엔트리 바이브코딩

AI 기반 엔트리 작품 제작·검증 하네스

하네스는 어떻게 작동할까

AI가 엔트리 작품을 만들고, 실행해 보고, 고치는 과정을 단계별로 따라갑니다.

이영호 · 경희대학교 소프트웨어융합학과



시작은 AI 에이전트에게 보내는 요청입니다

AI 에이전트 설명용 예문

엔트리로 레이싱 게임 만들어줘

하네스의 지식과 도구로 작업을 시작합니다

요청을 입력하세요

실제 제작 요청

2026-09-23 · 원문

이 프로젝트의 지식과 도구를 활용해, 유튜브에서 “이게 엔트리로 된다고?”라는 반응이 나올 만한 아웃런 스타일 3D 레이싱 게임(커브·언덕·라이벌 차·랩타임)을 최종본까지 완성해줘. 순수 엔트리 블록만으로 동작하는 .ent여야 하고, 중간에 묻지 말고 실제 플레이 검증까지 끝낸 뒤 보고해줘.

표시한 부분: 요청에 담긴 조건

설명을 위해 줄인 예문입니다. 실제 요청은 더 자세했고, 플레이 검증까지 포함했습니다.

기존 도구 둘 사이를 하네스가 잇습니다



요청을 받으면 하네스 안에서 6단계가 이어집니다

1 명세 작성
AI
`spec-dsl.mjs`
게임의 오브젝트와 동작을 중간 명세로
명세 spec.mjs

2 정적 검사
도구
`validateSpec`
생성 전에 블록·입력·참조 확인
통과 · 오류 위치

3 작품 생성
도구
`make-ent.mjs`
명세를 .ent 작품 파일로 변환
.ent 작품 파일

4 실행·관찰
도구
`editor-harness play-check`
실행하고 키를 눌러 변화를 기록
관찰 보고

5 작품별 검사
도구
`verify.mjs`
기대값과 실제값을 비교
통과·실패 집계

6 피드백
AI
AI 에이전트
결과를 읽고 명세를 고침
고친 명세

고친 명세로 ①부터 다시

● AI가 판단하는 단계 ● 도구가 처리하는 단계

① AI가 게임 동작을 중간 명세로 씁니다

AI

AI 에이전트

게임 동작을 정하고 명세로 작성

이 10줄에 담긴 동작

- ↑ 가속
- ↓ 브레이크
- ← → 조향

중간 명세 · DSL `games/sunset-drive/spec.mjs` 281–290행

실제 제작 명세 일부

```
281 if_(eq(v('state'), 2), [
282   if_(anyKey(key.up, key.z), [
283     inc('speed', mul(v('dt'), mul(ACC, sub(1.1, mul(g('spd'), 0.7))))), s('throttle', 1),
284   ], [
285     if_(anyKey(key.down, key.x), [], [inc('speed', mul(v('dt'), -DECEL))]),
286   ]),
287   if_(anyKey(key.down, key.x), [inc('speed', mul(v('dt'), -BRAKE)), s('brake', 1)]),
288   s('steer', 0),
289   if_(isPressed(key.left), [inc('px', mul(v('dt'), mul(-2.3, g('spd')))), s('steer', -1)]),
290   if_(isPressed(key.right), [inc('px', mul(v('dt'), mul(2.3, g('spd')))), s('steer', 1)]),
```

↑ · Z 속도 올리기 ↓ · X 브레이크 ← · → 좌우 위치 바꾸기

DSL — 엔트리 블록 명세를 쓰기 쉽게 만든 작성 함수입니다. 이 명세가 검사를 거쳐 엔트리 블록이 됩니다.

② 작품 파일을 만들기 전에 명세부터 검사합니다

명세 spec.mjs



생성 전에 · 실행하지 않고 파일만 읽음

```
node tools/make-ent.mjs <spec> --check
```

validateSpec 이 확인하는 것

블록 종류
엔트리에 등록된 블록인가 · 블록 목록 474종과 대조

입력 형식
입력 칸의 개수와 값의 종류가 맞는가

변수·함수 참조
선언된 변수·함수·오브젝트를 가리키는가

통과 → ③ 작품 생성으로

오류 → 위치와 이유를 보고 → AI가 명세 수정

생성 명령도 같은 검사를 먼저 거쳐, 오류가 남은 명세로는 .ent를 만들지 않습니다. 게임이 의도대로 움직이는지는 ④ 실행·관찰에서 확인합니다.

③ 생성기가 명세를 엔트리 작품 파일로 바꿉니다



명세를 블록·변수·함수 구조로 바꾸고 그림·소리를 함께 묶어, 엔트리가 여는 .ent 작품 파일을 만듭니다.

④ 실제로 실행하고, 무엇이 바뀌는지 기록합니다

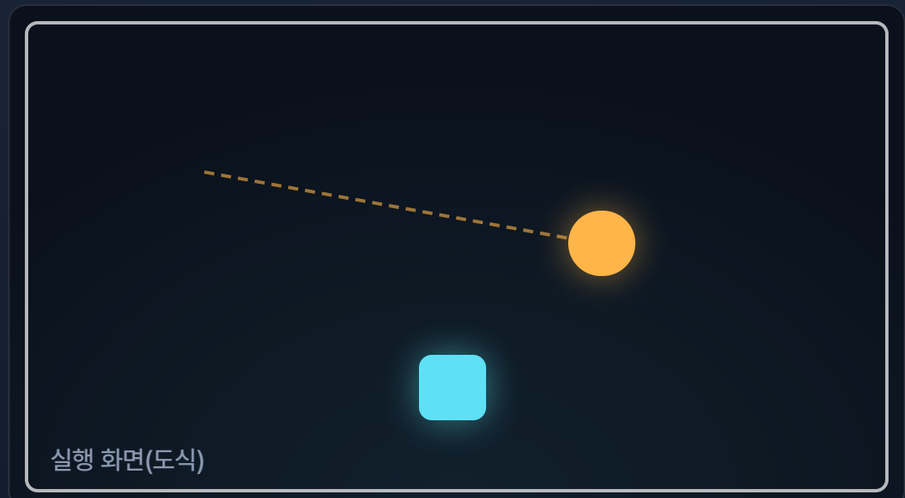
키 입력



keydown · keyup 이벤트만 보냄

키마다 0.6초씩 누르기

로컬 편집기 · 엔트리 엔진 — 헤드리스 실행



준비 신호 확인 | 작품 불러오기 | ▶ 실행

관찰 기록 play-check

불러오기
준비 신호를 확인하고 작품을 불러와 실행

그대로 2초
오브젝트별 움직임 · 바뀐 변수·글자·모양

키마다 0.6초
누르지 않은 실행과 비교해 달라진 점

화면 점검
가려져 안 그려지는 오브젝트 · 무대 밖 글상자

화면 캡처
무대 화면 한 장을 함께 남김

변화를 기록해 보고할 뿐, 게임 규칙이 맞는지는 판정하지 않습니다. 판정은 ⑤ 작품별 검사가 맡습니다.

⑤ 게임마다 지켜야 할 동작을 기대값과 비교합니다

작품별 규칙 검사

games/sunset-drive/verify.mjs

확인할 동작	기대	보관 기록	
↑ 3초 누르기	180 km/h 이상	198 km/h	✓
←, → 차례로 누르기	왼쪽·오른쪽으로 이동	px +0.01 → −0.65 → +0.27	✓
3바퀴 주행	결승 뒤 결과판	랩 38.9 · 36.0 · 36.0초, 결과판 표시	✓
저장 후 다시 열기	그림·소리·블록 유지	그림 98 · 소리 8 · 블록 3,031	✓

기대값과 실제값을 비교해 통과·실패를 집계합니다.

보관 검사 기록

2026-09-23

66 / 66 항목 통과

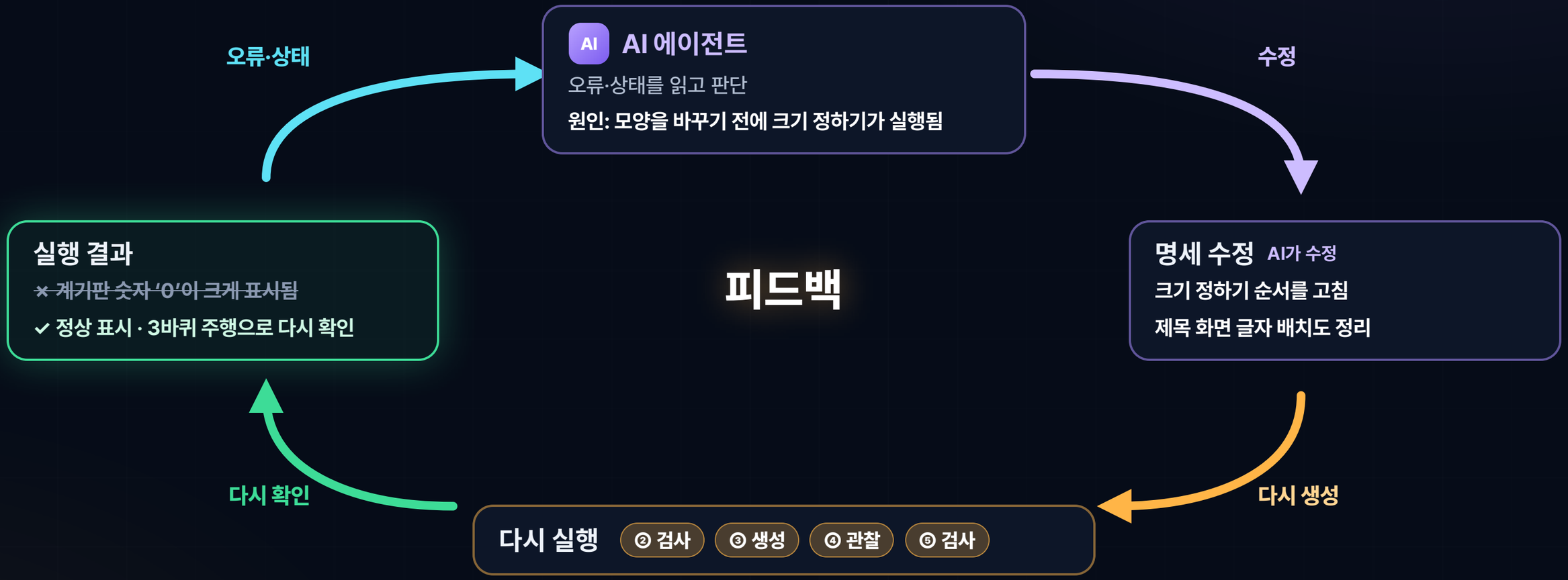
시작·구조	11
입력·조작	16
소리	15
레이스 진행	17
저장 후 다시 열기	7

sunset-drive_002.ent · 로컬 Entry 4.0.20 · 헤드리스 Chromium

당시 보관 기록입니다. 이 발표 자료를 만들며 새로 실행한 검사가 아닙니다.

⑥ AI가 결과를 읽고 고친 뒤, 다시 확인합니다

실제 제작 기록의 예 · 2026-09-23



실제 제작 기록에서도 같은 순서가 반복됐습니다

준비	**지식 문서와 기존 도구부터 확인**	작업 공간의 지식 문서와 .ent 제작 도구를 먼저 읽음	AI
①·②	**명세 작성 → 정적 검사 통과**	검토에서 찾은 문제 몇 가지를 첫 빌드 전에 먼저 고침	AI 도구
③·④	**빌드 후 헤드리스로 실행**	키 입력 봇으로 타이틀·카운트다운·주행 · 62 fps, 오류 0	도구
④→⑥	**캡처 화면에서 버그 2개 발견**	계기판 숫자 '0'이 크게 표시 · 제목 글자 겹침 → 고쳐서 다시 빌드	AI
④	**3바퀴 주행으로 다시 확인**	랩 계산 · 결승 · 결과 화면까지 오류 없이 완주	도구
⑤ 완료	**검사 통과 확인 → 문서 작성**	README와 지식 문서에 확인한 내용을 남겨 다음 작업에 재사용	도구 AI

2026-09-23 제작 작업 기록(영어)을 우리말로 요약했습니다. 중간의 소리 검사와 세부 다듬기는 생략했습니다.

05 · 정리

판단은 AI가, 확인은 하네스의 도구가 맡습니다

1

기존 도구를 잇는 하네스

AI 에이전트와 엔트리 엔진은 기존 도구를 쓰고, 그 사이의 지식·명세·생성·실행·검사 도구를 만들었습니다.

2

만들기 전에 검사, 만든 뒤에 실행

명세 작성 → 정적 검사 → 작품 생성 → 실행·관찰 → 작품별 검사 → 피드백

3

고치고 다시 확인

AI가 오류와 상태 보고를 읽고 명세를 고친 뒤 다시 검사합니다. 실제 제작 기록에서도 이 반복이 있었습니다.

> 자동 검사는 정해 둔 항목을 확인합니다. 모든 게임의 완성이나 성공을 보장하지는 않습니다.
> 하네스 구현과 검사에도 AI 코딩 도우미를 활용했습니다.

팀 **엔트리 블럭코딩을 위한 하네스제작**
경희대학교 소프트웨어융합학과 · 이영호 (1인 팀)

하네스 소스 코드 · GitHub ↗

자료실로 돌아가기