0 구역 (ZeroSector)

develop period	@2025년 3월 11일 → 2025년 6월 1일
roll	기획 팀장 프로그래밍 프로젝트 매니저
category	언리얼 게임 팀프로젝트
size	Large
FrameWork	Unreal
언어	C++

개요

- 전남대학교 4학년 1학기 캡스톤디자인 프로젝트를 통해 만든 게임
- 현재 개발 진행 중인 프로젝트라 설명이 미흡한 점 정말 죄송합니다. 아래 깃허브에 올라가 있는 이슈 및 코드를 봐주시면 감사하겠습니다.

전체 코드는 다음 깃에 올라가 있습니다.

```
https://github.com/SingABrro/ZeroSectorProject
```

게임 설명

게임의 스테이지는 크게 낮과 밤 두가지로 나뉘어져 있습니다.

낮에는 추리를 하는 맵으로 단서를 찾고 NPC들과 대화를 통해 진짜 치료제를 개발할 수 있는 연구원을 찾아갑니다.

밤에는 좀비와 전투를 하는 맵으로 주무기를 선택후 전투 스테이지로 이동하여 밀려오는 좀비 웨이브를 견뎌내고 모든 좀비를 섬멸해야 합니다.

스토리

생화학 무기로 활용하기 위해 좀비 바이러스를 연구하던 어느 군 소속 연구소. 바이러스에 '숙주의 경질화' 라는 치명적인 결함이 발견되어 프로젝트 존폐가 논의되던 상황.갑작스레 발현된 실험체들의 공격성에 연구소 구역이 습격을 받고, 확산 방지를 위해 각 섹터를 격리하게 된다.

연구소장은 상부에 지원을 요청하나 상부는 이를 빌미로 연구소를 통째로 은폐하기로 결정하고, 연구소의 폭격을 지시한다. 연구원들의 숙소가 있는 A구역에서 불침번을 서던 주인공(플레이어)은 A구역까지 넘어온 좀비를 마주치게 된다.좀비를 처

치하고 나니 소란에 잠에서 깬 연구원S에게 설명을 듣는다. 치료제 제조법을 연구한 연구원을 찾기 시작한다.

게임 목표

- 치료제 제작법을 알고 있는 연구원을 찾아서 탈출하기
  - 낮: 연구원 추리 + 무기 강화
  - 밤: 시나리오 진행도에 따른 구역 점령(좀비 처치)

기본 조작

기본 조작은 다음과 같습니다.

- 키보드

W, A, S, D	상하좌우 방향 이동
WA, WD, SA, SD	대각선 방향 이동
Q	무기 교체
R	장전
F	상호작용
Tab	개인 수첩
Esc	메뉴 선택 창

- 마우스 (밤 한정)

좌 클릭	탄 격발
우 클릭	정조준
마우스 휠	무기 교체

개발

git lfs를 사용해 작업물을 공유

본인파트

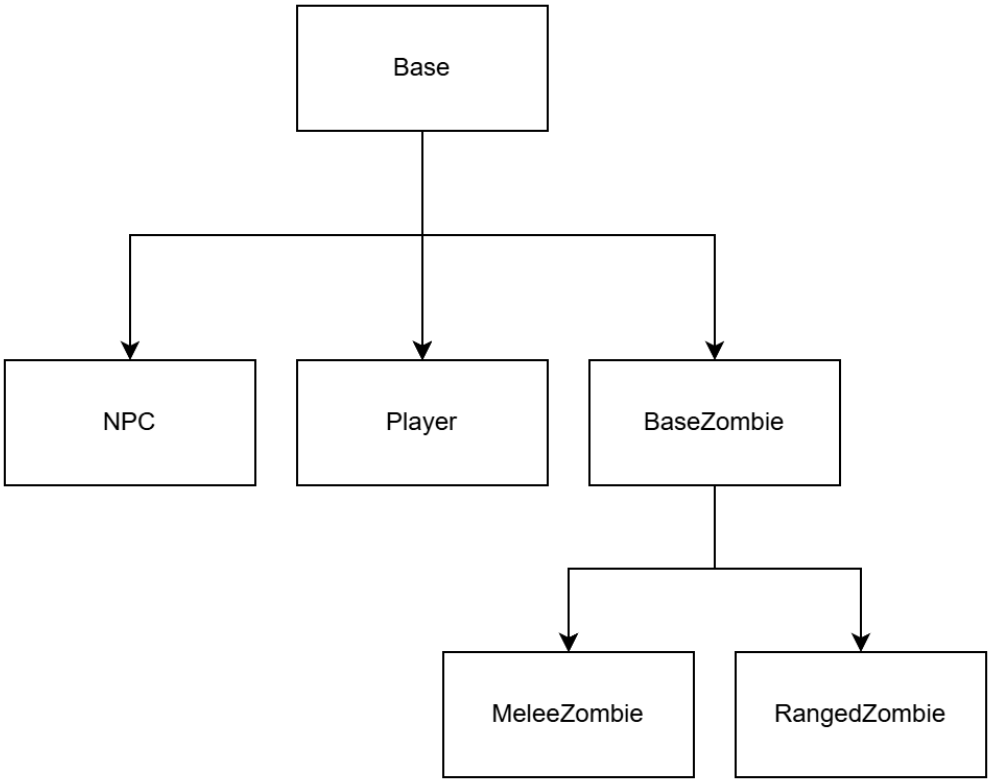
메인 메뉴, 개인 수첩 UI를 제외한 모든 파트

git branch 전략을 찾고 적용

- 플레이어
  - 기본 조작
  - 상호 작용
- 전투 시스템
  - 무기 개발 (권총, 라이플, 샷건)
  - 무기 강화

- 게임 시스템
  - 낮과 밤
  - 게임 규칙
- 대화 시스템
  - NPC와 플레이어 간 대화
- 스텟 시스템
- 상호작용 시스템
  - NPC 및 상호작용 가능한 액터 인식
- UI
  - HUD
  - 무기 강화 UI
  - Fade In & Out
- 몬스터
  - 좀비 AI
  - 좀비 공격
- 애니메이션
  - 캐릭터 애니메이션 블루프린트
  - 몬스터 애니메이션 블루프린트
  - 애니메이션 리타게팅
- Git Branch 전략
  - Trunk-Based

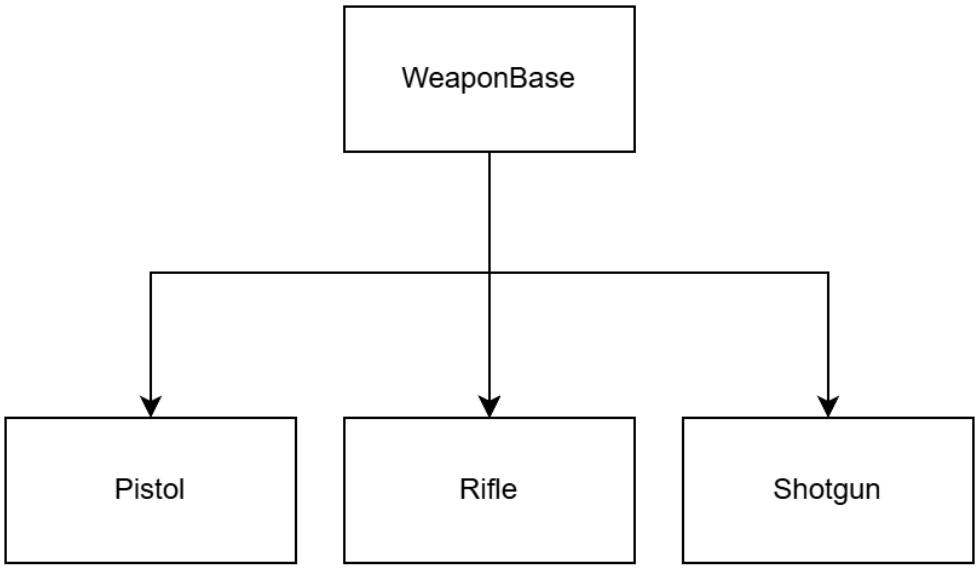
플레이어



- Character 클래스 상속 구조

- Base 클래스
  - IGenericTeamAgentInterface 를 구현해 NPC, Zombie, Player 팀을 구별
  - IZeroClassIdentifierInterface 를 구현해 각 객체마다 이름을 부여한 후 Map 컨테이너의 Key 값으로 이름 설정, Value 값으로 객체에 필요한 데이터 설정
- Player 클래스
  - Player 클래스 자체는 기능이 없으며, 각 기능별 컴포넌트를 구현해 Player 클래스에 부착함으로써 프로젝트 확장성, 디커플링을 확보
  - 낮과 밤에 사용하는 조작법이 다르기 때문에 낮 전용, 밤 전용 InputComponent를 생성
    - 이렇게 함으로써 낮에 필요한 데이터는 낮에만 메모리에 올라가 있으며, 밤에 필요한 데이터 또한 밤에만 메모리에 올라가게 할 수 있었음
  - Player가 상호작용을 통해 생성되는 UI Widget 을 관리하는 UIComponent 생성
  - Player의 스텟을 관리하는 StatComponent 생성
  - 각 컴포넌트들은 델리게이트를 만들어 Player 클래스에서 델리게이트에 필요한 함수를 바인딩하여 통신

전투 시스템



- Weapon 클래스 상속 구조

- 샌드박스 패턴을 사용하여 구현
- 원래는 상위 클래스에서 protected 이상의 접근 지정을 하여 하위 클래스에 기능을 제공해야 하지만, 언리얼은 리플리케이션 기능을 통해 상위 클래스의 함수를 같이 호출할 수 있기 때문에, 상위 클래스에서 제공해야 하는 기능을 private으로 캡슐화 한 뒤 하위 클래스에서 재정의 할 가상 함수에서 호출함으로써 구현

```
void AZeroWeaponBase::Fire()
{
    if (bIsFire || CurrentAmmo <= 0) return;
    bIsFire = true;

    switch (WeaponType)
    {
    case EWeaponType::EPistol:
        PistolFire();
        break;
    case EWeaponType::ERifle:
        RifleFire();
        break;
    case EWeaponType::EShotgun:
        ShotgunFire();
        break;
    default:
        return;
    }
}
```

- WeaponType을 통해 무기 종류 구별
- Fire() 가상 함수에서 현재 무기 종류에 따라 다른 공격을 호출함

- 각 무기의 스텟은 무기 액터가 생성된 뒤, 현재 무기 종류에 따라 DataTable에서 스텟을 가져와 적용
- 총기 반동, 탄창, 무기 대미지 적용 등 구현 완료

대화 시스템

1. 대화 컴포넌트 만들기
- UZeroDialogueComponent 생성

2. 대화 컴포넌트에 인터페이스 구현
- IZeroDialogueInterface를 UZeroDialogueComponent 가 구현

3. 캐릭터가 대화 컴포넌트가 있는 액터를 감지
- 콜리전을 만들어서 GetComponentByInterface<>(); 으로 Interface 포인터 받기
- Interface가 존재한다면 특정 키를 눌렀을 때 대화 진행

4. 대화 진행
- NPC 대화창 설정

```
void UZeroDialogueComponent::StartDialogue()
{
    RotationToPlayer();

    DialogueTable = UZeroSingleton::Get().GetDialogueTable(0);
  ZE_LOG(LogZeroSector, Warning, TEXT("Dialogue Loaded: %s"), *DialogueTable.Dialogue.ToString
());
    DialogueWidgetPtr = CreateWidget<UZeroDialogueWidget>(GetWorld(), DialogueWidgetClass);
    DialogueWidgetPtr→AddToViewport();
    DialogueWidgetPtr→SetDialogueText(ActorName, DialogueTable.Dialogue);

    if (DialogueTable.bIsOpenOption)
    {
        InputModeGameAndUI();
        for (const auto& DialogueOptionTable : DialogueTable.OptionDialogues)
        {
            DialogueOptionSpawn(DialogueOptionTable);
        }
        return;
    }
}
```

- 대화 데이터 테이블 가져오기
- UI 생성 및 표시
- 유저 대화 선택 창 표시하기

- 유저 대화창 설정

```
void UZeroDialogueComponent::DialogueOptionSpawn(const FZeroDialogueOptionDataTable& InDialo
gueOptionTable)
{
   DialogueOptionWidgetPtr = CreateWidget<UZeroDialogueOptionWidget>(GetWorld(), DialogueOpti
onWidgetClass);
   DialogueOptionWidgetPtr->SetDialogueComp(this);
   DialogueOptionDataInit(InDialogueOptionTable.DataTable, InDialogueOptionTable.RowIndex);
   DialogueWidgetPtr->GetScrollBox()->AddChild(DialogueOptionWidgetPtr);
   DialogueOptionWidgetPtr->SetDialogueOptionText(InDialogueOptionTable.OptionDialogue);
}
```

5. 유저가 대화 박스 선택 시

```
void UZeroDialogueComponent::OnClickedOption(FZeroDialogueDataTable InDialogueTable)
{
   DialogueWidgetPtr->GetScrollBox()->ClearChildren();

   DialogueTable = InDialogueTable;
   ZE_LOG(LogZeroSector, Warning, TEXT("Dialogue Loaded: %s"), *DialogueTable.Dialogue.ToString
());

   DialogueWidgetPtr->UpdateDialogue(DialogueTable.Dialogue);
   if (DialogueTable.bIsOpenOption)
   {
      for (const auto& DialogueOptionTable : DialogueTable.OptionDialogues)
      {
         DialogueOptionSpawn(DialogueOptionTable);
      }
   }
   else if (DialogueTable.bIsEnd)
   {
      DialogueWidgetPtr->GetScrollBox()->ClearChildren();

      FTimerHandle Timer;
      GetWorld()->GetTimerManager().SetTimer(Timer, [&]()
         {
            DialogueWidgetPtr->RemoveFromParent();
            OnFinishedDialogue.ExecuteIfBound();
            InputModeGameOnly();
         }, 3.f, false);
      DialogueTable = UZeroSingleton::Get().GetDialogueTable(0);
   }
}
```

- 남은 대화가 있다면 반복한다.
- 마지막 대화라면 대화 UI를 없애고 대화가 끝났음을 플레이어에게 알린다.

6. 대화 종료

```
DECLARE_DELEGATE(FOnFinishedDialogue)

class ZEROSECTOR_API IZeroDialogueInterface
{
   GENERATED_BODY()

public:
   virtual void StartDialogue() = 0;
   virtual void SetupFinishedDialogueDelegate(const FOnFinishedDialogue& InOnFinishedDialogue) =
0;
};
```

- UZeroDialogueComponent와 Player 캐릭터 양쪽 모두 참조하고 있는 인터페이스에 델리게이트를 하나 선언

```
//UZeroDialogueComponent
else if (DialogueTable.bIsEnd)
{
   DialogueWidgetPtr→GetScrollBox()→ClearChildren();

   FTimerHandle Timer;
   GetWorld()→GetTimerManager().SetTimer(Timer, [&]()
      {
         DialogueWidgetPtr→RemoveFromParent();
         OnFinishedDialogue.ExecuteIfBound();
         InputModeGameOnly();
      }, 3.f, false);
   DialogueTable = UZeroSingleton::Get().GetDialogueTable(0);
}

//AZeroCharacterPlayer
FOnFinishedDialogue OnFinishedDialogue;
OnFinishedDialogue.BindLambda([&]()
   {
      SetDefaultMovement();
   });
DialogueInterface→SetupFinishedDialogueDelegate(OnFinishedDialogue);

SetDialogueMovement();
```

- 델리게이트에 람다함수를 바인딩
- 대화의 끝에 다다르면 타이머를 이용해 몇 초 후 델리게이트 실행

대화 시스템 사용법

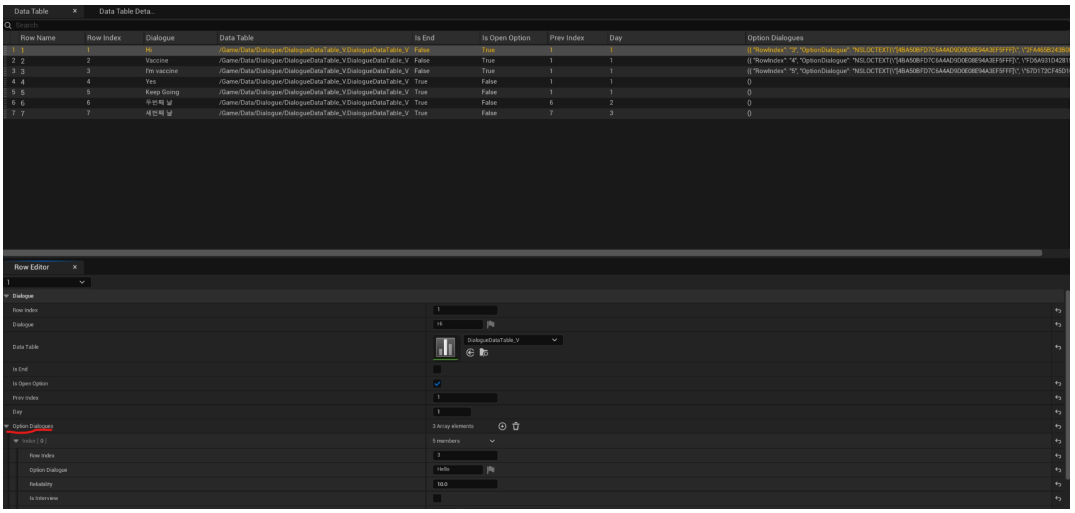
데이터 테이블

- Row Name : 해당 행을 가리키고 있는 키
- Row Index : 이 대사 다음에 나올 대사를 가리키는 키

- Dialogue : 대화 창에 나올 대사
- Data Table : NPC 자기 자신의 Data Table 집어 넣으면 됨
- Is End : 박스 체크 시 해당 대사가 대화의 끝을 나타냄
- Is Open Option : NPC의 대화에 Player가 답변을 할 수 있음
- Prev Index : NPC한테 처음 말걸 때 나와야 하는 대사를 가리킴
- Day : 몇 번째 날의 대사인지 나타냄
- Option Dialogues : Player가 선택할 수 있는 대화 데이터 테이블

Player가 선택할 수 있는 대화 데이터 테이블을 보는 방법은 NPC의 대화 데이터 테이블에서 한 행을 선택하고 아래 Row Editor에 있는 Option Dialogues를 펼쳐보면 됨

+버튼을 누르면 추가할 수 있음 (요소 하나당 Player의 대화 선택지가 하나 늘어남)



Player가 선택할 수 있는 대화 데이터 테이블

- Row Index : Player가 이 대사를 선택하면 다음 나올 NPC의 대사를 가리키는 키
- Option Dialogue : 대화 선택 창에 뜨는 대사
- Reliability : 신뢰도 (20으로 설정되어 있으면 해당 대사 선택시 대화중인 NPC의 신뢰도가 20오름)
- Is Interview : 면담 요청시 나올 대사 (True라면 면담 요청하고 대화했을 때 해당 대화 선택지가 보임)
- Data Table : 현재 수정중인 Data Table 삽입

- 1일차 대사부터 7일차 대사까지 순서대로 데이터 테이블에 넣으면 사용 가능

- NPC → Player 로의 단방향 대화는 다음과 같이 데이터를 넣으면 됨(Speedwagon 대화 테이블)

	Row Name	Row Index	Dialogue	Data Table	Is End	Is Open Optio	Prev Inde	Day	Option Dialogues
1	1	2	Hi	/Game/Data/Dialogue/DialogueDataTable_S.DialogueDataTable_S	False	False	1	0	()
2	2	3	Speedwagon	/Game/Data/Dialogue/DialogueDataTable_S.DialogueDataTable_S	False	False	1	0	()
3	3	4	I'm Speedwagon	/Game/Data/Dialogue/DialogueDataTable_S.DialogueDataTable_S	False	False	1	0	()
4	4	4	Yes	/Game/Data/Dialogue/DialogueDataTable_S.DialogueDataTable_S	True	False	1	0	()
5	5	5	Keep Going	/Game/Data/Dialogue/DialogueDataTable_S.DialogueDataTable_S	True	False	1	0	()