BENCHMARKING EMBODIED FLUID INTELLIGENCE

Anonymous authors
Paper under double-blind review

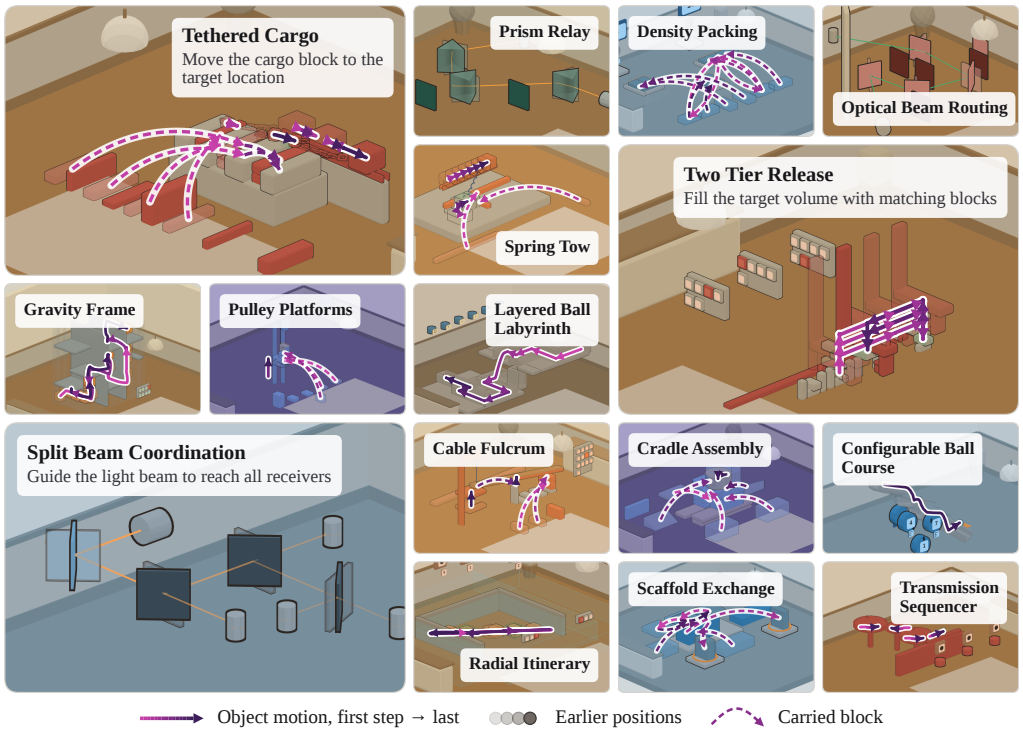

Figure 1: We introduce a benchmark for **embodied fluid intelligence**. Each task involves a novel physical mechanism and a goal for multimodal LLM agents. An agent must efficiently discover the environment's affordances, infer causal relationships between actions and outcomes, and compose action sequences that accomplish the goal. **More examples at iclr-fluid-int.pages.dev (anonymous)**.

ABSTRACT

We study fluid intelligence, the capacity to efficiently infer and exploit the actionable structure of an unfamiliar environment from limited action–perception experience, in interactive 3D worlds. We present Flint, a benchmark of 180 tasks, validated for solvability, unfamiliarity, and meaningful controlled variation. Each task is an unfamiliar puzzle in a physics-enabled 3D room. Instead of manually creating this benchmark, we design a framework that uses coding agents to automatically create such tasks. In particular, we ask coding agents to author task generators that allow us to generate not only individual tasks but their controlled variants. Evaluating frontier multimodal LLM agents on Flint, we find that GPT-6 Astra and Claude Opus 5.5 have increased the solve rate by > 35 percentage points over prior models. Yet, while they saturate many other benchmarks, they leave more than a third of the tasks unsolved on Flint. Our analysis sheds light on the challenges and opportunities for advancing embodied fluid intelligence.

1 INTRODUCTION

A hallmark of intelligence is the ability to solve problems for which prior experience provides no ready-made solutions. Psychologists distinguish fluid intelligence – the capacity to reason through

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.

novel problems – from crystallized intelligence accumulated through experience (Cattell, 1963). In artificial intelligence, Chollet (2019) argues that task performance can be acquired through priors and training, whereas "true" intelligence is revealed by skill-acquisition efficiency. Humans and other animals display this ability. Goffin's cockatoo, for example, can work out how an unfamiliar man-made puzzle functions through exploration (Auersperg et al., 2013).

In this work, we study *embodied fluid intelligence* – the capacity to efficiently infer and exploit the affordances and causal structure of an unfamiliar 3D environment, experienced from an immersive first-person perspective. We present Flint, a benchmark for embodied **fl**uid **int**elligence. In each Flint episode, the agent is embodied in a novel physics-enabled environment that is populated by unfamiliar mechanisms. The agent must infer the affordances (what can be acted upon) and the environment's causal structure (the effects of various actions). The agent must accomplish this efficiently, given a limited action budget, via its intrinsic first-person view.

The methodology for Flint task generation stems from the following desiderata. (a) Novelty, meaning the task-specific actionable structure should not be easily predictable by agents without interactive exploration. (b) Breadth, meaning the tasks must span many distinct mechanisms. Sampling many layouts from one mechanism increases statistical coverage, but not the scope of intelligence being tested. This calls for diversity in task mechanisms, rather than task instances. (c) Control over analysis, including knowing how tasks relate to each other, and exposing structured interventions on properties such as causal depth via task variants.

Flint comprises 180 tasks sampled from 60 distinct task generators. Fig. 1 shows some examples. Each task is an unfamiliar puzzle in an interactive 3D playroom. An agent moves, observes, and acts on objects in a playroom to achieve a goal. Fig. 2 shows an agent working its way through one of the 180 playrooms. For scalable creation of task generators, we use coding agents. However, instead of asking coding agents to author isolated tasks (Xia et al., 2026; Faldor et al., 2025), our framework makes them author parametric task generators by composing a library of verified assets inside a physics-enabled 3D engine. Different generators define different task families with unique action–effect mechanisms and goals, providing novelty and breadth across problems. Within each generator, task instances preserve the mechanism while including parameters that vary properties relevant to embodied fluid intelligence (*e.g.* occlusions that make mechanism discovery in 3D difficult), thus supporting controlled comparisons.

Our analysis confirms that Flint tasks are solvable through the available interaction interface, that their affordances and causal structure are not readily inferred by frontier agents from passive observations alone, and that the task generators produce meaningful controlled variations. Our experimental results reveal a clear separation at the current frontier of multimodal LLM agents: GPT-6 Astra and Claude Opus 5.5 reach 60.6% and 44.4% solve rates, respectively, at our largest cost allowance, whereas no other evaluated agent exceeds 10%. Analyzing this gap, we find that these two agents

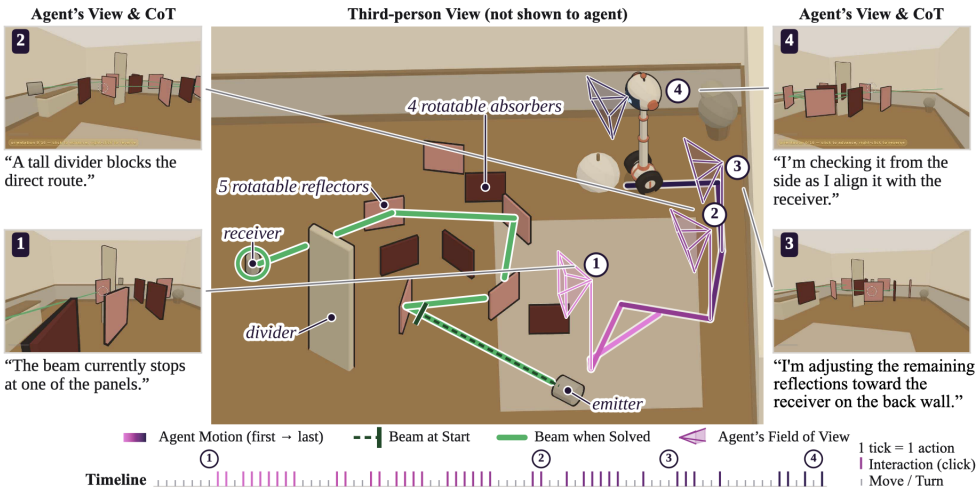

Figure 2: We show an agent (GPT-6 Astra) solving an optical beam routing task in Flint. Please see the videos in the supplemental material and the website.

are substantially more efficient at grounding intended interactions in the 3D environment, which is strongly associated with solve rate across models. Despite this progress, more than a third of Flint remains unsolved by frontier agents. Our analysis using controlled task variants points to a recurring limitation of GPT-6 Astra, the most capable agent we have evaluated. It may form a plausible but imperfect hypothesis and then fail to update it as new evidence emerges, thus trapping itself in a conceptual "local minimum".

2 Related Work

Fluid intelligence evaluation. Psychometric theories distinguish the ability to reason through unfamiliar problems from knowledge and skills retrieved from prior experience (Cattell, 1963; Horn & Cattell, 1966). Transposed to AI evaluation, this distinction raises a basic attribution problem, where a benchmark score reflects both what an agent brings to the task and what it can learn from the evidence at hand. To address this problem, PGM (Barrett et al., 2018) and RAVEN (Zhang et al., 2019) test systematic reasoning over abstract visual structures, while ARC (Chollet, 2019) requires induction of a task rule from a small set of demonstrations. More recently, GENIUS (An et al., 2026) tests multimodal models on inducing novel patterns, following ad-hoc constraints, and adapting to contextual rules in visual generation. These benchmarks reduce reliance on familiar semantics, but often provide the evidence needed for inference directly to the agent. More interactive settings couple reasoning with the choice of what evidence to gather. Comparative-cognition experiments have shown that animals can uncover the operation of unfamiliar apparatuses through active manipulation attempts (Auersperg et al., 2013). In AI evaluation, Animal-AI (Crosby et al., 2020; Voudouris et al., 2025) evaluates transfer to novel configurations composed from a simulator and interaction repertoire available during training. ARC-AGI-3 (ARC Prize Foundation, 2026) evaluates the acquisition of an unfamiliar environment's goal and mechanism at test time through interactions. We extend this line of work to embodied fluid intelligence, where agents must efficiently infer and exploit the affordances and causal structure in unfamiliar immersive environments to accomplish goals.

Spatial intelligence benchmarks. Spatial intelligence benchmarks commonly evaluate scene understanding, memory, and functional reasoning from recorded observations (Yang et al., 2025a; Zhang et al., 2026). Embodied AI benchmarks (Kolve et al., 2017; Savva et al., 2019; Shridhar et al., 2020; Li et al., 2023; Yang et al., 2025b) add active navigation, manipulation, and planning in 3D environments. Benchmarks structured around household environments, however, are largely organized around familiar places, objects, and activities, and thus lean heavily on crystallized intelligence. MineDojo (Fan et al., 2022) provides substantial variation in goals and world configurations, but every episode remains governed by an established game mechanism; abundant public demonstrations and documentation further skew this setting towards crystallized intelligence. Megaverse (Petrenko et al., 2021) provides a high-throughput platform for training embodied agents in immersive environments. PHYRE (Bakhtin et al., 2019) and I-PHYRE (Li et al., 2024) test intuitive physical reasoning. VisEscape (Lim et al., 2025), MM-Escape (Wang et al., 2025), and ESI-Bench (Hong et al., 2026) require interactive exploration for unconventional tasks in familiar household environments. Flint's creation framework focuses on interactive challenges in novel environments with unfamiliar mechanisms, enabling controlled study of embodied fluid intelligence.

Spatial task generation. Many spatial intelligence benchmarks (Kolve et al., 2017; Savva et al., 2019; Li et al., 2023; Hong et al., 2026) rely on manually authored tasks. Procedural generation (Cobbe et al., 2020; Deitke et al., 2022) has expanded level and scene distributions. More recently, code-generating agents have been used to accelerate and scale the production of tasks and environments, as well as policy evaluation (Wang et al., 2024a;b; Faldor et al., 2025; Xia et al., 2026; Chen et al., 2026). Our framework uses coding agents to author task generators; each generator can spawn a family of playrooms with controlled variation factors.

3 Authoring Task Generators via Coding Agents

Flint's creation framework creates parametric task generators rather than individual tasks. Each generator defines an action–effect mechanism and produces a controlled family of interactive immersive environments. Different generators broaden the mechanisms under evaluation, while controlled variation within each generator's family supports targeted analysis of multimodal agents. We first

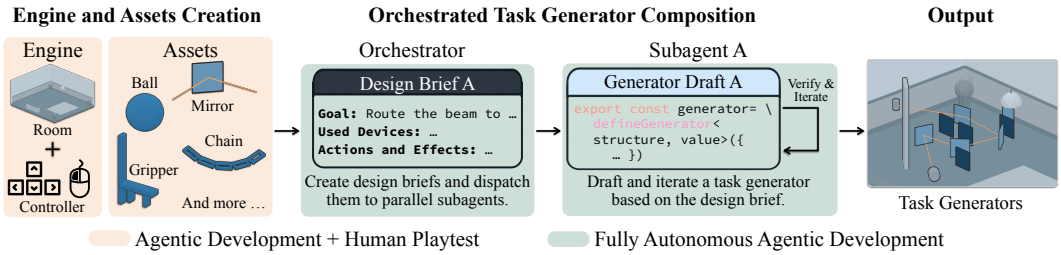

Figure 3: **Flint's creation framework.** The framework starts by creating the interactive engine, including a shared room shell for all tasks, a general controller, and a library of interactive assets, all playtested by humans. Then an orchestrated multi-agent composition procedure autonomously composes the assets into diverse task generators.

describe the tasks we aim to create, then explain how generators define controlled task families, and finally the coding-agent workflow for authoring them.

Embodied tasks with unfamiliar mechanisms. Each task is an independently playable puzzle in an interactive 3D playroom. The room contains assets such as pulleys, grippers, or turnable mirrors, coupled through a task-specific composition, with a visually grounded and textually specified goal, such as "move the ball to the marked platform". Each room is driven by a mechanism defined as the set of rules about what can be interacted with, what effects each interaction has, and how the action-effect coupling relates to the goal. An agent receives egocentric visual observations and acts through a common low-level interface for navigation, camera control, and object interaction. The interface does not expose hidden world state or semantic explanations of interactions. To solve a task, the agent must choose interventions, observe their consequences, infer enough of the underlying mechanism, and compose actions to reach the goal within one episode. Central to our tasks is mechanism unfamiliarity: a room's mechanism is hard to predict without interacting with it. We achieve this by placing tasks in rooms unlike everyday environments, with causal structures that depart from everyday conventions. The scenes evolve through simulated physical and causal processes based on the agent's actions. Progress can be inferred only by observing the consequences of one's actions within the playroom, as there is no extrinsic reward.

Parametric generators as controlled task families. A large collection of isolated tasks provides scale, but little structure for understanding what makes a new problem difficult. For better understanding, we create families of tasks that vary along controlled factors. To this end, we create parametric task generators rather than individual tasks. Such parametric generators are the basic unit in Flint's creation workflow. Each generator is based on one task mechanism, which specifies the roles of its assets, the available interactions, the resulting state transitions, and the principle by which the goal is achieved. Instances from the same generator preserve this mechanism while varying its concrete realization by changing certain properties. For example, an optical beam routing generator may always require orienting reflectors to direct a beam from an emitter to a receiver, while varying the relay depth, spatial arrangement, and available orientations. Within a generator, different properties are configurable as parameters, which lets us construct matched tasks that preserve the mechanism and surrounding context while changing factors such as ambiguity among candidate actions, action–effect separation, or causal depth. Scaling across generators expands the range of mechanisms, while coordinated sampling within a generator provides controlled interventions for diagnosis. Fig. 4 provides examples of controlled variation.

Agentic creation of task generators. Manually implementing many distinct controlled task families is expensive. Our framework therefore uses a multi-agent workflow to author generators from a shared library of verified interactive assets and simulation capabilities. We first use a dedicated coding agent to develop the environment engine that implements the rendering and interaction framework, with physics simulation support. In this step, the coding agent also develops a library of interactive assets such as turnable optical reflectors. Both the engine and the asset library are implemented in ThreeJS (Cabello & Three.js Authors, 2010) with physics support from Rapier (Dimforge, 2020). A human playtester participates in this step to verify the engine and assets through playtesting. Once the engine and assets are finalized, we start the parallel agentic development of task generators. We have a dedicated orchestrating coding agent to author generator briefs, which specify the intended

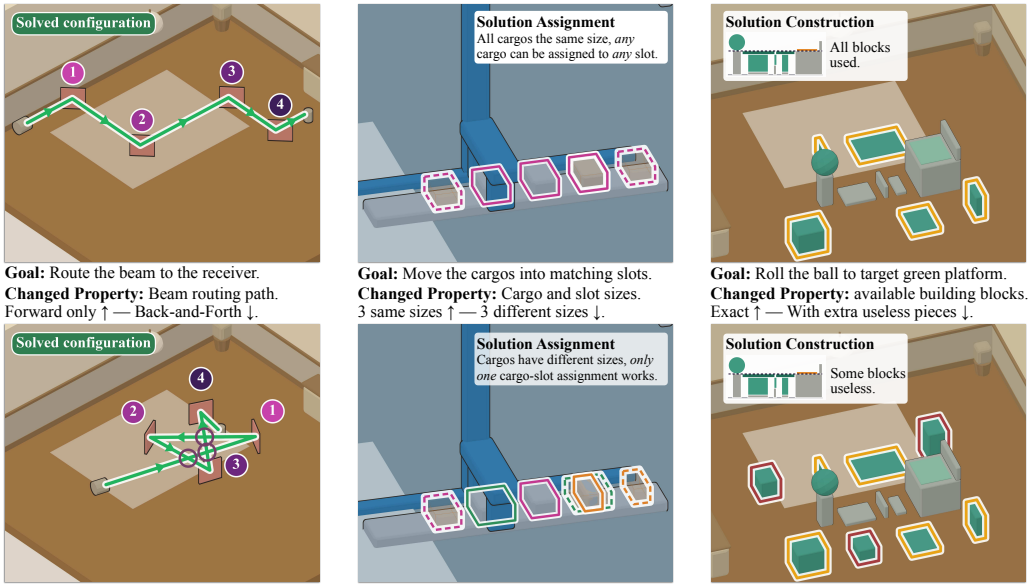

Figure 4: **Controlled variation.** Three pairs of tasks (top, bottom), sampled from three respective generators. Each pair differs by a specific property controlled by its generator.

action–effect principle, success condition, and candidate parametric properties the generator can vary. This orchestrator also ensures the uniqueness of each created generator by making at least one of the aspects above, except the parametric properties, unique among all generators. A fleet of subagents then receives the design briefs and develops in parallel. They compose the required assets into a task generator that samples task instances, records the intended solution, and exposes the declared variations. The resulting artifact from each subagent is a task family, not a single authored task. Once a generator is drafted, before admission to the generator collection, it needs to pass an extensive set of automated verification tests (see Sec. 4) prepared during the engine and asset development stage. Drafts failing the checks are returned to the subagents, which either discard them or refine and resubmit them. The verified generators provide the building blocks of Flint as described next.

4 Flint: A Benchmark for Embodied Fluid Intelligence

Curation. We curate Flint as an evaluation-only benchmark of 180 tasks from 60 distinct task generators, matching the scale of existing agent benchmarks (Huang et al., 2026). Under Flint's creation framework in Sec. 3, we use coding agents to collect 60 task generators. For each generator, we ask the coding agent to first iterate on its own to pass all verification tests in the creation framework. These tests cover aspects including scene validity, observability, simulation stability, and solvability. Additional checks screen for triviality of tasks (e.g., solvable by a random policy). See Appendix A for more details. Generators that do not pass the tests after a few iteration rounds are filtered out. Generators that pass are further examined by a human through the playrooms they generate. For each task, we show the human tester a replay of the solution produced by the generator's oracle solver. We ask the tester the following questions: Is the goal clearly visible? Are the action effects visible? Is the environment free of geometric defects? And is the reason for each action in the solution also visible? To assist the tester's judgment, we also provide an interactive version of the same task for active inspection. Only tasks with all positive answers to the above questions are admitted into the benchmark. We sample 4 tasks per generator and only accept generators whose 4 tasks are all accepted. In our human playtest, we find that 91% of the generators that pass the automated framework tests are subsequently accepted by the human tester. See Appendix D for more statistics on the generator creation. Once the generators are finalized, we sample 3 random tasks from each generator to form the 180 tasks in Flint.

Validation. For the final 180 tasks in Flint, we further examine their validity for our intended evaluation on embodied fluid intelligence. Our validation focuses on 2 questions: 1) **Can the task be solved using the available actions and observations?** The aforementioned unit tests and human tests

have proven that our tasks are solvable using given actions, and the actions can be discovered visually. Hence, the answer is positive for all 180 tasks. 2) **Are the interactive mechanisms unfamiliar?** To test this, we quiz AI agents about the 60 generators in Flint. The quiz tests how accurately agents can identify interactable objects (object accuracy) and predict the effects of interactions with them (effect accuracy), *without interactive exploration*. Our results show that object accuracy spans from 33.28% to 55.23% among agents, whereas effect accuracy ranges from 6.80% to 16.16%. This implies that while agents might be familiar with certain interactive assets in Flint, it is novel to them how the interaction works, namely the mechanism. See Appendix B for more details. In addition, we validate our collected 60 generators on their capability of generating meaningful variation. To this end, we sample two pairs from each generator. Each pair differs in one property controlled by the generator. We present a human tester with the paired playrooms and ask whether they exhibit the intended difference. The answer is positive for 88.33% of the tested pairs (106 out of 120). This suggests that the agent-authored generators are largely effective in intentionally producing controlled variation for fine-grained analysis of fluid intelligence. Note that we did not filter any authored control properties.

Evaluation protocol. We evaluate agents through interactive episodes in the Flint playrooms. For each episode, the agent starts from a fresh environment and model context. It is given a prompt describing the goal of the task and the available actions. In a modified protocol used in our later analysis, the prompt also includes a textual description of the mechanism, without revealing any part of the solution (termed L0 protocol. The former goal-only prompt is Flint's default (termed L1 protocol). L0 protocol is used for specific controlled experiments. Actions in Flint are general input interfaces for agents, including movement, turning, panoramic observation of surroundings, and clicking. Clicking is the action that signifies an interaction attempt – if the agent clicks on an interactable object, a corresponding interaction takes place. In different tasks, a click can have different semantics, from pressing a button to picking up a brick. The agent is given only the newly rendered frames as feedback after each action. No structured textual feedback is provided off-screen during the episode. All previously observed frames remain accessible to the agent throughout the episode. The agent also has a "scratchpad" folder where it can write code in Python or bash, take notes, or process the observed frames. To avoid endless episodes, we set the time limit to a generous 2 hours, and the allowed interaction limit to $10\times$ the oracle interaction length for the respective task.

Task distribution. The 180 tasks in Flint span a wide range of mechanisms. These mechanisms contain different physics simulations, including rigid-body dynamics, simplified geometric optics, gravity and force fields, rigid-body contact, articulated kinematics, and linkage kinematics. On average, each task uses 21 assets from Flint's creation framework, with 5 unique asset types. In terms of episode length, the oracle solver solution for each task has 3 to 41 interactions, with the average being 9, or 8 to 134 overall actions including navigation and aiming, with the average at 34. To analyze performance by mechanism, we categorize the 60 generators into 12 categories based on the generator's key assets and physical principles, such as optical beams, weight & springs, rolling-ball courses, gripper & arm, support construction, and lever & tipping balance. The categories are detailed in Appendix G. All the task generators are listed in Appendix H.

5 Benchmarking Embodied Fluid Intelligence

Setup. We evaluate a set of multimodal LLM agents, including both black-box and open-weight models. Specifically, we benchmark GPT-6 Astra, Claude Opus 5, Claude Opus 5.5, Gemini 3.7 Flash, and Claude Sonnet 5 as representative black-box models. We select Kimi K3 and Qwen 3.8 27B as strong recent open-weights models. We follow prior agent benchmarks (Zhou et al., 2024) in evaluating each agent for one rollout per task under the same interaction interface, with a fresh context for every episode. To account for differences in inference cost across systems, we sweep the maximum cost allowed per task from \$2 to \$20 and report solve rate as a function of this cost cap. At the maximum cost, we additionally report success-weighted interaction efficiency (SIE). SIE is defined as $\mathrm{SIE}_i = \mathbf{1}[\mathrm{success}_i]\, c_i^{\mathrm{ref}} / \max(c_i^{\mathrm{ref}}, c_i)$, where c_i and c_i^{ref} are the agent and reference interaction counts, respectively; $\mathrm{success}_i$ is the binary success indicator. SIE measures the interaction efficiency relative to the verified reference solution.

Results. Aggregate evaluation results are shown in Fig. 5. We make several observations. First, the strongest closed agents massively outperform open-weight models. Second, GPT-6 Astra has a clear lead along the cost–performance frontier. Across the full range of cost limits, GPT-6 Astra

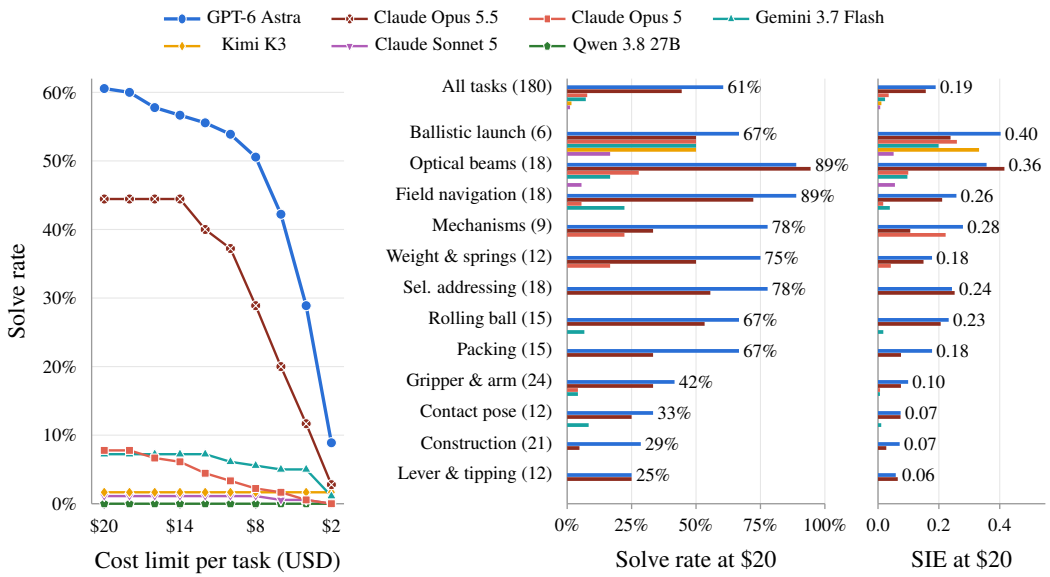

Figure 5: **Quantitative evaluation on Flint.** *Left:* solve rate as a function of maximum allowed cost per task. *Right:* solve rate and success-weighted interaction efficiency (SIE) at the maximum cost per task ($20), over all tasks and broken down by mechanism category, with the number of tasks per category in parentheses and the quantified performance of GPT-6 Astra at the end of each row.

achieves a higher solve rate, reaching approximately 61% success at the $20 cap. Claude Opus 5.5 is the closest, reaching 44.4%, while the other agents remain below 10% at the $20 cap. Third, despite this progress, GPT-6 Astra still leaves roughly 40% of Flint unsolved even at the largest cost allowance, and its interaction efficiency remains low. Fourth, GPT-6 Astra leads in ten of the twelve mechanism categories, while Claude Opus 5.5 is ahead in Optical beams (94% vs 89%) and ties with it in Lever & tipping balance. In some categories, Astra's solve rate is substantially lower than its average (*e.g.* Construction). These observations motivate us to ask two follow-up questions: What behaviors distinguish recent frontier models like GPT-6 Astra from weaker agents in Flint? And what still limits the frontier performance?

Understanding the leap of recent frontier models. To answer the first question, recall from Sec. 3 that solving a task means choosing interventions and learning from their consequences, and that each intervention is itself an embodied problem: the agent must find the right object, reach it and click it. A click becomes an *accepted interaction* when the environment acts on it, for example by pressing a button or picking up an object, and every accepted interaction changes the room and so gives the agent evidence about its mechanism. We measure how much effort an agent spends to get there with the Actions to Accepted Interactions (ATAI) metric: the number of actions it takes for each accepted interaction, counting its turns and moves as well as its clicks that are not accepted. Lower is better. GPT-6 Astra stands clearly apart: it needs 3.96 actions per accepted interaction, Claude Opus 5.5 8.10, and the other agents 14.9–20.6. ATAI also ranks the seven agents almost exactly as their solve rates do (Spearman $\rho = -0.96$). The gap opens up in the episodes that agents fail (Fig. 6a). When the other agents solve a task, their ATAI is much closer to that of GPT-6 Astra, but when they fail, it rises far above it, while GPT-6 Astra stays at 5.1. Much of this extra effort goes into long periods of 30 or more actions without a single accepted interaction. Such periods take up 18% of the actions of Claude Opus 5.5 and 29–50% of those of the weaker agents, but only 7% of those of GPT-6 Astra (Fig. 9a in Appendix F). What distinguishes Astra, followed by Opus 5.5, is therefore that they rarely get stuck, so they keep gathering evidence about the mechanism throughout the episode.

The actions an agent takes before an accepted interaction are of two kinds: navigation (turns and moves) and clicks that the environment does not accept. GPT-6 Astra keeps both costs low: it makes only 2.7 turns and moves per click, and 79% of its clicks are accepted. Its few turns and moves go together with how it navigates. Every action returns a new frame, but the agent does not have to look at it before acting again. It can therefore plan its approach to an object from frames it has already

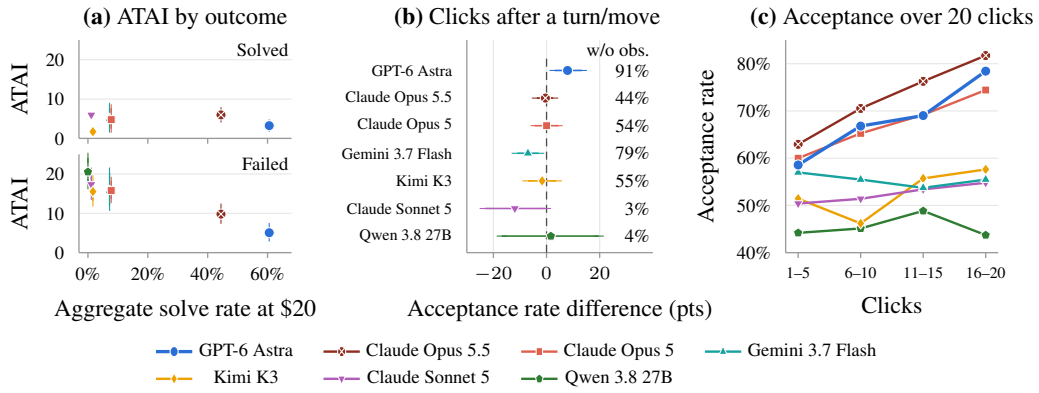

Figure 6: **How GPT-6 Astra turns observation into interaction.** **(a)** Actions to Accepted Interactions (ATAI) is the number of actions an agent takes for each click that the environment accepts, counting its turns and moves as well as the clicks that the environment does not accept; lower is better. ATAI is plotted against solve rate, separately for solved and failed episodes (95% CI); Kimi K3 and Claude Sonnet 5 solve too few tasks (3 and 2) for an interval, and Qwen 3.8 27B solves none. **(b)** After a turn or a move, an agent can click right away, before seeing the new frame (a click *without observation*), or only after it could look at that frame. Each point is the fraction of clicks without observation that are accepted minus that of the other clicks, computed within each episode and then averaged (95% CI). A positive value means that clicks sent without looking are accepted more often. The percentages headed "w/o obs." give the share of clicks sent without observation. **(c)** Do agents get better at clicking as they explore? Each line shows what fraction of an agent's clicks are accepted, first for its 1st to 5th clicks, and so on up to its 20th click. For each agent we use only the episodes in which it clicks at least 20 times, so all points of a line come from the same episodes, and a rising line means that its clicks are accepted more often later in these episodes.

seen, and send the turns, the steps and the click together in one script. GPT-6 Astra sends 91% of the clicks that follow a turn or a move in this way, *without observation* of the new frame, and these clicks are accepted even more often than the ones it sends after it could look (7.9 percentage points more within the same episode; Fig. 6b). It thus aims well without checking each step, which suggests that it can predict where it will be pointing after a turn or a move. Claude Opus 5.5 has its clicks accepted just as often (80%), but it plans this way for only 44% of such clicks and makes twice as many turns and moves per click (5.5). The other agents fall behind on both counts.

Clicks that are not accepted complement the picture (Fig. 6c). To see whether agents click better as they explore, we take each agent's episodes with at least 20 clicks, split their first 20 clicks into blocks of five, and measure in each block the share of clicks that the environment accepts. Since every block comes from the same episodes, a rise indicates that the agents make use of what they have seen earlier in the episode. For GPT-6 Astra and the two Opus models this share rises from 59–63% in the first block to 74–82% in the last, as fewer clicks hit nothing or are refused, even in the tasks they fail, whereas the other agents show no clear change. As Astra interacts efficiently even in failed tasks, its failures likely lie in how it uses the gathered evidence, which we examine next.

What Remains at the Frontier? Despite its lead, GPT-6 Astra still leaves 40% headroom on Flint at our largest cost allowance. We first ask how much of this gap comes from acquiring the task mechanism itself. For the 71 tasks that GPT-6 Astra fails under our goal-only L1 protocol (Sec. 4), we rerun the same tasks under L0, where the prompt additionally provides the correct mechanism description while still withholding the solution. Of these failed tasks, 34 flip to success, a 47.9% recovery rate; had they been solved in the primary evaluation, the aggregate solve rate would increase from 60.6% to approximately 79.4%. Thus, correctly acquiring the mechanism is a substantial part of the remaining error, while the other L0 failures indicate additional sources of difficulty, such as fine-scale spatial perception and planning efficiency (see Appendix C for more details).

A closer inspection of the L1 failures, using the reasoning summaries of the failed episodes, reveals another failure source. GPT-6 Astra often forms a hypothesis about the mechanism early in an episode, but the hypothesis can be inaccurate in a critical detail. Subsequent exploration then tends to remain concentrated around local adjustments consistent with that interpretation, rather than seeking

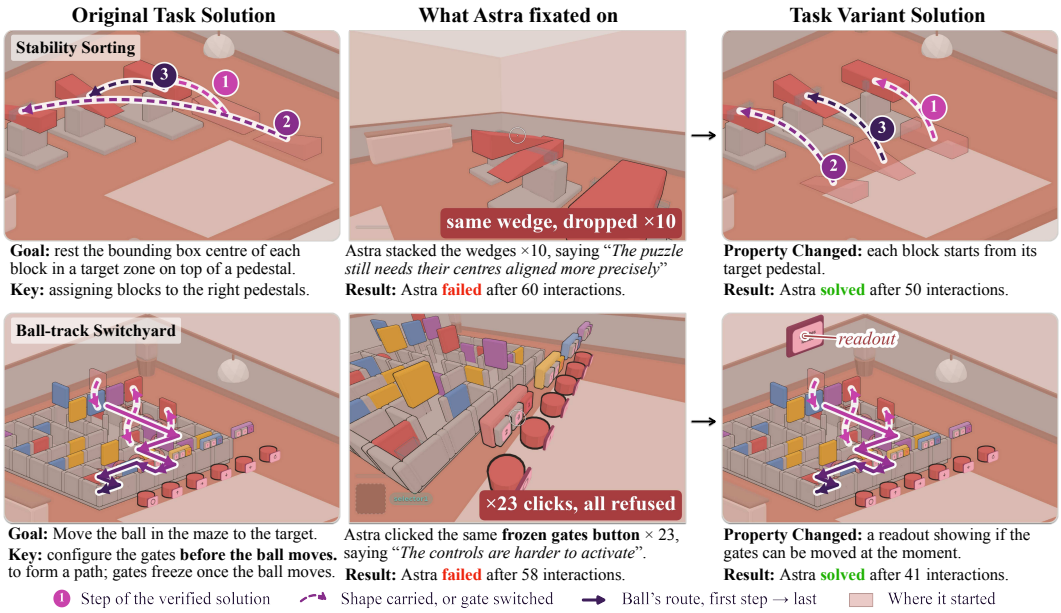

Figure 7: GPT-6 Astra is prone to limited yet repeated local attempts when larger-scale refinement of its mechanism understanding is needed. We show 2 such episodes in 2 rows. Each row starts with the original task in Flint that Astra failed. In the middle, we show where Astra was stuck in local attempts. On the right, we show task variants that make the bottleneck less likely (top) or indicate it visually (bottom), leading to Astra solving the tasks.

interventions that would distinguish it from alternative causal explanations, so GPT-6 Astra is *trapped* in limited yet repeated local attempts (Fig. 7). The issue is therefore often not the absence of an initial hypothesis, but insufficient *refinement* of that hypothesis as new evidence becomes available. Because the action plan is built on the inferred mechanism, failure to revise the mechanism also limits the agent's ability to revise its plan. We test this interpretation directly using the controlled variants provided by our task generators. For 10 representative Astra failure cases exhibiting such a local "trap", we manually pick a matched variant from the generator that alters only the part of the task related to the false part of the initial hypothesis by either removing the "trap" or making it visually distinguishable to the agent. Under these targeted interventions, 7 of the 10 failed runs flip to success (2 are shown in Fig. 7). This supports the hypothesis that a key remaining limitation is not simply perceiving or acting in 3D, where Astra already shows substantial strength, but continually revising an internal account of how an unfamiliar world works and reorganizing behavior when later evidence favors a different explanation. Improving this capacity for hypothesis refinement and plan revision appears to be an important direction for further progress in embodied fluid intelligence.

6 Conclusion

A cockatoo facing an unfamiliar puzzle box does not need a manual. It pokes, watches, and tries again until the box gives way. Flint asks the same of AI agents, placing them in 3D playrooms whose mechanisms they have never seen and letting them learn only from what their own actions reveal. The picture that emerges is encouraging. The newest frontier agents solve 44–61% of the tasks, whereas every other agent stays below 10%. Their lead comes from acting better, turning what they see into interventions that yield evidence. What remains is just as telling. A third of Flint is solved by no agent, and when the strongest agent fails, it usually has an idea but holds on to it too long. Changing only the part of the task that its first hypothesis got wrong turns 7 of 10 such failures into successes. The next step for embodied fluid intelligence is therefore not only to see and act in 3D, but to doubt well, seeking the interventions that could prove one's own hypothesis wrong. Because Flint is written by coding agents as generators rather than as fixed tasks, it can grow along with the models it measures: as coding agents improve, they can author richer mechanisms and fresh variants on demand, keeping the benchmark updated for the agents it tests. We hope Flint will serve as a playground where machines, like the cockatoo, learn to work out a new world by playing with it.

AI USE STATEMENT

AI systems are central to this work in two roles, both described in the main text. First, coding agents built the environment engine, the interactive asset library, and all 60 task generators through the workflow in Sec. 3. Every generator passed automated verification tests and human review before it entered Flint (Sec. 4). Second, multimodal LLM agents are the subjects of our evaluation (Sec. 5). Beyond these roles, we used LLM-based assistants to help write analysis and plotting scripts and to edit the wording of the manuscript. All research ideas, experimental designs, and conclusions are the authors' own. The authors checked all AI-assisted content and take full responsibility for it.

ETHICS STATEMENT

Flint consists entirely of synthetic 3D environments built from procedurally generated geometry and physics, and contains no personal, copyrighted, or otherwise sensitive data. The only human involvement was playtesting the engine and judging rendered tasks for solvability, visibility, and variant validity; no personal information was collected from the participants. The benchmark is evaluation-only and is intended to measure how agents acquire unfamiliar mechanisms, not to train them. Better embodied fluid intelligence may eventually make physical agents more capable and more autonomous, which carries both benefits and risks. We believe that measuring these abilities openly and carefully, and understanding where they fail, is an important part of developing such agents responsibly.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe the task creation framework in Sec. 3, and the curation and validation procedure, including the automated tests and the human review criteria, in Sec. 4. The evaluation protocol in Sec. 4 specifies the prompts (L1 and L0), the action and observation interface, and the time and interaction limits, and Sec. 5 gives the evaluated model versions, the cost caps, and the definition of SIE. Appendix G and Appendix H describe every mechanism category and task generator, Appendix F defines the interaction measures and reports them with bootstrap confidence intervals, and Appendix I shows representative episodes. We will release the engine, the asset library, all 60 generators with the seeds that produce the 180 tasks, the evaluation harness with its prompts, and the recorded agent trajectories. Because the closed-source models we evaluate are served through APIs that may change over time, their scores may not be exactly reproducible later; the released trajectories keep every reported result open to inspection.

REFERENCES

Ruichuan An, Sihan Yang, Ziyu Guo, Wei Dai, Zijun Shen, Haodong Li, Renrui Zhang, Xinyu Wei, Guopeng Li, Wenshan Wu, and Wentao Zhang. GENIUS: Generative fluid intelligence evaluation suite. *arXiv preprint arXiv:2602.11144*, 2026.

ARC Prize Foundation. ARC-AGI-3: A new challenge for frontier agentic intelligence. *arXiv preprint arXiv:2603.24621*, 2026.

Alice MI Auersperg, Alex Kacelnik, and Auguste MP von Bayern. Explorative learning and functional inferences on a five-step means-means-end problem in goffin's cockatoos (cacatua goffini). *PloS one*, 8(7):e68979, 2013.

Anton Bakhtin, Laurens van der Maaten, Justin Johnson, Laura Gustafson, and Ross Girshick. PHYRE: A new benchmark for physical reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 511–520. PMLR, 2018.

Ricardo Cabello and Three.js Authors. Three.js: A javascript 3d library. https://threejs.org/, 2010. Accessed: 2026-09-22.

Raymond B. Cattell. Theory of fluid and crystallized intelligence: A critical experiment. *Journal of Educational Psychology*, 54(1):1–22, 1963. doi: 10.1037/h0046743.

Yiteng Chen, Huiping Zhuang, Wenbo Li, Shiyi Wang, Xiangyu Zhao, and Qingyao Wu. ManipEvalAgent: Promptable and efficient evaluation framework for robotic manipulation policies. In *International Conference on Learning Representations (ICLR)*, 2026.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2048–2056. PMLR, 2020.

Matthew Crosby, Benjamin Beyret, Murray Shanahan, José Hernández-Orallo, Lucy Cheke, and Marta Halina. The animal-ai testbed and competition. In *Proceedings of the NeurIPS 2019 Competition and Demonstration Track*, volume 123 of *Proceedings of Machine Learning Research*, pp. 164–176. PMLR, 2020.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. Procthor: Large-scale embodied ai using procedural generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Dimforge. Rapier: A high-performance 2d and 3d physics engine. https://rapier.rs/, 2020. Accessed: 2026-09-22.

Maxence Faldor, Jenny Zhang, Antoine Cully, and Jeff Clune. OMNI-EPIC: Open-endedness via models of human notions of interestingness with environments programmed in code. In *International Conference on Learning Representations (ICLR)*, 2025.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. MineDojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 18343–18362, 2022.

Yining Hong, Jiageng Liu, Han Yin, Manling Li, Leonidas Guibas, Li Fei-Fei, Jiajun Wu, and Yejin Choi. ESI-Bench: Towards embodied spatial intelligence that closes the perception–action loop. *arXiv preprint arXiv:2605.18746*, 2026.

John L. Horn and Raymond B. Cattell. Refinement and test of the theory of fluid and crystallized general intelligences. *Journal of Educational Psychology*, 57(5):253–270, 1966. doi: 10.1037/h0023816.

Wenqi Huang, Charley Lee, Leonard Tng, and Serena Ge. DeepSWE: Measuring frontier coding agents on original, long-horizon engineering tasks. *arXiv preprint arXiv:2607.07946*, 2026.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Aniruddha Kembhavi, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, et al. Behavior-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *Conference on Robot Learning*, pp. 80–93. PMLR, 2023.

Shiqian Li, Kewen Wu, Chi Zhang, and Yixin Zhu. I-PHYRE: Interactive physical reasoning. In *International Conference on Learning Representations (ICLR)*, 2024.

Seungwon Lim, Sungwoong Kim, Jihwan Yu, Sungjae Lee, Jiwan Chung, and Youngjae Yu. VisEscape: A benchmark for evaluating exploration-driven decision-making in virtual escape rooms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 16020–16047. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-main.810.

Aleksei Petrenko, Erik Wijmans, Brennan Shacklett, and Vladlen Koltun. Megaverse: Simulating embodied agents at one million experiences per second. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8556–8566. PMLR, 2021.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10740–10749, 2020.

Konstantinos Voudouris, Ben Slater, Lucy G Cheke, Wout Schellaert, José Hernández-Orallo, Marta Halina, Matishalin Patel, Ibrahim Alhas, Matteo G Mecattaf, John Burden, et al. The animal-ai environment: A virtual laboratory for comparative cognition and artificial intelligence research. *Behavior Research Methods*, 57(4):107, 2025.

Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. GenSim: Generating robotic simulation tasks via large language models. In *International Conference on Learning Representations (ICLR)*, 2024a.

Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. RoboGen: Towards unleashing infinite data for automated robot learning via generative simulation. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51936–51983. PMLR, 2024b.

Ziyue Wang, Yurui Dong, Fuwen Luo, Minyuan Ruan, Zhili Cheng, Chi Chen, Peng Li, and Yang Liu. How do multimodal large language models handle complex multimodal reasoning? placing them in an extensible escape game. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4807–4817, 2025. doi: 10.1109/ICCV51701.2025.00457.

Hongchi Xia, Xuan Li, Zhaoshuo Li, Qianli Ma, Jiashu Xu, Ming-Yu Liu, Yin Cui, Tsung-Yi Lin, Wei-Chiu Ma, Shenlong Wang, Shuran Song, and Fangyin Wei. Sage: Scalable agentic 3d scene generation for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22358–22368, June 2026.

Jihan Yang, Shusheng Yang, Anjali W. Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

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

Chi Zhang, Feng Gao, Baoxiong Jia, Yixin Zhu, and Song-Chun Zhu. RAVEN: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5317–5327, 2019.

Le Zhang, Jihan Yang, Soundarya Krishnan, Jimit Majmudar, Xiou Ge, Prasoon Puri, Prathamesh Nandkishor Saraf, Shruti Bhargava, Dhivya Piraviperumal, Yinan Ling, Cindy Pan, Hong Yu, Aishwarya Agrawal, and Bo-Hsiang Tseng. From where things are to what they are for: Benchmarking spatial-functional intelligence in multimodal LLMs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

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*, 2024.

A Automatic Tests in Flint's Creation Framework

The automatic tests assess whether a generator produces valid, playable tasks whose solutions depend on the intended mechanisms.

Task validity and playability. We check that scene specifications are consistent, geometry is free of unintended intersections, controls are accessible, and task-relevant cues are visible. We also test simulation stability, the clarity of physical outcomes, and whether the intended solution reaches the goal through the available player actions.

Dependence on the intended mechanism. We check that solving a task requires using its intended mechanism and screen for information that inadvertently reveals the answer. Additional tests detect tasks that can be solved by simple strategies that ignore the mechanism, such as random interactions or setting every control to its limit.

Variation across tasks. We test whether generators produce valid instances across the requested parameter settings. We also screen for unintended reuse of the same solution across different instances.

B Details of Familiarity Tests

We assess agents' familiarity with task mechanisms through quizzes, without allowing them to interact with the room. For each quiz, we provide five views covering the room and ask the agent to identify interactable objects and the effects of left-clicking them. Human raters compare each answer with the ground-truth objects and their interaction effects. We score object identification and effect prediction separately, dividing the number of correct identifications or predictions by the number of objects in the union of the answer and ground truth. We average each score over all quizzes for an agent and report the results in Fig. 8.

C Remaining Failures of GPT-6 Astra in L0

Solving some tasks in Flint requires precise spatial perception and manipulation, as well as efficient planning, even when the mechanism is given. Under L0, GPT-6 Astra still fails 37 tasks spanning 17 generators. They are concentrated in Support construction (11 tasks), Gripper & arm (9), and Lever & tipping balance (8), which together account for $28/37$ failures (75.7%). These tasks require arranging contacts, supports, or joint movements so that objects reach a particular pose.

The recorded trajectories suggest difficulty placing objects precisely. In an instance of *gripper clearance*, GPT-6 Astra alternates between blocks and repeatedly adjusts their placement, reporting offsets from the jaws and obstruction by the fixed plates. In another instance of *cable fulcrum*, it revisits the beam's depth alignment and the support height after finding a cable setting that appears to bring the beam close to the reference. Across the 37 failed episodes, the click acceptance rate averages 89.2%, with each episode weighted equally. Thus, interactions often take effect while failing to produce the required spatial arrangement; acceptance alone does not establish progress toward the goal.

The trajectories also suggest difficulty correcting mistakes without undoing earlier progress. In an instance of *gripped release*, the agent withdraws the slab and lets the beam settle, then finds that the lowered beam obstructs the bracket where it wants to reposition a support block. It proposes returning the slab and resetting the beam, but reaches the interaction limit while attempting this correction. Repeated placement adjustments and reconstruction can consume the available interactions: 28 of the 37 episodes end at the interaction limit, and the other 9 at the cost limit. These failures suggest difficulty placing objects precisely and correcting mistakes within the available budgets, even when the mechanism is given.

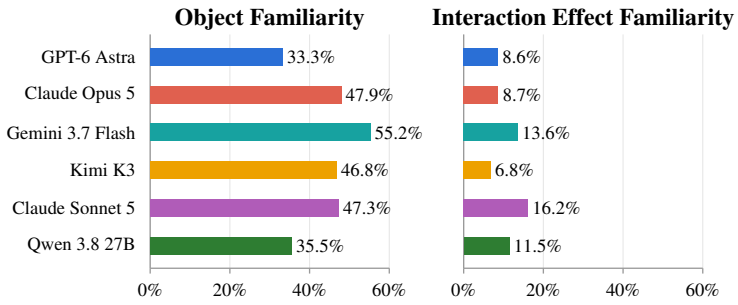

Figure 8: **Mechanism familiarity.** We measure familiarity as the accuracy of identifying interactable objects and the effects of interactions.

	Jobs	Share
Launched	80	100%
Passed automated tests	44	55%
Accepted by human tests	40	50%
Avg. cost per job		$21.58
Avg. time per job		94.5 mins

Table 1: **Task generator authoring statistics.** Outcomes of 80 generator creation jobs. Jobs whose generator does not pass the automated tests are set aside, and the rest are reviewed by a human. 91% of the generators that pass the automated tests are accepted. Cost and time are averaged over all jobs.

D Understanding Generator Creation Performance

To test the performance of generator authoring in Flint's creation framework, we launched 80 generator-creation jobs (the Subagent column in Fig. 3) and report their outcomes, costs, and durations in Tab. 1. In our actual creation, generators that pass the automated tests undergo human review before acceptance, as described in Sec. 4. For this evaluation batch, human reviewers accepted 40 of the 44 generators that passed the automated tests (90.9%).

E Cost Calculation of Different Agents

We measure inference cost in USD by summing request costs within each episode. For request t, let I_t, R_t, W_t, and O_t denote uncached input tokens, cache-read tokens, cache-write tokens, and output tokens, respectively. Input counts include image tokens, and output counts include reasoning tokens exactly once. Given the corresponding rates $p_m^I, p_m^R, p_m^W, p_m^O$ for model m, in USD per million tokens,

$$c_{m,t} = 10^{-6}\left(p_m^I I_t + p_m^R R_t + p_m^W W_t + p_m^O O_t\right), \qquad C_m = \sum_t c_{m,t}. \tag{1}$$

The table lists the evaluation rates in USD per million tokens:

Agent	Input (p_m^I)	Cache read (p_m^R)	Cache write (p_m^W)	Output (p_m^O)
GPT-6 Astra ($N_t \leq 272$k)	10.00	1.00	12.50	50.00
GPT-6 Astra ($N_t > 272$k)	20.00	2.00	25.00	75.00
Claude Opus 5.5	4.00	0.20	5.00	20.00
Claude Opus 5	5.00	0.50	6.25	25.00
Gemini 3.7 Flash	1.50	0.15	—	7.50
Kimi K3	3.00	0.30	—	15.00
Claude Sonnet 5	2.00	0.20	2.50	10.00
Qwen 3.8 27B	0.425	0.085	—	2.55

Here, N_t denotes the total number of input tokens in request t, including cached tokens. We use the standard API rates as of September 25, 2026.

F Interaction Analyses

We define the interaction measures used in Fig. 6, describe their limitations, and report additional comparisons across agents. Tab. 2 summarizes the results.

Table 2: **Interaction measures per agent**, grouped by the figure panel or analysis they support. Brackets give 95% bootstrap confidence intervals, pts denotes percentage points, and n is the number of episodes a value is computed over. All quantities are defined in the text of this appendix.

	GPT-6 Astra	Claude Opus 5.5	Claude Opus 5	Gemini 3.7 Flash	Kimi K3	Claude Sonnet 5	Qwen 3.8 27B
Solve rate at $20 (%)	60.6	44.4	7.8	7.2	1.7	1.1	0.0
ATAI by episode outcome (Fig. 6a)							
ATAI, all episodes	3.96	8.10	14.94	14.94	15.31	17.23	20.55
ATAI, solved episodes	3.24	5.98	4.78	4.98	1.71	5.98	–
n	109	80	14	13	3	2	0
ATAI, failed episodes	5.07	9.80	15.80	15.71	15.54	17.36	20.55
Turns, moves and clicks							
Turns and moves per click	2.7	5.5	6.4	4.2	3.9	4.8	5.2
failed episodes	2.8	5.9	6.7	4.2	4.0	4.8	5.2
Click acceptance rate (%)	79	80	70	54	50	53	48
failed episodes	76	78	68	53	49	53	48
Long periods without an accepted interaction (Fig. 9a)							
Long periods, share of all periods (%)	0.6	2.9	6.1	5.2	4.2	6.9	6.4
first half of the episode	0.8	2.6	6.5	5.8	2.1	5.8	3.6
second half of the episode	0.4	3.2	5.9	4.8	6.5	7.8	8.9
Share of actions in long periods (%)	7	18	38	50	29	41	39
Median period length, excl. immediate repeats (actions)	2	5	6	4	5	7	6
Clicks right after a turn or move (Fig. 6b)							
Sent without observation (%)	91	44	54	79	55	3	4
Acceptance rate, without minus with observation (pts)	+7.9	−0.4	+0.0	−7.0	−1.7	−11.9	+1.6
95% CI	[3.1, 13.3]	[−3.6, 2.5]	[−4.5, 4.5]	[−11.0, −3.0]	[−7.0, 3.7]	[−23.1, −0.4]	[−16.9, 19.6]
episodes compared	153	175	171	169	154	53	24
Click acceptance rate over the first 20 clicks (Fig. 6c)							
Acceptance rate, clicks 1–5 (%)	59	63	60	57	51	50	44
Acceptance rate, clicks 16–20 (%)	78	82	74	55	58	55	44
change (pts)	+19.9	+18.8	+14.4	−1.5	+6.2	+4.4	−0.5
95% CI	[14.0, 25.4]	[13.9, 23.6]	[5.2, 23.9]	[−6.7, 3.7]	[−4.8, 17.1]	[−2.4, 11.0]	[−15.4, 14.4]
n	153	150	61	172	42	100	43
Change, failed episodes only (pts)	+14.6	+17.8	+14.5	−0.9	+6.2	+4.6	−0.5
95% CI	[5.4, 23.1]	[11.7, 23.6]	[4.7, 24.0]	[−6.5, 4.7]	[−5.2, 18.1]	[−2.0, 11.3]	[−14.9, 14.4]
n	71	99	55	163	42	99	43
Repeating a click that did nothing (Fig. 9b)							
Misses followed by the same click (%)	0.8	2.1	4.7	12.7	10.0	12.9	12.1

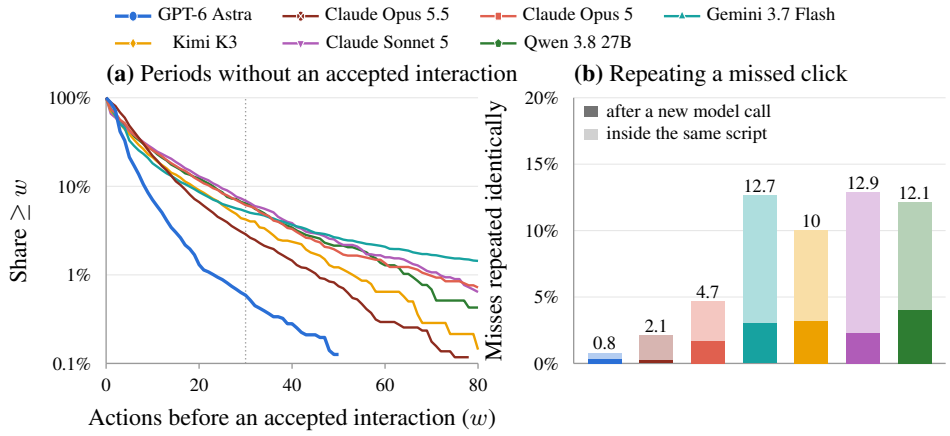

Figure 9: **How agents get stuck. (a)** For all actions taken before an accepted interaction, the share of periods that last w actions or more; the dotted line marks the 30-action threshold for a long period. **(b)** Share of misses, clicks that hit nothing, that the agent follows with exactly the same click. The dark part of each bar counts repeats sent after a new model call, when the agent could have seen the miss, and the light part counts repeats sent in the same script. All 972 repeats missed again.

Data and conventions. Every agent plays every task once. For each episode we replay the recorded actions in the engine, which tells us how the environment resolved every click. Turns, moves and clicks count as actions; panoramas, which only show the surroundings, do not. A click becomes an accepted interaction when the environment acts on it, for example by pressing a button, picking up an

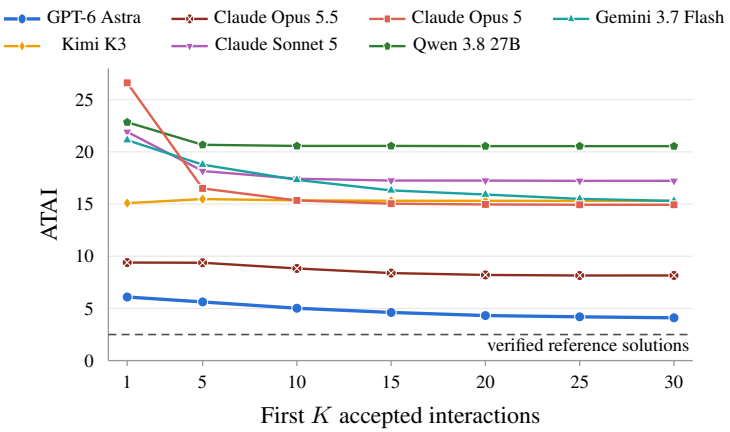

Figure 10: **ATAI over the first K accepted interactions.** ATAI of the first K accepted interactions of each episode, averaged over all episodes. The dashed line gives the ATAI of the verified reference solutions of the same tasks (2.5 actions per accepted interaction). An episode with fewer than K accepted interactions contributes its full-episode ATAI, so the curves of the weaker agents flatten mostly because their episodes run out of accepted interactions: at $K = 30$, this applies to 92–100% of the episodes of Claude Opus 5, Kimi K3, Claude Sonnet 5 and Qwen 3.8 27B.

object or putting down the one the agent carries. Otherwise the click either hits nothing, which we call a miss, or is refused by the environment. Only clicks count toward the interaction limit (Sec. 4), whether or not they are accepted. Clicks that are not accepted are therefore part of the cost of the next accepted interaction, and we count them among the actions taken before it. The click acceptance rate of a group of clicks is the number of its clicks that become accepted interactions divided by the number of clicks in the group. An episode is solved when it succeeds within the $20 cost cap, and an agent's solve rate is the fraction of the 180 tasks it solves. All confidence intervals are 95% bootstrap intervals, and differences between two percentages are given in percentage points (pts).

ATAI by episode outcome (Fig. 6a). The ATAI of an episode is the average number of actions taken before each of its accepted interactions. The actions after the last accepted interaction form one more group, and an episode without any accepted interaction is charged its full length. An agent's ATAI is the average over its episodes. Ranking the seven agents from lowest to highest ATAI reproduces their ranking by solve rate almost exactly (Spearman $\rho = -0.96$). The same relation holds within the Opus family: Claude Opus 5.5 has lower ATAI and a higher solve rate than Claude Opus 5 (Tab. 2). The agents differ most in the episodes they fail. ATAI is much closer across agents on solved tasks; on failed tasks, it ranges from 15.5 to 20.6 for the weaker agents, compared with 9.8 for Claude Opus 5.5 and 5.1 for GPT-6 Astra. Solved-episode averages for the weaker agents rest on few tasks, especially for Kimi K3 and Claude Sonnet 5, which solve only 3 and 2; Qwen 3.8 27B solves none. Because each agent solves a different set of tasks, these averages could in principle reflect task difficulty rather than the agents themselves. The gap persists in two comparisons on matched tasks. First, GPT-6 Astra solves every task that any agent other than Claude Opus 5.5 solves, and on the tasks it shares with another agent it still needs about half as many actions per accepted interaction as Claude Opus 5.5 (2.91 *vs*. 5.96 over 70 tasks), Claude Opus 5 (2.32 *vs*. 4.78 over 14 tasks) and Gemini 3.7 Flash (2.80 *vs*. 4.98 over 13 tasks). Second, of the 71 tasks that GPT-6 Astra fails, Claude Opus 5.5 solves 10 and no other agent solves any. On each of these tasks we divide the ATAI of GPT-6 Astra by the geometric mean of the other agents' ATAI, and the ratio is below one on 68 of the 71 tasks, with a median of 0.31. These comparisons show that GPT-6 Astra maintains lower ATAI on matched tasks, including those it fails.

Turns, moves and clicks. Since ATAI counts turns, moves and clicks, it can be split into two parts: the number of turns and moves an agent makes per click, and its click acceptance rate. Turns and moves per click is the total number of turns and moves an agent makes, including moves blocked by an obstacle, divided by its total number of clicks. The acceptance rate here covers all clicks, not only the first 20 as in Fig. 6c. We compute both over all of an agent's episodes, and over its failed episodes only. GPT-6 Astra and Claude Opus 5.5 have similar click acceptance rates (79% and 80%),

but Claude Opus 5.5 makes about twice as many turns and moves per click (5.5 versus 2.7). The remaining agents are behind on both measures. These patterns also hold in failed episodes (Tab. 2).

Long periods without an accepted interaction (Fig. 9a). We call the actions an agent takes between two consecutive accepted interactions, or between the start of the episode and its first accepted interaction, a *period*, so that every accepted interaction closes one period; the actions after the last accepted interaction of an episode do not form a period. The panel shows how often a period lasts w actions or more, and we call a period of 30 or more actions a long period. Long periods account for 29–50% of the actions in the weaker agents' periods, compared with 7% for GPT-6 Astra and 18% for Claude Opus 5.5. Measured per period, long periods are thus 7 to 12 times as frequent for the weaker agents as for GPT-6 Astra, and five times as frequent for Claude Opus 5.5. GPT-6 Astra is faster on ordinary periods too. Leaving aside immediate repeats, in which an agent clicks the same object twice in a row and so takes no action in between (12% of the accepted interactions of GPT-6 Astra, 11% of those of Claude Opus 5.5, and 24–34% of the others'), the median period lasts 2 actions for GPT-6 Astra and 4–7 for the other agents. Long periods also change over the course of an episode. Assigning each period to the first or the second half of its episode, according to whether its accepted interaction falls in the first or the second half of the episode's actions, long periods become rarer for GPT-6 Astra, from 0.8% of periods in the first half to 0.4% in the second. They change little for the two Opus models and Gemini 3.7 Flash, and become more frequent for Kimi K3, Claude Sonnet 5 and Qwen 3.8 27B (Tab. 2). For GPT-6 Astra, the decline in long periods accompanies an increase in click acceptance (Fig. 6c). The two Opus models also improve their click acceptance, but their long-period rates change little, and they spend about twice as many turns and moves on each click as GPT-6 Astra (5.5 and 6.4 against 2.7). Navigation efficiency may contribute to this difference, but the comparison across three models does not establish a causal relationship.

Clicks right after a turn or move (Fig. 6b). This panel looks at every click that directly follows a turn or a move. Such an action returns a new frame, which the agent sees only if it opens it, since all harnesses require an explicit call to view an image; panoramas play no role here. A click is sent *without observation* when the agent opens no image between the action and the click, that is, when it sends both in the same script. The agent has then planned its whole approach to the object from frames it has already seen, turning or stepping toward the object and clicking it in one go. Opening an image costs no action, so skipping it does not lower ATAI by itself. What it can indicate is how precisely an agent plans. An agent that can reach and hit an object in one planned sequence, instead of approaching it step by step and checking along the way, needs few turns and moves per click, but only if its plan is accurate, since a click that misses is not accepted and still counts toward the interaction limit. The agents run in different harnesses, so we identify these clicks with one timing rule that applies to all of them: a click that comes less than 3 seconds of active time after the preceding action was, in practice, sent in the same script. We can verify this rule against exact logs only for GPT-6 Astra, as described below; for the other agents we assume that it holds equally. Any later click was sent after a new model call, when the agent could have opened the new frame, and we say that it was sent with observation, although the agent may not have opened the frame (the logs of GPT-6 Astra, discussed below, show that it did so for 91% of these clicks). The percentages headed "w/o obs." in Fig. 6b, repeated in Tab. 2, give the share of these clicks sent without observation. For each episode we compute the click acceptance rate separately for the two kinds of click, take the difference, and average it over the episodes that contain both kinds, which are the episodes compared in Tab. 2. Both rates thus come from the same task and the same run, and the confidence intervals resample episodes. GPT-6 Astra sends 91% of these clicks without observation, and within each episode they are accepted 7.9 points more often than the ones it sends with observation. Claude Opus 5.5 sends such clicks without observation less often, while Gemini 3.7 Flash and Claude Sonnet 5 have lower acceptance rates when they do so. Claude Sonnet 5 and Qwen 3.8 27B so rarely click without observation that only 53 and 24 of their episodes contain both kinds, hence their wide intervals. Keeping only clicks after a turn, and leaving out moves, gives almost the same values (GPT-6 Astra $+9.6$, Gemini 3.7 Flash -7.3, Claude Sonnet 5 -13.5 points).

The comparison is observational, because agents choose when to look. A positive difference for GPT-6 Astra therefore does not mean that skipping the observation makes its clicks more likely to be accepted. For GPT-6 Astra, clicks without and with observation follow a miss or a refused click at similar rates (21% and 19%, respectively).

For GPT-6 Astra we can check the timing rule directly, because its session logs record every tool call and every image it opened. On the 7,328 clicks after a turn or move whose tool calls we can recover

exactly, the rule and the logs agree in 99.5% of cases, and when GPT-6 Astra sent a click after a new model call, it had opened the new frame 91% of the time. Using the exact logs instead of the timing rule, or counting a click as sent with observation only when GPT-6 Astra actually opened the new frame, leaves the difference at $+7.9$ points ($[3.1, 12.8]$ and $[3.1, 13.0]$).

Targets visible in earlier observations. We checked whether clicks that GPT-6 Astra sent without observation targeted locations visible in previously observed frames. Its episodes replay to their recorded outcome from the logged actions alone, and its tool calls touch only the files and interface given to every agent. We reconstructed the camera pose at every click from the engine's turning rules and located the point under the crosshair in the frames GPT-6 Astra had opened. Of the $5{,}952$ clicks that GPT-6 Astra made after turning away from the last frame it observed, 95% were aimed at a point visible in a frame it had observed earlier from the same position. Most of the rest target objects it had already operated, or put down an object it was carrying.

Click acceptance rate over the first 20 clicks (Fig. 6c). The panel counts every click, whether it becomes an accepted interaction, misses or is refused, and reports the click acceptance rate in blocks of five over the first 20 clicks. For each agent it uses one fixed set of episodes, those with at least 20 clicks, so that a rise or fall along the curve reflects change within the same episodes rather than a change in which episodes remain. This leaves 42 to 172 episodes per agent and favors longer episodes. The panel measures changes in click acceptance within this subset, rather than overall acceptance rates. The initial click acceptance rate of GPT-6 Astra is similar to those of the two Opus models and Gemini 3.7 Flash (Tab. 2). Over these episodes, the acceptance rate of GPT-6 Astra rises by 19.9 points from its first five clicks to its last five ($[14.0, 25.4]$; the intervals resample episodes). Both kinds of failure become rarer: misses fall from 24% to 13% of its clicks and refused clicks from 18% to 9%, which suggests that it learns both where to aim and which objects respond. Later clicks may also be easier, for example when the agent returns to an object that has already worked, so this is an interpretation rather than a direct measurement of learning. Both Opus models also improve: for Claude Opus 5.5, misses fall from 20% to 9% and refused clicks from 17% to 9%, while Claude Opus 5 improves mostly through fewer misses. The remaining agents show no clear change; their confidence intervals include zero. The result persists when we count only clicks preceded by a turn or a move, which test a new aim rather than repeat the last one (GPT-6 Astra $+20.0$, $[14.7, 25.6]$; Gemini 3.7 Flash -3.0, $[-9.0, 2.7]$). Click acceptance also improves within failed episodes. Over the failed episodes with at least 20 clicks, the acceptance rate of GPT-6 Astra rises by 14.6 points ($[5.4, 23.1]$, 71 episodes). Both Opus models also improve, while the remaining agents again show no clear change.

ATAI over the first K accepted interactions (Fig. 10) is limited by episodes with fewer accepted interactions. An episode with fewer than K accepted interactions can only contribute its full-episode ATAI, which is common for the weaker agents: at $K = 30$, it applies to 92–100% of the episodes of Claude Opus 5, Kimi K3, Claude Sonnet 5 and Qwen 3.8 27B. Their curves flatten mostly because their episodes run out of accepted interactions; this flattening does not establish that the agents stop improving. Following one fixed set of episodes, as Fig. 6c does, avoids this problem. The figure also marks the ATAI of the verified reference solutions of the same tasks, 2.5 actions per accepted interaction. Relative to this reference, GPT-6 Astra needs 1.6 times as many actions per accepted interaction, Claude Opus 5.5 3.2 times, and the other agents 6 to 8 times.

Repeating a click that did nothing. Fig. 9b shows how the agents respond to a miss. Each bar gives the share of misses that the agent follows with exactly the same click, meaning that its very next action is the same click with no turn or move in between. All 972 such repeats also missed. GPT-6 Astra repeats a miss after only 0.8% of its misses, against 2.1% for Claude Opus 5.5 and 4.7–12.9% for the other agents. The light part of each bar counts repeats sent in the same script as the miss, before the agent could see it, and the dark part counts repeats sent after a new model call, when the agent could have noticed that nothing happened. Even then, Claude Opus 5.5 repeats a miss about as rarely as GPT-6 Astra, whereas the other agents do so 5 to 11 times as often.

G Mechanism Categories

To analyze performance by mechanism (Fig. 5), we assign each of the 60 generators in Flint to one of 12 mechanism categories. Categories are assigned per generator, so all tasks sampled from a generator share its category. We place each generator according to what determines whether its goal

is achieved. Specifically, we consider what the agent controls (*e.g*. mirrors, a launcher, a force field, a gripper, or loose pieces), which physical process decides the outcome (*e.g*. optics, projectile flight, field forces, contact and stability, torque balance, or mass and spring readings), and what the success condition evaluates (*e.g*. lit receivers, the position of cargo, a mass total, floor coverage, or the final pose of a body). The visual theme and presentation of a task play no role. When the controlled device and the deciding physics suggest different categories, we follow the physics. For example, a task in which the agent moves a mirror on a slider is an optical task, because the path of the light decides the outcome. We describe the categories below, in the order used in Fig. 5, and list the generators in each.

Ballistic launch (2 generators, 6 tasks). A ball is launched once and then flies or rebounds without further input. The agent sets up the launch, or the surface from which the ball rebounds, and the resulting flight path decides the outcome. Generators: *ballistic lanes* and *bank-shot construction*.

Optical beams (6 generators, 18 tasks). The outcome depends on the path of a light beam through reflection, refraction, or splitting. The agent rotates or moves optical elements so that the beam lights one or more receivers, and in some tasks keeps a forbidden receiver dark. Generators: *beam routing*, *prism relay*, *selective illumination*, *shared reflection plane*, *split-beam coordination*, and *translating reflector*.

Field / gravity navigation (6 generators, 18 tasks). The agent moves free cargo to a goal by combining pushes from force fields or by changing the direction of gravity over time. Success depends only on where the cargo ends up. Generators: *field handoff*, *force-null dock*, *radial itinerary*, *reconfigurable maze*, *gravity frame*, and *cargo as gate*.

Mechanisms & checkpoints (3 generators, 9 tasks). The agent steps a driven mechanism, such as a gear-and-rack transmission, or a set of tokens, sockets, and buttons, through an ordered sequence of checkpoint states. Generators: *bidirectional rack*, *transmission sequencer*, and *socket relay*.

Weight & springs (4 generators, 12 tasks). Success is judged by a measured physical quantity: the total mass on a sensor, or the equilibrium height of a spring-supported platform. The agent must infer object weights, choose spring stiffness, and place loads accordingly. Generators: *weight distribution*, *intersecting weigh zones*, *load tuning*, and *spring checkpoints*.

Selective addressing (6 generators, 18 tasks). A single shared command acts on several bodies at once, and the puzzle is to arrange which bodies it reaches. The agent does so by moving ballast, parking a body at a particular position or height, placing a portable field emitter, or exploiting differences in friction. Generators: *altitude addressing*, *ballast addressing*, *boundary memory*, *travelling stencil*, *friction sieve*, and *effector cascade*.

Rolling-ball courses (5 generators, 15 tasks). The agent steers a rolling ball through a course by tilting the board, setting track switches, or configuring ramps, so that the ball reaches its resting points or target bay. Generators: *ball-track switchyard*, *configurable ball course*, *junction checkpoint tour*, *layered ball labyrinth*, and *synchronized ball courtyards*.

Packing & allocation (5 generators, 15 tasks). The agent allocates pieces from a limited inventory to bays, columns, or stations. Success is judged by coverage, containment, or fit, rather than by how a structure responds to load. Generators: *packing bays*, *skyline allocation*, *density packing*, *stability sorting*, and *crossing trestles*.

Gripper & arm (8 generators, 24 tasks). The agent operates a gripper, an articulated arm, or a linkage to grasp, carry, push, or release loads. The central difficulty is operating the effector itself. Generators: *grip and lift*, *grip load transfer*, *gripped release*, *gripper clearance*, *station commitment*, *two-tier release*, *arm corridor*, and *linkage carry*.

Contact-settled pose (4 generators, 12 tasks). A body is driven by a force field, gravity, a spring tether, or a chain until contact with a guide, brace, or crossing fixes its final pose. The agent arranges these contacts so that the body comes to rest in the target pose. Generators: *radial contour*, *braced rocking key*, *elastic guides*, and *tethered cargo*.

Support construction (7 generators, 21 tasks). The agent builds supports, decks, or stacks at the right heights so that a structure or load comes to rest in the target pose. Generators: *three-point plane*, *load-bearing construction*, *negative-space viaduct*, *scaffold exchange*, *cradle assembly*, *descending catch*, and *pulley platforms*.

Lever & tipping balance (4 generators, 12 tasks). The outcome is a torque or tipping equilibrium. The agent places counterweights or fulcrums, or arranges bodies so that they tip into mutual support. Generators: *cantilever balance*, *cable fulcrum*, *unload the key*, and *leaning pair revival*.

H TASK GENERATORS

Flint contains 180 tasks sampled from 60 task generators, three tasks per generator. This appendix describes each generator, grouped by the mechanism categories of Appendix G. For each generator, we state what the room contains, what the agent can do, and what the goal is. Tasks from the same generator share this mechanism, but differ in their concrete layout and in the controlled properties the generator exposes (Sec. 3).

H.1 BALLISTIC LAUNCH

Ballistic lanes. Two launchers fire balls at a fixed angle, each with a nine-position power dial. One lane is open and serves as a reference, while the other constrains the flight through its launch height, a raised receiver, or a hurdle or aperture the ball must clear. The goal is to land both balls on their target pads.

Bank-shot construction. A ball on a low-friction plate is fired by a launcher with a fixed speed and heading, so the shot itself cannot be aimed. The agent chooses one of four portable banks, each with a fixed angle, and seats it on the plate so that the single launch rebounds off it, passes through an opening in the far rail, and drops into a sunken basin.

H.2 OPTICAL BEAMS

Beam routing. A single beam passes through a chain of three or four mirrors, each rotating in discrete steps. The agent rotates the mirrors so that the beam reaches the goal receiver. Some tasks add screens that hide parts of the chain, rotatable absorbers unrelated to the goal, or a wrong setting that produces a near miss.

Prism relay. Three triangular prisms, each with 16 rotation positions, refract a continuous beam; two first-stage prisms feed a third. The agent must compose the refraction at the entry and exit of each prism to steer the beam onto a single receiver.

Selective illumination. An emitter, two beam splitters, and a mirror send light through a room that contains an absorbing plate and three or four receivers. The agent rotates the four optical elements so that the useful light travels around the plate to the target receivers, while a forbidden branch stays dark.

Shared reflection plane. Two emitters strike a single mirror mounted on a hinge, so turning the hinge changes both reflections at once and also shifts the points where they reflect. One reflected beam travels directly to its receiver, and the other travels to a second mirror that rotates independently. The agent must find the hinge stop and mirror setting that light both receivers at the same time.

Split-beam coordination. A beam from one emitter passes an upstream mirror and then a tree of three splitters that feeds four receivers. The agent must infer how the splitters depend on one another and set all four plates so that every receiver is lit at once.

Translating reflector. A mirror with a fixed orientation rides a carriage along a stepped rail. Moving the carriage decides whether the mirror intercepts the beam and where the reflection begins. A second, rotatable mirror then redirects the reflected beam onto the receiver.

H.3 FIELD / GRAVITY NAVIGATION

Field handoff. One or two cubes, which cannot be picked up, sit in a walled enclosure with baffles. Two local force fields, each set to neutral or to one of four horizontal directions, push the cargo within their regions. The agent composes these pushes to hand each cube from one region to the next until it rests in its target outline.

Force-null dock. A cube sits in a walled pen whose baffles form a winding path toward a docking corridor. Two force fields overlap near the dock and add their pushes where they overlap. The agent chooses the two settings whose sum drives the cube to the dock, where the combined force vanishes and holds it in place.

Radial itinerary. A cargo sits in a tray between two radial field sources, each of which can attract, repel, or stay off. The agent must move the cargo through an ordered list of checkpoint regions around the two sources, recorded by lamps above them, and finally into a destination region.

Reconfigurable maze. The agent drives a cargo with a force field through a corridor maze whose layout changes with a sliding gate. The gate must first be retracted so that the cargo can cross, and then restored so that it serves as the face that stops the cargo at its target.

Gravity frame. One to three loads sit on shelves inside a translucent enclosure. The agent selects one of eight gravity directions at a time, and the loads slide and fall until they settle. The goal is to route every load through the shelf chambers into its own target outline, with all loads in place at once.

Cargo as gate. Two blocks of different widths are driven by the same directional field through a hall with a narrow throat. The agent parks one block in a bay where its body closes the throat and stops the other block at its target, and then moves the first block on to its own target.

H.4 Mechanisms & Checkpoints

Bidirectional rack. Clicking the gears of a transmission moves a rack-mounted pusher back and forth between two loads. The agent must push the two loads in opposite directions onto their elevated targets, so that both are in place at the same time.

Transmission sequencer. Coupled gears position a rack pointer, and three stations each have a button and a board that records a checkpoint. The agent must bring the pointer to each station's target setting and press its button. This requires combining two gear inputs that advance the pointer by different steps and in opposite directions.

Socket relay. Four stations each contain a socket, a button, and a tilting checkpoint post, and a set of portable tokens fits every socket. A press records a checkpoint only under the right token arrangement, and some records persist while others reset. The agent must move the tokens between sockets to record all four checkpoints, shown on a completion board, while extra stations respond to presses without counting toward completion.

H.5 Weight & Springs

Weight distribution. Four cubes of unknown weight, sometimes with two extra candidates, must be distributed over pads watched by two mass sensors. The agent must infer the weights and place the cubes so that both sensors read their displayed targets at once. In some tasks, a middle pad is counted by both sensors.

Intersecting weigh zones. Five cubes can be placed on six seats watched by three overlapping sensing regions, each with its own scale and displayed target. Because one seat can count toward more than one scale, the agent must find a single placement that satisfies all three totals.

Load tuning. A spring platform has five stiffness settings, and pressing a trial button lets it settle under the loads placed on it. The agent must choose the right loads among up to eight pieces and the right spring setting, so that the platform settles at the height of a target outline.

Spring checkpoints. A spring platform with three stiffness settings must settle at three different target heights, each confirmed with its own button. The agent recomposes the same set of loads, whose relative masses can be judged only from their sizes, and changes the stiffness to reach each height.

H.6 Selective Addressing

Altitude addressing. A cargo sits in a lattice of bays between a floor deck and a ceiling, with a ledge over part of the room. Two force fields share one direction control, but each acts only within its own

range of heights. The agent uses a gravity override to park the cargo at the height where the right field acts, so that the single control moves it to the goal.

Ballast addressing. Three sleds sit in separate cells under one shared directional field. Portable weights increase a sled's friction without adding drive, so a loaded sled stays put under a push that moves the others. The agent moves the weights to decide which sleds each command reaches and brings every sled to its target.

Boundary memory. Two cubes share a tray covered by two force fields, whose regions meet at a wall with a single opening. A cube pushed out of a field's region no longer responds to that field. The agent uses this to hold one cube in place while commanding the other, so that both reach their targets.

Travelling stencil. Three cubes sit in three parallel channels, and a portable emitter projects a directional field over the channel where it is set down. The agent carries the emitter from channel to channel to push each cube to its target, and finally returns the emitter to its own bay.

Friction sieve. Three cargos with low, medium, and high friction sit in parallel lanes under one field. A given push strength moves only the cargos whose friction it overcomes, and a single portable brace can stop a cargo that the push would otherwise move. The agent combines push strengths and brace placements to bring each cargo to its target.

Effector cascade. One fixed field pushes two slabs along perpendicular channels, and each slab carries its own field that moves with it. Where a slab stops therefore determines which bodies its field reaches next. The agent must position the slabs so that this chain of fields brings every body to its target.

H.7 Rolling-Ball Courses

Ball-track switchyard. A ball rolls on a tilting board with switchable track barriers. Four three-way controls each open one of three passages at two separate places on the board, and the controls lock once the board is tilted. The agent must set the controls in advance and then tilt the board so that the ball comes to rest in the marked bay.

Configurable ball course. A ball runs down a course of three or four ramps, each with a seven-position direction dial. The agent sets the ramps so that each one receives the ball from the previous ramp and sends it on toward the target, and then releases the ball.

Junction checkpoint tour. A ball moves on a fixed track, steered by four direction buttons. At shared junctions, the places where the ball can stop depend on the direction from which it arrives. The agent must steer the ball through four checkpoints and then return it home.

Layered ball labyrinth. Tilting the board in one of four directions sends a ball from one stable resting point to the next, through corridors on two offset decks. A one-way drop connects the decks, so the agent must plan the sequence of tilts that brings the ball to its target.

Synchronized ball courtyards. Two balls in separate courts respond to the same five-button direction pad. The agent must find a sequence of shared commands that parks both balls in their bays.

H.8 Packing & Allocation

Packing bays. The agent combines portable cuboids to cover the floors of two bays. Every piece must be released, settled, and fully inside a bay, and together the pieces must cover enough of each bay floor.

Skyline allocation. Three narrow columns of different heights must each lift a flat cap to a common height, inside a marked region. The agent allocates riser pieces of different heights and widths among the columns, and a cap stays in place only if it rests on a piece wide enough to carry it.

Density packing. Six dominoes with masses one to six, in two shapes, must cover two transparent bays, each with a scale and a target load. The agent must cover both bays, match both displayed loads, and leave every piece fully inside a bay or the staging area.

Stability sorting. Portable shapes, including wedges, must be placed on two or three stands. Whether a shape stays in place depends on its stability under gravity on that stand, so the agent must work out which shape belongs on which stand.

Crossing trestles. Two long beams rest on four pedestals of different heights and must end in two target outlines that cross each other at different heights. The agent raises each pedestal to its beam by stacking props from one shared, limited inventory.

H.9 Gripper & Arm

Grip and lift. Two payloads sit between gripper jaws that close independently but share one lift. The agent must grip both payloads and raise the shared lift so that both reach their elevated targets at the same time.

Grip load transfer. A gripper travels along a track, with controls for jaw opening, lift, and position. A broad carrier covers the cargo beneath it, so the agent must first move the carrier to free the cargo and then place both on separate rests.

Gripped release. A beam rests on a fixed support and on a prop held between the jaws of a gripper. The agent first seats a catcher block on one of several brackets, then grips the prop and pulls it away, so that the beam tips onto the catcher and settles in the target pose.

Gripper clearance. The agent chooses one of several portable payloads of different depths, loads it into a gripper, and lifts it. The lifted payload must pass a fixed aperture, so only a payload of the right size succeeds.

Station commitment. A gripper travels along a shelf that holds three loads of equal width and different depths. Each load must reach the station its outline asks for, but the gripper cannot pass the other loads while carrying one, so the agent must plan the order of moves.

Two-tier release. A gripper with stepped jaws carries two loads at once: one held by the grip pads, and one riding on ledges above them. Opening the jaws drops only the gripped load, so the agent must plan where each load is released.

Arm corridor. A two-joint horizontal arm, with five positions per joint, pushes a free cargo through a corridor bounded by low rails. The agent must sequence the joint settings so that the arm pushes the cargo into its target pocket.

Linkage carry. A four-bar linkage with seven crank settings moves a frame along a curved path. The agent places a portable cube on the frame, sometimes on spacers, and turns the crank to carry it to an elevated target.

H.10 Contact-Settled Pose

Radial contour. A radial field can attract or repel a cylindrical cargo on a pad. The agent chooses one of several guide blocks, each with its own orientation, and places it on a lattice of seats. The field then presses the cargo against the guide, and the cargo settles where the guide meets a stop face.

Braced rocking key. A cuboid cargo sits between two guide rails on a deck. The agent selects a gravity direction, which presses the cargo against one rail and moves it along that rail, and places a portable stop face on a lattice. The cargo rocks along the rail until the stop face brakes it, and the two contacts fix its final pose.

Elastic guides. A load is attached by a spring to a carriage with seven positions. The agent places low guides against the lower body of the load and moves the carriage, so that the spring draws the load along the guides until it settles in the target.

Tethered cargo. A carriage pulls a chain-linked cargo across three crossings, some of which lack a supporting surface. The agent must first complete the crossings with movable blocks and then pull the cargo into place with the carriage.

H.11 Support Construction

Three-point plane. The agent stands a chosen column on each of three footings and then lowers a broad plate onto the column tops. The plate settles into the plane defined by the three column heights, and it must match an outlined target in both height and tilt.

Load-bearing construction. The agent places piers and deck pieces to build a bridge over two bays, and then rolls a launched ball across it. The ball crosses only if every support has the right height.

Negative-space viaduct. The agent builds elevated decks from portable columns and slabs, level with two fixed ledges. A force field then slides caps across the decks, while the space beneath them must stay clear for a third cargo to pass.

Scaffold exchange. Two decks rest on two portable supports. The agent must replace these supports with others and return the originals to matching sockets, while keeping the decks aligned throughout.

Cradle assembly. The agent places two opposed portable wedges so that together they hold a released upright cylinder at an elevated target.

Descending catch. A winch lowers a platform that carries two bars. Each bar is caught by a fixed ledge on one side and, on the other, by a beam that the agent seats on a bracket; only a beam of the right thickness leaves its bar level. The agent builds both catch seats and then lowers the platform once, so that each bar is left on its own seat.

Pulley platforms. A pulley raises a platform through five stops. The agent stacks portable spacers on the platform and chooses the pulley stroke, so that a narrower cargo block placed on top reaches an elevated target.

H.12 Lever & Tipping Balance

Cantilever balance. A portable deck rests on a narrow fulcrum and carries a load on its overhanging arm. The agent must choose and place counterweight blocks, whose size shows their weight, so that the deck and the load both settle inside their target outlines at the same time.

Cable fulcrum. A loose beam rests with one end on a winch-driven platform. The agent seats a portable fulcrum on one of four pads and chooses how far to wind the cable, so that the beam settles at the tilt of a target outline.

Unload the key. A loose beam rests on two keys that stand on a low-friction deck. The agent places counterweights on the beam so that it tips onto fixed rests and lifts off the keys, after which a field can slide each key out to its target.

Leaning pair revival. Two thick slabs are released standing on opposite plinth edges and tip toward each other. The agent seats one portable cradle to catch them, and the slabs must end up held by their mutual contact at the target lean.

I Qualitative Rollouts

We show a selection of agent rollouts on Flint. Each of the following page is one rollout, and each row is one step, chosen automatically where the agent's click changed the world or it wrote substantial reasoning. The first and last steps are always shown. The left column renders the room from a fixed third-person camera that the agent never sees. It overlays the path walked so far, the clicks so far, and the agent's current position and field of view. Anything that changed since the previous row, such as a rotated mirror, a moved block or a rerouted beam, is highlighted in yellow. The middle column is the exact frame the agent received after that action. The right column is the model's reasoning for that action, or the text it output where the model exposes no reasoning. The text is verbatim and truncated to the height of the row. The banner at the bottom of each page gives the outcome.

I.1 ROLLOUTS OF GPT-6 ASTRA

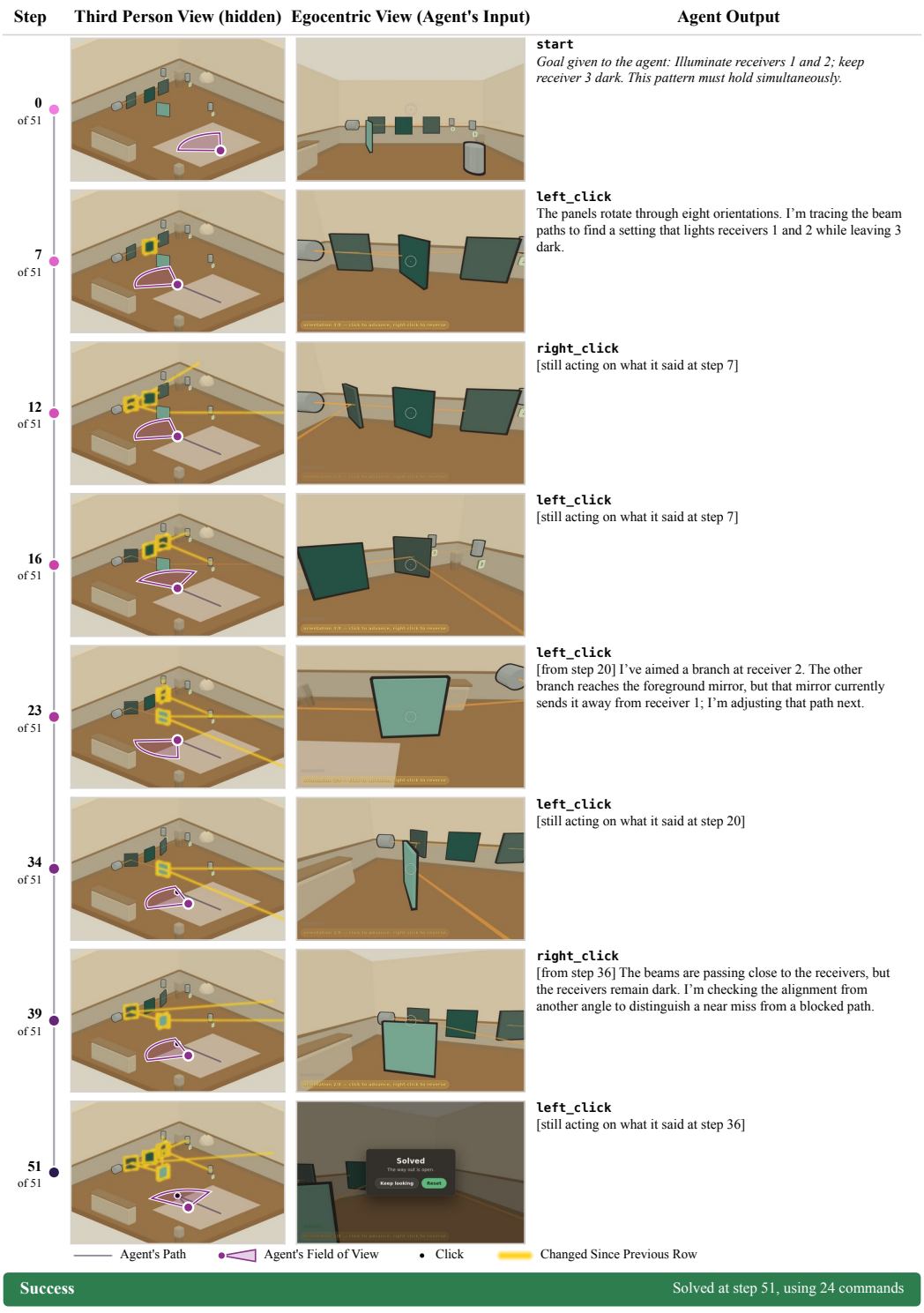

Figure 11: **GPT-6 Astra** on `selective-illumination` (random variant), a success. Goal: *Illuminate receivers 1 and 2; keep receiver 3 dark. This pattern must hold simultaneously.* Layout as described in Appendix I.

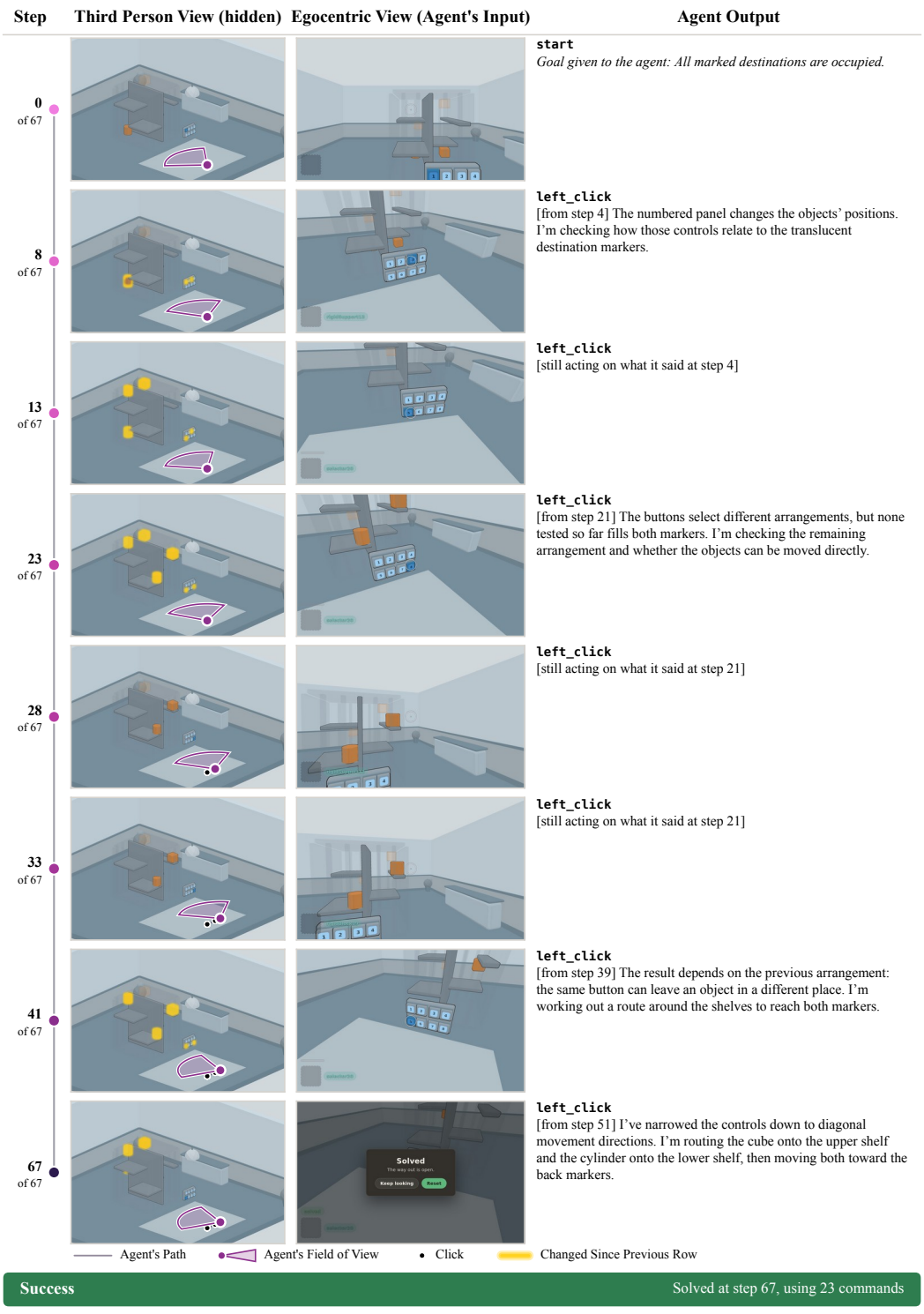

Figure 12: **GPT-6 Astra** on `gravity-frame` (random variant), a success. Goal: *All marked destinations are occupied.* Layout as described in Appendix I.

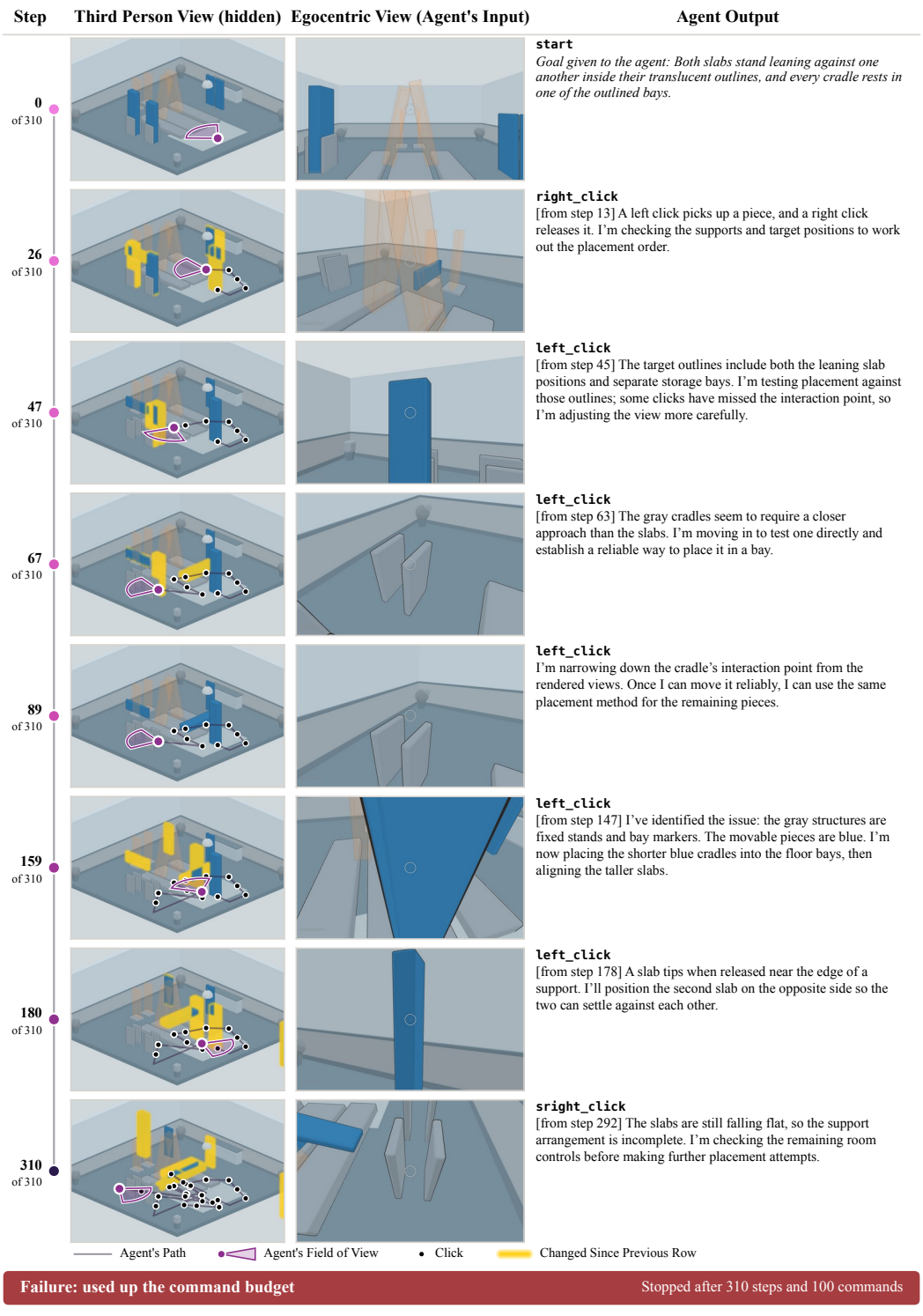

Figure 13: **GPT-6 Astra** on `leaning-pair-revival` (simple variant), a failure (command budget used up). Goal: *Both slabs stand leaning against one another inside their translucent outlines, and every cradle rests in one of the outlined bays.* Layout as described in Appendix I.

I.2 ROLLOUTS OF CLAUDE OPUS 5

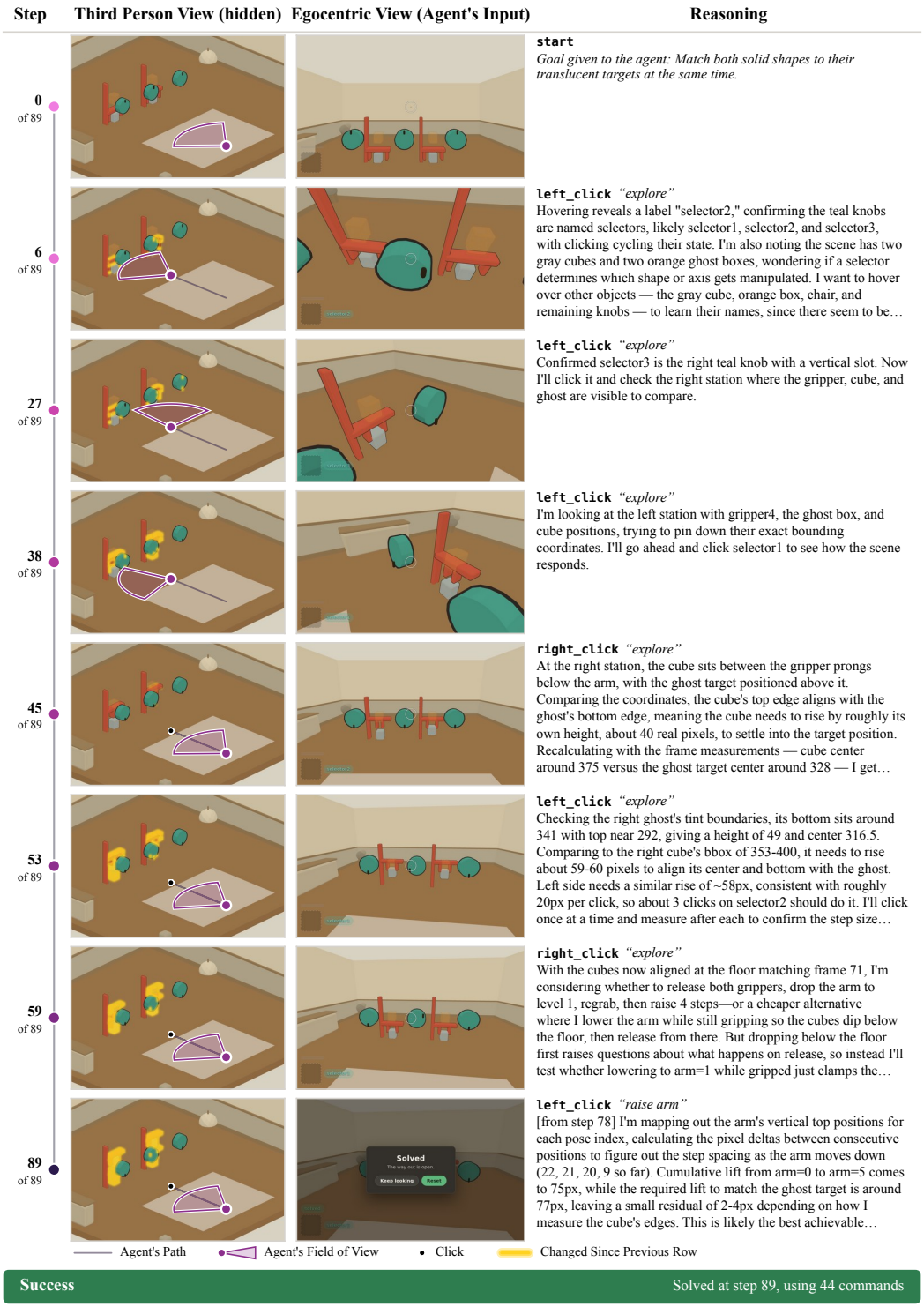

Figure 14: **Claude Opus 5** on `grip-and-lift` (random variant), a success. Goal: *Match both solid shapes to their translucent targets at the same time.* Layout as described in Appendix I.

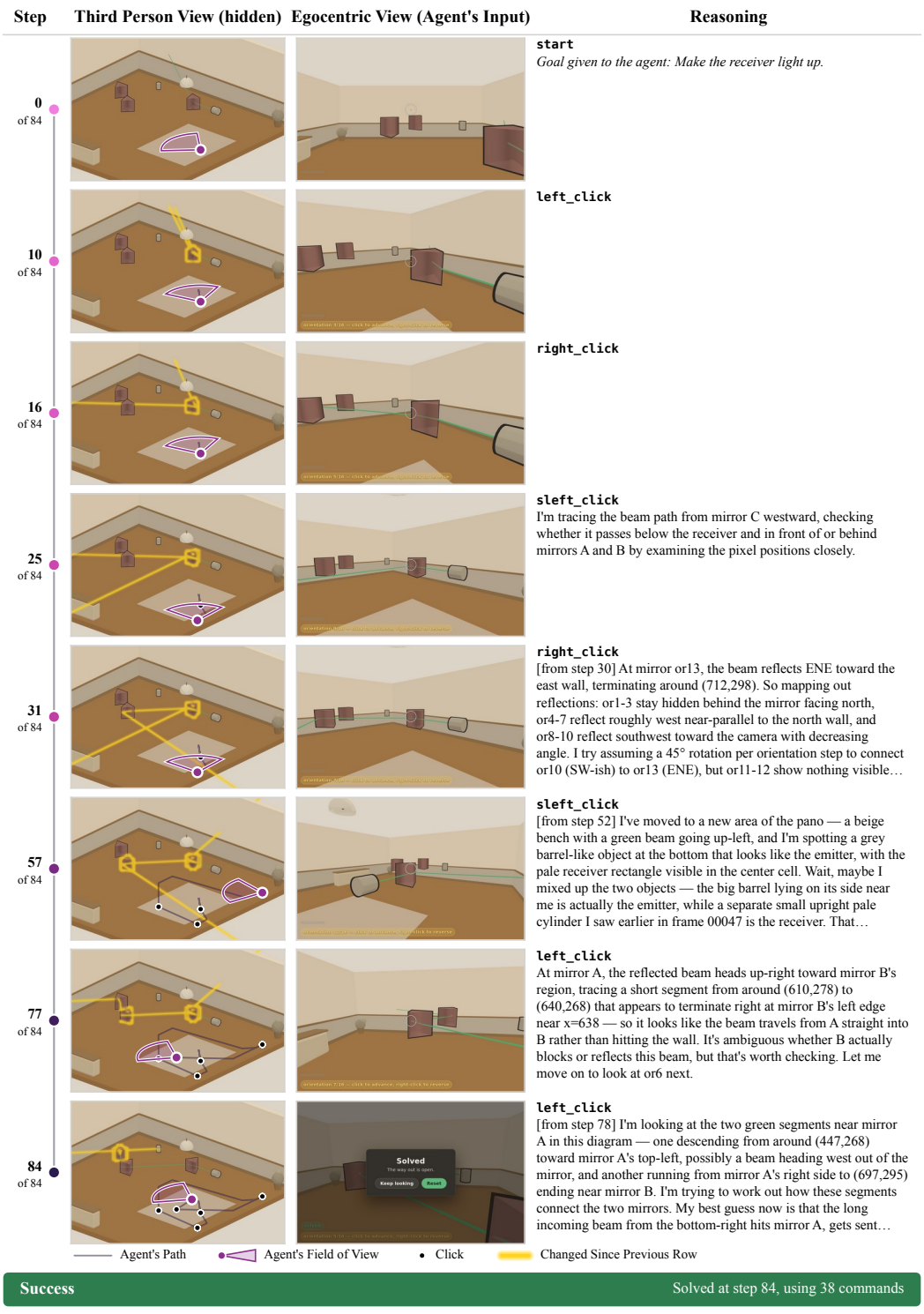

Figure 15: **Claude Opus 5** on `prism-relay` (random variant), a success. Goal: *Make the receiver light up.* Layout as described in Appendix I.

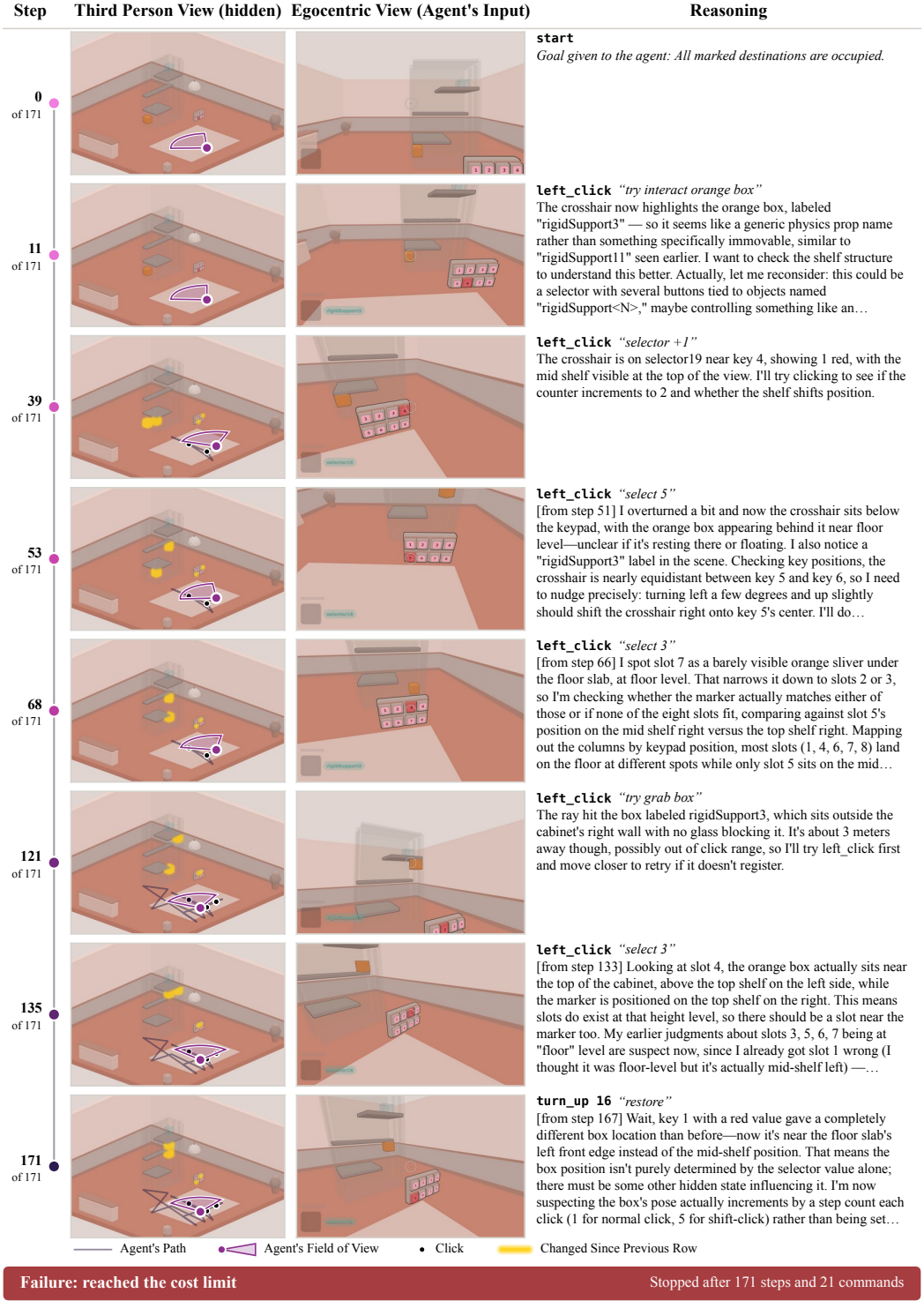

Figure 16: **Claude Opus 5** on `gravity-frame` (simple variant), a failure (cost limit reached). Goal: *All marked destinations are occupied.* Layout as described in Appendix I.

I.3 ROLLOUTS OF GEMINI 3.7 FLASH

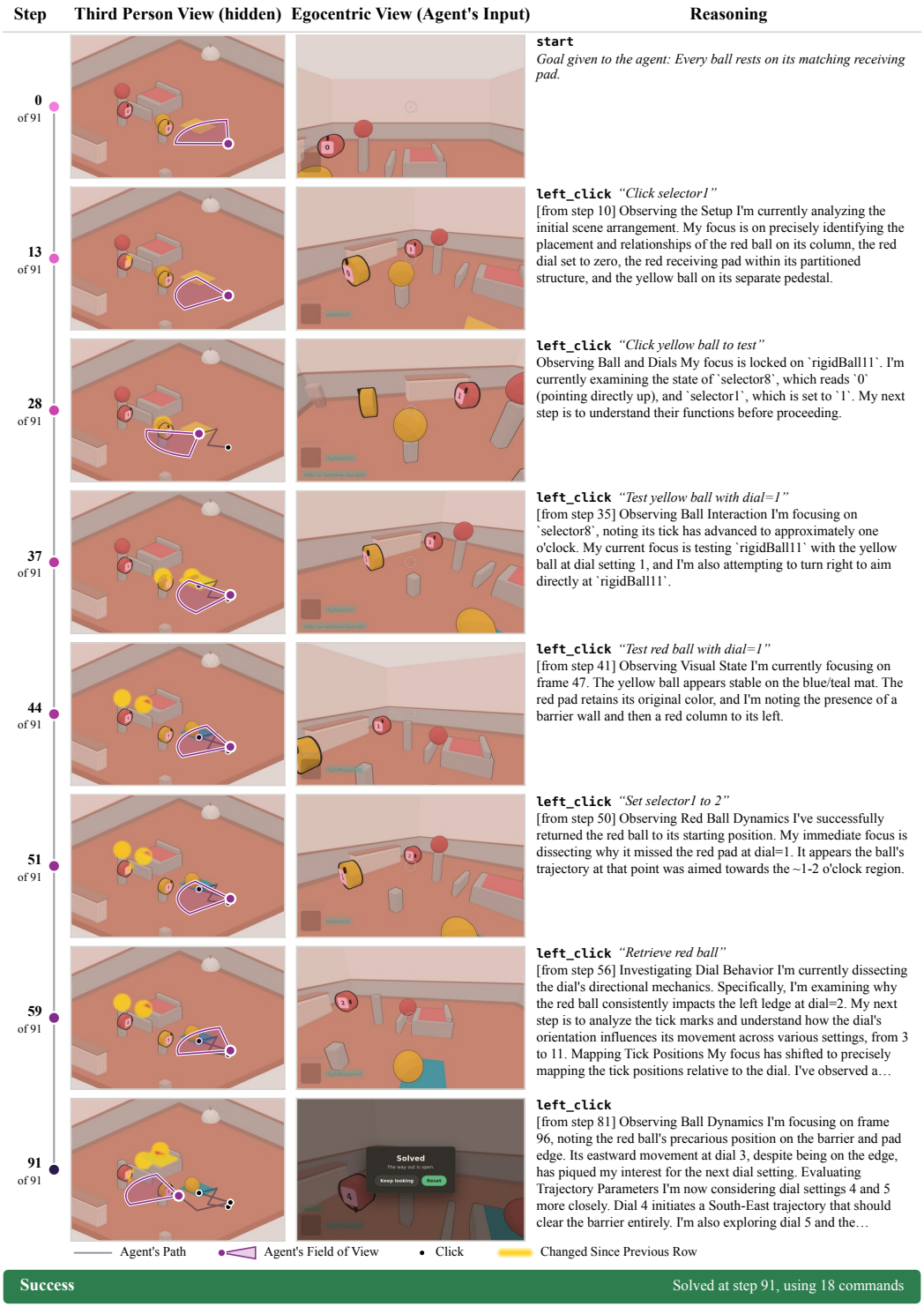

Figure 17: **Gemini 3.7 Flash** on `ballistic-lanes` (random variant), a success. Goal: *Every ball rests on its matching receiving pad.* Layout as described in Appendix I.

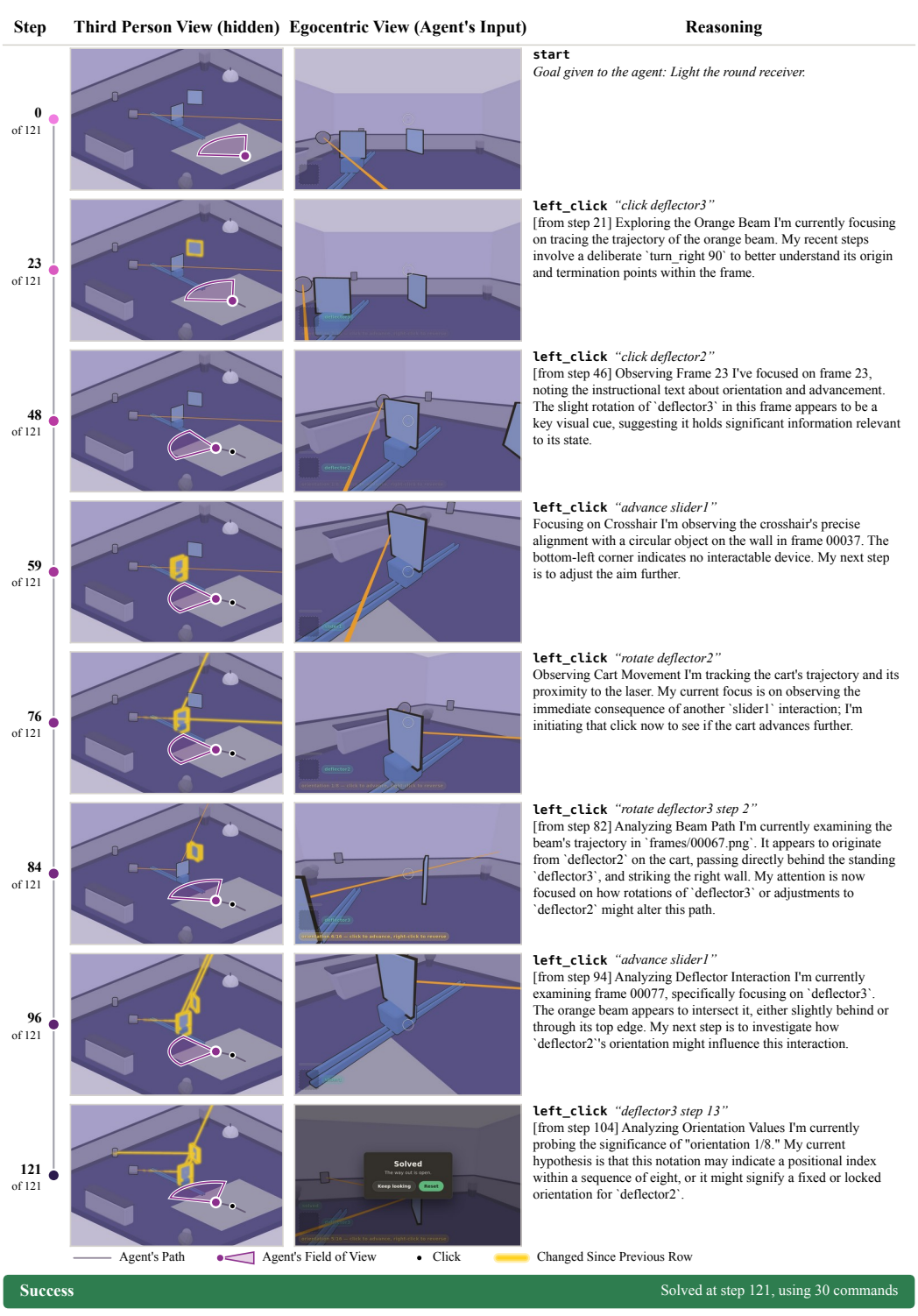

Figure 18: **Gemini 3.7 Flash** on `translating-reflector` (random variant), a success. Goal: *Light the round receiver.* Layout as described in Appendix I.

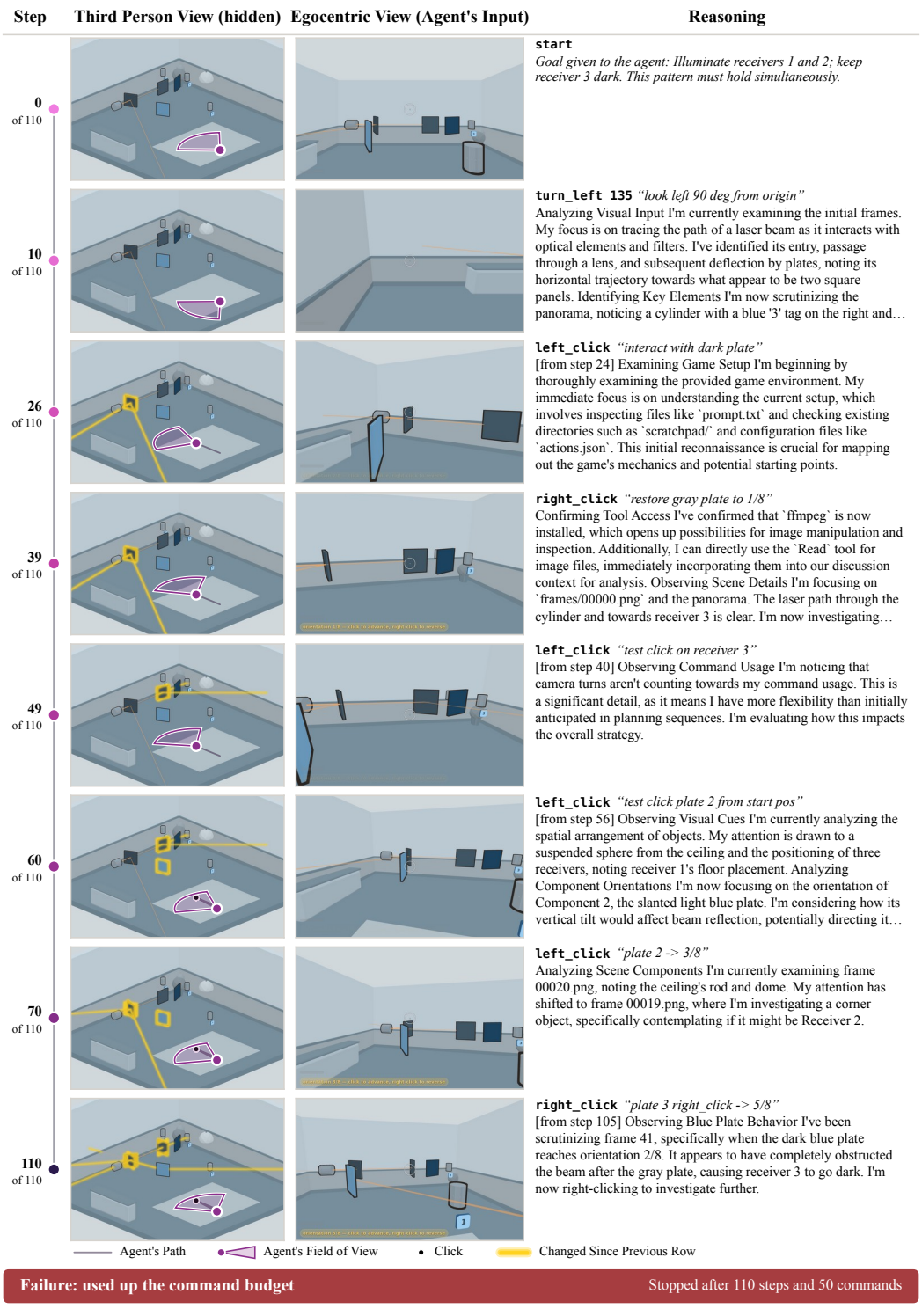

Figure 19: **Gemini 3.7 Flash** on `selective-illumination` (simple variant), a failure (command budget used up). Goal: *Illuminate receivers 1 and 2; keep receiver 3 dark. This pattern must hold simultaneously.* Layout as described in Appendix I.

I.4 ROLLOUTS OF KIMI K3

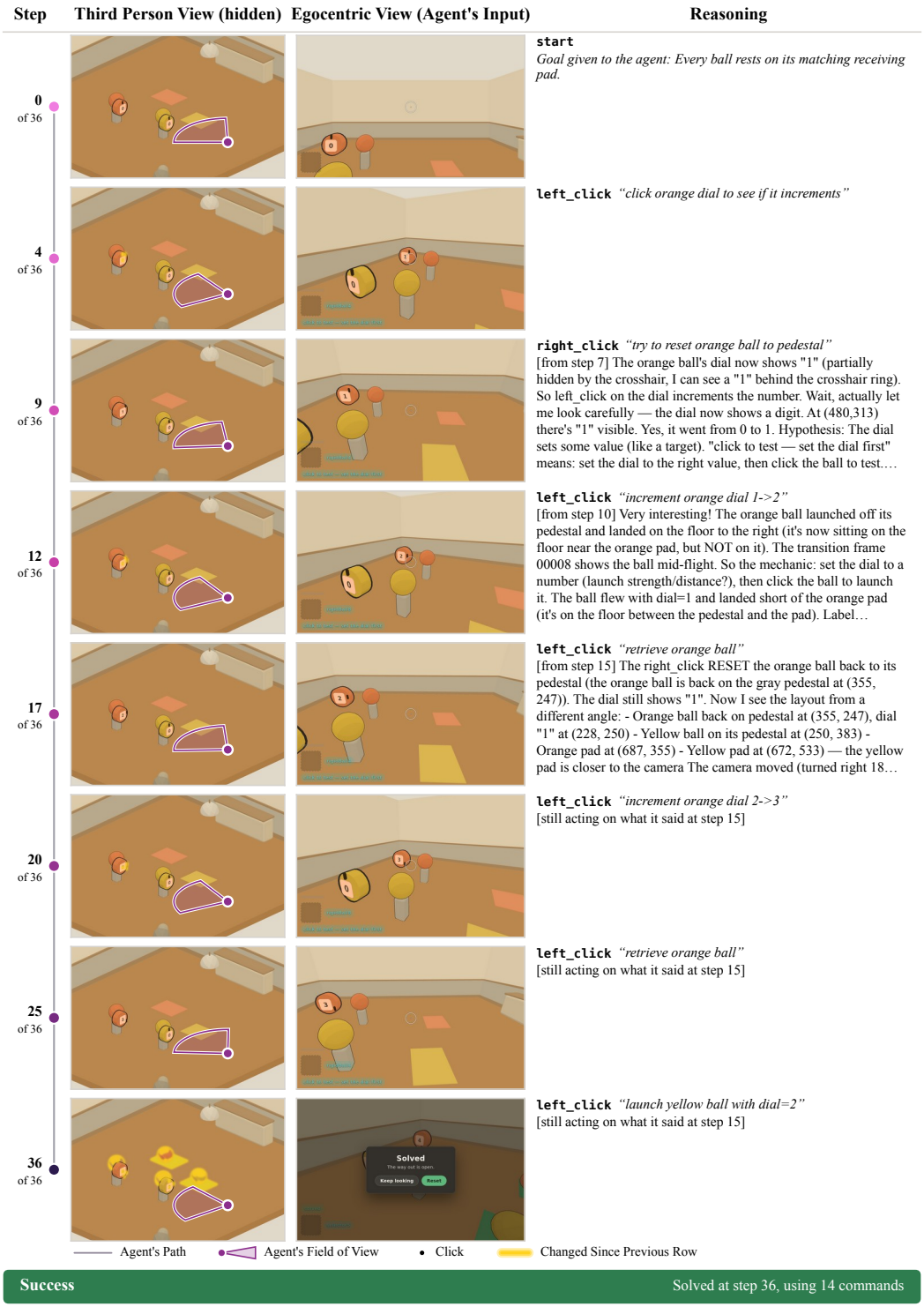

Figure 20: **Kimi K3** on `ballistic-lanes` (simple variant), a success. Goal: *Every ball rests on its matching receiving pad.* Layout as described in Appendix I.

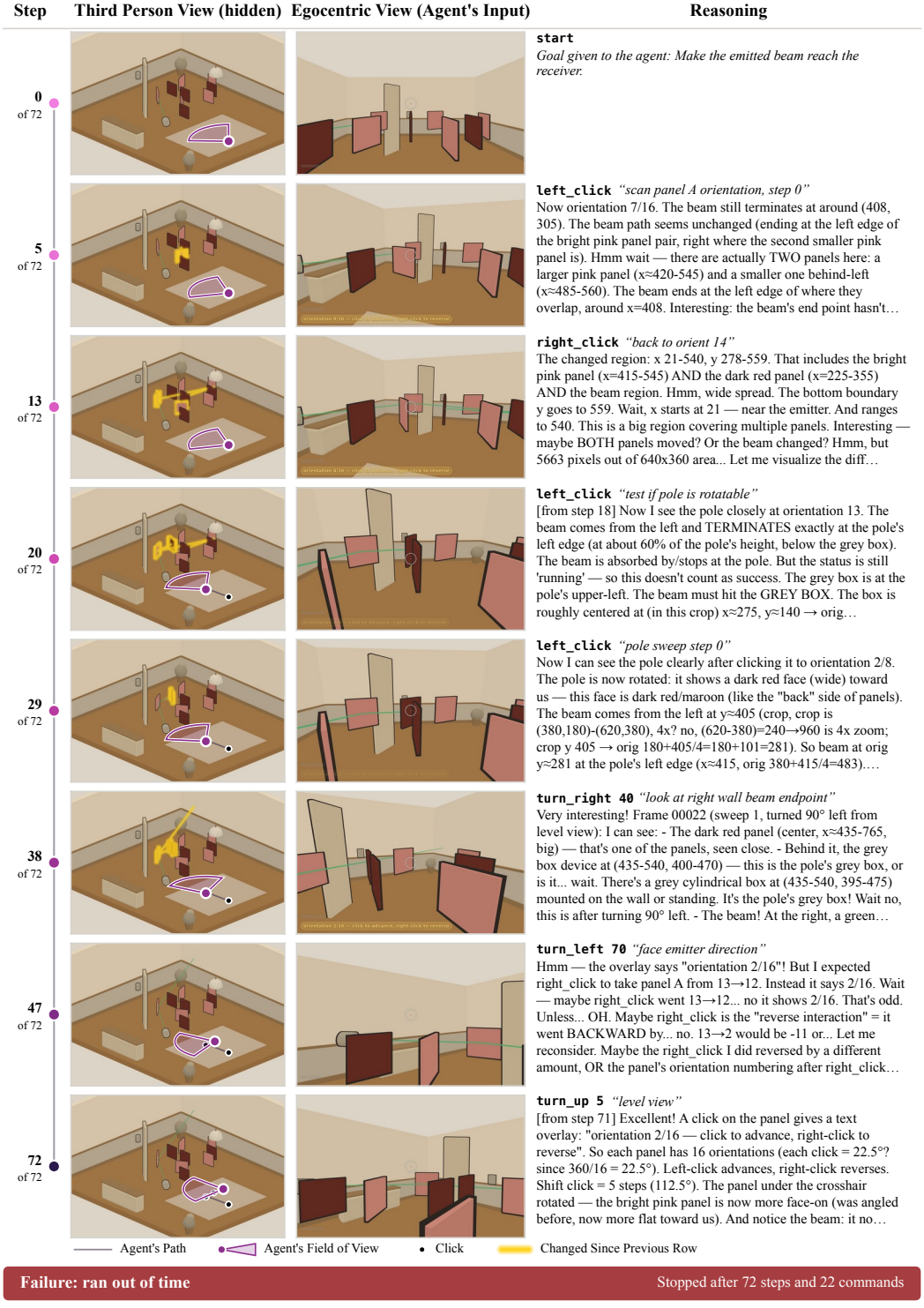

Figure 21: **Kimi K3** on `beam-routing` (hard variant), a failure (out of time). Goal: *Make the emitted beam reach the receiver.* Layout as described in Appendix I.

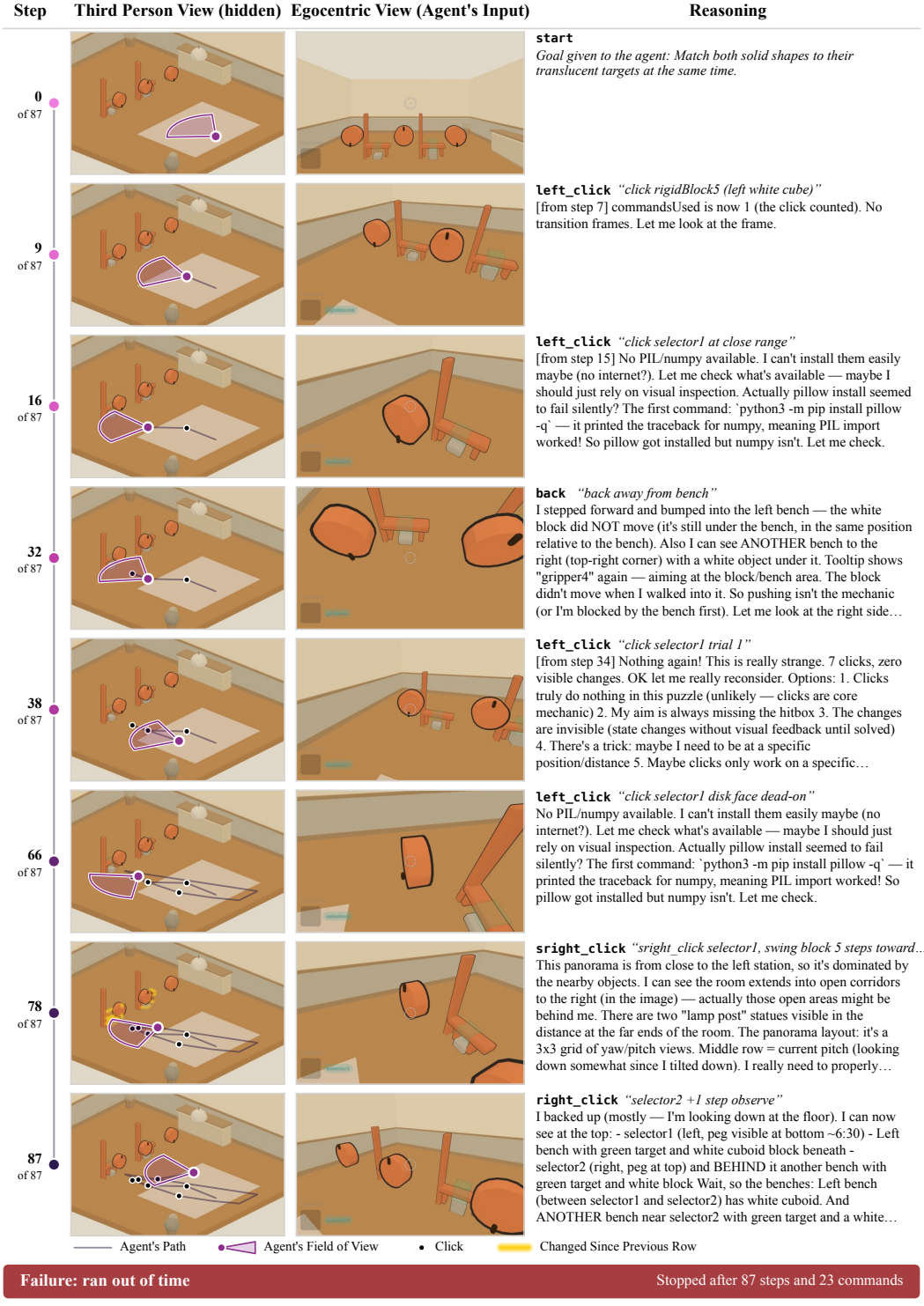

Figure 22: **Kimi K3** on `grip-and-lift` (simple variant), a failure (out of time). Goal: *Match both solid shapes to their translucent targets at the same time.* Layout as described in Appendix I.

I.5 ROLLOUTS OF CLAUDE SONNET 5

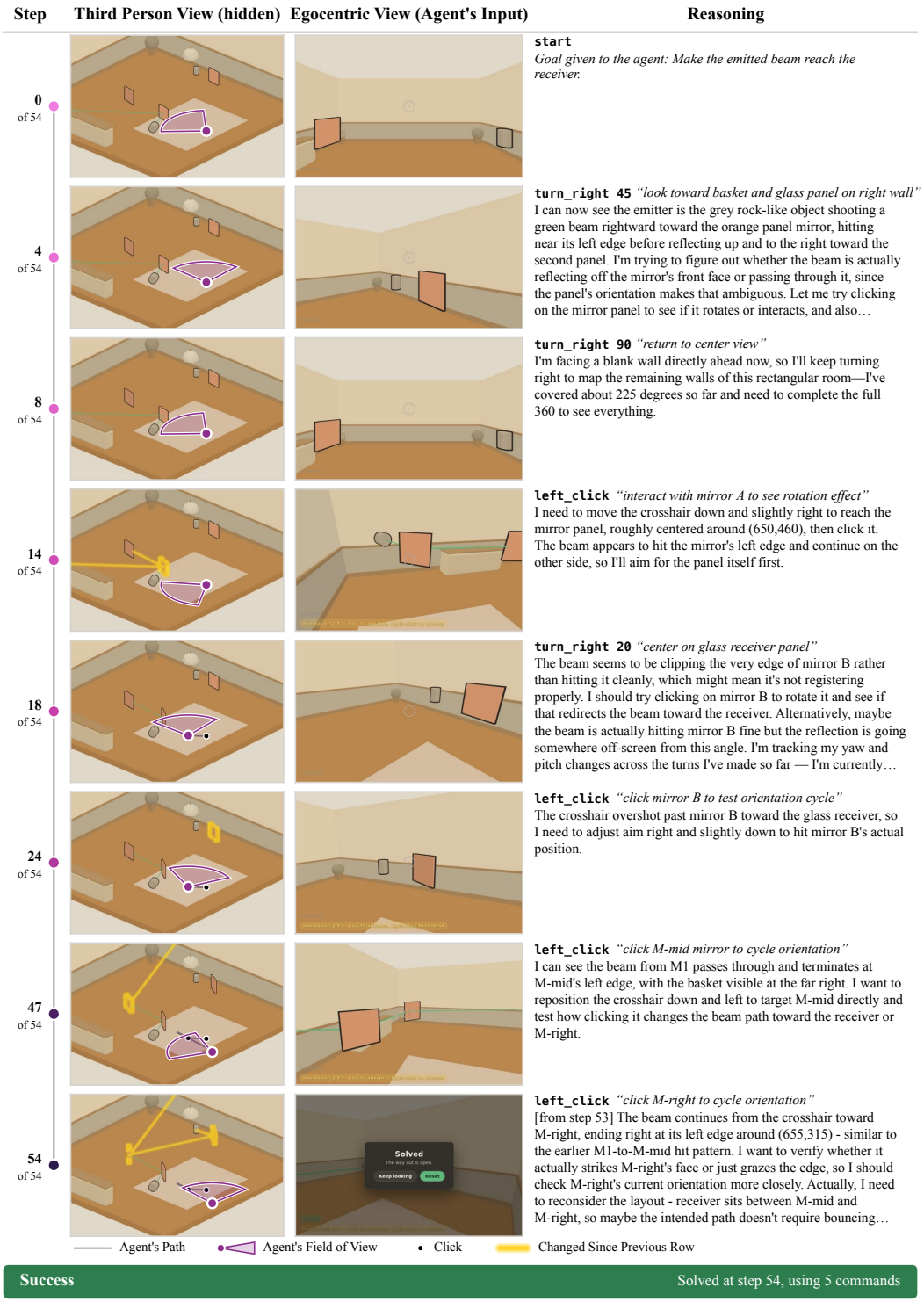

Figure 23: **Claude Sonnet 5** on `beam-routing` (simple variant), a success. Goal: *Make the emitted beam reach the receiver.* Layout as described in Appendix I.

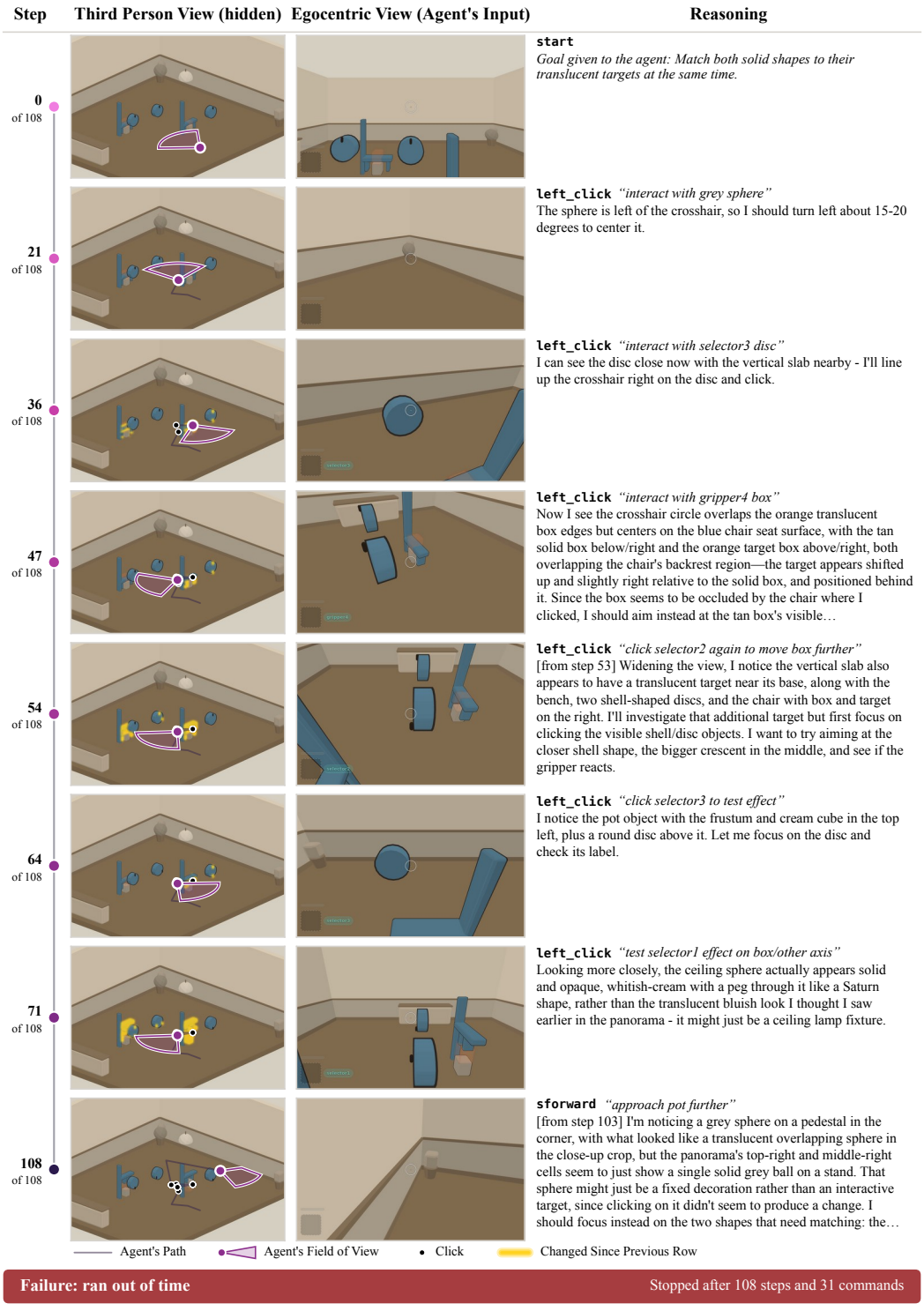

Figure 24: **Claude Sonnet 5** on `grip-and-lift` (hard variant), a failure (out of time). Goal: *Match both solid shapes to their translucent targets at the same time.* Layout as described in Appendix I.

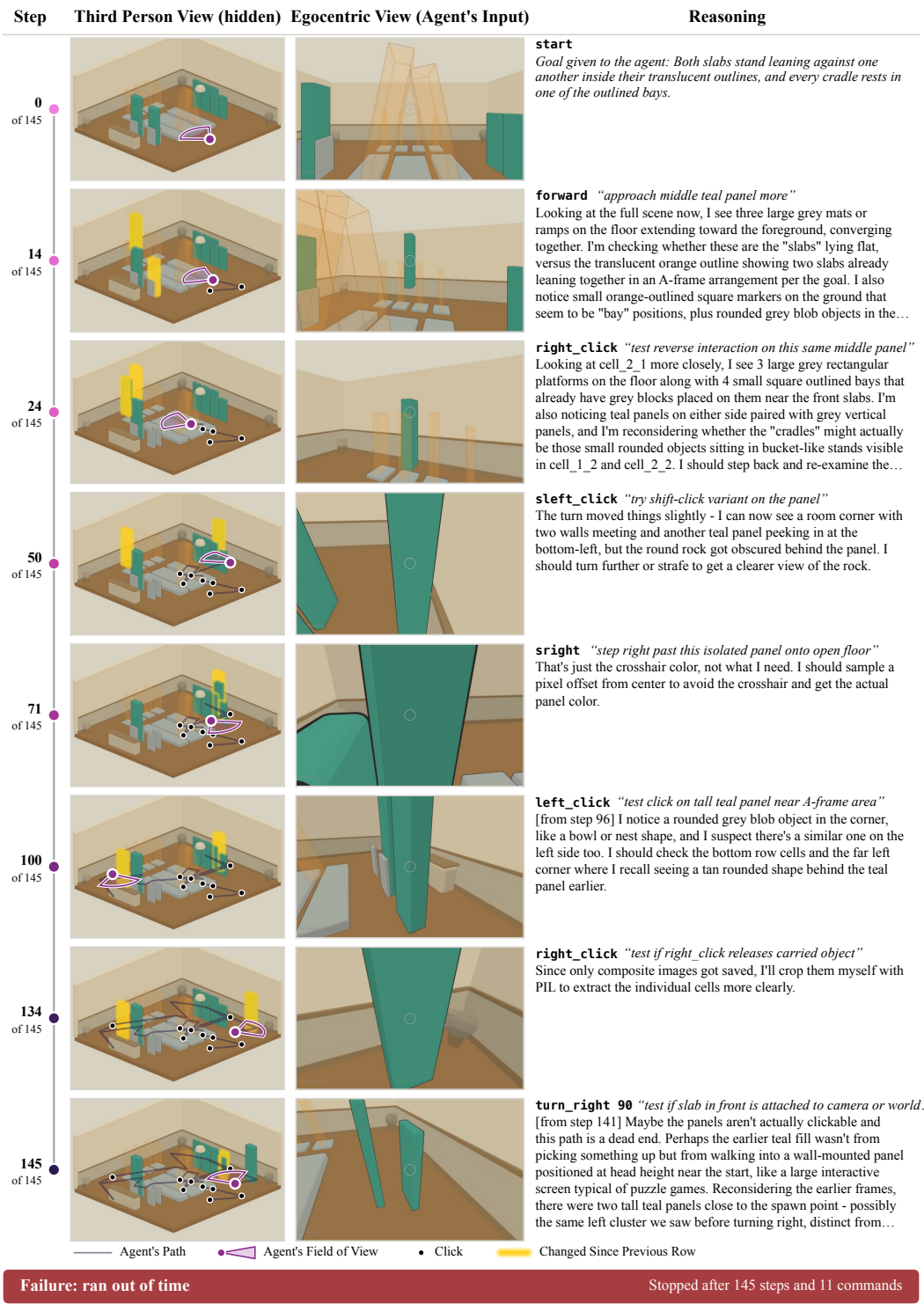

Figure 25: **Claude Sonnet 5** on `leaning-pair-revival` (random variant), a failure (out of time). Goal: *Both slabs stand leaning against one another inside their translucent outlines, and every cradle rests in one of the outlined bays.* Layout as described in Appendix I.

I.6 ROLLOUTS OF QWEN 3.8 27B

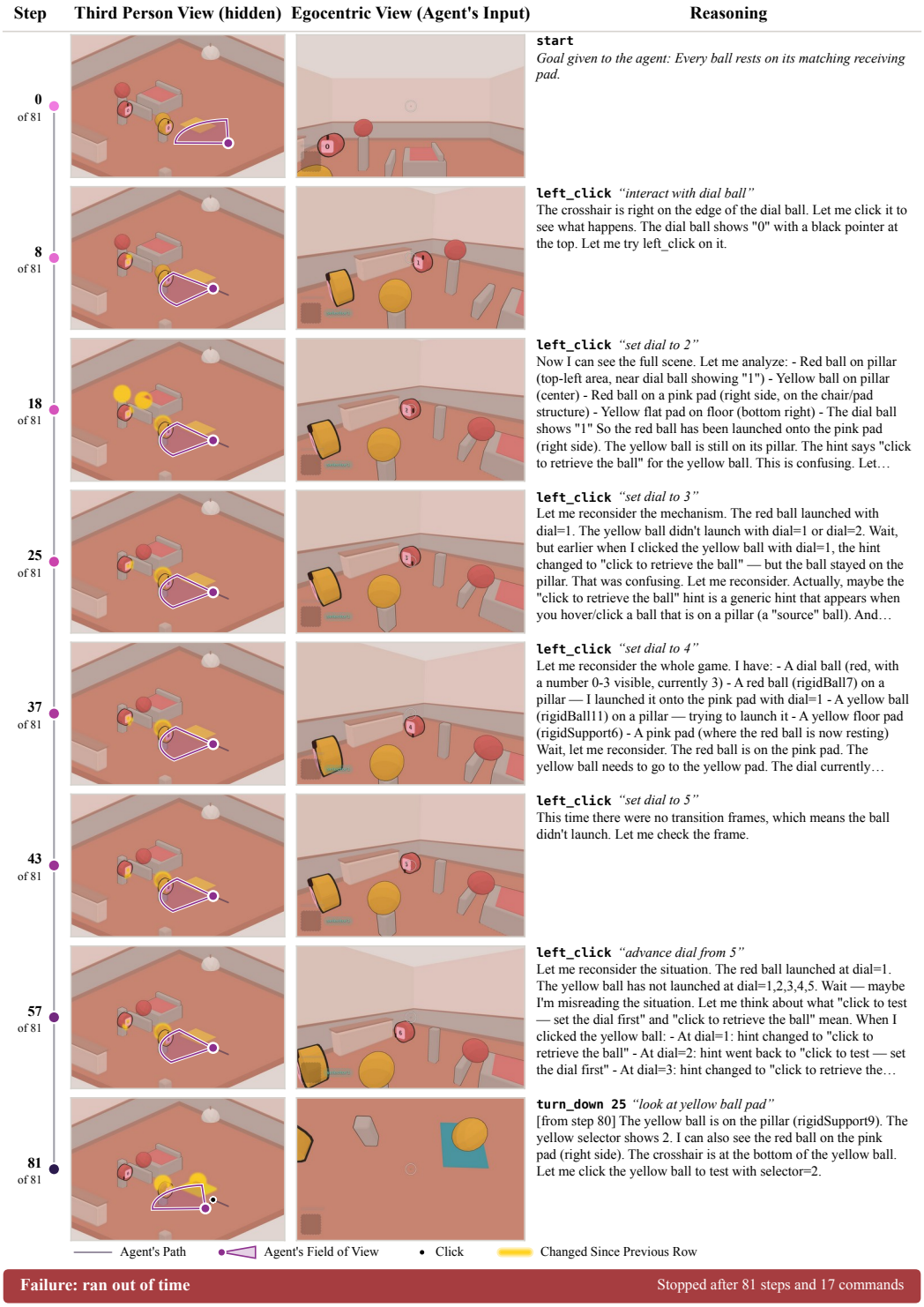

Figure 26: **Qwen 3.8 27B** on `ballistic-lanes` (random variant), a failure (out of time). Goal: *Every ball rests on its matching receiving pad.* Layout as described in Appendix I.

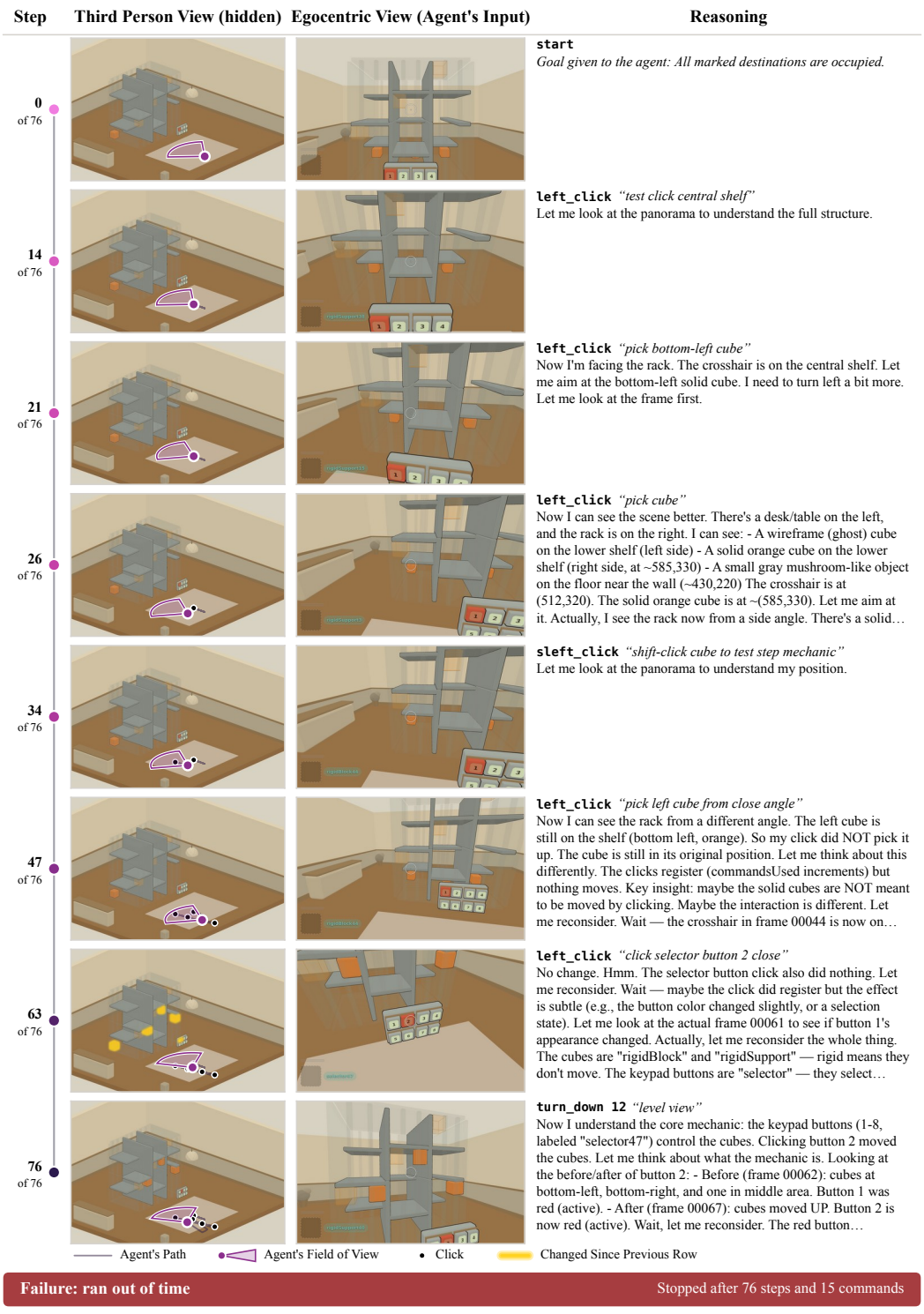

Figure 27: **Qwen 3.8 27B** on `gravity-frame` (hard variant), a failure (out of time). Goal: *All marked destinations are occupied.* Layout as described in Appendix I.

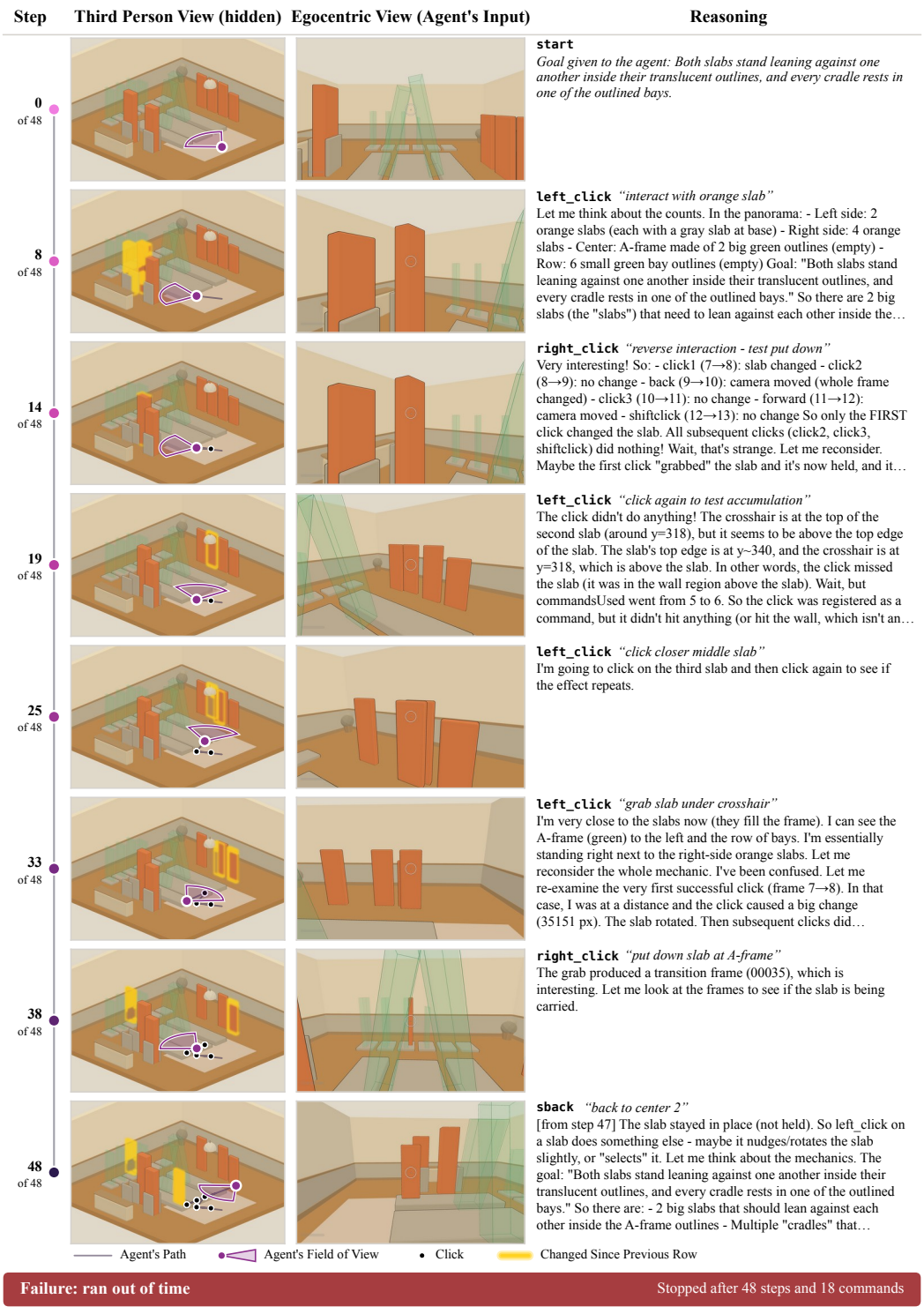

Figure 28: **Qwen 3.8 27B** on `leaning-pair-revival` (hard variant), a failure (out of time). Goal: *Both slabs stand leaning against one another inside their translucent outlines, and every cradle rests in one of the outlined bays.* Layout as described in Appendix I.