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

#pragma once

#include "CoreMinimal.h"
#include "GameFramework/Character.h"
#include"Particles/ParticleSystem.h"
#include "Maincharacter.generated.h"

UENUM(BlueprintType)
enum class EMovementStatus : uint8 {
    EMS_Normal UMETA(DisplayName = "Normal"),
    EMS_Sprinting UMETA(DisplayName = "Sprinting"),
    EMS_Max UMETA(DisplayName="DefaultMax"),
    EMS_Dead UMETA(DisplayName="dead")

};
UENUM(blueprinttype)
enum class EStaminaStatus :uint8 {
    ESS_Normal UMETA(DisplayName = "Normal"),
    ESS_BelowMin UMETA(DisplayName = "Belowmin"),
    ESS_Exhausted UMETA(DisplayName = "Exhausted"),
    ESS_ExhaustedRecovering UMETA(DisplayName="ExhaustedRecovering"),
    ESS_MAX UMETA(DisplayName="DefaultMax")

};


UCLASS()
class PARK_API AMaincharacter : public ACharacter
{
    GENERATED_BODY()

public:
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "AI")
        class UParticleSystem* HitParticles;

    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "AI")
        class USoundCue* HitSound;



    FORCEINLINE void SetHasCombatTarget(bool HasTarget) { bHasCombatTarget = HasTarget; }

    UPROPERTY(VisibleAnywhere, Blueprintreadonly, category = "Controller")
        class AMainPlayerController* MainPlayerController;

    // Sets default values for this character's properties
    AMaincharacter();
    UPROPERTY(EditDefaultsOnly,BlueprintReadOnly,Category="Items")
```

```
51      class AWeapon1* EquippedWeapon;
52
53
54
55      UPROPERTY(visibleanywhere,blueprintreadonly,category="Anims")
56      bool bAttacking;
57
58      void die();
59      virtual void Jump() override;
60
61      void attack();
62
63      UPROPERTY(EditDefaultsOnly,BlueprintReadOnly,category="Anims")
64      class UAnimMontage* Combatmontage;
65
66      UFUNCTION(BlueprintCallable)
67      void AttackEnd();
68
69      UPROPERTY(Blueprintreadwrite,visibleanywhere,category="Combat")
70      FVector Combattargetlocation;
71      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Items")
72      class Aitem* ActiveOverLappingItem;
73
74      FORCEINLINE void SetEquippedWeapon(AWeapon1* WeaponToSet)
          { EquippedWeapon = WeaponToSet; }
75      FORCEINLINE void SetActiveOverLappingItem(Aitem* Item)
          { ActiveOverLappingItem = Item; }
76
77      UPROPERTY(VisibleAnywhere,BlueprintReadWrite,Category="Enums")
78      EMovementStatus MovementStatus;
79
80      UPROPERTY(VisibleAnywhere, BlueprintReadWrite, Category = "Enums")
81      EStaminaStatus StaminaStatus;
82
83      FORCEINLINE void SetStaminaStatus(EStaminaStatus Status)
          { StaminaStatus = Status; }
84      UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "movement")
85      float StaminaDrainRate;
86      UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "movement")
87      float MinSprintStamina;
88
89      UPROPERTY(EditAnywhere,Blueprintreadonly,Category="Combat")
90      bool bHasCombatTarget;
91
92      virtual float TakeDamage(float DamageAmount,struct FDamageEvent
          const& DamageEvent,class AController* EventInstigator,AActor*
          DamageCauser) override;
93
94
95      FRotator  GetLookAtRotationYaw(FVector Target);
96
97
98      float InterpSpeed;
```

```
 99
100     bool bInterptoEnemy;
101
102     void SetInterptoEnemy(bool Interp);
103
104     UPROPERTY(VisibleAnywhere,BlueprintReadOnly,Category="Combat")
105     class Aenemy1* CombatTarget;
106
107     UFUNCTION(BlueprintCallable)
108     void deathend();
109
110     FORCEINLINE void SetCombatTarget(Aenemy1* Target) { CombatTarget =
          Target; }
111     /*Set movement status and changing the running speed*/
112     void SetMovementStatus(EMovementStatus Status);
113     UPROPERTY(EditAnywhere,BlueprintReadOnly,category="Running")
114     float Runningspeed;
115     UPROPERTY(EditAnywhere, BlueprintReadOnly, category = "Running")
116     float sprintingspeed;
117
118     bool bShiftKeyDown;
119     /*press to start sprinting*/
120     void ShiftKeyDown();
121     /*Release to stop sprinting*/
122     void ShiftKeyup();
123     /*rolling*/
124     void rkey();
125     UPROPERTY(visibleAnywhere, blueprintreadwrite)
126         bool isrpressed;
127
128
129
130
131     /*positioning camera behind the player*/
132     UPROPERTY(VisibleAnyWhere, BlueprintReadOnly, Category = Camera,
          meta = (AllowPrivateAccess = "true"))
133     class USpringArmComponent* CameraBoom;
134     UPROPERTY(VisibleAnyWhere,BlueprintReadWrite)
135     class UStaticMeshComponent* staticmesh;
136
137     /* this is the camera that follows*/
138     UPROPERTY(VisibleAnyWhere, BlueprintReadWrite, Category = Camera,
          meta = (AllowPrivateAccess = "true"))
139     class UCameraComponent* FollowCamera;
140     /**/
141     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera)
142     float baseturnrate;
143     UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = Camera)
144     float baselookuprate;
145
146
147
148
```

```
149     UPROPERTY(EditAnywhere,BlueprintReadwrite, category = "Player stats")
150     double Health;
151     UPROPERTY(EditDefaultsOnly,BlueprintReadOnly,category="Player stats")
152     double  Maxhealth;
153
154     UPROPERTY(EditAnywhere, BlueprintReadwrite,  category = "Player stats")
155     double stamina;
156
157     UPROPERTY(EditDefaultsOnly, BlueprintReadOnly,  category = "Player stats")
158     double Maxstamina;
159
160     void decrementhealth(float amount);
161
162 protected:
163     // Called when the game starts or when spawned
164     virtual void BeginPlay() override;
165
166 public:
167     // Called every frame
168     virtual void Tick(float DeltaTime) override;
169
170     // Called to bind functionality to input
171     virtual void SetupPlayerInputComponent(class UInputComponent* PlayerInputComponent) override;
172
173     void MoveForward(float value);
174     void MoveRight(float value);
175     void TurnAtRate(float rate);
176     void LookUpRate(float rate);
177
178     bool bLMBdown;
179     void LMBdown();
180     void LMBup();
181
182     //void DecrementHealth(float health);
183     //virtual float  TakeDamage(float DamageAmount, struct FDamageEvent const& DamageEvent, class AController* EventInstigator, AActor* DamageCauser)override;
184
185     FORCEINLINE class USpringArmComponent* GetCameraBoom() const { return CameraBoom; }
186     FORCEINLINE class UCameraComponent* GetFollowCamera()const { return FollowCamera; }
187
188
189
190     void UpdateCombatTarget();
191     UPROPERTY(EditAnywhere,BlueprintReadWrite,Category="Combat")
192         TSubclassOf<Aenemy1>enemyfilter;
```

```
193 };
194
```