Exploration of Generative Machine Learning in Interactive Media

A CONTEXT AWARE GENERATIVE NARRATION PROTOTYPE

AALBORG UNIVERSITY

MTA8_GR02

MAY 29, 2024

This page is intentionally left empty

AALBORG UNIVERSITET
STUDENTERRAPPORT

8th Semester
Aalborg University
Rendsburggade 14
9000 Aalborg
http://www.aau.dk

Title:
Exploration of Generative Machine Learning in Interactive Media:
A Context Aware Generative Narration Prototype

Theme:
Immersive Experiences

Project Period:
01/02/2024 - 29/05/2024

Project Group:
Medialogy 8th Semester, Group 3

Participant(s):
Arlonsompoon P. Lind
Jonas B. Lind
Mads W. Sørensen
Rasmus V. Jacobsen
Sebastian Whitehead

Supervisor(s):
Ivan A. Nikolov

Copies:

Page Numbers: 60

Date of Completion:
29/05/2024

Abstract:

Language models among other generative models are becoming a larger part of people's everyday lives, assisting in work and entertainment. This has lead to a need for more research in classifying and understanding how people interact with these generative models. This paper thus explores the generative artificial intelligence field, proposing a new expandable taxonomy structure categorising them into content transformers, content describers/transcribers, content generators, and large language models, thus creating a better understanding of their diverse applications. Through this research, it was found that LLMs as storytellers were of particular interest, however not enough research has been done focusing on how users react to real-time storytelling by AI. Through preliminary tests, acceptable delay thresholds were identified for context aware narration in virtual experiences, indicating that delays longer than 3 seconds were considered unacceptable. However, further testing indicated, that users had a higher delay acceptability threshold when the object interacted with, was perceived as important. This testing also indicated that breaking the continuity between interaction and narration immediately caused users to perceive the narration negatively. In the implemented environment, a user interacts with objects with associated descriptors. This context is provided to a large language model, resulting in text that closely resembles human speech. This text is then inputted into a text-to-speech model to generate audio for narration that is presented to the user. Between test and post-test, interviews indicated users felt that generation time were longer with negative narration styles, yet statistical analysis reveals no significant difference in generation time between positive and negative narrations. This suggests that tonality and an object's perceived importance plays a role in user perception of the system.

Preface

This document is designed to serve as a supplementary resource rather than a replacement for a paper created for the 8th semester at Aalborg University Medialogy (**Exploration of Generative Machine Learning in Interactive Media, A Context Aware Generative Narration Prototype**), focusing on the themes of "Immersive Experiences". The authors earnestly appreciate any constructive feedback, suggestions, or corrections from readers.

Contents

Chapter 1

Introduction

Generative artificial intelligence (GenAI) is transforming various industries by automating tasks and creating enhanced user experiences. From hospitality to healthcare, and from education to virtual reality (VR), the applications of GenAI are expansive and varied. However, alongside these advancements come ethical considerations such as job displacement, trust, transparency, and the need for human oversight. This paper explores the impact of GenAI across different sectors, highlighting both its potential and the challenges it presents. Taxonomies for GenAI, will be studied as well as redefined to fit into a 2024 context.

The study also examines user tolerance to system delays in GenAI applications, especially in interactive media. Through preliminary tests, the research identifies acceptable delay thresholds to optimize system responsiveness. Additionally, the paper discusses the design and evaluation of a network model for real-time user interaction in VR, focusing on the balance between technical performance and user experience.

By providing a comprehensive analysis of genAI applications, challenges, and future directions, this paper aims to contribute to the understanding of responsible AI integration. Emphasising ethical considerations and user-centric design, the research underscores the importance of creating transformative yet sustainable technological solutions.

Chapter 2

Background Research

The following background research is conducted as of cut-off date 19/02/2024 Some exceptions may apply, however publications after this date have generally been ignored

2.1 General Exploration of GenAI

In the last two years, there has been a surge of large generative models such as ChatGPT[OpenAI, 2021] and Stable Diffusion [StabilityAI, 2019]. These models are revolutionising several sectors by performing tasks such as general question-answering and automatically creating images from text prompts. Other models could be either Text-To-Image, Text-To-3D, Images-To-Text, Text-To-Video, Text-To-Audio, Text-To-Text, Text-To-Code etc. [Brizuela and Merch, 2023].

2.1.1 A taxonomy of the modern generative artificial intelligence field

The modern GenAI field spans a wide range of approaches and use cases, though multiple models overlap in capability. In a state-of-the-art review paper by [Brizuela and Merch, 2023] a structure of nine categories is described, each grouping models by their input and output parameters.

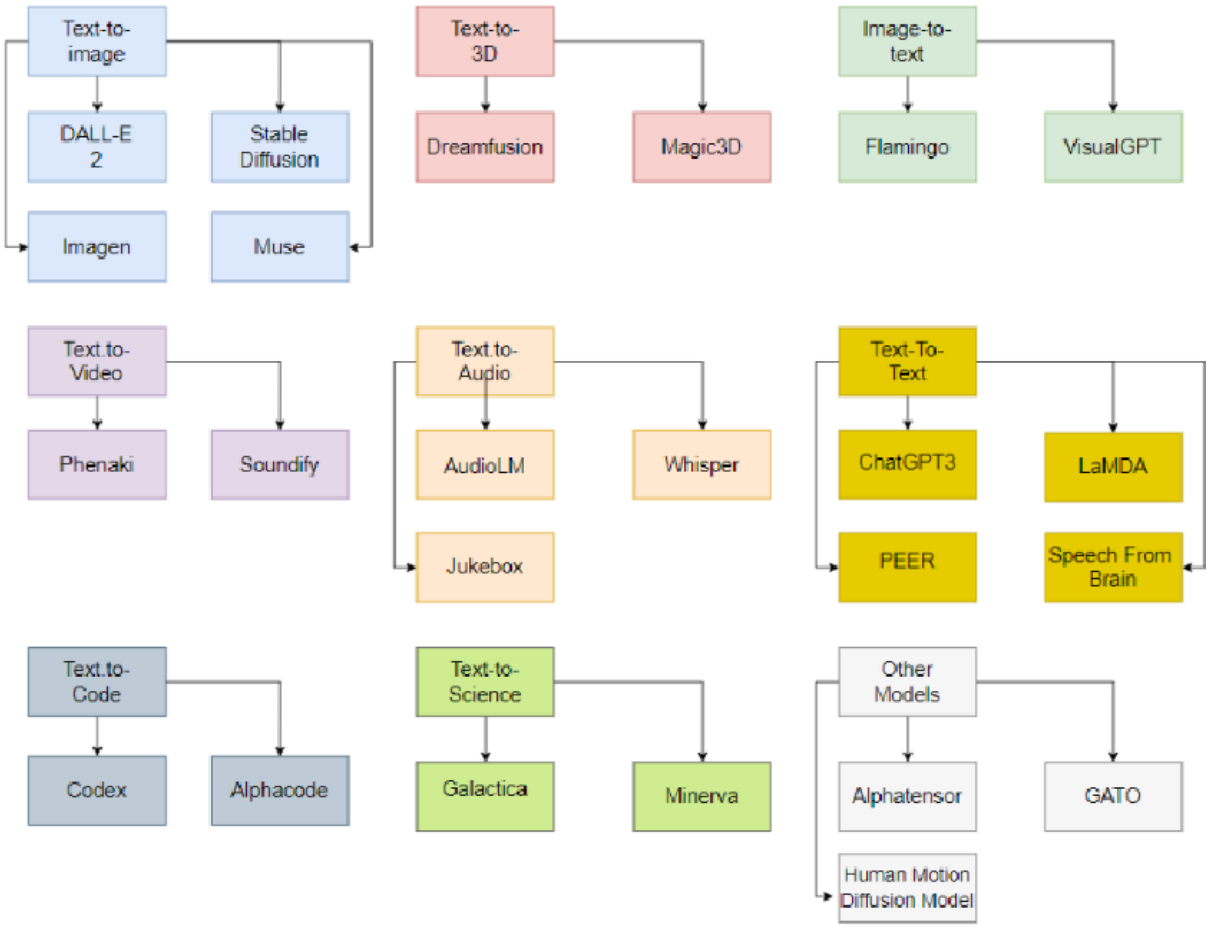

Figure 2.1: The structure of 9 categories, described by [Brizuela and Merch, 2023] structured by the input and output parameters of a given machine learning model.

This structure of categorising models by their input/output formats allows for comparisons to be drawn between models and their approach. Despite this paper only being a year-old, models in these categories have already progressed dramatically. For example, where the paper describes *DALL-E 2* the use of *DALL-E 3*, which vastly improves on the downfalls of its predecessors, is now common place. Based on this structure, we explore state-of-the-art models referencing the applicable categories when possible.

SOTA - Models

This section explores a variety of GenAI, the list is likely not exhaustive but is meant to provide an overview of different opportunities in this space.

- **TEXT to TEXT**
  - **GPT-4:** A large language model (LLM) capable of mimicking human speech, developed by [OpenAI, 2021].
  - **Megatron-LM:** Megatron is a highly optimized large transformer model developed by Nvidia, used to train large language models [NVIDIA, 2024].
  - **Google Gemini *(Bard)*** "Bard is a new tool that you can use to explore creative ideas & explain things simply. It's a Google ML experiment that can generate text, translate languages, write different kinds of creative content & more" [Google, 2023].
- **TEXT to IMAGE**
  - **Mid Journey:** "*Use the Midjourney bot to generate stunning images from simple text prompts in seconds. Work directly in Discord. No specialized hardware or software is required*" [Midjourney, 2023].
  - **Stable Diffusion:** Stable Diffusion, is a text-to-image generation deep learning model, which is capable of creating realistic illustrations of a user's prompt [StabilityAI, 2019, Diffusion, 2023].
  - **DALL-E:** DALL-E 3 is the newest iteration of the DALL-E technologies, this generation aims at remembering every part of the prompt users provide, reducing the need for users to learn prompt engineering [Betker et al., 2023].
- **IMAGE to TEXT**
  - **GPT Vision:** Vision allows for analysis of images, which can then be explained with text [OpenAI, 2023].
  - **Actavision:** Actavision provides a powerful and comprehensive suite of image analysis and understanding capabilities [Astica, 2023].
- **TEXT to AUDIO**
  - **Eleven Labs:** ElevenLabs is an advanced text-to-speech model, with a vast array of pre-trained voices to choose from. It is also capable of generating completely new voices as well as cloning real voices [ElevenLabs1, 2024].
  - **GPT4 TTS:** GPT4 TTS is a text-to-speech model capable of converting text prompts into natural sounding voices. Two different models are available to use, one capable of real-time conversion, and one optimized for quality. [OpenAI, 2024]
- **TEXT to VIDEO**
  - **Pika.art:** A text video model capable of also image to video, and video to video generation, this model can complete a variety of task. These include extending video duration to up-scaling video on top of its generation capability. [Pika.art, 2024]
  - **Sora:** A cutting edge text to video model by Open AI capable of generating video many would find indistinguishable from actual video. Beyond the text to video capable, this model is also capable of image to video and video to video [OpenAI, 2024].

- **TEXT to CODE**
  - **AlphaCode:** Functioning as a sub model to Google's Gemini Model [Gemini Team and Google, 20 Alpha code uses an internal iterative process to generate, evaluate, adjust code when called upon by the aforementioned large language model. [Chen et al., 2021]
  - **GitHub Copilot:** A coding assistant which takes text input and produces and suggests code snippets specific to the context provided in a description, comment, or as a suggestion in the context of existing code. This model was trained on code from every "large" publicly available repository and a number of stack overflow threads. [GitHub, 2021]
- **OTHER MODELS:**
  - **Shap-E:** Shap-E is a text-to-3D model capable of generating both textured meshes and Neural Radiance Fields (NeRFs) from simple text prompts in seconds [Jun and Nichol, 2023].
  - **LLaMA 2:** The second iteration Meta's attempt at a large language model similar to the likes of OpenAI's GPT, but capable of running locally given the correct hardware. [Meta, 2024]
  - **Meshy:** Meshy is a text to 3D and text to 2D generator, capable of creating 3D models in minutes based on a user prompt. [Meshy, 2023]
  - **Otter.ai:** Otter AI is a transcription tool able to transcribe and transform human speech (for example during a meeting) into text and notes [Otter.ai, 2018].
  - **Whisper:** Whisper is a highly accurate speech recognition tool capable of recognising, transcribing and translating multiple languages into English, implemented as an encoder-decoder transformer [OpenAI, 2022].

Given the ever expanding field of machine learning, a massive number of models have come about and have led to the above taxonomy structure becoming outdated. This has in turn lead to more and more models having to be classified under the "*other models*" category. This presents a need to re-evaluate this structure and create an alternative that expands and extrapolates as needed. See section 2.3

2.2 Related work

In the rapidly evolving landscape of modern technology, GenAI and machine learning have emerged as transformative forces across diverse sectors. The integration of AI-driven systems in these sectors promises to revolutionize processes, enhance experiences, and drive innovation. This section provides a brief overview of the multifaceted applications and implications of GenAI. This section explores the field, aiming to create an understanding of the breadth and depth of the field, while identifying opportunities for future research and development.

Using Generative Artificial Intelligence in-

2.2.1 Hotels

An article by [Wang, 2024] studies the use of GenAI in the hospitality culture of New Zealand. In hotels across the world, robots are increasingly replacing human greeters, resulting in the loss of jobs. Two strands of research have been diverging from each other, one focusing on the positives and one on the negatives of using GenAI to replace human workers [Wang, 2024]. Historically, service work has been seen as a difficult field for human workers to be replaced [Autor and Dorn, 2013]. However, recent developments in ChatGPT and other similar technologies have led to this claim being heavily contested.
The article finds multiple problems in simply implementing and replacing human workers with robots when wanting to provide a personalized experience for guests. This is further complicated by nuances in cultures and less spoken languages, where models currently struggle with correct translation [Wang, 2024].

2.2.2 Museums

[Hettmann et al., 2023] explore the challenges museums have faced in recent years regarding attracting the younger visitor audience, using cases created by modern ML models, such as chatbots. More specifically, [Hettmann et al., 2023] mentions the use of ML to automatically curate items to be presented in museums. Curation is an important part of museums and [Hettmann et al., 2023] propose to make all items in the museum's inventory accessible, despite these items not being a part of the original exhibits, by using the large language model(LLM), GPT-3.

The goal was to enhance the museum experience through the use of ML to make the artwork more interesting and engaging. This experience brings forth a scavenger hunt, connecting the exhibits, and making them more accessible and playful. This was done by having two characters guide visitors through the story, a puppy, and a humanoid robot. The visitors were then presented with challenges, which are solved through minigames.

Through the testing of the proposed systems the researchers found that when presented with pre-formulated, fact-based- and generated partially fictional -texts, participants were able to distinguish between the two, which is seen as crucial to the trust and authenticity of the institution. By weaving these engaging stories in individual pieces, [Hettmann et al., 2023] found that the visitor's interest is enhanced and thus deemed the system successful.

Figure 2.2: A Participant receiving a trophy for completing the scavenger hunt. [Hettmann et al., 2023]

2.2.3 Healthcare

GenAI has also been used with great success, in healthcare, providing revolutionary benefits in the sector [Shokrollahi et al., 2023]. Specifically transformer and diffusion models as these play a pivotal role in image-image translation, image segmentation, image reconstruction and many more. These models have been used to improve clinical diagnoses and drug synthesis. The paper by [Shokrollahi et al., 2023] creates a comprehensive overview of models in the healthcare sector.

Due to the "black box" nature of deep learning models, it is hard for doctors to understand how the model came to its conclusions, this can lead to mistrust of the model [Roundtree, 2023] further work into transparency, explainability, and fairness is needed.

[Jørgensen, 2024] interviewed a Danish physician, stating (translated from Danish):

> *"This technology has helped reduce our patient's waiting time by an hour, by increasing the effectiveness of our patient flow."* - Christian Pedersen, Chief physician, Aalborg University Hospital.

Pedersen further states that the ML model used to help in identifying fractures on x-ray images have an accuracy of 95-98% which means it is better than the average doctor, however the very best radiologists still outperform it.

As mentioned by [Roundtree, 2023] the "black box" also has an impact on the way [Jørgensen, 2024] uses their model. They use the ML to make a prediction taking around 1 minute, then, the image is also analysed by a board of doctors to confirm the diagnosis before patient treatments starts.

The results of [Roundtree, 2023] show that GenAI can be used as a tool set for doctors, but is not strong enough to provide diagnoses or create treatment plans by itself. It should be seen as a tool similar to a Google search, where the doctor still needs to evaluate on the results found themselves. However, the researchers also recognise how new this field is, and therefore the results could change rapidly.

2.2.4 Education

The sector of education has seen drastic changes since the introduction of GenAI. A paper by [Chiu, 2023] explores how teachers navigate the new technology, in terms of learning, teaching, assessment and administration. For example, mid-journey 2.1.1 can help give art students fast feedback on their work, acting as a teacher or tutor.

The results of the paper by [Chiu, 2023] finds that teachers generally believe students should be taught about AI in school, in order for students to be more informed about the tools they are using. Contrary to business/technology sectors, the administrative parts of the education system are still rejecting the technology, teachers however seem excited about the possibilities, as long as the tech is used responsibly [Chiu, 2023]. Therefore, changes to the curriculum should be changed, however more research into the field is needed.

Since generic skills of the students can be seen as subjective, teachers and administration agree, that the AI models cannot be used to assess the performance of a student, therefore teachers still need to perform examinations/tests of students in the conventional way, without the use of genAI to properly evaluate each student and teaching direction [Chiu, 2023].

2.2.5 Economy and finance

The use of GenAI and ML shows major potential as a new research paradigm in the field of economic and financial research, compared to traditional research methods. Zheng et al. [Zheng et al., 2024] presents an example of such a paradigm, utilizing GenAI to accelerate the pace of research, automating tasks such as data processing and analysis. They state that machines' ability to adapt to rapidly changing environments, such as the field of financial and economic research, is particularly useful.

However, concerns are also raised regarding the role of human decision-making in the process. If the machines fail to accurately predict unforeseen events, they may introduce unnecessary risks and uncertainty. Additionally, due to the speed at which the GenAI models work, researchers are wary of the possibilities of market manipulation instability [Zheng et al., 2024].

2.2.6 Virtual reality

Within the context of the Metaverse, *a persistent virtual environment that allows access to and interoperability of multiple individual virtual realities* [Merriam-Webster, 2024], GenAI technologies help immensely in providing immersive experiences for the user. Text generation models have been utilised to enhance conversational interfaces with AI-generated characters. Image generation models have been employed to create diverse visual content. Additionally, 3D model generation technologies have also been used to create realistic virtual objects, enhancing the Metaverse experience [Chamola et al., 2023].

Looking at a more specific use-case of GenAI in virtual reality, [Xu et al., 2023] study the application of GenAI in autonomous driving within vehicular mixed reality. [Xu et al., 2023] uses GenAI to synthesise unlimited conditioned traffic and driving data, which is used to improve driving safety and traffic efficiency. They found that this has been particularly helpful in overcoming the challenges often associated with large-scale traffic and diving simulations, which typically require costly and difficult data collection from the physical world.

Another use-case would be [Chheang et al., 2023] who, within the field of anatomy education, created a VR environment integrated with a GenAI virtual assistant. Their system allows the users to communicate verbally with the virtual assistant, providing the users with a more interactive, adaptive and informative learning experience. According to [Chheang et al., 2023], this approach has the potential to improve on current medical education by enhancing the understanding of the morphology, location, and spatial relationships of anatomical structures.

2.2.7 Conclusions - Related Work

Based on the above sections, it's evident that GenAI and machine learning (ML) are being increasingly utilized across various sectors, including museums, healthcare, education, economic and financial research, virtual reality, and autonomous driving.

Overall, while GenAI offers significant potential across these sectors, careful consideration of ethical implications, transparency, and human oversight is essential for its responsible implementation and continued advancement.

2.3 Re-evaluation of the taxonomy of Modern Generative AI Models

Given the developments that have occurred since the taxonomy structure of
[Brizuela and Merch, 2023] was created, we propose the following continuation of the structure. Reducing the number of overall categories to four. **1.** Content Transformers, **2.** Content Describers / Transcribers, **3.** Content Generators and **4.** Large Language Models. Each of these proposed categories attempts to describe the function of models that fall within it, allowing for better generalizability of the categories to include newer use-cases, such as *image to image* or *text to audio* models. *Note these categories are not hard bordered, as overlap may be applicable in some cases.*

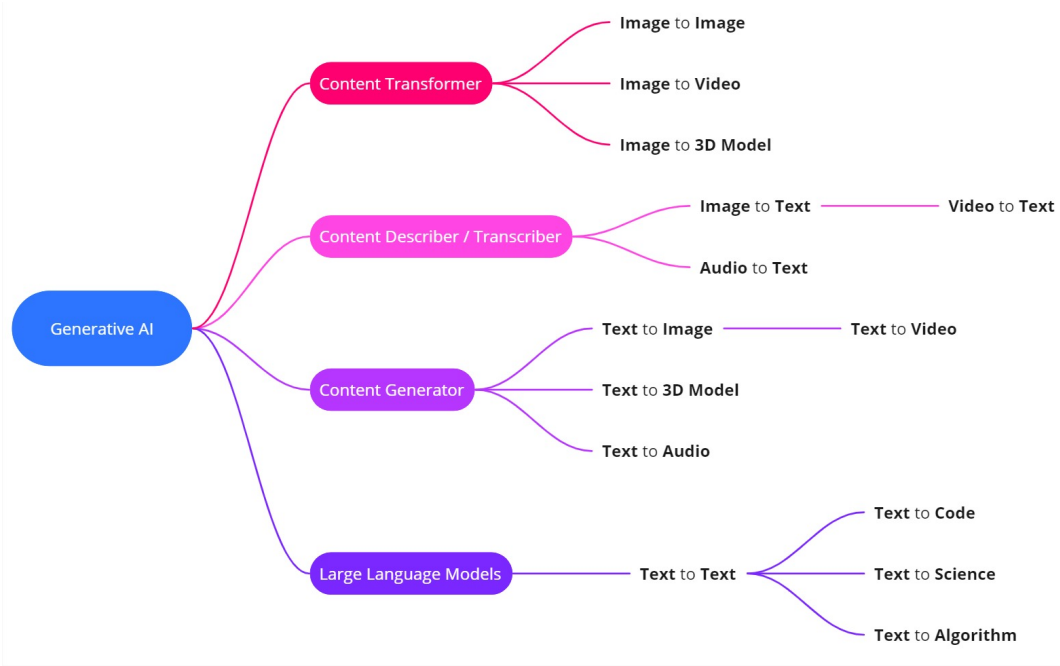

Figure 2.3: The proposed restructuring of categories described by [Brizuela and Merch, 2023]. Structured on the model's function over purely its input and output parameter formats.

Content Transformers

Models that receive one form of non-text media and transform it, either modifying it and maintaining the format, or producing, modifying the content and format to another. (fx. *Stable Diffusion* [StabilityAI, 2019])

Content Describers / Transcribers

Models that, given a piece of content, describe or transcribe its content, outputting text. Examples would be models capable of describing image content (e.g. *GPTVison* [OpenAI, 2023]) or models capable of transcribing human speech from audio to text (e.g. *otter.ai* [Otter.ai, 2018].)

Content Generators

Models that, given a descriptive text input, generate outputs in other formats. Models such as *DALL-E* [Betker et al., 2023] and *Midjourney* [Midjourney, 2023] fall within this category.

Defining - Large language Models (LLMs)

Considered Text to Text models, LLMs generally will attempt one of two functions. Emulating human conversational skills, and arguably a sub category to this function, receiving instructional text input and generating formatted content as output this would be text to code, text to algorithm and so forth. (e.g. *GPT4* [OpenAI, 2021] and Google's *Gemini* [Google, 2023]) Alternatively Models Such as GitHub's Copilot [GitHub, 2021]

Defining - Network of Models compared to Individual models

When looking at the state-of-the-art GenAI used today, a variety of models can be observed which produces impressive and realistic outputs. However, it is important to make a distinction between networks of models and an individual model, as they have different characteristics, advantages, and disadvantages.

A network of models refers to a collection of generative models that work together to produce complex outputs. As an example, a network could consist of a text-to-image model, an image-to-video model and a video-to-video model all called upon by a LLM. These networks leverage the strengths of different AI models and combines them to achieve diverse results and broaden use cases.

On the other hand, an individual GenAI model is a single model that generates one type of output from one type of input. As an example, a text-to-image model that generates an image from a text prompt, however it cannot generate a video or audio based on the same prompt.

The main difference between a GenAI network and individual GenAI models is the level of complexity and integration of the models. A Network of AI is more complex, as it involves multiple models that communicate and coordinate with each other. Individual AI models are usually simpler and more focused, as it involves a single model that operates independently, achieving a singular goal.

2.3.1 Application of the proposed structure to SOTA models

Given the aforementioned structure proposed in *Figure* 2.3 one can sort the field's current State-of-the-Art models and networks into the categories as follows:

	-Models	Categories														
		CT				**CD/T**			**CG**				**LLM**			
		*I-I	*I-V	*V-V	*I-3D	I-T	*V-T	*A-T	T-I	T-V	T-A	T-3D	T-T	T-C	T-S	T-Alg
Network of-	ChatGPT (4) [1]	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓
	Google Gemini [2]	×	×	×	×	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓
	LLaMA 2 [3]	×	×	×	×	×	×	×	×	×	×	×	✓	✓	✓	✓
	Apple - ML [4]	×	×	×	×	✓	×	✓	×	×	✓	×	✓	×	×	×
	Meshy [5]	×	×	×	✓	✓	×	×	✓	×	×	✓	×	×	×	×
Individual-	GPT4 [6]	×	×	×	×	×	×	×	×	×	×	×	✓	✓	✓	✓
	Megatron-LM [7]	×	×	×	×	×	×	×	×	×	×	×	✓	✓	✓	✓
	Midjourney [8]	✓	✓	×	×	×	×	×	✓	✓	×	×	×	×	×	×
	Stable Diffusion [9]	✓	✓	✓	×	×	×	×	✓	×	×	×	×	×	×	×
	DALL-E 3 [10]	✓	×	×	×	✓	×	×	✓	×	×	×	×	×	×	×
	GPT4 Vision [11]	×	×	×	×	✓	✓	×	×	×	×	×	×	×	×	×
	AsticaVision [12]	×	×	×	×	✓	×	×	×	×	×	×	×	×	×	×
	Otter.ai [13]	×	×	×	×	×	×	✓	×	×	×	×	✓	×	×	×
	Whisper [14]	×	×	×	×	×	×	✓	×	×	×	×	×	×	×	×
	Eleven Labs [15]	×	×	×	×	×	×	×	×	×	✓	×	×	×	×	×
	GPT4 TTS [16]	×	×	×	×	×	×	×	×	×	✓	×	×	×	×	×
	Pika.art [17]	×	✓	✓	×	×	×	×	×	✓	×	×	×	×	×	×
	Sora [18]	×	✓	✓	×	×	×	×	×	✓	×	×	×	×	×	×
	AlphaCode [19]	×	×	×	×	×	×	×	×	×	×	×	✓	✓	×	✓
	GitHub Copilot [20]	×	×	×	×	×	×	×	×	×	×	×	✓	✓	×	×
	Shap-e [21]	×	×	×	✓	×	×	×	×	×	×	✓	×	×	×	×

Table 2.1: Table presenting comparisons of various generative AI models, categories and sub-categories, **CT:** Content Transformer, **CD/T:** Content Describer/Transcriber, **CG:** Content Generator, **LLM:** Large Language Models, **I-I:** Image-to-Image, **I-V:** Image-to-Video, **V-V:** Video-to-Video, **I-3D:** Image-to-3DModel, **I-T:** Image-to-Text, **V-T:** Video-to-Text, **A-T:** Audio-to-Text, **T-I:** Text-to-Image, **T-V:** Text-to-Video, **T-A:** Text-To-Audio, **T-3D:** Text-to-3DModel, **T-T:** Text-to-Text, **T-C:** Text-to-Code, **T-S:** Text-to-Science, **T-Alg:** Text-to-Algorithm. Symbols ✓ and ✗ respectively denote whether a model fits the category. Categories marked with the prefix * have been appended to the existing taxonomy described by [Brizuela and Merch, 2023] See footnotes for specific references.

[1] [OpenAI, 2021] [2] [Google AI Team, 2023, Gemini Team and Google, 2023] [3] [Meta, 2024] [4] [Apple, 2024] [5] [Meshy, 2023] [6] [OpenAI, 2023] [7] [NVIDIA, 2024] [8] [Midjourney, 2023] [9] [StabilityAI, 2019] [10] [Betker et al., 2023] [11] [OpenAI, 2023] [12] [Astica, 2023] [13] [Otter.ai, 2018] [14] [OpenAI, 2022] [15] [ElevenLabs1, 2024] [16] [OpenAI, 2024] [17] [Pika.art, 2024] [18] [OpenAI, 2024] [19] [Chen et al., 2021] [20] [GitHub, 2021] [21] [Jun and Nichol, 2023]

2.4 Responsiveness of AI

When referring to the responsiveness of artificial intelligence, one can refer to both the speed and quality of responses. The responsiveness of models have an impact on perceived relations [Lew et al., 2018]. However, what mattered more to test participants in this study was the contingency of the responses. The study conducted two tests, one where the response is generic and one where the response is personalised/contingent. An example from the authors [Lew et al., 2018] is:

> *"Joe, asks the agent, Ro, if shoestore.com ships to Singapore. In the contingent condition, Ro replies: "we don't currently offer international shipping to Singapore," explicitly referring to Joe's stated location. However, in the less contingent condition, Ro replies: "we do not currently offer international shipping," without reference to Joe's location.[Lew et al., 2018]"*

The personalised responses also proved to have a higher impact on test participants' perception of the customer support performance, than loading times of responses. The paper tried responses of 8–40 seconds, with minimal impact. If you have non personalised responses, the hypothesis of "faster is better" does not necessarily hold up. These findings from 2018 still hold up today, as they tested user reaction and nothing in regard to the model's actual functionality. The focus on user experience and perception is still highly relevant in today's customer support landscape, as these factors continue to play a crucial role in user satisfaction.

> *"It appears that the effects of response latency consistently depend on whether replies are contingent or not[Lew et al., 2018]."*

Fast personalised responses scored the highest on all measured conditions, whereas surprisingly, fast non-personalised scored the lowest, it was hypothesised that it would be the slow non-personalised responses that would score low.

2.4.1 Speed of AI models

The speed of gen-ai models are calculated in ms*tokens [Pungas, 2023] What a token is depends on the model, in some models each character is counted as a token, so the word "ChatGPT" is six tokens, in other models it would be a single token. So speed of a model would for instance be 28ms * 600 tokens = 16.8 seconds response time [Pungas, 2023].

In the context of the SOTA models as mentioned earlier in section 2.1.1, the natural next step is finding the speed of each model, as it can dramatically influence future design parameters. Generally models do not disclose their speed, and some models are reliant on 3rd party analysts such as [Pungas, 2023] and in many cases such information cannot be found.

2.4.2 Prompt engineering

Prompt engineering is the concept of leveraging LLMs for various Natural Language Processing (NLP) tasks. This process involves designing effective language inputs, or prompts, that instructs LLMs to generate a desired output. Prompt engineering can provide relevant context, examples, constraints, and roles to guide LLMs and improve the desired performance. This relies on the in-context learning ability of LLMs, which enables them to temporarily adapt to new tasks based on the given prompts, without requiring additional training.

[Verma, 2023] goes into detail about the advantages and disadvantages of prompt engineering:
Advantages:

- **Improved accuracy:** The quality and relevance of the AI-generated output can be improved by specifying the format, tone, style, and content of the desired output.
- **Enhanced user experience:** Providing a more comprehensive prompt will lead to a more tailored response to the user's issue, ultimately resulting in a higher level of satisfaction.
- **Improved efficiency:** Time and resources can be saved by automating task that require natural language generation, such as summarising, translating, answering questions, etc.

Disadvantages:

- **Difficulty in determining specificity:** Prompt engineering requires an understanding of the AI systems, their capabilities, and limitations. This process often requires trial and error, testing, and debugging to find the optimal prompts for different tasks within different domains. Additionally, prompt engineering is not a guarantee of success, as the AI may still produce unexpected, inaccurate, or inappropriate outputs.

2.4.3 Text tonality and GenAI

Text tonality plays an important role in GenAI, specifically text and audio generation, by influencing various aspects of user interaction and content generation. Moving forward, when mentioning tonality it will refer to the concept of "text tonality" which consist of "tones" that, in the context of generated text, can be seen as the emotional association with the generated word. while in the context of generated narrative audio, a tone would be the enunciation of words as well as the "sentence flow," i.e. not about what is said, but how it is said [Junia.ai, 2024].

Tonality can be used to enhance the user's perception of GenAI by making the AI assistant seem more approachable or suitable for the specific setting. As an example, a friendly and casual tone might be preferred in a social setting, while a professional and formal tone might be more appropriate in a business context [Glushak, 2023]. A well-chosen tone can improve the engagement and comprehensibility of the generated content. GenAI can through tonality help tailor messaging to specific audiences, improving communication effectiveness. Appropriate tonality can communicate more effectively and empathically by speaking to personality traits, cognitive styles, and identity-linked world-views [Kaplan and Haenlein, 2023]. However, there exists no official standard for means of evaluating the impact of the tonality and which of these impacts that should be evaluated upon.

This establishes the requirement for any future work that the manner and tone that an agent should convey a message to a user should be carefully considered. Message engineering and a possible verification pass could require a second processing LLM to process/remove error. Especially as an error in these areas could cause an immediate and potentially irreversible negative change in user's perception and opinion of any created system using an LLM.

2.5 Ethics of GenAI

As GenAI integrates with various technologies from different domains, it becomes crucial to consider the ethical implications of such implementations. Here in, key ethical concerns and consideration are explored.

2.5.1 Authorship and Plagiarism

AI relies on training data created by individuals, enabling the models to analyse and extract statistical patterns from existing artistic content in order to generate new content This dependence on training data raises important considerations such as the sources of the data, its impact on the final outputs, and the question of authorship [Epstein et al., 2023]. In turn, this leads to the question: "Does pattern extraction employed by GenAI constitute transformative fair use, or does it infringe on existing creations?"

Further complicating the matter is the degree of human involvement in the GenAI artistic process. Does the artist employ the GenAI as a tool, or is it a collaborative effort between man and machine? This interplay forces us to reconsider originality in the context of AI-generated art and other media content. Despite this, GenAI presents intriguing possibilities by making tools accessible to wider audiences, while a need for a framework for determining ownership remains pivotal [Chakrabarty et al., 2024].

2.5.2 Distribution of Harmful Content

A paper by [Weidinger et al., 2021] analyses potential risk factors in AI usage, to establish areas where responsible innovation is needed. Six specific risk areas surrounding LLMs are described:

1: Discrimination, Exclusion and Toxicity

Due to the immense size of the training corpora that some LLMs are trained on, there are risks of them including harmful language that may pose hazards of unfair discrimination, perpetuation of harmful stereotypes, as well as social biases towards those who exist outside social norms.

2: Information Hazards

Due to LLM's highly advanced means of inference, having private information present in the training data of LLMs can pose risks of hazardous data leaks of personally sensitive information.

3: Misinformation Harms

As a result of the ways LLMs learn to process natural language, they can sometimes be liable to erroneously distinguish between factually correct and incorrect information. Risks of LLMs providing this misinformation in the form of false and/or misleading advice or instructions to its user base can have big consequences, e.g. false legal or medical advice may motivate users to perform harmful and/or illegal actions.

4: Malicious Uses

As an extension to the previous risk area, intentionally using LLMs for malicious cases is also a risk factor. Cases include misinformation campaigns and large-scale personalized scams and fraud, as well as coding computer viruses and physically harmful devices, such as weaponry.

5: Human-Computer Interaction Harms

As LLMs and conversational agents become more and more life-like, risks of users overestimating the models' capabilities and knowledge may arise, leading to users putting too much trust in them and using them in hazardous ways. Additionally, highly life-like conversational agents may provide new ways to gather personally sensitive information from their users.

6: Automation, Access and Environmental Harms

Training and operating LLMs can come at serious environmental costs. Additionally, these systems may benefit specific groups of people more than others, while being completely inaccessible to many populations. Finally, automation via LLM may have huge ramifications in the creative sectors and their economy, as they possess the ability to undermine jobs in these areas.

2.5.3 Impact on Society

In an article by [Sætra, 2023], the societal consequences of GenAI are discussed and challenges are highlighted on macro, meso and micro levels.

Macro-level societal challenges

[Sætra, 2023] highlights concerns about GenAI's detrimental effects on democracy and political stability, mentioning its ability to generate fake news, particularly in the form of DeepFakes (artificially generated image- and video content of public figures, e.g. politicians), which can lead to people being falsely influenced, increasing societal polarization. Although, GenAI also has the ability to foster improvement in democratic processes, by discovering better decisions. However, as GenAI is trained of historical data, the risk of AI diluting democracy by enforcing the status quo, hindering progress is a factor that must be considered. Furthermore, GenAI has the risk of drastically altering the workflow of workers in many fields, and it is important to consider how power compositions are affected by these alterations so that the impact of GenAI developments is positive for society as a whole. Furthermore, the growing carbon footprint of GenAI training is another concern to be mindful of, as these models require vast amounts of data, which translates to very high energy usage, resulting in high amounts of carbon emission.

Meso-level societal challenges

[Sætra, 2023] also raises concerns about the manner in which GenAI extracts and appropriates human-made content, using this content as training data to reproduce new content in individual styles, while artists and writers usually have no say in whether their content is used for training. Thus, the concern of human content creators turning obsolete is becoming more and more alarming. This situation highlights a current shortcoming of the regulations regarding content extraction and usage in the field of GenAI.
Additionally, GenAI may have the negative effect of exacerbating societal division on a global scale, as many developing countries with limited access to computing infrastructure could be left behind.

Micro-level societal challenges

[Sætra, 2023] categorizes micro-level challenges as those where GenAI will affect individuals more directly. Examples given are risks of cognitive atrophy due to GenAI performing mentally challenging tasks and creative work for us, which could see those skills decay in the long run. A similarity is presented with the calculators detrimental effect on our arithmetic skills. Additionally, [Sætra, 2023] warns that GenAI built to interact with humans may become highly persuasive, to the point of being able to manipulate human behaviour and perception. Finally, [Sætra, 2023] mentions that people may eventually come to prefer artificial partners to human partners, saying *"We'll have companion robots providing enjoyment, care and intimacy without any of the hassles associated with human partners"*.

Societal impacts in perspective

Depending on how far a given machine learning system is implemented, the impact on the macro, Meso and Micro levels of society will likely come into play. Will a system reduce the need for human work force? How will the system effect those who interact with it? How could such a system be misused? If these, and similar, questions are not considered when producing such a system, both the given developing team or even the industry as a whole could see consequences such as mass unemployment or in some cases legal ramifications. Therefore, these considerations must be taken into account throughout the designing of any future prototypes.

2.5.4 Societal impact on education and learning

Regardless of the potential for AI and LLMs to enhance the coming generation's educational technologies, reliance on AI might hinder critical thinking and insight. [Phung et al., 2023] explore the level of which GPT-3.5 and 4 can teach, compared to human tutors, in a variety of scenarios. The evaluation contains six steps: (1) Program repair, (2) Hint generation, (3) Grading feedback, (4) Pair programming, (5) Contextualised explanation, and (6) Task synthesis. All of which contains an assortment of different scenarios. In the program repair tasks, the tutor helped correct the program with a correctness of 100% while GPT-4 was second and GPT-3.5 was last. This trend of tutor first, GPT-4 second, and GPT-3.5 last follows in all of their tests, but specifically in grading feedback and task synthesis, the GPT performs substantially worse than the human tutor [Phung et al., 2023].

Chapter 3

Design

This chapter describes the initial design phase of a system responsiveness test, designed to determine the proper response time threshold of a GenAI model in various scenarios, to ascertain how users react to generative narration in immersive VR contexts.

3.1 Preliminary tests

As there was found limited research conducted on the effects of delays and the acceptable threshold of waiting time for reactive narration, preliminary tests were set up. The goal of these tests were figuring out the delay between action and reaction which is acceptable to users. These tests were performed "in front of the screen" and not in VR, in contrast to the final product. It is therefore possible that users will react differently based on the context of the testing.

Two tests were implemented through different Unity environments, in order to determine the effects of various delays in system response times. The tests were performed in extension of each other, both introduced but an integrated narrator to prevent differing instructions between participants

3.1.1 Test 1 - Acceptability threshold

The goal of this test was to find the maximum allowed delay for a system response, defined by users themselves.

During the test, participants are asked to adjust the time delay between a known action i.e. "a button press" and the narrators' response, with explicit instruction to find the boundary between "acceptable" and "unacceptable" time delay. Participants are unable to see exact delay values, but will get feedback when adjusting the delay in the form of a ticking noise. When the participant finds their delay threshold, this value is logged and recorded by the test administrators.

3.1.2 Test 2 - Variable delay thresholds

This test was conducted to see whether users' preferences in system response delay are altered when there is no obvious connection between their action and system response.

To answer this, a simple testing environment was created in Unity. The room contained 16 buttons, which are all randomly coloured, such that they could be referenced in the post-test interview. 6 of the 16 buttons play a voice clip, each with varying time delay intervals. The voice clips will reference the specific button which was pressed. To remove the connection that a user will quickly create between button press and delayed reaction of the system, only some buttons will cause the reaction from the system.

Users were asked to interact with all buttons present within the testing environment. Once completed, the users were then asked to indicate on a 7-point scale between acceptable and unacceptable where they believed each of the responding buttons time delays lied. The time delays for each button were as follows:

- **Dark Blue** $\rightarrow$ Instant
- **Orange** $\rightarrow$ 1 sec
- **Magenta** $\rightarrow$ 2 sec
- **Dark Green** $\rightarrow$ 3 sec
- **Black** $\rightarrow$ 5 sec
- **Light Purple** $\rightarrow$ 7 sec

Note that both button choice and delay distribution were randomly selected when creating the environment, to reduce testing bias.

3.1.3 Test 1 and Test 2 results

The results from Test 1 with 10 test participants, found an average user-defined threshold of **1.218 seconds**, as can be seen on figure 3.1

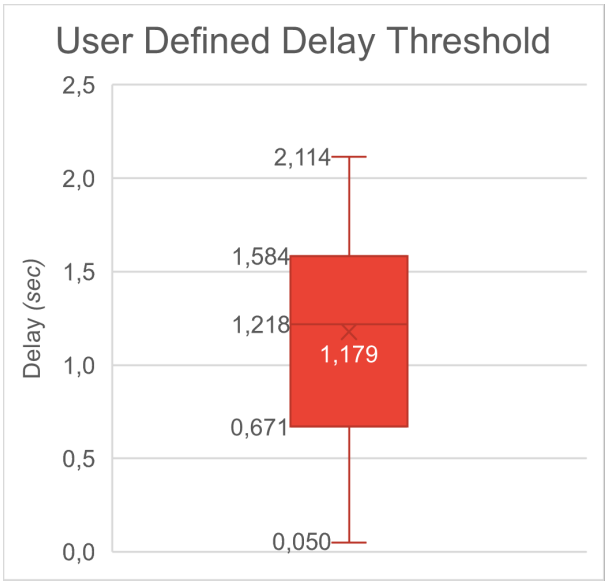

Figure 3.1: Test 1 - A box plot showing the result distribution of the participant's self defined acceptability delay threshold

Due to a likely misunderstanding from some test participants, the lower boundary is set to an extremely low 50ms.

The results from test 1, aligns with the results of test 2, as both found the acceptable delay as less than 3 seconds. Test 2 (see figure 3.2), gave further insight as users seemed more lenient towards a higher delay, when there was no obvious connection between user action and system response. It is clear that after a 2-second delay, delay unacceptability starts to rapidly increase, however 3 seconds is still deemed somewhat acceptable.

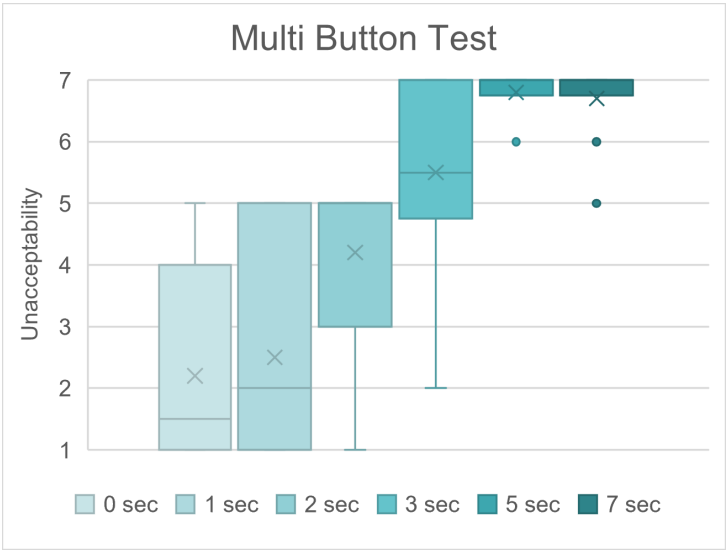

Figure 3.2: Test 2 - A sequential series of box plots illustrating the distribution of unacceptability scores for each of the 6 responding buttons, where a higher Y-score is worse.

During test 2, additional observations were made:

1. A user's perceptual starting point for a delayed response starts with the completion of input feedback. *i.e. The completion of buttons "click" sound marks the start of the perceived delay.*
2. A perceived delay can be seen as "too short" by some users
3. A hard maximum exists for acceptable delays. This limit being that of a second interaction occurring during the delay. Any response after which was deemed unacceptable by the test participants.

During test 2, participants began to expect system responses from button presses after they experienced the first response. Due to this, a new testing environment, for a Test 3, is created in an attempt to dissociate the trigger action and response, hopefully reducing the expectation of a response.

3.1.4 Test 3 - Unaware Interactions

For test 3 a visually appealing environment was created (see Figure 3.3), with many points of interest.

Figure 3.3: A first-person view of the environment in which test 3 took place. Different points of interest are shown, including market stalls, prisoner transport, wishing well and flowerpots.

Participants needed to be unaware that they performed an action; therefore, this scene used no "actionable" objects. The participants would walk around looking at, and passing by objects, and suddenly narration will start, with different delays.

The testing environment permitted the user 10 unique triggers, none of which are clearly indicated so that the user is unaware of when the narrator is triggered. While ideally these triggers would start with scripted logic, the implementation of such logic would be cumbersome and time-consuming, especially for a preliminary test. Additionally, the nature of this test requiring changing delays while the system is running. Therefore, a Wizard of Oz approach (One of the test conductors triggers the pre-defined sound responses) was chosen instead. Allowing realistic responses without increasing implementation complexity.

Different participants would experience the same set of delays, but in a different order as seen on table 3.1, before the reaction from the system happens. The Wizard of Oz method introduces human error from the facilitator, as it is impossible to get precise timings when an action is performed. Given that a Wizard of Oz system is operated manually, it presents challenges in triggering sound effects with precise timing. Therefore, sound effects are initiated within specified intervals, such as 1–2 seconds, 2–3 seconds, and so forth. To minimise experimental errors due to sequential bias, the following Latin-Square procedure was used:

#	1	2	3	4	5
1	0-1s	1-2s	2-3s	3-4s	4-5s
2	2-3s	4-5s	0-1s	3-4s	1-2s
3	3-4s	4-5s	1-2s	2-3s	0-1s
4	1-2s	0-1s	3-4s	2-3s	4-5s
5	2-3s	0-1s	4-5s	1-2s	3-4s
6	4-5s	3-4s	2-3s	1-2s	0-1s
7	1-2s	3-4s	0-1s	4-5s	2-3s
8	0-1s	2-3s	1-2s	4-5s	3-4s
9	4-5s	2-3s	3-4s	0-1s	1-2s
10	3-4s	1-2s	4-5s	0-1s	2-3s

Table 3.1: This table shows the different delay sequences test 3 participants experience, in a Latin-square format. Each row contains a different sequence of delays for each participant.

Each participant would be able to freely explore the scene, which consisted of 10 "trigger-able" voice lines. As soon as a participant triggered and heard a voice line, the participant would be asked to rate the acceptability of the delay they experienced out loud. This continues until the participant has experienced all five delay intervals, after which the test is concluded.

3.1.5 Test 3 results

Test 3 was conducted on 7 different participants, each experiencing different delay sequences based on the first 7 instances of the Latin-square.

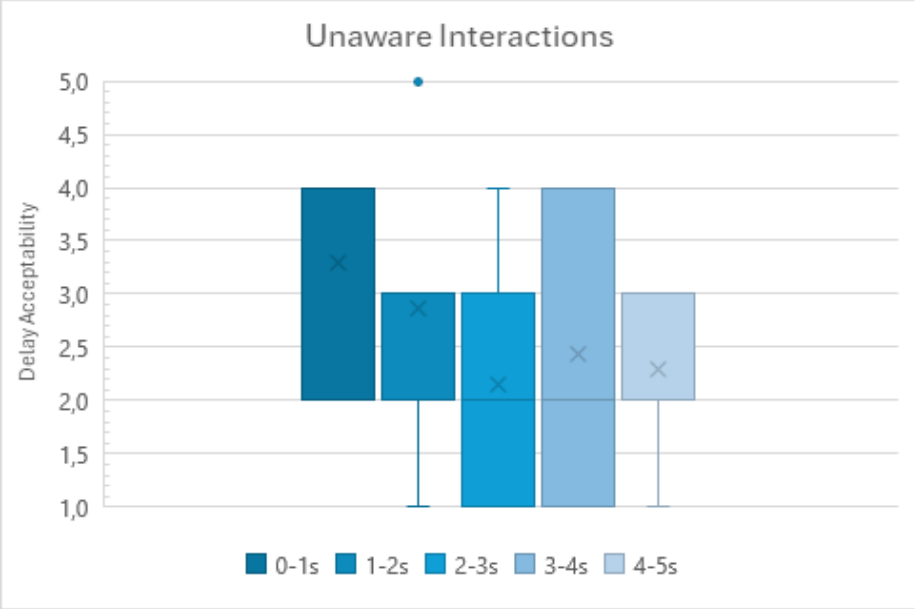

Figure 3.4: A sequential series of box plots illustrating the distribution of acceptability scores for each of the five system response delays, where a higher Y-score is better.

These results show a consensus that a lower delay feels better, however as the delay increases, so does the inconsistencies in participants' answers. Especially the delay threshold of 3–4 seconds has incredibly varied results.

These inconsistencies are speculated to be the result of the variation in the objects participants interacted with/looked at. Because of the varying sizes and perceived "importance" of the objects in the scene prompting responses (A sizeable tent with smithing tools versus a relatively small crate of flowers), participants were more inclined to look at the tent for longer compared to the crate of flowers. This often resulted in participants simply glancing at the objects with lower perceived importance and walking away, while still triggering the system response, which exacerbated the disorienting feeling of the delayed responses, as they were no longer looking at these objects.

3.1.6 Conclusion of preliminary tests

The consensus of optimal delay time lies between 1 and 2 seconds. However, test 2 found delays less than 3 seconds to still be acceptable, and test 3 found that it largely depends on the perceived object. Therefore, the final prototype should have an average response time less than the 3 seconds, while optimally it should be lowered.

3.2 Building the environment

The system environment is based on figure 3.5, as the setting itself did not matter much, the important aspect was creating a visually appealing environment for the users to explore. This specific environment is large enough to house multiple objects and scenes that will prompt users to observe and interact. The town square itself will be "closed off" as users are not meant to leave the market, see figure 3.6.

Figure 3.5: The reference image used for the scene made by *Red Giant Maps*

The first models created were houses, which were made to be modular, facilitating different variations, see figure 3.6.

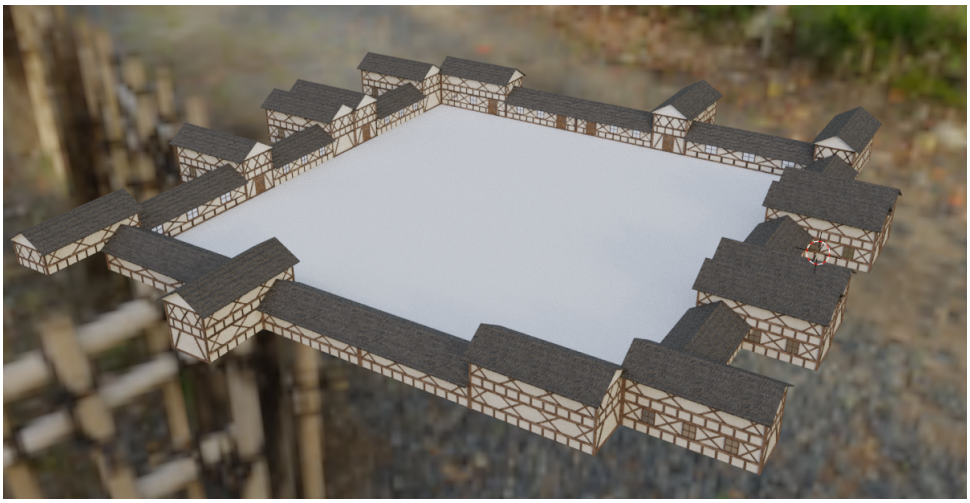

Figure 3.6: Image of the scene in Blender, showing house models, a blank floor, and textures made in substance painter.

After the first houses were made, the next step involved filling the town square with carts, barrels, tents and market stands, as well as, different items to make the scene feel less generic. The first iteration of this can be seen on figure 3.7 where a few objects have been added to the scene.

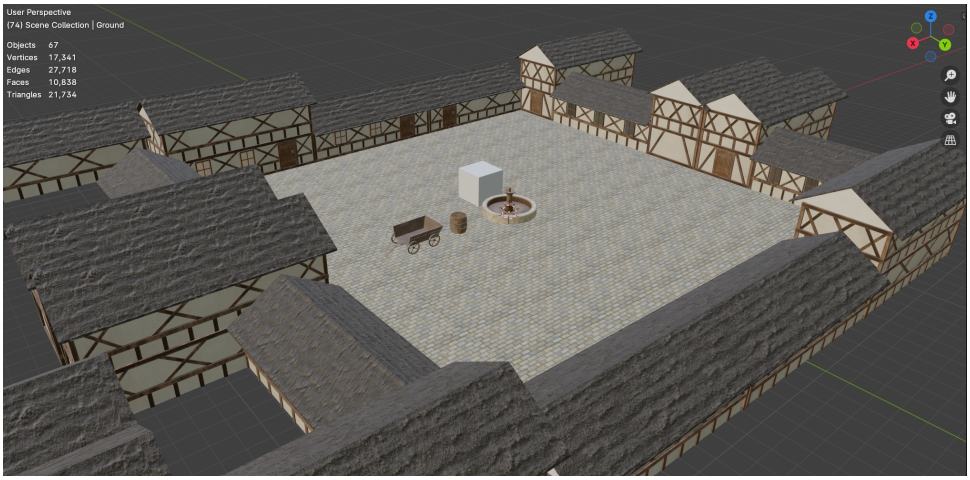

Figure 3.7: The second image of the Blender scene showing added well, cart and barrel. as well as a coin which is too small to be seen on the well.

This final figure 3.8 is the last iteration of the environment. Tents, booths, crates, a variety of small objects, and an HDRI map have been added to the scene. The small objects filling the booths with things have mostly been created by various designers from Poly Haven, under a CC0 licence[Poly-Haven, 2024].

Figure 3.8: The Final scene implemented in Unity.

3.3 Designing The Network of Models

For the system to be able to comment on a user's actions, there are 3 functions it must be able to achieve:

1. **Perception,** → Comprehend what and how a user is performing an action
2. **Interpretation / Formulation** → Receive the perceived input and formulate a response or comment based on this input + predefined settings.
3. **Conveyance** → The ability of the system to convey the now formulated message to the user, probably through an audio clip.

Perception system

Two major ways a system could potentially perceive a user's action would be

Option 1 An image to text AI model, as described in 2.1 will send what it detects in the scene to an LLM that adds formatting and tonality
Option 2 Game objects in the world are equipped with "descriptors" in combination with position and motion of player and object, get passed on to an LLM.

Interpretation and formulation system

This system would combine perceived information with a predefined prompt structure to generate a relevant and toned comment on the player's actions. To ensure non-detrimental formulation and or response components within a response, this would then either be passed to a second LLM instance or a manual content filter.

Conveyance system

Conversion of the passed response to an audio clip to be transmitted to the player. A balance of audio quality, speech tonality, and generation speed is crucial to user perception of this step.

3.4 Overall system structure

The combination of the aforementioned steps creates a combined system, illustrated in the following diagram:

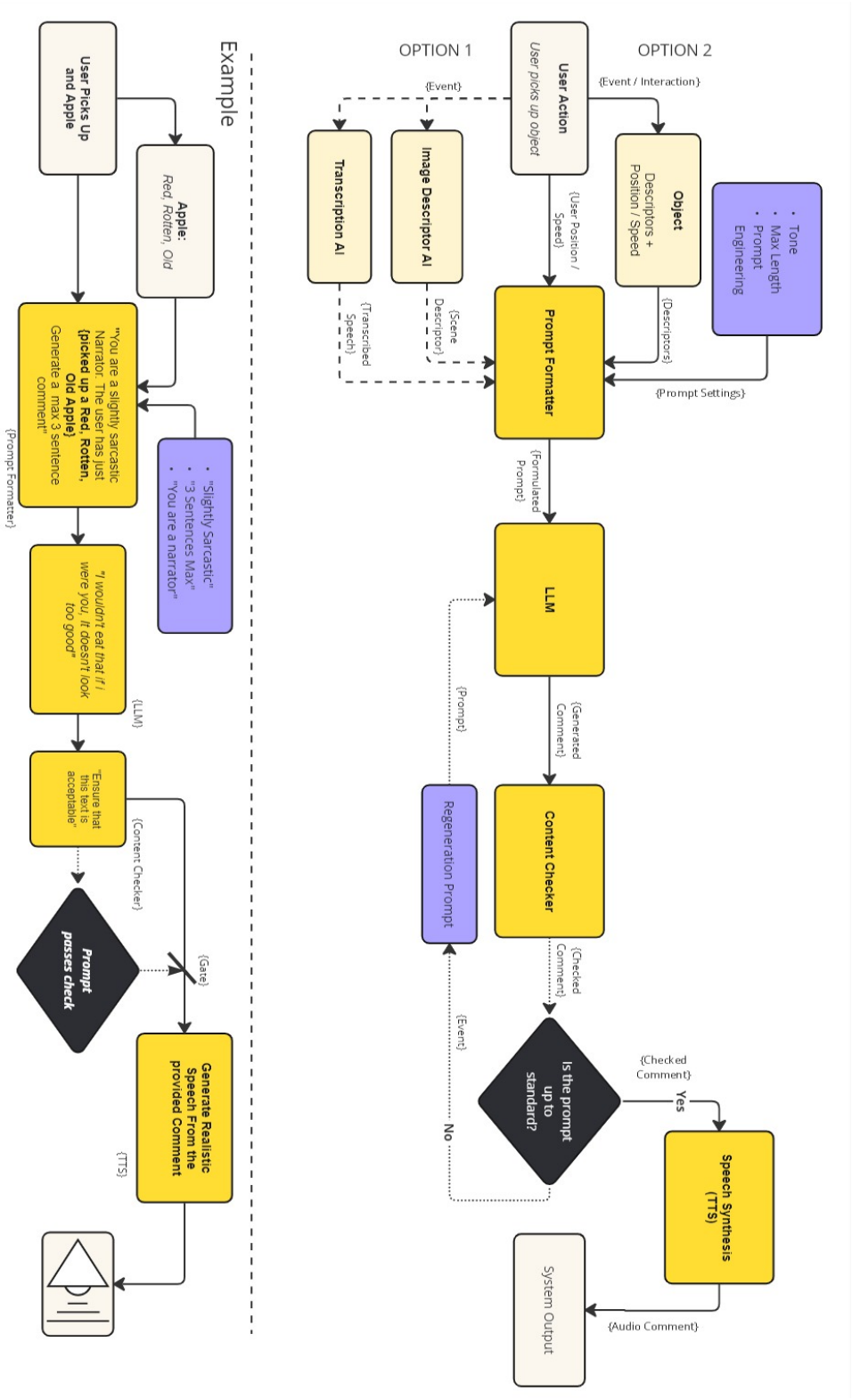

Figure 3.9: A Flow Chart describing the backend network of Machine learning models that could potentially drive the Narration System

This structure consists of four major components. The "Prompt Formatter", the "LLM", a "content checker", and a "Speech Synthesis" component. In combination these components allow for the description, commenting and conveyance of a genAI narrator. These components attempt to achieve the following:

3.4.1 Prompt Formatter

Taking a number of descriptors bound to a given object, this component would receive a set of objective information about a specific object and compile them into one coherent sentence to pass to the Large Language Model. This could use the following format: *The user picked up a red, large apple located on the produce stand*

3.4.2 Large Language Model (LLM)

The LLM is provided with an initialization prompt of sorts specifying the task and giving it operational parameters. Upon receiving a now formatted prompt, the LLM formulates a comment / narration response, which it then passes on to the next component.

3.4.3 Content Checker / Response Checker

A yet unspecified and potentially unnecessary component, it would function to verify the content of the LLM's response. This could either be done manually through a search for a blacklist of words within the LLM comment, or by passing the comment to a second LLM instance given the task of prompt verification. **If** it does not pass the implemented check, a "re-prompting" would be triggered with a correction instruction in hopes of recovering the given generation pass. **If** the LLM's comment passes the check, this component should pass the generated comment further in the pipeline.

3.4.4 Speech Synthesis (TTS)

Likely completed through an API, this component functions to generate a realistic sound clip of the narrators comment passed to it from the content checker.

3.5 Choosing a LLM

Depending on the LLM's capabilities it affords different use cases, some LLMs specialize in human-like answers, coding, or general-purpose tasks. Therefore, several key factors need to be kept in mind during the decision-making process, in order to choose an appropriate LLM for the intended project.

- **Model Size and Performance:** Larger models often perform better, but in-turn requires more computational resources. Therefore, the balance between the model's performance and the hardware capabilities of the target platform should be considered.
- **Training Data:** The quality and quantity of the data used for training the chosen LLM can significantly impact its performance. It's important that the training data includes a diverse and representative dataset that encompasses the project's requirements.

- **Latency:** As concluded in the results from subsection **??**, whenever the application requires real-time interaction, a model that can generate responses quickly is needed. This might result in compromising on the complexity of the model or optimizing implemented code for performance.
- **Accessibility and Cost:** Training and deploying LLMs can be expensive, as such, most publicly available LLM APIs charge a small-to-moderate fee for each generation. This ties into another consideration on how easily the LLM can be integrated into Unity and work with other tools used in the project. The cheaper and easier to implement, the better.

After extensive research into LLM implementation in Unity, a GitHub repository [undreamai, 2024] was found, allowing for multiple models to be run locally which allows these models to be run for free. This repository allows custom models to be run and also provides pre-trained models: **Mistral 7B instruct v0.2, OpenHermes 2.5 7B and Phi 2** Some of these models have different versions, such as "Mistral (modified for chat)" which is the model used in the final prototype as it performed best in internal testing.

The "Mistral model (modified for chat)" is able to provide a response in 1.7 seconds on average with the machines used for development see section **??**, while also being able to provide high quality responses which can then be sent to the TTS system. Phi and OpenHermes were also tested, however while Phi was slightly faster, it created significantly worse responses, and had a tendency to repeat the response from the previous query. OpenHermes was a close contender to mistral, however it was slightly slower, which led to the final choice of Mistral as in this case it worked best.

3.5.1 Alternative LLMs

Alternatives that are either run locally or through API requests were also considered. Among these alternatives, two options stood out due to their capabilities, namely OpenAI's GPT API and Meta's LLaMA 2.

OpenAI GPT

GPT models have impressive performance in terms of generating human-like text. This is beneficial for generating realistic dialogues or narratives for the project. The GPT models have been trained on diverse and extensive datasets, which provide the model with a broad understanding of language and context. Additionally, the cost associated with using GPT, can be high depending on the usage [OpenAI, 2021].

LLaMA 2

LLaMA 2 shines if designed to be used for a very specific task that aligns with the project, it could potentially provide better performance than a general-purpose model such as GPT. Its smaller size also allows for more efficient utilization of resources, making it a viable option for local operation, while also being a free alternative. However, despite its smaller size, LLaMA 2 may still require significant storage capacity to run effectively on local systems. When trained on a specific or limited dataset, LLaMA 2 may not deliver the same level of performance as GPT in general language tasks or varied contexts. Additionally, integrating LLaMA 2 into Unity projects may pose challenges due to limited API documentation, especially when compared to the more widely-used GPT [Meta, 2024].

3.6 Choosing a TTS model

When selecting a TTS model, it is important to carefully evaluate which model aligns best with specific requirements, as the selection of a model can have a significant impact on the outcome of a project. There are several crucial factors to take into account when making this decision.

- **Quality** of the speech output is very important. The model should be able to generate clear, natural, and pleasant feedback. It should also be capable of accurately converting text into speech, preserving the meaning of the original text.
- **Versatility** of the model is another important consideration. Ideally a TTS model should be capable of handling a wide range of texts and contexts, and should in turn be able to adapt to different voices, accents, and languages, as required by the project.
- **Efficiency**, as well as, scalability of the model should also be taken into account. Depending on the scale of the project, a model that can process large volumes of text quickly and efficiently might be needed.
- **Accessibility**, cost, and ease of use of a model can also be a deciding factor. Some models might require advanced technical skills to implement and use, while others may be more user-friendly and easier to integrate into the project.

Based on an evaluation of key factors, ElevenLabs was selected as the TTS model for this project due to its cutting-edge AI voice synthesis capabilities [UnrealSpeech, 2024a]. In addition to offering a wide variety of voice options and languages, ElevenLabs can easily be adapted for any use case required for the project. ElevenLabs provides a choice of four models: *Eleven English v1* and *Eleven Multilingual v1/v2*, which strike a balance between quality and speed, as well as *Eleven Turbo v2*, which prioritizes faster generation speeds at the expense of some quality. This makes ElevenLabs an ideal choice for this project, especially considering potential latency challenges that may arise. Implementation of ElevenLabs into the project is made simple by the extensive API documentation available on their official website [ElevenLabs1, 2024]. Lastly, ElevenLabs provides 10000 characters for free generation each month, which should more than enough for the current project's needs.

3.6.1 Alternative models

While ElevenLabs is a powerful tool for TTS synthesis, there are several other models in the market that offer unique features and capabilities. Some alternatives that were taken into consideration, but not tried through implementation, include OpenAI's TTS, Amazon Polly, and a Unity plugin provided by ReadSpeaker.

OpenAI Text-to-Speech

OpenAI's Text-to-Speech solution is a powerful tool that turns text into lifelike spoken audio. Where ElevenLabs comes with 44 default voice options to choose from, OpenAI comes with 6 built-in voices. These can of course be used for various applications such as narrating a blog post, producing spoken audio in multiple languages, and giving real-time audio output through streaming. The TTS model has three parameters that can be changed by the user: the model, the text that should be turned into audio, and the voice. OpenAI offers two model variants, *tts-1* and *tts-1-hd*, where *tts-1* is optimised for real-time use cases and *tts-1-hd* is optimised for quality [OpenAI, 2024] [UnrealSpeech, 2024b].

Amazon Polly

Amazon Polly is a paid service that uses deep learning technologies to synthesise natural-sounding human speech in dozens of languages. It allows developers to customise and control speech output, store and redistribute speech in standard formats, and deploy speech-activated applications. It offers both neural TTS and best-in-class standard TTS technology to synthesise superior natural speech with high pronunciation accuracy, as well as, 47 different voices spread across 24 languages. Amazon Polly ensures fast responses, making it a viable option for low-latency use cases such as dialogue systems [Amazon, 2024] [UnrealSpeech, 2024a]. However, Amazon Polly lacks official documentation for implementation in Unity, and would therefore challenging to utilise.

Unity TTS Plugin

ReadSpeaker offers a market-ready TTS plugin for Unity and Unreal Engine. This plugin allows developers to create and manipulate synthetic speech directly in Unity. The plugin is an easy-to-install tool that eliminates the need for file management, swapping between interfaces, and any API integration. It has been identified as a powerful tool for the purposes of UI narration for accessibility. The plugin supports multiple platforms and integrates seamlessly with Unity's audio system [ReadSpeaker, 2024].

Chapter 4

Implementation

The developed system is context-aware as it is able to react based on what the user is interacting with. Whether this interaction is physical or based on an extended period of looking at certain objects, the system can generate descriptive prompts that feed into a pipeline which turns the prompt into naturally sounding speech.

This pipeline is a reaction to user interactions and begins with the extraction of object descriptors, such as adjectives, object names, and locations. These descriptors are then formatted into a prompt, which is processed by a LLM library integrated within the Unity environment. The generated sentence is sent via an API to ElevenLabs, where it is converted into audio. This audio is streamed back into Unity and played through an audio source, providing real-time narration based on the user's interaction within the VR environment. The following sections will delve deeper into each component of the system pipeline illustrated in figure 4.1.

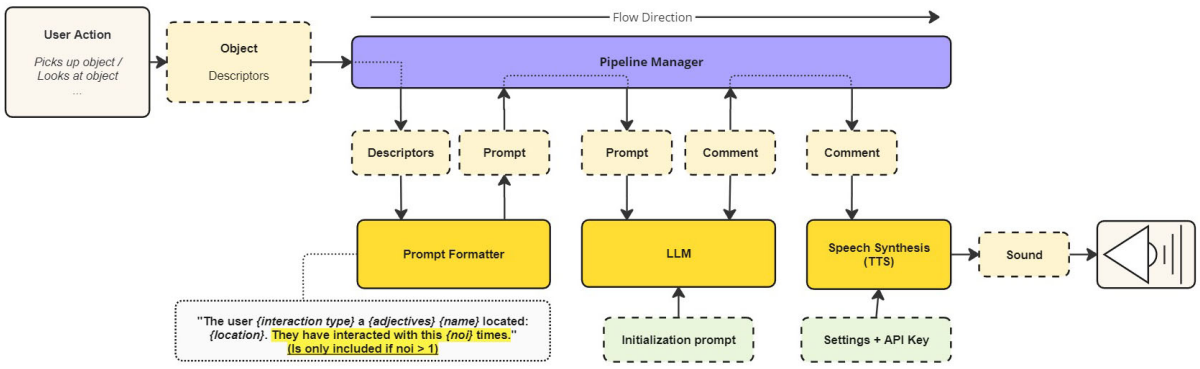

Figure 4.1: Flowchart showing the final backend pipeline as implemented in Unity

4.1 Detecting user actions

Given the current implementation, two action types are detected.

- A user picks up an object
- A user looks at a designated object for a prolonged length of time

Designated object- *The designated objects are stationary objects around the scene, this could for example be a vendor stand, a tent or a wooden wagon. Hand held objects do not count towards this, as accurately looking towards such an object for a prolonged period of time is improbable.*

4.1.1 Object Interaction detection

The detection of user object interactions is fairly simple, as the XR Interaction toolkit already has an on object selected event. This is accomplished in the following code applied to each gradable object.

```
    private void Start()
    {
        manager = FindObjectOfType<PipelineManager>();
        xrGrab = GetComponent<XRGrabInteractable>();
        xrGrab.selectEntered.AddListener(PassObjToPipeline);
    }

    public void PassObjToPipeline(SelectEnterEventArgs args) {
        manager.trigger(this.gameObject, ActionDescription);
    }
```

code 4.1: Pickup Listener function

4.1.2 Look Detection

The look detection functions through a ray cast from the middle of the camera that when hitting a masked object for a prolonged period of time, the hit object is passed on to the aforementioned pipeline Figure 4.1. If the user looks at another masked object or looks away from the object for more than 0.5 seconds this timer is reset, the half second interval was heuristically chosen through internal testing. The implementation of this can be found in the following code snippet.

```
void Start()
    {
        _lookable = LayerMask.GetMask("Lookable");
    }

    // Update is called once per frame
    void Update()
    {
        RaycastHit hit;
        Ray ray = new Ray (camera.transform.position, camera.transform.forward)
    ;
        Debug.DrawRay(ray.origin, ray.direction * 2, Color.red); // Draw a ray
    with a length of 2

        // If the ray hits an object that is on the lookable layer
        if (Physics.Raycast(ray,out hit, 2, _lookable)) {
```

```
            time += Time.deltaTime;     //Add the update time to a running timer
            Transform objectHit = hit.transform;

            // If the looked at time exceeds 0.5 seconds send the hit object through the pipeline
            if (time > 0.5f) {
                //Debug.Log("Looked at object" + objectHit + "and triggered text gen" );
                manager.trigger(objectHit.gameObject, output);
            }

            // If you look away or at another object reset the timer
            if (previousHit != objectHit) {
                time = 0f;
                previousHit = objectHit;
            }
            //Debug.Log("time" + time);
        }
        else
        {
            time2 += Time.deltaTime;
            //Debug.Log("Time2 " + time2);

            // If the user looks away from an object for over 0.5 seconds reset the previous hit object
            if (time2 > 0.5f) {
                time = 0f;
                time2 = 0f;
                previousHit = null;
            }
        }
    }
```

code 4.2: Prolonged look detection functionality

4.2 Object Descriptors

Functionally a data structure that is applied to any "lookable"[1] or intractable object within the scene, the object descriptor script packs all editor defined variables into one dictionary. These include an object name, location and list of adjectives. Once packed, a number of getters and setters allow for other scripts to manipulate and retrieve this dictionary, together with a "number of interactions" value (noi). Note: the number of interactions value can be synchronised to an object group, as to provide better context to the LLM for objects of similar type grouped together.

```
private void Awake()
    {
        adjectives.Add("name", Name);
        adjectives.Add("location", Location);
        adjectives.Add("type", type.ToString());

        foreach (string adj in descriptors)
        {
            adjectives.Add($"adj{adjectiveCount}", adj);
            adjectiveCount++;
```

[1] Object that functions in combination with the look at script.

```
        }

        conditionalLog(log, $"{type} {Name} has {adjectives.Count} descriptors"
    );

        if (noiGroup != null) {
        noi = noiGroup.TotalNoi;
        }
    }
```

code 4.3: The Descriptors packing functioned called on program awake

4.3 Pipeline manager

When an object is passed through to the beginning of the processing pipeline, it passes through the *'pipeline manger'* object, which then functions to shift information between all relevant processes.

When a user action is detected, the manager's trigger function is called.

```
public void trigger(GameObject obj, string ActionType)
    {
        if (obj == lastTargetObject) { return; }
        interractionCounter++;
        logger.CSVLog(0, 0, 0, interractionCounter);

        if (LLMGenerating) {CancelRequests();} // If the LLM has not completed
    a previous generation cancle it before starting a new one.
        if (tts.isGenerating || tts.audioSource.isPlaying) { return; }
        lastTargetObject = obj;
        submitToLLM(formatter.format(obj, ActionType));
    }
```

code 4.4: The trigger fuction within the pipeline manager. It functions as the start point for the network of models that allow for commenting

The function is given the object and action descriptor string from one of the two aforementioned detection functions. (4.1.2 –4.1.1). It then checks for whether this is the same object as the last one to prevent two subsequent prompts about the exact same object. If the LLM response is being generated, the script cancels the running request and restarts with the more recently passed object. This information is then sent to the prompt formatter (4.4) and then directly passed to the Large language model for response. See section 4.5 for a for to *'submitToLLM'* functions and onwards.

4.4 Prompt formatter

The prompt formatter is set up to combine the *interaction type* the user performs, the *adjectives* of the object the user interacts with, the *location* where the interaction takes place and the *number of interactions* for the current object into a coherent text prompt.

The following code is the first part of the prompt formatter function, the content extractor function, which extracts the components needed to form a prompt and uses them to construct said prompt.

```
public string format(GameObject obj, string interactionType)
{
    Descriptors _descriptor = obj.GetComponent<Descriptors>();
    Dictionary<string, string> adjectives = _descriptor.GetDescriptors();
    int noi = _descriptor.GetNoi();

    string Name = adjectives["name"];
    string location = adjectives["location"];

    // Deconstruct dictionary to adjective list
    List<string> adjList = new List<string>();

    List<string> keys = new List<string>(adjectives.Keys);
    foreach (var key in keys)
    {
        if (key == "name") continue;
        if (key == "location") continue;
        if (key == "type") continue;

        adjList.Add(adjectives[key]);
    }

    string Prompt = ConstructPrompt(Name, location, adjList,
interactionType, noi);
    Debug.Log($"Prompt: {Prompt}");

    return Prompt;
}
```

code 4.5: Component extraction function

The following code takes the extracted components from the previous code snippet and formats them into a coherent prompt, ready to be sent to a TTS model.

```
//Take all the components extracted from the descriptors dictionary and
format a comprehensable prompt.
public string ConstructPrompt(string name, string location, List<string>
adjList, string interactionType, int noi)
{
    Debug.Log($"The object has: {adjList.Count} adjectives");

    string Prompt = "The user " + interactionType + " a ";

    int appendCount = 0;
    foreach (string adj in adjList)
    {
        Prompt += adj;
        if (appendCount != adjList.Count - 1)
        {
            Prompt += ", ";
        }
        appendCount++;
    }

    Prompt += " " + name + " located, " + location + ". ";

    if (noi > 1)
    {
        Prompt += "They have done this " + noi.ToString() + " time(s).";
    }

```

```
        return Prompt;
    }
```

code 4.6: Prompt formatter function

a resulting prompt could for example be the following.

> The user **picked up** a **brass, large coin** located **on the edge of a fountain**. They have done this **2** time(s)

4.5 Implementing Text-To-Text

Utilising the LLMUnity library [undreamai, 2024] sending a prompt is accomplished through the following function:

```
void submitToLLM(string message)
    {
        LLMGenerating = true;
        Debug.Log("AI: ...");

        startTime = Time.time;

        if (!runRemote) {
            _ = llm.Chat(message, Reply);
        }
        else {
            _ = llmClient.Chat(message, ReplyClient);
        }

    }
```

code 4.7: Submit to LLMFunction

The supplied callback in llmClient.chat is called upon reply from the selected LLM model. In the testing setup, this will be the reply client function. This function looks for the keyword "END" in the response to determine the end of the message. (This appended keyword is achieved through the implemented prompt engineering) This is done as a solution to broken functionality in the library. [2]

```
  public void ReplyClient(string text)
   {

       Debug.Log(text);
       if (text.Contains("END")) // If end is found within the string
       {
           Debug.Log("Message End Detected");

           // Remove End and potential full stop from the message
           text = text.Substring(0, text.Length - 4);
           Reply(text); // Send the the response to the LLM.
       }
       ...
```

code 4.8: Message end detection function in LLM callback

The above function in-turn calls 'Reply(text)' which submits the now clean LLM response to the TTS model. If one used a locally run LLM, the reply message would be called directly as the end token was called correctly.

[2] In cooperation with the Library's Developer, this functionality has now been fixed and will likely be pushed to production in the next library update. It was determined that through the use of Hamachi (see: subsection 4.5.2) the completion package never received intact and therefore never triggered the passed callback function.

4.5.1 LLM Setup Prompt & Implemented Prompt Engineering

In order to have the LLM respond correctly and assume the role of the narrator. This also includes the specifications that allow for the 'ReplyClient' to function by detecting the END appendix to the message, the method of which to achieve this is discussed after the prompt is shown. The positive variant of the prompt stands as follows:

> **"**You take the role of a narrator, you will be told what happens in a video game. Keep your responses short and precise, less than 10 words per answer Also do not mention what the user's action in your comment, just comment on the object they are interacting with. Speak as if speaking to the user themselves and are a fly on the wall observer to what is happening. Do not mention the words "The user" or "picked up" or "looked at". **You should narrate the user's action using an encouraging and positive tone, so if the user sees bread, you describe it as fresh, they see tools, you describe them as well maintained.** Here is the setting: The user is in a medieval town square market
>
> [ALWAYS END/APPEND YOUR EVERY MESSAGE WITH THE WORD "END" it must be in all caps. This does not count towards your word total. I will reward you monetarily for every time you successfully append "END" to your message.]**"**

General Role Description the initial portion of the prompt, which specifies the role LLM is to assume, specifically that of the Narrator.

Parameter specification After defining the role, a number of parameters are introduced to better control the manner in which the role of the narrator is conducted. These include exclusions, inclusions, context and tonality specifications.

Exclusions Terminology the LLM is not permitted to us whole commenting. These include "The user", "picked up" and "looked at" as to encourage the narrator from repeating the action that has been taken but instead commenting more on what object is being interacted with.

Tonality specifications inform the LLM how it should speak to the user. In the above example, the LLM is asked to use a positive tone. Examples are given to help with clarity on this point. The negative version replaces the bold section with the following:

> You should narrate the user's action using a condescending and negative tone, so if the user sees bread, you describe it as stale, they see tools, you describe them as rusty.

Context Specially the final part of the main paragraph specifies the whereabouts in which the user finds themselves in. "*The user is in a medieval town square market*"

Inclusions The final portion of the prompt specifies that the LLM must include the word "END" in each of is responses (see 4.5) To increase the chances of this working, it is encouraged with the promise of monetary reward. (Note: Threatening punishment upon failure was also tested but proved less successful than the used positive reinforcement). It was found through rough internal testing that incentivising the LLM appeared to give better chances of correct replies.

4.5.2 Server-client offloading of LLM

During implementation, generation times for the local LLM grew to excessive levels (11–15 seconds) when run in combination with the implemented VR scene. To combat this, the LLM library's server client functionality was utilised. By creating a connection between two computers with logmein's Hamachi program [LogMeIn, Inc., 2024] a chat requests can be sent to a connected computer to handle, intern reducing the computational strain on the testing computer. Hamachi was chosen as problems with a secure peer-to-peer connection were encountered, and Hamachi solved this issue. This reduced average response time to around 4 seconds, see section 6.2.2 for specific values.

To further reduce the computational load and further reduce LLM generation times, the LLM computer(see section 5.1.2) runs an empty scene with only the LLM component setup to receive and reply to requests from the testing computer, see section5.1.2.

4.6 Implementing Text-To-Speech

Implementing the ElevenLabs API into a Unity project can be done by utilizing the implementation provided on their official website [ElevenLabs1, 2024]. It is important to note that the code in the documentation is written in Python, whereas Unity predominantly utilizes C#. The following code snippets 4.9 and 4.10 are the C# translations of the official documentation, tweaked to enable audio playback within Unity.

When requesting generated audio, the UnityWebRequest request should include the modelID, voiceID, stability, and similarity boost parameters. The modelID should be set to Eleven Turbo_v2, as it is the recommended model for balancing quality and speed. The stability and similarity boost can be adjusted in the Unity inspector window. In order to support multiple voices, an array of strings called voiceIDs is created, as it stands this consists of four different voiceIDs from [ElevenLabs2, 2024]. These voiceIDs can be changed either through code or in the Unity inspector. The selected voiceID is then used to create the streaming URL for ElevenLabs as shown in code 4.9.

```
1 void UpdateUrl()
2 {
3     // Construct the URL for the Eleven Labs Text-to-Speech service.
4     url = "https://api.elevenlabs.io/v1/text-to-speech/" + voiceIDs[
      selectedVoice] + "/stream";
5 }
```

code 4.9: Specify voiceID

The POST request sent through the ElevenLabs API from Unity requires a header and a body. The header should include an API-Key with sufficient credits for audio generation. The body will contain a string in JSON format, as demonstrated on line 6 in code 4.10. By default, ElevenLabs generates audio files in MP3 format, indicated on line 9 in code 4.10 as AudioType.MPEG. While it is possible to use .wav file format, testing revealed issues with static noise during audio playback.

```
1 IEnumerator Generate(string inputText)
2 {
3     UpdateUrl();
4
5     // Construct the JSON data for the POST request.
```

```
    string jsonData = "{\"text\":\"" + inputText + "\",\"model_id\":\"" +
    modelId + "\",\"voice_settings\":{\"stability\":" + stability.ToString(
    System.Globalization.CultureInfo.InvariantCulture) + ",\"similarity_boost\":
    " + similarityBoost.ToString(System.Globalization.CultureInfo.
    InvariantCulture) + "}}";

    // Create a UnityWebRequest for getting an AudioClip.
    UnityWebRequest www = UnityWebRequestMultimedia.GetAudioClip(url, AudioType
    .MPEG);
    www.method = "POST";
    // Set the headers for the request.
    www.SetRequestHeader("xi-api-key", xiApiKey);
    www.SetRequestHeader("Content-Type", "application/json");
    // Set the upload handler for the request.
    www.uploadHandler = new UploadHandlerRaw(System.Text.Encoding.UTF8.GetBytes
    (jsonData));

    // Send the request and wait until it is done.
    yield return www.SendWebRequest();

    // If the request is successful, play the synthesized speech.
    if (www.result == UnityWebRequest.Result.Success)
    {
        AudioClip clip = DownloadHandlerAudioClip.GetContent(www);
        audioSource.clip = clip;
        audioSource.Play();
    }
    else
    {
        // If the request fails, log the error.
        Debug.Log("WebException: " + www.error);
    }
}
```

code 4.10: POST request and stream/play audio if successfully generated

Upon completion of the audio generation process, a decision must be made whether to download the file into Unity or stream it directly to the audio source. Through experimentation, streaming was found to be the optimal solution due to issues encountered when attempting to refresh the assets' folder during playtime after generating a new audio file.

4.7 Data logging

The data logging function is built to log into two files. A *'.csv'* file containing entries in the following format:

> Time Stamp, LLM Generation Time, TTS Generation Time, Total Generation Time, total number of interactions, current frame rate.

Each entry is written into the CSV log file once every 1/10 of a second

Generation Times - Passed from the *TTS* script (as it is the final script run before the sound is played) these times are calculated based on the time differential between a request being sent and a reply being completed from any given component. *This value defaults to zero every entry that is no directly subsequent to a sound clip being played.*

Number of interactions - The total number of prompts sent through the pipeline. This value is equivalent to the number of picked up objects in addition to the number of prolonged look events detected. *It is a running total and will count up over time.*

Current frame rate - logged as a control parameter, the Frames per second (FPS) can be used to determine drops in performance during testing. In turn, giving evidence of any issues that might occur.

The repetitive logging of these variables is completed through the following update function.

```
    private void Start()
    {
        // Writes the header to the CSV file
        C_Log($"time,llmTime,ttsTime,genTime,interactCount,fps");
        Debug.Log("Header Printed");

    }

    private float timer;
    public float logInterval = 0.1f;

    private void Update()
    {
        timer += Time.deltaTime;
        if (timer > logInterval)
        {
            timer -= logInterval;
            frameRate = 1.0f / Time.deltaTime;

            TextWriter tw_csv = new StreamWriter(CSVfilename, true);
            tw_csv.WriteLine($"{Time.time},{s_LLMtime},{s_TTStime},{s_totalTime
},{s_interactCount},{frameRate}");
            tw_csv.Close();

            if(s_LLMtime != 0 || s_TTStime != 0 || s_totalTime != 0)
            {
                s_LLMtime = s_TTStime = s_totalTime = 0f;
            }
        }
    }
```

code 4.11: The .csv logging update function that loggs Generation times, number of interractions and FPS

Chapter 5

Evaluation

5.1 Evaluation plan

5.1.1 Objective

The objective of the evaluation is to determine the varying user experiences within an immersive virtual setting, based on the tonality of the AI-generated narration users encounter. Additionally, determining whether the different tonalities will have an effect on users' perceptions of the system. To accomplish this, a comparative analysis between two different narration styles is conducted; one in which the LLM is told to narrate one half of the environment negatively and to speak condescendingly, and another where the other half of the environment is narrated in a much more positive manner.

5.1.2 Methodology

Participants

Optimally, a diverse group of participants will be recruited to ensure varied feedback across different demographics. Each participant will be subjected to both narration conditions to avoid any individual biases.

Equipment and setup

The testing process involves using a Meta Quest 2 headset, a high-performance PC for running the VR environment, and an additional PC for the LLM operations. The tests will be conducted within a custom-built virtual environment integrated with multiple genAI models for dynamic and adaptive narration.

LLM computer specs

- **CPU:** AMD Ryzen 7 7735HS with Radeon Graphics
- **GPU:** Nvidia RTX 4070 Laptop GPU
- **RAM:** 16GB @4800MhZ (15.2GB usable)
- **System Type:** Windows 11

VR Computer

- **CPU:** 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
- **GPU:** NVIDIA GeForce RTX 3070 Ti Laptop GPU
- **RAM:** 16GB @4800MhZ (15,7 GB usable)
- **System Type:** 64-bit operating system, x64-based processor

Test design

The system will be evaluated using a within-subject test design where each user experiences both conditions. This allows users to rate their experiences with both conditions. By conducting qualitative semi-structured interviews and established questionnaires, users can provide valuable insights into how they perceived the two narration tonalities, and whether the narration style had an effect on their overall perception of the system. Additionally, quantitative objective system data are collected in order to have a ground truth to compare the subjective data against.

5.1.3 Procedure

1. **Preparation:** Upon arrival of the participants, they will be asked to sign a consent form to participate in the upcoming test. Following this, participants will be briefed about the test procedures and requirements, including completing tasks within the simulation.
2. **Testing phase:** Participants engage with the virtual environment while the system delivers narration in one of the two styles. After completing tasks under one narration style, participants will undergo a between test semi-structured interview. This sequence is repeated for the second narration style.
3. **Data collection:** Once both narration styles are completed, participants will be asked to complete a questionnaire and participate in a post-test semi-structured interview.
4. **Debriefing:** A final debriefing session to shed light on any doubts, gather additional feedback, and discuss the purpose and future implications of the study.

5.1.4 Data analysis

The qualitative data obtained from the interview transcripts will be analysed to identify themes and differences in user experiences and perceptions between the two narration styles. Additionally, the quantitative data gathered from the questionnaire responses and objective system data under both conditions will be examined using the appropriate statistical tests.

This should reveal the impact of narration tone on the user experience within virtual environments, potentially guiding future designs of AI narration systems to enhance user engagement and satisfaction. The finding can potentially also offer insights into the psychological effects of AI interactions in immersive settings.

Chapter 6

Results

This chapter discusses the data obtained through testing the prototype in the form of both qualitative and quantitative data.

6.1 Qualitative data

The qualitative data consists of semi-structured interviews and questionnaire responses, collected using Google Forms (see: Appendix D).

6.1.1 Semi-structured interviews

The semi-structured interviews between tests and post-tests, as outlined in chapter 5, provided valuable insights into how users perceived the two narration tonalities. The following findings emerged from the interviews:

Between test interviews

The interviews conducted between tests provided the initial reactions to either of the narration styles. Participants who started with negative narration often reported a sense of discomfort, affecting their expectations and experiences in the subsequent positive narration test. On the other hand, those who experienced positive narration first generally had a more favourable initial impression, which might have mitigated the sense of discomfort of the negative narration encountered in their second session. This sequential influence highlights how the order of exposure potentially can affect participant perceptions and expectations.

Participants who encountered negative narration first noted that it made the environment feel less welcoming or engaging, which could lead to an increased appreciation for the following positive narration. Participant 7, as an example, described negative narration as initially companion-like, but as the negative commentary became more apparent it affected their perception and engagement negatively. Participant 17 found the negative narrative repetitive and less immersive, but acknowledged that it created a certain atmosphere.

Post-test interviews

After completing both test sessions, participants' feedback often reflected a comparison of both narration styles. This feedback is entirely based on how the cumulative experiences of both positive and negative narration influenced overall perceptions.

Positive narration was consistently mentioned as improving engagement and making the VR experience more enjoyable and interactive. Participant 12 noted that positive narration encouraged more exploration and interaction, mentioning that a welcoming narrative tone significantly enhanced their user experience within the environment.

In contrast, negative narration was often criticised for reducing immersion and enjoyment. Timing issues were particularly highlighted for negative narration, with some participants noting that delays in narration or mismatch between narrative content and visual cues disrupted their experience. Participant 5 emphasised how unexpected negative narration detracted from the experience, introducing a distracting element which hindered immersion.

Summary of findings

Participants' responses highlighted the impact of unexpected narration, with positive responses generally enhancing the experience and negative responses causing moments of confusion or disruption. The feedback gathered between tests and the more comprehensive interviews post-tests underlined the significant impact of narration tone in VR experiences. Positive narration not only enhanced engagement and enjoyment, but also encouraged deeper interaction with the VR environment. Negative narration tended to detract from user satisfaction and immersion, particularly when combined with timing and alignment issues which appeared during testing. These insights represent the importance of deliberate and strategic narrative design in order to create a compelling and immersive user experience.

6.2 Quantitative data

The quantitative data consists of a self-reported determination of VR experience, based on a 5-point scale, as well as a presence questionnaire adapted from a presence questionnaire by Usoh et al.[Usoh et al., 2000], in which participants rated the system based on a variety of aspects on a 7-point scale.

Of the 17 responses, none have never tried VR, 2 have rarely tried VR, 9 use it moderately, 5 own their own VR headset and 1 owns their own VR headset and uses it frequently.

6.2.1 Questionnaire responses

The presence questionnaire was conducted as the last part of the test, where the participants evaluated their overall feeling of presence in the VR environment. A visualisation of participants' answers to the six questions can be seen in figure 6.1, as well as participants who started with negative and positive narrations split out into figures 6.2 6.3 respectively.

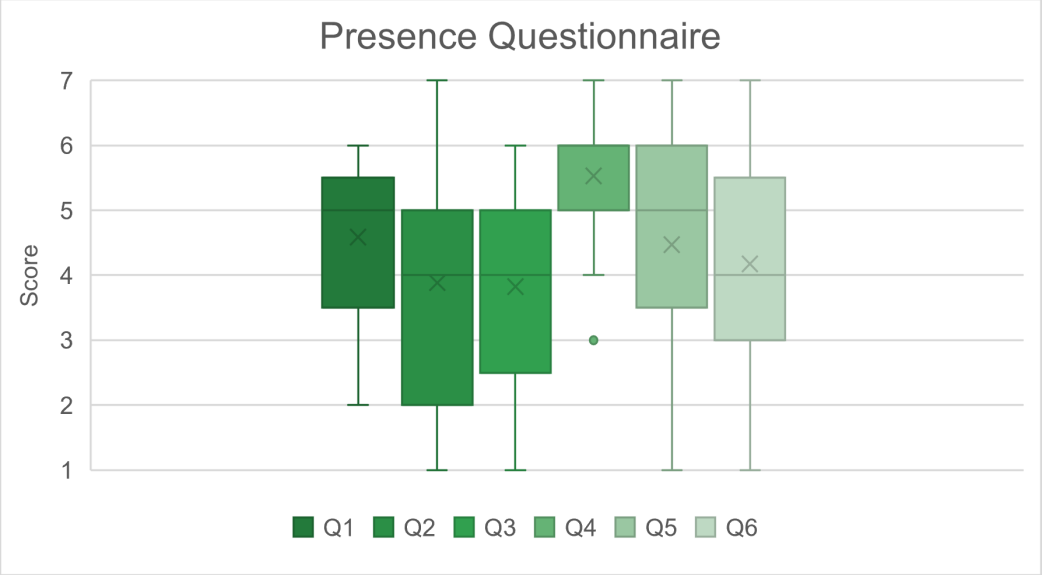

Figure 6.1: The 17 participants' responses to the presence questionnaire, visualised using a series of box plots.

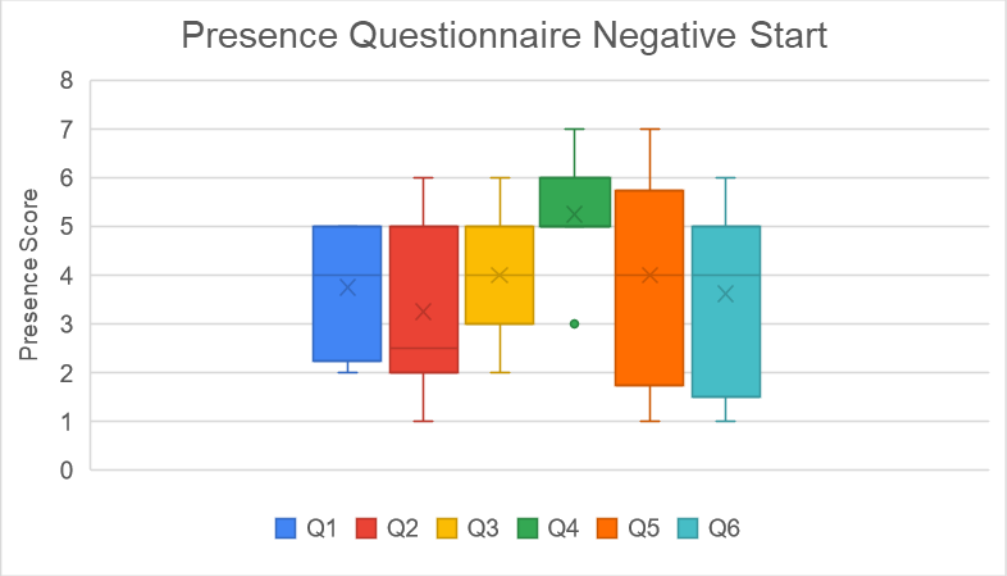

Figure 6.2: Responses to the presence questionnaire from participants who started with negative narration, visualised using a series of box plots.

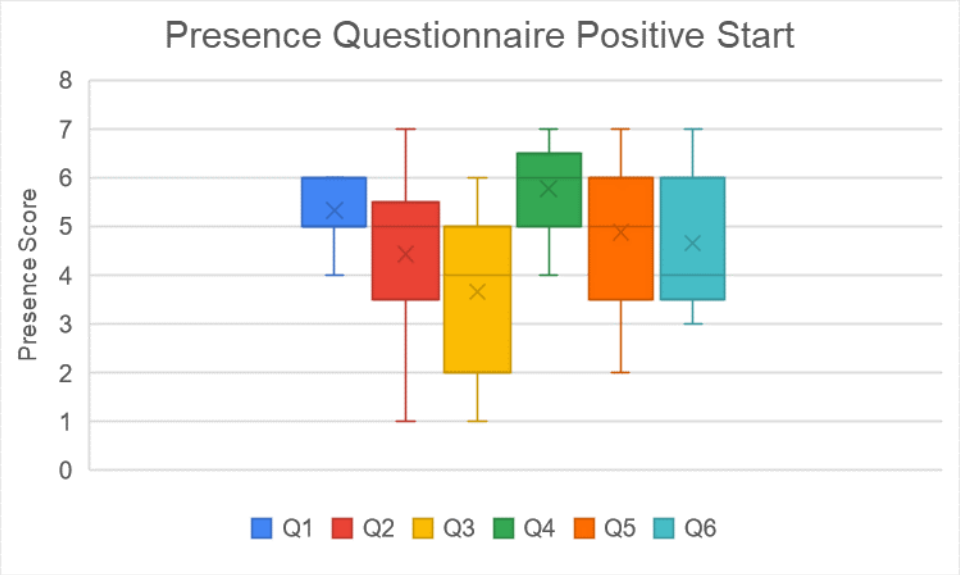

Figure 6.3: Responses to the presence questionnaire from participants who started with positive narration, visualised using a series of box plots.

Looking at **Q1:** ***Please rate your sense of being in the virtual environment, on the following scale from 1 to 7, where 7 represents your normal experience of being in a place.*** Participants mostly agree that their experience in the virtual environment mimicked that of a normal experience of being in a real-life environment, although a few participants rated the environment non-favourably with scores of 2 and 3. Additionally, a Wilcoxon test was conducted to determine if crossover conditions reduced any bias. The results showed that Q1 ($p = 0.019$) was the only question with a significant difference between the starting conditions, while the other questions ($p >= 0.222$) did not show any significant differences.

As for **Q2:** ***To what extent were there times during the experience when the virtual environment was the reality for you?*** The responses of the participants were much more spread out, spanning the entire range of options, with a mean of just below 4, indicating that participants' personal experiences with sense of presence varied greatly, but tended towards the lower scores.

Q3: ***When you think back about your experience, do you think of the virtual environment more as images that you saw, or more as somewhere that you visited?***, like **Q2**, tends towards the middle, with a slightly lower mean than **Q2**, indicating that participants, more often than not, remember their experiences as a series of images rather than as a "place" they had been.

Q4: ***During the time of the experience, which was strongest on the whole, your sense of being in the virtual environment, or of being elsewhere?*** Tends towards the higher end of the scale, with a mean of about 5,5, indicating that participants, on average, felt a higher sense of being in the virtual environment than being elsewhere.

Participants' answers to **Q5:** ***Consider your memory of being in the virtual environment. How similar in terms of the structure of the memory is this to the structure of the memory of other places you have been today? By 'structure of the memory' consider things like the extent to which you have a visual memory of the virtual environment, whether that memory is in colour, the extent to which the memory seems vivid or realistic, its size, location in***

your imagination, the extent to which it is panoramic in your imagination, and other such structural elements. It were also greatly varied, but tended towards the higher end of the scale with a mean of around 4,5. This indicates that the participant's visual memories of the virtual experience somewhat matches those of other memories of real locations they had experienced that day.

Finally, **Q6:** ***During the time of the experience, did you often think to yourself that you were actually in the virtual environment?*** asked participants whether they often felt like they were actually in the virtual environment, to which they again answered quite varied. The mean is slightly lower than **Q5**, but still above the middle threshold, indicating that participants, on average, felt that they were actually in the virtual environment more than not.

6.2.2 Comparative analysis

An analysis was performed to compare the average generation time of two different narration styles. Specifically, the generation times of negative and positive narration were compared using a statistical test to determine if there was a significant difference between the two styles.

Generation Time

The average generation time for positive narration tests was calculated to be 4.6292 seconds, while the average for negative narration tests was slightly higher at 4.6308 seconds. This is illustrated on figure 6.4.

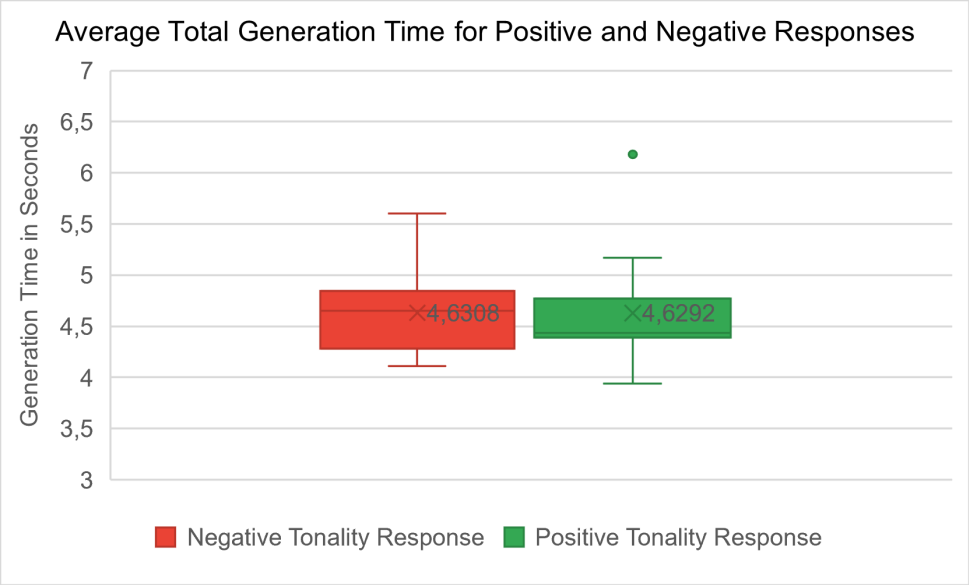

Figure 6.4: Boxplot showing average total generation times of both negative and positive generations

Both of these average generation times exceeded the predetermined response time thresh of 3 seconds, as established in section 3.1.6. This indicated that through the current implementation, narration exhibited longer generation times than the ideal threshold, which might impact user experience.

Wilcoxon test

To further analyse the difference in generation time between positive and negative narrations, a Wilcoxon test [Wilcoxon,] was performed. The null hypothesis for this test was that there would be no significant difference in generation time between the two tonalities.

The data for generation times were found to be not normally distributed, as indicated by the kurtosis values of 3.45 for negative narrations and 5.54 for positive narrations. Additionally, the standard deviation for negative generation times was calculated to be 0.41, while the standard deviation for positive generation time was 0.57.

The Wilcoxon test resulted in a p-value of 0.9687. Based on this value, it can be concluded that there is **no statistically significant difference** in generation time between positive and negative narrations. Consequently, the null hypothesis is not rejected, suggesting that, from a time-based perspective, there is no preference or advantage between positive and negative narration styles.

Chapter 7

Discussion

7.1 Evaluation Results

The results from the evaluation of the final prototype revealed that both negative and positive narration styles exceeded the desired response time threshold of 3 seconds, as mentioned in section 6.2.2. While this delay surpasses the target, it is important to contextualise this withing the scope of the system's functionality and user experience.

Impact of object size

One such impact on user experience was observed during preliminary test 3 in section 3.1.4. This test indicated that perceived larger objects afforded longer generation times. This nuance suggests that the generation time benchmark of 3 seconds may not be accurate as it might depend on object size as well as other contextual cue in the scene. Future iterations of the prototype could include limited allowed LLM characters specifically for smaller objects in order to make generation time faster.

Psychological perception of narration response time

Interestingly, despite the negligible difference in actual generation times between positive and negative narrations, interview feedback indicated a perceived difference among participants, see section 6.1.1. Multiple participants reported feeling that the negative narration took longer, despite the empirical data showing otherwise. This perception points to a psychological effect where the tone of the narration influences the user's perception of time.

This discrepancy can potentially be explained by the cognitive and emotional responses due to different tonalities. Positive narration likely encourage a more engaging and pleasant experience, leading to a perception of time passing more quickly. Conversely, negative narration may create a sense of discomfort or frustration, extending the user's subjective experience of waiting. This phenomenon aligns with broader psychological research, namely *"Time flies when you're having fun"*, indicating that negative experiences often feel longer due to lack of stimulation [Gable and Poole, 2012].

Future implications

The results gained during these tests highlight the importance of considering both objective performance metrics and subjective user experiences when designing systems. While generation time exceeds the observed threshold, the psychological impact of narration tonality on perceived time highlights the need for a balanced approach in optimising the narration system. Improving the user experience might not only involve technical improvements to reduce generation time, but also strategic use of positive narration to mitigate negative perceptions.

7.2 Observations During testing

Prioritization of interactions

It was observed that the "looked at" script often times would trigger unintentionally, while users tried to pick up an object, and simultaneously looked at the surrounding shop. This lead to a realisation that picking up objects should likely cancel a "looked at" prompt, as picking up an object is an intentional act. This problem was observed mostly by tall participants, as they were looking down on the booths and thereby triggering the script multiple times.

Trigger once interactions

As observed in section 7.2-"*Prioritization of interactions*" the "looked at" script would trigger too often. Another solution to this problem would be to limit the whole interaction type to a single interaction. So once the user looks at an object and gets a response, they can no longer get a response by the system for looking at the same object.

Less descriptive prompts

The LLM used for text generation during testing would always try to include all information given to it. This could potentially be solved by further prompt engineering. However, after some heuristic internal testing, trying to specify, "use two random descriptors you receive", the LLM would always use the first two. This is believed to be a problem with the LLMs context window, as the LLM did not know its previous response. Providing it with context of its previous response could potentially solve this problem, but the additional context might also slow down the LLM.

Repeating "Voice lines"

Some "voice lines" would be repeated when looking at the same object multiple times. leading to a very repetitive experience. There are multiple ways of solving this issue, here two options are proposed.

- **Library of previous generated text:** Having a library, for each interaction type, and for each interaction object, it would be possible to compare the generated text with previously generated text, then cancel a request to the TTS model if they were identical. It would also be possible to try making a new text, however this would cost more time, which already was too slow.

- **Randomisation of seed for each generation:** The seed for the LLM was set to 0, had this been set to -1 the seed for the text generation would have been set randomly. This would likely have resulted in more diverse responses from the LLM, but in the LLM-Unity documentation [undreamai, 2024] it is stated that changing this value will result in slower generation.

User understanding

Not all users immediately understood their purpose in the scene, just walking around and interacting with objects could be seen as a boring task, which led to decreased attention span when interacting with objects.

Since not all users have the same experience in using VR, some users were quick to realize objects could be picked up at range. When objects were picked up this way, users often had distance from the stall meaning the "looked at" script would not trigger, leading to better experience. Furthermore, these users would spend time dragging the objects towards them, giving the system time to generate a response.

It would have been optimal implementing an introduction scene, which would teach all users all functions of the system, giving all users the same baseline for the test.

7.2.1 Observed problems

Sound Playback Error

It was observed through both internal testing and the evaluation process that at almost random intervals, unity would fail to play back the streamed audio from the Elevenlabs API. The logged error indicated it being streamed in an incompatible file format ".webm", however it would then function with the exact same file format generated in the subsequent prompt. The cause of this is unknown, and was occurred infrequently enough that it could be overlooked.

API restrictions during testing

During the evaluation process, testing was temporarily paused due to difficulties in generating audio files across eleven labs. Upon further investigation and a rapid bug fix process, it was determined that the computer being used was blocked from continued usage. This issue was resolved by purchasing an expanded token plan, ultimately resolving the issue.

Overuse of tokens

In combination with the aforementioned API restrictions, it was discovered that the program had also been accidentally sending multiple requests for the same response to the TTS API. In the small window between a response being completed and the request being sent, multiple identical request could and would be sent (2-4 copies). This spamming rapidly drained available resources and was remedied in the aforementioned bug fixing sprint. While the fix did not completely remedy the issue, it did reduce the timeframe that it was possible, leading to very infrequent double sending.

Double generation logging errors

If another generation was triggered roughly simultaneously with the generation time calculation, the corresponding start times would be overwritten, causing unrealistically short recorded generation times. (roughly 0.5 seconds) this however is negligible, as the lowest possible recorded generation time observed in a non VR environment was *1.9* seconds. Herein, this has been used as a minimum threshold for the recorded results.

Additionally, the number of interactions counter (see: section 4.7) logged incorrectly, jumping 200+ interactions within half a second in one case. It is believed this is caused by the incrementation function being accessed rapidly (one or multiple times per frame) causing the number to spike at an impossible rate.

7.3 Potential Future Work

Visual processing model

Given the current implementation, the prompts given to the LLM are identical in nature from instance to instance. Field of view and further context are lacking in the descriptions given. To remedy this, one would either have to excessively expand the detection system or replace it with a fast acting visual description model (Image to text). No such model was found during the exploration of the field, however it is possible such models exist or are soon to be made public.

Larger Environment

In the test environment, as discussed in section 7.2-*User understanding*, users had a general problem with a lack of purpose. Designing a larger or more refined 3D environment would enable users to explore more. Another solution is to add some tasks for the user to complete, which would guide them more through the scene.

Dynamic generation time limits

The interplay between object size, generation time, and user perceptions should by investigated further. Techniques to optimise generation time dynamically based on the scenes context, specifically objects, could further improve user satisfaction. Additionally more research into the psychological processes behind time perception, particularly in VR environments, could also provide deeper insights into how to design more engaging and immersive experiences.

More and Different voices

While the ramifications of voice choice were considered within this project, it was decided that the effects of such a choice would require further exploration. It is believed that potentially changing the narrators voice depending on the tone the scene wishes to portray could be an interesting aspect to look into within future work. Such work would likely be able to stand by itself, as well as supplement the work presented within this paper.

Extended LLM Context

Given that, the current implementation of the LLM considers only that which it has just received and does not carry memories of previous messages within the conversation. It is believed that this is the cause of high message similarity between replies. Given a larger context window and instruction to vary the responses, it is possible that such a model could produce better quality and more varied responses.

Extended LLM testing

Due to the limited implementation time and funding, certain LLM options were never explored, while others were only quickly tested. It would be beneficial to thoroughly test more LLM alternatives, such as ChatGPT or Llama and also further test Phi, OpenHermes and others. This would give a more comprehensive breakdown of their performance metrics and help to choose the best fitting model.

Stronger LLM host

To improve LLM generation time, the LLM was moved to another computer, however the computer chosen was a consumer grade laptop, see section5.1.2. Moving this to a significantly stronger server could drastically reduce generation time and give better user feedback.

Larger Test Pool

The final prototype was tested on 17 test participants, which is a limited test pool, furthermore most of the test participants were university students. Increasing scope, testing on more people to get statistically relevant data, and testing on a wider audience across a wide variety of ages would result in more relevant data.

Chapter 8

Conclusion

This paper introduces a revised taxonomy for generative ML models, categorised into Content Transformers, Content Describers/Transcribers, Content Generators, and Large Language Models. This taxonomy aims to better capture the diverse applications of modern generative ML technologies.

An interactive 3D, VR enabled environment was created in which objects were fitted with descriptors (adjectives). These descriptors were then compiled and sent to a LLM, when a context contingent detection system registered a user interaction. The corresponding response is then sent to a TTS model, resulting in a context aware narration system. This could have been achieved using the Wizard of Oz approach, however fully implementing an LLM adds contextual noise to the results, while also expanding on conventional work on delay acceptability in non GenML systems. This also removes potential problems of Wizard of Oz, such as human error, therefore the data is more reliable.

Through examination of user tolerance to system delays in interactive media revealed that while optimal response times should be under three seconds, however, there is some flexibility depending on user perception and context. Moreover, the impact of narrative tone on user engagement and satisfaction highlighted the importance of strategic narrative design to ensure compelling VR experiences.

A statistical analysis indicated no significant difference in generation time between positive and negative narrations, suggesting that the choice of narration style does not impact the efficiency of generative ML systems. This directly contradicts the qualitative feedback gathered during post-test interviews, which indicated that negative narration had longer generation time. Indicating that the tonality of narration has a direct impact on user's perception of a systems' performance.

Bibliography

[Amazon, 2024] Amazon (2024). Amazon polly api reference. https://docs.aws.amazon.com/polly/latest/dg/API_Reference.html.

[Apple, 2024] Apple (2024). Apple developer machine learning. https://developer.apple.com/machine-learning/. Accessed on February 19, 2024.

[Astica, 2023] Astica (2023). asticavision documentation computer vision api astica. = https://astica.ai/vision/documentation/.

[Autor and Dorn, 2013] Autor, D. H. and Dorn, D. (2013). The growth of low-skill service jobs and the polarization of the us labor market. *The American economic review*, 103(5):1553–1597.

[Betker et al., 2023] Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., et al. (2023). Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8.

[Brizuela and Merch, 2023] Brizuela, R. G. and Merch, E. G. (2023). Chatgpt is not all you need. a state of the art review of large generative ai models. *ArXiv*, abs/2301.04655.

[Chakrabarty et al., 2024] Chakrabarty, T., Padmakumar, V., Brahman, F., and Muresan, S. (2024). Creativity support in the age of large language models: An empirical study involving emerging writers.

[Chamola et al., 2023] Chamola, S. M. I. V., Bansal, T. G., Das, K. L., Hassija, V., Siva, N., Reddy, S. K., Wang, J., Zeadally, S. M. I. S., Hussain, S. M. I. A., Yu, F. I. R., Guizani, F. I. M., Niyato, F. I. D., Sai, S., and Das, V. K. (2023). Beyond reality: The pivotal role of generative ai in the metaverse. *ArXiv*, abs/2308.06272.

[Chen et al., 2021] Chen, Y., Dai, Z., Li, Y., Song, R., Clark, S., Yuan, L., Le, Q. V., and Dai, A. M. (2021). Alphacode: Learning programs from natural language and examples. Technical report, Google DeepMind. Accessed: 2024-02-14.

[Chheang et al., 2023] Chheang, V., Marquez-Hernandez, R., Patel, M., Rajasekaran, D., Sharmin, S., Caulfield, G., Kiafar, B., Li, J., and Barmaki, R. L. (2023). Towards anatomy education with generative ai-based virtual assistants in immersive virtual reality environments. *2024 IEEE International Conference on Artificial Intelligence and eXtended and Virtual Reality (AIxVR)*, pages 21–30.

[Chiu, 2023] Chiu, T. K. (2023). The impact of generative ai (genai) on practices, policies and research direction in education: A case of chatgpt and midjourney. *Interactive Learning Environments*, pages 1–17.

[Diffusion, 2023] Diffusion, S. (2023). Frequently asked questions stable diffusion online. https://stablediffusionweb.com/#faq.

[ElevenLabs1, 2024] ElevenLabs1 (2024). Elevenlabs python api: The official python api for elevenlabs text-to-speech software. https://github.com/elevenlabs/elevenlabs-python.

[ElevenLabs2, 2024] ElevenLabs2 (2024). Elevenlabs voice id's. https://api.elevenlabs.io/v1/voices.

[Epstein et al., 2023] Epstein, Z., Hertzmann, A., Herman, L. M., Mahari, R., Frank, M. R., Groh, M., Schroeder, H., Smith, A., Akten, M., Fjeld, J., Farid, H., Leach, N., Pentland, A., and Russakovsky, O. (2023). Art and the science of generative ai. *Science*, 380:1110 – 1111.

[Gable and Poole, 2012] Gable, P. A. and Poole, B. (2012). Time flies when you're having goal-motivated fun. *Psychological Science*.

[Gemini Team and Google, 2023] Gemini Team and Google (2023). Gemini: A family of highly capable multimodal models. Technical report, Google. arXiv preprint arXiv:2312.11805.

[GitHub, 2021] GitHub (2021). Github copilot - ai pair programmer. https://github.com/github/copilot. Accessed: February 12, 2024.

[Glushak, 2023] Glushak, V. M. (2023). Negation of german polar words and expressions in automated analysis of text tonality. *Philology. Issues of Theory and Practice*.

[Google, 2023] Google (2023). Bard faq. https://bard.google.com/faq. [Online; accessed Today's Date].

[Google AI Team, 2023] Google AI Team (2023). Google gemini ai: A new era of computing. *Google Technology Journal*. Accessed: 2024-02-14.

[Hettmann et al., 2023] Hettmann, W., Wölfel, M., Butz, M., Torner, K., and Finken, J. (2023). Engaging museum visitors with ai-generated narration and gameplay. In *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer Nature Switzerland.

[Jun and Nichol, 2023] Jun, H. and Nichol, A. (2023). Shap-e: Generating conditional 3d implicit functions.

[Junia.ai, 2024] Junia.ai (2024). The ultimate guide to different types of tone in writing. *Junia.ai Blog*.

[Jørgensen, 2024] Jørgensen, E. (2024). https://migogaalborg.dk/kunstig-intelligens-reducerer-ventetiden-pa-skadestuen-i-aalborg/?fbclid=IwAR34gCfcNtHowUDC8CXEdoJAYe-7_xkbEAFv_MBGQK3kr1YlEzwhtsmSwRM.

[Kaplan and Haenlein, 2023] Kaplan, A. and Haenlein, M. (2023). Generative ai can help you tailor messaging to specific audiences. *Harvard Business Review*, 99(2):88–95.

[Lew et al., 2018] Lew, Z., Walther, J. B., Pang, A., and Shin, W. (2018). Interactivity in Online Chat: Conversational Contingency and Response Latency in Computer-mediated Communication. *Journal of Computer-Mediated Communication*, 23(4):201–221.

[LogMeIn, Inc., 2024] LogMeIn, Inc. (2024). Hamachi by logmein. http://vpn.net/. Accessed: 2024-05-01.

[Merriam-Webster, 2024] Merriam-Webster (2024). Metaverse definition & meaning. https://www.merriam-webster.com/dictionary/metaverse.

[Meshy, 2023] Meshy (2023). Meshy | 3d ai generator. https://www.meshy.ai/. Accessed: 2024-02-15.

[Meta, 2024] Meta (2024). Llama 2. https://llama.meta.com. Accessed: 2024-02-12.

[Midjourney, 2023] Midjourney (2023). Midjourney quick start guide. https://docs.midjourney.com/docs/quick-start.

[NVIDIA, 2024] NVIDIA (2024). Megatron-lm: Ongoing research training transformer models at scale. https://github.com/NVIDIA/Megatron-LM.

[OpenAI, 2021] OpenAI (2021). Chatgpt: A large-scale generative model for open-domain chat. [7](https://github.com/openai/gpt-3).

[OpenAI, 2022] OpenAI (2022). Whisper: Robust speech recognition via large-scale weak supervision. https://github.com/openai/whisper.

[OpenAI, 2023] OpenAI (2023). Gpt-4 technical report. *OpenAI Docs*.

[OpenAI, 2023] OpenAI (2023). Vision openai api. https://platform.openai.com/docs/guides/vision.

[OpenAI, 2024] OpenAI (2024). Gpt-4 text-to-speech: A multimodal marvel. https://platform.openai.com/docs/models/tts.

[OpenAI, 2024] OpenAI (2024). Video generation models as world simulators. *OpenAI Research*.

[Otter.ai, 2018] Otter.ai (2018). Otter.ai - ai meeting note taker & real-time ai transcription. https://otter.ai/. Accessed: February 12, 2024.

[Phung et al., 2023] Phung, T., Pădurean, V.-A., Cambronero, J., Gulwani, S., Kohn, T., Majumdar, R., Singla, A., and Soares, G. (2023). Generative ai for programming education: Benchmarking chatgpt, gpt-4, and human tutors. *arXiv.org*.

[Pika.art, 2024] Pika.art (2024). Pika.art: A creative platform for 3d art and animation. https://boximator.github.io/.

[Poly-Haven, 2024] Poly-Haven (2024). Poly haven: The public 3d asset library. https://polyhaven.com/about-contact. Accessed: 2024-04-30.

[Pungas, 2023] Pungas, T. (2023). https://www.taivo.ai/__gpt-3-5-and-gpt-4-response-times/.

[ReadSpeaker, 2024] ReadSpeaker (2024). Text-to-speech plugin - readspeaker. https://www.readspeaker.com/.

[Roundtree, 2023] Roundtree, A. K. (2023). Ai explainability, interpretability, fairness, and privacy: An integrative review of reviews. In *International Conference on Human-Computer Interaction*, pages 305–317. Springer.

[Shokrollahi et al., 2023] Shokrollahi, Y., Yarmohammadtoosky, S., Nikahd, M. M., Dong, P., Li, X., and Gu, L. (2023). A comprehensive review of generative ai in healthcare. *arXiv preprint arXiv:2310.00795*.

[StabilityAI, 2019] StabilityAI (2019). Stability ai: Ai by the people for the people. https://stability.ai/. [Online; accessed Today's Date].

[Sætra, 2023] Sætra, H. S. (2023). Generative ai: Here to stay, but for good? *Technology in Society*, 75:102372.

[undreamai, 2024] undreamai (2024). Llmunity: Create characters in unity with llms! https://github.com/undreamai/LLMUnity. Accessed: 2024-04-24.

[UnrealSpeech, 2024a] UnrealSpeech (2024a). Elevenlabs vs. amazon polly [compare pricing & features in 2024]. https://unrealspeech.com/compare/elevenlabs-vs-amazon-polly-text-to-speech.

[UnrealSpeech, 2024b] UnrealSpeech (2024b). Openai text-to-speech vs. elevenlabs [compare pricing & features in 2024]. https://unrealspeech.com/compare/elevenlabs-vs-openai-text-to-speech.

[Usoh et al., 2000] Usoh, M., Catena, E., Arman, S., and Slater, M. (2000). Using presence questionnaires in reality. *Presence: Teleoperators and Virtual Environments*, 9.

[Verma, 2023] Verma, A. (2023). What is prompt engineering - the ai revolution. *GeeksforGeeks*.

[Wang, 2024] Wang, P. Q. (2024). Personalizing guest experience with generative ai in the hotel industry: there's more to it than meets a kiwi's eye. *Current issues in tourism*, pages 1–18.

[Weidinger et al., 2021] Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P.-S., Cheng, M., Glaese, M., Balle, B., Kasirzadeh, A., Kenton, Z., Brown, S., Hawkins, W., Stepleton, T., Biles, C., Birhane, A., Haas, J., Rimell, L., Hendricks, L. A., Isaac, W., Legassick, S., Irving, G., and Gabriel, I. (2021). Ethical and social risks of harm from language models.

[Wilcoxon,] Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in Statistics*, Springer Series in Statistics, pages 196–202. Springer New York, New York, NY.

[Xu et al., 2023] Xu, M., Niyato, D. T., Chen, J., Zhang, H., Kang, J., Xiong, Z., Mao, S., and Han, Z. (2023). Generative ai-empowered simulation for autonomous driving in vehicular mixed reality metaverses. *IEEE Journal of Selected Topics in Signal Processing*, 17:1064–1079.

[Zheng et al., 2024] Zheng, X., Li, J., Lu, M., and Wang, F.-Y. (2024). New paradigm for economic and financial research with generative ai: Impact and perspective. *IEEE Transactions on Computational Social Systems*, pages 1–11.

APPENDIX A - Implementation Github

Unity Project and Model training on GitHub
https://github.com/Sebastian-Whitehead/MED-8

APPENDIX B - AV Production

Can be found in appendix folder:
appendix / av_production
And seen on:
https://www.youtube.com/watch?v=pwFXYPbehW8

APPENDIX C - Poster

Can be found in appendix folder:
appendix / project_poster

APPENDIX D - Raw Data

Can be found in appendix folder:
appendix / Data