMovement = true; *// Rotate character to moving direction*GetCharacterMovement()->RotationRate = FRotator(0.f, 640.f, 0.f);  
GetCharacterMovement()->bConstrainToPlane = true;  
GetCharacterMovement()->bSnapToPlaneAtStart = true;

创建弹簧臂和相机组件

*// Create a camera boom...*CameraBoom = CreateDefaultSubobject<USpringArmComponent>(TEXT("CameraBoom"));  
CameraBoom->SetupAttachment(RootComponent);  
CameraBoom->SetUsingAbsoluteRotation(true); *// Don't want arm to rotate when character does*CameraBoom->TargetArmLength = 800.f;  
CameraBoom->SetRelativeRotation(FRotator(-60.f, 0.f, 0.f));  
CameraBoom->bDoCollisionTest = false; *// Don't want to pull camera in when it collides with level  
  
// Create a camera...*TopDownCameraComponent = CreateDefaultSubobject<UCameraComponent>(TEXT("TopDownCamera"));  
TopDownCameraComponent->SetupAttachment(CameraBoom, USpringArmComponent::SocketName);  
TopDownCameraComponent->bUsePawnControlRotation = false; *// Camera does not rotate relative to arm*

Actor激活tick

*// Activate ticking in order to update the cursor every frame.*PrimaryActorTick.bCanEverTick = true;  
PrimaryActorTick.bStartWithTickEnabled = true;

编译依赖

using UnrealBuildTool;  
  
public class ThirdPerson : ModuleRules  
{  
 public ThirdPerson(ReadOnlyTargetRules Target) : base(Target)  
 {  
 PCHUsage = PCHUsageMode.**UseExplicitOrSharedPCHs**;  
  
 PublicDependencyModuleNames.AddRange(new string[] { "Core", "CoreUObject", "Engine", "InputCore", "HeadMountedDisplay" });  
 }  
}

yaw输入

*// calculate delta for this frame from the rate information*AddControllerYawInput(Rate \* TurnRateGamepad \* GetWorld()->GetDeltaSeconds());

Pitch输入

*// calculate delta for this frame from the rate information*AddControllerPitchInput(Rate \* TurnRateGamepad \* GetWorld()->GetDeltaSeconds());

往前移动 X

*// find out which way is forward*const FRotator Rotation = Controller->GetControlRotation();  
const FRotator YawRotation(0, Rotation.Yaw, 0);  
  
*// get forward vector*const FVector Direction = FRotationMatrix(YawRotation).GetUnitAxis(EAxis::X);  
AddMovementInput(Direction, Value);

往右移动

*// find out which way is right*const FRotator Rotation = Controller->GetControlRotation();  
const FRotator YawRotation(0, Rotation.Yaw, 0);  
  
*// get right vector*const FVector Direction = FRotationMatrix(YawRotation).GetUnitAxis(EAxis::Y);  
*// add movement in that direction*AddMovementInput(Direction, Value);

将当前鼠标2D位置转换为世界空间3D位置和方向

APlayerController\* PC = Cast<APlayerController>(GetController())

FVector Start, Dir, End;  
PC->DeprojectMousePositionToWorld(Start, Dir);  
End = Start + (Dir \* 8000.0f);  
TraceForBlock(Start, End, true);

射线检测start---end 是否有接触

FHitResult HitResult;  
GetWorld()->LineTraceSingleByChannel(HitResult, Start, End, ECC\_Visibility);  
if (bDrawDebugHelpers)  
{  
 DrawDebugLine(GetWorld(), Start, HitResult.Location, FColor::Red);  
 DrawDebugSolidBox(GetWorld(), HitResult.Location, FVector(20.0f), FColor::Blue);  
}  
if (HitResult.GetHitObjectHandle().IsValid())  
{  
 if (HitResult.GetHitObjectHandle() != CurrentBlockFocus)  
 {  
 if (CurrentBlockFocus)  
 {  
 CurrentBlockFocus->Highlight(false);  
 }  
 APuzzleGameBlock\* HitBlock = HitResult.GetHitObjectHandle().FetchActor<APuzzleGameBlock>();  
 if (HitBlock)  
 {  
 HitBlock->Highlight(true);  
 }  
 CurrentBlockFocus = HitBlock;  
 }  
}

头衔设备操作类

UHeadMountedDisplayFunctionLibrary

创建mesh设置位置旋转

*// Create a mesh component that will be used when being viewed from a '1st person' view (when controlling this pawn)*Mesh1P = CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("CharacterMesh1P"));  
Mesh1P->SetOnlyOwnerSee(true);  
Mesh1P->SetupAttachment(FirstPersonCameraComponent);  
Mesh1P->bCastDynamicShadow = false;  
Mesh1P->CastShadow = false;  
Mesh1P->SetRelativeRotation(FRotator(1.9f, -19.19f, 5.2f));  
Mesh1P->SetRelativeLocation(FVector(-0.5f, -4.4f, -155.7f));

创建一个球体

CollisionComp = CreateDefaultSubobject<USphereComponent>(TEXT("SphereComp"));  
CollisionComp->InitSphereRadius(5.0f);  
CollisionComp->BodyInstance.SetCollisionProfileName("Projectile");  
CollisionComp->OnComponentHit.AddDynamic(this, &AShootGameProjectile::OnHit); *// set up a notification for when this component hits something blocking  
  
// Players can't walk on it*CollisionComp->SetWalkableSlopeOverride(FWalkableSlopeOverride(WalkableSlope\_Unwalkable, 0.f));  
CollisionComp->CanCharacterStepUpOn = ECB\_Yes;

抛物线控制球体运动

*// Use a ProjectileMovementComponent to govern this projectile's movement*ProjectileMovement = CreateDefaultSubobject<UProjectileMovementComponent>(TEXT("ProjectileComp"));  
ProjectileMovement->UpdatedComponent = CollisionComp;  
ProjectileMovement->InitialSpeed = 3000.f;  
ProjectileMovement->MaxSpeed = 3000.f;  
ProjectileMovement->bRotationFollowsVelocity = true;  
ProjectileMovement->bShouldBounce = true;

获取玩家相机的一种方式PlayerCameraManager

APlayerController\* PlayerController = Cast<APlayerController>(Character->GetController());  
const FRotator SpawnRotation = PlayerController->PlayerCameraManager->GetCameraRotation();

在指定位置生成子弹

if(Character == nullptr || Character->GetController() == nullptr)  
{  
 return;  
}  
  
*// Try and fire a projectile*if (ProjectileClass != nullptr)  
{  
 UWorld\* const World = GetWorld();  
 if (World != nullptr)  
 {  
 APlayerController\* PlayerController = Cast<APlayerController>(Character->GetController());  
 const FRotator SpawnRotation = PlayerController->PlayerCameraManager->GetCameraRotation();  
 *// MuzzleOffset is in camera space, so transform it to world space before offsetting from the character location to find the final muzzle position* const FVector SpawnLocation = GetOwner()->GetActorLocation() + SpawnRotation.RotateVector(MuzzleOffset);  
  
 *//Set Spawn Collision Handling Override* FActorSpawnParameters ActorSpawnParams;  
 ActorSpawnParams.SpawnCollisionHandlingOverride = ESpawnActorCollisionHandlingMethod::AdjustIfPossibleButDontSpawnIfColliding;  
  
 *// Spawn the projectile at the muzzle* World->SpawnActor<AShootGameProjectile>(ProjectileClass, SpawnLocation, SpawnRotation, ActorSpawnParams);  
 }  
}

播放音频

UGameplayStatics::PlaySoundAtLocation(this, FireSound, Character->GetActorLocation());

播放蒙太奇动画

*// Try and play a firing animation if specified*if (FireAnimation != nullptr)  
{  
 *// Get the animation object for the arms mesh* UAnimInstance\* AnimInstance = Character->GetMesh1P()->GetAnimInstance();  
 if (AnimInstance != nullptr)  
 {  
 AnimInstance->Montage\_Play(FireAnimation, 1.f);  
 }  
}