// Fill out your copyright notice in the Description page of Project Settings.

#include "Characters/SlashCharacter.h"

#include "GameFramework/SpringArmComponent.h"

#include "Camera/CameraComponent.h"

#include "GameFramework/CharacterMovementComponent.h"

#include "Components/StaticMeshComponent.h"

#include "Components/AttributeComponent.h"

#include "GroomComponent.h"

#include "Items/Items.h"

#include "Items/Weapons/Weapon.h"

#include "Items/Soul.h"

#include "Items/Treasure.h"

#include "Animation/AnimMontage.h"

#include "HUD/SlashHUD.h"

#include "HUD/SlashOverlay.h"

#include "Interfaces/PickupInterface.h"

ASlashCharacter::ASlashCharacter()

{

PrimaryActorTick.bCanEverTick = true;

bUseControllerRotationPitch = false;

bUseControllerRotationRoll = false;

bUseControllerRotationYaw = false;

GetCharacterMovement()->bOrientRotationToMovement = true;

GetCharacterMovement()->RotationRate = FRotator(0.f, 400.f, 0.f);

GetMesh()->SetCollisionObjectType(ECollisionChannel::ECC\_WorldDynamic);

GetMesh()->SetCollisionResponseToAllChannels(ECollisionResponse::ECR\_Ignore);

GetMesh()->SetCollisionResponseToChannel(ECollisionChannel::ECC\_Visibility, ECollisionResponse::ECR\_Block);

GetMesh()->SetCollisionResponseToChannel(ECollisionChannel::ECC\_WorldDynamic, ECollisionResponse::ECR\_Overlap);

GetMesh()->SetGenerateOverlapEvents(true);

CameraBoom = CreateDefaultSubobject<USpringArmComponent>(TEXT("CameraBoom"));

CameraBoom->SetupAttachment(GetRootComponent());

CameraBoom->TargetArmLength = 300.f;

ViewCamera = CreateDefaultSubobject<UCameraComponent>(TEXT("ViewCamera"));

ViewCamera->SetupAttachment(CameraBoom);

Hair = CreateDefaultSubobject<UGroomComponent>(TEXT("Hair"));

Hair->SetupAttachment(GetMesh());

Hair->AttachmentName = FString("head");

Eyebrows = CreateDefaultSubobject<UGroomComponent>(TEXT("Eyebrows"));

Eyebrows->SetupAttachment(GetMesh());

Eyebrows->AttachmentName = FString("head");

}

void ASlashCharacter::Tick(float DeltaTime)

{

if (Attributes && SlashOverlay)

{

Attributes->RegenStamin(DeltaTime);

SlashOverlay->SetStaminaBarPercent(Attributes->GetStaminaPercent());

}

}

void ASlashCharacter::SetupPlayerInputComponent(UInputComponent\* PlayerInputComponent)

{

Super::SetupPlayerInputComponent(PlayerInputComponent);

PlayerInputComponent->BindAxis(FName("MoveForward"), this, &ASlashCharacter::MoveForward);

PlayerInputComponent->BindAxis(FName("MoveRight"), this, &ASlashCharacter::MoveRight);

PlayerInputComponent->BindAxis(FName("Turn"), this, &ASlashCharacter::Turn);

PlayerInputComponent->BindAxis(FName("LookUp"), this, &ASlashCharacter::LookUp);

PlayerInputComponent->BindAction(FName("Jump"), IE\_Pressed, this, &ACharacter::Jump);

PlayerInputComponent->BindAction(FName("Equip"), IE\_Pressed, this, &ASlashCharacter::EKeyPressed);

PlayerInputComponent->BindAction(FName("Attack"), IE\_Pressed, this, &ASlashCharacter::Attack);

PlayerInputComponent->BindAction(FName("Dodge"), IE\_Pressed, this, &ASlashCharacter::Dodge);

}

void ASlashCharacter::Jump()

{

if (IsUnoccupied())

{

Super::Jump();

}

}

float ASlashCharacter::TakeDamage(float DamageAmount, FDamageEvent const& DamageEvent, AController\* EventInstigator, AActor\* DamageCauser)

{

HandleDamage(DamageAmount);

SetHUDHealth();

return DamageAmount;

}

void ASlashCharacter::GetHit\_Implementation(const FVector& ImpactPoint, AActor\* Hitter)

{

Super::GetHit\_Implementation(ImpactPoint, Hitter);

SetWeaponCollisionEnabled(ECollisionEnabled::NoCollision);

if (Attributes && Attributes->GetHealthPercent() > 0)

{

ActionState = EActionState::EAS\_HitReaction;

}

}

void ASlashCharacter::SetOverlappingItem(AItems\* Items)

{

OverlappingItem = Items;

}

void ASlashCharacter::AddSouls(ASoul\* Souls)

{

if (Attributes && SlashOverlay)

{

Attributes->AddSouls(Souls->GetSouls());

SlashOverlay->SetSouls(Attributes->GetSouls());

}

}

void ASlashCharacter::AddGold(ATreasure\* Treasure)

{

if (Attributes && SlashOverlay)

{

Attributes->AddGold(Treasure->GetGold());

SlashOverlay->SetGold(Attributes->GetGold());

}

}

void ASlashCharacter::BeginPlay()

{

Super::BeginPlay();

Tags.Add(FName("EngageableTarget"));

IntializeSlashOverlay();

}

void ASlashCharacter::MoveForward(float Value)

{

if (ActionState != EActionState::EAS\_Unoccupied) return;

if (Controller && (Value != 0.f))

{

const FRotator ControlRotation = GetControlRotation();

const FRotator YawRotation(0.f, ControlRotation.Yaw, 0.f);

const FVector Direction = FRotationMatrix(YawRotation).GetUnitAxis(EAxis::X);

AddMovementInput(Direction, Value);

}

}

void ASlashCharacter::MoveRight(float Value)

{

if (ActionState != EActionState::EAS\_Unoccupied) return;

if (Controller && (Value != 0.f))

{

const FRotator ControlRotation = GetControlRotation();

const FRotator YawRotation(0.f, ControlRotation.Yaw, 0.f);

const FVector Direction = FRotationMatrix(YawRotation).GetUnitAxis(EAxis::Y);

AddMovementInput(Direction, Value);

}

}

void ASlashCharacter::Turn(float Value)

{

AddControllerYawInput(Value);

}

void ASlashCharacter::LookUp(float Value)

{

AddControllerPitchInput(Value);

}

void ASlashCharacter::EKeyPressed()

{

AWeapon\* OverlappingWeapon = Cast<AWeapon>(OverlappingItem);

if (OverlappingWeapon)

{

if (EquippedWeapon)

{

EquippedWeapon->Destroy();

}

EquipWeapon(OverlappingWeapon);

}

else

{

if (CanDisArm())

{

DisArm();

}

else if (CanArm())

{

Arm();

}

}

}

void ASlashCharacter::Attack()

{

if (CanAttack())

{

PlayAttackMontage();

ActionState = EActionState::EAS\_Attacking;

}

}

void ASlashCharacter::Dodge()

{

if (IsOccupied() || !HasEnoughStamina()) return;

PlayDodgeMonatge();

ActionState = EActionState::EAS\_Dodge;

if (Attributes && SlashOverlay)

{

Attributes->UseStamina(Attributes->GetDodgeCost());

SlashOverlay->SetStaminaBarPercent(Attributes->GetStaminaPercent());

}

}

void ASlashCharacter::EquipWeapon(AWeapon\* Weapon)

{

Weapon->Equip(GetMesh(), FName("RightHandSocket"), this, this);

CharacterState = ECharacterState::ECS\_EquippedOneHandedWeapon;

OverlappingItem = nullptr;

EquippedWeapon = Weapon;

}

void ASlashCharacter::AttackEnd()

{

ActionState = EActionState::EAS\_Unoccupied;

}

void ASlashCharacter::DodgeEnd()

{

Super::DodgeEnd();

ActionState = EActionState::EAS\_Unoccupied;

}

bool ASlashCharacter::CanAttack()

{

return ActionState == EActionState::EAS\_Unoccupied &&

CharacterState != ECharacterState::ECS\_Unequipped;

}

bool ASlashCharacter::CanDisArm()

{

return ActionState == EActionState::EAS\_Unoccupied &&

CharacterState != ECharacterState::ECS\_Unequipped;

}

bool ASlashCharacter::CanArm()

{

return ActionState == EActionState::EAS\_Unoccupied &&

CharacterState == ECharacterState::ECS\_Unequipped &&

EquippedWeapon;

}

void ASlashCharacter::DisArm()

{

PlayEquipMontage("Unequip");

CharacterState = ECharacterState::ECS\_Unequipped;

ActionState = EActionState::EAS\_EquippingWeapon;

}

void ASlashCharacter::Arm()

{

PlayEquipMontage("Equip");

CharacterState = ECharacterState::ECS\_EquippedOneHandedWeapon;

ActionState = EActionState::EAS\_EquippingWeapon;

}

void ASlashCharacter::AttachWeaponToBack()

{

if (EquippedWeapon)

{

EquippedWeapon->AttachMeshToSocket(GetMesh(), FName("SpineSocket"));

}

}

void ASlashCharacter::AttachWeaponToHand()

{

if (EquippedWeapon)

{

EquippedWeapon->AttachMeshToSocket(GetMesh(), FName("RightHandSocket"));

}

}

void ASlashCharacter::PlayEquipMontage(const FName& SectionName)

{

UAnimInstance\* AnimInstance = GetMesh()->GetAnimInstance();

if (AnimInstance && EquipMontage)

{

AnimInstance->Montage\_Play(EquipMontage);

AnimInstance->Montage\_JumpToSection(SectionName, EquipMontage);

}

}

void ASlashCharacter::Die\_Implementation()

{

Super::Die\_Implementation();

ActionState = EActionState::EAS\_Dead;

DisableMeshCollision();

}

bool ASlashCharacter::HasEnoughStamina()

{

return Attributes && Attributes->GetStamina() > Attributes->GetDodgeCost();

}

bool ASlashCharacter::IsOccupied()

{

return ActionState != EActionState::EAS\_Unoccupied;

}

void ASlashCharacter::FinishEquipping()

{

ActionState = EActionState::EAS\_Unoccupied;

}

void ASlashCharacter::HitReactEnd()

{

ActionState = EActionState::EAS\_Unoccupied;

}

bool ASlashCharacter::IsUnoccupied()

{

return ActionState == EActionState::EAS\_Unoccupied;

}

void ASlashCharacter::IntializeSlashOverlay()

{

APlayerController\* PlayerController = Cast<APlayerController>(GetController());

if (PlayerController)

{

ASlashHUD\* SlashHUD = Cast<ASlashHUD>(PlayerController->GetHUD());

if (SlashHUD)

{

SlashOverlay = SlashHUD->GetSlashOverlay();

if (SlashOverlay && Attributes)

{

SlashOverlay->SetHealthBarPercent(Attributes->GetHealthPercent());

SlashOverlay->SetStaminaBarPercent(1.f);

SlashOverlay->SetGold(0);

SlashOverlay->SetSouls(0);

}

}

}

}

void ASlashCharacter::SetHUDHealth()

{

if (SlashOverlay && Attributes)

{

SlashOverlay->SetHealthBarPercent(Attributes->GetHealthPercent());

}

}