演示视频：

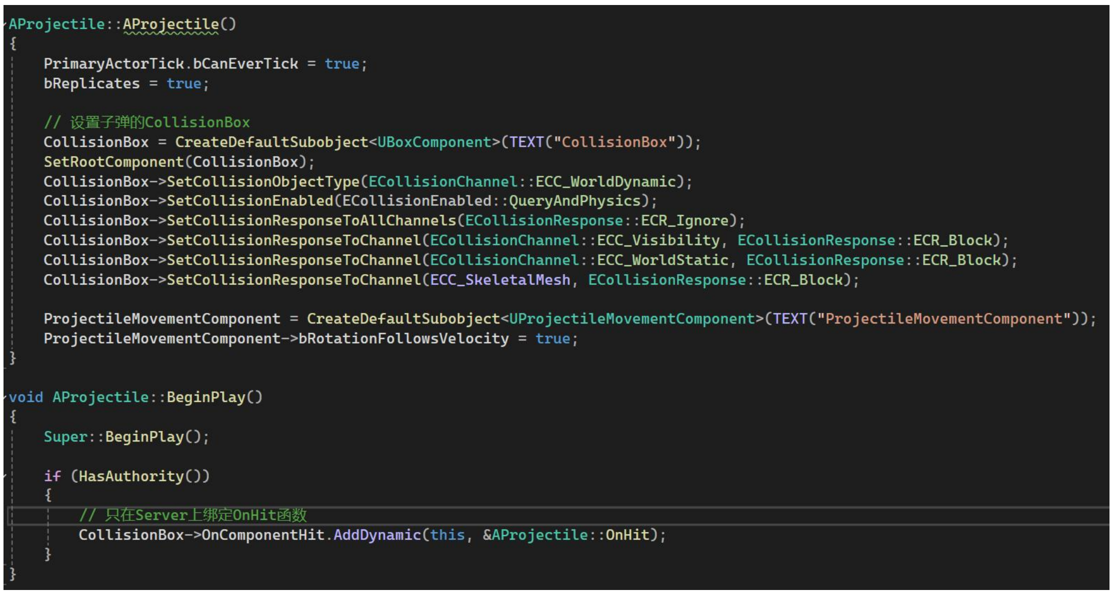
https://www.bilibili.com/video/BV1GDwzeFEyL/?spm_id_from=333.1365.list.card_archive.click&vd_source=f9ab6cfb3a9dcd7822b3ada50a02c2ae

多人联机演示视频（有点小卡）：

https://www.bilibili.com/video/BV1QSwzefE9Y/?spm_id_from=333.1365.list.card_archive.click&vd_source=f9ab6cfb3a9dcd7822b3ada50a02c2ae

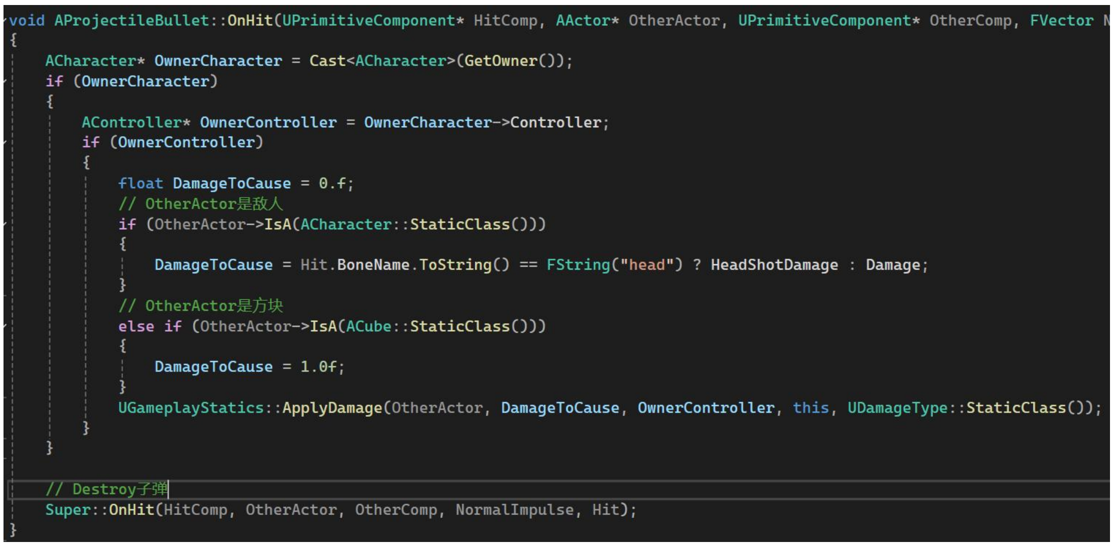
射击命中方块，获得积分 1 分

步骤 1：子弹的 CollisionBox、Add Onhit function



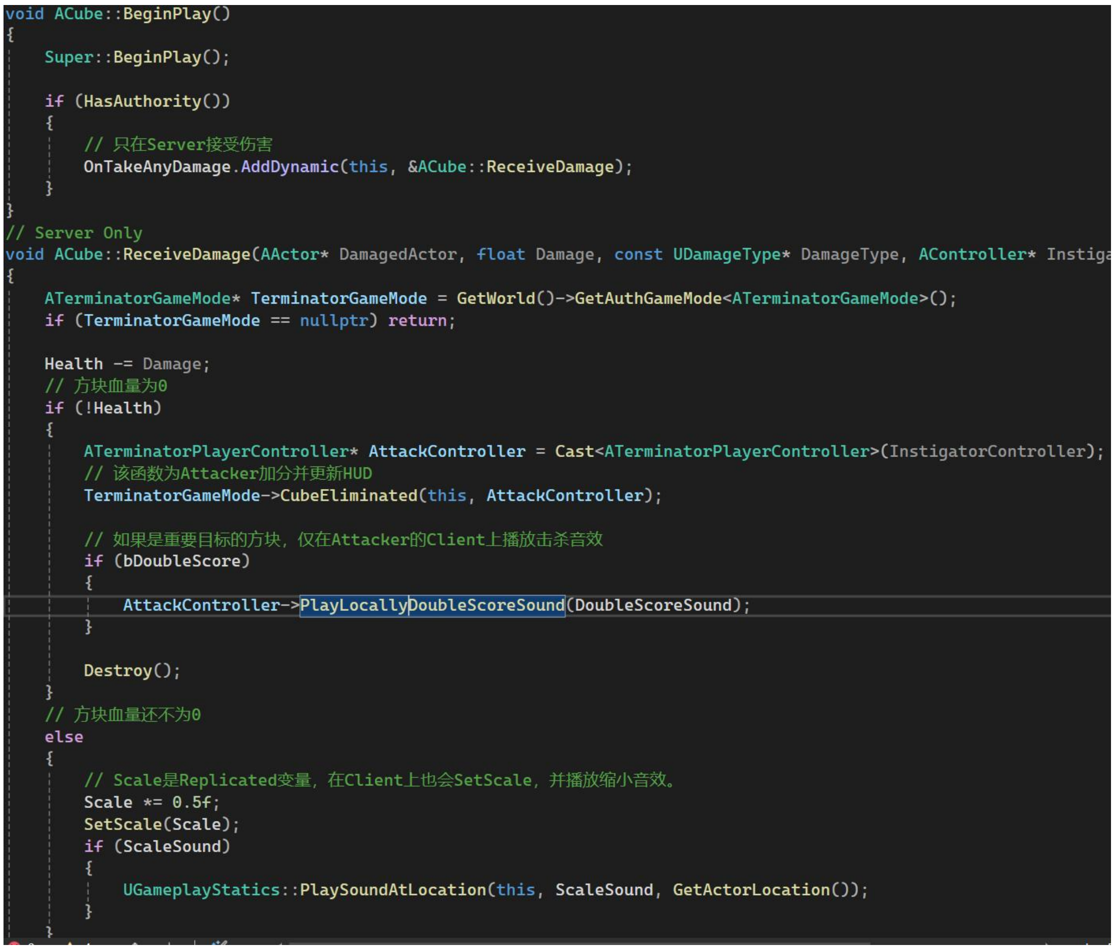
```
AProjectile::AProjectile()
{
    PrimaryActorTick.bCanEverTick = true;
    bReplicates = true;

    // 设置子弹的CollisionBox
    CollisionBox = CreateDefaultSubobject<UBoxComponent>(TEXT("CollisionBox"));
    SetRootComponent(CollisionBox);
    CollisionBox->SetCollisionObjectType(ECollisionChannel::ECC_WorldDynamic);
    CollisionBox->SetCollisionEnabled(ECollisionEnabled::QueryAndPhysics);
    CollisionBox->SetCollisionResponseToAllChannels(ECollisionResponse::ECR_Ignore);
    CollisionBox->SetCollisionResponseToChannel(ECollisionChannel::ECC_Visibility, ECollisionResponse::ECR_Block);
    CollisionBox->SetCollisionResponseToChannel(ECollisionChannel::ECC_WorldStatic, ECollisionResponse::ECR_Block);
    CollisionBox->SetCollisionResponseToChannel(ECC_SkeletalMesh, ECollisionResponse::ECR_Block);

    ProjectileMovementComponent = CreateDefaultSubobject<UProjectileMovementComponent>(TEXT("ProjectileMovementComponent"));
    ProjectileMovementComponent->bRotationFollowsVelocity = true;
}

void AProjectile::BeginPlay()
{
    Super::BeginPlay();

    if (HasAuthority())
    {
        // 只在Server上绑定OnHit函数
        CollisionBox->OnComponentHit.AddDynamic(this, &AProjectile::OnHit);
    }
}
```

步骤 2：子弹的 Onhit function



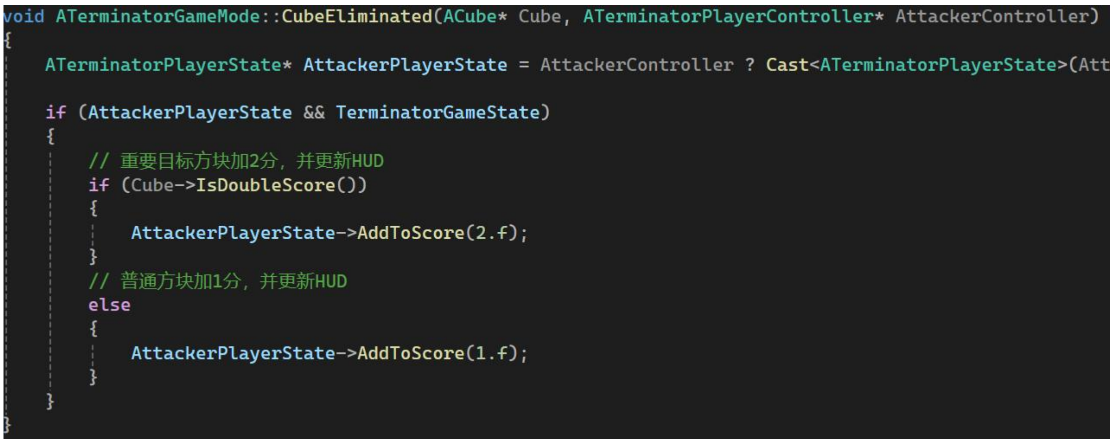
```
void AProjectileBullet::OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor, UPrimitiveComponent* OtherComp, FVector N
{
    ACharacter* OwnerCharacter = Cast<ACharacter>(GetOwner());
    if (OwnerCharacter)
    {
        AController* OwnerController = OwnerCharacter->Controller;
        if (OwnerController)
        {
            float DamageToCause = 0.f;
            // OtherActor是敌人
            if (OtherActor->IsA(ACharacter::StaticClass()))
            {
                DamageToCause = Hit.BoneName.ToString() == FString("head") ? HeadShotDamage : Damage;
            }
            // OtherActor是方块
            else if (OtherActor->IsA(ACube::StaticClass()))
            {
                DamageToCause = 1.0f;
            }
            UGameplayStatics::ApplyDamage(OtherActor, DamageToCause, OwnerController, this, UDamageType::StaticClass());
        }
    }

    // Destroy子弹
    Super::OnHit(HitComp, OtherActor, OtherComp, NormalImpulse, Hit);
}
```

步骤 3：方块 Receive Damage



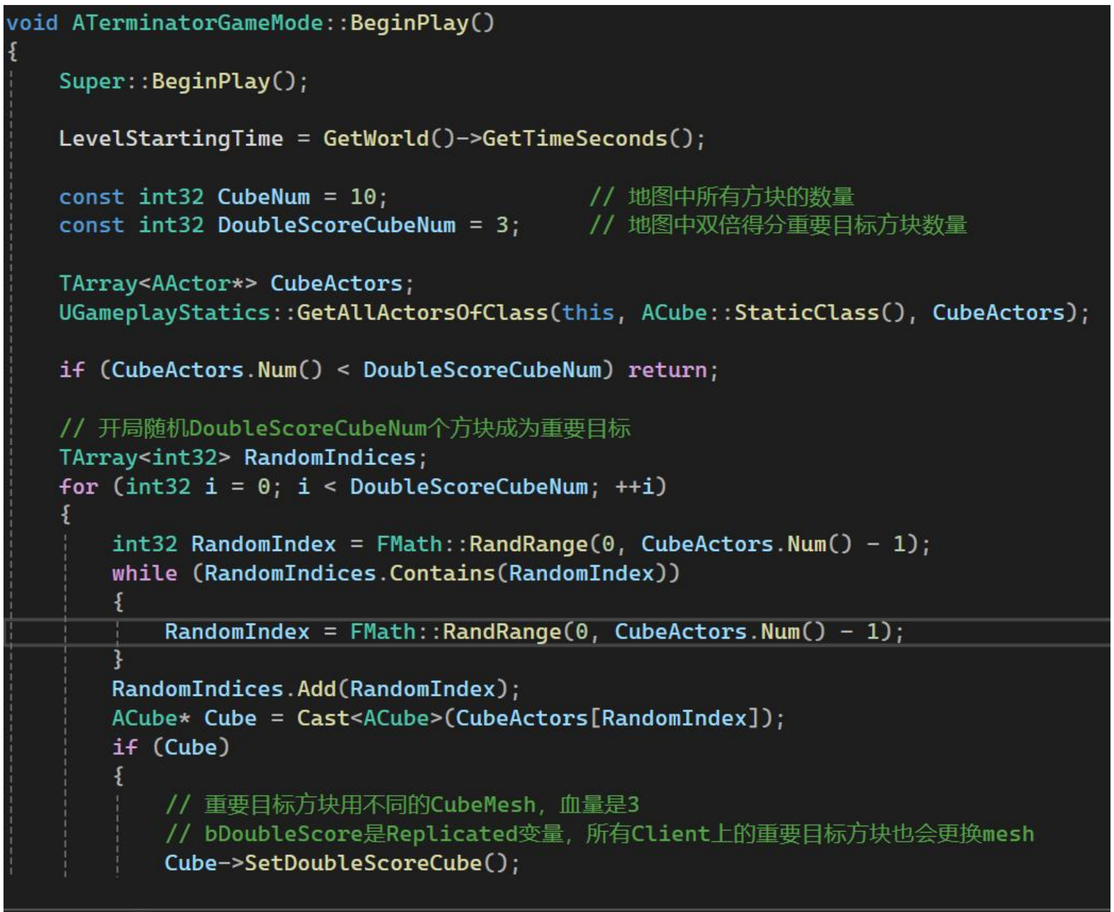
```
void ACube::BeginPlay()
{
    Super::BeginPlay();

    if (HasAuthority())
    {
        // 只在Server接受伤害
        OnTakeAnyDamage.AddDynamic(this, &ACube::ReceiveDamage);
    }
}
// Server Only
void ACube::ReceiveDamage(AActor* DamagedActor, float Damage, const UDamageType* DamageType, AController* Instiga
{
    ATerminatorGameMode* TerminatorGameMode = GetWorld()->GetAuthGameMode<ATerminatorGameMode>();
    if (TerminatorGameMode == nullptr) return;

    Health -= Damage;
    // 方块血量为0
    if (!Health)
    {
        ATerminatorPlayerController* AttackController = Cast<ATerminatorPlayerController>(InstigatorController);
        // 该函数为Attacker加分并更新HUD
        TerminatorGameMode->CubeEliminated(this, AttackController);

        // 如果是重要目标的方块，仅在Attacker的Client上播放击杀音效
        if (bDoubleScore)
        {
            AttackController->PlayLocallyDoubleScoreSound(DoubleScoreSound);
        }

        Destroy();
    }
    // 方块血量还不为0
    else
    {
        // Scale是Replicated变量，在Client上也会SetScale，并播放缩小音效。
        Scale *= 0.5f;
        SetScale(Scale);
        if (ScaleSound)
        {
            UGameplayStatics::PlaySoundAtLocation(this, ScaleSound, GetActorLocation());
        }
    }
```

步骤 4：在 GameMode 中给 AttackerPlayerState 加分并更新 HUD

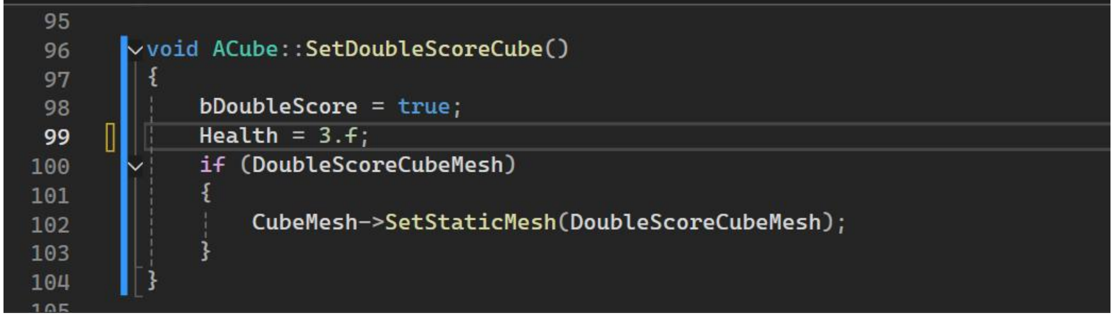


```cpp
void ATerminatorGameMode::CubeEliminated(ACube* Cube, ATerminatorPlayerController* AttackerController)
{
    ATerminatorPlayerState* AttackerPlayerState = AttackerController ? Cast<ATerminatorPlayerState>(Att

    if (AttackerPlayerState && TerminatorGameState)
    {
        // 重要目标方块加2分，并更新HUD
        if (Cube->IsDoubleScore())
        {
            AttackerPlayerState->AddToScore(2.f);
        }
        // 普通方块加1分，并更新HUD
        else
        {
            AttackerPlayerState->AddToScore(1.f);
        }
    }
}
```

步骤 5：游戏开始时，随机 N 个方块成为“重要目标”。

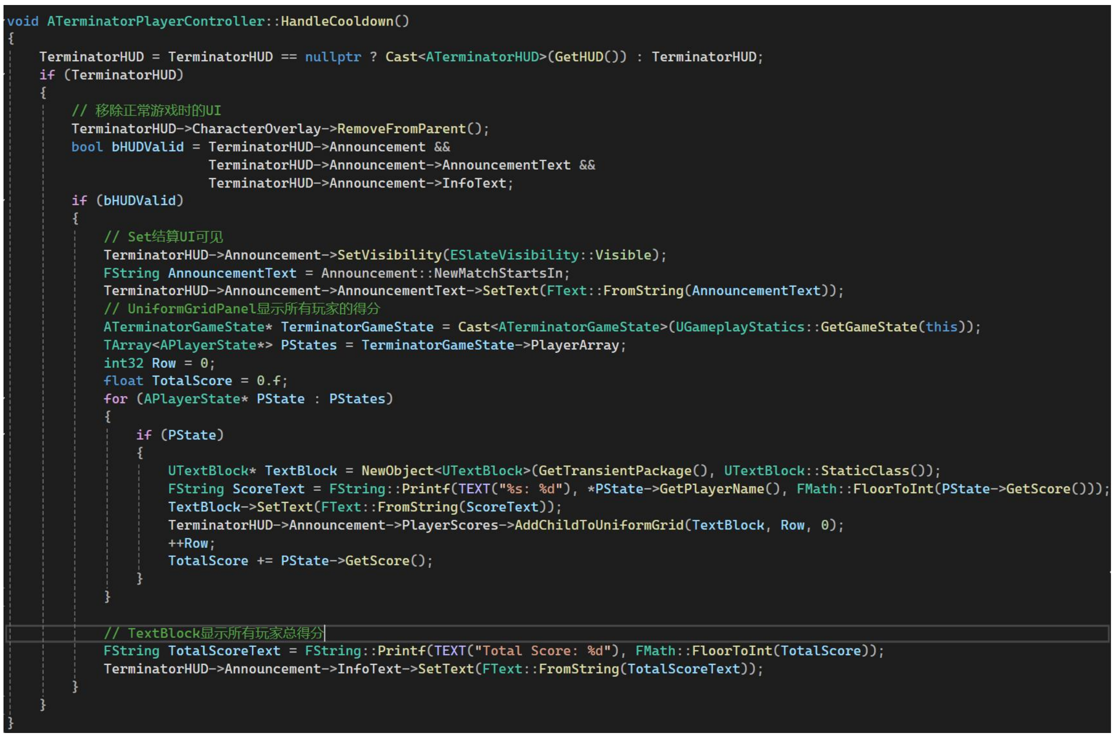


```cpp
void ATerminatorGameMode::BeginPlay()
{
    Super::BeginPlay();

    LevelStartingTime = GetWorld()->GetTimeSeconds();

    const int32 CubeNum = 10;                // 地图中所有方块的数量
    const int32 DoubleScoreCubeNum = 3;      // 地图中双倍得分重要目标方块数量

    TArray<AActor*> CubeActors;
    UGameplayStatics::GetAllActorsOfClass(this, ACube::StaticClass(), CubeActors);

    if (CubeActors.Num() < DoubleScoreCubeNum) return;

    // 开局随机DoubleScoreCubeNum个方块成为重要目标
    TArray<int32> RandomIndices;
    for (int32 i = 0; i < DoubleScoreCubeNum; ++i)
    {
        int32 RandomIndex = FMath::RandRange(0, CubeActors.Num() - 1);
        while (RandomIndices.Contains(RandomIndex))
        {
            RandomIndex = FMath::RandRange(0, CubeActors.Num() - 1);
        }
        RandomIndices.Add(RandomIndex);
        ACube* Cube = Cast<ACube>(CubeActors[RandomIndex]);
        if (Cube)
        {
            // 重要目标方块用不同的CubeMesh，血量是3
            // bDoubleScore是Replicated变量，所有Client上的重要目标方块也会更换mesh
            Cube->SetDoubleScoreCube();
```



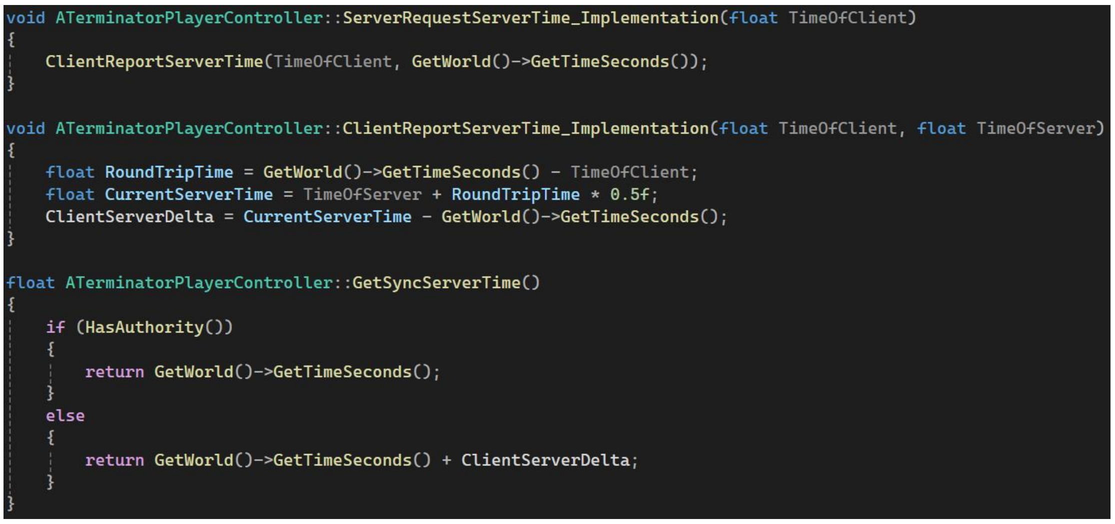
```cpp
void ACube::SetDoubleScoreCube()
{
    bDoubleScore = true;
    Health = 3.f;
    if (DoubleScoreCubeMesh)
    {
        CubeMesh->SetStaticMesh(DoubleScoreCubeMesh);
    }
}
```

步骤 6：游戏结束时，显示每个玩家得分和总得分。在 GameMode 中 Override OnMatchState 函数，该函数中遍历所有 PlayerController。游戏结束时，Controller 的 OnMatchState 函数调用 HandleCoolDown 函数显示游戏结算 UI。

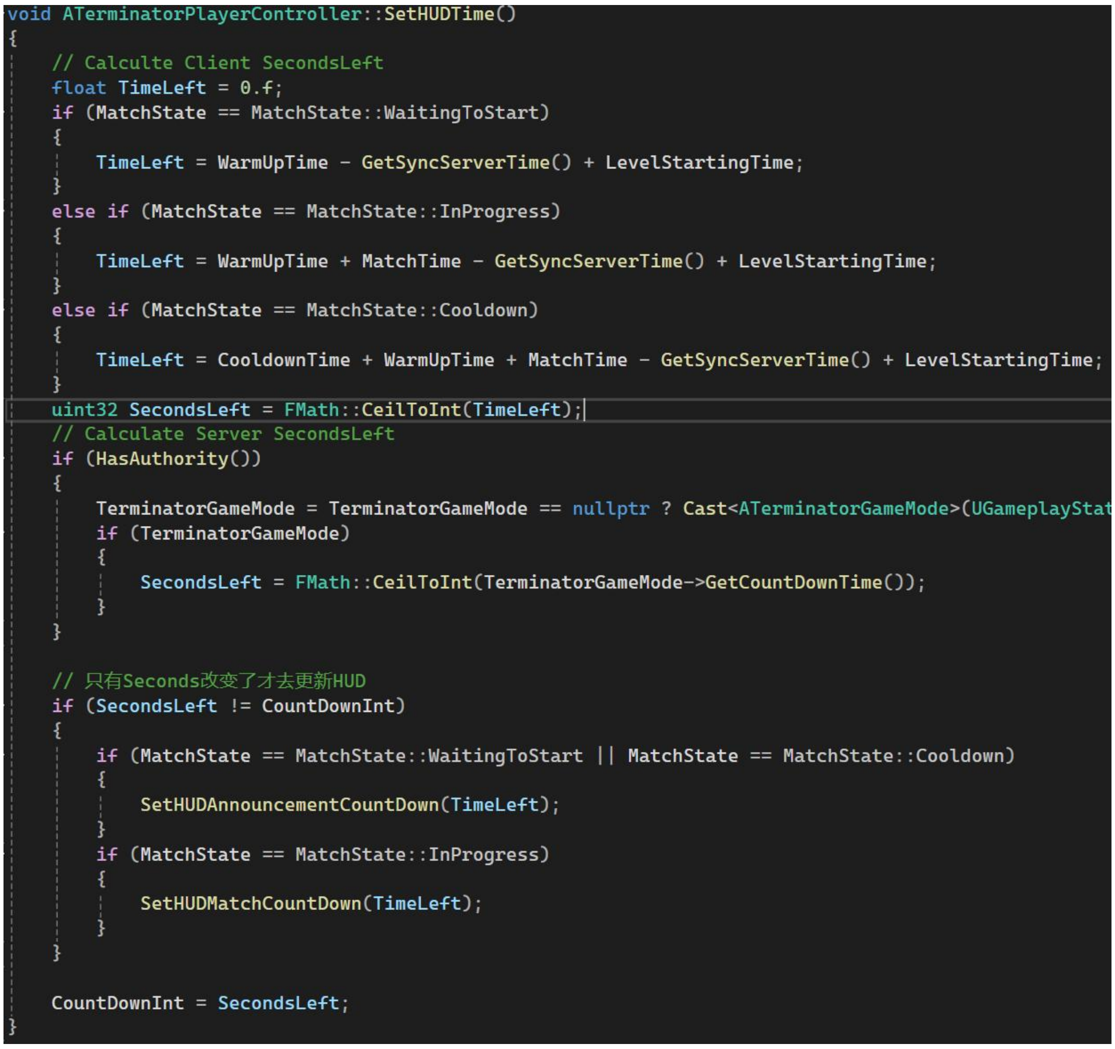


```cpp
void ATerminatorPlayerController::HandleCooldown()
{
    TerminatorHUD = TerminatorHUD == nullptr ? Cast<ATerminatorHUD>(GetHUD()) : TerminatorHUD;
    if (TerminatorHUD)
    {
        // 移除正常游戏时的UI
        TerminatorHUD->CharacterOverlay->RemoveFromParent();
        bool bHUDValid = TerminatorHUD->Announcement &&
                         TerminatorHUD->Announcement->AnnouncementText &&
                         TerminatorHUD->Announcement->InfoText;
        if (bHUDValid)
        {
            // Set结算UI可见
            TerminatorHUD->Announcement->SetVisibility(ESlateVisibility::Visible);
            FString AnnouncementText = Announcement::NewMatchStartsIn;
            TerminatorHUD->Announcement->AnnouncementText->SetText(FText::FromString(AnnouncementText));
            // UniformGridPanel显示所有玩家的得分
            ATerminatorGameState* TerminatorGameState = Cast<ATerminatorGameState>(UGameplayStatics::GetGameState(this));
            TArray<APlayerState*> PStates = TerminatorGameState->PlayerArray;
            int32 Row = 0;
            float TotalScore = 0.f;
            for (APlayerState* PState : PStates)
            {
                if (PState)
                {
                    UTextBlock* TextBlock = NewObject<UTextBlock>(GetTransientPackage(), UTextBlock::StaticClass());
                    FString ScoreText = FString::Printf(TEXT("%s: %d"), *PState->GetPlayerName(), FMath::FloorToInt(PState->GetScore()));
                    TextBlock->SetText(FText::FromString(ScoreText));
                    TerminatorHUD->Announcement->PlayerScores->AddChildToUniformGrid(TextBlock, Row, 0);
                    ++Row;
                    TotalScore += PState->GetScore();
                }
            }

            // TextBlock显示所有玩家总得分
            FString TotalScoreText = FString::Printf(TEXT("Total Score: %d"), FMath::FloorToInt(TotalScore));
            TerminatorHUD->Announcement->InfoText->SetText(FText::FromString(TotalScoreText));
        }
    }
}
```

步骤 7：游戏中，显示剩余游戏时间。同步服务端和客户端的时间。



```cpp
void ATerminatorPlayerController::ServerRequestServerTime_Implementation(float TimeOfClient)
{
    ClientReportServerTime(TimeOfClient, GetWorld()->GetTimeSeconds());
}

void ATerminatorPlayerController::ClientReportServerTime_Implementation(float TimeOfClient, float TimeOfServer)
{
    float RoundTripTime = GetWorld()->GetTimeSeconds() - TimeOfClient;
    float CurrentServerTime = TimeOfServer + RoundTripTime * 0.5f;
    ClientServerDelta = CurrentServerTime - GetWorld()->GetTimeSeconds();
}

float ATerminatorPlayerController::GetSyncServerTime()
{
    if (HasAuthority())
    {
        return GetWorld()->GetTimeSeconds();
    }
    else
    {
        return GetWorld()->GetTimeSeconds() + ClientServerDelta;
    }
}
```



```
void ATerminatorPlayerController::SetHUDTime()
{
    // Calculte Client SecondsLeft
    float TimeLeft = 0.f;
    if (MatchState == MatchState::WaitingToStart)
    {
        TimeLeft = WarmUpTime - GetSyncServerTime() + LevelStartingTime;
    }
    else if (MatchState == MatchState::InProgress)
    {
        TimeLeft = WarmUpTime + MatchTime - GetSyncServerTime() + LevelStartingTime;
    }
    else if (MatchState == MatchState::Cooldown)
    {
        TimeLeft = CooldownTime + WarmUpTime + MatchTime - GetSyncServerTime() + LevelStartingTime;
    }
    uint32 SecondsLeft = FMath::CeilToInt(TimeLeft);
    // Calculate Server SecondsLeft
    if (HasAuthority())
    {
        TerminatorGameMode = TerminatorGameMode == nullptr ? Cast<ATerminatorGameMode>(UGameplayStat
        if (TerminatorGameMode)
        {
            SecondsLeft = FMath::CeilToInt(TerminatorGameMode->GetCountDownTime());
        }
    }

    // 只有Seconds改变了才去更新HUD
    if (SecondsLeft != CountDownInt)
    {
        if (MatchState == MatchState::WaitingToStart || MatchState == MatchState::Cooldown)
        {
            SetHUDAnnouncementCountDown(TimeLeft);
        }
        if (MatchState == MatchState::InProgress)
        {
            SetHUDMatchCountDown(TimeLeft);
        }
    }

    CountDownInt = SecondsLeft;
}
```