Planning-Based Game Development: A PDDL Approach to Gameplay Observability and Meta-Awareness

Lucas Gobbi Bergholz[1], **Bruno César Ribas**[1,2]

[1]Faculdade de Ciências e Tecnologias em Engenharia – Universidade de Brasília (UnB)

[2]Departamento de Informática – Universidade Federal do Paraná (UFPR)

lucas.bergholz@gmail.com, bruno.ribas@unb.br

Abstract. Introduction*: This work presents Gobbi's Goblins, a text-based role-playing game (RPG) that leverages Artificial Intelligence (AI) planning to develop interactive storylines and foster player agency.* ***Objective:*** *To investigate how planning-based systems can go beyond dynamic storyline generation by promoting player agency and meta-awareness through narrative observability.* ***Methodology****: The project employs numeric planning to formally represent the game domain and quest system. Player interactions through text commands update the planning state, allowing the system to dynamically recompute and expose, at any moment, the set of reachable narrative outcomes.* ***Results:*** *This architecture aims to enable meta-awareness by making narrative consequences explicit, enhancing player agency as users deliberately navigate toward or away from desired endings based on real-time feedback.*
Keywords *Artificial Intelligence, Automated Planning, Interactive Storylines, Player Agency, Digital Games.*

1. Introduction

Modeling real-world systems is often challenging due to the large number of intangible variables that influence them. However, in controlled environments with well-defined rules, such as digital games, AI Planning becomes particularly effective [Haslum et al. 2019].

The use of AI Planning in games is not new. Earlier in the century, games such as Halo 2 demonstrated its potential to enhance player experience [Microsoft Learn 2021]. Relatively few works have explored its application to game narratives, but two standout examples are the works of [Li e Riedl 2010] and [Soares de Lima et al. 2014]. By representing a game's storyline as a planning problem, it becomes possible to generate experiences that adapt to player decisions, making narrative progression more engaging.

To explore this idea, this work adapts planning-based narrative generation techniques to a role-playing game (RPG) called Gobbi's Goblins[1]. Key challenges include modeling narrative structures as planning problems, managing dynamic world states, and ensuring coherent responses to player input.

In particular, it demonstrates how numeric planning can be used not only for modeling but also for executing a fully playable game. The proposed architecture also aims to enable *narrative observability*, by allowing players to inspect the set of reachable

[1]Available at: https://github.com/LucasBergholz/GobbisGoblins

outcomes of the game at runtime. This, in turn, promotes *meta-awareness*, understood as the player's ability to reason about the narrative structure and the consequences of their actions during gameplay [Flavell 1979].

In summary, the main contributions of this work are: (i) the design of a planning-centric architecture for game development, (ii) the implementation of a fully playable game using numeric PDDL (Planning Domain Definition Language), and (iii) the introduction of narrative observability, via planning, as a mechanism to promote player meta-awareness and agency in interactive storytelling.

This paper is organized as follows. Section 2 presents the background concepts. Section 3 introduces Gobbi's Goblins game. Section 4 describes the game architecture. Section 5 details the PDDL representation. Finally, Section 6 discusses contributions compared to the related work and Section 7 concludes the paper and presents future work.

2. Background

This section introduces key concepts necessary for understanding this work. The first subsection discusses the relationship between Player Experience and Game Narrative. The second and third subsections present related work that serves as the foundation for this research. Finally, the last subsection describes an important work that is integrated into the main architecture of the game.

2.1. Player Experience and Game Narrative

As the game industry has grown into the largest entertainment sector worldwide [Arora 2023], understanding player engagement has become central to game design. According to [Schell 2014], the goal of game design is to create experiences that sustain player attention over time.

Engagement can arise from factors such as goal completion, exploration, social interaction, and competition, often associated with player typologies [Bartle 1996]. Among these, narrative plays a key role in structuring player experience.

Narratives can be understood as sequences of events forming a coherent whole [Prince 2003]. In games, they correspond to event sequences within a virtual environment [Li e Riedl 2010], with the defining characteristic of interactivity, as player actions influence their progression.

Thus, narrative provides context and coherence to player actions, supporting immersion and engagement. This establishes the foundation for approaches that seek to model and manipulate narrative structures computationally.

2.2. Planning-Based Approaches to Individualized Game Narratives

Planning for Individualized Experiences with Quest-Centric Game Adaptation, presented by Li and Riedl at ICAPS 2010, proposes modeling game narratives as plans, enabling the dynamic adaptation of storylines while preserving narrative coherence.

By representing narratives as partial-order plans, with causal relations linking actions, and temporal ordering constraints, the study concludes that adapting narrative content to individual player preferences and behavior can significantly enhance player experience. While the proposed adaptation algorithm requires further empirical

validation, the work demonstrates the strong potential of AI planning as a foundation for interactive storytelling [Li e Riedl 2010].

2.3. Hierarchical and Dynamic Quest Generation

Another relevant line of research is presented in Hierarchical Generation of Dynamic and Nondeterministic Quests in Games [Soares de Lima et al. 2014]. This work focuses on the dynamic generation of quests using nondeterministic hierarchical planning, integrating planning, execution, and monitoring to handle player-driven variability.

In this approach, quests are treated as fundamental storytelling units, particularly within role-playing games. Narratives are represented as hierarchies of quests, each modeled as an independent planning problem, where higher-level quests are decomposed into sub-quests and tasks, which aligns with the principles of Hierarchical Task Network (HTN) planning [Soares de Lima et al. 2014].

There are two main modules in the proposed architecture: a Quest Planner responsible for generating plans and a Quest Monitor that observes execution and detects deviations caused by player actions. If discrepancies arise between the expected and actual world states, replanning is triggered to maintain narrative consistency.

The authors conclude that planning-based quest generation offers new possibilities for interactive storytelling by combining nondeterminism, adaptation, and player modeling. Subsequent work by [Soares de Lima et al. 2016] extends this framework by incorporating player behavior modeling through machine learning.

2.4. bni: A pddl to C compiler with integrated repl for interactive testing

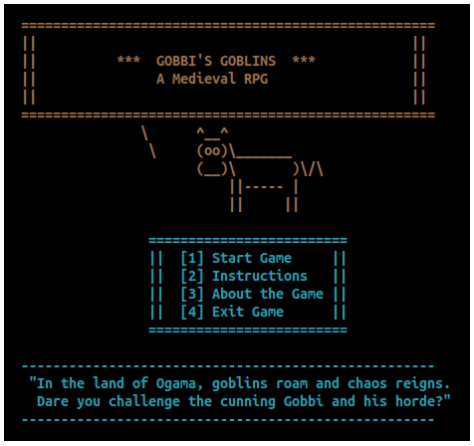
The bni software consists of an integrated parser and REPL environment, responsible for converting PDDL files into manipulable data structures in the C programming language, allowing the user to interact with the world state defined in the domain file. First, through its parser, the module reads and converts the PDDL domain and problem files into C data structures. This process translates predicates, functions, and actions from their declarative representation into executable C code [Ribeiro et al. 2025].

Second, the REPL (Read-Eval-Print Loop) component of the bni module acts as the interpreter of the model during runtime. It continuously receives inputs from the user, validates whether it corresponds to an available action, checks preconditions, applies effects, and updates the world state accordingly [Ribeiro et al. 2025]. To implement this, the bni REPL module enumerates the possible actions provided by the `pddl.c` domain file (created by the parser) into the user command prompt and waits for the user input.

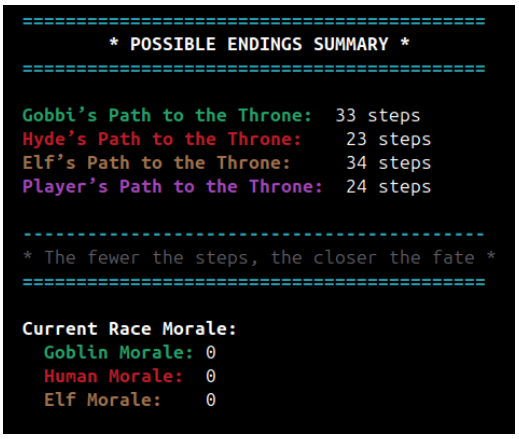
3. The Game

The game, titled Gobbi's Goblins, places the player in a medieval world set in the kingdom of Ogama, which is under the chaotic rule of a goblin mafia led by the cunning Gobbi. The kingdom, ruled by the controversial King Jekyll Hyde, faces serious structural problems as the city has become overcrowded beyond its current capacity. Food shortages and lack of housing haunt Ogama's future. Amidst this, crime continues to rise.

In the peripheral district of Goblin's Den, a popular yet dangerous leader emerges in the name of Gobbi, who, through his gang, spreads chaos throughout the city, causing



(a) ASCII art of Main Menu.



(b) Morale System interface.

Figure 1. Game interfaces: (a) Main menu in ASCII; (b) Morale system.

severe impacts on the kingdom structure as a whole. Jekyll hires the player to deal with Gobbi, offering any reward in exchange for ending the gang. The player must, amidst the mysteries surrounding the true goals of both Hyde and Gobbi, decide what to do.

3.1. Gameplay

The game is played entirely through the terminal, with human–computer interaction occurring via user text inputs. Figure 1a presents the game main menu providing a look on how the game looks in the terminal. For the player to perform an action in the game world, the player is shown a list of available actions and must either type the corresponding number of the desired action or write it out in full. The available player actions reflect the game's spatial and temporal relationships, meaning that, depending on the current location and moment in the game, a specific set of actions becomes available.

Similar to the structure proposed in prior works, the game is organized into quests, each with a self-contained narrative. Gobbi's Goblins features ten quests, of which only two are mandatory: the encounter with Gobbi The Goblin and the final return to King Hyde. The remaining quests can be completed in any order, allowing players to engage in between two and ten quests per playthrough, resulting in a non-linear narrative shaped by player choices and moral alignment.

The game also includes three playable races (Human, Elf, and Goblin) and five main locations. Each quest follows a linear sequence of actions with variation at its conclusion, which in turn can influence a set of three scores that the player has (a morale system) with each of the races. These scores define preconditions and effects for actions inside the game and define its ending, crowning one of the main characters as the King.

There are two categories of actions: quest-related and non-quest-related. Actions limited to a quest's scope can only be performed while that quest is active. Non-quest actions, on the other hand, can be performed at any moment. The player is free to move between locations at will, even during a quest, but must complete the current quest before beginning a new one.

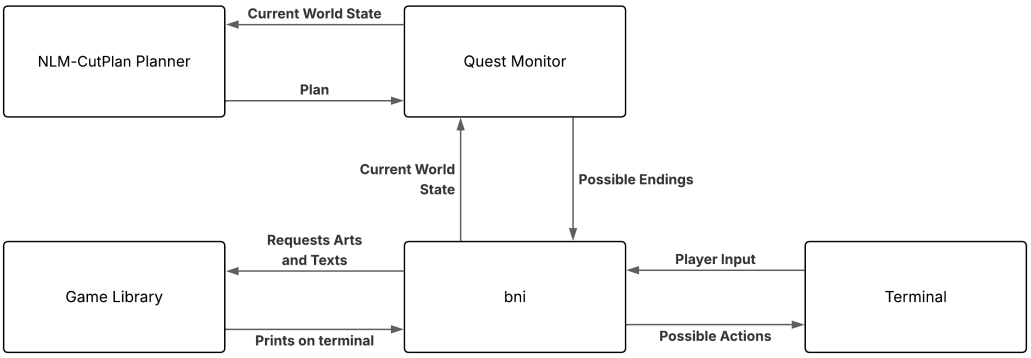


Figure 2. Block Diagram of Gobbi's Goblins Architecture.

4. Game Architecture

The game's architecture is composed of five main modules: the Terminal, `bni`, Game Library, Quest Monitor, and the planner. Together, these modules form the core structure of the system, each fulfilling a specific role to ensure the game operates as a cohesive and fully functional experience.

Figure 2 presents the architecture of the Gobbi's Goblins game through a block diagram, illustrating its internal modules. The following subsections detail each of these modules, explaining their functionality as well as the reason behind their interactions.

4.1. Terminal

This module serves as the primary interface between the player and the game Gobbi's Goblins. It operates directly through the computer's command-line terminal, where all interactions with the game take place. Through the terminal, the player can type commands to perform actions, while the system responds with narrative text, arts, and feedbacks. After an action is inserted in the Terminal, it is sent to `bni` module.

4.2. bni Parser and REPL

For Gobbi's Goblins, the original bni software could not be used directly in the game, as it was designed for classical planning problems, while the game is modeled as a numerical planning problem. With this in mind, modifications were made to the source code to properly integrate bni with the game's PDDL files.

The bni parser effectively transforms the planning model into a format that the game can be compiled and interacted with. The bni REPL module on the other hand receives the player's input, validates it by checking for a corresponding action, and, if valid, executes the action within the current game state. Finally, the REPL updates the player interface, presenting the available actions at that moment, as shown in Figure 3.

Alongside the possible actions, the adapted REPL has fixed commands to show the game map, the list of possible endings (Morale System) and the rendering of ASCII art representations of the game, which are all stored in the Game Library Module. These arts include the Main Menu, Map of Ogama, arts for Gobbi, Chief Elf and King Hyde.

This allows for a dynamic and interactive gameplay loop, where every decision made by the player is grounded in the logical structure defined in the PDDL model. If

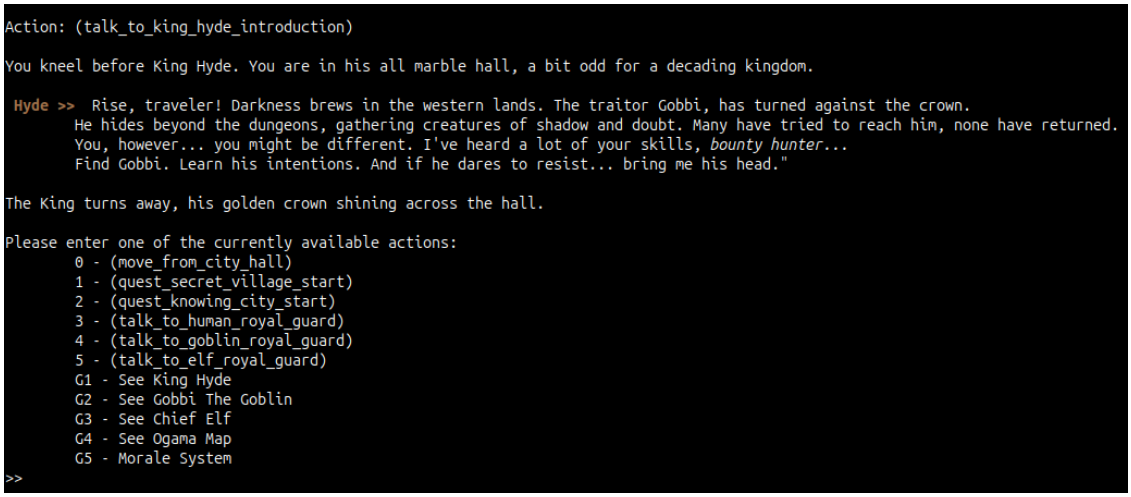


Figure 3. User interface example.

the command inserted by the user is a valid action, the Game Library is called, or, if the command is to see the Morale System of the game, then the Quest Monitor is called.

4.3. Game Library

The Game Library is divided into four main functions: `print_menu`, `repl_menu`, `print_arts`, and `print_action_text`. Unlike Soares' Quest Library architecture, it does not store new quests but rather serves as a repository of visual and narrative resources used throughout the game. These resources include text descriptions that accompany player actions and provide narrative context, and the game's ASCII art elements.

The `print_menu` function displays the main ASCII art menu, while `repl_menu` controls its behavior based on player input, handling options such as instructions, synopsis, game start, or exit. Additionally, the `print_arts` function stores and displays ASCII representations of the game world, including characters and the map of Ogama, which can be accessed at any point during gameplay.

Finally, the `print_action_text` function handles the in-game narrative feedback system. Each time the player performs an action, a corresponding descriptive text is printed to enrich the storytelling. To achieve this, a data structure similar to a JSON object (called `action_text`) was created. This structure stores pairs of action names and associated texts, functioning like a key–value mapping. The function iterates through this array of `action_text` entries until it finds the one matching the executed action, printing the corresponding text on the terminal.

4.4. Quest Monitor

The Quest Monitor is the heart of this project. Since the primary objective of this work is to demonstrate a method for implementing planning within a game system for the manipulation of narratives, the Quest Monitor is what transforms the system from a game developed in PDDL and C into a meaningful architectural contribution.

For this module to operate correctly, a well-defined pipeline is required to execute its main function: verifying which game endings are still achievable and determining the

minimum number of steps needed to reach each of them. The primary objective of the Quest Monitor in Gobbi's Goblins game is to provide the player with information about the endings, which of them are still possible and in how many steps/actions can it be reached.

The step count tells the player which specific ending they already care about is closer or further away given their current choices. The player has been building alignment with one faction through their decisions, and the step count reflects the narrative consequences of those choices in real time. The number of steps serves as a feedback signal for deliberate narrative navigation. This module is only active when invoked by the player during gameplay, when the game's Morale System is displayed (as shown in Figure 1b).

The Quest Monitor process begins by retrieving the current world state from the bni module. To evaluate how to reach each ending, it is necessary to get all active predicates and the value of each PDDL function in that point in time. Next, the system must generate the corresponding problem files, containing the current world state and a corresponding ending. This means that the Quest Monitor executes four planning jobs, as there are four possible kings in the game.

Finally, the Quest Monitor updates the game's Morale System, allowing the player to track the evolving state of their campaign and understand how their actions influence the narrative outcome.

4.5. NLM-CutPlan Planner

The final module of the system to be described is the planner. This component is invoked by the Quest Monitor to generate plans for each possible ending of the game. The planner chosen for this project is NLM-CutPlan, specifically built to handle numeric planning [Kuroiwa et al. 2023, Shleyfman et al. 2023]. A more simplified version of NLM-CutPlan was used, referred to by the authors as Numerical Fast Downward (IPC 2023 version). The main difference between them is that the NLM-CutPlan relies on the IBM CPLEX Optimization Solver[2], a proprietary software package.

5. PDDL Representation

Developing a game directly in PDDL is, in many ways, an unconventional endeavor. PDDL was originally created for automated planning and artificial intelligence research, meant to describe logical state transitions and action effects [Haslum et al. 2019]. Using it as the foundation of a playable game challenges its traditional purpose and highlights the expressive power of planning languages.

In this context, game mechanics, quests, and player interactions are modeled as logical constructs, and narrative progression emerges from state transitions and goal satisfaction rather than predefined scripts. As a result, developing games in PDDL becomes both experimental and conceptually rich, since spatial and temporal relationships must be explicitly defined within the constraints of planning formalisms.

[2]Available at: https://www.ibm.com/br-pt/products/ilog-cplex-optimization-studio

```
(:predicates
  ;Location
  (inCityHall) (inElfLair)
  (inNorthDungeon) (inSouthDungeon)
  (inGoblinsDen) (nowhere)

  ;Races
  (isHuman) (isElf) (isGoblin)

  ;City Hall Predicates
  (playerKing) (gobbiKing) (hydeKing)
  (elfKing) (hydeRuns)

  ;Mission Predicates
  (questActive) (questKC)
  (questKCDone) (KC1) (KC2)
)

(:functions (total-cost) (goblinScore)
  (humanScore) (elfScore))
```

(a) Game domain in PDDL.

```
(:action move-to-city-hall
    :parameters ()
    :precondition (and (not (gameOver))
                  (not (inCityHall))
                  (nowhere))
    :effect (and (inCityHall)
            (not (nowhere))
            (increase (total-cost) 0))
)


(:action move-from-city-hall
    :parameters ()
    :precondition (and (not (gameOver))
                  (inCityHall)
                  (not (nowhere)))
    :effect (and (not (inCityHall))
            (nowhere)
            (increase (total-cost) 0))
)
```

(b) Movement actions.

Figure 4. PDDL snippets used in the game: (a) domain definition; (b) movement actions.

5.1. Spatial Representation.

To represent spatial relationships, the game defines specific predicates that encode the player's current location on the map. Predicates such as `(inCityHall)` and `(inGoblinsDen)`, shown in Figure 4a, indicate that the player is present in a particular location. As for the predicate `(nowhere)`, it represents a transitional state in which the player is not located in any specific area.

This design restricts the player to two spatial conditions: being in a location or being in transit. When `(nowhere)` becomes true, triggered when the player leaves a location, the only admissible actions are movement actions to another location. This constraint arises because every action in the game is associated with a spatial marker specifying where it can be executed. As a result, spatial consistency is enforced at the planning level, as illustrated in Figure 4b.

5.2. Temporal Representation

Each quest follows a linear structure, with branching occurring at its ending, where players choose between antagonistic actions that affect their scores. Temporal progression within a quest is modeled through predicates that combine an abbreviation of the quest name with a numeric suffix indicating its current step.

For instance, `(KC2)` denotes that the second stage of the quest has been completed, while predicates such as `(questKCDone)` prevent completed quests from being repeated. Additionally, the predicate `(questActive)` ensures that only one quest is active at a time, becoming true at the start of a quest and false upon its completion.

Although most quests are structurally independent, their outcomes may influence one another. In particular, quest completion and intermediate actions can affect the player's score, which in turn impacts the possible outcomes of other quests.

5.3. Morale System and Problem Files.

A major highlight of the domain design is the morale system, which drives the generation of multiple possible endings. In this system, four PDDL functions were defined, each initialized to zero at the start of the game: `(humanScore)`, `(goblinScore)`, `(elfScore)`, and `(total-cost)`.

Certain actions, especially the final steps of quests, apply numeric effects that increase or decrease these scores. By the end of the game, the set of possible endings is determined by the quests completed and the final score values. These scores have a direct impact on narrative outcomes.

As for the game's problem file, two distinct versions are required: one to be used by the bni parser, and another for the Morale System. The first version, used by the bni module to produce the C version of the game domain, has a straightforward purpose: setting the initial world state of the game. This includes setting all scores to zero and placing the player at the starting location, the City Hall.

In this version, the `:goal` block is intentionally minimal, containing only the predicate `(gameOver)`, present in any possible ending of the game. This predicate becomes true only when one of the possible endings is reached. Once this occurs, the bni module detects that the goal state has been satisfied and terminates the program accordingly.

The problem file used by the Quest Monitor to build the Morale System differs from the parser version in two key areas: the `:init` block and the `:goal` block. Since the Quest Monitor is invoked while the game is in progress, the initial world state defined in `:init` must accurately reflect the current state of the game. Therefore, the predicates and score values correspond to the player's progress, marking which quests have been completed and which predicates are currently active.

As for the goal definition, the Quest Monitor performs multiple planning calls, one for each possible ending. In each of these calls, the `:goal` block contains two predicates: `(gameOver)` and the specific predicate representing the targeted ending. This setup allows the system to verify, for every possible narrative conclusion, whether it can be achieved from the current world state, and, if so, in how many planning steps.

6. Comparison with related work

As discussed previously, prior work has made important contributions to planning-based dynamic narratives. This section compares the present work with these approaches to clarify its contributions. Table 1 summarizes key characteristics of the works by Li and Riedl (2010), Soares et al. (2014), and this work, highlighting their main similarities and differences.

A primary distinction between the proposed work and that of Li and Riedl lies in the stage at which planning is applied within the game lifecycle. While Li and Riedl perform narrative adaptation prior to gameplay, modifying story structures offline based on player models, both the present work and Soares et al. employ planning during runtime. This enables player actions to directly influence the unfolding narrative in real time, resulting in a more immediate and responsive interactive experience.

Table 1. Comparative Matrix with key differences between approaches

	Li & Riedl	Soares	This Work
Planning Paradigm	DPOP	HTN	Numerical Planning
Individualized Personalization	✓	✗	✗
Planning-Based Game	✗	✗	✓
Real-time Execution	✗	✓	✓
Narrative Observability	✗	✗	✓

The key difference between this work and Soares et al. lies in how planning is integrated into the game architecture. While Soares et al. uses planning as a subsystem for quest generation and adaptation, this work adopts a planning-centric approach in which the entire game is built upon planning formalisms. Planning not only orchestrates the narrative flow but also defines the game's world model and rules, meaning that all possible interactions are constrained by the PDDL domain.

Additionally, this work introduces real-time narrative observability through the Morale System, allowing players to monitor story progression and receive immediate feedback on their path toward different endings. By exposing the narrative state space, the system aims to promote meta-awareness, defined as the ability to reflect upon and monitor one's own cognitive processes [Flavell 1979]. In this context, players become aware of both the narrative and its underlying structure, enabling a new form of engagement in which they can deliberately reason about and steer the story toward desired outcomes.

7. Conclusion and Future Work

In this work, we presented an approach for integrating automated planning into digital game systems, with a focus on the verification of narrative structures through the use of PDDL. This project demonstrates how planning can serve as a reasoning layer within a game's architecture, capable of analyzing the current world state, verifying narrative consistency, and identifying possible endings dynamically. It opens new possibilities for developing games where the narrative is not only scripted but also can emerge from the player's decisions.

There is still significant room for improvement in this project, especially when considering its foundation on Soares's original architecture. The most obvious next step is the evolution of the narrative monitoring process from verifying only the game's endings to tracking the entire chronological progression of the story. This enhancement would require transitioning from purely numeric planning to hierarchical numeric planning, allowing for a more structured representation of quests and subquests throughout gameplay.

Another important direction is to further validate the proposed architecture by moving beyond a terminal-based game. Implementing this system within modern game engines and more commonly used industry frameworks would not only test its scalability and integration potential but also demonstrate how automated planning and narrative verification can coexist naturally within contemporary game development environments.

References

Arora, K. (2023). The gaming industry: A behemoth with unprecedented global reach. Available at: https://www.forbes.com/councils/forbesagencycouncil/2023/11/17/the-gaming-industry-a-behemoth-with-unprecedented-global-reach/. Accessed: 2025-07-02.

Bartle, R. (1996). Hearts, clubs, diamonds, spades: Players who suit muds. Available at: https://mud.co.uk/richard/hcds.htm. Accessed: 2025-07-02.

Flavell, J. (1979). Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *American Psychologist*, 34:906–911.

Haslum, P., Lipovetzky, N., e Magazzeni, D. (2019). *An Introduction to the Planning Domain Definition Language*. Morgan & Claypool Publishers.

Kuroiwa, R., Shleyfman, A., e Beck, J. C. (2023). Extracting and exploiting bounds of numeric variables for optimal linear numeric planning. In *Proc. ECAI*.

Li, B. e Riedl, M. O. (2010). Planning for individualized experiences with quest-centric game adaptation. In *Proceedings of the ICAPS 2010 Workshop on Planning in Games*, Toronto, Canada.

Microsoft Learn (2021). Ai overview – halo 2. Available at: https://learn.microsoft.com/en-us/halo-master-chief-collection/h2/ai/aioverview. Accessed: 28 jun. 2025.

Prince, G. (2003). *A Dictionary of Narratology*. University of Nebraska Press, Lincoln, revised edition.

Ribeiro, B., Penha, I., e Ribas, B. (2025). bni: A pddl to c compiler with integrated repl for interactive testing. In *Proceedings of the 22nd National Meeting on Artificial and Computational Intelligence*, pages 1785–1796, Porto Alegre, RS, Brasil. SBC.

Schell, J. (2014). *The Art of Game Design: A Book of Lenses*. A. K. Peters, Ltd., USA, 2nd edition.

Shleyfman, A., Kuroiwa, R., e Beck, J. C. (2023). Symmetry detection and breaking in cost-optimal numeric planning. In *Proc. ICAPS*, pages 393–401.

Soares de Lima, E., Feijo, B., e Furtado, A. L. (2014). Hierarchical generation of dynamic and nondeterministic quests in games. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology*, ACE '14, New York, NY, USA. Association for Computing Machinery.

Soares de Lima, E., Feijo, B., e Furtado, A. L. (2016). Player behavior and personality modeling for interactive storytelling in games. *Entertainment Computing*, 28:32–48.