PERSLIS RESEARCH

Rules at the Wheel

Tanks, language models and an honest loss

Rule pilots in Atari BattleZone and BZFlag, measured against simple baselines, three language-model drivers and the engine's own AI — and the replay that shows why the engine's AI wins

Perslis Research | September 2026

Research prototype · simulation and games · not a certified safety system

Abstract. We measure rule pilots (priority-ordered rules with no neural network in the loop that decides) in two tank games and report every comparison, including the one we lose. In Atari 2600 BattleZone, reading only the screen, the pilot averages 27,000 over ten seeds against 3,000 for random play and 0 without a trigger; a dodge rule built from 25 recorded deaths and 403 replayed shell events was rejected at verification (24,800 against 28,000). In BZFlag, a 3D free-for-all where every driver reads the same engine state, the headline ten-minute match (one match) ended: BZFlag's built-in AI 38–6, our rules 20–20, DeepSeek 10–18 at 1.0 s per decision, Claude 1–15 at 7.6 s per decision through a command-line interface (a transport limit, not a model measurement), llama3.2:3b 0–10. Without time pressure the rules pass 8 of 8 exam situations, Claude 6, DeepSeek 5. Over 38 counted matches (7 runs excluded with reasons) the engine's AI beat our rules in every equal head-to-head. Replays of the last two seconds before each death show why: in one match, 16 of our 27 replayed deaths were a dodge that flagged the fatal shot a median 1.2 s early, then let go of it a median 9 times at its 4 m corridor edge; 13 of the engine's 15 were shots in view under 0.5 s. A single-packet reading had called most of ours "dodged too late". A wider corridor, since promoted, still loses its one head-to-head match, now on offence.

1 Introduction

A real-time game is a cheap, repeatable and unforgiving place to ask what a decision-maker is worth. The tank does not wait for the driver to finish thinking, the score is kept by the game rather than by the experimenter, and every run can be recorded. This paper asks three questions of one kind of driver, a *rule pilot*: a short list of explicit rules, most urgent first, that turns what the pilot perceives into one action, with no neural network in the loop that decides.

Q1. Is the pilot better than doing nothing useful? We compare it with random play and with the same pilot minus its trigger.

Q2. Is it better than a language model handed the same facts? We give Claude, DeepSeek and a small local Llama the same situation, prompt, output schema and servo, and let each drive its own tank in the same match.

Q3. Is it better than the game's own AI, written and tuned by the game's developers? And if not, can we show why, from evidence rather than from a guess?

The answers are yes, yes, and *no*. The third answer is the most useful thing in the paper. In BZFlag, the engine's built-in AI beat our rules in each of the six matches with language models and in all nine matches in which the two met alone in equal numbers, and the first account our own tooling gave of that loss was

wrong. We report how it was wrong, the replay analysis that replaced it, and what the replays say: our pilot dies mostly to shots it saw coming for more than a second and started to dodge, then abandoned at the edge of its own dodge corridor; the engine's AI dies mostly to shots it could not have seen in time.

1.1 What we did

The first game is Atari 2600 BattleZone in the Arcade Learning Environment [4, 9]. The pilot reads two regions of the 210×160 screen with array operations and never reads the emulator's memory. The second is BZFlag [6], an open-source 3D tank game, in which each tank is its own game client with its own driver: our rules, three language models, and BZFlag's own AI. A small patch to the client hands each tank's controls to an external process and passes it one state packet per frame; the same packet feeds every driver we wrote, and BZFlag's AI runs its upstream code unchanged. Every match is recorded, and every excluded match is listed with its reason. The two games differ in what the pilot perceives: pixels in BattleZone, the engine's own state in BZFlag, where a separate image check, reported in its own paper [18], can only withhold a shot.

1.2 Contributions

C1. A pixel-only rule pilot with its baselines and a rejected rule (§3). On BattleZone the pilot scores 27,000 against 3,000 for random play and 0 without a trigger, over the same ten seeds. A dodge rule built from 25 death windows and 403 replayed shell events lost at verification, 24,800 against 28,000, and was rejected; we report the rejection as a result.

C2. A like-for-like arena for rules, language models and a game's own AI (§§4–5). Same facts, one prompt, one schema, one parser, one servo; failed calls are counted, never covered for; every match recorded; 38 counted matches and 7 excluded runs, each with a reason. The headline match is one match, and we say so wherever it is used.

C3. A replay-proven account of a loss (§6). Two seconds of packets before each death are replayed through every candidate dodge rule. In the observation match, 16 of our 27 replayed deaths were a dodge abandoned at its corridor edge; 13 of the engine's 15 were shots in view for under half a second. We show why a single-packet reading misattributed ours, and why the pilot's own learning loop did not find the cause earlier.

C4. A measured firing gate (§7). With the witness gate on, 0 of 75 shots in the focus match were blind; the gate's cost was one close encounter in which it held fire twelve times.

1.3 What we claim, and what we do not

We claim the measurements above, at the sample sizes stated beside each. We do not claim that rules are better than learned policies in general, that our pilot plays either game well, or that the language-model results measure the models: the only route to Claude available on the test machine was a command-line interface that starts afresh for every decision (§4.5), and the Llama tank ran a model of about three billion parameters locally on the same machine. Nothing here has been replicated by a third party. The learning loop in §6 was still running when the data were frozen; its state at the cutoff, including what is *not yet proven*, is what we report.

Read these three facts with every table. (1) The headline arena result is **one ten-minute match**. (2) Claude drove through the `claude -p` command-line interface because no API key was available on the machine. The interface starts afresh for every decision, so its 7.6 s per decision measures that transport as much as the model; **no faster Claude number is claimed**. (3) BZFlag's own AI, not our rules, is the strongest driver measured.

2 Terms, measures and the record

Rule pilot. A function from a perceived situation to one game action, written as an ordered list of rules. The first rule whose condition holds decides; each decision returns the action, the rule that fired and a one-line reason, so the record shows why every action was taken. The rules hold no fitted parameters. In BattleZone the situation is read from the screen (§3); in BZFlag it is the engine state packet that every driver receives (§4). "No neural network in the loop that decides" means exactly this: no learned component chooses or ranks actions. The language-model drivers are, of course, neural networks; they are what the rules are compared with. The witness check of §7 is learned, but it can only withhold a shot, never choose one.

Measures. BattleZone keeps its own score, which rises only when the player destroys something. In BZFlag every tank is an enemy of every other (free-for-all); we report kills and deaths per tank, their ratio (K/D), kills per tank-minute, and, for the learning loop, the net score (kills − deaths) per tank-minute that the loop itself optimises. *Decision time* for a model is the wall-clock time from sending its situation to receiving its reply, failed calls included.

The record. Every BZFlag match writes a spectator video, a scoreboard every 10 s, final results, and a black box per tank. The black box grew during the day: the first matches keep every model decision (situation, reply, latency) and a 1 Hz sample of every tank; later matches add each tank's own view, every rule decision, every shot and held shot, and, from 00:15 on 27 September, every death with the last two seconds of packets before it. We read only the JSON files; the videos were not used to derive any number. A match counts when it finished writing results, lasted at least 1.5 minutes, had at least two kinds of driver, and carries no `INVALID.md` file explaining why its premise broke. Excluded runs are listed in Table 7, never dropped silently.

Cutoff. The BZFlag learning loop (§6.7) was running for another session while this paper was written. We froze the data at the match that finished at **01:05:50 EDT on 27 September 2026** (run `20260927-010140`): 45 run directories from 18:11 on 26 September. Nothing recorded after that is used. The analysis scripts import the lanes' own analysis code at a pinned commit, so later edits to that code cannot change our numbers (Appendix A).

3 BattleZone from pixels

3.1 Setup

Atari 2600 BattleZone runs in the Arcade Learning Environment [4, 9], package `ale_py` 0.11.2, as the environment `ALE/BattleZone-v5`. The v5 environment repeats the previous action with probability 0.25 ("sticky actions" [9]), which keeps a fixed action sequence from replaying one game. The pilot decides once every 4 frames, 15 times per second of game time, and may use all 18 joystick actions. An episode ends at game over (five lives) or at 30,000 frames; the cap never bound. Each arm plays seeds 0–9, the same seeds for every arm. Two baselines bracket the pilot: *random*, a uniform choice among the 18 actions at every decision, and *never-fire*, the same pilot with its trigger removed. Because the game's score only rises when the player destroys something, never-fire must score exactly zero; it is in every table to show that the score is the pilot's own.

3.2 The eye

The pilot never reads the emulator's memory. It reads two regions of the raw 210 × 160 RGB frame with array operations (Table 1). Two facts were measured on the running game before any rule was written:

the radar sweep is a *dashed* ray (2-pixel dashes on alternate rows, held about 4 frames per angle), so a component touching the radar's centre, or two or more components on one ray, are the sweep and a lone dot is an enemy; and turning right moves both view objects and radar blips to the left, at about 0.9 pixels and about 1° per frame, which fixes the sign of every turn.

region	where (rows, columns)	what is read, and how
radar	y 3–35, x 74–95; centre (19, 84)	white components; drop those touching the centre and any two or more on one ray (within 9°); a lone dot is a blip: bearing (0 = ahead) and range fraction
ground band	y 96–138; $x \geq 8$	each row is one flat colour, so a pixel off its row's colour belongs to an object; 2-D connected components; a component of at most 16 px and 5 rows is a shell
gunsight	column $x = 78$	the aim error is an object's column offset from the sight

Table 1. The BattleZone eye. Columns 0–7 are masked (the 2600's horizontal-motion bar reads as an object). The mountains (y 62–95) are skipped because their silhouette mixes colours.

3.3 The pilot

Three rules, most urgent first. **ENGAGE**: an object of at least 20 pixels is in the ground band; turn toward it and fire when it is within 3 pixels of the sight (turn and fire together within 8). The 2600 needs the button released between shots, so the pilot fires at most every other decision. **HUNT**: a blip is on the radar (remembered for 8 decisions, because the sweep can hide it); drive at it when it is within 10° of ahead, turn in place when it is behind, and turn while driving otherwise. **SEARCH**: nothing is known; drive forward for 40 decisions of every 60 and sweep right for the other 20.

3.4 Results

arm (seeds 0–9)	mean score	sd	median	range	kills per game
random	3,000	3,590	1,500	0–12,000	0–6
never-fire	0	0	0	0–0	0
rules	**27,000**	8,380	23,500	15,000–44,000	15–35

Table 2. BattleZone, first evaluation: ten games per arm, the same seeds for every arm, sticky actions 0.25, one decision per 4 frames. Every game in every arm ended at game over. Rules against random: Welch $t = 8.3$ (12.2 d.f.).

The rule pilot scores a mean of **27,000** (sd 8,380) against **3,000** (sd 3,590) for random play and **0** for never-fire (Table 2, Figure 1). ENGAGE decided 69.8% of the pilot's decisions, HUNT 26.6% and SEARCH 3.6%. Every game still ended with all five lives lost, after 1.5 to 2.8 minutes of game time.

Context, not a comparison. Mnih et al. report BattleZone scores of 2,360 for random play, 37,800 for a professional human tester and 26,300 (±7,725) for DQN [10, Extended Data Table 2]. Their protocol differs from ours on every axis that matters: 30 evaluation episodes of up to five minutes, up to 30 no-op actions at the start, an ε-greedy policy with $\varepsilon = 0.05$, no sticky actions, and a random agent that acts at 10 Hz; the human score is the mean of about 20 five-minute episodes after about two hours of practice. Evaluation protocols on the ALE are known to move scores substantially [9]. Read loosely, the rules land in the range of the 2015 DQN agent and below the human reference; later deep reinforcement-learning agents exceed the human reference on all 57 ALE games [3]. The pilot is not competitive with the state of the art, and it is not meant to be.

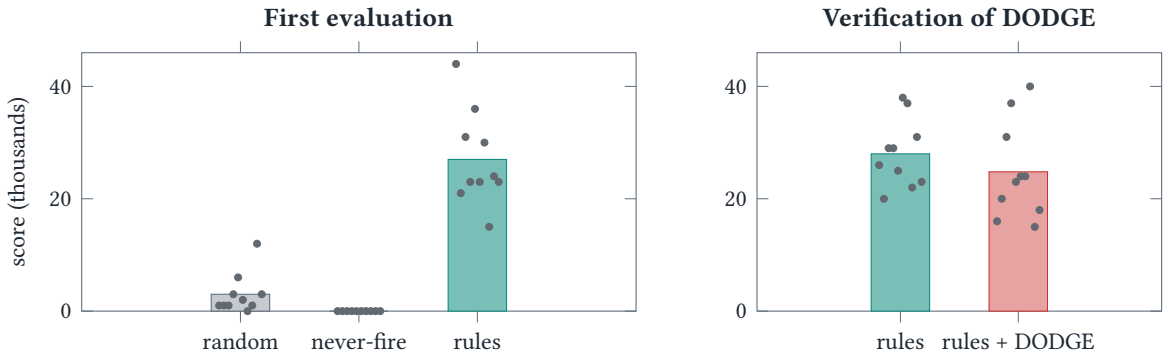

Figure 1. BattleZone mean score (bars) and every game (dots), seeds 0–9 in each arm. Left: the first evaluation. Right: the verification after the eye was improved, with and without the DODGE rule. The spread is wide: ten seeds per arm separate the pilot from random play clearly and do not separate the two arms on the right.

3.5 Learning from deaths, and a rule we rejected

Since every game ended at game over, the next step was the deaths. A forensic pass recorded the last 30 decisions and the last 12 frames before each of the 25 deaths in seeds 0–4 (Figure 2). Read by hand, roughly eight deaths show an enemy shell coming down the screen, roughly eight a shell inside the columns of the tank the pilot was shooting at, and roughly seven no visible cause; these counts are approximate readings of the frame strips, not a classifier's output. The second pattern exposed a defect: the eye grouped the ground band by column runs, so a shell under its own tank merged into the tank. The eye now uses 2-D components and flags shells.

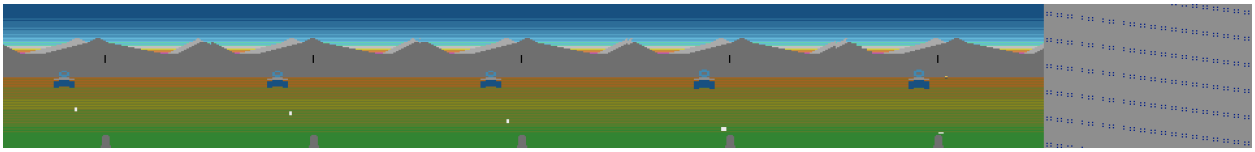

Figure 2. The last six of the twelve recorded frames before one death (seed 1, third life), oldest left, cropped to rows 40–149 and scaled 2× as recorded. The pilot is turning onto the blue tank left of the sight; the enemy's shell (white) comes down the screen; the right-most frame shows the game's hit effect. Frames from the lane's forensic record, not re-rendered.

To choose a dodge by experiment rather than intuition, an offline *dodge lab* snapshotted the emulator at every incoming shell and replayed nine candidate evasions from the same instant, each held for 6 decisions and followed by 25 decisions of the plain rules. The lab uses the emulator's save-state as an oracle, which the pilot never does at play time. It ran with sticky actions off so that a restored state replays exactly. Its first run was invalid: with sticky actions off the game is deterministic, and all seeds replayed one game; varied no-op starts (1 to 60 per seed) fixed that. Table 3 gives the conditional result that matters: of the 41 events in which the plain rules died, driving *forward* survived 9.

A DODGE rule (drive forward for 6 decisions when a shell is coming down the screen) was then verified on the normal protocol. With the improved eye, the pilot scored **28,000** (sd 6,055) without DODGE and **24,800** (sd 8,600) with it (Figure 1, right). Per seed, DODGE scored higher in 3, lower in 6 and tied in 1; the paired difference, −3,200, gives $t(9) = -0.94$. The loss is not significant at ten seeds, but there is no sign of a gain, and DODGE took 21.8% of all decisions away from aiming. **The rule was rejected**; the pilot ships without it, and the arm stays in the harness so the measurement can be repeated. The eye fix alone moved 27,000 to 28,000, which is within noise (Welch $t = 0.31$). Every game in both arms still ended at game over.

evasion held for 6 decisions	survived, all 403 events	saved, of the 41 the rules lost
forward	369 (91.6%)	**9**
reverse	356	7
turn toward the shell	360	4
forward + turn toward	362	4
none (the plain rules)	362 (89.8%)	0
turn away; reverse + either turn; forward + turn away	348–351	0

Table 3. The dodge lab (offline, emulator save-states, sticky actions off, 403 incoming-shell events over varied starts).

What the rejection shows

A rule can be justified by its own evidence and still lose. Forward survived more shells than any other evasion, and still cost more score than it saved, because the evidence measured survival and the game measures destruction. The same lesson, that a locally correct rule can be globally wrong, is the fail-first paper's Freeway result [15]; here the verification step caught it before the rule shipped.

4 The BZFlag arena

4.1 The game and the server

BZFlag is an open-source multiplayer 3D tank game [6]; we built it from source at upstream commit `4299415`. The server ran free-for-all with up to two shots in flight per tank (`-ms 2`) and a generated world at building density 3. With no world file, the server generates a random world seeded from the clock, so **every match is played on a new random layout** of boxes and pyramids (teleporters were not enabled); within a match every tank shares it. Engine defaults set the physics every driver lives with: shots fly at 100 m/s and expire at 350 m (so a tank's reload is 3.5 s), and a tank is 6.0 m long, 2.8 m wide, moves at up to 25 m/s and turns at most 45° per second.

4.2 A tap in the client, and the engine's AI untouched

A small patch (`FloorLink`), enabled only by environment variables, makes the client's autopilot hand its decision to an external process. On every frame the client sends one JSON packet with the tank's own pose, the enemy tanks with their distance, bearing and clear line of sight, the shots in flight within 200 m and within a tank's height, and the open distance along eight bearings; it applies the newest command it has received (turn rate, speed, fire). If no command arrives for one second the client's own AI takes over, except in the language-model tanks, which are set to stop instead, so that BZFlag's AI can never quietly drive a model's tank. BZFlag's own AI is the upstream `AutoPilot` code, unchanged: the patch adds a five-line hook at the top of it and nothing else. That AI dodges the worst shot within 100 m flying at it within about 16°, weaves toward its target and backs off when within 50 m and ready to fire, leads its target by 0.3 s, and fires when its aim error is small and the line of sight is clear. It runs inside the client, on every frame, with no transport delay; our controllers answer over a local socket, so their command reaches the tank on a later frame.

4.3 The drivers

The rule pilot. Version 1, which played the model matches, has four rules, most urgent first. **DODGE**: a shot whose straight path passes within 4 m of the tank and reaches its closest approach within 1.2 s; drive across its path, on the side the tank is already on. **ENGAGE**: an enemy in clear line of sight within shot range; aim at where it will be when the shot arrives, fire when the lead point is within about a tank-width of the sight line, then swing 60° off the enemy's line for one second (a "juke"). **HUNT**: turn toward the

tank	driver	route	decides
`floor-rules`	our rule pilot (v1 in the model matches)	local UDP, every frame	≈100 per s
`claude-opus5`	Claude, model id `claude-opus-5`, effort "low"	`claude -p` command-line interface (no API key on the machine)	own pace
`deepseek-chat`	DeepSeek `deepseek-chat`	HTTPS API, JSON mode, temperature 0	own pace
`ollama-llama3b`	Llama 3.2, 3B (`llama3.2:3b`)	local Ollama, schema as output format, temperature 0	own pace
`bzflag-autopilot`	BZFlag's built-in AI	in the client, upstream code	every frame

Table 4. The *models* lineup: one tank per driver, all enemies of each other. Two other lineups contain no models: *eye-drill* (three rule tanks against three BZFlag AI tanks) and the learning loop's A/B lineup (two tanks on the current rules, two on a candidate, two BZFlag AI; §6.7).

nearest enemy and drive, detouring toward the most open bearing when a wall is within 12 m. **EXPLORE**: drive toward the most open bearing. Version 2 added an UNSTICK rule after DODGE (§5.5) and made the juke and the dodge prefer the side with open space. The learning loop's candidates are version 2 with one named behaviour changed (§6.7).

4.4 The fairness contract

- **Same facts.** Every driver we wrote reads the same packet. A model receives it as a compact situation report: up to five nearest enemies with distance, bearing and whether the shot is clear; up to five incoming shots with distance, bearing and whether each is flying toward the tank; the open space on eight bearings; and how long the tank has not moved. The rules read the same fields.
- **One interface for every model.** One system prompt, one JSON output schema (each provider's structured-output mode where it has one), one parser. The model chooses how far to turn, how fast to drive and whether to fire now, after the turn, or not at all.
- **A servo that only executes.** Between decisions a servo carries out the model's last command: it turns the tank by the amount the model asked for, relative to the heading the model saw, and fires once when that heading is reached if the model said "after the turn". It never picks a target and never dodges.
- **No cover for failure.** A failed, slow or unparseable call leaves the tank on its last command, and is counted.
- **No decisions on stale state.** A model is never asked about a situation older than 1.5 s. This guard was added after the first ten-minute match, in which the Claude tank's client left the game after about ten seconds and its driver went on "deciding" on the frozen last packet for ten minutes (Table 7).
- **The witness gate binds only the drivers we run.** When witness firing is on (§7), our rules and the model tanks fire only on an enemy the eye confirms; BZFlag's AI never consults it.

4.5 The Claude transport

Claude's arena numbers are transport-bound. The test machine had no Anthropic API credentials, so every Claude decision ran the logged-in `claude` command-line interface in print mode (`claude -p`), from an empty directory, with tools disabled and no session persistence. Every call starts the interface afresh. The driver contains the official-SDK path (structured output, effort setting), and it would run automatically if an API key were present, but it **was never exercised**. The same model's median decision time was 24.0 s in the driving exam, which used the interface's JSON-schema option, and 7.5–9.5 s in the four longer arena matches, which did not. We therefore report Claude's arena results as results of *Claude through this interface*, and claim no number for Claude through its API.

5 Results in the arena

5.1 The headline match

The headline is match `20260926-183228`: ten minutes (9.92 played), five tanks, one per driver, started at 18:32:35 on 26 September. All five tanks were present throughout (60,140–68,231 state packets each, no client restarts). Every kill in the match is someone's death: 69 of each.

tank	kills	deaths	K/D	kills/min	decisions	median decision time	failed calls
BZFlag AI	**38**	6	6.33	3.83	every frame	in the client	–
our rules (v1)	**20**	20	1.00	2.02	60,140	≈0.01 s (every frame)	–
DeepSeek	**10**	18	0.56	1.01	415	1.01 s	2 (+12 unparseable)
Claude (CLI)	**1**	15	0.07	0.10	58	7.55 s	1
llama3.2:3b	**0**	10	0.00	0.00	213	2.55 s	0

Table 5. The headline match, **one ten-minute match** ($n = 1$). Median decision time is the lane's upper median of each model's decision times, failed calls included. Spread of decision times (10th–90th percentile): Claude 6.6–17.5 s (maximum 61.1 s), DeepSeek 0.82–1.61 s, llama3.2:3b 2.18–3.64 s. Share of a model's commands that asked to fire: Claude 55/58, DeepSeek 393/415, llama3.2:3b 213/213.

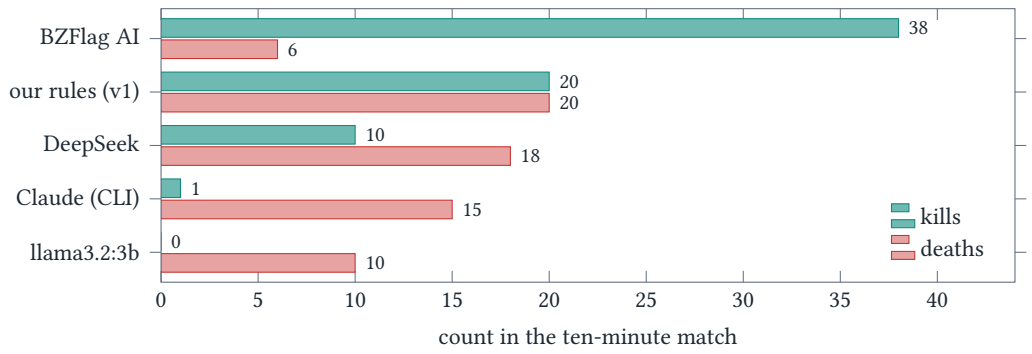

Figure 3. Kills and deaths per tank in the headline match (one match). The ordering is clear; the ratios are one sample, not a distribution.

The ordering in Table 5 and Figure 3 is the result: BZFlag's AI first by a wide margin, our rules second, then DeepSeek, Claude and llama3.2:3b. Our rules beat every language model and were themselves beaten by the engine's AI, which killed almost twice as often and died less than a third as often.

5.2 What a decision costs

Every model decided on its own clock. In the headline match the rules made 60,140 decisions, DeepSeek 415, llama3.2:3b 213 and Claude 58: the rules decided about 145 times as often as the fastest model. Figure 4 places every tank of every counted model match by how often it decided and how often it killed. The clusters did not move between matches: Claude's median decision time was 7.53–9.47 s in the four longer matches (5.9–9.9 minutes played; 12.78 and 16.29 s in the two short early ones), DeepSeek's 1.01–1.12 s (2.97 and 5.29 s), llama3.2:3b's 2.55–3.96 s (1.66 and 2.36 s).

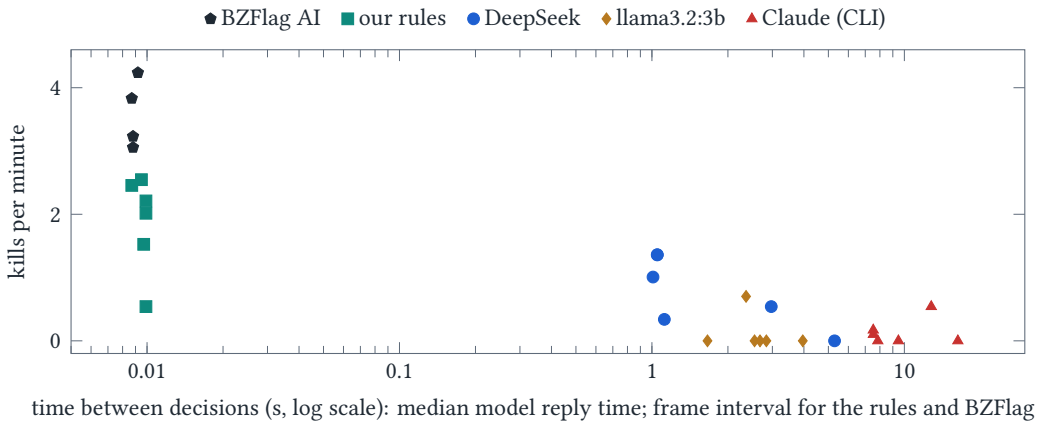

Figure 4. Every tank in the six counted model matches (one point per tank per match; BZFlag AI's frame interval was not recorded in the first two). Rules and BZFlag's AI decide every frame, about a hundred times a second; the models decide once every one to sixteen seconds. The first two matches were short (1.9 and 2.9 minutes).

5.3 The exam: the same models without time pressure

The arena mixes judgement with speed. The driving exam separates them: eight hand-built situations, each a real state packet with an objectively correct answer (enemy to the left; to the right; dead ahead; a shot from the side; a shot head-on; an enemy behind a building with a wall ahead; nothing around; an enemy directly behind), answered without any clock. A model commands a turn *amount*, which the servo carries out blind until its next decision, so the exam grades the amount; the rules turn continuously until lined up, so for them only the direction is graded.

driver	correct (of 8)	median decision time
our rules	**8**	instant
Claude `claude-opus-5` (CLI, JSON-schema option)	**6**	24.0 s
DeepSeek `deepseek-chat`	**5**	6.2 s
Llama `llama3.2:3b`	**4**	1.45 s
Qwen `qwen2.5-coder:7b`	3	3.0 s
Mistral `mistral:7b-instruct`	2	3.7 s

Table 6. The driving exam (one attempt per situation per driver, 48 answers in all). Claude missed the side shot (it turned 90° instead of crossing the shot's path) and the enemy behind a building (it turned toward the hidden enemy and fired into the building). DeepSeek missed both shots and the building.

Claude was the best model on the exam (6 of 8) and the second-worst in the arena. One caution about the small local models: llama3.2:3b gave nearly the same answer to all eight situations (fire after the turn, half speed, a right turn of 10° to 30°), and Mistral always turned right; llama's four passes come from graders that this near-constant answer happens to satisfy. The exam is small, and it measures only

whether a driver knows what to do.

5.4 Over time: every counted match

To the cutoff, 45 run directories were recorded. Thirty-eight matches count: 192 minutes of play, six with the models lineup and thirty-two with rules and BZFlag's AI only. Seven runs are excluded (Table 7).

run	played	why excluded
`20260926-182000`	9.9 min	`INVALID.md`: the Claude tank's client left about 10 s in (the run's note names a crash in macOS's OpenGL-to-Metal layer as the likely cause); its driver went on deciding on the frozen last packet for ten minutes. Led to client supervision and the 1.5 s stale-state guard.
`20260926-191409`	0.3 min	shorter than 1.5 minutes
`20260926-192915`	9.9 min	`INVALID.md`: the eye's training ran inside the process that hosts the controllers and starved them (our rules went 9–21, against 15–10 and 13–11 in the two matches before); training moved to its own process afterwards
`20260926-214830`	0.8 min	shorter than 1.5 minutes
`20260926-221854`	0.8 min	shorter than 1.5 minutes
`20260926-224233`	0.8 min	shorter than 1.5 minutes
`20260927-002751`	–	`INVALID.md`: aborted when the learning loop was handed to new code as the match began; no results

Table 7. Every excluded run to the cutoff, with its reason. No client was restarted in any recorded match after supervision was added.

	6 model matches (32.3 tank-min each)			all 38 counted matches		
driver family	kills–deaths	K/D	kills/min	kills–deaths	K/D	tank-min
BZFlag AI	113–18	6.28	3.50	1,278–730	1.75	442.1
our rules	65–57	1.14	2.01	629–1,024	0.61	442.1
candidate rules (A/B)	–	–	–	249–278	0.90	139.5
DeepSeek	29–55	0.53	0.90	29–55	0.53	32.3
Claude (CLI)	3–47	0.06	0.09	3–47	0.06	32.3
llama3.2:3b	2–32	0.06	0.06	2–32	0.06	32.3

Table 8. Pooled over counted matches. "Our rules" pools every version of the pilot and every lineup, so the right-hand columns are a record, not a comparison of equals; equal comparisons are in §6.

In the model matches our rules had a higher K/D than every model in five of six. The exception is the first match, 1.9 minutes long, in which the rules went 1–5 against DeepSeek's 1–1 and Claude's 1–2. BZFlag's AI had the highest K/D in all six model matches, and in every match with rules only up to the moment the learning loop promoted a new pilot (the first 29 counted matches). In the nine counted matches after that promotion, a rules family (the new pilot or its candidate) had a higher K/D than BZFlag's AI in five; all five are A/B matches, in which four of the six tanks are ours, so many of our kills and deaths are against each other. In the one equal head-to-head match of that period, BZFlag's AI still won (§6.7). Figure 5 shows every counted match.

5.5 The stuck rule

The first recorded matches logged no positions, and the recordings showed tanks pinned against walls. From then on every tank was handed the same two facts, computed identically for all drivers: how long it had not really moved (less than 1.5 m in 2 s) and how far it had moved. Pilot v2 added an UNSTICK rule: after 0.8 s stuck, pivot toward the more open side while driving for 1.2 s; if that does not free the tank,

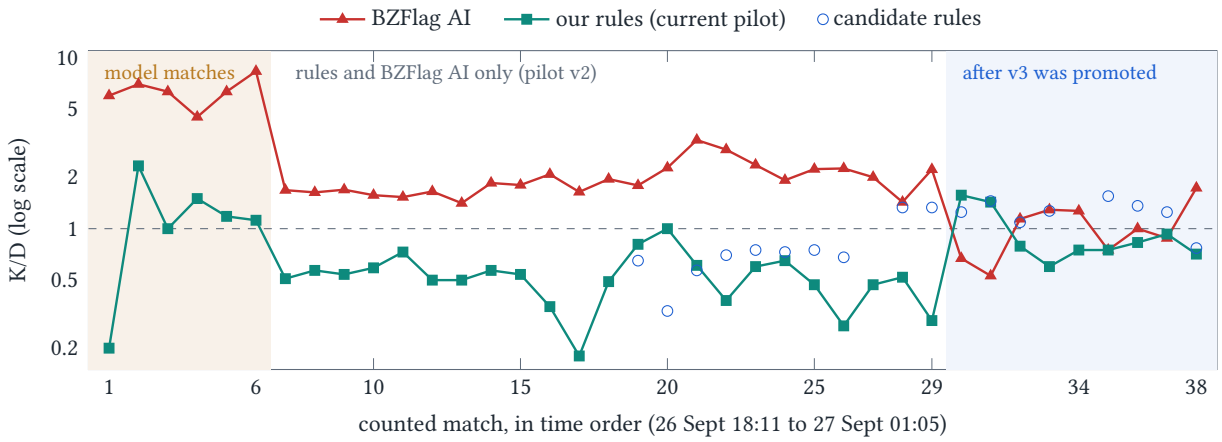

Figure 5. K/D of each driver family in each of the 38 counted matches. Conditions change along the axis: pilot v1 in matches 1–4, v2 from match 5, v3 from match 30; the eye ran from match 6 and gated firing in four matches; lineups and lengths vary (1.9–24.9 minutes). The dashed line is K/D = 1.

reverse for 1.0 s and pivot the *other* way, never repeating the move that failed. It follows the stuck rule of the VDSG runtime [16] (turn toward open space; never repeat the move that failed), without VDSG's calibrated escape length. In two six-minute model matches, the rules tank was pinned against a wall for **14.7%** of its alive time with v1 and **0.1%** with v2. The model tanks, handed the same facts, were pinned too in the v1 match: Claude 9.7%, llama3.2:3b 3.7%, DeepSeek 0.3%.

6 Why we lose, from the replays

6.1 The equal comparison

The pooled record of §5 mixes pilots and lineups. The equal comparison is the *head-to-head*: rule tanks that all fly one pilot version against the same number of BZFlag AI tanks, nothing else in the match, no witness gate and no human orders. Eight counted matches meet that definition for pilot v2, with 215.4 tank-minutes per side:

v2 head-to-head (8 matches)	kills–deaths	K/D	kills per min	deaths per min
BZFlag AI	645–389	**1.66**	2.99	1.81
our rules v2	301–554	**0.54**	1.40	2.57

BZFlag's AI had the higher K/D in each of the eight. The gap is on both halves of the score: our pilot killed about half as often and died about 1.4 times as often. The rest of this section is about the second half, because that is where the pilot's own learning loop looked first, and where it looked wrongly.

6.2 The first account: one packet per death

Before replays existed, each death kept the last state packet and the pilot's last few decisions. A rule-based classifier gave each death one cause from that snapshot: *dodged too late* if the last decision was a dodge, *shot while aiming* if it was ENGAGE lining up, and so on. Over the 139 deaths of pilot v2 in the loop's first seven matches, it called **88 (63%) "dodged too late"**. The loop acted on that reading and tried a longer dodge window (react at 1.8 s instead of 1.2 s). The A/B test rejected it (§6.7).

6.3 Replaying the last two seconds

From 00:15 on 27 September every death row carries the last two seconds of state packets, one per 40 ms. The replay reconstructs the fatal shot: the shot closest to hitting in the last packet (predicted to pass within

8 m of where the tank is going), followed back packet by packet along its straight path, so that a shot is never credited with a warning from before it was fired. Every candidate dodge rule is then re-run on every packet of that path: the pilot's own rule (closest approach to where the tank *is*, 4 m corridor, 1.2 s window), the same rule with 6 m and 8 m corridors, a moving-frame rule (closest approach to where the tank is *going*), and BZFlag's own cone. A rule warns *in time* if it first flagged the shot at least 0.5 s before the hit. Each death gets the first cause that matches:

cause	condition
shot up close	the fatal shot was in view for less than 0.5 s; no dodge could react
dodge failed	the pilot's own rule flagged it in time and kept flagging it; the tank did not get clear
dodge abandoned	the pilot's own rule flagged it in time, *then let go of it* at least once
leading shot	the own rule flagged it too late; a moving-frame rule would have flagged it in time
passed too close	the own rule flagged it too late; a wider corridor would have flagged it in time
unflagged shot	in view long enough, and no rule flagged it in time

6.4 What the replays show

The observation match `20260927-001524` put three v2 tanks against three BZFlag AI tanks for four minutes: ours 14–30, BZFlag's AI 32–16. Of our 30 deaths, 27 had a replay; of the engine's 16, 15 did (Figure 6).

- **Our v2: 16 dodge abandoned**, 9 leading shot, 1 shot up close, 1 dodge failed. The fatal shot had been in view for a median of **1.42 s**.
- **BZFlag's AI: 13 shot up close**, 1 leading shot, 1 dodge abandoned (judged against our v2 rule, since the engine's AI has its own). Its fatal shots had been in view for a median of **0.40 s**.

Our pilot died mostly to shots it saw coming and began to dodge; the engine's AI died mostly to shots nothing could have dodged. The pattern holds on every replayed death to the cutoff: 34 of the 67 replayed v2 deaths (51%) were dodge abandoned, and 124 of 157 replayed BZFlag AI deaths (79%) were shot up close, with a median time in view of 0.33 s.

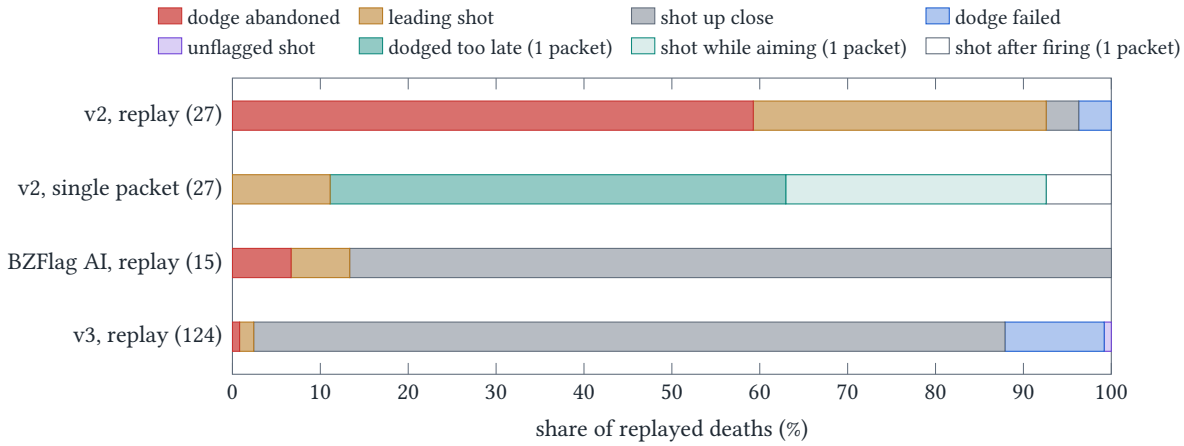

Figure 6. Causes of death. Top two bars: the same 27 deaths of our v2 in the observation match, read from two seconds of replay and from the last packet only. Third: BZFlag's AI in the same match. Bottom: every replayed death of pilot v3 to the cutoff (11 matches). Counts: v2 replay 16 dodge abandoned, 9 leading shot, 1 shot up close, 1 dodge failed; v2 single packet 14 dodged too late, 8 shot while aiming, 3 leading shot, 2 shot after firing; BZFlag AI 13 shot up close, 1 leading shot, 1 dodge abandoned; v3 106 shot up close, 14 dodge failed, 2 leading shot, 1 unflagged, 1 dodge abandoned.

6.5 The mechanism: a dodge that dithers at its own edge

A dodge abandoned is a switching rule chattering at its threshold. The pilot dodges by driving across the shot's path. As soon as the shot's predicted closest approach to the tank passes 4 m, the rule's condition is false and control falls to the next rule, ENGAGE, which turns and drives the tank toward its target. That motion carries the tank back toward the shot's path, the closest approach falls under 4 m, and DODGE fires again. Figure 7 shows one death packet by packet, and Table 9 the pilot's last decisions before it. The pilot's full decision log shows the same death at every decision: ENGAGE aimed at enemy 5 from 3.0 s before the hit; DODGE took over 1.17 s before it and held for 0.33 s; after that the two rules alternated ten more times in the last 0.84 s.

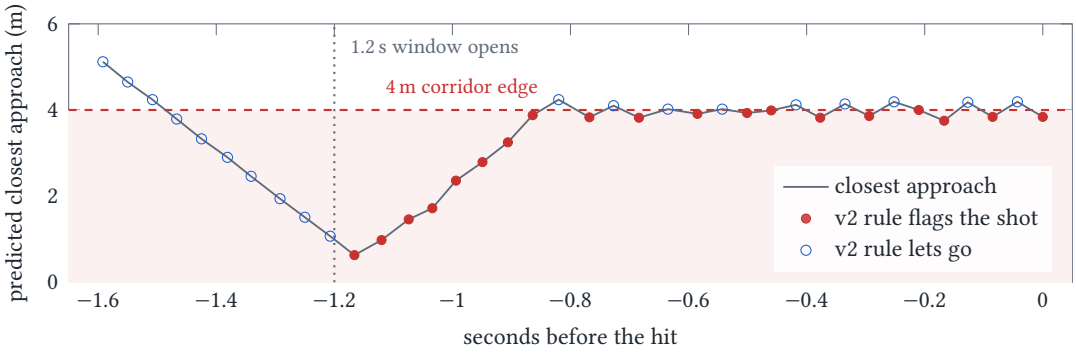

Figure 7. One dodge-abandoned death (tank `floor-rules`, its first death in the observation match), from the replay: the fatal shot's predicted closest approach to the tank at each packet (40 ms apart). The tank drives into the shot's path while aiming; the rule flags the shot 1.17 s before the hit, when it enters the 1.2 s window; the dodge carries the tank out to 3.9 m; from there the closest approach oscillates between 3.75 m and 4.24 m, the rule flagging and releasing on alternate packets (9 releases), until the hit.

ms	rule	the pilot's own reason
0	ENGAGE	enemy 5 168 m, lead −14.4° – aiming
39	DODGE	shot passes 4.0 m away in 0.27 s – cross its path
82	ENGAGE	enemy 5 166 m, lead −14.5° – aiming
143	DODGE	shot passes 4.0 m away in 0.16 s – cross its path
197	ENGAGE	enemy 5 164 m, lead −16.0° – aiming
238	DODGE	shot passes 4.0 m away in 0.08 s – cross its path
280	ENGAGE	enemy 5 162 m, lead −16.5° – aiming
322	DODGE	shot passes 3.8 m away in 0.01 s – cross its path

Table 9. The last eight decisions the black box kept before the death of Figure 7, as recorded. The rules alternate every 40–60 ms.

The lane's first account of these deaths said the target was usually the tank that fired, so that the pilot turned back into the shooter's line. The record does not support that as a rule. Placing the pilot's last target from the bearing and distance stored with each death, the target lay within 15° of the line the fatal shot came along in only 3 of the 16 dodge-abandoned deaths, and more than 90° from it, away from where the shot came from, in 12. (Enemy positions are those at the moment of death, up to 1.8 s after the shot was first seen, so the test is coarse.) More often the pilot was aiming at another tank while a shot from elsewhere closed in. What repeats in every one of these deaths is the release at the corridor edge.

Across the 16 dodge-abandoned deaths, the pilot's own rule first flagged the fatal shot a median of **1.17 s** before the hit (range 0.50–1.23 s) and let go of it a median of **9** times (range 3–13). The warning was early

enough; the pilot did not keep it. (Over all 27 replayed deaths, the same medians are 1.02 s and 6; the lane's own record quotes these two numbers.) A tank that turns at most 45° per second cannot afford a manoeuvre that undoes itself every few packets.

Chattering at a switching threshold is an old problem with an old remedy: hysteresis, as in the Schmitt trigger [12] and in hysteresis switching for supervisory control [7]. Switch *on* at one threshold and *off* only at a wider one. The pilot's gene `dodge_commit` is exactly that: once a dodge starts, it keeps its side until that shot has passed or its closest approach is twice the corridor. A wider corridor moves the edge outward without adding hysteresis.

6.6 Why the single-packet account was wrong

A last packet captures whichever phase of the dither happened to be current at the hit. Of the 16 dodge-abandoned deaths, the single-packet classifier called 9 "dodged too late" (the last decision was a dodge), 5 "shot while aiming" (it was ENGAGE) and 2 "leading shot". Its prescription, dodge earlier, was aimed at a pilot that had in fact flagged the shot more than a second early, and the A/B test rejected it.

The first replay version was not right either. It measured how long each rule had been flagging the shot *without a break* up to the hit. A dithering rule has almost no unbroken stretch at the end, so that version read the same 27 deaths as "passed too close" (15), leading shot (8), unflagged (2), shot up close (1) and dodge failed (1), and named a 6 m corridor as the fix. Counting the first flag and the releases separately, rather than only the final unbroken stretch, is what exposed the dither. The 6 m corridor was tested anyway; §6.7 reports what happened.

6.7 The learning loop, and why it did not learn sooner

The pilot has a learning loop in the fail-first shape of [15]: observe deaths, explain them, change one named behaviour (a *gene*), verify, keep or reject. Genes cover the dodge's frame (where the tank is, where it is going, or BZFlag's cone), corridor and window, how the pilot closes in (half speed straight or BZFlag's weave), a back-off distance, the spacing of its two shots, the post-shot juke, a feed-forward term in the aim, and `dodge_commit`. Every gene is born off. Verification is an A/B test inside the same matches: two tanks on the current pilot, two on the candidate, two BZFlag AI tanks, same random map, same moment. Each arm scores net (kills − deaths) per tank-minute; the difference is divided by a Poisson standard error, $\sqrt{k+d}$ per tank-minute for each arm. The loop promotes at $z \geq 2$, rejects at $z \leq -1$, and otherwise records "not proven".

time	base	change tried (the cause it answered)	matches	z	decision
23:39	v2	dodge window 1.8 s (dodged too late, 57%, single packet)	2	−1.65	rejected
23:52	v2	weave while closing in (shot while aiming, 24%)	3	+0.69	not proven
00:04	v2	moving-frame dodge (leading shot, 9%)	3	+0.77	not proven
00:27	v2	6 m corridor (passed too close, 56%, first replay version)	2	+3.08	**promoted: v3**
00:40	v3	back off within 50 m (shot up close, 91%)	3	+0.43	not proven
00:44	v3	the same change, pooled	4	+1.03	not proven
01:01	v3	the same change, pooled	7	+1.80	not proven
01:05	v3	the same change, pooled	8	+1.74	**not proven**

Table 10. Every decision the loop recorded up to the cutoff (times on 26–27 September). Each match is 4 minutes. z is the loop's own statistic over all matches the change had so far. `dodge_commit` was never tried.

Three things kept the loop from finding the cause earlier (Table 10).

(1) **The explanation was wrong.** Single-packet forensics sent the loop after "dodged too late", and the change it proposed was rejected, correctly. A loop can only test the hypotheses its evidence proposes.

(2) **Evidence was thrown away.** A change that was "not proven" after three matches was set aside for good, with its matches. The moving-frame dodge had been ahead in all three of its matches and the weave in two of three. The loop now pools a change's matches across generations, up to nine, until it is decided.

(3) **The fix was not in the gene space.** No gene said "keep dodging the same shot". `dodge_commit` was added once the replays showed the dither. To the cutoff it had not been tested in a match, because by then the promoted pilot no longer died that way (below).

Where the loop stood at the cutoff. The 6 m corridor was promoted as v3 at 00:27, after two A/B matches in which its tanks went 32–24 against the current pilot's 17–42 ($z = +3.08$). With a wider corridor a release happens farther from the shot's path, and v3's deaths changed character: of 124 replayed v3 deaths in 11 matches to the cutoff, **106 (85%) were shot up close**, 14 dodge failed, 2 leading shot, 1 unflagged, and **1 dodge abandoned**. That is the profile of BZFlag's AI's deaths, not of v2's. In the one equal head-to-head match v3 has played (`20260927-004504`, 11.7 tank-minutes per side), BZFlag's AI still won, 28–22 against v3's 15–20. v3 died less often than the engine's AI (1.71 against 1.88 deaths per tank-minute) and killed about half as often (1.28 against 2.39). **On that single match, the gap has moved from defence to offence.** The next change the loop tried, BZFlag's own back-off within 50 m, was ahead in seven of its eight matches and still **not proven** at the cutoff ($z = +1.74$). We report the promotion with two cautions. Two four-minute matches are little evidence. And a loop that tests after every match and promotes at $z \geq 2$ looks repeatedly at accumulating data, so its real false-promotion rate is higher than a single z-test's [2].

7 The witness gate on firing

In BZFlag every driver decides from the engine's state, and the engine knows things the tank cannot see: an enemy behind a building is still in the packet. A pilot that fires on the packet alone can fire *blind*. The lane therefore runs a second witness: a learned check that looks at the tank's own rendered view and says, for each enemy the engine claims, whether that enemy is visible (*seen*), not visible, or cannot be judged. The eye itself, how it was taught by the engine, and the gate it had to pass before it could be armed are the subject of a separate paper [18]; the idea of the eye as a second witness to the engine's first comes from VDSG [16]. Here we describe only its role and measure its effect on firing.

Role. With witness firing on, our rules may fire only at an enemy the eye currently confirms, and a model tank's shot is held unless the eye confirms the enemy nearest its gun line. BZFlag's AI never consults the eye. The eye judges claims only out to 350 m, and it abstains at *point blank*, under 12 m, where a tank overflows the view. Every shot of our tanks is graded against the engine's depth buffer at the moment it is fired: a shot whose target was at most 5% visible is *blind*.

The focus match. Match `20260926-225220` ran three v2 rule tanks, with witness firing on, against three BZFlag AI tanks for three minutes, with every shot and every held shot in the black box. **Of 75 shots, 0 were blind**, and the eye had confirmed the target of all 75. The pilot held fire 26 times. Twelve of those holds were a single encounter: one of our tanks and one enemy 0.2–1.3 m apart for 1.75 s, the eye abstaining at point blank and the pilot holding again and again. For the other fourteen, the recorded verdict was "not seen" in eight, "cannot judge" in three and "seen" in one (we did not investigate that mismatch), and two had no verdict recorded.

Across matches. In every counted match whose results record graded shots, blind shots were rare with and without the gate: 1 in 197 with witness firing on (three matches) and 10 in 3,012 with it off (24 matches).

The pilot's own test, fire only on a clear line of sight in the engine, already avoids most blind shots. In these matches, then, the gate's measurable effect is the holds, and at point blank those holds are shots the pilot needed. What the gate is for is a driver whose aim we do not write: a model tank. No counted match ran model tanks with witness firing on, so gating a model's trigger is built and pinned by a unit test, and **not yet measured in a match**. Trusting the engine's line of sight inside 12 m, where the eye cannot frame the target, is an obvious remedy for the point-blank cost; it has not been built.

8 Discussion

8.1 What a decision rate buys

In BZFlag a shot crosses 100 m in one second. The fatal shots our v2 pilot died to were in view for a median of 1.4 s, and those that killed BZFlag's AI for 0.4 s. A driver that decides every 7.5 s cannot answer a shot in time; its only defence is a command it issued seconds earlier. A driver that decides every second gets, at best, one decision inside the warning our own pilot had. The servo that carries a model's command never dodges, by design: the comparison is between drivers, and a servo that dodged would be our rules driving the model's tank. Under that contract the models' tanks could aim, but they could not avoid being shot, and their deaths per minute show it.

The models did not compensate by driving cautiously. In the headline match their mean commanded speed, sampled once a second while alive, was 0.64 of full for Claude and 0.80 for DeepSeek, against 0.71 for the rules (0.42 for llama3.2:3b). That differs from our driving-simulator study, in which every language model at the wheel crawled at 8–10 km/h because it drove blind between glances of 0.8–2.0 s [17, §5.4]. The mechanism is the same, a held command between glances; the response differs because a car's worst case is its own speed and a tank's is someone else's shot.

The rules decided about 145 times as often as the fastest model. We do not claim that speed is the whole story: the rules also beat every model in the exam, where time does not count. Among the models, the fastest (DeepSeek) did best in the arena and the best on the exam (Claude) finished below it; llama3.2:3b, faster than Claude but giving a near-constant answer, did worst.

8.2 What the exam shows, and what the arena shows

The exam asks whether a driver knows what to do in one frozen situation; the arena asks whether it can do it in time, continuously. Claude was the best model on the exam (6 of 8) and the second-worst in the arena (1–15); DeepSeek scored 5 of 8 and was the best model in the arena. Neither measure alone ranks the drivers.

The exam has a sharper limit, and it is the one that matters for our own pilot: **a one-frame exam cannot contain a flaw that lives in a sequence**. Every decision in Table 9 is correct on its own terms: dodge a shot predicted to pass within 4 m, aim when none is. The failure is in the switching between them, visible only across packets. The rules scored 8 of 8 on the exam, and the dither behind most of the v2 pilot's replayed deaths is invisible to it.

8.3 Why the engine's AI winning is the most important result

Every other opponent in this paper was chosen or built by us: random play and a disarmed pilot to show the rules are not trivial, and language models behind an interface of our design. BZFlag's AI is the only opponent we did not write, and it is the only one that tells us how far our rules are from good rules. It is part of the upstream game, written by the game's own contributors, and it ran unchanged. Without it, the record would read "rules beat every model" and stop there. With it, the record says: *you do not need a neural network in the loop to beat these language-model drivers; you do need better rules than ours to beat the engine.* The second clause is the finding.

The engine's AI also taught by example. Its deaths are almost all shots nobody could have dodged,

which is what the deaths of a pilot that dodges well should look like. After the 6 m corridor was promoted, our pilot's deaths took the same shape (85% shot up close). In the one equal match since, it died less often than the engine's AI and still lost, because it killed half as often. That is a single four-minute match, and we read it as a direction to look, not as a result.

8.4 What the loss says about learning from failure

The pilot's learning loop had a sound verification step and a poor explanation step. The A/B test rejected the wrong fix the single-packet forensics proposed, which is what verification is for. It could not propose the right fix, because a loop tests only the hypotheses its evidence generates. The replays changed the evidence twice. The first version read the dither as a corridor that was too narrow, because it measured only the final unbroken warning. The second counted the first warning and the releases separately, and named the mechanism. The fail-first paper warns that credit assignment fails silently [15]; here it failed loudly enough to be caught only because the replays kept two seconds, not one packet. The same shape appears in BattleZone at a smaller scale: the dodge lab's evidence, survival, was not the game's objective, destruction, and only the verification on the game's own score caught it.

9 Related work

Atari evaluation. The Arcade Learning Environment made Atari 2600 games a standard benchmark [4]; Machado et al. documented how far evaluation protocols had diverged and introduced sticky actions, which our BattleZone runs use [9]. Mnih et al.'s DQN reported BattleZone scores for random play, a human tester and the agent [10], which we quote only as context because the protocols differ. Later agents exceed the human reference on every game [3]. Our BattleZone pilot is not a contender on that axis; its point is a pixel-only rule baseline with its failures measured.

Language models as game drivers. Multimodal models have been benchmarked as low-level Atari policies and found not yet capable of zero-shot control, with visual and spatial reasoning named as a cause [14]. BALROG measures agentic reasoning of language and vision-language models across games of increasing difficulty and finds that they struggle as tasks get harder [11]. TextStarCraft II lets language models play a real-time strategy game through a text interface [8]. Our arena differs in three ways: every model drives its own tank in the same real-time 3D match as a rule pilot and the game's own AI; every driver reads the same state through one interface; and each model's decision interval is measured and reported beside its score, because in a real-time game the interval is part of the result. Our driving-simulator study made the same observation about latency from the other side, a shield covering the blind time between a model's glances [17].

Rules at the wheel. Priority-ordered reactive rules are old: Brooks's layered control put the most urgent behaviour on top [5]. BZFlag's own AI is a hand-written rule AI of the same kind: dodge, get unstuck, chase, look for a flag, navigate, in that order [6]. Our pilot's structure is not new. What we add is the record: rules measured against baselines, models and an independent rule AI in the same matches, with every loss kept.

Switching and hysteresis. A threshold rule that switches without hysteresis chatters at its threshold; hysteresis, switching on at one level and off at another, is the classical remedy, from the Schmitt trigger [12] to hysteresis-based switching in supervisory control [7]. The pilot's dodge abandonment is this failure in a game, and the `dodge_commit` gene is this remedy. It is not yet measured.

Runtime checks on actions. Placing a verified check between a controller and its actuator is the Simplex architecture [13] and, for learned controllers, shielding [1]. The witness gate is a narrow instance: it can only remove the fire action, and only when a second witness does not confirm the target. The order of authority it respects, in which a check may narrow what the driver does and never widen it, is the one the fail-first and VDSG papers formalise [15, 16]. The eye itself is described in [18].

Learning loops and their statistics. The pilot's loop has the fail-first shape: observe verified failures, explain them, change one named behaviour, verify, keep or reject [15]. Its verification tests after every match against a fixed z threshold, a sequential design whose repeated looks inflate the false-positive rate [2]; we report its promotions with that caution.

10 Limits and threats to validity

Status. Research prototype, measured in two games on one machine (Apple M4, 16 GB). Nothing here is qualified for any use outside a game. Every number is ours, and none has been replicated by a third party.

Small samples. The headline arena result is one ten-minute match. There are six counted model matches, two of them under three minutes. Pilot v3 has played one equal head-to-head match. The loop's A/B matches last four minutes each. The exam asks each situation once. BattleZone uses ten seeds per arm, which separates the pilot from random play but not the DODGE arm from the pilot. We state n beside each result and do not pool across conditions except where a table says so.

One configuration per model. Each language model ran with one system prompt, one output schema, one servo and no examples in the prompt; we did not tune prompts per model. Claude ran through the `claude -p` interface with effort "low", never through its API; DeepSeek through its public API from one location; Llama as a 3B model on the test machine. A different prompt, interface, effort setting or model size could change any of these results. The servo never dodges for a model, by design. A hybrid in which a model chooses targets and rules dodge was not tested.

Unequal footing, where it exists. BZFlag's AI runs inside the client with no transport delay; our controllers answer over a local socket, and the command lands on a later frame. The state packet reports shots within 200 m and a tank's height; the engine's AI reads its own world directly, though its dodge considers only shots within 100 m. In free-for-all matches with unequal numbers per family, a family's K/D includes fights between its own tanks. The A/B lineup has four of our tanks and two of the engine's. We say one driver beats another only from the models lineup (one tank per driver) or the head-to-head lineup (equal numbers of two drivers); the A/B lineup compares a pilot only with its candidate.

The maps. Every match is played on a new random world generated by the server, which adds variance between matches and removes any chance of tuning to one map. Comparisons within a match share the map; comparisons across matches do not.

The forensics. A death's cause is one label, and the first matching cause wins. The labels depend on thresholds: a warning 0.5 s before the hit counts as in time; the fatal shot must pass within 8 m; packets are kept at one per 40 ms; shots are assumed to fly straight. No person has checked the labels against the video. The single-packet classifier was wrong in a way the replays exposed, and the replay classifier could be wrong in ways we have not found.

The loop's statistics. The A/B statistic treats kills and deaths as Poisson counts and tests after every match. Repeated looks inflate the chance of a false promotion [2], and v3 was promoted after two four-minute matches. The loop kept running after the cutoff, so v3 may be superseded or regress; this paper reports the record to 01:05:50 on 27 September 2026 and nothing after it.

BattleZone specifics. One ROM, ten seeds, sticky actions at the v5 default. The hand reading of the 25 death strips is approximate. The DQN and human numbers are quoted from a different protocol and are context, not a comparison.

Crashes and exclusions. BZFlag's OpenGL client can crash under macOS; one such loss invalidated the first ten-minute match. After supervision was added, no client exited in any recorded match to the cutoff. Seven runs are excluded, each with its reason (Table 7). All measurements were made by the group that built the systems.

11 Conclusion

A rule pilot with no neural network in the loop that decides plays Atari BattleZone from pixels far above random play. In a real-time 3D arena it beats three language-model drivers given the same facts, in five of six counted model matches and pooled, helped by deciding a hundred times a second where they decide once every one to sixteen seconds. It loses to the game's own AI in every equal match. The loss is the useful result. Its first explanation was wrong, its second only half right, and its third, a replay of the last two seconds before each death, showed a pilot that saw the fatal shot more than a second early, started to dodge, and let go of the dodge at the edge of its own corridor, again and again. A wider corridor has since changed how the pilot dies. It has not yet made the pilot win, and the hysteresis fix the replay points to has not been tested. We will report it when it has been, including if it loses.

References

[1] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, U. Topcu. Safe reinforcement learning via shielding. In *Proc. AAAI Conference on Artificial Intelligence*, 2018.

[2] P. Armitage, C. K. McPherson, B. C. Rowe. Repeated significance tests on accumulating data. *Journal of the Royal Statistical Society, Series A* 132(2):235–244, 1969.

[3] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, C. Blundell. Agent57: outperforming the Atari human benchmark. arXiv:2003.13350, 2020.

[4] M. G. Bellemare, Y. Naddaf, J. Veness, M. Bowling. The Arcade Learning Environment: an evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279, 2013.

[5] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* 2(1):14–23, 1986.

[6] BZFlag. Open-source multiplayer 3D tank battle game. github.com/BZFlag-Dev/bzflag; built from commit `4299415` (8 September 2026).

[7] D. Liberzon. *Switching in Systems and Control*. Birkhäuser, Boston, 2003.

[8] W. Ma, Q. Mi, Y. Zeng, X. Yan, Y. Wu, R. Lin, H. Zhang, J. Wang. Large language models play StarCraft II: benchmarks and a chain of summarization approach. arXiv:2312.11865, 2023.

[9] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, M. Bowling. Revisiting the Arcade Learning Environment: evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research* 61, 2018. doi:10.1613/jair.5699.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533, 2015.

[11] D. Paglieri, B. Cupiał, S. Coward, U. Piterbarg, M. Wolczyk, A. Khan, E. Pignatelli, Ł. Kuciński, L. Pinto, R. Fergus, J. N. Foerster, J. Parker-Holder, T. Rocktäschel. BALROG: benchmarking agentic LLM and VLM reasoning on games. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2411.13543.

[12] O. H. Schmitt. A thermionic trigger. *Journal of Scientific Instruments* 15(1):24, 1938.

[13] L. Sha. Using simplicity to control complexity. *IEEE Software* 18(4):20–28, 2001.

[14] N. R. Waytowich, D. White, MD Sunbeam, V. G. Goecks. Atari-GPT: benchmarking multimodal large language models as low-level policies in Atari games. arXiv:2408.15950, 2024.

[15] Perslis Research. Fail-first models: failure becomes structure. Research paper, 2026. research.perslis.com/fail-first

[16] Perslis Research. VDSG: a commanded admission-control runtime for autonomous agents. Systems paper, 2026. research.perslis.com/vdsg

[17] Perslis Research. Runtime admission control on a photoreal driving simulator. Empirical study, 2026. research.perslis.com/carla-admission

[18] Perslis Research. The witness eye. Paper in preparation, 2026. research.perslis.com/witness-eye

A Where each number comes from

Every number is listed with its source in the accompanying claims file (`CLAIMS.md`). The sources are the two lanes' own evidence files, read without modification; nothing was re-run in either game. Three analysis scripts recompute every derived number, importing the BZFlag lane's own analysis code (match counting, forensics, pilot genes) at the commit current when the data were frozen (`6f76e6a`), so later edits to the live lane cannot change these numbers.

result	source
BattleZone arms, per-seed scores, DODGE verification	the tank lane's `evidence/eval_v1.jsonl` and `eval_v2.jsonl` (ten seeds per arm); statistics by `analysis/battlezone_stats.py`
Dodge lab	the tank lane's `evidence/dodge_lab.json`
Death windows and Figure 2	the tank lane's `evidence/deaths/` (25 windows; the figure crops `1_3.png`)
DQN, human and random reference scores	[10], Extended Data Table 2 and Methods
Arena matches, headline table, decision times, pooled record, exclusions, stuck shares	each match's `meta.json`, `results.json`, `scoreboard.jsonl` and black boxes on the offload drive, to run `20260927-010140`; `analysis/arena_analysis.py`
Driving exam	the BZFlag lane's `evidence/exam.jsonl` (48 answers); `analysis/exam_stats.py`
Death causes, replays, the example trace	each match's `deaths-*.jsonl` (two seconds of packets per death, from 00:15 on 27 September), classified by the lane's forensics at `6f76e6a`, and by its earlier versions `96259ea` and `07a04e2` for comparison
Learning-loop decisions	the BZFlag lane's `evidence/pilots/evolution.jsonl` and `evolve.log`, records to 01:05:50
Witness firing	`shots-*.jsonl` (per shot, from run `20260926-225220`) and `results.json` counters
Engine facts (shot speed, tank size, turn rate, the AI's rules, the random world)	BZFlag source at `4299415`: `src/common/global.cxx`, `src/bzflag/AutoPilot.cxx`, `src/bzfs/bzfs.cxx`, `WorldGenerators.cxx`

Reproducing the numbers. From the paper's directory, run `python3 -B` on `analysis/battlezone_stats.py`, `arena_analysis.py` and `exam_stats.py` (they write `analysis/out/*.json`), then `figure_data.py` (the figure coordinates used by this PDF and by the web edition). The scripts only read; `-B` keeps Python from writing bytecode into the lanes. Figure 2 is six frames of the tank lane's recorded death strip, cropped and not re-rendered; every other figure is drawn from the numbers in the tables beside it, through those coordinate files.