PORTFOLIO

버그헌터 클라이언트 대체불가능한

근면성실 언제나 성장중

김재완

CONTACT

PHONE : 010 9471 3449 MAIL : dkwl635@naver.com

GitHub

https://github.com/dkwl635

이력사항

김재완 1999.03.17

클라이언트 프로그래머

010-9471-3449

dkwl635@naver.com

서울 강서구 방화동

학력사항

2015.03~2018.02	강서공업고등학교 친환경에너지화학과

경력사항

2022.09~2023.11	IOI 게임즈 개발팀 사원	모바일 2D 게임 게발 팀 : 프로그래머(1), 아트(1), 기획(1) 업무 : 게임 프로토타입의 기본 전투 로직, 상점, 데이터 관리, 구글 로그인 연동
2018.02~2019.01	(재)환경보건 기술연구원 사원	현장 대기 샘플링 업무 수행

교육내용

2024.12 ~ 현재	(도봉SW2기)언리얼엔진 생성형 AI를활용한 게임 개발자 양성과정
2024.02 ~ 2024.08	[언리얼엔진5]게임 클라이언트 & 메타버스 프로그래머 양성과정
2021.09 ~ 2022.05	(게임콘텐츠제작)게임 프로그래밍 개발자 양성 과정

기술스펙

UNREAL ENGINE

- C++, 블루프린트를 사용한 프로젝트 진행
- 언리얼 네트워크 RPC를 활용한 멀티 플레이 프로젝트 진행
- BT를 사용한 보스및 일반몬스터 제작
- UMG를 사용하여 인벤토리 및 여러 UI 제작
- 레벨 시퀀스를 이용한 컷신 제작

unity

- 모바일 롤플레잉 게임 개발
- 포톤서버 활용한 1대1 미니 게임 프로젝트 개발
- 미니 RPG 프로젝트 개발
- 모바일 뱀서라이크 프로젝트 개발
- 구글 로그인 및 구글 인앱결제 구현

Tool Visual Studio, SVN, Sourcetree, GitHub

대외 활동

2025.06~2025.06	2025메타버스 엑스포(언리얼 게임 부스 운영)
2025.05~2025.05	2025 플레이 엑스포(참관)
2023.05~2023.05	2023 플레이 엑스포(참관)
2018.11~2018.11	2018 지스타(참관)

CONTENTS

EscapeFromHuman(2025년 엑스포 참여)

작업 인원 : 클라이언트 3명
작업 파트 : 아이템, UI, 게임 시스템
개발 기간 : : 2025년 5월 7일 ~ 2025년 6월 27일 (약 2달))

네트워크 프로젝트

작업 인원 : 클라이언트 2명
작업 파트 : 캐릭터(사냥꾼), 네트워크 연동, UI
개발 기간 : : 2025년 4월 3일 ~ 2025년 4월 30일 (약 1달))

VR 프로젝트

작업 인원 : 클라이언트 2명
작업 파트 : 반사 레이저 방, 문제 풀이방
개발 기간 : 2025년 3월19일 ~ 2025년 4월 1일 (약 2주)

액션 프로젝트

작업 인원 : 클라이언트 2명
작업 파트 : 본인 : 보스(AI), 엔딩 컷신
개발 기간 개발 기간 : 2025년 02월 03일 ~ 2025년 3월 06일 (약 1달)

RPG 프로젝트

작업 인원 : 클라이언트 2명
작업 파트 : 몬스터(일반, 보스), 아이템, 인벤토리
개발 기간 : 2024년 12월17일 ~ 2025년 1월 22일 (약 1달)

유니티 모바일 프로젝트(기업 프로젝트)

작업 인원 : 클라이언트 1명 , 아트 1명, 기획 1명
작업 파트 : 게임 프로토타입의 전투 로직, 상점, 데이터 관리, 구글로그인 등 전반적인 모든 시스템
개발 기간 : 2022년 09월 ~ 2023년 11월

Escape From Human

프로젝트 개요

1대1 네트워크 대전 게임으로, 플레이어는 동물 캐릭터를 조작하여 맵에 주기적으로 생성되는 점수 오브젝트 수집하고 자신의 기지로 운반하여 점수를 획득

박치기, 스킬, 아이템을 활용하여 상대를 방해 할 수 있음
운반된 점수는 다시 빼앗는 것도 가능

AI 사냥꾼과 차량이 맵에 등장하여 플레이어를 방해
플레이어 스킬 "얼음"을 통해 안전하게 이동가능

본인 : 아이템, UI, 게임 시스템 개발

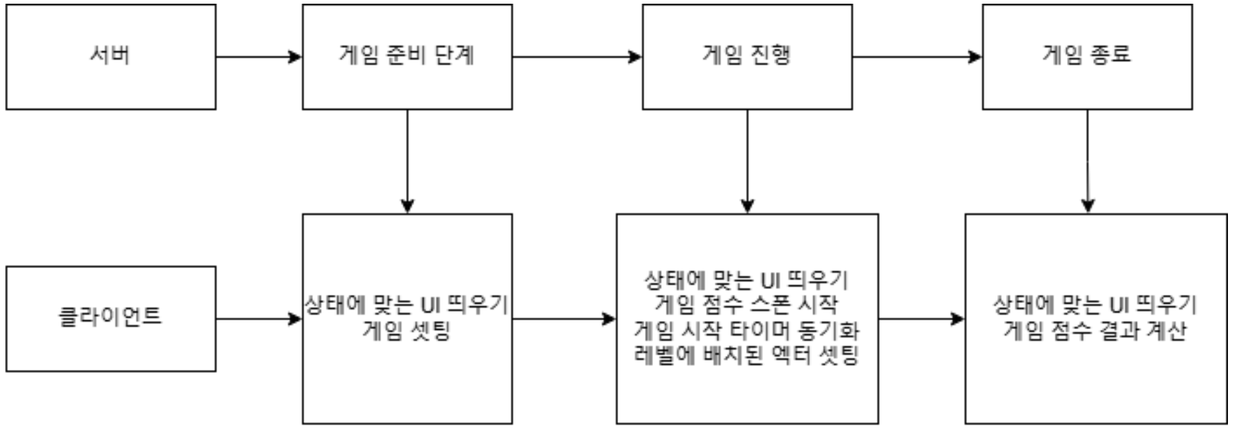
개발 기간 : 2025년 5월 7일 ~ 2025년 6월 27일 (약 2달)

개발 일정

Milestone	2025년 5월				2025년 6월			
	1주차	2주차	3주차	4주차	5주차	6주차	7주차	8주차
기획	게임 기획	기획 수정						
프로토 타입		로직 / 네트워크						
알파				UI / 레벨 및 컨셉 맞추기				
베타						최종 점검	엑스포 참여	이슈 수정

Escape From Human

게임 전체 진행

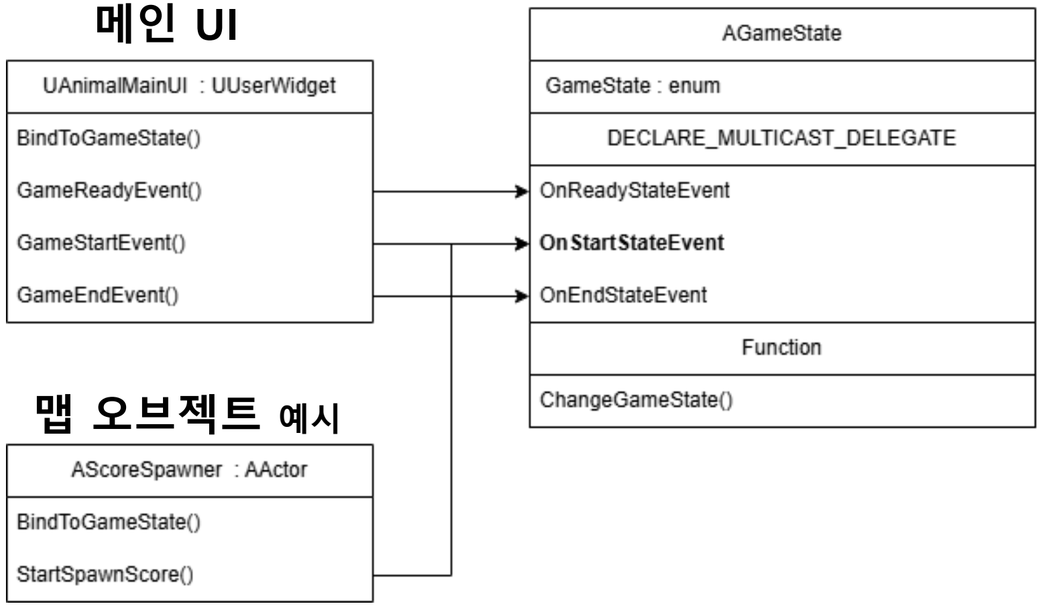


게임 스테이트 상태

- 게임 스테이트에서 게임 상태 전환과 게임 이벤트 발생 시 UI 및 엑터 시스템과 연동이 가능하도록 델리게이트를 선언하여 브로드캐스트 방식으로 이벤트 전달
- UI 전환, 연출, 데이터 동기화 등 게임 상태에 맞게 대응할 수 있는 구조 확보

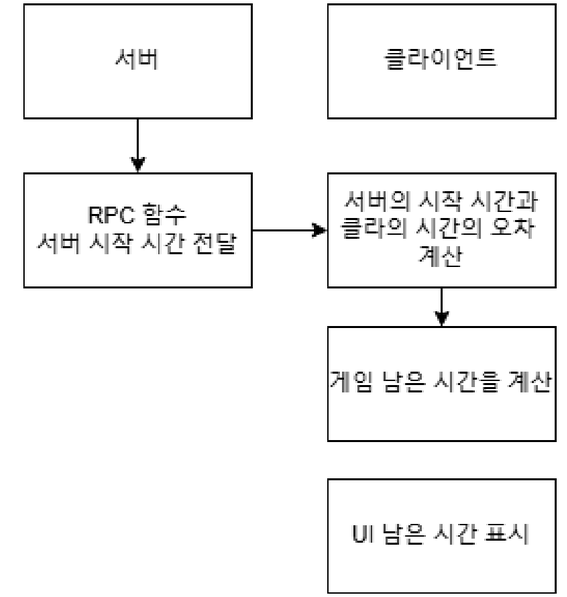
게임 스테이트

메인 UI



맵 오브젝트 예시

게임서버 타이머 동기화

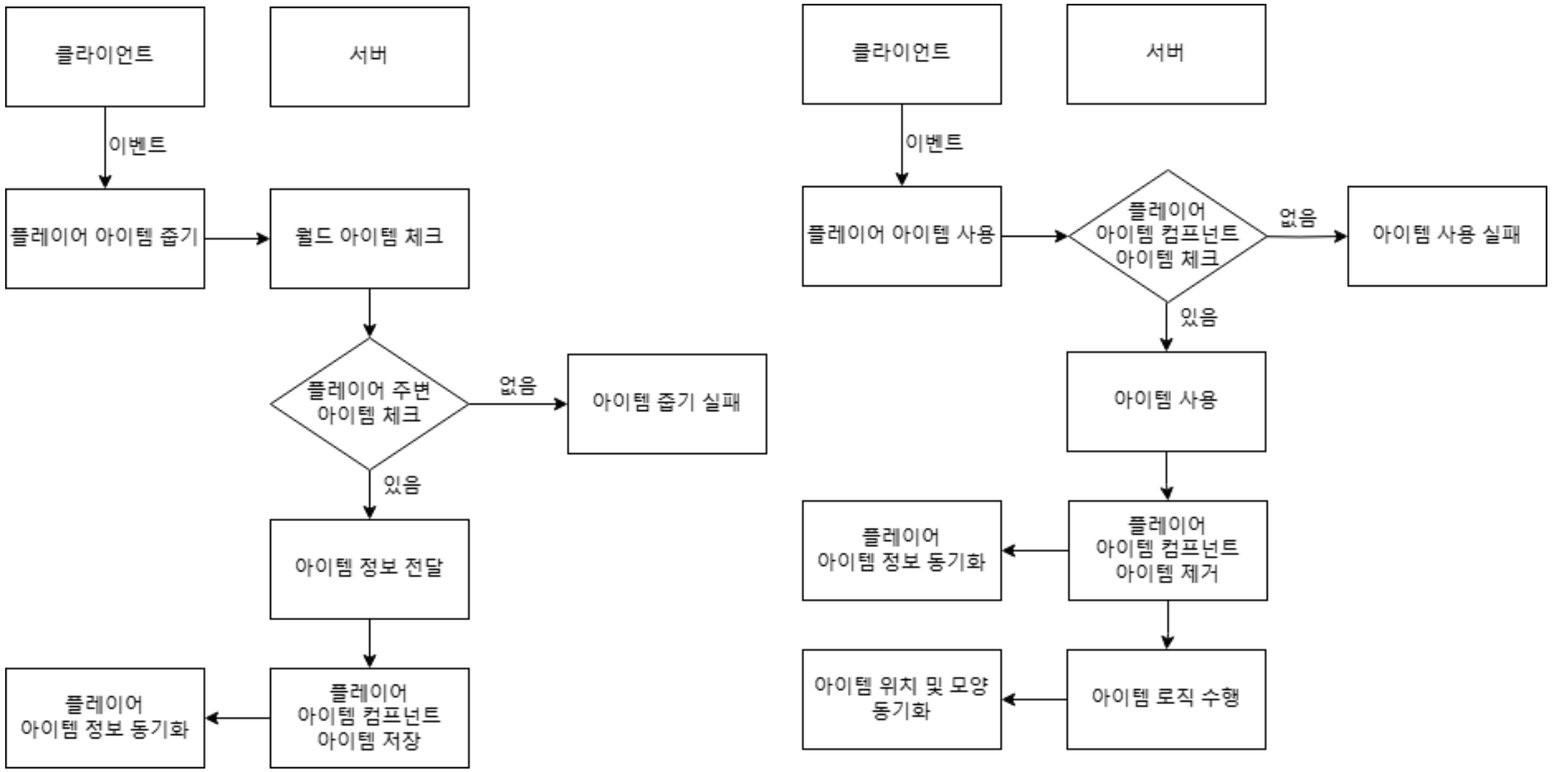


Escape From Human

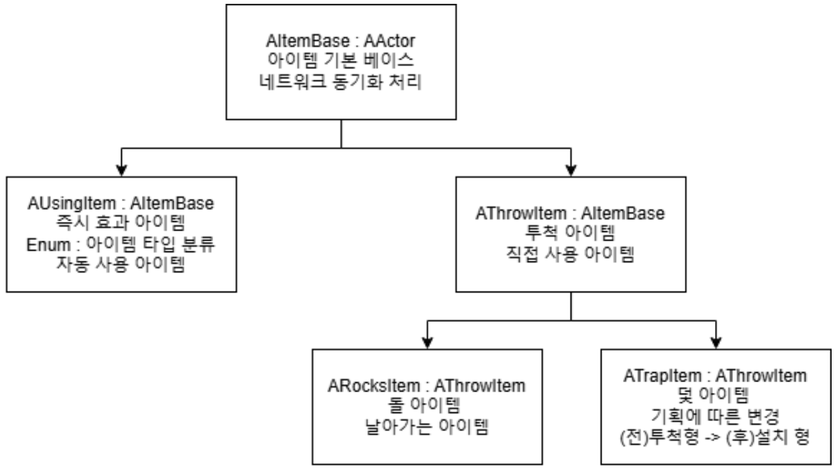
아이템

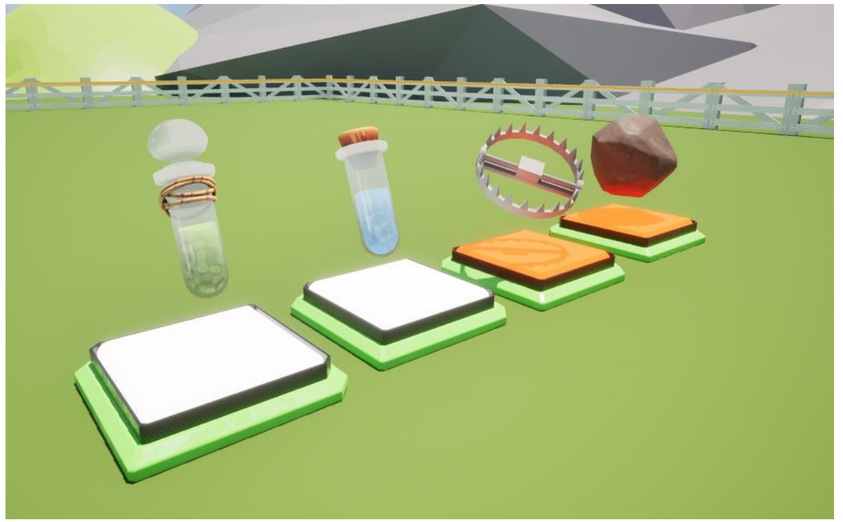
- 캐릭터 전용 ItemComponent를 설계 하여 투척용, 설치형, 버프형 아이템을 구분하여 사용및 관리 하고 RPC기반을 통한 네트워크 연동 구현

네트워크 구조



아이템 구조

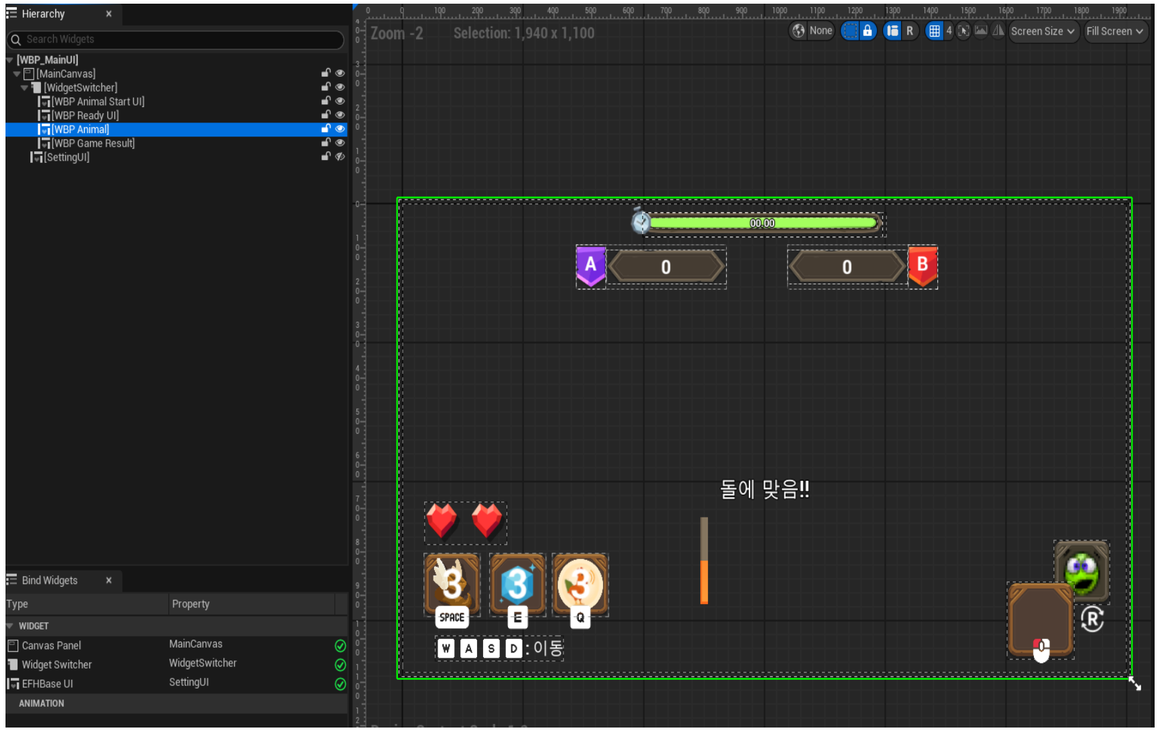




Escape From Human

UI

- 모든 UI위젯은 UEFHBaseUI 클래스를 상속받아 공통 기능 구현
- 메인 UI 에 하위 필요한 UI를 바인딩하여 필요한 게임 상태에 따라 하위 UI를 표시
- UWidgetSwitcher를 사용하여 상태에 하나만 나오게 구현



- 캐릭터 선택 UI
  - 캐릭터 데이터에 따른 설명 및 이미지
  - 선택 캐릭터 포커싱
- 게임 준비 UI
  - 키보드 설명
  - 게임 준비 상황
- 게임 중 UI
  - 게임 진행 상황 및 시간
  - 캐릭터 스킬 , 스킬 사용시 남은 쿨타임 표시 , 애니메이션
  - 캐릭터 아이템 슬롯
- 게임 결과 UI
  - 게임 스코어에 따른 결과 이미지 등 이펙트

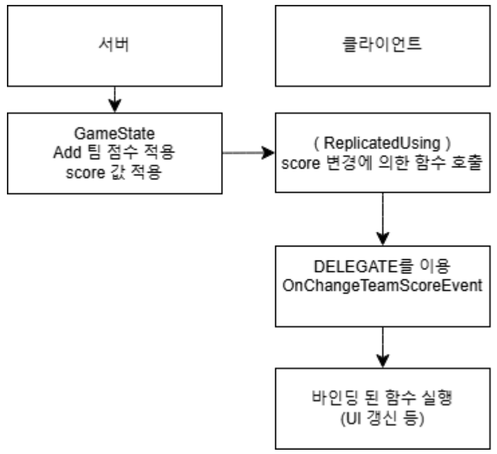
Escape From Human

기타 오브젝트

점수 오브젝트 (과일, 채소)

- 일정시간마다 지정된 위치배열에서 랜덤으로 스폰
- 캐릭터가 콜리전 오버랩 되면 캐릭터 등에 부착후 기지로 들어가면 기지에 저장 -> 1점 획득
- 다른 팀의 캐릭터가 오면 다시 가져가 점수를 기지 밖으로 가져가면 -> 1점 감소

스코어 연동



PostProcessVolume

- PostProcessMat를 이용하여 좀더 게임 컨셉에 맞게 디자인
- 시간에 따라 노을이 지는 느낌이 나도록 구현

적용 전

적용 후

Escape From Human

엑스포 참가 피드백

게임 시연 피드백 - 1

1. 게임 조작이 어렵다
    - 아이템을 먹고 교체 하는 과정이 생각보다 복잡
    - 게임 시연 당시 연령층이 낮고 게임을 많이 접하지 못한 경우가 다수

개선 방법 : (예정)
아이템 자동 습득으로 교체 및 아이템 사용순서 고정으로 하여 교체 키 제거
기존 Q,E 키 하나를 마우스 한쪽으로 이관

2. 플레이어 시작 대기 시간 불균형
    - 기존 두명의 플레이어 캐릭터 선택 후 40초 후 본 게임 시작
    - 하지만 게임 미숙및 낮은 연령층으로 인한 게임 설명 부족
    - 개발 시간 부족으로 인한 튜토리얼 설명 부족

개선 방법 : (개선 완료)
시간이 지나고 나서 시작이 아닌 준비완료 키를 만들어 2명 모두 게임 완료가 되면 10초 뒤 본게임 시작

개선 후 : 설명할 시간 충분하여 시연자가 충분히 게임 테스트를 진행 가능

추후 개선 : 튜토리얼 단계 구현

Escape From Human

엑스포 참가 피드백

게임 시연 피드백 - 2

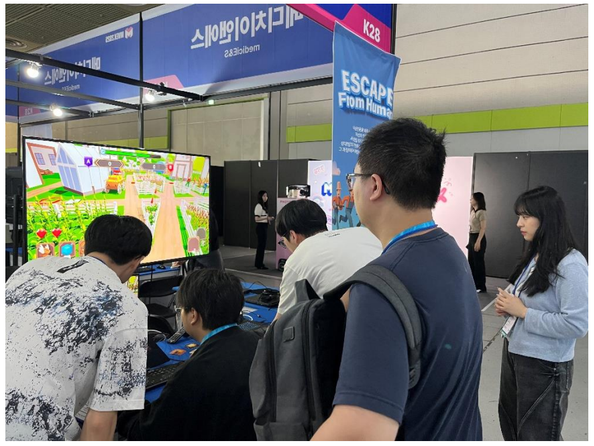
3. 벨런스 조절
    - 캐릭터 간의 게임 벨런스 테스트 미흡
    - 단순 스킬의 쿨타임 변경 벨런스 필요
    - 시연 도중 개발 및 패치 시간이 오래 걸림

개선 방법 :
게임 쿨타임 데이터를 JSON파일을 이용하여 외부에서 수정가능하도록 변경

4. 단순 버그
    - 박치기 스킬 사용시 이미 콜리전이 겹친 상태라면 박치기 이벤트가 발생 X
    - 클라이언트 접속 중 서버에서 캐릭터가 선택하면 뒤에 들어온 클라이언트에서 서버 캐릭터가 안보임
    - 자신의 집에서 점수를 강탈 당할 때 방해 하면 점수가 원래 위치로 가지 않는 버그

Escape From Human

엑스포 참가 사진



네트워크 프로젝트

프로젝트 개요

숨바꼭질 기반 액션 캐주얼 게임 Witch It를 모작
언리얼 온라인 서브시스템을 이용한 세션 및 네트워크 붙이기

작업 인원 : 클라이언트 2명

본인 : 캐릭터(사냥꾼), 네트워크 연동, UI

팀원1 : 캐릭터(마녀) , 레벨디자인, 세션 만들기

개발 기간 : 2025년 4월 3일 ~ 2025년 4월 30일 (약 1달)

개발 일정

Milestone	2025년 4월			
	1주차	2주차	3주차	4주차
기획	게임 선정			
프로토 타입		사냥꾼/ UI		
알파			네트워크/게임 로직	
베타				게임 텍스트/ 이펙트

네트워크 프로젝트

캐릭터(사냥꾼)

캐릭터 클래스 설계

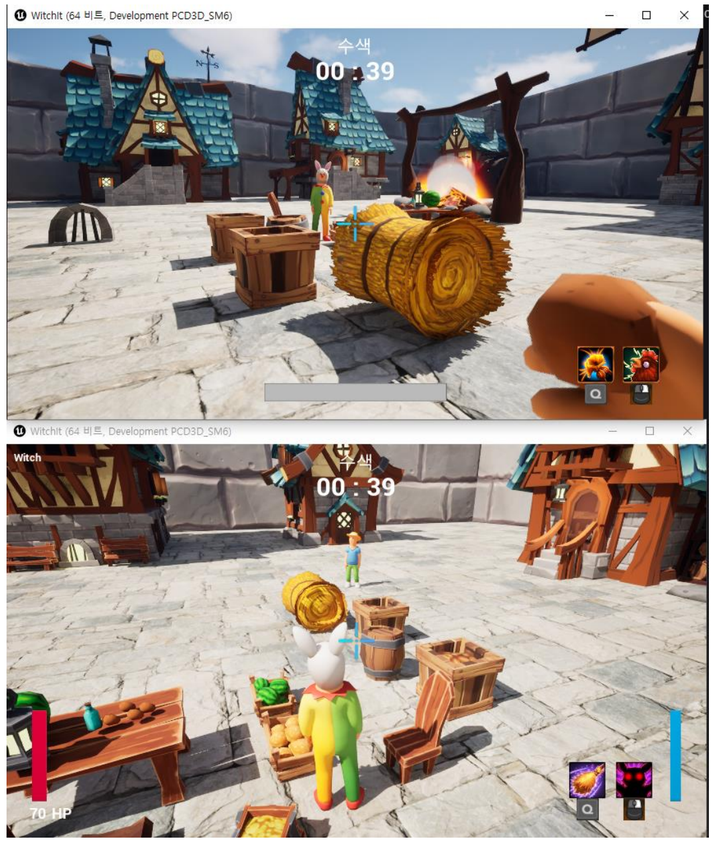
- 캐릭터 사냥꾼, 마녀의 상위 클래스를 만들고 공통 로직 수행
- 각 사냥꾼, 마녀에서 각각의 기능 로직 개발

애니메이션

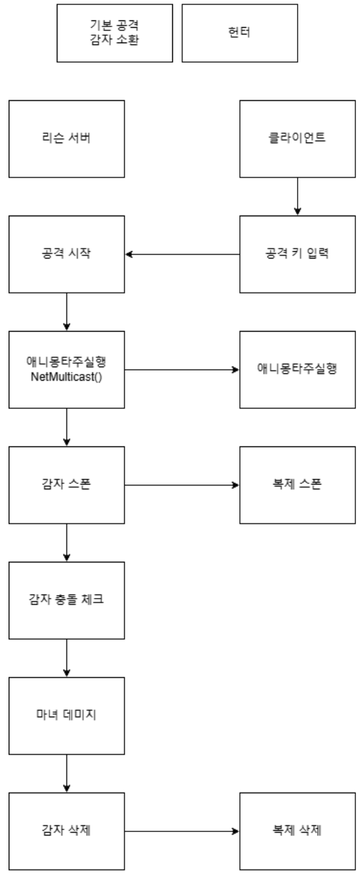
- 던지기 애니메이션이 제자리 에셋이라 애니메이션 본 블랜딩을 사용하여 자연스러운 애니메이션 연출
- 공격 애님몽타주 실행시 공격 함수 실행

사냥꾼 이동

- 기본 키보드 조작 구현
- 감자(투사체)를 이용한 상호작용 구현
- 공격에 스테미나 시스템 적용



네트워크 플로우



네트워크 프로젝트

스킬

스킬 컴포넌트 클래스 설계

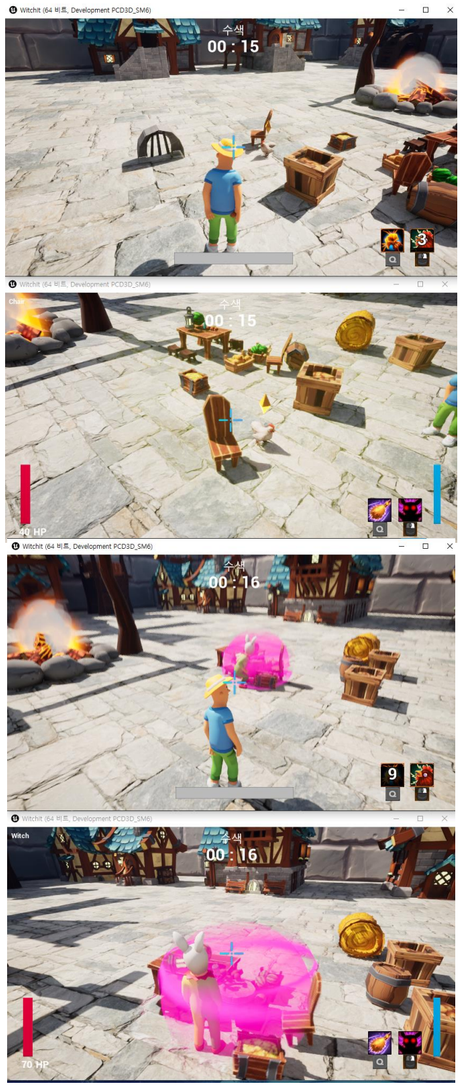
- 캐릭터의 스킬을 사용 및 관리 하는 컴포넌트
- 스킬 사용 입력에 따라 사용 가능한 스킬 발동 및 네트워크 연동 지원

스킬객체 클래스 설계

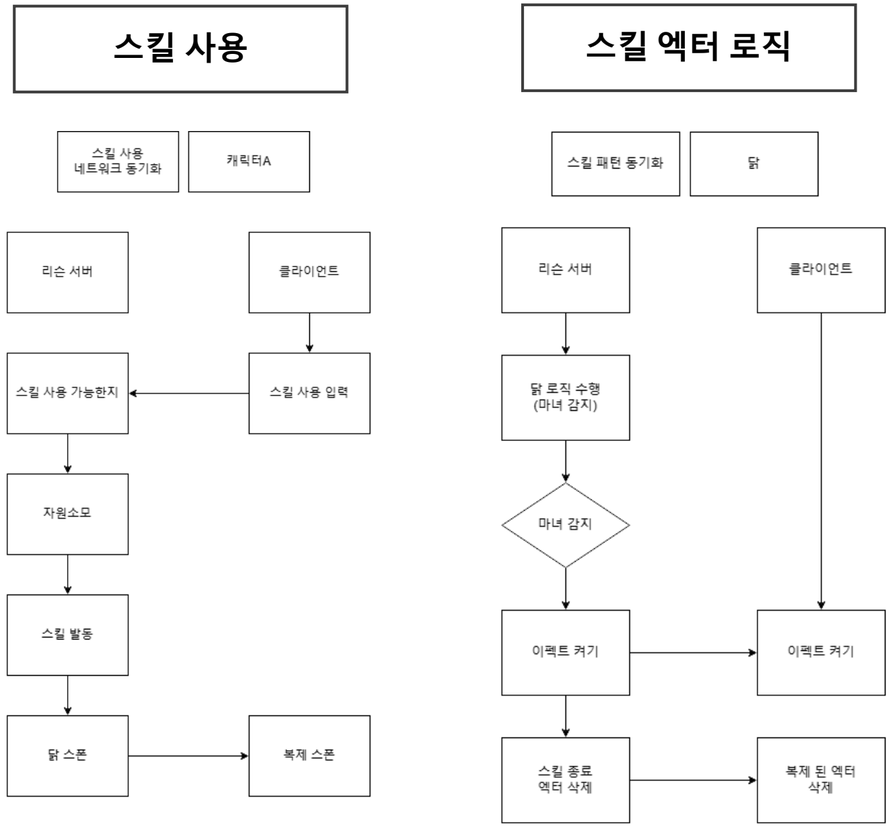
- Uobject 기반 설계
- 공통 기능 가상함수를 만들어 필요한 부분만 설계 가능

```
UCLASS( Blueprintable )
class WITCHIT_API USkillBase : public UObject
{
    GENERATED_BODY()

public:
    //스킬을 사용할수 있는지
    virtual bool IsStart ();
    //스킬 게임 시 초기화 함수
    virtual void SkillInit ();
    //스킬 자원 소모 함수
    virtual void UseResource();
    //스킬 동작 시작 함수
    virtual void StartSkill ();
    //스킬 동작 시 돌아가는 Tick 함수
    virtual void SkillTick ( float DeltaTime );
    //스킬 키 입력 트리거
    virtual void TriggerSkill ( );
    //스킬 키 입력 종료
    virtual void CompletedSkill ( );

public:
    //현재 가지고 있는 주인 객체
    UPROPERTY (VisibleAnywhere )
    ABaseCharacter* SkillOwner;
```

```
UCLASS()
class WITCHIT_API USkillChicken : public USkillBase
{
    GENERATED_BODY()

public:
    //스킬을 사용할수 있는지
    virtual bool IsStart () override;
    //스킬 자원 소모 함수
    virtual void UseResource ();
    //스킬 게임 시 초기화 함수
    virtual void SkillInit () override;
    //스킬 동작 시작 함수
    virtual void StartSkill () override;



private:
    UPROPERTY(EditDefaultsOnly , Category = SkillChicken )
    TSubclassOf<class AChicken> ChickenBP;

};
```



네트워크 플로우

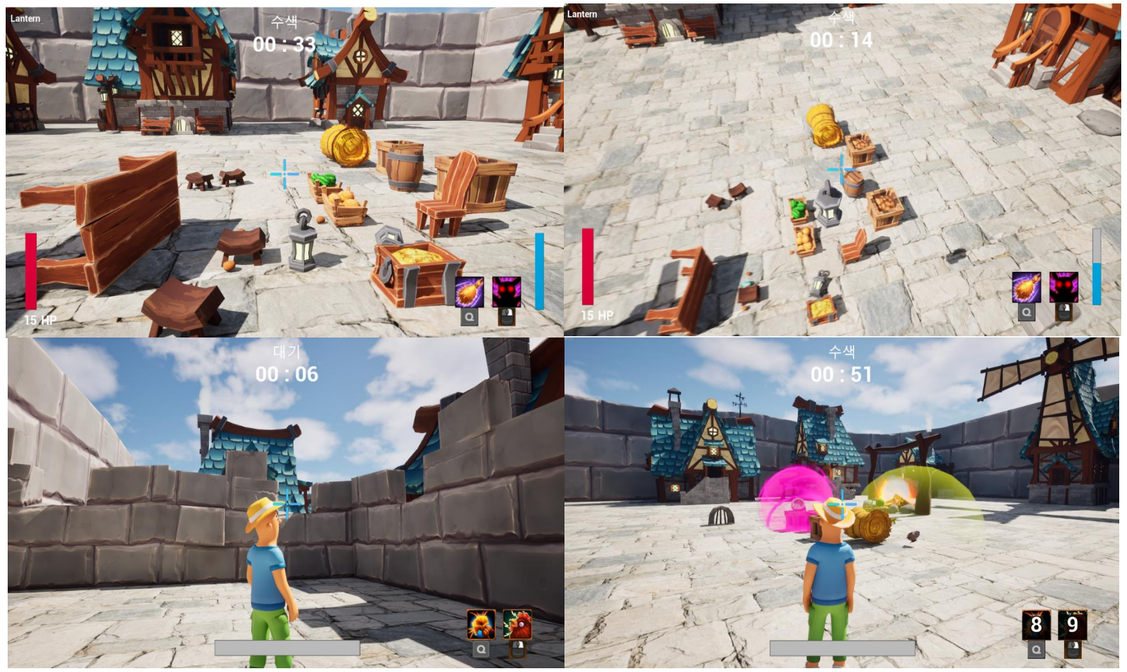


네트워크 프로젝트

UI

팀별 인게임 UI

- 마녀, 사냥꾼 팀에 맞는 UI 배치 및 텍스트 설정
- 각 스테이터스에 따른 이벤트 발생시 델리게이트 함수를 활용하여 UI 갱신



플레이어 팀 선택 UI

- 마녀, 사냥꾼 팀에 맞는 UI 배치 및 텍스트 설정
- 플레이어 선택에 따라 팀 플레이어 리스트 및 인원수 갱신

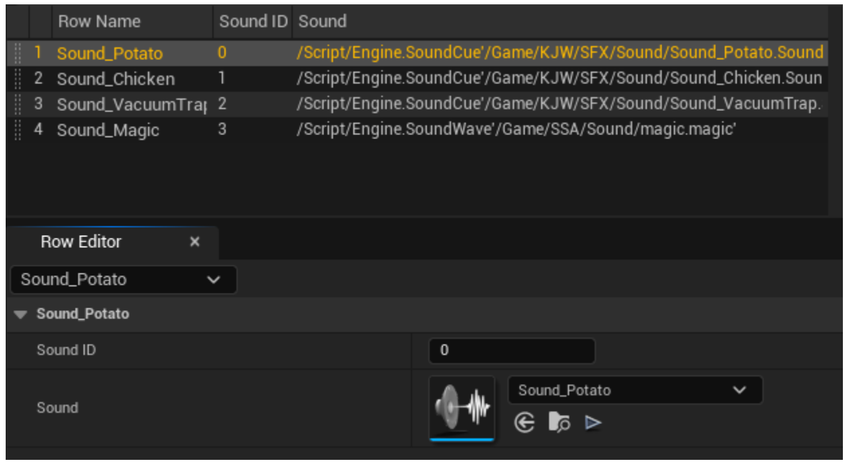
플레이어 선택->플레이어스테이트 갱신->서버->모든 클라이언트 UI 갱신

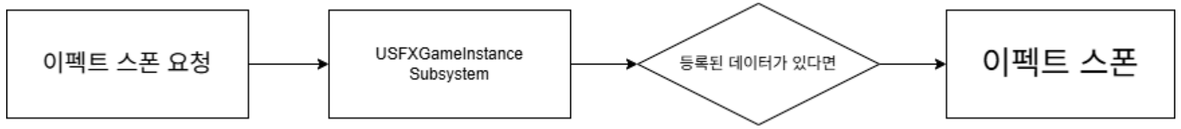
네트워크 프로젝트

사운드/이펙트

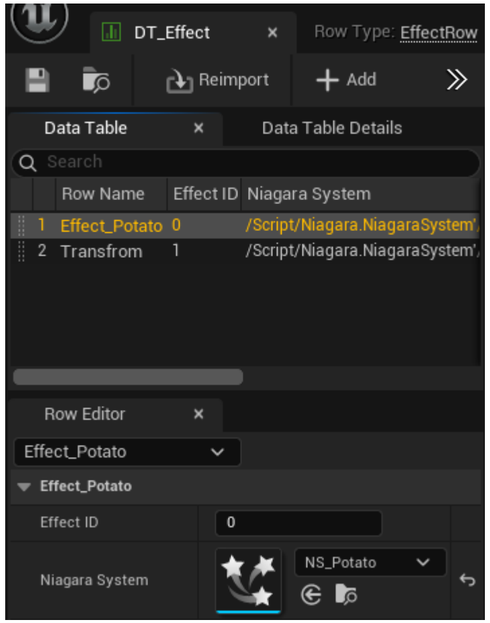
사운드/이펙트 매니저

- 모든 클래스에서 접근을 위한 클래스 설계
- GameInstaceSubsystem class 를 활용
- 이펙트 및 사운드 효과를 사용 또는 소환를 이 클래스에서 수행
- 게임 시작시 필요한 데이터를 테이블에 넣어 필요할때 찾아 사용






```cpp
void APotato::OnHitComponent ( UPrimitiveComponent* HitComponent , AActor* OtherActor , UPrimitiveComponent* OtherComp , FVector NormalImpulse , const FHitResult& Hit )
{
    AWitch* Witch = Cast<AWitch> ( OtherActor );

    USFXGameInstanceSubsystem* subsystem = GetGameInstance ()->GetSubsystem<USFXGameInstanceSubsystem> ();
    if (subsystem)
    {
        subsystem->PlaySound ( GetWorld () , GetActorLocation () , 0 );

        if (Witch)
        {
            subsystem->SpawnEffect ( GetWorld () , GetActorLocation () , FRotator::ZeroRotator , 0 );
        }
    }
```



네트워크 프로젝트

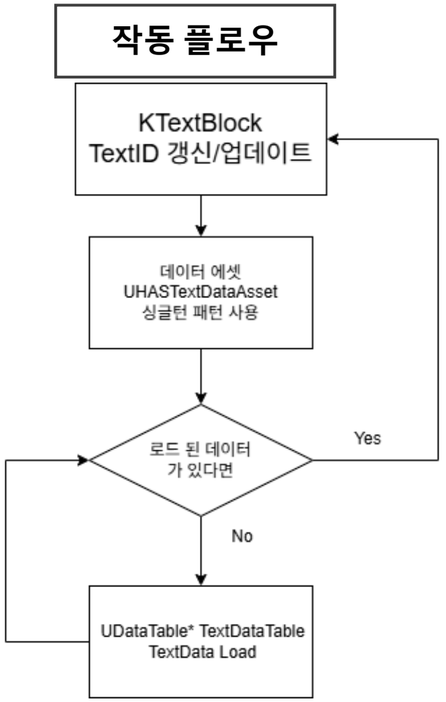
다국어 시스템

KTextBlock 커스텀

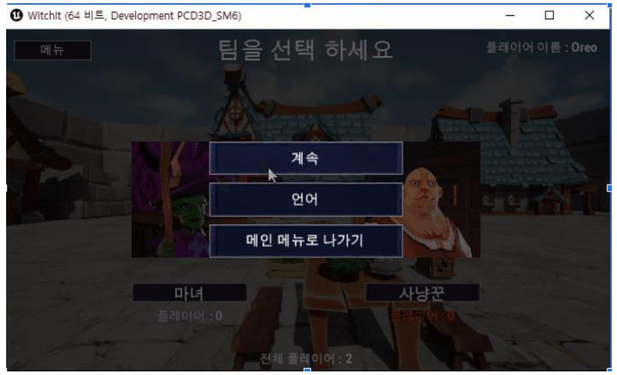
- UTextBlock를 상속받아 로컬라이징 가능하게 설계
- 테이블에 있는 인덱스를 사용시 설정에 맞는 텍스트를 가져올 수 있도록 구현
- 에디터 상에서도 바로 실시간으로 사용가능 하도록 설계
- 런타임 에서는 GameInstanceSubsystem에서 텍스트 변환
- 에디터 에서는 DataAsset를 싱글턴 패턴을 활용하여 텍스트 변환

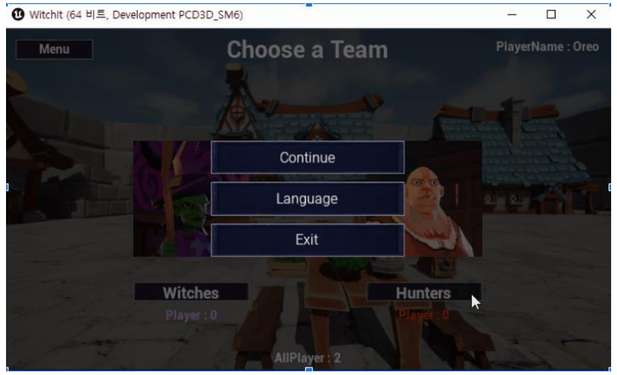
```
UCLASS()
class WITCHIT_API UHASTextDataAsset : public UDataAsset
{
    GENERATED_BODY()
public:
    // 싱글턴 패턴을 위한 정적 함수
    static UHASTextDataAsset* GetInstance ();
public:
    // 텍스트 조회 함수
    FString GetTextByIndex ( int32 Key );
public:
    UFUNCTION ( CallInEditor , Category = "CacheText" )
    void Editor_CacheTagToTextMap ()
    {
        UE_LOG ( LogTemp , Warning , TEXT ( "CacheTagToTextMap" ) );
        GetInstance ();
        CacheTagToTextMap ();
    }
    //0, 한국어 , 영어
    UPROPERTY ( EditAnywhere , BlueprintReadWrite )
    int32 Language = 0;
private:
    void CacheTagToTextMap ();
    TMap<int32 , FString> KeyKorTextMap;
    TMap<int32 , FString> KeyEngTextMap;

    // 싱글턴 인스턴스를 위한 static 변수
    static UHASTextDataAsset* SingletonInstance;
    // 싱글턴 인스턴스를 가져오는 함수
    static void InitializeSingleton ();
    // DataTable 세팅
    UPROPERTY ( EditDefaultsOnly , Category = "KText" )
    class UDataTable* TextDataTable;
```

작동 플로우



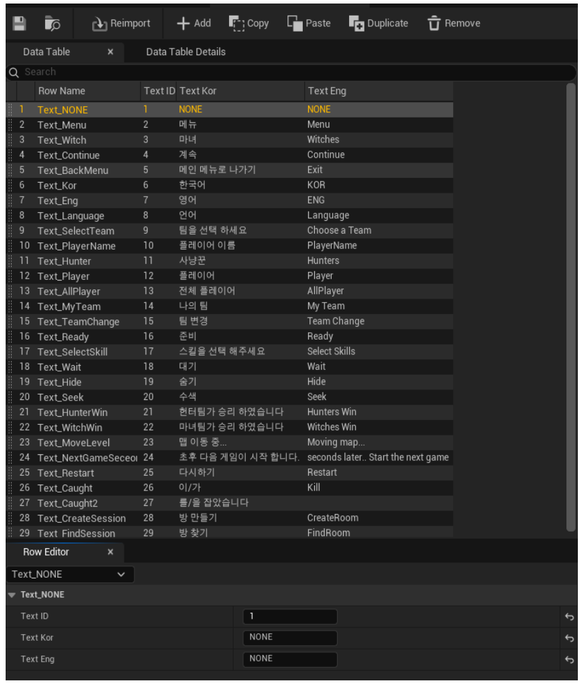
실제 사용 시





KTextBlock

TextTable

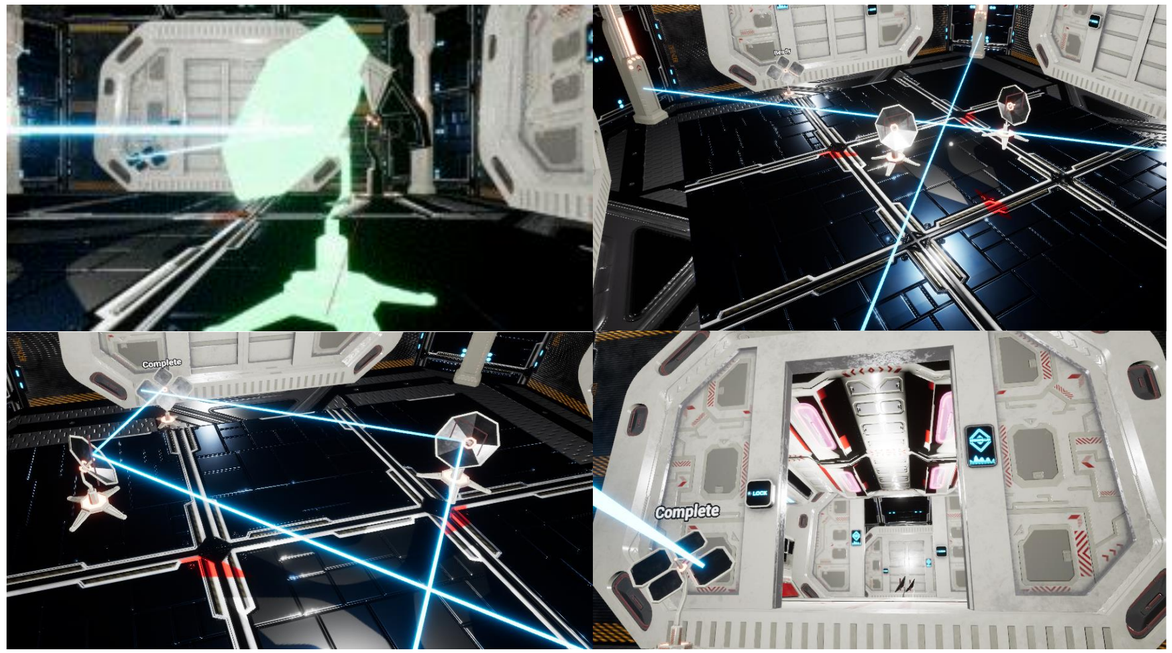


VR 프로젝트

프로젝트 개요

VR 컨트롤을 사용하여 방 컨셉에 맞는 문제를 풀어 탈출하는
방탈출 게임

작업 인원 : 클라이언트 2명
본인 : 반사 레이저 방, 문제 풀이방
팀원1 : 플레이어 컨트롤러, 2중퍼즐 방
개발 기간 : 2025년 3월19일 ~ 2025년 4월 1일 (약 2주)



개발 일정

Milestone	2024년 3월	
	1주차	2주차
기획/프로토타입	레이저방 기믹/ 문제 풀이 기믹 구현	
알파/베타		VR 연결 및 레이저 방 디테일 작업

VR 프로젝트

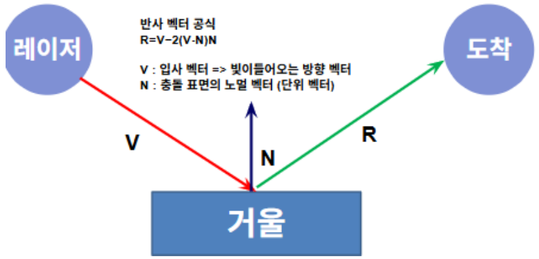
레이저 방 — **시작 지점에서 나오는 레이저를 도착 지점까지 가게 만들면 탈출 하는 방**

레이저 생성 방법

시작 → 거울 → 도착지

- 시작 : 라인트레이스 발사, EndPoint 배열 생성
- 충돌 확인 : 거울 -> 다시반복
  도착지 -> 클리어
- 레이저 이펙트 EndPoint 배열에 있는 포인트를 이용해서 이펙트 위치 및 길이 설정
- 이펙트는 오브젝트 풀링을 TQueue를 이용하여 사용

반사 각도 구현




```cpp
public:
    static FVector GetReflectionVector(const FVector& IncidentVector, const FVector& SurfaceNormal)
    {
        //기본 공식
        //R == V - 2 * N * (V dot N)
        //
        //V : 입사 벡터 => 빛이들어오는 방향 벡터
        //N : 충돌 표면의 노멀 벡터 (단위 벡터)

        FVector V = IncidentVector; // 입사 벡터
        FVector N = SurfaceNormal;  // 노멀 벡터

        //(V dot N)
        float DotProduct = FVector::DotProduct(V, N);  // 1. V와 N의 내적 계산
        //N * (V dot N)
        FVector ProjectionOntoNormal = DotProduct * N; // 2. 노멀 방향 성분 찾기
        // V - 2 * N * (V dot N)
        FVector ReflectionVector = V - 2 * ProjectionOntoNormal; // 3. 반사 벡터 계산

        return ReflectionVector;
    }
```

거울 (반사, VR 오브젝트)

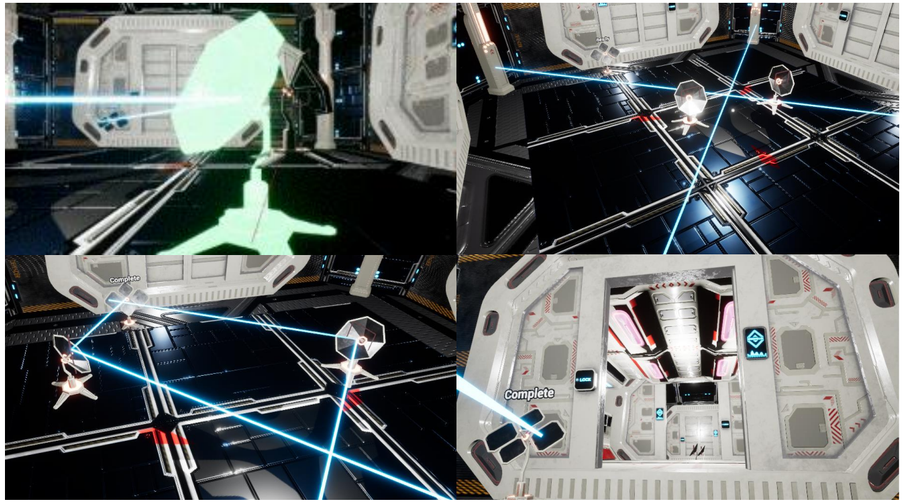
오브젝트 선택 StartGrab() → 임시 오브젝트 Grabbing() → 오브젝트 설치 StopGrab()

오브젝트 선택 상태 : 손으로 이동 및 스케일 축소 구현

그랩 중 상태 : 임시 오브젝트를 소환 -> 위치 이동 -> 정확한 위치 지정이 힘들어 GridSanp을 사용하여 보정 이동 , VR 기기 L스틱으로 회전 구현

그랩 끝 상태 : 손에 있는 오브젝트를 임시 오브젝트 위치 회전을 적용시켜 적용

구현 방 모습



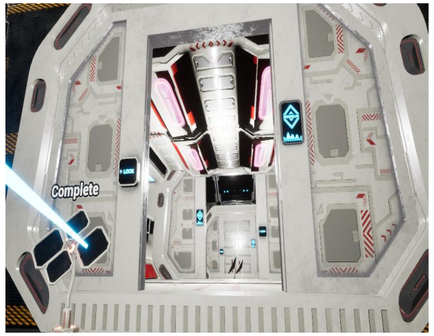
VR 프로젝트

레이저 방

Lerp와 Easing 를 이용한 연출

- 내가 원하는 애니메이션을 구할수 없어 사용 하기 됨
- 기존 Lerp만 사용해서는 연출 적인 느낌이 나지 않아 Easing 함수를 조합 하여 원하는 느낌의 연출을 만들어냄

- 문이 열리는 애니메이션이 없어 문짝의 위치값을 조절하여 처음에 빠르게 열리는 연출을 만듬

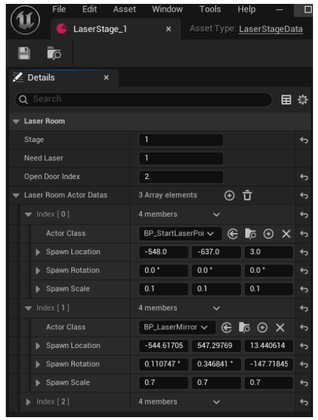


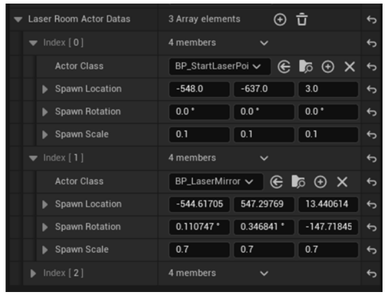

```cpp
template <typename T>
static T Lerp(const T& A, const T& B, float Alpha , EEasing Easing = EEasing::Linear)
{
    //기본 Lerp 공식
    // linear
    //Lerp = A + (B - A ) * Alpha
    //FMath::Lerp(LDoor_ClosePos, LDoor_OpenPos, percent);
    float x = Alpha;// 0 ~ 1
    switch (Easing)
    {
    case EEasing::Linear: { break;}
    //느리게 시작 → 점점 가속
    case EEasing::EaseOutQuint:{x = 1 - FMath::Pow(1 - x, 5);break;}
    //빠르게 시작 → 점점 감속
    case EEasing::EaseInQuint:{ x =  x * x * x * x * x; break; }
    default:
        break;
    }

    return A + (B - A) * x;
}
```

스테이지 비동기 생성

- 스테이지를 레벨에 배치해 두는게 아니고 게임 시작시 데이터를 가져와 알맞게 생성

스테이지 데이터 확인 → 위치 확인 → 오브젝트 스폰





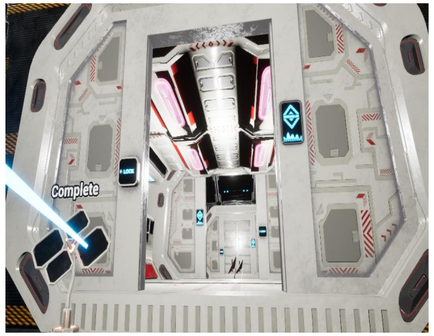
VR 프로젝트

레이저 방

Lerp와 Easing 를 이용한 연출

- 내가 원하는 애니메이션을 구할수 없어 사용 하기 됨
- 기존 Lerp만 사용해서는 연출 적인 느낌이 나지 않아 Easing 함수를 조합 하여 원하는 느낌의 연출을 만들어냄

- 문이 열리는 애니메이션이 없어 문짝의 위치값을 조절하여 처음에 빠르게 열리는 연출을 만듬

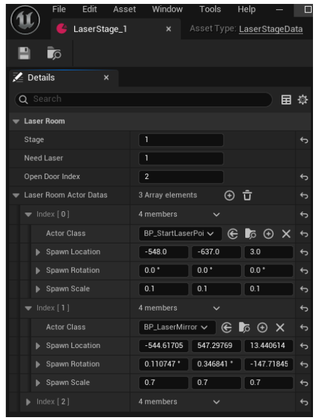


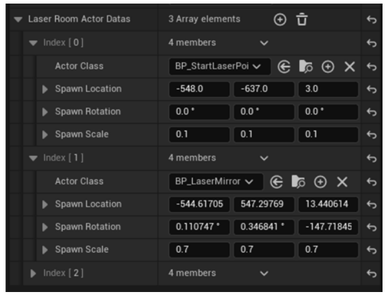

```cpp
template <typename T>
static T Lerp(const T& A, const T& B, float Alpha , EEasing Easing = EEasing::Linear)
{
    //기본 Lerp 공식
    // linear
    //Lerp = A + (B - A ) * Alpha
    //FMath::Lerp(LDoor_ClosePos, LDoor_OpenPos, percent);
    float x = Alpha;// 0 ~ 1
    switch (Easing)
    {
    case EEasing::Linear: { break;}
    //느리게 시작 → 점점 가속
    case EEasing::EaseOutQuint:{x = 1 - FMath::Pow(1 - x, 5);break;}
    //빠르게 시작 → 점점 감속
    case EEasing::EaseInQuint:{ x =  x * x * x * x * x; break; }
    default:
        break;
    }

    return A + (B - A) * x;
}
```

스테이지 비동기 생성

- 스테이지를 레벨에 배치해 두는게 아니고 게임 시작시 데이터를 가져와 알맞게 생성

스테이지 데이터 확인 → 위치 확인 → 오브젝트 스폰



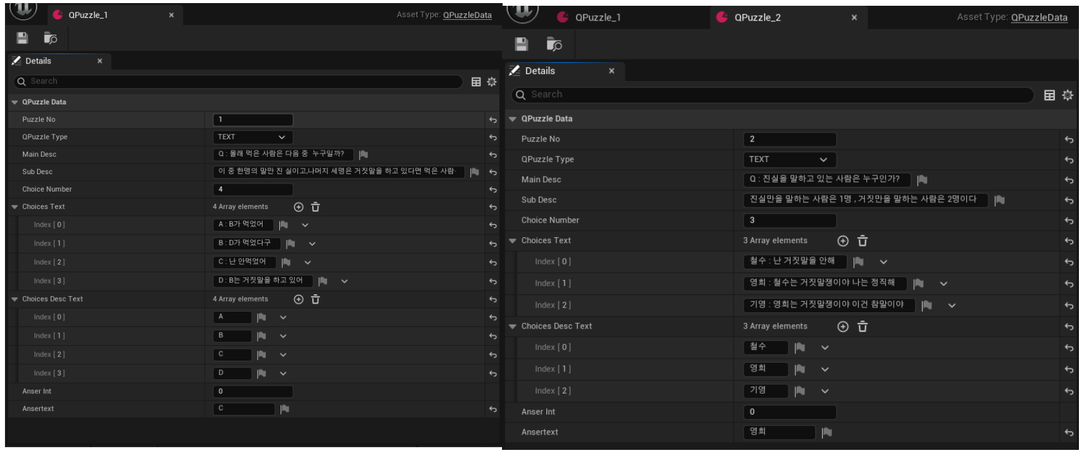


VR 프로젝트

객관식 문제 방 — **문제화면에 나오는 문제를 읽고 패드에 정답에 해당하는 버튼을 누르는 방**

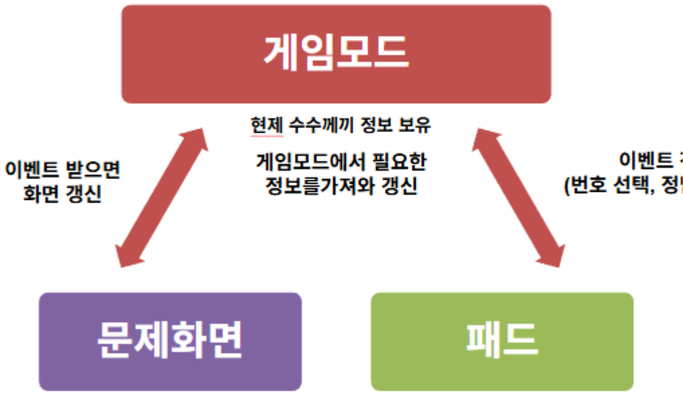
문제 진행 순서

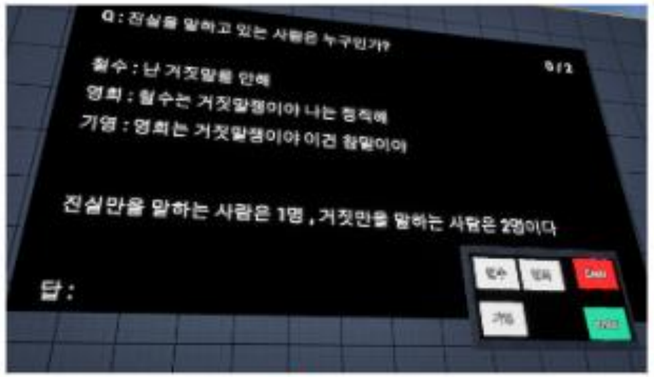
데이터 에셋을 활용하여 미리 여러 문제를 저장



문제화면과 패드는 바로 소통하는게 아니라
중간 게임모드를 통해 이벤트 전달

패드 쪽이나 문제화면 쪽 코드를 수정해도
관리 하기 쉬워 유지보수 편함





액션 프로젝트

프로젝트 개요

갓 오브 워 라그나로크 게임 중 토르와의 전투 부분 구현을 목표로 한 프로젝트

작업 인원 : 클라이언트 2명
본인 : 보스(토르), 엔딩 컷신
팀원1 :플레이어(크레토스)

개발 기간 : 2025년 02월 03일 ~ 2025년 3월 06일 (약 4주)

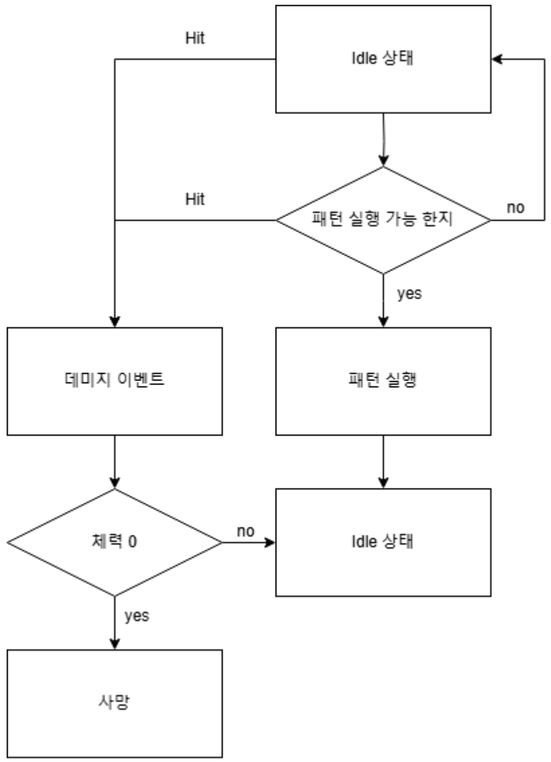
개발 일정

Milestone	2월			3월
	2주차	3주차	4주차	1주차
프로토 타입	토르 기본 공격 및 패턴 구현			
알파			플레이어와의 상호작용	
베타				컷 신 제작

액션 프로젝트

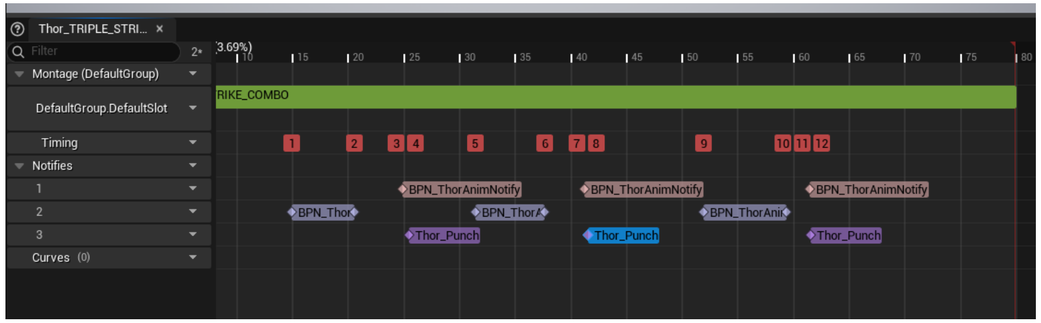
보스(토르)

- 여러 패턴을 사용 할 수 있고 상태는 하나만 소유 하기 때문에 으로 FSM 구현
- 각 패턴은 UThorPattern 클래스로 분리하여 제작
- Enum 과 Tmap을 활용하여 패턴 구조 맵핑



```
public:
    UFUNCTION(BlueprintCallable)
    void ChangePattarn(EThorPattern NewPattern);
    UFUNCTION(BlueprintCallable)
    void StartPattarn(EThorPattern NewPattern);
    UFUNCTION(BlueprintCallable)
    void TickPattarn();
    UFUNCTION(BlueprintCallable)
    void EndPattarn(EThorPattern NewPattern);
private:
    UPROPERTY()
    class UThorPattern* CurPattern;
    UPROPERTY()
    TMap<EThorPattern , class UThorPattern*> Patterns;
```

패턴 몽타주



AnimNotify, AnimNotifyState 를 활용하여 공격판정 및 이벤트 연결

```
protected:
    //FUNCTION -> C++ -> Blueprint
    virtual void StartPattern_C();
    virtual void StopPattern_C();
    virtual void EndPattern_C();
    virtual bool TickPattern_C();

    virtual void NotifyEventPattern_C(int32 EventIndex);
    virtual void NotifyBeginPattern_C(int32 EventIndex , float TotalDuration);
    virtual void NotifyEndPattrern_C(int32 EventIndex);
    virtual void NotifyTickPattrern_C(int32 EventIndex, float FrameDeltaTime);
```

토르의 패턴

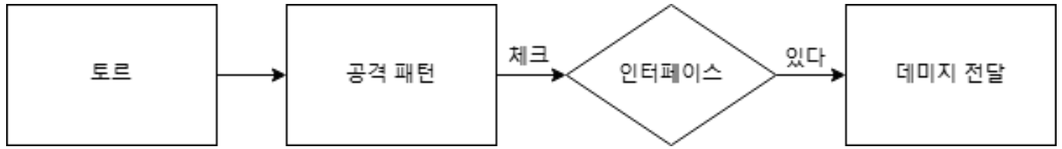
액션 프로젝트

캐릭터 상호작용

- 플레이어 캐릭터와 보스의 캐릭터의 데미지 상호작용 구현
- 캐릭터 제작 파트와의 소스 충돌을 막고자 상호작용은 인터페이스 구조를 사용

```cpp
//WOG 전용 데미지 구조체
USTRUCT(BlueprintType)
struct FWOG_DamageEvent
{
    GENERATED_BODY()
public:
    //데미지 수치
    float DamageValue = 0.0f;
    //스턴 수치
    float StunValue = 0.0f;
    //데미지 포인트
    FVector HitPoint = FVector::ZeroVector;
};
```

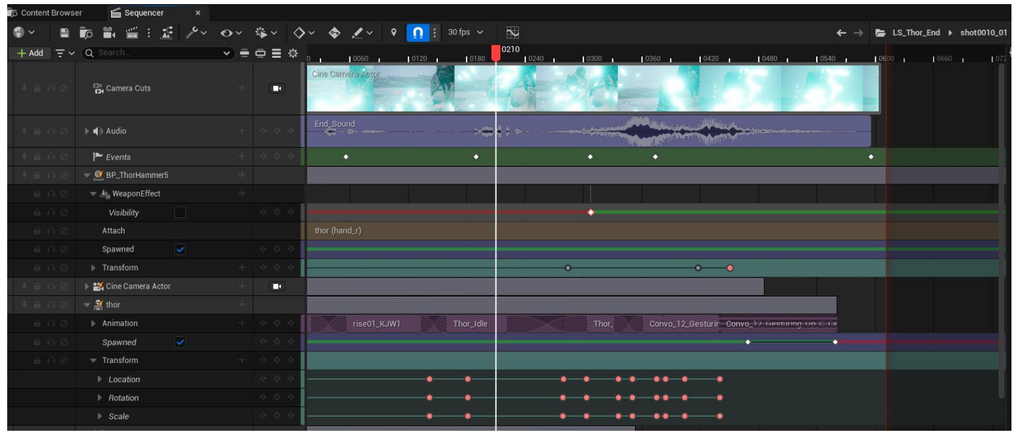
```cpp
class TEAMPROJECT_WOG_API ICombatInterface
{
    GENERATED_BODY()
protected:
    EWOG_Character_State CharacterState = EWOG_Character_State::NONE;
};
// Add interface functions to this class. This is the class that will be inherited to implement this interface.
public:
    //현재 상태 리턴
    EWOG_Character_State GetCharacterState() { return CharacterState;}
    //상태 변경시키는 함수
    virtual void SetCharacterState(EWOG_Character_State NewState) = 0;
    //WOG 전용 데미지 주는 함수
    virtual void TakeDamage(const FWOG_DamageEvent& DamageEvent , ICombatInterface* DamageCauser) = 0;
    //애니메이션을 강제로 시켜야할경우
    virtual class USkeletalMeshComponent* GetSkeletalMesh() = 0;
    //뭔가 조작이 필요할때
    virtual AActor* GetActor() = 0;
};
```




```cpp
void AThor::TakeDamage( const FWOG_DamageEvent& DamageEvent, ICombatInterface* DamageCauser)
{
    if ( Hp <= 0 ) { return; }
    if ( CurPattern == GetPattern(EThorPattern::STUN) ) { return; }

    float DamageValue = DamageEvent.DamageValue;

    if ( GEngine )
    {
        GEngine->AddOnScreenDebugMessage(-1 , 5.0f , FColor::Red , FString::Printf(TEXT("Thor : Damage Value: %f") , DamageValue));
    }

    IsHit = true;

    Hp -= DamageValue;
    StunValue += 10;
```

컷신

실제 게임에도 컷신이 많아 레벨 시퀀스를 이용하여 컷신을 제작



RPG 프로젝트

프로젝트 개요

쿼터뷰 액션 RPG 게임인 페스오브엑자일을 모티브한 프로젝트

작업 인원 : 클라이언트 2명
본인 : 몬스터(일반, 보스), 아이템, 인벤토리
팀원1 : 캐릭터, 스킬, 레벨

개발 기간 : 2024년 12월17일 ~ 2025년 1월 22일 (약 5주)

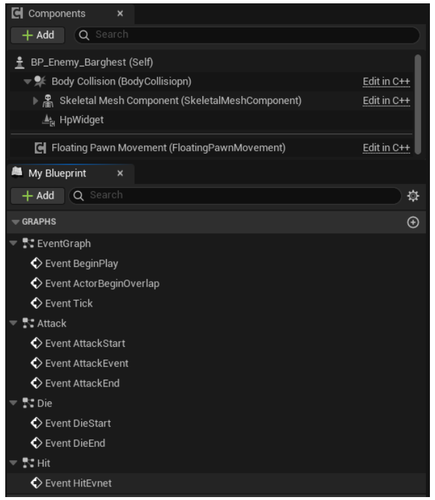
개발 일정

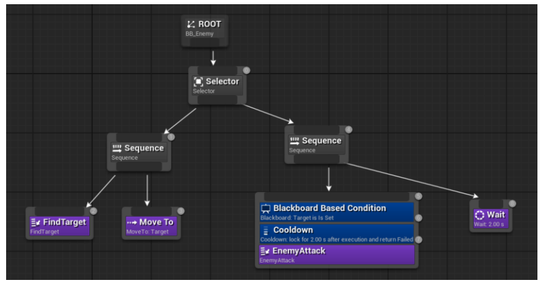
Milestone	2024년 12월		2025년 1월		
	3주차	4주차	1주차	2주차	3주차
기획	게임 선정				
프로토 타입		몬스터 (일반 , 보스) , 게임 사이클			
알파				아이템	
베타					인벤토리 , UI

액션 프로젝트

몬스터 2종

- 단순 근거리 1종 , 원거리 1종 구현
- C++ 로 베이스 클래스에 가상함수를 구현하여 블루프린트로 각각 기능 구현
- 공통된 BT로 AI 기능 구현

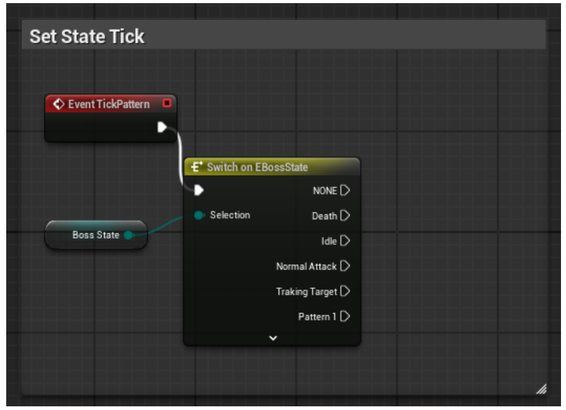




보스

- C++ 로 패턴의 쿨타임에 따라 행동 실행
- 각 패턴은 Enum를 사용하여 분리
- 각 패턴 기능은 블루프린트 또는 C++로 로직 수행

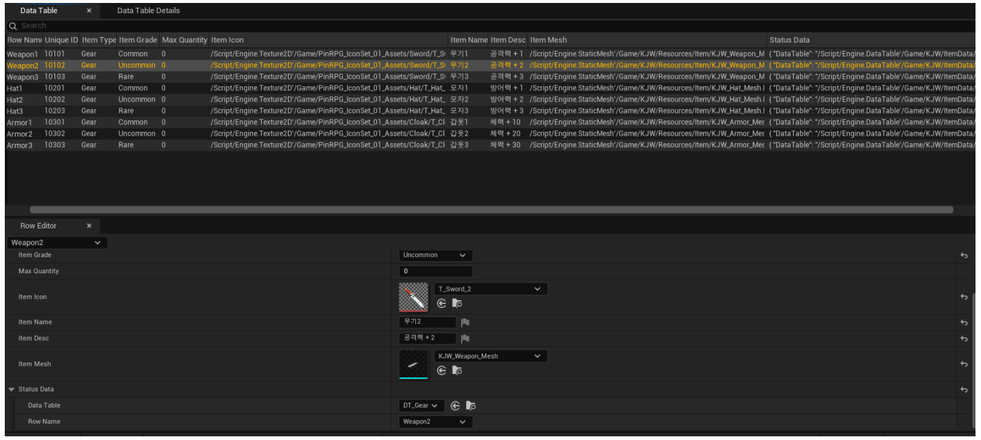

```
UENUM(BlueprintType)
enum class EBossState : uint8
{
    NONE UMETA(DisplayName = "NONE"),
    Death UMETA(DisplayName = "Death"),
    Idle UMETA(DisplayName = "Idle"),
    NormalAttack UMETA(DisplayName = "NormalAttack"),
    TrakingTarget UMETA(DisplayName = "TrakingTarget"),
    Pattern1 UMETA(DisplayName = "Pattern1"),
    Pattern2 UMETA(DisplayName = "Pattern2"),
    Stun UMETA(DisplayName = "Stun"),

};
```



액션 프로젝트

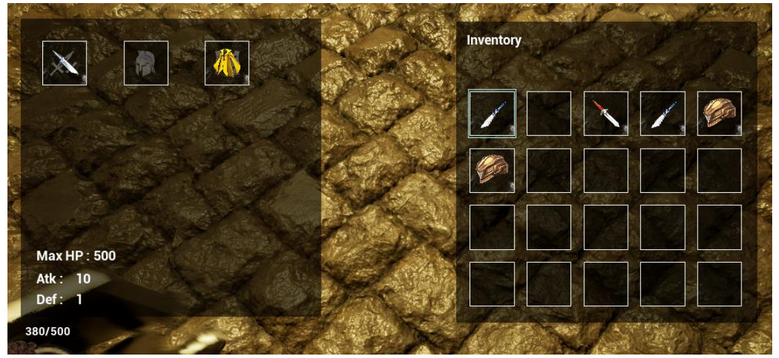
아이템

- UObect를 상속 받은 아이템 베이스 클래스 설계
- 아이템 데이터 구조 설계(등급, 아이콘, 수량 등)
- 아이템마다 상세 능력치 경우 별도의 테이블을 만들어 관리



인벤토리

- 각 슬롯을 누르고 누른 순서에 따라 필요한 로직 수행
  - 가방->장비창 : 아이템 장착
  - 장비창 -> 가방 : 아이템 장착 해제
  - 빈슬롯 경우 아무것도 하지않음
- 핵심 로직 구현은 GameInstanceSubsystem 을 활용하여 모든 곳에서 접근 하여 수행할수 있도록 구현




```cpp
class UUISlotBase;

UCLASS()
class TEAMPROJECT_POEKOR_API USlotGameInstanceSubsystem : public UGameInstanceSubsystem
{
    GENERATED_BODY()


public:
    bool IsOnClickedSlot;
    TWeakObjectPtr<UUISlotBase> ClickedSlot;
    TWeakObjectPtr<UWorld> SlotWorld;

    UFUNCTION(BlueprintCallable)
    void OnClikedSlot(UUISlotBase* NewClickedSlot);
    UFUNCTION(BlueprintCallable)
    void FailedMoveSlot();

protected:
    void ClearClickSlot();
    void SetNewClickSlot(UUISlotBase* NewClickedSlot);
    void MoveSlotEvent(UUISlotBase* FromSlot, UUISlotBase* ToSlot);


};
```

```cpp
void USlotGameInstanceSubsystem::MoveSlotEvent(UUISlotBase* FromSlot, UUISlotBase* ToSlot)
{
    EUISlotType fromSlotType = FromSlot->GetSlotType();
    EUISlotType toSlotType = ToSlot->GetSlotType();

    //Inven Item Swap
    if (fromSlotType == EUISlotType::Inven && toSlotType == EUISlotType::Inven){ ... }
    //Equip Item or UnEquip
    else if ((fromSlotType == EUISlotType::Inven && toSlotType == EUISlotType::Gear) ||
        (fromSlotType == EUISlotType::Gear && toSlotType == EUISlotType::Inven)){ ... }

}
```

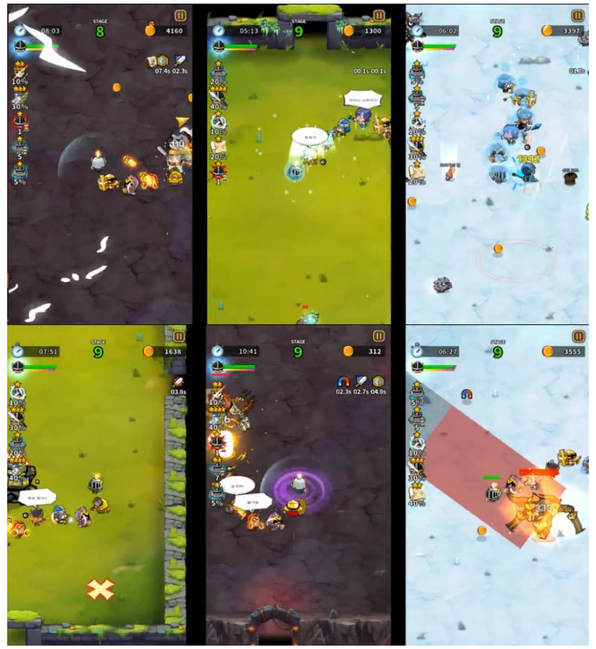
유니티 모바일 프로젝트

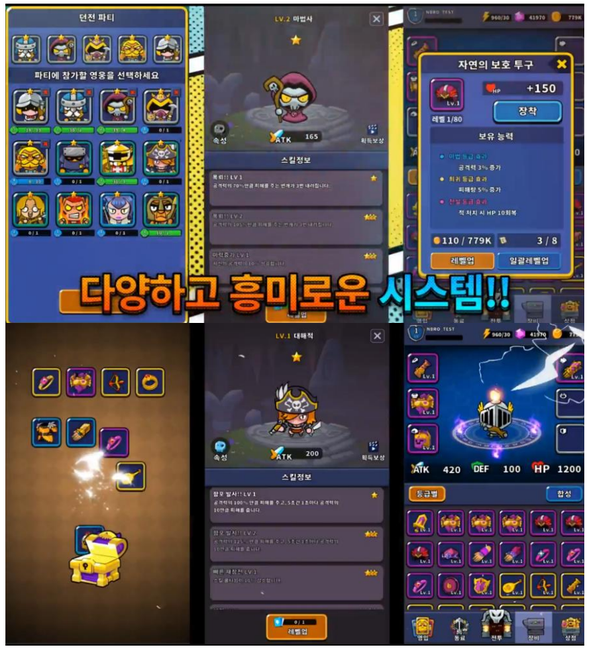
프로젝트 개요

회사 : IOI게임즈

작업 인원 : 클라이언트(1) , 아트(1), 기획(1)
본인 : 게임 프로토타입의 전투로직, 상점, 데이터 관리,
구글로그인 등 전반적인 모든 시스템

개발 기간 : 2023년 02월 ~ 2023년 11월





개발 일정

2023년	2~3월	4~7월	8~9월	10~11월
기획	기획안 수립			
프로토 타입		여러 프로토 타입 개발		
알파			프로토타입 업그레이드 디자인 적용	
베타				구글, 스토어 등록

THANK YOU

버그헌터 클라이언트 대체불가능한

근면성실 언제나 성장중

김재완