시티러쉬 — 메인기획서

> 재구성 기획서 (역설계) / 작성 기준 시점: 2026-07-08 (프로젝트 시작 시점) / 문서 버전: Ver 0.1 프로젝트 ID: P013 / 근거 수준: B / 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

차량 조작

- 의도(Why): 추격전의 기본 쾌감은 차를 마음대로 몰 수 있는 데서 나온다. 조작은 단순해야 캐주얼 플레이어도 즉시 쓴다.
- 설계(How): 입력 W/S(또는 방향키 상하)=가속/후진, A/D=조향, Space=브레이크, R=리스폰. 전륜 두 바퀴에 `motorTorque = 입력 × motorForce`, `steerAngle = 입력 × maxSteerAngle`을 적용하고, Space 시 4바퀴 `brakeTorque = breakForce`. 씬 값은 motorForce 2000, breakForce 3000, maxSteerAngle 40. 가속 입력 절댓값이 0.1 미만이면 속도를 매 물리 프레임 0.993배로 줄여 자연 감속을 만든다. 대안인 감속 전용 drag 값 대신 속도 곱을 택한 이유는 구현이 간단해서다. [추정] 바퀴 메쉬는 `GetWorldPose`로 동기화. 속도 표시는 `m/s × 3.6`을 km/h로 정수 반올림.
- 필요(What): 차량 프리팹(Rigidbody, WheelCollider 4, 바퀴 메쉬 4), 입력 축 `Horizontal`/`Vertical`.
- 작업 방법: 차량 프리팹 조립 → 수치 튜닝 → 속도계 연결. 검증은 플레이 테스트.
- 완료 기준: 직진, 코너링, 급정거가 자연스럽고 뒤집힘이 드물다.
- 리스크/대안: 질량/서스펜션 미조정 시 불안정. 리스폰으로 보완.
- 근거: [근거: CarController.cs, Speedometer.cs, Proj_TEST.unity]

경찰 추격 AI

- 의도(Why): 플레이어가 계속 움직이도록 만드는 압력원.
- 설계(How): NavMeshAgent 목적지를 매 프레임 `player.position`으로 갱신. NavMesh 위가 아니면 5m 반경에서 `SamplePosition`으로 위치를 보정(`Warp`). 플레이어 `IsGhostMode`이거나 `isPaused`면 `agent.isStopped = true`. 충돌 시 `Player` 태그 확인 → `TakeDamage(10)` → 3초 정지. 단순 추적을 택한 이유는 NavMesh 기능만으로 충분하고 개발 기간이 짧아서다. [추정]
- 필요(What): 베이크된 NavMesh(AI Navigation), 경찰 프리팹(NavMeshAgent/Collider), 태그 `Player`.
- 작업 방법: NavMesh 베이크 → 경찰 배치 → `player` 연결.
- 완료 기준: 경찰이 도로를 따라 플레이어에게 도달.
- 리스크/대안: 경찰이 건물 사이에서 막힘 → 워프 보정. 경찰 대수/속도는 씬 값 미확인.
- 근거: [근거: EnemyChaser.cs, vehicle_system_analysis.md]

체력, 피격, 무적

- 의도(Why): 실패가 너무 빨리 오지 않게 하고 긴장은 유지한다.

- 설계(How): maxHP 100, 피격당 10, 무적 1.0초, HP는 0 아래로 내려가지 않는다(`Mathf.Max`). 피격 후 HP가 남으면 무적 코루틴, 0이면 `GameManager.GameOver()`. 경찰 3초 정지가 있어 이론상 연속 10회 충돌로 게임오버. 3초 정지 + 1초 무적이 겹치도록 설계한 이유는 두 번째 안전장치를 두려는 것이다. [추정]
- 필요(What): `PlayerHP`, `UIManager`, `GameManager` 싱글톤.
- 작업 방법: 컴포넌트 부착 → UI 슬라이더 연결.
- 완료 기준: 피격 시 HP 바 감소, 0에서 패널 표시.
- 리스크/대안: 경찰이 여러 대면 정지가 각 개체 단위여서 합산 피격이 빠르다. 전역 무적으로 일부 보완됨.
- 근거: [근거: PlayerHP.cs, EnemyChaser.cs]

리스폰과 고스트 모드

- 의도(Why): 끼임, 전복, 포위 상태에서 플레이어를 구제한다.
- 설계(How): R키 → 모든 `spawnPoints` 중 거리 최단 지점 선택 → 속도/회전 속도 0 → 위치/회전 이동 → 3초(`ghostTime`) 동안 플레이어 콜라이더와 `enemyCars` 콜라이더 간 `Physics.IgnoreCollision` + 경찰 정지. 쿨다운/횟수 제한 없음. 가까운 지점을 택한 이유는 진행 상황을 되돌리지 않기 위해서다. [추정]
- 필요(What): 맵에 `spawnPoints` 배치, 경찰 배열 등록.
- 작업 방법: 스폰 포인트를 도로 구간마다 배치.
- 완료 기준: R 후 3초간 경찰이 접촉 못 한다.
- 리스크/대안: 횟수 제한이 없어 남용 가능 → 쿨다운 [제안값 10초].
- 근거: [근거: CarController.cs, vehicle_system_analysis.md]

수집과 탈출

- 의도(Why): 열쇠 수집은 이동 목적, 탈출은 마무리 목표.
- 설계(How): 아이템이 플레이어 트리거에 닿으면 `ItemCollector.CollectItem()` → 수집 수+1, UI `n / 전체` → 전부 모으면 `goal.SetActive(true)`. 탈출구 트리거 진입 시 `IsAllCollected()` 재확인 → 클리어 패널/차량 소리 정지/`timeScale = 0`. 이중 검사(활성화+진입 시 검사)는 안전을 위한 중복이다. [추정]
- 필요(What): 아이템 프리팹(Trigger), 탈출구 오브젝트(Trigger, 초기 비활성).
- 작업 방법: 아이템 배치 후 `totalItemCount`를 수동 입력.
- 완료 기준: 아이템 수와 `totalItemCount`가 일치하고 탈출구가 열린다.
- 리스크/대안: 수동 입력이라 배치 수와 어긋나면 탈출구가 안 열림 → 씬 시작 시 자동 집계 [제안값].
- 근거: [근거: ItemCollector.cs, Item.cs, Goal.cs]

2. 구현 방법

- 클라이언트 구조: 매니저 2종(`GameManager`, `UIManager`)은 정적 `Instance` 싱글톤, 나머지는 컴포넌트 간 직접 참조. [근거: GameManager.cs, UIManager.cs]
- 상태 머신: 게임 전체 상태는 별도 enum 없이 `Time.timeScale`(0/1)과 패널 활성 여부로 표현. 엔진음만 Idle→Starting→Looping→Ending 상태 머신. [근거: GameManager.cs, vehicle_system_analysis.md]

- 시스템 연결: 경찰→`PlayerHP`→`UIManager`/`GameManager`, 아이템→`ItemCollector`→탈출구, `CarController.IsGhostMode`→`EnemyChaser`.
- 데이터: 모든 수치는 인스펙터 필드. 별도 파이프라인 없음.
- 도구: 맵 자동 생성 `MDMapCreator`(스카이, 소형, 상점 건물, 잔디/도로 타일 사용, 카메라 orthographic 배경색 RGB 44/110/120), 정렬 `NewMapSceneSorter`(카테고리 부모 5개). [근거: MDMapCreator.cs, NewMapSceneSorter.cs]
- 입력 처리는 구 `Input` API 사용. 신규 Input System 패키지는 설치만 됨. [근거: CarController.cs, 빌드 Managed 목록]

3. 시스템 간 상호작용과 밸런스 기준

변수	값	영향	근거
maxHP	100	허용 피격 10회	PlayerHP.cs
피격 데미지	10	위와 동일	EnemyChaser.cs
피격 무적	1.0초	연속 피격 방지	PlayerHP.cs
경찰 정지	3초	도주 구간 길이	EnemyChaser.cs
고스트 시간	3초	리스폰 후 안전	CarController.cs
motorForce / breakForce / maxSteerAngle	2000 / 3000 / 40	가속, 정지, 선회	Proj_TEST.unity
엔진 피치	0.8~2.0, 30m/s 기준	속도 피드백	CarController.cs
- 밸런스 축: 경찰 속도 대비 차량 최고속도(경찰 속도는 미확인), 아이템 총수와 맵 크기. 기준값 기록 없음.			
- 난이도 곡선: 단일 스테이지, 난이도 단계 기록 없음.			

4. 게임의 나머지 부분의 방향

- 콘텐츠 구조: 단일 도시 맵. 씬 4개(`SampleScene`, `TEST`, `Proj_TEST`, `NewMap`)와 빌드 3개 씬. 최종 구성 미확인. [근거: Scenes 폴더, level0~2]
- 메타 진행/저장: 없음. 기록 없음.
- UI/UX: HP, 수집, 속도 HUD, 클리어/게임오버 패널. 메인 메뉴 여부 미확인.
- 아트/사운드: 03 세부기획서 참조.
- 확장 방향(제안): 경찰 다종화, 난이도 곡선, 시간 제한 [제안값 없음].

5. 제작 과정 설계

- 순서: 프로젝트/렌더 설정 → 차량 → 경찰 AI/HP → 수집/탈출 → 맵 → UI/사운드 → 포스트 프로세싱 → 빌드. [추정: 커밋 순서와 의존 관계]

- 병렬 가능: 맵 제작(`MDMapCreator`)과 차량/AI 구현, 사운드 클립과 UI. 3인 분담 기록 없음.
- AI/도구 분담: 코드 분석 문서와 배치 스크립트는 AI 도구가 맡은 정황. [추정: batch_*.log, analysis_results.md]
- 검증: 테스트 씬(`TEST`, `Proj_TEST`)에서 플레이. 자동 테스트 없음.
- 인수인계: 분석 문서 2종이 사실상 인수인계 자료. HANDOFF는 기록 없음.

6. 세부기획서 목록

문서	정할 것
03_세부기획서_차량추격경험	차량 튜닝값, 경찰 AI, 피격/리스폰 규칙
03_세부기획서_레벨맵구조	도시 맵 생성, 아이템/스폰 배치, NavMesh
03_세부기획서_UIUX사운드	HUD, 패널, 엔진/브레이크음, 포스트 프로세싱
03_세부기획서_데이터리소스	인스펙터 수치 명세, 에셋 목록