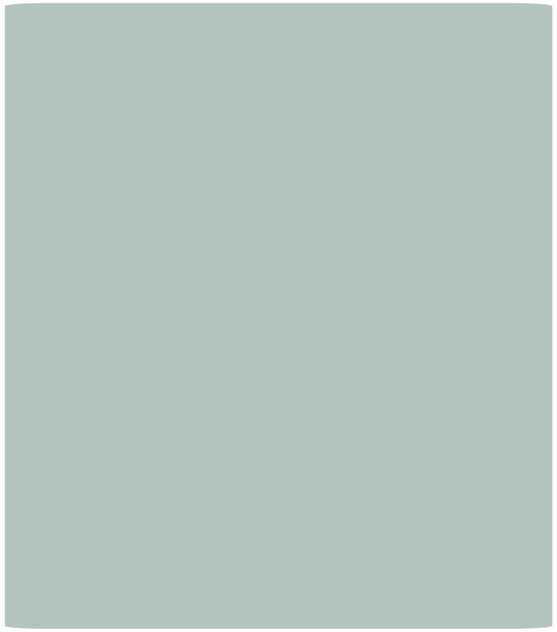
상상을 현실로 만드는 개발자

Portfolio

홍혜승

홍혜승



2000.10.02

010-6706-9200

hhs9200@gmail.com

https://github.com/hhs920

학력사항

2020.03 ~ 2024.02	서일대학교 정보통신공학과 졸업
2016.03 ~ 2019.02	원묵고등학교 졸업

교육사항

2024.12 ~ 2025.06 (6개월 과정)	청년취업사관학교 도봉캠퍼스 2기 언리얼 엔진과 생성형 AI를 활용한 콘텐츠 개발자 양성 과정

경험

2025.06.19	로스트아크 모바일 비전 프리뷰 출시 전 콘텐츠 체험
2025.05.22	플레이 엑스포 (Play X4) 각종 게임 부스 참여
2023.11.17	지스타 2023 각종 게임 부스 참여

자격증

2020.09	네트워크 관리사

Unreal Engine

- 컷신 제작(Camera Rig & Level Sequence)
- Geometry Collection을 이용한 파편화 효과 구현
- Steam 연동 세션 시스템 구축, UI 애니메이션을 통한 시각 효과 구현
- Tag 시스템을 활용한 상태 및 상호작용 관리

C++

- FSM 기반 Enemy AI 행동 트리 설계 및 전투 로직 구현
- Geometry Collection의 물리 속성 (충돌, 중력, 속도) 세밀 조정
- RPC / Multicast 기반 멀티플레이어 네트워크 동기화 처리

GitHub / SourceTree

- SourceTree를 통한 Git 버전 관리 및 팀 프로젝트 협업 브랜치 분기 및 병합, 커밋 로그 관리, 팀원 간 충돌 방지 등 협업 중심의 버전 관리 수행

- AI 도구를 활용한 코드 최적화, 로직 설계, 디버깅 아이디어 도출
- 문제 해결 접근 방식을 확장, 반복적인 코드 작성 및 오류 수정 과정 간소화.

Trello Jira

- 업무 분담 및 일정 관리를 위한 애자일 기반 프로젝트 관리 도구 활용

UnderCooked

프로젝트 개요

장르 : 시뮬레이션 요리 게임 오버쿡드(Overcooked) 모작
목표 : 플레이어 조작, 상호작용 구현, 네트워크 동기화,
세션 제작, UI 제작

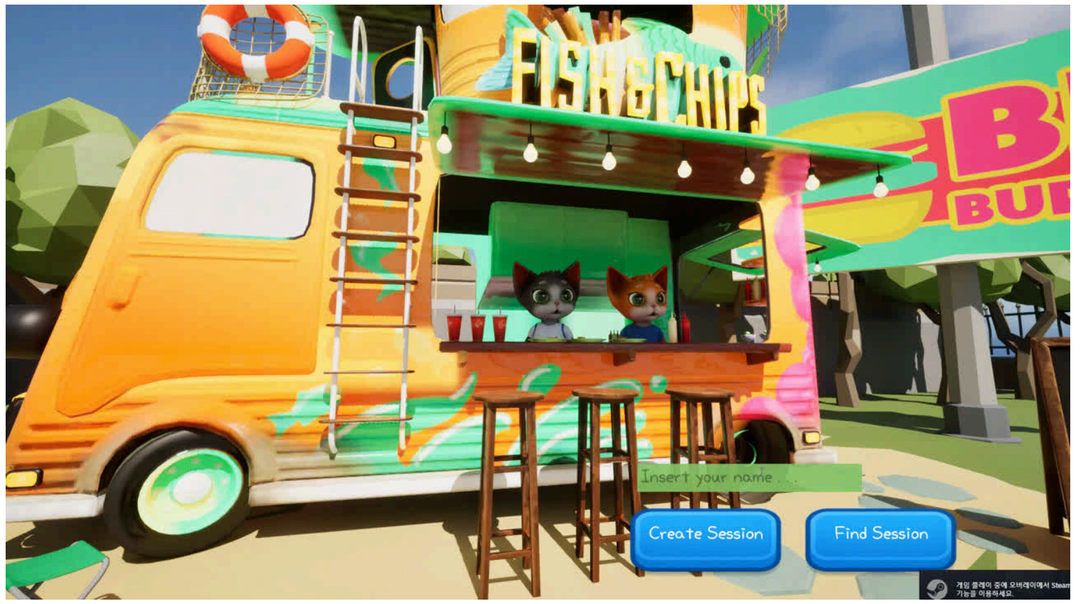
개발 기간 2025.04.07 ~ 2025.04.30 (24일)

제작 인원 3명

사용 언어 Unreal Engine5 C++

주요 구현 내용

- 플레이어 이동 및 대시 기능 구현
- 상호작용 시 8방향 캐릭터 고정 시스템 적용
- 재료 줍기, 던지기, 조리, 설거지 등
  다중 상호작용 기능 개발
- UI 연동: 게임 엔딩, 세션 UI 버튼
- 세션 제작 및 네트워크 동기화 구현
- UE5 C++ 기반의 네트워크 멀티플레이 구조 설계



개발 일정

TASKS	WEEK 1	WEEK 2	WEEK 3	WEEK 4
기획	기획, 에셋 적용 플레이어 이동			
프로토		상호작용 제작 (썰기, 조리, Grab, Drop...)		
알파			네트워크 동기화 세션 제작	
베타				맵 / VFX / SFX UI 제작

UnderCooked

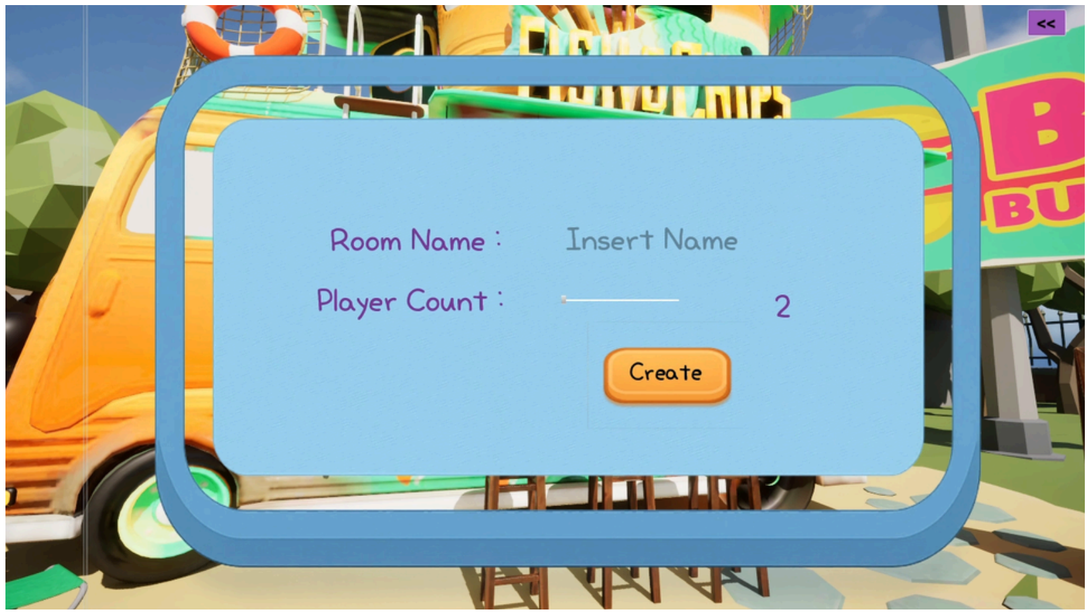
- 기본 WASD 이동 외에 빠르게 이동 가능한 Dash 기능 구현
- 이동/대시 애니메이션 전환 및 사운드 연동 처리
- 던지기 / 소화기 사용 시 캐릭터가 8방향을 바라보며 조준
- 잡기 / 놓기 / 던지기 / 조리 / 설거지 등 다양한 상호작용 구현
- 상호작용 시 각 기능별로 범위 체크 → 상태 변경 → 애니메이션/타이머 동작 순으로 처리 재료는 일정 거리 안에서만 줍기 가능,
  도마 또는 냄비도 일정 거리 안에서 조리 가능

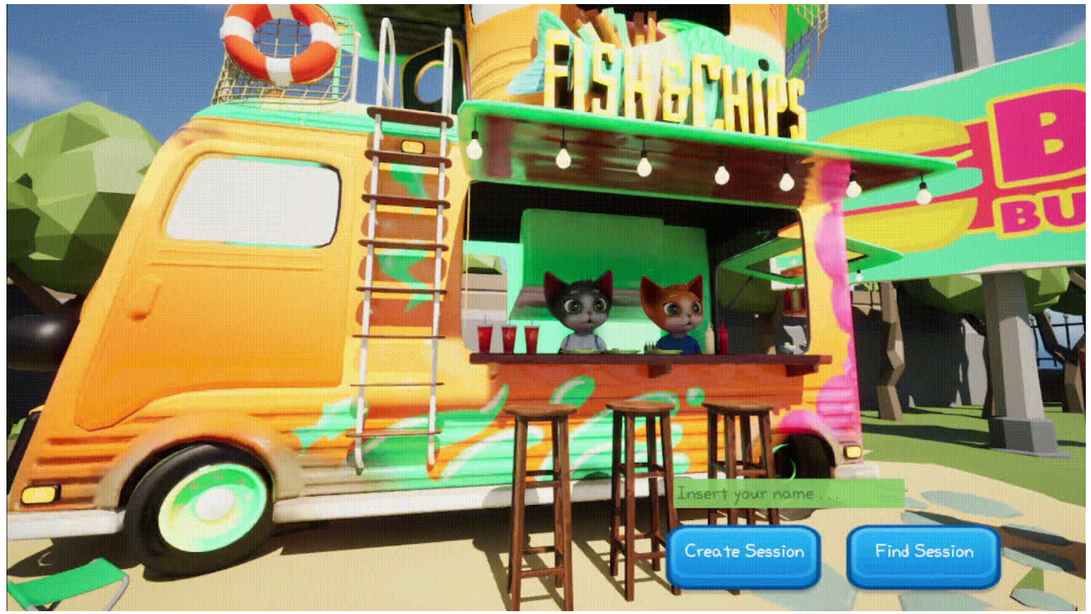
- C++ 기반의 RPC / Multicast 구조를 통해
  플레이어 간의 상태를 정확히 동기화
  ex) A 플레이어가 재료를 던지면
  모든 클라이언트에서 던지는 애니메이션 및 위치 일치

- 로컬 입력 → 서버 판단 → 모든 클라이언트 반영 구조로 처리하여
  게임플레이 사이의 지연 최소화

UnderCooked

- 세션 호스트 / 조인 기능을 포함한 멀티플레이 환경 구현
- 세션 목록 생성, UI 버튼 누른 후 서버 탐색 및 조인 가능
- 블루프린트로 구성된 플레이어 캐릭터를 직접 레벨에 배치
- 캐릭터에 애니메이션을 반복 재생하도록 구성
- 정적인 UI 배경이 아닌 실제 게임 씬과 연결된 듯한
  몰입감 있는 로비 화면을 구성





UnderCooked

- UI 애니메이션을 활용해 게임 클리어 시 별(★)이 등장하는 연출을 구현.

- 별 아이콘이 확대되었다가 다시 줄어드는 스케일 애니메이션을 통해
  단순한 아이콘 표시보다 강조되고 생동감 있는 시각 효과 제공.

- UMG 내에서 KeyFrame 기반으로 제작됨
  이펙트 타이밍과 UI 요소의 크기/투명도 조절을 통해 클리어 보상의 느낌을 극대화.

```cpp
void UEndGameUI::ShowResultPanel()
{
    if (TimeOverPanel)
    {
        TimeOverPanel->SetVisibility(ESlateVisibility::Hidden);
    }

    if (ResultPanel)
    {
        ResultPanel->SetVisibility(ESlateVisibility::Visible);
    }

    if (ScoreText)
    {
        ScoreText->SetText(FText::FromString(FString::Printf( Fmt TEXT("%d"), CachedScore)));
    }

    if (StarAppearAnim)
    {
        PlayAnimation(StarAppearAnim);
    }

    for (int32 i = 0; i < StarImages.Num(); ++i)
    {
        if (StarImages[i])
        {
            StarImages[i]->SetVisibility(i < CachedStarCount ? ESlateVisibility::Visible : ESlateVisibility::Hidden);
        }
    }
```

RON

프로젝트 개요

장르 : 전술 FPS 슈팅 게임 – 레디 오어 낫 모작
목표 : FSM을 이용한 AI 로직 구현, 컷신 제작

개발 기간 2025.01.23 ~ 2025.02.24 (33일)
제작 인원 2명
사용 언어 Unreal Engine5 C++

주요 구현 내용

- 에너미 AI 행동 트리(FSM 기반) 설계 및 구현
- 공격, Patrol, 추적 등의 AI 상태 정의 및 전이 구현
- 캐릭터 애니메이션 상태머신 구성
- 코드 최적화 및 유지보수 용이성을 고려한 구조 변경
- 레벨 시퀀스 시네마틱을 이용한 컷신 제작

개발 일정

TASKS	WEEK 1	WEEK 2	WEEK 3	WEEK 4
기획	기획 에셋 적용			
프로토		FSM제작 PatrolPath 제작		
알파			플레이어 공격 → 도주 or 항복	
베타				맵 / VFX / SFX 컷신 제작

RON

Enemy는 **FSM 구조**를 기반으로 행동함.
기본적으로 Patrol → Detect → Attack의 전투 흐름을 따름.

Enemy는 지정된 PatrolPath를 따라 순찰하며
시야 범위 내에서 플레이어를 감지하면 Attack 상태로 전환됨.

전투 중 HP가 일정 수치 이하로 감소하면,
Enemy가 사망하는 것이 아닌
조건에 따라 항복하거나 도주하는 상태로 전이.


```cpp
void UPistolEnemyFSM::EscapeState()
{
    if (bIsEscaping) return;
    bIsEscaping = true;

    // 도망 위치 설정 (랜덤 or 특정 위치)
    escapeLocation = FVector( InX: 2126.275427f,  InY: -3586.238729f,  InZ: 71.715371f); // 예제 좌표

    // AI 이동 시작 (한 번만 실행)
    if (ai)
    {
        EPathFollowingRequestResult::Type result = ai->MoveToLocation(escapeLocation);

        if (result == EPathFollowingRequestResult::Failed)
        {
            GetRandomPositionInNavMesh(me->GetActorLocation(),  radius: 1000.0f,  dest: [&] escapeLocation);
            ai->MoveToLocation(escapeLocation);
        }
    }
```

```cpp
    // 도망 애니메이션 실행
    if (anim && anim->EnemyRun)
    {
        me->PlayAnimMontage(anim->EnemyRun,  InPlayRate: 1.0f,  StartSectionName: TEXT("Escape"));
    }
    // 이동 속도 증가 (예: 원래 속도의 3배로 설정)
    if (me)
    {
        me->GetCharacterMovement()->MaxWalkSpeed = moveSpeed * 3.0f;  // 이동 속도를 증가시킴
    }
    anim->AnimState = mState;
}
```

- 도주(Escape)
  - Enemy는 도망 위치를 탐색하고,
    AIController::MoveToLocation()을 통해 이동.
  - 실패 시에는 NavMesh 내에서 유효한 위치를
    재탐색하여 경로 재설정.
  - 도주 중에는 기존 이동 속도의 3배로 가속되며, Escape 애니메이션을 재생.
    → 중복 실행 방지를 위한 bIsEscaping 플래그를 통해 FSM 안정성 확보

- 항복(Surrender)
  - Enemy의 체력이 surrenderHP 이하로 떨어졌을 때, 공격을 중단하고 AI 이동을 정지.
  - 이후 항복 애니메이션을 재생, 들고 있던 무기를 바닥에 떨어뜨림.
  - DropHoldingEquipment() 함수를 통해 실제 Combat Component와
    연결된 장비를 분리하도록 구현.

```cpp
if (hp <= surrenderHP && mState != EEnemyState::Surrender)  // 항복 상태 체크
{
    mState = EEnemyState::Surrender;
    ai->StopMovement();

    if (anim && anim->EnemySurrender)
    {
        me->PlayAnimMontage(anim->EnemySurrender,  InPlayRate: 1.0f,  StartSectionName: TEXT("Surrender"));
        me->CombatComp->DropHoldingEquipment(); // 들고있던 장비(총)을 드랍한다.
        PRINT_LOG(TEXT( Fmt Fmt "적이 항복했습니다!"));
    }
    return;
}
```

RON

프로젝트 내 시네마틱 연출을 위해 Level Sequence 시스템을 사용하여 약 30초 분량의 컷신을 제작.
시네마틱의 몰입감 있는 시점 이동을 구현하기 위해 Camera Rig Rail 컴포넌트 활용.

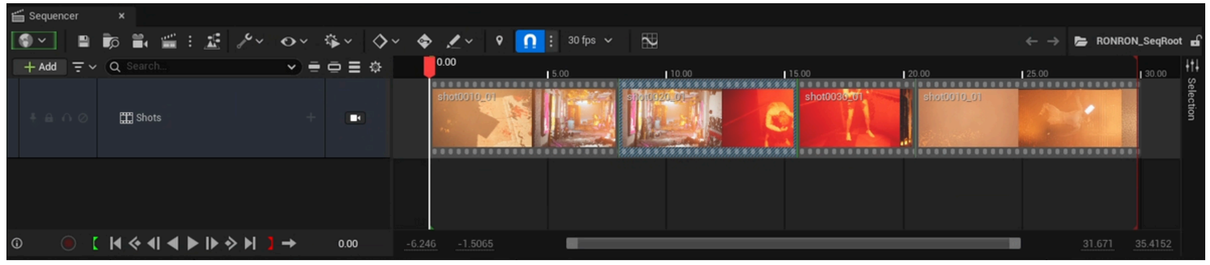
- 카메라 시점의 자연스러운 추적 및 이동을 위해
  Camera Rig Rail과 Rail Camera Actor를 사용

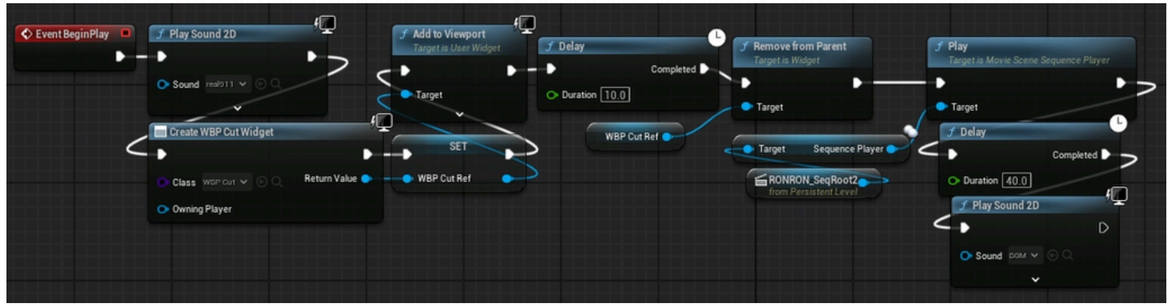
- 시퀀서 내에서 Camera Rig Rail의 Spline Point를 편집하여
  카메라가 이동하는 경로를 자유롭게 커스터마이징

- Rail Camera Actor를 레일에 탑재하고, 시퀀서에서 키프레임을 설정하여
  → 초반 줌인 → 중간 회전 이동 → 후반 캐릭터 클로즈업처럼
  다양한 시점 전환을 자연스럽게 연결

- Rail Camera의 FOV 조정 및 포커스 제어를 통해 퀄리티 향상.

- 레벨 블루프린트에서 BGM 재생과 동시에
  Level Sequence를 실행하여 컷신이 자연스럽게 시작되도록 구성.





MATRIX

프로젝트 개요

장르 : 액션 퍼즐 FPS 슈팅 게임 SUPERHOT 모작
목표 : FSM을 이용한 AI 로직 구현
지오메트리 컬렉션을 사용한 파편화 기능 구현

개발 기간 2025.03.19 ~ 2025.04.01 (14일)
제작 인원 2명
사용 언어 Unreal Engine5 C++

주요 구현 내용

- FSM 기반의 에너미 상태 관리 및 AI 로직 구현
- 시간 정지/재개 컨셉과 연동된 에너미 동작 처리
- 에너미 사망 시 Geometry Collection을 이용한 파편화 연출

개발 일정

TASKS	WEEK 1	WEEK 1-1	WEEK 2-1	WEEK 2-2
기획	기획 에셋 적용			
프로토		원/근거리 공격, WeaponComponent 제작		
알파			projectile 구현 (+ Tail) 충돌 설정	
베타				에너미 사망시 파편화, 맵 / VFX/ SFX

MATRIX

시간 정지/재개 기반의 FPS 게임으로, 핵심 시스템 중 하나인 에너미 FSM과
파편화 사망 연출 구현.

- 에너미 Skeletal Mesh의 사망 연출을 위해 Geometry Collection Asset을 제작
  - 메쉬를 Fracture Tool을 통해 내부적으로 파편화 처리
  - Block 형태로 설정하여 Superhot의 유리조각처럼 부서지는 느낌 재현

- 기존의 Skeletal Mesh에서 사망 시
  Geometry Collection Actor로 교체되도록 처리
  - 캐릭터가 사망하면 기존 매쉬는 숨기고,
    해당 위치에 파편화된 Geometry Collection을 스폰하여 교체

- Geometry Collection은 Chaos Physics 시스템과 연동되어
  실제 충돌, 중력, 속도에 따라 파편들이 물리적으로 튕김.

- 플레이어의 총알이 명중한 위치와 방향을 계산하여
  Impulse 값을 해당 방향으로 전달, 현실감 있는 파편 튕김 효과 구현

```cpp
void ASHEnemy::OnDeath()
{
    if (bIsDead) return;
    bIsDead = true;

    // SkeletalMesh 비활성화
    GetMesh()->SetVisibility(false);
    GetMesh()->SetCollisionEnabled( NewType: ECollisionEnabled::NoCollision);

    if (GeometryCollectionComp)
    {
        // 파편 위치/회전 설정
        GeometryCollectionComp->SetWorldLocation(GetActorLocation());
        GeometryCollectionComp->SetWorldRotation(GetActorRotation());

        // 파편화 활성화
        GeometryCollectionComp->SetVisibility(true);
        GeometryCollectionComp->SetSimulatePhysics( bEnabled: true);
        GeometryCollectionComp->SetEnableGravity(true);
        GeometryCollectionComp->SetCollisionEnabled( NewType: ECollisionEnabled::QueryAndPhysics);
        GeometryCollectionComp->SetCollisionObjectType( Channel: ECC_PhysicsBody);
        GeometryCollectionComp->SetAllPhysicsLinearVelocity( NewVel: FVector::ZeroVector);
        GeometryCollectionComp->SetAllPhysicsAngularVelocityInDegrees( NewAngVel: FVector::ZeroVector);

        // 외부 충격 및 폭발 효과 적용
        GeometryCollectionComp->ApplyExternalStrain( ItemIndex: 1000.0f, GetActorLocation(), Radius: 150.0f, PropagationDepth: 1);
        FVector ExplosionCenter = GetActorLocation() + FVector( inX: 0.0f, inY: 0.0f, inZ: 120.0f);
        GeometryCollectionComp->AddRadialImpulse(ExplosionCenter, Radius: 1200.0f, Strength: 1000.0f, ERadialImpulseFalloff::RIF_Linear, bVelChange: true);
        GeometryCollectionComp->AddForce(FVector( inX: 0.0f, inY: 0.0f, inZ: -8000.0f), NAME_None, bAccelChange: true);

        // 일정 시간 후 파편 제거
        GetWorld()->GetTimerManager().SetTimer( &: DestructionTimerHandle, InObj: this, &ASHEnemy::DestroyFragments, InRate: 3.0f, InbLoop: false);
    }
}
```

스턴 맛 좀 봐라

프로젝트 개요

장르 : 수렵 액션 게임 몬스터헌터 모작
목표 : 몬스터 헌터의 전투 시스템, 플레이어 조작 구현

개발 기간 2024.12.23 ~ 2025.01.22 (31일)
제작 인원 3명
사용 언어 Unreal Engine5 Blueprint

주요 구현 내용

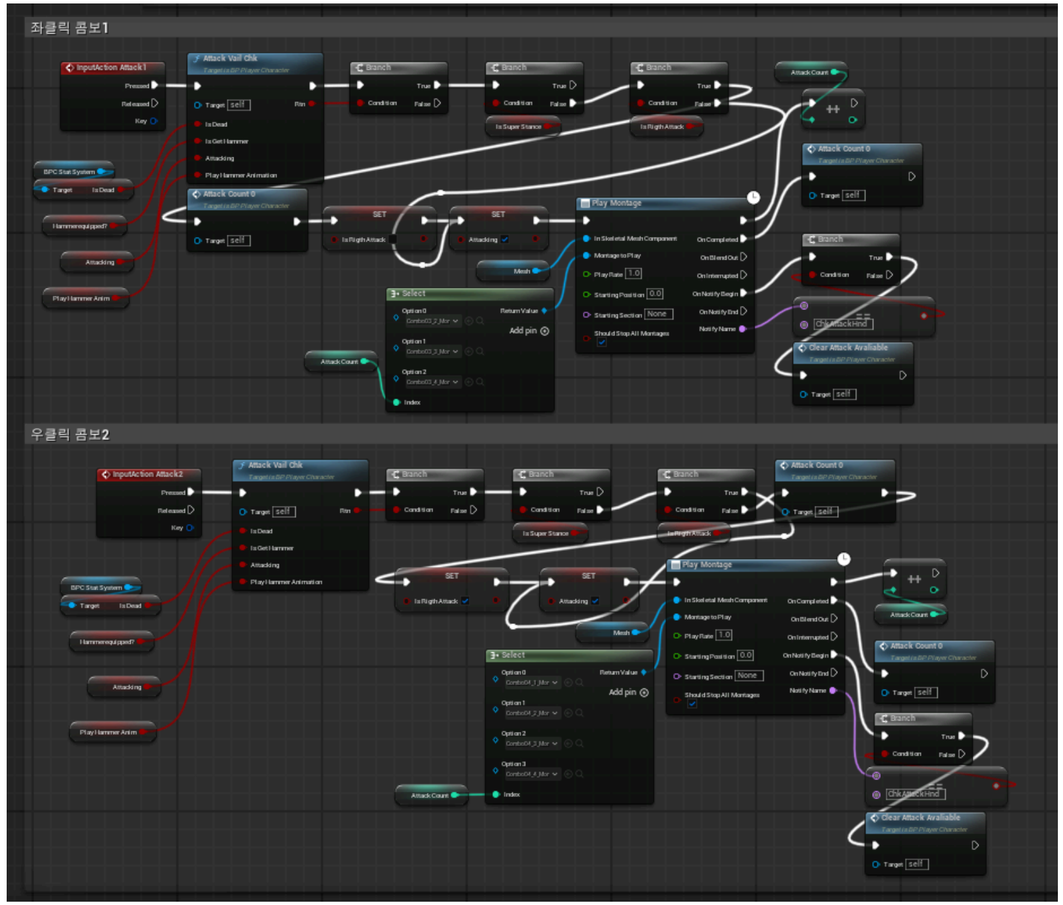
- 플레이어 이동, 회피, 콤보 공격 기능 구현
- 몬스터와의 전투 시스템 설계 및 연동
- 체력 및 스태미나 상태를 반영하는 UI 시스템 구현
- 불필요한 변수 제거, 노드 단순화
  → 함수 기반 구조로 리팩토링

개발 일정

TASKS	WEEK 1	WEEK 2	WEEK 3	WEEK 4
기획	기획 에셋 적용			
프로토		플레이어 공격, 피격, 사망		
알파			체력, 스테미나 UI 공격 LineTrace	
베타				에너미 충돌 처리 VFX/ SFX

스턴 맛 좀 봐라

- 플레이어의 기본 이동, 회피, 콤보 공격 기능을 블루프린트로 구성하여 직관적인 전투 시스템 구현
- 회피 기능은 이동 방향 입력을 기반으로 일정 거리만큼 빠르게 이동하며, 일정 시간 무적 상태 부여
- 콤보 공격은 애니메이션 노티파이와 타이밍 조건을 활용해 공격이 자연스럽게 연결되도록 구성
- 몬스터와의 전투 시스템은 충돌 이벤트, 데미지 계산, 피격 리액션 등
  여러 요소가 상호작용하는 구조로 설계
- 공격을 받을 시 몬스터가 반응하고 HP가 감소하는 로직을 블루프린트로 설계.



감사합니다.