School of Computing
FACULTY OF ENGINEERING

UNIVERSITY OF LEEDS

Developing an Artificial Intelligence Player for the Game 7 Wonders

Roque Madeira de Carvalho Chan

Submitted in accordance with the requirements for the degree of BSc Applied Computer Science

2017/2018

40 credits

The candidate confirms that the following have been submitted.

Items	Format	Recipient(s) and Date
Project Report	Report	SSO (02/05/18)
Project Report	PDF	Minerva (02/05/18)
Code	GitLab Repository	Supervisor, Assessor (02/05/18)

Type of project: Exploratory Software

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of Student) _____

Summary

The purpose of the project to create an artificial player that is strong enough to provide a good challenge to the human player in 7 Wonders. The reason for choosing 7 Wonders because its classification differs to the traditional board game such as chess, Go or Checkers.

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Brandon Bennett for all the support and time he provided throughout my project.

Secondly, I would also like to thank my assessor Dr. Isolde Adler for her feedback and resources.

Finally, I would like to thank my colleagues in Logik lab, especially Nabil Sadeg for the moral support and helping with the tests.

Contents

Chapter 1

Introduction

1.1 Introduction

Artificial Intelligence (AI) has always been understood, since its initial conception, as the capability of a machine to perform an intelligent human behaviour. As the field has become more organised and developed, AI has been steadily improving and expanding. In the 20th century 1950 an English mathematician named Alan Turing published a paper entitled "Computing Machinery and Intelligence" [17], which opened the door to the field that would later be called AI. This had been years before the community adopted the term Artificial Intelligence as coined by John McCarthy, however further background research shows of Denis Diderot formulating on the philosophy of AI way before its conception "If they find a parrot who could answer to everything, I would claim it to be an intelligent being without hesitation." [10]. Humanity has improved upon the concept of what it means by Artificial Intelligence and has led to questions such as "Can Machines Think" [17]. AI has been split into further different fields, although they do not come into interaction with one another they can end up interpreting and crossing into one another's fields, few examples of fields in AI come into topics such as Robotics, Problem Solving, Knowledge Representation, Learning. The driving research question behind my exploratory software is that "Is it possible to create an AI that provides a good challenge to the game 7 Wonders which is unique to these variants of board games, which are more traditional in the sense of its AI being basic, include; chess, checkers, Go".

1.2 The Aim of the project

The area in which I will be delivering and providing a thorough analysis in would be AI in board games and card game. The aim of my project is to create an Artificial player who can play and compete with the user himself or herself, this means an AI which can follow, comprehend and adjust itself to the rules of the game as well as being able to make changes to its decision based on its interactions of variables. The very project in itself is very exciting, the end aim of the project's goal is to create a command line version of the board game "7 Wonders" along with an AI player to play against. The game does not currently exist for the command line version, an application version does exist for mobile devices but not for computers on desktop or website. This project in itself will teach me

a lot, not just about AI development, but also the game development which is also very much my passion.

1.3 The Objective of the project

The overall objective of the project would be to:

- Produce a command line version of the game 7 Wonders
- Pursue and document my background research into the implementation of AI in board games
- Develop and Artificial player capable to play the game
- Test the AI against a human player
- Evaluate the performance of the AI

1.4 Deliverable

The minimum set of deliverables would include a report which comprises of:

- Command Line version of the board game, 7 Wonders
- Artificial Intelligent Player to play against
- Read-me file which states rules and how-to-play manual

1.5 Initial Planning

The project will begin by being split into 4 main areas:

- Report Writing: write a comprehensive report which will introduce the concept of AI, the project I will be pursuing as well as detailed documentation surrounding the project
- Implementing the Game: I will implement the game 7 Wonders, and have full working features of the basic game so the user is able to compete and play with the Artificial Intelligence that is created.
- Implementing the AI: I will implement the AI so that it is able to give the player a good challenge and is also capable of understanding rules and gameplay to a level where it is efficient and complete to a certain level. The AI can both be “clever”

and "stupid", depending on how it chooses to pursue the scoring of each score set. It can do very well in achieving one type of score but fail at the rest, therefore, score badly in the entire game thus making it "stupid".

- Testing: The game will include much testing to make sure all the bugs at most are ironed out and the game is able to run to a certain state in which it is stable. This will be provided with proof and documentation of testing.

The report will also be updated accordingly to as the development of the project progresses and this will ensure complete clarity when it comes to the functioning of the game and to where the game has developed to unto the final target.

1.5.1 Project Management

The method I used for the development process is called Waterfall [7], the Waterfall method is achieved by taking sequential steps towards the objectives, it is a less iterative and less flexible method. I have followed the Waterfall method exactly because it will mitigate the mistakes which could be encountered by myself whilst developing the project. The stages of the Waterfall method is structured in consist of Conception, Initiation, Analysis, Design, Construction, Testing, Deployment and Maintenance.

The way the structure was adopted in the project was omitting any stages that were not needed therefore Conception, Initiation and Analysis were taken out of the staging process because the game already exists in card form and therefore the rules had to be only transferred into a computer system and then the development project began. Deployment and Maintenance were also removed due to the fact that they were not needed in this type of project, this is because my project is an exploratory software project, therefore the production quality would not be considered important. I had been able to mitigate errors included, being able to design and think ahead of likely errors it would cause whilst planning had allowed me to overcome the mistakes. Another method was to thoroughly test the new code with the old one using Git (version control) so that I would have a backup and I could easily compare trace the changes made to the code and reverse the problem wherever it may be caused. To organise my tasks in a to-do list which would help me continue with my project in concrete steps so that I do not leave out any tasks in the process essential to the development of the project. Whilst developing upon waterfall, I had to clearly output my learned mistakes and problems onto the next one because it would help me overcome them, and this snowball effect would help me mitigate further mistakes by making me more conscious to them.

1.5.2 Timeline

The first two weeks are used to choose and decide on the project, these are weeks 2 and 3. Week 4 was the confirmation week where our chosen project had to be allocated and confirmed. Weeks 5 through to 7 were purely for research upon the project. Most of the first semester, weeks 7 to 11 were used for writing the intermediate report.

After submitting the intermediate report, the feedback was next and planning for the implementation of the game. Weeks 14 to 17 were planning for the examination period preparation and the exams. The time in the second semester was divided into half, one half would be creating and testing the game, the other half of the semester was testing and creating the AI. The report shall be updated alongside the development phases. The implementation of the game takes slightly more time than the AI because the AI requires a correct implementation of the game in order to have an equally corresponding result.

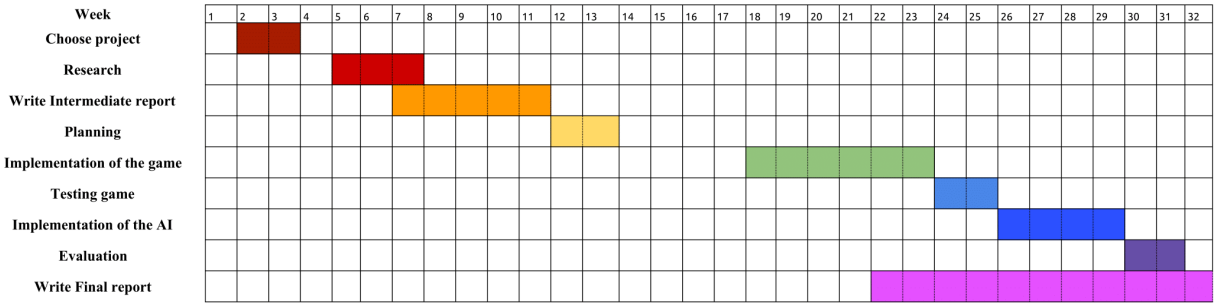

Figure 1.1: Project Planned Timeline

1.5.3 Revisit Timeline

The second semester was hectic and issues following the plan as shown in figure 1.2. This is due to other coursework deadlines and the amount of time I had to complete them as well as the objectives I had initially laid out for the final year project in semester 2. Therefore, the objectives had been changed to suit the newer time-line which made sure that with the time constraints that I had left in the project, the objectives would be fulfilled, this meant spending overtime working on the project, as I had very limited time to complete certain objectives, which introduced more difficulties for me to overcome when it came with the development process. Testing the game had been reduced from 2 weeks to 1 week of where the development of the AI is simultaneously carried out with the testing. The implementation of the game had also taken more time than I expected in which the implementation of the AI had been reduced from 4 weeks to 3 weeks.

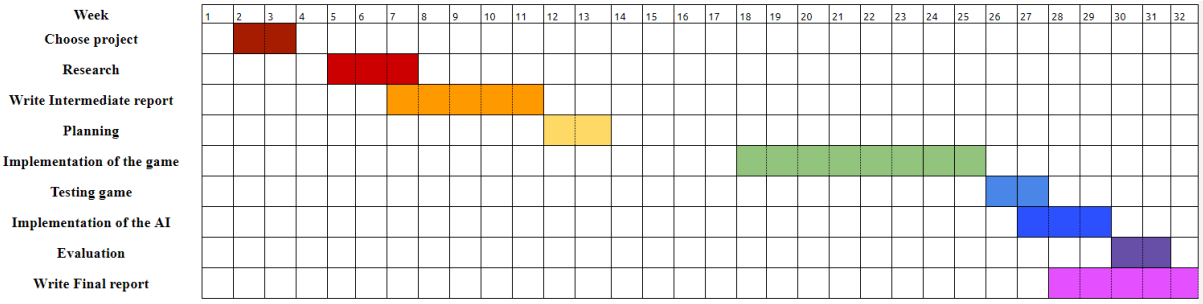

Figure 1.2: Actual Timeline

1.6 Relevant to Degree

The module that had relevance to this project is Artificial Intelligence to which I enrolled during the second year. This module helped me to understand the basic theory concept of AI and also provided some practical experiences on the implementation of AI. Besides the knowledge of AI, the module and other modules such as Web Services and Web Data, Web Application Development and Software Engineering also helped me during the project. Although those web based modules do not relate to AI, it helped me to sharpen my skill in python which was used to implement the command line of 7 Wonders.

Chapter 2

Background Research

2.1 Artificial intelligence

Much work has been done in the past on designing Artificial Intelligence (AI) programs to play "classic" board games, such as Chess, Checkers, Othello, and Go. Many of these games have programs that are sufficiently advanced that they beat the best human players [15][12][9][16]. In the last ten to twenty years, however, there has been a rise in "abstract" or "European-style" board games. These differ from the board games many of us have played as children (such as Monopoly or Life) in several areas: the games are typically short, many finishing in 90 minutes or less; the games usually emphasize player interaction in some way (components such as bidding, competing for scarce resources, or trading/negotiation are commonly seen); and the games often are based around hidden information, so that nobody can know the whole state of the game. These factors, especially the last two, make designing an AI for these games a challenge, and so much less has been done analysing these games, and what has been done has much room for improvement [11], although it may be difficult to create AI for these games, there is not much audience or return of investment for developers to create an AI for old and past board games, as there is not much demand reason for AI in that particular game will not be really needed. Another factor is that only mainstream games now stand out, and only they will have AI being developed while other game companies have been developing AI programs for sale, usually as mobile applications.

2.2 Game AI in real world

As examples of recent successes in the development of AI in board games, here are some game AIs that can outperform world-class human players most of the time.

2.2.1 Deep Blue

Deep Blue was a chess-playing computer developed by IBM [3]. It is known for being the first chess-playing computer system to win both a chess game and a chess match against a reigning world champion under regular time controls. However, Deep Blue does not in any normal sense know how to play chess strategically or understand it as humans do: it

mainly relies on searching through very large numbers of possible move sequences. This means that the program is using a brute force algorithm which simply calculating the best possible moves and going down that path, thus this proves the AI is not, in fact, all that clever but simply cheating the human out of play through its pre-recorded move set using brute force algorithm.

Deep Blue won its first game against a world champion on 10 February 1996, when it defeated Garry Kasparov in game one of a six-game match. However, Kasparov won three and drew two of the following five games, defeating Deep Blue by a score of 4-2. Deep Blue then evolved was then upgraded to include much more algorithms, and played Kasparov again in May 1997. Deep Blue then won by a score of $3^1/_2 - 2^1/_2$ and becoming the first computer system to defeat a reigning world champion in a match under standard chess tournament time controls, however, it did include its brute force algorithm which may raise a question. Kasparov accused IBM of cheating and demanded a rematch. IBM refused and retired Deep Blue.

2.2.2 AlphaGO

Google's AI subsidiary DeepMind has unveiled the latest version of its Go-playing software, AlphaGo Zero [1]. The new program is a significantly better player than the version that beat the world champion in Go earlier this year, but, more importantly, it is also entirely self-taught. DeepMind says this means the company is one step closer to creating general purpose algorithms that can intelligently tackle some of the hardest problems in science, from designing new drugs to more accurately model the effects of climate change.

The original AlphaGo demonstrated superhuman Go-playing ability but needed the expertise of human players to get there. Namely, it used a dataset of more than 100,000 Go games as a starting point for its own knowledge. AlphaGo Zero, by comparison, has only been programmed with the basic rules of Go. Everything else it learned from scratch. As described in a paper published in Nature today, Zero developed its Go skills by competing against itself. It started with random moves on the board, but every time it won, Zero updated its own system and played itself again and again. Millions of times over. After three days of self-play, Zero was strong enough to defeat the version of itself that beat 18-times world champion Lee Sedol. After 40 days, it had a 90 percent win rate against the most advanced version of the original AlphaGo software. DeepMind says this makes it arguably the strongest Go player in history.

2.2.3 Elon Musk's AI on Dota2

It happened with Chess and Go, and it finally happened with eSports this summer. Elon Musk-backed Artificial Intelligence company "OpenAI" [5] used a bot to wallop the best DOTA2 players in the world. To be honest, it was not even close. Instead of trying to program the perfect bot (a task that would have required an exhausting amount of programming due to the complicated nature of the game) OpenAI simply created a bot that learned through trial and error.

Over the course of playing thousands of games against itself, the bot kept the behaviours that lead to victory and shed the ones that got it killed. As it is shown in the video [6], the bot independently discovered established strategies and invented some things that professional observers are studying to implement in their own game-play.

OpenAI began its training in March of 2017 and by July it was already taking semi-pro players to his knees. By early August it was carving its way through the best players in the world, finally taking down the best human player in the world on August 10th.

2.3 Game Theory

2.3.1 What is Game Theory

Game theory [13] is the study of human conflict and cooperation within a competitive situation. In some respects, game theory is the science of strategy, or at least the optimal decision-making of independent and competing actors in a strategic setting. The key pioneers of game theory were mathematicians John von Neumann and John Nash, as well as economist Oskar Morgenstern.

An Example of Game Theory

One of the most popular and basic game theory strategies is Prisoner's Dilemma [14]. This concept explores the decision-making strategy taken by two individuals who, by acting in their own individual best interest, end up with worse outcomes than if they had cooperated with each other in the first place.

In the Prisoner's Dilemma, two suspects who have been apprehended for a crime are held in separate rooms and cannot communicate with each other. The prosecutor informs each of them individually that if Suspect 1 confesses and testifies against the other, suspect 1 can go free, but if suspect 1 does not cooperate and Suspect 2 does, Suspect 1 will be sentenced to three years in prison. If both confess, they will get a two-year sentence, and

if neither confesses, they will be sentenced to one year in prison. If they both confess and pinpoint the blame to each other, they will both get a five-year sentence.

While cooperation is the best strategy for the two suspects when confronted with such a dilemma, research shows that most rational people prefer to confess and testify against the other person rather than stay silent and take the chance that the other party confesses. Given the fact that no communication has been convened between two prisoners, the best solution would be to confess yourself as in either scenario whatever the other chooses, with the law of probability you will be given the least sentences.

2.3.2 Game type

Cooperative / Non-cooperative

In cooperative game theory [4], players can make binding agreements before playing the game, e.g. how to share pay-off. In non-cooperative game theory, on the other hand, agreements are not binding. This translates to individual players being the cornerstone of non-cooperative game theory, while cooperative game theory considers coalitions of players. Further along the line, equilibria are different, as the concept of a Nash equilibrium (no single agent can gain by unilaterally deviating) is not a very strong solution concept if a group of agents is able to gain by jointly changing their strategies. This last bit is, of course, only relevant if they can all agree on their strategies beforehand.

Symmetric / Asymmetric

An asymmetric game is when a game provides a payoff for playing a particular strategy that depends solely on strategies which are employed by the opposing member, not on the one who is directly playing them. If the identities of both players can be changed without changing the actual pay-off itself into the strategies then the game is symmetric. To give an example something like the prisoner's dilemma or the stag hunt are symmetric games and they follow a 2x2 format. Chess is another popular symmetrical game type.

Asymmetric games are games which are not identical in strategy sets for the players involved. For example, this would be similar to the dictator game where there are different sets and type of strategies of each player involved in the game. In exceptional cases, games can have different strategies involved for each player yet still be asymmetric.

Zero-sum / Non zero-sum

Zero-sum games provide more special cases of constant-sum games, this is in which the choices made by the players involved can neither increase or decrease the available resource. In zero-sum games, the total benefit to all players within the games, partaking to all the strategies involved is and always will be added to zero. An example of a zero-sum game would be the likes of poker, which ignores the possibility of the houses' cut because a player wins exactly the amount of one opponent's loss.

Simultaneous / Sequential

Simultaneous games are when both players move simultaneously, or if they do not move simultaneously, the later players are unaware of the earlier players' action, thus making it simultaneous.

Sequential games are games where later players have some knowledge about earlier actions. This does not have to be perfect information about every action of the earlier players, it can still be very little knowledge. For example, a player may know that an earlier player did not perform on one action while he does not know which of the other available actions the first player could have performed, this is seen in card games such as Pokemon or Yugioh.

Perfect information and imperfect information

A perfect information game consists of when all players know the moves which are made by previous players, games like infinite chess, Go, Tic-Tac-Toe and checkers are all examples of games which are perfect information.

Imperfect information [8] games include many card games, such as Poker, Yugioh, Bridge are games that show moves but the moves may be covered which are yet unseen by the opposing player, whereas complete will require the players to know of all strategies and all possible actions performed before they are.

Deterministic and Stochastic

Deterministic is the type of games where the same action performed in the same state will guarantee the same result. Hence, probability does not affect the gain of the action. A Stochastic game is where the actions performed inside the game happen by chance and are not determined, as defined by a "chance of nature". For example, a roll of a dice is a chance of nature depending on what number it may fall upon making it a variant of a Stochastic game.

2.4 Artificial intelligence algorithms

The algorithms the AI uses allow it to compare each and every move (path) and use it to optimally achieve its goal. This leads to the point where the AI is almost making decision-based upon certainty when not timed. An AI that has no time limit on making its decision would likely always result in a victory for the Artificial Intelligence depend on the computational power and algorithm being used such as brute force all possible move, in this example that game would be chess or go which had a large search tree of possible moves. The algorithm will make sure that the AI is the perfect player so to speak, in this sense the AI is fully aware of the rules of the game and makes decisions best on the best possible scenario for its turn.

For a game like 7 wonders, the AI cannot guarantee a win because of the luck element in some game states (in this case the players' wonder boards and the card sets dealt out at the beginning of each age), so it will give the player the best game it can without assuring a victory for itself. As we shall see, the AI developed uses a heuristic-based algorithm to select what move to play and this has been found to produce strong play in relation to human players.

Before considering the best approach to AI for 7 Wonders, we shall look at some of the most common techniques that have been used in game AI.

2.4.1 Minimax

The minimax algorithm is a way of finding an optimal move in a two player game. In the search tree for a two-player game, there are two kinds of nodes, nodes representing your moves and nodes representing your opponent's moves. Nodes representing your moves are generally drawn as squares (or possibly upward pointing triangles). These are also called MAX nodes[figure 2.1]. The goal at a MAX node is to maximize the value of the sub-tree rooted at that node. To do this, a MAX node chooses the child with the greatest value, and that becomes the value of the MAX node.

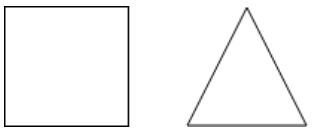

Figure 2.1: The representation of MAX nodes in the search tree

Nodes representing your opponent's moves are generally drawn as circles (or possibly as downward pointing triangles).These are also called MIN nodes[figure 2.3]. The goal at a MIN node is to minimize the value of the sub-tree rooted at that node. To do this, a MIN

node chooses the child with the smallest value, and that becomes the value of the MIN node.

Figure 2.2: The representation of Min nodes in the search tree

Here is an example of a simple game tree in which minimax is performed:

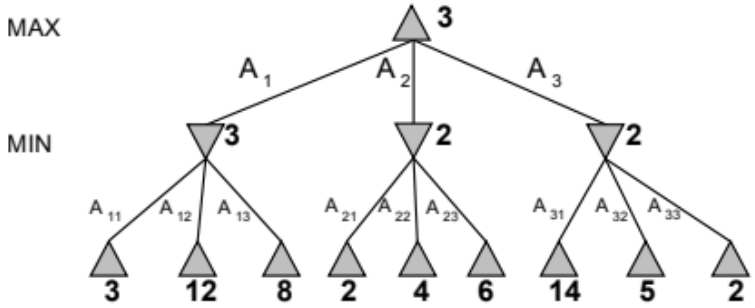

Figure 2.3: The representation of Min nodes in the search tree

2.4.2 Alpha beta pruning

Alpha-Beta pruning is not a new algorithm, rather an optimization technique for minimax algorithm. It reduces the computation time by a huge factor. This allows us to search much faster and even go into deeper levels in the game tree. It cuts off branches in the game tree which need not be searched because there already exists a better move available. It is called Alpha-Beta pruning because it passes 2 extra parameters in the minimax function, namely alpha and beta.

The algorithm maintains two values, alpha, and beta which represent the minimum score that the maximizing player is assured of and the maximum score that the minimizing player is assured of respectively. Initially alpha is negative infinity and beta is positive infinity, i.e. both players start with their worst possible score. It can happen that when choosing a certain branch of a certain node the maximum score that the minimizing player is assured of becomes less than the minimum score that the maximizing player is assured of (beta <= alpha). If this is the case, the parent node should not choose this node, because it will make the score for the parent node worse. Therefore, the other branches of the node do not have to be explored

2.4.3 Maxmini

Maximax (Optimist)

The Maximax looks at the best that could happen under each action and then chooses the action with the largest value. They assume that they will get the most possible and then they take the action with the best case scenario. The maximum of the maximums or the "best of the best". This is the lotto player; they see large pay-off and ignore the probabilities.

Maximin (Pessimist)

Unlike Maximax, Maximin looks at the worst that could happen under each action and then choose the action with the largest pay-off which is similar to minimax. They assume that the worst that can happen, and then they take the action with the best worst-case scenario. The maximum of the minimums or the "best of the worst". This is the person who puts their money into a savings account because they could lose money in the stock market.

2.4.4 Heuristic State

In computer science, specifically in algorithms related to path-finding, a heuristic function is said to be admissible if it never overestimates the cost of reaching the goal, i.e. the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path. A heuristic is a method that might not always find the best solution but is guaranteed to find a good solution in reasonable time. By sacrificing completeness it increases efficiency. Useful in solving tough problems which could not be solved any other way. Solutions take an infinite time or very long time to compute. The classic example of heuristic search methods is the travelling salesman problem.

Heuristic Search methods Generate and Test Algorithm

- Generate a possible solution which can either be a point in the problem space or a path from the initial state.
- Test to see if this possible solution is a real solution by comparing the state reached with the set of goal states.
- If it is a real solution, return. Otherwise repeat from 1.

This method is basically a depth first search as complete solutions must be created before testing. It is often called the British Museum method as it is like looking for an exhibit at random. A heuristic is needed to sharpen up the search. Consider the problem of four 6-sided cubes, and each side of the cube is painted in one of four colours. The four cubes are placed next to one another and the problem lies in arranging them so that the four available colours are displayed whichever way the 4 cubes are viewed. The problem can only be solved if there are at least four sides coloured in each colour and the number of options tested can be reduced using heuristics if the most popular colour is hidden by the adjacent cube.

2.4.5 Machine learning

Machine learning is a field of computer science that gives computers the ability to learn without being specifically programmed. Machine learning is closely related to computational statistics, these also intersect with one another. This also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter sub-field focuses more on exploratory data analysis and is known as unsupervised learning. Machine learning can also be unsupervised and be used to learn and establish baseline behavioural profiles for various entities and then used to find meaningful anomalies.

Within the field of data analytic, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytic. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

2.5 7 Wonder Board Game

Figure 2.4: The cover of 7 Wonders board game

The game 7 Wonders [2] also includes many expansions which can add to the rules and game-play of the basic version. There are also many other variants of 7 Wonders although in my report we will be going over the initial and most basic version of the game.

2.5.1 Setup

7 Wonders is a dedicated deck card game that features ancient civilizations each with their own resources and rewards of completing each stage of wonder. At the start of the game, each player randomly receives a game board called a "Wonder board". Each board is shuffled and each player flips a respective card for each wonder, this determines whether or not a player receives the "A" or "B" side of the wonder card [figure 2.5].

Figure 2.5: wonder card

Each of the wonder cards depicts one of Antipater of Sidon original Seven Wonders of the Ancient World in its own variant. Players place cards representing various materials

and structures around their Wonder boards and begin the game. The boards are double-sided, the wonders on side A are generally easier to build, while those on side B grant more interesting benefits and are for more experienced players.

7 Wonders is played over three ages, these are known in the game as Ages I, II and III, each age uses its own deck of cards. In each age, seven cards are randomly dealt with each player. The game uses a card-drafting mechanic, this means that once per turn, each player selects a card to play from his or her hand, then passes the remaining cards which are to be face down to the next player each turn. This process is repeated until five out of the seven cards have been played. At this point, each player must choose to play one of his remaining two cards and discard the other.

2.5.2 Rules

Each age card represents a structure, and playing a card is referred to as building a structure. To build a structure, a player must first pay the construction cost shown in the top left of the card [Figure 2.6], in coins or in one or more of the seven resource types, then lay it down by his or her Wonder board. A player lacking the resources available may pay his direct neighbours to use their resources without compromise, normally at two coins per resource, if available. The cost of trading resources can be reduced by building a structure which allows cheap trading.

Instead of building a structure, a player may choose either to discard an Age card to earn three coins from the bank or to use the card to build a stage of his or her wonder. The Wonder boards have from two to four stages, shown at the bottom of the board. In order to build a wonder stage, a player must pay the resource cost listed on the stage, then put an age card underneath the wonder board in the appropriate place starting from the base of the wonder (left-hand side) to the top of the wonder (right-hand side). There are seven types(two types of resource cards and five types of structure cards) of Age cards, representing different types of structures, which are determined by the colour of their background [figure 2.6]:

- Resource Cards:
  - Brown cards (raw materials) provide one or two of the four raw material resources used in the game (wood, ore, clay brick and stone).
  - Grey cards (manufactured goods) provide one of the three manufactured goods used in the game (glass, papyrus and textiles).
- Structure Cards:
  - Red cards (military structures) contain 'shield' symbols; these are added together to give a player's military strength, which is used in conflict resolution

at the end of each age.

- Yellow cards (commercial structures) have several effects: they can grant coins, resources and/or victory points or decrease the cost of buying resources from neighbours.
- Green cards (scientific structures): each card has one of three symbols. Combinations of the symbols are worth victory points.
- Blue cards (civic structures): all grant a fixed number of victory points.
- Purple cards (guilds) generally grant victory points based on the structures a player and/or his neighbours have built.

Figure 2.6: Wonder board with different sets of cards

Brown and grey cards only appear in the Age I and II decks, purple cards only appear in the Age III deck. Some cards provide a chain which allow the player to build certain cards for free [figure 2.7]

Figure 2.7: free construction library when the player built Scriptorium during age I

At the end of each age, military conflicts are resolved between neighbours. This is done by comparing the number of shield symbols on the players' red cards and awarding victory points accordingly. Once all three decks have been played, players tally their scores in all the different developed areas (civil, scientific, commercial etc.)

2.5.3 Scoring

Same as every other game, there can only be one winner which mean the player with the most victory points wins. In the base game, there are seven ways that allow the player to obtain point:

- Military victories - 1 point for each victory (having the most shields) during the first age, 3 for the second age and 5 for the third age. Each defeat makes a player lose 1 victory point regardless of the age.
- Gold coins - One point for every 3 coins a player hold at the end of the game.
- Wonder stages - Many of the wonder stages grant the player a fixed number of victory points.
- Civic structures (blue cards) - Each structure grants player a fixed number of victory points.
- Commercial Structures (yellow cards) - Some and only Age III commercial structures can grant player victory points based on certain structures a player and/or his neighbours have built.
- Guilds (purple cards) - The guilds provide several means of gaining victory points, most of which are based on the types of structure a player and/or his neighbours have built.
- Scientific structures (green cards) - Each green card has a symbol on it - tablet, compass or gear. One card of a type grants one victory point, but two cards grant four, the number of points granted is equal to the number of symbols possessed squared. Additionally, each set of tablet, compass and gear possessed is worth 7 points.

Chapter 3

Implementation of the 7 Wonders

3.1 Planning

Before beginning the implementation of the game, it is important to plan accordingly by analysing the set of game rules provided in the rules book. Besides the rules, it is also important to organise and construct a work-flow diagram to demonstrate how the game will be played. As I am following the waterfall project management approach, it is essential to create a strong and well-constructed plan which cover all the rules and action occurs within the game. It is a good method to plan in advance because the creation and purpose of the plan are to help with the foreseeing and management of the project, if there is a good plan in any project then a good strategy is in place, which will help to achieve the goals of the project's completion. It can also mitigate problems which may occur at any stage of the development, and plan them in advance helps to notify certain major problems which may arise.

3.1.1 Work-flow

To put the game into layman's terms, building upon the previous paragraphs mentioned before [section 2.5.1], the game begins with players selecting a board which comes with its own unique set of goals, in this game there are a total number of 7 boards, thus at most 7 players can play at the same time. All of the ages have their own sets of cards that they come with depending number of persons, these cards will be distributed among the players in which all player should receive exactly seven cards and that no one player will have a straight advantage before the distribution of the cards. These cards will help each individual player build towards a specific goal of completing their Wonder board as well as going forward to accumulate points which in the late game (depending on the player's strategy) will help towards having a higher score in the end.

The card drafting system will work simultaneously with all other players, as everyone takes their turn at the same time. This system begins with a player choosing a card, then choosing an action to play along with the chosen card, then the chosen card and actions are reviewed once everyone has chosen card and action. Then the players will put the rest of the hand face down and once upon everyone has done the same the cards are switched through other players depending upon the age of which the round is played and so the

next turn begins. Once the hands have been played until there is only one remaining card, that single remaining card will be added into the discard pile before the age will advance into next age where the new set of cards are introduced. The war stage will begin at the end of each age, this will be carried out with checking the number of military power each player has against its neighbour in which the one with the most military power comes out on the top with the points it earns as victory tokens, these depend on the age as a reward and the rest will receive defeat token. The picking process will then continue until the end of the third age, in which there will not be any more cards distributed and the final score will then be tallied up in order to determine the scoreboard and the victor.

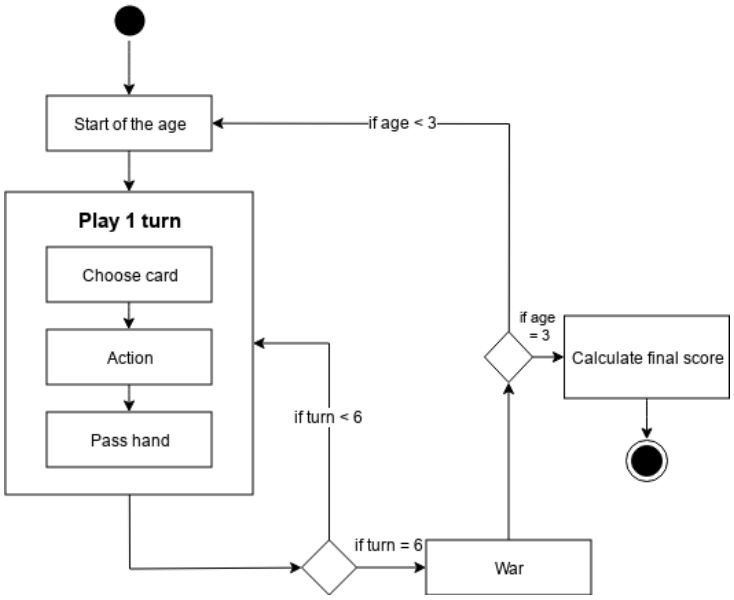

Figure 3.1: The work-flow diagram of the game 7 Wonders

3.2 Game Implementation

3.2.1 Language

The programming language which I have chosen is the Python. This language has been used to design the Artificial Intelligence player and the basic command line interface of the 7 Wonders. The benefits of Python include the fact that it allows you to make it very interactive, along with it being modular and dynamic which helps enable the user to carry out more precise and simple coded instructions to the compiler. The programming Language Python is also very portable and it is easier to understand in the sense that the syntax does not require a user to go to lengths to read and understand. It is a high-level programming language and can be extensible in other programming languages if need be.

The programming language chosen is Python, this is chosen over Java because it suits the purpose and the objective, another factor is that I am more knowledgeable in Python

than Java and Python also provides me more simple use cases which help me advance the building of the needed AI implementation which the game heavily depends upon. The biggest difference between the two languages is that Java is a statically typed and Python is a dynamically typed. Python is strongly but dynamically typed. This means names in the code are bound to strongly typed objects at runtime. The only condition on the type of object a name refers to is that it supports the operations required for the particular object instances in the program. This will make Python easy to write and understand as analysing the language code is as time competitive as any other language a user wants to learn.

3.2.2 User Interface

The user interface I have implemented for the application is the command line based interface [figure 3.2]. This is because the time constraints were limited and there is no better way to begin a game design other than to implement and test it first in a command line based structure (this means that in the future I could enhance upon the command line structure with greater visuals and switch entirely to a Graphical User Interface). Because the project is an exploratory software based project, the User Interface does not need to be up to modern standards for a release as its all experimental. However, the game heavily relies on graphics in order to achieve the full gaming experience, I have only managed to implement a simpler user interface due to the fact that it will take a long time to implement a whole graphical based user interface. The command line based interface is one of the text-based interfaces, with limited graphical qualities. This means that most of the time in development is spent on making the user interface look as clear and understandable as I can with the tools provided for such a Command line based interface structure. There is also lots of interactivity in the user interface as the player will have to be informed of certain instances of the gameplay, this means for example when the user has to build a certain card, it will give feedback to that user on how to build that card set with trading and also other options such as the cost related to purchasing resources for the neighbour to build that card set or complete a Wonder stage and display the reward of the next wonder stage.

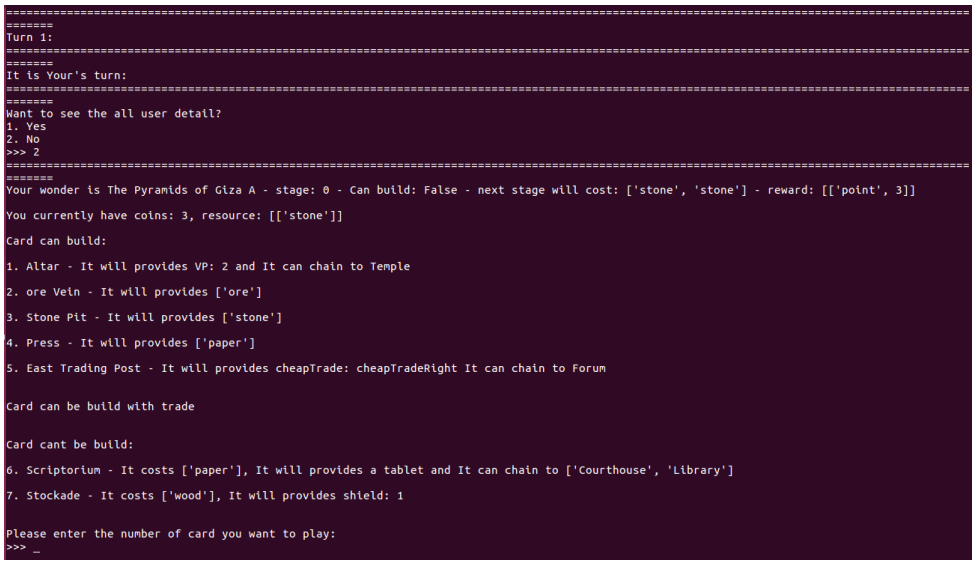

Figure 3.2: User Interface showing what action can be done on the card

3.2.3 Modification of the game

The game which is implemented is slightly different in comparison to the original. The games modifications have been adjusted to make it work with a computer-based game the changes and modifications will be reviewed as follows. The first modification to the game is to do with the fact that the game is played in a sequential sequence rather than a simultaneous sequence, this is because it requires techniques which when implementing the game myself I am not too familiar with, these include multi threading and the time constraint which would take longer to implement. This modification really changed the game type and strategy of the game as the player could plan their move depending on the previous players' move in the same turn which creates some sort of advantages over the previous player.

The next modification to the game came in the form of allowing the user to trade on their own initiative. This change welcomes the user to auto trade which is predefined rather than giving them options to trade or not, it is all automatically completed. This could be both a good or a bad change which the player does not need to keep track of the other players. However, this does limit some sort of strategy like not giving coins to the neighbour with less coin resource forcing them to waste a turn on discard card for coins.

3.2.4 Difficulty

The difficulties that have arisen whilst developing the game include such things as:

Upon creating an algorithm to check whether it is justified for the user to use the resources they may not have, this is done by an algorithm which would check the resources the user

has and see whether or not the user can make a move according to the resources. This was especially difficult to implement because it needs to find the most optimal way to compare both the players' resources and the needed allocation of resources to make a move that is being played by the user. This is more difficult to the fact that certain cards give you alternative resources in which where decisions have to be prioritised whether or not the player can select one resource at any given time as the player cannot select both. This difficulty was overcome through, removing all the single resources probability cases, this is done by rather than working out all probabilities, remove the ones that are single matched resources and work on the alternative resources as one of the resources will not be needed.

Here is a scenario to demonstrate how this would be a difficult problem to solve, and how the algorithm would solve this problem. The player currently possesses these sort of resources [figure 3.3 (a)] and the player is planning to build this card [figure 3.3 (b)] which requires 2 ores, 1 bricks, 1 stone and 1 wood. In this current problem, the Algorithm would have to check whether the player can build the card or not if the player has sufficient resources or not. The algorithm will first cancel out the wood and the ore because as the figure shows, the right-hand side cards provide only 1 single resources to the player and after cancelling the two single resources, the algorithm will then look for alternative resources of which the only one of the resources will be needed. In this case, it would be the first card on the left-hand side, of which brick would only be needed as wood has no longer be needed as it been cancelled out by the single resource, as next the algorithm will look for other alternative resources and so on. If both resources would be needed the algorithm will prioritise on the resources which cannot be purchased by neighbour rather than focus on the ones that can but would hinder gameplay for the player.

(a) **(b)**

Figure 3.3: Scenario 1 — player got resource cards "a" and want to build card "b"

The second difficulty was in calculating the best result from the green card set, this was difficult because it exists a card which provides alternative symbol in which it can be transfer into one of the three symbol and the green set is scalable, which works in the number of symbol to the power of two, and also each set would give you 7 points. This difficulty was overcome by calculating all the probabilities of the alternative card. This

can be done because there is only three possible symbol that the card can transfer to which mean that it can be easily computable.

The next scenario is about the green play card, the player owns the four green cards [figure 3.4 (a)], these are 1 tablet and 3 gear, which should contribute a total of 10 points to the player (1 from the tablet and 9 from the gear which is the number of gear to the power 2). However, the user also owns the purple card [figure 3.4 (b)], which could transfer into one of the symbols of the green cards, so the problem is which symbol should it transfer into, therefore allowing the user to get the maximum result. The algorithm will try to transfer the purple card into each symbol which then calculates the differences and take the maximum one. In this case, if the purple card transfers as a gear, the user will gain an extra 7 points. Whereas, if the purple transfers into a tablet it will gain 3 additional points and if the purple card transfers into a tablet, this will contribute a point because the user has completed a set which gives 7 points and also since its a symbol this will give an extra 1 point.

(a) **(b)**

Figure 3.4: Scenario 2 — player got green cards "a" and also got purple card "b"

Chapter 4

Implementation of AI

4.1 Planning

In order to begin the development phase of the AI there has to be an initial understanding of the game's classification. This allows the critical formatting of the approach as well as the required experience in the game playing, thereby a winning strategy can be constructed for the success of the AI.

4.1.1 Classification

Classification will talk about how the game types of game theory will fit into the game 7 Wonders. It will describe how each game type fits into the characteristics of the game 7 Wonders.

Below are the game types and how their characteristics are implemented inside 7 Wonders:

- Non-Cooperative/Cooperative — This game is non-cooperative but can also be treated as a cooperative game in a sense that the other players can try to focus on minimising the profit of the highest player through the game-play however through not using communication whatsoever, because players do not technically cooperate because there is no interaction that requires cooperation to settle agreements. Trading in 7 Wonders is not technically cooperating because the player the player who is selling resources to the purchasing player does not have a choice to refuse them.
- Asymmetric — The game is Asymmetric because different boards have different advantages and strategies relating to the boards, based upon what resources are available in the initial offering also the rewards that come with the wonder building stages.
- Non-zero-sum — The amount of gain from the player does not determine the loss the other players attain, as they just did not score so well. An example would be a card that would be worth ten points to your player would not be worth 10 points to other players, thus it would not be as big of a loss if they had that card since it is not worth as much as it is to you.
- Simultaneous — The game is simultaneous because every player takes their turn until everyone has finished deciding their cards, then they review their play and

mutually agree and finish their card actions at the end of the round, and proceed to the next round thereafter.

- Perfect Information/Imperfect Information — This is because every player has knowledge of what the player is building upon the board and what their board requires/provides. The game does also consist of some imperfect information such as the number of the purple cards, which depend on the number of players playing the game, therefore the player will not be able to know what purple cards can be available towards the last age stage of the game and the player could discard or sacrifice a card to build a wonder which the other player would not know whether or not which type of card is consisted of the player.
- Deterministic/Stochastic — When you perform any action you can determine the outcome of the card before it is being played. However, the game can also be seen as stochastic because of the distribution of the cards, because the number of cards is fixed, but because it is randomly distributed the player may not be able to obtain the cards. The purple play cards make it limited because the number of play cards depends on the number of players in the game, therefore, not all purple cards will be available in the game.

4.1.2 Winning Strategy

There is no one obvious and conclusive winning strategy for the game 7 Wonders. However, I have managed to deduct some overall guidelines that work well with all boards, after spending many games playing the 7 Wonders board game (physical) with my colleagues.

- Do not focus on any one single card type, this is because you have to balance the point in other scoring method types which are available because even though you can score really high on one particular card type, you will still lose because there exist more scoring method types.
- Military is essential towards the late stage of the game because the points of the military(scoring) will scale significantly in stages. According to personal experience, military cards have always been the last to be considered by players, but also equally provide just as much value to the score.
- Always complete your Wonder stages, this is due to the fact that it can support the player substantially in later stages and give a more variety of effects to help play.
- Coin and cheap trading are also important if you do not have a reasonable amount of resources because in the late game the resources will be required to play and build in the game effectively.

- Do not overlook into the scalability of the green cards, because it is easy for other players to know your winning strategy and react upon it (i.e. removing the green cards from the hand via discard or building wonder). However it is still quite beneficial of collecting green cards because of the chain building, it provides us two kinds of chain to build upon (the green and red chains) and collecting a set of green cards is sufficient enough for points in the green cards set.

4.1.3 Approach

The approach will begin with me creating an AI that will choose random cards and a random action for its move. This will help me to test the game much quicker, the reasoning behind will be mentioned in the testing phase later in section 3. It will include cards chosen at random but not chosen consequential to the best action, this will help set a minimum target for my AI to pursue the future development. There will also be a rule-based AI which would be used to capture intelligence, this will essentially be an AI that will follow the rules accurately, checking the statement (i.e. Check the Card) then proceeding to confirm that statement true or false. The rule-based AI will follow a set of rules that are implemented already within the game. The rule-based AI will follow the game as it is not flexible in its decisions.

After the rule-based AI, I am going to implement a heuristic state evaluation AI, which will evaluate the current hand, depending on the current state and also the neighbours, this AI algorithm will be potentially better than Minimax as it can deal with random distribution of cards, and is also more efficient in game that has large states (there will be 128 Billion states not including action, trading and random distribution of the cards), it sacrifices accuracy, completeness, optimality or precision in turn for speed.

Due to my own personal experience on the game, I have not chosen Minimax as my first AI algorithm to implement for this game because unlike chess as an example I do not have to think certain moves ahead more than what is necessary toward my play. In 7 Wonders the cards are swapping around players every single turn, which means that when you evaluate the hand, you may not actually be able to play the card on hand since you have to different hands. Besides my personal experience on the game, I did not choose Minimax due to the classification of the game type which Minimax would theoretically not be better as effective as Minimax is better for 2 player only games (a 2 player zero-sum game i.e chess).

4.2 AI Implementation

In this section, I will be mainly focused on the AI which will be implemented into the game and how it will and have help to achieve the aim of the project. Beside the implementation of the AI, it is important to implement functions which will return all the possible move in the current state and update the action based on the move the AI has chosen. The possible moves are define as a list which contain the card and action as shown [figure 4.1].

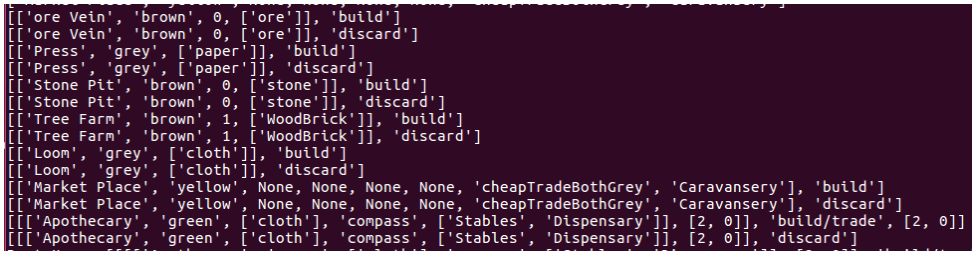

Figure 4.1: All possible move in the current state

Random choosing AI

The random choosing AI in which it will choose a random card out of all the option and from the chosen card, it will then randomly choose the action from all the possible action. The aim of this AI is to test the game and capture any bug that could potentially crash the game. This is important because, in order to test all the functions implemented in the game, it would require me to play all 7 wonder boards manually in a way to make sure that all wonders will be built and all card type will be built in at least one of the wonder city. This would take approximately 1 to 2 hours in order to do so and the process will be quite fatiguing especially when game crashing bugs occur, therefore focusing the testing to begin from scratch. However, since this AI will only choose randomly which, there may exist probability in which not all game crashing bug will be captured. Besides capturing bug, it also used to set a bare minimum result to test and evaluate the future developed AI as it could simulate inexperienced player in which, they have no idea how to play or score better in the game.

heuristic state evaluation AI

The Heuristic state evaluation AI in which it will estimate the value of each possible card and action perform to the card in the hand. The Heuristic approach will focus on speed rather the accuracy which means the move is chosen does not mean it is the best move. This is because Heuristic AI does not tend to analyse the actual contribution of the card, therefore, it increases speed. This approach is quite popular on traditional board game such as chess and go due to large amount of possible game state.

In order for the Heuristic AI to estimate the value of the card and action, A evaluation function is required. Evaluation function requires multiplier variable in order to estimate the value of move which does not provide direct benefit such as Victory Point in 7Wonder. In 7Wonder, multiplier variable is needed to estimate value for resource(brown card set and grey card set), coin gain and spend and the yellow card set (cheap trading). Beside those card set, the green card set also requires a multiplier variable simply even though it provides direct scoring. This is because the green card set is scalable and the initial value for the first card of each symbol is always 1 point which is a really low value compared to the blue card set. This will make the AI always avoid picking the green card set based on its value.

For this AI, I had implemented multiple evaluation functions depending on the card set and the wonder rewards.

For the resource-based reward, A player will only need maximum 4 for each brown resource and 2 for each grey resource. The evaluation function will return 0 when the player estimates the resource of which the player owns the maximum amount. The function will also consider the resource of the player's neighbours because the player can simply purchase the resource from the neighbours using coin resource. However, it is still worth to get to the resource even the player and its neighbour had the maximum number of resource because coin resource could potentially benefit the neighbour to win. Here is the evaluation function that is implemented to the AI. For my AI, the initial value for both resources is 2, resource own by player depends on the type of resource(0.5 for the brown and 1 for the grey) and resource own by both neighbour will be half of resource own by the player.

$$initialvalue - playerown * amount - neighboursown * amount \tag{4.1}$$

For the coin based reward, the multiplier can not be too high because owning too many coins does not necessarily make the player a winner. Otherwise, the AI will always discard the card to earn coins. For my AI, the initial value of each coin is 0.4 which it is not too low or too high.

$$Value = Coin * Amount \tag{4.2}$$

Apart from estimating the coin gain, The AI should also consider estimating the value lost from building card or wonder by spending coin resources to purchase resources. For my AI, the initial value for each coin trade to its neighbour is 0.5. The initial value cannot be too high otherwise the AI will refuse to purchase the resource from its neighbour.

$$Lost = Coinlost * Amount \tag{4.3}$$

For cheap trading, the value depends on the resource that the neighbour owns because cheap trading is pointless if the neighbour does not have any resource to trade. For my AI, The multiplier of the brown is 0.5 and the multiplier of the grey is 1.

$$Value = multiplier * amount\_of\_the\_resource \tag{4.4}$$

For the green value, the multiplier will only be used to increase the value of the first symbol for the green card to avoid the AI not picking the card. After the first symbol, the value of the green will be the actual value it provide. For my AI, the multiplier is 5 which the AI will choose a green card whenever it can build it.

$$Value = multipler * 1^2 \tag{4.5}$$

$$Value = newgreenscore - currentgreenscore \tag{4.6}$$

For the others such as the blue card set, red card set, and purple card set, the blue and purple will contribute direct points to the player. On the other hand, red does not contribute direct points to the player which I implemented the following equation to make it an option for the AI. The equation allows the AI to choose the red card even when it is the highest military city to make sure its neighbour will not outpower the AI.

$$Value = AgeVictoryToken/2 \tag{4.7}$$

$$Value = AgeVictoryToken \tag{4.8}$$

Chapter 5

Testing

5.1 Result

In order to test the performance of the AI, we must first gather the users game data (player vs player data). This is because we are trying to implement an Artificial Player which can compete against a human, therefore we need to understand the basics of how the match is played when three human players (or more) play the game, this means finding the average results from human players (after multiple games) and trying to get the AI to achieve similar results. The number of data sets I gathered was 12 total, this is not enough in total because each match lasts approximately 20 - 30 minutes and it takes a lot of time especially to gather players to play the game for that amount of time in order to get the exact results. After gathering the dataset of human player versus human player, I will also need to gather data sets in which the AI player will play against the AI player in order to form a comparison. I would also want to gather data set in which the human player will play against the AI player. However, due to the command line interface, it make it hard for the human player to understand or to follow.

5.2 Comparison

As shown in figure 5.1 the results gathered in which the averages are shown along with the average of the lower bound and the upper bound. In this figure, it clearly shows the averages of the heuristic state evaluation AI is outperformed the random choosing AI in all aspect which is what we should be expected. The Average Point and lower bound of a Heuristic state evaluation AI is slightly lower compared to the Human player. However, it does not mean that the AI will not outperform the Human player. As the game could only have one winner which suggests that it is more important to consider the upper bound when it comes to comparing performance. By comparing the upper bound, we can conduct that the AI is scoring much higher points compared to the Human player, therefore, in theory, the AI should be able to provide a good challenge to the human AI.

	Average Point	Average of the lower bound	Average of the upper bound
Human	46.2	38.5	53.8
Heuristic state evaluation AI	45.5	32.6	54.7
Random Choosing AI	24.3	15	35.9

Figure 5.1: Scoring table

Finally to wrap up the comparison, figure 5.2 shows that the pattern of the Heuristic AI and the Human, they follow a similar wave of gameplay, however, due to the limited amount of data from humans playing I was not able to conduct the full final results in order to make it so that the Heuristic and Human player data were more or less identical, this still, however, had shown that the Heuristic AI was far better than the Random choosing AI when it came consistencies matching a Human dataset.

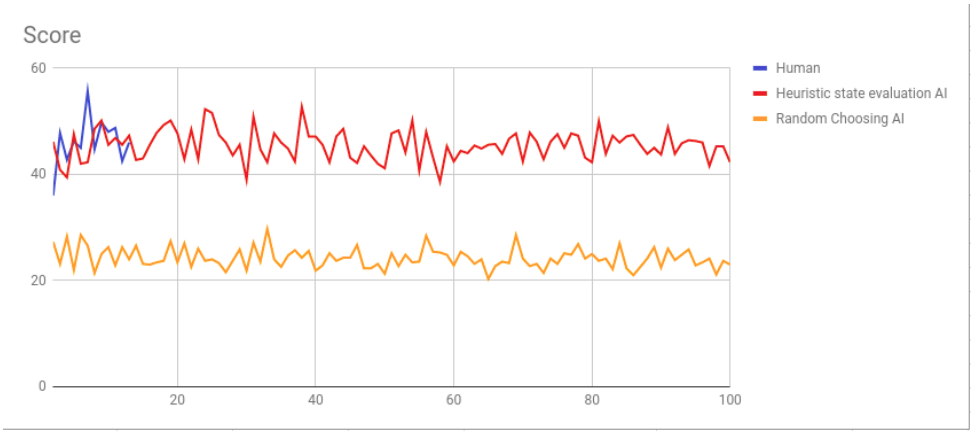

Figure 5.2: Average score in 100 game

5.3 Evaluation

To finally evaluate my board game AI, I had concluded once the implementation of the game is complete, this meant that the AI had been fully working against a human player with limited to no bugs. Due to the time constraints and the issue of having no human players to test the game against the AI, I was not able to fully complete that side of the evaluation, however going up on the datasets provided I can conclude that the Heuristic state evaluation AI would be good enough to provide a challenge to the human player. The limitations of the Heuristic state AI would be the initial value for the resources would be too low, this would mean that the AI would prefer choosing cards that provide direct value rather than building on the resources which would be needed for planning in the later stages. Another factor would be that the green cards initial values is too high which means that the AI may focus on only getting the green card, which does not actually contribute well enough in the end for the AIs' score. Beside the estimate of value, the main factor of limitation would be the fact that the AI only consider what good for the current state instead of what could be good for the later state.

Chapter 6

Conclusion

6.1 Revisiting the Aim and Objective

After revisiting my aim and objectives, the aims which are to create an artificial player which could provide a challenge to the player and increase my experience in both learning and develop game and AI. The aims had been achieved along with the objectives as I have concluded to finish creating the AI as well as learning much more than what I had initially known in the area of Artificial Intelligence. The objectives were well managed and proceeded to be completed just in the order I had originally planned them, making the project overall a success.

6.1.1 Objective Comparison

1. Produce a command line version of the game 7 Wonders
2. Pursue and document my background research into the implementation of AI in board games
3. Develop and Artificial player capable to play the game
4. Test the AI against a human player
5. Evaluate the performance of the AI

Objective 1 was achieved as shown in chapter 3. The command line version of the game 7 Wonders is implemented in python and it allows the user to play with up to 6 AIs. The game being implemented was close to the original version of the game in which the modification was mention in section 3.2.3.

Objective 2 was achieved as shown in chapter 2 in which I have provided research on different artificial intelligence algorithms that could be implemented. Apart from AI algorithms, I also provided research on the game theory which could help to justify which algorithms could potentially perform well in 7 Wonders.

Objective 3 was achieved as shown in chapter 4 in which I have provided the classification of the game 7 Wonders and also what approach have I planned to follow. Apart classification and approach, I have also include the implementation of the AI.

Objective 4 was achieved as shown in chapter 5. However, the testing is different compare to the original in which I am not able to test the AI against human player. In exchange, I have collected and compare the datasets for both human play against human and AI play against AI.

Objective 5 was achieved as shown in chapter 5 section 5.3 in which I have evaluated the heuristic AI based on the dataset I have collected for the testing. Apart from the dataset, I have also evaluated the AI based on the observation of the game's process.

6.2 Personal Reflection

Looking back upon the project, there are lots of accomplishments which have stood out, such as creating a game entirely for the first time myself, and also having created an artificial player which can provide a good challenge to the human player, these achievements will enhance my understanding along with the experience I go through whilst having accomplished these on my view of AI and the game development.

Looking back upon the difficulties encountered by me in the project and the things I would definitely improve upon would be as followed:

- Stick with the time-line - Go through and follow the initial planning time-line strictly which will allow me to be more organised with the project side of development and also keeping track the process, as well as finishing things in a sequence will help me to gather the necessary objectives in order. In this project I have failed to stick with the time-line as it has led me off course, therefore causing me to fall behind on the time-lines objectives. This is because when I planned the time-line, I was not considering the workload on other coursework as I was doing a 40-40 split on the credits of modules in which I should have done more on the first semester.
- Be more organised - Being more organised when it comes to the planning of the project, this would help as I have personally not been able to plan properly through the project, making a case or certain errors which occurred to me whilst I was working on the project, knowing later I had forgotten to do something in which it results on me to rewrite the game in the middle of the development phase.
- Improve Coding Style - I would improve upon the workings of my coding style by making my code more easier to read and understand by not making it so over complicated and mess for the others and also myself. The reason that some of the code would potentially be reused by other functions which I implement but due to bad planning on structure, I have to create a separate function for both.

6.3 Further Work

Further work I would expand upon in the future or would have done if the time constraints had been non-existent include:

- Graphical User Interface - This is because the most of game in the modern day, heavily rely on graphics as they allow more interactivity when it comes to the user enjoying the game as well as making better visuals helps the perception of the user to fully comprehend the game in a matter where nothing is left to imagination. Same as 7 wonders in real life, we see the interactivity and the art of the cards come to life when playing with other players on the visual board game, this brings a whole new aspect to keeping track of your cards as well as providing a perception where you can view all moves along with actions done correctly. The current command line interface is difficult to understand and does not keep the user motivated to keep playing, as it does not provide any eye catchy visuals neither does it provide any sustenance to replay-ability when it comes to the game.
- Improve upon gameplay modification - Certain aspects would have to be changed, these would begin with the modifications I have previously done to the game, such as sequential gameplay where the game is played out sequentially, this would have to be reverted to its original state in which the proper board game is played out in a simultaneous sequence where the players all process there moves at the same time.
- Implement different type of AI - I would begin to implement different types of AI into the game such as, AI such as Minimax which proceeds to look further down stages where it can identify for the most optimal choice and proceed to choose that path, perhaps using machine learning as well to enhance my personal experience on AI learning mechanisms and methodology.
- Proper Testing - As I have mentioned in section 5.2 in which I am not able to carry out testing with human player play against AI players due to the command line interface. This allows me to properly evaluate the performance of the heuristic AI in order to improve upon it.

References

[1] Alphago zero: Google deepmind supercomputer learns 3,000 years of human knowledge in 40 days. https://www.telegraph.co.uk/science/2017/10/18/alphago-zero-google-deepmind-supercomputer-learns-3000-years/.

[2] Board game geek. https://boardgamegeek.com/boardgame/68448/7-wonders.

[3] Deep blue. http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/. Accessed: 2018-01-26.

[4] Introduction to cooperative game theory. http://www.uib.cat/depart/deeweb/pdi/hdeelbm0/arxius_decisions_and_games/Cooperative_game_theory.pdf.

[5] Openai. https://blog.openai.com/dota-2/.

[6] Video explain dota and the implementation of ai — https://www.youtube.com/watch?v=l92J1UvHf6M, note = Accessed: 2018-01-25.

[7] Y. Bassil. A Simulation Model for the Waterfall Software Development Life Cycle. *ArXiv e-prints*, 2012.

[8] G. Bonanno. Game Theory (Open Access textbook with 165 solved exercises). *ArXiv e-prints*, 2015.

[9] M. Buro. Improving heuristic mini-max search by supervised learning. 134.

[10] D. Diderot. Philosophical thoughts. 1777.

[11] D.Robilliard, C.Fonlupt, and F.Teytaud. Monte-carlo tree search for the game of "7 wonders". 134.

[12] J.Schaffer, J.Culberson, N.Treleoar, B.Knight, P.Lu, and D.Szafron. Artificial intelligence - checker. 53.

[13] L. KocÂÿkesen and E. A. Ok. An introduction to game theory. pages 5–19, 2007.

[14] S. Kuhn. Prisoner's dilemma. 2017.

[15] M.Campbell, A. Jr., and F.Hsu. Artificial intelligence - deep blue. 134.

[16] A. L. Samuel. some studies in machine learning using the game of checkers. ii — recent progress. pages 601–617, 1967.

[17] A. M. Turing. Computing machinery and intelligence. 59(236):433–460, 1950.

Appendices

Appendix A

External Material

- Atom — Atom was used to develop the game and AI
- Pycharm — Pycharm was used to debug the game and AI
- Gitlab — Gitlab was used to version control the game and AI which my code will be available in `https://gitlab.com/Sc15rmdc/final-year-project`
- Google sheet — Google sheet was used to create the Gantt Chart
- Google docs — Before I was introduced to overleaf, google docs was used to write my report
- Overleaf — Overleaf was used to write my report

Appendix B

Ethical Issues Addressed

During the testing phase of my project, I had invited 9 participants to play the physical board game 7 Wonders, in which the result of the game will be collected for the purpose of testing the AI's performance. All participants were provided with project information sheet to understand the purpose of the data collection and were required to sign a consent form. Those data will be published anonymously within the report in the testing chapter and those forms will be submitted in an envelope along with the final report.