Wizard Card Game

CS39440 Major Project Report

Author: Eithen Howard (esh3@aber.ac.uk)
Supervisor: Dr/Prof. Christine Zargesr (chz8@aber.ac.uk)

21th February 2019
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in Computer Science (G400)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, U.K.

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 Registry (AR) 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 Eithen Dennis Steven Howard

Date ....

Consent to share this work

By including my name below, I hereby agree to this project's report and technical work being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name Eithen Dennis Steven Howard

Date ....

Acknowledgements

I am grateful to...

I'd like to thank...

Abstract

For this project I will be creating a Wizard Card Game which will be used to have an environment for a Monte Carlo Tree Search Artificial Intelligence. Wizard is a Card Game which a normal 52 card deck and 4 wizard and jester cards. Wizards are the highest and jesters are the lowest cards in the game. In the game bid for a round and try to win as many tricks in that round as possible, if you′re closer to your original bid the more point you will score. A trick can be won by having the highest card that match the trump suit (the card played after the hands are handed out) or the dealer′s suit. Another way to win is by play a wizard card which trumps all other cards. Once the rounds are over the person with the highest score wins.

The Monte Carlo Tree Search is the AI technique I will be using which involves creating a tree from the state that the game is currently in. This is then randomly expanded by selecting a card to play. It will then select a state from the ones that have been expanded. This will then simulate a game and will provide a score for how close it was to the original bid and give the score to the original child state. This is then repeated for a specified amount of time, then the one that has the highest ratio of score to how many times it is visited, this will then give the best card to be played.

Contents

List of Figures

List of Tables

Chapter 1

Background and Analysis

1.1 Background

In this section I will be talking about the Wizard Card Game, how the game works and how you can win the game and show in this video [1]. I will also discuss the Artificial Intelligence techniques that were consider using for the opponents, which are neural networks, minmax and Monte Carlo algorithms. This will be expanded upon to how I concluded using the Monte Carlo Tree search for this game.

1.1.1 Wizard Card Game

1.1.1.1 Dealing and Setup

Wizard is a card game that contains 52 normal deck cards and 8 extra cards 4 of which are called wizards and the other 4 are called jesters [2]. Each player is dealt the same number of cards that is the round. For example, round 1 means each person get 1 card but round 10 means they each get ten cards. This is then played until the is no cards left to be dealt out mean for 3 people there should be 20 round and 15 rounds for 4 people. A trump card is also dealt from the deck at the beginning of the round from the top. After each play in a round the dealer is changed clockwise.

1.1.1.2 Playing and Bidding

One the trump has been dealt, the person to the left of the dealer states how many tricks they think they will win in this round. The maximum they can bid it the number of cards they have in their hands. Once everyone has put their individual bids in, the player to the left of the dealer puts down the first card which can be any card they want, then each player can either match the suit that the first player put down or the better play it to match the trump suit or play a wizard to win. Otherwise they can play off suit or play a jester to lose.

1.1.1.3 Scoring

To score points in this game you must use the bid on how many tricks you think you can win, giving you 20 point for being right and 10 point for every trick you win. Otherwise you get negative 10 point per every trick you were over or under for that round. For example, if you guess 4 tricks to win and only one 1 you would lose 30 points.

1.1.2 Monte Carlo Algorithm

The Monte Carlo Algorithm is probabilistic search algorithm that creates a tree based on the current states of the problem. This is then randomly expanded from the root node and simulated what might happen, under specific constrains such a iterations of the simulation or under an amount time. This will then give a score back on what how well the simulation wen. once this is done the best option will be picked based on what has the highest score from the tree that has currently been created. The next sections below will show the 4 phases that are used in the algorithm. [h]

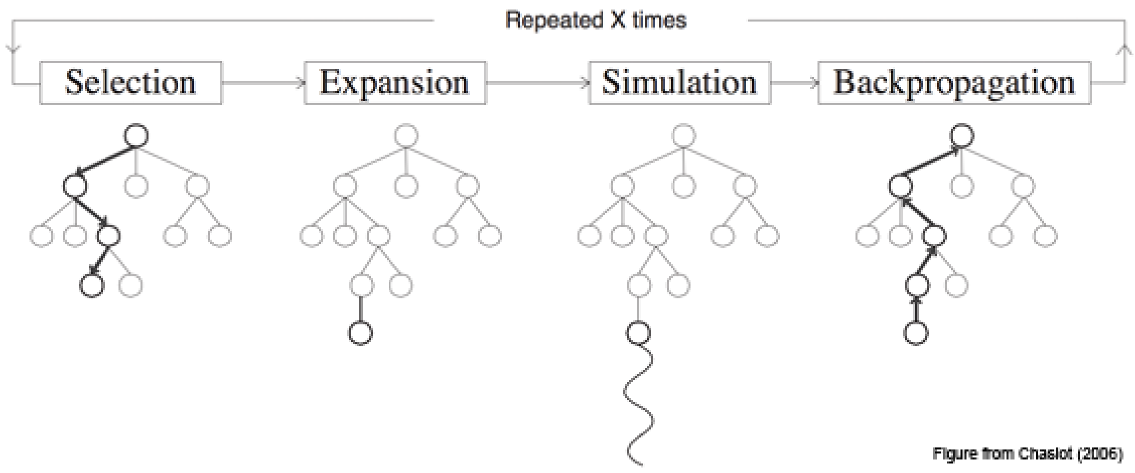

Figure 1.1: Diagram Showing Steps in Monte Carlo Tree Search

1.1.2.1 Selection

For this phase in the algorithms it will start with the root node that shows the initial state of the game. This will then select the child node that has the best win rating. this win rating will be selected by the use if Uct(Upper Confidence Bound for Trees). The formula used for this is:

- w_i : This variable will contain the number of wins the node will have after *i* amount of moves.
- n_i : This is the number of simulation after the *i* amount of moves
- c : This is this exploration parameter and should theoretically be equal to .

- t : the total number of simulations performs for the parent node. [3]

1.1.2.2 Expansion

After the selection of a node is complete we do the expansion of this node. This is done by adding a new child node to the tree that was optimally reached during the selection process [4].

1.1.2.3 Simulation

Once the node has been expanded ,we perform a simulation of the state until it reach the end of what we are trying to achieve. For example in the project we would simulate the selection of the cards until there are non left and how large there score is.

1.1.2.4 Backpropagation

In this part of the process we look at the result of the simulation and give a reward for how good the simulation was. In the context of this game the close the score is to the initial bid the better it is, meaning that we shall reward the child node with a higher score the larger the score and closer to the bid.

Once this is done under it will be repeat for a set number of iterations or under a specific time period. This will then create an asymmetric search tree that grows after each iteration through the steps. Once it is done iterating, it will pick the best child node best on the best score divided by the number of visits that specific node has.

1.1.2.5 Advantages

- Compared to other AI algorithms this is quite a simple algorithm to implement.
- As this is a heuristic algorithm is does not need to everything that happens in the game apart from the rules of the game, how it ends the simulation/leaf node and what cards the player may have. this is because it can find it own moves and randomly playout the game.
- The tree that has been creates can be used for the future meaning less computational expense creating more nodes.
- As you can select the number of iterations/time given, it means you can make the tree as small or large as you would like.

1.1.2.6 Disadvantages

- The larger the tree growths the more rapidly memory expensive.

- The algorithm normally needs a large amount of iteration to be able to effectively pick the best path, meaning that it will have to take amount of time to construct a large enough search space.

1.1.3 Similar Algorithms

In this section, I will be talking about the two other Algorithms that could have been used for the Artificial intelligence of the game. Discussion on why they was not used and why the Monte Carlo Tree Search was the one to be picked will also be talked about.

1.1.3.1 Neural Network

Neural Network are artificially created based on the neuronal structure of the mammalian cerebral cortex but on much smaller scales. [5]

1.1.3.2 MinMax

1.2 Analysis

In this section I will be talking about what I will be aiming doing achieve with this product and what technology I will be using during this process.

1.2.1 Project Aims

The aim for this project is create a functional software version of the card game Wizard. This will also contain an Artificial intelligence Opponent that uses the Monte Carlo Algorithm to select a card for that round. The game will be simplified to only have 3 players, two of which will be the AI and one which is the human player. Each player will have 15 cards each and play until there are no cards left for 5 rounds. [h] Below shows the descriptions of the different use cases:

- Enter bid - A bid must be selected based on what cards are in the players hands for what score they think they will get.
- Select Play Card - A play card must be selected based on what the trump card is, what the other players have player and what cards the player has to use.
- See Trump Card - Printing of the Trump card so that a selection of the play card can be made and see if it is valid card to play.
- See Cards in hand - Print the cards that are currently in the hand of the player so a card can be selected from that hand.

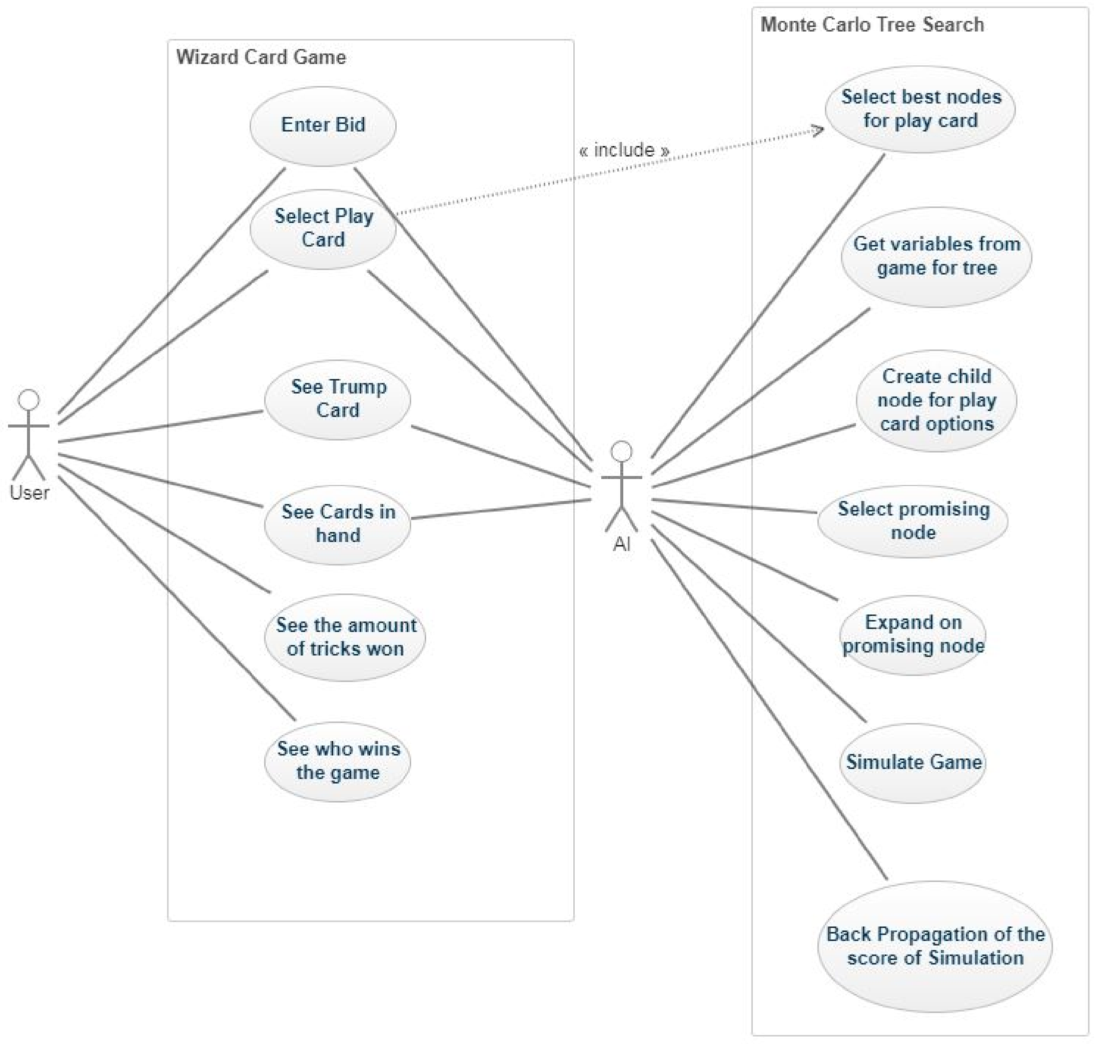

Figure 1.2: Diagram Showing the Use Case for the Project

- See the amount of trick won - Show the number of tricks that they have won at the end of the round so they can see how close it was to their bid.
- See who wins the game - To show who won overall in the game
- Select best nodes for play card - Select the node with the largest score to visit ratio and play the card in the game for that player.
- Get Variables from game to tree - Gets all the variables needed to create the tree such as the cards in the players hand and what the trump card is.
- Create child node for play card options - Create child nodes from the current game state that contain the possible moves played by the players.
- Select promising node - Selects a node to explore to see simulate and see how good of an option it is.

- Expand on promising node - Expands on the node it is currently exploring the make the tree and only stops after it cannot expand any more.
- Simulate Game - Simulate the game to show the what may happen in the game and see if it is a good option. item Back Propagation of the score of Simulation - Gives a score to the child of the root node based on how good the simulation was and how many times that node was visited.

1.2.2 Technology Overview

This section will contain information on the technologies that are used in the project and how the help to achieve the goals that have been set.

1.2.2.1 Programming Language and Software

The programming language used for the project is was java [6] and the IDE for to implement it was IntelliJ IDEA 2018, this was because of the experience in using this language made it a better option than the others. As Java is an object oriented programming language, it makes it easier for changes to the program in the future for different type of AI that could be used or implemented for the Wizard card game [7]. As this was more focus on the AI aspect of the game the fact that java atomically allocates and has garage collection made it easier to focus on making the AI. As for the IDE, it was chosen for the easy of use for java and in the way in which the debugging works will make it easier to find problems with the program.

Another project that will be use in the project will be TeXworks [8] using the Latex to produce a pdf, which is a word processors for the development of the project report. This was chosen as it may be more difficult to initially write with over another processor such as Microsoft Word but as there is some experience in the use of it, using it can produce a better document and can be quicker to produce with such functions as the automated content page being produces as you develop [9]. using latex makes it easier to separate out different sections of works without having the completely changing everything.

1.2.2.2 Hardware

As this project doesnâĂŹt require a large amount of data space or processing speed to function, simple hardware would suffice in it usage. However a large amount of memory would be useful for large amount of iterations in the tree search. The hardware that will be used the build the software is:

- Processor: Intel(R) Core(TM) i7-7700HQ @ 2.80GHz
- Installed Memory(RAM): 16.00 GB
- System Type: 64-bit Operating System

- : Operating System: Windows 10 Home

With the hardware that is being used, there is ample memory to give the tree search a large number for testing and then reduce it for a more standard amount of memory.

1.3 Research Method and Software Process

1.3.1 Foreseen Challenges

One of the challenges that will be faced during the project will be making sure that the all of the rules of the game are applied and that all of the player within the game will follow it. Another problem that will arise will be trying to implement the Monte Carlo Tree Search as a lot if the reference are for two player games meaning it will have to be manipulated to work for the game. Deciding on how to score what it the best simulation will also be difficult as it will depend on how close they are to the bid or how many tricks the players win. applying Rules, Implementing Mont Carlo Tree search and optimising.

1.3.2 Process Methodology

There was a lot of options for developing my project. In the end I decide to you a agile approach based on Scrum. This would give me an iterative way of progressing in the project, use the sprints in the methodology to progressively improve on the previous work that I have done, adding more functionality and complexity to it as it is being developed. This led to the conclusion that it was a better option then traditional methods such as the waterfall model. This is because the work can start on the software whilst writing up the report without having to do redo all the work that has previously done when changes are made to the design of the software.

1.3.2.1 Version Control

The version control during the development of this project will control use Git. This will be using a private GitLab repository of which my project will be stored in. The repository will allow me to back up my code regular and manage it. This would mean it could get back to a previous implementation of my code if there later version has broken. This will also be using this for my latex documentation which will also abide by these rules. As GitLab is externally hosted, it will be easily recoverable if there is a failure in the local storage.

Chapter 2

Design

2.1 Overall Architecture

2.1.1 Class Descriptions

2.1.1.1 Game Classes

The diagram shows the Entity Relationship diagram of the classes in the core game package.

Card This Card class created Card objects that will contain variables for the suit, the value of the card in relation to the other and the number displayed on the card such as a wizard or an ace. It will also contain methods to construct the card and getters and setters for each of the variables. Card will also contain a copying constructor for the use of the AI. This class will mainly be used in the generation of the deck but will also be used in the in other methods to refer to the suit and number(the value of the card in reference to the game, like ace being worth 14) used within the cards

Deck Deck is a class for create a deck object to be used for in the classes of game, round and the Monte Carlo Tree Search. It contains variables for an array of cards, and array of the different values there are for the cards and the numbers they each will have. There is also a Boolean variable to check if the deck has been shuffled. The deck class will also contain values to create and the wizards and jester cards to be added to the normal 52 card deck that is generated with a different method. there will also be a method to shuffle the deck of cards so that each player gets different cards for each round.

Player The player class constructs the

Round

Rules

Game

Suit

Main

2.1.1.2 Aritifical Intelligence Classes

This section will show you the classes associated with Monte Carlo Tree Search AI.

Node

Tree

GameState

MonteCarloTreeSearch

UCT

2.1.2 Data Flow Diagrams

2.2 Some detailed design

In this section I will outline the important Algorithms to be used in the program and discuss how the User interface will looked and function.

2.2.1 Algorithm Design

2.2.1.1 Scoring

2.2.1.2 WinSim

2.2.1.3 Rules

2.2.1.4 Play Hand

2.3 User Interface

As this project is mainly based on the Artificial Intelligence, only a simple User interface is needed. So, using a Terminal UI seems the best choice. This will include showing the cards that the player has in their hand and the card they will play. This interface will also allow you to bid at the beginning of the round after the trump card and hands have been given out. After the round has been completed it shows the number of tricks that has been won, the score they have received based on their bid and trick they have won.

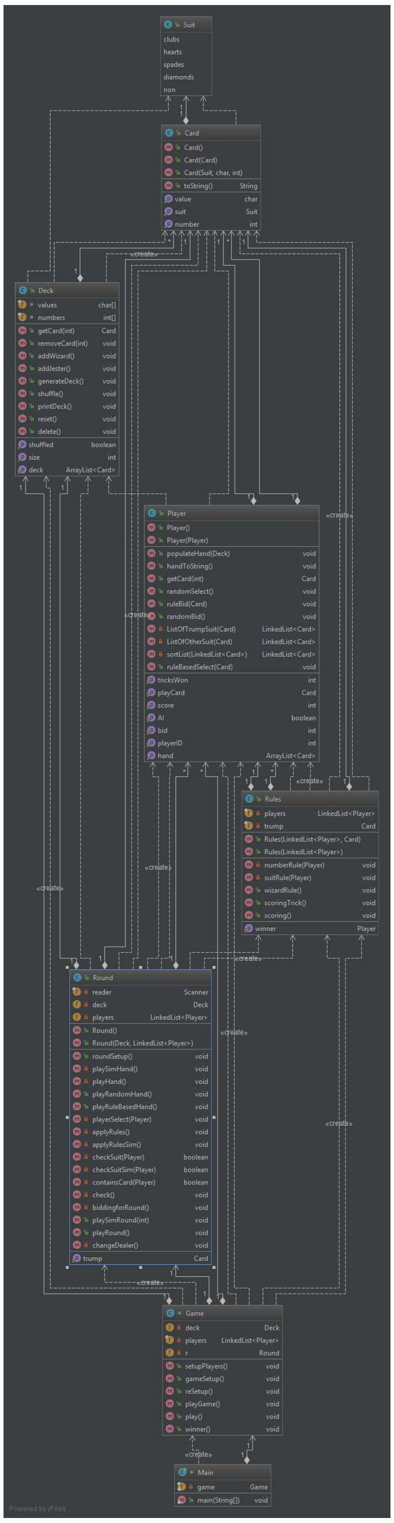

Figure 2.1: Diagram Showing the Class Diagram of the Game Classes

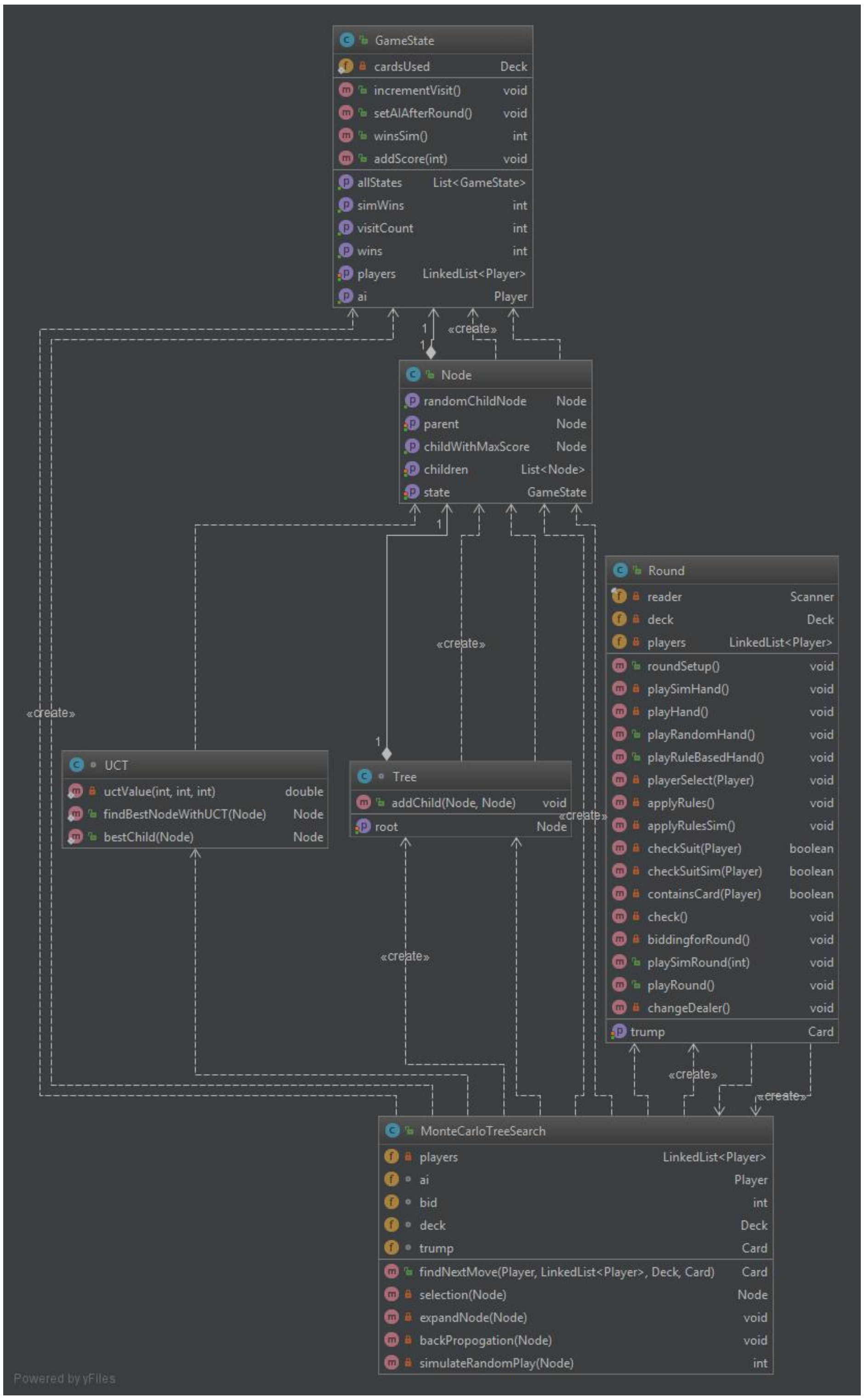

Figure 2.2: Diagram Showing the Class Diagram of the AI Classes and the linking Round Class

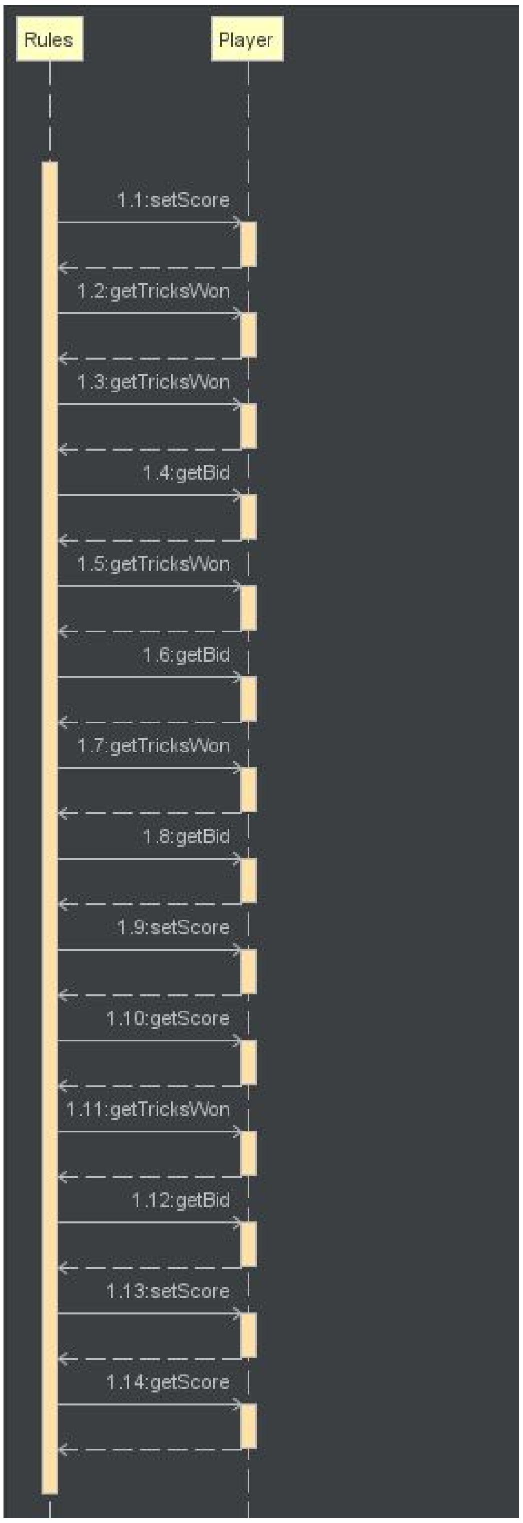

Figure 2.3: Sequence Diagram of Scoring Method for Algorithm Description

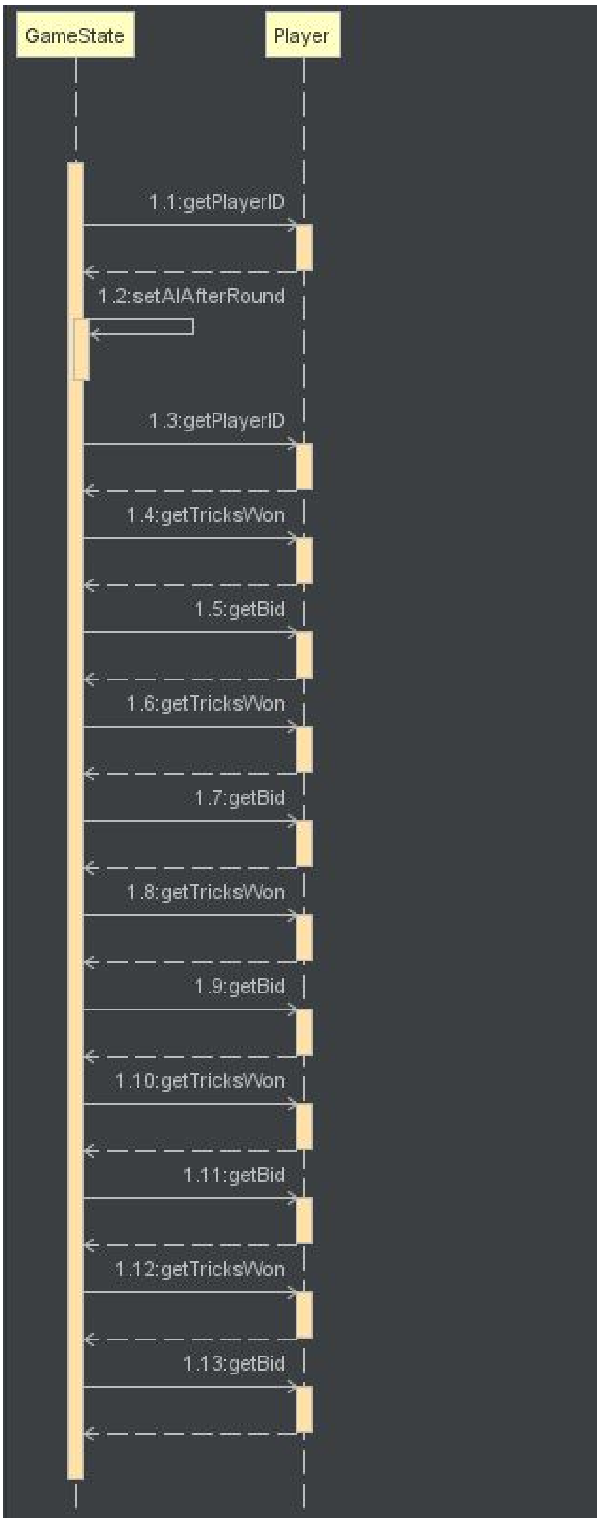

Figure 2.4: Sequence Diagram of WinSim Method of Winning simulation Algorithm

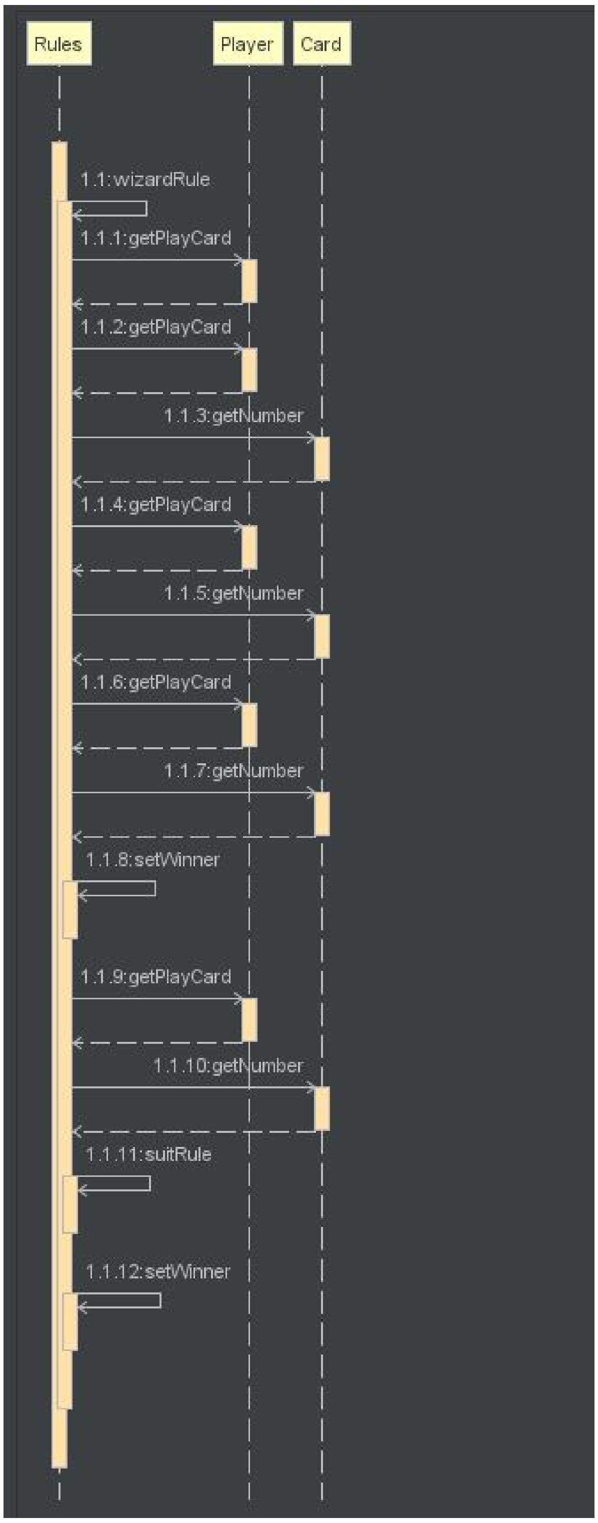

Figure 2.5: Sequence Diagram for the Rules Algorithm

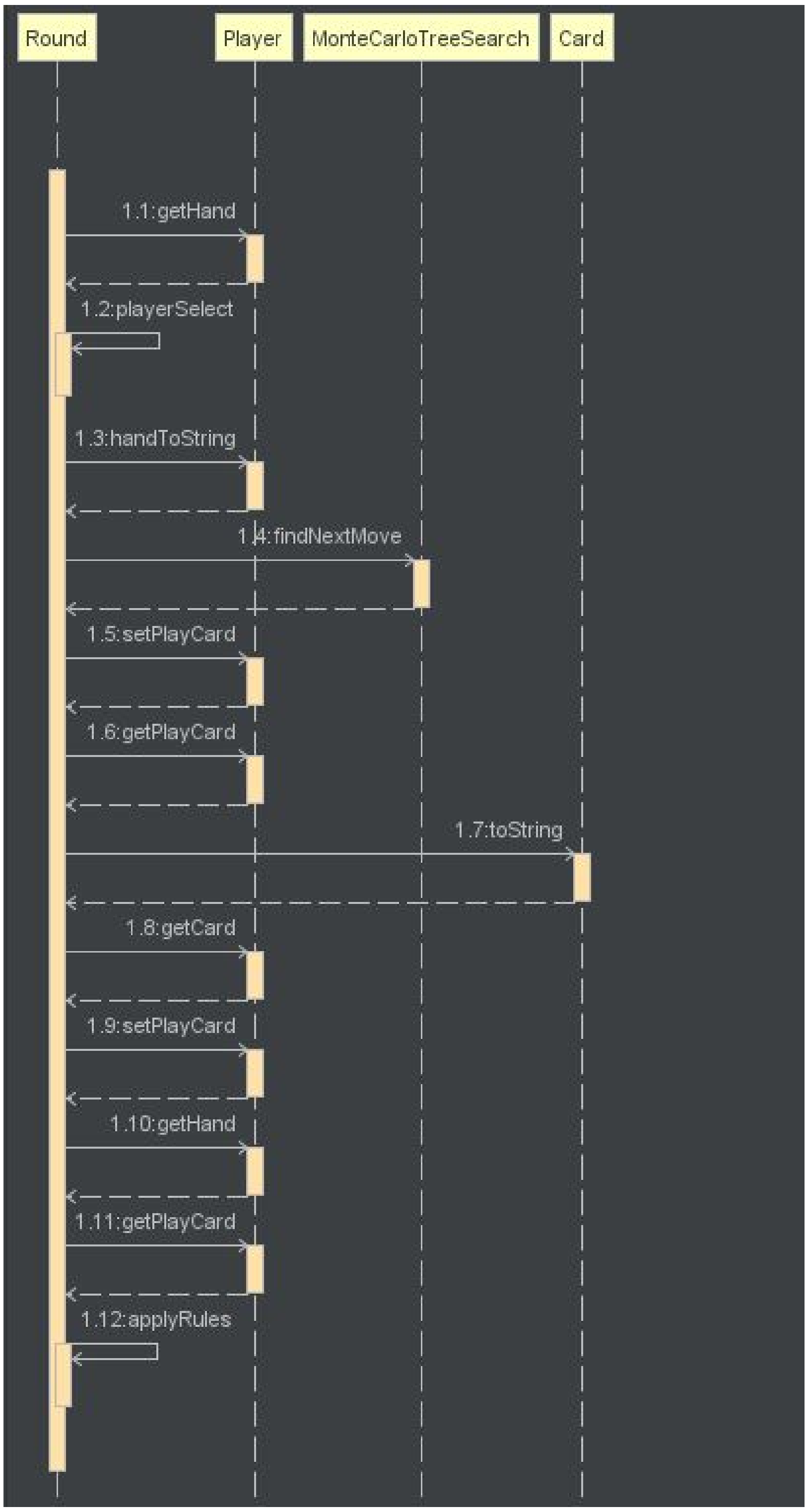

Figure 2.6: Sequence Diagram of Play Hand method for the Algorithm that shows what happens when a card is played

Chapter 3

Implementation

3.1 Introduction

In my implementation chapter, the way in which it built the project will be discussed and what was done in the sprints. As it has used a Scrum approach to build the Wizard card game, each iteration of the game has been split into sprints of which specific tasks had to be completed for the sprint to be done.

3.2 Sprints

3.2.1 Sprint 1- Setup

For me to begin the project, there was setup needed to be done to ensure that it can have a safe place to store the project and that it is built in. This was done by setting up a GitLab repository for the Wizard card game to be stored on allowing me to access different versions of the game as it progresses. I also made sure that there was access for my Supervisor to see the progress that I was making. It was also set up a private blog for me to keep track of the progress and problems faced during the implementation. It was also made sure that it had the correct IDE to use on my personal computer for the development of the game. The IDE that was chosen was IntelliJ IDEA 2018 and was chosen for it easy of use and the familiarity with the system. After this was installed and working correctly i added a few initial classes for a idea of what the game will look like. These classes include:

- AI - The classes to hold all of the code for the Artificial Intelligence Players that will be the main players opponents.
- Card - A constructor class that makes the cards to be used in the deck that initially only contained variables for the suit and value of the cards.
- Deck - This classes will make and array of cards containing the normal 52 cards

deck and the extra 4 wizards and jester cards. This will then be used in the player and game classes to use their cards in its array.

- Game - This class is to initially contain all the methods for applying the game rules and start the game.
- Player - the player class will contain methods to bid, play cards and see the cards the player has in his hands. It will also have variables for the score they have and how many tricks they have won in that specific round.
- Suit - Suit will be a enumerator classes that contains the 4 different suits of heart, spade, diamond and clubs. It will also have a null variable for the jester and wizard cards.

This classes will then give me a basic overview of how the project will be initially structured which can then be built on over time. The was problems to begin with with linking my local repository to the GitLab one but this was shortly fixed by doing it through the Git Bash rather than the website itself.

3.2.2 Sprint 2-Create Shuffled Deck

For this part of my implementation I tasked myself to create a deck for the Game to use and the for this to be shuffled randomly so it the cards every player will have will be different. This involved create a variable method for generated a normal 52 card deck and then adding the jesters and wizards afterwards. This in order deck would then use a shuffling algorithm [10], making sure that the player would get different hands each round. This had a simple UI that showed the initial generated deck with all the cards contained in it. This then ordered deck will be shuffled and printed out on the UI in it new shuffled state to show that it has worked and manually tested.

3.2.3 Sprint 3– Create Players and Play Cards

Creating Player object for the deck to be used for was the next section of work that need to be completed in this iteration. This classes contain variables for a bid to be set; an array of cards for there hand; there current score; how many tricks in that round that have won and the card that would currently be in play for a trick. This would then have getting and setter for the variables to be used. A method to populate the hand of 15 cards from the deck used in the round was also added. This allowed me to create a simple UI that showed the hand that the Player currently holder and be able to select a card from the deck to be used in the game.

3.2.4 Sprint 4 - Create Rounds and Simple AI

During this sprint, it was tasked to develop rounds for the game to be played and create a simple Artificial intelligence that would randomly bid and select cards from their respective hands. During this part of the process a problem arose in trying to use the AI class as an extension of the Player class. So, it was decided that the best course of action was to create the methods for the AI in the Player class so that it would still function in the game. Another class called round was also added into the system to handle each round of the game containing classes to play the players card until they had no cards left and change the dealer for each trick in the round. This allowed for multiple rounds to be added to the game when and if needed.

3.2.5 Sprint 5 - Apply Rules

Once the Game functioned properly and the AI was working in a simple but effective for testing way, the next step was to apply to rules of the Wizards card game so that a winner could be decide for each of the rounds. This involve adding a new variable to the player class that counted the amount of trick the player wold win for that round. This would then be used and compared to the bid so that a score can be given to the players. The rules that were added to check who has the best the cards this is done back the wizardRule(), suitRule() and numberRule() methods to find the best card, once the best card has been found the amount of trickswon for the player is increased by one. A new Rules class was added to implement the methods to be referenced containing variable for the players, the winner of the trick and the trump used for the trick.

Another rule that was added was the scoring of the players compared to bid, which checked the difference between the bid and the tricks won and gives a score based on that. I initially had problems as i made the rules a child class of the Round class so that I could reference the trump and players but it would found to be easier to make it a separate class that adds the trump and players from the Round class. Another rule also needed to be added too to make sure that it was working properly which was that the player can only play cards of the same suit at the lead player or the trump card if they are available in there deck. These methods were added into the Round class as problem arose when trying to implement them into the Rules class.

3.2.6 Sprint 6 - Monte Carlo Tree Search AI

For the last and longest of the sprint the Monte Carlo Tree Search Algorithm for the AI. For this sections, a separate package was created called MonteCarlo. The contains the classes for the Tree search to work this include GameState, MonteCarloTreeSearch, Tree and UCT. Below explains each of the classes and what they do:

- GameState - Contains the data for what state the game is currently in such as what cards are in the players hands and how many tricks they have won. It contains method to get all of the variable to be use in the tree search, an array of all the possible moves that the AI can make in that state and methods to increase the variables that shows the number of visits the node has had and the score it has gotten.
- MonteCarloTreeSearch - This is the classes that has methods to perform the tree

search itself, which includes the selection, expansion, simulation and back propagation parts of the search. This is done under a specific number of iterations to create the tree and then selects the best child node based on the visits and win score and choices the play card used in that node.

- Node - This a standard node class created for tree search containing variables for the parent of the node, the child nodes that link to it and the game state for that node. It also contains a class to show which child node has the maximum score in that game state.

- Tree - The tree class is used to create the tree to be searched to find the best card to be used by the AI. This contains a variable called root and getter and setter to retrieve it. This is then built upon by adding children to this node until a leaf node is reached.

- UCT - The Upper Confidence Bound applied to Trees (UCT) class is used in the selection part of the Monte Carlo Tree search and find the best node to expand upon based on the ratio of simulation score and the amount of time the node has been visited.

  During this part of the project that was a problem with the fact that the player objects were only be referred to as a shallow copy and not a deep copy when being used in the simulation part of the search [11]. This meant that it only simulated once as it was using the object in the game and not its own player object. This meant a deep copy method was needed to clone the player objects for use in tree search. Once this was done it seemed to work a lot better after a few adjustments to the UCT class.

Chapter 4

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Provide information in the body of your report and the appendix to explain the testing that has been performed. How does this testing address the requirements and design for the project?

How comprehensive is the testing within the constraints of the project? Are you testing the normal working behaviour? Are you testing the exceptional behaviour, e.g. error conditions? Are you testing security issues if they are relevant for your project?

Have you tested your system on “real users”? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

Whilst testing with “real users” can be useful, don’t see it as a way to shortcut detailed testing of your own. Think about issues discussed in the lectures about until testing, integration testing, etc. User testing without sensible testing of your own is not a useful activity.

The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

Evaluation

5.1 Introduction

5.2 Aim

list aim and how well at acheiving

5.3 Objectives

lisrt objective and how well acheieved

5.4 Design

5.5 Project Management

Would of been better to use mix of FDD and scrum.

5.6 Testing

use of unit testing and maual tests, white and black box. What could of been better.

5.7 Further Development

Examiners expect to find a section addressing questions such as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Other questions can be addressed as appropriate for a project.

The questions are an indication of issues you should consider. They are not intended as a specification of a list of sections.

The evaluation is regarded as an important part of the project report; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things in the work and aspects of the work that could be improved. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

In the latter stages of the module, we will discuss the evaluation. That will probably be around week 9, although that differs each year.

Annotated Bibliography

[1] "How to play:wizard video," July 2014. [Online]. Available: www.youtube.com/watch?v=kgRoZ-Yd7tg

[2] "Wizard card game desciptive overview," accessed in March 2019. [Online]. Available: https://boardgamegeek.com/boardgame/1465/wizard

[3] "Mcts with example code," Oct. 2019, accessed 16 April 2019. [Online]. Available: https://www.baeldung.com/java-monte-carlo-tree-search

[4] "Mcts source," accessed April 2019. [Online]. Available: https://www.geeksforgeeks.org/ml-monte-carlo-tree-search-mcts/

[5] "Introduction to neural networks," 2019, accessed: 11th April. [Online]. Available: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

[6] "Java-intro." [Online]. Available: https://www.w3schools.com/java/java_intro.asp

[7] "Reasons to use java." [Online]. Available: https://www.mindsmapped.com/java-advantages-and-disadvantages/

[8] "Texworks." [Online]. Available: http://www.tug.org/texworks/

[9] M. Kovic, "Why i write with latex(and why you should too)," 26th June 2019.

[10] "Shuffling algorithm," accessed in March 2019. [Online]. Available: www.geeksforgeeks.org/shuffle-a-given-array-using-fisher-yates-shuffle-algorithm/

[11] "Shallow and deep copying." [Online]. Available: https://javarevisited.blogspot.com/2014/03/how-to-clone-collection-in-java-deep-copy-vs-shallow.html

[12] "Mcts description," accessed in March 2019. [Online]. Available: mcts.ai/about/index.html

Appendices

The appendices are for additional content that is useful to support the discussion in the report. It is material that is not necessarily needed in the body of the report, but its inclusion in the appendices makes it easy to access.

For example, if you have developed a Design Specification document as part of a plan-driven approach for the project, then it would be appropriate to include that document as an appendix. In the body of your report you would highlight the most interesting aspects of the design, referring your reader to the full specification for further detail.

If you have taken an agile approach to developing the project, then you may be less likely to have developed a full requirements specification. Perhaps you use stories to keep track of the functionality and the 'future conversations'. It might not be relevant to include all of those in the body of your report. Instead, you might include those in an appendix.

There is a balance to be struck between what is relevant to include in the body of your report and whether additional supporting evidence is appropriate in the appendices. Speak to your supervisor or the module coordinator if you have questions about this.

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. If third party code or libraries are used, your work will build on that to produce notable new work. The key requirement is that we understand what your original work is and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

The following is an example of what you might say.

Apache POI library - The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the client's existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [**?**]. The library is released using the Apache License [**?**]. This library was used without modification.

Include as many declarations as appropriate for your work. The specific wording is less important than the fact that you are declaring the relevant work.

Appendix B

Ethics Submission

Image not readable or empty
/aber/www/webapps/ethics_research/assets/img/ethics_pdf_header.png

26/03/2019

For your information, please find below a copy of your recently completed online ethics assessment

Next steps

Please refer to the email accompanying this attachment for details on the correct ethical approval route for this project. You should also review the content below for any ethical issues which have been flagged for your attention

Staff research - if you have completed this assessment for a grant application, you are not required to obtain approval until you have received confirmation that the grant has been awarded.

Please remember that collection must not commence until approval has been confirmed.

In case of any further queries, please visit www.aber.ac.uk/ethics or contact ethics@aber.ac.uk quoting reference number **12484**.

Assesment Details

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
esh3@aber.ac.uk

Full Name
Eithen Dennis Steven Howard

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 - Wizard Card Game

Proposed Start Date
28/1/2019

Proposed Completion Date
3/05/2015

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?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
My Project, "Wizard Card Game", will look at implementing a Monte Carlo Tree Search AI for selecting playable cards in the version of the game that i have created.

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?
No

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:

Appendix C

Code Examples

For some projects, it might be relevant to include some code extracts in an appendix. You are not expected to put all of your code here - the correct place for all of your code is in the technical submission that is made in addition to the Project Report. However, if there are some notable aspects of the code that you discuss, including that in an appendix might be useful to make it easier for your readers to access.

As a general guide, if you are discussing short extracts of code then you are advised to include such code in the body of the report. If there is a longer extract that is relevant, then you might include it as shown in the following section.

Only include code in the appendix if that code is discussed and referred to in the body of the report.

3.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [**?**].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
```

```
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
  /*-----------------------------------------------------*/
  /* Minimum Standard Random Number Generator            */
  /* Taken from Numerical recipies in C                  */
  /* Based on Park and Miller with Bays Durham Shuffle */
  /* Coupled Schrage methods for extra periodicity       */
  /* Always call with negative number to initialise      */
  /*-----------------------------------------------------*/

  int j;
  long k;
  static long idum2=123456789;
  static long iy=0;
  static long iv[NTAB];
  double temp;

  if (*idum <=0)
  {
    if (-(*idum) < 1)
    {
      *idum = 1;
    }else
    {
      *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7;j>=0;j--)
    {
      k = (*idum)/IQ1;
      *idum = IA1 *(*idum-k*IQ1) - IR1*k;
      if (*idum < 0)
      {
        *idum += IM1;
      }
      if (j < NTAB)
      {
        iv[j] = *idum;
      }
    }
    iy = iv[0];
  }
  k = (*idum)/IQ1;
  *idum = IA1*(*idum-k*IQ1) - IR1*k;
  if (*idum < 0)
  {
    *idum += IM1;
  }
  k = (idum2)/IQ2;
  idum2 = IA2*(idum2-k*IQ2) - IR2*k;
  if (idum2 < 0)
  {
    idum2 += IM2;
  }
  j = iy/NDIV;
  iy=iv[j] - idum2;
  iv[j] = *idum;
  if (iy < 1)
  {
    iy += IMM1;
  }
  if ((temp=AM*iy) > RNMX)
  {
    return RNMX;
  }else
  {
    return temp;
  }
}
```