BuildTest (차량 게임 역컴파일 분석) — 메인기획서

> 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. 핵심 메카닉스

분석 대상 제품(차량 추격/탈출 게임)의 메카닉스를 소스에서 읽은 대로 서술한다.

차량 주행

- 의도(Why): 추격전의 쾌감은 차량 조작감에서 나오므로 물리 기반 WheelCollider로 구현한다. [추정: WheelCollider 4개 사용]
- 설계(How): `Input.GetAxis("Vertical")`×`motorForce`를 앞바퀴 2개 `motorTorque`에 적용(전륜구동). `Horizontal`×`maxSteerAngle`을 앞바퀴 `steerAngle`에 적용. Space 키는 4바퀴 `brakeTorque=breakForce`. 입력이 0.1 미만이면 속도를 매 물리 프레임 0.993배로 감쇠시켜 자연 정지를 돕는다. 바퀴 메시는 `GetWorldPose`로 매 프레임 동기화한다. 속도계는 `linearVelocity.magnitude × 3.6`(km/h). [근거: `CarController.cs`]
- 필요(What): Rigidbody, WheelCollider 4, 바퀴 Transform 4, AudioSource 2.
- 작업 방법: 물리 처리는 FixedUpdate에서 입력→모터→조향→바퀴→사운드 순으로 호출. 검증은 실제 주행.
- 완료 기준: 가속, 조향, 브레이크, 자연 감속이 입력대로 반응한다.
- 리스크/대안: `motorForce`, `breakForce`, `maxSteerAngle`은 인스펙터 값이라 미확인. 후진 시 별도 처리 없음(음수 토크로 후진).
- 근거: [근거: `CarController.cs`]

리스폰과 유령 모드

- 의도(Why): 차량이 뒤집히거나 끼었을 때 탈출 수단을 주고, 추격자에게 바로 맞지 않게 보호한다. [추정: 코드 구조에서 역산]
- 설계(How): R 키 → 등록된 `spawnPoints` 중 현재 위치에서 가장 가까운 지점으로 위치/회전 이동, 선속도/각속도 0. 그 후 `ghostTime`(기본 3초) 동안 자기 콜라이더와 `enemyCars`의 모든 콜라이더 사이 `Physics.IgnoreCollision`을 켠다. 유령 중에는 적이 추적과 사이렌을 멈춘다. [근거: `CarController.cs`, `EnemyChaser.cs`]
- 필요(What): 스폰 포인트 Transform 배열, 적 차량 배열.
- 작업 방법: 코루틴 `GhostMode`.
- 완료 기준: 3초 후 충돌 복구, 적 추적 재개.
- 리스크/대안: 리스폰에 쿨다운/페널티가 없어 남용 가능. 별도 제한은 소스에 없다. 적 배열이 비어 있으면 유령 효과 없음.
- 근거: [근거: `CarController.cs`]

추격 AI

- 의도(Why): 적이 플레이어를 단순 추종하지 않고 예측 사격식 차단을 하게 해 긴장을 만든다. [추정: predictionTime 필드]
- 설계(How): NavMeshAgent는 경로 계산에만 쓰고(`updatePosition=false`, `updateRotation=false`), 실제 이동은 코드가 한다. `updateInterval`(0.1초)마다 목표를 `player.position + 플레이어 속도 × predictionTime`(0.5초)로 갱신한다. 원하는 방향으로 `rotationSpeed`(360도/초)로 회전하고, 각도 차이/90에 비례해 속도를 `cornerSlowDown`(0.5)까지 낮춘다(코너 감속). 속도는 `agent.acceleration`으로 가감속, 최종 `agent.nextPosition`에 위치를 반영한다. NavMesh 밖이면 반경 5 안 가장 가까운 점으로 워프한다. [근거: `EnemyChaser.cs`]
- 필요(What): 베이크된 NavMesh, `Unity.AI.Navigation`, 플레이어 Rigidbody.
- 작업 방법: Update에서 처리.
- 완료 기준: 적이 NavMesh를 따라 플레이어 예상 위치로 향하고 코너에서 감속한다.
- 리스크/대안: `agent.speed`, `acceleration` 값은 미확인. 업데이트 간격이 짧아 다수 적에서 비용이 늘 수 있다.
- 근거: [근거: `EnemyChaser.cs`]

피격, HP, 게임오버

- 의도(Why): 접촉이 즉사가 아니라 누적 피해이므로 추격이 오래 지속되는 긴장을 만든다. [추정: HP 100, 피해 10]
- 설계(How): 적 `OnCollisionEnter`가 Player 태그와 닿으면 충돌음 재생, `PlayerHP.TakeDamage(10)`, 추격 3.5초 정지. `PlayerHP`는 최대 100, 피격 후 1초 무적. HP 0이면 `GameManager.GameOver()`: 패널 표시, 적/차량 사운드 정지, `Time.timeScale=0`. 재시작은 `timeScale=1` 후 현재 씬 재로드. [근거: `EnemyChaser.cs`, `PlayerHP.cs`, `GameManager.cs`]
- 필요(What): `UIManager`의 HP 슬라이더/텍스트, 게임오버 패널.
- 작업 방법: 싱글톤 `GameManager.Instance`, `UIManager.Instance`.
- 완료 기준: 피격 시 HP UI 갱신, 10회 피격 시 게임오버(무적/정지 시간 고려).
- 리스크/대안: 무적 1초가 추격 정지 3.5초보다 짧아 실질 연속 피격은 정지 시간이 결정한다.
- 근거: [근거: `PlayerHP.cs`, `UIManager.cs`]

아이템 수집과 목표

- 의도(Why): 단순 도착이 아니라 맵 탐색을 강제해 추격과 경로 선택 사이의 긴장을 만든다. [추정: 목표 잠금 구조]
- 설계(How): `ItemCollector`가 `totalItemCount`와 현재 수를 관리하고 `n /총수` 텍스트를 갱신한다. 시작 시 `goal` 오브젝트를 비활성화하고, 수집 수가 총수 이상이면 활성화한다. `Item`은 트리거 진입 시 부모의 `ItemCollector.CollectItem()` 호출 후 파괴. `Goal` 진입 시 전부 수집 상태면 클리어 패널 표시/사운드 정지/`timeScale=0`. 수집 미완료 시 로그만 출력. [근거: `ItemCollector.cs`, `Item.cs`, `Goal.cs`]
- 필요(What): 아이템 프리팹과 총수 설정(총수 값 미확인).

- 작업 방법: 아이템은 트리거 콜라이더.
- 완료 기준: 모두 수집 시 골이 나타나고 진입하면 클리어.
- 리스크/대안: 골이 숨겨져 있어 미완료 진입 안내 UI는 없다(Debug.Log 한 줄).
- 근거: [근거: `ItemCollector.cs`, `Goal.cs`]

오디오 피드백

- 의도(Why): 속도와 위협 거리를 소리로 전달해 UI 없이도 상황을 알게 한다. [추정: 사이렌 거리 비례 볼륨]
- 설계(How): 엔진음은 속도 0.5 m/s 초과 시 재생, 피치는 `Lerp(0.8, 2, 속도/30)`. 브레이크음은 Space 입력과 속도 1 초과 동시 충족 시 재생. 사이렌은 적과 거리 `sirenDistance`(50) 이내에서 볼륨을 `Lerp(1, 0, 거리/50)`로 조절. [근거: `CarController.cs`, `EnemyChaser.cs`]
- 필요(What): 엔진, 브레이크, 사이렌, 충돌 오디오 클립(내용 미확인).
- 작업 방법: 기록 없음.
- 완료 기준: 접근할수록 사이렌이 커진다.
- 리스크/대안: 거리 비례가 직선이라 근접 구간 변화가 둔하다는 점은 [추정].
- 근거: [근거: `CarController.cs`, `EnemyChaser.cs`]

도보, 문, 열쇠 모듈

- 의도(Why): 도보 탐험 장면용 모듈로 보인다. 본 게임과의 연결 미확인. [추정: 소스 간 호출 없음]
- 설계(How): `PlayerMovement`는 Rigidbody에 힘을 주고 속도를 `2×runSpeed`로 제한(Shift 시 `runMultiplier=2`), 마우스로 카메라 상하/몸체 좌우 회전. `DoorController`는 상태 Closed/Opened/Jammed, E키 토글, 열쇠 필요 시 `Door_Jam` 애니메이션, `KeyController`가 Player 태그 접촉 시 `gotKey=true`. [근거: `PlayerMovement.cs`, `DoorController.cs`, `KeyController.cs`]
- 필요(What): Animation 클립 `Door_Open`, `Door_Close`, `Door_Jam`.
- 작업 방법: 기록 없음.
- 완료 기준: 해당 없음(사용 여부 확인 후 판단).
- 리스크/대안: `Readme.cs`는 Unity 템플릿 잔재로 게임과 무관. `ObjectRotate`, `ObjectRotate2`는 축 회전 연출용(기본 20도/초). [근거: `Readme.cs`, `ObjectRotate.cs`]
- 근거: [근거: 위 소스]

2. 구현 방법

- 클라이언트 구조: 싱글톤 `GameManager`/`UIManager`, 컴포넌트형 MonoBehaviour. `FindAnyObjectByType`/`FindObjectsByType`로 차량/적을 찾아 사운드를 정지한다. [근거: `GameManager.cs`]
- 입력: 레거시 `Input` 클래스(InputLegacyModule). InputSystem 패키지는 포함되었으나 소스는 쓰지 않는다. [근거: csproj, 소스]
- 씬 흐름: `MainMenu.StartGame` → `TutorialScene` → `TutorialManager.NextPage`가 페이지를 모두 넘기면 `RaceScene`. [근거: `MainMenu.cs`, `TutorialManager.cs`]

- 분석 작업 파이프라인: 빌드 → 역컴파일 → `decompiled_src` + csproj → 읽기/요약. csproj의 참조는 `My project_PLS2_Data\Managed` 안 DLL을 가리킨다. [근거: csproj] [추정: 역컴파일 도구 미확인]

3. 시스템 간 상호작용과 밸런스 기준

- 핵심 변수: 적 수/속도, 아이템 총수, 맵 크기, 플레이어 HP(100), 피해(10), 무적(1초), 추격 정지(3.5초), 유령 시간(3초). [근거: 각 소스]
- 상호작용: 피격은 추격 정지를 불러 접촉 후 거리 확보 시간을 만든다. 리스폰은 유령 모드로 적을 정지시킨다. 아이템 완료는 골 노출을 만든다.
- 밸런스 해석: 피격 10회로 사망, 접촉마다 3.5초 정지라 구조적 난이도는 낮은 편이다. [추정: 수치 비교, 적 수는 미확인]
- 곡선/확률: 기록 없음.

4. 게임의 나머지 부분의 방향

- 콘텐츠 구조: 씬 3개(`level0~2`)로 추정. 레이스 씬 1개가 본 플레이. 스테이지 확장 구조는 소스에 없다. [근거: Glob 결과]
- 메타 진행: 없음. 저장/설정: 소스에 없음(기록 없음).
- UI/UX: HP 슬라이더, 속도계, 아이템 카운터, 게임오버/클리어 패널, 튜토리얼 페이지. [근거: 소스]
- 아트/사운드: 미확인(에셋 분석 범위 밖).

5. 제작 과정 설계

- 순서: 파일 목록 확인 → 소스 읽기 → 클래스 분류 → 흐름도 → 수치 표 → 미확인 목록.
- 병렬: 소스 분석(AI)과 씬/에셋 도구 조사는 병렬 가능. [추정]
- 검증: 소스 라인 근거 대조. 실행 검증은 `My project_PLS2.exe` 실행으로 가능하나 수행 기록 없음.
- 인수인계: 본 폴더 문서 6종이 인수인계 자료.

6. 세부기획서 목록

- `03_세부기획서_분석대상-규칙/데이터명세.md`: 클래스 명세, 파라미터 표, 미확인 값 복원 계획, 검수 기준.