Sentient Sandbox: Modify Worlds with Language Models

Joshua Jung, Michael Li, Eric Bae

Introduction

- Our **goal** is to allow a more intuitive way to approach human to machine interaction with virtual realities.
- Through natural language commands, we allow users to modify 3D VR environment using their voice
- LLMs have shown success in other fields like 2D image editing but real-time interactive environments have not been explored deeply as of late

Implementation Details

- Relationship Graph - a key component that encodes the properties of objects in the scene and their relationships with each other
- Prompt Engineering - we engineered our prompt to GPT-4o mini with ease of use in mind
- Collision Detection - checks for overlaps between oriented bounding boxes (OBB) using Unity Collider and Physics APIs for calculations

Technology Stack

We used a variety of technologies to accomplish this:

- OpenAI Whisper for speech to text
- GPT-4o Mini for text processing
- Unity for 3D rendering and VR integration
- Meta Quest 3 Headset
- Unity assets for scene building

Pipeline

As shown by (Figure 2), we describe how we achieve scene modification with voice

1. Speech recognition
   a. While exploring the scene, the user can speak out loud
   b. We record their request and convert to a text string to process
2. Natural Language processing
   a. with the transcribed text string, we extract key parts of the request
   b. Parts: target to modify, type of command, and command parameters
3. Scene Graph Analysis
   a. We have a relationship graph of the objects currently in the scene that provides context, increasing consistency of the desired output
   b. encodes not only the objects in the scene but also a bidirectional relationship between each object on where it is relative to each other
4. Collision Detection and Validation
   a. We run through many checks to make sure that any modifications made to the scene wouldn't have unintended consequences
   b. One such check is potential collisions between objects that would be inconsistent with Unity's physic engine
5. Scene Update and Execution
   a. Once validated, we will apply the command with its defined parameters to the 3D environment

Evaluation

During our testing, we made sure to test key performance metrics that reflect on user satisfaction:

- Latency
  - Currently, our latency is about 5 seconds for the user request to be analyzed and for our system to make the modification to the scene itself
  - In general, this would be good enough for many practical purposes such as editing but not so much when almost instantaneous updates are required such as simulations
- Accuracy of Modification
  - One of the biggest performance metrics is if the system correctly recognizes the intention of the human speaking
  - What we noticed is because of our use of a relationship graph, the our model works best when commands are given in relative to other objects currently in the scene that the system can recognize
  - This means saying "move the tree left of the house up by 5 units" would have a significantly higher success rate than "move the tree in the left up by 5 units" due to the system's ability to recognize the target object relative to other objects

Discussion

Benefits

- From our testing, it helps with enhanced accessibility, providing an interface for non-experts in Unity scene building to modify the scene
- Real time interaction is also a key feature. Seeing the user's request be processed in real time without having to rerun the scene or fiddle with numbers and code is a massive advantage
- It's also extremely scalable and flexible. Everyone has a different way of communicating the same message and our system can handle nuances in speech patterns. It's also very scalable, allowing the addition of new commands

Limitations

Alternate explanation for inconclusive results:

- The finite number of allowed commands severely limits the different number of ways the user can interact with their environment
- Currently limited to modification of an existing scene and not yet on object creation

Future Work

Improvements to try given time in future:

- Adding more scenes and objects to work as a benchmark to test more variety
- Adding more commands and degrees of freedom for the user to interact with
- Allow the user's phrasing to be more flexible, such as when commands are not as clear
- Object creation - instead of just modifying the scene, we would want to allow true creativity by allowing the user to create what they visualize into the scene
- Built-in language translation. As mentioned before, different people have different ways to communicate. This not only means how they talk, but also what language they use. We would like to add automatic language detection for more than just English

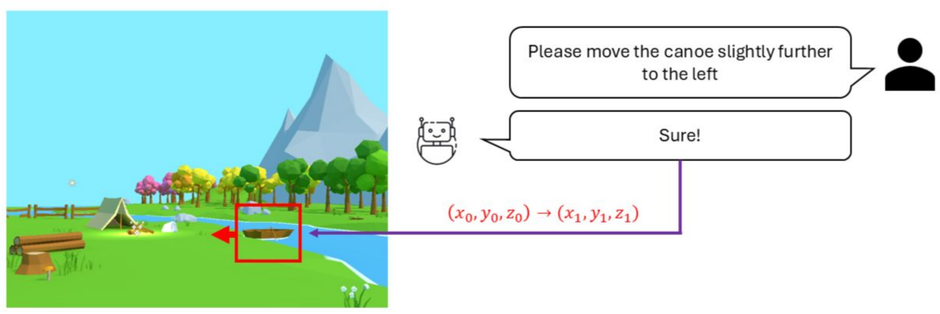

Figure 1: Example Workflow of Our Product

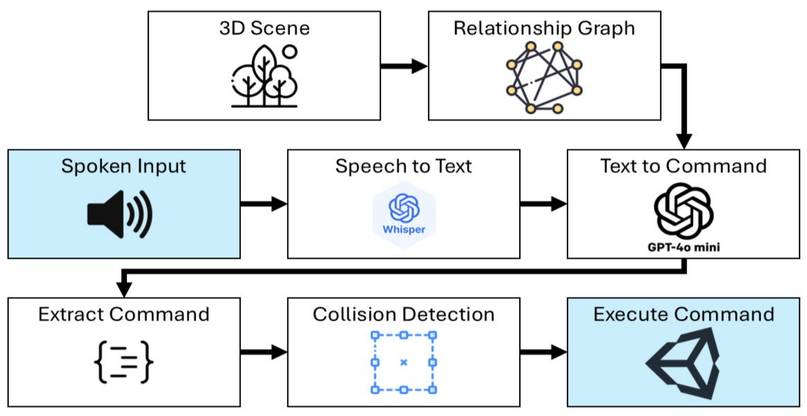

Figure 2: Pipeline