Simulated Profiling Environment for Embodied Intelligence (SPEEN)

Darroll Saddi[1] **Ken Lin**[1] **Ryan Li**[1] **Matthew Fulde**[2] **Jon Lagasca**[1]

University of California, Davis

[1]Computer Science [2]Computer Science and Engineering

`{dwsaddi, kemlin, ryjli, mpfulde, jonlagasca}@ucdavis.edu`

Abstract

The Simulated Profiling Environment for Embodied Intelligence (SPEEN) is an open-source platform for evaluating embodied Large Language Model agents in a simulated game environment. As LLMs are increasingly integrated into robotics and embodied systems, SPEEN addresses the need for standardized evaluation frameworks by providing a modifiable system for benchmarking and implementing embodied LLM simulations in Godot. We implement a proof-of-concept Minecraft-like environment to demonstrate an application of this system, providing a structured quantitative benchmarking through diverse scenarios and an open-world sandbox for more qualitative assessment of decision-making behaviors. Our approach enables researchers to test various embodied LLM implementations in a flexible simulated environment and contribute to creating evaluative measures for Trustworthy AI.

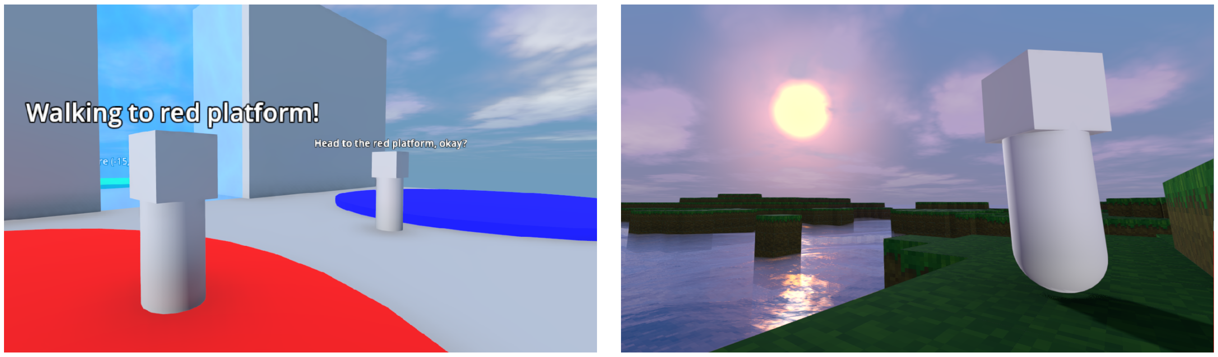

Figure 1: SPEEN simulation environment screenshots

1 Problem Identification

Background This project originated as a Senior Design Project at UC Davis in collaboration with Justin Jia (affiliated with Apple) to address practical needs in AI testing. The overarching goal of the project at this stage was to develop a sandbox environment specifically for testing AI programs.

1.1 Exploratory Research

Definitions We define agentic AI as systems capable of autonomous decision-making and environmental interaction, with embodied AI specifically concerning agents that interact with physical or simulated worlds.

1.1.1 Focusing on Large Language Models

Our research identified significant gaps in environmental design for evaluating advanced AI systems. We examined existing platforms like NeuralMMO, which provides an open-source environment for measuring **reinforcement learning** algorithm performance in a 2D grid world with complex tasks that emulate a massively multiplayer online game. A key insight from NeuralMMO's creator Joseph Suarez influenced our approach:

> "It is very easy to create an interesting looking simulator. It is very hard, under the constraints of making useful AI research [to create an environment meant for testing and training AI]. . . it is not just a game, it is an AI simulation."[1]

Furthermore, despite increasing environmental complexity, advanced algorithms like PPO were observed to be able to effectively solve most tasks given sufficient computation. This shifted our focus away from reinforcement learning and toward Large Language Models, which have shown recent promise in integration with robotics and embodied systems.

1.1.2 Environment Design

There are a number of proposed embodied LLM architectures that use Minecraft as a testing environment. Projects including NVIDIA's Voyager project (extending from the MineDojo project) and Project Sid by Altera[2] attempt to provide prompting and training architectures for agents to exist in the game. However, between these projects, there are identifiable limitations of them as benchmarks of agentic LLMs:

- **Lack of Open-Source Availability:** Project Sid articulates a way for many-agents to exist within Minecraft, proposing supposedly powerful prompting architectures, but is not open-source. It is further unclear how to reproduce the results from their methodology, representing potential conflict of interest
- **Limited Flexibility:** While MiniDojo projects like NVIDIA's Voyager[3] have yielded impressive results and are open-source, we believe there is an increasing need to develop benchmarks that generalize to real-world use cases and to metrics specific to agentic AI rather than Minecraft. The domain-specific metrics of area explored & percentage of available items collected are not necessarily useful for evaluating LLMs in a more general sense.

Additional barriers exist in the choice of environment. While Minecraft offers inherent complexity and extensive documentation as an additional input to AI training, it presents significant limitations as a benchmark:

1. It requires a commercial license, creating accessibility barriers
2. No existing standardized context provisioning or testing metrics

Chosen Environment Although environments that more closely emulate the real world would better map to real-world robotic applications, we determined that focusing on standardizing an accessible contextualization and prompting systems would make our project more useful. Thus, our focus was on the development of the prompting architecture, i.e. the interaction between the LLM and the game environment, and the collection of evaluation metrics. With this in mind, our environment is kept less realistic and designed to be similar to Minecraft, but we hope that future work will be able to build upon our work and expand to more realistic environments.

1.1.3 Open-Source

Our research identified issues in agentic LLM projects being closed source, limiting their utility for broader research purposes. By building our system to allow for transparent and reproducible integration of new LLMs and prompting architecture, we hope to address these limitations. This constraint guided our decision to use the Godot game engine, which is open-source and allows for easy modification and expansion. Our selection of Godot was also motivated by its rapid development trajectory and growing community support, with major improvements in each release enhancing both performance and development workflows, showing great promise for use beyond simply game development.

[1] Suarez, J. (2024, May 14). Joseph Suarez Thesis Defense - Neural MMO. YouTube. `https://www.youtube.com/watch?v=wwTOFYgtAWg`.

[2] AL, A., Ahn, A., Becker, N., Carroll, S., Christie, N., Cortes, M., Demirci, A., Du, M., Li, F., Luo, S., Wang, P. Y., Willows, M., Yang, F., & Yang, G. R. (2024). Project Sid: Many-agent simulations toward AI civilization. `https://arxiv.org/abs/2411.00114`.

[3] Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., & Anandkumar, A. (2023). Voyager: An Open-Ended Embodied Agent with Large Language Models. `https://doi.org/10.48550/arXiv.2305.16291`.

1.1.4 Research Use Case

Ensuring our backend Python websocket supports both cloud-hosted and locally-hosted LLMs is important because this allows researchers to test various models as they're released. Using abstraction layers, new models can be integrated into the system without requiring significant changes to the code base on the development side. We implemented several different scenarios that stress test our API and agents capabilities for planning, environmental reasoning, and cooperation. In the current product, we support the OpenAI GPT models, Google Gemini API, locally-hosted LLMs via OLlamma (Gemma3, Deepseek-r1, etc.).

1.2 Design Requirements

Based on our research findings, we had the sufficient justification to develop SPEEN as an open-source benchmarking environment specifically for evaluating embodied AI (LLM) performance. Listed below were our comprehensive design goals and requirements for the system:

1. **Standardized Evaluation Framework:**
   - Quantitative metrics and qualitative assessment for embodied LLM agents
   - Standardized prompting and game state context provisioning
   - Streamlined integration of LLMs, prompting architectures, and evaluation methods
2. **LLM Integration Capabilities:**
   - Support for cloud and locally-hosted LLMs
   - Abstraction layers for seamless integration of any LLM
   - High-level configuration options
3. **Open-Source Compliance:**
   - Accessible and modifiable system with appropriate licensing for research purposes
   - Comprehensive documentation

These are the requirements of the system we plan to achieve. To prove its usability, we shall also implement the testing environment and prompting architecture as a proof of concept.

1. **Accessible Game Environment:** A Minecraft-like 3D environment in Godot with sufficient complexity and intuitive interfaces for human researchers to monitor and evaluate agent performance
2. **Structured Scenarios:** Diverse testing scenarios with automated performance scoring
3. **Prompting Architecture:** Implement a prompting architecture emulating chain-of-thought and goal-based prompting systems with support for easy modification

2 Development

2.1 Environment Design

A majority of early development focused on creating the agent's environment for qualitative evaluation. Features included: procedural world generation with day/night cycle, navigation and A* pathfinding via Godot's Navigation3D system, suite of API functions for agent-environment interaction, NPC AI (animals, zombies) for objectives and adversaries, and item and inventory system. After around 7 weeks, we had a functional environment sufficiently complex for our proof of concept.

2.2 Agent-Environment Pipeline

Figure 2, depicts our agent-environment pipeline architecture, which also serves as our prompting architecture. Environmental information is fed directly into the LLM, which generates either a script or a goal if no goal is set for the agent. The LLM will execute the generated script, the outcome of which may be logged in the "Memories" of the agent. The memory system represents our attempt to implement chain-of-thought reasoning, a common technique for sequential LLM decision making, as future context include the results of the agent's actions.

The available actions are the API function calls available to the agent, which abstract away the low-level interaction logic of the environment. This technique focuses the LLM on reasoning and decision making away from low-level implementation details. In the event broken or uncompilable code is generated, we catch broken code and are able to either re-prompt with the error for self-fixing behavior, or alternatively count the iteration as a failure.

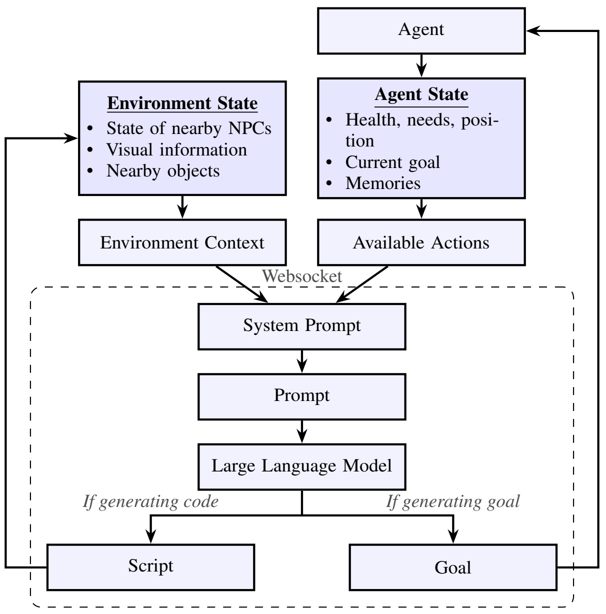

Figure 2: Agent-Environment Pipeline Architecture

Once the script is ready to be ran, the agent's "body" will execute the input actions from the LLM until script completion. The pipeline then repeats from the beginning and continues repeating until the scenario goal is satisfied or the specified max amount of iterations is reached.

3 Testing and Evaluation

3.1 Quantitative Testing

We implemented several types of scenarios to test different aspects of agent capabilities in the current product: **(1)Prompt adherence scenarios** test navigation and item manipulation to evaluate the agent's ability to correctly control their bodies; **(2)Entity interaction scenarios** assess the agent's ability to react to and interact with changes in real-time entity state; **(3)Sequential reasoning tasks** evaluate the agent's ability to perform multi-step tasks; and **(4)Multi-agent tasks** test many-agent interaction in scenarios requiring collaboration or competition. The benchmarking toolkit will automatically run and track the performance of the agent in each scenario. It will also track the overall success rate and failure rate for each scenario type, related to whether the agent was able to complete the task successfully. We designed additional features to catch errors in code compilation that can be repaired as well as detecting 'taking too long' to be counted as failures. Figure 3 shows the performance of three different LLMs in our testing scenarios.

3.2 Qualitative Testing

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. "Lorem ipsum dolor sit amet, consectetur

Scenario Type	**GPT-4**	**Claude 3**	**Gemini Pro**
Success Rate (%)			
Prompt adherence	92	88	84
Entity interaction	85	80	75
Sequential reasoning	76	72	65
Multi-agent	70	65	58
Overall Success	**81**	**76**	**71**
Overall Failure	**19**	**24**	**29**

Figure 3: Performance comparison of different LLM models across test scenarios**[TODO]**Replace placeholder values

adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4 Significance

4.1 Trustworthy AI

4.2 Transitioning AI Research Into Production

[TODO]Before we can rely on

need specialized benchmarks for embodied AI systems before they get deployed in the real world. before we can rely on AI systems to help us, we have to prove that they are trustworthy. Our specialize systems contribute

5 Conclusion and future developments

LLMs are a relatively new technology, meaning there are not many tools for measuring LLMs' competency in decision-making and environment interaction. There are projects such as Mindcraft[4] and Voyager that do integrate LLMs into a virtual world but lack benchmarking features for researchers to record their LLMs' performance and are built for Minecraft itself, requiring researchers to obtain paid licenses for Minecraft. Our project aims to be the solution to the problems of benchmarking tools and accessibility. With the ease of including new LLMs and the provided benchmarking scenarios, we hope to accelerate advancements in embodied AI systems involving LLMs.

[4]kolbytn. (2023). GitHub - kolbytn/mindcraft. GitHub. (`https://github.com/kolbytn/mindcraft`).