MIMIC: Integrating Diverse Personality Traits for Better Game Testing Using Large Language Model

Yifei Chen
Electrical and Computer Engineering
McGill University
Montréal, Canada
yifei.chen@mail.mcgill.ca

Sarra Habchi
Cohere
Canada
sarra.habchi@cohere.com

Lili Wei[1]
Electrical and Computer Engineering
McGill University
Montréal, Canada
lili.wei@mcgill.ca

***Abstract*—Modern video games pose significant challenges for traditional automated testing algorithms, yet intensive testing is crucial to ensure game quality. To address these challenges, researchers designed gaming agents using Reinforcement Learning, Imitation Learning, or Large Language Models. However, these agents often neglect the diverse strategies employed by human players due to their different personalities, resulting in repetitive solutions in similar situations. Without mimicking varied gaming strategies, these agents struggle to trigger diverse in-game interactions or uncover edge cases.**

In this paper, we present MIMIC, a novel framework that integrates diverse personality traits into gaming agents, enabling them to adopt different gaming strategies for similar situations. By mimicking different playstyles, MIMIC can achieve higher test coverage and richer in-game interactions across different games. It also outperforms state-of-the-art agents in Minecraft by achieving a higher task completion rate and providing more diverse solutions. These results highlight MIMIC's significant potential for effective game testing.

***Index Terms*—Artificial Intelligence, Human-Like Gaming Agents, Personality-Driven Gaming Agents, Automated Game Testing, Large Language Models (LLMs).**

I. Introduction

Modern video games have become one of the most significant entertainment sectors, generating USD 183.9 billion globally in 2023 [1]. To maintain game quality, rigorous testing has become essential, reflected in the growth of the game testing services market, valued at USD 321.4 million in 2025 and projected to reach 670.5 million by 2033 [2].

Yet modern games pose significant challenges for traditional automated testing [3]. A common technique, "record and replay", where human interactions are captured and reused as test cases [4]. While effective at reproducing known scenarios, this approach often fails in nondeterministic gaming environments or when games evolve [5]. As a result, frequent updates to the recordings are required, making it inefficient for modern game development cycles.

To address this limitation, researchers have explored agent-based testing leveraging machine learning (ML) techniques such as reinforcement learning (RL) [6] and imitation learning (IL) [7]. While effective at executing test plans, RL depends on rigid reward functions and IL relies on expert demonstrations, limiting its generalization to new tasks or games [8]. More recently, Large Language Models (LLMs) have been applied to gaming agents. And such LLM-based agents have demonstrated impressive adaptability in solving complex tasks across diverse games [9]–[13].

A common limitation of both ML- and LLM-based agents is overlooking the diverse strategies players adopt for the same task, shaped by their personalities. Humans may approach a task conservatively or aggressively [14], but existing agents often ignore such behavioural diversity and generate repetitive solutions. This limits their ability to thoroughly explore games or uncover edge cases, reducing effectiveness in game testing.

To address this challenge, we propose MIMIC, an LLM-based framework that mimics different gameplay personalities to generate diverse solutions for the same in-game tasks and achieve higher coverage. Our key insight comes from real-world gameplay, where players may approach the same tasks with varied strategies shaped by their personalities. Yee et al. [15] found significant correlations between personality traits and behaviours in *World of Warcraft*. Similarly, Narnia et al. [16] showed that in-game player behaviours align with real-world personality traits. For example, emotional players may prefer levelling alone to reduce negative feedback from others, whereas extroverts prefer social questing. Inspired by these observations, MIMIC integrates diverse personality traits into gaming agents to simulate realistic, diverse behaviours.

MIMIC leverages LLMs to align agent behaviour with specific personality traits. For example, when facing an opponent, a cautious agent may avoid combat, while an aggressive one would attack directly. A Memory System further records past gameplay and retrieves useful experiences, allowing agents to accumulate knowledge over time and consistently solve complex tasks in line with their personalities.

To assess MIMIC's effectiveness, we use it to test two open-source games of varying complexity, measuring both *code-level* and *interaction-level* coverage. In the small-scale game, MIMIC showed performance comparable to human testers, reaching **100% *combinatorial coverage***, which measures how thoroughly agents explore combinations of in-game actions and parameters, and narrowing the gap in code-level coverage. In the large-scale game, it consistently outperformed random-based baselines, achieving up to **1.30**× higher branch coverage

[1]Lili Wei is the corresponding author.

and **14.46**$\times$ greater combinatorial coverage.

We further evaluated MIMIC in the widely used real-world game **Minecraft** [17], comparing it against the state-of-the-art agent **ODYSSEY** [18]. Both were assigned the same suite of in-game tasks. MIMIC not only outperformed ODYSSEY in task completion but also showed greater behavioural diversity in six of eight tasks, yielding broader coverage of gameplay scenarios. The key contributions of this paper are:

- We propose a novel framework that integrates gaming agents with diverse personalities, enabling more diverse and effective game testing.
- We conduct two studies across three games and demonstrate MIMIC's effectiveness in solving complex tasks and achieving higher coverage through diverse solutions.
- To facilitate future research, we made MIMIC and all the used prompts public [19].

II. Background

A. Retrieval-Augmented Generation for LLMs

Large Language Models (LLMs) are deep learning systems with billions of parameters trained on massive datasets. They generate human-like text and code with contextual and semantic awareness. Public interest surged after the release of ChatGPT, which reached about 180 million users in 2024 [20].

However, LLMs are prone to "hallucinations" [21], plausible but incorrect or fabricated content that may deviate from user inputs [22]. To mitigate hallucinations and improve efficiency, Retrieval-Augmented Generation (RAG) was introduced [23]. RAG has two steps: *Retrieval*, which queries a predefined database for relevant information, and *Generation*, which combines the retrieved content with the user query as input to the LLM. This approach improves accuracy in question answering, and has become widely studied for its efficiency and flexibility [24], [25].

B. Modelling Gamer Traits

In 1996, Bartle categorized player behaviours into four personality traits [26], shaping how players engage with games. Subsequent research expanded this work, modelling gamer behaviours through personality-based classifications [27]–[32]. A recent study synthesized these efforts into seven traits: *Achievement*, *Adrenaline*, *Aggression*, *Caution*, *Completion*, *Curiosity*, and *Efficiency* [33]. We leverage these traits to prompt our agents to mimic human behaviours.

III. Motivating Example

In this section, we use a task from Minecraft as a motivating example to discuss the limitations of existing work and highlight the motivation of MIMIC.

The task *Obtain 1 diamond* is long-hailed as a significant challenge in the community [34] and serves as the focus of the NeurIPS MineRL Competition [35]. It requires completing at least 13 sequential sub-goals, each with multiple variants, making it a long-horizon task that typically takes humans over ten minutes to solve [36]. Players must also handle dynamic requirements posed by the game, such as hunger or safety, further expanding the solution space.

Existing gaming agents are optimized for task completion, often producing homogeneous, repetitive behaviours. For example, in our evaluation, we observed that ODYSSEY [18], a state-of-the-art LLM-based agent, consistently followed a single optimized path to obtain a diamond, regardless of environmental variations. While such agents achieve high task completion rates, their behaviours diverge from human players, who rarely pursue tasks in strictly optimized ways. Instead, human players exhibit adaptive and diverse behaviours in response to spontaneous in-game events, shaped by their personality traits [15], [16]. For example, an aggressive player may fight creatures for rewards while pursuing the diamond task, even if it does not directly advance the primary goal. As a result, the existing gaming agents fail to emulate this diversity and adaptability, limiting their ability to cover the wide range of unpredictable in-game scenarios and reducing their overall effectiveness for game testing.

This motivates us to propose MIMIC, an LLM-based agent framework that integrates personality traits into the core planning process. Unlike existing agents that narrowly pursue optimal action sequences, MIMIC leverages recent advances in LLMs capable of simulating consistent personality traits [37]–[40] to model gameplay behaviours that more closely resemble those of real human players. By conditioning its dynamic Planner on distinct personality prompts, MIMIC generates strategies that are task-oriented and driven by personality-specific tendencies. This enables it to pursue goals while continuously responding to in-game events in a manner consistent with a player of that personality, leading to more diverse, realistic, and interaction-rich testing trajectories. For example, in finishing the *Obtain 1 Diamond* task, our *aggressive* agent dedicated 21.52% of its actions to combat, frequently upgrading armour and engaging a variety of creatures. In contrast, the *cautious* agent avoided combat entirely, prioritizing safety by crafting torches before mining. Meanwhile, the *adrenaline-seeking* agent actively crafted swords and explored high-risk areas to encounter enemies, reflecting a strong preference for challenge-oriented interactions. These results demonstrate that by integrating personality into our gaming agent, MIMIC can generate meaningful actions in response to various environments according to the specified personality, driving exploration towards more diverse scenarios.

Furthermore, human decision-making is shaped not only by personality but also by experience and preferences [41]–[43]. To model this, we introduce a Memory System that records past actions and outcomes as memories. The relevant and preferred ones are then retrieved to guide consistent, human-like decisions.

By combining personality-driven planning with memory-aware decision-making, MIMIC delivers a novel testing framework that mimics the behavioural diversity of human players and enables broader exploration of in-game scenarios.

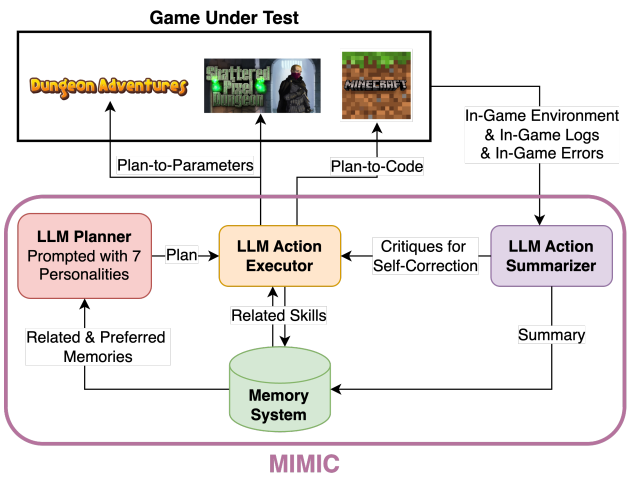


Fig. 1. Overview of the MIMIC framework, comprising three LLM-based components: LLM Planner, LLM Action Executor, and LLM Action Summarizer, alongside a non-LLM-based component, the Memory System.

IV. Approach

Fig. 1 presents an overview of the MIMIC framework, which consists of four components: the Planner, Action Summarizer, Action Executor, and Memory System.

The Planner is the core module, generating action plans from predefined personality traits and past experiences. These experiences are stored in the Memory System, with the Action Summarizer analyzing execution results to produce summaries. The Action Executor then translates the Planner's output into in-game actions. In each action iteration, the Planner produces a plan, the Executor executes it, and the Summarizer records feedback as new memory to guide future planning. The following sections detail each component.

A. Planner

The LLM Planner is the core component that integrates personality traits into the decision-making process of MIMIC. Unlike prior agents that follow an optimal action sequence, our planner is prompted with personality traits to generate plans that reflect diverse human-like behaviours. These plans simulate how players with different personalities may approach the same goal, enabling more varied and realistic gameplay.

To handle both immediate actions and long-term objectives in accordance with the personality traits, we adopt a hybrid planning design. This allows the Planner to dynamically alternate between low-level reactive planning and high-level goal decomposition, enhancing both adaptability and personality alignment. This hybrid design increases behavioural diversity and improves test coverage in complex game environments.

In addition to personality traits, the Planner also receives information about the in-game environment and game mechanics, enabling it to generate grounded, context-aware plans. To support coherent progression, previously executed plans are also provided as reference (retrieved from the Memory System in Section IV-D), allowing the Planner to build on past decisions and maintain continuity throughout gameplay.

1) Mimicking Gamer Personality: To integrate personality-driven behaviours into MIMIC, we find a personality model to guide it. We leveraged the model from PathOS [33], which synthesizes seven personality traits, *Achievement*, *Adrenaline*, *Aggression*, *Caution*, *Completion*, *Curiosity*, and *Efficiency*, from nine player modelling studies spanning 1981–2017. These traits are behaviourally grounded, well-defined, and generalizable, making them directly applicable to MIMIC.

Alternative personality models are less suitable. Generic models (e.g., Big Five, MBTI) are insufficient to capture gameplay behaviours. Some game-specific models exist, but are narrower than PathOS. For example, Narnia et al. [16] derived a model from a single game, limiting generalizability. A later study [44] defined four personality traits, all of which are subsets of PathOS, and another work [45] examined motivations behind player behaviours but did not define personalities, reducing applicability to MIMIC's framework.

Additionally, we mapped high-level game entities defined by PathOS to equivalent terms in specific games. PathOS defines nine entity types in total. For example, "Enemy Hazard", described as "A hostile character, etc., which could incite combat", maps to "enemies" in DA and SPD, and to "mobs" in Minecraft, which uses a different term. Such mappings are straightforward to construct from game code and documentation, facilitating the extension of MIMIC to new games.

2) Hybrid Planning to Accomplish Complex Tasks: Many LLM-based agents generate only the next immediate action based on the current game state [10], [46], [47]. We refer to this strategy as a Bottom-Up Planner, which accomplishes tasks through individual actions. While effective for simple tasks, this approach often fails on complex goals requiring long-horizon steps [11]. For instance, when tasked to *craft a tool*, the agent may plan to *collect resources* but later use them for unrelated actions, losing sight of the original goal.

We introduce the **Hybrid Planner**, which dynamically switches between Bottom-Up and Top-Down strategies to track goals and task progress better. It combines both strengths: the Bottom-Up Planner generates immediate, low-level actions, while the Top-Down Planner decomposes high-level goals into sub-plans using an LLM-based module. Each sub-plan is executed sequentially, and terminates once all sub-plans are completed or any of them is deemed infeasible. This process helps the agent stay aligned with complex goals. The Planner begins with Bottom-Up planning. Once MIMIC completes a predefined number of tasks, it switches from the Bottom-Up to the Top-Down Planner. Subsequently, the Hybrid Planner dynamically alternates between the two modes based on *plan diversity*, which is measured by tracking repeated actions and interacted objects across consecutive plans. If no new actions or objects are detected over a defined window, the Planner switches modes to encourage more varied exploration in MIMIC. To reduce hallucination where the Planner generates infeasible plans, we adopt the Prompt Chaining technique [48], where each prompt builds on the output of the previous one to maintain contextual continuity. In our system, generated plans are verified against the game's definitions, and revision prompts are issued when misalignments are detected. This iterative process improves the feasibility and precision of the resulting plans.

B. Action Summarizer

The LLM Action Summarizer evaluates each iteration's execution by determining whether it successfully accomplishes the plan, and then generates a summary based on the outcome of the evaluation. To mitigate unrealistic expectations that may misalign with the game state [21], [22], we apply Prompt Chaining as introduced in IV-A2. In this process, the Summarizer first prompts the LLM to predict the expected outcomes and game logs resulting from the action plan. Using these predictions as inputs, the Summarizer evaluates each action by comparing the inputs against the actual execution results. It then generates a reflective summary, leveraging the Chain-of-Thought (CoT) technique [49], where a rationale accompanies every statement. These summaries are stored in the Memory System to inform future planning (see IV-D).

C. Action Executor

The Action Executor connects MIMIC to the game under test by translating action plans into executable forms. It supports two interface types: code snippets (*Plan-to-Code*) and API input parameters (*Plan-to-Parameters*).

1) Plan-to-Code Translator: Some games expose control interfaces through SDKs or APIs for basic actions, requiring testers to prepare custom code scripts to assemble the APIs for complex tasks. For example, Minecraft's Mineflayer API [50] lacks support for advanced actions, demanding extra scripting.

To address this, the Action Executor uses a Plan-to-Code Translator to convert Plans into executable code snippets that interact with game APIs. It generates reusable scripts ("Skills") based on basic API examples that MIMIC can invoke directly. The Action Summarizer then verifies execution against game states and logs, providing feedback to refine Skills when they fail to achieve the intended plans. This loop is essential, as LLMs often produce syntax errors, logic bugs, or infinite loops [51], [52]. To further address issues like infinite loops or infeasible tasks, we introduce another LLM module to allocate execution time based on plan complexity and MIMIC's personality, e.g., aggressive agents allow more time for combat, while cautious ones allow less.

2) Plan-to-Parameters Translator: Exposed APIs directly map to actions in some games. In such cases, the Action Executor translates high-level plans into API parameters, enabling seamless interaction by MIMIC.

3) Custom Translators: The Action Executor supports two translators, enabling MIMIC to adapt to diverse games. For games with unique architectures, developers can create custom APIs and integrate them with a well-designed Executor to bridge MIMIC and the game.

D. Memory System

The Memory System stores actions, in-game environments, and Summaries as Memories, which are later retrieved to guide planning. These Memories help the Planner generate context-aware actions aligned with the agent's personality. However, as action iterations grow, including all Memories in prompts becomes infeasible due to token limits and increased hallucination risk [53]–[55]. To address this, we adopt a Retrieval-Augmented Generation (RAG) approach [23] to manage and retrieve only the most relevant Memories. This reduces token overhead and mitigates hallucinations [24], [25].

Retrieval of Preferred Plans: To simulate the influence of human preferences in decision-making [56], each Memory is paired with a *preference summary*, an LLM-generated reflection conditioned on the given personality, describing how the action and outcome preferred by such a personality. A *preference score* is then computed using cosine similarity between this summary and the personality prompt, and the top five scoring Memories are retrieved as preferred plans.

Retrieval of Related Memories: To mimic how human players recall past experiences [41], the Memory System retrieves the top five most relevant Memories based on Cosine Similarity between the current and past in-game environments. It supplies them to the Planner for the next planning phase.

Retrieval of Skills: As introduced in Section IV-C1, reusable code snippets, called Skills, are generated to interact with games lacking SDKs or complete APIs. Each generated Skill is stored with a textual description. When generating new code, the Action Executor retrieves the top five most similar Skills based on Cosine Similarity between the current plan and the description, enabling reuse or providing references.

V. Evaluation

To evaluate the performance of MIMIC, we conducted two complementary studies: an *effectiveness study* and a *usefulness study* across three different game subjects. These studies address the following research questions:

- **RQ1**: How effective is MIMIC in achieving code coverage?
- **RQ2**: How effective is MIMIC in covering diverse in-game behaviours and interactions?
- **RQ3**: How does MIMIC perform compared to state-of-the-art tools in completing given tasks?
- **RQ4**: How diverse are MIMIC's solutions in solving the given tasks compared to existing tools?

The *effectiveness study* answers **RQ1** and **RQ2** by evaluating MIMIC's *code-* and *interaction-level* coverage in two open-source games. The *usefulness study* answers **RQ3** and **RQ4** by comparing MIMIC with a state-of-the-art LLM agent in **Minecraft (MC)** [17], a widely adopted subject with rich agent baselines [8]. This real-world, closed-source setting focuses on *task completion* and *solution diversity*, highlighting MIMIC's practical performance against existing tools.

A. Effectiveness Evaluation (RQ1 & RQ2)

This section evaluates MIMIC's effectiveness in terms of *code-level* and *interaction-level* coverage using two open-

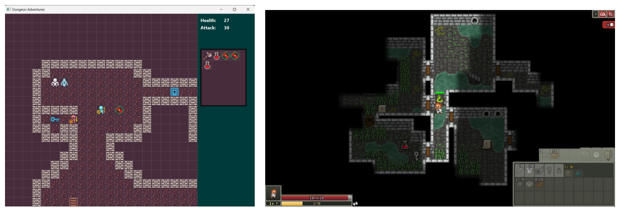


Fig. 2. Screenshots of the Dungeon Adventures (left) and Shattered Pixel Dungeon (right).

source games: ***Dungeon Adventures (DA)*** [57] and ***Shattered Pixel Dungeon (SPD)*** [58] (see screenshots in Fig. 2).

*1) **Experimental Setup**:* To accommodate LLM response latency, we selected non-time-sensitive games and integrated lightweight API layers for interaction, without altering any game logic or mechanics. This preserved original gameplay for realistic, unbiased evaluation.

DA is a small-scale, turn-based Role-Playing Game (RPG) with four levels, four item types, and four enemy types. Players can move, collect items, or engage in proximity-based combat. The gameplay is simple, with limited operations and flexibility. The testing objective for this game is to defeat the boss.

SPD is a large-scale, turn-based roguelike RPG, with randomly generated maps, items, and enemies. Since its 2014 release, SPD has 1M+ downloads and 3,900 GitHub stars. It supports complex actions, such as crafting and upgrading, with 25 levels, 250+ item types, and 65+ enemy types. The testing objective for this game is to complete the dungeon.

All LLMs in MIMIC used GPT-4o (version 2024-08-06) [59] via API. Experiments were run on a 32GB RAM, 64-bit Windows 11 machine with an Intel i7-10750H CPU @ 2.60GHz (6 cores).

*2) **Baselines**:* We compared MIMIC against five baselines.

Ablated baselines: To assess the impact of the Memory System, personality components, and the Summarizer, we include two ablated versions of MIMIC: MIMIC-P, which contains only the LLM Planner, and MIMIC-P+S, which adds the Summarizer but omits both the Memory System and personality modules.

Human baselines: Human testers serve as a manual testing baseline, a common practice in game development [3]. To match the number of personalities in MIMIC, we recruited seven testers to play both games. While results from a single group of humans may not capture full variability, our sample size was constrained by budget limitations. To mitigate this threat, all testers were experienced, each with over five years of gaming experience and averaging 6.8 hours of gameplay per week. Participants were compensated 20 CAD per hour, and no identifiable information was collected. Moreover, our results show that this group size is sufficient to highlight the performance gap between MIMIC and humans (see Section VII).

Random baselines: We do not include existing agents as baselines for RQ1-2, as they are highly tailored to specific games and cannot be easily adapted to new games. Instead, to represent automated random testing strategies, we implemented two Monkey Testing variants [60]–[62]. **Dumb Monkey** randomly invokes exposed APIs with unconstrained inputs, while **Smart Monkey** ensures that all invoked actions and parameters are valid, triggering only meaningful actions.

*3) **Metrics**:*

***Code Level Metrics**:* To assess code-level effectiveness, we measured *code coverage* and *branch coverage*, the de facto test coverage criteria to evaluate test cases. Since automated tools interact via APIs rather than the UI, all UI-related code, unreachable through agents, was excluded. Additionally, we excluded code modifications from our side for API development and log instrumentation, as these were inaccessible to some tools. Coverage was measured using JaCoCo [63].

***Interaction Level Metrics**:* While code-level coverage measures how much of a game's internal logic is executed, it fails to capture how agents interact with the game world. For instance, branch coverage may confirm that an item was used. However, it cannot detect whether multiple items were used concurrently or under specific in-game conditions, which are the scenarios common in gameplay that can trigger unforeseen bugs. We propose to use **combinatorial coverage** to address this gap and evaluate diverse action-parameter combinations across conditions. Likewise, code coverage might show that a collision handler was invoked, but not the agent encountered edge cases like clipping through walls or misaligned hitboxes; such issues are more effectively assessed by **navigation coverage**, which measures the spatial exploration. Together, these interaction-level metrics reveal behaviour-driven bugs that code-level metrics often miss.

- **Combinatorial Coverage:** It measures the percentage of the combinations of actions and parameters explored [64], [65]. To model these combinations, we define a *combinatorial rule* as a tuple of an *action type* (e.g., *use*, *throw*, *eat*) and up to four parameters: subject items, targets, carrying items, and character upgrades. *Subject items* are the primary objects involved in the action; *targets* are entities they interact with; *carrying items* and *character upgrades* are boolean flags indicating inventory or acquired abilities, which may trigger special interactions. We analyzed game source code and classes to map entities to these types and to generate combinations, each representing a distinct scenario. For example, `[throw, stone, door, potion, ¬random_upgrade]` corresponds to "the player throws a stone at a door while carrying a potion without the random upgrade". Complete definitions are available on our project website [19].
- **Navigation Coverage:** It measures how thoroughly MIMIC explores the game's spatial layout. We track reachable locations across rooms and levels, then compute the proportion visited. In SPD, this includes room transitions, alternate paths, and secret areas. High navigation coverage reflects a tool's ability to uncover nonobvious paths, adapt to environmental complexity, and reveal pathfinding-related issues.

Although the numerical maximum of coverage is 100%, this is rarely achievable in practice due to factors like unreachable code (e.g., dead code) and infeasible gameplay combinations. Meanwhile, precisely detecting them remains an open challenge, requiring extensive analysis and effort. Thus, the practical maximum coverage is lower. While MIMIC remains far from these bounds, our results show that integrating personality-driven planning into LLM-based agents improves both behavioural diversity and code coverage over existing game testing tools, highlighting ongoing gaps in automated game testing and motivating further research.

*4) **Execution Setup**:* To reflect MIMIC's integration of seven personalities, **one complete run consists of seven individual runs**, each corresponding to a distinct personality. All experiments were repeated thrice to account for randomness in MIMIC's behaviour, resulting in 21 runs. For fairness, the same setup was applied to all other automated baselines.

For MIMIC-P, MIMIC-P+S, and human testers in **SPD**, only one complete run (seven individual runs) was conducted. Since these baselines do not incorporate personality variations, a single complete run with seven repetitions sufficiently accounts for their behavioural randomness. Our evaluation confirms that this reduced run count did not impact the reliability of performance comparisons: human testers generally outperformed automated tools, while MIMIC-P frequently stalled within the first hour, and MIMIC-P+S consistently achieved lower final coverage than the complete version of MIMIC.

- **Setup for DA**: Automated tools were given a time limit of one and a half hours per run. Based on observed efficiency, human testers were allocated one hour per run, as their coverage typically converged more quickly and extended playtime offered diminishing returns. The game was modified to automatically restart upon player death or victory, with no other changes to its original design.
- **Setup for SPD**: All tools and testers were given a four-hour time limit per run. The game auto-restarted after each death or victory, generating a new map using a predefined list of random seeds to ensure consistency across SPD's roguelike environment for different tools.

*5) **Code Level Effectiveness (RQ1)**:* Fig. 3 and Fig. 4 show code and branch coverage over time in DA and SPD, respectively. In DA, MIMIC achieved the highest code (95.67%) and branch (92.77%) coverage within 35 minutes. In SPD, despite the game's larger scale and complexity, MIMIC still outperformed all automated tools, reaching 30.50% code and 24.49% branch coverage.

Compared to humans and Monkeys, MIMIC demonstrated superior performance. While its early progress in DA was slightly slower than that of human testers, it eventually matched their final coverage levels with a higher branch coverage. Against the Monkeys, MIMIC achieved 1.16× higher coverage in both code and branches. In SPD, MIMIC achieved 3% higher coverage on average than Smart Monkey, corresponding to 11,776 additional lines and 1,115 more branches, and 6% higher coverage than Dumb Monkey.

Compared to ablated versions, MIMIC consistently achieved the best results. In DA, MIMIC achieved 1.06× higher code and branch coverages than MIMIC-P, and ultimately outperformed MIMIC-P+S in branch coverage. In SPD, MIMIC achieved 6% higher average coverage than MIMIC-P across both metrics. Although the improvement over MIMIC-P+S was minor, with 0.21% in code and 0.04% in branch coverage, this is mainly due to action throughput differences: MIMIC-P+S executed around 2,500 actions per run with two LLM components, while MIMIC executed only 1,500 with three to four LLM components, including memory retrieval. To better demonstrate the impact of the throughput differences, Fig. 5 plots coverage per action iteration, where MIMIC consistently achieved higher coverage efficiency, outperforming MIMIC-P+S by around 2% in coverage per action. These results confirm that all components in MIMIC holistically contribute to the performance. MIMIC-P+S's improvement over MIMIC-P underscores the Summarizer's role in maintaining planning context. With MIMIC outperforming MIMIC-P+S with even fewer actions, Memory System and personality-driven planning are proven to be effective in driving more diverse and effective gameplay exploration.

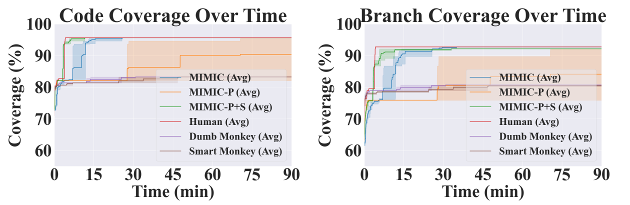


Fig. 3. Code coverage (left) and branch coverage (right) for **Dungeon Adventures (DA)**. The shaded areas represent the range across three runs, while the solid lines indicate the average coverage over time. The human completed only one complete run (no shaded area).

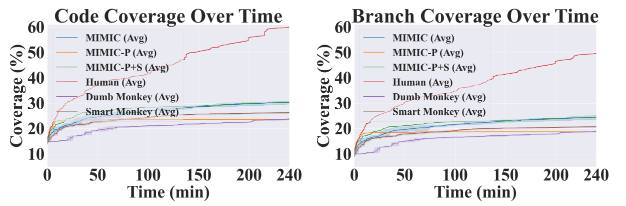


Fig. 4. Code Coverage (left) and Branch Coverage (right) for **Shattered Pixel Dungeon (SPD)** Over Time. MIMIC-P, MIMIC-P+S, and Human completed only one complete run (no shaded area).

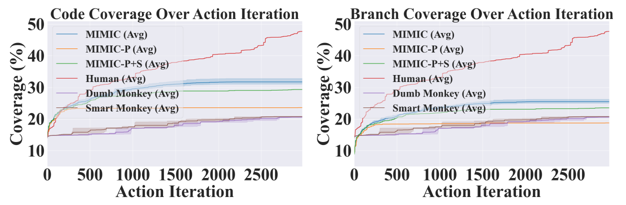


Fig. 5. Code Coverage (left) and Branch Coverage (right) for **Shattered Pixel Dungeon (SPD)** Over Action Iteration.

Currently, MIMIC uses ChatGPT-4o via the OpenAI API, which introduces communication overhead and is not dedicated to game interaction or personality mimicking. Despite this, MIMIC still shows effectiveness in achieving high code-level coverage. Future work will explore locally fine-tuned LLMs tailored to game-specific tasks, which could further enhance efficiency and better align the system with the demands of automated game testing.

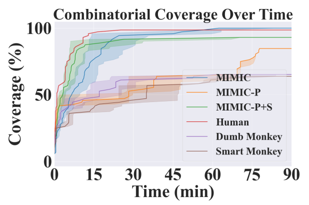
*6) **Interaction Level Effectiveness (RQ2):*** MIMIC demonstrates strong effectiveness over automated tools in exploring diverse and meaningful gameplay interactions, as reflected in both combinatorial and navigation coverage metrics. In DA, MIMIC achieves 100% combinatorial coverage across 72 defined combinations (Fig. 6a) and records the highest average navigation coverage among automated tools (Table I). In SPD, despite its significantly larger, procedurally generated environment, MIMIC attains the highest combinatorial coverage among automated tools, covering 0.188% (21,319 out of 11.3 million combinations, Fig. 6b), and reaches the deepest average navigation levels and highest averaged navigation coverage among automated tools (Table I). For consistency, navigation coverage in SPD is reported over the first four levels due to its random seeding of large-scale maps, which also aligns with the typical exploration range across tools. These results demonstrate MIMIC's ability to navigate and interact within complex and large-scale game environments.

Compared to human testers, MIMIC achieved lower navigation coverage but eventually surpassed them in combinatorial coverage in DA. In both games, human testers maintained an edge in level progression and navigation coverage. However, relative to Dumb Monkey and Smart Monkey, MIMIC showed consistent advantages: in DA, it achieved 1.5× higher combinatorial coverage; in SPD, it covered 2.51× more combinations than Smart Monkey and reached significantly greater navigation depth and coverage. Although MIMIC did not outperform human testers in SPD, its continuous improvement without saturation over four hours underscores its capacity for long-term exploration and interaction learning.

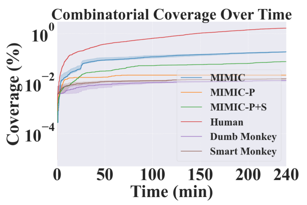
MIMIC also consistently outperformed its ablated variants. In DA, it achieved 1.18× higher combinatorial coverage than MIMIC-P and 1.07× more than MIMIC-P+S. In SPD, MIMIC reached 0.188% coverage, substantially outperforming MIMIC-P (0.022%) and MIMIC-P+S (0.075%), which is especially significant given the scale of the combination space. In navigation, MIMIC explored both more deeply and with higher averaged coverage than its ablations (1.28×-1.35× higher). These gains highlight the role of memory retrieval and personality-driven planning in promoting more varied, goal-aware exploration.

To reduce bias from predefined rules, we also measured coverage over all interactable object types (e.g., terrain, characters, items), observing consistent trends. This confirms the value of interaction-level metrics and MIMIC's effectiveness in testing complex gameplay scenarios.

Since only one complete run was conducted for the human group in both games, statistical tests are not appropriate, as



(a) Combinatorial Coverage in DA



(b) Combinatorial Coverage in SPD

Fig. 6. Combinatorial coverage for **Dungeon Adventures (DA)** (left) and **log-scaled** coverage for **Shattered Pixel Dungeon (SPD)** (right). The shaded areas represent the range across three runs, while the solid lines indicate the mean time to cover individual combinations.

TABLE I
AVERAGE LEVELS EXPLORED & AVERAGE NAVIGATION COVERAGE ACROSS THE MAP BY DIFFERENT TOOLS.

Game	Testing Tool	Avg. Lvl. Explored ± Std.	Avg. Nav. Cov. ± Std.
DA	MIMIC	3.00 ± 1.13	52.16 ± 5.56
	MIMIC-P	2.00 ± 1.32	23.62 ± 9.13
	MIMIC-P+S	3.07 ± 1.39	45.67 ± 1.93
	Human	2.43 ± 0.87	99.45 ± N/A
	Dumb Monkey	1.93 ± 0.26	32.08 ± 2.88
	Smart Monkey	1.71 ± 0.47	30.22 ± 3.72
SPD	MIMIC	2.28 ± 0.92	14.41 ± 7.51
	MIMIC-P	1.90 ± 0.66	11.27 ± 5.77
	MIMIC-P+S	1.94 ± 0.82	10.67 ± 4.92
	Human	3.89 ± 2.90	47.07 ± 20.29
	Dumb Monkey	1.07 ± 0.27	5.75 ± 3.15
	Smart Monkey	1.05 ± 0.24	5.77 ± 3.49

they require multiple samples to estimate population parameters. Instead, we report confidence intervals for these two groups in Table II. In DA, the intervals are close across all three metrics, showing comparable performance with MIMIC achieving marginal improvements. In SPD, a more complex environment, the human group achieves substantially higher coverage with nonoverlapping intervals, highlighting the significant performance gap between humans and MIMIC.

B. Usefulness Evaluation (RQ3 & RQ4)

This section evaluates MIMIC's usefulness by comparing it to a state-of-the-art LLM-based agent in **Minecraft (MC)**, focusing on two dimensions: *task completion* (RQ3) and *solution diversity* (RQ4). These are measured by success rates and variation in task-solving interactions, respectively, to assess MIMIC's performance relative to existing tools.

*1) **Experimental Setup:*** MC is selected as the subject for this study due to its rich action space and open-ended gameplay, making it widely used for gaming agent evaluation [8]. Its popularity has led to the development of many LLM-based agents, enabling meaningful comparisons.

We compared with **ODYSSEY** [18] in its original design as the baseline since it's a state-of-the-art gaming agent for MC with strong problem-solving performance (e.g., achieving over 90% success on the challenging *Obtain 1 Diamond* task). It consistently outperforms other existing agents, including Voyager [10], GITM [11], VPT [66], and DEPS [34]. ODYSSEY comprises a Planner, an Action Executor, and a Critic for summarizing plans. Unlike MIMIC's code-generation approach, its

TABLE II
95% CONFIDENCE INTERVALS OF COVERAGE FOR MIMIC AND HUMAN. NOTE THAT THE HUMAN BASELINE INCLUDES ONLY ONE COMPLETE RUN, SO THE INTERVAL EQUALS THE OBSERVED AVERAGE.

Game	Tool	Code	Branch	Combinatorial
DA	MIMIC	[95.7%, 95.7%]	[93.06%, 93.06%]	[92.88%, 100.0%]
	Human*	[94.94%, 94.94%]	[89.02%, 89.02%]	[98.61%, 98.61%]
SPD	MIMIC	[28.62%, 32.37%]	[22.38%, 26.59%]	[0.105%, 0.271%]
	Human*	[59.97%, 59.97%]	[49.63%, 49.63%]	[1.646%, 1.646%]

TABLE III
TASK SUITE USED FOR THE COMPARISON.

Task ID	Task	Complexity	Category
GD#1	Make 1 Sugar	1	Tech Tree + Harvest
GD#2	Shear 1 Sheep	2	Tech Tree + Harvest
GD#3	Cook 1 Meat	3	Tech Tree + Harvest
GD#4	Obtain 1 Diamond	4	Harvest
TL#1	Combat 1 Cave Spider	1	Combat
TL#2	Combat 1 Skeleton	2	Combat
TL#3	Combat 1 Spider	3	Combat
TL#4	Survive 1 Day	4	Survival

Executor selects from a library of 183 pre-coded functions, ranks the top ten via Cosine Similarity, and uses an LLM to invoke the most relevant one. ODYSSEY uses MineMA-8B and MineMA-70B, both fine-tuned LLaMA-3 models [67]. Following its original evaluation, we use MineMA-8B in all ODYSSEY experiments, where it was used in most tasks.

To preserve the integrity of the baseline and ensure a fair comparison, we evaluated ODYSSEY strictly with its original skill functions. While combining our Plan-to-Code Translator with ODYSSEY may improve success rates by generating richer skills, it would not increase the solution diversity. This limitation stems from ODYSSEY's LLM planner, which lacks personalities to generate varied solutions for the same goals. Replacing the code-generation component does not address this limitation. Moreover, removing the hardcoded skills would effectively make ODYSSEY similar to our MIMIC-P+S baseline (identical to MIMIC, but without personalities; see Section V-A2), which already underperforms MIMIC in solution diversity (Section V-A5 and V-A6).

Since MC is closed-source, MIMIC interacts with the game using its Plan-to-Code Translator (Section IV-C1) to generate Skills via Mineflayer [50]. All LLM components in MIMIC use GPT-4o (version 2024-08-06) [59], accessed via API calls. All experiments were conducted on a machine with 128 GB unified memory, running macOS Sequoia 15.2 and powered by an Apple M3 Max processor with 16 cores.

*2) **Evaluation Setup**:* Unlike previously tested games, MC is not level-based and lacks predefined milestones. For fair comparison, we adopted ODYSSEY's task suite, selecting eight diverse tasks within budget limits. Each task was executed on three randomly selected maps, with three complete runs per tool (each comprising seven individual runs). For MIMIC, each run included all seven personalities.

Tasks were categorized by the MineDojo benchmark [68] into four groups: *combat*, *tech tree*, *harvest*, and *survive*. To capture varying complexity, we sampled ODYSSEY tasks, referring to the minimum, median, and maximum number of action iterations required for ODYSSEY to achieve, and added two long-horizon tasks: *Obtain 1 Diamond* and *Survive 1 Day*. The final suite is summarized in Table III, grouped into two categories for clarity:

- **Goal-Driven Tasks**: These tasks are under the *harvest* category with clear goal to collect a specific item. Each task had a one-hour time limit, except *Obtain 1 Diamond*, which was allocated two hours due to its complexity.
- **Time-Limited Tasks**: These tasks emphasize performance within a fixed time window (i.e., one in-game day) rather than completing a specific collection goal, including all *combat* tasks and *Survive 1 Day*. For combat tasks, agents were teleported to a battle arena after one day, and success was defined by defeating a specified creature.

*3) **Task Completion (RQ3)**:* To evaluate MIMIC's effectiveness in completing in-game tasks, we measured success rate within a fixed time budget and average completion time. As described in Section V-B1, each complete run comprises seven individual runs. For fairness in the RQ4 diversity comparison, we conducted additional ODYSSEY runs when its total runtime was shorter than MIMIC's.

***Goal-Driven Tasks**:* MIMIC achieved a 100% success rate across all tasks, including the more complex ones, as shown in Table IV. In contrast, ODYSSEY succeeded in only 10 out of 21 runs for *Shear 1 Sheep* and just 2 out of 21 runs for *Cook 1 Meat*, consistent with its original performance reports. The superior performance of MIMIC is mainly attributed to MIMIC's Hybrid Planner. Unlike ODYSSEY's Top-Down Planner, which strictly follows hierarchical decomposition, MIMIC can dynamically fall back to immediate, low-level planning when high-level subgoals repeatedly fail. This flexibility enables MIMIC to adapt its strategy in response to execution failures, improving robustness and task completion in complex or unpredictable scenarios.

In terms of efficiency, MIMIC outperformed ODYSSEY in most cases, except for the most challenging task, *Obtain 1 Diamond*, where ODYSSEY completed all runs significantly faster than MIMIC. However, its inconsistent performance on simpler tasks suggests it may be overfitting to this particular benchmark. To understand MIMIC's longer completion time in *Obtain 1 Diamond*, we analyzed the performance across its seven personality-driven agents. The *aggressive* personality emerged as an outlier, averaging 102.44 minutes, while others completed the task in 25–35 minutes. Further inspection revealed that aggressive agents often prioritized combat over task progression, spending more time preparing for battles unrelated to the task, consistent with their defined personality traits. This demonstrates MIMIC's ability to emulate diverse player types and realistic human decision-making.

These results show that MIMIC not only maintains high task completion across varying complexity levels but also consistently reflects its consistency with defined personality traits, resulting in more realistic actions over speed optimization.

TABLE IV
PERFORMANCE COMPARISON OF DIFFERENT AGENTS ON THE GOAL-DRIVEN TASKS. "TIME (MIN)" REFERS TO THE AVERAGE MINUTES SPENT IN COMPLETING THE TASKS. ALL EVALUATIONS ARE ONLY CALCULATED FOR SUCCESSFUL TASKS. "±" INDICATES ONE STANDARD DEVIATION OF THE AVERAGE EVALUATION OVER SUCCESSFUL TASKS.

Task	Map	Group	Success Rate (%)	Time (min)
Make 1 Sugar	Map 1	MIMIC	7 / 7 (100.0)	**7.55 ± 2.67**
		ODYSSEY	7 / 7 (100.0)	10.6 ± 8.43
	Map 2	MIMIC	7 / 7 (100.0)	**7.9 ± 3.86**
		ODYSSEY	7 / 7 (100.0)	8.61 ± 2.11
	Map 3	MIMIC	**7 / 7 (100.0)**	15.2 ± 10.79
		ODYSSEY	10 / 11 (90.90)	**7.7 ± 5.77**
Shear 1 Sheep	Map 1	MIMIC	**7 / 7 (100.0)**	**17.68 ± 6.28**
		ODYSSEY	3 / 7 (42.86)	28.64 ± 18.03
	Map 2	MIMIC	**7 / 7 (100.0)**	21.69 ± 3.25
		ODYSSEY	3 / 7 (42.86)	**16.75 ± 4.85**
	Map 3	MIMIC	**7 / 7 (100.0)**	**22.79 ± 6.54**
		ODYSSEY	4 / 7 (57.14)	38.11 ± 11.1
Cook 1 Meat	Map 1	MIMIC	**7 / 7 (100.0)**	**15.95 ± 7.24**
		ODYSSEY	0 / 7 (0.0)	N/A
	Map 2	MIMIC	**7 / 7 (100.0)**	**11.86 ± 9.68**
		ODYSSEY	1 / 7 (14.29)	16.86 ± 0.0
	Map 3	MIMIC	**7 / 7 (100.0)**	**16.25 ± 6.08**
		ODYSSEY	1 / 7 (14.29)	26.01 ± 0.0
Obtain 1 Diamond	Map 1	MIMIC	7 / 7 (100.0)	53.49 ± 56.69
		ODYSSEY	64 / 64 (100.0)	**6.01 ± 2.14**
	Map 2	MIMIC	7 / 7 (100.0)	34.04 ± 23.38
		ODYSSEY	29 / 29 (100.0)	**8.26 ± 2.97**
	Map 3	MIMIC	7 / 7 (100.0)	37.77 ± 11.09
		ODYSSEY	23 / 23 (100.0)	**11.69 ± 5.05**

Time-Limited Task: With a fixed time budget for time-limited tasks, we compare only success rates. Both MIMIC and ODYSSEY achieved 100% across all tasks.

*4) **Task Solution Diversity (RQ4)**:* To analyze the diversity of task solutions, we collected all actions performed during task completion. We computed Shannon Entropy [69], treating each action as an individual data point to quantify variability in agent behaviour. Recognizing that the sequence of actions can also influence the game, we extended this analysis by applying n-gram-based Shannon Entropy [70], [71]. It treats subsequences of n consecutive actions as individual data points, providing a more nuanced evaluation of solution diversity. Higher Shannon Entropy indicates greater diversity.

As shown in Fig. 7, MIMIC and ODYSSEY perform similarly on simple goal-driven tasks (*Make 1 Sugar* and *Shear 1 Sheep*), which is expected since straightforward tasks leave little room for behavioural diversity. For more complex tasks and across all n-gram levels, MIMIC consistently exhibits higher entropy. To validate this statistically, we conducted one-tailed paired Student's t-tests for each task (Table V). All p-values are below 0.05, except for the simple tasks, confirming that MIMIC achieves significantly greater solution diversity than ODYSSEY in complex tasks. Overall, these results show that MIMIC generates diverse action sequences for complex scenarios while maintaining high success rates.

*5) **Discussion**:* Unlike MIMIC, which dynamically generates code, ODYSSEY directly invokes pre-coded functions (Skills) that align with the current plan. While this allows ODYSSEY to execute actions instantly, MIMIC requires more time to generate and refine code during execution.

Despite its faster performance, ODYSSEY demands a significantly larger upfront investment to develop and maintain a comprehensive Skill Library. This dependency also limits its flexibility in handling unforeseen or novel tasks generated by the Planner, reducing its capacity to test diverse interactions. In several cases, this led to imprecise task execution. For instance, when the plan specified "smelt iron ore into iron ingots" but no matching function was available, ODYSSEY repeatedly invoked the unrelated "mine raw iron" function based on similarity scores, resulting in incorrect and incomplete execution. This limitation also explains ODYSSEY's higher solution diversity in two simpler tasks, where it repeatedly performs unrelated actions due to misaligned function invocation. To confirm this limitation of ODYSSEY, we replaced the Skill Library of ODYSSEY with our example Skills. The results show that it failed to solve any tasks beyond the specified Skills. This constraint ultimately reduces its flexibility and precision in problem-solving and game-testing scenarios.

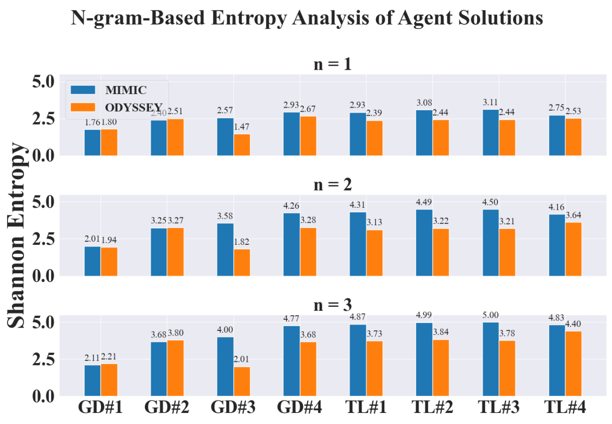


Fig. 7. Average Shannon Entropy of solutions generated by different agents in solving Minecraft tasks. Results are shown separately for N-grams with varying values of n. Each label stands for a task with an ID given in Table III.

TABLE V
RESULTS OF THE ONE-TAILED PAIRED STUDENT'S T-TEST. THE NULL HYPOTHESIS IS THAT MIMIC'S SOLUTION DIVERSITY IS LESS THAN OR EQUAL TO ODYSSEY'S FOR ACTION SUBSEQUENCES OF LENGTH $n \in [1, 3]$. P-VALUES < 0.05 (MARKED WITH "*") REJECT THE NULL, SHOWING THAT MIMIC ACHIEVES HIGHER SOLUTION DIVERSITY.

Task	P-value
Make 1 Sugar	0.643
Shear 1 Sheep	0.939
Cook 1 Meat	0.0132*
Obtain 1 Diamond	0.0487*
Combat 1 Cave Spider	0.0220*
Combat 1 Skeleton	0.0170*
Combat 1 Spider	0.0166*
Survive 1 Day	0.0247*

VI. RELATED WORK

Machine-Learning-Based Game Agents: Recent advances in game agents have leveraged reinforcement learning (RL) and imitation learning (IL). RL agents like OpenAI Five [9] and AlphaStar [72] excel in real-time strategy games via self-learning, often surpassing human players. IL approaches like AlphaGo [73] and VPT [66] improve learning efficiency by mimicking expert gameplay. However, both methods rely on primitive actions and predefined rewards or demonstrations, restricting exploration and adaptability [74], [75], making such tools often fail to explore diverse behaviours in games systematically. Furthermore, RL tools often depend on developer-defined rewards [76], introducing expert bias, whereas IL tools rely heavily on human demonstrations, limiting flexibility. Both methods often operate as black boxes with little interpretability of the decision-making process [8], hindering generalization to novel tasks.

Our approach addresses these gaps by integrating personality into an LLM-based Hybrid Planner that supports diverse strategies and richer exploration. Unlike prior black-box models, MIMIC offers transparent reasoning via prompt chaining, and demonstrates strong cross-game and cross-scenario adaptability through successful deployments and effective performance in three different games.

LLM-Based Game Agents*:* Recent advances highlight the potential of LLMs in game agents. Frameworks like ReAct [77] showcase LLMs' planning ability under dynamic conditions, while DEPS [34], ODYSSEY [18], Voyager [10], and others [11], [36], [78] tackle complex tasks in a large-scale game, Minecraft, building on prior successes in domains from board games [12] to video games [79]. Hybrid approaches like Auto MC-Reward [8], combining RL with LLM planning, further enhance adaptability and performance.

Most prior work optimizes agents for task completion, often producing homogeneous and repetitive behaviours. In contrast, we introduce MIMIC, an LLM-based agent explicitly designed for game testing. Leveraging a personality-driven Hybrid Planner, MIMIC adapts to varied environments through distinct personalities, promoting diverse exploration and mirroring how human testers evaluate games.

Agents Mimicking Human Behaviour*:* Previous studies have shown that LLMs can mimic personality traits [37]–[40], primarily through question-answering tasks. While effective in static text-based settings, these approaches lack functional agent implementations or practical applications. In contrast, MIMIC extends personality mimicking into games, enabling actionable behaviours beyond question answering.

Google's 2023 study [55] explored multi-agent simulations using LLMs with memory and social reasoning in a sandbox world. While showcasing complex social behaviour, it was limited to sandbox simulations without adaptability to real games. MIMIC builds on these ideas with a deployable, game-agnostic framework for testing real-world games through memory-driven, personality-conditioned planning.

PathOS [33], an RL-based agent, modelled personality through reward shaping for level design but focused only on navigation and lacked a full planning–execution pipeline. It also suffered typical RL limitations: restricted exploration, low interpretability, and poor generalization from manually tuned rewards. MIMIC addresses these gaps by combining LLM-based planning with a Memory System and Summarizer, enabling transparent, adaptive decision-making and flexible generalization across diverse games.

VII. Threats To Validity

LLM Selection*:* The choice of LLM may impact the validity of MIMIC's performance. This study uses GPT-4o (version 2024-08-06) [59] via API, incurring latency and monetary cost (thousands of USD for all experiments). Training data also limited its precision in game-specific reasoning and personality mimicking. Future work will address these challenges with locally hosted, fine-tuned LLMs optimized for gameplay and personality alignment. Despite these constraints, MIMIC achieved superior results in automated game testing.

Personality Trait Definitions*:* The personality prompts used by the Planner may influence the validity of MIMIC's performance. The seven personalities in MIMIC are directly taken from PathOS [33], which synthesizes traits grounded in real player behaviours from prior research. These traits and the behaviours of MIMIC were not independently validated against real player data, as MIMIC's goal is not to reproduce ground-truth human behaviour but to integrate personality traits to generate diverse behaviours. Although the prompts may not perfectly capture each trait, and LLMs cannot fully replicate human contextual adaptability, our evaluation shows that incorporating personality traits substantially increases action diversity. In practice, MIMIC consistently exhibited behaviours that were distinct and aligned with their respective traits across similar scenarios, demonstrating MIMIC's effectiveness despite this limitation.

Game Subject Selection*:* The selection of game subjects may affect the generalizability of our findings. This study focused on games that vary in scale and type but are limited to non-time-sensitive RPGs, specifically, dungeon crawlers, one of the most popular subgenres in this category. Future work will expand to other game types to further validate MIMIC's adaptability across broader gameplay contexts.

Human Tester Sample Size*:* We recruited seven human testers for the evaluation group. While a larger pool would improve generalizability, our sample size was limited by budget constraints. In DA, testers show identical code/branch coverage with variation in combinatorial coverage (Table VI). Because DA has few elements to cover, the numerical differences between testers remain small, making the influence on generalizability marginal. In SPD, variance appears across all coverage types. Although absolute variance in combinatorial coverage is small, the low overall human coverage makes relative variation appear larger. Since human testers serve only as a baseline for assessing MIMIC, the variance is not the focus of our evaluation. Nevertheless, when combined with the confidence interval comparisons in Section V-A, the substantial gap between MIMIC and humans in SPD remains robust. This suggests that additional human runs are unlikely to change the conclusion that, in complex environments, significant performance gaps persist between them.

TABLE VI
STANDARD DEVIATION AND VARIANCE ACROSS SEVEN HUMAN RUNS.

Game	Coverage Type	Standard Deviation	Variance
DA	Code	0.00%	0.00%
	Branch	0.00%	0.00%
	Combinatorial	4.45%	19.76%
SPD	Code	3.70%	13.71%
	Branch	3.14%	9.85%
	Combinatorial	0.093%	0.87%

Bug Detection Limitations: While MIMIC's Summarizer iteratively analyzes outputs during interaction, it can fail to detect or handle in-game bugs effectively, causing repeated task failures without crashing the game. Future work will incorporate game state analysis to improve bug detection. Meanwhile, developers should review consistently failed plans by MIMIC to identify potential bugs.

VIII. FUTURE WORK & IMPLICATIONS

Our findings highlight both the promise and the limitations of personality-driven agents for automated game testing. While MIMIC demonstrates clear improvements in behavioural diversity and coverage, humans still outperform it across many dimensions, revealing a substantial gap. Bridging this gap offers opportunities to advance game testing toward better reflecting diverse user behaviours.

A natural next step is to move beyond fixed personality prompts toward adaptive profiles that evolve with context. Leveraging its flexibility in integrating diverse personality forms, MIMIC can adapt in multiple ways. For example, the Memory System could periodically reflect on experiences to adjust behaviours over time, while traits could be learned from real player trajectories, with fine-tuned LLMs supporting more authentic evolution. These directions bring agents closer to human-like adaptability and position MIMIC as a foundation for broader research on personality-aligned game agents.

Another key direction is to broaden the environments in which such agents operate. Deploying MIMIC in a new game involves only minor adaptations to the Planner and the Action Executor. For the Planner, the prompt is adjusted to describe the game's mechanics and map in-game elements to personality traits (Section IV-A1). For the Executor, the available APIs should be exposed to allow MIMIC's interaction with the environment. We are also deploying MIMIC to additional game types, including larger-scale games such as non-turn-based RPGs and Massively Multiplayer Online (MMO) games.

However, inference latency limits its applicability in time-sensitive game types (e.g., First Person Shooter (FPS)). In our experiments, each action averaged 12.4 seconds, making MIMIC impractical for real-time deployment. The monetary cost is at \$0.06 USD/\$0.05 USD with/without code generation. At this rate, gameplay sessions with thousands of actions (e.g., SPD) could become expensive, limiting MIMIC for larger-scale use. To mitigate these challenges, we are exploring smaller, fine-tuned local models, and as LLMs continue to become faster and cheaper, MIMIC can play an increasingly impactful role in automated game testing.

MIMIC's results show that integrating personality traits into automated agents substantially improves behavioural diversity and coverage in game testing. For practitioners, MIMIC provides a practical tool to uncover edge cases and diverse usage patterns. For researchers, it introduces a new methodology for designing and evaluating test agents with personality in mind.

Looking forward, we see opportunities to expand this line of work. Beyond the direct directions already discussed, one avenue is to extend personality-driven agents within games, enriching user experiences through more realistic Non-Player Characters (NPCs). Beyond games, personality-aware automation also applies to domains like User Interface (UI) testing and Human-Computer Interactions (HCIs), where diverse navigation paths help uncover edge behaviours that traditional tools may miss. In this sense, MIMIC marks a step toward behaviourally rich, personality-aware testing methodologies across automated software engineering practice.

IX. CONCLUSION

Inspired by the diverse strategies of human players during gameplay, we introduced MIMIC, a novel testing framework that integrates personality traits into LLM-based gaming agents. Utilizing a Hybrid Planner to emulate varied in-game behaviours through a Memory System that accumulates experience, MIMIC enables personality-aligned decision-making, enhancing behavioural diversity and testing effectiveness.

We validated MIMIC on two open-source games of varying complexity, where it consistently outperformed random-based baselines and ablated versions in both code and interaction-level coverage. In Minecraft, it also surpassed a state-of-the-art LLM agent, achieving higher task success and greater strategic diversity. These results highlight MIMIC's ability to generate personality-driven actions across environments and drive the exploration toward more diverse scenarios.

Overall, MIMIC represents a substantial advance in automated game testing, introducing a scalable, personality-driven framework that adapts to dynamic environments and goals. Our results confirm its effectiveness and establish MIMIC as a powerful, generalizable tool for testing modern, complex games at scale.

Furthermore, MIMIC offers direct value by helping practitioners uncover edge cases and diverse usage patterns, and giving researchers a methodology for designing and evaluating personality-driven test agents. In a broader context, this work highlights the potential of personality-aware automation to enrich player experiences in games and to extend its capabilities to other domains such as UI testing and HCI.

ACKNOWLEDGMENT

The authors would like to thank ASE 2025 reviewers for their constructive comments. This work is supported by the Natural Sciences and Engineering Research Council of Canada Discovery Grant (Grant No. RGCPIN-2022-03744 and Grant No. DGECR-2022-00378) and Fonds de recherche du Québec-secteur Nature et technologies (Grant No.363482 [80]).

References

[1] T. Wijman, "Last looks: Our final 2023 games market estimates and forecasts," May 2024. [Online]. Available: https://newzoo.com/resources/blog/last-looks-the-global-games-market-in-2023

[2] "Game Testing Service Market Size & Share Trends, 2033." [Online]. Available: https://www.globalgrowthinsights.com/market-reports/game-testing-service-market-108174

[3] C. Politowski, F. Petrillo, and Y.-G. Guéhéneuc, "A survey of video game testing," in *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*. Madrid, Spain: IEEE, 2021, pp. 90–99.

[4] M. Ostrowski and S. Aroudj, "Automated regression testing within video game development," *GSTF Journal on Computing (JoC)*, vol. 3, no. 2, p. 10, 2013. [Online]. Available: https://doi.org/10.7603/s40601-013-0010-4

[5] I. S. W. B. Prasetya, F. P. Ricós, F. M. Kifetew, D. Prandi, S. Shirzadehhajimahmood, T. E. J. Vos, P. Paska, K. Hovorka, R. Ferdous, A. Susi, and J. Davidson, "An agent-based approach to automated game testing: An experience report," in *Proceedings of the 13th International Workshop on Automating Test Case Design, Selection and Evaluation (A-TEST '22)*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3548659.3561305

[6] J. Pfau, J. D. Smeddinck, and R. Malaka, "Automated game testing with icarus: Intelligent completion of adventure riddles via unsupervised solving," in *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '17 Extended Abstracts)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 153–164. [Online]. Available: https://doi.org/10.1145/3130859.3131439

[7] P. V. Amadori, T. Bradley, R. Spick, and G. Moss, "Robust imitation learning for automated game testing," 2024. [Online]. Available: https://arxiv.org/abs/2401.04572

[8] H. Li, X. Yang, Z. Wang, X. Zhu, J. Zhou, Y. Qiao, X. Wang, H. Li, L. Lu, and J. Dai, "Auto mc-reward: Automated dense reward design with large language models for minecraft," 2024. [Online]. Available: https://arxiv.org/abs/2312.09238

[9] OpenAI, 2019. [Online]. Available: https://openai.com/index/openai-five-defeats-dota-2-world-champions/

[10] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," 2023. [Online]. Available: https://arxiv.org/abs/2305.16291

[11] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang, Y. Qiao, Z. Zhang, and J. Dai, "Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory," 2023. [Online]. Available: https://arxiv.org/abs/2305.17144

[12] X. Feng, Y. Luo, Z. Wang, H. Tang, M. Yang, K. Shao, D. Mguni, Y. Du, and J. Wang, "Chessgpt: Bridging policy learning and language modeling," 2023. [Online]. Available: https://arxiv.org/abs/2306.09200

[13] Z.-J. Pang, R.-Z. Liu, Z.-Y. Meng, Y. Zhang, Y. Yu, and T. Lu, "On reinforcement learning for full-length game of starcraft," 2019. [Online]. Available: https://arxiv.org/abs/1809.09095

[14] W. Peng, M. Liu, and Y. Mou, "Do aggressive people play violent computer games in a more aggressive way? individual difference and idiosyncratic game-playing experience," *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, vol. 11, pp. 157–61, 05 2008.

[15] N. Yee, N. Ducheneaut, L. Nelson, and P. Likarish, "Introverted elves & conscientious gnomes: the expression of personality in world of warcraft," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 753–762. [Online]. Available: https://doi.org/10.1145/1978942.1979052

[16] N. C. Worth and A. S. Book, "Personality and behavior in a massively multiplayer online role-playing game," *Computers in Human Behavior*, vol. 38, pp. 322–330, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563214003446

[17] M. A. T. M. Corporation., "Minecraft," 2025. [Online]. Available: https://www.minecraft.net/en-us

[18] S. Liu, Y. Li, K. Zhang, Z. Cui, W. Fang, Y. Zheng, T. Zheng, and M. Song, "Odyssey: Empowering minecraft agents with open-world skills," 2024. [Online]. Available: https://arxiv.org/abs/2407.15325

[19] MIMIC-Persona, "Webpage of mimic," Jan 2025. [Online]. Available: https://mimic-persona.github.io/MIMIC-Home-Page/

[20] F. Duarte. (2024) Number of chatgpt users (mar 2024). Exploding Topics. [Online]. Available: https://explodingtopics.com/blog/chatgpt-users

[21] G. Marcus, "The next decade in ai: Four steps towards robust artificial intelligence," 2020. [Online]. Available: https://arxiv.org/abs/2002.06177

[22] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, "A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions," 2023.

[23] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021. [Online]. Available: https://arxiv.org/abs/2005.11401

[24] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu, "Evaluation of retrieval-augmented generation: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2405.07437

[25] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang, "Retrieval-augmented generation for large language models: A survey," 2024. [Online]. Available: https://arxiv.org/abs/2312.10997

[26] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit muds," *Journal of MUD research*, vol. 1, no. 1, p. 19, 1996. [Online]. Available: https://mud.co.uk/richard/hcds.htm

[27] A. S. Kahn, C. Shen, L. Lu, R. A. Ratan, S. Coary, J. Hou, J. Meng, J. Osborn, and D. Williams, "The trojan player typology: A cross-genre, cross-cultural, behaviorally validated scale of video game play motivations," *Computers in Human Behavior*, vol. 49, pp. 354–361, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0747563215002046

[28] T. W. Malone, "Toward a theory of intrinsically motivating instruction," *Cognitive Science*, vol. 5, no. 4, pp. 333–369, 1981. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog0504_2

[29] L. E. Nacke, C. Bateman, and R. L. Mandryk, "Brainhex: A neurobiological gamer typology survey," *Entertainment Computing*, vol. 5, no. 1, pp. 55–62, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1875952113000086

[30] A. Tychsen and A. Canossa, "Defining personas in games using metrics," in *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, ser. Future Play '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 73–80. [Online]. Available: https://doi.org/10.1145/1496984.1496997

[31] J. Vahlo, J. K. Kaakinen, S. K. Holm, and A. Koponen, "Digital Game Dynamics Preferences and Player Types," *Journal of Computer-Mediated Communication*, vol. 22, no. 2, pp. 88–103, 03 2017. [Online]. Available: https://doi.org/10.1111/jcc4.12181

[32] B. G. Weber and M. Mateas, "A data mining approach to strategy prediction," in *2009 IEEE Symposium on Computational Intelligence and Games*. Milan, Italy: IEEE, 2009, pp. 140–147.

[33] S. Stahlke, A. Nova, and P. Mirza-Babaei, "Artificial players in the design process: Developing an automated testing tool for game level and world design," in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 267–280. [Online]. Available: https://doi.org/10.1145/3410404.3414249

[34] Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang, "Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents," 2024. [Online]. Available: https://arxiv.org/abs/2302.01560

[35] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov, "Minerl: A large-scale dataset of minecraft demonstrations," 2019. [Online]. Available: https://arxiv.org/abs/1907.13440

[36] Z. Wang, S. Cai, A. Liu, Y. Jin, J. Hou, B. Zhang, H. Lin, Z. He, Z. Zheng, Y. Yang, X. Ma, and Y. Liang, "Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models," 2023. [Online]. Available: https://arxiv.org/abs/2311.05997

[37] L. L. Cava and A. Tagarelli, "Open models, closed minds? on agents capabilities in mimicking human personalities through open large language models," 2024. [Online]. Available: https://arxiv.org/abs/2401.07115

[38] G. Serapio-García, M. Safdari, C. Crepy, L. Sun, S. Fitz, P. Romero, M. Abdulhai, A. Faust, and M. Matarić, "Personality

traits in large language models," 2023. [Online]. Available: https://arxiv.org/abs/2307.00184

[39] A. Sorokovikova, N. Fedorova, S. Rezagholi, and I. P. Yamshchikov, "Llms simulate big five personality traits: Further evidence," 2024. [Online]. Available: https://arxiv.org/abs/2402.01765

[40] M. Miotto, N. Rossberg, and B. Kleinberg, "Who is gpt-3? an exploration of personality, values and demographics," 2022. [Online]. Available: https://arxiv.org/abs/2209.14338

[41] D. Gentner, "Structure-mapping: A theoretical framework for analogy," *Cognitive Science*, vol. 7, no. 2, pp. 155–170, 1983. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0364021383800093

[42] Y. Bouzzine, I. Tabiica, N. Galandi, and R. Lueg, "What can nudging offer to reduce workplace sexual harassment? a conceptual review," *World Development Sustainability*, vol. 4, p. 100149, 06 2024.

[43] D. Ariely, *Predictably Irrational: The Hidden Forces That Shape Our Decisions*. United States: HarperCollins, Feb. 2008, vol. 304.

[44] C. Potard, A. Henry, A. Boudoukha, R. Courtois, A. Laurent, and B. Lignier, "Video game players' personality traits: An exploratory cluster approach to identifying gaming preferences," *Psychology of Popular Media Culture*, vol. 9, no. 4, p. 499–512, May 2019.

[45] R. Habibi, J. Pfau, and M. S. El-Nasr, " Modeling Player Personality Factors from In-Game Behavior and Affective Expression ," in *2023 11th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*. Los Alamitos, CA, USA: IEEE Computer Society, Sep. 2023, pp. 1–8. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ACIIW59127.2023.10388138

[46] Z. Zhao, W. Chai, X. Wang, L. Boyi, S. Hao, S. Cao, T. Ye, and G. Wang, "See and think: Embodied agent in virtual environment," 2024. [Online]. Available: https://arxiv.org/abs/2311.15209

[47] M. Shridhar, X. Yuan, M.-A. Côté, Y. Bisk, A. Trischler, and M. Hausknecht, "ALFWorld: Aligning Text and Embodied Environments for Interactive Learning," in *Proceedings of the International Conference on Learning Representations (ICLR)*. Vienna, Austria: ICLR, 2021. [Online]. Available: https://arxiv.org/abs/2010.03768

[48] V. Gadesha and E. Kavlakoglu, "What is prompt chaining?" Dec. 2024. [Online]. Available: https://www.ibm.com/think/topics/prompt-chaining

[49] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023. [Online]. Available: https://arxiv.org/abs/2201.11903

[50] PrismarineJS, 2025. [Online]. Available: https://prismarinejs.github.io/mineflayer/#/

[51] F. Lin, D. J. Kim, Tse-Husn, and Chen, "When llm-based code generation meets the software development process," 2024. [Online]. Available: https://arxiv.org/abs/2403.15852

[52] U. Antero, F. Blanco, J. Oñativia, D. Salle, and B. Sierra, "Harnessing the power of large language models for automated code generation and verification," *Robotics*, vol. 13, p. 137, 09 2024.

[53] T. Lanham, A. Chen, A. Radhakrishnan, B. Steiner, C. Denison, D. Hernandez, D. Li, E. Durmus, E. Hubinger, J. Kernion, K. Lukošiūtė, K. Nguyen, N. Cheng, N. Joseph, N. Schiefer, O. Rausch, R. Larson, S. McCandlish, S. Kundu, S. Kadavath, S. Yang, T. Henighan, T. Maxwell, T. Telleen-Lawton, T. Hume, Z. Hatfield-Dodds, J. Kaplan, J. Brauner, S. R. Bowman, and E. Perez, "Measuring faithfulness in chain-of-thought reasoning," 2023. [Online]. Available: https://arxiv.org/abs/2307.13702

[54] W. Kojohnjaratkul, "Evaluate llms like gpt-4, gpt-3.5, and llama-2-70b-chat on their ability to respond to an increasing number of system prompt constraints." 2023. [Online]. Available: https://github.com/wiskojo/overwhelm-llm-eval

[55] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," 2023. [Online]. Available: https://arxiv.org/abs/2304.03442

[56] P. Todd, J. Ortega, J. Davis, G. Gigerenzer, D. Goldstein, A. Goodie, R. Hertwig, U. Hoffrage, K. Laskey, L. Martignon, and G. Miller, *Simple Heuristics That Make Us Smart*. Oxford, United Kingdom: Oxford University Press, Jan. 1999.

[57] Stelmaszczykadrian, "Github - stelmaszczykadrian/dungeon-adventures," Jun. 2023. [Online]. Available: https://github.com/stelmaszczykadrian/Dungeon-Adventures

[58] E. Debenham, "Shattered pixel dungeon," 2025. [Online]. Available: https://shatteredpixel.com/

[59] OpenAI, 2024. [Online]. Available: https://platform.openai.com/docs/models#gpt-4o

[60] I. Pearson Education, 2025. [Online]. Available: https://www.informit.com/articles/article.aspx?p=19806

[61] AndroidDev, 2023. [Online]. Available: https://developer.android.com/studio/test/other-testing-tools/monkey

[62] Y. Zheng, X. Xie, T. Su, L. Ma, J. Hao, Z. Meng, Y. Liu, R. Shen, Y. Chen, and C. Fan, "Wuji: Automatic online combat game testing using evolutionary deep reinforcement learning," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. San Diego, CA, USA: IEEE, 2019, pp. 772–784.

[63] EclEmma, Apr. 2024. [Online]. Available: https://www.eclemma.org/jacoco/index.html

[64] D. R. Kuhn, I. D. Mendoza, R. N. Kacker, and Y. Lei, "Combinatorial coverage measurement concepts and applications," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*. Luxembourg, Luxembourg: IEEE, 2013, pp. 352–361.

[65] D. Kuhn, R. Kacker, and Y. Lei, "Combinatorial coverage as an aspect of test quality," *CrossTalk*, vol. 28, pp. 19–23, 03 2015.

[66] B. Baker, I. Akkaya, P. Zhokhov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune, "Video pretraining (vpt): Learning to act by watching unlabeled online videos," 2022. [Online]. Available: https://arxiv.org/abs/2206.11795

[67] S. Liu, Y. Li, K. Zhang, Z. Cui, W. Fang, Y. Zheng, T. Zheng, and M. Song, Aug. 2024. [Online]. Available: https://huggingface.co/Aiwensile2/MineMA-8B

[68] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar, "Minedojo: Building open-ended embodied agents with internet-scale knowledge," 2022. [Online]. Available: https://arxiv.org/abs/2206.08853

[69] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, p. 379–423, Jul 1948.

[70] M. Damashek, "Gauging similarity with n-grams: Language-independent categorization of text," *Science*, vol. 267, no. 5199, pp. 843–848, 1995.

[71] S. Stich, "Entropy for n-grams," Aug. 2013. [Online]. Available: http://normal-extensions.com/2013/08/04/entropy-for-n-grams/

[72] M. Mathieu, S. Ozair, S. Srinivasan, C. Gulcehre, S. Zhang, R. Jiang, T. L. Paine, R. Powell, K. Żołna, J. Schrittwieser, D. Choi, P. Georgiev, D. Toyama, A. Huang, R. Ring, I. Babuschkin, T. Ewalds, M. Bordbar, S. Henderson, S. G. Colmenarejo, A. van den Oord, W. M. Czarnecki, N. de Freitas, and O. Vinyals, "Alphastar unplugged: Large-scale offline reinforcement learning," 2023. [Online]. Available: https://arxiv.org/abs/2308.03526

[73] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," 2017. [Online]. Available: https://arxiv.org/abs/1712.01815

[74] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "Go-explore: a new approach for hard-exploration problems," 2021. [Online]. Available: https://arxiv.org/abs/1901.10995

[75] J. Huizinga and J. Clune, "Evolving multimodal robot behavior via many stepping stones with the combinatorial multi-objective evolutionary algorithm," 2019. [Online]. Available: https://arxiv.org/abs/1807.03392

[76] P. Ladosz, L. Weng, M. Kim, and H. Oh, "Exploration in deep reinforcement learning: A survey," *Information Fusion*, vol. 85, p. 1–22, Sep. 2022. [Online]. Available: http://dx.doi.org/10.1016/j.inffus.2022.03.003

[77] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," 2023. [Online]. Available: https://arxiv.org/abs/2210.03629

[78] S. Lifshitz, K. Paster, H. Chan, J. Ba, and S. McIlraith, "Steve-1: A generative model for text-to-behavior in minecraft," 2024. [Online]. Available: https://arxiv.org/abs/2306.00937

[79] W. Ma, Q. Mi, Y. Zeng, X. Yan, Y. Wu, R. Lin, H. Zhang, and J. Wang, "Large language models play starcraft ii: Benchmarks and a chain of summarization approach," 2024. [Online]. Available: https://arxiv.org/abs/2312.11865

[80] "Combiner l'analyse statique et la génération de tests pour une assurance qualité améliorée : Une exploration des problèmes de compatibilité android," 2025.