A SMART RPG GAME-BASED ENGLISH LEARNING PLATFORM USING GENERATIVE ARTIFICIAL INTELLIGENCE AND NATURE LANGUAGE PROCESSING

Hongjia Meng [1], Moddwyn Andaya [2]

[1] Kantonale Mittelschule Uri, Gotthardstrasse 59, 6460 Altdorf, Uri
[2] California State Polytechnic University, Pomona, CA, 91768,
Irvine, CA 92620

ABSTRACT

Language barriers continue to challenge immigrants as they adapt to new environments, often limiting their confidence and social integration. Many existing language-learning applications fail to provide immersive, adaptive, and emotionally supportive experiences—particularly for beginners.This game is about learning language by talking and interacting with NPCs in real life scenes [1]. The aim is to remove the language barrier of a newcomer who doesn't have enough motivation and bravery to talk in real life by simulating real life in a game, where the player only talks to NPCs.This paper introduces an AI-powered language learning game designed to simulate real-world conversations in a safe and engaging virtual environment. Players navigate through a city, interact with dynamic NPCs, and receive language support through a personalized guide who speaks their native language. This guide gradually shifts to the target language, helping users build confidence and fluency over time.

The system leverages OpenAI's GPT-4o model to deliver context-aware, level-appropriate dialogue, ensuring that players are neither overwhelmed nor under-challenged. A user study showed that the game effectively fosters engagement and supports language acquisition, though it also highlighted areas for improvement in navigation and user interface design. Compared to traditional apps, this game offers a richer, more supportive learning experience by combining adaptive AI, immersive storytelling, and real-time conversational practice. Ongoing development will enhance usability and explore features such as multiplayer interaction to further support language learners.

KEYWORDS

Language Learning Gamification, NPC Interaction, Immersive Language Acquisition, Real-Life Simulation

1. INTRODUCTION

Most immigrants have a hard time integrating due to language barriers. Even when the immigrants tried their best to prepare, they often lacked the ability to communicate freely and without thinking about grammar or vocabulary rules. This leads to less confidence and a psychological barrier. For example, they would hesitate to talk to neighbors if it is not necessary, because during the conversation, they risk making "funny" or "embarrassing" mistakes. It could also be the case that they simply need to think too much about all the linguistics rules during a

David C. Wyld et al. (Eds): NLCAI, AIFU, CCSEA, BIoT, SEA, SIPRO, BDML, CLOUD – 2025
pp. 51-59, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151004

conversation, and the words don't flow. What worsened the situation was bullying in public places, such as schools, as the immigrants might not be able to respond or comprehend quickly enough. Based on a 2012 report from the National Education Association, one-fourth of all Asian American students are English learners, and 54 percent of the overall Asian American population are harassed in school. Also, many schools are not prepared enough for the education of language for immigrants [2]. Many student books as well as curriculums are fully in English, which is not helpful for students who struggle with English from the very basics.

Even though there are already many apps focusing on interactive and game-like language learning, none of them combined both adaptivity, creativity and efficiency. My game, using AI-controlled NPCs, can best adapt to the player's language level and give useful hints on suitable vocabulary with the real life scenes and objects [3]. My game, in contrast to other AI based language games, includes a guide who follows the player and who can talk in the player's native language, which helps beginners a lot. The game also has a storyline which helps the player to know more about the scenes and talk to NPCs.

The problem can be split into 2 parts: a) to boost the confidence of immigrants when talking in real life scenes and b) to enable people who barely speak any English to get started.

For the first part, a game out of different scenes are made, such that the players can walk through the city and visit different places, learning the vocabulary by real life simulation [4]. Since the player knows that they are talking to an AI, they will have less burdens of being laughed at and they can talk more freely. The AI generated responses will also be leveled according to the player, so that the player doesn't need to search words up and can concentrate on the conversation. To address the second part, there's a "guide" in the game, who will be following the player and helping whenever needed. This also means that the guide can talk in the native language of the player, and for beginners, the guide can be a great help. To make the players not fully rely on the guide and thus always speaking their native language, the AI guide will gradually also adjust its level.

In our experiment, we aimed to evaluate how well the app engages the user in a fun way, easy access, and effectiveness for learning a new language. By surveying 10 participants asking them how fun, how easy to use, and how easy to learn their target language in the app by rating 1 to 10, we managed to get a result of overall positive experience. The game showed a general enjoyable experience (7.0 average rating) and highly effective for learning (9/0 average rating). However, ease of use received the lowest average rating (5.0) indicating some challenges within the app's UI or map navigation. These results suggest that while the game successfully helps users achieve our goal of engaging and fun language learning, some improvements in the UI and map guidance are needed. Overall, the experiment was successful identifying strengths and areas of improvement within our game. Since the game is still in its prototype phase, refining navigation and adding features like tutorials can enhance overall user experience.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Real Life

Sample Skeptical Question: Are the conversations really similar to real life?

Sample Response: The prompts for the AI are adjusted in a way that it imitates real life residents and their reactions. This means that they respond with grammar and vocabulary which are close to the player's level and which are not too advanced. They also pay attention to the degree of understanding of the player, offering explanations if the player seem to be hesitating.

2.2. Different Language

Sample Skeptical Question: What if users want to learn different languages other than English?

Sample Response: In the app, we allow the user to choose a different language to learn by typing any language they want. The AI will process your guide to start helping you learn the language you want.

2.3. Methods

Sample Skeptical Question: Are there other ways of learning languages other than speaking to others?

Sample Response: As of right now, speaking to NPCs is the only way of learning language as well as getting help from your guide [5]. Future plans may include learning other objects around the world for more engaging and interactive experience.

3. SOLUTION

When starting a game, you are presented with two inputs that allow you to write your own native language and the language you are trying to learn. The AI knowing what your native language tells your NPC guides what language to translate to whenever you need help understanding. Every other NPC in the world and your own guide will naturally speak the learning language you have inputted (default English) [6]. All of the conversations are started by clicking on the NPCs where a chat box shows up and where you can start typing out your conversations. All of these components are used or controlled by the OpenAI API that makes all of them work seamlessly [7].

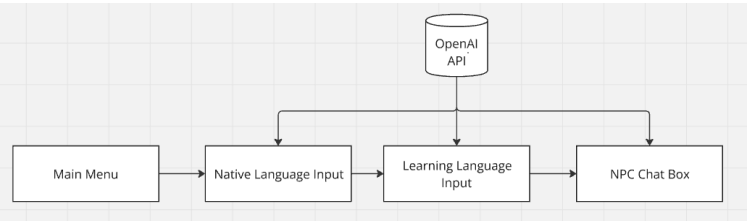

Figure 1. Overview of the solution

One of the most important components in the app is the AI manager. The AI manager controls the conversation between me and an AI assistant. We configure this AI assistant to our likings specifically creating human-like NPCs we can chat to. We use the OpenAI API to connect to a language model, allowing the AI assistant to generate dynamic, context-aware responses tailored to user preferences and interaction history.

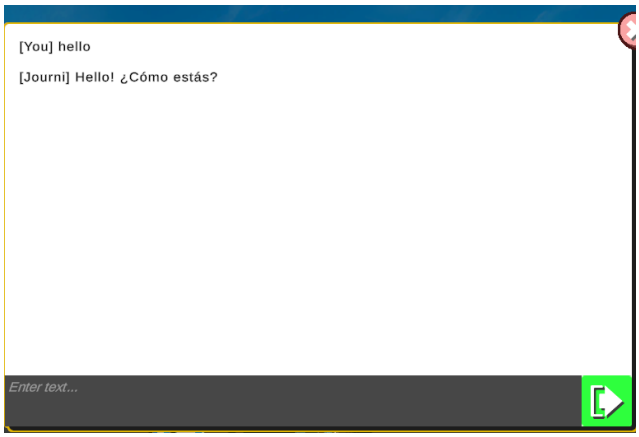

Figure 2. Screenshot of the response

```
using System.Collections.Generic;
using System.Threading.Tasks;
using OpenAI;
using OpenAI.Chat;
using OpenAI.Models;

3 references
public class AIManager : Singleton<AIManager>
{
    2 references
    OpenAIClient client;

    0 references
    void Start()
    {
        client = new OpenAIClient();
    }

    1 reference
    public async Task SendChatAsync(string input, List<Message> messages, System.Action<string> OnOutput)
    {
        if (string.IsNullOrEmpty(input))
        {
            await Task.CompletedTask;
            return;
        }

        var chatRequest = new ChatRequest(messages: messages, model: Model.GPT4o);
        var response = await client.ChatEndpoint.GetCompletionAsync(chatRequest);
        string output = response.FirstChoice;

        OnOutput?.Invoke(output);
    }
}
```

Figure 3. Screenshot of code 1

After we import the OpenAI API package, we can now start using it in our project. To access all OpenAI functionalities, we need to access them through the OpenAIClient class. The SendChatAsync function handles all inputs and output feedback from the assistant asynchronously. The function takes in an input string (the user's message), a list of Messages that acts as a storage of our conversations, and an Action event that returns the assistant's response. Inside the code, we want to first make a ChatRequest that contains the list of Messages and the model we want to use. In this project, we use OpenAI's latest smartest and fastest model GPT4o [8]. Then we await for an asynchronous function ChatEndpoint.GetCompletionAsync(chatRequest) from the OpenAIClient that sends our request to the assistant awaiting for a response back. By calling response.FirstChoice, we can get the assistant's output where we pass it in to the invoke of OnOutput Action event.

Chatting to the NPCs is another one of our most important components following the AI manager [9]. The AINPC script makes use of the AIManager to make a connection between the contact between AI and the configuration of an NPC. In this script, we add instructions and details about the NPC itself.

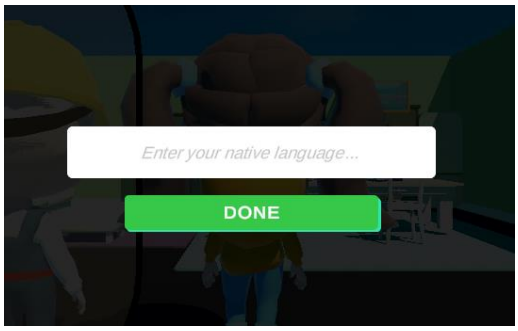

Figure 4. Screenshot of language box

```
2 references
public void SetInstructions(string language = null)
{
    hasSetInstructions = true;

    string randomName = NameGenerator.Instance.GetRandomName(gender);
    npcName = randomName;
    systemInstruction += "\nYour name will be " + npcName;

    if(string.IsNullOrEmpty(language))
    {
        systemInstruction += "\nYou dont know my native language, you can ask me.";
    }
    else
    {
        systemInstruction += "\nMy native language is " + language;
    }
    AddToMessages(Role.System, systemInstruction);
}

0 references
public void ResetMessages()
{
    messages.Clear();
    conversations.Clear();

    hasSetInstructions = false;
}

1 reference
public void SendChat(string input)
{
    if(isProcessing == false)
        SendToAI(input);
}
```

Figure 5. Screenshot of code 2

In this code, we store information about the instructions, name, gender, and other details about a specific NPC. The SetInstruction function will initialize a set of instructions of how we want an NPC to talk to us. It can take in a specific language to let the NPC know what language you speak. Different NPCs can have different instructions like how our guide NPC knows our native language that helps us translate, while other NPCs around town are human-like pedestrians with a set language you are trying to learn. The SetInstruction function will also give the NPC a random name and every NPC will have their own details of their life to give a different interaction between each other. Any inputs sent to this NPC are sent to the AIManager class and receive a message back, storing it all in a list of conversations [10]. Each NPC has their own list of conversations so we can always return to any NPC and continue a chat.

Another one of our most important components is that interaction system. Both the Interactor and the Interactable class work together to allow you to click on any object such as an NPC to interact with them. Any interactions can be different. When interacting with an NPC, a chat box will show up that access your conversation between you and this NPC only.

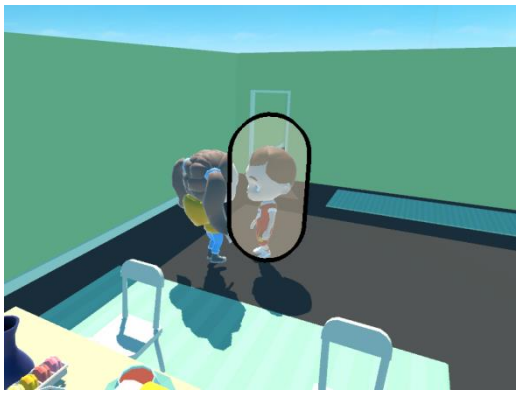

Figure 6. Screenshot of the NPC

```
void Update()
{
    Ray ray = camera.ScreenPointToRay(Input.mousePosition);
    if (Physics.Raycast(ray, out hit, range, interactionLayer))
    {
        HandleItemOutline();

        Interactable[] interactables = hit.transform.GetComponents<Interactable>();
        if (interactables.Length > 0)
        {
            foreach (var interactable in interactables)
            {
                InteractOptions option = options.Find(x => x.identifier == interactable.identifier);
                if (option == null || !interactable.canInteract || option.ignoredEventObjects.Any(x => x.activeInHierarchy))
                    continue;

                // Interaction logic
                if (option.mouseEvent && Input.GetMouseButtonDown(option.mouseIndex))
                {
                    interactable.Interact();
                }
                else if (!option.mouseEvent && Input.GetKeyDown(option.keyCode))
                {
                    interactable.Interact();
                }
            }
        }
    }
    else
    {
        ClearCurrentOutline();
    }
}
```

Figure 7. Screenshot of code 3

In the Interactor class, we use basic Unity functionalities to create the interaction system [14]. By casting a Raycast from the position of our mouse, we can create an invisible ray from where we click and get information on objects that the ray has hit. If this invisible ray hits an object within range, then we look for Intractable scripts within them. Each Interactables has something called an InteractionIdentifier which simply tells the script whether this is a basic Interact or Talk interaction. We use this information to either display different messages for each interaction in the UI screen or perform interactions with either different keybinds or mouse events. Once we perform one of the events, we simply call the found Interactble's Interact function. All Interactables that are found are also handled with outlining that visually shows the player what they're interacting with.

4. EXPERIMENT

The main focus of the program is to help people who are learning a foreign language in a more fun and interactive way. We wanted to test out how well the programs do just that, whether it was fun, easy to navigate, and helped learn a language easily.

We will ask to survey 10 people and survey them with 3 questions that fit the criteria of the game. We will ask them whether the game was fun as it is important that players are engaged, easy to navigate so players are less worried about game UI but more on learning focused, and did it help them learn the language easily more than any other methods they have done. We will have them rate each 1-10, 1 being not at all to 10 being very much so.

Fun	Easy of Use	Easy Learning
7	4	8
9	3	10
7	5	9
6	6	9
9	5	8
4	7	10
5	6	7
7	7	9
7	4	10
9	3	10

Figure 8. Table of experiment

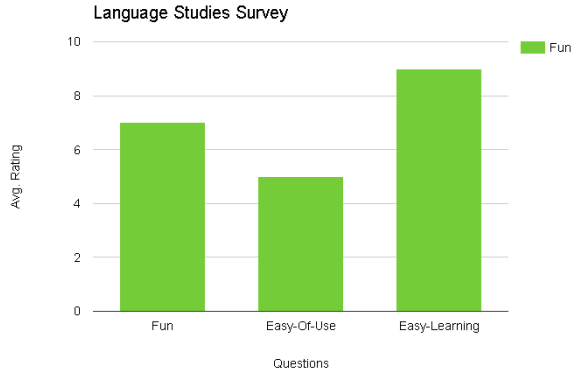

Figure 9. Figure of experiment

The survey showed overall positive feedback in making language learning fun and engaging. The game was rated with an average of 7 as fun reaching low of 5 and high of 9. The game is still in prototype and there are many more ways we can add to make the experience more fun and more content to make it overall appealing. The game also rated with an average of 9 in easy learning with low of 7 and high of 10. The AI is versatile to the user's current needs and performs real life like scenarios and conversations with a guide by your side that really helps out learning in times of need. And finally, easy of use rated with an average of 5 with low of 3 and high of 7. Again, as this game is in prototype, some features may be hard to notice or navigate through. Without the system's help like tutorials or navigation guides, players can have a hard time figuring out their next step.

5. RELATED WORK

From the game noun town, the players can learn vocabulary by going through a town and picking up objects. They will then get told the name of the object in the language they are learning, as "you'll hear audio (usually Japanese, unless talking to Lexibot) and see Japanese or English words written on a display screen"[11]. This game is similar to the game I developed in the sense that it is also through playing around in real life simulation. However, my game provides a more detailed and full preparation for entering a foreign country, because not only are the vocabulary important, but also how to form sentences. My game also simulates real life conversations with AIs.

From the game Duolinguo, the player can follow a routine with a set of problems to solve every day in the new language. “They set a certain amount of experience points as a daily goal and get bonuses for achieving it. Completing one lesson per day (achievement) adds one day to the Streak, which gets completely reset to zero if no lessons are completed on any given day (reinforcement)[12].“ In comparison to this, my game has more interaction and real life simulation. It also has a storyline to make the player stick to the game, which is similar to the reinforcement point of Duolinguo. Because the AI is adaptive to the player, it can also suit the needs of the player better (like which topics to focus on).

A game with a similar goal is Vernacular. This is a game with VR glasses, where the characters and NPCs are also controlled by AI. This game focuses on learning Japanese, including working “with consultants and natives of Japan to ensure that not only the language but also the culture is as genuine as possible.” [13]. It can be admitted that the scenes are more delicate and real-life based, but this game is less efficient for a beginner to learn the language. For someone who cannot speak a word of Japanese, the beginning would basically be guessing around. In contrast, my game gives the opportunity to talk to a guide (who’s always nearby) in the player’s mother language to clarify any confusions.

6. CONCLUSIONS

Firstly, the project is in its beginning, and there are many scenes which still need polishing. Many objects in scenes cannot be picked up or moved by the player, for example.

Also, the AI is not yet adjusted to find the exact language level of the player in the first conversations, so in the beginning, it might use words which are either too difficult or too basic. The prompt for the AI needs further adjustments.

Lastly, the current game doesn’t allow the players to interact with each other. Generally, being in some sort of a society helps to motivate players to learn, and players can also learn by chatting with other players instead of only a group of AIs [15].

REFERENCES

[1] Met, Myriam. "Learning language through content: Learning content through language." Foreign language annals 24.4 (1991): 281.

[2] Hornberger, Nancy H. "Language policy, language education, language rights: Indigenous, immigrant, and international perspectives." Language in society 27.4 (1998): 439-458.

[3] Zeng, Guanwei. "A review of AI-based game NPCs research." Applied and Computational Engineering 15 (2023): 155-159.

[4] Nation, IS Paul. "Beginning to learn foreign vocabulary: A review of the research." RELC journal 13.1 (1982): 14-36.

[5] Bruckner, Thomas, Steffen Bernet, and Henry Guldner. "The active NPC converter and its loss-balancing control." IEEE Transactions on Industrial Electronics 52.3 (2005): 855-868.

[6] Gee, James Paul. "Learning language as a matter of learning social languages within discourses." Language learning and teacher education: A sociocultural approach 18 (2004): 13-31.

[7] Auger, Tom, and Emma Saroyan. "Overview of the OpenAI APIs." Generative AI for Web Development: Building Web Applications Powered by OpenAI APIs and Next. js. Berkeley, CA: Apress, 2024. 87-116.

[8] Hesham, Alaa, and Abeer Hamdy. "Fine-Tuning GPT-4o-Mini for Programming Questions Generation." 2024 International Conference on Computer and Applications (ICCA). IEEE, 2024.

[9] Zargham, Nima, et al. "Let’s talk games: An expert exploration of speech interaction with NPCs." International Journal of Human–Computer Interaction 41.5 (2025): 3592-3612.

[10] Jeon, Yongwoog Andrew. "Let me transfer you to our AI-based manager: Impact of manager-level job titles assigned to AI-based agents on marketing outcomes." Journal of Business Research 145 (2022): 892-904.
[11] Shortt, Mitchell, et al. "Gamification in mobile-assisted language learning: A systematic review of Duolingo literature from public release of 2012 to early 2020." Computer Assisted Language Learning 36.3 (2023): 517-554.
[12] Barzilai, Sarit, and Ina Blau. "Scaffolding game-based learning: Impact on learning achievements, perceived learning, and game experiences." Computers & Education 70 (2014): 65-79.
[13] Heil, Catherine Regina, et al. "A review of mobile language learning applications: Trends, challenges, and opportunities." The EuroCALL Review 24.2 (2016): 32-50.
[14] Ekqvist, Kasperi. "Implementing User Interface Functionality for Mobile Games in Unity Game Engine." (2017).
[15] Schimanke, Florian, Robert Mertens, and Bettina Sophie Huck. "Player types in mobile learning games–playing patterns and motivation." 2018 IEEE International Symposium on Multimedia (ISM). IEEE, 2018.