Martian Lawyers Club Game Programming Task - Ajmal Rizni

Game Design Plan

- The language model needs to output a clear command that can be used by the game to determine the outcome of a piece capture. To do this, the model must be given relevant details about the capture and instructed to give its answer in a specific format.
- Once the boolean outcome of a fight has been determined (based on the LLM's output), the result of the fight must be communicated to the player in a clear and satisfying way. To do this, I'll feed the language model a brief description of the fight, and ask it to retell the story in a few sentences.
- In order to make the generated stories more interesting and show more personality, I'll take inspiration from social simulation mechanics in games like Rimworld and the Nemesis system from Shadow of Mordor.

Implementation Part 1: Outcome

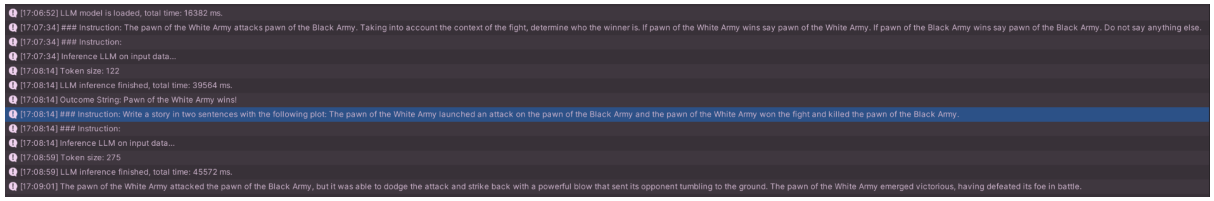
- First, I familiarised myself with the codebase by examining the relevant scripts and functions.
- In order for the outcome of a piece capture to be determined by the GPT4all model, I created an asynchronous version of the calculateVictory() function. The function feeds the LLM a basic description of the fight, and asks it to determine its outcome and present the answer in a specific format.
- I experimented with different types of prompts, and found that the most effective way to get the desired result was to plainly describe the fight in natural language, ask the LLM for an outcome, and instruct it to adhere to the given format. The prompt isn't 100% effective, but it generally generates an acceptable response. Below is an example of one of the provided prompts:

Instruction: The Bishop of the White Army attacks black Knight. Taking into account the context of the fight, determine who the winner is. If Bishop of the White Army wins say Bishop of the White Army. If black Knight wins say black Knight. Do not say anything else.

Implementation Part 2: Description

- Once the outcome string is generated, I check the string with the StartsWith() function in order to determine the winner of the fight. Then the LLM is queried with a new prompt to generate a description of the fight in the form of a written story. Below is an example of this prompt:

Instruction: Write a story in two sentences with the following plot: The Bishop of the White Army launched an attack on the black Knight and the Bishop of the White Army won the fight and killed the black Knight.



Implementation Part 3: Vengeance

- Each character is assigned a unique ID held in the Chessman script. When a character/chess piece spawns, a list of the character's friends is randomly generated.
- When a character dies (ie. a piece is removed from the board), any alive characters that were friends with the fallen character is given a modifier string in the following format:
- ***The black Knight was enraged by the death of their friend, the pawn of the Black Army.***
- Then, whenever the grieving character engages in a battle, this modifier string is placed below the prompt, creating a more interesting description which takes into account the history and relationships of the story. Below is an of an LLM-generated story:

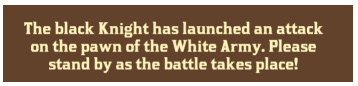
The Queen of England was devastated by the loss of her dear friend, the English Bishop. She had always been a strong leader and protector for her people, but in this moment, she felt helpless and alone. The Queen of England rallied her troops and led them into battle against the French foot soldier. After a fierce struggle that left many casualties on both sides, the Queen of England emerged victorious. But even as she celebrated their victory, the Queen of England couldn't shake off the weight of loss that had settled heavily in her heart.

Findings & adjustments

- When the temperature is set to a low value, the LLM tends to repeat itself, which is unsatisfying for the player. However, when the value is set too high, the response generated often doesn't make much sense. I experimented with different values and landed around a temperature of 0.4.

Temperature=0.28 Temperature=0.7

- Since the LLM inference speed is quite slow, I decided to give the player some immediate feedback once the battle starts, so that they know to wait for the outcome of the battle.



- I found that when characters had names like "pawn of the White Army" the LLM would sometimes generate responses in the theme of a Chess game, which isn't the intention of the narrative generation in this game. So I decided to change the names of each character in the theme of an Anglo-French medieval war. (eg. "French foot soldier", "Knight of England").

Notes & Further Ideas

- I'm not certain if having the LLM determine the outcome of a piece capture is worthwhile. It's not clear if the LLM really takes the context of the fight into account, and with the slow inference speed it may make more sense to determine the outcome of a battle through a simple variable-weighting algorithm. This would halve the waiting time of each battle, and it may generate more satisfying results.
- Alternatively, if a more sophisticated Large Language Model were used, it may be possible to provide the model much more information about the context and personalities of each character, and have it generate more interesting outcomes based on the total history of the game. For example, an LLM could be used to not just generate fight outcomes, but also the effects that a battle has on each character's emotions and relationships.
- At the moment, the LLM only takes into account if a character's friend has died, but it could be expanded to produce even more varied and dynamic stories. For example, characters could have sworn enemies, or they could get weakened in battle, or characters on opposite sides of the war could even have a Romeo & Juliet relationship.

Screenshots