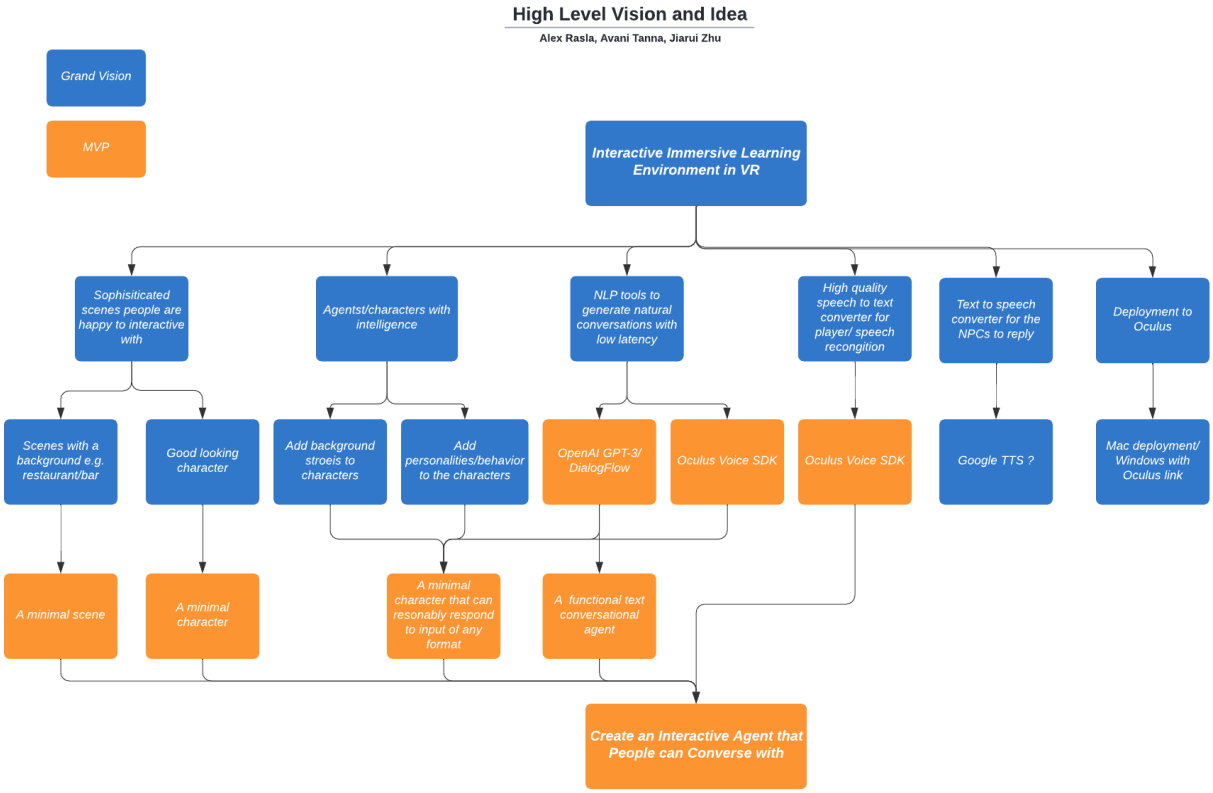
Creating Interactive Agents in an Immersive Environment

Alex Rasla, Avani Tanna, Jiarui Zhu



Introduction

Based on personal experiences with learning new languages and the process of learning in general – be it through language apps or taking classes – we realize how discouraging, nerve-racking, difficult, and error-prone it can be for foreigners to speak freely in another language. To help users better speak a language without the pressure and anxiety to be perfect, and enhance the experience of learning a language through VR immersive settings (e.g., simulating a ‘teacher’ on the other side), our grand vision focused on approaches in VR that can help enhance a user’s overall language learning experience. As a stepping stone to this goal, this project was an attempt to build an immersive learning environment in order to demonstrate and encourage social experiences with the help of the technology in a manner that does not let the user become entirely dependent on the technology. In order to do so, we propose to build

interactive agents in an immersive setting with which we can converse with on the Oculus Quest 2.

Challenges

Some of the challenges with this included integrating a natural language processing (NLP) model in order for the agent to be able to respond, using a speech recognition platform so the user can interact with the agent, and ensuring the agent moves and behaves in a realistic way. While it may seem these challenges have already been incorporated into gaming and virtual environments, the combination of all these aspects together in real time is very difficult.

The most important part of our project was to ensure our interatable agent responds in non-trivial or pre-programmed ways. The best and most efficient way to do this was to use OpenAI's GPT-3 model, which has proven to be able to carry very realistic conversations. However, in order to use this model, we needed to be able to incorporate it into Unity and deploy it on the Quest 2. Through our preliminary exploration earlier in the quarter, we found a third party library that uses GPT-3 with Unity and were able to deploy it on the Quest 2.

Another important challenge to overcome was our speech to text integration. While this isn't as important as integrating an NLP model into our application, it was an important piece of our immersive environment. In order to do so, we used the VoiceSDK feature in the Oculus Integration package in Unity. The SDK had a minor latency problem, but we combatted this with a "thinking" animation from the interactive character.

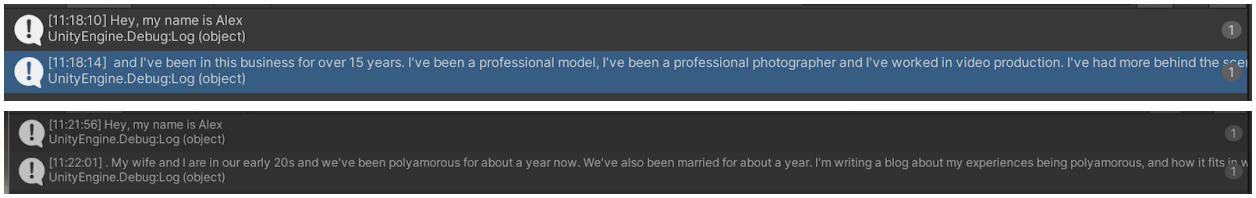
Technology Explorations

We decided to first tackle the main challenges of the project, i.e. to create an interactive agent in VR using the Oculus Quest 2 and then use NLP based techniques to implement English models such that the user is able to communicate with the agent. We ran into significant challenges when installing, deploying, and modifying existing packages for our NLP and Speech to Text processing, but managed to successfully incorporate each aspect into our project.

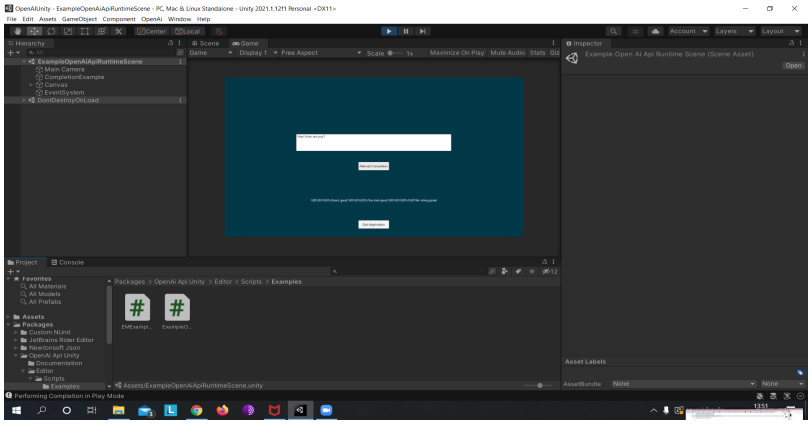
OpenAI

Although GPT-3 has been available for use for a few months, there is no official OpenAI Unity library. Nevertheless, Unity developers have tried integrating GPT-3 since it came out by creating their own packages and APIs. We explored two APIs, both of which

contained similar input/output sample scenes. Thankfully, we were able to run both the API's using the sample scenes provided by the packages. In this package the scene simply printed GPT-3's response in the Unity console as shown below:



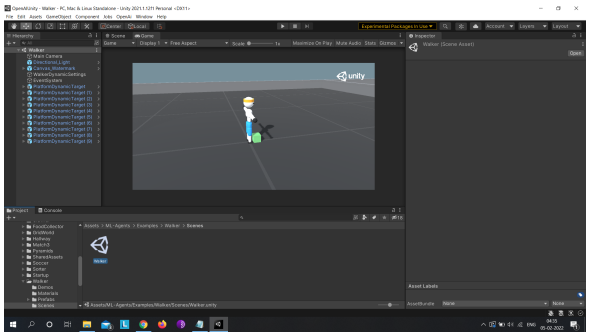
In this package's sample scenes, the user can write text in a text box and click on 'Attempt Completion'. This prompts OpenAI to complete the text or respond back within the relevant context. Here is a snapshot of the working example:

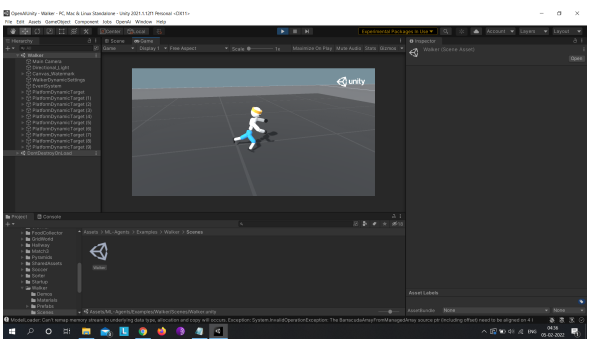


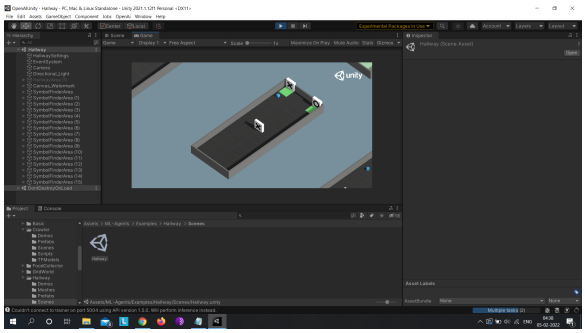
Integrating OpenAI/GPT3 was very integral to our final project. This exploration paved the way for us to further explore GPT-3 and related functionalities for our project. Since our project revolves around being able to create an interactable agent that can carry a conversation, we needed a way for the interatable agent to be able to give non-trivial or pre-programmed responses. The best and most efficient way to do this was to use OpenAI's GPT-3 model, which gave us NLP based functionalities such as text completion that are required for generating text/conversations with agents in our project.

ML Agents

The Unity Machine Learning Agents Toolkit (ML-Agents) is an open-source project that enables games and simulations to serve as environments for training intelligent agents. This technology would have provided us with intelligent agents that can be trained for VR/AR projects, reinforcement learning, or other ML based models. However, the toolkit was fairly challenging to install and train on the Quest 2, and used too much processing power so we decided not to include it in our final project. Below are some screenshots of this exploration, showing that we successfully installed and integrated ml-agents in Unity. Given more time, we would have liked to further explore interactive, intelligent agents and their functionalities and use it for our grand vision.







Below are a few resources that helped us with the exploration, and some screenshots of the agents in Unity:

- Make a more engaging game w/ ML-Agents | Machine learning bots for game development | Reinforcement learning | Unity
- Unity ML-Agents Setup — Immersive Limit

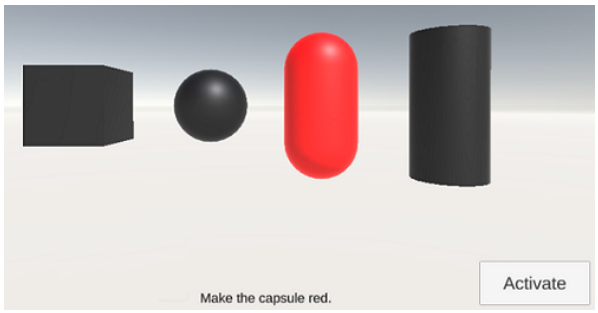
Oculus VoiceSDK

To explore the immersive aspect of our project, we decided to see if it was possible to do speech to text (STT) with the Quest 2. After exploring what other developers have achieved with STT and the Quest 2, we were underwhelmed by what had been done. Most attempts tried to use previously working Android STT methods and deploy it on the Oculus, but this obviously was extremely buggy and carried many issues alongside it (for those that succeeded). However, the Oculus Integration package in Unity recently released a VoiceSDK. This SDK is supposed to be used to integrate custom voice commands into Oculus devices. Some of the features include: automatic speech recognition to process voice requests into text, NLP for processing text into user intents, built-in activation methods, personalize voice requests baked on the user and app state using dynamic entities, real-time transcription in 13 languages.

It was released in November 2021, so there were not a lot of available resources and tutorials that we could utilize. However, as a proof of concept, we installed one of their given sample scenes onto Quest 2 which, after some debugging, was able to consistently recognize and translate speech to text. For this exploration, we ran one app

that utilizes the Voice SDK to change the color of the shapes with voice input, and one that starts and stops a timer.

Ideally, for an interactive immersive learning environment for a foreign language, we would like the user to have conversations with the characters in the VR. Simply using a text box in VR does not give the user the feeling of realism. Thus, we found it necessary to be able to have the player use their voice to interact with an agent, and have it translated into text for GPT-3 to process.



Here is some related work:

- Oculus Voice SDK
- Speech as Input in Virtual Reality
- OpenAI Unity Integration

MVP Design

The most important part of our project was to ensure our interactable agent responds in non-trivial or pre-programmed ways. The best and most efficient way to do this is to use NLP APIs such as OpenAI or DialogFlow, which have proven to be able to carry very realistic conversations. In order to use these models, we need to be able to incorporate them into Unity and deploy it on Quest 2. Throughout this process, we had to learn the libraries' scripts/code bases and how to use them for our specific use case. Normally, this wouldn't be a difficult task, but because they were third party libraries, the documentation was poor and they were not made for native deployment on the Quest 2.

DialogFlow

One of the directions for our MVP was to explore the use of DialogFlow (continuation of Api.ai) with Unity. Based on a research project published in AAMAS 2021, we tried to implement a conversational AI in Unity. We replicated their implementation of the DialogFlow API, overcame several installation related and implementation based

challenges with Apache ActiveMQ (message broker/client that allows for messages to be sent back and forth between the user and the agent), and successfully created an MVP with a functional text conversation with the agent. Here is a video of a demonstration of this MVP.

OpenAI

In another iteration of our MVP, we used an OpenAI Unity wrapper mentioned in our technology explorations to create an agent with GPT-3. Using this library, we were able to create a character with a given personality or behavior. As a result, the character was able to give responses in accordance with their specific personality. This makes not only every conversation with a specific character unique, but also will allow us to develop and converse with different types of characters within a scene. For this MVP, we were also able to incorporate speech recognition with the Oculus Voice SDK so our conversations with the interactable agent do not require entering text. Here is a link to a video of this MVP.

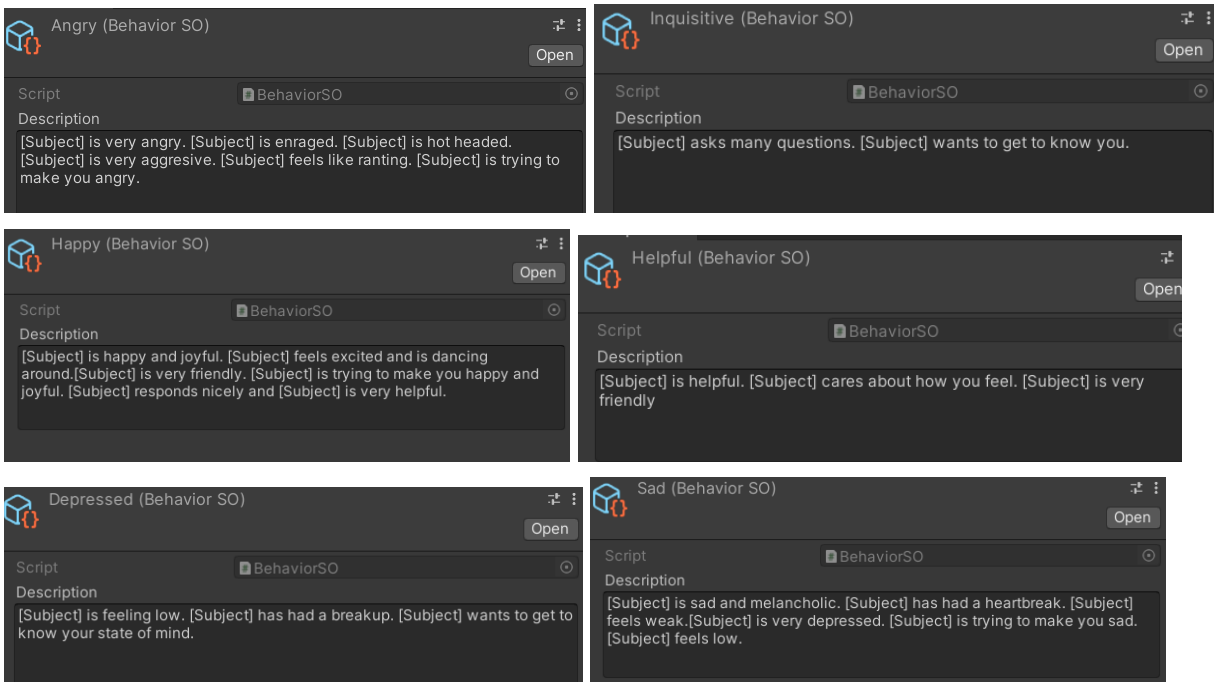
Once we got these technologies to work, we faced design choices for our final project between two interesting state of the art options. Ultimately, since OpenAI provided us with more flexibility with regards to incorporating speech recognition, compatibility with Oculus and Unity (as per our explorations - this is not the ground truth), and avoided a client-server approach as in DialogFlow, we decided to work with OpenAI. To build upon our initial OpenAI MVP, we further explored behavior scripts of the agents and created different characters with different behaviors/personalities. We also discovered that ML agents required a lot of training for our project, so we decided to build up on some of the previous work done in this area, such as in this project.

Final Project

Behaviors

We implemented three different behaviors using asset scripts in Unity that can help the character obtain a sort of a personality based on a behavior description. Below are the screenshots of examples of a combination of assets that we use, for instance, angry and inquisitive, happy and helpful, and sad and depressed. Each of the three characters in our project have a unique kind of behavior. This was a trial and error procedure where we played around with natural language input until we could successfully generate examples with significantly different behaviors. We continued to modify the behavior descriptions until we felt that the conversations represented their respective personalities descriptions.

However, it should be noted that OpenAI text generation relies on many, many other factors and may not just be controlled with behavior scripts. These behavior descriptions helped us nudge the conversations and character reactions in the right direction, but still depended heavily on the flow of the text generation and the conversation. To this end, we observe that the behavioral differences are not perfect and may require a few trials for better user experience.



Below are some results from different behavioral conversations with AI agents. You can see how different the conversations are:

- Angry and Inquisitive
- Happy and Helpful
- Sad and Depressed

User Experience

Once we were able to successfully create these behaviors, we built more immersive environments in Unity, and a menu that allows the user to select the type of character using a left and right toggle button. The female is the happy and helpful character, the male with the hair is the angry and inquisitive character, and the male without the hair is

the sad and depressed character. Once a character is selected, the user can click on the confirm button to talk to the agent in a unique, immersive scene. We chose a beach scene for the happy and helpful female, a city scene for the angry and inquisitive male, and a forest scene for the sad and depressed male.

Here is the video of the intermediary version of our app (with menu and three characters, without scenes).

Quest 2 Deployment

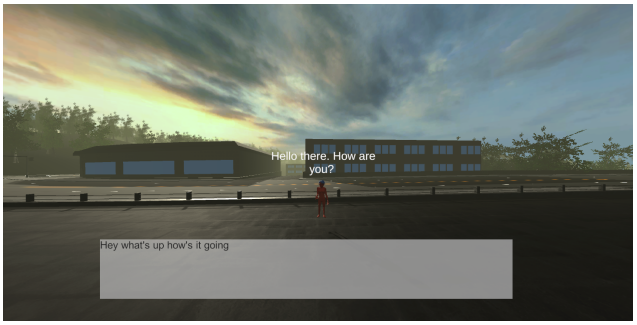
To deploy the application to the Oculus:

1. Add all the scenes to the Scenes In Build in Build Settings
2. Make sure the internet requirement is set to always in player settings
3. Connect the oculus with the mac and allow mac to access the data
4. In Build Settings, choose Android. In Run Device, choose the Oculus that is connected
5. Click Build And Run
6. Once the application is deployed to oculus, allow GPT-3 in oculus and allow audio recording
7. Make sure the Oculus is connected to internet and the mic is on

To run the applications on Oculus:

1. In the menu scene, you can press the trigger to choose left or right button to select different characters (each character will bring you to a unique scene)
2. Press the trigger to confirm your selection
3. Once you're brought to the scene, to begin a conversation with the agent, simply press the trigger on the Quest 2 controller

Below are the example snapshots of our character selection and scenes.



Evaluation/Results

Results:

- Links to the recorded video within Oculus with team members:
  - Alex
  - Avani
  - Jack
- Links to the recorded video in Unity to show the different behaviors of our agents:
  - Angry and Inquisitive
  - Happy and Helpful
  - Sad and Depressed
- Example screenshots of our scenes above

We successfully built three different characters with different behaviors and scenes in Unity, allowing the user to select the agent they want to converse with. In order to see how different the outputs were and to record how different the conversations for each of the users were, all three team members recorded videos. Here are the links to the videos showing the different conversations with the agents: Alex, Avani, and Jack. In each of the recorded videos, each team member chose one character and had a conversation to show the unique preset behavior of the character. In general, we think those behaviors match our expectations. We also have other videos recorded in Unity to show the distinct behaviors of our characters. We notice that even if we have pre-set the behaviors of characters, those behaviors are not super stable in the trials and highly depend on the flow of the conversations.

With the technology at hand, we think that the conversations are quite believable and form a good basis for future work towards our bigger goal of creating a language experience. It is difficult to say if we would be able to simulate “language teachers” and build agents that can perfect languages but we can surely see potential in terms of creating immersive environments for language learning such that a user can open up and converse freely with an agent. We believe that training and fine-tuning the language models can further make it better. Our work on the project definitely got us closer to the

overall goal although it needs more future work and user studies. We think this project also contributes to the interactive conversations in AR/VR in general since not too many people have done this before.

We have achieved most of our formative goals. In our final version, we were able to come up with three sophisticated scenes and deploy them to Oculus along with characters with different behaviors. We implemented a beach scene with a happy character within it, a racing city scene with an angry character, and a forest scene with a sad/depressed scene within it. These 3D scenes are successfuly deployable in both Unity and Oculus. Unfortunately though, users are not able to move around freely in the scenes, but this was not a priority for the purpose of our application and grand vision. As far as characters go, we were able to create three characters with different behaviors (happy and helpful, angry and inquisitive, sad and depressed). The recorded videos show the distinct behavior of each character in the conversation. We realize that our characters are not “truly” intelligent in a way that can replace a normal human being in a conversation and that their behaviors highly depend on the flow of the conversation. Ideally, with more time and resources, we would like to generate more stable and intelligent characters but fine-tuning, or training our own language models for our specific use-case.

In our grand vision, the characters were able to recognize the player's voice input thanks to the VoiceSDK that comes with the Oculus and generate a reply and display it on the screen with a reasonable amount of latency. While we were able to fully implement the speech to text and NLP processing of our grand vision, we were unable to incorporate a good text to speech converter compatible with Unity and Oculus for our NPCs to use. This would be one of the last technical steps to achieve our grand vision of the interactive and immersive learning environment.

Future Work

As mentioned above, one thing we haven’t achieved in the final presentation, but is important for the immersive environment, is the text to speech translation for the agents’ responses. Currently, we are only able to display the response in text and output it on the screen. However, this definitely disturbs the flow of the conversation and is neither fully immersive nor interactive. Ideally, we want each character to be able to respond with speech/sound in their own unique tones. We noted that this might have been possible with a paid functionality like Google TTS (subject to compatibility issues) but no good open source options were available. One aspect of our future work would be to incorporate this text to speech in the response so we can create a fully interactive environment.

We would also like to make the scenes/environments more interactive and interesting. Right now we have sophisticated 3D scenes but the player is not able to move around and interact with them. Ideally, the player should be able to move around freely within the scene and explore the environment. We also want to put multiple characters/agents in the same scene and try to generate organic/natural group conversations. This would be a huge step towards truly immersive/interactive environments.

As far as agents/characters go, we were able to create 3 characters with 3 distinct moods or personalities in our final version. However, we also found that the behaviors are highly unpredictable and dependent on the flow of the conversation. In the future, we would like to look into this and generate more meaningful conversations that are compatible with the behavior settings of our characters. We also want to add background stories to our characters. So instead of talking to an angry or a happy character, we can talk to a World War II veteran or a bartender who has met thousands of people.

Our original goal of this project is to build interactive agents that people can converse with to facilitate foregin language learning and practice. However, we discovered more potential in the process of developing this application and building the stepping stones to achieving this. We believe the idea of interactive and immersive conversations in VR can allow for a variety of experiences and applications in VR, and is inevitably going to be a very widespread technology in VR.

Contributions

Alex: For this project, I worked on exploring and integrating GPT-3 with the Oculus Quest 2, and ensuring the Oculus VoiceSDK worked for our use case. This required exploring and installing a variety of third party applications where OpenAI was potentially compatible with Unity/VR, integrating these NLP libraries with the VoiceSDK STT transcription feature, and ensuring it was deployable and worked on the Quest 2. I evaluated the choices for the NLP and STT by its applicability to our MVP, final project, and grand vision. Once the NLP and SST technology was chosen, I worked on implementing the communication between the VoiceSDK transcription and the GPT-3 in a Unity scene by ensuring the transcription text was passed to the GPT-3 backend, and managing the client/server access tokens for both OpenAI and VoiceSDK. I also worked on various UI features such as displaying the transcribed text on a Unity canvas, integrating the Quest 2 controller inputs for activating the voice input, and showing the agent's response.

Avani: For this project, I came up with the grand vision idea, worked on exploring and integrating OpenAI and ML agents toolkit with Unity, and ensured that text completion works and is compatible for our use case. This required exploring, installing, implementing and connecting several components of a variety of third party packages/wrappers. I also explored DialogFlow, a continuation of Api.ai, (that also included implementing a client and server interaction using ActiveMQ) and created a fully functional text conversation with an agent that was shown in our MVP. I implemented the behavior scripts and went through several trial and error procedures to come to the right natural language input that enabled us to nudge the conversation with the agent in a direction that would match the agent's distinct 'personality' or 'behavior'. To demonstrate this, I created three distinct behaviors - angry and inquisitive, happy and helpful, and sad and depressed (please look at the videos to see very believable conversations!). To further enhance our project, I implemented three characters with a menu/app that would allow us to toggle (left and right) between them and hence converse with agents having different behaviors (this is an integral part to have realistic agents for our grand vision). Overall, this involved state of the art technology comparisons, overcoming several installation and implementation challenges, decision making to resolve the said challenges and choose the appropriate technology, working with behavior scripts, and creating an immersive experience.

Jiarui (Jack): For this project, I first explored the Oculus Quest 2 deployment from Unity with a Mac since I own an Oculus. I also explored some core features of VoiceSDK such as NLP for processing text into user intents and automatic speech recognition to process voice requests. Alex's experience on VoiceSDK, together with my exploration results, determined our choice of VoiceSDK for our project in the MVP and final version. I also explored some text to speech converters, such as Google TTS, for our NPCs to reply, but unfortunately found no free compatible TTS. I helped Avani with creating the menu scene by finding an online tutorial and ensuring that it worked for our project. In order to give our users a more immersive experience, I implemented three 3D scenes and placed a different character in each of the scenes. I worked with the Unity SceneManager together with Avani's character selection menu to create three scene destinations for the users to explore. Finally, I worked with Alex to ensure the successful and smooth deployment of our built project to Oculus from Mac.