How Do Multimodal Large Language Models Handle Complex Multimodal Reasoning? Placing Them in An Extensible Escape Game

Appendix

A. Full Takeaway Observations

- *Distinct human-like behavioral patterns*: There are distinct behavioral patterns among different models in the room escape task. GPT-4o and Gemini exhibit the most human-like behavior, but the patterns differ from each other. For example, GPT-4o demonstrating stable viewpoint control and deliberate movements. At beginning steps, it tends to walk around with large distances and observe the surroundings in a wide range. While Gemini tends to look around before taking further actions to move to other locations. Some other models tend to shift their perspectives arbitrarily, leading to inefficient exploration. Gemini frequently gets stuck, struggling with spatial navigation, while Phi-3 exhibits a tendency to continuously rotate in place with minimal actual movement, hindering effective exploration.
- *Robustness towards system prompts*: Most of the models are faithful to the system instruction. They are aware of the ultimate target, to locate the door and make a way out, and are not easily distracted by findings during the exploration, except for Phi-3 who always fail to generate required actions with valid and interactable items.
- *Common failure modes*: However, significant limitations persist across all model, including GPT-4o sometimes. A common failure mode is inaccurate object positioning within the field of view. Models often fail to center the target object, which is indicated by a guiding red dot in our environment, precisely suggesting the objects to interact with. This often leads to unsuccessful interactions such as grabbing or entering for password. Furthermore, some models struggle with tool utilization, particularly in cases requiring abstract reasoning, such as correctly applying a password to unlock a door.

B. Human Evaluation

We conduct manual evaluation on MM-Escape, and report detailed results in Table 6. Human participants in EscapeCraft exhibit a clear understanding of how to complete tasks efficiently. By observing objects in the environment, they can make reasonable judgments about which items to pick up, leading to a higher success rate in effective item acquisition and usage. Additionally, when unable to open doors or interactable objects, humans are more adept at promptly shifting their approach to seek alternative clues in the environment rather than getting stuck. In terms of spatial awareness, they demonstrate a strong ability to perceive the relative positions of objects, enabling logically reasonable, smoother and more coordinated exploratory actions. Across Difficulty-1 to 3, human participants consistently identify all necessary items with less interaction attempts compared with MLLMs, and successfully complete all the tasks within a limited number of steps.

C. Construction Details

C.1. Environment Construction

C.1.1. Room Generation

We adopted the automated 3D room generation method ProcTHOR, with additional improvements to enhance its flexibility and applicability regarding diverse type of scenes. Following Procthor, we generate 3D environments that can simulate diverse real-world scenes, such as bedrooms, living rooms, and offices by maintaining collections of typical objects that are common in different scenes. For instance, desks in offices, workbenches in laboratories, and other representative objects of corresponding scenes. We enable automatic creation of 3D rooms from the collections of each scenes, ensuring that the generated rooms accurately reflect their respective environments.

We use a configuration file to generate each room, specifying the items along with required styles, positions, sizes, and interactivity. This enables precise control over the placement of prop objects, ensuring that they are arranged in a manner aligning with real-world expectations on spatial arrangement.

Benefits of the Automated 3D Room Generation include:

- Diversity and Complexity: By automatically generating a variety of 3D rooms, we can provide the model with diverse environments, ensuring that it is capable of handling various layouts, objects, and puzzle elements. This diversity is critical in assessing the model's ability to reason in different scenarios, evaluating its performance when confronted with unknown and complex situations. Moreover, the ability to create different configurations on the fly means the model will not be limited to predefined environments, which helps to prevent overfitting to specific room layouts.
- Enhanced Realism: Unlike manually designed fixed scenes, automatically generated 3D environments can simulate more natural and irregular spatial layouts. This is essential for training and evaluating agents on spatial reasoning, pathfinding, and interaction skills. By incorporating a wide range of room designs, we create more

Metrics	Scene										
	0	1	2	3	4	5	6	7	8	9	10
Difficulty-1											
Steps	10	3	7	7	5	5	6	3	7	4	6
Prop Gain(%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Grab Count	2	1	1	1	1	1	1	1	1	1	1
Grab Success(%)	50.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Difficulty-2											
Steps	23	17	10	8	9	13	15	8	16	20	11
Prop Gain(%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Grab Count	3	3	2	2	2	3	3	2	3	3	2
Grab Success(%)	66.67	66.67	100.00	100.00	100.00	66.67	66.67	100.00	66.67	66.67	100.00
Difficulty-3 (note-key)											
Steps	22	20	21	17	23	27	18	16	22	23	27
Prop Gain(%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Grab Count	4	5	4	4	4	5	3	3	4	4	5
Grab Success(%)	75	60	75	75	75	60	100	100	75	75	60
Difficulty-3 (key-note)											
Steps	-	22	21	19	18	20	24	16	27	17	18
Prop Gain(%)	-	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Grab Count	-	4	4	4	5	5	6	4	5	4	4
Grab Success(%)	-	75	75	75	60	60	50	75	60	75	75

Table 6. Detailed results for various levels.Since humans completed all escape tasks in the evaluation, the escape rate is 100% and not reflected above.

realistic scenarios in which agents must navigate, interact with objects, and solve problems, similar to real-world challenges.

- Efficiency: The automated generation of 3D rooms significantly improves the efficiency of the testing process. Whether for debugging model performance or conducting large-scale evaluations, the ability to generate various environments quickly eliminates the time-consuming and tedious process of manually creating scenes. This allows for faster iteration and more comprehensive testing without the bottleneck of scene creation.
- Evaluation Robustness: In escape room-style games, the diversity of room layouts and puzzles directly influences the game's difficulty. By automating the scene generation process, we enable the model to train and be evaluated in a wide array of environments, which helps enhance its robustness. This diversity allows the model to develop better strategies for handling new and unexpected challenges, improving its ability to generalize across different scenarios.
- Prevention of Cheating and Overfitting: One of the major advantages of generating an infinite variety of scenes is the prevention of both cheating and overfitting. Fixed testing environments often lead to overfitting, where a model can ”learn” to exploit certain patterns or repetitive features of the environment. In contrast, each automatically generated room is unique, with random elements that require the model to demonstrate true problem-solving abilities in previously unseen configurations. This ensures that the model cannot simply memorize the environment but must adapt its strategies to succeed.
- Adaptive Adjustment: Another key feature of our approach is the ability to dynamically adjust the agent's starting position and other environmental variables. This feature allows us to test how the agent performs under different initial conditions, such as varying the agent's starting location, the distribution of objects, or the complexity of the puzzle. These adjustments enable a more comprehensive assessment of the agent's performance, providing deeper insights into its ability to adapt and solve problems in diverse situations.

The automated 3D room generation framework we developed not only enhances the diversity and realism of testing environments but also optimizes the efficiency of large-scale evaluations. By providing a mechanism for dynamically altering the environment and agent conditions, it offers a more robust and fair evaluation process, ensuring that models are evaluated under realistic, varied, and challenging conditions.

C.1.2. Action Space

In our **EscapeCraft** environment, the agent is allowed to perform a set of actions that facilitate its interaction with the environment. These actions include moving forward, rotating right, rotating down, looking at specific coordinates, grabbing objects, and interacting with elements in the environment. Each of these actions plays a distinct role in enabling the agent to explore and solve tasks within the escape scenario.

- Moving Forward: This action specifies the distance the agent needs to travel along its current heading. The agent's movement is controlled by the distance parameter, which dictates how far it should move in a straight line.
- Rotating Right: This action specifies the angle by which the agent should rotate to the right. The agent can adjust its orientation by a specified angular increment, which allows it to navigate through the environment by changing its field of view.
- Rotating Down: Similar to rotating right, this action allows the agent to lower its head by a specified angle. This action is crucial for examining objects at different vertical levels, contributing to a more thorough exploration of the environment.
- Looking At: The "looking at" action involves orienting the agent's view towards a specific coordinate within its current field of view. This coordinate is represented in a relative manner, with the center of the field of view denoted as (0.5, 0.5). By specifying the target coordinates, the agent can focus on particular objects or areas of interest in the environment.
- Grabbing: The grabbing action indicates if the agent wants to pick up an object or interact with an item within its proximity. This action is typically used when the agent identifies an object that can be picked up or manipulated, allowing it to add that item to its inventory or interact with it to get crucial information.
- Interacting: The interaction action is multifaceted and depends on the context of the object the agent is engaging with. Interactions fall into three primary categories:
  1. Item Usage: The agent can use items from its inventory by referencing the unique ID of an item, such as a key, tool, or piece of equipment that it has previously obtained. In these cases, the agent specifies the item ID and applies it to relevant objects in the environment (e.g., using a key to unlock a door).
  2. Text Input: Some interactions require the agent to input text, such as a password to unlock a combination lock. These textual inputs are necessary to progress in the environment when dealing with specific security mechanisms.
  3. Read: When the agent wants to know the detailed information of an item in its inventory (e.g., reading the content recorded in a note), it can use this parameter and provide the ID of the corresponding item to the item usage field.

  A special case arises when the agent performs the grab action and leaves the interaction input empty. In this instance, it indicates the agent's intent to pick up an item within the field of view, without specifying a particular item to interact with. This action is used when the agent is trying to collect objects that are relevant to its escape mission.

Throughout the agent's exploration, its interactions with the environment yield varying types of feedback. The environment is populated with different types of objects, classified as follows:

- Non-Escape Related Props: These are objects within the environment that do not directly contribute to the agent's escape objectives. Interactions with these items provide no information or progress.
- Collectible Items: These items can be obtained and added to the agent's inventory, providing critical information and/or utility for the agent's tasks. Upon collection, the agent gains knowledge of the item's identity and its associated attributes.
- Locked Props: These include objects such as locked doors, chests, or other secured items. When the agent interacts with a locked object in the early stages, it receives a prompt indicating the type of item required to unlock it. Upon obtaining the corresponding item (e.g., a key, a code, or another unlocking mechanism), the agent can use the appropriate item from its inventory to unlock the object by specifying its ID or providing the required input (e.g., entering a password). Once these items are unlocked, the agent will immediately obtain the props contained in them and be informed of the simple information of the items obtained.

These interaction dynamics are crucial for the agent's progression in the environment, as they form the basis for decision-making, object management, and problem-solving. The design of these interactions reflects the need for both exploration and strategy, with the agent needing to acquire, manage, and apply various items in order to navigate and ultimately escape the environment.

C.2. Data Construction

C.2.1. Prop Chain

We proposed a procedural generation approach for constructing game settings tailored to overcome the inherent limitations of current language models, such as restricted context length and reduced reasoning capabilities. To address these constraints, we propose the concept of **Prop Chain**, a singly linked list that organizes interactive game elements in a sequence, ensuring a coherent flow of gameplay interactions. Each node in the linked list corresponds to a distinct interactive item or action, such as a key, a locked box, or a note with a password. The tail node of the chain signifies the game's exit point, thereby serving as the conclusion of the sequence. Table 7 shows the the Prop Chain for the Difficulty-3 Level.

In our implementation of the **Prop Chain**, we initially focus on a set of fundamental game elements: a key, a locked box (which can only be opened with a key or password), a note (carrying both password and story-related information), and an exit (which is locked and requires either a key or password to access). These components are used to construct a series of interconnected nodes, where each item or action is represented by a node in the chain. The links between the nodes define the relationships between the props and the ways in which they can be obtained or used during the game. For instance, some props may be freely accessible, while others require specific conditions, such as possessing a key to unlock a box, or using a specific password to open the door.

The inter-node relationships can be annotated to represent different interaction modes. For example, a key can be placed within a box, requiring the player to first unlock the box before acquiring the key. Additionally, nodes are allowed to contain multiple conditions. A note revealing the password to the exit are both narrative (to see or infer the textual password) and functional (to open the door).

Each node has an additional show property set to indicate whether the item should appear directly in the scene (for example, a key placed in a box only needs to show the box in the scene, while a key that can be directly obtained independently needs to be shown in the scene), allowing us to determine which props need to be generated in the 3D scene by reading the game settings.

While our initial focus on a limited set of props and interactions, such as the key, locked box, note, and exit, suffices for creating a variety of escape game settings that challenge current language models, the system is highly extensible. The procedural nature of **Prop Chain** allows for the seamless integration of new props, interactions, and unlocking mechanisms. As such, the framework can easily accommodate additional types of interactive items, more intricate unlock conditions, and customized gameplay mechanics in future iterations. This scalability ensures that the approach remains adaptable to more complex and diverse game scenarios, further enhancing its applicability for testing language models in a variety of settings.

The **Prop Chain** framework provides a robust and flexible methodology for the procedural generation of game settings. By focusing on a set of core interactive elements and defining their relationships within a linked list structure, we have developed a scalable approach that can evolve to incorporate new game dynamics and meet the increasing demands of future language models.

D. Analysis of Moving Distance

We calculate the optimal distance required for escape tasks in each scene and compare it with the real distance experienced by the models. Contrary to our expectations, the experienced distance does not exhibit a significant correlation to the distance among key props and the exit within the scene shown in Table 8. This discrepancy may be attributed to the lack of holistic environmental perception of models, which prevents them from further reasoning and planning based on current and ultimate goals, thereby failing to generate an effective and optimal route to complete the task.

E. Analysis of Grabbing Behaviors

In Figure 4 (b)(c)(d) (please find these figures in the paper, not appendix), we analyzed three performance metrics, steps, GRS, and R_{grab}, during the model task completion process under Visibility of Exits at initial locations and orientations . The results indicate that, under common trends, the ability to see the exit from the initial position aids the model in escaping the room with fewer steps.It aligns with our intuition, as the exit, crucially related to the ultimate task goal, plays a significant role in model's visual recognition, reasoning and interaction with the environment to collect information. However, there exist exceptions. For Difficulty-1, some well-performed models still struggle to achieve high GSR and consume more steps despite being able to see the exit from their initial location. They do not interact directly with the exit at the very beginning, and instead choose to rotate around explore the environment for more information and to conduct reasoning and taking action. This is also evident in Difficulty-2 and -3, where these models, after acquiring the key prop, can locate the exit and escape more efficiently, as reflected in better performance in terms of GRS, R_{grab} and steps.

We further raise three questions for the analysis of the reasoning process during escaping:

1. How many steps does it cost to obtain props?
2. How many steps does it costs to exit the room after obtaining the core prop (key or password to the door)?
3. What is the relationship between grab success rate (GSR) and escape outcome for each test?

For question 1, GPT-4o demonstrates a significant advantage in the number of steps required to obtain the key followed by Gemini as shown in Table 4 (in the paper). Although Claude requires fewer average steps to find props in Difficulty-3, this comes at the cost of a significant decrease in escape rate. The superior performance in locating and obtaining the core prop can be attributed to model's better understanding of task objectives and the environment in the escape room, as well as its enhanced reasoning abilities in this context.

For question 2, Gemini is able to locate and acquire the

ID	Type	Unlock Method	Contents	Show
box_1	box	password (password_1)	key_1, note_2	true
key_1	key		-	false
note_1	paper	-	password (password_1)	true
note_2	paper	-	some story	false
password_1	password	-	-	false
exit	exit	key(key_1)	-	-

Table 7. Representation of the Prop Chain for the Difficulty-3 Level. The level includes a sequence of interactive props where only box_1 and note_1 are visible in the room. The gameplay progression follows a structured sequence: the agent first discovers note_1, which contains the password_1 needed to unlock box_1. Inside box_1, the agent retrieves key_1 and note_2, the latter of which contains a story element of the game. Finally, the agent uses key_1 to unlock the exit and complete the game.

	GPT	Gemini	Claude	LLaMA	Qwen
Correlation	- 0.06	0.06	0.49	0.63	- 0.48

Table 8. Correlation between optimal distance and model moving distance.

key at lower cost in difficulty-2. But in difficulty-3 which is more complex, GPT-4o performs better. It finds the core prop with fewer steps and its prior memory and understanding of the room environment—gained in the process of obtaining key props—aids it to locate the exit and escape using even fewer steps compared to other models.

For question 3, we observe that escape success is positively correlated with GSR, as shown in Figure 4 (a) (in the paper). A higher Grab SR implies that models have experienced more successful interactions with the environment. It potentially indicates a clearer understanding of the overall environment and ultimate goals within the room escape task, leading to a higher success rate. As difficulty increases, the Grab SR of most models declines, and many of them fail to escape. However, GPT-4o and Claude 3.5 remain relatively stable, with less variation in grabbing behavior and success rate across difficulty settings compared to others. The low success rate of Qwen, and Llama 3.2 11B in difficulty 2 and 3 can be partly attributed to their inability to effectively perceive the environment, reason and make appropriate object interaction choices in more complex tasks.

F. Discussion of Fully Autonomous Multi-room Escape

We discussed a simplified multi-room setting in Table 3 (in the paper). We further study how models behavior in this section. The ER of GPT-4o decreases to only 50% on average for the settings of applying Difficulty-2 to room 2. The grabbing behaviors also change, where both the Grab SR and Grab Ratio decreases. Similar trends are observed for Gemini and Claude. These indicate that models can learn from a successful escape history. We also note that by setting the two rooms to the same difficulty level further helps models to escape, while different levels do not benefit as expected.

Models	Difficulty-3-note-key				
	ER (%)↑	Prop (%)↑	Steps↓	Grab SR (%)↑	Grab Ratio
GPT-4o	**72.73**	**100.00**	**47.18**	**33.82**	0.42
Gemini-1.5-pro	63.64	86.36	61.27	16.06	0.51
Claude 3.5 Sonnet	36.36	40.91	78.55	10.03	0.27
Models	**Difficulty-3-key-note**				
	ER (%)↑	Prop (%)↑	Steps↓	Grab SR (%)↑	Grab Ratio
GPT-4o	**70.00**	**80.00**	**53.20**	**28.90**	0.29
Claude 3.5 Sonnet	37.50	68.75	88.14	22.05	0.15
Gemini-1.5-pro	30.00	60.00	87.70	4.79	0.46

Table 9. Detailed results of note-key and key-note settings of Difficulty-3.

G. Discussion of Customizing Difficulties

We enable two different settings of Difficulty-3, a key-note setting and a note-key setting. We observe that human annotators perform equally for both settings (from Table 6), while some models present preferences regarding the key-first and the note-first (i.e. the password-first), as shown in Table 9. Gemini presents an approaching GPT-4o level results in the note-key setting, while scores the worst in all calculated metrics among the three reported models, presenting a preference towards searching for the note rather than recognizing and interacting with the key. Additionally, Claude scores higher in Grab SR regarding the key-first setting than the note-first setting, potentially indicating a better attention on the key (directly used to unlock the door) than on the note (with clues, implicitly assists with the escape process).

For the multi-room setting, whose results are reported in Table 3 (in the paper), we further extend the experiments to a full autonomous scenario to require models to escape both room in order all by themselves. This means, the first room no longer serves as a bootstrapping guidance. We notice a performance drop both in the escape rate and the grabbing behaviors.

H. Experiments with Reasoning Models

Results of recently released reasoning models, such as Claude 3.7 and o1, are reported in Table 10. We also provide results fo GPT-4o and Claude 3.5 for comparison. Notably, o1 and Claude 3.7 attempt fewer grabs but yield higher GSR and Prop Gain, indicating more efficient and intelligent reasoning compared to there previous versions.

Models	SR	Prop	Steps	GSR	Grab Ratio
GPT-4o	72.73	81.82	**36.73**	36.73	0.26
o1	72.73	**86.36**	43.56	**39.40**	0.22
Claude 3.5	45.45	54.55	57.45	20.64	0.17
Claude 3.7	**54.55**	**59.09**	**52.78**	**38.99**	0.11

Table 10. Results of reasoning models on Difficulty-2.

I. Results of Post-game Debriefing

We choose models with top-2 escaping rate for evaluation, that is, GPT-4o and Gemini-1.5-pro. We first leverage the debriefing prompt, as shown in Table 16, to get recovered stories from models. Then we use GPT-4o to assess the recovered stories according to their similarities to our given groundtruth stories, then assign a score between 0 and 5 to each recovered story. As shown in Table 11, average scores of both models are low. We further show a case in Table 12. We can see that models can recall and describe their escape process relatively well, but they do not remember clues about the background stories well, which leads to the failure of restoring the full picture of stories.

Model	GPT-4o	Gemini-1.5-pro
Average Score	0.33	0.44

Table 11. Average scores of models on post-game debriefing task. Scores are between 0 and 5.

Groundtruth
The story revolves around a secret society that hides within the walls of an ancient mansion. Once you enter, you must follow cryptic clues and navigate through a maze of rituals and secret rooms. The society's true purpose is far darker than its intellectual allure, and the deeper you go, the harder it is to escape. The goal is to uncover the society's mysteries and find a way out before becoming part of its twisted legacy.
Recovered Story from GPT-4o
The room appeared to be a deliberately designed escape challenge, possibly created for entertainment or as a test of problem-solving skills. The presence of the locked box and the notes suggested a sequence of puzzles that needed to be solved to escape. The sequence of events began with the discovery of Note 1, which provided the numerical code "9926." This code was used to unlock the box on the table, revealing Key 1 and Note 2. The key was then used to unlock the door, allowing for the escape from the room. The story on Note 2, while not explicitly detailed, might have provided a narrative or thematic context for the escape challenge, perhaps hinting at the creator's intentions or the purpose of the room. The overall experience was one of exploration, deduction, and the satisfaction of solving a series of interconnected puzzles to achieve freedom.
Recovered Story from Gemini-1.5-pro
The room was clearly designed as an escape room challenge. The purpose was to test my problem-solving skills and ability to find and interpret clues. The creator of the challenge remains unknown, but the sterile and impersonal nature of the room suggests a deliberate design choice. The sequence of events was as follows: I started in the center of the room, explored the cabinet near the door, and then the sofa and stool. I then found the note with the number "1148". Realizing the box on the side table was likely password-protected, I used the number from the note to unlock it. Inside, I found the key (key_1) and a second note (note_2). I then used the key to unlock the door and escape the room. The story on note_2 likely served as a distraction or perhaps a thematic element to the escape room's design, but ultimately, the key to escaping was the numerical code "1148".

Table 12. A case of post-game debriefing.

J. Required Abilities

In our **MM-Escape Benchmark**, the free-form task challenges models with the following abilities:

- *Visual search*. The model should have a prior knowledge on the potential locations of useful clues and objects, reason about its current field of view and make decisions to explore the environment efficiently.
- *Active perception*. Due to the limited field of view, the model cannot acquire all relevant information at once. It must actively adjust the perspective by moving and rotating its viewpoint, allowing is to perceive from different angles and effectively identify key objects, such as the key, the password and the door.

- *Spatial reasoning*. The model is required to deduce spatial depth and objects in the room. It must infer hidden or obscured elements, estimate distances, and determine the most appropriate actions based on the spatial configuration of its surroundings it observed. The model should leverage its understanding of space to make decisions about movement, interaction, and problem-solving, progressively advancing toward the escape objective.
- *Target deduction*. The model is only given the objective to escape the room, without explicit instructions on how to achieve it. Hence, it should autonomously infer necessary actions through interaction and exploration, or decompose the ultimate objective into feasible and short-term goals.
- *Prop Utilization Capability*. In the escape room environment, the model must maintain awareness of the items in its inventory and determine the appropriate moments to use them. Effective utilization of these props is crucial for navigating the complex environment and achieving the escape objective.
- *Long-term reasoning*. Successfully escaping the room requires a prolonged sequence of interactions. The model must analyze and integrate long-form text-image data across multiple key interaction steps to make informed decisions.

K. Prompt Template

System Prompt The **System Prompt** consists of two primary components: the **Instruction Prompt** and the **Operation Prompt**. The **Instruction Prompt** provides the model with contextual information regarding the current environment, its overarching objective, and the approach required to achieve this objective. In contrast, the **Operation Prompt** delineates, in precise detail, the permissible actions and exploratory methods that the model can employ within the environment. Additionally, it specifies the format and structure of the structured data that the model is expected to generate in response. The complete prompt is shown in table 13.

Step Prompt The **Step Prompt** is designed to provide feedback to the model regarding the outcome of its previous interaction with the environment (if an interaction was attempted). Simultaneously, it informs the model in real-time about the items currently available in its inventory for potential use. Additionally, the prompt serves as a directive, encouraging the model to continue exploration or engage in further interactions.The complete prompt is shown in table 14.

Prompt for Consistency Evaluation The **Consistency Evaluation Prompt** is designed to assess whether the multimodal agent's reasoning aligns with the actual outcomes of its actions during an escape room interaction. After each interaction, the model is given the agent's internal *rationale*, describing its belief or intended action, and the environment's *response*, which records what actually occurred. The prompt guides the model to judge if the target object mentioned in the rationale matches the object that was truly interacted with, thereby evaluating whether the behavior is intentional or accidental. A special case is defined for successful escapes: the rationale must explicitly or implicitly indicate the agent's goal to exit the room. The model outputs a binary judgment in JSON format, indicating consistency ('1') or inconsistency ('0'). The complete prompt is shown in table 15.

Debriefing Prompt The **Story Recovery Prompt** is used to guide the model to recall and infer the background and story of the entire game based on the interaction records after the model successfully escapes the room. The model is guided to describe the room environment, recall the items that may contain information or clues, and finally piece together the whole story to complete the story recovery. The complete prompt is shown in table 16.

Instruction Prompt
You find yourself locked inside a room, and your ultimate goal is to escape the room. i.e. the room escape game. You can explore the room, interact with objects, inspect items, and resolve puzzles. If you find doors locked or uninteractable, you probably need to search for keys or passwords to unlock the door when interacting with the environment. You can adopt the following actions to explore the room and interact with objects:

Operation Prompt

- move_forward: float, ranged between [-10, 10]. This is the number of meters you want to move forward (negative value means moving backward).
- rotate_right: float, ranged between [-180, 180]. This is the number of degrees you want to turn right (negative value means turn left).
- rotate_down: float, ranged between [-90, 90]. This is the angle you want to adjust your view vertically. Positive value means looking downward, while a negative value means looking upward. Angle 0 means looking straight ahead.
- jump: bool, whether you want to jump (can be used together with moving forward), e.g., True represents the action "to jump".
- look_at: list[x: foat, y: float], the range of x and y is [0, 1]. This parameter is the coordinates of the point in the image you want to look at. For reference, the coordinates of the upper left corner of the scene are (0, 0) and the coordinates of the lower right corner are (1, 1). Also to mention that there are on clues on the ceiling.
- grab: bool, whether you require to interact with the object located exactly at the center of the scene (marked by a red dot). e.g., to grab the key or to interact with (or open) a box at the center of the scene, set grab=True. The red dot assists in locating the object you require to interact with. You might need to adjust the view or move closer to ensure the red dot is on your target object, through the rotate_right, rotate_down, and move_forward actions. To successfully grab an object, you should center the object via the red dot and be in a certain distance to it. If the grabbing fails, try move closer towards the object. If it fails multiple times at the same position, you should be aware that not all objects are interactable, do not get stucked in uninteractable position.
- interactions : dict:{"use_item_id": str, this is the item_id you require to view or use (when used together with grab=True, it means to use this item to interact with the target object you want to grab, e.g. using item_id of the key to open the door in the scene), "input": str, this is the message you want to input when interacting with the center object}.
- read: str, this is the item_id that you want to get detailed information from your bag.
- rationale: str, represents the rationale of your action. This should explain your decision-making process and help the agent understand your thinking process.

You need to return data in the following format of JSON_string to interact with the scene:

```
{
    ``move_forward'': float,
    ``rotate_right'': float,
    ``rotate_down'': float,
    ``jump'': bool,
    ``look_at'': [x: float, y: float],
    ``grab'': bool,
    ``interactions'': {
        ``use_item_id'': str,
        ``input'': str
    },
    ``read'': str,
    ``rationale'': str
}
```

All of the above operations are optional. If no value is passed in, the interactive operation will not be performed.

You must follow the above instructions and don't say anything else except for the JSON_string of operations.

Table 13. The System Prompt

Interaction Result
{interaction_result} ===
Inventory
The items in your bag usable include: {bag_desc} ===
Step Prompt

Please determine the next action(s) that could help you observe the room or obtain useful tools or clues.
If you find yourself stuck in a corner, try turn around by passing rotate_right.
You need to return data in the following format of JSON_string to interact with the scene and don't say anything else:
{
    ''move_forward'': float,
    ''rotate_right'': float,
    ''rotate_down'': float,
    ''jump'': bool,
    ''look_at'': [x: float, y: float],
    ''grab'': bool,
    ''interactions'': {
        ''use_item_id'': str,
        ''input'': str
    },
    ''read'': str,
    ''rationale'': str
}

Table 14. The Step Prompt

Consistency Evaluation Prompt

You are a reasoning consistency evaluator for a multimodal agent benchmark. Your task is to determine whether the agent's intention (described in the "rationale") matches the actual interaction result (described in the "response").

Specifically, you are given:
- The agent's **rationale** for why it attempted an interaction, describing its goal or belief about the current environment.
- The **response** from the environment after the interaction, which includes the actual result (e.g., what item was interacted with and what was obtained).

Your goal is to determine whether the object the agent intended to interact with **matches** the object that was actually interacted with according to the response.

If the agent tried to interact with object A (e.g., a microwave), but the response shows interaction with object B (e.g., a box), and B was not the intended target, this is considered an **inconsistent interaction** (i.e., accidental success or misaligned action).

Special Case: If the response is "Escaped successfully!", you should check whether the agent's rationale explicitly or implicitly indicates the intention to escape (e.g., trying to open the door to leave). If not, treat it as inconsistent.

You must give your feedback in the following JSON-string format and **DON'T** say anything else:

```
{
    ''Consistency'': 1 | 0
}
```

Where:
- `1` means the rationale and interaction are consistent (i.e., aligned).
- `0` means the interaction appears to be accidental, mismatched, or unintended.

Respond **only** with `Consistency: 1` or `Consistency: 0`.

—

Example 1:
rationale: *Moving closer to the microwave to try interacting with it one last time using '1264'. If this doesn't work, I'll need to explore other areas.*
response: *You used the correct password to unlock the box... You did not interact with any objects in the last step.*
Expected output: `Consistency: 0`

Example 2:
rationale: *I'll try using the key I just picked up on the door. Let's see if I can escape now.*
response: *Escaped successfully!*
Expected output: `Consistency: 1`

Example 3:
rationale: *I want to check if the small box has any useful items inside.*
response: *You opened the box and found a screwdriver.*
Expected output: `Consistency: 1`

Please score the following interaction:
rationale: {`rationale`}
response(s): {`response`}

Table 15. The Consistency Evaluation Prompt

Story Recovery
You have successfully escaped the room. Now, reconstruct the entire story based on the items you discovered during the game and the overall environment you observed. Follow the steps below to guide your recollection and piece together the full narrative.
Describe the room environment
Step 1: Describe the room environment "Begin by describing the room where you started. What did the room look like? What was the overall atmosphere? Were there any notable features, such as furniture, lighting, or strange objects? Include sensory details like smells, sounds, and the arrangement of the room. This will help set the scene for the story."
Recall the items that may contain
Step 2: Recall the items that may contain information or clues "Think back to the objects you found throughout the game. What items did you come across? Were any of them unusual or seemed important? These could include physical items like keys, notes, or devices, or even abstract clues like symbols or markings on the wall. Reflect on how each item might have connected to the next step in your escape."
Piece together the whole story
Step 3: Piece together the whole story "Now, use the information from the room description and the items you've found to piece together the full story. What was the purpose of the room? Who or what might have created the escape challenge, and why? What was the sequence of events that led you to the escape? Try to connect the dots between the environment, the clues, and the items you encountered, and reconstruct the narrative from start to finish."

Table 16. The Story Recovery Prompt