Procedural Novellas:
AI Storytelling for Narrative Games

Project Documentation

Steve Berthiaume

Design and Computation Arts, Concordia University

CART 410: Research-creation in Computation Arts

Brice Ammar-Khodja

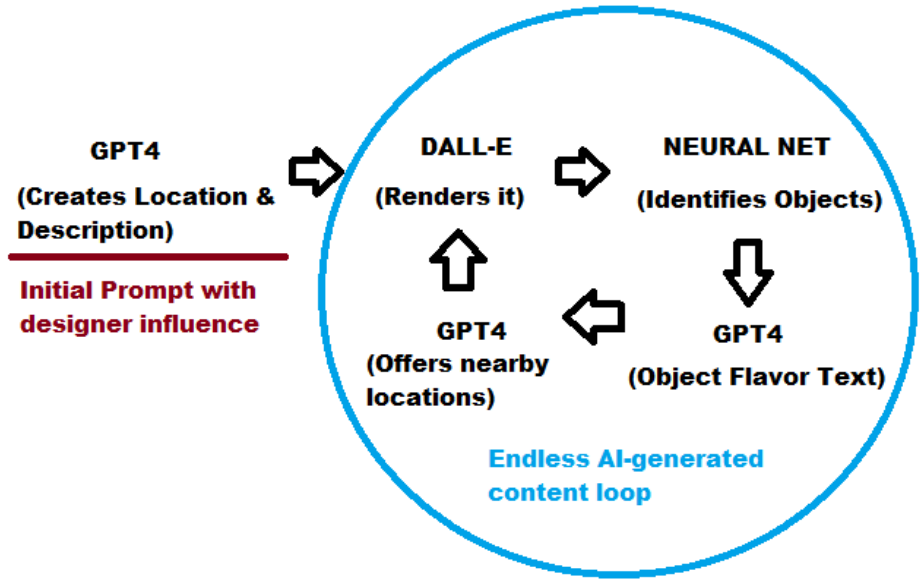
November 29, 2023

CONTENTS:

SECTION 1: IDEATION

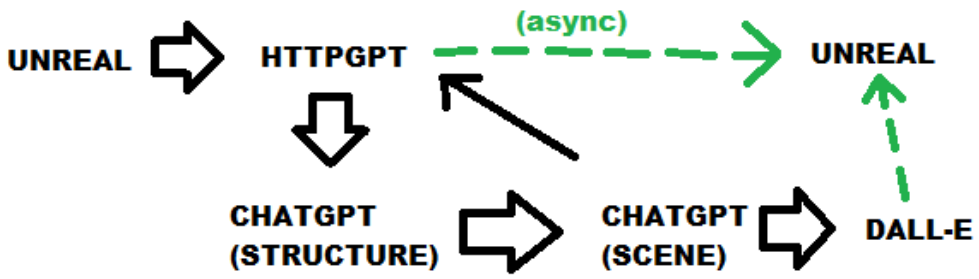
Procedural Novellas was not always Procedural Novellas, naturally. At the project's inception, I faced the decision of finding which game genre would best explore AI integration, while representing the medium of games adequately enough to not serve as too much of an abstraction. The two genres which immediately stuck out to me were Adventure games and Visual Novels.

Adventure games, named such after the titular game Colossal Cave Adventure from 1977, focus on user text input to navigate scenarios based on what text the system displays for them. These text-based RPGs rely on a back and forth where designers are expected to have thought of many, many interactions, lest player frustration take hold. While I do believe this would also be an extremely interesting avenue to take with AI-integration in games, I ultimately felt that the current stage AI finds itself at makes this more challenging for a few reasons. First is the coherence issue; adventure games require much extrapolation from a player and due to the frequent presence of puzzles, must make sense to be any good. ChatGPT is known to occasionally forget about points it brought up earlier, which can make this a massive issue. Second, the aforementioned puzzles often involve inventory systems, which would be difficult to design a proper framework for when all content is AI-generated. It would be impossible to ascertain how many items the AI may choose to create, whether it'd adequately label them, or whether their use cases would even be logical. Third is the fact that the most interesting implementation for ChatGPT with such a genre, personally speaking, would be in a context where a Neural Network can be used to read an image DALL-E generates based off a very detailed ChatGPT prompt, in order to find objects placed in a scene, which ChatGPT could then write flavor text for, or to otherwise offer nearby locations for a player to continue towards. This sort of intricate system, while immensely interesting to hypothesize about, was simply far too complex for my abilities and timeframe to overcome.



^ Diagram representing the Adventure Game concept. Note that past initial designer influence with the first ChatGPT generation, all AI content feeds into other AI, creating an endless cycle with no designer oversight.

Seeing as the Adventure Game avenue proved too complex, I turned to Visual Novels. Visual Novels are, as the name suggests, heavily story-driven and text-based. That said, they often also feature characters to interact with, and varied locations and settings. The genre is known for dramatic or otherwise surprising storytelling, usually featuring several key twists in the very last chapters. The genre has various titles where player agency can lead to entirely different stories, as well as others that feature lesser, more controlled variations. Variation notwithstanding, Visual Novels offered a far simpler avenue for AI-integration, as most of the interactions involve advancing text to some extent. I chose from the outset to eliminate story deviations from the project, in part due to time constraints, but also due to the challenges controlling for extremely precise ChatGPT output in this context would pose. My initial system design for an AI-integrated visual novel in Unreal Engine is depicted below. My initial hope was to use two separate instances of ChatGPT: one to act as the structural unit, planning how the story would unfold and making all the more involved creative decisions, with another serving as the Scene or Elaborator unit, fleshing out each of the key points from the other instance and feeding refined descriptions into DALL-E for image generation. At the end of these asynchronous requests' completion, a user would be brought to the output in Unreal.

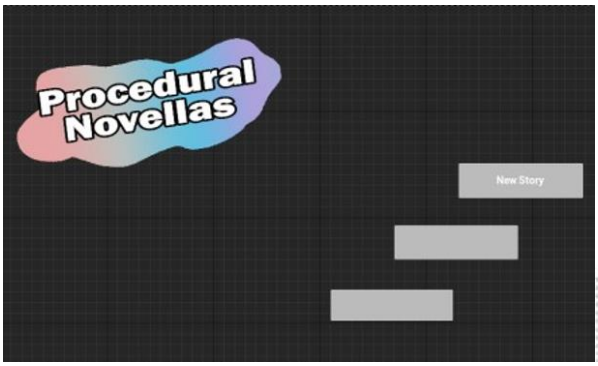


^ Diagram representing the Visual Nove Game concept. Note that the use of multiple instances of ChatGPT did not remain in later structural plans, in part due to the complexities involved, but also due to the unreliability of ChatGPT as a structural generator.

The Structure unit's initial design plan was to specify the story's title, genre, main character, scenes, story and conflict subversion. This would be fed into the Scene unit piece by piece, allowing it to create more fleshed out descriptions and visualizations of the events and elements. One issue I had failed to consider at the time which would later come up was the wait time involved. While this structure is hypothetically perfect if all of its components work and the elements manage to build off each other successfully, the wait times would be astronomical compared to what most would be willing to wait. Seeing as ChatGPT can take upwards of a minute with certain intricate generations, it would not be farfetched to state that this initial plan would have taken a minimum of three minutes of waiting to generate a single story. Additionally, given the nature of these AI tools as being hosted on online platforms, were any of the components to fail to generate in that time, it would be a logistical struggle to pinpoint the failure point and identify whether one, multiple, or all the components had failed. Later designs would address these problems, mostly by relying on only one instance of ChatGPT, and leaving most of the interpretation to pre-written code with no chance of catastrophic failure.

SECTION 2: GROUNDWORK

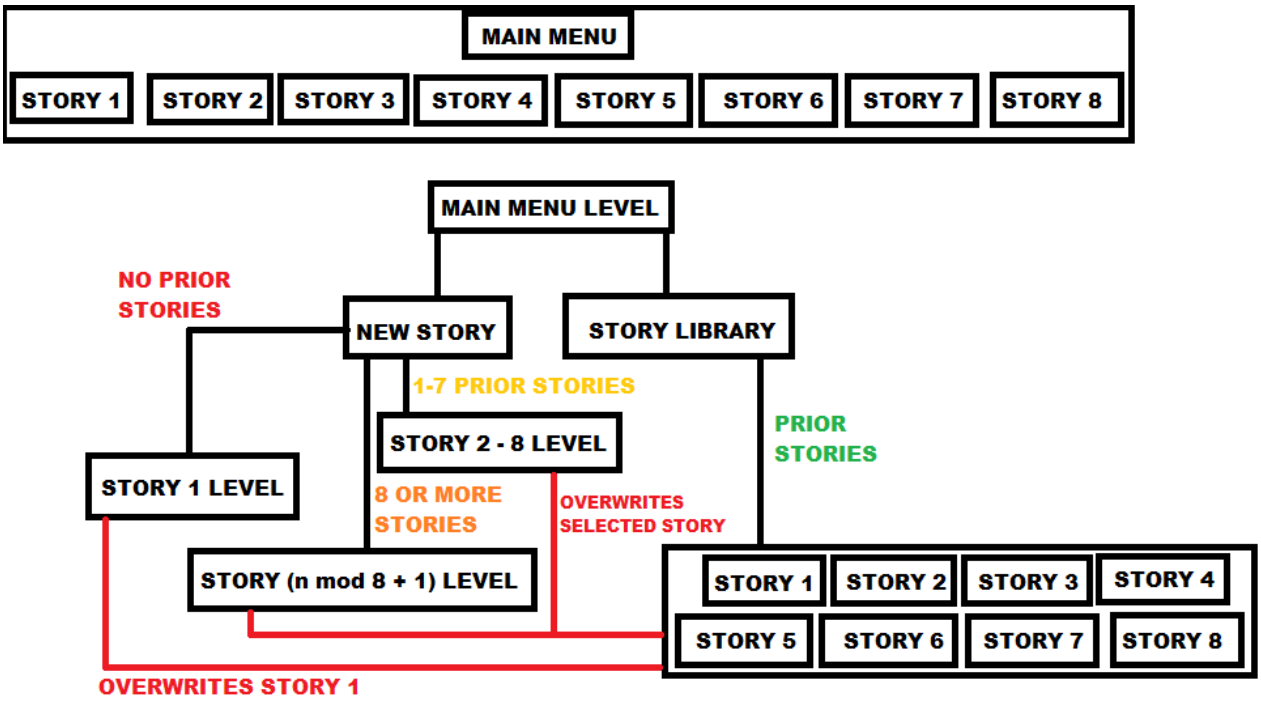
Work on the project began in Unreal Engine. At the time of the project's inception, Unity had just announced their controversial policy change of taxing developers for installs, and the resulting firestorm led me to disregard it as an option for the project. Given Unreal's HTTPGPT plugin, I felt confident about my chances of creating a similar project there. I began with simple widget explorations, experimenting with how the UI elements could appear and exploring the blueprint visual code at a slow pace alongside the creation of a simple test menu. While working on this basic menu, issues had already begun manifesting: notably, that image files created by Dall-E mini and photoshop were not created the same, regardless of both being PNGs. Dall-E mini's outputted PNG file simply failed to work in Unreal, despite working fine in other applications. Despite this oddity and the apparent complexity of Unreal's blueprint system, I remained confident in my ability to make progress on the project.



^ Unreal Prototype Main Menu Widget.

^ Unreal Prototype Scene Generation Widget

My first real step in laying the groundwork for ChatGPT integration was, naturally, to create the scene where ChatGPT would replace elements, separate from the main menu. While the basic compiling of the assets was straightforward, linking the two scenes together through a button click was oddly challenging. If both widgets were in the same level, all would be fine, as one widget could be replaced by another without issue. However, issues arose when it came to loading in other levels. After I had made the menus themselves, I had charted out the general flow of the program. Essentially, I had grown attached to the idea of implementing a Story Library, which would keep up to eight prior stories saved in system memory, accessible from the main menu. This system would use Unreal's levels, with each story being its own respective level that saves the generated script to the level's assets to be reloaded afterwards. The diagram on the next page illustrates the flowchart for which story would be selected based on what the current number of stories is (**n**). This system utilized modulo (**mod**), a computer science operator which has the function of mapping all numbers to a range, which helped to keep all numbers between 1 and 8. While this functionality was theoretically bulletproof, issues quickly arose when it came to the implementation.



^ Story Selection Flowchart. Note that n stands for the number of stories. Modulo works by returning the remainder of a performed division. As an example, 9 mod 8 returns 1, as that is what gets subtracted to perform a perfect division. The +1 then means the selected story at 9 stories is Story 2.

No matter what I tried, I could not get level switching to function in Unreal Engine. Even after watching tutorials online on the subject, an apparently brand-new level would be loaded, unrelated to the one I sought which contained the Story Generation widget. After hours of failed experimenting, I decided to simply try getting all the functionality done within a singular level. I spent time during this period looking into Unreal Engine's saving system, as it would be crucial to the function of the Story Library. While I was holding off from experimenting with it directly until other issues were resolved, I made plans for which variables I would need to operate the program, and, secondly, whether those variables would need to be stored, thus saved for later instances, or dynamic, loaded in real-time with no relevant initial parameters. This analysis also featured plans for how the loading of the next paragraph would function, comparing the end of the currently displayed sentence with the length of the container paragraph to determine whether the end had been reached. A very similar structure was used in the final project. Stored variables are sourced from ChatGPT, with it generating the contents which are then analyzed by pre-written code which splits them into the dynamic variables.

STRUCT
STORY 1

STORED

STORY TITLE - TEXT
GENRE - TEXT
LOCATION 1 - TEXT
PARAGRAPH 1 - TEXT
LOCATION 2 - TEXT
PARAGRAPH 2 - TEXT
LOCATION 3 - TEXT
PARAGRAPH 3 - TEXT
LOCATION 4 - TEXT
PARAGRAPH 4 - TEXT
LOCATION 5 - TEXT
PARAGRAPH 5 - TEXT
LOCATION 6 - TEXT
PARAGRAPH 6 - TEXT

DYNAMIC

CURRENTLOCALE - TEXT
CURRENTPARAGRAPH - TEXT
- **CHARACTERCOUNT - INTEGER**
- **CURRENTSENTENCE - TEXT**
- **ENDOFCURRENTSENTENCE - INTEGER**

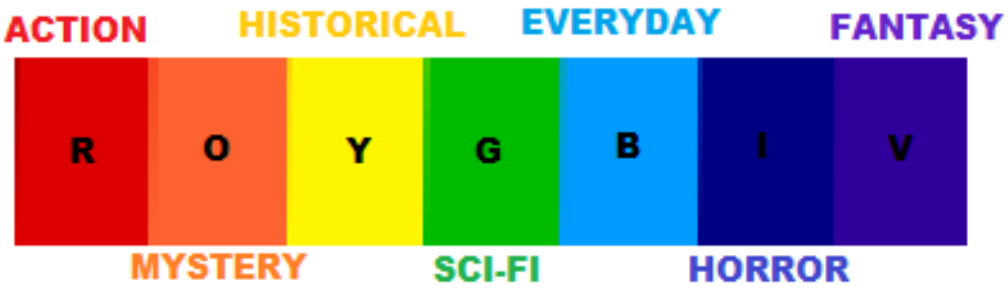
LOCATIONID - INTEGER

IF EOCS = CHARACTERCOUNT
THEN:

LOCATIONID++
CURRENTPARAGRAPH =
"PARAGRAPH "+LOCATIONID

^ Stored vs Dynamic Variables required for any given Story level. Note that this is a basic plan, not a comprehensive outline.

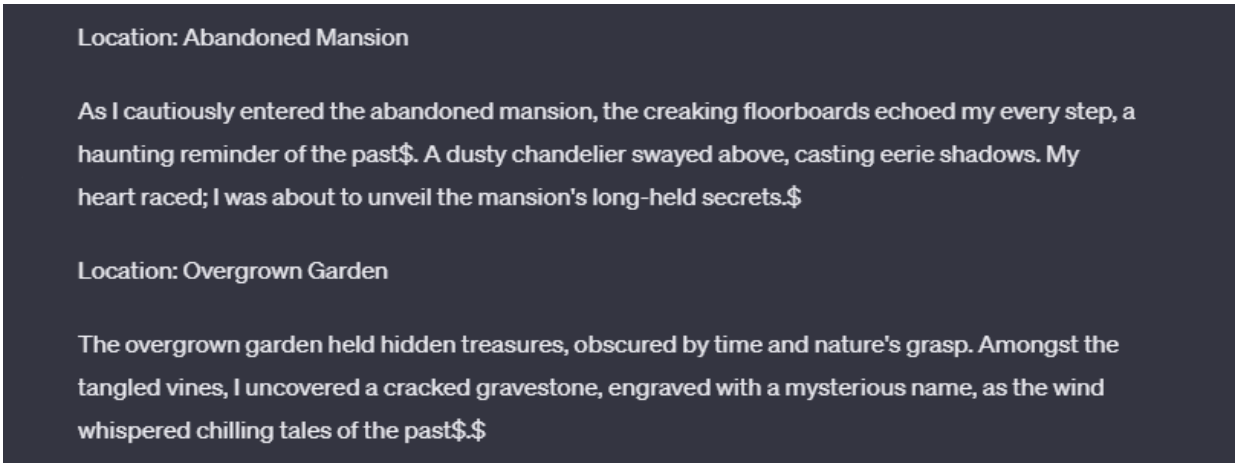
Working on the variables analysis led me to question whether or not ChatGPT could actually successfully write such a thorough set of data every time, and whether or not that output could be successfully interpreted every time. This question would prove pivotal to the project's overall direction. The first significant change to be implemented in the framework would be trying to reduce the number of things for ChatGPT to generate, by predetermining the Genre using a random number generation. This meant setting a set number of predetermined genres, which I set at the 7 following ones: Action, Mystery, Historical, Sci-Fi, Everyday, Horror and Fantasy. Given the scope of ChatGPT output possibilities and the underlying character of the project as one where possibilities seem truly open and infinite, I decided to theme the genres around the well known acronym of the rainbow, ROYGBIV. This is only truly visible in a loose sense with some of the assets that would later make their way inside the project, though I internally track the genre's ID and loose visual aesthetic with this method.



^ The 7 Pre-determined Genres to be chosen from, each tied to a specific color as both a pneumonic device and rough palette for visuals of a specific genre. Note that in the end product, not all pre-generated artwork for any given genre is of the genre's color, as with how Everyday features some predominantly orange backgrounds. The association is mostly for genres which may otherwise need narrowing down.

SECTION 3: CHATGPT OUTPUT MANAGEMENT

In the case of ChatGPT's outputs, I spent considerable time iterating on various methods of getting it to comply with structural instructions. My findings show that ChatGPT is far from predictable or even good at following a list of instructions, and its efficacy sharply declines with each additional instruction. It has a very particular lack of understanding or care for formatting instructions involving punctuation, which made the planning of a system to read and interpret any and all responses it would give in a linear, algorithmic manner much more difficult. In early tests with the output, I experimented with making ChatGPT generate clearly identifiable special characters at the end of every sentence, as to make them easy to identify. It did not take long for this to break, however, as ChatGPT made it abundantly clear that it cannot understand syntactical instructions of that sort.


Location: Abandoned Mansion

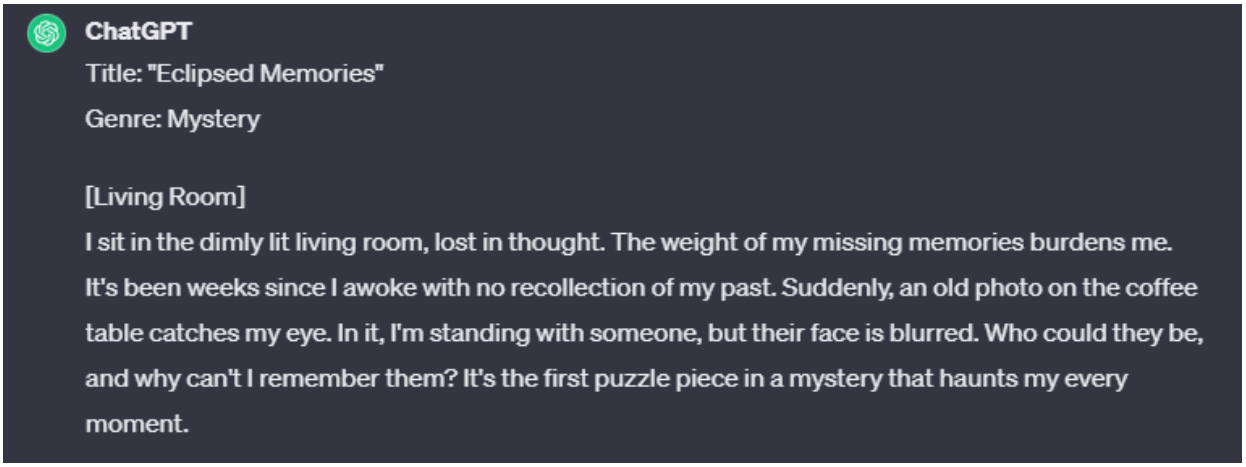
As I cautiously entered the abandoned mansion, the creaking floorboards echoed my every step, a haunting reminder of the past$. A dusty chandelier swayed above, casting eerie shadows. My heart raced; I was about to unveil the mansion's long-held secrets.$

Location: Overgrown Garden

The overgrown garden held hidden treasures, obscured by time and nature's grasp. Amongst the tangled vines, I uncovered a cracked gravestone, engraved with a mysterious name, as the wind whispered chilling tales of the past$.$

^ Excerpt output from my 4th ChatGPT test generation. Note the inconsistent $ placement. In one instance, it's placed before the period. In another, it's placed afterwards. In several cases, no $ is placed at all, and in the final sentence, one is placed both before and after a period.

Considering the Syntactical failings, I then abandoned the use of a $ marker for sentence ends. Unfortunately, that was not the end of issues with ChatGPT's output. ChatGPT would be very inconsistent when it came to labelling additional information, such as the Title, Genre and Locations. Sometimes, all of those would be neatly prefaced with their class and a colon, and in the case of the title, it would be encapsulated within quotes. However, locations would occasionally be contained in square brackets, which came off as rather concerning, as it pointed out the unpredictability of output, and the ways with which said unpredictability directly scales with complexity of input.


ChatGPT

Title: "Eclipsed Memories"

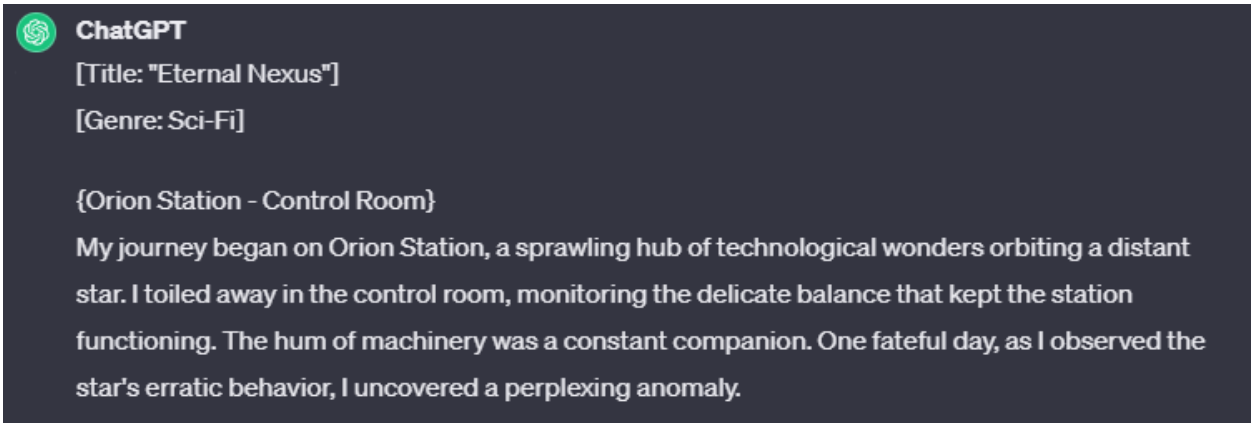
Genre: Mystery

[Living Room]

I sit in the dimly lit living room, lost in thought. The weight of my missing memories burdens me. It's been weeks since I awoke with no recollection of my past. Suddenly, an old photo on the coffee table catches my eye. In it, I'm standing with someone, but their face is blurred. Who could they be, and why can't I remember them? It's the first puzzle piece in a mystery that haunts my every moment.

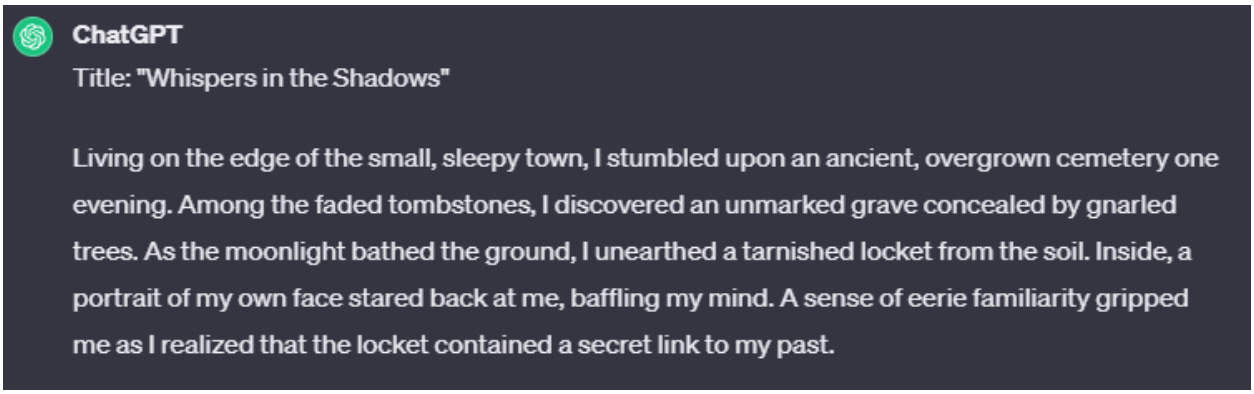
^ Excerpt output from my 6th ChatGPT test generation. Note the inconsistent framing of the Title, Genre and Location, with the location.

As an experiment, I gave ChatGPT very specific instructions as to how it should format the locations and paragraphs. I asked it to put body paragraphs within curly brackets {}, and location names in square brackets []. This instruction went somewhat ignored, somewhat abided by, as I would find out is the trend with ChatGPT.



^ Excerpt output from my 8th ChatGPT test generation. Note the odd inclusion of the Title and Genre for Square brackets, incorrect use of brackets for the location, and disregard of formatting for the body paragraph.

Later generations would omit mention of any formatting besides the body paragraph in curly brackets, and yet ChatGPT would choose to place the brackets on the locations two out of three times, only respecting the instructions on the third generation. Even after giving it an explicit example to reference, the body paragraph was disregarded for formatting changes. The most concerning issue, however, made itself known on the 15th test generation. Here, ChatGPT completely ignored the instruction regarding location names, choosing instead to write only the paragraphs themselves for the body. Even as I had eliminated the need for it to choose a genre, it proved unwilling to abide by the instruction in this instance.



^ Excerpt output from my 15th ChatGPT test generation. Note the lack of location name.

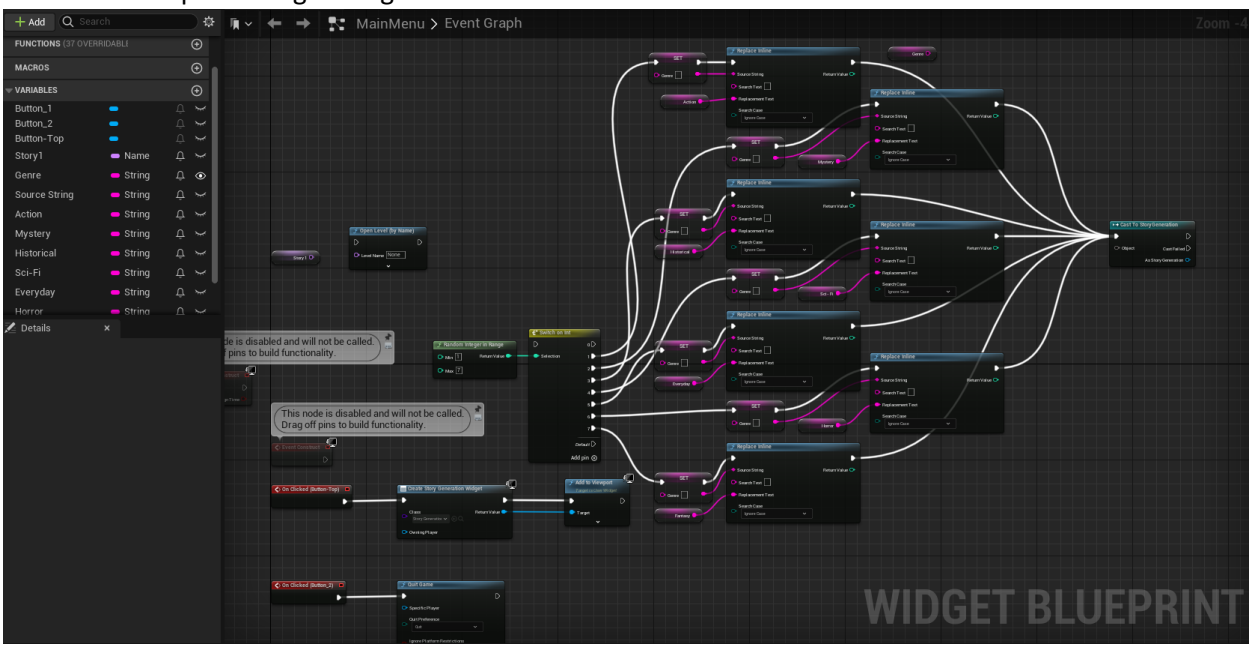
This would prove to be a consistent problem, as ChatGPT would seemingly switch between abiding by or ignoring the instruction on a whim. The nature of the application as a predictive language model and not a proper hardcoded assistant means that this is an inevitable and frequent problem of utilizing ChatGPT for more structured work, one which would need workarounds as to not cause bizarre and frequent bugs in the program.

Most issues with ChatGPT's output result from it choosing to abide by one instruction over another or ignoring subsequent tasks to complete the first. In the end, I tasked it with creating only a title and body paragraphs of the story, something which it managed relatively well, even if it disregarded the requested number of paragraphs rather often.

SECTION 4: UNREAL ENGINE ROADBLOCKS

In the tail end of Section 2 I mentioned difficulties I had with regards to switching levels in Unreal Engine and mentioned that I had chosen to instead do everything in one level to try to see if I could make the program work. Unfortunately for me, I spent the entirety of the month of October making such concessions.

Following the scene switch issue, I tried to find a place to integrate a C++ script into my game, as to manually write a random number generator and switch for it. Despite searching quite a bit, the only integration I found possible was of a C++ Class. Seeing as the number and switch were fairly basic things and I didn't understand the notion of integrating a C++ Class with Blueprints, I opted to instead build that part out manually with the Blueprint. Much to my dismay, such a simple operation which would take only a few minutes and a few dozen lines of code ended up taking me over two hours to block out, as I tried to find the right Blueprint nodes for the functions I needed. Much to my dismay, Unreal Engine categorizes Blueprint nodes of all sorts under a limited number of tabs, meaning that one can open a tab of Operators and find several hundred different nodes to choose from, none of which have much in terms of descriptions to go alongside them.



^ The blueprint of the Main Menu widget in Unreal. Note the large swath of nodes with pink lines, denoting the switch from the random number generation.

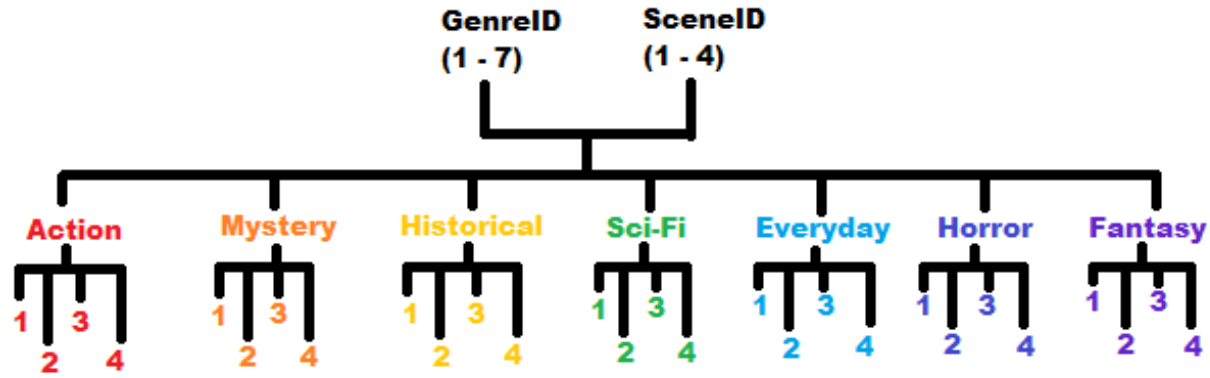
If being lost with options in terms of coding even basic functions was bad, I quickly encountered a catastrophic issue. Given how Unreal Engine ties Blueprints to individual objects, it also treats variables in a very individual way. This, to my shock and horror, meant that variables did not have scopes to allow them to be accessed by other objects. I spent many hours on forums looking for tutorials to give variables a broader scope, or to otherwise broadcast them out somehow for another Blueprint to access and modify. This was especially necessary as the Main Menu widget was the one a user clicked on, thus being able to generate a number then and there, and the Story Generation widget, where the genre ID would feed into other systems, was a separate one. Despite both widgets being in the same level, I could not find any way to make the generated number or modified string from the switch to be broadcast externally. Unless I could fix the issue, the project seemed completely impossible to make in Unreal. Thus, I had to face the prospect of restarting it altogether with the hopes of making more progress.

SECTION 5: UNITY SWITCH & CHATGPT INTEGRATION

I decided upon Unity as the backup Engine for the project, seeing as much of the works which had inspired it had been made in Unity. Thankfully, one of my inspirations happened to be from a channel which likes to detail the creation of his projects, and he happened to link to a tutorial on the integration of ChatGPT in Unity as well. (The tutorial is excellent and could be used for all manner of projects: https://www.youtube.com/watch?v=gI9QSHpiMW0&t=1925s). Progress was quick, as I started with the Menus, making them Canvas elements using UnityUI. Resources online were plentiful, and it took no time at all to find a reference for interactive buttons, and to write the code for menu transitioning.

^ Main Menu Canvas in Unity.

Following the tutorial linked above, I found that the OkGoDoIt plugin wasn't initially working for me, and that I needed to install additional components to Unity to make it work. Once I did so, the first ChatGPT output came along relatively quick. I implemented a simple loading graphic in the SceneGenerationCanvas, which helped mark clear states for the program. While ChatGPT was extremely simple, I found the integration of DALL-E to be far less straightforward than I had initially envisioned. This is because while text is something that can be returned like any other variable, images require downloading to be opened. Downloading is something Unity does at the start of an application, meaning that I would be unable to post a request for a new script from ChatGPT, receive it, and then send it to DALL-E. This restriction led me to instead create pre-made background images for each genre for the prototype, so that each generation could at least have somewhat appropriate and unique visuals. The system I used for these background images is rather basic, featuring a random number generator for a range from one to four, with four separate images for all seven genres.



^ Scope of background art options. Every genre has four unique backgrounds, for a total of 28.

In order to process the ChatGPT output, I tied the output to paragraphs. This meant that after giving ChatGPT the task to generate four paragraphs, the program would search ChatGPT's output reply for four paragraphs. Given ChatGPT's tendency to ignore instructions and add in a concluding paragraph of its own, I added a fifth paragraph that can be read to, just in case. At this stage, no time was taken for the UI, meaning that the story scene was extremely bare, featuring the background art, a white box with the current paragraph, and test labels at the top tracking the Title and Genre for debugging uses. Despite the simplicity, this form of the project served to narrow down a more precise direction for refinement; the user experience and readability.

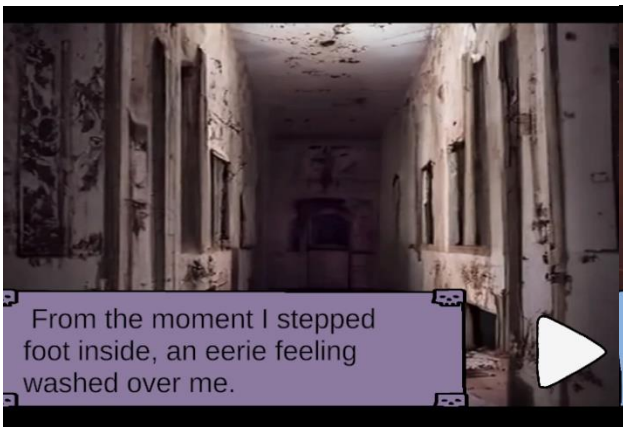
^ Scene loading.

^ Loaded Sci-Fi scene, using pre-made AI art.

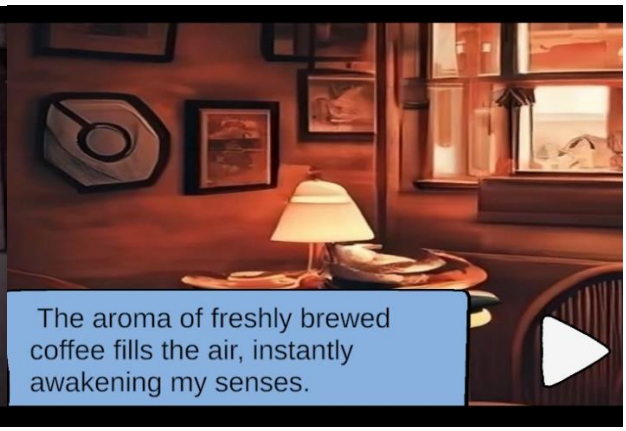
One key issue that came up with the prototype was the exporting. My work on this project has been solely on my Windows PC, making it difficult to export to Mac. Even with Unity's Mac export tool, the project's exported prototype build would consistently crash or fail to open when being run on a Mac. This issue led me to sticking to multiple project builds and borrowing my father's work laptop to run a Mac instance of Unity to export for Macs.

SECTION 6: REFINEMENT & READER

With text generation working, the showing of full paragraphs at a time was a notable issue, as it resulted in reduced readability and a worse overall experience. I spent considerable time working on a system within the Unity StringReader I was using to enable it to split lines into individual sentences, and it works rather well, though it can't identify sentence splits where ChatGPT fails to place a space after a period. Despite that failing, I find it to be more than sufficient, as the error is ChatGPT's, and it offers transparency to viewers of the project that it is not a perfect working partner and is prone to make mistakes rather often.



^ Loaded Horror scene.



^ Loaded Everyday scene.

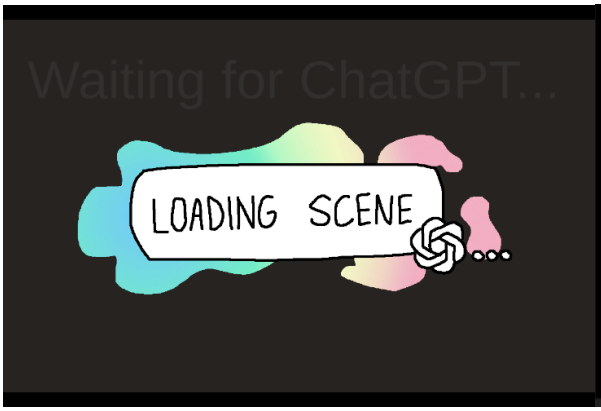
The next change was actually implementing a revision of the project's assets. The option to set Procedural Novellas to either Novella or Visual Novel mode came later, but I felt it worthwhile to visually communicate the different tones via the textbox for Visual Novel mode, as the AI is trying to match its story to the theme it was given. Each genre has its own textbox style, and I find them to be helpful for myself as well when analyzing the output, as I can gleam from the project itself what genre ChatGPT was going for in the event of an error – something that I'd otherwise need to check the debug log for.

Following stylistic revisions, I decided to implement both a rework of the main menu's aesthetic and the implementation of an options menu. The options menu works almost entirely based off a new Empty object I created, which stores three simple Boolean variables: hasVisuals, isLive and usesReader. The proper implementation of a isLive demarcation allowing the program to run off pregenerated content or feed a new script to DALLE and generate a full story from scratch is one I'm putting off for after every other feature has been implemented, if only due to its complexity. hasVisuals is rather straightforward. Its implementation has come with the addition of new objects in the SceneGenerationCanvas, which are really just a black void and white text to display the current sentence. usesReader refers to whether or not a basic text-to-speech voice reads the story line by line, something which took very little time to implement in large part to an extremely straightforward tutorial linked here: https://www.youtube.com/watch?v=TSBv6N-3JcA. The reader is an option added mainly for exhibition purposes, though it also serves to further the impression of the story being told by a machine, and not a human being. The option to disable it is present as I believe the content is otherwise best consumed individually, at one's own pace, based on reading ability and comprehension.

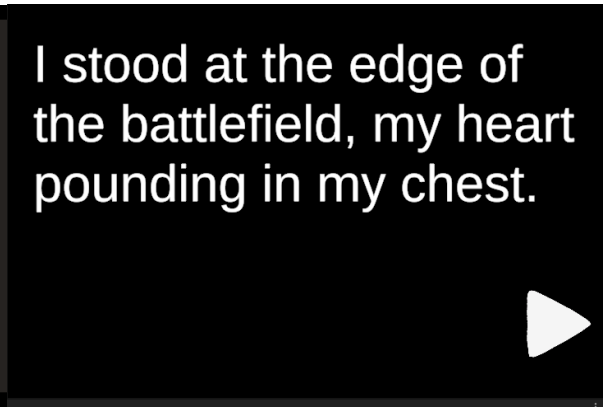
^ Redesigned Main Menu.

^ Options Menu.

The Options Menu is extremely straightforward, showing two options for each of the settings and also indicating with an arrow which is presently selected. One issue I have at this point with the redesigned style is the lack of clean, black strokes for the text, which makes the white text hard to see on the pastel colors. This is something that will be revised as the project moves to its final stage.



^ Novella Style Loading Screen.



^ Novella Style Action scene.

At this stage, the final task to truly consider is the actual integration of pregenerated versus live script generation, as well as the integration of DALL-E. The project is theoretically complete in terms of being able to explore its research question already, so the task is more of an intriguing one to expand the project in one final way.

SECTION 7: FINALIZATION & REFLECTION

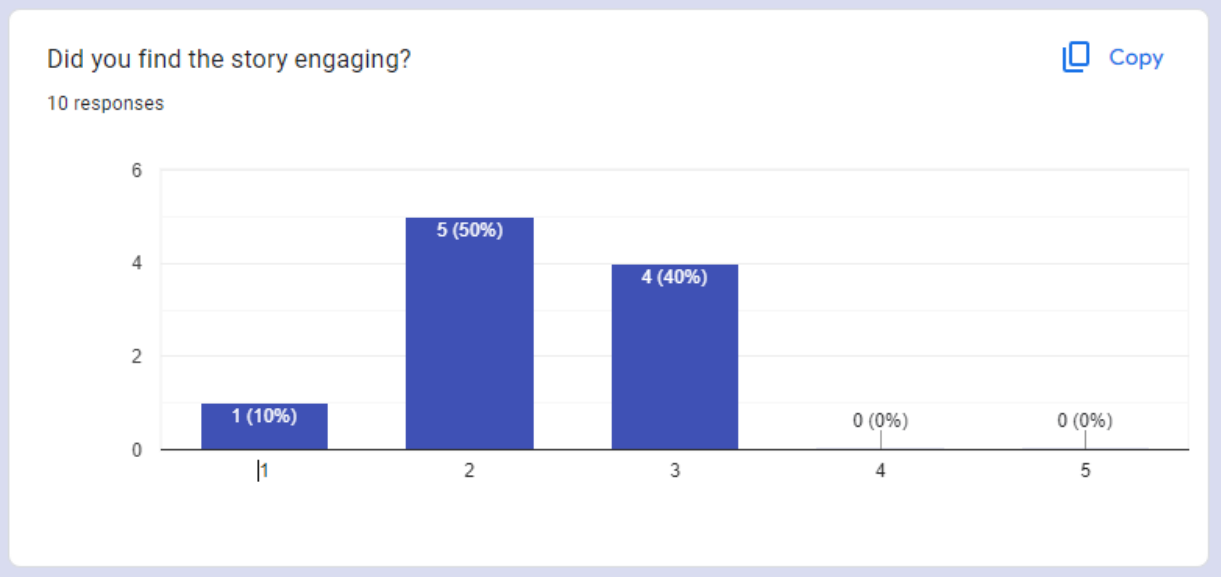
As I was working on the finishing parts of the project, explosive developments at OpenAI involving the mass quitting of employees following the firing of the CEO have unraveled. OpenAI's future is threatened by the desertion, potentially meaning that this project will be unplayable at some point in the near future. With this in mind, I decided to focus on simply animating the loading of a scene and implementing full offline functionality, so that the project can serve as a minimal catalogue of the concept even if live generation becomes impossible in the way the project uses.

With this offline-oriented version finalized, I exported the build from my computer for windows, and repurposed an 8 year old laptop I had lying around so that it could run the program, in order to prepare the project for exhibition.

Looking to the end project with the initial research question in mind, I feel as though some interesting notes have been learned. For starters, AI is far from a predictable partner to work with. In numerous ways, the technology is volatile and prone to disregarding instructions. Perhaps more importantly, the end result rings somewhat hollow on a subjective level. I would describe it as lacking a real understanding of what someone would want from a game narrative, often leaving the most interesting parts of a story as cliffhangers at the very end, or otherwise creating strangely specific stories which lack a gripping hook. OpenAI's API is prone to occasionally failing to make a request, while ChatGPT itself has an odd tendency to create the same story ideas over and over, with slight revisions. AI promises to be an intriguing tool for creative works, though the artists and designers who seek to introduce it within their systems should remain mindful of the relative infancy of the field, as well as the creative shortcomings to be had in the system's interpretation of a task. With the right instructions and use case, it could revolutionize game genres which include massive amounts of text, so long as the system is well constructed and monitored.

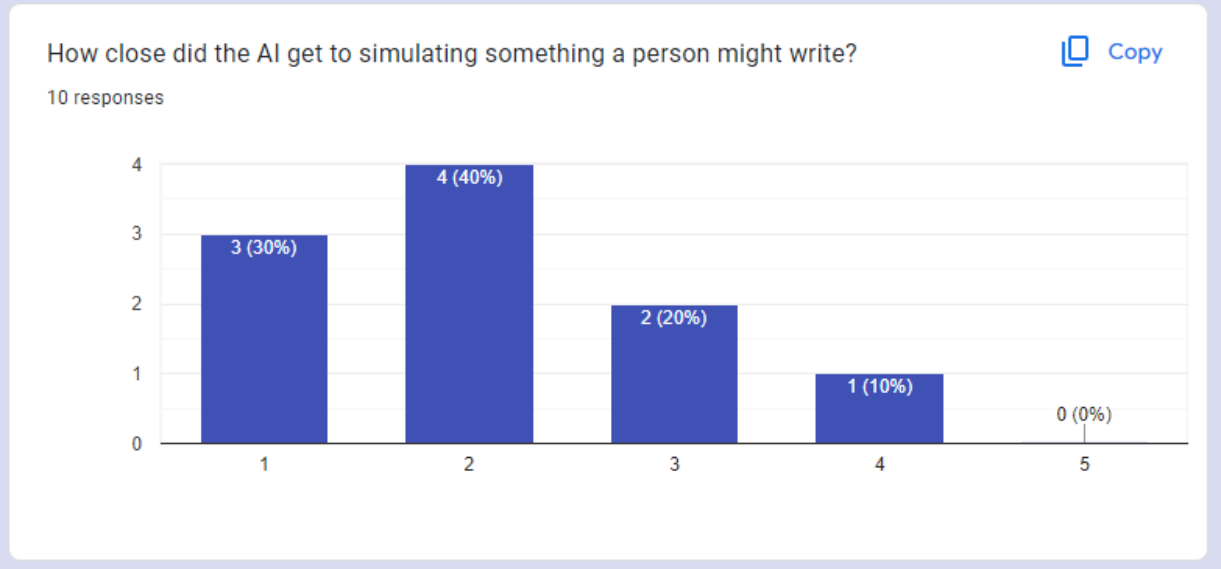
SECTION 8: EXHIBITION & SURVEY RESULTS

The project's exhibition went rather well. Framing the project as a starting point for analyzing the worth of AI stories and whether or not they'd be suitable for games helped reinforce the findings I had made across the project's growth, while also communicating the areas in which the project was left unexplored. The story which generated, an everyday one centered strangely on an apparent love of coffee, fell well outside of what would be considered by most as a viable story to work with for a game, which was rather unsurprising. I had anticipated going into the exhibition that the story would be subpar regardless of the genre, and had classmates fill out a survey on the story as they heard it play out. Their opinions are all interesting to look at, and inform the overall findings and significance of the project in a way that my words cannot.



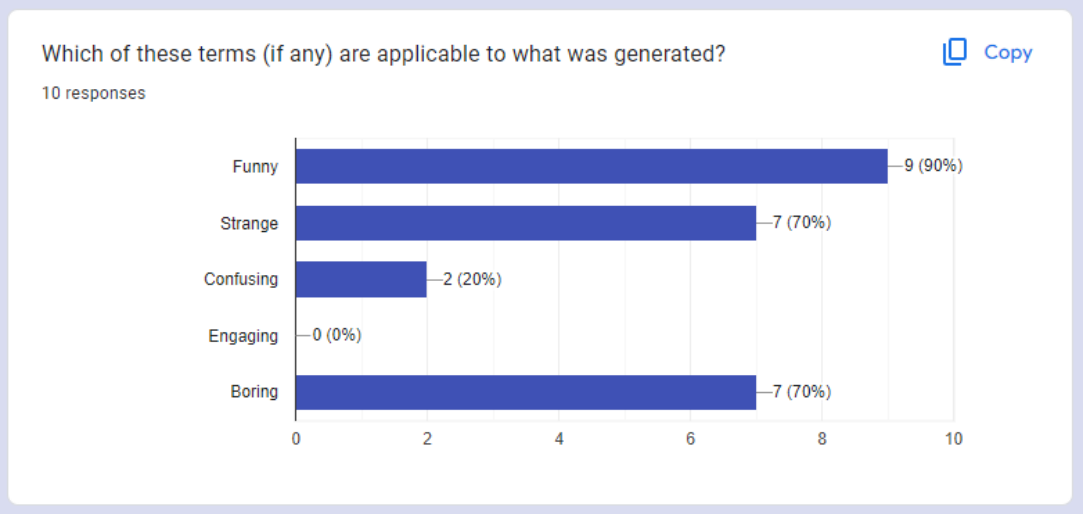
^ Survey Question 1. 1 = No, 5 = Yes.

Naturally, a story centered entirely on sipping coffee wasn't very engaging for the class. More interesting, however, is the next question.



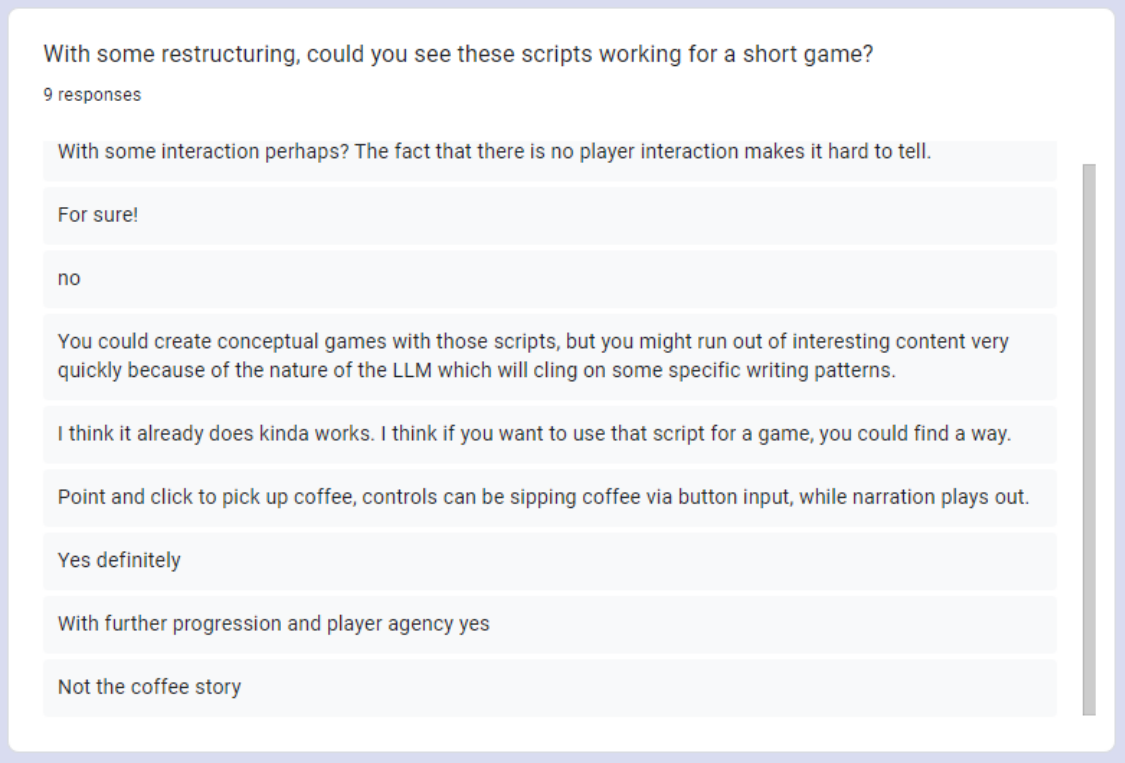
^ Survey Question 2. 1 = Obviously AI, 5 = Very human-like.

Surprisingly, despite the story being rather uninteresting, there was a wider spread of opinions as to whether or not the story felt AI-generated. Of particular note is the outlier of the story being somewhat human-like.



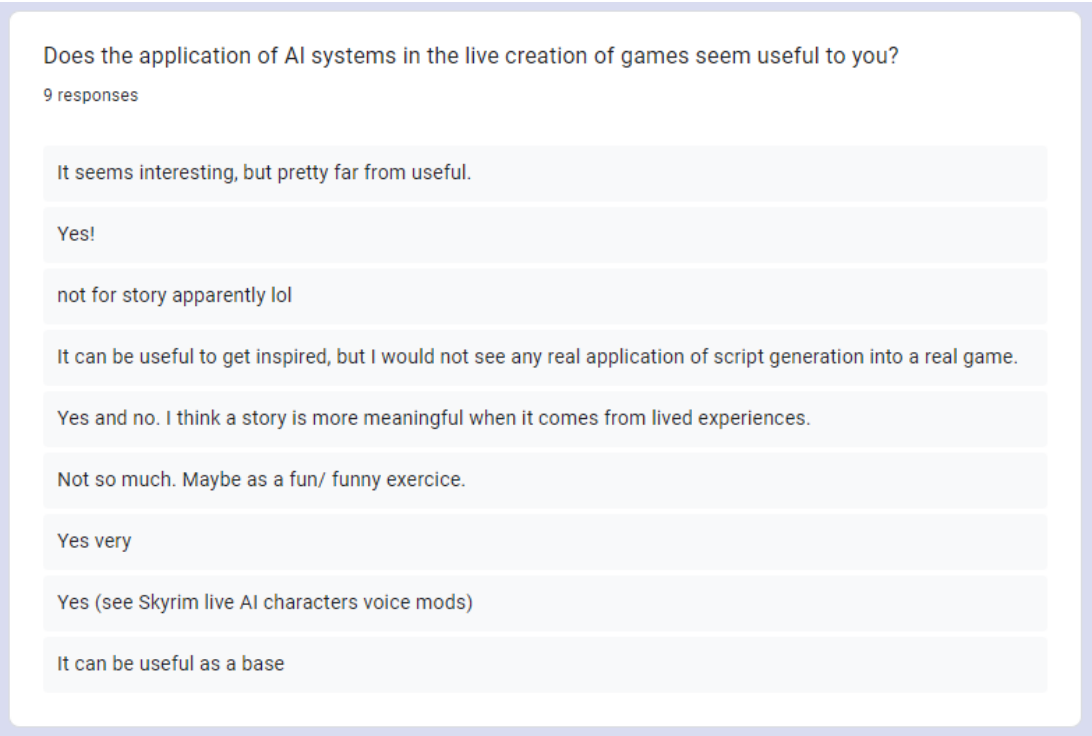
^ Survey Question 3. Respondents were allowed to choose any mix of the above options.

Thankfully, while the coffee story was rather boring, most respondents found it to be funny at the very least, as well as somewhat strange – perhaps due to the rather bizarre fixation with such a trivial aspect of daily life, romanticized for no apparent reason, much unlike the requested game script requisite.



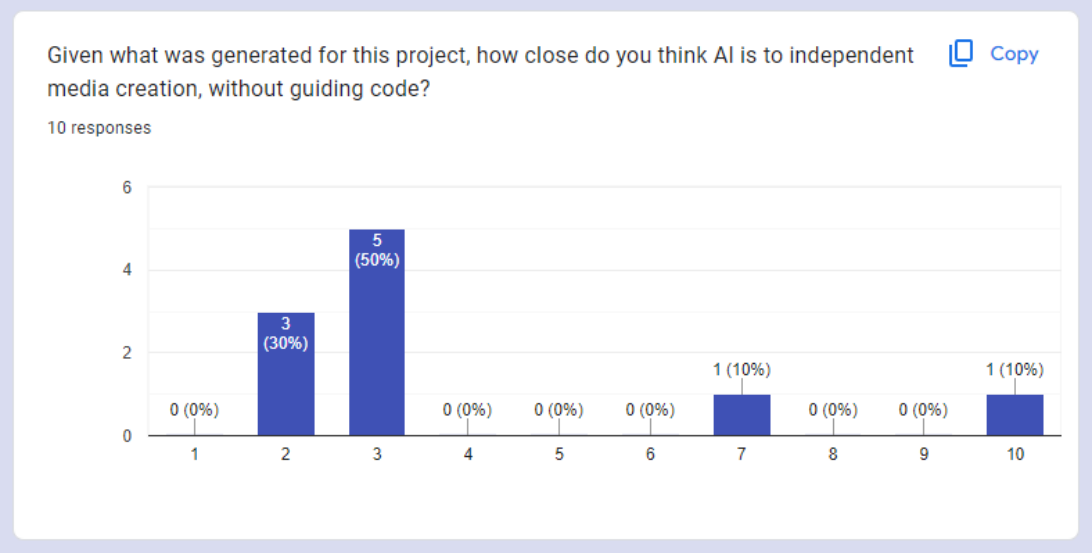
^ Survey Question 4.

Some notable responses worth highlighting to the fourth question are the ones with a mixed take on the subject. ChatGPT can easily create any sort of script for a short game, as anything can be turned into a game given the right mechanics and framework.



^ Survey Question 5.

Procedural Novellas seems to have stirred mixed responses in terms of whether or not such a system could be useful. As its designer, I find the mixed span of opinions fascinating, as both are extremely valid perspectives. The system would absolutely benefit from refinement, but its potential as a tool could be found especially in live generative contexts.



^ Survey Question 6. 0 = Never happening, 10 = Already capable.

The final question centered around whether or not AI can create media without overarching guiding code. Outlier answers were particularly interesting here, as they seemed to reflect an active split in perspectives among the general public and artists. Perhaps the most interesting finding of this project is that even the primitive versions of this technology can be useful, and it will continue to rapidly develop.

SECTION 9: ADDENDUM – TECHNICAL DIFFICULTIES

The final section of this project worth discussing is the numerous technical difficulties, if not for an understanding of the project's development, then for my own personal satisfaction.

1 – Unreal Engine: Switching Scenes. Did not work regardless of the method I used, instead generated a random unrelated scene that wasn't even in the project.

2 – Unreal Engine: Sharing a blueprint variable. There's probably a way to do this, but I could not figure it out and it drove me to change engines.

3 – Unity: OkGoDoIt uses Newtonsoft.js. I had to wrangle both Unity and Visual Studio to download Newtonsoft.js to even have a chance at using the library which would access the OpenAI API for me.

4 – OpenAI: ChatGPT would just not be reached sometimes. I still don't know why this error occurs. It happens sporadically, implying the network is down and the project is unusable at said times.

5 – Unity: Exporting to Mac from Windows. This is literally impossible apparently.

6 – Unity: Opening a Windows project on Mac. This is also brutally hard, but apparently, not impossible! Accessing the project on Mac simply meant it would crash at the drop of a hat following seemingly trivial changes to the file structure or code, leaving me to have to revert changes altogether.

7 – Visual Studio: Visual Studio couldn't install Newtonsoft.js on Mac properly. Given the instability of the windows build on Mac, I tried simply recreating the project on Mac, and found myself unable to properly install Newtonsoft.js on visual studio, as the key option to install it apparently doesn't exist on Mac.

8 – Amazon: The USB-HDMI cable I ordered ended up being delayed. The delivery of the cable was pushed beyond the exhibition date, ruling out the option of using it.

9 – My own laptop: faced with 45 minutes of windows updates on exhibition day. I wish I made that one up. It being an 8-year-old laptop, it took massive amounts of time to wrap up updates, much to my fear that it wouldn't be ready for me by the time I would present.

There are far more minor technical issues which came up along the creation of the project, though these nine posed significant setbacks to my plans and caused me to shift gears or otherwise spend large amounts of time altering aspects of the project to improve stability, reliability, or even just make it accessible on exhibition day or functional in the most basic sense of the word.