Evaluating Text-to-Speech and Voice Recognition integration into Unity for real-time gaming

Dylan Boylan

BSc. Software Design with VR & Gaming 2024

Technological University of the Shannon

Declaration

I hereby certify that the material, which is submitted in this thesis towards the award of BSc. Software Design is entirely my own work and has not been submitted for any academic assessment other than part fulfilment of the above-named award.

Future students may use the material contained in this thesis provided that the source is acknowledged in full.

Signed – Dylan Boylan

Date – 14/04/2024

Abstract

Text-to-Speech (TTS) and Voice Recognition are two impressive technologies which have both seen significant evolution over the past ten years. Both a tool for accessibility and interaction, both technologies can be seen everywhere, from mobile phone assistants to game design. This research aims to look at these technologies from a game design perspective, and test how this technology performs in a real-time Unity game scenario and discover some use cases for them in game design.

This research will also test performance of these technologies by comparing a variety of different popular, cutting-edge TTS and Voice Recognition models, and testing metrics such as response time, recognition accuracy and reliability of response times.

Acknowledgements

I would like to express my appreciation for my supervisor, John Barrett, who provided me with vital academic guidance throughout my research. I would also like to thank my university, Technological University of the Shannon (TUS), for providing resources and facilities which were crucial to the completion of my research.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Introduction

With the TTS market expected to rise to over ~USD 4353 million by 2028 [1], and with recent advancements and investment into TTS technologies from tech giants such as Microsoft, Amazon and OpenAI; it is clear that TTS is only going to expand and improve. Along with the TTS market, the Voice Recognition market is expected to rise to ~USD 6700 million by 2028, which is why researching such technologies is very important. [2]

This research will be focused on using these technologies for gaming; however, it is important to highlight that these technologies are used in many different technologies used in everyday life, from mobile phones to cars.

1.2 Why is TTS important?

As stated above, the TTS market is expected to grow from ~USD 2187 million in 2022 to ~USD 4353 million by 2028 [1], with huge investments from massive tech companies such as Amazon, Microsoft, and Google. In gaming, TTS has been around for a long time, mainly it is used for accessibility features for people who have trouble reading or are blind. Accessibility for visually impaired people is extremely important as according to the Wired, the number of blind gamers is likely in the tens of millions.

While it is challenging for developers to make a triple A game completely playable for a blind person, The Last of Us Part 2 has seen huge praise for Naughty Dogs effort to make the game so inclusive, offering a huge range of accessibility options.

Another important use for TTS in gaming is for NPC dialogue. For indie developers, voice acting is always a problem due to a lack of budget. Some TTS technologies, such as AWS Polly, aim to make TTS as human like as possible, offering a huge range of voices to choose from.

1.3 Why is Voice Recognition important?

Voice Recognition is also seeing huge improvements and investments from industry giants, such as Microsoft, IBM and Google. As discussed above, the market is expected rise massively, from ~USD 2649 million in 2022, to ~USD 6691 million by 2028 [2]. These figures show that the Voice Recognition market is expected to almost triple in six years, this is likely due to the importance of this technology, and the adaptation of Voice Recognition in so many everyday technologies from smart phones to cars. [15]

One of the most known use of TTS and Voice Recognition is virtual assistants such as Amazons Alexa, or Apples Siri [15]. These technologies aim to perform tasks using voice control, allowing for fun, accessibility and overall making life easier.

In gaming, much like TTS, Voice Recognition allows for accessibility. It does this by allowing the user to translate their voice into text, which can be to perform certain commands/tasks or even control a character using only your voice. An example of this is a game called “Scream Go Hero”, where the player uses only their voice to move the character.

1.4 Research Aims and Objectives

1. To analyse different TTS models currently available and compare them to determine which technologies are best for real-time Unity use.
2. To analyse different Voice recognition models currently available and compare them to determine which technologies are best for real-time Unity use.
3. Explore and integrate TTS and Voice recognition features such as UI narration and voice commands.
4. Deploy these technologies and use the findings to create a game featuring TTS and Voice Recognition technologies.

Chapter 2: Background Research

2.1 Introduction

TTS became extremely popular during the 2010s[15], with devices such as Amazon Echo, and Google Home becoming a common household device. According to consumer research by Ampere Analysis, in 2021, more than half of all households which have internet access in the UK, also own a voice assistant device such as Amazon Echo, and Google Home [15]. With this in mind, the TTS market is only expected to grow; in 2022, the TTS market was worth ~USD 2187 million and is expected to be worth ~USD 4353 million by 2028. [1]

In the past, TTS has always been focused on delivering clear, coherent sentences, but is now more focused on realism, and mimicking human voices. In an article published by Sabir Ahmed, a TTS enthusiast and founder of Fliki TTS, he says "Gone are the days of robotic voices that made you want to hit the mute button. With neural TTS, you can now have a computer-generated voice that sounds almost human-like! Thanks to deep learning algorithms, TTS models can now analyse and mimic human speech patterns, intonation, and pitch, making the experience more natural and engaging." [10]

By analysing TTS trends, such as voicing cloning, emotional TTS and even singing, it is clear TTS is moving in a direction which could bring around huge benefits to gaming, accessibility, marketing and more [10]. However, Sabir talks about how dangerous this technology could be by discussing ethical concerns and breaches of privacy. With that being said, advancements made in the last few years have made serious leaps in making AI voices indistinguishable to human voices, but how do these technologies fare when dealing with real-time scenarios, requiring minimal response times, without sacrificing naturalness of the voice. [10]

Voice recognition, much like TTS is a relatively new technology which has seen a massive surge of popularity in recent years, mostly due to mobile phone assistants such as Apples Siri and Google Assistant [15]. Voice recognition is very closely linked to TTS, so, much like TTS, it is also expected to see a huge rise in the market, from ~USD 2649 million, all the way up to ~USD 6691 million by 2028. [2]

Unlike TTS, voice recognition is already very powerful and is accurate enough to the point where improvements are not being made to the technology as much as there was back in the 2010s. With that being said, there is still some work to be done to improve speech recognition for real-world needs. In a blog post by a researcher named Mike Seltzer, who works for Meta, he claims that Meta is committed to advancing speech recognition technology by addressing smaller problems such as misrecognizing certain rare words, which in his opinion, is enough to ruin the experience. This is a thought worth looking into. [11]

In a fast-paced real-time scenario, where speech recognition accuracy matters, are currently available voice recognition technologies good enough to be widely used in gaming or could small problems such as misrecognizing rare words actually be enough to spoil the experience.

2.2 Real-time use of TTS and Voice Recognition

Problems arise when dealing with TTS and voice recognition real time, especially when small mistakes could yield incorrect results that throw off the realism of the experience. An example of this is with voice recognition in a video game accessibility scenario where the user can only use their voice due to disability, if the voice recognition is not picking up the correct word from time to time, it could lead to frustration and could make the experience poor.

In game design, there are plenty of uses for TTS and voice recognition technologies. Some of these uses include AI Non-player characters (NPCs), User Interface mechanics (UI) and accessibility features like narration. In cases like narration, and UI mechanics which use TTS, it is very possible to pre-generate audio clips and play them when required.

However, for cases like NPC’s, it is better to generate these clips in real-time for a multitude of reasons. One reason for this could be because of AI generated responses for NPC’s; In a scenario where NPC’s can be interacted with and could generate a response using an Application Programming Interface (API) to an application like ChatGPT, responses could be fetched in real-time, thus it is impossible to pre-generate TTS audio clips. Another reason could be because of storage issues; In a game which has hundreds of NPCs with lots of dialogue, these TTS audio clips if pre-generated would take up a lot of unnecessary storage.

Voice recognition also has plenty of uses in game design. Similar to TTS, Voice recognition in general is very useful for accessibility, but could also be used in gameplay features such as movement, talking to NPC’s or voice commands. A game called “Scream Go Hero” by video game publisher Ketchapp, uses voice recognition in a very unique way, by making it the main mechanic in the game. In “Scream Go Hero”, the player uses their voice to move their character, the louder their voice, the higher the jump. This is a great example of how voice recognition could be used as a mechanic in a game, to create hand-free gaming, along with being a tool for creating gaming more inclusive and accessible.

In terms of accessibility, voice recognition is a great technology which makes gaming more accessible for people who have limited mobility. In a research paper published by Moyen Mohammad Mustaquim, Moyen aims to find a standard way of using voice commands in video games, which use voice recognition. He states, “By using automatic speech recognition systems (ASR), voice driven commands can be used to control the game, which can thus open the possibility for people with motor system difficulty to be included in game communities.” [12].

2.3 Problems with Voice Recognition and TTS

Some of the major problems with using TTS in real-time include; Slow response times, quality of the generated audio, problems with replicating tone/emotion and finally the inability to replicate natural rhythm of human speech.

Some of the biggest problems with use of Voice Recognition in real-time include; Slow response times, accuracy of the recognition, problems with accents/languages.

Most of the problems involved in TTS and voice recognition have the potential to destroy immersion, so it is vital to try to minimize these problems by using the best technologies currently available.

2.4 Conclusion

This chapter discussed some of the recent trends with TTS and Voice Recognition and discusses how fast the technology has made its way into everyday life, and where it is going in the future. The statistics shown along with the work of previous research display the importance of this technology which justifies the testing and research done in this study.

The next chapter, “System Design”, there will be in depth discussion and display of the integration process of TTS and Voice Recognition models into Unity, along with a segment dedicated to the requirements of the entire process, and finally a section on the architecture of the design of all the aspects implemented in this research.

Chapter 3: System Design

3.1 Introduction

This chapter will go over in depth the implementation of multiple different TTS and Voice Recognition models into Unity, how fair and extensive testing was done on them to determine their strengths and weaknesses and will justify the conclusions made in the final chapter of this research paper. Also, there will be mention of some important design decisions in a game created for this research called "Project Ozymandias", where TTS and Voice Recognition was implemented into multiple parts of the game to display different use cases for TTS and voice recognition in game design.

The first segment is the "Requirements" section. This section will discuss exactly what was done to compare TTS models, including how the particular TTS and Voice Recognition models were selected to compare to each other and finally, some of the metrics used to compare them, including accuracy, response times, etc.

The next segment in this chapter is the "Architecture" section. This section will go into how the TTS and Voice Recognition models were tested, how the test procedure ensured the tests were fair and how TTS and Voice Recognition models worked with Unity. This section will provide architecture diagrams to display exactly how text for TTS, written by the user, is sent from Unity, through an API and how it is transferred into audio and sent back to Unity where it is played. Similarly for Voice Recognition, how an audio clip recorded by the user, is sent from Unity, through an API and how it is transferred into text and sent back to Unity where it is displayer.

In the "Design" section, the scripts used in testing will be explained, along with how the game that was made for this research which displays a multitude of different uses for TTS and Voice recognition was designed. This section will also explain how UI for the game, the NPCs and finally the story of the game were all designed with TTS and Speech Recognition in mind. All of this is relevant because they show how TTS and Voice Recognition could be used in game design.

The final section of this chapter is the "Implementation" section. This segment will give a dive deep into how the TTS and Voice recognition models were integrated into Unity. There will also be discussions and outlining some issues that were encountered when trying to implement certain TTS or Voice recognition models, and how these issues were either addressed or how other alternatives were found in order to work around them.

3.2 Requirements

There are hundreds of TTS and Voice Recognition models available for integration and use in Unity. So, in order to make a decision on which TTS and Voice Recognition models to use, they could be chosen based on these factors; the TTS or Voice Recognition model must theoretically be possible to implement into Unity, the TTS or Voice Recognition model must be free, or have a free trial in order to fairly compare them, the models must not be offline models (this means that the models used have to use internet, as opposed to running locally). With these factors in mind, the TTS and Voice Recognition models chosen are below.

For TTS, some of the most popular TTS models, including Oculus TTS and AWS Polly stood out, and were selected due to their popularity and constant improvements. However, it was worth looking at a smaller, lesser-known TTS model along with Oculus TTS and AWS Polly, so the final TTS model that would be researched and tested is called Piper TTS, which is a newer, open-source TTS model. Piper TTS was selected because it was open source, it would be interesting to compare this TTS model to models from giant companies like Amazon and Meta. For Voice Recognition, OpenAI Whisper, Oculus Voice Recognition and HuggingFace API were chosen as the three models that I would test and compare.

Again, HuggingFace was the lesser known out of the three, so it would be interesting to see how HuggingFace would compare to the others.

Once all the TTS models are integrated, they will be tested by checking response times, and also the relationship between the length of the sentence being converted from text to speech, and the response time. This will be done fairly by performing a test six times to ensure that there are no external factors such as internet hiccups or one-time API connection issues affecting the result. Once the test has been completed six times, the result taken will be the median of the set, in order to ensure that if there are any severe errors in one of the tests, it won't affect the result. The findings of these tests will be noted down and will be compared to each other at the end. Once all the TTS's have been tested and compared, they will be examined to see their strengths and weaknesses. Finally, a decision will be made to determine which TTS will be best to use in different aspects of game design, depending on their strengths and weaknesses. E.g. If one of the TTS models is better at reading out longer sentences, then it could be better for story narration or something that uses longer sentences. Another example could be if one of the TTS models has quicker response times, then it could be good for something like NPC dialogue, or UI narration.

Similar to TTS, once all the Voice Recognition models are integrated into Unity, they will be tested, and compared to each other. For testing, the first test will check response times for each of the voice recognition models. This test will be done to check response times for one second voice clips, two seconds, four seconds, six seconds and finally ten seconds. Alongside these tests, the accuracy of the text received from the API will be noted and will be compared to the other models at the end. Once again, for each set of tests, there will be six of the same tests done to ensure that any one-time errors don't affect the result. Also, each test in the same set will be done using the same set of words, for example, for the one second voice clip set, the word "Hello" could be used. Once these tests are completed, they will be compared to the other voice recognition models, and a decision will be made to determine which voice recognition model will be best to use in different aspects of game design, for example, if a model has better response times but a lesser recognition accuracy, it might be better for conversional use such as speaking to an NPC or another player. However, if a model has a slower response time, but greater recognition accuracy, it may be better for voice commands or something that requires greater accuracy.

3.3 Architecture

Once the TTS models are integrated into Unity, text can be transferred into audio using Unity. This is done by inputting text into a text box, and then sending this text using an API. Once the model has received this text, it is normalized, processed, and synthesized into speech. It is then sent back to Unity, where it can be played. The entire process could take less than one second, depending on optimization, message length and finally quality of the speech. For neural TTS, this process could potentially take longer. Below is an architecture diagram explaining exactly how TTS and Unity communicate with each other.

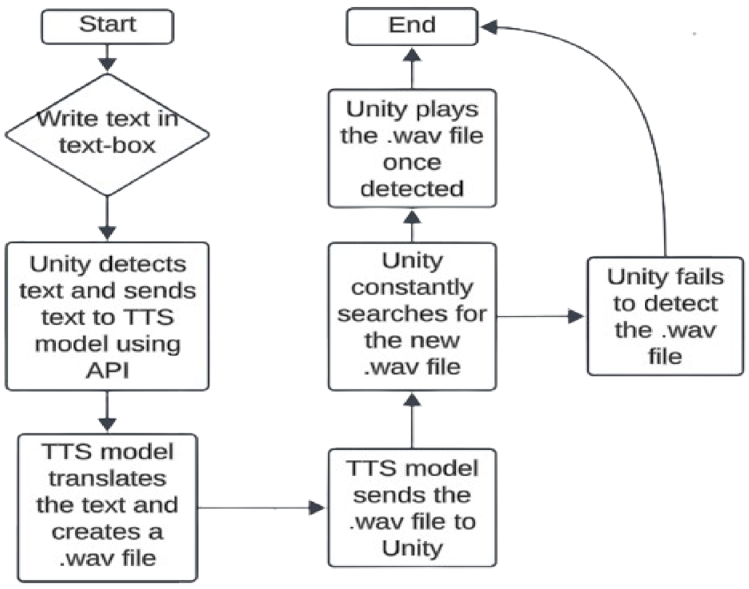
Start
Write text in text-box
Unity detects text and sends text to TTS model using API
TTS model translates the text and creates a .wav file
TTS model sends the .wav file to Unity
Unity constantly searches for the new .wav file
Unity plays the .wav file once detected
Unity fails to detect the .wav file
End

Figure 3.3.1: Simple architecture diagram to display the outputs and inputs of Unity and the TTS model.

Once the Voice Recognition models are integrated into Unity, speech can be transferred into text using Unity, simply by speaking into a microphone. Once the user is finished speaking, the file is saved as a audio file, and is sent to the Voice Recognition model using an API. Once the model has received this file, it is processed, using pattern matching and decoding, and is turned into text. Once it is transferred to text, the text is sent back to Unity, where it will be displayed in a text box. This process could be done in less than a second, but could potentially take longer depending on optimization, size of the audio file and if the model allows for special features such as speaker recognition or emotion detection. Below is an architecture diagram which displays exactly how Unity communicated with a Voice Recognition model.

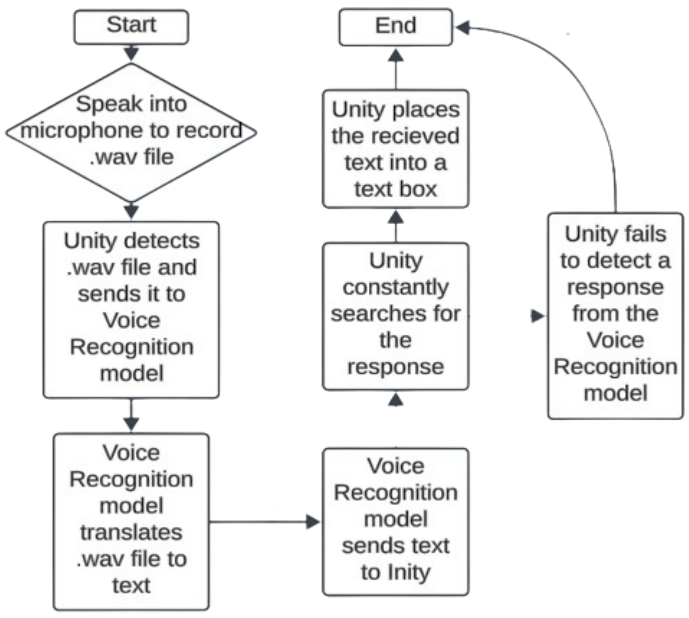
Start
Speak into microphone to record .wav file
Unity detects .wav file and sends it to Voice Recognition model
Voice Recognition model translates .wav file to text
Voice Recognition model sends text to Inity
Unity constantly searches for the response
Unity places the recieved text into a text box
End
Unity fails to detect a response from the Voice Recognition model

Figure 3.3.2: Simple architecture diagram to display the outputs and inputs of Unity and the Speech Recognition model.

Lastly, once all the TTS and Voice Recognition models are implemented into Unity, they will be extensively tested and compared. These tests are designed to be as fair as possible in order to yield the best and most accurate results. The first metric I tested for both TTS and Voice Recognition was response time. For TTS, this was tested by starting a timer once the enter button was pressed, and stopping the timer once the audio file sent by the TTS model starts playing. For Voice Recognition, this was done by starting a timer as soon as the "Stop" button was pressed to stop the recording, and stopping the timer as soon as the text box which would hold the text sent by the Voice Recognition model was received.

In order to ensure that these tests were fairly executed, each test was ran six times, and the median of each test was taken as the result of the text. This was done to counteract any one-time errors that would occasionally occur, such as internet hiccups or failure to connect to the model using the API. If instead the results were calculated by calculating the mean of each set of tests, one bad result could completely throw off the tests and could lead to unfair comparison.

For the Voice Recognition, each test was similarly carried out six times, but in this case, it needed speech. In order to accurately and fairly execute these tests, the tests needed to be done by one person, in order to ensure that different pitches or accents would not be a factor in the test. Also, for each test, the same sentences were used, this is important because there could be issues with using "rare" words, or words that are not as commonly used in sentences as other words might be.

With all these factors taken into account, the tests were carried out, and each result was taken note of using Microsoft Excel. With all of the findings, graphs could be drawn up to demonstrate the results of each test.

3.4 Design

Once the TTS and Voice Recognition models were working and set up for testing, some simple scripts were made to set up timers to accurately measure response times for the TTS and Voice Recognition. For TTS, this was done by starting a timer as soon as the enter button was pressed, and stopping the timer as soon as the audio file sent by the TTS model is received in Unity and starts playing. For Voice Recognition, this was done by starting a timer when the “Stop” button was pressed to stop the recording, and stopping the timer as soon as the text box which holds the text that is sent from the Voice Recognition model to Unity was received.

When the TTS and Voice Recognition models were finished being tested, it was time to add them to the different parts of a game. The first part of the game which was designed to allow for TTS was the UI. In the game, your task is to collect points which you can use to unlock skills in the UI, where there is text displaying what the skill is called, and a brief description of what the skill is. In this section, TTS could be used to narrate these descriptions to make the game more accessible to those who have trouble reading.

The next part of the game which was designed to allow for TTS and Voice Recognition is Story. In the game, the player has to uncover the story by inputting commands into a terminal, the commands used can be found around the map, and once the player put a valid command into the terminal, text will show up giving the player little pieces of the story which when put together, will give the player the full story. This section allows for use of both TTS and Voice Recognition. For the Voice Recognition part, instead of the player manually typing in commands, instead the player can speak into a microphone and speak the words they are trying to input instead. For the TTS, the text displayed when the user enters a correct command would be spoken by a TTS model. Both the use of TTS and Voice Recognition make the game more accessible.

The last part of the game which could include TTS and Voice Recognition is with NPC dialogue. In this game, NPCs can talk to each other, so along with text displaying on the screen, the NPCs dialogue could be spoken by the NPCs using TTS. A good use for Voice Recognition could be speaking to the NPCs. Combining the NPCs speech using TTS and being able to speak to the NPCs with Voice Recognition, the experience could be like speaking to a real person is the NPCs dialogue was generated using AI. However, for an experience like this to be immersive, both the TTS and Speech Recognition would need to have small response times, and good recognition accuracy.

3.5 Implementation

Most of the TTS models had the same process to integrate them into Unity. For the Oculus TTS and AWS Polly, it was quite easy to do this. Firstly, they both supported HTTP API, so it was very easy to connect Unity to Amazons and Metas models. They both used personal randomly generated API keys which could be entered into Unity by using plugins created by Amazon and Meta. Once the API keys were entered into Unity, scripts had to be made to be able to bridge the connection between Unity and Amazon/meta, and to send the text to the model for it to be translated into speech and sent back as an audio file. For Piper TTS, it was a completely different process. Since the model had very support because of how new the model is, and because it didn’t have native Unity support, it was difficult to integrate this into Unity. Because of this, despite plenty of effort and research, a decision was made to leave Piper TTS behind and instead focus on the two other TTS models by Meta and Amazon.

For Voice Recognition, Oculus Speech Recognition and OpenAI Whisper both followed the same process to implement into Unity. Both OpenAI and Oculus provide packages which can be added to Unity by simply importing and opening them. For HuggingFace, it was implemented into unity by using an API key which is generated on Huggingface.co and entered into

Unity where it can be used. HuggingFace also provides a script which can be added into Unity for use. HuggingFace also supports text to image, translation and much more which can be used alongside the Speech Recognition.

3.6 Conclusion

In conclusion, the requirements set for all the TTS and Voice Recognition models were very centred around integration and real-time use with Unity. So the TTS models chosen were Oculus TTS and AWS Polly, and the Speech Recognition models that were chosen were Oculus Speech Recognition, HuggingFace and finally OpenAI Whisper. Originally, Piper TTS was chosen alongside the other two TTS models, but due to issues with integration into Unity, a decision was made to focus on the other two models and stop researching Piper TTS.

Once the models were all chosen, they were tested and compared to each other, where they were then analysed for strengths and weaknesses, and assigned use cases in different section of game design based on these strength and weaknesses.

Some of the sections of game design which were looked at for potential use of TTS and/or Speech Recognition were Story, UI and NPCs. During this research, a game was created where these sections were focused on and designed in a way which could allow for integration of TTS and/or Speech Recognition.

Chapter 4: Testing and Evaluation

4.1 Introduction

This chapter will go over the method used to test the different TTS and Voice Recognition models, the results of the testing, and an overall description of the testing section of this research. The overall purpose of the testing section is to give a clear overview of each of the TTS and Voice Recognition models strengths and weaknesses, along with where they can be used in game design based on those strengths and weaknesses. For example, if a TTS model is particularly strong in having quick response times, then they would be effective in being used in a case which prioritizes quick response times, such as NPC dialogue or some sort of fast paced scenario. Another example for Voice Recognition could be if the player is inputting commands, then it would be more important for the Voice Recognition model to have a higher recognition accuracy, over having rapid response times.

In the "Testing" section of this chapter, there will first be discussion about the setup of the tests, followed by descriptions of the methods used to test the TTS and Voice Recognition models. Next, there will be methods used to ensure fairness across the testing stage. Finally, there will be discussion about versions of applications used during this testing, including the version of Unity, Visual Studio and each of the TTS and Voice Recognition models used. Throughout, there will be rationale provided for decisions made in the testing section.

The next section in this chapter is the "Evaluation" segment. This section will go into detail on the results yielded from the tests and make sense of the results by using graphs and clear descriptions. In this section, there will also be comparisons to other research done on this topic to see how the research done in these tests compare to other tests done on similar contributions.

Finally, the conclusion will give a brief overall perspective of the testing process, including the metrics tested during the testing phase, and how these tests help give a better understanding of where TTS could be used in gaming.

4.2 Testing

To begin testing, first, the testing environment had to be set up. The version of Unity that was used was Unity Version 2021.3.9f1. Alongside Unity, Visual Studio 2019 was used for coding. These versions were kept consistent throughout the testing process in order to ensure fairness. For Voice Recognition, the version of Oculus Speech Recognition was V3.0.2, version 0.8.0 for HuggingFace, and finally version 0.2.0 for OpenAI whisper. For TTS, the version of Oculus TTS was V3.0.2 and version 1.32.74 for AWS Polly. Again, these versions were all kept consistent throughout the entire testing process to ensure fairness.

The TTS models were tested on two metrics, words translated and response time. For the Speech Recognition models, they were tested on two metrics, response time and recognition accuracy. The reason for these metrics was because they best represented the effectiveness of the models.

With the metrics decided on, the next step was the actual process of testing. For both TTS and Voice Recognition, six groups of tests were done for each model, each group of tests were done six times each, to ensure that no one-time errors or external factors such as internet hiccups or API connection issues would throw off the results. Once each group was tested six times each, the median of the results was taken, again, to ensure that no one-time errors would be considered in the overall results.

With the design of the tests done, the next step was to decide what each group of tests would assess. For TTS, it was decided that test one would check response times of a message 10 words long, test two would check

50, then 100, 200, 500, 1000. For Voice Recognition, test one would the recognition accuracy and response time of an audio clip with 1 second of speaking, 2 seconds, 4 seconds, 6 seconds, 10 seconds and finally 20 seconds.

4.3 Evaluation

Below are graphs made to display the results yielded from the tests described above;

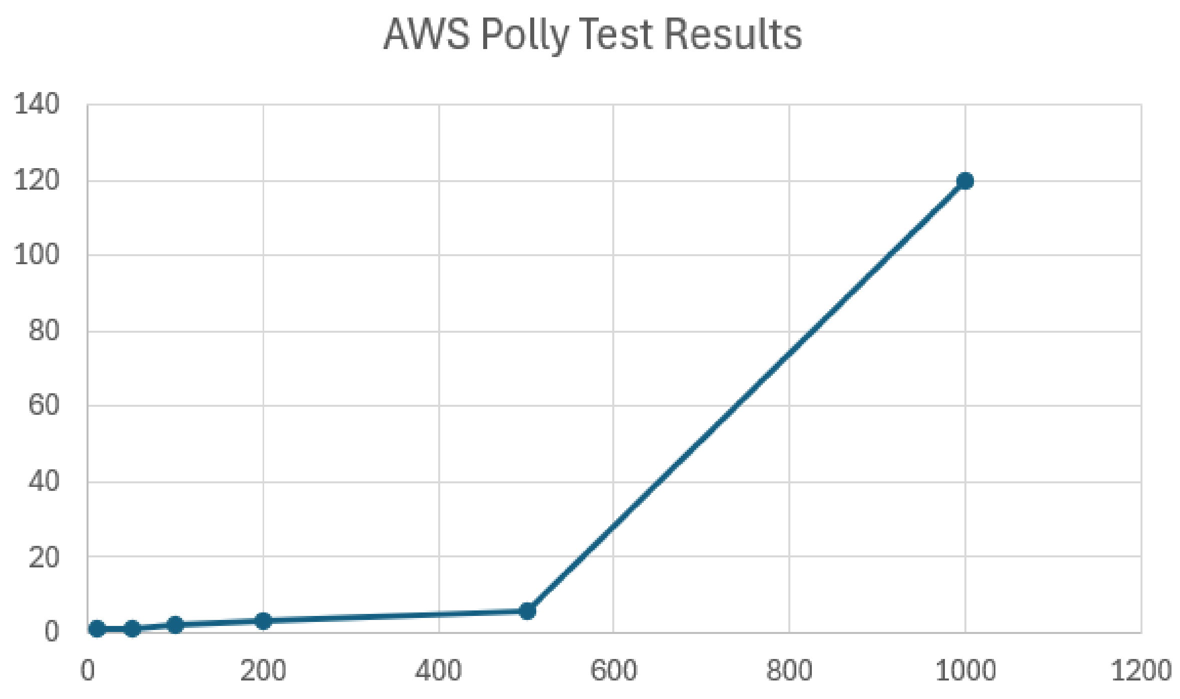

Figure 4.3.1: Relationship between response time (x-axis) and words (y-axis) for AWS Polly

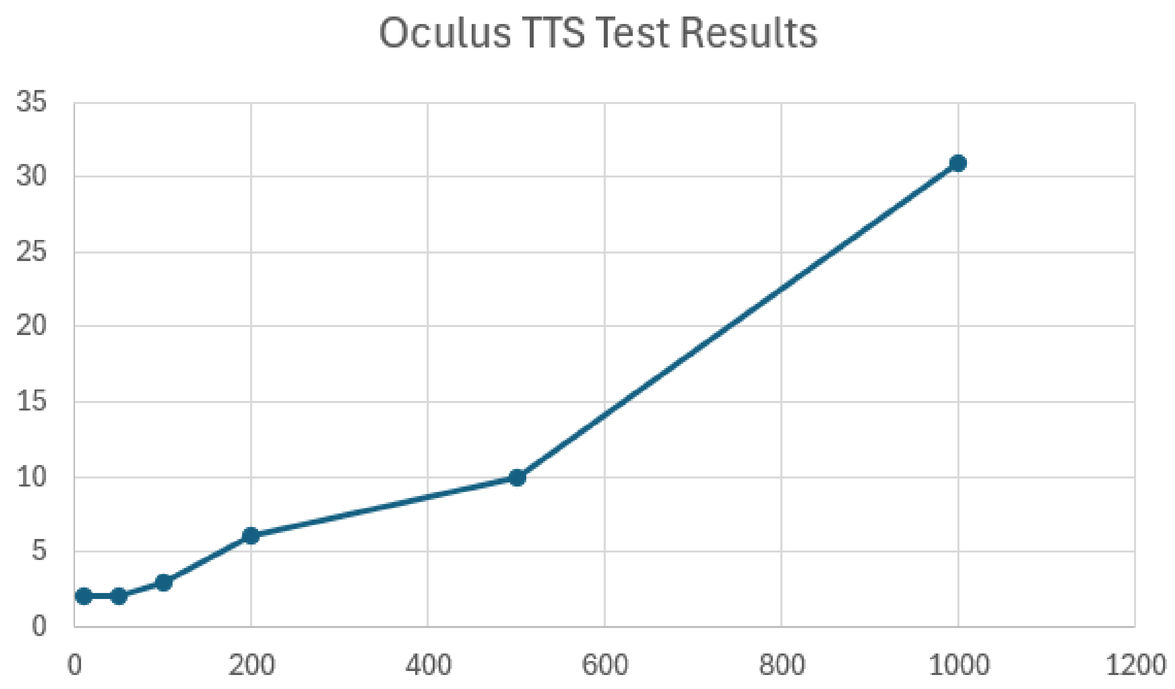

Figure 4.3.2: Relationship between response time (x-axis) and words (y-axis) for Oculus TTS

Figure 4.3.1 and Figure 4.3.2 show the results for the tests done on the Oculus TTS and AWS Polly. From the results, it is clear that in Unity, AWS Polly has all round much faster response times over Oculus TTS. In the AWS Polly results, you can see that the response times are all sub ten seconds, apart from the 1000-word test. This test eventually gave an error every time the test was ran, hence why it goes all the way up to 120 seconds.

For the Oculus TTS test results, you can see that for the lower word tests, all the results have slightly slower response times, but even the 1000-word tests don’t go over the 32 second mark. This means that the Oculus TTS could be better for longer sentences, but AWS Polly is better for shorter sentences.

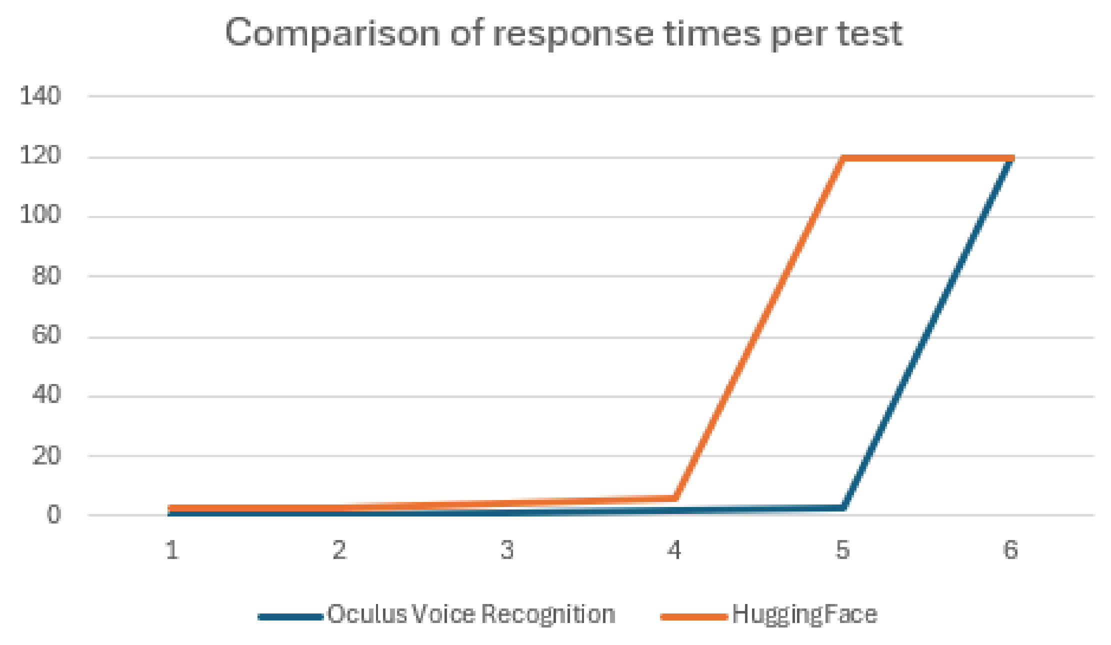

Figure 4.3.3: Graph showing the difference between HuggingFace and Oculus Speech Recognition response times.

As you can see above, both Oculus and HuggingFace have similar response times, with Oculus Speech Recognition having slightly faster response times, and not timing out until the later test. HuggingFace timed out for every test on text 5, which was testing response speeds of 10 second audio clips. However, Oculus only timed out on the final test, which was testing 20 second audio clips. Overall, it can be determined from these results that for unity, Oculus Speech Recognition is the clear choice.

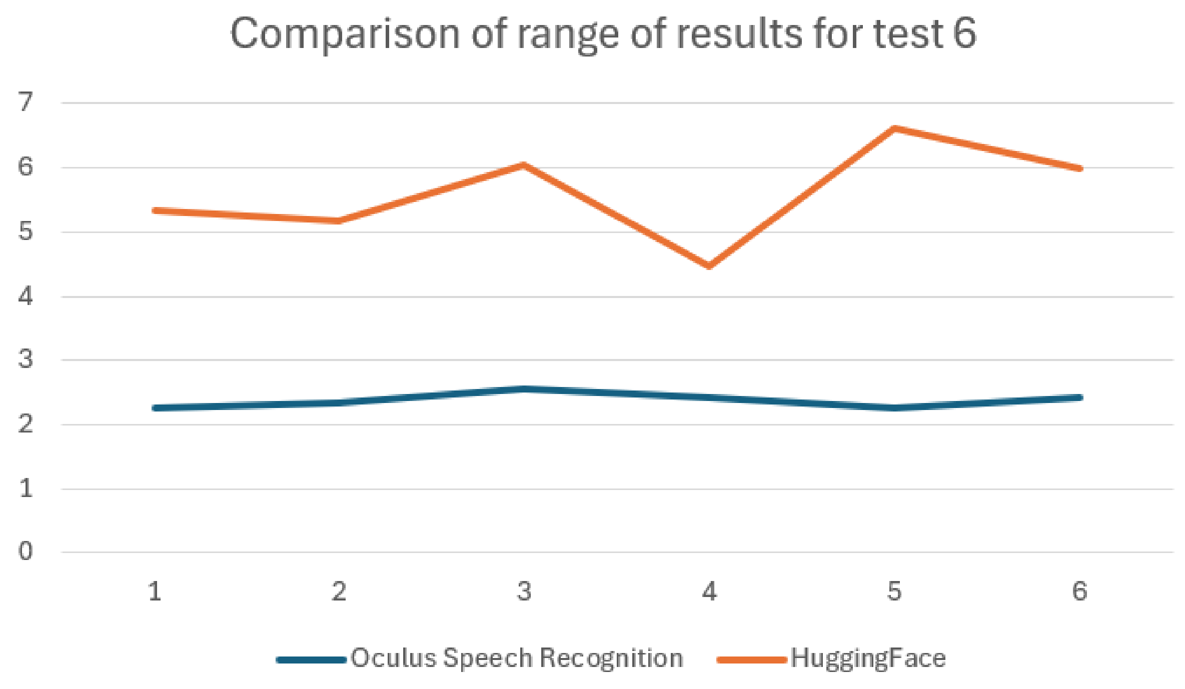

Figure 4.3.4: Graph showing the range of response times from a set of results.

Above is another graph showing the range of response times for test 6. This graph shows that Oculus Speech Recognition also give very reliable response times, unlike HuggingFace which had slightly more varied results. In a study done by a researcher named Haris Isyanto, in a similar study, in one of his graphs, he found that response times varied, but still roughly rounded to the same whole number, +/- 1 [13]. The results for HuggingFace from this research is a little bit more varied than that, but from these results, you could say that Oculus Speech Recognition align with his findings. Below is a graph from his studies.

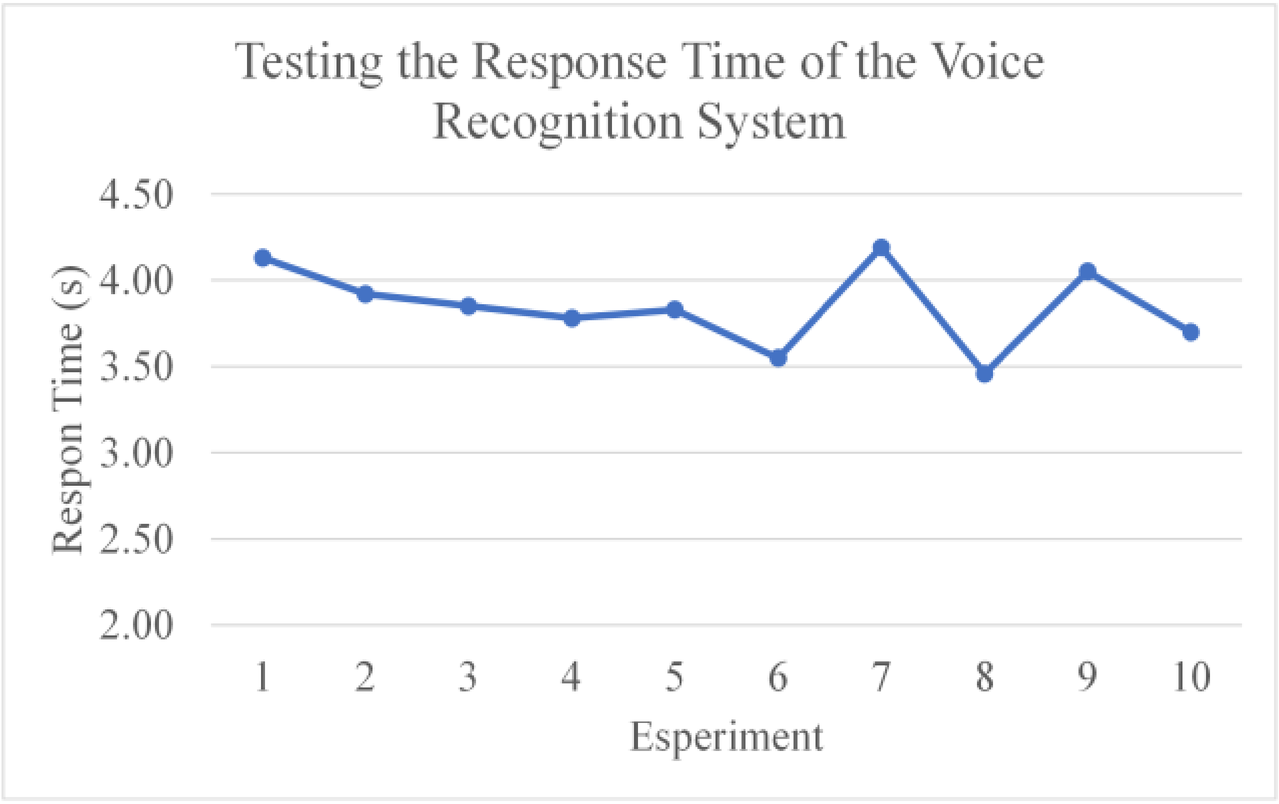

Figure 4.3.5: Graph from Haris Isyanto research on voice recognition reliability [13]

Finally, the last metric that was tested was recognition accuracy for the Voice Recognition models that were tested. From the tests done on Oculus Speech Recognition and HuggingFace, it is evident that Oculus Speech Recognition has a much better recognition accuracy. Oculus had a an overall average recognition accuracy of ~84%, whereas HuggingFace had an overall average of ~70%, a difference of ~14%. In a blog post by a researcher named Mike Seltzer, who works for Meta, he states that misrecognizing certain rare words is enough to ruin the experience. [11] Also in a study released in 2023 by author Bergur Thormundsson, which compared and examined statistics released by Voice Recognition technology companies in 2021, he found that the highest accuracy rate at the time was 86% by a model called Rev.ai, followed by Google, Microsoft, Amazon and other, with a recognition accuracy of ~84%. The results in this study can be backed by these results, along with the results of the studies done by Haris Isyanto, which measured the variability of results.

4.4 Conclusion

In conclusion, the results of the testing phase show that for TTS, there is no clear better choice to choose from, as they both have their own strengths and weaknesses. For AWS Polly, the response times are slightly faster over Oculus TTS, however it was also discovered that AWS Polly would time out when it was sent a text file that was 500 words long, whereas Oculus TTS would time out when it was sent a text file that was 1000 words long.

With these findings in mind, A model like AWS Polly would be better for a scenario which requires rapid response times, such as NPC dialogue. This is because NPC dialogue could be fast paced and would require rapid response times. A model like Oculus TTS could be better for use in a scenario which requires the model to be able to translate massive messages into speech, such as story narration.

Along with analysing the findings from this research, they were compared to previous research done on the same area to see if the findings in this research can be backed, as well as if using these models in Unity would affect recognition accuracy or the range of variation of response times with Voice Recognition. By comparing the results from this study to the results of previous researchers works, it can be assumed that Unity does not negatively or otherwise affect recognition accuracy or the range of variation of response times.

With these results in mind, TTS and Voice Recognition models can be used for real-time Unity use, without sacrificing response times, accuracy, or recognition accuracy. This is very helpful for game developers as with the recent surge in TTS and Voice Recognition technological developments, it means that game designers can use this technology to improve their games in multiple different ways.

The final chapter will give a summary of the thesis, along with implications of this research and will provide a set of recommendations for future research in the area.

Chapter 5: Conclusions

5.1 Introduction

This chapter will provide a full summary of the research and results of this study. In the summary, there will be discussion about how the research aims and objectives which were set out at the beginning of this paper were answered, along with any important notes about some important information that was brought up in the thesis which is relevant but does not necessarily fit in the objectives.

Next, there will be an overall reflection on the implications of this study, why this study is important and who could benefit from this study. Finally, there will be a set of recommendations provided which will talk about how this study could be approved upon, along with some issues identified in this study and how they could be addressed.

5.2 Reflection

The findings in this study show that TTS and Voice Recognition models which are not necessarily created for real-time Unity use, can indeed be used for that purpose. There is a massive list of uses for TTS and Voice Recognition in game design, some of which include, story narration, NPC dialogue, UI narration, sound effects, voice acting and a huge array of options for accessibility just to name a few.

Not only is this study interesting for game developers, but it gives an interesting look at some of the TTS and Voice Recognition models used in this study. For example, it provides a good in depth look at AWS Polly's response times, and limitations. Similarly, for Oculus Voice Recognition, this study gives a good overview of its response times, limitations, range of variation of results, and finally recognition accuracy.

5.3 Recommendations

Finally, this chapter will provide a set of recommendations, which will discuss how this study can be approved upon, along with some issues identified in this study and how they could be addressed. The first issue with the design of this study, was how the TTS and Voice Recognition models were selected. Instead of picking just a popular TTS, and a TTS from a massive company, there should have been more models to compare. This would have given a better overview of where TTS is at this time. Another issue with this selection process, is the range of different uses for these TTS technologies, which were not taken into account in this study. For example, AWS Polly has 47 unique voices which can be used, whereas Oculus TTS only has 14 voices to choose from.

For where this study could go in the future; It could be interesting to look at more metrics to compare these models on, such as naturalness of speech for TTS for example. Also, it would be interesting to look at different features which are provided by a lot of voice recognition and TTS models, such as translation. Another study could look at comparing free to use TTS and voice recognition models to paid models, to see if they are worth the cost when compared using metrics like naturalness, response times and others.

Another study could be integration into other software/ technologies like smart phone applications or VR headsets, this could be done using a similar approach to this study, by first integrating the models into the technology and then testing them. Finally, one idea that was originally meant to feature in this thesis was using an open-source, newer model to compare with massive companies like Meta or Amazon. For this study, a TTS model which stood out was Piper TTS, but unfortunately due to issues with Unity integration, the idea was dropped.

With all these recommendations, it is clear that there is a lot of potential research to be done in this area, especially because of how fast these technologies are advancing, and how fast it came to be in everyday technology. It is exciting to see where this technology, and continued research in the area could lead to in the future.

References

[1] Latest Business Trends & Insights, "*Text-to-Speech Market Size Strategies and Tactics: Top Players Insights, Innovations, & 2031 Outlook*" 2023. Can be found at: https://www.linkedin.com/pulse/text-to-speech-market-size-strategies/ [Accessed 28/03/24]

[2] IRB Trending Market Insights, "*"Speech-to-text API Market Size" | will reach US$ 6691.16 million Million ln 2031*" 2023. Can be found at: https://www.linkedin.com/pulse/speech-to-text-api-market-size-reach-us-669116-eshef/ [Accessed 28/03/24]

[3] 360marketupdates.com, " *GLOBAL SPEECH-TO-TEXT API MARKET PROFESSIONAL SURVEY BY TYPES, APPLICATIONS, AND PLAYERS, WITH REGIONAL GROWTH RATE ANALYSIS AND DEVELOPMENT SITUATION, FROM 2023 TO 2028*" 2023. Can be found at: https://www.360marketupdates.com/global-speech-to-text-api-market-24825159 [Accessed 29/03/24]

[4] industryresearch.biz, " *GLOBAL TEXT-TO-SPEECH INDUSTRY RESEARCH REPORT 2023, COMPETITIVE LANDSCAPE, MARKET SIZE, REGIONAL STATUS AND PROSPECT*" 2023. Can be found at: https://www.industryresearch.biz/global-text-to-speech-industry-research-report-2023-competitive-landscape-market-22381148 [Accessed 29/03/24]

[5] Geoffrey Bunting, " *Games Are More Visually Accessible Than Ever. It's Just the Beginning*" 2023. Can be found at: https://www.wired.com/story/future-video-games-visual-accessibility/#:~:text=There%20are%20an%20estimated%20253,in%20the%20tens%20of%20millions. [Accessed 29/03/24]

[6] Can I Play That?, " *5 Games for Blind and Visually Impaired Players*" 2023. Can be found at: https://caniplaythat.com/2021/10/21/5-games-for-blind-and-visually-impaired-players/ [Accessed 01/04/24]

[7] Rob Clark, Hanna Silen, Tom Kenter, Ralph Leith, "*Evaluating Long-form Text-to-Speech: Comparing the Ratings of Sentences and Paragraphs*" 2019. Can be found at: https://arxiv.org/abs/1909.03965/ [Accessed 01/04/24]

[8] Jonathan Easton, "*Voice assistants in more than half of UK homes*" 2021. Can be found at: https://www.digitaltveurope.com/2021/06/08/voice-assistants-in-more-than-half-of-uk-homes/ [Accessed 01/04/24]

[9] Unreal Speech, "*Exploring the Future of Text-to-Speech Technology: Trends and Applications*" 2023. Can be found at: https://blog.unrealspeech.com/exploring-the-future-of-text-to-speech-technology-trends-and-applications/#the-evolution-of-text-to-speech-from-robotic-to-realistic/ [Accessed 01/04/24]

[10] Sabir Ahmed, "*The Future of Text-to-Speech Technology*" 2024. Can be found at: https://fliki.ai/blog/future-text-to-speech/ [Accessed 01/04/24]

[11] Mike Seltzer, "*New advances in speech recognition to power AR experiences and more*" 2022. Can be found at: https://ai.meta.com/blog/new-advances-in-speech-recognition-to-power-ar-experiences-and-more/ [Accessed 01/04/24]

[12] Moyen Mohammad Mustaquim, "*Automatic speech recognition- an approach for designing inclusive games*" 2011. Can be found at: https://link.springer.com/article/10.1007/s11042-011-0918-7 [Accessed 01/04/24]

[13] Haris Isyanto, "*Voice Recognition Security Reliability Analysis Using Deep Learning Convolutional Neural Network Algorithm*" 2022. Can be found at: https://www.researchgate.net/publication/361630474_Voice_Recognition_Security_Reliability_Analysis_Using_Deep_Learning_Convolutional_Neural_Network_Algorithm [Accessed 01/04/24]

[14] Bergur Thormundsson, "*Speech-to-Text transcript accuracy rate among leading companies worldwide in 2021*" 2023. Can be found at: https://www.statista.com/statistics/1133833/speech-to-text-transcript-accuracy-rate-among-leading-companies/ [Accessed 01/04/24]

[15] Raghu Boddu, " From Siri to Alexa: A Brief History of Conversational AI and Virtual Assistants" 2023. Can be found at: https://www.linkedin.com/pulse/from-siri-alexa-brief-history-conversational-ai-raghu/

Glossary

Table 5.3.1: Glossary table

	Voice Recognition	Technology which translated spoken words into text.		
	Open source	Original source code of the technology is freely available.		
	Real-time	In this context, Real-time means the content generated is happening as the game is active, as opposed to being pre-generated.		
	Recognition Accuracy	Metric used to evaluate a voice recognition models ability to correctly interpret spoken works into text.		

List of Abbreviation

Table 5.3.2: Abbreviation table

	TTS	Text-to Speech		
	API	Application Programming Interface		
	HTTP	Hypertext Transfer Protocol		
	UI	User Interface		
	NPC	Non-player Character		
	ASR	Automatic Speech Recognition Systems		

=================END OF DOCUMENT=================