바운스페이스 — 메인기획서

> 재구성 기획서 (역설계) / 작성 기준 시점: 2026-11-06 (프로젝트 시작 시점) / 문서 버전: Ver 0.1 프로젝트 ID: P005 / 근거 수준: A / 작성: Claude (재구성) 이 문서는 결과물과 제작 기록으로부터 기획 의도를 역추정해 당시 작성한 것처럼 재구성한 것이다. 확인된 사실과 추정을 [근거]/[추정] 태그로 구분한다.

용어. **충격파**: 기본 무기, 직선 고속 투사체. **특수 능력**: 방 대기실에서 고르는 장비 5종. **장외**: 맵 경계 밖으로 나가 탈락하는 것. **튕김(KNOCKED)**: 맞거나 폭발에 휘말려 날아가는 상태. **사용 횟수**: 기본 공격과 특수 능력이 각각 가진 카운터. **틱**: 서버 시뮬레이션 1단계(30Hz).

1. 핵심 메카닉스

1.1 이동 상태와 거울 반사

- 의도(Why): 공중 궤도를 고정해 "읽는 재미"와 "정확한 네트워크 예측"을 동시에 얻는다. 입사각이 유지되는 반사는 플레이어가 벽을 이용한 궤도를 계산할 수 있게 한다.
- 설계(How): 상태는 `GROUNDED`(WASD 이동, 벽 미끄러짐), `AIRBORNE`(입력으로 궤도 변경 불가, 벽은 거울 반사, 바닥은 착지), `KNOCKED`(벽/바닥 모두 반사, 법선 방향 속도가 `landingSpeed`(1.5)보다 낮으면 착지), `DEAD`. 반사식은 `v' = e * (v - 2 * dot(v, n) * n)`이며 `restitution` 0.85, 일반 점프 중 벽은 `jumpWallRestitution` 0.7(수치 확정). 착지 임계값은 바닥에서 작게 끝없이 튀는 현상을 막기 위해 둔다. 걷기 가능 경사는 45도 이하. 땅에서 맞아도 위쪽 성분을 보장(`groundHitMinLift` 0.125)해 `KNOCKED`로 띄운다. 플레이어끼리는 피아 구분 없이 접근 속도 3m/s 이상이고 한쪽이 공중이면 동질량 탄성 충돌로 둘 다 `KNOCKED`가 된다.
- 필요(What): 구-삼각형 충돌, 균일 격자 분할, `balance.ts` 수치, 이동 테스트.
- 작업 방법: 상태 전이 → 지상 이동 → 점프 → 반사 → 착지 판정 순으로 구현. 담당 AI, 검증은 단위 테스트(공중 입력 무시, 45도 반사, 유한 틱 내 착지).
- 완료 기준: 위 세 테스트 통과, 브라우저에서 달리고 점프하고 벽에 반사된다.
- 리스크/대안: 벽 관통 → 이동 경로 전체 판정, 40m/s 관통 테스트. 땅에서 1회 후 무기력 → 체공 시간 단축을 먼저 시도.
- 근거: [근거: `IMPLEMENTATION_PLAN.md` 2.1, 3장, `balance.ts`, T-003, T-004]

1.2 점프 곡선과 걸음

- 의도(Why): 달 중력의 느린 호를 만들되, 전진 점프가 지나치게 멀리 가지 않도록 한다. 지상 이동은 달의 긴 박자로 느리게 한다.
- 설계(How): 중력 1.5m/s²(달 1.62에 가까운 계산하기 쉬운 값). 제자리 점프는 수직 4.5m/s(높이 6.75m, 체공 6초라고 계획서가 계산). 전진 점프는 각도 62.5도, 점프 순간 수평 3.6m/s에서 0.5초에 걸쳐 약 2.34m/s로 줄인 뒤 등속. 지상 `runSpeed` 3.375m/s. 속도 구간으로 판단해 스냅샷에 타이머를 넣지 않는다.
- 필요(What): `jumpSpeed`, `jumpAngleDeg`, `jumpLaunchSpeed`, `jumpSettleSeconds`, `runSpeed`, 화면 보빙.

- 작업 방법: 수치를 `balance.ts`에서만 바꾸고 계획서 3장을 함께 갱신한다. 사용자 플레이로 조정.
- 완료 기준: 사용자가 이동 느낌을 승인(G1).
- 리스크/대안: 수직 4.5는 에이전트 선택값이라 재확인 필요로 기록됨. 이후에도 수차례 사용자 요청으로 변경.
- 근거: [근거: `balance.ts`, `HANDOFF.md` 로그 11-06~11-07]

1.3 사용 횟수

- 의도(Why): 연사를 막고 "쏘려면 이동 상태가 필요"하게 만들어 플레이어가 계속 뛰게 한다.
- 설계(How): 기본 공격과 특수 능력이 각각 카운터를 갖는다. 착지하면 지상 1회, 이륙(점프 또는 피격)하면 공중 3회로 설정. 튕김 중 벽/바닥에 반사되면 공중 횟수와 제트팩이 충전(지상 횟수는 착지해야 충전). 사용 간 최소 간격 0.35초. 부활 시 전부 초기화. 제트팩은 공중 체류당 1회가 우선한다.
- 필요(What): 카운터 상태, 상태 전이 훅, HUD 두 줄 탄창(오른쪽 아래).
- 작업 방법: 규칙 단위 테스트로 충전 규칙을 모두 검증.
- 완료 기준: 2.2의 충전 규칙이 모두 테스트로 검증된다.
- 리스크/대안: 지상 1회가 답답함 → 수치 조정.
- 근거: [근거: `IMPLEMENTATION_PLAN.md` 2.2, `HANDOFF.md` T-007, 로그 11-07]

1.4 충격파와 착탄 폭발

- 의도(Why): 기본 무기가 곧 장외 도구다. 즉시 판정이 아닌 투사체로 해 방패 반사와 앞당김 보정을 가능하게 한다.
- 설계(How): 속도 70m/s, 중력 영향 없음, 사거리 90m, 피탄 반지름 0.9m(플레이어 키와 같은 지름), 직격 충격 16m/s. 플레이어, 벽, 바닥에 닿으면 반지름 2.7m 착탄 폭발(가장자리 6m/s, 중심에 가까울수록 강함). 사거리가 다해 사라질 때는 폭발하지 않는다. 쏜 본인은 아군 공격 설정과 무관하게 폭발에 휘말린다(D14). 발사 반동은 없다. 넉백은 현재 속도에 더한다.
- 필요(What): 투사체 시뮬레이션, 선분-캡슐 판정, 폭발 범위 판정, 이펙트.
- 작업 방법: 투사체 → 적중 → 넉백 → 장외/킬 귀속 → 허수아비로 확인.
- 완료 기준: 맞은 대상이 진행 방향으로 날아가고, 공중 피격은 속도 가산, 땅 피격은 위로 뜬다.
- 리스크/대안: 직접 맞히기 난이도 → G1 피드백으로 피탄 확대/폭발 추가.
- 근거: [근거: `IMPLEMENTATION_PLAN.md` 2.3, 1.2, `balance.ts`, `HANDOFF.md` T-008, 로그 11-06 G1]

1.5 특수 능력 5종

- 의도(Why): 같은 기본 이동 위에서 역할 차이를 만든다. 구속(볼라), 이동(제트팩), 방어(방패), 지속 교란(로켓), 지형/폭발(C4).
- 설계(How): 공통 장비 인터페이스 위에 5종. 대기실에서 선택, 부활 대기 중 Q로 변경(D18).

장비	규칙	수치(확정 표기)
로켓 추진체	적중 시 맞은 지점의 몸 기준 좌표를 저장해 추력과 회전을 가한다. 최대 3개 부착, 끝나면 몸 회전이 0.5초에 걸쳐 복귀. 벽에 맞으면 폭발 없이 소멸	속도 30, 추력 12, 지속 2.5초, 회전 0.3/최대 8 rad/s

장비	규칙	수치(확정 표기)
볼라건	발 기준 0.9m 이상이면 상체(공격/특수 능력 잠금), 미만이면 하체(점프 외 이동 잠금). 하체가 묶인 채 착지하면 수평 속도 0	속도 45, 지속 3초, 사거리 60
제트팩	공중 체류당 1회, 보는 방향으로 속도 방향 전환. 속력 유지, 최소 속력 보장. 튕김 중에도 사용 가능	최소 8m/s
반사 방패	전방 120도, 0.8초. 충격파, 로켓, 볼라를 반사하고 소유권이 방패 사용자에게 이전	0.8초, 120도
C4	상태표 `READY → THROWN → ATTACHED → SPENT → READY`, 폭탄은 동시에 1개, 설치, 격발, 재충전, 회수가 각각 1회 사용. 공중 3회가 한 주기와 맞음. 설치자도 폭발에 휘말림	던짐 18m/s, 반경 10m, 충격 22(중심)/8(가장자리), 회수 2.5m

- 필요(What): 몸 회전(`bodyQuat`, `angularVel`), 장비별 서버 상태, 스냅샷의 상태 비트, 1인칭 표현.
- 작업 방법: 각 장비를 공용 시뮬레이션 → 서버 동기화 → 클라이언트 표시 순으로 한 티켓씩 구현.
- 완료 기준: 계획서 2.3의 상태표 전 전이가 테스트로 검증된다. C4 4번째 입력 거부, 아군 공격 off에서 아군은 영향 없고 자신은 영향받는다.
- 리스크/대안: 로켓 회전 멀미 → 회전 완화(카메라 20%만 적용, 궤도 동일), 시작 화면 안내. C4 투척 속도 18은 저중력에서 맵 밖으로 날아가기 쉬워 G2에서 볼 값으로 기록되었다.
- 근거: [근거: `IMPLEMENTATION_PLAN.md` 2.3, 3장, `HANDOFF.md` T-021~T-026, 로그 11-06]

1.6 탈락/킬 귀속/점수

- 의도(Why): 체력이 없으므로 탈락 판정과 킬 판정이 공정하고 단순해야 한다.
- 설계(How): 경계 `killY`/`killRadius`/`killTop`(맵 데이터, `mare-01`은 -12/110/60)를 넘으면 탈락. 킬은 "그 생에서 나를 마지막으로 맞힌 다른 사람"에게 귀속(시간 제한 없음). 아군 킬/자살은 킬에서 제외. 맞힌 적이 없으면 자폭. 데스매치 점수는 킬 +1, 자폭 -1, 아군 킬 -1. 부활 3초 후, 보호 2초(공격하면 즉시 해제).
- 필요(What): 마지막 가격자 기록, 연쇄 충돌 귀속(맞아서 날던 사람이 부딪히면 처음 맞힌 사람에게 이어짐).
- 작업 방법: 킬 귀속 단위 테스트, 사인 추정은 클라이언트(`net/kill-cause.ts`)에서 이벤트로 판단.
- 완료 기준: 킬 귀속 테스트 통과, 허수아비 장외 시 킬 귀속.
- 리스크/대안: `balance.ts`의 `killCreditWindow` 10초가 남아 있어 규칙 설명과 어긋난다(미확인).
- 근거: [근거: `IMPLEMENTATION_PLAN.md` 2.4, 6.3, `HANDOFF.md` 로그 11-07]

2. 구현 방법

2.1 서버 권위 구조와 모노레포

- 의도(Why): 위치/적중을 클라이언트가 주장하지 못하게 하고, 규칙 코드를 한 번만 쓴다.
- 설계(How): npm workspaces `packages/shared`(규칙, 물리, 프로토콜), `apps/server`(Node.js + `ws`, 틱 루프, 방, 채팅), `apps/client`(Three.js, 입력, UI, 네트워크). 서버 틱 30Hz, 스냅샷 20Hz, 플레이어별 입력 큐(최대 4개, 틱당 1개 소비). 위치는 $x \mathrel{+}= v*dt + 0.5*a*dt^2$로 적분해 틱 주기와 무관한 같은 포물선을 그린다. 방마다 `Match`를 가진다.
- 필요(What): `ws`, Vite, 틱 루프, 입력 검증.
- 작업 방법: 프로토콜 → 틱 루프 → 예측/보정 → 원격 표시 → 네트워크 충격파 → 지연 시뮬레이션 순.

- 완료 기준: 2개 클라이언트가 서로를 쏴 장외시킨다. 지연 100ms에서 내 이동이 끊기지 않는다.
- 리스크/대안: WebSocket 지연 → 공중 외삽으로 완화, 전송 계층을 인터페이스로 분리해 WebTransport 추가 여지.
- 근거: [근거: `IMPLEMENTATION_PLAN.md` 4.1~4.3, `HANDOFF.md` T-010~T-015]

2.2 예측, 보간, 외삽

- 의도(Why): 공중 궤도가 결정적이라는 규칙을 활용해 FPS 네트워크의 가장 어려운 문제를 줄인다.
- 설계(How): 내 캐릭터는 입력 즉시 예측하고 서버 값이 오면 남은 입력을 다시 적용해 보정(큰 오차는 몇 틱에 나눠 흡수). 지상의 타인은 3틱 과거 보간, 공중의 타인은 공용 물리로 현재 서버 시각까지 외삽. 발사는 즉시 표시하고 서버 사건으로 이어받으며, 서버는 왕복 지연의 절반(최대 `maxFireLead` 0.1초)만큼 앞당겨 생성. 피격/탈락은 서버 이벤트로만 확정. 발사 입력에 클라이언트의 최종 조준 방향을 포함(회전 중 어긋남 방지).
- 필요(What): `net/prediction.ts`, `net/remote-players.ts`, ping/pong, 지연 시뮬레이션(`&lag`, `&jitter`).
- 작업 방법: 시뮬레이션 테스트(`network-lag.test.ts`)와 실서버 측정(`tools/net-hit-check.ts`)으로 명중률 확인.
- 완료 기준: 왕복 100ms/흔들림 20ms에서 공중 표적 명중이 납득할 수준. 기록된 시뮬레이션 값은 공중 초속 3/8/16m에서 100/100/92%, 실서버 실시간 초속 8m 90%, 초속 16m 70%.
- 리스크/대안: 땅 위 타인은 과거를 그려 움직이는 대상은 조금 앞을 쏴야 한다. 사람 조준 체감은 미확인.
- 근거: [근거: `IMPLEMENTATION_PLAN.md` 4.3, `HANDOFF.md` 로그 T-012~T-015]

2.3 모드 공통 구조와 방 흐름

- 의도(Why): 모드 6종을 같은 틀로 구현해 추가/수정이 쉽게 한다.
- 설계(How): `createRules(world, settings)`가 모드 규칙 객체를 만들고 결과(`MatchResult`), 점수(`ScoreRow`), 거점(`CapturePoint`), 상태(`ModeStatus`)를 공통 형태로 반환한다. 방 단계는 `waiting` → `countdown`(5초) → `playing` → `result`(15초) → `waiting`. 방장 전용 설정, 서버가 잘못된 모드/팀 조합을 거부.
- 필요(What): 모드별 규칙, 결과 화면, 관전.
- 작업 방법: 봇 입력 시뮬레이션으로 서버 없이 전 모드 종료 확인(T-032).
- 완료 기준: 모든 모드가 승자 또는 무승부로 끝난다.
- 리스크/대안: 모드 규칙 세부는 03 모드 문서 참조.
- 근거: [근거: `modes/rules.ts` 시그니처, `IMPLEMENTATION_PLAN.md` 2.6, `HANDOFF.md` T-027~T-032]

3. 시스템 간 상호작용과 밸런스 기준

- 핵심 변수는 중력, 충격 크기(16/22), 맵 반경(60), 장외 경계(-12/110/60), 사용 횟수(1/3)다. 계획서는 이 조합을 "중앙에서 1발은 안전, 가장자리 근처에서는 1발로 장외 위험"이 되게 맞췄다고 계산한다. [근거: `IMPLEMENTATION_PLAN.md` 3장 체감 표]
- 계획서 계산값: 땅에서 충격파 1회 맞으면 약 42m, 수직으로는 약 85m(위 경계 60m 초과). [근거: 3장]
- 계산으로 정한 시작값이며 플레이로 검증되지 않았다고 명시되어 있다. [근거: `HANDOFF.md` 2장]

- 상호작용: 방패 ↔ 모든 투사체(소유권 이전), 로켓 ↔ 제트팩(튕김 중 사용 가능), 볼라 하체 ↔ 착지(수평 속도 0), C4 ↔ 사용 횟수(3회 한 주기).

4. 게임의 나머지 부분의 방향

- **콘텐츠 구조:** 모드 6종 × 맵(첫 버전 1종, 이후 반사판 맵 추가) × 방 설정(인원 2~22, 팀 0/2~5, 아군 공격, 제한 시간, 목표 킬). [근거: 2.6, 로그 11-08]
- **메타 진행:** 없음(계정, 전적, 랭킹 제외). [근거: D4, 10장]
- **UI/UX:** 세부는 03 UI, 아트, 사운드 문서.
- **저장/설정:** `localStorage`의 `playerToken`/입장 코드, 음량, 음소거, 언어, 회전 완화. 서버 DB 없음. [근거: 2.6, T-016, 로그 11-08]
- **범위 밖:** 로그인, 매치메이킹, 모바일, 음성 채팅, 신고, 추가 맵/꾸미기, 관리자 도구, 결제. [근거: 10장]

5. 제작 과정 설계

- 순서: 재미 검증을 가장 먼저(오프라인 샌드박스) → 네트워크 → 로비 → 무기 → 모드 → 리소스 → 안정화. 선행은 `shared`의 수치와 타입이며, 서버/클라이언트가 같은 상수를 가져온다. [근거: `IMPLEMENTATION_PLAN.md` 5장, T-001]
- 병렬 가능: 규칙 단위 테스트와 화면 작업은 독립적이다. 단 계획서 기준 같은 AI 세션이 순차로 진행했다. [추정: 티켓 순서대로 단일 세션 로그]
- 분담: 에이전트가 구현/테스트, 사용자가 결정과 게이트(G1~G4) 승인. 에이전트는 계약, 결제, 비밀 정보 기록을 하지 않는다. [근거: `HANDOFF.md` G3]
- 검증: 티켓마다 `npm run check`, 화면 티켓은 브라우저 직접 실행, 흐름은 Playwright(`npm run e2e`), 부하는 봇 22명 10분(`npm run bots`), 네트워크는 `npm run net:check`. [근거: `HANDOFF.md` 5장, 6장]
- 인수인계: `HANDOFF.md`에 티켓/진행 기록/변경 금지 결정 D1~D21을 두고, 새 세션이 "다음 미완료 티켓"부터 이어간다. [근거: `HANDOFF.md` 1장]

6. 세부기획서 목록

문서	정할 것
`03_세부기획서_모드와맵.md`	모드 6종 규칙/승패, 점령전, 맵 JSON, 방 설정, 봇
`03_세부기획서_네트워크와데이터.md`	프로토콜, 스냅샷 필드, 세션/입력 방어, 수치 표, 배포/운영
`03_세부기획서_UIUX와아트사운드.md`	화면, HUD, 조작, 캐릭터 시점 표현, 리소스 예산, 이펙트, 사운드