School of Electronic Engineering
and Computer Science

Final Report

Programme of study:
Information Technology Management for Business with Industrial Experience

Spoken Word: Exploring Accessibility in Video Games

Supervisor:
Dr Tijana Timotijevic

Student Name:
Sebastian Chumaceiro Rangel

Final Year
Undergraduate Project 2020/21

Word Count: 10852
Date: 04/05/21

Abstract

With the increasing popularity and abundance of video games globally, a variety of demographics have begun relying on gaming as one of their primary forms of entertainment. However, a common trend of lacking accessibility features can be found across many of both mainstream and niche games. Though there are many hardware solutions currently out in the market to cater to the physically impaired, the software side of gaming often does not expand far beyond the standard remappable controls, captions and occasionally colour blindness assist options. This project aims to create a simple game and explore alternative ways to play, with a primary focus on using IBM Watson's speech recognition API to assist those with disabilities (primarily motor impairments), who find it difficult to navigate a game and its menus using the usual input hardware.

The objective of this report is to showcase an original game developed with the constraints of disabled users in mind. It will explore a multitude of unique accessibility options and select several considerations to not only facilitate a low barrier of entry, but also expand upon traditional ways of play. The developed game will be a PC application, created using the Unity game engine and will attempt to implement techniques and mechanics from existing computer games that possess exceptional examples of various accessibility options. The report will also contain references to such games and examine how the features implemented there aid disabled users.

An evaluation of the developed solution will also be included, showing how successful the implemented accessibility features are in the eyes of an accessibility expert. Analysing if the tailored accessibility features have promoted access and introduced interesting new gameplay mechanics for disabled users.

Contents

Chapter 1: Introduction 6

1.1 Background 6

1.2 Problem Statement 7

1.3 Aim 8

1.4 Objectives 9

1.5 Research Questions 9

Chapter 2: Market Research & Literature Review 10

2.1 Current Accessibility Approaches in Games 10

2.1.1 How Are Current Approaches Lacking? 11

2.2 Quality of Life, HCI & Inclusive Design 12

2.2.1 How Do Games Help the Disabled? 12

2.2.2 What Makes a Game Unplayable? 12

2.2.3 How Should Games be Designed? 12

Chapter 3: Requirements, Skills & Method of Evaluation 14

3.1 Requirements 15

3.1.1 Basic 15

3.1.2 Intermediate 16

3.1.3 Advanced 16

3.1.4 User Interface Mock-ups 17

3.2 Minimum Viable Product 18

3.2.1 Use Cases 18

3.3 Theoretical Knowledge & Practical Skills 19

3.3.1 Knowledge and Skills Required 19

3.3.2 Resources To Be Used 19

3.4 Gantt Chart 22

3.5 Method of Evaluation 23

3.6 Risk Assessment 24

Chapter 4: Presentation 26
4.1 Development Environment 26
4.1.1 Set-Up 26
4.1.2 Project Structure 27
4.2 Overview of Platform and Technologies 28
4.2.1 Unity Engine 28
4.2.2 IBM Watson 29
4.3 Features Implemented 32
4.3.1 Handling User Input 32
4.3.2 Voice Commands and IBM Services 38
4.3.3 Sequence Diagram 45
4.4 Link to Repositories and Documentation 46
4.4.1 Spoken Word 46
4.4.2 IBM SDKs 46
4.4.3 Unity Engine Packages 46
Chapter 5: Requirements & MVP Review 47
5.1 Requirements Review 48
5.1.1 Basic Requirements Review 48
5.1.2 Intermediate Requirements Review 49
5.1.3 Advanced Requirements Review 49
5.2 MVP Review 51
Chapter 6: Evaluation 52
6.1 Self-Evaluation – Spoken Word Walkthrough 52
6.1.1 KM / Gamepad Walkthrough 52
6.1.2 Voice Controls Walkthrough 55
6.1.3 Summary 58
6.2 Evaluation of Work 68
6.2.1 What Could Have Been Done Differently 68
6.2.2 Future Work 69
Chapter 7: Conclusion 70

References 71

Appendix A 74

7.1.1 General 74

7.1.2 Motor 75

7.1.3 Cognitive 77

7.1.4 Vision 78

7.1.5 Auditory 79

Sdfs**Error! Bookmark not defined.**

Appendices (as needed) ..... **Error! Bookmark not defined.**

Chapter 1: **Introduction**

1.1 Background

As of 2019 the most common reported forms of disability in the UK were those concerning mobility, stamina, and dexterity, taking up around 48%, 36% and 26% of the disabled population respectively. With other impairments such as vision and hearing impairments taking up a combined 23% of the disabled population. What is most alarming about this data however, is that between 2009 and 2019 the percentage of the UK population that reported a disability has increased from 19% to 21%. (Family Resources Survey 2018/19, 2020).

Over the years, video games have become more and more mainstream as a form of entertainment. For example, in 2018, the global gaming market generated a value of $134.9B (Newzoo, 2018), a figure that has continued to grow, and has been estimated to reach revenues of $159B by the end of 2020 (Reuters, 2020). While an explanation for the recent growth is in no doubt partly due to the recent COVID-19 induced lockdowns, it can also be explained by the unique experiences provided by the medium through its degree of interactivity. One that cannot be imitated by other entertainment mediums such as books and films. Yet, as the popularity of video games increase, so too does the group of people unable to play video because of their disabilities.

In recent years there have been examples within the games industry of large developers and hardware manufacturers catering to disabled gamers, with products like Microsoft's Xbox adaptive controller released in 2018 and Logitech's extension on it, with their adaptive gaming kit. However, there have been even fewer standout examples of innovation in software, with Naughty Dog's *The Last of Us Part 2* being part of that memorable few; whilst the majority of other popular games will go not expand beyond the standard captions options. To make matters worse, massively popular games intended for all audiences such as Nintendo's *Animal Crossing: New Horizons* released in March of 2020 have made it impossible to change anything about the presentation of the game with the absence of any game options or accessibility options whatsoever. The focus of this project is then to create a game that implements many accessibility considerations both common and uncommon in existing games and other software, as well as evaluate the result to measure the chosen method's effectiveness in promoting accessibility and combatting current industry trends.

1.2 Problem Statement

In 2010 a study conducted in the U.S. determined that an estimated 2% of the U.S. population is unable to play video games entirely, and that an additional 9% of the U.S. population can play games, but at a diminished experience due to disability or physical impairment (Yuan, Folmer and Harris, 2010). Another survey conducted 2 years prior by Information Solutions Group for the game development studio PopCap Games found that approximately 1 in 5 casual gamers had some form of physical, cognitive, or developmental disability (Garber, 2013). Though the population of disabled gamers is high, the vast majority of modern video games have a severe lack of accessibility options, and the needs of the disabled are often not considered by developers.

There are many speculations as to why developers seem to be disregarding accessibility as a priority, but according to Garber it most likely in part due to developers being unaware of the issues experienced by the disabled, and not realising just how many people would benefit from built-in accessibility in their games. To make matters worse, commercial video games are expensive to make. Development studios are faced with limited hours, manpower and funding and the extra workload required to accommodate the disabled most times does not make the cut. In fact, the list goes on; with some game developers concerned that the inclusion of accessibility options could compromise the challenging aspects and design choices of their game (Garber, 2013).

On the other hand, game developers that have had noteworthy implementation of accessibility considerations in the past, such as Valve's implementation of captions in 2004's Half Life 2 (noteworthy at the time), found that the impact on the project's schedule was negligible (Bierre, Chetwynd, Ellis, Hinn, Ludi, Westin, 2005). Additionally, developers can implement accessibility considerations without compromising gameplay if the considerations are evaluated at the time of conceptualising the gameplay. What this means is that accessibility features would not affect the game loop negatively. Something that works in favour of both developers and disabled gamers, as the thing that accessibility advocates want most is accessibility, not at the expense of the gameplay.

1.3 Aim

Games are a versatile medium, they provide a virtual space to do things that would otherwise be impossible for many to accomplish. For disabled gamers especially, games allow them to experience what many people take for granted in daily life. It helps them connect socially despite geographical boundaries and physical limitations, and most importantly, it helps them overcome anxiety and feelings of depression that many people with disability face (Miller, 2019)

The aim of this project is to develop a simple video game using the Unity engine that showcases a variety of different techniques and methods that accommodate disabled gamers. Though the focus will be on using the IBM Watson text to speech SDK to alleviate issues experienced by people with motor deficiencies, there will also be other accessibility options, some commonly seen in modern games, and others that are not so common.

The game to be developed will be named Spoken Word and is to have accessibility considerations throughout the entire game, from the game design and gameplay to the layout and navigation of menus and UI. Many existing video games that do have accessibility options hide them away deep inside menus and sub menus making it difficult for some users to reach them in the first place. Spoken Word will aim to reduce such issues as best as possible.

Spoken Word will implement a variety of different ways to access game mechanics that are standard in the medium and essential to the core gameplay of Spoken Word. For example, character movement will be able to be controlled with standard keyboard / mouse or controller input as well as via voice input using a microphone. Additionally, controls that are specific to certain puzzles will also have voice input alternatives.

All considerations will be implemented via software solutions. External hardware support for assistive technologies like switches, eye trackers and blink detectors, though ideal, are not the focus of Spoken Word as they would be unlikely to be implemented in commercial games. The aim of Spoken Word is to provide a collection of considerations currently used and likely to be used in commercial games to both enhance gameplay and promote access, such as with the use of voice recognition.

The overall aim of this project is to present disabled gamers with a game that has no trouble adapting to the specific needs of the individual, as well as to provide an example to the industry of how games could and should be designed to accommodate more players.

1.4 Objectives

To achieve this aim, the following objectives will have to be met:

- Investigate the current techniques being used in the industry to promote accessibility. Summarise the impact each technique has on an application.
    - Identify which of these techniques could be implemented into Spoken Word.
- Investigate existing video games, regardless of platform, that have shown exemplary consideration for accessibility, focus on motor impairment.
- Identify what worked best in those games.
    - Analyse how each consideration worked regarding the technology behind it.
    - Analyse if and how each consideration changed the experience regarding the presentation of the game (gameplay). Did it compromise gameplay in any way?
- Develop a simple game with a few varied puzzles and a short narrative that uses many common and uncommon accessibility considerations.
- To implement use of the IBM Watson Unity SDK and utilise text to speech for many gameplay elements.
- To evaluate the accessibility of the developed game through a subject matter expert's evaluation.

1.5 Research Questions

The following questions will be explored throughout the development of Spoken Word: (Some overlap with objectives above)

1. What are the current approaches to accessibility seen in games?
    a. Are they lacking in any way?
2. Do video games improve the lives of the disabled, and if so, how?
3. What elements of video games are the most tolerable by people with motor impairments, and what elements are least tolerable?
    a. What makes a game unplayable by someone with a particular disability, and what are they looking for when purchasing a new game.
4. How can games be best tailored to the disabled, without compromising game design or drastically altering the player's experience.

Chapter 2: **Market Research & Literature Review**

2.1 Current Accessibility Approaches in Games

Using the Game Accessibility Guidelines recommended by the International Game Developers Association, Game Accessibility Special Interest Group (henceforth, GASIG), this section will explain the different approaches to promoting accessibility in games that are possible and have been seen in past games. All approaches analysed will be grouped in categories relating to the corresponding impairments they are accounting for. In each category, the approaches that are to be implemented in the project will be highlighted in the final column to show which approaches will be applied to Spoken Word (Y for “yes”, N for “no”). Sources can be found here, GASIG, *igda-gasig.org*; Accessibility Guidelines, *gameaccessibilityguidelines.com*.

Please see Appendix A for a full breakdown of accessibility considerations.

2.1.1 How Are Current Approaches Lacking?

One of the ways that current games are struggling in when approaching accessibility is in building a completely accessible product. As mentioned previously in this report, many games only implement considerations that are standard across the industry, such as subtitles and colour blindness assist. Whereas there are very few examples of a completely accessible game. The first issue resides in the difficulty of defining complete accessibility. Specific needs vary from individual to individual so building something that meets the needs of everyone would be impossible. Where developers should focus their efforts however is in building enough considerations into their game so that a group of players with a variety of different abilities, disabilities and in different environments are able to adapt the game to experience it comfortably and suit their needs.

Another way that current games are lacking, with the exception of a niche few, are that the game design is not conceived with accessibility considerations in mind. For motor function specifically, developers will often implement alternative input support on PC releases but not implement unconventional input methods like described in *G5* and *G6* because of game design challenges. Some gameplay elements, like driving for example, would be extremely difficult to retrofit into existing gameplay systems midway or towards the end of the development lifecycle as entire control systems would have to be rewritten to accommodate control via a microphone. To solve this, developers need to employ inclusive design from the beginning of the development process. By doing so, designers and artist do not have to compromise the game's vision, and engineers are able to develop the systems required for the chosen considerations.

2.2 Quality of Life, HCI & Inclusive Design

This section will aim to help provide an answer the four questions posed in section 1.5.

2.2.1 How Do Games Help the Disabled?

As explored previously in section 1.3, games enable the disabled to do things that would normally not be so easy for them. Serious games, those designed for purposes other than entertainment, can help with developing math, language and reading skills (Griffiths, 2002). While literature is sparse, serious games have also shown to improve the cognitive abilities of those with intellectual disabilities (Jiménez, Pulina and Lanfranchi, 2015). Such benefits are not limited to serious games either, a study by Ilg et al., 2012, showed that a group of 10 children who played a variety Xbox Kinect games (a motion sensor for the Xbox console) found that at the end of the study, the children's symptoms of ataxia had been somewhat alleviated. Similarly, a feasibility test for the rehabilitation of teenagers with cerebral palsy found the game *Nintendo Wii Sports* did help with the process of rehabilitation (Deutsch et al., 2008).

Outside of alleviating symptoms of disabilities however, games also help in areas that would be difficult for people with disabilities such as socialising, playing sports (albeit this would take place in a virtual space), and travelling (also virtually) (Miller, 2019).

2.2.2 What Makes a Game Unplayable?

Assuming a user has no disabilities other than motor-function related impairments, it would be safe to say that the aspects that make a game unplayable for them would be those that make most use of fine motor controls and manual dexterity (Langdon et al., 2014). As expanded upon previously in section 2.1 and Appendix A, any gameplay elements that make use of fine motor control or manual dexterity will be supplemented with alternative controls and the through the implementation of voice controls in Spoken Word.

2.2.3 How Should Games be Designed?

Thus, the question of how games should be designed (and developed) comes into focus. When designing accessible games, we want to be sure that not only do they avoid aspects that make it unplayable for people with certain disabilities. But if it's the focus also provide some sort of benefit beyond simply entertainment. As described by Grammenos et al., there are usually two approaches that can be adopted when

addressing the issue of accessibility in video games. First, is making inaccessible games operationally accessible using third part technologies. In practice, the issues arise through inherent barriers and bottlenecks from incompatibility between the two systems. Success can be achieved given enough effort is put towards compatibility, though Grammenos argues that often it is because of coincidence rather than appropriate design and development considerations.

The second approach is when games are developed from scratch, targeted at a specific audience, such as developing an audio-based game for the visually impaired. The first approach can lead to poor interaction quality and usability, and while the second approach is more promising, it can lead to inflated development costs and a limited return on investment. With the market for games specifically tailored to certain categories of impairment being much smaller.

Spoken Word will use a hybrid of the two approaches, developing a game from scratch that can be enjoyed by anyone regardless of having a disability or not. Whilst making use of third-party technology (IBM Watson) to facilitate interaction for users with motor control deficiencies. This should mitigate the interaction issues described in the first approach, as bespoke accessibility considerations will be developed for Spoken Word; the issue of a small market will also be addressed as Spoken Word will be built for able and disabled players alike.

Chapter 3: **Requirements, Skills & Method of Evaluation**

This section will detail the set of requirements identified for Spoken Word and the required theoretical knowledge and practical skills that I will need to develop Spoken Word. The requirements will be split into 3 tables, corresponding to basic, intermediate, and advanced requirements respectively. Each grouping will feature a combination of both functional and non-functional requirements, sub-categorised into the 3 main categories of impairment, motor, sensory (audio and visual), and cognitive using the consideration keys found in section 2.1. Requirements will end with design mock-ups of the game's initial settings screen and user interface.

Following that I will detail the theoretical knowledge and practical skills I will need to develop to create Spoken Word as well as a description of the minimum viable product to be developed. Finally, this section will conclude with a brief description of how Spoken Word will be evaluated to measure its success in applying accessibility consideration and inclusive design.

3.1 Requirements

The following requirements will describe the key characteristics that are to define Spoken Word as an accessible product. They are ordered from basic to advanced, in regard to difficulty to implement, my personal predisposition of order to implement, and from minimum viable product to the ideal (as in commercially viable) product. Under the *Type* column, '*F'* will replace '*Functional'* and '*NF'* will replace '*Non-Functional'*.

3.1.1 Basic

Key	Description	Type	Priority	Consideration
Br1.	A fully functional 3D exploration / puzzle game with a simple narrative.	F	High	C3
Br2.	Design a simple and intuitive control scheme that requires little learning.	NF	High	M8
Br3.	Standard control input using keyboard and mouse.	F	High	M3
Br4.	At least 1 puzzle to be completed.	NF	High	N/A
Br5.	Standard control input using a Microsoft Xbox game pad.	F	High	M3
Br6.	Speech recognition using the Watson Speech Recognition API for alternative character control input.	F	High	G5, G6, M3
Br7.	Speech recognition that allows for intuitive navigation of dummy menus.	F	High	G5, G6, M3, M9
Br8.	Game mechanics in the core gameplay loop that are designed entirely around and rely on speech recognition.	NF	High	G6, M3, M9
Br9.	Ensure all information essential to the gameplay and narrative are conveyed clearly and in multiple forms.	NF	High	C2, V1, V3, A1, A4, A5
Br10.	Subtitles and captions options are present for all audio tracks that provide information to the user.	F	High	A1, A4, A5
Br11.	The game must greet players with all accessibility options upon launching the game.	F	High	G2, A3
Br12.	Offer a list of included accessibility features upon launching the game.	NF	Medium	G2
Br13.	Ability to adjust look sensitivity.	F	Medium	M1, M2
Br14.	Present tutorials for all gameplay elements and sequences.	F	High	C2, V3
Br15.	Clear, high contrast and adjustable size font for all information conveyed via text.	F	High	C1, V2, V4, A1, A3, A4
Br16.	Avoid flickering images, quick movements and unexpected events happening on screen.	NF	High	M5, M7, C4

3.1.2 Intermediate

Key	Description	Type	Priority	Consideration
Ir1.	Allow controls to be remapped to the user's liking.	F	Medium	M1, M2, M5, M8
Ir2.	Allow the game to be played in either windowed or full screen mode. Resolution must be able to be adjusted as well.	F	High	M6
Ir3.	Offer alternatives to inputs unique to certain puzzles or gameplay sequences.	F	Low	G5, M1, M3, M6, M8, M9
Ir4.	At least 3 puzzles to be completed.	NF	Medium	N/A
Ir5.	Provide volume controls for all audio tracks. E.g. effects, music and dialogue.	F	Low	A2, A3

3.1.3 Advanced

Key	Description	Type	Priority	Consideration
Ar1.	Provide the ability to adjust post processing effects like contrast and saturation.	F	Low	V1, V2
Ar2.	Screen reading for any on-screen text in the game and menus.	F	Low	V5
Ar3.	The game should have a complete, concise narrative and game loop with 3 to 5 clear objectives. Should not feel like a demo or trial.	NF	High	N/A
Ar4.	Adjustable graphics options for low-end computers.	F	Low	M6, V1, V2

3.1.4 User Interface Mock-ups

The following two images are user interface mock-ups of the core game loop screen and start screen. Subject to change during development.

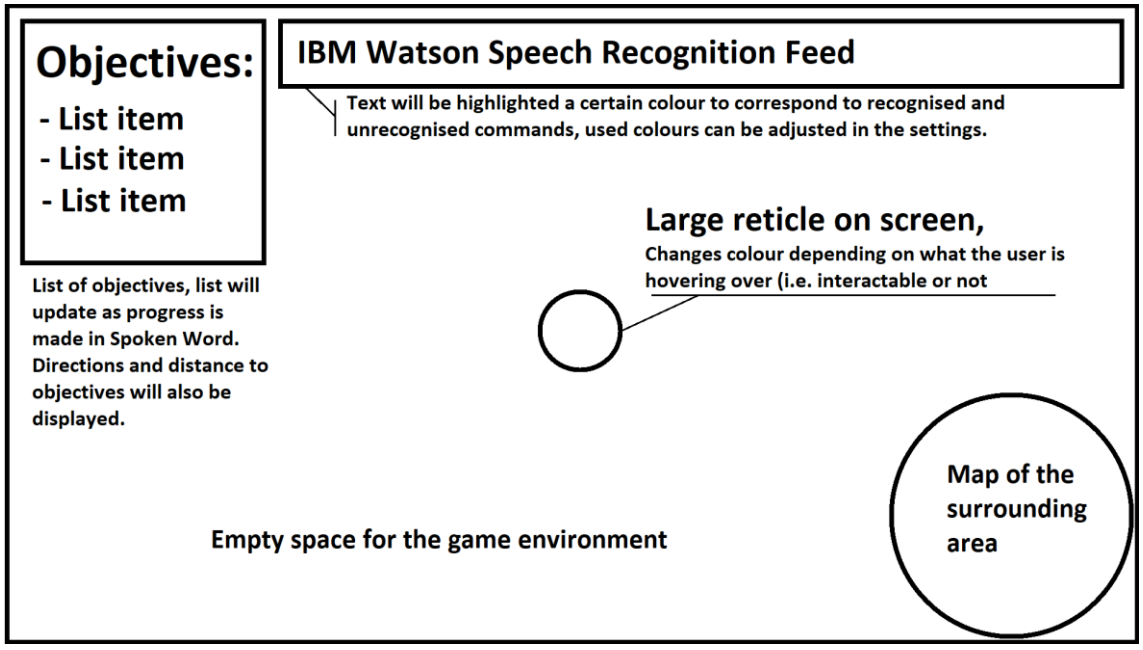


Figure 1. Game Loop User Interface Draft

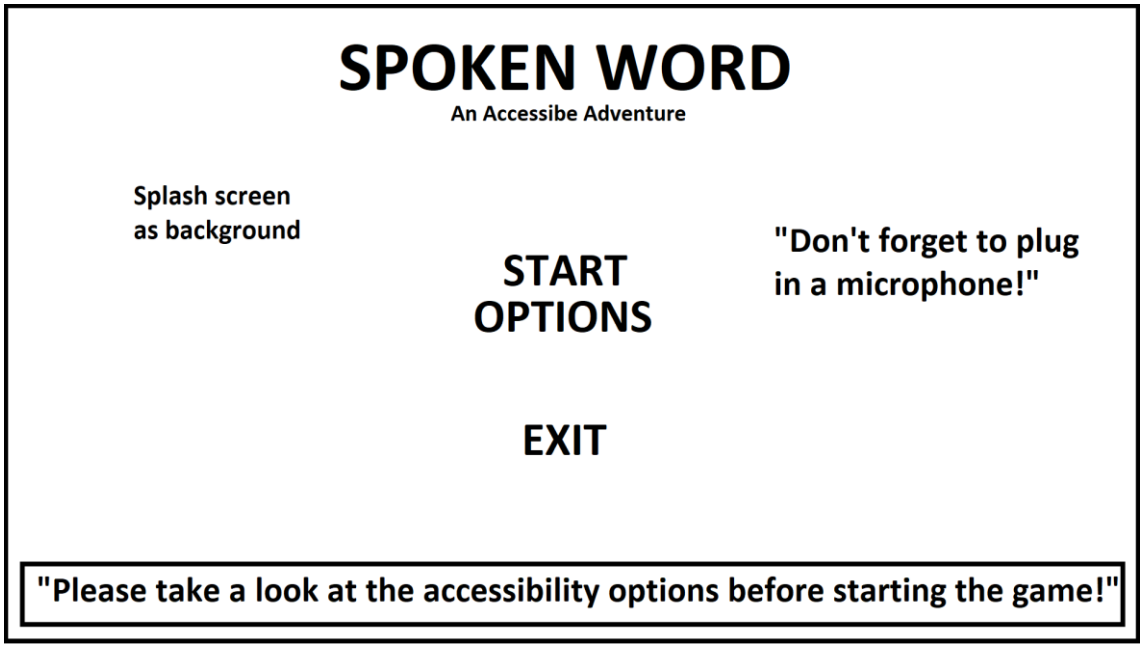


Figure 2. Start Menu User Interface

3.2 Minimum Viable Product

To achieve the MVP version of Spoken Word, the minimum requirements to be fulfilled would have to be all the basic requirements, Br1 to Br16, in addition to Ir1 and Ir5. The reasoning behind this being that these requirements would have to be met just to come close the current accessibility standard of modern commercial games, while still lacking major general settings such as graphics options. At this requirement level, Spoken Word would also be unique from other games through its use of speech recognition as a gameplay element. Something that few games make use of at all. The use of speech recognition (Br6, Br7) would promote access to players with sever motor function impairments, while also providing an interesting way to solve gameplay challenges. Ideally, Spoken Word would also still implement standard accessibility considerations used by many people both able bodied and otherwise, such as captions, remappable controls and audio controls. However, this is not required by the MVP, as the purpose of Spoken Word is to evaluate accessibility considerations aimed at those with motor control impairments in the context of video games. The primary consideration being voice controls.

Spoken Word can then be improved upon as a game by implementing requirements Ir4 and Ar3. It can also be improved upon as an accessible product by implementing the remaining requirements, Ir2, Ir3 and particularly Ar1, Ar2 and Ar4. Ar4 benefitting all users, as PC specification does not correlate with the ability of the user.

3.2.1 Use Cases

Spoken Word will have one main use case from the perspective of the player. Regardless of the player's ability and experience with games or their impairment, a player will play the game for enjoyment or to try out the voice controls out of curiosity. The game will start when the player decides and will continue until the player decides to close the game or completes all objectives. At that point Spoken Word will be complete and the application will close. The player can choose to either play the game using voice controls or standard controls. Additionally, they can choose to adjust the available settings such as character look sensitivity to suit their needs.

Hence, a use case diagram appears as follows:

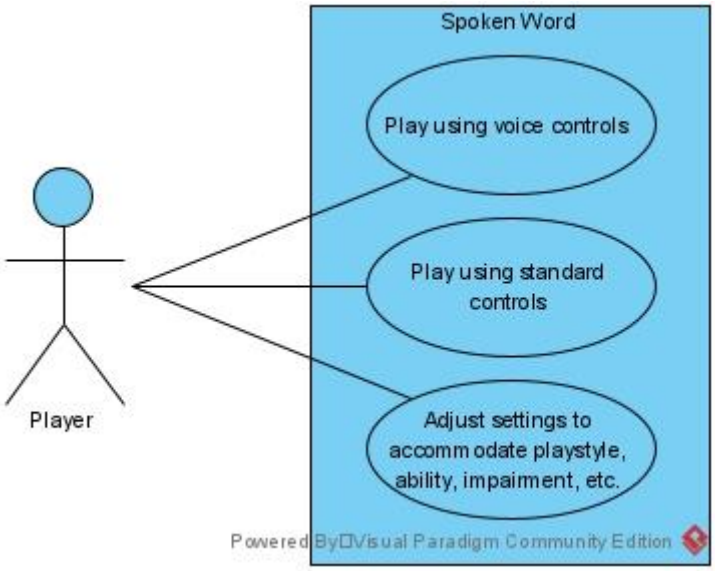


Figure 3, Spoken Word - Use Case Diagram

3.3 Theoretical Knowledge & Practical Skills

3.3.1 Knowledge and Skills Required

The game engine chosen to develop Spoken Word is Unity 3D, partially because of my prior experience with C#, the default scripting language used in the engine, and partially because of the existence of the IBM Watson Unity SDK. Though I have previously used IBM Watson during my industrial placement in 2019/20 I had not been exposed to the Unity SDK. Additionally, the official Unity Scripting APIs will be used in the project for some of the advanced requirements, such as Ar1 and Ar2 where engine code may need to be modified (provided those requirements are implemented). Therefore, this period in which I will be learning the Unity engine, learning how to use the Watson SDK, and creating prototype games has been accounted for in the project Gantt chart.

In terms of the theoretical knowledge required, I will need to conduct more research on inclusive design for games, and game design in general so that I am able to create engaging gameplay that promotes accessibility without altering the experience.

3.3.2 Resources To Be Used

The resource I will use to learn the Unity engine will be the *Complete C# Unity Game Developer 3D* course by Ben Tristem and Rick Davidson. In addition to online resources and articles that I will reference as I progress through requirements. This course should also aid me in learning game design principles and in creating an engaging game. For inclusive design I will be using the book *Inclusive Designing* by P.M. Langdon and J.

Lazar. Later in the development of Spoken Word I hope to be able to use what I learn in the Interaction Design module I will be taking next semester to review the prototypes I create before then.

Finally, for inspiration I will be looking at existing games that have stood out for innovating on accessibility within games, such as *The Last of Us Part II* and *Uncharted 4*.

Figure 4. Vision Accessibility Pre-set - The Last of Us Part II

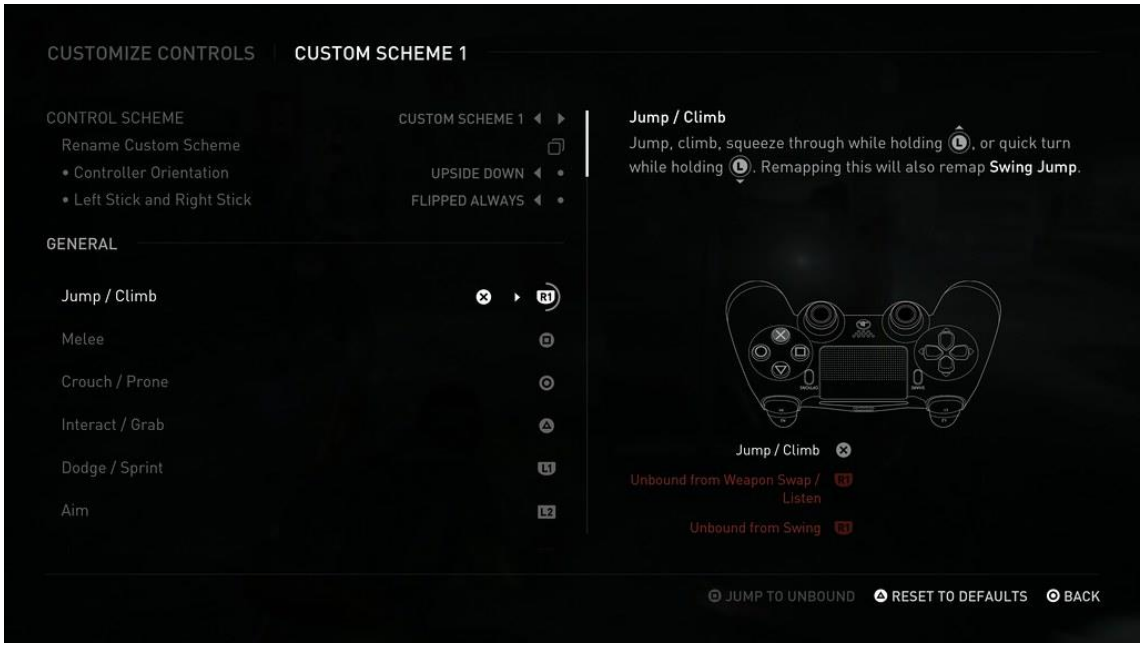


Figure 5. Alternate Controls - The Last of Us Part II

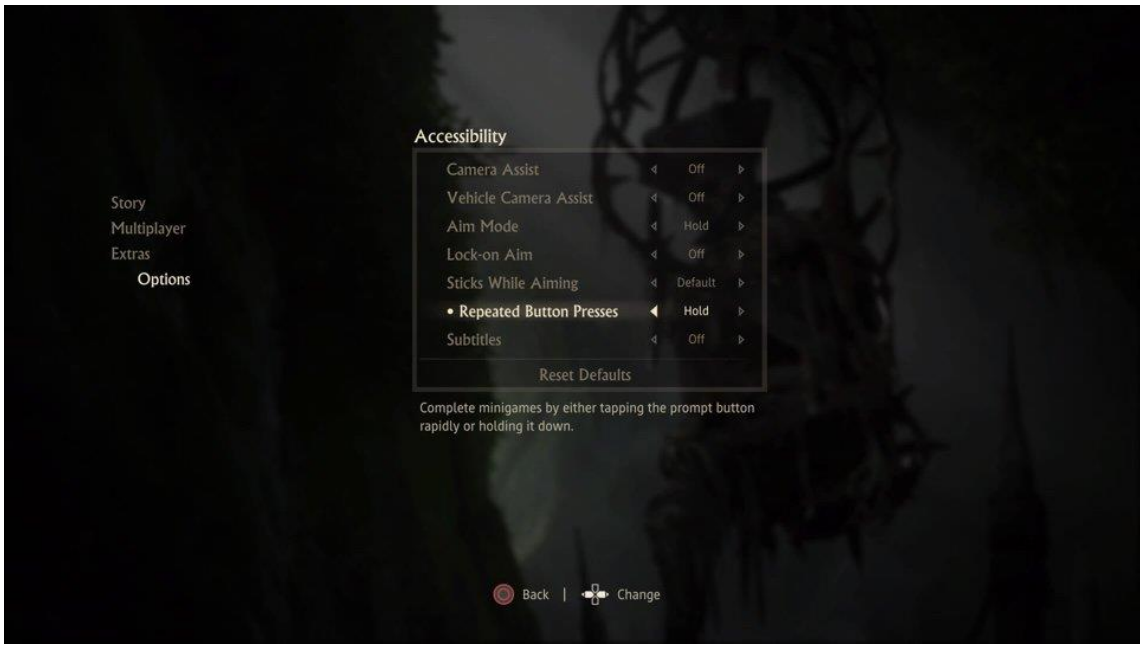
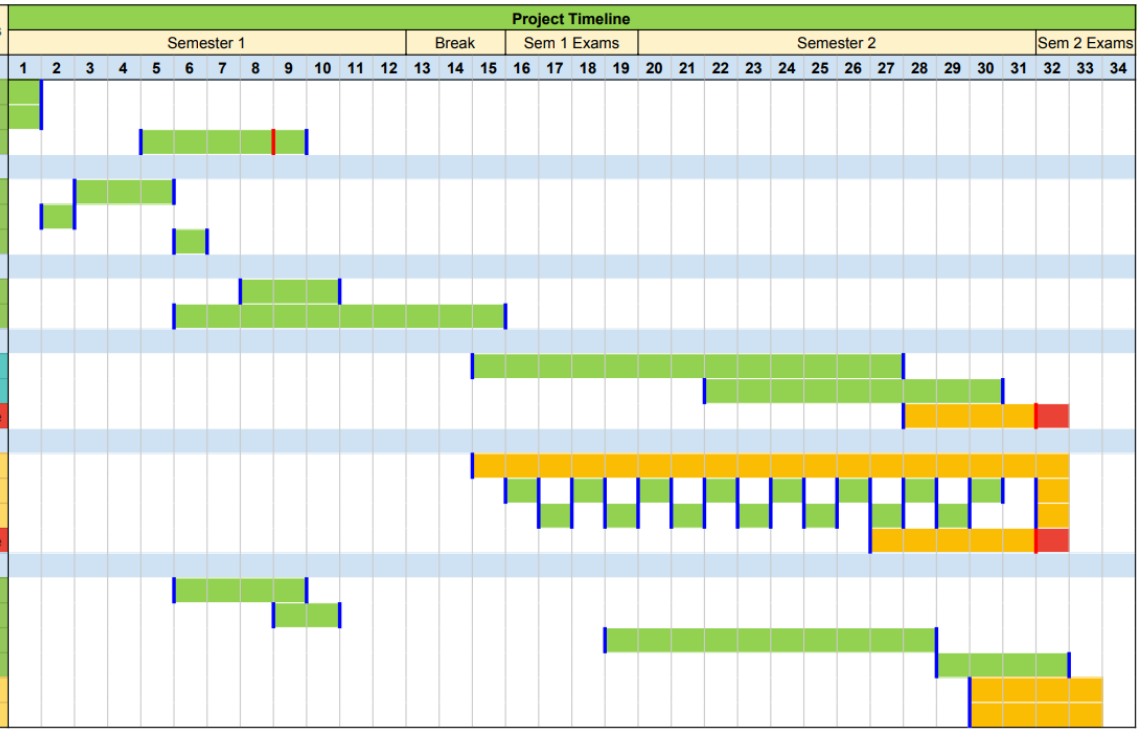


Figure 6. Accessibility Options - Uncharted 4

3.4 Gantt Chart

The following Gantt chart has been created to show the project schedule.

Task	Est. Time in Weeks	Start Date	End Date	Progress
Project Scoping & Research Gathering				
Determine project idea and discuss with supervisor	1	21.09	28.09	Done
Determine the scope of the project	1	21.09	28.09	Done
Collect relevant research articles and journals	5+	26.10	30.11	Done
Identify Requirements & Technologies Used				
Identify current approaches and examples of game accessibility	3	05.10	02.11	Done
Identify the technology used to develop project	1	28.09	05.10	Done
Identify requirements and corresponding consideration	1	02.11	09.11	Done
Design Mockups, Build Prototypes & Learn Skills				
Design mockups	2-3	16.11	07.12	Done
Develop skills where necessary (through prototyping)	10	02.11	11.01	Done
Development				
Basic requirements	9	04.01	05.04	Done*
Intermediate requirements	4	22.02	26.04	Done*
Advanced requirements	4	05.04	-	Overdue
Testing and Evaluation				
Unit testing	15	04.01	-	In Prog.
Integraton testing	8	11.01	-	In Prog.
Regression & sanity testing	7	18.01	-	In Prog.
User acceptance testing	4	29.03	-	Overdue
Deadline deliverables				
Interim report	4	02.11	30.11	Done
Presentation slides	2	23.11	07.12	Done
Draft report	10	01.02	12.04	Done
Final report	4	12.04	04.05	Done
Final presentation slides	4	19.04	-	In Prog.
Project video	4	19.04	-	In Prog.



Done* = Completed to match the amended MVP in section 5.2, though more work can be done to complete all requirements.

3.5 Method of Evaluation

As with any complex project, Spoken Word will be developed following an iterative approach and will be tested along several stages and iterations of the project's lifecycle. Testing to be conducted by myself will go as follows:

- **Unit testing** will be conducted as each requirement is completed. This will evaluate that individual components work as expected as they are implemented into the game.
- **Integration testing** will be conducted as related requirements are fulfilled. For example, Br6, Br7 and Br8 all revolve around the use of speech recognition. Hence, they will be integrated and tested together once all have passed their unit evaluations and have been signed off by myself. This will ensure that game components work as expected when together.
- Finally, a combination of **regression and sanity** testing will be conducted during the development of new requirements, to make sure that at no point during adding a new feature have previous requirement's functionalities been altered.

Most important of all however, **user acceptance testing** will be performed in conjunction with Special Effect, a charity specialising in aiding disabled gamers within the UK. Real users with motor deficiencies will not be testing the game for ethical reasons, so instead, Special Effect staff has been contacted and has agreed to collaborate on an evaluation. This is the most critical part of testing as subject matter experts will provide a critical evaluation on the voice controls for Spoken Word.

- The charity selected for this testing is Special Effect, a UK based charity that helps the physically disabled get started enjoying games as any person would.
- Special Effect was contacted in late March once a playable demo had been developed (reflected in the project Gantt chart), in hopes of receiving constructive criticisms and an evaluation of the demo to date. Soon after an Occupational Therapist at Special Effect agreed to take part in an evaluation.
- A playable build, controls reference sheet and questionnaire form were sent to Special Effect. Unfortunately, as of writing this report the evaluation has not been returned however, so instead a self-evaluation and walkthrough has taken place.
- Additionally, a small group of able users was chosen to play test the demo of Spoken Word using both standard controls and voice controls (more details in section 6.2)

3.6 Risk Assessment

Risk	Effect	Probability	Impact	Preventative Measures
Inability to correctly implement accessibility considerations	The results of this would be a game that is not fit to purpose, it would be no more accessible than the average game.	High	Severe	Test thoroughly throughout the development process. Do not take the learning period lightly, the entirety of the development process relies on knowing how to use the Unity and IBM Watson.
Unable to retrieve speech to text data from the IBM Watson SDK	Many of the core requirements that cater Spoken Word to players with motor disabilities would be useless.	Medium	Severe	Read documentation and follow tutorials provided by IBM closely. Spend time prototyping so that I have past examples to refer to if I run into issues during development.
Using deprecated version of the IBM Watson SDK	Could run into compatibility issue with the Unity engine, could cause several errors that would slow down development.	High	Medium	Read official documentation, many versions have been deprecated as is. Stick to version of Unity that are officially confirmed to be compatible with the SDK.
Inability to become proficient enough with Unity.	Potentially very problematic, could lengthen development times past what's described in the Gantt chart.	Medium	Severe	Again, do not take the learning period lightly. Refer to past projects for simple components and functionality that can be easily forgotten.
Poor time management.	This could result in worse grade, unfinished work or compromised quality of work. Already experienced with the research collection and literature review.	High	High	Set time aside every day or every other day to work on the project and project report, evaluate progress and set feasible goals.
Loss of work.	Losing work due to data corruption, etc. Could push back the project timeline and result in compromised project quality.	Low	High	Make proper use of source control (git) so that any computer or human failure does not cause massive setbacks.
Insufficient / incorrectly analysed requirements.	A game that's not fit for its purpose. Could mean in wasted time and effort developing a project that was faulty from the start.	Medium / Low	Severe	Review project requirements with supervisor and ask myself during testing after each commit whether the right requirements are being fulfilled.
Poor testing.	The game is likely to contain making bugs or could possibly not work at all depending on the severity of the bug.	Low	Medium / High	Consistent testing throughout the entire development process. Test prototypes after every commit and after each requirement is fulfilled. Use a variety of testing methods. Conduct end user testing when the game is near completion.

Geographical / physical disasters.	All progress could be lost, potentially weeks of work. Could cause delays in the project timeline.	Low	High	Good use of source control and keeping hardware safe should prevent any setbacks.
Scope too large.	If the scope is too large, the project timeline may be delayed or it may become difficult to complete all requirements affecting the quality of the end result.	High	High	Critically evaluate what the exact focus of the project is and what is necessary to complete it. Cut out any unnecessary requirements.
Unable to meet the MVPs requirements.	If the MVP cannot be met by the project's due date, it is possible that the end result may not be fit for purpose and or the evaluation could suffer as a result.	High	High	Same as above, critically evaluate the MVP and make sure it is in line with the focus of Spoken Word. Amend the MVP if necessary.
Unable to conduct an evaluation in time.	As the main evaluation will be conducted by a third party, it is possible that it does not finish in time for when the project is due.	Medium	High	Contact the third party well in advance and prepare an evaluation copy of the project for them to evaluate, set deadlines with the third party.
Unable to implement accessibility considerations beyond motor considerations.	This means that Spoken Word would only be fit for able users and users with motor deficiencies.	High	Low	Keep list of requirements realistic; don't try to implement every consideration found in modern games.

Chapter 4: **Presentation**

This section will outline the steps required to set up the development environment. As well as the high-level design and implementation of Spoken Word with references to source code snippets.

4.1 Development Environment

4.1.1 Set-Up

1. Create an IBM Cloud account if you wish to create your own assistant and speech to text services.
2. Create a speech to text resource and assistant resource and take note of each of their API keys and URLs as well as the assistant's workspace ID.
3. Install Unity Hub and Unity 2019.4.13f1 or later.
4. Make sure to install Visual Studio 2019 or later when going through the Unity Hub installation process.
5. Clone the Spoken Word GitHub repository found at: https://github.com/sendelivery/Spoken-Word
6. Clone the IBM Unity SDK and SDK core from the respective repositories:
7. https://github.com/IBM/unity-sdk-core, https://github.com/watson-developer-cloud/unity-sdk
8. Inside Unity Hub, click "Add" and navigate to the "FYP – Spoken Word" folder cloned from the Spoken Word repo.
9. Open the Spoken Word project from Unity Hub, once the Unity editor is open, set Visual Studio 2019 as the preferred code editor from the project settings.
10. Drag the IBM SDK Core and Unity SDK into the "assets" folder under the "project" tab.
11. Give Unity time to import all the assets, then click play or begin making changes.
12. If you have created your own speech to text and assistant services, copy and paste the API keys, URLs and workspace ID into the respective fields under the 'Watson Services' game object.

4.1.2 Project Structure

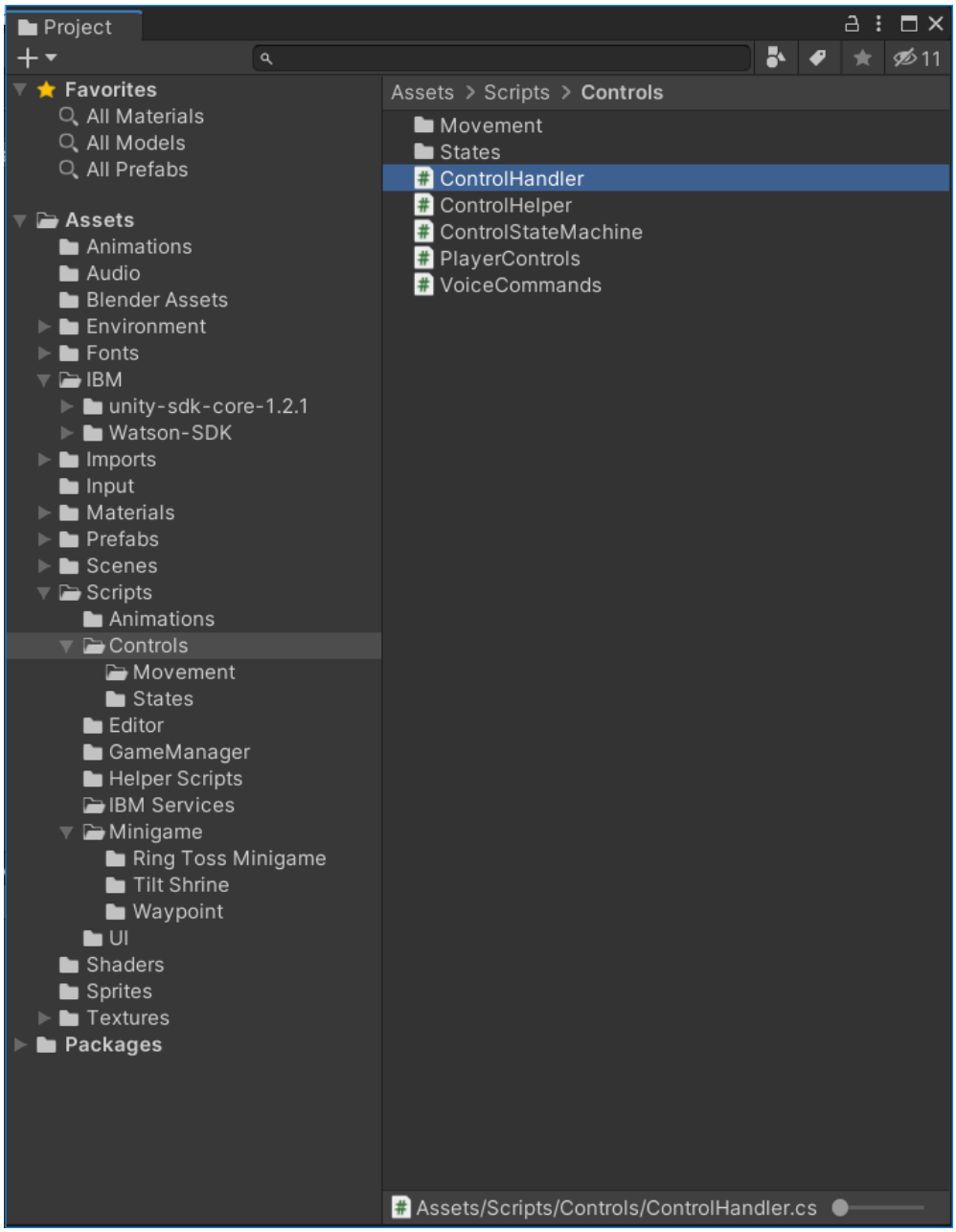


Figure 7, Unity Engine Project Structure – ControlHandler.cs is highlighted as an example.

In Figure 7 we can the project structure of Spoken Word. The “Assets” folder at the top of the project hierarchy is the main folder within a Unity project, it contains all the code, 3D models, textures, etc. that are used by the game. Most APIs and SDKs assume everything is located within the assets folder (Unity Technologies, 2020). All the code written by myself is under the “Scripts” folder. Below the assets folder is the “Packages” folder, this folder contains any imported official Unity packages that provide a wide range of enhancements and extensions to the Unity editor (Unity Technologies, 2020).

A list of the Unity packages used can found in section 4.4.3, though it is not necessary to install them as they come included in the Spoken Word repository.

4.2 Overview of Platform and Technologies

This section will outline the high-level design of Spoken Word, without going into detail about design and architecture of the Unity engine or IBM Watson itself. However, some Unity and IBM Watson concepts will be explained to better understand the code written for Spoken Word.

4.2.1 Unity Engine

4.2.1.1 GameObjects

In short, every object in the game world, regardless of what it is, is a game object. From lights, cameras, items, and the player itself. On their own, game objects cannot do anything, instead they are controlled via C# scripts you write, properties you set, and components you attach to those game objects. All game objects have a transform component attached by default, from which we can manipulate their position, rotation, and scale relative to the world (Unity Technologies, 2020).

4.2.1.2 Scenes

A scene in Unity is where all the development work takes place. Game objects all other content live inside a scene. Each scene can be thought of as a level in the game (Unity Technologies, 2020). Since Spoken Word only uses one environment, everything is in a single scene.

4.2.1.3 Unity Engine Order of Execution

Running any script that inherits from Unity's base class MonoBehaviour must execute several event functions defined by Unity in a predetermined order. Two of the most important events are the Start function and Update function. Start, is a function that is only every called once for any given script and is up to the developer if they want to include into their script or not. Awake is a function like Start, that only happens once in a script's lifetime and even before Start. A scripts lifetime starts when a scene is loaded and ends when a scene is unloaded (Unity Technologies, 2020).

The Update function is called every frame by the engine. Here is where a developer would put the logic that they want executed constantly as the game is running. FixedUpdate is like Update, but happens before any physics related updates. For example, if we have a lightbulb game object that is turned off by default, the Start function

could be used to turn it on, and then the Update function could be used to smoothly cycle through different colours over time (Unity Technologies, 2020).

4.2.2 IBM Watson

IBM Watson is an AI platform developed and managed by IBM. Generally seen as a business solution, it has a wide range of business-ready tools designed for the easy adoption of artificial intelligence in any number of functional areas within an organisation such as financial operation, advertising and even healthcare (IBM, 2021). For Spoken Word, Watson is being used for two main functions. First to convert microphone input recordings into text (speech to text), and second to process that text and extract relevant information that can be translated into in-game actions. To do this, the following two IBM Watson services are being employed.

4.2.2.1 Speech-To-Text

The Watson Speech to Text service (STT) is used to transcribe audio to text, enabling speech transcription capabilities for any application that uses it (IBM, 2021). From the IBM Cloud homepage, a STT resource can be created following the steps outlined in 4.1.1. The service is then connected to within Spoken Word using the API endpoint key and endpoint URL, shown in Figure 25.

4.2.2.2 Watson Assistant

The Watson Assistant service is used primarily for businesses to create branded chatbots or ‘assistants’ and integrate them into any of their own services or business areas. The idea is that customers interact with assistant by sending a message through any of the channels the assistant is integrated in (Facebook Messenger for example). The assistant received this message, sends it to the appropriate resolution path (IBM, 2021).

In Spoken Word, the conversation skill resolution path is being used, which interprets the customer’s message further, gathering any relevant information it can find from the message. Depending on how the developer sets it up, the assistant could then respond to the customer with a follow up message or perform some transaction on their behalf.

The conversation skill, also known as a dialogue skill (Figure 8), is being used in Spoken Word to extract the extra information known as intents and entities. While this information can be used for the assistant to respond with dialogue, instead it is being used for actions that can be performed within the game, more detail in 4.3.

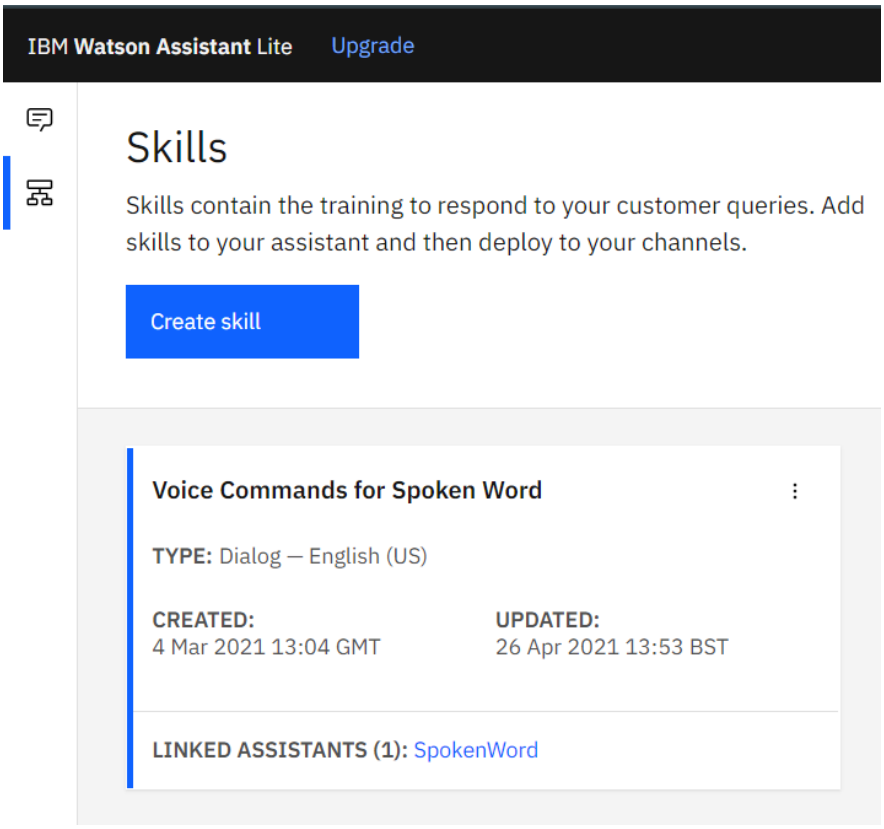


Figure 8, Voice Commands Dialogue Skill

4.2.2.2.1 Intents

An intent is the purpose of a user's message to an assistant. For example, user's intent could be to book a flight, and their message to the assistant with that intent could be along the lines of "Book me a flight to Paris." or "Buy a plane ticket to Paris.". In Spoken Word intents are used to signal specific actions to be performed in game.

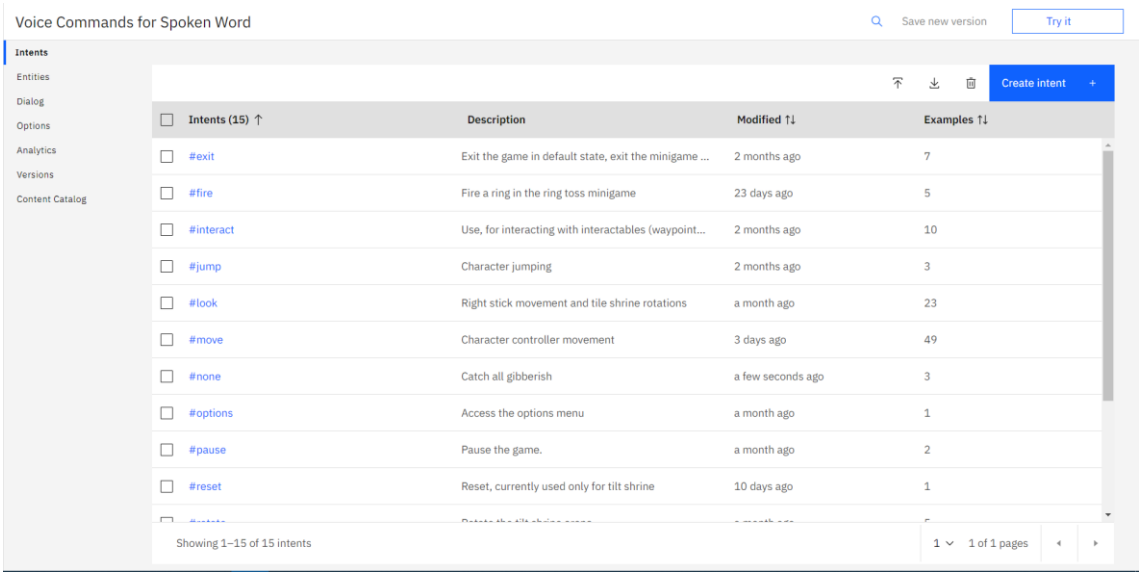


Figure 9, Spoken Word Assistant - List of Intents

4.2.2.2.2 Utterances

Utterances are the messages that a user could potentially send when they have a particular intent in mind. Inside the assistant service, utterances, or examples, are given to each intent to train the assistant model. This way, when a message comes in that looks similar, but is not the same the assistant will still be able to understand the intent behind it.

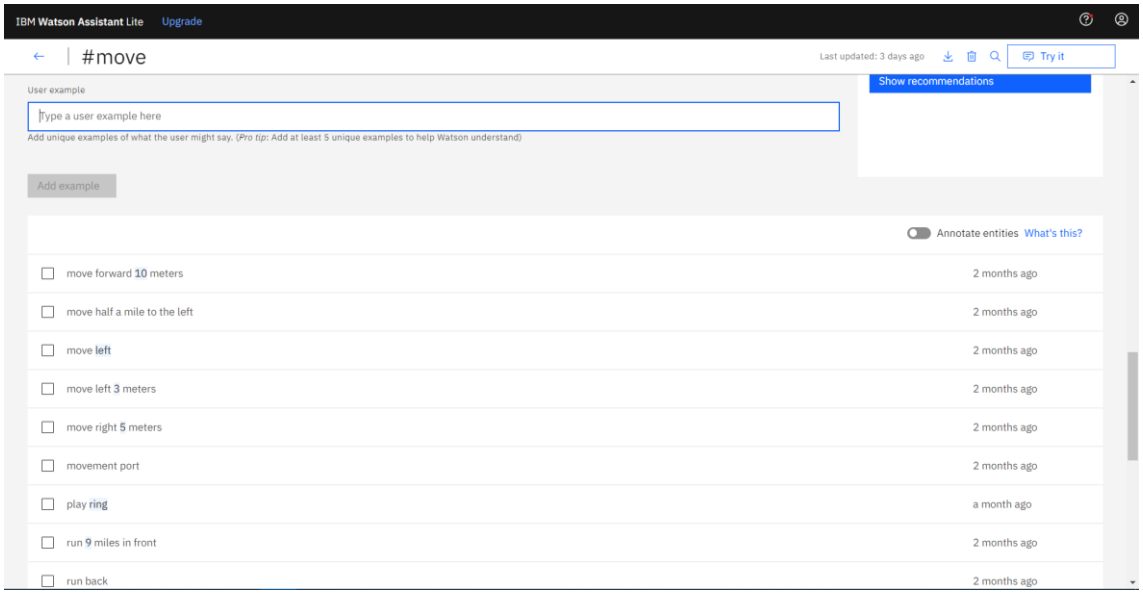


Figure 10, Spoken Word Assistant - Examples for 'move' Intent.

4.2.2.2.3 Entities

Entities are pieces of information that can be extracted from an utterance and give us more information about how the user wants a particular intent to be carried out. Using the same example as before, in the utterance “Book me a flight to Paris.” the location “Paris” could be extracted and used by the booking system. In Spoken Word, entities are used in many places for similar reasons. For example, the direction entity (Figure 11) which is used to specify direction in intents like ‘move’ or ‘look’ where the player will want to do either of those actions with a direction having been specified.

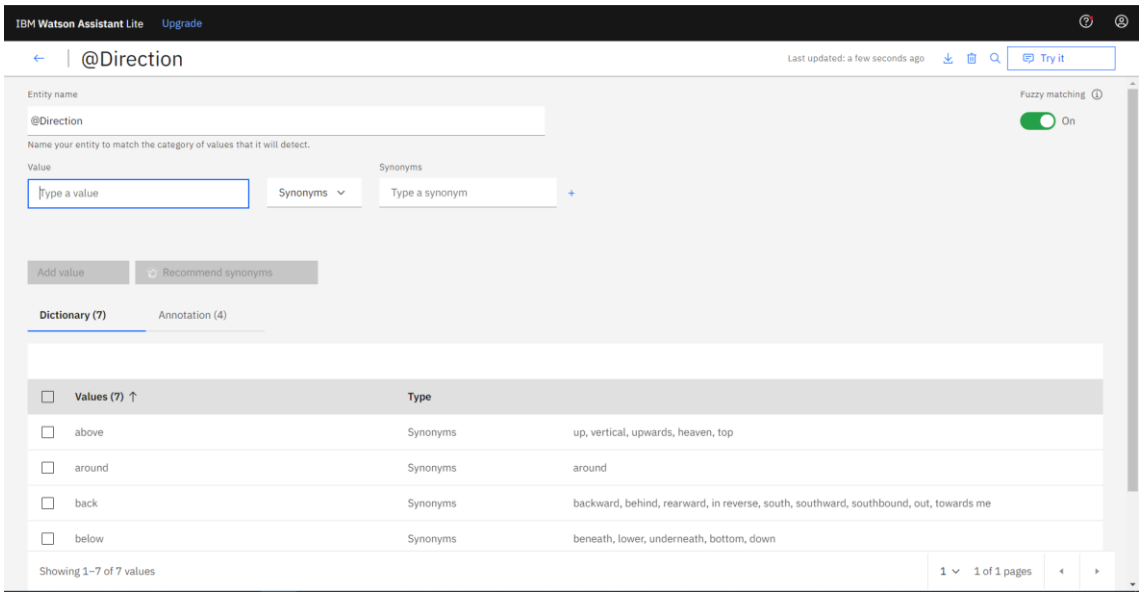


Figure 11, Spoken Word Assistant - Direction Entities.

4.3 Features Implemented

4.3.1 Handling User Input

4.3.1.1 Overview

Correctly handling user input is an important feature of Spoken Word because of the requirement of supporting not only keyboard / mouse (hereafter, *KM*) and game controller (hereafter, gamepad) input, but also supporting input via a microphone (Br1. through Br6.). For this reason, many systems had to be written twice, once for KM and gamepad input, and the second time for microphone input. Figure 12 shows a simplified class diagram of the overall control system.

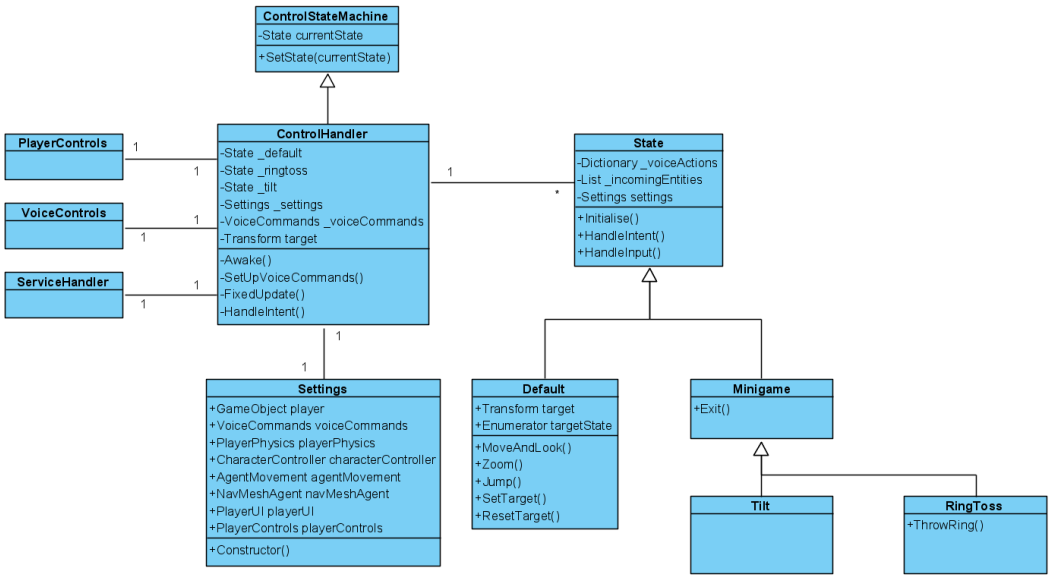


Figure 12, Control System Class Diagram

The control handler, which is attached to the 'Player' game object uses the state machine pattern to set the current control state. Upon starting the game, the player is immediately assigned the _default control state in ControlHandler's Awake function. Then, depending on which minigame the user decides to interact with, the player game object will be assigned either the _ringtoss or _tilt control state. This solution is highly extensible as we'd only need to add a new State class and action map (more detail below) to create a new set of controls schemes for a new minigame. Hence, adhering to the open / close software engineering principle.

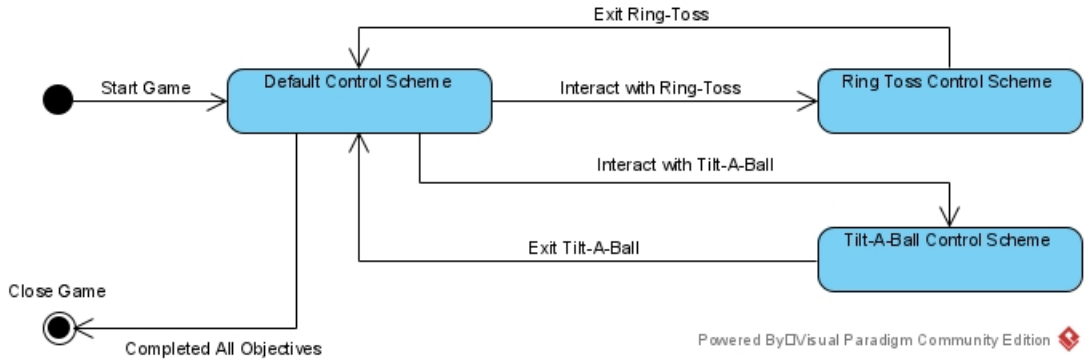


Figure 13, Control System State Diagram

Each state has a reference to a number of useful game objects, components, scripts and variables through the _settings attribute. Most important is the VoiceCommands script that is used alongside each state's _voiceActions dictionary to assign functions within VoiceCommands to intents that are unique to that state.

Finally, on the left-hand side of Figure 12 are the player controls, voice controls, and service handler classes. PlayerControls is a C# script generated by Unity's Input System package (Figure 14). Using this package, KM and gamepad input could be assigned to *Actions* and Actions assigned to *Action Maps.*

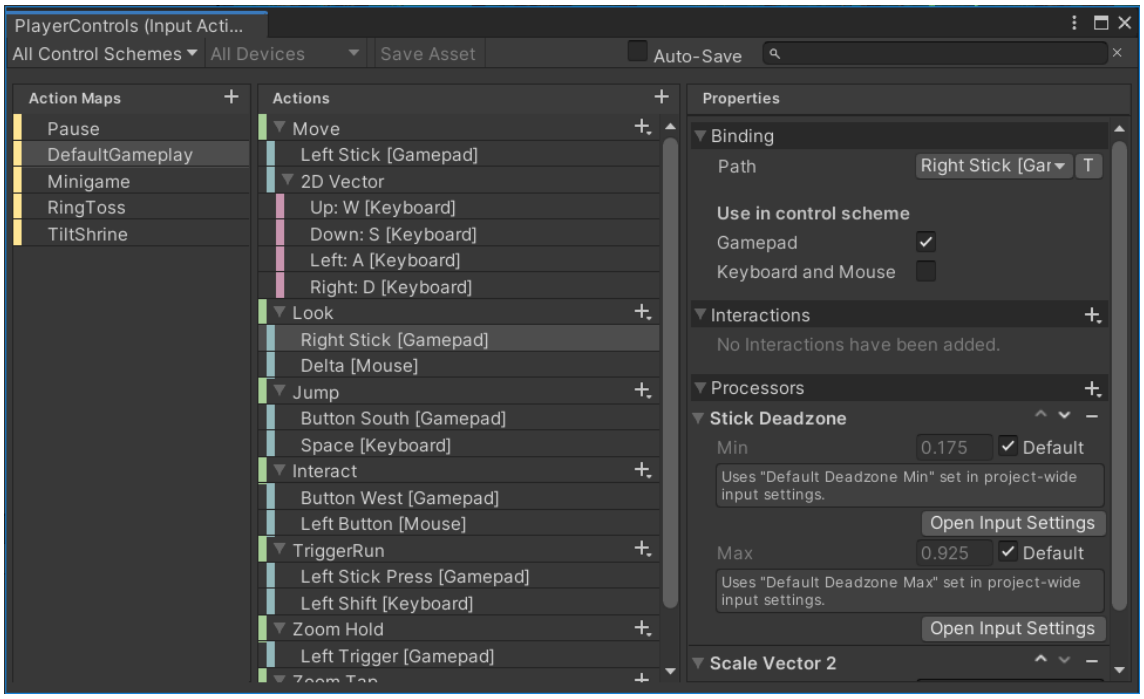


Figure 14, Input System - Action Maps, Actions and Properties

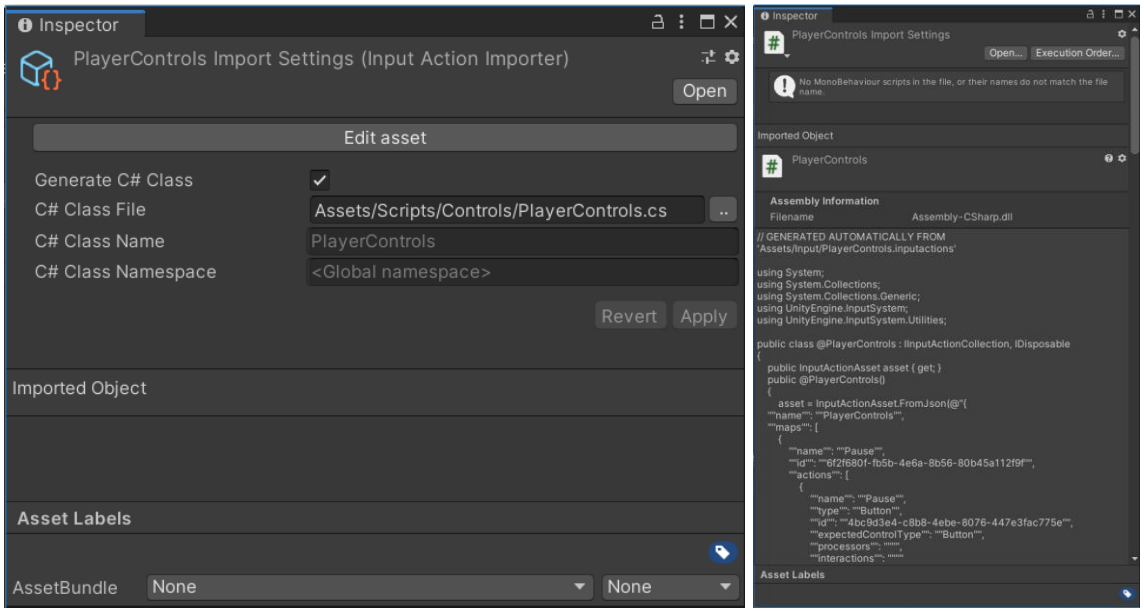


Figure 15, Input System - Generated C# Class - Player Controls

Once the code was generated, actions could be tied to functions as seen in Figure 18. For voice actions, as mentioned previously, IBM Watson and the IBM Unity SDK was used. The VoiceControls script was then used to handle actual in-game actions (more information in section 4.3.2).

4.3.1.2 Code

```
private void Awake()
{
    PlayerControls controls = new PlayerControls();

    // Start and Select (Pause and Options)
    controls.Pause.Enable();
    controls.Pause.Pause.performed += ctx => Pause(controls);
    controls.Pause.Options.performed += ctx => Options(controls);

    SetUpVoiceCommands();

    // _settings contains references to any objects or components needed by each state
    _settings = new Settings(this.gameObject, ref voiceCommands, playerPhysics, ref characterController,
        agentMovement, ref navMeshAgent, runSpeed, sensitivity, Camera.main, controls, playerUI);

    mouseLook = _settings.cam.GetComponent<MouseLook>();

    if (_default == null || _ringtoss == null || _tilt == null)
    {
        _default = new Default(ref _settings);
        _ringtoss = new RingToss(ref _settings);
        _tilt = new TiltShrine(ref _settings);
    }

    SetState(_default);
}
```

Figure 16, ControlHandler.cs – Awake() function.

```
void FixedUpdate()
{
	Debug.Log(state); // Log our current state.

	state.HandleInput();
	CheckTarget();
}
```

Figure 17, ControlHandler.cs - FixedUpdate() function.

```
public override void Initialise()
{
    base.Initialise();

    VoiceCommands voiceCommands = settings.voiceCommands;

    // Set voice commands actions
    _voiceActions.Add("move", () => voiceCommands.Move(incomingEntities, inputText));
    _voiceActions.Add("look", () => voiceCommands.Look(incomingEntities, inputText));
    _voiceActions.Add("zoom", () => VoiceZoom(incomingEntities));
    _voiceActions.Add("jump", () => Jump());
    _voiceActions.Add("interact", () => Interact());

    // Movement
    settings.playerControls.DefaultGameplay.Move.performed += ctx => move = ctx.ReadValue<Vector2>();
    settings.playerControls.DefaultGameplay.Move.canceled += _ => move = Vector2.zero;
    settings.playerControls.DefaultGameplay.TriggerRun.performed += _ => run = true;
    settings.playerControls.DefaultGameplay.TriggerRun.canceled += _ => run = false;
    speed = settings.runSpeed;

    // Look & Zoom
    settings.playerControls.DefaultGameplay.Look.performed += ctx => look = ctx.ReadValue<Vector2>();
    settings.playerControls.DefaultGameplay.Look.canceled += _ => look = Vector2.zero;
    settings.playerControls.DefaultGameplay.ZoomHold.performed += _ => zoom = true;
    settings.playerControls.DefaultGameplay.ZoomHold.canceled += _ => zoom = false;
```

```
    settings.playerControls.DefaultGameplay.ZoomTap.performed += _ => zoom = !zoom;
    zoom = false;

    // Jump
    settings.playerControls.DefaultGameplay.Jump.performed += _ => Jump();

    // Interaction
    settings.playerControls.DefaultGameplay.Interact.performed += _ => Interact();

    // Disable the navMeshAgent component and enable the default gameplay action map
    settings.navMeshAgent.enabled = false;

    // Enable the default action map
    Enable();
}
```

Figure 18, Default.cs - State initialisation.

In Figure 16 and Figure 17 the control handler is initialising in the Awake() function. The action maps for pausing the game and opening the options menu are enabled as they are not tied to any states. Next, the voice commands and the settings member are set up. Finally, the states are initialised, and the default state is set.

The control handler then calls the state.HandleInput function to correctly handle KM or controller input. This happens from the FixedUpdate function (similar to Update but happens before physics checks) so that the HandleInput call happens every frame. Afterwards, we check if a target has been set via voice controls for pathfinding so that users who are using voice controls are also able to move around.

Within the default state's initialisation function in Figure 18, voice commands are defined, and functions are tied to the corresponding actions in the default action map before enabling the action map. This initialisation is only called the first time that a state is set.

4.3.2 Voice Commands and IBM Services

4.3.2.1 Voice Commands

The VoiceCommands script is a special class in the context of the Unity and the rest of the scripts in Spoken Word. It is not attached to any game objects but is instead instantiated at runtime and then attached to a game object by the control handler (Figure 16 line 10). What this function call (Figure 19) does is create a game object called "Voice Commands" and attach the VoiceCommand script to it. The reason being that special functions, unique to Unity called coroutines must be executed from a script that is attached to a game object. Coroutines, allow functions to be executed over several frames, in contrast to ordinary functions that would normally execute entirely in a single frame update (Unity Technologies, 2020).

This means that actions in-game like movement that occurs over the period of time that the user is providing an input, appear to function similarly when using voice commands. An example is shown below in Figure 20 where the look intent has been identified, and a different coroutine is called depending on the direction extracted from the utterance's entities. Figure 21 shows the coroutine itself. Execution stops at line 10, returning to the same place on the following frame to continue the for loop.

```
// Need to attach voice commands to an object because
// coroutines can only be called if attached to an object,
private void SetUpVoiceCommands()
{
    GameObject obj = new GameObject("Voice Commands");
    obj.AddComponent<VoiceCommands>();
    voiceCommands = obj.GetComponent<VoiceCommands>();
    voiceCommands.player = this.gameObject;
    // Only reliable way of getting the camera in awake, GameManger.cs does not
have it yet.
    voiceCommands.cam = Camera.main;
}
```

Figure 19, ControlHandler.cs - SetUpVoiceCommands function.

```
for(int i = 0; i < direction.Count; i++)
{
    // break out of the for loop if i is 2, basically don't look in 3 directions
at once
    if (i == 2) break;

    switch (direction[i])
    {
        case "above":
            Debug.Log("Up, Look " + direction + " " + angle);
            StartCoroutine(LookOnX(new Vector3(-1, 0, 0), duration, angle));
            break;
        case "below":
            Debug.Log("Down, Look " + direction + " " + angle);
            StartCoroutine(LookOnX(new Vector3(1, 0, 0), duration, angle));
            break;
        case "right":
            Debug.Log("Right, Look " + direction + " " + angle);
            StartCoroutine(LookOnY(new Vector3(0, 1, 0), duration, angle));
            break;
        case "left":
            Debug.Log("Left, Look " + direction + " " + angle);
            StartCoroutine(LookOnY(new Vector3(0, -1, 0), duration, angle));
            break;
        case "around":
            Debug.Log("Around, Look " + direction + " " + 180);
            StartCoroutine(LookOnY(new Vector3(0, -1, 0), duration, 180f));
            break;
    }
}
```

Figure 20, VoiceCommands.cs - Snippet of the Look method.

```
private IEnumerator LookOnY(Vector3 direction, float duration, float angle)
{
    Quaternion fromAngle = player.transform.rotation;
    Quaternion toAngle = Quaternion.Euler(player.transform.eulerAngles +
(direction * angle));

    for (var t = 0f; t < 1; t += Time.deltaTime / duration)
    {
        Debug.Log("rotating");
        player.transform.rotation = Quaternion.Lerp(fromAngle, toAngle, t);
        yield return null;
    }
}
```

Figure 21, VoiceCommands.cs - LookOnY method, similar method for X axis exists.

Furthermore, the voice commands script follows a singleton design pattern to ensure that only one game object with the VoiceCommands script exists at any point in time.

```
public static VoiceCommands Instance { get { return _instance; } }

private void Awake()
{
    if (_instance != null && _instance != this)
    {
        Destroy(this.gameObject);
    }
    else
    {
        _instance = this;

        // We use the target for pathfinding so get a reference to it here.
        // Not the most efficient way, but only done once so overhead is not bad.
        target = GameObject.FindGameObjectWithTag("Target").transform;
    }
}
```

Figure 22, VoiceCommands.cs - Singleton pattern.

Decorator-like functions that wrap these coroutines are then assigned to strings that correspond to intents in each control state's _voiceActions dictionary (Figure 18).

4.3.2.2 IBM Watson Service Handler

The ServiceHandler script is the final piece of the puzzle for voice controls. The code in this script is slightly edited code taken from the IBM VR Speech Sandbox samples on GitHub as there was no need to create a solution from scratch. Like the VoiceCommands script it is attached to a game object to make use of coroutines, however it is not instantiated at runtime. Instead, it exists from the start of the scene until the end of the game.

```
void Start()
{
    player = GameManager.player;

    LogSystem.InstallDefaultReactors();
    Runnable.Run(CreateServices());
}
```

Figure 23, ServiceHandler.cs - We create the text-to-speech and assistant service in Start().

```
private IEnumerator CreateServices()
{
    // Creating STT Service ------------------------------------------------------------------------------------------------------
    if (string.IsNullOrEmpty(_speechToTextIamApikey))
    {
        throw new IBMException("Please provide IAM ApiKey for the STT service.");
    }

    IamAuthenticator sttAuthenticator = new IamAuthenticator(apikey: _speechToTextIamApikey);

    //  Wait for token data
    while (!sttAuthenticator.CanAuthenticate())
        yield return null;

    _speechToTextService = new SpeechToTextService(sttAuthenticator);
    if (!string.IsNullOrEmpty(_speechToTextServiceUrl))
    {
        _speechToTextService.SetServiceUrl(_speechToTextServiceUrl);
    }
    _speechToTextService.StreamMultipart = true;

    Active = true;

    // Creating Assistant Service ------------------------------------------------------------------------------------------------
    IamAuthenticator assistantAuthenticator = new IamAuthenticator(apikey: "<API KEY>");

    //  Wait for tokendata
```

```
    while (!assistantAuthenticator.CanAuthenticate())
        yield return null;

    _assistantService = new AssistantService("2019-02-18", assistantAuthenticator);
    _assistantService.SetServiceUrl("<SERVICE URL>");

    // ----------------------------------------------------------------------------------------------------------------------

    StartRecording();
}
```

Figure 24, ServiceHandler.cs - The TTS service and assistant service are being created here.

As most of the logic behind the code in Figure 24 is specific to the IBM Watson Unity SDK, it will not be explained in detail. What is important to know is that the text-to-speech service and assistant service outlined in 4.2.2 is being connected to via its API endpoints in the ServiceHandler script (Figure 25).

Additionally, we are also accessing the user's microphone if they have one available on their computer. In the event a valid microphone is not found, an error will be thrown, but the game will continue to function without voice control functionality.

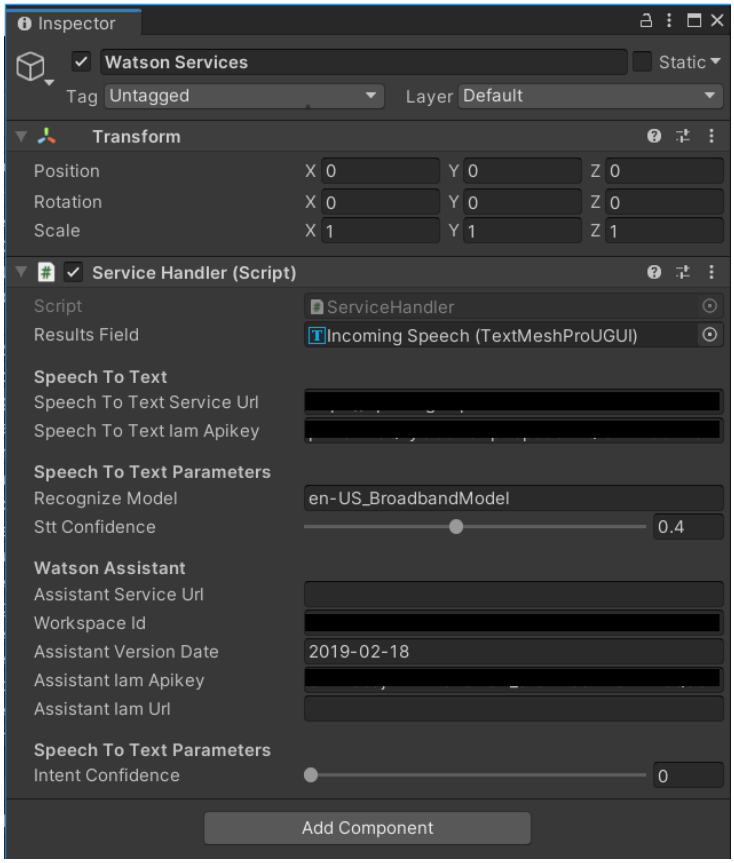


Figure 25, Watson Services GameObject – API keys and URLs are blacked out.

```
// Send message if stt confidence is above a certain threshold
if (res.final && alt.confidence >= _sttConfidence)
{
    string utterance = alt.transcript;
    Debug.Log("Sending message: " + utterance + " with confidence: " +
alt.confidence);

    MessageInput input = new MessageInput()
    {
        Text = utterance
    };

    _assistantService.Message(OnMessage, _workspaceId, input);
    GameManager.TakeSnapshot();

}
```

Figure 26, Snippet of code from the ServiceHandler.cs

In Figure 26, the confidence level of the output received from the speech to text service is being checked against the set threshold. If it meets it, that text is then being send to the assistant service, to decipher intent and entities. Finally in Figure 27, if any intents are recognized by the assistant service, those are extracted along any entities and sent to the control handler's HandleIntent() function. Where the extracted intent's corresponding voice command is triggered from the current state's _voiceActions dictionary outlined in 4.3.1.

```
if (resp != null && resp.Result.Intents.Count != 0)
{
    Debug.Log("Number of Received Intents: " + resp.Result.Intents.Count);

    for (int i = 0; i < resp.Result.Intents.Count; i++)
       {
        Debug.Log("Received Intent: " + resp.Result.Intents[i].Intent);
        if (resp.Result.Entities.Count > 0)
              {
            for (int j = 0; j < resp.Result.Entities.Count; j++)
                     {
                Debug.Log("Received Entities: " +
resp.Result.Entities[j].Entity);
                Debug.Log("Received Entiites Value: " +
resp.Result.Entities[j].Value);
                    }

        } else
              {
            Debug.Log("No entities received");
              }
    }

    player.GetComponent<Control.ControlHandler>().HandleIntent
        (
            resp.Result.Intents,
            resp.Result.Entities,
            resp.Result.Input.Text,
            _intentConfidence
        );
}
else
{
    Debug.Log("Failed to invoke OnMessage();");
}
```

Figure 27, ServiceHandler.cs, - If statement to send extracted intents and entities to the ControlHandler.cs

4.3.3 Sequence Diagram

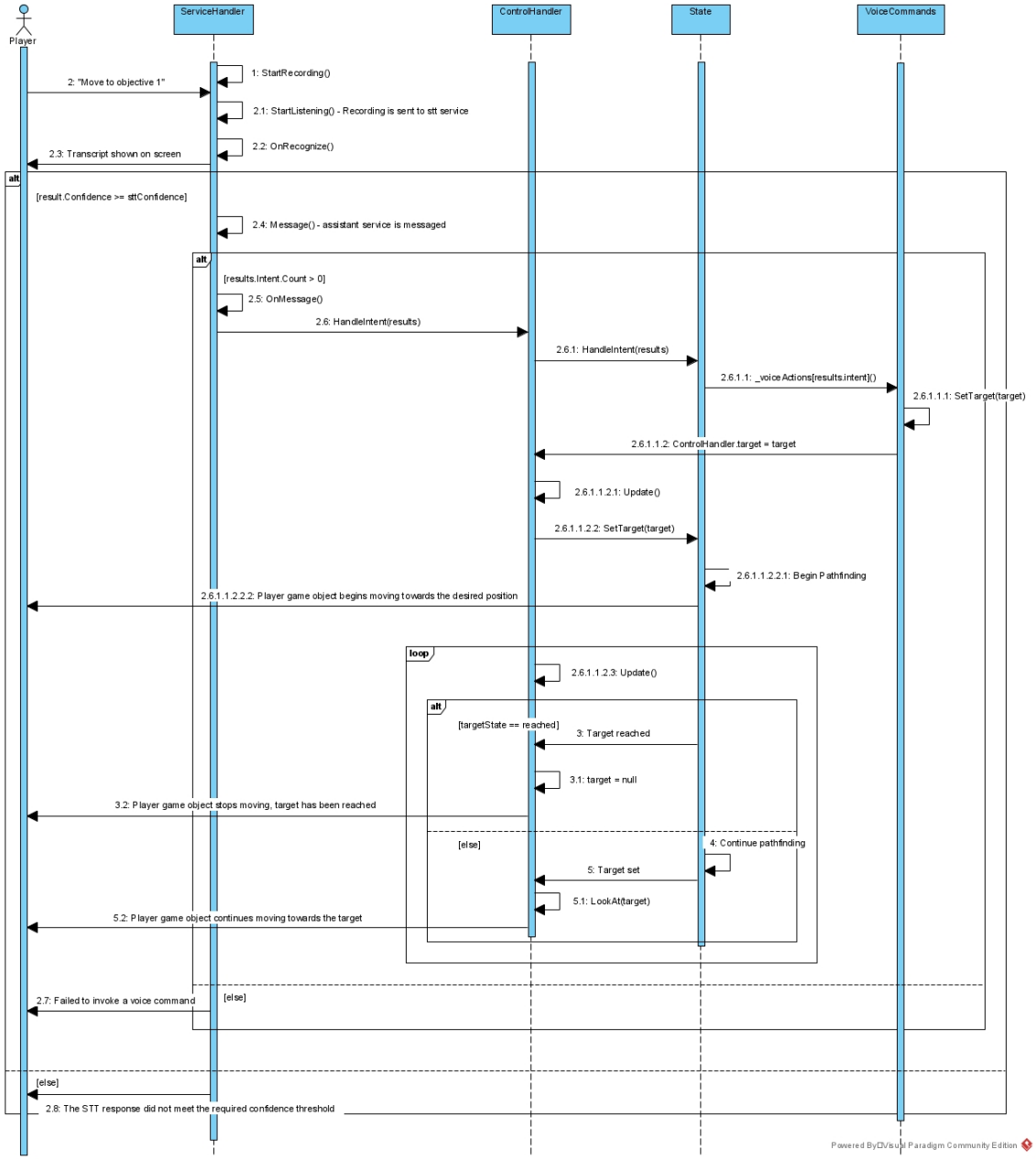


Figure 28, Sequence diagram of a user using voice commands to move their character.

Figure 28 depicts a slightly simplified sequence diagram of how Spoken Word processes a user's voice command. In this case, the action the user wishes to perform is to move their character from where they are, to the first objective. A very similar sequence happens for every single voice action.

4.4 Link to Repositories and Documentation

4.4.1 Spoken Word

Repository: https://github.com/sendelivery/Spoken-Word

Documentation: Submitted as supporting material.

Evaluation Form: https://forms.gle/U5B93m36XxjTQ42G8

4.4.2 IBM SDKs

IBM Unity SDK Core: https://github.com/IBM/unity-sdk-core

IBM Unity SDK for Watson: https://github.com/watson-developer-cloud/unity-sdk

IBM VR Speech Sandbox: https://github.com/IBM/vr-speech-sandbox-cardboard

4.4.3 Unity Engine Packages

Input System:
https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/index.html

Chapter 5: **Requirements & MVP Review**

By the time of evaluation many requirements listed in section Requirements3.1 had still not been met. Additionally, the MVP which consisted of requirements Br1. through to Ir5., had also not been met. While this risk was accounted for collectively in the risks register outlined in section 3.6, the large scope was not considered deeply enough, and the project suffered as a result. Despite this, enough requirements were met in order for Spoken Word to be evaluated for its main purpose of providing alternative controls for users with motor function impairments (shown in the following chapter).

For this reason, the following review of the basic, intermediate, and advanced requirements has taken place, and amendments have been made to the proposed minimum viable product described previously in section 3.2.

- *'X'* = Not Met
- '*P'* = Partially Met
- *'Y'* = Met

5.1 Requirements Review

5.1.1 Basic Requirements Review

Key	Description	Met	Notes
Br1.	A fully functional 3D exploration / puzzle game with a simple narrative.	P	While there is not narrative, Spoken Word is a fully functional 3D game with elements of exploration.
Br2.	Design a simple and intuitive control scheme that requires little learning.	Y	The control scheme follows a standard control layout seen in many modern games. Using the control reference sheet in section 4.4.1, it does not take long to learn.
Br3.	Standard control input using keyboard and mouse.	Y	KM, gamepad, and voice controls are present.
Br4.	At least 1 puzzle to be completed.	Y	2 puzzles are present.
Br5.	Standard control input using a Microsoft Xbox game pad.	Y	Same as Br3.
Br6.	Speech recognition using the Watson Speech Recognition API for alternative character control input.	Y	Also present for all existing puzzles, not only character control.
Br7.	Speech recognition that allows for intuitive navigation of dummy menus.	X	Dummy menus do not currently exist in Spoken Word, the priority for this requirement was set to low previously so can be left for future development effort.
Br8.	Game mechanics in the core gameplay loop that are designed entirely around and rely on speech recognition.	P	While all gameplay supports voice control, there are not any gameplay aspects within Spoken Word that are specifically designed with voice control in mind. Instead, most of the development effort was spent in making standard games with no particular focus for accessibility impairment, and then developing the control system to enable gameplay through voice control.
Br9.	Ensure all information essential to the gameplay and narrative are conveyed clearly and in multiple forms.	X	This aligns with requirements Br1., Br10., and Ar2. Development did not progress to a stage where this requirement reached the top of the list for features to implement so it was unfortunately left out despite originally high priority.
Br10.	Subtitles and captions options are present for all audio tracks that provide information to the user.	X	Aside from the background audio track there is no other audio in the Spoken Word. For this reason, this requirement was not met and will be left for future development efforts.
Br11.	The game must greet players with all accessibility options upon launching the game.	X	Other than game speed in the pause menu, there are no other user tweakable accessibility settings. For this reason, this requirement was not met and is left for future development.
Br12.	Offer a list of included accessibility features upon launching the game.	P	Partially met through the control reference sheet in section 4.4.1.

Br13.	Ability to adjust look sensitivity.	Y	Present in the options menu.
Br14.	Present tutorials for all gameplay elements and sequences.	P	Partially met through the Spoken Word Reference Sheet in section 4.4.1.
Br15.	Clear, high contrast and adjustable size font for all information conveyed via text.	P	All text is in a large readable font and against a background that ensure high contrast. However, there is no functionality to adjust text.
Br16.	Avoid flickering images, quick movements and unexpected events happening on screen.	Y	Spoken Word is a calm game without any flickering images, quick movements, or unexpected events on screen.

5.1.2 Intermediate Requirements Review

Key	Description	Met	Notes
Ir1.	Allow controls to be remapped to the user's liking.	P	Partially met, controls can be remapped very easily by the developer, but the option to do so has not been exposed to the user at runtime.
Ir2.	Allow the game to be played in either windowed or full screen mode. Resolution must be able to be adjusted as well.	P	Spoken Word can be played in full screen or windowed mode. Only the developer can change the display resolution however.
Ir3.	Offer alternatives to inputs unique to certain puzzles or gameplay sequences.	P	This requirement is met by being able to use a gamepad, KM and voice input to control the game however the player is most comfortable. No settings must be adjusted whatsoever; they work interchangeably at the user's command.
Ir4.	At least 3 puzzles to be completed.	X	Currently there are 2 minigames in Spoken Word, a third was planned, but there were other requirements to meet before implementing a third so this requirement can be left for future work.
Ir5.	Provide volume controls for all audio tracks. E.g., effects, music, and dialogue.	X	Like Br10., the only audio track present in Spoken Word is the background audio track so the requirement was not a priority. This can be left for future work when more audio tracks are added to Spoken Word.

5.1.3 Advanced Requirements Review

Key	Description	Met	Notes
Ar1.	Provide the ability to adjust post processing effects like contrast and saturation.	X	This requirement does not meet the needs of users with motor function disabilities specifically, so it was given a low priority. For this reason, Ar1. was not met.
Ar2.	Screen reading for any on-screen text in the game and menus.	X	Same as above.

Ar3.	The game should have a complete, concise narrative and game loop with 3 to 5 clear objectives. Should not feel like a demo or trial.	X	While the requirement of making Spoken Word not feel like a demo or trial was important. The scope of this requirement was too large for the development timeline; hence it was not met.
Ar4.	Adjustable graphics options for low-end computers.	X	These settings are easily adjustable to the developer, however in addition to the low priority, the effort required to expose these options to users would have extended development times. Hence, this requirement was not met.

5.2 MVP Review

Previously, the MVP version of Spoken Word proposed the minimum requirements to be all requirements from Br1. through to Ir5. However, due to how development progressed and how many of the requirements do not specifically address motor function disabilities the MVP has been amended to the following:

Key	Met
Br1.	P
Br2.	Y
Br3.	Y
Br4.	Y
Br5.	Y
Br6.	Y
Br7.	X
Br8.	P
Ir1.	P
Ir2.	P

These requirements adhere specifically to the focus of voice controls being explored as an accessibility option for users with motor deficiencies. Other requirements would ideally be implemented in a final version of Spoken Word, however. As meeting all requirements would elevate the game from being a demonstration of voice controls for video games to more of a completed product.

Chapter 6: **Evaluation**

This section will evaluate Spoken Word to measure the success of the project, its limitations, threats to validity, and propose future work to improve Spoken Word. Unfortunately, as of writing this report Special Effect has not completed their evaluation of Spoken Word, for that reason a software walkthrough, self-evaluation and user evaluation has taken place instead.

6.1 Self-Evaluation – Spoken Word Walkthrough

6.1.1 KM / Gamepad Walkthrough

Upon launching the game, the first thing the user is presented with is a set of instructions on how to complete the game. The screen will then fade in and control will be given to the player, from here they can choose which objective to complete first (Figure 30Figure 30, Main game world.).

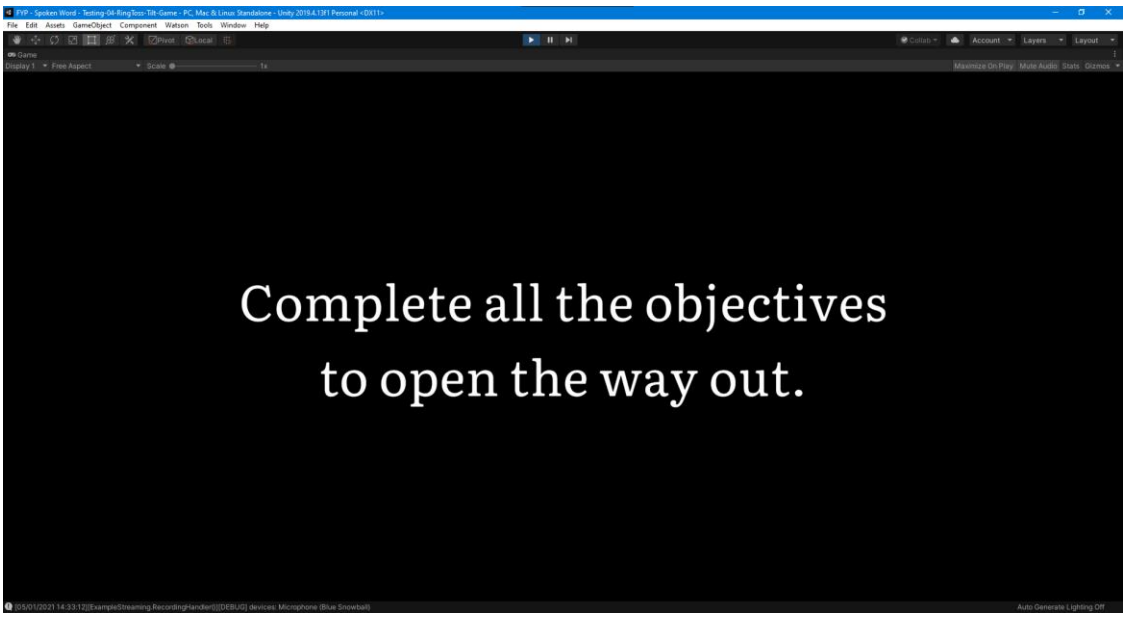


Figure 29, Introductory screen.

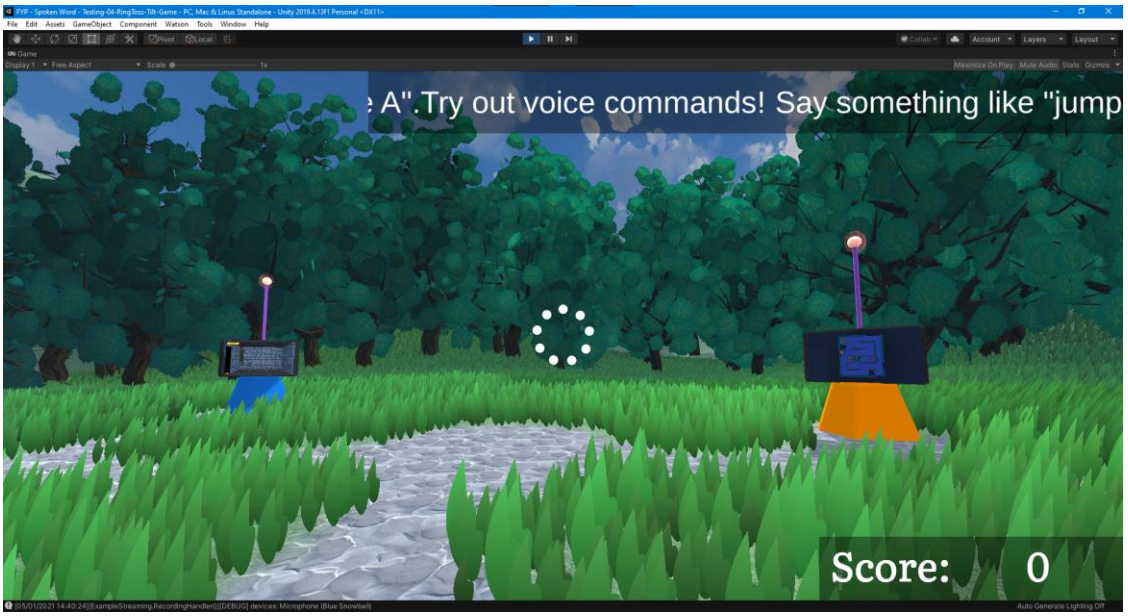


Figure 30, Main game world.

Using either the KM controls or gamepad controls, players can move, jump, and look around the world. Interacting with one of the objectives will move the camera in closer, the control scheme will then switch to that of the objective they chose to play (Figure 31). The player could leave any objective halfway complete if they chose and move on to the next one. However, both objectives must be fully completed before the game can end. Once both objectives have been completed, a small cinematic will play showing the player that the exit has opened (Figure 32).

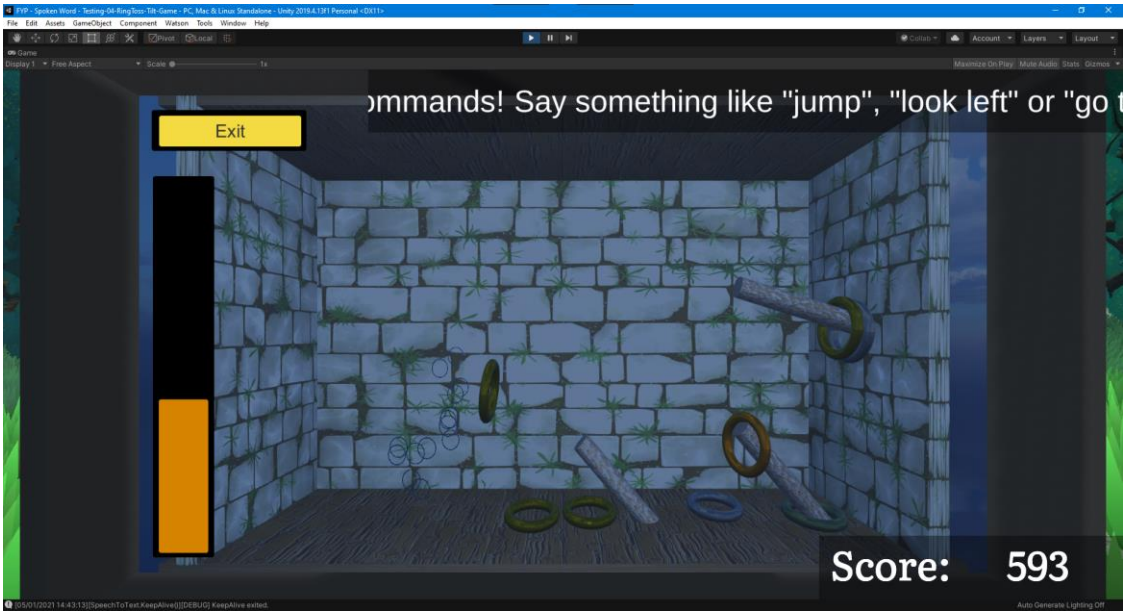


Figure 31, Objective 1 - Ring Toss minigame.

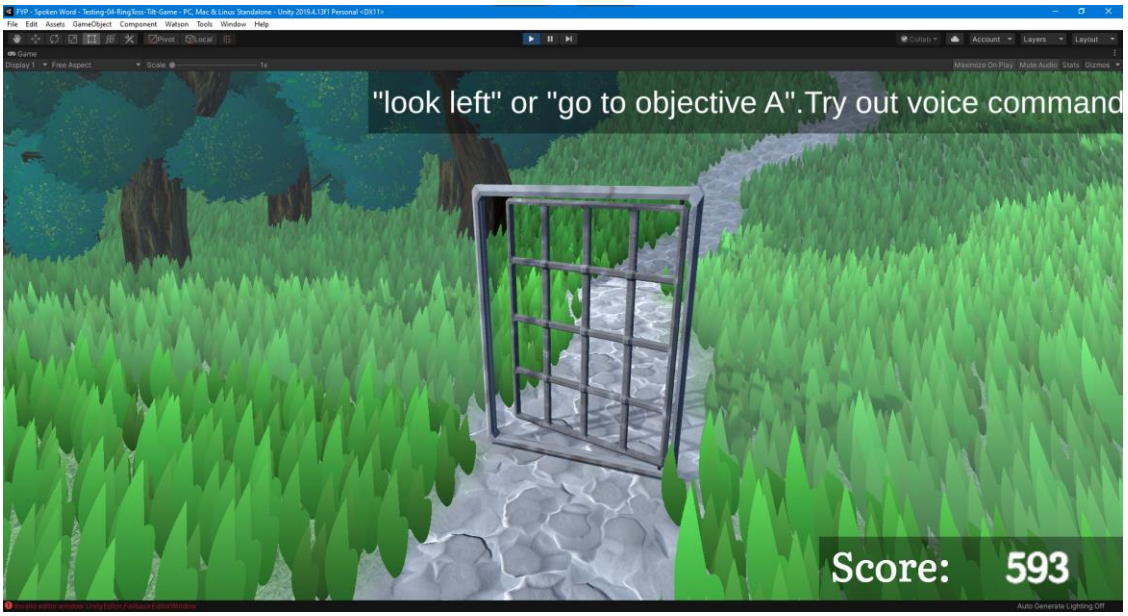


Figure 32, Exit opening cinematic.

The player can then choose to exit the game if they wish by walking near the exit (Figure 33). At any point during gameplay the player can pause the game or open the options screen. Opening the pause screen (Figure 34) exposes the option to slow down the game speed, meaning all in game events happen slower, including movement, and minigame events. This is particularly useful for time-based games like in the Ring-Toss minigame in Figure 31. Opening the options screen exposes the option to change the look sensitivity (Figure 35), satisfying requirement Br13.

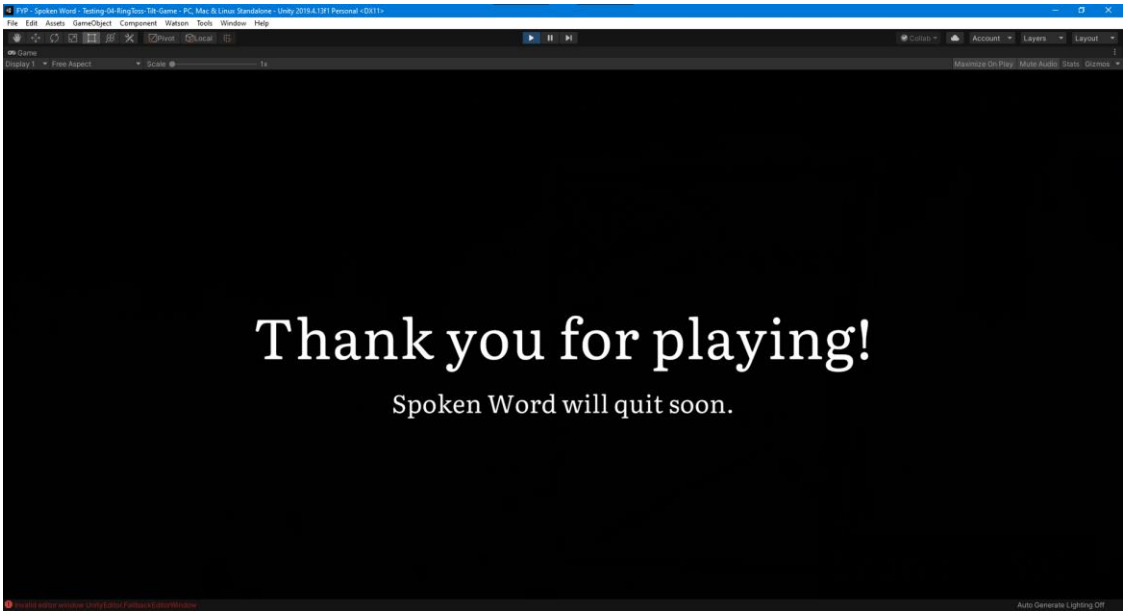


Figure 33, End screen - Spoken Word closes shortly after.

Figure 34, Pause menu - Adjust game speed in bottom left.

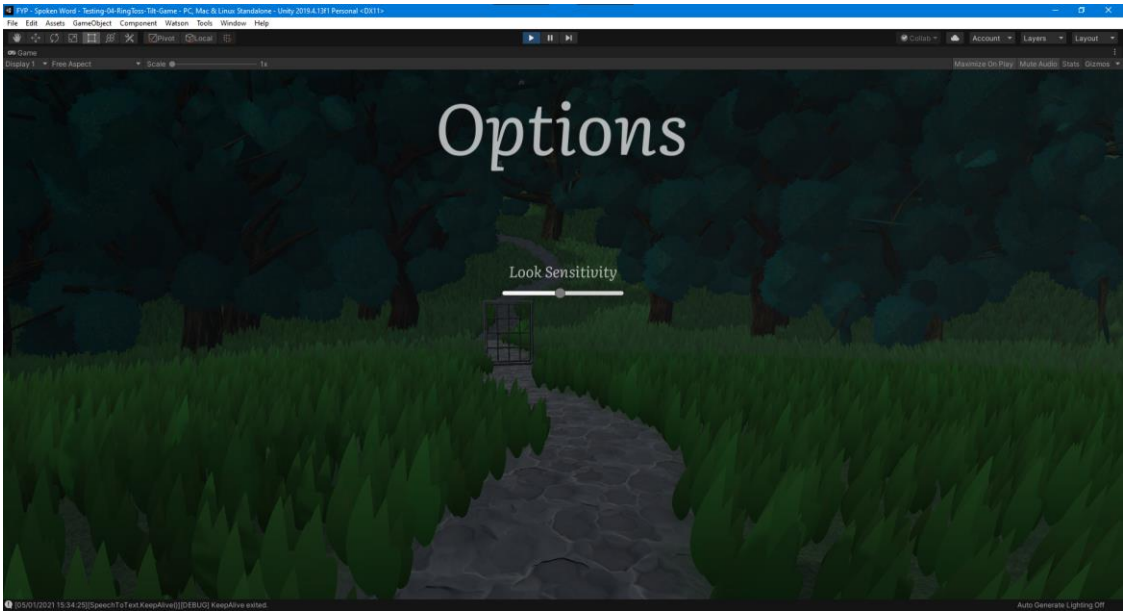


Figure 35, Options menu - Adjust look sensitivity from here.

6.1.2 Voice Controls Walkthrough

Using voice controls, the experience is largely the same. Upon launch, Spoken Word will connect to any microphone that is enabled on the computer it is running on. Nothing else will have to be done by the user to enable voice controls. From here, again the player can choose where to go. Using voice controls the player can say any utterance along the lines of “Move to objective …” to go to the objective they would like to complete first as in Figure 36.

Figure 36, Using voice controls for pathfinding.

Following this, the player can interact with the objective to play it (Figure 37). Figure 38 shows the player playing the *Tilt-A-Ball* minigame using voice controls. Directions and amount to tilt can be specified, for example “Tilt left a lot” will tilt the maze 45 degrees to the left. If the maze is too difficult the player can skip it by saying “Skip” or by pressing the skip button on KM / gamepad (Figure 39). Being able to switch between voice controls to KM and gamepad at a moment’s notice can be extremely helpful for some users as they would be able to tailor inputs to what best suits their needs.

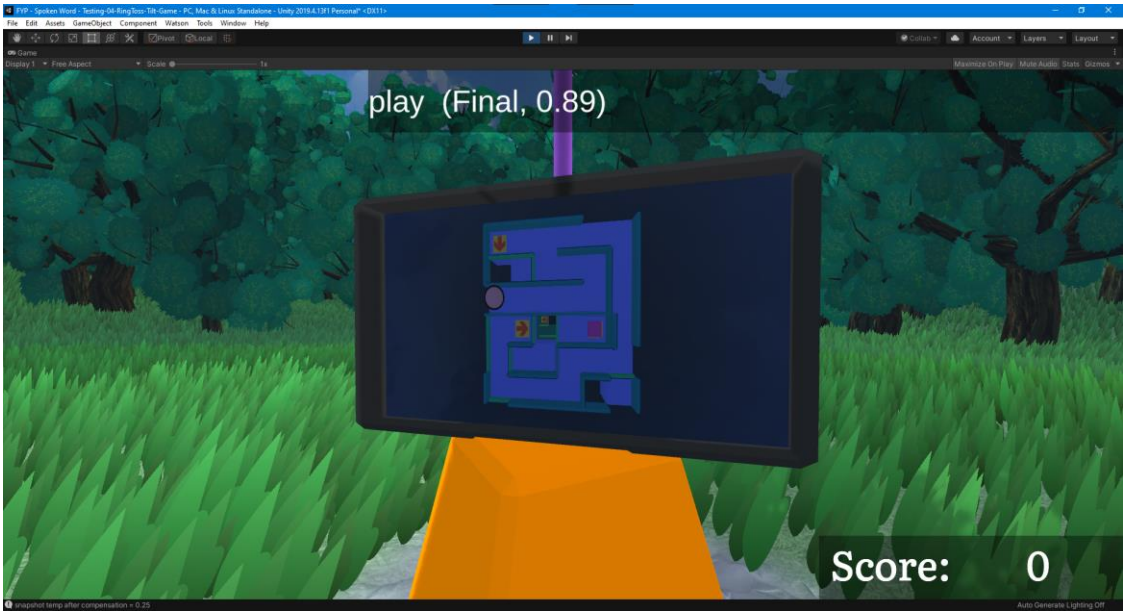


Figure 37, Using voice controls to interact with an objective.

Figure 38, Using voice controls to play a minigame.

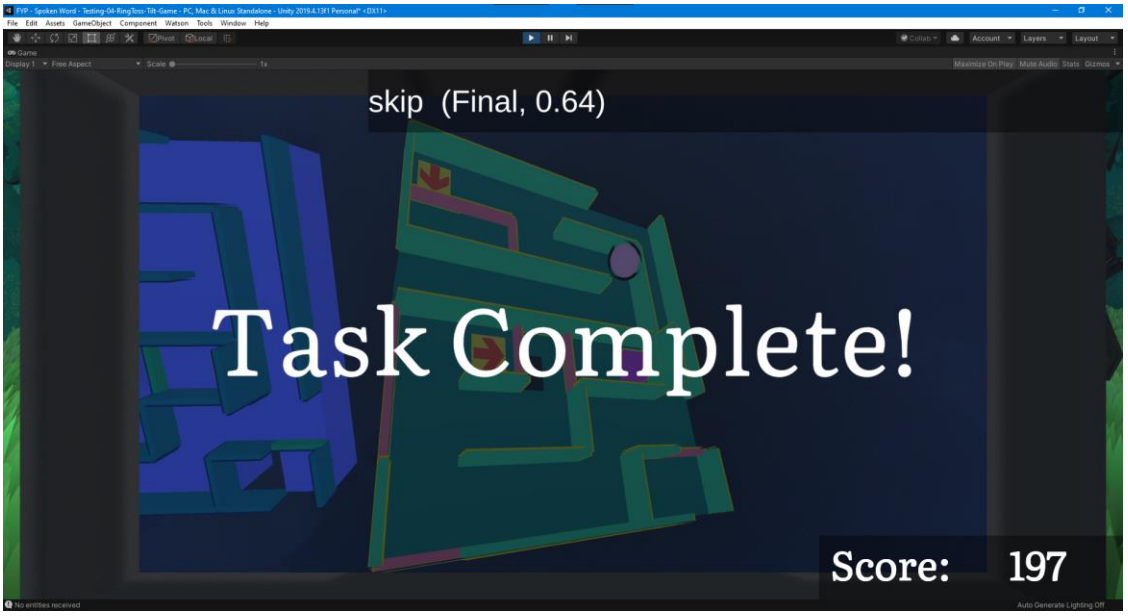


Figure 39, Difficult levels can be skipped with voice controls or standard controls.

The first objective, *Ring-Toss*, can also be completed with voice controls using the command “Fire” (Figure 40). Finally, once the player has completed both objectives the same cinematic as in Figure 32 will play and the door will open for the player to leave. To leave, the player can then move there themselves with commands like “Turn 90 degrees to the left” and “Walk forward”, or they can simply say “Exit the game” and the player character will pathfind and move to the exit (Figure 41).

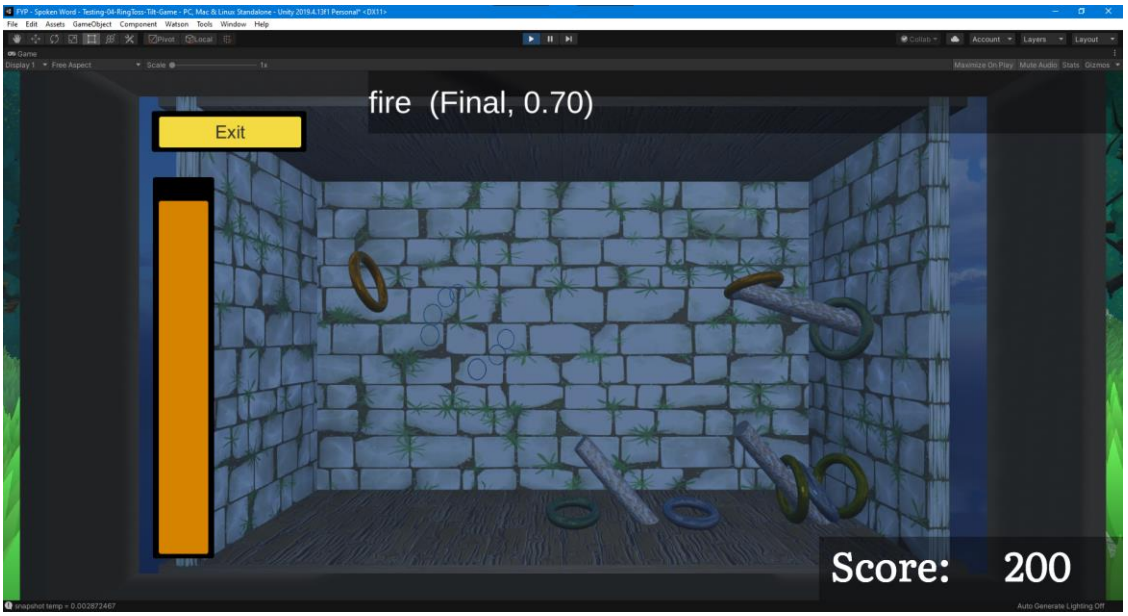


Figure 40, Ring-Toss minigame being played with voice controls.

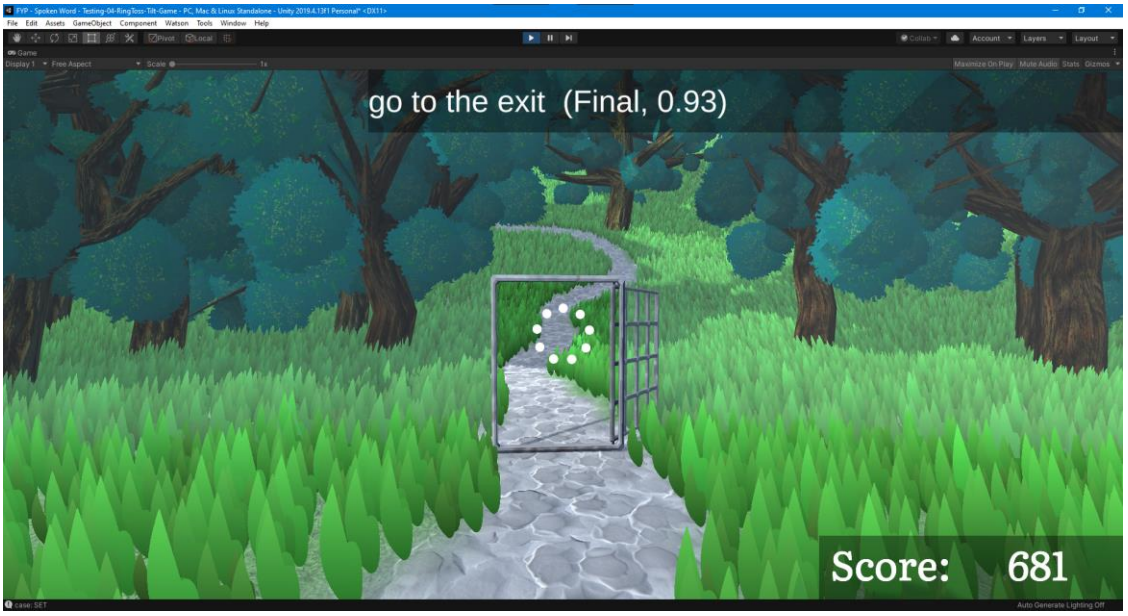


Figure 41, Pathfinding to finish the level.

6.1.3 Summary

Overall, Spoken Word is successful in providing an environment in which voice controls can be critically evaluated for video games. Although many of the requirements originally planned were not met as discussed in section 5.1, all requirements catering to motor control deficiencies specifically have been implemented. Unfortunately, one exception was made in the form of menu navigation using voice control. However, this omission does not detract from the focus of Spoken Word which was to evaluate voice controls in the context of video games; as menu voice control can be found on many modern

devices such as mobile phones. Since it is a technology that has already gained the focus of developers and favour of consumers, its implementation was not essential for Spoken Word.

6.2 User Evaluation

For the user evaluation of Spoken Word, completely able-bodied family members and friends were chosen to play through Spoken Word and complete an evaluation form. This evaluation form was designed with reference to the book Inclusive Design by Langdon, et al., and can be found in section 4.4.1. Some of the results of this evaluation are as follows.

6.2.1 Section 1

This section gathers information about the users and their thoughts on accessibility in video games overall. This ideally supports the research findings from Chapter 2:.

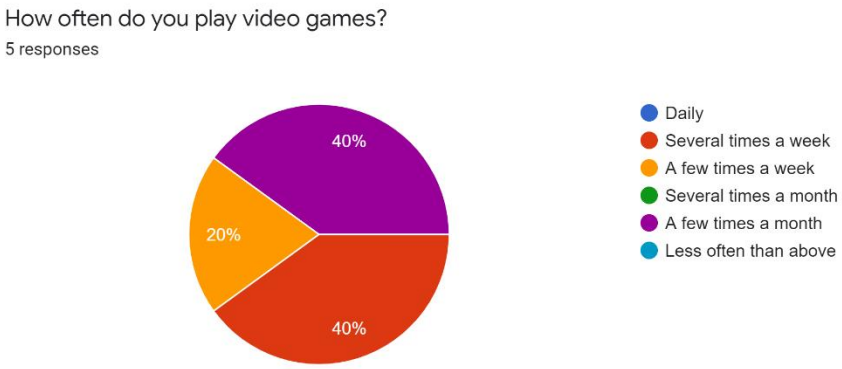


Figure 42, Section 1 - Question 1

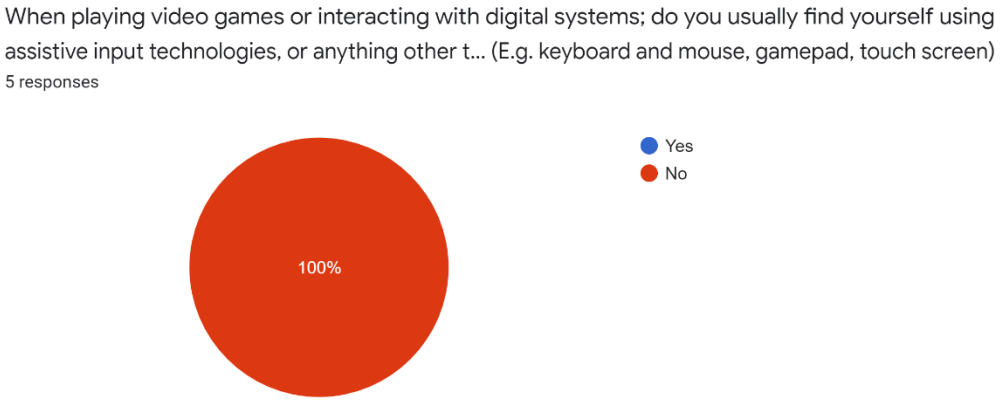


Figure 43, Section 1 - Question 2

A variety of people with different relationships with video games were chosen. All users were completely able-bodied and showed no signs of debilitating impairments.

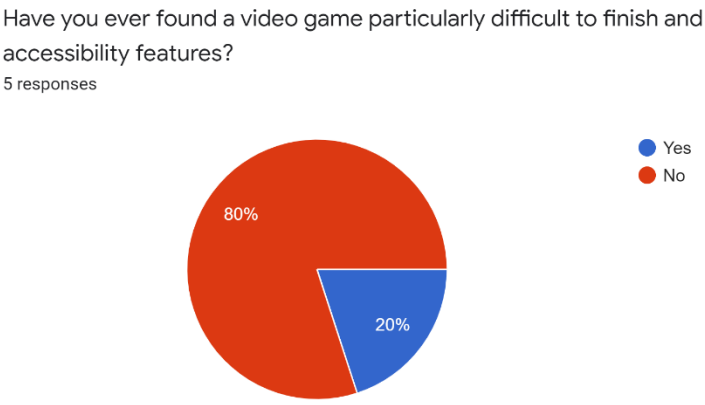


Figure 44, Section 1 - Question 7

One user found it difficult in the past to play a video game due to its lack of accessibility features. While the sample size is small, it supports that lack of accessibility features is a problem, especially if it is affecting users with no discernible disability.

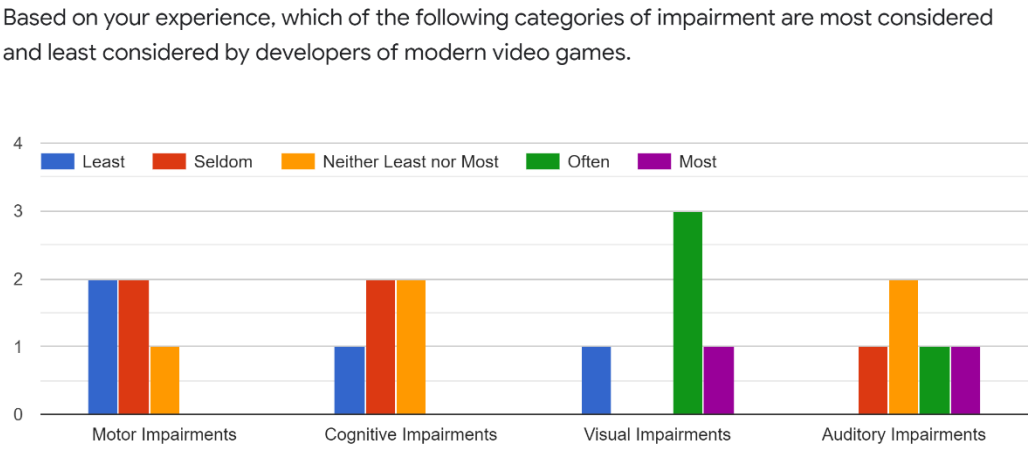


Figure 45, Section 1 - Question 8

In Figure 45 we can see that generally visual and auditory impairments are seen as the most accounted for by modern video games. Other disabilities that fall under cognitive and motor function do not get as much attention. Presumably because they are more difficult to alleviate.

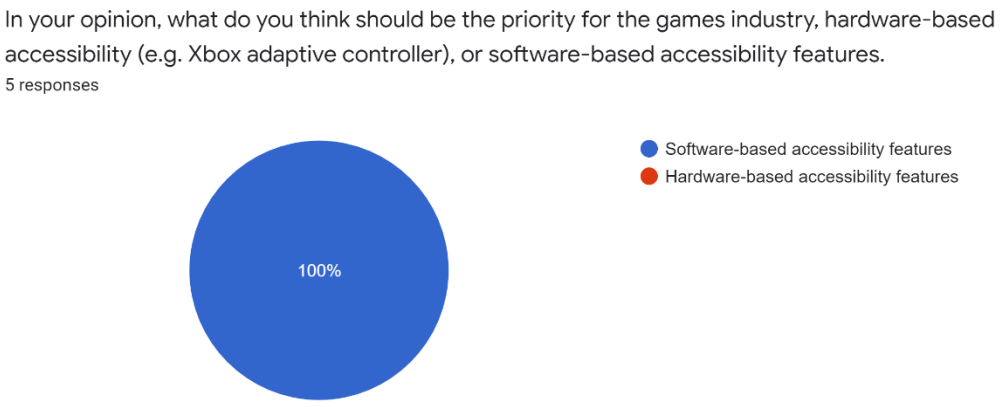


Figure 46, Section 1 - Question 10

All users agreed that software-based accessibility features should take priority over hardware-based features. Presumably, this is because enough hardware support exists, and developers should now focus on bespoke accessibility features that benefit the players of their own games. This is supported by a follow up question, where users were asked if enough emphasis was being placed on built.

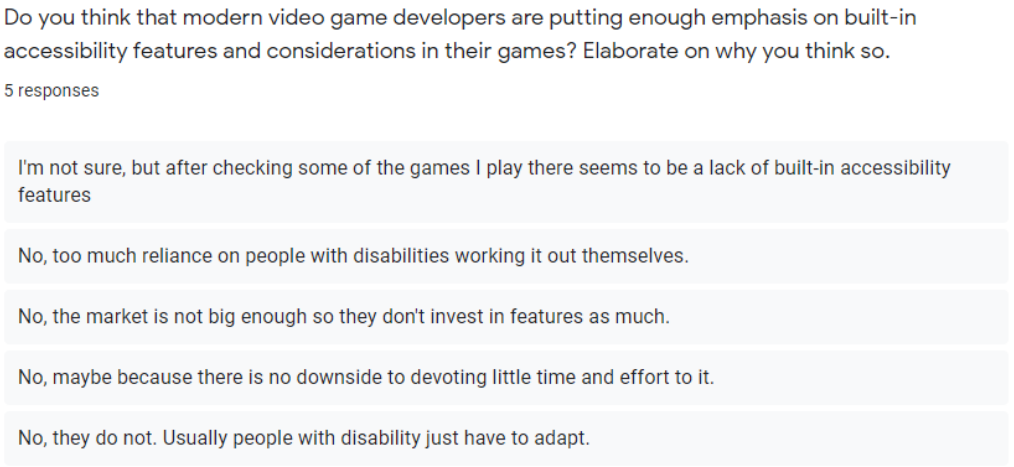


Figure 47, Section 1 - Question 12

Finally, all users agreed with the notion that gaming is beneficial for people with disabilities.

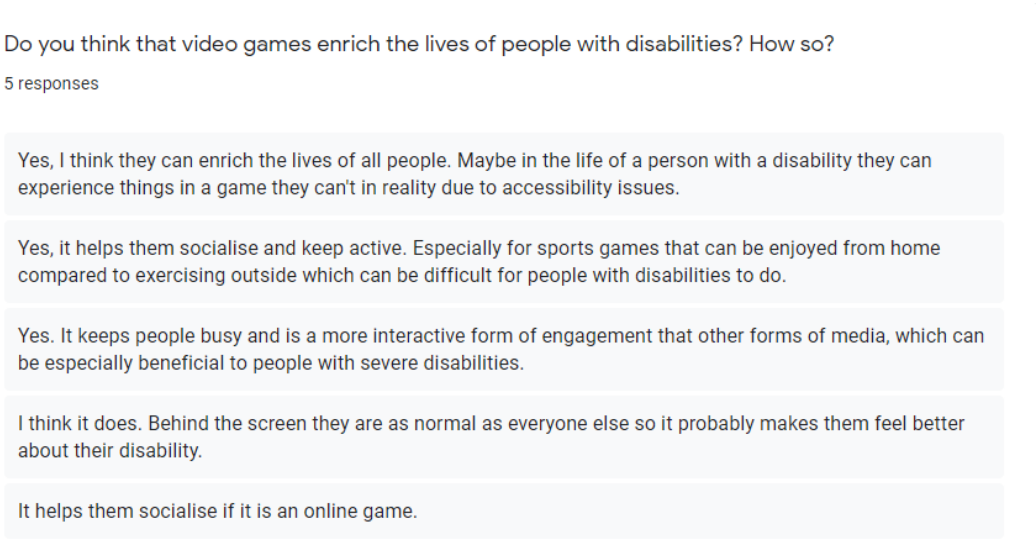


Figure 48, Section 1 - Question 11

6.2.2 Section 2

This section gathers feedback on the user's experience playing Spoken Word itself. This section provides qualitative feedback on the game and how it could be improved.

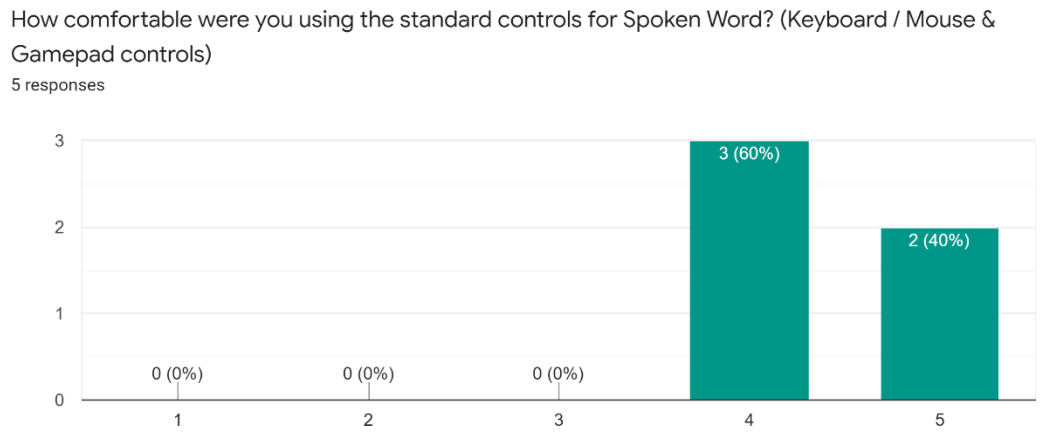


Figure 49, Section 2 - Question 1

As we can see, all users were relatively comfortable using the standard controls. Any issues experienced with the standard controls were isolated to the Tilt-A-Ball game.

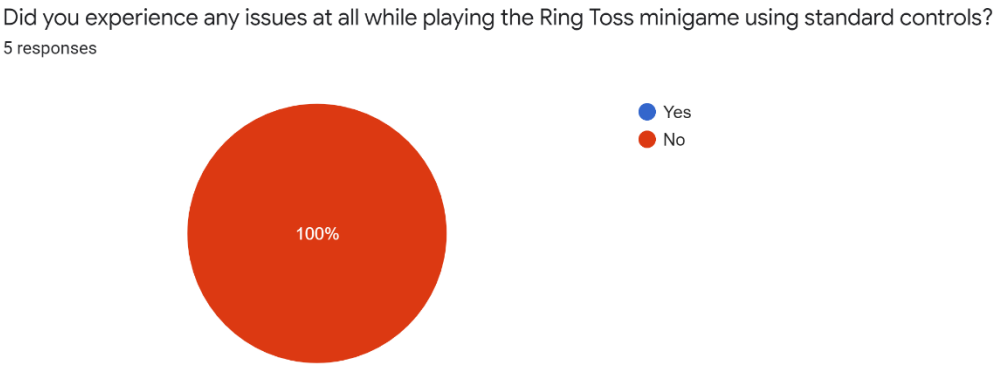


Figure 50, Section 2 - Question 2

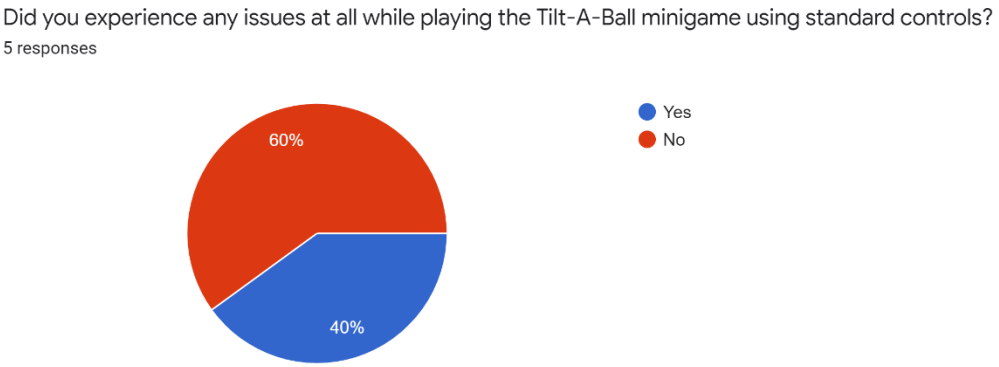


Figure 51, Section 2 - Question 4

The reasons why were due to a bug in the Tilt-A-Ball game. Looking into this myself I could confirm this was not due to the control system itself so overall I am happy with the feedback on the standard controls.

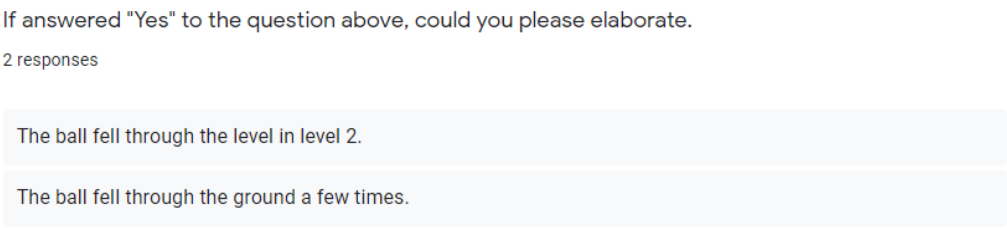


Figure 52, Section 2 - Question 5

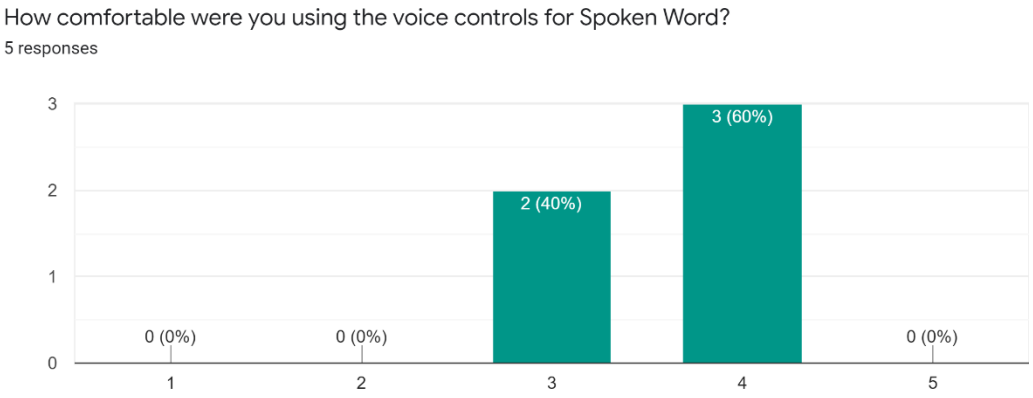


Figure 53, Section 2 - Question 6

Overall, it seems the users were less comfortable using the voice controls. However, this is not unexpected as all users answered 'No' to the question asking if they regularly make use of assistive input technology. Thankfully, most users found themselves becoming more accustomed to the voice controls as they progressed through Spoken Word.

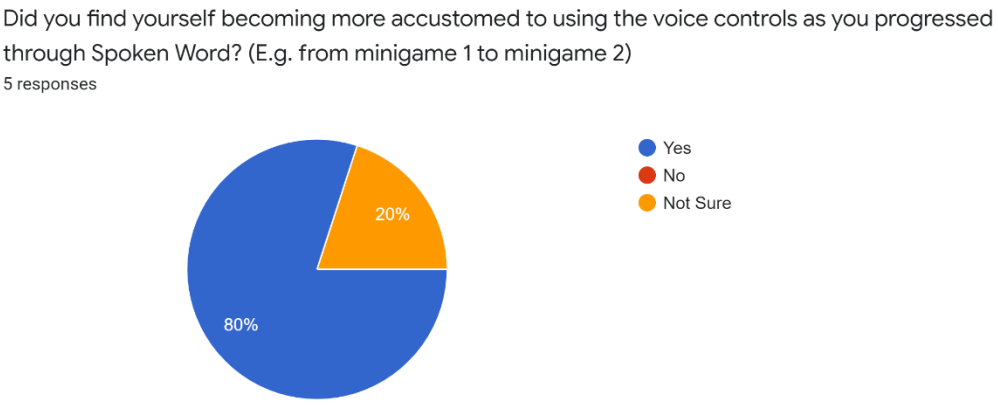


Figure 54, Section 2 - Question 7

However, most users did find issues with the voice controls that made them difficult to use. The most prominent issue being that the users could not mute their microphone while playing, meaning unwanted actions could take place either as a result of reacting to gameplay or due to background noises.

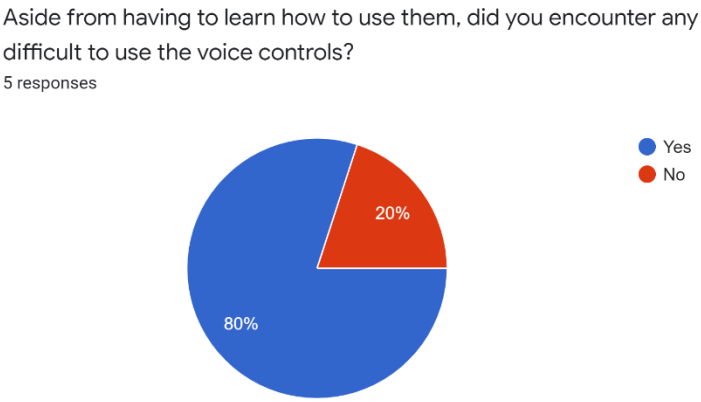


Figure 55, Section 2 - Question 10

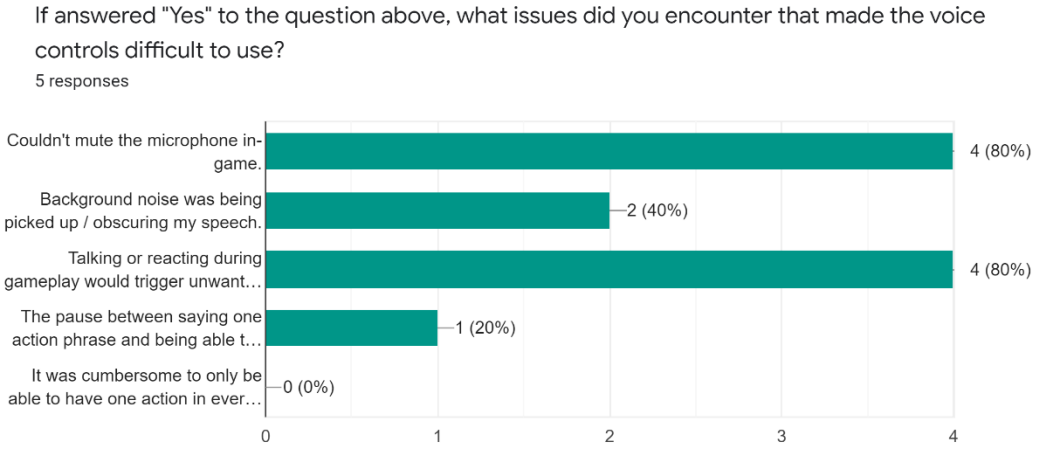


Figure 56, Section 2 - Question 11

Furthermore, most users also found issues with the Ring-Toss and Tilt-A-Ball game when playing with voice controls. While the voice controls were functional, it seems that many of the users had troubles. For Ring-Toss, the delay between saying a command and it happening in game made the gameplay feel awkward. For Tilt-A-Ball, the issue was that players could not move the ball as well as they could when using standard controls, this is somewhat expected as it is difficult to achieve the same level of fine motor control using just voice controls, through it was disappointing to see 80% of users had the same issue.

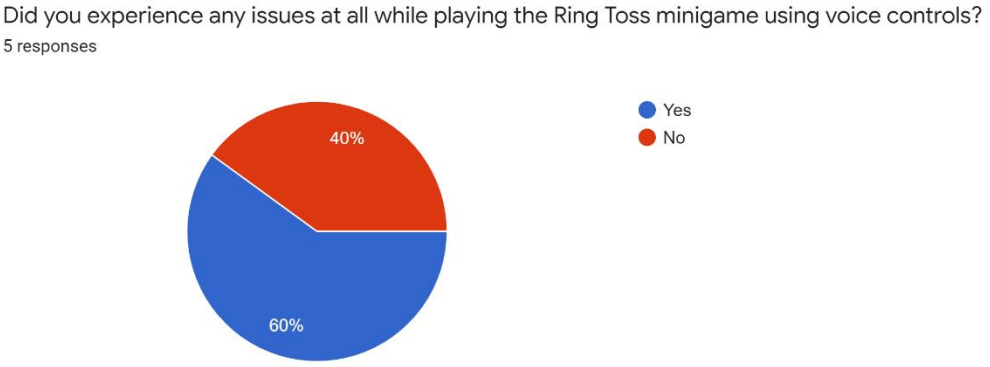


Figure 57, Section 2 – Question 12

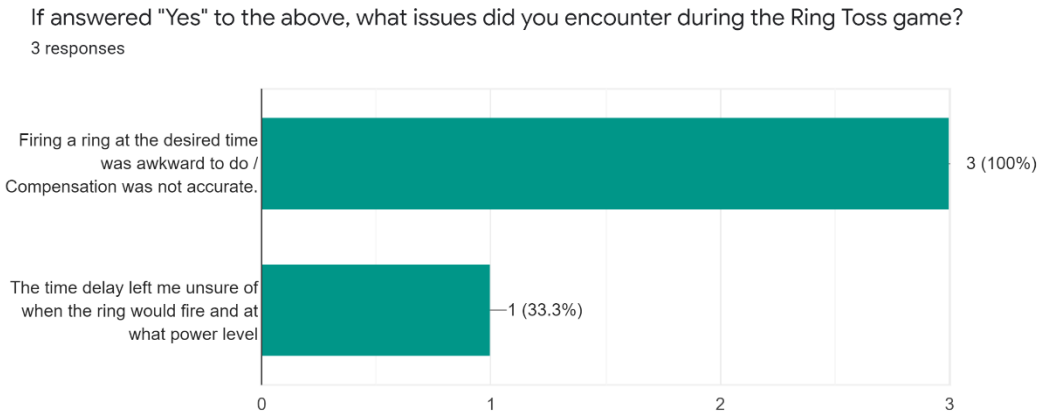


Figure 58, Section 2 - Question 13

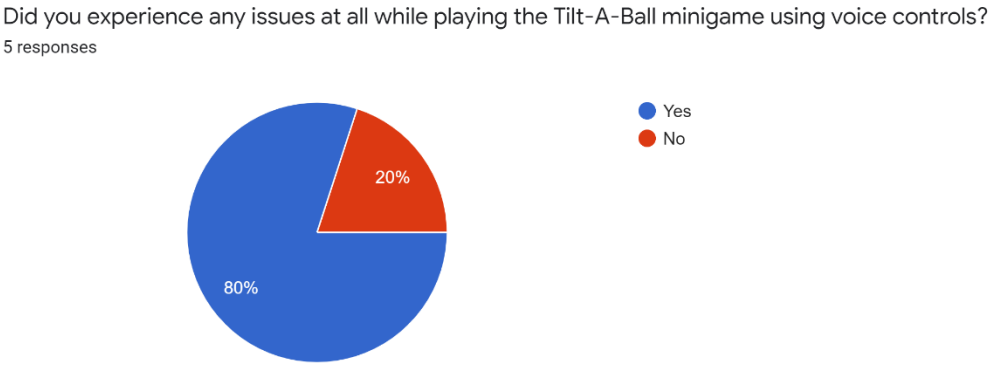


Figure 59, Section 2 - Question 14

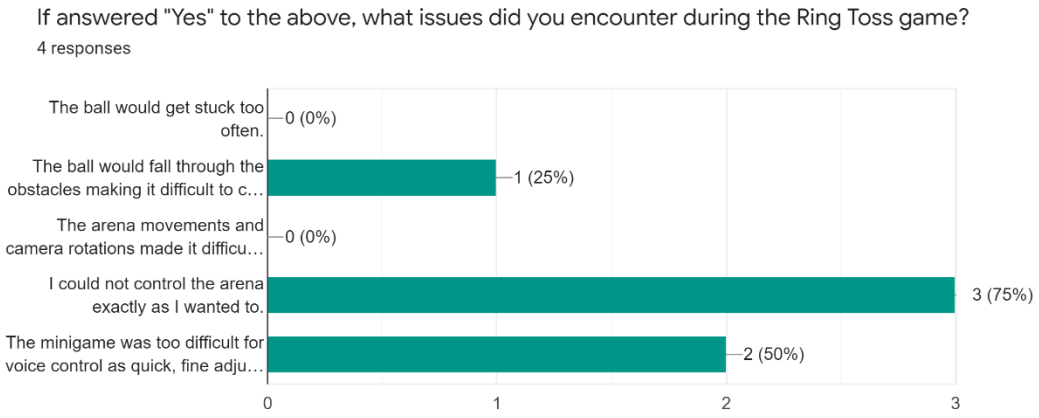


Figure 60, Section 2 - Question 15

Despite these issues, the users agreed that Tilt-A-Ball was the better demonstrator of voice controls between the two. Most of the users agreeing that time-based games like Ring-Toss are more difficult to play in comparison to puzzle games like Tilt-A-Ball, as the processing delay made it difficult to fire the ring exactly when desired.

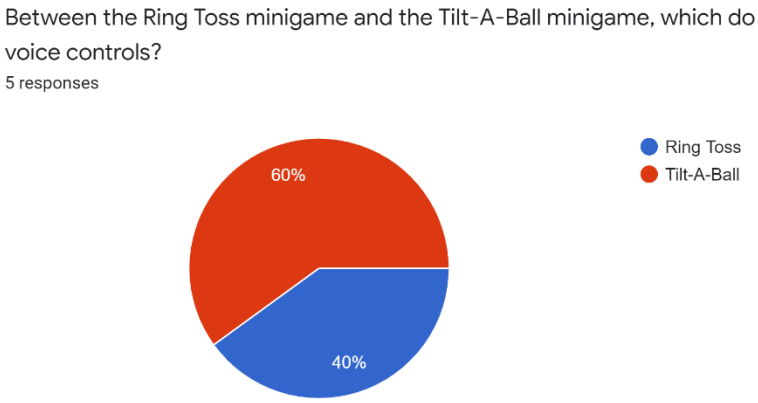


Figure 61, Section 2 - Question 17

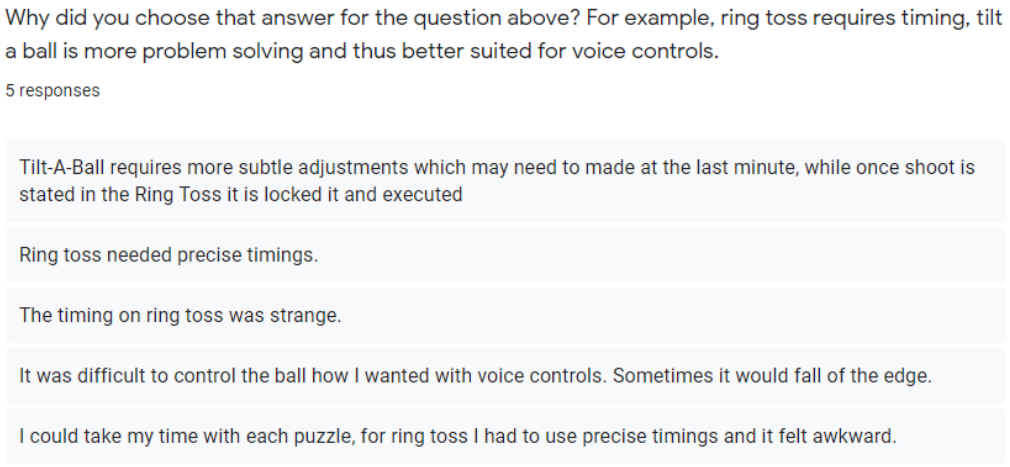


Figure 62, Section 2 - Question 18

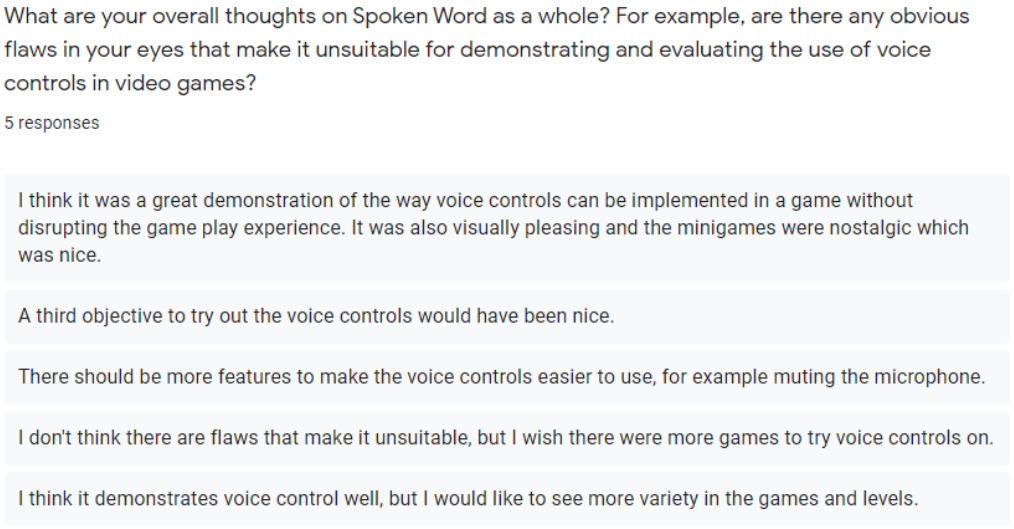


Figure 63, Section 2 - Question 19

Overall, the users agreed that Spoken Word was a suitable demonstration of voice controls. The main flaw being that there should have been more games to try out as the two currently present were not enough to build enough of a cohesive feeling towards voice controls in video games.

6.3 Evaluation of Work

6.3.1 What Could Have Been Done Differently

In retrospect there is not much that I would change about the development of Spoken Word. It was developed successfully using the technology chosen from the start with no

major issues. However, one change that would have benefited the outcome would be to reduce the scope of Spoken Word. While the chosen requirements and original MVP would have produced a great solution fit for purpose, the amount to be completed in the given amount of time was too optimistic and could not be met.

One change to the development that would have improved Spoken Word would be to put the effort in to display information in multiple forms (Br9.), at the time when such information was being developed and implemented into the game. For example, when developing the starting instructions screen, a voice recording instructing the same should have been created and implemented at the time. Doing this at the end of the development cycle proved to be too much work to take on and would have likely not have been completed to a high standard so the requirement had to be cut.

6.3.2 Future Work

In the future it is likely that I will keep working on Spoken Word until it is at a stage where I am happy with it and have completed all the requirements set out at the start of development. At the very least, I would continue making experimental games in Unity as I have come to enjoy the development process through this project.

My focus would be on developing the code architecture to allow for straightforward implementation of further requirements. For example, requirement Br7. specifically, would require a lot of effort to make it so that UI elements could be controlled via the voice controls. All UI elements and their positions relative to each other would need to be stored in a data structure that corresponds to the control scheme or screen that the player is in. A function in the voice commands script would then be written to change the selected UI element depending on an extracted direction entity from IBM Watson. Additionally, work would have to be done on the Watson Assistant so that UI elements could also be selected by name and not just direction. Doing this would tick off all the motor-control based requirements outlined previously.

Furthermore, in response to the user feedback in section 6.2I would like to spend more time working on the processing speed of the voice controls and in developing more minigames that demonstrate the potential of voice controls for video games. Unfortunately for the former point, the processing of voice controls is currently heavily reliant on internet speed, as voice recordings have to be sent to the IBM service hosted on live servers and then back again to retrieve intent. One solution could be to switch STT services to one that runs locally on the user’s machine. While it would require an overhaul of the voice control system, it is something I would like to explore further as part of future work.

Chapter 7: **Conclusion**

In conclusion, Spoken Word has reached a close to MVP-like state with still a few requirements to be completed. However, as it is I believe Spoken Word presents an environment in which voice controls can be critically evaluated for slower paced, exploration and puzzle-based video games. An environment where time-critical games like Ring-Toss can be compared against more problem-solving oriented games like Tilt-A-Ball to evaluate which style best suits voice control. Furthermore, players can make use of the robust input system to play in the way that is most comfortable for them, whether it be using voice control or otherwise.

As mentioned in previous sections, given more time there is a lot more to be accomplished that would improve functionality, accessibility, enjoyment and elevate Spoken Word to something that is closer to a commercial product. However, I believe I have achieved many of the objectives outlined in section 1.3 and I am satisfied with what I have learned and the challenges I overcame. Given I do not have the strongest software engineering background, I am happy to have been able to learn more about and apply software engineering theory and games engineering skills in a context about helping others, and I look forward to doing more of the same in the future.

References

Bierre, K., Chetwynd, J., Ellis, B., Hinn, D.M., Westin, T. and Ludi, S., 2005 Game Not Over: Accessibility Issues in Video Games. [online] Available at: https://www.researchgate.net/profile/Kevin-Bierre-2/publication/267403944_Game_Not_Over_Accessibility_Issues_in_Video_Games/links/546de0d70cf2a7492c560d87/Game-Not-Over-Accessibility-Issues-in-Video-Games.pdf [Accessed 1 Nov. 2020].Department for Work and Pensions, 2020. *Family Resources Survey 2018/19*.

Deutsch, J.E., Borbely, M., Filler, J., Huhn, K. and Guarrera-Bowlby, P. (2008). Use of a Low-Cost, Commercially Available Gaming Console (Wii) for Rehabilitation of an Adolescent With Cerebral Palsy. *Physical Therapy*, 88(10), pp.1196–1207.

Ellis, B., Ford-Williams, G., Graham, L., Grammenos, D., Hamilton, I., Lee, E., Manion, J. and Westin, T., 2020. Game Accessibility Guidelines | Full List. [online] gameaccessibilityguidelines.com. Available at: http://gameaccessibilityguidelines.com/full-list/ [Accessed 22 November 2020].

Game Accessibility Special Interest Group, I., 2020. For Developers & Researchers - IGDA Game Accessibility SIG. [online] IGDA Game Accessibility SIG. Available at: https://igda-gasig.org/how/for-developers-researchers/ [Accessed 9 November 2020].

Grammenos, D., Savidis, A. and Stephanidis, C. (2009). Designing universally accessible games. *Computers in Entertainment*, 7(1), p.

Griffiths, M. D. (2002). The educational benefits of videogames. Education and health, 20(3), 47-51.

Hawken Miller (2019). “It’s my escape.” How video games help people cope with disabilities. [online] Washington Post. Available at: https://www.washingtonpost.com/video-games/2019/10/14/its-my-escape-how-video-games-help-people-cope-with-disabilities/ [Accessed 3 May 2021].

https://www.reuters.com/. 2020. Report: Gaming Revenue To Top $159B In 2020. Available at: https://www.reuters.com/article/esports-business-gaming-revenues-idUSFLM8jkJMl [Accessed 10 November 2020].

IBM (2020). Getting Started with Speech to Text. [online] Cloud.Ibm.com. Available at: https://cloud.ibm.com/docs/speech-to-text?topic=speech-to-text-gettingStarted [Accessed 3 Mar. 2021].

IBM (2021a). About Watson Assistant. [online] Cloud.Ibm.com. Available at: https://cloud.ibm.com/docs/assistant?topic=assistant-index [Accessed 3 Feb. 2021].

IBM (2021b). Getting Started with Watson Assistant. [online] Cloud.Ibm.com. Available at: https://cloud.ibm.com/docs/assistant?topic=assistant-getting-started [Accessed 3 Mar. 2021].

IBM (2019). IBM/vr-speech-sandbox-cardboard. [online] GitHub. Available at: https://github.com/IBM/vr-speech-sandbox-cardboard [Accessed 4 Mar. 2021].

Ilg, W., Schatton, C., Schicks, J., Giese, M.A., Schols, L. and Synofzik, M. (2012). Video game-based coordinative training improves ataxia in children with degenerative ataxia. *Neurology*, 79(20), pp.2056–2060.

L. Garber, "Game Accessibility: Enabling Everyone to Play," in Computer, vol. 46, no. 6, pp. 14-18, June 2013, doi: 10.1109/MC.2013.206.

Langdon, P.M., Lazar, J., Heylighen, A. and Dong, H. (2014). *Inclusive Designing*. Cham Springer International Publishing.

Rodríguez Jiménez, M., Pulina, F. and Lanfranchi, S. (2015). *Video games and Intellectual Disabilities: a literature review -IRCCS Life Span and Disability*. [online] *Research Gate*. Available at: https://www.researchgate.net/publication/291829431_Video_games_and_Intellectual_Disabilities_a_literature_review_-IRCCS_Life_Span_and_Disability [Accessed 6 Jan. 2021].

Unity Technologies (2020a). Unity - Manual: GameObjects. [online] Unity3d.com. Available at: https://docs.unity3d.com/Manual/GameObjects.html [Accessed 10 Jan. 2021].

Unity Technologies (2020b). Unity - Manual: Order of execution for event functions. [online] Unity3d.com. Available at: https://docs.unity3d.com/Manual/ExecutionOrder.html [Accessed 10 Jan. 2021].

Unity Technologies (2020c). Unity - Manual: Package Manager. [online] Unity3d.com. Available at: https://docs.unity3d.com/Manual/class-PackageManager.html [Accessed 3 May 2021].

Unity Technologies (2020d). Unity - Manual: Scenes. [online] Unity3d.com. Available at: https://docs.unity3d.com/Manual/CreatingScenes.html [Accessed 10 Jan. 2021].

Unity Technologies (2020e). Unity - Manual: Coroutines. [online] Unity3d.com. Available at: https://docs.unity3d.com/Manual/Coroutines.html [Accessed 10 Jan. 2021].

Unity Technologies (2020f). Unity - Manual: Special folder names. [online] Unity3d.com. Available at: https://docs.unity3d.com/Manual/SpecialFolders.html [Accessed 10 Jan. 2021].

Unity Technologies (2020g). Unity - Scripting API: MonoBehaviour.Awake(). [online] Unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html [Accessed 10 Jan. 2021].

Unity Technologies (2020h). Unity - Scripting API: MonoBehaviour.FixedUpdate(). [online] Unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html [Accessed 10 Jan. 2021].

Unity Technologies (2020i). Unity - Scripting API: MonoBehaviour.Update(). [online] Unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/MonoBehaviour.Update.html [Accessed 10 Jan. 2021].

Warman, P., 2018. Newzoo Cuts Global Games Forecast For 2018 To $134.9 Billion; Lower Mobile Growth Partially Offset By Very Strong Growth In Console Segment | Newzoo. [online] Newzoo. Available at: https://newzoo.com/insights/articles/newzoo-cuts-global-games-forecast-for-2018-to-134-9-billion/ [Accessed 10 November 2020].

Yuan, B., Folmer, E. and Harris, F., 2010. Game accessibility: a survey. Universal Access in the Information Society, 10(1), pp.81-100.

Appendix A

General

Key	Description	Seen In	Positives	Notes	
G1.	Offer different difficulty levels. Allow difficulty to be adjusted during gameplay.	Vast majority of narrative or single player games.	Allows players of different abilities the same experience. What may be challenging to one person with a certain disability, may be easy to another. Custom difficulty promotes inclusion to everyone.	It is important to not have demeaning names for certain difficulty levels. Not entirely applicable to Spoken Word as it will only feature set puzzles of a certain difficulty.	N
G2.	Provide a detailed list and description of accessibility features in-game.	Nier: Automata, The Last of Us Part II.	By listing all features explicitly, players do not have to go looking for them and the work that went into many features doesn't go wasted.	In Spoken Word, all accessibility options will be displayed on the opening screen of the game. That way players can tailor their experience before playing.	Y
G3.	Allow settings to be saved to different profiles.	World of Warcraft, Counter Strike: Global Offensive	Especially beneficial for charities that work with disabled gamers, as hardware is likely to be shared between multiple people.	This will not be included in Spoken Word as it is outside of the project's scope.	N
G4.	Means to bypass gameplay elements that are not essential or part of the core gameplay.	L.A. Noire, Nier: Automata.	Allows players to skip skill checks and requirements that are unique to certain points in a game. This way there's no need to learn new control schemes or game mechanics which could confuse or prove difficult for some players.	This will not be included in Spoken Word as the core game will be designed with accessibility considerations in mind. Meaning that all areas of the game will be similar to one another and have intuitive controls while still being interesting and engaging.	N
G5.	Ensure speech input is only required as an alternative input method.	Playstation 4 interface, Xbox One interface.	Utilising speech recognition can be massively beneficial for players with little to no fine motor control. However, it can also exclude those who are physically unable to speak.	Spoken Word will utilise speech input as a main way to provide game inputs and is integral to the way the game can be played. However, it will only be used in conjunction with traditional input so as not to exclude any demographic.	Y

G6.	Speech recognition will be based on individual words from a small vocabulary.	No best practice examples.	Many people with speech impairments are still able to communicate, but with a limited range and vocabulary. Ensuring no complex words or phrases are needed for input means that they will not be excluded.	Spoken Word will only use a limited vocabulary for character control. E.g. “Go to [location name]”. Other voice commands for solving puzzles will be visible on screen.	Y

Motor

Key	Description	Seen In	Positives	Notes	
M1.	Allow controls to be remapped / reconfigured.	Vast majority of PC releases, mainstream consoles.	Allows players to customise the controls to suit their individual needs	Offered on consoles at the platform level, this means game UIs won’t reflect changes and can cause confusion.	Y
M2.	Options to adjust sensitivity.	Vast majority of game releases. Splatoon 2, Counter Strike.	Allows players to customise the joystick and mouse sensitivity to suit their playstyle / disability.	Can be useful for people with hand tremors, sensitivity can be set to low to mitigate effects caused by tremors. Can also be used in conjunction with gyroscope controls for aiming.	Y
M3.	Support more than one input device.	Vast majority of PC releases. Call of Duty: Modern Warfare (PS4)	Allows players to use input devices that are suited to their motor skills and physical characteristics.	Controller support inherently means Xbox adaptive controller support. Outside of accessibility, some users prefer controllers over keyboard and mouse and vice versa.	Y
M4.	Options to adjust game speed.	Celeste, The Last of Us Part II, Tropico 5	Provides users with slow reaction times due to mobility or cognitive impairments to have enough time to process the information on screen.	Can be to the benefit of older players without disability as well. Adjusting speed can mean playing in slow motion or reducing in-game passage of time (common in strategy games).	N
M5.	Avoid repetitive quick inputs. E.g. button mashing and quick time events.	Uncharted 4, The Witcher 2	These events usually require rapid, repetitive precision. Can be a barrier or exclude players with arthritis or other impairments.	While quick time events that require rapid precision have their place, the same effect that does not exclude anyone can be accomplished with holding down an input.	Y
M6.	For PC games, support windowed mode for compatibility with overlaid virtual keyboards.	Vast majority of PC releases, Torchlight 2	Many players who are unable to use a physical keyboard may opt to use a virtual one instead that can be controlled via a mouse or eye tracking software.	For these to work, the game must be in windowed mode, and allow other applications to sit over the top.	Y

M7.	Do not make precise timings essential to gameplay. Otherwise, offer alternatives.	Final Fantasy XV (FFXV)	Allows players who find it difficult to move around input devices to have the time to plan and execute their intended input without having to worry about time related penalties.	Can be applied to puzzle games similar to Candy Crush and Bejewelled by offering unlimited time game modes. Can also be applied to action games like FFXV, where wait mode can be used mid battle to evaluate next steps.	Y
M8.	Implement simple control schemes that are compatible with external assistive technologies.	Forza 4	Allows players who are already comfortable with their current assistive technology to be able to apply it to a game.	Greatly beneficial for players with severe impairments that find extreme difficulty in using a computer without the use of eye tracking for example.	N
M9.	Consider supplementary input and controls as part of the game design.	Phasmophobia	Promotes accessibility at a basic level. Promotes the feeling of normality as opposed to being accommodated for.	Games like Phasmophobia use speech recognition as a core part of its gameplay. This concept can be expanded to be used in all areas of the game included character control and menu control.	Y

Cognitive

Key	Description	Seen In	Positives	Notes	
C1.	Readable font, clear language, clear formatting.	Guacamelee!, Destiny 2	This not only helps those with low reading ability, but also benefits users with visual impairments that may find it difficult read certain fonts and sizes.	Players have a wide of ability, preference and viewing environments. Allowing the choice of font, while ideal is not always necessary. Setting a large default size, or adjustable size is a good first step.	Y
C2.	Include tutorials, contextual in-game tips and guidance, reminders of goals during gameplay, reminders of controls during gameplay. Reinforce text with visuals.	Monster Hunter: World, XCOM: Enemy Unknown, Skyrim	Tutorials, tips, guidance and reminders can be extremely beneficial if they are correctly spread out throughout gameplay. People with cognitive impairments such as learning disabilities would benefit the most from consistent reinforcement of game objective and mechanics.	Used in conjunction with a means of practice without failure and allowing players to progress through text prompts at their own pace will make it much easier for people with cognitive disabilities to get comfortable with the control scheme and objectives of a game.	Y
C3.	Implement a simple and clear narrative structure.	To the Moon, Monument Valley	Complex stories can easily confuse those who have difficulty remembering or understanding complex concepts.	For many games, a complex narrative is often the appeal and selling point. However, when this isn't the case, a simple narrative is more suited.	Y
C4.	Avoid flickering images and repetitive patterns. Avoid sudden unexpected movement or events.	Towerfall Even The Ocean	Seizures caused by video games are very common. Especially within many competitive titles where fast paced action can cause screens to drastically change frame by frame. Avoiding this type of action is best in games geared towards the disabled.	Even if all common triggers are avoided, there are still many possibilities that a seizure can be induced from a game. Always have a seizure warning for sequences that seem especially problematic.	Y

Vision

Key	Description	Seen In	Positives	Notes	
V1.	Ensure essential information is not expressed by colour alone. Or ability to adjust the colour scheme.	Destiny, For Honor, Splatoon	Red and green colour blindness is very common, presenting players with options to adjust the colour scheme of UI or gameplay elements means they wont miss out on any important information.	Especially important in games where colour is a key factor, such as Splatoon.	Y
V2.	Choice of text / caption colour or high contrast by default. Ability to adjust post-processing effects such as contrast and saturation.	RuneScape, Fallout 3, The Last of Us Part II, For Honor, Uncharted.	Adjustable UI contrast and gameplay post processing effects means visually impaired users suffering from colour blindness to partial blindness to adjust the game experience so that they don't miss out on any vital information on happening on screen, whether it be text or gameplay.	Though high contrast text is generally beneficial it can cause some issues for people with dyslexia. Different colour combinations work for different people.	Y
V3.	Give clear indication of elements that are interactive in a level.	Doom, Most video game releases via button prompts.	Players with visual impairments and or that are new to video games will find this especially useful as they learn how to play a new game.	This can be reinforced by using clear and consistent metaphors and conventions throughout an entire game. Achieved by having clear contrast to surrounding objects.	Y
V4.	Allow font size to be adjusted.	Final Fantasy XIII-2, The Last of Us Part II	Readable UI and captions are integral for all players. Having a slider to adjust font size allows players to tailor their experience.	Setting a good default size is of course very beneficial and easier to implement, however is unlikely to meet everyone's tastes regarding design and aesthetic.	Y
V5.	Pre-recorded voice overs for all text in menus and during gameplay. Screen reader support is another method of achieving this.	FREEQ	Players will be able to understand game direction even with complete lack of vision.	Can be expensive to produce voice overs as large games may have a large amount of text. However, it is technically easier to implement than screen reader support.	Y
V6.	Provide a voiced GPS and or pingable audio map.	The Last of Us Part II	The ability to ping important items in the surrounding area can make complex 3D environments navigable by people with little to no vision.	A voice description track, while time consuming and costly to implement would likely be easier than developing a sonar-style pulse to orientate players.	N

Auditory

Key	Description	Seen In	Positives	Notes	
A1.	Provide subtitles for all important / supplementary speech and captions for all background noises.	Most game releases. Tomb Raider, Portal 2, Destiny 2.	Useful to alleviate issues encountered by players with hearing loss, but also used by many that have no discernible disabilities. Captions also help people understand atmospheric sounds and the direction that they are coming from.	Important speech means significant impact to gameplay and other elements of the game. They are used for all kinds of reasons and are common in all forms of media. If subtitles and captions are used, it's important to have options that allows the customisation of subtitles.	Y
A2.	Provide separate volume controls or mutes for all audio tracks. E.g. sound effects, speech, background noises, music, etc.	Most game releases, Diablo 3, Starcraft 2.	Hearing loss can make some audio frequencies difficult to hear for many players, basic audio control can alleviate this issue for the affected people.	Also beneficial for people with cognitive impairments, too many audio sources can make it difficult to single out or focus on important ones. Additionally, beneficial for people with visual impairments, individual sounds become especially important when not relying on visual cues.	Y
A3.	Ensure audio options including captions, audio levels, stereo / mono options can be adjusted before the game starts or any sound is played.	The Last of Us Part II Destiny 2, Far Cry 3.	Allowing this means any information presented at the start of the game can be comfortably consumed by a player. Enabling subtitles by default is a good way to alleviate this, however full control over audio settings before playing audio would be the best solution.	The most commonly missed piece of information in any game is the introductory cinematic. Many times this is because players are not given an option to customise their sounds settings to suit their environment and physical characteristics.	Y
A4.	Captions and subtitles must be presented at an appropriate words-per-minute.	No best practice example, though usually not an issue either.	Speech happens at a faster rate than reading, so keeping subtitles at a legible pace is extremely helpful for the partial and completely deaf.	When creating captions, try not to cut down or paraphrase speech in a way that will alter the meaning of the information being presented.	Y
A5.	Ensure supplementary information conveyed by audio is also represented with visuals. Ensure essential information is not conveyed by sound alone.	Call of Duty series, Gears of War series, Everybody's Gone To The Rapture.	Sound cues that have no visual components will be entirely inaccessible to the completely deaf. Visual cues like arrows that point to where you are being shot from or who is speaking can solve this.	Aesthetic sounds like music and steps are not necessary to be given visual cues. If too many are provided it's possible to create clutter on the screen which could cause visibility issues.	Y