탈출 성 — 메인기획서

> 재구성 기획서 (역설계) / 작성 기준 시점: 2026-06-05 (프로젝트 시작 시점) / 문서 버전: Ver 0.1 프로젝트 ID: P014 / 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.

용어: **런(전역) 표** = 스테이지를 넘어 공유되는 표. **스테이지 표** = 한 보드의 배치/스크립트 표. **효과 그룹** = 같은 `group_id`를 `seq` 순으로 실행하는 효과 묶음. **칸 타입** = `tiles.type` (예 `1f_trap`).

1. 핵심 메카닉스

턴 진행과 이동

- 의도(Why): 뱀과 사다리처럼 한 번의 입력(주사위)으로 한 턴이 끝나는 단순한 리듬을 만들고, 판단은 갈림길과 이벤트 선택지로 옮긴다.
- 설계(How): 주사위 표기는 `config.dice`(`d6`, `2d6`, `d20`, `3d4+2`), 없으면 `dice_sides`, 그것도 없으면 d6. 이동은 한 칸씩 진행하며 갈림길(나가는 edge가 둘 이상)에서는 선택 팝업이 뜨고, `tiles.stop=1`인 칸은 남은 이동을 접고 멈춘다. 이동 선택 이벤트 `e_move_choice`는 "주사위 {roll}이 나왔습니다. 이동하시겠습니까?"로 이동/정지를 고르게 하며, 정지는 `moveDenied` 플래그를 세운다. 대안인 자동 이동은 선택 요소가 없어 택하지 않았다.
- 필요(What): `Dice.ts`, `Board.ts`, `Rng.ts`(판 시드), edges/tiles 표.
- 작업 방법: 주사위 파서 → 그래프 이동 → 갈림길 UI → stop 처리. 검증은 `data:validate`(도달 불가 칸, 막다른 칸, 조건부 출구만 있는 갈림길).
- 완료 기준: 어떤 그래프(고리, 십자, 나무)에서도 이동이 막히지 않는다.
- 리스크/대안: 현재 `dice_sides=100`은 25칸 보드에서 한 번에 보드를 넘기므로 d6 또는 `2d6`으로 되돌린다. [제안값]
- 근거: [근거: README.md, docs/AUTHORING.md, config.csv, choices.csv]

칸 생성과 이벤트 추첨

- 의도(Why): 같은 맵이라도 판마다 다른 경험을 주고, 설계자가 고정 칸과 랜덤 칸을 섞을 수 있게 한다.
- 설계(How): "타입"과 "도착 이벤트"를 독립된 두 축으로 둔다. 타입이 빈 칸은 `region`의 `generation` 가중치로 뽑고, 이벤트가 빈 칸은 `tile_events`에서 `tile_type`별 풀을 `weight`와 `cond_group`으로 뽑는다. `locked=1`이면 보존한다. 난수는 판 시드로 만들고 세이브에도 저장해 이어하기에서 같은 보드를 복원한다. 한 가지 축만 고정하는 4가지 조합이 모두 가능한 것이 대안(타입 고정만 지원)보다 낫다고 판단했다.
- 필요(What): generation.csv, tile_events.csv, BoardGenerator.ts, TileEventResolver.ts, SaveManager.ts.
- 작업 방법: 가중치 설계 → 풀 작성 → 시드 재현 테스트.
- 완료 기준: 같은 시드에서 같은 보드, 빈 시드에서 매번 다른 보드.
- 리스크/대안: `default` 타입(비중 35)에 `tile_events` 항목이 없어 아무 일도 안 일어난다. 의도적 휴식 칸인지 누락인지 미확인. [근거: generation.csv, tile_events.csv]
- 근거: [근거: README.md, docs/AUTHORING.md, generation.csv, tile_events.csv]

이벤트, 선택지, 효과, 조건

- 의도(Why): 모든 사건을 "조건 → 이벤트 → 선택지 → 효과" 네 표의 조합으로 표현해 코드 없이 사건을 만든다.
- 설계(How): 이벤트 `type`은 EFFECT/CHOICE/MOVE/NONE. EFFECT는 `payload`(효과 그룹)를 즉시 실행, CHOICE는 `choices`를 띄워 선택 효과를 실행하고 `next_event`로 연쇄한다. 효과 `kind`는 GAIN/LOSE/SET/MOVE_TO/MOVE_DELTA/SKIP_TURN/SET_FLAG/MSG/RUN_EVENT/APPLY_STATUS/CLEAR_STATUS/DEFEAT_ENEMY/STUN_ENEMY/PL 조건은 한 행이 술어 하나이고 같은 그룹은 AND, 비교어는 min/max/over/under/is/not, 대상은 스탯 id, `flag.x`, `status.x`, `item.x`. 본문은 `{roll}` `{hp}` `{flag.x}` 등으로 치환한다. OR을 지원하지 않는 대신 플래그 우회로 단순성을 지킨다.
- 필요(What): EventEngine.ts, Condition.ts, effects.ts, events/choices/effects/conditions 표.
- 작업 방법: 효과 → 조건 → 이벤트 → 선택지 순으로 id를 만들고 참조 정합성을 검증기로 확인.
- 완료 기준: 참조 끊김 0, 모든 CHOICE에 선택 가능한 선택지가 하나 이상.
- 리스크/대안: 비활성 조건 때문에 모든 선택지가 막히는 교착 가능. 항상 열린 선택지 하나를 두는 규칙을 둔다.
- 근거: [근거: README.md, docs/AUTHORING.md, docs/UI-EXTENSION-PLAN.md, events.csv, choices.csv, effects.csv, conditions.csv]

자원, 상태, 파생, 임계

- 의도(Why): HP, 골드, 횃불 같은 수치와 독/실드 같은 지속 상태를 한 표에 모아 어휘를 통일하고 스테이지 간 이월을 쉽게 한다.
- 설계(How): `stats.csv`의 `kind`가 number/bool/status. `decay`는 턴마다 자동 증감(기력 −1), status는 `duration`, `max_stacks`, `per_turn_target/amount`로 자족(재생 hp+1, 저주 hp−1, 각 3턴). `derived.csv`는 선형결합으로 자동 계산(eval 없음). `thresholds.csv`는 선을 넘는 순간 once/repeat로 이벤트/효과를 발동. 현재 `escape`에서 쓰는 수치: HP 0~20 시작 20, 골드 0~999 시작 5, 횃불 0~15 시작 5, 기력 0~5 시작 5(턴당 −1), 명성 0~10, 보호막(status, 1턴), 열쇠 `it_Key`(bool 0~3). [근거: stats.csv]
- 필요(What): Player.ts, stats/derived/thresholds 표.
- 작업 방법: 자원 목록 확정 → 표시 방식(bar/counter/flag/hidden) → 상태 효과.
- 완료 기준: 범위 밖 값이 잘리고(min/max) HUD에 규칙대로 표시.
- 리스크/대안: 기력, 횃불, 마나, 공격력, 위협 스탯이 콘텐츠에서 쓰이는지 불명(이벤트/효과에 참조 없음). 미사용 스탯은 정리하거나 용도를 정한다. [근거: stats.csv, effects.csv]
- 근거: [근거: docs/GUIDE.md, docs/AUTHORING.md, stats.csv]

적과 추적 AI

- 의도(Why): 순수 운 게임에 시간 압박을 주어 "빨리 열쇠를 모아야 한다"는 긴장을 만든다.
- 설계(How): 로직은 `registerBehavior`로 코드에, 배치/수치는 `enemies.csv`에 둔다. 플레이어 턴 뒤 적을 하나씩 깨워 기절 검사 → `cond_group` 활성 검사 → 행동 함수를 `speed`번 호출한다. 함수는 `{move} {attack} {}` 중 하나만 답한다. 기본 5종: chase(BFS 접근), wander, ambush(시야에 들면 영구 활성), turret, ranged. 접촉 해소는 `contact`(move/pass/both)에 따라 `event` 실행 또는 `damage`만큼 HP 피해. `state`는 적별 상태 객체지만 세이브에는 `tileId/stun/alive/activated`만 저장된다. 현재 추적자 `pursuer`는 시작 `t_00`, speed 2, damage 5, vision 99, `spawn_turn=3`, `collapse_trail=1`, 이벤트 `e_spawn_pursuer`("어둠 속에서 섬뜩한 기운이 따라붙기 시작한다...")로 등장. 아이템 EMP로 1턴 기절시킬 수 있다.
- 필요(What): behaviors.ts, enemies.csv, 이벤트/아이템.
- 작업 방법: 행동 → 접촉 이벤트 → 밸런스(속도, 피해) → 세이브 호환 확인.
- 완료 기준: 추적자가 도달 불가/교착 없이 접근하고, 기절/세이브 복원이 맞다.
- 리스크/대안: HP 최대 20에 피해 5, speed 2, vision 99는 매우 가혹하다. 붕대, 실드, EMP 수급과 함께 재조정이 필요하다. `pursuer` 행동명이 기본 5종에 없어 등록 여부 미확인. [근거: enemies.csv, docs/AI-BEHAVIORS.md]
- 근거: [근거: docs/AI-BEHAVIORS.md, enemies.csv, events.csv, items.csv]

아이템과 열쇠 탈출

- 의도(Why): 우연히 받은 자원을 "쓸 타이밍"으로 바꾸고, 열쇠 3개라는 명확한 승리 조건을 준다.
- 설계(How): `items.csv`: 붕대(HP+3, 소모, 최대 3, HP≤4일 때만 사용), 실드(보호막 1턴, 최대 2), EMP(접촉한 적 1턴 기절, 최대 1), 열쇠(비소모, 최대 3). 목표 칸 `t_24`의 고정 이벤트 `e_door_choice`는 "굳게 닫힌 문"이며 `item.it_key min 3`을 만족할 때만 "열쇠 3개를 사용해 문을 열고 탈출한다"가 활성, 아니면 "돌아간다"(1칸 후퇴). 열쇠 지급은 `e_random_item_04`로 설계했다(풀 연결은 미완, 아래 리스크 참조).
- 필요(What): items/effects/conditions, 이벤트.
- 작업 방법: 아이템 정의 → 획득 경로 → 사용 조건 → 문 이벤트.
- 완료 기준: 열쇠 3개 모은 뒤 문 선택지가 활성화되고 승리로 이어진다.
- 리스크/대안: `ef_door_open`이 `SET_FLAG it_key 3`이라 열쇠를 소모하지 않고 승리 연결(엔딩)도 비어 있다. 열쇠 지급 이벤트 `e_random_item_04`는 `tile_events.csv`의 어떤 풀에도 연결돼 있지 않고(아이템 이벤트 중 `e_random_item_02`만 `test_hp` 칸 `t_02`에 연결), 따라서 현재 데이터로는 열쇠 3개를 얻을 경로가 없다. [근거: effects.csv, tile_events.csv, endings.csv]
- 근거: [근거: items.csv, choices.csv, conditions.csv, effects.csv, events.csv]

엔딩/VN 연출

- 의도(Why): 결과를 자원/플래그로 나누어 다양한 결말을 제공하고, 핵심 장면은 비주얼 노벨 화면으로 몰입시킨다.
- 설계(How): `endings.csv`는 `when`(victory/defeat/any)과 `cond_group`, `priority`로 가장 우선하는 조건 일치 결말을 고른다(구조). 이벤트의 `background/speaker/sprite` 중 하나라도 채우면 화면 약 90%를 덮는 VN 오버레이가 뜨고 좌측 배경+캐릭터, 우측 상단 대사창, 아래 선택지로 구성된다. 캐릭터는 `assets.csv`의 `kind=sprite` 행(`speaker/expr/pos`)이며 이벤트당 1명이다. 타자기 연출은 UI 확장 M에서 구현됐다.
- 필요(What): VNOverlay.ts, endings.csv, vn_* 이미지.
- 작업 방법: 엔딩 조건 설계 → 문구 → 이미지 → 연결.
- 완료 기준: 승리/패배 각각 최소 1개, 조건이 겹쳐도 하나만 선택.
- 리스크/대안: 현재 `endings.csv`가 비어 있어 결과 화면이 기본 처리로 추정. 엔딩 이미지 4종(`img_ending_001~004`)이 이미 있어 연결 후보. [근거: endings.csv, 이미지 파일 목록]
- 근거: [근거: README.md, docs/AUTHORING.md, docs/UI-EXTENSION-PLAN.md]

2. 구현 방법

- 구조: 씬 5종(Menu, StageSelect, Game, Result, Meta)을 `SceneManager`가 전환하고, 게임 씬이 엔진(Board, EventEngine, Player, 적 처리)과 `CanvasRenderer`, DOM UI(Hud, Popup, VNOverlay)를 묶는다. [근거: src 파일 목록]
- 데이터: `DataSource` 인터페이스를 `CsvSource`가 구현해 `loadGameData`가 전역 1회 캐시 + 스테이지 병합. 표는 문자열 id를 `Map`으로 인덱싱해 푼다. 구글 시트 등으로 바꿀 때는 `DataSource`만 교체한다. [근거: README.md, docs/ERD.md]
- 상태 머신: 게임 씬이 턴 단계를 순서대로 진행(주사위 → 이동 → 이벤트 → 효과 → 적 → 턴 종료 훅). [추정: README.md의 한 턴 흐름 서술을 단계로 정리]
- 핵심 알고리즘: 가중치 추첨(생성/이벤트), 무방향 BFS(적 `nextHop/distance`), 선형결합 파생 스탯, 조건 AND 평가. [근거: docs/AI-BEHAVIORS.md, docs/GUIDE.md]
- 저장: `SaveManager`가 멀티 슬롯 세이브, 판 시드, 적 상태를 저장. [근거: .git/logs/HEAD, docs/AI-BEHAVIORS.md]
- 다국어: `data:strings`로 ko 추출 → 번역 채움 → `strings_<lang>.json` 빌드, `ui.csv`의 `lang`으로 선택, 미번역은 한국어로 폴백. `strings_en.json` 존재. [근거: README.md, 파일 목록]
- 디버그: 주소에 `?debug`를 붙이면 `window.__debug`로 접근. [근거: README.md]

3. 시스템 간 상호작용과 밸런스 기준

- 핵심 변수: HP(20), 주사위 면수, 칸 타입 비율, 함정 피해(1/1/2/3), 회복(+1/+2), 후퇴/전진 이동(−3~+3), 추적자 속도/피해, 열쇠 3개.
- 함정 풀: 가중치 20/25/20/15에 피해 1, 1, 2, 3 → 칸당 기대 피해는 (20×1+25×1+20×2+15×3)/80 = 1.5 [근거: tile_events.csv 계산]. 층당 함정 비중 15%.
- 랜덤 칸 풀: 층마다 회복 2 + 스토리 1 + 이동 6 항목이 가중치 11로 같아, 이동 이벤트가 약 2/3를 차지한다. [근거: tile_events.csv 계산]
- 칸 타입 비율은 모든 층 동일(35/15/30/15). 층 상승에 따른 난이도 곡선은 아직 없다. 층별 비율/피해 증가는 [제안값]으로 `generation/tile_events` 값을 조정해 만든다.

- 압박 구조: 추적자(3턴 후 등장) ↔ 열쇠 수집 속도 ↔ 회복/실드 아이템 수급이 한 축. 기력의 턴당 −1 감소는 시간 압박 장치로 보이나 콘텐츠 연결 미확인.

4. 게임의 나머지 부분의 방향

- 콘텐츠 구조: 1f~5f 구역(region)을 가진 하나의 연속 보드. 칸 좌표(0~8, 0~8)와 층 `region`이 연결된다. 층별 전용 이벤트 풀이 이미 `1f_~5f_` 접두로 분리되어 있어 층 개성화가 가능하다. [근거: tiles.csv, tile_events.csv]
- 메타 진행: `meta.csv`(영구 업그레이드)와 `stages.csv.unlock_cost`. 방향: 판 종료 시 메타 포인트(rep 등) 환산 → 업그레이드 구매 → 다음 판 시작 스탯 반영. [추정: meta의 target이 stats.id이고 런 시작 시 반영된다는 문서]
- UI/UX: 03 UIUX/리소스 문서에서 상세화.
- 아트/사운드: 이미지 규격 통일, BGM 추가, 엔딩/스토리 컷 연결. 03 문서 참고.
- 저장/설정: 멀티 슬롯 세이브, `ui.csv`/`config.csv` 설정. 런타임 설정 패널은 미구현(UI 확장 잔여). [근거: docs/UI-EXTENSION-PLAN.md]

5. 제작 과정 설계

- 선행: 표 스키마/검증기 → 엔진 코어 → 샘플 스테이지 → 콘텐츠. 병렬 가능: 아트/사운드 제작, 번역, 스토리 본문.
- AI/도구 분담: AI(Antigravity/Claude)가 코드, 문서, 데이터를 작성하고 사용자가 방향과 확인을 맡는다. 엑셀 도구가 기획자 편집을 담당. [근거: 인벤토리 요약, docs/TOOLING.md]
- 검증 절차: `npm run data:validate` → `npm run dev` 플레이 → QA용 스테이지(`vntest`)에서 기능 단위 확인. 빌드는 `npm run build`(tsc 포함). [근거: package.json, .git/logs/HEAD]
- 인수인계: README + docs 5종이 사실상 인수인계 문서. HANDOFF 문서는 기록 없음. 다음 작업 시 정본 사본 확정과 HANDOFF 작성이 선행되어야 한다.

6. 세부기획서 목록

문서	정할 것
03_세부기획서_콘텐츠와밸런스	층별 구성, 이벤트 풀, 열쇠 경로, 추적자/아이템 수치, 엔딩
03_세부기획서_데이터툴링	표 스키마 명세, 검증 규칙, 엑셀 워크플로, 번역 파이프라인
03_세부기획서_UIUX와리소스	UiSettings 키 규약, HUD, 팝업, VN 화면, 아트/사운드 리소스 목록