

AI 게임 제작실

움직이고, 피하고, 다시 시작하는 작은 게임 만들기

작은 게임 하나를 끝까지

이 책의 목표는 거대한 게임을 만드는 일이 아닙니다. 좌우로 움직이고, 장애물에 닿으면 끝나며, 다시 시작할 수 있는 작은 한 판을 만드는 일입니다. 그림이 움직이는 것보다 중요한 것은 놀렀을 때 예상한 일이 일어나는 것입니다.

먼저 완성 예제를 열어 봅니다. 그다음 AI에게 규칙을 설명하고, 한 부분씩 고치며 결과를 확인합니다. AI가 작성한 코드를 처음부터 모두 읽을 필요는 없습니다. 대신 바뀌면 안 되는 규칙을 말하고, 화면에서 확인할 줄은 알아야 합니다.

실습에는 기존 브라우저와 Python 3가 필요합니다. 새 게임 엔진이나 npm 패키지는 사용하지 않습니다. 필요한 도구가 없는 환경이라면 설치부터 따라 하지 말고 준비 상태를 먼저 확인하세요. 게임 실행 자체에는 계정·유료 API가 필요하지 않지만, AI 수정 요청에는 사용하는 도구의 조건이 적용됩니다.

이 책의 우주선·장애물·코드는 별도로 작성한 교육 예제입니다. AIToolsLab 회피 게임 제작기에서는 조작과 일시정지 경계를 살피는 관점을 참고했습니다. 원작의 캐릭터나 코드를 복제하는 실습은 아닙니다.

목차 · 시작에서 움직임까지

01 먼저 한 판을 해봅니다 · 5쪽

시작·이동·일시정지·끝난 화면을 직접 봅니다. 한 판 관찰 메모를 남기고 무엇이 달라져야 하는지 설명합니다.

02 AI에게 게임의 규칙을 말합니다 · 9쪽

분위기 설명을 행동·숫자·끝나는 조건으로 바꿉니다. 처음 만들기 요청문과 내가 정한 게임 규칙표를 함께 만듭니다.

03 캐릭터를 움직이고 화면 끝에서 멈춥니다 · 15쪽

방향키·버튼·드래그를 같은 이동 규칙으로 연결합니다. 누른 뒤 놓기, 화면의 좌표, 그림이 잘리지 않는 경계를 살펴봅니다.

먼저 한 판을 하고, 바꿀 것을 한 가지로 좁힌 뒤, 같은 행동으로 다시 확인하세요. 이미지 옆의 설명을 읽고 실제 동작은 실습 HTML에서 따라 해 봅니다.

목차 · 끝과 다시 시작까지

04 부딪히면 한 판이 끝나게 합니다 · 23쪽

충돌과 시간 종료가 겹쳐도 결과를 한 번만 정합니다.

05 멈추고 다시 시작해도 같은 규칙으로 · 31쪽

시간·장애물·입력을 함께 멈추고 새 한 판을 시작합니다.

06 보기 좋게 고치고 내 게임으로 바꿉니다 · 39쪽

버튼과 설명을 다듬고 속도 한 값으로 작은 변형을 만듭니다.

실습 파일을 여는 법 · 47쪽

자료와 확인 범위 · 48쪽

만든 자료와 이용 안내 · 49쪽

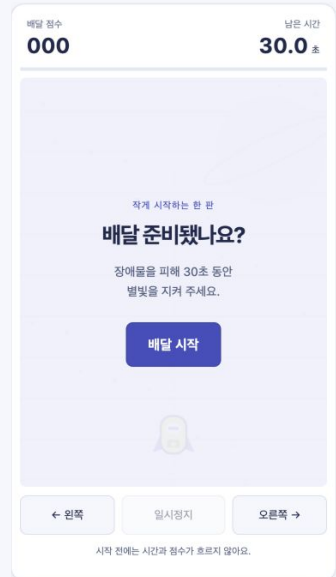
마지막에 남겨 둘 질문 · 50쪽

그림 옆의 실제 실행은 새 예제의 브라우저 화면입니다. 합성 조건은 특정 장면을 빨리 살피도록 시간이나 장애물을 준비한 시험입니다. 설명 도해는 원리를 그린 그림이지 실행 화면이 아닙니다.

01 먼저 한 판을 해봅니다

‘게임을 만들어 줘’라고 말하기 전에, 내가 만들 게임이 어떤 순서로 움직이는지 먼저 경험해 봅시다. 이번 결과물의 이름은 ‘별빛 배달부’입니다. 우주선을 좌우로 움직여 내려오는 장애물을 피하는 짧은 게임입니다.

실습 파일을 풀면 examples 폴더가 보입니다. 그 안의 README를 열고, 같은 폴더에서 명령창을 여세요. 기존 Python 3로 다음 명령을 실행합니다. 다른 프로그램이 이 포트를 쓰고 있다면 임의로 종료하지 말고 주소와 포트를 함께 바꿔야 합니다.



실제 Chrome 실행 화면 · 주요 동작은 본문에서 설명

```
python3 -m http.server 8767 --bind 127.0.0.1
```

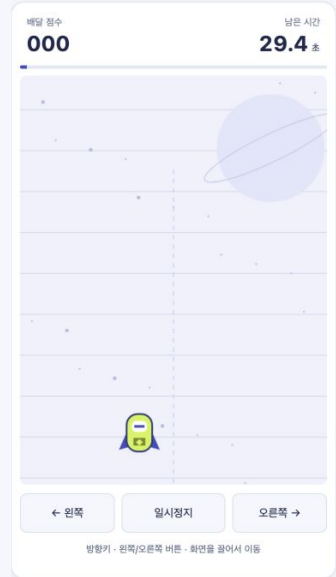
브라우저 주소창에 `http://127.0.0.1:8767/game-v1/` 을 입력합니다. 준비 화면에 점수 000과 남은 시간 30.0초가 보이면 출발할 수 있습니다. 아직 시작하지 않았으므로 기다려도 숫자는 바뀌지 않아야 합니다.

이 명령은 선택한 폴더의 파일을 같은 컴퓨터에서 열어 보는 용도입니다. 인터넷에 게임을 공개하는 절차가 아닙니다. Python 공식 문서도 `http.server`를 운영용 서버로 권장하지 않습니다. [Python http.server](#), 확인 2026-10-18.

시작하고, 조금 옆으로

‘배달 시작’을 누르면 한 판이 움직입니다. 우주선은 아래쪽에 있고 장애물은 위에서 내려옵니다. 처음부터 멀리 움직이지 말고, 오른쪽으로 조금 갔다가 왼쪽으로 돌아와 보세요. 어느 정도 움직이는지 느끼는 것이 먼저입니다.

키보드가 있으면 좌우 방향키를 씁니다. 화면 아래의 왼쪽·오른쪽 버튼을 눌러도 되고, 게임 화면을 좌우로 끌어도 됩니다. 서로 다른 입력 방법이지만 목적은 같습니다. 우주선의 가로 위치를 바꾸는 것입니다.



실제 Chrome 실행 화면 · 주요 동작은 본문에서 설명

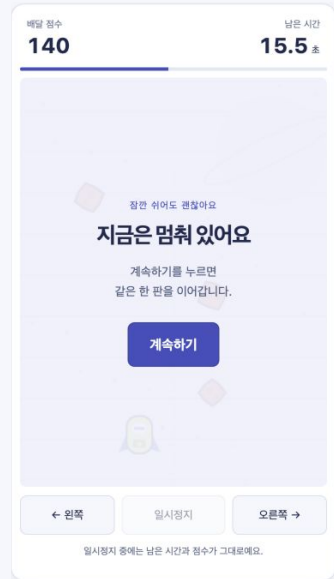
실제 Chrome 조작에서는 화면을 끌자 우주선이 왼쪽으로 옮겨졌습니다. 이 그림은 새 예제를 조작해 얻은 실제 화면입니다. 완성 이미지를 만들어 실행 화면처럼 붙인 것이 아닙니다.

입력이 먹지 않는다면 먼저 게임 화면을 한 번 선택하세요. 방향키가 페이지를 스크롤하는지, 게임을 움직이든지 구별해야 합니다. 동시에 여러 방법을 쓰기보다 한 방법씩 확인하면 원인을 찾기 쉽습니다.

잠깐 멈추는 것도 기능입니다

장애물이 가까워졌을 때 ‘일시정지’를 눌러 보세요. 우주선만 멈춘 것처럼 보이는 것으로는 부족합니다. 남은 시간, 점수, 장애물 위치도 함께 멈춰야 합니다. 화면이 가려져 있어도 뒤에서 시간이 흐른다면 같은 한 판을 쉬었다 이어가는 것이 아닙니다.

실제 조작에서 140점일 때 멈췄습니다. 남은 시간은 15.5초였고, 나중에 확인해도 같은 숫자였습니다. ‘계속하기’를 누른 뒤 다시 움직이는 장면도 확인했습니다. 특정 컴퓨터의 이 확인을 모든 기기의 성능 보장으로 읽지는 마세요.



실제 Chrome 실행 화면 · 주요 동작은 본문에서 설명

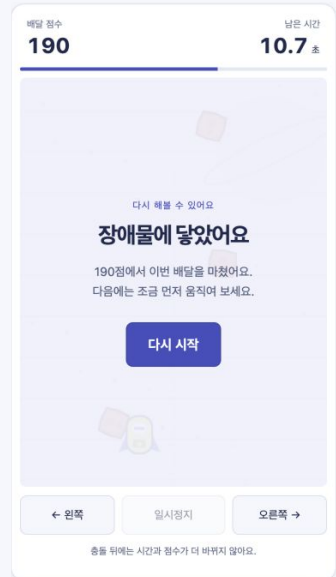
일시정지는 새 게임을 만드는 버튼이 아닙니다. 지금까지의 점수와 장애물을 보관한 채 진행만 멈춥니다. 반대로 ‘다시 시작’은 점수와 시간을 처음으로 돌리는 새 한 판입니다. 비슷해 보여도 다르게 동작해야 합니다.

다른 탭으로 옮겼다 돌아올 때도 직접 계속하기를 누르게 했습니다. 자리를 비운 동안 게임이 끝나 버리는 일을 막기 위해서입니다. 뒤 장에서는 이 규칙이 왜 입력 처리와 연결되는지 살펴봅니다.

끝난 뒤에 확인할 것

장애물에 닿으면 한 판이 끝납니다. 끝난 뒤 점수와 시간이 더 바뀌지 않는지 잠깐 살펴보세요. 결과 화면이 떴는데 뒤에서 점수가 오르거나 우주선이 계속 움직이면, 화면만 끝났고 게임의 처리는 끝나지 않은 셈입니다.

실제 한 판에서는 190점에서 충돌했고 남은 시간은 10.7초였습니다. 이것은 30초를 완주했다는 기록이 아닙니다. 제한 시간 직전으로 옮겨 종료만 확인한 별도 합성 시험과도 구분합니다.

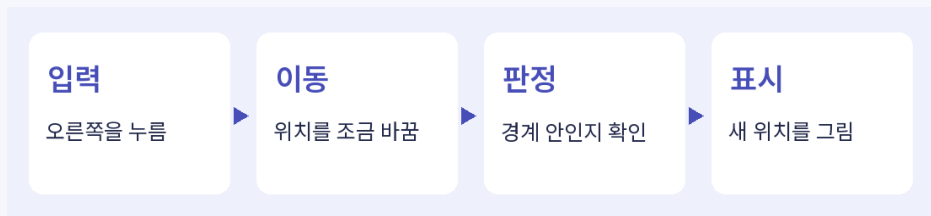


실제 Chrome 실행 화면 · 주요 동작은 본문에서 설명

이번 장의 결과물은 ‘한 판 관찰 메모’입니다. 예를 들어 ‘시작 전 000점 유지 → 드래그로 왼쪽 이동 → 140점에서 일시정지 → 직접 재개 → 190점에서 충돌 종료’라고 적을 수 있습니다. 잘한 점보다 행동과 결과가 이어지도록 적는 것이 중요합니다.

내 메모에는 ‘한 행동: ____ / 화면에서 달라진 것: ____ / 달라지면 안 되는 것: ____ / 다시 확인할 것: ____’을 남겨 보세요. 이제 AI에게 설명할 재료가 생겼습니다. 다음 장에서는 이 경험을 짧고 분명한 규칙으로 바꿉니다.

02 AI에게 게임의 규칙을 말합니다



설명 도해 · 예제의 규칙과 좌표를 시각화한 그림입니다.

‘귀엽고 재미있는 우주 게임’은 분위기를 전하지만, 버튼을 눌렀을 때 무슨 일이 생겨야 하는지는 정하지 못합니다. AI가 보기 좋은 화면을 만들었어도 시작·충돌·재시작이 내 기대와 다를 수 있는 이유입니다.

첫 요청에는 완성할 한 판을 말합니다. 무엇을 움직이고, 무엇을 피하며, 언제 끝나는지부터 정하세요. 이번 게임은 아래쪽 우주선을 좌우로 움직이고, 내려오는 장애물을 피하며, 진행 시간 30초가 되거나 장애물에 닿으면 끝납니다.

여기서 30초는 교육 예제의 설계값입니다. 누구에게나 가장 재미있는 길이라고 측정한 수치가 아닙니다. 독자가 한 판을 짧게 관찰하고 여러 번 다시 해 볼 수 있도록 정한 값입니다.

AI에게 규칙을 말기는 것이 아니라, 내가 정한 규칙을 파일로 옮기도록 돕게 합니다. 초보자도 이 결정은 직접 할 수 있습니다. 복잡한 이름을 외우는 것보다 ‘내가 눌렀을 때 이렇게 되면 좋겠다’를 구체적으로 말하는 일입니다.

숫자는 적게, 뜻은 분명하게

점수 규칙도 한 문장으로 정합니다. ‘게임이 진행된 온전한 1초마다 10점, 30초를 채우면 300점.’ 일시정지한 동안에는 진행 시간이 늘지 않으므로 점수도 오르지 않습니다.

‘온전한 1초’라는 말이 빠지면 0.5초에 5점을 줄지, 1초가 지난 뒤 10점을 줄지 모호해집니다. 여기서는 1.9초에도 10점입니다. 아래 표는 실제 플레이 성적이 아니라 정해 둔 규칙을 계산한 예입니다.

진행 시간	점수	이유
0.5초	0점	아직 1초 전
1.9초	10점	온전한 1초
14.5초	140점	온전한 14초
30초	300점	한 판의 상한

이 숫자만으로 게임의 난이도가 결정되지는 않습니다. 장애물이 내려오는 속도와 간격도 영향을 줍니다. 처음부터 숫자를 많이 조절하지 말고, 우선 시간과 점수의 관계가 맞는지 확인하세요. 작은 규칙 하나가 정확해야 다음 수정도 비교할 수 있습니다.

이번에는 하지 않을 일도 말합니다

순위를 넣고, 소리를 넣고, 계정을 만들고 싶을 수 있습니다. 하지만 기능이 늘면 확인할 것도 늘어납니다. 처음에는 제대로 끝나는 한 판이 목표이므로 기록 저장·로그인·광고·온라인 순위·소리는 빼 둡니다.

이렇게 범위를 정하는 것은 기능을 포기하는 일이 아닙니다. 무엇이 잘못됐는지 구분할 수 있는 크기로 시작하는 일입니다. 이번 게임에서 점수가 사라진다면 새로고침 후 저장을 약속하지 않았기 때문일 수 있습니다. 반대로 ‘다시 시작’을 눌렀는데 앞 판 점수가 남는다면 명확한 오류입니다.

하지 않을 일을 적어 두면 AI가 편의를 위해 외부 서비스나 패키지를 추가하는 것도 막을 수 있습니다. ‘새 설치 없이, 지금 있는 브라우저와 Python으로 실행할 수 있는 파일만’이라고 요청하세요. 실제로 없는 도구를 있다고 가정해서는 안 됩니다.

외부 업로드, 새 계정 연결, 결제, 공개 배포가 필요하다는 답을 받으면 그 단계에서 멈춥니다. 작은 로컬 게임을 만드는 요청이 그런 행동까지 자동으로 허락하는 것은 아닙니다. 파일을 어디에 만들고 무엇을 보존할지는 사람이 정합니다.

처음 만들기 요청문

다음은 실습 규칙을 다시 사용할 수 있게 정리한 교육용 요청문입니다. 특정 AI 제품에 그대로 실행해 같은 결과를 얻었다는 대화 기록이나 성공률 자료는 아닙니다.

기존 파일은 보존하고 새 폴더에 ‘별빛 배달부’를 만들어 줘. 우주선을 좌우로 움직여 내려오는 장애물을 피하는 한 판이야. 방향키, 화면 아래 버튼, 화면 드래그로 움직이게 해 줘. 진행 시간은 30초, 온전한 1초마다 10점이고 최대 300점이야. 장애물에 닿으면 한 번만 끝나고 결과 뒤에는 시간과 점수가 바뀌지 않게 해 줘.

일시정지하면 시간·장애물·점수·이동을 함께 멈추고, 직접 계속하기를 누르면 이어 가게 해 줘. 다시 시작은 새 한 판이야. 계정·소리·기록 저장·외부 전송은 넣지 마. 새 설치나 외부 서비스가 필요하면 먼저 알려 주고 멈춰 줘. 만든 파일과 확인 방법도 설명해 줘.

여기에 화면의 색과 느낌을 짧게 보낼 수 있습니다. 다만 색 설명으로 동작 규칙을 덮지 마세요. ‘밝은 바탕, 뚜렷한 우주선, 큰 조작 버튼’이면 처음 요청에는 충분합니다. 먼저 눌러 보고, 그 뒤에 예쁘게 고치는 편이 빠르게 원인을 찾는 데 도움이 됩니다.

AI의 답이 맞는지 확인하는 순서

파일 목록을 받았다고 완성으로 생각하지 마세요. 우선 새 폴더 안에만 파일이 생겼는지 봅니다. 그다음 안내된 주소를 열고, 시작하지 않은 화면에서 숫자가 가만히 있는지 확인합니다. 마지막으로 한 판을 직접 움직여 봅니다.

화면이 열리지 않으면 새 기능을 추가하지 않습니다. 서버 명령을 실행한 폴더가 맞는지, 주소의 포트가 같은지부터 대조하세요. Python 명령을 찾을 수 없다는 오류라면 파일 내용과는 다른 준비 문제입니다. 설치 명령을 무작정 이어 실행하지 말고 현재 환경을 먼저 확인합니다.

다음으로 시작·멈춤·충돌·다시 시작을 각각 해 봅니다. 규칙 한 줄과 행동 하나를 짝지으면 좋습니다. ‘일시정지’라는 글자가 있는지는 화면 검사이고, 실제로 시간이 멈췄는지는 동작 검사입니다. 둘 중 하나만 확인해서는 충분하지 않습니다.

AI가 ‘테스트 완료’라고 답하더라도 어떤 시험인지 물어보세요. 규칙 함수에 가상 값을 넣은 검사와 브라우저 버튼을 직접 누른 검사는 서로 다른 증거입니다. 둘 다 유용하지만, 어느 하나를 다른 하나처럼 소개하면 수정해야 할 부분을 놓치게 됩니다.

한 장으로 정리한 내 게임 규칙

이번 장의 작업물은 짧은 규칙표입니다. 어려운 문서 형식보다, 실행 화면과 비교할 수 있는 문장이 중요합니다. 아래는 이번 실습에서 이미 정한 내용을 채운 예입니다.

물어볼 것	별빛 배달부의 답
무엇을 움직이나요?	아래쪽 우주선을 좌우로
무엇을 피하나요?	위에서 내려오는 장애물
언제 끝나나요?	충돌 또는 진행 30초
멈추면 무엇이 유지되나요?	시간·점수·장애물 위치
다시 시작하면요?	0점·30초의 새 한 판
이번에 놓지 않는 것은요?	소리·저장·계정·온라인 순위

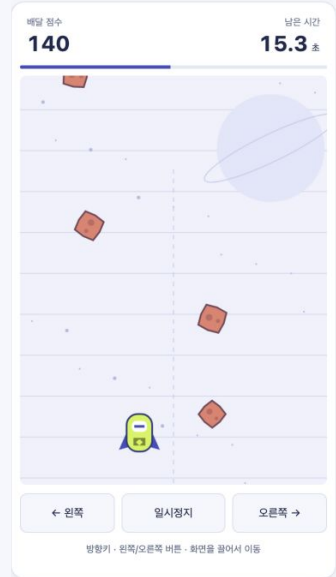
내 게임으로 바꾸고 싶다면 이 여섯 질문의 답만 별도 메모에 적어 보세요. ‘대상: ____ / 피할 것: ____ / 끝나는 조건: ____ / 멈춤: ____ / 재시작: ____ / 제외: ____’가 빈 양식입니다. 아직 기능을 늘리지 말고 이름이나 색부터 바꾸는 것도 괜찮습니다.

지금은 무엇을 만들지 정했습니다. 다음 장에서는 실제 입력을 우주선의 움직임으로 바꿉니다. 충돌 판정의 자세한 처리와 재시작 전체 구조는 뒤 장에서 다루므로, 여기서는 이동과 화면 경계만 따로 보겠습니다.

03 캐릭터를 움직이고 화면 끝에서 멈춥니다

오른쪽을 눌렀는데 우주선이 너무 멀리 가거나, 버튼에서 손을 뗐는데도 계속 움직인다면 무엇부터 고칠까요? 그림을 바꾸기 전에 입력과 움직임을 나누어 생각해 봅시다.

입력은 ‘오른쪽으로 가고 싶다’는 신호입니다. 움직임은 그 신호를 받아 우주선의 위치를 바꾸는 일입니다. 이 둘을 나누면 방향키와 버튼과 드래그가 같은 이동 규칙을 쓸 수 있습니다. 각 입력마다 따로 점수나 충돌을 처리할 필요도 없습니다.

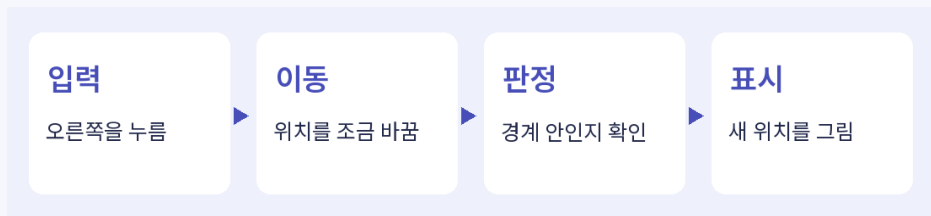


실제 Chrome 실행 화면 · 주요 동작은 본문에서 설명

이번 장의 확인 범위는 가로 이동입니다. 위아래 이동이나 점프를 더하지 않습니다. 먼저 오른쪽으로 갔다가 멈추고, 왼쪽으로 돌아오고, 양 끝에서 화면 밖으로 나가지 않는지 봅니다.

실제 플레이 화면에서 우주선과 장애물의 위치를 살펴보세요. 이 그림은 정지된 PDF 한 컷입니다. 조작의 느낌은 같은 예제 HTML에서 확인합니다. 보기 좋은 정지 그림과 손에 맞는 조작은 다른 문제입니다.

누르고 있는 동안만 움직이기



설명 도해 · 예제의 규칙과 좌표를 시각화한 그림입니다.

버튼을 한 번 누를 때마다 우주선을 조금씩 옮기면 단순해 보입니다. 그러나 누르고 있는 동안 부드럽게 움직이는 게임에서는 ‘지금 어느 방향을 누르고 있는가’를 기억하는 편이 이해하기 쉽습니다.

오른쪽을 누르면 오른쪽 입력이 켜지고, 손을 떼면 꺼집니다. 화면을 갱신할 때 그 입력이 켜져 있으면 우주선을 조금 오른쪽으로 옮깁니다. 왼쪽도 같은 방식입니다. 양쪽이 동시에 눌리면 이번 예제에서는 서로 상쇄해 움직이지 않도록 했습니다.

핵심은 ‘떼기’를 ‘누르기’만큼 중요하게 다루는 것입니다. 누를 때의 처리만 만들면 방향이 켜진 채 남아 버릴 수 있습니다. 계속 움직이는 문제가 생기면 속도 숫자보다 입력이 언제 꺼지는지 먼저 보세요.

AI에게는 ‘방향키와 버튼은 누르고 있는 동안만 이동하고, 놓거나 입력이 취소 되면 멈추게 해 줘’라고 요청할 수 있습니다. 다만 이것은 수정 방향을 말하는 문장입니다. 실제로 고쳐졌는지는 버튼을 눌렀다 떼는 행동으로 확인해야 합니다.

빠른 컴퓨터에서 더 빨라지지 않게

화면을 한 번 그릴 때마다 무조건 같은 거리를 옮기면, 화면을 자주 그리는 환경에서 더 빠르게 움직일 수 있습니다. 움직임을 화면 횟수에만 매달지 않는 이유입니다.

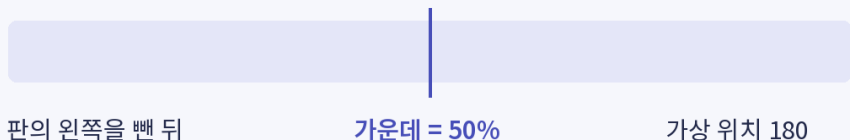
이 예제는 게임 규칙을 작은 시간 간격으로 나누어 처리합니다. 이동 속도는 가상 화면 단위로 초당 300입니다. 한 번의 기본 처리 간격이 1/60초이므로 오른쪽 입력이 계속 켜져 있으면 한 처리에서 5만큼 옮깁니다. 이것은 예제의 계산 방법이지, 모든 기기에서 초당 60장으로 실행된다는 측정 결과가 아닙니다.

브라우저의 `requestAnimationFrame`은 다음 화면 갱신 시점에 함수를 호출하도록 요청하는 방법입니다. 계속 갱신하려면 다음 요청을 다시 해야 합니다. 호출 간격을 고정된 성능 수치로 생각하면 안 됩니다. [MDN requestAnimationFrame](#), 확인 2026-10-18.

초보 단계에서 계산 구조를 전부 새로 만들 필요는 없습니다. ‘화면 갱신 횟수에 따라 게임 속도가 달라지지 않게 하고, 숨졌다가 돌아올 때 밀린 시간을 한꺼번에 적용하지 말아 줘’라고 요청한 뒤 실제 동작을 비교하세요. 성능 문제가 보이면 어떤 기기와 행동에서 생겼는지도 함께 남깁니다.

화면을 끌 때 위치가 어긋난다면

같은 가운데, 다른 화면 크기



설명 도해 · 예제의 규칙과 좌표를 시각화한 그림입니다.

키보드는 방향을 말하지만, 드래그는 가고 싶은 위치를 가리킵니다. 여기서 주의할 점은 화면의 가로 위치와 게임 안의 가로 위치가 같은 숫자가 아닐 수 있다는 것입니다.

게임의 가상 가로 폭은 360입니다. 브라우저에서는 공간에 맞춰 더 작거나 크게 보입니다. 브라우저 창의 왼쪽부터 댄 위치를 그대로 우주선에 넣으면 게임 판이 오른쪽에 있을 때 크게 어긋납니다.

먼저 게임판의 왼쪽 위치를 빼고, 현재 보이는 게임판 너비에 대한 비율을 구합니다. 그 비율을 가상 폭 360에 맞추면 됩니다. 판의 가운데를 가리켰다면 화면 크기와 상관없이 가상 위치 180이 되는 원리입니다.

문제를 설명할 때는 ‘드래그가 이상하다’ 대신 ‘작은 화면에서 판 가운데를 눌렀는데 우주선이 오른쪽 끝으로 간다’고 적어 보세요. 무엇을 기대했고 무엇을 봤는지가 함께 있어야 AI도 고칠 지점을 좁힐 수 있습니다. 이 원리를 설명하는 도해의 숫자는 위치 계산 예이며 실제 기기 측정값이 아닙니다.

화면 끝에서 멈추는 이유

가운데가 아니라 몸 전체가 보이게



설명 도해 · 예제의 규칙과 좌표를 시각화한 그림입니다.

우주선의 가운데가 화면 안에 있다고 몸 전체가 보이는 것은 아닙니다. 가운데 위치를 가로 폭의 끝인 360까지 허용하면 우주선의 오른쪽 부분이 잘릴 수 있습니다.

이번 예제에서는 우주선 가운데를 22부터 338 사이로 제한합니다. 이 여백은 예제의 크기와 모양에 맞춘 설계값입니다. 0과 360을 기준으로만 막는 대신, 우주선 전체가 보이도록 양 끝에 공간을 남겼습니다.

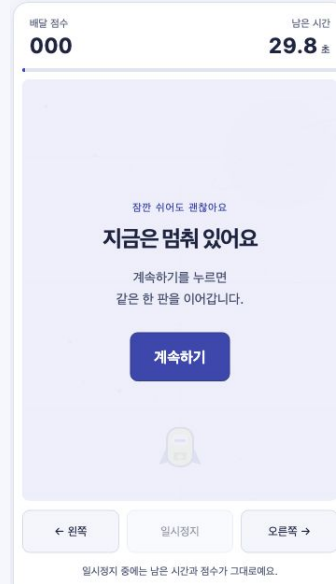
AI에게는 ‘우주선의 가운데만 보지 말고 몸체가 게임판 밖으로 잘리지 않게 좌우 경계를 제한해 줘’라고 요청합니다. 새 그림으로 바꿨다면 같은 22를 무조건 재사용하지 말고 그림의 실제 폭을 다시 보아야 합니다.

확인도 간단합니다. 왼쪽으로 충분히 움직인 뒤 더 누르고, 오른쪽에서도 같은 행동을 합니다. 그림이 경계에서 멈추는지와 손을 떼 뒤 입력이 꺼지는지를 따로 봅니다. 함수 시험에서는 좌우 경계를 벗어나는 값을 주어 제한되는지 확인했고, Chrome 조작에서도 오른쪽 끝까지 이동한 결과를 확인했습니다.

작은 화면에서는 버튼부터

컴퓨터에서는 드래그가 편해도 작은 화면에서는 손가락이 우주선을 가릴 수 있습니다. 그래서 아래쪽에 왼쪽·일시정지·오른쪽 버튼을 따로 두었습니다. 그림을 가리는 동작과 그림 밖에서 조작하는 동작을 선택할 수 있게 한 것입니다.

너비 390과 360의 브라우저 화면으로 줄여 배치를 확인했습니다. 조작 버튼 높이는 46 화면 단위였고 가로 넘침은 관찰되지 않았습니다. 360 너비에서는 오른쪽 버튼 가장자리 부근을 실제로 눌러 우주선이 움직인 것도 확인했습니다.



실제 Chrome390×844 조건 · 실제 휴대폰 시험 아님

이 결과는 실제 휴대폰의 터치 시험이 아닙니다. 작은 브라우저 화면을 점검한 기록입니다. 손가락 크기, 기기 화면, 모바일 브라우저의 동작은 별도로 확인해야 합니다. 숫자 하나로 ‘모바일 검수 완료’라고 부르지 않는 이유입니다.

오른쪽 버튼과 드래그가 서로 간섭한다면 조작 영역이 겹쳤는지 먼저 봅니다. 버튼을 누른 행동이 동시에 게임판 드래그로 처리되면 예상과 다른 움직임이 생길 수 있습니다. 조작 방법이 많을수록 한 방법씩 따로 시험하는 편이 안전합니다.

한 부분만 바꾸는 요청문

이동이 너무 빠르다고 느껴지면 속도와 색과 장애물을 한꺼번에 바꾸지 않습니다. 먼저 이동 속도만 바꾸고 같은 행동을 다시 해 보세요. 아래는 한 부분 수정을 설명하기 위한 교육용 재구성입니다. 이미 실행해 효과를 측정한 수정 기록은 아닙니다.

기존 동작은 보존하고 우주선의 좌우 이동 속도만 조정해 줘. 바꿀 숫자와 파일을 먼저 설명해 줘. 방향키·버튼·드래그 모두 같은 속도 제한을 쓰게 하고, 좌우 경계·점수·시간·장애물 규칙은 바꾸지 마. 수정 전 파일도 남겨 줘. 확인할 행동을 세 가지 이내로 알려 줘.

수정한 파일을 열기 전에는 무엇이 바뀌었는지 설명을 읽습니다. 요청하지 않은 패키지나 외부 서비스가 들어갔다면 진행을 멈추고 이유를 묻습니다. 속도를 고치는 작업에 새 계정 연결이 필요하다는 말은 이번 범위와 맞지 않습니다.

다시 실행한 뒤 오른쪽으로 이동하고, 손을 떼고, 화면 끝까지 가 봅니다. 느려졌다는 느낌만 보지 말고 멈춤과 경계가 그대로인지도 살펴보세요. 원하던 결과가 아니면 여러 숫자를 연달아 바꾸기보다 마지막 수정 전으로 돌아가 비교합니다.

움직임을 설명할 수 있게 됐습니다

이번 장의 작업물은 ‘이동 확인 카드’입니다. 한 가지 결과만 적지 말고 입력·이동·정지를 이어 적어 보세요. 이번 실습의 좁은 화면 확인에서는 오른쪽 버튼을 누른 뒤 가상 위치가 305가 되었고, 손을 뗀 뒤 입력 방향은 0으로 비워졌습니다. 305가 모든 환경에서 나와야 하는 정답은 아닙니다. 누른 시간과 관찰 시점에 따라 달라집니다.

이동 카드의 빈 양식은 ‘입력 방법: ____ / 누른 곳: ____ / 기대한 방향: ____ / 실제 이동: ____ / 놓은 뒤: ____ / 경계에서: ____’입니다. 예측과 관찰이 다르면 그 줄만 AI에게 전달해도 됩니다.

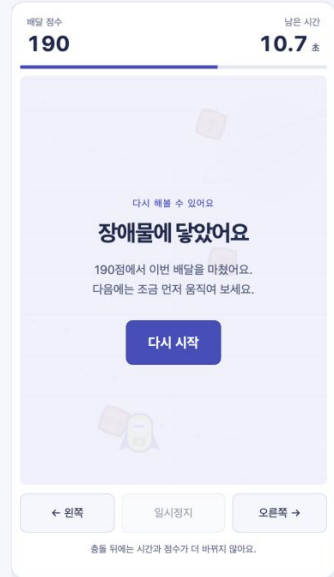
이제 ‘캐릭터를 움직여 줘’보다 구체적으로 말할 수 있습니다. 무엇을 누르는 동안 움직여야 하는지, 화면 좌표를 어떻게 맞추는지, 끝에서는 어디까지 허용할지 정했습니다. 그림을 바꾸더라도 이 질문들은 그대로 쓸 수 있습니다.

다음 장은 부딪혔을 때입니다. 이동이 멈추는 것과 한 판이 끝나는 것은 다릅니다. 충돌했는데 점수는 계속 오르거나 결과가 두 번 뜨는 문제는 종료 규칙에서 다루겠습니다. 일시정지 문제는 05장에서 오류판과 정상판을 같은 조건으로 비교하며 살펴봅니다.

04 부딪히면 한 판이 끝나게 합니다

장애물을 피하다가 우주선이 닿았습니다. 결과 화면이 나왔으니 끝난 걸까요? 그 뒤에도 점수가 오르거나, 다시 시작하자 결과가 곧바로 뜬다면 아직 끝나는 규칙이 맞지 않은 것입니다. 충돌 그림보다 ‘끝난 뒤 더 하지 않는 것’을 먼저 살펴봅시다.

03장에서는 우주선을 움직이고 화면 끝을 지켰습니다. 이제는 같은 움직임 위에 한 가지 질문을 더합니다. ‘지금 우주선이 장애물에 닿았는가?’ 답이 맞으면 이번 한 판을 한 번만 끝내야 합니다.



실제 Chrome 실행 화면 · 주요 동작은 본문에서 설명

실제 플레이에서는 190점에서 충돌했고 남은 시간은 10.7초였습니다. 결과 화면이 그 점수를 보여 준 뒤 더 바뀌지 않았습니다. 이 장면은 실제 조작의 실패 결과이지, 제한 시간 30초를 채운 성공 장면이 아닙니다.

이 장의 목표는 더 정교한 물리 계산이 아닙니다. 독자가 납득할 크기로 닿음을 정하고, 결과를 한 번만 남기고, 끝난 뒤 진행을 막는 것입니다. 새 무기나 생명수를 추가하지 않고도 게임의 약속을 훨씬 분명하게 만들 수 있습니다.

보이는 그림과 닿는 범위는 다릅니다

우주선은 몸체와 날개가 있고 장애물은 울퉁불퉁합니다. 그림의 모든 점을 비교하려 하면 첫 게임이 복잡해집니다. 이번 예제는 우주선과 장애물을 각각 작은 원으로 보고, 두 원이 만나면 닿았다고 판단합니다.

우주선의 원은 반지름 16, 장애물 하나의 원은 그 장애물에 정한 크기를 씁니다. 예를 들어 장애물 반지름이 18이면 두 중심의 거리가 34 이하일 때 닿은 것으로 처리합니다. 이 숫자는 가상 화면에서 쓰는 설계값이며 실제 센티미터나 제품 공식 기준이 아닙니다.

원이 그림의 윤곽과 완전히 같지는 않습니다. 따라서 ‘그림의 모든 가장자리를 정확히 판정한다’고 설명하면 안 됩니다. 우주선을 크게 바꿨는데 원만 그대로 두면 그림이 겹쳐도 통과하거나 멀리서 끝날 수 있습니다.

합성 시험 화면의 ‘충돌 원 보기’를 켜면 판정 범위를 살필 수 있습니다. 이 표시는 설명을 돕는 안내입니다. 원의 크기를 바꿀지 결정할 때는 먼저 무엇이 불공평하게 느껴졌는지 적고, 그림과 범위를 함께 보세요. 화려한 효과를 추가하는 것으로 판정의 어긋남이 해결되지는 않습니다.

한 번 처리할 때 묻는 순서

진행 중일 때만, 정한 순서로



설명 도해 · 예제의 규칙과 좌표를 시각화한 그림입니다.

한 판이 진행되는 동안 코드는 작은 시간 단위로 일을 반복합니다. 이번 예제에서는 먼저 입력에 따라 우주선을 옮기고 진행 시간과 점수를 계산합니다. 다음으로 장애물을 내린 뒤 닿았는지 확인합니다. 닿지 않았다면 제한 시간이 끝났는지 봅니다.

이 순서를 미리 정하는 이유는 같은 처리 안에서 충돌과 시간 종료가 함께 생길 수 있기 때문입니다. 30초가 된 바로 그 처리에서 장애물에도 닿았다면, 어느 결과를 보여 줄까요? 이번 게임은 충돌을 우선하도록 정했습니다.

다른 게임에서는 시간 완료를 먼저 인정할 수도 있습니다. 어느 쪽이 유일한 정답은 아닙니다. 중요한 것은 만든 사람이 규칙을 정하고, AI가 작성한 파일도 그 순서를 지키게 하는 것입니다.

게임판을 빠르게 넘겨 보다가 우연히 그런 순간을 만나길 기다릴 필요는 없습니다. 함수 시험에서는 종료 직전 시간과 충돌 위치를 직접 준비해 같은 처리에 넣었습니다. 실제 플레이와 다른 합성 조건이지만, 경계 규칙 하나를 반복해서 살피는 데 도움이 됩니다.

끝났다는 결정을 한 곳에서

충돌 코드에서 결과를 띄우고, 시간 코드에서 또 결과를 띄우면 같은 한 판이 두 번 끝날 수 있습니다. 닳았다는 판단과 끝났다는 결정을 나누는 편이 안전합니다. 닳음이나 시간은 이유이고, 결과 확정은 한 번만 하는 일입니다.

예제의 종료 처리는 지금 상태가 ‘진행 중’일 때만 작동합니다. 한 번 충돌 결과가 정해지면 상태가 ‘끝남’으로 바뀝니다. 다음 처리에서 다시 닳았다고 판단해도 진행 중이 아니므로 같은 결과를 더 만들지 않습니다.

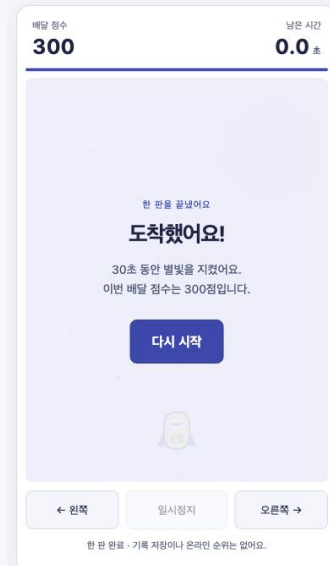
독자가 코드를 볼 때는 함수 이름을 외우기보다 질문을 붙여 보세요. ‘이미 끝난 게임인데 또 끝내려 하는가?’ ‘끝난 뒤에도 점수나 위치를 바꾸는가?’ 화면에 결과가 보인다는 사실만으로 이 질문의 답이 정해지지 않습니다.

AI에게 수정할 곳을 한곳으로 모아 달라고 요청하는 이유도 여기에 있습니다. 결과 화면마다 서로 다른 초기화나 점수 계산이 들어 있으면 나중에 한 부분만 고치기 어렵습니다. 결과를 확정한 뒤 화면은 그 결과를 보여 주는 역할만 맡게 합니다.

끝나는 순간을 일부러 만들어 봅니다

시간 종료를 확인하려고 매번 30초를 살아남을 필요는 없습니다. 합성 시험의 ‘종료 직전 가상 시간’은 진행 시간을 29.95초로 옮기고 장애물을 비워 종료 부분만 살핍니다. 이때 보이는 도착 화면은 실제 30초 완주 기록이 아닙니다.

이 조건에서는 다음 처리들이 이어진 뒤 시간이 30초, 점수가 300점으로 고정돼야 합니다. 실제 합성 시험에서도 이 결과를 확인했습니다. 그 다음에는 방향키를 누르더라도 끝난 한 판의 시간이나 점수가 바뀌지 않아야 합니다.



실제 브라우저에 만든 합성 종료 조건 · 실제30초 완주 아님

충돌은 별도로 봅니다. ‘충돌용 장애물 넣기’는 우주선의 현재 위치에 장애물을 넣어 종료를 재현합니다. 이런 시험에서는 얼마나 잘 피했는지 평가하지 않습니다. 다음 한 번이 결과 한 번으로 이어지지만 확인합니다.

두 시험을 섞어 ‘게임이 쉽다’거나 ‘끝까지 실행이 안정적이다’라고 확대하지 마세요. 특정 조건이 맞다는 증거와 전체 플레이 경험이 좋다는 평가는 다릅니다. 실제 한 판과 합성 시험을 함께 남기면 각각 무엇을 확인했는지 분명해집니다.

화면만 가리면 생기는 문제

이런 잘못된 수정을 가정해 봅시다. 장애물에 닿자 큰 안내창은 띄우지만, 시간과 점수를 계산하는 반복 처리는 그대로 둡니다. 사람은 끝났다고 생각하는데 코드 안에서는 아직 진행 중입니다.

안내창 뒤에서 점수가 오르면 결과에 적힌 숫자와 실제 값이 달라질 수 있습니다. 다시 시작할 때 남아 있던 장애물이 곧바로 부딪히거나, 이전 결과 처리가 뒤늦게 나타날 수도 있습니다. 여기서 설명한 장면은 잘못된 구조를 이해하기 위한 가정이며 이 게임에서 관찰했다고 꾸민 실행 기록이 아닙니다.

이럴 때 안내창 색이나 글자를 먼저 고치지 않습니다. 결과를 확정된 뒤 진행 상태가 바뀌는지, 진행 중이 아니면 이동·시간·충돌 계산을 건너뛰는지부터 봅니다. 보이는 화면과 안쪽 규칙이 같은 상태를 가리켜야 합니다.

수정 뒤에는 결과 화면을 잠시 두고 숫자가 고정되는지 확인합니다. 이어서 다시 시작을 누르고 처음 점수가 0인지 봅니다. 뒤 장에서는 이 재시작을 더 자세히 다룹니다. 지금은 끝나는 처리와 새로 시작하는 처리가 이어지되 섞이지 않는다는 점을 기억하세요.

문제를 설명해 고치는 요청문

다음은 종료 문제가 있을 때 사용할 교육용 요청문입니다. 이 문장으로 독립 AI 제품을 반복 실행해 성공률을 측정한 기록은 아닙니다. 내가 본 증상과 기대를 함께 적는 방법을 보여 줍니다.

결과 화면은 나오지만 점수가 계속 오르는 문제가 있어. 충돌 또는 시간 종료가 정해지면 한 판을 한 번만 끝내고, 이후 이동·시간·점수·충돌 처리가 멈추게 고쳐 줘. 두 종료 조건이 같은 처리에서 생기면 충돌을 우선해. 기존 입력·그림·점수 규칙은 보존하고 수정 전 파일도 남겨 줘.

실제 버튼을 눌러 확인한 것과 가상 시간·장애물을 넣어 확인한 것을 나눠 알려 줘. 끝난 뒤의 숫자 고정, 같은 한 판의 종료 횟수, 다시 시작의 초기 상태를 확인해 줘. 새 도구나 외부 연결이 필요하면 실행하지 말고 먼저 설명해 줘.

여기에 ‘어느 화면에서 무엇을 누른 뒤 몇 점이 계속 올랐는지’를 보낼 수 있습니다. 오류 문구가 있으면 그대로 복사하되 개인 정보나 계정 정보는 빼세요. 막연히 처음부터 다시 만들라고 하기보다 종료 처리만 좁혀 고치면 정상 동작을 보존하기 쉽습니다.

이번 한 판의 끝을 기록합니다

이번 장의 작업물은 ‘종료 확인 메모’입니다. 결과 화면 한 장보다 어떻게 그 결과를 만들었는지가 중요합니다. 아래는 이번 예제에서 확인한 내용을 실제와 합성으로 나눈 기록입니다.

조건	확인한 결과	범위
실제 플레이 중 충돌	190점·남은 10.7초에 서 종료	실제 조작
장애물을 현재 위치에 삽입	한 판 종료 횟수 1	합성 충돌
시간을 29.95초로 준비	30초·300점으로 종료	합성 시간
끝난 상태에서 추가 처 리	시간·점수·위치 유지	함수 시험

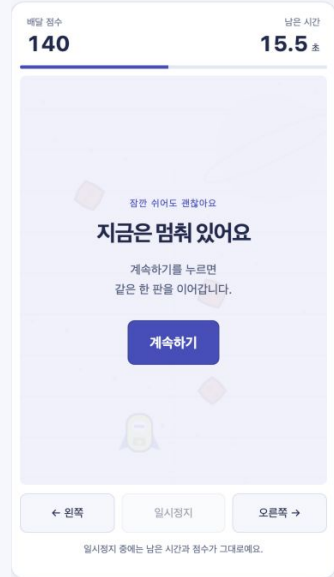
내 기록의 빈 양식은 ‘조건: ____ / 실제 또는 합성: ____ / 끝난 이유: ____ / 결과 숫자: ____ / 이후에도 고정됐는가: ____ / 남은 문제: ____’입니다. 같은 결과처럼 보여도 실제와 합성을 다른 줄에 남겨 보세요.

이제 한 판이 끝나는 이유와 끝난 뒤 하지 않을 일을 정했습니다. 다음 장에서는 끝내지 않고 쉬는 경우를 다룹니다. 일시정지는 결과 확정이 아니라 진행을 보관하는 일입니다. 무엇을 보관하고 무엇을 비울지 구별하면 멈춤과 다시 시작 이 더 예측 가능해집니다.

05 멈추고 다시 시작해도 같은 규칙으로

게임을 하다가 잠깐 다른 일을 봐야 할 때가 있습니다. 돌아왔더니 장애물에 닿아 끝나 있다면, 멈춤 버튼이 있어도 마음 놓고 쉬기 어렵습니다. 이 장에서는 쉬던 한 판을 이어가는 일과 새 한 판을 만드는 일을 구분합니다.

일시정지는 지금 점수와 장애물 위치를 보관한 채 진행을 멈춥니다. 다시 시작은 그 기록을 버리고 새 한 판으로 바꿉니다. 두 버튼이 비슷한 코드를 쓸 수는 있어도 사용자에게 약속하는 결과는 다릅니다.



실제 Chrome 실행 화면 · 주요 동작은 본문에서 설명

실제 Chrome 조작에서 140점·남은 15.5초일 때 멈춤 화면을 얻었습니다. 나중에 확인해도 같은 숫자였고 직접 계속하기를 눌러 다시 움직였습니다. 01장에서 본 장면을 이제는 ‘무엇이 유지됐고 무엇이 비워졌는가’로 읽어 봅시다.

앞 장에서 이미 끝난 상태는 더 처리하지 않게 했습니다. 같은 생각을 일시정지에도 씁니다. 다만 끝난 결과를 만드는 대신, 현재 한 판을 그대로 두고 진행할 자격만 잠깐 꺼둡니다.

보관할 것과 비울 것을 나눕니다

멈출 때 점수와 남은 시간, 우주선과 장애물의 위치는 남겨 둡니다. 계속하기 뒤 같은 장면을 이어가려면 필요한 값입니다. 이 값까지 초기화하면 일시정지가 아니라 재시작이 됩니다.

반대로 누르고 있던 방향과 드래그 목표는 비웁니다. 손을 떼는 신호가 게임에 전달되지 않을 수도 있기 때문입니다. 오른쪽을 누르다가 다른 창으로 옮겼다면 돌아올 때 ‘오른쪽을 계속 누르는 중’이라고 기억해서는 안 됩니다.

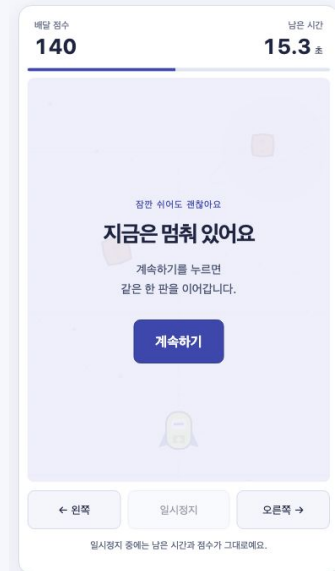
계속하기를 눌렀을 때의 시간 기준도 새로 잡습니다. 멈추기 전 시각과 돌아온 시각의 차이를 그대로 이동량에 넣으면 쉬었던 시간이 한꺼번에 적용될 수 있습니다. 우주선이 순간 이동한 것처럼 보인다면 그림보다 이 시간 기준을 먼저 살펴보세요.

이 설명을 코드 용어로 외울 필요는 없습니다. ‘한 판의 기록은 보관하고, 지금 손이 누르고 있다는 기억은 비우며, 재개 시각은 새로 잡는다’고 말하면 됩니다. 사람의 기대가 먼저이고 변수 이름은 그 기대를 옮기는 도구입니다.

멈췄는데 시간이 흐른다면

일시정지 안내가 보이는데 남은 시간이 줄어듭니다. 우주선이 멈췄고 버튼도 비활성으로 보이므로 처음에는 알아채기 어렵습니다. 하지만 쉬고 돌아온 사람은 같은 게임을 이어갈 수 없습니다. 화면을 덮는 일과 게임을 멈추는 일이 다르기 때문입니다.

비교를 위해 before.html에는 일부러 이 오류를 넣었습니다. ‘멈춘 장면 준비’로 진행 7.2초의 합성 조건을 만든 뒤, 멈춘 상태에서도 시간과 점수만 증가하게 했습니다. 실제 원작 게임에서 발견한 오류라고 소개하는 사례가 아닙니다.



의도적 오류판 · 합성 시작 조건에서 시간 흐름 관찰

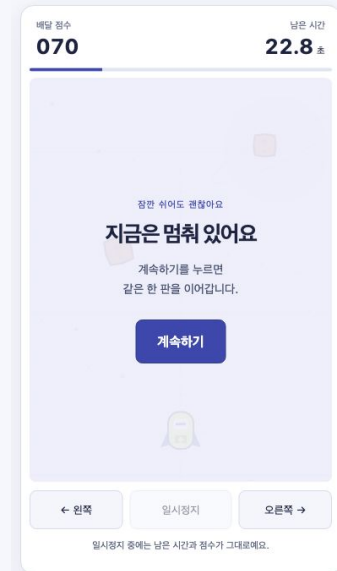
실제 관찰에서 멈춘 상태의 진행 시간은 7.4167초에서 14.3167초로 늘었고 점수는 70점에서 140점으로 바뀌었습니다. 장애물 위치는 그대로였습니다. 따라서 ‘모두 멈췄다’는 규칙을 어겼다고 판단할 수 있습니다. 이 숫자는 오류판 관찰값이지 플레이어의 실력이나 제품 성능 지표가 아닙니다.

AI에게는 ‘멈춘 안내가 보이지만 점수와 시간이 증가한다. 우주선뿐 아니라 시간·점수·장애물·충돌 처리를 함께 멈춰 달라’고 설명합니다. 기대한 결과와 실제 결과를 함께 적으면 그림만 바꾸는 잘못된 수정을 줄일 수 있습니다.

같은 조건으로 고친 결과 보기

정상판에서는 진행 중이 아닐 때 게임 규칙을 더 처리하지 않습니다. 멈추면 화면 갱신 연결을 끊고 입력도 비웁니다. 계속하기를 누르면 이전 시각을 그대로 끌고 오지 않고 다시 기준을 잡아 이어 갑니다. 쉬고 있던 시간만큼 갑자기 움직이는 일을 막으려는 것입니다.

동일하게 진행 7.2초·70점의 합성 장면을 준비해 비교했습니다. 정상판은 나중에 다시 보아도 7.2초·70점이었고 장애물 위치도 같았습니다. 가상 상태를 만든 시험이므로 ‘실제 7.2초 플레이 성공’이라고 부르지는 않습니다.



정상판 · 같은 합성 시작 조건에서 값 유지

합성 함수 시험에서는 멈춘 상태에서 이동 입력을 넣어 60번 처리한 뒤에도 상태가 같은지 보았습니다. 이 시험만으로 실제 버튼과 탭 전환까지 확인했다고 할 수는 없습니다. 실제 브라우저에서 멈추고 돌아오는 행동은 다음으로 따로 확인합니다.

이 수정의 핵심은 코드를 많이 추가하는 것이 아니라 ‘지금 진행해도 되는가’를 한곳에서 결정하는 것입니다. 그 결정이 맞으면 화면을 가리는 처리와 게임 규칙을 멈추는 처리가 같은 상태를 바라보게 됩니다.

탭을 바꿨다가 돌아왔을 때

사용자가 다른 탭으로 옮기면 게임을 보지 않는 동안 시간이 흐르지 않도록 멈춥니다. 다시 게임 탭을 골라도 자동으로 재개하지 않습니다. 돌아온 사람이 장애물 위치를 보고 직접 계속하기를 누를 시간을 남기는 정책입니다.

브라우저는 문서의 보임 상태가 바뀔 때 `visibilitychange`라는 알림을 보냅니다. 예제는 숨겨졌을 때 멈춤을 처리하고, 창의 초점이 빠지는 경우도 함께 살펴봅니다. 알림이 있다는 사실과 우리 게임이 올바르게 멈춘다는 사실은 별도로 확인해야 합니다. [MDN visibilitychange](#), 확인 2026-10-18.

제작 중 도구의 디버깅 연결 상태에서는 탭을 바꿔도 페이지가 계속 보이는 것으로 관찰했습니다. 그 결과를 통과 근거로 쓰지 않고, 해당 도구 연결만 종료한 일반 Chrome에서 실제 탭을 선택해 확인했습니다. 복귀 뒤 게임이 멈춰 있었고 직접 계속하기를 눌러야 재개했습니다.

이 사례는 게임 자체 오류와 시험 환경의 차이를 구분하는 법도 보여 줍니다. 새 프로그램을 설치하거나 보안 설정을 바꾸기 전에, 어떤 환경에서 어떤 신호가 실제로 발생했는지부터 살펴보세요. 관찰하지 않은 휴대폰이나 다른 브라우저까지 통과로 넓히지는 않습니다.

손을 떼지 못한 상황까지 생각합니다

버튼을 누른 뒤 손가락이 화면 밖으로 나가거나 다른 동작에 끊길 수 있습니다. 이런 경우에는 보통의 ‘놓기’ 신호와 별도로 입력 취소를 처리해야 합니다. 오른쪽으로 가라는 기억이 남는 일을 막기 위해서입니다.

브라우저의 `pointercancel`은 포인터 동작이 이어지지 못할 때 전달될 수 있는 알림입니다. 예제는 놓기와 취소에서 해당 입력을 비우도록 했습니다. [MDN pointercancel](#), 확인 2026-10-18.

여기서 실제로 확인한 것은 키를 누르고 떼기, 화면 끌기와 버튼 놓기, 그리고 별도 합성 취소 함수의 입력 해제입니다. 실제 휴대폰에서 알림이나 제스처로 터치가 중단되는 상황까지 해 봤다는 뜻은 아닙니다. 같은 ‘취소 처리’라는 말 안에서도 확인 범위를 나눠 적어야 합니다.

초보 독자는 이런 알림 이름을 외우지 않아도 됩니다. ‘손을 떼거나 입력이 끊기거나 창을 나가면 이동 기억을 비워 줘’라고 요청하고, 어떤 행동까지 실제로 시험했는지 설명을 받아 보세요. 눈앞의 그림보다 놓친 입력이 문제일 수 있습니다.

다시 시작은 새 한 판 하나만

몇 번 다시 시작했더니 게임이 빨라지는 문제를 상상해 봅시다. 버튼을 누를 때마다 화면 갱신 연결을 새로 더하고 이전 연결을 정리하지 않으면 여러 진행이 겹칠 수 있습니다. 재시작은 새 연결을 쌓는 일이 아니라 새 상태를 준비하는 일이어야 합니다.

예제는 기존 화면 갱신 예약을 취소하고 입력을 비운 뒤, 처음 상태를 만듭니다. 시간 0초·점수 0점·장애물 없음에서 출발하고 진행을 한 번 연결합니다. 일시 정지한 한 판을 재개하는 버튼과 구분됩니다.

실제 UI의 합성 충돌·재시작을 세 차례 이어 확인했습니다. 예제 안에서 기록한 동시 화면 갱신 예약의 최댓값은 1이었습니다. 이 값은 교육용 계측 범위의 관찰이며 브라우저 전체의 성능이나 모든 이벤트 등록 상태를 보증하는 숫자는 아닙니다.

재시작을 확인할 때는 첫 화면 숫자만 보지 말고 여러 번 해 본 뒤에도 같은 규칙인지 봅니다. 이전 장애물이 남지 않는지, 점수가 이어지지 않는지, 입력을 놓았는데 움직이지 않는지도 함께 살펴보세요. 문제가 생기면 직전 정상 파일로 돌아갈 수 있도록 수정 전 파일을 남깁니다.

쉬었다 이어가기와 새로 시작하기

이번 장의 작업물은 ‘멈춤·재개 기록’입니다. 채운 예는 ‘140점·남은 15.5초에서 수동 정지 / 같은 숫자 유지 / 직접 계속하기 뒤 이동 재개’입니다. 별도 줄에는 ‘일반 Chrome 탭 전환 뒤 정지 / 돌아와도 자동 재개 없음’이라고 적습니다.

내 양식에는 ‘멈춤 이유: ____ / 남겨 둔 값: ____ / 비운 입력: ____ / 돌아온 뒤 표시: ____ / 재개 조건: ____ / 새 한 판의 처음 값: ____’을 남겨 보세요. 실제 조작인지 합성 상태인지도 덧붙입니다.

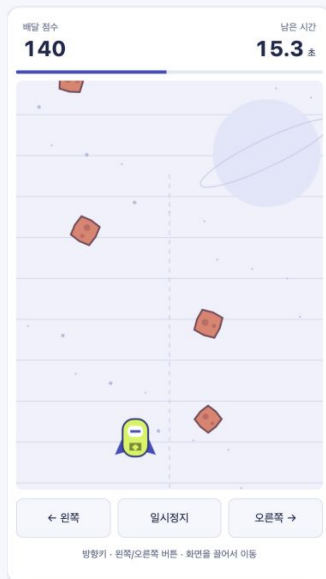
확인한 결과와 기대가 다르면 AI에게 그 줄만 전달합니다. 멈췄을 때 0점이 됐다면 보관해야 할 기록을 지운 문제이고, 재시작 뒤 140점이 남았다면 초기화해야 할 기록을 남긴 문제입니다. 서로 반대의 수정이 필요하므로 ‘초기화가 이상하다’고만 말하면 충분하지 않습니다.

이제 한 판이 시작하고, 움직이고, 끝나고, 쉬었다 이어지고, 다시 시작됩니다. 마지막 장에서는 이런 동작을 보존한 채 보기 좋게 다듬습니다. 화면을 바꾸더라도 지금 확인한 약속들이 유지되는지가 기준입니다.

06 보기 좋게 고치고 내 게임으로 바꿉니다

버튼도 눌러고 한 판도 끝냅니다. 이제 색과 모양을 바꿔도 될까요? 가능합니다. 다만 ‘예쁘게 바꾸기’가 점수나 움직임까지 몰래 바꾸는 요청이 되지 않도록 나누어 진행합니다.

이번 장에서는 화면을 읽기 좋게 다듬는 일과 이동 속도 한 값을 바꾸는 일을 구분합니다. 먼저 무엇을 보고 눌러야 하는지 분명하게 만들고, 그 다음 기본판을 보존한 별도 비교판에서 속도만 바꿉니다.



실제 Chrome 실행 화면 · 주요 동작은 본문에서 설명

별빛 배달부는 밝은 바탕에 우주선과 장애물을 다른 모양으로 두고, 점수와 남은 시간을 위에 모았습니다. 시작·정지·끝난 이유는 글로도 보여 줍니다. 색을 구별하는 것만으로 중요한 상태를 알아내게 하지 않기 위해서입니다.

초보자가 첫 게임을 만들 때 멋진 효과를 많이 넣을 필요는 없습니다. 어디를 누르는지, 왜 멈췄는지, 다음에 무엇을 할 수 있는지 보인다면 이미 중요한 디자인을 한 것입니다. 이번에는 소리·화면 흔들림·새 단계를 더하지 않습니다.

같은 말이 같은 행동을 가리키게

‘시작’, ‘계속’, ‘다시’는 짧은 말이지만 서로 다른 약속입니다. 준비 화면의 배달 시작은 새 한 판을 열고, 멈춘 화면의 계속하기는 보관한 한 판을 이어갑니다. 끝난 화면의 다시 시작은 결과를 버리고 새로 시작합니다.

만약 멈춘 화면에도 ‘다시 시작’이라고 써 두면 사용자는 점수가 사라질까 망설일 수 있습니다. 반대로 끝난 화면의 버튼을 ‘계속’이라고 쓰면 같은 결과 뒤에서 이어지는 것으로 기대할 수 있습니다. 동작과 문구가 어긋나면 안내가 짧아도 어렵습니다.

AI에게 문구만 고칠 때는 어느 상태에서 보이는 말을 바꿀지 지정합니다. ‘멈춘 화면의 버튼만 계속하기로, 끝난 화면은 다시 시작으로 유지해 줘’처럼 말하세요. 코드 전체를 재구성할 이유가 없는 수정입니다.

수정 뒤에는 세 화면을 각각 열어 봅니다. 버튼에 써 있는 말과 실제 동작이 맞는지 확인해야 합니다. 이름이 좋아 보여도 눌렀을 때 반대 동작을 한다면 디자인은 아직 끝나지 않았습니다. 새 문구의 검수와 기존 규칙의 확인을 함께 하세요.

게임 그림은 조작을 돕습니다

우주선과 장애물이 비슷한 색과 모양이면 플레이어가 무엇을 피해야 하는지 늦게 알아챱니다. 배경 장식이 더 눈에 띄어도 마찬가지입니다. 이번 그림은 우주선을 밝은 라임색, 장애물을 붉은 다각형으로 나눴습니다.

이 그림들은 새 예제를 위해 코드로 그렸습니다. 외부 캐릭터나 원작 이미지를 가져오지 않았습니다. 내가 만든 그림을 쓰는 경우에도 원본과 수정본을 나눠 두면 어떤 크기에 맞춰 충돌 범위를 정했는지 추적하기 쉽습니다.

그림을 새로 바꾸고 싶다면 먼저 몸 전체의 폭과 가운데 위치를 확인하세요. 모양을 키워 놓고 03장의 경계와 04장의 닿는 범위를 그대로 쓰면 게임의 느낌도 달라집니다. 색만 바꾸는 일과 크기를 바꾸는 일은 영향 범위가 다릅니다.

이번 작은 변형에서는 그림을 바꾸지 않습니다. 처음 만든 모습을 보존하고 속도 하나만 비교하기로 했기 때문입니다. 이미지를 새로 생성하거나 외부 에셋을 쓰고 싶다면 사용 조건과 비용부터 별도로 확인하세요. 출처 한 줄을 적는 것만으로 이용 권한이 생기는 것은 아닙니다.

여러 값을 한꺼번에 바꾸지 않습니다

우주선은 더 느리게, 장애물은 더 빠르게, 판 길이는 더 짧게 바꿨다고 가정해 봅시다. 새 판이 어려워졌어도 무엇 때문인지 알기 어렵습니다. 결과가 달라졌다는 사실과 이유를 설명할 수 있다는 것은 다릅니다.

이번 비교판에서는 우주선의 이동 속도만 300에서 240으로 바꿉니다. 화면 배지에는 속도 비교판임을 표시합니다. 한 판 30초, 1초마다 10점, 장애물의 규칙과 좌우 경계는 그대로 둡니다. 원본 game-v1 폴더를 수정하지 않고 speed-v1에 별도 파일을 둡니다.

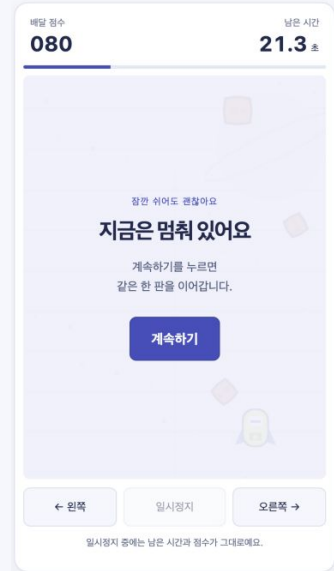
240이 초보자에게 더 좋다고 조사한 것은 아닙니다. 이동이 느려지면 미세한 조절이 편할 수도 있지만 장애물에서 벗어나기 어려울 수도 있습니다. 한 숫자의 차이를 관찰하려고 고른 교육용 설계값입니다.

독자가 자기 게임을 고칠 때도 ‘바꿀 값 하나 / 그대로 둘 규칙 / 비교할 행동’을 먼저 적어 보세요. 바뀐 파일을 설명할 수 없다면 아직 수정 범위가 너무 큰 것일 수 있습니다. 원본이 남아 있어야 차이를 다시 확인하고 돌아갈 수도 있습니다.

속도 한 값을 바꾼 비교판

실습의 examples 폴더에서 서버를 실행한 상태라면 <http://127.0.0.1:8767/speed-v1/> 을 엽니다. 기본판과 같은 게임이지만 화면 배지로 비교판을 구분할 수 있습니다. 처음에는 방향키나 버튼 한 방법만 사용해 보세요.

함수 시험에서는 장애물이 닿지 않는 같은 시작 조건을 만들었습니다. 가상 가로 위치 22에서 오른쪽 입력을 1초 동안 넣자 기본판은 322, 비교판은 262가 됐습니다. 각각 300과 240만큼 이동한 결과입니다. 두 판의 점수는 10점으로 같았습니다.



실제 비교판 버튼 이동 뒤 멈춤 · 322/262는 별도 합성 계산

이 숫자는 합성 입력을 작은 시간 간격 60번으로 처리한 계산 결과입니다. 실제로 사람 손이 정확히 1초 누른 기록도, 모든 기기의 움직임 측정값도 아닙니다. 실제 조작에서는 누른 시간과 관찰 시점이 달라지므로 같은 위치 숫자가 반드시 나올 필요는 없습니다.

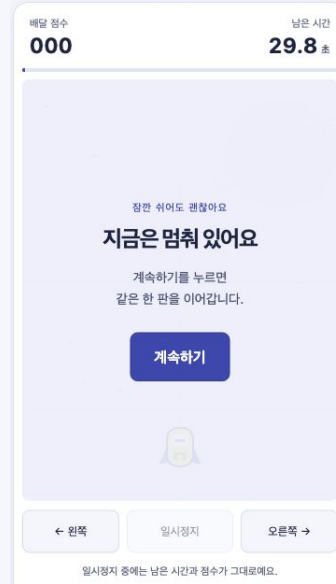
비교판은 실제 Chrome에서 버튼 이동·놓기·수동 정지도 확인했습니다. 정확히 1초 누르는 속도 측정은 아닙니다.

살펴볼 것은 방향, 손을 놓았을 때의 정지, 끝에서의 경계입니다. 숫자 하나를 바꾼 뒤에도 앞 장에서 확인한 약속이 유지되는지 보세요. 실행하지 않은 환경까지 확인했다고 적지 말고, 직접 본 것과 계산만 확인한 것을 나눠 기록합니다.

좋아 보인다는 말 뒤에 확인을 붙입니다

색과 간격을 바꿨더니 보기 좋아졌습니다. 하지만 작은 화면에서는 오른쪽 버튼이 가장자리 밖으로 밀릴 수 있습니다. 버튼 글자가 보이는 것과 그 버튼을 누를 수 있는 것은 다르므로 조작 영역까지 살펴야 합니다.

기본판은 390·360 너비의 브라우저 배치와 조작을 확인했습니다. 이 결과는 같은 기본판 파일의 해당 조건에만 적용됩니다. 그림의 크기나 버튼 간격, 새 문구를 바꾸면 영향받는 화면을 다시 확인해야 합니다.



실제 Chrome390×844 조건 · 실제 휴대폰 시험 아님

모든 시험을 매번 처음부터 반복할 필요는 없습니다. 파일이 그대로이고 확인하려는 조건도 같다면 기록을 재사용할 수 있습니다. 반대로 속도 비교판이 생겼는데 기본판의 실행 화면만 보고 비교판도 모두 통과했다고 해서는 안 됩니다.

특히 실제 휴대폰·다른 브라우저·처음 읽는 사람의 이해도는 별도 확인입니다. 브라우저 폭을 줄인 것으로 실제 기기 검수까지 끝내지 않습니다. 내가 만든 게임의 한계도 함께 적어 두면 다음 수정에서 무엇부터 볼지 정하기 쉬워집니다.

내 게임 변경 카드

이번 장의 작업물은 ‘변경 카드’입니다. 채운 예는 ‘목표: 이동 속도 하나의 차이를 보기 / 원본: game-v1 보존 / 수정본: speed-v1 / 변경: 300→240 / 유지: 시간·점수·장애물·경계·정지 / 확인: 같은 합성 1초 이동 비교’입니다.

내 양식은 ‘바꾸려는 이유: ____ / 보존할 원본: ____ / 새 폴더: ____ / 바꿀 값 하나: ____ / 그대로 둘 규칙: ____ / 실제 확인한 행동: ____ / 합성으로만 확인한 것: ____ / 되돌릴 파일: ____’입니다. 작업을 맡기기 전에 앞쪽을 채우고, 실행한 뒤 뒤쪽을 채워 보세요.

AI에게 요청할 때는 이 카드를 붙여도 됩니다. 결과를 받은 뒤에는 요청하지 않은 변경이 없는지 설명을 읽고, 파일을 열어 행동을 확인합니다. 만족스럽지 않다면 변화의 이유를 더 적거나 원본으로 돌아갑니다. 계속 덧고치는 것만이 진전은 아닙니다.

이 작은 습관은 더 큰 앱에서도 쓸 수 있습니다. 다만 이 책의 예제를 새 서비스 전체에 곧바로 적용할 수 있다고 보장하지는 않습니다. 여기서 배운 것은 한 번에 한 가지를 바꾸고, 약속한 동작을 증거로 확인하는 방법입니다.

처음부터 끝까지,작은 한 판

우리는 먼저 완성 예제를 한 판 해 보고,AI에게 설명할 규칙을 정했습니다. 움직임과 경계를 맞춘 다음에는 부딪혔을 때 결과를 한 번만 정했습니다. 쉬는 동안에는 한 판의 기록을 보관했고,새로 시작할 때는 이전 입력과 기록을 비웠습니다.

이 과정에서 ‘화면에 있다’와 ‘동작한다’를 계속 나눠 봤습니다. 일시정지 안내가 보인다고 시간이 멈춘 것은 아니었고,작은 화면을 캡처했다고 실제 휴대폰을 시험한 것도 아니었습니다. 실제·합성·미확인을 나눈 기록이 다음 행동을 정하는 데 도움이 됐습니다.

완성한 것은 계정도 순위도 소리도 없는 작은 교육용 게임입니다. 그 크기를 지켰기 때문에 무엇을 누르면 어떤 결과가 나오는지 설명할 수 있었습니다. 기능수보다 내가 확인할 수 있는 한 판을 만드는 것이 이 책의 목적입니다.

이제 새로운 기능을 더하기 전에 다른 사람에게 한 판을 부탁해 보세요. 어디를 눌러야 할지 몰랐는지,왜 끝났는지 이해했는지 물어봅니다. 아직 직접 관찰하지 않았다면 독자가 쉽게 이해했다고 단정하지 않습니다. 다음 시작점은 더 많은 명령이 아니라,눈앞의 한 사람에게서 얻는 구체적인 질문입니다.

실습 파일을 여는 법

실습 ZIP을 새 폴더에 풀면 examples 안에 game-v1과 speed-v1이 있습니다. 기본판은 별빛 배달부, 비교판은 우주선 이동 속도만 바꾼 별도 예제입니다. 먼저 기본판부터 여세요. 오류판 before.html은 정상 실행 파일이 아닙니다.

examples 폴더에서 기존 Python 3로 아래 명령을 실행합니다. 다른 프로그램이 같은 포트를 쓰고 있다면 임의 종료하지 말고 포트와 안내 주소를 함께 바꾸세요.

```
python3 -m http.server 8767 --bind 127.0.0.1
```

기본판은 <http://127.0.0.1:8767/game-v1/>, 속도 비교판은 <http://127.0.0.1:8767/speed-v1/> 입니다. 두 주소 모두 같은 컴퓨터의 실습 폴더를 가리킵니다. 끝내려면 서버를 실행한 명령 창에서 Ctrl+C를 누릅니다.

파일을 더블클릭하는 file: 주소는 이 모듈 예제의 기준 실행 경로가 아닙니다. 화면이 열리지 않으면 서버를 실행한 폴더와 주소부터 맞춰 보세요. 필요한 도구가 없을 때 새 설치를 자동 진행하는 절차는 이 실습에 포함하지 않습니다.

자료와 확인 범위

이 책은 AIToolsLab의 한 손 회피 게임 제작기를 참고해 입력·멈춤·끝나는 규칙을 새 교육 예제로 구성했습니다. 원작의 코드·캐릭터·배포 파일을 복제하지 않았습니다. 참고 제작기는 <https://aitoolslab.win/games/dont-get-poopod/>에 연결되어 있으며,이번 구성에서는 로컬 원문을 읽어 참고했습니다.

브라우저 화면 갱신의 설명은 [MDN requestAnimationFrame](#),보임 상태 변화는 [MDN visibilitychange](#),입력 취소는 [MDN pointercancel](#),로컬 파일 제공은 [Python http.server](#)를 확인했습니다. 확인일은 2026-10-18입니다.

기본 예제는 Chrome 실제 조작,함수 시험,합성 충돌·시간 종료,좁은 브라우저 화면과 추출본 실행으로 나누어 확인했습니다. 속도 비교의 322와 262는 같은 합성 1초 입력의 계산 결과입니다. 실제 손 입력의 정확한 도달점이라고 보장하지 않습니다.

실제 휴대폰·다른 브라우저·PDF 리더·초보 독자의 이해도는 별도 확인 대상입니다. 실제 30초 완주 기록과 합성 종료 화면도 구분했습니다. 확인한 범위를 넘는 성능·접근성·쉬운 사용의 보증을 하지 않습니다.

만든 자료와 이용 안내

제목은 《AI 게임 제작실》,부제는 ‘움직이고,피하고,다시 시작하는 작은 게임 만들기’,저자는 Coreline-AI입니다. 이 책의 우주선·장애물·도해·표지는 별도로 작성한 코드 기반 그림이며 제품 실행 화면과 구분해 표시했습니다.

실습에서 수정할 파일은 새 폴더에 보관하고 원본을 남겨 두세요. AI에게 제공하는 오류 설명에는 계정 정보·개인 정보·비밀 키를 넣지 않습니다. 외부 업로드·계정 연결·결제·공개 배포는 이 책의 로컬 실행 과정과 별개로 결정할 일입니다.

다른 사람의 그림·글·음원을 추가할 때는 그 자료의 이용 범위를 확인해야 합니다. 출처를 적는 일과 사용 조건을 확인하는 일은 구분하세요. 새 그림을 만들거나 수정한 경우에도 사용한 자료와 파일을 남겨 두면 나중에 범위를 확인하기 쉽습니다.

PDF에 사용한 글꼴은 별도 보관한 AIWorkSans 두 파일이며 관련 SIL Open Font License 고지를 실습 안내 자료와 함께 둡니다. 글꼴의 이용 고지가 책·이미지·원작 게임 전체의 권리를 대신 설명하지는 않습니다. 실습의 가상 점수와 장애물은 교육용 규칙이지 경제적 보상이나 실제 성과가 아닙니다.

마지막에 남겨 둘 질문

책을 덮기 전에 완성한 한 판을 다시 열어 보세요. 시작하기 전에는 숫자가 움직이지 않는지, 누르고 놓는 행동이 맞는지, 끝난 뒤에는 결과가 고정되는지 직접 살펴봅니다. 정답 문장을 외우는 것보다 눈으로 확인하는 일이 중요합니다.

그다음 자신의 말로 설명해 봅니다. ‘내가 만든 것은 무엇인가?’ 한 판은 언제 끝나는가? ‘잠깐 쉬면 무엇이 남고 무엇이 비워지는가?’ ‘숫자 하나를 바꾼 뒤 무엇을 다시 봤는가?’ 이 질문에 짧게 답할 수 있다면 다음 수정을 맡길 준비가 된 것입니다.

도움을 받을 때는 마지막으로 정상 동작한 파일과 바꾸려는 이유를 함께 보여주세요. AI의 긴 답보다 구체적인 한 행동이 문제를 잘 설명할 때가 많습니다. 무엇을 눌렀고 무엇을 기대했고 실제로 무엇이 달랐는지 적으면 됩니다.

첫 게임이 작아도 괜찮습니다. 내가 규칙을 정하고, 작게 바꾸고, 확인하고, 되돌릴 수 있다면 더 이상 결과를 바라보기만 하는 사람이 아닙니다. 다음 작업도 그 한 판에서 시작하세요. 새로운 기능을 많이 약속하기보다 지킬 수 있는 작은 약속을 하나씩 늘려 가면 됩니다.