Throne

develop period	@2024년 8월 2일 → 2024년 8월 29일
roll	프로그래밍
category	개인프로젝트 게임
size	Large
FrameWork	Unreal
언어	C++

개요

"언리얼" 공부를 진행하며 개인으로 만든 간단한 데모 게임입니다.

- 프로젝트 이름 : Throne

- 엔진 : Unreal Engine 5.4
- 장르 : 3D 3인칭 소울류
- 개발 인원 : 1명
- 제작 기간 : 2024.08.04 ~ 2024.08.28

전체 코드는 다음 깃에 올라가 있습니다.

https://github.com/kimkyungjae1112/Throne

게임설명

Throne은 소울류 게임을 참고해 만든 3인칭 시점으로 던전을 탐색하는 게임입니다.

맵을 탐색하고 몰려오는 적들을 물리치며, 맵 곳곳에 숨은 보상을 찾아 강해질 수 있습니다.

마지막으로 최종 보스를 잡으면 게임이 끝나게 됩니다.

플레이 영상

https://prod-files-secure.s3.us-west-2.amazonaws.com/7ebf7000-f855-492a-973c-2e002d905ac7/4a4583a7-42d1-4a3a-b9b4-5a46eae70cc4/%ED%99%94%EB%A9%B4_%EB%85%B9%ED%99%94_%EC%A4%91_2025-01-19_224605.mp4

개발

모든 요소를 C++을 이용해 만들고자 하였고, 모르는 부분은 강의 영상과 구글링을 통해 채워나갔습니다.

소스코드

▼ AI

AIController

AIController는 NPC 폰에 빙의되어 플레이어가 직접 조종하지 않아도 폰의 동작을 가능하게 합니다. 저는 BehaviorTree를 사용해 NPC의 동작을 결정하기 위해 AIController에 다음과 같은 코드를 작성했습니다.

```
public:
    void RunAI();
    void StopAI();
```

AIController의 조종을 받는 폰에서 BehaviorTree의 실행을 On/Off 할 수 있도록 RunAI()와 StopAI()를 제공합니다. 빙의시 바로 AI가 작동할 수 있도록 OnPossess 가상 함수에서 RunAI를 호출하고 있습니다.

BTDecorator_AttackInRange

UBTDecorator 클래스를 상속받아 제작했으며 AI의 공격범위 안에 Target(플레이어)가 있는지 검사하는 클래스입니다.

BTService_Detect

UBTService 클래스를 상속받아 제작했으며 AI의 탐지범위 안에 Target(플레이어)가 있는지 검사하는 클래스입니다. UWorld 클래스에 있는 OverlapMultiByChannel을 사용해 Player 채널에 있는 폰을 검사하도록 설계했습니다. 해당 채널에 감지되는 폰이 있다면 Blackboard의 Target 변수에 해당 폰을 대입하고 감지 되지 않았다면 nullptr을 대입합니다.

BTTask

UBTTask_Node를 상속받아 만든 클래스들입니다.

BTTask_Attack

AI의 공격 행동을 지시하는 클래스입니다. AIInterface에 선언된 델리게이트와 함수를 먼저 보면

```
DECLARE_DELEGATE(FAIMonsterAttackFinished)

virtual void AttackByAI(class UAnimMontage* InAnimMontage) = 0;
```

해당 코드들이 Attack 클래스와 함께 호출됩니다. 모든 AI들은 기본 클래스인 Enemy를 상속받게 됩니다. Enemy는 AIInterface를 구현하고 있습니다. 그래서 Enemy를 상속받은 AI들이 AttackByAI() 함수를 재정의하여 Montage 포인터를 넘겨주기만 하면 해당 몽타주를 실행하도록 설계했습니다.

공격 몽타주가 끝나면 FAIMonsterAttackFinished 델리게이트를 실행하여 Attack 클래스가 Succeeded를 리턴하며 끝납니다.

BTTask_TurnToTarget

AI가 공격을 진행할 때 Target(플레이어)를 바라볼 수 있게 하는 클래스입니다. RInterpTo() 보간 함수를 사용하여 Target을 향해 서서히 회전하도록 설계했습니다.

BTTask_PatrolToFind

AI의 패트롤 기능을 담당하는 클래스입니다.

BTTask_Aiming

궁수 몬스터의 활 시위를 당기는 연출을 하기 위한 클래스입니다. 시위를 당기는 몽타주가 끝나면 끝Succeeded를 리턴하며 끝납니다.

BTTask_StopAnimation

궁수 몬스터의 Aiming 클래스가 진행되던 중 Target(플레이어)가 탐지 범위 밖으로 나갔을 시 활 시위를 당기던 몽타주를 끝내기 위한 클래스입니다. Aiming 클래스를 진행하며 손에 화살이 생성되게 되는데 그것 또한 삭제합니다.

▼ Notify

피격 판정을 체크하는 클래스

공격 몽타주 중간에 들어가 공격 판정을 하기 위한 클래스들입니다.

각 클래스들은 인터페이스를 통해 공격의 주체인 Character 클래스들과 연결되어 있습니다.

AttackHitCheckNotify

캐릭터의 기본 공격에 대한 공격 판정입니다. 구현은 인터페이스를 통해 이루어집니다.

JumpAttackDoneNotify

캐릭터의 점프 공격에 대한 공격 판정입니다. 구현은 인터페이스를 통해 이루어집니다.

KickAttackNotify

캐릭터의 킥 공격에 대한 공격 판정입니다. 구현은 인터페이스를 통해 이루어집니다.

BossAttackHitCheckAnimNotify

보스 몬스터의 기본 공격에 대한 공격 판정입니다. 구현은 인터페이스를 통해 이루어집니다.

EnemyAttackHitCheckNotify

기본 몬스터의 기본 공격에 대한 공격 판정입니다. 구현은 인터페이스를 통해 이루어집니다.

ArcherArrowSpawnNotify

해당 클래스는 궁수 몬스터의 활 시위를 당기고 발사하는 몽타주 중간에 화살을 생성하기 위한 클래스입니다. 인터페이스를 통해 궁수 몬스터와 연결되어 있습니다. 구현은 인터페이스를 통해 이루어집니다.

▼ Animation

CharacterAnimInstance

캐릭터의 상태를 관찰하고 Animation_Blueprint에서 상태에 따라 애니메이션을 변경하기 위해 변수들을 저장하는 클래스입니다.

땅 위에서 움직임, 전투 상태, 방패를 든 상태, 사다리를 타고 있는지, 나이프를 조준하고 있는지 등을 NativeUpdateAnimation() 함수를 통해 업데이트 합니다.

▼ Character

ThroneCharacter

캐릭터 클래스는 입력 처리를 담당하고 있으며, 기능들은 최대한 컴포넌트에 구현해 커플링을 지양하고 클래스가 방대해 지는 것을 막고자 하였습니다.

컴포넌트

실질적인 기능 구현을 담당하고 있는 컴포넌트들입니다.

- AbilityComponent

    - 전투부터 상호작용까지 캐릭터가 행하는 모든 행동을 구현하고 있습니다.
- StatComponent
    - 캐릭터의 스텟을 관리하는 컴포넌트입니다.

상태

캐릭터는 3가지의 상태를 가지며 각 상태에 따라 각기 다른 애니메이션과 입력을 가집니다.

- 기본(Default) 상태
    - 무기를 들고 있지 않은 상태며 게임이 시작할 때 갖는 기본 상태값 입니다.
- 전투(HoldWeapon) 상태
    - 무기를 들고 있는 상태며 해당 아이템 획득시 가질 수 있는 상태값 입니다.
    - 무기를 가진 상태라면 기본 상태와 전투 상태를 왔다갔다 할 수 있습니다.
- 사다리(Ladder) 상태
    - 사다리에 타있는 상태이며 움직임이 사다리를 타는 것으로 제한됩니다.

입력

입력은 기본 상태, 전투 상태, 사다리에 타있는 상태에 따라 변경됩니다.

기본, 전투 상태에서 공통 입력

- WASD - 움직임
- Mouse XY - 화면 전환
- SpaceBar - 점프
- E - 상호 작용
- F - 킥 공격
- Q - 무기 칼집에서 꺼내기 및 집어넣기

전투 상태

- Mouse Left Click - Knife 투척(조준 상태)
- Mouse Right Click - Knife 조준

무기 상태

- Mouse Left Click - 기본 공격, 점프 상태에선 내려 찍는 공격
- Mouse Right Click - 방어

사다리 상태

- 사다리를 탈 때에는 InputMappingContext를 교체합니다.
- WS - 위, 아래 움직임
- Mouse XY - 화면 전환
- E - 상호 작용

상호작용

무기 장착 상호작용을 위한 함수인 AcquisitionItem를 제외하고 나머지 상호작용들은 해당 상호작용 오브젝트의 BoxCollision 안에 들어갔을시 캐릭터가 가지고 있는 포인터에 대입되고 동작하도록 설계했습니다. 박스에서 벗어날 시 포인터는 nullptr 이 되며 상호작용 함수들에서 null 검사를 먼저 진행합니다. 또한 캐릭터가 상호작용 범위 안으로 들어갈시 UI가 표시됩니다.

- 레버 상호작용
- Door 상호작용
- DragonGate 상호작용
- 사다리 상호작용
- 무기 장착 상호작용
- 상호작용 범위 안으로 들어올시 나타나는 UI

HUD

캐릭터의 현재 스텟에 따라 동적으로 변화하는 UI 입니다. Tick에서의 구현을 통해 일정 시간이 지나면 자동으로 회복되도록 구현했습니다.

- 체력바
- 기력바

▼ Component

캐릭터와 적 클래스에 부착하는 컴포넌트 클래스입니다.

캐릭터와 적 클래스의 스텟을 관리하며 캐릭터 기능을 담당하는 Ability Component가 있습니다.

AbilityComponent

캐릭터의 공격, 스킬과 각종 상호작용을 하는 기능을 정의하고 있습니다.

CharacterStatComponent

캐릭터의 스텟을 관리하는 클래스입니다.

EnemyStatComponent

적의 스텟을 관리하는 클래스입니다.

▼ Enemy

적 클래스

적들의 스켈레탈 메쉬, 공격, 공격 판정, 히트 판정, 애니메이션 몽타주 등 정보를 가지고 있는 클래스입니다.

모든 적은 Enemy 클래스를 상속받으며 Enemy 클래스는 AIInterface를 상속받아 구현함으로써 Enemy 계통 클래스들에게 필요한 기본적인 데이터들을 줄 수 있습니다.

근거리 계열 몬스터들의 공격 판정은 특정 각도 안에 캐릭터가 있을 시 작동합니다.

EnemyArcher

궁수 몬스터입니다. 원거리에서 화살을 쏘며, 활을 들고 움직이는 애니메이션을 비롯해 공격, 피격 애니메이션들이 구현되어 있고 캐릭터와 대미지를 주고 받을 수 있습니다.

EnemyKnight

검사 몬스터입니다. 근거리에서 캐릭터와 싸우며 궁수 몬스터와 마찬가지로 애니메이션들과 대미지를 주고 받는 기능이 구현되어 있습니다. 블루프린트에서 다른 스켈레탈 메쉬로 캐릭터의 외형 및 무기를 바꿀 수 있습니다.

EnemyBoss

검사 몬스터와 동일합니다.

▼ Gimmick

캐릭터와 상호작용 이벤트 클래스

캐릭터와 상호작용할 수 있는 물체들의 클래스입니다.

캐릭터가 해당 클래스로 만들어진 인스턴스의 일정 범위 안으로 들어갔을 시 인터페이스를 통해 캐릭터 클래스가 해당 클래스의 포인터를 가지게 됩니다.

추가로 UI가 띄워지며 상호작용을 할 수 있습니다.

Door

enum class 를 사용해 열림, 닫힘 상태를 검사하고 보간 함수를 이용해 부드러운 회전을 구현했습니다.

DragonGate

enum class 를 사용해 문의 크기를 나눴습니다. 크기에 따라 캐릭터가 문을 여는 모션이 달라지며 캐릭터는 DragonGate에 부착된 SceneComponent의 좌표로 이동하여 문을 여는 모션이 자연스럽게 이어질 수 있도록 합니다.

GateLever

Portcullis라는 문을 열기 위한 게이트 레버로, enum class 를 사용해 열림, 닫힘 상태를 검사합니다. 현재 상태에 따라 애니메이션이 달라지며, Portcullis 또한 열리고 닫힙니다. 캐릭터는 GateLever에 부착된 SceneComponent의 좌표로 이동하여 레버를 당기는 모션을 자연스럽게 구현합니다.

Ladder

`OnConstruction` 함수를 이용해 월드에 사다리를 쉽게 만들 수 있도록 했습니다. 사다리의 맨 위와 아래에 BoxCollision을 설치하고 캐릭터가 사다리를 타거나 내려올 때 BoxCollision에 바인딩된 델리게이트 함수가 실행되도록 했습니다. 바인딩된 함수들의 역할은 UI를 띄우거나, 캐릭터가 현재 사다리를 오르고 있는지 검사를 수행합니다.

캐릭터는 사다리를 다 올랐을 때, 내려올 때, 오르락 내리락하는 애니메이션을 현재 상태에 맞춰 재생합니다.

Portcullis

GateLever의 static 델리게이트를 받아와서 열리고, 닫히는 함수를 실행합니다.

하지만 static 델리게이트에 함수를 등록해 월드에 있는 모든 문이 함께 열리고 닫히는 현상을 해결하지 못했습니다.

WeaponBox

해당 인스턴스의 일정 범위 안으로 들어가면 ItemData를 캐릭터가 받을 수 있으며 상호작용 키를 누를 시 캐릭터가 무기를 가지고 있게 됩니다.

BossStartTrigger

보스 스테이지에 입장 했을 때 보스의 AIController가 작동이 되고 보스의 체력바를 띄우는 역할을 합니다.

▼ Interface

인터페이스

1. 캐릭터나 적의 공격을 판정하기 위한 Notify 클래스와 클래스, 적 클래스를 이어주는 인터페이스
2. 캐릭터의 HUD 위젯을 위한 인터페이스
3. AI가 작동 시 필요한 인터페이스
4. 궁수 몬스터의 공격 판정을 위한 인터페이스

인터페이스는 캐릭터와 같이 큰 클래스를 직접 참조하지 않으며 확장의 편의성을 위해 사용합니다.

▼ UI

UI

캐릭터의 체력바, 기력바, 보스 몬스터의 체력바를 정의하고 있습니다.

HpBarWidget

캐릭터의 체력과 비례하여 체력바를 조정합니다.

EnergyBarWidget

캐릭터의 기력과 비례하여 기력바를 조정합니다.

BossHpBarWidget

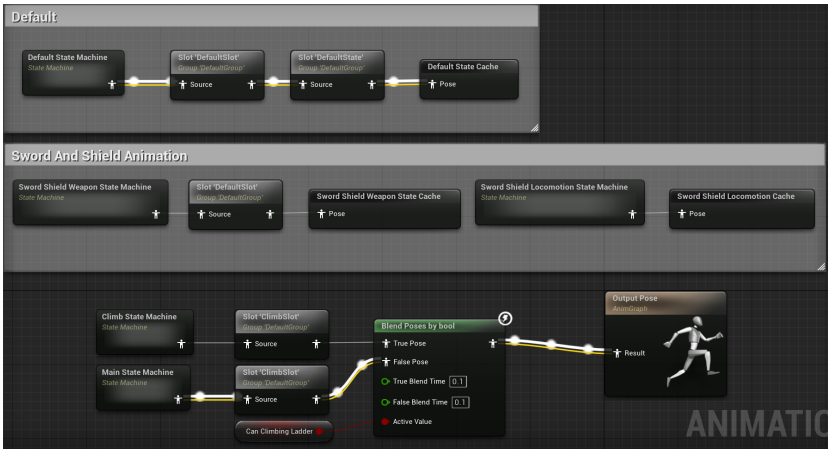
보스의 체력과 비례하여 체력바를 조정합니다.

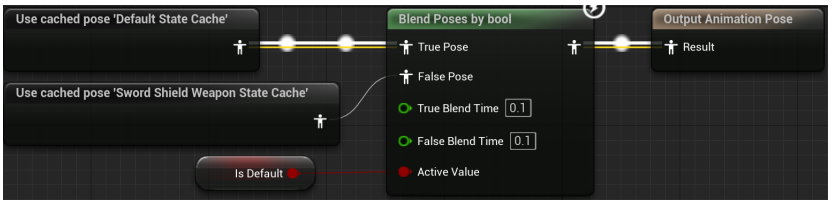
HUDWidget

HpBarWidget과 EnergyBarWidget을 포인터로 받아 게임 화면에 띄워줍니다. 캐릭터 스텟 컴포넌트의 델리게이트에 체력바, 기력바를 조정하는 함수를 등록하여 체력바와 기력바가 현재 체력과 기력에 비례해 조정될 수 있도록 구현했습니다.

애니메이션

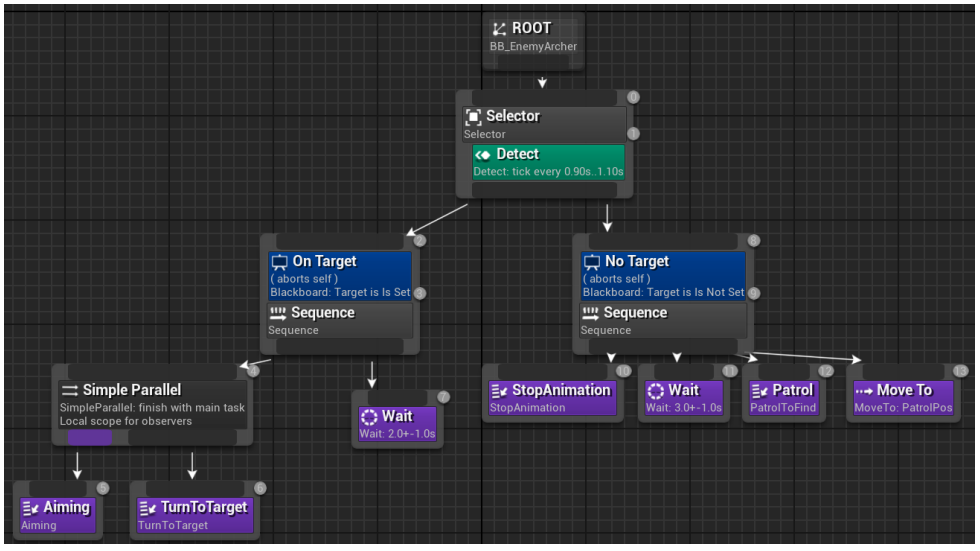
캐릭터 애니메이션 블루프린트



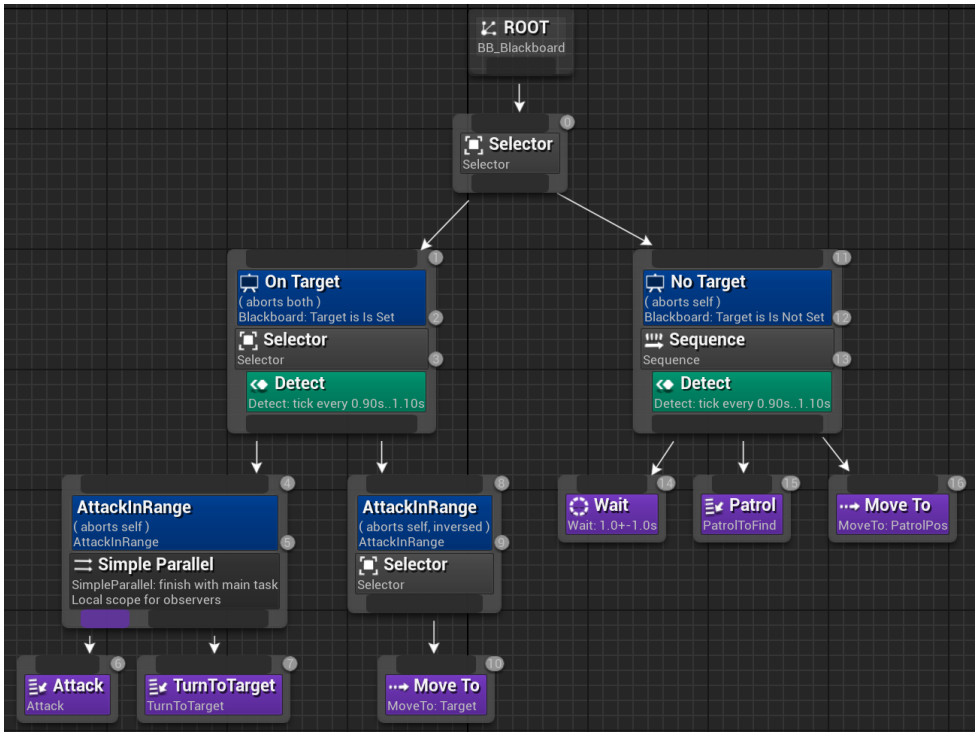


Behavior Tree

궁수 몬스터의 Behavior Tree



일반 몬스터의 Behavior Tree



회고

▼ 새로 알게된 사실들

OnConstruction

```
virtual void OnConstruction(const FTransform& Trasnform)
```

- Actor 클래스에 public 으로 선언된 가상함수이다.
- 에디터에서 액터의 변화를 바로 살펴볼 수 있다.
- 액터의 Transform이 변화할 때 호출된다.

액터에 컴포넌트를 붙이는 시점은 세 부분 정도로 나눠 볼 수 있다.

1. Class Construction 과정
2. Blueprint Construction 과정
3. Begin Play 혹은 실행 중 과정

함수	설명	연계되는 호출
virtual PostLoad()	에디터나 게임 플레이 도중 혹은 디스크에서 로드되어 배치된 경우 제일 처음으로 호출된다.	UActorComponent::OnComponentCreated()로드된 경우를 제외하고 모든 생성된 컴포넌트에 호출된다.
virtual PreRegisterAllComponents()	Actor의 생성 도중 호출되며, native root component를 가지고 있는 상태이다.	UActorComponent::RegisterComponent()native **root** components를 가지고 있는 상태에서 호출된다.

virtual PostRegisterAllComponents()	모든 Actor 컴포넌트가 등록되고 호출딘다.	
virtual PostActorCreated()	에디터나 게임 플레이 도중 생성되었을 때 호출되며 레벨에 의해 로드된 경우 호출되지 않는다.	
virtual OnConstruction(const FTransform&)	블루프린트 생성을 호출하는 ExecuteConstruction의 끝으로 호출되며 모든 블루프린트로 생성되는 컴포넌트가 생성되고 등록된다. 항상 게임 플레이 중 스폰된 Actor에만 호출된다.	
virtual PreInitializeComponents()	컴포넌트의 InitializeComponent 직전에 호출된다.게임 플레이 중에만 호출	UActorComponent::Activate(bool bReset): 컴포넌트의 bAutoActivate가 true 인 경우 호출된다. UActorComponent::InitializeComponent(): 컴포넌트의 bWantsInitializeComponent가 true 인경우 호출된다.
virtual PostInitializeComponents()	Actor 컴포넌트가 완전히 초기화된 이후 호출된다. 게임 플레이 중에만 호출	
virtual BeginPlay()	레벨이 Tick되기 시작할 때 호출된다. 게임 플레이 중에만 호출	

Spline

- UPrimitiveComponent를 상속받아 만들어진 컴포넌트
- 클래스의 이름은 USplineComponent 이다.
- 정해둔 포인트를 기준으로 움직이게 한다거나, 사다리처럼 일정 거리를 기준으로 메쉬를 생성할 수 있다.

Static Delegate

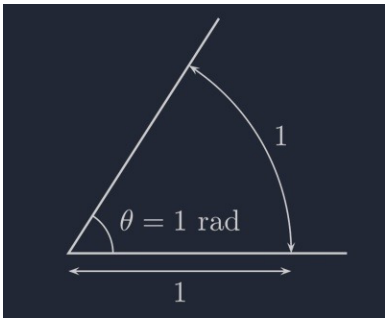
- 델리게이트를 전역으로 선언할 수 있다.
- 어떤 클래스에서든 헤더 파일을 포함한 후 ClassName::DelegateStructName 으로 접근할 수 있다.

Degree & Radian

Degree

- 각은 같은 끝점을 갖는 두 반직선이 이루는 도형이다.
- 각의 두 변이 벌어진 정도, 각의 크기를 나타내는 양을 각도라고 한다.

Radian



- 호의 길이가 반지름과 같게 되는 만큼의 각을 1 라디안(radian)이라고 정의한다.
- 라디안도 각도와 마친가지로 절대적인 각도의 단위이고, 1 radian 은 약 57.3도에 해당한다.

Degree와 Radian의 관계

- 파이 = 3.141592... * 지름
- 180 degree = 파이 radian
- 1 degree = 파이 / 180 radian
- 파이 radian = 180 degree
- 1 radian = 180 / 파이 degree

아쉬운점 (2024.11.14)

1. 사운드나 이펙트, UI 등을 통해 게임의 완성도를 채우지 못한점
2. 전투시스템의 완성도
    a. 방어판정을 개발하지 않은점 → IsShield 상태인지 검사해서 대미지를 감소시킨 뒤 적용하면 될 것같다.
    b. 공격판정이 너무 단조롭다. → 무기 소켓의 트랜스폼을 가져와서 LineTrace로 닿았는지 검사하는 것도 고려
3. 적 AI 의 완성도 부족
    a. AI Perception, EQS 등의 사용 부재

의문점 (2024.11.14)

1. 상호작용을 위해 상호작용 할 객체의 포인터를 캐릭터가 가지고 있는게 맞는가?