PlayCompass: Recursive Self-Improvement for Game Generation with Visually Grounded Evolving Rubrics

Fuxiang Zhai[1] Sixiang Chen[1] Tengjun Huang[1] Lei Zhu[1,2]
[1]The Hong Kong University of Science and Technology (Guangzhou)
[2]The Hong Kong University of Science and Technology
fzhai598@connect.hkust-gz.edu.cn

Project page: https://fuxiangzhai.github.io/PlayCompass/

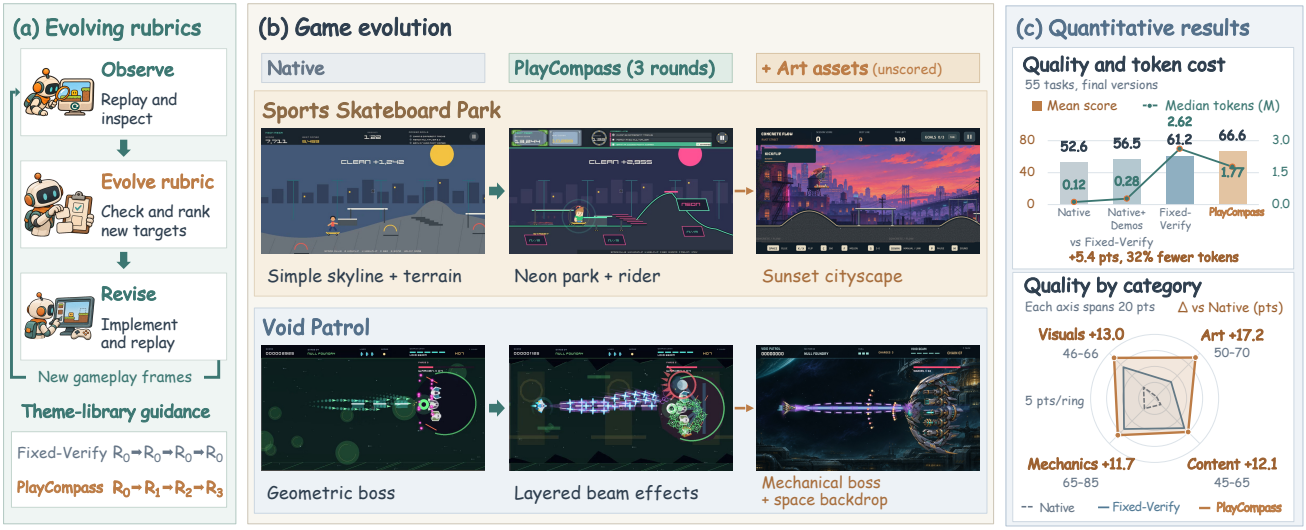


Figure 1. **PlayCompass improves games by evolving visually grounded rubrics.** **(a)** Rubrics evolve from replay frames and theme-library guidance to steer game revision. **(b)** Two Native games improve in just three PlayCompass rounds (rightmost versions add art assets and remain unscored). **(c)** Across 55 tasks, PlayCompass gains 14.0 points over Native and outperforms the fixed-rubric loop by 5.4 points with 32% lower median billable token use (task means over two development runs for PlayCompass and the fixed-rubric loop).

Abstract

*Coding agents can turn text briefs into playable games whose initial builds often lack content depth and visual polish. A fixed rubric can miss opportunities for improvement that emerge as a game changes, and short gameplay recordings can omit the states needed to verify an improvement. We present **PlayCompass**, a training-free framework for recursive game development that pairs a fixed anchor rubric for core requirements with an evolving rubric for improvement targets. Each round, PlayCompass uses gameplay frames from the current build to propose new targets and check which targets have been met. It ranks open targets using a theme library distilled from scored tasks outside the evaluation set and frozen before evaluation. After each accepted revision, showcase demos capture the core loop and changes to inform the next rubric update. On 55 GameCraft-Bench tasks with one model in every role, PlayCompass improves the native coding agent on every task, by 14.0 points on average, and outperforms a fixed-rubric loop by 5.4 points with 32% fewer tokens; human raters prefer its games on 25 of 30 tasks.*

1. Introduction

A game is judged by what players see on screen, not by what its code contains. Coding agents can generate playable games from natural-language briefs in real game engines [4, 8, 20, 24, 33, 37]. Their builds can offer lim-

ited content and rely on placeholder graphics. GameCraft-Bench [24] replays submitted input traces and scores the resulting gameplay recordings against hidden expert rubrics tailored to the intended experience of each game. On our 55 tasks in our study, the initial Codex submissions average 69.5/100 for core mechanics, 47.7/100 for content depth, and 50.8/100 for art and presentation (Tab. 1).

A self-improvement loop lets the agent play its current build, inspect the resulting gameplay, and use that evidence to guide code revisions [3, 17, 21, 27, 40, 41]. The loop needs its own criteria to decide what to change and check whether each revision meets its goal. The game brief sets the core requirements. A rubric written solely from the brief can miss opportunities for improvement that emerge as the game evolves. Verification also depends on which states a recording captures. A replay of the opening level offers little evidence about a mechanic that only appears in a later stage, even when it works correctly. How can the loop use gameplay evidence to update and verify its improvement targets while preserving the brief's core requirements?

We introduce **PlayCompass**, a training-free framework for recursive self-improvement that uses visual feedback to evolve the game and its target rubric together while keeping the brief's core requirements in a fixed anchor rubric (Fig. 2). The anchor guides repairs, and the loop uses gameplay frames from the current build to propose new targets and check which targets have been met. In our analysis, gameplay recordings more often provided sufficient evidence for art and presentation targets than for content targets, so we focus new proposals on that category. The loop prioritizes open targets using a frozen theme library distilled from public tasks outside the evaluation set. Within each source task, we compare higher-scoring and lower-scoring game versions to extract reusable lessons. The coding agent pursues the selected targets. For each newly retained build, a presenter creates showcase demos of the core gameplay loop and visible changes. Replays of these demos provide fresh visual evidence for the next rubric update.

On 55 GameCraft-Bench [24] tasks, PlayCompass improves official scores by 14.0 points on average over the initial Codex submissions. We compare PlayCompass with Fixed-Verify, a fixed-rubric refinement loop, using the same starting games and per-round coding budget over three development rounds. In this comparison, PlayCompass gains 5.4 points on average and reduces median billable token use per task by 32% (Tab. 1). PlayCompass leads on 40 of the 55 tasks, with the largest category gain in art and presentation ($+10.8$ points). In additional evaluations, PlayCompass scores higher on average than its ablated variants and the official implementations of RSIGame [40] and RGC [3] on 24 GameCraft-Bench tasks, gains 5.2 points on average over Fixed-Verify on 12 SWE-Game [4] tasks, and receives more preference votes than Fixed-Verify on 25 of 30 tasks in a blind video study.

Our main contributions are as follows:

- We introduce PlayCompass, a training-free framework for recursive game development that evolves the game and its target rubric together. A fixed anchor rubric guides checks and repairs for the brief's core requirements. Each round, the loop uses current gameplay frames to propose targets and verify which targets have been met.
- We construct a frozen theme library by comparing higher-scoring and lower-scoring versions of the same public tasks. We distill their score differences and judge rationales into reusable design lessons that guide target proposal and ranking on new games.
- We show that PlayCompass outscores native on all 55 GameCraft-Bench tasks, with a mean gain of 14.0 points. Against Fixed-Verify, a fixed-rubric loop with matched starting builds and per-round coding budgets, PlayCompass gains 5.4 points on average while reducing median billable token use per task.
- We find that per-round target updates and cross-task themes each improve scores in component ablations. PlayCompass also outscores the official implementations of RSIGame and RGC in our shared three-round GameCraft-Bench setup and improves on Fixed-Verify on 12 SWE-Game [4] tasks.

2. Related Work

2.1. Agentic game generation

Benchmarks for agentic game development cover targeted edits to existing projects [5] and the construction of games from textual specifications [4, 10, 20, 26, 31, 34, 44]. GameCraft-Bench [24] evaluates Godot projects through replayed gameplay. For each task, this protocol lets us compare initial and refined builds under the same official rubric.

In iterative development, gameplay evidence also guides what the agent changes next. Gameplay recordings and GUI playtests capture in-game behavior [17, 21], and runtime state injection lets verifiers test specific states directly [19]. A planner can use independent QA evidence to choose the next development increment [41]. RSIGame [40] uses an evolving checklist of issues, improvement opportunities, priorities, and verification outcomes to choose the next development direction. Recursive Game Creator (RGC) [3] uses gameplay trajectories and rendered observations to assess each build against game-specific criteria, recommend revisions, and select versions. In PlayCompass, a fixed anchor rubric guides checks and repairs for the brief's requirements. Each round, the loop uses gameplay frames to propose new targets and check existing ones in a separate target rubric, then ranks open targets for subsequent revisions.



Figure 2. **PlayCompass evolves the game and its target rubric together.** Gameplay frames guide target proposal and review. A frozen theme library guides target proposal and ranking. The designer turns selected criteria into a development plan and an acceptance contract that specifies what should appear on screen and how to reach it. Showcase replays from the retained build provide fresh visual evidence for the next rubric update (bottom arrow). In round 3 of `rhythm-garden`, similar result cards (left) motivate target #11 (center). Water Waltz and Birdsong Echo replays (right) show distinct result-card illustrations in at least two activities, satisfying the criterion.

2.2. Visual feedback and gameplay evidence

Visual feedback can guide revisions to generated media and to the code that produces visual artifacts. In image and video generation, feedback from candidate outputs guides prompt revisions or direct edits to the generated content [23, 39, 42]. For layouts, 3D scenes, and webpages produced by code, rendered outputs can guide targeted code revisions or provide a training signal [13, 15, 16, 22, 30, 32].

In game development, visual observations from interactive tests help diagnose failures and guide repairs to generated builds [28, 33]. What a gameplay recording reveals depends on the actions taken during play. A replay may reach the relevant state and still end before the specific player interaction needed to assess a target, leaving the reviewer with insufficient evidence. To compare behavior across versions, PlayCompass replays selected demos from the previous build with identical inputs. For each newly retained build, a presenter also scripts new showcase demos that show the core gameplay loop and target-related changes, providing fresh evidence for the next round of target checks.

2.3. Evolving rubrics and target prioritization

Rubrics specify explicit criteria for evaluating open-ended outputs [1]. Text-to-image evaluators translate prompt requirements into questions about objects, attributes, and relations [6, 14, 18]. VideoArgus [43] constructs a rubric for each input and applies it to all corresponding candidate videos to enable consistent comparisons. Criterion-level judgments can also be combined into rewards for reinforcement learning in language and image generation [9, 12].

Eureka [25] uses feedback from policy training to revise reward programs, and evolving-rubric methods adapt evaluation criteria during language model policy optimization [7, 11, 29, 36, 38]. RuHarness [2] updates behavioral criteria from execution trajectories and prioritizes unresolved weaknesses to guide harness revisions while keeping the task objective fixed. PlayCompass uses gameplay evidence to update a separate rubric of visually verifiable targets for the next game revision. A fixed anchor rubric guides checks and repairs for the brief's requirements.

When several targets remain open, experience from other tasks can help decide which to pursue first. Voyager [35] reuses executable skills, and ExpeL [45] distills transferable lessons from past trajectories, including comparisons of successful and failed attempts on the same task. PlayCompass compares higher-scoring and lower-scoring game versions within public tasks outside the evaluation set and distills their score differences and judge rationales into a frozen theme library. The loop combines these theme priors with category weights and each target's current review verdict to rank open targets for subsequent revisions.

3. Method

Task and official score. Each GameCraft-Bench [24] task provides a text brief b, a Godot workspace, and shared 2D assets. A submission consists of a game project g and a

set $\mathcal{D}$ of up to ten gameplay input traces (demos). Let $\mathrm{BUILD}(g, \mathcal{D}) \in \{0, 1\}$ indicate whether g launches and at least one demo parses. The verifier replays each valid trace τ in a fresh Godot instance, recording up to 20 s as $\rho(g, \tau)$, and a multimodal judge scores the recordings against a hidden expert rubric, yielding category scores $s_k(g, \mathcal{D}) \in [0, 1]$ and the official score

$$S(g, \mathcal{D}) = \begin{cases} \sum_k w_k \, s_k(g, \mathcal{D}), & \mathrm{BUILD}(g, \mathcal{D}) = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The public weights $w = (0.15, 0.35, 0.15, 0.35)$ correspond to core mechanics, content depth, functional visuals, and art and presentation. The judge averages persistent properties over demos but takes the maximum for scenario-specific requirements [24], which thus earn credit only if some demo reaches the relevant state. Refinement never accesses the hidden rubric or official scores.

Overview. Starting from a Codex submission $(g_0, \mathcal{D}_0)$, PlayCompass refines the game and its demos over R rounds without training (Fig. 2). It keeps a fixed anchor rubric $\mathcal{R}_0$ for the brief's requirements, open targets $\mathcal{T}_r$, fulfilled targets $\mathcal{F}_r$, and a frozen theme library Θ. With $\mathcal{T}_0 = \mathcal{F}_0 = \varnothing$ and round-zero evidence E_0, round r performs

$$\begin{aligned} (\mathcal{T}_r, \mathcal{F}_r) &= \textsc{Evolve}(\mathcal{T}_{r-1}, \mathcal{F}_{r-1}, E_{r-1}; \mathcal{R}_0, \Theta), \\ \tilde{g}_r &= \textsc{Develop}(g_{r-1}, \mathcal{R}_0, \mathcal{T}_r, E_{r-1}), \\ g_r &= \textsc{Keep}(g_{r-1}, \tilde{g}_r), \\ (\mathcal{D}_r, E_r) &= \textsc{Observe}(g_r; \mathcal{T}_r, \mathcal{F}_r). \end{aligned} \quad (2)$$

If KEEP retains g_{r-1}, we reuse $\mathcal{D}_{r-1}$ and E_{r-1}, and we finally submit $(g_R, \mathcal{D}_R)$. All roles share one model: a target proposer and a reviewer maintain the rubric, a designer plans rounds, Codex edits the game, a QA checker tests revisions, and a presenter writes showcase demos.

3.1. Gameplay evidence for refinement

For each newly retained build g_r, OBSERVE collects sandbox frames and runtime logs $\sigma(g_r, \tau)$ and benchmark replay recordings $\rho(g_r, \tau)$ of traces τ:

$$\begin{aligned} E_r = &\{\sigma(g_r, \tau) : \tau \in P \cup \Pi_r\} \\ &\cup \{\rho(g_r, \tau) : \tau \in \mathcal{D}_r \cup \bar{\mathcal{D}}_{r-1}\}. \end{aligned} \quad (3)$$

The four probes P are traces scripted from b in round zero and replayed on every build. A player policy, also written in round zero, reads game-reported state to produce six episodes Π_r without per-action model calls; this state never counts as evidence. The demo set $\mathcal{D}_r$ holds up to five showcase demos for g_r, followed by inherited Codex demos that still replay, and $\bar{\mathcal{D}}_{r-1}$ replays the previous build's first three showcase demos to compare builds under identical inputs.

The presenter scripts showcase demos from the brief, source code, screenshots, and target highlights: the first shows the core loop, and later ones show up to six implemented highlights in normal play. The four consecutive frames with the largest pixel change in each showcase recording form a *motion strip*, and probes fill $\mathcal{D}_R$ up to the ten-trace limit for the final submission.

An internal vision-language model (VLM) reviewer checks each criterion c using only frames and logs from E (at most 24 of about 700 frames per call). It returns a verdict $y(c; E) \in \{\mathsf{pass}, \mathsf{fail}, \varnothing\}$. A pass requires observed evidence that the game satisfies c, a fail requires an observed violation, and $\varnothing$ denotes insufficient evidence for either. Each pass or fail cites supporting frames or log lines. PlayCompass uses this *lean* review by default; in *full review*, used by the verified baselines (Sec. 4.1) and an eight-task pilot of PlayCompass, an evidence agent also writes up to three traces per round to reach states of criteria marked $\varnothing$, so that the reviewer can re-check them.

3.2. Evolving the target rubric

Anchor rubric $\mathcal{R}_0$. In round zero, we derive $\mathcal{R}_0$ (at most 16 criteria) from b, assign each criterion a public category, mark those the brief makes mandatory, and check g_0 using E_0. $\mathcal{R}_0$ stays fixed in PlayCompass and the fixed-rubric baseline, and its core-mechanics and mandatory criteria form $\mathcal{R}_0^{\mathrm{core}}$, any fail of which triggers repair (Eq. (7)).

Theme library Θ (across tasks). We build Θ offline from public tasks disjoint from the evaluation set, pairing officially scored versions $p = (g^+, g^-)$ of the same task whose scores differ by at least 5 points (90 pairs from 23 tasks). For each pair, we retain up to three requirements with the largest positive gaps from g^- to g^+, weighted by their contribution to S, with the judge's rationales for both versions. A meta agent distills these comparisons into 23 reusable, genre-tagged themes grouped by category, each traceable to its supporting pairs.

For pair p, let $\Delta_k(p)$ be the sum of its retained weighted gaps in category k. For a theme h in category k, let $\mathcal{M}_h$ contain its supporting pairs and $\mathcal{M}_k$ all pairs with a retained requirement in k. The theme prior is

$$\theta(h) = \mathrm{clip}_{[0.5,\,1.5]}\left(\frac{\mathrm{mean}_{p \in \mathcal{M}_h} \Delta_k(p)}{\mathrm{mean}_{p \in \mathcal{M}_k} \Delta_k(p)}\right). \quad (4)$$

A prior above 1 means that the supporting pairs' mean gap exceeds the category baseline, and clipping limits the influence of themes supported by few pairs. We validate and freeze the library before evaluation (Sec. 4.5).

Target rubric (within a task). A *target* specifies an improvement verifiable in normal play. Before round r, the target proposer suggests up to four targets $\mathcal{N}_r$ from the frames of g_{r-1} in E_{r-1}, with showcase frames first. It applies genre-matched themes that fit b and address gaps visible in the frames. Each proposal specifies a visible outcome, where to verify it, its source (frames, brief, or theme),

and an optional theme h_t; debug overlays, test-only hooks, and content removal are excluded. PlayCompass proposes targets only for art and presentation, whose changes recordings verify far more often than content changes (Sec. 4.5); other categories rely on anchor criteria.

$\textsc{Dedup}_\delta$ drops proposals whose word-level Jaccard similarity to an anchor criterion, an earlier target, or an earlier retained proposal is at least $\delta = 0.5$, and the reviewer checks the rest and the open targets against E_{r-1}:

$$\begin{aligned} \mathcal{C}_r &= \mathcal{T}_{r-1} \cup \textsc{Dedup}_\delta\left(\mathcal{N}_r; \mathcal{R}_0 \cup \mathcal{T}_{r-1} \cup \mathcal{F}_{r-1}\right), \\ \mathcal{F}_r &= \mathcal{F}_{r-1} \cup \left\{t \in \mathcal{C}_r : y(t; E_{r-1}) = \textsf{pass}\right\}, \\ \mathcal{T}_r &= \operatorname{top}_K^{\psi}\left(\mathcal{C}_r \setminus \mathcal{F}_r\right). \end{aligned} \tag{5}$$

The open set keeps up to $K = 8$ targets marked fail or $\varnothing$, ranked by

$$\psi(t) = \theta(h_t)\, w_{k_t}\, \gamma(t), \qquad \gamma(t) = \begin{cases} 1, & y(t; E_{r-1}) = \textsf{fail}, \\ 0.7, & y(t; E_{r-1}) = \varnothing, \end{cases} \tag{6}$$

where k_t is the category of t and $\theta(h_t) = 1$ for untagged targets. The designer sees open targets in this order; w_{k_t} is constant for PlayCompass, which proposes only art targets, but matters for variants with content targets (Sec. 4.4). Targets beyond the top K are dropped but may be proposed again, and fulfilled targets are not re-checked; instead, they supply highlights for later showcase demos.

3.3. Recursive game development

Direction. Each round repairs the build or extends one category $k \in \{\textsf{art}, \textsf{content}\}$, where $n_k(r)$ counts earlier extensions of k. The preferred direction is

$$\hat{d}_r = \begin{cases} \textsf{repair}, & \exists\, c \in \mathcal{R}_0^{\text{core}} : y(c; E_{r-1}) = \textsf{fail}, \\ \textsf{art}, & \text{else if } n_{\textsf{art}}(r) \le n_{\textsf{content}}(r), \\ \textsf{content}, & \text{otherwise}. \end{cases} \tag{7}$$

Repairs take precedence, and extensions alternate, starting with art. A category is eligible if it has an open target or an anchor criterion marked fail or $\varnothing$. The round direction d_r falls back to the other category if $\hat{d}_r$ is ineligible, and to deepening passed content criteria (or passed art criteria if none exist) if neither category is eligible.

Develop. Given d_r, the designer selects criteria $\mathcal{R}_r^\star \subseteq \mathcal{R}_0 \cup \mathcal{T}_r$ (anchor failures for a repair; eligible anchor criteria and top-ranked targets of the category for an extension) and writes an *acceptance contract* of three to eight checks, each naming an onscreen outcome and the inputs that reach it. Codex edits a sandboxed copy of g_{r-1} and writes a QA trace per check in up to three sessions sharing a 20-minute budget. After each session, the QA checker builds the game, replays the traces, and marks each check *done*, *not done*, or *not visible*. Editing stops when all checks are done, a session changes nothing or yields no report, or a limit is reached, and the last candidate is $\tilde{g}_r$.

Version retention. KEEP sets $g_r = \tilde{g}_r$ if the candidate passes build and runtime checks and differs from g_{r-1} (project hash), and keeps g_{r-1} otherwise. PlayCompass thus retains valid changes that the reviewer cannot yet confirm, which verified acceptance would reject at a cost in official score (Sec. 4.5). Verified baselines accept a candidate only after rubric checks and blind pairwise comparisons with the incumbent in both presentation orders.

4. Experiments

4.1. Setup

Tasks. We split the 140 GameCraft-Bench [24] tasks into disjoint pools for main evaluation, development (six tasks), and theme-library construction and validation. The 55 main tasks span all 15 genres and the native-score tertiles (20 low, 18 middle, 17 high); a 24-task ablation subset is drawn by a fixed hash within each tertile. We configured PlayCompass after the verified loops had run on 29 of these tasks (Sec. 4.5) and also report the other 26.

Baselines. **Native** is the initial Codex submission $(g_0, \mathcal{D}_0)$ from the official harness. **Native+Demos** keeps g_0 but submits presenter-authored showcase demos first, isolating the effect of demos. **Fixed-Verify**, our primary iterative baseline, uses the same evidence with the fixed anchor rubric $\mathcal{R}_0$. It repairs on core fails and otherwise extends whichever of content depth and art and presentation has the larger weighted gap in $\mathcal{R}_0$. Following RGC [3], it accepts a candidate only after rubric checks and blind pairwise comparisons with the incumbent in both presentation orders, and it uses full review. Both loops share g_0, the frozen round-zero package (anchor rubric, probes, player policy, and initial evidence), the presenter, and the asset library, and run three rounds with a 20-minute Codex budget each; Sec. 4.4 separates their differences in rubric, scheduling, and acceptance. On the ablation subset, we also run the official RSIGame [40] and RGC code once per task from g_0 for three rounds and score each export with its own demos.

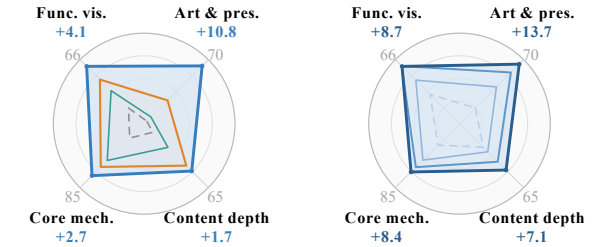
Evaluation. The official verifier records demo replays in a network-isolated sandbox, and the unmodified upstream scorer applies the hidden rubric with GPT-5.6. All agents use GPT-5.6-sol without image or 3D generation. We run PlayCompass and Fixed-Verify twice per task (Evolving-Verify twice on the 29 design tasks; other methods once); repeated runs share the native game and round-zero package but make fresh model calls. Each export is scored twice on the $100\,S$ scale (Eq. (1)), using two independent recordings for real-time tasks and two passes over one recording otherwise. We report paired task-level differences with 95% bootstrap CIs (10,000 resamples), two-sided Wilcoxon tests, and win/tie/loss (W/T/L) counts.

Table 1. **Main results on 55 GameCraft-Bench tasks.** Official scores (points) are task means, with category weights in parentheses. Green: gain of PlayCompass over Fixed-Verify. Δ: mean paired difference in the overall score, with a 95% bootstrap CI over tasks for the comparison with Fixed-Verify. Tok.: median billable tokens per task. h: median wall-clock hours per run. All methods use GPT-5.6-sol and start from the Native game.

		Core mech.	Content depth	Func. visuals	Art & pres.	Overall Δ			
Method	Overall	(M, 0.15)	(D, 0.35)	(V, 0.15)	(A, 0.35)	vs. Native	vs. Fixed-Verify [95% CI]	Tok.	h
Native (Codex)	52.6	69.5	47.7	50.9	50.8	–	–	0.12M	–
Native+Demos (our demos)	56.5	76.5	52.4	56.3	52.1	+4.0	–	0.28M	0.7
Fixed-Verify (fixed rubric)	61.2	78.5	58.1	59.8	57.3	+8.6	–	2.62M	2.5
PlayCompass (ours)	**66.6** +5.4	**81.2** +2.7	**59.8** +1.7	**63.9** +4.1	**68.1** +10.8	**+14.0**	**+5.4** [3.1, 7.9]	1.77M	1.7

Table 2. **Robustness of the gain** of PlayCompass over Fixed-Verify (score, points). n: tasks. W/T/L: tasks on which PlayCompass is better, tied, or worse. Bold: 95% CI excludes 0.

	Setting	n	Δ [95% CI]	W/T/L
	Main comparison (Tab. 1)	55	**+5.4** [3.1, 7.9]	40/0/15
	Outside the design tasks	26	**+6.0** [2.8, 9.7]	19/0/7
Tertile	low native score	20	**+9.3** [4.7, 14.6]	16/0/4
	mid native score	18	+3.2 [−0.3, 6.8]	10/0/8
	high native score	17	**+3.0** [1.4, 4.7]	14/0/3
Weights	equal category weights	55	**+4.8** [2.6, 7.4]	40/0/15
	without art and pres.	55	**+2.5** [0.2, 5.0]	34/0/21
Judge	Gemini-2.5-Flash	55	**+3.3** [0.9, 5.8]	36/1/18
	Gemini-3-Flash	55	+2.3 [−0.02, 4.7]	33/0/22
Humans	blind preference	30	–	25/3/2
Benchmark	SWE-Game	12	**+5.2** [0.2, 10.2]	9/0/3



(a) Final versions, 55 tasks (b) PlayCompass by round, 24 tasks

Figure 3. **Category profiles** (task means, points; each axis spans 20 points, with the rim value in gray). (a) Final versions on the 55 main tasks; numbers give PlayCompass −Fixed-Verify. (b) PlayCompass after each round on the 24 ablation tasks; numbers give final−start.

4.2. Main results

PlayCompass improves every game and outperforms the fixed-rubric loop. Over three rounds, PlayCompass raises the mean official score from 52.6 to 66.6 and improves on all 55 tasks (Tab. 1). Against Fixed-Verify, it achieves a mean paired gain of 5.4 points (95% CI [3.1, 7.9]) and scores higher on 40 of the 55 tasks and in 12 of the 15 genres. The gain concentrates in art and presentation (+10.8 points, Fig. 3a), where PlayCompass proposes targets, while the content-depth gain (+1.7, 95% CI [−0.9, 4.3]) is not significant (Sec. 4.5).

Development adds to the demo gain. Because the judge takes the maximum over demos for scenario-specific requirements, demos alone change the score: with the game unchanged, Native+Demos gains 4.0 points over Native (95% CI [2.8, 5.3]). Relative to Native+Demos, development adds 4.6 points for Fixed-Verify and 10.0 points for PlayCompass (95% CIs [3.2, 6.2] and [7.8, 12.4]). These gains reflect both games and demos; Sec. 4.3 compares the games under identical inputs.

The gain holds across tasks, weights, and judges. The gain is positive in every native-score tertile and largest in the lowest (+9.3 points, Tab. 2), where initial builds leave the most room for improvement. On the 26 tasks not used to configure PlayCompass, it is 6.0 points [2.8, 9.7], and excluding art and presentation still leaves 2.5 points [0.2, 5.0]. Judges from another model family score all versions higher, compressing differences: the gain shrinks to 3.3 points [0.9, 5.8] with Gemini-2.5-Flash and 2.3 points [−0.02, 4.7] with Gemini-3-Flash, but both keep a significant art gain. On 12 SWE-Game tasks [4], which weight required functionality at 85%, the gain is 5.2 points [0.2, 10.2] (Wilcoxon $p = 0.09$).

PlayCompass uses 32% fewer tokens. Per task, PlayCompass uses a median of 1.77M billable tokens and 1.7 hours, versus 2.62M and 2.5 hours for Fixed-Verify (Tab. 1). The savings come from verification rather than the rubric: reviews, pairwise comparisons, and evidence collection cost 0.48M versus 1.39M tokens per run, because PlayCompass keeps every valid build and uses lean review, while Codex editing costs are comparable.

PlayCompass outperforms RSIGame and RGC in our setting. On the ablation subset (Tab. 3), PlayCompass exceeds RSIGame by 11.7 points [8.6, 15.1] and RGC by 12.5 points [9.4, 15.7], scoring higher on 23 and all 24 tasks, with gains in all four categories (12.2 and 11.8 points in art and presentation). Both run their official code with our model, start, budget, and judge, so the comparison reflects

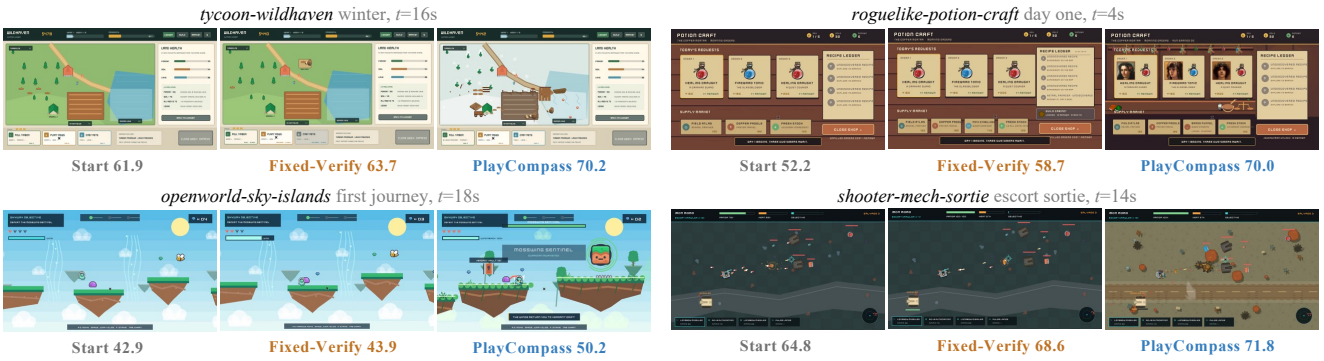


Figure 4. **Final versions under identical inputs.** For each task, one showcase demo of the shared start (the native game with the presenter's demos) is replayed on the final versions of Fixed-Verify and PlayCompass and cut at the same time; numbers are official scores. On the same screens, PlayCompass adds a winter scene with a dock (`tycoon-wildhaven`), customer portraits (`roguelike-potion-craft`), planted islands with their guardian (`openworld-sky-islands`), and desert terrain (`shooter-mech-sortie`). Figure 1b shows the rounds of PlayCompass.

our setting, not their original setups. Under their own rules, RSIGame keeps its starting version on 16 of the 24 tasks, and RGC gains only 2.3 points $[-0.9, 5.8]$.

4.3. Qualitative results and user study

Final builds differ visibly under identical inputs. Because Fig. 4 replays the same showcase demo of the shared start on both final builds and cuts at the same time, the added scene art and illustrated interface elements come from the games rather than from the demos written for each version.

Human raters prefer PlayCompass. In a blind side-by-side study, 24 volunteer raters compared showcase videos of the first-run final builds of PlayCompass and Fixed-Verify on 30 tasks, two per genre, drawn by a fixed hash before any video was watched. Each task received four judgments in balanced left–right order, and raters chose a side or no clear difference. By task-level majority, raters preferred PlayCompass on 25 tasks and Fixed-Verify on two, with three ties (sign test, $p = 5.6 \times 10^{-6}$); over 120 judgments, PlayCompass won 88 to 9 with 23 ties (Krippendorff's $\alpha = 0.78$). On five-point scales, PlayCompass was also rated higher for visual clarity ($+0.42$) and playability ($+0.24$), with slightly fewer bugs reported (-0.12). Since each side shows its own showcase demos, these preferences concern the games as presented to players, rather than under identical inputs as in Fig. 4.

4.4. Ablations

All variants in Tab. 3 share the native games and budget of PlayCompass. ours-text-feedback hides frames from development agents but keeps the presenter; ours-no-evolution uses only the anchor rubric; ours-static-rubric proposes targets once; ours-no-themes drops the theme library; and Evolving-Verify adds content and art targets to Fixed-Verify, keeping its scheduling and acceptance.

Table 3. **Ablations** on the 24 ablation tasks: variant minus PlayCompass (points; 95% CI for the overall score), where PlayCompass scores 66.4. Frames: agents see recorded frames (✓) or only logs, state, and source (✗). Rubric: anchor only (fixed), evolving targets with or without themes, or targets proposed once (static). Keep: every valid build (all) or verified builds only (verif.). Middle: components of PlayCompass. Bottom: other loops, including the official RSIGame and RGC code (one run each); Evolving-Verify, Fixed-Verify, RSIGame, and RGC are compared on main-experiment scores, where PlayCompass scores 67.0. Bold: the CI excludes 0.

Arm	Frames	Rubric	Keep	Δ overall	Δ art
PlayCompass	✓	$\mathcal{T}_r+\Theta$	all	0	0
ours-text-feedback	✗	$\mathcal{T}_r+\Theta$	all	**−6.1** [−8.6, −3.6]	**−6.0**
ours-no-evolution	✓	fixed	all	**−4.7** [−8.0, −1.5]	**−6.8**
ours-static-rubric	✓	static	all	**−4.8** [−7.7, −2.0]	**−7.1**
ours-no-themes	✓	$\mathcal{T}_r$	all	**−4.6** [−7.1, −2.3]	**−6.8**
Evolving-Verify	✓	$\mathcal{T}_r+\Theta$	verif.	**−6.4** [−10.2, −3.0]	**−10.8**
Fixed-Verify	✓	fixed	verif.	**−7.8** [−12.1, −4.0]	**−12.3**
RSIGame	official code			**−11.7** [−15.1, −8.6]	−12.2
RGC	official code			**−12.5** [−15.7, −9.4]	−11.8
Native+Demos	no development			**−9.8** [−12.8, −7.2]	**−13.7**

Visual feedback improves all four categories. Without frames, the score drops by 6.1 points [3.6, 8.6] and PlayCompass wins on 20 of 24 tasks. All four categories fall, by 4.1 (M), 6.7 (D), 6.6 (V), and 6.0 (A) points, suggesting that frames help the designer and QA checker beyond targets.

Targets must be re-grounded on each build. Removing targets, proposing them once, or dropping themes lowers the score by 4.6–4.8 points (art: 6.8–7.1). One-shot targets are no better than none, so the gain comes from re-grounding targets on the frames of each new build. Since targets also guide the presenter, these gaps reflect changes to both the games and their demos.

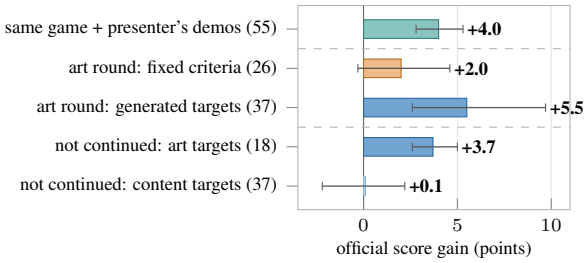


Figure 5. **What the loop sees and targets determines what it gains.** Mean official score gains (points, 95% CIs; counts in parentheses). Top: the unchanged native game with the presenter's demos (tasks). Bottom: single rounds relative to the build they extend, grouped by what they pursued (rounds): art rounds of Fixed-Verify (fixed criteria) and Evolving-Verify (generated targets), then rejected (not continued) Evolving-Verify candidates.

Retention and target evolution both matter. On these tasks, PlayCompass leads Evolving-Verify by 6.4 points (95% CI [3.0, 10.2]) and Fixed-Verify by 7.8 points (95% CI [4.0, 12.1]). Evolving targets recover only 1.4 of these points in the verified loop but are worth 4.7 points in PlayCompass, so they pay off when the loop keeps the builds that pursue them. Internal reviews marked 65.9% of selected art targets as achieved in 110 PlayCompass runs, but only 25% in 58 Evolving-Verify runs, which achieve 1.0 targets per run on average versus 4.0 for PlayCompass.

4.5. Analysis

We examine why the gains follow the evidence that recordings provide. For each saved candidate of the verified loops, including rejected ones, the round gain is its official score minus that of the build entering the round, which Fig. 5 contrasts with the demo-only gain of Native+Demos.

Gains accumulate over rounds. On the ablation tasks, the mean score of PlayCompass rises from 56.6 at round zero (Native+Demos) to 60.7, 64.4, and 66.4 after rounds 1–3 (one run per task, single-pass scoring, Fig. 3b). Every final build beats its round-zero version, by 9.8 points on average [7.2, 12.8], led by art (+13.7) and content depth (+7.1). Per-round gains shrink from 4.1 to 3.7 and 2.0 points but stay positive, so further rounds may still help.

Recordings verify art targets far more often than content targets. Across 58 Evolving-Verify runs on 29 tasks, the reviewer marked 18% of art-target checks and 62% of content-target checks as *insufficient evidence*. More specific targets widen rather than close the gap: in a six-task development comparison, they raised these rates from 15% and 52% to 31% and 76%. Content often lives in states that a 20-second recording of a fresh game rarely reaches, whereas art is visible on almost every screen, which motivates proposing only art targets while content relies on anchor criteria.

Verified acceptance discards useful art changes. Rejected Evolving-Verify candidates still gained 3.7 official points on average (95% CI [2.6, 5.0]) across 18 art-target rounds, but only 0.1 points [−2.2, 2.2] across 37 content-target rounds (Fig. 5). Gating on the reviewer thus rejects real art improvements, so PlayCompass keeps every valid build. Where PlayCompass trails Fixed-Verify most, it still leads in art but misses content states, such as a second arena or a landed catch, that no 20-second demo reaches.

Gains are not fitted to the internal reviewer. The final builds of PlayCompass have a 9.5-point lower internal art pass rate than those of Fixed-Verify, since lean review runs without re-checks, yet they score higher on the official art requirements. No arm uses an image generator, so the added art is drawn through the engine or taken from the asset library that all arms share.

Themes transfer lessons, not task details. On six held-out public tasks, a theme matched the better version's advantage in 19 of 24 pairs, but in one pair after swapping the judge's rationales. In PlayCompass, 39% of theme-derived proposals were achieved, versus 24% of those from frames or the brief, and targets are less similar to the hidden art requirements than the brief-derived anchor rubric (TF-IDF 0.08 vs. 0.14), indicating no task-level leakage.

5. Conclusion

We presented PlayCompass, a training-free framework that refines a game together with its target rubric: a fixed anchor rubric preserves the brief's requirements, while targets proposed and verified on each build's gameplay frames and ranked by a frozen theme library decide what to improve next. On 55 GameCraft-Bench tasks, PlayCompass improves every native game, gains 5.4 points on average over a fixed-rubric loop with 32% fewer billable tokens, and is preferred by human raters on 25 of 30 tasks. Our analysis suggests that self-improving loops gain most where their evidence can show success, and that keeping only revisions an internal reviewer can confirm discards real gains.

Limitations. We study 2D Godot games over three rounds with one agent backbone, and the official judge shares its model family. Content gains remain small because many content changes appear only in states that short recordings of a fresh game rarely reach; target-specific probes or state injection that reach these states are a natural next step.

References

[1] Hao Chen, Ziyu Han, Yukun Yan, Qingfu Zhu, Maosong Sun, and Wanxiang Che. From holistic evaluation to structured criteria: Rubrics across the evolving llm landscape. *arXiv preprint arXiv:2606.08625*, 2026. 3

[2] Hao Chen, Yukun Yan, Ziyu Han, Ye He, Chuanzhe Guo, Shangzhan Li, Qi Shi, Qingfu Zhu, Maosong Sun, and Wanxiang Che. When the critic falls behind: Co-evolving

rubrics and harnesses for continual agent self-improvement. *arXiv preprint arXiv:2610.36826*, 2026. 3

[3] Jiajun Chen, Haoyu Wu, Mingda Jia, and Xihui Liu. Recursive game creator: An agentic product-level experience-oriented game harness. *arXiv preprint arXiv:2610.08621*, 2026. 2, 5

[4] Xiaoyu Chen, Lai Wei, Jin Wang, Xiangyu Zou, Ruochen Fan, Enze Luo, Mingzhe Yao, Jiahui Zhu, Yuhua Wen, Linghe Kong, et al. Swe-game: Can coding agents build the games we want? *arXiv preprint arXiv:2609.33678*, 2026. 1, 2, 6

[5] Wayne Chi, Yixiong Fang, Arnav Yayavaram, Siddharth Yayavaram, Seth Karten, Qiuhong Anna Wei, Runkun Chen, Alexander Wang, Valerie Chen, Ameet Talwalkar, et al. Gamedevbench: Evaluating agentic capabilities through game development. *arXiv preprint arXiv:2602.11103*, 2026. 2

[6] Jaemin Cho, Yushi Hu, Jason Baldridge, Roopal Garg, Peter Anderson, Ranjay Krishna, Mohit Bansal, Jordi Pont-Tuset, and Su Wang. Davidsonian scene graph: Improving reliability in fine-grained evaluation for text-to-image generation. In *International conference on learning representations*, pages 15625–15645, 2024. 3

[7] Hongxin Ding, Baixiang Huang, Yue Fang, Weibin Liao, Zheng Li, Jinyang Zhang, Zhijing Wu, Junfeng Zhao, and Yasha Wang. Evorubrics: Dynamic rubrics as rewards via adversarial co-evolution for llm reinforcement learning. *arXiv preprint arXiv:2606.23038*, 2026. 3

[8] Zixuan Dong, Yanghai Wang, Hanyan Bian, Jiaxi Tao, Junze Li, Jiadong Wang, Xinyu Che, Lin Liu, Baoyun Peng, Xiaodong Wang, et al. From code to playable systems: Cegame-bench for long-horizon game creation agents. *arXiv preprint arXiv:2610.26098*, 2026. 1

[9] Xuelu Feng, Yunsheng Li, Ziyu Wan, Zixuan Gao, Junsong Yuan, Dongdong Chen, and Chunming Qiao. Rubricrl: Simple generalizable rewards for text-to-image generation. In *European Conference on Computer Vision*, pages 131–149. Springer, 2026. 3

[10] Roberto Gallotta, Graham Todd, Marvin Zammit, Sam Earle, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Large language models and games: A survey and roadmap. *IEEE Transactions on Games*, 2024. 2

[11] Xin Guan, Xiaomeng Hu, Shen Huang, Zhenyi Wang, Bo Zhang, Zijian Li, Pengjun Xie, Bo Liu, and Jiuxin Cao. Evorubric: Self-evolving rubric-driven rl for open-ended generation. *arXiv preprint arXiv:2605.29847*, 2026. 3

[12] Anisha Gunjal, Anthony Wang, Elaine Lau, Vaskar Nath, Yunzhong He, Bing Liu, and Sean Hendryx. Rubrics as rewards: Reinforcement learning beyond verifiable domains. In *International Conference on Learning Representations*, pages 127924–127945, 2026. 3

[13] Junrong Guo, Shancheng Fang, Yadong Qu, and Hongtao Xie. Seeing is improving: Visual feedback for iterative text layout refinement. *arXiv preprint arXiv:2603.22187*, 2026. 3

[14] Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. In *2023 ieee/cvf international conference on computer vision (iccv)*, pages 20349–20360. IEEE, 2023. 3

[15] Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *ICML*, pages 19252–19282, 2024. 3

[16] Ian Huang, Guandao Yang, and Leonidas Guibas. Blenderalchemy: Editing 3d graphics with vision-language models. In *European Conference on Computer Vision*, pages 297–314. Springer, 2024. 3

[17] Yixu Huang, Bo Li, Li Na, Zhe Wang, Kaijie Chen, Haonan Ge, Qingyi Si, Yuanzhe Shen, Ruihan Yang, Guangjing Wang, et al. Gui agents for continual game generation. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 23260–23278, 2026. 2

[18] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21807–21818. IEEE, 2024. 3

[19] Chaobo Jia, Ruipeng Wan, Ting Sun, Weihao Tan, Borui Wan, Yuxuan Tong, Guangming Sheng, and Hong Xu. Gamegen-verifier: Parallel keypoint-based verification for llm-generated games via runtime state injection. *arXiv preprint arXiv:2605.07442*, 2026. 2

[20] Yilei Jiang, Jinyuan Hu, Qianyin Xiao, Yaozhi Zheng, Ruize Ma, Kaituo Feng, Jiaming Han, Tianshuo Peng, Kaixuan Fan, Manyuan Zhang, et al. Opengame: Open agentic coding for games. *arXiv preprint arXiv:2604.18394*, 2026. 1, 2

[21] Alexia Jolicoeur-Martineau. Multi-agent game generation and evaluation via audio-visual recordings. *arXiv preprint arXiv:2508.00632*, 2025. 2

[22] Yuhang Li, Chenchen Zhang, Ruilin Lv, Ao Liu, Ken Deng, Yuanxing Zhang, Jiaheng Liu, and Bo Zhou. Relook: Vision-grounded rl with a multimodal llm critic for agentic web coding. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25471–25485, 2026. 3

[23] Do Xuan Long, Xingchen Wan, Hootan Nakhost, Chen-Yu Lee, Tomas Pfister, and Sercan Ö Arik. Vista: A test-time self-improving video generation agent. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6021–6032, 2026. 3

[24] Tongxu Luo, Rongsheng Wang, Jiaxi Bi, Chenming Xu, Zhengyang Tang, Jianlong Chen, Juhao Liang, Ke Ji, Shuqi Guo, Yuhao Du, et al. Gamecraft-bench: Can agents build playable games end-to-end in a real game engine? *arXiv preprint arXiv:2606.17861*, 2026. 1, 2, 3, 4, 5

[25] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Jim Fan, et al. Eureka: Human-level reward design via coding large language models. In *International conference on learning Representations*, pages 26516–26560, 2024. 3

[26] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, pages 46534–46594. Curran Associates, Inc., 2023. 2

[27] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025. 2

[28] Zhiyuan Peng, Wei Tao, Xin Yin, Chenhao Ying, Yuan Luo, and Yiwen Guo. Playcoder: Making llm-generated gui code playable. *Proceedings of the ACM on Software Engineering*, 3(FSE):2003–2026, 2026. 3

[29] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625 (7995):468–475, 2024. 3

[30] Hannah Sansford, Derek HC Law, Wei Liu, Abhishek Tripathi, Niresh Agarwal, and Gerrit JJ Van den Burg. Vision-guided iterative refinement for frontend code generation. *arXiv preprint arXiv:2604.05839*, 2026. 3

[31] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8634–8652. Curran Associates, Inc., 2023. 2

[32] Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2code: Benchmarking multimodal code generation for automated front-end engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3956–3974, 2025. 3

[33] Tencent Multimodal RL Team, Xiangxin Zhou, Jiaqi Tang, Ke Ma, Wenhao Yang, Jiarui Yao, Boye Niu, Haiyang Guo, Yifei Zhao, Huanjin Yao, et al. Game-creation-agents: Autonomous game production. *arXiv preprint arXiv:2610.16776*, 2026. 1, 3

[34] Dani Valevski, Yaniv Leviathan, Moab Arar, and Shlomi Fruchter. Diffusion models are real-time game engines. In *International Conference on Learning Representations*, pages 73754–73776, 2025. 2

[35] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 3

[36] Jianze Wang, Kunwang Zheng, Ying Liu, Yu Cao, Qilong Zhang, Jinlong Chen, Hua Yang, and Qianglong Chen. Serpo: Self-evolving rubric policy optimization for open-ended test-time reinforcement learning. *arXiv preprint arXiv:2607.26873*, 2026. 3

[37] Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. Openhands: An open platform for ai software developers as generalist agents. In *International Conference on Learning Representations*, pages 65882–65919, 2025. 1

[38] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 63897–63911. PMLR, 2025. 3

[39] Tsung-Han Wu, Long Lian, Joseph E Gonzalez, Boyi Li, and Trevor Darrell. Self-correcting llm-controlled diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6327–6336. IEEE, 2024. 3

[40] Wenyi Wu, Minghao Fu, Jieyu You, Kun Zhou, Siqi Liu, Aayush Salvi, Yiheng Lin, Ce Zhang, Xiaohan Lan, Jiahui Zhu, et al. Rsigame: Autonomous agentic game development with recursive self-improvement. *arXiv preprint arXiv:2609.39045*, 2026. 2, 5

[41] Haoyang Yan, Min-le Su, Hangfan Zhang, Zhanhao Li, Chen Zhang, Shao Zhang, Yang Chen, Lei Bai, and Shuyue Hu. Harness-of-harness: Multi-day autonomous software development with continual improvement. *arXiv preprint arXiv:2609.01481*, 2026. 2

[42] Zhengyuan Yang, Jianfeng Wang, Linjie Li, Kevin Lin, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. Idea2img: Iterative self-refinement with gpt-4v for automatic image design and generation. In *European conference on computer vision*, pages 167–184. Springer, 2024. 3

[43] Ziyun Zeng, Zixuan Wang, Yongsheng Yu, Hang Hua, and Jiebo Luo. Videoargus: Agentic rubric-grounded unified evaluation for video generation and editing. *arXiv preprint arXiv:2608.05485*, 2026. 3

[44] Wei Zhang, Jian Yang, Renshuai Tao, Linzheng Chai, Shuyue Guo, Jiajun Wu, Xiaoming Chen, Ganqu Cui, Ning Ding, Xander Xu, et al. V-gamegym: Visual game generation for code large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 5613–5641, 2026. 2

[45] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 19632–19642, 2024. 3