

# Creative Making: MSc Advanced Project

University of the Arts, London  
Creative Computing Institute

## Dynamic Quest Generation Framework: Harnessing the Power of Large Language Models to Shape Emergent Narratives in Video Games

**Author:**

Eva Hayek

**Supervisor:**

Phoenix Perry

February 2024

---

## Acknowledgements

I would like to thank my supervisor and course leader Phoenix Perry for the understanding and support shown to me during these last few months, as well as for teaching us to find joy in our daily lives. This support has helped me immeasurably in being able to work on this project calmly and confidently. I would also like to thank all of my teachers, the CCI technicians and my fellow classmates for their help, guidance and inspiration during my time at the CCI.

---

## Abstract

This paper explores the concept of dynamic quest generation in video games, proposing a novel architecture that enables players to craft unique quests based on their input. A 3D role-playing game was built in Unity which allows players to enter their thoughts in an in-game journal. This architecture leverages a fine-tuned OpenAI GPT model (Generative Pre-trained transformer) via the OpenAI application programming interface (API) to produce quest series tailored to these journal entries, which are then translated into actionable and playable quests in the game environment. This paper investigates the mechanism to fine-tune a GPT model in order to produce a tailored model capable of consistently crafting quests in the desired format and which grasps the nuances of the game world. The iterative process of data collection and fine-tuning models led to the development of two models, with the second addressing the limitations of the first. Subsequently, the second fine-tuned model was integrated in the Unity game environment to test the technical feasibility of the quest series from start to finish. The paper concludes with a discussion of the architecture and general considerations of using GPT models in video games, as well as potential future avenues to extend this architecture.

---

# Table of Contents

|       |  |       |
|-------|--|-------|
| 1.    | Introduction   | p. 6  |
| 1.1   | Motivation   | p. 6  |
| 1.2   | Research Question  | p. 6  |
| 1.3   | Background   | p. 8  |
| 1.4   | Literature Review  | p. 9  |
| 1.5   | OpenAI's API - Fine-tuning and Function Calling Overview | p. 10 |
| 2.    | Methodology  | p. 11 |
| 2.1   | Building the Game  | p. 11 |
| 2.1.1 | Characters   | p. 11 |
| 2.1.2 | Locations  | p. 11 |
| 2.1.3 | Objects  | p. 11 |
| 2.1.4 | Journal System   | p. 12 |
| 2.1.5 | Quest System   | p. 12 |
| 2.2   | Incorporating GPT  | p. 13 |
| 2.2.1 | Overview   | p. 13 |
| 2.2.2 | System Instructions                                      | p. 13 |
| 2.2.3 | What to Pass Into GPT                                    | p. 13 |
| 2.3   | Using Fine-tuning to Improve the Model                   | p. 15 |
| 2.3.1 | Overview   | p. 15 |
| 2.3.2 | Creating the Fine-tuned Models                           | p. 15 |
| 2.4   | Data Collection  | p. 16 |
| 2.4.1 | Overview   | p. 16 |
| 2.4.2 | Game World Details                                       | p. 16 |
| 2.4.3 | Edge Cases   | p. 17 |
| 2.4.4 | Return Format  | p. 17 |
| 3.    | Evaluation   | p. 18 |
| 3.1   | Fine-tuning Models                                       | p. 18 |
| 3.1.1 | Model 1 - Technical Details                              | p. 18 |
| 3.1.2 | Model 1 - Testing  | p. 18 |
| 3.1.3 | Model 1 - Improvements                                   | p. 20 |
| 3.1.4 | Model 2 - Technical Details                              | p. 20 |
| 3.1.5 | Model 2 - Testing  | p. 20 |
| 3.1.6 | Model 2 - Improvements                                   | p. 23 |
| 3.2   | Game Play-through  | p. 24 |
| 3.2.1 | <i>Party Preparation Dilemma</i>                         | p. 25 |
| 3.2.2 | <i>Distant Note in a Bottle</i>                          | p. 27 |
| 3.2.3 | <i>Lawyer Quest: Pursuing Legal Knowledge</i>            | p. 29 |
| 4.    | Discussion   | p. 31 |
| 4.1   | Fine-tuning  | p. 32 |
| 4.1.1 | Training Dataset   | p. 32 |
| 4.1.2 | Incorporating Details of the Game World                  | p. 32 |

|       |   |       |
|-------|---|-------|
| 4.1.3 | Repetition of Output                              | p. 32 |
| 4.1.4 | Changing Fine-tuning Parameters                   | p. 33 |
| 4.2   | Technical Considerations                          | p. 34 |
| 4.2.1 | Tokenisation                                      | p. 34 |
| 4.2.2 | Dependency on OpenAI API                          | p. 34 |
| 4.2.3 | Error Handling                                    | p. 34 |
| 4.3   | Immersion and Interest                            | p. 35 |
| 4.3.1 | Importance of Engaging Quests                     | p. 35 |
| 5.    | Future Avenues                                    | p. 36 |
| 5.1   | Additional Quest Types                            | p. 36 |
| 5.2   | Incorporating Current World State                 | p. 36 |
| 5.3   | Synthesising World Information                    | p. 36 |
| 5.4   | Procedural Generation and Altering the Game World | p. 37 |
| 5.5   | Combining with Other A.I. Tools                   | p. 37 |
| 6.    | Conclusion  | p. 38 |
| 7.    | References  | p. 39 |

---

## 1. Introduction

### 1.1 Motivation

There were two initial motivations for this project. The first originated from an interest in exploring the use of large language models (LLMs) in generating storylines given player prompts and analysing how creative and coherent the output can be. The second stemmed from the possibility of an ‘infinite game’ - a game that continuously evolves to the player’s actions and in-game behaviours. Artificial intelligence (A.I.) has the potential to play a pivotal role in enhancing and building new quests, generating novel gameplay customised to player behaviour. The term ‘quest’ in this context means a goal, or set of goals, to accomplish in a game.

The original example for this was the popular RPG game *The Elder Scrolls: Skyrim*, developed by Bethesda. The game is composed of a mixture of main and side quests that the player must complete in order to advance in the game’s storyline. A subset of the side quests are randomly generated and follow specific templates (for example: find object *x* in location *y*). While these quests allow for virtually limitless gameplay as the player can initiate new variations when desired, they often become monotonous and ultimately inconsequential to the overall gameplay. How can LLMs be used to extend these quests to render them more engaging and compelling? How can data about the player be utilised to craft new quests and provide an increased sense of player agency? The potential of A.I. to generate unique quests is undoubtedly an exciting prospect in the gaming space.

### 1.2 Research Question

This paper investigates how to use a GPT model to dynamically generate quests in the context of an open-world, role-playing game. The project described in this paper aims to investigate and answer the following questions:

1. In what ways can a player communicate with a GPT model through the game they are playing?
2. How can the GPT model maximise its knowledge about the game world?
3. How can an architecture be designed such that it analyses the model’s responses and uses that output to dynamically generate quests?
4. How can the quest details that are returned by the model be programatically generated in the game world in an actionable and playable way?

As such, this paper puts forward an architecture that enables communication between a fine-tuned GPT model and the game environment such that the model designs executable quests tailored to player input. More specifically, the architecture is designed as follows:

1. The GPT-3.5 model is fine-tuned accordingly in order to customise it to its respective game world.
2. The architecture then facilitates communication between player input and the fine-tuned model, where the latter outputs a quest series in response to the former.
3. Finally, the architecture processes the model results to actualise the quest series within the game world, allowing the player to pursue this mission tailored to their original input.

In the context of this paper, these terms are defined as follows:

1. *Model* or *GPT model*: the OpenAI GPT-3.5 or fine-tuned model used to generate quests.
2. *Architecture*: the entire dynamic quest generation framework, including the game's code, the GPT model and the communication between the two.

### 1.3 Background

This project explores how large-language models (LLMs) can be used in video games to dynamically generate quests based on player input in the context of a role-playing game (RPG).

In late 2022, OpenAI released a powerful chatbot, ChatGPT, to the public. ChatGPT is effectively an interface for users to prompt and interact with a large language model called GPT-3.5, where GPT stands for Generative Pre-trained Transformer. The latest iteration of GPT is GPT-4. As explained in the article “A Short History of ChatGPT: How We Got To Where We Are Today”, written by Bernard Marr for *Forbes*, the launch of ChatGPT greatly popularised the GPT models and made them accessible to a much larger audience than ever before. The seemingly limitless potential of the chatbot felt novel, and a huge number of its applications across a multitude of sectors have emerged since then. Furthermore, its release revolutionised the A.I. sector, bringing the power of large language models to the forefront (Marr, 2023). One such industry where GPT holds immense promise is video games.

A.I. in general has already made significant strides in the industry, with multiple companies leveraging A.I. models to build tools that significantly enhance realism in games. An especially ubiquitous application is in the formation of sophisticated and intelligent non-player characters (NPCs). An example of such a company is InworldAI, who create character engines for NPCs in games. These engines harness the power of natural language and machine learning models in order to craft believable interactions with NPCs, greatly enhancing immersion for any game that employs the engines. In addition, in a 2021 article for *The Guardian*, Keith Stuart outlines recent advancements in the game A.I. space, noting how major game studios such as Sony and Electronic Arts are ramping up A.I. development as it is a powerful method to increase player agency in their games.

A number of video games leverage A.I. systems to craft engaging worlds and sophisticated NPCs. One such company mentioned in the *Guardian* article above is the developer Monolith which created the system Nemesis A.I. This system enabled enemy memory so that fights were remembered by NPCs, generating rivalries and a sense of history throughout gameplay (Stuart, 2021). This game is also mentioned in a 2023 article written by Matthew Urwin for the website *BuiltIn*. Urwin lists other examples, such as EA Sports’ *FIFA 22* that uses A.I. to improve player movements and Rockstar Games’ *Red Dead Redemption 2* that incorporates A.I. in a myriad of ways, such as having NPCs correct their own mistakes (Urwin, 2023).

ChatGPT in particular has been noted to be especially powerful in the games industry and holds the promise to greatly increase player immersion. An *AIContentfy* article written in May 2023 outlined the potential of ChatGPT to innovate this space, including the creation of believable NPCs as well as advancing player and NPC interactions through branching narratives. The article additionally delves into how GPT can be used to generate sophisticated storylines and customised quests by analysing player choices (ChatGPT in Gaming: The Future of Interaction and Storytelling, 2023). The latter trajectory is closely associated with the subject of this paper herein.

## 1.4 Literature Review

From an academic perspective, the use of A.I. in video games has been extensively researched over the last two decades. Narayek (2004) found that A.I. is an important consideration when players purchase games which underscores its longstanding relevance in the industry. He notes that game A.I. is not only concerned with optimising agents to win against the player, but also with crafting believable and immersive game worlds. This includes features such as “...deep NPC characters, automated storytelling with dynamic tension and emotion planning for the player, and so forth.”, which highlights the importance of utilising A.I. in games to enrich both the environment and the storyline (Nareyek, 2004). Yakan (2022) also shares this viewpoint, stating that A.I. in video games has a main goal of increasing player interest by fostering improved engagement in the game and creating a more vivid gaming experience.

In their 2015 paper, Safadi, Fonteneau and Ernst underline how designing autonomous systems akin to human intelligence is an increasingly challenging endeavour in the games industry, and highlight how popular game engines such as *Unity* address this challenge by allowing developers to create both realistic virtual environments and intelligent agents within the engines. This demonstrates a growing interest in leveraging A.I. in virtual environments to increase player immersion. Skinner and Walmsley (2019) echo this viewpoint, contending that A.I. techniques such as Neural Networks are contributing to the development of intelligent agents and that video games are an ideal field for the advancement of A.I. due to its lack of traditional boundaries.

Fairclough *et al.* investigated the possible research directions for A.I. in games, categorising these avenues by game genre. For RPGs specifically, they note the key challenges as being the creation of complex NPCs and ensuring coherence in storylines. This underscores one of the main challenges in promoting A.I. generated content in games, as consistency across characters and quests becomes notably more difficult to achieve. In their paper, the authors also discuss the Trinity College Dublin Game AI project that is focused on researching such avenues. One of these is the development of an interactive story engine which demonstrates that dynamically generated stories in games are a key area of interest amongst researchers in the field (Fairclough *et al.*, no date).

As OpenAI’s GPT model gained widespread popularity in the year 2023, an increasing number of researchers explored how large language models can be used to generate believable virtual agents in game environments. One such project was Park *et al.*’s. *Generative Agents: Interactive Simulacra of Human Behavior* (2023) which developed an architecture for intelligent agents that behave in a believable and human-like manner. The architecture leveraged a large language model to store and synthesise memories for each agent in the sandbox world, essentially creating a digital environment where agents organically interacted with each other and pursued their respective lives without human interference (Park *et al.*, 2023). This project clearly highlights the sheer power that large language models have both in simulating human behaviour, but also in processing vast amounts of information to produce output that acts rationally and consistently with past data. This research in particular served as a main point of inspiration for the dynamic quest generation project discussed in this paper.

## 1.5 OpenAI's API - Fine-tuning and Function Calling Overview

As the architecture defined in this paper makes use of OpenAI's fine-tuning methodology, I will briefly summarise both the fine-tuning and function calling methodologies. While the architecture described in this paper does not directly use function-calling, its potential as a future avenue for this project is explored in the Discussion section and so merits an overview. An in-depth explanation of both methodologies can be found on the OpenAI website, in the Documentation section (OpenAI Platform, no date).

As this documentation explains, fine-tuning is the process of customising a GPT model for a specific application. A dataset of at least ten examples (ideally as many as possible) is uploaded and fed into one of the existing GPT models, where each example is in the form of a Chat Completion conversation. In other words, each data point is a messages structure with the following entries:

1. A system role, where the entry is the system instructions
2. A user role, where the entry is the example user prompt
3. An assistant role, where the entry is the ideal GPT completion

With fine-tuning, a GPT model can be customised to a specific application and can grasp the technicalities such as the correct output format, output tone and how to handle edge cases. Fine-tuning also carries the benefit of reducing token count once the model is appropriately trained for the specific use case, as properly trained fine-tuned models require shorter system instructions and prompts to achieve the desired outcome compared with the default GPT model (OpenAI Platform, no date).

The documentation also explains how to use the application programming interface (API) for function calling. Function calling is a method to bridge a GPT model with an external codebase. By providing the model with the list of possible functions to use and querying it with a task, the model is able to decide which set of functions are best suited to accomplish the task at hand. It packages these function calls, complete with parameters, as a JSON object. This JSON object can then be parsed in the external codebase and the functions called accordingly (OpenAI Platform, no date). This methodology essentially provides the model with a window to the code, allowing it to suggest the ideal flow of logic to attain the desired outcome.

---

## 2. Methodology

### 2.1 Building the Game

To investigate the research question, I built a first-person role-playing game in the Unity game engine. The game is a three-dimensional (3D) game set in a small open-world. The world is broken down into three main locations: a forest, a village and a beach environment. All the code for the game was written in Unity's main language, C#.

The game's main premise is that the player creates their own adventures via journal entries. There is a built-in journal within the game that the player can write their thoughts in, which in turn generates an API call to the GPT model. The model uses the journal entry to create a relevant quest series that is then implemented in the game world, allowing the player to pursue their unique mission.

#### 2.1.1 Characters

The player takes the role of a Fox. There are three non-player characters (NPCs) present in the game world. For each NPC, I first designed their main characteristics, and prompted ChatGPT to create a two sentence summary:

1. Baker Chip (Bear): A friendly baker who enjoys making people laugh with jokes and sharing his baked goods. He also likes experimenting with new doughnut flavours.
2. Explorer Ivy (Fox): Nature enthusiast who explores forests and beaches, passionate about uncovering natural mysteries and eager to share her discoveries with others.
3. Bean (Rabbit): Coffee aficionado, passionate about coffee and coffee art. Enjoys writing a novel during off-hours.

All of the characters were modelled in Blender. They were then imported into Adobe's Mixamo software for rigging and animation, and finally imported into Unity to use as game assets.

#### 2.1.2 Locations

The game's environment was built using Unity's terrain tools, and divided into three areas: the village, the forest and the beach. These areas were sectioned off using box colliders and sized accordingly, such that each area had a specific boundary that could be understood by the game's code. These three locations are taken into account by GPT.

3D assets were used to add detail to the world and to create sections such as a marketplace, town square and neighbourhood. These sections are purely for decorative purposes and are not taken into account by the GPT model.

The majority of the game assets used for the environment design and objects were downloaded from the Unity Asset Store.

#### 2.1.3 Objects

As one of the three quest types is to search for objects in the game world, an inventory system for all potential objects to find was implemented in the codebase. In order to maximise the narrative

potential for object-finding quests, flexible objects were chosen such as a bottle, various fruits, a cube, a treasure chest, etc. This enables the GPT model to easily adapt these objects to different stories and situations as it sees fit.

#### 2.1.4 Journal System

The game's journal system allows the player to record their thoughts in the game interface. Each journal entry gets stored in a list of entries, and creates an API call to the GPT model. The player can access the journal via a button on the screen, where they can record a new entry or scroll through previous ones. Each journal entry constitutes a user prompt to the GPT model.

#### 2.1.5 Quest System

To explore the topic of dynamic quest generation, it is essential to build out a modular quest system that can easily be extended to include various quest types. This also allows the GPT model to better understand the main technical differences between each quest type.

The game uses *quest series* rather than individual quest objects. Each quest series has a title, description and list of *quest steps*. Each quest step is a singular, actionable quest that the player has to complete.

Each quest step is one of the following:

1. Collect coins (COLLECT\_COINS): the player must collect a specified amount of coins
2. Talk to an NPC (TALK\_NPC): the player must engage in conversation with a specified NPC
3. Find object (FIND\_OBJECT): the player must find the specified object within the game world

As a result, quest series can be composed of a set of the same quest type or a mixture of different quest types. Completing each quest step in the series completes the series. The user interface (UI) indicates this by blurring out the quest steps and marking the quest as completed, serving as visual cues for the player.

From a code standpoint, the overall quest system in the game is broadly encapsulated by the following logic:

1. A quest series manager that stores incoming quest series and manages the state of each series (active, inactive or completed).
2. Individual quest classes that encapsulate the logic for each of the three quest types.
3. A JSON parser that reads in a quest series represented by a JSON object, and create an instance of the quest series class. This process iterates over each quest step for the series and deserialises them into individual instances of their respective quest type.

Completing a quest series may reward the player with coins, the amount of which is also returned by the model.

## 2.2 Incorporating GPT

### 2.2.1 Overview

OpenAI's GPT-3.5 Turbo model was incorporated into the Unity game using the OpenAI API. A GPT manager was created in the codebase to make and receive these API calls, and is responsible for communicating directly with the relevant journal and quest-related scripts to send instructions to the GPT model and parse the JSON output.

The GPT model was tasked with taking in player input (in the form of a journal prompt) and then creating and returning the quest series as a specific JSON object. The following fields are mandatory:

1. Series Title: the title of the quest series
2. Series: a short 1-2 sentence description
3. Series Steps: an array of individual quest objects, in order of completion

Each quest step is represented by its own JSON object, including mandatory fields such as Title, Quest Type and Description. Depending on the quest type, the corresponding JSON object will include relevant fields to the type:

1. COLLECT\_COINS: Add number of coins
2. TALK\_NPC: Add NPC name and dialogue
3. FIND\_OBJECT: Add object to find and location to search in

### 2.2.2 System Instructions

The initial step of providing system instructions to the GPT model is crucial for constructing a quest generation architecture that is tailored to the game world. This step sets the model up to understand its role in the game, the input that will be passed in and the exact output that it should produce. Additionally, NPC names, available 3D assets and world locations are programatically included at the end of the instructions. The addition of these values is important as it gives the model more context on the game world and offers parameter options for the various quest types. The automated addition of these values ensures that the system instructions are updated dynamically which in turn enhances overall system flexibility.

### 2.2.3 What to Pass Into GPT

Apart from the system instructions, the GPT API requires a user prompt. In this architecture, this is the direct user input, i.e. the journal entry. A messages list is created in the code that initially contains the system instructions. As the player inputs journal entries, these get created as 'User' role messages and appended to the messages list. This list gets sent to the model via the API, and the first result returned gets parsed by the code.

The first result is also a critical part of the messages history. When the GPT model responds to the API request, a new assistant message is created where the role is 'Assistant' and the corresponding message is the first result returned. This message then gets added to the messages list. With this ever expanding history, each API call has a comprehensive list of journal entries and quest series

that were previously returned by the model. This provides the GPT model with a record of all player inputs and quests that they have to complete.

## 2.3 Using Fine-tuning to Improve the Model

### 2.3.1 Overview

A key investigation in this project was how utilising OpenAI's fine-tuning mechanism enhances the dynamic quest generation framework through the creation of a GPT model that is truly customised to the game world. Two fine-tuned models were created for this project.

As explained in the Introduction, fine-tuning enables further training on a GPT model using a dataset customised to a specific goal. As a result, a properly fine-tuned model requires less example prompts to achieve the desired outcome, thus streamlining the model prompting process and lowering monetary costs. In a similar vein, the system instructions can also be refined accordingly. Additionally, fine-tuning provides the model with a more complete picture of the game world that would otherwise be challenging to do via the system instructions and few-shot prompting.

After testing in OpenAI's playground, a number of limitations were found within the first model which are discussed in the Evaluation section. These were addressed either through additional examples in the dataset or changes to existing examples, and this updated dataset was used to train the second fine-tuned model.

### 2.3.2 Creating the Fine-tuned Models

The following steps were taken to create the fine-tuned models:

1. Compile a list of required examples to cover the game world's detail and edge cases.
2. Use ChatGPT to create a user prompt and assistant response, where the user prompt is player input (the journal entry) and the assistant response is the correctly formatted quest series.
3. A variation of the OpenAI Cookbook's *Data preparation and analysis for chat model fine-tuning* (Wu and Fishman, 2023) script was used to validate the data set and format it correctly prior to fine-tuning.
4. A file of the training data was created and uploaded to OpenAI.
5. The fine-tuning job is then created with the corresponding file object, using either the OpenAI API or directly on OpenAI's website. The GPT model that is fine-tuned is the gpt-3.5-turbo model.
6. The fine-tuned model can be tested in the OpenAI playground and/or referenced directly in the Unity editor for use in the game.

## 2.4 Data Collection

Ideally, the comprehensive dataset used to train the fine-tuned model should contain a sufficient number of examples that cover the different facets of the game world, as to provide the model with a complete depiction of the game's nuances. If the dataset paints an accurate picture, the model will have an easier time creating quests that are cohesive and relevant to the game.

In order to successfully fine-tune GPT for this use case, there are two main objectives:

1. Getting the model to understand the game world
2. Getting the model to return a quest series object in the correct JSON format

### 2.4.1 Overview

I used ChatGPT to streamline the data collection process. Firstly, I provided ChatGPT with a set of comprehensive instructions to set its role and inform it of the required output format. To create the dataset used for fine-tuning, I then prompted ChatGPT to create both a journal entry and a quest series that matches a specific goal, such as getting to know a NPC. ChatGPT creates both the user entry and the associated quest series in the correct JSON format. To ensure both accuracy and cohesion with the game world, I manually went through each result to correct any part of the output that did not make sense in the context of the game.

The final dataset consisted of 48 examples.

### 2.4.2 Game World Details

As mentioned above, a significant advantage of fine-tuning is that it allows the model to understand variability in different game world parameters that would be challenging to portray otherwise. In this game, there are several parameters that are important to train on:

1. The overall mood of the game, i.e. a forest-based, relaxed game
2. The tone of the quest series
3. NPC names, species and characteristics
4. Potential objects to look for
5. Different parts of the game world (locations)

By providing examples of journal entries/quest series combinations that cover these different parameters, the model will be more adept at creating similar quests in real time.

A comprehensive list of different parameters to train on was compiled, with the main categories consisting of several examples each. The following examples are just a subset of the total list:

1. Example for getting to know Chip - that he's a baker
2. Example for getting to know Chip - that he likes to experiment with doughnut flavours
3. Example for getting to know Ivy - that she's an explorer
4. Example for getting to know Bean - that he's a coffee lover
5. Example for finding an object of type BOTTLE - find a bottle of milk
6. Example for finding an object of type BOTTLE - bring a bottle of wine to a party

Each of these examples would have a journal entry indicating some desire to accomplish the goal (example: “*I should get to know Chip better*”) and the corresponding quest series that explores that goal.

### 2.4.3 Edge Cases

Another important consideration is the unpredictability of player input. If the user input is a clear, concise thought that fits easily into the game (for example: “*I’d love to explore the forest today!*”) then it should be easier for the model to generate quests that address the journal entry, are contextually relevant to the game world, and technically feasible for the architecture to implement in the actual game environment.

However, the player may input something that is gibberish (for example: “*aekajdshkjasdhf*”) or makes little sense in the context of the game (for example: “*I can’t wait to go race cars*”). In these cases, the model will struggle to return something that is actionable in the world, thus causing the game’s code to either reject the quest series or accept it despite the quest being technically impossible to implement in the game. In order to address these edge cases, I added similar examples to the above to the training set. The aim is to instruct the model how to behave in such situations in order to enhance its overall robustness.

Furthermore, as the quests are unknown during development, it is impossible to prepare 3D assets for everything that could be the object of a FIND\_OBJECT quest. In these cases, setting a default asset safeguards against the model returning quests with an undefined object. For example, a quest that asks the player to find an umbrella when it is not a viable game asset would be impossible for the architecture to technically execute. A certain level of creativity is necessary here, and the model should return something that is feasible within the context of the game world. For instance, it could instruct the player to find the object CUBE and rationalise it with the following description: “A special cube can shelter you from the rain”. Although this does not make as much sense as using an umbrella, it is both technically possible in the game and justified to the player.

### 2.4.4 Return Format

The additional condition that is critical to train on is the format of the output. Providing the model with training data that consistently follows the same JSON structure drastically increases the chances of the correct format being returned in the game.

The JSON result will be different based on which types of quests are included in the series. The three quest types (COLLECT\_COINS, TALK\_TO\_NPC and FIND\_OBJECT) each have different fields in their respective JSON representations. The training dataset should be comprised of a number of these different quest types, so that the model forms a solid understanding of the variations between the corresponding JSON objects.

## 3. Evaluation

### 3.1 Fine-tuning Models

The fine-tuned models were tested in OpenAI's playground before implementing it in the Unity game.

#### 3.1.1 Model 1 - Technical Details

The first fine-tuned model was based off of the architecture of gpt-3.5-turbo-0613. The initial dataset was comprised of 45,303 tokens and the total number of training epochs was 3. The training loss of the model was 0.3064.

#### 3.1.2 Model 1 - Testing

Testing the first fine-tuned model quickly revealed three main shortfalls with the new model:

1. NPC details are incorrect and incoherent with the game world



USER I wonder what the story behind the village is

ASSISTANT `{"SERIES_TITLE": "The Tale of a Village", "SERIES_DESC": "Embark on a journey to uncover the rich history and stories of a vibrant village deep within the forest.", "SERIES_STEPS": [{"QUEST_TYPE": "TALK_NPC", "QUEST_TITLE": "Converse with Ivy", "QUEST_DESC": "Seek out Ivy, the wise owl, and unravel the mysterious tale of the village's founding.", "NPC_NAME": "IVY", "DIALOGUE": [{"Tag": "FOX", "Speech": "Ivy, tell me the story of our village. How did it come to be?"}, {"Tag": "IVY", "Speech": "Ah, the village's story is ancient. Legend has it that it all began with a lost treasure..."}]}, {"QUEST_TYPE": "TALK_NPC", "QUEST_TITLE": "Bean's Recollections", "QUEST_DESC": "Discover more about the village's past by chatting with Bean, the adventurous squirrel.", "NPC_NAME": "BEAN", "DIALOGUE": [{"Tag": "FOX", "Speech": "Hey, Bean! Ivy told me about the lost treasure. Do you know anything more?"}]}]`

+ Add message

**Figure 1** - First fine-tuned model: playground example which shows incorrect information about Bean

The example in Figure 1 shows a quest that is technically possible; the NPC tags are defined values, the quest types exist, and both the whole quest series object and each quest step object follow its defined JSON format. However, the yellow highlight shows Ivy incorrectly described as an owl and Bean as a squirrel. This makes it confusing for the player, who may be searching the entire world

for a pair of non-existent woodland animals, which consequently breaks the immersion and cohesion of the quest series.

2. The quest series are either incomplete and/or inconsequential

The screenshot shows a user interface for generating quest series. On the left, there's a 'USER' section with a text input field containing the message: "I wonder what the story behind the village is". To the right, under 'ASSISTANT', is the generated JSON output:

```
"{\"SERIES_TITLE\": \"The Village Chronicles\", \"SERIES_DESC\": \"Uncover the secrets of the village through its history, culture, and the tales of its inhabitants.\", \"SERIES_STEPS\": [{\"QUEST_TYPE\": \"TALK_NPC\", \"QUEST_TITLE\": \"Meet Bean, the Village Historian\", \"QUEST_DESC\": \"Head to the village and seek out Bean, the resident expert on its history. Listen to Bean's stories and start unraveling the village's past.\", \"NPC_NAME\": \"BEAN\", \"DIALOGUE\": [{\"Tag\": \"FOX\", \"Speech\": \"Bean, I'm curious about the story behind the village. Could you share some insights?\"}, {\"Tag\": \"BEAN\", \"Speech\": \"Ah, the village has a rich history, my friend. Let me take you on a journey through time.\"}]}]}
```

Below the JSON output, there's a button labeled "+ Add message" with a plus sign icon.

**Figure 2** - First fine-tuned model: playground example showing an incomplete quest

The same user prompt generated a different quest series as seen in Figure 2. Again, this generated quest series is technically doable from a code standpoint. However, the content is both inaccurate - Bean is not defined as an expert on the town's history as seen in the yellow highlight - and incomplete; it does not provide Bean's dialogue for explaining the town's history (green highlight). As a result, the returned quest series does not actually achieve the objective of the series description and ultimately feels insubstantial.

3. The quest objectives and details can be nonsensical

|           |  |
|-----------|--|
| USER      | I'm feeling lonely these days...   |
| ASSISTANT | <pre>{   "SERIES_TITLE": "Finding Connection in Nature",   "SERIES_DESC": "Embark on a journey of self-discovery and find solace and companionship in the serene beauty of the forest.",   "SERIES_STEPS": [     {       "QUEST_TYPE": "FIND_OBJECT",       "QUEST_TITLE": "Seeking Serenity at the Beach",       "QUEST_DESC": "Head to the beach, a place of tranquility and reflection. Find solace in the calming sound of the waves and the gentle embrace of nature.",       "OBJECT": "WATERMELON",       "LOCATION": "BEACH"     },     {       "QUEST_TYPE": "TALK_NPC",       "QUEST_TITLE": "Contemplation with Ivy",       "QUEST_DESC": "Talk to Ivy, the forest wanderer, and share your feelings of loneliness. Discover the wisdom she holds and the connection she finds in nature.",       "NPC_NAME": "IVY",       "DIALOGUE": [         {           "Tag": "FOX",           "Speech": "Ivy, I've been feeling lonely lately. How do you find peace in the forest?"         },         {           "Tag": "IVY",           "Speech": "Loneliness can be healed by nature's embrace, Fox. Let's find solace amidst the trees."         }       ]     }   ] }</pre> |

**Figure 3** - First fine-tuned model: playground example showing nonsensical quest details

As can be seen in the example in Figure 3, the model can return quests that are essentially nonsense. The first quest step, highlighted in yellow, *Seeking Serenity at the Beach*, has a quest description that is not aligned to the quest objective which is to search for a watermelon (highlighted in green). The actionable component of the quest makes no sense in the context of the journal entry and quest description.

### 3.1.3 Model 1 - Improvements

It is important to note however that the fine-tuned model indicates an immediate improvement over the generic gpt-3.5-turbo model; it uses streamlined instructions, returns quest series in the correct JSON format, and populates quest information with valid game information such NPC names, locations and objects.

The second fine-tuned model is designed to mitigate the limitations described above and produce quest series that are both more accurate with regards to the designed world and more holistically complete as a story. As a result, I augmented the original dataset with examples that provided increased details about the characters and the game world. I additionally ensured that each quest was complete such that there was no open-ended dialogue and the final quest step provided a natural end to the series.

### 3.1.4 Model 2 - Technical Details

The second fine-tuned model was based off of **gpt-3.5-turbo-1106**. The new dataset used for training was made up of 84,618 tokens. The training was done over 3 epochs and had a training loss of 0.4377.

### 3.1.5 Model 2 - Testing

The initial testing for model 2 was also performed in OpenAI's playground environment. The same system instructions were provided to the model, and similar journal entries were passed in as user

prompts. Below are two examples that illustrate both general improvements of the new model as well as similar limitations.

## 1. Mismatch between technical details and quest descriptions

|           |  |
|-----------|--|
| USER      | I think there's a secret in the forest...  |
| ASSISTANT | <pre>"\"SERIES_TITLE\": \"Enigma of the Forest: Unveiling Nature's Secrets\", \"SERIES_DESC\": \"Satisfy your curiosity and delve into the mysteries of the forest. Journey with Fox to uncover the hidden secrets and unravel the enigmas that nature has woven within the woodland.\", \"SERIES_STEPS\": [   {\"QUEST_TYPE\": \"FIND_OBJECT\", \"QUEST_TITLE\": \"Whispers Among the Trees\", \"QUEST_DESC\": \"Listen carefully to the whispers of the ancient trees. They might hold clues to the secret nestled within the heart of the forest.\", \"OBJECT\": \"CHEESE\", \"LOCATION\": \"FOREST\"},   {\"QUEST_TYPE\": \"TALK_NPC\", \"QUEST_TITLE\": \"Seeking Guidance from Ivy\", \"QUEST_DESC\": \"Share the forest's whispers with Ivy, the wise librarian, and seek her guidance. Ivy's knowledge extends to the depths of the forest's mysteries.\", \"NPC_NAME\": \"IVY\", \"DIALOGUE\": [{\"Tag\": \"FOX\", \"Speech\": \"Ivy, the forest seems to be hiding a secret. I discovered a clue, but I'm unsure where to look next.\"}, {\"Tag\": \"IVY\", \"Speech\": \"The forest harbors many secrets, Fox. Let this clue guide you, and the truth will be revealed.\"]}], {\"QUEST_TYPE\": \"FIND_OBJECT\", \"QUEST_TITLE\": \"Lost Journal of Explorers\", \"QUEST_DESC\": \"Embark on a quest to find the lost journal of past explorers. Their insights might shed light on the forest's secret and lead you deeper into the mystery.\", \"OBJECT\": \"BOTTLE\", \"LOCATION\": \"FOREST\"}, {\"QUEST_TYPE\": \"TALK_NPC\", \"QUEST_TITLE\": \"Deciphering the Journal\", \"QUEST_DESC\": \"Thoroughly examine the recovered journal with Ivy and decipher the cryptic messages within. The secrets of the forest are often hidden in plain sight.\", \"NPC_NAME\": \"IVY\", \"DIALOGUE\": [{\"Tag\": \"FOX\", \"Speech\": \"Ivy, I found the lost journal of explorers. Its pages are filled with riddles. Can you help me decipher them?\"}, {\"Tag\": \"IVY\", \"Speech\": \"Let's unravel the mysteries together, Fox. The forest's secrets will soon be revealed.\"]}], {\"QUEST_TYPE\": \"TALK_NPC\", \"QUEST_TITLE\": \"Revealing the Forest's Secret\", \"QUEST_DESC\": \"Armed with the knowledge from the journal, return to the heart of the forest, and unlock the long-hidden secret that has intrigued you. Nature's enigmas await your discovery.\", \"NPC_NAME\": \"IVY\", \"DIALOGUE\": [{\"Tag\": \"FOX\", \"Speech\": \"Ivy, the journal's clues led me deeper into the forest. I believe I'm on the verge of unveiling its long-hidden secret.\"}, {\"Tag\": \"IVY\", \"Speech\": \"Continue your journey, Fox. The forest's secret is a gift to those who seek its mysteries.\"]}], {\"QUEST_TYPE\": \"TALK_NPC\", \"QUEST_TITLE\": \"Celebrating the Revelation\", \"QUEST_DESC\": \"Celebrate the revelation of the forest's secret with Ivy. Your quest for knowledge has forged a deeper connection with the natural world.\", \"NPC_NAME\": \"IVY\", \"DIALOGUE\": [{\"Tag\": \"FOX\", \"Speech\": \"Ivy, I've uncovered the secret of the forest! It's a tale as old as time but hidden from view.\"}, {\"Tag\": \"IVY\", \"Speech\": \"Well done, Fox! The forest's mysteries are a testament to the wonders of nature.\"]}]}</pre> |

**Figure 4** - Second fine-tuned model: playground example

This shortcoming is similar to the one of the first fine-tuned model. In Figure 4, the quest step highlighted in yellow demonstrates a mismatch between the quests description of listening to the

trees with the practical task of searching for cheese. Furthermore, in the second quest step (highlighted in green), the description instructs a user to find the ‘journal of past explorers’. However, this quest is interpreted by the codebase as a FIND\_OBJECT quest with the object type BOTTLE. This is obviously misleading to the player, who may be searching for a lost journal in the game world only to find none.

## 2. Misleading quest instructions

In a similar vein, the series above in Figure 4 presents misleading instructions in a subsequent quest step (highlighted in blue). The description instructs the player to return to the forest to reveal its secrets, when in actuality it expects the player to engage in conversation with Ivy. Again, this can render the quest frustrating from a player perspective as it turns the experience of pursuing a quest into a game of trial and error.

|                  |  |
|------------------|--|
| <b>USER</b>      | Chip seems upset lately  |
| <b>ASSISTANT</b> | <pre>"\"SERIES_TITLE\": \"Troubled Trails: Unraveling Chip's Dilemma\", \"SERIES_DESC\": \"Join Fox in a quest to understand and alleviate Chip's distress. Explore the forest to find the cause of Chip's troubles and offer your fox wisdom.\", \"SERIES_STEPS\": [{\"QUEST_TYPE\": \"TALK_NPC\", \"QUEST_TITLE\": \"Concern for Chip\", \"QUEST_DESC\": \"Express your concern to Bean about Chip's recent demeanor. Find out if Bean knows what's been bothering Chip.\", \"NPC_NAME\": \"BEAN\", \"DIALOGUE\": [{\"Tag\": \"FOX\", \"Speech\": \"Bean, I've noticed that Chip seems upset lately. Do you know what's been troubling him?\", \"Tag\": \"BEAN\", \"Speech\": \"Indeed, Fox. He's been searching for something precious. Maybe a quest to help him would lift his spirits.\"}, {\"QUEST_TYPE\": \"FIND_OBJECT\", \"QUEST_TITLE\": \"Discovering Chip's Treasure Hunt\", \"QUEST_DESC\": \"Explore the forest to find a clue about the treasure Chip seeks. Perhaps, this 'treasure' holds the key to his happiness.\", \"OBJECT\": \"CUBE_A\", \"LOCATION\": \"FOREST\", \"QUEST_TYPE\": \"TALK_NPC\", \"QUEST_TITLE\": \"Revealing the Treasure Clue to Chip\", \"QUEST_DESC\": \"Once you've found the clue, share it with Chip and embark on a treasure hunt together. This quest might bring much-needed joy to his heart.\", \"NPC_NAME\": \"CHIP\", \"DIALOGUE\": [{\"Tag\": \"FOX\", \"Speech\": \"Chip, I've found a clue about the treasure you seek. Let's solve this mystery together and bring a smile to your face.\", \"Tag\": \"CHIP\", \"Speech\": \"You're a true friend, Fox. Let's uncover the treasure and lift the weight off my heart.\"}, {\"QUEST_TYPE\": \"TALK_NPC\", \"QUEST_TITLE\": \"Joy of Discovery\", \"QUEST_DESC\": \"Celebrate the discovery with Chip and offer him words of encouragement. A friend's support can brighten even the gloomiest days.\", \"NPC_NAME\": \"CHIP\", \"DIALOGUE\": [{\"Tag\": \"FOX\", \"Speech\": \"Chip, the treasure hunt was a success! I'm glad to see you smiling again.\", \"Tag\": \"CHIP\", \"Speech\": \"Thanks, Fox. Your friendship and help mean the world to me.\"]}]}}"</pre> |

**Figure 5** - Second fine-tuned model: playground example showing misleading instructions

This limitation can also be observed in the quest series shown in Figure 5, albeit to a lesser extent. In the quest step highlighted yellow, we can see that the player is expected to search for treasure in the forest. However, the object to search for is the default cube (CUBE\_A). This is a good example of when the quest object (treasure) does not exist as a game asset and the model falls back to its default object. It would have been better for the quest description to acknowledge that the object is just a cube, and say something along the lines of: “*Search for a cube that holds the treasure of the forest*”. This would keep the quest description and its technical details in the game aligned and enhance the overall player experience through consistency in the quest instructions.

### 3.1.6 Model 2 - Improvements

While the quest series are technically accurate from a code standpoint, and more involved than the output returned by the first model, there are still limitations that should be addressed for future iterations. There are clearly improvements to make towards clarifying the quest instructions and creating storylines that are both compelling and substantial.

However, the second fine-tuned model produces output that is contextually relevant to the game world and style, as well as technically attainable in the codebase. The quest series are each customised to the user prompt and playable. As a result, the second fine-tuned model was incorporated in Unity to use in the game environment.

### 3.2 Game Play-through

Another critical component to examine is the full pipeline of the architecture, from the ease of inputting a journal entry during gameplay to the technical feasibility of the model's output in the actual game environment. From a technical standpoint, the overall architecture must be able to:

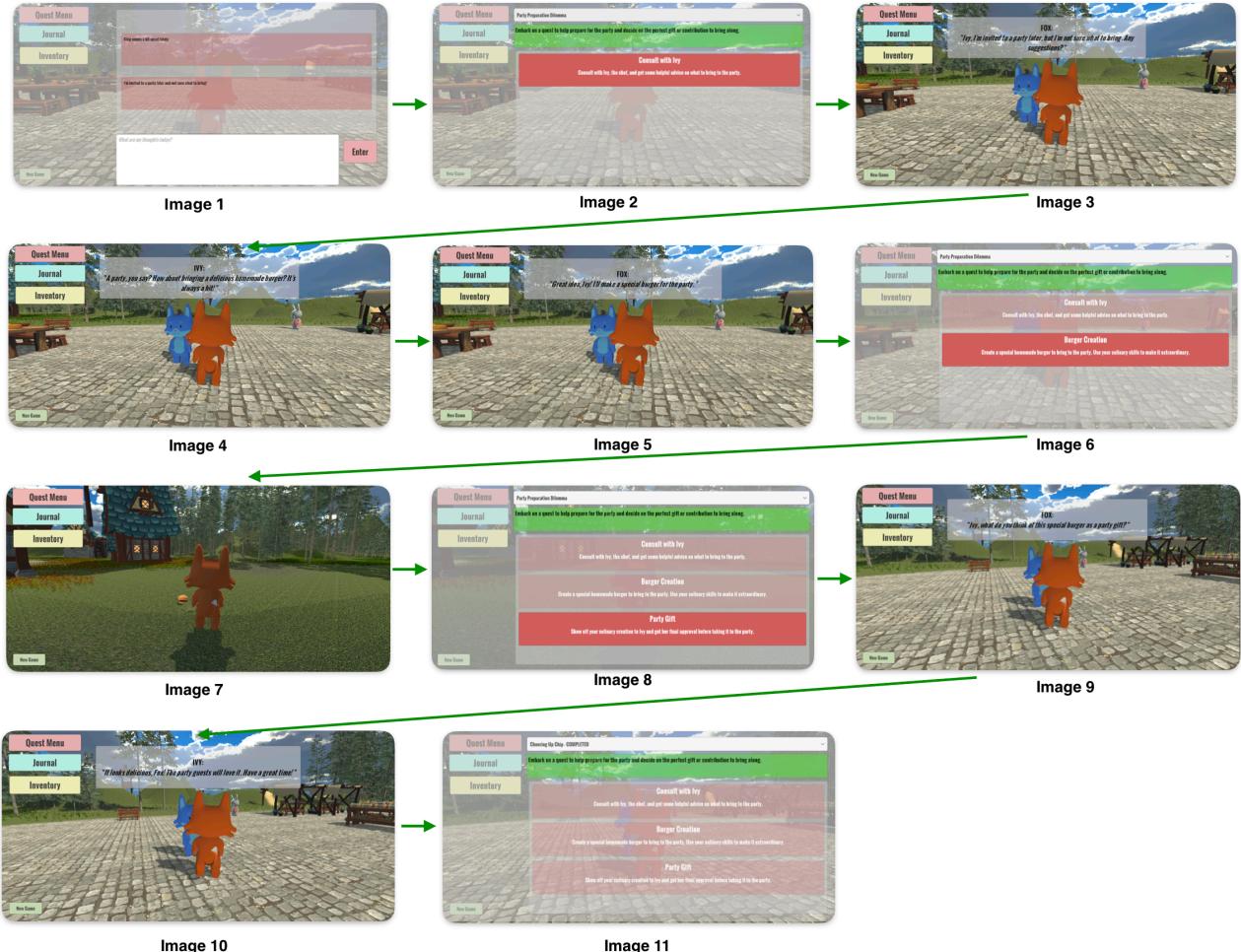
1. Allow the user to input a journal entry.
2. Package the system instructions and journal entry into system and user prompts.
3. Send this packaged input to the model.
4. Receive the model's output.
5. Deserialise the JSON object to create the corresponding quest series where each of the quest steps have also been correctly initialised as instances of their respective quest type.
6. Have the quest manager either start or queue the quest series.
7. When the quest series begins, have the quest manager correctly handle each quest step and handle any processing steps needed (such as releasing objects from the pool, storing dialogue, etc).
8. Complete the quest series when each step has been accomplished and display this to the player accordingly.

To test this, the second fine-tuned model referenced directly in the Unity game editor.

The following section will provide three in-game examples: *Party Preparation Dilemma*, *Distant Note in a Bottle* and *Lawyer Quest: Pursuing Legal Knowledge*. These highlight how the architecture is successfully able to achieve the goals described above and produce playable quest series in the 3D game environment. The screenshots were taken from a game play-through, from journal entry to quest completion, and demonstrate the player's perspective.

### 3.2.1 Party Preparation Dilemma

In this quest, the player expresses uncertainty over what to bring to a party.



**Figure 6** - series of screenshots showing the quest *Party Preparation Dilemma*

The series demonstrated in Figure 6 is made up of the following elements:

- [Image 1] The entry "*I'm invited to a party later and not sure what to bring!*" is added to the journal. This generates an API call to the fine-tuned GPT model.
- [Image 2] The model responds with the quest series *Party Preparation Dilemma*, which is added to the quest menu.
- [Image 3] The first step *Consult with Ivy* is activated, where the player must approach Ivy and get advice. This is a TALK\_NPC quest.
- [Image 4-5] Both sides of the conversation are provided by the model. Ivy recommends "... *bringing a delicious homemade burger!*" to which Fox (the player) agrees. The quest step is then completed once the conversation ends.
- [Image 6] The next quest step, *Burger Creation*, begins and instructs the player to make a burger to bring to the party. This is a FIND\_OBJECT quest.

6. [Image 7] The player must look for a burger that has been created in the game world. Once this has been found the quest step completes.
7. [Image 8] The next step, *Party Gift*, begins. The player must re-engage in discussion with Ivy to get her approval of the burger. This is a TALK\_NPC quest.
8. [Image 9-10] Ivy approves and claims that “...it looks delicious!” Once the conversation comes to an end, the quest step completes.
9. [Image 11] As the player has finished each quest step, the quest series completes. This is indicated in the quest menu accordingly.

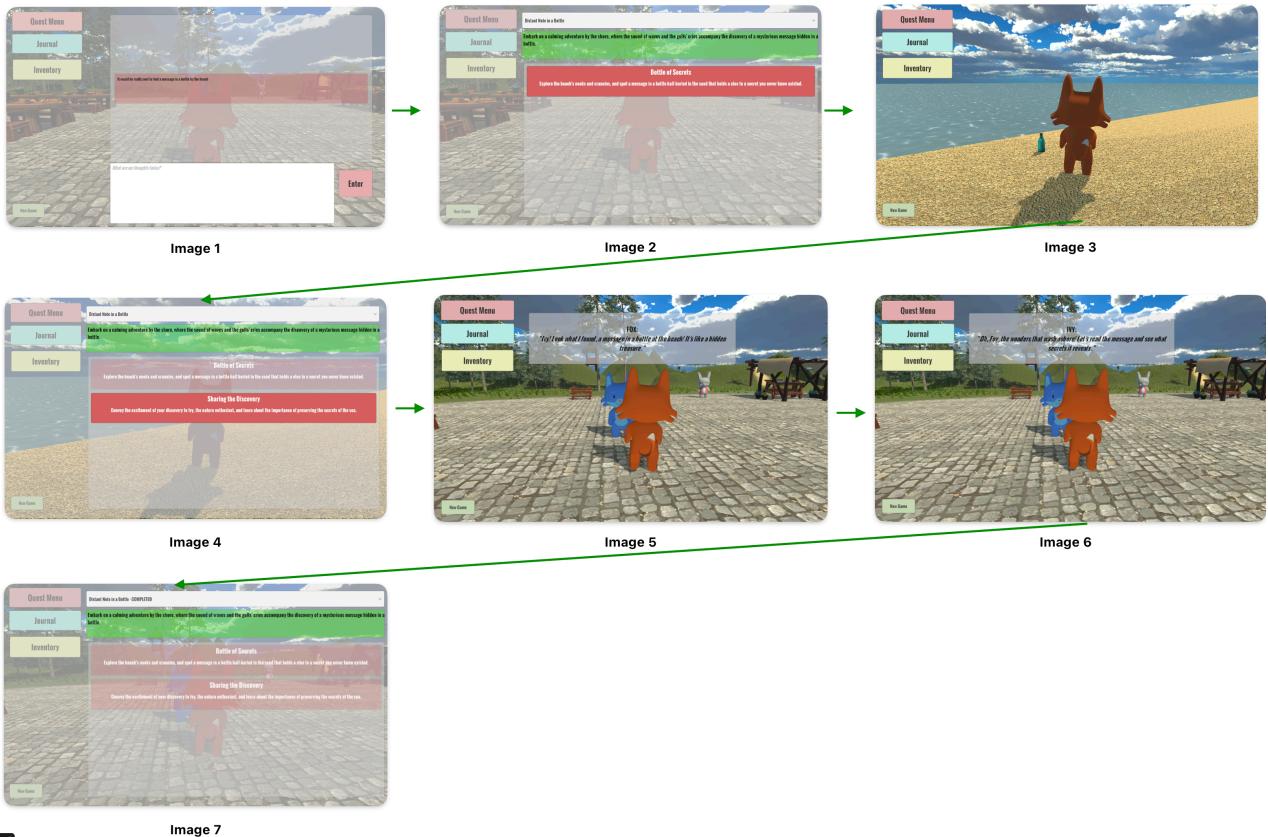
The limitations with the content of the series have already been discussed above in section 3.1.5. The same model was used to produce this series. From a technical standpoint however, the example highlights how the architecture is able to go from player input to a playable quest series successfully. Each quest step, albeit with misleading instructions at times, is playable and consistent with the rest of the series.

For the FIND\_OBJECT quest (*Burger Creation*, step 6), there are two important considerations to address. The first is that the model built the quest series around a burger, a viable game asset according to the system instructions. This indicates that the model was able to creatively devise a legitimate quest within the game’s specified constraints. The second is that the correct game asset of the 3D burger was successfully instantiated in the virtual world. These two considerations underline the effective communication and harmony between the model’s output and the code architecture.

These observations equally apply to the two TALK\_NPC quests, which instruct the user to speak to Ivy (*Consult with Ivy*, *Party Gift*, steps 3-4 and 7-8). The code architecture correctly assigns the dialogue of the quest to the character with the tag IVY. In addition, speaking twice to Ivy rather than once to two separate character adds continuity to the quest series, demonstrating a level of progression in the model’s output and keeping the player engaged through increased cohesion in the quest storyline.

### 3.2.2 Distant Note in a Bottle

This quest series is relatively short, but highlights the power of the fine-tuned model in generating quest series that take place in contextually relevant locations.



**Figure 7** - series of screenshots showing the quest *Distant Note in a Bottle*

The series demonstrated in Figure 7 is made up of the following elements:

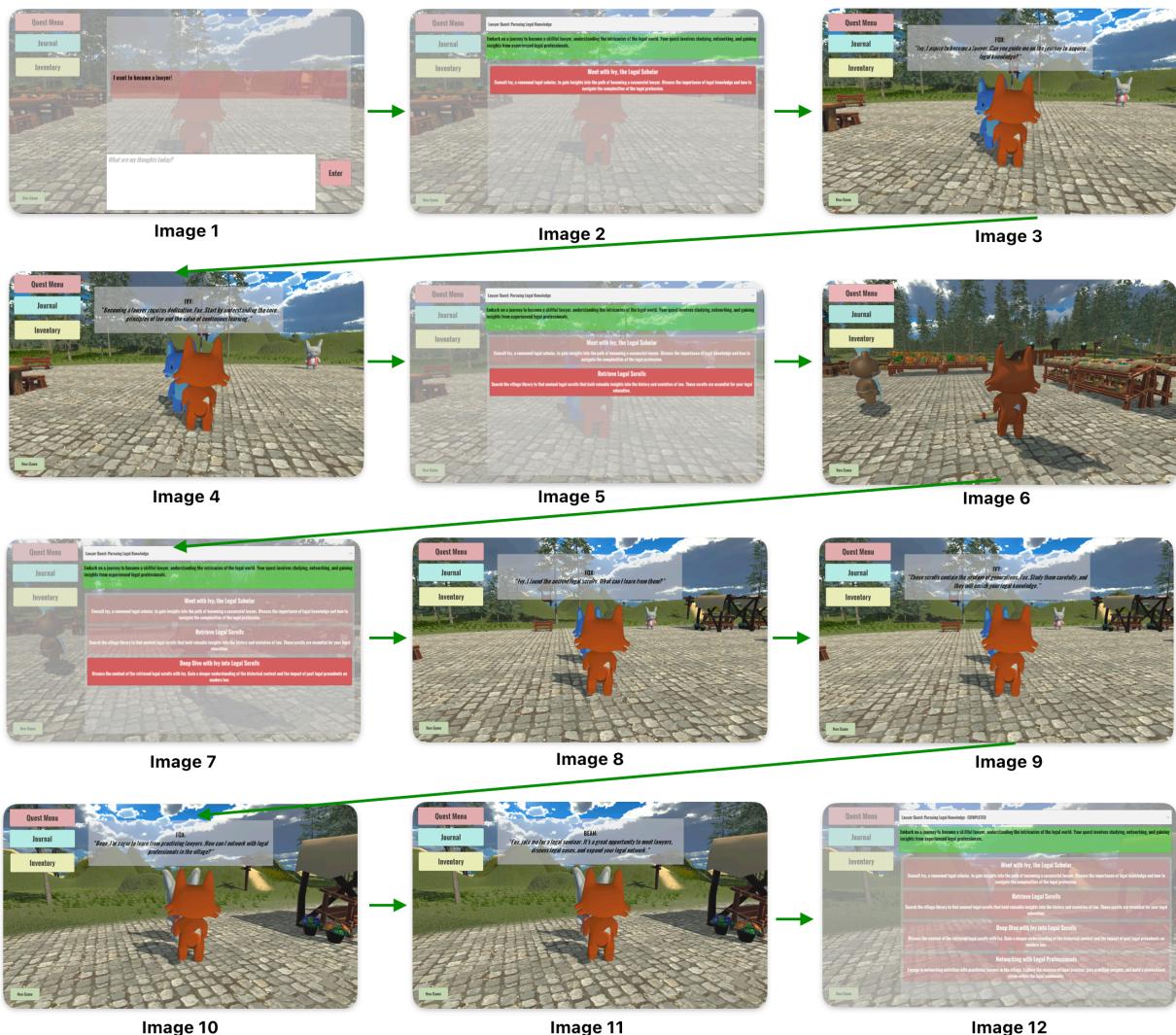
- [Image 1] The player enters the following journal entry: “*It would be really cool to find a message in a bottle by the beach*”.
- [Image 2] The quest series *Distant Note in a Bottle* is added to the quest menu.
- [Image 3] The first step *Bottle of Secrets* begins, which is a FIND\_OBJECT quest. The player must look for a bottle which is located somewhere on the beach.
- [Image 4] Once the player has found the bottle sitting by the water, the next step *Sharing the Discovery* activates. This is a TALK\_NPC quest in which Fox (the player) has to engage Ivy in conversation to talk about the bottle.
- [Image 5-7] Fox discusses the findings with Ivy and the quest series comes to an end.

Similarly to the previous example, this quest series demonstrates how the architecture successfully achieves each goal in its intended sequence of steps. The player input is successfully translated into an API call to the fine-tuned model which generates a related quest series, and the resulting output is parsed into actionable quests by the codebase.

It is important to note that the player's reference of the beach was directly incorporated into the quest series produced by the model. Location was one of the parameters that the fine-tuned model was trained on. This highlights how the fine-tuned model is able to successfully incorporate the nuances of the player entry into its generated quest series. Ultimately, this mechanism provides the player with a higher sense of agency as the details of the player input directly impact the specifics of the playable quest series.

### 3.2.3 Lawyer Quest: Pursuing Legal Knowledge

The final example underscores the adaptability of the fine-tuned model when presented with new game assets, such as objects or NPCs. In this case, I added the 3D asset PAPER\_SCROLL to the list of potential objects to find. Adding it in the Unity editor included it automatically to the system instructions. This item was not present in any training examples during the fine-tuning process, so the model effectively has not seen it before.



**Figure 7** - series of screenshots showing the quest *Lawyer Quest: Pursuing Legal Knowledge*

The series in Figure 8 consists of the following elements:

- [Image 1] The journal entry “*I want to become a lawyer!*” is added to the journal.
- [Image 2] The quest series *Lawyer Quest: Pursuing Legal Knowledge* begins with the first quest step, *Meet with Ivy, the Legal Scholar*.
- [Image 3-4] In this TALK\_NPC quest, Fox (the player) asks Ivy for advice on becoming a lawyer. Ivy suggests starting by “...understanding the core principles of law and the value of continuous learning”.

4. [Image 5] The next quest, *Retrieve Legal Scrolls*, begins. This is a FIND\_OBJECT quest.
5. [Image 6] The player must find a paper scroll in the village.
6. [Image 7] Once this has been found, the TALK\_NPC quest step *Deep Dive with Ivy into Legal Scroll* begins.
7. [Image 8-9] Fox tells Ivy that the legal scrolls have been found, and Ivy encourages Fox to study them for more knowledge.
8. [Image 10-11] The next TALK\_NPC quest begins, which is *Networking with Legal Professionals*. Fox chats to Bean about networking with fellow lawyers, to which Bean invites Fox to a legal seminar. [Note: the screenshot showing this quest step active in the quest menu is missing from Figure 7, but can be seen in Image 12].
9. [Image 12] The quest series completes.

This play-through illustrates that the architecture can successfully deliberate new game assets as viable options for the quests technical details. Even though the fine-tuned model had not seen a paper scroll as a potential game asset during training, it correctly rationalised that a paper scroll was the most suitable object for the FIND\_OBJECT quest when compared to the other available assets specified in the system instructions. This adds a layer of flexibility to the architecture in which the model adapts to the details of the game world dynamically. Assets such as objects and NPCs can be added freely during development without the need for additional fine-tuning.

---

## 4. Discussion

Overall, the results highlight two key observations regarding the fine-tuning process and the integration of the fine-tuned model in the game environment:

1. Firstly, fine-tuning allows for streamlined system instructions and an ability to tailor the model to the various constraints of the game world, yet requires further iterations as well as a larger dataset to capture the nuances of the world and generate more complex quest series.
2. Nevertheless, the communication between the fine-tuned model and the code architecture is successfully able to translate the model's output into playable quests, start to finish, in the game environment.

These observations confirm that the described framework succeeds in achieving the goal of dynamic quest generation in the context of a specific game world. Such an approach enhances player agency by allowing the player to craft their own game experience.

In the following section, I discuss important considerations of the proposed framework and the overall goal of dynamic quest generation using a GPT model.

## 4.1 Fine-tuning

The more detail the game world has and the more constraints the GPT model needs to follow, the more fine-tuning a tailored model is the appropriate choice. As demonstrated so far, such an approach greatly cuts down on the need for verbose system instructions and avoids having to hand-hold the model with additional examples at run-time.

### 4.1.1 Training Dataset

An important consideration is the dataset used to train the fine-tuned model. OpenAI requires a minimum of 10 examples but are adamant in their guidance that a larger dataset produces enhanced output (OpenAI Platform, no date). In this project, the dataset used to train the final model consisted of 48 examples. This went a long way towards providing the model with a solid understanding of the output format and the technicalities of the game world (NPC names, objects, etc). However, it quickly became clear that more examples are needed across a much larger range of parameters to fully capture the nuances of the game world and produce more sophisticated and cohesive quests. The training dataset should ideally be comprised of examples that capture parameters such as dialogue length, dialogue tone of each NPC, types of objects to find and quest series length in order to generate more meaningful quests.

### 4.1.2 Incorporating Details of the Game World

As can be seen in both fine-tuned models, even with a dataset that supposedly captured the predetermined personalities of the NPCs, the GPT model was often incorrect in its descriptions of these characters. It prescribed them attributes that were not present in their original description (such as referring to Ivy as the town historian) and frequently got the species wrong. While special attention was given to specify character details in each example, there were not enough examples overall for the model to properly learn their characteristics. This explains the model's tendency to fabricate details in such a way that matches the pattern found in the data, without actually grasping the concrete information these details represent.

As discussed, these are relatively minor mistakes in that they do not prevent the code from instantiating the quest series since the technical details are correct. However, they do make it confusing for the player who can only follow the quest description which is ultimately misleading. This will likely increase frustration for the player and decrease narrative cohesion.

### 4.1.3 Repetition of Output

Furthermore, a larger dataset is needed to prevent the model from consistently producing similar output that feels as though it is following a template. The creativity of the GPT model can feel stifled if the examples are too similar and increases the risk of the model overfitting.

Most popular RPG games have different types of quests, usually broken down into main quests (which advance the main storyline) and side quests (tangential missions that do not necessarily play into the main storyline but accomplish smaller side goals). This should be another parameter to fine-tune the model on in order to output different types of quest series and varied levels of quest significance. This level of variety would add another level of dimension to the overall gameplay and increase the complexity of the overall game.

#### **4.1.4 Changing Fine-tuning Parameters**

This project did not investigate changes in the parameters of fine-tuning the model. OpenAI's default parameters were chosen during fine-tuning, and the dependent variable investigated was the data provided to the model. Nevertheless, it would be an interesting endeavour to alter the parameters during the fine-tuning process to examine how these could achieve a more desired outcome.

## 4.2 Technical Considerations

### 4.2.1 Tokenisation

As with any project that utilises the OpenAI API, one of the main limitations is the monetary cost incurred. This tradeoff is present at all stages in the communication between the game and the GPT model, beginning with the system instructions step. Detailed instructions increase the chances of coherent and correctly-formatted model output, but quickly escalate token count and setup expenses.

In addition, each journal entry input by the player generates an API call and incurs a cost. Moreover, this cost can rapidly increase when no character limit is present. Key considerations include whether each journal entry necessitates quest generation, the implementation of a character limit, and how journal entries are sent to the model (individually or appended to a list). The third consideration is especially important as it can significantly alter the output returned by the model. Providing the model with the current status of all quests ensures that it effectively has a personalised quest history. This grants it the choice to create quests that build off previous ones, enhancing the sense of progression and continuity in the game. However, sending the appended list of input and output can increase token costs exponentially.

To strike a balance between the personalisation of the GPT model's output and the monetary cost of each call, a system should be devised to limit the number of tokens sent to the API. Such a method could be to only send the last  $x$  journal entries/quest series pairs so that the model retains the latest information and drops the earlier information. While this would cause it to essentially develop a short-term memory, it would still take into account the latest developments in the game, increasing consistency in the story and reducing the number of tokens being considered by the model.

These considerations will depend on the particular use case and the game being developed.

### 4.2.2 Dependency on OpenAI API

Relying solely on the OpenAI API in any game is not recommended as it exposes the game to external issues such as latency problems and server downtime. In addition, as this game relies entirely on a GPT model for its quest system, it assumes the player has an OpenAI account, keys and sufficient funds in the account for API calls. To release this game in a production environment would necessitate a streamlined user account system that would create an OpenAI account, set up any required API keys, and fund the account as needed.

### 4.2.3 Error Handling

It is critical to implement robust error handling in the game's code in order to safeguard against any malformed GPT output. In this project, the code segment that parses the JSON and creates the quest series is wrapped in a try/catch block which prevents the game from crashing if the GPT model output cannot be understood by the code. In addition, such safeguarding techniques are required for incorrect information returned by the model, such as a TALK\_NPC quest that refers to a non-existent character or a FIND\_OBJECT quest that links to a non-existent game asset. This can be handled by a similar mechanism (a try/catch block) allowing the game to silently discard the quest series or step. This architecture employs an alternative technique for TALK\_NPC and FIND\_OBJECT quests, generating a default character in the former case and a cube in the latter.

## 4.3 Immersion and Interest

### 4.3.1 Importance of Engaging Quests

A crucial consideration of this dynamic quest generation architecture is the engagement level of the GPT model's generated quests for players. While these quests are technically playable and are relevant to the player input, they are ultimately not as immersive as those found in popular RPGS which are crafted by teams of narrative designers. Ultimately the quests produced by the GPT model can feel insubstantial, as their completion does not significantly contribute to overall game progression.

Two strategies are proposed to combat this issue and maximise the dynamic quest generation architecture's potential. The first strategy involves utilising the architecture as a complement to a quest library created by human narrative designers. This would enable professionals to craft the main storyline and the model would extend it with supporting quests tailored to player input.

The second strategy is similar in that it incorporates human narrative designers, albeit with a difference in the role that the model would play. The strategy consists of training the model on an existing library of quests written by the narrative designers, ideally with a wealth of quest types, so that it can learn from legitimate stories written by professionals. The model can then generate quests of a similar style and complexity, tailored to the desired tone and style of the game. It is also vital to ensure that this library contains plenty of examples that capture the details of the game world, as quests that do not align with the world break immersion.

These two approaches would achieve the main goal of crafting compelling yet personalised games via the partnership of narrative design and the dynamic quest generation architecture.

## 5. Future Avenues

This project successfully implemented the core architecture for dynamic quest generation using an LLM. There are a multitude of future avenues for augmenting this architecture, thereby greatly enhancing its ability to craft bespoke and immersive gameplay experiences. In this section, I discuss five of these paths.

### 5.1 Additional Quest Types

Perhaps one of the most evident ways to extend this architecture is by introducing new quest types. The architecture described in this paper implemented three main quest types: collecting coins, finding objects and talking to NPCs. This consequently limits the quest series generated to a combination of these categories. In order to diversify the quest series and enrich the GPT model’s output, further types can be incorporated; these can include fighting enemies, construction tasks, solving puzzles, or improving an existing object in the game. These quest types should ideally align with the genre of the game being developed. For example, an action game will focus more on combat quests while a simulator game would prioritise business building tasks. Clearly, augmenting the architecture with additional quest types is an effective mechanism to enhance its output and consequently boost player engagement.

### 5.2 Incorporating Current World State

There’s a wealth of information to collect during gameplay, including NPC interactions, visited locations, quest status and objects seen by the player. Incorporating this data as part of the GPT model input holds immense potential for enhancing the depth and dynamic nature of quests; it adds a layer of player customisation, further creating unique gameplay based on player actions. Implementing this approach requires strategic thought about the role different categories of data would play in shaping quests, determining which input is relevant and how to alter the narrative scope of quests accordingly. In addition, the training data for the fine-tuned model would need to be updated so that the model adapts to the new example quests and learns how to consider these details at run-time.

### 5.3 Synthesising World Information

Furthermore, the architecture’s efficacy would be significantly enhanced by synthesising the game world information before sending it to the GPT model, in a framework akin to that developed by Park *et al.* (2023). As discussed previously, a major challenge of increasing model input is the subsequent increased token counts. Moreover, sending the model raw data regarding the player’s actions and state of the game world without accompanying context (such as when it happened and event details) carries the risk of introducing significant noise to the model and consequently having it generate nonsensical quests. As a result, introducing a system that processes this raw data in such a way that provides the model with a snapshot of the player’s perspective is crucial, as it facilitates a contextualised understanding of player actions and the game world while aiming to keep token costs low.

In Park *et al.*’s 2023 study, a memory stream was implemented such that agents could synthesise their interactions and generate high-level reflections about their experiences in the digital world. The paper also put forward a proprietary retrieval function that is able to discern what is relevant input for the LLM to consider when deciding the agents future actions. Establishing a comparable

framework in this dynamic quest generation architecture would go a long way in both tailoring quests to player behaviour and game status as well as mitigating token count. Synthesising this data is a powerful mechanism for enhancing the model's intelligence and memory regarding the world's state and player behaviour, thus crafting more immersive and cohesive quests as well as heightening player agency.

#### **5.4 Procedural Generation and Altering the Game World**

Creating a dynamic world that responds to quest outcomes is a captivating direction for this project, albeit one that presents significant technical challenges. For example, if the outcome of a quest is the villagers gathering for a party, adorning the town square with balloons and streamers would seamlessly bridge the visual world with the game's narrative. This approach would not only reinforce the coherence of the quest series but also promote the game world from a mere setting into an integral part of the game's experience. Witnessing tangible visual shifts as a direct results of player actions greatly augments the sense of autonomy and influence within the game world.

The current architecture lacks the capability to alter the 3D environment, and would require an additional method of communication between the GPT model and the codebase. Leveraging OpenAI's function calling methodology would be an effective way to address this limitation. As explained in the Introduction, the function calling methodology enables the model to return a function call, along with its parameters defined, which can then be executed in the codebase. This designated function would manage changes in the 3D environment, ranging from simple tasks such as relocating an NPC to complex ones like procedurally generating a new section in the world. Integrating this function call with the quest series output enables the model to determine both the quest series and the corresponding world alterations, ensuring consistency between the story and the game environment.

#### **5.5 Combining with Other A.I. Tools**

Finally, merging this dynamic quest generation architecture with other A.I. tools holds the potential to create worlds that are truly alive, offering players a heightened sense of agency and significantly enhancing immersion. As mentioned in the Introduction, NPC character engines allow for seamless conversations with characters in the world and by themselves are very powerful in creating dynamic worlds. Integrating these conversations into the dynamic quest generation architecture to derive related quests introduces a novel layer of player agency not yet seen in commercial games. This affords players genuine influence over the digital world they inhabit.

---

## 6. Conclusion

In conclusion, this paper proposed a novel dynamic quest generation architecture that leverages a fine-tuned GPT model to craft distinctive quests based on player input. The main goal was to explore the extent to which players can alter the game world they find themselves in and how a LLM, specifically the OpenAI GPT-3.5 model, can be employed to achieve this objective in the context of the game's quest structure.

To explore this goal, a two-part project was undertaken accordingly. The first component consisted of building a 3D role-playing game in the Unity game engine in which players input their thoughts in a journal which make API calls to the GPT model, generating related quest series accordingly. The second component explored fine-tuning the GPT-3.5 model in order to tailor a new model that outputs quest series in the desired format and which grasps the contextual nuances of the game world. A training dataset was compiled accordingly which consists of a number of examples that explore these different facets of the game.

Using this dataset, a fine-tuned model was created and subsequently tested, and its limitations were improved upon in a second fine-tuned model. Testing of the second model highlighted that while the model lacked an advanced level of narrative depth, its output was consistently technically correct and aligned with the overall tone and theme of the game. To confirm this, the model was integrated into the Unity game and tested with several different journal inputs. Each generated quest series was translated by the game's code into actionable quests and which the player could execute in the game in its entirety.

This paper then delved into a discussion of the results and pondered the technical considerations of this architecture, such as its tokenisation limitations and substance of the generated content. The prospect of integrating this architecture with the work of human narrative designers was put forward, as it would allow the model to learn from and expand upon meaningful storylines crafted by professionals. Finally, future avenues for extending the architecture were explored.

---

## 7. References

- Fairclough, C. *et al.* (no date) ‘Research Directions for AI in Computer Games’.
- Marr, B. (2023) *A Short History Of ChatGPT: How We Got To Where We Are Today*, *Forbes*. Available at: <https://www.forbes.com/sites/bernardmarr/2023/05/19/a-short-history-of-chatgpt-how-we-got-to-where-we-are-today/> (Accessed: 26 January 2024).
- Nareyek, A. (2004) ‘AI in Computer Games: Smarter games are making for a better user experience. What does the future hold?’, *Queue*, 1(10), pp. 58–65. Available at: <https://doi.org/10.1145/971564.971593>.
- Park, J.S. *et al.* (2023) ‘Generative Agents: Interactive Simulacra of Human Behavior’. arXiv. Available at: <http://arxiv.org/abs/2304.03442> (Accessed: 18 August 2023).
- Safadi, F., Fonteneau, R. and Ernst, D. (2015) ‘Artificial Intelligence in Video Games: Towards a Unified Framework’, *International Journal of Computer Games Technology*, 2015, p. e271296. Available at: <https://doi.org/10.1155/2015/271296>.
- Skinner, G. and Walmsley, T. (2019) ‘Artificial Intelligence and Deep Learning in Video Games A Brief Review’, in *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS). 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)*, pp. 404–408. Available at: <https://doi.org/10.1109/CCOMS.2019.8821783>.
- Stuart, K. (2021) ‘Think, fight, feel: how video game artificial intelligence is evolving’, *The Guardian*, 19 July. Available at: <https://www.theguardian.com/games/2021/jul/19/video-gaming-artificial-intelligence-ai-is-evolving> (Accessed: 3 July 2023).
- Urwin, M. (2023) *Artificial Intelligence in Gaming (+ 10 AI Games to Know)*, *BuiltIn*. Available at: <https://builtin.com/artificial-intelligence/ai-games> (Accessed: 3 July 2023).
- Wu, M. and Fishman, S. (2023) *Data preparation and analysis for chat model fine-tuning | OpenAI Cookbook*. Available at: [https://cookbook.openai.com/examples/chat\\_finetuning\\_data\\_prep](https://cookbook.openai.com/examples/chat_finetuning_data_prep) (Accessed: 22 November 2023).
- Yakan, S.A. (2022) ‘Analysis of Development of Artificial Intelligence in the Game Industry’, *International Journal of Cyber and IT Service Management*, 2(2), pp. 111–116. Available at: <https://doi.org/10.34306/ijcitsm.v2i2.100>.
- Home / Bethesda.net* (no date). Available at: <https://bethesda.net/en/dashboard> (Accessed: 28 January 2024).
- Foundation, B. (no date) ‘blender.org - Home of the Blender project - Free and Open 3D Creation Software’, *blender.org*. Available at: <https://www.blender.org/> (Accessed: 17 February 2024).
- Inworld – The most advanced Character Engine for AI NPCs* (no date). Available at: <https://inworld.ai/> (Accessed: 26 January 2024).
- Mixamo* (no date). Available at: <https://www.mixamo.com/> (Accessed: 26 January 2024).

*OpenAI Platform* (no date). Available at: <https://platform.openai.com> (Accessed: 28 January 2024).

*Unity Real-Time Development Platform / 3D, 2D, VR & AR Engine* (no date) *Unity*. Available at: <https://unity.com> (Accessed: 17 February 2024).