2025 PORTFOLIO

Game Client Programmer

전재연

모두가 좋아할 게임을 만들자!

nidhaku31@gmail.com
010-7374-3359

개인 정보

이름 전재연

이메일 nidhaku31@gmail.com

번호 010-7374-3359

주소 인천광역시 부평구

학력 및 교육이력

2020.03 ~20205.02 청강문화산업대학교 게임과 20학번 졸업

2025.05 ~20205.07 [스마일게이트] 샌드박스 게임제작툴 사용성 검증 및 강화

하고 싶은 말

두 번의 학내 프로젝트를 통해
전시를 위한 작품을 준비하며
개발 경험을 축적했습니다.
현재는 직접 게임을 출시하기 위해
팀을 조직하여
인디게임 개발을 진행 중입니다.

주어진 기획을
충실히 구현하는 것에 그치지 않고,
기획자와 적극적으로 소통하며
새로운 아이디어와 개선 방안을 제안하여
게임의 완성도 향상에 기여해왔습니다.

활동

2024 BIC 세미오픈월드 어드벤처
Azure Fields
BIC 루키부문 참여

참여 프로젝트

2023.07 ~2023.11 청강대학교 학기작
탑뷰 슈팅게임
Office Fury
클라이언트&
프로그래머 파트장

2024.02 ~2024.12 청강대학교 졸업작
세미오픈월드 어드벤처
Azure Fields
클라이언트 프로그래머

2025.01 ~ 인디게임팀 리벨로
2.5D 턴제배틀
리벨로 레코드 개발중
팀장&클라이언트&기획

저의 역량과 강점을 자신 있게 소개합니다.

프로젝트를 통해 얻은 경험과 성장 과정을
저의 강점으로 삼고 있습니다.
앞으로도 끊임없이 배우고,
더 나은 개발자가 되기 위해
노력하겠습니다.

UNITY C# 스크립트

C#을 주력으로 사용하여,
다양한 게임 프로젝트에서 원하는 기능을
스스로 구현할 수 있는 실력을 키웠습니다.

UNITY C# 에디터

기획자들이 유니티 내에서 쉽게 조정하고
이해할 수 있도록
에디터 기능을 커스텀 제작했습니다.

기획자와의 커뮤니케이션

기획자들과 적극적으로 소통하며
게임의 퀄리티를 향상시키기 위한
다양한 기능을 제안하고 제작해왔습니다.

프로그래머들과의 협업

다른 프로그래머들과 역할을 나누고,
코드 통합과 문제 해결을 위한
협업을 경험했습니다.

PROJECT.03

LIBELLO RECORD

2.5D 턴제 배틀 시뮬레이션

기간 | 개발 중(2025.01~)

인원 | 8인 (기획 2, 프로그래머 1, 아트 4, QA 1)

도구 | UNITY, C#

주요 역할
1. 턴제배틀 시스템 구현
2. 전투 시스템 구현
3. 버프 디버프 구현

'리벨로 피스'라 불리는 기적의 노트는,

적은 내용을 실체화시키는 신비한 힘을 지니고 있다.

리벨로 피스를 둘러싼 끝없는 욕망과 갈등 속에서,

오직 진실을 좇는 두 명의 탐정이 세상을 향해 나아간다.

1. 턴제배틀 시스템 구현

문제 | 게임의 가장 기본적인 메인시스템, 턴제배틀을 구현해야합니다.

방법 | 유희왕의 턴 페이즈를 참고하여 턴제 배틀의 기본 시스템을 마련했습니다.

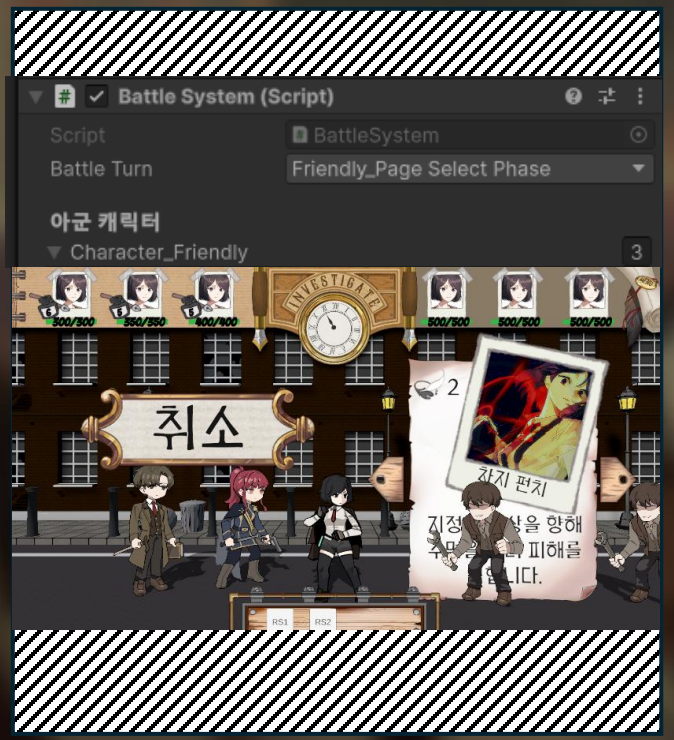
효과 | 페이즈를 구분함으로써 플레이어가 페이즈마다 할 수 있는 행동과 버프 및 공격의 처리를 확실하게 부분지어서 통제할 수 있습니다.

배틀 턴의 구분

```
#region 현재 배틀맵의 종류
참조 3개
public enum BattleMapType
{
    Basic, //잡몹전, 모든 적들이 사망시 클리어
    Boss, //보스전, 적캐릭터1(보스) 사망시 클리어
}

private BattleMapType battleMapType;
#endregion

#region 턴구성
참조 58개
public enum BattleTurn
{
    Friendly_StartPhase, //시작페이즈
    Friendly_CharacterSelectPhase,   //아군 캐릭터 선택 페이즈
    Friendly_PageSelectPhase,   //아군 페이지선택
    Friendly_Page_FriendlyTargetingPhase,    //아군 페이지의 대상 선택
    Friendly_Page_EnemyTargetingPhase,    //아군 페이지의 대상 선택
    Friendly_PageActivePhase,   //아군 페이지 진행
    Friendly_Reasoning1TargetingSelectPhase, //추리스킬1 대상선택
    Friendly_Reasoning2TargetingSelectPhase, //추리스킬2 대상선택
    Friendly_ReasoningActivePhase, //추리스킬 대상선택
    Friendly_EndPhase,   //엔드페이즈
    Enemy_StartPhase,
    Enemy_CharacterSelectPhase,
    Enemy_PageSelectPhase,
    Enemy_PageTargetingPhase,
    Enemy_PageActivePhase,
    Enemy_EndPhase,
    Battle_End, //전투종료!!!

}
public BattleTurn battleTurn;
#endregion
```

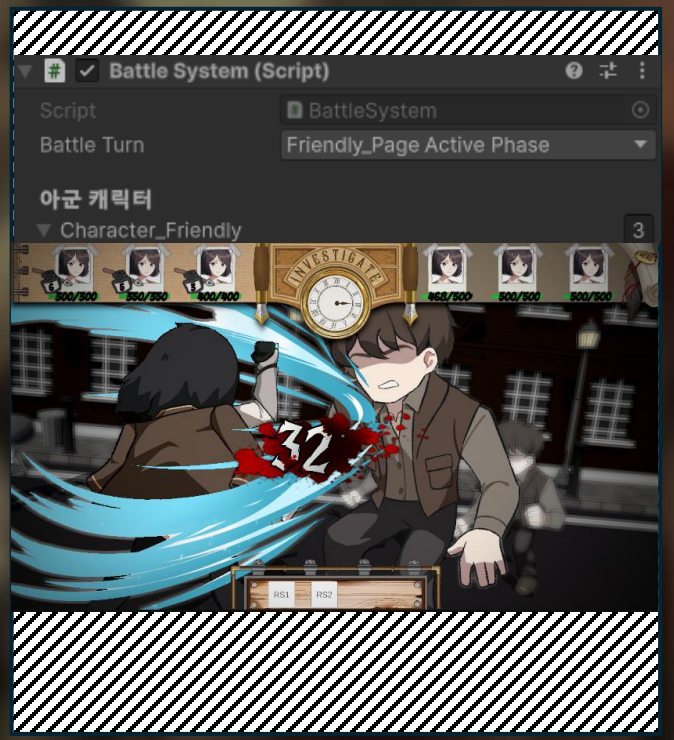
턴제 배틀의 시스템을 구현하기 위해서
유희왕 카드 게임의 시스템인 턴의 구분을 참고하여
각 페이즈마다 할 수 있는 플레이어의 행동과 처리를 구현합니다.

페이즈에 따른 행동가능



페이즈에 따라서 플레이어가 할 수 있는 행동을
구분지음으로써 예상치못한 변수를 차단하고 있습니다.

특정 페이즈에선 플레이어 조작 불가



플레이어가 조작을 하면 안되는 페이즈에서는
조작 불가로 전환하여 혹여나 생길
버그여부를 차단하고 있습니다.

2. 전투 시스템 구현

문제 | 턴제 배틀에서 중요한 배틀시스템의 처리방식을 구현해야 했습니다.

방법 | 데미지 계산 및 애니메이션의 처리는 동일한 함수내에서 처리합니다. 다만, 스킬의 연출 및 애니메이션은 스킬마다 차이가 있기에 커스텀이 가능하도록 스크립트테이블로 수정이 어느정도 커스텀이 가능하도록 처리합니다.

효과 |.스킬마다 약간씩 다른 연출을 보여주며, 애니메이션 시간도 스킬마다 다르게 통제할 수 있기에 일부 스킬들에게 연출에 특별함을 더할 수 있는 기회가 생겼습니다.

각 스킬에 따른 연출 및 데미지 처리 구분

```
int FinalDamage()
{
    int damage = Random.Range(activePage.page_MinDamage, activePage.page_MaxDamage);
    int finalDamage = damage + ((damage * character_Amplification.character_BuffValue) / 100) -
    return finalDamage;
}

#endregion
#region 페이지타입을 확인하고 페이지 적용
참조 4개
public void PageActive()
{

    battleUI.CharacterUiDataUpdate();
    #region 페이지 타입 확인
    switch (activePage.pageActiveType)
    {
        Single_Attack 적군 한명에게 대상 공격

        Aoe_Attack 적군에게 광역공격

        Single_Heal 아군 한명에게 대상 힐

        My_Heal 자기자신에게 힐

        Aoe_Heal 아군 전체에게 힐

        Single_Shield 아군 한명에게 보호막

        My_Shield 자기자신에게 보호막

        Aoe_Shield 아군 전체에게 보호막
    }
    #endregion
    battleUI.CharacterUiDataUpdate();
}
#endregion
버프/디버프 부여
버프/디버프 적중
페이지에 따른 애니메이션 출력
```

스킬의 종류마다 다른 처리방식을 적용하여
연출 및 애니메이션, 데미지 처리에 차별점을 둡니다.

스크립트 테이블을 통하여 기획자들의 쉬운 수정

스킬은 스크립트 테이블을 통하여 구성할 수 있기에
기획자들도 간단하게 수치 및 구성을 수정하여
테스트를 진행할 수 있습니다.

확장성이 좋도록 설계

각 스킬마다 다른 연출, 다른 애니메이션을 적용함으로써
스킬마다 고유한 연출을 보여줄 수 있습니다.

배틀 시스템 구현 플레이 테스트 영상

https://youtu.be/tgHwMYVoAD4?si=SXZbHgrkFVslmUe4

3. 버프 및 디버프의 구현

문제 | 버프 및 디버프를 구현해야합니다. 버프는 스타트 페이즈에 처리되는 디버프, 엔드페이즈 처리 디버프, 스택이 추가될 때마다 처리되는 디버프 총 3종이 존재합니다.

방법 | 각 페이즈마다 처리할 수 있게 페이즈에 도달하면 1회 실행되는 함수로 설정한 다음, 캐릭터마다 버프 스택을 확인할 수 있도록 클래스를 추가합니다.

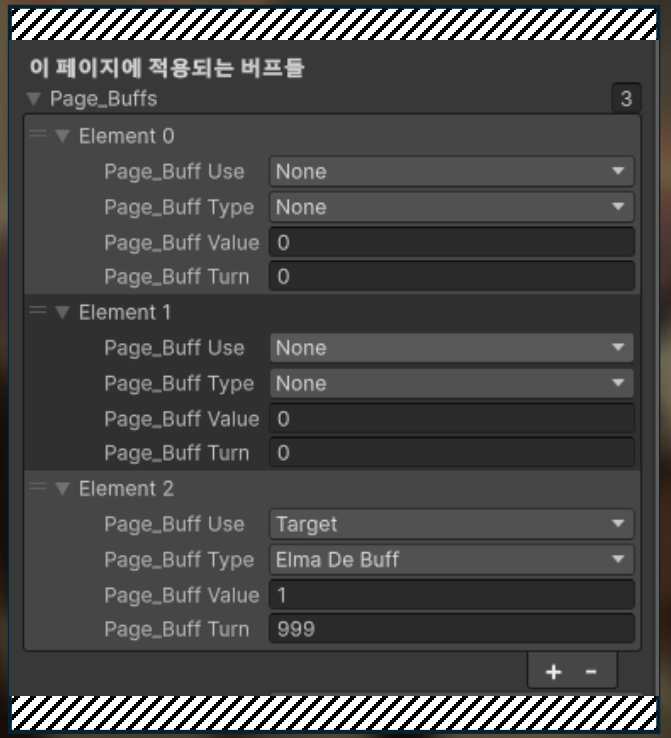
효과 | 버프 및 디버프의 효과 처리가 각 페이즈마다 적용되며, 버프 및 디버프를 활용한 게임플레이의 재미를 더했습니다.

버프 적용 및 처리

```
#region 버프/디버프 적용
//스타트 페이즈
참조 0개
public void BuffActive_StrtPhase()
{

}
참조 2개
public IEnumerator BuffActive_EndPhase()
{


    yield return new WaitForSeconds(1.0f);
    DamageUiPrintOff();

    yield return null;
}
참조 1개
public IEnumerator BuffActive_BattleFinish()
{



    for (int i = 0; i < activePage.page_Buffs.Length; i++)
    {
        //엘마 디버프가 3이상일을 경우 기본 30 + (버프수치-3)*15
        if (targetCharacter.character_ElmaDeBuff.character_BuffValue >= 3)
        {
            yield return new WaitForSeconds(1.0f);
            targetCharacter.DamageUiPrintOn(30 + ((character_ElmaDeBuff.character_BuffValue - 3) * 15));
            targetCharacter.character_Hp -= 30 + ((character_ElmaDeBuff.character_BuffValue - 3) * 15);
            targetCharacter.targetCharacter = this;

            yield return new WaitForSeconds(0.5f);
            targetCharacter.DamageUiPrintOff();

        }

        //어그로 버프처리
        if (activePage.page_Buffs[i].page_BuffType == PageTable.BuffType.Aggro)
        {
            targetCharacter.targetCharacter = this;
        }
    }

    yield return null;
}
#endregion
```

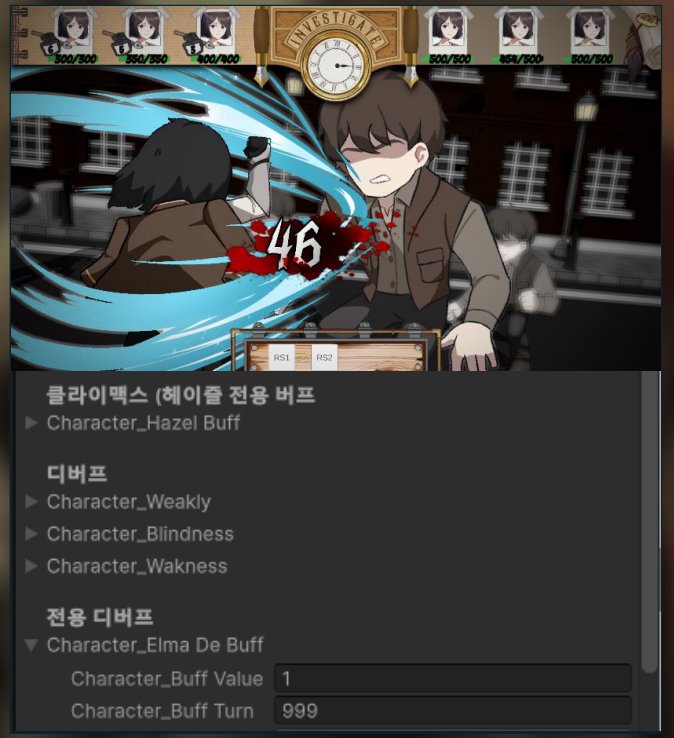
스타트 페이즈, 엔드 페이즈, 스택이 쌓일 때마다
버프를 처리합니다.

스킬 테이블에서 버프 디버프 부여 및 타입 수치 등등 조절



버프 디버프의 부여 여부는 스크립트 테이블을 통해
쉽게 수정하고 테스트할 수 있습니다.

버프 디버프 적용 확인



공격 적중 시, 버프 및 디버프 카운트가 쌓입니다.

PROJECT.02

LUKA : In the AZURE FIELD

세미오픈월드 3D 어드벤처

기간 | 10개월 (2024.02~2024.12)

인원 | 13인 (기획 3, 플밍 3, 아트 7)

도구 | UNITY, C#

주요 역할
1. 맵의 다양한 기믹들
2. 빠른 속도로 이동하다가 부딪칠 경우 기절
3. 스플라인을 활용한 맵제한 시스템

세계는 끝없는 홍수로 무너졌다.

모든 것이 물에 잠긴 그날 이후,

소년 루카는 오직 하나의 목표를 품고 깨어난다.

바다를 넘어, 사라진 쌍둥이 누나 '에린'을 찾아 나서는

모험이 시작된다.

플레이 영상링크 | https://youtu.be/zT7DDU9Dgss?si=2yfiVKTc4oSpIni5

1. 맵의 다양한 기믹들

문제 | 세미 오픈월드 게임인만큼 맵에 다양한 기믹들을 많이 추가해야합니다.

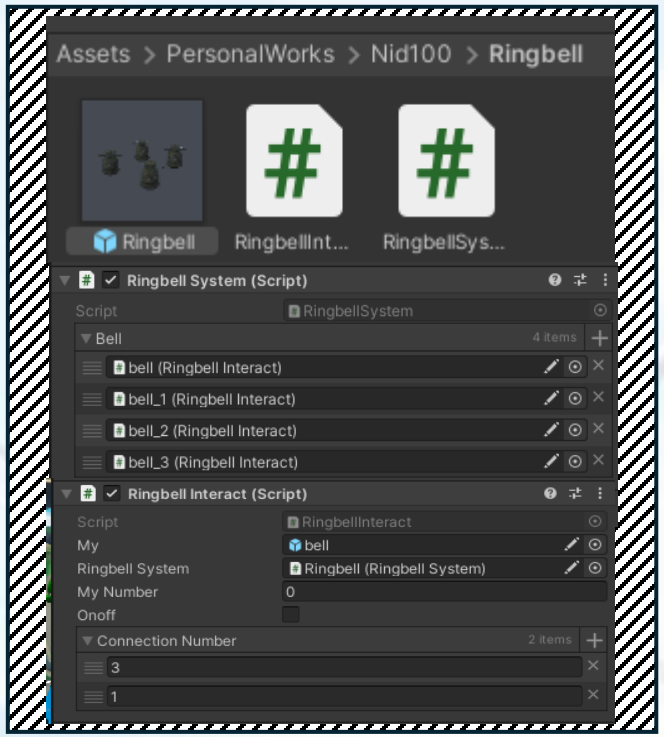
방법 | 순서대로 제단을 활성화시키는 기믹, 물 엘리베이터, 파도를 흘려보내 종을 울리는 기믹 등등 다양한 기믹을 기획자들이 배치하게 쉽게 제작하였습니다.

효과 | 퍼즐/탐험 요소를 강화하며 레벨 디자인의 자유도를 향상시켰습니다. 또한 프로그래머가 해야할 일을 기획자도 할 수 있게 작업하여 작업의 효율성을 올렸습니다.

물 엘리베이터 기믹

물엘리베이터 입니다.

기획자가 쉽게 배치할 수 있도록 인스펙터 최대한 간소화



만든 기믹들은 기획자들이 배치하기 편하도록
매뉴얼도 작성하였습니다.

파도를 흘려보내 종을 울리는 기믹

대부분의 몬스터는 플레이어를 추적하지만
예외로 플레이어를 보면 도망치는 몬스터도 있습니다.
해당 몬스터는 추격값에 -를 해서 도망치게 하였습니다.

2.빠른 속도로 이동하다가 부딪치면 기절

문제 | 캐릭터가 빠른 속도로 이동 중 암초에 부딪칠 때, 기절하는 장애물 시스템을 구현해야 했습니다.

방법 | 플레이어의 이동 속도를 계산하여, 일정 속도 이상으로 충돌했을 경우에만 기절하도록 설정했습니다. 단, 암초에 부딪치더라도 플레이어가 바위 표면을 따라 속도가 감소하지 않고 흘러가듯 이동할 경우 기절하지않도록 예외 처리를 적용했습니다.

효과 | 속도에 따른 위험성과 조작의 긴장감을 부여하여 게임의 재미를 추구하였습니다.

보드를 타고 빠른 속도를 이동 중에
정통으로 부딪치면 기절합니다.

플레이어가 빠른 속도로 이동 중에
암초와 부딪치면 기절합니다.

ReefCrash 코루틴

```
/// <summary>
/// 플레이어가 조각배 탑승 중에 암초에 충돌할 경우
/// </summary>

IEnumerator ReefCrash()
{
    DisableControls();
    animator.SetTrigger("ReefCrash");
    RuntimeManager.PlayOneShot(sound_SailboatBump);
    AbortBooster();
    driftActive = false;
    stunEffect.Play(true);
    stoneAttackEffect.Play(true);

    rBody.velocity = new Vector3(0f, rBody.velocity.y, 0f);
    rBody.AddForce(-transform.forward * reefCrashPower, ForceMode.Impulse);

    yield return new WaitForSeconds(reefCrashStifftime);


    SailboatQuit();
    //rBody.AddForce(Vector3.back * reefCrashPower, ForceMode.Impulse);
    //rBody.AddForce(Vector3.down * reefCrashPower, ForceMode.Impulse);
    yield return new WaitForSeconds(reefCrashbindTime);


    EnableControls();
}

float reefCrashStifftime = 0.5f;
float reefCrashbindTime = 3.0f;
float reefCrashPower = 15.0f;
float boatGroundingTimer = 0f;
```

플레이어의 현재 속도를 실시간으로 확인하여
암초에 부딪치고 속도가 감속되면
기절하게 만드는 코루틴입니다.

벽면을 타고 비스듬히 이동하면
속도가 감속하지않고 이동할 수 있습니다.

암초에 부딪쳐도 일정속도 이하로 떨어지지않게
벽면을 비스듬히 잘타면
기절하지않고 계속 빠른 속도로 이동할 수 있습니다.

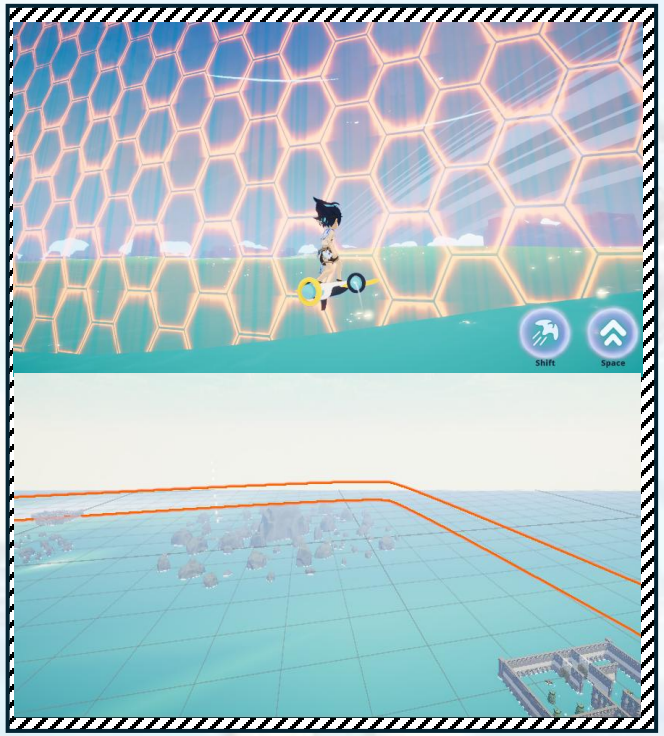
3. 스플라인을 활용한 맵제한 시스템

문제 | 캐릭터가 맵에서 벗어날 경우, 컨텐츠가 존재하지않기 때문에 어느정도 돌아다닐 수 있는 반경을 제한할 필요가 있습니다.

방법 | 스플라인을 활용해 맵을 제한하고 혹여나 맵밖으로 벗어난다하더라도 정해진 지점으로 다시 복귀할 수 있게 합니다.

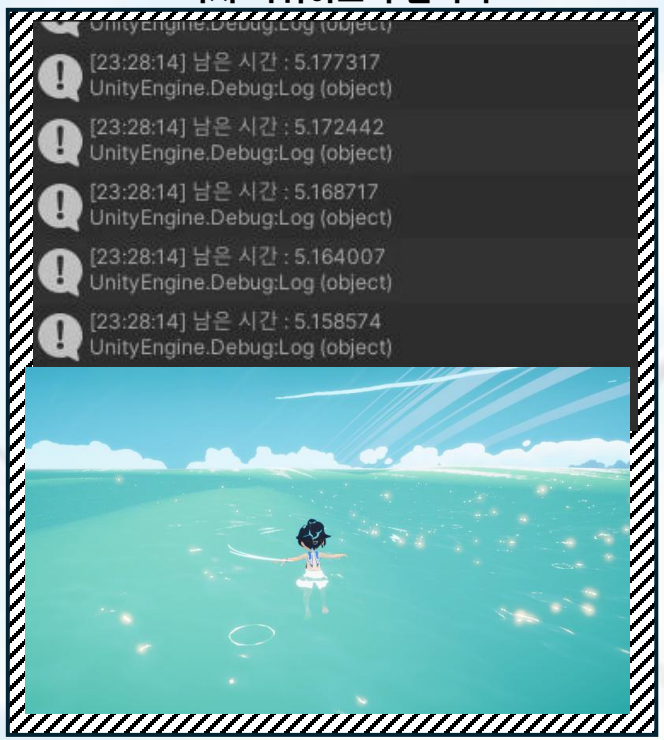
효과 | 속도에 따른 위험성과 조작의 긴장감을 부여하여 게임의 재미를 추구하였습니다.

스플라인으로 정해진 라인은
결계로 인해 나갈 수 없습니다.



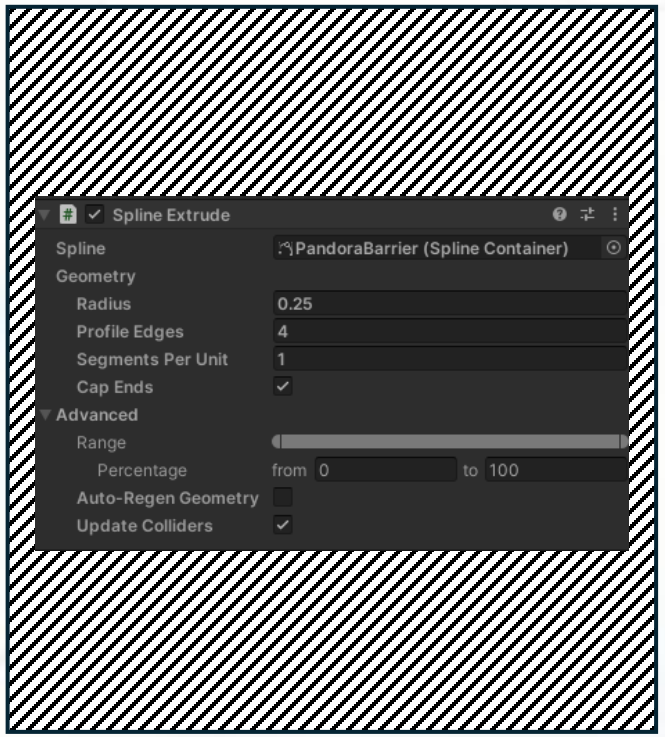
기본적으로 스플라인 바깥을 나가지 못하게하여
최대한 컨텐츠를 즐길 수 있게합니다.

혹여나 맵밖을 벗어나도
다시 복귀하도록 합니다.



플레이어가 맵밖을 벗어나면 카운트 다운이 시작되며
0되면 강제로 세이브지점으로 이동시킵니다.

스플라인 맵제한 인스펙터



스플라인을 활용한 첫번째 기능입니다.

PROJECT.01

Office Fury

탑뷰 슈팅게임

기간 | 5개월 (2023.07~2023.11)

인원 | 18인 (기획 4, 플밍 2, 아트 12)

도구 | UNITY, C#

주요 역할

1. 플레이어의 이동 및 애니메이션의 자연스러운 연출
2. 화염방사기 시스템 구현
3. 몬스터 종류별로 개별 FSM구성 및 구현

플레이 영상링크 | https://youtu.be/4fctmbRANDU

어느 날, 회사원들이 서류괴물로 변해버렸다!
플레이어는 화염방사기를 들고,
지긋지긋한 서류더미를 화끈하게 처치하며
혼란스러운 사무실을 탈출해야 한다.

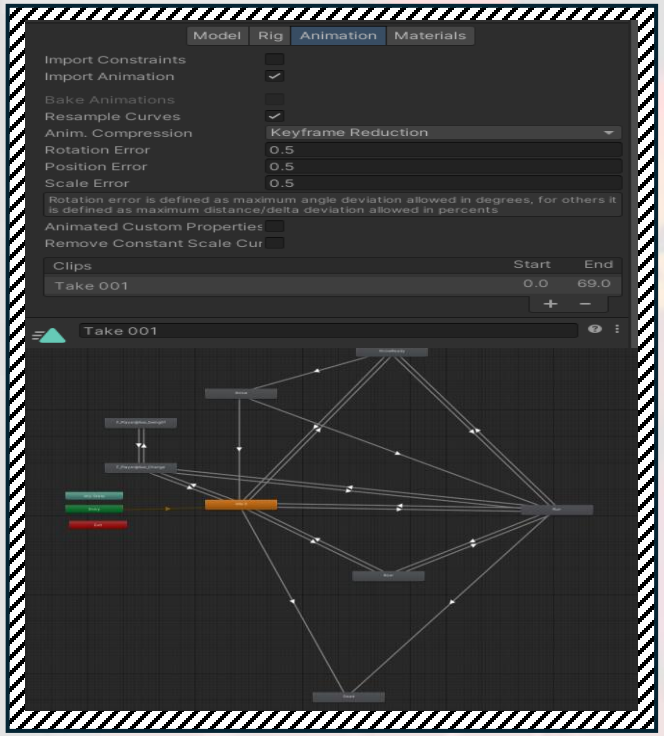
1. 플레이어 이동 및 조작, 애니메이션 시스템

문제 | 리소스 수정이 불가능한 상황에서 자연스러운 캐릭터 조작이 필요했기 때문.

방법 | 입력값과 시선 방향을 활용해 상하반신 연동 없이 자연스러운 애니메이션 구현.

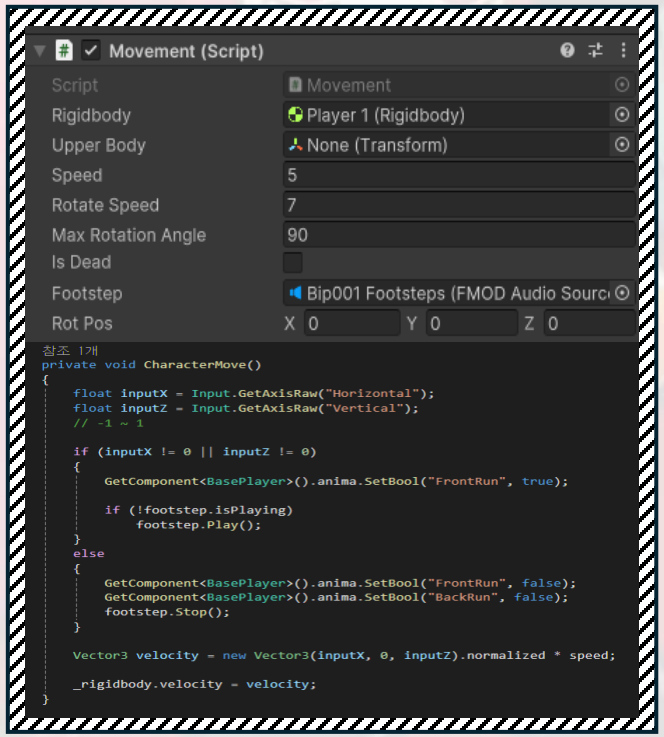
효과 | 리소스 수정 없이 조작감 개선 및 개발 일정 준수.

캐릭터 3D 모델링
캐릭터 애니메이션



당시 아직 배우고 있던 부족한 학생이었으므로
여러 이슈가 많았습니다. 그 중 하나가 캐릭터 애니메이션이
상반신과 하반신이 따로 움직이지 않던 이슈였습니다.

스크립트 및
인스펙터



시간이 촉박했고 학교 전시회에 일정을 맞춰야했기에
스크립트로 RAY의 방향과 키입력값에 맞춰
애니메이션이 출력되도록 하였습니다.

애니메이션이 자연스럽게
동작하는 모습

어색함이 크게 없고, 기한에 맞춰 해결하는 것에
성공하여 무사히 전시하게 되었습니다.

2. 화염방사기 시스템 구현

문제 | 이펙터로부터 받은 화염 이펙트를 적용했을 때, 연출이 자연스럽지 않다는 문제가 발생.

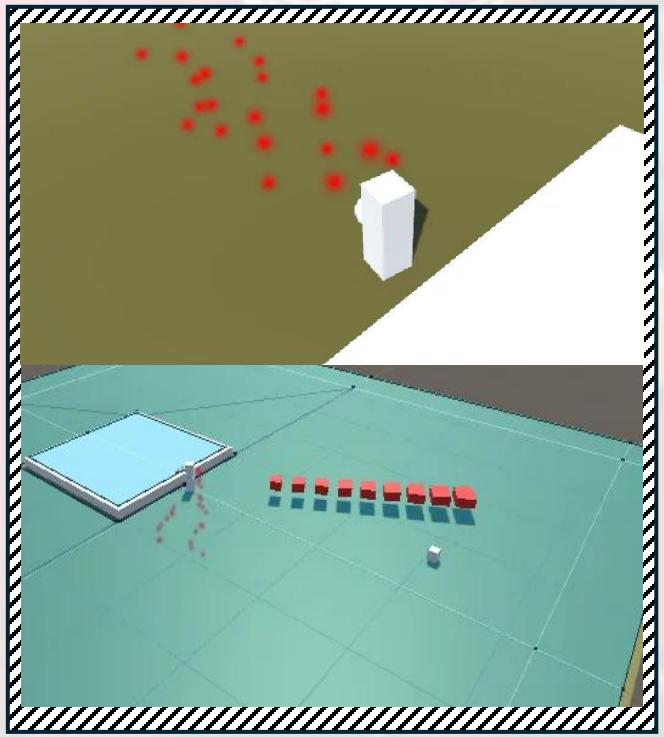
방법 | 프리팹 생성을 통해 다수의 화염 오브젝트를 발사하여, 연속적으로 퍼지는 자연스러운 화염방사기 이펙트를 구현.

효과 | 하나의 이펙트만 사용했을 때보다, 여러 이펙트가 이어지며 점진적으로 불이 퍼지는 자연스러운 화염 연출을 완성.

초기 화염 이펙트

초기 이펙트는 갑자기 불기둥이 생성되서
화염방사기의 총구에서 발사된다는 느낌이 없음

여러 방식의 화염발사를 테스트



결국 2번째 이미지의 방식으로 이펙트를 적용했을 때,
자연스러운 화염발사 이펙트가 적용되는 것을
확인하고 해당 방식을 채택하였습니다.

이펙트 적용시 자연스러운 화염방사기

불이 작은 상태에서
점점 커지는 자연스러운 화염방사기
구현을 완료했습니다.

2. 몬스터의 종류별로 개별 FSM 구성 및 구현

문제 | 몬스터마다 서로 다른 FSM을 적용해야하는 상황.

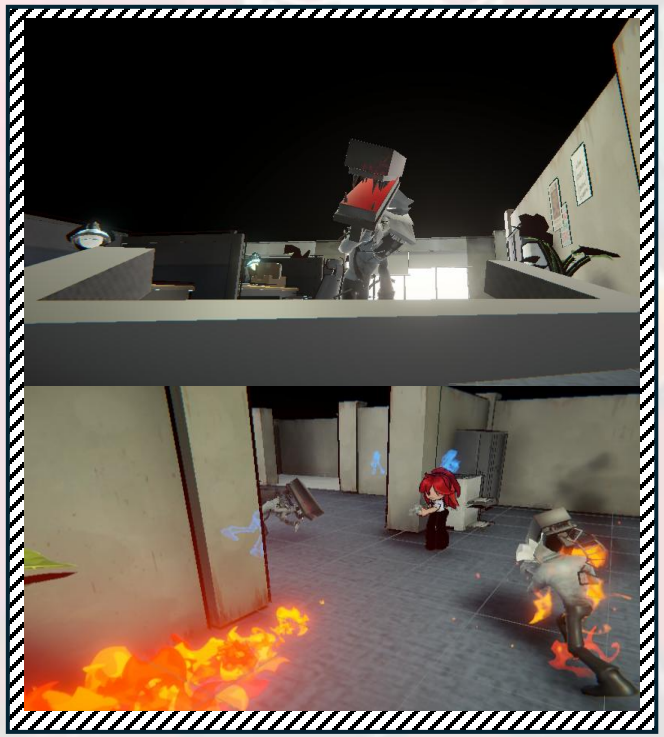
방법 | BaseMonster 스크립트를 부모로 사용하여 여러가지 FSM을 구현. 공통적으로 사용하는 기능은 통합

효과 | 모든 몬스터들이 공통적으로 사용하는 변수나 함수는 공유하기에 가독성이 좋고 최적화된 몬스터 FSM 스크립트 구현

자폭 몬스터
마인크래프트의 크리퍼와 동일

플레이어를 추적하는 건 동일하지만,
플레이어에게 근접하면 공격이 아니라 스스로 폭발하여
데미지를 줍니다.

원거리 공격을 하는 몬스터



원거리 공격을 하는 몬스터입니다.
플레이어를 추적하는 건 동일하지만
플레이어가 사정거리안으로 들어오면 공격합니다.

플레이어로부터 도망치는 몬스터와
플레이어를 추격하는 몬스터

대부분의 몬스터는 플레이어를 추적하지만
예외로 플레이어를 보면 도망치는 몬스터도 있습니다.
해당 몬스터는 추격값에 -를 해서 도망치게 하였습니다.

Thank you!
감사합니다!

2025 PORTFOLIO

전재연
nidhaku31@gmail.com
010-7374-3359

최종 업데이트 2025 09 18