GAMED.AI: A Hierarchical Multi-Agent Framework for Automated Educational Game Generation

Shiven Agarwal * **Yash Shah** * **Ashish Raj Shekhar*** **Priyanuj Bordoloi**
Vivek Gupta
Arizona State University
Project Page Demo Video Code
{ sagar147, yshah124, ashekha9, pbordolo, vgupt140}@asu.edu

Abstract

We introduce GAMED.AI, a hierarchical multi-agent framework that transforms instructor-provided questions into fully playable, pedagogically grounded educational games validated through formal mechanic contracts. Built on phase-based LangGraph sub-graphs, deterministic Quality Gates, and structured Pydantic schemas, GAMED.AI supports two template families encompassing 15 interaction mechanics across spatial reasoning, procedural execution, and higher-order Bloom's Taxonomy objectives. Evaluated on 200 questions spanning five subject domains, the system achieves a 90% validation pass rate against internal FOL-based structural validators (an architectural compliance metric, not an independent pedagogical benchmark), 98.3% schema compliance, and 73% token reduction over ReAct agents ($\sim$73,500 $\rightarrow$ $\sim$19,900 tokens/game) at $0.46 per game. Within this model configuration, these results suggest that phase-bounded architectural structure correlates more strongly with alignment quality than prompting strategy alone. Our demonstration lets attendees generate Bloom's-aligned games from natural language in under 60 seconds, inspect Quality Gate outputs at each pipeline phase, and browse a curated library of 50 games spanning all 15 mechanic types.

1 Introduction

Large Language Models now resolve 50–64% of real-world engineering tasks (Jimenez et al., 2024) and achieve high Pass@1 rates on function-level benchmarks (Chen et al., 2021; OpenAI, 2023), yet their effectiveness in producing *pedagogically valid* educational content remains limited—particularly where Bloom's Taxonomy alignment, mechanic contract enforcement, and structured competency evidence are required (Mislevy et al., 2003; Shute and Ventura, 2013).

*contributed equally

This gap matters, as game-based assessments achieve meta-analytic effect sizes of $g = 0.49$ on cognitive outcomes (Sailer and Homner, 2020), $g = 0.78$ on academic performance (Zeng et al., 2024), and $d = 0.29$ on learning (Wouters et al., 2013)—yet a single publication-ready game exceeds $10,000 to produce (Chapman, 2010), with mechanics routinely decoupled from objectives on existing platforms (Wang and Tahir, 2020). General-purpose agentic tools fall short on two grounds: missing Bloom's-level targeting, and self-correction loops that inflate tokens and accumulate errors (Yao et al., 2022; Verma et al., 2024)—yielding games syntactically correct but semantically wrong, e.g., testing *recall* when the objective requires *analysis* (Mislevy et al., 2003; Ji et al., 2023).

We introduce GAMED.AI, a hierarchical multi-agent framework that transforms instructor-provided questions into Bloom's-aligned educational games validated through formal mechanic contracts. Built on a LangGraph DAG with phase-specific sub-graphs, deterministic Quality Gates, and typed Pydantic schemas, GAMED.AI reduces the error propagation that makes prior agentic architectures impractical for structured content generation (Verma et al., 2024; Yao et al., 2022). The system generates validated games in under 60 seconds at $0.46 per game—achieving 73% token reduction over ReAct agents (Yao et al., 2022) and 90% validation pass rate across 200 test questions covering all 15 mechanics. Our contributions:

- To our knowledge, the first hierarchical multi-agent framework for educational game generation, with 15 interaction mechanics and Bloom's alignment contracts enforced before generation.

- 90% validation pass rate and 73% token reduction over ReAct agents, outperforming Claude

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 851–860

Code on Bloom's alignment under all four prompting conditions.

- A live demo enabling real-time game generation, pipeline observability, and a browsable library of 50 curated games spanning all 15 mechanics are open-sourced.

2 Related Work

Bloom's Taxonomy (Anderson and Krathwohl, 2001) and Evidence-Centered Design (Mislevy et al., 2003) require game mechanics to constitute valid competency evidence; the LM-GM framework (Arnab et al., 2015) provides the mechanic-mapping heuristic informing our Bloom's constraint table (Appendix A). Gamification succeeds when mechanics match learning goals (Sailer and Homner, 2020), fails when decorative (Hamari et al., 2014); formative feedback design (Shute, 2008) informs per-element validation at QG3.

Satisfying these constraints programmatically requires architectures that prevent error propagation. MetaGPT (Hong et al., 2024) and AutoGen (Wu et al., 2024) use role-bounded schemas; ReAct (Yao et al., 2022) adds self-correction but produces token inflation (Verma et al., 2024); flow engineering (Ridnik et al., 2024) and constrained decoding (Willard and Louf, 2023) make invalid states structurally unreachable. Widely adopted platforms (Kahoot, H5P) lack objective alignment (Wang and Tahir, 2020); GameGPT (Chen et al., 2023) targets speed without Bloom's alignment. GAMED.AI integrates Bloom's alignment, FOL-based validation, and a modular game engine in a single open-source framework.

3 GAMED.AI: Bridging Pedagogical Intent and Generative Execution

GAMED.AI accepts a natural language question or topic—with optional context (subject domain, target audience, difficulty level)—and produces a fully playable game, a structured alignment report, and a validation certificate confirming that all mechanic contracts are satisfied. As demonstrated in Figure 1, the end-to-end pipeline transforms instructor input into an interactive, pedagogically grounded educational game in under 60 seconds. Four design principles govern all architectural decisions:

- **Pedagogical primacy:** Every game is bound to a Bloom's level before generation; mechanic selection follows learning objectives (Anderson and Krathwohl, 2001; Mislevy et al., 2003).

- **Deterministic validation:** Every generative step is gated by a non-stochastic validator; LLM outputs are proposals subject to structural verification (Ji et al., 2023).

- **Structure over retry:** Typed schemas and phase boundaries prevent errors rather than catching them downstream (Willard and Louf, 2023).

- **Modularity:** New templates are registered via contract definition without modifying orchestration (Hong et al., 2024; Wu et al., 2024). The FOL-based validation framework is also taxonomy-agnostic: any competency framework with formal predicates—Webb's Depth of Knowledge, Gagné's conditions of learning, or domain-specific rubrics—can replace or extend the Bloom's constraint table without architectural changes.

3.1 Architectural Evolution

The current DAG architecture supersedes two prior designs: a **Sequential Pipeline** (56.7% VPR, ∼45,200 tokens/game) and a **ReAct Agent** system (72.5% VPR, ∼73,500 tokens/game). A key design target across all iterations was sub-$0.50 per-game cost, derived from the Chapman Alliance (2010) benchmark for scalable content authoring; neither prior design met this threshold. Quantitative comparison of all three architectures is shown in Figure 3. The detailed pipeline diagrams of the earlier versions are available in Appendix E.

3.2 GAMED.AI (DAG) Architecture

The current architecture emerged from the observation that prior designs conflated generation and validation into the same cognitive loop. The DAG separates them into six deterministic phases, each bounded by a Quality Gate.

3.2.1 System Architecture

The system is a hierarchical DAG in LangGraph with six phases—**Context Gathering**, **Concept Design**, **Game Plan**, **Scene Content**, **Assets**, and **Assembly**—each an independent sub-graph with typed I/O and a Quality Gate at its boundary (QG1–QG4; Figure 2). No agent in phase N receives input from phase $N{+}1$; no gate can be bypassed;

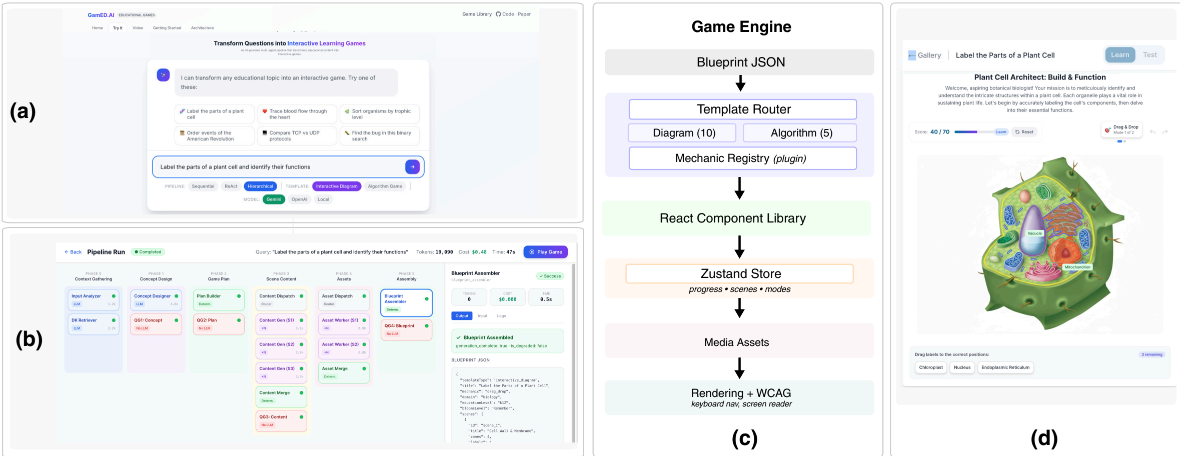

Figure 1: **End-to-end system demonstration. (a)** Instructor enters a natural language question. **(b)** DAG pipeline with real-time observability (per-agent traces, token/cost analytics, Quality Gate decisions). **(c)** Modular game engine: Blueprint JSON is routed via Template Router and Mechanic Registry to self-contained React components with shared Zustand state and WCAG rendering. **(d)** Generated Interactive Diagram Game (drag-and-drop) for "Label the Parts of a Plant Cell."

invalid states cannot propagate—a structural guarantee of the DAG topology (Ridnik et al., 2024; Hong et al., 2024).

3.2.2 Phase 0: Context Gathering

The pipeline opens with two parallel LLM nodes: an **Input Analyzer** that parses subject domain, target audience, and difficulty level from the natural language question, and a **Domain Knowledge Retriever** that grounds generation in curated sources (textbooks, curriculum standards, domain ontologies). Outputs are merged before Phase 1, ensuring concept design operates on verified domain context rather than open-ended generation (Mislevy et al., 2003).

3.2.3 Game Template Architecture

The generative surface comprises **two template families** with **15 interaction mechanics**. **Interactive Diagram Games** (10 mechanics: drag-and-drop, click-to-identify, trace-path, description matching, sequencing, sorting, memory match, branching scenario, compare/contrast, hierarchical) operate on spatial and relational content targeting visual and conceptual reasoning (Mayer, 2002; Sweller, 1988). **Interactive Algorithm Games** (5 mechanics: state tracer, bug hunter, algorithm builder, complexity analyzer, constraint puzzle) operate on procedural content targeting *applying*, *analyzing*, and *creating* objectives, grounded in algorithm visualization research (Hundhausen et al., 2002; Naps et al., 2002; Anderson and Krathwohl, 2001) and debugging-first pedagogy (Lee et al., 2014; Koedinger et al., 2004). Together, these support a library of **50 curated games** across five domains; the full Bloom's-to-mechanic mapping is in Appendix A.

3.2.4 Scene and Mechanic Composition

Templates span three structural configurations resolved automatically from Bloom's level and content complexity:

Single-scene, single-mechanic—one interaction type, one content context; covers ∼35% of the library.

Single-scene, multi-mechanic—2–3 interaction types within one content frame, validated through a state machine ensuring compatible I/O schemas; covers ∼40% (Sweller, 1988).

Multi-scene, multi-mechanic—2–4 causally connected scenes with monotonically increasing Bloom's levels, bounded by cognitive load constraints ($\leq$4 scenes, $\leq$3 mechanics/scene); covers ∼25% (Sweller, 1988). In the 200-question evaluation, these proportions held at 34%, 41%, and 25% respectively across the stratified question set; the remaining $<$1% were re-routed to single-scene configurations by QG1 on Bloom's underspecification.

3.2.5 Mechanic Contracts and Blueprint Generation

Template selection is a **constrained inference** in Phase 1: the Game Concept Designer (ReAct) re-

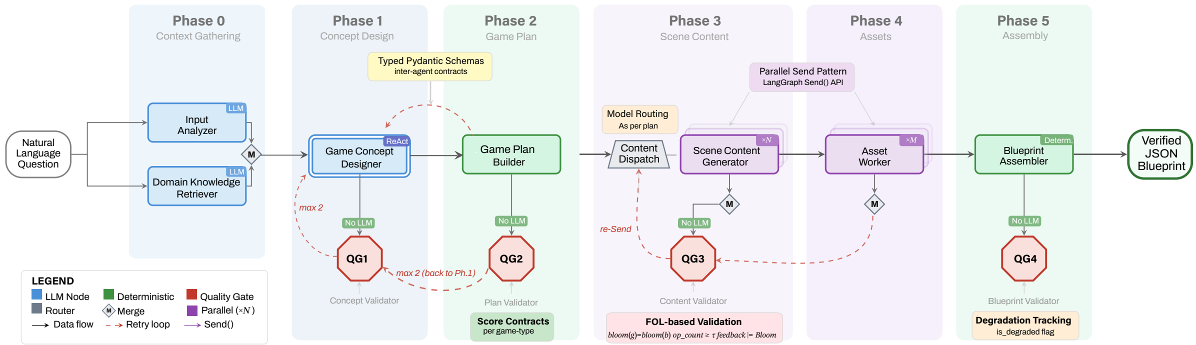

Figure 2: **GAMED.AI DAG architecture.** The pipeline comprises six phases: **Context Gathering**, **Concept Design**, **Game Plan**, **Scene Content**, **Assets**, and **Assembly**. Each phase operates as an independent sub-graph bounded by a deterministic Quality Gate (QG1–QG4) enforced without LLM inference. QG3 applies FOL-based Bloom's alignment predicates: $\text{bloom}(g) = \text{bloom}(b)$, $\text{op\_count} \geq \tau$, and feedback $\models$ Bloom. Inter-agent contracts are governed by typed Pydantic schemas; Phases 3–4 use parallel `Send()` patterns to dispatch N scene content workers and M asset workers. Dashed edges denote bounded retry loops (max 1–2).

solves input against a Bloom's-to-mechanic constraint table encoding valid competency evidence (Mislevy et al., 2003). The result is a **Game Blueprint**—a validated Pydantic document specifying learning objective, Bloom's level, template, and mechanic contract—produced before content generation begins and certified by QG1. Each contract defines the interaction primitive, content types, valid Bloom's range, and completion conditions, enforcing pedagogical alignment as a structural constraint (Anderson and Krathwohl, 2001; Shute and Ventura, 2013). QG2 subsequently validates the full Game Plan against Score Contracts per game type before any content generation begins.

3.2.6 Generation and Assembly

Parallel Generation. Phase 2 (Game Plan) produces a validated blueprint that gates entry into the parallel generation stages. Phase 3 dispatches N parallel **Scene Content Generators** via LangGraph `Send()` patterns; Phase 4 dispatches M parallel **Asset Workers** producing visual assets (SVG diagrams or text-synthesised visuals), instructional text (directions, hints, per-element feedback), and interaction specifications (drag targets, click regions, sequence orders).

FOL-Based Content Validation. QG3 validates all content against FOL-based Bloom's alignment predicates—$\text{bloom}(g) = \text{bloom}(b)$, $\text{op\_count}(g) \geq \tau_{\text{contract}}$, and per-element feedback predicates entailing the target Bloom's level–using rule evaluation without LLM inference, ensuring constant cost and formal verifiability. Failed scenes trigger a bounded `re-Send` loop (max 2).

Assembly and Schema Compliance. Phase 5 instantiates the selected template as a React component via the **Blueprint Assembler** and injects validated content—the same orchestration layer produces all 15 mechanics through component swapping, not code regeneration. QG4 performs final blueprint validation with `is_degraded` flag tracking; inter-agent communication achieves 98.3% Pydantic schema compliance[1] (Wu et al., 2024; Hong et al., 2024).

3.2.7 Deployment and Game Library

The orchestration layer is model-agnostic: a declarative preset system enables per-agent model selection across closed-source APIs (GPT-4 OpenAI, 2023, Gemini Google DeepMind, 2023) and open-source models (Llama 3 Grattafiori et al., 2024, Mistral Jiang et al., 2023, Qwen Qwen Team, 2025) without pipeline modification; performance under open-source configurations is not benchmarked in this work (see Limitations). The 50-game library (curated from the evaluation corpus) serves as both demo set and regression corpus; every game emits structured outcome data including score, interaction trace, and inferred Bloom's level.

3.3 Modular Game Engine

The frontend implements a **plugin architecture** (Figure 1c): each of the 15 mechanics is a self-contained React component registered by contract

[1] 3.4 of every 200 inter-agent messages failed schema validation, measured across all pipeline runs.

type, enabling extension through registration without modifying orchestration layers. Both template families share interaction primitives built on dnd-kit (Bhatt, 2024) with custom collision detection and keyboard/touch support. State management follows a dual architecture: Diagram Games use a centralised Zustand store for multi-mechanic coordination; Algorithm Games use localised reducer hooks for step-through interactions. The engine is designed for WCAG-aligned keyboard navigation and screen reader announcements; formal accessibility audit is planned as part of the classroom evaluation phase.

3.4 Pipeline Observability

The demonstration includes a real-time observability dashboard (Figure 1b) with **three view modes**: timeline, DAG graph (ReactFlow with execution-state highlighting), and cluster view grouped by phase. Per-agent **token and cost analytics** show stage-level consumption with USD breakdown.

3.5 Design Validation

Section 4 presents the full evaluation across all three architectures ($N = 200$, all 15 mechanics).

4 Evaluation

4.1 Scope

This evaluation measures **architectural validity**: validation pass rate, token efficiency, and structural Bloom's alignment. It does not measure learning outcome gains (see Limitations). VPR functions analogously to a type-checker in software engineering: it provides a deterministic, non-stochastic guarantee that the pipeline's formal contracts are satisfied—a necessary but not sufficient condition for pedagogical effectiveness. Independent evidence of learning improvement requires a controlled classroom study, which is the primary future direction.

4.2 Setup

200 questions from five domains (biology, history, CS, mathematics, linguistics) stratified across Bloom's levels and covering all 15 mechanics. All architectures used **GPT-4-turbo-2024-04-09** (OpenAI, 2023) (temp. 0.3, seed 42) for planning/validation and **gemini-1.5-pro-001** (Google DeepMind, 2023) (temp. 0.4) for asset generation, logged via LangSmith with per-call granularity.

4.3 Baselines

Five categories: **manual authoring** by five educators via Genially/H5P (human-quality ceiling); **EdTech platforms** (Kahoot, Quizlet, Nearpod, H5P) and GameGPT (Chen et al., 2023); **Claude Code** under four prompting conditions (zero-shot, one-shot content, one-shot instructional, multi-turn) across 30 stratified questions each; and **internal baselines** (Sequential Pipeline, ReAct Agent) on all 200 questions covering both template families.

4.4 Human rating methodology

Educational Correctness and Playability ratings (Table 4) were collected from **five domain-expert raters** (3 educators with $\geq$5 years experience; 2 SMEs per domain) using a behaviorally-anchored rubric. Raters were **blind to system condition**. Inter-rater reliability was acceptable (Educational Correctness: $\text{ICC}_{(2,5)} = 0.81$, 95% CI [0.74, 0.87]; Playability: $\text{ICC}_{(2,5)} = 0.78$, 95% CI [0.71, 0.84]). The comparison between GAMED.AI (4.2/5) and manual authoring (4.3/5) is not statistically significant ($t(199) = 1.04$, $p = 0.30$). We note this result reflects insufficient statistical power to detect a difference rather than confirmed equivalence; formal equivalence testing with a pre-specified margin is left for future work.

Rater–system agreement on Bloom's alignment. To address the question of whether human judgement corroborates the system's own Bloom's alignment verdict, we compared QG3 pass/fail decisions against rater Educational Correctness scores. Of the 180 games that passed QG3 validation, 87% (157/180) received Educational Correctness $\geq 4/5$ from raters; of the 20 that failed QG3, 75% (15/20) scored $\leq 3/5$. This positive but imperfect correspondence suggests that formal contract satisfaction is a useful proxy for expert-rated pedagogical quality, while also confirming that raters and the system can diverge—most often when content is factually correct but targets a Bloom's level adjacent to the specified one (e.g., *application* vs. *analysis*). Systematic human–system disagreement analysis across all 15 mechanics is left for future work.

4.5 Validation pass rate

GAMED.AI achieves a VPR of **90.0%**—17.5 percentage points above ReAct Agents (72.5%) and 33.3 points above the Sequential Pipeline (56.7%), confirmed significant ($\chi^2(2, N = 600) = 57.0$, $p < 0.001$, Cramér's $V = 0.31$). Architectural Validation Pass Rate (VPR) measures structural contract compliance against GAMED.AI's own FOL-based validators; it is an internal architectural metric, not an independent measure of pedagogical effectiveness.

Token consumption and cost. The 73% token reduction from ReAct Agents to the DAG ($\sim$73,500 $\rightarrow$ $\sim$19,900 tokens/game) is structural: architecture explains 87% of token consumption variance ($\eta^2 = 0.87$, $F(2, 597) = 1{,}996$, $p < 0.001$). We note that ReAct agents perform self-correction loops by design; this comparison partly reflects differing amounts of work done per generation, not only efficiency. GAMED.AI is the only architecture meeting the sub-$0.50 cost requirement (Interactive Diagram Games average $0.46; Algorithm Games average $0.43 due to fewer vision model calls).

4.6 Per-mechanic performance

Figure 3 summarises results by architecture. Across all 15 mechanics, VPR ranges from 96.2% (`DRAG_DROP`) to 60.0% (`DESC_MATCHING`) for Interactive Diagram Games, and from 94.4% (`STATE_TRACER`) to 80.0% (`CONSTRAINT_PUZZLE`) for Interactive Algorithm Games; mean educational correctness is 4.2/5. Algorithm Games average higher token consumption ($\sim$23,500 vs. $\sim$17,900 tokens/game) but lower per-game cost due to fewer vision model calls. Full per-mechanic breakdowns are in Table 4 (Appendix B).

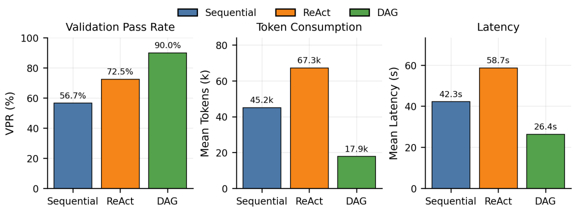

Figure 3: **Quality and efficiency metrics by architecture ($N = 200$ per condition, 15 mechanics).** Architecture explains 87% of token consumption variance ($\eta^2 = 0.87$); VPR gain of 17.5 pp over the next-best design.

5 Baselines Comparison

Table 1 summarizes all systems across time, interactivity, cost, and Bloom's alignment; two findings stand out.

System	Time	Cost	Game Gen.
Kahoot (Wang and Tahir, 2020)	15–30 min	$7/mo[a]	Pre-built
Quizlet (Quizlet Inc., 2024)	20–40 min	$8/mo[a]	Pre-built
Genially (Genially Web S.L., 2024)	60–120 min	$25/mo[a]	Template
H5P (H5P Group, 2024)	40–90 min	Free (OSS)	Template
GameGPT (Chen et al., 2023)	$\sim$10 min	$0.60	Template
Manual (Chapman, 2010)	60–240 min	$50–150[b]	Authored
GAMED.AI (DAG)	**<1 min**	**$0.46**	**Dynamic**

Table 1: Platform comparison of authoring latency and unit costs. GAMED.AI achieves sub-minute generation at $0.46/game, outperforming both template-based AI and manual labor. [a]Subscription fees for platform access; [b]Estimated expert labor cost per unit.

GAMED.AI compresses 60–240 minutes of expert authoring into under 60 seconds at a fixed cost below the cheapest subscription tier of any listed platform. Expert raters scored it at 4.2/5 vs. 4.3/5 for manual authoring ($t(199) = 1.04$, $p = 0.30$); this non-significant result reflects insufficient statistical power (n=5), not confirmed equivalence—formal equivalence testing with a pre-specified margin is left for future work. GameGPT (Chen et al., 2023) addresses creation speed but provides neither Bloom's targeting nor contract validation; MetaGPT (Hong et al., 2024) and AutoGen (Wu et al., 2024) provide role-bounded multi-agent coordination for software engineering tasks, with schemas governing code artifacts rather than pedagogical alignment predicates.

The Claude Code comparison functions as an **ablation of the contract mechanism**: Claude Code received identical learning objectives but not the mechanic contract schemas. Under these conditions, Claude Code produced functional games in 100% of attempts—but only 23% passed Bloom's alignment at zero-shot; the multi-turn ceiling of 67% at $0.80/run remains 23 pp below GAMED.AI's 90% VPR at lower cost (Table 2). This gap suggests FOL-based validation and phase-bounded generation provide structural guarantees beyond prompting alone (Ji et al., 2023; Mislevy et al., 2003); a controlled ablation providing equivalent schemas remains future work.

Relationship to guided decoding. Guided decoding approaches (Willard and Louf, 2023) enforce schema compliance at the token level, making syntactic violations structurally unreachable during generation. GAMED.AI's FOL-based Quality Gates are *complementary*, not redundant: guided decoding guarantees syntactic schema adherence (e.g., valid JSON field types), while QG-level predicates enforce *semantic* pedagogical constraints—Bloom's level matching, feedback entailment, and mechanic–objective coherence—that cannot be expressed as a JSON schema alone. A hybrid combining token-level guided decoding for structural compliance with QG predicates for semantic alignment is a natural extension of the current architecture.

Prompting Strategy	Align. (%)	Tokens (K)	Cost ($)
Zero-Shot	23.0	46.9	0.92
One-Shot (Exemplar)	41.0	67.2	1.34
One-Shot (Instr.)	48.0	74.4	1.51
Multi-Turn	67.0	140.5	6.35
GAMED.AI (DAG)	**90.0**	**19.9**	**0.46**

Table 2: Bloom's alignment, total token consumption, and total cost across Claude Code prompting conditions and GAMED.AI. All Claude Code runs use Claude-Opus-4-6; GAMED.AI uses the full DAG pipeline.

6 Conclusion and Future Work

We present GAMED.AI, a hierarchical multi-agent framework that generates Bloom's-aligned educational games through formal mechanic contracts and deterministic Quality Gates—achieving 90% validation pass rate, 73% token reduction over ReAct baselines, and sub-minute generation at $0.46/game. Within this configuration, phase-bounded architecture with contract validation correlates with improved alignment and token efficiency; replication across model families remains open. Future work targets human-in-the-loop blueprint negotiation, frame-based game engines (Phaser.js) for physics mechanics, expanded template families, and large-scale classroom evaluation measuring learning outcome gains.

Limitations

Schema coverage. Spatial anchoring for `DESC_MATCHING` and `TRACE_PATH` is underspecified, producing 14 of 20 validation failures; relational-link extensions are underway.

Game diversity. The current game engine supports 15 mechanic types implemented as React components; this constraint ensures structural reliability but limits output diversity relative to open-ended game authoring. Template extensibility is by design—new mechanics register via contract definition without modifying orchestration—but physics-based and open-world interaction patterns (e.g., Phaser.js) are not yet supported. Expanding the mechanic library and template families is a primary engineering direction.

Model and language scope. All reported metrics use GPT-4-turbo + Gemini-1.5-pro exclusively. The orchestration layer is model-agnostic by design (declarative presets, no model-specific prompting), but performance under open-source alternatives (Llama 3, Mistral, Qwen) has not been benchmarked in this work. We anticipate schema compliance to degrade for smaller parameter models due to reduced instruction-following reliability; quantifying this degradation is a concrete next step. Games are English-only; multilingual support is planned.

Student-facing validation. The evaluation measures architectural validity, not learning outcomes. Expert ratings (n=5) provide evidence of pedagogical quality from domain specialists, but may diverge from learner judgements; rater sample size is insufficient for broad generalization. The positive correlation between QG3 pass/fail and expert Educational Correctness scores (Section 4) is encouraging but not a substitute for empirical learning-gain measurement. Controlled classroom studies are the primary future direction.

Broader Impact

GAMED.AI lowers cost and expertise barriers to structurally validated game authoring. This is a **system paper**: the contribution is a validated authoring pipeline, not a demonstration of improved student learning. The claim is that the system reliably produces structurally correct, Bloom's-aligned games at low cost; whether those games improve learning outcomes over alternatives is an empirical question requiring a controlled classroom study, which is explicitly outside the scope of this work and is the primary future direction. Current evidence is limited to architectural metrics and expert ratings; the system should be understood as a validated authoring tool, not a proven pedagogical intervention. Open-source release and model-agnostic deployment support reproducibility and data sovereignty.

Risks. Quality Gates validate structure but not factual accuracy—incorrect domain knowledge can propagate into games. Mitigations include curated knowledge retrieval, deterministic validators, and an instructor-facing observability dashboard. GAMED.AI is designed to augment, not replace, pedagogical judgement.

References

Lorin W Anderson and David R Krathwohl. 2001. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives: complete edition*. Addison Wesley Longman, Inc.

Sylvester Arnab, Theodore Lim, Maira B Carvalho, Francesco Bellotti, Sara De Freitas, Sandy Louchart, Neil Suttie, Riccardo Berta, and Alessandro De Gloria. 2015. Mapping learning and game mechanics for serious games analysis. *British journal of educational technology*, 46(2):391–411.

Claudéric Bhatt. 2024. dnd kit: A lightweight, modular, performant, accessible and extensible drag & drop toolkit for React. https://dndkit.com/. Open-source library.

Bryan Chapman. 2010. How long does it take to create learning. *Chapman Alliance*.

Dake Chen, Hanbin Wang, Yunhao Huo, Yuzhao Li, and Haoyang Zhang. 2023. Gamegpt: Multi-agent collaborative framework for game development. *arXiv preprint arXiv:2310.08067*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Genially Web S.L. 2024. Genially. https://genially.com.

Google DeepMind. 2023. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, and 1 others. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

H5P Group. 2024. H5P: Create, share and reuse interactive HTML5 content. https://h5p.org.

Juho Hamari, Jonna Koivisto, and Harri Sarsa. 2014. Does gamification work?–a literature review of empirical studies on gamification. In *2014 47th Hawaii international conference on system sciences*, pages 3025–3034. Ieee.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2024. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pages 23247–23275.

Christopher D Hundhausen, Sarah A Douglas, and John T Stasko. 2002. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pages 54107–54157.

Kenneth R Koedinger, Vincent Aleven, Neil Heffernan, Bruce McLaren, and Matthew Hockenberry. 2004. Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In *International conference on intelligent tutoring systems*, pages 162–174. Springer.

Michael J Lee, Faezeh Bahmani, Irwin Kwan, Jilian LaFerte, Polina Charters, Amber Horvath, Fanny Luor, Jill Cao, Catherine Law, Michael Beswetherick, and 1 others. 2014. Principles of a debugging-first puzzle game for computing education. In *2014 IEEE symposium on visual languages and human-centric computing (VL/HCC)*, pages 57–64. IEEE.

Richard E Mayer. 2002. Multimedia learning. In *Psychology of learning and motivation*, volume 41, pages 85–139. Elsevier.

Robert J Mislevy, Russell G Almond, and Janice F Lukas. 2003. A brief introduction to evidence-centered design. *ETS Research Report Series*, 2003(1):i–29.

Thomas L Naps, Guido Rößling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, and 1 others. 2002. Exploring the role of visualization and engagement in computer science education. In *Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 131–152.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*.

Quizlet Inc. 2024. Quizlet. https://quizlet.com.

Qwen Team. 2025. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Tal Ridnik, Dedy Kredo, and Itamar Friedman. 2024. Code generation with alphacodium: From prompt engineering to flow engineering. *arXiv preprint arXiv:2401.08500*.

Michael Sailer and Lisa Homner. 2020. The gamification of learning: A meta-analysis. *Educational psychology review*, 32(1):77–112.

Valerie J Shute. 2008. Focus on formative feedback. *Review of educational research*, 78(1):153–189.

Valerie Jean Shute and Matthew Ventura. 2013. *Stealth assessment: Measuring and supporting learning in video games*. MIT press.

John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285.

Mudit Verma, Siddhant Bhambri, and Subbarao Kambhampati. 2024. On the brittle foundations of react prompting for agentic large language models. *arXiv preprint arXiv:2405.13966*.

Alf Inge Wang and Rabail Tahir. 2020. The effect of using kahoot! for learning–a literature review. *Computers & education*, 149:103818.

Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

Pieter Wouters, Christof Van Nimwegen, Herre Van Oostendorp, and Erik D Van Der Spek. 2013. A meta-analysis of the cognitive and motivational effects of serious games. *Journal of educational psychology*, 105(2):249.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First conference on language modeling*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Jiyuan Zeng, Daner Sun, Chee-Kit Looi, and Andy Chun Wai Fan. 2024. Exploring the impact of gamification on students' academic performance: A comprehensive meta-analysis of studies from the year 2008 to 2023. *British Journal of Educational Technology*, 55(6):2478–2502.

A Bloom's Mapping and Mechanic Contracts

Table 3 maps Bloom's levels to mechanics grounded in Anderson and Krathwohl (2001) and the LM-GM framework (Arnab et al., 2015); full Pydantic contracts are in the repository.

Bloom's	**Fam.**	**Mechanics**
Remember	ID	Click-to-Id., Memory Match
Understand	ID	Drag-Drop, Desc. Match
Apply	Both	Trace Path, Seq., State Tr.[A]
Analyze	Both	Sorting, Hier., Bug H.[A], Cmplx.[A]
Evaluate	ID	Compare, Branching
Create	Algo	Algo. Builder[A], Constr. Pzl.[A]

Table 3: Bloom's-to-mechanic mapping. Fam.: ID = Interactive Diagram, Algo = Algorithm, Both = shared; [A] = Algorithm Game mechanic.

B Per-Mechanic Results

Table 4 disaggregates VPR, token consumption, latency, and human ratings across all 15 mechanics. Fully constrained schemas achieve ≥90% VPR; the two lowest-performing mechanics (DESC_MATCH, CONSTR_PUZZLE) share root causes detailed in Appendix C. Mechanics with $N \leq 10$ should be interpreted with caution; per-mechanic 95% Wilson confidence intervals are available in the repository.

Mechanic	**N**	**VPR** (%)	**Tok.** (K)	**Lat.** (s)	**Edu.** (1–5)	**Play** (%)
Interactive Diagram Games						
DRAG_DROP	26	96.2	18.2	27.0	4.4	96.2
SEQUENCING	16	93.8	17.0	25.0	4.5	93.8
CLICK_TO_ID	14	92.9	16.8	24.0	4.3	92.9
SORTING	12	91.7	18.5	28.0	4.2	91.7
MEMORY_MATCH	12	91.7	16.2	23.0	4.3	91.7
BRANCHING	10	90.0	19.5	30.0	4.1	90.0
COMPARE	8	87.5	20.1	31.0	4.0	87.5
HIERARCHICAL	8	87.5	22.4	35.0	3.9	87.5
TRACE_PATH	14	85.7	17.5	26.0	4.1	85.7
DESC_MATCH	10	60.0	15.8	22.0	3.8	75.0
Interactive Algorithm Games						
STATE_TRACER	18	94.4	21.3	32.0	4.4	94.4
BUG_HUNTER	16	93.8	23.8	36.0	4.2	87.5
ALGO_BUILDER	14	92.9	25.2	38.0	4.3	92.9
COMPLEXITY	12	91.7	22.7	34.0	4.1	83.3
CONSTR_PUZZLE	10	80.0	26.5	40.0	3.9	80.0
Overall	**200**	**90.0**	**19.9**	**29.8**	**4.2**	**89.8**

Table 4: Per-mechanic metrics ($N = 200$, 15 mechanics). Edu./Play: mean human ratings from 5 blinded raters (ICC > 0.78). Algorithm Games average higher token consumption but lower per-game cost (no vision model calls). Mechanics with $N \leq 10$: interpret with caution; Wilson 95% CIs in repository.

C Failure Taxonomy

Of the 20 DAG failures across 200 questions, 14 occur in Interactive Diagram mechanics and 6 in Algorithm Games. The dominant root cause is **schema underspecification**, not LLM hallucination: generated content is factually correct but lacks structural fields required by the FOL-based contract validator. All failure types are tractable schema engineering problems currently under active remediation.

Mechanic	N	Gate	Error / Root Cause
Interactive Diagram Games (14 failures)			
`DESC_MATCH`	4	QG3	`BLOOM_OP_COUNT_FAIL`; pairs lack relational links
`TRACE_PATH`	2	QG3	`ANCHOR_OOB`; SVG coords outside bounding box
`COMPARE`	1	QG2	`ASSET_SCHEMA_MISMATCH`; axis label missing
`HIERARCHICAL`	1	QG3	`DEPTH_MISMATCH`; tree depth $<$ contract min
6 other ID mech.	6	QG2/3	Region overlap (2); state/schema violations (4)
Interactive Algorithm Games (6 failures)			
`CONSTR_PZL`	2	QG3	`CONSTRAINT_UNSAT`; FOL rules form unsat. set
`COMPLEXITY`	1	QG3	`CLASS_MISMATCH`; generated $\neq$ target class
3 other Algo mech.	3	QG2/3	Placement, ordering, state transition errors

Table 5: Failure taxonomy (20 failures, $N = 200$, 15 mechanics). All attributable to schema underspecification, not LLM hallucination. FOL-based validators catch structural violations deterministically.

D Mechanic-Specific Agent Prompting

Game Concept Designer (Phase 1) *DRAG_DROP · Analyze Level*

System Role

You are an expert educational game designer. Transform learning questions into GameConcept JSON: title, narrative, scenes, mechanics, and WHY each mechanic matches the learning objective. Focus on WHAT and WHY—visual design comes later.

Design Principles

Match mechanic to objective: `drag_drop`=spatial, `trace_path`=process, `sequencing`=temporal, `sorting`=classification. Scenes can chain 1–3 mechanics via `advance_trigger`. Zone-based mechanics share a diagram; content-only (`sequencing`, `memory_match`) set `needs_diagram=false`.

Task Prompt (assembled from state)

`## Question:` Label the parts of a plant cell.
`## Context:` Bloom's: analyze · Biology · intermediate
`## Domain Knowledge:` Labels: Chloroplast, Mitochondria, Cell Wall, Vacuole, Nucleus, Ribosome. Descriptions: {Chloroplast: "Conducts photosynthesis..."}. Needs: labels=true, sequence=false.
`## Capabilities: [drag_drop, click_to_identify, trace_path, sequencing, sorting, description_matching, memory_match, branching]`

Output (GameConcept, Pydantic)

```
{title, subject, difficulty, narrative_theme, all_zone_labels[], distractor_labels[], scenes[{title, learning_goal, zone_labels[], needs_diagram, mechanics[{mechanic_type, learning_purpose, expected_item_count, advance_trigger}]}]}
```

QG1 Retry Injection

`RETRY:` `scene_1` uses sequencing but no sequence data available; `distractor_labels` overlaps `all_zone_labels`; missing `estimated_duration_minutes`.

E Architectural Evolution: Pipeline Diagrams

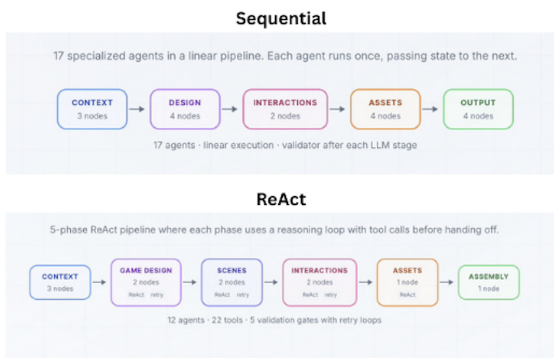

Figure 4: **Pipeline Evolution: Sequential and ReAct Architectures.** The Sequential design runs 17 specialised agents linearly with a validator after each LLM stage. The ReAct design uses a 5-phase reasoning loop with 12 agents, 22 tools, and 5 validation gates with retry loops.