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

[TODO]this is supposed to only be 5 pages, with allowance for an appendix
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 critical 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 both structured quantitative benchmarking through diverse scenarios and an open-world sandbox for qualitative assessment of decision-making behaviors. Our approach enables researchers to test various embodied LLM implementations in a flexible simulated environment and contribute to developing robust evaluative measures for Trustworthy AI.

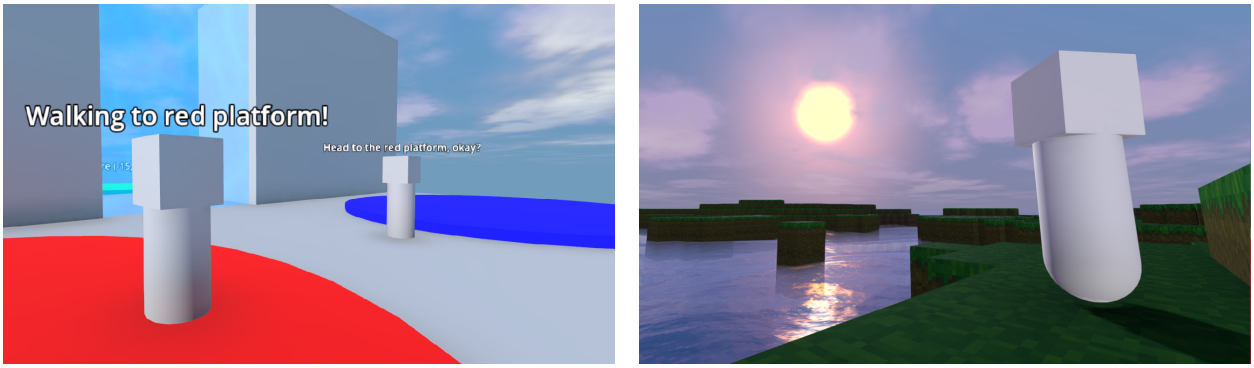

Figure 1: SPEEN simulation environment screenshots

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, these projects exhibit several critical limitations as benchmarks for 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. The methodology for reproducing their results remains unclear, potentially creating barriers to validation and further scientific progress.
- **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 or for real-world applications.

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 While environments that more closely emulate the real world would better map to real-world robotic applications, we believed that focusing on standardizing an accessible contextualization and prompting system would make our project more useful. Thus, our primary focus became the development of modifiable prompting architecture—i.e., the interaction between the LLM and the game environment—and the evaluation of agentic LLMs in a equally modular simulated environment. With this in mind, we designed our environment to simpler and more similar to Minecraft for our proof-of-concept, but we hope that future work will expand our systems to more realistic or complex 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 illustrates 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 to execute or a goal if no goal is currently set for the agent. The generated script is executed by the environment, and the outcome is logged in the agent's "Memories," which updates the agent state, enabling persistent contextual awareness across iterations.

This memory system implements chain-of-thought reasoning, allowing the agent to build upon previous experiences and maintain contextual awareness across multiple decision cycles, which proved crucial for solving complex sequential tasks.

The available actions are API function calls that abstract away the low-level interaction logic of the environment. This abstraction focuses the LLM on higher-level reasoning and decision making rather than implementation details. For error handling, we implemented a robust system that catches broken or uncompilable code and can either re-prompt the LLM with the specific error for self-correction or count the iteration as a failure for benchmarking purposes.

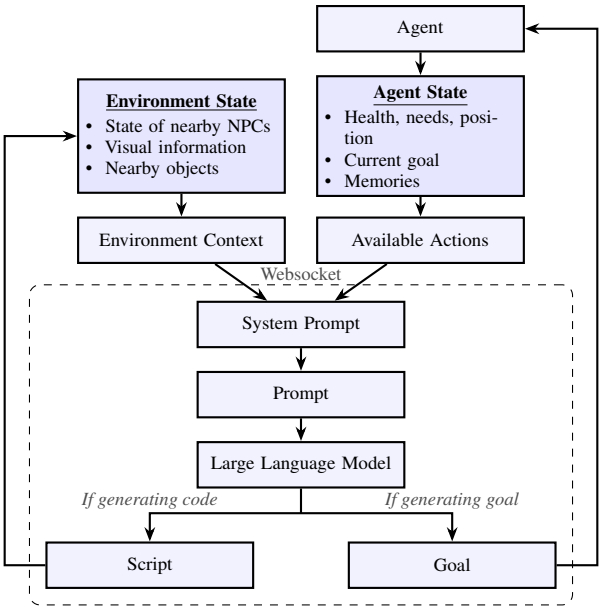

Figure 2: Agent-Environment Pipeline Architecture

Once the script is validated, the agent's "body" executes the actions sequentially until completion. The pipeline then repeats, gathering updated environmental information and continuing until either the scenario goal is satisfied or the maximum allowed iterations is reached.

3 Testing and Evaluation

3.1 Quantitative Testing

We implemented several types of scenarios to test different aspects of agent capabilities: **(1) Prompt adherence scenarios** test navigation and item manipulation to evaluate the agent's ability to correctly control their virtual body; **(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 requiring planning; and **(4) Multi-agent tasks** test many-agent interaction in scenarios requiring collaboration or competition.

Our benchmarking toolkit automatically tracks performance metrics including success rate, completion time, and error recovery attempts for each scenario. We designed the system to detect and categorize failures, including code compilation errors that can potentially be repaired and timeouts where the agent fails to make progress within a reasonable timeframe. Figure 3 shows the performance of three different LLMs in our testing scenarios.

3.2 Qualitative Testing

Beyond quantitative metrics, we conducted qualitative evaluations to assess agent behavior in open-ended scenarios. Our observations revealed several notable patterns: Our environment and prompting architecture designs translated well to the following qualitative observations, which we believe are important for future work in embodied AI:

Chain of Thought When allowed to iteratively set goals and react to the results of their own actions, agents were able to succeed in more complex tasks.

Adaptability By providing standardized context to the LLM in addition to a basic system prompt, we observed that most LLMs were capable of adapting to available information within environment and generating appropriate responses. This was especially thanks to our Memory systems and context building solutions. For multimodal LLMs, this was particularly effective, as they were able to use visual information to inform their decisions. This contextualization system allows environments, architectures, and LLMs to be easily tested by allowing for the passing of any required information. **[TODO]**this sucks

Cooperative Behavior In multi-agent scenarios, we observed emergent cooperative behaviors which would develop division-of-labor strategies when working toward common goals without explicit prompting. This emergent coordination

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]**remove placeholders

suggests potential for more complex social dynamics in future iterations of the system, and for further evaluative research into the effectiveness of deployments of these types of systems in real-world applications.

4 Significance

4.1 Trustworthy AI

As embodied AI systems become increasingly integrated into real-world applications, establishing trust through rigorous evaluation becomes paramount. SPEEN directly addresses this need by providing a simulation framework for testing and benchmarking LLMs in a controlled setting, enabling researchers and developers to assess the reliability and safety of their AI systems and architectures before deployment. Our current product accomplishes this by providing the following:

(1) Reproducible Benchmarking: Our open-source and easily modifiable environment allows for the extension of our system to new scenarios and architectures, enabling reproducible testing across different models and implementations. This establishes a foundation for valid comparisons and trustworthy claims about model capabilities.

(2) Simulated Testing: By allowing researchers to test different prompting architectures and LLMs in a controlled, simulated environment, SPEEN enables the evaluation of prompting strategies, LLM performance, agent behavior, and the overall effectiveness of embodied AI systems before real-world deployment.

(3) Standardized Metrics: SPEEN provides a consistent set of environment examples that help evaluate the performance of embodied AI systems across multiple quantitative and/or qualitative dimensions, including planning, reasoning, task completion rate, adaptability, and cooperation. These standardized metrics facilitate meaningful comparisons between different approaches and track progress in the field specific to embodied AI.

4.2 Transitioning AI Research Into Production

SPEEN bridges the gap between research and application of new AI technologies by introducing a standardized evaluation framework for the emerging field of embodied AI. We believe it critical that AI systems undergo rigorous testing and evaluation before deployment, and SPEEN provides a platform for researchers to assess the performance and reliability of their models in a controlled environment. As previously mentioned, we believe that allowing our systems to be extended to new architectures and models is critical to the success of our project as a benchmarking and testing framework. We therefore place an emphasis on the clean documentation of our system and the extensibility of the different components that make up SPEEN, always keeping in mind the potential for our work to be further developed. By providing these capabilities, SPEEN contributes to the responsible advancement of embodied AI from research concepts to practical applications, helping ensure that both our current systems and future systems are able to be used reliably, safely, and effectively.

5 Conclusion and Future Work

Large Language Models are a relatively new technology, with limited tools available for measuring their competency in decision-making and environment interaction, particularly in embodied contexts. While existing projects such as Mindcraft[4] and Voyager integrate LLMs into virtual worlds, they lack comprehensive benchmarking features and require commercial licenses, creating barriers to research accessibility.

SPEEN addresses these limitations by providing an open-source, flexible framework for rigorous evaluation of embodied LLM agents. Our system's ability to seamlessly integrate different LLMs, coupled with standardized benchmarking

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

scenarios designed to be easily modifiable, offers researchers a powerful tool for comparative analysis of embodied AI systems.

Future work will focus on three key areas: (1) expanding the range of testing scenarios to cover more diverse and challenging tasks, (2) developing more sophisticated multi-agent interactions to assess emergent social behaviors, and (3) refining evaluation metrics to better correlate with real-world application requirements. We also plan to develop environmental constraints that better approximate real-world deployments of embodied AI systems in robotics and beyond.

By making SPEEN available to the broader research community, we hope to accelerate progress in embodied AI by establishing common standards for evaluation and comparison, ultimately contributing to the development of more capable, reliable, and trustworthy embodied intelligence systems.

[TODO]this is supposed to only be 5 pages, with allowance for an appendix