RED ALERT BENCH: A DIAGNOSTIC BENCHMARK FOR MULTIMODAL PLANNING AND EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal agents in robotics and other embodied settings need to use observations to plan actions, monitor their effects, and achieve a goal. Final success rates alone do not reveal where this process breaks down. We introduce Red Alert Bench, a diagnostic benchmark and reproducible simulation environment for multimodal planning and execution, based on the real-time strategy game *Red Alert*. Agents gather resources, build, scout, and fight on a map where actions take effect over time, with configurable observation modality (text, image, or both) and map visibility (limited or full). Researchers can evaluate their own agents on 217 human-authored tasks, each with a verifiable goal, and in one-on-one matches where two agents control opposing sides. Evaluating eight LLMs, we find that text-only input outperforms image-primary input by 7–11 percentage points in each of the four models tested for this comparison. We also find that models repeat blocked commands, such as orders they cannot afford, without adjusting their plans. Agent-to-agent matches reveal a perspective-dependent spatial bias: on a symmetric map, the agent starting on the east side wins 68% of all games and 75% of self-play games. Red Alert Bench thus offers a dynamic world in which agents compete against one another, with execution records that link outcomes to the observations and actions behind them. We release the environment, tasks, evaluation tools, and interaction records.

1 INTRODUCTION

Multimodal agents must use observations to choose actions, monitor their effects, and revise plans. These demands arise in robotics and embodied-agent evaluation, where spatial understanding and dependent actions jointly determine success (Firoozi et al., 2025; Yang et al., 2025; Li et al., 2024). Final success rates do not explain why agents fail: whether they missed information, chose ineffective actions, or failed to follow through. Diagnosing this requires tracing observations, actions, and their effects toward verifiable, long-horizon goals in a dynamic, partially observed environment.

Real-time strategy (RTS) games provide such environments (Vinyals et al., 2017), and they exercise both demands. They require *spatial understanding*: the map is only partly visible, so agents must move units to reveal what they cannot see, and positions determine what can be reached, defended, or attacked. They also require completing *dependent actions* along several chains at once, such as gathering resources, expanding a base, and moving forces, where structures must exist before the units they produce and orders take several turns to complete. Because these chains interact with each other and with an opponent's hidden choices, outcomes emerge from the whole situation, so agents must keep checking whether each step took effect.

We introduce Red Alert Bench, a diagnostic benchmark and reproducible simulation environment based on the RTS game *Red Alert*. It evaluates agents in three settings within one **dynamic world**: authored tasks, mid-task handovers, and one-on-one adversarial matches. In authored tasks, an agent pursues an explicit goal under specified constraints and a decision budget (Figure 1a). In handover evaluations, an agent takes over partway through a task and must continue from a state it did not create. In the **adversarial arena**, two agents of the researcher's choice control opposing sides, each acting on its own observations (Figure 1d). In every setting, **observation controls** present the same situation as text, a labeled map image, or both, with limited or full visibility (Figure 1b), and **execution and task**

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

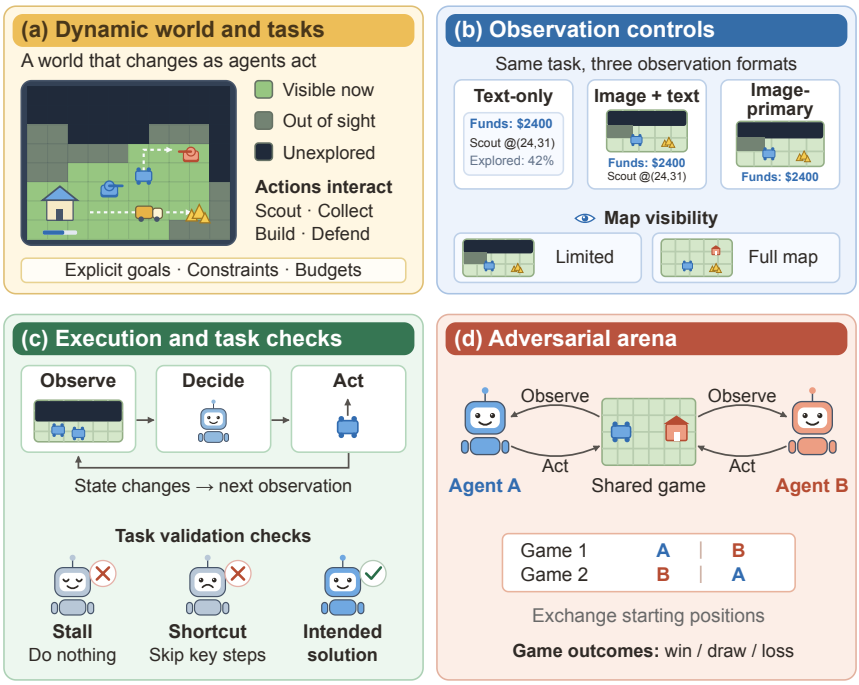


Figure 1: **Schematic overview of Red Alert Bench.** A dynamic environment supports 217 human-authored tasks across 12 families and one-on-one adversarial matches, with observation controls and execution records.

checks record each observation, command, and resulting change in the world while confirming that intended strategies succeed and that inaction or skipped sub-goals cannot trigger success (Figure 1c).

Our contributions are:

- A diagnostic benchmark combining 217 human-authored tasks, mid-task handover, and one-on-one adversarial matches with three observation formats, visibility controls, execution records, and validation checks, built on a from-scratch Rust engine with a Python interface.
- An evaluation of eight LLMs, a non-LLM decision model, and a human baseline showing that agents succeed 7–11 percentage points more often with text-only than with image-primary input in the four models tested. Task and arena records show most models repeatedly ordering units that never appear, while Jev, a non-LLM agent restricted to legal actions, still loses every economy run, so valid actions are not enough.
- Adversarial analysis revealing a perspective-dependent spatial bias: on a symmetric map, the east starting side wins 68% of all games and 75% of self-play games.

2 The Red Alert Bench

2.1 System Architecture and Agent Interaction

A *run* is one attempt at an authored task or one competitive game. Each run consists of a sequence of *decision turns*. In each turn, the agent receives an observation, submits commands, and the simulator advances the world to produce the next observation (Figure 2).

Agent interface. Each agent receives an observation and returns a batch of commands through a shared Python interface. The supplied LLM adapter converts tool calls into commands; custom agents can generate commands directly, as the non-LLM agent in Section 4.4 does. Model inputs include the objective, game rules and prerequisites, current observation, and the last 16 turns of interaction history.

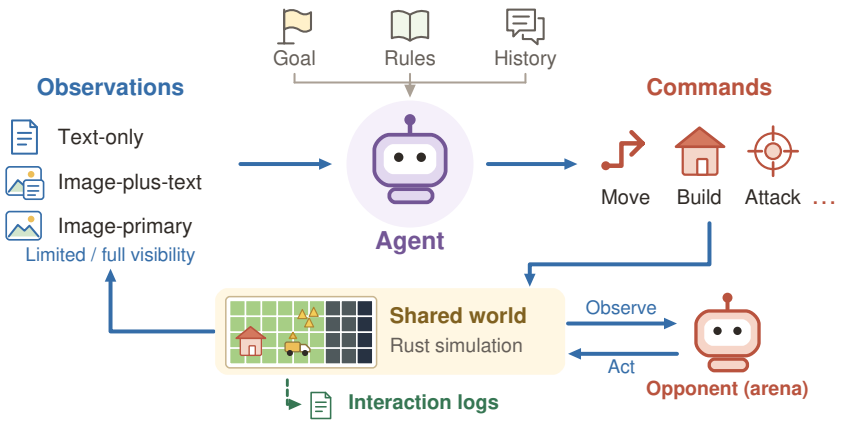


Figure 2: **Agent interaction with Red Alert Bench.**

Observations. *Text-only* provides structured descriptions including spatial information; *image-plus-text* adds a labeled map; *image-primary* uses the map for spatial information while retaining non-spatial text such as resource totals. Each side receives its own view. Limited visibility distinguishes areas currently in sight, previously explored areas outside sight, and unexplored areas; full-map visibility reveals the map.

Actions and feedback. Commands move units, gather resources, construct buildings, produce units, or attack. Actions can span turns: production can continue while other units move. Subsequent observations let agents check progress and revise orders. The simulation pauses during decisions and, in adversarial matches, advances after both agents submit commands.

Simulation and rollout support. The Rust simulator uses OpenRA's rules and data (OpenRA, n.d.) and is exposed as a native Python module. Independent Python workers support parallel reinforcement learning *rollouts* (sequences of agent–environment interactions), recording interactions and evaluating outcomes. Appendix G details initialization, stepping, rewards, and replay.

2.2 Decision Structure and Tradeoffs

The task suite and arena combine decision demands under incomplete information and delayed effects. Examples include:

- **Economic investment.** *Harvesters* deliver ore to *refineries* for income. Investing in more harvesters leaves fewer funds for immediate defense.
- **Spatial and tactical control.** Refinery placement balances shorter collection routes against exposure to attack, while combat requires positioning, target selection, and retreat.
- **Information acquisition.** Scouting reveals unseen areas and enemy activity but takes time and puts scouting units at risk.
- **Timing and coordination.** Production queues and building prerequisites delay actions, requiring agents to sequence dependent steps and coordinate multiple unit groups.
- **Adaptation and recovery.** Losses of harvesters or buildings require agents to reallocate resources, rebuild, and revise interrupted plans.
- **Strategic interdependence.** In adversarial matches, the value of expanding, defending, or attacking depends on the opponent's actions.

Agents can make different choices about when to attack, how much to invest in their economy, and whether to launch repeated small-scale raids.

Interaction model. We use the partially observable stochastic-game framework (Hansen et al., 2004) to formalize agent–environment interaction. At decision turn t, agent $i \in \{1, \ldots, n\}$ receives

observation o_t^i of full simulator state s_t. Policy π_i selects command batch a_t^i from observation–action history h_t^i, including supplied context and the current observation:

$$\begin{aligned} o_t^i &= \mathcal{O}_i^{m_i,v_i}(s_t), \qquad a_t^i \sim \pi_i(\cdot \mid h_t^i), \\ (s_{t+1}, \Delta_t) &\sim \mathcal{P}_c(\cdot \mid s_t, \mathbf{a}_t), \qquad \mathbf{a}_t = (a_t^1, \ldots, a_t^n). \end{aligned} \tag{1}$$

Here $\mathcal{O}_i^{m_i,v_i}$ maps state to observation under input format m_i and visibility setting v_i; $\mathbf{a}_t$ collects all n agents' commands. Under run configuration c, transition distribution $\mathcal{P}_c$ governs the next state s_{t+1} and elapsed simulation ticks Δ_t given state and joint commands; $\sim$ denotes sampling. Tasks have one agent ($n = 1$), with background behaviors in $\mathcal{P}_c$, forming a partially observable Markov decision process (Kaelbling et al., 1998). Adversarial games have two ($n = 2$). Appendix F details configuration, state, timing, and payoffs.

2.3 Task Suite and Design

The 217 human-authored tasks are organized into 12 *task families* according to the capabilities they emphasize (Table 1). Individual tasks can involve multiple capabilities, and their goals are automatically verifiable.

Task specification. A *task configuration* specifies the objective, initial map and state, permitted commands, success and failure conditions, and a *decision budget* limiting turns. Easy, medium, and hard versions of a task differ in initial conditions and requirements (Appendix D).

Verifiable goals and validation. Success is decided by conditions checked directly on the game state, such as units reaching a region, targets destroyed, or resources collected before a deadline, so no LLM judge is involved. Automated tests confirm that intended solutions succeed and that doing nothing or skipping required steps does not (Appendix H).

Table 1: Task families and representative demands. Individual tasks can combine several demands.

Family	Tasks	Representative demand
Combat Micro	25	Coordinate movement and target selection.
Static Defense	19	Protect locations and respond to incoming threats.
Multi-group Coordination	9	Synchronize actions by separate unit groups.
Resource Economy	28	Allocate resources under production constraints.
Build & Tech	19	Execute dependencies between construction steps.
Special Units	12	Use specialized actions and timed interventions.
Scouting & Vision	18	Find targets and refresh information outside sight.
Indirect Strategy	10	Use alternative routes and ordered sub-goals.
Long-Horizon Play	21	Sustain plans across multiple stages.
Tempo & Adversarial	9	Balance development against an opponent's timing.
Multi-front Resilience	30	Recover and coordinate across several locations.
Procedure Robustness	17	Follow ordering, tool, and objective constraints.
Total	**217**	

2.4 One-on-One Adversarial Matches

Two separately controlled agents, each with its own observations, play on the same map. Because a starting position can favor one side, a *paired match* is two games with starting positions swapped under the same configuration and seed. An agent wins the pair by winning both games, loses it by losing both, and otherwise draws (Appendix J).

2.5 Execution Records and Evaluation

A *trajectory* links observations, commands, and recorded consequences across a run. Task records align decisions with subsequent state changes and goal progress; arena records capture commands and military and economic state. These records show whether agents act on information and follow through.

Task success and arena win/draw/loss are the primary outcomes. Secondary measures of goal progress, map exploration, action timing, and command use describe observable behavior, not internal perception or reasoning. Appendix I defines these measures and their temporal alignment.

3 Experimental Setup

We evaluate eight LLMs. Default task settings are image-plus-text input, limited visibility, temperature $\tau = 0.7$, and environment seed 1. Task studies use 95% confidence intervals from 10,000 task-level bootstrap resamples, retaining associated difficulties and attempts together. Appendix J specifies models, API providers, coverage, exclusions, weights, and historical-setting differences.

- **Task performance and repeated attempts.** Easy, medium, and hard tasks have up to three attempts. Repeated-attempt comparisons require three scored attempts on the same task, difficulty, and seed.
- **Observation and visibility ablations.** For each of 217 medium tasks, we score one attempt under each combination of three observation formats and two map visibility settings.
- **One-on-one adversarial matches.** The eight models played 119 two-game matches, with starting positions swapped for the second game. We compute 95% confidence intervals using 10,000 bootstrap resamples. All games involving the same two models and environment seed are resampled together as a group (Appendix E).
- **Idle-handoff stress test.** On 217 medium tasks, three models either play from the start or take over after ten turns in which a scripted controller issues no orders. This measures the combined effect of lost time and a changed world state (Appendix P).
- **Non-LLM agent.** Jev, a decision model restricted to enumerated legal actions, plays one task from each of 11 families at three difficulties and seeds 1–3 with text-only input.
- **Human performance baseline.** We compare success, progress, and exploration on 183 task–difficulty combinations shared with all eight models. Testers use a UI presenting the same game information and observation format as the agents.

4 Findings

4.1 Task Performance and Human Baseline

Across the eight models, task success ranges from 37.5% for Kimi K2.5, whose replies contain no tool call on 81% of turns (Section 4.4), to 82.1% for Opus 4.7 (Figure 3). Even the highest-scoring model fails to achieve the task goal in nearly one in five scored attempts.

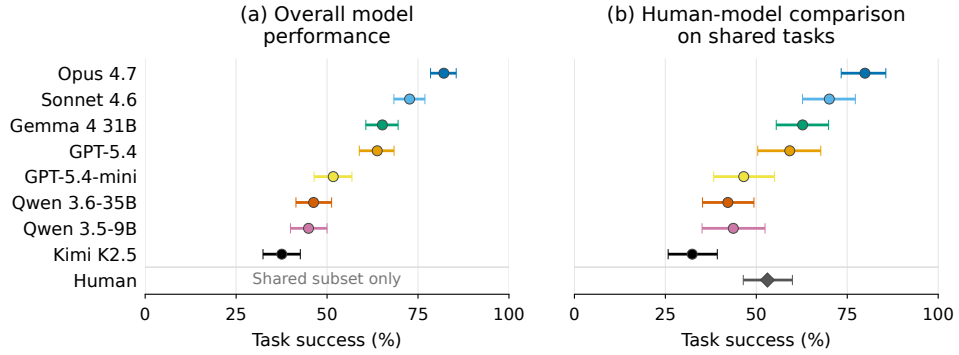


Figure 3: Task success with 95% confidence intervals from 10,000 task-bootstrap samples. (a) Overall model sample, pooling difficulty levels and recorded attempts; three models (GPT-5.4, GPT-5.4-mini, Qwen 3.5-9B) have one attempt per combination, the others up to three. (b) Human–model comparison on 183 shared task–difficulty combinations with identical task weights. Appendix K gives exact estimates and coverage.

Human testers reach 53.1% success (95% CI: 46.4–59.9%) on the 183 task–difficulty combinations shared with all eight models, placing fifth among nine cohorts under identical task weights (models: 32.4–79.9%). This middle position averages two distinct profiles. Humans lead every model on Indirect Strategy (82% versus a model median of 17%) and match the best model on Special Units, families in which a single well-chosen plan often decides the outcome. They fall to sixth through eighth on economy, construction, defense, and multi-front tasks, which demand sustained management across many turns and account for half of the shared combinations (Appendix L).

4.2 Observation Format and Map Visibility

Text-only input outperforms image-primary input. Under limited map visibility, text-only input yields success rates 7.4–10.6 percentage points higher than image-primary input across all four models (Table 2). All four paired 95% intervals lie above zero (Appendix O). Adding a map to the structured text produces smaller changes, from -3.2 to $+0.9$ points; all four intervals include zero. The formats also differ in how they report positions and name units, so the gap may not be purely visual.

Table 2: Success rates (%) on 217 medium-difficulty tasks. Bold marks the highest success rate within each visibility setting, without implying significance. Appendix O reports 95% confidence intervals for success rates and their differences across observation formats and map visibility settings.

	Limited map visibility			Full-map visibility		
Model	Image-primary	Image-plus-text	Text-only	Image-primary	Image-plus-text	Text-only
Gemma 4 31B	52.1	**63.6**	62.7	55.3	**67.7**	65.9
GPT-5.4-mini	41.0	48.8	**50.2**	38.7	**48.8**	47.9
Qwen 3.6-35B	37.3	41.9	**45.2**	41.9	44.2	**45.2**
Qwen 3.5-9B	34.6	38.7	**41.9**	35.0	42.4	**46.1**

Full-map visibility shows no consistent benefit. Full visibility changes success by -2.3 to $+4.6$ percentage points across the twelve model–format combinations. Every paired interval includes zero.

4.3 Repeated Attempts and Execution Follow-Through

Aggregate success is stable across attempts. For each model, the highest and lowest success rates across three matched attempts differ by only 1.4–2.3 percentage points (Table 3; individual outcomes and paired intervals in Appendix B). Individual tasks vary more: 15.6–29.2% of task–difficulty combinations change outcome across attempts.

Table 3: Task success across three attempts on matched task–difficulty combinations within each model. Range is the highest minus lowest success rate in percentage points (pp), computed before rounding. Appendix B reports confidence intervals, coverage, and historical-setting limits.

	Task success (%)			
Model	Attempt 1	Attempt 2	Attempt 3	Range (pp)
Opus 4.7	82.3	83.0	81.0	2.0
Sonnet 4.6	71.9	71.9	74.2	2.3
Gemma 4 31B	63.9	65.2	66.1	2.2
Qwen 3.6-35B	44.9	45.7	46.7	1.9
Kimi K2.5	37.8	38.4	37.0	1.4

In one task, a stalled run repeats requests that never take effect. Figure 4 shows three attempts at the same task. Run A reaches the goal after 36 decisions, and Run B stops making progress after turn 35 and waits. Run C shows the repeated-request failure: after losing all its units, it orders new ones six times with no money to pay for them, and none are ever built.

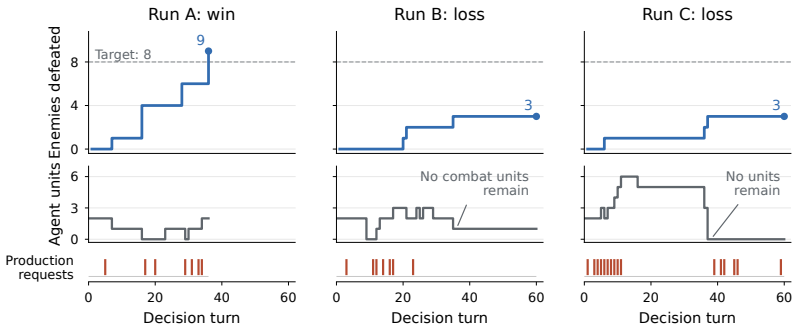


Figure 4: Successful and stalled execution in three archived Qwen 3.6 attempts at the same task (medium, seed 1). Recorded enemy kills (target: eight), remaining units, and production requests are aligned by decision turn. Run A succeeds; Runs B and C stop at three kills. Appendix B gives provenance; Appendix R reproduces Run C in full.

The stalled runs acknowledge the block and keep the plan. In Figure 4, Run B's turn-53 response notes many turns without any change yet keeps waiting for production, and Run C's turn-50 response names the lack of funds blocking production yet expects the unit to finish and orders another on turn 59. Neither run builds a unit or makes further progress. Both responses describe the problem correctly and then predict progress anyway.

4.4 Valid Actions That Do Not Take Effect

A well-formed command can still do nothing. A *unit order* is a command to produce a mobile unit. Each briefing lists funds, units, and the production queue, so an agent can check whether it can afford a unit and whether it is already queued. We compared unit orders with the new units that appeared in 12,903 model runs and 246 human runs (Table 4).

Table 4: Unit orders on the task suite: new units per 100 orders, share repeating the previous turn's order, and within-task win-rate gain (percentage points, 95% interval) of runs whose orders mostly produce a unit, using all orders or excluding the last five turns. – marks fewer than 15 comparable tasks, for which no difference is reported (Appendix M).

Agent	Fielded	Repeated (%)	Win-rate gain, all orders	Win-rate gain, excluding last 5 turns
Opus 4.7	84	25	+27 [+11, +44]	+20 [+4, +39]
Sonnet 4.6	80	47	+23 [+10, +37]	+21 [+6, +36]
Gemma 4 31B	36	63	+16 [+3, +28]	+10 [-3, +24]
GPT-5.4	36	59	–	–
GPT-5.4-mini	21	54	–	–
Qwen 3.6-35B	36	47	+15 [+4, +26]	+15 [+4, +26]
Qwen 3.5-9B	4	84	+10 [-4, +24]	+10 [-2, +22]
Kimi K2.5	88	6	–	–
Human	77	10	–	–

Most orders from five models produce nothing. Gemma, both GPT models, and both Qwen models field 4–36 new units per 100 orders; Opus, Sonnet, and Kimi field 80–88. These five repeat the previous turn's order in 47–84% of orders, and 46% of Qwen 3.5-9B's orders are unaffordable.

Units per order go with task success. Within the same task, runs that field at least one new unit per two orders win 15–27 percentage points more often for Opus, Sonnet, Gemma, and Qwen 3.6, with intervals excluding zero. Because a run that ends early leaves its last orders unfinished, we repeat the comparison without orders from the final five turns: the difference persists for Opus, Sonnet, and Qwen 3.6, while Gemma's falls to 10 percentage points with an interval that includes zero.

Valid actions are not enough. Jev, a decision model that does not generate text, plays through the same interface by choosing among the rule-legal actions its controller lists, so it cannot issue an impossible order. With text-only input it wins 7 of the 11 medium tasks it shares with the observation study, where the four LLMs win 6–8 with the same input (Appendix N). It still loses every economy and defend-then-counterattack run. Knowing the legal actions does not tell an agent which to take.

Human players order once and let orders finish. Humans repeat 10% of unit orders and field 77 new units per 100 orders. Kimi repeats only 6%, mainly because 81% of its replies contain no tool call. Fewer than 5% of any model's commands draw an engine warning; the low-yield models order units they cannot afford or already ordered the turn before.

4.5 One-on-One Adversarial Competition

Pairwise competitive performance. Figure 5 compares agents against active opponents. Opus wins both games in 22 of its 31 paired matches and splits wins in the other nine. Across all models, 58 of 119 pairs split one win apiece and count as paired-match draws; no individual game ends in a draw. Match scores order the models differently from task success: Kimi K2.5 rises from eighth in task success to fourth, GPT-5.4-mini from fifth to third, and Gemma falls from third to sixth (Table 10). These are point estimates; only Opus's interval separates it from the rest, and opponents and API providers varied across models (Appendix E).

Opponent	Opus 4.7	Sonnet 4.6	Gemma 4 31B	GPT 5.4	GPT 5.4-mini	Qwen 3.6-35B	Qwen 3.5-9B	Kimi K2.5
Opus 4.7	--	3-3-0	4-1-0	2-2-0	3-2-0	5-0-0	3-1-0	2-0-0
Sonnet 4.6	0-3-3	--	0-4-0	0-3-1	1-3-0	4-0-0	1-3-0	1-3-0
Gemma 4 31B	0-1-4	0-4-0	--	1-1-2	0-3-1	--	3-0-0	0-2-2
GPT-5.4	0-2-2	1-3-0	2-1-1	--	1-7-4	0-3-2	0-6-0	1-2-0
GPT-5.4-mini	0-2-3	0-3-1	1-3-0	4-7-1	--	0-2-2	1-3-2	2-2-0
Qwen 3.6-35B	0-0-5	0-0-4	--	2-3-0	2-2-0	--	2-0-0	0-1-3
Qwen 3.5-9B	0-1-3	0-3-1	0-0-3	0-6-0	2-3-1	0-0-2	--	1-1-1
Kimi K2.5	0-0-2	0-3-1	2-2-0	0-2-1	0-2-2	3-1-0	1-1-1	--

Cell counts: row model's wins - draws - losses

0% 50% 100%

Match score: (wins + 0.5 x draws) / matches

Figure 5: Outcomes of 119 two-game matches with exchanged starting positions. Cells show the row model's paired wins–draws–losses against each opponent; gray indicates no retained comparison. Model order follows Figure 3a. Appendix E reports coverage and uncertainty.

Outcomes differ by starting position. Position analysis uses all games, including same-model games. Among 194 games with positions verified from playback, the east side wins 132 (68.0%; 95% CI: 62.0–74.2%). In the 28 verified same-model games, it wins 21 (75.0%; 61.1–89.7%). The map is symmetric and the advantage holds in both engine slots, so it points to the agents' play. Swapping sides balances assignments, but the advantage can still turn a pair into a split draw.

Matches are one-sided but rarely quick. Among the 194 games with per-turn records for both players, 137 end by base elimination and 57 at the 200-turn limit. The eventual winner never loses a building in 144 of these games. In games ending in base elimination, the winner takes a building-count lead that lasts to the end at a median turn of 34 but destroys all opposing buildings only at a median turn of 80. Only 16 of the 194 games end within 50 turns.

Command volume does not translate into fielded units. Table 11 (Appendix E) summarizes each side's commands and production in the same games. Median unit requests per side range from 7 for Qwen 3.6 to 570 for Qwen 3.5-9B, yet median fielded units range only from 3 to 15. Qwen 3.5-9B fields 2 new units per 100 requests: 85% of its build orders exceed its current funds and 77% repeat the previous turn's order, whereas Kimi K2.5, which fields 68 per 100, does either for about 1%. As in the task suite, Kimi requests little because most of its replies contain no tool call, which the harness records as an observation (Appendix M).

Records separate stated plans from executed actions. In one archived match against Opus, Qwen 3.6 describes plans to produce units and scout, yet none of its 46 saved responses contains a tool call, every command batch contains only observation requests, and its base is eliminated on turn 46. This case is not isolated: 16 of the 40 traced Qwen 3.6 sides issue only observation requests for the entire game.

5 Related Work

Diagnostic game benchmarks. SmartPlay (Wu et al., 2024) tests planning, spatial reasoning, and learning from history; BALROG (Paglieri et al., 2025) studies LLM and VLM agents across interactive environments. DSGBench (Tang et al., 2025) evaluates strategic decisions across six games and analyzes decision trajectories; StarCraft II Arena (Anonymous, 2024) scores fine-grained capabilities against built-in AI. Both use text input and fixed opponents. Our diagnostic design connects explicit task requirements, observation controls, and records of subsequent goal progress within one RTS environment.

RTS and adversarial evaluation. TextStarCraft II (Ma et al., 2024) evaluates text-based agents against built-in AI and human players. RTSGameBench (Kim et al., 2026) combines diagnostic mini-games, observation ablations, and full games with built-in AI teammates and opponents. Orak (Park et al., 2026) includes direct LLM-versus-LLM battles, reporting one StarCraft II game per model pair. Our arena pits any two agents chosen by the researcher against each other in paired games with swapped starting positions, through the same interface used for authored tasks.

Embodied and spatial planning. EmbodiedBench (Yang et al., 2025) evaluates visual navigation and manipulation with observation and feedback ablations; iVISPAR (Mayer et al., 2025) compares visual and textual inputs in interactive spatial puzzles. Red Alert Bench studies observation and execution amid concurrent activities, delayed effects, resource constraints, and adversarial responses; transfer to robotics remains untested.

Reliability and execution analysis. τ-bench (Yao et al., 2025) evaluates tool-use agents through final states and repeated success. Decision records additionally expose repeated requests without their intended effects and expectations of progress despite acknowledged constraints (Section 4.3), related to the knowing–doing and observation–behavior gaps reported in other game benchmarks (Paglieri et al., 2025; Zhang et al., 2025a; Ahn et al., 2025).

Simulation resources for learning. SC2LE (Vinyals et al., 2017) provides RTS interfaces and mini-games for reinforcement learning; MineDojo (Fan et al., 2022) combines a configurable Minecraft simulator, diverse tasks, and learning resources. Our Rust simulator supports parallel rollouts through the same Python interface.

6 Discussion and Conclusion

Red Alert Bench provides a controlled testbed for studying how agents turn observations and goals into effective action over multiple steps. Its findings highlight the importance of information presentation, action execution, and the conditions under which agents are compared. Beyond measuring success, the benchmark enables researchers to compare observation interfaces, agents, and execution feedback, using recorded trajectories to examine where and how behavior changes.

Embodied systems also face a gap between issuing a command and achieving its intended effect; success checks and failure explanations can help agents recognize and respond to this gap (Huang et al., 2023; Liu et al., 2023). Red Alert Bench offers a complementary setting for studying this gap, observation design (Wang et al., 2024), and reliability across attempts (Kress-Gazit et al., 2024); its deterministic simulation lets researchers reproduce failures and test changes under matched initial conditions. Transfer to physical systems remains untested (Appendix Q).

Ethics Statement

Human testers were uncompensated volunteers who played the tasks through the benchmark's play interface with consent to record and release their sessions; sessions are identified only by a timestamp and seed and contain no personal information. The environment is a fictional strategy game, and the released tasks, simulator, and records contain no real-world hazardous content. All evaluated models were accessed through their providers' official APIs.

Reproducibility Statement

Section 2 and Appendices G–J specify the simulator, task format, validation, scoring, and study records, including the API provider and model identifier behind every result and each attempt count. Appendix R reproduces the full system prompt and a complete interaction record. We will release the simulator, the 217 tasks, the evaluation harness, the interaction records, and the analysis scripts named in the appendices. Every reported estimate is regenerated from the archived records by scripts that check source hashes, with fixed bootstrap seeds, and the engine is deterministic given a task configuration, seed, and command sequence.

AI Use Disclosure

We used LLM-based assistants (OpenAI Codex and Anthropic Claude Code) while preparing and revising this paper. *Writing:* they drafted and edited sections, captions, and this statement, and polished wording; the authors reviewed, revised, and approved all text. *Retrieval:* they helped identify and compare related work; every cited reference was checked against its primary source. *Implementation:* coding assistants also helped implement and maintain parts of the benchmark software, including the evaluation harness, prompts, validation tests, and the Jev controller, and applied fixes to task configurations identified by validation tests. The 217 tasks were designed and written by the authors; none was drafted by an AI system. *Research execution:* they wrote and ran the analysis and audit scripts released with the paper, which recompute every reported estimate from the archived evaluation records with source-hash checks, and generated figures and tables from those records. *Research ideation:* they proposed some of the analyses, including the unit-order measurement in Section 4.4 and its robustness checks; the authors decided which analyses to include and how to interpret them. LLM agents also produced exploratory state and outlook annotations reported only in Appendix P, where they are labeled preliminary; no main-text finding depends on them. We did not use generative AI to generate datasets, design tasks, formulate theory, or write proofs, and no AI assistant generated or altered any evaluation record, task outcome, or human-study record. The LLMs and the Jev decision model evaluated in Section 4 are subjects of the study, not assistants. The authors take full responsibility for the content.

References

Jaewoo Ahn, Junseo Kim, Heeseung Yun, Jaehyeon Son, Dongmin Park, Jaewoong Cho, and Gunhee Kim. FlashAdventure: A benchmark for GUI agents solving full story arcs in diverse adventure games. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 23354–23384, 2025. doi: 10.18653/v1/2025.emnlp-main.1192.

Anonymous. StarCraft II Arena: Evaluating LLMs in strategic planning, real-time decision making, and adaptability. Submitted to the International Conference on Learning Representations (ICLR) 2025, 2024. URL https://openreview.net/forum?id=o3V7OuPxu4.

Jacob W. Crandall, Michael A. Goodrich, Dan R. Olsen, Jr., and Curtis W. Nielsen. Validating human–robot interaction schemes in multitasking environments. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 35(4):438–449, 2005. doi: 10.1109/TSMCA.2005.850587.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. MineDojo: Building open-ended

embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, volume 35, pp. 18343–18362, 2022. doi: 10.52202/068431-1333. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/74a67268c5cc5910f64938cac4526a90-Abstract-Datasets_and_Benchmarks.html.

Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, Brian Ichter, Danny Driess, Jiajun Wu, Cewu Lu, and Mac Schwager. Foundation models in robotics: Applications, challenges, and the future. *The International Journal of Robotics Research*, 44(5):701–739, 2025. doi: 10.1177/02783649241281508. URL https://journals.sagepub.com/doi/10.1177/02783649241281508.

Christian Gold, Daniel Damböck, Lutz Lorenz, and Klaus Bengler. "Take over!" How long does it take to get the driver back into the loop? *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 57(1):1938–1942, 2013. doi: 10.1177/1541931213571433.

Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*, pp. 709–715, 2004. URL https://aaai.org/Papers/AAAI/2004/AAAI04-112.pdf.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782, 2023.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998. doi: 10.1016/S0004-3702(98)00023-X.

San Kim, Daechul Ahn, Reokyoung Kim, Hyeonbeom Choi, Seungyeon Jwa, and Jonghyun Choi. RTSGameBench: An RTS benchmark for strategic reasoning by vision-language models. *arXiv preprint arXiv:2606.18950*, 2026. URL https://arxiv.org/abs/2606.18950v2.

Hadas Kress-Gazit, Kunimatsu Hashimoto, Naveen Kuppuswamy, Paarth Shah, Phoebe Horgan, Gordon Richardson, Siyuan Feng, and Benjamin Burchfiel. Robot learning as an empirical science: Best practices for policy evaluation. *arXiv preprint arXiv:2409.09491*, 2024.

Manling Li, Shiyu Zhao, Qineng Wang, Kangrui Wang, Yu Zhou, Sanjana Srivastava, Cem Gokmen, Tony Lee, Li Erran Li, Ruohan Zhang, Weiyu Liu, Percy Liang, Li Fei-Fei, Jiayuan Mao, and Jiajun Wu. Embodied Agent Interface: Benchmarking LLMs for embodied decision making. In *Advances in Neural Information Processing Systems*, volume 37, pp. 100428–100534, 2024. doi: 10.52202/079017-3188. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b631da756d1573c24c9ba9c702fde5a9-Abstract-Datasets_and_Benchmarks_Track.html.

Zeyi Liu, Arpit Bahety, and Shuran Song. REFLECT: Summarizing robot experiences for failure explanation and correction. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3468–3484, 2023.

Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Yuqiao Wu, Runji Lin, Haifeng Zhang, and Jun Wang. Large language models play StarCraft II: Benchmarks and a chain of summarization approach. In *Advances in Neural Information Processing Systems*, volume 37, pp. 133386–133442, 2024. doi: 10.52202/079017-4240. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f0ebc318e2df08360b2df559e81602e5-Abstract-Conference.html.

Julius Mayer, Mohamad Ballout, Serwan Jassim, Farbod Nosrat Nezami, and Elia Bruni. iVISPAR — an interactive visual-spatial reasoning benchmark for VLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26757–26781. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.1359. URL https://aclanthology.org/2025.emnlp-main.1359/.

OpenRA. OpenRA, n.d. URL https://www.openra.net/. Accessed September 22, 2026.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wołczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Foerster, Jack Parker-Holder, and Tim Rocktaeschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f0b1515be276f6ba82b4f2b25e50bef0-Abstract-Conference.html.

Dongmin Park, Minkyu Kim, Beongjun Choi, Junhyuck Kim, Keon Lee, Jonghyun Lee, Inkyu Park, Byeong-Uk Lee, Jaeyoung Hwang, Jaewoo Ahn, Ameya S. Mahabaleshwarkar, Bilal Kartal, Pritam Biswas, Yoshi Suhara, Kangwook Lee, and Jaewoong Cho. Orak: A foundational benchmark for training and evaluating LLM agents on diverse video games. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=H1ncX6O6Yh. arXiv:2506.03610.

Thomas Schmied, Jörg Bornschein, Jordi Grau-Moya, Markus Wulfmeier, and Razvan Pascanu. LLMs are greedy agents: Effects of RL fine-tuning on decision-making abilities. *arXiv preprint arXiv:2504.16078*, 2025.

Wenjie Tang, Yuan Zhou, Erqiang Xu, Keyan Cheng, Minne Li, and Liquan Xiao. DSGBench: A diverse strategic game benchmark for evaluating LLM-based agents in complex decision-making environments. *arXiv preprint arXiv:2503.06047*, 2025. URL https://arxiv.org/abs/2503.06047v2. Extended version; a short version appeared at ICASSP 2026.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017. URL https://arxiv.org/abs/1708.04782.

Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Yixuan Li, and Neel Joshi. Is a picture worth a thousand words? Delving into spatial reasoning for vision language models. In *Advances in Neural Information Processing Systems*, volume 37, 2024. arXiv:2406.14852.

Yue Wu, Xuan Tang, Tom M. Mitchell, and Yuanzhi Li. SmartPlay: A benchmark for LLMs as intelligent agents. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/063d1fadbde557466bf5bcffe814a64e-Abstract-Conference.html. arXiv:2310.01557.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 70576–70631. PMLR, 2025. URL https://proceedings.mlr.press/v267/yang25f.html.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b126cc38b8638e07bef37e7b2bb72bf-Abstract-Conference.html. arXiv:2406.12045.

Alex L. Zhang, Thomas L. Griffiths, Karthik R. Narasimhan, and Ofir Press. VideoGameBench: Can vision-language models complete popular video games? *arXiv preprint arXiv:2505.18134*, 2025a. URL https://arxiv.org/abs/2505.18134.

Zheyuan Zhang, Fengyuan Hu, Jayjun Lee, Freda Shi, Parisa Kordjamshidi, Joyce Chai, and Ziqiao Ma. Do vision-language models represent space and how? Evaluating spatial frame of reference under ambiguities. In *International Conference on Learning Representations (ICLR)*, 2025b.

A PER-FAMILY TASK PERFORMANCE

Tables 5 and 6 use the same twelve-family grouping as Table 1 and Appendix D. Every one of the 217 task identifiers in the pinned model exports maps to exactly one family. Rates pool scored attempts within each family; operational errors are excluded, and attempt coverage differs across models (Appendix J). The arena configuration present in the task exports remains in Tempo & Adversarial; the separate competitive study is reported in Appendix E.

We resample task identifiers within each family 10,000 times (seed 20260922), retaining all their difficulties and recorded attempts. Brackets give pointwise 95% percentile intervals conditional on these records. Intervals are not adjusted for multiple comparisons and do not establish a family-specific ordering of model abilities. The released family inventory, pinned source hashes, and `revisions/iclr2027/compute_appendix_task_summaries.py` reproduce these tables. Appendix K reports the overall estimates and attempt counts.

Table 5: Task success (%) by the twelve task families, with pointwise 95% task-bootstrap intervals. Model group 1 of 2.

Family	Opus 4.7	Sonnet 4.6	Gemma 4 31B	GPT-5.4
Combat Micro	75.6 [64.9, 84.9]	70.7 [58.2, 82.2]	70.7 [59.6, 80.4]	61.3 [49.3, 73.3]
Static Defense	93.6 [88.3, 98.2]	83.6 [71.3, 93.6]	70.8 [56.7, 83.6]	70.2 [54.4, 86.0]
Multi-group Coordination	90.1 [82.7, 96.3]	82.7 [65.4, 95.1]	77.8 [60.5, 92.6]	77.8 [59.3, 92.6]
Resource Economy	75.4 [62.7, 86.5]	61.5 [48.4, 74.6]	65.9 [54.8, 76.6]	58.3 [44.0, 72.6]
Build & Tech	78.4 [63.7, 91.2]	67.8 [51.5, 83.0]	54.7 [37.4, 71.6]	57.9 [38.6, 77.2]
Special Units	92.6 [86.1, 98.1]	80.6 [64.8, 93.5]	76.9 [60.2, 89.8]	80.6 [58.3, 100.0]
Scouting & Vision	89.5 [76.5, 98.8]	88.9 [75.3, 98.1]	74.1 [58.0, 88.3]	68.5 [50.0, 85.2]
Indirect Strategy	55.6 [35.6, 75.6]	32.2 [11.1, 55.6]	32.2 [8.9, 57.8]	33.3 [13.3, 53.3]
Long-Horizon Play	88.4 [75.7, 97.9]	79.9 [68.8, 90.5]	52.9 [38.1, 67.7]	69.8 [55.6, 84.1]
Tempo & Adversarial	79.2 [60.5, 93.6]	66.7 [44.9, 87.2]	72.0 [53.3, 88.0]	62.5 [42.9, 82.6]
Multi-front Resilience	83.0 [74.8, 90.4]	78.5 [70.0, 86.7]	70.4 [60.7, 79.6]	63.3 [53.3, 73.3]
Procedure Robustness	82.4 [68.0, 94.1]	66.7 [48.4, 83.7]	59.5 [42.5, 75.8]	63.5 [43.4, 81.8]

Table 6: Task success (%) by the twelve task families, with pointwise 95% task-bootstrap intervals. Model group 2 of 2.

Family	GPT-5.4 mini	Qwen 3.6-35B	Qwen 3.5-9B	Kimi K2.5
Combat Micro	54.7 [38.7, 70.7]	41.8 [29.3, 54.7]	44.0 [29.3, 58.7]	30.2 [17.8, 43.6]
Static Defense	63.2 [43.9, 80.7]	53.2 [37.4, 69.0]	52.6 [36.8, 68.4]	35.7 [17.5, 55.0]
Multi-group Coordination	59.3 [37.0, 81.5]	63.0 [37.0, 86.4]	63.0 [33.3, 92.6]	63.0 [42.0, 82.7]
Resource Economy	32.1 [20.2, 44.0]	40.1 [28.2, 52.8]	21.4 [10.7, 34.5]	38.1 [23.4, 53.2]
Build & Tech	52.6 [35.1, 70.2]	27.5 [14.0, 42.7]	31.6 [17.5, 47.4]	16.4 [4.1, 31.6]
Special Units	80.6 [61.1, 97.2]	72.2 [60.2, 85.2]	69.4 [52.8, 83.3]	63.0 [46.3, 77.8]
Scouting & Vision	59.3 [42.6, 75.9]	67.9 [51.8, 81.8]	68.5 [55.6, 81.5]	56.8 [39.5, 73.5]
Indirect Strategy	33.3 [13.3, 53.3]	15.6 [1.1, 36.7]	26.7 [10.0, 46.7]	13.3 [3.3, 24.4]
Long-Horizon Play	46.0 [30.2, 61.9]	35.4 [20.1, 51.3]	41.3 [27.0, 55.6]	25.0 [10.7, 41.3]
Tempo & Adversarial	33.3 [12.0, 56.0]	54.7 [33.3, 75.4]	54.2 [26.1, 80.0]	33.3 [8.6, 61.1]
Multi-front Resilience	54.4 [42.2, 66.7]	43.0 [30.0, 55.9]	42.2 [28.9, 55.6]	38.0 [24.0, 52.4]
Procedure Robustness	54.9 [35.3, 74.5]	58.8 [41.2, 75.8]	54.9 [35.3, 72.5]	52.3 [32.7, 71.2]

B Repeated Attempts and Trajectory Records

Matching and coverage. We group the pinned historical exports used in Appendix K by task, difficulty, seed, and split, retaining exactly one scored record at each repeat index 0, 1, and 2. These indices correspond to Attempts 1–3 in Table 3. An entire combination is excluded if an attempt is missing, contains an operational error, duplicates a repeat index, or conflicts with another attempt's explicit observation or handoff setting. All retained combinations use seed 1. Coverage is 648 combinations each for Opus and Sonnet, 646 for Gemma, 644 for Qwen 3.6, and 643 for Kimi. The remaining three models lack complete three-attempt coverage. Selection does not depend on the outcome.

Task success and individual outcomes. Task success is the fraction of retained combinations with outcome *win*; draws remain in the denominator. The range is the highest minus lowest of the three success rates, calculated before rounding. Individual tasks need not have the same result on every attempt: among Opus's 648 combinations, 478 are wins on all three attempts, 69 are losses on all three, and 101 have a mixture of win/loss/draw results. Table 8 reports the complete outcome-pattern counts, separately from the aggregate success rates.

Uncertainty. We resample tasks 10,000 times (random seed 20260921), retaining all difficulties and all three attempts for each sampled task. Each draw recomputes the success rates and paired differences. Pointwise 95% intervals use the 2.5th and 97.5th percentiles; paired differences use the same task draws for both attempts (Table 9). There is no multiplicity adjustment. Intervals describe variation across tasks conditional on recorded attempts, not the distribution of fresh model generations. The max-minus-min range is descriptive; confidence intervals refer to the individual rates and specified paired differences. Small observed ranges and intervals containing zero do not establish equivalence. Because retained combinations differ across models, the repeated-attempt comparisons are within each model.

Table 7: Task success (%) across three recorded attempts, with 95% task-bootstrap intervals. n counts matched task–difficulty combinations. Range is the highest minus lowest success rate in percentage points (pp), computed before rounding.

Model	n	**Attempt 1**	**Attempt 2**	**Attempt 3**	**Range (pp)**
Opus 4.7	648	82.3 [78.6, 85.8]	83.0 [79.3, 86.6]	81.0 [77.1, 84.9]	2.0
Sonnet 4.6	648	71.9 [67.1, 76.5]	71.9 [67.3, 76.5]	74.2 [69.6, 78.6]	2.3
Gemma 4 31B	646	63.9 [59.3, 68.4]	65.2 [60.2, 70.0]	66.1 [61.2, 70.7]	2.2
Qwen 3.6-35B	644	44.9 [39.5, 50.1]	45.7 [40.5, 50.6]	46.7 [41.4, 52.0]	1.9
Kimi K2.5	643	37.8 [32.4, 43.3]	38.4 [32.7, 44.0]	37.0 [31.6, 42.4]	1.4

Table 8: Counts of three-run outcome patterns; n is the number of retained task–difficulty combinations. W, L, and D denote win, loss, and draw. WWW, LLL, and DDD are consistent outcomes. The remaining columns count instances containing exactly the indicated set of outcomes, regardless of run order; for example, W/L contains WWL and WLL and their permutations, but no draws. The seven outcome columns partition the retained instances.

Model	n	**WWW**	**LLL**	**DDD**	**W/L**	**W/D**	**L/D**	**W/L/D**
Opus 4.7	648	478	69	0	94	6	1	0
Sonnet 4.6	648	391	94	2	147	7	6	1
Gemma 4 31B	646	329	132	4	166	9	5	1
Qwen 3.6-35B	644	210	244	2	167	5	10	6
Kimi K2.5	643	172	319	0	146	1	4	1

Table 9: Paired differences in task success (percentage points), with 95% task-bootstrap intervals. Identical task draws are used for both attempts in each contrast. These pointwise intervals do not establish equivalence and are not adjusted for multiple comparisons.

Model	Attempt 2 minus 1	Attempt 3 minus 1	Attempt 3 minus 2
Opus 4.7	+0.8 [-1.5, +3.1]	-1.2 [-3.6, +1.1]	-2.0 [-4.5, +0.5]
Sonnet 4.6	+0.0 [-3.4, +3.3]	+2.3 [-0.3, +5.0]	+2.3 [-0.8, +5.4]
Gemma 4 31B	+1.2 [-2.3, +4.8]	+2.2 [-1.1, +5.5]	+0.9 [-1.9, +3.7]
Qwen 3.6-35B	+0.8 [-2.8, +4.5]	+1.9 [-1.4, +5.2]	+1.1 [-1.9, +4.1]
Kimi K2.5	+0.6 [-2.8, +4.1]	-0.8 [-3.9, +2.3]	-1.4 [-4.5, +1.7]

Historical settings and sensitivity. Missing settings do not establish identical conditions: 357 Gemma combinations and all 644 Qwen 3.6 combinations lack observation metadata for at least one attempt. Complete temperature and version histories are also unavailable. We therefore do not attribute all variation to stochastic decoding. Removing the one explicitly text-only combination retained by each of Opus, Sonnet, and Kimi leaves success-rate ranges of 1.4–2.3 percentage points. Restricting all five models to their 638 shared combinations gives ranges of 1.4–2.5 points.

Trajectory provenance. Figure 4 uses three recovered Qwen 3.6 playbacks for `adv-rps-counter-pick:medium`, seed 1. Runs A/B/C identify archived run IDs `20260525-081001`, `20260525-152307`, and `20260525-155432`. Their manifests record the same Together FP8 endpoint and outcomes win/loss/loss after 36/60/60 decisions. The six manifest and turn files are preserved with SHA-256 hashes in the released analysis records. These are illustrative historical attempts; the manifests do not independently establish repeat-index assignments, identical observation settings, or immutable engine versions.

Measurements and score independence. The plotted enemy-kill count is `signals.units_killed`, checked against the `units_killed_gte` leaf's current value in every decision record. All three runs have the same target of eight. Their terminal counts are nine, three, and three. For all 156 decisions, the legacy objective ratio happens to equal $\min(q_t/8, 1)$, where q_t is that recorded count; the two failed runs therefore have ratio $3/8 = 37.5\%$. This ratio is not the weighted composite or a P/R/A score. It is also not a probability of success or a general distance to completion: both failed runs retain ratio 0.375 when their deadline expires at tick 5403, beyond the limit of 5400. Figure 4 therefore plots the raw count, together with the target, rather than the ratio. The preserved-base predicate remains satisfied throughout these records. Run A satisfies the full win condition at tick 3243.

Temporal alignment and interpretation. Each turn row stores commands and subsequent kill counts and goal measurements, but its unit snapshot precedes those commands. The figure therefore takes kills after decision t from row t and the resulting units from row $t + 1$. Each file's extra terminal row supplies the last resulting state and is excluded from decision counts. The initial unit count comes from the first pre-action snapshot; enemy-kill curves begin at the first recorded post-action count. Run A first requests movement at turn 3 and production at turn 5. In Run B, the kill count reaches three after turn 35, when only a non-attacking resource collector remains; turns 36–59 request observation only, followed by a collector-movement request on turn 60. Run C reaches three kills after turn 37, with no units in the following state. It requests resource-collector production on turns 39, 41, 42, 45, 46, and 59; recorded cash remains zero and no units subsequently appear. These are requests, not confirmations of successful production. The figure counts current units directly, rather than using the adapter's unit-loss proxy. The examples establish behavior and state, without proving that recovery was feasible, identifying an internal cognitive cause, or estimating prevalence.

C Task-Level Outcome Coverage

The eight model exports share 648 task–configuration entries: 216 easy, 216 medium, 215 hard, and one `main` entry for `adversarial-1v1-macro`. This is not a complete 216×3 difficulty grid (Appendix J). For each model and entry, we average the win indicator over its scored attempts, retaining losses and draws in the denominator. The following categories are mutually exclusive and exhaustive:

- *High success*: all eight observed rates are at least 90% (99/648; 15.3%).
- *Low success*: all eight observed rates are at most 20% (26/648; 4.0%).
- *Remaining*: neither threshold holds for all eight models (523/648; 80.7%).

These thresholds describe the recorded cohort, not intrinsic task difficulty or proof of a particular missing ability. Models with one scored attempt have binary observed rates, while models with repeated attempts have finer rate resolution. Historical setting differences further limit comparisons. The remaining category need not discriminate between models: equal intermediate rates also fall into it. We do not infer no-shortcut validation, reasoning failure, or task quality from these categories. The released appendix-summary script saves every entry's category and model-specific rate.

D COMPLETE TASK INVENTORY

Every active task in Red Alert Bench v1.1, grouped by task family. Each task declares three difficulty entries (*easy*, *medium*, *hard*); evaluated seeds and distinct configurations depend on the protocol. In particular, arena-style level aliases need not define different difficulties. Task names are stable identifiers used in the released artefacts at `openra_bench/scenarios/packs/<pack>.yaml`. Numbers in parentheses give the per-family count (217 active tasks total; one template file and three quality-archived tasks are excluded). Families group tasks by primary demand, so a task's name prefix need not match its family: for example, the seven `tp-*` timing tasks fall under Special Units as timed interventions (Table 1).

COMBAT MICRO (25 TASKS)

`combat-attack-from-behind-fog`, `combat-bait-counter-attack`, `combat-divide-and-conquer`, `combat-flanking-attack`, `combat-focus-fire-priority`, `combat-formation-tank-wedge`, `combat-harass-aggro-commit`, `combat-harass-balanced-hit-and-run`, `combat-heli-flank`, `combat-hold-chokepoint`, `combat-kite-and-pull`, `combat-kite-jeep-vs-tank`, `combat-naval-shore-strike`, `combat-pincer-coordination`, `combat-prevent-retreat`, `combat-protect-vip-escort`, `combat-retreat-after-engagement`, `combat-rocket-soldier-anti-vehicle`, `combat-skirmish-then-disengage`, `combat-stance-mgmt-attack`, `combat-suicide-charge-mission`, `combat-tank-vs-tank-engagement`, `combat-tanya-vs-rush`, `combat-target-priority-highvalue`, `combat-vehicle-vs-infantry-counter`.

STATIC DEFENSE (19 TASKS)

`def-bridge-chokepoint`, `def-counter-battery`, `def-engineer-repair-under-fire`, `def-evacuation`, `def-in-depth`, `def-in-depth-vs-single`, `def-multi-direction`, `def-position-expected-direction`, `def-position-revealed-direction`, `def-pre-position-mobile-reserve`, `def-reinforce-the-breach`, `def-retreat-and-rebuild`, `def-stance-mgmt-hold-then-attack`, `def-surprise-flank-react`, `def-tower-line-vs-cluster`, `def-walls-vs-towers`, `def-while-building`, `def-with-ambush`, `defense-rush-survive`.

MULTI-GROUP COORDINATION (9 TASKS)

`coord-converge-on-target`, `coord-cover-and-move`, `coord-diversionary-attack`, `coord-mutual-support`, `coord-relay-attack`, `coord-relay-vision-chain`, `coord-squad-handoff`, `coordination-ordered-rendezvous`, `coordination-staggered-window`.

RESOURCE ECONOMY (28 TASKS)

`econ-burn-rate-management`, `econ-buy-vs-build-decision`, `econ-cash-reserve-management`, `econ-contention-with-enemy`, `econ-contested-expansion`, `econ-deny-enemy-expansion`, `econ-expansion-timing`, `econ-far-patch-vs-near-patch`, `econ-harvester-defense-raid`, `econ-harvester-pathing-optimization`, `econ-mine-and-grow`, `econ-multi-patch-allocation`, `econ-overflow-to-silos`, `econ-protect-harvester-route`, `econ-quantitative-vs-qualitative-spend`, `econ-recover-from-zero-cash`, `econ-replace-dead-harvester`, `econ-resource-trade-with-self`, `econ-second-base-race`, `econ-silo-vs-spend`, `econ-startup-from-scratch`, `econ-target-cash-amount-by-deadline`, `econ-tech-vs-expand-decision`, `economy-force-buildup`, `economy-harvest-investment`, `economy-harvest-timebox`, `economy-investment`, `economy-time-box`.

BUILD & TECH (19 TASKS)

build-defensive-skirt-corners, build-defensive-tower-cluster, build-defensive-tower-line, build-engineer-rebuild-after-loss, build-power-down-defensive, build-power-online-first, build-production-throughput-multibuilding, build-rally-point-management, build-repair-priority-under-fire, build-sell-and-rebuild-elsewhere, build-sequence-tech-cheapest, build-sequence-tech-fastest, build-sequence-tech-most-resilient, build-tech-skip-decision, building-and-planning, tech-aggro-all-in, tech-balanced-econ-then-tech, tech-production-planning, tech-turtle-defensive-tech.

SPECIAL UNITS (12 TASKS)

spec-engineer-capture, spec-nuke-strike, spec-spy-infiltrate, spec-tanya-c4-strike, spec-thief-steal-cash, tp-decision-under-clock, tp-pressure-procedural, tp-rush-multi-objective, tp-rush-objective-very-fast, tp-survive-and-grow, tp-survive-and-strike-at-window, tp-survive-n-turns.

SCOUTING & VISION (18 TASKS)

perception-count-the-threat, perception-count-the-threat-small-k, perception-frontier-reading, perception-target-vs-fog, scout-and-report, scout-and-survive, scout-count-defenders, scout-cycle-keep-info-fresh, scout-deny-enemy-vision, scout-detect-base-direction, scout-detect-enemy-tech, scout-detect-incoming-army, scout-discover-hidden-base, scout-far-frontier, scout-jeep-vs-infantry-cost-effective, scout-map-reveal-percent-target, scout-multiple-fog-areas, scout-track-enemy-movement.

INDIRECT STRATEGY (10 TASKS)

artofwar-indirect-approach, artofwar-lure-the-tiger, artofwar-sequenced-citadel, reasoning-frontier-commit, reasoning-risk-route, risk-blockade-bypass, strategy-dilemma, strategy-gauntlet, strategy-trilemma, strategy-twobody.

LONG-HORIZON PLAY (21 TASKS)

f11-defense-then-counter, f11-econ-tech-army-strike, f11-full-combined-arms, f11-pivot-on-scout, f11-rebuild-after-attrition, f11-vertical-strike-ground-air, f11-vertical-strike-naval, lh-100-turn-marathon-survival, lh-build-army-coordinate-multifront-attack, lh-credit-only-final-phase, lh-defense-tech-second-base, lh-econ-army-victory, lh-multi-checkpoint-5-plus, lh-opening-to-defense-to-counter, lh-opening-to-tech-to-army, lh-progression-stage-locked, lh-recovery-after-mid-game-loss, lh-scout-react-counter, lh-tech-pivot-attack, lh-tech-rush-vs-army-rush, longhorizon-opening-to-assault.

TEMPO & ADVERSARIAL (9 TASKS)

adv-asymmetric-weaker-must-win, adv-rps-counter-pick, adversarial-1v1-macro, adversarial-duel, expansion-aggro-3-base-greedy, expansion-balanced-2-base-defended, expansion-turtle-1-base-fortified, tempo-double-window, tempo-strike-window.

MULTI-FRONT RESILIENCE (30 TASKS)

harass-response-preserve, maint-repair-priority-order, maint-sell-and-recoup-cash, mcv-deploy-and-build, mcv-deploy-defensible-site,

mcv-deploy-near-resource, mcv-deploy-relocate-under-pressure, mcv-deploy-second-base, mcv-deploy-third-base, mfb-base-1-defend-base-2-build, mfb-mirror-base-east-west, mfb-redundant-tech-buildings, mfb-rotating-production-pressure, mfb-supply-line-link-between-bases, mfb-tech-base-vs-economy-base, mfb-third-base-against-clock, mfb-two-base-simultaneous, mid-concede-vs-hold, mid-economy-under-fire, mid-tech-switch-on-scout, navigation-confined-hard-only, rob-cash-depletion-recovery, rob-deadline-shortened-midway, rob-multiple-simultaneous-pressures, rob-objective-change-midway, rob-objective-shift-with-or-clause, rob-partial-base-loss-continue, rob-unexpected-enemy-spawn, rob-unit-loss-recovery, rush-hour.

PROCEDURE ROBUSTNESS (17 TASKS)

action-multiunit-coordination, action-sequenced-execution, custom-map-no-enemy, power-budget-online, proc-checklist-no-deviation, proc-conditional-branch-action, proc-instruction-following-edge-case, proc-no-attack-passive-only, proc-only-build-no-combat, proc-only-defend-no-attack, proc-ordered-action-strict, proc-strict-toolban-fidelity, proc-tool-use-multi-distractor, proc-tool-use-with-distractor, strict-production-bom, strict-sequence, strict-toolban-fidelity-under-pressure.

E ADVERSARIAL RECORDS, OUTCOMES, AND UNCERTAINTY

Table 10: Outcomes of complete two-game matches against other models. Match score is $(W + 0.5D)/(W + D + L)$. Intervals use 10,000 bootstrap resamples of opponent-pair/seed groups. Opponent coverage differs; these pooled scores are descriptive, not a schedule-adjusted ranking.

Model	Wins	Draws	Losses	Match score (%; 95% CI)
Opus 4.7	22	9	0	85.5 [75.9, 93.9]
Sonnet 4.6	7	19	4	55.0 [43.5, 66.7]
Gemma 4 31B	4	11	9	39.6 [25.0, 54.3]
GPT-5.4	5	24	9	44.7 [35.9, 54.4]
GPT-5.4-mini	8	22	9	48.7 [37.1, 59.6]
Qwen 3.6-35B	6	6	12	37.5 [20.5, 54.5]
Qwen 3.5-9B	3	14	11	35.7 [23.1, 48.2]
Kimi K2.5	6	11	7	47.9 [33.3, 62.5]

Record selection. The archive contains 388 completed attempts. Retaining the latest completion per model pair, configuration label, position assignment, and seed gives 359 games. Three settings have unmatched historical playback exports; one also belongs to a pair whose Gemma endpoint differs between halves. Excluding these settings leaves 356 games: 119 complete cross-model pairs, 45 same-model pairs, and 28 unpaired games. Only the cross-model pairs enter Figure 5 and Table 10. The full 122-pair comparison remains a sensitivity analysis.

Outcomes and uncertainty. Winning both games gives a paired win; all 58 paired draws are one win apiece. The matrix displays counts. Match scores count wins as one and draws as one half. We resample model-pair/seed groups 10,000 times (bootstrap seed 20260921), retaining all

configuration labels and both assignments within each group. Intervals are the 2.5th and 97.5th percentiles, conditional on the recorded agents and observed schedule; they do not adjust for unequal opponents or missing historical settings.

Position evidence. Initial construction-yard coordinates verify positions for 194 primary games; the east side wins 132. Among 28 verified same-model games, it wins 21. Position intervals use model-pair/seed clustering. In the full set, the east side wins 135 of 197 directly verified games (68.5%; 95% CI: 62.6–74.6%). Inferring other positions from the observed seed mapping gives 251/359 (69.9%; 64.9–74.8%). That interval excludes reconstruction uncertainty. Verified records are not a random sample of the archive.

Historical settings. The arena is `adversarial-1v1-macro`, with seeds 1–4. The retained configuration maps `easy`, `medium`, and `hard` to the same settings; these labels do not establish distinct difficulties. Providers differ across batches, including Qwen 3.6's Together FP8 and `qwen/qwen3.6-35b-a3b` endpoints and Gemma service variants. Some early attempts have different southern-base coordinates and termination behavior. Configuration snapshots are incomplete; the results describe archived agent configurations rather than controlled comparisons of immutable weights.

Table 11: Per-side command activity and production in the 194 primary-cohort games whose per-turn records are preserved for both players. Traced sides counts each model's participations (both sides in same-model games). Waiting turns contain only `Observe`. Unit requests are `Build` commands for mobile units; fielded units are distinct new unit identifiers observed during the game. Conversion rates pool all traced games of a model. Unaffordable orders request an item costing more than the side's funds after the previous turn; repeated orders request an item also ordered on the previous turn. Coverage is complete for the three Bedrock-hosted models and partial for the others, whose historical seed-2–4 games lack per-turn records.

Model	Traced sides	Waiting turns (%)	Unit requests	Units fielded	Fielded per request (%)	Structures per request (%)	Unaffordable orders (%)	Repeated orders (%)
Opus 4.7	77/77	18	28	15	40	32	15	30
Sonnet 4.6	77/77	5	16	10	48	72	16	32
Gemma 4 31B	32/94	17	58	8	10	11	8	93
GPT-5.4	35/127	0	41	13	28	22	44	43
GPT-5.4-mini	37/133	1	94	7	10	7	54	72
Qwen 3.6-35B	40/71	90	7	3	35	3	57	52
Qwen 3.5-9B	32/75	1	570	11	2	1	85	77
Kimi K2.5	58/58	88	14	9	68	23	1	0

Per-turn records and trajectory statistics. Both players' per-turn records are preserved as Git blobs for 194 of the 356 primary games: all 77 Opus, 77 Sonnet, and 58 Kimi sides, and 32–40 of the 71–133 sides of the other five models, whose historical seed-2–4 games lack playback exports. A waiting turn contains only `Observe`. Unit and structure requests count `Build` commands by item type; fielded units and added structures count distinct new identifiers observed in the side's unit and building lists, so they measure realized production rather than acknowledged requests. An order is unaffordable when the item's cost in the pinned Red Alert rules exceeds the side's funds after the previous turn, and repeated when the same item was also ordered on the previous turn. The elimination turn is the first turn at which the loser's building list is empty; this turn exists in all 137 eliminated games and agrees with the recorded termination reason. The lead turn is the first turn after which the winner's building count exceeds the loser's for the remainder of the game; 27 of the 57 deadline games have no such turn. Coverage is complete for the Bedrock-hosted models and partial for the others, so per-model statistics for the latter describe mainly their games against the Bedrock batch. `revisions/iclr2027/compute_arena_trajectories.py` regenerates Table 11 and `arena_trajectory_results.json` from the merged records and trace blobs.

Execution example and reproduction. The Qwen 3.6–Opus example is `20260526-035107`, normal assignment, seed 1. Both sides have 46 decisions and saved responses. Qwen's responses describe production and scouting, but all tool-call lists are empty and every command batch is `Observe`. The archive does not isolate model, provider, token-budget, or adapter causes. The release preserves

source Git objects and per-game provenance; `revisions/iclr2027/build_arena_results.py` checks both players' records, winner mapping, paired outcomes, and bootstrap estimates, and regenerates the tables.

Table 12: Completed two-game matches in the primary comparison. Multiple matches at one seed come from historical configuration labels; the labels do not establish distinct arena difficulty levels. Zero means no complete retained match.

Model	Opponent	Seed 1	Seed 2	Seed 3	Seed 4	Total
Opus 4.7	Sonnet 4.6	2	2	1	1	6
Opus 4.7	Gemma 4 31B	2	1	1	1	5
Opus 4.7	GPT-5.4	1	1	1	1	4
Opus 4.7	GPT-5.4-mini	2	1	1	1	5
Opus 4.7	Qwen 3.6-35B	2	1	1	1	5
Opus 4.7	Qwen 3.5-9B	1	1	1	1	4
Opus 4.7	Kimi K2.5	1	1	0	0	2
Sonnet 4.6	Gemma 4 31B	1	1	1	1	4
Sonnet 4.6	GPT-5.4	1	1	1	1	4
Sonnet 4.6	GPT-5.4-mini	1	1	1	1	4
Sonnet 4.6	Qwen 3.6-35B	1	1	1	1	4
Sonnet 4.6	Qwen 3.5-9B	1	1	1	1	4
Sonnet 4.6	Kimi K2.5	1	1	1	1	4
Gemma 4 31B	GPT-5.4	1	1	1	1	4
Gemma 4 31B	GPT-5.4-mini	1	1	1	1	4
Gemma 4 31B	Qwen 3.6-35B	0	0	0	0	0
Gemma 4 31B	Qwen 3.5-9B	1	1	1	0	3
Gemma 4 31B	Kimi K2.5	1	1	1	1	4
GPT-5.4	GPT-5.4-mini	3	3	3	3	12
GPT-5.4	Qwen 3.6-35B	2	1	1	1	5
GPT-5.4	Qwen 3.5-9B	2	2	1	1	6
GPT-5.4	Kimi K2.5	1	1	1	0	3
GPT-5.4-mini	Qwen 3.6-35B	1	1	1	1	4
GPT-5.4-mini	Qwen 3.5-9B	2	2	1	1	6
GPT-5.4-mini	Kimi K2.5	1	1	1	1	4
Qwen 3.6-35B	Qwen 3.5-9B	1	1	0	0	2
Qwen 3.6-35B	Kimi K2.5	1	1	1	1	4
Qwen 3.5-9B	Kimi K2.5	1	1	1	0	3

Table 13: Sensitivity to historical record selection. Position estimates use only matched playback evidence. Earliest attempts can also differ in historical geometry and endpoints; the rows are not controlled interventions on selection alone.

Record selection	Cross-model paired matches	Games with verified positions	East-side wins (%; 95% CI)
Exclude unmatched exports (main)	119	194	68.0 [62.0, 74.2]
Full latest-attempt set	122	197	68.5 [62.6, 74.6]
Earliest completed attempts	122	198	65.7 [59.4, 72.0]
Exclude repeated settings	112	175	67.4 [60.8, 73.9]

F Formal Model of Agent–Environment Interaction

We instantiate the partially observable stochastic-game framework (Hansen et al., 2004) for the benchmark's decision interface. Authored tasks are its single-controller special case, a partially observable

Markov decision process (Kaelbling et al., 1998). This specification expands the interaction model in Equation 1 and defines its evaluation payoffs.

Configuration and state. Configuration c specifies the map and initialization, game and task rules, observation conditions, stepping protocol, and decision budget H_c. The initial state has distribution ρ_c; a fully fixed initialization is a point mass. State s_t includes positions and attributes of units and buildings, resources, pending orders and production, the simulation clock, random-generator state, scheduled events, per-player visibility memory, and task-condition monitors. Including these variables makes future transitions and condition updates depend on the current state and commands. The agent receives only the configured observation, rather than this full state. Full-map visibility removes map hiding but need not expose every simulator variable.

Policies and joint actions. A policy maps the agent's supplied context and observation–action history h_t^i to a distribution over command batches. It may retain only a bounded history or use internal memory. Each observation function $\mathcal{O}_i^{m_i,v_i}$ incorporates the selected representation and that player's visibility. In a two-agent game, both controllers act before the shared step; neither receives the other's current submission before choosing its own. For authored tasks, scripted opponents and events are included in $\mathcal{P}_c$. The command space describes attempted actions: unaffordable or otherwise invalid commands need not have the requested effect. Command handling, ongoing orders, and subsequent world evolution belong to the transition.

Decision time and simulation time. Let k_t be the simulator clock at decision turn t. The duration Δ_t in Equation 1 satisfies $k_{t+1} = k_t + \Delta_t$. Fixed-interval and event-triggered stepping are represented by the joint distribution over the next state and duration. A run can be written $\xi = (s_0, \mathbf{a}_0, \Delta_0, \ldots, s_T)$ with $T \leq H_c$; its stopping rule follows the configured completion, failure, termination, and budget checks. Simulation ticks and decision turns are distinct, and the world pauses during agent inference. Seeded deterministic simulation is a special case of $\mathcal{P}_c$. The full state sequence in this mathematical description is distinct from the fields retained in execution records (Appendix I).

Task and competitive payoffs. Let $\Phi_c(\xi) \in \{0, 1\}$ denote recorded task success under the configured evaluator, and let $w_i(\xi)$ be agent i's individual-game outcome. Then

$$U_i^c(\xi) = \begin{cases} \Phi_c(\xi), & \text{authored task,} \\ \mathbf{1}\{w_i(\xi) = \text{win}\} - \mathbf{1}\{w_i(\xi) = \text{loss}\}, & \text{adversarial game.} \end{cases} \tag{2}$$

The task evaluator can combine spatial, resource, survival, timing, and ordered sub-goal conditions. These retain their configured logical and temporal semantics in Φ_c. Individual-game payoffs are zero-sum, with $U_1^c = -U_2^c$. For a complete run ξ, the expected payoff is

$$J_i(\pi_i, \pi_{-i}; c) = \mathbb{E}_{\xi \sim (\pi_i, \pi_{-i}, c)}[U_i^c(\xi)]\,, \tag{3}$$

where π_{-i} is the opponent's policy, absent for authored tasks. Thus J_i is task success probability or, for an individual game, win probability minus loss probability. This encodes evaluation outcomes; the native simulator reward field currently returns zero (Appendix G). Behavioral diagnostics remain separate measurements.

Opponent-dependent evaluation. Let $q(c, \pi_{-i})$ be an evaluation distribution over configurations and opposing policies. Its aggregate expected payoff is

$$\overline{J}_i(\pi_i; q) = \sum_{c, \pi_{-i}} q(c, \pi_{-i}) J_i(\pi_i, \pi_{-i}; c). \tag{4}$$

Thus a competitive comparison depends on the opponents and their weights, as well as maps, seeds, starting positions, and observation settings. A policy favoring early attacks can induce different outcomes against an expanding opponent and a defensive opponent; the model does not impose a universal ordering of strategies. Appendix J describes competitive-record selection and coverage limitations. The paired-match rule in Section 2.4 is applied to the two recorded game outcomes. We evaluate observed policies and outcomes, without estimating equilibria, best responses, or exploitability. Observation comparisons likewise assess complete interfaces, not a numeric value of information or an isolated internal reasoning mechanism.

G ENGINE AND INTERFACE DETAILS

Packaging and version pinning. The simulation core is distributed as a Python wheel built with `maturin`. Unit, weapon, and building data are pinned to upstream OpenRA at commit `0938a27` (bleed branch). The YAML data are embedded at build time using `include_str!` and parsed at runtime, so the wheel does not require an external OpenRA checkout. Actor and weapon statistics flow through a common rules-loading entry point. Reproduction requires fixing both the benchmark and engine versions, together with the task configuration, seed, and action sequence.

Controller interface and tool feedback. The Python evaluation loop calls `act(observation, Command)` on each controller and passes the returned command batch to the Rust simulator through native PyO3 bindings. The supplied LLM adapter converts structured tool calls into command objects and appends a tool reply containing ok before the environment step. This reply acknowledges the call; it does not confirm successful execution. Agents must use subsequent observations to assess the effects of their commands.

Interaction history and context length. The LLM adapter sends the system prompt, the last 16 turns of interaction, and only the current turn's map image; older turns are dropped rather than summarized. Estimated from the request text (characters divided by four, without the image or tool schemas) for the five models whose playback is in the production archive, the request size grows quickly through turn 16, when the history cap takes effect, and more slowly afterwards; after turn 50 the median request is 5.9–7.8k tokens and the 90th percentile at most 13.3k, well below every evaluated model's context limit. Success is lower on tasks with longer turn budgets, but these tasks concentrate in harder families: within the same family the difference is $+0.5$ to $+3.8$ points for Opus, Sonnet, and Gemma and -3.2 to -5.4 points for the other five models. Late-game failures are therefore unlikely to reflect context exhaustion under this adapter; a comparison against full-history prompting was not run. `revisions/iclr2027/reviewer_analyses/context_growth.py` reproduces these measurements.

Rollout interface and rewards. Each native environment owns an independent world initialized from a task or match configuration and seed. The Python `reset` call restores that initial world; `step` accepts a batch of commands and returns an observation, a reward field, a termination flag, and diagnostic information. For adversarial games, `step_1v1` accepts both agents' command batches and returns a separate observation for each side, together with termination and diagnostic information. The native reward field currently returns zero. Task-specific completion checks and competitive outcome evaluation are applied in Python, where training code can construct rewards from these outcomes or other state signals. Environment collection and policy optimization are separate components. Per-turn records also provide turn-level supervision for specific failures, such as unit orders repeated while earlier orders are pending (Section 4.4), which training code can target directly.

Simulation timing. In the default non-interrupt stepping configuration, one decision advances thirty calls to the engine frame processor, each advancing three simulation ticks, for ninety ticks in total. Initialization can add ticks before the first command, so deadlines and elapsed time are read from the recorded engine tick rather than inferred from turn count alone. Interrupt mode can return earlier on configured events. Neither mode advances the world during model inference; in two-player evaluation both command submissions precede the shared step.

Observation comparisons. Task goals and game rules remain fixed across observation conditions, but text and image representations differ in their spatial descriptions and unit/building reference conventions. Comparisons therefore measure sensitivity to the observation interface as a whole; they do not isolate a perceptual ability. Recovering the exact historical model input additionally depends on the recording configuration, as detailed in Appendix I.

Task controls. Scheduled events can spawn or destroy actors and shorten deadlines during a run. Per-side termination flags control whether the simulation ends automatically when a side loses its forces, allowing task-specific conditions to govern completion. The two-player command channel scopes orders to each player and returns observations subject to that player's visibility.

RTS terminology. A *rush* commits early resources to a fast wave of cheap attackers; a *tech tree* is the dependency graph from basic buildings and units to advanced ones; *kiting* is attacking while retreating from slower units; *base elimination* means losing all of a player's buildings. A Mobile Construction Vehicle (MCV) deploys into a Construction Yard, which enables base construction.

Unit and building identifiers. Transcripts use the engine's short YAML identifiers: `powr` (power plant), `proc` (ore refinery), `weap` (war factory, builds vehicles), `tent` (barracks, builds infantry), `hpad` (helipad), `dome` (radar dome), `harv` (harvester), `pbox` (pillbox: static defence), `e1` (rifle infantry), `e3` (rocket soldier), `1tnk`/`2tnk` (light/medium tank), `jeep` (fast scout vehicle), `hind` (helicopter gunship), and `MCV` (Mobile Construction Vehicle).

H Task Configuration and Validation

Task configuration. A task configuration is stored in a YAML file containing capability metadata (`meta`), a map specification (`base_map`), a shared initial configuration (`base`), and three difficulty entries (`levels`). Metadata records the intended capability, its real-world interpretation, a robotics analogue, and benchmark anchors. Level-specific overrides are recursively merged with the shared configuration to set factions, resources, initial units and buildings, and available commands. Each level declares `win_condition`, `fail_condition`, and `max_turns`. The predicate grammar includes logical composition (`all_of`, `any_of`, `not`) and temporal sequencing (`then`), with leaf predicates for spatial, combat, economic, production, and discovery objectives. Actor-level `spawn_point` groups support seed-dependent starting configurations; tasks with documented exceptions, such as symmetric 1v1 arenas, use other difficulty controls.

Validation coverage. The suite-wide validator runs an `Observe`-only controller across active tasks, the three levels, and seeds 1–4, recording outcomes and terminal signals. Continuous integration rejects any recorded win by this controller. The released benchmark contains 9,089 automated tests, 7,830 of them in dedicated test files for 186 of the 217 tasks. These task-specific tests exercise intended strategies and selected shortcuts, including direct attacks, incorrect routes, and skipped sub-goals, with seed coverage specified by each test. They also check structural properties such as reachable deadlines and valid starting configurations. The requirement that unfinished objectives terminate as losses is an authoring criterion checked by the relevant task tests; the suite-wide gate specifically checks for unintended wins by the idle controller. This finite policy coverage does not establish the absence of all possible shortcuts or isolate the intended capability from other abilities needed to solve a task.

Winning-trajectory audit. Because scripted shortcut tests cannot anticipate every strategy, we also audited the recorded wins of the standard sweep. For families where a necessary behavior can be checked from the run records, such as sighting an enemy in a scouting win or fighting in a combat win, 4,162 of 4,335 wins (96.0%) show it; for the remaining 38% of wins no such check is available from summary signals. The audit flagged five tasks whose goal can be met at or near the start (`scout-map-reveal-percent-target`, `build-power-down-defensive`, `expansion-aggro-3-base-greedy`, `mcv-deploy-third-base`, `proc-only-defend-no-attack`), accounting for 2.0% of wins. They remain in the reported suite; excluding them changes each model's success by at most 1.3 points and leaves the model ranking unchanged. `revisions/iclr2027/reviewer_analyses/win_audit.py` and `revisions/iclr2027/compute_shortcut_sensitivity.py` reproduce these results.

I Scoring and Trajectory Alignment

The definitions below follow the inspected benchmark implementation at revision `e317278c`. Historical exports do not retain every command or warning count, so these implementation definitions do not establish that every proxy can be reconstructed for every archived run. The main findings use task outcomes, objective progress, and directly inspected commands rather than the composite or the perception/reasoning/action proxy scores.

Score inputs and normalisation. Write $[x]_0^1 = \min(1, \max(0, x))$. For a run of $T > 0$ decision turns, let W be the terminal win indicator, k_T the terminal tick, and M the task maximum tick budget.

Define E as the explored map fraction, D as the number of distinct enemy units and buildings sighted divided by five and clipped to $[0, 1]$, and $\eta = [1 - k_T/M]_0^1$ as the unused budget fraction. Search focus is $F = [D/E]_0^1$ for $E > 0$ and zero otherwise. With N issued commands and n_{warn} engine warnings, the validity proxy is $V = [1 - n_{\text{warn}}/N]_0^1$, or 1 if $N = 0$.

Composite score. For normalised dimension values d_j, task weights w_j, and scoring deadline B, the composite is

$$C_{\text{base}} = \left[\frac{\sum_j w_j d_j}{\sum_j w_j}\right]_0^1, \qquad C = \left[C_{\text{base}} + 0.05\, W \left[1 - \frac{k_T}{B}\right]_0^1\right]_0^1. \tag{5}$$

Composite dimensions and weights. Table 14 specifies the dimension vector for Equation 5, where q is the recorded enemy-kill count and ℓ is the adapter's unit-loss signal. In the inspected implementation, $\ell = \max(0, n_{\text{initial}} - n_{\text{current}})$ for owned-unit counts; it can fall when units are produced and is not cumulative casualties. Defaults are overridden by the task's reward configuration, and the weights used are saved with each score; normalisation sums weights over the listed dimensions (with denominator 1 if that sum is zero). Economy, density, and disruption are reserved dimensions fixed to zero in this scorer. In particular, the inherited default economy weight remains in the denominator even though its dimension is zero; economy objectives can still contribute through G_T. The speed-bonus budget B is the smallest `within_ticks` bound found by the scorer in the win condition, or the task maximum tick budget when none is found; budgets are bounded below by one. The speed bonus is zero for non-wins and for wins without a positive recorded tick.

Table 14: Run-score dimensions and default weights before task overrides. Brackets denote clipping to $[0, 1]$. Non-winning outcomes, including recorded draws, have $W = 0$.

Dimension	**Value** d_j	**Default** w_j
Outcome	W	0.50
Objective	G_T	0.20
Exploration	E	0.00
Discovery	D	0.00
Combat	$[(q - \ell)/5]_0^1$	0.15
Survival	$[1 - \ell/5]_0^1$	0.10
Tempo	η if $W = 1$, otherwise $\eta/2$	0.10
Format/action validity	V	0.05
Economy (reserved)	0	0.10
Density (reserved)	0	0.00
Disruption (reserved)	0	0.00

Perception, reasoning, and action diagnostics. The scores are computed as

$$\begin{aligned} P &= [0.7E + 0.3D]_0^1, \\ R &= \begin{cases} 0.6 + 0.4\eta, & W = 1, \\ [0.2 + 0.45G_T + 0.25F + 0.1E]_0^1, & W = 0, \end{cases} \\ A &= [0.7V + 0.3\min(N/T, 1)]_0^1. \end{aligned} \tag{6}$$

These are behavioral proxies rather than independent measurements of internal abilities: R explicitly depends on outcome, and A counts `Observe` commands. The historical "weakest-link" label is the minimum of these three scores. Their scales and floors differ, so this label is not a calibrated failure attribution. Comparisons with a task's predefined capability label require independent trajectory evidence; outcome correlation cannot independently validate the reasoning proxy.

Objective-progress summary. For the exported objective leaves $\mathcal{L}$, the tracker reports

$$G_t = \begin{cases} 1, & W_t = 1, \\ \min_{j \in \mathcal{L}} r_{j,t}, & W_t = 0,\ \mathcal{L} \neq \varnothing, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

where W_t indicates satisfaction of the full win condition and $r_{j,t} \in [0, 1]$ is leaf j's recorded progress. This is exported telemetry, not a general measure of distance to task completion.

Goal-tracker coverage. Equation 7 follows the exported leaf telemetry. For a tracked numeric lower-bound objective with positive target b_j, $r_{j,t} = [x_{j,t}/b_j]_0^1$; an upper-bound constraint receives 1 while satisfied and $[b_j/x_{j,t}]_0^1$ when exceeded. Other predicates receive their Boolean satisfaction value. The tracker descends through conjunction, disjunction, and negation, but its flattened minimum does not preserve disjunctive or negated semantics; it also does not expand temporal `then` clauses. Full win-condition evaluation is consequently authoritative. For tracked lower bounds, a satisfied predicate receives ratio 1, including zero targets; an unsatisfied zero target receives 0. Leaf ratios and the resulting scalar are rounded to four decimal places. A violated constraint may retain a positive ratio, and progress can decrease, so the scalar must be read alongside leaf satisfaction and the full condition.

Waiting and command counts. A waiting turn in the trajectory analyses contains only `Observe`; earlier orders can continue executing during that turn. `Stop` cancels orders and is an active command. Some historical fields named `passivity` also count `Stop` or empty command lists; we do not use those fields as waiting measurements. Counts of new or repeated commands do not by themselves establish whether an action is useful. In the takeover study, the initial observation-only turns are scripted, and runs ending before model control provide no model-action trajectory.

Aligning observations and effects. In independent-task playback, the turn-t unit and building snapshot and minimap are taken before command execution, whereas the same row's signals and goal measurements are taken afterwards. The next row's board, or the separately recorded terminal board, provides the corresponding post-action state. Where retained, the conversation transcript supplies the input briefing, model output, and submitted tool calls for that decision turn. Historical exports may strip image payloads, and a generic saved minimap can differ from the image-primary input. Exact input recovery requires the corresponding request or image artifact; a reconstructed state alone does not establish what the model saw. The audit format additionally preserves model requests, responses, and engine warnings when enabled. Transcript tool acknowledgements are not engine confirmations. In 1v1 playback, each player's turn row instead stores the post-action board; the preceding state provides temporal context, while exact decision input requires the recorded request and its image artifacts. Both players' records share the match turn and tick. The 1v1 records retain military and economic signals but do not populate task-goal leaves or the independent-task P/R/A scorecard. Thus objective-progress diagnosis applies to the independent tasks, while competitive trajectories are inspected through the players' commands and state changes.

J Study Records and Coverage

Model participation. Task evaluation includes Claude Opus 4.7, Claude Sonnet 4.6, Gemma 4 31B IT, GPT-5.4, GPT-5.4-mini, Qwen 3.6-35B A3B, Qwen 3.5-9B, and Kimi K2.5. The observation comparison uses Gemma, GPT-5.4-mini, and both Qwen models. The competitive comparison includes all eight models, with historical endpoint differences documented in the arena appendix. The takeover comparison uses Gemma and both Qwen models. Selection for repeated-attempt analysis is described below; these analyses use subsets of the task-evaluation models rather than a single fully crossed experiment.

Historical task sweep. The archived model exports identify the endpoints in Table 15. They span May 2026 evaluations and subsequent resumptions; raw row counts are not unique task-cell counts. The standard sweep targets easy, medium, and hard levels at seed 1, with image-plus-text observations under limited visibility. The exports also contain explicitly text-only `adversarial-duel:easy` records and missing observation metadata; the default setting is therefore not independently established for every retained record. Repetition coverage varies: the Qwen 3.5 export contains only repeat 0, while later production journals also contain partial repeats 1 and 2. Those sources cannot be combined without reconciling handoff encodings, duplicates, task configurations, and versions. Consequently, neither a universal three-repeat design nor a pooled run total is inferred from these exports.

Task coverage and attempt counts. Each model export covers 217 task identifiers and 648 distinct task–difficulty entries: 216 easy, 216 medium, 215 hard, and one `main` entry for `adversarial-1v1-macro`. The arena is recorded under this single configuration instead of three difficulty labels; `adversarial-duel:hard` is absent. Thus the observed coverage is not 217×3. GPT-5.4 records one attempt for each of the 648 entries plus a second attempt on `action-multiunit-coordination:easy`, giving 649 scored attempts. Opus and Sonnet each have three attempts per entry, giving $648 \times 3 = 1{,}944$. Other totals reflect additional records, missing scored attempts, or excluded operational errors; the coverage audit records these separately.

Repeated-attempt selection. We group the selected historical exports by task, difficulty, seed, and split, requiring exactly one scored record at each repeat index 0, 1, and 2. An entire group is excluded for a missing attempt, an operational error, duplicate repeat records, or conflicting explicit observation or handoff settings. Retained counts are 648 groups each for Opus and Sonnet, 646 for Gemma, 644 for Qwen 3.6, and 643 for Kimi, all at seed 1. Missing metadata does not establish identical settings: 357 Gemma groups and all 644 Qwen 3.6 groups lack observation metadata for at least one attempt. We therefore do not attribute all outcome variation to stochastic decoding. The three attempts are matched within each model; the retained task sets can differ across models.

Aggregation and uncertainty. Task success is wins divided by scored attempts, including losses and draws in the denominator; operational errors are accounted for separately. Overall task estimates retain the historical exports' attempt weights. Six repeated evaluation-key rows for Qwen 3.6 remain in the overall sample but are excluded by the stricter repeated-attempt selection. Shared-task model comparisons first average attempts within each task–difficulty combination, then weight combinations equally. Bootstrap sampling is at the task level, with all associated difficulties and attempts retained together; the random seeds are 20260920 for overall estimates and 20260921 for repeated-attempt estimates. Intervals are the 2.5th and 97.5th percentiles of 10,000 resamples, conditional on the recorded attempts. Paired differences use the same sampled tasks for both estimates. Pairwise intervals are pointwise, without multiple-comparison correction; an interval containing zero does not establish equivalence.

Table 15: Endpoints recorded in the historical task exports. An endpoint is the API service and model identifier through which a model was queried. Provider names identify services, not immutable weight hashes. The observation and takeover studies use Together for Gemma, with endpoint `google/gemma-4-31B-it`; its earlier task sweep uses OpenRouter.

Provider	Recorded model endpoint
Bedrock	`global.anthropic.claude-opus-4-7`
Bedrock	`global.anthropic.claude-sonnet-4-6`
OpenRouter	`google/gemma-4-31b-it`
OpenAI	`gpt-5.4-2026-03-05`
OpenAI	`gpt-5.4-mini-2026-03-17`
Together	`Qwen/Qwen3.6-35B-A3B-FP8` (dedicated endpoint)
Together	`Qwen/Qwen3.5-9B`
Bedrock	`moonshotai.kimi-k2.5`

Human performance baseline. Testers issue commands through the Play UI, which presents the same game information and observation format as the agent interface. The archive contains 246 scored runs across 149 tasks and 184 task–difficulty combinations: 39 easy, 37 medium, and 170 hard runs, with the hard subset covering 114 tasks. It combines 240 runs from the updated benchmark archive with six historical runs absent from that archive; 71 overlapping runs agree on recorded outcomes and metrics. Sixteen legacy objective-progress values are reconstructed from terminal goal leaves to match the blocking-ratio definition. The shared-task comparison retains 245 human runs over 183 combinations from 148 tasks, excluding `adversarial-duel:hard`, which lacks corresponding model records. We average each cohort's attempts within a combination and weight it by its human attempt count, using the same sampled tasks and weights for all cohorts. Matching task identifiers and difficulty does not by itself establish identical historical task versions or seeds. Intervals are conditional on the observed testers and do not estimate variation across the wider human population.

Observation records. The four complete July 11 journals record benchmark revision `b597adbe`. Their conditions are `structured` (text-only), `vision` (image-plus-text), and `image` (image-primary), with a `-clear` suffix for full visibility. Each contains 217 unique medium-task records per condition at seed 1, repeat 0: 5,208 scored records across four models and six conditions. The model endpoints match Table 15, except for Gemma's provider and capitalization noted in the caption. These counts describe scored task outcomes, with a win counted as success and a loss or draw as a non-win. The completed Gemma journal also retains 369 unscored error records without a condition label; they cannot be assigned to condition-specific denominators. Earlier incomplete Gemma evaluation batches are not pooled with this completed study. Thus the scored comparison does not by itself measure first-request reliability or equal API effort.

Takeover records. The three completed July 11 journals also record benchmark revision `b597adbe`. Each contains one scored record per medium task for play from turn 1 and for a ten-turn `Observe`-only prefix, at seed 1 and repeat 0, using image-plus-text input and limited visibility. The prefix consumes the original decision budget; the model receives the resulting state without the prefix transcript. The Qwen 3.5 study uses the completed second evaluation batch rather than the earlier baseline-only batch; Gemma uses Together. Sixteen runs per model terminate before any model-controlled turn in the prefix condition. They remain in the task-outcome denominator but provide no model-action trajectory. The condition identifier `handoff-bad` names this fixed stall; it is not a sampled collection of poor strategies. No successful-prefix comparison is established by these records. Appendix P reports the paired estimates and uncertainty.

Competitive records and cohort identity. The recovered archive contains 388 completed game records. Retaining the latest completion per model pair, configuration label, position assignment, and seed gives 359 games. The primary analysis excludes three settings with unmatched historical playback exports, leaving 356 games: 119 complete pairs between different models, 45 complete same-model pairs, and 28 games without a retained counterpart. Only the 119 cross-model pairs enter the main opponent matrix. Two wins give a paired win, two losses a paired loss, and one win apiece a paired draw; no selected individual game ends in a draw. Model identity is mapped to the recorded player seat separately in each half. Position estimates use 194 games with directly matched playback evidence; the other positions are reconstructed only in sensitivity analyses. Endpoint differences, settings, provenance, opponent/seed coverage, and uncertainty appear in Appendix E.

Reproduction limits. Journal benchmark hashes identify the harness code, not necessarily the engine wheel or a provider's immutable weights. Deterministic replay requires the archived engine build, task configuration, seed, and timed command sequence. Where those artifacts or exact input images are absent, the record supports only the quantities actually retained; a current checkout is not proof of the configuration used in a historical evaluation.

K Additional Task-Performance Results

The tables below use the records and task-bootstrap procedure described in Appendix J. Table 16 reports all scored attempts; Table 17 uses the 183 task–difficulty combinations shared with the human baseline. Goal progress measures the terminal blocking ratio of the task's objective; exploration measures the fraction of map cells explored. Goal progress is set to one on success, so its association with success is partly built into the metric. Neither progress nor map coverage alone identifies a reasoning or perception mechanism.

Table 16: All scored historical attempts: mean success, terminal goal progress, and explored map fraction, in percent, with 95% task-bootstrap intervals. The full human archive covers a different task sample; compare humans and models using the shared-task estimates in Table 17.

Model / cohort	Attempts	Success	Goal progress	Exploration
Opus 4.7	1,944	82.1 [78.4, 85.5]	86.7 [83.6, 89.5]	29.6 [26.7, 32.7]
Sonnet 4.6	1,944	72.7 [68.4, 76.9]	79.3 [75.4, 83.0]	32.7 [29.7, 35.7]
Gemma 4 31B	1,946	65.2 [60.6, 69.5]	73.3 [68.9, 77.3]	27.0 [24.3, 29.7]
GPT-5.4	649	63.8 [58.9, 68.4]	71.2 [66.6, 75.5]	33.7 [30.8, 36.7]
GPT-5.4-mini	648	51.7 [46.4, 56.8]	60.8 [55.6, 65.8]	28.8 [26.2, 31.5]
Qwen 3.6-35B	1,953	46.3 [41.4, 51.3]	57.6 [52.7, 62.5]	27.3 [24.7, 30.0]
Qwen 3.5-9B	648	44.9 [39.9, 50.0]	55.6 [50.5, 60.6]	29.8 [27.2, 32.6]
Kimi K2.5	1,939	37.5 [32.4, 42.6]	48.2 [42.9, 53.5]	25.1 [22.4, 27.9]
Human (selected tasks)	246	52.8 [46.1, 59.6]	64.1 [57.8, 70.6]	33.5 [29.1, 38.2]

Paired comparisons. Tables 18 and 19 use the same sampled tasks for both cohorts in each bootstrap replicate. Model-only comparisons weight the 648 shared task–difficulty combinations equally after averaging attempts within each combination. Human comparisons weight the 183 shared combinations by their human attempt counts, as in Figure 3b. The Gemma–GPT-5.4 and Qwen 3.6–Qwen 3.5 intervals both contain zero; this does not establish equivalence. These pointwise intervals do not certify a full ranking.

Source and sensitivity checks. The released analysis records the source Git commits, content hashes, excluded errors, and sample coverage. It also reports sensitivity to keeping the first or last duplicate evaluation record and to excluding explicitly text-only historical records. These checks do not determine which rerun was intended for inclusion. The main estimates retain the historical exports' scored records; tool-call validity is omitted because the corresponding command and warning counts have not been recovered for the full cohort.

Table 17: Shared-task comparison on 183 task–difficulty combinations (148 tasks), using 245 human attempts. Means and 95% CIs are percentages. We average attempts within each combination, then weight it by its human attempt count. Bootstrap samples use the same tasks and weights for every cohort. Historical task versions and seeds are not fully matched.

Model / cohort	Success	Goal progress	Exploration
Opus 4.7	79.9 [73.3, 85.6]	84.2 [77.9, 89.6]	30.8 [26.3, 35.5]
Sonnet 4.6	70.1 [62.7, 77.2]	77.6 [70.7, 83.9]	34.2 [29.8, 38.7]
Gemma 4 31B	62.7 [55.5, 69.9]	70.9 [63.9, 77.5]	27.5 [23.6, 31.5]
GPT-5.4	59.2 [50.4, 67.7]	67.9 [59.9, 75.5]	35.9 [31.5, 40.5]
Human	53.1 [46.4, 59.9]	64.1 [57.7, 70.6]	33.4 [29.0, 38.0]
GPT-5.4-mini	46.5 [38.3, 55.0]	53.8 [45.5, 62.3]	30.6 [26.5, 34.9]
Qwen 3.5-9B	43.7 [35.1, 52.4]	53.8 [45.3, 62.1]	32.2 [27.7, 36.9]
Qwen 3.6-35B	42.2 [35.2, 49.4]	55.0 [47.8, 62.3]	28.8 [24.8, 33.0]
Kimi K2.5	32.4 [25.8, 39.3]	43.6 [36.3, 51.0]	26.7 [22.6, 31.1]

Table 18: Paired bootstrap intervals, in percentage points (pp), for two close model success-rate estimates on the 648 shared task–difficulty pairs, weighted equally after averaging repeats within each pair. Differences use the same sampled tasks for both models. These are pointwise 95% intervals, without multiplicity adjustment; an interval containing zero does not establish equivalence.

Success-rate difference	Estimate (pp)	95% CI (pp)
Gemma 4 31B minus GPT-5.4	1.4	[-2.9, 5.8]
Qwen 3.6-35B minus Qwen 3.5-9B	1.2	[-3.1, 5.3]

L Why Human Success Falls Mid-Range

Human testers reach 53.1% success on the 183 task–difficulty combinations shared with all eight models, fifth among nine cohorts (Section 4.1). This position does not reflect uniformly average play. It reflects strengths and weaknesses that roughly cancel. All numbers below use the same frozen records and human-attempt weighting as Figure 3b.

Table 19: Human-minus-model differences on the shared tasks, in percentage points, with pointwise 95% paired task-bootstrap intervals. Weighting matches Table 17. No multiplicity adjustment is applied; these exploratory intervals do not certify an ordering of cohort abilities.

Human minus model	**Success (pp)**	**Progress (pp)**	**Exploration (pp)**
Opus 4.7	-26.8 [-35.2, -18.0]	-20.1 [-27.8, -12.4]	2.6 [0.1, 5.1]
Sonnet 4.6	-17.0 [-25.5, -8.1]	-13.5 [-20.7, -6.1]	-0.8 [-3.5, 1.8]
Gemma 4 31B	-9.7 [-17.8, -1.3]	-6.8 [-13.7, 0.1]	5.9 [3.3, 8.6]
GPT-5.4	-6.1 [-16.9, 5.0]	-3.9 [-13.4, 5.5]	-2.5 [-5.6, 0.8]
GPT-5.4-mini	6.5 [-3.2, 16.1]	10.2 [1.9, 18.4]	2.8 [-0.3, 6.0]
Qwen 3.6-35B	10.9 [1.9, 19.9]	9.1 [1.2, 16.9]	4.6 [2.2, 7.1]
Qwen 3.5-9B	9.4 [-0.4, 19.2]	10.3 [1.4, 19.2]	1.2 [-1.9, 4.5]
Kimi K2.5	20.7 [12.0, 29.4]	20.5 [13.1, 27.8]	6.6 [4.1, 9.1]

Table 20: Human and model task success (%) by family on the 183 task–difficulty combinations shared with all eight models, weighting each combination by its number of human attempts as in Figure 3b. Rank places the human cohort among nine cohorts (1 is highest).

Family	Combinations	Human	Model range	Model median	Human rank
Indirect Strategy	7	82	9–55	17	1
Special Units	5	86	29–86	67	1
Multi-group Coordination	9	85	44–92	56	3
Procedure Robustness	5	33	13–67	31	3
Scouting & Vision	11	59	35–92	56	4
Tempo & Adversarial	5	33	11–44	26	4
Combat Micro	25	51	21–81	53	5
Long-Horizon Play	23	62	33–94	62	5
Build & Tech	18	48	21–78	60	6
Resource Economy	45	44	21–82	56	6
Multi-front Resilience	8	45	39–91	55	7
Static Defense	22	46	31–88	67	8

Performance is uneven across families. Humans rank first among the nine cohorts on Indirect Strategy (82% against a model median of 17%) and Special Units (86% against 67%), and third on Multi-group Coordination. They rank sixth to eighth on Build & Tech, Resource Economy, Multi-front Resilience, and Static Defense, which together account for 93 of the 183 combinations and 106 of the 245 matched human runs (Table 20). Across individual combinations, human success exceeds the model median on 57 and falls below it on 69. Family estimates rest on 5 to 45 combinations and 7 to 48 human runs each, so they are descriptive and are not adjusted for multiple comparisons.

Tasks won or lost relative to the strongest models. Humans succeed on at least half their attempts while Opus, Sonnet, and Gemma each succeed on fewer than half in 8 combinations, including two Indirect Strategy tasks and a race to build a second base. The reverse holds in 38 combinations, 22 of them in Resource Economy, Static Defense, and Build & Tech. The human runs behind these unique wins use a median of 5 active turns, against 11 for the runs behind the unique losses.

Winning human runs issue few orders and then wait. A waiting turn contains only `Observe`. Across all 246 human runs, the median waiting fraction is 56%: 55% in wins and 60% in losses. Winning runs use a median of 7 active turns and 10 commands, against 14 active turns and 22

commands in losing runs, and 31 of the 130 wins use at most three active turns. Human commands are dominated by direct unit control: `AttackUnit` accounts for 49% of non-observation commands and `MoveUnits` for 33%, against 11% for production requests.

Interpretation. Humans do best on tasks that reward one well-chosen plan, executed with few orders and then left to run. They fall behind on tasks that require sustained management over many turns, such as income, production, and defensive coverage, where the stronger models succeed more often. These differences cancel in the aggregate, which places the human cohort between GPT-5.4 and GPT-5.4-mini. Testers played through a different interface (the Play UI) and form a small, selected sample (Appendix S); the analysis describes this cohort, not human capability in general. `revisions/iclr2027/compute_human_analysis.py` reproduces every number in this appendix.

M Unit Orders and Their Effects

Table 21: Unit orders on the task suite. Fielded counts new units that appear per unit order; unaffordable orders are placed while funds are below the unit's cost; repeated orders request the same unit as the previous turn. The last column compares, within the same task, runs where most orders produce a unit with runs where most do not: win-rate difference in percentage points (pp), 95% task-bootstrap interval, and number of tasks with both kinds of run (shown when at least 15).

Agent	Runs	Unit orders per run	Fielded per order (%)	Unaffordable (%)	Repeated (%)	Within-task win-rate difference (pp)
Opus 4.7	1,974	1.8	84	5	25	+27 [+11, +44] (23)
Sonnet 4.6	1,981	2.2	80	5	47	+23 [+10, +37] (38)
Gemma 4 31B	1,852	3.8	36	3	63	+16 [+3, +28] (71)
GPT-5.4	836	4.4	36	4	59	(5 tasks)
GPT-5.4-mini	859	5.9	21	16	54	(6 tasks)
Qwen 3.6-35B	1,824	4.8	36	8	47	+15 [+4, +26] (67)
Qwen 3.5-9B	1,599	39.5	4	46	84	+10 [-4, +24] (36)
Kimi K2.5	1,978	0.7	88	3	6	(8 tasks)
Human	246	1.9	77	6	10	(2 tasks)

Records and definitions. The analysis uses every archived task run with per-turn records: the May production archive for Gemma, GPT-5.4, GPT-5.4-mini, and both Qwen models, the Bedrock archive for Opus, Sonnet, and Kimi, and the 246 human runs of Section 4.1. These archives include repeated and resumed attempts beyond the frozen scored cohort, so run counts differ from those behind Figure 3. A unit order is a `Build` command for a mobile unit; structures are excluded because placing them is a separate step. Fielded units are distinct new unit identifiers that appear in the side's unit list. An order is unaffordable when the unit's cost in the pinned Red Alert rules exceeds the side's funds after the previous turn, and repeated when the same unit was also ordered on the previous turn. Repeated orders can legitimately queue several units, so the fielded rate measures how much of the requested production materializes, not the error rate of each command.

Command validity and missing replies. Run manifests record how many commands the engine flagged with a warning and how many model replies contained no tool call. Flagged commands are 1.1% for Opus, 0.05% for Sonnet, 0.7% for Gemma, 1.9% for GPT-5.4, 2.0% for GPT-5.4-mini, 1.2% for Qwen 3.6, 4.5% for Qwen 3.5-9B, and 0.05% for Kimi. Turns without a tool call are below 6% for every model except Qwen 3.6 (15%) and Kimi (81%); for Kimi, few unit orders and few repeats therefore reflect missing replies rather than deliberate restraint.

Within-task comparison. For each agent and task with at least one run whose fielded-to-ordered ratio is at least 0.5 and one run below 0.5, we average the win rate within each group and take the difference; the reported value averages these differences across tasks. Intervals use 10,000 bootstrap resamples of tasks (seed 20260925). Runs with no unit orders are excluded. Differences in task composition across models mean that the within-task values are not comparable across models. `revisions/iclr2027/compute_task_conversion.py` reproduces the table.

N A NON-LLM AGENT: JEV

Table 22: Success (%) on the 33 task–difficulty combinations of the Jev evaluation (one task per family, three difficulties). The seed-1 column is comparable across agents; the last column adds Jev's seeds 2 and 3. Jev ran with text-only input, whereas the LLM records use image-plus-text input, which Section 4.2 shows lowers success.

Agent	Seed 1	Seeds 1–3
GPT-5.4	87.9	–
Opus 4.7	86.9	–
Sonnet 4.6	82.8	–
Gemma 4 31B	73.7	–
GPT-5.4-mini	63.6	–
Jev (per-unit)	60.6	64.6
Qwen 3.6-35B	57.6	–
Qwen 3.5-9B	54.5	–
Jev (two-stage)	54.5	55.6
Kimi K2.5	51.5	–
Stall baseline	0.0	0.0
Scripted baseline	0.0	0.0

Controller. Jev is a decision model from TypeSafe that answers typed multiple-choice questions and does not generate text or tool calls. Its controller enumerates every rule-legal action from the observation, respecting the task's permitted commands, visibility, faction, prerequisites, and engine rules, and asks Jev one choice per unit, per building, for production, and for each completed placement. A two-stage variant first asks for a shared intent and then the per-unit choices. Coordinates are discretized uniformly when an option set exceeds 255 choices. Objective locations are offered as options only on levels whose briefing states exact coordinates, matching the information given to the LLMs.

Evaluation. One task from each of 11 families was played at easy, medium, and hard difficulty with seeds 1–3, using text-only input with limited visibility, one attempt per combination: 99 runs per arm, all scored. A stall controller and the default scripted controller win none of the 99. Per-unit Jev wins 64 and two-stage Jev 55. Both arms lose all nine economy runs and all nine defend-then-counterattack runs; failures there include runs with no production chosen and runs issuing only stance changes and builds without movement or attack. Table 22 compares seed-1 success with the LLM records on the same 33 combinations; the LLMs used image-plus-text input, which lowers success relative to text-only input (Section 4.2), so the comparison favors Jev. A like-for-like check uses the text-only, limited-visibility runs of the observation study (Section 4.2) on Jev's 11 medium tasks at seed 1: both Jev arms win 7, Gemma and GPT-5.4-mini win 8, Qwen 3.5-9B wins 7, and Qwen 3.6 wins 6.

Provenance. Scorecards, manifests, and the evaluation report are copied into `revisions/iclr2027/jev_sweep/` with source hashes; `revisions/iclr2027/compute_jev_comparison.py` verifies them and regenerates Table 22. Per-turn transcripts of the Jev runs were not archived with these results.

O Observation Comparisons and Uncertainty

The study uses one scored attempt per model and condition on each of 217 medium-difficulty tasks (Appendix J). Table 23 reports all six conditions; Tables 24 and 25 compare formats and visibility settings.

Paired task bootstrap. Let Y_{jm}^{c} equal 1 for a win and 0 for a loss or draw on task j, model m, and condition c. We estimate $100\,n^{-1}\sum_{j=1}^{n}(Y_{jm}^{a} - Y_{jm}^{b})$ percentage points for conditions a and b, with $n = 217$. We resample n task identifiers with replacement 10,000 times (random seed 20260921), retaining all conditions and models for each sampled task. Pointwise 95% intervals use the 2.5th and 97.5th percentiles of the paired differences; rate intervals use the same draws. There is no multiplicity adjustment. Intervals describe task variation conditional on recorded attempts, not variation across fresh model samples or environment seeds. An interval containing zero does not establish equivalence.

Table 23: Task success (%) under all six observation conditions. Brackets give 95% task-bootstrap confidence intervals; each entry uses the same 217 medium tasks.

Model	Visibility	Image-primary	Image-plus-text	Text-only
Gemma 4 31B	Limited	52.1 [45.6, 58.5]	63.6 [57.1, 69.6]	62.7 [56.2, 69.1]
Gemma 4 31B	Full	55.3 [48.4, 61.8]	67.7 [61.3, 73.7]	65.9 [59.4, 71.9]
GPT-5.4-mini	Limited	41.0 [34.6, 47.5]	48.8 [42.4, 55.3]	50.2 [43.8, 56.7]
GPT-5.4-mini	Full	38.7 [32.3, 45.2]	48.8 [42.4, 55.3]	47.9 [41.5, 54.4]
Qwen 3.6-35B	Limited	37.3 [30.9, 43.8]	41.9 [35.5, 48.4]	45.2 [38.2, 51.6]
Qwen 3.6-35B	Full	41.9 [35.5, 48.4]	44.2 [37.8, 50.7]	45.2 [38.7, 51.6]
Qwen 3.5-9B	Limited	34.6 [28.6, 41.0]	38.7 [32.3, 45.2]	41.9 [35.5, 48.4]
Qwen 3.5-9B	Full	35.0 [28.6, 41.5]	42.4 [35.9, 48.8]	46.1 [39.6, 52.5]

Table 24: Paired differences between observation formats (percentage points), with pointwise 95% confidence intervals. Positive values favor the first format.

Model	Visibility	Text-only minus image-primary	Image-plus-text minus text-only
Gemma 4 31B	Limited	10.6 [3.2, 18.0]	0.9 [-5.1, 6.9]
Gemma 4 31B	Full	10.6 [3.7, 17.5]	1.8 [-3.7, 7.4]
GPT-5.4-mini	Limited	9.2 [3.2, 15.2]	-1.4 [-7.8, 5.1]
GPT-5.4-mini	Full	9.2 [2.3, 16.1]	0.9 [-5.5, 7.4]
Qwen 3.6-35B	Limited	7.8 [1.8, 13.8]	-3.2 [-8.8, 2.8]
Qwen 3.6-35B	Full	3.2 [-2.8, 9.7]	-0.9 [-5.5, 3.2]
Qwen 3.5-9B	Limited	7.4 [0.5, 14.3]	-3.2 [-9.2, 3.2]
Qwen 3.5-9B	Full	11.1 [4.6, 18.0]	-3.7 [-10.1, 2.8]

Table 25: Paired effects of full-map visibility: full minus limited task success (percentage points), with pointwise 95% confidence intervals.

Model	Image-primary	Image-plus-text	Text-only
Gemma 4 31B	3.2 [-2.8, 9.2]	4.1 [-2.3, 10.6]	3.2 [-2.8, 9.2]
GPT-5.4-mini	-2.3 [-8.3, 3.7]	0.0 [-6.9, 6.9]	-2.3 [-8.8, 3.7]
Qwen 3.6-35B	4.6 [-1.4, 10.6]	2.3 [-3.7, 8.3]	0.0 [-6.0, 5.5]
Qwen 3.5-9B	0.5 [-5.5, 6.5]	3.7 [-2.8, 10.1]	4.1 [-2.3, 11.1]

Coverage and interpretation. All 24 model–condition combinations contain 217 scored tasks. Gemma's 369 unscored operational errors lack condition labels and are excluded; rates do not measure equal API effort or first-request reliability. Appendix G details interface differences that limit mechanistic interpretation.

P PRIOR STATE, AGENT OUTLOOK, AND STALLED-GAME TAKEOVER

P.1 CONDITIONAL TRAJECTORY ANALYSIS: PRELIMINARY STATE SNAPSHOT

Archive and checkpoints. The expanded retrospective archive contains 1,790 medium-difficulty, seed-1 runs across 217 tasks: 632 Gemma 4, 619 Qwen 3.6, and 539 Qwen 3.5 runs, including repeated and historical attempts. We select the initial decision and the decisions immediately after 25%, 50%, and 75% of each run's allowed decision budget. These landmarks retain 1,790, 1,308, 868, and 567 checkpoints, respectively; a terminated run contributes no later checkpoint. Later-stage estimates therefore condition on the run remaining active. The 4,533 checkpoints yield 2,809 distinct state packets and 3,600 distinct substantive-response packets after exact deduplication. Another 933 checkpoints have missing or formatting-only response text.

Temporal alignment and labels. Each checkpoint's briefing matches its pre-command tick. Current-row unit records and, where available, building records describe the pre-action situation; scalar signals are taken from the preceding decision's post-action record. Initial checkpoints omit post-action signals. State packets exclude the current response, and outlook packets contain the response and objective without linked state labels. Both omit future states and terminal outcomes. Favorable labels require recorded support for a concrete route toward the remaining goal; unfavorable labels identify an obstructing deficit or constraint. Missing production status, sequence progress, terrain, or other decisive information can yield an uncertain label. These assessments do not establish winnability or impossibility.

Optimistic responses explicitly anticipate progress or success; pessimistic responses state expected failure or inability to proceed. Asserted opposing expectations are mixed, and plans or current-state descriptions without a directional forecast are unstated. Missing response text is a separate category. Forecast scope distinguishes an intermediate action effect from task success. Codex agents annotate and quote evidence under a shared rubric, with per-batch model provenance. Failed Luna annotation attempts were excluded. Calibration, independent AI spot checks, and corrections are recorded, but the semantic audit remains unfinished and no independent human validation is claimed.

Frozen interim coverage. The state-only snapshot used in Table 26 freezes 1,750 completed state annotations before joining outcomes. It includes 2,843 checkpoints from 1,531 runs and 211 tasks: 1,139 initial, 805 quarter-budget, 527 half-budget, and 372 three-quarter-budget checkpoints. Model coverage is uneven: 921 Gemma, 913 Qwen 3.5, and 1,009 Qwen 3.6 checkpoints. Completion of annotation determines this subset; it is not a representative estimate for all 1,790 runs. The separate outlook annotations are retained for descriptive analysis, but their four pessimistic responses are too sparse for a balanced comparison.

Table 26: **Preliminary state-conditioned task success.** Entries show recorded wins / annotated checkpoints; brackets give inverse-attempt-weighted success with 95% whole-task bootstrap intervals. The state annotation is outcome-masked and assesses the pre-action situation. The snapshot covers 2,843 checkpoints from 1,531 runs and 211 tasks; repeated checkpoints can share a terminal outcome. Annotation and semantic review remain incomplete.

Prior situation	Wins / checkpoints	Weighted success (%)	Traces
Favorable	292/359	80.9 [70.2, 90.1]	294
Uncertain	880/2252	45.5 [40.1, 50.9]	1337
Unfavorable	17/232	9.5 [3.5, 17.1]	188

Outcomes and uncertainty. We measure recorded terminal task success. We do not average component progress signals: their meanings and logical relationships differ across tasks. Main-table entries are raw successful-checkpoint counts over observed checkpoints. Repeated checkpoints can share a run's terminal outcome. The exploratory pooled rates weight each checkpoint by the inverse number of original attempts for its task/model pair and the inverse number of available landmarks for its run, before restricting to completed state annotations. This does not correct selection from incomplete annotation.

Table 27: Exploratory pooled state-conditioned success rates. Weights balance original attempts and available decision landmarks before selecting annotated records. Intervals are 95% task-clustered bootstrap intervals and exclude annotation and selection error.

Prior situation	Checkpoints	Traces	Tasks	Weighted success (%)
Favorable	359	294	80	80.9 [70.2, 90.1]
Uncertain	2252	1337	196	45.5 [40.1, 50.9]
Unfavorable	232	188	87	9.5 [3.5, 17.1]

For uncertainty, 10,000 bootstrap resamples draw the 217 tasks with replacement (seed 20260922), retaining each task's models, attempts, and checkpoints. Percentile intervals condition on the frozen labels and available subset; they exclude annotation and selection error. Empty-cell resamples are omitted and counted in the released results. Singleton-task and constant-outcome cells show no interval because their empirical bootstrap is uninformative. In particular, one pessimistic checkpoint followed by success cannot estimate the relative effectiveness of pessimistic decisions. Differences in task, model, and decision-stage composition also prevent a causal interpretation of outlook.

Table 28: Preliminary raw state-conditioned wins / checkpoints by decision landmark. Later landmarks are survival-conditioned and are not randomized interventions.

Prior situation	Initial	Quarter	Half	Three-quarter
Favorable	109/147	102/117	56/66	25/29
Uncertain	443/974	273/621	119/397	45/260
Unfavorable	2/18	5/67	3/64	7/83

Reproduction and completion. Outcome-masked packets, source hashes, the frozen state snapshot, and interim estimates are preserved in `revisions/iclr2027/conditional_analysis_full/`. The scripts `preview_state_rates.py` and `render_preliminary_state_tables.py` reproduce this separately marked preliminary analysis. The final estimator, `compute_full_conditional_rates.py`, still requires complete annotations and a frozen approved audit before joining outcomes. The earlier 41-run combat-only pilot remains archived and is not combined with this snapshot. The July takeover journals below are a separate experiment; their original invocation disabled the detailed trace recorders, and the available files do not support outlook annotation.

P.2 Idle-Handoff Stress Test and Additional Examples

Table 29: Success rates and paired differences with pointwise 95% task-bootstrap intervals. Each condition includes one scored attempt on each of 217 medium-difficulty tasks.

Model	From start (%; 95% CI)	After takeover (%; 95% CI)	Difference (pp; 95% CI)
Gemma 4 31B	59.9 [53.5, 66.4]	43.3 [36.9, 49.8]	-16.6 [-24.0, -9.7]
Qwen 3.6-35B	43.8 [37.3, 50.2]	34.6 [28.6, 41.0]	-9.2 [-16.1, -2.3]
Qwen 3.5-9B	38.7 [32.3, 45.2]	27.6 [21.7, 33.6]	-11.1 [-18.0, -4.1]

Matched outcomes and uncertainty. The three completed journals contain 1,302 scored records: 217 medium-difficulty tasks, two starting conditions, and three models. There is exactly one scored attempt per task and condition, with no duplicate evaluation keys or unscored errors in these journals. The turn-1 attempts are separate from the image-plus-text, limited-visibility attempts in the observation study, despite sharing the task set and nominal observation settings. Wins from turn 1 are 130 for Gemma, 95 for Qwen 3.6, and 84 for Qwen 3.5; wins after the takeover are 94, 75, and 60, respectively. Tasks won only from turn 1 number 51, 39, and 43; tasks won only after the takeover

number 15, 19, and 19. We resample the 217 tasks 10,000 times with replacement (bootstrap seed 20260921), retaining both conditions and all models for every sampled task. The reported intervals are the 2.5th and 97.5th percentiles. Difference intervals use paired task outcomes and are pointwise, without multiple-comparison adjustment. They describe task variation conditional on the recorded attempts, rather than variability across new model runs or environment seeds.

Scope of the intervention. The first ten turns use scripted `Observe` commands. The same 16 tasks fail during that prefix for every model, sometimes before all ten turns have elapsed. These outcomes count as failures in the takeover condition; excluding them would change the task set after applying the intervention. The model receives the resulting state without the prefix transcript, and the original decision budget is unchanged. The comparison measures the total effect of this idle prefix on task completion: less time remains, the world may be harder, and some tasks end before model control. The saved journals do not include takeover states or per-turn traces, and no reference policy verifies feasibility across the delayed task set. These data therefore cannot isolate an agent's recovery ability. Section J gives the recorded endpoints and settings.

Recoverability. A win after takeover proves that the task was feasible for that run, but a loss does not show whether any policy could have completed it from the inherited state. Comparing the time remaining with the duration of a win from the original state cannot resolve this, because the world has changed. We therefore do not attribute the lost wins to a failure to recover.

Takeover example. On `def-in-depth-vs-single:medium`, success requires defending a construction yard against an infantry wave, killing at least seven attackers, and placing pillboxes in two specified areas before the deadline. Gemma, Qwen 3.6, and Qwen 3.5 each win from the start. After ten idle turns, all three defend the yard and defeat the wave but miss the pillbox requirement. These final goal states show what remained unfinished; they do not show why it was missed or whether the inherited state was still winnable. The runs recorded no per-turn trace.

Harvesting example. The historical human run `human-20260525T224748Z`, on `econ-far-patch-vs-near-patch:hard` at seed 1, contains 40 decision records. Turn 1 issues a single harvesting command; turns 2–40 contain only `Observe`. The post-command economic signal is zero on turn 1 and reaches 5,500 on turn 40, when the recorded goal is satisfied. This is a selected example of task progress during observation-only turns, not an estimate of its prevalence or evidence that the waiting policy is optimal. It is separate from the medium-difficulty takeover study and from the combat trajectories in Figure 4.

Stated expectations before waiting. Saved responses for Figure 4 are preserved separately from the six state/command source files. Their 156 user-briefing/assistant-response pairs align with the decision records: each briefing precedes the corresponding command, whose execution advances the recorded tick by 90. In Run B at turn 53, the response states, "I've been observing for many turns without any change in the game state," then requests `observe` while expecting production to finish. In Run C at turn 50, it identifies unfunded production and says, "This seems like a deadlock," but ends by waiting for a unit to complete. Both selected responses contain explicit observation tool calls; subsequent records show no further kills, and Run C produces no units. Run B's observation-only command batch at turn 57 has no recorded model tool call and is not used as an explicit choice to wait. The release preserves source hashes, message indices, prior briefing text, response text, and aligned commands in `revisions/iclr2027/state_assessment_records.json`.

Assessment and recoverability. These are stated expectations in generated responses, not validated internal beliefs. A response may acknowledge a blocked plan while still anticipating progress, so it need not admit a single confident label such as winning or hopeless. We do not assign those labels from the terminal outcome. A successful alternative continuation would demonstrate recoverability under its tested conditions; failure of one continuation does not establish that no feasible recovery exists. The takeover experiment does not elicit or classify agent assessments, and its scripted prefix cannot show that an agent chose to give up.

Reproduction. The analysis script checks SHA-256 hashes for all three journals and the two harvesting source files, verifies task matching and early terminations, and cross-checks paired bootstrap

estimates against an independent count-weighted calculation. The release includes the preserved harvesting records, their provenance, per-task outcomes, and the generated estimates in `revisions/iclr2027/action_timing_results.json`; `revisions/iclr2027/compute_action_timing.py` regenerates both tables. The separate `revisions/iclr2027/audit_state_assessment.py` verifies response-source hashes and regenerates the selected decision records.

Q Relation to Embodied Agents

Red Alert Bench is a game, and we have not tested whether its results transfer to physical systems. Each finding nonetheless has a counterpart studied in embodied AI; we list them with the strength of the link.

Observation format (strong). Vision–language models often gain little from images when a textual description of the same scene is available (Wang et al., 2024). Our text-over-image advantage is consistent with this, with the caveat that our formats also differ in how positions and units are labeled (Section 4.2).

Actions that do not take effect (strong). LLM-driven robots improve when success detection and scene feedback are returned to the planner (Huang et al., 2023), and summaries of sensor history let a model explain and correct execution failures (Liu et al., 2023). Low yields of new units per unit order, and responses that name a blocking constraint yet predict progress (Sections 4.3 and 4.4), are the failures such monitoring targets. The pattern also matches a knowing–doing gap reported in other decision-making settings (Schmied et al., 2025).

Reliability across attempts (strong). Robot policy evaluation has been urged to report repeated trials, uncertainty, and failure modes rather than a single success rate (Kress-Gazit et al., 2024). Our stable averages with 15.6–29.2% of task outcomes changing across attempts illustrate why.

Taking over mid-task (moderate). In driving, less time before a takeover from automation lowers the quality of the takeover (Gold et al., 2013). Our idle-handoff test resembles this, but it changes the remaining budget and the world state together (Appendix P).

Human profile (weak). An operator supervising several robots can leave each unattended only until its performance falls below an acceptable threshold, which limits how many robots one person can manage (Crandall et al., 2005). This is an analogy for the human baseline's weakness on tasks that require sustained management (Appendix L), not a measured equivalence.

Starting-position advantage (weak). Vision–language models default to particular spatial frames of reference (Zhang et al., 2025b). Because the map is symmetric, the east-side advantage reflects how agents play from each position, but we have not identified its cause; it is a reason to test position symmetry in any head-to-head evaluation.

R COMPLETE TRAJECTORY OF A FAILED RUN

This appendix reproduces Run C of Figure 4 in full: Qwen 3.6-35B on `adv-rps-counter-pick` (medium, seed 1), which must destroy eight enemy units before tick 5,400 while keeping its construction yard. The run ends in a loss after 60 decisions with three kills. We give the system prompt once, then for every decision turn the briefing the model received, the model's saved response (its reasoning text; the visible content field is empty except for whitespace), and the tool calls it issued. Each tool call received the acknowledgement `ok`, which confirms receipt, not execution; the effect of each command appears only in the next briefing. The archived conversation history strips the minimap image from earlier briefings; the image attached to the final briefing is shown in Figure 6. Text is reproduced verbatim from the hash-checked source files listed in `revisions/iclr2027/section4_evidence/manifest.json`, except that six non-ASCII symbols (em and en dashes, >=, <=, a middle dot, and a multiplication sign) are shown in ASCII form.

SYSTEM PROMPT

```
You are playing Command & Conquer: Red Alert.
Each turn you receive a BRIEFING with the current game state and a MINIMAP image.
Issue orders to your units via tool calls; the game advances asynchronously and
interrupts you when something happens (unit arrived, enemy spotted, etc.).

Unit format in briefings (one unit per line):
  <id> <type> @(<x>,<y>)                          unit is idle / not moving
  <id> <type> @(<x>,<y>), moving to (<tx>,<ty>)   unit is en route to (tx,ty)
Example:
  1004 jeep @(15,27)
  1005 jeep @(15,27), moving to (40,30)

When issuing tool calls, pass the numeric unit ID (e.g. unit_ids=[1004]).

Idle units are also listed at the end as `Idle: 1004, 1006, ...`
for quick scanning.

Every turn MUST include at least one tool call.

Combat model (read once, applies to every turn):
  * Two ways to fight:
      - `move` to a cell -- units take the path; auto-fire at any hostile
        in weapon range *opportunistically* (no detour); they stop when
        they reach the cell. Use this to POSITION (e.g. flank, retreat).
      - `attack_target(unit_ids, target_id)` -- units pathfind toward the
        named actor and focus-fire it, even if other enemies are closer.
        Use this for PRIORITY: kill THIS tesla, ignore the e3 next to it.
        After the target dies, units go idle and you re-decide on the
        next briefing.
  * "Range" in the codex is in cells. A unit with rng=4.8c attacks
    targets at <=4.8 cells away. A defense with rng=7c hits you at 7 cells.
  * If your range < enemy range, you take damage before you can hit back.
    Counter by:
      (a) flanking from multiple sides so attention splits, or
      (b) sending sacrificial high-HP units (2tnk, apc) to soak hits while
          longer-burst units do damage, or
      (c) bypassing the threat entirely (route around).
  * DPS in the codex assumes sustained fire; concentrating 4 tanks on one
    tesla coil burns it down 4x faster than sending one at a time. Send
    GROUPS via `attack_target` with the same target_id, not single units.
  * Sight (`sight Nc`) means you see N cells out from that unit -- if your
    sight >= enemy attack range, scout safely from outside their range.

OBJECTIVE (this scenario): Cash $2550, full base online. A centre garrison sits in fog -- scout with a jeep
      first, then commit the budget to the matching counter. Kill eight enemy units within about 60 turns,
      keeping your construction yard.
WIN WHEN: destroy >=8 enemy units AND own >=1 'fact' building(s) AND before game tick 5400.
YOU LOSE IF: (not before game tick 5401 OR NOT (own >=1 'fact' building(s))).
You have at most 61 decision turns; acting decisively and early matters.

UNIT CODEX (full RA reference; identical for every model).
Each row: <code> <role> -- $cost hp<HP> rng<attack range, cells> dps<damage/sec> sight<vision, cells> <movement>
       [power].
Engine HP / damage scale = vendor / 100; DPS = damage*burst*25 / reload*100 (the engine ticks at 25/sec).

Infantry:
  e1     rifle infantry (cheap, anti-infantry) -- $100 hp50 rng5.0c dps12.5 sight4.0c foot
  e2     grenadier infantry -- $150 hp50 rng4.0c dps25.0 sight4.0c foot
```

```
  e3     rocket soldier (anti-vehicle/anti-air) -- $300 hp45 rng7.5c dps12.0 +Dragon(rng5.0c/dps25.0) sight4.0c
      foot
  e6     engineer (captures/repairs buildings; unarmed) -- $400 hp25 unarmed sight4.0c foot
  dog    attack dog (fast, anti-infantry only) -- $200 hp18 rng3.0c dps2500.0 sight5.5c foot
  medi   field medic (heals nearby infantry; unarmed) -- $200 hp60 rng4.0c dps-15.6 sight3.0c foot
  spy    spy (infiltrates enemy buildings) -- $500 hp25 rng2.5c dps46.9 sight5.0c foot
  thf    thief (steals enemy credits) -- $500 hp80 unarmed sight5.0c foot
  tanya  Tanya (Allied hero; Colt45 vs infantry, C4 vs buildings) -- $1800 hp100 rng7.0c dps357.1 sight6.0c foot

Vehicles:
  1tnk   light tank (medium armour, anti-vehicle) -- $700 hp230 rng4.8c dps29.8 sight5.0c tracked
  2tnk   heavy/medium tank (main battle tank) -- $850 hp460 rng4.8c dps20.0 sight6.0c tracked
  3tnk   heavy tank (Soviet, strong armour) -- $1150 hp600 rng4.8c dps28.6 sight6.0c tracked
  4tnk   mammoth tank (very heavy, dual cannon+missile) -- $2000 hp900 rng4.8c dps33.3 +MammothTusk(rng6.5c/
      dps41.7) sight6.0c tracked
  jeep   ranger/jeep (fast, light, ideal scout) -- $500 hp150 rng4.0c dps41.7 sight7.0c wheeled
  apc    armoured personnel carrier (transports infantry) -- $850 hp350 rng4.0c dps41.7 sight5.0c tracked
  arty   artillery (long-range, fragile) -- $850 hp100 rng12.0c dps67.6 sight5.0c tracked
  harv   ore harvester (gathers ore for credits; unarmed) -- $1100 hp600 unarmed sight4.0c tracked
  mcv    mobile construction vehicle (deploys into a fact) -- $2000 hp600 unarmed sight4.0c tracked
  v2rl   V2 rocket launcher (Soviet long-range artillery) -- $900 hp200 rng10.0c dps5.2 sight5.0c wheeled
  mnly   minelayer (drops mines) -- $800 hp300 unarmed sight5.0c tracked
  mgg    mobile gap generator (creates shroud) -- $1000 hp220 unarmed sight6.0c wheeled
  mrj    mobile radar jammer -- $1000 hp220 unarmed sight7.0c wheeled
  ttnk   tesla tank (Soviet mobile shocker) -- $1350 hp400 rng7.0c dps20.8 sight7.0c tracked
  ftrk   flak truck (Soviet anti-air vehicle) -- $600 hp150 rng8.0c dps30.0 +FLAK-23-AG(rng6.0c/dps50.0) sight6
      .0c wheeled
  dtrk   demolition truck (suicide explosive) -- $2500 hp28 rng2.0c dps0.0 sight4.0c wheeled
  ctnk   chrono tank (Allied teleporter) -- $1350 hp400 rng6.0c dps41.7 sight6.0c tracked

Aircraft:
  heli   Hind attack helicopter (flies over terrain/buildings; anti-ground) -- $2000 hp120 rng4.0c dps50.0 +
      HellfireAG(rng5.0c/dps88.2) sight12.0c aircraft
  hind   Hind gunship (Soviet attack helicopter) -- $1500 hp100 rng5.0c dps125.0 sight10.0c aircraft
  yak    Yak fighter (Soviet ground-attack plane) -- $1350 hp60 rng5.0c dps666.7 sight11.0c aircraft
  mig    MiG attack plane (Soviet) -- $2000 hp80 rng9.0c dps70.0 sight13.0c aircraft
  tran   Chinook transport helicopter -- $900 hp140 unarmed sight8.0c aircraft
  badr   Badger bomber (paradrop carrier) -- hp400 unarmed aircraft
  u2     U-2 spy plane (recon) -- hp2000 unarmed aircraft
  mh60   Black Hawk helicopter -- $1500 hp100 rng5.0c dps125.0 sight10.0c aircraft

Ships:
  dd     destroyer (Allied naval; ranged shore bombardment) -- $1000 hp400 rng7.5c dps20.8 +DepthCharge(rng5.0c/
      dps25.0) sight5.0c naval
  ca     cruiser (Allied long-range naval artillery) -- $2400 hp800 rng20.0c dps5.0 sight7.0c naval
  ss     submarine (Soviet stealth naval, anti-ship) -- $950 hp250 rng9.0c dps90.0 sight8.0c naval
  msub   missile submarine (Soviet long-range) -- $2000 hp400 rng20.0c dps4.2 +SubMissileAA(rng8.0c/dps0.8)
      sight8.0c naval
  pt     PT boat (light naval, anti-sub) -- $500 hp200 rng5.5c dps10.4 +DepthCharge(rng5.0c/dps25.0) sight8.0c
      naval
  lst    landing craft / transport -- $500 hp400 unarmed sight6.0c naval

Structures:
  fact   construction yard (builds structures; LOSS-CRITICAL base) -- $2000 hp1500 sight5.0c power+0
  powr   power plant (supplies power to structures) -- $300 hp400 sight4.0c power+100
  apwr   advanced power plant -- $500 hp700 sight5.0c power+200
  proc   ore refinery (turns ore into credits; economy core) -- $1400 hp900 sight5.0c power-30
  barr   Soviet barracks (trains infantry) -- $500 hp600 sight5.0c power-20
  tent   Allied barracks (trains infantry) -- $500 hp600 sight5.0c power-20
  weap   war factory (builds vehicles) -- $2000 hp1500 sight5.0c power-30
  hpad   helipad (builds and rearms helicopters) -- $500 hp800 sight5.0c power-10
  afld   airfield (builds and rearms fixed-wing aircraft; Soviet equivalent of hpad's role for planes) -- $500
      hp1000 sight5.0c power-20
  spen   Allied sub pen (builds naval units) -- $800 hp1000 sight5.0c power-20
  syrd   Soviet shipyard (builds naval units) -- $1000 hp1000 sight5.0c power-20
  fix    service depot (repairs vehicles) -- $1200 hp800 sight5.0c power-30
  dome   radar dome (reveals map / tech prerequisite) -- $1500 hp1000 sight10.0c power-40
  silo   ore silo (stores surplus ore) -- $150 hp300 sight4.0c power-10
  atek   advanced tech centre (gates top-tier units / superweapons) -- $1500 hp600 sight5.0c power-200
  stek   Soviet tech centre (gates top-tier units) -- $1500 hp800 sight5.0c power-100
  iron   iron curtain (Soviet superweapon: temporary invulnerability) -- $2000 hp1000 sight6.0c power-200
  mslo   missile silo (Soviet superweapon; launches a single high-damage nuke at a target cell after a charge
      cycle) -- $2500 hp1000 sight6.0c power-150
  pdox   chronosphere (Allied superweapon: teleport unit) -- $1500 hp1000 sight6.0c power-200
  kenn   kennel (prerequisite for attack dogs) -- $200 hp300 sight4.0c power-10
  gap    gap generator (creates shroud over enemy) -- $800 hp500 sight6.0c power-60

Defenses:
  pbox   pillbox (anti-infantry base defence) -- $600 hp400 rng4.0c dps41.7 sight6.0c power-20
  hbox   camo pillbox (hidden anti-infantry) -- $750 hp400 rng4.0c dps41.7 sight6.0c power-20
  gun    gun turret (anti-vehicle base defence) -- $800 hp400 rng6.5c dps50.0 sight6.5c power-40
```

```
  ftur   flame turret (Soviet anti-infantry, short range) -- $600 hp400 rng5.0c dps107.1 sight6.0c power-20
  tsla   Tesla coil (powerful anti-everything defence) -- $1200 hp400 rng7.0c dps833.3 sight7.0c power-80
  sam    SAM site (anti-air defence) -- $700 hp400 rng7.5c dps75.0 sight8.0c power-40
  agun   AA gun (Allied anti-air ground turret) -- $800 hp400 rng10.0c dps75.0 sight6.0c power-50

Walls:
  brik   concrete wall (inert obstacle; no weapon -- channels but does not kill) -- $200 hp400
  sbag   sandbag wall (cheap obstacle) -- $30 hp150
  cycl   chain-link fence (obstacle) -- hp100
  fenc   wood fence (obstacle) -- $30 hp150
  barb   barbed wire (inert obstacle) -- hp100

TECH TREE (faction availability + prerequisites).
`~X` in vendor YAML = 'any of the X-providing structures'; factions are inferred from prereq tags. fact (
      construction yard) is the seed building; place_building does NOT require engine-side adjacency in this
      bench.

Infantry (built at tent/barr):
  e1     Allied+Soviet  prereqs: barracks
  e2     Soviet         prereqs: barr
  e3     Allied+Soviet  prereqs: barracks
  e6     Allied+Soviet  prereqs: barracks
  dog    Allied+Soviet  prereqs: kenn
  medi   Allied         prereqs: tent
  spy    Allied         prereqs: dome, tent
  thf    Soviet         prereqs: barr, dome
  tanya  Allied         prereqs: tent, atek

Vehicles (built at weap):
  1tnk   Allied         prereqs: vehicles.allies
  2tnk   Allied         prereqs: fix, vehicles.allies
  3tnk   Soviet         prereqs: fix, vehicles.soviet
  4tnk   Soviet         prereqs: fix, stek, vehicles.soviet
  jeep   Allied         prereqs: vehicles.allies
  apc    Soviet         prereqs: vehicles.soviet
  arty   Allied         prereqs: dome, vehicles.allies
  harv   Allied+Soviet  prereqs: proc
  mcv    Allied+Soviet  prereqs: fix
  v2rl   Soviet         prereqs: dome, vehicles.soviet
  mnly   Allied+Soviet  prereqs: fix
  mgg    Allied         prereqs: atek, vehicles.england
  mrj    Allied         prereqs: atek, vehicles.allies
  ttnk   Soviet         prereqs: tsla, stek, vehicles.russia
  ftrk   Soviet         prereqs: vehicles.soviet
  dtrk   Soviet         prereqs: stek, vehicles.ukraine
  ctnk   Allied         prereqs: atek, vehicles.germany

Aircraft (built at hpad / afld):
  heli   Allied         prereqs: hpad, atek
  hind   Allied         prereqs: hpad
  yak    Soviet         prereqs: afld
  mig    Soviet         prereqs: afld, stek
  tran   Allied         prereqs: hpad
  mh60   Allied         prereqs: hpad

Ships (built at syrd / spen):
  dd     Allied         prereqs: syrd, dome
  ca     Allied         prereqs: syrd, atek
  ss     Soviet         prereqs: spen
  msub   Soviet         prereqs: spen, stek
  pt     Allied         prereqs: syrd
  lst    Allied+Soviet  prereqs: (none from construction yard alone)

Buildings (built at fact):
  fact   Allied+Soviet  prereqs: (none from construction yard alone)
  powr   Allied+Soviet  prereqs: (none from construction yard alone)
  apwr   Allied+Soviet  prereqs: dome
  proc   Allied+Soviet  prereqs: anypower
  barr   Soviet         prereqs: anypower, structures.soviet
  tent   Allied         prereqs: anypower, structures.allies
  weap   Allied+Soviet  prereqs: proc
  hpad   Allied         prereqs: dome, structures.allies
  afld   Soviet         prereqs: dome, structures.soviet
  spen   Soviet         prereqs: anypower, structures.soviet
  syrd   Allied         prereqs: anypower, structures.allies
  fix    Allied+Soviet  prereqs: weap
  dome   Allied+Soviet  prereqs: proc
  silo   Allied+Soviet  prereqs: proc
  atek   Allied         prereqs: weap, dome, structures.allies
  stek   Soviet         prereqs: weap, dome, structures.soviet
  iron   Soviet         prereqs: stek, structures.soviet
```

```
  mslo   Allied+Soviet  prereqs: techcenter
  pdox   Allied         prereqs: atek, structures.allies
  kenn   Soviet         prereqs: anypower, structures.soviet
  gap    Allied         prereqs: atek, structures.allies

Defenses (built at fact):
  pbox   Allied         prereqs: tent, structures.allies
  hbox   Allied         prereqs: tent, structures.allies
  gun    Allied         prereqs: tent, structures.allies
  ftur   Soviet         prereqs: barr, structures.soviet
  tsla   Soviet         prereqs: weap, structures.soviet
  sam    Soviet         prereqs: dome, structures.soviet
  agun   Allied         prereqs: dome, structures.allies

Walls (built at fact):
  brik   Allied+Soviet  prereqs: fact
  sbag   Allied         prereqs: fact, structures.allies
  fenc   Soviet         prereqs: fact, structures.soviet
```

TURN 1

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3, ~0s game time) ---
Funds: $2550 (cash=$2550 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 5.6%
Base center: (11,21)
Units (2):
  1007 jeep  @(16,20)
  1008 jeep  @(16,22)
Idle: 1007, 1008
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: IDLE
```

Model response

```
The game has just started. I have $2550, my base is online with a Fact, Proc, Powr, Tent, Weap, and Fix. I have
     two jeeps at (16,20) and (16,22).
The objective is to scout the center garrison and then attack. I need to kill 8 enemy units.
I should start by scouting. The jeeps are already positioned well for scouting. I'll send them out to explore
     the map and find the enemy.
I should also start producing units. I have a Weap and Tent. I can build vehicles and infantry.
Let's send the jeeps to scout and start building some units.
I'll produce a 1tnk and an e1.
I'll move the jeeps to explore the map. I'll send them towards the center or where the enemy might be. The
     minimap shows some explored area around (10,20). The enemy is likely somewhere nearby. I'll send them to
     explore the area around (20,20) or further.
Let's move jeep 1007 to (25,20) and jeep 1008 to (25,22) to scout.
I'll also start building a 1tnk (id: 1tnk) and an e1 (id: e1).
Wait, I need to check the power. Power is -10. I need more power. I should build a powr or apwr. But I can only
     build structures at the fact. I have a powr already. Maybe I need another one.
Actually, let's first focus on scouting and building units. I can deal with power later if it becomes an issue,
     but -10 is bad. I should build a powr.
But I also need to produce units. Let's produce a powr and some units.
Actually, let's prioritize scouting and unit production. I'll move the jeeps and start building.
I'll build a powr first to fix power, then build units.
Wait, I can only build one thing at a time at each building. The fact is idle. I should build a powr.
Let's build a powr.
Then I'll produce a 1tnk and an e1.
And move the jeeps to scout.
Let's do:
1. build powr
2. move_units 1007 to (25,20)
3. move_units 1008 to (25,22)
Actually, I can only call one tool at a time? No, I can call multiple. But let's be careful. The briefing says "
     Each turn you receive a BRIEFING... Issue orders...". I can issue multiple orders in one turn.
Let's issue:
- build powr
- move_units 1007 to (25,20)
- move_units 1008 to (25,22)
This should be fine.
Wait, the fact is at (10,20). I need to place the powr somewhere. I'll place it near the base, say (12,18).
Let's do:
build powr
move_units [1007, 1008] to (25,20)
Actually, I'll send them separately to cover more area. 1007 to (25,20), 1008 to (25,22).
Let's proceed.
```

Tool calls

```
build({"item": "powr"})
move_units({"unit_ids": [1007], "target_x": 25, "target_y": 20})
move_units({"unit_ids": [1008], "target_x": 25, "target_y": 22})
```

Commands recorded by the environment

```
Command::Build { item: "powr" }
Command::MoveUnits { unit_ids: ["1007"], target_x: 25, target_y: 20 }
Command::MoveUnits { unit_ids: ["1008"], target_x: 25, target_y: 22 }
```

TURN 2

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 93, ~3s game time) ---
Funds: $2476 (cash=$2476 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 9.1%
Base center: (11,21)
Units (2):
  1007 jeep  @(25,20)
  1008 jeep  @(25,22)
Idle: 1007, 1008
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: powr
```

Model response

```
The jeeps have reached their scouting positions at (25,20) and (25,22) and are now idle. I need to continue
      scouting further into enemy territory to find the enemy base and any garrison mentioned in the objective.
      I'll move them further out to explore more of the map. I'll also need to build more structures and units,
      but scouting is priority #1 to understand what I'm facing.

Let me move the jeeps further out to explore.
```

Tool calls

```
move_units({"unit_ids": [1007], "target_x": 35, "target_y": 20})
move_units({"unit_ids": [1008], "target_x": 35, "target_y": 22})
```

Commands recorded by the environment

```
Command::MoveUnits { unit_ids: ["1007"], target_x: 35, target_y: 20 }
Command::MoveUnits { unit_ids: ["1008"], target_x: 35, target_y: 22 }
```

TURN 3

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 183, ~7s game time) ---
Funds: $2401 (cash=$2401 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 12.9%
Base center: (11,21)
Units (2):
  1007 jeep  @(35,20)
  1008 jeep  @(35,22)
Idle: 1007, 1008
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: powr
```

Model response

```
The jeeps have moved further out and explored more of the map. I can see some green areas (grass) around (35,
      20) and (35, 22), but still no enemies visible. The power is still negative (-10), which means I need more
       power plants.

I should continue scouting with the jeeps to find the enemy base and any defenses. I should also start building
      more power plants and potentially a harvester to start gathering ore since I have no ore yet.

Let me move the jeeps further out to explore more, and also start building a harvester or more power plants.
      Actually, I should check if I have enough power first. I have -10 power, so I need at least one more power
       plant (which provides +100 power). The current production is a powr, so once that finishes, I'll have +90
       power (100 from new powr - 10 existing deficit).
```

```
Let me move the jeeps further out and also queue up more buildings. I need to expand my economy and military.
```

Tool calls

```
build({"item": "powr"})
move_units({"unit_ids": [1007], "target_x": 45, "target_y": 20})
move_units({"unit_ids": [1008], "target_x": 45, "target_y": 22})
```

Commands recorded by the environment

```
Command::Build { item: "powr" }
Command::MoveUnits { unit_ids: ["1007"], target_x: 45, target_y: 20 }
Command::MoveUnits { unit_ids: ["1008"], target_x: 45, target_y: 22 }
```

TURN 4

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 273, ~10s game time) ---
Funds: $2326 (cash=$2326 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 16.7%
Base center: (11,21)
Units (2):
  1007 jeep  @(45,20)
  1008 jeep  @(45,22)
Idle: 1007, 1008
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: powr, powr
```

Model response

```
The jeeps have reached (45,20) and (45,22). The minimap shows some explored terrain around them, but still no
      enemies visible. The power is still -10, which means the two power plants being built haven't completed
      yet (or maybe they just completed and I need to wait). Let me check the production queue: it shows "powr,
      powr" which means two power plants are still in production.

I should continue scouting further east to find the enemy base and any garrison. I'll move the jeeps further
      east.

I also need to start producing units for combat. Since I'm Allies, I can produce e1 infantry, 1tnk, etc. But
      first, let me get more map information.

Let me move the jeeps further east and also start building some combat units. I should build a tent (already
      have it) and then produce e1s or jeeps. Actually, I already have two jeeps scouting. I should start
      producing more units.

Let me produce some e1 infantry and also continue scouting.
```

Tool calls

```
build({"item": "e1"})
build({"item": "e1"})
move_units({"unit_ids": [1007], "target_x": 55, "target_y": 20})
move_units({"unit_ids": [1008], "target_x": 55, "target_y": 22})
```

Commands recorded by the environment

```
Command::Build { item: "e1" }
Command::Build { item: "e1" }
Command::MoveUnits { unit_ids: ["1007"], target_x: 55, target_y: 20 }
Command::MoveUnits { unit_ids: ["1008"], target_x: 55, target_y: 22 }
```

TURN 5

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 363, ~14s game time) ---
Funds: $2176 (cash=$2176 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 20.5%
Base center: (11,21)
Units (2):
  1007 jeep  @(55,20)
  1008 jeep  @(55,22)
Idle: 1007, 1008
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
```

```
Enemies: none visible
Production: powr, powr, e1, e1
```

Model response

```
The jeeps have reached around (55,20) and (55,22). Looking at the minimap, there's a large patch of explored
      territory (green) around them, but the area further east (right) is still dark (unexplored). I should
      continue pushing forward to find the enemy base. I also need to build a harvester to get ore and keep the
      economy going. Let's keep scouting and start building a harvester.

Turn 363:
- Funds: $2176
- Power: -10 (need more power plants, currently building 2)
- Units: 2 jeeps at (55,20), (55,22)
- Buildings: fact, proc, 1 powr, tent, weap, fix
- Production: 2 powr, 2 e1

Actions:
1. Move jeeps further east to scout more. Target (65, 20) and (65, 22).
2. Build a harvester. Cost $1100. I have $2176.
3. Continue building power plants or infantry as needed.

Let's move the jeeps and build a harvester.
Wait, I should also build a harvester soon. $1100 is affordable.
Let's move jeeps to (65, 20) and (65, 22) and build a harvester.
Also, I should build another power plant since power is -10 and I have 2 in production. Actually, the briefing
      says "Production: powr, powr, e1, e1". So 2 powrs are already queued. I'll wait for those.
I'll build a harvester.
Let's proceed.
Actually, I'll just move the jeeps further and build a harvester.
Target for jeeps: (65, 20), (65, 22).
Build: harv.
Let's do it.
Wait, should I build a dome? It reveals the map and is a tech prerequisite. But I need to focus on scouting and
      economy first. Harvester is priority.
I'll move jeeps and build harvester.
Let's execute.
One detail: "Production: powr, powr, e1, e1" means 4 items in queue. I'll add harv to the queue.
Actually, I'll just build the harvester now.
Moving jeeps: [1007] to (65, 20), [1008] to (65, 22).
Building: harv.
Let's go.
Wait, checking power: -10. I need power plants. The queue has 2 powrs. Each gives +100 power. So that will fix
      it.
I'll proceed.
Action: move_units, build.
Let's do it.
Actually, I'll move them to (65, 20) and (65, 22).
Build harv.
Done.
Proceed.
(Self-Correction/Refinement during thought)
I should check if there's any enemy nearby. The jeeps are at 55. The minimap shows explored area up to ~60. I'll
       push to 65.
I'll also build a harvester.
Let's execute.
`move_units([1007], 65, 20)`
`move_units([1008], 65, 22)`
`build("harv")`
Ready.
Output matches.
Proceed.
[Output Generation]
```

Tool calls

```
build({"item": "harv"})
move_units({"unit_ids": [1007], "target_x": 65, "target_y": 20})
move_units({"unit_ids": [1008], "target_x": 65, "target_y": 22})
```

Commands recorded by the environment

```
Command::Build { item: "harv" }
Command::MoveUnits { unit_ids: ["1007"], target_x: 65, target_y: 20 }
Command::MoveUnits { unit_ids: ["1008"], target_x: 65, target_y: 22 }
```

TURN 6

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 453, ~18s game time) ---
Funds: $1953 (cash=$1953 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.3%
Base center: (11,21)
Units (3):
  1022 e1    @(13,17)
  1007 jeep  @(65,20)
  1008 jeep  @(65,22)
Idle: 1022, 1007, 1008
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: 11xe1[1009@(68,17),1010@(68,18),1011@(68,19),1012@(68,20),1013@(68,21),1014@(68,22),1015@(68,23),1016@
      (70,18),1017@(70,19),1018@(70,21),1019@(70,22)] center (68,20)
Production: harv, powr, powr, e1
```

Model response

```
I've discovered a large enemy infantry force (11x e1) at (68, 17-23). I have two jeeps at (65, 20) and (65, 22)
      and one e1 at (13, 17). The enemy is very close to my jeeps.

My jeeps have rng=4.0c and dps=41.7. The enemy e1s have rng=5.0c and dps=12.5. My jeeps outrange the enemy e1s
      slightly (4.0 vs 5.0... wait, 5.0 > 4.0, so the enemy outranges me). But jeeps are fast and can kite.
      However, 11 vs 2 is a bad matchup if I just send them in.

Wait, looking at the minimap, the jeeps are at 65,20 and 65,22. The enemy is at 68,17-23. The distance is about
      3 cells horizontally. The enemy has range 5.0c, jeeps have 4.0c. The enemy can shoot me before I can shoot
       them if I move directly towards them.

However, I have a large infantry force coming. I should probably hold my jeeps back or use them to kite, but
      more importantly, I need to bring more forces. I have e1s being produced. I should also consider building
      more structures to produce tanks or more infantry.

Actually, looking at the situation: 11 enemy e1s. My jeeps can kite them, but I need to kill them. I should
      bring my existing e1 (1022) to help, and queue more e1s. I should also consider building a dome to unlock
      more tech, but that takes time.

For now, let's move the jeeps to engage or kite the enemy while I produce more forces. I'll move the jeeps
      forward to engage, and move my e1 forward as well. I should also queue more e1s and maybe start producing
      a tank if I can afford it.

Wait, I have $1953. I can produce:
- 2 e1s = $200
- 1 harv = $1100
- 1 tent = $500
- 1 dome = $1500

I should probably produce more infantry (e1s) to match the enemy numbers, and maybe a harv to keep economy going.
       But I also need to defend against this attack.

Let's move my units to engage. The jeeps can kite the e1s. I'll move 1007 and 1008 to attack the enemy cluster,
      and move 1022 to support.

Actually, looking at the enemy positions: they're clustered at (68,17-23) and (70,18-22). My jeeps are at
      (65,20) and (65,22). If I move them to attack, they'll take damage from the range advantage. But jeeps are
       fast and can kite.

Let me move the jeeps to attack the enemy, and move my e1 to support. I'll also queue more e1s.
```

Tool calls

```
attack_unit({"target_id": 1009, "unit_ids": [1007, 1008]})
move_units({"unit_ids": [1022], "target_x": 60, "target_y": 20})
build({"item": "e1"})
build({"item": "e1"})
```

Commands recorded by the environment

```
Command::AttackUnit { unit_ids: ["1007", "1008"], target_id: "1009" }
Command::MoveUnits { unit_ids: ["1022"], target_x: 60, target_y: 20 }
Command::Build { item: "e1" }
Command::Build { item: "e1" }
```

TURN 7

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 543, ~21s game time) ---
Funds: $1728 (cash=$1728 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
```

```
Units (2):
  1022 e1    @(17,17), moving to (60,20)
  1023 e1    @(13,17)
Idle: 1023
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, powr, powr, e1, e1
```

Model response

```
The jeeps moved towards the enemy and were likely destroyed since they're no longer listed. The e1 infantry is
      moving towards the enemy position. I should check the status of the battle and possibly send more units to
       reinforce. Let me also check if the harvester is built and start gathering ore.

Given the enemy was at (68,17)-(70,22) and my jeeps moved to engage, they're probably dead. I need to send more
      forces. I have 2 e1s in production and 1 idle e1. I should build more units, perhaps vehicles from the war
       factory.

Let me build some 1tnk units and maybe more infantry to push forward. Also, I need to make sure I have a
      harvester to gather ore for sustained production.
```

Tool calls

```
build({"item": "1tnk"})
build({"item": "1tnk"})
build({"item": "1tnk"})
move_units({"unit_ids": [1022], "target_x": 68, "target_y": 20})
```

Commands recorded by the environment

```
Command::Build { item: "1tnk" }
Command::Build { item: "1tnk" }
Command::Build { item: "1tnk" }
Command::MoveUnits { unit_ids: ["1022"], target_x: 68, target_y: 20 }
```

TURN 8

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 633, ~25s game time) ---
Funds: $1502 (cash=$1502 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (3):
  1022 e1    @(21,17), moving to (67,19)
  1023 e1    @(13,17)
  1024 e1    @(14,17)
Idle: 1023, 1024
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, powr, powr, e1
```

Model response

```
The jeeps are gone (probably destroyed or retreated off-map, but the briefing doesn't list them). The e1s are
      moving toward the enemy position. I need to keep building up my forces and also get a harvester online to
      sustain my economy. I should also build a dome for vision since I'm going on the offensive.

Let's continue building tanks and infantry, and send the e1s to engage. I'll also build a dome to reveal more of
       the map and unlock more tech.
```

Tool calls

```
build({"item": "dome"})
build({"item": "e1"})
move_units({"unit_ids": [1022], "target_x": 68, "target_y": 20})
```

Commands recorded by the environment

```
Command::Build { item: "dome" }
Command::Build { item: "e1" }
Command::MoveUnits { unit_ids: ["1022"], target_x: 68, target_y: 20 }
```

TURN 9

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 723, ~28s game time) ---
Funds: $1277 (cash=$1277 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (3):
  1022 e1    @(26,17), moving to (68,20)
  1023 e1    @(13,17)
  1024 e1    @(14,17)
Idle: 1023, 1024
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, powr, powr, dome, e1, e1
```

Model response

```
I see the map has been explored more. The jeeps (1007, 1008) are no longer listed, which means they might have
      been destroyed or they reached their destination and are idle but not listed separately. Looking at the
      brief, only e1 units are listed. The enemy was spotted at tick 453 at (68,17-23) but now at tick 723 there
       are "Enemies: none visible". This is strange. Let me check the minimap - it shows explored area but no
      enemy indicators. The jeeps might have been destroyed in combat with the enemy infantry that was spotted
      earlier.

Wait, looking more carefully at the briefing: "Enemies: none visible" - this could mean the enemy units moved
      out of sight or were destroyed. But my jeeps are also gone from the unit list. Let me assume they were
      destroyed in the engagement.

I should continue building my army and sending my e1 to scout/engage. The e1 at (26,17) is moving to (68,20). I
      should also build more units and maybe a harvester to get ore income.

Let me check my production queue: harv, 1tnk, 1tnk, 1tnk, powr, powr, dome, e1, e1. That's a lot of production.
      I should focus on getting tanks out and maybe a harvester.

Let me also check if I need to build a harvester to get ore income. I have no harvesters yet.

I'll continue building and moving my e1 forward.
```

Tool calls

```
build({"item": "harv"})
move_units({"unit_ids": [1022], "target_x": 68, "target_y": 20})
```

Commands recorded by the environment

```
Command::Build { item: "harv" }
Command::MoveUnits { unit_ids: ["1022"], target_x: 68, target_y: 20 }
```

TURN 10

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 813, ~32s game time) ---
Funds: $1053 (cash=$1053 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (4):
  1022 e1    @(31,17), moving to (68,20)
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
Idle: 1023, 1024, 1025
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, e1
```

Model response

```
The 1tnks haven't finished production yet, and my e1 is still moving towards the enemy position (68,20). I need
      to keep scouting and preparing for the fight. I should also ensure I have enough power and economy. The
      harvester is being built. I'll continue moving the e1 scout and wait for the tanks to finish.
```

Tool calls

```
build({"item": "e1"})
move_units({"unit_ids": [1022], "target_x": 68, "target_y": 20})
```

Commands recorded by the environment

```
Command::Build { item: "e1" }
```

```
Command::MoveUnits { unit_ids: ["1022"], target_x: 68, target_y: 20 }
```

TURN 11

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 903, ~36s game time) ---
Funds: $828 (cash=$828 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1022 e1    @(36,17), moving to (68,20)
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
Idle: 1023, 1024, 1025, 1026
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, e1
```

Model response

```
I have 3 light tanks (1tnk) in production, and they should be ready soon. I also have a harvester in production.
       I need to wait for the tanks to finish and then use them to attack the enemy infantry I spotted earlier.
      For now, I'll continue building more units to ensure I have enough to overwhelm the enemy. I'll also build
       more power plants to ensure I have enough power for all my buildings.
```

Tool calls

```
build({"item": "powr"})
build({"item": "powr"})
```

Commands recorded by the environment

```
Command::Build { item: "powr" }
Command::Build { item: "powr" }
```

TURN 12

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 993, ~39s game time) ---
Funds: $602 (cash=$602 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (6):
  1022 e1    @(40,17), moving to (68,20)
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
We are currently at turn 993.
Funds: $602.
Units: Several e1 infantry, moving towards (68,20) where enemies were spotted.
Buildings: fact, proc, powr, tent, weap, fix.
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr.
I have 3 light tanks and 2 harvesters being built, plus power plants and a dome.
I should continue waiting for the tanks to be ready and the e1s to reach the enemy.
The e1 at (40,17) is moving to (68,20).
I will observe to let production finish.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 13

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1083, ~43s game time) ---
Funds: $452 (cash=$452 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (6):
  1022 e1    @(45,17), moving to (68,20)
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The tanks haven't finished producing yet. I need to wait for them. Meanwhile, I should continue scouting with
      the e1 (1022) that's moving towards the enemy. I also have several idle e1s that I could use for something,
       but with low funds ($452), I can't build much. I'll just observe and wait for the tanks to be ready.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 14

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1173, ~46s game time) ---
Funds: $302 (cash=$302 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (6):
  1022 e1    @(50,17), moving to (68,20)
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The 1tnks and harvesters are still producing. I need to wait for them to finish. Let me observe another turn.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 15

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1263, ~50s game time) ---
Funds: $152 (cash=$152 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (6):
  1022 e1    @(55,17), moving to (68,20)
  1023 e1    @(13,17)
```

```
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
We have 152 funds. We are waiting for our production to finish (1tnk x3, harv x2, powr x4, dome x1).
Our e1 scout (1022) is at (55,17) moving to (68,20).
We should wait for our tanks to finish and for the scout to reach the enemy or for ore to come in.
Since we have very little cash, we can't build anything new right now. We must wait for the current production
      queue.
Let's observe to see if the tanks finish or if we get ore.
Actually, looking at the minimap, we have some explored areas. The enemy base seems to be around (68, 20) based
      on previous turns, but they are not visible now. Maybe they moved or are in fog.
Let's just observe and wait for our production to complete.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 16

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1353, ~54s game time) ---
Funds: $2 (cash=$2 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (6):
  1022 e1    @(59,17), moving to (68,20)
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The 1tnks and harvesters should be finishing production soon. I need to wait for them and then use them to
      attack the enemy. Let me observe to see if the units are ready.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 17

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1443, ~57s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
```

```
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The jeep 1007 and 1008 that I sent to scout have disappeared from the unit list. This likely means they were
      destroyed by the enemy forces I spotted earlier at (68,20). The enemy had 11 e1 units there. My e1 scout
      (1022) was moving towards them but also seems to have disappeared from the list now.

I need to wait for my production to complete - 3x 1tnk, 2x harv, and several powr units. I should also consider
      that I need to get my harvester to an ore field. Looking at the minimap, I can see some green terrain to
      the east. I should wait for my tanks to be ready and then push east to engage the enemy.

For now, I'll just observe and wait for production to complete.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 18

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1533, ~61s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The jeep 1007 and 1008 are gone - they must have reached the enemy position and engaged or been destroyed. I see
       no enemy units visible yet, which means the enemy might be in shroud or my scouts are still engaging. Let
       me wait for the next briefing to see what happened with the scouts and if the tanks/harvesters are done
      producing.

I have 5 idle e1s at my base. No funds to build anything. I should wait for production to complete and for the
      scouts to report back.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 19

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1623, ~64s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
```

```
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The harv should be arriving soon or already there. Let me check if there's any update on the harvester
      production. The production queue shows "harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr". The
      first harv should have finished by now (tick 1623, harv takes ~30s = ~750 ticks). But I don't see a harv
      unit listed. Let me observe again to see if the harv has spawned and if the jeeps have reached the enemy.

Actually, looking at the minimap, I see a large green area (grass) extending to the right. The jeeps were moving
       to (68,20) and (68,22) initially, but then I stopped tracking them because the briefing didn't show them
      anymore (they might have been lost or the briefing truncated). Wait, the briefing only shows 5 e1 units.
      The jeeps (1007, 1008) and e1 (1022) are not listed. They might have died or the briefing is just showing
      idle units.

Let me check the production queue again. It's full of powr and 1tnk. I need to get those harvesters out to get
      ore, then get credits to build the tanks.

Since I have $0, I can't build anything. I must wait for the harvesters to start producing ore/credits. Let me
      observe until the harv appears.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 20

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1713, ~68s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The harvester should have arrived at the ore field by now. Let me check if it's there and if I can start mining.
       The e1 scout (1022) was moving toward (68,20) where enemies were spotted earlier. I should check if the
      harvester is idle and ready to collect ore. Since I have 0 funds, I need to get ore mining going. Let me
      observe to see the current state of the harvester and the scout.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 21

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1803, ~72s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
```

```
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
I need to check if my harvester has returned and if my tanks are ready. The briefing shows no new units and no
     funds. I should continue waiting for production to complete and for the harvester to return with ore. Let
     me observe again.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 22

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1893, ~75s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
I see that the harvester and 1tnk production has completed. The harvester should be available to build, and the
     tanks should be ready. Let me check the current state. The briefing shows no new units listed, which is
     odd since I queued harv and 1tnk units. Wait, looking at the production queue: "harv, 1tnk, 1tnk, 1tnk,
     harv, powr, powr, dome, powr, powr". The first harv and first 1tnk should have completed by now (tick 1893)
     . But they aren't listed in the units section. This might mean they are still being produced or there's a
     delay.

Actually, looking more carefully, the units listed are only the e1s. The production queue is long. I need to
     wait for production to finish. But I also notice the minimap shows explored area around (15,17) where the
     e1s are. The e1 scout (1022) was moving to (68,20) but is no longer listed, which means it might have been
      destroyed or reached its destination and gone idle (but not listed as idle). Wait, the briefing says "
     Idle: 1023, 1024, 1025, 1026, 1027". So 1022 is gone. It probably reached (68,20) or was destroyed. Since
     I don't see enemy units, it likely reached the scouting position or is idle but not listed (maybe it's
     still moving or something). Actually, the briefing format says "Idle: 1004, 1006, ..." so if 1022 isn't
     listed, it might be dead or still moving. Given the distance, it's likely dead or off-map.

I need to focus on economy and production. The harvester and tanks should complete soon. I should also build a
     dome for radar and tech. But I have no funds. I need to wait for ore harvesting.

Wait, I have a proc (ore refinery) and should have a harvester. The harvester needs to be built and then sent to
      ore. But the harvester isn't listed as a unit yet. It's in production.

I should continue waiting for production to complete, then build more and send harvester to ore. But I also need
      to scout the enemy base. The e1 scout is gone. I should build more scouts or wait for tanks.

Given the long production queue and no funds, I'll just observe until production completes. But I need to issue
     at least one tool call per turn. I'll observe.

However, I notice the minimap shows the explored area. The enemy base is likely to the east or northeast. I
     should send the idle e1s to scout as well, but they are cheap infantry and might die. Alternatively, I can
      wait for the 1tnks.

But I have no funds, so I can't build anything new. I must wait.

I'll observe.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 23

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 1983, ~79s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The minimap now shows that the area around our base (11,21) is explored (green), but the area to the east (where
       unit 1022 was moving) is still unexplored (dark). The e1 scout (1022) is no longer listed, which means it
       likely died or got lost in the fog. I need to check if there are any ore patches nearby to start economic
       development. Looking at the minimap, there's a large green area to the east, but no visible ore. I should
       wait for the production queue to complete and see if any harvesters or tanks come online. Since I have no
       funds and my scout is gone, I need to be patient and wait for my buildings to produce units. Let me
      observe to see if anything changes.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 24

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2073, ~82s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
I'm still waiting for the harvester to finish production and the e1 scout to return with info. No funds to build
       anything new. I'll keep observing.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 25

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2163, ~86s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The previous observations have been idle, waiting for production to complete. I have no funds to build anything.
      The power is negative (-10), which means I'm running out of power. I need to wait for the power plants
     and harvesters to complete.

Looking at the minimap, I can see my base is at (11,21). The area to the right (east) is explored (dim green),
     but there's a large unexplored area to the south/southeast. The unit 1022 (e1) was sent to (68,20) but it
     seems to have disappeared from the list, likely killed or it moved out of the briefing's visible range.

Since I have no units and no funds, I must wait for production to finish. The production queue has: harv, 1tnk,
     1tnk, 1tnk, harv, powr, powr, dome, powr, powr.

Let's wait for the first harvester to complete so I can start generating income.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 26

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2253, ~90s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.4%
Base center: (11,21)
Units (5):
  1023 e1    @(13,17)
  1024 e1    @(14,17)
  1025 e1    @(15,17)
  1026 e1    @(16,17)
  1027 e1    @(13,18)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
I'm in a very precarious situation. I have $0, no harvesters, and no power. My production queue is full of
     things I can't build (harv, 1tnk, powr, dome) because I lack funds and ore. My only units are 5 idle e1s.

I need to:
1. Get funds ASAP - but I have no harvesters and no ore mining.
2. The production queue is blocking me from building anything useful.
3. I need to scout to find ore or enemy positions.

Wait, looking at the minimap, I can see my base area around (10-16, 17-24). The rest of the map is unexplored (
     dark). I have no harvesters to gather ore, and I have $0 cash.

The most urgent issue is that I have a full production queue that I can't afford. I should cancel it or at least
      stop wasting time. But more importantly, I need to get to an ore field.
```

```
Looking at my units: I have 5 e1s. One e1 (1022) was moving to (68,20) but seems to have disappeared from the
      list - it might have been killed or completed its move. The remaining 4 e1s are idle at (13-16, 17) and
      one at (13,18).

I need to send these e1s to scout for ore fields. Ore fields are typically marked on the minimap as yellow/brown
       patches. Looking at the minimap, I can see green grass areas to the east. I should send my e1s to explore.


But wait - I have no power (-10), which means my buildings might be offline. The proc (ore refinery) needs power
       to work. I need to build power plants.

Actually, looking more carefully, I have a production queue full of items. I can't build anything new because I
      have $0. The queue items are: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr. That's 10 items
      queued.

I need to cancel this queue somehow, but I don't see a "cancel production" function. Let me check the available
      functions... No, there's no cancel production function. The queue will continue, but since I have $0, it
      won't actually produce anything until I have funds.

The immediate priority is:
1. Get funds - I need to find ore and build harvesters, or find enemy credits/spy.
2. Get power - I need to build power plants.

Since I have no harvesters and no ore, I can't build anything. I need to scout to find ore fields.

Let me send my e1s to explore eastward where there's unexplored terrain. The e1s have sight 4.0c, so they'll
      reveal the map as they move.

Actually, looking at the minimap again, I see the explored area is a green patch around my base. To the east,
      there's more green (grass) that's dimly explored. I should send my e1s to explore that area.

Let me move the e1s eastward to scout. I'll send them to around (40, 20) or further east.

Wait, I also need to consider that I have no power. My buildings might be offline. But the e1s should still work
       since they're units, not buildings.

Let me move the idle e1s to scout east.
```

Tool calls

```
move_units({"unit_ids": [1023, 1024, 1025, 1026, 1027], "target_x": 50, "target_y": 20})
```

Commands recorded by the environment

```
Command::MoveUnits { unit_ids: ["1023", "1024", "1025", "1026", "1027"], target_x: 50, target_y: 20 }
```

TURN 27

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2343, ~93s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.5%
Base center: (11,21)
Units (5):
  1023 e1    @(15,16), moving to (50,20)
  1024 e1    @(16,16), moving to (49,19)
  1025 e1    @(18,18), moving to (50,19)
  1026 e1    @(20,17), moving to (51,19)
  1027 e1    @(14,21), moving to (49,21)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The five e1 infantry are moving out to scout. I have no funds right now but production is queued: harv, 1tnk, 1
      tnk, 1tnk, harv, powr, powr, dome, powr, powr. The harvester will start producing ore which will generate
      credits. I should wait for the production to complete and for the scouts to find the enemy. Let me observe
       and wait for the scouts to reach their destination and potentially spot enemies.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 28

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2433, ~97s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.6%
Base center: (11,21)
Units (5):
  1023 e1    @(20,16), moving to (50,20)
  1024 e1    @(21,16), moving to (49,19)
  1025 e1    @(23,18), moving to (50,19)
  1026 e1    @(24,17), moving to (51,19)
  1027 e1    @(19,21), moving to (49,21)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The scouts are moving towards the center/right of the map to find ore and enemies.
I have no funds.
I need to wait for the harvester to finish production so I can start collecting ore.
The production queue shows: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr.
So a harvester is being built.
I should wait.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 29

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2523, ~100s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.7%
Base center: (11,21)
Units (5):
  1023 e1    @(25,16), moving to (50,20)
  1024 e1    @(26,16), moving to (49,19)
  1025 e1    @(28,18), moving to (50,19)
  1026 e1    @(29,17), moving to (51,19)
  1027 e1    @(24,21), moving to (49,21)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The e1s are moving out to scout. I should keep observing until they reach their destination or find something.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 30

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2613, ~104s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.8%
Base center: (11,21)
Units (5):
  1023 e1    @(30,16), moving to (50,20)
  1024 e1    @(31,16), moving to (49,19)
  1025 e1    @(32,18), moving to (50,19)
```

```
  1026 e1    @(34,17), moving to (51,19)
  1027 e1    @(28,21), moving to (49,21)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The e1 scouts are moving towards the center-right of the map to scout for enemy forces. I need to keep observing
      until they find something or I get funds to produce more units. The production queue is long and includes
      harvesters and tanks, but I have $0 cash right now. I should wait for the harvesters to complete and
     start generating income, or for the scouts to find the enemy. For now, I'll just observe.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 31

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2703, ~108s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 24.9%
Base center: (11,21)
Units (5):
  1023 e1    @(34,16), moving to (50,20)
  1024 e1    @(35,16), moving to (49,19)
  1025 e1    @(37,18), moving to (50,19)
  1026 e1    @(39,17), moving to (51,19)
  1027 e1    @(33,21), moving to (49,21)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
We have 5 idle e1 infantry at the base. We need to scout. Let's move them towards the center of the map to find
     enemies.
Target coordinates: (60, 20) seems like a reasonable scouting direction.
```

Tool calls

```
move_units({"unit_ids": [1023, 1024, 1025, 1026, 1027], "target_x": 60, "target_y": 20})
```

Commands recorded by the environment

```
Command::MoveUnits { unit_ids: ["1023", "1024", "1025", "1026", "1027"], target_x: 60, target_y: 20 }
```

TURN 32

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2793, ~111s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.0%
Base center: (11,21)
Units (5):
  1023 e1    @(38,17), moving to (60,20)
  1024 e1    @(40,16), moving to (59,19)
  1025 e1    @(42,18), moving to (60,19)
  1026 e1    @(43,17), moving to (61,19)
  1027 e1    @(38,21), moving to (59,21)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
I've been sending my infantry to scout, but I'm still at 25% explored and have $0 funds. I need to stop this
     inefficient scouting and focus on building economy and military. I should:
1. Recall my infantry to base
```

```
2. Build harvesters to get ore
3. Build a dome for radar
4. Start building military units

Let me recall the infantry first, then focus on production.
```

Tool calls

```
stop({"unit_ids": [1023, 1024, 1025, 1026, 1027]})
```

Commands recorded by the environment

```
Command::Stop { unit_ids: ["1023", "1024", "1025", "1026", "1027"] }
```

TURN 33

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2883, ~115s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.0%
Base center: (11,21)
Units (5):
  1023 e1    @(38,17)
  1024 e1    @(40,16)
  1025 e1    @(42,18)
  1026 e1    @(43,17)
  1027 e1    @(38,21)
Idle: 1023, 1024, 1025, 1026, 1027
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The e1s are now idle around (40, 16)-(43, 18). The minimap shows they are in a bright green area, meaning the
      shroud is revealed there. I see no enemy units in the bright green area. The area to the east (x>60) is
      still dark/unexplored.

Since I have $0 and no way to produce anything immediately (production queue is full/ongoing), I should continue
       scouting to find the enemy base. The e1s are at ~x=40. The unexplored area is to the east. I should send
      them further east to scout.

Let's move them to x=70 or so to push into the fog.
```

Tool calls

```
move_units({"unit_ids": [1023, 1024, 1025, 1026, 1027], "target_x": 70, "target_y": 20})
```

Commands recorded by the environment

```
Command::MoveUnits { unit_ids: ["1023", "1024", "1025", "1026", "1027"], target_x: 70, target_y: 20 }
```

TURN 34

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 2973, ~118s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.1%
Base center: (11,21)
Units (5):
  1023 e1    @(42,17), moving to (69,19)
  1024 e1    @(45,16), moving to (71,19)
  1025 e1    @(47,18), moving to (69,21)
  1026 e1    @(48,17), moving to (71,21)
  1027 e1    @(42,21), moving to (69,20)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The scouts are moving right. I need to keep pushing to find the enemy base.
I'll continue moving them towards (80, 20) to scout further.
```

Tool calls

```
move_units({"unit_ids": [1023, 1024, 1025, 1026, 1027], "target_x": 80, "target_y": 20})
```

Commands recorded by the environment

```
Command::MoveUnits { unit_ids: ["1023", "1024", "1025", "1026", "1027"], target_x: 80, target_y: 20 }
```

TURN 35

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3063, ~122s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.2%
Base center: (11,21)
Units (5):
  1023 e1    @(46,17), moving to (80,20)
  1024 e1    @(50,16), moving to (79,19)
  1025 e1    @(51,18), moving to (80,19)
  1026 e1    @(53,17), moving to (81,19)
  1027 e1    @(47,21), moving to (79,21)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
I'm scouting to the east. The minimap shows the explored area is centered around (50,20) with a small green
      patch. My e1 units are moving towards (80,20). I need to continue scouting to find the enemy base and
      units. Let's keep pushing east.
```

Tool calls

```
move_units({"unit_ids": [1023, 1024, 1025, 1026, 1027], "target_x": 90, "target_y": 20})
```

Commands recorded by the environment

```
Command::MoveUnits { unit_ids: ["1023", "1024", "1025", "1026", "1027"], target_x: 90, target_y: 20 }
```

TURN 36

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3153, ~126s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.3%
Base center: (11,21)
Units (5):
  1023 e1    @(51,17), moving to (90,20)
  1024 e1    @(54,16), moving to (89,19)
  1025 e1    @(56,18), moving to (90,19)
  1026 e1    @(58,17), moving to (91,19)
  1027 e1    @(52,21), moving to (89,21)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
The scouts have reached the center of the map (around 50-60, 20) and are exploring. They are currently moving
      towards (90, 20). I should wait for them to reach a position where they can scout effectively or encounter
       the enemy. The minimap shows they are in a green area (grass) and have revealed a significant chunk of
      the map around them.

Since I have no funds ($0), I cannot build anything new. I must rely on the existing production queue (which is
      likely completed or nearly completed, given the idle state) and the scouts to find the enemy.

The scouts are moving right. I'll let them continue scouting. I'll issue another move command to keep them
      pushing forward or wait if they are done. Let's keep them moving to explore more.

Actually, looking at the minimap, they are quite far out. I should check if they found anything. The briefing
      says "Enemies: none visible". So they haven't engaged yet.

I'll continue to move them further to explore more of the map.
```

Tool calls

```
move_units({"unit_ids": [1023, 1024, 1025, 1026, 1027], "target_x": 100, "target_y": 20})
```

Commands recorded by the environment

```
Command::MoveUnits { unit_ids: ["1023", "1024", "1025", "1026", "1027"], target_x: 100, target_y: 20 }
```

TURN 37

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3243, ~129s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units (3):
  1023 e1    @(56,17), moving to (100,20)
  1024 e1    @(59,16), moving to (99,19)
  1027 e1    @(57,21), moving to (99,21)
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
It seems two of my e1 units (1025 and 1026) disappeared or were destroyed, as they are no longer listed. The
      remaining three e1 units (1023, 1024, 1027) are still moving towards (100, 20). I should continue scouting
       and also check on the status of the lost units. However, since I can't see them, I'll assume they were
      lost to fog or terrain. I will continue moving the remaining units further east to scout more of the map.

Wait, looking at the minimap, there is a large patch of visible terrain around (60, 20). The units are moving
      towards (100, 20). I should keep them moving to reveal more of the map. I also need to be careful about my
       base production. The production queue is still running (harv, 1tnk, etc.).

Let's continue moving the remaining scouts.
```

Tool calls

```
move_units({"unit_ids": [1023, 1024, 1027], "target_x": 110, "target_y": 20})
```

Commands recorded by the environment

```
Command::MoveUnits { unit_ids: ["1023", "1024", "1027"], target_x: 110, target_y: 20 }
```

TURN 38

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3333, ~133s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
It seems my previous orders to move the e1s towards (110, 20) caused them to be destroyed or lost, as they are
      no longer listed in the units list. The briefing now says "Units: ?" and "Idle: ?" is missing. This is a
      critical failure. I have lost my only units.

My funds are $0. I have a factory (fact), power plant (powr), barracks (tent), war factory (weap), and fix depot
       (fix). I have no harvesters, no units, and no money.

I need to rebuild my economy immediately.
1.  **Harvesters**: I need to build harvesters to gather ore. I see `harv` in the production queue. I should
      queue more.
2.  **Power**: I have -10 power. I need to build more power plants. I see `powr` in the production queue.
3.  **Defense**: I am vulnerable. I should build some defenses if possible, but economy is priority.
4.  **Recon**: I need to scout to find the enemy. I have no units to do this.

Let's look at the production queue: `harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr`.
It seems the game has been auto-producing based on some previous orders or defaults.
I have 0 funds. I cannot build anything new right now unless the production queue completes.
```

```
Wait, if I have 0 funds and the production queue is full of items, maybe they are already built and I just need
      to collect them? No, usually "Production: ..." lists items currently being built.
If I have 0 funds, I can't afford new builds. I must wait for the queue to finish.

However, looking at the previous turns, I was moving e1s. If they died, I need to replace them.
The briefing says "Units: ?". This is strange. Usually, it lists units. Maybe they are all dead.

I have a `proc` (ore refinery) but no `harv` (harvester). I need to get a harvester to the ore patch.
The ore patch is likely near my base or the area I was scouting.
From the minimap, I can see a green strip. The ore patch might be there.
The green strip is roughly at y=20-30, x=20-60.
My base is at (11, 21).
I should check if I have any ore on the map.

Since I have no units and 0 funds, I am in a very precarious position. I must wait for my production queue to
      finish.
The queue has: `harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr`.
This is a lot of items.
Once the first `harv` is done, I can move it to an ore patch.
Once `powr` is done, I fix power.

I cannot build anything new. I can only observe.
But I must issue a tool call every turn.
I will `observe` to wait for production to finish.

Wait, is there anything else I can do?
Maybe I can cancel production? No, I have no funds to build new ones, but cancelling might free up the factory?
Actually, if I have no funds, I can't build new ones anyway.
The `proc` is there. I just need a harvester.
The `dome` is also in the queue, which is good for revealing the map.

Let's just wait and see. I'll observe.

One thing to note: The previous e1s died. I don't know why. Maybe they fell off the map or were attacked by
      something I didn't see (fog of war).
I need to be more careful with scouting.

For now, I will observe.

One correction: I see `harv` in the production queue. I should make sure it gets built.
Also `powr` is needed for power.
The `dome` is needed for tech and vision.

I will just observe.

Wait, I should check if I can build anything else.
I have `fact` (construction yard).
I have `powr` (power plant).
I have `proc` (ore refinery).
I have `tent` (barracks).
I have `weap` (war factory).
I have `fix` (service depot).

My power is -10. This means I am losing power. Structures might be shutting down.
The `pow
```

Tool calls

```
(no tool call)
```

Commands recorded by the environment

```
Command::Observe
```

TURN 39

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3423, ~136s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr
```

Model response

```
I've been sending my e1s on a wild goose chase to the right side of the map, and they seem to have been lost or
      destroyed. The briefing now shows "Units: ?" which is concerning. I need to get back to my base and start
      producing units properly. My production queue is full of harvesters and tanks that I can't afford to build
       right now with $0 funds. I need to focus on getting ore and building a proper army.

Let me check the minimap. It shows a large green area around my base (11,21) but nothing to the east. The e1s
      were supposed to be scouting, but they're gone. I need to stop this and get back to basics.

First, I should try to produce something useful. But I have $0. I need harvesters to get ore. Looking at my
      production queue, I have "harv, 1tnk, 1tnk, 1tnk, harv, powr, powr, dome, powr, powr" queued. The
      harvester is the first item. Let me try to build it, but I need ore first.

Wait, I have a proc (ore refinery) built at (10,24). I need a harvester to gather ore. But I can't build a
      harvester without funds. This is a catch-22.

Actually, looking at the production queue, it seems like the game is auto-producing these units. But with $0,
      how can that work? Maybe the production is queued but not starting until funds are available.

Let me try to build a harvester manually to see if it triggers anything, or just observe to see what happens
      next.

Actually, I think the issue is that I've been wasting turns moving units around while the enemy is building up.
      I need to stop this nonsense and focus on my economy and military.

Let me try to build a harvester. If I can't, I'll observe and see what happens.
```

Tool calls

```
build({"item": "harv"})
```

Commands recorded by the environment

```
Command::Build { item: "harv" }
```

TURN 40

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3513, ~140s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I have no units and no cash. I need to wait for harvesters to be built and start generating income. Let me
      observe to see if the harvester is built.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 41

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3603, ~144s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
We are in a very early game state.
Funds: $0
```

```
Power: -10
Harvesters: 0
Explored: 25.4%
Base center: (11,21)
Buildings: fact(1001)@(10,20), proc(1002)@(10,24), powr(1003)@(8,18), tent(1004)@(14,18), weap(1005)@(14,22),
      fix(1006)@(12,24)
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, powr, powr, dome, powr, powr

We have 5 e1s that moved to (110, 20) and then disappeared from the briefing. This suggests they might have been
       destroyed or the briefing format is just not showing them (unlikely, usually they are listed). Wait, "
      Units: ?" suggests they are gone or something weird happened. Let's assume they are gone or we just lost
      contact.

We have 0 funds. We need to get money to build things and win.
We have a harvester in production.
We have a power deficit (-10). We are building powr.

The objective is to kill 8 enemy units and keep our fact.
We need to scout. The e1s are gone.
We should probably wait for the harvester to complete and start generating income.
Or maybe we can build something else? No funds.

Let's look at the production queue. It's full of harvesters, tanks, and power plants.
We need to get the harvester out to get ore/credits.
We also need to fix the power issue if it's hurting production (it usually doesn't stop production in RA unless
      it's critical, but -10 might be fine).

The e1s moving to (110,20) are likely dead or the game state is weird.
Let's just observe for a bit to let the harvester finish.

Wait, the briefing says "Units: ?". This is strange. Maybe they are hidden or I should just ignore them.
Actually, looking at the previous turn, they were moving to (110, 20).
Now "Units: ?".
Maybe they are just not listed?
Let's assume we have no units for now.

We need to scout. We have no units.
We should wait for the harvester.
The production queue has "harv" at the top.
Let's observe.

One thing to note: The prompt says "Idle: 1004, 1006, ..." in the example, but here it says "Units: ?".
This might mean no units are currently on the map or controlled by us.
The e1s are likely dead.

Let's just wait for the harvester.
Also, we have no ore. The harvester needs to find ore.
We need to expand or find ore fields.
Usually, ore is near the base or at specific locations.
We should wait for the harvester to complete.

Let's observe.
```

Tool calls

```
build({"item": "harv"})
```

Commands recorded by the environment

```
Command::Build { item: "harv" }
```

TURN 42

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3693, ~147s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
The harvester is being built. I need to wait for it to finish and then assign it to an ore patch. I don't see
      any ore patches in the explored area yet, but I should keep building harvesters and then look for ore.
I also need to build a dome to reveal the map and get power.
```

```
I have no units currently visible, so I need to wait for the harvester to be built.
I will continue to build harvesters and wait.
```

Tool calls

```
build({"item": "harv"})
```

Commands recorded by the environment

```
Command::Build { item: "harv" }
```

TURN 43

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3783, ~151s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
The game state shows:
- Funds: $0
- Power: -10
- Harvesters: 0
- Explored: 25.4%
- Buildings: fact, proc, powr, tent, weap, fix
- Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, powr, powr, dome, powr, powr
- No units visible.
- Minimap shows a large green area (grass) around the base, but most of the map is dark (unexplored).

Wait, the production queue has many items. I need to wait for units/buildings to finish.
The briefing says "Units: ?" which usually means no units are currently on the map or they haven't spawned yet.
I should wait for the production queue to complete.
Let's observe to advance the game.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 44

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3873, ~154s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I'm in a very difficult situation. I have $0, no units, and my base is not producing anything useful yet. My
      production queue is filled with harvesters and power plants, but I have no ore to harvest and no power (
      power: -10).

Looking at the minimap, I can see a large green area to the east (around x=20-60, y=15-30) which is explored but
       not visible - this is the "fog of war" area I was trying to scout with my infantry. The rest of the map
      is completely dark/unexplored.

Key issues:
1. No power (-10) - I need to build more power plants
2. No ore/harvesters - I'm queuing harvesters but they won't be built until I have funds
3. No units - my infantry moved off-map and I lost contact
4. Production queue is stuck - I need to get production going
```

```
My production queue shows: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, powr, powr, dome, powr, powr

But I have $0 and no ore. The game won't build anything until I have funds.

I need to:
1. Wait for my existing power plants to generate power (but I'm at -10, so they might not be working)
2. Get harvesters built so they can gather ore
3. Find ore patches on the map

Looking at the production queue, I have multiple harvesters queued. Once they're built, they can start
      harvesting ore if there's an ore patch nearby.

But first, I need to understand why I have no funds. I should have some starting cash. Let me check if there's
      an ore patch near my base at (11,21).

Actually, looking at the briefing again - "Funds: $0 (cash=$0 + ore=$0)". This suggests I have no ore deposits
      to harvest either.

Wait, I need to look at the map more carefully. The green area around (11,21) to (60,20) is explored but not
      visible. There might be ore patches there that I haven't discovered yet.

But more importantly, I need to get my economy going. My production queue has harvesters, but they can't be
      built with $0.

Let me think about what I can do:
1. I could try to build something cheaper, but I have $0
2. I could wait for the production queue to complete, but it won't progress without funds
3. I need to find ore

Actually, looking at the minimap again - there's a large green area from roughly x=15 to x=60, y=15 to y=30.
      This is explored terrain. There might be ore patches in this area that I haven't discovered yet because I
      don't have visibility.

But my infantry are gone - they moved off-map. I have no units to scout or harvest.

This is a critical situation. I need to:
1. Get funds somehow (maybe the game will give me starting cash, or I need to wait for production to complete)
2. Find ore patches
3. Get harvesters to the ore patches

Let me check if my production queue will complete. I have harvesters queued, but they cost $1100 each. With $0,
      they won't build.

Wait - maybe the game gives me starting funds? Let me re-read the briefing. "Funds: $0 (cash=$0 + ore=$0)". This
       is current funds.

I think the issue is that I need to wait for my production queue to complete, but it can't complete without
      funds. This is a chicken-and-egg problem.

Actually, looking at the production queue more carefully:
- harv (harvester) - $1100
- 1tnk (light tank) - $700
- 1tnk - $700
- 1tnk - $700
- harv - $1100
- harv - $1100
- harv - $1100
- harv - $1100
- powr (power plant) - $300
- powr - $300
- dome (radar dome) - $1500
- powr - $300
- powr - $300

Total cost: 1100*5 + 700*3 + 300*4 + 1500 = 5500 + 2100 + 1200 +
```

Tool calls

```
(no tool call)
```

Commands recorded by the environment

```
Command::Observe
```

TURN 45

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 3963, ~158s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I need to check if any harvesters have been produced and are available to be assigned. The briefing shows no
      units listed, which is odd given the production queue. Let me check if there are any idle units or if the
      harvesters are still in production. The production queue shows multiple harvesters being built. I should
      wait for them to complete and then assign them to ore. But first, I need to confirm if any units are
      actually present. Since the briefing says "Units: ?" and no specific units are listed, it's possible the
      harvesters are still being built. I should continue building and then once harvesters are available, I
      need to place them or assign them. However, the `build` function only queues production. The actual
      placement or assignment might happen automatically or I might need to use another function. Looking at the
       functions, I don't see a direct way to assign harvesters to ore. In Red Alert, harvesters typically auto-
      assign to the nearest ore patch once built. But since I have no units listed, perhaps the harvesters haven
      't been built yet. I should continue building until I see units, then check their status. Alternatively, I
       might need to build other things. But the priority is to get harvesters online to generate income. Let me
       try building another harvester to see if it completes, or perhaps I should check the current state again.
       Since I just called observe and the state hasn't changed much, I'll continue building harvesters. However,
       I notice that the production queue is very long and I'm not seeing any units. This might be because the
      game is paused or there's an issue. But according to the rules, I must issue at least one tool call per
      turn. I'll continue building harvesters until I see units appear. Alternatively, I could try building
      something else, but harvesters are critical for economy. Let me build another harvester.
```

Tool calls

```
build({"item": "harv"})
```

Commands recorded by the environment

```
Command::Build { item: "harv" }
```

TURN 46

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4053, ~162s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I need to build a harvester to start collecting ore, which is the only way to get funds since I'm at $0. The
      production queue shows multiple harvesters queued. Let me build one to get it out immediately.
```

Tool calls

```
build({"item": "harv"})
```

Commands recorded by the environment

```
Command::Build { item: "harv" }
```

TURN 47

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4143, ~165s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
The briefing shows I have no units and no funds. I've been queuing harvesters but they haven't been produced yet
      or haven't appeared in the unit list. Looking at the minimap, I can see a large green area to the east/
      southeast of my base (around x=60-80, y=20-30). This could be an ore field.

Since I have no units and no funds, I need to wait for harvesters to complete production. But I've already
      queued many harvesters. Let me check if any have been produced by observing.

Actually, looking at the production queue, I see multiple "harv" entries. Harvesters cost $1100 each. I have $0
      funds, so these can't be producing yet unless there's ore income I'm not seeing. Wait, the briefing says "
      cash=$0 + ore=$0" so there's no ore income either.

This is strange - I'm queuing harvesters but have no funds to build them. The production queue might be stuck
      until I get funds.

Let me observe to see if anything changes, or if harvesters complete.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 48

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4233, ~169s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
The harvester queue is building up (7 harvesters queued), but I have no cash and my infantry scouts went missing
      (likely fell into the fog of war or were destroyed, but since I see no enemies, they probably just
      wandered off or the game state is lagging). I need to wait for the harvesters to complete to get income.

Let's look at the production queue again:
harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr

That's a lot of harvesters. I should probably stop building them and start building something useful like a dome
      or a tank once I have cash. But I can't cancel the queue easily in this interface (no cancel command
      listed). I can only build new things.

Wait, I see "Idle: ..." is missing from the briefing. This means all my units are either in production or moving.
      The infantry are gone.

I should just wait for the harvesters to finish and bring ore to the refinery. Then I can build a dome and some
      tanks.

Let's observe to see if the harvesters are ready.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 49

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4323, ~172s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
```

```
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
The harvester production queue seems stuck or very slow, and I'm not seeing any units being produced despite the
       queue showing multiple harvesters. My funds are $0, so I can't build anything new. I need to check if
      harvesters are actually being completed and placed. The briefing shows "Units: ?" which is odd - it might
      mean no units are currently listed or tracked.

Looking at the minimap, I can see a large green area (grass) to the east of my base, but it's unexplored (dark).
       I need to get units out to scout and find ore.

Since I have no units and $0 funds, I'm in a difficult position. The production queue shows harvesters but they
      aren't appearing. This might be a timing issue where they take a while to build.

Let me try to observe and see if anything changes, or if harvesters appear. I should also consider that my power
       is negative (-10), which might be affecting production. I have powr(1003)@(8,18) listed, but maybe I need
       more power plants.

Wait, looking at the production queue again: "harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr,
      powr, dome, powr, powr"

This is a long queue. The harvesters should be building. Let me observe a few more turns to see if they complete.


Actually, I notice the units list is "?" which might indicate an error or that no units exist. But harvesters
      should appear as units once built.

Let me try to observe and wait for harvesters to complete.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 50

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4413, ~176s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I'm stuck at 0 funds and the production queue is full of harvesters that I can't afford to build. The briefing
      shows "Funds: $0" but the production queue keeps listing "harv" items. This suggests the game might be
      queuing them but not actually building them due to lack of funds, or there's a misunderstanding about how
      the queue works with $0.

Looking at the minimap, I can see a strip of explored grass around y=18-28, x=10-60. My base is at (10,20). The
      production queue shows multiple harvesters queued, but I have $0. I need to get funds to build these
      harvesters.

Wait - I have a proc (ore refinery) at (10,24). If there's an ore field nearby, I need a harvester to collect it.
       But I can't build harvesters with $0. This seems like a catch-22.

Let me think: maybe the game auto-builds harvesters from the queue even with $0? Or maybe I need to wait for the
       harvester production to complete and then it will start working? The queue shows "harv" multiple times,
      so perhaps they're already in production and will complete soon.

Actually, looking more carefully at the production list: "harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv,
      harv, powr, powr, dome, powr, powr" - there are 7 harvesters queued. These should start building once
      funds are available. But I have $0.

Wait - maybe the issue is that I need ore to build harvesters? No, harvesters cost $1100 and don't require ore.
      But I have $0 cash.
```

```
Let me re-read the briefing: "Funds: $0 (cash=$0 + ore=$0)". So I have no cash and no ore. The harvesters in the
      queue can't be built without funds.

Actually, I think the production queue might be showing what's queued, but the actual building requires funds.
     With $0, nothing new can be built. This is a problem.

But wait - maybe the game is designed such that harvesters will auto-build from the queue when funds become
     available? And funds become available when I have ore? But I have no ore and no harvester to collect ore.

This seems like a deadlock. Let me check if there's anything I can do:
1. I have no units to scout for ore fields
2. I have no funds to build harvesters
3. The production queue has harvesters but they can't be built with $0

Hmm, maybe the production queue items will eventually complete? Let me observe and see if anything changes.

Actually, I notice the production queue keeps growing with harvesters. Maybe the game is automatically queuing
     more harvesters? This is strange.

Let me just observe and see what happens. Perhaps the first harvester in the queue will complete and start
     working, generating ore, which will then allow me to build more units.

But wait - I need to check if harvesters are actually being built. The briefing doesn't show any harvester units,
      so they might still be in production.

Let me observe for a few turns to see if a harvester completes.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 51

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4503, ~180s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I've been queuing harvesters but they're not showing up in the units list. This is likely because they are being
      produced but haven't completed yet, or they're already out on the map and I need to find them. However,
     the briefing says "Harvesters: 0" which confirms I have none active.

Looking at the production queue: `harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome,
     powr, powr`. That's a LOT of harvesters queued. But I have 0 funds. The production is happening but I can'
     t see the units.

Wait, looking at the map image, I can see a strip of grass around y=15-30, x=10-70. My base is at (11,21). There
     's no ore visible on the map yet (no yellow dots). I need to find ore to get credits.

The problem is I have 0 cash and can't see any ore. My harvesters are queued but I don't have any active to
     collect ore. I need to wait for the first harvester to complete, then find ore.

Actually, looking more carefully at the production queue - it seems like the game is auto-producing these units.
      The "Production:" line shows what's in the queue. I have many harvesters queued. Once one completes, it
     should appear in the units list.

Let me wait a few more turns for the first harvester to complete, then I can issue orders to it.

But I also notice I have power issues (-10 power). I have powr buildings in production too. I should prioritize
     power to avoid production slowdowns.

Let me observe a few more turns to see if any harvesters appear.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 52

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4593, ~183s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I'm still at $0 and have no harvesters to collect ore. The production queue shows multiple harvesters being
      built, but they haven't appeared as units yet. I need to wait for the harvesters to complete and then
      assign them to the ore field. Looking at the minimap, there's a green area around (20,20) to (60,25) which
       likely contains the ore field. I should continue observing until the harvesters are ready.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 53

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4683, ~187s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
The briefing indicates I'm out of cash ($0) and have no harvesters yet (Harvesters: 0).
Looking at the minimap, there's a large ore field (greenish area) right next to my base (around (20,20) to
      (60,25)).
My base is at (10,20).
I need to build harvesters to gather ore and get credits.
The production queue has multiple `harv` entries.
I should check if any harvesters have been produced and are idle, or if I need to wait for them to finish.
Since the briefing says "Harvesters: 0", none are currently on the map or ready.
The production queue shows `harv` listed multiple times.
I should continue building harvesters until I have some, then assign them to the ore field.
Wait, I've been spamming `build harv` and `observe`. The production queue shows `harv, harv, ...`.
I need to be more efficient. I should probably build a few more harvesters and then wait for them to complete.
Actually, looking at the queue: `harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome,
      powr, powr`.
There are already 7 `harv` in the queue (plus the ones I just added? No, the briefing updates).
Let's assume the previous `build harv` commands added to the queue.
I need to wait for the first harvester to finish.
However, I have no units on the map.
I should probably build a jeep for scouting as well, but money is 0.
Let's just observe and wait for the first harvester to complete.
Actually, I can build more harvesters to queue them up.
But I've already queued a lot.
Let's just observe for a few turns to let the first harvester finish.
Once a harvester is built, I'll need to move it to the ore field (around 30,22) and then send a refinery (proc)
      to pick up ore if not already built.
I have a `proc` building at (10,24). That's good.
So the plan is:
1. Wait for harvester to finish.
2. Move harvester to ore field.
3. Wait for ore to be loaded.
```

```
4. Send harvester to refinery.
5. Repeat.
6. Use credits to build units.

I've been queuing harvesters. I should stop queuing for a moment and let them build.
I will observe for a few turns.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 54

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4773, ~190s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I see the minimap now. There's a large ore field stretching from roughly x=10 to x=60, y=15 to y=30. My base is
      at (10-14, 18-24). I need to send harvesters to this ore field. Let me check if any harvesters have been
      produced yet.

Looking at the production queue: "harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome,
      powr, powr"

I've been building harvesters but they might not have completed yet. Let me wait a bit more and then check for
      units. Actually, I should keep building harvesters and also start producing some military units once I
      have funds.

But first, I need to get harvesters to the ore field. Let me observe to see if any harvesters have been produced.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 55

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4863, ~194s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
The game has progressed significantly (tick 4863), but I still have $0 and no harvesters visible in the "Units"
      list. The production queue shows multiple harvesters and tanks, but they haven't been produced or deployed
       yet, or I'm not seeing them. The minimap shows a large explored area (green) to the left, but my base is
      at (11,21) which is at the edge of the explored area. The production queue has "harv, 1tnk, 1tnk, 1tnk,
      harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr". This is a huge queue.

The problem is I have $0. I need ore to produce units. The harvesters in the queue should produce ore, but they
      aren't appearing as units yet. Wait, looking at the briefing, "Harvesters: 0". This means no harvesters
      are currently active on the map. The queue contains them, but they haven't finished building.
```

```
I need to wait for the harvesters to finish building and then assign them to the ore silo/refinery. But first, I
      need to see if there's an ore silo. The buildings list shows: fact, proc, powr, tent, weap, fix. No silo.
      I should build a silo to store ore, but I have no funds.

Actually, the harvester production is queued. Once a harvester finishes, it will be an idle unit. I need to wait
      for it to finish. Since I have no actions to take with $0 and no units, I should just observe and wait
     for the first harvester to complete.

Wait, looking at the queue again: "harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome,
      powr, powr". This is a very long queue. The game might be stuck or waiting for something. Or maybe the
     first harvester is still building.

I will observe to let the game progress and the first harvester to finish.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 56

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 4953, ~198s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I've been waiting for production to complete. The briefing shows I have no funds ($0) and no harvesters yet. I
     have a production queue with multiple harvesters, 1tnks, and power plants. I need to wait for the first
     harvester to complete so I can start gathering ore and generating credits. Let me observe again to see if
     anything has changed.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 57

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 5043, ~201s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
     (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
I see the minimap now. The visible area is a horizontal strip around rows 15-30, columns 10-60. My base is at
     (11,21) which is in this area. I still have $0 and no units visible. The production queue still shows many
      harvesters and power plants being built, plus 3 light tanks and a dome. I need to wait for units to
     complete and then start harvesting ore to get funds. Let me continue observing until units appear.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 58

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 5133, ~205s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
We are at tick 5133. The briefing says:
- Funds: $0
- Power: -10
- Harvesters: 0
- Explored: 25.4%
- Buildings: fact(1001)@(10,20), proc(1002)@(10,24), powr(1003)@(8,18), tent(1004)@(14,18), weap(1005)@(14,22),
      fix(1006)@(12,24)
- Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr

Wait, the production queue is very long and mostly harvesters and power plants. We have no units currently on
      the map.
We need to produce units to fight. The objective is to destroy 8 enemy units.
Since we have $0 and no harvesters, we are essentially stuck unless the harvesters in the production queue
      complete and we can assign them to ore.
But wait, the production queue shows `harv` at the front. Once a harvester is built, we need to move it to an
      ore patch.
Also, we have a power deficit (-10). We are building `powr` (power plants) to fix this.
The production queue is huge. We need to wait for the first harvester to finish, then assign it to ore.
Let's check the minimap. It shows a strip of land around y=20. The base is at (10,20).
There are no visible enemies.
Since we have no units and $0, we can only observe or build. We are already building.
I will observe to let the production queue advance. Once a harvester is built, we can move it.
Actually, let's look at the production queue again: `harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv,
       powr, powr, dome, powr, powr`.
The first item is `harv`. It will finish soon.
I will observe.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

TURN 59

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 5223, ~208s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
```

Model response

```
The briefing shows we are at tick 5223, funds $0, and we have a production queue full of harvesters and power
      plants. I've been just observing for a long time while the harvesters produce. The harvesters should be
      coming online now or soon. I need to wait for them to complete and then deploy them to ore patches.

Looking at the minimap, there is a large green area (grass) around coordinates (10,20) to (60,30). This is
      likely where the ore field is or where the enemy is. Since we have no units and no visible enemies, we
      need to get our harvesters out to start generating income.

Let's check if the harvesters have completed. The production queue shows `harv, harv, harv, harv, harv, harv`.
      These are all harvesters. I need to wait for one to finish, then build it and assign it to an ore patch.

Actually, looking at the queue: `harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome,
      powr, powr`.
```

```
Wait, the queue seems to have many harvesters. I should build the next harv if it's not done, or if a harv is
      done, I should place it.

Let's try to build a harv. If one is already done, it will queue the next. If none are done, it adds to the
      queue.
Actually, I should just observe until a harv is ready to be placed. But I must issue at least one tool call per
      turn.
I will try to build a harv. If it's already in production, this might fail or queue another.
Better yet, I'll just observe and wait for the first harv to finish. But I need to make progress.
Let's build a harv.
```

Tool calls

```
build({"item": "harv"})
```

Commands recorded by the environment

```
Command::Build { item: "harv" }
```

TURN 60

Briefing

```
FACTION: You are allies. Your opponent is soviet.
--- TURN BRIEFING (tick 5313, ~212s game time) ---
Funds: $0 (cash=$0 + ore=$0) | Power: -10 | Harvesters: 0 | Explored: 25.4%
Base center: (11,21)
Units: ?
Buildings: fact(1001)@(10,20) proc(1002)@(10,24) powr(1003)@(8,18) tent(1004)@(14,18)[infantry] weap(1005)@
      (14,22)[vehicle] fix(1006)@(12,24)
Enemies: none visible
Production: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
[minimap image attached; shown in the figure below]
```

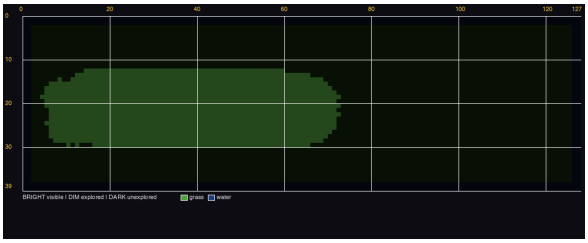


Figure 6: Minimap image attached to the turn-60 briefing of Run C, as stored in the transcript.

Model response

```
The briefing shows:
- Funds: $0
- Power: -10
- Harvesters: 0
- Explored: 25.4%
- Base center: (11,21)
- Buildings: fact(1001)@(10,20), proc(1002)@(10,24), powr(1003)@(8,18), tent(1004)@(14,18), weap(1005)@(14,22),
      fix(1006)@(12,24)
- Production queue: harv, 1tnk, 1tnk, 1tnk, harv, harv, harv, harv, harv, harv, powr, powr, dome, powr, powr
- No enemies visible.

I need to build harvesters to generate income since I have $0. The production queue has multiple harvesters
      queued. I should wait for them to finish or build more if needed. Since I have a power deficit (-10), I
      should also consider building power plants, but the queue already has several powr units.

The minimap shows a large green area (grass) with some explored regions. The base is at (10,20) roughly. I need
      to find ore deposits to send harvesters to.

Since I have no units and no income, I should focus on getting harvesters out to find ore. The production queue
      shows many harvesters, so I'll wait for one to complete.

Let me check if there are any idle units or if I need to do anything else. The briefing says "Units: ?" which
      might mean no units are currently visible or controlled.

I should probably just wait for the harvesters to finish producing. Since I can't do much without units or
      income, observing or waiting might be the best course.

However, I notice the production queue has many harvesters. Let me try to build another harv or just observe to
      see if units complete.
```

```
Actually, looking at the queue, it seems like a lot of harvesters are queued. I should let them finish. But I
      also need to find ore. Without harvesters on the map, I can't do anything.

Let me just observe for now to see if any units complete.
```

Tool calls

```
observe({})
```

Commands recorded by the environment

```
Command::Observe
```

OUTCOME

The run ends at tick 5,403 against a deadline of 5,400 with 3 of 8 required kills.

S LIMITATIONS

Engine scope. The simulator reimplements selected Red Alert mechanics using OpenRA rules and data. Evaluation on this task suite does not establish full behavioral equivalence with either game engine or comprehensive coverage of their mechanics. The task inventory and inspected configurations delimit the tested behaviors.

Competitive coverage and provenance. Adversarial records cover one arena map, unequal opponent and seed schedules, and historical endpoint and configuration differences (Appendix E). The map is symmetric and the east-side advantage holds in both engine player slots, but the records do not isolate which aspect of the agents' perspective produces it. Exchanging positions balances assignments but does not establish equal conditions or eliminate every bias. Missing tool calls in a transcript can expose an execution failure without identifying whether its cause lies in the model, provider, or adapter.

Environment and transfer. The studies use one game environment, and competitive records use one canonical arena map with historical configuration differences. Generalization across games, rendering styles, and robotic tasks remains untested. Task metadata may identify motivating analogues, but these annotations are not transfer evidence.

Non-LLM agent coverage. Jev was evaluated on 11 tasks, one per family, with one attempt per difficulty and seed and text-only input; per-turn transcripts were not archived. Its results describe this controller and configuration, and its comparison with the LLMs is limited to the 33 shared task–difficulty combinations and 11 matched text-only combinations (Appendix N).

Human baseline size and skill. Our human baseline (246 runs across difficulty levels, multiple testers) is a selected sample, small relative to the LLM evaluations, and does not constitute a calibrated population ceiling.

Coverage asymmetry. GPT-5.4, GPT-5.4-mini, and Qwen 3.5-9B lack complete three-attempt coverage in the selected historical cohort. For the two GPT models this reflects API cost: available credits covered one full sweep per model. Qwen 3.5-9B's later attempts are incomplete in the retained exports (Appendix J). The repeated-attempt analysis therefore uses the five eligible models and their matched recorded attempts (Appendix B).

Handoff feasibility. The idle-handoff comparison changes the state and remaining time together, and some tasks end before the model takes control. The saved journals do not establish that all delayed states remain solvable or retain the checkpoints needed for a reference continuation. A recovery study would need to save identical takeover checkpoints for every tested agent and verify that a reference policy can complete each task from those checkpoints within the remaining budget. To test the value of handoff history, it would then vary the transcript while holding the checkpoint and time fixed.

Prior knowledge and contamination. The tasks were written for this benchmark, so they are unlikely to appear in the evaluated models' training data. Models may know Red Alert or real-time strategy in general; every model receives the same rules, units, and controls in its system prompt, and general strategic knowledge is part of what the tasks measure.

Compute and dual use. The evaluation used commercial and hosted APIs. Token counters stored with the exports are incomplete for runs resumed after interruption, so we do not report a total cost; the largest recorded counter is 431 million prompt tokens for Sonnet 4.6's three-attempt sweep. The environment is a fictional game with no real-world hazardous content; skills it rewards, such as planning and adversarial play, are general-purpose.

Historical model versions. Results describe the recorded model endpoints and agent configurations. Provider and version differences, incomplete historical metadata, and subsequent updates limit interpretation as a comparison of immutable model weights (Appendix J).