University of Nottingham
School of Computer Science

Game Development -
Rithmomachia with Artificial Intelligence

December 6, 2019

Submitted by:
Student ID: 4308281
Athullya Roy
School of Computer Science
psyar7@nottingham.ac.uk

Supervised by:
Venanzio Capretta
School of Computer Science
venanzio.capretta@nottingham.ac.uk

Table of Contents

1. Introduction and Motivation

1.1 A brief history

From the conception of board games in the early dynastic period till today, board games have been played and enjoyed by individuals of all race and nature. From royalty to deprived masonry workers, board games have been significant to both education and recreation throughout history. Especially, due to the increased use of the internet and a recent surge in the gaming industry, more people are becoming aware and investing their time into playing board games. This is reflected on the fact that the global board games market is predicted to rise over $12 billon by the year 2023. [1] Observing this recent growth in the industry, I decided to base the project around the development of such a game – rithmomachia.

Rithmomachia, also known as "The Philosopher's Game" or "The Battle of Numbers", is a long forgotten mathematical board game which at its peak was once a strong rival to Chess. The game was once popularised by students and academia for its representation of Boethian mathematical philosophy and its need for quick mental arithmetic. However, its popularity has been fluctuating since its formation in medieval Europe, where it had a life span of over 500 years. [2] The game and its rules has been rediscovered by historians by analysing the handful of medieval literature which discussed the game. However, different versions of the descriptions exists as different authors documented it contrarily. Therefore, the rules and structure stated below purely indicates the version which I am interested in developing.

1.2 Rules and Structure – Basic

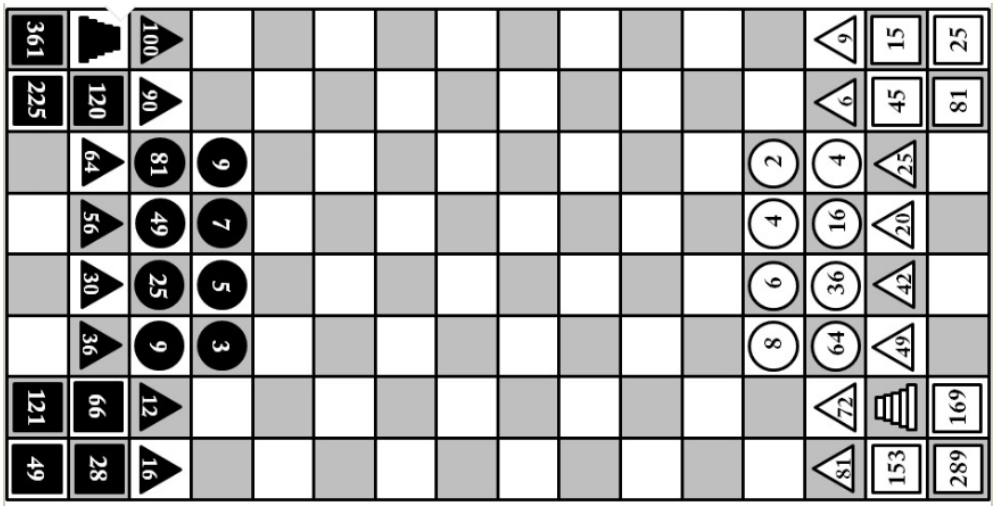

Figure 1: Rithmomachia Board - Initial State [3]

A sixteen by eight checkered board is required to play the game and the two players are referred to as black (odd) and white (even). Unlike traditional board games, in rithmomachia information is asymmetrical. Players are given unique sets of pieces which consist of the shapes - squares, circles and triangles. One square on each side is replaced by a pyramid – which is formed by stacking certain pieces together. All pieces contain inscribed unique numbers. Similar to chess, the different shapes have unique directions they can move in. [4]

- *Squares*: Allowed to move four steps horizontally, vertically or diagonally
- *Circles*: Allowed to move one step horizontally, vertically or diagonally
- *Triangles*: Allowed to move three spaces horizontally, vertically or diagonally
- *Pyramids*: Follow the rules of the pieces it consists of.

In rithmomachia, the players have to capture the opponent's pieces, in one of the four following ways.

- *By Meeting*: If a piece lands on a position occupied by the opponents piece and they both have the same value then the opponent's piece is removed. However, the player's position does not change.
- *By Assault*: If a piece with a small value and a piece with a larger value are aligned unobstructed on the board, and the smaller value multiplied by the number of empty squares between them is equal to the larger piece, the larger piece is captured.
- *By Ambush*: If a piece is positioned in the middle of 2 opponents pieces, and the sum of the two pieces equal to the piece in the middle, then it is captured.
- *By Siege*: If a piece is surrounded by opponent pieces on all four sides, then the piece is captured.

The game consists of two forms of victories – common and glorious. A common victory occurs when a certain number of pieces (decided by players beforehand) is captured by either parties. On the other hand, glorious victories occur when 3 pieces from a player reaches the opponents side of the board and they are arranged in either an arithmetic, geometric or harmonic progression. The shapes has to form an unobstructed line (horizontal or vertical) or a triangle on the board.

1.3 Rules and Structure – Mathematical Perspective

All the numbers on the boards are derived from each other through complex mathematical relations. For example, the second row of triangles are derived from the first by summing the value of the shape above (if first row, then the shape beside it) and adding a fraction of the number on the first row. For example, 81 (white triangle) is derived by adding 72 with $1/8^{th}$ of 72. Similarly, 42 (white triangle) is derived using the formula 36 + (1/6 * 36). [5]

The pyramids have a unique formation of numbers – white pyramids include 36, 25, 16, 9, 4 and 1 (sum = 91), whereas the black pyramids consist of 64, 49, 36, 25, 16 (sum = 190). White pyramids are known as a perfect pyramid because 91 is the sum of first 6 squares, whereas back pyramids are imperfect because $190 = 8^2 + 7^2 + 6^2 + 5^2 + 4^2$. Likewise, the numbers on all the shapes and rows follow strict mathematical rules.

1.4 History of Artificial Intelligence in Board Games

Artificial intelligence players for board games are quite famous and have been very successful in development. The first significant step towards this is described in C. E. Shannon's publication in 1950, where he introduced the theory of the minimax algorithm.[6] Even though this was only a mere idea at the time, the algorithm is still hugely relevant in the field of artificial intelligence to this day. This was first implemented in a game of checkers, developed by A. Samuel, which is the first game developed with elements of artificial intelligence. [7]

Since then, there has been many developments in the field of AI in board games. DeepMind's AlphaGo Zero is an AI developed to play the Chinese game of Go. By the year 2017, AlphaGo became the best Go player in the world by beating the world champion in all 3 games played. [8] AlphaGo learned using reinforcement learning – a neural network learned by playing against itself till it couldn't be defeated in the game. As it plays, the neural network is tuned and parameters updated so that it can better predict the opponents moves, hence has a higher chance of winning each time.

1.5 Motivation

There is an ever-growing demand of board games in the online market. With more players engaging with online board games, it was deemed that it would be appropriate to create an online version of rithmomachia. The most significant cause for developing this project is the originality of the task. From research, I have found that there are no existing versions of the game on the internet; i.e. it has never been implemented before. This could be due to the lack of popularity of the game or simply because of its complexity. The originality of the project motivated me to conduct further research on the history and origins of the game. I was fascinated by how this was once widely popular but in today's time even board game enthusiasts remain unaware of it.

Even though this European mathematical game has comparisons to chess, as both are board games enjoyed by intellectuals, the characteristics of rithmomachia are unique and distinguishes it from the rest. The main feature of rithmomachia which differentiates it from other board games that are available to be played online, is its structural foundations in mathematics. For example, winning is not as easy as capturing pieces in rithmomachia – it involves sufficient intelligence; which is why it was once popular among students. This is reflected in the games Glorious Victory, where the player has to tactically move a piece to the opponent's side of the board and then form an arithmetic, geometric or harmonic progression. These progressions are also combined for the user to attain higher levels of victories. This influence from the Pythagoreans Greek mathematics and its unique set of game rules has motivated me further to complete this project.

In addition to this, an incentive behind this project was to add a new perspective to an ancient game by creating an artificial intelligent component to it. From the establishment of the term "Artificial Intelligence", board games has been at the forefront of its domain of study. This is because games are hard and interesting problems which requires some sort of intelligence to solve. As well as this, since this game does not exist in an online platform yet, the challenge to find solutions to adapt existing Artificial Intelligence algorithms was motivating.

2. Related Work

Due to the lack of popularity of rithmomachia, the game is yet not available to be played online. However, AI development for other board game such as Chess and Go have been very successful in the past. Therefore, I have researched into existing chess engines; understanding the algorithms followed can then influence how I approach building the AI for rithmomachia.

2.1 StockFish

StockFish is one of the strongest chess engines currently available. It begins by evaluating the current board position by developing a legal-move tree, hence generating an evaluation function. This evaluates how good the current positions of the pieces on the board are. When evaluating the position, StockFish considers several human-programmed rules such as checking if the game is close to the end. Endgame weights differently to midgame as deeper searches would not need to be conducted, as winning/loosing is more close to hand. It also analyses positions to see if there are any trapped pieces that will be captured if they move or if any pieces are not protected – this should have a higher priority in the next move.

To find the best possible position, StockFish follows a search algorithm which is summarised here. Firstly, it analyses the current position and a set of legal moves which it could take - this produces the possible positions. It then analyses the possible positions, with the past positions and weakness in

opponent's position to identify the strength of the position. This results in the best possible solution being generated.

StockFish generates a transposition table which is a hash table that stores the previously performed searches. This helps reduce the search space of the tree, hence making it more efficient. It uses a modified version of minimax for searching, and then uses a variation of alpha beta pruning to make the searches more efficient. While creating rithmomachia, this will influence the project as I will attempt to implement a transposition table so that the same path isn't searched multiple times. [9]

In addition to this, StockFish also uses different heuristics depending on whether the game is in the beginning or if it is midway. At the beginning of the game it is programmed so that it develops minor pieces such as knights and bishops as well as trying to spread out and control the middle of the board. However, as the game progresses, StockFish focuses more on the opponent and tries to take initiative by attacking the white and setting "traps" which encourages the opponents to make bad moves.

2.2 AlphaZero

AlphaZero originates from DeepBlue's AlphaGo, and it has been declared the worldwide strongest AI-performance, when it beat StockFish in a 100-game match. In the match Alpha-Go did not lose once, instead it won the game 28 times and drew the rest of the times. The main differences between both relies on the algorithms adapted by both; for example, where StockFish uses an alpha-beta search engine, AlphaZero opts to use a Monte-Carlo Tree Search (MCTS). [10]

One of the main advantages of AlphaZero over StockFish is that it replaces the handcrafted knowledge-based minimax search in StockFish with a deep neural network and a general-purpose tree search algorithm. StockFish relies on a handcrafted minimax algorithm, which was designed with 1000's of heuristics implemented by chess champions, whereas AlphaZero learns using a deep neural network. From this neural network, AlphaZero has taught itself by playing many games against itself the rules of Chess, Go and Shogi. [11] For chess, the neural networks takes the board position as an input and produces a vector of move probabilities for each action as an output. AlphaZero learns these move probabilities and from self-play evaluates the usefulness of the move probabilities. It then uses this knowledge to improve the moves in future games.

2.3 Reflections

As rithmomachia is not available on the online platform and an artificial intelligent engine has not been designed for it, my project will significantly vary from both these existing chess engines. However, since I am designing an artificial intelligence game opponent, I have drawn influences from these works. For example, I will initially try and implement a minimax algorithm as done by StockFish and when that successfully works, try and implement a MCTS as implemented by AlphaZero. However, despite AlphaZero, I will not implement the Machine Learning component by creating a deep convolutional residual neural network. This is because of the limited timespan and this might not be feasible due to the requirements needed to implement this.

3. Description of the Work

3.1. Aims and Objectives

The aim of this project is to develop a game of rithmomachia and to produce an AI to play as the user's opponent. The core objectives which determines the success of the project are:

a) *Through extensive research, gain an understanding of the various approaches and algorithms used to develop AI for board games.*
   Research into min-max algorithm, alpha-beta pruning, Monte-Carlo searching, machine learning etc. This research should provide sufficient information to decide which implementations will be the most feasible.

b) *Develop a game of rithmomachia which implements the rules and multiple forms of victories stated above.*
   Research into the libraries that can aid the game development and implement the game so it is authentic to the traditional board game.

c) *Design and develop a graphics interface for the game.*
   Ensure that the representation of the game is easy and simple for the user's to understand.

d) *Enable two human players to compete against each other.*
   This should be indication that the first part of game development is complete – The next focus should be developing the AI for the game.

e) *Develop an AI to play against a human player.*
   Start by designing a minimax algorithm. Continue developing better AI so that there is a higher chance of the computer winning the game.

f) *Conduct the end user testing thoroughly*
   Ensure that the user testing is done thoroughly so that statistics can be produced to understand how likely the AI opponent is to win a game.

Even though, this is a software development based project, functional requirements are not necessary as the main stages are identified above. However, I have further expanded the game development stage here to indicate a few of the requirements; this is a non-exhaustive list, to allow flexibility while working:

1. The system should start the game with the initial board layout shown in Figure 1.
2. The game pieces should follow appropriate game rules
3. Illegal moves should be prevented.
4. When pieces are captured, they should be displayed to the user as an indication of progress of the game.
5. The system should have a user-friendly graphics interface.
6. The system should implement common victory conditions.
7. The system should implement glorious victory conditions.
8. A method should be implemented to keep track of the scores of both players.
9. The AI should evaluate possible moves and have an appropriate move generation algorithm.
10. System should use the minimax algorithm to evaluate the AI opponents moves
11. System should use alpha-beta pruning to make the AI more efficient.
12. System should implement the Monte-Carlo Tree Search for the AI opponent's move generation.

I have ensured that all these requirements will be met by the project deadline by developing a project plan, where the tasks are divided into sub-sections and each section has an internal deadline.

4. Methodology

After conducting extensive research, I have concluded that the programming component of this project can be broken down to three significant parts – developing a text-based game that fully works, developing the graphics for the game and developing the AI for the game.

4.1 Developing Rithmomachy

StockFish stores the chessboard in bitboards where each square is represented by 1 or 0 indicating whether a piece is present or not. It stores each move in 16 bits where bits 0 to 5 stores the destination square (from 0 to 63), bits 6 to 11 stores the origin square (from 0 to 63), bits 12 to 13 stores the promotion piece type, and the final bits store special move indicators (i.e. promotion etc).[12] However, due to the more complicated shape structures in rithmomachia, where more information has to be held regarding the pieces and as multiple pieces can be build up to form other shapes, I opted out of using bitboards. Instead to store the board and the move information, I have created matrixes.

4.2 Implementing an Artificial Intelligence Opponent

There are many different approaches and algorithms that could be used to develop the AI component. From researching different algorithms, I have come across the following which seems to be feasible and applicable to this problem.

4.2.1 MiniMax and Alpha-Beta Pruning

Minimax is a backtracking algorithm which recursively searches through a game tree to find the optimal move for a player, assuming that the opponent's moves are always optimal as well. An evaluation function can be developed to assess the current game state and the possible end states to deem which player is likely to win with each move. Based on these values returned from the function, the minimax algorithm can be applied by assuming that at each node, the player whose turn it is chooses the value which maximises their chance of winning, and the opponent would choose the value which minimises the player's chance.

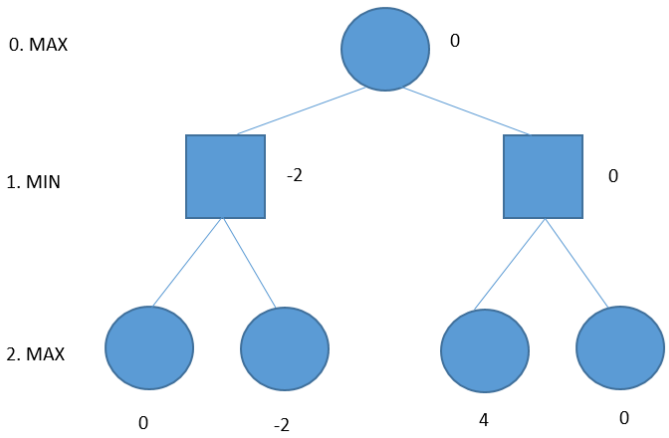

Figure 2: Minimax Algorithm

In the diagram shown, the root node symbolises the move the current player has made. Level 1 indicates the opponent's possible moves; it has been reduced to two nodes for simplicity. The next level is the current player's turn where they will either win the game (evaluation = 4) or draw the game (evaluation = 0). Minimax performs a depth-first search where it aims at searching down the levels before searching horizontally.

However, the main drawback of the minimax algorithm is that since it uses uninformed searching (depth-first search), the search tree will grow rapidly with each move the algorithm analyses; hence, it is computationally infeasible. This would affect storage space and will decrease the speed of the program significantly. However, a solution to this issue lies in alpha-beta pruning, where certain

branches are pruned (not evaluated) to prevent wasting computation time or storage space. StockFish uses a minimax algorithm with modified alpha-beta pruning.

4.2.2 Monte-Carlo Tree Search

A more advanced machine learning algorithm is the Monte-Carlo Tree Search. MCTS is a heuristic reinforcement learning algorithm which attempts to find the optimal move by only exploring certain nodes in the search tree. It consists of 4 stages – selection, expansion, simulation and backpropagation. The selection stage uses an evaluation function to choose the child nodes which has a higher chance of winning. When a node has been selected, the MCTS moves onto the expansion stage, where child nodes of the selected node is added to the search tree. In the simulation stage, a random playout is carried out from the selected child node till the game is completed. Finally, the result of the playout is used to update information on all the nodes that has been visited so far – this is the backpropagation stage. [13]

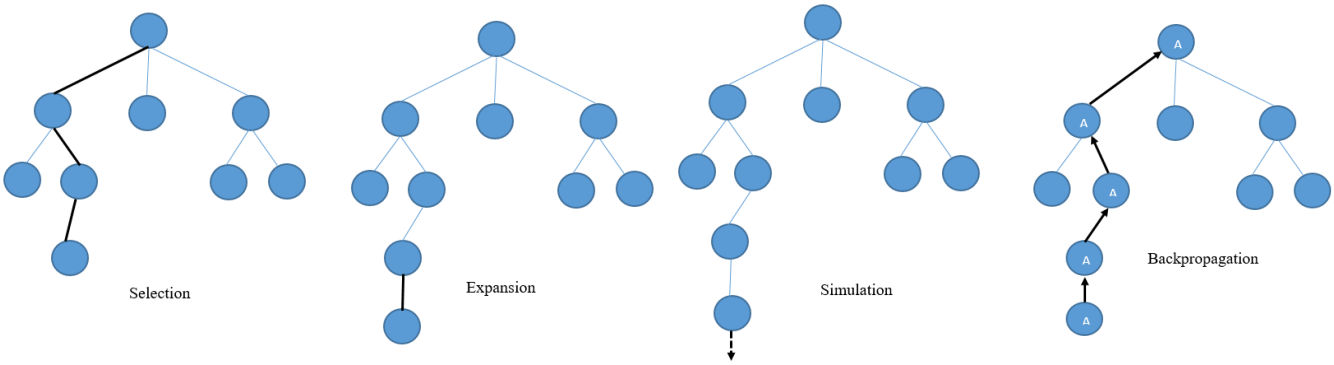

Figure 3: Monte-Carlo Tree Search

DeepMind's AlphaZero uses MCTS to evaluate the possible moves and choose the optimal one. The MCTS stores four statistics in each edge of the tree, N, W, Q and P, where N is the number of times an action (edge) has been taken from the state (node), W is the total value of the next state, Q is the mean value of the next state (i.e W/N), and P is the prior probability of selecting an action derived from the neural network. [14]

AlphaZero then carries out the four stages of the Monte Carlo Tree Search. It selects by choosing the action that maximises Q+U, where U is a function of P and N, which increases when the branch hasn't been greatly explored. AlphaZero relies on performing more exploration towards the beginning of the game and more exploitation as the game progresses. In the next step, the tree is expanded until a leaf node is reached. The game state of the leaf is then passed to the neural network, which make predictions about the move probabilities and the value of the state for the current player. After this, AlphaZero backs up the previous edges that were traversed to get to the leaf. Finally, it selects a move, either deterministically, for competitive play, or stochastically, for training. The selected move then becomes the new root node and unwanted nodes from the tree are discarded. This process is repeated by AlphaZero till the game is completed. [15]

5. Design

The problem which the project addresses is to develop a game of Rithmomachy and create an artificial intelligent opponent for the game, which has a high chance of winning any given game. To address the issue, the solution which was concluded on after conducting research was to implement the game using Python and develop the graphics using Pygame. After this, the AI feature will initially be developed using minimax. Due to the lack of efficiency of the algorithm, this would not guarantee on winning most of the matches played. I will then improve the algorithm by implementing alpha-beta pruning; this increases the efficiency of the AI developed. Finally, I would implement the Monte-Carlo Tree Search, which is a strong AI algorithm, hence increases the probability of the games played being won.

I have decided to start building from minimax and then when confident with the basic AI, build the system to use alpha beta pruning and then Monte-Carlo. This will ensure that progress is consistently made with the project, alongside confirming that the game implemented works logically. As well as this, when both algorithms are implemented, they could both be utilised in the final product as different game levels; i.e. give the user the option to play with the "stronger" or "weaker" AI opponent. This will increase the usability of the end software.

The figure shows a simple design of the user interface of the game. Pieces captured are visible to the user so they are aware of the progress of the game. The green squares indicates the possible moves the piece selected can make; since the game has a complicated set of rules, this should aid the player in considering all their choices hence making the game easier for them. The bar at the bottom indicates how strong the moves made are. This was inspired from chess.com [16], which is a famous online chess platform. A help button is also provided which will display the instructions to the user.

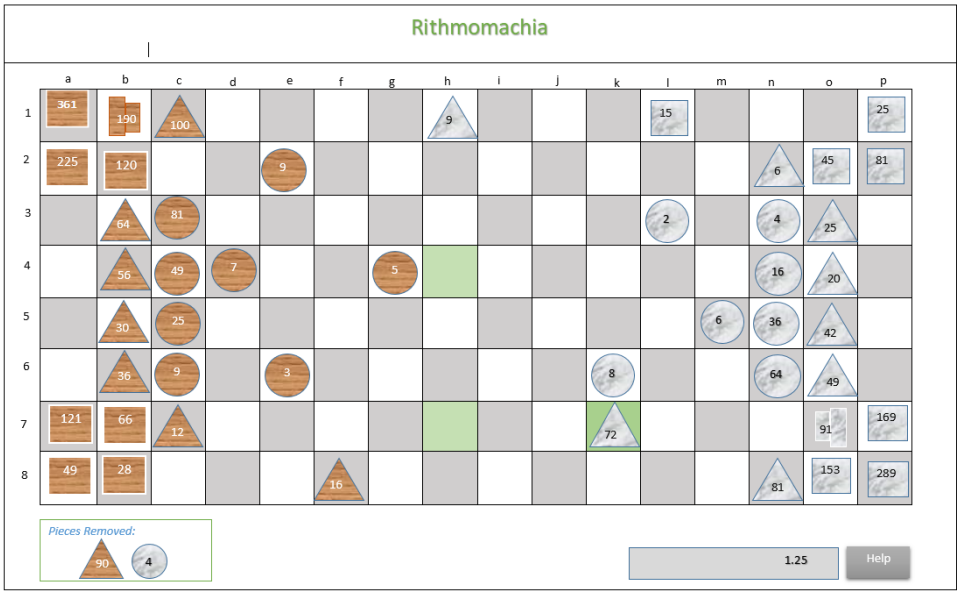

Figure 4: Design of the user interface

6. Implementation

The game will be implemented in Python 3.7 using the IDE PyCharm. Various approaches could be taken to develop the graphics for the game. The most basic would be to use Pygame, which is an open-source Python programming library build on top of SDL and is used for graphics development. Another approach would be to use a game engine, such as OpenGL. If this is the case, I will have to install PyOpenGL, which is a Python OpenGL binding. The main benefits of using Pygame is that it

is easy to learn and there is lots of support and tutorials available online. As well as that, Pygame should be sufficient for a 2d game development. Therefore, I have decided to use Pygame over game engines as it will be easier and more convenient for this scenario.

I will also need to use libraries such as NumPy and math to calculate formulas for functions such as the evaluation function. A matrix will be implemented to store the board. As well as this, a transposition table will be implemented to store results of previous searches in the form of a hash table. All code will be uploaded to a Gitlab repository.

7. Progress

7.1 Initial Plan

Component	ID	Task	Progress
Planning	1	Propose a project plan to supervisor	100%
Documentation	2	Complete the ethics checklist	100%
C	3	Research into existing AI for games	100%
A	4	Develop a text-based board, players and pieces	100%
A	5	Program the enforcement of game rules	50%
Documentation	6	Write the Interim Report for dissertation (deadline 6th December)	100%
A	7	Implement the winning conditions – common and glorious victories	N/A
B	8	Develop the graphics for the game	N/A
C	9	Further research into AI and algorithms	N/A
C	10	Develop the MiniMax algorithm	N/A
C	11	Develop Alpha-Beta Pruning	N/A
C	12	Develop Monte-Carlo Searching	N/A
C	13	Improve the AI further	N/A
Testing	14	Test the program	N/A
Documentation	15	Write final dissertation (deadline 20th April)	N/A

The project was designed with the waterfall software methodology in mind. The programming component of this project can be broken down to three significant parts – (A) developing a text-based game that works, (B) developing the graphics for the game and (C) developing the AI for the game. These are colour coded in the Gantt chart and shown in the table above.

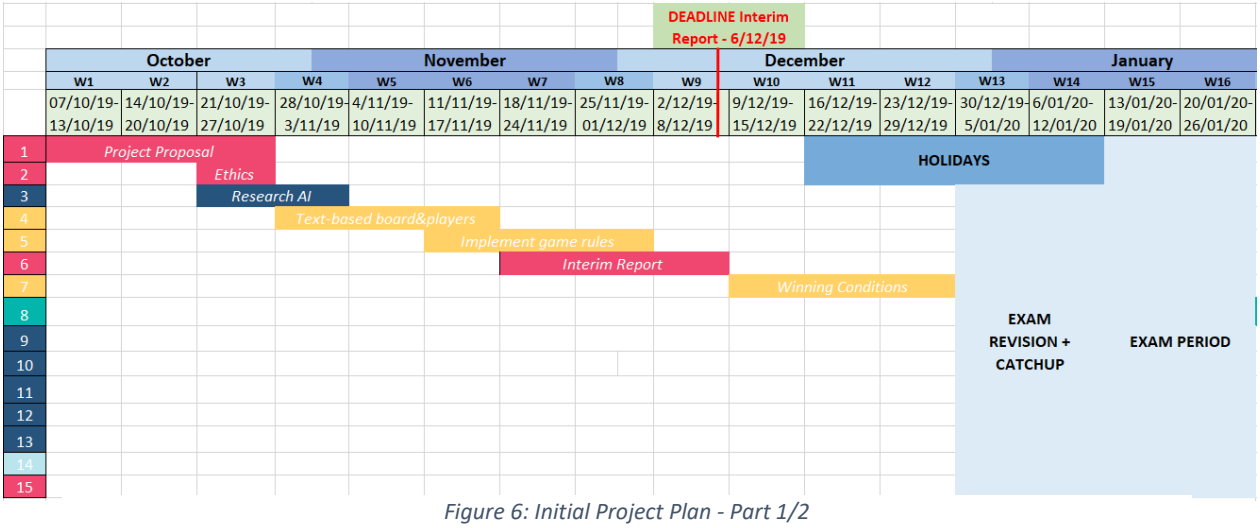

Figure 6: Initial Project Plan - Part 1/2

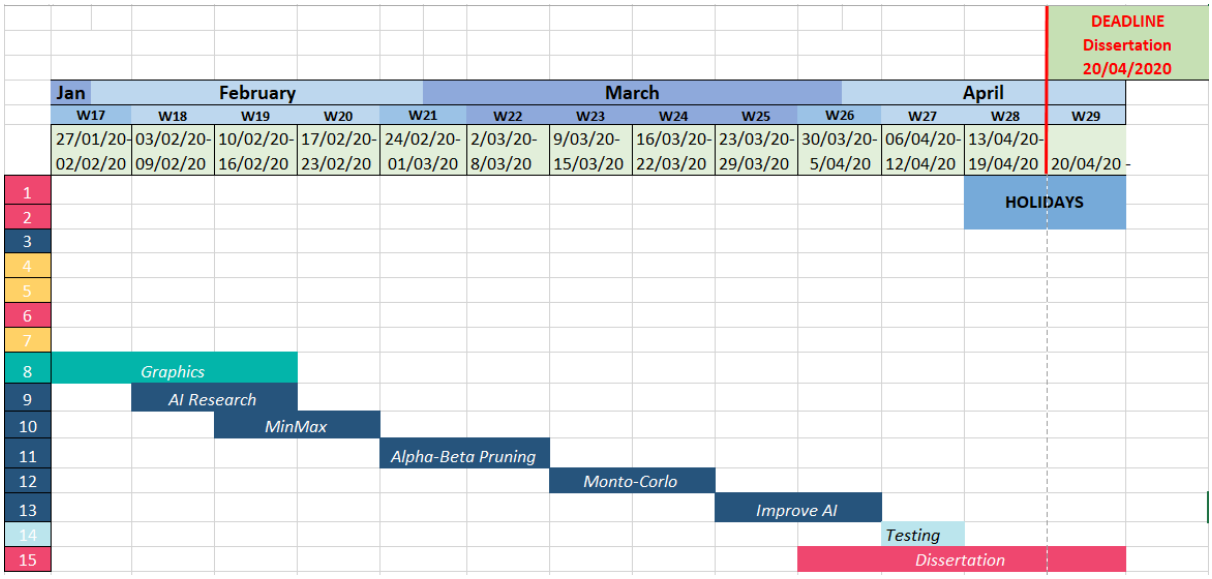

Figure 7: Initial Project Plan - Part 2/2

7.2 Modifications

As shown in the Gantt chart, more time was taken than planned for in a few of the tasks. Currently, I need to finish implementing the game rules, but I have extended the deadline for that till week 12. The main reason for having to extend the time needed for the tasks was because when planning, I should have been considerate that I would have more work this semester, compared to the next, as my workload is split as 70 credits in semester 1 and 50 credits in semester 2. While planning, this should have been taken into consideration and more time should have been assigned per task this semester. In addition to this, I have fallen behind on the workload and required additional time because of coursework and other workload taking up a significant amount of time. While planning, I should have allowed some time for catch up, where I can simplify focus on catching up on tasks necessary. This is taken into consideration while updating the Gantt chart, and a week has been allowed for catch-up.

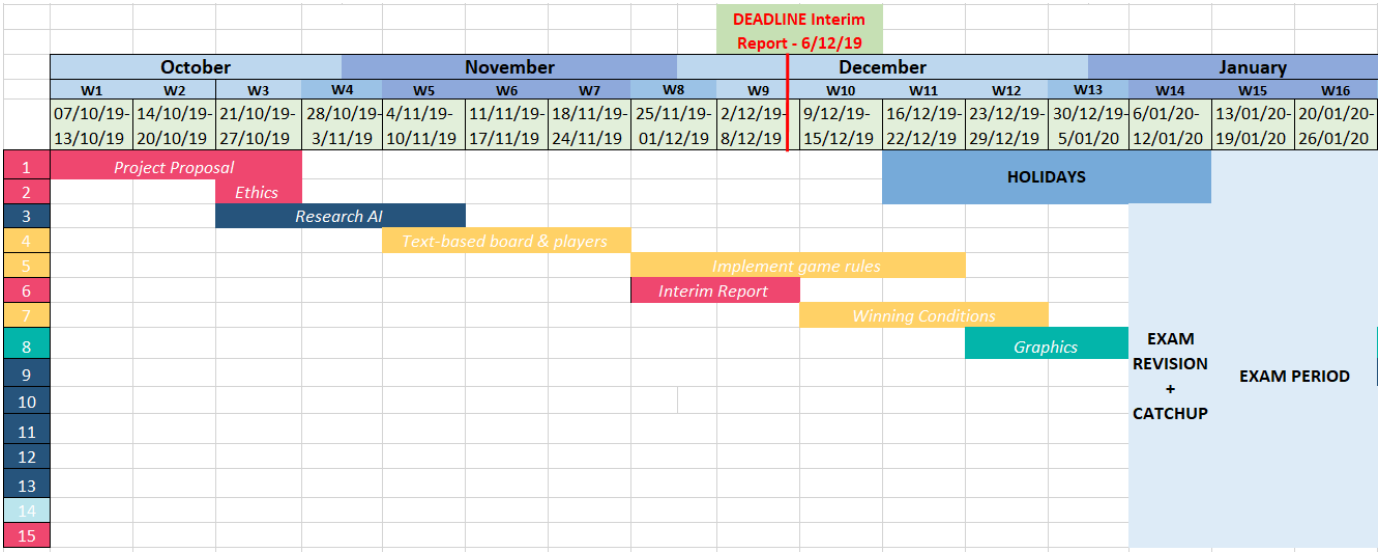

Figure 8: Updated Project Plan - Part 1/2

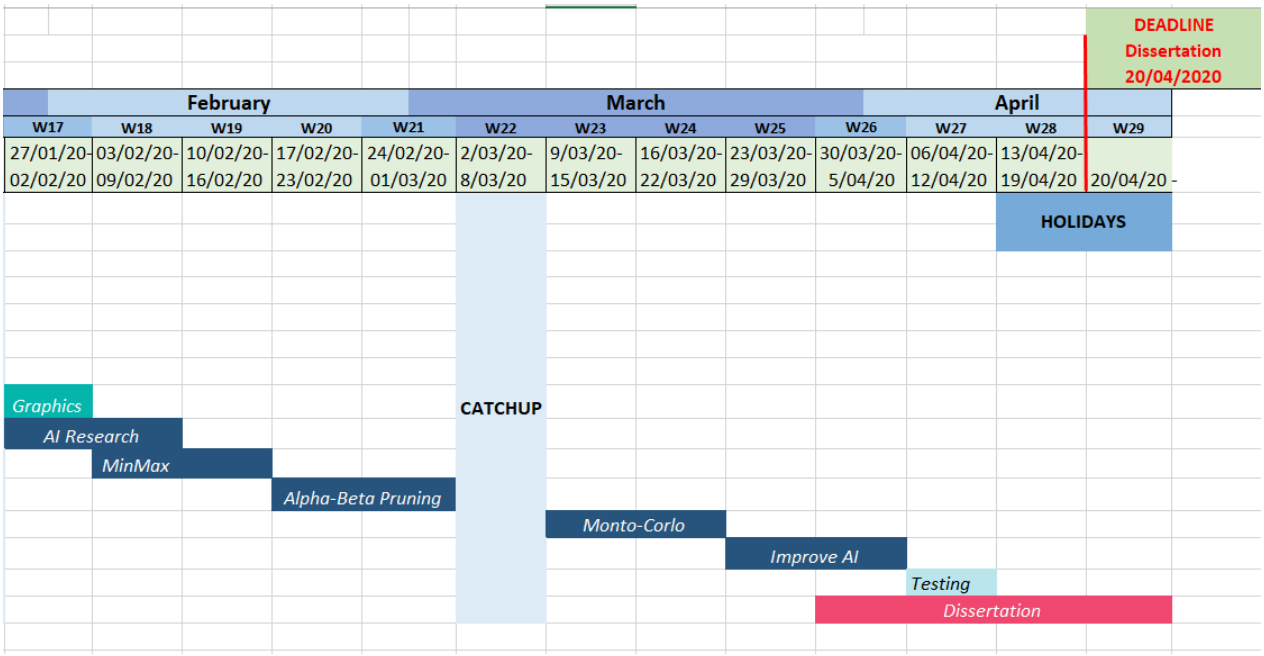

Figure 9: Updated Project Plan - Part 2/2

7.3 Contributions and Reflections

Since October, I have been consistently working on the project. However, because I did not take into consideration that I would have less time this semester compared to the next as I am doing 70 credits this semester and only 50 next, I am behind by 1 task. This was also partly due to me not considering other work load such as having to attend interviews and preparing job applications. Despite this, I think I have made significant progress, both in relation to conducting relevant research as well as developing the software.

To prevent this from happening next semester, I will ensure that I will commit more hours to completing the dissertation next semester. In addition to this, I will hold meetings more regularly so that this work load will be equally priorities as the rest of the coursework.

8. Bibliography

1) Wood, L. (2018) 'Global Board Games Market Outlook and Forecast 2018-2023: Major Players are Asmodee Editions, Hasbro, Mattel & Ravensburger', PR Newswire, 07 August. Available at: https://www.prnewswire.com/news-releases/global-board-games-market-outlook-and-forecast2018-2023-major-players-are-asmodee-editions-hasbro-mattel--ravensburger-300693174.html
2) Moyer, A. E. (2001) 'The Philosophers' Game, Rithmomachia in Medieval and Renaissance Europe', United States of America: University of Michigan.
3) Tomas, H.M and Borquez. D, (2015) 'Breve Historia de los Juegos de Mesa', Ludoteca de Pampala Press. Available at: https://ludotecapampala.wordpress.com/2016/06/01/rithmomachia-bhjm-11/
4) Newton, D. P. (1984) 'Rithmomachia', *Mathematics in School*, Vol 13, no. 2, The Mathematical Association.
5) Smith, D.E and Eaton, C.C, (1911) 'Rithmomachia, the Great Medieval Number Game', *The American Mathematical Monthly*, Vol 18, No. 4, Taylor & Francis, Ltd
6) Shannon, C.E, (1950) 'Programming a Computer for Playing Chess', *Philosophical Magazine*, Ser.7, Vol 41, No. 314, Taylor & Francis, Ltd
7) Samuel, A.L (1959) 'Some Studies in Machine Learning Using the Game of Checkers', *IBM Journal of Research and Development*, Vol 3, no. 3, IBM.
8) Silver, D. and Hassabis. D. (2017) 'AlphaGo Zero: Starting from scratch', DeepMind, 18 October. Available at: https://deepmind.com/blog/article/alphago-zero-starting-scratch
9) Ray, C. (2019) *How Stockfish Works: An Evaluation of the Databases Behind the Top Open-Source Chess Engine – good fibrations*. [online] Rin.io. Available at: http://rin.io/chess-engine/ [Accessed 2 Dec. 2019].
10) Anton, R. (2018) 'Revaluation of AI engine alpha zero, a self-learning algorithm, reveals lack of proof of best engine, and an advancement of artificial intelligence via multiple roots', *Open Access Journal of Mathematical and Theoretical Physics*, Vol 1, no 2, MedCrave.
11) Silver, D., Hubert, T., Schrittwieser, J. and Hassabis, D. (2019). *AlphaZero: Shedding new light on the grand games of chess, shogi and Go*. [online] Deepmind. Available at: https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go [Accessed 6 Dec. 2019].
12) Romstad, T. et al (2018) Stockfish [online] github.com. Available at: https://github.com/official-stockfish [Accessed 3 Dec. 2019]
13) Sharma, S. (2018) 'Monte Carlo Tree Search', *Towards Data Science*, 01 August. Available at: https://towardsdatascience.com/monte-carlo-tree-search-158a917a8baa
14) Yannakakis, G.N and Togelius, J. (2018) 'Artificial Intelligence and Games', Springer
15) Silver, D. et al (2018) 'A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play', DeepMind, 06 December, available at https://deepmind.com/research/publications/general-reinforcement-learning-algorithm-masters-chess-shogi-and-go-through-self-play
16) Chess.com. (2019). *Chess.com - Play Chess Online - Free Games*. [online] Available at: https://www.chess.com/ [Accessed 6 Dec. 2019].