Demonstration of MANA: Multi-perspective Alignment of NPC Agents

Amir Tahmasbi
atahmasb@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Keshav Singh
sing1532@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Shaheer Raza
raza15@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Owen Mei
omei@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Jason Wu
jasonwu@purdue.edu
Purdue University
West Lafayette, Indiana, USA

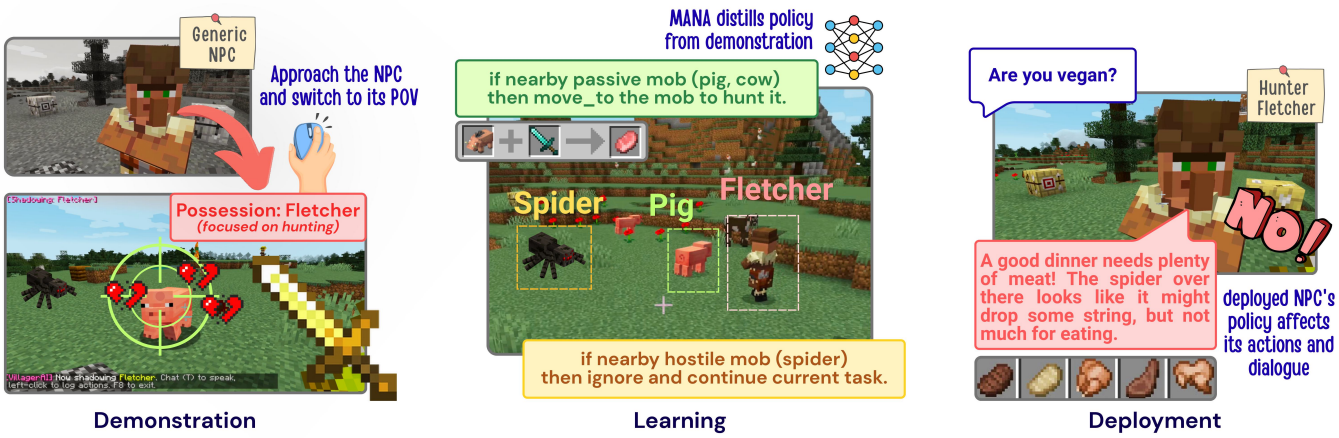


Figure 1: We implement MANA in Minecraft to align NPC behavior. The developer first takes over the NPC's body to demonstrate contextual behavior (*Demonstration*). MANA distills the demonstration into a generalizable policy (*Learning*). The NPC then acts and speaks automatically, driven by its learned profile and policy (*Deployment*). Here, an in-game NPC, Fletcher, is aligned to hunt, fill its inventory with meat, provide persona-relevant dialogue.

Abstract

We present MANA (Multi-perspective Alignment of NPC Agents), an interactive approach for authoring NPC behavior through embodied demonstration. We show MANA in the context of Minecraft, where a developer aligns NPCs while playing the game itself. Activating MANA lets the developer dynamically switch into the point of view (POV) of any character and demonstrate, in-situ, how it would act given the surrounding context, through both movement and dialogue. MANA then distills each demonstration into a generalized policy that maps in-game context to actions. Finally, once the developer switches to another NPC or exits MANA mode, this generalized policy drives the NPC on its own. We demonstrate MANA by aligning two initially generic Minecraft NPCs: Fletcher and Farmer to act aggressively and cowardly, respectively. After roughly 2 minutes of in-game play, each character exhibits consistent and believable behaviors across new, unseen scenarios.

CCS Concepts

• **Human-centered computing** → **Interactive systems and tools**.

Keywords

Programming by Demonstration, NPC behavior authoring, embodied interaction, large language models, game development tools

ACM Reference Format:
Amir Tahmasbi, Keshav Singh, Shaheer Raza, Owen Mei, and Jason Wu. 2026. Demonstration of MANA: Multi-perspective Alignment of NPC Agents. In *The 39th Annual ACM Symposium on User Interface Software and Technology (UIST Adjunct '26), November 02–05, 2026, Detroit, MI, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3830397.3842492

This work is licensed under a Creative Commons Attribution 4.0 International License.

UIST Adjunct '26, Detroit, MI, USA

© 2026 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2855-6/2026/11
https://doi.org/10.1145/3830397.3842492

1 Introduction

Non-Player Characters (NPCs) serve important roles in games where they are often necessary for game progression and enrich the gameplay experience. Traditionally, developers defined NPC

behavior through manually-authored logic (e.g., behavior trees and goal-oriented action planners) [6, 11]. However, it is difficult to anticipate a wide-range of in-game contexts and explicitly specify behavior for them all. To this end, LLMs have contributed to (i) development-time authoring of NPC scripts [3] and (ii) supporting the creation of fully AI-driven characters [7, 10]. While fully AI-driven NPCs can dynamically generate a wider range of behaviors and create more immersive experiences, it is difficult for developers to define their behaviors. Previous research found that end-users often struggle to author [12] and iterate through prompts [8].

To address this challenge, programming-by-demonstration (PBD) approaches are applied in domains such as interface design [5] and robotics [1] as a way to specify behavior through examples rather than code. PBD has also been explored for NPCs [2, 4, 9], but prior approaches often require separate training scenarios and offer limited generalization beyond the demonstrated situations. We introduce MANA (Multi-perspective Alignment of NPC Agents), which lets developers inhabit multiple NPCs and demonstrate behavior as relevant contexts naturally arise during gameplay. MANA distills each demonstration into a character-specific policy that preserves distinct personas across new situations, actions, and dialogue.

2 MANA Implementation

We implement MANA as a Minecraft mod, working in three phases: *Demonstration*, *Learning*, and *Deployment*.

Demonstration. The developer walks up to any NPC and selects the *possession* option to assume control of their POV. In this mode, the developer can choose to move, fight, build, and talk as the character, while the system logs everything in the background. Formally, a demonstration is a pair $\mathcal{D} = (\mathcal{U}, \mathcal{E})$: $\mathcal{U}$ is the speech stream, dialogue exchanged in general chat or by approaching other NPCs and opening a chat-box with them; $\mathcal{E}$ is the embodied record, logged every T ticks, pairing each action performed with the neighborhood around the agent at that moment,

$$\mathcal{E} = \big\langle (a, \mathcal{N}) \big\rangle, \qquad a \in \mathcal{A}, \quad \mathcal{N} = \{(c, d, \rho)\},$$

where a is a primitive action from the vocabulary $\mathcal{A}$ (e.g. `place_block`, `hit_entity`, `move_to`), and $\mathcal{N}$ lists nearby mobs, items, and NPCs, each with a direction d (e.g. `"N"`) and a distance ρ in blocks.

Learning. Once the developer releases the NPC, an LLM reviews the recorded speech and actions and distills them into a behavior profile $\mathcal{B} = \Sigma(\mathcal{D}) = (\mathcal{V}, \mathcal{P})$: $\mathcal{V}$ is a natural-language persona capturing the character's personality and tone, and $\mathcal{P}$ is an interpretable action policy, a set of goals governed by spatial, sequencing, and persistence rules that determine when the character attacks, ignores an entity, or builds.

Deployment. From here, the NPC acts on its own. At each moment it perceives an observation O, its current neighborhood $\mathcal{N}$ together with world context, and responds through the profile-conditioned policy $\mathcal{A} = \pi(\mathcal{B}, O)$, generating its own speech and invoking actions from $\mathcal{A}$, along with their arguments, to drive itself autonomously.

3 Scenario 1: Battle

The environment is populated with hostile mobs at night. Here we possess two NPCs and align them with opposing temperaments. We then observe the resulting behavior from their distinct profiles.

Aggressive Fletcher. The developer possesses the Fletcher and fights zombies and creepers on sight while keeping distance from the player. Each engagement is logged every 20 ticks as an action key, e.g. `"hit_entity"`, together with its `"target":"creeper"`, and the surrounding neighborhood, giving each nearby entity's direction and distance relative to the agent, e.g. `"dir":"NW","dist":12` for the creeper and `"dir":"E","dist":3` for the player standing back. The language is also captured through the chat-box: the player opens a conversation and the developer improvises the Fletcher's dialogue in real time, For example, their response to the player's remark, *"hunting creepers is so risky!"*, is *"life is short dude! let's have fun, don't worry about that,"* which gives the model clues about their attitude toward life and their tone.

After demonstration, the model learns the persona and action policy. From the temporal logs, learning phase distills an ordered goal policy: a nearby hostile mob triggers a `move_to` and `attack` response, while the player being nearby triggers `move_to` within talking range; and from the dialogues, the resulting persona's `"tone"` is casual, confident, thrill-seeking, with slang word-pieces like "dude!" recurring in their responses. At deployment, the Fletcher profile drives the NPC autonomously: it hunts hostiles and narrates its own actions in voice, e.g. *"There's a creeper nearby, bro. Time to get ready for some action and take it out before it blows up the place,"* or, on noticing the player, *"Gotta stick close, bro, especially since we're in battle mode!"*

Cowardly Farmer. Under the same threat conditions, the developer possesses the Farmer and performs the opposite response: as zombies approach, the Farmer flees, builds an enclosed shelter, and stays inside. The construction is logged with action key `"place_block"`, with material `"warped_planks"`, the time of day `"NIGHT"`, the agent's unchanged health `"hp":"20/20"`, and a hostile creeper still nearby. Learning phase produces an avoidance-first policy: a hostile entity nearby while building does not interrupt construction, and even a friendly villager approaching during a crisis is ignored in favor of continuing the primary survival task. Critically, the shelter itself becomes part of the profile. Each demonstrated block is stored by its relative offset and material under a named template, with the relative position of blocks to each other, and the shelter's world *xyz* coordinates are saved alongside it. This lets the deployed Farmer rebuild the same shelter and return to the same location on subsequent nights rather than improvising a new one. At deployment, safety overrides any competing instruction: if the player orders it to *"leave the shelter and play football with me,"* at night, the Farmer refuses and remains inside until the threat or during `"NIGHT"` state has passed.

4 Scenario 2: Socialization

In this scenario, NPCs live through a peaceful, clear day doing their routines, occasionally talking to each other when the player prompts them with a task. One of our goals here is to show that, while language and action demonstration together make for a richer character than either alone, even a single modality can expand into the other.

The developer shapes the Fletcher NPC by having it hunt passive

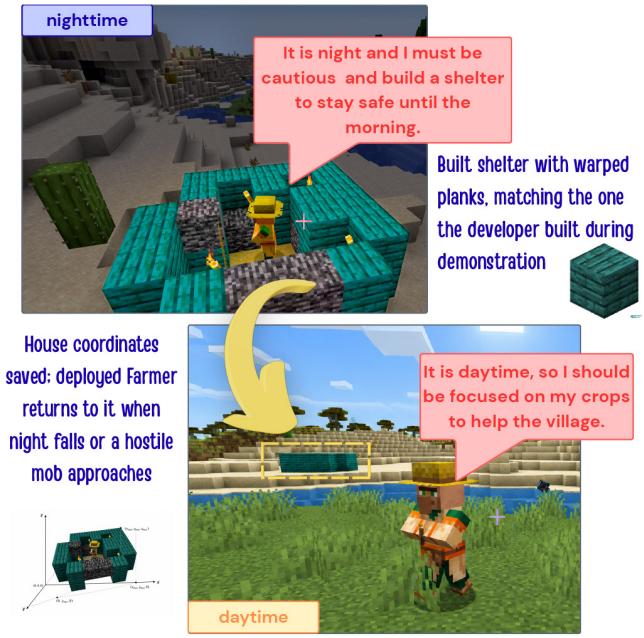


Figure 2: Deployed Cowardly Farmer behavior across day and night. At night, it returns to a shelter location learned during demonstration. During the day, it farms when no threat is present.

mobs like pigs in the environment, with each kill logged as an inventory change, e.g. `"added":["porkchop:1"]`, without ever using the chat-box channel to shape its persona. After demonstration, the Learning phase generates a policy in which a nearby passive mob, triggers a `"move_to"` and `"hit_entity"` response, while a nearby hostile mob, e.g. `"creeper, spider"`, is met with `"ignore and continue current task"`, showing the model distinguishing between mob types purely from action demonstration. Notably, even though language was never demonstrated, the Fletcher's actions still surface through language at deployment: asked what an ideal dinner for the party would be, it responds *"I'd say a good dinner needs plenty of meat,"* and when asked whether it is vegan, it flatly says no, since it hunts.

These traits also surface in NPC-to-NPC interaction. We prompted our two shaped characters, the *cowardly Farmer* and the *hunter Fletcher*, to choose a gift for the player. The Farmer proposed bread and a torch, while Fletcher proposed a sturdy pickaxe or a sharp sword; the two ultimately converged on the torch, with the Farmer insisting on safety as its reasoning and Fletcher accepting the torch as practical for exploration and keeping shadows at bay. Additionally, when asked to organize the party, the Farmer offered, *"I can help with farming for party food, brother, no need for risky hunting,"* to which Fletcher replied, *"No need for risky hunting? That's your way, Farmer, but a feast needs a good game!"* The Farmer concluded, *"It's good we both contribute in our own ways, brother! I understand your need to hunt, but please be careful."* The exchange reflects the two personas established in demonstrations: a cautious Farmer and a risk-taking Fletcher, projected here into an entirely new social context without any additional demonstration.

References

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A Survey of Robot Learning from Demonstration. *Robotics and Autonomous Systems* 57, 5 (2009), 469–483.

[2] John Drake, Alla Safonova, and Maxim Likhachev. 2016. Demonstration-Based Training of Non-Player Character Tactical Behaviors. In *Proceedings of the Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. 30–36.

[3] Game Developer. 2023. Here Are More Details on Ubisoft's Narrative AI Tools from GDC 2023. https://www.gamedeveloper.com/marketing/here-are-more-details-on-ubisoft-s-narrative-ai-tools-from-gdc-2023. Accessed 2026-07-02.

[4] Manish Mehta, Santiago Ontañón, Tom Amundsen, and Ashwin Ram. 2009. Authoring Behaviors for Games Using Learning from Demonstration. In *Workshop on Case-Based Reasoning for Computer Games, 8th International Conference on Case-Based Reasoning (ICCBR)*. AAAI Press, Menlo Park, CA, USA, 107–116.

[5] Brad A. Myers. 1992. Demonstrational Interfaces: A Step Beyond Direct Manipulation. *Computer* 25, 8 (1992), 61–73. doi:10.1109/2.153286

[6] Jeff Orkin. 2006. Three States and a Plan: The A.I. of F.E.A.R. https://api.semanticscholar.org/CorpusID:62493110

[7] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST)*.

[8] Soya Park, J.D. Zamfirescu-Pereira, and Chinmay Kulkarni. 2025. Model Behavior Specification by Leveraging LLM Self-Playing and Self-Improving. *arXiv preprint arXiv:2503.03967* (2025).

[9] Ismael Sagredo-Olivenza, Pedro Pablo Gómez-Martín, Marco Antonio Gómez-Martín, and Pedro Antonio González-Calero. 2019. Trained Behavior Trees: Programming by Demonstration to Support AI Game Designers. *IEEE Transactions on Games* 11, 1 (2019), 5–14. doi:10.1109/TG.2017.2771831

[10] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An Open-Ended Embodied Agent with Large Language Models. *arXiv preprint arXiv:2305.16291* (2023).

[11] Georgios N. Yannakakis and Julian Togelius. 2018. *Artificial Intelligence and Games*. Springer Publishing Company, Incorporated.

[12] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can't Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–21. doi:10.1145/3544548.3581388