Logo

Description automatically generated

Icon

Description automatically generated

CTEC3451 Development Project

**“Creating a Control Mapping Program for Game Accessibility”**

Final Deliverable

Project Supervisor: Jethro Shell

Project Author: Ruya Kumru-Holroyd (P2512547)

Word Count: 0000

Contents

[Acknowledgements 3](#_Toc103801283)

[Introduction 3](#_Toc103801284)

[Background 3](#_Toc103801285)

[Main Body 3](#_Toc103801286)

[Development Cycle 3](#_Toc103801287)

[Basic Functionality 3](#_Toc103801288)

[System Design 3](#_Toc103801289)

[Underlying Data Structures & Algorithms 3](#_Toc103801290)

[User Interface 3](#_Toc103801291)

[Testing 4](#_Toc103801292)

[Critical Evaluation 4](#_Toc103801293)

[Project Evaluation 4](#_Toc103801294)

[Evaluation of your approach 4](#_Toc103801295)

[Evaluation of tools used: 4](#_Toc103801296)

[Final paragraph 4](#_Toc103801297)

# Acknowledgements

To begin this project report, I would like to give thanks to people who helped me along the development process and my time at De Montfort University, this includes all the lecturers and lab tutors who have provided me with a variety of knowledge and skills, throughout all the different modules. They have also been incredibly helpful when I have asked for their assistance.

I would also like to thank Dr Jethro Shell, my supervisor for this Development Project, who has guided me and helped me throughout since the start of this project, as well as helping to come up with the project itself. And generally, being supportive of my project and its themes.

Finally, I would like to thank my friends and family for their support over the years at De Montfort University. My family who has supported me and been there for me throughout, my housemates who provided encouragement and finally a massive thanks to the friends that I have met on the Computer Games Programming course, who have shared their knowledge and help in times of need, and also given fun and laughter through working together whether in person or on a Discord server, providing me the confidence and motivation in completing this project and other modules.

# Abstract

There are millions of video games available worldwide, but how many are actually accessible and well represented? The answer is less than you might expect, which is why this project was created to demonstrate that it is possible to do so and that it can be used and altered for future usage.

This report details the development of a control mapping program for game accessibility, as well as the two prototype games created to test the control mapping program, including the system design of the project, the development cycle, the functionalities, user interfaces and the testing of the project. Finally, providing a critical analysis of the project to conclude the Development Project.

The project was created using Unity3D version 2020.3.12f1.

# Introduction

## Background

The main product of this project is a Control Mapping Program for Game Accessibility, it maps the control settings from one game and loads it to another. Included with the project, are two Prototype Games in which I created to test and use the mapping program between; “Wheelchair Basketball Shoot”, a 2D basketball shooter, and “Bye-Bye Thoughts”, a 3D first person shooter.

The main functions of this project, is by loading one of the prototype games, navigate through its’ menu and to the ‘game settings’ scene where it allows the player to select and change their key bindings and what type of input device they are using, whether it be keyboard and mouse or a controller, and then save these changes which saves to a json file in an accessible file for both prototype games. Then, the player can load the other prototype game, go into its game settings after navigating through the menu and press load and it loads the changes in key bindings to this second prototype game. Unfortunately, the keyboard and its key bindings are the only input device that can be currently mapped and saved, but the use of a controller input in the second prototype is possible, the first one does not use a controller due to the simple controls.

Timeline

Description automatically generated with low confidence*Graphical user interface

Description automatically generated*Diagram

Description automatically generated

*Figure 1: Navigation of the first prototype*

*A screenshot of a video game

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generatedA screenshot of a video game

Description automatically generatedA screenshot of a video game

Description automatically generated*

*Figure 2: Navigation of the second prototype*

## The need for the project

Such a program and its prototypes are needed because both of these games include representation of disabilities within them to deliver the importance and the ability to include these in video games, as the Games Industry lacks this with only a handful of games that include appropriate representation. The same goes for accessibility in games, it is usually an after-thought in the development in video games, which is why a program such as this is necessary, so that the developers can include it into their games with ease, which allows them to improve on the accessibility their game provides and may even give them further incentives to include more accessibility options and disability representation. It was also a chance to develop personal skills and become more familiar and confident in creating games using Unity 3D.

## Objectives

The aim of this project is to create a program that will map the specific control settings and its key bindings, off a prototype game and transfer them onto a different prototype game, to allow for accessible gaming. And to develop two prototype games and to include disability representation within them to ensure the possibilities and ability to do. As well as being accessible with the ability to set key bindings and save the changes in the settings using json and load the saved changes. The software that is created will be well constructed and work as intended. Furthermore, another objective is User Testing by gaining primary data from a group of individuals in testing the two prototype games and mapping their selected input decisions and key bindings and output it. Finally, to document the process in the form of a project development report.

# Main Body

## Development Methodology

The development lifecycle of this project, started with planning on what the project would be

The methodology for development applied to this project was the Agile Methodology. This allowed the components of the projects development to be broken down into a number of stages, completing each stage before moving onto the next, making sure that all the requirements had been met and there were no bugs or issues in the functionality of that stage. These components would be broken down based on the size of each task, and these become the focus of each sprint. Choosing Agile Methodology was beneficial for this project, as there was already a clear split of the tasks, with the control mapping program itself and the requirements of two prototype games needed for testing, these tasks were already large components and were focuses of each sprint. Agile methodology also allows for breaking up the tasks within each component further, such as within making one prototype game there is many tasks that one would focus on such as designing and implementation, before completing, testing, and moving onto the next stage.

A Gantt chart was created at the start of the project to split up the tasks and plan the development time for each one. The Gantt chart which can be seen in Figure 3, can show that the timing of some tasks took place later than expected or took longer than originally had planned. Although, it was accurate in the planning of the order in which tasks would be executed. The implementation of the control mapping program took a lot longer and later than expected, this was due to there still being small fixes to the prototype games that had to be completed before moving onto that stage, and also due to outstanding circumstances such as illnesses, jobs, and other university module commitments. The time management could have been a lot better, and if it had been then some of the tasks that were implemented later than planned, could have been on track with the Gantt chart.

## Basic Functionality

In the first report for this project, the functional requirements were set out and identified, this was done by constructing a Use Case Diagram (UCD) for the main sections that required their own functional requirements and use cases. There were two made, one for the main menus for each of the prototype games, with its own requirements and path, and the second being the game prototypes functional requirements themselves. Figures 4 and 5 show these.

*Diagram

Description automatically generatedDiagram, schematic

Description automatically generatedFigure 4: Use Case Diagram for the Main Menu scenes*

*Figure 5: Use Case Diagram for the Game Prototypes*

These diagrams show each functional requirements in the two game prototypes, the control mapping program itself is included in the main menu as that is where it is saved and loaded, within the game settings which is only accessed through the main menus.

In description, the way that the Player can navigate through the games and use the control mapping program, is by running one of the prototype games, for example Wheelchair Basketball Shoot, the player opens up first to the main menu which includes three buttons; play game, game settings, and exit game. Pressing play game takes the user through the Wheelchair Basketball Shoot game play, there is a back button that will take the user back to the main menu once they are finished with the gameplay. Pressing game settings brings the user to a settings page which includes the buttons and key bindings for the game. There is also save and load buttons which allows the user to save their key binding preferences, which is done by pressing one of the key binding’s buttons, pressing again with the new key that the player would like to be bound to the action, which is described next to the button, and that sets the new key bind and by pressing save this will save it to a json file that can then be loaded either in this prototype or the other prototype Bye-Bye Thoughts.

## Control Mapping Program

### Analysis of requirements

The Control Mapping program is designed to help with game accessibility, capturing the mapping of keyboard controls, how these mappings can move from two different playable prototype games and help benefits disabled players gaming experiences, making games more accessible and easier to play. The player is supplied with two playable prototype games that include their own menu and game settings sections where they can save their own key bindings or load already previously set ones from another game, which are saved by using json and a created a json file that is accessible for both games using the same directory path, as the scripts create a folder called ‘SaveData’ with a json file inside, which is in the users Documents file on their own computer.

Controls, per game prototype, are relatively simple and easy to understand which is important due to the fact “if they are too difficult to learn or if the game experience becomes uninteresting then you’re creating barriers that only skilled and patient players will be able to overcome,” (Carrera, 2016). This is especially necessary when considering accessibility in games, as players won’t be able to play the games if they are difficult for them or unable to control.

### Design

To keep the control mapping program simple, easy to understand and accessible for all, the code itself is within the back-end of the project, and the front-end where it is accessed is the two prototypes’ games menu/game settings UI which can change the key bindings and save and load the keycodes from the json file.

The ‘program’ itself is a collection of scripts made using C# language and Json. There is a save manager that includes the write and read functions to and from the json file, as well as creating the directory in which it is held and there is a save object script, which is slightly different between the two prototypes as it holds the key codes needed in each of its game settings scene. Those two scripts bring the essence of the control mapping program, but the game settings and game manager scripts were also essential in putting it together. Both prototypes have all 4 scripts so theoretically by taking these scripts and including them in another game, just by editing the game settings and game manager for that game, if a basis script was created it could be more executable as a program. However, that is not something that was created in this project, but it is something that would be developed in the future.

### Implementation

When first developing the control mapping program, the first step was to create key bindings in a prototype game and make it able to switch and save so that the player could choose any key they would like to play with. So, when first saving these key bindings so that they would be set and playable, PlayerPrefs was used instead of json. This was because when researching and testing ways to change the keys a tutorial included PlayerPrefs which saves key codes in the computer’s registry. This meant, that when the prototype game was closed and then opened again, any changes in the key binding had been automatically set. This did work very well, however it was not ideal when it came to mapping the key changes from one prototype to another, because as mentioned earlier, PlayerPrefs saves the key codes to the registry which is not an easily found or accessible directory. Although it is possible, it was not suitable for the project at the time, and json allowed for an easy creation of a directory in the users Documents folder which is more accessible.

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated  
*Figure 6: Game settings menu in second prototype, with save and load and changeable key buttons*

## Prototype 1: Wheelchair Basketball Shoot

### Analysis of requirements

The first prototype, titled “Wheelchair Basketball Shoot” is a simple as the name suggests, it is a 2D basketball shooter game where the player aims and releases a ball to a basketball net, gaining points for each shot. The major components of this prototype were to have a functional game with player movement controls and gameplay, include representation. And most importantly the game settings menu where the player can set and change the key-bindings for these controls.

The representation of disability in games, for this prototype, as explored within the Literature review in the first report, has come from (Brody, 2020) AbleGamers article about the need for more disabilities in the games we play, where it was suggested that sports games can be adapted to include disabilities such as basketball games can have a wheelchair basketball player. So, it was decided to create this prototype from this idea and showcasing the possibility of including representation this way.

### Design

There are 2 levels currently built in the prototype, the second more difficult than the first, but there would be multiple levels if this game were to be built to completion. The default keys set to control the player are the keys A and D on a keyboard, to move left and right respectively on the screen, the ball follows as the player moves. The ball can be aimed and released by using the mouse, holding down the left mouse button and dragging down on the screen to aim the ball into the basketball net. The strength at which the ball is fired at depends on the speed and duration of the mouse button being held down. It is possible to invert the y position of the ball drag with a mouse, instead of top to bottom you can change it to bottom to top in the game’s settings. There was a planned line renderer to be added to the game prototype but due to lack of time and needing to create the mapping program, this was not added.

### Implementation

The action needed to aim and shoot the ball using the mouse, by dragging on the screen and aiming at the basketball net, is accessible because the player can start and end the drag position at any point on the screen in their preferred way, instead of a specific control area. There is also an option to use a key and hold it down and it will projectile in the direction of the basketball hoop the strength depending on how long the key is held down for. Although, this option does not currently work within the prototype. It has been tested that if it is selected and a key has been selected, that key is saved in the json files and can be loaded and will translate in the game but currently if it were to be selected, nothing would happen as it has not been implemented. This was added mainly to test json files and also to be a future functionality of the game for a possible future completed version of the game, past the prototype.

There are two levels included the first being a simple throw and shoot, the second the basketball hoop moves adding some difficulty, as these games are prototypes, only two levels were necessary as it was enough to test the robustness of the game and its controls.

Diagram

Description automatically generatedDiagram

Description automatically generated  
*Figure 7: Screenshots of Prototype 1 game, Level 1*

Diagram

Description automatically generatedDiagram

Description automatically generated  
*Figure 8: Screenshot of Prototype 1 game, Level 2*

All the art assets within this prototype were all created by using a sprite draw website called Piskel, (Piskel - Free online sprite editor, n.d.), the animations were also created in the Unity editor.

Logo

Description automatically generated Logo

Description automatically generated  
*Figure 9: Character art and animation stages of staionary and ball thrown in Prototype 1*

## Prototype 2: Bye-Bye Thoughts

### Analysis of requirements

The second prototype game is titled “Bye-Bye Thoughts”, it is a simple 3D first person shooter which represents mental health. The player works their way through a 3D environment which is darkened, and they must go around and shoot the ‘Bad Thoughts’ away before it consumes them and for each ‘Bad Thought’ the player shoots the scene becomes brighter until it is normal daylight and all the thoughts had been expelled. There was also going to be a timer so if the player did not shoot all the targets in time, the screen would fade to black and a game over scene would appear. However, this was decided to be something that would be implemented in the full game rather than the prototype version.

### Design and Implementation

The player can be controlled by default with a Keyboard and Mouse, using WASD respectively for Forward, Left, Back and Right. Space bar set for jumping, Left Shift which if held down the player would run, and the mouse left click button to shoot. These can all be changed in the game settings menu with the keyboard drop menu selected and all the respective buttons. The shoot button can also be chosen, such as if the player would like to use a keyboard button to shoot, they could set any key or leave the default Left Control key just by pressing the button that does not say mouse, if they press the one labelled mouse, shooting will be controlled with the mouse left click. If the controller drop menu is selected, the players controller will be able to be used to control and shoot, however it is not currently possible to rebind the controller buttons.

As it is a 3D game, there is a 3D camera that is essential to the game, a follow camera is used to create the perception of a first-person shooter, with a pistol in the front view, attached to the player and camera so that it moves with the two as they navigate through the 3D world.

The environment looks quite simple and very fitting as a prototype, there was effort made to make it look somewhat aesthetically pleasing, by testing and making a terrain with some bumps and mountains, and very simple shootable objects that simply have the text ‘bad thoughts’ on them. For the fully developed game these would be cloud shaped bubbles that would have texts of actual bad thoughts that a person with depression could have and they would dissipate when the player shot them.

Each ‘Bad Thought’ object is spawned using a script, where the number of spawned objects can be set in the Unity Editor, as well as the radius for how far out and up they can spawn and the centre of the spawn point. Further screenshots can be found in Figure 11.

A screenshot of a video game

Description automatically generated with medium confidence  
*Figure 10: Screenshot of Prototype 2 game*

## Underlying Data Structures & Algorithms

-original key bindings

-Json

-Show player prefs stuff

-Physics for prototype 1 game

-Gameplay for prototype 2

Some of the underlying data structures and algorithms in this project are the json files with the C# save objects and save manager scripts, the way the inputs were set to each action from the game manager and game settings scripts to the player movements and first-person controller in the second prototype.

## User Interface

### Analysis of requirements

Due to the two games being prototypes, the User Interface is relatively simple in its design and functionality. That being said, The user interface is critical since the mapped control data is saved and loaded through the game settings in each game prototype, so it must be accessible and simple for us. It must be completely functioning and able to interact with items in game prototypes as well as for control mapping.

### Design

The usage of text and buttons, as well as a dropdown menu in the second prototype to switch between keyboard and gamepad inputs, is unique to this project. The text and buttons are used to configure key bindings and button mapping for game settings and controls, as well as store and load control mapping data to and from the json file. Other UI within the game prototypes is texts such as score counts, level number and a reload complete sign. Finally, the main menus of each prototype include buttons that have 3 purposes, Play Game, Game Settings, and Exit Game.

A picture containing diagram

Description automatically generatedGraphical user interface, website

Description automatically generated  
*Figure 11: The main menu screens of each prototype*

### Implementation

The Game Settings menu for the first prototype, only has a few changeable key options and game options due to the very simple game mechanics. As can be seen in Figure 9, there is only two options for moving the character which is moving it left and right. The option for choosing ball drag/throw with a key press and the player can set which key that is, or using the mouse to drag the ball, and finally the option of having the y position of the ball drag inverted or not.

Graphical user interface

Description automatically generated  
*Figure 12: Game Settings scene in Prototype 1*

For the second prototypes game settings scene, there are a lot more buttons and keys to change. As can be seen in Figure 6 above, a screenshot from the second prototype, there are four options for movement, a button for Jump, Shoot and Walk/Run. These can all be saved to the json files, and whatever the left and right buttons are set to, these become the same in the first prototype, this is mapping the key-bind controls from one prototype to another. The same by setting the left and right keys in the first one, the left and right keys in the second one will also change, Figure 12 shows this.

Graphical user interface

Description automatically generated Graphical user interface, application

Description automatically generated  
*Figure 13: Showing the mapping of the key-bind controls between the two prototypes*

## Issues Encountered

It was found out that the base Unity Input Manager does not allow for re-bindings of any kind, controller, or keyboard, whilst it is on runtime, this messed up the way that things were originally planned such Inputs and mapping the controls, so there had to rethink and a change of plans.   
Key bindings for the first prototype, Wheelchair Basketball Shoot, was a quick fix and relatively easy because when first researching ways of changing controls, a custom input manager was set up with changeable key-bindings only, but at the time I could only change these key-bindings in the inspector and not in the game itself during runtime. But despite this, it gave a foundation to build on after having to change the original plans.   
The Unity Package, Input System, could have been used but it was too new and there was not enough knowledge about it for development and it would mean having to re-build a lot of things from the start which was not viable for the point in time in the development process

The Second Prototype, Bye-Bye Thoughts, included standard assets from the Unity Asset store which had its own First-Person Controller which was included into the game. This was chosen for the prototype so it could be built more easily, and more time could be spent on the mapping program itself.   
This backfired as the scripts in the folders for the First-Person Controller used the Input Manager that is built into Unity which was decided was going to be used and re-bind controls using that, but it was found out too late in the development process that it was not possible to rebind Input Manager controls during runtime.   
Therefore, the controls had to be changed in the assets scripts to do key-bindings and keyboard controls, but the use of the Input Manager was included still, to allow for controller use which the player can switch to in settings but cannot change the control button bindings.

Another issue from using the First-Person Controller was that because it was in the Standard Assets, there was only the option of downloading all the files and could not delete some of the files that weren’t needed because some were connected to the scripts that were being used for the controller, which adds a lot of unnecessary data, files, and folders to the Unity Project. Also, because those scripts were in their own folder, for some of the created scripts, work-around had to be found to get the scripts to work with each other. A way this was done was just by creating some of the new scripts in that folder as it was able to be accessed in the game either way, and there wasn’t a need to connect to the controller kept in the Assets Scripts folder. This was not ideal and definitely not robust, but it was the only option at the time.

### Bugs Remaining

If the player starts the game in the second prototype and then goes back and tries to change the input method, switching from keyboard to controller, the controller input may switch on, but the controller drops down menu does not show. But if the player loads to the game settings first from the main menu when the game is first loaded, the dropdown menus work as intended.

# Testing

Testing is a crucial aspect of the development and advancement of a project because it ensures that a program/application is functioning and free of mistakes. Because a project has functional requirements, tests can be used to ensure that they are met. This project has a number of test cases that are connected to each functionality for sections of the project. The full test table is found the appendix.

## Test Case 1

This tests the functionality of the main menu and its interactions such as the buttons and loading into different scenes, as well as the game application loading and starting with the main menu scene. Figure 14 and 15 shows the buttons are interactable, when the mouse hovers over the button it is highlighted. Each button when pressed, loads their respective scenes, or executes functions. Play Game loads the game scene, Game Settings loads the settings scene and Exit Game exits the game from the build. Figure 16 and 17 shows the order of the build which proves that the main menu is the first scene to appear when the game is built. This test case applies to both of the game prototypes.

A screenshot of a computer

Description automatically generated with low confidence   
*Figure 14*

*Diagram

Description automatically generated*  
*Figure 15*

A picture containing graphical user interface

Description automatically generated  
*Figure 16*

*Graphical user interface

Description automatically generated with medium confidence  
Figure 17*

## Test Case 2

This tests the functionality of the Game Settings and its interactions such as the buttons, loading the scene and back into a different scene, saving and loading from a json file. Test 2.1 is proven with the game settings loaded in the build, the buttons are also shown to be interactable with the mouse hovering a button highlighting it, shown in Figure 18 and 19.

The key-bindings can be changed when the button is pressed on and a key is pressed, the button text changes to the pressed key, proving Test 2.2. And when save is pressed that key is saved to the json file, proving Test 2.4, Figure 20 shows a json file before a key change, and Figure 21 shows it afterwards.

The load button works as Figure 22 shows newly loaded keys from the json file, proving Test 2.5.

The option buttons within both of the prototypes, the first having the option between mouse and keyboard for the ball drag/throw and the second having keyboard or mouse for shooting, these were proven to work with the key binding working on the keyboard side, and the debug log in the console showing that the button had been selected. Figure 23 and 24 shows this working in the unity editor, the console visible on the bottom half of the image. This proves Test 2.3

Finally, the back button when pressed redirects the player back to the main menu scene, with all the data saved in the json file and the game can now be played with the new key bindings.

Graphical user interface, diagram

Description automatically generated  
Figure 18: Game settings for prototype 1

Graphical user interface

Description automatically generated with low confidence  
Figure 19: Game settings for prototype 2

A screenshot of a computer

Description automatically generated  
Figure 20: Original set key code bindings

Text

Description automatically generatedFigure 21: Newly set key code bindings

A picture containing graphical user interface

Description automatically generated  
Figure 22: Loaded key codes in prototype 2 game settings

Graphical user interface, text, application

Description automatically generated Figure 23

Graphical user interface, application

Description automatically generated  
Figure 24

## Test Case 3

# Critical Evaluation

## Project Evaluation

The Project’s aims have been met, it does what it was set out to do, maps controls from one game to another. Including fully functional prototype games that can be played to test the controls and mapping program as well as show disability representation in games. I believe it is fairly good in saving the players preferred key bindings and loading them to another game, however it does not include controller re-bindings and saving and loading those. Although the key bindings could be considered enough to showcase the ability of the mapping program, one of the aims were for control mapping as well as keyboard.

Despite this, I am proud of the project I have been able to create, including being able to make a functional mapping program. Parts I have been particularly proud of is using json to read a file that holds the keycodes for the games and write the saved controls onto a different prototype, this is because it allows the games to be more accessible for people. I am also particularly proud of the design and implementation of the prototypes themselves; it was very important to me to include representation of disabilities in these prototypes, physical disability shown in the first and mental health incorporated in the second, to showcase that there can be more done to help disabled people feel more included in the world, this includes in video games, it helps them feel seen and understood. Personally, the second prototype’s themes resonate more with me, as a person struggling with mental health, I liked the idea of expelling a bad object and watching the scene get brighter.

Things that I was unable to implement, were fully functional controls for the first prototype, and not being able to rebind controller buttons. This is something I would prioritise in implementing had I been given more time; this mainly was not implemented due to poor planning on having to change my plans for how Inputs would work in the prototype games. This could be fixed by adding the Unity Package, Input System, a relatively new package that was released in 2020. This allows for inputs placed in the system to be rebound whilst the game is on runtime which the Input Manager is unable to do.

Another thing I would implement given more time, is creating a Unity project just for the mapping program. Having a single scene filled with buttons that can be set, having a long list of buttons and actions to cover a wide range of games and their possible settings, and using the json files and game settings and game manager files that I used as the ‘mapping program’ for the prototypes and place them in this unity project and that itself could be a more functional and robust Control Mapping Program and an actual program instead of just working only through game settings scenes between games, saving bindings on there and loading them up onto any game (in this case the two prototypes). I believe this would be relatively easy to implement, and I had planned this at one point so there is already a Unity Project set up named Mapping Program, with a few text and button UIs but nothing functional at this time.

## Evaluation of your approach

-In terms of project management approach and in terms of development methods or research methods used

I believe the project management approach I took, by using Agile Methodology to break down each component and task into stages, acting as a sprint, was the correct way to manage and develop my project. As it was relatively clear what each main component would be, this being the two prototypes and the mapping program, it was simple to divide the tasks in each one, plan, design and implement before moving to the next task or stage.

I think the way I went about my research whilst developing the project, could have been revised. My method was to research code and algorithms as I went along in my development, following the sprint head and sticking to finishing that sprint before researching anything else ahead of it, therefore anything else in a stage above the sprint I was currently on at the time. This approach worked mostly for the beginning stages of the project, but this later caused me to develop the prototype games using the Input Manager, because I assumed there would be a way of being able to rebind the keyboard and controller controls whilst the prototypes were in runtime. Because I did not research this part ahead of creating the prototype games, I ended up having to rethink and change my original plans of how to create the control mapping. This cost me, as in the end I was only able to implement mapping of the keyboard controls and change its key binds and not the controller and re-binding the controllers’ buttons.

-How successful was the approach you adopted?

I believe that the approach I adopted for project management was fairly successful, I completed the tasks in the order in which I had planned to do them, and for a majority of the tasks it took me as long as I expected it would take, but there were a few discrepancies as can be seen in the Gantt chart in the appendix, for example development of the control mapping program took a lot longer than anticipated due to changing approaches in how it would be implemented. For the research methods, as mentioned above I believe that they weren’t quite successful as it was down to poor planning ahead that I had to rethink the way I was to create the mapping program.

-What techniques did you use that worked well? Why?

-What did you learn by doing the project? – did you meet your academic objectives

-How would you do it better if you did the project again?

## Evaluation of tools used:

-What languages, libraries, environments etc did you use for the development? Or, what libraries, methods, techniques did you use for the research?

-Were they suitable? What were their main plus and negative points?

## Final paragraph

-end on a positive note

# References

Brody, C. (2020). *The Need For More Disabilities In The Games We Play*. [online] ablegamers.org. Available at: <<https://ablegamers.org/the-need-for-more-disabilities-in-the-games-we-play/>> [Accessed 6 Jan. 2022].

Carrera, S. (2016). *Accessibility in Games: Including People with Disabilities*. Germany: LAP LAMBERT Academic Publishing.

Piskelapp.com. n.d. *Piskel - Free online sprite editor*. [online] Available at: <https://www.piskelapp.com/> [Accessed 27 December 2021].

# Appendix

## Project Availability

This project can be accessed through a public GitHub repository under the username RuyaKH, the link for this repository is; <https://github.com/RuyaKH/Final-Year-Project>

Test Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Case ID | Test Case Objective | Test Case Description | Expected Result | Pass or Fail |
| 1.0 |  | **Main Menu** |  |  |
| 1.1 | Test images and buttons are loaded to the main menu scene | Open Game application/Main menu | Asset images and buttons load to scene | **Pass** |
| 1.2 | Test that the play game button loads player into game scene | Click on Play Game button | Redirected to Game scene | **Pass** |
| 1.3 | Test that the game settings button loads player into game settings scene | Click on Game Settings button | Redirected to game settings scene | **Pass** |
| 1.4 | Test that the exit game button exits the game application | Click on Exit Game button | Game application is exited | **Pass** |
| 2.0 |  | **Game Settings** |  |  |
| 2.1 | Test that game settings scene is loaded with assets, buttons, and text | Load into game settings scene | Text, buttons, and assets loaded | **Pass** |
| 2.2 | Test that key-bindings can be changed when press key on button and text changes to new key | Click button on key-binding to change and press new key | Button text shows new key set and that is new binding | **Pass** |
| 2.3 | Test that the selected button for between two options is correctly set. | Click on option button | The debug line for the selected option shows up in the console | **Pass** |
| 2.4 | Test that set controls and key-bindings changed can be saved to database | Click on save button | New controls and key-bindings can be saved | **Pass** |
| 2.5 | Test those controls mapped that have been saved in database can be loaded to game | Click on load control maps button | Saved controller/key-binding settings in database loaded to game settings and buttons changed | **Pass** |
| 2.6 | Test back button leads player back to the Main Menu scene | Click on back button | Redirected to Main Menu scene | **Pass** |
| 3.0 |  | **Game Prototype 1** |  |  |
| 3.1 | Test game runs and plays on load, with all assets and UI loaded and scripts running with no error | Load into game scene | Images and assets loaded, player loaded, text and button UI loaded, enemy loaded | **Pass** |
| 3.2 | Test 2D colliders loaded and collide with relevant objects | Move player or ball towards a 2D collider | Player and ball collide, stop moving on collision | **Pass** |
| 3.3 | Test player and ball objects are moveable with set or default controls | Use controls set or default A and D keys to move player and ball | Player and ball objects move | **Pass** |
| 3.4 | Test that player animation occurs when moving or shooting | Move player or shoot ball | Player animation occurs | **Fail** |
| 3.5 | Test that physics in game scene work correctly, gravity working | Load into game scene | Ball and Player use gravity when moving and shooting | **Pass** |
| 3.6 | Test ball spins and bounces when collided with | Ball falls to ground once shot | Ball spins and bounces | **Pass** |
| 3.7 | Test ball is kinematic when not shot | Move player and ball | Ball does not move from player hand when not shooting | **Pass** |
| 3.8 | Test ball is dynamic when shot | Shoot ball | Ball moves freely in the game scene, with physics reacting | **Pass** |
| 3.9 | Test when ball and basketball net collide, ball and player reset | Aim ball and shoot at basketball net | Player and ball reset to original start points (checkpoints) when ball has collided, and ball is reset to kinematic state | **Pass** |
| 3.10 | Test when ball and basketball net collide, score count increases by 3 | Aim ball and shoot at basketball net | Score count UI on game scene is increased by 3 points | **Pass** |
| 3.11 | Test player and ball do not move using move left and right controls whilst ball is dynamic | Move left and right when ball is dynamic | Player and ball should not move using the move left and right controls | **Pass** |
| 3.12 | Test that ball only allows one mouse drag once shot | Drag on screen as shooting | Ball should not move | **Fail** |
| 3.13 | Test there is a line renderer and is correctly drawn to screen when player aiming to shoot | Drag on screen as shooting | Line renderer appears on screen that aims the ball towards the basketball net | **Fail** |
| 3.14 | Test there are multiple levels, each one loaded after the next | Complete a level | Game reset and new level loaded with new scripts/game mechanics | **Fail** |
| 3.15 | Test the key bindings can be changed using the key binding script in Unity Inspector | Change key-binding script using inspector | Controls changed to different/new keys and can be used | **Pass** |
| 3.16 | Test the on screen drag shoots the ball with increased power the further the distance dragged | Drag on screen as shooting, try different distances of start and end points of drag | The longer the drag, the more power the ball has when shooting and vice versa | **Pass** |
| 3.17 | Test that game mechanic scripts are working as intended and no errors appear on console | Run and play game and look at console | No errors appear on console and game runs smoothly | **Pass** |
| 4.0 |  | **Game Prototype 2** |  |  |
| 4.1 | Test game runs and plays on load, with all assets and UI loaded and scripts running with no error | Load into game scene | Images and assets loaded, player loaded, text and button UI loaded, enemy loaded | **Pass** |
| 4.2 | Test that the camera follows the player as they move around the game scene | Run game and move player | Player moves around game scene using controls set or default controls | **Pass** |
| 4.3 | Test that pistol is positioned on the camera correctly and does not move or wander | Run game and move player and look around with camera | Pistol positioned in place and does not move out of place when moving/looking around | **Pass** |
| 4.4 | Test the player movement of moving left, right, forward, back, rotate left and right and jumping | Use set keys/controls to move player | Player moves around game scene smoothly | **Pass** |
| 4.5 | Test Pistol animation when shooting | Press key/control button set to shoot | Pistol moves up and down in an animation motion to reflect shooting | **Pass** |
| 4.6 | Test when shoot button pressed, bullet projectile fired, and bullet instantiated | Press key/control button set to shoot | Bullet fired from pistol in a straight-line using force | **Fail** |
| 4.7 | Test terrain and world objects colliders are set and collide with player | Move player towards terrain/world objects to collide | Player stops moving when colliding with objects | **Pass** |
| 4.8 | Test when bullet collides with enemy object, bullet and enemy disappears | Shoot bullet at enemy and colliding with it | Enemy and bullet objects destroyed | **Fail** |
| 4.9 | Test when bullet collides with enemy object, score increases by 1 | Shoot bullet at enemy and colliding with it | Score count and UI increased by 1 | **Fail** |
| 4.10 | Test when bullet collides with terrain or other world objects, physics occurs | Shoot bullet at objects or terrain and colliding with it | Bullets ricochet off objects and terrain/ground, bouncing off or lay on ground | **Fail** |
| 4.11 | Test there are multiple levels, each one loaded after the next | Complete a level | Game reset and new level loaded with new scripts/game mechanics | **Fail** |
| 4.12 | Test game lighting works correctly | Run game scene | Lighting is on and shows as expected, not different in anyway | **Pass** |
| 4.13 | Test enemy defeated animation occurs | Defeat an enemy | Explosion or fading animation | **Fail** |
| 4.14 | Test the game audio works, walking steps with player moving and gunshot sounds | Run game scene | Hear game audio as player moves around hear footsteps, and gunshots when shooting occurs | **Pass** |

## Figure Table

|  |  |  |
| --- | --- | --- |
| Figure No. | Image | Description |
| 1 |  | The basic functional process of the first prototype game, starting from left to right |
| 2 |  | The basic functional process of the second prototype game, starting from top left to the bottom right. |
| 3 |  | The projects Gantt Chart |
| 4 | Diagram, schematic  Description automatically generated | Use Case Diagram for the Main Menu’s for each game prototype |
| 5 | Diagram  Description automatically generated | Use Case Diagram for the game prototypes |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |
| 11 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |