WHAT SHOULD LLMS SEE, AND HOW OFTEN SHOULD THEY PLAN? WORLD-STATE ENCODINGS FOR REAL-TIME GAMES

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-time game planning with large language models (LLMs) raises two ordered questions: how often should the planner decide, and how should the world state be encoded? We study *training-free high-level planning* in a one-versus-four asymmetric pursuit game, where a frozen LLM selects tactical tasks and targets while a fixed, pretrained RL controller executes frame-level actions. First, an interval sweep with paired output-token budgets reveals an inverted-U trend in chaser win rate under both planners: performance peaks at 4 s and declines at shorter and longer tested intervals. Shorter intervals pair with small output budgets, yielding shallow thinking and suboptimal plans, while longer ones deliver plans to a board that has moved on. Second, at this fixed interval, we compare plain text, topology, graph-text trees, and a visual minimap across seven maps and two planner models, holding the shared decision context and controller fixed. We propose *chessboard*, a local text grid whose layout mirrors map geometry and whose annotations associate entities with their states, and a JSON variant carrying identical information. Both *chessboard* variants improve the chaser win rate over the no-encoding baseline on every evaluated map under both models. The JSON variant leads on all aggregate gameplay metrics, reaching mean win rates of 62.8% and 56.8%, respectively 7.8 and 7.2 percentage points above the corresponding baselines. These findings highlight planning interval and state organization as complementary design choices for real-time LLM agents: neither more frequent replanning nor additional structure or spatial detail alone guarantees better performance.

1 INTRODUCTION

A real-time game punishes both slow and shallow planning. An LLM decision is useful only if it lands in time and still selects the right plan, and the two requirements pull in opposite directions: more thinking buys a better plan at the cost of latency, while less thinking saves latency but yields a shallow, easily wrong plan. Consider a one-versus-four asymmetric pursuit game in which an LLM plans for a chaser tasked with stopping four survivors from repairing machines and escaping through gates. Figure 1 left shows the failure of thinking too little: the planner chases the nearest survivor at low latency and misses the cluster with greater tactical value—fast, but wrong. Figure 1 right shows the failure of thinking too much: a deep multi-step plan predicts rotations, cuts paths, and guards gates, yet lands after the target has already retreated through the gate, and the decision fails. Figure 1 middle shows the case this paper studies: an appropriately scoped plan—a single chase that both catches a survivor and cuts off the repair in progress at low latency. Such a real-time game therefore calls for planning with low latency and appropriate thinking, not the deepest thinking.

We study *training-free high-level planning* (Lin et al., 2023; Wang et al., 2024) for the chaser in this game. The planner is not asked to play the game frame by frame (Figure 2): every few seconds, a frozen LLM selects tactical tasks and targets based on the current game state, and this selection is passed to a fixed low-level RL controller (Hösch et al., 2026) as its input; the controller then outputs the frame-level movement, attack, and interaction actions. We do not compare frontier planner models, which pay high latency and cost, and we do not fine-tune a model for this game, which requires game-specific training. The question is how to present the game state so that a frozen LLM's planning ability is fully used.

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.

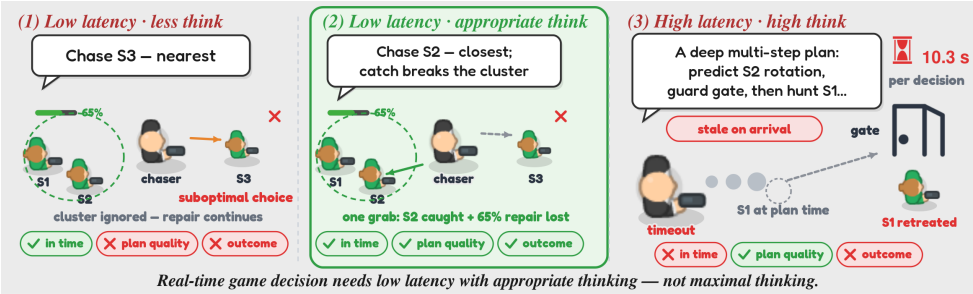

Figure 1: Latency versus thinking depth in one tactical decision. *(1) Left*: too little thinking chases the nearest target and misses the cluster with greater tactical value. *(2) Middle*: an appropriately scoped plan chases the closest cluster member, catching a survivor and cutting off the repair in progress. *(3) Right*: a deep multi-step plan lands after the target has retreated through the gate.

These two requirements—landing in time and selecting the right plan—map to two ordered design questions: what planning interval, with its paired output-token budget, is defensible in this system, and, under a fixed interval, what world-state encoding helps the planner act? Both are better answered in the live game, where decisions face delayed execution, opponent responses, and a changing state; an offline test that lacks these conditions can misjudge how well an encoding helps.

The interval question comes first: encodings are compared at one fixed interval, so the interval has to be chosen before that comparison. The encoding question then asks how the map should be shown to the planner (Fatemi et al., 2024; Martorell, 2025; Xypolopoulos et al., 2024; Zhao et al., 2023; Wong et al., 2026). The comparison holds the shared prompt and the controller fixed, varying only the world-state encoding—the controlled variable of this study. Within the interval, the planner can output only a few hundred tokens, so a clearly organized rendering lets it pick the right task and target in a few tokens, while a poorly organized one wastes the budget reconstructing spatial layout from text (Sun et al., 2025; Hu et al., 2025). Our contributions are:

- **A calibrated planning interval**: a controlled interval sweep reveals an inverted-U in outcome peaking at 4 s, with measured mechanisms at both ends—on the short side, the 2 s interval pairs with a small output budget, so each decision thinks too shallowly to select a good plan; on the long side, each plan acts on a state that has already gone stale (§4.2).
- **A controlled, training-free comparison of world-state encodings**: we propose *chessboard*, a text grid whose layout mirrors the game map, with a JSON variant carrying the same information. Under one explicit setup, the same high-level planner, and the calibrated interval, we compare both and prior-work encodings—plain text, topology, graph-text tree, and visual minimap—against the no-encoding baseline (the fixed prompt without the world-state encoding) across multiple maps (§3.2, §4.3).
- **Chessboard wins, and structure explains why**: chessboard raises the chaser's win rate on every evaluated map, while the other encodings show no stable improvement. Component ablations across three encoding families show that removing one component at a time hurts in nearly every comparison (§4.3, §4.3.2).

2 Related Work

Training-free control in interactive environments. Hierarchical LLM agents and planning agents act in interactive environments without task-specific training, mostly at coarse time scales (Lin et al., 2023; Li et al., 2026; Zhang et al., 2025; Han et al., 2025; Wang et al., 2024). Closest to our setup, Hösch et al. (2026) pair an LLM planner with RL executors in competitive multi-agent games; we share this division of labor but treat the planning interval and the world-state encoding as measured design variables rather than fixed system choices.

World-state encodings for LLMs. Graph, spatial, and structured input formats change LLM behavior on static tasks (Fatemi et al., 2024; Xu et al., 2026; Wang et al., 2026; Zhao et al., 2023;

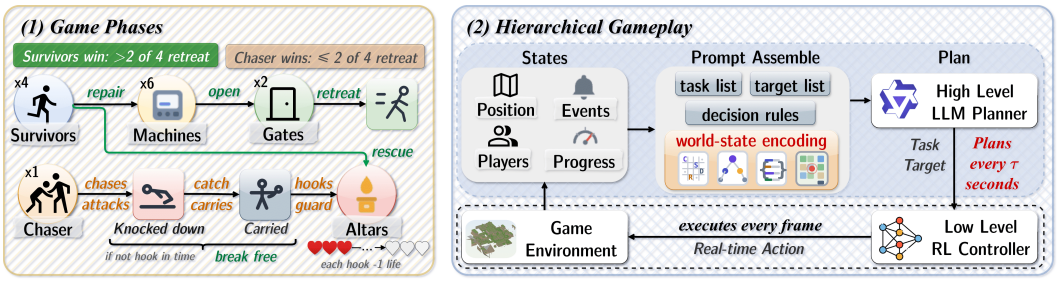

Figure 2: System overview. *(Left)* Game phases: survivors repair machines, open the retreat gate, and retreat ($> 2/4$ survivors win); the chaser knocks survivors down, carries them, and hooks them onto altars. *(Right)* Hierarchical gameplay: the engine state is assembled into a prompt with the ***world-state encoding*** as the controlled variable (§3.2); a frozen LLM planner selects a task and target every τ-second planning ***interval***, conditioning a fixed low-level RL controller.

Martorell, 2025), and presentation—binding, compactness, spatial layout—matters as much as content in state-design work (Zhang et al., 2026; Liu et al., 2024; Yamada et al., 2024; Sun et al., 2025; Gao et al., 2024) and in game benchmarks (Paglieri et al., 2025; Hu et al., 2025). These studies vary the input on a fixed task; we hold a fixed closed-loop planner and ask the converse: which encoding lets the planner pick better tasks and targets in the live game, and where each fails. We also include a visual minimap, connecting to text-versus-image state studies (Wong et al., 2026; Goodyear et al., 2025); our minimap result is limited to a frozen planner, not per-frame visual control.

3 Planning Interval and World-State Encoding

3.1 From Hierarchical Control to the Planning Interval

The testbed is a one-versus-four asymmetric pursuit game: the LLM-controlled *chaser* faces four *survivors* controlled by RL bots on a map approximately 300 m across, with altars, machines, and gates (Figure 2). Survivors progress through three phases—repairing machines, opening the retreat gate, and retreating—and win if more than two of the four retreat; the chaser stops them by hooking survivors onto altars to drain their three lives (full rules in Appendix B). The adversarial state keeps changing: at each frame, the state s evolves under the chaser's action and the four survivors' behavior.

Hierarchical control. Control is decomposed hierarchically (Figure 2, right): every τ seconds, the LLM planner reads a prompt containing a world-state encoding $e(s)$ of the current state and outputs a tactical decision (t, z)—a task t and a target z. Planner and controller run at different time scales: the planner decides once per interval, while the fixed reinforcement-learning (RL) controller executes every frame. The game engine resolves the chosen target z into a goal location and passes the controller the task t, this location, and the distance to the target. The controller then outputs the frame-level movement, attack, and interaction actions until the next decision, so the LLM specifies a persistent tactical intent rather than frame-level actions. Both layers serve one objective: maximizing the chaser win rate. This study varies **the decision interval** τ and **the world-state encoding** e.

Planning interval and staleness. The interval τ is a design choice of the system, and it is bound to the planner's output-token cap: each interval is paired with an output-token limit that bounds how deeply the planner may think. Because each decision is executed over the whole interval that follows it, a world-state encoding helps the chaser only if the plans derived from it still work when executed a few seconds later: the planner takes time to respond, the game keeps changing while it thinks, and a decision made for an earlier state may land after the game has already moved on. We define *state staleness* as this mismatch at the moment a decision lands: the more the live game state has drifted from the state the decision was made for, the staler the decision. The interval is calibrated first: §4.2 sweeps τ and reports the chaser win rate and staleness at each level, and the world-state encoding comparisons then hold the interval fixed.

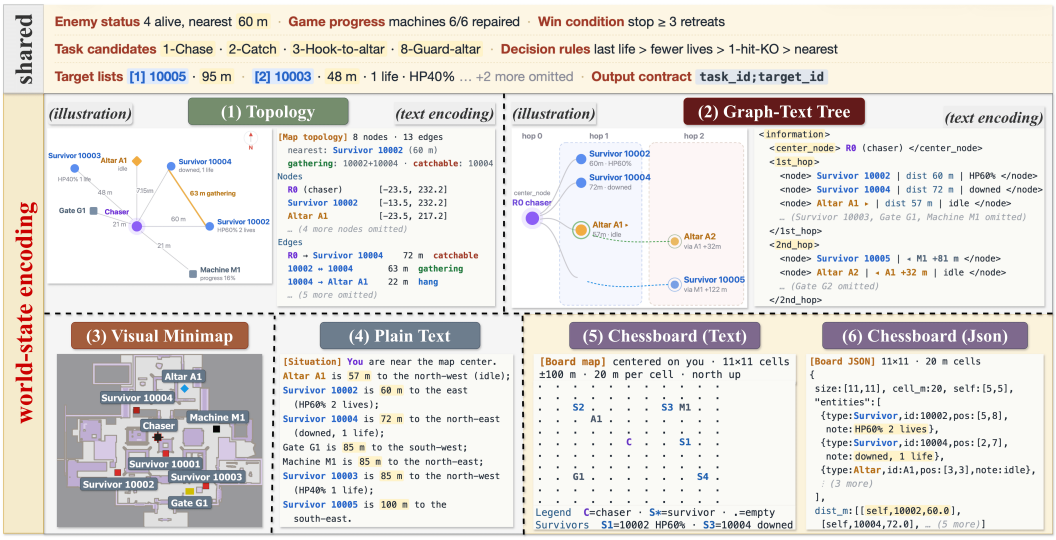

Figure 3: World-state encoding formats. The shared top block is held fixed across conditions; only the world-state encoding below it varies. Topology and graph-text tree (top row); visual minimap, plain text, and our chessboard and chessboard JSON (bottom row).

3.2 World-State Encodings

For the LLM planner's decision, the task t comes from a task menu $\mathcal{T}(s)$ of six tactical types—chase, catch, hook-to-altar, destroy-machine, guard-gate, guard-altar—and the target z from that task's candidate-target list $\mathcal{Z}(s,t)$ (survivors, altars, machines, or gates, as appropriate). The output contract is a valid *task_id;target_id* pair from these current lists (Appendix C).

What is controlled. The one controlled part of the prompt is the world-state encoding e: the description of where entities are and how they relate on the map (the lower part of Figure 3). The rest of the prompt—the decision interface above, plus game progress, and decision principles—is held fixed across conditions (the top part of Figure 3). The planner must answer within the interval, so its thinking is capped by a small output-token budget (§3.1). Conditions differ along the five dimensions of Table 1: whether the encoding is *structured* (a list, a tree, or a grid) or prose; whether that structure is *hierarchical* (nested levels around the chaser) or flat; whether the view is a *local window* around the chaser or the whole map; whether the encoding describes each entity's position *spatially*—direction and distance relative to the chaser—beyond the bare target-list distances that every condition receives; and whether complex map coordinates are converted into *simplified coordinates*—simple grid indices that are easy to read directly.

The compared encodings. Figure 3 renders the same state in every format. *No encoding* is the baseline: the planner receives only the shared fixed block of Figure 3, without a world-state encoding. *Topology* (panel 1) lists landmarks and entities across the whole map, with the path distances between them, in the node-and-edge list format of graph-linearization studies (Xypolopoulos et al., 2024); unreachable pairs carry no edge, as in Zhang et al. (2026). *Graph-text tree* (panel 2) expands spatial relations hierarchically—entities as nodes, path distances in meters as edge labels—following the syntax-tree serialization of Zhao et al. (2023); the tree extends two hops from the chaser, a local neighborhood rather than a full-map view. *Visual minimap* (panel 3) renders the map as an image instead of text: a different input modality under the same planning setup, as in text-versus-image state studies (Wong et al., 2026; Goodyear et al., 2025). *Plain text* (panel 4) describes entities in prose—each as a direction, a distance in meters, and a rich state, ordered by distance—as in Martorell (2025).

The chessboard encoding. However, the above encodings each have clear drawbacks. First, the visual minimap shows the layout, but positions are pixels: distances cannot be read as values, the

Encoding	Structured	Hierarchical	Local window	Spatial desc.	Simpl. coords
no encoding	×	×	×	×	×
plain text	×	×	×	✓	×
topology	✓	×	×	✓	×
graph-text tree	✓	✓	✓	✓	×
visual minimap	×	×	×	×	×
chessboard (ours)	✓	×	✓	✓	✓
chessboard JSON (ours)	✓	×	✓	✓	✓

Table 1: Encoding contrast. *Structured*: explicit structure (list, tree, grid) rather than prose or pixels; *Hierarchical*: nested levels rather than a flat list; *Local window*: a window around the chaser rather than the whole map, keeping nearby decision-relevant entities in focus; *Spatial desc.*: each entity's position described relative to the chaser (direction and distance, or a grid position)—bare distance values in the fixed target lists are available to every condition and do not count; *Simpl. coords*: complex map coordinates converted into simple grid indices that are easy to read directly.

image must be grounded against the text-based menu, and pixel-to-text conversion spends the output budget. Second, plain text, topology, and the graph-text tree state positions as raw game coordinates and metric distances, which are laborious for an LLM to convert into relative positions and directions. We propose *chessboard* (panel 5) to address both drawbacks: it renders a 200 m window around the chaser as an 11×11 character grid indexed by map geometry, so the layout is shown directly in the grid and complex map coordinates become simple grid indices that are easy to read; legend and annotation rows attach entity identities and states to the grid cells, and entities outside the window are listed rather than dropped. The *chessboard JSON* variant (panel 6) carries the same information as structured key–value text: each entity becomes a record with explicit fields, a format LLMs parse natively, so attributes are read field by field instead of scanned out of the grid.

4 Results

4.1 Experimental Setup

4.1.1 Platform, Planner, and Conditions

Game agent platform. The game and its rules are specified in §3.1. Within a batch, all conditions share the same chaser–survivor lineup: the four survivors are controlled by a single strongest RL model; spawn configurations and game versions are identical; all conditions run on the same seven maps (anonymized as M1–M7, Figure 5); a game has no time limit and ends only when a terminal condition is met (Appendix B). Each condition is run with 5 seeds on every map, 800 games per seed—28,000 games per condition.

LLM base models. We validate the study on two LLM planners, Qwen3-Omni-30B-A3B and Qwen3.5-35B-A3B (Qwen Team, 2025), both served with one stack: vLLM (Kwon et al., 2023) deployment on identical GPU hardware across all conditions—performance comparable to mainstream accelerators—with no input-context limit, greedy decoding (temperature 0, top-k 1), and a per-decision output cap of 1024 tokens, set by the serving stack; the sweep (§4.2) instead adds a word limit to the prompt at each interval. The LLM plans at the tactical level: it selects a $(\textit{task}, \textit{target})$ pair that conditions the low-level reinforcement-learning controller, never frame-level actions.

Comparison conditions. The compared encodings adapt prior-work formats: *plain text* (Martorell, 2025), *topology* (Xypolopoulos et al., 2024; Zhang et al., 2026), *graph-text tree* (Zhao et al., 2023), and the *visual minimap* (Wong et al., 2026; Goodyear et al., 2025), defined in §3.2; *chessboard* and its JSON variant are ours. The *no-encoding baseline* keeps the shared decision context and removes only the world-state encoding. The *RL baseline* is a PPO-trained (Schulman et al., 2017) RL policy without any LLM.

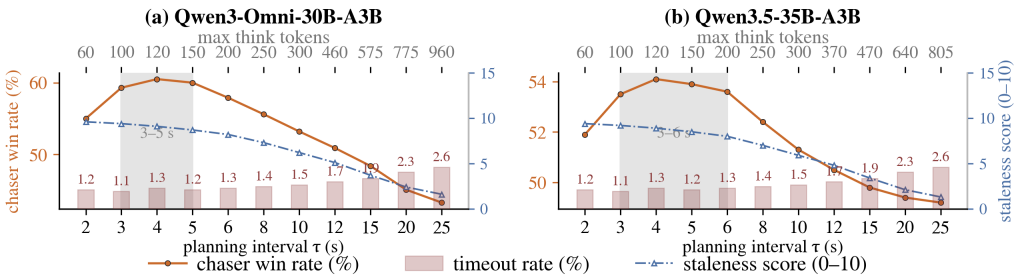

Figure 4: Planning-interval sweep. Chaser win rate (left axis, solid), the LLM-judged staleness score (right axis, dash-dotted, 0–10; higher means the decision better matches the board at execution), and the response timeout rate (bars) as functions of τ. Shaded bands mark each model's effective window; the top axis shows the paired maximum output-token budget.

4.1.2 Metrics

Episode-level metrics. The primary metrics are *chaser win rate* (1− survivor win rate) and *eliminations per game* (4− retreats: each game has four survivors that either retreat or are eliminated); both metrics are higher-is-better for the chaser.

Task-level metrics. Task-level metrics measure the steps between the start of a game and its final outcome. The *task achievement rate* (TAR) is the fraction of tasks that reach their assigned goal, and the *task achievement time* (TAT) is the median duration of achieved tasks; failed tasks are excluded rather than censored. TAR is reported for three of the chaser's tactical tasks—chase, hook-to-altar, and guard-altar—which carry the large majority of task assignments in the evaluated games. For chase, achievement requires the assigned target to be knocked down before the planner switches task or target; failure is recorded when the target survives to a switch within 30 s of assignment or to the 180 s timeout. For hook-to-altar, achievement requires the carried survivor to be hooked onto an altar without breaking free. For guard-altar, achievement requires the hooked survivor to be eliminated without rescue. No TAT for guard-altar is reported: elimination duration is governed by survivor health and the game clock rather than by the planner, so a median time would not measure planner behavior.

4.2 Calibrating the Planning Interval

Controlled interval sweep. The sweep varies the planning interval τ from 2 s to 25 s across eleven levels (Figure 4). Because a decision must land inside its interval, each level also caps the planner's output tokens, with the cap chosen so that generating the longest allowed response takes no longer than the interval itself: 60 tokens at 2 s and 120 at 4 s on both models, rising to 805 at 25 s on Qwen3.5-35B-A3B and 960 on Qwen3-Omni-30B-A3B (caps from each planner's stress test, Appendix E). This cap is enforced by instruction—the prompt asks the planner to keep its answer within a word limit—rather than by the serving stack's output-length parameter, which would truncate a response mid-thought. We cap the output, not the input, because input is cheap: the full prompt is processed at 2,000–3,000 tokens per second (time-to-first-token 0.24 s on Qwen3-Omni-30B-A3B, 0.39 s on Qwen3.5-35B-A3B), while output is generated at only about 30 tokens per second—a 60–100× gap—so the cap acts precisely on the side that limits response time. In addition, both sweeps use the chessboard encoding.

Sweep results. Both models show the same inverted-U in chaser win rate (Figure 4): performance peaks at the 4 s interval and falls off on both sides. On Qwen3-Omni-30B-A3B the peak-to-edge drop is large (60.5% at 4 s versus 55.0% at 2 s and 43.3% at 25 s); on Qwen3.5-35B-A3B the same shape appears with a smaller amplitude (54.1% at 4 s versus 51.9% and 49.2%). The shaded bands mark each model's effective window—3–5 s and 3–6 s respectively—and both cover the 4 s default used by the encoding comparisons. Why does performance fall on both sides? The staleness score—how well a decision still matches the game state when it executes, scored 0–10 by a blind LLM judge (GLM-5.3-Max (GLM et al., 2024), procedure in Appendix E)—explains the long end: it falls

World-state encoding	Episode-level		TAR (%)			Median TAT (s)	
	Win rate ↑	Eliminate ↑	Chase ↑	Hook ↑	Guard-altar ↑	Chase ↓	Hook ↓
Planner: Qwen3-Omni-30B-A3B							
no encoding (baseline)	55.0±3.9	1.91±0.12	41.0±2.2	80.9±4.0	37.4±2.2	133.7±3.5	15.1±0.4
graph-text tree	57.6±2.9	1.95±0.10	42.8±2.3	82.6±4.1	39.9±2.4	128.6±3.4	14.5±0.4
topology	58.2±2.8	1.95±0.09	44.1±2.4	84.1±4.1	41.2±2.4	124.4±3.3	14.0±0.4
plain text	56.3±2.4	1.91±0.08	39.6±2.1	79.2±3.9	35.8±2.1	139.9±3.7	15.8±0.4
vis. minimap	54.3±4.1	1.83±0.13	38.1±2.1	77.4±3.8	33.6±2.0	147.2±3.9	16.4±0.4
chessboard (ours)	60.5±2.6	2.03±0.09	46.5±2.5	86.8±4.3	44.7±2.6	117.2±3.1	13.2±0.3
chessboard JSON (ours)	**62.8±2.5**	**2.10±0.08**	**48.2±2.6**	**88.5±4.3**	**46.8±2.8**	**112.0±3.0**	**12.7±0.3**
Planner: Qwen3.5-35B-A3B							
no encoding (baseline)	49.6±3.3	1.68±0.09	40.8±2.2	80.2±3.9	37.8±2.2	138.9±3.7	15.7±0.4
graph-text tree	51.4±2.9	1.70±0.09	39.2±2.1	78.7±3.9	36.2±2.1	144.1±3.8	16.2±0.4
topology	49.3±3.3	1.65±0.10	34.4±1.9	73.3±3.6	30.2±1.8	165.8±4.4	18.5±0.5
plain text	45.3±2.5	1.54±0.07	35.9±1.9	74.8±3.7	31.8±1.9	158.6±4.2	17.8±0.5
vis. minimap	47.7±2.4	1.64±0.07	37.5±2.0	76.9±3.8	34.0±2.0	150.3±4.0	17.0±0.4
chessboard (ours)	54.1±3.1	1.82±0.10	42.9±2.3	82.8±4.1	40.4±2.4	131.7±3.5	14.9±0.4
chessboard JSON (ours)	**56.8±2.9**	**1.90±0.10**	**44.6±2.4**	**84.4±4.1**	**42.3±2.5**	**126.5±3.3**	**14.3±0.4**
RL baseline (no LLM)	41.2±2.5	1.28±0.05	33.0±1.8	71.8±3.5	28.5±1.7	172.0±4.6	19.1±0.5

Table 2: Main comparison across world-state encodings, averaged over the seven evaluated maps and grouped by planner model. Entries are mean ± standard error across maps (percentage points for rates); metrics are defined in §4.1.2 (Hook abbreviates hook-to-altar; no TAT for guard-altar, whose duration is not planner-governed). Best (red) and second-best (blue) are highlighted within each model block; the planner-independent RL baseline is not ranked.

steadily from 9.6/9.4 at 2 s to 1.6/1.3 at 25 s on the two models, tracking the win-rate decline. At 25 s a decision is typically made for a state the game has already left. Moreover, timeouts are not the cause: only 1.1%–2.6% of responses time out at any level, too few to account for the 17.2 pp drop from the 4 s peak. The short end fails differently: staleness there is at its best (9.6 at 2 s), yet win rate still sits 5.5 pp below the peak—the 2 s interval allows too few output tokens for deep thinking, so each decision is shallow. In short, long intervals make plans stale, and short intervals make them shallow.

4.3 Comparing World-State Encodings

4.3.1 Main Comparison

Table 2 aggregates the completed same-map comparisons per condition; metrics are defined in §4.1.2. The response timeout rate is not reported: it stays below 1% for every encoding at the 4 s default interval.

Which encodings help? Table 2 confirms the drawback analysis of §3.2. The chessboard family leads throughout: chessboard gains 5.5 pp over no encoding under Qwen3-Omni-30B-A3B and 4.5 pp under Qwen3.5-35B-A3B, chessboard JSON 7.8 pp and 7.2 pp, with matching TAR and TAT leads under both models (per-task assignment distribution in Appendix D). The gains from the prior-work text encodings are small and shrink across the model change: the tree and topology gain 2.6 pp and 3.2 pp under Qwen3-Omni-30B-A3B, yet under Qwen3.5-35B-A3B the tree's gain shrinks to 1.8 pp and the topology falls 0.3 pp below the baseline; plain text swings from $+1.3$ pp to -4.3 pp—all three pay the coordinate-conversion cost. The visual minimap is the only condition below the baseline under both models: distances stay implicit in pixels, and pixel-to-text conversion spends the output budget. The RL baseline, at 41.2% without any LLM, trails every LLM condition. The chessboard family removes exactly what the others pay for: the layout is read off the grid, and distances are explicit values in a single modality.

Chessboard wins on every map. Per-map win rates (Figure 5, compared within a panel only) preserve the chessboard advantage everywhere: chessboard improves on no encoding on all seven

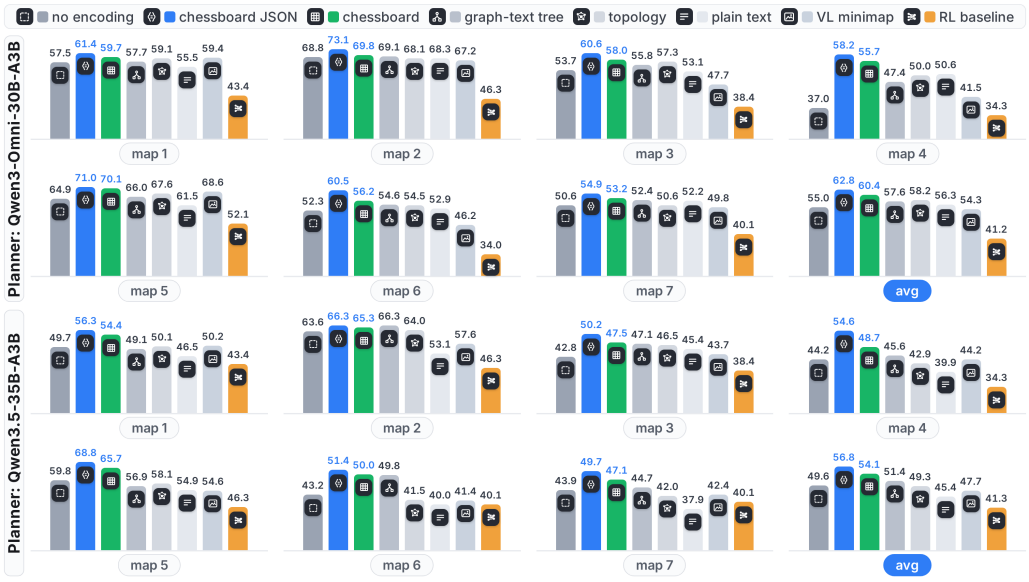

Figure 5: Chaser win rate per evaluated map (M1–M7) and the seven-map average (last panel), under two planner models: top row Qwen3-Omni-30B-A3B, bottom row Qwen3.5-35B-A3B. Each panel shows the eight conditions of Table 2; numeric labels give the exact chaser win rate.

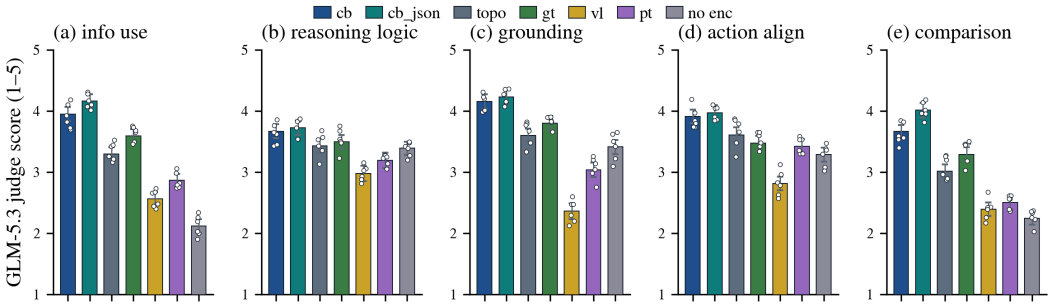

Figure 6: LLM-as-judge assessment of decision outputs across the seven conditions (Qwen3-Omni-30B-A3B planner) on five dimensions: (a) information utilization, (b) reasoning logic, (c) grounding, (d) action alignment, and (e) comparison. Bars give the per-encoding mean with bootstrap 95% confidence intervals; white dots are per-map means; scores range 1–5. Procedure and prompt in Appendix F.

maps under both models—from a 1.0 pp gain on M2 to 18.7 pp on M4 under Qwen3-Omni-30B-A3B, and 1.7–6.8 pp under Qwen3.5-35B-A3B. The chessboard JSON variant improves on both no encoding and chessboard itself in all fourteen per-map comparisons under each planner.

4.3.2 Structure and Conversion Cost

Judge-scored decision quality follows the same ordering. Fig. 6 reports a blind LLM judge (GLM-5.3-Max (GLM et al., 2024)) scoring decision outputs sampled from the Qwen3-Omni-30B-A3B runs on five dimensions: *information utilization* (using concrete input fields), *reasoning logic* (self-consistent chains), *grounding* (statements match the input), *action alignment* (the decision follows the reasoning), and *comparison* (candidates ranked). The ordering mirrors the outcome table of §4.3: the chessboard family leads on every dimension, chessboard JSON the top scorer throughout, with the tree and topology in the middle. The two weakest outcome conditions fail in different ways. Under no encoding, reasoning logic scores on par with the mid-field—a planner can narrate a self-

	Chaser win rate (%), higher is better						
Family / variant	M1	M2	M3	M4	M5	M6	M7
Chessboard family							
chessboard (full)	59.7	69.8	58.0	55.7	70.1	56.2	53.7
chessboard, full map (15×15)	55.6	68.7	55.7	51.1	69.7	54.8	52.7
chessboard, no legend	56.1	67.4	55.1	47.5	66.6	53.4	52.1
chessboard JSON (full)	**61.4**	**73.1**	**60.6**	**58.2**	**71.0**	**60.5**	**54.9**
chessboard JSON, full map	56.3	69.6	58.2	53.3	70.0	55.2	53.2
chessboard JSON, no legend	58.6	69.5	56.8	51.0	67.4	54.5	52.1
Graph-text family							
graph-text tree (full)	57.7	69.1	55.8	47.4	66.0	54.6	52.4
graph-text tree, no slimming	55.2	66.9	55.0	46.4	65.8	53.0	50.9
graph-text tree, no second hop	53.1	65.3	52.9	37.0	55.0	47.2	50.0
graph-text tree, simplified coords	**58.6**	**69.5**	**57.0**	**48.9**	**67.5**	**55.8**	**53.6**
Topology family							
topology (full)	59.1	68.1	57.3	50.0	67.6	54.5	50.6
topology, no enrichment	54.8	66.8	55.6	47.5	57.2	49.3	47.9
topology, simplified coords	**59.4**	**69.3**	**57.9**	**51.3**	**69.0**	**55.9**	**51.9**

Table 3: Within-family ablations under the Qwen3-Omni-30B-A3B planner: chaser win rate (%). **Bold** = best within family per map; comparisons are within-family and directional.

consistent plan with little input—but information utilization and comparison are the lowest: with nothing to read, the planner neither cites concrete fields nor compares candidates. The visual minimap scores worst on grounding and action alignment—the process-side counterpart of its outcome deficit.

Structure matters within one family. Table 3 removes one component at a time from each family—the chessboard's window and legend, the tree's slimming pass and second hop, the topology's enrichment—and adds, for the tree and topology, a simplified-coordinates variant replacing raw map coordinates with grid indices (the chessboard's third advantage, Table 1). Every removal lowers chaser win rate: the second hop and the enrichment by up to 11.0 pp and 10.4 pp on all seven maps, the chessboard window and legend in all 28 comparisons—components of every kind carry weight, so the benefit of a world-state encoding comes from how the state is organized, not from structuredness in general. The simplified-coordinates variants are the only additions that help: best in their families on every map, edging out the raw-coordinate full encodings by 0.3–1.5 pp. The chessboard's coordinate simplification transfers to the linear formats, though not enough to close the gap. The cross-map comparisons are directional; a controlled single-variable ablation on M3 ($n \geq 800$ per condition) confirms them within the graph-text family: flattening the hierarchy costs 10.7 pp, removing relation labels 3.0 pp and key-entity annotation 2.7 pp, while widening the window changes little.

5 Conclusion

We studied training-free high-level planning in a one-versus-four asymmetric pursuit game, where the two design choices are the planning interval and the world-state encoding. Through a controlled interval sweep, we calibrated the interval at 4 s and attributed the performance declines at short and long intervals to shallow thinking and stale states, respectively. Through a controlled encoding comparison at that interval, we found the chessboard encodings improving on the no-encoding baseline on every map under both models while prior-work encodings supply no stable gains, and showed—via component ablations and a blind LLM judge—that the benefit comes from how the state is organized, not from structure or spatial detail alone. Planning interval and state organization are thus complementary choices: the interval bounds how much the planner can think, and the encoding decides how far that thinking goes.

AI USE STATEMENT

In this work, we used generative AI tools (LLM-based coding assistants) for writing and editing assistance: drafting and polishing prose, editing and cross-checking the LaTeX sources (tables, figures, and the appendix structure), unifying terminology across sections, and assisting with small data-processing and plotting scripts that render tables and figures from recorded experiment snapshots. We have not used generative AI assistants for research ideation, experimental design, running the game experiments, or collecting gameplay data. Gameplay and latency metrics, including win rates and task achievement rates, are computed from the evaluation platform's recorded logs and archived snapshots, not generated by AI. Staleness and output-quality scores are produced by the LLM-as-judge procedures described in Appendices E and F. We have reviewed all and every quantitative claim in the paper was traced back to its source file or log. Citation suggestions made by AI tools were verified against the original papers by the authors. We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

Yunpeng Gao, Zhigang Wang, Pengfei Han, Linglin Jing, Dong Wang, and Bin Zhao. Exploring spatial representation to enhance LLM reasoning in aerial vision-language navigation. *arXiv preprint arXiv:2410.08500*, 2024. Introduces the Semantic-Topo-Metric Representation (STMR).

Zhipu AI GLM, Aohan Liu, Bao Xiao, Kai Wang, Huipeng Xue, et al. ChatGLM: A family of open large language models from GLM-130B to GLM-4 series. *arXiv preprint arXiv:2406.12793*, 2024.

Lyle Goodyear, Rachel Guo, and Ramesh Johari. The effect of state representation on LLM agent behavior in dynamic routing games. *arXiv preprint arXiv:2506.15624*, 2025.

Haoyu Han, Yaochen Xie, Hui Liu, Xianfeng Tang, Sreyashi Nag, William Headden, Yang Li, Chen Luo, Shuiwang Ji, Qi He, and Jiliang Tang. Reasoning with graphs: Structuring implicit knowledge to enhance LLMs reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25698–25714, 2025.

Jannik Hösch, Alessandro Sestini, Florian Fuchs, Amir Baghi, Joakim Bergdahl, Iolanda Leite, Konrad Tollmar, Jean-Philippe Barrette-LaPierre, and Linus Gisslén. Hierarchical control in multi-agent games: LLM-based planning and RL execution. *arXiv preprint arXiv:2606.20014*, 2026.

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. lmgame-bench: How good are LLMs at playing games? *arXiv preprint arXiv:2505.15146*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pp. 611–626, 2023.

Chen Li, Sijie Cheng, Yuelin Zhang, Junxi Li, Maozhi Huang, Yang Liu, and Wenbing Huang. GraphThink: Graph-enhanced LLM thinking for long-horizon embodied task planning. *arXiv preprint arXiv:2608.07905*, 2026.

Yi Liao, Yu Gu, Yuan Sui, Zining Zhu, Yifan Lu, Guohua Tang, Zhongqian Sun, and Wei Yang. Think in games: Learning to reason in games via reinforcement learning with large language models. *arXiv preprint arXiv:2508.21365*, 2025.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Prithviraj Ammanabrolu, Yejin Choi, and Xiang Ren. SwiftSage: A generative agent with fast and slow thinking for complex interactive tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12, 2024.

Nicolas Martorell. From text to space: Mapping abstract spatial models in LLMs during a grid-world navigation task. *arXiv preprint arXiv:2502.16690*, 2025.

Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wołczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. BALROG: Benchmarking agentic LLM and VLM reasoning on games. In *International Conference on Learning Representations (ICLR)*, 2025.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Yuanfu Sun, Zhengnan Ma, Yi Fang, Jing Ma, and Qiaoyu Tan. GraphICL: Unlocking graph learning potential in LLMs through structured prompt design. *arXiv preprint arXiv:2501.15755*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024.

Yuxiang Wang, Xinnan Dai, Wenqi Fan, and Yao Ma. Exploring graph learning tasks with pure LLMs: A comprehensive benchmark and investigation. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 7909–7942, 2026.

Annie Wong, Aske Plaat, Thomas Bäck, Niki van Stein, and Anna V. Kononova. State design matters: How representations shape dynamic reasoning in large language models. *Transactions on Machine Learning Research*, 2026.

Hao Xu, Xiangru Jian, Xinjian Zhao, Wei Pang, Chao Zhang, Suyuchen Wang, Qixin Zhang, Zhengyuan Dong, Joao Monteiro, Bang Liu, Qiuzhuang Sun, and Tianshu Yu. GraphOmni: A comprehensive and extensible benchmark framework for large language models on graph-theoretic tasks. In *International Conference on Learning Representations (ICLR)*, 2026.

Christos Xypolopoulos, Guokan Shang, Xiao Fei, Giannis Nikolentzos, Hadi Abdine, Iakovos Evdaimon, Michail Chatzianastasis, Giorgos Stamou, and Michalis Vazirgiannis. Graph linearization methods for reasoning on graphs with large language models. *arXiv preprint arXiv:2410.19494*, 2024.

Yutaro Yamada, Yihan Bao, Andrew K. Lampinen, Jungo Kasai, and Ilker Yildirim. Evaluating spatial understanding of large language models. *Transactions on Machine Learning Research*, 2024.

Mike Zhang, Kaixian Qu, Vaishakh Patil, Cesar Cadena, and Marco Hutter. Tag map: A text-based map for spatial reasoning and navigation with large language models. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2120–2146. PMLR, 2025.

Xudong Zhang, Jian Yang, Shengkai Wang, Jiangpeng Tian, Shaowen Chen, Xian Wei, Ke Li, and Xiong You. The sword, shield, and achilles' heel: Characterizing the linguistic inductive bias of large language models for spatial reasoning in navigation planning. *arXiv preprint arXiv:2605.31404*, 2026.

Jianan Zhao, Le Zhuo, Yikang Shen, Meng Qu, Kai Liu, Michael Bronstein, Zhaocheng Zhu, and Jian Tang. GraphText: Graph reasoning in text space. *arXiv preprint arXiv:2310.01089*, 2023.

A Discussion and Limitations

Evidence-calibrated guidance. Three recommendations have different evidential status. First, the completed study supports checking state encodings in the actual planning setup rather than assuming that more detail or more formatting helps. Second, we calibrate the 3–5 s interval against outcomes, freshness, and process measures; it is an effective trade-off within this system, not a universal optimum. Third, within-family ablations give the accessible-fields hypothesis initial support—removing any single component of the strongest encodings hurts on most maps—but it remains a design hypothesis, not yet a validated universal recipe. Per-frame LLM/VLM control lies outside the evaluated setup; its latency and cost are motivations for this design, not impossibility claims.

Limitations. (1) *Limited model and serving scope*: all experiments use two planner models from one family on one serving stack. We do not compare proprietary frontier models or claim that the selected planners are capacity-optimal; replication across model families is required to test robustness. (2) *Single game and engine*: we offer conditional principles and testable hypotheses, not universal theorems. (3) *Chinese prompts and labels*: effects may interact with language. (4) *Mostly directional evidence*: beyond the controlled structure ablation, the cross-map mean comparisons carry no formal statistical tests, and conclusions about effect magnitudes are directional. One exception is sign consistency: the chessboard-versus-baseline gains are positive in all fourteen map–model pairs (§4.3); under a no-effect null, each pair's gain is equally likely to be positive or negative, so fourteen same-sign gains have probability $2^{-14} \approx 6 \times 10^{-5}$ (sign test). This is a single pre-specified comparison, not a selection over the $7 \times 2 \times 7$ table, so no multiple-comparison correction applies; even correcting for the six encoding-versus-baseline comparisons the table reports would give $p \approx 3.7 \times 10^{-4}$. The significance attaches to the sign consistency, not to each map's magnitude: the per-map gains range from 1.0 to 18.7pp, and dropping the largest (M4) still leaves all six remaining maps positive and a mean gain of 3.3pp, above every prior-work encoding's mean gain. (5) *Interval scope*: the sweep pairs each interval with its output-token budget—a constraint of the deployed system, where a decision must land within its interval—so the two are not independently manipulated; the end-effect attributions rely on the staleness score and timeout measurements rather than on decoupled arms. The representation study uses one fixed interval, and the interval calibration holds only for the tested model, game pace, and setup. (6) *Setup scope*: no comparison to fine-tuned planners or per-frame LLM/VLM agents is made.

B Game Rules

Progression and terminal rules. The game advances through three phases: repairing machines, opening the retreat gate once enough machines are repaired, and retreating through the open gate. The chaser eliminates survivors individually. Two attacks are required to knock a survivor down, and knock-down itself costs no life. A carried survivor who is not hooked in time breaks free and returns to play. Hooking a survivor onto an altar deducts one of the survivor's three lives; every additional 60 s on the altar deducts another, so a survivor who stays hooked for the full 120 s loses all three lives and is eliminated, unless teammate rescue interrupts the hold first. The survivors win when more than two of the four retreat; the chaser wins by holding retreats to two or fewer.

Game duration. Although no explicit clock bounds a game, two rules push games to end: the altar timer drains hooked survivors on a fixed schedule, and each survivor has only three lives. In the evaluated games the effect is that play ends within a narrow band: games last 2,000–3,000 frames at 30 fps, about 7–10 minutes, and no runaway long game appears in the logs. Long games with their extra decisions cannot systematically favor one encoding over another.

Observation scope. A common choice is to give the planner a player-visible state, with hidden information excluded to keep the gameplay realistic (Liao et al., 2025). We instead give every condition the same complete game state read from the engine: entity positions, states, and game progress. Visibility is therefore not a variable in this study. Windowed encodings such as the chessboard simply select a region of this state when rendering it; the question we study is how a complete state is rendered and organized, not how an agent observes under partial observability.

C THE LLM ACTION SET, ENUMERATED

This appendix expands the state-indexed tactical action set of §3.1. The planner's decision space is never the raw entity or coordinate space: each interval, the prompt presents a task menu $\mathcal{T}(s)$ and, under each listed task t, an explicit candidate-target list $\mathcal{Z}(s,t)$, and the output contract is one valid pair drawn from $\mathcal{A}_{\mathrm{LLM}}(s) = \bigcup_{t\in\mathcal{T}(s)}\{t\} \times \mathcal{Z}(s,t)$. Both the menu membership and the candidate contents are functions of the live state, re-evaluated every interval.

Table 4: The six tactical task types, the state condition under which each enters the menu $\mathcal{T}(s)$, and the composition of its candidate-target list $\mathcal{Z}(s,t)$. "Task switch" denotes the experiment-specific gate that can disable a task type entirely (§3.1); a task must satisfy both its switch and its state condition to appear in the menu.

Task t	Enters $\mathcal{T}(s)$ when	Candidate targets $\mathcal{Z}(s,t)$
chase	the chaser is not carrying a survivor	free (non-downed) survivors, each annotated with distance, a health band, and—during the open-gate phase—the survivor's distance to the nearest gate
catch	some survivor is downed	downed survivors, flagged when one is being rescued
hook-to-altar	the chaser is carrying a survivor and a free altar exists	unoccupied, unclaimed altars, distance-sorted
destroy-machine	task switch on and some machine is partially repaired	machines, annotated with distance, repair progress, and destructibility
guard-gate	task switch on and no gate is open yet	still-closed gates, annotated with a guard priority and the nearest survivor's distance
guard-altar	task switch on and some survivor is hooked on an altar	the hooked survivors (equivalently, the occupied altars); the guarded position is fixed

Table 4 makes the state-indexing concrete. Availability is conjunctive: a task type is offered only if its experiment-level switch is on *and* its state condition holds, so the menu itself breathes with the state machine—carrying a survivor removes *chase* and adds *hook-to-altar*; a downed survivor adds *catch*; a completed hook opens the *guard-altar* window until the survivor is rescued or eliminated. Candidate lists are likewise filtered and annotated rather than exhaustive: occupied altars are excluded from *hook-to-altar*, downed survivors are excluded from *chase* (they belong to *catch*), and each list is distance-ordered with per-target attributes so that every candidate in the prompt is a real, currently valid choice.

Figure 7 traces one interval end-to-end.

Game state s. The chaser is carrying survivor `D2`. Altars: `A1` occupied, `A2` free (distance 8), `A3` free (distance 14). Survivors: `D1` free (distance 20), `D3` free (distance 33), `D4` downed (distance 6, being rescued). One machine is under repair (`M2`, 46%); all gates are closed.

Task menu $\mathcal{T}(s)$. `catch`, `hook-to-altar`, `destroy-machine` – *chase* is excluded because the chaser is carrying; *guard-altar* because no survivor is hooked on an altar; *guard-gate* because no gate is eligible in this phase.

Candidate lists $\mathcal{Z}(s,t)$. `catch`: `D4` (distance 6, being rescued). `hook-to-altar`: `A2` (distance 8), `A3` (distance 14)—`A1` occupied, excluded. `destroy-machine`: `M2` (distance 21, repair 46%).

Planner output. `hook-to-altar;A2` $\Rightarrow (t,z) =$ (*hook-to-altar*, `A2`).

Resolution. The pair is resolved to a task one-hot code, the world-space goal coordinates of `A2`, and a designated-entity marker, which condition the fixed RL controller until the next decision (§3.1). Once the hook completes, the deterministic continuations of the state machine take over (a hooked survivor opens the *guard-altar* window), consistent with the assignment counting in Appendix D.

Figure 7: A worked example of the decision contract at one planning interval. The menu and every candidate list are constructed from the current board state; the planner selects one pair, which is resolved to controller-side features.

Two properties follow from this construction. First, the per-decision output space is small and fully enumerable at each interval (menu entries times their candidates), which keeps the contract checkable: a reply naming a task absent from the menu, or a target absent from that task's list, is invalid and is not executed; the RL controller keeps executing the previous task and target. Second, because menu membership and candidate contents are state-conditioned, changes in the action set itself are informative about the game phase—a property the assignment-share analysis of Appendix D exploits when it explains which tasks dominate the planner's decisions.

D Task Assignment Distribution

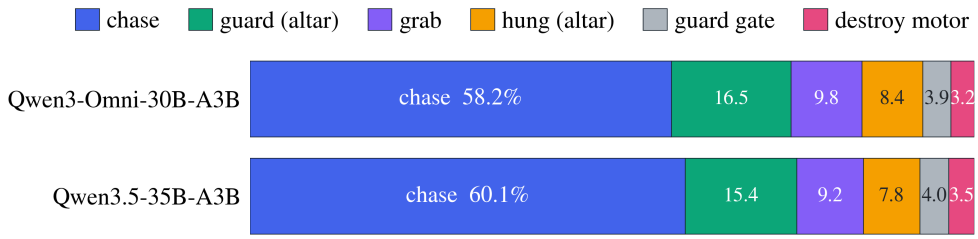

Figure 8: Per-task assignment shares of the planner over the evaluated games, for the two planner models. Each bar aggregates one decision assignment (`plan;target` output per planning interval); the six tasks are chase, guard-altar, catch, hook-to-altar, guard-gate, and destroy-machine (rendered as guard, grab, hung, guard gate, and destroy motor in the figure).

Figure 8 reports the per-task assignment shares behind the three-task focus of Table 2. The counting unit is one decision assignment; assignments produced by the deterministic state-machine overrides (carrying $\rightarrow$ hook-to-altar, hooked $\rightarrow$ guard-altar, knocked-down $\rightarrow$ catch) are counted under their respective tasks, so the shares include both free planner choices and rule-driven continuations. The distribution follows the structure of the game. Chase dominates (58–60% under both models): it is available whenever the chaser is not carrying a survivor, serves as the default state of the task machine, and carries the only win-advancing chain. Guard-altar is second (15–17%): every successful hook opens a stay-by-the-altar window that lasts until the survivor is rescued or eliminated. The two carry-phase tasks have much narrower availability—catch requires a downed survivor within reach, and hook-to-altar requires the chaser to be carrying one with a free altar available—so together they carry 17–18%. Guard-gate and destroy-machine are gated to late-game and mid-game niches respectively and remain in the low single digits; both are enabled in the evaluated setup but are omitted from Table 2 for space and reported here for completeness.

The three focus tasks of Table 2 (chase, hook-to-altar, guard-altar) jointly account for 83.1% and 83.3% of assignments under the two planners, and adding catch raises coverage above 92%; the main-text restriction to three tasks therefore retains the large majority of planner decisions. The two mixtures are nearly identical across planner models, so the restriction does not depend on which planner is used.

E Interval-Sweep Details

Table 5 lists the per-level data behind Figure 4: outcome, behavior, and the full process-metric set (generation latency and response timeout rate) for both models; the LLM-judged staleness scores are listed under the staleness-judge procedure below. Both sweeps use the chessboard encoding. All comparisons are directional and within-level.

Staleness judge. The staleness score plotted in Figure 4 is produced by an LLM judge (GLM-5.3-Max (GLM et al., 2024)) under the same procedure as Appendix F: blind to condition, interval, and map labels, decoding temperature 0. For each sampled decision the judge receives three blocks—the game state the decision was made for, the game state when the decision lands for execution, and the decision itself—and scores 0–10 how well the decision still matches the state at landing,

Table 5: Complete planning-interval sweep data corresponding to Figure 4 (eleven levels, two planner models). Chaser win rate (1 − survivor win rate, higher is better); retreats per game; p50/p95 are generation latencies; timeout is the response timeout rate (decisions that do not arrive within the interval; the RL controller keeps executing the previous task and target).

			Outcome		Gen. latency		
τ (s)	max out tok.	n	chaser win (%)	retreats	p50 (s)	p95 (s)	timeout (%)
Qwen3-Omni-30B-A3B							
2	60	912	55.0	2.42	1.19	2.81	1.2
3	100	1034	59.3	2.21	1.22	2.76	1.1
4	120	1165	60.5	2.15	1.21	2.79	1.3
5	150	1087	60.0	2.18	1.20	2.83	1.2
6	200	996	57.9	2.31	1.23	2.78	1.3
8	250	943	55.6	2.48	1.18	2.75	1.4
10	300	889	53.2	2.63	1.22	2.80	1.5
12	460	862	50.9	2.79	1.19	2.77	1.7
15	575	918	48.4	2.95	1.21	2.82	1.9
20	775	874	45.1	3.18	1.20	2.79	2.3
25	960	831	43.3	3.32	1.22	2.84	2.6
Qwen3.5-35B-A3B							
2	60	1042	51.9	2.63	1.36	3.02	1.2
3	100	1121	53.5	2.58	1.38	2.98	1.1
4	120	1168	54.1	2.55	1.37	3.01	1.3
5	150	1103	53.9	2.56	1.39	3.05	1.2
6	200	1057	53.6	2.59	1.41	2.99	1.3
8	250	984	52.4	2.65	1.37	2.97	1.4
10	300	937	51.3	2.72	1.38	3.01	1.5
12	370	908	50.5	2.80	1.36	2.98	1.7
15	470	942	49.8	2.89	1.39	3.04	1.9
20	640	876	49.4	3.01	1.37	3.00	2.3
25	805	839	49.2	3.12	1.38	3.09	2.6

operationalizing the staleness definition of §3.1: the highest score means the decision remains fully applicable, the lowest that the situation it addressed no longer exists. Scores are averaged per interval level and computed separately for each model's sweep; the per-level means behind Figure 4 are 9.6, 9.4, 9.1, 8.7, 8.2, 7.3, 6.2, 5.1, 3.7, 2.4, 1.6 (Qwen3-Omni-30B-A3B) and 9.4, 9.2, 8.9, 8.5, 8.0, 7.0, 5.9, 4.8, 3.4, 2.1, 1.3 (Qwen3.5-35B-A3B) for $\tau = 2$–25 s. The Qwen3.5 sequence sits 0.2–0.3 lower at every level, matching its 0.2 s longer p50 generation latency: the slower a decision lands, the further the game has moved from the state it was made for—the staleness definition of §3.1 measured directly.

Budget–interval coupling. The sweep varies the interval and its output-token cap together, so a reader may ask whether the long-end decline comes from the interval or from the larger budget slowing the response. Three measurements separate the two. First, the caps are not a free variable: a decision must land inside its interval, and each cap is set from the serving stress test so that a full-budget response still lands in time—raising a cap beyond that point only creates timed-out, discarded decisions. Second, the slow-response path is directly contradicted by the measurements: the timeout rate stays within 1.1–2.6% at every level, and the generation latency is flat across levels (p50 1.18–1.23 s and p95 2.75–2.84 s on Qwen3-Omni-30B-A3B; p50 1.36–1.41 s on Qwen3.5-35B-A3B, Table 5)—a 25 s decision is no slower to produce than a 4 s one, so the extra budget is not spent on longer generation. Third, the staleness score measures content match rather than response speed, and it falls monotonically with τ: the long-end decisions fail because the game has moved on, not because they arrive late. On the short end, the same score is at its best (9.6/9.4 at 2 s), which rules out staleness; the remaining difference is the budget itself—a 2 s interval allows at most 60 tokens of reasoning. A fully decoupled sweep (fixed interval, varying budget) has no meaningful configuration inside this deployed system, as the limitations also acknowledge.

Staleness judge prompt (English translation; the original was administered in Chinese).

> You are a strict evaluator. A game AI agent made a [Decision] at some moment based on the [Board at Decision Time]; because generating the decision takes time, by the time the decision begins executing, the board has become the [Board at Landing Time]. Evaluate how well the decision still applies to the [Board at Landing Time], giving an integer score 0–10: the higher the score, the better the decision matches the landing board.
>
> Scoring criteria: 9–10 = the key information the decision relied on (target position, friend–foe status, task progress, threat distribution) is materially unchanged at landing; the decision can execute as is; 7–8 = key information essentially unchanged, the decision remains sound, minor adjustment only; 4–6 = some key information has changed (e.g., target moved, a threat appeared or disappeared); the decision direction remains usable but at a discount; 1–3 = key information greatly changed, the decision is clearly stale (e.g., target already caught, task already done, route already blocked); 0 = the situation the decision addressed no longer exists on the landing board.
>
> Output only JSON (nothing else): `{"staleness": <0--10>, "reason": "<rationale ≤30 characters>"}`

Serving load. The planner serving endpoint is characterized by a dedicated load test at 200 requests per minute (QPM 200; 1,200 requests, chat-mode serving with one-sentence reasoning, ≈40–50 output tokens per decision): TTFT p50 394 ms and end-to-end latency p50 2.03 s, p90 2.71 s, p99 9.67 s. End-to-end latency is dominated by the output length (correlation ≈0.9): the 60–75-character tier (≈40–50 tokens) that the one-sentence constraint targets has end-to-end p90 2.1 s, well inside the response budget, whereas runaway long-form reasoning (>120 characters) exceeds it. The controlled evaluations run at the same saturated QPM 200 through concurrent games, so the latency and timeout figures in §4.2 (and §4.3) reflect the deployed serving conditions rather than an unloaded best case.

Full load-test statistics. The load test serves Qwen3.5-35B-A3B with vLLM on the same fixed GPU hardware through the chat-mode endpoint (server-managed reasoning), sustaining 200 requests per minute over 1,200 requests that cycle a 308-scenario pool of the evaluation prompts. Table 6 reports the full latency distributions. Output quality under load: 99.2% of decisions (1,190/1,200) are format-valid `plan;target` outputs, one is empty (runaway reasoning consuming the output budget), and nine fall back to a degraded format; reasoning averages 61 characters (one sentence, as instructed; p90 ≈ 71) and decisions average ≈18 characters including delimiters. The end-to-end tail is bounded and stable: 48 requests (4.0%) exceed 5 s, appearing uniformly at random rather than degrading over time, consistent with server-side concurrency queuing (≈7 requests in flight at QPM 200) rather than client-side behavior.

Table 6: Load-test latency distributions (vLLM, 1,200 requests each). Top: Qwen3.5-35B-A3B at QPM 200, chat-mode serving; bottom: Qwen3-Omni-30B-A3B, same procedure. "–" = not reported. TTFT is time-to-first-token; E2E is end-to-end decision latency.

Metric (ms)	min	P50	P90	P99	max	mean
Qwen3.5-35B-A3B						
TTFT	266	394	435	633	884	401
E2E	1,151	2,028	2,707	9,668	20,470	2,288
Qwen3-Omni-30B-A3B						
TTFT	–	237	277	438	588	–
E2E	–	1,614	2,040	2,705	3,120	–

Table 7 cross-tabulates end-to-end latency against output length in a follow-up tuned run of the same endpoint (1,200 requests; streaming usage is not reported, so token counts are estimated at ≈1.5 characters per token). Output length dominates decision latency (correlation ≈0.9): the instructed one-sentence tier (60–75 characters, ≈40–50 tokens) stays at p90 2.1 s, whereas runaway long-form reasoning (>120 characters, ≥80 tokens) exceeds the response budget even at the median—the serving-side justification for the strict output-length constraint and the paired output-token budgets of the interval sweep.

Does the instructed cap hold? The sweep's token caps are instructions rather than serving-side truncation, so adherence is verified on both planners. In the load-test traces above (counting the full

Table 7: End-to-end decision latency by output length (tuned run, QPM 200, 1,200 requests; token counts estimated at ≈1.5 characters/token).

Output length	≈tokens	n	E2E P50 (s)	P90 (s)	P99 (s)
30–45 chars	20–30	184	1.28	1.46	1.52
45–60 chars	30–40	524	1.58	1.76	1.92
60–75 chars	40–50	346	1.88	2.08	2.20
75–90 chars	50–60	96	2.15	2.43	2.73
90–120 chars	60–80	34	2.47	2.79	2.83
>120 chars	≥80	16	6.33	15.27	15.32

response, reasoning plus decision), over 98% of Qwen3.5-35B-A3B responses stay within 120 characters (≈80 tokens), and the longest Qwen3-Omni-30B-A3B response is ≈120 tokens, completing within 3.1 s—inside the response budget even at the extreme. In the sweep, output length is bounded through generation latency: no level exceeds p95 3.09 s (Table 5), and with output generated at ≈30 tokens per second after a 0.24–0.39 s time-to-first-token, actual outputs stay below ≈90 tokens even at the tail. Every budget from 100 tokens upward is therefore headroom the planner does not fill; the 60-token budget at τ=2 s is the only one within the tail's reach. An overrunning response is never executed truncated: it lands whole within the response budget or is discarded by the timeout, leaving the controller on its previous decision—the failure mode the design accepts in exchange for never acting on a cut-off plan.

F LLM-AS-JUDGE EVALUATION PROCEDURE

This appendix documents the procedure of the LLM-as-judge evaluation used for the output-quality dimensions (information utilization, reasoning logic, grounding, action alignment, and comparison).

Judge configuration. A large model (GLM-5.3-Max (GLM et al., 2024)) scores each sampled decision output on five dimensions using integer scores from 1 to 5. The judge is *blind*: prompts contain no condition, encoding, or map labels. Decoding temperature is fixed at 0. Each judge call presents three blocks: the agent's input information (the world-state encoding section when one is rendered, otherwise the encoding-free context), the agent's reasoning trace, and its final decision. The judge must reply with a single JSON object containing the five scores and one evidence quote of at most 40 characters excerpted from the reasoning trace, retained for auditability.

Sampling and aggregation. Per encoding condition we sample $n = 1000$ valid decision frames from the Qwen3-Omni-30B-A3B runs (valid decisions carrying a non-empty reasoning trace), stratified across maps in proportion to per-directory frame counts with a fixed random seed for reproducibility. Scores are aggregated per (condition, map) cell as means with bootstrap 95% confidence intervals (10,000 resamples); per-condition summaries weight cells by sample size.

The five dimensions.

- **Info use (information utilization).** Whether the reasoning trace actually reads and uses concrete input information (positions, distances, identifiers, states, topology). 5: systematically uses input information for comparison, ranking, or priority reasoning without contradiction; 1: not used at all, or uses information that does not exist in the input (hallucination).
- **Reasoning logic.** Self-consistency and support of the reasoning chain. 5: evidence–reasoning–conclusion closes the loop and considers alternatives or priority rules; 1: internal contradictions, or conclusions conflicting with the trace's own evidence.
- **Grounding (factual consistency; reverse-scored).** Agreement between factual statements in the trace and the input information. 5: all factual statements match the input, no fabrication; 1: extensive fabrication of information absent from the input.
- **Action alignment.** Consistency between the final decision and the trace's own conclusion. 5: the decision strictly executes the reasoned conclusion; 1: decision unrelated to the reasoning.

- **Comparison.** Whether multiple candidate targets are explicitly compared, ranked, or prioritized. 5: explicit multi-candidate comparison with a stated selection rule; 1: no comparison, or unsupported choice.

Reliability and scope. The judge protocol has known limits that we state explicitly. We report no inter-rater reliability: the evaluation uses a single judge model at temperature 0, so the scores measure one model's assessment, not a consensus. Two dimensions are mechanical in character—grounding compares factual statements against the input, and action alignment compares the decision against the trace's own conclusion—so their rubrics are in principle machine-checkable; the other three (information utilization, reasoning logic, comparison) carry subjective judgment with no human ground-truth comparison in this study. The evidence value of the scores rests on triangulation rather than on calibration: the judge is blind to conditions, and its ordering of encodings matches the outcome ordering of Table 2, which is measured independently by the game engine. Score magnitudes should be read as relative ranks across conditions, not as absolute quality measurements.

Judge prompt (English translation; the original was administered in Chinese).

> You are a strict evaluator. Given what a game AI agent saw in a single decision frame—[Input Information] (which may or may not contain a structured world-state encoding), its [Reasoning], and its [Decision]—score it on the following five dimensions, each an integer 1–5, and quote one excerpt of at most 40 characters from the reasoning as evidence.
>
> Dimension 1, info_use (information utilization): whether the reasoning actually reads and uses concrete content from the input (positions/distances/identifiers/states/topological relations). 5 = systematically uses input information for comparison, ranking, or priority reasoning, without contradicting the input; 4 = accurately cites specific input information in multiple places, with the decision rationale tied to it; 3 = cites only one specific piece of information, and the decision rests mainly elsewhere; 2 = only vague paraphrase, no specific input content; 1 = not used at all, or uses information absent from the input (hallucination).
>
> Dimension 2, logic (reasoning logic): the self-consistency of the reasoning chain and how well it supports the conclusion. 5 = evidence→reasoning→conclusion forms a complete loop, considers alternatives or priority rules, with no unexplained assumptions; 4 = chain complete, trade-offs have stated reasons; 3 = a recognizable chain exists but with jumps or unexplained trade-offs; 2 = conclusion unsupported, most of the chain missing; 1 = self-contradictory, or the conclusion directly conflicts with the evidence it cites.
>
> Dimension 3, grounding (factual consistency): the agreement between factual statements in the reasoning and the input information (reverse-scored: higher agreement, higher score). 5 = all factual statements about the input fully match it, no fabrication; 4 = essentially consistent, with only isolated imprecise wording; 3 = a few statements inconsistent with the input; 2 = multiple contradictions; 1 = extensive fabrication of information absent from the input.
>
> Dimension 4, action_align (decision alignment): consistency between the final decision and the reasoning's conclusion. 5 = the decision strictly executes the reasoned conclusion (target/action consistent with the reasoning); 4 = consistent, with only wording-level deviation; 3 = partially related, with clear drift; 2 = largely decoupled; 1 = unrelated to the reasoning.
>
> Dimension 5, comparison (comparison and trade-off): whether candidate targets are explicitly compared/ranked/prioritized. 5 = explicitly compares multiple candidates (distance/state/payoff) with a stated selection rule; 4 = comparison present but single-dimension, rule recognizable; 3 = implicit comparison (e.g., "pick the nearest"), not unfolded; 2 = almost no comparison, target chosen directly; 1 = no comparison, or the choice is unsupported.
>
> Output only JSON (nothing else): {"info_use": <1--5>, "logic": <1--5>, "grounding": <1--5>, "action_align": <1--5>, "comparison": <1--5>, "evidence": "<quote ≤40 characters>"}