GAME PROGRAMMER PORTFOLIO

박진원 Jinwon Park

플레이 규칙을 구현하고,
게임 상태를 서버와 연결합니다.

C++과 C#으로 Unreal·Unity 팀 게임의 전투, 퍼즐 상호작용, 카메라·UI와 클라우드 저장을 구현했습니다. 기획·아트·프로그래머와 동작 조건을 맞추고, 맡은 기능을 레벨과 팀 코드에 연결했습니다.

C++ / C# / Unreal Engine / Unity

HTTP·JSON · ASP.NET Core · Azure Cosmos DB · Git / Perforce

프로젝트와 담당 기능

02-03 **Triad of Valor**
보스·패링·체력 처리 / 계정 연동·클라우드 저장

04 **An Omen in the Mirror**
피아노·문·창문과 퍼즐 상호작용

05-06 **함께 만든 게임과 개인 게임**
ROOT·Jumppers / Way Back Home·Dice & Castle

07-08 **소프트웨어와 설계**
개인 앱·FinOps Guard / TraceAX·PostgreSQL

09 **C++ 시스템 구현**
경로 탐색·공간 구조·동시성·진단 도구

DigiPen Institute of Technology

B.S. in Computer Science in Real-Time Interactive Simulation | 2026.04 졸업

온라인 포트폴리오 github.com/nowjinpark

Triad of Valor

Unreal Engine 5 · 2025.09-2026.04 · Kimbap Games 6인 팀 · 3D 협동 액션 RPG

보스 공격 상태·패링·체력 처리를 맡았습니다. 이미지는 팀 공개 튜토리얼 영상의 실제 플레이 장면입니다.

공격이 끝나거나 실패하면 상태를 정리

보스의 공격 시작·종료와 피격·사망 반응을 BossCharacter에 구현했습니다. 공격 중 여부와 재사용 시간을 확인한 뒤 서버 RPC로 몽타주를 재생하고, 재생 종료·실패 시 공격 상태를 정리하도록 했습니다. 보스 체력 UI도 함께 연결했습니다.

애니메이션 구간에 맞춰 패링을 판정

일반 적과 보스가 애니메이션의 정해진 구간에서만 패링할 수 있도록 연결했습니다. AnimNotifyState의 시작과 종료가 패링 가능 상태를 바꾸게 해, 전투 판정과 애니메이션의 기준을 맞췄습니다.

포인트 데미지의 중복 바인딩 방지

담당 HealthComponent.cpp에서 PointDamage를 추가로 바인딩하지 않도록 처리했습니다. AnyDamage와 함께 발생하는 포인트 데미지를 두 번 처리하지 않게 한 조정입니다. 체력 변경은 서버 권한으로 제한하고 값을 복제하며 변경 이벤트를 방송합니다.

팀 게임 튜토리얼 영상

담당: 보스 공격 상태·체력 이벤트 처리·패링 알림. 보스 AI와 페이즈, 플레이어의 다른 기능은 팀원들의 모듈과 함께 동작합니다.

Triad of Valor - Cloud Save

Unreal C++ · ASP.NET Core · Azure Cosmos DB · 계정 연동과 진행 상태 저장

게임 상태를 저장 API와 레벨 전환 흐름에 연결했습니다.

CloudSaveComponent와 C# 저장·분석 API를 구현해 캐릭터의 진행 데이터를 JSON으로 저장·조회했습니다. 불러온 결과는 팀이 만든 인벤토리·장착 상태·레벨 전환 흐름에 적용하도록 연결했습니다.

Unreal Client

게임 상태 직렬화·HTTP 요청
서버 권한으로 데이터 적용

ASP.NET Core

중첩 객체와 배열 보존
기본값 처리·응답 정규화

Cosmos DB

플레이어·캐릭터 문서 저장
조회와 분석 API에 제공

인벤토리 구조를 보존하는 저장·조회

인벤토리의 중첩 객체와 배열을 보존하고, 조회한 JSON을 게임에서 다시 읽을 수 있도록 처리했습니다. 돈·플레이 시간·직업·체력과 슬롯 데이터를 API에 전달하고, 누락된 필드에만 기본값을 채웁니다. 조회 시 달라질 수 있는 JSON 표현도 정규화했습니다.

레벨 전환 중 목적지 위치를 지키는 연동

팀의 레벨 전환 처리에 CloudSaveComponent를 연결해 저장·조회가 이어지도록 했습니다. 이전 좌표가 새 레벨의 포털 도착 위치를 덮을 수 있어, 팀 캐릭터 코드의 전환 캐시와 지역별 위치 조건을 따라 연동했습니다. 전환 캐시와 위치 복원 코드는 팀 공동 구현입니다.

계정 요청의 오류 처리와 저장 데이터 분석

AuthSubsystem과 Slate 로그인 화면에서 제한 시간·네트워크 실패·HTTP 오류를 구분하고 응답을 Game Thread로 전달했습니다. 별도 분석 API는 저장된 캐릭터 데이터의 요약, 지역·플레이 시간 구간별 분포와 CSV 출력을 제공합니다.

담당: CloudSaveComponent·C# API·계정 연동. 저장 API는 수업 예제에서 출발해 교수·AI의 도움을 받았습니다. 인벤토리 적용·전환 캐시·위치 복원은 팀 공동 코드입니다.

An Omen in the Mirror

Unreal Engine 5.4 · 2024-2025 · Clockworks 팀 프로젝트 · 게임플레이 프로그래머

피아노·문·창문의 상호작용을 맡았습니다. 과거와 미래의 저택을 오가는 퍼즐 어드벤처의 팀 공동 제작 화면입니다.

앞선 오입력 뒤에도 정답 시퀀스를 인식

정답 앞에 다른 음을 눌러도 뒤이어 입력한 올바른 시퀀스를 인식하도록 수정했습니다. 입력이 쌓여 재시도를 방해하지 않도록 대기 시간에 따른 초기화와 수동 초기화를 추가했습니다. 정답 시퀀스와 연결할 오브젝트는 Level Editor에서 설정하게 했습니다.

문·창문·램프의 상태와 상호작용

플레이어가 서 있는 방향에 맞춰 문이 열리도록 조정하고, 애니메이션 중에는 중복 조작을 막았습니다. 잠긴 문의 안내 UI 조건과 문틀 구조를 정리하고, 창문과 램프의 동작을 퍼즐에 연결했습니다.

기획·아트와 동작 조건을 맞춰 레벨에 연결

기획·오디오 담당자와 입력 규칙과 오브젝트 연결 방식을 조정하고, 아트 모델의 피벗에 맞춰 Blueprint를 수정했습니다. 팀이 정한 공통 퍼즐 구조를 바탕으로 피아노·문과 압력판 프로토타입을 레벨에 연결했습니다.

담당: 퍼즐 오브젝트·상호작용·입력 규칙. 환경 아트·세계관·전체 레벨은 팀 공동 작업입니다.
Unreal Engine 5.4 · Blueprint · C++ · UMG · Perforce · ClickUp

ROOT//ACCESS / Jumppers

브라우저 게임과 Unity 팀 게임에서 조작·카메라·플레이 흐름을 구현했습니다.

ROOT//ACCESS

2026.08 · HTML / CSS / JavaScript · 개인 기획·AI 보조 개발·검증

Codex와 웹 잠입 퍼즐을 만들고 캠페인·해답 재생 검사로 진행 흐름을 확인했습니다. 카드와 패널 위에서 경비를 피하는 6개 장·12개 작전·36개 구간에 이어하기·백업·복구와 한국어·영어 UI를 연결했습니다.

좌표 갱신 처리를 고친 뒤, 개발 당시 60초 순찰 검사로 경비의 이동이 이어지는지 확인했습니다. 화면 배치가 바뀌면 경비가 멈추던 문제를 AI와 조사하고, 배치 변경 후의 좌표 처리에 수정 범위를 좁혔습니다.

ROOT//ACCESS 플레이

Jumppers

2025.11 · Unity 6 · Nintendo Switch 개발킷으로 제작한 팀 게임

점프를 따라가는 카메라와 HUD·메뉴·착지 효과를 구현했습니다. 수직 속도에 따른 선행값, dead zone과 SmoothDamp로 카메라 추적을 조정하고 점수·타이머, 일시정지·재시작과 FPS 옵션을 연결했습니다.

Nintendo Switch 개발킷을 사용한 Unity 프로젝트이며, 별도의 WebGL 빌드로 웹에서도 플레이할 수 있습니다. 카메라·UI·착지 피드백은 제 담당이고, 이동과 절차적 생성은 팀원들의 작업입니다.

Jumppers 웹 플레이

카메라 PR #2

UI PR #3

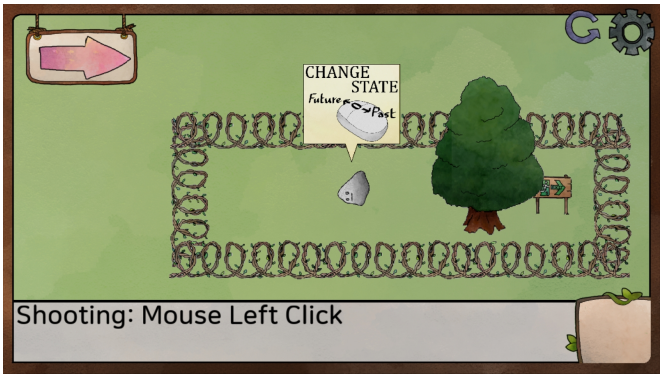
두 이미지는 실제 게임 화면입니다.

Way Back Home / Dice & Castle

팀 자체 엔진의 물리·충돌부터 턴제 전투와 튜토리얼까지

Way Back Home

학부 GAM200/250 · 팀 자체 C++ 엔진 BLUE · JEANs 3인 팀

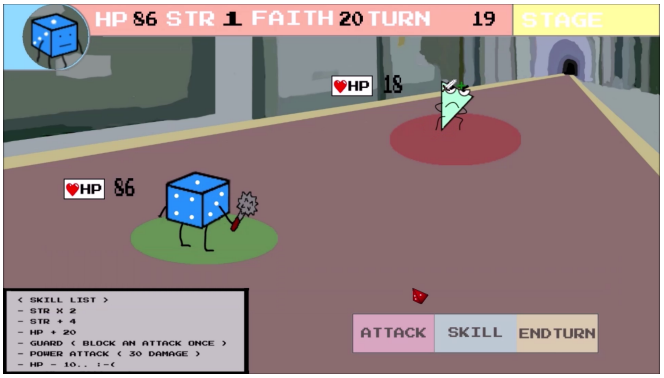


물리 프로그래머·디자인 리드로 원·사각형 충돌, Rigidbody와 퍼즐 컨트롤러·레벨을 구현했습니다. 팀이 확장한 BLUE 엔진 위에서 사물의 시간을 바꾸고 오브젝트를 조합해 길을 여는 퍼즐을 만들었습니다.

벽 접촉 시 이전 위치로 돌아가 이동하지 못하는 현상을 접촉 보정값과 좌표 갱신을 조정하며 다뤘습니다. 문·벽·웅덩이에 물리 기능을 연결했습니다. BLUE는 팀원이 시작한 엔진을 JEANs 팀이 함께 확장한 결과물입니다.

Dice & Castle

학부 2학년 · C++ 수업 제공 프레임워크 · CJJ 4인 팀



리드 디자이너이자 전투 시스템의 주 구현자로 턴·공격·스킬·승패와 튜토리얼을 연결했습니다. 주사위로 보드를 이동하며 만나는 전투에서, 플레이어와 적이 순서대로 행동하도록 구현했습니다.

같은 행동이 반복 처리되지 않도록 버튼 입력을 전투 상태에 연결했습니다. 커진 전투 로직에서 불필요한 코드를 정리하고 설명을 보강했습니다. 수업 제공 프레임워크에서 Battle과 튜토리얼을 맡았고, 보드 시스템은 팀원이 담당했습니다.

Way Back Home의 최종 게임 화면과 Dice & Castle 제출 영상의 전투 장면입니다. 게임 전체와 아트는 팀 공동 결과물입니다.

소프트웨어 프로젝트

일상의 사용 흐름을 앱으로 구현하고, 데이터 검증과 복구 기능을 연결했습니다.

FinOps Guard

2026.07-08 · Next.js / Gemini / Firestore

팀 프로토타입으로 AI 응답 검증부터 정책 판정·감사 이력·테스트 결제까지 연결했습니다. JSON·Zod 검사와 예산·한도·중복 정책을 거쳐 Solana Devnet에서 정산합니다.

Personal Knowledge OS

2026.04-07 · Electron / React / SQLite

노트·문서·링크를 폴더와 태그로 정리하고 통합 검색하도록 구현했습니다. 자료 간 연결을 지원하고 JSON 가져오기에서 ID·소유권·첨부 경로를 검사합니다.

Personal Planner OS

2026.04-07 · Electron / TypeScript / SQLite

오늘 할 일·달력·시간 블록과 루틴·일기·회고를 한 앱에 연결했습니다. 작업·일정을 편집하고 휴지통에서 복원하며, JSON 가져오기 전 내용을 미리 확인할 수 있습니다.

Career & Capital OS

2026.04-07 · Electron / React / SQLite

지원 회사·직무·전형 이력과 다음 행동을 관리하도록 구현했습니다. 제출 자료·면접 준비·작업·일정을 연결하고 학습 기록과 재무 거래도 함께 관리합니다.

Piano Practice OS

2026.05-07 · Android / Kotlin / Compose

MIDI·MusicXML 악보를 88키 안내와 손별 연습에 연결했습니다. 템포 조절·구간 반복·합성 미리듣기, PDF 보기·곡 보관과 마지막 연습 위치 저장을 제공합니다.

개인 앱은 필요와 사용 흐름을 정한 뒤 AI와 구현·검증을 반복했습니다. FinOps Guard의 처리 흐름은 팀 공동 결과물입니다.

설계와 데이터베이스 실습

변경 영향을 문서로 정리하고, 실행계획과 재현 실습으로 데이터베이스 동작을 확인했습니다.

TraceAX

2026.08 · SKALA 개인 설계 프로젝트 · OpenAPI / DBML / Python

요구사항이 바뀔 때 함께 확인할 화면·API·데이터·테스트를 연결하는 절차를 설계했습니다. 누락 확인부터 변경 등록·영향 검토·승인까지 흐름을 정의하고, AI 제안과 담당자의 최종 판단을 구분했습니다.

요구사항정의서·화면설계서·데이터 모델·API 명세를 작성했습니다. 17개 기능, 8개 업무 화면과 공통 메뉴, 15개 테이블·19개 API 연산을 정의한 설계 프로젝트입니다.

PostgreSQL Lab

2026 · SKALA 개인 실습 · SQL / EXPLAIN ANALYZE

인덱스를 추가해도 쿼리가 느려질 수 있음을 실행계획으로 확인했습니다. 1천만 행의 테스트 데이터에서 단일·복합·커버링 인덱스를 비교하며, 조회 조건과 컬럼 구성에 따라 접근 방식이 달라지는 이유를 살펴봤습니다.

날짜 조건을 범위 조건으로 바꾸고 OFFSET과 keyset 페이지네이션을 비교했습니다. READ COMMITTED·REPEATABLE READ의 차이와 데드락을 재현하고, 잠금 순서를 맞추는 회피 원리를 정리했습니다.

AI와 구현하고 검증하는 방식

요구사항·규칙·계획을 Markdown으로 정리해 AI와 진행할 작업의 범위를 나눴습니다. GitHub PR과 AI의 코드 리뷰·빌드·타입 검사·기능 점검 결과를 검토하고, 필요한 수정을 요청했습니다.

화면의 동작 조건과 데이터의 연결을 먼저 정리하고, 생성된 코드를 실제 실행 결과와 대조합니다. 게임 저장에서도 요청·응답과 데이터베이스의 값이 같은 의미를 유지하는지 확인했습니다.

GitHub 프로필

C++로 익힌 시스템 구현

DigiPen 수업 프레임워크를 바탕으로 작성한 개인 과제와 공동 기술 데모

경로 탐색과 공간 구조 | 2025 · CS380 / CS350

A*에 휴리스틱과 코너 이동 조건, 경로 단순화·보간을 적용했습니다. 기하 과제에서는 교차 판정, Dynamic AABB Tree, BSP, GJK를 구현하며 충돌 후보를 찾는 과정을 익혔습니다.

동기화와 병렬 처리 | 2025 · CS355

pthread·mutex·condition variable로 작업을 동기화하고, worker 기반 병렬 정렬과 CAS 자료구조를 구현했습니다. 공동 과제에서는 연결 리스트와 관련 테스트를 맡았습니다.

오류를 추적하는 도구 | 2025 · CS315

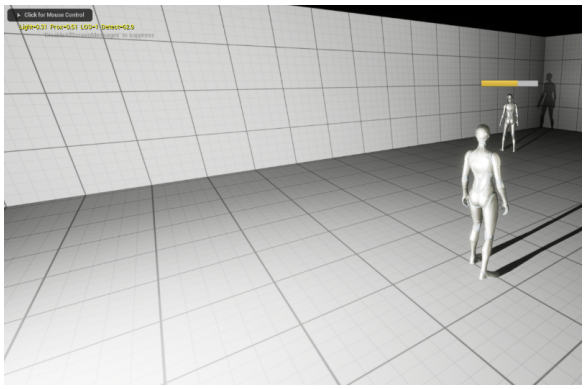
SEH·MiniDump·DbgHelp로 크래시를 기록하고 메모리 할당·해제를 추적했습니다. 실행 위치를 샘플링해 심볼로 해석하고 CSV에 기록하는 프로파일러 프로토타입도 작성했습니다.

렌더링·네트워크·알고리즘 | 2024 · CS300 / CS260 / CS330

OpenGL의 G-buffer·지연 렌더링·노멀 매핑·반사·굴절을 구현했습니다. WinSock TCP·UDP 통신, 자원 제한이 있는 그래프 탐색과 최근접 점 쌍의 분할 정복도 다뤘습니다.

Light Awareness Demo

2025.11 · Unreal C++ · Jin Park / Minki Cho 공동 개발



광원·거리·시야 조건을 적의 감지에 반영하는 잠입 기술 데모를 공동 개발했습니다. 디버그 UI의 Light·Proximity·LOS·Detection 수치와 감지 게이지·행동 변화를 함께 확인할 수 있도록 구성했습니다.

공동 보고서의 실제 테스트 화면입니다. Light Awareness Demo는 공동 개발이며, 위 수업 과제는 제공 프레임워크를 바탕으로 한 구현입니다.