Accessible Codenames

Jamie Callan
979687

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Bachelor of Science

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

April 28, 2022

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ..... (candidate)

Date .....

Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

Abstract

Games have become an integral part of life today with an estimated 3.24 billion people playing them around the world. [1] But of those people, not all of them can play them exactly as the creators intended. Some people may need only slight changes, but others may need drastic changes in order for them to play it enjoyably. Whilst many game creators do try to accommodate for these people, they aren't always do everything they could when they have so much time and such big budgets to work on these games.

My project intends to show how you can implement accessibility features into games, without ruining the experience for those who do not need them and showing that people with these disabilities can still play with those who do not.

I accomplished this by recreating a board game and having accessibility be the primary focus outside of just recreating the actual game. Unlike the large companies creating games today I am only 1 person and with no where near as much time as they have, I was unable to take every possible disability into account, so this is mostly a proof of concept that games can be made with a focus on accessibility, and still be successful.

Acknowledgements

I would first like to thank my supervisor, Ms Casey Hopkins for her guidance throughout the development of my project and for giving meaningful feedback to both this and my initial document.

I would also like to thank my mother, father and sister, who have always been there for me to help with any problems I encountered, and always supporting me and the choices I've made in life up to this point.

My friends and colleagues have also made these past few years much more enjoyable and have been great to work with on various projects. Although a year and a half was spent away from them for online learning, I'm glad I was able to work with them in person for my final year.

Finally, I just want to thank Tango, my cat, for always being a source of comfort for whenever I was stressed, and the same for his brother Jasper who never failed to make me smile for all 11 years of his life.

Contents

List of Tables **v**

List of Figures **vi**

1 Introduction **1**

1.1 Motivations 1

1.2 Aims & Objectives 2

2 Background Research **3**

2.1 Other solutions to this problem 3

2.2 Tools used 6

3 Project Development **9**

3.1 Software Life Cycle 9

3.2 Android Application 10

3.3 Online Server 18

3.4 Accessibility Features 18

3.5 Risk Analysis 20

3.6 Aims & Objectives - revisited 21

4 Testing & User Study **22**

4.1 Testing 22

4.2 User Study 24

5 Reflection and Evaluation **26**

5.1 Final Product 26

5.2 Reflection 27

6 Future Work **28**
6.1 Android Application 28
6.2 Online Server 28
6.3 Accessibility Features 28

7 Summary **30**

Bibliography **31**

Appendices **32**

A Examples of server code **33**

B Example of client code **35**

List of Tables

4.1 Testing Table 23

List of Figures

2.1 Mobile Operating System Market Share. 6

3.1 Initial design for the Settings scene. 11
3.2 Initial design for the Local Game scene. 11
3.3 The first version of the Settings scene. 13
3.4 The first version of the Local Game scene. 14
3.5 The improved Settings scene. 16
3.6 The improved Local Game scene. 16
3.7 The final implementation of the Colour Options scene. 20

Chapter 1

Introduction

My project was recreating the board game 'Codenames' in an Android application with a strong focus on accessibility, to allow those with visual impairments to be able to play the game at the same time as those without visual impairments whilst not ruining the experience for either group.

The game can be simplified down to, "two teams compete by each having a 'spymaster' give one-word clues that can point to multiple words on the board. The other players on the team attempt to guess their team's words while avoiding the words of the other team" [2].

As the game is a primarily word based game, I thought that it would benefit greatly from having the accessibility features I planned to include, which were Text-to-Speech and being able to customise the colours of different elements in the app.

1.1 Motivations

I had three main motivations when it came to making this project, firstly was that the existing versions of the game, those being the physical board game and the web based version, weren't extremely accessible, and with 2.2 billion people worldwide with some form of visual impairment [3], I wanted to make a version of the game that could be played and enjoyed by more people.

Similarly to my previous point, I wanted those who played the game to not feel separate from those without visual impairments, and to feel as though they can play the game without relying on other people to change their play style to accommodate for their needs.

And finally I just wanted to make a fun game that can be played with friends and family

both locally and online, because what's the point of including these accessibility features if the game is boring and nobody wants to play it.

1.2 Aims & Objectives

The main aim of the project was to recreate the game Codenames as an Android application as closely as possible, whilst including the ability to play either locally with friends and family or online with random people too.

The secondary aim, was to try and make the app more accessibly friendly, specifically targeting those with visual impairment, without making the experience horrible for those without the impairments.

In order to achieve these aims, I set myself the following objectives that had to be met:

1. Create an Android application, supporting both a local and an online version of the game.
2. Create a server capable of passing the data between Android devices for the online version of the game.
3. Implement a Text-to-Speech system into the game.
4. Implement the ability to customise colours in-app.
5. Test the 'full' application and gather feedback.
6. Implement features/changes into the app based on feedback received.
7. Final tests on the full application.

Chapter 2

Background Research

For this chapter I will discuss some other games with accessibility features and how they implemented them to benefit their users. Following that, I will mention some of the tools I used in order to create my project and why I chose them.

2.1 Other solutions to this problem

Accessibility in games has become a more prominent feature in recent years, with studios having a much bigger focus on making their games as accessible as possible and whilst some of these games are very different from the one I've chosen to recreate, a lot of their features still apply.

2.1.1 Forza Horizon 5

At the 2021 Game Awards, Forza Horizon 5 won the 'Innovation in Accessibility' award. It was the first triple-A game to introduce sign language, "offering an alternative to subtitles for deaf and hard-of-hearing players." [4]

However, this wasn't the only accessibility feature they included, the game also allowed the player to 'slow down' their gameplay to allow for more reaction time. They also have an option to allow players to progress through the story without even completing the required challenges.

There are also various visual and audio settings such as language, control over multiple different volume levels, colourblind options and a high contrast mode. The subtitles are also quite customiseable, you can change the text size, background opacity and other options.

The game also supports both Text-to-Speech and Speech-to-Text as well as supporting a screen reader. The Text-to-Speech and Speech-to-Text features are used for communicating with other players you find in the online world, whilst the screen reader is used for reading out menu items as you highlight them.

For a racing game, these are some great accessibility features, combined with Xbox's own adaptive controller [5] this is a great step towards allowing anyone to enjoy the games they want to play.

2.1.2 Lego Star Wars: The Skywalker Saga

The accessibility features in this game focus more on just making the game more enjoyable and easier to play for those who use them, rather than completely changing the game to accommodate for them.

When you first launch the game you are presented with the option to view the accessibility options, which I felt was a very good idea and something I implemented into my own project. The game certainly isn't perfect for its implementation of accessibility features, for example if you choose to play the game with mouse and keyboard, navigating menus only responds to keyboard input, while mouse input may feel more natural, the game does not support it. Another feature that doesn't feel right on keyboard and mouse is flight controls.

"For Lego Star Wars: The Skywalker Saga accessibility, flying is one of the game's weak points. Controls feel rough, even on low sensitivity, and trying to keep my focus on the enemy ship or target during combat feels exhausting with lots of spinning and rotating" [6].

As for features the game does right, it features aim assist to make it easier to hit targets by slowing the movement of your reticle and the option to auto complete or entirely skip quick time events, which involve you pressing a single button over and over in quick succession.

Whilst the game does not have amazing accessibility support, it is another sign of games having much more of a focus on implementing them. As compared to previous entries in the Lego games series, it is the first to introduce proper accessibility settings [7].

2.1.3 The Last of Us Part II

Another game that won an award for 'Innovation in Accessibility' was The Last of Us Part II, this was the winner of the award in 2020 [8].

The game featured three different categories for its accessibility features, those being for vision, hearing and motor accessibility. Each containing a wide variety of options to benefit those who needed them.

Vision accessibility contained features such as Text-to-Speech, high contrast and the option to skip puzzles. The Text-to-Speech feature is more than just reading aloud text from items found in the game such as notes, but it also functions as a screen reader, to read the highlighted options in menus. The option to skip puzzles is also very nice as some puzzles can require you to look for hints which may be well hidden and difficult for those with visual impairments to find.

The hearing accessibility features are mostly indicators and cues for when events happen in the game, such as cues for when you enter combat or indications of how aware enemies are to your position. These features are very nice for a game like this, as in a lot of situations you need to avoid being detected so having a visual cue for that is very useful if you are unable to hear the audio cues.

Finally, the motor accessibility features are ways of making the game easier to play such as auto aim, auto pick up and infinite breath. These are all features that make the game easier to play, as you don't have to perform actions as quickly or have fast reactions in moments where they would normally be required [9].

Put together, this is another great example of how accessibility is improving in games.

2.1.4 Uno

Uno is one of the most famous card games out there, becoming the number one games property in the US in 2018 [10]. It's a very simple game and Mattel have done quite well to implement accessibility features to the game. However it is not done in the same way as other games by including them in the base game, instead if you want the accessibility version of the game, you'll have to purchase the specific version you want and they'll only support one or the other.

The first accessible deck they released was the colourblind deck in 2017 [11]. This version features the ColorADD system which represents colours as shapes such as triangles and lines and then surrounding those shapes in an empty box to indicate the lighter version of that colour [12].

The second deck was released in 2019 and this one featured braille on the cards to indicate what the card was [13]. This is a fantastic way for any blind players to still be able to play the game, as long as they are able to read braille.

The only real issue with these decks is that they are seperate, so if you were to play with someone who was blind and another person who was colourblind, you would have to make a compromise as the braille deck does not contain the ColorADD symbols and vice versa. If they were to release a deck combining both of those, it would be a near perfect deck for accessibility.

2.2 Tools used

These are the primary tools I used to create my project. I used other tools such as the programming environments, but those are very basic tools for a project and in some cases there wasn't a large difference in the choices.

2.2.1 Android

When it came to deciding what platform to develop the app for there were only really two options, those being Android and iOS. As you can see in the graph below, ten years ago there was a lot more competition in the market, with Android and iOS not even being on top, but in late 2012 that started to change with Android's rapid growth and iOS maintaining its hold of around 20-30% of market share. Ever since late 2015, Android has always held more than 64% of market share so that alone was a big enough reason to chose Android as the system to develop my project for.

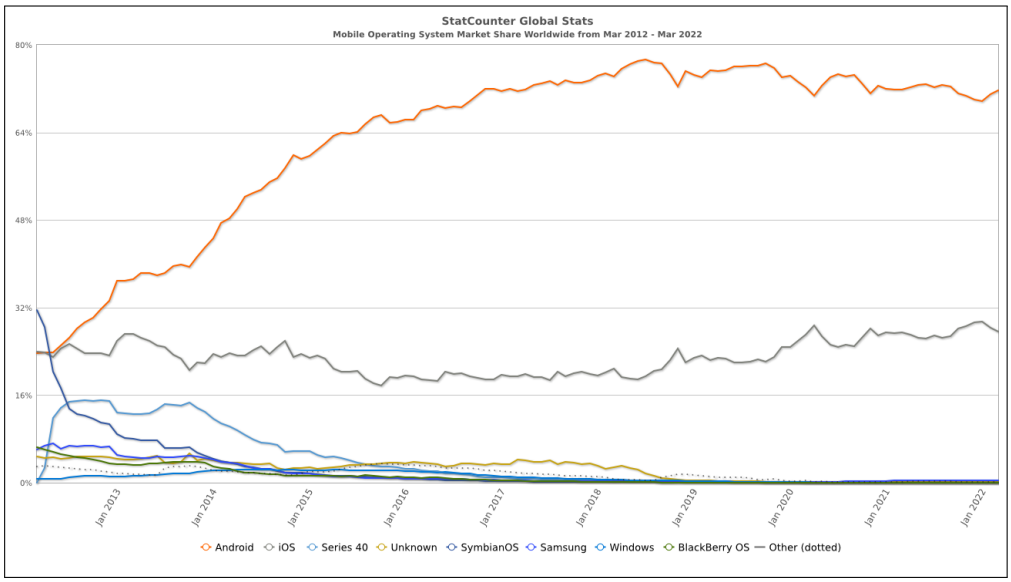

Figure 2.1: Mobile Operating System Market Share 3/2012 - 3/2022 [14].

Another reason why I chose Android as the operating system to use was because I was taking the CSC-306 Writing Mobile Apps module, in which we developed Android apps. Doing this module introduced me to many features of Android that I could implement into my project which I personally think improved the quality of the end result very well.

2.2.2 Kotlin

After choosing Android as the operating system to develop the project in, I was again presented with two options for the programming language to use. The two options that I had were Java or Kotlin.

As I was entering my final year at University, I had spent the last two years learning Java so originally I thought this would be the best language for me to write the program in due to this experience. However, because I was going to be doing the mobile apps module which was going to teach us Kotlin, it seemed like that could also be a possible option as doing that module alongside the project could improve my skill with Kotlin greatly.

In order to decide which language to use, I did some research to find the differences between the two languages and the advantages of each, when I came across an article making the bold claim in its title "Anything Java can do Kotlin can do better" [15]. I was instantly surprised upon seeing them create an object in around 50 lines in Java, only then to create the equivalent in Kotlin in only a single line. Seeing this, as well as other articles such as "Why I switched from Java to Kotlin" [16] and "5 Reasons Why Developers Choose Kotlin Over Java" [17] I decided to use Kotlin as my language of choice for the project.

Once I'd decided on using Kotlin for the project, I then had to actually learn the language and while my lectures in the Writing Mobile Apps module were helpful, there were a lot of things either not taught in the module, or they would be taught much later on whilst I needed to know how to do it earlier. So in the end, a lot of the things I learnt were self taught where I had to look at the documentation to get an understanding of how features worked or look at existing solutions to any problems I encountered.

2.2.3 Node.js

Since I wanted to make an online version of the game, that allowed people to play with those on other devices whether they were in the same room as you or not, I needed a way to get the apps to communicate with one another.

The best solution I found to do this was to use sockets, which is defined as "one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to" [18].

One way of implementing these into my app that I found was by using the socket class built into Kotlin, going with this approach however would involve the users having to share the host and port that they are connected to in order for others to connect too. A much better approach would be having a separate server that the app would already know of, so the only information you would need to connect with your friends would be the name of the room they've set up for you to join.

The first method I stumbled upon ended up being the one I went with, which was using Socket.IO, which is "a library that enables low-latency, bidirectional and event-based communication between a client and a server" [19]. Using the event-based communication allowed me to have features such as emitting a request to join a game from the client and then the server will listen for a request to join a game, which it could then handle to determine whether the user is able to join the game or not, and then emit a message back to the client which would be listening for an event to say if it could or could not join the game.

The downside to using this however, was that the server would have to be written in the Node.js web framework, which meant that I would have to use JavaScript. This language wasn't taught in any of my modules I'd studied so far at university and it wasn't going to be covered in any I was taking in my final year, so I had to learn it all myself. Fortunately the documentation on the Socket.IO website was very thorough so it wasn't too much trouble to write the code relating to listening and emitting events, however everything outside of that I had to learn for myself, by looking at documentation and other solutions to what I needed the server to do.

Chapter 3

Project Development

3.1 Software Life Cycle

For my project I followed the Scrum software life cycle. I felt that this was the best methodology for me to use as it allowed me to build up the project in stages, implementing a new feature and then testing it to ensure it worked properly before moving on to the next stage, which in a lot of cases required the previous stage to be working before I could move on to it.

For the "product backlog" of the scrum development cycle I listed all of the features I needed for the app and broke them down into small tasks. The following is a simplified version of the backlog I followed.

- Add all menu scenes with basic navigation.
- Add functionality to options in settings scene.
- Add instructions to how to play scene.
- Add restrictions to the game options page to only allow valid game rules.
- Implement full functionality to local game.
- Set up join game scene for when server is ready.
- Set up create game scene for when server is ready.
- Set up online game scene for when server is ready.
- Allow users to connect to the server.
- Allow users to create game rooms.

- Allow users to join rooms.
- Allow users to request game details.
- Implement users sending game functions, hints, guesses and chat etc.
- Add Text-to-Speech functionality to all buttons and text.
- Add Text-to-Speech functionality to the specific buttons in games with their rules.
- Add colour picker when clicking options in colour options scene.
- Update colours on all scenes if changed from default.

So I started at the top of this list and eventually made my way to the end, testing what I had implemented after every stage. The only exception to this was creating the online section of the app, as I had chosen to make it so that the online section would be inaccessible without a connection to the server, so until I created the server, I could not test it. Fortunately it did mostly work with a few changes after I had created the server.

Looking back, I think that this was a fine methodology to follow, but perhaps something like the iterative process may have been a better option as that is a bit more suited to building up a program in iterations as different versions, rather than just slowly putting the program together with scrum.

3.2 Android Application

The process of creating the Android application was definitely the most time consuming part of the project and not just in terms of actually writing all of the code for it, but I also spent a decent amount of time designing the layouts for all of the different scenes.

3.2.1 Initial Designs

Before I started the actual implementation of any of the different scenes throughout the app, I made a rough design for the layout of the objects in that scene. These designs were very useful as I'm not being particularly experienced at building layouts in XML, so having the rough design to follow certainly made it easier.

A lot of the layouts just had a large title at the top, with a back button in the corner and then underneath the title were any buttons to navigate to other scenes from the current scene. There were of course exceptions to this, such as the settings, colour options, game options

and local/online game scenes and below I've included my initial design for the settings and local/online game scenes.

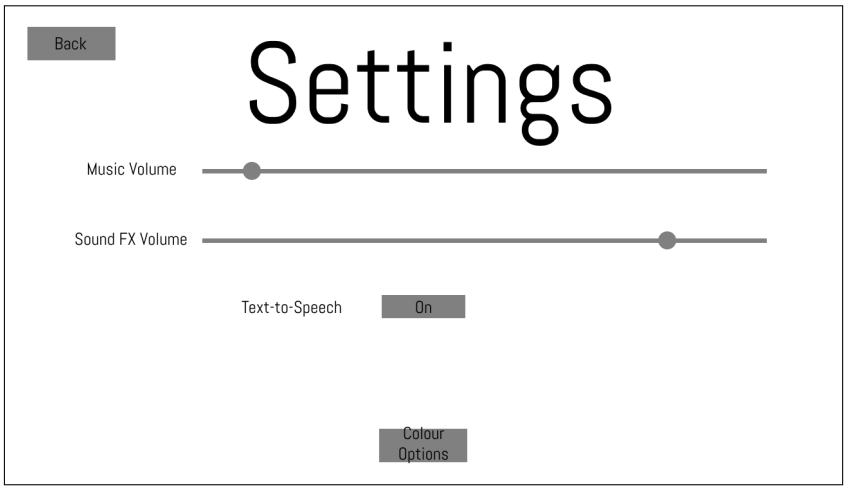

Figure 3.1: My initial design for the Settings scene.

Figure 3.1 is what I initially designed for the settings scene to look like. It is one of the more basic scenes, but contains some elements not found anywhere else in the app such as the sliders for the volume of the background music and sound effects. It also has a back button to return the app back to either the main menu or the initial start scene, depending on where it came from. The colour options button at the bottom will take the user to the scene where they can adjust the colours of different elements in the app. And finally the Text-to-Speech button will be a toggle for whether Text-to-Speech is enabled or not.

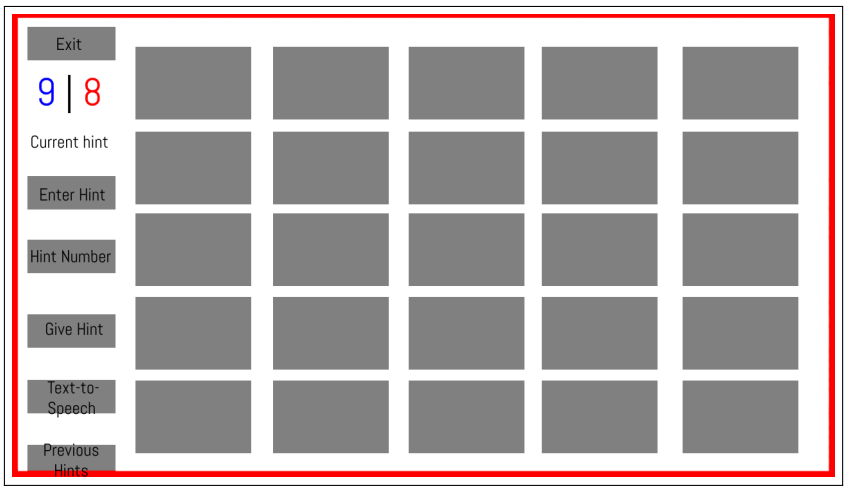

Figure 3.2: My initial design for the Local Game scene.

Figure 3.2 is my initial design for how the scene will look for when a user is playing a local, offline game. There are some elements missing such as the pop up box that would appear to give messages, but since that would block out most of the screen I've not included it in the design.

The left side of the scene is filled with the various game operations, as well as the exit button, words remaining for either team and the most recent hint. "Enter Hint" is not actually a button, but an entry box to type the hint in and similarly the hint number will be a drop down box to give the hint number, the other options are buttons. As they do not all fit onto the screen cleanly, these items would be inside a ScrollView, to ensure they can all be selected easily. The red outline also indicates which teams turn it is in the game, with team A being blue and team B being red by default.

3.2.2 First Version

I completed the first "full" version of the project in early January, a little earlier than I expected, which was good as it gave me extra time to work on any improvements I could think of from my self testing and also when it came to implementing feedback from my user study.

For the first version, I essentially went about recreating all layouts for all the scenes based on my initial designs, such as the one's shown in chapter 3.2.1. After all the layouts were complete, it was then about giving them functionality. A lot of the buttons were just navigating to different scenes but in some cases they had to pass data between those scenes. This was accomplished by passing extra data when I launched the new scene and then retrieving it on the newly loaded scene.

Once I could successfully navigate throughout the app I then had to add functionality to the less simple elements. I started with the settings scene, which required the volume and sound effects volume to be adjusted based on the value of the slider. And also the toggle button to enable or disable Text-to-Speech needed to function. Reading their values was no problem, but this was where I first encountered an issue. I needed their values to remain even after closing the app.

In order to do this, I did some research and found out about SharedPreferences which "points to a file containing key-value pairs and provides simple methods to read and write them" [20]. This meant that I could store the values for volume or anything else I needed to save and then whenever a scene needed to access that value, I could call a method to retrieve it.

Figure 3.3: The first version of the Settings scene.

Next was just adding the rules to the how to play scene which was very simple I just had to write them up. But following that I worked on the game options scene, which was what allowed you to customise the game experience. If you wanted to change which team went first, change how many bomb words there were or add some custom words into the mix, you would do that here. I had to ensure that any changes made to the game would still end up with a valid game, for example if you made one team have much less words than normal, you would need to designate those words to another category. Also, any custom words added could not be a duplicate of an existing word, so I had to read through the file containing all of the default words and check for duplicates.

After this I implemented the local and online game. I implemented the local game first as that would be the easiest and a lot of the logic could be reused in the online version with some changes. This was by far the most challenging part yet, as there were so many scenarios to consider, so it took a lot of thinking but I was able to implement the game in a way I was happy with.

Following the local game, all I had left to add functionality to was the online section of the game. I started with joining a room, which simply called the server to ask for a list of all of the public rooms and it would populate the ScrollView with all of the public rooms available to join. There was also the option to join a private room, which had a required password to join, so I had to handle if the password was incorrect.

After allowing users to join a room, I then set up creating rooms. This was very similar to creating a local room, you could go to the game options scene and customise the game, but you also had to set the name of the room and if you wanted to, you could make it private which

meant a password was required.

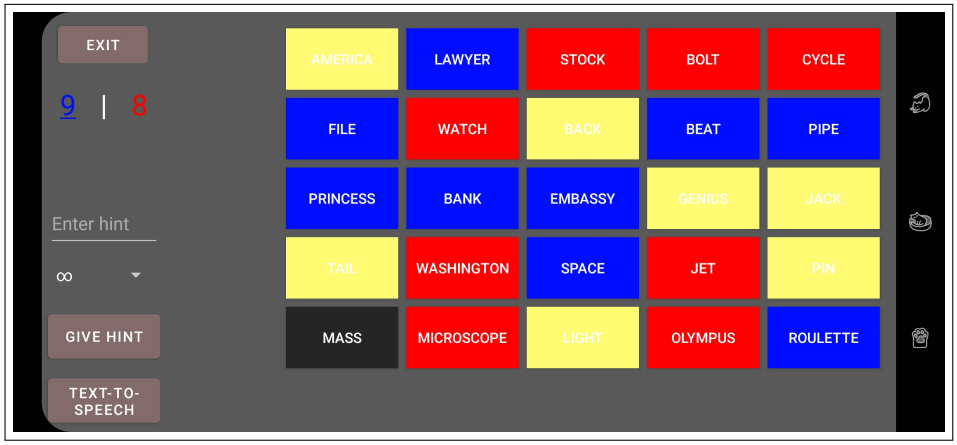

Figure 3.4: The first version of the Local Game scene.

Next came the most difficult part of the entire project which was implementing the online game. At first I thought it would be fairly straightforward and I could reuse a lot of code from the local variant, but unlike with the local variant, the person on each device would have a different experience. There were people on both teams and on each team there was a spymaster, that meant four possible different views on the same scene. A lot of logic such as whether hints were valid or not could be copied over from the local variant, but some things like after an incorrect guess was made, were very tricky to implement as each possible role in the game would react differently. So when I was implementing one feature, I had to think of how the other three roles would react to this change.

As I had not yet created the server, I was unable to test any of the online parts until later, which wasn't too much of a problem for the join and create room scenes as I was quite confident in those, but the online game scene was a whole different story.

The final features for the Android application were the accessibility features. I started with the Text-to-Speech features which involved making every element in the app that had text on it, read aloud that text when you performed a long press on it. This was fairly straightforward, but for some pieces of text I had to remove the new line character from the string to be read aloud as the speech would pause whenever it came across one.

After this, I then made the Text-to-Speech buttons in local and online games read aloud the correct words. As it gives you the choices of reading words from your team, the enemy team, words that have been guessed yet or not, etc. So for the local game it depended on what phase the game was currently in, as you don't want it reading out your teams words, whilst

in the guessing phase as that's an easy way to win. Similarly, with the online game it mostly depended on what your role was to decide what words would and wouldn't be read.

Once Text-to-Speech was complete I just had to add the colour picker to the colour options scene, there are eight different colour categories, so clicking each button would bring up a colour picker and you could pick whatever colour you wanted those elements to be. The choices are team A, team B, bomb square, neutral squares, unmodified squares which are for words that haven't been guessed yet, the application background, menu buttons and menu text. To accomplish this, I used a colour picker library I found, which accomplished what I needed very well. [21]

Finally, once the customised colours had been selected, I needed a method on every scene to update the colours of every element to what they should be. For most scenes this wasn't an issue however as per usual, the local and online games were a bit more problematic. However, using similar techniques as I used to solve the text-to-speech problems, it wasn't too much of an issue.

With all of those features implemented, the first version of the application was complete and I went on to review it myself and implement any improvements I could think of.

3.2.3 Self Improvements

Since I had a bit of extra time before I needed to conduct the user study, I decided to do some improvements to the first version. A lot of the changes I made were fixing some simple bugs I found but the main thing I decided to do was redesign the user interface as I wasn't happy with a lot of it.

Although for the most part I was happy with my initial designs, I don't think I did a very good job at recreating them in the app, so with the redesign it was mostly about recreating them more accurately, but in some cases I did make changes from the initial designs as now that I had some experience making layouts for mobile I felt as though I could make improvements from those designs.

One of the first things I changed on all scenes, was replacing all basic elements such as buttons and text with their MaterialDesign counterpart which is something I learned in the mobile apps module, these were just much cleaner looking versions of the default Android views in a lot of cases, in others it just allowed more customisation. For example, in the settings scene as shown in figure 3.5, the sliders to adjust the volume changed by having the 'ball' that you drag to adjust the volume be slightly bigger than the default slider, but also as

you move it you are able to display a label above it indicating what percent the volume level is at.

Figure 3.5: The improved Settings scene.

Other than just replacing elements with their MaterialDesign counterparts, for a lot of scenes I moved around some of the elements and made them bigger so that they are easier to interact with, and on the main menu I added images above the buttons to indicate a bit more clearly what their function is. Whilst all scenes had changes to them, I think the biggest change was the local and online game scenes.

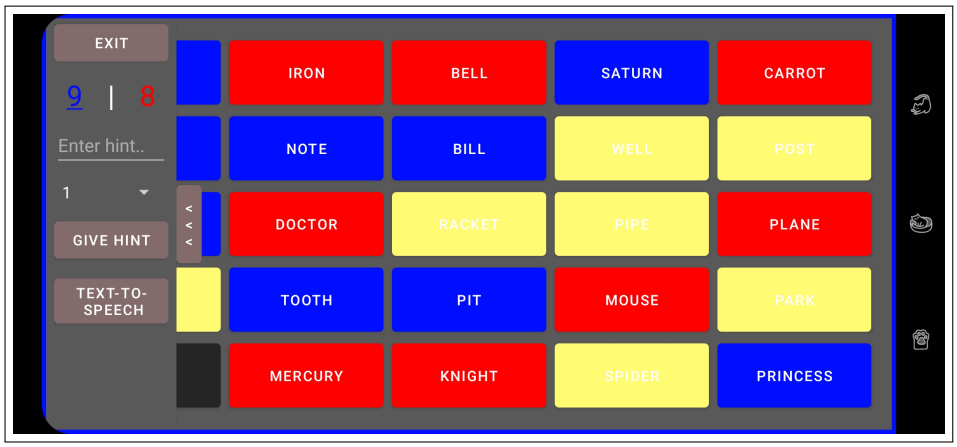

Figure 3.6: The improved Local Game scene.

The main change I wanted to make to the game screen was making the word buttons larger, but unfortunately I still needed the game operations menu on the left side, so I figured out how to add animations to the scene and was able to make it so that side menu could be closed and

reopened by tapping the arrow button to the right of it. Another change I made to this scene was adding the border around the edge to indicate which teams turn it was. I did have this in my initial design but forgot to include it in the first version, so I made sure to include it this time.

Overall, I was much happier with the user interface in this version of the app, so I was ready to get some feedback from others on the whole app to see what else I could improve.

3.2.4 Final Version

Based on feedback I received in my user study, I had a few features to implement into the app to make the experience better. Unfortunately, some of the suggested features would require large amounts of my code to be rewritten on both the Android application as well as the server. However, I was able to implement two of the requested features and also I made some changes to the suggested areas.

The first feature I worked on adding was the ability to create teams for a local game. This could be useful if you were playing with a larger group of people and maybe you forgot whos team you were on, or who the spymaster for your team was. Since I already had viewing teams in the online version, that was simple enough to implement, but the creation of teams had to be done from scratch. I ended up adding a button to the local setup scene, to set up teams, from there you can enter the spymasters name for each team, and list all of the members on each team.

The other requested feature that I added was having different saves of colour combinations. I implemented this by changing the reset button on the colour options scene, to open up a menu. From there you could save a colour set, load a colour set or reset the current selection. I allowed for three different saves as I felt that was enough, however if in the future I found a better way of laying out the scene and also a better way of storing the data, then I could allow more.

As for general improvements, I increased the size of buttons on some scenes as quite a few people mentioned that on some scenes they were quite small, whilst having a large amount of empty space on that scene.

Once those features were implemented, the final version of my Android application was complete.

3.3 Online Server

Working on the online server was the part I was most afraid of as I'd never used JavaScript before, but it ended up being quite similar to Java with its syntax so a lot of what I'd learnt in Java could be reused here.

The way I went about creating the server, was that a user would create a room, then their client would generate all of the details of the game such as the words to use in the game and which category they fall into. Then once all of this was generated it would send that to the server with the create room event.

Once the room is created, the host can then send the start game request however, until all the conditions have been met for a game to start, they will be sent back a start fail event. So the next step was to allow users to join the room. This just needed some verification to ensure that the user would be able to join the room, so for example they couldn't have the same nickname as an existing user in that room and the room had to actually exist. Once they were allowed to join they were added to the list of users.

Now that a user could join a room, they had to get the details from the server to be able to recreate it on their end, so they would send an event and the server would simply send them all of the details.

After that, it was then about implementing all of the gameplay features. These were all fairly simple on the server side as the client would send an event for the action they were performing such as guessing a word or giving a hint, the validation to make sure that it was an action they were able to perform is checked on the client, and then the server would send out an event to all players in the game to handle that event.

See appendix A for some of the implementations of the server code and appendix B for how the client handles it.

3.4 Accessibility Features

With accessibility being the secondary aim of the project, I considered a lot of different possible features I could include. These included, Text-to-Speech, Speech-to-Text, high contrast, customising colours and vibration.

3.4.1 Text-to-Speech

Text-to-Speech is the process of turning virtual text into speech. The idea that I had for implementing this into the app was making it so that if you performed a long press onto an element in the app containing text it would read aloud what that text was.

After implementing this into the app, I felt as though certain pieces of text could be read aloud automatically, such as when a hint is given to the players, or if any kind of information box pops up.

Since the app was still going to be used by those who wouldn't benefit from Text-to-Speech, I made it so that when the app launched for the first time, you would be presented with an option to disable Text-to-Speech, in case you never went to the settings scene to disable it there.

3.4.2 Speech-to-Text

Speech-to-Text is the opposite of Text-to-Speech, where you speak into the microphone of your device and it will turn that into virtual text. The places that I would use this in the app were few and it would mostly benefit giving hints and chatting in online games.

However, whilst this would be a good feature for those who may not be able to type on the virtual keyboard of their device, a lot of devices have this feature built in automatically, so I felt as though my time would be better spent on perfecting other features.

3.4.3 High Contrast

High contrast mode is the process of changing the colours of the user interface to make sure that there is a big enough contrast between all the different elements, to ensure readability.

The way I would implement this would be by making the background a solid black. With text being the complete opposite, white. Then menu buttons and the different team colours being different enough to ensure readability.

I never got to far into thinking with this feature as I decided it would be better to just allow the user full customisation on what colours everything would be.

3.4.4 Customising Colours

Customising colours was the second feature I fully implemented into the app and it was allowing the users to select what colour they want elements in the app to be.

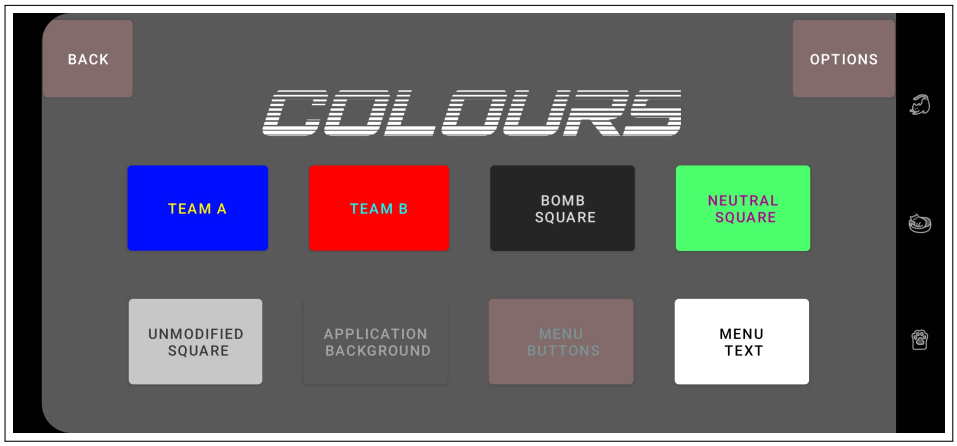

Figure 3.7: The final implementation of the Colour Options scene.

As you can see in figure 3.7, there are eight different categories for you to customise. The button colour is what indicates the selected colour, the text is simply inverted to make reading it easier. The currently selected colours are the default ones, and you can simply click the reset button in the options menu if you want to restore them.

3.4.5 Vibration

The final feature I intended on implementing was vibration, which I almost did include in the app but the places I was intending on using it would be when you made a correct or incorrect guess in a game, but with the combination of sound effects to get this point across and also Text-to-Speech reading aloud events mid game, I felt as though the job that vibration would have done was already being covered by those features.

3.5 Risk Analysis

Looking back at the risk analysis I created for my initial document, I was fortunate that I did not encounter many of these or any other risks that I had not considered, however there were two I had prepared for that I did encounter.

The first risk that I did encounter was quite a major one, as my working environment became unusable and I had to wait to replace some components before I could continue working. Fortunately I had planned for this and I had been putting my progress onto a private GitHub repository so when I was finally able to get up and running again, I had only lost a small amount of progress from when I had last pushed changes to the repository.

The other risk that I encountered was also a slightly problematic one which was that the server to connect clients wasn't working. I encountered this risk twice during the project, both times for different reasons. The first was a simple bug which I encountered during testing, the other was out of my control as the accommodation I was staying at whilst at University, was not allowing my devices to connect to the locally hosted server. Fortunately my plan of mitigation, which was to contact my supervisor and see if they could give support or knew someone who could, was able to present me with a solution by hosting it on a virtual machine.

3.6 Aims & Objectives - revisited

Looking at my original aims & objectives from chapter 1.2, I feel as though I have met all of these.

I had seven objectives that I needed to meet in order to complete the aims and I think that I was able to successfully complete them all. The first two involved me creating the game in an Android app and also creating the server to allow games to be played across multiple devices, which the game does support.

The next two objectives were implementing the accessibility features which I did and they both work as intended.

The final three objectives were about testing the application and implementing any feedback received from a user study I would conduct, which I did conduct and I did implement the feedback that I was able to in the time I had left.

As for my two aims. The first was to recreate the game Codenames as an Android application with support for local or online play and I believe that my app does that extremely well. The second aim was to make the app as accessibility friendly as possible, without harming the experience for those without the need for them. Given that changing the colours is completely optional and the Text-to-Speech option is presented on first launch for whether you wish to disable it, I think that it doesn't intrude on those with no need for them. As for those who need them, I think it certainly makes the game more playable than any existing version of the game.

Chapter 4

Testing & User Study

4.1 Testing

Following the scrum software life cycle, I performed testing after I had completed a designated section of the app following each sprint. In doing this, I uncovered small bugs in almost every stage, where I had forgotten to add a new layout to the AndroidManifest file which crashed the app on trying to load that layout or where I didn't assign a reference to a button properly so whenever I tapped it nothing happened. But those were simple fixes, there were certainly a lot more difficult issues I faced when testing the program.

The approach I took to testing, was looking at everything I had implemented in the current sprint, and then checking that every line of code was reachable and performed as expected. This approach definitely helped in a lot of areas where, for example, from the coding perspective it looked like an if statement has a case for every possible possibility, but then when I came to testing the actual app, I had sometimes missed a possibility.

Most of the significant bugs that I encountered were in the actual games, not the different menus in the app, however there were two bugs that were particularly tricky to solve. One was to do with the background music and how it would not resume playing if the app was minimised and reopened. This was due to the way I had implemented it, which was by making it a service, so when the app was minimised it would call the onPause method which I had set to pause the music, but upon reopening it the onStartCommand method was not called, instead the app would be calling onResume in the background but I had not given any extra functionality to this, so I had to tell it to resume playing inside of the onResume method.

The other bug that took a long time to fix was getting the app to connect to the server,

I had followed everything on the documentation correctly, yet for some reason it refused to connect. I eventually found the solution online which involved me adding a single line to the AndroidManifest file to allow clear text traffic to be used, which was required due to my server not being hosted on a secure network. If I were to actually release the app to the public, then I would host the server on a secure network so this wouldn't be required.

Below are some of the tests I performed on the local and online variants of the game. These are not all the tests I performed as there would be too many to list. All bugs that were found were fixed for the final version.

Test Description	Action	Expected Output	Actual Output
Check that hints with spaces are not allowed	Enter a hint with a space	Pop up message to say the hint is invalid	Hint accepted
Check that words cannot be pressed whilst in the spymaster phase	Press a word whilst in the spymaster phase	Nothing should happen	Nothing happens
Check that the Text-to-Speech reads the correct words	Tap the clicked words button mid game	Words that have been clicked should be read aloud	Words that have been clicked are read aloud
Check that clicking an incorrect word move to the correct phase	Tap a word belonging to the other team, or a neutral word	Phase becomes the opposing teams spymaster phase	Phase correctly changed, but the coloured outline to indicate which teams turn it is, does not update correctly.
Check that clicking the bomb word ends the game, with the opposing team winning	Tap the bomb word	Game ends in opposing teams victory	Game ends in opposing teams victory
Actions performed on one device correctly update on all others	Play test a game with 4 devices	Game should play out exactly the same on all devices	Game played correctly with a few issues. View teams box was missing spymaster symbol sometimes. Also, the turn action button would often appear for the wrong player.

Table 4.1

4.2 User Study

In order to get some feedback on the project from those not as familiar with it as I was, I conducted a small user study with a few people to see what improvements they could potentially suggest.

I let them play around with the app for 10 minutes and also showed them a video I had prepared of how the online mode worked and then gave them a form to fill out with some statements which they could then mark as strongly agree, agree, neutral, disagree or strongly disagree.

I split each section of the app that I was seeking feedback for into different sections so there was a user interface section, the how to play scene, local and online games were separate and finally accessibility features. Each section also had a box to type any improvements you had for that section.

Some examples of the statements are:

- The user interface is easy to navigate.
- Instructions are clear.
- Everything works as expected.
- Text-to-Speech volume is at a good level.

I received mostly strongly agrees to these statements with some agrees and a few neutrals, but fortunately no disagrees or strongly disagrees, so that meant that I had done a good job with the work I'd produced.

From the feedback I received, the user interface needed some buttons to be bigger, this was fairly easy to implement. The other suggestion relating to the user interface was making some of the text input boxes bigger. There are only a few places where text input is used, so I asked the participant if they meant any specific input box and they said the one for giving hints was the one they felt was too small. Unfortunately if I was to increase its size, then the side menu would cover even more of the screen which is something I didn't want to do, so I ended up leaving it as it was.

For the how to play scene, the only piece of feedback I received was splitting the game rules and tutorial into different sections, rather than have multiple paragraphs on the screen at once. This was quite a simple feature to implement, just changing text based on what button

was clicked. This also gave me the opportunity to rewrite this section and go into more detail on each specific section.

As for local games, I received three pieces of feedback. The first was an indicator for what to do next, originally I thought this would be a good idea, but after getting the response about improving the how to play scene, I felt as though if they were to read my updated version of that then this feature wouldn't be required. The next feature was the ability to change settings during a game, which is something that I agree would be a good feature to include, however due to the small impact the settings have on a game in progress and the fact that it would require a lot of code to be rewritten, I chose not to implement it. The final piece of feedback was allowing teams to be created before a game began, so that in case people forgot whos team they were on, it could be viewed in-game. This wouldn't be too hard to implement as I already had viewing teams set up in online games, I just needed them to be created.

Online games received the same feedback as local games except for the teams point, but it also received two extra pieces of feedback which were the ability for people to spectate the game. I don't agree that this should be a feature as I'm not sure why you would want to spectate a game with random players, and if you were spectating a friends game, there already exists ways of screen sharing with relative ease. The other point was allowing users to join a game in progress. I agree that this feature would be a good addition, however due to the way I had constructed the server and the online section of the Android app, this change would require a lot of rewriting existing sections and I wouldn't have been able to complete it in time. For similar reasons, the suggestion to allow settings to be changed during the game, that was suggested for both local and online, could not be implemented without rewriting a lot of the existing code.

The final feedback area was the accessibility features and the feedback points were having a standard set of colour combinations, so the default and then other variations for perhaps a lighter theme rather than the default dark and also allow users to save their own combinations if they wanted to swap between them. I don't think having a set of standard colour combinations is a worthwhile feature as giving the user freedom on what colours they want allows them to change any colours they may not be happy with, so if they wanted a lighter theme, they can just change the background colour. As for saving combinations, this wasn't too much of an issue to implement, I capped the number of saves at three, as I felt as though you wouldn't have much use for any more than that.

Chapter 5

Reflection and Evaluation

5.1 Final Product

Overall, I'm very happy with the project I created. Not taking the accessibility features into account, I think my Android application reflects the original game extremely well, and with the added features of custom words and allowing more words to be dedicated to specific categories so that one team has a handicap or making more bomb words to increase overall challenge. With the inclusion of those extra features I think it makes the game more enjoyable in the long run as if the game becomes too easy for you, then there exists the options to make it more of a challenge.

As for the accessibility features, I think the ones that I implemented were done quite well, I think the colour picker is quite easy to use and with it offering a very large number of possibilities, there should be no problems with people disliking the colour theme of the app as anything they dislike can easily be changed. Then the Text-to-Speech, I think the way that I used it was quite smart, making it so that a long press action on anything with text would read it aloud, then in an actual game having it read out specific words, so if you wanted to know what words are left to be picked, or what words you have already guessed etc, you could do that.

Even though I'm happy with what I created, I definitely know there is room for improvement and given the opportunity I'd love to continue work on it and make it as good as I know it could be, but I'll talk more about those improvements in chapter 6.

5.2 Reflection

If I were to do the project again, I think I would use a different software life cycle, that being the iterative process as it just seemed to be more appropriate compared to the way I was using scrum. But as for the actual application itself, I would spend more time creating designs for the user interface, as with the existing project, it's the only part I'm not fully happy with so spending more time designing how they should look is definitely something I would like to do.

For the server, I'll go into more detail in chapter 6.2 but essentially I would make it so that the server creates the game and clients just send and receive updates from it. To go along with this, I would also make joining public games in the app a better experience by allowing you to filter by certain game rules, or see how many people are already in each lobby.

As for accessibility features, I would probably try to find a different way of selecting colours, as whilst the current colour picker functions perfectly fine, it is very basic, and I'm sure that either a better library exists, or given enough time I could create one myself without having to rely on an existing library.

Chapter 6

Future Work

6.1 Android Application

If I was to work on the application more there are two big things I would like to do. Firstly, I would redesign all of the layouts in the app to be a lot cleaner and modern, as currently it looks very basic. I would have preferred to have done this already but unfortunately it's not something I am particularly skilled at, and it would have taken a lot of time to design them to look as good as I imagine.

6.2 Online Server

For the server, it currently works by having the client build the game on their end, and then the server just passes information to each client connected to that room to progress the game. I think a better way to do it, would be to have the game be constructed on the server and it stores the progress of the game locally, that way I would be able to allow players to rejoin if they had disconnected or allow somebody else to become the host after the original host quit instead of closing the room.

6.3 Accessibility Features

An accessibility feature I would like to implement would be a "full accessibility" mode, which I've only thought a little about so it may not be particularly feasible, but I've imagined it as having one side of the screen respond to swipe movements such as swiping up, left, right or down to navigate menus with the newly selected option being automatically read aloud and the

right side of the screen responding to touch input, whether it be a short press or a long press to perform different actions.

This would ideally be a way for people with little to no vision to still play the game, however this may not be a particularly enjoyable way to play the game, so as I said above it may not be feasible. It may also not even be possible to do in the current environment I've built the app in and I may have to swap to a different environment such as Unity.

I would also like to introduce different Text-to-Speech voices, as currently it only supports one British voice, but the Text-to-Speech engine supports many others, I would just have to implement a method of selecting one.

I could also revisit some of the accessibility features I originally looked into in chapter 3.4 and see about implementing them in a better way.

Chapter 7

Summary

To summarise, accessibility in games is something that is very important, especially with the growing amount of people suffering from these disabilities. Playing games is a great way to reduce stress [22] and for those who get stressed about how their disability will affect their future, playing a game that does everything it can to ensure a smooth and fun playstyle to fit their needs is much more likely to help them, than one that doesn't care about them and does the bare minimum to try and accommodate for them.

The game I created is much simpler and is a lot easier to implement accessibility features for than a lot of the games releasing today. But if I was able to create this game and implement the accessibility features I did to the level of quality that I did in just a few months, then why can't the giant triple-A studios implement them into their games when they have years to develop them.

Bibliography

[1] (2021) Number of gamers worldwide 2021 | statista. [Online]. Available: https://www.statista.com/statistics/293304/number-video-gamers/

[2] (2017) Codenames (board game). [Online]. Available: https://en.wikipedia.org/wiki/Codenames_(board_game)

[3] (2021) Vision impairment and blindness. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment

[4] (2021) Forza horizon 5 wins innovation in accessibility award for sign language implementation. [Online]. Available: https://www.thegamer.com/forza-horizon-5-accessibility-sign-language/

[5] Xbox adaptive controller | xbox. [Online]. Available: https://www.xbox.com/en-GB/accessories/controllers/xbox-adaptive-controller

[6] (2022) Lego star wars: The skywalker saga accessibility review — can i play that. [Online]. Available: https://caniplaythat.com/2022/04/20/lego-star-wars-the-skywalker-saga-accessibility-review-can-i-play-that/

[7] Lego series accessibility report - android, mac, pc, ps4, switch, xbox one and ios - family video game database. [Online]. Available: https://www.taminggaming.com/en-gb/accessibility/Lego

[8] (2020) The last of us part 2 dominates at the game awards. [Online]. Available: https://www.gamesindustry.biz/articles/2020-12-11-the-last-of-us-part-2-sweeps-the-game-awards

[9] The last of us part ii - accessibility. [Online]. Available: https://www.playstation.com/en-gb/games/the-last-of-us-part-ii/accessibility/

[10] (2018) Uno® holds title as #1 games property in the united states. [Online]. Available: https://corporate.mattel.com/news/unoR-holds-title-as-1-games-property-in-the-united-states

[11] (2017) Uno® introduces the first card game for the colorblind. [Online]. Available: https://corporate.mattel.com/news/unoR-introduces-the-first-card-game-for-the-colorblind

[12] Coloradd - coloradd code as a color identification tool for colorblind people. [Online]. Available: https://www.coloradd.net/en/coloradd-code/

[13] (2019) Uno® introduces first official braille deck. [Online]. Available: https://corporate.mattel.com/news/unoR-introduces-first-official-braille-deck

[14] (2022) Mobile operating system market share worldwide | statcounter global stats. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide

[15] Z. Westlake. (2017) Anything java can do kotlin can do better. [Online]. Available: https://medium.com/pinterest-engineering/anything-java-can-do-kotlin-can-do-better-a1c1ddae8ffd

[16] S. Brown. (2020) Why i switched from java to kotlin. [Online]. Available: https://opensource.com/article/20/6/java-kotlin

[17] D. Misal. (2019) 5 reasons why developers choose kotlin over java. [Online]. Available: https://analyticsindiamag.com/5-reasons-why-developers-choose-kotlin-over-java/

[18] What is a socket? (the java™ tutorials > custom networking > all about sockets). [Online]. Available: https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

[19] (2022) Introduction | socket.io. [Online]. Available: https://socket.io/docs/v4/

[20] (2021) Save key-value data | android developers. [Online]. Available: https://developer.android.com/training/data-storage/shared-preferences

[21] Github - yukuku/ambilwarna: Android color picker aka ambilwarna library ("pick a color" in indonesian). [Online]. Available: https://github.com/yukuku/ambilwarna

[22] K. L. Do video games reduce stress? (backed by research) | healthy gamer. [Online]. Available: https://www.healthygamer.gg/blog/do-video-games-reduce-stress

Appendix A

Examples of server code

```
1  //Creates a game room.
2    socket.once("createRoom", (user, roomName, password, allWords, bombWords,
         neutralWords, teamASquares,
3                    teamBSquares, startingTeam) => {
4      //Sets up the room with all details
5      let newRoom = { roomName, password, users: [], closed: false, started: false
           , teamAUsers: [], teamBUsers: [],
6        teamASpy: undefined, teamBSpy: undefined, allWords, bombWords,
             neutralWords, teamASquares, teamBSquares,
7        startingTeam };
8
9      //Checks that a room with this name doesn't already exist.
10     if (rooms[roomName] !== undefined) {
11       socket.emit("createFail", "The room " + roomName + " already exists.")
12     } else {
13       //Creates room.
14       rooms[newRoom.roomName] = newRoom;
15
16       console.log("CS: " + user + " created room: " + roomName + ". Password: "
             + password);
17
18       //Adds host to list of users.
19       socket.join(roomName);
20       newRoom.users.push(user);
```

Listing A.1: Code for creating a room in the server.

```
1   //Used for when a WordButton has been pressed so that all users can update their
        game status.
2     socket.on("wordButton", (word, username, roomName) => {
3         io.to(roomName).emit("wordButton", word, username);
4         console.log("CS: Word " + word + " selected in room " + roomName + " by "
              + username + ".");
5     });
6
7     //Used for when a user chooses to end their teams turn.
8     socket.on("endTurn", (roomName) => {
9         io.to(roomName).emit("endTurn");
10        console.log("CS: Turn ending in room " + roomName + ".");
11    });
12
13    //Used for when a spymaster gives a hint to their team.
14    socket.on("hint", (hint, roomName) => {
15        console.log("CS: Hint " + hint + " for room " + roomName + ".");
16        io.to(roomName).emit("hint", hint);
17    });
18
19    //Used for sending chat messages to the room.
20    socket.on("chat", (user, team, message, roomName) => {
21        console.log("CS: Message from user " + user + " on team " + team + ": " +
              message);
22
23        switch(team) {
24            case "A":
25                //Emits to the room that a message from Team A has been sent.
26                io.to(roomName).emit("teamAChat", user, message);
27                break;
28            case "B":
29                //Emits to the room that a message from Team B has been sent.
30                io.to(roomName).emit("teamBChat", user, message);
31                break;
32        }
33    });
```

Listing A.2: Some of the event listener and emitters for online games.

Appendix B

Example of client code

```
1  //When game is valid and has begun
2          SocketConnection.socket.on("startSuccess") {
3              runOnUiThread {
4                  //Hide buttons only used during setup
5                  startGame?.visibility = View.GONE
6                  changeTeam?.visibility = View.GONE
7
8                  //Set the correct starting phase
9                  gamePhase = if (startingTeam == "A") {
10                     OnlinePhase.TEAM_A_SPY
11                 } else {
12                     OnlinePhase.TEAM_B_SPY
13                 }
14
15                 //If player is spymaster, disable their chatting options and set
                       text of the turn action button
16                 if (player?.isSpymaster == true) {
17                     turnAction?.setText(R.string.give_hint)
18                     chatEdit?.visibility = View.GONE
19                     sendChat?.visibility = View.GONE
20                 } else {
21                     turnAction?.setText(R.string.end_turn)
22                 }
23
24                 //If player is spymaster and it's their teams turn, show their
                       options
25                 if (player?.isSpymaster == true && player?.team == "A" &&
                       gamePhase == OnlinePhase.TEAM_A_SPY) {
26                     editHint?.visibility = View.VISIBLE
27                     hintNumber?.visibility = View.VISIBLE
28                     turnAction?.visibility = View.VISIBLE
```

```
                }

                if (player?.isSpymaster == true && player?.team == "B" &&
                    gamePhase == OnlinePhase.TEAM_B_SPY) {
                    editHint?.visibility = View.VISIBLE
                    hintNumber?.visibility = View.VISIBLE
                    turnAction?.visibility = View.VISIBLE
                }

                //Update the hint spinner and colours of word buttons
                updateSpinner()
                updateWordColours()
            }
        }
```

Listing B.1: Code for when a game has been started successfully.

```
startGame?.setOnClickListener {
            buttonClick?.start()

            SocketConnection.socket.emit("startGame")
        }

        requestSpymaster?.setOnClickListener {
            buttonClick?.start()

            SocketConnection.socket.emit(
                "requestSpymaster",
                player?.nickname,
                roomName,
                player?.team
            )
        }

        changeTeam?.setOnClickListener {
            buttonClick?.start()

            if (player?.team == "A") {
                SocketConnection.socket.emit("chooseTeam", player?.nickname, "B",
                    roomName)
            } else {
                SocketConnection.socket.emit("chooseTeam", player?.nickname, "A",
                    roomName)
            }
        }
```

Listing B.2: Some emitters used by clients to start game, request spymaster and change team.