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 well-documented and modifiable environment for benchmarking these agents. The framework offers both structured quantitative benchmarking through diverse scenarios measuring specific capabilities and an open-world sandbox for qualitative assessment of decision-making behaviors over extended periods. These complementary approaches enable researchers to evaluate how effectively embodied LLMs solve unseen tasks, communicate, and interact with responsive environments—key metrics for real-world applications of embodied AI systems.

1 Problem Identification

1.1 Background

This project originated as a Senior Design Project at UC Davis, where our team collaborated with Justin Jia (affiliated with Apple) to address practical needs in AI testing. The general goal of the project at this stage was to develop a sandbox environment specifically for testing AI programs. Initial discussions led us to focus on Large Language Models (LLMs) as a primary technology of interest.

1.2 Exploratory Research

1.2.1 Definitions

We define agentic AI as systems capable of autonomous decision-making and environmental interaction. Embodied AI, our primary focus, represents a subset specifically concerned with agents that interact with physical or simulated physical worlds.

1.2.2 Focusing on Large Language Models

Our research identified significant gaps in environmental design for evaluating advanced AI systems. We examined existing benchmarking platforms, particularly NeuralMMO, which provides an open-source environment for measuring **reinforcement learning** algorithm performance. NeuralMMO is a game environment with high discrete input complexity, requiring agents to navigate a 2D grid world with tasks including resource management, exploration, and agent interaction. One crucial insight from Joseph Suarez, NeuralMMO's creator, significantly 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."[TODO]cite his thesis defense

However, despite continuous added complexity to the environment over the past few years, we observed that advanced algorithms like Proximal Policy Optimization (PPO), given sufficient compute, could effectively solve most presented tasks.

This observation shifted our focus from reinforcement learning toward Large Language Models, which represent more recent advancements in artificial intelligence. We were particularly interested in the growing integration of LLMs into robotics and embodied systems.

1.2.3 Environment Design

Several proposed embodied LLM architectures use Minecraft as their testing environment. Projects including NVIDIA's contributions to the MineDojo project and Project Sid by Altera attempt to provide prompting and training architectures for agents to exist in Minecraft. However, there are identifiable problems between these projects:

- **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 produced by their environment.
- **Limited Flexibility:** While MiniDojo projects like NVIDIA's Voyager 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, it presents significant limitations for research purposes:

1. requires a commercial license, creating accessibility barriers
2. There is no standardized method for providing game state information to agents

Although environments that more closely emulate the real world would better map to real-world robotic applications, we determined that focusing on standardizing contextualization and prompting systems would be more beneficial. We decided to focus on the development of the prompting architecture, e.g. 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 still similar to Minecraft, but we hope that future work will be able to build upon our work could be expanded environments that better map to real-world applications.

[TODO]Mention Google Deepmind Creating Multimodal Interactive Agents with Imitation and Self-Supervised Learning

1.2.4 Open-Source Requirements

Our research identified that many successful LLM evaluation solutions are not open-source, limiting their utility for broader research purposes. By building our system with easy integration of new LLMs and prompting architectures, we aim to address this limitation. Our open-source approach ensures transparency, reproducibility, and adaptability—core values for scientific research tools. This constraint guided our decision to use the Godot game engine, which is open-source and allows for easy modification and expansion, and we believe is heading in a positive directions for potential use in environment design and research.

1.2.5 Research Use Case

Since the system is intended for benchmarking, we wanted to ensure the backend Python websocket, which connects to the game environment, supported both cloud-hosted and locally-hosted LLMs.

This flexibility allows researchers and enthusiasts to test the performance of various LLMs, especially as new models are released. Using abstraction layers, new models can be integrated into the system without requiring significant changes to the codebase on the development side. For higher-level access, a configuration file will be used to streamline the connection of the backend to a locally-hosted LLM, and provide users default options that demonstrate how to connect additional models.

In terms of measurement and validation strategy, we will implement a series of scenarios that test the capabilities both high and low-performing LLMs, and tracking success rate (adherence to the goal of the scenario). These scenarios will serve as our metrics capture meaningful dimensions of embodied intelligence like planning, adaptability, and environmental reasoning.

In the current product, we support the OpenAI API for GPT models, Google Gemini API, and have tested the integration of locally hosted LLMs using Ollama such as Gemma3, Deepseek-r1, and other available models. These are also largely the models we used for testing our implementation of the prompting architecture and agentic pipelines.

1.3 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 overarching design goals and requirements for the system:

1. **Standardized Evaluation Framework:**
    - Introduce a novel quantitative metrics and qualitative assessment framework for embodied LLM agents.
    - Provide standardized architecture for prompting and game state context provisioning.
    - Streamlined integration of new LLMs, prompting architectures, or evaluation methods.
2. **LLM Integration Capabilities:**
    - Support both cloud-hosted and locally-hosted LLM connections for research/evaluation purposes.
    - Provide abstraction layers for integration of LLMs when cloud hosting.
    - Include high-level configuration options for connection parameters and API settings.
3. **Open-Source Compliance:**
    - As a research tool, ensure the system is accessible and modifiable for researchers and enthusiasts.
    - Ensure all components follow appropriate open-source licensing.
    - Provide comprehensive documentation of system architecture and project structure.

With these requirements of a benchmarking system in mind, we shall also implement the quantitative evaluation methods, qualitative evaluation methods, and the prompting architecture as a proof of concept.

1. **Accessible Game Environment:**
    - Develop a Minecraft-like 3D environment using the open-source Godot engine.
    - Create a world with sufficient complexity for testing embodied AI behaviors.
    - Design intuitive interfaces for human researchers to monitor and evaluate agent performance.
2. **Structured Scenarios:**
    - Implement diverse testing scenarios targeting specific embodied capabilities.
    - Create both directed tasks and open-ended challenges.
    - Support automated performance scoring where appropriate.
3. **Prompting Architecture:**
    - Implement a prompting architecture emulating chain-of-thought and goal-based prompting systems.
    - Allowing for easy modification and expansion of the prompting system.

2 Development

2.1 Early Development

A majority of our early development was spent on building the infrastructure needed for our LLMs to properly interact with the world. This included:

- building terrain generation including ore generation with a day and night cycle
- integrating Godot's navigation tool set
- implementing player controls and LLM commands
- designing NPC AI to act as objectives and adversaries for benchmarking (animals, zombies)
- creating an inventory system and a collection of items

We originally approached the project with the intent of creating a full game roughly similar to Minecraft but quickly realized it was not feasible with the 20-weeks development time we have. Thus, we deemed what we had was sufficient and transitioned to designing how to feed our LLMs game state information and relay to the game our LLMs' response, the core aspect of our project.

2.2 The LLM-game pipeline

[TODO]Include a picture of our llm to game diagram LLMs' receive text input, usually a set of instructions or an inquiry and are sometimes attached to additional context, and then the LLM generate a text response. In our case, we inform the LLM of

- background context to inform the LLM it's in a simulated environment and is controlling a character's body in the world
- it's goal it needs to achieve
- game data such as inventory content, nearby entities, and available actions

The LLMs can't directly control their characters like how we can with controllers because they can only generate text. Thus our solution is to have the LLM write code scripts that its assigned character body will execute. We provided the LLM with additional instructions to only write its response in code. Any natural language in its response will disrupt script execution, so emphasizing this constraint to the LLM was important. As for us, the developers, we designed API functions to abstract away the low-level logic so the LLM will not have to include it in its response. We wanted the LLM to focus on translating its overall reasoning into code and not worry about little details that would disrupt execution if written improperly. However, LLMs are bound to make mistakes, including the little detailed errors that we tried to reduce, so we simply record it for our benchmarking. Overall, the steps of our pipeline can be summarized as

1. Feed the LLM the context data
2. LLM generates a response in script format
3. LLM's game character executes the LLM's intent based on the functions the LLM wrote
4. repeat

With our pipelines complete, we moved on to designing scenarios that will measure the LLMs' reasoning and correctness.

3 Testing and Evaluation

3.1 Benchmarking Scenarios

We implemented a variety of scenarios to test different aspects of agent capabilities:

1. Basic navigation: Testing the agent's ability to navigate to specific locations
2. Item manipulation: Testing picking up and giving items

3. Entity Interaction: Testing communication and combat with other entities
4. Sequential Tasks: Testing the agent's ability to perform multi-step tasks

Each scenario includes success criteria and automatic performance tracking.

3.2 Evaluation Metric

The system tracks several key metrics:

- Success Rate: Percentage of tasks completed successfully
- Failure Rate: Percentage of tasks that failed to complete
  - Counting errors in code compilation and logical errors as failures

4 Connecting to Sandia Mission Areas and NAE Grand Challenges

4.1 Relevance to Sandia Mission Areas

Our project aligns with several Sandia National Laboratories' mission areas:

1. Computing and Information Sciences: SPEEN provides a platform for evaluating and improving computational intelligence systems, directly supporting Sandia's work in advanced computing and software engineering. Our system enables systematic testing of AI capabilities.
2. National Security Technologies: By developing evaluation frameworks for embodied AI, we contribute to technologies that could enhance decision-making systems relevant to nation security applications. Autonomous AI systems require rigorous evaluation before deployment, which SPEEN facilitates
3. Engineering Sciences: The design and implementation of our evaluation environment draws on principles from systems engineering, human-machine interfaces, and simulation technology, all core engineering areas within Sandia's scope of research

4.2 Connection to NAE Grand Challenges

Our work addresses aspects of multiple National Academy of Engineering Grand Challenges

1. Engineer the Tools of Scientific Discovery: SPEEN serves as a tool for scientific discovery in the field of artificial intelligence, allowing researchers to systematically evaluate and improve embodied AI systems. This directly supports NAE's challenge to develop better tools for scientific understanding
2. Enhanced Virtual Reality: Our platform advances techniques for creating interactive, responsive virtual environments with intelligent agents. These environments can simulate and test what AI systems that could operate in the physical world.
3. Advanced AI for Improved Quality of Life: By developing better evaluation methods for embodied AI, we contribute to the broader goal of creating AI systems that can effectively assist humans in daily life and critical operations. Reliable AI requires thorough testing like those we've developed.

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 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.