Portfolio

버그를 찾고 해결하는걸 좋아하는
클라이언트 개발자 백무송입니다.

HISTORY.

클라이언트 개발자 ,
백무송

PROFILE.

Name: 백무송 (1999.06.11)

Contact: 010-9957-3155

Email: musong0611@naver.com

Git: sleeptired

Blog: 백무송 개발 블로그

SKILL.

Unreal Engine 5

C++

Unity

C#

EDUCATION & TRAINING.

2026.	내일배움캠프 Unreal8기
2024.	DigiPen Academy 1기
2018.	동의대학교 응용소프트웨어
2015.	부산동고등학교

PROJECT.

2026.	자동차 자율주행시뮬레이션
2025.	Edge Drive
2024.	Wothingthing
2026.	TeamName (TEXT-RPG)
2023.	기타 프로젝트

KEYWORD.

최적화 | 분석 | 리팩토링 | 협업 & 조율

동적 환경 반응형 스플라인 기반 자율주행 및 사고 취약 구간 분석 시뮬레이터

기간: 2026.05.04 ~ 2026.05.27

사용 기술: Unreal Engine 5, C++, Git

담당 기능: 차량 , 날씨 시스템, 코드 통합

트러블 슈팅 & 문제 해결.

문제.

팀원의 요청으로 레벨 내에 자율주행 차량을 하나 더 배치하여 실제 도로 환경을 구성했을 때,
프레임이 60에서 30까지 떨어지는 (FPS 저하) 현상이 발생

원인.

플레이어 차량과 NPC 차량이 동일한 클래스를 사용하고 있어 NPC 차량임에도 불구하고 플레이어 전용인 라이다 센서(비동기 라인 트레이스, BEV 텍스처 변환), 카메라 센서(Scene Capture 2D, 렌즈 왜곡 연산), 날씨 파티클(나이아가라 시뮬레이션), 데이터 로거(UTM 좌표 변환, CSV 파일 I/O) 컴포넌트들이 모두 스폰되고 매 프레임마다 연산을 수행하여 프레임 드랍 유발

해결

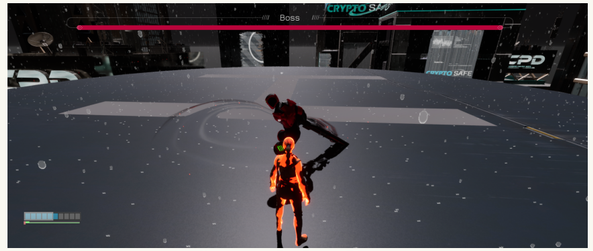
BeginPlay 단계에서 플레이어인지 확인하는 로직을 추가하여 아닐경우, 자율주행에 필수적인 SplineFollower를 제외한 모든 센서 및 컴포넌트를 제거하게 하여 FPS 저하 현상 해결
추후 무거운 센서와 카메라를 가진 Player와 경량화된 자율주행 모듈만 가진 NPC를 따로 상속받고 관리하도록 분리했습니다.

Edge Drive. (3D소울류 게임)

기간: 2024.12 ~ 2025.02

사용 기술: Unreal Engine 5, C++, BP

담당 기능: 보스, 데미지 시스템, 에셋 적용



트러블 슈팅 & 문제 해결.

문제.

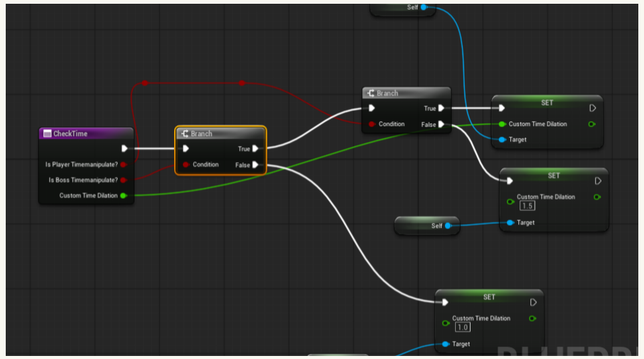
보스 AI가 자신의 속도를 빠르게 하는 시간 가속 스킬을 사용하면 다음 패턴으로 넘어가지 않고 멈추거나 허공에 공격하는 현상이 발생

원인.

Behavior Tree의 Tick 주기가 원인이라 판단하여 디버깅 했으나, 근본적인 원인은 엔진의 Delay 노드와 타이머가 Global Time Dilation의 영향을 받는 방식이였음.
보스에게 Custom Time Dilation을 적용하여 속도를 높였음에도, 내부 로직의 Delay은 World Time을 기준으로 동작하여, 보정된 속도만큼 대기 시간이 길어지거나 짧아지는 왜곡이 발생

해결

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




```
UFUNCTION(BlueprintCallable, Category = "BossTime")
void CheckTime(bool bIsPlayerTimemanipulate, bool bIsBossTimemanipulate, float InCustomTimeDilation);

// 소스 파일 (.cpp)
void AEdgeDrive_Boss_Base::CheckTime(bool bIsPlayerTimemanipulate, bool bIsBossTimemanipulate, float InCustomTimeDilation)
{
    // 첫 번째 분기: 보스가 시간 조작을 하고 있는가? (K2Node_IfThenElse_10)
    if (bIsBossTimemanipulate)
    {
        // 두 번째 분기: 플레이어도 시간 조작을 하고 있는가? (K2Node_IfThenElse_0)
        if (bIsPlayerTimemanipulate)
        {
            // True: 입력받은 Custom Time Dilation 값 적용 (K2Node_VariableSet_27)
            this->CustomTimeDilation = InCustomTimeDilation;
        }
        else
        {
            // False: 1.5 배속 적용 (K2Node_VariableSet_28)
            this->CustomTimeDilation = 1.5f;
        }
    }
    else
    {
        // False: 1.0 정상 속도 적용 (K2Node_VariableSet_29)
        this->CustomTimeDilation = 1.0f;
    }
}
```

Wothingthing (2D 액션 게임)

기간: 2024.09 ~ 2024.10

사용 기술: Open GL, C++, Git

담당 기능: 적 AI, 충돌 감지, 애니메이션

트러블 슈팅 & 문제 해결.

문제.

다수의 적 AI를 맵에 배치했을 때, 마지막으로 스폰된 AI만 지형(낭떠러지)을 정상적으로 인식하고,
나머지 AI들은 낭떠러지를 인식하지 못해 추락하는 버그 발생.

원인.

충돌 처리 과정에서 매개변술플랫폼 정보를 전달할 때 단일 공유 변수를 사용하여 참조 덮어쓰기가 발생함.
적 AI가 '마지막으로 생성된 AI의 플랫폼 메모리 주소'로 덮어쓰기가 되어, 각 AI가 발밑에 있는 실제 지형과 참조하는 지형이 불일치하는 현상이 원인.

해결

옵저버 패턴(Observer Pattern) 기반의 이벤트 주도 아키텍처를 도입함.
전역 충돌 매니저에서 적이 지형과 닿는 순간 객체 간 1:1 매핑 정보를 담은 이벤트(Event)를 발행(Broadcast)하고, AI 내부의 구독자(Subscriber)가 자신에게 해당하는 이벤트만 수신하여 지형 정보를 갱신하도록 설계함.

```
//Enemy
for (int i = 0; i < Enemy.size(); i++)
{
    if (ColliderManager::GetInst()->IsCollision(Enemy[i], obj))
    {
        HandleCollision(Enemy[i], obj);
        //AI COMP
        Enemy_Platform_Collision_Event* e_p_c_e = new Enemy_Platform_Collision_Event(obj, enemy:Enemy[i]);
        EventManager::GetInst()->AddEvent(e_p_c_e);

    }
}
```

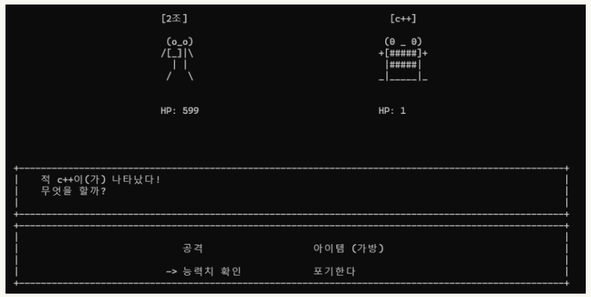
```
void ChasePlatFormSettor::OnEvent(Event* ev)
{
    Enemy_Platform_Collision_Event* e_p_c_e = static_cast<Enemy_Platform_Collision_Event*>(ev);
    if (e_p_c_e->enemy == Enemy_Chase->GetOwner())
    {
        Enemy_Chase->PlatForm = e_p_c_e->platform;
    }
}
```

TeamName (TEXT-RPG)

기간: 2026.03.25 ~ 2026.03.31

사용 기술: C++, Git

담당 기능: 씬 매니저 제작, 코드 통합



트러블 슈팅 & 문제 해결.

문제.

Stack 구조의 SceneManager에서 인벤토리를 열었다가 닫고 기존 씬으로 돌아올 때, 기존 씬의 몬스터 데이터가 유지안되는 현상 발생했습니다.

원인.

스택 기반 씬 복귀 과정에서 최초 1회만 실행되어야 할 Init() 함수가 중복 호출되어, 새 몬스터 객체를 생성하고 기존 몬스터 데이터를 덮어쓰는 생명주기 설계 오류가 원인이였습니다.

해결

씬 매니저의 전에 씬(화면)으로 돌아가는 로직에서 Init() 호출을 제거하여, 팝업 씬 종료 시 초기화 없이 기존 스택에 보존된 씬의 Update()와 Render()만 재개되도록 생명주기 파이프라인을 수정하였습니다.


```
void SceneManager::Return_Scene()
{
    if ( !SceneStack.empty() )
    {
        SceneStack.top()->Exit();

        delete SceneStack.top();//동적으로 받은 씬이니까 힙 메모리 해제

        SceneStack.pop();//꺼내야하니까 pop
    }
    //트러블 슈팅용
    //if ( !SceneStack.empty() )
    //{
    //
    //  SceneStack.top()->Init();
    //}
}
```

버그 화면

협업/조율.

TEXT-RPG

프로젝트 초기 기획 단계에서 역할 경계 모호로 인한 코드 충돌 및 협업 저하 리스크가 예상되어, 팀원 간 역할 분담 및 아키텍처 설계에 난항 발생.
시스템 간의 의존성을 최소화할 수 있는 모듈화된 아키텍처 초안을 설계하여 팀에 제안함.
전체 구조를 Game/Scene Manager, Player, 상점/인벤토리, 몬스터, UI로 영역으로 분할하여,역할 분담을 주도적으로 완수함.

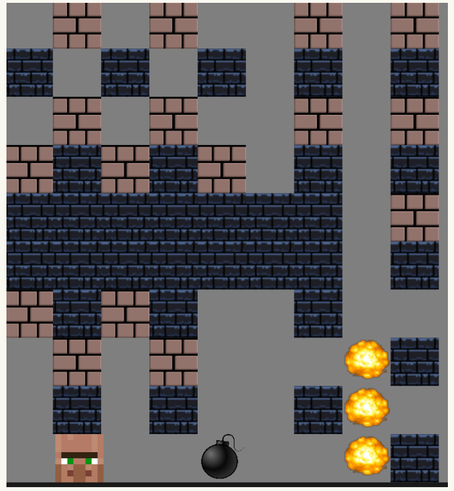
EdgeDrive

EdgeDrive 프로젝트 기획 단계에서 전투 시스템 구현 방식(독립된 3D/2D 레벨 분할 vs 3D 전투 내 2D 패턴 기믹 적용)을 두고 팀원 간 의견 충돌 발생.
각 팀원이 제시한 방향성의 핵심 의도와 장단점을 객관적으로 정리하여 공유하고, 프로젝트의 전체적인 흐름을 기준으로 조율을 진행.
'3D 전투 중 2D 기믹 적용' 방식으로 합의를 도출한 뒤, 제안자에게 2D 패턴 구현을 일임함. 동시에 다른 팀원과는 3D 보스와 플레이어 로직으로 역할을 명확히 분담하여 갈등을 해결하고 개발을 진행함

자동차 자율주행시뮬레이션

팀원이 자율 주행 로직 구현 중 차량이 목적지 도착 후 브레이크를 실행 할 시 후진하는 버그 발생
시행착오 내역을 팀원과 적극적으로 리뷰하며 문제를 이관받음.
이후 표면적 증상이 아닌 언리얼 엔진(Chaos Vehicle)의 코드를 분석.
엔진 내부 상태 머신의 오토 후진 유발 변수를 식별하고, 이를 수동 제어 모드로 변경하여 버그를 해결.

기타 프로젝트.



2D 봄버맨 게임

기간: 2024.10 ~ 2024.1.01

사용 기술: Open GL, DearImGui, C++, Git

담당 기능

DearImGui 맵 에디터 기능
맵 및 오브젝트 Json파일로 관리
피격 판정

HEX_RIS (TETRIS+ SUPER HEXAGON)

기간: 2024.06.17 ~ 2024.06.24

사용 기술: C, Git

담당 기능

게임 생존 시간 및 최고 기록 기능
배경음악 및 효과음 재생 기능
충돌 판정 로직

SEA BUSAN (2D 플랫폼 게임)

기간: 2024.03 ~ 2024.06

사용 기술: Unity, C#

담당 기능

대화 시스템 및 UI
미니게임 점프맵
포탈 기능

2D 핵 앤 슬래시 게임

기간: 2023.11 ~ 2023.12

사용 기술: Unity, C#

담당 기능

몬스터 생성 및 추적 기능
무기 교체 시스템
애니메이션

Thank you

THANK YOU FOR READING.