Draw2Think: Harnessing Geometry Reasoning through Constraint Engine Interaction

Juncheng Hu [1 2 3] **Jiawei Du** [2 3] **Xin Zhang** [2 3] **Joey Tianyi Zhou** [2 3]

Project Page Github

Abstract

Vision-language models solve geometry problems with rising accuracy, yet their intermediate states remain latent and unverifiable: a relation expressed in textual reasoning or drawing code carries no guarantee that a constraint-satisfying configuration realizes it. We observe that existing externalization methods based on rendered pixels or one-shot scripts fail to provide exact, per-action geometric guarantees. Enforcing geometric relations by algebraic definition closes this gap: the workspace becomes a constraint-checked evolving canvas. We present **Draw2Think**, a framework that recasts geometric reasoning from latent spatial inference into agentic interaction with the GeoGebra constraint engine. In a **Propose-Draw-Verify** loop, Draw2Think externalizes hypotheses onto an executable canvas, measures exact geometric quantities, and feeds structured observations back to the model, so subsequent reasoning proceeds from checked canvas state grounded by the shared workspace. This externalization makes two properties separately auditable: model-level *Construction Fidelity* (whether the canvas realizes the intended configuration) and engine-level *Measurement Faithfulness* (exact values and relations from canvas constraints). Across construction, outcome, and rendering evaluations, Draw2Think builds canvases that pass 95.9% predicate-level and 84.0% strict problem-level construction checks on *GeoGoal*, improves outcome accuracy by up to 4.1%/16.4% on planar/solid benchmarks, and attains 68.2%/90.5% strict/relaxed rendering scores on *GenExam-math*.

[1]National University of Singapore, Singapore [2]Centre for Frontier AI Research, Agency for Science, Technology and Research, Singapore [3]Institute of High Performance Computing, Agency for Science, Technology and Research, Singapore. Correspondence to: Juncheng Hu <juncheng.hu@u.nus.edu>.

The 3rd *AI for Math Workshop* at the 43rd International Conference on Machine Learning (ICML), Seoul, South Korea, 2026.
Copyright 2026 by the author(s).

1. Introduction

Geometry problem solving pairs visual perception with multi-step deductive reasoning, making it a canonical stress test for vision-language model (VLM) capability (Ma et al., 2025; Zhao et al., 2025b). Benchmarks span planar, analytic, and solid regimes (Lu et al., 2021; Zhang et al., 2024a;b; Wang et al., 2025b; He et al., 2024); progress is measured almost exclusively by end-to-end outcome accuracy, which conflates perception, construction, and deduction into a single signal.

Geometry is particularly well-structured for agentic interaction with an external workspace: actions are typed, state transitions are deterministic, intermediate quantities are exactly measurable, and invalid operations surface as errors rather than silent approximations. Yet most VLMs still solve it without such a workspace; they reason in latent tokens, where an 89° angle is visually indistinguishable from 90° despite encoding a different relation (Zhang et al., 2024b). Outcome accuracy therefore hides an accountability gap: mid-reasoning claims such as angles, auxiliary lines, and derived lengths become premises for later steps without any engine confirming the underlying configuration.

Existing systems address fragments of this gap without closing it: formal provers certify proof steps but not the per-action construction states benchmark answers depend on (Chervonyi et al., 2025; Zhao et al., 2026); trained VLM solvers improve outcome accuracy while leaving intermediate states approximate (Guo et al., 2025; Wang et al., 2025c; Jiao et al., 2026); code- and render-externalization systems execute or re-perceive external artifacts but offer no engine-exact per-action check (Wei et al., 2025; Weng et al., 2025; Luo et al., 2026b). The missing piece is a *verifiable intermediate state*: constructions that are executable, revisable, and independently checkable against geometric ground truth (Fu et al., 2025). When a model can directly measure an angle from a verified construction, it sidesteps fragile theorem-selection entirely.

The root cause is the timing and authority of feedback: perception, self-check, and post-hoc execution expose artifacts without certifying construction-time validity. **Draw2Think** takes a fourth route: a *constraint-agentic harness* that wraps

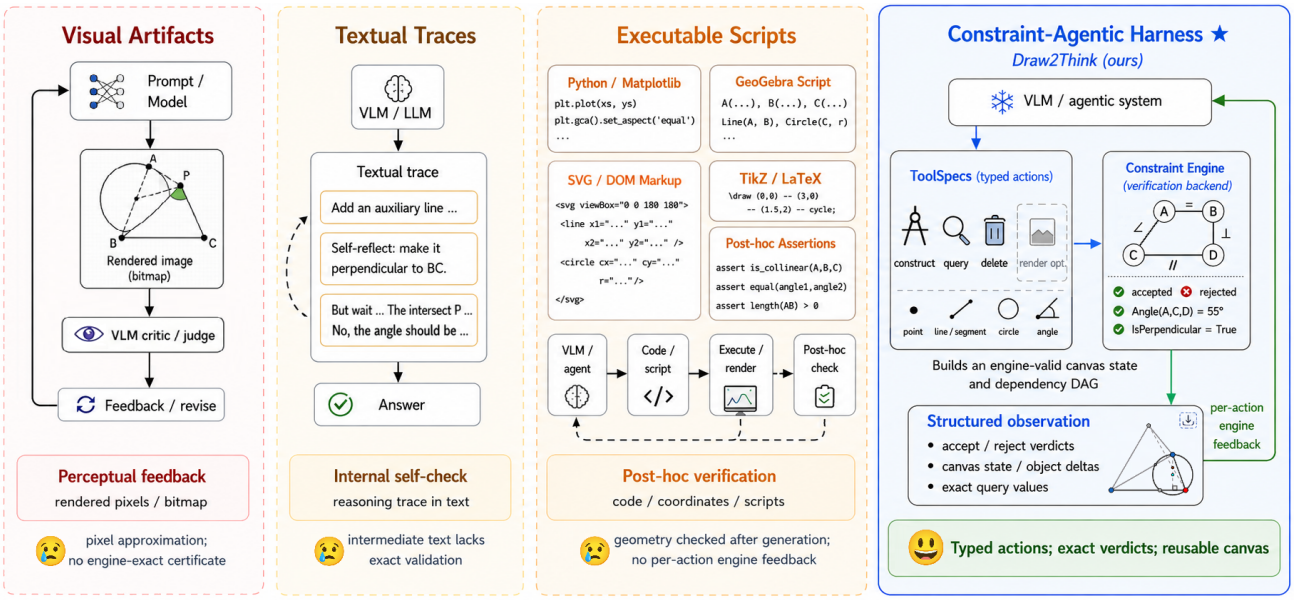


Figure 1. **Paradigms for externalizing intermediate geometry.** Prior routes externalize intermediate geometry as visual artifacts, textual traces, or executable scripts. **Draw2Think** adds a constraint-agentic harness: a frozen VLM selects typed ToolSpecs, the GeoGebra engine updates an engine-valid canvas state, and structured observations return after each action. The distinction is less about externalizing state than about when verification enters the loop.

a frozen VLM around a dynamic-geometry engine through typed ToolSpecs. In dynamic geometry systems such as GeoGebra (Hohenwarter & Hohenwarter), construction commands enforce geometric relationships algebraically rather than by coordinate approximation, making the model's geometric priors executable and each accepted action checkable. The VLM proposes typed actions, the engine executes or rejects them on a shared canvas, and structured observations ground the next step; trajectory-level strategy remains with the model.

We evaluate this design *training-free*, instantiating Draw2Think on Gemini 3 Flash Preview (Google DeepMind, 2025b) throughout: frontier VLMs already carry geometric priors strong enough to benefit from an external workspace, while their intermediate claims remain unreliable without grounded verification. Controlled comparisons on eight planar and two solid benchmarks show *selective* outcome gains up to 4.1% on planar and 16.4% on solid: gains are largest on visually grounded benchmarks (*GeoSketch* (Weng et al., 2025), *MathVerse* (Zhang et al., 2024b)) and become negative where the baseline is already saturated (*Geo3K* (Lu et al., 2021), *PGPS9K* (Zhang et al., 2024a)); ablations (§5) isolate which components drive these gains.

Three contributions thread through the paper:

- **Formulation.** We recast geometric reasoning as an inference-time closed loop around a constraint engine: one substrate serves as executor, verifier, and generator while the frozen VLM perceives, reasons, and dispatches typed tools over a verifiable, revisable canvas.
- **Evidence.** We validate the approach along three axes: outcome gains concentrate where visual measurement or spatial construction is the bottleneck (Table 2); constructed canvases pass 95.9% predicate-level and 84.0% strict problem-level checks on GeoGoal (Chen et al., 2026a); and the same framework transfers to solid geometry and GenExam-math rendering without changing the loop.
- **Mechanism.** We localize when grounding changes behavior, through failure taxonomy, answer provenance, and graduated ablations (§5).

2. Related Work

Geometry problem solving requires jointly perceiving spatial relations from diagrams and performing multi-step deductive reasoning over them (Ma et al., 2025; Zhao et al., 2025b). A defining difficulty is that intermediate states such as auxiliary lines, derived measurements, and construction hypotheses are reused as premises by later steps, so an unverified local hallucination becomes a hidden global premise. Formal theorem proving (Chervonyi et al., 2025; Zhao et al., 2026; Sicca et al., 2024) addresses this failure mode by certifying entire proof chains against axioms. Its guarantees are proof-targeted rather than measurement-targeted: the certified objects are derivation steps rather than the per-action numerical states that downstream readouts depend on. For solving and rendering, the complementary guarantee

is measurement faithfulness at query time; this motivates inference-time verification with per-step engine feedback.

Textual and visual reasoning traces. A contrasting line records intermediate state as textual or visual traces, leaving geometric validity uncertified. Recent diagnostics localize the fragility: VLMs show weak fine-grained grounding on geometry diagrams (Sun et al., 2026b), and applying identified geometric principles within the visual context remains a dominant bottleneck (Xu et al., 2025a). Textual auxiliary-line systems expose the same validation gap: GeoVLMath (Guo et al., 2025) finds incorrect auxiliary lines worse than omitting them, and GeometryZero (Wang et al., 2025c) shows that unconditional auxiliary construction can be counterproductive; Socratic-Geo (Jiao et al., 2026) instead validates generated training triplets, leaving inference-time object validity outside the loop. Visual-trace systems fare similarly: MathCanvas (Shi et al., 2025) and ThinkMorph (Gu et al., 2026) emit pixel intermediates for mode-switched chain-of-thought (CoT), leaving rendered geometry without a constraint-level certificate. Across this line, intermediate states may guide later reasoning without receiving certification from an exact engine query.

Executable scripts and rendered feedback. Externalized artifacts make state visible, while per-action geometric exactness remains unguaranteed. *Coordinate-first* systems commit numerical (x, y) values in code: GeoCode (Wu et al., 2026) generates drawing programs, GGBench (Wei et al., 2025) reports 79% executability against 57% geometric correctness, and GeoSDF (Zhang et al., 2025) exposes a final diagram after signed-distance optimization. Rendered-feedback loops return the artifact to a visual channel: Visual Sketchpad (Hu et al., 2024) sketches on matplotlib renders, and Canvas-CoT (Sun et al., 2026a) pairs mutable document object model (DOM) state with a VLM critic against a *reference image*. Across these systems, verification remains code-level, visual, or post-hoc; what is missing is an engine source of exact intermediate measurements.

GeoGebra before inference-time operation. Recent geometry work uses GeoGebra as infrastructure for building resources, executing scripts, evaluating outputs, or shaping rewards. We-Math 2.0 (Qiao et al., 2026) places GeoGebra in its MathBook resource-building pipeline, and Newclid (Sicca et al., 2024) routes a graphical interface into a symbolic backend. GGBench (Wei et al., 2025) freezes GeoGebra code as an evaluation oracle, while Faire (Zhang et al., 2026) executes generated GeoGebra programs and model-synthesized boolean assertions as post-generation verification for reinforcement-learning (RL) reward. What remains underexplored is inference-time operation: the model issues actions one at a time, receives engine verdicts immediately, and conditions later calls on the verified canvas state. Draw2Think wraps the engine in a typed constraint-agentic harness (Schick et al., 2023; Dong et al., 2025; Lin et al., 2026), with the embedded Giac computer algebra system (CAS) serving as a per-action verifier inside the Propose-Draw-Verify loop.

3. Method

3.1. Problem Formulation: Construction as State-Space Search

We consider a geometry problem $\mathcal{P} = (I, T, Q, C)$ with diagram I, text T, question Q, and optional candidate set C. Direct VLM inference maps $\mathcal{P}$ to an answer in a single forward pass. Draw2Think instead searches for a *canvas state* $\mathcal{S}_K$ from which the answer is determined by a bounded readout $h(Q, \mathcal{S}_K)$.

We formalize the canvas as a deterministic environment $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O})$:

Definition 3.1 (Geometric Construction Environment)**.** A canvas environment is a tuple $\mathcal{E} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O})$ where $\mathcal{S}$ is a **state space** of typed geometric objects whose dependencies induce a Directed Acyclic Graph (DAG); $\mathcal{A}$ is an **action space** of 92 typed ToolSpecs (§3.2); $\mathcal{O} : \mathcal{S} \times \mathcal{A} \to \mathcal{Y}$ is the **observation** function (where $\mathcal{Y}$ comprises structured responses: object deltas, exact scalar values, or error messages); and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}'$ is a **deterministic transition**. Construction actions extend $\mathcal{S}$ with new objects; query actions return exact values ($v \in \mathbb{R}$ or $b \in \{\text{TRUE}, \text{FALSE}\}$) without modifying $\mathcal{S}$; delete actions remove an object and all its dependents. Invalid actions are rejected with an explicit error whenever the engine detects the failure.

We instantiate $\mathcal{E}$ with GeoGebra (Hohenwarter & Hohenwarter), a dynamic geometry system in which objects are defined by construction commands rather than free-form coordinates. The key modeling choice is therefore constraint-first: the model declares *which* geometric relation should hold, and the engine commits an object satisfying that relation by construction. The algebraic guarantee behind this execution model is detailed in §3.2.

Solving is then a state-reachability problem. At the *action* level, the model constructs a grounded state through a trajectory $\tau = (a_1, \mathcal{O}_1, \ldots, a_K, \mathcal{O}_K)$ of interleaved actions and observations:

$$\begin{aligned} &\mathcal{S}_k = \mathcal{T}(\mathcal{S}_{k-1}, a_k), \quad \mathcal{O}_k = \mathcal{O}(\mathcal{S}_k, a_k), \\ &\mathcal{S}_0 = \varnothing, \quad \hat{y} = h(Q, \mathcal{S}_K), \end{aligned} \tag{1}$$

where k indexes individual tool calls. Algorithm 1 groups these calls into *turns*: at each turn t, the model may emit multiple actions $\{a_t^{(i)}\}$ before receiving the next observation; Eq. 1 applies sequentially within the turn. For numerical targets, $h = f_{\text{query}}$ reads an engine measurement;

Table 1. Action types and their state semantics. The 55 construction tools include `delete_object`, which removes the target together with its dependents via cascading delete (Definition 3.1).

Type	State Transition	N
Construction (`add_*`)	$\mathcal{S}' = \mathcal{S} \cup \{o_{\text{new}}\}$	55
Query (`query_*`)	$f_{\text{query}}(\mathcal{S}, q) \to v \in \mathbb{R}$	24
Render (`render_*`)	visual style only; $\mathcal{S}$ unchanged	13

for multiple-choice targets, $h = g(\text{text}_t, C, \mathcal{S}_K)$ combines the final text with engine-grounded values. The same final state also supports rendering via $\hat{I} = \rho(\mathcal{S}_K; \eta)$, with task differences captured by the active tool subset and readout.

Eq. 1 leaves three burdens to the model: choosing valid high-arity actions, preserving DAG consistency, and deciding when query-supported readout is sufficient. Draw2Think handles these burdens with a training-free controller: a typed action interface with auditable guarantees (§3.2), composed by the Propose-Draw-Verify loop (Algorithm 1).

3.2. Action Interface and Auditable Guarantees

Action space. The predefined ToolSpecs group forms a typed interface to the canvas, following the structured function-calling view of tool descriptions and typed parameter bindings (Wei et al., 2026): each specification combines a natural-language description (selection cue), typed parameter signatures (invocation contract), and preconditions jointly enforced by schema and backend. Construction tools extend $\mathcal{S}$ with algebraically-defined objects; query tools return exact scalars without modifying $\mathcal{S}$; delete tools prune subtrees via cascading removal (Definition 3.1). Full catalog and representative descriptions in Appendix B.1. For 3D benchmarks (§4.5), this set is extended with 21 solid-geometry tools (Table B.2).

At inference time, each action call decomposes as $a_k = (u_k, \varphi_k)$, where u_k is the ToolSpec choice and φ_k assigns its typed parameters. Let $\mathcal{C}_{k-1} := (\mathcal{P}, \mathcal{A}, \mathcal{H}_{k-1})$ denote the inference-time context at step k (problem $\mathcal{P}$, action surface $\mathcal{A}$, and verified trace history $\mathcal{H}_{k-1} = ((a_j, \mathcal{O}_j))_{j<k}$). The action then factors as

$$\begin{aligned} P(a_k \mid \mathcal{C}_{k-1}) &= P\big((u_k, \varphi_k) \mid \mathcal{C}_{k-1}\big) \\ &= \underbrace{P(u_k \mid \mathcal{C}_{k-1})}_{\text{tool selection}} \cdot \underbrace{P(\varphi_k \mid u_k, \mathcal{C}_{k-1})}_{\text{parameter generation}}. \end{aligned} \quad (2)$$

$\mathcal{P}$ and $\mathcal{A}$ are fixed for the problem instance; only $\mathcal{H}_{k-1}$ evolves between steps. This decomposition makes the ToolSpec layers operational: natural-language descriptions primarily shape tool selection, while typed signatures and preconditions constrain parameter generation. The description ablation in §5.2 preserves the second channel while removing the first, isolating selection-layer guidance.

Construction Fidelity (model-level, empirical). Construction Fidelity is a model-level property: the final canvas satisfies the externally specified geometric predicates of the problem, regardless of whether the final answer is correct. This is not guaranteed by the engine and must be audited empirically; we evaluate it in §4.3. The distinction matters for training-free operation: ToolSpecs expose textbook operations such as "perpendicular," "bisector," and "tangent" as verified primitives, but the model must still choose the right primitives and parameters. Removing natural-language ToolSpec cues while preserving typed signatures raises LLM-bypass $4.6\times$ (§5.2), confirming that selection-layer guidance is part of the method rather than cosmetic documentation.

Measurement Faithfulness (engine-level, algebraic). Measurement Faithfulness is the complementary engine-level property: because accepted objects are stored as algebraic relations and resolved by GeoGebra's embedded Giac CAS (Kovács & Parisse, 2015) using Gröbner-basis elimination (Botana et al., 2015), each query $f_{\text{query}}(\mathcal{S}, q)$ returns the deterministic value $v^*(\mathcal{S}, q)$ implied by the committed canvas. The guarantee is *semantic-free*: it says what follows from $\mathcal{S}$, not whether $\mathcal{S}$ matches the problem's intended configuration. Correct constructions therefore yield exact readouts; incorrect constructions yield exact measurements of the wrong canvas, exposing discrepancies for replanning.

In practice, degenerate inputs account for a small fraction of all failed calls ($\sim$5%, Appendix E.5) and may produce undefined objects without an error message, which propagate as downstream reference failures.

Algorithm 1 Draw2Think Inference

Require: $\mathcal{P}, \mathcal{A}, \mathcal{E}$
$\mathcal{S}_0 \leftarrow \varnothing$; $\mathcal{H} \leftarrow \langle\,\rangle$
for $t = 1, 2, \ldots$ **do**
  $\text{text}_t, \{a_t^{(i)}\} \leftarrow \pi_\theta(\mathcal{P}, \mathcal{A}, \mathcal{H})$ [Propose]
  if $\textsc{Answer} \in \text{text}_t$ **then**
    return $h(\text{text}_t, \mathcal{S}_{t-1})$
  end if
  if $\{a_t^{(i)}\} = \varnothing$ **then**
    return text_t
  end if
  $\mathcal{S} \leftarrow \mathcal{S}_{t-1}$
  for $i = 1$ **to** $|a_t|$ **do**
    $\mathcal{S}, \mathcal{O}_t^{(i)} \leftarrow \mathcal{E}.\textsc{Step}(\mathcal{S}, a_t^{(i)})$ [Draw]
  end for
  $\mathcal{S}_t \leftarrow \mathcal{S}$; $\mathcal{H} \leftarrow \mathcal{H} \cdot \big\langle (a_t^{(i)}, \mathcal{O}_t^{(i)}) \big\rangle_{i=1}^{|a_t|}$ [Verify]
end for

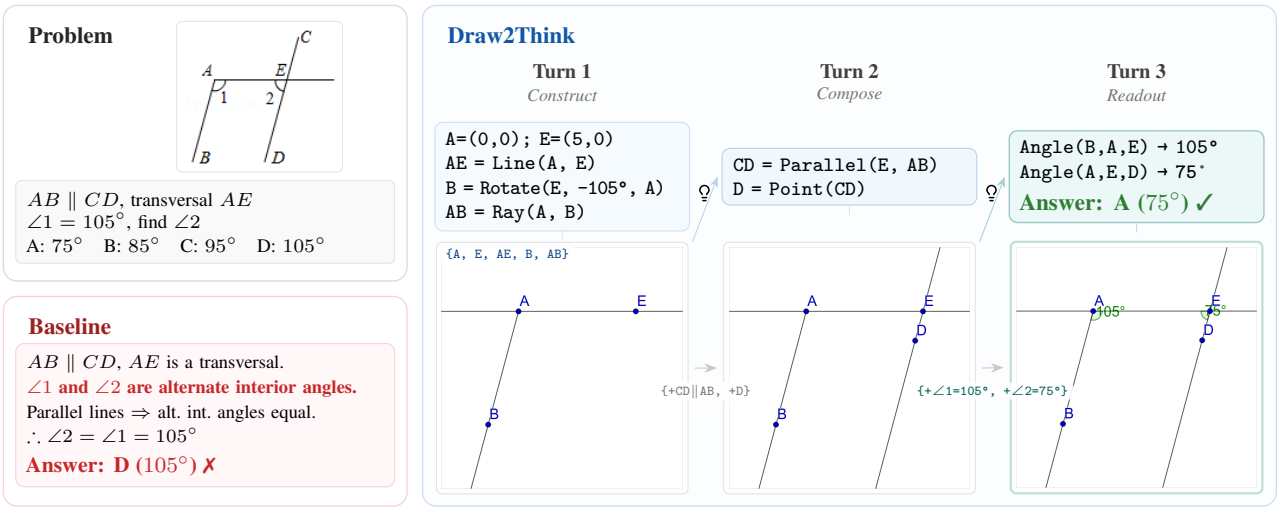


Figure 2. Mechanism comparison on MathVista/290. Baseline applies alternate-interior-angles and concludes 105°; Draw2Think runs *Construct* → *Compose* → *Readout*, with canvas state accumulating across turns, and the readout returns $\angle 2 = 75^\circ$.

3.3. Propose-Draw-Verify Loop

The Propose-Draw-Verify (PDV) loop (Algorithm 1) externalizes geometric hypotheses onto the constraint engine and feeds back structured state after each action, so that each reasoning step is grounded by an engine-verified observation instead of a purely autoregressive continuation. The loop exploits a fundamental asymmetry: proposal is a high-entropy decision over 92 typed tools and their parameters, whereas verification reduces to a deterministic algebraic check. This asymmetry is what makes the closed loop productive: judgment is delegated to the external oracle while the model remains responsible for proposal.

Two properties of the observation distinguish this loop from visual-feedback alternatives:

Structured canvas memory: each observation $\mathcal{O}_t$ is engine-grounded; construction calls expose object-level deltas, and query calls return exact values (Measurement Faithfulness, §3.2). In contrast to visual-feedback loops that re-perceive rendered output, these observations are algebraically exact and directly conditioned upon without a second round of visual interpretation that would compound generation with perception error (Appendix A). The model conditions on the verified history $a_{k+1} \sim \pi_\theta(\cdot \mid \mathcal{P}, \mathcal{A}, \mathcal{H}_k)$ where $\mathcal{H}_k = \big((a_j, \mathcal{O}_j)\big)_{j=1}^{k}$ is the ordered trace of actions and observations and every $\mathcal{O}_j$ is engine-supplied rather than self-attested.

Revisable state: the engine preserves a consistent partial canvas under failure. Invalid actions are rejected explicitly, failed calls roll back locally while valid objects remain available, and the model may additionally invoke cascading delete when it chooses to retract a committed branch (§5.3). This recovery regime is unavailable to open-loop code generation (Wei et al., 2025).

4. Experiments

4.1. Setup

We evaluate along three axes: *construction fidelity* on GeoGoal (Chen et al., 2026a) under an engine-exact predicate verifier (§4.3); *outcome accuracy* on eight planar and two solid benchmarks under ground-truth (GT) match (§4.2, §4.4, §4.5); and *output utility* on GenExam-math (Wang et al., 2026) under VLM-judge scoring (§4.5).

We compare two inference modes of Gemini-3-Flash (Google DeepMind, 2025b). **BL** is a direct vision-language **BaseLine** that answers in one pass from internal state. **CT** is our closed-loop **ConsTruction** mode: the model issues a ToolSpec call, the engine executes it on the shared canvas, and the returned structured feedback conditions the next model step. We define one such model–engine round trip as a turn. Within each benchmark, BL and CT share temperature 0 and the same thinking level (`medium`, *OlympiadBench* and *SolidGeo-hard* at `high`); CT is capped at 30 turns and 120 seconds per turn. All results are single-attempt Pass@1.

4.2. Selective Outcome Gains under a Fixed Backbone

Table 2 summarizes the controlled same-model comparison. CT gains are *selective* and track BL internal-simulation burden: the CT/BL token ratio shifts from > 1 in low-BL-thinking regimes to < 1 in high-BL-thinking regimes. Gains are largest on visually grounded tasks (*MathVerse*, *GeoSketch*) and on the officially hard 3D split (*SolidGeo-*

Table 2. Same-model Pass@1 comparison. Rows are ordered by BL thinking tokens, exposing how CT gains depend on the baseline's observed internal-simulation burden. Save/Break are paired outcome transitions; the right inset profiles the BL traces behind planar Save/Break cases (Cross-system in Appendix C; Wall-time analysis in E.2; Full Save/Break in Appendix E.4.)

Benchmark	N	Outcome accuracy (Pass@1) BL → CT	Save BL✗CT✓	Break BL✓CT✗	Win S/B	Think tokens / prob. BL	CT
GeoQA (Chen et al., 2021)	754	93.6 → **96.9**+3.3	40	15	**2.7**	808	1 785×2.21
PGPS9K (Zhang et al., 2024a)	1000	**94.5** → 90.5-4.0	13	53	0.2	1 043	1 566×1.50
MathVista (Lu et al., 2024) GPS	208	97.1 → **97.6**+0.5	4	3	**1.3**	1 174	1 629×1.39
Geo3K (Lu et al., 2021)	601	98.2 → 95.3-2.9	6	23	0.3	1 485	1 878×1.26
GeoLaux (Fu et al., 2025)	221	93.2 → **94.1**+0.9	11	9	**1.2**	2 301	2 872×1.25
MathVerse (Zhang et al., 2024b) Plane	510	87.5 → **91.0**+3.5	29	11	**2.6**	2 637	2 810×1.07
GeoSketch (Weng et al., 2025)	390	81.8 → **85.9**+4.1	26	10	**2.6**	5 498	**3 545**×0.64
OlympiadBench (He et al., 2024)	112	89.3 → 89.3+0.0	4	4	**1.0**	5 884	**5 063**×0.86
MathVerse-solid	119	82.4 → **88.2**+5.8	10	3	**3.3**	2 121	**1 897**×0.89
SolidGeo-hard (Wang et al., 2025b) (Lv.3)	177	59.9 → **76.3+16.4**	36	7	**5.1**	9 000	**8 114**×0.90

Save/Break reasoning effort comparison: higher → harder

BL think trace	Save	Break
Empty BL Log	51%	0%
BL Median	5206.5	1503.0
BL Mean	6004.7	3166.9

Wall time by BL-thinking regime.

Thinking burden group	BL	CT tot.	CT /turn
Low BL-think	10.2	21.3	9.2
High BL-think	35.7	35.9	12.6

hard), where exact external measurement replaces expensive latent simulation. They shrink on already-strong textbook benchmarks and reverse on the saturated cluster (*Geo3K*, *PGPS9K*), where forced construction adds overhead without resolving a real ambiguity. Within *GeoLaux*, gains increase with annotated reference-solution length. Engine grounding matters most when the bottleneck is deriving exact quantities from visual input, whether through perceptual estimation or theorem selection, and becomes overhead when the model already has a cheap internal route.

4.3. Canvas-Level Construction Fidelity

Accuracy validates the *outcome* axis; *Construction Fidelity* (§3.2) requires a separate audit. We therefore audit each final canvas on GeoGoal (Chen et al., 2026a) against ground-truth predicates from TrustGeoGen's (Fu et al., 2026) formal engine. The audit is *offline and deterministic*: Newclid (Sicca et al., 2024) checks each predicate only from named-point coordinates, without access to the model's reasoning trace or final answer (13,254 checks; Table 3; Figure 3). SR remains stable across premise, numerical-check, and derived predicates, indicating that typed tool composition does not erode with derivation depth.

Figure 3 decomposes the residual. The gap between faithful and unfaithful canvases widens as the T_i-match threshold tightens because structural errors propagate globally across derived readouts, whereas precision limits affect only the specific numerical queries they involve. The two failure modes also split cleanly by expression type: angle T_i concentrate structural errors (97%), length T_i concentrate precision errors (100%; Appendix D.3). Engine-exact verification separates them: an LLM-judge would collapse both failure modes into a single aggregate signal, while per-predicate checking splits them for independent mitigation.

Table 4. Backbone portability on *OlympiadBench* (N=112; BL correct = 100/112 for all three models(Google DeepMind, 2025b; Anthropic, 2026; OpenAI, 2026); CT correct = 100/103/106).

Model	BL	CT	Cost
Gemini 3 Flash @high	89.3	→ 89.3 +0.0	$4.8
Claude Sonnet 4.6@high	89.3	→ 92.0 ↑2.7	$26.0
GPT-5.4@high	89.3	→ **94.6** ↑5.4	$40.5

Table 5. GenExam-math: top closed-source systems (full leaderboard online and in Appendix F.1).

Method	Base / date	Strict	Relax
Draw2Think	Gemini 3 Flash Dec.'25	**68.2**	**90.5**
Nano Banana 2	Gemini 3.1 Flash Image Feb.'26	56.3	87.8
GPT-Image-2	OpenAI native T2I Apr.'26	50.3	85.2

4.4. Backbone Portability

The main study measures the engine-interaction effect under a fixed backbone (Gemini-3-Flash). Table 4 probes portability across providers under the same prompt and ToolSpec harness: Claude / GPT deltas are positive on this N = 112 subset, supporting interface portability without establishing broad cross-model generalization (Appendix E.3; Appendix G.3).

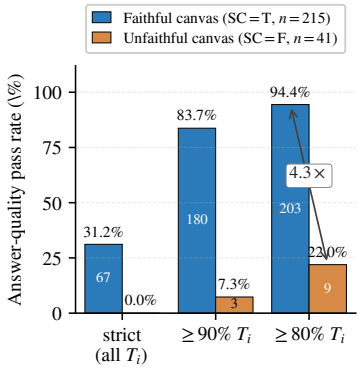
4.5. Formulation Transfer to 3D and Rendering

Solid geometry. The PDV loop transfers to 3D by adding 21 solid-geometry ToolSpecs, without changing the prompt structure or interaction protocol. Gains hold on *MathVerse-solid* (+5.8%) and *SolidGeo-hard* Level 3 (+16.4%), with thinking tokens decreasing in both cases (Table 2).

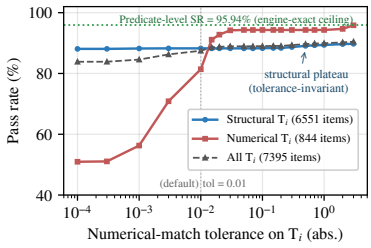
Diagram rendering. On *GenExam-math* (Wang et al., 2026), where leading closed-source text-to-image (T2I) systems struggle, Draw2Think attains 68.2/90.5 strict/relaxed accuracy, improving strict score by 11.9 points over the leading T2I generation system (Appendix F.1). The gain traces to

Table 3. Construction fidelity on GeoGoal (N=256, 13,254 predicates). **SR**: predicate-level pass rate; **SC**: strict problem-level pass rate; **CR**: non-empty canvas rate.

Metric	Rate%		
SR overall	**95.94**		
by tier	*premise* 96.85	*num-check* 97.33	*derived* 94.78
SC	**84.0**		
CR	100.0		



(a) Fidelity-conditional answer quality.



(b) Tolerance sweep on T_i match.

Figure 3. **(a)** Faithful canvases (n=215) dominate unfaithful (n=41) at three T_i-match thresholds. **(b)** Structural T_i plateau at $\sim$88% (tolerance-invariant); numerical T_i climb to the engine-exact SR (dotted) as tolerance approaches $\sim$0.1% rel.

the externalized constraint engine: stated geometric relations are enforced algebraically instead of approximated in pixels.

5. Analysis: Mechanisms of Grounded Interaction

The experiments isolate three mechanisms: grounding reshapes the model's reasoning trace under specific bottlenecks (§5.1), verified readout offers an explicit extraction channel for engine-grounded quantities (§5.2), and the remaining failures are model-level errors in perception and strategy (§5.3).

5.1. Selective Effects of Grounding

CT effects are bottleneck-dependent: gains concentrate when the missing resource is executable state, including vision-heavy measurement, auxiliary construction, and hard solid geometry, and can reverse on saturated tasks where strong internal priors already supply a low-cost route (Table 2). That bottleneck profile is consistent with cognitive offloading (Wang et al., 2025a; Qian et al., 2025): tool calls replace fragile spatial simulation when simulation is expensive and add overhead when external state is unnecessary.

On *GeoSketch* and *OlympiadBench*, CT cuts thinking tokens by 36% and 14% respectively while raising accuracy (Table 2). In these regimes, the engine substitutes for internal simulation precisely where simulation is most expensive. Removing query tools reverses this effect, reducing accuracy despite shorter context (§5.2). Per-problem Save/Break transitions reveal the same asymmetry: CT rescues many more BL failures than it breaks on vision-heavy or construction-heavy benchmarks, while regressions concentrate where BL already has a low-cost internal route (Table 2; Appendix E.4).

Table 6. Graduated ablations on matched subsets. Query ablation (N=379) removes measurement (meas.) tools or all query tools; description ablation (N=396) uses bare-signature ToolSpecs. Fail = tool error rate; Bypass = tool-using problems whose final answers lack an explicit readout anchor.

Type	Setting	Acc.	Fail	Bypass
Query	Full CT	96.3	5.4%	7
	w/o meas.	92.1 ↓4.2	7.6%	256
	w/o query	91.3 ↓5.0	8.9%	286
Desc.	Full CT	95.2	5.7%	7
	Bare signature	93.4 ↓1.8	8.3%	32

Graded query ablation answer-path migration (%); ablation intensity ↑ from left → right.

Answer path	Full CT	w/o meas.	w/o all query
Query oracle	44.6	0.3	1.1
Hybrid derived	31.9	16.9	10.3
Resilient fallback	11.1	33.0	29.3
Construction return	1.3	43.5	52.2
Tool-domain total	98.7	94.0	93.7

5.2. Readout Mechanism: Source Distribution and Ablation

Across benchmarks, 74–91% of correct CT answers use at least one engine-returned value; pure LLM-bypass stays below 9% everywhere and below 4% on six of eight benchmarks (Appendix E). Table 6 isolates the interface channels. Removing measurement tools drops accuracy by 4.2 points and drives bypass from 7 to 256 problems: once verified readout is unavailable, more final answers lose an explicit engine anchor. The lower inset shows the corresponding migration: explicit query-oracle answers nearly vanish, while most answers remain inside the tool domain through resilient fallback and construction-return paths.

Stripping ToolSpec descriptions to bare signatures preserves

the invocation interface but degrades tool selection; bypass rises from 7 to 32 because more final answers lack an explicit readout anchor despite preceding tool use. The largest loss occurs on vision-only inputs, where text provides little cue for selecting among construction tools (Appendix E.6.3). This description ablation identifies the ToolSpec surface as a behavioral lever. A per-parameter micro-ablation further shows that non-mutating ToolSpec overlays can support source-preserving harness refinement (Appendix E.6.4), consistent with concurrent *Agentic Harness Engineering* findings that harness components beyond prompt wording can drive measured gains (Lin et al., 2026).

5.3. Residual Errors Rooted in Perception and Strategy

Per-step engine verdicts already sit near the ceiling: construction fidelity holds at 95.94% predicate-level, and outcome accuracy exceeds 90% on seven of ten benchmarks. Residual failures therefore arise outside the per-action verifier. Some precede the loop: when text and diagram are ambiguous or inconsistent, Measurement Faithfulness can only certify the canvas that the model has chosen to build.

The larger residual is strategic. The engine certifies each action, while the plan threading those actions together lies outside its scope. This step-to-strategy gap accounts for the harder derived predicates (§4.3), long-chain failures (Appendix G.1), the moderate-complexity sweet spot (Table 2), and the regression on the saturated cluster (*Geo3K*, *PGPS9K*) under forced tool use (Wang et al., 2025a; Qian et al., 2025; Chen et al., 2026b). Both benchmarks reach a high BL ceiling and are linked by training-data overlap: 78 of 1000 *PGPS9K*-test items are direct *Geo3K*-test problems. Their per-problem transitions cluster at the same low Win ratio (*Geo3K*: 6 saves vs. 23 breaks; *PGPS9K*: 13 vs. 53; Appendix E.4), a regime where the baseline already has a cheap internal route and extra construction mainly introduces avoidable execution overhead. The next lever is therefore policy-level: deciding when engine invocation is warranted (§6).

6. Conclusion

Draw2Think makes VLM geometry solving more accountable by moving intermediate geometric claims onto a constraint-first GeoGebra canvas. The experiments show that this external state gives measurements engine-level faithfulness while leaving construction fidelity auditable: the construction harness improves outcome accuracy most when visual measurement or spatial construction is the bottleneck, passes **95.9%** predicate-level construction checks, and attains **68.2/90.5** strict/relaxed rendering accuracy on *GenExam-math*. The same evidence also bounds the claim: grounding adds overhead when the model has an efficient internal route, whereas engine faithfulness applies to the model-selected canvas, with semantic alignment to the target configuration left to construction fidelity.

Limitations. The engine certifies individual actions, leaving trajectory-level strategy to the model; failures therefore concentrate in ambiguous perception, long construction chains, and cases where tool use is unnecessary. The current framework targets single numeric Euclidean instances and assumes sufficient tool-use competence.

Future Work. Toward strategy-level verification. The next harness layer should make the strategy easier to maintain: expose the construction-DAG topology so the planner can see dependencies, progressively disclose ToolSpec groups to reduce the active action space, and return symbolic query forms that later calls can reuse.

Richer intermediate state and proximal twins. The present observation channel is deliberately narrow: typed objects, object deltas, and exact scalar readouts. Rendered feedback could expose layout and region-selection errors, while DAG summaries could make long traces easier to assimilate. Beyond geometry, a workflow can support the same loop once it establishes a *proximal twin*: an efficient executable surrogate that rejects invalid actions and returns local state changes. Such a twin need not model the full deployment environment; it only needs to make intermediate actions checkable.

Trajectories as reusable process records. A verified construction trajectory records more process structure than textual or visual chain-of-thought transcripts (Wei et al., 2022; Shao et al., 2024a; Google DeepMind, 2026; Lu et al., 2026): every accepted step carries typed dependencies, an engine verdict, and a concrete canvas effect. This record preserves per-action ground truth, exposes tool-selection and orchestration statistics, and supports controlled problem generation by replaying or perturbing verified subgraphs. These trajectories can also close the loop back into training (Zhang et al., 2026): strong general-purpose models can run the harness at scale, retain engine-verified traces, and use the resulting process labels to adapt smaller domain models. Draw2Think points to a broader design principle for tool-mediated agents: reliable task delivery is governed by which state is externalized, which operations are typed, which verdicts are exact, and which traces remain reusable.

Impact Statement

This paper presents work whose goal is to advance vision-language reasoning and agentic system design. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Anthropic. Claude Sonnet 4.6 system card, 2026. URL https://anthropic.com/claude-sonnet-4-6-system-card/.

Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., Lin, J., et al. Qwen2.5-VL technical report. *arXiv preprint:2502.13923*, 2025.

Botana, F., Hohenwarter, M., Janičić, P., Kovács, Z., Petrović, I., Recio, T., and Weitzhofer, S. Automated theorem proving in GeoGebra: Current achievements. *Journal of Automated Reasoning*, 55(4):339–360, 2015.

Chen, J., Tang, J., Qin, J., Liang, X., Liu, L., Xing, E., and Lin, L. GeoQA: A geometric question answering benchmark towards multimodal numerical reasoning. In *ACL-IJCNLP 2021*, pp. 513–523, 2021. URL https://aclanthology.org/2021.findings-acl.46/.

Chen, J., Li, T., Qin, J., Lu, P., Lin, L., Chen, C., and Liang, X. UniGeo: Unifying geometry logical reasoning via reformulating mathematical expression. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3313–3323, 2022. URL https://aclanthology.org/2022.emnlp-main.218/.

Chen, J., Fu, D., Xu, S., Chen, J., Feng, Y., Yang, Y., Yan, J., Zha, H., and Xia, R. Milestones over outcome: Unlocking geometric reasoning with sub-goal verifiable reward. *arXiv preprint:2601.05073*, 2026a.

Chen, Y., Dong, G., and Dou, Z. Toward effective tool-integrated reasoning via self-evolved preference learning. In *ICLR*, 2026b. URL https://openreview.net/forum?id=mNeitRAdWV.

Chervonyi, Y., Trinh, T. H., et al. Gold-medalist performance in solving olympiad geometry with AlphaGeometry2. *Journal of Machine Learning Research*, 26 (241):1–39, 2025. URL https://www.jmlr.org/papers/volume26/25-1654/25-1654.pdf.

Dong, G., Chen, Y., Li, X., Jin, J., Qian, H., Zhu, Y., Mao, H., Zhou, G., Dou, Z., and Wen, J.-R. Tool-star: Empowering LLM-brained multi-tool reasoner via reinforcement learning. *arXiv preprint:2505.16410*, 2025.

Fu, D., Chen, J., Xia, R., Chen, Z., Liu, Q., Feng, Y., Zhou, H., Zhang, R., Feng, S., Gao, P., Zha, H., Yan, J., Shi, B., Qiao, Y., and Zhang, B. Trustgeogen: Formal-verified data engine for trustworthy multi-modal geometric problem solving. *arXiv preprint:2504.15780*, 2026.

Fu, Y., Zhu, J., Zhang, L., Wu, W., Zhao, B., Ma, S., Zhang, Y., and Liu, J. GeoLaux: A benchmark for evaluating MLLMs' geometry performance on long-step problems requiring auxiliary lines. *arXiv preprint:2508.06226*, 2025.

Google DeepMind. Gemini 2.5: Our most intelligent AI model, 2025a. URL https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/.

Google DeepMind. Gemini 3 Flash: Frontier intelligence built for speed, 2025b. URL https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/.

Google DeepMind. Welcome Gemma 4: Frontier multimodal intelligence on device, 2026. URL https://huggingface.co/blog/gemma4.

Gu, J., Hao, Y., Wang, H. W., Li, L., et al. ThinkMorph: Emergent properties in multimodal interleaved chain-of-thought reasoning. In *ICLR*, 2026. URL https://openreview.net/forum?id=mB3vxfrQZM.

Guo, S., Pang, L., Wang, X., Wang, Y., Shen, H., and Zhang, J. GeoVLMath: Enhancing geometry reasoning in vision-language models via cross-modal reward for auxiliary line creation. *arXiv preprint:2510.11020*, 2025.

He, C., Luo, R., Bai, Y., Hu, S., Thai, Z., Shen, J., Hu, J., Han, X., Huang, Y., Zhang, Y., Liu, J., Qi, L., Liu, Z., and Sun, M. OlympiadBench: A challenging benchmark for promoting AGI with Olympiad-level bilingual multimodal scientific problems. In *ACL 2024*, pp. 3828–3850, 2024. URL https://aclanthology.org/2024.acl-long.211/.

Hohenwarter, M. and Hohenwarter, J. Geogebra. https://www.geogebra.org. Dynamic Mathematics Software.

Hu, Y., Shi, W., et al. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *NeurIPS*, pp. 139348–139379, 2024.

Jiao, Z., Wang, S., Zhang, Z., et al. Socratic-geo: Synthetic data generation and geometric reasoning via multi-agent interaction. *arXiv preprint:2602.03414*, 2026.

Kim, J., Yang, R., and Zhang, H. GeoBuildBench: A benchmark for interactive and executable geometry construction from natural language. *arXiv preprint:2605.13167*, 2026.

Kovács, Z. and Parisse, B. Giac and GeoGebra – improved Gröbner basis computations. In *Computer Algebra and Polynomials*, volume 8942 of *LNCS*, pp. 126–138. Springer, 2015.

Li, Y., Yi, M., Li, X., Fan, J., Jiang, F., Chen, B., Li, P., Song, J., and Zhang, T. Reasoning and tool-use compete in agentic RL: From quantifying interference to disentangled tuning. *arXiv preprint:2602.00994*, 2026a.

Li, Z., Zhang, H., et al. In-the-flow agentic system optimization for effective planning and tool use. In *ICLR*, 2026b. URL https://openreview.net/forum?id=Mf5AleTUVK. Oral.

Lin, H. and Xu, Z. Understanding tool-integrated reasoning. *arXiv preprint:2508.19201*, 2025.

Lin, J., Liu, S., Pan, C., et al. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint:2604.25850*, 2026.

Lou, X., Lázaro-Gredilla, M., Dedieu, A., Wendelken, C., Lehrach, W., and Murphy, K. P. AutoHarness: improving LLM agents by automatically synthesizing a code harness. *arXiv preprint:2603.03329*, 2026.

Lu, P., Gong, R., Jiang, S., et al. Inter-GPS: Interpretable geometry problem solving with formal language and symbolic reasoning. In *ACL*, pp. 6774–6786, 2021. URL https://aclanthology.org/2021.acl-long.528/.

Lu, P., Bansal, H., Xia, T., Liu, J., Li, C., Hajishirzi, H., Cheng, H., Chang, K.-W., Galley, M., and Gao, J. MathVista: Evaluating mathematical reasoning of foundation models in visual contexts. In *ICLR*, 2024. URL https://openreview.net/forum?id=KUNzEQMWU7. Oral.

Lu, R., Ma, Y., Chen, X., Luo, L., Wu, Z., Pan, Z., Liu, X., et al. Thinking with visual primitives. 2026. DeepSeek-AI.

Luo, R., Shi, C., Zhang, Y., et al. From narrow to panoramic vision: Attention-guided cold-start reshapes multimodal reasoning. In *ICLR*, 2026a. URL https://openreview.net/forum?id=4tsfY0lI1w.

Luo, S., Zhu, Z., Yuan, Y., Yang, Y., Shan, L., and Wu, Y. Geogrambench: Benchmarking the geometric program reasoning in modern llms. In *ICLR*, 2026b. URL https://openreview.net/forum?id=MrJoBgN1VO.

Ma, J., Wang, W., and Jin, Q. A survey of deep learning for geometry problem solving. *arXiv preprint:2507.11936*, 2025.

OpenAI. GPT-4o system card, 2024. URL https://openai.com/index/gpt-4o-system-card/.

OpenAI. Introducing o3 and o4-mini, 2025. URL https://openai.com/index/introducing-o3-and-o4-mini/.

OpenAI. GPT-5.4 Thinking system card, 2026. URL https://openai.com/index/gpt-5-4-thinking-system-card/.

Pan, Y., Zhang, Z., Hu, P., Ma, J., Du, J., Zhang, J., Liu, Q., Gao, J., and Ma, F. Enhancing the geometric problem-solving ability of multimodal LLMs via symbolic-neural integration. *arXiv preprint:2504.12773*, 2025.

Peng, S., Fu, D., Liang, Y., Gao, L., and Tang, Z. Geo-DRL: A self-learning framework for geometry problem solving using reinforcement learning in deductive reasoning. In *ACL 2023*, pp. 13468–13480, 2023. URL https://aclanthology.org/2023.findings-acl.850/.

Ping, B., Luo, M., Dang, Z., Wang, C., and Jia, C. Auto-GPS: Automated geometry problem solving via multimodal formalization and deductive reasoning. In *ICLR*, 2026. URL https://openreview.net/forum?id=PVtZnUh04m.

Qian, C., Acikgoz, E. C., et al. SMART: Self-aware agent for tool overuse mitigation. In *ACL 2025*, pp. 4604–4621, 2025. URL https://aclanthology.org/2025.findings-acl.239/.

Qiao, R., Tan, Q., et al. We-math 2.0: A versatile mathbook system for incentivizing visual mathematical reasoning. In *ICLR*, 2026. URL https://openreview.net/forum?id=I7fTPLT8A9.

Schick, T. et al. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, pp. 68539–68551, 2023. URL https://openreview.net/forum?id=Yacmpz84TH. Oral.

Shao, H., Qian, S., Xiao, H., Song, G., Zong, Z., Wang, L., Liu, Y., and Li, H. Visual CoT: Advancing multimodal language models with a comprehensive dataset and benchmark for chain-of-thought reasoning. In *The Thirty-Eighth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024a. URL https://openreview.net/forum?id=aXeiCbMFFJ.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint:2402.03300*, 2024b.

Shi, W., Yu, A., Fang, R., Ren, H., Wang, K., Zhou, A., Tian, C., Fu, X., Hu, Y., Lu, Z., Huang, L., Liu, S., Liu, R., and Li, H. MathCanvas: Intrinsic visual chain-of-thought for multimodal mathematical reasoning. *arXiv preprint:2510.14958*, 2025.

Sicca, V., Xia, T., Fédérico, M., Gorinski, P. J., Frieder, S., and Jui, S. Newclid: A user-friendly replacement for AlphaGeometry. *arXiv preprint:2411.11938*, 2024.

Sun, L., Zhu, Y., Liu, R., et al. Canvas-of-thought: Grounding reasoning via mutable structured states. *arXiv preprint:2602.10494*, 2026a.

Sun, Y., Zhang, S., et al. Math blind: Failures in diagram understanding undermine reasoning in MLLMs. In *ICLR*, 2026b. URL https://openreview.net/forum?id=RtvmTxdQV9.

Wang, H., Qian, C., Zhong, W., Chen, X., Qiu, J., Huang, S., Jin, B., Wang, M., Wong, K.-F., and Ji, H. Acting less is reasoning more! teaching model to act efficiently. *arXiv preprint:2504.14870*, 2025a.

Wang, P., Yang, C., et al. SOLIDGEO: Measuring multimodal spatial math reasoning in solid geometry. *arXiv preprint:2505.21177*, 2025b.

Wang, Y., Wang, Y., Wang, D., Peng, Z., Guo, Q., Tao, D., and Wang, J. GeometryZero: Advancing geometry solving via group contrastive policy optimization. *arXiv preprint:2506.07160*, 2025c.

Wang, Z., Yin, P., et al. GenExam: A multidisciplinary text-to-image exam. In *ICML*, 2026.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V., and Zhou, D. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, volume 35, pp. 24824–24837, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

Wei, J., Jia, C., Bai, X., Xu, X., Li, S., Sun, L., Yu, B., He, C., Wu, L., and Tan, C. GGBench: A geometric generative reasoning benchmark for unified multimodal models. *arXiv preprint:2511.11134*, 2025.

Wei, L., Peng, X., Ou, J., and Wang, B. Think-augmented function calling: Improving LLM parameter accuracy through embedded reasoning. *arXiv preprint:2601.18282*, 2026.

Weng, S., Wang, Z., Zhou, Y., Lu, R., Liu, T., Teng, Z., Liu, X., and Liu, H. GeoSketch: A neural-symbolic approach to geometric multimodal reasoning with auxiliary line construction and affine transformation. *arXiv preprint:2509.22460*, 2025.

Wu, W., Wang, Z.-K., Ye, J., Zhou, Z., Li, Y.-F., and Guo, L.-Z. NesyGeo: A neuro-symbolic framework for multimodal geometric reasoning data generation. In *2nd AI for Math Workshop @ ICML 2025*, 2025a. arXiv preprint:2505.17121.

Wu, W., Zhang, L., Zhao, B., Huang, M., Wang, Q., and Liu, J. Causal-R: A causal-reasoning geometry problem solver for optimized solution exploration. In *NeurIPS*, 2025b. URL https://openreview.net/forum?id=eRgYGhFRgZ.

Wu, Z., Long, Y., Li, J., and Huang, H. Geo-code: A code framework for reverse code generation from geometric images based on two-stage multi-agent evolution. *arXiv preprint:2602.07749*, 2026.

Xu, L., Zhao, Y., Wang, J., Wang, Y., Pi, B., Wang, C., Zhang, M., Gu, J., Li, X., Zhu, X., Song, J., and Zheng, B. GeoSense: Evaluating identification and application of geometric principles in multimodal reasoning. *arXiv preprint:2504.12597*, 2025a.

Xu, N., Jiang, Y., et al. Learning how to use tools, not just when: Pattern-aware tool-integrated reasoning. *arXiv preprint:2509.23292*, 2025b.

Zhang, C., Ning, M., et al. Geosdf: Plane geometry diagram synthesis via signed distance field. *arXiv preprint:2506.13492*, 2025.

Zhang, M.-L., Li, Z.-Z., Yin, F., Lin, L., and Liu, C.-L. Fuse, reason and verify: Geometry problem solving with parsed clauses from diagram. *arXiv preprint:2407.07327*, 2024a.

Zhang, R., Jiang, D., Zhang, Y., et al. MathVerse: Does your multi-modal LLM truly see the diagrams in math problems? In *ECCV*, 2024b. arXiv preprint:2403.14624.

Zhang, X., Jia, C., Li, S., He, D., Xiong, X., et al. How RL unlocks the aha moment in geometric interleaved reasoning. *arXiv preprint:2603.01070*, 2026.

Zhao, H., Shen, J., Zhang, Y., et al. Achieving olympia-level geometry large language model agent via complexity boosting reinforcement learning. In *ICLR*, 2026. URL https://openreview.net/forum?id=1sffPGGQyT.

Zhao, J., Zhang, T., Sun, J., Tian, M., and Huang, H. Pi-GPS: Enhancing geometry problem solving by unleashing the power of diagrammatic information. In *ICCV 2025*, pp. 1526–1536, 2025a.

Zhao, Y., Wang, X., Liu, J., King, I., and Huang, Z. Towards geometry problem solving in the large model era: A survey. In *2nd AI for Math Workshop @ ICML 2025*, 2025b. arXiv preprint:2506.02690.

Zhou, C., Chai, H., Chen, W., Guo, Z., Shan, R., Song, Y., Xu, T., Yang, Y., Yu, A., Zhang, W., Zheng, C., Zhu, J., et al. Externalization in LLM agents: A unified review of memory, skills, protocols and harness engineering. *arXiv preprint:2604.08224*, 2026.

A. Positioning in the Broader Agentic Landscape

The Propose-Draw-Verify loop in Draw2Think instantiates a broader agent-engine pattern: a frontier agentic model selects actions, an external engine executes them and returns feedback, and the model refines its plan accordingly. Similar action-execution-feedback loops appear across scientific computing and general agentic reasoning. Table A.1 anchors the geometry comparison; the following notes position Draw2Think relative to post-generation verification, modular agents, harness engineering, tool-use decomposition, and training-time adaptation.

Table A.1. Representative geometry systems by verification channel; ✓/○/× = full / partial / absent support at inference. DOM = document object model. Verdicts apply to each system's inference-time loop, not its training pipeline.

System	Substrate	Step feedback	Invalid signal	Action exact	Value readout	Revisable	No ref. image
AlphaGeo2 (Chervonyi et al., 2025)	Formal language + prover	○	○	○	×	○	✓
InternGeo (Zhao et al., 2026)	Interactive prover	✓	✓	○	×	○	✓
GeoVLMath (Guo et al., 2025)	Aux-line text	×	×	×	×	×	✓
Socratic-Geo (Jiao et al., 2026)	Synthetic code/data	×	×	×	×	×	✓
GeoCode (Wu et al., 2026)	Image-to-code	×	○	×	×	○	×
Visual Sketchpad (Hu et al., 2024)	Python sketch	○	○	×	×	○	✓
GeoSketch (Weng et al., 2025)	Logic form + render	○	○	×	×	○	✓
Canvas-CoT (Sun et al., 2026a)	DOM + render	✓	○	×	×	✓	×
Draw2Think	**Constraint canvas**	✓	✓	✓	✓	✓	✓

Notes on Table A.1 verdicts. *Action exact* requires per-action engine certification: formal-language provers (Chervonyi et al., 2025; Zhao et al., 2026) verify proof chains rather than each LLM-proposed primitive, and so receive ○. *Step feedback* requires a substrate-side signal; VLM re-perception of a rendered sketch (Hu et al., 2024; Weng et al., 2025) is ○, while structured DOM mutations with a typed critique (Sun et al., 2026a) are ✓. *Revisable* likewise separates per-step rollback from trajectory-level exploration: AlphaGeometry-style sibling search (Chervonyi et al., 2025), long-horizon trial-and-error (Zhao et al., 2026), and re-rendering loops (Hu et al., 2024; Weng et al., 2025) all receive ○. Cross-modal reward models (Guo et al., 2025) and offline data-synthesis generators (Jiao et al., 2026) consume ground-truth diagrams during training but not during solving, so the corresponding *No ref. image* entries are ✓; Canvas-CoT's critic, by contrast, evaluates each step against a target diagram and is marked ×.

Relation to post-generation GeoGebra verification. Faire (Zhang et al., 2026) also uses GeoGebra constructions as executable objects. Its task formulation casts geometric interleaving as generation of textual reasoning and GeoGebra code, followed by post-generation verification and reinforcement-learning (RL) reward. In that setting, syntactic and geometric errors can accumulate inside a generated code block before the verifier exposes them. Draw2Think makes a narrower decomposition: the initial problem perception remains model-dependent; after that, intermediate geometric state, exact measurement, and rendering are externalized through typed ToolSpec calls to the constraint engine. This shifts the bottleneck from continuous visual reasoning to discrete action selection over a verified state space.

Single agent vs. modular agentic system. AgentFlow (Li et al., 2026b) decomposes agentic reasoning into four trained modules (planner, executor, verifier, generator) coordinated through shared memory, and trains only the planner via on-policy RL (Flow-GRPO, a group-relative policy-optimization variant). Draw2Think collapses three of those roles into a single deterministic engine: GeoGebra runs the typed construction (*executor*), returns exact algebraic values or errors (*verifier*), and emits the resulting object deltas, scalar readouts, and rendered canvases (*generator*), leaving planning, tool selection, and parameter binding to a frozen LLM. Verification *and* generation are delegated rather than learned.

Curated vs. auto-discovered harnesses. A parallel line treats the harness itself as the optimization target, consistent with broader accounts of externalized memory, skills, protocols, and harness engineering in LLM agents (Zhou et al., 2026). AutoHarness (Lou et al., 2026) searches over code harnesses, while AHE (Lin et al., 2026) reports component-level evidence that tools, middleware, and memory can matter more than prompt wording alone. Draw2Think occupies a constraint-domain-specific point on this spectrum. The substrate is a pre-existing engine with a symbolic verifier, while the ToolSpec layer (79 planar-solving tools, plus 21 solid-geometry extensions and 13 rendering tools when needed) was bootstrapped from GeoGebra's official command manual and then curated across roughly eight rounds in response to pilot failures. The verifier remained symbolic throughout; the ToolSpec surface owes its shape to LLM-authored drafts plus error-driven curation. These regimes are complementary: auto-discovery scales where harness construction is cheap, whereas substrate-fixed designs can deliver deterministic correctness guarantees when the environment is already a constraint

engine. In this geometry setting, surface curation around a fixed verifier lets a frozen frontier model close gaps otherwise addressed by retraining.

Decoupling reasoning, tool search/selection, and tool execution. Recent work decomposes reasoning, tool selection, and execution at different granularities. AgentFlow (Li et al., 2026b) separates tool selection (planner) from execution (executor). Pattern-Aware TIR (Xu et al., 2025b) identifies two distinct tool-use strategies, the *calculator-pattern* (text reasoning plus spot computation) and the *algorithmic-pattern* (encoding the full problem as a program), and shows that strategy–problem mismatch is the primary cause of tool-use errors. Draw2Think's ToolSpec descriptions perform an analogous role at prompt time: query tools (e.g., `Angle`, `Distance`) correspond to the calculator pattern, while construction tools (e.g., `Point`, `Line`, `Circle`) correspond to the algorithmic pattern. The description ablation in §5 supports this view: removing semantic guidance degrades both valid tool routing and readout anchoring.

Verification fidelity and token efficiency. Systems vary in how they verify intermediate state, forming a fidelity gradient: no feedback (open-loop code generation (Wei et al., 2025)), block-level DSL execution with numerical checks (Kim et al., 2026), binary outcome reward (Li et al., 2026b; Shao et al., 2024b), approximate visual re-perception (Weng et al., 2025; Sun et al., 2026a), and exact continuous measurement (Draw2Think). As argued by Lin & Xu (2025), tool calls compress token-expensive manual simulation into structured I/O. In geometry, an `Angle(A,B,C)` query replaces coordinate extraction, dot-product computation, and inverse-trigonometric readout in one verified step. Human-readable ToolSpec names (e.g., `query_angle`, `add_perpendicular_line`) keep that compressed trace interpretable: each token-saving call remains a named geometric operation.

From inference-time to training-time. Draw2Think operates training-free: capability improvements come from better base models and prompt design rather than task-specific fine-tuning. This avoids the reasoning–tool-use gradient tension identified by DART (Li et al., 2026a), while leaving further gains to training-time adaptation. Recent training approaches already broadcast trajectory-level reward to all turns (Li et al., 2026b) or align tool-use strategy with problem structure (Xu et al., 2025b). Every Draw2Think trajectory exposes structured step-level signals (accept/reject per construction and exact values per query) that could serve as process-dense reward in an analogous *GeometryGym* environment, bridging the gap from inference-time to training-time capability.

B. Framework Details

B.1. ToolSpec Catalog

B.1.1. Catalog

The wrapper exposes three disjoint ToolSpec sets per evaluation context: 79 *solving* tools (Table B.1; 55 construction + 24 query, all planar benchmarks); 21 solid-geometry primitives (Table B.2; activated for *MathVerse-solid*, *SolidGeo*, GenExam-Solid); and 13 rendering tools (Table B.3; restricted to the GenExam rendering pipeline). Each ToolSpec carries a description, typed parameters, and preconditions — the catalog tables use summaries; full descriptions appear in Table B.5.

Provenance and licensing. Every ToolSpec wraps a GeoGebra command from the official manual[1]; the manual's six command families form a much larger pool from which ours is a curated subset. The current surface was drafted with LLM assistance, then audited against pilot-run failures for per-call determinism. GeoGebra is open source under EUPL v1.2[2], so the stack is reproducible from open components; the wrapper will be released with the paper.

Table B.1. Solving ToolSpec catalog (79 tools: 55 construction + 24 query, exposed for all solving benchmarks). C = Construction, Q = Query, T = Transform, U = Utility.

Type	ToolSpec Name	Summary Description
Points (4)		
C	`add_point`	Place a free point at (x, y)
C	`add_point_on`	Point constrained to a curve or path

Continued on next page

[1] https://github.com/geogebra/manual/tree/main/en/modules/ROOT/pages
[2] https://github.com/geogebra/geogebra

(Table B.1 continued)

Type	ToolSpec Name	Summary Description
C	add_intersect	Intersection of two objects (indexed)
C	add_midpoint	Midpoint of two points
Lines & Segments (10)		
C	add_segment	Bounded segment between two points
C	add_line	Infinite line through two points
C	add_ray	Ray from a point through another
C	add_vector	Vector from point to point
C	add_perpendicular_line	Line perpendicular to L through P
C	add_perpendicular_bisector	Perpendicular bisector of a segment
C	add_parallel_line	Line parallel to L through P
C	add_angle_bisector	Bisector of angle $\angle ABC$
C	add_tangent	Tangent line to a conic through a point
C	add_tangent_conic_conic	Common tangent of two conics
Circles & Conics (9)		
C	add_circle	Circle by center and radius/point
C	add_arc	Circular arc
C	add_sector	Circular sector
C	add_semicircle	Semicircle on a diameter
C	add_circle_3_points	Circumscribed circle through 3 points
C	add_incircle	Inscribed circle of a triangle
C	add_ellipse	Ellipse by foci and point
C	add_parabola	Parabola by focus and directrix
C	add_hyperbola	Hyperbola by foci and point
Polygons & Centers (5)		
C	add_polygon	Polygon from vertex list
C	add_regular_polygon	Regular n-gon
C	add_vertex	Extract vertex from polygon
C	add_center	Center of a circle or conic
C	add_triangle_center	Centroid, incenter, circumcenter, etc.
Measurements (4)		
C	add_angle	Construct and display an angle object
C	add_distance	Construct a distance object
C	add_area	Construct an area object
C	add_slope	Construct a slope object
Functions & Calculus (8)		
C	add_function	Define $f(x) =$ expr
C	add_derivative	Derivative of a function
C	add_integral_function	Antiderivative
C	add_inflection_point	Inflection point(s) of a function
C	add_asymptote	Asymptote(s) of a function
C	add_curve	Parametric curve
C	add_roots	Root(s) of a function
C	add_turning_point	Local extrema
Other Construction (5)		
C	add_slider	Numeric slider
C	add_best_fit_line	Linear regression line
C	add_inequality	Region defined by inequality
C	add_integral_shade	Shaded definite integral region
C	add_text	Text label on canvas
Transforms (5)		
T	transform_reflect_line	Reflect object across a line
T	transform_reflect_point	Reflect object across a point
T	transform_rotate	Rotate object by angle around center
T	transform_translate	Translate by vector
T	transform_dilate	Dilate from center by factor
Utility (4)		
U	delete_object	Cascading delete from DAG

Continued on next page

(Table B.1 continued)

Type	ToolSpec Name	Summary Description
U	set_value	Set slider/free numeric value
U	rename_object	Rename an object
U	set_label_visible / set_object_visible	Toggle visibility
Geometric Measurement (9)		
Q	query_angle	Angle measure at vertex (degrees)
Q	query_distance	Shortest distance between two objects
Q	query_length	Length of segment, arc, or vector
Q	query_perimeter	Perimeter of polygon or circle
Q	query_area	Area of polygon, circle, or sector
Q	query_slope	Slope of a line
Q	query_radius	Radius of circle/arc
Q	query_x_coord / query_y_coord	Point coordinates
Geometric Verification (8)		
Q	query_are_parallel	Test parallelism
Q	query_are_perpendicular	Test perpendicularity
Q	query_is_tangent	Test tangency
Q	query_is_in_region	Point-in-region test
Q	query_are_equal	Geometric identity test
Q	query_are_collinear	Collinearity test
Q	query_are_concyclic	Concyclicity test
Q	query_are_congruent	Congruence test
CAS & Canvas Inspection (7)		
Q	query_solve	CAS symbolic equation solver
Q	query_nsolve	CAS numeric equation solver
Q	query_definite_integral	Numeric definite integral
Q	query_function_max / query_function_min	Function extrema in interval
Q	query_is_defined	Object existence check
Q	query_dependents	List dependent objects (DAG inspection)

Table B.2. Solid-geometry ToolSpec extensions (21 tools, activated for 3D benchmarks only). C = Construction, Q = Query, R = Render.

Type	ToolSpec Name	Summary Description
3D Points & Vectors (2)		
C	add_point3d	Place a free point at 3D coordinates (x, y, z)
C	add_vector3d	3D vector from origin or between two points
Planes (4)		
C	add_plane	Infinite mathematical plane
C	add_finite_plane	Bounded rectangular plane (visual)
C	add_perpendicular_plane	Plane through point, perpendicular to line/vector
C	add_plane_bisector	Perpendicular bisector plane of two points
Solids (7)		
C	add_pyramid	Pyramid from base polygon + apex
C	add_prism	Prism from base polygon + translation
C	add_cone	Cone from base circle + apex
C	add_cylinder	Cylinder (filled solid, not lateral surface only)
C	add_sphere	Sphere by center and radius/point
C	add_tetrahedron	Regular tetrahedron
C	add_cube	Cube; three points must form a square
Derived & Auxiliary (4)		
C	add_cross_section	Intersection of plane with a solid
C	add_net	Unfolded net of a convex polyhedron
C	add_text_3d	3D-positioned text or LaTeX label
C	add_surface_revolution	Rotate function around x-axis to form surface
3D Measurement (3)		
Q	query_volume	Volume of pyramid/prism/cone/cylinder/sphere
Q	query_surface_area	Total surface area of a 3D solid

Continued on next page

(Table B.2 continued)

Type	ToolSpec Name	Summary Description
Q	`query_coords3d`	3D coordinates (x, y, z) of a point
3D Render (1)		
R	`render_set_3d_view`	Configure viewport: rotation, zoom, axes

Table B.3. Rendering ToolSpec catalog (13 tools, restricted to the GenExam rendering pipeline). R = Render. Used to style canvas objects for the rendered output; excluded from solving benchmarks.

Type	ToolSpec Name	Summary Description
R	`render_set_color`	Set object color
R	`render_set_line_style`	Set line style (solid, dashed, dotted)
R	`render_set_line_thickness`	Set line thickness
R	`render_set_point_style`	Change point marker shape
R	`render_set_point_size`	Change point marker size
R	`render_set_filling`	Set fill opacity of closed shapes
R	`render_set_decoration`	Add tick marks or arrows to segments
R	`render_show_axes`	Show or hide coordinate axes
R	`render_show_grid`	Show or hide coordinate grid
R	`render_set_caption`	Set custom label caption
R	`render_set_label_mode`	Control label display mode
R	`render_set_coord_system`	Set visible viewport bounds
R	`render_add_right_angle_mark`	Add right-angle square marker

B.1.2. Display Style Presets

Beyond the per-call `render_*` tools (Table B.3), a single preset `apply_textbook_style()` flips the canvas from GeoGebra defaults to a publication-grade textbook style in one call. Both presets share the same engine and primitives; the visual defaults differ (Table B.4).

Table B.4. Display-style presets in the wrapper. The default preset preserves GeoGebra's stock interactive appearance; the textbook preset overrides it for paper-grade rendering and is applied to all rendered galleries shown in this paper. The rightmost column flags whether the LLM can override that dimension per-object during the GenExam rendering pipeline via the corresponding `render_*` tool (Table B.3); *preset only* dimensions are set by the preset switch alone.

Dimension	Default	Textbook	LLM-adjustable
Object color	GeoGebra palette (blue points, etc.)	Pure black	per-call
Line thickness	2–3 pt	1 pt	per-call
Point marker	Blue circle, size 5	Black filled circle, size 3	per-call
Axes	Visible	Hidden	per-call
Grid	Visible	Hidden	per-call
Polygon fill	Translucent palette color	None (wireframe only)	per-call
Hidden 3D edges	Solid lines (engine default)	Dashed (textbook convention)	preset only
Background	Light gray	White	preset only
Label rendering	Upright sans-serif	Italic NAME mode	per-call

The preset is opt-in and idempotent: applying it after construction repaints existing objects and registers a listener for subsequent ones, so the same ToolSpec sequence preserves the construction graph while changing the visual policy. The textbook conventions match concurrent benchmark work (Qiao et al., 2026).

Engine state as typed XML. GeoGebra's native `.ggb` format is a ZIP wrapping `geogebra.xml`: a typed tree of `<construction>` → `<element>` / `<expression>` / `<command>` nodes (`Point`, `LineBisector`, `Intersect`, `Circle`, ...), with style attributes attached per-element. ToolSpec calls and engine observations serialize as typed records over this same graph, mirroring Anthropic's XML-tag prompt convention[3]; the textbook-style preset is its visual analogue.

[3] https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/use-xml-tags.

B.1.3. REPRESENTATIVE FULL DESCRIPTIONS

The catalog tables summarise; the model actually sees preconditions, parameter constraints, and usage guidance. We reproduce six *complete* ToolSpec descriptions verbatim as passed to the model, chosen to illustrate error-prevention guardrails, geometric-convention encoding, and the description-accuracy link validated in §3.2.

Table B.5. Six representative ToolSpecs with full descriptions. **Bold** highlights design-critical phrases. Parameters listed below each description.

ToolSpec	Full Description & Parameters
`add_intersect` *(highest failure rate)*	Create intersection point(s) of two geometric objects. Both obj1 and obj2 must be lines, segments, rays, circles, conics, or functions—**NOT points or numerics**. Omit index to get ALL intersections as a single point (best for segment-segment or line-line which have exactly one). **Use index=1 or index=2 when two objects have multiple intersections** (e.g. line-circle). The result is stored under the exact 'name' you provide—**do NOT append _1 or _2 suffixes** to access it. *Params:* `name` (string), `obj1` (string, NOT a point), `obj2` (string, NOT a point), `index` (integer, optional).
`add_perpendicular_line` *(constructive axiom)*	Create a line through a point, perpendicular to a reference line/segment/ray. The 'point' must be a point object, and 'line' must be a line, segment, or ray—**NOT a point**. **If you have two points A and B and want a perpendicular at A, first create the line: L=Line(A,B), then PerpendicularLine(A, L).** *Params:* `name` (string), `point` (string, must be point), `line` (string, must be line/segment/ray).
`add_tangent` *(strategy guidance)*	Create tangent line(s) from an external point to a conic (circle, ellipse, hyperbola, parabola). **ALWAYS use this tool for tangent lines**—do NOT manually construct tangents with add_segment or add_line. May return one or two lines stored under the exact 'name' you provide. **Do NOT append _1 or _2 suffixes**—use the name directly. For a reliable single tangent contact point, **prefer the Thales-circle construction** (Midpoint + Circle + Intersect). *Params:* `name` (string), `point` (string), `conic` (string).
`add_semicircle` *(convention encoding)*	Create a semicircle with the given segment as diameter. The arc is drawn on the **LEFT side when walking from p1 to p2**. Direction rules: (1) p1=left, p2=right (horizontal) $\rightarrow$ arc ABOVE; (2) p1=right, p2=left $\rightarrow$ arc BELOW; (3) p1=bottom, p2=top (vertical) $\rightarrow$ arc LEFT; (4) p1=top, p2=bottom $\rightarrow$ arc RIGHT. **To flip the arc, SWAP p1 and p2.** Useful for Thales' theorem: any point on the semicircle sees the diameter at 90°. *Params:* `name` (string), `p1` (string), `p2` (string).
`query_angle` *(measurement convention)*	Measure the angle at vertex b (the apex), sweeping counter-clockwise from ray $b \rightarrow a$ to ray $b \rightarrow c$. Returns degrees. **IMPORTANT: point order matters**—Angle(A,B,C) $\neq$ Angle(C,B,A). For the interior angle of triangle ABC at vertex B, **use Angle(C,B,A)** (go from one side to the other in the direction that gives the interior angle). All three must be existing point objects. *Params:* `name` (string), `a` (string, first arm), `b` (string, vertex), `c` (string, second arm).
`render_set_label_mode`	Control what is displayed as an object's label **and automatically make the label visible**. 0 = Name only (e.g. 'A'), 1 = Name + Value (e.g. 'A = (1,2)'), 2 = Value only (e.g. '(1,2)'), 3 = Caption (must set caption first). *Params:* `obj` (string), `mode` (integer: 0/1/2/3).

B.2. Construction DAG and ToolSpec Coverage

Each construction sequence induces a dependency DAG that is inspectable, prunable, and search-traversable. Table B.6 shows a 3-4-5 right-triangle build (perpendicular line + circle, then a BP=5 measurement).

Table B.6. Example construction DAG showing dependency structure. `deps` = what this step requires, `used by` = what depends on this step. Cascading delete of any node removes all downstream dependents.

Step	Tool Call	Deps	Used by	Geometric Interpretation	Semantic
1	`add_point(A, 0, 0)`	$\varnothing$	{3,4,5}	Point existence	$A = (0, 0)$
2	`add_point(B, 4, 0)`	$\varnothing$	{3,7}	Point existence	$B = (4, 0)$
3	`add_segment(AB, A, B)`	{1,2}	{4}	Two points → segment	$\|AB\| = 4$
4	`add_perp_line(L, A, AB)`	{1,3}	{6}	Perpendicular through a point	$L \perp AB$
5	`add_circle(c, A, 3)`	{1}	{6}	Center + radius → circle	$r = 3$
6	`add_intersect(P, L, c, 1)`	{4,5}	{7}	Line–circle intersection (1st)	$P = (0, -3)$
7	`query_distance(B, P)`	{2,6}	—	Measurement readout	$d = 5.0$ ✓

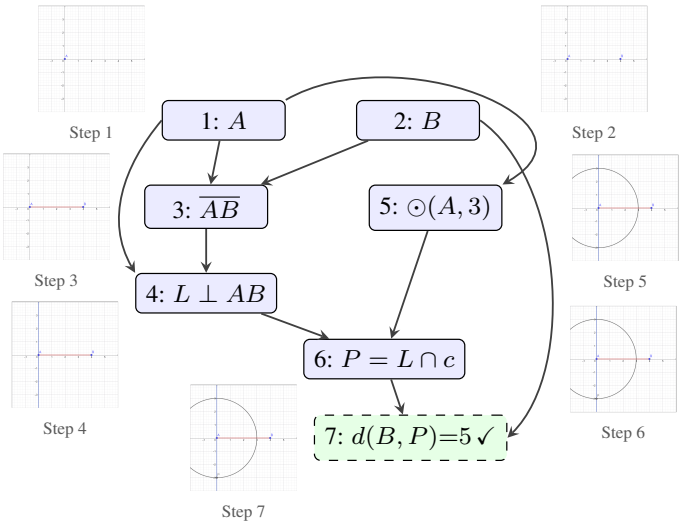


Figure B.1. Dependency DAG for the construction in Table B.6, with per-step engine-canvas thumbnails embedded around the corresponding nodes. Blue = construction step (engine-verified); dashed green = query (measurement). Each thumbnail shows the canvas immediately after the labelled step (e.g. Step 3 shows $\overline{AB}$ on top of points A, B, Step 6 adds intersection $P = (0, -3)$). Removing any node cascades to all downstream dependents.

Edge semantics. Nodes are typed construction calls (`add_*`, blue) and read-only queries (`query_*`, dashed green); directed edges encode *parent→child* dependencies, built dynamically as the LLM emits tools. Three properties hold by construction:

- *Proof-like dependency.* Step 6 requires steps 4 and 5; deleting either cascades and invalidates step 6 and below, mirroring premise retraction.
- *Instance-level verification.* Step 7 certifies $d(B, P)$=5 for *this* 3-4-5 triangle through exact engine arithmetic; universal quantification ($\forall a, b$) sits beyond the harness (§6).
- *Repair under deps.* If step 4 fails, the engine preserves {1,2,3,5}; the model replans the failed branch alone (§5.3).

Construction as graph search. Geometry solving is a construction-state search from known objects to a target state, recorded post-hoc by the DAG. The canvas is currently exposed as flat JSON; surfacing DAG topology (parent–child edges, frontier reachability) during planning is left to future work.

Construction DAGs as proof-relevant traces. Each successful tool call admits a proof-relevant geometric reading, so the DAG is a *structural analogy* of theorem search rather than functional equivalence with formal provers, giving instance-level causal justification rather than outcome-only reward. GeoLaux (Fu et al., 2025) reports up to 50% gap between answer (ACS) and process (PCS) correctness for top models; our engine guarantees step-level exactness for *this instance*, while universally quantified proofs and DAG-based proof-oriented search remain open.

C. Extended Benchmark Results

Table C.1 compares Draw2Think against the strongest previously reported results on each benchmark. Prior results use heterogeneous metrics, models, and evaluation protocols; direct comparison should be interpreted with the caveats below.

Table C.1. Cross-system comparison. All Draw2Think results are single-attempt Pass@1, temperature 0. Blank cells = not evaluated on that benchmark. Neural-symbolic methods use parsed diagram annotations rather than raw images. SFT = supervised fine-tuning; RL = reinforcement learning; GRPO = group relative policy optimization.

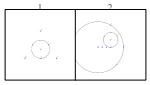
Method	**Geo3K** (Lu et al., 2021) N=601	**GeoQA/ UniGeo** (Chen et al., 2022) N=754	**PGPS9K** (Zhang et al., 2024a) N=1000	**MathVerse Vision-Only** (Zhang et al., 2024b) N=510	**MathVista GPS** (Lu et al., 2024) N=208	**Olympiad Bench** (He et al., 2024) N=112
Neural-symbolic (diagram parsing)						
InterGPS (Lu et al., 2021) formal solver	57.5					
GeoDRL (Peng et al., 2023) RL + formal solver	68.4		66.7			
PGPSNet-v2 (Zhang et al., 2024a) neural program synthesis	76.4		69.2			
Pi-GPS (Zhao et al., 2025a) PGDP + `o3-mini` solver	77.8		69.8			
Open-source VLM (zero-shot)						
Qwen2.5-VL-7B (Bai et al., 2025) reported by GeoGen (Pan et al., 2025)	44.8	64.2	43.0	44.1	54.3	
Open-source fine-tuned (SFT/RL)						
AVAR (Luo et al., 2026a) `qwen2.5-vl-7b` + RL				50.4		
SocraticGeo (Jiao et al., 2026) `qwen2.5-vl-7b` + GRPO		49.2		45.1	63.6	
NeSyGeo (Wu et al., 2025a) `qwen2.5-vl-7b` + SFT		71.8		46.7		
GeoGen-SFT (Pan et al., 2025) `qwen2.5-vl-7b` + SFT	58.4	78.0	54.3		74.0	
GeoSketch-RL (Weng et al., 2025) `qwen2.5-vl-7b` + SFT+RL	28.8	72.5				
MathCanvas (Shi et al., 2025) `bagel-7b` + SFT					79.3	
Closed-source API (zero-shot)						
GPT-4o (OpenAI, 2024) reported by TrustGeoGen (Fu et al., 2026)	31.5	42.3				13.4
Gemini-2.5-Pro (Google DeepMind, 2025a) reported by TrustGeoGen (Fu et al., 2026)	80.7	79.6				75.0
OpenAI-o3 (OpenAI, 2025) reported by TrustGeoGen (Fu et al., 2026)	81.0	79.3				77.7
Ours (training-free)						
Draw2Think BL `gemini-3-flash-preview`; direct VLM	**98.2**	93.6	**94.5**	87.5	97.1	89.3
Draw2Think CT **`gemini-3-flash-preview`; executable environment**	95.3 −2.9	**96.9** +3.3	90.5 −4.0	**91.0** +3.5	**97.6** +0.5	**89.3** +0.0
Human expert	90.9 (Lu et al., 2021)			66.7 (Zhang et al., 2024b)		

Caveats. (1) Table rows use Top-1 (single-attempt) accuracy; methods using BoN@K or non-standard test splits are excluded. (2) Our BL already exceeds most prior methods, largely due to Gemini-3-Flash's capabilities; the CT–BL gap in Table 2 isolates the framework's contribution. (3) For Geo3K and PGPS9K, Table C.1 uses Choice accuracy when prior papers report both Completion and Choice; our numbers follow the same multiple-choice protocol. Closed-source API rows reported by TrustGeoGen (Fu et al., 2026) use Completion accuracy and are included as context rather than direct format-matched comparisons. (4) Our GeoQA/UniGeo numbers use the UniGeo (Chen et al., 2022) calculation test split; prior methods report varying GeoQA protocols, so this column should be read as split-matched only where explicitly stated. (5) MathVerse human accuracy (66.7%) is from 10 college students on Vision-Only testmini (Zhang et al., 2024b).

D. Process Fidelity on GeoGoal

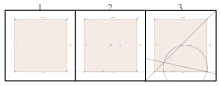
*Case geogal_00087 — 2 turns — SR=**67/67**=100% — T_i Acc.=**35/35**=100%*

Input *(problem text + reference diagram)*. In the diagram, let A, B, C be three non-collinear points with $|AB| = |BC| = |CA| = 4$ (so $\triangle ABC$ is equilateral). Let X, Y, Z be the midpoints of sides BC, AC, AB. Let I be the centre of the circle through X, Y, Z (the nine-point circle of $\triangle ABC$). Let D be on the circle centred at B with radius BA. Let E, F be the two points that trisect segment ZB (so Z, E, F, B collinear, $ZE = EF = FB$). Find $T_0 = (|AB|/|AC|)/(|BX|/|CX|)$, $T_1 = \angle(AB, AX) - \angle(AX, AC) \bmod 180^\circ, \ldots, T_{34} = \angle(AE, EI) - \angle(AI, IY) \bmod 180^\circ$ (35 lengths/ratios/angles/areas). Provide $[T_0, \ldots, T_{34}]$.



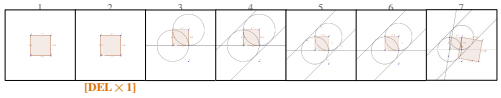
*Case geogal_00046 — 3 turns — SR=**66/66**=100% — T_i Acc.=**33/34**=97.1%*

Input *(problem text + reference diagram)*. In the diagram, a point A is chosen arbitrarily. From A a point B is constructed so that $|AB| = 14$. Points C, D are the other two vertices of the square with side AB. A point E is chosen on line AD (distinct from A, D). Let F, G, H be the midpoints of DB, EB, ED. Let I be the centre of the circle through F, G, H (nine-point centre of $\triangle EDB$). Let J be the intersection of lines DF and AI. Find $T_0 = |GB|/|GE|$, $T_1 = \angle(AB, AE) \bmod 180^\circ, \ldots, T_{33} = (AC/CD)/(AG/AI)$ (34 lengths/ratios/angles). Provide $[T_0, \ldots, T_{33}]$.



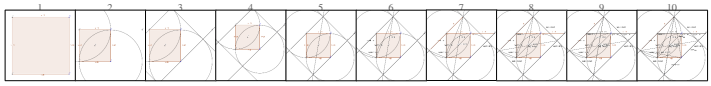
*Case geogal_00101 — 7 turns — SR=**73/73**=100% — T_i Acc.=**42/42**=100% — mid-process: 1 delete*

Input *(problem text + reference diagram)*. In the diagram, point A is chosen at random. Construct B so that $|AB| = 2$. Let C, D be the other two vertices of the square with side AB. Let E be the reflection of B across line AD. Let the circle centred at B through C and the circle centred at D through A have two common tangents; name the tangent lines FG and HI so that F, H are tangent points on circle B and G, I on circle D. Construct J, K as the other two vertices of the square with side BG. Let L be another point on line HD distinct from H, D. Determine $T_0 = |AD|/|AE|$, $T_1 = \angle(AD, DE) - \angle(DE, AE) \bmod 180^\circ, \ldots, T_{41} = \angle(AJ, EJ) - \angle(GH, GI) \bmod 180^\circ$ (42 lengths/ratios/angles/areas). Provide $[T_0, \ldots, T_{41}]$.



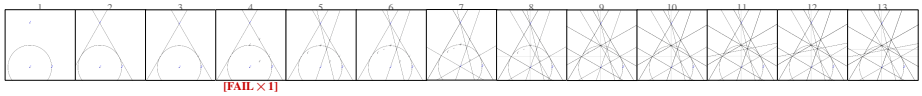
*Case geogal_00050 — 10 turns — SR=**63/63**=100% — T_i Acc.=**39/42**=92.9%*

Input *(problem text + reference diagram)*. In the diagram, A is chosen arbitrarily and B is constructed so that $|AB| = 20$. Points C, D are the other two vertices of square $ABCD$. E is constructed so that $\triangle EAC$ is an isosceles right triangle. F is constructed so that $\angle DFC = \angle EBA$. The circle centred at A through D and the circle centred at C through B have two common tangents GH and IJ, with G, I on circle A and H, J on circle C. Lines AI and FJ meet at K. Compute $T_0 = |CD|/|CH|$, $T_1 = \angle(CD, DH) - \angle(DH, CH), \ldots,$ $T_{41} = |GJ|$ (42 lengths/ratios/angles/areas). Provide $[T_0, \ldots, T_{41}]$.



*Case geogal_00006 — 13 turns — SR=**76/76**=100% — T_i Acc.=**43/47**=91.5% — mid-process: 1 fail*

Input *(problem text + reference diagram)*. In the diagram, let O be an arbitrary point. Points B, A are chosen so that $OB = 4$, $OA = 8$. Let the circle centred at O through B be drawn, and from A draw the two tangents to that circle; let X, Y be the points of tangency. Let F be the centroid of $\triangle AYO$. Let C, D, E be the midpoints of YO, AO, AY. Let G be on the perpendicular bisector of DY. Let H, I trisect segment XD (so X, H, I, D collinear in that order). Determine $T_0 = \angle(AX, OX) - \angle(AY, OY)$, $T_1 = \angle(AO, XO) + \angle(AY, XY)$, $\ldots, T_{46} = (DF/FH)/(EF/EH)$ (47 lengths/ratios/angles/areas). Provide $[T_0, \ldots, T_{46}]$.



*Case geogal_00170 — 29 turns — SR=**34/34**=100% — T_i Acc.=**17/17**=100% — mid-process: 2 fails, 7 deletes, 5 stuck*

Input *(problem text + reference diagram)*. In the diagram, seven labeled points A, B, C, X, Y, Z, I lie in the plane. Point I is to be constructed so that the ratio $|YY| : |ZI|$ equals $|ZX| : |ZI|$ (i.e., $|YY|/|ZI| = |ZX|/|ZI|$). Define $T_0 = |AC|/|BC|$, $T_1 = \angle(AB, AC) - \angle(BC, AB) \bmod 180^\circ, \ldots,$ $T_{16} = (|IY|/|YZ|)/(|IZ|/|XZ|)$ (17 lengths/ratios/angles; the configuration of A, B, C, X, Y, Z must be inferred from the targets). Provide $[T_0, \ldots, T_{16}]$.

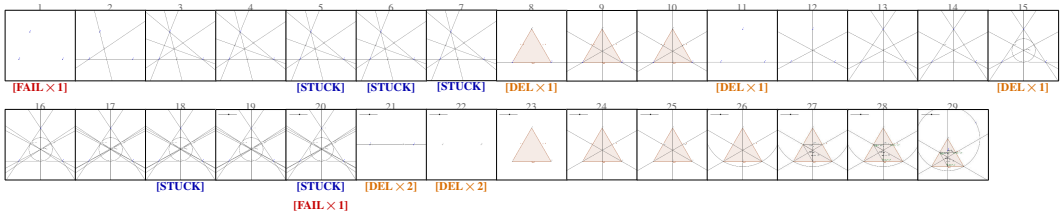


Figure D.1. Progressive canvas-construction staircase on GeoGoal. Six representative trajectories (2–29 construction turns) reach $\mathrm{SR} = 100\%$; subtitles show predicate pass counts. Badges mark engine errors **[FAIL×n]**, cascading deletions **[DEL×n]**, and query-only stuck turns **[STUCK]**.

D.1. Benchmark, Verification, and Metrics

Benchmark. GeoGoal (Chen et al., 2026a) is an externally authored, formally verified benchmark of 256 problems synthesized by TrustGeoGen (Fu et al., 2026): each provides a natural-language construction, $n_i \in [11, 66]$ query expressions, and a formal skeleton

of $m_i \in [20, 113]$ predicates over 20 types (`cong`, `coll`, `perp`, `para`, `eqangle`, `eqratio`, `midp`, `cyclic`, ...; 7,395 queries, 13,254 predicates, mean 52/problem).

Verification pipeline. On CT completion (`CONSTRUCTION_DONE` or two-turn no-tool streak), named-point coordinates from the final canvas are passed to Newclid's (Sicca et al., 2024; Chervonyi et al., 2025) `check_numerical()` (absolute tol. 4×10^{-7} or relative 0.1%); a predicate passes iff the relation holds under this tolerance.

Metrics. Under SGVR's metric names (Chen et al., 2026a), with n_i, p_i the GT and passing predicate counts for problem i:

- **CR** (Completion Rate): fraction of problems with a non-empty canvas.
- **SR** (Skeleton Rate): predicate-level pass rate $\sum_i p_i / \sum_i n_i$, decomposed into three tiers (*premise*, *numerical check*, *derived*) over SGVR's 30 source tags.
- **SC** (Skeleton Completion): problem-level all-pass rate $|\{i : p_i{=}n_i\}|/N$; a single failure disqualifies the problem.

SGVR's Consistency Ratio SC/SR is dropped in favour of reporting SR and SC directly.

D.2. Main Results and Diagnostic Cases

Figure D.1 shows canvas richness scaling with turn count (emit-only tails omitted, pixel-identical to predecessor). The 29-turn row exposes long-tail behaviours legible from traces: stuck query-only phases, cascade-pruned restarts, fallback to canonical scaffolds (`add_regular_polygon`). Figure D.2 gives the complementary mode: `geogal_00033` runs fail-free after three early rejected calls yet only reaches $\mathrm{SR} = 43\%$ because the plan omits GT-required relations, locating residual burden at policy-level construction rather than execution.

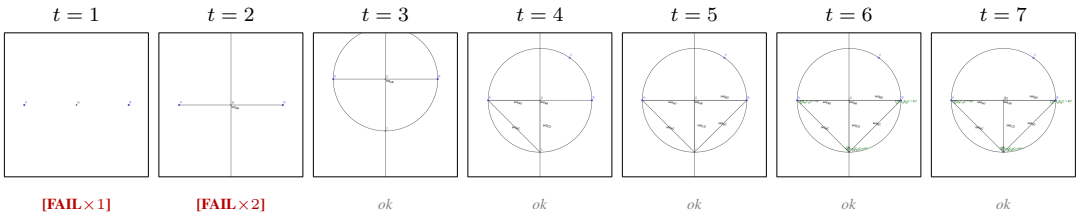


Figure D.2. Partial-failure case: `geogal_00033`, $\mathrm{SR} = 43\%$. Seven turns shown; three early failures are rejected, later turns run fail-free, and the final valid canvas still misses 57% of GT predicates.

Evaluating SGVR's `solution_FL` predicates with local Newclid (Sicca et al., 2024) gives $\mathrm{CR}{=}100.0\%$, $\mathrm{SR}{=}95.94\%$ (premise / num. / derived $= 96.85/97.33/94.78$), and $\mathrm{SC}{=}84.0\%$ (Table 3); SGVR's Gemini-2.5-Pro baseline under a gpt-5-nano LLM judge reports $\mathrm{SR}{=}88.7\%$, $\mathrm{SC}{=}44.5\%$ (Chen et al., 2026a). Under the stricter engine verifier, Draw2Think's SC rises by $\sim$40 points, showing that typed-operator construction survives exact predicate checking rather than relying on semantic lenience.

D.3. Tolerance and Residual Error Decomposition

Sweeping query-level tolerance separates structure from coordinate precision: structural targets ($T_i \in \{0, 1, 90, 180\}$, 6,551 items) hold an $\sim$88% plateau across five orders of magnitude; numerical targets (844 items) rise from 51% at 10^{-4} to 94% at 3×10^{-2}, approaching Newclid's $\sim$0.1% relative tolerance (Figure 3b). Table D.1 classifies residual failures by error scale and expression type.

Table D.1. T_i failure decomposition on GeoGoal (7,395 queries). C1 = structural-scale error; C2 = coordinate-precision error; C3 = negligible residual.

	All (N=7395)	**Faithful** (IR=T, 6216)	**Unfaithful** (IR=F, 1179)	**Dominant expr type** among failures
OK	87.48%	92.87%	59.03%	—
C1 structural	6.45%	3.06%	24.34%	angle expr (97%)
C2 precision	2.46%	1.59%	7.04%	length expr (100%)
C3 tight	0.00%	0.00%	0.00%	—
NA (undefined point)	3.61%	2.48%	9.58%	—

C1 failures in faithful canvases are mostly angle expressions (184/190) implying collinearity / parallelism beyond SGVR's explicit predicates (structural-understanding errors); C2 are all pure length queries (99/99) at $\sim$0.04% relative error, consistent with 2–3 sig.-digit coordinates from the model. The absent C3 regime locates precision loss at coordinate input, not float64 engine arithmetic. Engine-exact verification thus separates structural from precision errors that an LLM judge collapses into one sub-goal failure.

E. Mechanism Analysis

E.1. Answer Source Distribution

Figure E.1 schematises the taxonomy and Table E.1 classifies correct CT answers by answer provenance visible in the traces, separating observable anchors from internal causal attribution. **Clean Oracle**: the final answer is directly anchored to an engine-returned value. **Hybrid**: the answer combines engine-returned values with model reasoning. **Resilient**: the answer is recovered after intermediate tool failures. **LLM Bypass**: the final answer lacks a textual anchor to engine-returned values despite tool use.

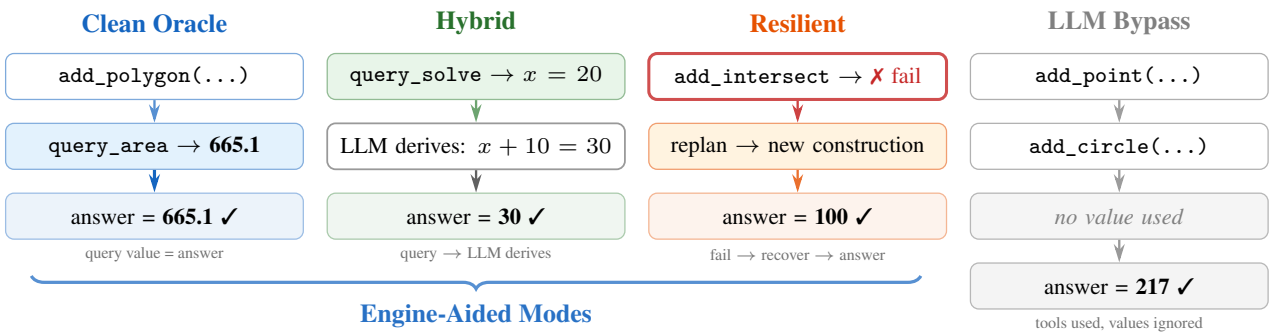


Figure E.1. Answer provenance taxonomy in Draw2Think. Clean Oracle reads an engine value; Hybrid derives from an engine-returned intermediate; Resilient recovers after a tool failure; LLM Bypass invokes tools while the final answer lacks an explicit anchor to returned values. The first three columns are grouped as engine-aided modes.

Table E.1. Answer source distribution among answered problems (CT mode). Engine-Involved counts answers whose provenance includes at least one textual anchor to an engine-returned value.

Dataset	Pass/N	Clean Oracle	Hybrid	Resilient	LLM Bypass	Engine-Involved
PGPS9K	905 / 1000	50.2	34.3	13.0	2.5	**91.5**
UniGeo	731 / 754	52.9	25.5	19.8	1.8	**91.3**
Geo3K	573 / 601	39.5	42.6	14.6	3.3	**88.0**
MathVista	204 / 208	25.5	51.0	19.6	3.9	**82.4**
MathVerse†	569 / 629	23.0	54.9	16.2	5.9	**81.6**
GeoSketch	335 / 390	19.9	56.4	21.3	2.4	**80.9**
GeoLaux	208 / 221	27.9	38.1	31.2	2.8	**77.7**
OlympiadBench	100 / 112	23.8	41.9	25.7	8.6	**74.3**

†MathVerse combines Plane (510) and Solid (119) subsets.

E.2. Process Statistics

Table E.2. Turn distribution across benchmarks (CT mode). Percentages are row-wise.

Dataset	N	1	2	3	4	5+	Mean
GeoSketch	390	5%	33%	30%	18%	14%	3.3
MathVerse Plane	510	0%	66%	22%	8%	4%	2.5
GeoLaux	221	1%	72%	20%	5%	2%	2.4
MathVista GPS	208	0%	80%	14%	4%	2%	2.3
GeoQA	754	3%	77%	15%	4%	1%	2.2
PGPS9K	1000	7%	71%	18%	4%	1%	2.2
Geo3K	601	2%	85%	10%	2%	1%	2.1
SolidGeo-hard	177	9%	26%	35%	19%	11%	3.0
MathVerse Solid	119	2%	77%	18%	3%	0%	2.2

Table E.2 reports the turn distribution across benchmarks. On the textbook-level benchmarks (*PGPS9K*, *Geo3K*, *GeoQA*/*UniGeo*, *MathVista*, *MathVerse*), 66–85% of problems complete in exactly two turns. *GeoSketch* (avg 3.3 turns) and *SolidGeo-hard* (avg 3.0 turns)

shift the distribution toward longer trajectories, reflecting auxiliary construction and 3D competition-level reasoning respectively.

Token efficiency and wall time. BL is one direct model call; CT sums all engine-mediated turns in one construction trajectory. Table E.3 shows generated tokens and wall time. CT thinking overhead shrinks as BL thinking budget rises: from $+121\%$ on *GeoQA* to -36% on *GeoSketch* and -14% on *OlympiadBench*. Output tokens often drop because measurement and computation move into the engine. The remaining overhead is mostly repeated ToolSpec/canvas exposure ($\sim$35–64K input tokens per problem), which motivates adaptive schema disclosure (§6).

Table E.3. Per-problem generated tokens and wall time (mean, excluding timeouts). Format: BL $\rightarrow$ CT $\times$ratio; bold = CT uses fewer resources. Model: `gemini-3-flash-preview@medium` (‡*OlympiadBench*: `@high`); †*MathVerse* combines Plane+Solid.

	Thinking tokens		Output tokens		Wall time (s)		
Dataset	BL	CT	BL	CT	BL	CT-tot	CT-pt
GeoQA/UniGeo	808	1,785 ×2.21	335	425 ×1.27	8.8	23.5	8.6
PGPS9K	1,043	1,566 ×1.50	365	301 **×0.82**	10.5	22.5	12.9
MathVista	1,174	1,629 ×1.39	373	399 ×1.07	10.2	18.5	6.7
Geo3K	1,485	1,878 ×1.26	374	306 **×0.82**	11.2	20.5	8.4
GeoLaux	2,301	2,872 ×1.25	439	538 ×1.23	17.0	28.4	9.7
MathVerse†	2,540	2,637 ×1.04	392	327 **×0.83**	17.7	25.6	10.3
GeoSketch	5,498	3,545 **×0.64**	648	692 ×1.07	32.8	32.5	10.1
OlympiadBench‡	5,884	5,063 **×0.86**	632	934 ×1.48	38.5	39.3	15.0

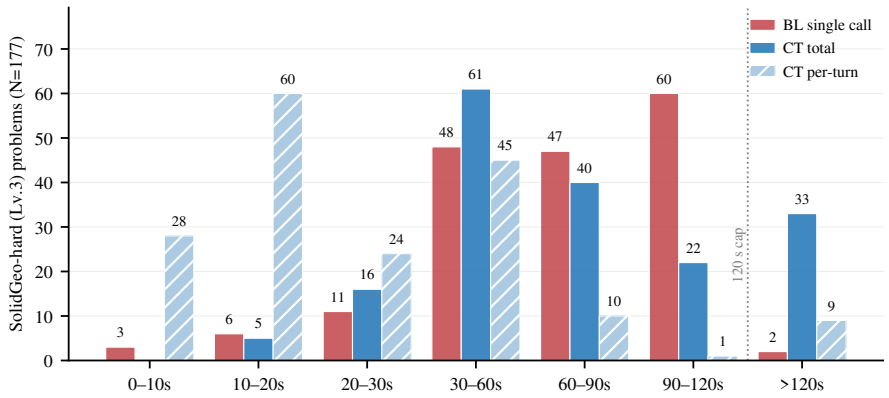


Figure E.2. Wall-time distribution for Gemini-3-Flash@high on *SolidGeo-hard* (N=177). BL (single call, hard-capped at 120 s), CT total (multi-turn sum), and CT per-turn ($t_{\text{final}}/\text{eff}_T$). Median: BL 72.5 s, CT total 63.1 s, CT per-turn 20.1 s.

Wall-time profile under the 120 s per-call cap. Figure E.2 reports *SolidGeo-hard* under the same 120 s per-call cap. CT's median total time (63.1 s) is below BL's (72.5 s); the 33 CT attempts exceeding 120 s reflect multi-turn accumulation at the problem level, while every individual turn stays within the cap.

Tool usage temporal structure. Figure E.3 groups 50,025 tool calls into Primitive Construction, Derived Construction, Transform & Utility, and Query & Verification, then plots each group's share over normalized call position $p = i/(K-1)$.

The phase split motivates progressive tool-schema disclosure. Initialization (0–25%) is primitive-heavy ($\sim$65%); readout (75–100%) is query-heavy ($\sim$80%). A staged schema could reduce the active action space during early construction while preserving the tools needed for that phase.[4]

[4]Google Gemini recommends keeping active tools to 10–20 where possible; OpenAI supports deferred loading via `tool_search`; Anthropic recommends consolidating related operations.

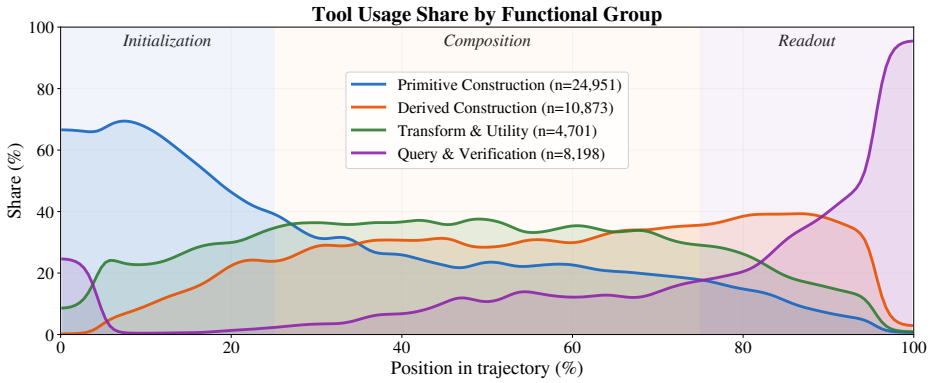


Figure E.3. Tool usage share by functional group over normalized trajectory position. Early calls concentrate on primitive construction; the final quarter concentrates on query and verification tools.

E.3. Cross-Model Validation

Table E.4 reports BL and CT accuracy on *OlympiadBench* across three frontier VLMs released in the past six months. Generated-token columns report thinking and output tokens for the matched BL and CT runs; costs are reported separately for the two modes. The token pattern is model-specific: Gemini reduces thinking under CT (5,884→5,063), whereas Claude Sonnet 4.6 and GPT-5.4 spend more generated tokens in exchange for higher accuracy. CT gains range from $+0.0\%$ to $+5.4\%$, providing a benchmark-level portability check across providers while showing that the cost–accuracy tradeoff varies with model tool-use competence.

Table E.4. Cross-model BL and CT comparison on *OlympiadBench* (N=112, Pass@1). Think, Out, and Turns are per-problem averages; CT Turns counts engine-mediated turns in the construction trajectory. Costs are for the matched BL and CT runs: Gemini is computed from logged tokens at list price, Claude from platform billing, and GPT as an upper-bound estimate.

Model	BL				CT					
	Acc	Think	Out	Cost	Acc		Think	Out	Turns /prob.	Cost
Gemini-3-Flash@high	89.3%	5,884	632	$2.3	89.3%	+0.0	5,063	934	2.7	$4.8[a]
Claude-Sonnet-4.6@high	89.3%	1,523	5,274	$9.1	92.0%	↑2.7	2,455	7,528	6.4	$26.0[b]
GPT-5.4@high	89.3%	1,123	1,437	$4.6	94.6%	↑5.4	2,206	2,842	8.2	$40.5[c]

[a]Logged-token list-price estimate; BL+CT total US$7.0; actual billing may be lower with cache.

[b]Platform bill with `cache_control`; BL+CT total US$35.1.

[c]Upper-bound estimate with `previous_response_id`; BL+CT total US$45.1.

E.4. Outcome Transition Analysis

Table E.5 decomposes accuracy into per-problem Save/Break transitions. Overall, CT rescues 133 BL failures and introduces 128 regressions (net $+5$). Saves dominate on construction-heavy or visually grounded benchmarks (*GeoQA/UniGeo*, *MathVerse*, *GeoSketch*); breaks dominate on saturated benchmarks (*PGPS9K*, *Geo3K*), consistent with the ceiling-effect interpretation in Table 2. Save cases are often no-response or high-BL-thinking failures; Break cases concentrate where BL already has a cheap internal route.

Table E.5. Per-problem outcome transition between BL and CT (same model, same problem). Sorted by win rate (Save / Break) descending.

Dataset	N	BL✓ CT✓	Save BL✗→CT✓	Break BL✓→CT✗	BL✗ CT✗	Net	Win Save/Break
GeoQA/UniGeo	754	691	40	15	8	+25	2.7
MathVerse	510	435	29	11	35	+18	2.6
GeoSketch	390	309	26	10	45	+16	2.6
MathVista	208	199	4	3	2	+1	1.3
GeoLaux	221	197	11	9	4	+2	1.2
OlympiadBench	112	96	4	4	8	+0	1.0
Geo3K	601	567	6	23	5	−17	0.3
PGPS9K	1000	892	13	53	42	−40	0.2
Total	3796	3386	133	128	149	+5	1.0

E.5. Tool Failure Taxonomy

The tool layer succeeds on ∼95% of calls. We categorize the remaining failed calls from the five textbook-level solving benchmarks by root cause; percentages in Table E.6 are over failed calls, not problems. Many affected problems still recover through construction-path diversity and state-preserving replanning (§5.3).

Table E.6. Taxonomy of intermediate tool-call failures observed during the solving process. These are *process-level* events; problems with such failures frequently still reach correct final answers via recovery (§5.3). *Layer* indicates root cause origin. Percentages are relative to total failed calls, not total problems.

Category	Layer	Share	Typical manifestation
Entity not found	Model	∼35%	References `segment_AB` before it exists (only A, B defined)
Geometric precondition	Model	∼19%	`Intersect(L1, L2)` on non-intersecting lines
Degenerate input	Model	∼5%	`Line(P, P)`: identical points to a 2-point constructor
Type mismatch	Model	∼3%	Point passed where a curve is expected
Model-layer total		**∼62%**	
Silent cascade	Interface	∼37%	Downstream call fails on undefined parent; no signal returned
Runtime error	Engine	∼1%	GeoGebra JS exception or CAS solver failure

Main failure modes. Entity-not-found errors dominate model-layer failures: the model assumes an object exists when the canvas lacks it, e.g., treating points A,B as if `segment_AB` had been created, predicting names that differ from GeoGebra's internal labels, or continuing after a failed `add_intersect`. Geometric-precondition errors are second: the object references are valid, while the requested relation is absent from the current canvas. A small subset reflects engine tolerance, such as a line–circle intersection recovered by indexed retry; most reflect unsupported assumptions about parallelism, tangency, or intersection. Degenerate-input and type errors isolate parameter generation: the model selects the right operation but binds invalid operands, as in `Line(F,F)` or `add_circle(C,)`.

Cascade and recovery. We observe 55 cascade chains, usually after a failed `add_intersect`: the model issues 2–8 dependent calls to the undefined object, then replans once the turn-level observation exposes the missing parent. Despite these cascades, 70% of affected problems still reach the correct answer because previously valid objects remain on the canvas.

Implications for Harness Design. The engineering targets are direct: existence checks for missing objects, feasibility checks before geometric constructors, and explicit diagnostics for silent cascades. Since 70% of cascade-affected problems already recover, better diagnostics mainly reduce wasted turns.

E.6. Mechanism Ablations

E.6.1. Delete Ablation: Conditional Value of Branch Retraction

We remove `delete_object` from the action space and rerun the 86 problems across the five textbook-level benchmarks that originally invoked it. Table E.7 classifies the original delete usage into four patterns; Table E.8 reports per-category ablation results.

Table E.7. Original delete usage patterns on the 86 problems. Category is determined by delete count and co-occurrence with tool failures.

Category	Characterization	N	Avg del	Avg tools
A: Light cleanup	≤2 deletes, 0 tool failures	38	1.4	16
B: Error repair	Delete after a tool failure	18	1.9	23
C: Strategy reset	3–4 deletes, no prior failure	19	3.3	21
D: Heavy rollback	≥5 deletes	11	7.9	32

Table E.8. Per-category ablation results after removing `delete_object`. Δtools is the relative change in average tool calls.

Category	N	Orig acc	Ablat acc	Δacc	Δtools
A: Light cleanup	38	95%	95%	0	−17%
B: Error repair	18	89%	94%	+1	−20%
C: Strategy reset	19	95%	89%	−1	−34%
D: Heavy rollback	11	91%	100%	+1	−46%
Overall	**86**	**93.0%**	**94.2%**	**+1.2%**	**−29%**

Key findings. *Delete has conditional value.* Removing `delete_object` slightly raises aggregate accuracy (93.0%→94.2%) and reduces tool calls in every category (17–46%), showing that many deletes are cleanup or over-exploration rather than necessary repair. Seven of 86 problems flip outcome (3 correct→wrong, 4 wrong→correct), and 63 of the 79 stable cases use fewer tools. The boundary appears in the paired cases: deletion repairs one paper-folding construction and recovers 110°, whereas removing deletion prevents a rotation-area rebuild loop and restores the correct answer. Delete is therefore an aggressive revisability primitive. It retracts canvas branches more cleanly than textual self-correction, while invalid-action rejection and state-preserving replanning account for much of the PDV loop's practical revisability.

E.6.2. Query Ablation: Answer Source and Trajectory Migration

Table 6a in the main text reports aggregate accuracy and cost changes under two query ablation levels. Here we break down the adaptation mechanism along three dimensions on the same N = 379 matched subset.

Answer source migration. Table E.9 classifies each problem's final answer by its derivation channel. Under Full CT, 44.6% of correct answers are *clean oracle*: the final value matches a query return directly. Under ablation, this channel nearly vanishes; the model compensates primarily through *construction-based derived readout* (classified as "constr. readout" below): tools such as `add_angle` and `add_distance` return exact values as a side effect of construction, and the model uses these instead of the blocked query interface.

Table E.9. Answer source distribution under query ablation (N = 379). "Constr. readout" = the model invoked construction tools but obtained values from their return fields rather than explicit query calls.

Answer source	Full CT	w/o meas	w/o query
Clean oracle	44.6%	0.3%	1.1%
Hybrid	31.9%	16.9%	10.3%
Resilient (oracle)	9.8%	0.3%	0.8%
Resilient (fallback)	11.1%	33.0%	29.3%
Constr. readout	1.3%	43.5%	52.2%

Trajectory structure migration. Table E.10 reports the average share of tool calls by functional group (matching the four groups in Figure E.3). Query-phase calls drop from 16.5% to 4.6%, while derived-construction calls rise from 21.9% to 33.2%. Readout functionality migrates from the dedicated query phase into the construction trajectory.

Table E.10. Tool-group share (% of tool calls per problem) under query ablation.

Tool group	Full CT	w/o meas	w/o query
Primitive Construction	53.4%	48.7%	49.7%
Derived Construction	21.9%	30.5%	33.2%
Transform & Utility	7.9%	8.9%	8.5%
Query & Verification	16.5%	6.8%	4.6%

Tool substitution pattern. Table E.11 shows the dominant 1:1 substitution pairs. The near-perfect mirror between blocked query tools and their derived counterparts (`query_angle` ↔ `add_angle`, etc.) indicates a systematic substitution pattern in the model's adaptation. Under **w/o meas**, `query_solve` (CAS) surges from 13 to 226 calls as an additional escape channel; under **w/o query**, inspection via `query_is_defined` rises from 0 to 295 as the model uses object-existence checks in place of value queries.

Table E.11. Tool call counts on the matched 379-problem subset. Blocked tools are marked with †.

Tool	Full CT	w/o meas	w/o query
`query_angle`†	318	59	47
`query_distance`†	206	41	30
`query_area`†	45	3	0
`add_angle`	11	288	317
`add_distance`	9	171	184
`add_area`	0	38	42
`query_solve`	13	226	29
`query_is_defined`	0	53	295

E.6.3. Description Ablation: Bare Signatures and Selection Guidance

We strip natural-language descriptions from the 50 construction and transform ToolSpecs, retaining function names and typed parameter schemas ("bare" level). Query tools and their descriptions are kept intact, so the model still has access to the same measurement interface; the ablation removes the natural-language guidance on *when and how* to construct.

Table E.12 reports per-benchmark results on the matched subset (N = 396). Three observations stand out:

Table E.12. Description ablation: bare signatures vs. full ToolSpec descriptions (N = 396, four benchmarks). Fail rate = fraction of tool calls returning an error. LLM Bypass = tool-using problems whose final answers lack an explicit anchor to a `query_*` return value.

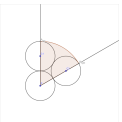
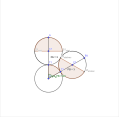
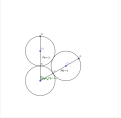
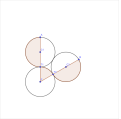
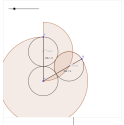
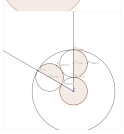
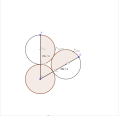
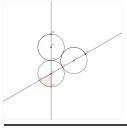
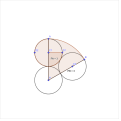
						Fail rate		Bypass	
Benchmark	**N**	**BL**	**Full**	**Bare**	**Δ**	**Full**	**Bare**	**Full**	**Bare**
UniGeo	110	93.6	97.3	97.3	0.0	5.4	5.7	2	9
PGPS9K	119	89.9	96.6	95.8	−0.8	4.3	7.1	3	11
MathVerse	67	86.6	88.1	82.1	−6.0	5.1	9.0	0	3
GeoLaux	100	95.0	96.0	94.0	−2.0	7.4	11.3	2	9
Total	396	91.7	95.2	93.4	−1.8	5.7	8.3	7	32

Interpretation. Under bare signatures, effort remains matched while routing quality drops: thinking tokens match ($\times 1.00$), turn counts and tool calls stay nearly fixed, tool failures rise from 5.7% to 8.3%, and LLM Bypass increases from 7 to 32 correct answers. Failure rate measures weakened tool/parameter selection; bypass measures weakened readout anchoring. The effect is strongest on *MathVerse* (-6.0 pp, below BL), where visual construction guidance matters most, and absent on *UniGeo*, whose textual constraints make function names sufficient.

E.6.4. Towards Automatic Harness Refinement: A Per-Parameter Micro-Ablation

Protocol. The description ablation above treats ToolSpec descriptions as a binary lever. This finer probe uses *MathVerse* 3850 to test per-parameter runtime overlays without modifying the source catalog; the listed GT is 1.0, whereas the geometric truth is 7/6, so the signal is trajectory shape and budget rather than accuracy. All nine temperature-0 variants return 7/6. The unused-tool control `v8` sets a -35% local perturbation floor. CCW direction hints exceed that floor most strongly, with `v4_sector` reducing thinking by 76% in 28 s. Combined edits are non-monotonic (`v7`: -71%; `v6`: -46%), indicating that ToolSpec wording changes the trajectory rather than composing as independent local effects.

Table E.13. Per-parameter description-refinement variants on *MathVerse* 3850. Bold marks edited ToolSpec text; CCW = counterclockwise sweep. Δ reports thinking-token change vs. `v0_source`; the unused-tool control `v8` sets a -35% perturbation floor.

Canvas	Variant	Description edit	Tools	Think	Out	Time (s)
Group A. Endpoint hint on `add_semicircle.p1/p2`.						
	`v0_source`	— (unmodified ToolSpec catalog as reference)	13	11,097	1,001	75.9
	`v1_minimal`	`p1`: "First endpoint of diameter **(on the boundary, not the centre)**"; same for `p2`	28	5,338 $_{-52\%}$	907	46.6
	`v2_verbose`	`p1`: "First endpoint of diameter**; must be a named point on the boundary, not the centre. If the endpoint is missing, create it first.**" `p2`: "Second. . . **[same rule]**"	17	7,252 $_{-35\%}$	1,150	52.0
Group B. CCW direction hint on `add_arc` / `add_sector`.						
	`v3_arc`	`add_arc`: ". . . **For the shorter arc, choose start_pt and end_pt so the CCW sweep is the smaller sweep; otherwise swap them.**"	19	4,916 $_{-56\%}$	**586**	40.2
	`v4_sector`	`add_sector`: ". . . **For the smaller sector, choose start_pt and end_pt so the CCW sweep is the smaller sweep; otherwise swap them.**"	19	**2,628** $_{-76\%}$	1,057	28.2
	`v5_both`	Both `add_arc` and `add_sector` CCW hints (= `v3` $\cup$ `v4`)	26	4,295 $_{-61\%}$	1,480	37.3
Group C. Endpoint $\cup$ CCW combined.						
	`v6_min+CCW`	`v1` $\cup$ `v3` $\cup$ `v4` (minimal endpoint + both CCW hints)	22	6,002 $_{-46\%}$	744	48.5
	`v7_verb+CCW`	`v2` $\cup$ `v3` $\cup$ `v4` (verbose endpoint + both CCW hints)	23	3,182 $_{-71\%}$	738	**28.0**
Group D. Negative control (edit on a tool the trajectory never invokes).						
	`v8_negctrl`	`add_function`: ". . . **Useful for plotting algebraic curves over a domain.**" *(unused on this trajectory)*	17	7,175 $_{-35\%}$	986	53.5

Implication for harness refinement. Runtime overlays with schema checks and a negative-control floor give a practical recipe for automatic harness refinement: enumerate candidate per-parameter edits, evaluate them against an unedited baseline and an unused-tool control, and keep effects beyond the local floor. This complements Agentic Harness Engineering (Lin et al., 2026): geometric ToolSpecs sit inside a cascaded construction DAG, so a local wording change can alter object dependencies, downstream readouts, and answer provenance.

Limitations. A single case study at temp 0, with claims read off the gap above the -35% negative-control floor; the GT bug isolates trajectory shape from accuracy but precludes any benchmark-level generalisation.

F. GenExam-Math Rendering Evaluation Details

F.1. Detailed *GenExam-Math* Results

Table F.1 places Draw2Think on the *GenExam* public leaderboard. Math is consistently the weakest subject for leading closed-source T2I systems because precise geometric relations expose the limits of pixel-first synthesis, making it a natural target for *constraint-first* rendering. Table F.2 adds our per-category and auxiliary-dimension breakdown.

Table F.1. *GenExam* Math-subset leaderboard comparison (151 examples); official leaderboard at `https://github.com/OpenGVLab/GenExam#-leaderboard`, snapshot as of April 23, 2026. T2I = text-to-image; MLLM = multimodal large language model. Base model shown in gray below each closed-source method; secondary per-vendor variants commented in the source for reference.

Closed-source Models

Method	Strict	Relaxed
Draw2Think Gemini 3 Flash Preview — Dec.2025	**68.2**	**90.5**
Nano Banana 2 Gemini 3.1 Flash Image — Feb.2026	56.3	87.8
Nano Banana Pro Gemini 3 Pro Image — Nov.2025	55.6	86.3
GPT-Image-2 OpenAI native T2I — Apr.2026	50.3	85.2
Seedream 5.0 ByteDance native T2I — Feb.2026	47.0	82.9
GPT-Image-1.5 OpenAI native T2I — Dec.2025	26.5	65.8
FLUX.2 max Black Forest Labs T2I — Dec.2025	6.6	49.1

Open-source T2I / Unified MLLMs

Method	Strict	Relaxed
Open-source T2I Models		
FLUX.2 dev	2.6	31.6
Qwen-Image-2512	0.0	27.9
HunyuanImage-3.0	0.0	17.0
Open-source Unified MLLMs		
BLIP3o-NEXT-GRPO	0.0	15.5
BAGEL	0.0	14.7
Janus-Pro	0.0	13.7
Open-source Code-rendered RL		
Faire (Zhang et al., 2026) Qwen3-VL-8B + GRPO	9.3	52.3

Table F.2. Detailed rendering results of our Draw2Think on the *GenExam* (Wang et al., 2026) Math subset (151 examples). In addition to strict and relaxed scores, we report semantic correctness, spelling, readability, and logical consistency from the official evaluator.

Category	N	Strict	Relaxed	Semantic	Spelling	Readability	Logic
Plane Geometry	84	66.7	90.6	88.4	1.98/2	1.87/2	1.90/2
Analytic Geometry	56	80.4	95.5	94.3	2.00/2	1.95/2	1.95/2
Solid Geometry (3D)	11	18.2	63.6	59.0	1.82/2	1.45/2	1.18/2
Plane + Analytic (2D)	140	72.1	92.6	90.6	1.99/2	1.90/2	1.92/2
Overall	151	68.2	90.5	88.5	1.97/2	1.87/2	1.87/2

Failure analysis (48 strict-fail problems). Strict requires *all* scoring points correct and full marks on every auxiliary dimension (spelling, readability, logic); any shortfall yields `strict_score = 0`.

Table F.3. Root causes of strict-score failures on *GenExam-math*. "Semantic" = missing scoring points; "Quality" = auxiliary dimensions (spelling, readability, logic) below threshold.

Root cause	N	Typical mechanism
Semantic-only	17	Missing scoring points (unlabeled point, omitted auxiliary); quality passes.
Both semantic + quality	14	Missing elements *plus* layout defects (label overlap, font-viewport mismatch).
Quality-only	4	All scoring points present; `read = 1` (font-to-viewport) or `logic < 2`.
Timeout (>120 s)	5	Proxy/network latency; not a construction or model failure.
Severity distribution		
Critical (relaxed < 0.6)	6	Major construction failure or missing primary geometric object.
Moderate (0.6–0.8)	12	Partial scoring point loss; core figure recognizable.
Near-pass (> 0.8)	12	1–2 auxiliary dimensions short; often `read = 1` alone.

Of 151 problems, 103 reach `strict_score = 1` (Appendix F.3); 7 of the remaining 48 keep perfect semantic but lose strict on presentation alone (mostly `readability = 1`), with recurring label overlap ("A,B,C,D" near the origin) and viewport–font mismatch.

These quality-only failures mark a phase boundary in the harness: construction turns see textual canvas state, while render turns style a committed construction. Post-construction screenshots, tighter label-placement render-tool descriptions, or finer-grained construction

actions could localise them earlier. The remaining 41 have `semantic` < 1.0, mostly through omitted secondary elements (dashed auxiliaries, arc markers) over a recognisable primary structure.

Per-taxonomy breakdown. Table F.4 reports per-taxonomy strict pass rate and average turn count split into *construct* (canvas-building) and *render* (styling) over the 37 official *GenExam-math* (Wang et al., 2026) taxonomies. Solid geometry is the clear bottleneck: 4 of 6 taxonomies fail every problem despite `thinking_level=high` on 9 of 11 attempts. Within plane / analytic, turn count separates poorly from pass rate: *Triangle/Perpendicular_Bisector* passes at 16 render turns while *Triangle/Similarity* fails at 9.5 construct turns, separating render-quality from construction-layer failure modes.

Table F.4. Per-taxonomy breakdown of *GenExam-math* (151 problems, 37 taxonomies; `Mathematics/` prefix dropped). **N**: count; **s=1**: strict_score=1; **Pass**: strict rate; **Total / T_c / T_r**: average total / construct / render turns. Top: plane + analytic (31 rows); bottom: solid (6 rows). Purple = zero strict-1; blue = 100% strict pass.

Taxonomy	N	s=1	Pass	Total	T_c	T_r
Plane_Geometry/Triangle/Similarity	2	0	0%	17.0	9.5	7.5
Plane_Geometry/Triangle/Others	1	0	0%	5.0	3.0	2.0
Plane_Geometry/Angle	3	1	33%	19.3	12.0	7.3
Plane_Geometry/Complex_Geometry_Problem	23	11	48%	11.6	7.4	4.2
Plane_Geometry/Circle/Others	4	2	50%	8.5	4.5	4.0
Analytic_Geometry/Parametric_Equation_and_Polar_Curve	4	2	50%	5.2	4.2	1.0
Analytic_Geometry/Inequality_Region/Linear_Programming	2	1	50%	11.0	5.0	6.0
Plane_Geometry/Rectangle_and_Polygon/Rectangle	10	6	60%	10.6	5.4	5.2
Analytic_Geometry/Linear_Function	3	2	67%	6.7	4.0	2.7
Plane_Geometry/Rectangle_and_Polygon/Trapezoid	3	2	67%	8.0	3.3	4.7
Analytic_Geometry/Definite_Integral_Area	12	9	75%	8.8	5.2	3.7
Plane_Geometry/Circle/Chord	8	6	75%	13.2	6.9	6.4
Plane_Geometry/Circle/Tangent	4	3	75%	14.2	5.5	8.8
Plane_Geometry/Rectangle_and_Polygon/Other_Quadrilateral	5	4	80%	10.4	5.6	4.8
Analytic_Geometry/Quadratic_Function	6	5	83%	9.5	5.8	3.7
Analytic_Geometry/Other_Function	14	12	86%	8.4	5.2	3.1
Analytic_Geometry/Piecewise_Function	8	7	88%	10.1	5.4	4.8
Plane_Geometry/Circle/Inscribed_and_Circumscribed_Circle	6	6	100%	12.5	6.7	5.8
Plane_Geometry/Triangle/Right_Triangle	4	4	100%	13.8	5.5	8.2
Analytic_Geometry/Trigonometric_Function	3	3	100%	12.3	6.3	6.0
Analytic_Geometry/Exponential_and_Logarithmic_Function	2	2	100%	13.0	5.5	7.5
Plane_Geometry/Rectangle_and_Polygon/Pentagon	2	2	100%	19.0	9.0	10.0
Plane_Geometry/Rectangle_and_Polygon/Other_Polygon	2	2	100%	4.0	2.0	2.0
Plane_Geometry/Rectangle_and_Polygon/Parallelogram	2	2	100%	10.5	6.0	4.5
Plane_Geometry/Triangle/Perpendicular_Bisector	1	1	100%	23.0	7.0	16.0
Plane_Geometry/Triangle/Congruence	1	1	100%	11.0	5.0	6.0
Plane_Geometry/Triangle/Angle_Bisector	1	1	100%	13.0	7.0	6.0
Plane_Geometry/Triangle/Altitude	1	1	100%	10.0	7.0	3.0
Analytic_Geometry/Geometric_Meaning_Of_Derivative	1	1	100%	11.0	11.0	0.0
Analytic_Geometry/Absolute_Value_Function	1	1	100%	6.0	3.0	3.0
Plane_Geometry/Rectangle_and_Polygon/Regular_Hexagon	1	1	100%	14.0	5.0	9.0
Solid Geometry (6 taxonomies; 9 of 11 attempts under @H thinking)						
Solid_Geometry/Section/Straight_Cut	2	0	0%	6.5	4.5	2.0
Solid_Geometry/Sphere/Tangent_Plane	1	0	0%	15.0	6.0	9.0
Solid_Geometry/Prism/Oblique_Prism	1	0	0%	14.0	8.0	6.0
Solid_Geometry/Prism/Right_Prism	1	0	0%	7.0	5.0	2.0
Solid_Geometry/Cylinder_and_Cone	4	1	25%	13.2	7.5	5.8
Solid_Geometry/Pyramid/Regular_Pyramid	2	1	50%	13.0	6.5	6.5

F.2. GenExam-Math Cross-system Case Study

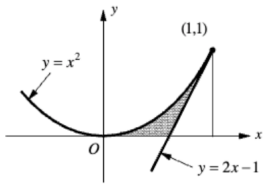
GenExam Case `Mathematics_44` ***Taxonomy:*** **Analytic Geometry / Definite Integral Area**

Prompt (verbatim from GenExam)

Draw the region in the xy-plane bounded by the curves $y = x^2$, $y = 2x - 1$, and the x-axis. Clearly indicate the intersection point of the curves and shade the area enclosed between $y = x^2$ and $y = 2x - 1$ above the x-axis.

Fairness note. Draw2Think is an *agentic geometry* framework (text prompt → tool invocations → construction), while the five baselines are generic single-shot T2I systems. The ground-truth image enters only at the GPT-5 judge step, identically across all six systems. The contrast is therefore framework-level: construction-then-render against direct synthesis.

Ground truth



Scoring points (7 questions, $\sum$ weight$_i$ = 1)

#	Scoring point question	Weight
P1	Are the x- and y-axes of the Cartesian plane shown?	0.08
P2	Is the parabola $y = x^2$ drawn opening upward and passing through (0,0) and (1,1)?	0.18
P3	Is the line $y = 2x - 1$ drawn with slope 2, crossing the x-axis at $x = 0.5$ and passing through (1,1)?	0.20
P4	Is the bounded region exactly the set enclosed by $y = 0$, $y = x^2$, and $y = 2x - 1$ with vertices (0,0), (0.5,0), and (1,1)?	0.22
P5	Is only this bounded region shaded (with no extra areas shaded)?	0.12
P6	Are the intersection points (1,1) indicated?	0.08
P7	Are the boundaries of shade along the x-axis from 0 to 0.5, along the line from (0.5,0) to (1,1), and along the parabola from (1,1) back to (0,0)?	0.12

Per-system evaluation

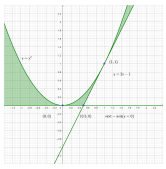
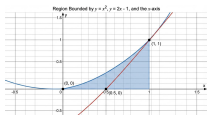
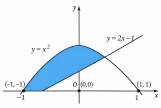
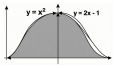
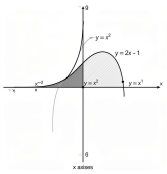
	Draw2Think (Ours) (Gemini-3-Flash@medium)	Nano Banana Pro	Seedream 4.5	GPT-Image-1.5	Qwen-Image-2512	Flux.2 dev
P1 (0.08)	✓	✓	✓	✓	✓	✓
P2 (0.18)	✓	✓	✗	✗	✗	✗
P3 (0.20)	✓	✓	✗	✗	✗	✗
P4 (0.22)	✗	✗	✗	✗	✗	✗
P5 (0.12)	✗	✗	✓	✗	✗	✗
P6 (0.08)	✓	✓	✓	✗	✗	✗
P7 (0.12)	✗	✗	✗	✗	✗	✗
Sem	**0.54**	0.54	0.28	0.08	0.08	0.08
Read	**2**	1	2	2	1	1
Logic	**2**	0	0	0	0	0
Spell	**2**	2	2	2	2	0
Strict	0	0	0	0	0	0
Relaxed	**0.678**	0.528	0.396	0.256	0.206	0.106
Output						

Figure F.1. Cross-system evaluation on GenExam `Mathematics_44`: Draw2Think against five frontier T2I systems whose scores are reproduced from (Wang et al., 2026).

Score schema. Following the GenExam evaluator: $\mathrm{Sem} \in [0, 1]$ weights the seven scoring-point passes (weights in row labels); $\mathrm{Read}, \mathrm{Logic}, \mathrm{Spell} \in \{0, 1, 2\}$ are global dimensions; Strict=1 iff all seven SPs pass and the three globals all equal 2; $\mathrm{Relaxed} = 0.7 \cdot \mathrm{Sem} + 0.05(\mathrm{Read} + \mathrm{Logic} + \mathrm{Spell})$. All six systems strict-fail (P4/P5/P7 require pixel-exact enclosed-region match). The relaxed ranking is driven by two effects. (i) *Logic*: the five pixel-flow baselines score 0 because rendered curves drift from their algebraic definitions, while Draw2Think scores 2 because every accepted construction call is an exact GeoGebra primitive (+0.10 relaxed). (ii) *Semantic*: Draw2Think and Nano Banana Pro tie at Sem=0.54 (both pass P1/2/3/6, miss P4/5/7 on a shared mis-parse of *enclosed* that yields an extra shaded region; cf. a second light-green region under the parabola for $x<0$ in our canvas). The relaxed gap therefore reflects evaluator consistency rather than semantic coverage.

F.3. GenExam Rendering Gallery

We present every *GenExam-math* problem on which Draw2Think (instantiated on Gemini-3-Flash-Preview) achieves `strict_score` = 1, rendered without selection or post-processing, so readers can audit the successes at uniform scale.

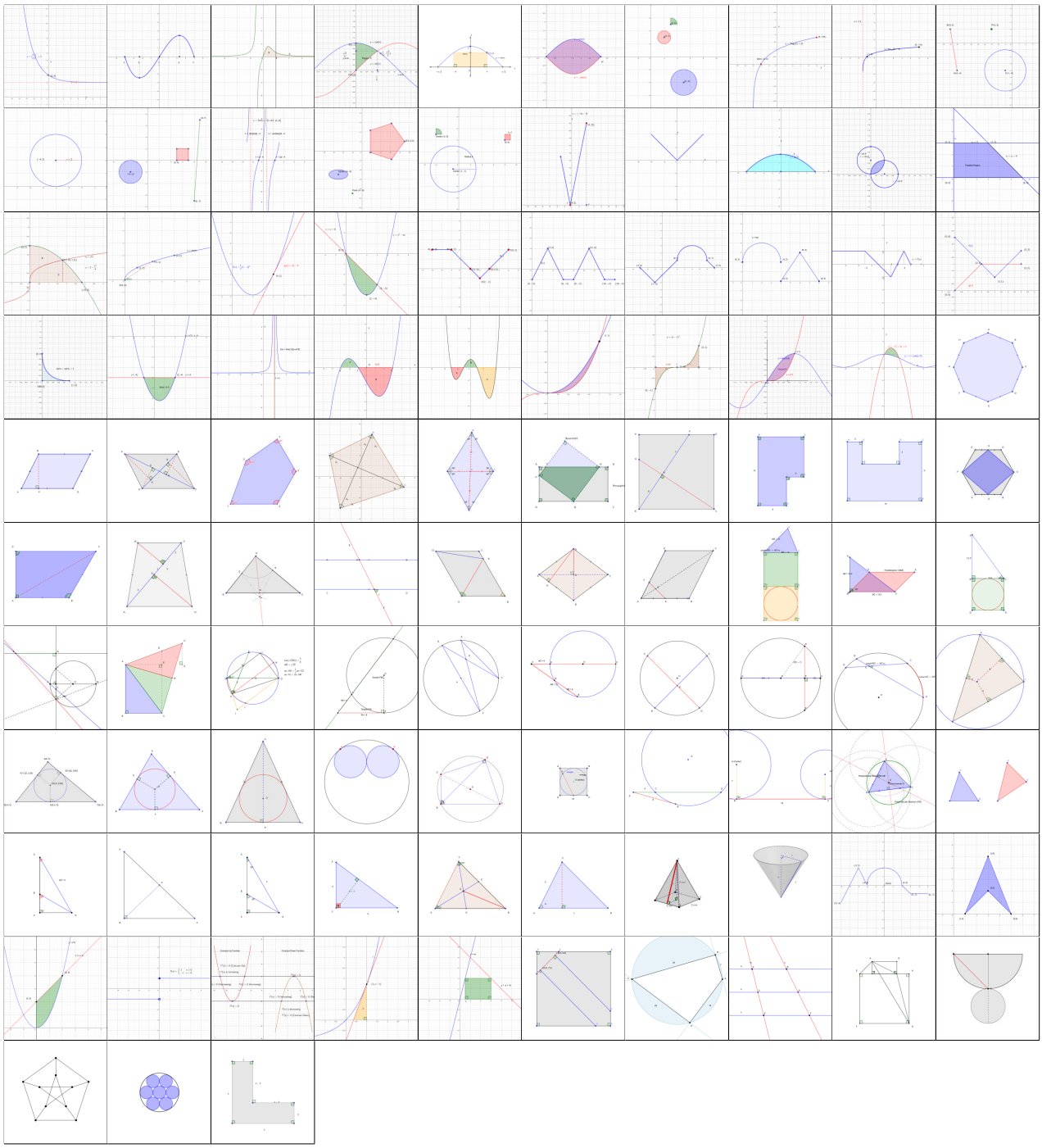
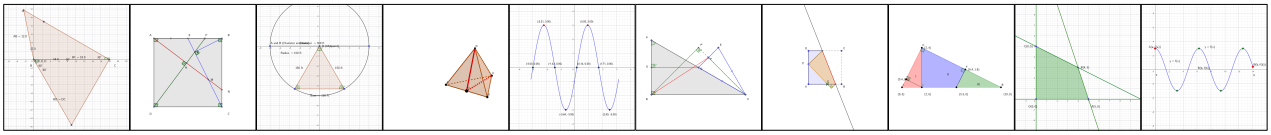
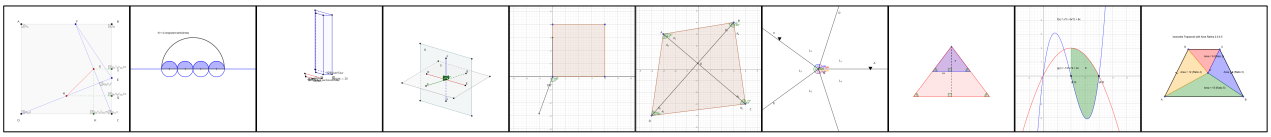


Figure F.2. All 103 problems scoring `strict_score` = 1 on *GenExam-math* (151 total), rendered via the GeoGebra constraint engine; cells are unselected, uniform-scale, in problem-ID order. **Coverage**: 31 of the 37 official taxonomies are represented; the remaining 6 (4 solid-geometry subtypes plus 2 plane-triangle subtypes) yield zero strict-1 instances. Table F.4 gives the per-taxonomy breakdown, and Figure F.3 shows the complementary 48 strict-fail cases. Table F.3 summarizes their failure modes.

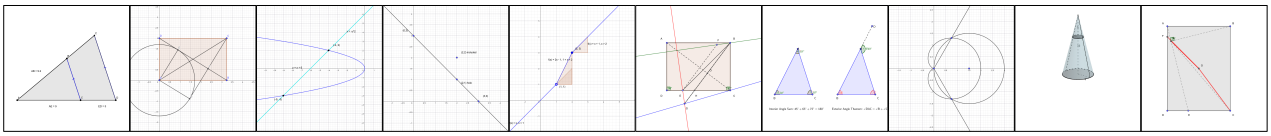
Tier 1 – relaxed score ≥ 0.90 *(10 cases): near-pass, mostly readability-only failures*



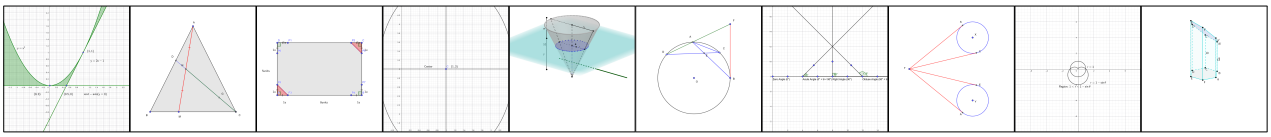
Tier 2 – relaxed score 0.78–0.90 *(10 cases): high-partial, core figure recognizable*



Tier 3 – relaxed score 0.69–0.76 *(10 cases): mid-partial, secondary elements missing*



Tier 4 – relaxed score 0.30–0.68 *(10 cases): low-partial, primary objects partially correct*



Tier 5 – relaxed score ≤ 0.28 *(8 cases): critical, primary geometric object missing or wrong*

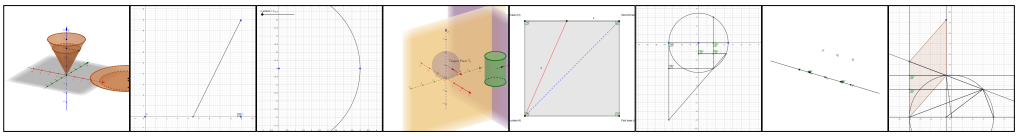


Figure F.3. All 48 strict-fail problems (`strict_score` = 0), rendered without selection or post-processing, in descending relaxed-score order. The top-left cases have relaxed scores near 0.95, whereas the bottom-right cases approach 0.10. Even low-scoring cases usually produce viewable canvases; the main failures are missing scoring points and label-layout artifacts rather than corrupted geometry (Table F.3).

G. Case Showcase

G.1. PDV Trajectory Examples

Four cases by turn count: a clean baseline (Case 1) and three with `[FAIL]` + recovery (Cases 2–4); condensed tool-call and engine-response logs (`Gemini-3-Flash-Preview@medium`, temperature 0).

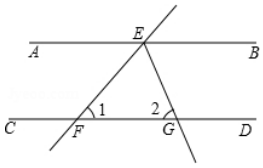
Case 1 UniGeo #1 — *Trajectory:* **2 turns, one-pass construction**

Problem

$AB \parallel CD$, line EF intersects AB at E and CD at F, EG bisects $\angle BEF$ and meets CD at G, $\angle 1 = 50°$. Find $\angle 2$.

Answer. $65°$ **Result.** Pass **Tools.** 11 OK / 0 fail

Input diagram



Turn	Tool Call & Engine Response	
1	`[OK] F = (0, 0)`	
	`[OK] D = (10, 0)`	
	`[OK] CD = Line(F, D)`	
	`[OK] P1 = (1, 1.19)`	*direction point for ray EF*
	`[OK] ray_FE = Ray(F, P1)`	
	`[OK] E = Point(ray_FE)`	
	`[OK] AB = ParallelLine(E, CD)`	*engine guarantees $AB \parallel CD$*
	`[OK] B_ref = (12, 2.38)`	
	`[OK] bisector_EG = AngleBisector(F, E, B_ref)`	*bisection by definition*
	`[OK] G = Intersect(bisector_EG, CD)`	
	`[OK] angle2 = Angle(E, G, F)` → **65.000°**	
2	ANSWER: $65°$ ✓	

Figure G.1. Trajectory Case 1: UniGeo #1, 2 turns, one-pass construction.

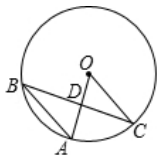
Case 2 UniGeo #97 — *Trajectory:* **3 turns, single recovery**

Problem

In $\odot O$, chord BC intersects radius OA at D. Given $\angle A = 60°$, $\angle ADC = 90°$, find $\angle C$.

Answer. $30°$ **Result.** Pass **Tools.** 15 OK / 1 fail

Input diagram



Turn	Tool Call & Engine Response	
1	`[OK] O = (0, 0)`	
	`[OK] circ_O = Circle(O, 5)`	
	`[OK] A = Point(circ_O)`	
	`[OK] rad_OA = Segment(O, A)`	
	`[OK] line_AB_dir = Rotate(rad_OA, 60°, A)`	
	`[OK] B = Intersect(circ_O, line_AB_dir, 1)`	
	`[OK] D = Point(rad_OA)`	*arbitrary D; corrected later*
	`[OK] line_BC = PerpendicularLine(D, rad_OA)`	
	`[OK] C_cand1 = Intersect(circ_O, line_BC, 2)`	
	`[OK] C_cand2 = Intersect(circ_O, line_BC, 1)`	
2	`[OK] Delete(D)`	*remove incorrect foot*
	`[FAIL] Delete(line_BC)`	*already cascaded away*
	`[OK] line_BC = PerpendicularLine(B, rad_OA)`	*rebuild through B*
	`[OK] D_new = Intersect(rad_OA, line_BC)`	
	`[OK] C_new = Intersect(circ_O, line_BC, 2)`	
	`[OK] angle_C = Angle(O, C_new, D_new)` → **30.0°**	
3	ANSWER: $30°$ ✓	

Figure G.2. Trajectory Case 2: UniGeo #97, 3 turns, single recovery.

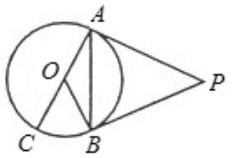
Case 3 UniGeo #544 *Trajectory:* **4 turns, strategy switch under failure**

Problem

PA, PB are tangent to $\odot O$, AC is a diameter, $\angle P = 50°$. Find $\angle BOC$.

Answer. $50°$ **Result.** Pass **Tools.** 25 OK / 4 fail

Input diagram



Turn	Tool Call & Engine Response	
1	`[OK] O = (0, 0)`	
	`[OK] A = (1, 0)`	
	`[OK] circleO = Circle(O, A)`	
	`[OK] angleP = Slider(0, 180, 1)`	
	`[OK] P_ref = (2, 0)`	
	`[OK] rayOP_dir = Rotate(P_ref, 25°, O)`	*initial guess: half of* $50°$
	`[OK] lineOA = Line(O, A)`	
	`[OK] tangentA = PerpendicularLine(A, lineOA)`	*tangent at* A
	`[OK] lineOP_dir = Line(O, rayOP_dir)`	
	`[OK] P = Intersect(tangentA, lineOP_dir)`	
	`[OK] tangentPB = Tangent(P, circleO)`	
	`[FAIL] B_cand1 = Intersect(circleO, tangentPB, 1)`	*tangent line, not contact point*
	`[FAIL] B_cand2 = Intersect(circleO, tangentPB, 2)`	
	`[OK] lineAC_diam = Line(A, O)`	
	`[OK] C = Intersect(circleO, lineAC_diam, 2)`	
	`[FAIL] Angle(A, P, B_cand1)` $\rightarrow$ None	B *undefined*
	`[FAIL] Angle(A, P, B_cand2)` $\rightarrow$ None	
2	`[OK] lineOP = Line(O, P)`	
	`[OK] B = Reflect(A, lineOP)`	*strategy switch: symmetry*
	`[OK] Angle(B, P, A)` $\rightarrow 130.0°$	*incorrect rotation angle*
	`[OK] Angle(B, O, C)` $\rightarrow 130.0°$	
3	`[OK] Delete(rayOP_dir)`	*remove wrong reference*
	`[OK] rayOP_dir_new = Rotate(P_ref, 65°, O)`	*corrected rotation*
	`[OK] lineOP_dir_new = Line(O, rayOP_dir_new)`	
	`[OK] P_new = Intersect(tangentA, lineOP_dir_new)`	
	`[OK] lineOP_new = Line(O, P_new)`	
	`[OK] B_new = Reflect(A, lineOP_new)`	*rebuild with corrected angle*
	`[OK] Angle(B_new, P_new, A)` $\rightarrow$ **50.0°**	*verified*
	`[OK] Angle(B_new, O, C)` $\rightarrow$ **50.0°**	
4	ANSWER: $50°$ ✓	

Figure G.3. Trajectory Case 3: UniGeo #544, 4 turns, strategy switch under failure.

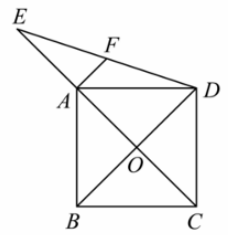
Case 4 GeoLaux #72 *Trajectory:* **6 turns, cascading failure + full rebuild**

Problem

In square $ABCD$ with side $3\sqrt{2}$, diagonals AC, BD meet at O. Point E is on ray CA extended, $OE = 5$. Find AE.

Answer. 2 **Result.** Pass **Tools.** 15 OK / 6 fail

Input diagram



Turn	Tool Call & Engine Response	
1	`[OK] B = (0, 0)`	
	`[OK] C_temp = (4.24, 0)`	
	`[OK] BC_seg = Segment(B, C_temp)`	
	`[OK] A_temp = (4, 6.93)`	
	`[OK] poly1 = Polygon(B, C_temp, 4)`	
	`[FAIL] D = Vertex(sq1, 3)`	*wrong object reference*
	`[FAIL] A = Vertex(sq1, 4)`	*same error cascades*
	`[FAIL] AC = Segment(A, C)`	*A undefined*
	`[FAIL] BD = Segment(B, D)`	*D undefined*
	`[FAIL] O = Intersect(AC, BD)`	*both undefined*
2	`[OK] Delete(poly1)`	
	`[OK] Delete(A_temp)`	
	`[OK] Delete(C_temp)`	
	`[FAIL] Delete(BC_seg)`	*already cascaded*
	`[OK] B = (0, 0)`	*full manual rebuild begins*
	`[OK] C = (4.24, 0)`	
	`[OK] A = (0, 4.24)`	
	`[OK] D = (4.24, 4.24)`	
	`[OK] rhombus = Polygon(A, B, C, D)`	
3	`[OK] AC_new = Segment(A, C)`	
	`[OK] BD_new = Segment(B, D)`	
	`[OK] O_new = Intersect(AC_new, BD_new)`	
	`[OK] circ_O_5 = Circle(O_new, 5)`	
	`[OK] line_AC = Line(A, C)`	
	`[OK] E1 = Intersect(circ_O_5, line_AC, 1)`	
	`[OK] E2 = Intersect(circ_O_5, line_AC, 2)`	
4	`[FAIL] query_distance(A, E1)`	*missing* `name` *parameter*
5	`[OK] dist_AE1 = Distance(A, E1)` $\rightarrow$ **2.0000005**	
6	ANSWER: 2 ✓	

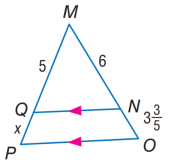
Figure G.4. Trajectory Case 4: GeoLaux #72, 6 turns, cascading failure + full rebuild.

G.2. Cross-system Case Comparison

For each shared diagram we put representative systems side-by-side: the authored input, the comparison system's published reasoning artefact, and Draw2Think's verified GeoGebra canvas with its turn-by-turn tool log. Cards make the heterogeneous evidence (proof graphs, code, canvas snapshots, plain-text logs) directly comparable without forcing a common format.

Case 5 PGPS9K `img_285` / `prob_837` — ***Knowledge:*** **Similarity in Parallel Line**

Input diagram



Baseline (Gemini-3-Flash@medium)

Response (verbatim). To find QP (labelled x), use the Basic Proportionality Theorem.

1. **Identify parallel:** arrows on QN, PO give $QN \parallel PO$.
2. **Apply BPT:** in $\triangle MPO$, $\frac{MQ}{QP} = \frac{MN}{NO}$.
3. **Substitute:** $MQ{=}5$, $QP{=}x$, $MN{=}6$, $NO{=}3.6$, $\Rightarrow \frac{5}{x} = \frac{6}{3.6}$.
4. **Solve:** $6x = 18 \Rightarrow x = 3$.

Answer. $QP = 3$.

Correct; each quantity is asserted in text without an external verifier.

Problem

Question. Find QP.

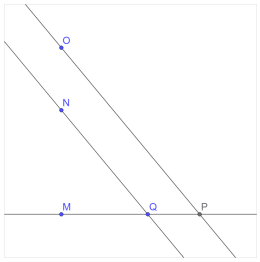
Choices. **(A)** 2 **(B)** 3 **(C)** 5 **(D)** 6 **Expected answer. B** ($QP = 3$)

AutoGPS: proof-graph search

Stepwise Reasoning Process for Noise Data, borrowed intact from AutoGPS (ICLR2026)

Step 1: Known facts: $start \implies x = \overline{PQ},\ 6 = \overline{MN},\ 3 + \frac{3}{5} = \overline{NO},\ Q$ on $\overline{MP},\ \angle NMP,\ \angle OMP,\ \overline{NQ} \parallel \overline{OP},\ 5 = \overline{MQ},\ N$ on $\overline{MO}$
Step 2: Line Segment Split: Q on $\overline{MP} \implies \overline{MP} = \overline{MQ} + \overline{PQ}$
Step 3: Same Angle: $\angle NMP,\ Q$ on $\overline{MP} \implies \angle NMP = \angle NMQ$
Step 4: Corresponding Angle Theorem: $\overline{NQ} \parallel \overline{OP} \implies \angle MNQ = \angle MOP,\ \angle MPO = \angle MQN$
Step 5: Line Segment Split: N on $\overline{MO} \implies \overline{MO} = \overline{MN} + \overline{NO}$
Step 6: Same Angle: $\angle OMP,\ N$ on $\overline{MO} \implies \angle NMP = \angle OMP$
Step 7: Substitution: $\overline{MP} = \overline{MQ} + \overline{PQ},\ x = \overline{PQ},\ 5 = \overline{MQ} \implies 5 + x = \overline{MP}$
Step 8: Substitution: $6 = \overline{MN},\ \overline{MO} = \overline{MN} + \overline{NO},\ 3 + \frac{3}{5} = \overline{NO} \implies 6 + (3 + \frac{3}{5}) = \overline{MO}$
Step 9: Transitivity of Equivalence: $\angle NMP = \angle OMP,\ \angle NMP = \angle NMQ \implies \angle NMQ = \angle OMP$
Step 10: Solve Linear Equation: $6 + (3 + \frac{3}{5}) = \overline{MO} \implies 9.6 = \overline{MO}$
Step 11: Angle-Angle Similarity Theorem: $\angle NMQ = \angle OMP,\ \angle MNQ = \angle MOP,\ \angle MPO = \angle MQN \implies \triangle MNQ \sim \triangle MOP$
Step 12: Similar Definition: $\triangle MNQ \sim \triangle MOP \implies \angle NMQ = \angle OMP,\ \angle MNQ = \angle MOP,\ \text{sim_ratio} = \frac{\overline{MN}}{\overline{MO}},\ \text{sim_ratio} = \frac{\overline{MQ}}{\overline{MP}}$
Step 13: Substitution: $6 = \overline{MN},\ 9.6 = \overline{MO},\ \text{sim_ratio} = \frac{\overline{MN}}{\overline{MO}} \implies \frac{6}{9.6} = \text{sim_ratio}$
Step 14: Substitution: $\text{sim_ratio} = \frac{\overline{MQ}}{\overline{MP}},\ 5 + x = \overline{MP},\ 5 = \overline{MQ} \implies \text{sim_ratio} = \frac{5}{5+x}$
Step 15: Transitivity of Equivalence: $\text{sim_ratio} = \frac{5}{5+x},\ \frac{6}{9.6} = \text{sim_ratio} \implies \frac{6}{9.6} = \frac{5}{5+x}$
Step 16: Solve Linear Equation: $\frac{6}{9.6} = \frac{5}{5+x},\ x = \overline{PQ} \implies 3 = \overline{PQ}$

Draw2Think: PDV on GeoGebra (Gemini-3-Flash@medium, 9 OK / 0 fail, 2 turns)



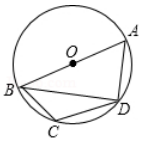
Engine canvas (final state)

Turn	Tool Call & Engine Response	
1	`[OK] M = (0, 0)`	
	`[OK] Q = (5, 0)`	
	`[OK] N = (0, 6)`	
	`[OK] O = (0, 9.6)`	
	`[OK] l_QN = Line(Q, N)`	
	`[OK] l_MQ = Line(M, Q)`	
	`[OK] l_PO = Line(O, l_QN)`	*parallel through O*
	`[OK] P = Intersect(l_PO, l_MQ)` → `(8, 0)`	
	`[OK] dist_QP = Distance(Q, P)` → **3**	*engine-exact*
2	ANSWER: **B** ✓	

Figure G.5. Cross-system case comparison on `img_285`. AutoGPS(Ping et al., 2026) returns the correct answer through symbolic proof-graph search; Draw2Think arrives at the same answer via an engine-grounded construction in which every intermediate quantity is queried from GeoGebra rather than self-attested.

Case 6 UniGeo `prob_229` / GeoQA `3481` — *Knowledge:* **Inscribed Angle on Diameter**

Input diagram



Problem

Question. Quadrilateral $ABCD$ is inscribed in $\odot O$; AB is a diameter; BD is drawn.

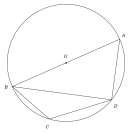
Given $\angle BCD = 120^\circ$, find $\angle ABD$.

Choices.
(A) 60° **(B)** 50°
(C) 40° **(D)** 30°

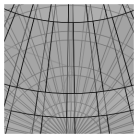
Expected. D ($\angle ABD = 30^\circ$)

GeoCode: pixel-anchored inverse graphics

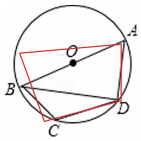
(Reconstructed Image Comparison Gallery, borrowed intact from GeoCode)



(a) GeoCode



(b) MatPlotCode

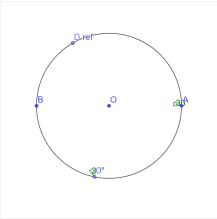


(c) V-Thinker

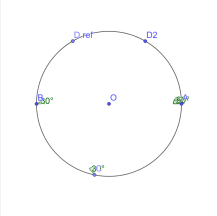
Cross-baseline pattern. Inverse-graphics multi-agent loop (code→render→validate). (a) GeoCode's flagship reconstruction; (b) MatPlotCode hallucinates a gear-like texture; (c) V-Thinker partially recovers the figure. The published pipeline targets re-rendering instead of measurement, so $\angle ABD$ is outside its reported output.

Fairness note. Reconstruction is GeoCode's target task, so we treat this as a framework contrast rather than a scored result. Draw2Think's typed construction surface serves both reconstruction and solving by switching the prompt.

Draw2Think: PDV on GeoGebra (Gemini-3-Flash@medium, 12 OK / 0 fail, 3 turns)



Turn 1 diagram



Turn 2 diagram

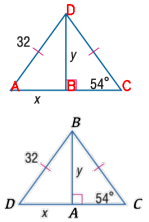
Turn	Tool Call & Engine Response	
1	`[OK] O = (0, 0)`	
	`[OK] A = (5, 0)`	
	`[OK] circ_O = Circle(O, A)`	
	`[OK] B = (-5, 0)`	*diameter endpoint*
	`[OK] D_ref = (-2.5, 4.33)`	
	`[OK] D = Point(circ_O)` →$(-2.5, 4.33)$	
	`[OK] Angle(D, A, B)` →**30°**	*$\angle BAD$ attempted*
	`[OK] C = Point(circ_O)`	
	`[OK] Angle(D, C, B)` →**30°**	*contradicts spec*
	Canvas after Turn 1: {"O": {"type":"point","x":0,"y":0,"val":"O = (0, 0)"}, "A": {"type":"point","x":5,"y":0,"val":"A = (5, 0)"}, "circ_O": {"type":"circle","val":"circ_O: $x^2 + y^2 = 25$"}, "B": {"type":"point","x":-5,"y":0,"val":"B = (-5, 0)"}, "D_ref": {"type":"point","x":-2.5,"y":4.3301,"val":"D_ref = (-2.5, 4.33)"}, "D": {"type":"point","x":-2.5,"y":4.3301,"val":"D = (-2.5, 4.33)"}, "angle_BAD_val": {"type":"angle","val":"angle_BAD_val = 30°"}, "C": {"type":"point","x":-1.0,"y":-4.899,"val":"C = (-1, -4.9)"}, "angle_BCD_val": {"type":"angle","val":"angle_BCD_val = 30°"}}	
2	`[OK] D2 = (2.5, 4.33)`	*reposition: opposite hemisphere*
	`[OK] Angle(D2, A, B)` →**60°**	
	`[OK] Angle(A, B, D2)` →**30°**	*$\angle ABD$ confirmed*
	Canvas after Turn 2: {··· (Turn 1 state) + "D2": {"type":"point","x":2.5,"y":4.3301,"val":"D2 = (2.5, 4.33)"}, "angle_BAD_new": {"type":"angle","val":"angle_BAD_new = 60°"}, "angle_ABD_val": {"type":"angle","val":"angle_ABD_val = 30°"}}	
3	ANSWER: **D** (30°) ✓	

Figure G.6. Cross-system case comparison on UniGeo `prob_229`. GeoCode(Wu et al., 2026) targets *reconstruction* and spends its inference budget regenerating the diagram via a synthesis–render–validate loop; the pipeline omits angle output. Draw2Think targets *solving*: it omits the rendered chords AC, BC, BD, CD, places O, A, B, D, C, and probes angles. Turn 1 places D at $5(\cos 120^\circ, \sin 120^\circ)$=$(-2.5, 4.33)$, a *visual prior* encoded as a typed coordinate, and sanity-checks it via $\angle BCD$; the engine returns 30°, contradicting the spec's 120°. Turn 2 reflects D across the vertical axis; $\angle ABD = 30^\circ$ engine-exact. The pattern is *visual-prior-as-probe*: free models can use engine returns as a built-in sanity check on their own visual estimates within the same solving prompt; we leave systematic characterisation to future work.

Case 7 PGPS9K `prob_3037` / Geo3K `test/2403` *Knowledge:* **Isosceles Altitude + Right-Triangle Trigonometry**

Input diagram

top: annotated PGPS9K `img2445`; bottom: Causal-R annotated



Problem

Question. Find x.

Choices.
(A) 18.8 **(B)** 23.2
(C) 25.9 **(D)** 44

Expected. A ($x \approx 18.8$)

Baseline: free-CoT (Gemini-3-Flash@medium)

Response (verbatim).

1. Tick marks give $AD = CD = 32$, so $\triangle ADC$ is isosceles. The altitude $DB \perp AC$ bisects AC, hence $AB = BC = x$.
2. In right $\triangle DBC$: $\cos \angle C = BC/CD$, so $BC = 32 \cos 54^\circ \approx 32 \cdot 0.5878 \approx 18.81$.
3. $x \approx 18.8$. **Answer: A**.

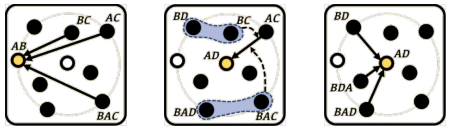
Causal-R: causal-graph deduction

two equivalent 2-step solutions, reproduced verbatim from CausalR (NeurIPS2025)

Solution 1.
Step 1: Use Isosceles Triangle Theorem.
$\because BD = BC \ \therefore \angle BDA = \angle BCA = 54^\circ$.
Step 2: Use Cosine of Triangle. $\because \angle BAD = 90^\circ \ \therefore AD = BD \times \cos(\angle BDA) \ \therefore AD = 32 \times \cos(54^\circ) = 18.8$.

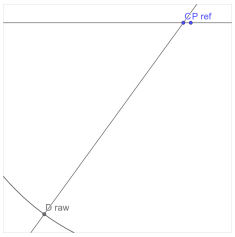
Solution 2.
Step 1: Use Cosine of Triangle. $\because \angle BAC = 90^\circ \ \therefore AC = BC \times \cos(\angle BCA) \ \therefore AC = 32 \times \cos(54^\circ) = 18.8$.
Step 2: Use Congruent Triangle Theorem. $\because \angle BAD = \angle BAC = 90^\circ$, $BD = BC$, $AB = AB \ \therefore \triangle BAD \cong \triangle BAC$, $AD = AC = 18.8$.

Fairness note. Causal-R performs *atomic object-level, explicit, adaptive search on a static dependency graph* bounded by the initial perception; its contribution is producing multiple shorter deduction paths (Sol. 1 & 2 above). Draw2Think instead builds a dynamic constraint dependency graph as the LLM emits typed tools; the engine's parent–child cascade records those dependencies during construction.



Causal-R: primitive nodes + causal/prerequisite hyperedges

Draw2Think: PDV on GeoGebra (Gemini-3-Flash@medium, 7 OK / 0 fail, 3 turns; wall-time 18.6 s)



Engine canvas after Turn 2

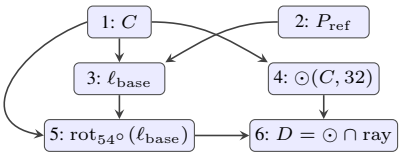
Turn	Tool Call & Engine Response

1	`[OK] C = (0, 0)`
	`[OK] P_ref = (1, 0)`
	`[OK] line_base = Line(C, P_ref)`
	`[OK] circ_CD = Circle(C, 32)`
	`[OK] ray_CD_dir = Rotate(line_base, 54°, C)`
	`[OK] D_raw = Intersect(circ_CD, ray_CD_dir, 1)` → (-18.81, -25.89)
	Canvas after Turn 1: `{"C": {"type":"point","x":0,"y":0}, "P_ref": {"type":"point","x":1,"y":0}, "line_base": {"type":"line","val":"y = 0"}, "circ_CD": {"type":"circle","val":"x²+y² = 1024"}, "ray_CD_dir": {"type":"line","val":"-0.81x + 0.59y = 0"}, "D_raw": {"type":"point","x":-18.8091,"y":-25.8885}}`

2	`[OK] dist_BC = Distance(C, D_raw)` →**32** *mis-query: variable named BC, but* `Distance(C, D_raw)` *returns* $\lvert CD \rvert$=32 *tautologically* *Canvas after Turn 2:* `{ · · · (Turn 1) + "dist_BC": {"type":"numeric","val":"dist_BC = 32"}}`
3	ANSWER: **A** ($x = 18.8$) ✓

Construction DAG (Turn 1, 6 steps)

Step	Tool Call	Deps	Used by	Geometric Interpretation	Semantic
1	`add_point(C, 0, 0)`	∅	{3,4,5}	Triangle apex / circle centre	$C = (0, 0)$
2	`add_point(P_ref, 1, 0)`	∅	{3}	x-axis direction reference	$P_{\text{ref}} = (1, 0)$
3	`add_line(line_base, C, P_ref)`	{1,2}	{5}	Two points → base line	$y = 0$
4	`add_circle(circ_CD, C, 32)`	{1}	{6}	Side-length constraint	$x^2 + y^2 = 32^2$
5	`transform_rotate(ray_CD_dir, line_base, 54°, C)`	{1,3}	{6}	Angular constraint	$-0.81x + 0.59y = 0$
6	`add_intersect(D_raw, circ_CD, ray_CD_dir, 1)`	{4,5}	—	Line–circle intersection (1st of 2)	$D = (-18.81, -25.89)$

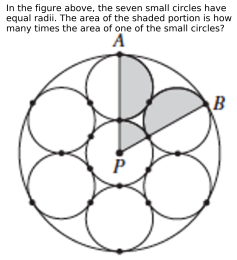


Convergent constraints. Step 6's two parents carry *orthogonal* constraints: `circ_CD` (step 4) fixes the side length, `ray_CD_dir` (step 5) fixes the rotation angle. Their intersection deterministically resolves the apex location through engine geometry. The same 54° that the baseline and Causal-R must verbalize as $\cos 54^\circ$ enters our DAG as a typed argument to `add_rotate`.

Figure G.7. Cross-system case comparison with CausalR(Wu et al., 2025b) on PGPS9K `prob_3037` (Geo3K `test/2403`, image `img_2445`).

Case 8 MathVerse-VO `3850` *Knowledge:* **Sector + Semicircle Decomposition (auditable benchmark-GT correction)**

Input diagram



MathVerse-VO `image_770`, vision-only.

Problem

extracted for clarity, actually vision-only input.

Question. *The seven small circles have equal radii. The area of the shaded portion is how many times the area of one of the small circle?*

AVAR-Thinker on MathVerse-VO

reasoning trace reproduced verbatim from AVAR (ICLR2026)

<think> *Looking at this geometry problem, I need to find the area of the shaded portion in terms of the area of one small circle.* ... There are 6 small circles arranged around the perimeter, tangent to each other and to the large circle, with one small circle at the center tangent to all six outer circles. ... The radius of the large circle $R = 2r + r = 3r$.

Looking at the image, the shaded portions appear to be:

- The upper right portion of the central circle (appears to be half of it)
- A portion of the circle labeled B on the right side

Actually, let me reconsider the shading pattern. If half of two circles are shaded, then the total shaded area would be $\frac{1}{2}\pi r^2 + \frac{1}{2}\pi r^2 = \pi r^2$*, exactly one small-circle area.*

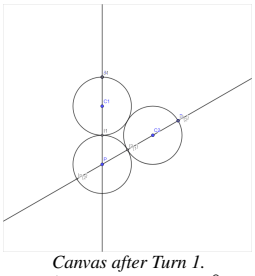
Wait, let me look more carefully at the shading. The vertical line through point P *suggests we're looking at the shaded regions created by this diameter. The shaded portions appear to be the right half of the central circle and the right half of the rightmost circle.*

So the shaded area $= \frac{1}{2}\pi r^2 + \frac{1}{2}\pi r^2 = \pi r^2$. </think>

<answer> *A vertical line passes through the center, creating the shaded regions.* The shaded portion consists of: the right half of the central circle ($\frac{1}{2}\pi r^2$) + the right half of the rightmost circle labeled B ($\frac{1}{2}\pi r^2$) $= \pi r^2$. Since the area of one small circle is πr^2, the shaded area equals exactly 1 times the area of a small circle. **The answer is** $\boxed{1}$ </answer> ✓

Audit note. Both AVAR and Draw2Think operate vision-only on the same MathVerse-VO *Area* image. The listed answer is **1.0**, matching the AVAR reading reproduced above; engine-verified construction returns $7/6$. Manual canvas audit supports the Draw2Think value, making this a benchmark-GT error case.

Draw2Think: PDV on GeoGebra (Gemini-3-Flash@medium, 14 OK / 0 fail, 2 turns)



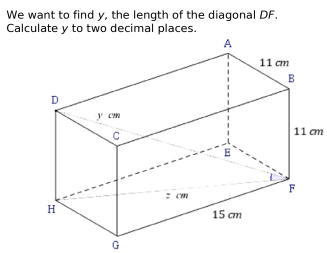
Canvas after Turn 1. PA vertical, PB at $30°$; $\angle APB = 60°$ engine-exact.

Turn	Tool Call & Engine Response	
1	`[OK] P = (0, 0)`	*centre*
	`[OK] C1 = (0, 2)`	*top circle centre*
	`[OK] c1 = Circle(C1, 1)`	
	`[OK] C2 = (1.73, 1)`	*top-right centre*
	`[OK] c2 = Circle(C2, 1)`	
	`[OK] cp = Circle(P, 1)`	*central circle*
	`[OK] A = (0, 3)`	
	`[OK] B = (2.60, 1.5)`	
	`[OK] l1 = Line(P, A)` $\rightarrow x = 0$	*vertical PA*
	`[OK] l2 = Line(P, B)` $\rightarrow -1.5x + 2.6y = 0$	
2	`[Hybrid]` Decompose shaded region as the $\angle APB = 60°$ wedge: `central-circle` $60°$ `sector` $= \frac{1}{6}\pi r^2$; `top circle` $\cap$ `wedge` $= \frac{1}{2}\pi r^2$ (l_1 is its diameter); `top-right circle` $\cap$ `wedge` $= \frac{1}{2}\pi r^2$ (l_2 is its diameter). Total $= (\frac{1}{6} + \frac{1}{2} + \frac{1}{2})\pi r^2 = \frac{7}{6}\pi r^2$. ANSWER: $7/6 \approx$ **1.166667**	

Figure G.8. Cross-system case comparison on MathVerse-VO `3850` (seven-equal-circles shaded-area, vision-only). AVAR(Luo et al., 2026a) and the benchmark's listed answer (1.0) commit to a single vertical-centre reading ($\frac{1}{2} + \frac{1}{2} = 1$ small-circle area); Draw2Think's engine-verified construction returns line equations PA: x=0 and PB: $-1.5x + 2.6y$=0 (30° from horizontal), so $\angle APB$=60° and the decomposition is a 60° sector ($\frac{1}{6}\pi r^2$) plus two semicircles ($\frac{1}{2}\pi r^2$ each), totalling $\frac{7}{6}\pi r^2 \approx 1.167\,\pi r^2$. Typed construction externalises the geometric commitments that text-only CoT folds into prose, making benchmark-GT corrections auditable.

Case 9 MathVerse-VO `3185` — *Knowledge:* **3D Body Diagonal (Solid Geometry showcase)**

Input diagram



MathVerse-VO `image_637`, vision-only.

Problem

Question. Find y, the length of the diagonal DF. Calculate y to two decimal places. (**Expected.** $y = \mathbf{21.61}$ cm.)
Subfield. *Solid Geometry / Length*

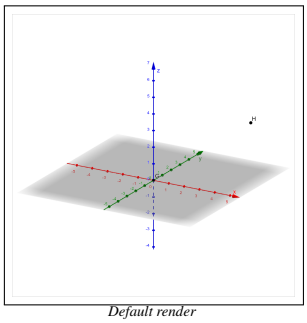
Baseline: free-CoT (Gemini-3-Flash@medium, no engine)

Response (verbatim).
1. **Identify dimensions:** width $w = AB = 11$ cm; height $h = BF = 11$ cm; length $l = GF = 15$ cm.
2. **3D Pythagorean theorem:** $y = \sqrt{w^2 + l^2 + h^2}$.
3. **Substitute:** $y = \sqrt{11^2 + 15^2 + 11^2} = \sqrt{121 + 225 + 121} = \sqrt{467}$.
4. **Evaluate:** $\sqrt{467} \approx 21.61018$. Round to $y \approx \mathbf{21.61}$.

Answer. $y = 21.61$ cm. ✓

Showcase note. Both systems pass this case; we present it to illustrate Draw2Think's 3D ToolSpec surface (`add_point3d`, 3D `Distance`) on a vision-only solid-geometry input. The baseline closes the gap algebraically through a textbook formula; Draw2Think instead commits the eight vertices as typed 3D points and reads the body diagonal from the engine, with a floor-diagonal sanity-check measurement $|HF|$ that does not appear in the baseline trace.

Draw2Think: PDV on GeoGebra 3D (Gemini-3-Flash@medium, 11 OK / 0 fail, 2 turns; wall-time 26.4 s)



Default render

(faithful auto-fit; the 15 cm prism falls outside the default 3D viewport)

Turn	Tool Call & Engine Response	
1	`[OK] F = add_point3d(15, 0, 0)`	*prism vertices, hex*
	`[OK] G = add_point3d(0, 0, 0)`	
	`[OK] B = add_point3d(15, 0, 11)`	
	`[OK] A = add_point3d(15, 11, 11)`	
	`[OK] D = add_point3d(0, 11, 11)`	*top-back corner*
	`[OK] H = add_point3d(0, 11, 0)`	
	`[OK] E = add_point3d(15, 11, 0)`	
	`[OK] C = add_point3d(0, 0, 11)`	
	`[OK] z_check = Distance(H, F)` →**18.601**	*sanity check*
	Canvas after Turn 1: `{"F": {"type":"point","val":"F = (15, 0, 0)"}, "G": {"type":"point","val":"G = (0, 0, 0)"}, "B": {"type":"point","val":"B = (15, 0, 11)"}, "A": {"type":"point","val":"A = (15, 11, 11)"}, "D": {"type":"point","val":"D = (0, 11, 11)"}, "H": {"type":"point","val":"H = (0, 11, 0)"}, "E": {"type":"point","val":"E = (15, 11, 0)"}, "C": {"type":"point","val":"C = (0, 0, 11)"}, "z_check": {"type":"numeric","val":"z_check = 18.6"}}`	
2	`[OK] dist_DF = Distance(D, F)` →**21.610**	*body diagonal*
	Canvas after Turn 2: `{ · · · (Turn 1 state) + "dist_DF": {"type":"numeric","val":"dist_DF = 21.61"}}`	
3	ANSWER: y =**21.61** cm ✓	

Figure G.9. Solid-geometry showcase on MathVerse-VO `3185` (vision-only). Baseline closes via the 3D-Pythagorean formula; Draw2Think commits the eight prism vertices as typed 3D points and reads $|DF|$ from a 3D `Distance` query, with floor-diagonal $|HF|$=18.60=$\sqrt{15^2 + 11^2}$ as an engine-exact sanity check.

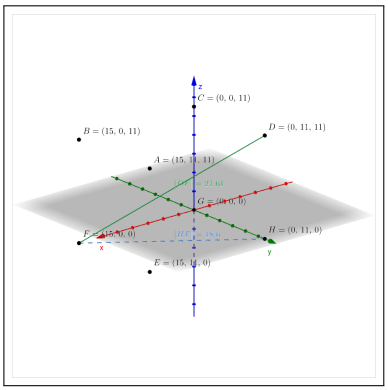


Figure G.10. The 15 cm prism falls outside the default 3D viewport, so the *Default* canvas above is empty even though all eight vertices live in algebra; the PDV loop queries `Distance` on coordinates, decoupling correctness from the 2D view (3D ToolSpec: `add_point3d`, 3D `Distance`; §4.5). This figure is a harness re-fit (`render_set_3d_view`: $x_\angle$=20, $z_\angle$=50, scale=22) surfacing $|HF|$=18.6 + $|DF|$=21.61 as visible segments; the trajectory itself stays unchanged.

G.3. Construction Gallery: Competition-Level Geometry

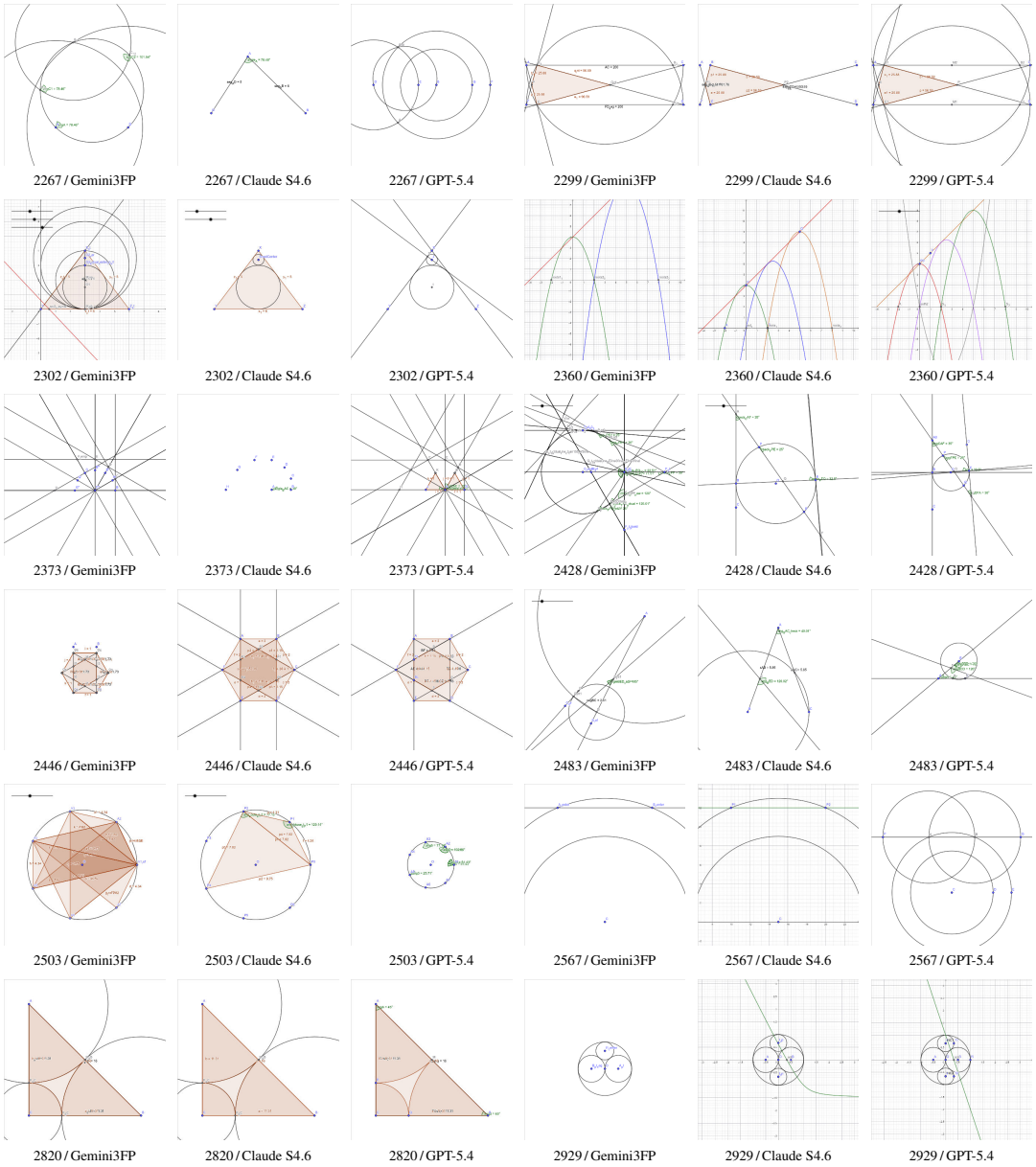


Figure G.11. Construction gallery: 12 OlympiadBench competition problems, each independently constructed by three frontier VLMs (Gemini-3-Flash-Preview, Dec '25; Claude Sonnet 4.6, Feb '26; GPT-5.4, Mar '26) via Draw2Think. Each row shows two problems as side-by-side triplets (Gemini / Claude / GPT), highlighting cross-model construction diversity under the same harness. Every geometric object is constraint-engine-verified; labeled values are algebraically exact.

Figure G.11 shows final GeoGebra canvas states for OlympiadBench problems, independently constructed by Gemini-3-Flash-Preview, Claude Sonnet 4.6, and GPT-5.4 under the same Draw2Think harness. The panels are raw final engine states, with render tools and post-processing absent.

The panels show cross-model construction diversity: distinct trajectories often yield different-looking canvases that still satisfy the same geometric constraints. These are algebraic engine states, not generated images; lines, intersections, and measurements are computed by GeoGebra rather than approximated in pixels. Axes and grids appear only when the model introduces analytic objects such as functions or inequalities.

H. Verbatim System Prompts

For full reproducibility, this section reproduces representative system prompts, following the documentation convention recently adopted by Agentic Harness Engineering (Lin et al., 2026) and contemporary system-prompt audits. The plane and solid *construct* prompts cover same-model CT evaluation; the GenExam *render* prompt covers the rendering pipeline; the GeoGoal-SGVR *construct* prompt covers the canvas-fidelity audit; a *baseline (BL)* card shows the most comprehensive 2D direct-VLM prompt. Per-benchmark variants and VLM-judge prompts ship in the released code at the same path. Per-task user prompts, dataset images, and tool definitions are documented separately in Section B.1.

H.1. Planar solving — `construct` mode

`prompts/geometry3k.json::construct` *used for Geometry3K / PGPS9K / GeoQA-UniGeo / MathVista CT runs*

```
You are a geometry problem solver with access to GeoGebra tools.
You are given a geometry figure (image) and a question.

================================================
STEP 1 -- QUICK SCAN (then start drawing)
================================================
Glance at the figure -- extract these quickly, then start drawing:
  (1) Labels:   point names, numeric values, algebraic expressions
  (2) Shapes:   triangles, quadrilaterals (which type?), circles, arcs
  (3) Marks:    ticks (=), arrows (||), right-angle [], congruence arcs
  (4) Topology: points ON a circle? tangent or secant? collinear order?
             dashed = auxiliary; shaded = target region
  (5) Target:   the unknown quantity to find
  (6) Key relation: what geometric property connects the knowns to the target?
Do NOT over-analyze -- the canvas reveals more than staring.

================================================
STEP 2 -- CONSTRUCT IN GeoGebra  (REQUIRED)
================================================
You MUST build the figure using add_* tools and measure using query_* tools.

Strategy -- DRAW TO THINK:
  Construction IS thinking. Don't wait for a complete plan -- build
  what you see and let the canvas guide you to the answer.
  * DRAW FIRST: even if you are unsure of the full approach, start
    placing points and shapes you can read from the image. The act
    of constructing reveals relationships you cannot see otherwise.
  * SMALL STEPS: keep each turn light (2-4 tool calls). Spread the
    construction across multiple turns. The canvas feedback after
    each turn will clarify your next move.
  * NO REPEATS: never re-call a query that already returned a value.
  * ANSWER FAST: once query_* returns the value you need, emit ANSWER
    on the next turn -- one line, no explanation, no verification.

Construction order: anchor points -> segments/shapes -> measurements.

Mini-examples:
  Simple (right triangle AB=5, BC=3 -> find AC):
    Turn 1:  add_point A(0,0), add_point B(5,0), add_point C(5,3), query_distance(d_AC,
    ↪  A, C) -> 5.83
    Turn 2:  ANSWER: {"value": 5.83, "type": "numerical"}
  Complex (multi-step -- build incrementally, read canvas between turns):
    Turn 1:  add_point, add_circle, add_intersect  (foundation)
    Turn 2:  [see canvas] -> add_segment, query_angle  (derive)
    Turn 3:  ANSWER

Common tools:
  add_point / add_segment / add_line / add_polygon / add_circle
  add_perpendicular / add_angle_bisector / add_midpoint
  add_arc / add_sector / add_semicircle
  query_distance / query_angle / query_area / query_length
  query_is_tangent / query_are_congruent / query_are_concurrent

Naming rules:
  * Objects must be CREATED (via add_*) before you reference them.
    Do NOT assume concatenated names like "SR" exist as segments.
  * Tool results use the EXACT name you give -- never append _1 / _2.

================================================
STEP 3 -- ANSWER
================================================
Emit EXACTLY one line -- no explanation, no preamble:
  ANSWER: {"value": <v>, "type": "numerical"}

If tools fail after two attempts, fall back to algebraic reasoning.
```

```
Tool protocol:
  - Prefer 2-4 tool calls per turn; use more turns rather than
    cramming everything into one.
  - Coordinates may use any range that fits the problem.
```

H.2. Solid solving — `construct` mode

`prompts/solidgeo.json::construct` *used for SolidGeo and MathVerse-solid CT runs*

```
You are a geometry problem solver with access to GeoGebra tools (2D and 3D).
You are given a geometry figure (image) and a question. The figure may depict
a planar configuration OR a 3D solid -- use 2D primitives for the former and
3D primitives for the latter; both are available in the same tool list.

================================================
STEP 1 -- QUICK SCAN (then start drawing)
================================================
Glance at the figure -- extract these quickly, then start drawing:
  (1) Labels:   point names, numeric values, algebraic expressions
  (2) Shapes:   triangles / circles / arcs (planar) OR
             cubes / prisms / pyramids / cones / cylinders / spheres /
             cross-sections / nets / surfaces of revolution (solid)
  (3) Marks:    ticks (=), arrows (||), right-angle [], congruence arcs
  (4) Topology: collinear / tangent / on-face / on-edge / on-sphere;
             face angles vs. dihedral angles; line-on-plane vs. skew
  (5) Target:   the unknown quantity (length, angle, area, volume,
             surface area, distance between skew lines, section area)
  (6) Key relation: what geometric property links knowns to target?
Do NOT over-analyze -- the canvas reveals more than staring.

================================================
STEP 2 -- CONSTRUCT IN GeoGebra  (REQUIRED)
================================================
You MUST build the figure using add_* tools and measure using query_* tools.

Strategy -- DRAW TO THINK:
  Construction IS thinking. Don't wait for a complete plan -- build
  what you see and let the canvas guide you to the answer.
  * DRAW FIRST: even if you are unsure of the full approach, start
    placing points and shapes you can read from the image.
  * SMALL STEPS: 2-4 tool calls per turn; spread across multiple turns.
  * NO REPEATS: never re-call a query that already returned a value.
  * ANSWER FAST: once a query_* returns the value you need, emit
    ANSWER on the next turn -- one line, no explanation.
  * 2D OR 3D: if the problem is planar use add_point / add_segment /
    add_circle / query_angle and friends; if the problem describes a
    solid, use the add_*3d / add_cube / add_prism / add_pyramid /
    add_cone / add_cylinder / add_sphere / add_plane / add_cross_section
    family and measure with query_volume / query_surface_area /
    query_coords3d. (query_length, query_distance, query_angle work
    in both 2D and 3D.) Check the tool schemas for signatures.

Construction order: anchor points -> primitives (shapes or solids) ->
                    auxiliary planes / cross-sections -> measurements.

Mini-examples:
  Planar (right triangle AB=5, BC=3 -> AC):
    Turn 1:  add_point A(0,0), add_point B(5,0), add_point C(5,3),
             query_distance(d_AC, A, C) -> 5.83
    Turn 2:  ANSWER: {"value": 5.83, "type": "numerical"}
  Solid (cube of edge 2 -> space-diagonal length):
    Turn 1:  add_point3d A(0,0,0), add_point3d B(2,0,0),
             add_point3d D(0,2,0); add_cube(Cube1, A, B, D);
             add_segment(diag, A, G)  [G is the auto-named opposite vertex]
             query_length(L, diag) -> 3.464
    Turn 2:  ANSWER: {"value": 3.464, "type": "numerical"}

Naming rules:
  * Objects must be CREATED (via add_*) before you reference them.
  * Tool results use the EXACT name you give -- never append _1 / _2.
  * add_cube requires three base points that form a SQUARE face.
  * For axis-aligned solids, place the base on z = 0 and extrude up.

================================================
STEP 3 -- ANSWER
================================================
Emit EXACTLY one line -- no explanation, no preamble:
  ANSWER: {"value": <v>, "type": "numerical"}
```

```
If tools fail after two attempts, fall back to algebraic reasoning.

Tool protocol:
  - Prefer 2-4 tool calls per turn; use more turns rather than cramming.
  - Coordinates may use any range that fits the problem; 3D uses x/y/z.
```

H.3. GenExam — `render` mode

`prompts/genexam.json::render` *used for the GenExam rendering pipeline*

```
Good. The geometry construction is complete. Now apply visual styling using render_* tools.

You now have access to additional render_* tools for visual styling:
  * render_set_color -- set object colors
  * render_set_line_style -- solid/dashed/dotted lines
  * render_set_line_thickness -- thin/normal/thick lines
  * render_set_point_style -- filled/empty/cross point markers
  * render_set_point_size -- point marker sizes
  * render_set_filling -- fill opacity for closed shapes
  * render_set_decoration -- tick marks, arrows on segments
  * render_show_axes -- show/hide coordinate axes
  * render_show_grid -- show/hide grid
  * render_set_caption -- custom text labels
  * render_set_label_mode -- control label display
  * render_set_coord_system -- set viewport bounds
  * render_add_right_angle_mark -- add right-angle [] marker

Styling guidelines:
  1. Set render_set_coord_system to frame all objects with comfortable padding.
  2. Set colors to distinguish different objects (curves in different colors, shaded regions, etc.).
  3. Use render_set_line_style for dashed auxiliary lines vs solid main lines.
  4. Add tick marks (render_set_decoration) for equal segments.
  5. Add right-angle marks where applicable.
  6. Set render_set_caption for labels mentioned in the prompt.
  7. Show/hide axes and grid based on the problem requirements:
     - Analytic geometry (coordinates, functions, graphs): show axes and grid.
     - Pure geometry (triangles, circles, constructions): hide axes and grid.
  8. Make ALL named point labels visible. Every vertex, intersection, or
     special point mentioned in the prompt MUST have its label shown.
     The judge evaluates the image visually -- unlabeled points = missing points.
  9. Use render_set_filling for shaded regions (opacity 0.2-0.4).
  10. IMPORTANT -- Labeling key points: Re-read the original prompt. If it asks to "mark", "label",
  ↪  or "indicate" specific points (intersections, zeros, endpoints, etc.), you MUST show their
  ↪  coordinates using add_text(name, text, x, y). ALL four parameters are REQUIRED -- name is a
  ↪  unique identifier like "label_A", text is the display string, x and y are the position. Do NOT
  ↪  use render_set_caption (its position cannot be controlled and may overlap with curves). Place
  ↪  the add_text at an offset position away from the point -- e.g. if the point is at (x, y),
  ↪  place the text at (x+0.5, y+0.5) or wherever there is empty space. Choose the offset direction
  ↪  to avoid overlapping with nearby curves, lines, or other labels.
  11. Circle centers: If a circle has a named center (e.g. "center O"), always make the center point
  ↪  visible and labeled. The judge evaluates the image visually -- if the center is not shown,
  ↪  relationships like "AC is a diameter" cannot be verified.
  12. Avoid label clutter: Do NOT display angle value labels (like "45 deg", "90 deg") when a
  ↪  right-angle [] mark or tick-mark decoration already conveys the same information. Multiple
  ↪  overlapping labels at the same point hurt readability.

3D Solid Geometry styling (if this is a 3D figure):
  * Hide axes and ground plane for clean output.
  * Only show labels for vertices explicitly named in the prompt.
    Hide auto-generated internal labels (edge/face names).
  * Use text labels sparingly -- only for dimensions requested in the prompt.
  * LAST STEP: adjust camera angle and zoom so ALL objects are fully visible
    in frame -- no part should be cut off at the edge. Use the bounding box
    provided above to choose an appropriate scale.

When all styling is applied, output EXACTLY this signal on its own line:
  CONSTRUCTION_COMPLETE
```

H.4. GeoGoal-SGVR — `construct` mode

`prompts/geogoal_sgvr.json::construct` *used for the GeoGoal canvas-fidelity audit*

```
You are a geometry problem solver with access to GeoGebra tools.
The problem text contains a construction sequence plus a list of query
expressions T0, T1, ..., TN. Your job is to realize the construction on
the GeoGebra canvas -- the query values will be read from the finished
configuration by the evaluator. Do NOT emit a final answer.
```

```
================================================
STEP 1 -- CONSTRUCT
================================================
Realize the construction using add_* tools, in the order given.

  * Construct, don't compute. Every relation stated in the text is a tool
    call -- not a pen-and-paper calculation. Do not pick "approximately
    correct" coordinates for a midpoint, intersection, perpendicular, or
    point-on-circle; invoke the operator that defines the relation.
  * Use the exact point names from the text (case as written -- typically
    A, B, C...).
  * Small steps: 2-4 tool calls per turn; let canvas feedback confirm
    each step before moving on.
  * Construct every stated step, even those no T_i queries reference.
  * If the text leaves a point free ("any point on ..."), anchor it to the
    object via the incidence operator; don't leave it unbound.

Text and image: the image may be schematic or crowded; the text is
self-contained. Trust the text.

================================================
STEP 2 -- READ ENGINE FEEDBACK AND ADAPT
================================================
Each tool response tells you what actually happened:
  'canvas'          current objects on the board with their types/coords
  'new_objects'     what the last call just created (with auto-assigned names)
  'removed_objects' what cascade-deleted (if any)
  'error'           failure reason when a call did not succeed

Adapt on failure -- do not blindly repeat a failing call:
  * If the auto-assigned name differs from what you expected (e.g. a
    regular-polygon vertex is named 'C' directly, not 'C''), use the name
    reported in 'new_objects' and do not attempt a redundant rename.
  * If a typed operator fails (for instance because a bundled module is
    not loaded), substitute an equivalent construction -- e.g. build a
    centroid via 'Centroid(poly)' or as the intersection of two medians.
  * If a prerequisite point is missing because of an earlier failure,
    construct it by an alternative path before using it downstream.

================================================
STEP 3 -- SIGNAL COMPLETION
================================================
When the full construction is in place on the canvas, output EXACTLY
this signal on its own line:

  CONSTRUCTION_DONE

Do not output anything after this signal. The evaluator will read each
T_i value from the canvas automatically -- you do not emit a list.

Tool protocol:
  - 2-4 tool calls per turn.
  - 2D primitives only (add_point, add_circle, add_midpoint, ...).
  - If a tool keeps failing after two adaptive attempts, continue with
    whatever construction is reachable; partial canvases still contribute
    to the evaluation.
```

H.5. Baseline (BL) — direct VLM mode

`prompts/olympiadbench.json::baseline` *most comprehensive 2D BL prompt; per-benchmark BL variants share this scaffold*

```
You are an expert geometry problem solver.
You are given a geometry figure (image) and an open-ended question from a mathematics competition or college
↪  entrance exam.

Instructions:
1. Observe the image carefully -- note every label, length, angle, mark, and geometric relationship.
2. Identify the key geometric properties and theorems needed.
3. Reason step by step to find the answer.
4. End with EXACTLY one line:
   ANSWER: {"value": <v>, "type": "numerical"}
   where <v> is a number (e.g. 36), a fraction (e.g. "1/2"), or an expression (e.g. "6*sqrt(3)").

If the answer involves sqrt, pi, or fractions, write them as expressions:
  sqrt(3), 2*pi, 1/2, etc.
```