Maces and Talons

Final Report for CS39440 Major Project

Author: Callum Gwynedd Hay Hutchinson (cgh2@aber.ac.uk)
Supervisor: Dr Bernie Tiddeman (bpt@aber.ac.uk)

16th April 2018

Version 2.0 (Release)

This report is submitted as partial fulfilment of a BSc degree in
Computer Science including Integrated Industrial and Professional Training (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name: Callum Gwynedd Hay Hutchinson

Date: 04/05/2018

Consent to share this work

By including my name below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name: Callum Gwynedd Hay Hutchinson

Date: 04/05/2018

Acknowledgements

I am grateful to my family and friends for supporting me and allowing me the time to complete this project.

I am grateful to the anonymous users who helped test the project.

I'd like to thank Dr Bernie Tiddeman for providing support and inspiration to the project during meetings.

Abstract

This aim of this Project is to transform a medieval Scandinavian board game known as Hnefatafl, into a Windows and Android video game. The game will feature an offline mode and an online mode, the offline mode will provide two player game functionalities from a single device. The online mode will allow two different games to connect to each other through a matchmaking system, which will be provided by a server. The server application will be bespoke for the project, will use asynchronous TCP to allow large scalability and reliability.

The game client and server were both created using C#, the game was made in the Unity engine and the server an application in Visual Studio. This decision was good and bad, there were changes that had to be made during development to get the server to work on the hosting application. The hosting application used was Amazon AWS EC2 and provided a flexible free tier with just enough performance to run the server.

There are several different techniques used during the project, Feature Driven Development was used for the structuring how the tasks were under taken, Black-box testing was one of the main forms of testing the different aspects of the Project and User acceptance testing was used to discover how anonymous users would react to the game client.

The Project was successful and provided the features it set out to produce, but there were elements that could have been improved. These features relate to the games user experience, due to lack of testing during the development of the project certain aspects were not as user friendly as intended, such as teaching the rules and readability of text.

Contents

1. Background, Analysis & Process

1.1. Background

The Project aims to create a video game based on the Scandinavian medieval board game 'Hnefatafl' [7], this game will feature a single device two player hot seat mode (offline multiplayer) and an online multiplayer mode using a custom server.

Having previously developed two Android applications and several years of experience developing in Unity 3D, this Project idea was interesting to me because I could use skills that I had learnt at University and skills developed in my spare time to produce the most technical game that I have created.

Hnefatafl is a variant of the larger game Tafl [6], which is a board game in where two forces much like chess compete against each other to win, unlike chess the two sides are uneven and have different win conditions. As the board game moved around northern Europe different rulesets were developed so were layouts to the board and pieces [5], Hnefatafl which is the Scandinavian variety uses a board that is 11 squares by 11 squares (11x11) or less commonly 13x13, there are other known types such as Ard-Ri and Tawlbyund which use 7x7 and 11x11 respectively.

The two different board sizes for Hnefatafl also presents two different configurations for the pieces as shown in Figure 1, while the outside 'Barbarian' pieces remain the same the Vikings in the centre have a slightly different formation between the two board sizes.

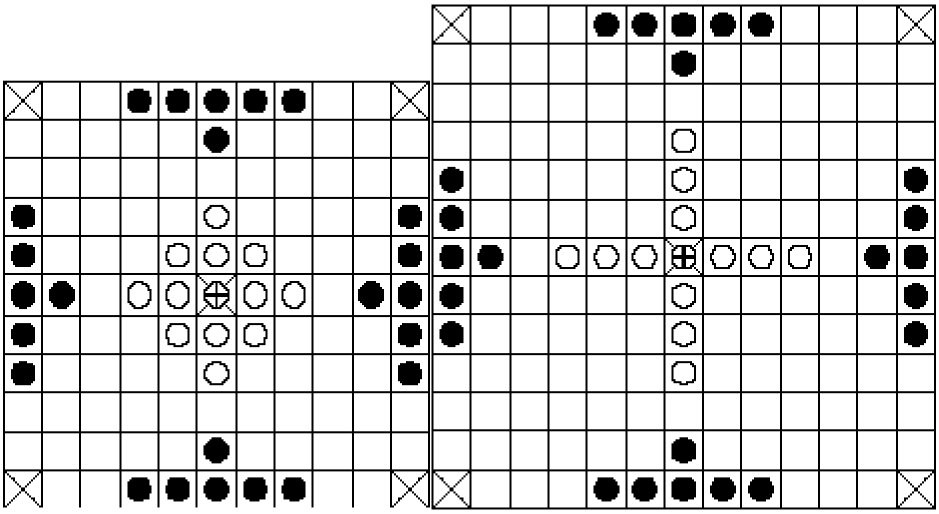


Figure 1 [5]

The Project will attempt to recreate both board size within the game, but if time does not permit, this will be reduced to just using the more commonly used board size 11x11 and the corresponding piece layout.

The rules of Hnefatafl are a lot simpler than those of chess, all pieces move in the same manner, in straight lines horizontally and vertically as far as they want until hitting an obstruction. This is much the same as a rook in chess except that you do not take a piece by going on to its space, pieces in Hnefatafl cannot pass through or go on to space occupied by another piece, this allows for strategies that rely on blocking the opposing team off and trapping them. To take a piece it must have two opposing pieces on two opposite sides, therefore creating a line of three, the taken piece is removed from the game. The one unique piece is the King this is placed in the centre of the board and is surrounded by the defenders or in the case of Hnefatafl the Vikings, to take/capture the King it must be surrounded on all four sides by opposing pieces or edges of the game board. Figure 2 shows the different cases where the King could be taken, the difficulty of taking the King is made easier by the uneven numbers, the Barbarians have almost double the number of pieces at 24 versus the Viking's 13.

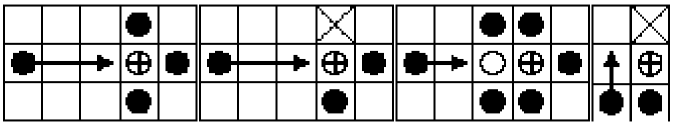


Figure 2, [5]

The objectives of both teams are different, the 'Barbarian's' sole goal is to capture the King, theoretically, this also could lead to them capturing all the other Viking pieces but this is not necessary for them to win. The Vikings must get their King to one of the four corners of the board to win, or capture all the opposing Barbarians but this is highly unlikely due to the imbalance in the two sides, this outcome would only happen if the person controlling the Barbarians is very inexperienced.

1.1.1. Similar Systems

There are several Android applications that are published on the ‘Google Play Store’ and emulate Hnefatafl, they provide features that the Project aims to produce so they can be analysed to see what works well and if anything can be improved. Due to the nature of the game, many of the features are similar in each version and most of these will be included but some will have to be left out due to development time constraints.

Hnefatafl [2]

The first application that was looked at contains a mixture of features some that will not be included in the Project, this version only contains offline modes, a hot-seat mode as the Project will contain and an offline AI (Artificial Intelligence) mode against a varying difficulty of opponents.

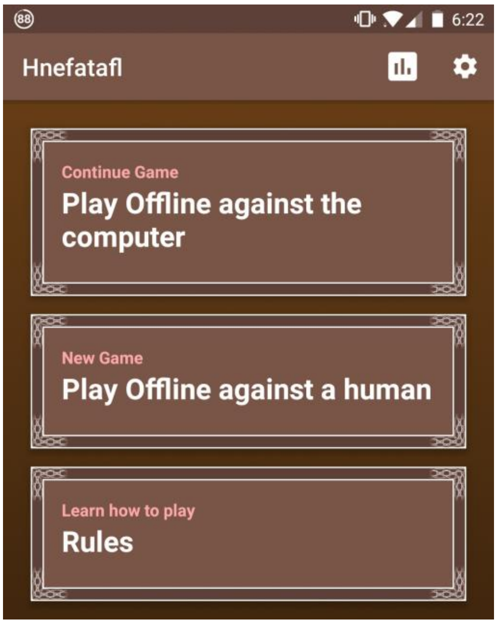
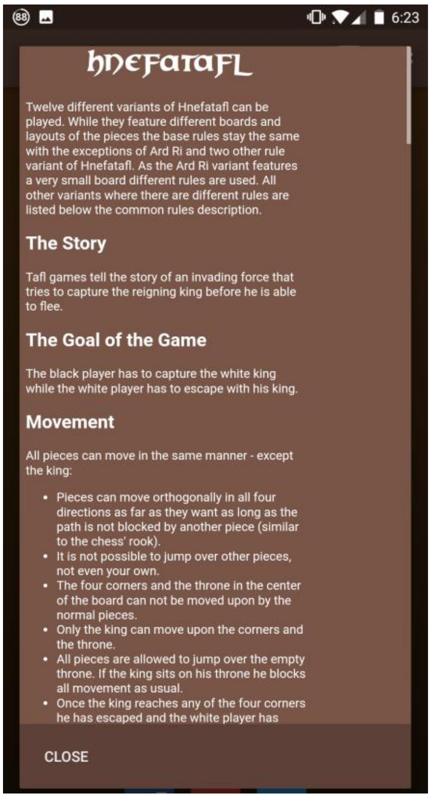

6:22
Hnefatafl
Continue Game
Play Offline against the computer
New Game
Play Offline against a human
Learn how to play
Rules

Figure 3

Figure 3 shows the user interface is minimal and while not overly attractive it achieves its purpose providing clear options to the user, this will be something that the Project will implement due limited art resources.

Figure 4 is a useful feature that shows the rules of the game to the player, since Hnefatafl and other Tafl are quite different to other games it is hard to show this visually, this may be something that can be improved on within the Project.



Hnefatafl

Twelve different variants of Hnefatafl can be played. While they feature different boards and layouts of the pieces the base rules stay the same with the exceptions of Ard Ri and two other rule variant of Hnefatafl. As the Ard Ri variant features a very small board different rules are used. All other variants where there are different rules are listed below the common rules description.

The Story

Tafl games tell the story of an invading force that tries to capture the reigning king before he is able to flee.

The Goal of the Game

The black player has to capture the white king while the white player has to escape with his king.

Movement

All pieces can move in the same manner - except the king:

- Pieces can move orthogonally in all four directions as far as they want as long as the path is not blocked by another piece (similar to the chess' rook).
- It is not possible to jump over other pieces, not even your own.
- The four corners and the throne in the center of the board can not be moved upon by the normal pieces.
- Only the king can move upon the corners and the throne.
- All pieces are allowed to jump over the empty throne. If the king sits on his throne he blocks all movement as usual.
- Once the king reaches any of the four corners he has escaped and the white player has

CLOSE

Figure 4

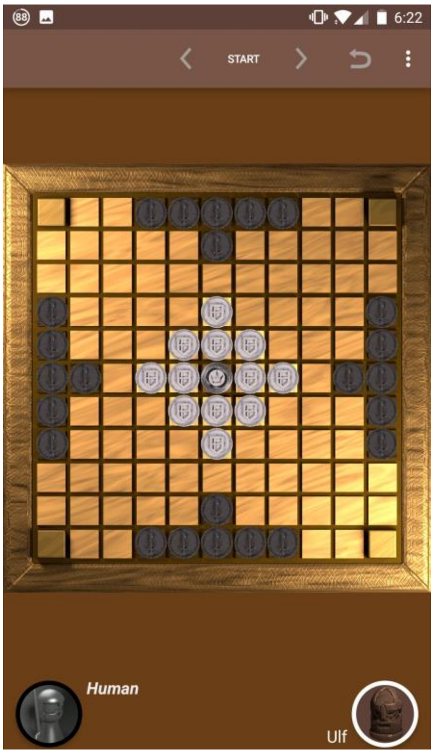
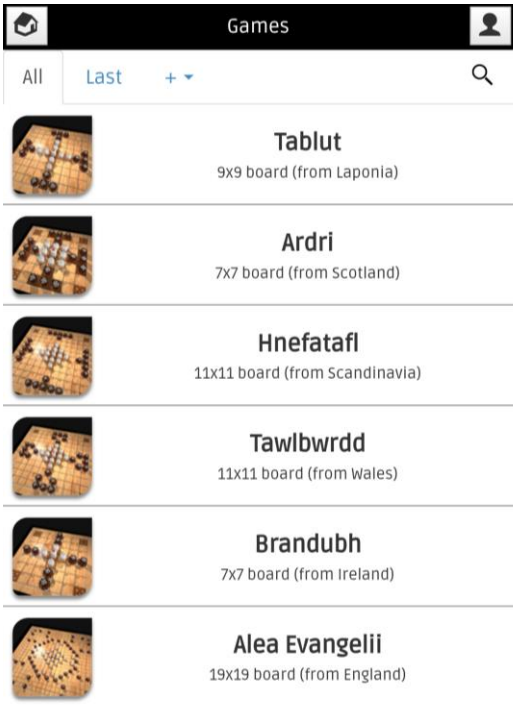


Figure 5

Much like the user interface this game features a simplistic board design, this can be seen in Figure 5. It does allow the user to select assorted styles of playing pieces, and this game also allows the selection of different board sizes and layouts of pieces

While the Project will focus solely on the Hnefatafl version of the game, the simplistic user interface and game view could be of inspiration.

Tafl [3]


Games

All Last +

Tablut
9x9 board (from Laponia)

Ardri
7x7 board (from Scotland)

Hnefatafl
11x11 board (from Scandinavia)

Tawlbwrdd
11x11 board (from Wales)

Brandubh
7x7 board (from Ireland)

Alea Evangelii
19x19 board (from England)

Figure 6

This game offers offline play against the computer and online multiplayer, it is unusual that is doesn’t offer offline multiplayer as that is the easiest method to implement. Much like the previously analysed game [2], this offers different layouts for the boards as seen in Figure 6. With these different board layouts there should be accompanying rules changes, but after testing out the board layout for Hnefatafl there are slight inconstancies with the other two games that were analysed. These changes in the rules could be attributed to personal interpretation by the developer as the game itself has many different variations and origins.

The Project will stick to the most commonly established rules as this will be easier for inexperienced users to pick up and returning users of the system to remember.

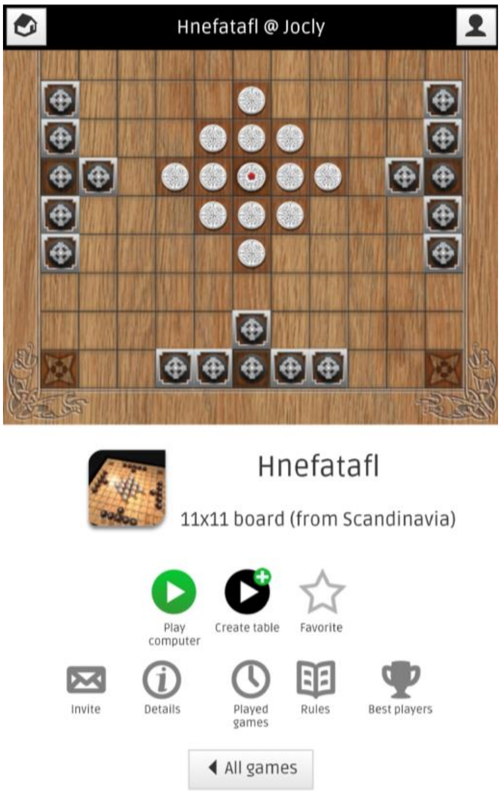


Figure 7

The user interface when starting a game (Figure 7) is confusing and does not easily show how to setup multiplayer games. The buttons for game setup should be obviously distinguishable from the other information buttons, in this case, they are different colours to the grey icons for other information, but their size could have been increased to improve their prominence on the screen. The image showing the game board is slightly unnecessary as we have previously chosen which board we want to use and could free up space for clearer buttons.

The game view (Figure 8) has the most simplistic interface compared to all the comparisons and represents the game in a classic top-down 2D view. It doesn't show any additional information including which team is starting, this can make it confusing if you are playing the computer and you have been selected to go first (which during my testing happened every time).

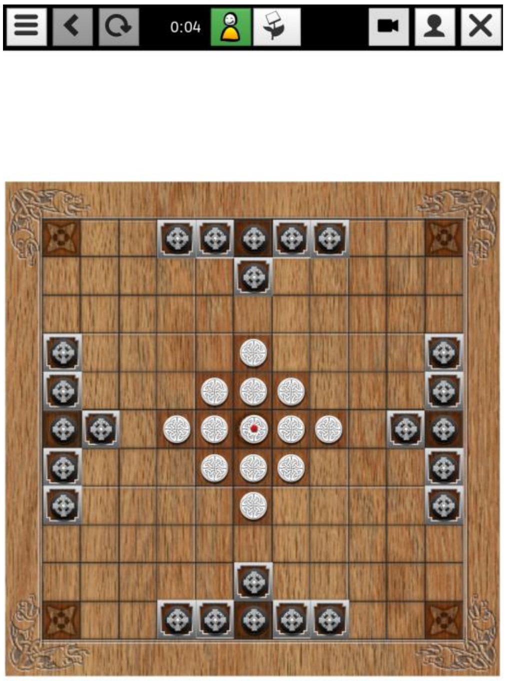


Figure 8

Hnefatafl [4]

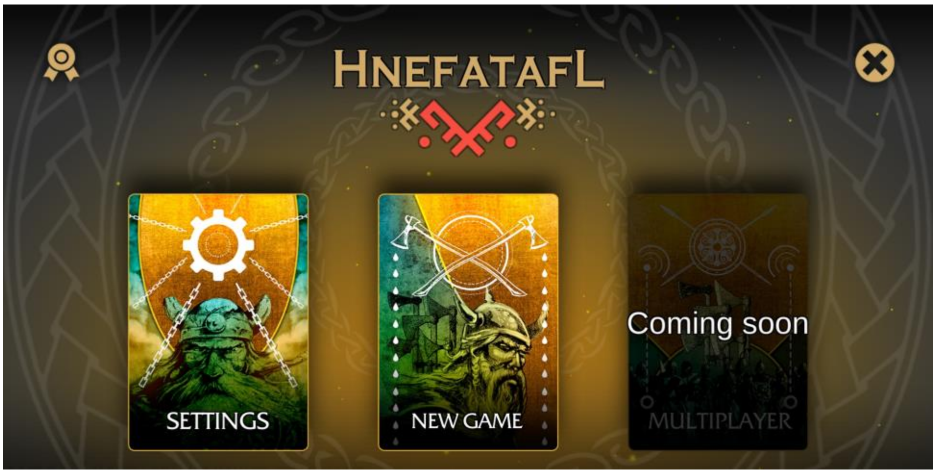


Figure 9

This game was by far the most graphically appealing game and a lot of effort had been put into art assets, the main menu (Figure 9) is very clean and directs the user to the main aspects of the game. Interestingly the game was made with the same game engine which the Project is going to be made in which shows the graphics and polish that it is capable of.

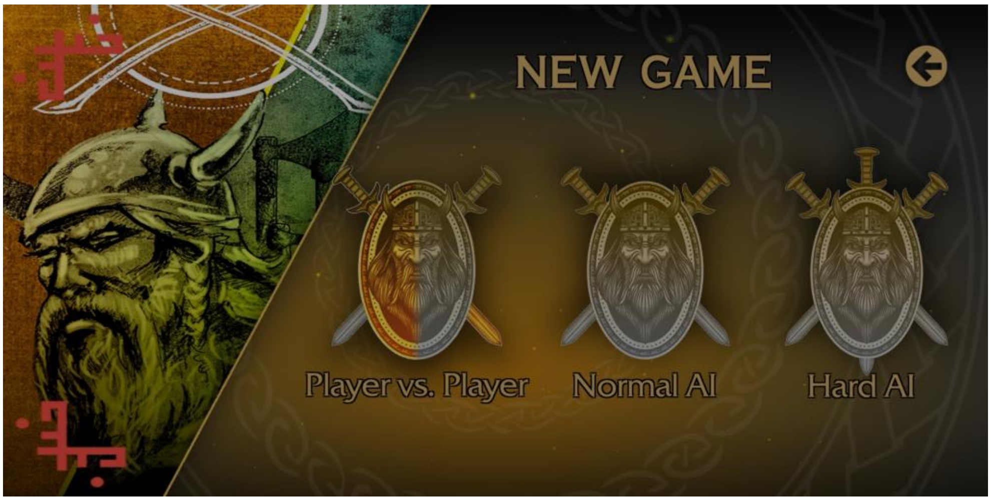


Figure 10

Several game modes exist in this game (Figure 10), all of them are offline currently with a greyed-out box for multiplayer shown in Figure 9. There exist two different difficulty levels of AI opponent which provides a small choice in variety compared to [2] which had 10+ difficulty levels. The Project will not focus on an AI opponent and focus on multiplayer so there is not much to learn here.

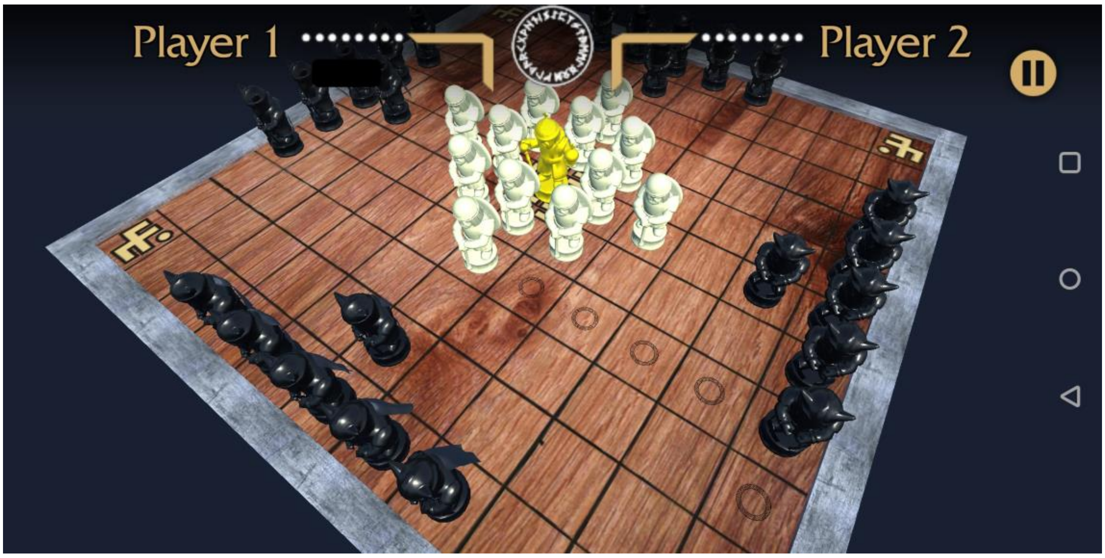


Figure 11

The game view is the most unique of the three games, changing styles from a 2D top-down view to a 3D view (Figure 11) with a modelled game board and pieces while also implementing lighting and shaders. During gameplay on this version, there were several gameplay choices which may have been intentional but were possibly bugs, such as when a piece was selected at the beginning of your turn you could not then select another piece and must move the piece selected. This shows as a first-time player to the game that even small bugs can be deterring to a player and good first impressions are paramount.

Conclusion of similar systems

None of the existing interpretations that were analysed, contain the exact mix of features which the Project aims to bring, but it has given the chance to study how each one implemented their version of the features that will be included. Online and offline multiplayer will play a big part, unfortunately, there is only one game with functioning multiplayer but all of them feature offline multiplayer to look at. Graphically the style that stood out the most was the one used the least and makes a static board game feel the most interactive, while a 2D view is simplistic a 3D view if done right draws the user in.

1.1.2. Development Research

For the Project, there are several different technology choices that must be made as they will be the key tools used during development, changing any of these during the Project could possibly require starting the entire Project again or at least large parts.

Unity

Having personally developed basic Android applications using the Unity game engine this was the immediate choice when deciding how to implement the Project. Having previous developed in Java this was a possibility but even when using a framework such as the one provided by Android Studio [8] the amount of work to develop certain features for example, a 3D environment as well as a built in shader and lighting system, this made Unity a better option.

The choice to use Unity, allows easy exporting of the Project to a wide variety of platforms. Java would also allow the ability to run on many different systems, but they require the JRE (Java Runtime Environment) to be installed, Unity provides native builds for each platform that it is exported to, this means machines can run it without the need to install extra software.

C# or Java

Developing the Multiplayer server will be one of the most technically challenging parts of the Project, the language to develop it in then is very important and must be chosen carefully. The two languages that are in consideration are C# and Java, syntactically these languages are practically identical and some code can literally be copied from one to another, their main difference is in how they are run.

Java being an interpreted language requires the JRE (Java Runtime Environment) to be installed on target machines, there is some associated performance overhead because the code must be executed and converted to machine code at runtime. C# on the other hand is built by a compiler into an executable binary, this typically can only be run on a Windows but due to the language being developed using the Common Language Interface (CLI) specification [9] it can be run on several other systems.

The two languages are very similar and besides the target platforms there are no other major differences between the two, for this Project it will be easier to develop in C# as the scripts used in the Unity Engine are in C# and code can be shared across applications to speed up development.

TCP vs UDP

When implementing an online multiplayer system there are different networking protocols to consider, TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) [10]. They are both different methods of transferring data packets across a network, each with contrasting features to the other.

TCP is the slower but more reliable protocol [11], it achieves this by sending each the packets with reference to the order they were sent in, e.g. packet 1 has a reference of one, packet 2 has a reference of two, etc. This allows the receiving machine to know how to order the packets and if any have been lost, if 2 arrives before 1, it will wait until it has 1 and then process them in order. This is where the speed can decrease, if many packets are arriving out of order then it will take time to wait for the correct ones to arrive. Even worse can be if a packet does not make it at all, then the receiving machine must send a request for the missing packet to be resent which creates more time delay.

Conversely UDP does not have any built-in ordering and unlike TCP all the packets do not follow the same route to the target machine. If a packet finds a faster route than the packets that left before it, then it could arrive before they do. This is where errors can occur with out of order packets, unless a method of ordering the packets has been implemented then they will just be processed as they arrive, even if they are early.

UDP is more suited to a real-time application where you need the data continuously sending across the network and can afford some data loss with the benefit of speed. TCP even with its drawback of increased overhead when needing to request packets to be resent, it can guarantee that the packets will arrive and that none of the data is lost [12]. The Project will be implementing a turn based system for the game, therefore a real-time solution is not needed. The user will never noticeably experience the difference between TCP or UDP so the most reliable protocol will have to be used.

1.2. Analysis

Using the background research, the Project can take shape and provide some criteria to follow and will would be produced at the end of the Project.

To allow more time to develop the key defining features of the game client, the Unity engine was decided to be a better choice over making the game from scratch in Java or even using the Android Studio to provide some support for Android. Using Unity will allow for exporting the game to Android and Windows from within the engine and not require any laborious process to get the game built for both, thus allowing for quick turnaround when testing builds and fixing bugs.

Having studied the art styles of the other applications available, simplistic menus allow for a more understandable user interface that doesn't overwhelm users with lots of options immediately. The main menu should be just a couple buttons leading to the different game modes and a button to display the help screen. When in the game view, the board should not be obstructed and useful information such as, who's turn it is and how many pieces from each side have been taken should be displayed in an unobtrusive manner.

In the similar applications the majority opted for a simple top down 2D view, this was easy to use but didn't provide any immersion. Seeing tiles on a board is not as aesthetically appealing as physical models that move along the board using animation. The game client the Project will develop will use the Unity engine to full advantage and leverage its 3D capabilities including lighting and shading.

Through the research the games that were analysed implemented a combination of online multiplayer or an artificial intelligence opponent (some with varying difficulty). Both features are a large undertaking and only one would be achievable in the time frame, while also including offline multiplayer and a respectable level of polish. Due to personal experience previously developing an online multiplayer server using Unity, the Project will include an online multiplayer mode once a custom server application is developed.

The online multiplayer server would have to be designed as an application separate to the main game, to allow it to be run on a remote machine. Through the research of the best programming language to use, out of the two chosen there was no clear winner, this meant C# was selected due to it also being used as the scripting language in the Unity game, allowing the sharing of network code and several classes. The server will also be using a TCP connection to transfer the data between clients, this is slower (relative to UDP) but more reliable connection and will allow for packets to arrive in order so that turns do not go faulty because of out of order packets.

1.2.1. Requirements

With the research and analysis done the items that will be produced for this Project will include:

R1 Windows and Android Game Client

It will be the same game but released and built for two different platforms, they will remain playable and relevant to each platform through different techniques.

User Interface

The Unity engine provides several user interface components these can be scaled in a variety of manners, using a scaling method that keeps the UI anchored at set points on the screen allows for lots of different resolution devices and even older aspect ratios. There will need to be a main menu for selecting game modes, as well as a help button which will bring up a screen showing visually how to play the game. When playing the game, it is good to know how many pieces have been taken so the game will keep track of this and display it for both users on the user interface.

Sounds

To make the game more engaging sounds will need to be used to provides the users actions with more weight, buttons on the user interfaces should have a satisfying click sound so that it is apparent when they have been clicked. During the game different sounds should be played to indicate different actions such as, Vikings winning, Barbarians winning, Vikings taking a piece, Barbarians taking a piece and whenever a piece moves but nothing is taken. This will all provide the user with extra information in case they miss what happened visually.

Help Menu

The help menu will have to explain all the rules of the game so that inexperienced players will understand how to play, because the game is not common there is a large likelihood that most users downloading the game will not have heard of the original. To explain the rules, they should be presented next to screenshots of the game re-enacting the scenarios being described.

3D View and Scrolling

In the game view the game board and pieces will be represented in a 3D view, with the board being able to turn and zoom in and out. The pieces will be represented by models which shows the difference between pieces, Vikings, Barbarian and King. Using lighting and shaders the game should look attractive but not be too performance heavy that it detriments the user experience.

R2 Multiplayer Server Application

The server will allow users to connect and be added to a queue of waiting players, this queue will be taking players from the front to pair up and send in to games. Once two players have been paired up they will be given a game ID and communication between them should begin. The server acts at the middle man between the two clients and passes the game board between the two, this way the clients never receive any data from each other and keeps the system secure. Once a game has finished they will both be returned to the main menu of their client, where they may queue back up for another match.

1.3. Process

For the Project Feature Driven Development ethos was chosen to guide development, for a single developer with a set time frame this allows for development that focuses on the key features first. When developing a game there are many features that can be added and the danger is to continuously keep adding extra features that may not directly progress towards the final product. Polish features such as attractive user interface and sounds while not necessary do provide an increased user experience so these need to be prioritised higher than normal, so that the game still gives a good impression.

Using the requirements as a basis, each one had their key features listed, they determined the main parts to focus on. Main priority was to develop the offline multiplayer first on the game client as this sets out most of the game code that could then be used in the online version. The online server application took second precedent due to it requiring a functioning game client to be useful. The user interface was a key component that would normally have been finished towards the end of a Project, but because of the short time frame it was prioritized alongside the multiplayer server.

To record the progress and keep regular backups of all aspects of the Project, version control software will be used and updated frequently. Github [21] will be used as it allows for free private repositories and online backups, issues can also be recorded online for fixing later.

2. Design

2.1. Overall Architecture

The Project is split into two distinct components, the main game client which will features two different modes and a multiplayer server which will allow for communication between multiple clients for online games. Both components are going to be written in C# which allows for them to share some code when it comes to the networking communications between them, there will be a network object that will be the same on both sides to help facilitate the communication.

2.2. Feature List

The Feature Driven Development process that has been chosen means that a list of features should be compiled and ranked in level of importance, this will be the feature list, as programming develops the features should be taken off this list in order of importance. If it is realised that a feature needs to have its priority changed then this must be done to allow the program to be the most functional by the Project deadline.

The importance values that were decided are 'Low' for a feature that is not vital and will not massively change the Project, 'High' for a feature whose exclusion will be noticeable and 'Imperative' a feature that is necessary for the Project to meet the major requirements.

Task ID	Task	Importance	Estimated Time to Complete
1	Offline Multiplayer – One of the key features of the game, this allows two people on a single device to play against each other.	Imperative	4 weeks
2	Online Multiplayer – The other main feature of the game, two players can play against each other over a custom server.	Imperative	4 weeks
3	Animations – when pieces move, they can be animated to slide to their new locations to add to the visual effect the game provides	Low	1 week
4	Audio Effects – When certain actions in the game take place, audio should be played to enhance the user experience. For example, when pieces are moving or taken and when UI elements are clicked.	Low	Less than a week
5	Visual Effects – When pieces are selected they can change colour to visually show this, the 3D view can contain post processing effects to make it look more attractive.	Low	Less than a week
6	User Interface – Even though a basic user interface will be mandatory an attractive user interface will heighten the user experience and make the game more enjoyable to play.	High	1 week
7	Multiple Board Sizes – allow for the user to be able to play on more than just the standard board size, this will require creating a new layout for all pieces. If more than two board sizes are implemented this could be made easier with an algorithm.	Low	2 weeks

2.3. Detailed Design

2.3.1. Game Client

When in a game the board will be viewed from an angle allowing a 3D view of the board, this will obscure some pieces on the far side so there will need to be a method of rotating the board to get to the other side. The easiest way to rotate the board will be through on-screen buttons, but for Android users, a more intuitive way will be to drag the screen from side to side. This method would not work on Windows however, but using arrow keys could be a better solution for those devices. The ability to zoom in and out could also be implemented, but that is not a priority and will only be added if it becomes necessary or the mains features are finished.

Moving pieces around the board will be a main aspect of the interaction of the game, firstly the pieces themselves will need to be interactable so that a player can choose which piece they want to select. Once a piece is selected the user will need to know where they can move it, even though the movement constraints are straightforward we can’t trust the user to know the rules perfectly. Highlighting spaces on the floor where the piece may move will show the options available to the user, there are a couple of options for moving the piece to the chosen location. First the user could tap on the space they want to move to, the other is they can drag the piece onto the space. The later will be more complex so initially tapping on the desired space will be easier to implement, dragging will be another luxury feature that could be added at the end. Another luxury option would have the piece animate to the space selected, such as a slide, this will would look good as the pieces can only move in straight lines so there would be no need to awkwardly move around pieces.

2.3.2. Server Application

The server will be made in Visual Studio as a WPF (Windows Presentation Foundation) application, this will provide a user interface for the server operator to interact with. The server will act as a middle man between clients and will pair them up for games, during a game the board state will be passed between the clients through the server.

There will be a log manager in the server that will handle all the messages that it receives, this will be output to a visual display on the user interface and to a text file. The log text file will allow the server operator to read back through the log when an error occurs, for this to function effectively messages must be located all over the program to be output when a problem arises. Whenever a try-catch block is used the resulting error will need to be output to the log as these will be most likely errors that may occur.

2.3.3. Network Code

In both applications there will need to be some shared code to allow the networking to function, this was one of the main factors when choosing to use C# for the server programming.

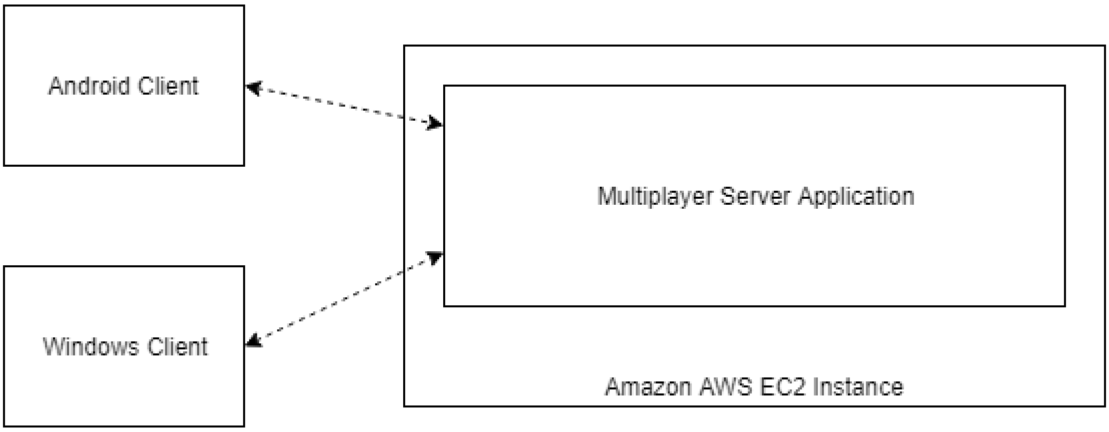


Figure 12

Figure 12 shows an overview of how the online system will work, clients will communicate with each other through the server application. The server will be open to the internet and will continuously accept connections from clients. The game client will be published for both Windows and Android but they will be able to communicate with each other allowing cross platform play.

Network Loop

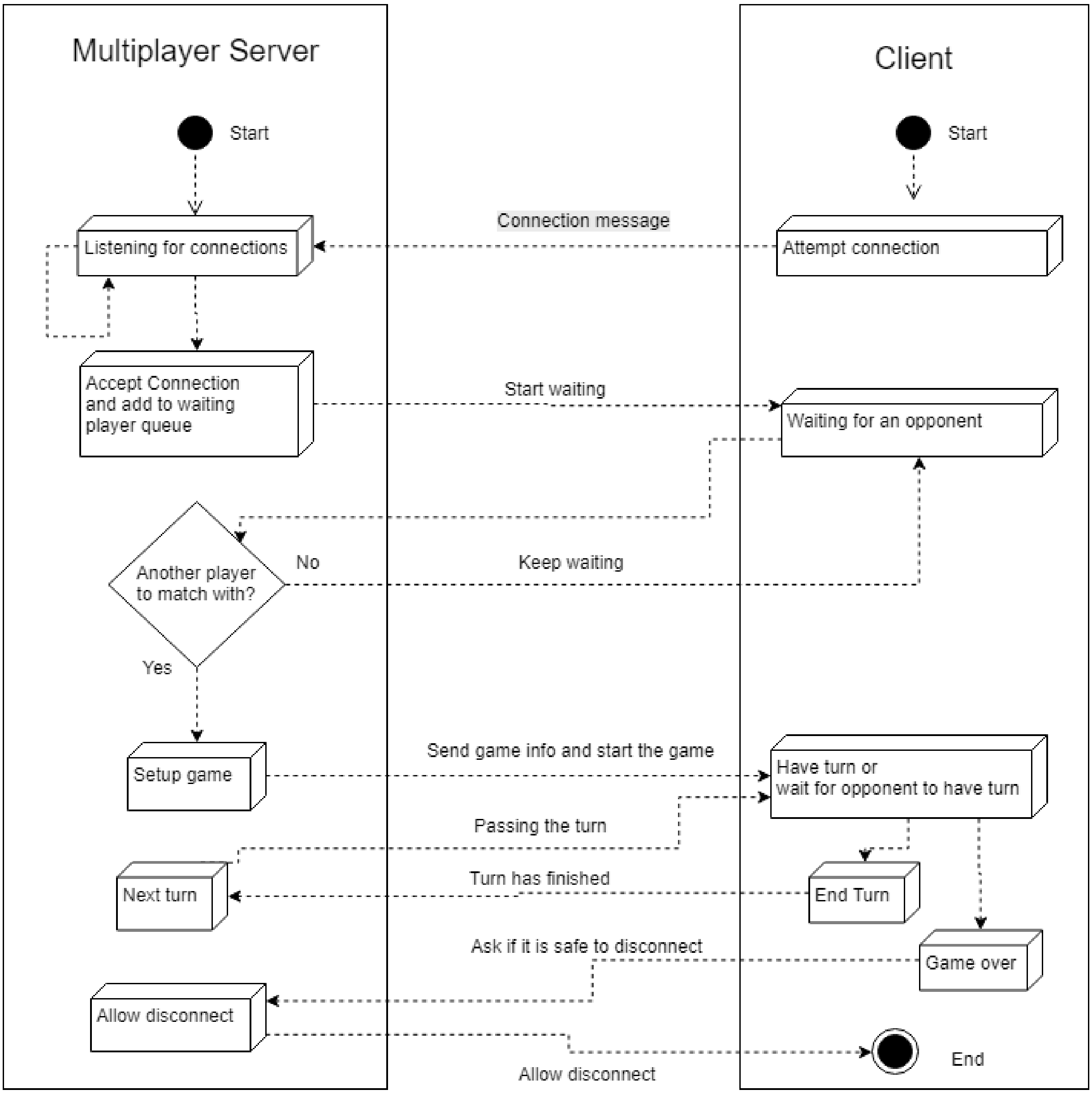


Figure 13

Figure 13 shows how the interaction between the client and server should function. The server will continuously listen for incoming connections from clients, once a client has attempted to connect it will be added to a queue of waiting players. The player will have to wait until there are at least two people waiting and then the server will pair them up, this is treated as a queue and theoretically could have lots of people waiting if the server is overloaded, but realistically as soon as it has two users they will be taken out and the queue will never get above 3.

The connection uses a couple of loops, first when the player is trying to be paired up with another player and secondly when the game is running. During the first stage the client will send a message to the server to see if a second player has been found, the server will either respond with the information to start the game or will respond saying that the client needs to wait a bit longer.

The downside with multiple client servers is that a constant connection is hard to maintain when more clients connect, a better solution is to remember a client but instead of continuously sending data back and forth only give it to them when they ask for it. In this diagram the client will wait before asking if there is another player available, while they are waiting they are leaving room for the server to manage other connections that are of higher importance. If there was not wait time the client and server would send messages between each other as fast as the network and their hardware would allow.

The other loop appears when a game is active between two clients, when a game is setup both players will be sent information about the game, specifically who will go first. The second player will now have to keep asking the server if it is their turn, much like the waiting for players a wait time will need to be implemented so that the server is not overloaded. The first player will take their turn and then say to the server they are done and send the board over, the server then changes its reference to the game to say the turns have changed and the new board configuration. When the second player asks now, the server will respond telling them it is their turn and what the board now looks like. The downside of this method is that there is no checking from the server to make sure what the clients are sending it is a valid board, they could be cheating. This could be remedied by having checking code on the server making it authoritative, but the development of this feature is of low priority.

Network Message Object

To help transfer information between the server and the client a class will be used on both sides, this will provide a common structure for sending and receiving messages. Both programs will be using C# so this class will be identical, it will contain methods to convert an object into a string representation and a method that can convert a string into an object of the class.

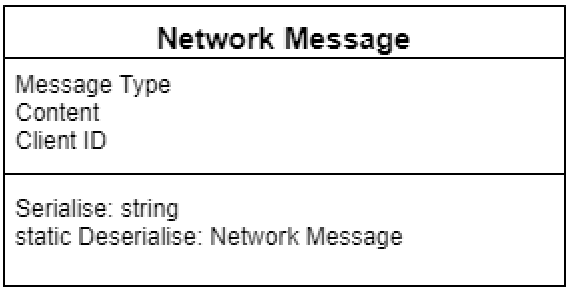


Figure 14

Figure 14 shows the variables that the class will contain, the Message Type will be a custom Enumerator that will be used to choose how to handle the message when it’s received. The content variable will just be a string and will not always be used, when messages are sent to connect, disconnect, find a game they will not need to transfer any content. The main time the content will be used is when transferring the board state between players during a match, the game board will be converted into a series of coordinates and piece types this will then be converted to a board on the receiving end.

2.4. User Interface Design

2.4.1. Game Client

The user interface for the game client should be simple and useable on a touch screens because it will be running on Windows and Android devices, for this purpose large buttons are useful. There will only need to be three buttons on the main screen, one will take the user to offline multiplayer, another will transition to online multiplayer search and there will be a help button to show the user a screen in which they can learn the rules of the game.

Unity engine provides a drag and drop method of creating user interfaces and has many settings to customise elements that are put on to the screen, an important feature that will help with developing for Windows and Android is UI Scaling. There are different settings for how the user interface should scale on different size devices, the best method for the Projects application will be to scale each element to scale with the screen size [20].

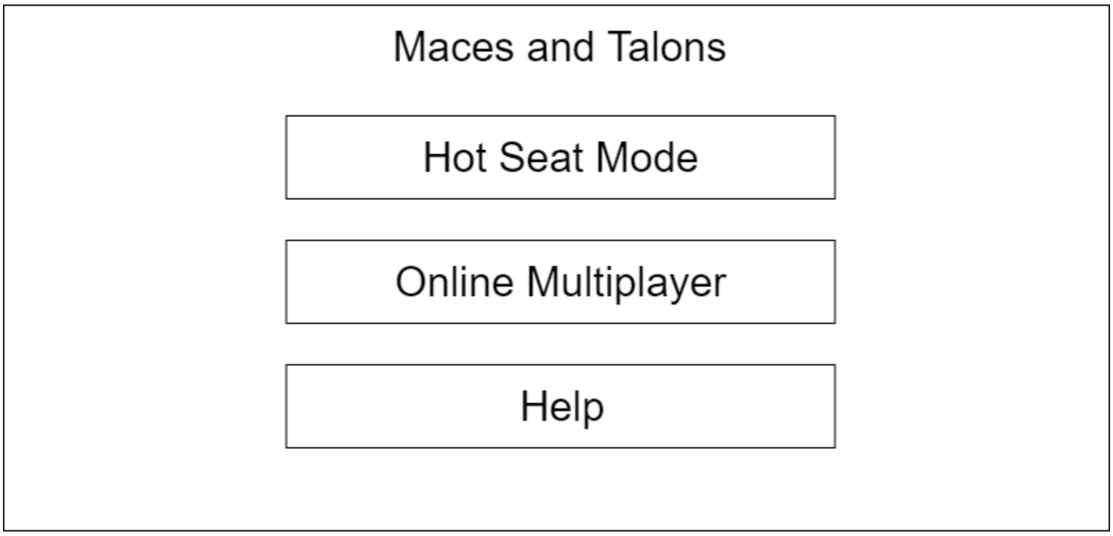


Figure 15

Figure 15 shows a brief design of the planned user interface, the three buttons take up most of the screen for easier clicking. This could appear quite large on Windows computers so a different version may need to be created to fit better on a larger screen, but using the Unity scaling it will still work on many devices. For Windows devices there would probably have to be a button that allows the user to exit the application in case they are running in full screen mode.

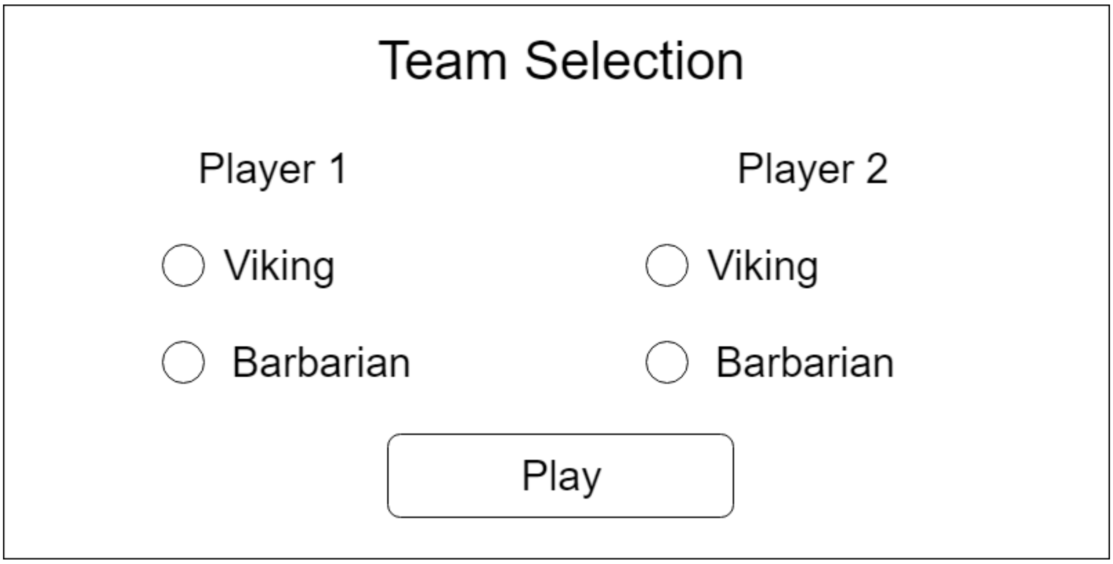


Figure 14

When the user selects offline multiplayer both players will need to select which team they want to be on, this screen (Figure 16) will have radio buttons for each player to allow them to choose their team. Using radio buttons, when one player selects a team the other player should automatically have the opposite team selected. Having a system in place to stop both players having the same team selected will stop their being any errors when trying to move to the game. When the play button is clicked the players will be taken to the game view, unless somehow, they have the same team selected in which case they will be told to pick different side, or if neither have selected a team.

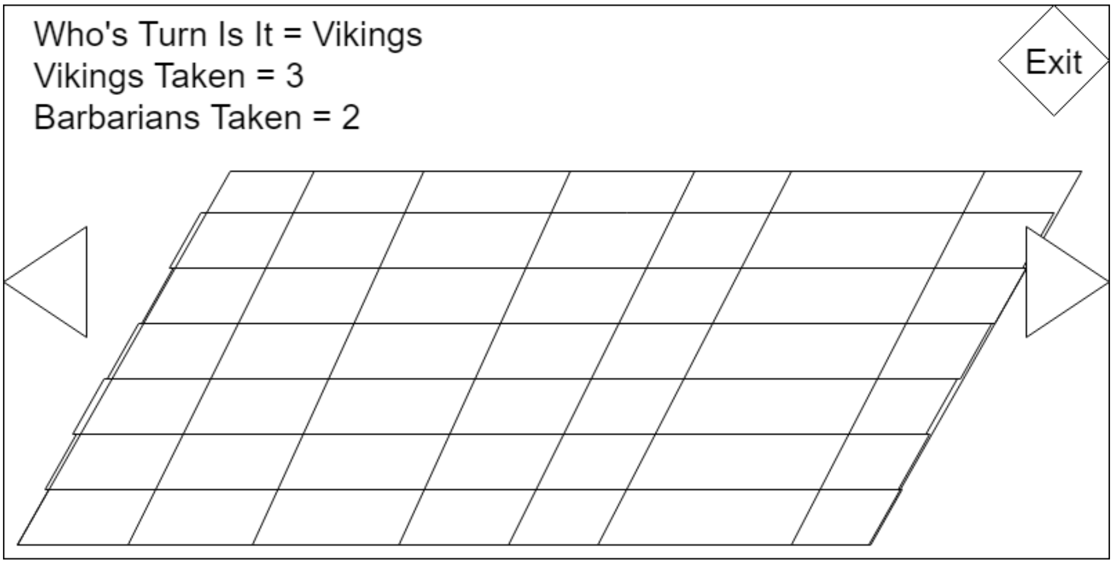
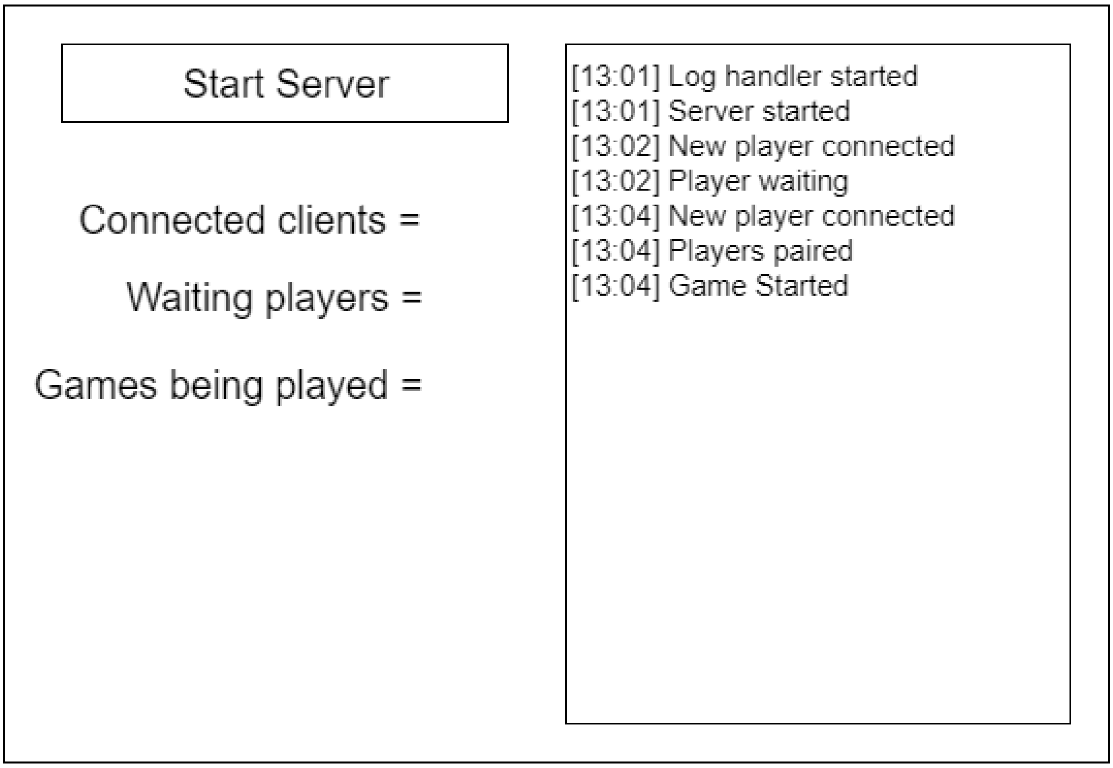

Who's Turn Is It = Vikings
Vikings Taken = 3
Barbarians Taken = 2

Figure 137

The main game view (Figure 17) will be where the actual board game is played. Most of the screen will be taken up by the board and pieces, players will select pieces from their team and move them around. To get a better or different angle of the board there will be buttons to allow the board to rotate, this will give a different vantage point for the players to select their pieces and movements. Information will be displayed to the users, such as who’s turn it currently is and how many of each side have been taken from the game. There will be a button that will allow quitting to the main menu at any time, this will not save the state of the board.

2.4.2. Server Application

The server application’s user interface will show information that will be useful to the operator such as how many clients are connected and how many games are currently being played. Figure 18 shows an example of the layout that would be used, it shows the log manager and how it will display different messages related to the operation of the server. If anything goes wrong with the server the messages in this log could be useful and as discussed in the server design, they will be saved to disk as log files.



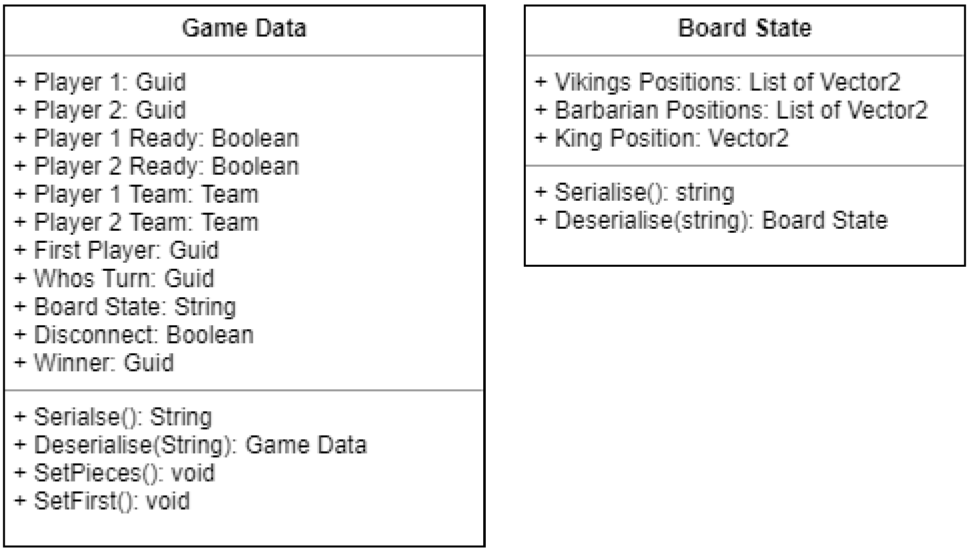
```
[13:01] Log handler started
[13:01] Server started
[13:02] New player connected
[13:02] Player waiting
[13:04] New player connected
[13:04] Players paired
[13:04] Game Started
```

Figure 15

Unlike the game client there will be no other screens for the server to display so implementing this on the server will be not require much time. One of the tools that will be used is Visual Studio 2017 [18], this provides a graphical editor for creating user interfaces and will allow rapid development of the server UI.

2.5. Mid-Project Design

During development of the Project there were aspects that had been overlooked or had not been address in the original design phase, it was then necessary to plan how these features would be structured. This was not a concern because the Feature Driven Development process that the Project was following allows for iterative planning and implementation.



Game Data

+ Player 1: Guid
+ Player 2: Guid
+ Player 1 Ready: Boolean
+ Player 2 Ready: Boolean
+ Player 1 Team: Team
+ Player 2 Team: Team
+ First Player: Guid
+ Whos Turn: Guid
+ Board State: String
+ Disconnect: Boolean
+ Winner: Guid

+ Serialse(): String
+ Deserialise(String): Game Data
+ SetPieces(): void
+ SetFirst(): void

Board State

+ Vikings Positions: List of Vector2
+ Barbarian Positions: List of Vector2
+ King Position: Vector2

+ Serialise(): string
+ Deserialise(string): Board State

Figure 16

Figure 19 shows the two main classes that had to be designed for use on the server and the game client, they will help with formatting data about a game being played between two clients and allow the server to attach appropriate data to the reference such as if a player has disconnected. Being able to tell a player whether the other player has disconnected or won the game will allow their client to display relevant messages and then leave the game view.

The Board State class is needed to convert the positions of all the pieces on the game board into a series of coordinates and piece values separated by a value which can then be taken out such as a '/' (forward slash), the server will never have to handle the board state it just needs to pass it within the Game Data in its string representation.

3. Implementation

3.1. Game Client

Most of the games client’s implementation followed the design very well, there was some user interface changes once testing had been done and better ways were found to create it. Originally when selecting a team in the offline mode there were going to be drop down boxes where the two users would select which side they were on, this felt unnecessary and for only two selections was inconvenient. It was therefore changed to be a set of buttons that toggle the other players team when one person selects a team, this kept the two teams opposite and no problems with both people selecting the same team was possible.

3.1.1. User Interface

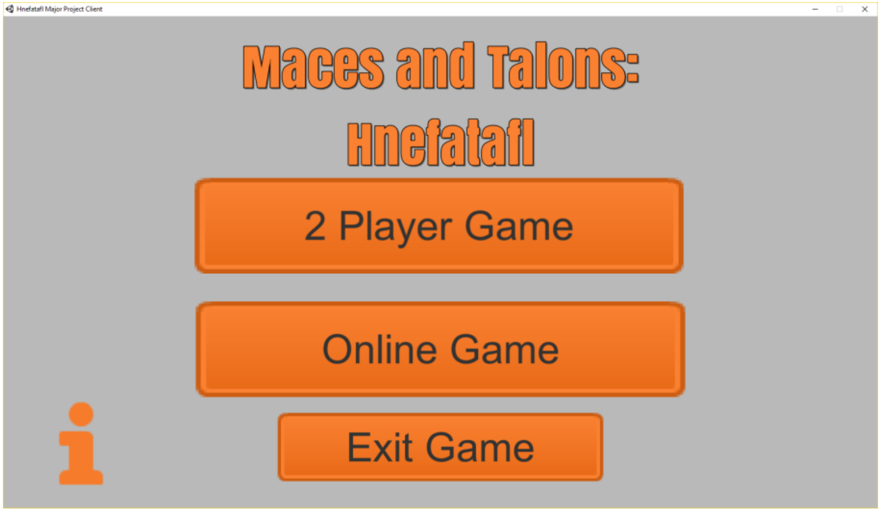


Figure 20

The user interface (Figure 20) matched the sketches from the design quite closely, the main menu had slight alterations moving the help button to an icon in a corner and an exit button to help Windows users quit. All the user interfaces in the game client use art assets from the Kenney Game Asset pack [19], this is a large collection of game art, but only a few icons are used for menus.

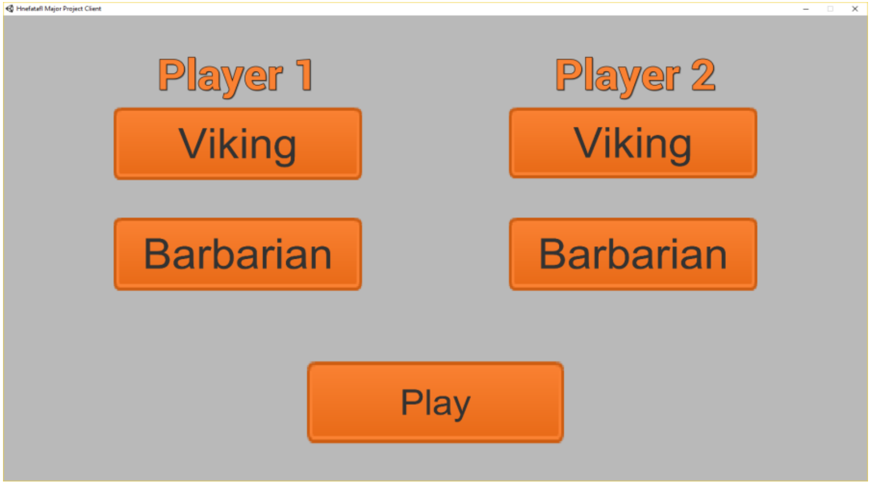
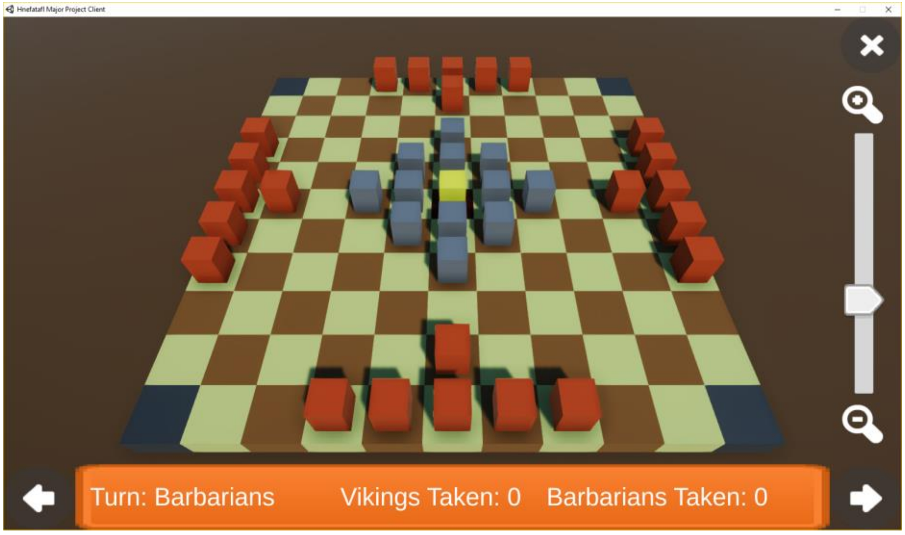


Figure 21

The team selection view changed several times, the design suggested radio buttons, but it was initially implemented using drop down boxes, these had poor interaction on touch screens so were changed to a custom toggle switch system which switch between the teams and through the use of sound provide a satisfying click (Figure 21).

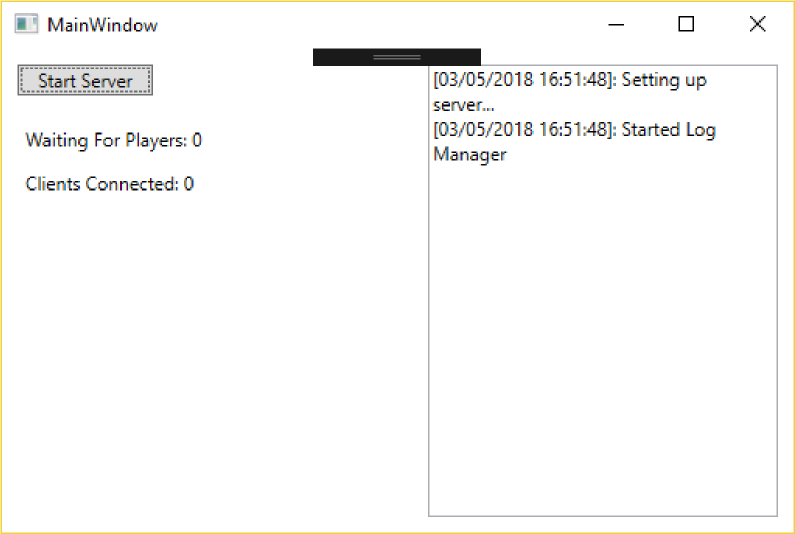


Turn: Barbarians Vikings Taken: 0 Barbarians Taken: 0

Figure 22

The game view (Figure 22) user interface moved the game information from the top left corner into a banner on the bottom of the screen, the arrows to rotate the screen were also moved to the bottom in line the with information panel. A zoom control was theorised in the design but had no physical drawing made of its appearance or where it would be placed, it fits between the exit button and the right rotate arrow, it is easy to user on Android and Windows.

The server had a user interface until a point in development where the application had to be changed into a console application, so that it could be executed on more operation systems, the user interface that was created (Figure 23) held a strong resemblance to the planned version. Implementing the log display and player connection information, as well as a button to allow the starting of there server.



Waiting For Players: 0

Clients Connected: 0

```
[03/05/2018 16:51:48]: Setting up server...
[03/05/2018 16:51:48]: Started Log Manager
```

Figure 23

3.1.2. Game generation

In both the offline mode and multiplayer mode there is quite a lot of code that is reused, the offline code was implemented first but with the aspect that it could be used later for online. Everything is separated into classes, which allowed for easy modification and creation of child classes when making the online multiplayer code.

Originally at the beginning of the design phase there was the possibility of including more than one size of board, because Hnefatafl can be played on a 11x11 or 13x13 board. This meant that an algorithm was designed to allow a board of nth width to be generated if n is odd.

To go with the board generation, pieces are generated into their position as well, this is where the limitation of board size was introduced. The 11x11 board was created first and the pieces are setup specifically for that board type, to create the board, a class was implemented that creates the physical tiles. A reference is kept in the main game class so that when pieces move to a new tile it can be checked to see if it is a corner or throne tile. When the board size was going to be variable this would have allowed for checking to still work because the corners would move, a rigid coordinate system wouldn't have made this work.

3.1.3. Taking pieces

A large technical part of the code was detecting when pieces had been taken by the other team, the location of pieces are stored in a two-dimensional array that represents the game board. From this the game can detect where a piece is in relation to other pieces, this allows for quick checking.

During the development of this code there were lots of out of bounds problems, due to some pieces on the edge of the board. When a piece is checking is it has been taken it looks at all four of its sides, initially this meant that a piece on the edge of the board would try to check a space that didn't exist off the edge. Four checks had to be added depending on which edge of the board a piece was on, that allowed the algorithm to skip the empty space. What made it harder was that normal pieces

(Vikings and Barbarians) can't be taken by a theoretical enemy in the space off the edge, but the King can, which meant that different code had to be used for checking if the King had been taken.

3.1.4. Selectable Pieces and Movement Tiles

Making a piece selectable was made straight forward by Unity, the engine has a feature to detect which point on the screen the mouse is over and can also detect screen input. Using an event function 'OnMouseOver' whenever the user is hovering over the desired piece they can click or press the screen and the game will check if that is a piece may be selected. This is a section of code that had to be modified when adding the multiplayer aspect, the game reuses code for both offline and online. If it is an online game the Network Manager needs to be referenced and the team can be accessed, if the game is offline a simple toggle is used to just flip between team when one makes a move.

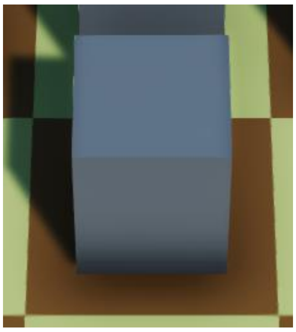


Figure 24

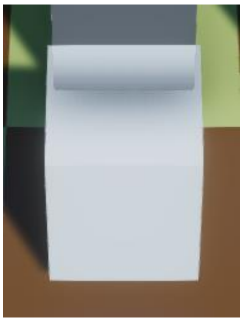


Figure 25

To show the user whether they have selected a piece the game changes the colour of the piece, using a Unity Store package MK Glow Free [1] this provides a variety of custom shaders for use in a game without having to spend time understanding how to create custom shaders. When used the piece looks like it has been lit up when selected, Figure 24 shows a piece that has not been selected and Figure 25 shows the same piece that has now been selected. When planning out how to show the piece selection other methods such as changing the colour of the tile the piece was one were considered, however they did not stand out as much as lighting up the piece.

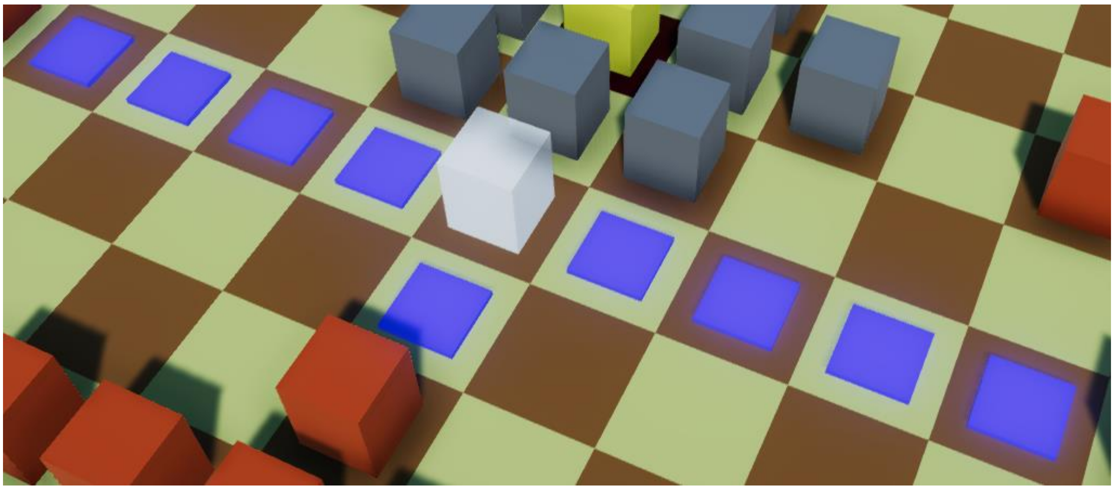


Figure 17

Once a piece has been selected, it needs to move. This is achieved by generating tiles for the user to click when a piece is selected, this can be seen in Figure 26. By then clicking on these tiles, the piece will move to the selected location. To generate these movement tiles the game must check what possible moves can be made by the selected piece, there were several problems with this feature, when pieces were being moved they were not correctly changing their location in the array representation of the game.

The board can be thought of in two ways, there is the graphical representation that the user sees and the code array that's stores the locations of the pieces. If the two representations do not get updated together it can lead to problems and bugs, this happened when the graphical level was changed so the piece had appeared to move but in the code it was still at its original location. This meant that when another piece looked like it could move over a tile that was previously occupied it couldn't, because the space was technically still occupied. It took a while for the source of this problem to be located, but once both representations were in sync the movement functioned correctly.

3.2. Multiplayer Server

There are two versions of the server, one provides a graphical user interface while the other operates in a console window. These two versions provide the exact same features to the clients, they only affect the information shown to the operator of the server.

The graphical version uses WPF (Windows Presentation Foundation) forms to create a look that is identical to most standard Windows applications, this provides features that allowed the creation of a real-time log that outputs the messages the server is handling. A couple of displays will also show the number of users connected to the server and how many are waiting for a game. The console mode has no graphical user interface and just displays the real-time log in the console as messages get processed, otherwise the functionality of the server is the same.

There was point where the server was going to be stopped because the user interface had an estimated development time and the server was very close to cutting into that time, which would have led to neither being finished. Fortunately, the server was completed within its estimated development time so both features were finished.

3.2.1. Setting up games

When a user connects to the server they are placed at the back of a waiting queue, this contains all the clients currently waiting for a game. When the queue contains two or more clients, two are taken from the front and a game is generated. The server then randomises which team each player will be and who will go first, this is all added to an object that stores a reference to all information about the game (the players, who's turn it is, whether anyone has won, if a player has disconnected).

Once the game has been setup on the server the clients are informed that a game has been found and the game is transferred to the client, they will know whether they are first by reading the game data and checking the first player with their own ID, if they match then they will be going first.

To play a game the clients send the state of the game board in a message when passing the turn, this is then relayed by the server and the next client replaces the pieces in their game with the new positions that it received.

3.2.2. Ending games

When one client has won the game, their client will show a victory message and allow them to quit, during this time a message has been sent to the server that will say that the game has been won. The server will now start to clean up, it changes its reference to the game to say that it has finished and informs the other user that they have lost. At this point the server disposes of the reference to the game and removes the connection variables of the two clients, as when a game ends both users are sent back to the main menu so are effectively kicked from the server.

3.2.3. Server Hosting

When testing the server during development it could be run on the same machine as the test clients and connected to using the machines local IP address. However, when larger testing needed to happen over the internet it had to be deployed to a server, it could have continued to be run on a development machine but opening it up to the internet would not be ideal for security reasons.

A better solution was to upload it to an existing server hosting service which would increase security and allow for a more reliable uptime that didn't rely on the development machine stay powered on permanently. The hosting service chosen was an Amazon AWS EC2 [25] container which provides a free tier with low performance, but with the size of testing that would occur during the Project this was sufficient.

Initially a Windows server instance was used but unfortunately the free tier of EC2 did not have enough performance to run the operating system, let alone the server. A Linux version had to be used instead, however this lead to its own problems because the existing server application was designed to run on Windows. The original application would not run on Linux, fortunately C# can be run on Linux using the Mono library [15] (a cross platform .Net framework). Mono allows C# programs that do not use any Windows specific features like WPF (Windows Presentation Foundation) forms or WinForms (Windows user interface libraries).

To create a server that works on Linux all of the WPF elements had to be stripped from the existing server application, this was easily done by creating a completely new console application then transferring the necessary code to the new application. The new application works well on the free tier of EC2 and can handle multiple connections.

3.3. Network Code

Some sections of the code that makes the online mode work is shared across the server and client, combined with the use of C# in both implementations this allows for a robust system that allows message to be sent over the network. The server implements some features that allow it to authenticate the messages the client devices send and uses a system that will enable large scalability.

3.3.1. Network Message

The main class that defines a message between the server and client is the Message class, this defines values for the message that needs to be sent, user id, client id and the type of message. When a message is sent from the server there will not be a user id associated with the message because that is only used when authenticating a client message 3.3.3.

This message class provides the ability to convert an instance of the class to a string representation to then be transferred as a series of bytes over the network, this is the serialise method. When a message is received it must be converted back to a message object and this can be done statically through the use of the deserialise method, this will take the string and split it up into the variable components then convert them back into the types that they were previously in. Using this method of serialising and deserialising it is possible to maintain a standard way to convert the message on both the server and client.

3.3.2. Connection and message types

The connection type used by the server is TCP allowing for messages to be ordered and a more reliable connection, the way this then operates on the server is through a multithreaded asynchronous connection. The server is constantly listening for new connections from a client, when a device attempts to connect, the server adds the client to a list of connected clients. The game client generates a device id to that it maintains while the game is open, this is stored as a GUID (Globally Unique Identifier). GUIDs are a large 128 bits value that has a practically infinite number of combinations, there is an insignificant chance that there will be two clients with the same GUID [14].

Microsoft has examples for ways to setup server code, the initial code used to make the server client connection work, was code from one of these examples [13]. This code then had all unnecessary features stripped from it and it was also simplified, it used several different methods that could be combined and simplified to make the code more readable. Once there was a basic server client infrastructure implemented messages could be sent back and forth using the message class. One of the values stored in the message is a message type, this is used to determine how to handle the data that has been sent in the message. Through use of these message types a large switch statement allows for handling the messages and send an appropriate response.

3.3.3. Authentication

The server authenticates user messages in a couple of ways, first using the message class and using a combination of ids. If a message is sent to the server and it doesn’t correctly deserialise into an instance of the message class then either the message is corrupt or the message was not sent from an official client. If it doesn’t deserialise correctly it will be discarded and no further processing will occur.

Once a message has been deserialised it must then be confirmed that it has been sent from an authorised client, when a client connects it is given an id by the server this id must be sent with every message the client wants to transmit. If the client sends a message and the id does not match the servers stored value then the message is disregarded, this stops third party programs from connecting to the server even if they discover how to correctly format a message.

4. Testing

4.1. Overall Approach to Testing

The Project used several methods for testing, first was the continuous testing of components during development as they were implemented, to not allow the program to get riddled with serious bugs. Development would halt until a bug had been fixed, this process allowed for a much more successful testing at the end of the Project. At the end of the Project black-box testing was employed to develop a thorough check list that the features worked as they were supposed to. The final testing was User acceptance testing, as the Project is a video game application, it is ideal that it be tested by an audience to get initial reactions.

4.2. Testing During Development

Testing while developing is essential to make sure newly implemented features work correctly, if the Project moves onto creating new features while there is known issues this can lead to it being harder to fix later. When a problem needs fixing in the code the location of the error must first be found, this depends on what kind of error has occurred.

If a runtime error occurs Unity and Visual Studio have inbuilt error reporting, but they do not always point to the exact problem. This is when the use of messages in the suspected area help, by placing outputs to a console log at different stages it can be established how far the program gets before crashing.

The other main error is a logic error, these are the hardest to find as they will allow the program to run but will not function as intended. Much like runtime error locating, the best method for finding the source of these errors is a good knowledge of where events happening in the code. This can help narrow the search down, and then through the use of console message the exact point can be located.

4.3. Black-box Testing

There are two main aspects to black box test, the functionality of the game client including the menus and offline mode and then the server application which will also require testing the online mode of the game client.

4.3.1. Game Client

The following tests will cover the game client menus, game scenarios and game user interface.

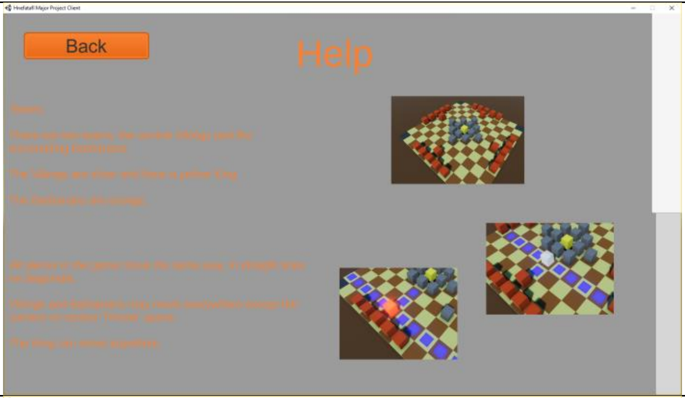
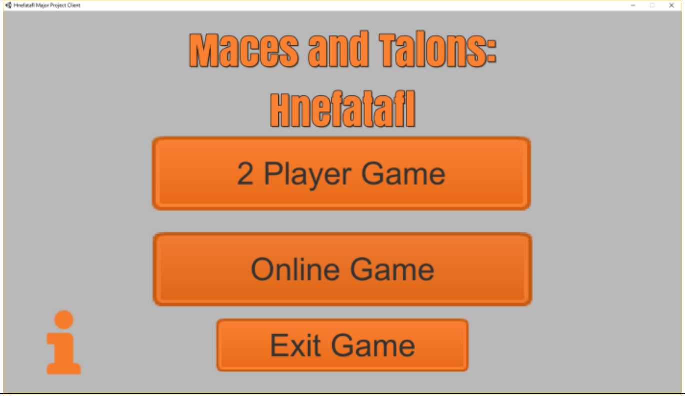
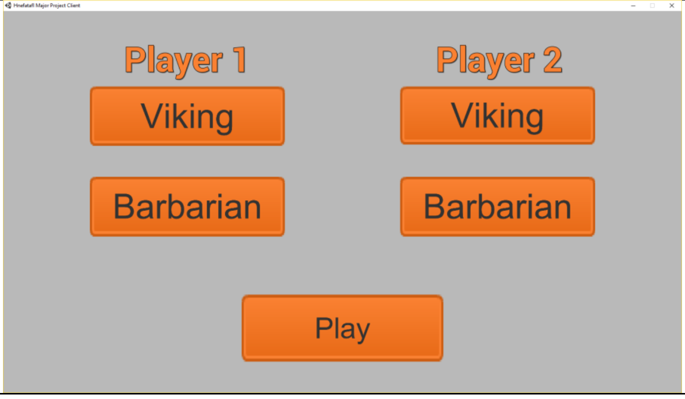
Test Outline

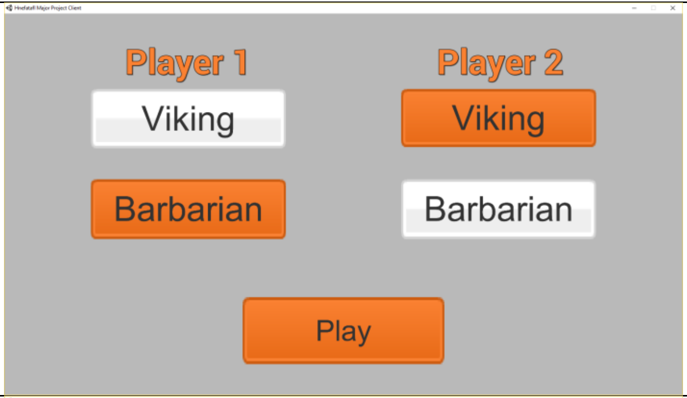
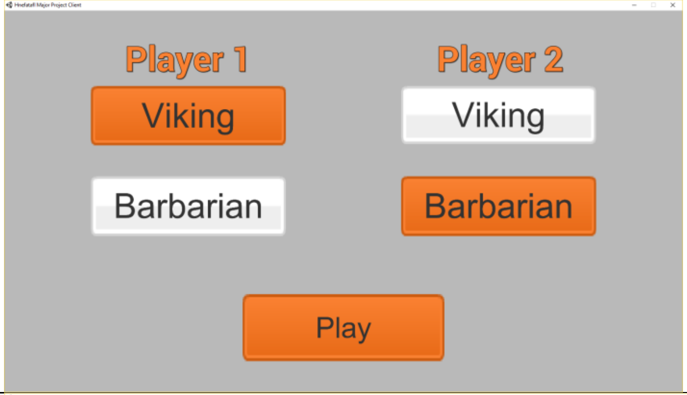
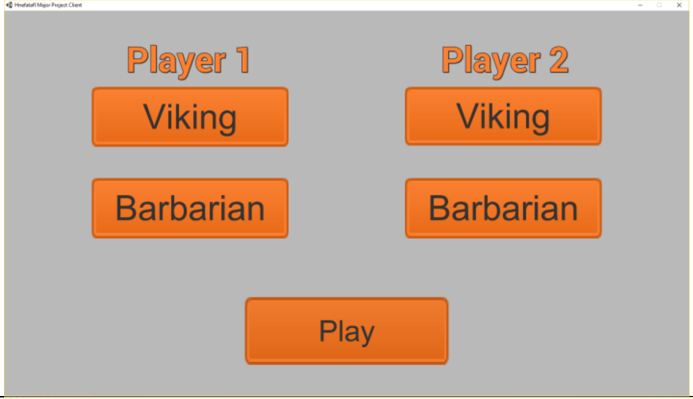
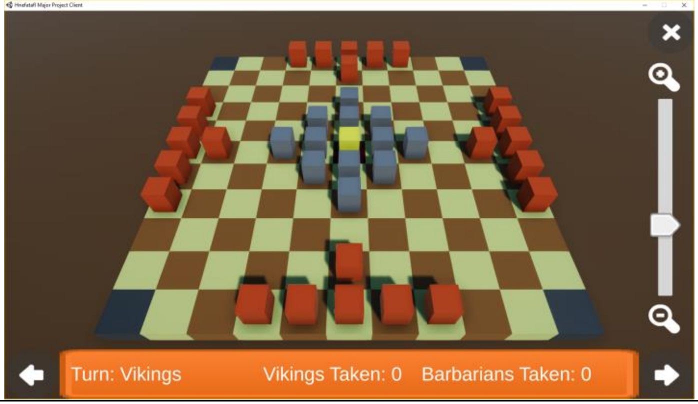
Test ID	Test Description	Expected Results
1	Clicking information button.	Help screen is displayed.
2	Click back button of information screen.	Returns to the main menu screen.
3	Clicking offline game button.	Taken to team selection screen.

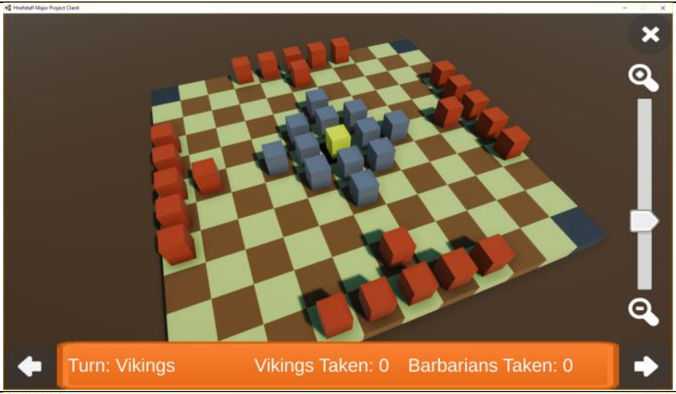
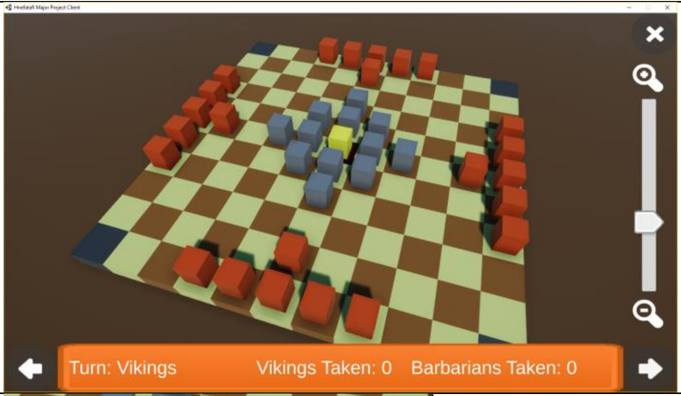
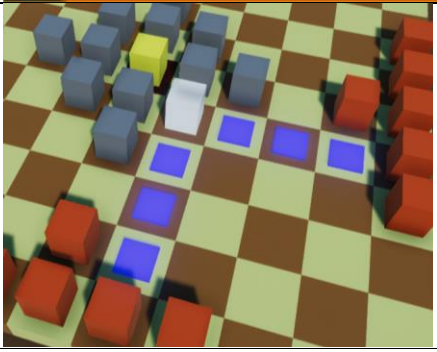
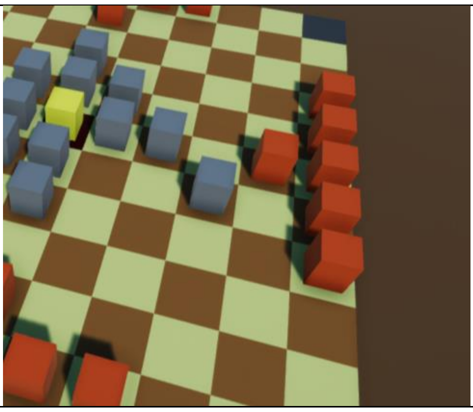
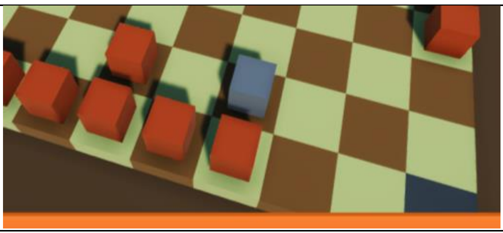
4	Clicking Viking on Player 1.	Viking for Player 1 should be highlighted and Barbarian should be highlighted for Player 2.
5	Clicking Viking on Player 1 while selected.	Viking for Player 1 should become unselected and Barbarian for Player 2 should become unselected, then Barbarian for Player 1 is highlighted and Viking for Player 2 is highlighted.
6	Clicking Play while no team are selected.	Should stay on same screen and display an error message saying that no teams are selected.
7	Click Play while teams are selected.	Screen should change to the game view.
8	Clicking Rotate left button.	Rotates the screen left.
9	Click Rotate right button.	Rotates the screen right.
10	Click on piece for current team.	Piece is selected and the possible moves are displayed.
11	Click on piece for current team while it is selected.	Piece should become unselected and movement tiles should disappear.
12	Click on piece for the waiting team.	Piece is not selected and nothing happens.
13	With a piece selected clicking on a movement tile.	Piece will move to the desired location.
14	With a piece selected clicking on any space that is not a movement tile.	Nothing will happen and the movement tiles should still be visible.
15	Have Barbarian against the side and Viking next to it	Nothing
16	Two Barbarians on perpendicular sides to a Viking.	Nothing
17	Two Barbarians on opposite sides to a Viking.	Viking Piece should be taken and removed from the board, the user interface should update the number of Vikings taken by +1.
18	Moving a Viking.	The current turn label should now display Barbarians.
19	Moving a Barbarian.	The current turn label should show Vikings.
20	Two Barbarians on opposite sides of the King.	Nothing
21	Three Barbarians on three sides of the king, with an empty space or Viking on the other side.	Nothing
22	Three Barbarians on three sides of the king and the other side is the edge of the board.	King should be taken and a message to declare the Barbarians the winners should appear.
23	The King moving onto one of the corners.	Message declaring the Vikings as the winners appears.
24	When a non-King piece is selected that is in line of sight of a corner.	The movement tiles should stop before the corner not allowing it to move onto the corner.
25	Dragging the zoom slider.	Game view should zoom.
26	Clicking the exit button.	Text appears and buttons appear to allow the user to quit.
27	Clicking a piece for the active team while the exit menu is displayed.	Piece should not be selected.
28	Clicking the cancel button on the exit menu.	Closes the menu and resumes the game.
29	Click the accept button on the exit menu.	Send the game back to the main menu.

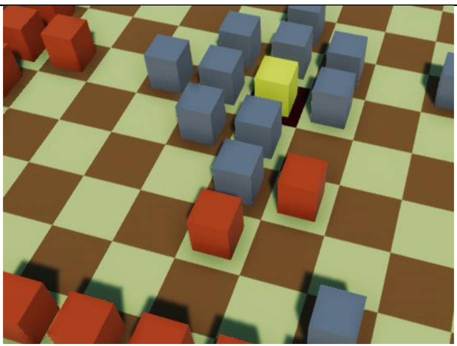
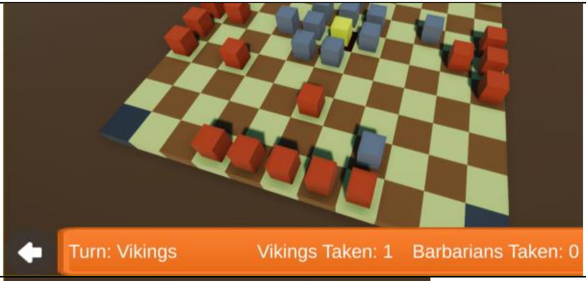
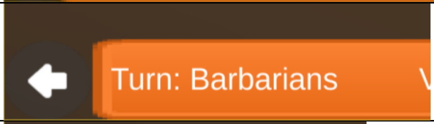
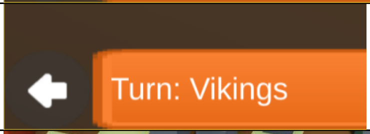
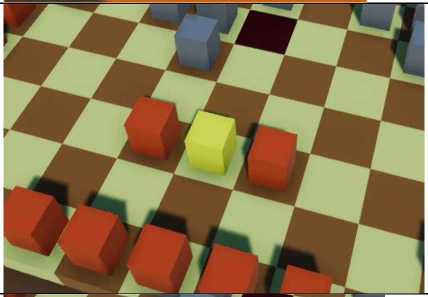
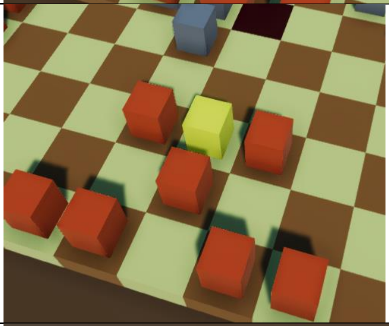
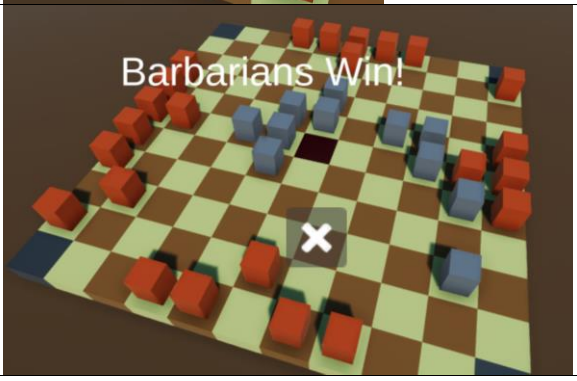
30	Click the accept button when the Barbarians win.	Main menu appears.
31	Click the accept button when the Vikings win.	Main menu appears.

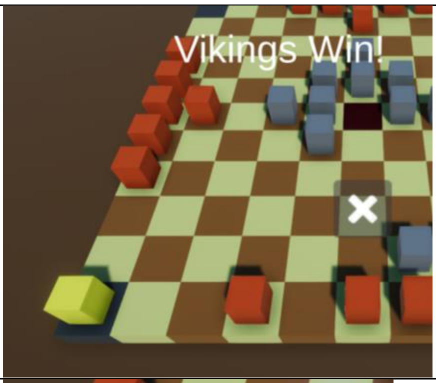
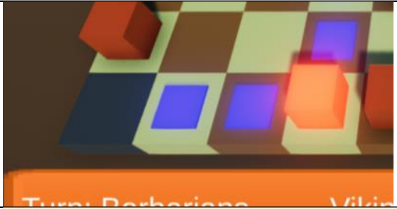
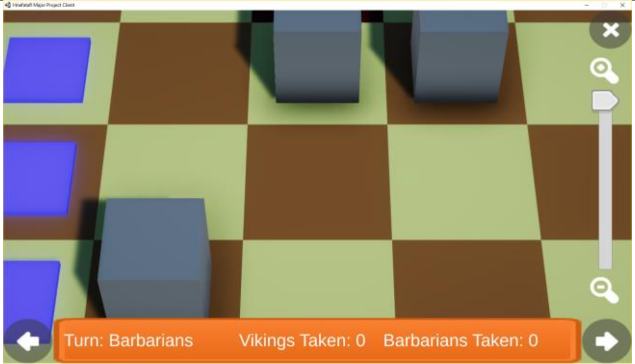
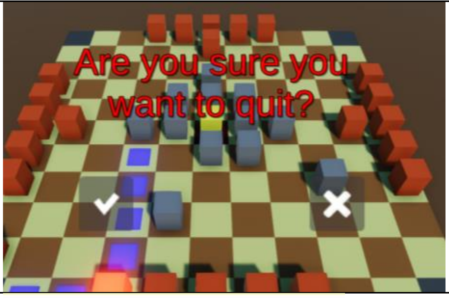
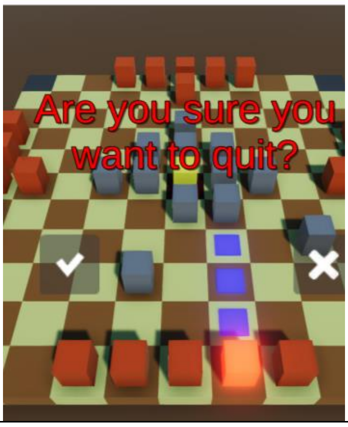
Test Results

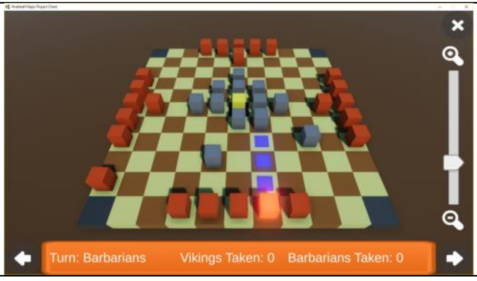
Test ID	Result	Relevant evidence	Pass or Fail
1	Help screen appears.		Pass
2	Main menu is displayed.	 Hnefatafl Major Project Client Maces and Talons: Hnefatafl 2 Player Game Online Game Exit Game	Pass
3	Taken to team selection.	 Hnefatafl Major Project Client Player 1 Player 2 Viking Viking Barbarian Barbarian Play	Pass

4	Viking is highlighted for Player 1 and Barbarian is highlighted for Player 2.	 Player 1 Player 2 Viking Viking Barbarian Barbarian Play	Pass
5	Barbarian is highlighted for Player 1 and Viking for Player 2.	 Player 1 Player 2 Viking Viking Barbarian Barbarian Play	Pass
6	Stays on the same screen but doesn't show an error message.	 Player 1 Player 2 Viking Viking Barbarian Barbarian Play	Fail
7	Screen changes to the game view.	Turn: Vikings Vikings Taken: 0 Barbarians Taken: 0	Pass

8	Screen rotates left.	 Turn: Vikings Vikings Taken: 0 Barbarians Taken: 0	Pass
9	Screen rotates right.	 Turn: Vikings Vikings Taken: 0 Barbarians Taken: 0	Pass
10	Piece becomes highlighted and the movement tiles become visible.		Pass
11	Nothing happens		Fail
12	Nothing happens		Pass
13	The piece moves to the selected location.		Pass
14	Nothing happens		Pass
15	Nothing happens		Pass

16	Nothing happens		Pass
17	Viking was taken and user interface element displaying number of taken Vikings increased.	 Turn: Vikings Vikings Taken: 1 Barbarians Taken: 0	Pass
18	Turn label displays Barbarians.	 Turn: Barbarians V	Pass
19	Turn label displays Vikings.	 Turn: Vikings	Pass
20	Nothing happens		Pass
21	Nothing happens		Pass
22	King disappears and message saying the Barbarians win appears.	 Barbarians Win!	Pass

23	Message declaring Vikings as winners appears.	 Vikings Win!	Pass
24	Movement tiles will not allow movement onto the corner.	 Turn: Barbarians	Pass
25	Game view zoomed in.	 Turn: Barbarians Vikings Taken: 0 Barbarians Taken: 0	Pass
26	Exit menu appears.	 Are you sure you want to quit?	Pass
27	Piece was selected.	 Are you sure you want to quit?	Fail

28	Closes the exit menu and returns to the game.		Pass
29	Main menu is displayed		Pass
30	Main menu is displayed		Pass
31	Main menu is displayed		Pass

Analysis

These results are good, only two fails, both of which are not game breaking but also straight forward fixes.

4.3.2. Server

This section of testing will be used to test the server application and the online multiplayer functionality of the game client.

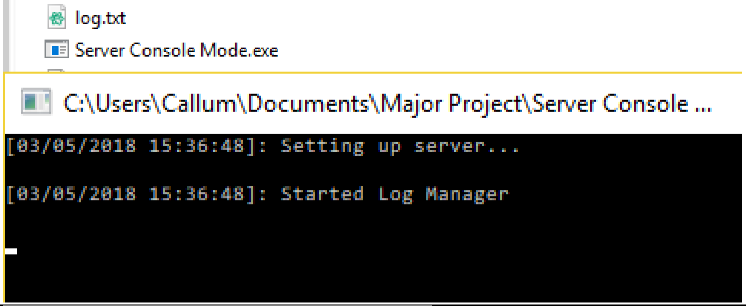
Test Outline

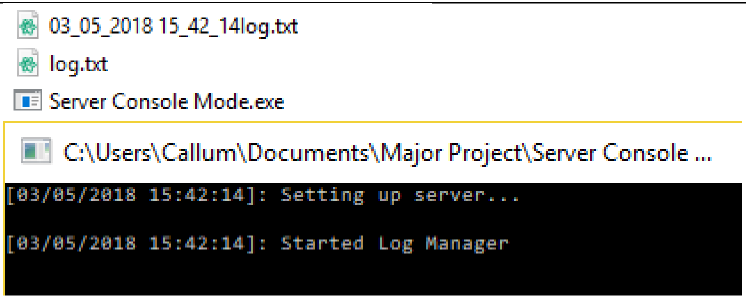
Test ID	Test Description	Expected Result
1	Starting the server without existing log file.	A log file will be generated at the location of the server program. The server will display that the server has started.
2	Starting the server with existing log file.	An existing log file will be renamed with a name related to the current time and date, then a new log file is created.
3	When the multiplayer button is clicked in the game client, while no other users are connected to the server.	Game client displays a message related to waiting for a game/opponent. Server will show that a client has connected and that they are waiting.
4	Leaving a single game client connected to the server waiting for a player for 10 minutes.	Nothing interesting, the server should output that the player is waiting every few seconds and the game client will just keep displaying the waiting message.
5	Game client goes to online multiplayer while there is another client waiting for a player.	The server will accept the connection, for a moment there will be two players in the queue, then the server will pair the two clients up. Both clients will display a message, relating to an opponent/game being found.
6	Disconnecting one of the clients through the exit menu.	Both clients should be returned to the main menu. The server should be sending messages to the client that exited to allow it to disconnect, and tell the other client to leave the game.
7	Disconnecting one of the clients by exiting the application.	The client that exited, closes. The other client gets returned to the main menu. The server should be sending messages to the client that

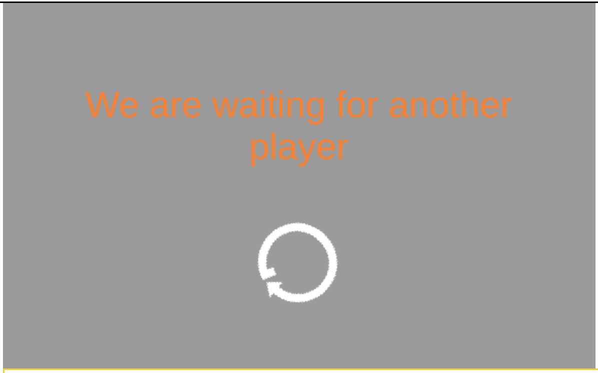
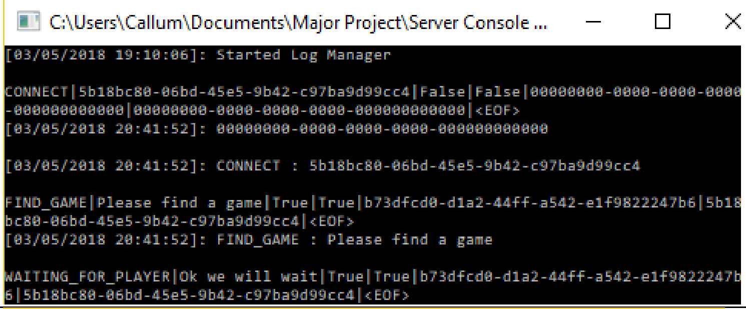
		exited to allow it to quit safely, and tell the other client to leave the game.
8	Moving a piece.	The board of the client that took the move will play like the offline mode, then the board of the waiting client should adjust to show the piece that was moved on the other client. The server will show messages telling the current player that it is now their turn.
9	Taking a piece.	The board of the client that took the move will play like the offline mode, then the board of the waiting client should adjust to show the piece that was moved on the other client. The server will show messages telling the current player that it is now their turn.
10	Winning with the Vikings.	A display should appear to show that the client has won, a message should be sent to the other client to show a losing message.
11	Winning with the Barbarians.	A display should appear to show that the client has won, a message should be sent to the other client to show a losing message.
12	Clicking the quit button on Vikings win.	The client should be returned to the main menu and the other client should be allowed to stay, they should be able to quit in their own time.
13	Clicking the quit button on Barbarian win.	The client should be returned to the main menu and the other client should be allowed to stay, they should be able to quit in their own time.
14	Third client connecting while there are two clients currently connected to the game.	They will just be added to the queue and waiting messages will be sent, much like the results for a single client connection.
15	Server closing while there is a client waiting for a game.	Client should be returned to the main menu.
16	Server closing while there is a game currently playing.	Both clients should be returned to the main menu.
17	Leaving the wait queue by clicking the back button.	Client will return to the main menu and the server will display a message that shows it allowed the client to leave.
18	Leaving the wait queue by closing the application.	The server will remove the client from the wait queue.
19	Clicking the Online Multiplayer button while there is no internet.	A message should appear to warn that the client device has not internet connection.
20	Clicking the Online Multiplayer button while the server is not running.	A message should appear to warn that the server is offline.

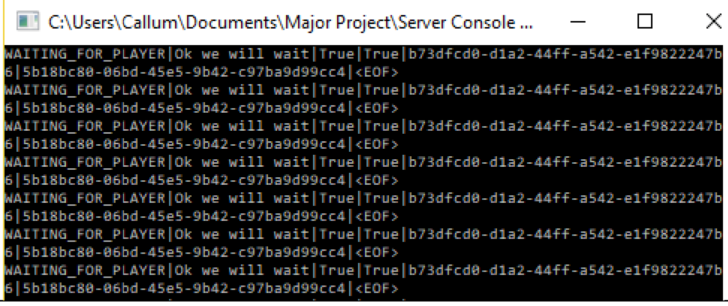
Test Results

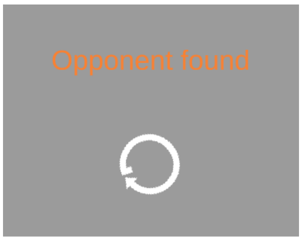
Test ID	Result	Relevant Evidence	Pass of Fail

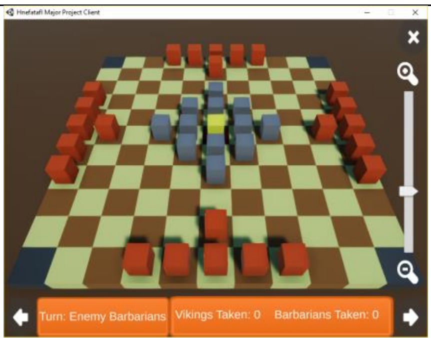
1	Server starts and log file is generated.	
		log.txt
		Server Console Mode.exe
		C:\Users\Callum\Documents\Major Project\Server Console ...
		`[03/05/2018 15:36:48]: Setting up server...`
		`[03/05/2018 15:36:48]: Started Log Manager`
			Pass

2	Server starts and previous log is renamed and new log it generated.	
		03_05_2018 15_42_14log.txt
		log.txt
		Server Console Mode.exe
		C:\Users\Callum\Documents\Major Project\Server Console ...
		`[03/05/2018 15:42:14]: Setting up server...`
		`[03/05/2018 15:42:14]: Started Log Manager`
			Pass

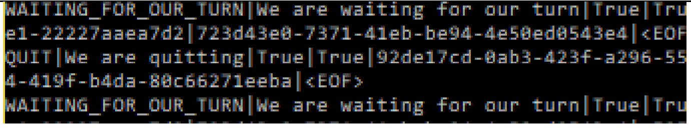
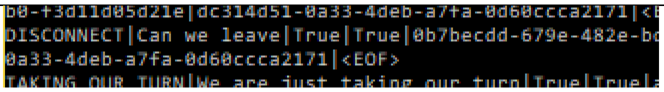
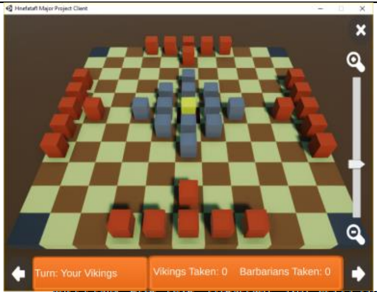
3	Game client shows a waiting message. The server shows messages for the client connecting and sending messages to find a game, this then moves on to messages saying the client is now waiting for an opponent.	
		We are waiting for another player
		
		C:\Users\Callum\Documents\Major Project\Server Console ...
		`[03/05/2018 19:10:06]: Started Log Manager`
		`CONNECT|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|False|False|00000000-0000-0000-0000-000000000000|00000000-0000-0000-0000-000000000000|<EOF>`
		`[03/05/2018 20:41:52]: 00000000-0000-0000-0000-000000000000`
		`[03/05/2018 20:41:52]: CONNECT : 5b18bc80-06bd-45e5-9b42-c97ba9d99cc4`
		`FIND_GAME|Please find a game|True|True|b73dfcd0-d1a2-44ff-a542-e1f9822247b6|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|<EOF>`
		`[03/05/2018 20:41:52]: FIND_GAME : Please find a game`
		`WAITING_FOR_PLAYER|Ok we will wait|True|True|b73dfcd0-d1a2-44ff-a542-e1f9822247b6|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|<EOF>`
			Pass

4	The server kept reporting that the client is waiting, it is the same client because their user id is the same in every message.	 C:\Users\Callum\Documents\Major Project\Server Console ... `WAITING_FOR_PLAYER|Ok we will wait|True|True|b73dfcd0-d1a2-44ff-a542-e1f9822247b6|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|<EOF>` `WAITING_FOR_PLAYER|Ok we will wait|True|True|b73dfcd0-d1a2-44ff-a542-e1f9822247b6|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|<EOF>` `WAITING_FOR_PLAYER|Ok we will wait|True|True|b73dfcd0-d1a2-44ff-a542-e1f9822247b6|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|<EOF>` `WAITING_FOR_PLAYER|Ok we will wait|True|True|b73dfcd0-d1a2-44ff-a542-e1f9822247b6|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|<EOF>` `WAITING_FOR_PLAYER|Ok we will wait|True|True|b73dfcd0-d1a2-44ff-a542-e1f9822247b6|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|<EOF>` `WAITING_FOR_PLAYER|Ok we will wait|True|True|b73dfcd0-d1a2-44ff-a542-e1f9822247b6|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|<EOF>` `WAITING_FOR_PLAYER|Ok we will wait|True|True|b73dfcd0-d1a2-44ff-a542-e1f9822247b6|5b18bc80-06bd-45e5-9b42-c97ba9d99cc4|<EOF>`	Pass

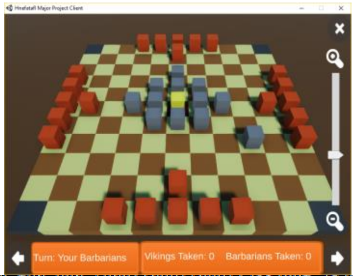
<table>
<tr><td>5</td><td>Both clients show a message related to an opponent being found and are then taken to the game.</td><td>



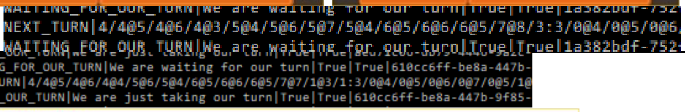
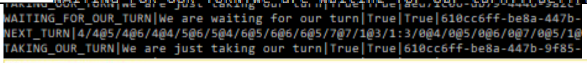
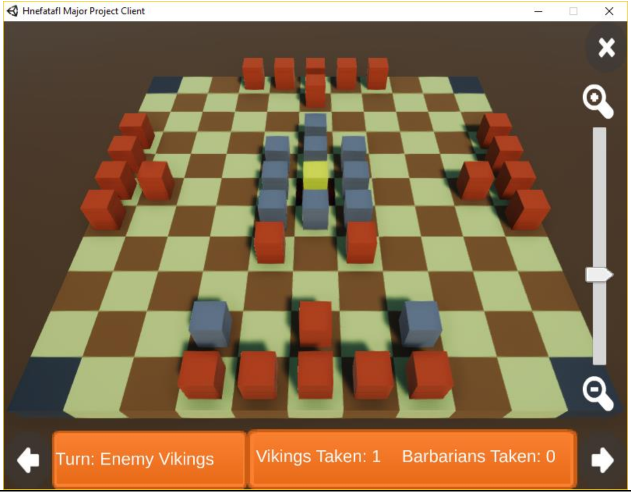
Turn: Enemy Barbarians
Vikings Taken: 0
Barbarians Taken: 0

</td><td>Pass</td></tr>
<tr><td>6</td><td>After a small delay between the two clients, they both exit to the main menu. The server shows the client sent a message to quit.</td><td>
WAITING_FOR_OUR_TURN|We are waiting for our turn|True|Tru
e1-22227aaea7d2|723d43e0-7371-41eb-be94-4e50ed0543e4|&lt;EOF
QUIT|We are quitting|True|True|92de17cd-0ab3-423f-a296-55
4-419f-b4da-80c66271eeba|&lt;EOF&gt;
WAITING_FOR_OUR_TURN|We are waiting for our turn|True|Tru</td><td>Pass</td></tr>
<tr><td>7</td><td>The client closes and the other client goes back to the main menu, the server shows a disconnect message.</td><td>
b0-f3d11d05d21e|dc314d51-0a33-4deb-a7fa-0d60ccca2171|&lt;E
DISCONNECT|Can we leave|True|True|0b7becdd-679e-482e-bc
0a33-4deb-a7fa-0d60ccca2171|&lt;EOF&gt;
TAKING_OUR_TURN|We are just taking our turn|True|True|a</td><td>Pass</td></tr>
<tr><td>8</td><td>The Viking is moved on the Viking team client and then seconds later it is moved on the Barbarian client, the server shows a message passing the turn and the coordinates of the pieces.</td><td>

Hnefatafl Major Project Client
Turn: Your Vikings
Vikings Taken: 0
Barbarians Taken: 0

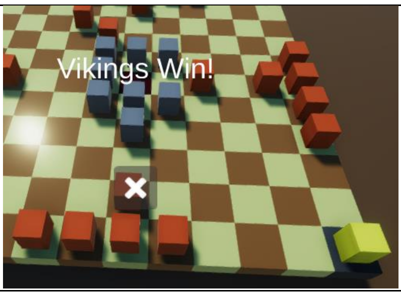
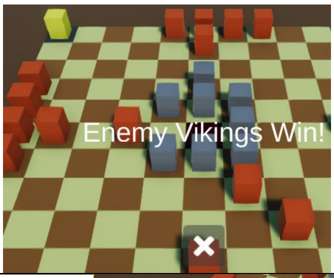
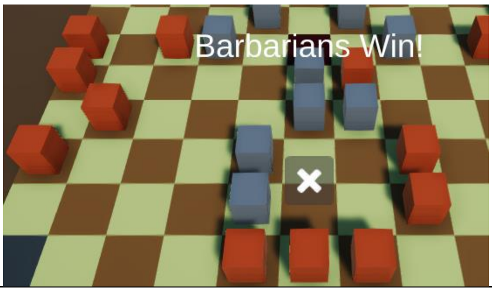
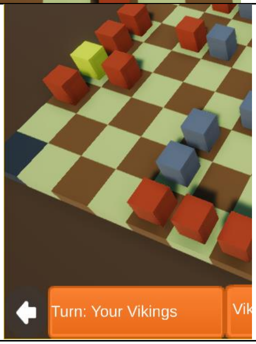
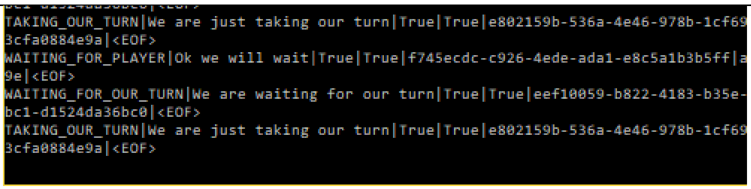
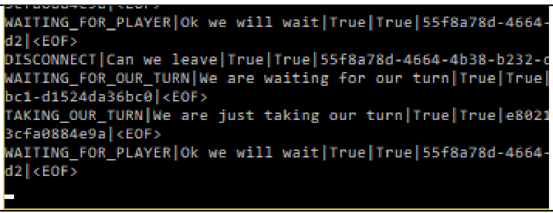


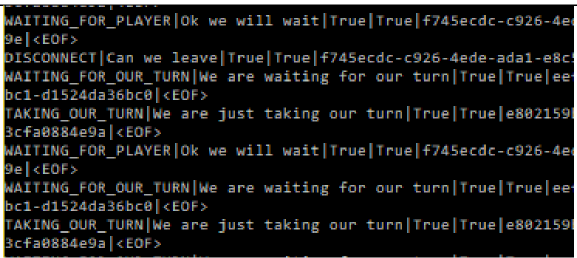
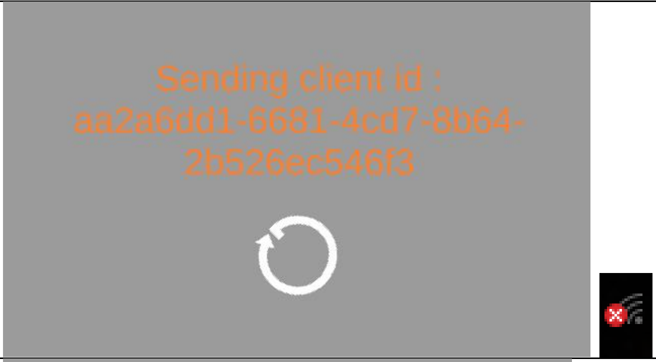
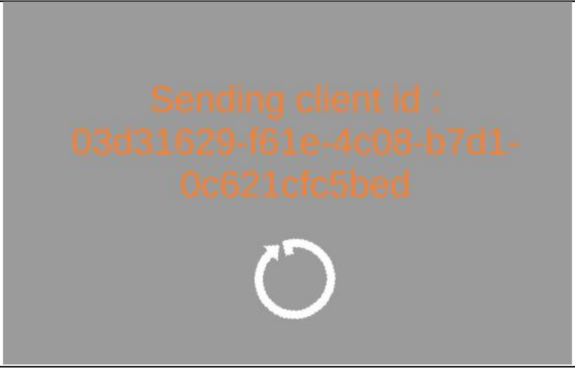
Hnefatafl Major Project Client
Turn: Your Barbarians
Vikings Taken: 0
Barbarians Taken: 0


WAITING_FOR_OUR_TURN|We are waiting for our turn|True|True|1a382bdf-752
NEXT_TURN|4/4@5/4@6/4@3/5@4/5@6/5@7/5@4/6@5/6@6/6@5/7@8/3:3/0@4/0@5/0@6,
WAITING_FOR_OUR_TURN|We are waiting for our turn|True|True|1a382bdf-752-</td><td>Pass</td></tr>
<tr><td>9</td><td>Piece is successfully taken on current turn client and is then replicated on the other client as the turn is passed, the server shows the message for the turn being handed over.</td><td>
TAKING_OUR_TURN|We are just taking our turn|True|True|aed7
WAITING_FOR_OUR_TURN|We are waiting for our turn|True|True|610cc6ff-be8a-447b-
NEXT_TURN|4/4@5/4@6/4@4/5@6/5@4/6@5/6@6/6@5/7@7/1@3/1:3/0@4/0@5/0@6/0@7/0@5/1@
TAKING_OUR_TURN|We are just taking our turn|True|True|610cc6ff-be8a-447b-9f85-


Hnefatafl Major Project Client
Turn: Enemy Vikings
Vikings Taken: 1
Barbarians Taken: 0

</td><td>Pass</td></tr>
</table>

10	The winning client has a message telling them they've win and the losing team has a message telling them that the opponent won.	 	Pass
11	The Barbarian client displays a win message but the Viking client just acts like it is now their turn and no message appears.	 	Fail
12	The client is returned to the main menu and the opponent is left still in the game looking at their message.		Pass
13	The client is returned to the main menu and the opponent is still in the game view, due to a fail in test 11 this is also a fail.		Fail
14	The client is just added to the queue and is waiting while the other two clients are still in the game, the server shows a waiting player messages while also showing the taking turn messages.	 TAKING_OUR_TURN|We are just taking our turn|True|True|e802159b-536a-4e46-978b-1cf69 3cfa0884e9a|<EOF> WAITING_FOR_PLAYER|Ok we will wait|True|True|f745ecdc-c926-4ede-ada1-e8c5a1b3b5ff|a 9e|<EOF> WAITING_FOR_OUR_TURN|We are waiting for our turn|True|True|eef10059-b822-4183-b35e- bc1-d1524da36bc0|<EOF> TAKING_OUR_TURN|We are just taking our turn|True|True|e802159b-536a-4e46-978b-1cf69 3cfa0884e9a|<EOF>	Pass
15	Server closes but the game client remains in the waiting screen with the waiting logo still animating.		Fail
16	Server closes but the game clients remain in the game.		Fail
17	Client returns to the main menu and the server displays a message relating to the disconnect.	 WAITING_FOR_PLAYER|Ok we will wait|True|True|55f8a78d-4664- d2|<EOF> DISCONNECT|Can we leave|True|True|55f8a78d-4664-4b38-b232-c WAITING_FOR_OUR_TURN|We are waiting for our turn|True|True| bc1-d1524da36bc0|<EOF> TAKING_OUR_TURN|We are just taking our turn|True|True|e8021 3cfa0884e9a|<EOF> WAITING_FOR_PLAYER|Ok we will wait|True|True|55f8a78d-4664- d2|<EOF>	Pass

18	The client exits and the server displays messages related to the disconnect.	 WAITING_FOR_PLAYER|Ok we will wait|True|True|f745ecdc-c926-4e 9e|<EOF> DISCONNECT|Can we leave|True|True|f745ecdc-c926-4ede-ada1-e8c WAITING_FOR_OUR_TURN|We are waiting for our turn|True|True|ee bc1-d1524da36bc0|<EOF> TAKING_OUR_TURN|We are just taking our turn|True|True|e802159 3cfa0884e9a|<EOF> WAITING_FOR_PLAYER|Ok we will wait|True|True|f745ecdc-c926-4e 9e|<EOF> WAITING_FOR_OUR_TURN|We are waiting for our turn|True|True|ee bc1-d1524da36bc0|<EOF> TAKING_OUR_TURN|We are just taking our turn|True|True|e802159 3cfa0884e9a|<EOF>	Pass
19	Client is taken to the waiting screen and is shown a message related to initial connection to the server.	 Sending client id : aa2a6dd1-6681-4cd7-8b64- 2b526ec546f3	Fail
20	The client was taken to the waiting screen and is shown a message related to initial connection with the server.	 Sending client id : 03d31629-f61e-4c08-b7d1- 0c621cfc5bed	Fail

Analysis

There are several errors in the interaction between clients and server, Test 11 and 13 could be due to programming oversight and a feature has just been forgotten to be added as Test 10 which tests a similar case but for the other team passed. There seems to be a big problem with the server closing and the clients not knowing how to handle that, as well as attempting to connect to it when the server isn't online or the client has no network connection.

4.4. User Testing

To test the user experience of the game, the Project Android application [23] was promoted online and a user feedback form was offered to be filled out, the form can be found at Appendix C. Using Google Forms [24] an anonymous feedback form was created and was distributed via a URL link, the form also provides graphs that shows the responses visually as pie charts. Unfortunately, there were only five responses which was disappointing, the form was advertised on Reddit and due to high traffic must have been pushed down the webpage. It was posted there to ensure the form was completely anonymous, posting on Facebook would possibly have led to a higher yield but could have been biased.

Even though there was a lack of quantity with the answers, there is still some conclusions that can be extracted from some of the answers. Figure 27 shows how many people didn't understand how to play the game, this result is far from ideal. The nature with this game is the rules are most likely completely foreign to any new users, so reading the help page is important for them understanding the game. What this result shows is either 3 people didn't go to the help page or they didn't understand the rules if they did, the other two either understood the provided rules or may researched the rules for themselves. For both cases there could have been some further questioning to narrow down the reason for their responses.

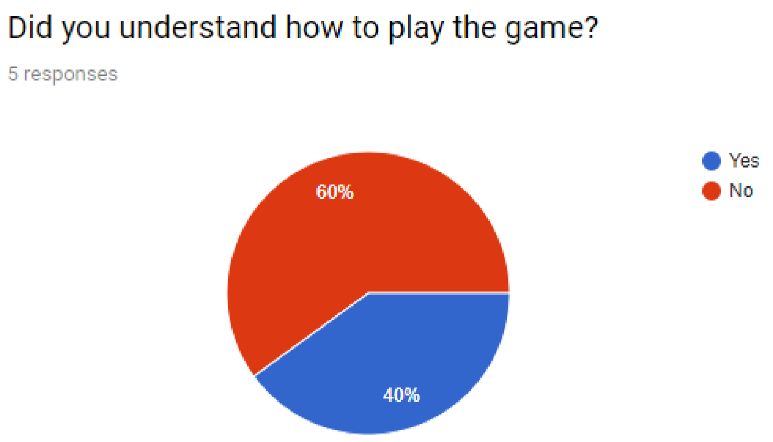
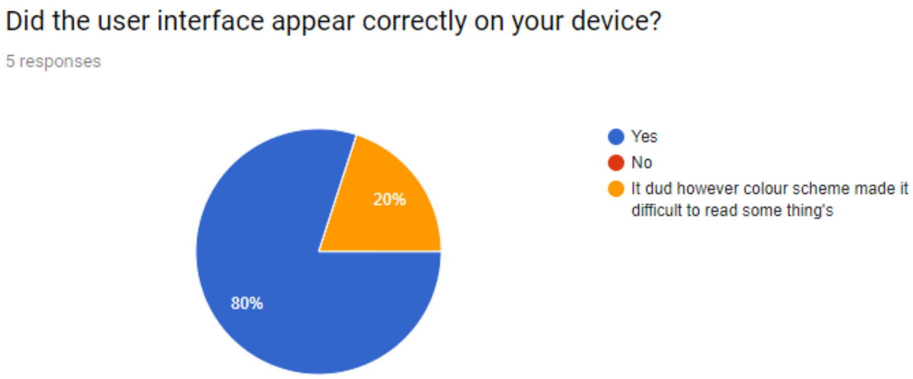


Figure 18

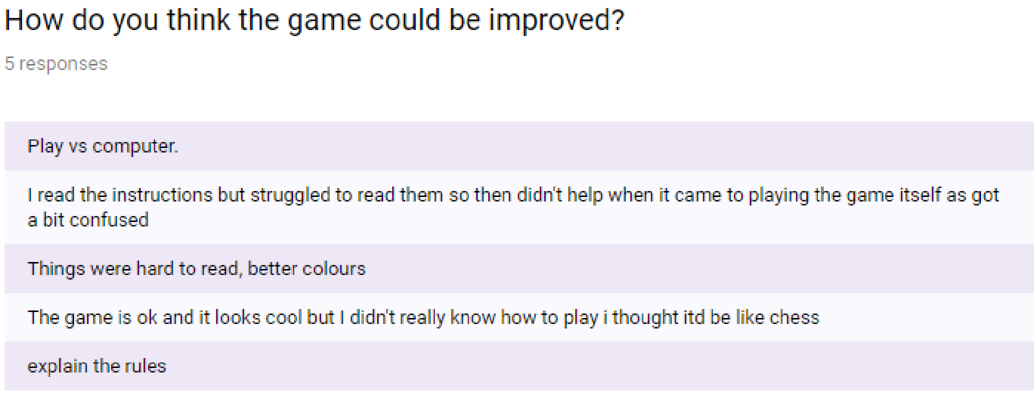


Did the user interface appear correctly on your device?

5 responses

Figure 19

While most of the responses to the question in Figure 28 are good and mean the work put into ensuring user interface scaling was not wasted, there was one response that can be learnt from. Although it is not what the question was intending to ask for, it points out a weakness in the readability of the user interface. This point is reiterated by two responses in relation to the question in Figure 29, 'I read the instructions but struggle to read them' and 'Things were hard to read, better colours'. If a mid-development survey had been sent out this problem could have been found earlier and fixed by consulting with possible users to found out better colour schemes.



How do you think the game could be improved?

5 responses

Play vs computer.

I read the instructions but struggled to read them so then didn't help when it came to playing the game itself as got a bit confused

Things were hard to read, better colours

The game is ok and it looks cool but I didn't really know how to play i thought itd be like chess

explain the rules

Figure 20

Overall the two weaknesses discovered through this series of testing was that users need to be taught the rules in a better way, possibly through a series of tutorial levels and teach the rules step by step, or clearer instructions that are mandatory to read on the first start-up of the game. The user interface in general also needs to be changed to be more appealing to the users and more readable, this could have contributed to the lack of understanding of the rules.

5. Critical Evaluation

5.1. Research

When doing the research there was no consideration towards how the server application would be hosted, this had to be solved before wider user testing could occur. During development this could be tested on the local machine, but that wasn't ideal for opening to the internet. Even during the testing while users where using the server, the solution chosen was still not ideal and kept crashing constantly. This was not an error in the server application, as it could run on a local machine indefinitely, but the Amazon EC2 Instance seemed to not keep connections open for extended periods of time.

5.2. Design Decisions

When comparing to the feature list outlined in the Design, there are several features that didn't get completed. Fortunately, these were all low priority tasks but they would have added an increased level of polish to the game. The requirements list could have been more detailed, and broken up some of the larger requirements into smaller goals. Tasks such as Offline Multiplayer is a large topic and is made up of smaller sections, from one point of view it should be broken up to allow a more detailed design structure but on the other hand they are useless unless all of them are implemented for the overall goal.

Using a network message class allowed for very easy transferring between server and client, the serialise and deserialise methods once implemented early on had to be changed quite a lot. The initial structure of the class changed several times through development and a few new variables were added to it the allow different interactions between the server and client.

After doing research into similar systems it was decided to make a 3D view game, while this seemed like an appropriate choice and provided more technical challenges it possibly makes the game trickier to play. While it provides a more visually stimulating game, if the objective is to play the game the easiest way possible then the 3D view is not the right solution. Features such as rotating and zooming in would be unnecessary and would allow more time towards the development of the server had a 2D view be adopted, possibly allowing time for an authoritative server to be implemented.

In the user testing there were comments relating to the colour scheme used and that it could be improved, during research and design there should have been some study into what colour schemes are appealing and easily readable. While the Project is a good technical piece of work it misses out on a good user experience, this is due to the lack of user engagement during the development process.

5.3. Testing

The testing undertaken was ok but there was more that could have been done to ensure the reliability of more of the Project, as well as more tests earlier on to drive the development of the Project.

While the black box testing provided satisfactory results and only revealed a few errors in the program, the main flaw is with black box testing itself. There will always be more test scenarios and there is a high chance that the ones that have been missed, are ones that would fail because they are edge cases. When a Project has only a single developer who is designing the code, and then subsequently designing the testing for the program there will be much missed in the same way that proof reading your own written work can lead to blindly missing errors.

An outside person could have also compiled a list of test scenarios, or multiple people varying from no programming experience to experienced to see how the scenarios differ. This could then be combined into a more extensive list of scenarios which would cover less likely scenarios.

The size of the user acceptance testing results was not ideal and more should have been acquired to ensure a better outlook on the general opinion, where friends and family would have most likely given more opinions there is a higher chance that they would have been positively biased. The chosen route of posting anonymously on Reddit, allowed the results to reflect the true thoughts of absolute strangers but unfortunately did not return a satisfactory yield.

Mid-Project user acceptance testing would have allowed a wider pool of opinions to help the development, aspects such as the 3D or 2D view could have been addressed earlier on in development when those aspects were not finished. It could have also helped drive the development in a direction that would lead to a better game to play and enjoy.

Stress testing on the multiplayer server could have been done by creating a test application, this program could send many connections to the server. It would then simulate the game messages between the test clients and the server, this would see how the server can handle many connections and how much it slows down when processing multiple matchmaking pairs simultaneously.

5.4. Tools

The success of the tools chosen for the Project vary, using the Unity engine was the right choice allowing for more time developing the game instead of making a custom engine for the game. It provided a good set of tools for developing 3D models and lighting, the user interface was easy to implement using a drag and drop system although tricky to get the scaling to work for multiple resolutions.

Using C# for the server may not have been the best decision, unfortunately little thought was given to a practical solution for testing the server at the beginning of the Project, and Java may have been an easier option due to its compatibility with lots of platforms. A solution was worked out using the original C# Project but it did have to be converted into a slightly different application as imagined getting it to work on the Amazon AWS Linux server.

5.5. Future Work

If the Project were to continue development there are several features that could be added or improved upon.

5.5.1. Authoritative Server

Currently the server has no checking to make sure the board states being transferred between the clients and that the clients are making valid moves. Now it relies on the checking that has been built into the client, but if anyone modified the game client somehow or intercepted the packets being sent to the server then this would not be detected by the server.

This could be implemented by holding the board state of each game on the server, whenever the turn is passed check that only a single piece has been moved and that where it moved to is a valid location. This would however put a lot more processing on the server between turns and on the current hosting setup would probably cause performance issues.

5.5.2. Online Lobby and Friends

Currently the only way of getting into an online multiplayer game is by being randomly being matched with other people, if the game grew there would most likely be a demand for users to play against their friends. By creating a lobby system where players can create matches and allow their friend to select to play them, would solve this. This would require a solution seen in many games in which a password would be needed to lock the game out to strangers.

5.5.3. Google Play integration

This feature would only be relevant to the Android version and would drive development towards just supporting Android, this would most likely be most of the devices running the game.

Using the Google Play Game Services (GPGS) [16] the game would be able to make use of integrated leader boards and users could sign in using a Google account. Building upon the online lobby idea put forward in 5.5.2. the GPGS could also enable sending game requests to friends on your google account and sharing leader board information with them. This would all add a level of risk, because currently the server does not share any information between clients and this solution would require opening them up to their Google profiles.

5.5.4. 2D View

After research it was decided that a 3D view would be used, but the option for a 2D view could be developed to allow the user to select how they want to play. This could allow for lower end devices to be able to download and play the game.

5.5.5. AI opponent

When researching other similar applications, a main feature that many of them shared was a mode that allowed you to play against a computer opponent, this would be a good addition to the Project and allows a user to play the game even when they have no one with them or the server is empty.

This feature would require extensive development and would possibly be an entirely separate Project due to the complexity. Many of the other applications had several levels of difficulty, machine learning could allow the AI to become very proficient at playing the game much like the Google DeepMind AI [17] plays GO, it is open source and would possibly serve as a basis to create an AI for the Project.

5.6. Conclusion

The Project has achieved its goal, of transforming a medieval board game in to a playable video game. The two key features (offline multiplayer and online multiplayer) of the Project were successfully implemented and provide an acceptable level of play. The Project's weakness is user experience, when introducing new users to the game the hardest part is teaching the rules. In a physical conversation the rules can be taught in a matter of minutes but when a user downloads the Project they must learn from everything provided within the game, while the game does provide the rules to users a lot of the time it didn't help.

During user feedback it was discovered that the colour scheme chosen was not the easiest to read, the colour scheme was chosen from a website that provides colour schemes [22]. This scheme should have been tested earlier on in development to check that it was appealing to users, other different schemes could have been shown at the same time to find the best. The game client could have been improved by getting early hands on user feedback to drive the development.

Overall the Project is a success and provides the main requirements it set out to achieve, but some aspects could be improved namely the user experience when playing the game.

6. Appendices

A. Third-Party Code and Libraries

MK Glow Unity Asset *Free [1] – This asset provides some fancy glowing materials which I am using for selected pieces in the game.

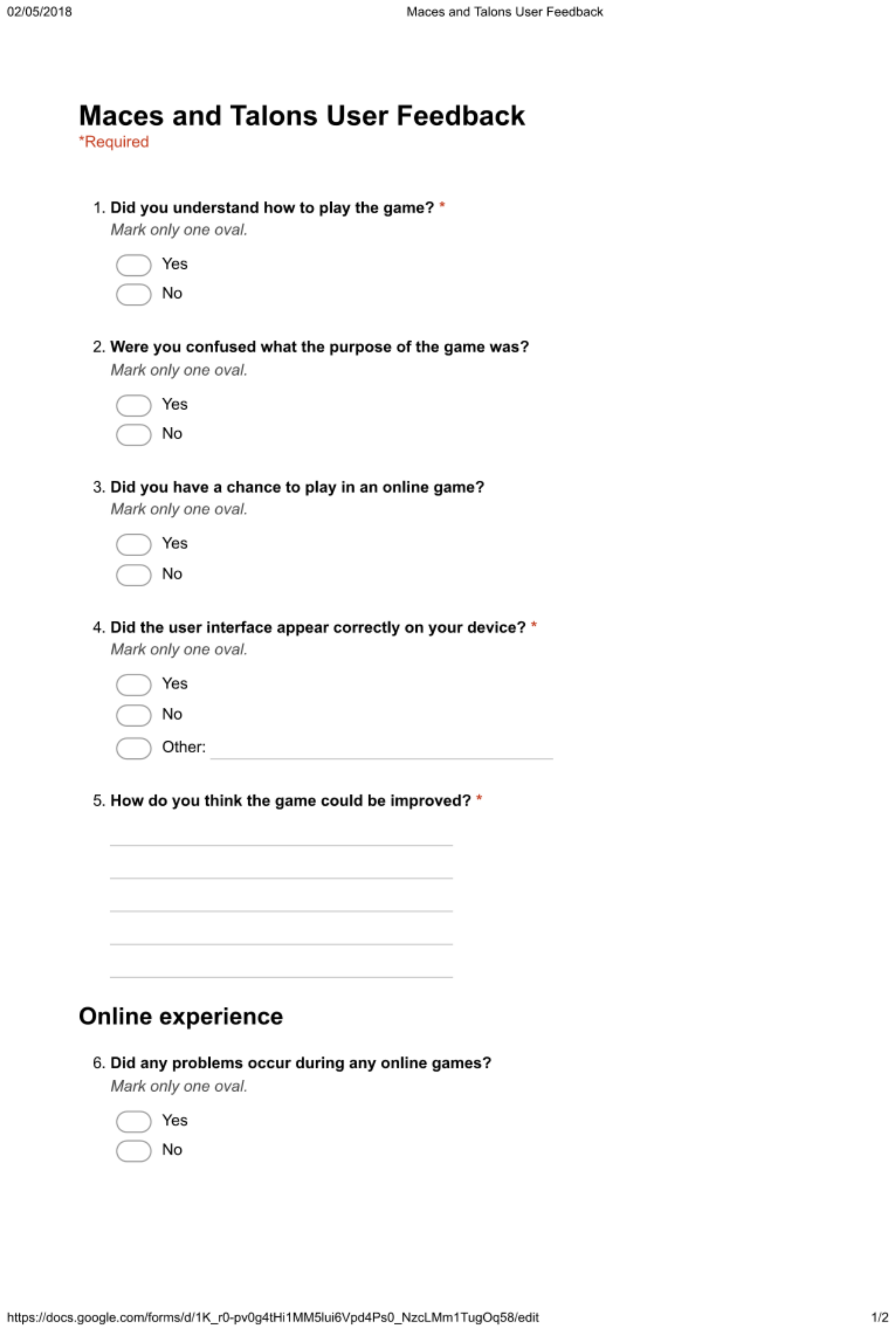
Kenney Game Assets [19] – A collection of art assets, they are published under the Creative Commons Zero v1.0 Universal License. All the buttons in the game use are from this pack.

Microsoft Asynchronous Socket Server Example [13] – This code provided the main starting point for creating the connection between the server and client, while the code was used mainly in its entirety to start out with it has since been stripped down significantly. In the source code for the Project it has been indicated where sections of this example still exist and which part of original to the Project.

B. Glossary

Serialise – The process of turning a class into a series of bytes.
Deserialise – The process of turning a series of bytes into a specified instance of a class.

C. User Feedback Form



02/05/2018 Maces and Talons User Feedback

Maces and Talons User Feedback

*Required

1. **Did you understand how to play the game? ***
   Mark only one oval.
   - ○ Yes
   - ○ No

2. **Were you confused what the purpose of the game was?**
   Mark only one oval.
   - ○ Yes
   - ○ No

3. **Did you have a chance to play in an online game?**
   Mark only one oval.
   - ○ Yes
   - ○ No

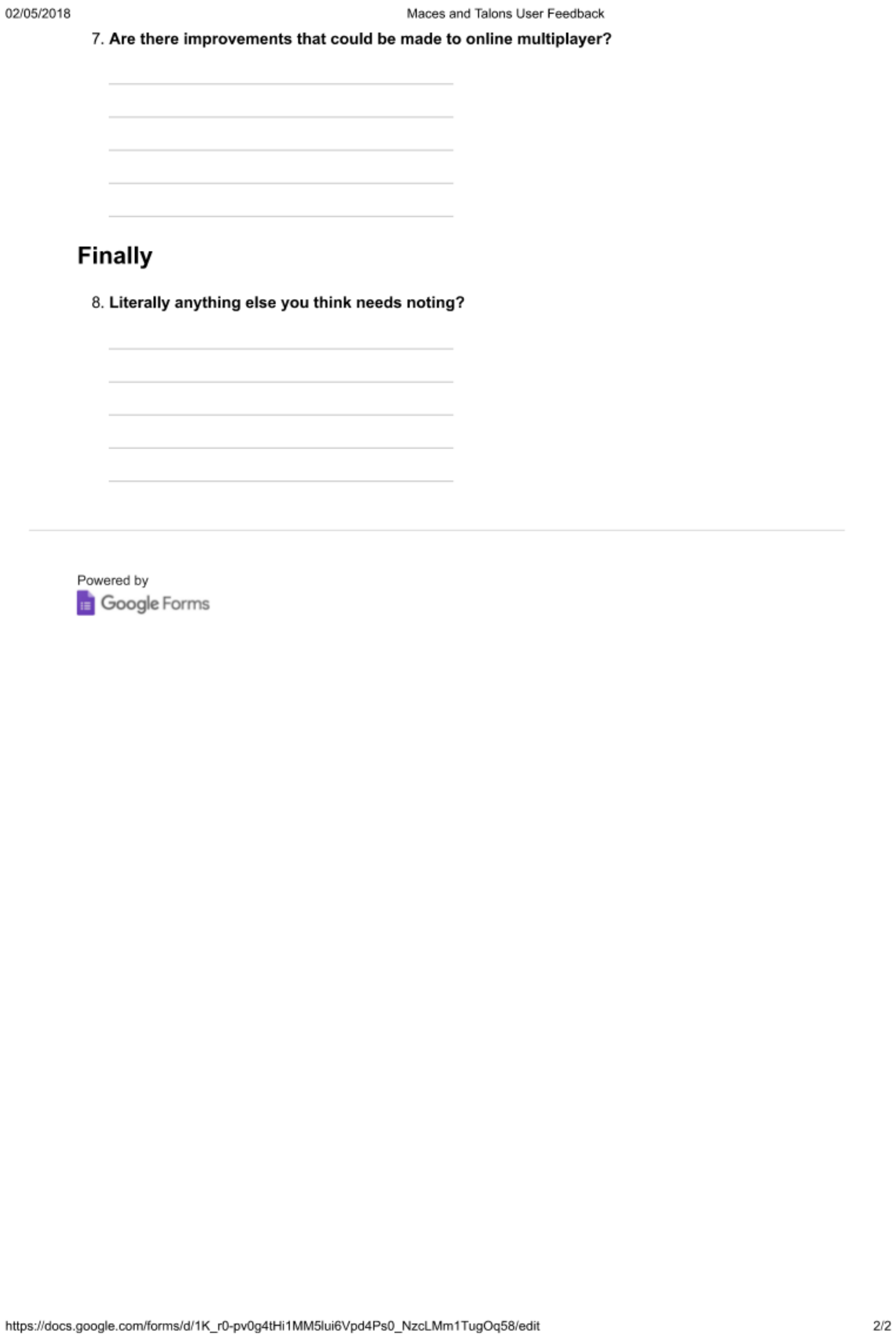
4. **Did the user interface appear correctly on your device? ***
   Mark only one oval.
   - ○ Yes
   - ○ No
   - ○ Other: ______

5. **How do you think the game could be improved? ***

Online experience

6. **Did any problems occur during any online games?**
   Mark only one oval.
   - ○ Yes
   - ○ No

https://docs.google.com/forms/d/1K_r0-pv0g4tHi1MM5lui6Vpd4Ps0_NzcLMm1TugOq58/edit 1/2



02/05/2018 Maces and Talons User Feedback

7. **Are there improvements that could be made to online multiplayer?**

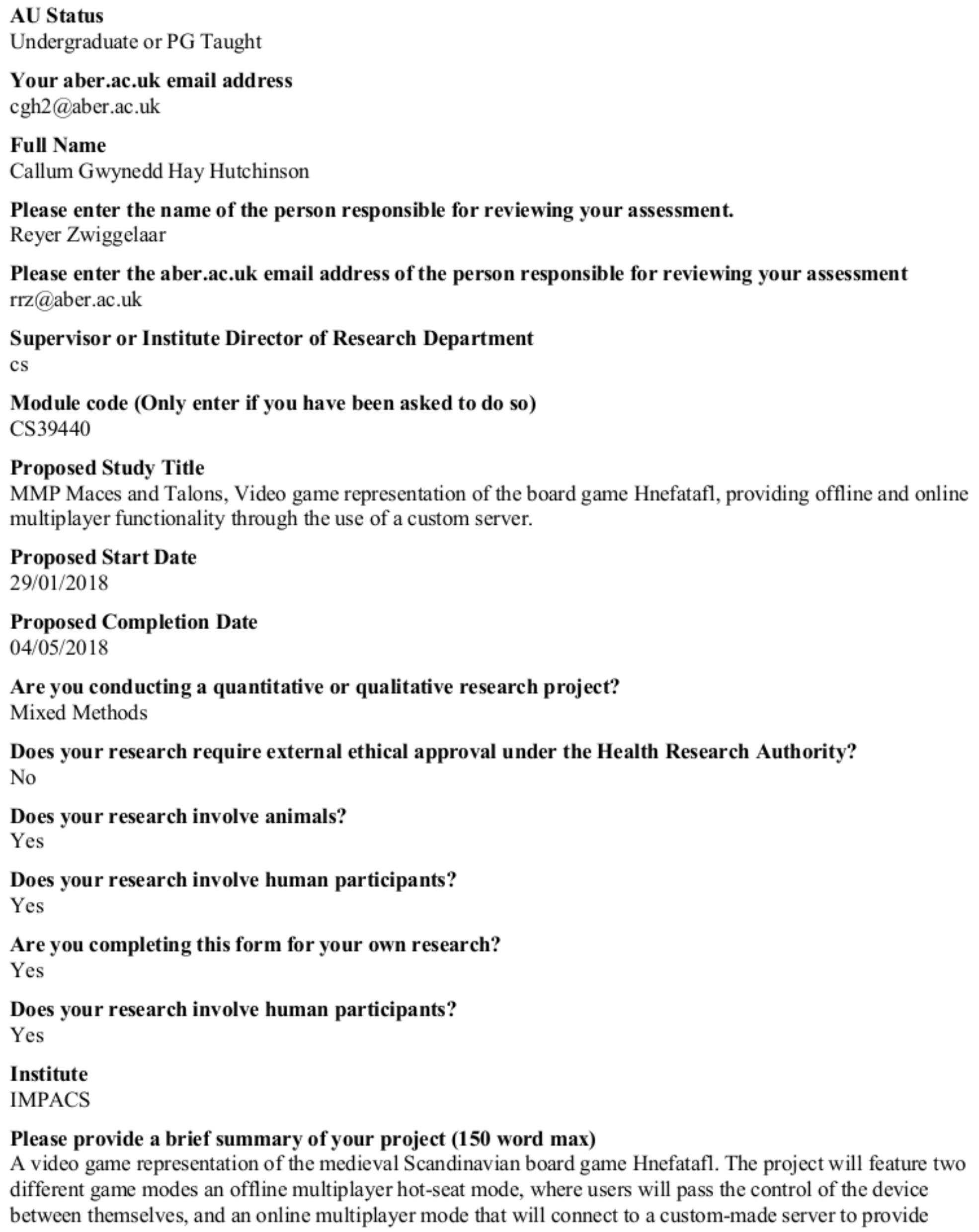
Finally

8. **Literally anything else you think needs noting?**

Powered by Google Forms

https://docs.google.com/forms/d/1K_r0-pv0g4tHi1MM5lui6Vpd4Ps0_NzcLMm1TugOq58/edit 2/2

D. Ethics Submission



AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
cgh2@aber.ac.uk

Full Name
Callum Gwynedd Hay Hutchinson

Please enter the name of the person responsible for reviewing your assessment.
Reyer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your assessment
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
MMP Maces and Talons, Video game representation of the board game Hnefatafl, providing offline and online multiplayer functionality through the use of a custom server.

Proposed Start Date
29/01/2018

Proposed Completion Date
04/05/2018

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
Yes

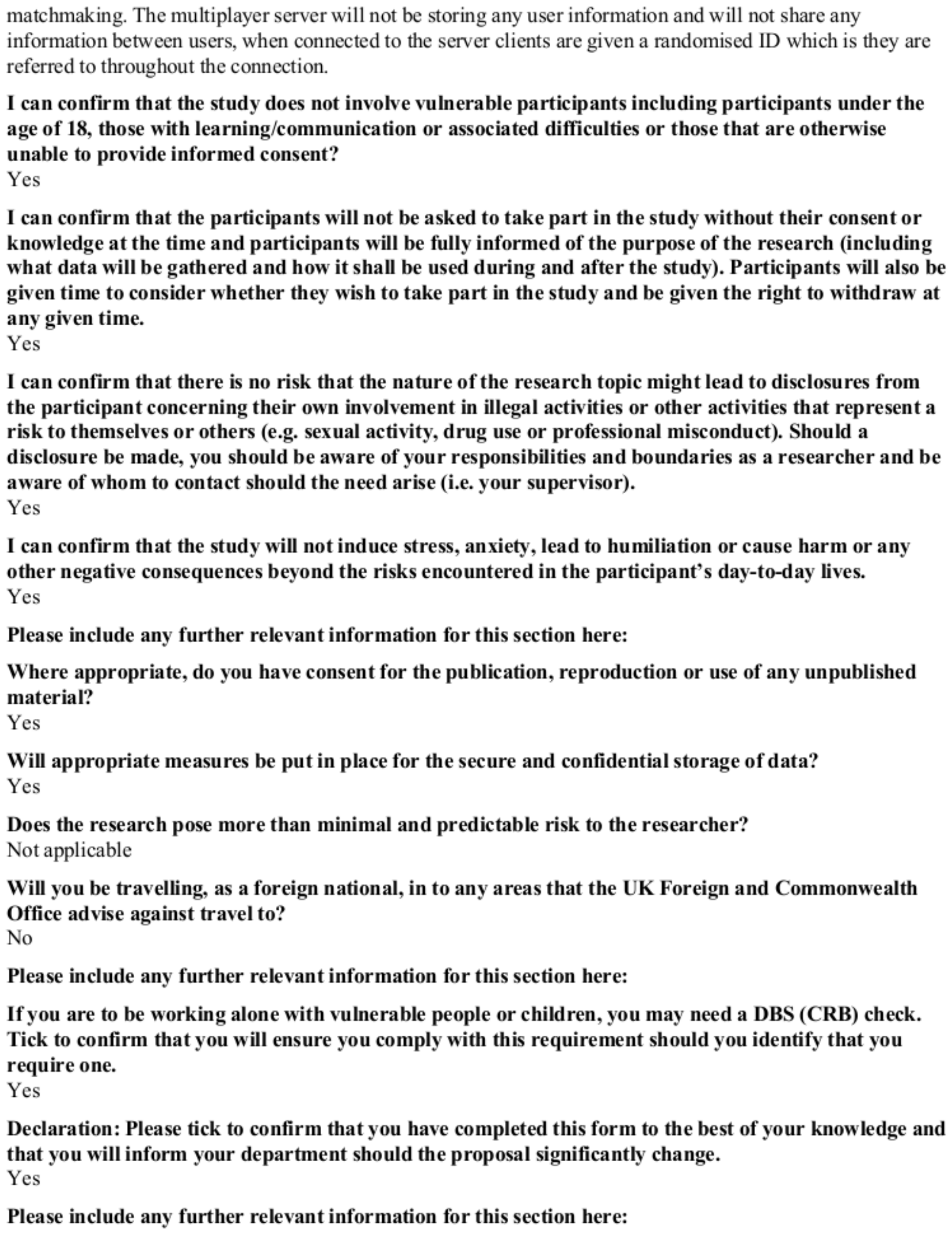
Does your research involve human participants?
Yes

Are you completing this form for your own research?
Yes

Does your research involve human participants?
Yes

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
A video game representation of the medieval Scandinavian board game Hnefatafl. The project will feature two different game modes an offline multiplayer hot-seat mode, where users will pass the control of the device between themselves, and an online multiplayer mode that will connect to a custom-made server to provide



matchmaking. The multiplayer server will not be storing any user information and will not share any information between users, when connected to the server clients are given a randomised ID which is they are referred to throughout the connection.

I can confirm that the study does not involve vulnerable participants including participants under the age of 18, those with learning/communication or associated difficulties or those that are otherwise unable to provide informed consent?
Yes

I can confirm that the participants will not be asked to take part in the study without their consent or knowledge at the time and participants will be fully informed of the purpose of the research (including what data will be gathered and how it shall be used during and after the study). Participants will also be given time to consider whether they wish to take part in the study and be given the right to withdraw at any given time.
Yes

I can confirm that there is no risk that the nature of the research topic might lead to disclosures from the participant concerning their own involvement in illegal activities or other activities that represent a risk to themselves or others (e.g. sexual activity, drug use or professional misconduct). Should a disclosure be made, you should be aware of your responsibilities and boundaries as a researcher and be aware of whom to contact should the need arise (i.e. your supervisor).
Yes

I can confirm that the study will not induce stress, anxiety, lead to humiliation or cause harm or any other negative consequences beyond the risks encountered in the participant's day-to-day lives.
Yes

Please include any further relevant information for this section here:

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Yes

Will appropriate measures be put in place for the secure and confidential storage of data?
Yes

Does the research pose more than minimal and predictable risk to the researcher?
Not applicable

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

Please include any further relevant information for this section here:

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

Please include any further relevant information for this section here:

Annotated Bibliography

[1] MK Glow Free – https://assetstore.unity.com/packages/vfx/shaders/fullscreen-camera-effects/mk-glow-free-28044

This Unity asset provides some fancy glowing materials.

[2] Hnefatafl (Philippe Schober) (Google Play Store) – https://play.google.com/store/apps/details?id=com.fellhuhn.hnefatafl

A game that was analysed off the Google Play Store because of it Hnefatafl gameplay.

[3] Tafl (Jocly) (Google Play Store) – https://play.google.com/store/apps/details?id=com.jocly.android.app10_vc_as

A game that was analysed off the Google Play Store because of it Hnefatafl gameplay.

[4] Hnefatafl (MeatballsTeam) (Google Play Store) – https://play.google.com/store/apps/details?id=com.meatballsteam.hnefatafl

A game that was analysed off the Google Play Store because of it Hnefatafl gameplay.

[5] Hnefatafl The Viking Game – http://www.gamecabinet.com/history/Hnef.html

This website has lots of information about Hnefatafl and was the main site used when learning the rules for the Project.

[6] Hnefatafl – https://boardgamegeek.com/boardgame/2932/hnefatafl

Another site used to learn the rules of the game and differences between the other varieties of Tafl

[7] The History of Hnefatafl – http://tafl.cyningstan.com/page/3/the-history-of-hnefatafl

Contains historical information about Hnefatafl and were the game originated.

[8] Android Studio – https://developer.android.com/studio/

An IDE and visual editor for Android applications, a possible candidate for developing the Project.

[9] CLI – Common Language Interface – https://www.c-sharpcorner.com/UploadFile/f64127/C-Sharp-tutorial-part-2-cli-clr/

A specification that C# was built upon that describes the way it can run and the environments it can run in.

[10] TCP vs UDP – https://www.diffen.com/difference/TCP_vs_UDP

A comparison of the differences between TCP and UDP.

[11] UDP vs TCP – Gaffer on games - https://gafferongames.com/post/udp_vs_tcp/

A comparison of the differences between TCP and UDP

[12] Real-Time Data Transfer – https://www.lifesize.com/en/video-conferencing-blog/tcp-vs-udp

Compares real-time applications of TCP and UDP in relation to transferring data.

[13] Microsoft Asynchronous Socket Server Example - https://docs.microsoft.com/en-us/dotnet/framework/network-programming/asynchronous-server-socket-example

The main example of code that was used as a starting point for the server code.

[14] The quick guide to guids – https://betterexplained.com/articles/the-quick-guide-to-guids/

A helpful guide explaining why Guids are useful.

[15] Mono Library – https://www.mono-Project.com/

An open source version of .Net.

[16] Google Play Games Services – https://developers.google.com/games/services/

A library that allows games to integrate and access Google Play Games.

[17] Google Deepmind – https://deepmind.com

An open source artificial intelligence developed by google.

[18] Visual Studio 2017 – https://www.visualstudio.com

A Microsoft development environment that can be used for several different programming languages, including C#.

[19] Kenney Game Assets –
https://kenney.itch.io/kenney-donation

A collection of art resources that can be used in games, under the Creative Commons Zero v1.0 Universal License. Pack was purchased for personal use several years ago.

[20] Unity Canvas Scaler –
https://docs.unity3d.com/Manual/script-CanvasScaler.html

A feature within Unity that allows for scaling user interfaces easily.

[21] Github –
https://github.com/

A git version control website that was used during the development of the Project.

[22] Coolors –
https://coolors.co/

A website that provides randomized colour schemes.

[23] Maces and Talons: Hnefatafl Project Application -
https://play.google.com/store/apps/details?id=uk.co.callumhutchy.macesandtalons

The application for this Project’s game client

[24] Google Forms –
https://www.google.co.uk/forms/about/

A web application that allows the creation and distribution of survey forms, the results are displayed in a graphical layout and can be exported to csv.

[25] Amazon AWS EC2 –
https://aws.amazon.com/ec2/

Scalable server hosting in the Amazon cloud.