THE WORLD DOESN'T WAIT: WINDOWED ACTION COMMITMENT FOR REAL-TIME GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-time GUI agents choose actions from screenshots while the interface continues to change. An action that is correct for the observed state can therefore be invalid when executed. Preparing several actions reduces repeated inference, but leaves later actions tied to increasingly old observations. We study how to preserve useful decisions within such a plan. Our central insight is that task choices can outlive their original execution contexts. We introduce windowed action commitment, which binds a bounded action sequence to observable targets. The executor uses fresh visual feedback to retain compatible actions, skip expired independent prefixes, and rebind moved targets, replanning when the remaining commitments cannot be used. Compared with single-step execution in the faster settings of two instrumented Android tasks, successor reuse raises correctness from 12% to 32% at equal mean model-call count; target rebinding raises correctness from 53.0% to 71.7% while reducing calls by 53.1%. Cross-model diagnostics distinguish constructing a usable window from executing it successfully. These findings identify selective reuse of prepared decisions as a mechanism for maintaining task progress while interfaces change.

1 INTRODUCTION

A GUI agent can choose the right action for a screenshot and still fail when that action reaches the device. In real-time interaction, the environment continues to evolve while the model reasons, so the interface at execution can differ from the one used for planning. Recent benchmarks study GUI dynamics and continuously changing observations (Liu et al., 2026a; Yao et al., 2026). This gap makes action validity time dependent: a planned action must still apply to the target encountered when it is executed.

Preparing actions in advance offers a way to reduce repeated inference. In GUI control, ShowUI-π predicts cursor-action chunks (Hu et al., 2026), while guarded branches prepare responses to transient events (Dong et al., 2026). In robotics, real-time chunking overlaps chunk generation with execution (Black et al., 2025b). As a prepared sequence is executed, however, its remaining actions rely on increasingly old observations. A change can invalidate part of the sequence while leaving other decisions useful, yet both replanning and checking consume time. We therefore ask how an agent can identify and reuse these decisions before their targets change again.

Our central insight is that a task choice can remain useful after its execution context has changed. An answer computed from a preview can still be used when that question becomes current, even if the preceding question has expired (Figure 1). Likewise, moving a target can require a new tap location without requiring a new task choice. Reuse depends on separating the decision about the task from the conditions under which it can be executed.

We introduce *windowed action commitment*, a revocable form of action chunking for real-time GUI interaction. Each window couples a bounded sequence of model-selected actions with descriptions of their visible targets. At execution time, the runtime checks fresh screenshots to retain compatible actions, skip expired independent prefixes, or rebind moved targets. If the remaining commitments cannot be used, it revokes them and requests a new plan. These checks compare pixel-derived

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.

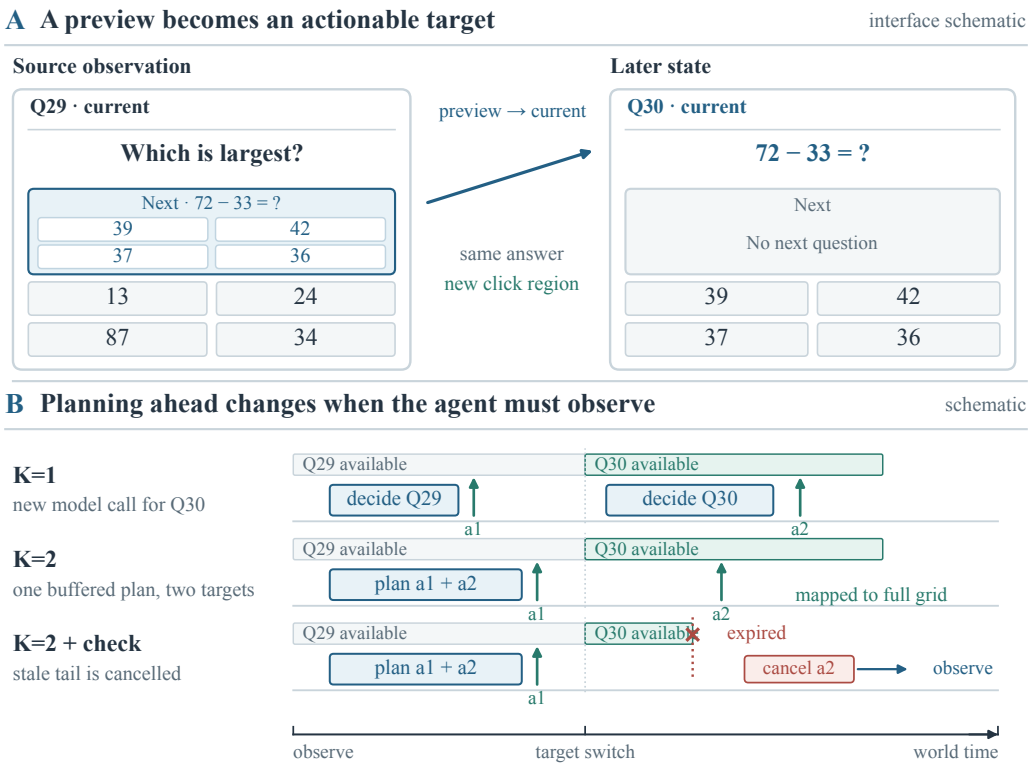

Figure 1: **Decision reuse depends on the target's execution context.** Redrawn TurboQuiz interfaces show a preview becoming the current question. The schematic schedules compare a new decision for each action, a buffered two-action window, and a two-action window with target checks and cancellation. The question and answer coordinates change roles as the interface advances. Interface content follows a historical episode; geometry and schedules are schematic, with no measured durations or outcomes implied.

target descriptions without another model decision, allowing the runtime to revise execution while preserving the model's task choices.

Our contributions are threefold:

1. **A formulation of selective decision reuse.** We characterize real-time GUI interaction through action age, target availability, and target position, separating a model's task choice from its execution context. This formulation explains how prepared decisions can remain useful when an earlier target expires or the intended target moves, making partial plan expiration an explicit problem for action-sequence execution.
2. **Windowed action commitment for real-time GUIs.** We introduce a bounded, revocable execution contract that couples model-selected actions with visible target descriptions. Fresh visual feedback lets the executor retain compatible actions, skip expired independent prefixes, and rebind moved targets, with replanning when the remaining commitments cannot be used. These operations preserve the model's task choices without an additional model decision for each target check.
3. **A design principle for execution-time feedback.** Across two Android tasks with controlled clocks, we show how selective reuse turns fresh visual feedback into task progress under rapid interface change. Paired ablations and target attribution trace the gains to preserving actionable successors and rebinding moved targets. This motivates evaluating feedback by the prepared work it preserves alongside its checking cost. Diagnostics across four additional models identify usable window construction as a prerequisite for reuse.

2 VALIDITY AND REUSE OF ACTION COMMITMENTS

Environment clock and action age. Let $s(t)$ be an application state that evolves during inference, and let o_n be a screenshot acquired at time t_n. A decision requests at most K actions and returns $a_{n,1:k_n}$, with actual length $k_n \leq K$; action i reaches the device at $u_{n,i}$. Its age is $u_{n,i} - t_n$. A target z_i can be actionable only during an interval $I(z_i) = [b_i, e_i)$, so executing the intended action requires the target to be present at injection. Selecting a correct action from o_n does not by itself establish $u_{n,i} \in I(z_i)$.

Window size and temporal extent. We call K the requested window size, measured in actions. Let J_n contain the indices of actions actually injected. For $J_n \neq \varnothing$, the realized temporal extent is $W_n = \max_{i \in J_n} u_{n,i} - t_n$; it is undefined when the entire plan is canceled. W_n includes inference, pacing, observations, and execution overhead. It therefore cannot be inferred from K or an inter-action gap alone. For a plan aimed entirely at the current target, its last injection must precede that target's expiration. A preview-grounded plan can instead refer to several future targets with different validity intervals. The current target's remaining lifetime is then an inappropriate upper bound for the entire window.

Planning and execution-time checks. The model decision maps the source screenshot to task actions. An execution-time check asks whether a pending action still has a compatible target. We represent their interface as

$$p_n = \big(o_n, \{(i, a_{n,i}, \hat{z}_{n,i}, g_{n,i})\}_{i=1}^{k_n}\big), \tag{1}$$

where $\hat{z}_{n,i}$ is a visible target description and $g_{n,i}$ is a scheduling gap. The model supplies task choices; the runtime can attach the target description and gap. A compatibility test $m(o, \hat{z})$ compares a fresh screenshot with that description. It approximates current target compatibility, without predicting its remaining lifetime. A moved target additionally requires rebinding the selected point to its current position.

Partial expiration and useful computation. For two independent target actions, $t \geq e_1$ and $t \in I(z_2)$ describe a moment when the first target has expired while the second is actionable. Computation spent preparing the second action can still be used. Our executor combines advance preparation with target matching to select that action. This explains the complementary roles tested by the factorial experiment: preparation supplies another candidate, and feedback identifies whether it can be used. A target can also remain actionable after leaving its original coordinate; identity-based rebinding then retains the selected operation. In both cases, checking consumes time, so its benefit depends on preserving useful actions before their opportunities pass.

We evaluate this tradeoff through completed task outcomes, calls, and wall time. Source correctness, target correspondence, and reward are distinct: an answer can be correct for its source question, reach a different question, and still happen to receive task credit. Action-level attribution establishes which prepared computation actually reaches its intended target.

3 TARGET-AWARE WINDOW EXECUTION

The execution contract maintains a bounded sequence together with each action's expected target. A fresh observation determines the next action to inject and which remaining commitments to revoke. We instantiate this contract with visible question previews and moving ticket targets.

3.1 CONSTRUCTING A BOUNDED COMMITMENT

TurboQuiz displays a current question with four large answer buttons and a miniature preview of the next question. We restrict the window to $K \in \{1, 2\}$, matching the amount of visible future information. All arms receive the same full screenshot. The model returns four button centers in normalized image coordinates and an ordered list of answer indices: the current answer, followed by the preview answer when $K = 2$. Indices follow a fixed top-left, top-right, bottom-left, bottom-right ordering. The preview supplies answer content; its eventual tap uses a model-predicted center in the large current-answer grid.

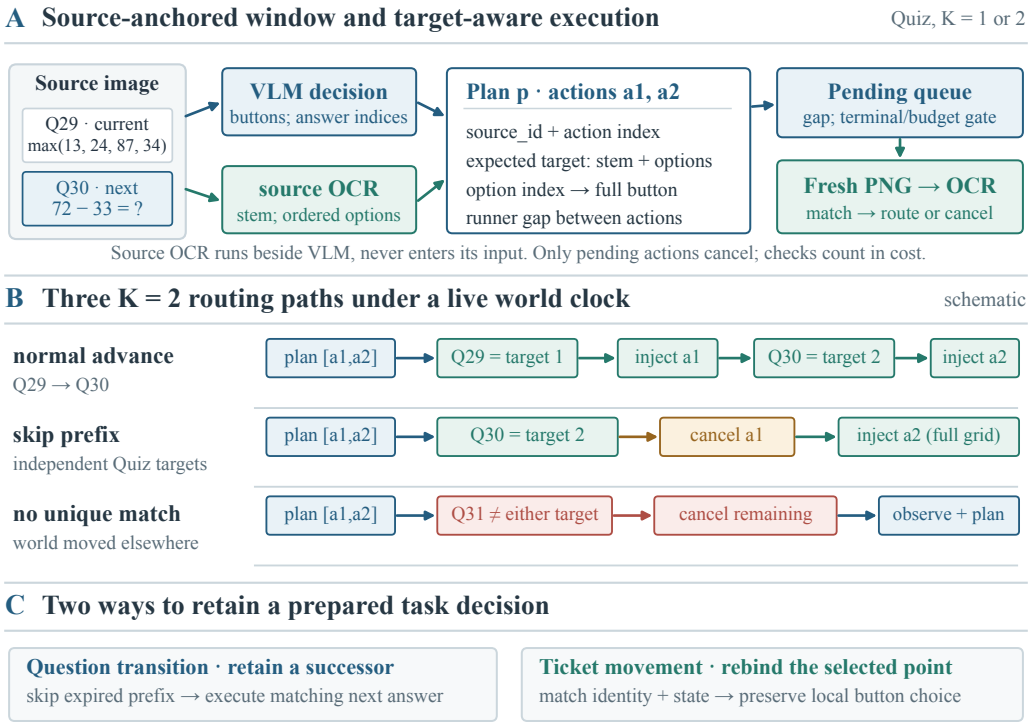

Figure 2: **Target-aware execution of a bounded window.** The schematic source observation supplies pixels to the model and source OCR in parallel. The runtime joins task actions with visible target labels, then checks the current screenshot before dispatch. Three schematic traces show normal progression, cancellation of an expired independent prefix, and cancellation of the tail followed by replanning. The bottom strip relates successor selection in Quiz to identity-based target rebinding in Ticket Queue.

The Quiz executor buffers the complete JSON response before dispatch. It validates the format, index range, and coordinate range, then maps each answer index to a button center. It does not correct coordinates using an environment answer key. The parser also accepts a single-action response under $K = 2$, including at the final question. Actual plan length is logged separately from requested window size. No plan extends to unseen questions.

In feedback arms, source OCR runs on the same screenshot concurrently with the model call. It extracts the current and next question stems and their ordered option strings. After both computations finish, the executor attaches these expected labels to the corresponding actions (Figure 2). OCR does not provide an answer or additional text to the model. This parallel work shares the source observation, so the target description refers to the screen on which the model made its decision.

3.2 Checking, Executing, and Revoking

Let c denote the first unconsumed action in the window. For Quiz, before an injection the feedback executor acquires a new screenshot o and forms

$$M = \{i \in \{c, \ldots, k_n\} : m(o, \hat{z}_{n,i}) = 1\}. \tag{2}$$

The implementation compares normalized question text and all four ordered option strings. It preserves arithmetic operators and option order. Missing, ambiguous, or low-confidence reads do not produce a match. A window then follows one of three paths:

1. **Continue**: if the unique match is the first remaining action, inject it and advance the cursor.
2. **Skip an expired prefix**: if the unique match is a later action, cancel the preceding actions and inject the matching action. This rule relies on the independence of Quiz questions.

3. **Reobserve**: if there is no unique match, cancel the remaining window and begin a new model decision from a fresh screenshot.

A canceled action is never replayed. A successful first tap can advance the question immediately, after which the second action is checked against the new current target. If inference has already consumed the first question's lifetime, the same procedure can route directly to the visible next target. If both planned targets have passed, it requests a new plan.

All variants use the same fixed 700 ms inter-action gap and 150 ms cooldown settings. Feedback adds screenshot and OCR work to the live clock, including the residual interval between checking and tapping. A match refers to the check time; the offline ledger separately measures correspondence at injection. This distinction also exposes repeated-label aliases and target changes between the check and tap.

Rebinding a moved target. In Ticket Queue, the source reader binds each model-selected point to a ticket identifier, issue, and OPEN state. A fresh screenshot locates the same compatible ticket. With the internal card layout fixed, the executor transfers the selected point's relative position to the new card, retaining the model's button choice. If identity or state cannot be matched uniquely, it revokes the tail. This instance has no successor preview or prefix skipping. Original-coordinate checking is its cancellation-only ablation; Appendix C specifies the coordinate transform.

4 Tasks and Experimental Protocol

4.1 Controllable clocks and information boundaries

We use two instrumented interfaces on Android emulators to isolate changes in target availability and position. TurboQuiz reveals a successor before it becomes actionable; Dynamic Ticket Queue moves targets that are already actionable. Seeded task content and controllable clocks let paired policies face the same task content at different rates of interface change.

TurboQuiz. An episode contains 30 generated questions, mixing maximum-number selection and arithmetic. Four options appear in a fixed 2×2 grid; answering or reaching the time-to-live (TTL) advances the question. A smaller NEXT region previews the following question and options. The formal comparison uses TTLs of 7.5 s (L3) and 2.5 s (L6), with identical seeded question sequences across policies. The application remains live during model inference, source OCR, and target checks.

The policy uses screenshots, model outputs, and pixel-derived target labels. Question IDs, answer keys, and transition timestamps are reserved for offline evaluation. Reset preparation verifies the paused initial render before starting the live episode. A terminal lifecycle signal ends interaction; it supplies no answer selection information. The ledger is captured after the decision cutoff to score the run consistently.

Dynamic Ticket Queue. Three simultaneously actionable tickets carry visible identifiers, issue categories, and routing buttons. Positions change every 7.5 or 2.5 s; ten batches with a 15 s lifetime allocate 30 tickets. We cross $K \in \{1, 2\}$ with three execution rules: original coordinates (A/B), original-coordinate checks (C/D), and target rebinding (E/F). Ten new paired seeds at both clocks yield 120 episodes, all complete, using Qwen3.8-27B-SPOT with 80-call and 240-second budgets. Fast F$-$A is the designated primary comparison. An earlier 80-episode cancellation comparison remains separate in Appendix C.

4.2 Factorial window experiment

The TurboQuiz matrix crosses window size and target feedback:

Policy	Requested action limit	Fresh target check
A: single step	$K = 1$	no
B: two-action window	$K = 2$	no
C: single step + feedback	$K = 1$	yes
D: two-action window + feedback	$K = 2$	yes

The matrix contains ten paired seeds at each TTL, for 80 episodes. All four arms use the same Qwen3.8-27B-SPOT endpoint, temperature zero, thinking disabled, a 768-token output cap, and no SDK retries. Budgets are 40 model calls and 600 active seconds per episode, with a 90-second call timeout. The source and current-screen reader uses fixed Quiz regions and a pinned RapidOCR implementation with a 0.80 confidence threshold and two CPU threads. These settings are fixed before data collection.

Each seed–TTL block assigns all four arms to one emulator, rotates their order, and retains the device mapping. The run uses six KVM-backed emulators, with four workers for the first 56 episodes and six for the final 24. This design pairs task content and device within a block while recording serving and observation costs. A call budget includes unsuccessful calls and replanning; feedback does not receive free extra model decisions.

Four additional models repeat this matrix (320 episodes), with each model's endpoint fixed across arms (Appendix D).

Endpoints and contrasts. The primary endpoint is correct answers divided by the fixed 30-question episode budget. Expired, unanswered, and unpresented questions remain in that denominator. Missing scoring telemetry produces a missing value. We also report model calls, token usage, active wall time, cancellation and replanning counts, and the time spent in source OCR and fresh target checks. Together, these measurements distinguish completing more work from spending more time or withholding actions.

The primary contrast is D minus A at the fast 2.5-second TTL. Secondary contrasts are B minus A (windowing without feedback), D minus B (feedback within a window), and C minus A (feedback without windowing). The interaction $(D-B)-(C-A)$ tests whether feedback helps the two-action policy differently from the single-step policy. Per-TTL comparisons use the ten paired seeds; any pooled uncertainty clusters by seed across TTLs. All four cells and both deadlines are retained. The reported intervals use a common post hoc paired percentile bootstrap with 20,000 resamples of the ten seeds; secondary contrasts describe the factorial effects.

5 Results and Mechanism Analysis

The experiments test whether visual feedback preserves prepared work at a useful cost. We first measure task outcomes, then use paired ablations and action-level attribution to explain the gains. The model-interface and release-timing studies distinguish this execution mechanism from two prerequisites: preparing multiple actions and making them available promptly.

5.1 Windows recover useful actions under a changing screen

For Qwen3.8-27B at the 2.5 s deadline, the target-aware two-action policy D increases mean correct answers from 3.6 to 9.6 out of 30 relative to single-step policy A (Table 1). The paired gain is 20.0 percentage points, with a 95% interval of $[14.0, 25.7]$ points; all ten seeds improve. This gain occurs at equal mean model-call count: 14.9 per episode for both policies. D uses more wall time, 63.5 versus 46.6 s, while completing more questions correctly. Omitting any seed leaves the mean gain between 18.5 and 21.9 points.

At the 7.5 s deadline, D uses 15.0 calls versus A's 29.9, a 49.8% reduction, with mean correct counts of 21.9 and 21.8. The paired correctness difference is 0.3 points with an interval of $[-3.3, 4.3]$ points. Unguarded windows B also reduce calls to 15.0 and finish in 62.8 s, compared with D's 93.2 s. The slower regime exposes the computation saved by preparing two answers and the extra time spent checking them.

Table 1: Cross-model Quiz results at the 2.5 s TTL under the original ordered-array interface. A: single step; D: requested $K = 2$ with feedback. Scores and calls are means over ten paired seeds, including budget cutoffs. The last column counts two-action/valid D plans. Serving latency differs across models (Section 5.4).

Model	Correct / 30		Model calls		Two-action D
	A	D	A	D	plans
Qwen3.8-27B	3.6	9.6	14.9	14.9	143/149
Qwen3-VL-8B	19.7	3.5	30.0	15.1	0/151
UI-Venus-1.5-8B	20.0	3.3	30.0	15.0	0/150
MAI-UI-8B	15.4	2.0	32.0	21.9	0/155
UI-Venus-2-9B	18.0	4.4	30.1	15.5	0/153

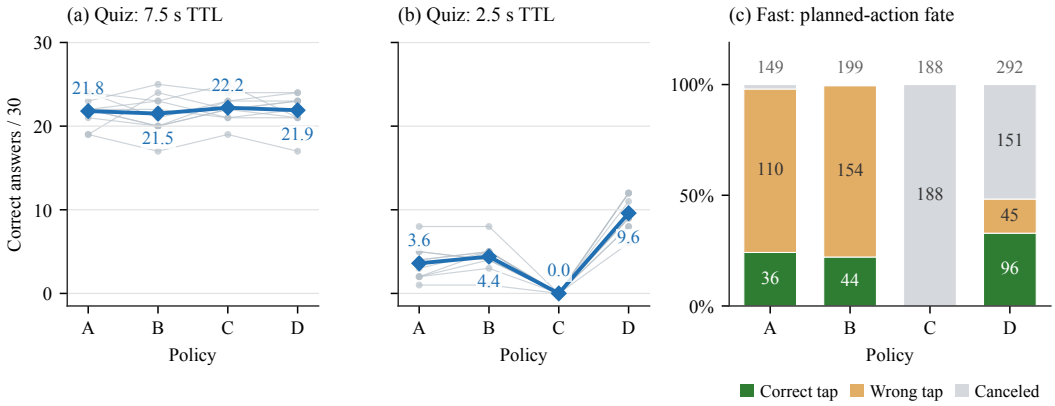

Figure 3: **Window utility and the fate of prepared actions.** Results for Qwen3.8-27B. (a,b) Gray lines connect the four policies for each of ten seeds; blue diamonds mark episode means. (c) Fast-deadline outcomes of all planned actions: numbers inside segments are counts, and totals above bars are the planned-action denominators. C cancels every action. D executes 141 prepared second actions after skipping their first actions, producing 96 correct answers. A/B/C/D denote single step, window, single step with feedback, and window with feedback.

5.2 The gain comes from executing a still-valid successor

The factorial comparison distinguishes windowing from checking. At the fast deadline, unguarded windows B achieve 4.4 correct answers, while single-step checking C achieves zero: all 188 planned actions in C are canceled. Combining the window with feedback yields D's 9.6 correct answers. D exceeds B by 17.3 percentage points; the interaction $(D - B) - (C - A)$ is 29.3 points. Figure 3 connects these task outcomes to the actions that actually reach the application.

The execution path is particularly clear in D. Of 149 plans, 143 contain two actions and six contain one. There are 141 plans in which the first action is canceled and the second is injected; the other eight plans are canceled in full. Thus every one of D's 141 injections uses a prepared successor after a prefix skip. Offline attribution matches 136 of these injections to the intended question: 96 are correct and 40 answer that question incorrectly. The remaining five injections reach a different target and are wrong. This separates the recovered execution opportunity from the model's answer-selection accuracy.

Single-step A and unguarded B produce 146 and 198 fast-deadline injections, respectively, with none matching the source target in the retained pixel/event attribution. Their 36 and 44 credited answers occur on other questions. Source correctness and task reward therefore give different accounts of the same tap. All 1,679 injections across the Quiz matrix have target labels. Appendix B reports the full cross-classification, question types, costs, and a logged prefix skip.

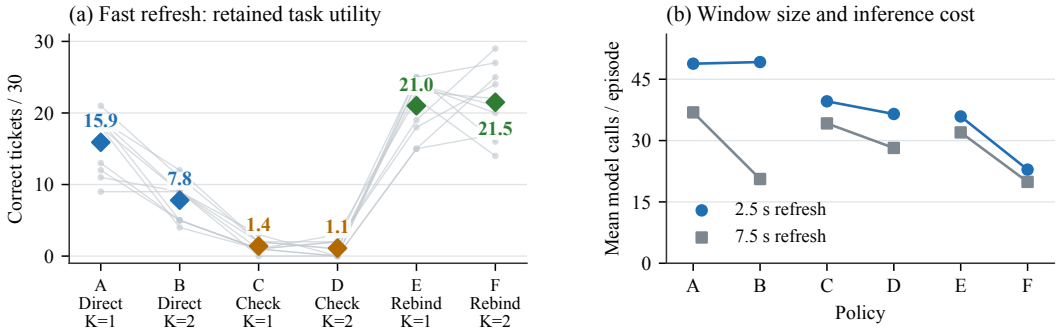

Figure 4: **Preserving decisions across target movement.** (a) Fast-refresh correctness: gray lines connect the six policies for each of ten seeds; diamonds mark means. (b) Mean model calls at both clocks; each segment joins K=1 and K=2 within one execution rule. Rebinding restores task progress and makes the prepared window useful.

5.3 Rebinding preserves a decision when its target moves

At the 2.5 s refresh, rebinding two-action windows F increases mean correct tickets from 15.9 to 21.5 out of 30 relative to single-step A. The paired gain is 18.7 percentage points, with a 95% interval of $[7.3, 30.3]$ points and improvements in nine of ten seeds. Calls fall from 48.8 to 22.9 per episode, a 53.1% reduction; wall time falls from 130.4 to 121.7 s. Figure 4 separates this outcome from the cost of constructing and checking a plan.

The six arms identify what preserves useful work. Unguarded windows B complete 7.8 tickets; coordinate checks C/D complete 1.4/1.1. D cancels 691 planned actions and issues 29. Rebinding F instead issues 325 actions, 297 at updated coordinates, and cancels 49. Rebinding also benefits single-step E, which completes 21.0 tickets using 35.9 calls. Adding a second action reduces calls by 36.2% relative to E; F$-$E correctness is 1.7 points ($[-11.3, 14.7]$). Thus target rebinding supplies the completion gain in this regime, and windowing reduces inference cost.

At the 7.5 s refresh, direct A/B achieve 29.5/29.6 correct tickets. F achieves 28.1 with 19.9 calls, versus A's 36.9; its correctness difference from A is -4.7 points ($[-9.7, -0.7]$). When coordinates persist longer, the extra checks can consume useful time. The full six-arm matrix and receipt coverage appear in Appendix C.

5.4 Window construction and execution across models

Table 1 contrasts five models under the original interface. Across the four additional models, all 3,997 valid B/D plans contain one action despite requested $K = 2$. D therefore has no prepared successor to reuse. Their direct-vLLM calls take about 0.8 s, versus 2.5 s for the 27B service, while fresh checks add 1.1–1.3 s. Window realization and serving latency jointly determine the execution regime.

The original prompt requests CURRENT and NEXT but shows a one-element array. On 12 fixed nonterminal screenshots per model via Theta, explicit two-answer prompts produce two actions in 47/48 calls. Structured CURRENT/NEXT windows then yield two actions on all 57 source-verified nonterminal inputs in complete local fast-D episodes, one fixed seed per model; 36 of 37 attributable successor injections reach their intended targets. These results establish window construction and routing. Appendix D reports the full formal matrices and separate diagnostic protocols.

5.5 Release timing and task validity measure different effects

The matched Tetris comparison isolates release timing across 60 seed–gravity pairs, with overlap and pipelining disabled. Incremental release reduces first-decision injection latency from 6.70 to 2.88 s (56.95%) and increases preterminal injections from 120.40 to 175.17 per episode. Yet 115 of 120 episodes score zero, with a paired mean score difference of -0.25. Earlier release increases activity without observed task-score improvement (Appendix A).

6 Related Work

GUI tasks and grounding. GUI research connects interaction data, executable tasks, and visual grounding. Mind2Web and Android in the Wild supply human interaction sequences (Deng et al., 2023; Rawles et al., 2023); WebArena, VisualWebArena, AndroidWorld, and OSWorld provide executable evaluation (Zhou et al., 2024; Koh et al., 2024; Rawles et al., 2025; Xie et al., 2024). SeeClick and UGround learn grounding (Cheng et al., 2024; Gou et al., 2025), with ScreenSpot-Pro testing high-resolution professional interfaces (Li et al., 2025). UI-TARS and ShowUI integrate perception with action generation (Qin et al., 2025; Lin et al., 2025), while iSHIFT adapts perception and reasoning effort (Mehrotra et al., 2025). Our commitments connect these grounded choices to their later execution contexts.

Dynamic observation and execution latency. DynamicGUIBench studies interface changes between actions (Liu et al., 2026a); LivingScreen evaluates observation control on continuously playing content (Yao et al., 2026). Agent-Computer Observation Interfaces decouple ongoing perception from discrete actions (Li & Shi, 2026). OSWorld-Human compares agent and human efficiency, identifying model calls as a major latency cost (Abhyankar et al., 2026). We measure how elapsed time changes commitment usability, separating source correctness, target correspondence, and task credit.

Prepared policies and reusable execution. AAPT precomputes guarded branches during idle periods and routes observations to preauthorized actions (Dong et al., 2026). ActionEngine constructs state-machine memory, synthesizes programs, and repairs failures through visual regrounding (Zhong et al., 2026). Our paired comparisons isolate which prepared decisions survive target transitions or movement, and whether feedback preserves enough work to offset its cost.

Action sequences and asynchronous control. ACT, Diffusion Policy, and π_0 generate action sequences for robot control (Zhao et al., 2023; Chi et al., 2023; Black et al., 2025a). RTC overlaps chunk generation and execution conditioned on committed actions (Black et al., 2025b). Training-time conditioning, Legato, and DiscreteRTC address continuation and asynchronous generation (Black et al., 2025c; Liu et al., 2026b; Wang et al., 2026). ShowUI-π generates GUI action chunks for clicks and continuous drags (Hu et al., 2026). Our discrete GUI contract can revoke an unexecuted tail, skip an independent expired prefix, or rebind a point while preserving its selected operation. Appendix A also relates release timing to speculative agent serving.

7 Discussion and Conclusion

Conclusion. Useful computation can survive changes to an action's execution context. Windowed action commitment retains that computation by separating task choices from when and where they are executed. Across previewed questions and moving tickets, the measured gains come from preserving an actionable successor or rebinding a selected operation to its target. Checking alone can exhaust opportunities through cancellation. The resulting design principle is to evaluate feedback by the prepared work it preserves, alongside the time and inference cost it consumes.

Reproducibility. Section 3 specifies the execution rules, and Section 4 describes the tasks, model settings, budgets, and paired evaluation protocol. The appendices report additional protocol details, model-interface diagnostics, and statistical analyses. Appendix J documents the experimental records and verification procedures for each campaign.

AI use statement

LLMs and other AI tools assisted with writing and polishing, literature retrieval and discovery, research ideation and execution, and drafting manuscript sections. Research assistance covered conceptual formulation, hypothesis refinement, experimental design, implementation, data processing and analysis, result interpretation, and figure and artifact preparation. Validation included execution tests, outcome replay, and checks against source measurements. The authors are responsible for the final text, code, and scientific claims.

REFERENCES

Reyna Abhyankar, Qi Qi, and Yiying Zhang. OSWorld-Human: Benchmarking the Efficiency of Computer-Use Agents. In *Proceedings of Machine Learning and Systems*, volume 8, pp. 482–494, 2026. URL https://proceedings.mlsys.org/paper_files/paper/2026/hash/5edb57c05c81d04beb716ef1d542fe9e-Abstract-Conference.html.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report, 2025. URL https://arxiv.org/abs/2511.21631.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, 2025a. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. In *Advances in Neural Information Processing Systems*, volume 38, pp. 33383–33407, 2025b. doi: 10.52202/085713-1122. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/300ccb2187dedd4edcc07f7e76d8e553-Abstract-Conference.html.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-Time Action Conditioning for Efficient Real-Time Chunking, 2025c. URL https://arxiv.org/abs/2512.05964.

Kanzhi Cheng, Qiushi Sun, Yougang Chu, Fangzhi Xu, Yantao Li, Jianbing Zhang, and Zhiyong Wu. SeeClick: Harnessing GUI Grounding for Advanced Visual GUI Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9313–9332, 2024. doi: 10.18653/v1/2024.acl-long.505. URL https://aclanthology.org/2024.acl-long.505/.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://roboticsproceedings.org/rss19/p026.html.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a Generalist Agent for the Web. In *Advances in Neural Information Processing Systems*, volume 36, pp. 28091–28114, 2023. doi: 10.52202/075280-1220. URL https://papers.neurips.cc/paper_files/paper/2023/hash/5950bf290a1570ea401bf98882128160-Abstract-Datasets_and_Benchmarks.html.

Zihan Dong, Rui Qian, Qishi Zhan, Dongshen Peng, Kaixin Li, and Yu Li. Why Are GUI Agents Correct but Late? Decode on the Decision-Time Critical Path, Tested with Pre-Compiled Policy Trees, 2026. URL https://arxiv.org/abs/2607.28399.

Boyu Gou, Demi Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the Digital World as Humans Do: Universal Visual Grounding for GUI Agents. In *International Conference on Learning Representations*, pp. 30851–30883, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4ca0e369689dadb25a5345ba9755ad6f-Abstract-Conference.html.

Coleman Hooper, Minwoo Kang, Suhong Moon, Nicholas Lee, Eric Wen, John Wawrzynek, Michael W. Mahoney, Yakun Sophia Shao, Amir Gholami, and Kurt Keutzer. Speculative Interaction Agents: Building Real-Time Agents with Asynchronous I/O and Speculative Tool Calling, 2026. URL https://arxiv.org/abs/2605.13360.

Siyuan Hu, Kevin Qinghong Lin, and Mike Zheng Shou. ShowUI-π: Flow-based generative models as gui dexterous hands. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8130–8140, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Hu_ShowUI-p_Flow-based_Generative_Models_as_GUI_Dexterous_Hands_CVPR_2026_paper.html.

Haoyu Huang, Jinfa Huang, Zhongwei Wan, Xiawu Zheng, Rongrong Ji, and Jiebo Luo. SpecEyes: Accelerating Agentic Multimodal LLMs via Speculative Perception and Planning, 2026. URL https://arxiv.org/abs/2603.23483.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 881–905, 2024. doi: 10.18653/v1/2024.acl-long.50. URL https://aclanthology.org/2024.acl-long.50/.

Bojie Li and Noah Shi. Agent-Computer Observation Interfaces Enable Dynamic Computer Use, 2026. URL https://arxiv.org/abs/2606.29472.

Kaixin Li, Ziyang Meng, Hongzhan Lin, Ziyang Luo, Yuchen Tian, Jing Ma, Zhiyong Huang, and Tat-Seng Chua. ScreenSpot-Pro: GUI Grounding for Professional High-Resolution Computer Use. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 8778–8786, 2025. doi: 10.1145/3746027.3755688. URL https://doi.org/10.1145/3746027.3755688.

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. ShowUI: One Vision-Language-Action Model for GUI Visual Agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19498–19508, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Lin_ShowUI_One_Vision-Language-Action_Model_for_GUI_Visual_Agent_CVPR_2025_paper.html.

Enqi Liu, Liyuan Pan, Zhi Gao, Yan Yang, Chenrui Shi, Yang Liu, Jingrong Wu, and Qing Li. Benchmarking and Improving GUI Agents in High-Dynamic Environments, 2026a. URL https://arxiv.org/abs/2604.25380.

Yufeng Liu, Hang Yu, Juntu Zhao, Bocheng Li, Di Zhang, Mingzhu Li, Wenxuan Wu, Yingdong Hu, Junyuan Xie, Junliang Guo, Dequan Wang, and Yang Gao. Learning Native Continuation for Action Chunking Flow Policies. In *Proceedings of Robotics: Science and Systems*, 2026b. doi: 10.15607/RSS.2026.XXII.058. URL https://roboticsproceedings.org/rss22/p058.html.

Sarthak Mehrotra, Sairam V C Rebbapragada, Mani Hemanth Reddy Bonthu, and Vineeth N Balasubramanian. iSHIFT: Lightweight Slow-Fast GUI Agent with Adaptive Perception, 2025. URL https://arxiv.org/abs/2512.22009.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: Pioneering Automated GUI Interaction with Native Agents, 2025. URL https://arxiv.org/abs/2501.12326.

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A Dynamic Benchmarking Environment for Autonomous Agents. In *International Conference on Learning Representations*,

pp. 406–441, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/01a83bc2f2732a58e6aa731e659e7101-Abstract-Conference.html.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. AndroidInTheWild: A Large-Scale Dataset For Android Device Control. In *Advances in Neural Information Processing Systems*, volume 36, pp. 59708–59728, 2023. doi: 10.52202/075280-2609. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/bbbb6308b402fe909c39dd29950c32e0-Abstract-Datasets_and_Benchmarks.html.

Yifan Sui, Han Zhao, Rui Ma, Zhiyuan He, Hao Wang, Jianxun Li, Kaiqiang Xu, Kai Chen, and Yuqing Yang. Parallelizing Tool Execution and LLM Generation for Low-Latency Agent Serving, 2026. URL https://arxiv.org/abs/2603.18897.

Venus Team, Zhuohan Cai, Haoxing Chen, Jiaxuan Chen, Weizhi Chen, Changlong Gao, Zhangxuan Gu, Yuan Guo, Yusong Hu, Jianrong Jiang, Jianguo Li, Runze Li, Jinzhen Lin, Zhenyu Ma, Changhua Meng, Han Peng, Xinyu Qiu, Shuheng Shen, Zhongyi Shui, Weiqiang Wang, Ming Wen, Zhuoer Xu, Hang Yan, Kaiwen Yang, Ruilin Yao, Nanjun Yu, Zhengwen Zeng, Lianrui Zhang, Yunzhu Zhang, Zhe Zhao, and Beitong Zhou. UI-Venus-2 Technical Report, 2026a. URL https://arxiv.org/abs/2609.00028.

Venus Team, Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, Beitong Zhou, Xingran Zhou, Weizhi Chen, Sunhao Dai, Jingya Dou, Yichen Gong, Yuan Guo, Zhenlin Guo, Feng Li, Qian Li, Jinzhen Lin, Yuqi Zhou, Linchao Zhu, Liang Chen, Zhenyu Guo, Changhua Meng, and Weiqiang Wang. UI-Venus-1.5 Technical Report, 2026b. URL https://arxiv.org/abs/2602.09082.

Pengcheng Wang, Kaiwen Hong, Chensheng Peng, Katherine Driggs-Campbell, Masayoshi Tomizuka, Chenfeng Xu, and Chen Tang. DiscreteRTC: Discrete Diffusion Policies are Natural Asynchronous Executors, 2026. URL https://arxiv.org/abs/2604.25050.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. In *Advances in Neural Information Processing Systems*, volume 37, pp. 52040–52094, 2024. doi: 10.52202/079017-1650. URL https://proceedings.nips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Jiashu Yao, Heyan Huang, Daiqing Wu, Wangke Chen, Huaxi Ai, Haoyu Wen, Zeming Liu, and Yuhang Guo. Benchmarking Living-Screen-Native GUI Agents on Short-Video Platforms, 2026. URL https://arxiv.org/abs/2606.04701.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://roboticsproceedings.org/rss19/p016.html.

Hongbin Zhong, Fazle Faisal, Luis França, Tanakorn Leesatapornwongsa, Adriana Szekeres, Kexin Rong, and Suman Nath. ActionEngine: From Reactive to Programmatic GUI Agents via State Machine Memory, 2026. URL https://arxiv.org/abs/2602.20502.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, and Steven Hoi. MAI-UI Technical Report: Real-World Centric Foundation GUI Agents, 2025. URL https://arxiv.org/abs/2512.22047.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *International Conference on Learning Representations*, pp. 15585–15606, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

A MATCHED RELEASE: TIMING, ACTIVITY, AND TASK UTILITY

Relation to agent serving. PASTE overlaps predicted tool invocations with model generation (Sui et al., 2026); Speculative Interaction Agents decouple reasoning from asynchronous input and tool responses (Hooper et al., 2026). SpecEyes uses a lightweight speculative planner to shorten multimodal tool chains (Huang et al., 2026). These systems change the scheduling or amount of computation. Our matched comparison changes when an already generated legal action becomes available, while the target-aware experiments separately test whether prepared work remains usable.

Matched protocol. Tetris supplies a separate test of action release timing. Six gravity rates, 800/650/500/370/250/160 ms per row (L1–L6), shorten placement opportunities as speed increases. The application records locks, line clears, and score; a hard drop can cause a lock without improving the board. We compare buffered and incremental release for Qwen3.8-27B-SPOT at temperature zero with thinking disabled. Ten seeds at each gravity level produce 60 matched pairs and 120 episodes. Two batches contribute five disjoint seeds each under the same runner, prompt, decoding settings, budgets, and metric schema. Requested and applied seeds are checked. Paired arms run sequentially on the same emulator with balanced AB/BA ordering.

Each episode permits up to 40 runner steps. Both arms use a maximum of 12 clicks per chunk, a 220 ms injection pace, and a 380 ms spawn-settle gate. Neither arm sets the output-token limit client-side; both use the same service default. Overlap, clamping, and pipelining are disabled. We measure from the first decision's request start to the end of its first successful injection (`inject_end`), alongside episode score. All 120 recorded episodes have a first injection and a successful run status, with no episode-level reruns. The ten seeds recur across levels; aggregate uncertainty must therefore treat the seed as a cluster rather than treating 60 pairs as independent task draws.

Across 60 paired Tetris instances, incremental release reduces mean first-decision request-to-first-injection latency from 6699.37 to 2883.94 ms, a paired mean difference of -3815.44 ms and a 56.95% reduction relative to the buffered mean. Streaming injects earlier in 57 of 60 pairs. The mean difference is negative at every gravity level (Figure 5). A post hoc paired bootstrap over the ten seed clusters gives a 95% percentile interval of $[-4741.19, -2901.39]$ ms (20,000 resamples). The measured advantage concerns the first injection of the first decision, rather than episode completion time or every action in a plan.

The same comparison has a severe task-score floor. Of 120 episodes, 115 score zero. Buffered execution has two nonzero scores, 20 and 25; streaming has three, 5, 5, and 20. The respective means are 0.75 and 0.50, yielding a paired mean difference of -0.25 (95% seed-cluster bootstrap interval $[-1.00, 0.33]$). These outcomes provide no observed score improvement from release acceleration. Sparse nonzero scores also limit the experiment's ability to distinguish modest utility effects. They do not identify whether the dominant failure is action selection, target mismatch, or accumulated board errors.

More activity without more task progress. The release change also increases full-episode interaction. Mean recorded turns rise from 11.58 to 16.23, and successful injections before terminal from 120.40 to 175.17. Mean survival is 104.65 versus 105.11 s (paired difference 0.47 s; 95% interval $[-6.26, 6.82]$). Thus the increase in activity accompanies a near-zero score regime. The archived summaries report no parsing failures, empty-action turns, or token-cap terminations. Appendix A.1 gives the paired intervals and terminal-event accounting.

The effect persists across the two collection batches: the mean first-injection differences are -4.33 and -3.30 s. Omitting any one seed cluster leaves the pooled difference between -4.11 and -3.53 s. Per-episode injection rates rise from 1.08 to 1.65 successful preterminal injections per second. The increase in activity therefore persists after normalizing by survival time (Appendix A.2).

This result supplies a controlled reason to evaluate the commitment itself. Faster release changes when an action arrives; it does not test whether a future action still refers to the visible target. The Quiz matrix makes that second factor explicit while holding release mode fixed.

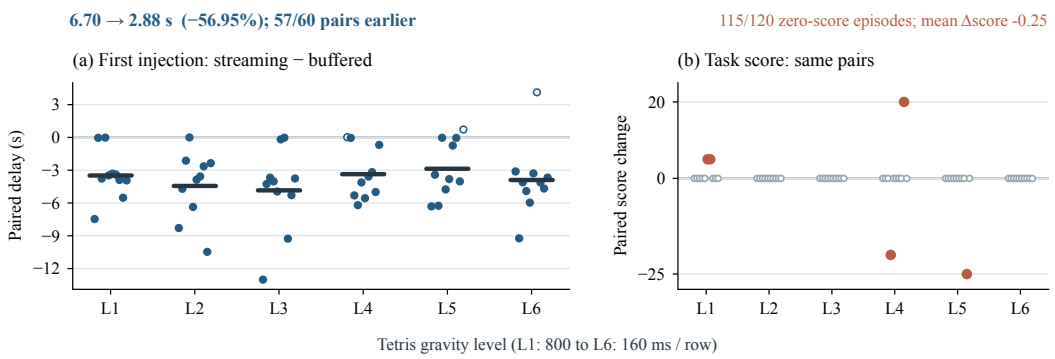

Figure 5: **Matched release comparison: earlier injection without an observed score gain.** Each point is streaming minus buffered for one seed at one gravity level; horizontal bars in the left panel show level means. Left: first-decision request-to-first-injection delay. Right: episode score. Ten seeds are reused across six levels, giving 60 pairs and 120 episodes; the seed is the cluster for pooled inference. Values use per-episode summaries cross-checked against run manifests; raw events were not independently replayed.

A.1 Full-episode activity

Table 2 extends the first-injection result to whole episodes. Buffered and streaming runs record 695 and 974 turns, respectively. Their raw successful-injection counts are 7,240 and 10,529; 16 and 19 are recorded after the terminal event and are excluded from the table's interaction counts. The remaining totals are 7,224 and 10,510. All 120 survival records end in telemetry game-over, with no censored survival observations. The summary fields report zero parse failures, empty-action turns, and token-cap terminations across both arms. These checks establish the recorded output and execution activity, without establishing that the selected actions were appropriate for the board.

The archived deadline-hit field applies to a restricted gravity-lock subset, and its per-episode eligible/ambiguous counts are unavailable in the recovered files. We therefore do not pool that field into an overall validity rate. The field historically called causal control uses timing-based cause assignment; it is excluded from causal mechanism claims. The table uses directly counted episode outcomes and reports post hoc uncertainty over the ten paired seed clusters.

Table 2: Full-episode activity and utility in the matched Tetris release comparison (60 episodes per arm). Entries are episode means; differences are streaming minus buffered. Intervals are post hoc paired seed-cluster bootstrap intervals (10 seeds, 20,000 resamples). Injection counts exclude events recorded after terminal. Values use archived per-episode summaries; raw events were not independently replayed.

Measure	Buffered	Streaming	Difference	95% interval
Recorded turns	11.58	16.23	+4.65	[+3.95, +5.30]
Successful injections before terminal	120.40	175.17	+54.77	[+46.85, +61.88]
Survival (s)	104.65	105.11	+0.47	[-6.26, +6.82]
Score	0.75	0.50	-0.25	[-1.00, +0.33]
Lines cleared	0.03	0.05	+0.02	[-0.03, +0.07]

A.2 Gravity, collection batches, and seed sensitivity

Table 3 separates the matched comparison by gravity. The mean first-injection difference is negative at every level, and the mean number of successful preterminal injections increases at every level. Survival behaves differently: its paired mean decreases at L1–L3 and increases at L4–L6. Averaging survival across levels therefore hides opposing descriptive patterns. These per-level analyses are post hoc; their intervals quantify uncertainty without supporting a family of six significance claims.

Table 3: Gravity-stratified matched release results. Each row has 10 seed pairs. Latency concerns the first injection of the first decision; differences are streaming minus buffered. Intervals are post hoc paired seed bootstrap intervals. Nonzero counts are buffered, streaming. Activity excludes injections after terminal.

Level	Buffered (s)	Stream (s)	Difference [95% interval]	Δ actions	Δ survival (s)	Nonzero score
1	6.03	2.55	-3.48 [-4.78, -2.18]	+62.1	-15.44	0/10, 2/10
2	7.25	2.81	-4.44 [-6.34, -2.71]	+64.1	-11.63	0/10, 0/10
3	7.44	2.60	-4.84 [-7.21, -2.72]	+56.5	-3.55	0/10, 0/10
4	6.23	2.87	-3.36 [-4.69, -1.95]	+65.9	+11.48	1/10, 1/10
5	6.07	3.20	-2.87 [-4.39, -1.33]	+46.2	+12.49	1/10, 0/10
6	7.17	3.27	-3.90 [-5.66, -1.74]	+33.8	+9.45	0/10, 0/10

Interaction rate. To distinguish more interaction from simply surviving longer, we divide each episode's successful preterminal injections by its survival time. The average episode rate rises from 1.08 to 1.65 injections per second, a paired increase of 0.57 (95% seed-cluster interval $[0.53, 0.61]$). This is an average of episode rates, rather than the ratio of total actions to total survival. Figure 6 places rate, first-injection delay, and survival on separate axes. All three quantities concern execution activity; score remains the task endpoint.

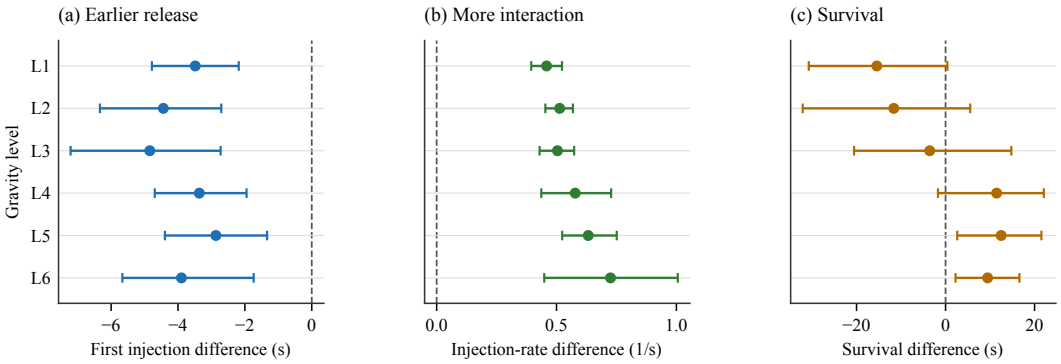

Figure 6: **Release acceleration changes activity more consistently than survival.** Points are paired streaming-minus-buffered means at each gravity level; bars are post hoc 95% paired bootstrap intervals over ten seeds. The first panel measures first-decision injection delay, the second the per-episode preterminal injection rate, and the third survival. Each panel retains its own units and zero reference.

Collection-batch consistency. The two batches contain five disjoint seeds each and 30 pairs each. Their mean first-injection differences are -4.33 s (95% seed-cluster interval $[-5.51, -3.23]$) and -3.30 s ($[-4.72, -2.06]$); 29/30 and 28/30 pairs, respectively, favor earlier streaming injection. Each bootstrap resamples the five seeds within that batch while retaining all six gravity levels. This check shows that the pooled latency direction occurs in both collections; it does not estimate a causal effect of collection time.

Influence of individual seeds. Removing one seed cluster at a time retains its six levels and paired arms as a unit. Across the ten omissions, the pooled first-injection difference ranges from -4.11 to -3.53 s, the injection-count difference from $+53.26$ to $+57.09$, and the injection-rate difference from $+0.55$ to $+0.58$ per second. The survival difference ranges from -1.26 to $+2.59$ s. Thus the observed latency and activity directions are insensitive to any single seed, whereas the small pooled survival difference changes sign. This sensitivity analysis uses the same episodes and supplies no additional independent replication.

B TurboQuiz: Target Correspondence, Task Content, and Costs

The 80 enrolled episodes each resolve all 30 allocated questions and finish without infrastructure or model-call errors. This section decomposes the outcome along the path from a model plan to an environment answer. All tables use the same episodes as Table 4.

Table 4: TurboQuiz task outcomes and execution cost. Each row contains ten episodes with 30 allocated questions each. Correct, model calls, and active wall time are episode means. All 80 episodes are complete; expired and unanswered questions remain in the fixed denominator.

TTL	Policy	n	Correct / 30	Calls	Wall (s)
7.5 s	A: Single step	10	21.80	29.90	94.5
7.5 s	B: Two step window	10	21.50	15.00	62.8
7.5 s	C: Single step + feedback	10	22.20	29.90	125.3
7.5 s	D: Two step + feedback	10	21.90	15.00	93.2
2.5 s	A: Single step	10	3.60	14.90	46.6
2.5 s	B: Two step window	10	4.40	10.00	40.3
2.5 s	C: Single step + feedback	10	0.00	18.80	78.1
2.5 s	D: Two step + feedback	10	9.60	14.90	63.5

Factorial effects and seed stability. Table 5 reports all designated contrasts for both deadlines. The primary fast D−A effect is positive for all ten seed pairs. Its leave-one-seed-out means span 18.5–21.9 percentage points. Intervals resample paired seeds; individual questions and injections are used for mechanism counts, not as additional independent experimental units.

Table 5: Paired policy contrasts in TurboQuiz. Differences and 95% intervals are in percentage points of correct/30. Each contrast uses ten seed pairs; intervals use the common 20,000-resample post hoc analysis. The fast D−A contrast is the designated primary comparison; the other contrasts describe the factorial effects.

Clock (s)	Contrast	Difference	95% interval
7.5	Full policy: D−A	+0.3	[-3.3, +4.3]
7.5	Window: B−A	-1.0	[-5.3, +4.0]
7.5	Feedback in window: D−B	+1.3	[-2.3, +5.3]
7.5	Single-step check: C−A	+1.3	[-1.0, +4.0]
7.5	Interaction	+0.0	[-5.0, +4.7]
2.5	Full policy: D−A	+20.0	[+14.0, +25.7]
2.5	Window: B−A	+2.7	[+0.0, +5.3]
2.5	Feedback in window: D−B	+17.3	[+12.0, +22.3]
2.5	Single-step check: C−A	-12.0	[-16.3, -8.3]
2.5	Interaction	+29.3	[+24.3, +34.0]

Intended targets and credited answers. A tap may solve the question seen by the model but arrive on a different question. Conversely, a stale tap can receive task credit on an unintended question. Table 6 retains every question outcome, and Figure 7 separates target identity from credited correctness. In fast A, 107 of 146 injected answers are correct for their source target, yet all reach another target; only 25 of those 107 receive task credit. In fast D, all 96 source-correct answers that reach their intended question receive credit. The 40 matched-but-wrong answers identify an answer-selection error after successful target routing.

Actual window length and checking time. Table 7 reports the fraction of parsed plans that actually contain two actions, canceled actions, and the costs of model calls and checks. At the fast deadline, D prepares 292 actions, cancels 151, and injects 141. C cancels all 188 actions it prepares. The online check includes screenshot acquisition and OCR; source OCR overlaps model inference, so its full duration is not added again to the execution critical path.

Table 6: Question outcomes and execution correspondence. Each row contains 10 episodes and 300 allocated questions. Match/label uses injections with known target identity; all 1,679 injections have labels. Unans. counts unresolved questions. Calls/correct is a pooled ratio and is undefined when no question is answered correctly.

TTL	Arm	Correct	Wrong	Expired	Unans.	Match/label	Calls/correct
7.5	A	218	81	1	0	298/299	1.37
7.5	B	215	83	2	0	294/298	0.70
7.5	C	222	77	1	0	298/299	1.35
7.5	D	219	79	2	0	298/298	0.68
2.5	A	36	110	154	0	0/146	4.14
2.5	B	44	154	102	0	0/198	2.27
2.5	C	0	0	300	0	0/0	–
2.5	D	96	45	159	0	136/141	1.55

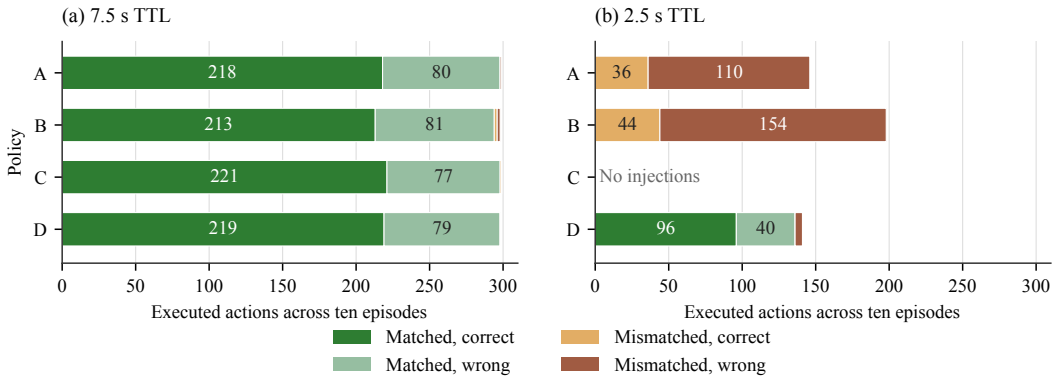

Figure 7: **Target correspondence and task credit are distinct.** Counts pool ten episodes per policy at each TTL. Green segments reach the intended target; amber/brown segments reach another target. The legend crosses target identity with credited correctness. All Quiz injections have an offline target label; the fast C policy makes no injection.

Table 7: Actual commitments and computation in TurboQuiz. Counts pool ten episodes per row. Two-action plans are divided by valid parsed plans; canceled counts refer to actions. Model time is per call, and check time includes the fresh screenshot and OCR. Source OCR runs beside inference and is accounted for separately.

TTL	Arm	Two-action plans	Canceled	Checks	Model (s)	Check (s)
7.5	A	0/299	0	0	2.41	–
7.5	B	148/150	0	0	2.71	–
7.5	C	0/299	0	299	2.37	1.08
7.5	D	150/150	2	298	2.56	1.08
2.5	A	0/149	3	0	2.39	–
2.5	B	99/100	1	0	2.55	–
2.5	C	0/188	188	181	2.44	1.07
2.5	D	143/149	151	143	2.49	1.07

Task content and tokens. Both question types improve under fast D relative to A: maximum-number questions increase from 21 to 65 correct out of 150, and arithmetic questions from 15 to 31 out of 150. The gains therefore occur in both retained task categories. Table 8 also reports total prompt-plus-completion tokens per correct answer. This pooled efficiency metric keeps unsuccessful calls and unanswered questions in the experiment; it is undefined for C at the fast deadline because that policy completes no question correctly.

Table 8: Task content and token efficiency in TurboQuiz. Every row allocates 150 maximum-number and 150 arithmetic questions. Tokens/correct pools recorded prompt and completion tokens; it is undefined when the policy completes no question correctly.

TTL	Arm	Maximum correct	Arithmetic correct	Tokens/correct
7.5	A	149/150	69/150	3,956
7.5	B	145/150	70/150	2,022
7.5	C	149/150	73/150	3,884
7.5	D	147/150	72/150	1,985
2.5	A	21/150	15/150	11,937
2.5	B	22/150	22/150	6,586
2.5	C	0/150	0/150	–
2.5	D	65/150	31/150	4,499

A logged prefix skip. In seed 925117, the third decision observes $36 + 93$ as CURRENT and maximum$(49, 48, 24, 92)$ as NEXT. The model prepares answer indices 0 and 3. Inference takes 2.26 s, followed by a 1.11 s fresh observation and check. The new screen matches the second target, so the executor cancels the first answer and clicks the large answer button for 92. The ledger records a correct answer to the intended question. Source-image age at injection lies in [3.43,3.95] s; this interval reflects screenshot acquisition, not the future target's remaining lifetime. This is the first successful prefix skip in the first formal fast-D seed, selected to illustrate the logged routing path.

C Dynamic Ticket Queue: Preserving Decisions Across Motion

TurboQuiz exposes a future question before it becomes actionable. Dynamic Ticket Queue presents concurrently actionable targets whose locations change. The task tests whether feedback can retain a routing decision when its original coordinate becomes stale.

Instrument and information boundary. Ten batches of three tickets give a fixed denominator of 30. Each ticket has a unique visible identifier, issue category, and status. The screen states the routing rule: NETWORK to OPS, PAYMENT to BILLING, and ACCESS to SECURITY. The model selects the corresponding button within an OPEN ticket. Answered tickets remain visible and disabled. A batch advances after all tickets resolve or after 15 s. A seeded derangement permutes the three cards every 7.5 or 2.5 s while preserving ticket identities. The instrument runs in an Android browser. Source and fresh-screen readers inspect pixels; the scoring ledger is used offline.

From a coordinate to a target-relative commitment. The model returns up to one or two normalized points for distinct visible OPEN tickets, in top-to-bottom order. There is no future-batch preview. Source OCR binds a point to the ticket identifier, issue, state, and card bounds. Original-coordinate checking tests the identity at that same point; target rebinding searches the fresh screenshot for the same identifier with unchanged issue and OPEN state. Missing or ambiguous matches cancel the remaining window. Both rules preserve order and revoke the tail on failure; neither skips a prefix.

Let the model point, converted to pixels, be (x, y) in source bounds (x_0, y_0, x_1, y_1). The rebinding rule retains its relative position,

$$(u, v) = \left(\frac{x - x_0}{x_1 - x_0}, \frac{y - y_0}{y_1 - y_0} \right), \qquad (x', y') = (x'_0 + u(x'_1 - x'_0),\ y'_0 + v(y'_1 - y'_0)), \tag{3}$$

where primed bounds belong to the matched live card. The resulting point is rounded back to normalized coordinates and checked to lie inside that card. The transform uses identity and geometry, retaining the model's within-card button choice. It has no access to the routing rule or the correct answer. Here the internal card layout is fixed as cards move.

Six-arm paired comparison. A/B use original coordinates with $K = 1/2$; C/D add original-coordinate checks; E/F use target rebinding. The 120-episode matrix uses ten fresh seeds, disjoint from the earlier cancellation experiment and development episodes, crossed with both refresh rates. Each seed–refresh block assigns all six arms to one device in rotated order. Four KVM emulators run four workers. All arms use Qwen3.8-27B-SPOT, temperature zero, thinking disabled, 768 output tokens, 80 calls, 240 active seconds, a 90-second call timeout, and zero SDK retries. Inter-action gaps are 700 ms and cooldowns 150 ms. Source OCR overlaps inference; checks, model calls, and injections all occur on the live clock. Every enrolled episode completes, without a failed attempt or rerun. Fast F−A is the designated primary contrast; the remaining contrasts separate relocation and window effects.

Table 9: Ticket Queue with target relocation: ten paired seeds per row and 30 allocated tickets per episode. A/B execute original coordinates, C/D check original coordinates, and E/F rebind the selected point to its ticket. Odd/even letters request one/two actions. All 120 episodes resolve all 30 tickets; outcomes and costs are episode means.

Refresh (s)	Arm	Correct/30	Wrong	Expired	Calls	Wall (s)
7.5	A	29.5	0.2	0.3	36.9	96.2
7.5	B	29.6	0.3	0.1	20.6	64.7
7.5	C	23.1	2.9	4.0	34.2	132.8
7.5	D	28.4	0.7	0.9	28.2	133.0
7.5	E	25.3	2.7	2.0	32.0	122.8
7.5	F	28.1	0.9	1.0	19.9	105.3
2.5	A	15.9	8.3	5.8	48.8	130.4
2.5	B	7.8	14.0	8.2	49.2	140.4
2.5	C	1.4	2.6	26.0	39.6	151.7
2.5	D	1.1	1.5	27.4	36.5	151.8
2.5	E	21.0	4.8	4.2	35.9	139.7
2.5	F	21.5	4.6	3.9	22.9	121.7

Table 10: Paired relocation contrasts in percentage points of correct/30. Ten seeds are resampled 20,000 times per contrast with the same post hoc bootstrap as the Quiz analysis. Fast F−A is the designated primary comparison.

Refresh (s)	Contrast	Difference	95% interval
7.5	F−A	-4.7	[-9.7, -0.7]
7.5	F−D	-1.0	[-4.3, +2.7]
7.5	F−B	-5.0	[-10.3, -0.3]
7.5	E−C	+7.3	[+1.0, +14.3]
7.5	E−A	-14.0	[-17.7, -10.3]
7.5	F−E	+9.3	[+2.7, +15.0]
7.5	$(F-B)-(E-A)$	+9.0	[+2.0, +15.3]
2.5	F−A	+18.7	[+7.3, +30.3]
2.5	F−D	+68.0	[+57.3, +78.0]
2.5	F−B	+45.7	[+34.7, +56.3]
2.5	E−C	+65.3	[+58.0, +72.0]
2.5	E−A	+17.0	[+5.3, +29.7]
2.5	F−E	+1.7	[-11.3, +14.7]
2.5	$(F-B)-(E-A)$	+28.7	[+10.3, +47.7]

Seed stability and task content. Fast F−A improves nine of ten paired seeds; omitting any one seed leaves a mean gain of 14.8–21.9 percentage points. Each routing category contributes 100

allocated tickets across ten episodes. Correct outcomes rise from 55 to 71 for ACCESS, 46 to 74 for NETWORK, and 58 to 70 for PAYMENT. Gains therefore occur across all three visible routing rules. Total prompt-plus-completion tokens per correct ticket fall from 8,574 to 2,985, counting every call. These descriptive counts use the same episodes as the paired analysis.

Action retention and cost. At fast refresh, F makes 370 checks: 299 identify a moved compatible target, 29 match the original point, 21 reject a changed state, and 21 lack a unique identity. It injects 325 actions, including 297 relocated points, and cancels 49 actions. A successful check can still be followed by an episode cutoff, explaining why successful checks exceed injections. Mean model time is 2.24 s per call and a fresh screenshot/check takes 1.21 s. Source OCR averages 0.76 s and overlaps inference. At slower refresh, fewer points need relocation and checking adds cost to already viable direct execution. Table 11 reports the realized plans and costs at both clocks.

Table 11: Prepared actions and execution cost in the relocation matrix. Plan counts and cancellations pool ten episodes per row. Moved counts injected actions whose coordinates changed. Model and check times are per operation; checks include fresh screenshot acquisition.

Refresh	Arm	Two-action plans	Canceled	Moved	Model (s)	Check (s)
7.5	A	0/360	1	0	1.94	–
7.5	B	100/206	0	0	2.11	–
7.5	C	0/338	55	0	1.94	1.29
7.5	D	105/282	88	0	2.05	1.25
7.5	E	0/310	4	24	1.94	1.26
7.5	F	105/196	4	87	2.13	1.36
2.5	A	0/485	4	0	2.02	–
2.5	B	102/491	8	0	2.05	–
2.5	C	0/391	349	0	1.91	1.32
2.5	D	358/362	691	0	2.11	1.30
2.5	E	0/358	23	275	1.92	1.29
2.5	F	145/229	49	297	2.24	1.21

Target correspondence and receipt coverage. An answer receipt is assigned to an injection only if it occurs from ADB start through 100 ms after return, before the next injection, and within 8 pixels of the actual executed point. Multiple candidates remain unknown. Source binding uses the planned point; receipt matching uses the relocated point when applicable. Strict ADB-interval coverage is reported separately because receipt time is not physical tap time. In fast F, 261 of 325 injections have target labels: 192 reach the intended ticket and are correct, while 69 reach another ticket, including 23 credited routes. The other 64 remain unknown. All 261 labeled receipts use the allowance. Thus rebinding can preserve a decision while the residual check-to-tap interval still permits mismatch. Episode scores are independently replayed from the full ledger and do not depend on injection attribution.

A logged relocation. In the first fast-F seed, the first model response selects normalized point $(794, 451)$ on an ACCESS ticket in the first row. After a 2.40 s model call, the fresh screen places that identifier in the second row. The executor retains its local button position and injects $(794, 672)$. The ledger records a correct route for the intended ticket. This first enrolled action illustrates the coordinate transform; it is not a separate estimate of its success rate.

C.1 Earlier cancellation comparison

The earlier 80-episode matrix crosses $K = 1/2$ with no checks or original-coordinate checks, using ten other paired seeds at each refresh rate. It uses the same task, model, budgets, and timing settings. All four policies run on the same emulator within each rotated seed–refresh block. This experiment contains no rebinding arm. Its results remain separate from the six-arm matrix; the new comparison retains these policies as controls on fresh seeds.

Table 12: Receipt attribution for the relocation matrix. Labeled uses the fixed 100 ms allowance and 8 pixel tolerance at the actual executed coordinate; Strict uses the ADB interval alone. Mismatch is counted within labeled injections, and unknowns remain separate. Task outcomes use the full ledger.

Refresh	Arm	Injections	Strict	Labeled	Unknown	Mismatch
7.5	A	359	0	297	62	3
7.5	B	306	8	299	7	1
7.5	C	283	0	260	23	35
7.5	D	299	0	291	8	8
7.5	E	306	3	280	26	37
7.5	F	297	2	290	7	14
2.5	A	480	3	242	238	135
2.5	B	585	4	218	367	190
2.5	C	42	0	40	2	38
2.5	D	29	0	26	3	22
2.5	E	335	0	258	77	75
2.5	F	325	0	261	64	69

Task outcomes and plan utilization. Table 13 reports all eight cells. At the slower refresh, the unguarded window B completes 29.7 tickets with 20.0 calls, versus A's 28.9 tickets with 34.9 calls. At the faster refresh, both windowing and checking reduce completion relative to A. The fast D plans contain 376 two-action and four single-action outputs. Of those parsed plans, 349 cancel both actions, 25 execute the first and cancel the second, two execute both, and four cancel their single action. This yields 727 cancellations and 29 injections. The ledger records nine correct, 19 wrong, and 272 expired tickets, with no unresolved ticket.

Table 13: Dynamic Ticket Queue results, ten episodes per row and 30 allocated tickets per episode. Entries are episode means. Refresh controls position changes; every ticket batch has a 15-second lifetime. Issued and canceled count individual planned actions.

Refresh (s)	Arm	Correct/30	Calls	Wall (s)	Issued	Canceled
7.5	A	28.9	34.9	92.2	33.7	0.1
7.5	B	29.7	20.0	60.1	30.0	0.0
7.5	C	25.6	35.1	130.7	28.7	5.7
7.5	D	28.6	29.4	128.9	29.3	9.4
2.5	A	16.2	45.5	122.9	44.3	0.5
2.5	B	9.1	52.3	140.0	61.7	0.6
2.5	C	1.7	41.8	151.4	5.4	36.2
2.5	D	0.9	38.7	153.0	2.9	72.7

Table 14: Paired policy contrasts in Ticket Queue. Differences and 95% intervals are in percentage points of correct/30. Each contrast uses ten seed pairs; intervals use the common 20,000-resample post hoc analysis. The fast D−A contrast is the designated primary comparison; the other contrasts describe the factorial effects.

Clock (s)	Contrast	Difference	95% interval
7.5	Full policy: D−A	-1.0	[-5.7, +6.0]
7.5	Window: B−A	+2.7	[-1.0, +9.0]
7.5	Feedback in window: D−B	-3.7	[-5.7, -1.7]
7.5	Single-step check: C−A	-11.0	[-17.7, -3.7]
7.5	Interaction	+7.3	[+0.7, +13.3]
2.5	Full policy: D−A	-51.0	[-62.7, -37.7]
2.5	Window: B−A	-23.7	[-35.7, -10.7]
2.5	Feedback in window: D−B	-27.3	[-31.7, -22.7]
2.5	Single-step check: C−A	-48.3	[-57.7, -36.7]
2.5	Interaction	+21.0	[+9.7, +31.0]

Checking costs and receipt coverage. Fast D's model calls average 2.10 s and its fresh observation/check operations average 1.14 s. Its 399 checks comprise 368 target changes, 30 matches, and one unreadable live target. Source OCR averages 0.66 s and runs concurrently with model inference. Target attribution labels 28 of the 29 injections, including six matched/correct, three mismatched/correct, and 19 mismatched/wrong; one injection remains unknown. None of these 29 receipts lies inside the strict ADB interval, so their identity attribution uses the specified receipt allowance. Table 15 reports both coverage levels for every policy and bounds the contribution of unknowns. The correct/30 comparisons come from the complete event ledger and do not depend on attributing every injection.

Table 15: Ticket Queue injection attribution. Strict counts receipts inside the ADB interval. Labeled additionally allows the fixed 100 ms receipt delay and 8 pixel coordinate tolerance. The last column bounds the mismatch percentage over all injections by treating every unknown as a match or a mismatch. Task scores use the full event ledger independently.

Refresh	Arm	Taps	Strict	Labeled	Unknown	Mismatch	Bounds (%)
7.5	A	337	0	292	45	3	[0.9, 14.2]
7.5	B	300	13	300	0	0	[0.0, 0.0]
7.5	C	287	0	271	16	20	[7.0, 12.5]
7.5	D	293	0	292	1	5	[1.7, 2.0]
2.5	A	443	6	244	199	121	[27.3, 72.2]
2.5	B	617	6	228	389	199	[32.3, 95.3]
2.5	C	54	1	53	1	49	[90.7, 92.6]
2.5	D	29	0	28	1	22	[75.9, 79.3]

D Window Construction Across Models

The execution contract requires prepared actions for more than one target. We examine this prerequisite through a four-model paired matrix, controlled screenshot prompts, and complete local episodes. These stages retain their own protocols and denominators: 320 formal episodes, 280 screenshot calls, and 38 local integration attempts.

D.1 Requested and realized window lengths

Qwen3-VL-8B-Instruct (Bai et al., 2025), UI-Venus-1.5-8B (Venus Team et al., 2026b), MAI-UI-8B (Zhou et al., 2025), and UI-Venus-2-9B (Venus Team et al., 2026a) each run the original four-arm Quiz matrix on the same ten paired seeds and two TTLs. Models are served through direct vLLM endpoints; each matrix holds its endpoint and decoding settings fixed across arms. The original ordered-array prompt, 768-token limit, temperature zero, disabled thinking, 40-call budget, and parser are unchanged. The parser accepts one or two answers when $K = 2$ to accommodate a final question without NEXT. All 320 enrolled episodes are retained: 299 reach the terminal state and 21 MAI episodes reach the call budget. Unanswered and unpresented questions remain in the 30-question denominator.

All 3,997 valid B/D plans contain one action, across all 160 $K = 2$ episodes. By comparison, the original 27B matrix contains 540 two-action plans among 549 valid B/D plans. Thus the requested window size produces different actual lengths under this interface. The smaller models' fast A policies achieve 15.4–20.0 correct answers, while D achieves 2.0–4.4. With a single prepared action, feedback has no successor available after a target transition.

The requests explicitly say to plan CURRENT followed by visible NEXT, but the sole JSON example contains a one-element answer array. Source inspection confirms that NEXT is often present; for example, all 151 fast-D source reads for Qwen3-VL detect it, while all 151 plans remain single-step. This rules out absent preview content as an explanation for that cell. Across the four models' fast D arms, 526 of 536 injections have target labels and 520 reach another question; ten remain unknown. The result links the observed loss to execution without a prepared successor.

Table 16: Four additional models under the original ordered-array Quiz protocol. Each entry is mean correct/30 followed by mean calls over ten paired seeds; all 320 enrolled episodes remain in the denominator, including budget cutoffs. A/B/C/D have the definitions in Section 4.2.

Model	TTL (s)	A	B	C	D
Qwen3-VL-8B	7.5	19.9 / 30.0	19.7 / 30.0	20.1 / 30.0	19.5 / 30.0
Qwen3-VL-8B	2.5	19.7 / 30.0	19.1 / 30.0	3.6 / 15.2	3.5 / 15.1
UI-Venus-1.5-8B	7.5	19.8 / 30.0	19.5 / 30.0	20.0 / 30.0	19.7 / 30.0
UI-Venus-1.5-8B	2.5	20.0 / 30.0	18.8 / 29.9	3.6 / 15.1	3.3 / 15.0
MAI-UI-8B	7.5	15.5 / 37.7	11.7 / 39.5	14.7 / 37.8	11.4 / 39.8
MAI-UI-8B	2.5	15.4 / 32.0	11.3 / 33.8	2.5 / 19.5	2.0 / 21.9
UI-Venus-2-9B	7.5	18.9 / 30.5	18.1 / 30.5	18.9 / 30.4	18.8 / 30.4
UI-Venus-2-9B	2.5	18.0 / 30.1	19.2 / 30.1	4.4 / 15.4	4.4 / 15.5

Calls for these directly served models take roughly 0.8 s, while fresh checking adds approximately 1.1–1.3 s. Source recognition also overlaps each checked decision. This changes the timing balance relative to the 27B service. MAI produces 810 parse failures and UI-Venus-2 produces 30, of which 29 contain out-of-range answer indices. These failures consume calls. The matrix therefore measures the transfer of the complete original interface and runtime, including model-specific output behavior and serving cost.

D.2 Separating output length, target reading, and reasoning

A fixed development set contains 12 screenshots with CURRENT and NEXT, plus two terminal screenshots. Each model receives these same images through its Theta gateway route, with temperature zero, disabled thinking, 768 output tokens, a 90-second timeout, and no SDK retries. Three coordinate-producing conditions compare the original prompt, a two-element example, and explicit two-answer requirements. Two further conditions read NEXT alone or produce separately named CURRENT and NEXT objects containing the question, options, and answer. The latter two omit coordinates to isolate visual reading and answer selection. All 280 calls, including failures, remain in the diagnostic record.

Table 17: Window construction and screenshot diagnostics. Original $K = 2$ reports actual two-action plans in the formal B/D cells, across both TTLs. Parse errors include all four arms. Explicit-two and named-target columns use the same 12 nonterminal development screenshots per model via the Theta gateway. They measure output behavior, independently of live episode score.

Model	Original $K = 2$	Parse errors	Explicit two	NEXT options	NEXT correct
Qwen3-VL-8B	0/1051	0	12/12	12/12	7/12
UI-Venus-1.5-8B	0/1049	0	12/12	12/12	7/12
MAI-UI-8B	0/847	810	11/12	12/12	7/12
UI-Venus-2-9B	0/1050	30	12/12	12/12	7/12

Explicit two-answer requirements yield two actions on 12/12 nonterminal images for Qwen3-VL and both UI-Venus models, and 11/12 for MAI. Separately named targets recover the exact NEXT options on 12/12 images for every model. Each then answers NEXT correctly on 7/12: maximum selection is 6/6 and arithmetic 1/6. Asking for NEXT alone yields the same correctness split. This localizes much of the remaining error to answer selection after successful preview reading. The first stage retains eight transport errors and twelve MAI parse failures; the named-target stage has no transport or JSON-syntax failures. UI-Venus-2 nevertheless fails to emit NEXT=null on both terminal images. Output length, semantic target binding, and end-of-window detection are distinct requirements.

D.3 Structured windows in complete Android episodes

The complete named interface restores the four large-button coordinates and returns separate CURRENT/NEXT question, options, and answer fields. The runtime retains independent source OCR for target binding; model transcriptions are recorded for offline analysis and never used to repair an answer. Two deterministic codec rules normalize a single surplus closing brace and exact string coordinate pairs. They preserve the supplied numbers and action choices. Replay of 506 nontransport responses leaves 413 previously accepted action sequences unchanged and admits 79 additional MAI responses with string coordinate pairs; 14 other responses remain invalid.

The initial integration crosses four models, both TTLs, and A/B/D on one fixed development seed. All attempts, including transport recoveries and the coordinate-codec follow-up, are retained. In total, 38 attempts comprise 30 complete 30-question episodes, seven transport interruptions, and one call-budget cutoff. Recovery enrollment follows a transport failure or an unattempted configuration, without selecting episodes by score. Protocol versions remain recorded separately. The current v3.2 follow-up completes four fast-D episodes and MAI's slow/fast A episodes; the latter score 20/30 and 11/30 with 30 and 24 calls. No current follow-up episode has a parse failure.

Table 18: Named CURRENT/NEXT windows in complete local Android episodes at the 2.5 s TTL, one fixed development seed per model. All use $K = 2$ with OCR feedback and protocol v3.2. Two-step is conditional on a source-verified nonterminal screenshot; NEXT correct is source-answer accuracy. Target match divides matched successor injections by those with an attributable target. MAI has one additional successor injection with unknown target.

Model	Correct/30	Calls	Two-step	NEXT correct	Target match
Qwen3-VL-8B	7	15	15/15	10/15	9/9
UI-Venus-1.5-8B	11	15	15/15	10/15	11/11
MAI-UI-8B	5	15	13/13	8/13	7/8
UI-Venus-2-9B	8	16	14/14	8/14	9/9

Across the four fast-D episodes, all 61 parsed plans contain two actions. Of these, 57 have source-verified nonterminal inputs; all 57 contain both actions, reproduce the NEXT option list, and answer NEXT correctly on 36. Independent source OCR matches the NEXT signature on 55/57. The executor skips 31 expired prefix actions and injects 38 successors: 37 have attributable targets, of which 36 match. One terminal MAI source induces an extra NEXT action that is retained in the record and canceled. Across all 38 attempts, replay finds no canceled action subsequently injected and no attributed button-index mismatch.

These complete episodes establish construction and routing of structured windows across the four models. Task accuracy also depends on solving both targets and fitting execution within their deadlines: fast-D correct counts range from 5 to 11 out of 30. The local emulator and Theta routes differ from the formal direct-vLLM service, and exact checkpoint revisions are not independently matched across those services. The development episodes support the realized execution path; their scores and failure history remain separate from the formal matrix and do not estimate a cross-model score gain. No model was trained on this named Quiz protocol.

E Runtime Timing and Behavioral Diagnostics

These earlier protocols examine how action timing relates to behavior: proximity to a lock, advance planning, and declared completion can each diverge from task progress. Their configurations differ from the paired experiments in the main text and are analyzed separately below.

E.1 Campaigns and scope of comparison

The reported configurations are protocol-only (SOP carry and disclosure) and TurboAsync full (the streaming implementation). The experiments comprise protocol-only speed sweeps, full-runtime Tetris campaigns, and a clock-by-thinking intervention. The protocol-only grid includes two models on both tasks (120 episodes) and three additional Track-A configurations (90 episodes). The full run-

time contributes 30 Tetris episodes each for Qwen3.8-27B and UI-Venus-1.5-8B, with five episodes at each of six gravity levels per model. Generation uses temperature 0; campaign definitions and per-level statistics appear in Appendix F.

The 60 full-runtime raw episodes were independently recounted, including tail events and steps without injections. The remaining SOP results use campaign reports and summaries. Full-runtime metadata do not establish seed matching with SOP. These campaigns characterize within-campaign behavior; the matched release experiment in Section A supplies the controlled runtime comparison.

E.2 Call latency and scheduled budgets

The window protocol exposes a measurable scheduling constraint: each call must finish within the lead-plus-window budget assigned to it. We report call-budget compliance separately from task outcomes because a completed response can still schedule ineffective or late actions.

Observed call budgets. At p50, synchronous thinking-on calls take 63.4 s for Qwen3.8-2.4T-A95B and 13.1 s for Qwen3.8-27B. With thinking disabled, the corresponding latencies are 2.3 and 1.9 s; protocol-only window calls take 13.9 and 5.3 s. The 27B model finishes within the supplied call budget on 93% of Track-A and 100% of Track-B calls, versus 44% and 99.8% for the 2.4T model. This lead-plus-window budget measures call completion rather than each action's remaining target lifetime (Appendix H).

SOP action-span occupancy remains 0.16–0.28 for the 27B model, with zero Tetris score throughout that campaign. The additional Qwen3.6-35B-A3B and UI-Venus-1.5-30B-A3B configurations have occupancy 0.09–0.24 and 0.26–0.36, respectively, also with zero score. The former uses shared-queue serving with observed tails up to 364 s, so those delays include serving contention. Per-level occupancy and drop-proximity tables are in Appendix F.

E.3 Lock proximity coexists with zero-score behavior

The full-runtime campaigns show a separation between action timing and task utility. UI-Venus-1.5-8B emits a single hard drop in 290/290 decisions and scores zero in all 30 episodes, despite per-level lock proximity of 0.54–0.85. Qwen3.8-27B has lock proximity 0.50–0.68 and scores zero in 27 of 30 episodes; the remaining scores are 5, 10, and 15. These measurements show that a responsive but degenerate action policy can yield substantial lock proximity without sustained task progress.

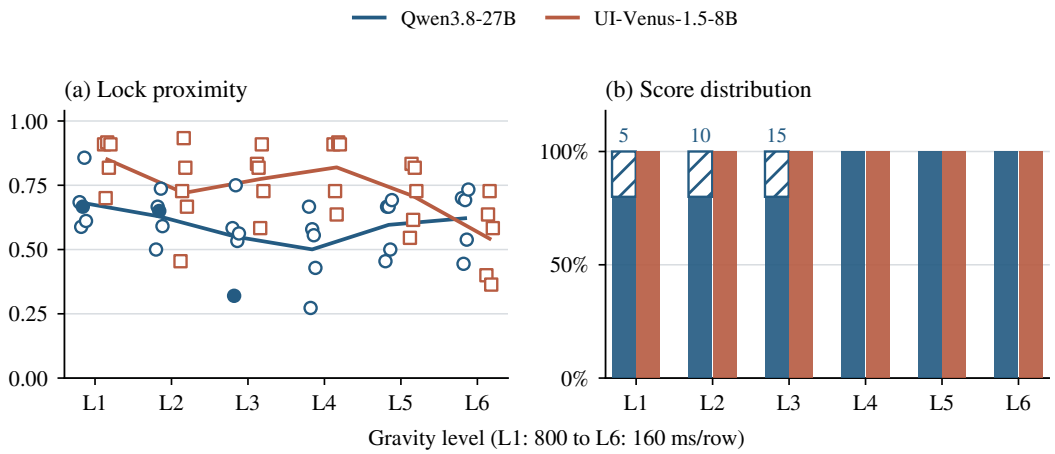

Figure 8: **Timing and task utility in full-runtime Tetris.** Five episodes per level and model. Left: open markers denote zero-score episodes, filled markers nonzero scores; lines join level means of episode-level C_{lock}. Right: solid bar height is the zero-score fraction; hatched remainders and labels show the three nonzero 27B scores. The model campaigns are separate rather than paired runtime comparisons.

E.4 Clock and thinking interventions change proximity, not the score floor

The Qwen3.8-27B factorial at L4 varies clock state and thinking while holding model and gravity fixed.

Table 19: Clock × thinking at Tetris L4 (370 ms/row) for Qwen3.8-27B. Five seeds per cell; entries are mean±std. All cells have zero score.

Clock	thinking	C_{time}	drop proximity	score
pause	on	0.02±0.01	0.55±0.12	0.00±0.00
pause	off	0.18±0.11	0.67±0.17	0.00±0.00
live	on	0.00±0.00	0.00±0.00	0.00±0.00
live	off	0.19±0.10	0.34±0.15	0.00±0.00

With thinking enabled in live mode, episodes terminate during the first 40–140 s model call, yielding zero occupancy and drop proximity. Disabling thinking raises drop proximity to 0.34. Pausing yields 0.55 with thinking and 0.67 without it. The paused thinking condition has action-span occupancy 0.02: most wall time lies outside first-to-last injection intervals. The intervention changes proximity while all four cells remain at zero score.

E.5 Preview-grounded commitments introduce prediction and tracking errors

On TurboQuiz, 95%–98% of window-arm answers are issued by actions planned before the question appears. These actions depend on interpreting the preview and mapping its content to the eventual live answer grid. For Qwen3.8-2.4T-A95B, preview-planned correctness falls from 0.88 at L1 to 0.46, 0.27, and 0.24 at L3, L4, and L5. Over the same levels, window accuracy moves from 0.12 below sync at L1 to 0.42–0.55 below sync at L3–L5. For Qwen3.8-27B at L2, preview-planned correctness is 0.60 and window accuracy is 0.10 below sync. At L6, the gap narrows as sync accuracy also declines (Figure 9). The campaign does not isolate the contributions of preview interpretation, timing, and early termination.

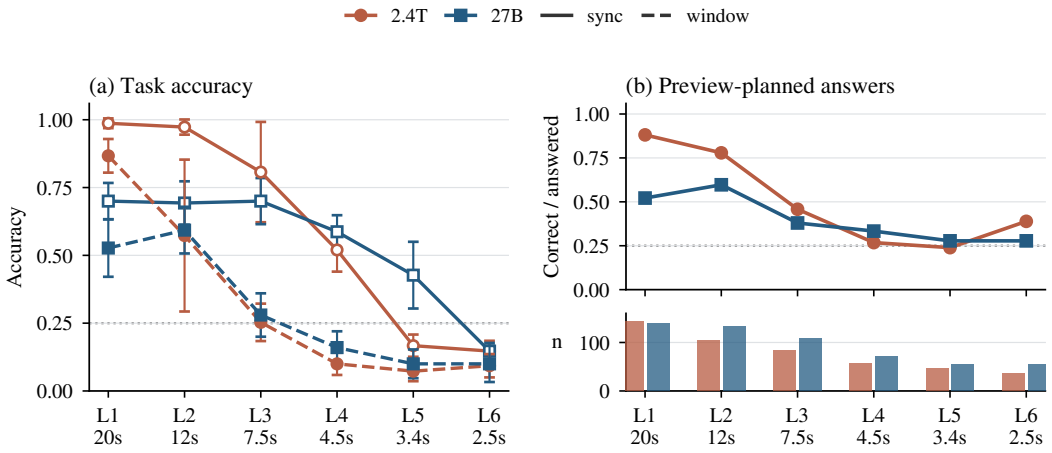

Figure 9: **TurboQuiz accuracy and preview planning across six deadlines.** Left: episode accuracy (mean±std, five episodes per cell) for sync and window arms. Right: correctness of pre-appearance commitments, pooled by question; the lower bars give the question counts behind each ratio. Horizontal dotted lines mark chance (0.25). Early exits remain in episode accuracy.

Episode completion introduces an additional failure. Qwen3.8-2.4T-A95B terminates itself in 5/30 window-arm episodes, compared with 0/30 for Qwen3.8-27B. Four terminations claim that all 30 questions were answered despite a server count of 25–29; the fifth stops at five and acknowledges incomplete progress. In the case in Appendix I, the model taps the correct next answer at its preview coordinates, then declares completion. The question has moved to the live grid by execution time,

so the second tap misses. The case links a concrete grounding error to a discrepancy between the model's declared progress and the environment ledger.

F Campaign Configurations and Per-Level Results

Table 20 reports the completed protocol-only and full-runtime configurations. Their different timing statistics characterize behavior within each runtime. The matched release experiment uses a common runner for its controlled comparison. Table 21 adds drop proximity for the protocol-only model configurations with lock telemetry.

Table 20: Runtime configurations (mean±std, five episodes per cell, live, thinking off). Protocol-only rows report action-span occupancy C_{time}; full-runtime rows report lock proximity C_{lock}. These metrics have different denominators and do not support cross-runtime effect sizes. Sync cells were not tabulated on a common metric. Appendix J gives definitions and provenance.

Model	runtime	L1	L2	L3	L4	L5	L6
SOP: action-span occupancy C_{time}							
Qwen3.8-2.4T-A95B	async	0.04±0.03	0.02±0.03	0.12±0.12	0.08±0.06	0.18±0.13	0.08±0.07
Qwen3.8-27B	async	0.23±0.04	0.16±0.06	0.28±0.09	0.18±0.04	0.20±0.08	0.16±0.08
Qwen3.6-35B-A3B	async	0.09±0.12	0.15±0.11	0.24±0.08	0.14±0.08	0.16±0.09	0.10±0.07
UI-Venus-1.5-8B	async	0.30±0.04	0.25±0.02	0.31±0.05	0.26±0.05	0.24±0.04	0.21±0.02
UI-Venus-1.5-30B-A3B[‡]	async	0.29±0.03	0.36±0.11	0.32±0.08	0.32±0.08	0.26±0.13	0.26±0.15
Streaming/full: lock proximity C_{lock}							
Qwen3.8-27B	async (full)	0.68±0.11	0.63±0.09	0.55±0.15	0.50±0.15	0.60±0.11	0.62±0.12
UI-Venus-1.5-8B	async (full)[†]	0.85±0.09	0.72±0.18	0.77±0.13	0.82±0.13	0.71±0.13	0.54±0.16

[†]degenerate windows: 290/290 UI-Venus-1.5-8B decisions commit a single hard-drop click, so high lock proximity coexists with zero task score (Section E.3). [‡]UI-Venus-1.5-30B-A3B prepends analysis text on most turns (turn-level format compliance 0.448); every emitted action parses and injects (action-level parse 1.000).

Table 21: Protocol-only drop proximity for the configurations with lock telemetry (mean±std, five episodes per cell, live, thinking off). Counting includes step-tail locks and gravity locks during steps without injections, using the SOP definition in Table 19. The Qwen3.8 protocol-only records lack lock telemetry, so this statistic cannot be reconstructed for those rows; their action-span occupancy is unaffected.

Model	L1	L2	L3	L4	L5	L6
UI-Venus-1.5-8B	0.69±0.10	0.67±0.00	0.58±0.06	0.52±0.10	0.38±0.04	0.27±0.05
Qwen3.6-35B-A3B	0.00±0.00	0.12±0.22	0.21±0.21	0.02±0.05	0.02±0.04	0.04±0.05
UI-Venus-1.5-30B-A3B	0.12±0.05	0.27±0.13	0.11±0.08	0.20±0.07	0.06±0.09	0.14±0.15

G Synthetic-instrument validation

A separate offline test replays a deterministic tiles environment through the scheduler and executor with a mock latency-controlled model. Its hit rate counts scheduled actions completed before their object's deadline. At 0.5–3 s latency, synchronous stepping achieves 0–8%; asynchronous windows achieve 97–100% once their temporal horizon exceeds latency.

Speculation improves hit rate by $+11$ pp at 3 s latency but reduces it from $97.2 \rightarrow 88.9\%$ at 1 s, where the baseline already meets most deadlines. Its effect therefore depends on the simulated latency regime. These measurements concern the controlled scheduler instrument, not on-device task performance. Per-latency raw counts were not independently recounted in the present audit.

H CALL LATENCY AND SCHEDULED WINDOW BUDGETS

Figure 10 shows the call-latency distributions underlying the budget-meeting fractions in Section E.2.

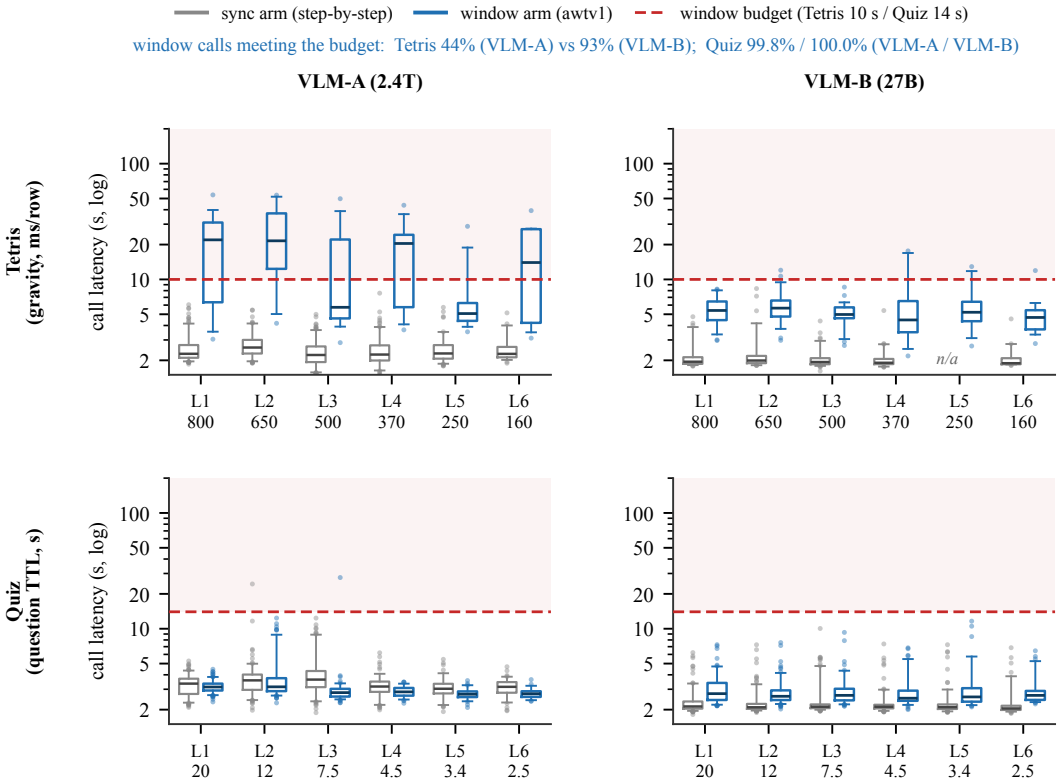

Figure 10: **Call latency and scheduled window budgets.** Per-call latency (log scale) for sync (gray) and protocol-only window (blue), by task, model, and speed. Boxes span p5–p95; dashed lines mark the 10 s Tetris and 14 s Quiz lead-plus-window budgets, with exceeding calls shaded. The header reports window-arm compliance; sync 27B Tetris L5 has no valid episodes. These call budgets are distinct from target deadlines.

I CASE ANATOMY: A MISFIRED PRE-COMMIT AND A FALSE COMPLETION CLAIM

Figure 11 dissects the final turn of one self-terminated Qwen3.8-2.4T-A95B episode from the Track-B campaign in Section E.5.

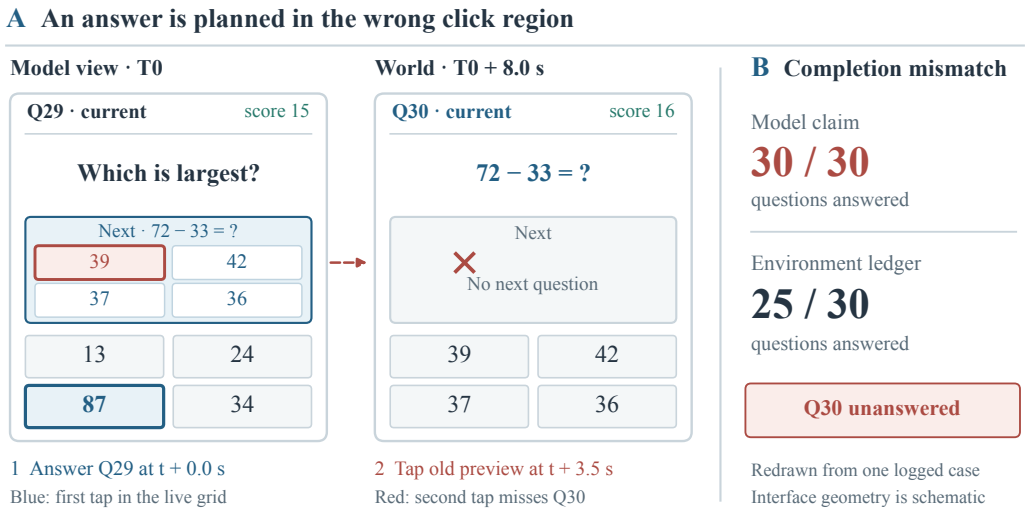

Figure 11: **Anatomy of a misfired pre-commit.** One Qwen3.8-2.4T-A95B TurboQuiz L2 episode (seed 12345). The interface is redrawn with schematic geometry and translated labels. Highlighted regions represent the logged taps: Q29's correct option at $t{+}0$ (265, 882), then Q30's correct answer at the *preview* coordinate at $t{+}3.5$ s (285, 547). The later state has an empty preview; Q30 was unanswered. Coordinates refer to the original screenshot, not the drawing. The model claimed all 30 answered, versus 25 in the server ledger. Across the campaign, $5/30$ episodes self-terminate, four with false completion claims; Qwen3.8-27B has $0/30$.

J MEASUREMENT DEFINITIONS AND EVIDENCE PROVENANCE

Matched release evidence. The 120-episode release comparison is separate from the historical 60-episode full-runtime audit below. Its figure input retains both episode IDs, per-episode first-injection delays and scores, seed clusters, and hashes of five recovered summary/manifest files. We cross-checked the two batch summaries and their manifests: 60 unique pairs cover six levels and ten seeds; requested and applied seeds agree, each recorded episode succeeds on its first episode attempt, and all have a first injection. Aggregates in the paper are recomputed from those pairs. A post hoc percentile bootstrap resamples ten seed clusters with replacement, retaining both arms and all six levels within each cluster. We use 20,000 replicates and random seed 20260925 for latency and score differences; this reanalysis was not preregistered. Raw event hashes are recorded in the manifests; the audit input comprises summary and manifest files rather than the corresponding raw events. This audit verifies the archived summaries and their alignment, and does not independently replay the 120 raw trajectories. The first-injection metric starts at the first model request and ends at the first successful injection of that decision.

Quiz target and score boundaries. The `quiz_target_window_v2` policy receives screenshot pixels only. Source OCR describes current and preview targets; fresh OCR describes the current target at a check. Neither reads the environment answer key or question ID. Offline scoring uses the environment ledger and a fixed denominator of 30 questions. Correct, wrong, expired, and unresolved outcomes remain distinct; missing telemetry remains missing. Target correspondence requires a stable source identity from event timing and source pixels. A screenshot overlapping a transition or an ambiguous identity is marked unknown and contributes to the reported audit-coverage denominator.

Formal matrix recovery and analysis. Both 80-episode matrices retain their frozen configuration, complete episode enrollment, model requests and responses, action logs, environment events, and source screenshots. Archive and per-file checksums match the collection-host records. The local Quiz analysis verifies all seed–TTL–policy cells, the 30-question denominator, call counts, and action-fate conservation; the Queue analysis replays every event ledger with the frozen scoring implementation. Reanalysis from the complete archives agrees with the lightweight record exports. Pixel target labels are retained from the collection-host offline audit; the local recovery validates the corresponding image files and recomputes the label aggregates. Development episodes remain outside both formal matrices.

The paper reports a common post hoc paired percentile bootstrap over ten seeds for each clock setting, with 20,000 resamples and random seed 20260925. This harmonizes the reported intervals across the two tasks while retaining the originally designated fast D−A comparisons. The collection summaries and their original interval draws remain in the archive. The question-level and injection-level decompositions are descriptive counts from these same episodes.

Relocation matrix recovery. The six-arm Ticket Queue experiment enrolls 120 episodes on new seeds with frozen bundle identity. We verify 9,633 delivered file hashes, every episode's configuration and runtime identity, and all 120 event-ledger scores. Replaying the stored pixel readings reproduces 2,826 check decisions, all coordinate transforms, and the bounded receipt attribution for 3,646 injections. The compact delivery omits PNG payloads while retaining their hashes; this pass replays the supplied OCR readings and does not independently recognize those omitted images. Both the original frozen summary intervals and the common paper bootstrap are retained. Differences in their last decimal reflect resampling draws, not a different episode set.

Model-window evidence. The four-model analysis verifies the additional 320 episodes together with the original 80 Quiz episodes: 25,173 compact files, 400 score replays, and all 40 cells agree with their delivered records. The smaller-model PNGs remain in their separate raw archives; target labels use the retained collection-host audit. Fixed-image diagnostics comprise 280 calls, not episodes. Local named-window integration verifies 2,711 files and scores for all 38 attempts. Formal ordered-array results, prompt diagnostics, and named-window development episodes have separate enrollment and protocol identities. Gateway aliases are recorded as requested models; returned gateway names do not independently identify checkpoint revisions.

Historical timing denominators. The SOP protocol-only campaign reports $C_{\mathrm{time}} = |\bigcup_j [f_j, l_j]|/(t_{\mathrm{end}} - t_0)$, where f_j, l_j are the first and last successful injection of turn j. This is action-span occupancy: it includes gaps between injections in a turn and assigns zero duration to a single-click turn. Its complement is not measured model waiting time. The streaming analyzer reports $C_{\mathrm{lock}} = N_{\mathrm{near}}/N_{\mathrm{locks}}$, where a recorded lock at t is near an injection if one occurs in $[t - 400, t + 60]$ ms. Positive offsets accommodate event-reception latency; the statistic is neither a strict before-deadline test nor completed-window coverage. The two metrics have different units and denominators, so no subtraction or cross-runner effect size is justified.

Control attribution. SOP's control proxy checks for a drop-coordinate click in the 700 ms before a recorded lock. The full-runtime statistic counts any injection in the $[-400, +60]$ ms band around a recorded lock; it does not filter action semantics. Temporal coincidence, logcat delay, repeated clicks, and incomplete event capture can all affect this proxy. A causal control metric requires an environment-side lock cause and stable piece identifier.

Audit scope and reproduction. `tools/audit_evidence.py` independently recomputes the 60 full-runtime episodes from local mirrored steps and metadata. It records episode IDs, file SHA256 values, and per-episode counts in `experiments/evidence_audit_20260923.json`; the same audit also retains historical short-run records excluded from this paper's results. Counts agree with the published full 27B source. Full 8B per-level pooled counts are $47/55, 44/60, 44/57, 46/56, 41/58, 30/55$. Agreement with logged events does not establish that every physical lock was captured. Bootstrap and terminal capture completeness remain separate checks. The full protocol-only/E2 raw campaigns were not available in the local audit; their statistics remain report/summary-backed. The old Qwen

protocol-only campaign has no precise lock-event field, so a retrospective causal-lock comparison cannot be obtained from that record.

Runtime and version boundaries. The protocol-only arm runs through `sop.runner` (awtv1); streaming/full uses `aw_bridge.run_tetris_aw`. Shared conceptual mechanisms do not make these paths a single-variable ablation. The full runtime's lock-trigger overlap can fall back to sequential completion. The September 22 recount merges tail events and includes no-injection steps. That change corrects lock denominators, not the definition of SOP wall-time occupancy. The SOP E2/E1/E5 control reports also use a different clock anchor and drop filter. These historical comparisons cannot isolate streaming, overlap, or clamping. The matched release experiment holds its runner and settings fixed, varying only release mode.

Seed provenance. The mirrored full-runtime metadata do not record a requested/applied seed, and the launcher requests five episodes per level rather than supplying the SOP campaign's explicit seed list. These are five repeated episodes, not verified seed-matched pairs across runners. The matched release and new Quiz protocols record requested and applied seeds. This does not retrospectively establish seed matching for the older campaigns.

Historical source records. The archived figure inputs `exp12.json`, `e1e5_async.json`, and `tafull.json` retain the protocol-only and full-runtime summaries. The accompanying provenance records identify the additional 8B full-runtime report and the September 22 lock recount. These sources retain their original metric names; the definitions above govern their interpretation here.