TULANE UNIVERSITY
SCHOOL *of* SCIENCE
& ENGINEERING

Large Language Model (LLM) Enhanced Video Game

Luke Gulson - Computer Science and Mathematics, Ben Russell - Computer Science and English, Bobby Becker - Computer Science and Philosophy

Alireza Shirvani - Computer Science Department

INTRODUCTION/ABSTRACT

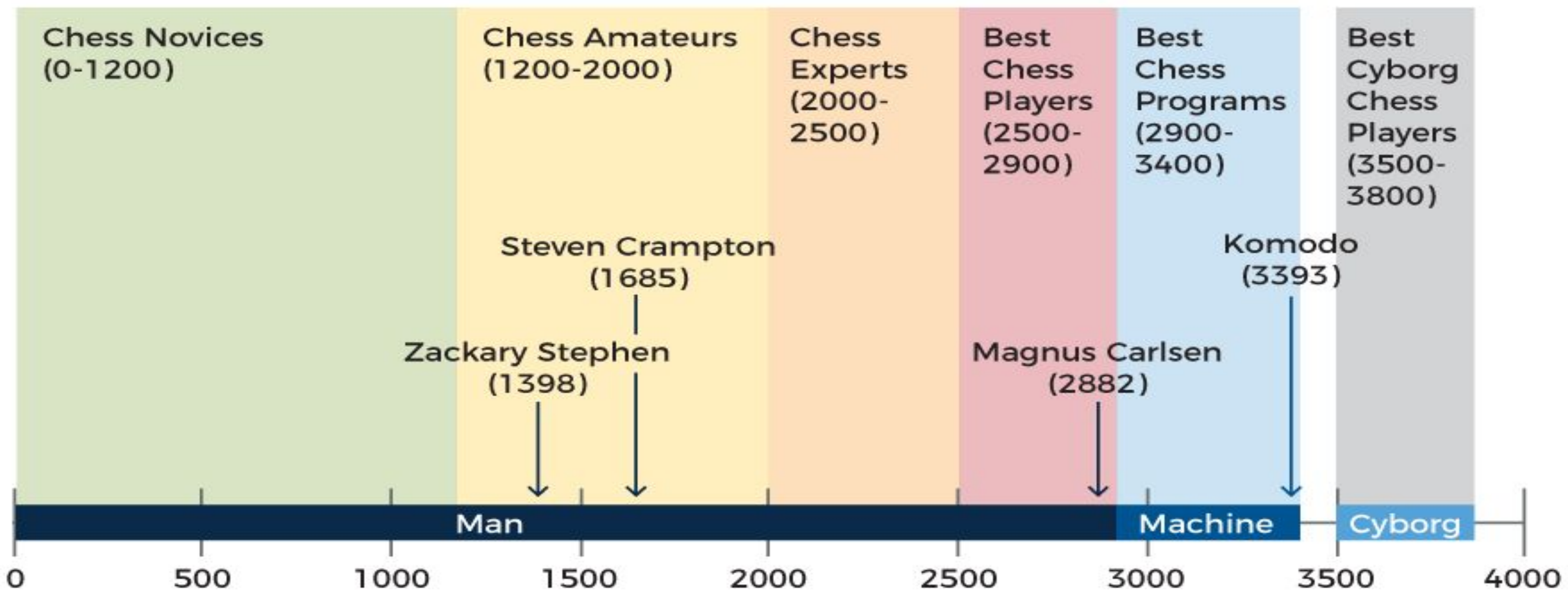
As creative AI content continues to grow, we face a pivotal question: how do we use these innovations to facilitate the creative process while keeping humans in control? Our large language model (LLM) enhanced video game is one approach to this dilemma. We designed a Unity demo for a third person fantasy RPG to test different applications of LLM. We use OpenAI's API for ChatGPT to generate character dialogue and flavor text to add replayability to our game, providing a different experience in every playthrough. By giving all characters access to a lore file written by us, we ensure that all LLM-generated dialogue is engaging, consistent, and synergistic with our overall narrative.

OBJECTIVE

Our goal is to demonstrate the possibilities of LLM technology in video games. We strived to avoid many of the pitfalls that creative usage of LLM can fall victim to, such as:

- Derivative content
- Bland output
- Inconsistency in memory

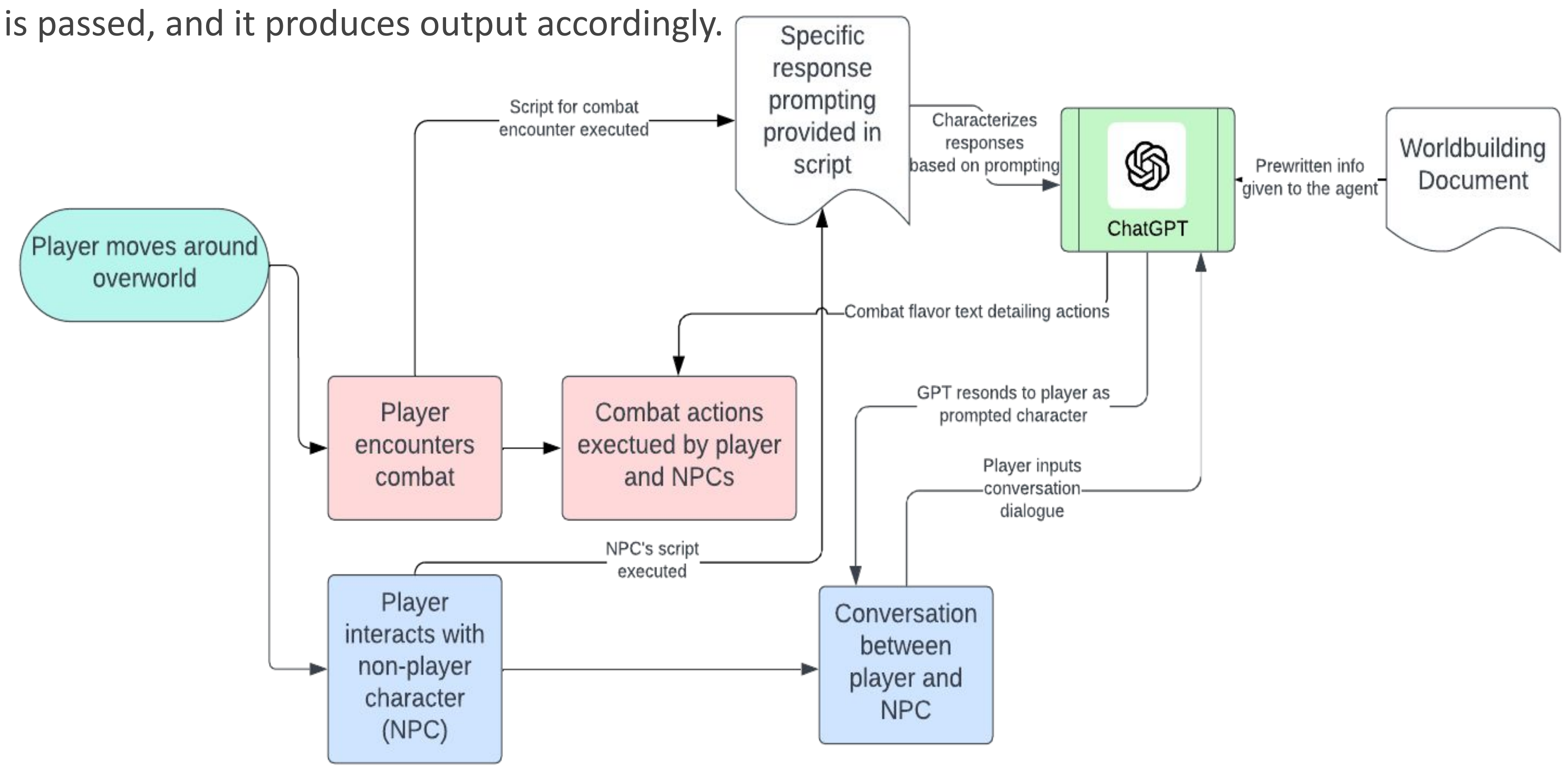
Early in our research, we found an article discussing the advantages of a combined human-AI "cyborg" chess agent. Indeed, cyborg agents have won against chess grandmasters in tournament settings, such as ZackS in the Freestyle Chess Tournament in 2005. This graph below demonstrates this effect; indeed, Steven Crampton and Zackary Stephen are the developers of ZackS.



Inspired by the increased performance human-AI collaboration, we wanted to apply the same principles to video games. By creating specific prompting for the LLM to utilize, we could produce a high-quality narrative in far less time than by writing dialogue and narration ourselves.

METHODS & APPROACH

The two platforms that we were instrumental to us pulling off this project were the Unity game building engine and OpenAI's API for GPT. Below is a diagram detailing how and when ChatGPT receives input and outputs logical responses in the Unity game engine. Responses are generated in real time based on pre-scripted prompts. When a script using ChatGPT is executed, the prompting is passed, and it produces output accordingly.


Specific response prompting provided in script
Script for combat encounter executed
Characterizes responses based on prompting
ChatGPT
Prewritten info given to the agent
Worldbuilding Document
Player moves around overworld
Combat flavor text detailing actions
Player encounters combat
Combat actions exectued by player and NPCs
GPT resonds to player as prompted character
Player inputs conversation dialogue
NPC's script executed
Player interacts with non-player character (NPC)
Conversation between player and NPC

NOVELTY

We feel our method of LLM implementation in the video game was one of our most novel features, as we still hold big-picture creative control. This may help assuage fears of AI infringing on writing positions by demonstrating how efficient a hybridized approach can be. We also are developing a way to have the LLM to create combat moves based on player input. The user specifies what kind of move they'd like to do, and ChatGPT comes up with the effects. This feature was not refined enough to make it into the final project, but we hope to develop this concept further.

FINDINGS

We found that combining human creativity with LLM efficiency proved worthwhile. With only a few sentences of prompting, we created distinct characters for ChatGPT to portray, saving our small development team a great deal of time. It is an exciting prospect to see what this rapidly developing technology is capable of now and hypothesize what our approach may yield in the future. Below are several screenshots of our methodology in action.

Combat Flavor Text:

To generate high-quality flavor text, we prompt the GPT model with selected information, such as the characters involved, the characters' stats and the type of attack used. By combining that information with custom prompts, we were able to get the GPT model to consistently generate immersive and original output. Here are some examples of its output:

After using a frost attack: "Handoo's frosty breath froze the very air around him, sending shards of ice raining down upon Gorgias."

After using a healing move: "Handoo's hands radiated a warm aura, enveloping him in a healing light that washed away his wounds and revitalized his spirit."

After getting attacked: "Gorgias's axe swung with thunderous force, meeting Handoo's defenses with a resounding clash."

NPC Dialogue Interactions

The data that our LLM uses to generate NPC dialogue comes from player inputs through a text interface. This allows the dialogue to be fluid and adaptable; if players choose, they probe into NPCs to learn more about the backstory of the characters.

We also gave hidden information to some NPCs; if players are clever enough, they can probe the characters to learn the locations of items or weakness of enemy characters. In addition, for one interaction, we use ChatGPT to measure the hostility of the player's dialogue. The game then uses this measurement to change the course of the game, either by adding an extra combat encounter or giving the player a bonus item.

While the final playthrough lasts only around 10 minutes, the variability of the text outputs allows the player to learn more about the world and unravel its mysteries in each and every playthrough.

Experimental GPT Features

In addition to flavor text and character dialogue interactions, we developed and tested a number of other features that could be integrated using an LLM that did not make our final product. By integrating GPT into a turn-based combat system with other language processing tools, we created a method that can give players the ability to write out their own combat moves. That data is then interpreted for the game to determine how much damage is done and what other effects their actions have on the combat encounter.

CONCLUSIONS & FUTURE WORK

The development of our game was an informative one. We correctly anticipated some obstacles, but some were unexpected when it came to integrating a generative LLM in Unity. It was a challenge to initially merge the two programs, but once we did the, ChatGPT API proved cooperative to work with. We feel that we successfully walked the line between LLM-generated and human generated content to produce a final product that demonstrates the capabilities of combining the two.

Moving forward, we would like to see the game expanded to further test the capabilities of using LLM in this way. It could prove beneficial to test how LLM could redistribute the workload of a small team of developers, in particular when it comes to writing dialogue trees for an expansive narrative. Based on this project, the potential is undeniably there, and we are excited to develop it further.

REFERENCES

https://www.kitces.com/blog/cyborg-chess-advisor-teach-about-future-financial-planning/
https://www.linkedin.com/pulse/freestyle-chess-future-marketing-gabriel-ehrlich/
https://en.chessbase.com/post/dark-horse-zacks-wins-freestyle-che-tournament