



# Forgotten Relic

Fantasy Holic Studio

---

# » A table of Contents

## 1 Introduce

게임 개요, 프로젝트 구성원 소개, 일정관리

## 2 Project Process

제작 과정 소개

## 3 Game Cinematic

게임 시네마틱 영상 시청

## 4 Project Feedback

프로젝트 피드백 및 플레이 영상 시청

# Part 1

## Introduce



# >>> Introduce



## 게임 개요

게임 제목 : Forgotten Relic

개발 엔진 : Unreal Engine 4.26.2

게임 장르 : 3D Back View 액션 중심 RPG

개발 언어 : Unreal C++

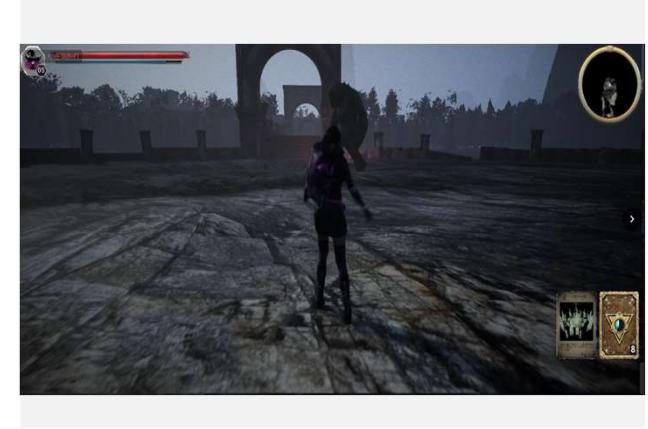
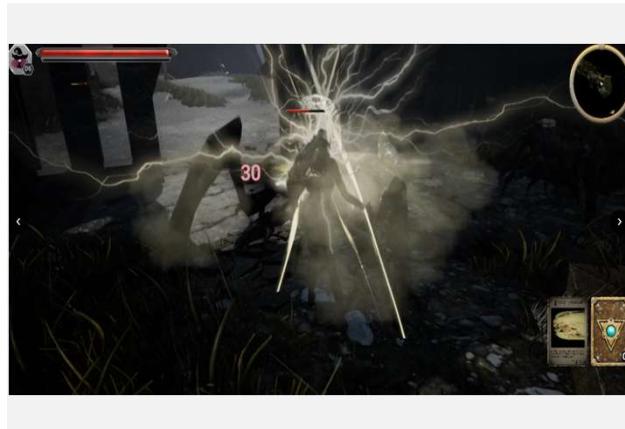
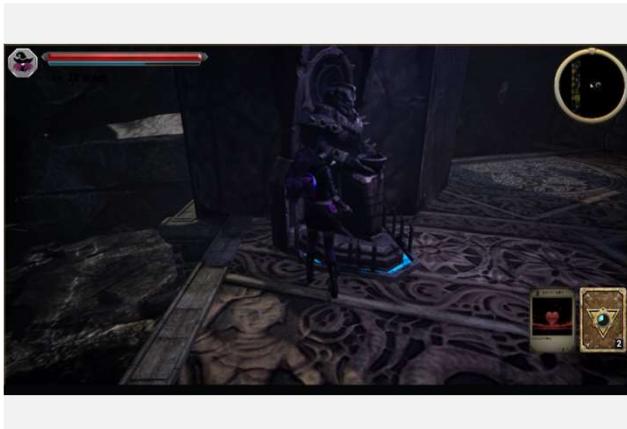
플랫폼 : PC버전(오프라인, 1인 플레이)

플레이 개요 : 황폐화된 마을을 구원할 수 있는  
마을의 유물을 가져간 이종족에  
복수하고 유물을 되찾아라!

# »»» Member Position

Name	Position	Task			Part
박기범	Project Manager	주요 시스템 기획	리소스 수집 & 아트	진행 계획 수립 및 관리	팀장
구자빈	Programer	주요 스킬 모션	게임 이펙트	데미지 외 각종 UI	팀원
박이준	Programer	카드 슬롯제작 성장 시스템	미니맵 등 게임 주요 UI	깃 허브 관리	팀원
이정우	Programer	전투 AI 구성	전투 플로우 기획	깃 허브 관리	팀원
장수미	Lever Designer	에셋 구성 / 최적화	레벨 기획	맵 디자인	팀원
황치문	Contents Designer	캐릭터 애니메이션	플레이어 동선 설계	시네마틱 영상 제작	팀원

# Game Play Flow



맵 탐사 → 카드 획득 → 몬스터 전투 → 카드 스킬 사용 → 캐릭터 성장 → 중간몬스터 처치 → 최종 보스 처치 → 유물 획득

# Production Overview



장르 선정



기획



구현



마무리

0224

게임 주제 및 장르 선정  
역할 배분

0304

게임 제안서 작성  
게임 기획(시스템, 콘텐츠)  
레벨 디자인

0315

상세 기능 구현  
UI 애니메이션  
시네마틱 영상 제작

0324

플레이 및 밸런스 보정(QA)  
버그 수정(QA)  
최종발표 준비

# Part 2

## Project Process

# Development Log

## 7. 파일명 및 변수 규칙

항목 <sup>(3)</sup>	세부내용 <sup>(2)</sup>
기획의도 <sup>(2)</sup>	- 파일명 및 변수 등의 혼동 방지를 위한 규칙을 마련한다. <sup>(3)</sup>
내용 <sup>(2)</sup>	- 일반적으로 스네이크 표기법을 따르되, 변수에 대해서는 파스칼 표기법을 따른다. <sup>(3)</sup>
항목 <sup>(3)</sup>	세부내용 <sup>(2)</sup>
변수 <sup>(3)</sup>	- C++로 진행한다(애니메이션의 경우 비허이비어 트리 구성을 위해 BP로 작업한다) <sup>(4)</sup> - 변수는 파스칼 표기법을 따른다. (첫 문자를 대문자로, 띄어쓰기 없이 대문자로 구분) <sup>(3)</sup>
문서명 <sup>(2)</sup>	- 프로젝트명_문서종류_작성자이름_버전.확장자(스네이크 표기법을 따름) <sup>(3)</sup>
파일명 <sup>(2)</sup>	- 사운드: Sound_종류_적용대상_파일명 (종류: BGM, BGS, SE, ME) <sup>(4)</sup> - 애니메이션: Ani_적용대상_파일명 <sup>(4)</sup> - 이미지: Img_종류_파일명 <sup>(4)</sup> - 이펙트: Eft_종류_적용대상_파일명 <sup>(4)</sup>

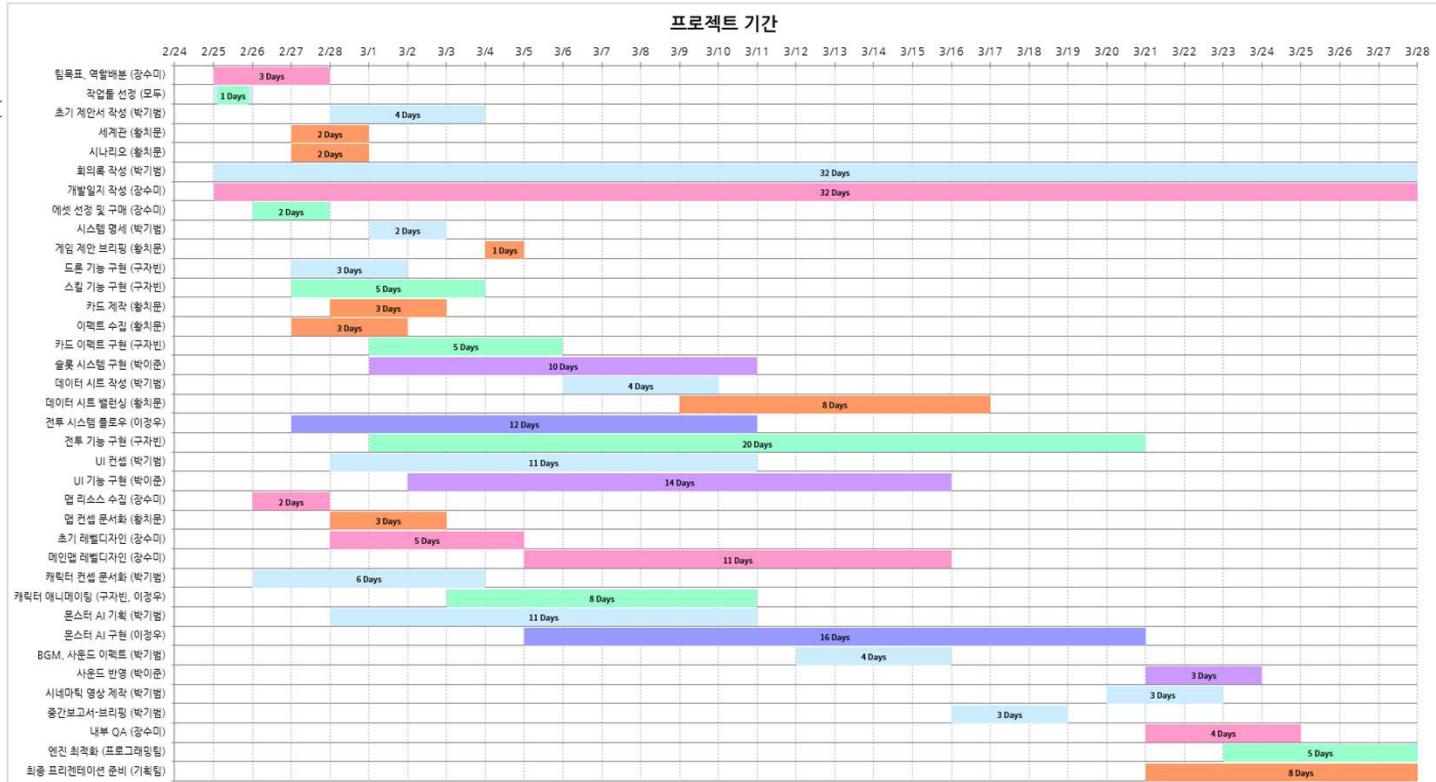
# 게임 제작 전체 과정을 기록한 프로젝트 진행 로그입니다.

파일 충돌 방지와 혼란을 방지하기 위해서 사전에 파일명과 변수 규칙을 정하고 공유하였습니다

# Gantt Chart

## 프로젝트 기간

시작일	마감일	기간	세로축 표시형식	항목	담당자
2022-02-25	2022-02-27	3	팀폭포, 역할배분 (장수미)	팀폭포, 역할배분	장수미
2022-02-25	2022-02-25	1	작업을 선정 (모두)	작업을 선정	모두
2022-02-28	2022-03-03	4	초기 제안서 작성 (박기범)	초기 제안서 작성	박기범
2022-02-27	2022-02-28	2	세계관 (황지운)	세계관	황지운
2022-02-27	2022-02-28	2	시나리오 (황지운)	시나리오	황지운
2022-02-25	2022-03-28	32	회원통 작성 (박기범)	회원통 작성	박기범
2022-02-25	2022-03-28	32	개발일지 작성 (장수미)	개발일지 작성	장수미
2022-02-26	2022-02-27	2	에셋 선정 및 구매 (장수미)	에셋 선정 및 구매	장수미
2022-03-01	2022-03-02	2	시스템 망세 (박기범)	시스템 망세	박기범
2022-03-04	2022-03-04	1	게임 제작 보드링 (황지운)	게임 제작 보드링	황지운
2022-02-27	2022-03-01	3	드론 기능 구현 (구자빈)	드론 기능 구현	구자빈
2022-02-27	2022-03-03	5	스킬 기능 구현 (구자빈)	스킬 기능 구현	구자빈
2022-02-28	2022-03-02	3	카드 제작 (황지운)	카드 제작	황지운
2022-02-27	2022-03-01	3	이펙트 수집 (황지운)	이펙트 수집	황지운
2022-03-01	2022-03-05	5	카드 이펙트 구현 (구자빈)	카드 이펙트 구현	구자빈
2022-03-01	2022-03-10	10	슬롯 시스템 구현 (박이준)	슬롯 시스템 구현	박이준
2022-03-06	2022-03-09	4	데이터 시트 작성 (박기범)	데이터 시트 작성	박기범
2022-03-09	2022-03-16	8	데이터 시트 벌행상 (황지운)	데이터 시트 벌행상	황지운
2022-02-27	2022-03-10	12	전투 시스템 플로우 (이정우)	전투 시스템 플로우	이정우
2022-03-01	2022-03-20	20	전투 기능 구현 (구자빈)	전투 기능 구현	구자빈
2022-02-28	2022-03-10	11	UI 컨셉 (박기범)	UI 컨셉	박기범
2022-03-02	2022-03-15	14	UI 기능 구현 (박이준)	UI 기능 구현	박이준
2022-02-26	2022-02-27	2	멀 리소스 수집 (장수미)	멀 리소스 수집	장수미
2022-02-28	2022-03-02	3	멀 컨셉 문서화 (황지운)	멀 컨셉 문서화	황지운
2022-02-28	2022-03-04	5	초기 레벨디자인 (장수미)	초기 레벨디자인	장수미
2022-03-05	2022-03-15	11	메인맵 레벨디자인 (장수미)	메인맵 레벨디자인	장수미
2022-02-26	2022-03-03	6	캐릭터 컨셉 문서화 (박기범)	캐릭터 컨셉 문서화	박기범
2022-03-03	2022-03-10	8	캐릭터 애니메이팅 (구자빈, 이정우)	캐릭터 애니메이팅	구자빈, 이정우
2022-02-28	2022-03-10	11	몬스터 AI 기획 (박기범)	몬스터 AI 기획	박기범
2022-03-05	2022-03-20	16	몬스터 AI 구현 (이정우)	몬스터 AI 구현	이정우
2022-03-12	2022-03-15	4	BGM, 사운드 이펙트 (박기범)	BGM, 사운드 이펙트	박기범
2022-03-21	2022-03-23	3	사운드 반영 (박이준)	사운드 반영	박이준
2022-03-20	2022-03-22	3	시네마틱 영상 제작 (박기범)	시네마틱 영상 제작	박기범
2022-03-16	2022-03-18	3	중간보고서-보리핑 (박기범)	중간보고서-보리핑	박기범
2022-03-21	2022-03-24	4	내부 QA (장수미)	내부 QA	장수미
2022-03-23	2022-03-27	5	엔진 최적화 (프로그래밍팀)	엔진 최적화	프로그래밍팀
2022-03-21	2022-03-28	8	최종 프리젠테이션 준비 (기획팀)	최종 프리젠테이션 준비	기획팀



게임 개발의 항목별 상세 일정은 엑셀로 간트 차트를 만들어서 관리하였습니다.

# Daily Log & The minutes of a meeting

이름 ↓	소유자	마지막으로 수정한 날짜	파일 크기
W 회의록 3월 25일.docx	나	2022. 3. 25. 나	17KB
W 회의록 3월 24일.docx	나	2022. 3. 24. 나	17KB
W 회의록 3월 23일.docx	나	2022. 3. 23. 나	18KB
W 회의록 3월 22일.docx	나	2022. 3. 23. 나	18KB
W 회의록 3월 21일.docx	나	2022. 3. 21. 나	131KB
W 회의록 3월 19일.docx	나	2022. 3. 18. 나	18KB
W 회의록 3월 17일.docx	나	2022. 3. 17. 나	17KB
W 회의록 3월 16일.docx	나	2022. 3. 16. 나	17KB
W 회의록 3월 15일.docx	나	2022. 3. 15. 나	16KB
W 회의록 3월 14일.docx	나	2022. 3. 14. 나	20KB
W 회의록 3월 11일.docx	나	2022. 3. 18. 나	17KB
W 회의록 3월 10일.docx	나	2022. 3. 10. 나	19KB
W 회의록 3월 8일.docx	나	2022. 3. 8. 나	18KB
W 회의록 3월 7일.docx	나	2022. 3. 7. 나	16KB
W 회의록 3월 4일.docx	나	2022. 3. 4. 나	17KB
W 회의록 3월 3일.docx	나	2022. 3. 4. 나	17KB
W 회의록 3월 2일.docx	나	2022. 3. 2. 나	17KB
W 회의록 2월 28일.docx	나	2022. 2. 28. 나	20KB
W 회의록 2월 27일.docx	나	2022. 2. 28. 나	16KB
W 회의록 2월 25일.docx	나	2022. 2. 25. 나	17KB
W 회의록 2월 24일.docx	나	2022. 2. 25. 나	16KB
W 회의록 2월 23일.docx	나	2022. 2. 25. 나	14KB

이름 ↓	소유자	마지막으로 수정한 날짜	파일 크기
E 20220325_개발일지	나	2022. 3. 25. 나	2KB
E 20220324_개발일지	SUMI	2022. 3. 28. 나	2KB
E 20220323_개발일지	SUMI	2022. 3. 28. 나	3KB
E 20220322_개발일지	SUMI	2022. 3. 28. 나	2KB
E 20220321_개발일지	나	2022. 3. 28. 나	2KB
E 20220318_개발일지	SUMI	2022. 3. 28. 나	1KB
E 20220317_개발일지	SUMI	2022. 3. 28. 나	1KB
E 20220316_개발일지	SUMI	2022. 3. 28. 나	1KB
E 20220315_개발일지	SUMI	2022. 3. 28. 나	1KB
E 20220314_개발일지	SUMI	2022. 3. 28. 나	2KB
E 20220311_개발일지	SUMI	2022. 3. 28. 나	1KB
E 20220310_개발일지	SUMI	2022. 3. 28. 나	1KB
E 20220308_개발일지	SUMI	2022. 3. 28. 나	2KB
E 20220307_개발일지	SUMI	2022. 3. 27. 나	2KB
E 20220304_개발일지	SUMI	2022. 3. 27. 나	2KB
E 20220303_개발일지	SUMI	2022. 3. 28. 나	2KB
E 20220302_개발일지	SUMI	2022. 3. 28. 나	2KB
E 20220228_개발일지	SUMI	2022. 3. 28. 나	2KB

개발일지 (2022-03-07)				
이름	계획	활동 사항	비고	
구자빈	1. 마우스 커서 위치로 스킬 발동. 2. 열려있던 창 개선	1. 마우스 커서 위치로 스킬 발동. 2. 열려있던 창 개선 3. 마우스 위치 초기화; 스킬 사용시에만 표현	프로그램	
개발일지 (2022-03-08)				
이름	계획	활동 사항	비고	
박기범	1. 몬스터 스텝 카드 2. 몬스터 에셋 할 3. 애니메이션 명세 4. 기획팀 회의	1. 2차 제작 공유 회의 2. 드론에 사용될 에시 카드 3. 구르기 모션 최적화	1. 스킬 시스템 구현 2. 캐릭터 모션 추가	
박이준	카드 리스트 시스템 완성 카드 확대 창 개설	카드 리스트 시스 템 확장 창 개설 - 개선사항 : UI 구	1. 월드 몬스터 예셋 확보 2. 몬스터 대이드시트 작성 3. 회의록 및 기록서 작성 4. 기획팀 회의	
이정우	1. AI 시스템 구현	1. 각 몬스터에 따라 애니메이션 구	1. 2차 제작 공유 회의 및 회의록 2. 몬스터 대이드시트 작성 3. 회의록 및 기록서 작성 4. 기획팀 회의	
박이준	1. 레벨 디자인 (조기 앱 디자인)	1. 레벨 디자인 (조기 앱 디자인)	1. 내 배치 확인, 디자인적인 요소 추가 2. 연리얼 뷰포드판의 C++화	
회의록(30일차)				
일시	작성자	작성자	유 희의 날짜 메시지 [자 수정]	
2022년 3월 24일	박기범	박기범	1. 레벨 소스 배치 2. 레벨 최적화	
참석자	구자빈, 박기범, 박이준, 이정우, 장수미 // 황지문- 휴가			
안건	최종 발표 일정 관련, 몬스터 배치 관련			
회의 내용	<p>&lt;최종 발표 일정 관련&gt;</p> <p>정기 주도 하여 상세 일정을 공유함 발표 자료는 26일 달달 11시 30분까지 구글드라이브에 업로드 유기자주인은 2022년 3월 25일 원 상호평가가 진행될 때 구글 문서로 실무조사 6시까지 제출 (제출일과 출판일은 2022년 3월 26일로 예상됨)</p> <p>도록색의 날짜를 (개인행마상) 친증발표자로, 회의 기록자는 MLP 사이트에 월요일 저녁 6시 30분까지 청중(팀원/팀장)에 티损 가능</p> <p>-기획팀은 주말인 경우 프로젝트 관련 기획서를 집중해서 다듬기 -발표일 티损 가능 일 17시 10분까지 제출을 완료하여 험(월수는 아니나 다른 회의의 경우 영향을 줄 수 있으므로 미리 예상되어 관리)</p> <p>-발표 구상의 경우 발표 10분 영상자료(리얼미션, 시네마25프로, 시네마25프로)→제작시연 전부보기(제작시작부터 투토리얼 진행 퍼시픽 워크숍에서 상당 석상 카드 회화까지) 시연 거미 몬스터와 같은 적극적인 보완 필요)</p> <p>-현재 보스까지 청자는 되어 있으나 모든 것을 최적화하는 것은 시간적으로 어려움이 있으니, 전반적 플랫폼과의 적자율에 집중하기 -박이준은 몬비(내일암 마루온) 및 뮤비(일 시스템) 설계 -구자빈 각종 버그 고정, 사용도 살피 -이정우- 몬스터 배치 보완 -장수미- 몬스터 보강 및 기획서 일부 보완 -황지문- 회의록 보완 -박기범- 최종발표 준비+기획서 일부 보완(설정시스템, 성장시스템)</p>			1. 다양한 종류의 몬스터 예셋 파악 2. 기획서 작성 3. 기획팀 회의

하루 일과 시작과 동시에, 회의록을 상시 작성하고, 일과 종료 후에 각자 개발일지를 작성하여 향후 일정의 방향을 다시 잡았습니다.

# ▶▶▶ Production Stage



# ▶▶▶ Introduction to Production parts

기획  
GAME DESIGN

전투시스템, 조작시스템, 성장시스템, 행운시스템  
카드 콘텐츠, 레벨콘텐츠, 인벤토리, 게임플로우,  
전투모션, UI 기획, 스테이터스 구성 등

박기범, 황치문

세부내용  
- 몬스터를 명세한다.  
- 몬스터의 체력, 플레이어의 체력을 설정한다.  
- 몬스터의 스폷을 설명한다.(이동 속도, 공격력 등)  
- 주집 가능한 카드와 기능 동작(스킬 동작법)을 설명한다.  
- 카드 DataSheet를 설명한다.  
- 게임 플레이 조작법을 설명한다.  
- 사용자가 게임을 즐기는 환경에 맞게 세팅할 수 있는 내용을 설명한다.  
- 카드에서 카드를 획득하는 방법과 인벤토리 슬롯(카드 창작 UI)을 설명한다.  
- 몬스터의 체력에 대한 시스템을 설명한다.

레벨디자인  
LEVEL DESIGN

구조도 작성, 캐릭터 이동 방향 설계,  
맵 구성, 액터 배치, 애니메이션, 콜라이더,  
라이팅 빌드, 맵 확장, 건물 배치, 규모 조절 등

장수미

개발  
DEVELOPMENT

몬스터 AI, 전투 기능, 스킬 효과, 캐릭터 이동, 데미지 표시,  
미니맵, 카드 애니 콜리저, 호버링, 애니메이션, 이벤트 구현,  
아펙트 구현, 게임 최적화

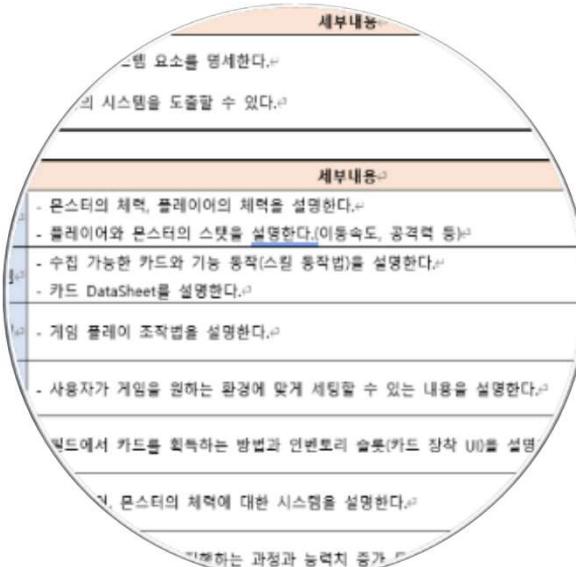
구자빈, 박이준, 이정우

```
this actor to call Tick();
bCanEverTick = true;
CollisionComp = CreateDefaultSubobject(UParticleSystem)(CollisionComp);
CollisionComp->InitBoxExtent(FVector(50, 50, 50));
CollisionComp->SetWalkableSlopeOverride(0.0f);
CollisionComp->CanCharacterStepUpOn = ECC_Walkable;
RootComponent = CollisionComp;

static ConstructorHelpers::FObjectFinder<UParticleSystem> PSC = CreateDefaultSubobject<UParticleSystem>(PSC);
PSC->SetTemplate(PS.Object);
PSC->SetupAttachment(CollisionComp);
InitialLifeSpan = 1.0f;

ACharacter = Cast<AMainCharacter>(GetOwner());
OnComponentBeginOverlap(CollisionComp, OnComponentBeginOverlap);
```

# 3 Part- first “Game Design”



GAME DESIGN

기획서 내용 보기



LEVEL DESIGN

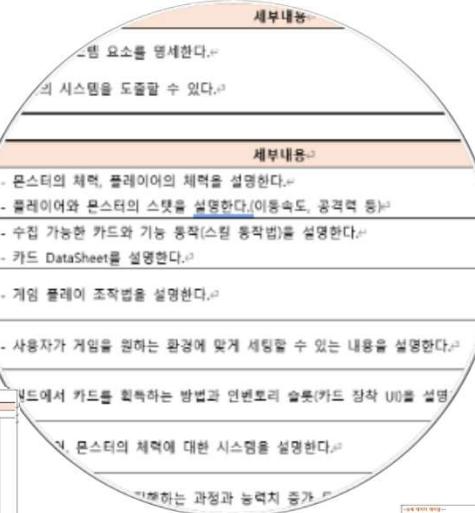
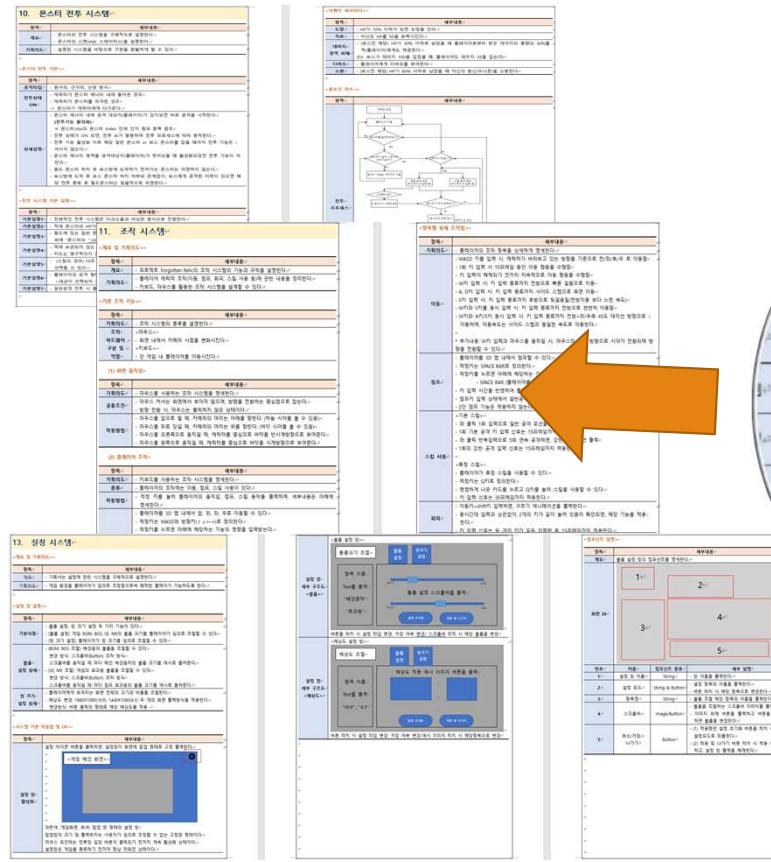
레벨디자인 문서 보기



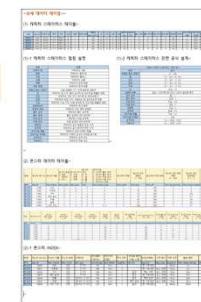
DEVELOPMENT

개발 코드 확인 하기

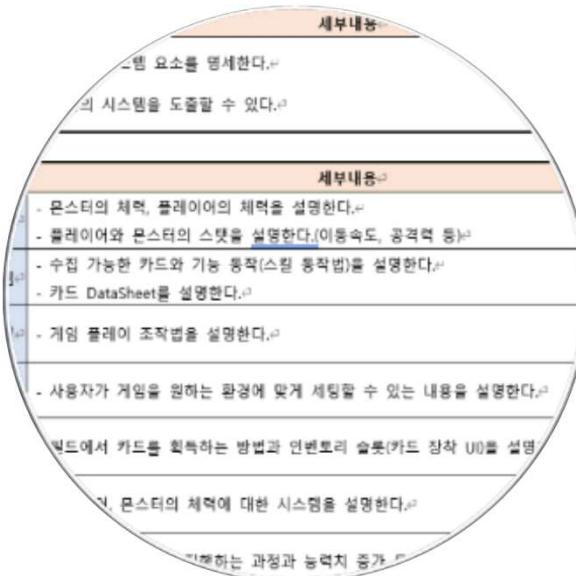
# 3 Part- first “Game Design”



# GAME DESIGN



# 3 Part- Second “Level Design”



GAME DESIGN

기획서 내용 보기



LEVEL DESIGN

레벨디자인 문서 보기



DEVELOPMENT

개발 코드 확인 하기

# 3 Part- Second “Level Design”



LEVEL DESIGN

## 1. 개요

구분	내용
구별ID	Level_ForgottenRelic
이름	遗忘遗迹
설정	중세 서부로 유래하는 석조 건물과 폐허
여러 구분	작업에서 가끔 사이드 캐릭터와 산악
구조 및 디자인	3D
플레이의 모니터링 규칙	단단한 풀스피웨이 전선을 탈출 상황 화면 표시에 놓인 배경에서 강제로 카드를 광활화하여 상당 수의 카드를 사용하는 경우
구조 및 디자인	6514631
사용 도구	Unreal Engine(4.26), Adobe Photoshop(2.0)
문서 편집	게임, Forgotten Relic의 각별 설정을 참조하여 전설에 대한 이해를 돕는다. 방의 구조 및 리도를 설명하여 등의 기획에 기여한다.

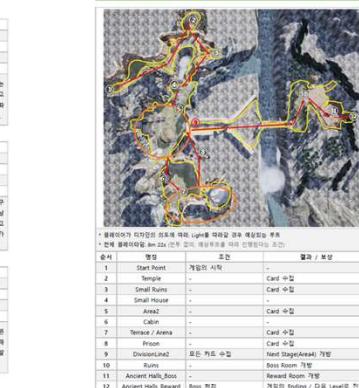
## 2. 의도 및 목표

의도	목표
소규모 세부 오픈 월드	6514631 사이즈의 레벨에 다양한 지형과 오브젝트를 세워 꾸며지는 는 의미로 전달된다. 또한 플레이어가 주간을 벗어나지 않는 환경에서 자주 찾은 모험을 즐길 수 있도록 대부분의 지형을 이용 가능하게 만들었다.
소품류와 전투방식	온갖 곤충과 벌레 같은 폐기물로 적을 파괴거나, 혹은 한 도시 막강하고 전투를 벌여보도록 한다.
플레이타입 확보	플레이어는 반드시 지나야 하는 지점을 설정하고 해당 지점으로 가는 길목을 충족시켜 적의 약점을 활용하는 유트로를 만든다.
스토리 / 분위기 전달	도도한 고대의 문화를 느낄 수 있는 아름다운 분위기를 제공한다. 대부분의 낙차들은 반쯤 펼쳐져 험준한 폐허로 만들어 주제를 유학자에게 전달하는 듯한 느낌을 준다.

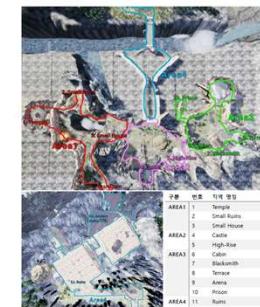
## 4-4-1. 주요 지역



## 5. 플레이 동선



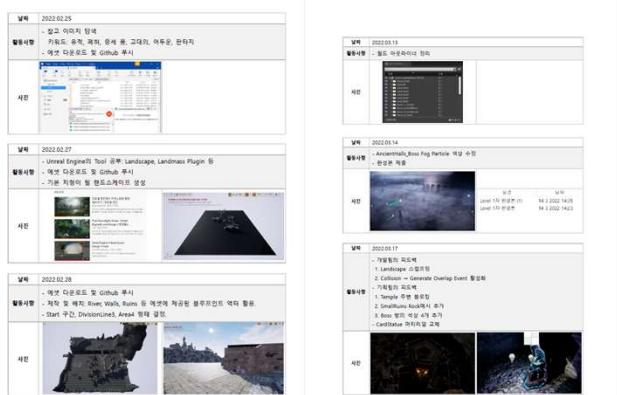
## 4. 레벨 구성



## 4-1-1. 주요 지역



## 7. 진행 과정



# »»» 3 Part- Second “Level Design”

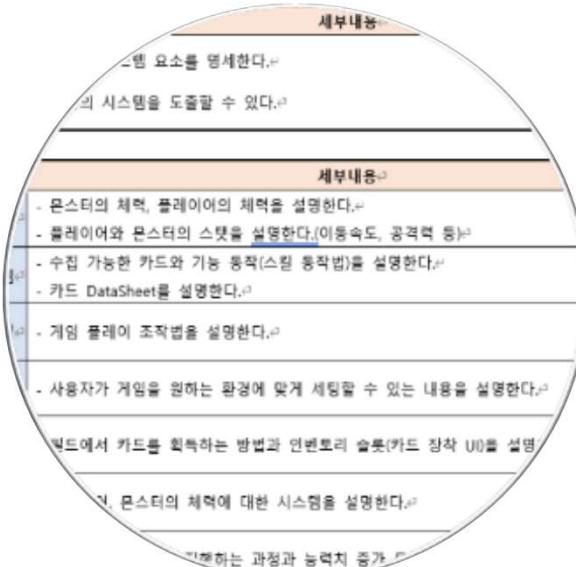


LEVEL DESIGN



완성 레벨

# 3 Part- Third “Game Development”



GAME DESIGN

기획서 내용 보기



LEVEL DESIGN

레벨디자인 문서 보기



DEVELOPMENT

개발 코드 확인 하기

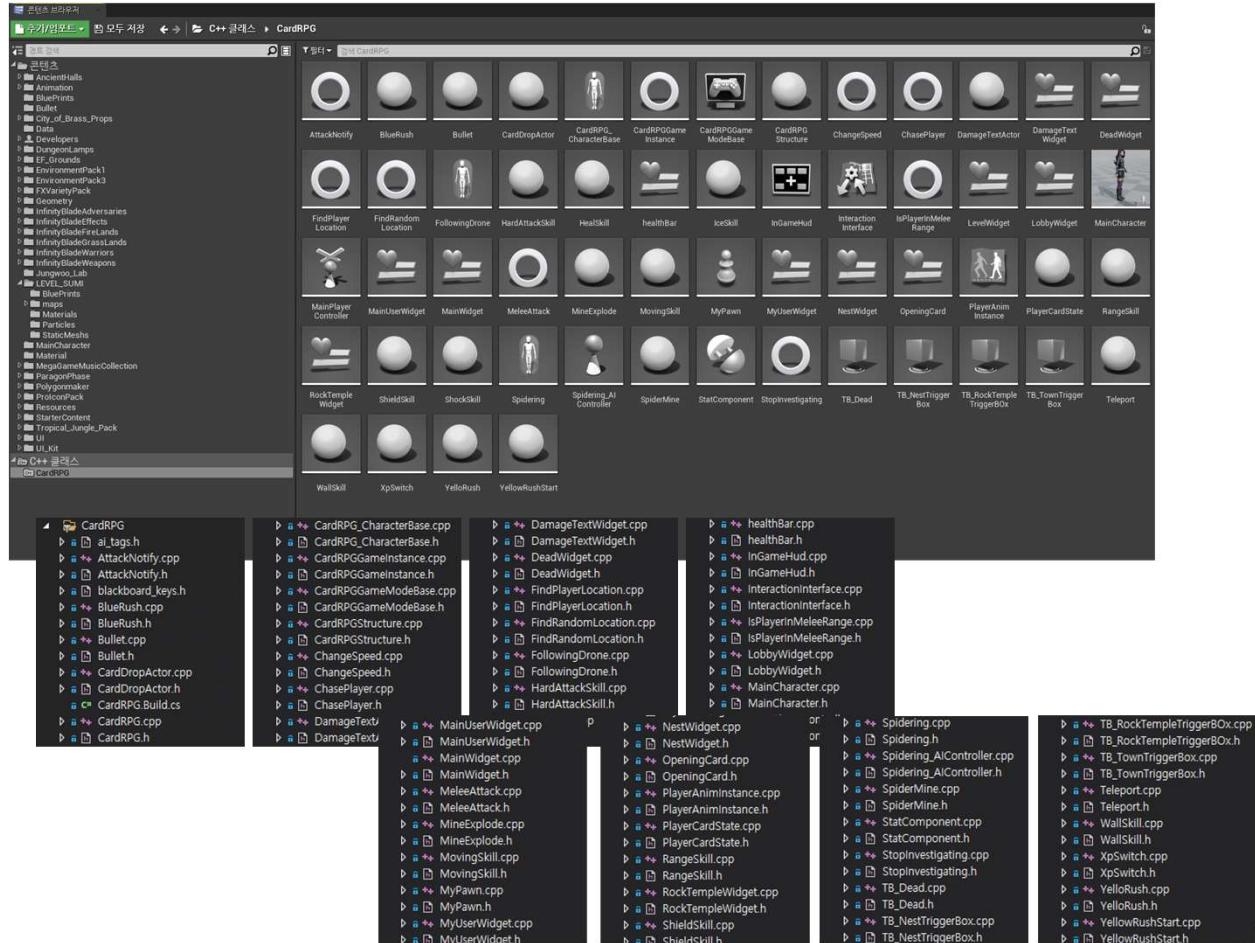
# 3 Part- Third “Game Development”

```
    // this actor to call Tick()
PrimaryActorTick.bCanEverTick = true;
CollisionComp = CreateDefaultSubobject();
CollisionComp->InitBoxExtent(FVector());
CollisionComp->SetWalkableSlopeOverride(0.0f);
CollisionComp->CanCharacterStepUpOn = ECC_Walkable;
RootComponent = CollisionComp;

static ConstructorHelpers::FObjectFinder
PSC = CreateDefaultSubobject(PS.Object);
PSC->SetTemplate(PS.Object);
PSC->SetupAttachment(CollisionComp);
InitialLifeSpan = 1.0f;

AInCharacter = Cast(GetOwner());
CollisionComp->OnComponentBeginOverlap(
    AInCharacter, CollisionEvent, CollisionData);
```

DEVELOPMENT



# 3 Part- Third “Game Development”

```
        // this actor to call Tick()
    PrimaryActorTick.bCanEverTick = true;
    CollisionComp = CreateDefaultSubobject();
    CollisionComp->InitBoxExtent(FVector(100, 100, 100));
    CollisionComp->SetWalkableSlopeOverride(0.0f);
    CollisionComp->CanCharacterStepUpOn = ECC_Walkable;
    RootComponent = CollisionComp;

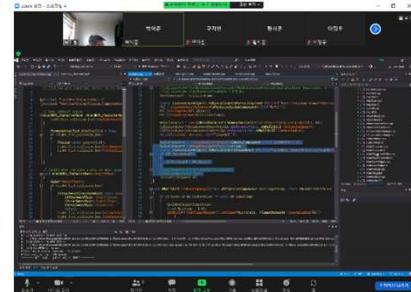
    static ConstructorHelpers::FObjectFinder
    PSC = CreateDefaultSubobject(PS.Object);
    PSC->SetTemplate(PS.Object);
    PSC->SetupAttachment(CollisionComp);
    InitialLifeSpan = 1.0f;

    AMainCharacter* InCharacter = Cast(GetOwner());
    CollisionComp->OnComponentBeginOverlap(InCharacter, this, FCollisionEventArgs());
}
```

DEVELOPMENT

```
void AMainCharacter::OnComponentBeginOverlap(UPrimitiveComponent* OverlappedComponent, AActor* OtherActor, UPrimitiveComponent* OtherComp, FVector NormalImpulse, const FHitResult& Hit)
{
    if (OtherActor != this)
    {
        if (OtherActor->IsA(AMainCharacter::StaticClass()))
        {
            AMainCharacter* InCharacter = Cast(OtherActor);
            if (InCharacter)
            {
                if (InCharacter->Health <= 0.0f)
                {
                    return;
                }
            }
        }
    }
}
```

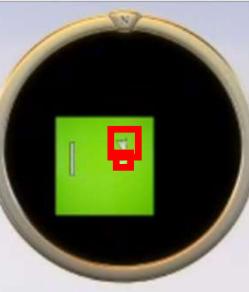
전투  
AI



사운드 삽입

```
void UAAttackNotify::NotifyBegin(USkeletalMeshComponent* Mesh, UAnimSequenceBase* Animation, float TotalDuration)
{
    if (Mesh == GetOwner())
    {
        if (AMainCharacter* character = Cast(Mesh->GetOwner()))
        {
            character->Attack_Start();
        }
    }
}

void UAAttackNotify::NotifyEnd(USkeletalMeshComponent* Mesh, UAnimSequenceBase* Animation)
{
    if (Mesh == GetOwner())
    {
        if (AMainCharacter* character = Cast(Mesh->GetOwner()))
        {
            character->Attack_End();
        }
    }
}
```



미니맵 구현

```
void UAAttackNotify::NotifyBegin(USkeletalMeshComponent* Mesh, UAnimSequenceBase* Animation, float TotalDuration)
{
    if (Mesh == GetOwner())
    {
        if (AMainCharacter* character = Cast(Mesh->GetOwner()))
        {
            character->Attack_Start();
        }
    }
}

void UAAttackNotify::NotifyEnd(USkeletalMeshComponent* Mesh, UAnimSequenceBase* Animation)
{
    if (Mesh == GetOwner())
    {
        if (AMainCharacter* character = Cast(Mesh->GetOwner()))
        {
            character->Attack_End();
        }
    }
}
```

일반 공격 - 평타

일반 공격 - Hard Attack

# 3 Part- Third “Game Development”

UI triggerBox

```

        // this actor to call Tick()
        PrimaryActorTick.bCanEverTick = true;
        CollisionComp = CreateDefaultSubobject(TEXT("CollisionComp"));
        CollisionComp->InitBoxExtent(FVector(100, 100, 100));
        CollisionComp->SetWalkableSlopeOverride(0.0f);
        CollisionComp->CanCharacterStepUpOn = ECC_Walkable;
        RootComponent = CollisionComp;

        static ConstructorHelpers::FObjectFinder<UParticleSystem> PSC;
        PSC = CreateDefaultSubobject<UParticleSystem>(PS.Object);
        PSC->SetTemplate(PS.Object);
        PSC->SetupAttachment(CollisionComp);
        InitialLifeSpan = 1.0f;

        MainCharacter = Cast<AMainCharacter>(GetOwner());
        CollisionComp->OnComponentBeginOverlap(this, MainCharacter, FCollisionEvent::CreateInitial(A));
    }

    void AUITriggerBox::OnOverlapBegin(class AActor* OtherActor, class Actor OtherActor)
    {
        if (MainCharacter)
        {
            MainCharacter->GetMainCharacter();
            MainCharacter->GetMainCharacter();
            MainCharacter->GetMainCharacter();
            MainCharacter->GetMainCharacter();
            MainCharacter->GetMainCharacter();
        }
    }

```

DEVELOPMENT

Damage UI

```

        // Set this actor to call Tick()
        PrimaryActorTick.bCanEverTick = true;
        CollisionComp = CreateDefaultSubobject<UParticleSystem>(TEXT("CollisionComp"));
        CollisionComp->InitBoxExtent(FVector(100, 100, 100));
        CollisionComp->SetWalkableSlopeOverride(0.0f);
        CollisionComp->CanCharacterStepUpOn = ECC_Walkable;
        RootComponent = CollisionComp;

        static ConstructorHelpers::FObjectFinder<UParticleSystem> PSC;
        PSC = CreateDefaultSubobject<UParticleSystem>(PS.Object);
        PSC->SetTemplate(PS.Object);
        PSC->SetupAttachment(CollisionComp);
        InitialLifeSpan = 1.0f;

        // Called when the game starts or when spawned
        void AUIManagerActor::BeginPlay()
        {
            Super::BeginPlay();
            MainCharacter = Cast<AMainCharacter>(GetOwner());
        }

        // Called every frame
        void AUIManagerActor::Tick(float DeltaTime)
        {
            Super::Tick(DeltaTime);
        }
    }

    void AUIManagerActor::OnOverlapBegin(class AActor* OtherActor, class Actor OtherActor)
    {
        if (OtherActor == MainCharacter && OtherActor == MainCharacter)
        {
            if (MainCharacter)
            {
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
            }
        }
    }

```



XP Actor

```

        // Set this actor to call Tick()
        PrimaryActorTick.bCanEverTick = true;
        CollisionComp = CreateDefaultSubobject<UParticleSystem>(TEXT("CollisionComp"));
        CollisionComp->InitBoxExtent(FVector(100, 100, 100));
        CollisionComp->SetWalkableSlopeOverride(0.0f);
        CollisionComp->CanCharacterStepUpOn = ECC_Walkable;
        RootComponent = CollisionComp;

        static ConstructorHelpers::FObjectFinder<UParticleSystem> PSC;
        PSC = CreateDefaultSubobject<UParticleSystem>(PS.Object);
        PSC->SetTemplate(PS.Object);
        PSC->SetupAttachment(CollisionComp);
        InitialLifeSpan = 1.0f;

        MainCharacter = Cast<AMainCharacter>(GetOwner());
    }

    void AXPActor::OnOverlapBegin(class AActor* OtherActor, class Actor OtherActor)
    {
        if (OtherActor == MainCharacter && OtherActor == MainCharacter)
        {
            if (MainCharacter)
            {
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
            }
        }
    }

```



UI triggerBox

```

        // Set this actor to call Tick()
        PrimaryActorTick.bCanEverTick = true;
        CollisionComp = CreateDefaultSubobject<UParticleSystem>(TEXT("CollisionComp"));
        CollisionComp->InitBoxExtent(FVector(100, 100, 100));
        CollisionComp->SetWalkableSlopeOverride(0.0f);
        CollisionComp->CanCharacterStepUpOn = ECC_Walkable;
        RootComponent = CollisionComp;

        static ConstructorHelpers::FObjectFinder<UParticleSystem> PSC;
        PSC = CreateDefaultSubobject<UParticleSystem>(PS.Object);
        PSC->SetTemplate(PS.Object);
        PSC->SetupAttachment(CollisionComp);
        InitialLifeSpan = 1.0f;

        MainCharacter = Cast<AMainCharacter>(GetOwner());
    }

    void AUIManagerActor::OnOverlapBegin(class AActor* OtherActor, class Actor OtherActor)
    {
        if (OtherActor == MainCharacter && OtherActor == MainCharacter)
        {
            if (MainCharacter)
            {
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
            }
        }
    }

```



Damage UI

```

        // Set this actor to call Tick()
        PrimaryActorTick.bCanEverTick = true;
        CollisionComp = CreateDefaultSubobject<UParticleSystem>(TEXT("CollisionComp"));
        CollisionComp->InitBoxExtent(FVector(100, 100, 100));
        CollisionComp->SetWalkableSlopeOverride(0.0f);
        CollisionComp->CanCharacterStepUpOn = ECC_Walkable;
        RootComponent = CollisionComp;

        static ConstructorHelpers::FObjectFinder<UParticleSystem> PSC;
        PSC = CreateDefaultSubobject<UParticleSystem>(PS.Object);
        PSC->SetTemplate(PS.Object);
        PSC->SetupAttachment(CollisionComp);
        InitialLifeSpan = 1.0f;

        MainCharacter = Cast<AMainCharacter>(GetOwner());
    }

    void AUIManagerActor::OnOverlapBegin(class AActor* OtherActor, class Actor OtherActor)
    {
        if (OtherActor == MainCharacter && OtherActor == MainCharacter)
        {
            if (MainCharacter)
            {
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
                MainCharacter->GetMainCharacter();
            }
        }
    }

```



# Part 3

## Cinematic Video



# Part 4

## Project Feedback



# Project Feedback

전체적인 게임의 완성도가 타 RPG보다 높지는 않았습니다.

이 점은 아쉬운 점으로 평가할 수 있겠으나, 한 달이라는 시간 내에 RPG게임의 일부분을 개발한 점은 충분히 만족할만 한 결과였습니다.

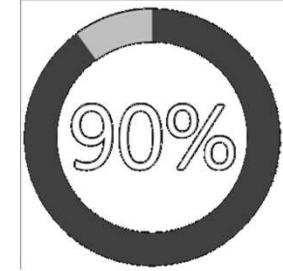
개발 과정에 있어서도 블루프린트로 제작하여 효율을 극대화 할 수 있었지만, 최적화 문제와 개발 역량을 키우고자 하는 목표 도달을 위해 블루프린트를 포기하고, 언리얼 C++로 코딩하면서 보다 새로운 경험을 할 수 있었습니다.

팀원들의 역할 분배가 고루 잘 되었고,

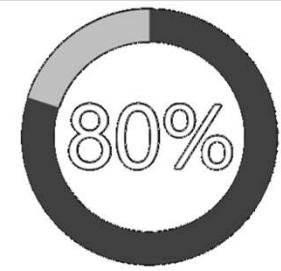
향후 실무에서 더욱 발전된 게임 개발 역량을 보여줄 수 있을 것 같아 좋았습니다.

팀원들의 역량을 과대평가한 부분과 오히려 과소평가한 부분이 있어 이런 부분들을 더 잘 조절하였다면 더 좋았을 텐데 하는 아쉬움도 있었습니다.

RPG게임 자체에 대한 이해도가 증가하였고, 게임을 완성시키기까지 일련의 과정을 배우고 정리할 수 있었습니다.



<프로젝트 진행도>



<프로젝트 완성도>

성장 계획  
Upgrade Plan

백업 시스템 구축  
- 자료 손실이 발생하지 않도록 자주 백업하는 습관이 필요함

프로젝트 관리 시스템 구축  
- 파일을 관리하고 회의 내용을 기록하고, 일정을 관리해주는 사람의 역할이 반드시 필요

다양한 장르의 게임 개발을 통해 역량 강화  
- RPG 외에도 캐주얼게임, 액션게임, 방탈출, FPS, 레이싱 등 다양한 장르의 게임 개발 경험

언리얼 엔진 숙련도 향상

RPG게임 자체에 대한 이해도 향상

C++ 코딩 경험을 통한 개발 역량 향상

제작 경험(일정 설계를 잘 할 수 있을 것)

제작에 있어서 꼭 필요한 아트팀의 중요성

# Thank You