```cpp


#include "Gun.h"

#include "Components/SkeletalMeshComponent.h"
#include "Kismet/GameplayStatics.h"
#include "DrawDebugHelpers.h"

AGun::AGun()
{
    PrimaryActorTick.bCanEverTick = true;

    Root = CreateDefaultSubobject<USceneComponent>(TEXT("Root"));
    SetRootComponent(Root);

    Mesh = CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("Mesh"));
    Mesh->SetupAttachment(Root);

}

void AGun::PullTrigger()
{
    UGameplayStatics::SpawnEmitterAttached(MuzzleFlash, Mesh, TEXT
      ("MuzzleFlashSocket"));
    UGameplayStatics::SpawnSoundAttached(MuzzleSound, Mesh, TEXT
      ("MuzzleFlashSocket"));

    FHitResult Hit;
    FVector ShotDirection;
    bool bSuccess = GunTrace(Hit, ShotDirection);
    if (bSuccess)
    {
        UGameplayStatics::SpawnEmitterAtLocation(GetWorld(), ImpactEffect,
          Hit.Location, ShotDirection.Rotation());
        UGameplayStatics::PlaySoundAtLocation(GetWorld(), ImpactSound,
          Hit.Location);

        AActor* HitActor = Hit.GetActor();
        if (HitActor != nullptr)
        {
            FPointDamageEvent DamageEvent(Damage, Hit, ShotDirection,
              nullptr);
            AController* OwnerController = GetOwnerController();
            HitActor->TakeDamage(Damage, DamageEvent, OwnerController,
              this);
        }
    }

```

```
44  }
45
46  void AGun::BeginPlay()
47  {
48      Super::BeginPlay();
49
50  }
51
52  void AGun::Tick(float DeltaTime)
53  {
54      Super::Tick(DeltaTime);
55
56  }
57
58  bool AGun::GunTrace(FHitResult& Hit, FVector& ShotDirection)
59  {
60      AController *OwnerController = GetOwnerController();
61      if (OwnerController == nullptr)
62          return false;
63
64      FVector Location;
65      FRotator Rotation;
66      OwnerController->GetPlayerViewPoint(Location, Rotation);
67      ShotDirection = -Rotation.Vector();
68
69      FVector End = Location + Rotation.Vector() * MaxRange;
70      FCollisionQueryParams Params;
71      Params.AddIgnoredActor(this);
72      Params.AddIgnoredActor(GetOwner());
73      return  GetWorld()->LineTraceSingleByChannel(Hit, Location, End,
          ECollisionChannel::ECC_GameTraceChannel1, Params);
74  }
75
76  AController* AGun::GetOwnerController() const
77  {
78      APawn* OwnerPawn = Cast<APawn>(GetOwner());
79      if (OwnerPawn == nullptr)
80          return nullptr;
81      return OwnerPawn->GetController();
82  }
83
84
```