Character 文件添加内容

```
51  protected:
52      /** Called for movement input */
53      void Move(const FInputActionValue& Value);
54
55      /** Called for looking input */
56      void Look(const FInputActionValue& Value);
57
58      // 新增积分属性
59      UPROPERTY(VisibleAnywhere, BlueprintReadOnly, Category = "Score")
        已在 0 个 Blueprints 中更改
60      int32 Score;
```

```
19  AHomework2Character::AHomework2Character()
20  {
21      // Set size for collision capsule
22      GetCapsuleComponent()->InitCapsuleSize(55.f, 96.0f);
23
24      // Create a CameraComponent
25      FirstPersonCameraComponent = CreateDefaultSubobject<UCameraComponent>(TEXT("FirstPersonCamera"));
26      FirstPersonCameraComponent->SetupAttachment(GetCapsuleComponent());
27      FirstPersonCameraComponent->SetRelativeLocation(FVector(-10.f, 0.f, 60.f)); // Position the camera
28      FirstPersonCameraComponent->bUsePawnControlRotation = true;
29
30      // Create a mesh component that will be used when being viewed from a '1st person' view (when controlling this pawn)
31      Mesh1P = CreateDefaultSubobject<USkeletalMeshComponent>(TEXT("CharacterMesh1P"));
32      Mesh1P->SetOnlyOwnerSee(true);
33      Mesh1P->SetupAttachment(FirstPersonCameraComponent);
34      Mesh1P->bCastDynamicShadow = false;
35      Mesh1P->CastShadow = false;
36      Mesh1P->SetRelativeLocation(FVector(-30.f, 0.f, -150.f));
37
38      // 初始化积分值
39      Score = 0;
40
41  }
```

```
70  public:
71      /** Returns Mesh1P subobject **/
72      USkeletalMeshComponent* GetMesh1P() const { return Mesh1P; }
73      /** Returns FirstPersonCameraComponent subobject **/
74      UCameraComponent* GetFirstPersonCameraComponent() const { return FirstPersonCameraComponent; }
75
76      /** 增加积分的函数 */
77      UFUNCTION(BlueprintCallable, Category = "Score")
        0 个 Blueprint 引用
78      void AddScore(int32 N);
79
80      /** 积分获取函数 */
81      UFUNCTION(BlueprintCallable, Category = "Score")
        0 个 Blueprint 引用
82      int GetScore() {
83          return Score;
84      }
```

```
43  // 积分函数增加并输入
44  void AHomework2Character::AddScore(int32 N)
45  {
46      Score += N;
47      UE_LOG(LogTemplateCharacter, Log, TEXT("Score added: %d, Current Score: %d"), N, Score);
48  }
```

Projectile 文件添加内容

```
30  public:
31      AHomework2Projectile();
32
33
34      /** called when projectile hits something */
35      UFUNCTION()
        0 个 Blueprint 引用
36      void OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor, UPrimitiveComponent* OtherComp, FVector NormalImpulse, const FHitResult& Hit);
37
38      /** Returns CollisionComp subobject **/
39      USphereComponent* GetCollisionComp() const { return CollisionComp; }
40      /** Returns ProjectileMovement subobject **/
41      UProjectileMovementComponent* GetProjectileMovement() const { return ProjectileMovement; }
42
43      // 蓝图中可修改的积分值
44      UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Projectile")
        已在 0 个 Blueprints 中更改
45      int32 ScoreAmount; // 在蓝图中设置积分
```

```
43      // 蓝图中可修改的积分值
44      UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Projectile")
        已在 0 个 Blueprints 中更改
45      int32 ScoreAmount; // 在蓝图中设置积分
46
47      UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Projectile")
        已在 0 个 Blueprints 中更改
48      int32 Times; // 在蓝图中设置倍数
49
```

```
46
47      UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Projectile")
        已在 0 个 Blueprints 中更改
48      int32 Times; // 在蓝图中设置倍数
49
50      // 保存发射该弹丸的角色
51      UPROPERTY()
        已在 0 个 Blueprints 中更改
52      AHomework2Character* FiringCharacter;
53
54      // 在 Homework2WeaponComponent.cpp 中调用
55      void SetFiringCharacter(AHomework2Character* Character)
56      {
57          FiringCharacter = Character;
58      }
```

```
54      // 在 Homework2WeaponComponent.cpp 中调用
55      void SetFiringCharacter(AHomework2Character* Character)
56      {
57          FiringCharacter = Character;
58      }
59
60      // 追踪每个命中的物体命中次数
61      UPROPERTY()
        已在 0 个 Blueprints 中更改
62      TMap<AActor*, int32> HitCounter; // 物体命中次数
63
64      // 命中后缩小的倍数
65      UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Projectile")
        已在 0 个 Blueprints 中更改
66      int32 SmallTimes;
67
68  };
```

```cpp
    // 默认积分值
    ScoreAmount = 10;

    // 缩小的默认倍数
    SmallTimes = 1;

    // 重要目标增加默认倍数
    Times = 2;

}
```

```cpp
void AHomework2Projectile::OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor, UPrimitiveComponent* OtherComp, FVector NormalImpulse, const FHitResult& Hit)
{
    if ((OtherActor != nullptr) && (OtherActor != this) && (OtherComp != nullptr) && OtherComp->IsSimulatingPhysics())
    {
        // FiringCharacter = Cast<AHomework2Character>(GetOwner());
        // UE_LOG(LogTemp, Log, TEXT("Character Name: %s"), *FiringCharacter->GetName());
        // 调用Character的计分函数，每次命中增加积分并打印

        // 找到射击角色
        if (FiringCharacter)
        {
            AStaticMeshActor* StaticMeshActor = Cast<AStaticMeshActor>(OtherActor);
```

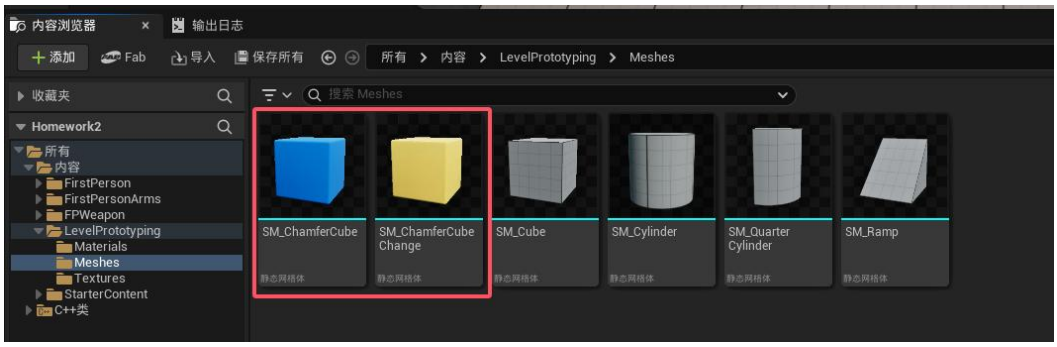
```cpp
        // 找到射击角色
        if (FiringCharacter)
        {
            AStaticMeshActor* StaticMeshActor = Cast<AStaticMeshActor>(OtherActor);

            if (StaticMeshActor && StaticMeshActor->GetStaticMeshComponent())
            {
                // 判断是否命中目标名称包含"SM_ChamferCube"
                if (StaticMeshActor->GetStaticMeshComponent()->GetStaticMesh()->GetName().Contains(TEXT("SM_ChamferCube"))) {
```

```cpp
            if (StaticMeshActor && StaticMeshActor->GetStaticMeshComponent())
            {
                // 判断是否命中目标名称包含"SM_ChamferCube"
                if (StaticMeshActor->GetStaticMeshComponent()->GetStaticMesh()->GetName().Contains(TEXT("SM_ChamferCube"))) {
                    // 判断是否命中重要目标
                    if (StaticMeshActor->GetStaticMeshComponent()->GetStaticMesh()->GetName() == TEXT("SM_ChamferCubeChange"))
                    {
                        int32& HitCount = HitCounter.FindOrAdd(OtherActor); // 获取或初始化该物体的命中次数

                        if (HitCount == 0)
                        {
                            // 第一次命中，缩小物体大小 Y 倍
                            FVector NewScale = StaticMeshActor->GetActorScale3D() * FMath::Pow(0.5f, SmallTimes);
                            StaticMeshActor->SetActorScale3D(NewScale);
                        }
                        else
                        {
                            StaticMeshActor->Destroy();
                        }

                        FiringCharacter->AddScore(Times * ScoreAmount);
                    }
                    else
                    {
                        int32& HitCount = HitCounter.FindOrAdd(OtherActor); // 获取或初始化该物体的命中次数

                        if (HitCount == 0)
                        {
                            // 第一次命中，缩小物体大小一倍
                            FVector NewScale = StaticMeshActor->GetActorScale3D() * FMath::Pow(0.5f, SmallTimes);
                            StaticMeshActor->SetActorScale3D(NewScale);
                        }
                        else
                        {
                            StaticMeshActor->Destroy();
                        }
                        FiringCharacter->AddScore(ScoreAmount);
                    }
                }

            }
            // 只加分，不缩小
            // 判断能否获得静态网格体
```

WeaponComponent 文件添加功能

```
55          // Spawn the projectile at the muzzle
56          // World->SpawnActor<AHomework2Projectile>(ProjectileClass, SpawnLocation, SpawnRotation, ActorSpawnParams);
57          AHomework2Projectile* SpawnedProjectile = World->SpawnActor<AHomework2Projectile>(ProjectileClass, SpawnLocation, SpawnRotation, ActorSpawnParams);
58
59          // 将发射的角色传输给 AHomework2Projectil.cpp
60          if (SpawnedProjectile)
61          {
62              SpawnedProjectile->SetFiringCharacter(Character);  // 设置发射者为当前角色
63          }
```

GameMode 文件添加功能



```h
public:
    AHomework2GameMode();

protected:
    // 重写BeginPlay方法
    virtual void BeginPlay() override;

    // 时间计时器句柄
    FTimerHandle EndGameTimerHandle;

    // 游戏结束倒计时的时间（可以在编辑器中设置）
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Game Settings")
    已在 0 个 Blueprints 中更改
    float TimeToEndGame;
```

```h
    // 声明保存所有玩家总分的变量
    int32 TotalScore;

    // 当前关卡名称（可以在编辑器中设置）
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Game Settings")
    FString CurrentLevelName;

    // 用于显示玩家积分和总积分的 UMG 控件
    /*UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "UI")
    class UUserWidget* ScoreboardWidget;*/

    // 结束游戏
    void EndGame();
```

```h
private:
    // 用于更改颜色的函数
    void ChangeColorOfCubes(int32 N);
public:
    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Weapon")
    已在 0 个 Blueprints 中更改
    int32 SpCubeNum; // 特殊方块个数，可修改

};
```

```cpp
AHomework2GameMode::AHomework2GameMode()
    : Super()
{
    // set default pawn class to our Blueprinted character
    static ConstructorHelpers::FClassFinder<APawn> PlayerPawnClassFinder(TEXT("/Game/FirstPerson/Blueprints/BP_FirstPersonCharacter"));
    DefaultPawnClass = PlayerPawnClassFinder.Class;

    // 特殊方块数默认为5
    SpCubeNum = 5;

    // 默认结束时间10s
    TimeToEndGame = 10.0f;

    // 总积分初始为0
    TotalScore = 0;

    // 关卡名称默认为"FirstPersonMap"
    CurrentLevelName = "FirstPersonMap";
}
```

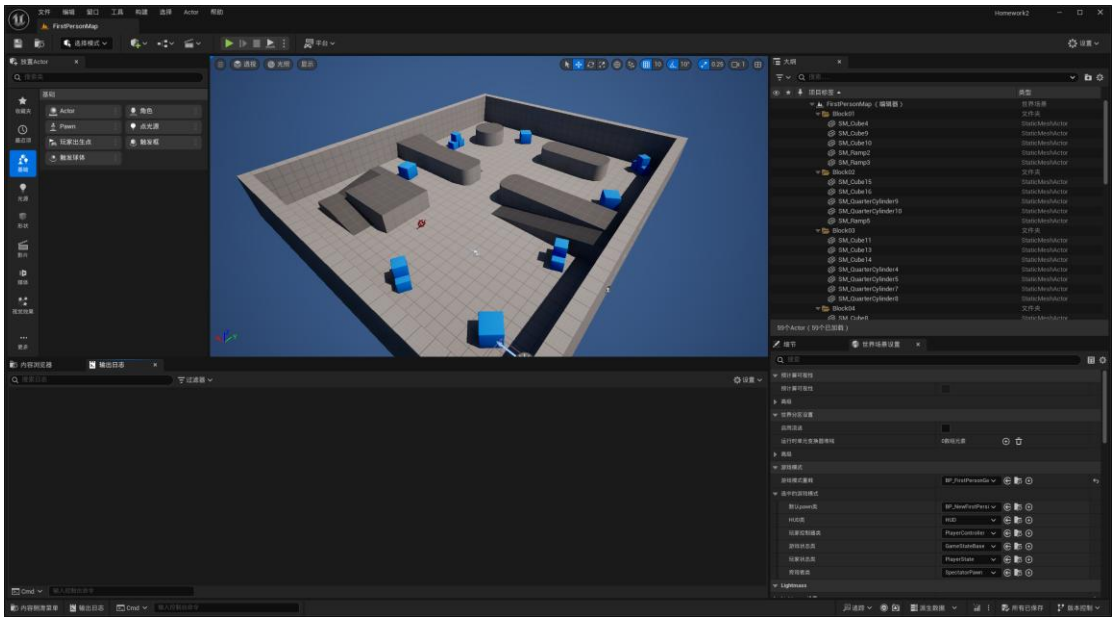
```cpp
void AHomework2GameMode::BeginPlay()
{
    Super::BeginPlay();  // 调用父类的BeginPlay()

    // 调用更改颜色的函数设置特殊方块
    ChangeColorOfCubes(SpCubeNum);

    // 设置计时器，在 TimeToEndGame 秒后结束游戏
    GetWorld()->GetTimerManager().SetTimer(EndGameTimerHandle, this, &AHomework2GameMode::EndGame, TimeToEndGame, false);

    // 更新 UMG 显示的积分
    // UpdateScoreboard();

}
```

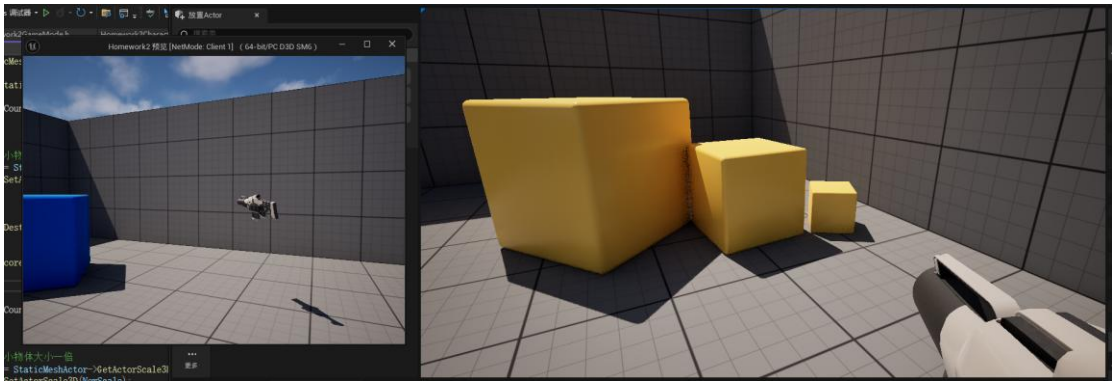
```cpp
void AHomework2GameMode::EndGame()
{
    // 游戏结束的逻辑
    UE_LOG(LogTemplateGameMode, Log, TEXT("Game Over!"));

    // 获取所有Homework2Character类型的角色
    TArray<AActor*> FoundCharacters;
    UGameplayStatics::GetAllActorsOfClass(GetWorld(), AHomework2Character::StaticClass(), FoundCharacters);

    // 遍历所有找到的角色，累加Score
    for (AActor* Actor : FoundCharacters)
    {
        AHomework2Character* Character = Cast<AHomework2Character>(Actor);
        if (Character)
        {
            // 使用GetScore()获取每个角色的分数
            TotalScore += Character->GetScore();
            // 输出个人分数
            UE_LOG(LogTemplateGameMode, Warning, TEXT("%s's Score: %d"), *Character->GetName(), Character->GetScore());
        }
    }

    // 输出总分
    UE_LOG(LogTemplateGameMode, Warning, TEXT("Total Score: %d"), TotalScore);

    // 在此处可以触发UI提示、结算等操作
    // 例如：显示总积分并结束游戏

    // 结束游戏并退出
    UKismetSystemLibrary::QuitGame(GetWorld(), nullptr, EQuitPreference::Quit, false);
}

```

```cpp
void AHomework2GameMode::ChangeColorOfCubes(int32 N)
{
    // UE_LOG(LogTemplateGameMode, Log, TEXT("In ChangeColorOfRandomSMChamferCubes"));
    TArray<AActor*> AllActors;
    UGameplayStatics::GetAllActorsOfClass(GetWorld(), AStaticMeshActor::StaticClass(), AllActors);

    TArray<AStaticMeshActor*> ChamferCubes;

    // 筛选出符合名称的物体
    for (AActor* Actor : AllActors)
    {
        // UE_LOG(LogTemplateGameMode, Log, TEXT("In Select"));
        AStaticMeshActor* StaticMeshActor = Cast<AStaticMeshActor>(Actor);
        if (StaticMeshActor && StaticMeshActor->GetStaticMeshComponent())
        {
            if (StaticMeshActor->GetStaticMeshComponent()->GetStaticMesh()->GetName() == TEXT("SM_ChamferCube"))
            {
                ChamferCubes.Add(StaticMeshActor);
            }
        }
    }

    // 如果符合条件的物体少于N个，随机选择所有
    if (ChamferCubes.Num() <= N)
    {
        N = ChamferCubes.Num();
    }

    // 初始化均为未选中
    TArray<bool> bIfChoose;
    for (int32 i = 0; i < ChamferCubes.Num(); ++i)
    {
        bIfChoose.Add(false);
    }

    // 加载新网格体
```

```cpp
    // 加载新网格体
    UStaticMesh* NewStaticMesh = Cast<UStaticMesh>(StaticLoadObject(UStaticMesh::StaticClass(), nullptr,
        TEXT("/Game/LevelPrototyping/Meshes/SM_ChamferCubeChange.SM_ChamferCubeChange")));
    if (!NewStaticMesh)
    {
        // UE_LOG(LogTemplateGameMode, Error, TEXT("Failed to load new static mesh!"));
        return;  // 如果无法加载新网格体，退出函数
    }

    // 随机选取N个物体并修改颜色
    for (int32 i = 0; i < N; ++i)
    {
        // UE_LOG(LogTemplateGameMode, Log, TEXT("In Choose Color"));
        AStaticMeshActor* SelectedActor = nullptr;
        while(1)
        {
            // 如果未选过则记录选中的Actor，否则重新选
            int32 RandomIndex = FMath::RandRange(0, ChamferCubes.Num() - 1);
            if (!bIfChoose[RandomIndex]) {
                SelectedActor = ChamferCubes[RandomIndex];
                bIfChoose[RandomIndex] = true;
                break;
            }
        }

        if (SelectedActor != nullptr)
        {
            // UE_LOG(LogTemplateGameMode, Log, TEXT("In Change StaticMesh "));
            SelectedActor->GetStaticMeshComponent()->SetStaticMesh(NewStaticMesh);  // 设置新网格体
        }
    }
}
```

运行结果展示





```
LogTemplateCharacter: Character Name: PlayerController_0 Score added: 20, Current Score: 20
LogTemplateGameMode: Game Over!
LogTemplateGameMode: Warning: BP_NewFirstPersionCharacter_C_0's Score: 20
LogTemplateGameMode: Warning: BP_NewFirstPersionCharacter_C_1's Score: 0
LogTemplateGameMode: Warning: Total Score: 20
```