# 백무송

# PORTFOLIO

## 나의 핵심 키워드

- 메모리 및 성능 최적화
- 의사소통, 커뮤니케이션
- 문제 해결을 위한 집념
- 배움에 대한 열정

---


## 기본사항

PHONE : 010 9957 3155 | Mail : musong0611@naver.com

거주지: 부산 | 군복무: 군 복무 완료(22사단 GOP)


## GIT HUB LINK

## BLOG LINK

# 스킬 & 이력사항

## 보유 기술






## 학력사항 & 교육이수

| | |
|---|---|
| 2026.02~현재 | 내일배움캠프 UNREAL |
| 2024.02~2025.02 | DIGIPEN ACADEMY IN BUSAN 수료 (부산콘텐츠아카데미) |
| 2018.03~2025.02 | 동의대학교 응용소프트웨어학과 졸업 |
| 2015.03~2018.02 | 부산동고등학교 졸업 |

## 프로젝트

| | |
|---|---|
| 2026.01~ 2026.01 | 3D_MOBA_NETWORK_GAME |
| 2025.08~ 현재 | Edge Drive GAME  블루프린트 C++ 변환 작업 |
| 2024.12~2025.02 | Edge Drive GAME 프로젝트 |
| 2024.09~2024.10 | WoThingThing 프로젝트 |
| 2024.06~2024.06 | HEX_RIS 프로젝트 |
| 2024.03~2024.06 | 부산 배경 2D 플랫폼 게임 프로젝트 |
| 2023.11~2023.12 | 2D 핵 앤 슬래시 게임 프로젝트 |

## 자격증

| | |
|---|---|
| 2025.11 | ITQ - 한글엑셀 A등급 |
| 2025.09 | ITQ - MS워드 A등급 |
| 2023.07 | 자동차운전면허증 2종보통 |

# PROJECT
# CONTENTS

01

## 3D_MOBA_NETWORK_GAME

개발 인원: 1명 (개인 프로젝트)

개발 기간: 2026.01.19 ~ 2026.01.22

02

## WOTHINGTHING

개발 인원: 3명 (팀프로젝트, 직책: 팀원)

개발 기간: 2024.09 ~ 2024.10

03

## Edge Drive GAME

개발 인원: 4명 (팀프로젝트, 직책: **팀장**)

개발 기간: 2024.12~2025.02 ,  2025.08~현재

04

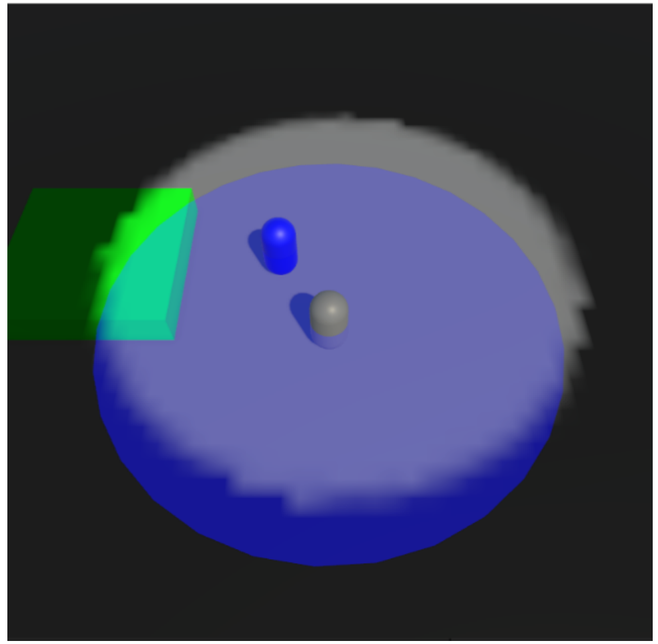
## 기타 프로젝트

개발 인원: 팀 프로젝트 + 개인프로젝트

개발 기간: 2023 ~ 2024

# 3D_MOBA_NETWORK_GAME

| Unity Netcode 기반 3D 쿼터뷰 MOBA 핵심 시스템 구현 | 사용 기술: Unity, Git, NGO, HLSL, C# | 2026.01.19~2026.01.22 |
|---|---|---|



## 전투 & 기반 시스템

- **논타겟 투사체 구현**

↳2D 마우스 좌표를 3D 월드 좌표로 투영 후 방향벡터로 구현

- **범위 공격 판정 시스템**

↳ 내적을 활용한 부채꼴 판정

- **비트 플래그를 이용한 상태 제어 시스템**

↳메모리와 성능을 고려한 비트 연산을 통한 상태 관리

- **타켓팅 공격 구현**

↳서버의 내부 보간으로 인해 발생하는 클라이언트 측
위치 지연 현상 파악 후 피격 시점 위치를 강제 동기화

- **오브젝트 풀링**

↳오브젝트 반복적인 생성/파괴로 인한 프레임 드랍 방지

- **SCRIPTABLE OBJECT 기반 스킬 시스템**

↳유지보수와 확장성 재사용성을 고려한 시스템 설계

## 시야 시스템

- **RTT 기반 동적 안개**

↳Culling Mask 방식을 채택

↳플레이어의 시야 영역을 별도의 Orthographic Camera 촬영

↳결과물을 Render Texture에 담아
맵 전체를 덮는 Fog Shader의 마스크 데이터로 전달.

↳Render Texture의 안개 영역으로, 시야 확보 영역매핑
셰이더가 데이터를 효율적으로 해석

- **공간 분할과 그리드로 부쉬 구현**

↳배열 인덱싱을 통한 부쉬 판정 로직

↳레이어 기반의 맵 오브젝트 스캔 및 2차원 그리드 데이터 전처리

↳공간 분할 알고리즘을 적용하여 오브젝트 탐색

- **멀티플레이어 시야 동기화**

↳서버에서 체크하여 로컬 플레이어의 시야 마스크만 활성

↳각 클라이언트가 독립적으로 시야 상태를 비교해
MeshRenderer를 제어

# 핵심 기술 & 트러블 슈팅

## 서버와 클라이언트 간의 투사체 위치 동기화 (LERP 보간)

- 문제

멀티 플레이 중 서버에 들어온 플레이어가 상대 플레이어한테 유도 투사체 공격할 시 충돌하지도 않았는데 투사체가 증발하는 현상 발생

- 원인

Network Transform의 기본 기능을 믿고 구현을 한게 원인

네트워크 지연이 발생하는 실제 환경에서는 클라이언트가 서버보다 과거의 위치를 그리기 때문에 닿기 전에 사라지는 현상이 발생

- 해결 방법

서버가 혼자 판단해서 지우지 말고, 클라이언트에게 타겟에 닿으면 그래픽 끄라는 명령을 내리는 방식

이렇게 하면 모든 클라이언트가 각자의 화면에서 타겟에 닿는 순간 사라지게 함

- 결과

이제 플레이어가 상대 플레이어한테 유도 투사체를 공격할 시 상대 플레이어를 충돌하고 투사체가 사라짐

## 코드 및 문제 동영상 링크

### 수정 전 코드

```
private void OnHit()
{
    DestroyProjectile();
}
```

### 수정 후 코드

```
private void OnHit()
{
    // 클라이언트에게 맞았으니 사라져라고 보냄
    if (_target.TryGetComponent(out NetworkObject targetNetObj))
    {
        OnHitClientRpc(targetNetObj.NetworkObjectId);
    }
    else
    {
        OnHitClientRpc(0);
    }
    StartCoroutine(DelayedDespawn());
}
[ClientRpc]
참조 2개
private void OnHitClientRpc(ulong targetId)
{
    // 위치 보정 (타겟 몸통으로 이동)
    if (targetId != 0 &&
        NetworkManager.Singleton.SpawnManager.
        SpawnedObjects.
        TryGetValue(targetId, out var targetObj))
    {
        transform.position = targetObj.transform.position + Vector3.up;
    }
    // 눈앞에서 사라지게 처리
    SetVisuals(false);
}
```

### 버그 발생 영상

### 문제 해결 영상

# 핵심 기술 & 트러블 슈팅

## 공간 분할과 그리드로 부쉬 구현

- 기술의 한계

플레이어가 부쉬를 감지하는 로직은 충돌 이벤트가 발생하면 유니티 오브젝트의 고유 번호를 활용하여 관리를 했는데 게임 볼륨이 커지면 부쉬 개수가 많아질 수록 성능과 메모리 최적화에 안 좋음

- 원인

**물리 엔진 의존성이 커서 부쉬 개수가 많이 늘어날 경우 충돌이 많아지고 물리 연산이 많아져서 프레임 저하가 발생함**

- 해결 방법

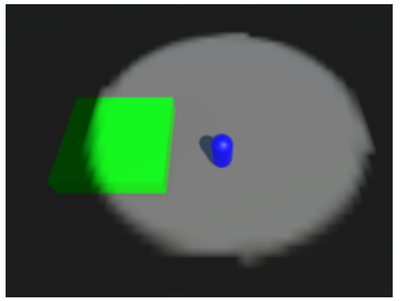
맵을 데이터 배열로 관리하는 공간분할 기법을 사용하여 맵에 가상의 선으로 칸을 나누고 게임 시작 시에만 레이어 감지를 통해 부쉬 위치를 배열에 미리 저장함
이후 플레이어가 움직일 때는 물리 연산 없이, 자신의 좌표를 수학적으로 계산해 배열 인덱스로 변환함으로써 현재 위치의 부쉬 정보를 확인하도록 최적화함

- 결과

위치가 업데이트되는 즉시 수학적으로 계산해버리므로, 위치 이동과 동시에 부쉬 은신 판정이 바로 이루어지고 빠르고 정확한 동기화가 가능해짐

## 코드 및 결과 사진

부쉬 밖 플레이어 시야



부쉬 안 플레이어 시야

핵심 기술 코드

```
private void BakeBushData()
{
    if (_bushMap == null) return;

    int xSize = _bushMap.GetLength(0);
    int zSize = _bushMap.GetLength(1);

    for (int x = 0; x < xSize; x++)
    {
        for (int z = 0; z < zSize; z++)
        {
            // 각 그리드 칸의 중심 월드 좌표 계산
            Vector3 cellWorldPos = GetWorldPos(x, z);

            // 부쉬 레이어만 감지
            Collider[] hitBushes = Physics.OverlapBox(cellWorldPos + Vector3.up * 0.5f,
                new Vector3(cellSize / 2, 1f, cellSize / 2), Quaternion.identity, bushLayer);

            if (hitBushes.Length > 0)
            {
                // 부쉬 오브젝트의 GetInstanceID를 고유 ID로 사용
                _bushMap[x, z] = hitBushes[0].gameObject.GetInstanceID();
            }
            else
            {
                _bushMap[x, z] = 0; // 부쉬 없음
            }
        }
    }
}
```

# Wothingthing

| 자체 커스텀 엔진을 활용하여 KATANA ZERO , THINGTHING 스타일의 전투를 구현한 2D 액션 게임 | 사용 기술: Custom Engine, Git, C++ | 2024.09~2024.10 |
|---|---|---|


트레일러

## 담당한 기능 및 구현 기술

- **State Machine을 이용하여 적 AI 구현**
  ↳조건에 따라 추적, 공격, 대기 상태로 변경 구현
  ↳보스가 피격 시 일정 시간 동안 경직 상태가 되도록 설계
- **충돌 감지영역 활용할 수 있게 구현**
  ↳충돌 판정을 AABB 방식으로 구현
  ↳플랫폼 모서리 감지 로직을 적용, 적이 추락하지 않게 구현
  ↳플레이어 감지 범위를 통해 AI 인식 기능 구현
- **이동, 공격, 사망 애니메이션을 상태별로 연동 및 동작**
  ↳스프라이트 시트 방식 이용

## 핵심 기술 및 트러블 슈팅(State Machine을 적용하여 적 AI의 제어)

- 구현 방법

컴포넌트 기반으로 적의 행동이 다양해질 것을 대비해, 상태를 클래스로 분리하여 관리하는 FSM 패턴으로 구현하고 매 프레임 행동을 처리하게 구현

- 문제 발생 및 원인

초기 구현 단계에서 AI 컴포넌트의 설정값(회전 시간, 정찰 범위 등)을 런타임 중에 변경했으나, 현재 실행 중인 상태 로직에 반영되지 않는 문제가 발생함

상태 객체를 생성할 때 생성자를 통해 데이터를 단순히 값으로 전달했던 것이 원인 (Call by Value)

- 해결방법

제를 해결하기 위해 Call by Value 방식을 Call by Reference 방식으로 리팩토링했음.

데이터의 원천인 AiComponent 자체의 포인터를 멤버로 소유

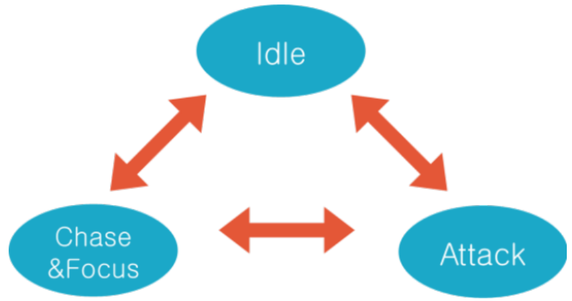
this 포인터를 통해 컴포넌트 자신을 주입

Getter 함수를 통해 매 프레임 최신 데이터에 직접 접근하도록 수정

- 결과

일일이 매개변수를 하드코딩으로 변수를 넘겨주지 않고 AiComponent저장된 변수를 참조하여 값을 받아올 수 있게 됨

### AI 상태 로직




### 수정 전 코드

```
void AiComponent::SetState(const std::string& state_name, const std::string& enemyCategories)
{
    e_state_name = state_name;
    e_Categories = enemyCategories;
    if (e_Categories == "Melee")
    {
        if (e_state_name == "IDLE")
        {
            ESM::IDLE* p = new ESM::IDLE(m_pOwner, Player, set_dir, Time_dir, PlatForm, e_state_name, FirstPlacePos);
            esm->ChangeState(p);//p를 넘겨주면 자기자신을 m_pOwner를 넘겨주는거니 참조 한다는거다
        }
    }
```

### 수정 후 코드

```
void AiComponent::SetState(const std::string& state_name, const std::string& enemyCategories)
{
    e_state_name = state_name;
    e_Categories = enemyCategories;

    if (e_Categories == "Melee")
    {
        if (e_state_name == "IDLE")
        {
            ESM::IDLE* p = new ESM::IDLE(m_pOwner, this, e_state_name);
            esm->ChangeState(p);
        }
    }
```

# Edge Drive GAME

**Unreal Engine5 기반 3D 소울라이크 보스 러시 게임 개발,팀장으로서 팀을 총괄**

**사용 기술: Unreal Engine, C++, Google Drive,Git**

**2024.12~2025.02**
**2025.08~현재 (블루프린트 C++전환)**



트레일러

## 담당한 기능 및 구현 기술

- **보스공격 패턴 기획 및 밸런스 조정**
  ↳EQS 활용하여 보스 순간이동 공격
  ↳Behavior Tree, AI Controller사용
- **데미지 시스템 설계, 보스의 공격 타입 컴포넌트로 관리**
  ↳틱 마다 검사하지않고 이벤트 기반으로 작동
  ↳공격하는 쪽에서 상대 확인 없이 인터페이스만 호출하면 됨
- **보스 에셋 적용**
  ↳IK RIG과 IK RETARGET을 사용

## 트러블 슈팅(보스 시간 가속 스킬오류)

- 문제

보스 AI가 시간 속도 조절 스킬을 사용하면 다음 패턴으로 넘어가지 않고 멈추거나 허공에 공격하는 현상이 발생

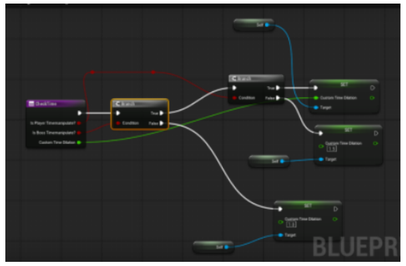
- 원인

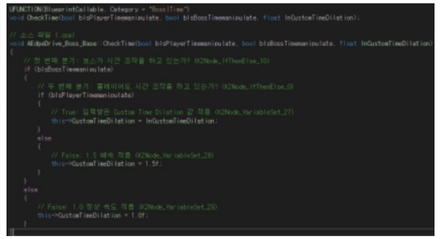
Behavior Tree의 Tick 주기가 원인이라 판단하여 디버깅 했으나, 근본적인 원인은 엔진의 Delay 노드와 타이머가 **Global Time Dilation의 영향을 받는 방식**이였음.

보스에게 Custom Time Dilation을 적용하여 속도를 높였음에도, 내부 로직의 Delay은 orld Time을 기준으로 동작하여, **보정된 속도만큼 대기 시간이 길어지거나 짧아지는 왜곡**이 발생

- 해결 방법

보스가 시간 가속 상태면 Custom Time Dilation 수치에 반비례하도록 Delay 시간을 보정하는 수식을 적용



## 핵심 기술(전투 데미지 시스템)

- 구현 방법

인터페이스, 데이터 구조체 및 열거형, 액터 컴포넌트, 이벤트 디스패처를 사용하여 데미지 데이터, 체력관리, 상태관리를 할 수 있음

- 기술 선택 이유

상속 방식은 협업 할 시 재사용성이 낮고 유연성이 부족

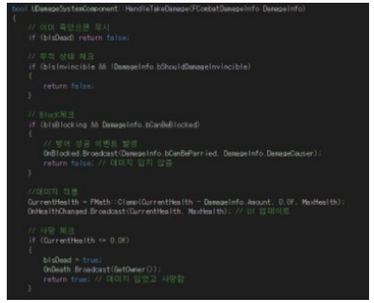
GAS (데이터 기반의 스킬 및 속성 관리 시스템)

처음 접해본 기술이라 담당한 다른 파트도 병행하면서 짧은 시간 내에 완벽하게 구현해야 하므로 제외

팀원이 쉽게 사용 가능한 **컴포넌트 형식**으로 구현

- 결과

모든 데미지 판정이 필요한 오브젝트에 사용가능하고 **다른 팀원도 간편하게 수정 및 사용 가능**



# 기타 프로젝트

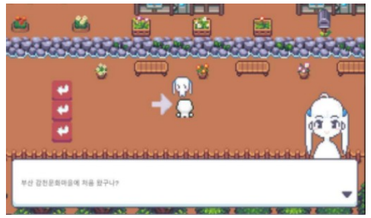

트레일러

## HEX_RIS

DigiPen C-Processing을 활용하여 장애물 회피와 Tetris 퍼즐 요소를 결합한 아케이드 스타일의 2D 게임 제작

- 플레이어와 장애물 간 충돌 판정 로직 구현
- 배경음악 및 효과음 재생 기능 추가
- 플레이어의 생존 시간 기록 및 최고 기록 비교 기능 구현
- 사용자 문서 폴더 내에 디렉토리를 생성 및 기록 저장 기능

**개발 인원: 2명 (팀 프로젝트)**
**사용 기술: C-Processing, C, Git**
**개발기간: 2024.06.17-2024.06.24**




## 부산 2D 플랫폼 게임

부산이 인구 유출이 막기위해 부산이 좋다는 이미지를 Unity를 이용해 부산을 소개하는 플랫폼 게임 구현

- 대화 시스템(텍스트) 및 대화창 UI 구현
- 미니게임 점프맵 게임 구현
- 레이캐스트를 이용한 오브젝트 식별
- 맵을 이동 할 수 있는 포탈 구현

**개발 인원: 4명 (팀 프로젝트)**
**사용 기술: Unity, Git, C#**
**개발기간: 2024.03-2024.06**


## 2D 핵 앤 슬래시 게임

Unity 입문 후 독학을 통해 핵앤슬래시 게임구현하며, 게임 메커니즘 및 유니티 활용을 중점으로 프로젝트 진행

- 몬스터 생성 방법 메모리 풀 사용
- 무작위 몬스터가 플레이어를 추적 기능 구현
- 버튼 기능을 통한 무기 교체 기능 구현
- 유니티 애니메이터를 통해 메인 캐릭터와 몬스터 이동의 자연스러움 구현

**개발 인원: 1명 (개인 프로젝트)**
**사용 기술: Unity, C#**
**개발기간: 2023.11-2023.12**