Next Gen NPCs

Ali Jradi Mohammad Jradi

July 8, 2025

Abstract

The ability to simulate believable human behavior has always been a foundational goal in artificial intelligence research, with wide-ranging applications in immersive environments, autonomous systems, social simulations, and multi-agent learning. Believable human proxies, agents that exhibit lifelike routines, memory-driven decision-making, and coherent social behavior, can serve as powerful tools for testing, interaction, and training when real human involvement is impractical or costly. Recent research, particularly following the advent of large language models (LLMs), has made significant strides in modeling such behavior by combining natural language processing with memory architectures, planning systems, and reflection mechanisms. These generative agents have demonstrated emergent behaviors that resemble human cognition and social interactions. However, a key limitation remains: While these agents are semantically coherent and behaviorally rich, they often lack real-time responsiveness. Most LLM-driven systems operate in batch or periodic modes, relying on serialized reasoning and delayed reflection that inhibit their ability to engage fluidly in dynamic, time-sensitive environments. This temporal disconnect reduces their believability in interactive settings where timing, interruption handling, and immediate adaptation are crucial to the perception of intelligence and social presence. This paper surveys the state of the art in the design of generative agents, highlights key architectural patterns that enhance believability, and proposes a novel extension to the framework introduced by Park et al. [13] that enables agents to engage in real-time decision making.

1 Introduction

From early simulations to modern AI research, the ambition to create artificial agents that behave like humans has been a long-standing challenge. Researchers have envisioned computational agents capable of acting autonomously, adapting to their environment, and interacting in socially plausible ways for over four decades. These visions span domains such as cognitive modeling [11], usability testing [4], virtual environments [3], ubiquitous computing [16], and entertainment media such as sandbox games and open-world simulations [8, 9]. At the core of this vision is the concept of believability: the ability of an agent to exhibit coherent, context-aware, and temporally consistent behavior that evokes a sense of life and intentionality.

Recent advances in large language models (LLMs) have significantly accelerated progress toward this goal. Systems such as those presented by Park et al. [13] introduce generative agents, autonomous characters powered by LLMs that simulate daily routines, form social bonds, recall past experiences, and make future plans through natural language interfaces. By extending LLMs with episodic memory, reflection mechanisms, and behavior planning, these systems have demonstrated the emergence of individual and collective behaviors that closely mimic human social dynamics. Other works have explored similar territory, from training embodied dialogue agents with realistic interaction patterns to simulating diverse social behaviors using multi-agent LLM systems.

Despite these advances, a crucial limitation persists. Most LLM-based agents operate in non-interactive or turn-based contexts, relying on synchronous calls to generate behavior in discrete steps. This inherently offline mode of thinking makes it difficult for agents to respond fluidly to environmental changes, user input, or the actions of other agents. In contrast, humans operate in real time, constantly perceiving, reacting, and adjusting as things unfold. The lack of such responsiveness in generative agents reduces their believability and limits their usefulness in interactive scenarios, such as games, collaborative interfaces, or simulation-based training environments.

Believability is not just about what an agent says or does, but also when and how it acts. Timing, interruption handling, and responsiveness are all vital to the perception of social intelligence. Without real-time reasoning, even the most semantically sophisticated agents may behave in ways that appear slow, disjointed, or robotic, breaking the illusion of life these systems aim to achieve.

In this paper, we explore a novel solution situated within a real-time sandbox multiplayer game environment, extending the framework introduced by Park et al. [13]. While their generative agents demonstrate rich social behavior through memory, reflection, and planning, they remain limited by the absence of real-time responsiveness. To address this, we propose a new architecture that integrates Goal-Oriented Action Planning (GOAP), a planning framework widely used in games for dynamic behavior selection, with the cognitive capabilities of generative agents. This hybrid architecture enables agents to reason and act asynchronously, allow-

ing for fluid, moment-to-moment decision making.

Our approach equips agents not only with the ability to recall past experiences and engage in natural conversations, but also with the capacity to react immediately to dynamic environmental stimuli and player interactions. These agents are not confined to scripted dialogues or predefined roles. Instead, they exist within the world as autonomous entities capable of perceiving changes, formulating goals, and executing adaptive behaviors in real time, similar to human players. By embedding GOAP within the generative agent framework, we bridge the gap between high-level cognition and responsive game AI, creating agents suitable for fast-paced, interactive environments where timing and reactivity are essential for maintaining believability and immersion.

This paper presents the design, implementation, and evaluation of this architecture in a multiplayer sandbox setting. We show how such agents can meaningfully participate in both emergent gameplay and social interactions. Through this work, we aim to move closer to truly interactive, lifelike AI characters capable of engaging players in real time.

2 Related Work

2.1 Traditional Behavior Models in Real-Time Game AI

2.1.1 Finite State Machines (FSMs)

Finite State Machines (FSMs) are among the earliest and most widely used tools in game AI. An FSM models an agent as a set of discrete states (e.g., "Patrol", "Alert", "Attack"), with transitions triggered by sensory events or internal conditions [10]. In practice, nearly every early game character AI has employed some form of FSM for behavior control. At each tick, the active state executes its logic (often through 'onUpdate' callbacks) and checks transitions to other states [7].

This simple, procedural paradigm makes FSMs easy to understand and implement [10]. However, FSMs suffer from **state explosion** and **rigidity**. As the number of states n grows, the number of possible transitions can grow on the order of $\mathcal{O}(n^2)$, quickly yielding a monolithic transition network that is hard to extend or debug [7]. Complex behaviors require many states and transitions, and adding new behaviors often entails inserting multiple new states and connecting them manually.

In short, FSMs can become unwieldy and inflexible for large NPC rule sets, and tend to produce repetitive, predictable actions that break player immersion. As one developer notes: "An FSM tells an AI exactly how to behave in every situation," making it fully procedural and often predictable [7]. Historically, FSMs remain common for simple agents or low-budget projects where authorial control is paramount, but designers must manually specify all behaviors and transitions without abstraction or reuse mechanisms.

2.1.2 Behavior Trees (BT)

Behavior trees were introduced in games in the mid-2000s (notably in *Halo 2*) to improve on FSMs' shortcomings [12]. A BT is a hierarchical decision structure rooted at a control node, with action or condition leaves at the bottom. Internal nodes (such as Sequence or Selector nodes) repeatedly "tick" their children: a Sequence executes children in order until one fails, while a Selector chooses the first succeeding child. Each leaf node defines a precondition and an action. During each update tick, the tree is traversed top-down, evaluating conditions and executing actions along one branch that leads to success [5]. In effect, behavior trees encode complex behaviors by composing small tasks into larger ones – for example, a "Fighting" parent behavior might have children "SwingSword" and "FireArrow," each with their own conditions.

Behavior trees became popular because they improve modularity and reuse. Unlike FSMs where transition logic is scattered across states, BTs collect control flow in a clear tree structure, with each subtree handling a logical subtask [5]. This hierarchical organization makes it easier for designers to add or remove behaviors: new subtrees can be slotted in without rewriting unrelated parts of the tree. Consequently, BTs scale better than flat FSMs as complexity grows. In practice, BTs dominate modern game AI (anecdotally, BTs are "arguably the most commonly applied technique in the AAA video game industry" since *Halo 2*) [12]. Major game engines often include BT editors (e.g., Unreal's Behavior Tree system), and many AAA NPCs (guards, companions, etc.) are built with BTs.

Nevertheless, BTs have limitations. Although more structured than FSMs, a behavior tree can still become large and hard to manage if not carefully designed. The default control nodes (especially selectors with fixed priority) produce deterministic decision sequences: an agent will always try higher-priority branches first. Without additional logic (e.g., utility nodes or randomness), this can lead to somewhat predictable behavior. Moreover, BTs are fundamentally reactive: they lack any internal model of future consequences beyond the current tick, so they share FSMs' difficulty with complex, long-term planning [5]. In scenarios requiring global strategy or learning, large static trees can become brittle. Some authors note that even with a BT, adding many behaviors can be challenging – each new action or condition often requires grafting a new leaf into the tree and re-authoring parts of its structure [2]. In summary, BTs are more modular and maintainable than FSMs [5], but still rely on hand-authored scripts and can exhibit limited flexibility or believability (agents often repeat the same sequence of tasks unless the tree is deliberately randomized or extended with heuristics).

2.1.3 Goal-Oriented Action Planning (GOAP)

Goal-oriented action planning (often known by the acronym GOAP or by terms like "Planner") represents a further step toward flexible AI. In GOAP, each agent has

a set of possible goals (desired world states) and a library of actions, each with preconditions and effects (and often a cost). A planning algorithm then automatically searches for a sequence of actions that transforms the current world state into one of the goals. In the classic implementation (as used in *F.E.A.R.*), FSMs were reduced to only a few high-level states (e.g., idle, combat ready), and the bulk of the behavior logic was handled by the planner [12]. The GOAP system separates what the agent wants (the goal) from how to achieve it, much like classical STRIPS planners [2].

In practice, a GOAP agent has a "goal manager" that picks an appropriate goal (e.g., "FindCover", "AttackPlayer"), and a planning component that builds a plan to satisfy that goal [12]. The planner may use a graph search (e.g., A* over an action graph) to assemble a viable sequence of actions. Because goals and actions are decoupled, the same goals can be achieved with different action sets for different characters (for instance, a guard and an assassin share the goal "EliminateEnemy" but have different weapons/actions) [2]. This decoupling also makes it easier to add new behaviors: designers add new actions and link them to goals rather than editing a giant state graph. GOAP was famously used in *F.E.A.R.* to create very dynamic enemy behavior: guards would automatically chain together actions like taking cover, reloading, throwing grenades, and flanking without those combinations being explicitly scripted [12].

However, GOAP has trade-offs. Planning in real time can be computationally expensive, especially as the number of actions and goals grows. Naïve planners may stall or behave slowly if many agents replan simultaneously, so game implementations must use heuristics or limit plan depth [12]. Moreover, purely deliberative planners can struggle with highly dynamic environments: if a plan is in execution and the world changes (e.g., the player moves), the agent may need to replan from scratch, causing pauses or irrational behavior. As a result, GOAP systems are often hybridized with reactive layers to handle immediate threats (for example, a BT or simple override may interrupt a plan if the agent is shot) [2].

2.1.4 Summary

In summary, FSMs, behavior trees, and GOAP represent a progression from simple scripted agents to more flexible planners. FSMs are easy to implement but do not scale to complex behavior without unwieldy state explosion and yield very predictable outcomes [2]. Behavior trees restore modularity and hierarchical structure [5], easing the design of medium-complexity AI, but still depend on static scripts and offer only limited adaptivity. GOAP allows agents to generate novel action sequences toward goals, improving flexibility and reuse of behaviors [12], yet incurs planning cost and can lead to coordination issues that harm believability [1]. These limitations of classical models set the stage for exploring newer approaches (such as LLM-driven generative agents with real-time responsiveness) that promise more scalable, flexible, and lifelike NPC behavior.

2.2 Generative AI in Simulating Human Agents

Recent work has begun to leverage large language models (LLMs) to create virtual agents that exhibit rich, human-like behavior. For example, Park et al. [13] introduced generative agents, software agents whose behavior is driven by an LLM augmented with explicit memory and planning modules. These agents "wake up, cook breakfast… form opinions, notice each other, and initiate conversations; they remember and reflect on days past as they plan the next day." Park et al. report that a small town of 25 such agents can autonomously generate emergent social activities (e.g., planning and attending a party) that appear coherent and plausible.

Other recent systems also integrate LLMs into interactive agents. For instance, Voyager [15] uses an LLM to power a Minecraft agent that continually explores, learns, and composes new skills. Large-scale simulators such as AgentSociety [14] have also been developed to populate entire virtual societies with LLM-driven individuals. In all these cases, the LLM imparts the agent with natural language memory, reasoning, and dialogue capabilities, producing behavior that users often judge as more believable and varied than rule-based or scripted NPCs.

Despite these advances in believability, existing LLM-agent frameworks lack true real-time reasoning. Because each decision or dialogue requires an LLM inference, agents operate on discrete time scales rather than continuously. For example, Park et al.'s implementation deliberately runs the simulation slower than wall clock time, roughly "one second real time is one minute game time" to allow LLM-based planning to occur [13]. Kaiya et al. [6] explicitly note that "achieving real-time interactions with humans at a low computational cost remains challenging." Even systems like Lyfe Agents, which introduce cost-saving heuristics such as hierarchical planning and memory summarization, still rely on asynchronous LLM queries to formulate each decision [6].

In practice, this means that LLM-driven agents cannot respond at high frame rates or in continuous control loops. They are better suited for task-level planning and dialogue. In short, while LLM-based generative agents excel at long-range planning and social reasoning [14], they do not provide the low-latency, reactive decision-making required for fully real-time control.

These limitations motivate hybrid approaches that combine LLM-driven cognition with fast reactive planning (such as GOAP) to achieve both authenticity and real-time responsiveness.

2.3 Generative Agents: Interactive Simulacra of Human Behavior

Our primary inspiration for this project stems from the paper titled *Generative Agents: Interactive Simulacra of Human Behaviour* [13]. This work represents one

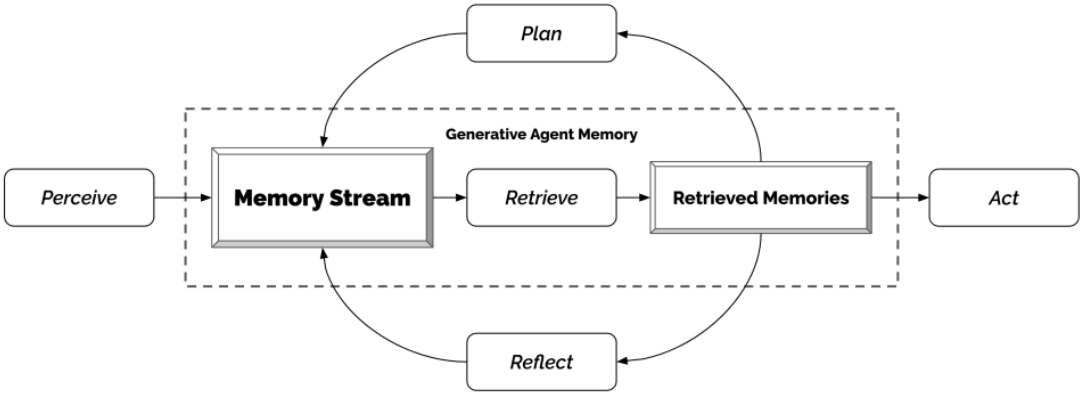

Figure 1: Generative Agent Architecture

of the first significant efforts to simulate human-like behaviour in NPCs through the use of large language models (LLMs). The core objective of the study was to construct a sandbox environment populated with multiple AI agents - each controlled by OpenAI's ChatGPT - that could form memories, engage in natural conversations, interact with the world and execute structured plans based on daily schedules and events.

As stated in the original paper:

> "In this paper, we introduce generative agents—agents that draw on generative models to simulate believable human behaviour—and demonstrate that they produce believable simulacra of both individual and emergent group behaviour." [13]

These generative agents were capable of reasoning about themselves, other agents, and their environment. They could formulate plans, adjust them in response to changes, and interact with users or each other through natural language. This level of emergent behaviour highlighted a new frontier in the pursuit of lifelike digital characters.

To achieve this, the researchers developed a novel agent architecture that integrates memory, reflection, and planning in a closed loop. The architecture is structured around three main components:

- **Memory Stream:** A long-term memory module that logs the agent's experiences in natural language. It uses a retrieval model that prioritizes memories based on relevance, recency, and importance to determine which past events should influence current behaviour.
- **Reflection Mechanism:** This component synthesizes retrieved memories into abstract inferences and beliefs over time. These reflections help agents form conclusions about themselves and others, enabling deeper self-awareness and socially responsive behaviour.
- **Planning Module:** Using reflections and environmental context, the planning module generates high-level daily goals and recursively decomposes them into actionable steps. The resulting behaviours are then logged back into memory, reinforcing the agent's evolving narrative.

3 Proposed Framework Architecture

This section outlines the architecture of our proposed framework for next-generation NPCs, which integrates cognitive AI capabilities with real-time planning and execution mechanisms. Building on the foundational work of Generative Agents [13], our model extends their architecture to support real-time interaction and action in dynamic multiplayer environments.

3.1 Overview

The core innovation of our architecture lies in the integration of three complementary systems:

- A **Perception Module**, which contains multiple sensors that allow agents to gather information about their environment and update the NPC's instance of the world state.
- A **Large Language Model (LLM)-driven Cognitive Module**, adapted from the Generative Agents framework [13], responsible for high-level reasoning, reflection, and goal generation based on memory, social context, and perception.
- A **Goal-Oriented Action Planning (GOAP) Module**, which handles real-time action selection and execution within a dynamic game world.

By decoupling deliberative cognition from reactive behavior, this architecture enables NPCs to exhibit both long-term believability and short-term responsiveness,

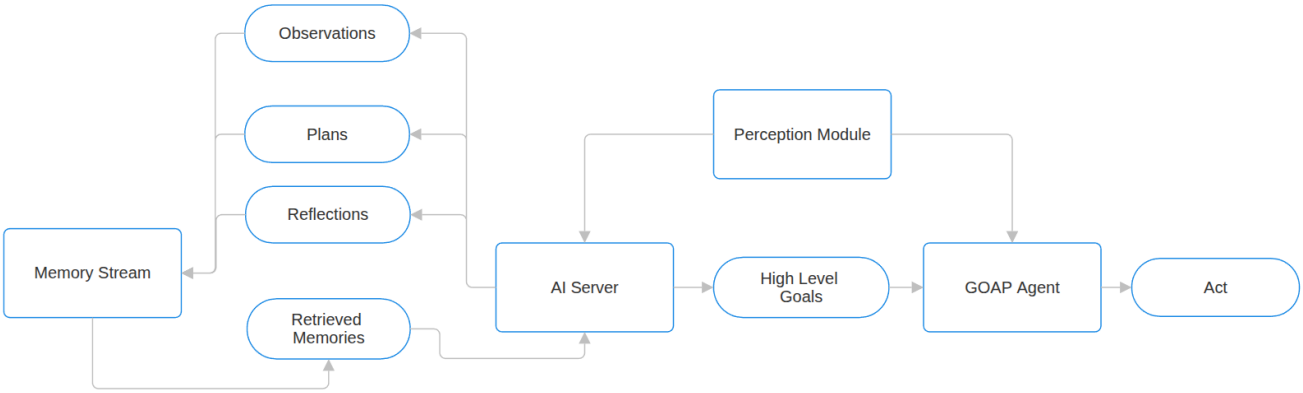

Figure 2: Proposed Framework Architecture

an essential requirement for real-time multiplayer or simulation-heavy environments. This dual-system approach mirrors human cognition, where spontaneous low-level actions (e.g., walking, dodging) are handled unconsciously, while higher-level planning and reflection (e.g., strategizing, conversing) require conscious thought. Similarly, our NPCs can immediately react to environmental changes via the GOAP layer, while their broader behavioral arc is shaped by reflective, goal-driven cognition via the LLM.

3.2 Component Breakdown

3.2.1 Perception Module

The Perception Module functions as the sensory system of the agent, continuously monitoring the world and maintaining an updated belief state. This module integrates various sensors that detect visual entities, auditory stimuli, spatial proximity, dialogue input, and internal status changes.

Its core responsibilities include:

- Populating and updating the agent's belief state, representing its subjective understanding of the world.
- Triggering immediate reactive behavior (e.g., dodging, responding to greetings).
- Informing the GOAP planner with the latest world state.
- Feeding structured observations into memory, enabling reflection and goal generation by the LLM.

Operating at high frequency with low latency, the Perception Module provides the cognitive and planning systems with an accurate and timely view of the world.

3.2.2 Cognitive Layer (LLM Module)

The LLM handles high-level cognitive processes including:

- Generating new goals based on internal motivations and external observations.
- Performing reflections by synthesizing memory into actionable insights.
- Engaging in social interactions and dialogue with other agents or players.

This component operates asynchronously and at a lower frequency, typically once every few seconds, to simulate deliberation and introspection. The LLM outputs prioritized goals in natural language, which are structured and injected into the GOAP system for execution.

3.2.3 GOAP Planning and Execution Layer

The GOAP module is responsible for physical action and decision-making in real time. It includes:

- A world state representation based on data received from the perception module.
- A set of atomic and composite actions, each defined with preconditions and effects.
- A dynamic planner that generates action sequences to satisfy goals.
- A real-time executor that dispatches actions at a fixed tick rate (e.g., 10–20Hz).

This layer enables the agent to adapt rapidly to new conditions (e.g., obstacles, threats), ensuring continuous and realistic behavior.

3.2.4 Goal Injection and Synchronization

To bridge the asynchronous LLM loop and the synchronous GOAP loop, a goal injection interface manages:

- Translating natural language goals into structured representations (e.g., `{"be_at_location": "townhall"}`).
- Encoding priority as a score from 1 to 10.
- Reconciling conflicts between current plans and new goals.

In urgent scenarios (e.g., combat), the GOAP layer can dynamically spawn local goals, while the LLM later reevaluates its strategy to accommodate the new context.

3.3 Real-Time Interaction Loop

The complete system operates in two asynchronous cycles:

LLM Loop (Slow Frequency):

- Reads from episodic and semantic memory.
- Uses retrieved memories and observations to create high level plans.
- Performs reflections to generate new goals.
- Injects goals into the GOAP layer as needed.

GOAP Loop (High Frequency):

- Reads the world state and current goals.
- Plans and executes actions accordingly.
- Reacts to environmental stimuli or interruptions.
- Feeds state updates back into memory.

This separation of concerns enables agents that are both responsive and believable, capable of dynamic real-time behavior and coherent long-term development.

4 Environment Implementation

4.1 Game World and Environment Selection

To evaluate our proposed agent architecture in a realistic, dynamic, and socially rich setting, we developed a custom multiplayer online role-playing game (MMORPG) as our primary testing environment. The decision to design an MMO was deliberate and grounded in the key attributes that make this genre especially suitable for our AI agents research. MMOs naturally support high-scale social interactions, where players and NPCs coexist in shared spaces and continuously interact. This creates a dense social fabric, allowing AI agents to engage in meaningful dialogue, form relationships, and coordinate or compete with both humans and other NPCs. Such an environment is essential to demonstrate the viability of our agents in emergent social contexts. Moreover, MMOs feature persistent worlds, where actions have lasting consequences and characters evolve over time. This persistence allows agents to form and rely on long-term memory, perform goal reflection, and demonstrate temporal continuity in their behavior, traits central to our cognitive framework. Finally, creating an MMO allowed us to test the capabilities of our AI agents across a range of unpredictable scenarios such as dynamic combat encounters, group events encouraging cooperation between agents and players such as raids, and spontaneous player alliances.

4.2 Used Technologies

To support the complex requirements of our real-time MMO environment, we employed a modern and modular technology stack composed of both web-based and AI-focused tools. The selection of technologies was guided by goals of scalability, maintainability, and seamless integration across various components.

4.2.1 Frontend

The game client frontend was developed using **React**, a declarative JavaScript library for building user interfaces using a component-based architecture. For styling, we used **Tailwind CSS**, a utility-first CSS framework, alongside **Shadcn/UI**, a component library for accessible and consistent design.

This stack enabled rapid development of responsive user interfaces, with gameplay features such as inventory management, interaction panels, and character creation integrated directly into the React component hierarchy.

4.2.2 Game Engine

We utilized **Phaser**, a lightweight 2D game engine for the web, to build the interactive game environment. Phaser provided fine-grained control over rendering, asset management, and input handling, which was essential for creating fluid, responsive interactions in a real-time multiplayer context.

4.2.3 Backend Infrastructure

The backend game server was implemented using **Express.js**, a minimal and flexible Node.js web application framework. Express handled key responsibilities such as API routing, user authentication, and database interaction. It served as the communication bridge between the game client, the multiplayer framework, and the AI subsystems.

4.2.4 Multiplayer Framework

We integrated **Colyseus**, a Node.js-based multiplayer framework, to manage real-time synchronization between clients and the game server. Colyseus provided support for room management, authoritative state control, and low-latency communication, enabling responsive and coordinated gameplay experiences for multiple users and NPCs.

4.2.5 AI Server

A separate **Python-based AI server** was developed using **FastAPI**, chosen for its high performance and ease of integration with machine learning libraries. The AI server was responsible for:

- LLM-based inference (e.g., goal generation, memory reflection)
- Dialogue and decision response formulation
- Long-term and short-term memory processing

Figure 3: Screenshot of the game environment

Communication with the main game server was handled asynchronously via WebSocket, allowing agents to interact in near real-time with their environment and other entities.

4.2.6 Data Storage

We used **MongoDB** as the primary database to store structured and semi-structured data, including:

- Player profiles and inventory
- NPC memory logs and state representations
- Event histories and world state snapshots

MongoDB's flexible document model aligned well with the dynamic and evolving nature of AI-generated content and game data.

4.2.7 Vector Indexing

To support semantic memory retrieval for generative agents, we integrated **FAISS** (Facebook AI Similarity Search) into the AI server. FAISS enabled high-performance vector similarity searches, allowing agents to:

- Retrieve relevant past conversations or experiences
- Identify contextual similarities for decision-making
- Perform memory-based reflection and planning

Embedding vectors were computed from agent memory entries and indexed for fast access, ensuring that the agents' behavior remained coherent and contextually grounded.

4.3 Supporting Systems

To create a rich and interactive environment that supports realistic agent behavior, we implemented several foundational systems within our in-game world:

4.3.1 Physics and Movement

A custom physics system governs entity movement and collision detection. It supports smooth and realistic movement with acceleration and deceleration, gravity, projectile mechanics, and jumping dynamics. These features provide a consistent and predictable physical world that both players and NPCs must navigate.

4.3.2 Combat System

Agents engage in combat using the same system as players, supporting both ranged and melee attacks, health tracking, hit resolution, and in-game skills and spells. This shared combat system enables dynamic, real-time encounters where NPCs can meaningfully participate and adapt based on context.

4.3.3 Class System

Both players and agents can select from a wide range of classes (e.g., warrior, mage), each with unique roles and playstyles. Roles include offensive (damage dealer), defensive (tank), and support (healer). Class identity affects behavior and strategy, enabling differentiated agent personalities and combat styles.

4.3.4 Character Customization

Visual customization allows for distinct and personalized characters for both players and agents. Although it has no direct mechanical effect, it enhances immersion and supports more natural interactions, agents can reference visual traits (e.g., armor, guild emblem) during conversations or recognize known entities based on appearance.

4.3.5 Status Effects and Crowd Control

This system introduces temporary effects such as stuns, slows, buffs, and debuffs, which can alter an entity's mobility or capabilities. NPC agents reason about these effects when planning: e.g., fleeing when stunned, cleansing debuffs, or avoiding combat when movement is impaired. The system is fully integrated with the GOAP planner for real-time adaptation.

4.3.6 Inventory System

Entities possess structured inventories for items such as weapons, potions, equipment, and quest objects. Agents can interact with their inventory in real time, equipping gear, using consumables, or managing item weight. This enables resource-aware planning (e.g., "equip sword," "use healing potion") and supports behaviors like trading and scavenging.

4.3.7 In-Game Proximity Chat

A spatialized chat system allows communication within a defined in-world radius. NPC agents use this for localized, naturalistic dialogue and overhearing conversations, which are logged to memory and used to infer social dynamics. It fosters emergent social interaction and context-aware dialogue between agents and players.

4.3.8 Pathfinding

We implemented an A*-based pathfinding system on a grid-based navigation mesh, shared by players and NPCs. Agents use this to reach destinations while avoiding hazards and blocked zones. The system supports real-time recalculations in dynamic environments, enabling flexible and goal-directed movement.

4.4 Evaluation

To evaluate the effectiveness of our proposed architecture, we conducted a series of experiments within our custom-built MMO environment. The primary goal of this evaluation is to determine whether the integration of a cognitive LLM-based layer with a real-time GOAP planner results in agents that are more believable, context-aware, and behaviorally adaptable compared to traditional rule-based agents and pure LLM-based systems.

4.4.1 Evaluation Criteria

Our evaluation focuses on the following dimensions:

- **Behavioral Adaptability:** The ability of agents to respond dynamically to changes in the environment and in social or combat contexts.
- **Social Coherence:** The presence of goal-driven, explainable behavior emerging from agent memory, planning, and reflection.
- **Responsiveness:** The latency of agent reactions to real-time events and stimuli.
- **Comparative Capability:** Performance and expressiveness compared to rule-based agents (FSM and pure GOAP).

4.4.2 Methodology

We implemented and compared the following agent configurations:

- **Rule-Based FSM Agents:** Traditional NPCs with hardcoded states and transitions (e.g., *idle* $\rightarrow$ *patrol* $\rightarrow$ *attack*).
- **GOAP-Based Agents:** NPCs using pure Goal-Oriented Action Planning without LLM integration. Used as a baseline for responsiveness and tactical reasoning.
- **LLM-Only Agents:** Agents powered solely by the LLM-based cognitive layer, capable of reflection and memory use but lacking real-time adaptation.
- **Next Gen NPCs:** Our proposed hybrid architecture, combining asynchronous cognitive planning with real-time reactive GOAP execution.

We tested these agents in various MMO scenarios, including:

- **Combat Encounters:** Multi-agent battles involving allies and enemies.
- **Social Interaction Scenes:** Dialogue between agents and players or among agents.
- **Open World Exploration:** Navigating dynamic environments with unpredictable changes.

Configuration Comparison

4.4.3 Observations in Emergent Scenarios

- **GOAP-only Agents:** These agents showed strong reactivity (e.g., *heal ally when* $HP < 50\%$) but lacked any narrative context or long-term intent behind actions. They could not reflect or adjust behavior unless externally re-triggered.
- **Next Gen NPCs:** These agents generated abstract, high-level goals (e.g., *"help the town guard"*, *"monitor the suspicious player"*) and translated them into concrete actions through GOAP (e.g., *"follow player," "talk to ally," "use buff spell"*).

Table 1: Comparison Across Agent Configurations

Configuration	Real-Time Responsiveness	Tactical Flexibility	Long-Term Memory	Social Coherence	Cognitive Abilities
FSM Agents	✓	✗	✗	✗	✗
GOAP Agents	✓	✓	✗	✗	✗
LLM Agents	✗	✗	✓	✓	✓
Next Gen NPCs	✓	✓	✓	✓	✓

- **Memory-Awareness:** NPCs using our full architecture exhibited differentiated behavior based on past interactions. For instance, they responded differently to players depending on prior dialogues or conflicts, something not possible with GOAP-only agents.

5 Future Work and Limitations

While our proposed architecture demonstrates the viability and effectiveness of combining cognitive and reactive components for lifelike agent behavior, there are several promising directions for extending and enhancing this framework in future iterations.

5.1 Agent-to-Agent Interaction

Currently, our agents interact primarily with players and passively observe other agents. A key area for extension is enabling more rich, goal-driven agent-to-agent interactions. This includes collaborative planning, the formation of relationships, shared memory constructs (e.g., gossip propagation), and coordinated group behaviors. Enabling agents to influence each other through both dialogue and shared actions would greatly enrich emergent social dynamics.

5.2 Enhanced Perception of Non-Chat Events

Although our agents can engage meaningfully through proximity chat and explicit dialogue, they currently have limited capacity to interpret non-verbal or non-chat stimuli, such as visual events or environmental cues (e.g., witnessing a theft or observing someone mining ore). Expanding the perceptual pipeline to more accurately encode and reason about in-world events would allow agents to respond more naturally and contextually to a wider range of gameplay situations.

5.3 Non-Combat Gameplay and Professions

Thus far, most evaluation scenarios have focused on combat and social interaction. Future work will involve introducing and testing non-combat gameplay systems, such as farming, fishing, crafting, and trading. These domains offer slower, simulation-driven contexts where agent autonomy and long-term goal modeling can be showcased. For instance, agents might take up a profession, maintain a daily routine, or trade goods based on local market conditions, all of which deepen immersion and illustrate sustained intentionality.

5.4 Expanding the Action Pool for Emergent Behavior

Our current action library supports a solid range of tactical and utility behaviors. However, we aim to expand the action set to enable more complex chains of behavior and emergent interactions. This includes compound actions (e.g., "gather ingredients to brew a potion"), abstract emotional or social gestures (e.g., "comfort a grieving NPC"), and context-specific routines. A broader action vocabulary will enhance the planner's expressivity and allow agents to more convincingly adapt to novel or compound goals.

By pursuing these directions, we aim to move closer to fully autonomous, believable agents that can operate meaningfully across a wide range of gameplay contexts, interact richly with both players and each other, and contribute to persistent, emergent narratives within multiplayer virtual worlds.

6 Conclusion

This paper presented a novel hybrid architecture for lifelike non-player characters (NPCs), combining the long-term planning and cognitive depth of generative agents with the real-time adaptability and responsiveness of a Goal-Oriented Action Planning (GOAP) system. Inspired by human cognitive processes, our framework separates deliberative thinking from reactive behavior, enabling agents to engage in both introspective goal generation and moment-to-moment decision-making within dynamic, multiplayer environments.

We implemented and evaluated this framework in a custom-built MMORPG environment chosen for its rich social dynamics, persistent world state, and emergent gameplay scenarios. Our experiments demonstrated that agents powered by our architecture significantly outperformed traditional FSM and pure GOAP-based systems in terms of behavioral coherence, adaptability, and believability. By integrating language-based cognition with plan-based action execution, our agents were able to reason, remember, react, and reflect in ways that resemble human behavior far more closely than prior approaches.

In doing so, we move a step closer toward the long-standing goal of truly intelligent, socially aware virtual characters, agents that do more than simply react to

stimuli, but instead participate meaningfully in unfolding narratives, adapt to player behavior, and evolve over time in persistent virtual worlds.

While our work establishes a promising foundation, there remains substantial room for expansion. Future efforts will focus on enabling deeper agent-to-agent collaboration, enhancing perceptual input beyond chat interactions, expanding into non-combat gameplay domains, and enriching the pool of actions to support even more nuanced, emergent behavior. We believe this direction offers transformative potential not only for games but for broader applications in simulation, education, training, and interactive storytelling.

References

[1] Stefano V. Albrecht and Peter Stone. Multi-agent cooperation and coordination with goal oriented action planning. *AAAI Conference on Artificial Intelligence*, 2021.

[2] Various Authors. Behavior selection algorithms and game ai pro articles. `http://www.gameaipro.com/GameAIPro/GameAIPro_Chapter04_Behavior_Selection_Algorithms.pdf`, 2015. Accessed: 2025-07-03.

[3] Bruce Blumberg, Marc Downie, Yuri Ivanov, Michael Berlin, Michael Patrick Johnson, and Bill Tomlinson. Integrated learning for interactive synthetic characters. *ACM Transactions on Graphics*, 21(3):417–426, 2002.

[4] Stuart K. Card, Thomas P. Moran, and Allen Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum, 1983.

[5] Michele Colledanchise and Petter Ögren. A survey of behavior trees in robotics and ai. *arXiv preprint arXiv:2005.05842*, 2020.

[6] Satoshi Kaiya, Michelangelo Naim, Jovana Kondic, Manuel Cortes, Jiaxin Ge, Shuying Luo, Guangyu Robert Yang, and Andrew Ahn. Lyfe agents: Generative agents for low-cost real-time social interactions. *arXiv preprint arXiv:2310.02172*, 2023.

[7] David Lightbown. Finite state machines: Implementing ai logic in a manageable way. In Steve Rabin, editor, *Game AI Pro: Collected Wisdom of Game AI Professionals*, pages 51–60. CRC Press, 2015. Accessed: 2025-07-03.

[8] Michael Mateas and Andrew Stern. Structuring content in the façade interactive drama architecture. In *Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, 2005.

[9] Josh McCoy, Mike Treanor, Ben Samuel, Noah Wardrip-Fruin, and Michael Mateas. Prom week: Social physics as gameplay. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, pages 319–321, 2011.

[10] Ian Millington and John Funge. *Artificial Intelligence for Games*. CRC Press, 2nd edition, 2009.

[11] Allen Newell. *Unified Theories of Cognition*. Harvard University Press, 1990.

[12] Jeff Orkin. Building the ai of f.e.a.r. with goal oriented action planning. `https://www.gamedeveloper.com/design/building-the-ai-of-f-e-a-r-with-goal-oriented-actio` 2006. Accessed: 2025-07-03.

[13] Joon Sung Park, Joseph O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*, 2023.

[14] Jinghua Piao, Yuwei Yan, Jun Zhang, Nian Li, Junbo Yan, Xiaochong Lan, Zhihong Lu, Zhiheng Zheng, Jing Yi Wang, Di Zhou, Chen Gao, Fengli Xu, Fang Zhang, Ke Rong, Jun Su, and Yong Li. Agentsociety: Large-scale simulation of llm-driven generative agents advances understanding of human behaviors and society. *arXiv preprint arXiv:2502.08691*, 2024.

[15] Wang Wang et al. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[16] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.