온리 업: 자오 — 메인기획서

> 재구성 기획서 (역설계) / 작성 기준 시점: 2026-08-04 (프로젝트 시작 시점) / 문서 버전: Ver 0.1 프로젝트 ID: P020 / 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

이동/점프

- 의도(Why): 점프 정밀도가 등반의 재미이므로 관대한 보조와 가변 점프를 제공한다.
- 설계(How): 걷기 4.4, 질주 7.2(m/s), 지상 가속 60, 공중 가속 24, 마찰 15. 점프 초속 11.6, 상승 중 점프키 유지 시 중력 24, 아니면 34, 종단 속도 44. 코요테 0.10초, 점프 버퍼 0.13초, 단차 자동 상승 0.38. 이동 발판 위에서 점프하면 발판 속도(수평 ±6, 수직 0~5로 제한)를 이어받는다. 플레이어는 높이 1.44, 반폭 0.30의 AABB다. [근거: game.js 상수/`step`] 타이틀에서 「질주 점프가 가장 멀리 간다」고 안내한다. [근거: index.html]
- 필요(What): 고정 스텝(1/120초, 프레임당 최대 10스텝) 루프.
- 작업 방법: 입력(WASD, Shift, Space) → 가속 → 점프 판정 → 중력 → 축별 이동/충돌 → 지면 스냅(하강 이동 발판에 붙임).
- 완료 기준: 프레임레이트와 무관하게 같은 점프 거리.
- 리스크/대안: 프레임 급락 시 dt 상한 0.05초로 폭주 방지 [근거: game.js `frame`].
- 근거: [근거: game.js]

추락과 복귀

- 의도(Why): 「올라온 만큼 떨어진다」는 손실감이 핵심. 단 접근성을 위해 선택지를 둔다.
- 설계(How): 착지 시 공중 최고점과의 낙차가 10m 초과면 추락으로 집계하고 「추락! -Nm」 토스트와 실패음을 낸다. 어시스트 OFF는 발판 위에 있는 한 그 자리에서 계속하며(체크포인트 없음), 고도 -30m 미만의 나락에 빠지면 시작점으로 돌아간다. 어시스트 ON은 마지막 확보한 정착지보다 6m 넘게 낮게 착지하면 0.28초 뒤 정착지로 복귀한다. 정착지는 쉼터 발판에서 수평 2.6m/수직 2.4m 안에 들면 확보된다. [근거: game.js `onLanded`, `respawn`, `checkCheckpoints`]
- 필요(What): 정착지 목록, 추락 카운터, 설정 저장.
- 작업 방법: 낙차 추적 → 판정 → 토스트/효과음 → 복귀.
- 완료 기준: 어시스트 설정이 새로고침 후 유지된다 [근거: game.js `ouz_assist`].
- 리스크/대안: 타이틀은 「바닥까지 떨어진다」처럼 읽히나 코드는 지형 위에 걸리면 그대로 이어진다. 의도 문구와 실제 규칙의 정합 필요(03 문서 미해결).
- 근거: [근거: index.html, game.js]

특수 발판 6종

- 의도(Why): 같은 점프만 반복하면 단조로우므로 발판 성격을 다양화한다.

- 설계(How): 트램폴린(반동 속도 19.5), 왕복 이동 발판, 수직 엘리베이터, 원형 궤도 발판, 상승 기류(구역 내 상승 속도 최대 9.5, 가속 60), 미끄럼 패드(마찰 1.4/가속 11). [근거: game.js, level.js]
- 필요(What): 충돌체 `kind`/`mover` 필드, 시간 함수 기반 이동.
- 작업 방법: 기록 없음.
- 완료 기준: 각 발판이 점프 성능 안에서 도달/이용 가능.
- 리스크/대안: 고속 이동 발판 관통 → 플레이어를 발판 변위만큼 매 프레임 이동.
- 근거: [근거: level.js]

카메라

- 의도(Why): 수직 장애물에서 시야 가림을 막는다.
- 설계(How): 마우스 시야(포인터 락, 락이 안 되면 드래그 폴백), 휠 줌 3~15, 피치 -1.30~1.15, 카메라와 발판 사이 레이캐스트로 가림 시 거리 단축, 낙하 속도에 비례해 줌 아웃(최대 2.2). 클리어 시 3.2초 회전 연출. [근거: game.js]
- 필요(What): 기록 없음.
- 작업 방법: 기록 없음.
- 완료 기준: 가림 시 카메라가 발판을 뚫지 않는다.
- 리스크/대안: 포인터 락 해제 시 자동 일시정지.
- 근거: [근거: game.js]

2. 구현 방법

- 구조: `index.html`(UI) → `util.js`(수학, 시드 난수, 램프 텍스처, Web Audio) → `character.js`(자오조립, 관성 체인) → `level.js`(탑 생성/렌더) → `game.js`(루프, 물리, 카메라, HUD). 전역 네임스페이스 `OU`. [근거: index.html 스크립트 순서, util.js]
- 상태 머신: title → play ↔ pause → goal → win 카드. [근거: game.js `state`]
- 핵심 알고리즘: 진행 커서 기반 절차 생성, Y 버킷 브로드페이즈(5m), 정적 박스 인스턴싱. [근거: level.js]
- 배포: `node build-single.js`로 스크립트 5개를 인라인해 단일 HTML 생성. 순서는 index.html과 동일해야 하며 `</script`/`<!--`이 있으면 오류. [근거: build-single.js]
- 디버그: `OU.debug`(상태 접근, 고도별 순간이동 `tp`). [근거: game.js]

3. 시스템 간 상호작용과 밸런스 기준

- 핵심 변수: 점프 도달 거리/높이와 패턴 간격의 관계. 패턴 간격은 질주 점프 한계를 넘지 않도록 설정된다. [추정: 간격 수치가 점프 속도에서 역산된 것으로 보임 — level.js 주석 「점프 성능에 맞춰 간격을 계산」]
- 난이도 곡선: 고도 비율 t로 발판 크기(예: 기본 2.6→1.5)와 수평 간격(2.9→4.6)이 선형 보간되고, t<0.16 EASY / <0.45 MID / <0.78 HARD / 이상 BRUTAL 패턴 집합이 적용된다. [근거: level.js]
- 이동 발판 속도는 최대 약 2.4~3.8m/s 범위로 생성된다. [근거: level.js `hz` 호출부]

4. 게임의 나머지 부분의 방향

- 콘텐츠: 단일 탑. 추가 탑/테마 구역은 기록 없음.

- 메타 진행: 최고 고도, 추락 횟수, 시간만 있다. 보상/해금 없음. [근거: game.js]
- UI/UX: 03 문서 참고. 저장은 `localStorage`(`ouz_best`, `ouz_assist`). 설정 화면은 없음.
- 아트/사운드: 고정된 절차 합성 사운드, BGM 없음 [근거: util.js].

5. 제작 과정 설계

- 순서: util → 이동/충돌 → 레벨 패턴 → 캐릭터 → HUD → 연출 → 단일 빌드. 캐릭터와 레벨은 병렬 가능. [추정: 파일 의존 순서]
- AI/도구 분담: 코드 전체를 AI가 작성한 것으로 추정(주석 어조) [추정]. 검증은 브라우저 플레이와 `OU.debug`.
- 인수인계: HANDOFF 없음. 이 문서 세트가 대체한다.

6. 세부기획서 목록

문서	정할 것
`03_세부기획서_레벨디자인과_수치.md`	발판 패턴 명세, 난이도 곡선, 물리 상수 표, 검증 방법, UI, 아트, 사운드 리소스 목록