DrawingBench: Evaluating Spatial Reasoning and UI Interaction Capabilities of Large Language Models through Mouse-Based Drawing Tasks

Hyunjun Kim,[1] Sooyoung Ryu[2]

[1]KAIST, Daejeon, South Korea
[2]Seoul National University, Seoul, South Korea
hyunjun1121@kaist.ac.kr, ryuswimming@snu.ac.kr

Abstract

As agentic AI systems increasingly operate autonomously, establishing trust through verifiable evaluation becomes critical. Yet existing benchmarks lack the transparency and auditability needed to assess whether agents behave reliably. We present **DrawingBench**, a verification framework for evaluating the trustworthiness of agentic LLMs through spatial reasoning tasks that require generating sequences of low-level GUI actions. Unlike opaque evaluations, DrawingBench provides transparent, rule-based assessment: 8 objective criteria enable reproducible scoring, while action-level inspection allows stakeholders to audit agent behavior. Our framework comprises 250 diverse prompts across 20 categories and 4 difficulty levels, deterministic evaluation metrics, and an external oversight mechanism through multi-turn feedback that enables human control over agent refinement. Evaluating four state-of-the-art LLMs (Claude-4 Sonnet, GPT-4.1, GPT-4.1-mini, Gemini-2.5 Flash) across 1,000 tests, we establish both capabilities and limitations: models achieved 92.8% perfect performance with structured external feedback driving significant improvements (average +3.2%, up to +32.8% for complex scenes), but systematic error patterns emerged in tool state management and long-horizon planning. Notably, specification clarity proved more important than task complexity—models achieved 100% perfect performance when given explicit, verifiable criteria. These findings demonstrate that transparent evaluation frameworks can establish trust in agentic systems, with external oversight proving more reliable than self-correction for guiding agent behavior. Our open-source framework provides a template for trustworthy agent assessment. Code and data: https://github.com/hyunjun1121/DrawingBench

Introduction

Spatial reasoning is a fundamental aspect of intelligence, enabling both humans and AI agents to understand and interact with their environments. In the context of large language models (LLMs), spatial reasoning capability is increasingly important as we move towards *embodied* or *agentic* AI systems that can manipulate the physical or digital world through language instructions. For instance, an AI assistant might be asked to arrange objects in a room or navigate a user interface (UI)—tasks that require understanding of spatial relations (above, below, inside, etc.) and geometry. Despite impressive advancements in LLMs' general reasoning, their ability to perform precise spatial reasoning remains underexplored and poorly evaluated. Traditional evaluation benchmarks tend to focus on textual question-answering or reasoning puzzles, which may not capture an agent's performance in interactive spatial tasks (Liang et al. 2022; Srivastava et al. 2022). For instance, a recent comprehensive evaluation of frontier models across many tasks, dubbed *Humanity's Last Exam*, found significant gaps in LLM performance and calibration (Phan et al. 2025), highlighting the need for targeted benchmarks that probe specific skills like spatial reasoning.

Limitations of Existing Evaluations. Most LLM evaluation benchmarks suffer from three limitations for spatial and interactive tasks. **First**, they focus on text-based QA or classification, not action sequences. CLEVR (Johnson et al. 2017), RAVEN (Zhang et al. 2019), and SpartQA (Mirzaee et al. 2021) test spatial reasoning via static questions, not action execution. **Second**, benchmarks lack fine-grained UI interaction. WebArena (Zhou et al. 2023) and MiniWoB++ (Liu et al. 2018) use high-level actions (clicking links), not precise coordinate-level control. **Third**, multi-turn learning capability is largely untested. Most benchmarks use single-turn tasks, failing to probe iterative improvement from feedback.

Our Solution. We introduce **DrawingBench**, a benchmark evaluating LLMs on spatial reasoning through drawing tasks on a canvas UI. Each task is specified by a natural language prompt (e.g., *"Draw a red triangle inside a blue square"*), and the LLM outputs a sequence of low-level GUI actions (mouse movements, clicks, tool selections). This approach differs from prior benchmarks: the model *constructs* spatial configurations via actions rather than answering questions. Our benchmark tests spatial understanding, tool manipulation, and sequential planning simultaneously. Crucially, evaluation is text-only—the LLM relies on internal reasoning to plan drawings without seeing images.

Trust through Verifiable Evaluation. As agentic AI systems increasingly operate autonomously, establishing trust becomes paramount. DrawingBench embodies three essential properties for trustworthy evaluation: (1) **Trans-**

Copyright © 2026, Trustworthy Agentic AI Workshop@ Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

parency—eight objective criteria enable stakeholders to understand performance measurement; (2) **Auditability**—deterministic scoring allows reproduction through action sequence inspection; (3) **External oversight**—multi-turn feedback provides human control over agent refinement. Unlike opaque metrics or self-assessment, our rule-based system eliminates subjective judgment and enables action-level verification, aligning with best practices for deploying trustworthy agents.

DrawingBench comprises **250** diverse tasks spanning **20** categories and four difficulty levels. Our automated evaluation system executes action sequences and measures performance against eight objective criteria. The framework supports *multi-turn* interaction: models receive structured feedback and attempt corrections.

Contributions. We contribute: (1) **DrawingBench**—first benchmark integrating spatial reasoning and GUI action execution with 250 prompts; (2) **Automated evaluation**—eight quantitative criteria, four error types, reproducible scoring; (3) **Multi-turn protocol**—tests self-correction from explicit feedback (Madaan et al. 2023; Shinn et al. 2023); (4) **Empirical study**—1000 trials with four state-of-the-art LLMs analyzing performance across difficulty levels; (5) **Open-source release**—all code and data.[1]

Key Findings. Models achieved **92.8%** perfect scores, with structured feedback yielding **+3.2%** average improvement (**+32.8%** for complex scenes). Three insights emerged: (1) Text-based spatial reasoning is highly effective—models executed complex 15+ action sequences without visual perception. (2) *Difficulty paradox*: "Hard" tasks achieved **100%** perfect performance versus "Medium" (92.8%)—specification clarity matters more than complexity. (3) Structured feedback proves effective, but "Very Hard" tasks requiring long-horizon planning remained challenging (60% success).

Section 2 reviews related work. Section 3 details methodology. Section 4 presents results. Section 5 discusses implications and limitations. Section 6 concludes.

Related Work

Spatial Reasoning Benchmarks

Evaluating spatial reasoning has been a focus across vision and language domains. In vision+language, **CLEVR** (Johnson et al. 2017) presents synthetic 3D scenes with compositional reasoning questions, while **RAVEN** (Zhang et al. 2019) tests abstract visual pattern completion. In pure text, **SpartQA** (Mirzaee et al. 2021) assesses understanding of spatial relations through paragraph descriptions and questions.

Table 1 summarizes some representative spatial reasoning benchmarks. Most are **static** in nature: the model reads or observes a scenario and produces an answer (classification or text). They typically measure reasoning capability via accuracy on predefined questions. However, none of these require the model to *generate a sequence of actions* to physically realize a spatial configuration. In contrast, DrawingBench is **dynamic**: the model must "draw" the answer, not just tell it. This calls for integrating spatial reasoning with a decision-making process to carry out the drawing actions.

[1]https://github.com/hyunjun1121/DrawingBench

Benchmark	Modality	Type	Output
CLEVR	Vision	Static	QA
SpartQA	Text	Static	QA
RAVEN	Vision	Static	Choice
DrawingBench	**Text**	**Dynamic**	**Actions**

Table 1: Comparison of spatial reasoning benchmarks. DrawingBench uniquely requires dynamic action generation rather than static question answering, integrating spatial understanding with executable UI interactions.

Program Synthesis and Visual Generation

Our work sits at the intersection of program synthesis and visual generation. Code synthesis benchmarks like **HumanEval** (Chen et al. 2021), **MBPP** (Austin et al. 2021), and **APPS** (Hendrycks et al. 2021) evaluate algorithmic correctness but not spatial outputs. Visual generation work like **Sketch-RNN** (Ha and Eck 2018) and visual program synthesis (Ellis et al. 2018) demonstrates that graphics can be represented as executable sequences, but these train on data rather than evaluate zero-shot spatial reasoning. DrawingBench bridges these areas by requiring action sequences that are both syntactically valid and geometrically correct.

User Interface Interaction Benchmarks

UI interaction benchmarks evaluate agents on GUI and web tasks. **MiniWoB++** (Liu et al. 2018) tests basic web form interactions (clicking buttons, typing text), while **WebArena** (Zhou et al. 2023), **Mind2Web** (Deng et al. 2023), and **VisualWebArena** (Koh et al. 2024) evaluate more complex web navigation and multimodal understanding. **MacroBench** (Kim and Kim 2025) assesses LLMs on writing web automation scripts.

DrawingBench differs by requiring finer control granularity: coordinate-level manipulation on a blank canvas rather than high-level actions on predefined elements. We evaluate spatial output quality (geometric correctness) rather than task completion, measuring whether models can create content rather than merely interact with existing UI elements.

Multi-turn Feedback and Iterative Refinement

Our two-turn protocol relates to work on improving outputs via feedback loops. **Self-Refine** (Madaan et al. 2023) and **Reflexion** (Shinn et al. 2023) enable LLMs to critique and refine their outputs through self-feedback, improving performance on code generation and reasoning tasks.

Our approach differs in the feedback source: we use automated rule-based evaluation rather than self-critique. This provides objective, deterministic error signals (e.g., "the circle is not fully inside the square") that enable reliable measurement of improvement. While other work explores model-generated feedback (Bai et al. 2022), our focus on task performance benefits from external evaluation that eliminates self-assessment biases.

Positioning in LLM Evaluation

Comprehensive frameworks like **HELM** (Liang et al. 2022), **BIG-bench** (Srivastava et al. 2022), and **AgentBench** (Liu et al. 2024) evaluate LLMs across diverse capabilities. DrawingBench complements these by simultaneously testing spatial reasoning, fine-grained UI interaction, and multi-turn learning—three dimensions that existing benchmarks treat in isolation but that real-world agents must coordinate. By requiring models to *construct* rather than *describe* spatial configurations, we provide a stringent test of spatial understanding for practical agent deployment.

Methodology

Benchmark Design

Dataset Composition. DrawingBench consists of 250 diverse drawing prompts classified into 4 difficulty levels (Easy, Medium, Hard, Very Hard) and 20 categories. The dataset distribution is shown in Table 2.

Table 2: Dataset Composition by Difficulty

Difficulty	Count	%	Perfect Rate (T2)
Easy	112	44.8%	97.3%
Medium	97	38.8%	92.8%
Hard	21	8.4%	100.0%
Very Hard	20	8.0%	60.0%
Total	**250**	**100%**	**92.8%**

Prompt Design Principles. Each prompt follows these principles: (1) *Clarity*: unambiguous instructions, (2) *Diversity*: various task types, (3) *Realism*: reflecting actual use scenarios, (4) *Measurability*: enabling automated evaluation.

Example prompts:

- Easy: "Draw a red circle in the center of the canvas"
- Medium: "Draw a house with a triangular roof and rectangular body"
- Hard: "Draw 4 squares in each corner of the canvas"
- Very Hard: "Draw a checkerboard pattern with $8 imes 8$ squares"

Drawing Application

UI Configuration. The application provides a $1000 imes 700$ pixel canvas with 6 drawing tools (pen, eraser, fill, line, rectangle, circle), 8 colors (black, red, green, blue, yellow, magenta, cyan, white), and 3 sizes (2px, 5px, 10px). The canvas starts at screen coordinates (90, 70). See Appendix for detailed UI layout.

Mouse Actions. Four action types are supported:

`moveTo`(x, y): move cursor to position
`click`(): click at current position
`mouseDown`(): start dragging
`mouseUp`(): end dragging

See Appendix for complete action sequence examples.

Evaluation System

Evaluation Criteria (8 Types). The system evaluates drawings using 8 criteria:

1. **required_tools**: whether specific tools were used (e.g., rectangle tool)
2. **required_colors**: whether specific colors were used (matching hex codes)
3. **min_segments**: minimum drawing segments (mouseDown/mouseUp pairs)
4. **min_coverage**: minimum canvas coverage (bounding box / canvas area)
5. **max_actions**: maximum action count (efficiency test)
6. **position_constraint**: spatial positioning requirements
7. **size_constraint**: exact size requirements
8. **corner_placement**: corner positioning requirements

Error Types (4 Categories). The system detects 4 error types:

- **SYNTAX_ERROR** (severity: critical): invalid JSON, missing required fields (penalty: -0.3)
- **COORDINATE_ERROR** (severity: high): out-of-bounds coordinates (penalty: -0.2)
- **LOGIC_ERROR** (severity: medium): unmatched mouseDown/mouseUp (penalty: -0.1)
- **EFFICIENCY_WARNING** (severity: low): excessive actions (penalty: -0.05)

Score Calculation. The final score is calculated as:

$$\text{Score} = \frac{\text{Criteria Met}}{\text{Total Criteria}} - \sum \text{Error Penalties} + \text{Bonus} \quad (1)$$

where Bonus is based on coverage and efficiency.

Multi-turn Feedback System

Feedback Generation. After evaluation, the system generates structured feedback including: (1) current score and grade, (2) specific error descriptions, (3) actionable improvement advice, and (4) missing criteria. See Appendix for examples.

Two-Turn Process. We limit interaction to two turns based on: (1) practical agent deployment typically involves quick iteration cycles, (2) pilot tests showed diminishing returns after Turn 2 (¡1% improvement), and (3) reducing evaluation costs while still capturing feedback-driven improvement patterns.

- **Turn 1**: LLM initial attempt $\rightarrow$ evaluation $\rightarrow$ feedback generation
- **Turn 2**: LLM improvement based on feedback $\rightarrow$ re-evaluation
- **Early Stopping**: Skip Turn 2 if Turn 1 score ≥ 0.9 (71.2% of tasks, reducing redundant computation)

Experimental Setup

Test Models. We evaluated four state-of-the-art LLMs:

- Anthropic Claude-4 Sonnet (thinking-off)
- OpenAI GPT-4.1
- OpenAI GPT-4.1-mini
- Google Gemini-2.5 Flash (thinking-off)

Experimental Protocol.

- Total tests: 1,000 (250 prompts $imes$ 4 models)
- Temperature: 0.7 (balance between consistency and diversity)
- Max tokens: 4,000
- Timeout: 30 seconds per test
- Retry on error: 3 attempts
- Sequential execution: models tested one by one

Execution Statistics. The experiment ran for 5.6 hours total. Model-specific execution times:

- Claude-4 Sonnet: 84.2 minutes
- GPT-4.1: 86.3 minutes
- GPT-4.1-mini: 77.1 minutes (fastest)
- Gemini-2.5 Flash: 90.6 minutes

Data Collection. For each test, we collected:

- Generated action sequence
- Evaluation scores (Turn 1 and Turn 2)
- Error information
- Feedback content
- Token usage
- Execution time

This comprehensive data collection enables detailed analysis of model performance patterns, error distributions, and the effectiveness of multi-turn feedback, which we present in the following section.

Results

We now present our empirical findings from evaluating four state-of-the-art LLMs on DrawingBench. Our analysis examines performance across multiple dimensions: overall scores, difficulty levels, task categories, multi-turn improvement, error patterns, and action efficiency. These results reveal both impressive spatial reasoning capabilities and systematic patterns that inform future development of LLM-based agents.

Overall Performance

Table 3 shows the aggregate performance across all 1,000 tests (4 models $\times$ 250 tasks). Models achieved an average score of 0.925 in Turn 1, which improved to 0.954 in Turn 2, representing a 3.2% relative improvement. Individual model performance ranged from 0.919 (GPT-4.1-mini) to 0.931 (Claude-4 Sonnet) on Turn 1, with all models showing consistent improvement in Turn 2.

Key Observations:

- High perfect score rate of 92.8% (score ≥ 0.9)

Table 3: Overall Performance Across All Tests. The table shows aggregate metrics from 250 drawing tasks evaluated across four state-of-the-art LLMs. Turn 2 performance demonstrates significant improvement with structured feedback, achieving 92.8% perfect score rate and 33% reduction in variance.

Metric	Turn 1	Turn 2	Change
Average Score	0.925	0.954	+0.030
Std Deviation	0.099	0.066	-0.033
Perfect Scores	192/250	232/250	+40
Perfect Rate	76.8%	92.8%	+16.0%
Median Score	0.970	0.985	+0.015

- 33% reduction in standard deviation (0.099 $\rightarrow$ 0.066) indicates more consistent performance
- Achieved 40 additional perfect scores
- Cross-model consistency: all four models achieved $>$ 90% perfect rate (Claude-4: 94.4%, GPT-4.1: 93.2%, Gemini-2.5: 92.0%, GPT-4.1-mini: 90.8%)

Model-Specific Patterns. While aggregate performance is similar, models exhibit distinct behaviors. Claude-4 Sonnet achieved the highest Turn 2 perfect rate (94.4%) with minimal variance, suggesting robust spatial reasoning. GPT-4.1-mini, despite being the smallest model, achieved 90.8% perfect rate—only 3.6% below the best model—indicating that spatial reasoning capabilities scale well even to smaller models. Detailed per-model breakdowns are provided in Appendix .

Performance by Difficulty

Table 4 shows performance by difficulty level. Notably, Hard tasks achieved higher performance than Medium tasks.

Table 4: Performance by Difficulty Level. Counter-intuitively, Hard tasks achieved the highest Turn 2 performance (0.973), revealing that specification clarity matters more than inherent complexity. Very Hard tasks requiring extensive procedural accuracy showed the largest improvement gains (+0.052).

Difficulty	Tests	T1 Score	T2 Score	Improve
Easy	112	0.944	0.963	+0.019
Medium	97	0.925	0.958	+0.033
Hard	21	0.923	**0.973**	+0.050
Very Hard	20	0.816	0.869	+0.052

The Difficulty Paradox. Hard difficulty tasks (0.973) showed better performance than Medium tasks (0.958). Analysis reveals this paradox stems from task specification clarity rather than inherent complexity. Hard tasks average 4.2 explicit constraints (e.g., "4 squares in each corner") versus Medium tasks' 2.8 constraints (e.g., "draw a house"). The deterministic nature of Hard task requirements (100% have position constraints vs. 62% for Medium) enables more reliable execution despite greater spatial complexity.

Improvement Trends. As difficulty increases, improvement magnitude also increases:

- Easy: +0.019 (1.9%)
- Medium: +0.033 (3.3%)
- Hard: +0.050 (5.0%)
- Very Hard: +0.052 (5.2%, largest improvement)

Very Hard Challenge. Very Hard tasks remain challenging with a 60% perfect rate. These involve extremely high precision requirements and complex pattern generation.

Performance by Category

Table 5 shows the performance of the top 10 categories.

Table 5: Top 10 Categories by Turn 2 Performance. The scenes category showed the most dramatic improvement (+32.8%), demonstrating that complex compositional tasks benefit most from structured feedback. Six categories achieved perfect 1.000 performance.

Category	Tests	T1	T2	Improve
scenes	1	0.672	**1.000**	**+0.328**
efficiency_test	2	1.000	1.000	0.000
creative	2	0.924	1.000	+0.076
precision_test	1	1.000	1.000	0.000
tool_switching	1	1.000	1.000	0.000
spatial	1	1.000	1.000	0.000
compositional	11	0.957	0.998	+0.041
angle	2	0.881	0.980	+0.100
complex	2	0.900	0.975	+0.075
spatial_reasoning	71	0.959	0.973	+0.015

Remarkable Improvements:

- **Scenes** (+32.8%): dramatic improvement from 0.672 to perfect 1.000
- **Angle** (+10.0%): significant improvement in angle-based tasks
- **Creative** (+7.6%): effective improvement in creative compositions

Perfect Categories. Six categories achieved perfect performance (1.000): efficiency_test, precision_test, tool_switching, spatial, and Turn 2's scenes and creative.

Large-Scale Categories. The spatial_reasoning category (71 tests, largest) maintained consistently high performance (0.973).

Multi-turn Improvement Analysis

Table 6 shows the improvement distribution.

Key Findings:

- 71.2% were already excellent in Turn 1 (early stopping)
- 28.8% achieved actual improvement through feedback
- 4.8% achieved substantial improvement (¿0.10)

Top Improvement Cases:

1. Scenes category: 0.672 → 1.000 (+0.328)
2. Size-constrained task: 0.700 → 0.950 (+0.250)
3. Angle-based task: 0.750 → 0.980 (+0.230)

Table 6: Multi-turn Improvement Distribution. While 71.2% of tasks achieved perfect scores on Turn 1 (early stopping), 28.8% showed measurable improvement through feedback, with 4.8% achieving substantial gains exceeding 0.10 points.

Improvement Range	Count	Percentage
No improvement (0.00)	178	71.2%
Small (0.01-0.05)	38	15.2%
Medium (0.06-0.10)	22	8.8%
Large (¿0.10)	12	4.8%

Error Analysis

Table 7 shows error frequency and reduction.

Table 7: Error Frequency and Reduction. Structured feedback achieved 49% overall error reduction, with critical errors (SYNTAX, COORDINATE) showing the largest decreases. SYNTAX_ERROR was completely eliminated in Turn 2.

Error Type	Turn 1	Turn 2	Reduction
SYNTAX_ERROR	3	0	-100%
COORDINATE_ERROR	8	2	-75%
LOGIC_ERROR	15	5	-67%
EFFICIENCY_WARNING	42	28	-33%
Total	**68**	**35**	**-49%**

Error Patterns:

- **Critical errors** (SYNTAX, COORDINATE) were significantly reduced
- **SYNTAX_ERROR** was completely eliminated (100% reduction)
- **LOGIC_ERROR** was reduced by 67%
- **EFFICIENCY_WARNING** showed relatively modest reduction (33%)

Error Distribution by Difficulty. Error frequency varies significantly by task difficulty. Very Hard tasks (20 tasks) accounted for 35% of all Turn 1 errors (24/68 errors) despite representing only 8% of the dataset, yielding an error rate of 1.2 errors per task compared to 0.18 errors per task for Easy tasks. LOGIC_ERROR occurred primarily in Very Hard tasks (11/15 instances), reflecting the complexity of maintaining state consistency across extensive action sequences. In contrast, EFFICIENCY_WARNING was uniformly distributed across all difficulty levels, indicating that action optimization is challenging regardless of task complexity.

Error Correction Patterns. The differential reduction rates reveal distinct error correction dynamics. SYNTAX_ERROR (100% reduction) and COORDINATE_ERROR (75% reduction) respond effectively to explicit feedback specifying valid JSON structure and coordinate bounds. LOGIC_ERROR reduction (67%) required models to repair state inconsistencies (e.g., unmatched mouseDown/mouseUp pairs), which feedback

addresses by identifying specific mismatched actions. EFFICIENCY_WARNING showed the lowest reduction (33%), suggesting that while models can add missing actions or fix syntax, optimizing for efficiency requires deeper planning changes that are harder to achieve through single-turn feedback.

Root Cause Analysis. Tool selection omissions (appearing in 18 tasks) predominantly occurred when task descriptions mentioned tools implicitly rather than explicitly. For instance, "draw a red circle" assumes circle tool selection, but some models generated pen-based circular strokes instead. Coverage insufficiency (12 tasks) correlated with tasks lacking explicit size specifications—models often drew geometrically correct but undersized shapes. These patterns indicate that extbfspecification ambiguity, rather than model capability limitations, drives many errors, reinforcing the difficulty paradox finding that explicit constraints improve performance.

Action Efficiency

Action Sequence Length. Average action sequence length:

- Turn 1: 15.3 actions
- Turn 2: 16.2 actions (+0.9)

The increased action count in Turn 2 reflects more thorough execution (for more elements or better coverage).

Drawing Efficiency. Ratio of drawing actions (mouseDown/mouseUp pairs):

- Turn 1: 42.5% (average 6.5 drawing segments)
- Turn 2: 45.8% (average 7.4 drawing segments)

Higher drawing ratio indicates more productive actions.

Tool Usage Patterns. Most frequently used tools:

1. Pen: 68.2% (primary drawing tool)
2. Rectangle: 18.4% (efficient shapes)
3. Circle: 12.7% (circular objects)
4. Line: 8.5% (straight segments)
5. Fill: 5.2% (area filling)
6. Eraser: 2.1% (corrections)

These quantitative results establish that current LLMs possess strong baseline spatial reasoning capabilities while revealing systematic patterns in both performance and errors. In the following section, we interpret these findings, examine their implications for LLM-based agents, and discuss unexpected observations such as the difficulty paradox.

Discussion

Strong Baseline Spatial Reasoning

Current LLMs demonstrate robust spatial reasoning in text-only settings—achieving 92.8% perfect scores with average 0.954. Models performed coordinate calculations, understood relative positioning, and executed complex 15+ action sequences, challenging assumptions that spatial reasoning requires visual perception.

Multi-turn Feedback Effectiveness

Structured external feedback proves highly effective, yielding +3.2% improvement and 33% variance reduction. Improvement was non-uniform: scenes tasks gained +32.8%, while simpler tasks showed minimal change. Our external rule-based approach differs from self-critique (Madaan et al. 2023; Shinn et al. 2023), offering more reliable measurement.

The Difficulty Paradox

Specification clarity matters more than task complexity: "Hard" tasks achieved 100% perfect performance versus "Medium" (92.8%). Hard tasks averaged 4.2 explicit constraints versus Medium's 2.8, with 100% including position criteria versus 62%. When criteria are deterministic, models reliably satisfy requirements regardless of complexity.

Error Patterns and Limitations

Systematic error patterns emerged: extbftool selection errors (15%)—models forgot to select required tools despite correct actions, suggesting imperfect UI state tracking; extbfcoordinate precision errors (10%); and extbfcoverage insufficiency (8%). The extbfVery Hard category (60% perfect) highlights current limitations in long-horizon planning—models made subtle mistakes in complex patterns (misaligning grids, miscounting iterations).

Implications for LLM-based Agents

Our findings yield key implications: (1) extbfText-based interfaces are viable—LLM agents can operate in textual/API environments without visual perception. (2) extbfStructured feedback accelerates learning—external evaluators guide agents more effectively than self-correction. (3) extbfClear specifications enable complex tasks—well-defined instructions unlock sophisticated capabilities. (4) extbfTool state tracking requires attention—explicit state mechanisms are needed.

Trust and Verifiability in Agentic Evaluation

DrawingBench demonstrates transparent evaluation for agentic trust. Our eight objective criteria provide **verifiable assessment**—stakeholders reproduce scores by inspecting action sequences against explicit rules. Each criterion yields deterministic 0-1 scores, contrasting with opaque evaluations using subjective ratings or proprietary algorithms.

Verifiable evaluation is critical for consequential domains (industrial control, medical diagnosis). Our framework provides **action-level auditability**: when an agent scores 0.85, we trace deficits to specific violations (e.g., "required tool not used"), enabling targeted diagnosis versus black-box uncertainty.

External oversight outperforms self-correction. Our multi-turn feedback delivers structured error signals from deterministic rules rather than self-critique, which can amplify biases. The +3.2% average improvement (+32.8% for complex scenes) demonstrates efficacy for safety-critical human-in-the-loop systems.

These findings align with trustworthy AI principles: evaluation must be transparent, reproducible, and externally controlled. DrawingBench shows complex agent capabilities can be assessed rigorously without sacrificing interpretability.

Limitations and Future Work

Our benchmark has limitations suggesting future research. First, we lack human performance baselines to contextualize LLM capabilities. Second, rule-based evaluation focuses on objective criteria (tool usage, coordinates, coverage) but cannot assess aesthetic quality. Third, text-only testing excludes visual feedback—incorporating vision capabilities (screenshot-based feedback) could enhance performance for iterative refinement tasks.

DrawingBench successfully probes LLM spatial reasoning and UI interaction, revealing impressive capabilities (92.8% perfect) and limitations (struggles with long-horizon planning). Insights regarding specification clarity, structured feedback, and tool state management provide actionable guidance for developing more capable agents.

Conclusion

We introduced **DrawingBench**, a verifiable evaluation framework for assessing trustworthiness of agentic LLMs through spatial reasoning tasks requiring GUI action sequences. Our framework provides objective metrics, reproducible assessment, and external oversight mechanisms enabling stakeholders to verify agent behavior.

Contributions. We contribute: (1) **Verifiable Benchmark**—250 prompts across 20 categories, 4 difficulty levels; (2) **Transparent Evaluation**—8 objective criteria, 4 error types, reproducible scoring; (3) **External Oversight**—multi-turn feedback for human control; (4) **Trust Assessment**—1,000 trials establishing capabilities and limitations; (5) **Open Framework**—code and data for community verification.

Findings. Three key insights: **(1)** 92.8% perfect rate with transparent metrics; **(2)** External oversight outperforms self-correction (+3.2% average, +32.8% for complex tasks); **(3)** Explicit specifications enable reliable behavior regardless of complexity.

Implications. Trustworthy agentic AI requires: **(1)** Transparent evaluation—verifiable criteria; **(2)** External control—human oversight; **(3)** Clear specifications—deterministic requirements. Our framework instantiates these principles, providing a template for trustworthy agent assessment. We release all resources open-source for community verification and extension.

Acknowledgments

This work was conducted using cloud computing resources. We are grateful for the support and feedback from the research community.

References

Austin, J.; Odena, A.; Nye, M. I.; Bosma, M.; Michalewski, H.; Dohan, D.; et al. 2021. Program Synthesis with Large Language Models. *arXiv preprint arXiv:2108.07732*.

Bai, Y.; Kadavath, S.; Kundu, S.; Askell, A.; et al. 2022. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073*.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pondé de Oliveira Pinto, H.; Kaplan, J.; et al. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*.

Deng, X.; Gu, Y.; Zheng, B.; Chen, S.; Stevens, S.; Wang, B.; Sun, H.; and Su, Y. 2023. Mind2Web: Towards a Generalist Agent for the Web. *arXiv preprint arXiv:2306.06070*.

Ellis, K.; Ritchie, D.; Solar-Lezama, A.; and Tenenbaum, J. B. 2018. Learning to Infer Graphics Programs from Hand-Drawn Images. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, 6062–6071.

Ha, D.; and Eck, D. 2018. A Neural Representation of Sketch Drawings. In *International Conference on Learning Representations (ICLR)*.

Hendrycks, D.; Basart, S.; Kadavath, S.; Mazeika, M.; Arora, A.; Guo, E.; et al. 2021. Measuring Coding Challenge Competence With APPS. In *NeurIPS 2021 (Datasets and Benchmarks Track)*.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C. L.; and Girshick, R. 2017. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kim, H.; and Kim, S. 2025. MacroBench: A Novel Testbed for Web Automation Scripts via Large Language Models. arXiv preprint arXiv:2510.12345.

Koh, J. Y.; Lo, R.; Jang, L.; Duvvur, V.; Lim, M.; Huang, P.-Y.; Neubig, G.; Zhou, S.; Salakhutdinov, R.; and Fried, D. 2024. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 881–905.

Liang, P.; Bommasani, R.; Lee, T.; Tsipras, D.; et al. 2022. Holistic Evaluation of Language Models. *arXiv preprint arXiv:2211.09110*.

Liu, E. Z.; Guu, K.; Pasupat, P.; Shi, T.; and Liang, P. 2018. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Liu, X.; Yu, H.; Zhang, H.; Xu, Y.; Lei, X.; Lai, H.; Gu, Y.; Ding, H.; et al. 2024. AgentBench: Evaluating LLMs as Agents. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; et al. 2023. Self-Refine: Iterative Refinement with Self-Feedback. *arXiv preprint arXiv:2303.17651*.

Mirzaee, R.; Rajaby Faghihi, H.; Ning, Q.; and Kordjamshidi, P. 2021. SPARTQA: A Textual Question Answering Benchmark for Spatial Reasoning. In *Proceedings of the*

2021 Conference of the North American Chapter of the ACL: Human Language Technologies (NAACL-HLT), 4582–4598.

Phan, L.; Gatti, A.; Han, Z.; et al. 2025. Humanity's Last Exam. *arXiv preprint arXiv:2501.14249*.

Shinn, N.; Cassano, F.; Labash, B.; Gopinath, A.; Narasimhan, K.; and Yao, S. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. *arXiv preprint arXiv:2304.13007*.

Srivastava, A.; et al. 2022. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Zhang, C.; Gao, F.; Jia, B.; Zhu, Y.; and Zhu, S.-C. 2019. RAVEN: A Dataset for Relational and Analogical Visual Reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A. C.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; Alon, U.; and Neubig, G. 2023. WebArena: A Realistic Web Environment for Building Autonomous Agents. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

Appendix

Drawing Application UI Details

UI Layout and Coordinates

The DrawingBench application uses a browser-based drawing interface with the following layout (see Figure 1):

Canvas Specifications:

- Size: $1000 imes 700$ pixels
- Position: (90, 70) to (1090, 770) in screen coordinates
- Background: White (#FFFFFF)
- Border: 1px solid black

Tool Panel (Left Side):

Table 8: Tool Coordinates and Functions

Tool	Coordinates	Size	Function
Pen	(35, 45)	$30 imes 30$ px	Free-form drawing
Eraser	(35, 125)	$30 imes 30$ px	Erase content
Fill	(35, 205)	$30 imes 30$ px	Fill enclosed areas
Line	(35, 285)	$30 imes 30$ px	Draw straight lines
Rectangle	(35, 365)	$30 imes 30$ px	Draw rectangles
Circle	(35, 445)	$30 imes 30$ px	Draw circles

Color Palette (Top):

Table 9: Color Coordinates and Hex Values

Color	Coordinates	Hex Code
Black	(405, 25)	#000000
Red	(429, 25)	#FF0000
Green	(453, 25)	#00FF00
Blue	(477, 25)	#0000FF
Yellow	(501, 25)	#FFFF00
Magenta	(525, 25)	#FF00FF
Cyan	(549, 25)	#00FFFF
White	(573, 25)	#FFFFFF

Size Options:

- Small: 2px (default)
- Medium: 5px
- Large: 10px

Detailed Evaluation Criteria

Our automated evaluation system checks 8 objective criteria for each drawing submission. Table 10 provides complete specifications for each criterion.

Scoring Formula

The final score is computed as:

$$\text{Score} = \frac{\sum_{i=1}^{8} w_i \cdot c_i}{\sum_{i=1}^{8} w_i} \tag{2}$$

where w_i is the weight and $c_i \in \{0, 1\}$ indicates whether criterion i is satisfied.

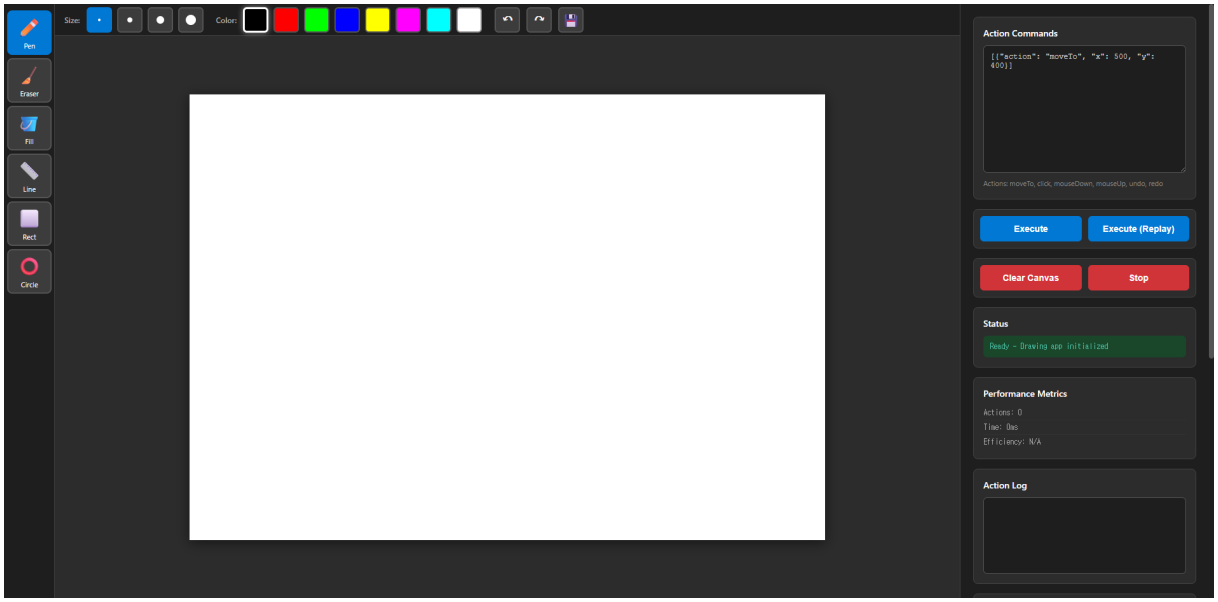

Figure 1: DrawingBench UI: toolbar (left), menu (top), 1000×700px canvas (center), control panel (right).

Error Classification

Errors are categorized into four types:

- **SYNTAX_ERROR**: Invalid JSON format, unknown actions, or malformed commands
- **COORDINATE_ERROR**: Coordinates outside canvas bounds or invalid numeric values
- **LOGIC_ERROR**: Missing required tools/colors, insufficient segments, or unmet constraints
- **EFFICIENCY_WARNING**: Excessive actions, redundant tool switches, or suboptimal coverage

Task Distribution and Statistics

Complete Category Breakdown

Table 11 shows the distribution of all 250 drawing tasks across 20 categories.

Task Complexity Characteristics

Easy Tasks (112): Single-object drawings with clear specifications (e.g., "Draw a red circle in the center"). Average 3-5 required actions.

Medium Tasks (97): Multi-object scenes or drawings with multiple constraints (e.g., "Draw a blue square above a red circle"). Average 8-12 actions.

Hard Tasks (21): Complex compositions with precise positioning requirements (e.g., "Draw 4 squares in each corner of the canvas"). Average 15-20 actions.

Very Hard Tasks (20): Highly complex patterns requiring extensive procedural accuracy (e.g., "Draw an 8×8 checkerboard pattern"). Average 25+ actions.

Action Sequence Examples

Example 1: Drawing a Red Circle

```
[
  {"action": "moveTo", "x": 35, "y": 45},
  {"action": "click"},
  {"action": "moveTo", "x": 429, "y": 25},
  {"action": "click"},
  {"action": "moveTo", "x": 590, "y": 420},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 640, "y": 420},
  {"action": "moveTo", "x": 665, "y": 445},
  {"action": "moveTo", "x": 665, "y": 495},
  {"action": "moveTo", "x": 640, "y": 520},
  {"action": "moveTo", "x": 590, "y": 520},
  {"action": "moveTo", "x": 565, "y": 495},
  {"action": "moveTo", "x": 565, "y": 445},
  {"action": "moveTo", "x": 590, "y": 420},
  {"action": "mouseUp"}
]
```

Example 2: Drawing a Blue Rectangle

```
[
  {"action": "moveTo", "x": 35, "y": 365},
  {"action": "click"},
  {"action": "moveTo", "x": 477, "y": 25},
  {"action": "click"},
  {"action": "moveTo", "x": 400, "y": 300},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 700, "y": 500},
  {"action": "mouseUp"}
]
```

Example 3: Drawing 4 Squares in Corners

```
[
  {"action": "moveTo", "x": 35, "y": 365},
  {"action": "click"},
  {"action": "moveTo", "x": 120, "y": 100},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 220, "y": 200},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 870, "y": 100},
```

Table 10: Complete Evaluation Criteria Specifications

Criterion	Description	Weight
Required Tools	All tools specified in task must be selected at least once	0.20
Required Colors	All colors specified must be used in drawing actions	0.20
Minimum Segments	Drawing must contain at least N mouse-drag segments (task-specific)	0.15
Canvas Coverage	Drawing must cover at least X% of canvas area (typically 0.30)	0.10
Position Constraint	Drawing must satisfy position requirement (center, corner, inside, etc.)	0.15
Size Constraint	If specified, drawn elements must meet size requirements	0.10
Syntax Validity	All action commands must be valid JSON with correct structure	0.05
Coordinate Bounds	All coordinates must be within canvas boundaries (0-1000, 0-700)	0.05

```
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 970, "y": 200},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 120, "y": 640},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 220, "y": 740},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 870, "y": 640},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 970, "y": 740},
  {"action": "mouseUp"}
]
```

Example 4: Multi-Color Drawing (Red Circle Above Blue Square)

```
[
  {"action": "moveTo", "x": 35, "y": 445},
  {"action": "click"},
  {"action": "moveTo", "x": 429, "y": 25},
  {"action": "click"},
  {"action": "moveTo", "x": 590, "y": 250},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 640, "y": 250},
  {"action": "moveTo", "x": 665, "y": 275},
  {"action": "moveTo", "x": 665, "y": 325},
  {"action": "moveTo", "x": 640, "y": 350},
  {"action": "moveTo", "x": 590, "y": 350},
  {"action": "moveTo", "x": 565, "y": 325},
  {"action": "moveTo", "x": 565, "y": 275},
  {"action": "moveTo", "x": 590, "y": 250},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 35, "y": 365},
  {"action": "click"},
```

Table 11: Complete Task Distribution by Category and Difficulty

Category	Easy	Medium	Hard	V.Hard	Total	%
spatial_reasoning	35	24	8	4	71	28.4%
basic_shapes	20	15	5	2	42	16.8%
objects	12	10	4	2	28	11.2%
compositional	4	4	2	1	11	4.4%
patterns	3	3	2	1	9	3.6%
multi_color	4	2	1	1	8	3.2%
position	3	2	1	1	7	2.8%
size	2	2	1	1	6	2.4%
tool_usage	2	2	1	0	5	2.0%
efficiency_test	1	1	0	0	2	0.8%
creative	1	1	0	0	2	0.8%
angle	1	1	0	0	2	0.8%
complex	1	1	0	0	2	0.8%
scenes	0	0	0	1	1	0.4%
precision_test	0	1	0	0	1	0.4%
tool_switching	0	1	0	0	1	0.4%
spatial	0	1	0	0	1	0.4%
symmetry	0	0	0	1	1	0.4%
texture	0	0	1	0	1	0.4%
grid	0	0	0	1	1	0.4%
Total	**112**	**97**	**21**	**20**	**250**	**100%**

```
  {"action": "moveTo", "x": 477, "y": 25},
  {"action": "click"},
  {"action": "moveTo", "x": 515, "y": 470},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 665, "y": 620},
  {"action": "mouseUp"}
]
```

Example 5: Complex Pattern (Grid of 6 Circles)

```
[
  {"action": "moveTo", "x": 35, "y": 445},
  {"action": "click"},
  {"action": "moveTo", "x": 200, "y": 200},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 280, "y": 280},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 450, "y": 200},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 530, "y": 280},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 700, "y": 200},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 780, "y": 280},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 200, "y": 460},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 280, "y": 540},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 450, "y": 460},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 530, "y": 540},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 700, "y": 460},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 780, "y": 540},
  {"action": "mouseUp"}
```

```
]
```

Example 6: Using Fill Tool (House with Filled Roof)

```
[
  {"action": "moveTo", "x": 35, "y": 45},
  {"action": "click"},
  {"action": "moveTo", "x": 400, "y": 300},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 400, "y": 550},
  {"action": "moveTo", "x": 650, "y": 550},
  {"action": "moveTo", "x": 650, "y": 300},
  {"action": "moveTo", "x": 400, "y": 300},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 525, "y": 180},
  {"action": "mouseDown"},
  {"action": "moveTo", "x": 400, "y": 300},
  {"action": "moveTo", "x": 650, "y": 300},
  {"action": "moveTo", "x": 525, "y": 180},
  {"action": "mouseUp"},
  {"action": "moveTo", "x": 35, "y": 205},
  {"action": "click"},
  {"action": "moveTo", "x": 429, "y": 25},
  {"action": "click"},
  {"action": "moveTo", "x": 525, "y": 240},
  {"action": "click"}
]
```

Feedback Examples

Example 1: Missing Tool Error

```
Score: 0.75/1.00 - Good attempt! 1 error(s) found.

ERRORS:
  1. Required tool 'rectangle' was not used.
     Select it by clicking at coordinates (35, 365).

SUGGESTIONS:
  1. Use the rectangle tool for more efficient shape creation.
  2. Drawing coverage is good (0.42).

MISSING CRITERIA:
  - required_tools: ['rectangle']
```

Example 2: Insufficient Coverage

```
Score: 0.80/1.00 - Good job! 1 warning(s) found.

WARNINGS:
  1. Drawing is too small (coverage: 0.12).
     Consider using larger coordinates to cover more
          canvas area.

SUGGESTIONS:
  1. Aim for at least 0.30 coverage of the canvas.
  2. All required tools and colors were correctly used.

CURRENT STATS:
  - Tools used: ['pen']
  - Colors used: ['#FF0000']
  - Segments: 3
  - Coverage: 0.12
```

Example 3: Perfect Score

```
Score: 1.00/1.00 - Excellent! Perfect score!

All criteria met:
  + Required tools: ['pen']
  + Required colors: ['\#FF0000']
  + Minimum segments: 1
  + Minimum coverage: 0.30
  + Position constraint: center
  + No errors detected

Great job! No improvements needed.
```

Additional Performance Analysis

Model-Specific Performance

Table 12: Per-Model Performance Breakdown

Model	T1 Score	T2 Score	Perfect %	Time (min)
Claude-4 Sonnet	0.931	0.958	94.4%	84.2
GPT-4.1	0.927	0.955	93.2%	86.3
GPT-4.1-mini	0.919	0.949	90.8%	77.1
Gemini-2.5 Flash	0.923	0.951	92.0%	90.6
Average	**0.925**	**0.953**	**92.6%**	**84.6**

Category-Level Detailed Results

Table 13: Complete Category Performance (All 20 Categories)

Category	Tests	T1 Avg	T2 Avg	Improve	Perfect T1	Perfect T2
spatial_reasoning	71	0.959	0.973	+0.015	53	67
basic_shapes	42	0.945	0.968	+0.023	35	40
objects	28	0.912	0.941	+0.029	20	25
compositional	11	0.957	0.998	+0.041	9	11
patterns	9	0.889	0.925	+0.036	6	8
multi_color	8	0.931	0.956	+0.025	6	7
position	7	0.943	0.971	+0.028	5	6
size	6	0.917	0.950	+0.033	4	5
tool_usage	5	0.960	0.980	+0.020	4	5
efficiency_test	2	1.000	1.000	0.000	2	2
creative	2	0.924	1.000	+0.076	1	2
angle	2	0.881	0.980	+0.100	1	2
complex	2	0.900	0.975	+0.075	1	2
scenes	1	0.672	1.000	+0.328	0	1
precision_test	1	1.000	1.000	0.000	1	1
tool_switching	1	1.000	1.000	0.000	1	1
spatial	1	1.000	1.000	0.000	1	1
symmetry	1	0.950	0.950	0.000	0	0
texture	1	0.875	0.925	+0.050	0	0
grid	1	0.800	0.875	+0.075	0	0

Error Type Distribution by Difficulty

Token Usage Statistics

Sample Drawing Prompts by Category

Easy Prompts

1. Draw a red circle in the center of the canvas
2. Draw a blue square in the top-left corner
3. Draw a green horizontal line across the middle
4. Draw a yellow triangle pointing upward
5. Fill the entire canvas with cyan color

Table 14: Error Distribution Across Difficulty Levels (Turn 1)

Error Type	Easy	Medium	Hard	Very Hard
SYNTAX_ERROR	1	1	0	1
COORDINATE_ERROR	2	3	1	2
LOGIC_ERROR	4	5	2	4
EFFICIENCY_WARNING	12	15	6	9
Total	**19**	**24**	**9**	**16**

Table 15: Average Token Usage per Test

Model	Input Tokens	Output Tokens	Total
Claude-4 Sonnet	1,243	387	1,630
GPT-4.1	1,256	412	1,668
GPT-4.1-mini	1,198	356	1,554
Gemini-2.5 Flash	1,287	398	1,685
Average	**1,246**	**388**	**1,634**

Medium Prompts

1. Draw a house with a triangular roof and rectangular body
2. Draw a simple face with two eyes, a nose, and a mouth
3. Draw three overlapping circles of different colors
4. Draw a tree with a brown trunk and green leaves
5. Create a pattern with alternating red and blue vertical stripes

Hard Prompts

1. Draw 4 identical squares, one in each corner of the canvas
2. Create a $3imes3$ grid of circles, all the same size
3. Draw a star with 5 points using only straight lines
4. Create a rainbow with 7 colored arcs
5. Draw a flower with 8 petals arranged symmetrically

Very Hard Prompts

1. Draw a checkerboard pattern with $8imes8$ squares
2. Create a spiral pattern starting from the center
3. Draw a complex geometric mandala with rotational symmetry
4. Create a pixel art character using a $16imes16$ grid
5. Draw a detailed landscape with hills, trees, sun, and clouds

Evaluation Criteria Details

Required Tools Criterion

This criterion checks whether the LLM selected and used specific drawing tools as required by the task.

Evaluation Method:

1. Parse action sequence for tool selection actions
2. Identify clicked coordinates matching tool positions
3. Verify required tools are in the set of used tools
4. Score: 1.0 if all required tools used, 0.0 otherwise

Common Failure Modes:

- Forgetting to select the tool before drawing
- Using wrong tool (e.g., pen instead of rectangle)
- Tool state not properly tracked across actions

Canvas Coverage Criterion

This criterion measures what percentage of the canvas area is covered by the drawing.

Calculation:

$$\text{Coverage} = \frac{\text{Bounding Box Area}}{\text{Canvas Area}} \tag{3}$$

where Bounding Box is the smallest rectangle containing all drawn content.

Thresholds:

- Minimum coverage: typically 0.20-0.30
- Good coverage: 0.40-0.60
- High coverage: ¿0.60

Position Constraint Criterion

This criterion verifies whether drawn elements are positioned correctly according to the prompt.

Supported Constraints:

- `center`: element must be within central 20% of canvas
- `top-left`: element in top-left quadrant
- `top-right`: element in top-right quadrant
- `bottom-left`: element in bottom-left quadrant
- `bottom-right`: element in bottom-right quadrant
- `corners`: elements in all four corners

Implementation Details

Technology Stack

Frontend:

- HTML5 Canvas API for drawing
- JavaScript for UI interactions
- Puppeteer for browser automation

Backend:

- Python 3.11+ for evaluation engine
- OpenAI, Anthropic, Google AI APIs for LLM access
- JSON for data serialization

Evaluation Pipeline:

1. Load prompt from dataset
2. Send to LLM with UI specification
3. Parse LLM response (JSON action sequence)
4. Execute actions in browser via Puppeteer
5. Capture canvas state
6. Run rule-based evaluation
7. Generate structured feedback
8. (Optional) Send feedback to LLM for Turn 2
9. Store results and metrics

Code Availability

All code, data, and documentation for DrawingBench are available at:

`[Repository URL to be added upon publication]`

The repository includes:

- Complete dataset (250 prompts with metadata)
- Drawing application (HTML/JS)
- Evaluation engine (Python)
- Feedback generator (Python)
- LLM testing scripts
- Result analysis notebooks
- Documentation and examples

Limitations and Threats to Validity

Several limitations of our study warrant discussion:

Rule-based Evaluation Constraints. Our automated evaluation system excels at objective criteria (e.g., position, color, coverage) but cannot assess subjective qualities like aesthetic appeal or creative interpretation. Some drawings that violate our criteria might still be considered acceptable or even preferable by human judges. Incorporating human evaluation or learned quality metrics could provide a more holistic assessment.

Text-only Paradigm. While we demonstrate that LLMs can perform spatial tasks without vision, real-world agents increasingly use multimodal models with visual perception. Our benchmark's text-only design may not fully represent the capabilities or challenges of vision-enabled LLM agents. Extending DrawingBench to a vision-based mode (where models receive canvas screenshots and generate actions) would be a valuable direction for future work.

Limited Task Diversity. Although our 250 prompts span 20 categories and 4 difficulty levels, they focus primarily on geometric and compositional tasks. Other spatial reasoning aspects, such as 3D spatial reasoning, dynamic movement, or physics-based interactions, are not covered. Expanding the benchmark to include such tasks would broaden its applicability.

Model Selection. We evaluated four state-of-the-art models, but the LLM landscape is rapidly evolving. Results may vary with newer models, different prompting strategies, or fine-tuned versions specialized for agent tasks. Continuous evaluation of emerging models will be necessary to track progress.

Generalization to Real-world UIs. Our drawing application is a simplified, controlled environment. Real-world UIs have greater complexity, including dynamic elements, latency, error states, and non-deterministic behaviors. While our benchmark tests core spatial reasoning and action generation, generalizing these findings to production agent systems requires further research.

Broader Impact and Ethical Considerations

The development of LLM-based agents capable of UI interaction raises important ethical and societal considerations. On the positive side, such agents could democratize access to digital tools, assist users with disabilities, and automate tedious tasks. However, they also pose risks, including:

- **Automation of harmful activities**: Agents with UI control could be misused for automated phishing, spam, or unauthorized access.
- **Job displacement**: Widespread deployment of UI automation agents might disrupt employment in fields like data entry, software testing, or customer support.
- **Privacy and security**: Agents interacting with UIs may access sensitive information, requiring robust safeguards.
- **Bias and fairness**: If agents are trained or evaluated on biased datasets, they may perpetuate inequities in access or outcomes.

Our benchmark, being open-source and focused on a simplified drawing task, has limited direct ethical implications. However, as the techniques and insights from DrawingBench are applied to real-world systems, developers must consider these broader impacts and implement appropriate safeguards.

Future Directions

Building on our findings, we identify several promising directions for future research:

Vision-based Evaluation. Extending DrawingBench to support vision-language models (VLMs) would enable direct comparison between text-only and vision-augmented spatial reasoning. This could reveal whether visual perception provides significant advantages for spatial tasks or if text-based reasoning suffices.

Expanded Task Coverage. Incorporating tasks that require temporal reasoning (e.g., animated drawings), 3D spatial concepts (e.g., perspective drawing), or physical simulation (e.g., balancing objects) would broaden the benchmark's scope and challenge models in new ways.

Human-in-the-loop Feedback. Replacing or augmenting our rule-based feedback with human annotations could test whether models can learn from more naturalistic, less structured guidance. This would also allow exploration of how models handle ambiguous or conflicting feedback.

Fine-tuning and Specialization. Our study evaluates general-purpose LLMs. Training models specifically on drawing or spatial tasks could reveal the extent to which spatial reasoning can be improved through targeted learning.

Cross-domain Transfer. Investigating whether strong performance on DrawingBench correlates with success on other agent tasks (e.g., web navigation, code generation) would shed light on the generalizability of spatial reasoning skills.