Wizard Card Game

CS39440 Major Project Report

Author: Eithen Howard (esh3@aber.ac.uk)
Supervisor: Dr/Prof. Christine Zarges (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 3rd May 2019

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 3rd May 2019

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

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]

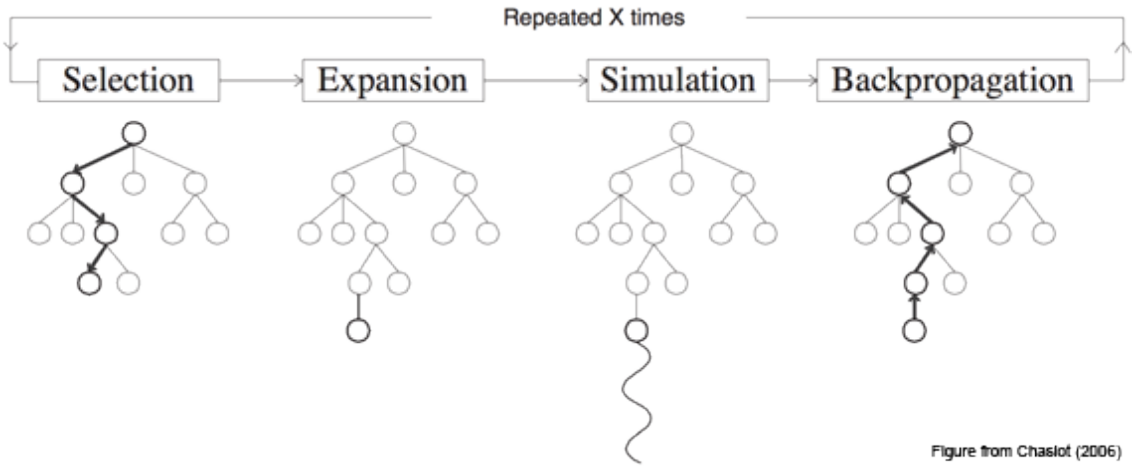

Figure 1.1: Diagram Showing Steps in Monte Carlo Tree Search

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] It consists of 3 different layer of input, output and another layer than transforms the input into something the output can use [6]. in this hidden layer will use learning rules that give different weights to the connections is the node as the neural network learns. One of the rules would be backpropagation, this is also used in the MCTS to apply scores to nodes that are promising. As it is a learning rule, it would require some data for the ANN to learn what the best options for what it is being use for are, such as the best card to use in a trick for this game. However, this make the use of this quite tricky as you never really know what is happing in the hidden layer of the network just like the black box test. This can make it difficult to optimise. It may also take a lot longer than using the MCTS to setup, as it would need sufficient data from the player the game to be competent at the game. This would take up a lot of processing power and memory for this be to achieve. However, this may be used with the Monte Carlo tree search to make are more effective AI in the future.

1.1.3.2 MinMax

For the Minmax algorithm to work it assumes that all of the players in the game are trying to pick the optimum choice. Meaning that the plays are playing for the opponentâĂŹs maximum loss. This algorithm would mainly be use for two player game like tic-tac-toe so that each player can pick their best and worst moves in turn when generating the search tree, it will create until a leaf node is reached. [7] The leaf nodes will then be scored to say if its a win, draw or lose. These scores are then check compared to the others under the child of the root node and the minimum value is picked for each. At this point went the scores reach the root node, the maximum value will be picked from the child nodes as the best option. This can be improving with the use of alpha-beta pruning which removes all of the nodes that will affect what happens in the final result. This would be great for the use of the game, however the is an added complexity of having a 3rd player which will increase the size of the tree(which the size can be controlled in MCTS).Also the minmax algorithm that all of the values are known by Ai which can be difficult to do for the wizard cards as you will never know what the trump may be and what cards the other players may have.

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

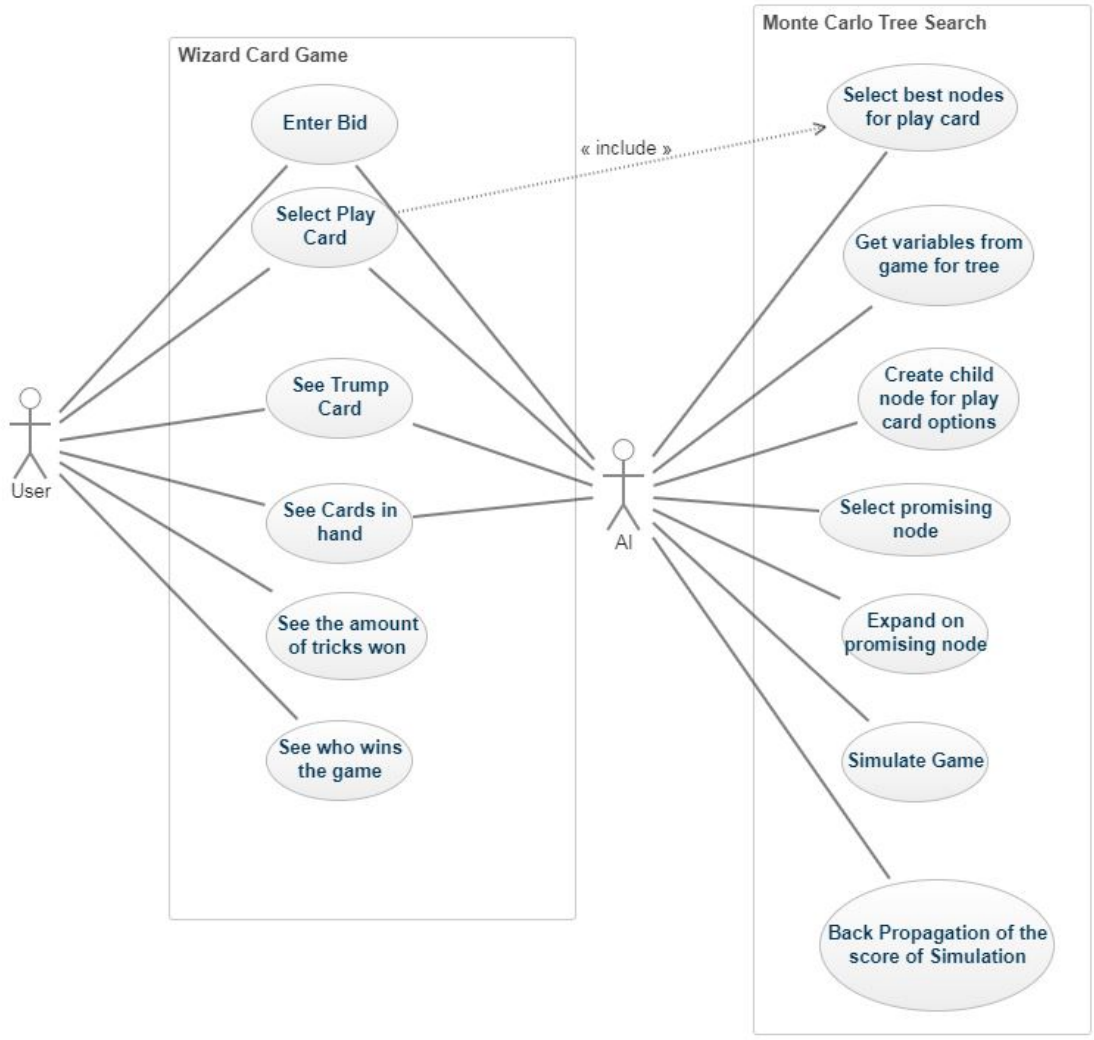

Figure 1.2: Diagram Showing the Use Case for the Project

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.
- 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 [8] 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 [9]. 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 [10] 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 [11]. 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

Implementation

2.1 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.

2.2 Sprints

2.2.1 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 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 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.

2.2.2 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 [12], 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.

2.2.3 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.

2.2.4 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.

2.2.5 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.

2.2.6 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 [13]. 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 3

Design

3.1 Overall Architecture

3.1.1 Class Descriptions

3.1.1.1 Game Classes

The game class section will describe the different classes used within the project and where they link with the use of diagrams to help illlustrate.

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 player objects that will contains variable for the player id; the cards they currently have in there hand; there bid and score; the card they have played for the trick; how many trick they have won and if it is an AI or normal player. This class will contain the getters and setts for the these variable whenever they will be

need. It will also contain methods to print out the hand they currently have; constructors to copy the players variables; to populate the hand of the deck from cards they are within the deck. the is methods to randomly and rule based selection the cards and bids for the more simple AI.

Round The Round class will contains variable that are received from the game class and to be used in the Monte Carlo package and rules class. It will contain method to reset specific variables to setup and reset up the round. This classes method will also the player to play a card for the main game and the simulation use in the tree search. There will also be methods to change the dealer, applies the rules to the cards that are played and to check if the cards are valid to play.

Rules The rule class contains variables to show who was the winner for that rule it was using, the list of players and the trump card used in the round. It contains method to see, check and compare the suits and numbers of the player and if it is a wizard card they have player the automatically win for that trick. It also will contain a method to score the players based on how many trick they have won and how close they are to their bid. This will contain the variable collected from the round and game classes.

Game The game class contains methods to setup up the game with variables for the array of 3 players and how many are human, the round and the deck used. This class also contains methods to reset up the game, how many rounds are players and who the winner of the game is.

Suit An enumerator class that contains the 5 different suits that the card could be such as heart, spades, clubs, diamonds and non for the wizard and jester cards.

Main This is the main class used to initialise the game and contains the game object for this to happen.

3.1.1.2 Aritifical Intelligence Classes

This section will show you the classes associated with Monte Carlo Tree Search AI it made more sense for the complex Ai to be separate from the main game.

Node The node class contains the variables to see who it parent node is and list of its child nodes and the game state that contains the cards used by the players and their getters and setters. Node also contains the methods to randomly pick a child node and get a child of the node with the maximum score by comparing the ratio of simulation score to the amount of time the node has been visited.

Tree The tree node is for the use of starting the tree and contains one node of which is the root of the tree where all the nodes expand from. It will contain methods to get and set this root and add child to it.

GameState Game State is used to get the variables from the game in the state it is currently in when the tree search is used. The means there will need to be variable for the players, the amount of simulation the state has won in the node, the amount of times it has been visited and the trump card. The class will also contain getter and setters for these variables, with methods to increment the visits and to set a score based on the simulation in the tree search. There will also be a method that will give a list of possible states that the game may have meaning the play cards they the player may choose to have.

MonteCarloTreeSearch This is the main class that will choose the next card that the player will using the Monte Carlo tree search. It will be linked to all of the different classes within this package as well as the variables it uses from the round class that it will perform the findNextMove() method. This method will repeat the methods of selecting a promising nod; expanding this promising node; simulating the game and back propagation of the node for several iterations until a suitably large tree is created. This tree will then be used to select the best child node for selection of the play card.

UCT The UCT(Upper Confidence applied for trees) is the used in the selection of the nodes and select the next node to expand based on the total visits, node wins and node visits. this class will contain methods to find the best child by comparing these variables mentioned previously.

3.2 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.

3.2.1 Algorithm Design

The algorithm here will have Sequence diagrams to show the methods to be used to apply to the algorithms.

3.2.1.1 Scoring

The scoring algorithm give a score to each of the player in the game by multiply the amount of trick by 10. The value is then reduced by the difference between the number of tricks it has won and the initial bid the player has made. However, if the player has same amount of tricks won to their bid, they will gain an extra 20 points.

3.2.1.2 WinSim

For this algorithms, it will give a based on how well the simulation has went for the player the MCTS is for. So if the score is equal to the bid it will be larger than if it was lower or higher than the bid.

3.2.1.3 Rules

For this algorithm, it will check to see if the player is a wizard or of the current winner is a winner. If current winner is it automatically wins. If not, it checks to see if it matches the trump suit or the dealers suit. If it does match the first players and is also trump it will check to see who has a higher valued cards and pick them as the winner but if it is a trump suit and the current winner isnâĂŹt then the new player will become the winner.

3.2.1.4 Play Hand

When the player is playing the hand, it will check to see if it is an AI or human player. If human, it will let the player select a card based on if it is a valid card to play. Otherwise, it is an AI, meaning the MCTS will be used to select the card. After this is will apply the rules to see who wins that specific trick.

3.3 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.

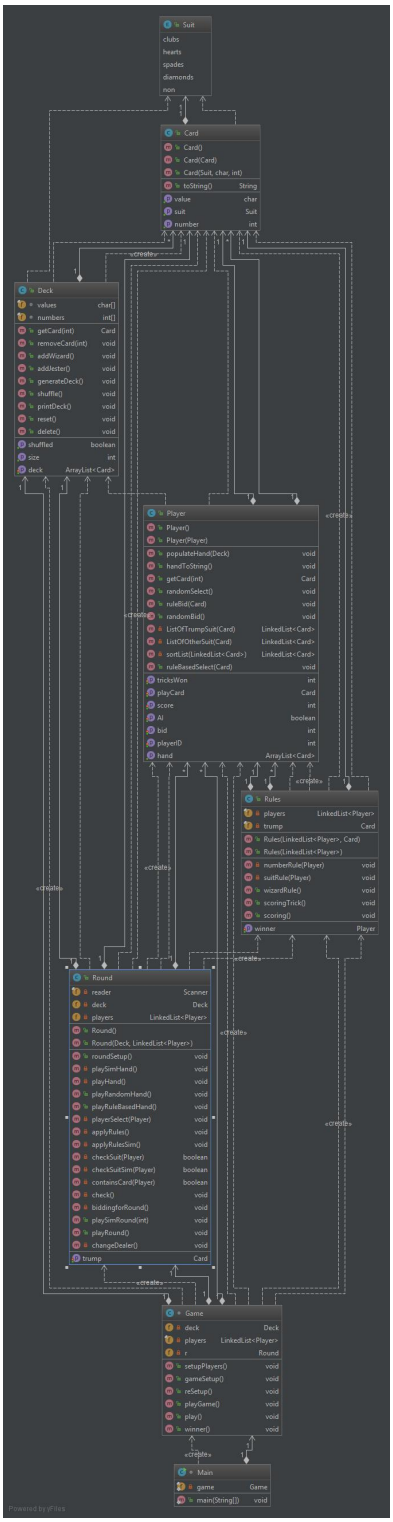

Figure 3.1: Diagram Showing the Class Diagram of the Game Classes

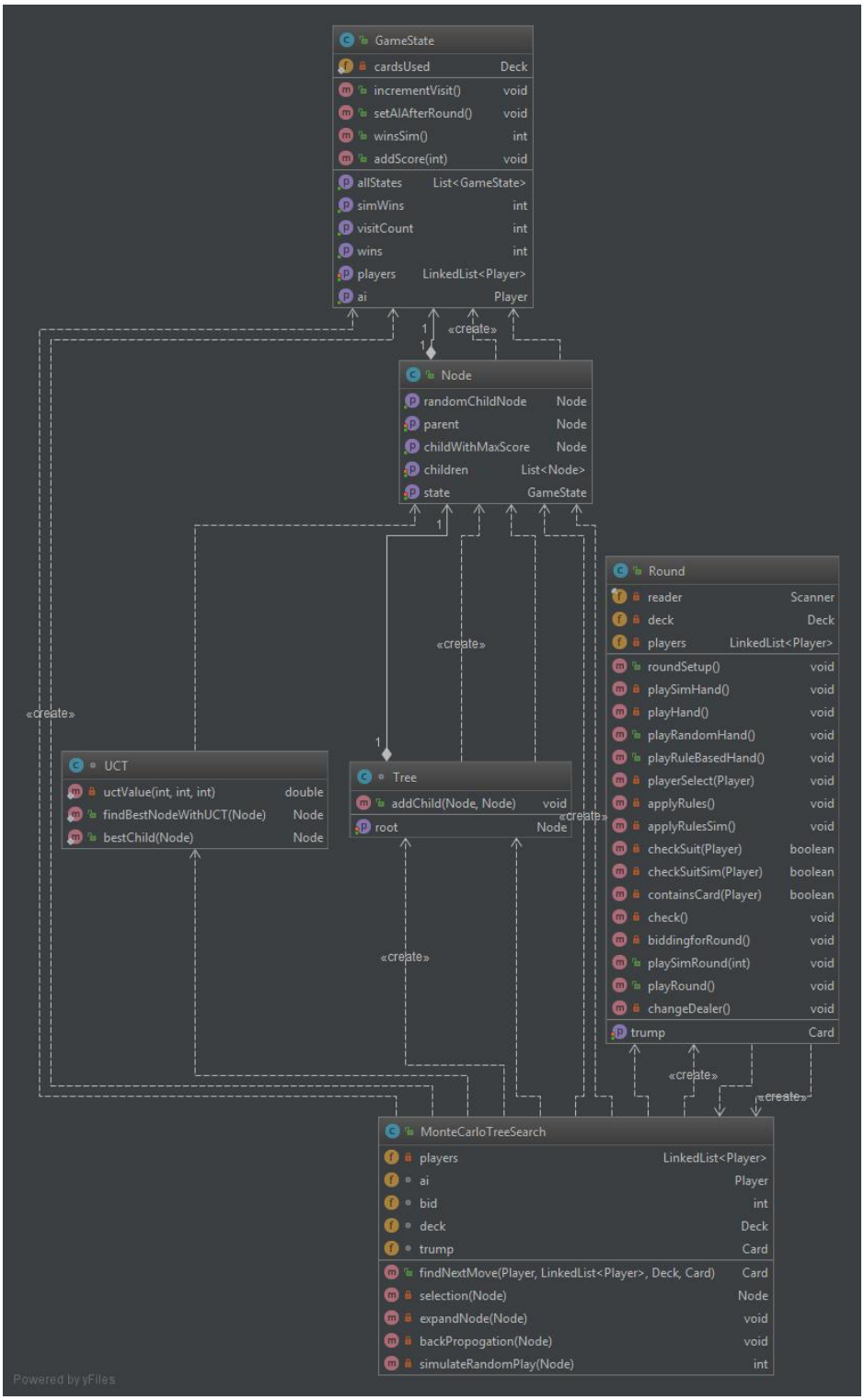

Figure 3.2: Diagram Showing the Class Diagram of the AI Classes and the linking Round Class

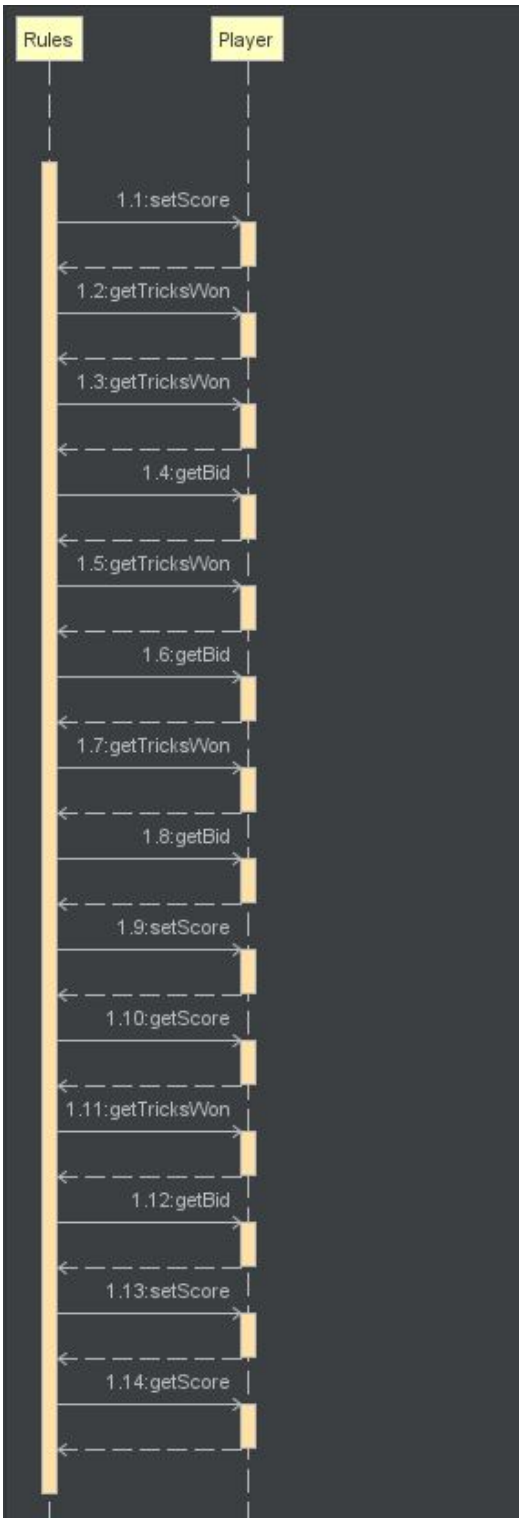

Figure 3.3: Sequence Diagram of Scoring Method for Algorithm Description

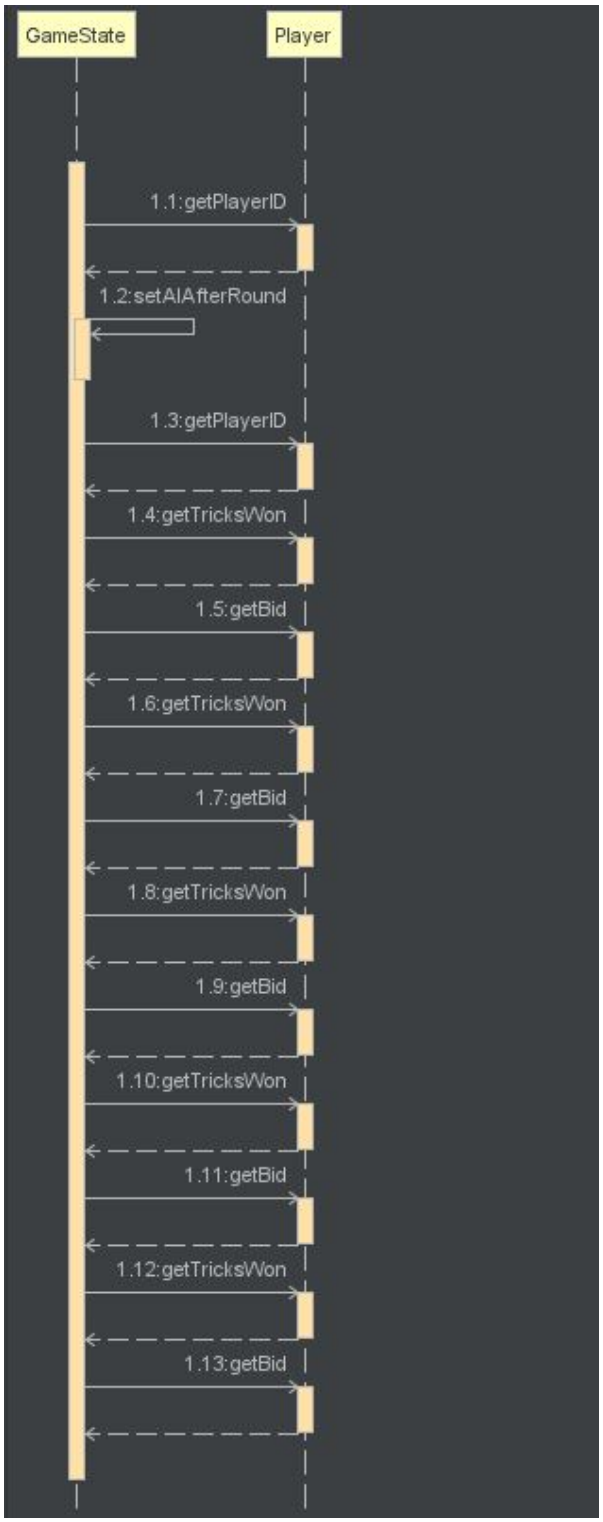

Figure 3.4: Sequence Diagram of WinSim Method of Winning simulation Algorithm

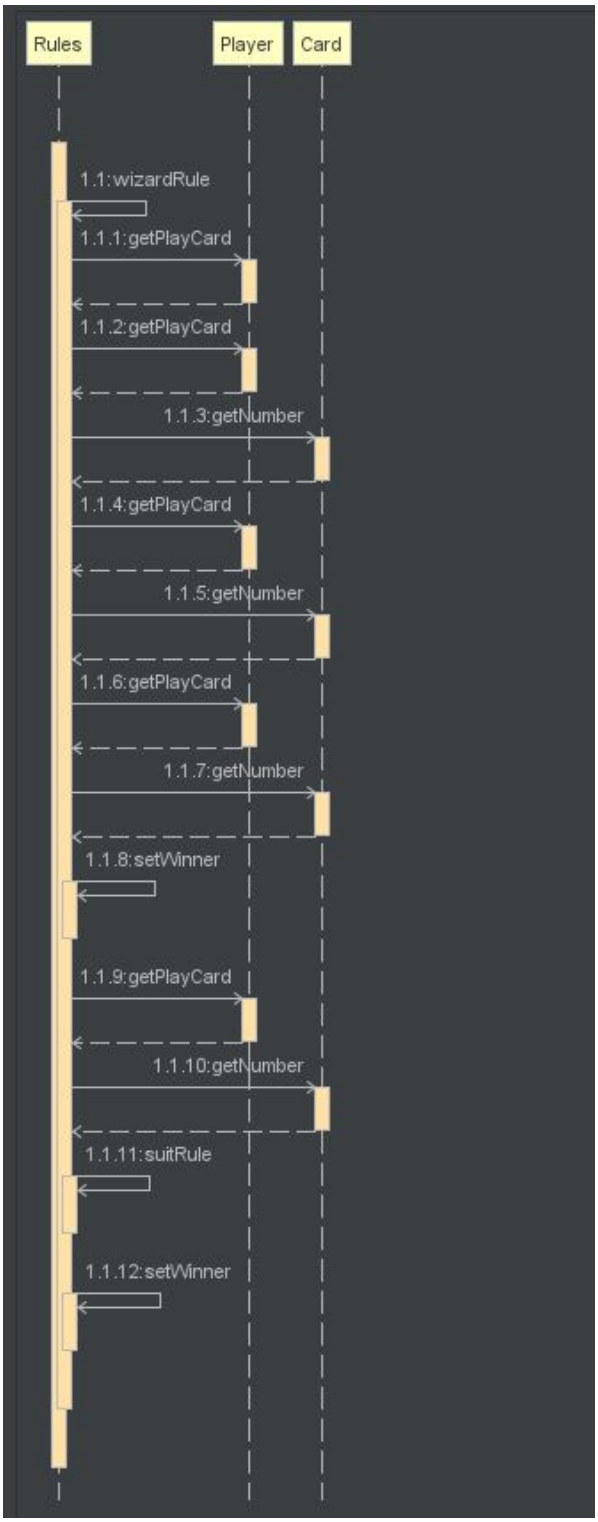

Figure 3.5: Sequence Diagram for the Rules Algorithm

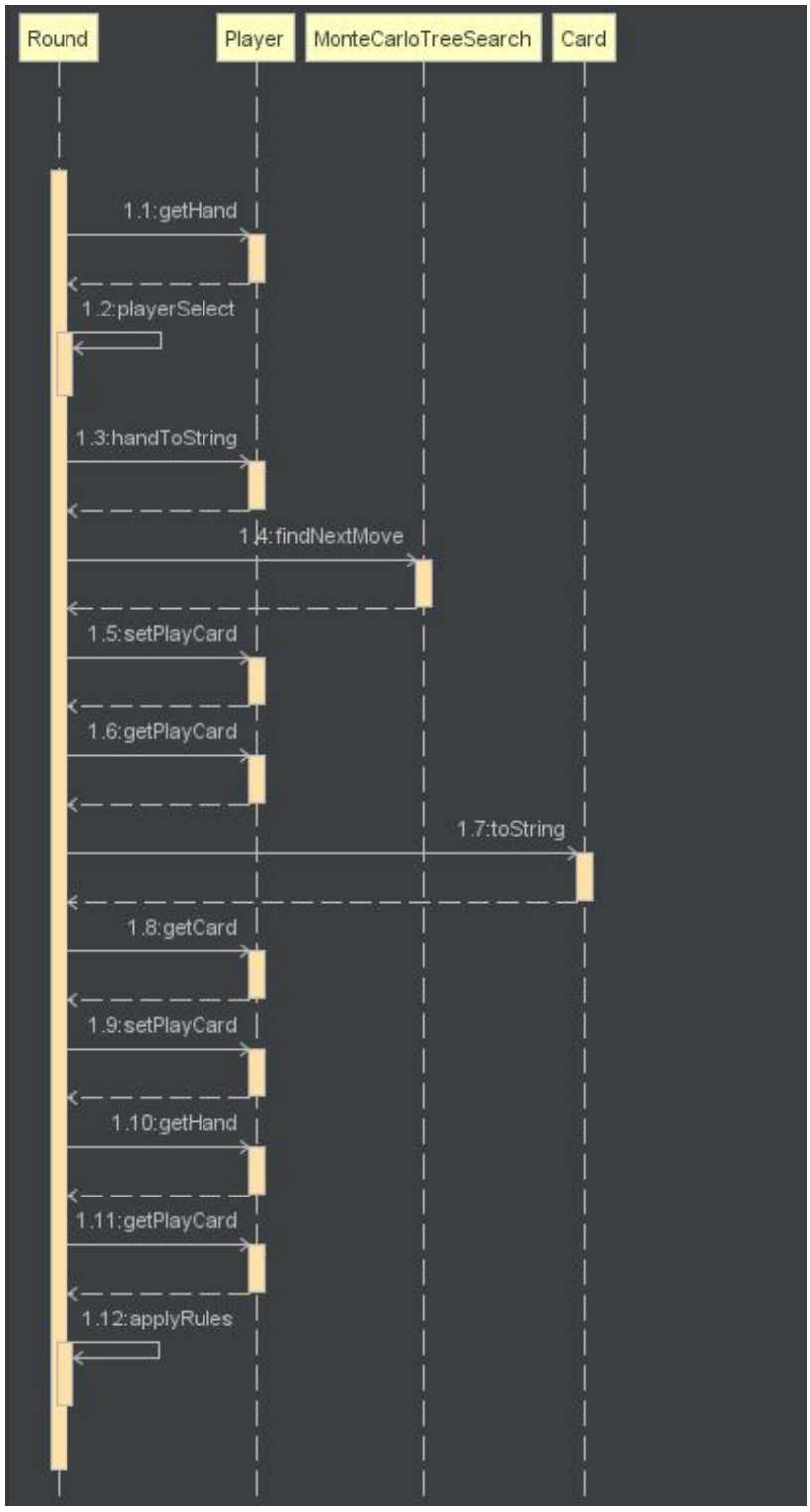

Figure 3.6: Sequence Diagram of Play Hand method for the Algorithm that shows what happens when a card is played

Chapter 4

Testing

For the testing aspect of this project it was decided to use the Junit package to test some of the functionality of the game with it AI. There will also be the use of Unit testing to check to see if the if the project has passed the basic needs for it to function properly. As the game was more create for the use of the Monte Carlo Tree search, testing with people to check it replay ability and how fun it was to play was not need.

4.0.1 Unit Testing

Junit was used during and after the project was fully functional to test the different aspect of the game that needed to work properly to ensure that it was up to standard. This involved create different Test classes to check each of the part. This test was mainly built to test the more vital parts of the system as if the core classes werenâĂŹt working properly, the more complex aspects wouldnâĂŹt either. This classed involved were:

- CardTest - To test that the card contains the suit, value and number that has been inputted.
- DeckTest - This class test that a 52-card deck with the extra 4 wizard and jester cards are created. It also tests that has been shuffled and that is does contain the wizard and jester cards.
- MonteCarloTreeSearchTest - The methods in this classes will check that the Tree search genrates a the UCT values correctly and that possible states are generated when the node is to be expanded upon as it is one of the crucial methods to create the tree of which to be search.
- RoundTest- This class will tests that the dealer do change after every trick is made; that when it play a hand, the correct person wins the tricks won point; checks that the hand of the player contains a card of the same suit as the trump or dealer and that a valid card is player by matching the dealer or trump card.
- RuleTest- For the ruletest classes, the methods will check to see if the wizard will be the winner overall against the other player; that the player who matches the trump

suit beats non trump suit matched player and that if they have the same suit, the higher valued card wins. There will also be method to make sure the jester always loses, and the first played wizard always wins. One more method will be in the class to check that each of the players will receive the correct score based on their tricks won and bid.

4.0.2 Acceptance Testing

Belows shows the acceptance test that the have been set for the game to functiion properly, it will conain information on what the test is, if its passed and if fail was was wrong.It also contains extra information if the test has passed but here is a problem that doesnt affect the main functionality to pass the test.

Test Number	Expected Functionality	Passed/Failed	If Failed,why?/ Extra information
1	Be able to enter a bid in the beginning of the round	Passed	
2	Be able to enter a Card to Play	Passed	Player must select card based on there allocated number displayed in the console
3	When selecting invalid card another can be selected	Passed	Displays Messages telling player to enter a card with a suit of the dealer or trump
4	If the player is the dealer any card may be selected	Passed	
5	If the first wizard card is played they always win the trick	Passed	
6	If Jester is played, they always lose the trick	Passed	
7	If two players have the same suit, the higher valued card wins	Passed	
8	A normal 52 card deck is generated with 4 wizard and jester cards	Passed	
9	The deck is shuffled randomly before card are handed out	Passed	
10	Be able to play 5 rounds of the game	Passed	
11	Each player has 15 cards	Passed	Shown in console by everyone having 15 cards in their hand
12	A trump card is selected from the deck	Passed	

13	If player wins game a message is displayed	Passed	
14	Each player is scored based on how many trick they have won and how close to their bid they were.	Passed	
15	The dealer changes each time a round ends	Passed	
16	The game is simulated for the AI	Passed	
Test Number	Expected Functionality	Passed/Failed	If Failed,why?/ Extra information
17	The root node in the MCTS has child nodes with different cards that are played	Passed	
18	The selection of a node to expand uses UCT	Passed	
19	The best node is selected based on the one with the highest score to visit ratio	Passed	

As the game has passed all of it acceptance tests, it can be assumed that the game works based on the tests that it has to pass.

4.0.3 Manual Testing

There was also the use of manually testing the game. This was done by playing the game to see if anything unusual or unexpected would happen such an invalid card being played. This was made easier to diagnose use the Intellij's debugger so the problem could be found quickly. This shone light onto the problem of the user being able to enter a letter, symbol or number that was invalid for the bid or play card. This could be fixed by either allowing the input of the name of the card as it displayed on the console or doing a loop to show an error until a valid input can be accepted.

Chapter 5

Evaluation

5.1 Introduction

For this chapter of work it will be discussed how the project went and the quality of the game that has been created based on my opinion. It will also discuss the how well achieves the aims set and the area in which it could be improved. There will also be sections to cover the design of the program, how well the methodology I use went, the testing of the functionality of the game and the future development of the game.

5.2 Aim

For the project I aimed to create a functional software version of the card game wizard, which I believe was achieved to a satisfactory degree. This is because it follows all the rules set by the description of the game to the different limitations that I have set for the game. This includes having 3 players playing against each other in the game. Of the three different player 2 of them are AI that use the Monte Carlo Tree Search Algorithm to choose their cards and a rule-based selection for their bid. The player also has 15 cards each and automatically plays the last round of cards as they only have one card each.

I feel all the aim were achieved to a degree that I useable and the user can see that they AI make a good decision. The aims also included a use-case diagrams, showing all the different use-cases the project would have to pass to be a good game and AI to use. This was separated out into the Game and the AI to be used. I personally feel all of the use cases in the wizard card game were met quite well with he use of the command line interface, as it allowed for all of the necessary data and displayed the hand, trump card, selected cards, how many trick they won in the round and who has won the game. This allows the player to see all of data that would be needed to play the game and see how well the AI has perform for that game it has played. However, I feel that aspects of the AI could have been improved such as the scoring for the simulations so that a better card could been selected. I feel that the aims that I set of this project worked quite well as there was definitive things that needs to be done for the project that I knew would need to be

achieved for the work to be completed.

5.3 Design

Due to the use of an agile methodology, the design of the project changed a lot from what I was initially going to be structured like. I feel that separate out the Monte Carlo Tree Search from the rest was the best decision to make for it design so that there is a clear separation between the both of them ,allowing for more algorithms to be implements easily with a few minor changes to the main games class to add the algorithms. This is thanks to the use of the object orientation of java; however, python would have also been a good alternative for the use of AI as I found a lot of other programs used when implementing AI due to there being a lot of libraries there to use. Some of the problem I have with the way in which the project has been design is that some of the methods used for AI in the player class. This could have been solved by creating a super class for both the human and AI class so that they share the methods they both need but can each have their own separate methods to use for themselves. There are also a few duplicated methods throughout the classes which can by create a separate method contains the duplicate code but due to the time constraints, the main concern what to make sure that everything was functioning properly. Another part of the class structure that could be improved upon is to reduce the amount of methods in the round class, as there are specific classes in it to be used for the simulation of the game for the tree search. This can be done by making a class that extends the normal round class but contains different functions in the methods use for the simulation such as the text sent to command line

As for the User interface, I believe it has a good simplistic style with the use of the terminal that shows the data it needs too so that more focus can be put on designing the AI. However, I feel there could be better expectation catching for when an invalid variable is entered, for which I would allow the user to enter a value even after the error was made. I also feel that entering there could be something to show how well the tree search perform by showing how long it took to do the amount of iterations or a way to show how many nodes where created. Overall I feel that the design was created to a standard that is good enough for the game to functional but there is still room for improvement over all part of the game.

5.4 Project Management

During the process of the project, the values of the Scrum methodology were taken into consideration as the development progressed. Throughout this time, the use of sprint was being used to the best of my ability but, I found it difficult to keep to the sprints as it as keeping interest in the longer sprint was hard and I would find myself wondering to improve or do other parts of the implementation. If I was able to keep to these sprints, maybe the design of the project was of been to a higher standard than it already it and it would of left the project easier to improve upon. Another part I tried to keep up was my openness about where I was in the project, however I would feel disheartened when I felt I was falling

behind but the sessions with my supervisor would clear up these doubts. For this project I felt I kept to the methodology as best as I could, but a feature driven development and a mix of scrum would have been a better method as I was mainly focusing on developing different features within the game and Ai instead of what I was doing in the sprints. This would also allow to complete different parts of the project at the same time. So, if a problem arise in one part, then another part can be worked on until a solution can be found.

5.5 Testing

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

5.6 Further Development

In this section i will be discussed the different future development this project may have in the future to improve upon its functionality and look.

GUI The addition of a graphical user interface is com

Improving MCTS

Neural Network

Saving and Loading Game

Mobile Version

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] L. Dormehl, "What is an artifical network?" January 5th 2019. [Online]. Available: https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/

[7] R. Jain, "Minimax algorithm with aplha-beta pruning," March 31 2017. [Online]. Available: https://www.hackerearth.com/blog/artificial-intelligence/minimax-algorithm-alpha-beta-pruning/

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

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

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

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

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

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

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

Appendices

Appendix A

Third-Party Code and Libraries

M CTS This the website use to help build the Monte Carlo Tree search for the Wizard Card Game but was used for a tic-tiac-toe game. https://github.com/eugenp/tutorials/blob/master/algorithms-miscellaneous-1/src/main/java/com/baeldung/algorithms/mcts/montecarlo/UCT.java

Appendix B

Ethics Submission

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

Shuffle

```
Random r = new Random();
        for(int i = deck.size()-1; i>0;i--){
            int jposition = r.nextInt(i+1);
            Card j= deck.get(jposition);
             Card temp = deck.get(i);
            deck.set(i,j);
            deck.set(jposition,temp);
        }
        isShuffled =true;
```

Scoring

```
  for (int i=0; i< players.size(); i++){
            Player p = players.get(i);
            p.setScore(10*p.getTricksWon());
            if(p.getTricksWon() < p.getBid() || p.getTricksWon() > p.getBid()){
                int diff = Math.abs(p.getTricksWon() - p.getBid());
                p.setScore(p.getScore() - (10*diff));
            }
            if(p.getTricksWon() == p.getBid()){
                p.setScore(p.getScore() + 20);
            }
        }
    }
```

Scoring Simulation

```
int i = ai.getPlayerID();
```

```
    setAIAfterRound();
    for(int j=0; j<players.size(); j++) {
        if(i == players.get(j).getPlayerID())
        if (players.get(j).getTricksWon() == players.get(j).getBid()) {
            return 5;
        } else if (players.get(j).getTricksWon() < players.get(j).getBid() + 2 &
            return 3;
        } else if (players.get(j).getTricksWon() > players.get(j).getBid() - 2 &
            return 2;
        } else {
            return 1;
        }
    }
    return Integer.parseInt(null);
```

Rules

```
    private void numberRule(Player p){
        if(winner.playCard.getNumber() < p.playCard.getNumber()){
            setWinner(p);
        }
    }

   private void suitRule(Player p){
            if(((p.getPlayCard().getSuit() == trump.getSuit()) && (winner.getPla
                 numberRule(p);
               }
else if(p.getPlayCard().getSuit() == trump.getSuit() && winner.getPlayCard().get
                setWinner(p);
            }

   }

   public void wizardRule(){
        for (int i=0; i<players.size(); i++){
            Player p = players.get(i);
            if(winner.getPlayCard() != null){
                if ((p.getPlayCard().getNumber() == 15 && winner.getPlayCard().g
                    setWinner(p);
                } else if(winner.getPlayCard().getNumber() != 15) {
                    suitRule(p);
                }
            }
            else{
                setWinner(p);
            }
        }
```

```
}
```