RoomRPG

develop period	@2024년 7월 3일 → 2024년 7월 30일
roll	프로그래밍
category	개인프로젝트 게임
size	Large
FrameWork	Unreal
언어	C++

개요

전체 코드는 다음 깃에 올라가 있습니다.

https://github.com/kimkyungjae1112/RoomRPG

게임 기획

언리얼을 이용한 첫 게임 개발로 언리얼 사용에 익숙해지기 위한 프로젝트

플레이 영상

https://prod-files-secure.s3.us-west-2.amazonaws.com/7ebf7000-f855-492a-973c-2e002d905ac7/9811508f-772a-4335-a101-ca5e2d36fc9d/RoomRPG_%ED%94%8C%EB%A0%88%EC%9D%B4_%EC%98%81%EC%83%81.mp4

구현한 기능들

캐릭터 관련 기능

1. 캐릭터 움직임
    a. 움직이기
    b. 공격 모션
    c. 애니메이션
    d. 스킬
2. 몬스터 움직임
    a. 몬스터 스폰
    b. 몬스터 종류
    c. 몬스터 행동 트리
3. 캐릭터와 몬스터 전투 시스템
4. 체력바

게임 시스템 관련 기능

1. 맵
    a. 메인메뉴 → 스테이지 → 엔딩 크레딧
2. UI
    a. 체력바
    b. 스킬 슬롯
    c. 경험치
3. 스텟
    a. 공격력, 체력, 이속 등

개발 기록

▼ 캐릭터

▼ 움직이기

2024년 7월 9일

- Enhanced Input 활용 움직임 구현
- Input 관련 선언문

▼ 코드

```
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Input")
TObjectPtr<class UInputMappingContext> InputMappingContext;

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Input")
TObjectPtr<class UInputAction> MoveAction;

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Input")
TObjectPtr<class UInputAction> LookAction;
```

```
UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Input")
TObjectPtr<class UInputAction> JumpAction;

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Input")
TObjectPtr<class UInputAction> RunAction;

UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "Input")
TObjectPtr<class UInputAction> AttackAction;

void Move(const FInputActionValue& Value);
void Look(const FInputActionValue& Value);
void Run();
void StopRuning();
void Attack();
```

▼ 애니메이션

2024년 7월 9일

- 걷기, 뛰기, 점프, 공격 애니메이션 적용
- AnimInstance 상속받은 클래스 이용

애니메이션 공부 정리

- 걷기, 뛰기
  - Bland Space 1D 이용
- 공격
  - 애니메이션 몽타주 이용
  - Montage_Play, Montage_SetEndDelegate, Montage_JumpToSection

▼ 코드

```
void ARCharacter::ComboActionBegin()
{
    CurrentCombo = 1;

    GetCharacterMovement()->SetMovementMode(EMovementMode::MOVE_None);

    const float AttackSpeedRate = 1.0f;
    UAnimInstance* AnimInstance = GetMesh()->GetAnimInstance();
    AnimInstance->Montage_Play(ComboActionMontage, AttackSpeedRate);

    FOnMontageEnded EndDelegate;
    EndDelegate.BindUObject(this, &ARCharacter::ComboActionEnd);
    AnimInstance->Montage_SetEndDelegate(EndDelegate, ComboActionMontage);

    ComboTimerHandle.Invalidate();
    SetComboCheckTimer();
```

```
}

void ARCharacter::ComboActionEnd(UAnimMontage* TargetMontage, bool IsPrope
{
    ensure(CurrentCombo != 0);
    CurrentCombo = 0;
    GetCharacterMovement()->SetMovementMode(EMovementMode::MOVE_Walking);
}

void ARCharacter::SetComboCheckTimer()
{
    int32 ComboIndex = CurrentCombo - 1;
    ensure(ComboActionData->EffectiveFrameCount.IsValidIndex(ComboIndex));

    const float AttackSpeedRate = 1.0f;
    float ComboEffectiveTime = (ComboActionData->EffectiveFrameCount[Combo
    if (ComboEffectiveTime > 0.0f)
    {
        GetWorld()->GetTimerManager().SetTimer(ComboTimerHandle, this, &AR
    }
}

void ARCharacter::ComboCheck()
{
    ComboTimerHandle.Invalidate();
    if (HasNextComboCommand)
    {
        UAnimInstance* AnimInstance = GetMesh()->GetAnimInstance();

        CurrentCombo = FMath::Clamp(CurrentCombo + 1, 1, ComboActionData->
        FName NextSection = *FString::Printf(TEXT("%s%d"), *ComboActionDat
        AnimInstance->Montage_JumpToSection(NextSection, ComboActionMontag
        SetComboCheckTimer();
        HasNextComboCommand = false;
    }
}
```

- Notify
  - 캐릭터 공격 판정 구현

▼ 코드

```
/* Notify 클래스 */
void UAnimNotify_AttackHitCheck::Notify(USkeletalMeshComponent* MeshComp,
{
    Super::Notify(MeshComp, Animation, EventReference);

    if (MeshComp)
    {
        IRAnimaionAttackInterface* AttackPawn = Cast<IRAnimaionAttackInter
```

```
        if (AttackPawn)
        {
            AttackPawn->AttackHitCheck();
        }
    }
}


/* 가상함수를 구현한 캐릭터 클래스 */
void ARCharacter::AttackHitCheck()
{
    FHitResult HitResult;
    FCollisionQueryParams Params(SCENE_QUERY_STAT(Attack), false, this);

    const float AttackRange = 50.0f;
    const float AttackRadius = 50.0f;
    const float AttackDamage = 30.0f;
    const FVector Start = GetActorLocation() + GetActorForwardVector() * G
    const FVector End = Start + GetActorForwardVector() * AttackRange;

    bool Hit = GetWorld()->SweepSingleByChannel(HitResult, Start, End, FQu
    if (Hit)
    {

    }

#if ENABLE_DRAW_DEBUG

    FVector CapsuleOrigin = Start + (End - Start) * 0.5f;
    float CapsuleHalfHeight = AttackRange * 0.5f;
    FColor DrawColor = Hit ? FColor::Green : FColor::Red;

    DrawDebugCapsule(GetWorld(), CapsuleOrigin, CapsuleHalfHeight, AttackR

#endif
}
```

2024년 7월 18일

- 죽음 애니메이션 적용
  - 캐릭터 스텟에서 체력 0되면 델리게이트로 브로드캐스트 보내게 함
  - 그럼 캐릭터에선 Dead Montage 실행해주는 함수만 호출하면 됨

▼ 코드

```
#pragma once

#include "CoreMinimal.h"
#include "Components/ActorComponent.h"
#include "RCharacterStat.generated.h"
```

```
DECLARE_MULTICAST_DELEGATE(FOnZeroDelegate)
DECLARE_MULTICAST_DELEGATE_OneParam(FOnChangeDelegate, float)

UCLASS( ClassGroup=(Custom), meta=(BlueprintSpawnableComponent) )
class ROOMRPG_API URCharacterStat : public UActorComponent
{
    GENERATED_BODY()

public:
    // Sets default values for this component's properties
    URCharacterStat();

public:
    FOnChangeDelegate OnChange;
    FOnZeroDelegate OnZero;

    FORCEINLINE float GetMaxHp() { return MaxHp; }
    FORCEINLINE float GetCurrentHp() { return CurrentHp; }

    float ApplyDamage(float InDamage);
    void SetHp(float NewHp);

private:
    float MaxHp;
    float CurrentHp;

};
```

▼ 스킬

2024년 7월 9일

- R 스킬 (회오리 치기) 구현
- 애니메이션 몽타주 이용했음

▼ 코드

```
void ARCharacter::R_SkillBegin()
{
    GetCharacterMovement()->SetMovementMode(EMovementMode::MOVE_None);
    UAnimInstance* AnimInstance = GetMesh()->GetAnimInstance();
    AnimInstance->Montage_Play(R_SkillActionMontage);

    FOnMontageEnded EndDelegate;
    EndDelegate.BindUObject(this, &ARCharacter::R_SkillEnd);
```

```
    AnimInstance->Montage_SetEndDelegate(EndDelegate, R_SkillActionMontage

}

void ARCharacter::R_SkillEnd(UAnimMontage* TargetMontage, bool IsProperlyE
{
    GetCharacterMovement()->SetMovementMode(EMovementMode::MOVE_Walking);
}
```

2024년 7월 10일

- R 스킬 피격 판정 구현
- OverlapMultiByChannel 이용
- 하지만 정작 몬스터는 안 맞는 버그 발생

▼ 코드

```
void ARCharacter::R_SkillHitCheck()
{
    FCollisionQueryParams Params(NAME_None, false, this);

    const float AttackRange = 400.0f;
    const float AttackDamage = 100.0f;

    DrawDebugCapsule(GetWorld(), GetActorLocation(), AttackRange, AttackRa


    const FVector Extent(AttackRange, AttackRange, 300.0f);
    TArray<FOverlapResult> OutOverlaps;

    bool Hit = GetWorld()->OverlapMultiByChannel(OutOverlaps, GetOwner()->
    if (Hit)
    {
        UE_LOG(LogTemp, Display, TEXT("스킬 발동"));

        FDamageEvent DamageEvent;
        for (const FOverlapResult& Result : OutOverlaps)
        {
            DrawDebugCapsule(GetWorld(), GetActorLocation(), AttackRange, 
            Result.GetActor()->TakeDamage(AttackDamage, DamageEvent, GetCo
            UE_LOG(LogTemp, Display, TEXT("맞은 액터 이름 : %s"), *Result.Get
        }
    }
}
```

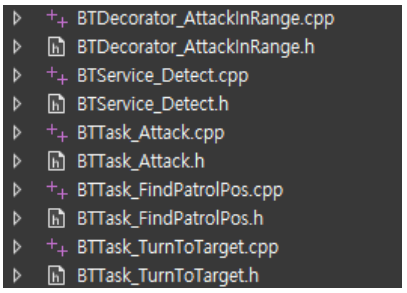
▼ 몬스터

▼ 몬스터 행동트리

2024년 7월 16일

- 몬스터 행동 트리 거의 완
  - 기본적인 공격, 추적, 경계 구현

2024년 7월 17일



- 몬스터 행동트리에 필요한 클래스들
  - 데코레이터와 서비스, 태스크 등이 있다.

▼ 몬스터와 전투

2024년 7월 17일

- 몬스터 체력 기능 추가
- 몬스터 일정 수준 체력 떨어지면 스킬 발동

2024년 7월 18일

- 몬스터 죽을 시 아이템 드랍함
  - DropItem 이란 Actor Component를 만들어서 몬스터에게 부착
  - 몬스터 스텟에서 체력 0되면 델리게이트로 브로드캐스트 보내게 함
  - DropItem에서 델리게이트에 등록하고 죽은 뒤 액터 소환하게 작성

▼ 코드

```
void UDropItem::Drop()
{
    FTimerHandle DeadTimer;
    GetWorld()->GetTimerManager().SetTimer(DeadTimer, [this]()
        {
            GetWorld()->SpawnActor<AActor>(ItemClass, GetOwner()->GetActor
        }, MonsStat->GetDestroyTime(), false);
}

void UDropItem::SetDropItem(URMonsterStat* InMonsStat)
{
    MonsStat = InMonsStat;
```

```
    MonsStat->OnHpZeroDelegate.AddUObject(this, &UDropItem::Drop);
}
```

▼ 보스

2024년 7월 20일

- 보스 캐릭터 메시 임포트 후 만들어져있던 AIController 이용해서 움직임 애니메이션 제작
- 콜리전 설정으로 캐릭터에게 공격 받은 후 죽음 애니메이션 재생

2024년 7월 21일

- 먼저 미리 만들어둔 Notify를 이용해 공격 몽타주에서 공격 판정 만들 수 있었음 근데 이때는 박스로 공격 범위를 설정했다.
- 몬스터 Stat을 조정하는 과정에서 EditInstanceOnly로 설정을 하기만 하면 보스나 몬스터가 먹통이 되가지고 한 5시간은 날린 것 같다. 이거는 어떻게 해결해야하지? 나는 Boss용 클래스를 하나 더 만들어서 해결했다만 너무 비효율의 극치같다.
- 스킬1
  - 보스의 체력이 몇퍼 이하가 되고, 이때 캐릭터한테 맞으면 일정 확률로 실행되게 하였다.
  - 몬스터를 소환하는 스킬이며 한 번 사용할 때마다 한마리를 소환한다.
- 스킬2
  - 나이아가라 이펙트를 활용했다.
  - 처음에는 그냥 불타오르는 이펙트
  - 바닥에 부딛히면 폭발 이펙트
  - 폭발이 끝나고나면 지면이 타는 이펙트로 구성했다.
  - OnComponentHit와 OnComponentOverlap을 활용해 구현할 수 있었다.
  - 이펙트와 거리에 따라 캐릭터가 입는 대미지가 달라지게 만들었다.
  - 보스 캐릭터와 보스가 던지는 액터를 서로 인터페이스로 연결해 OtherActor의 TakeDamage를 부를 수 있었다.

  ▼ 코드

```
// Fill out your copyright notice in the Description page of Project Se


#include "Prop/RBomb.h"
#include "GameFramework/ProjectileMovementComponent.h"
#include "NiagaraSystem.h"
#include "NiagaraComponent.h"
#include "Kismet/GameplayStatics.h"
#include "Components/SphereComponent.h"
#include "Interface/RBossBombInterface.h"

// Sets default values
ARBomb::ARBomb()
{
```

```
    // Set this actor to call Tick() every frame.  You can turn this of
    PrimaryActorTick.bCanEverTick = true;

    Mesh = CreateDefaultSubobject<UStaticMeshComponent>(TEXT("Mesh"));
    RootComponent = Mesh;

    Trigger = CreateDefaultSubobject<USphereComponent>(TEXT("Trigger"))
    Trigger->OnComponentBeginOverlap.AddDynamic(this, &ARBomb::OnBombBe
    Trigger->SetupAttachment(RootComponent);
    Trigger->SetCollisionProfileName(TEXT("RTrigger"));
    Trigger->SetActive(false);

    ProjectileComponent = CreateDefaultSubobject<UProjectileMovementCom
    ProjectileComponent->UpdatedComponent = Mesh;
    ProjectileComponent->InitialSpeed = 1000.f;
    ProjectileComponent->MaxSpeed = 1000.f;
    ProjectileComponent->bRotationFollowsVelocity = true;

    NiagaraComponent = CreateDefaultSubobject<UNiagaraComponent>(TEXT("

    static ConstructorHelpers::FObjectFinder<UNiagaraSystem> FirstFireR
    if (FirstFireRef.Object)
    {
        FirstFire = FirstFireRef.Object;
    }
    static ConstructorHelpers::FObjectFinder<UNiagaraSystem> ExplosionR
    if (ExplosionRef.Object)
    {
        Explosion = ExplosionRef.Object;
    }
    static ConstructorHelpers::FObjectFinder<UNiagaraSystem> NiagaraRef
    if (NiagaraRef.Object)
    {
        GroundFire = NiagaraRef.Object;
    }

    NiagaraComponent->SetupAttachment(RootComponent);
    NiagaraComponent->SetAsset(FirstFire);
    NiagaraComponent->bAutoActivate = true;
}

// Called when the game starts or when spawned
void ARBomb::BeginPlay()
{
    Super::BeginPlay();

    Mesh->OnComponentHit.AddDynamic(this, &ARBomb::OnHit);

}

// Called every frame
```

```
void ARBomb::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);

}

void ARBomb::SetDirection(const FVector& InDirection)
{
    ProjectileComponent->Velocity = InDirection * ProjectileComponent->
}

void ARBomb::OnHit(UPrimitiveComponent* HitComponent, AActor* OtherActo
{
    if (bFlag)
    {
        bFlag = false;
        Mesh->SetSimulatePhysics(false);
        Mesh->SetHiddenInGame(true);
        //Mesh->SetCollisionEnabled(ECollisionEnabled::NoCollision);
        Trigger->SetActive(true);
        NiagaraComponent->SetAsset(Explosion);

        FTimerHandle TimerHandle;
        GetWorld()->GetTimerManager().SetTimer(TimerHandle, [this]()
            {
                NiagaraComponent->SetAsset(GroundFire);
                UE_LOG(LogTemp, Display, TEXT("바닥불"));
            }, 1.5f, false);
    }


}

void ARBomb::OnBombBeginOverlap(UPrimitiveComponent* OverlappedComponen
{
    float Distance = FVector::Distance(GetActorLocation(), OtherActor->
    float Damage = (1 / Distance) * 3000;
    UE_LOG(LogTemp, Display, TEXT("Damage : %f"), Damage);

    IRBossBombInterface* Boss = Cast<IRBossBombInterface>(GetOwner());
    if (Boss)
    {
        Boss->TakeDamageByBomb(OtherActor, Damage);
    }
}
```

- 이때 처음에는 액터에 보스 캐릭터를 인클루드 했었는데 이건 상호 인클루드로 당연히 오류가 나는걸, 난 모르고 했다가 낭패봤다. 에디터가 실행이 안돼 이것도 해결하느라 시간이 많이 걸렸다.

▼ 게임 시스템 관련 기능

▼ 맵 제작

2024년 7월 16일

- 언리얼 에디터에서 애셋을 활용해 맵을 디자인하고 만들었다.

2024년 7월 18일

- 메인메뉴 - 메인 맵 - 엔딩 맵으로 이어지는 맵 구조를 구현했다.

▼ 리스폰

7월 22일

- PlayerController 에서 `GameHasEnded(AActor* EndGameFocus, bool bIsWinner)`

함수를 사용해서 게임이 끝나는 타이밍을 정할 수 있다. 캐릭터의 Dead 함수에서 호출하면 된다.

- 그 후 RestartLevel 함수를 호출해주면 된다.

```
GetWorldTimerManager().SetTimer(ReStartTimer, this,
&APlayerController::RestartLevel, 5.0f);
```

▼ 기믹

2024년 7월 19일

- 특정 아이템을 몇개 이상 획득 시 문이 열리는 기믹 제작

▼ 코드

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

    bCheckkey = false;
    Start = GetActorLocation();
    MoveTime = 4.0f;
}

void ARWall::Tick(float DeltaTime)
{
    Super::Tick(DeltaTime);
    if (bCheckkey)
    {
        FVector Current = GetActorLocation();
        FVector Target = Start + FVector(0.0f, 0.0f, 390.0f);
        float Speed = FVector::Distance(Start, Target) / MoveTime;

        FVector NewLocation = FMath::VInterpConstantTo(Current, Target, Delta
        SetActorLocation(NewLocation);
    }
```

```
}

void ARWall::OnWallBeginOverlap(UPrimitiveComponent* OverlappedComp, AActor*
{
    IRWallInterface* Player = Cast<IRWallInterface>(OtherActor);
    if (Player)
    {
        bCheckkey = Player->CheckKey();
        if (bCheckkey)
        {
            Player->WatchWall();
        }
    }
}
```