Informatyka techniczna i sztuczna inteligencja
Technical informatics and Artificial Intelligence
Engineer of XXI Century'2024

Dynamic in-game dialogue with AI based language models

Karolina Bolek [1], Marcin Bernaś [2*]

[1] *Faculty of Mechanical Engineering and Computer Science, University of Bielsko-Biala, Willowa 2, 43-300 Bielsko-Biala, Poland, karolina16478@gmail.com*
[2] *Faculty of Mechanical Engineering and Computer Science, University of Bielsko-Biala, Willowa 2, 43-300 Bielsko-Biala, Poland, mbernas@ubb.edu.pl*
* *Corresponding author mbernas@ubb.edu.pl*

Abstract: The paper compares existing solutions offering the use of large language models for conducting conversations in computer games. A scene was developed in the Unreal Engine that allows communication with selected solutions existing on the market. The methods of connection were analysed and then compared taking into account many criteria. The results of the work are a set of conclusions that should guide the designer when integrating the game with language models. The work also includes the author's implementation of communication with Chat GPT.

Keywords: large language models; Rest API; communication with NPCs; game development

Dynamiczne dialogi w grach z zastosowaniem modeli językowych opartych o SI

Karolina Bolek [1], Marcin Bernaś [2*]

Uniwersytet Bielsko-Bialski, Wydział Budowy Maszyn i Informatyki, Willowa 2, 43-300 Bielsko-Biala, Polska, karolina16478@gmail.com
[2] *Uniwersytet Bielsko-Bialski, Wydział Budowy Maszyn i Informatyki, Willowa 2, 43-300 Bielsko-Biala, Polska, mbernas@ubb.edu.pl* [3]
* *Corresponding author, mbernas@ubb.edu.pl*

Streszczenie: Praca porównuje istniejące rozwiązania oferujące zastosowanie dużych modeli językowych do prowadzenia rozmowy w grach komputerowych. Opracowano scenę w silniku Unreal Engine umożliwiającą komunikację z wybranymi rozwiązaniami istniejącymi na rynku. Przeanalizowano sposoby połączenia, a następnie dokonano ich porównania z uwzględnieniem wielu kryteriów. Wyniki pracy stanowią zestaw wniosków, którymi powinien kierować się projektant przy integracji gry z modelami językowymi. W pracy zawarto także autorską implementację komunikacji z Chatem GPT.

Słowa kluczowe: duże modele językowe; Rest API; komunikacja z NPC; projektowanie gier

1. Introduction

The topic of artificial intelligence is becoming more and more popular every year. Since 1993, AI has once again climbed higher in the popularity charts, and companies such as Facebook and Netflix have started using it for advertising. Smart devices such as Roomba robots have started to appear in homes [1]. However, the greatest growth of artificial intelligence begins in 2012. At that time, programs began to appear that were able to recognize images with the help of a trained neural network. Facebook began to work on chatbots, and Google programmed AlphaStar, which reached the level of a master in StarCraft 2. Another sharp leap in the field of artificial intelligence came in 2020 when OpenAI, which created ChatGPT, a language model that is now used in many industries such as GameDev and Customer Service.

Open AI is a company founded in 2015 that deals with the development of broadly understood artificial intelligence [2]. Initially, the work focused on machine learning used in games, etc. The first tool they released on the market was OpenAI Gym, which focused on developing reinforcement learning algorithms. In 2018, the world heard

DOI: https://doi.org/10.53052/9788367652292.18

about GPT (Generative Pre-Trained) for the first time. GPT is actually a neural network that learns from the collected input data to generate appropriate answers to the questions asked. In 2021, the company presented another tool we know today, which is Dall-E. It is an artificial intelligence model that generates images based on the text provided. ChatGPT appeared on the market in the year 2022, becoming the most advanced chatbot. Open AI has also released many other tools, such as: Codex, which is mainly aimed at developers to improve the process of creating code after analyzing it, or Whisper used for speech recognition. In early 2023, OpenAI began working with Microsoft, resulting in the addition of OpenAI tools to Microsoft Azure. The latest chat model in the 4-o version was presented on May 13, 2024. It generates much more natural responses, with acceptable images, videos as input. It responds almost as quickly as the GPT-4 Turbo model, and the cost of its use by the API is half the price. In addition, this model is better at interpreting images and sounds. The GPT 4-o model it is also the first model presented that supports all types of inputs. Currently, it can be used in the free version of an OpenAI account, but there are limitations in the form of the number of generated responses per hour [3] [5] [8].

Google Bard is a chatbot created by Google, which was announced on February 8, 2023. From March 21, 2023, a list of willing people waiting to test the chatbot began to be recorded, while the world premiere did not take place until May 10, 2023. Artificial intelligence initially operated on the LaMDA (Language Model for Dialogue Applications) model, then the model was changed to PaLM (Pathways Language Model). Finally, however, on December 6, 2023, it opted for Google Gemini [10]. The company decided to train the AI on a variety of inputs from books to websites. Unlike Chat GPT, where developers have made sure that AI does not use offensive language against users, Google Bard sometimes displays inappropriate content. It is often used as an alternative to chat GPT. As in the case of one of the most popular chat bots these days, Google offers two ways to use the chat. Paid and free with restrictions when generating responses. [11] Figure 2 of Google Gemini, formerly Bard 1.1.3.

ConvAI appeared on the market quite recently, in 2022. It is a platform that offers characters that have a built-in chat bot. This solution can be used in many different fields, from Gamedev to education or virtual assistants. ConvAI has great capabilities when it comes to building communicative chatbots. To make the conversation as realistic as possible, the company has introduced a new functionality, which are shares. NPCs are adapted to perform basic animations, such as speech, walking or crouching. Currently, this functionality is in beta, which sometimes causes problems in operation. Among other things, when an NPC follows a player and is asked to stop following, they must first finish speaking the dialogue and only then standstill, which can look unrealistic. Text-To-Speech is also used by default, in the free version there are two voices in English, in the paid version there is even the option to set the Polish language. In addition to animation, ConvAI also adapted its NPC models to support facial expressions through lip movement or emotion control. In its solution, the company used NVIDIA tools such as: NVIDIA ACE (Avatar Cloud Engine), which is responsible for supporting artificial intelligence, NVIDIA Audio2Face for adjusting facial animations, and NVIDIA Riva for enabling the Text-To-Speech solution. The company does not stop there and plans to use other NVIDIA tools to improve the creation of NPCs as much as possible. [12]

NVIDIA ACE is a tool that allows you to turn standard NPCs into communicative characters equipped with artificial intelligence. In addition to the speech itself, there is also functionality for translation, or body animations tailored to the spoken words. To use a clean version of NVIDIA ACE, you need to fill out a survey requesting access to the tool. However, it should be borne in mind that this is a fairly new solution, so you need to wait at least three weeks for the application to be considered and access to the test version by a huge number of applications. There are already projects on the market that enable with AI-powered avatars.

One example is the combination of the Skyrim game with an AI in the form of an NPC named Herika through a mod. This character reacts to the player's voice and the environment by generating appropriate descriptions. With the "Machinima tool", the player can customize the appearance, voice, and animations of the companion in any way they want. In addition, you can also change her personality, which will significantly affect the quality of the dialogues. Another example is Yandere AI Girlfriend Simulator. It is an escape room game from a Japanese creator, consisting mainly of conversations. The player's task is to escape from the room, but a possessive girl stands in the way, who tries to prevent it at all costs. To be able to play, you only need to have a key to OpenAI services, but if you want to get the full experience, you need Azure Speech Service, which is additionally paid. There is no single standard for accessing language models in games. Therefore, the aim of the study is to present the capabilities and quality of responses of various AI models for interaction with characters in computer games. The visualization of the results will be carried out in the form of a game project made on the Unreal Engine 5 [13] graphics engine. In addition, the costs of financing both solutions, ease of implementation and available documentation will be analyzed.

2. Implementation of test bed

The game being designed to support a mechanism that supports dynamic dialogues supports the following elements: moving the character around the map, NPC moving behind the player, the ability to talk to NPCs and the NPC's reaction to the player's presence [4]. Due to the fact that language models are too large to be supported in the game, its support is carried out through the client communicating with the external model according to Figure 1.

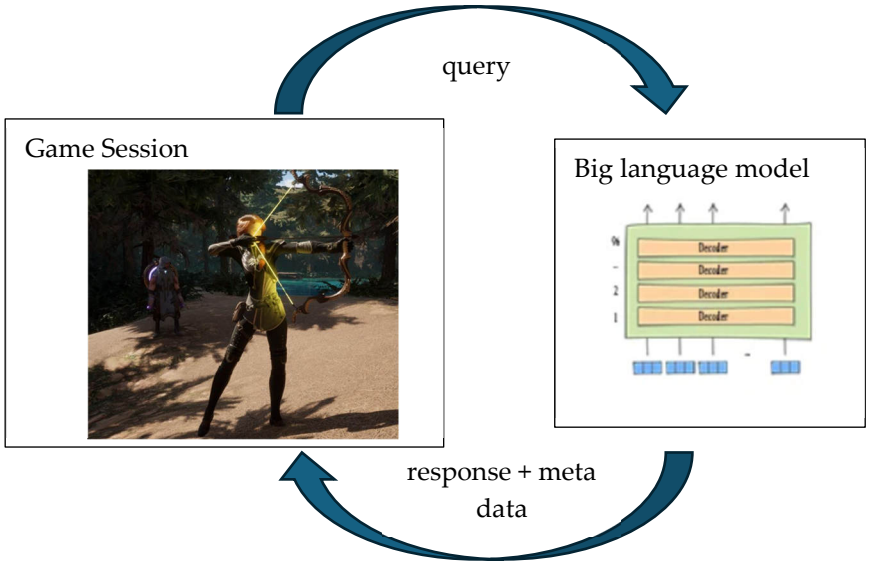

Figure 1. Model interaction between the game and the language model.

The game itself is developed in the Unreal Engine (UE) [13]. The engine allows you to create realistic-looking scenes, focusing on the reproduction of natural light or the possibility of using high-quality graphic objects. Initially, it was designed mainly for first-person shooters, but now it is an ideal solution for AAA games of almost any genre. Gameplay elements can be implemented in two ways, the first is the C++ programming language, and the second is blueprints. The latter will be used to implement the solutions in this presentation. The implementation for individual language models will vary, but the scheme of conduct will be consistent. It includes:

- Connection to the language model
- Train the model by defining the context of the response of a given avatar.
- As long as the avatar is active:
    - Retrieve information
    - Pre-validation (optional)
    - Query the model
    - Processing the response and including metadata in the avatar's emotions
    - Contextualizing the conversation (as long as the model is context-free)

Next, the next model implementations for the presented game will be presented.

2.1 Integration with OpenAI

At the time of writing, there is no official plugin that supports OpenAI in the Unreal Engine, but there are several possibilities for implementation. One of the available solutions is a plugin created by Kellan Mythen. There are three ways to install it. The first is to use the Epic marketplace, the second is to download the code directly from the author's git account, and the third is less secure is to download the content from the plugin's google drive. Before downloading, you should pay special attention to the version in which the project is created, as the extension is available on the marketplace only for UE 5.2-5.3. If you want to use OpenAI in an older version of Unreal, you need to download it via git. The Complete OpenAI API Plugin [7,8] covers all endpoints, including ChatGPT and Speech. The author provides the possibility of working directly in C++ code, but also the possibility of using blueprints. The plugin contains three modules, which are OpenAI as a runtime environment, OpenAIEditor as an editor, and OpenAITestRunner for testing.

To be able to use the package, you must first create an account with OpenAI. To use the second method of implementation, you need to add two plugins to the project, available in the Unreal Marketplace for free. One of them is Restful [http request], which can be used to send requests directly to the OpenAI assistant. The second plugin is Json Blueprint. It is needed because responses from OpenAI are sent in the form of json files. It should be noted that AI Assistant is currently in beta.

The operation on the sent and received data is based on json files, so before the request is sent to the platform, the data must be converted, in this case it will be the text entered by the player into the json type. In the figure (Fig. 2) it can be seen that the struct, i.e. a grouped data structure consisting of "roles" and "content", will be specifically replaced.

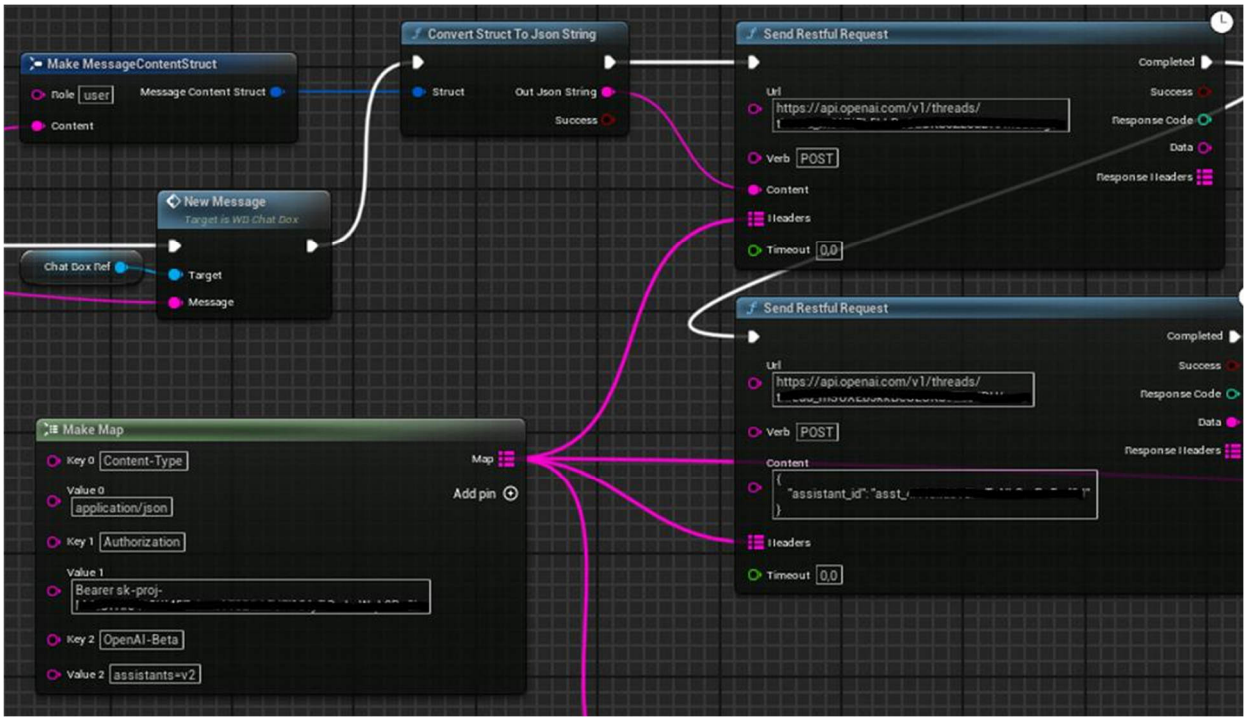

Figure 2. Blueprint fragment - headers & Requests

The data sent is exactly the role of the user, and the content will contain the text entered by the player. Below in order, there is a "New Message" block responsible for downloading data from the chat, where the user can enter a message. The "Chat Box Ref" target serves as a reference to the text box where the message is stored. Then, the need to properly communicate with the platform is to create a block containing the appropriate headers. Working closely with the documentation offered by OpenAI, the developer is informed what data the request needs to work properly, namely: type of message sent – it is necessary to set the value as application/json, authorization – this field must contain a working authorization key for the OpenAI platform, OpenAI -Beta – this text assistants=v2 must be included. This is an additional feature that can be available. Since it is optional, it does not need to be entered if the project does not use virtual assistants created on the OpenAI website. The next step to take is to send the appropriate requests. A map with headlines must be attached to each sent request. The first block of the request is responsible for sending the message content, because it is a sending operation, you should set the "Verb" field to "POST". In the "URL" there must be a link to the OpenAI platform, it is also necessary to add the number of the thread on which the request is to be handled. In the "Content" field, attach json with the message to be sent. In the second block of the request, you should also put a url with the thread number, a sign that the request will be of the "POST" type, and in "Content" you should enter the Id of the assistant created on the OpenAI platform (Fig. 3).

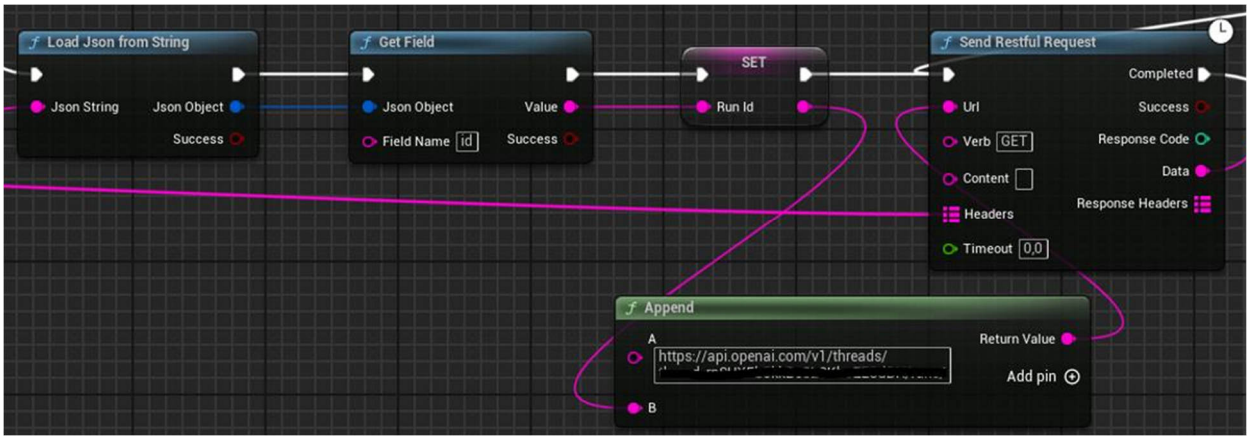

Figure 3. Blueprint Excerpt - Data Fetching

In the third part of the blueprint shown in the penultimate figure (Fig. 4), we once again create a Json object from the data generated in the previous step. We take the "status" from the facility to check if everything is working correctly. To make sure that everything is generated correctly, a two-second delay has been added. When the status is "completed", you can proceed to the last request, which is to download the file that contains the response generated by chat GPT. To make everything work, you should not forget to include headers and a url consisting of "Run Id" and a link containing the number of the thread on which we operate.

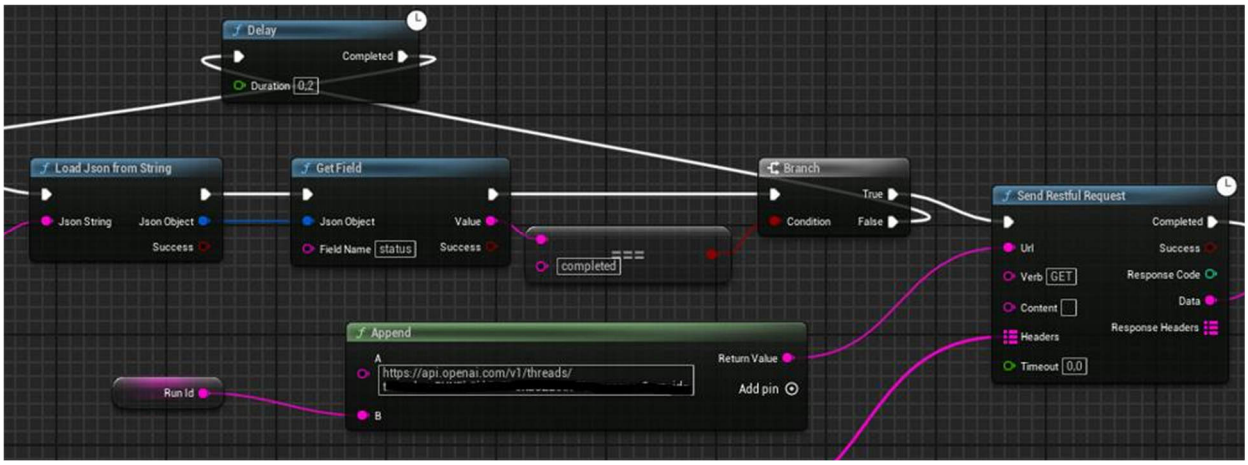

Figure 4. Blueprint excerpt - Downloading the generated message

The last step is to read the message itself, this process is shown in the last figure (Fig. 5). From the received data, we create a Json object, then because the received file is nested, we need to refer to each field separately, i.e. at the beginning we get information from the "data" fragment, then we search for the "content" field. It contains "text" and only in the "value" field is the generated message stored. The Append block was added to separate player messages from those sent by chat GPT. At the very end, all you have to do is send the text to the chat so that it is displayed to the player in the chat.

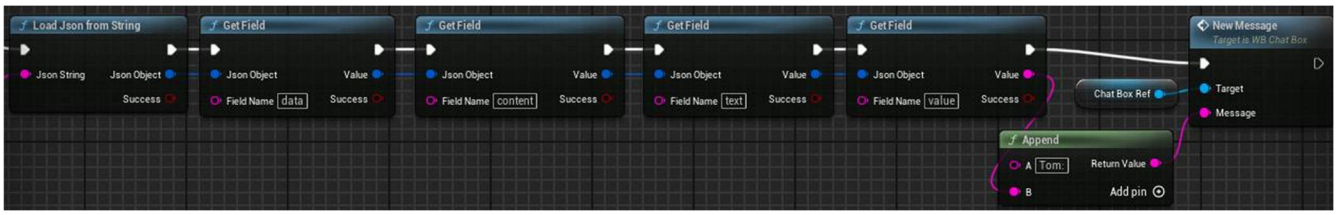

Figure 5. Blueprint excerpt - Reading the message

2.2. ConvAI Implementation

The implementation of ConvAI should be started by creating an account on the ConvAI platform. The basic version of the "Developer" account is free, and the number of available daily interactions is sufficient to run tests. It is also

important to note that only one NPC can be created in this version of the account. The next step is to download the free ConvAI plugin available in the Unreal Marketplace. This plugin has been updated to be able to use it in the latest versions of the engine, i.e. Unreal Engine version 5.4. Once the plugin is downloaded and installed, you need to add it in the Epic Games Launcher, in the library tab to the project you are creating. It is also necessary to search for the plugin in the game engine itself, in the plugins window, and select the box next to the appropriate log, which involves the need to restart the engine. When everything is properly connected, a new tab called ConvAI should show up in the project settings. To be able to contact an NPC created on the platform, it is required to paste the access key located on the platform.

With the key entered, the next step is to look into the folder where the project is located. You will need to modify the file called "DefaultEngine.ini" located in the "Config" directory. At the very top, add two lines: "[Voice] bEnabled=true". They are responsible for being able to use voice conversation when interacting with NPCs. In order for all data to be saved correctly, the engine must be restarted. With all the basic settings behind you, you can move on to adding the character itself. The plugin provides a ready-made blueprint "ConvAI baseCharacter". After adding it in the content browser, in the settings on the right, search for the field called Character ID. In the case of first-person projects, in addition, in the "BP_FirstPersonCharacter" in the "class settings" tab, you need to change the parent class to ConvAI basePlayer. This is enough to be able to start communicating via voice conversation.

The above implementation covers only the conversation itself. In order for it to proceed in the most realistic way, it is necessary to add a character with whom we can communicate. The added NPCs will come from the available, free characters from Metahuman. Once the hero model is on the stage, go to its blueprint and change it to "ConvaiBaseCharacter" in the "parent class" window. The NPCs from Metahuman have bones built in, so they can be animated at will. ConvAI also offers a set of basic animations that can be attached to the animations window. All you have to do is select the entire skeleton "Body" from the dice menu and set "Use Animation Blueprint" in the "Animation Mode" window, and lower in the "Anim Class" window you need to replace "None" with "Convai_Metahuman_BodyAnimation". These animations are general for the whole body, to add additional facial expressions, you can repeat these steps by changing the overall body skeleton to "Face" and then changing the blueprint to "Convai_Metahuman_FaceAnimation". It is also necessary to add the Character ID generated on the ConvAI platform in the general settings of the model. In the case of implementing NPCs following a player using OpenAI, this had to be done using blueprints from scratch. ConvAI offers some basic actions performed by a virtual companion, including following the player. To be able to use these options, go to the window, placeActors windows. Type "NavMeshBoundsVolume" in the search box, add it to the scene and expand it to cover the entire map. When you start the game, you can control your character with your voice to follow you, or you can ask them to stay in place.

Preparing training data in convAI means writing a backstory of our hero. However, there is a limit of a thousand words. If you don't have an idea for the right content, you can use a story generator. To best describe the backstory, ConvAI has provided some useful information that you can use when writing the text. This includes describing the character in the second person, i.e.: "You are an NPC in an adventure game", "You live in medieval times", or "You are the person who knows the most about the world around you". You should also limit yourself to a short, basic description of the personality, the world in which the character will reside. If the story is very extensive, you can use the "knowledge bank" available on the website, where you can upload or describe the story in more detail. Added files must have a .txt extension and their size must not exceed 1mb.

2.3. Other suport technologies

In addition to the communication itself, to verify the assumptions, it was necessary to create a scene containing a computer-controlled state (NPC) and the player character himself. As mentioned, in addition to the Unreal Engine itself, supporting technologies such as Metahuman, QuixelBridge, Lumen, Landscape, Foliage, Restful [http request] and Json Blueprint were used. Thanks to them, it was possible to prepare the environment for testing.

3. Analysis of solutions in the context of generating dynamic dialogues

As part of the analysis, separate projects were created, each responsible for developing a different artificial intelligence. To enable character movement, it is necessary to add the "Set Input Mode Game And UI" block, because in its absence the level will load, but the player will not be able to move around the map. Connecting the "Get Player Controller" block to this block also loads a blueprint responsible for character movement and other functionalities such as chat for communication with the GPT chat (Fig. 6).

Figure 6. Conversation with NPC in the game using text.

The game testbed was used to analyse the solution using various criteria.

3.1 Ease of implementation

The implementation of ConvAI is much simpler than the implementation of OpenAI. In the case of the first solution, it is enough to download one plugin, paste the API Key and Character ID in the appropriate places. When retrieving a person from metahuman, set ConvAI baseCharacter as the parent class, and ConvAI basePlayer should set itself as the parent class in "BP_FirstPersonCharacter". To synchronize animations, it is also enough to replace the bones of the model with body and face animations, then the model reacts to the spoken words by selecting appropriate body movements and appropriate facial expressions. To sum up, there is one way to implement it, but there are many possibilities to connect an avatar. Another advantage of using this solution is the fact that the course of the conversation is saved in a database on the platform's website, thanks to which we have access to previous conversations and can draw conclusions whether the chat learns and responds in a better way, or the opposite. The plugin also offers a clear chat with a writing and speech function, so the developer does not have to worry about creating additional functionalities at the beginning, because all the necessary elements are provided by ConvAI. OpenAI OpenAI can be implemented in at least two different ways. By using an unofficial OpenAI plugin, which, however, has several limitations, such as the blueprint proposed by the plugin cannot cope with a longer description of the NPC's history. The limit is a few short sentences, above which the virtual companion starts talking to himself. When the number of sentences decreases, it is possible to have a conversation, but because of the small amount of information, the chatbot can respond in unexpected ways. In addition, the chat loses context by not remembering previous messages.

The second option is to install two plugins available in the Unreal Marketplace. Communication between the in-game chat and the OpenAI API will consist of sending requests to the API and receiving generated content in the form of json files. When using the second solution, i.e. creating everything from scratch, the developer does not have to worry about the length of the story, because using the OpenAI assistant, all the information, from the description of the story to the selection of the chat model to be used, takes place on the OpenAI website. The context of the conversations is also known, so the conversations are natural. In the case of both solutions, developers are forced to create their own chat, because in the case of the first solution, the available chat widget covers the entire screen, preventing free gameplay, and the second solution requires creating all elements from scratch. To sum up, if a developer wants to test basic, pre-prepared functionalities, it is enough to use ConvAI, and if they want more freedom of action, for example in the form of a chat appearance or good animation, OpenAI will be a better solution, although more advanced.

3.2. ConvAI Available Documentation

ConvAI describes in detail the implementation of the plugin in the Unreal Engine and Unity. However, the platforms where you can use virtual assistants do not end there. Artificial intelligence can also be used on Discord, Roblox or Omniverse, among others. On the plus side, the documentation is divided into categories, so you can quickly get to the information you need.

In the case of the documentation provided by OpenAI, there is no exact division into platforms, only available functionalities. Sample API requests are presented in three languages: Curl, Python and Node.js. Despite the limitation of a small number of examples on various platforms, the aids are constructed in such a way that the tool can be easily implemented in any available place.

The documentation provided by NVIDIA is extremely detailed. The website includes an extensive table of contents including, among other things, the latest "Release Notes", steps to take when starting to work with ACE, and a list of available tools. After selecting a sample tool, another extensive table of contents is displayed with an extremely detailed description of its use, architecture, or implementation on a huge number of platforms. For Unreal Engine there is a chapter on installing the NVIDIA ACE plugin version 2.1. The documentation also provides a sample project called "Kairos" that demonstrates the NVIDIA ACE toolkit in action. NVIDIA also offers a number of tutorials designed to show how each tool works. The UNREAL ENGINE aids are further enhanced with screenshots from the engine window to convey knowledge as accurately as possible and facilitate implementation.

3.3 Price of the solution

OpenAI offers different models with different prices and response quality. Costs depend on the number of tokens used, which are divided into context tokens (transfer of information) and generated tokens (length of response). The cheapest GPT-3.5 Turbo model costs 0.014 cents for sending a message and 0.005 cents for generating a reply. GPT-4 is the most expensive model, with a cost of about PLN 1 per query. The GPT-4 Turbo version is cheaper, costing about 0.50 cents per query. The latest GPT-4-o costs about 0.20 cents per query.

ConvAI offers four types of accounts:

- Free: Free, with a limit of 100 messages per day and a 1MB knowledge bank.
- Gamer: $9 per month, 4000 messages per month, and a 5MB knowledge bank.
- Indie Dev: $29 per month, 10000 messages per month, additional queries for $0.005 each, 20MB knowledge bank.
- Professional: $99 per month, 40000 messages per month, additional queries for $0.0025 each, 100MB knowledge bank, access to alpha features.

While the NVIDIA ACE toolkit is in the testing phase, it can be accessed for free by completing a survey and requesting to join beta testing. The key problem with using the service is the waiting time for the response from the NVIDIA website due to the huge interest. Giving developers access for free provides an ideal opportunity to test the tool before potentially buying the full version.

3.4. Quality of the answers to the questions asked

To compare the quality of Chat GPT and ConvAI's responses, a set of several questions was asked to both technologies: an affirmative sentence about the game world "There's a nice lake in this forest," a question about the game world "Will this lake help me fix portals?" and a non-game question "What song was the most played on YouTube in 2010?"

Chat GPT version 3.5 in response to an affirmative sentence resumed the conversation, agreeing with the player, but used the unnatural word "contemplation". Answering a question about the game world, he agreed with the player and presented a way to use the lake. He answered a question unrelated to the game in general terms and changed the subject of the conversation. Version 4 responded similarly, again using the unnatural word "contemplation" and maintaining a serious tone. Version 4 Turbo handled the non-game question better by using a grammatical error, which made the answer feel more natural. Version 4-o reacted similarly to previous versions, but responded differently to the question about the lake and showed an overly serious tone of conversation.

ConvAI reacted briefly and snappily to the affirmative statement, not encouraging further conversation, but doing its job. He answered the question about the game world in the right number of words, introducing a sarcastic question to the player. When asked not related to the game, he suggested a lack of knowledge about YouTube and quickly changed the topic to the main task.

A comparison of these technologies shows that ConvAI is better at replicating the character's sarcastic behavior, which makes conversations seem more natural and concise, while Chat GPT maintains a more serious tone.

3.5. Dynamism of dialogues

To bring dynamism to dialogues, you need to know blueprints or C++. The implementation of chat GPT from OpenAI opens up great opportunities for the developer, allowing for extensive personalization. The element of dynamism in the design depicts the player's interaction in specific places. When chat GPT initiates a conversation, a message pop-up appears in front of the player. To close the window, the player must press the "Q" key twice. The virtual companion triggers the dialogue mechanism by passing through a trigger on the map.

The dynamism in ConvAI is based on the AI's reaction to words and texts typed by the player, and communication in the free version takes place only in English. For example, after the command "follow me", the NPC starts following the player, and after the command "stop following me", it stops doing so. ConvAI does not generate random messages, but reacts to specific player commands, which makes interactions more personalized and dynamic.

3.6. Generating messages in many running projects at the same time

Unreal Engine allows you to run up to four instances of a level at once. It analyzed how different AIs cope with generating messages in multiple windows.

Chat GPT tested on version 3.5 in three instances of the project showed some imperfections. In each instance, "hey" was entered. Chat did not correctly generate many responses at once - in one window there was one answer, in another two, and in the third all three. You can switch between windows using SHIFT + F1, which allows you to control the mouse outside the game window.

ConvAI was tested on the free version, also in three instances. Each window had its own individual figures and huts that were not visible in the other windows. The ConvAI plugin blocked the possibility of taking over the cursor outside the game window, which resulted in an automatic query being sent after it was typed.

4. Results

Choosing the right AI requires a thorough analysis of the functional requirements that the developer cares about the most. Finances are also an important issue, i.e. what maximum contribution can be spent on testing and gameplay. When taking the bill, you can follow the following conclusions:

- Using OpenAI is freer, because there are many more possibilities to use it, while ConvAI makes the user dependent on the type of package.
- The implementation of OpenAI requires more knowledge of the C++ programming language or blueprints, because if you want to create more demanding elements, you have to prepare them yourself, while using ConvAI, the basic interactions are implemented in the plugin plugged into the project.
- When testing answers in OpenAI, depending on the model used, there are varied prices tailored to the speed and quality of the responses. Using the free ConvAI package, this possibility is much smaller, as it is limited to 100 interactions per day, which is useful for the tests themselves, not the game.
- The implementation in both cases is quite simple, only OpenAI requires more advanced knowledge of blueprints.
- Proper preparation of the input data (NPC Story) works correctly in both cases.
- In ConvAI, it communicates with NPCs by default via a microphone and in writing, while in OpenAI you have to decide for yourself whether the conversation will be conducted in spoken or written form.
- The messages generated by different GPT chat models are of similar quality and mainly differ in cost.
- The ConvAI plugin is designed for one player by default, especially when running several instances of the project at once, when each instance has its own window. OpenAI, on the other hand, does not have any default settings, forcing the developer to analyze the game settings more deeply.

If the project is created only for private purposes, ConvAI is enough to test new functionalities, because the basic "Free" version is free. 100 messages per day is enough to play one game, and the ability to create only one character does not prevent you from using the tool. For larger, commercial projects, the prices of ConvAI packages may be too high. In this case, a better option will be the OpenAI platform, where you can flexibly top up your account and adjust the costs of queries depending on the selected GPT chat model.

Another important issue is the skills of the developer. For beginners who want to create a project at a low cost, ConvAI is a simple choice, with easy implementation of the key and character ID. OpenAI requires basic technical knowledge, and the documentation only supports examples in Python, Curl, and Node.js, which forces you to find solutions in C++ or blueprints on your own.

The desire to further expand the project is another important issue. ConvAI can be more difficult to expand without a thorough knowledge of the available features. OpenAI's Chat GPT offers more freedom to modify blueprints, which makes it more flexible.

Acknowledgments

The work was fulfilled within the framework of Erasmus+ project "The transferable training model - the best choice for training IT business leaders" (project no. 2023-2-PL01-KA220-HED-000179445). Namely, the work contributes to the project results concerning studying use cases of AI and IoT good practices (work packages 2 and 3).

Reference

1. Kaja Ines Raszyd, Alicja Wesołowska, Klaudia Tomaszewska. Sztuczna Inteligencja w nauce – jak studenci wykorzystują AI w edukacji wyższej. Akademia Zarządzania, ISSN 2544-512X, 8(3), pp. 373-400, 2024.
2. Fan, Xueman, J. Wu, and L. Tian. "A review of artificial intelligence for games." Artificial Intelligence in China: Proceedings of the International Conference on Artificial Intelligence in China. Singapore: Springer Singapore, 2020.
3. Konstantinos I. Roumeliotis and Nikolaos D. Tselikas, „ChatGPT and Open-AI Models: A Preliminary Review" 26.05.2023
4. Henk Venter and Wilhelm Ogterop, "Unreal Engine 5 Character Creation, Animation and Cinematics Create custom 3D assets and bring them to life in Unreal Engine 5 using MetaHuman, Lumen, and Nanite" 30.06 2022
5. Cameron Hashemi-Pour, "What is generative AI? Everything you need to know" URL: https://www.techtarget.com/searchenterpriseai/definition/OpenAI [access date: 01.06.2024]
6. University of Szczecin, "History of Artificial Intelligence" URL: https://ai.usz.edu.pl/historia/ [access date: 01.06.2024]
7. Marcos Romero, Brenden Sewell, Luis Cataldi, "Blueprints Visual Scripting for Unreal Engine 5. Unleash the true power of Blueprints to create impressive games and applications in UE5 - Third Edition" 02.05.2022
8. Konrad Mach, "The ChatGPT Bible. Harness the power of Artificial Intelligence" 14.12.2023
9. Stuart Butler, Tom Oliver, Christopher J. Headleand, „Game Development Patterns with Unreal Engine 5. Build maintainable and scalable systems with C++ and Blueprint" 05.01.2024
10. Jacek Ross, "Unity and C#. Game programming practice" 29.06.2020
11. Damian Kowal, "Google Bard – what is it?" URL: https://cyrekdigital.com/pl/bazawiedzy/google-bard/ 29.01.2024 [access date: 10.06.2024]
12. Yasmina Benkhoui, „Spotlight: Convai Reinvents Non-Playable Character Interactions" 08.01.2024 [access date: 10.06.2024]
13. Documentation from Unreal Engine, URL: https://dev.epicgames.com/documentation/pl/unreal-engine/unreal-engine-5-3-documentation?application_version=5.3, [access date: 03.06.2024]