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


#include "Weapon1.h"
#include"Maincharacter.h"
#include "Components/SkeletalMeshComponent.h"
#include"Engine/SkeletalMeshSocket.h"
#include "Engine/SkeletalMeshSocket.h"
#include"Sound/SoundCue.h"
#include"Kismet/GameplayStatics.h"
#include"Components/BoxComponent.h"
#include"enemy1.h"

AWeapon1::AWeapon1() {
    SkeletalMesh = CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("SkeletalMesh"));
    SkeletalMesh->SetupAttachment(GetRootComponent());

    Combatcollision = CreateDefaultSubobject<UBoxComponent>(TEXT("CombatCollision"));
    Combatcollision->SetupAttachment(GetRootComponent());




    Damage = 50.f;

}
void AWeapon1::BeginPlay()
{
    Super::BeginPlay();
    Combatcollision->OnComponentBeginOverlap.AddDynamic(this, &AWeapon1::combatOnOverLapbegin);
    Combatcollision->OnComponentEndOverlap.AddDynamic(this, &AWeapon1::combatOnOverLapend);

    Combatcollision->SetCollisionEnabled(ECollisionEnabled::NoCollision);
    Combatcollision->SetCollisionObjectType(ECollisionChannel::ECC_WorldDynamic);
    Combatcollision->SetCollisionResponseToAllChannels(ECollisionResponse::ECR_Ignore);
    Combatcollision->SetCollisionResponseToChannel(ECollisionChannel::ECC_Pawn, ECollisionResponse::ECR_Overlap);
 }

void AWeapon1::OnOverLapbegin(UPrimitiveComponent* OverlappedComponent, AActor* OtherActor, UPrimitiveComponent* OtherComp, int32 otherbodyindex, bool bFromSweep, const FHitResult& SweepResult) {
    Super::OnOverLapbegin(OverlappedComponent, OtherActor, OtherComp, otherbodyindex, bFromSweep, SweepResult);
    if (OtherActor)
```

```
42      {
43          AMaincharacter* main = Cast<AMaincharacter>(OtherActor);
44          if (main)
45          {
46              main ->SetActiveOverLappingItem(this);
47          }
48      }
49  }
50  void AWeapon1::OnOverLapend(UPrimitiveComponent* OverlappedComponent,
      AActor* OtherActor, UPrimitiveComponent* OtherComp, int32
      otherbodyindex) {
51      Super::OnOverLapend(OverlappedComponent, OtherActor, OtherComp,
          otherbodyindex);
52      AMaincharacter* main = Cast<AMaincharacter>(OtherActor);
53      if (main)
54      {
55          main ->SetActiveOverLappingItem(nullptr);
56      }
57  }
58  void AWeapon1:: Equip( AMaincharacter* Char) {
59      if (Char) {
60          SetInstigator(Char->GetController());
61          SkeletalMesh->SetCollisionResponseToChannel
              (ECollisionChannel::ECC_Camera,
              ECollisionResponse::ECR_Ignore);
62          SkeletalMesh->SetCollisionResponseToChannel
              (ECollisionChannel::ECC_Pawn,
              ECollisionResponse::ECR_Ignore);
63          SkeletalMesh->SetSimulatePhysics(false);
64
65          const USkeletalMeshSocket* RightHandSocket = Char->GetMesh()-
              >GetSocketByName("RightHandSocket");
66          if (RightHandSocket)
67          {
68              RightHandSocket->AttachActor(this, Char->GetMesh());
69
70
71              Char->SetEquippedWeapon(this);
72              Char->SetActiveOverLappingItem(nullptr);
73          }
74          if (Onequipsound)UGameplayStatics::PlaySound2D(this,
              Onequipsound);
75
76      }
77
78  }
79
80  void AWeapon1::combatOnOverLapbegin(UPrimitiveComponent*
      OverlappedComponent, AActor* OtherActor, UPrimitiveComponent*
      OtherComp, int32 otherbodyindex, bool bFromSweep, const FHitResult&
      SweepResult)
81  {
82      if (OtherActor)
```

```
 83      {
 84          Aenemy1* Enemy = Cast<Aenemy1>(OtherActor);
 85          if (Enemy)
 86          {
 87              if (Enemy->HitParticles)
 88              {
 89                  const USkeletalMeshSocket* WeaponSocket = SkeletalMesh->GetSocketByName("weaponsocket");
 90                  if (WeaponSocket)
 91                  {
 92                      FVector SocketLocation = WeaponSocket->GetSocketLocation(SkeletalMesh);
 93
 94                      UGameplayStatics::SpawnEmitterAtLocation(GetWorld(), Enemy->HitParticles, SocketLocation, FRotator(0.f), false);
 95                  }
 96
 97              }
 98              if (Enemy->HitSound)
 99              {
100                  UGameplayStatics::PlaySound2D(this, Enemy->HitSound);
101              }
102              if (DamageTypeClass)
103              {
104                  UGameplayStatics::ApplyDamage(Enemy, Damage, WeaponInstigator, this, DamageTypeClass);
105              }
106          }
107      }
108  }
109
110  void AWeapon1::combatOnOverLapend(UPrimitiveComponent* OverlappedComponent, AActor* OtherActor, UPrimitiveComponent* OtherComp, int32 otherbodyindex)
111  {
112
113  }
114
115  void AWeapon1::Activatecollision()
116  {
117      Combatcollision->SetCollisionEnabled(ECollisionEnabled::QueryOnly);
118
119  }
120
121  void AWeapon1::Deactivatecollision()
122  {
123      Combatcollision->SetCollisionEnabled(ECollisionEnabled::NoCollision);
124  }
125
```