기술

시뮬레이션 화면이 보여주는 것과 엔진이 계산하는 것은 같을까

방동걸 (주)웨이스 대표이사 2026년 8월 26일

저희는 공장을 짓거나 바꾸기 전에 컴퓨터 안에서 먼저 돌려 보는 가상공장 소프트웨어를 만들고 있습니다. 지난 글에서는 왜 만드는지를 적었는데, 이번 글은 그 반대쪽에 있는 이야기입니다. 만들고 있는 것을 이번 주에 외부 시점의 레드팀에 맡겨 검증했고, 거기서 나온 발견 중 가장 아팠던 것을 그대로 공개합니다. 웨이스가 VEXPLOR에 넣을 가상공장을 개발하는 중에 직접 겪은 일을 정리한 기록입니다.

검증은 관점이 다른 4개의 렌즈로 나눠 진행했습니다. 게임엔진 수준의 렌더링 품질, 시뮬레이션 계산의 정확성, 디지털 트윈으로서의 충실도, 그리고 성능과 확장성입니다. 각 렌즈가 코드를 실제로 읽으며 따졌고, 발견 32건 중 25건을 같은 날 고쳐서 반영했습니다. 그런데 며칠이 지나 남는 것은 개별 발견이 아니라 하나의 패턴입니다. 심각한 문제는 계산이 틀린 곳이 아니라, **화면이 말하는 것과 엔진이 실제로 계산하는 것이 갈라진 자리**에서 나왔습니다.

01

계산된 적 없는 숫자가 화면에 "감지"로 떠 있었습니다

가장 아픈 발견부터 말하겠습니다. 로봇 배치 화면에 "옆 로봇과 42mm 간섭이 감지되었습니다"라는 알림 패널이 있었습니다. 그럴듯해 보이지만, 이 42mm는 어떤 계산과도 연결돼 있지 않은 고정 문구였습니다. 초기에 화면 구성을 잡으면서 넣은 견본이 지워지지 않고 남아, 실제 판정처럼 보이고 있던 겁니다.

동작하지 않는 기능은 금방 들통납니다. 버튼을 눌렀는데 아무 일도 없으면 바로 신고가 들어옵니다. 그런데 그럴듯한 가짜 숫자는 신고가 들어오지 않습니다. 보는 사람이 믿고, 회의 자료에 들어가고, 판단의 근거가 됩니다. 검증 도구를 표방하는 소프트웨어에서 이것은 가장 나쁜 종류의 결함입니다. 저희가 이 도구의 제1원칙으로 "값을 만들지 않는다"를 걸어 두었는데, 정작 화면 한구석에서 그 원칙이 깨져 있었습니다.

수리는 패널을 지우는 것으로 끝내지 않았습니다. 그 자리에 진짜 계산을 배선했습니다. 로봇 팔이 교시 경로를 따라 움직일 때의 스윕 전체를 이웃 설비와 대조해 최소 여유를 실측하고, 그 실측값만 화면에 올라오게 했습니다.

02

화면과 엔진이 갈라진 자리가 모두 4곳이었습니다

같은 패턴으로 걸린 자리를 모으니 4곳이었습니다.

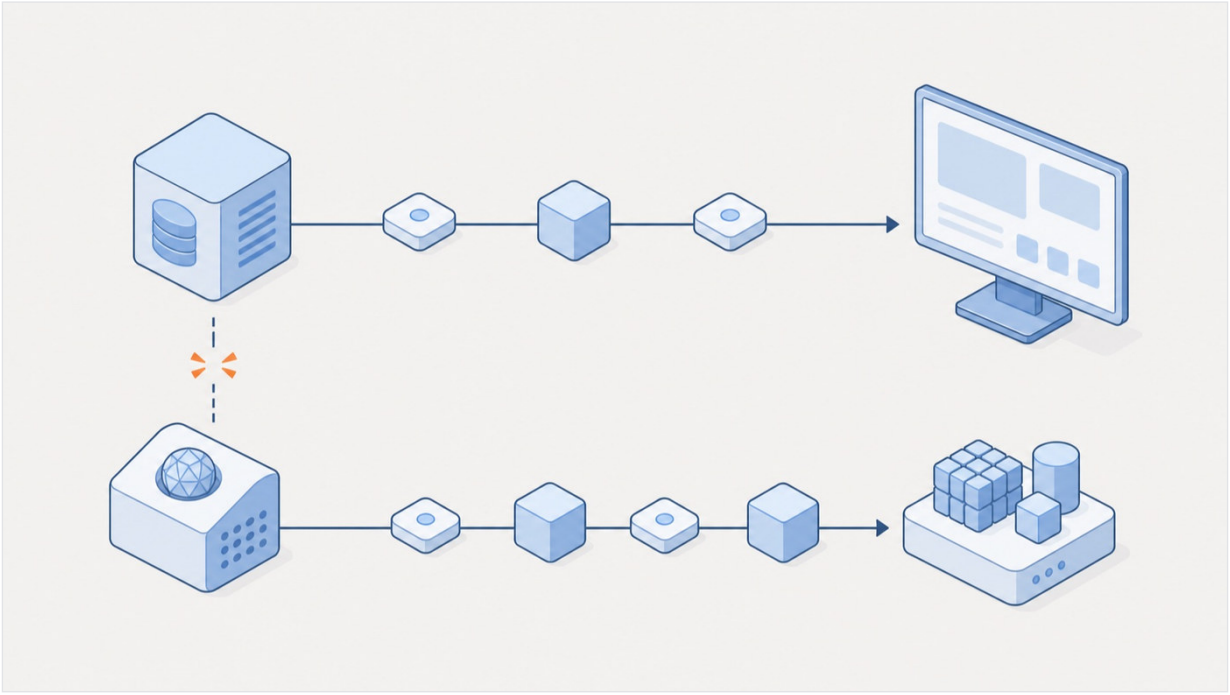
자리	화면이 말한 것	실제로 돌던 것
간섭 알림	42mm 간섭 감지	계산된 적 없음
버퍼 배분	배분이 반영됨	엔진이 무시
설비 회전	회전된 배치	판정은 회전 무시
성능 예산	상한을 지킴	검사가 죽어 있음

한 곳씩 보겠습니다. 버퍼 배분은 구간마다 재공 상한을 나눠 주는 기능인데, 화면에는 배분이 반영된 것으로 표시되면서 정작 화면의 시간을 진행시키는 엔진은 그 배분을 읽지 않고 있었습니다. 배분을 바꿔도 결과가 그대로인, 돌리나 마나 한 조절 손잡이였던 셈입니다.

이 결함이 생긴 경위가 전형적이라 덧붙입니다. 이 도구에는 계산 엔진이 둘 있습니다. 통계를 정확하게 뽑는 사건 단위 엔진과, 화면 재생을 부드럽게 돌리는 틱 단위 엔진입니다. 배분 기능은 사건 엔진에는 제대로 들어갔는데 틱 엔진에는 이식되지 않았고, 그래서 통계와 화면이 서로 다른 라인을 돌리고 있었습니다. 같은 규칙을 두 엔진이 각자 구현해야 하는 구조라면, 두 엔진이 같은 답을 내는지 맞춰 보는 동치 테스트로 묶어 두지 않는 한 언젠가는 갈라집니다. 이번 수리에서 그 동치 테스트까지 함께 넣었습니다.

설비 회전은 방향이 반대라 더 위험했습니다. 화면은 설비를 돌린 모습대로 그리는데, 간섭 판정은 회전하기 전의 외곽 상자로 재고 있었습니다. 도구가 겹침을 실제보다 없다고 말하는 쪽으로 틀리는 구조입니다. 검사 도구의 오류 중에서 가장 해로운 방향입니다.

성능 예산은 화면에 올릴 물량의 상한을 정해 둔 상수가 있었는데, 코드 전체에서 그 상수를 참조하는 곳이 한 군데도 없었습니다. 지키고 있다고 믿었던 상한이 사실은 어디에도 연결되지 않은 채 며칠을 지난 겁니다. 있다고 믿는 죽은 검사는, 없는 검사보다 나쁩니다. 없으면 만들기라도 하는데 있다고 믿으면 아무도 다시 보지 않기 때문입니다.



화면을 그리는 경로와 값을 계산하는 경로가 서로 다른 원본을 볼 때 갈라짐이 생긴다

03

왜 이런 일이 생기는지 따져 보면, 그리는 코드와 재는 코드가 다른 원본을 보고 있었습니다

네 자리의 공통 원인은 하나였습니다. 화면을 그리는 코드와 값을 재는 코드가 **각자 다른 원본을 보고 있었다**는 점입니다. 기능은 대개 화면에 먼저 생깁니다. 보여 주는 것이 빠르고, 계산은 나중에 따라옵니다. 그 사이를 잇는 계약이 없으면 둘은 각자 살게 되고, 시간이 지나면 반드시 다른 말을 합니다.

그래서 수리도 두 방향으로 했습니다.

- **원본을 하나로 합쳤다** — 안전 펜스의 배치 계획을 그리는 쪽과 세는 쪽이 같은 데이터를 보게 만들고, 상태 이름표처럼 여러 곳에서 쓰는 값은 한 곳에서만 정의하게 바꿨습니다.
- **갈라짐의 방향으로 회귀 테스트를 박았다** — 고친 자리마다 "화면과 판정이 같은 값을 봐야 한다"는 테스트를 추가해서, 같은 갈라짐이 다시 생기면 기계가 먼저 잡게 했습니다.

이번 검증에서 지적만 받은 것은 아닙니다. 같은 입력이면 같은 결과가 나오게 만든 재생 설계나, 시뮬레이션 결과가 대기행렬 이론의 계산값에 수렴하는지 맞춰 보는 검증 체계는 4개 렌즈 모두에게 잘돼 있다는 평가를 받았습니다. 그 대비가 오히려 요지를 선명하게 합니다. 계산의 뼈대가 좋아도, 화면과 엔진이 갈라진 자리 하나가 도구 전체의 신뢰를 깎습니다.

04

갈라짐만 있었던 것도 아닙니다 — 검사의 전제가 틀린 곳, 그림이 거짓말하는 곳도 있었습니다

패턴이 다른 발견 몇 개도 값이 컸습니다. 하나는 반대 방향의 결함, 즉 거짓 경보였습니다. 로봇 속도 검사가 관절만 움직이는 이동 구간에까지 직선 이동용 환산식을 들이대고 있었습니다. 그 환산은 팔 끝이 직선을 따라갈 때만 성립하는데, 팔이 제자리에서 방향을 바꾸는 구간에 적용하니 멀쩡한 경로에 속도 초과 경보가 떴습니다. 거짓 경보는 틀린 안심보다 안전해 보이지만, 몇 번 반복되면 사용자가 경보 자체를 무시하는 습관이 생깁니다. 그때부터는 진짜 경보도 같이 죽습니다. 수리는 검사를 그 전제가 성립하는 구간으로 한정하고, 거짓 경보가 다시 생기는 방향의 회귀 테스트를 붙이는 것이었습니다.

그림 자체가 정보를 지우고 있던 곳도 있었습니다. 화면을 밝게 보이려고 모든 재질에 자체 발광을 조금씩 줘 놨는데, 그 결과 그늘이 사라져 입체감이 죽어 있었습니다. 어두운 것을 조명으로 풀지 않고 물체를 빛나게 해서 푼, 일종의 노출 우회였습니다. 수리 방향은 발광을 실제로 빛나는 것들에게만 돌려주는 것이었습니다. 용접 아크와 신호 램프만 발광하게 하고, 도장 강판과 수지, 맨금속이 빛에 다르게 반응하도록 재질별 표면 응답을 나눴습니다. 시뮬레이션 화면에서 입체감은 장식이 아니라 설비 사이의 거리감을 읽는 정보라서, 이것도 충실도의 문제로 다뤘습니다.

30초짜리 작업에서 로봇 팔이 118ms 간격의 계단으로 뚝뚝 끊겨 움직이던 문제도 있었습니다. 진행도를 너무 거친 정밀도로 저장해서 생긴 양자화였고, 저장 정밀도를 한 단계 올려서 풀었습니다. 대기 중인 부품이 통계에는 있는데 화면과 요약판에서는 증발하던 문제, 기록 데이터가 상태를 한글 문자열로 수백만 칸 저장해서 메모리를 8배로 쓰던 문제도 이번에 함께 고쳤습니다. 하나하나는 작지만, 전부 "화면이 사실을 얼마나 있는 그대로 전하는가"라는 같은 질문에 속하는 발견들입니다.

05

고치는 것 못지않게, 고쳤다는 증거를 남기는 방식이 중요했습니다

반영 25건은 전부 같은 절차를 밟았습니다. 고치고, 그 결함이 다시 생기는 방향의 테스트를 붙이고, 형식 검사와 코드 검사를 0건으로 유지한 채 전체 테스트를 통과시키고, 마지막으로 브라우저에서 실제 화면을 열어 눈으로 확인했습니다. 이 마지막 단계를 강조하고 싶습니다. 화면과 엔진의 갈라짐이라는 결함은 성격상 자동 테스트만으로는 다 안 잡힙니다. 테스트는 코드에게 물어보는 것이고, 이 결함은 코드와 화면이 다른 말을 하는 문제라서, 결국 사람이 화면을 봐야 마무리가 됩니다.

- **고친 자리마다 회귀 테스트** — 같은 결함의 재발은 사람이 아니라 기계가 잡게 한다.
- **게이트는 0건 유지** — 형식 오류 0, 코드 검사 지적 0을 유지한 채로만 반영한다.
- **마지막은 실제 화면** — 렌더 결함과 갈라짐 결함은 브라우저에서 눈으로 확인해야 끝난다.

06

검증 도구를 만드는 쪽이 먼저 검증받아야 한다고 생각했습니다

이 소프트웨어는 남의 공장 라인을 짓기 전에 검증하겠다는 도구입니다. 그런 도구가 자기 화면에 계산된 적 없는 감지 문구를 띄우고 있었다는 것이 이번 검증의 요지이고, 그래서 이 글을 적습니다. 검증을 팔겠다는 쪽은 자기부터 검증받는 순서가 맞습니다.

이번에 반영하지 못한 발견 7건은 순서를 정해 목록으로 남겼습니다. 그중에는 통로 폭이나 안전 이격처럼 법규와 표준에서 인용해야 하는 값을 어떻게 다룰지 같은, 기능이 아니라 제품의 태도를 정하는 항목도 있습니다. 지어내면 안 되는 값은 출처와 함께 인용해야 한다는 원칙만 정해 두었고, 구현은 다음 순서입니다. 재미있는 것은 잔여 목록에 이번 패턴의 거울상도 있다는 점입니다. 무인 운반차의 가감속 계산은 엔진에 이미 구현돼 있는데, 그 값을 넣을 입력칸이 화면에 없어서 아무도 못 쓰고 있었습니다. 화면에는 있는데 엔진에 없는 것과, 엔진에는 있는데 화면에 없는 것. 둘 다 같은 병의 다른 증상입니다.



서로 다른 관점의 렌즈 4개가 같은 코드를 나눠 들여다보는 검증 구성

마지막으로 시제를 분명히 해 두겠습니다. 이 소프트웨어는 만들고 있는 중이고, 여기 적은 것은 완성 보고가 아니라 만드는 과정의 기록입니다. 그리고 시뮬레이션 안의 어떤 계산도 실제 설비의 시운전을 대체하지 않습니다. 그 한계를 화면에 문장으로 표시하는 것까지가, 이번 주에 저희가 고친 것들의 공통 방향이었습니다.

WACE(웨이스)는 공장이 스스로 판단하고 멈추지 않게 하는 제조 AI 운영체제 VEXPLOR를 만드는 회사다.

원문 blog.vexplor.com/post/when-screen-and-engine-disagree
발행 2026년 8월 26일 · 방동걸 (주)웨이스 대표이사

회사 주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 · 사업자등록번호 439-87-02852
주소 (18462) 경기 화성시 동탄구 동탄영천로 150, 에이동(A동) 1702-1708호 (영천동, 실리콘앨리동탄)
연락처 전화 070-4730-0629 · contact@wace.me
홈페이지 wace.me · linkedin.com/company/wace-inc

원문 홈페이지 링크드인