Turitto

Giuseppe Turitto <giuseppe@turitto.com>

specs
1 message

Giuseppe Turitto <giuseppe.turitto@kiwi.com> Sat, Oct 4, 2025 at 6:36 PM
To: giuseppe@turitto.com

Specification Document: Labyrinth: The Temporal Hunt (MVP)

This document provides the definitive specifications for the **Labyrinth: The Temporal Hunt** project. It outlines the core physics, communication protocols, and agent logic required to build the multi-agent text adventure simulator.

---

1. Core Definitions and Game Constants

All game calculations and physics are handled by the deterministic **Game State Manager (GSM)**.

Parameter	Value	Description
Grid Unit Size	meters	The base size of one *step* or *tile* in the simulation.
Maze Size (Conceptual)	km cube	Defines the conceptual scale of the world. Coordinates will be tracked using large integer step units.
User Max Height		Used for collision/LOS checks.
Minotaur Height		Emphasizes its size relative to the ceiling/floor clearance.
User Sight Radius	**2 steps (5 meters)**	Maximum distance for detecting paths or items.
Minotaur Sight Radius	**6 steps (15 meters)**	Minotaur's infrared vision advantage.
Jump Cooldown	**600 seconds (10 minutes)**	Deterministic timer for the Minotaur's temporal jump ability.
Lantern Paralysis	**120 seconds (2 minutes)**	Duration of the Minotaur's paralysis.
Lantern Respawn	**720 seconds (12 minutes)**	Cooldown after the **Lantern** is *used* before a new one can spawn.
Objectives	Red, Blue, Yellow Stones	Three unique items required to trigger the ESCAPED status.

---

2. Game State Manager (GSM) Logic

The GSM is the central authority and executes all deterministic rules based on the JSON input.

2.1 Movement and Time Flow

Command	Action	Time & Noise Calculation	Stop Conditions
MOVE [DIR] 1 (Walk/Crawl)	1 step/second. Low Noise signature. Stamina recovery.	Stops upon hitting a **COLLISION** (wall, ramp edge) or if **100 steps** are reached.	

Command	Action	Time & Noise Calculation	Stop Conditions
MOVE [DIR] 2 (Run)	2 steps/second (Max speed). **High Noise** signature. Stamina depletion.	Stops upon hitting a **COLLISION**, **100 steps** reached, or **ENCOUNTER** with Minotaur.	
GRAB / USE / LOOK / HALT	Interaction/Utility.	Consumes a minimum of **1 second** of game time.	N/A

Collision Logic: The GSM must iterate through the user's requested 100 steps. If a collision is detected at step , the user is placed at step , and the `steps_moved` is set to . The `time_taken` is calculated as .

2.2 Minotaur Temporal Logic

The Minotaur's actions during the chase phase are driven by its LLM output, but constrained by the GSM's fixed rules:

1. **Paralyzed Status:** If `lantern_active_timer > 0` , Minotaur status is **PARALYZED** regardless of other factors.
2. **Temporal Jump:** If the Minotaur LLM outputs `action: "JUMP"` , the GSM sets the status to **VANISHED** and resets the `cooldown_time` to 600s. The Minotaur's **position is held constant** (Positional Jump).
3. **Re-entry:** If status is **VANISHED** and the simulated jump duration (5-10 seconds) has elapsed, the status is set back to **CHASING_3D** at the same positional coordinates.
4. **Encounter:** If the User and Minotaur occupy the **same grid step** while the Minotaur is in **CHASING_3D** status, the game ends (`status: "DEATH"`).

---

3. Communication Protocol (JSON Schemas)

All interactions are governed by **Pydantic Schemas** to ensure data integrity (N.F.R. 2.1.1).

3.1 User Input Schema (Input to GSM)

The User Agent (Mistral) or human input must conform to this structure:

JSON

```
{
  "command": "MOVE" | "HALT" | "LOOK" | "GRAB" | "USE",
  "direction": "NORTH" | "SOUTH" | "UP RAMP" | null,
  "steps": 100,
  "speed": 1 | 2,
  "target": "RED STONE" | "LANTERN" | null
}
```

3.2 Minotaur Decision Schema (Output from Gemini)

The Minotaur Agent must return its calculated action based on this structure:

JSON

```
{
  "action": "PATHFIND" | "JUMP" | "WAIT" | "CHASE",
  "target_coords": {"x": 0, "y": 0, "z": 0} | null
}
```

3.3 Game State Response Schema (Output from GSM)

The GSM's primary output to the Streamlit UI and the next agent turn:

JSON

```
{
  "status": "SUCCESS" | "DEATH" | "ESCAPED" | "ERROR",
  "user_state": {
    "position": {"x": 0, "y": 0, "z": 0},
    "stamina_pct": 0.85,
    "inventory": ["RED STONE", "LANTERN"],
    "lantern_cooldown": 720
  },
  "environment": {
    "visible_paths": ["NORTH", "DOWN RAMP"],
    "visible_items": ["BLUE STONE"],
    "message": "Current Z-Level: 3",
    "steps_moved": 10,
    "time_taken": 5,
    "stop_reason": "COLLISION" | "ENCOUNTER" | "SUCCESS",
    "ambient_noise": "HIGH" | "LOW"
  },
  "minotaur_cue": {
    "proximity": "VERY CLOSE" | "VANISHED" | "PARALYZED",
    "audio_direction": "EAST" | null,
    "temporal_status": "CHASING_3D",
    "cooldown_time": 450
  },
  "raw_text_output": "The Minotaur materializes at its fixed re-entry position!"
}
```

---

4. Agent Logic and Learning (N.F.R. 3.2.2)

4.1 Learning Integration

- **Memory Manager (** `memory_manager.py` **):** Responsible for persisting past game outcomes (reflections) and retrieving them.
- **Prompt Injection:** At the start of a new game run, reflections summarizing relevant past failures (User deaths for the User Agent, User escapes for the Minotaur Agent) **must** be injected into the LLM's system prompt to enforce strategic adaptation.

4.2 Agent Responsibilities

Agent Role	Model	Primary Function	Learning Focus
Minotaur	**Gemini**	Predictive pursuit, maximizing the use of the rare Temporal Jump and sight advantage.	Identifying the User's evasion patterns following the Temporal Whine cue.
User	**Mistral**	**Risk management**, map memory, and calculating the optimal low-noise path to the next Stone objective.	Optimizing Lantern usage timing and maximizing distance during Minotaur cooldowns.

Specification Document: Labyrinth: The Temporal Hunt (MVP)

This document provides the definitive specifications for the **Labyrinth: The Temporal Hunt** project. It outlines the core physics, communication protocols, and agent logic required to build the multi-agent text adventure simulator.

---

1. Core Definitions and Game Constants

All game calculations and physics are handled by the deterministic **Game State Manager (GSM)**.

Agent Role	Model	Primary Function	Learning Focus

Parameter	Value	Description
Grid Unit Size	meters	The base size of one *step* or *tile* in the simulation.
Maze Size (Conceptual)	km cube	Defines the conceptual scale of the world. Coordinates will be tracked using large integer step units.
User Max Height		Used for collision/LOS checks.
Minotaur Height		Emphasizes its size relative to the ceiling/floor clearance.
User Sight Radius	**2 steps (5 meters)**	Maximum distance for detecting paths or items.
Minotaur Sight Radius	**6 steps (15 meters)**	Minotaur's infrared vision advantage.
Jump Cooldown	**600 seconds (10 minutes)**	Deterministic timer for the Minotaur's temporal jump ability.
Lantern Paralysis	**120 seconds (2 minutes)**	Duration of the Minotaur's paralysis.
Lantern Respawn	**720 seconds (12 minutes)**	Cooldown after the **Lantern** is *used* before a new one can spawn.
Objectives	Red, Blue, Yellow Stones	Three unique items required to trigger the ESCAPED status.

---

2. Game State Manager (GSM) Logic

The GSM is the central authority and executes all deterministic rules based on the JSON input.

2.1 Movement and Time Flow

Command	Action	Time & Noise Calculation	Stop Conditions
MOVE [DIR] 1 (Walk/Crawl)	1 step/second. Low Noise signature. Stamina recovery.	Stops upon hitting a **COLLISION** (wall, ramp edge) or if **100 steps** are reached.	
MOVE [DIR] 2 (Run)	2 steps/second (Max speed). **High Noise** signature. Stamina depletion.	Stops upon hitting a **COLLISION**, **100 steps** reached, or **ENCOUNTER** with Minotaur.	
GRAB / USE / LOOK / HALT	Interaction/Utility.	Consumes a minimum of **1 second** of game time.	N/A

Agent Role	Model	Primary Function	Learning Focus

Collision Logic: The GSM must iterate through the user's requested 100 steps. If a collision is detected at step , the user is placed at step , and the `steps_moved` is set to . The `time_taken` is calculated as .

2.2 Minotaur Temporal Logic

The Minotaur's actions during the chase phase are driven by its LLM output, but constrained by the GSM's fixed rules:

1. **Paralyzed Status:** If `lantern_active_timer > 0` , Minotaur status is **`PARALYZED`** regardless of other factors.
2. **Temporal Jump:** If the Minotaur LLM outputs `action: "JUMP"` , the GSM sets the status to **`VANISHED`** and resets the `cooldown_time` to 600s. The Minotaur's **position is held constant** (Positional Jump).
3. **Re-entry:** If status is **`VANISHED`** and the simulated jump duration (5-10 seconds) has elapsed, the status is set back to **`CHASING_3D`** at the same positional coordinates.
4. **Encounter:** If the User and Minotaur occupy the **same grid step** while the Minotaur is in **`CHASING_3D`** status, the game ends (`status: "DEATH"`).

---

3. Communication Protocol (JSON Schemas)

All interactions are governed by **Pydantic Schemas** to ensure data integrity (N.F.R. 2.1.1).

3.1 User Input Schema (Input to GSM)

The User Agent (Mistral) or human input must conform to this structure:

JSON

```
{
  "command": "MOVE" | "HALT" | "LOOK" | "GRAB" | "USE",
  "direction": "NORTH" | "SOUTH" | "UP RAMP" | null,
  "steps": 100,
  "speed": 1 | 2,
  "target": "RED STONE" | "LANTERN" | null
}
```

3.2 Minotaur Decision Schema (Output from Gemini)

The Minotaur Agent must return its calculated action based on this structure:

JSON

```
{
  "action": "PATHFIND" | "JUMP" | "WAIT" | "CHASE",
  "target_coords": {"x": 0, "y": 0, "z": 0} | null
}
```

3.3 Game State Response Schema (Output from GSM)

The GSM's primary output to the Streamlit UI and the next agent turn:

JSON

```
{
  "status": "SUCCESS" | "DEATH" | "ESCAPED" | "ERROR",
  "user_state": {
    "position": {"x": 0, "y": 0, "z": 0},
    "stamina_pct": 0.85,
    "inventory": ["RED STONE", "LANTERN"],
    "lantern_cooldown": 720
```

Agent Role	Model	Primary Function	Learning Focus


```
  },
  "environment": {
    "visible_paths": ["NORTH", "DOWN RAMP"],
    "visible_items": ["BLUE STONE"],
    "message": "Current Z-Level: 3",
    "steps_moved": 10,
    "time_taken": 5,
    "stop_reason": "COLLISION" | "ENCOUNTER" | "SUCCESS",
    "ambient_noise": "HIGH" | "LOW"
  },
  "minotaur_cue": {
    "proximity": "VERY CLOSE" | "VANISHED" | "PARALYZED",
    "audio_direction": "EAST" | null,
    "temporal_status": "CHASING_3D",
    "cooldown_time": 450
  },
  "raw_text_output": "The Minotaur materializes at its fixed re-entry position!"
}
```


---

4. Agent Logic and Learning (N.F.R. 3.2.2)

4.1 Learning Integration

- **Memory Manager (`memory_manager.py`):** Responsible for persisting past game outcomes (reflections) and retrieving them.
- **Prompt Injection:** At the start of a new game run, reflections summarizing relevant past failures (User deaths for the User Agent, User escapes for the Minotaur Agent) **must** be injected into the LLM's system prompt to enforce strategic adaptation.

4.2 Agent Responsibilities

Agent Role	Model	Primary Function	Learning Focus
Minotaur	**Gemini**	Predictive pursuit, maximizing the use of the rare Temporal Jump and sight advantage.	Identifying the User's evasion patterns following the Temporal Whine cue.
User	**Mistral**	**Risk management**, map memory, and calculating the optimal low-noise path to the next Stone objective.	Optimizing Lantern usage timing and maximizing distance during Minotaur cooldowns.
