BuildTest (차량 게임 역컴파일 분석) — 컨셉기획서

> 재구성 기획서 (역설계) / 작성 기준 시점: 2026-07-18 (프로젝트 시작 시점) / 문서 버전: Ver 0.1 프로젝트 ID: P041 / 근거 수준: C / 작성: Claude (재구성) 이 문서는 결과물과 제작 기록으로부터 기획 의도를 역추정해 당시 작성한 것처럼 재구성한 것이다. 확인된 사실과 추정을 [근거]/[추정] 태그로 구분한다.

1. 한 줄 컨셉과 기획 의도

한 줄 컨셉: 소스 없이 남은 Unity 차량 추격 게임 빌드를 역컴파일해, 규칙과 구조를 문서로 복원하는 분석 작업. [추정: decompiled_src와 Antigravity 대화 제목 "Analyzing Unity Game Structure"]

기획 의도. - 빌드(exe)만 있고 프로젝트 원본이 없으므로, 코드에서 게임의 규칙(조작/적 AI/승패)을 읽어 재사용 가능한 설계 지식으로 만든다. [추정: 원본 프로젝트 부재] - 분석 대상 게임은 "아이템을 모두 모아 골에 도달하되, 경찰차 같은 추격자를 피한다"는 차량 탈출형이다. [근거: `ItemCollector.cs`, `Goal.cs`, `EnemyChaser.cs`] - 분석 결과는 이후 유사 프로토타입 제작이나 다른 프로젝트와의 비교(P013 시티러쉬 등)의 기준 자료로 쓴다. [추정: raw/codex1.md가 SpaceMD_Proj와의 클래스 중복을 언급]

2. 타깃, 플랫폼, 엔진, 범위, 인원, 도구

- 플랫폼/엔진: PC(Windows) x64, Unity Mono 빌드(정확한 에디터 버전 미확인). 렌더 파이프라인은 URP 패키지가 포함되어 있다. [근거: `ScriptingAssemblies.json`, `My project_PLS2.exe`]
- 범위: C# 소스 17개(핵심 게임 클래스 약 14개, 템플릿 잔재 1~2개). 씬/에셋 내부 값 분석은 범위 밖. [근거: Glob 결과]
- 인원/도구: Antigravity(AI) 단독 분석. 역컴파일러 이름 미확인. [근거: raw/codex1.md]
- 규모: L(소규모 분석 과제).

3. 핵심 재미와 차별점

- 분석 대상의 플레이어 판타지: 차량으로 추격자를 따돌리며 맵의 아이템을 모아 탈출한다. [근거: `CarController.cs`, `EnemyChaser.cs`, `Goal.cs`]
- 대상 게임의 설계 특징: 충돌 직후 3.5초 추격 중단, 1초 무적, R키 리스폰 후 3초 유령 모드로, 억울한 연속 피격을 막는다. [근거: `EnemyChaser.cs`, `PlayerHP.cs`, `CarController.cs`]
- 분석 작업의 차별점: 파라미터가 아닌 "규칙 구조"를 중심으로 요약한다. [추정: 인스펙터 값이 소스에 없음]
- 레퍼런스: 기록 없음.

4. 세계관, 톤, 아트 방향

- 세계관: 기록 없음. 사이렌 오디오와 `EnemyChaser` 명칭에서 경찰 추격 상황으로 추정. [추정: sirenSound 필드]
- 톤: 아케이드형 레이싱/탈출. 속도계(km/h) 표시와 엔진음 피치 연동이 있다. [근거: `Speedometer.cs`, `CarController.cs`]
- 아트: 에셋 내부 미확인. 해당 없음(분석 범위 밖).

5. 게임 흐름 개요

- 흐름: 메인 메뉴 → `TutorialScene`(페이지 넘김) → `RaceScene`(본 플레이) → 게임오버 또는 클리어 패널. [근거: `MainMenu.cs`, `TutorialManager.cs`, `GameManager.cs`, `Goal.cs`]
- 코어 루프: 운전 → 아이템 수집(`n / 총수` 표시) → 전부 모으면 골 활성화 → 골 진입 시 클리어. 추격자 접촉 시 HP 10 감소. [근거: `ItemCollector.cs`, `Goal.cs`, `EnemyChaser.cs`]
- 실패: HP 0 → 게임오버, 시간 정지. 재시작은 현재 씬 재로드. [근거: `PlayerHP.cs`, `GameManager.cs`]

6. 제작 방향 (파트별 설계 블록)

시스템 파트 — 분석 대상 규칙 복원

- 의도(Why): 규칙은 코드에 집약되어 있으므로 시스템 요약이 가장 가치가 높다.
- 설계(How): 클래스별로 입력→상태→피드백을 표로 정리하고(02 문서), 클래스 간 호출 관계를 그린다. 씬 수치가 필요한 항목은 "미확인"으로 둔다.
- 필요(What): `decompiled_src` 17개 파일.
- 작업 방법: 소스 전수 읽기 → 클래스 역할 분류 → 호출 관계 정리. 담당 AI, 산출물 문서.
- 완료 기준: 모든 클래스가 02 문서의 어느 절에 대응되는지 표로 연결된다.
- 리스크/대안: 컴파일러가 만든 이름이 바뀐 로컬 변수는 의미 추정이 필요하다. 필드명과 문맥으로 보완한다.
- 근거: [근거: decompiled_src/*.cs] [추정: 분석 방식은 코드 구조에서 역산]

콘텐츠/UI 파트 — 씬 구성 추정

- 의도(Why): 씬 구성은 규칙의 실제 사용처를 보여 준다.
- 설계(How): 소스의 씬 이름과 `level0~2` 대응을 비교한다. 바이너리 씬은 열지 않고 소스 참조로만 추정한다.
- 필요(What): `level0~2`, 소스의 `LoadScene` 호출.
- 작업 방법: Grep으로 씬 이름 수집.
- 완료 기준: 씬 3개의 역할 가설이 표로 있다(가설 표기).
- 리스크/대안: 씬 순서는 빌드 설정을 확인하기 전까지 확정 불가. 에디터 도구 사용 시 확정.
- 근거: [근거: `MainMenu.cs`, `TutorialManager.cs`] [추정: level0~2 순서]

아트, 사운드, 데이터 파트

- 의도(Why): 사운드가 피드백 설계의 큰 부분이라 파라미터를 분리해 기록한다.
- 설계(How): 엔진 피치 범위(0.8~2), 사이렌 거리 50 등 소스 직접값만 표로 확정한다.
- 필요(What): `CarController.cs`, `EnemyChaser.cs`.
- 작업 방법: 상수 추출.
- 완료 기준: 03 문서 표 완성.
- 리스크/대안: 인스펙터 값은 `[SerializeField]`라 소스에 없다. AssetRipper 등으로 씬 직렬화 읽기가 대안이다(도구 사용 여부 미확인).
- 근거: [근거: `CarController.cs`, `EnemyChaser.cs`]

7. 제작 단계(마일스톤)

단계	목표	산출물
M1	역컴파일과 파일 목록 확보	`decompiled_src` (완료) [근거: Glob]
M2	클래스 역할/흐름 분석	분석 대화 100단계 [근거: raw/codex1.md]
M3	규칙/수치 문서화	본 재구성 문서(이번 작업)
M4	씬 직렬화 값 복원	미착수(기록 없음)

8. 리스크와 결정이 필요한 사항

- 빌드 소유권/원저작자 불명. 분석물을 외부에 쓰기 전 확인 필요. [추정: 제작자 미확인]
- 인스펙터 값 복원 여부와 도구 결정 필요.
- `PlayerMovement`, `DoorController`, `KeyController`는 도보/탐험 템플릿 코드로 보이며 본 게임에서 활성인지 확인 필요. [추정: 차량 게임 흐름에서 호출되지 않음]