언리얼5을 활용한 게임개발

팀 프로젝트 개발과정

프로젝트 개요

개발 인원 : 4인
사용 엔진 : Unreal Engine 5.4.3
개발 환경 : Window 10, Visual Studio 2022
개발 기간 : 14일
사용 언어 : C++
기본 아키텍처 : 클래스와 객체 관계

프로젝트 구조

1.플레이어 시스템(Player System)
2.AI 및 적 시스템(AI and Enemy System)
3.UI시스템(User Interface System)
4.사운드 및 비주얼 이펙트(Sound and Visual Effects)
5.데이터 및 통계 관리(Data and Stats Management)
6.상점 관리(Store Management)

1.Player System은 게임에서 플레이어 캐릭터를 제어하고 상호작용하는 모든 기능을 담당하는 모듈입니다.

이 시스템의 주요 요소는 크게 5가지라고 할 수 있습니다.

1-1.플레이어 컨트롤러(Player Controller)

1-2.플레이어 캐릭터(Player Character)

1-3.플레이어 상태 컴포넌트(Player Stat Component)

1-4.인벤토리 시스템 (Inventory System)

1-5.플레이어 UI (User Interface)

1-1플레이어 컨트롤러 (Player Controller):
플레이어의 움직임,공격,상호작용 등을 제어합니다. 구체적으로 UInputComponent와 UEnhancedInputComponent를 사용하여 다양한 입력 액션을 바인딩하고 이 입력에 대응하는 함수들을 호출합니다.

1-2 플레이어 캐릭터((Player Character)
게임 내에서 플레이어가 조종하는 실제 캐릭터로써 핵심 기술로는 UEnhancedInputLocalPlayerSubsystem 으로 입력작업을 처리하고 관리합니다.

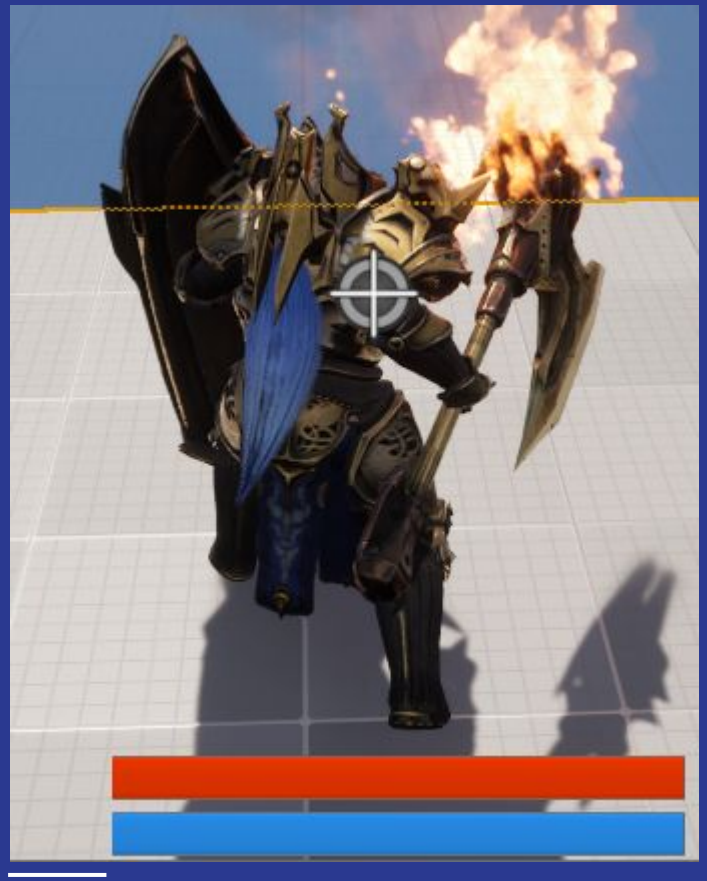
1-3 플레이어 상태 컴포넌트(Player Stat Component)
캐릭터의 체력,마나,경험치 등과 같은 상태를 관리하는 컴포넌트로 레벨업이나 아이템 사용에 따라 캐릭터의 능력치를 변화시킬 수 있습니다.

플레이어의 핵심 기술인 입력처리 부분

```
EnhancedInputComponent->BindAction(_moveAction, ETriggerEvent::Triggered, this, &ATFT_Player::Move);
```

1-4인벤토리 시스템(Inventory System) 플레이어가 소지할 수 있는 아이템들을 관리하는 시스템으로 UI와 연동되어 시각적으로 확인가능합니다.

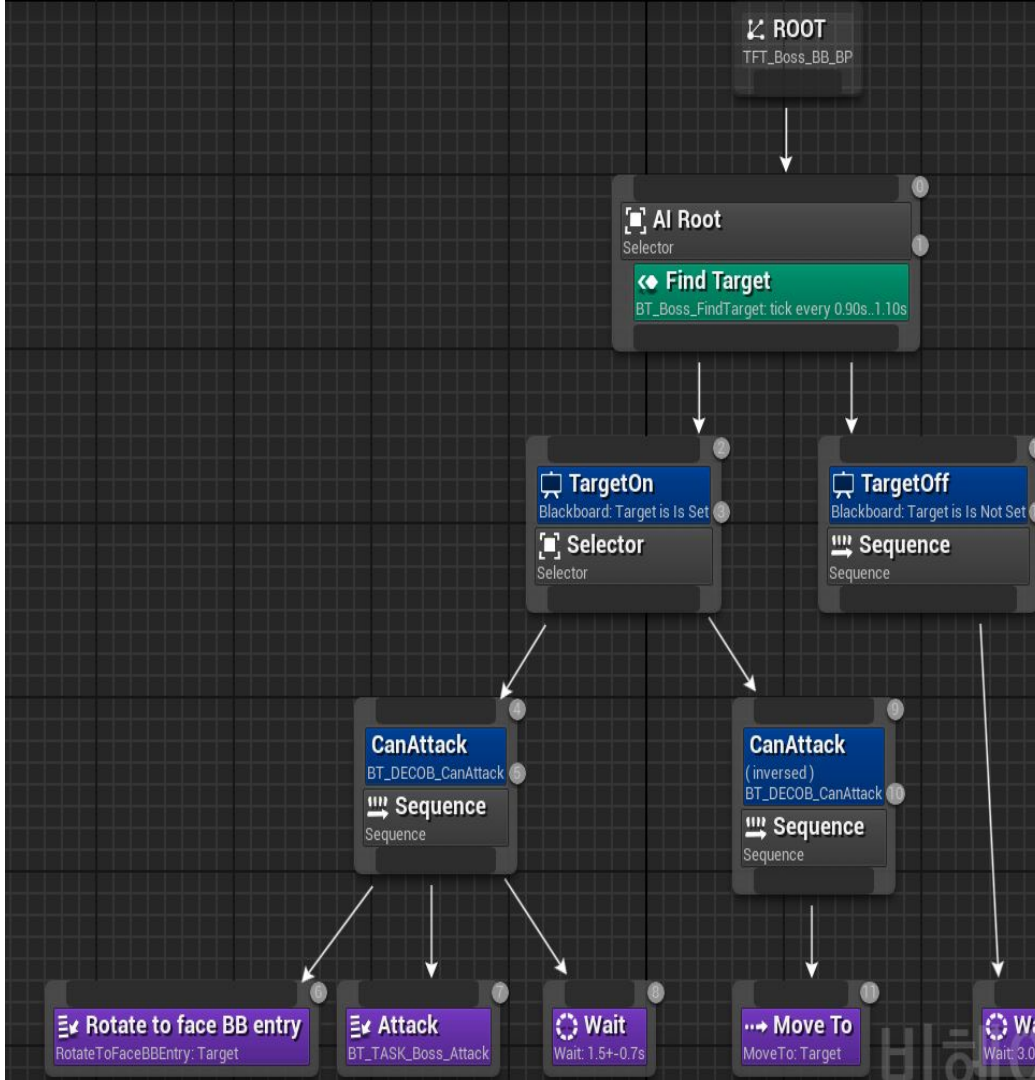
1-5플레이어 UI(User Interface) 게임 화면에 표시되는 플레이어의 상태,인벤토리 등을 보여주는 사용자 인터페이스 입니다.



2. AI 및 적 시스템

AI시스템은 비헤이비어 트리(Behavior Tree)를 중심으로 설계하여 복잡한 행동을 간단하고 직관적으로 설계했습니다.

이 시스템의 핵심은 보스AI로 체력이 일정 이하로 떨어지면 강력한 공격을 수행하거나 가장 많은 피해를 준 플레이어NPC나 플레이어를 타겟우선순위를 만들었습니다.



3.UI시스템(User Interface System)

UI시스템은 플레이어와 밀접한 관계로 캐릭터의 상태나 인벤토리 등을 보여주는 부분입니다. 핵심 부분은 Interact함수로 interact함수는 플레이어와 상호작용할 때 호출되며 인터페이스를 통해 아이템을 인벤토리에 추가하는거나 대화를 할 수 있는 등 다양한 상호작용을 합니다

```
void ATFT_Player::Interact(ATFT_Item* item)
{
    UE_LOG(LogTemp, Log, TEXT("Player Interact With NPC!"));

    AddItemPlayer(item);
}
```

4.사운드 및 비주얼 이펙트 (Sound and Visual Effect)

사운드는 CreateSoundCue함수를 통해 사운드의 주소를 가져와서 애니메이션과 map에서 사운드가 재생될 수 있도록 블루프린트로 추가해줬고 이펙트도 CreateParticleClass함수를 이용해서 주소를 가져오고 play함수에서 특정 이름을 가진 이펙트를 재생합니다

```
CreateSoundCue("Knight_Choice", "/Script/Engine.SoundCue'/Game/P
CreateSoundCue("Archer_Choice", "/Script/Engine.SoundCue'/Game/P
CreateSoundCue("Knight_Swing", "/Script/Engine.SoundCue'/Game/Bl
CreateSoundCue("Archer_Shoot", "/Script/Engine.SoundCue'/Game/Bl
CreateSoundCue("Monster_Normal_Swing", "/Script/Engine.SoundCue'
CreateSoundCue("Monster_Boss_Swing", "/Script/Engine.SoundCue'/G
CreateSoundCue("TeamAI_Knight_Swing", "/Script/Engine.SoundCue'/
```

```
CreateParticleClass(TEXT("P_Explosion"), TEXT("/Script/Engine.Blueprint'/G
CreateParticleClass(TEXT("N_Knight_Attack_Hit"), TEXT("/Script/Engine.Blue
CreateParticleClass(TEXT("N_Archer_Attack_Hit"), TEXT("/Script/Engine.Blue
CreateParticleClass(TEXT("N_Monster_Boss_Attack_Hit"), TEXT("/Script/Engin
CreateParticleClass(TEXT("Fireball"), TEXT("/Script/Engine.Blueprint'/Game
CreateParticleClass(TEXT("N_Player_LevelUp"), TEXT("/Script/Engine.Bluepri
```

```
if (_effectTable.Contains(name) == false)
    return;

auto findEffect = _effectTable[name].FindByPredicate(
    [](ATFT_Effect* effect)-> bool
    {
        if (effect->IsPlaying())
            return false;
        return true;
    });

if (findEffect)
    (*findEffect)->Play(effectType, location, rotator);
```

5.데이터 및 통계 관리(Data and Stats Management)

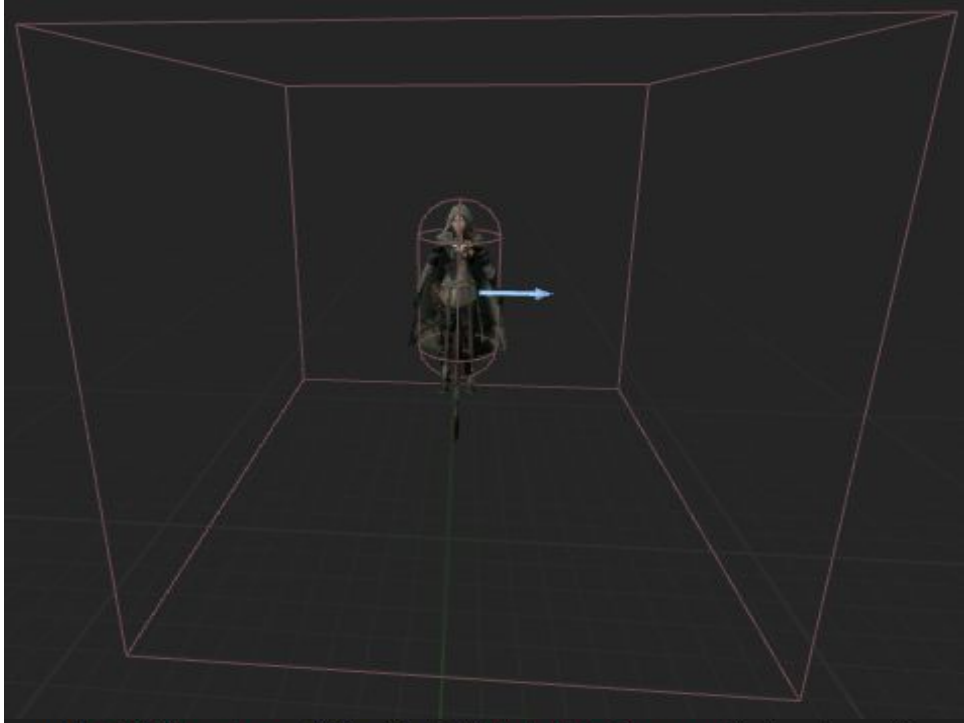
캐릭터의 스텟 시스템은 SetLevelAndInit으로 레벨에 맞는 스텟 데이터를 가져온 뒤 스탯을 초기화하고 레벨업 한 스텟과 아이템 습득 시 추가된 공격력을 반영하기 위해 AddItem_Attack으로 기본 공격력에 아이템 공격력을 더해서 공격력이 초기화하는 현상을 막아줍니다.

```
auto myGameInstance = Cast<UTFT_GameInstance>(GetWorld()->GetGameInstance());
if (myGameInstance)
{
    FTFT_StatData* data = myGameInstance->GetStatDataByLevel(level);
    _curLevel = level;
    _maxHp = data->maxHP;
    _curHp = _maxHp;
    _maxMp = data->maxMP;
    _curMp = _maxMp;
    _maxExp = data->maxExp;
    _attackDamage = data->attack;

    if (AddsItem_Attack != 0) _attackDamage += AddsItem_Attack;
```

6.상점 관리(Store Management)

상점 시스템은 npc에 TriggerBox를 이용하여 충돌 영역에 플레이어가 들어갔을 시 overlap함수를 이용하여 상점 인터페이스를 가져온 뒤 StoreOpenClose함수를 호출하여 상점 UI를 출력합니다



```
void ATFT_Npc_store::OnOverlapEnd(UPrimitiveComponent* OverlappedComp,
{
    ATFT_Player* player = Cast<ATFT_Player>(OtherActor);


    if (player != nullptr)
    {
        _interface = nullptr;
        _storeCom->GetStoreUI()->StoreOpenClose();
    }
}
```

감사합니다.