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


#include "Maincharacter.h"
#include "GameFramework/SpringArmComponent.h"
#include"GameFramework/PlayerController.h"
#include "Camera/CameraComponent.h"
#include"Components/CapsuleComponent.h"
#include"GameFrameWork/CharacterMovementComponent.h"
#include"Engine/World.h"
#include "Kismet/GameplayStatics.h"
#include"Components/SkeletalMeshComponent.h"
#include"Animation/AnimInstance.h"
#include"Weapon1.h"
#include"Sound/SoundCue.h"
#include"Kismet/KismetMathLibrary.h"
#include"enemy1.h"
#include"MainPlayerController.h"

// Sets default values
AMaincharacter::AMaincharacter()
{
    // Set this character to call Tick() every frame.  You can turn this off to improve performance if you don't need it.
    PrimaryActorTick.bCanEverTick = true;
    staticmesh = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("staticmesh"));
    /* creates camera boom i.e pulls towards the player  if there is collision */
    CameraBoom = CreateDefaultSubobject<USpringArmComponent>(TEXT("CameraBoom"));
    CameraBoom->SetupAttachment(GetRootComponent());
    CameraBoom->TargetArmLength = 600.f;//camera follows at this distance
    CameraBoom->bUsePawnControlRotation = true;//rotates arm based on the controller..controller is a inbuilt/pre written class


    /*creates a follow camera */
    FollowCamera = CreateDefaultSubobject<UCameraComponent>(TEXT("FollowCamera"));
    FollowCamera->SetupAttachment(CameraBoom, USpringArmComponent::SocketName); /*there is a socket present and the camera is attached in it */
    FollowCamera->bUsePawnControlRotation = false;/*optional*/

    GetCapsuleComponent()->SetCapsuleSize(48.f, 150.f);

    GetCharacterMovement()->bOrientRotationToMovement = true;
    GetCharacterMovement()->RotationRate = FRotator(0.0f, 540.f, 0.0f);
    GetCharacterMovement()->JumpZVelocity = 650.f;
    GetCharacterMovement()->AirControl = 0.2f;
```

```
44
45      bUseControllerRotationYaw = false;
46      bUseControllerRotationPitch = false;
47      baseturnrate = 300.f;
48      baselookuprate = 65.f;
49
50      bHasCombatTarget = false;
51
52           Maxhealth=100.f;
53           Health=65.f;
54
55           stamina=120.f;
56           Maxstamina=150.f;
57           Runningspeed = 650.f;
58           sprintingspeed = 950.f;
59           bShiftKeyDown = false;
60      /*Initialize enums*/
61           MovementStatus = EMovementStatus::EMS_Normal;
62           StaminaStatus = EStaminaStatus::ESS_Normal;
63           StaminaDrainRate = 25.f;
64           MinSprintStamina = 50.f;
65           isrpressed = false;
66
67           bLMBdown = false;
68
69
70           InterpSpeed = 15.f;
71           bInterptoEnemy = false;
72           bHasCombatTarget = false;
73
74  }
75
76  // Called when the game starts or when spawned
77  void AMaincharacter::BeginPlay()
78  {
79      Super::BeginPlay();
80
81      MainPlayerController = Cast<AMainPlayerController>(GetController
        ());
82  }
83
84  // Called every frame
85  void AMaincharacter::Tick(float DeltaTime)
86  {
87      Super::Tick(DeltaTime);
88      if (MovementStatus == EMovementStatus::EMS_Dead)
89      {
90          return;
91      }
92      float DeltaStamina = StaminaDrainRate * DeltaTime;
93      switch (StaminaStatus) {
94      case EStaminaStatus::ESS_Normal:
95          if (bShiftKeyDown) {
```

```
96              if (stamina - DeltaStamina <= MinSprintStamina) {
97                  SetStaminaStatus(EStaminaStatus::ESS_BelowMin);
98                  stamina -= DeltaStamina;
99              }
100             else {
101                 stamina -= DeltaStamina;
102
103             }
104             SetMovementStatus(EMovementStatus::EMS_Sprinting);
105         }
106         else {
107             if (stamina + DeltaStamina >= Maxstamina) {
108                 stamina = Maxstamina;
109
110             }
111             else {
112                 stamina += DeltaStamina;
113             }
114             SetMovementStatus(EMovementStatus::EMS_Normal);
115         }
116             break;
117     case EStaminaStatus::ESS_BelowMin:
118         if (bShiftKeyDown) {
119             if (stamina - DeltaStamina <= 0.f) {
120                 SetStaminaStatus(EStaminaStatus::ESS_Exhausted);
121                 stamina = 0;
122                 SetMovementStatus(EMovementStatus::EMS_Normal);
123             }
124             else {
125                 stamina -= DeltaStamina;
126                 SetMovementStatus(EMovementStatus::EMS_Sprinting);
127             }
128         }
129         else //shiftkeyup
130         {
131             if (stamina + DeltaStamina >= MinSprintStamina) {
132                 SetStaminaStatus(EStaminaStatus::ESS_Normal);
133                 stamina += DeltaStamina;
134             }
135             else {
136                 stamina += DeltaStamina;
137             }SetMovementStatus(EMovementStatus::EMS_Normal);
138         }
139         break;
140     case EStaminaStatus::ESS_Exhausted:
141         if (bShiftKeyDown) {
142             stamina = 0.f;
143         }
144         else {
145             SetStaminaStatus(EStaminaStatus::ESS_ExhaustedRecovering);
146             stamina += DeltaStamina;
147         }SetMovementStatus(EMovementStatus::EMS_Normal);
148         break;
```

```
149     case EStaminaStatus::ESS_ExhaustedRecovering:
150         if (stamina + DeltaStamina >= MinSprintStamina) {
151             SetStaminaStatus(EStaminaStatus::ESS_Normal);
152             stamina += DeltaStamina;
153         }
154         else {
155             stamina += DeltaStamina;
156         }
157         SetMovementStatus(EMovementStatus::EMS_Normal);
158         break;
159     }
160 
161 
162     if (bInterptoEnemy && CombatTarget)
163     {
164         FRotator LookAtYaw = GetLookAtRotationYaw(CombatTarget-
              >GetActorLocation());
165         FRotator InterpRotation = FMath::RInterpTo(GetActorRotation(),
              LookAtYaw, DeltaTime, InterpSpeed);
166 
167         SetActorRotation(InterpRotation);
168     }
169 
170     if (CombatTarget)
171     {
172         Combattargetlocation = CombatTarget->GetActorLocation();
173         if (MainPlayerController)
174         {
175             MainPlayerController->Enemylocation = Combattargetlocation;
176         }
177     }
178 }
179 FRotator AMaincharacter::GetLookAtRotationYaw(FVector Target)
180 {
181     FRotator Lookatrotation = UKismetMathLibrary::FindLookAtRotation
          (GetActorLocation(), Target);
182     FRotator LookAtRotationYaw  (0.f, Lookatrotation.Yaw, 0.f);
183     return LookAtRotationYaw;
184 }
185 
186 // Called to bind functionality to input
187 void AMaincharacter::SetupPlayerInputComponent(UInputComponent*
      PlayerInputComponent)
188 {
189     Super::SetupPlayerInputComponent(PlayerInputComponent);
190     PlayerInputComponent->BindAction
          ("Jump",IE_Pressed,this,&AMaincharacter::Jump );
191     PlayerInputComponent->BindAction( "Jump", IE_Released, this, &
          ACharacter::StopJumping );
192 
193     PlayerInputComponent->BindAction("LMB", IE_Pressed, this,
          &AMaincharacter::LMBdown);
194     PlayerInputComponent->BindAction("LMB", IE_Released, this,
```

```
        &AMaincharacter::LMBup);
195
196
197     PlayerInputComponent->BindAction("Sprint", IE_Pressed, this,
          &AMaincharacter::ShiftKeyDown);
198     PlayerInputComponent->BindAction("Sprint", IE_Released, this,
          &AMaincharacter::ShiftKeyup);
199
200     PlayerInputComponent->BindAction("roll", IE_Pressed, this,
          &AMaincharacter::rkey);
201
202
203
204     PlayerInputComponent->BindAxis
          ("MoveForward",this,&AMaincharacter::MoveForward );
205     PlayerInputComponent->BindAxis
          ("MoveRight",this,&AMaincharacter::MoveRight);
206
207
208     PlayerInputComponent->BindAxis("Turn", this,
          &APawn::AddControllerYawInput);
209     PlayerInputComponent->BindAxis("LookUp", this,
          &APawn::AddControllerPitchInput);
210     PlayerInputComponent->BindAxis("TurnRate", this,
          &AMaincharacter::TurnAtRate);
211     PlayerInputComponent->BindAxis("LookUpRate", this,
          &AMaincharacter::LookUpRate);
212
213
214
215
216
217
218 }
219
220 void AMaincharacter:: MoveForward(float value)
221 {
222     if (Controller != nullptr && value != 0 &&!bAttacking &&
          MovementStatus!=EMovementStatus::EMS_Dead) {
223         /*Find out which is forward direction*/
224         const FRotator rotation = Controller->GetControlRotation();
225         const FRotator YawRotation(0.f, rotation.Yaw, 0.f);
226         const FVector Direction = FRotationMatrix
              (YawRotation).GetUnitAxis(EAxis::X);
227         AddMovementInput(Direction, value);
228         AutoPossessPlayer = EAutoReceiveInput::Player0;
229     }
230
231
232 }
233 void::AMaincharacter:: MoveRight(float value)
234 {
235     if (Controller != nullptr && value != 0 && !bAttacking &&
```

```
        MovementStatus != EMovementStatus::EMS_Dead) {
236       /*Find out which is forward direction*/
237       const FRotator rotation = Controller->GetControlRotation();
238       const FRotator YawRotation(0.f, rotation.Yaw, 0.f);
239       const FVector Direction = FRotationMatrix
            (YawRotation).GetUnitAxis(EAxis::Y);
240       AddMovementInput(Direction, value);
241
242   }
243 }
244 void AMaincharacter::TurnAtRate(float rate) {
245   AddControllerYawInput(rate * baseturnrate *10* GetWorld()-
        >GetDeltaSeconds());
246
247 }
248 void AMaincharacter::LookUpRate(float rate) {
249   AddControllerPitchInput(rate * baselookuprate * GetWorld()-
        >GetDeltaSeconds());
250 }
251
252 void AMaincharacter::SetMovementStatus(EMovementStatus Status) {
253
254   MovementStatus = Status;
255   if (MovementStatus == EMovementStatus::EMS_Sprinting) {
256       GetCharacterMovement()->MaxWalkSpeed = sprintingspeed;
257   }
258   else {
259       GetCharacterMovement()->MaxWalkSpeed = Runningspeed;
260   }
261
262 }
263 void AMaincharacter::ShiftKeyDown() {
264   bShiftKeyDown = true;
265 }
266 void AMaincharacter::ShiftKeyup() {
267   bShiftKeyDown = false;
268 }
269 void AMaincharacter::rkey() {
270   isrpressed = true;
271 }
272 void AMaincharacter::LMBdown()
273 {
274   bLMBdown = true;
275   if (MovementStatus == EMovementStatus::EMS_Dead) return;
276   if (ActiveOverLappingItem)
277   {
278       AWeapon1* Weapon = Cast<AWeapon1>(ActiveOverLappingItem);
279       if (Weapon)
280       {
281
282           Weapon->Equip(this);
283           SetActiveOverLappingItem(nullptr);
284       }
```

```
285     }
286     else if (EquippedWeapon)
287     {
288             attack();
289 
290     }
291 
292 }
293 void AMaincharacter::LMBup() {
294     bLMBdown = false;
295 }
296 void AMaincharacter::attack()
297 {
298     if (!bAttacking && MovementStatus!=EMovementStatus::EMS_Dead) {
299 
300         bAttacking = true;
301         SetInterptoEnemy(true);
302 
303         UAnimInstance* AnimInstance = GetMesh()->GetAnimInstance();
304         if (AnimInstance && Combatmontage)
305         {
306             int32 Section = FMath::RandRange(0, 1);
307             switch (Section)
308             {
309             case 0 :
310                 AnimInstance->Montage_Play(Combatmontage, 2.2f);
311                 AnimInstance->Montage_JumpToSection(FName("Attack_1"),
                      Combatmontage);
312 
313                 break;
314             case 1:
315                 AnimInstance->Montage_Play(Combatmontage, 1.8f);
316                 AnimInstance->Montage_JumpToSection(FName("attack_2"),
                      Combatmontage);
317                 break;
318             default:;
319             }
320 
321 
322         }
323     }
324     if (EquippedWeapon->SwingSound)
325     {
326         UGameplayStatics::PlaySound2D(this, EquippedWeapon-
              >SwingSound);
327     }
328 }
329 void AMaincharacter::AttackEnd() {
330     bAttacking = false;
331     SetInterptoEnemy(false);
332     if (bLMBdown )
333     {
334         attack();
```

```
335     }
336 }
337
338 void AMaincharacter::SetInterptoEnemy(bool Interp)
339 {
340     bInterptoEnemy = Interp;
341
342 }
343  float AMaincharacter::TakeDamage(float DamageAmount, struct
      FDamageEvent const& DamageEvent, class AController* EventInstigator,
      AActor* DamageCauser)
344 {
345      if (Health - DamageAmount <= 0.f)
346      {
347          Health -= DamageAmount;
348          die();
349          if (DamageCauser)
350          {
351              Aenemy1* enemy = Cast<Aenemy1>(DamageCauser);
352              if (enemy)
353              {
354                  enemy->bhasvalidtarget = false;
355              }
356          }
357      }
358      else {
359          Health -= DamageAmount;
360      }
361      return DamageAmount;
362 }
363  void AMaincharacter::decrementhealth(float amount)
364  {
365
366  }
367  void AMaincharacter::die()
368  {
369      UAnimInstance* AnimInstance = GetMesh()->GetAnimInstance();
370      if (AnimInstance && Combatmontage)
371      {
372          AnimInstance->Montage_Play(Combatmontage, 1.0f);
373          AnimInstance->Montage_JumpToSection(FName("death"));
374      }
375      SetMovementStatus(EMovementStatus::EMS_Dead);
376
377  }
378  void AMaincharacter::Jump()
379  {
380      if (MovementStatus != EMovementStatus::EMS_Dead)
381      {
382          Super::Jump();
383      }
384  }
385  void AMaincharacter::deathend()
```

```cpp
{
    GetMesh()->bPauseAnims = true;
    GetMesh()->bNoSkeletonUpdate = true;

}
void AMaincharacter:: UpdateCombatTarget()
{
    TArray<AActor*>OverLappingActors;
    GetOverlappingActors(OverLappingActors, enemyfilter);

    if (OverLappingActors.Num() == 0)
    {
        if (MainPlayerController)
        {
            MainPlayerController->removeenemyhealthbar();

        }
        return;
    }
    Aenemy1*closestenemy = Cast<Aenemy1>(OverLappingActors[0]);
    if (closestenemy)
    {
        FVector Location = GetActorLocation();
        float MinDistance = (closestenemy->GetActorLocation() -
          Location).Size();
        for (auto Actor : OverLappingActors)
        {
            Aenemy1* enemy = Cast<Aenemy1>(Actor);
            if (enemy)
            {
                float DistanceToActor = (enemy->GetActorLocation() -
                 Location).Size();
                if (DistanceToActor < MinDistance)
                {
                    MinDistance = DistanceToActor;
                    closestenemy = enemy;
                }
            }
        }
        if (MainPlayerController)
        {
            MainPlayerController->displayenemyhealthbar();
        }
        SetCombatTarget(closestenemy);
        bHasCombatTarget = true;
    }

}
```