IOP Conf. Series: Materials Science and Engineering **864** (2020) 012091 doi:10.1088/1757-899X/864/1/012091

Study of Artificial Intelligence into Checkers Game using HTML and JavaScript

K Idzham K[1], W N Khalishah M[1], Steven Y W[1], M F Aminuddin M S[1], N Syawani H[1], Zain AM[1, 2] and Y Yusoff[1, 2]

[1]School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor Bahru, Johor, Malaysia.
[2]Applied Industrial Analytics Research Group, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor Bahru, Johor, Malaysia.

E-mail: azlanmz@utm.my

Abstract. Artificial Intelligence (AI) in the game industry matures and continues to expand due to its capabilities to explore and exploit in bringing gaming to life. In this paper, we described the game design for the game Checkers, which is developed by using an AI heuristic approach using HTML and JavaScript. A group of people chosen in this study to test the game, which is a validation process to confirm that the game goes as per requirement. The testing activity ends after the final bug fix and the result shows that the game executes properly as expected. The result indicates that the game meets the objective of the system, which is the game, may learn and explore by itself as a heuristic element.

1. Introduction

Today, modern dictionary Meriam-Webster defines artificial intelligence (AI) is a machine's ability to imitate intelligent human behavior [1]. The term "the science and engineering of making intelligent machines" has been invented by John McCarthy in 1956 [2]. For years, artificial intelligence has been rapidly increase implement into video games. The majority of video games, such as racing games, fighting game etc, have different controllable features, including neutral characters or enemy bots. Many industry and company argue that the AI video games have come a long way in that they have revolutionized the way people interact with any form of technology, even though numerous experts are skeptical concerned about such accusations, and especially the idea that these technologies correspond to the definition of "intelligence" standard in cognitive sciences [3].

The heuristic approach concept in AI based on knowledge or analysis of people's thinking. A heuristic technique is more rapid than conventional methods to solve a problem [4]. The aim of heuristic is to find a solution, which is sufficient to solve the problem within a reasonable period of time. The heuristic approach is a way of determining the search towards the goal. It is an informed way to figure out which node's neighbor leads to a target [5].

In the following section, we discussed our implementation of AI heuristic into checker game using HTML and JavaScript. Checkers or Draughts in British English is two players play a type of board game that can be dated back to 3000 B.C. Checkers. Checkers are played by two-players and each player begins with 12 coloured pieces. In this project, we used 20 coloured of pieces for each player to ensure the significant of the implement of the AI heuristic approach. The board consists of 100 squares with two distinct colours which are light and dark colours. Each of the coloured squares has 50 squares respectively. Each of the pieces will then place on the dark colour square near the player. The game is then started by the player that has white pieces. The game ended when one of the player's pieces has

been captured or a player cannot make another move. In this project, we have replaced the player of the dark piece into a computer agent-based that can compete against humans in the Checker game.

2. Related Works

In the process of research, we discovered that many previous technologies have implemented the heuristic method into the game industry. For this paper, we decided to use Alpha-beta pruning search as our main focus. Various search algorithms were designed to solve the shortest possible path problems until the Alpha-beta pruning seeks to reduce the number of nodes assessed in its search tree by a minimax algorithm. Minimax is a recursive algorithm used to select an optimal move for a player provided the other player plays optimally [6]. It has been used in other games like tic-tac-toe, chess, isola, checkers and many more.

Since in most games, the state area is too big for a thorough search, a heuristic method for non-terminal states needs to be bound to the search scope. These heuristic methods are often handmade and very specific to a specific game in game systems for specific games [7]. One of the examples is real-time heuristic search for pathfinding in video games. Time is split up to uniform intervals in the game time model. During each interval, an agent has three options which is to search, perform action, or search and perform simultaneously. The game model goal is to minimize the time intervals necessary for the agent to move from its initial state to the target state [8].

3. Methodology

3.1 Creating a Checkers game with integrated AI

The main idea of this project is to develop a Checkers game that integrated with artificial intelligence and the ability to act as the player's opponent using a heuristic approach. To achieve the result, we have used Atom for coding. The game is built based on HTML, CSS, JQuery, and JavaScript. Next, we differentiate the users into 2 groups, which is experienced and inexperienced. A web-based survey was distributed among the UTM students and which consists of different facilities and departments. We have collected 50 feedbacks within 2 weeks.

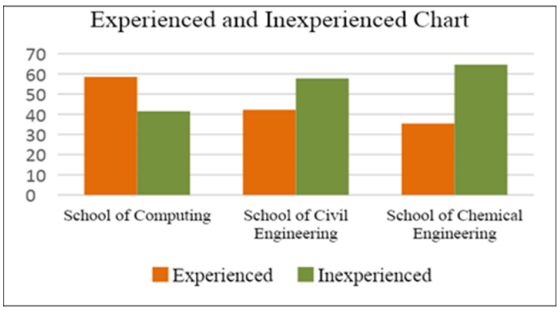

Figure 1. Experienced and Inexperienced Chart

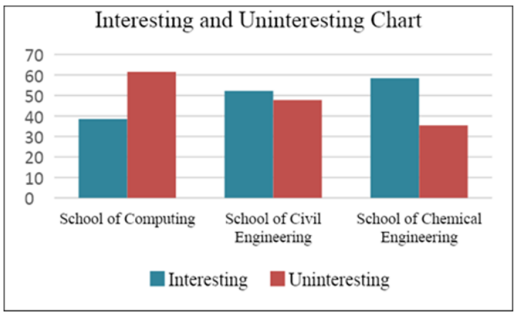

Figure 2. Interesting and Uninteresting Chart

According to the survey as in Figure 1, we have concluded that 45% of the students have experience with the game itself and know how the games mechanize works. They able know to win the game against the AI. Therefore, these students fall into the 'experienced players' category. The other 55% of students claimed that they did not familiar and unable to win against the AI's actions. From this, we perceive that they lack strategies due to their unfamiliarity with the gameplay. 52% of the players found that the game was fun to play while the other 48% said that they got bored after a few rounds (Figure 2).

3.2 The Checkers Game Flow

Figure 3 shows the process in the Checkers game. First, the player must choose a difficulty level which is “Novice” and “Optimal”. After that, a “Start” button will prompt and the player needs to click it to proceed with the gameplay. The player or white piece will move first followed by the AI. After the player ends their turn and if there is no winner yet, the game with continued with the AI’s turn. The process will repeat until there is one of the players who cannot move or out of pieces, then the game will end with the announcement of the winner. After that, the player can repeat the whole game by choosing the “Play Again” button or exit the game if the player does not wish to continue the game.

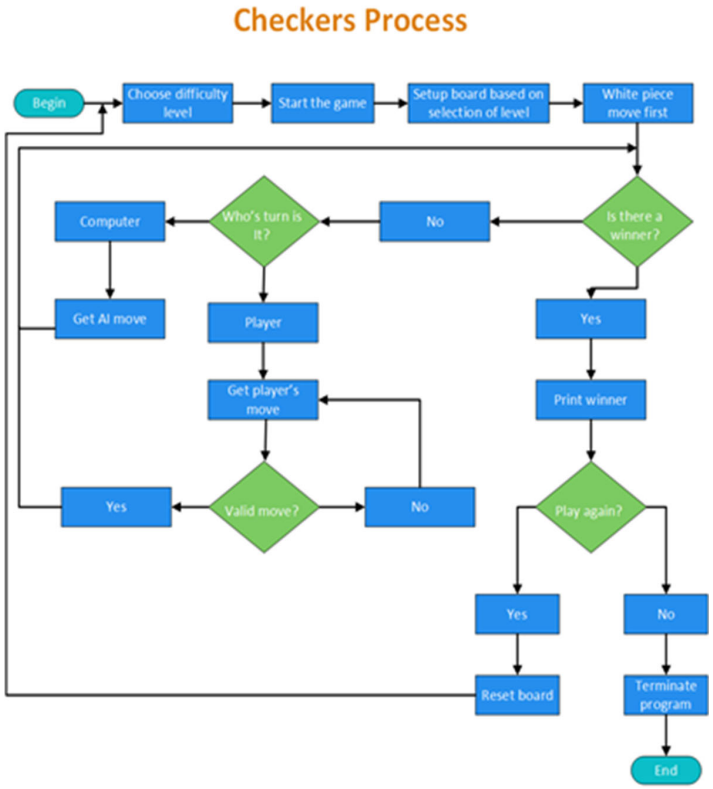

Figure 3. Flowchart of Checker Process

4. Development

4.1. User Interface

The game used a simple interface design with a score shown on the left side of the board (Figure 4). The player can start the game by choosing the difficulty level and then click the "Start" button. Twenty pieces placed on the dark square and each of the colours of pieces placed near the player’s row. Determine which one difficulty level user want to play. These affect how challenging the game is to play, and it runs on a scale of “Novice" and "Optimal".

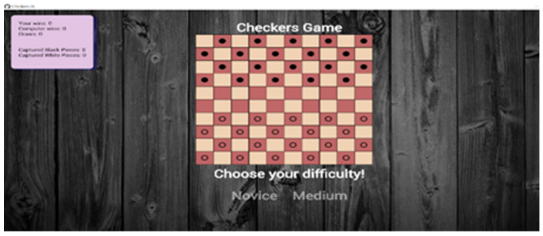

Figure 4. Checkers Main Interface

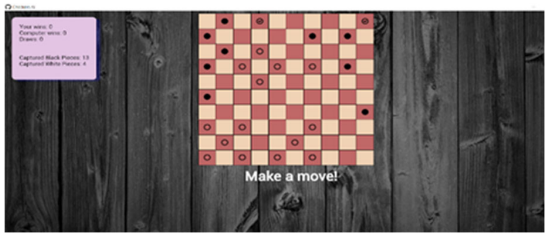

Figure 5. Player Moved Interface and King Interface

The squares become available for the player to click and move his piece diagonally. A player piece will become a “King” by reaching the opposite end of the row (Figure 5). If the piece of a player is “King”, a "√" is shown on the piece. At the beginning of the game, both of the players will have 20 pieces on the board. A piece will become a “King” by reaching the opposite end of the row of the board. The piece has restriction rules which are the pieces that cannot move backward and no single piece can move into a square that already being occupied. On the other hand, the “King” can move similar to the normal pieces but with forwarding or backward movement. After the player moves the piece, the turn will end and the computer will move their piece. The process will repeat until a player cannot move or does not have another piece. The winner will get one score and the captured pieces will show as well. Also, if both of the player and the computer do not have other moves, the score will be counted as “Draw”.

4.2. Algorithm and Documentation

Our approach is to create a computer based-agent using Alpha-Beta pruning in conjunction with the minimax algorithm. The data related to the pieces on the board are stored in the 2D array. There are several nested files which are ai.js, board.js, build.js, control.js and human.js but the major focus will be ai.js.

```
// ai chooses the optimal move 40% of the time, suboptimal (2nd choice) 60%
function takeANoviceMove(turn) {
  var available = game.currentState.allValidMoves(turn);
  var availableAction = [];

  // calculate score for each possible action
  var i = 0;
  while (i < available.length) {
    var action = new AIAction(available[i][0], available[i][1]);
    // get next state
    var next = action.applyTo(game.currentState);
    action.minimaxVal = minimaxValue(next, 2);
    availableAction.push(action);
    i++;
  }
```

Figure 6. How the “Novice” AI Implemented in the Checkers Game

```
  // ai chooses the optimal move
  function takeAMediumMove(turn) {    var available =
game.currentState.allValidMoves(turn);
    var availableActions = [];
    // calculate score for each possible action
    var i = 0;
    while (i < available.length) {
      var action = new AIAction(available[i][0], available[i][1]);
      // get next state
      var next = action.applyTo(game.currentState);
      action.minimaxVal = minimaxValue(next);
      availableActions.push(action);
      i++;
    }
```

Figure 7. How the “Optimal” AI implemented in the Checkers Game

We have decided to implement two different levels of difficulty which are novice or optimal. The method used to specify the level of difficulty is as coding shown in Figure 6 and Figure 7. In Figure 6 and Figure 7, both of them have the same function. To able the AI to determine every possible action needs to take, a while statement is made. The while statement will loop several times according to the size of array in the variable available. The major difference that separates both of the figure is action.minimaxVal= minimaxValue (next, 2);. It causes the AI in novice level to choose the unfit or minor possibility to move the pieces by dividing the next value with value of 2 in the novice level.

The stateScore act as the infinity value for the alpha and beta. In this project, we rename the alpha for white and beta for black as to indicate which the current turns are. The value of will always set as -1000000 and 1000000 for white and black respectively. These values will then pass down subsequent to other possibilities or also known as child. Each time when it is the AI turn, it will find the best possibility by choosing the minimum value before moving the pieces in the for-loop statement. The AI

will distinguish which. The AI will repeat the process until it can accept the best possibility. Also, the AI will save the computation time by pruning the other possibilities after it has found the minimizer has can guarantee a lesser or best value.

```
function minimaxValue(state, count = 0) {
 if (count == 5) {
   return Game.score(state);
 } else {
   var stateScore;
   if (state.turn == "W") {
     // for white turn, we maximize score, so we initialize with a score lower than the
function could reach
     stateScore = -100000;
   } else {
     // for black turn, we minimize score, so we initialize with a score higher than
the function could reach
     stateScore = 1000000;
   }
   var availableMoves = state.allValidMoves(state.turn);
   var availableNextStates = [];
   // calculates available next states
   for (i = 0; i < availableMoves.length; i++) {
     var action = new AIAction(availableMoves[i][0], availableMoves[i][1]);
     var nextState = action.applyTo(state);
     availableNextStates.push(nextState);
   }
   // gets minimax value for all the available next states
   availableNextStates.forEach( function(nextState) {
     var nextScore = minimaxValue(nextState, count + 1);
     if (state.turn == "W") {
       // maximize W
       if (nextScore > stateScore) {
         stateScore = nextScore;
       }
     } else {
       // minimize B
       if (nextScore < stateScore) {
         stateScore = nextScore;
       }
     }
   })
   return stateScore;
 }
}
```

Figure 8. The Algorithm for Alpha Beta Pruning in the Checkers Game

5. Testing

In this paper, three main aspects need to be considered in testing to be deemed a 'successful' test which is functional, performance and usability. The testing is about verifying do the Checker game meets the functional, performance and implementation requirements identified in the procurement specifications. A group of students was chosen in this study to test the functionality of the applications. The experiment result shows that the effectiveness of AI heuristic approach in the Checkers game.

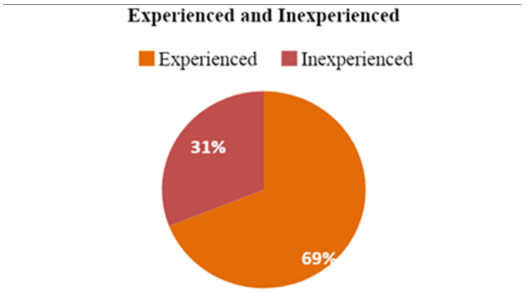

Figure 9. Experienced and Inexperienced Chart

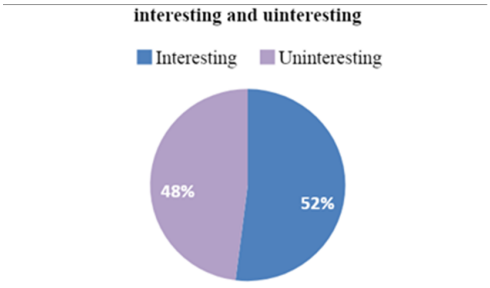

Figure 10. Interesting and Uninteresting Chart.

Survey data was obtained from UTM student's different faculties and departments which are from several students from the School of Computing, School of Civil Engineering and School of Chemical Engineering. The results of the findings are based on questions from the questionnaire, which have been divided and categorized. The pie chart shown that 69% user experienced how to play Checkers game while another 31% user does inexperienced in how does the Checkers game works. From this survey, we conclude that 48% the student found that the game was uninterested because most of the students think that it has a minor bug that affect the gameplay itself. Meanwhile, another 52% of the students think that the game was interested because of the AI implementation.

6. Conclusion

The aim of this study is to show the implementation of AI heuristic approach in the Checkers game. The search algorithm that can be found in the application is the Alpha-Beta pruning. In order to find the best possibility, the algorithm will seek to decrease the number of nodes that are evaluated by the minimax algorithm. The application is then tested by the user to determine the effectiveness of the A.I's heuristics in making decisions for each step of the game. We find that the level of AI can actually compete with an experienced checker player and occur only a minor bug problem, but does not pose any negative impact to the user experience and gameplay itself.

Acknowledgment

Special appreciations to editor and all reviewers on the useful advices and comments provided. The authors greatly acknowledge the Research Management Centre, Universiti Teknologi Malaysia (UTM) (HIR-vot. No. Q.J130000.2551.20H71).

References

[1] Merriam-Webster 2019 *Definition of Artificial Intelligence.*
[2] Martin C 2011 *The Independent.*
[3] AI Frontiers 2018 *Artificial Intelligence in Games.*
[4] Study.com. 2003 *Heuristic Methods in AI: Definition, Uses & Examples.*
[5] Vladimir B 2019 *Basic AI Concepts: A* Search Algorithm. Stack Abuse*
[6] Hart P E, Nilsson N J and Raphael B1968 IEEE *Trans. on Syst. Sci. and Cybernet.* **4** 2 100-107
[7] Neo4j 2019 *The A* algorithm.*
[8] Software Testing Fundamental 2019 *Functional Testing.*