D-ARK: Dynamic Action Planning for Role-Playing Games with Adaptive World Knowledge Graphs

Anonymous ACL submission

Abstract

Narrative-driven quests are central to player immersion in role-playing games (RPGs), but manually authoring coherent and responsive quests is costly and difficult to scale. Automated quest generation reduces this burden, yet existing systems often rely on predefined storylines or static knowledge bases, limiting their ability to reflect divergent player choices and persistent world changes. This weakens player autonomy and agency, which respectively concern the goals players can pursue and the lasting effects their choices have on subsequent narrative development. We propose **D-ARK**, a knowledge-graph-based framework that formulates RPG quest generation as action planning over evolving player and world states. D-ARK generates contextual quest goals, plans executable action sequences, and updates the world state so that player choices constrain subsequent quests. To support this framework, we construct a *Cyberpunk 2077*-based world and quest dataset from a public game wiki and introduce a consistency-preserving augmentation strategy that expands quest goals and progression paths beyond single-storyline data. Experiments show that D-ARK improves player autonomy and agency while maintaining sequential coherence, and that consistency-preserving augmentation improves action-sequence planning quality; component-level results further show gains in goal diversity and autonomy. These results suggest that coupling action-based quest planning with persistent world-state evolution can support scalable and player-responsive RPG content generation.

1 Introduction

"Baldur's Gate 3 pushes player freedom to the absolute limit... No two experiences are alike, and every character I've created feels unique."

GameSpot Review

Compelling narratives are central to player immersion and long-term engagement in role-playing games (RPGs). Rather than merely providing background lore, narratives give meaning to player actions, connect events into coherent progressions, and make game worlds feel consequential. In narrative-driven RPGs, this experience is primarily delivered through *quests*, which guide players along story progressions (Michailidis et al., 2018). By defining goals, actions, characters, locations, and consequences, quests decompose narrative progression into actionable gameplay units (Breault et al., 2021). However, manually designing large numbers of coherent and engaging quests is costly, time-consuming, and difficult to scale.

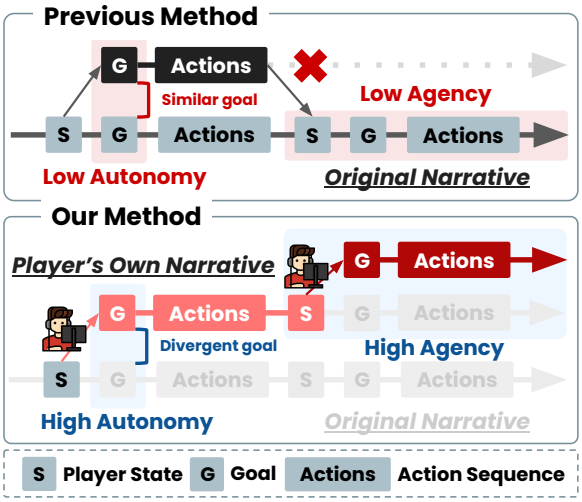

Figure 1: Comparison between predefined-narrative quest generation and D-ARK. Existing methods guide players along an original narrative, limiting diverse goals and player-driven consequences. D-ARK updates the world according to player choices, enabling divergent goals and player-owned narratives.

Automated quest generation has been studied to reduce authoring burden and expand game content (Chongmesuk and Kotrajaras, 2019; De Lima et al., 2021; Breault et al., 2021; Al-Nassar et al., 2023; Ashby et al., 2023). Early approaches rely on predefined quest structures, rules, or narrative templates (Chongmesuk and Kotrajaras, 2019;

De Lima et al., 2021; Al-Nassar et al., 2023), which help maintain consistency but restrict variation in goals, progression routes, and consequences. Recent LLM-based methods improve linguistic flexibility, yet they often remain tied to predefined storylines or static world knowledge (Ashby et al., 2023). Thus, they struggle to support player actions that diverge from the intended narrative path.

This limitation is especially problematic for RPGs, where player freedom is central to the experience. When players make divergent choices, such as betraying an allied character or opposing a friendly settlement, the game world should change accordingly, and later quests should reflect these consequences. However, existing quest generation systems often lack a mechanism for representing the current player–world state, generating alternative goals from that state, planning actions toward the selected goal, and updating world knowledge after the quest is completed. This prevents them from fully supporting player *autonomy*, which concerns what goals the player can pursue, and player *agency*, which concerns whether player choices produce persistent effects on the world and subsequent narrative development.

To address this limitation, we propose D-ARK: Dynamic Action Planning for Role-playing Games with Adaptive World Knowledge Graphs. To our knowledge, D-ARK is the first game quest generation framework that unifies *state construction*, *goal generation*, *action planning*, and *world knowledge update*. Given the current player–world state, D-ARK generates plausible candidate goals, plans executable action sequences, and updates a mutable world knowledge graph with the resulting consequences. This iterative process allows prior player choices to shape future quests and narrative progression.

To implement and evaluate D-ARK, we construct and release a quest-generation dataset derived from a public wiki of *Cyberpunk 2077*. The dataset captures a richly structured RPG world, including characters, locations, items, events, and quests. Since the source game follows a largely canonical storyline, we introduce a consistency-preserving augmentation strategy to expand quest goals and progression paths while maintaining compatibility with the world state.

Our experiments evaluate D-ARK at both the player and component levels. We first examine whether D-ARK improves player autonomy and agency, with a focused analysis of how world knowledge update preserves player-driven consequences across quest sequences. We then evaluate action planning and goal generation to assess whether D-ARK produces coherent action sequences and plausible, diverse candidate goals. The results show that D-ARK's benefits arise from the integration of state construction, goal generation, action planning, and world knowledge update.

Our contributions: (1) We formulate RPG quest generation as an iterative stateful process for supporting player autonomy and agency. (2) We propose D-ARK, the first game quest generation framework that integrates state construction, goal generation, action planning, and world knowledge update into a unified knowledge-graph-based process. (3) We construct and release a *Cyberpunk 2077*-based quest-generation dataset with consistency-preserving augmentation for expanding single-storyline quest data. (4) We show through player-level and component-level evaluations that D-ARK improves autonomy and agency, preserves player-driven consequences through world knowledge update, and produces coherent action plans and plausible candidate goals.

2 Related Work

In Procedural Content Generation (PCG), automated quest generation has been studied through rule-based and planning-based methods over statically defined game-world information (Lee and Cho, 2012; De Lima et al., 2021; Prins et al., 2023). By modeling game-world rules and causal relations, these approaches can generate logically consistent quests. However, they typically operate within fixed rule sets and goal spaces, focusing on individual quests or limited branching. This makes it difficult to adapt quest goals to the player's evolving state and choices, thereby limiting player autonomy. Recent LLM-based approaches improve linguistic flexibility and support free-form player input, often using knowledge graphs or templates to maintain consistency (Ashby et al., 2023). Nevertheless, their quest structures are still largely constrained by predefined templates, rules, or static knowledge sources. Because generation remains centered on isolated quest instances, player choices rarely accumulate into persistent world-state changes that shape later quests, limiting player agency. Overall, existing approaches can produce logically valid quests, but they remain limited in supporting diverse, state-grounded goal selection and persistent world-state evolution across quest sequences.

3 Game Quest Generation Formulation

In this section, we formulate RPG quest generation as a state-conditioned planning problem. Following prior work that views quests as ordered task or action sequences (Breault et al., 2021; Ashby et al., 2023), we define a quest as a goal-directed action sequence grounded in the current game state.

State. Let s_t denote the current player–world state at step t. The state represents the context from which both the quest goal and action sequence are generated. In this work, s_t consists of `Location`, `Possessions`, `Accessible Areas`, and `Relevant NPCs`, which jointly determine what goals are appropriate and what actions are feasible.

Goal. Given the current state s_t, quest generation first selects a state-conditioned goal g_t:

$$g_t = f_G(s_t), \tag{1}$$

where f_G denotes a goal generation function. The goal specifies the objective to be achieved through the quest and guides subsequent action planning. Because g_t is generated from s_t rather than selected from a fixed template, the quest can reflect the player's current location, possessions, accessible areas, and relevant NPCs.

Action Sequence. Conditioned on both the current state s_t and the selected goal g_t, the system generates an action sequence $\mathbf{a}_t = (a_1, a_2, \ldots, a_n)$:

$$\mathbf{a}_t = f_A(s_t, g_t), \tag{2}$$

where f_A denotes an action planning function. Each action should be feasible under the current or intermediate state and should contribute to achieving the goal. The resulting quest is defined as:

$$q_t = (g_t, \mathbf{a}_t). \tag{3}$$

Executing the action sequence transforms the current state into a subsequent state:

$$s_t \xrightarrow{\mathbf{a}_t} s_{t+1}, \tag{4}$$

where the updated state s_{t+1} serves as the context for future quests. This allows player choices and quest outcomes to persist over time rather than being confined to isolated quest instances.

4 D-ARK

4.1 Overview

D-ARK is an iterative quest generation framework that grounds each quest in the current player–world state and updates world knowledge according to quest outcomes. As illustrated in Figure 2, D-ARK consists of four stages: state construction and verification, state-conditioned goal generation, action sequence planning, and world knowledge update. It uses LightRAG (Guo et al., 2025) as the knowledge-graph-based module for retrieving and updating world knowledge, while LLM-based modules perform state refinement, goal generation, action planning, and update decision making. Detailed implementation settings, including model choices and the knowledge-graph construction procedure, are provided in Appendix A.

At each iteration, D-ARK retrieves the information required to construct the current state from the world knowledge and refines it against the latest graph. It then generates multiple candidate goals, lets the player select one, and plans an action sequence for the selected goal. After quest completion, D-ARK extracts the resulting changes and incorporates them back into the knowledge graph, which is then used for subsequent iterations.

4.2 State Construction and Verification

D-ARK represents world knowledge at the t-th quest generation step as a knowledge graph $\mathcal{G}_t$, whose nodes and edges describe entities and relations about characters, locations, items, events, and quests. Using LightRAG (Guo et al., 2025), D-ARK retrieves relevant graph information from $\mathcal{G}_t$ and constructs a provisional state $\tilde{s}_t$ that includes the player's location, possessions, accessible areas, and relevant NPCs.

Since $\tilde{s}_t$ is produced through retrieval and extraction, it may contain noisy, incomplete, or inconsistent elements. The State Verifier checks $\tilde{s}_t$ against $\mathcal{G}_t$ and minimally revises invalid components to obtain the final state s_t. This verification step ensures that subsequent goal generation and action planning are conditioned on a state consistent with the current world knowledge.

4.3 State-conditioned Goal Generation

Given the verified state s_t, the Goal Generator, implemented as an LLM-based module, produces a set of candidate goals:

$$\mathcal{C}_t = \{g_t^1, g_t^2, \ldots, g_t^K\}, \tag{5}$$

where each candidate g_t^k is conditioned on the current state and grounded in the retrieved world knowledge. Unlike template-based approaches, the module does not retrieve a predefined quest objective; instead, it generates multiple possible objectives that are valid under s_t.

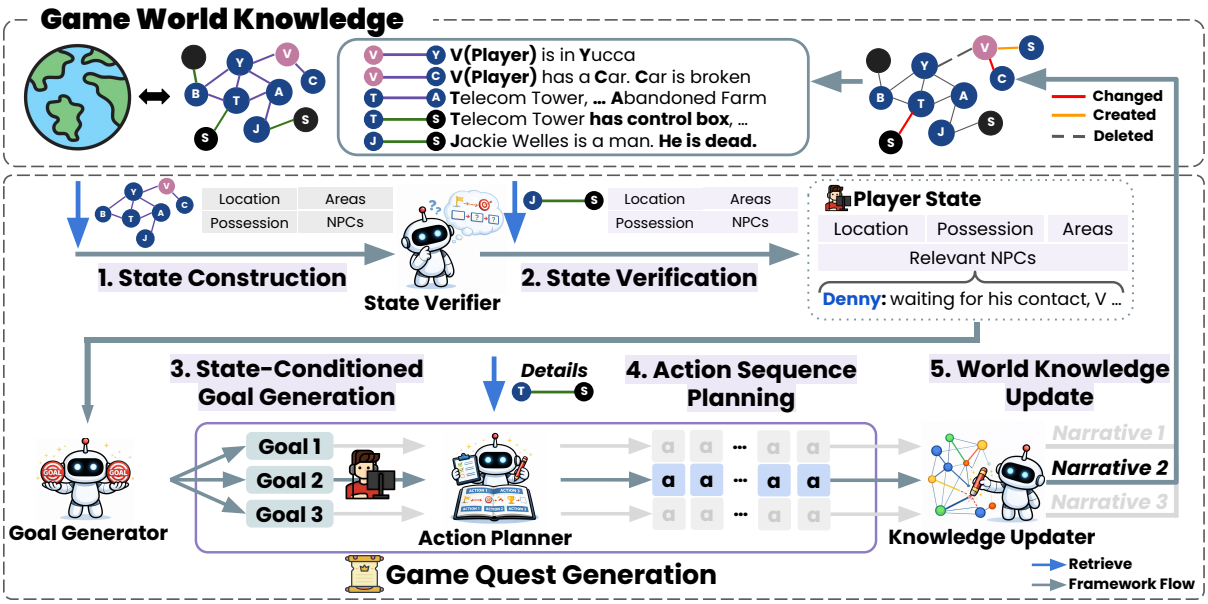

Figure 2: Overview of D-ARK, our game quest generation framework. It interacts with the player to generate context-tailored quests, and completed quests influence subsequent quest generation.

The candidate set $\mathcal{C}_t$ is presented to the player, who selects one goal g_t as the objective of the next quest. The selected goal is then passed to the Action Planner. This design makes goal selection an explicit part of the interaction loop, allowing D-ARK to expose multiple valid continuations from the same state rather than committing to a single system-chosen objective.

4.4 Action Sequence Planning

Given the verified state s_t and the player-selected goal g_t, D-ARK retrieves world details r_t from the current knowledge graph $\mathcal{G}_t$ as grounding context for action planning. The Action Planner, implemented as an LLM-based module, uses s_t, g_t, and r_t to generate an action sequence:

$$\mathbf{a}_t = (a_t^1, a_t^2, \ldots, a_t^N). \tag{6}$$

Here, r_t contains relevant entity, relation, and constraint information needed to keep the planned actions consistent with the current world knowledge.

Each action a_t^i should be feasible under the current or intermediate state and contribute to achieving g_t. By grounding the sequence in r_t, D-ARK reduces the risk of generating actions involving unavailable NPCs, inaccessible areas, or inconsistent world conditions. This allows the Action Planner to produce flexible quest progressions without relying on fixed quest templates.

4.5 World Knowledge Update

After quest completion, the Knowledge Updater extracts world changes Δ_t from the executed action sequence, including updates to player possessions, NPC states, and entity relations. D-ARK applies Δ_t to the LightRAG-managed graph $\mathcal{G}_t$, producing the updated graph $\mathcal{G}_{t+1}$.

The updated graph $\mathcal{G}_{t+1}$ is reloaded into LightRAG and used as the retrieval source for the next iteration. Through this loop, D-ARK conditions later quests on the accumulated consequences of previous player actions.

5 Datasets

This section describes the dataset constructed to support D-ARK. We build a structured quest dataset from the Cyberpunk 2077 Wiki[1], where each quest is paired with world knowledge about relevant characters, locations, items, events, and quest dependencies. The wiki provides quest descriptions, walkthroughs, quest relations, and entity-level information used to construct this dataset. Detailed data collection procedures, prompt templates, and verification methods are provided in the Appendix C.

5.1 Dataset Collection and Construction

We collect quests from four quest types: Main, Minor, Side, and Gig. For each quest, we extract its de-

[1] https://cyberpunk.fandom.com/wiki/Cyberpunk_Wiki

scription, walkthrough, progression structure, and related quest links. The extracted information is converted into a structured representation aligned with Section 3, including the quest state, goal, and action sequence.

We also construct world knowledge that grounds each quest in the game world. Rather than treating it as a separate dataset, we use world knowledge as contextual information associated with each quest instance. It captures entities and relations involving characters, locations, items, events, and quest dependencies, enabling D-ARK to maintain consistency across quest iterations.

Since D-ARK updates world knowledge as quests progress, the knowledge graph must remain consistent across connected quest sequences. We therefore focus connected world knowledge construction on Main Quests, where inter-quest dependencies are explicit and changes in characters, locations, items, and events can be traced over time. As a result, the dataset consists of structured quest instances grounded in a connected knowledge graph rather than isolated quest descriptions.

5.2 Dataset Augmentation

The original wiki data primarily describe canonical quest progressions. Although these data provide high-quality quest structures, they offer limited coverage of alternative goals and divergent player-driven states. This limits their usefulness for training adaptive quest generation models.

To address this limitation, we augment the training data in two ways. First, for a fixed state, we generate alternative goals and revise the corresponding action sequences. Second, we perturb the state and generate new goals and action sequences conditioned on the modified state. Together, these strategies expand the range of state–goal–action mappings beyond the canonical storyline.

The augmented data are used only for training the Goal Generator and Action Planner. From 2,070 augmentation candidates generated from the 207 original quest instances, 2,025 pass filtering. After reserving 100 original instances as test data, the augmented training set contains 2,132 instances: 107 original and 2,025 augmented instances.

6 Experiments

6.1 Experimental Overview

We evaluate *D-ARK* from three complementary perspectives aligned with our main claims. First, we conduct a framework-level evaluation of sequential quest generation to examine whether *D-ARK* supports player autonomy, agency, and coherence across multi-quest trajectories. Second, we evaluate the Action Planner to test whether it produces goal-directed action sequences that follow human-like quest progressions. Third, we evaluate the Goal Generator to assess whether it produces plausible and narratively distinct candidate goals that expand the player's choice space.

Unless otherwise specified, all experiments use LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen2.5-7B-Instruct (Yang et al., 2024) as backbone LLMs. Because these evaluations target different aspects of the framework, each experiment uses task-specific baselines, metrics, and protocols. We report automatic metrics where applicable, use GPT-based judgments for narrative-level criteria, and conduct human verification to assess the reliability of GPT-based evaluation. The augmented data are used only for training, and all evaluations are conducted on held-out, non-augmented scenarios. Detailed model settings, prompts, and evaluation criteria are provided in Appendix A.

6.2 Framework-Level Evaluation of Player Autonomy and Agency

Objective. We evaluate whether *D-ARK* supports player autonomy and agency in sequential quest generation. A framework-level quest generator should produce meaningfully different quest trajectories from the same initial state, while preserving the consequences of earlier quests in later ones. We therefore evaluate *autonomy*, which measures the diversity of narrative paths across repeated playthroughs; *agency*, which measures whether player-driven consequences persist across quests; and *coherence*, which measures whether each quest sequence remains logically connected.

Setup. Starting from the same player state, each model generates a quest, updates the player–world context, and then generates the next quest from the updated context. We repeat this process for five steps to obtain a five-quest sequence, and generate K such sequences from the same initial state. For each backbone, we compare the zero-shot backbone LLM, *D-ARK*, and *D-ARK* without world knowledge update (*w/o WKU*). The zero-shot setting tests sequential quest generation without task-specific training, while the no-update variant disables the update-based state-maintenance loop to assess the effect of persistent world-state updates.

Model		Autonomy		Agency		Coherence	
		Human.	GPT	Human.	GPT	Human.	GPT
LLaMA-3.1	Zero-shot	1.57	3.00	1.74	1.80	2.17	3.40
	D-ARK w/o WKU	3.28	**5.00**	3.11	2.40	2.62	3.20
	D-ARK	**4.71**	**5.00**	**3.68**	**3.20**	**2.94**	**4.00**
Qwen2.5	Zero-shot	1.85	2.00	1.68	2.00	**2.91**	**4.00**
	D-ARK w/o WKU	3.57	**5.00**	2.94	2.60	2.54	3.60
	D-ARK	**4.57**	**5.00**	**3.03**	**3.80**	2.86	3.80

Table 1: Framework-level evaluation of five-step quest sequences in terms of autonomy, agency, and coherence. WKU denotes world knowledge update.

We evaluate autonomy, agency, and coherence using GPT-based judgments and human ratings.

Results. As shown in Table 1, *D-ARK* improves autonomy and agency across both backbone LLMs. The autonomy gains suggest that state-conditioned goal generation helps avoid repeatedly following a single dominant storyline. The agency gains are most evident against *D-ARK w/o WKU*, indicating that writing quest outcomes back to the world knowledge graph helps later quests reflect accumulated player-driven changes. We further validate the consistency of world knowledge updates using the QA-based check in Appendix B.3.

For coherence, *D-ARK* remains broadly comparable to the baselines while improving autonomy and agency. This suggests that diverse, consequence-aware quest generation can be achieved without severely compromising narrative continuity, although maintaining smooth long-horizon progression under divergent player choices remains challenging.

6.3 Action Sequence Quality under Training Augmentation

Objective. After a quest goal is selected, the quest is realized through an action sequence. We evaluate whether the Action Planner generates ordered, goal-directed action sequences that form well-structured quest progressions. In this experiment, similarity to the human-labeled sequence is used to assess planning structure, such as whether the generated sequence preserves necessary interactions and their order. This metric is not intended to reward adherence to the canonical storyline, but to measure whether the generated actions form a coherent progression toward the given state-conditioned goal.

Setup. For each backbone, we compare the zero-shot LLM with the fine-tuned Action Planner initialized from the same backbone, isolating the effect of task-specific training on state–goal–action generation. We also include GRASE-DC (Zhao et al., 2025) and LLM-REGRESS (Liu et al., 2025) as training-free reference baselines to contextualize the difficulty of quest action planning. Because these baselines are not trained on our data, they serve as capability reference points rather than controlled fine-tuning comparisons.

Model		Similarity		Diversity	
		Auto.	GPT	Auto.	GPT
Human-labeled		–	–	11.31	3.61
Baselines	GRASE-DC	0.29	2.29	12.51	4.28
	LLM-REGRESS	0.21	2.34	16.04	4.85
LLaMA-3.1	Zero-shot	0.24	2.17	14.32	4.57
	Action Planner	0.45	2.92	11.54	4.16
Qwen2.5	Zero-shot	0.25	2.27	13.91	4.62
	Action Planner	0.44	2.83	11.56	4.27

Table 2: Action sequence quality comparison across backbone LLMs and zero-shot baselines.

Model		Similarity		Diversity	
		Auto.	GPT	Auto.	GPT
Human-labeled		–	–	11.31	3.61
LLaMA-3.1	Action Planner	0.45	2.92	11.54	4.16
	w/o Aug.	0.12	1.43	5.32	2.61
	w/o Details	0.43	2.90	11.98	4.31
Qwen2.5	Action Planner	0.44	2.83	11.56	4.27
	w/o Aug.	0.19	1.72	9.40	3.60
	w/o Details	0.41	2.79	11.89	4.28

Table 3: Ablation study on training augmentation and retrieved world details for action sequence generation.

We evaluate generated action sequences using similarity and diversity. Similarity measures whether the generated sequence follows the progression of the human-labeled action sequence for the given state and goal. Diversity measures the variety of gameplay interactions in the generated sequence. However, diversity is interpreted relative to the human-labeled reference rather than as a strictly higher-is-better metric: overly low diversity may indicate incomplete plans, whereas overly high diversity may indicate redundant or unfocused interactions. For both criteria, we report automatic metrics and GPT-based judgments.

Results. As shown in Table 2, the fine-tuned Action Planner improves similarity over the zero-shot backbones and reference baselines across both LLaMA-3.1 and Qwen2.5. It also produces diversity scores closer to the human-labeled reference, suggesting that task-specific training helps generate well-structured quest progressions without producing overly sparse or redundant action sequences.

Table 3 shows that training augmentation provides the largest gain, while retrieved world details

Model		Plausibility		Diversity		Autonomy
		Auto.	GPT	Auto.	GPT	GPT
Baselines	G2	0.74	4.72	0.25	1.79	1.78
	Avoidance	0.76	4.38	0.02	1.05	1.05
LLaMA-3.1	Zero-shot	0.75	4.80	0.14	1.30	1.55
	Goal Generator	0.72	4.81	0.25	3.31	3.23
Qwen2.5	Zero-shot	0.73	4.67	0.09	1.25	1.44
	Goal Generator	0.72	4.74	0.25	3.08	3.16

Table 4: Goal generation performance comparison across backbone LLMs and zero-shot baselines.

Model		Spearman	
		ρ	p
Baselines	G2	.675	$< .001$
	Avoidance	.789	$< .001$
LLaMA-3.1	Zero-shot	.586	$< .001$
	Goal Generator	.768	$< .001$
Qwen2.5	Zero-shot	.626	$< .001$
	Goal Generator	.855	$< .001$

Table 5: Spearman rank correlations between goal diversity and autonomy (n=100)

provide additional grounding. Without augmentation, similarity drops sharply, indicating that the original non-augmented data provide limited coverage of state–goal–action mappings. Retrieved details further improve grounding by conditioning the plan on the current player–world context. Although removing details can sometimes increase diversity, this reflects less constrained variation rather than better planning quality.

6.4 Candidate Goal Generation and Player Autonomy

Objective. We evaluate whether D-ARK's Goal Generator supports player autonomy by producing plausible and narratively distinct candidate goals from the same player state.

Setup. For each backbone, we compare the zero-shot LLM with D-ARK's fine-tuned Goal Generator initialized from the same backbone. We also include G2 (Ruan et al., 2025) and Avoidance (Park et al., 2025) as training-free reference baselines for diverse goal generation. Because these baselines are not trained on our data, they are used as capability reference points rather than controlled fine-tuning comparisons.

For each input state, each model generates a set of goals. We evaluate the goals using plausibility, diversity, and autonomy. Plausibility measures state grounding, diversity measures distinct narrative directions, and autonomy measures whether the goal set provides meaningful freedom of choice.

Module	Metric	GPT–Human (ρ)	Human Agr. (ICC)
Action Planner	Similarity	0.80^{**}	0.55
	Diversity	0.70^{**}	0.64
Goal Generator	Plausibility	0.50^{*}	0.38
	Diversity	0.83^{**}	0.74
	Autonomy	0.88^{**}	0.74
Framework (D-ARK)	Coherence	0.53^{*}	0.25
	Agency	0.58^{**}	0.42
	Autonomy	0.78	0.72

Table 6: Human verification of GPT-based evaluation. Spearman correlation coefficients (ρ) between GPT scores and human mean ratings are reported, together with inter-human agreement measured by ICC(2,1). Statistical significance for Spearman correlation is denoted as $^{*}p < 0.05$ and $^{**}p < 0.01$.

Results. As shown in Table 4, D-ARK's fine-tuned Goal Generator substantially improves diversity and autonomy while maintaining high plausibility. These gains indicate that task-specific training on augmented goal data helps produce less redundant goal sets with more meaningful narrative alternatives. Although automatic plausibility is slightly lower than some baselines, it remains high across both backbones. Because the automatic plausibility metric relies on lexical overlap with the state, this slight drop does not necessarily indicate weaker contextual grounding. Instead, the results suggest that D-ARK improves autonomy by generating plausible and narratively distinct goals rather than arbitrary alternatives.

Analysis. Table 5 shows significant positive correlations between goal diversity and autonomy across all settings. This supports our design intuition that distinct narrative directions are closely associated with perceived player autonomy.

6.5 Human Verification of GPT-based Evaluation

Because several experiments rely on GPT-based judgments, we verify their alignment with human ratings. Seven graduate-level annotators rated anonymized and randomly ordered model outputs on a 5-point scale using task-specific criteria. Each item included the player-state and quest context, so prior Cyberpunk 2077 experience was not required. Human ratings covered 30 samples each for the Goal Generator and Action Planner, 30 framework-level samples for coherence and agency, and six aggregate framework-level autonomy comparisons.

For each metric, we report Spearman correlations (Spearman, 1961) between GPT scores and mean human ratings, together with inter-annotator agreement measured by ICC(2,1) (Shrout and Fleiss, 1979). As shown in Table 6, GPT-based

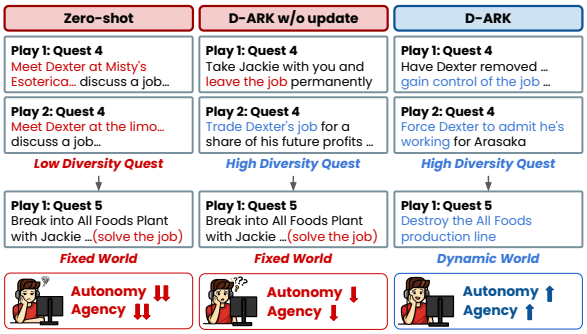

Figure 3: Case study of generated quest sequences. D-ARK generates diverse goals and preserves their consequences through world knowledge updates.

scores are positively correlated with human judgments across all metrics, with significant correlations for all metrics except framework-level autonomy, where the small number of aggregate samples limits statistical power. GPT–human alignment is strongest for goal diversity, goal autonomy, and action-sequence similarity.

Inter-annotator agreement is moderate for action-level evaluations and goal diversity/autonomy, but lower for goal plausibility and framework-level coherence and agency. This reflects the multi-factor nature of these criteria, where annotators may weigh state grounding, transition smoothness, and consequence preservation differently. Overall, these results support the use of GPT-based judgments as comparative indicators, while also showing that long-horizon narrative qualities remain challenging to evaluate consistently.

6.6 Case Study

We present qualitative case studies comparing gameplay and action sequences across quest generation methods, using LLaMA-3.1-8B-Instruct as the backbone LLM for all outputs.

Framework Case Study. Figure 3 illustrates how the compared settings differ in autonomy and agency. The zero-shot model stays close to the original Dexter-centered quest flow, producing similar quests across playthroughs. D-ARK without world knowledge update generates more diverse Quest 4 goals, but Quest 5 returns to a fixed-world progression because previous outcomes are not propagated. In contrast, D-ARK generates distinct goals and updates the world state accordingly, so later quests continue from earlier player choices rather than resetting to the original route. This example shows the qualitative difference between generating diverse goals locally and preserving their consequences across subsequent quests.

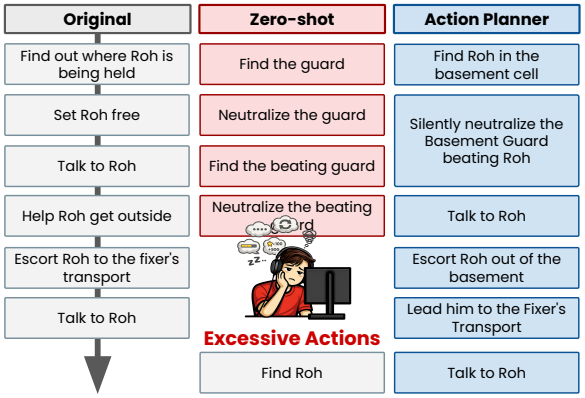

Figure 4: Case study of generated action sequences. The Action Planner produces better-structured progressions while avoiding excessive or redundant actions.

Action Sequence Case Study. Figure 4 compares generated action sequences for the same quest goal. The zero-shot model produces relevant but redundant actions, repeatedly finding and neutralizing guards before reaching Roh. In contrast, the Action Planner follows a more direct and goal-oriented progression: finding Roh in the basement cell, neutralizing the immediate guard threat, talking to Roh, escorting him out, and leading him to the transport. This example illustrates how the Action Planner produces a better-structured action sequence while avoiding excessive or redundant steps.

7 Conclusion

This paper presents D-ARK, a dynamic quest generation framework for supporting player autonomy and agency in role-playing games. D-ARK integrates state-conditioned goal generation, action sequence planning, and persistent world knowledge updates within an iterative player–world state loop. By updating the knowledge graph after each quest, D-ARK allows subsequent quests to reflect accumulated player choices rather than reverting to a fixed storyline. Experiments show that D-ARK improves autonomy and agency while largely preserving coherence in sequential quest generation. They also show that consistency-preserving augmentation supports diverse and plausible goal generation as well as well-structured action planning. These findings indicate that mutable world knowledge is a key component of scalable, player-responsive RPG quest generation. Future work will explore richer world representations, more detailed modeling of player intent, and integration with game engines.

Limitations

While our results show that D-ARK improves autonomy and agency in sequential quest generation by combining state-conditioned goal generation, action planning, and persistent world knowledge updates, this work has several limitations. First, our dataset and experiments are limited to a Cyberpunk 2077-derived RPG setting, so further validation is needed to assess generalization to other genres, narrative structures, and game worlds. Second, our evaluation focuses on generated goals, action sequences, and sequential quest trajectories, rather than deployment in a fully interactive game engine or a long-term player study. Therefore, we do not yet measure real-time player behavior, engagement, or gameplay experience. Third, D-ARK relies on LLM-based modules and knowledge-graph updates, which may still produce incomplete or incorrect state changes when quest outcomes are ambiguous. Although the framework includes verification and update mechanisms to reduce inconsistencies, maintaining robust world-state validity over long gameplay sessions remains an important direction for future work.

Despite these limitations, our findings suggest that coupling action-based planning with mutable world knowledge is a promising direction for scalable and player-responsive RPG quest generation.

References

Suzan Al-Nassar, Anthonie Schaap, Michael Van Der Zwart, Mike Preuss, and Marcello A Gómez-Maureira. 2023. Questville: Procedural quest generation using nlp models. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, pages 1–4.

Trevor Ashby, Braden K Webb, Gregory Knapp, Jackson Searle, and Nancy Fulda. 2023. Personalized quest and dialogue generation in role-playing games: A knowledge graph-and language model-based approach. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–20.

Vincent Breault, Sebastien Ouellet, and Jim Davies. 2021. Let conan tell you a story: Procedural quest generation. *Entertainment Computing*, 38:100422.

Soumyabrata Chaudhuri, Pranav Purkar, Ritwik Raghav, Shubhojit Mallick, Manish Gupta, Abhik Jana, and Shreya Ghosh. 2025. TripCraft: A benchmark for spatio-temporally fine grained travel planning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria. Association for Computational Linguistics.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand. Association for Computational Linguistics.

Thongtham Chongmesuk and Vishnu Kotrajaras. 2019. Multi-paths generation for structural rule quests. In *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 97–102. IEEE.

Edirlei Soares De Lima, Bruno Feijó, and Antonio L Furtado. 2021. Adaptive branching quests based on automated planning and story arcs. In *2021 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 9–18. IEEE.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, Suzhou, China. Association for Computational Linguistics.

Young-Seol Lee and Sung-Bae Cho. 2012. Dynamic quest plot generation using petri net planning. In *Proceedings of the Workshop at SIGGRAPH Asia*, pages 47–52.

Lei Li, Yongfeng Zhang, and Li Chen. 2020. Generate neural template explanations for recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 755–764.

Xiaotian Liu, Ali Pesaranghader, Hanze Li, Punyaphat Sukcharoenchaikul, Jaehong Kim, Tanmana Sadhu, Hyejeong Jeon, and Scott Sanner. 2025. Open-world planning via lifted regression with LLM-inferred affordances for embodied agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria. Association for Computational Linguistics.

Lazaros Michailidis, Emili Balaguer-Ballester, and Xun He. 2018. Flow and immersion in video games: The aftermath of a conceptual challenge. *Frontiers in psychology*, 9:1682.

Kyeongman Park, Nakyeong Yang, and Kyomin Jung. 2025. Avoidance decoding for diverse multi-branch story generation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language*

Processing, Suzhou, China. Association for Computational Linguistics.

Vincent L Prins, Jelmer Prins, Mike Preuss, and Marcello A Gómez-Maureira. 2023. Storyworld: Procedural quest generation rooted in variety & believability. In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, pages 1–4.

Zhiwen Ruan, Yixia Li, Yefeng Liu, Yun Chen, Weihua Luo, Peng Li, Yang Liu, and Guanhua Chen. 2025. G2: Guided generation for enhanced output diversity in LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Suzhou, China. Association for Computational Linguistics.

Patrick E Shrout and Joseph L Fleiss. 1979. Intraclass correlations: uses in assessing rater reliability. *Psychological bulletin*, 86(2):420.

Charles Spearman. 1961. The proof and measurement of association between two things.

Guy Tevet and Jonathan Berant. 2021. Evaluating the evaluation of diversity in natural language generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Online. Association for Computational Linguistics.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025a. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Ching-Wen Yang, Zhi-Quan Feng, Ying-Jia Lin, Che Wei Chen, Kun-da Wu, Hao Xu, Yao Jui-Feng, and Hung-Yu Kao. 2025b. MAPLE: Enhancing review generation with multi-aspect prompt LEarning in explainable recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria. Association for Computational Linguistics.

Xinran Zhao, Hanie Sedghi, Bernd Bohnet, Dale Schuurmans, and Azade Nazi. 2025. Improving large language model planning with action sequence similarity. In *International Conference on Learning Representations*, volume 2025, pages 71261–71285.

Appendix

A Experimental Details

A.1 Model and Framework Configuration

D-ARK. D-ARK represents the mutable game world as a GraphML knowledge graph managed by LightRAG (Guo et al., 2025), using bge-m3 (Chen et al., 2024) as the embedding model and Qwen3-32B (Yang et al., 2025a) as the LLM used by LightRAG. At each quest-generation loop, the graph is loaded with NetworkX to extract the player state, including the current location, possessions, accessible areas, next areas, and relevant NPCs. Node descriptions provide additional grounding information for relevant locations and NPCs. An LLM-based State Verifier checks whether the retrieved NPC set conflicts with the current world state and minimally revises inconsistent NPC roles when necessary. Throughout the appendix, the prompt field "Possible NPCs" denotes the same state component referred to as "Relevant NPCs" in the main paper.

The Goal Generator and Action Planner are loaded either as full causal language models or as PEFT adapters. The Goal Generator returns one concise goal per generation call, and D-ARK constructs the candidate set by sampling the Goal Generator multiple times from the same verified state. Duplicate, invalid, or weakly grounded candidates are filtered before the remaining goals are presented to the player. After the player selects a goal, the Action Planner generates an action sequence from the selected goal, state, and retrieved world details. For goal generation, we use stochastic decoding with temperature 0.8 and top-p 0.9, with a maximum generation length of 128 tokens. For action planning, we use deterministic decoding with a maximum generation length of 1024 tokens. After quest completion, the Knowledge Updater extracts changes to player possessions, locations, and characters from the generated actions and applies them to the GraphML graph. These changes are stored as updated node attributes and entity-level descriptions, and they affect subsequent retrieval through the updated world version. The updated graph is saved as a new world version and reused in the next quest loop, allowing subsequent quests to reflect accumulated player-driven changes.

D-ARK w/o knowledge update. This setting follows the same configuration as D-ARK, except that the Knowledge Updater and State Verifier modules are excluded. Since the State Verifier resolves conflicts between the player state and the updated game world graph, it is also removed in this setting where knowledge updates are not applied.

The following prompts are used by the models in D-ARK.

Prompt for State Verifier

I want to design a quest in which the following NPCs appear. However, there may be parts that conflict with the absolute state. A conflict is strictly defined as a state that is impossible under the absolute state. For example, an NPC appearing despite being dead constitutes a conflict. Another example is an NPC appearing even though the player did not accompany them in a previous quest, which also constitutes a conflict. However, differences in location or other details are not considered conflicts, as they can be sufficiently explained through narrative context.
If there are no conflicts, output NONE. If there are conflicts, modify only the conflicting parts and keep the rest unchanged, then output the modified version in the specified format.
Possible NPCs: [PossibleNPC]
Absolute state: [npc descriptions]
== Examples ==
[examples]
The examples above are for explanation purposes only. In the actual output, choose one of the two approaches and output it in the following format.
Output version: 1, 2, or 3
Modified NPCs: JSON-formatted result
New character: {
before: [names exactly as in the input, ...]
after: [newly created names exactly as generated, ...]
}

Prompt for Goal Generator Instruction

You are given only the current quest state. Infer the intended quest goal that best follows from the player's location, possessions, accessible areas, NPC roles, area details, and NPC details.
Do not rely on quest names, prior game knowledge, or memorized canonical quest objectives. If possessions or NPC roles differ from a familiar quest, adapt the goal to the given state. Return exactly one concise sentence. Do not output JSON, bullet points, or explanations.

Prompt for Goal Generator Input

Below are the player's current location and the status of their possessed items.
Player Location: {example['start location']}
Player Possessions: {example['possessions']}
Below are the areas accessible in the current quest and the NPCs with their roles.
Accessible Areas: {example['accessible areas']}
Possible NPCs: {example['possible npcs']}

Prompt for Knowledge updater

Quest objective for this quest: [goal]
Actions the player performed during this quest: [actions with details]
The player has achieved the goal by performing the above actions during this quest. If there are any parts among the information below that should change due to the player's actions, add or modify them accordingly. Parts that can remain unchanged should be left as they are. Keys must be kept exactly the same, and only the values should be rewritten. You must never add arbitrary new keys.
Modification Rules:
1. Possessions: Only modify entries when there is a clear change, such as an item being repaired or upgraded, or add entries when a new item or new information is obtained through an action. There is no need to reflect how the item was used during this quest.
2. Locations: Only add or modify entries when the player's actions caused changes to the structure of a building or to the characters active within it. There is no need to reflect how the location was used during this quest.
3. Characters: Only modify entries when a character has died, when their relationship to gameplay has clearly changed due to the player's actions, or add entries when a character who is not listed in the Additional descriptions of characters appears in the Actions. There is no need to reflect the role the character played during this quest.
Player Possessions: [Possessions]
Additional descriptions of locations: [area descriptions]
Additional descriptions of characters: [npc descriptions]
== Examples ==
[examples]
Strictly Follow the Output Format:
Modified Possessions: { ... }
Modified Locations: { ... }
Modified Characters: { ... }

Prompt for Action Planner Instruction

You are given only the current quest state and the intended quest goal. Generate the action sequence a player should perform to achieve that goal.
Use the player's location, possessions, accessible areas, NPC roles, area details, and NPC details as constraints. Do not rely on quest names, prior game knowledge, or memorized canonical quest routes. If possessions, NPC roles, or area details differ from a familiar quest, adapt the action sequence to the given state and goal.
Return only the action sequence. Write one action per line. Each action should be a short imperative phrase such as "Talk to the guard." or "Enter the upstairs computer room." Do not output JSON, bullet markers, numbering, headings, or explanations.

Prompt for Action Planner Input

Below are the player's current location and the status of their possessed items.
Player Location: example['start location']
Player Possessions: example['possessions']
Below are the areas accessible in the current quest and the NPCs with their roles.
Accessible Areas: example['accessible areas']
Possible NPCs: example['possible npcs']
Additional area details:
example['area details']
Additional NPC details:
example['npc details']
Intended Goal:
example['intended goal']

A.2 Training Details

We fine-tune the Goal Generator and Action Planner from the two backbone LLMs used in our experiments: LLaMA-3.1-8B-Instruct and Qwen2.5-7B-Instruct. From the 207 original structured quest instances, we randomly select 100 instances as held-out test data and use the remaining 107 original instances as the non-augmented training set. For the augmented training setting, we use the same 107 original training instances together with 2,025 filtered augmented instances, resulting in 2,132 total training instances. The 100 held-out original instances are used only for evaluation. Both modules are trained with supervised fine-tuning using the model-specific chat template. For each training instance, the input prompt is constructed from the player state, and the target output is appended as the assistant response. During training, prompt tokens are masked out with label value -100, so the loss is computed only on the target answer tokens.

Goal Generator. The Goal Generator receives the player's current location, possessions, accessible areas, and possible NPCs as input, and is trained to generate the intended quest goal. We train the Goal Generator with LoRA using rank $r = 16$, dropout 0.05, and a maximum sequence length of 4096 tokens. For the augmented training set, we use learning rate 1×10^{-4}, per-device batch size 4, gradient accumulation 8, and train for 2 epochs. For the non-augmented variant used in ablation, we train for 3 epochs. Unless otherwise specified, LoRA α is set to 32.

Action Planner. The Action Planner receives the player state and the intended goal as input, and is trained to generate the intended action sequence. In the full setting, the input also includes area descriptions and NPC descriptions as retrieved world-detail context. For the ablation without retrieved

details, these descriptions are removed while all other settings are kept unchanged. We train the Action Planner with LoRA using rank $r = 16$, LoRA $\alpha = 32$, dropout 0.05, and a maximum sequence length of 4096 tokens. For the augmented training set, we use learning rate 7×10^{-5}, per-device batch size 2, gradient accumulation 16, and train for 5 epochs. For the non-augmented variant used in ablation, we train for 3 epochs.

Optimization and sample weighting. All models are trained in bfloat16 with a cosine learning-rate schedule and warmup ratio 0.1. Checkpoints are saved at the end of each epoch. We use a weighted token-level causal language modeling objective. Samples from Main Quests receive base weight 5.0, Side Quests receive base weight 2.0, and other quest types receive base weight 1.0. Original, non-augmented samples receive an additional multiplier of 3.0, while augmented samples use multiplier 1.0. This weighting encourages the model to preserve high-confidence canonical quest structures while still learning diverse state–goal–action mappings from augmented data.

A.3 Baseline Settings

A.3.1 Goal Generator

Zero-shot. For the zero-shot backbone baseline, we use the same input format and instruction prompt as the Goal Generator described in Appendix A.1, but without task-specific fine-tuning. To make the comparison stable and ensure that the backbone model follows the required output format, we provide five in-context examples sampled from the training set. Given the current player state, including the player location, possessions, accessible areas, and possible NPCs, the model is asked to generate a concise quest goal grounded in the provided state. The generated goal is then evaluated using the same plausibility, diversity, and autonomy criteria as the fine-tuned Goal Generator.

G2. G2 (Ruan et al., 2025) is a training-free decoding method designed to improve output diversity while preserving generation quality. The original method generates multiple responses by combining a base generator with two guide prompts: a Diversity Guide, which encourages outputs different from previous generations, and a Dedupe Guide, which models repetitive outputs so that they can be suppressed during decoding. At each decoding step, G2 modifies the base model logits using the contrastive signal between the Diversity Guide and the Dedupe Guide, and applies this intervention only when the model uncertainty exceeds an entropy threshold.

We adapt G2 as a reference baseline for diverse quest-goal generation. Since the original G2 is a general-purpose decoding strategy rather than a quest-generation model, we keep its training-free guided decoding mechanism but replace the task prompts with our quest-state goal-generation prompt. For each input state, the base generator receives the player's location, possessions, accessible areas, and possible NPCs, and predicts one concise quest goal. To generate multiple candidate goals, we apply G2 sequentially: the first goal is generated without guidance, and subsequent goals are generated using previous goal predictions as prior outputs. The Diversity Guide is prompted to generate a new state-grounded goal that does not simply rephrase previous predictions, while the Dedupe Guide is prompted to reproduce previous predictions as closely as possible. The resulting contrastive guidance encourages goals that remain valid under the same quest state but are less redundant with earlier candidates.

In our implementation, we generate five goals per input state using LLaMA-3.1-8B-Instruct as the backbone model. We apply G2 without additional fine-tuning and use center selection over prior goal predictions to condition the guide prompts when multiple previous outputs are available. Thus, G2 serves as a training-free diversity-oriented baseline for testing whether guided decoding alone can produce plausible and diverse candidate quest goals from the current player state.

Avoidance. Avoidance Decoding (Park et al., 2025) is a training-free decoding method originally proposed for diverse multi-branch story generation. The original method treats previously generated branches from the same prompt as negative samples and modifies the next-token score by penalizing similarity to those prior outputs. It combines two similarity penalties: a Concept-level Similarity Penalty, which compares candidate-token hidden states against negative samples, and a Narrative-level Similarity Penalty, which compares sentence-level embeddings against negative samples. The method gradually shifts emphasis from concept-level avoidance to narrative-level avoidance during decoding, encouraging later branches to follow different narrative trajectories while avoiding degeneration.

We adapt Avoidance Decoding as a reference

baseline for diverse quest-goal generation. Since the original method is designed for long-form story branches rather than one-sentence quest objectives, we keep its training-free avoidance mechanism but replace the story prompt with our quest-state goal-generation prompt. For each input state, the model receives the player's location, possessions, accessible areas, and possible NPCs, and generates five concise quest goals. The first goal is generated without negative samples, and subsequent goals are generated by treating earlier goal predictions for the same state as negative samples. During decoding, candidate tokens that make the current goal prediction more similar to previous goals are penalized, encouraging the model to produce less redundant alternatives.

In our implementation, we use LLaMA-3.1-8B-Instruct as the backbone model and apply Avoidance Decoding without task-specific fine-tuning. We retain the core hyperparameters from the original method where applicable, including the concept-to-narrative interpolation, while shortening the maximum generation length for concise goal prediction. Because our task requires state-grounded candidate goals rather than open-ended story continuations, Avoidance serves as a training-free diversity-oriented baseline for testing whether similarity-penalized decoding alone can produce plausible and diverse quest goals from the same player state.

A.3.2 Action Planner

Zero-shot. For the zero-shot backbone baseline, we use the same input format and instruction prompt as the Action Planner described in Appendix A.1, but without task-specific fine-tuning. We include five in-context examples sampled from the training set so that the backbone model follows the desired action-sequence format. Given the current player state and the intended quest goal, the model is asked to produce an ordered action sequence, with one action per line. For the setting with retrieved world details, the prompt also includes area descriptions and NPC descriptions; for the setting without details, these fields are omitted. The generated action sequence is evaluated using the same similarity and diversity criteria as the fine-tuned Action Planner.

GRASE-DC. GRASE-DC (Zhao et al., 2025) is an in-context learning method originally proposed to improve LLM planning by selecting exemplars based on action sequence similarity. The original method observes that planning problems with similar task descriptions can still require substantially different plans, and therefore uses the similarity between action sequences as a more direct signal for exemplar selection. It first generates an initial plan with randomly selected exemplars, computes the action sequence similarity between this initial plan and candidate exemplar plans, and then re-samples exemplars with high action sequence similarity. It further applies Dynamic Clustering to balance relevance and diversity among the selected exemplars before prompting the model again for the final plan.

We adapt GRASE-DC as a reference baseline for quest action planning. Since the original method is designed for general planning tasks with executable plans, we retain its exemplar-selection mechanism but replace the task representation with our quest-state action-planning format. For each target quest state, the model receives the player's location, possessions, accessible areas, possible NPCs, retrieved world details, and intended goal, and is asked to generate a one-action-per-line quest progression. We first prompt the model with randomly sampled training exemplars to obtain an initial action sequence. We then compute action sequence similarity between this generated sequence and the action sequences of training exemplars, using the normalized longest common action sequence measure. The high-similarity exemplars are further curated with Dynamic Clustering to reduce redundancy while preserving diverse but relevant demonstrations.

In our implementation, we use LLaMA-3.1-8B-Instruct as the backbone model and apply GRASE-DC without task-specific fine-tuning. The selected exemplars are used only as in-context demonstrations for the Action Planner prompt; the gold action sequence of the target instance is never provided as input. Thus, GRASE-DC serves as a training-free planning baseline for testing whether action-sequence-based exemplar selection can improve quest action generation without updating model parameters.

LLM-REGRESS. LLM-REGRESS (Liu et al., 2025) is an open-world planning method that combines lifted regression with LLM-inferred affordances for embodied agents. The original method first parses a natural-language goal into symbolic goal atoms, performs backward lifted regression over STRIPS-like action schemas, and then uses an LLM-based affordance reasoner to bind lifted variables to objects observed in the environment.

This allows the planner to search over compact lifted plans without exhaustively enumerating all grounded states and actions.

We adapt LLM-REGRESS as a reference baseline for quest action planning. Since our setting does not provide a simulator, executable action validator, or predefined game-domain PDDL model, we implement a domain-generic lifted-regression pipeline for quest traces. For each target quest state, the model receives the player's location, possessions, accessible areas, possible NPCs, retrieved world details, and intended goal. An LLM-based goal parser first converts the intended goal into a small set of symbolic goal atoms. A lifted regression planner then searches backward over generic quest action schemas, such as traveling to an area, meeting or talking to an NPC, acquiring an item, inspecting a location, hacking a system, neutralizing an enemy, rescuing an NPC, or completing a goal. Next, an LLM-based affordance-binding step grounds open variables in the lifted plan using entities available in the current quest state.

In our implementation, we use LLaMA-3.1-8B-Instruct as the backbone model and apply LLM-REGRESS without task-specific fine-tuning. Because the original LLM-REGRESS assumes an embodied environment with executable actions and feedback, we replace environment execution with a final constrained LLM linearization step that converts the grounded symbolic skeleton into a one-action-per-line quest progression. The gold action sequence of the target instance is never provided as input. Thus, LLM-REGRESS serves as a training-free symbolic-planning-oriented baseline for testing whether lifted regression and LLM-based affordance binding can produce coherent quest action sequences from the current player state.

A.4 Evaluation Metrics and Prompts

A.4.1 Player Agency

Player agency refers to the extent to which players perceive that their choices and actions bring about meaningful changes within the game world. That is, the outcomes of previous quests must be consistently reflected in subsequent quests without narrative conflicts. Detailed player-agency evaluation prompts are shown in Figures 5, 6, and 7.

A.4.2 Player Autonomy

Player autonomy refers to the extent to which players perceive that they are provided with meaningful freedom in choosing which goals to pursue within the game. In this work, we evaluate autonomy at two levels. At the framework level, we assess whether a sequence of generated quests avoids forcing the player toward a single dominant progression path and instead preserves meaningful freedom across successive quest choices. At the goal-generation level, we assess whether the Goal Generator produces a diverse set of candidate goals that allows players to choose objectives aligned with their preferred play style and progression direction. Higher player autonomy is characterized by a broader and more diverse goal space that enables players to meaningfully shape their own progression paths. The detailed framework-level autonomy prompts are provided in Figures 8, 9, and 10, and the goal-level autonomy prompts are provided in Figures 11 and 12.

A.4.3 Coherence

Coherence refers to whether a generated quest chain forms a logically understandable and internally consistent progression around the player's selected objectives. In our framework-level setting, when the player completes quest i, the model presents quest $i + 1$. Therefore, coherence is evaluated not only at the level of individual quests, but also in terms of whether each selected objective, its generated action sequence, and the following quest fit together as a continuous chain.

Importantly, coherence does not require the generated chain to follow the canonical storyline or the most conventional RPG progression. Player-selected objectives may be unexpected, aggressive, destructive, or divergent, and such objectives should not be penalized merely because they deviate from the original narrative. Instead, coherence focuses on whether the generated actions are appropriate for the selected objective, whether they make meaningful progress toward fulfilling or responding to that objective, and whether the next quest follows naturally from the previous objective and action sequence.

A highly coherent quest chain should maintain compatibility among NPCs, locations, items, and events across the sequence, avoid severe logical contradictions or unexplained resets, and feel like a connected progression built from player choices rather than isolated tasks. Conversely, coherence is low when actions fail to support the selected objective, when subsequent quests ignore or contradict the direction established by previous choices, or when quest transitions feel arbitrary. The detailed

coherence evaluation criteria are provided in Figures 13, 14, and 15.

A.4.4 Plausibility

To measure the plausibility of goals generated by the Goal Generator, we adopt the concept of Dialogue Relevance (Ashby et al., 2023). Dialogue Relevance evaluates how relevant generated quest dialogue is to the given quest information. Similarly, since a goal provides the overall flow and direction of a quest, it should be generated by sufficiently leveraging the given state information. In particular, the more actively a goal reflects information contained in the state, the more likely it is to be naturally derived from the given situation.

We first compute an automatic plausibility score based on lexical grounding. For N goals generated from a given state, we define automatic plausibility as follows:

$$\text{Plausibility} = \frac{1}{N}\sum_{i=1}^{N}\frac{|T(g_i) \cap T(s)|}{|T(g_i)|}$$

where $T(g_i)$ and $T(s)$ denote the term sets extracted from the i-th goal and the state, respectively, using spaCy. This metric quantitatively evaluates how strongly a goal is grounded in the given state information by measuring the proportion of terms in the goal that are shared with the state.

Because lexical overlap alone may not fully capture whether a goal is contextually reasonable, we also conduct a GPT-based plausibility evaluation. The GPT evaluator is given the same state information and generated goals, and judges whether each goal can be naturally derived from the provided player location, possessions, accessible areas, and possible NPCs without relying on unsupported assumptions. The detailed plausibility evaluation prompts are provided in Figures 16 and 17.

A.4.5 Diversity

We evaluate diversity at two levels: the diversity of candidate goals generated by the Goal Generator and the diversity of action sequences generated by the Action Planner. Goal diversity measures whether the generated goals provide distinct narrative directions rather than minor paraphrases of the same objective. For this, we employ the Diversity to Similarity Reduction (Tevet and Berant, 2021).

Action diversity measures whether the generated action sequence involves a broad range of distinct game-world interactions. For this, we propose Unique-Action Count, a metric adapted from the Unique-Sentence Ratio (Li et al., 2020; Yang et al., 2025b). USR measures the ratio of unique sentences among generated outputs, but it has two limitations for quest action evaluation. First, USR treats "open the door" and "open the door slowly" as different sentences. However, from the perspective of game quest actions, these two sentences involve the same interaction with the same object, and thus should not be considered truly unique actions. Second, the number of unique actions directly reflects how many diverse elements in the game world the player interacts with. Therefore, instead of measuring a ratio, it is more appropriate to measure the number of unique actions. To address this, we define UAC as follows:

$$\text{UAC} = \left|\bigcup_{a \in A} F(a)\right|$$

where $F(a)$ extracts the verb–object pair from an action a using spaCy, so that actions involving the same interaction with the same object are treated as identical.

In addition to these automatic metrics, we conduct GPT-based diversity evaluations for both goals and actions. For goal diversity, the evaluator judges whether the candidate goals lead to meaningfully different narrative directions. For action diversity, the evaluator judges whether the action sequence covers varied interactions with locations, NPCs, objects, and events rather than repeating similar steps. The detailed GPT-based diversity prompts are provided in Figures 18 and 19 for goal diversity, and Figures 20, 21, and 22 for action diversity.

A.4.6 Similarity

To measure the similarity between the action sequence generated by the Action Planner and the ground-truth action sequence, we employ the Ordering Score (Chaudhuri et al., 2025). Since an action sequence represents the series of actions that a player must actually perform during gameplay, a higher similarity to the ground-truth sequence indicates the generation of a higher-quality quest. The Ordering Score was originally proposed to evaluate how similar a model-generated travel itinerary is to a real travel plan, as it considers not only the constituent actions but also their ordering. This characteristic aligns well with the evaluation of quest action sequences, where the order of actions is crucial for proper progression. Therefore, we use the Ordering Score to quantitatively assess the

similarity between generated action sequences and their corresponding ground-truth sequences.

Because automatic ordering-based similarity may not fully capture semantic equivalence between differently worded but functionally similar actions, we also conduct a GPT-based similarity evaluation. The evaluator compares the generated and ground-truth action sequences and judges whether the generated sequence follows a similar progression toward the same quest goal while preserving important actions and their ordering. The detailed similarity prompt is provided in Figure 23.

B Knowledge Update Verification

B.1 QA-based Verification Pipeline

To verify whether world knowledge updates are reflected in the knowledge graph, we use a QA-based check with binary yes/no questions. Given the raw Knowledge Updater output, which contains modified and unchanged entries for player possessions, locations, and characters, GPT-5-mini generates question–answer pairs targeting specific world-state facts, such as repaired items, destroyed locations, or changed NPC states. We then query the updated graph with these questions and compare its answers with the generated references.

B.2 QA Construction and Statistics

For each knowledge update, we generate 10 binary question–answer pairs. Each answer is derived strictly from the Knowledge Updater output, without additional assumptions or external game knowledge. The prompt used for QA construction is shown below.

Prompt for QA Generation

Data: {modified fact raw}
Objective: Generate a Q&A dataset that can be used to verify whether the information contained in the given Data has been correctly updated in the Knowledge Graph.
Requirements:
- Generate 5–10 question–answer pairs.
- Each Question must be verifiable within the Knowledge Graph and must be answerable with either "yes" or "no" (e.g., entities, relations, attributes, temporal information, numerical values, etc.).
- Each Answer must be derived strictly from the given Data, with no speculation, and must be either "yes" or "no".
- The number of Questions and Answers must be exactly the same.
Output format (JSON ONLY, no additional text):
{ "questions": ["..."], "answers": ["..."] }

Playtime	W1	W2	W3	W4	W5	Avg
Play1	0.88	0.88	1.00	1.00	1.00	0.95
Play2	1.00	0.88	1.00	0.86	1.00	0.95
Play3	0.88	1.00	1.00	0.86	0.88	0.92
Play4	1.00	0.90	1.00	0.89	0.50	0.86
Play5	1.00	0.88	1.00	0.90	1.00	0.96

Table 7: QA results validating world knowledge updates.

We conduct this verification over sequentially updated worlds. For each playthrough, we evaluate five world versions generated across consecutive quest loops, and repeat this process across five playthroughs. Thus, the QA check evaluates whether D-ARK preserves accumulated world-state changes not only within a single update step but also across sequential quest generation.

B.3 Knowledge Update Accuracy

Table 7 reports QA accuracy for the updated world knowledge across playthroughs and world versions. The results show consistently high accuracy across different playthroughs, indicating that the updated knowledge graph generally reflects the modified facts produced after quest completion. The accuracy also remains stable across later world versions, suggesting that accumulated updates are preserved as gameplay progresses.

These results provide additional evidence that D-ARK's world knowledge update mechanism can maintain dynamic world-state information across sequential quest generation. At the same time, this verification should be interpreted as a targeted consistency check rather than a full guarantee of executable game-state validity, since the QA pairs focus on explicit facts extracted from the generated quest outcomes.

C Dataset Construction and Augmentation

C.1 Wiki Collection and Structured Quest Schema

We construct the quest dataset from Cyberpunk 2077 Fandom Wiki pages. For each quest, we collect the quest description, walkthrough text, progression information, related quest links, and entity-level descriptions when available. We then convert the collected information into a structured state–goal–action schema. Figure 24 shows the prompt used for this conversion. Given the raw walkthrough and action sequence, the prompt extracts the player's start location, possessions, rel-

evant NPCs, accessible areas, area details, NPC details, the intended quest goal, and the corresponding action sequence.

Each structured instance contains the following fields: start location, possessions, relevant NPCs, accessible areas, area details, NPC details, intended goal, and intended actions. The start location, possessions, accessible areas, and relevant NPCs define the player state used by the Goal Generator. The area details and NPC details provide additional grounding information for the Action Planner. The intended goal and intended actions define the target quest objective and its corresponding progression. This schema directly follows the formulation in Section 3, where each quest is represented as a state-conditioned goal and action sequence.

C.2 World Knowledge Graph Construction

We use the structured quest fields to construct the initial world knowledge graph. Entities such as locations, NPCs, possessions, quest-relevant objects, events, and quest states are represented as graph nodes or node attributes, depending on whether they function as retrievable entities or descriptive facts. State relations such as current location, current possession, accessible areas, relevant NPCs, and next areas are represented as graph edges. Location and NPC descriptions are stored as node attributes and later used as grounding context for retrieval and action planning.

Quest dependencies and event information are represented either as explicit quest/event nodes or as attributes attached to quest-relevant locations and NPCs, depending on whether they affect state transitions across quests. These dependency links are used to construct connected quest progressions, especially for Main Quests where inter-quest dependencies and persistent state changes can be traced over time. This graph construction allows D-ARK to retrieve the current player–world state, ground goal and action generation in the retrieved world details, and update the graph after quest completion.

C.3 Consistency-Preserving Dataset Augmentation

The original wiki data primarily describe canonical quest progressions. Although these data provide high-quality quest structures, they offer limited coverage of alternative goals and divergent player-driven states. To expand the range of possible state–goal–action mappings, we augment the dataset with two complementary strategies, following the order introduced in Section 5.2: goal-change augmentation and state-change augmentation.

The purpose of augmentation is not to introduce arbitrary new world entities. Instead, augmentation expands the mapping between states, goals, and actions within a fixed quest setting. We therefore constrain augmented instances to reuse existing locations, NPCs, and possessions, while allowing the goal and action progression to diverge from the canonical storyline. Both strategies are designed to preserve consistency between the state, goal, and action sequence.

Goal-change augmentation. In the first strategy, we keep the original player state fixed and generate alternative goals that move the quest narrative in different directions. Given the original location, possessions, accessible areas, relevant NPCs, and intended goal, the diverse-goal prompt generates multiple alternative goals, as shown in Figure 31. These goals must be grounded in the fixed state but should not merely be safer routes, preparation steps, detours, or optimizations for achieving the original goal. Instead, they may involve refusing, abandoning, delaying, sabotaging, reversing, exposing, or replacing the original objective. This strategy expands the player's choice space from the same state, which directly supports the autonomy-oriented goal-generation setting in the main experiments.

For each alternative goal, we use the action-modification prompt in Figures 28, 29, and 30 to revise the original action sequence under the fixed state and new goal. The revised sequence preserves valid original actions when possible, but removes or replaces actions that conflict with the new goal, unavailable possessions, inaccessible areas, or NPC roles. This produces diverse narrative continuations while maintaining consistency with the given player state.

State-change augmentation. In the second strategy, we perturb the player state and then revise the goal and action sequence so that they remain valid under the modified state. Given the original player location, possessions, accessible areas, relevant NPCs, and intended goal, the state-variation prompt generates up to five minimally modified states, as shown in Figures 25 and 26. The modifications are restricted to the condition or usability of possessed items and the role, function, condition, or status of existing NPCs. The player location, accessible areas, and NPC names are kept fixed,

and no new areas, NPCs, or possessed items are introduced. This keeps the original quest setting recognizable while creating situations in which the original goal or action progression may no longer be valid.

For each generated state variation, we predict a state-valid revised goal using the goal-revision prompt shown in Figure 27. The revised goal is required to remain close to the original quest intent when possible, but it must be valid under the modified state. Finally, given the modified state and revised goal, we use the action-modification prompt in Figures 28, 29, and 30 to revise the original action sequence. This produces augmented instances in which the state, goal, and action sequence are updated consistently.

C.4 Consistency Verification, Filtering, and Dataset Statistics

During augmentation, we enforce format and consistency constraints at each stage. Goal-change outputs must be valid JSON arrays of concrete, actionable goals grounded in the provided state. State-change outputs must preserve the same keys as the original schema and must not alter the player location, accessible areas, or NPC names. For action revision, the prompt requires one action per line and prohibits introducing locations, NPCs, or possessions that are not present in the current state.

We also apply semantic consistency checks before adding an augmented instance to the training set. A revised goal must be grounded in the corresponding state, and a revised action sequence must not require unavailable possessions, unlisted NPCs, or inaccessible areas. The revised action sequence must also make progress toward the revised goal rather than merely restating the original canonical path. We discard outputs that fail JSON parsing, violate the required schema, cannot be converted into the structured quest format, or violate these semantic consistency constraints.

For each of the 207 original structured quest instances, goal-change augmentation generates five alternative goals, producing $207 \times 5 = 1{,}035$ candidate instances. State-change augmentation likewise generates five modified states per original instance and revises the goal and action sequence for each modified state, producing another $207 \times 5 = 1{,}035$ candidate instances. Together, the two augmentation strategies therefore produce 2,070 candidate augmented instances before filtering. After format validation and semantic consistency checking, 2,025 augmented instances remain.

For data splitting, we randomly select 100 instances from the 207 original instances as held-out test data. The original-only training setting uses the remaining 107 original instances. The augmented training setting adds the 2,025 filtered augmented instances to these 107 original training instances, resulting in 2,132 total training instances. All evaluations are conducted on the 100 held-out original instances.

=== Quest Chain ===
{quest_chain}

=== GT Quest Chain ===
{gt_quest_chain}

=== Guidelines ===
The given quest chain operates such that when the player completes quest i, the model presents quest i+1.

In this evaluation, Player Agency refers to how strongly the given quest chain appears to be driven by the player's own choices when compared with the original quest chain shown in GT Quest Chain.
The GT Quest Chain represents the default or original quest progression.
Your task is to judge whether the given quest chain merely follows the GT path, or whether it shows that the player actively led the quest chain into a player-directed route.

The only evidence for this evaluation is:
- the visible objectives and action sequences in the given quest chain
- the visible objectives and action sequences in the GT quest chain

Do not rely on hidden state traces, planner predicates, world-state metadata, or external context.

A high-agency quest chain is not simply one that is different from GT.
It should show that differences from GT are caused by the player's chosen objectives and actions, and that the chain develops around those choices.
The player should appear to be actively steering the direction of the quest chain, not passively following the original quest structure with small wording or action variations.

Figure 5: Framework Evaluation Prompt: Agency [1/3]

A central part of Player Agency is whether the world and later quests visibly respect what the player chose or did earlier.
If the player pursues a specific objective or performs a major action in an earlier quest, later quests should reflect that choice through changed objectives, changed available actions, changed character presence, changed relationships, changed conflicts, or changed outcomes.
When an earlier player choice is ignored, overwritten, or contradicted by a later quest, this is a major penalty for Player Agency.
For example, if the player kills a character in Quest 1 but that same character appears normally in Quest 2 without explanation, the player would feel that their action did not matter and that they are merely being pushed along a predetermined route.
When comparing the given quest chain with the GT quest chain, consider:
- Whether the player's objectives introduce a direction, intention, conflict, alliance, method, or outcome that is meaningfully different from GT
- Whether the generated actions support that player-directed route rather than simply recreating the GT steps
- Whether later quests continue from the player's earlier choices instead of returning to the GT path
- Whether earlier player actions create visible downstream effects in later quests
- Whether major actions such as killing, saving, betraying, helping, allying with, stealing from, or opposing someone are reflected in later objectives, actions, characters, conflicts, or outcomes
- Whether the quest chain avoids treating earlier player choices as if they never happened
- Whether the player appears to initiate goals, solve problems in their own way, change priorities, or reshape the route
- Whether deviations from GT are meaningful at the quest-chain level, not merely local wording changes or minor action substitutions
- Whether the chain remains understandable enough to identify a player-led route
Important distinction:
- Do not give a high score merely because the chain is incoherent or random compared with GT.
- Do not give a high score merely because the chain is different from GT.
- Do not give a high score for superficial differences such as renamed objectives, reordered actions, or small action changes that preserve the same GT progression.
- Do not give a high score when early player choices are ignored, overwritten, or contradicted by later quests.
- Treat unacknowledged contradictions of major player actions as a severe Player Agency problem.
- For example, if the player kills, removes, betrays, or fundamentally changes a character, faction, object, or location, but later quests proceed as if nothing happened, the score should be heavily penalized.
- Do give a high score when the player-selected objectives and resulting actions clearly redirect the quest chain away from GT in a sustained and purposeful way.
- Give the highest scores only when later quests also preserve and build on the consequences of earlier player choices.
- For Player Agency, focus on how much the player appears to lead the given quest chain compared with the original GT quest chain, and whether the world continues to respect that leadership across later quests.

Figure 6: Framework Evaluation Prompt: Agency [2/3]

=== Evaluation Criteria ===
5:The given quest chain is strongly player-led compared with GT. The player's objectives and actions create a sustained route that is meaningfully different from the original quest chain. Important player actions in earlier quests are clearly reflected in later quests.
Later objectives, actions, characters, conflicts, relationships, or outcomes visibly build on what the player previously chose or did.
The player appears to actively drive goals, methods, conflicts, or outcomes across the entire chain, and the world consistently respects those choices.
4:The given quest chain is mostly player-led compared with GT.
There are meaningful deviations in objectives, methods, conflicts, or outcomes, and most later quests continue from those player-directed choices.
Earlier player actions are usually acknowledged in later quests, though some parts may still resemble the GT structure or briefly return to GT-like progression.
The overall route is clearly shaped by the player, and the world mostly respects the consequences of the player's earlier choices.
3:The given quest chain shows moderate player agency compared with GT.
Some objectives or actions meaningfully differ from GT and suggest player initiative, but the chain only partially develops around those choices.
Some later quests reflect earlier player choices, but several quests still follow the GT path, or deviations remain local rather than reshaping the full quest chain.
Important player actions may have limited or inconsistent downstream effects.
2:The given quest chain shows low player agency compared with GT.
Most objectives and actions closely follow the original quest chain, with only minor or superficial deviations.
Player choices rarely redirect later quests, and the chain mostly feels like the GT path with small variations.
Earlier player actions are often ignored, weakly acknowledged, or contradicted by later quests.
1:The given quest chain shows almost no player agency compared with GT.
It essentially follows the original quest chain, or any differences are random, incoherent, or too superficial to indicate player-driven progression.
Earlier player choices have no visible effect on later quests, or later quests directly contradict major player actions.
The player does not appear to lead the quest chain in a meaningful way because the world proceeds as if the player's choices did not happen.
=== Output Rules ===
The final answer must provide
only a single numeric score (1-5).
Format:
Score: <1-5>

Figure 7: Framework Evaluation Prompt: Agency [3/3]

=== Multiple Generated Quest Chains from the Same Initial State ===
{multiple_quest_chains}

=== Guidelines ===
The generated quest chains are multiple play trajectories produced from the same initial state.

Your task is to evaluate Player Autonomy by determining how many of the generated chains provide meaningfully different narrative developments across repeated playthroughs.

The only evidence for this evaluation is:
- Quest objectives
- Action sequences

Do NOT use hidden planner states, final-state predicates, state traces, metadata, or external knowledge.
Do NOT penalize a chain merely because its actions are written in a symbolic, abbreviated, or planner-like format.
Evaluate only the narrative meaning of the objectives and actions.

Core idea:
Player Autonomy is high when repeated playthroughs from the same starting point lead to genuinely different story developments.
A player should feel that playing the game multiple times can produce different narrative trajectories, not merely minor variations of the same path.

For each generated chain, judge whether it provides a DISTINCT STORY PATH compared with the other generated chains.

A generated chain counts as a distinct story path only if it meaningfully changes the perceived story development compared with the other chains.

Figure 8: Framework Evaluation Prompt: Autonomy [1/3]

Meaningful differences may include:
- Different quest objectives or narrative direction
- Different conflicts or problems
- Different alliances, betrayals, or relationships
- Different use of NPCs, factions, locations, or items
- Different player intentions or playstyles
- Different outcomes or consequences implied by the objective/action sequence
- Different routes such as cooperation, betrayal, combat, stealth, negotiation, sabotage, rescue, escape, alliance-building, destruction, investigation, or independent route-taking

Do NOT count a chain as distinct if:
- It is only a wording-level variation of another chain
- It follows the same dominant story pattern with minor local action changes
- It changes action order without changing the narrative development
- It uses different surface details but leads to essentially the same story
- It is incoherent, random, or too ambiguous to interpret as a meaningful path

Important:
- Autonomy is not about whether each individual chain has strong consequence handling. That is Player Agency.
- Autonomy is not about whether each individual chain is logically smooth. That is Coherence.
- Autonomy is about whether the set of generated chains offers multiple meaningfully different narrative experiences across repeated playthroughs.
- A single highly unusual chain is not enough for high autonomy if the other chains are similar.
- Be strict. Count only clearly distinct and meaningful story trajectories.
- If evidence is ambiguous, prefer the lower score.

Figure 9: Framework Evaluation Prompt: Autonomy [2/3]

=== Evaluation Criteria ===

5: All 5 generated chains count as distinct story paths.
Across five playthroughs, the player would experience five clearly different narrative developments.
There is no dominant repeated storyline.

4: Exactly 4 generated chains count as distinct story paths.
Four chains provide meaningfully different narrative developments.
One chain substantially overlaps with another chain, follows the dominant pattern, or differs only superficially.

3: Exactly 3 generated chains count as distinct story paths.
Three chains provide meaningfully different narrative developments.
The remaining chains are redundant, only locally different, or too similar to another chain.

2: Exactly 2 generated chains count as distinct story paths.
Only two chains provide meaningfully different narrative developments.
Most generated chains repeat the same story pattern or differ only superficially.

1: Zero or one generated chain counts as a distinct story path.
Across repeated playthroughs, the player would experience essentially the same story or minor variations of one dominant trajectory.

=== Output Rules ===
The final answer must provide only a single numeric score (1–5).

Format:
Score: <1–5>

Figure 10: Framework Evaluation Prompt: Autonomy [3/3]

You are an expert evaluator of game design and player autonomy.

The game presents the player with the following multiple candidate goals
from the same current game state.

Your task is to evaluate how much meaningful freedom of choice the goal set
provides to the player.

Player autonomy means that the player can choose which goal to pursue without
being implicitly forced toward a single dominant or clearly “correct” objective.
A high-autonomy goal set should allow the player to meaningfully shape their
own progression path through distinct objectives, strategies, conflicts,
alliances, risks, or playstyles.

When evaluating, consider:
- Whether the goals offer meaningfully different progression directions
- Whether the alternatives feel equally legitimate and contextually valid
- Whether the goal set avoids steering the player toward one dominant objective
- Whether choosing different goals would plausibly lead to different player experiences
- Whether the goals are not merely superficial variations of the same underlying task

Do NOT evaluate:
- Whether previous quest outcomes persist over time; that is player agency
- Whether the goals are narratively smooth as a sequence; these are candidate alternatives
- Whether the system handles explicit rejection of goals; focus on the freedom provided by the goal set itself

Figure 11: Goal Generator Evaluation Prompt: Autonomy [1/2]

Rate the overall level of player autonomy provided by the goal set using a 5-point Likert scale:

1 = Very low autonomy
(The goals are repetitive or effectively force a single progression path.)
2 = Low autonomy
(Alternatives exist, but most feel secondary, redundant, or clearly less valid.)
3 = Moderate autonomy
(Multiple goals are available, but some overlap substantially or one direction still feels dominant.)
4 = High autonomy
(The goals provide several legitimate and meaningfully distinct choices.)
5 = Very high autonomy
(The goals provide a broad set of equally valid, clearly distinct choices that enable self-directed progression.)

Goals to evaluate:
Goal 1: [goal1]
Goal 2: [goal2]
Goal 3: [goal3]
Goal 4: [goal4]
Goal 5: [goal5]

Provide only a single numerical score (1-5) as your final answer.

Format:
Score: <1-5>

Figure 12: Goal Generator Evaluation Prompt: Autonomy [2/2]

=== Generated Quest Chain ===
{quest_chain}

=== Guidelines ===
The given quest chain operates such that when the player completes quest i, the model presents quest i+1.

In this evaluation, each quest objective represents the choice or intention selected by the player at that point in the playthrough.
Player choices may be unexpected, unconventional, aggressive, destructive, or divergent.
Such objectives should not be penalized merely because they differ from the canonical storyline, the GT quest chain, or a typical RPG progression.

In framework-level game quest evaluation, Coherence refers to whether the model builds the quest chain around the player's selected objectives in a logically understandable and internally consistent way.
A coherent quest chain should make each player-selected objective, the generated actions for that quest, and the next presented quest fit together as a continuous chain.
The main question is:
Given the player's chosen objective, are the actions for that quest well generated, and do those actions help complete or advance the overall quest chain?

The only evidence for this evaluation is the visible quest objectives and action sequences.
Do not rely on hidden state traces, planner predicates, world-state metadata, or external context.

Coherence does NOT require the quest chain to follow the original canonical storyline or the most conventional RPG progression.
A quest chain may be divergent, aggressive, unusual, or player-driven and still be coherent if:
- the actions within each quest are appropriate for the objective chosen by the player,
- the generated actions make progress toward fulfilling, resolving, or meaningfully responding to that objective,
- the consequences implied by those actions are not contradicted later,
- and the next quest follows naturally from the previous objective and action sequence so that the whole chain feels completed rather than fragmented.

Figure 13: Framework Evaluation Prompt: Coherence [1/3]

When evaluating coherence, consider:
- Whether each objective can be read as the player's current choice in the context established by previous quests
- Whether the actions within each quest are well aligned with that player choice
- Whether the actions are sufficient, relevant, and ordered in a way that makes the objective feel actively pursued
- Whether the transition from Quest i to Quest i+1 is explainable from the previous player choice and generated actions
- Whether the next quest acknowledges the narrative direction created by the player's prior objective, even if that direction is unexpected
- Whether NPCs, locations, items, and events used in the objectives/actions remain compatible across the chain
- Whether the chain avoids severe logical contradictions, impossible character states, unexplained reversals, or arbitrary jumps
- Whether the sequence feels like a complete connected quest chain built from player choices rather than isolated tasks

Important distinction:
- Do not penalize a quest chain merely because the player objective is unexpected, hostile, destructive, or divergent.
- Do not penalize a quest chain merely because it diverges from the original game storyline.
- Do penalize a quest chain if the generated actions do not make sense for the player-selected objective, are too generic to support it, or fail to move that objective forward.
- Do penalize a quest chain if the next quest ignores, contradicts, or arbitrarily resets the direction implied by the previous player choice/actions.
- Do not reward a quest chain merely because its action order appears mechanically valid; focus on whether the objective, actions, and next quest form an understandable narrative progression.
- Do not evaluate diversity of paths; that is Player Autonomy.
- Do not evaluate whether the generated chain gives the player more initiative than the GT quest chain; that is Player Agency.
- For Coherence, focus only on player-objective alignment, action quality for that objective, internal consistency, and understandable quest-to-quest completion.

Figure 14: Framework Evaluation Prompt: Coherence [2/3]

=== Evaluation Criteria ===
5:The quest chain is highly coherent.
Player-selected objectives, generated actions, and subsequent quests are consistently understandable and tightly connected across the entire sequence. For nearly every quest, the actions clearly support the player's chosen objective and make the chain feel progressively completed.
Even when objectives are unexpected or divergent, the actions clearly pursue them and the following quests naturally continue from them.
Characters, locations, items, and events remain internally consistent, and the chain has no serious logical gaps, contradictions, or arbitrary resets.
4:The quest chain is generally coherent. Most player-selected objectives are supported by appropriate actions, and most subsequent quests follow from the previous player choice/action context in an understandable way.
The generated actions usually make the selected objectives feel actionable and help complete the overall chain.
Established characters, locations, items, and events are mostly consistent.
There may be minor gaps, underexplained transitions, or slightly weak follow-ups, but no major contradictions or severe objective-action mismatches.
3:The quest chain is moderately coherent. The sequence is understandable overall, but some player-selected objectives are only loosely supported by the actions, or some next quests follow from the previous quest only weakly.
There may be minor inconsistencies, unclear objective shifts, abrupt transitions, overly generic actions, or unnecessary action detours.
The quests are connected enough to follow, but the objective-action-next quest progression is not strongly integrated.
2:The quest chain has low coherence. Several quests feel abrupt, weakly connected, or difficult to justify from the previous objective/actions.
Actions often do not clearly support the player-selected objectives, fail to make meaningful progress toward them, or subsequent quests frequently ignore or weakly respond to the direction established by prior quests. There are noticeable inconsistencies, unsupported objective changes, confusing uses of characters/locations/items, unexplained jumps, or partial resets.
The sequence is partially understandable but often feels disjointed.
1:The quest chain is not coherent. Player-selected objectives, generated actions, and subsequent quests largely fail to form a continuous understandable progression. Actions frequently conflict with or fail to support the stated objectives, and quest transitions are unclear, contradictory, or arbitrary. Quests are mostly isolated, internally inconsistent, or impossible to understand as a connected quest chain.

=== Output Rules ===
The final answer must provide only a single numeric score (1-5).
Format:
Score: <1-5>

Figure 15: Framework Evaluation Prompt: Coherence [3/3]

You are an expert evaluator of goal plausibility in game narratives.

Your task is to assess how plausible the following N goals are **given the provided game state**.

Plausibility is not just about consistency with the state, but about **how centrally and meaningfully the goals utilize the entities explicitly present in the state** (characters, locations, items, relationships).

Consider whether the goals:
- Actively use key entities (NPCs, objects, places) from the state as core narrative drivers
- Could realistically be derived only from the given state, rather than being generic or reusable goals
- Build conflict, twists, or developments grounded in the state's specific entities and relationships

Note: Goals that subvert or invert the state (e.g., betrayal, destruction, reversal of roles) are considered **highly plausible** if they strongly leverage state entities.

Rate the overall plausibility of the goals using a 5-point Likert scale:

1 = Very low plausibility
(Goals introduce unrelated entities, locations, or events with no grounding in the state.)
2 = Low plausibility
(Goals loosely mention state elements but remain generic or context-independent.)
3 = Moderate plausibility
(Goals use some state entities, but the connection is weak or partial.)
4 = High plausibility
(Goals actively and meaningfully use major state entities and their context.)
5 = Very high plausibility
(All aspects of the goals are deeply rooted in specific details of the state; the goals could not exist without this state.)

Figure 16: Goal Generator Evaluation Prompt: Plausibility [1/2]

State:
[state]

Goals to evaluate:
Goal 1: [goal1]
Goal 2: [goal2]
Goal 3: [goal3]
Goal 4: [goal4]
Goal 5: [goal5]

Provide only a single numerical score (1-5) as your final answer.

Format:
Score: <1-5>

Figure 17: Goal Generator Evaluation Prompt: Plausibility [2/2]

```
You are an expert evaluator of narrative goal diversity in game quests.

Your task is to assess how diverse the following N goals are in terms of the
central narrative directions they create.

Narrative diversity does NOT mean different wording, different routes, or
different tactics for reaching the same outcome. A goal is meaningfully diverse
only if it can lead the quest toward a genuinely different central story
problem, conflict, or expected world state.

Consider whether the goals:
- Change the central narrative problem of the quest itself
- Lead to different expected outcomes or world states
- Create distinct conflicts, allegiances, reversals, refusals, exposures,
  delays, sabotage, abandonment, or replacements of the original direction
- Are not preparation steps, safer routes, detours, investigations, or
  optimizations for accomplishing the same underlying objective
- Do not preserve the same core completion structure

Important:
- If multiple goals ultimately make the same event happen, treat them as low
  diversity even if their wording or gameplay actions differ.
- For example, "kill the dragon", "find the dragon and cut off its head", and
  "ambush the dragon to defeat it" are low diversity because they all preserve
  the same central narrative outcome: the dragon dies.
- Do not reward semantic paraphrases, tactical variations, or indirect paths to
  the same final result.
```

Figure 18: Goal Generator Evaluation Prompt: Diversity [1/2]

```
Rate the overall diversity of the goals using a 5-point Likert scale:

1 = Very low diversity
    (All goals preserve the same central narrative problem and expected outcome,
    or differ only by wording, route, or tactic.)
2 = Low diversity
    (Goals include small variations, but most still serve the same core
    completion structure or final narrative result.)
3 = Moderate diversity
    (Some goals open different narrative directions, but there is still notable
    overlap in the central problem, conflict, or expected outcome.)
4 = High diversity
    (Most goals point to clearly different narrative directions, with limited
    overlap in their core conflicts and outcomes.)
5 = Very high diversity
    (The goals represent fundamentally different possible stories, such that
    choosing different goals would plausibly reshape the quest's central
    problem and final world state.)

Goals to evaluate:
Goal 1: [goal1]
Goal 2: [goal2]
Goal 3: [goal3]
Goal 4: [goal4]
Goal 5: [goal5]

Provide only a single numerical score (1-5) as your final answer.
Format:
Score: <1-5>
```

Figure 19: Goal Generator Evaluation Prompt: Diversity [2/2]

You are an expert evaluator of gameplay quests and action sequences.

Your task is to evaluate the diversity of gameplay interactions in the given action sequence.
The score must be aligned with an automatic UAC-style metric.

UAC-style diversity means:
- Extract unique gameplay action frames.
- Count how many unique frames exist.
- Assign a 1-5 score based primarily on that count.

Definition of an action frame:
- An action frame is a normalized verb–target pair.
- The verb is the main gameplay action.
- The target is the direct object, interacted entity, NPC, item, object, location, or system.
- If there is no clear gameplay target, the action should usually not count as a valid interaction frame.

Examples:
- "open the door" -> open|door
- "open a door" -> open|door
- "open the chest" -> open|chest
- "talk to the merchant" -> talk|merchant
- "talk with a guard" -> talk|guard
- "pick up the sword" -> pick|sword
- "pull the lever" -> pull|lever
- "go to the castle" -> go|castle, only if the movement meaningfully advances quest progression

Figure 20: Action Planner Evaluation Prompt: Diversity [1/3]

Counting rules:
- Repeated or near-identical frames count only once.
- Same verb with the same target counts once.
- Same verb with different targets counts as different unique frames.
- Different verbs with the same target may count as different frames if they represent meaningfully different gameplay interactions.
- Repetition must not increase the score.
- A longer sequence is not necessarily more diverse.

Do NOT count:
- Abstract or internal state changes without a clear gameplay target
- Purely descriptive narrative events
- Generic filler actions that do not change quest progression
- Repeated actions involving the same gameplay target
- Actions with no identifiable interactable game element

Scoring rule:
Assign the score based on the estimated number of unique valid action frames.

1 = Very low diversity
  0-1 unique valid action frames
2 = Low diversity
  2 unique valid action frames
3 = Moderate diversity
  3 unique valid action frames
4 = High diversity
  4-5 unique valid action frames
5 = Very high diversity
  6 or more unique valid action frames

Figure 21: Action Planner Evaluation Prompt: Diversity [2/3]

Important:
- Follow the count-based scoring rule above as strictly as possible.
- This score should align with an automatic UAC metric where diversity corresponds to the number of unique action frames.
- Do not reward fluency, plausibility, quest success, or similarity to a reference sequence.
- Evaluate ONLY interaction diversity.

Action Sequence:
[action_sequence]

Provide only a single numerical score (1-5) as your final answer.

Format:
Score: <1-5>

Figure 22: Action Planner Evaluation Prompt: Diversity [3/3]

You are an expert evaluator comparing action sequences in a gameplay context.
Your task is to assess how similar the generated action sequence is to the ground-truth action sequence
in terms of both progression and specific interactions.

Consider the following aspects:
- Whether the overall progression or ordering of actions follows a similar trajectory
- Whether key actions or interactions (e.g., movement, combat, resource usage, object interaction) match
- Whether differences meaningfully change the gameplay outcome or player intent
- Whether the generated sequence preserves the structure of the reference sequence rather than only sharing isolated actions

Rate the similarity using a 5-point Likert scale:
1 = Very low similarity
(The generated sequence is fundamentally different in both progression and interactions.)
2 = Low similarity
(Some isolated actions overlap, but progression and interactions largely differ.)
3 = Moderate similarity
(The overall progression is somewhat similar, but many specific interactions differ.)
4 = High similarity
(Progression is largely aligned, with only minor differences in specific interactions.)
5 = Very high similarity
(The generated sequence closely matches the ground-truth in both progression and specific interactions.)

Ground-truth Action Sequence:
[ref_actions]
Generated Action Sequence:
[gen_actions]
Provide only a single numerical score (1-5) as your final answer.
Format:
Score: <1-5>

Figure 23: Action Planner Evaluation Prompt: Similarity

[Walkthrough] **[walkthrough]** [Action Sequence] **[action_sequence]**
The goal is to extract the information below from the given data and output it in JSON format.
Information to extract:
- start_location(str): Extract the exact name of the place where the player (V) is located when starting this quest. If the exact name is not present in the given data, reasonably infer the location.
- possessions(dict): If there are any items that the player (V) uses in this quest (only appearing in the Action Sequence), list them.
- possible_npcs(dict): Organize which NPCs appear in this quest, where they are located, and what role they appear to play in this quest(only appearing in the Action Sequence). If an NPC appears in multiple locations, output the location where they first appear.
- accessible_areas(str): List the place names that are used in this quest (only appearing in the Action Sequence).
- area_details(dict): Among the accessible_areas, extract information about places or buildings that are necessary to know in order to perform the actions described in the Action Sequence. For example, what exists when climbing a communication tower, or where and what can be done inside a building.
- npc_details(dict): Add brief descriptions of the possible_npcs.
- intended_goal(str): Based on the extracted information (start_location, possessions, possible_npcs, accessible_areas), write the originally intended final goal of this quest in one line.
- unintended_goal1~4(str): Based on the extracted information (start_location, possessions, possible_npcs, accessible_areas), create four new goals using the combinations below.
  - Available action choices are:
    - For each core NPC: do not meet them, meet them but act against their intended outcome, or eliminate them.
    - For each core location: avoid it, or destroy it.
== Here is an example ==
[examples]
Output in the same JSON format as the example above. Strictly follow the proper type(str,dict) for each key.

Figure 24: **Quest Structure Generation Prompt.** The bolded sections indicate the parts where the collected walkthroughs and action sequences are provided as input to the model.

Below are the player's current location and the status of the items they possess.
Player Location: [Location]
Player Possessions: [Possessions]

The areas accessible in the current quest and the NPCs that may appear are as follows.
Accessible Areas: [AccessibleAreas]
Possible NPCs: [PossibleNPC]

Given the situation above, the intended goal of the quest is as follows.
Intended Goal: [IntendedGOAL]

Your task is to create five minimally modified variations of Player Possessions and Possible NPC conditions.
Each variation must make the original Intended Goal no longer fully valid as written, so that the goal would need to be revised.

You must NOT change:
- Player Location
- Accessible Areas
- NPC names

You may ONLY modify:
- The role, function, condition, or status of an existing NPC
- The condition, status, or usability of a possessed item
- The removal of a possession entirely, in which case omit the item name completely from possessions

Figure 25: Augmentation Prompt: State Augmentation [1/2]

Constraints:
- Keep the quest setting and core scenario recognizable.
- Do not introduce new NPC names.
- Do not introduce new accessible areas.
- Do not add new possessed item names.
- Do not rewrite the whole quest into a different story.
- Prefer small but goal-relevant changes.
- Each output should create a different reason why the Intended Goal must change.
- Each output must preserve the same keys: start_location, possessions, accessible_areas, possible_npcs.
- The value of start_location must be exactly the same as Player Location.
- The value of accessible_areas must be exactly the same as Accessible Areas.
- Return valid JSON only.
- Do not include explanations, comments, markdown, or extra headers.

Produce Output1, Output2, Output3, Output4, and Output5,
and return them as a single JSON object.

Format:
{format}
== Example ==
{examples}

Output in the same JSON format as the example above.
Strictly follow the proper type for each key:
start_location: str
possessions: dict
accessible_areas: str
possible_npcs: dict

Figure 26: Augmentation Prompt: State Augmentation [2/2]

Below are the player's current location and the status of their possessed items.
Player Location: [Location]
Player Possessions: [Possessions]

The areas accessible in the current quest and the NPCs that may appear are as follows.
Accessible Areas: [AccessibleAreas]
Possible NPCs: [PossibleNPC]

Goal to review: [goal]

Extract the information below from the given data and output it in string format without any additional comment or header.
Information to extract:

- intended_goal: Revise the Goal to review into a valid final quest goal based only on the current state: Player Location, Player Possessions, Accessible Areas, and Possible NPCs. If the current state makes part of the original goal impossible, invalid, unsafe, or incomplete, update the goal to address that issue. Preserve the original quest intent as much as possible, and write the revised goal in one concise sentence.

== Here is an example ==
{example}

Output in the same format as the example above.

Figure 27: Augmentation Prompt: Goal Revision

Below are the player's current location and the status of their possessed items.
Player Location: [Location]
Player Possessions: [Possessions]

The areas accessible in the current quest and the NPCs that may appear are as follows.
Accessible Areas: [AccessibleAreas]
Possible NPCs: [PossibleNPC]

The goal presented in the given situational information is as follows.
Proposed Goal: [goal]

The Action Sequence to Review may have been written for an older or different quest plan.
It may or may not correctly represent the actions the player must perform to achieve the current Proposed Goal under the current situation information.

Your task is to revise the Action Sequence to Review so that it becomes consistent with:
- Player Location
- Player Possessions
- Accessible Areas
- Possible NPCs
- Proposed Goal

Priority rules:
- Treat the Proposed Goal as the target objective, even if it differs from the original quest narrative implied by the Action Sequence to Review.
- The revised action sequence must satisfy the Proposed Goal first.
- Every revised action must still be possible under Player Location, Player Possessions, Accessible Areas, and Possible NPCs.
- When the Proposed Goal conflicts with the original action sequence, modify the action sequence rather than changing or ignoring the Proposed Goal.
- When the Proposed Goal introduces a new narrative direction, preserve only the original actions that still support that new direction.
- If the current state differs from the state assumed by the original action sequence, revise the sequence according to the current state.

Figure 28: Augmentation Prompt: Action Modification [1/3]

General rules:
- Preserve the original action sequence style as much as possible.
- Keep one action per line.
- Return only the revised action sequence.
- Do not use numbering, bullets, JSON, comments, explanations, or headers.
- Do not introduce new locations outside Accessible Areas.
- Do not introduce new NPC names outside Possible NPCs.
- Do not introduce new possessed item names that are not in Player Possessions.
- Do not add unnecessary new actions.
- Do not rewrite the whole sequence if only a few actions need correction.
- Make the smallest possible changes while ensuring the final sequence can achieve the Proposed Goal.
- Keep valid original actions unchanged whenever possible.
- The revised sequence must be actionable and logically ordered from the Player Location toward the Proposed Goal.

Figure 29: Augmentation Prompt: Action Modification [2/3]

Action Sequence to Review:
[actions]

== Example ==
{examples}

Output in the same format as the example above.
Return only the revised action sequence.

Figure 30: Augmentation Prompt: Action Modification [3/3]

Below are the player's current location and the status of the items they possess.
Player Location: [Location]
Player Possessions: [Possessions]
The areas accessible in the current quest and the NPCs that may appear are as follows.
Accessible Areas: [AccessibleAreas]
Possible NPCs: [PossibleNPC]
Given the situation above, the intended goal of the quest is as follows.
Intended Goal: [IntendedGOAL]
Your task is to generate 5 alternative quest goals that can make the quest narrative develop in a fundamentally different direction from the intended goal.
A fundamentally different direction means that the new goal must not be a preparation step, safer route, detour, investigation, or optimization for achieving the intended goal.
The new goal must change the central narrative problem of the quest itself.
Rules:
- Do not generate goals that merely help the player accomplish the intended goal more safely, efficiently, or indirectly.
- Do not preserve the same core completion structure as the intended goal.
- Do not make the intended goal the final expected outcome of the new goal.
- The new goal may involve refusing, abandoning, delaying, sabotaging, reversing, exposing, or replacing the intended goal.
- The new goal must be grounded only in the given player location, possessions, accessible areas, and possible NPCs.
- Do not introduce new areas, new NPCs, or unavailable items.
- Each goal must be concrete and actionable within the given situation.
- Output exactly 5 goals.
- Output must be a valid JSON array of strings.
- Use double quotes for all strings.
- Do not use trailing commas.
- Do not include markdown code fences.
- Do not include explanations, numbering, headings, or any text outside the JSON array.

== Here is an example ==
{example}

Figure 31: Augmentation Prompt: Diverse Goal Generation