ARTIFICIAL INTELLIGENCE PROJECT 2 REPORT

Büşra Gökmen-150116027

Ömer Faruk Çakı-150117821

DESIGN DOCUMENT:

Class:	Fields:	Methods:
ConnectFour: ● Human Player vs Human Player	piece(int) = player's piece integer representation COLUMN_COUNT (int)= total number of column ROW_COUNT(int) = total number of row board(integer array) = board representation with 0 pieces game_over(Boolean) = control value to check end of game turn(int) = the value that holds the players' turns of playing	create_game_board(): -create board 2D array with number of row and column play_piece(board, row, col, piece): -make play with piece on the board with column number playable_location_control(board, col): - if position value is 0, this returns valid position get_next_row(board, col): - this return available row position for given column game_board(board): -return final state of board winner_check(board, piece): -this control four pieces for a player on horizontal, vertical and diagonal to win and if there are set of four, then return True

Class:	Fields:	Methods:
AI_PLAYER(AI): ● AI player using h1 vs AI player using h2 ● AI player using h2 vs AI player using h3 ● AI player using h1 vs AI player using h3 AI_PLAYER(Human): ● Human Player vs the Best AI Player Configuration	PLAYER(int) : opposite player's piece representation to AI AI (int): AI player's piece representation COLUMN_COUNT (int): total number of column ROW_COUNT(int) : total number of row board(integer array) : board representation with 0 pieces game_over(Boolean) : control value to check end of game turn(int) : the value that holds the players' turns of playing position_values(integer array): keep score values of positions as a 2D array	create_game_board(): -create board 2D array with number of row and column play_piece(board, row, col, piece): -make play with piece on the board with column number playable_location_control(board, col): - if position value is 0, this returns valid position get_next_row(board, col): - this return available row position for given column game_board(board): -return final state of board winner_check(board, piece): -this control four pieces for a player on horizantal, vertical and diagonal to win and if there are set of four, then return True check_square_position(board, piece): -this calculate and return total position of pieces score of current player consecutive_two(board, piece): -this return count of set of consecutive two pieces for current user consecutive_three(board, piece): -this return count of set of consecutive three pieces for current user consecutive_four(board, piece): -this return count of set of consecutive four pieces for current user

		find_playable_locations(board): - available position is added to playable locations and return playable locations check_game_over(board): - check game is over or not, then return True or False evaluation_1(board, piece): -For the current player, the numbers of consecutive two pieces, consecutive three pieces are multiplied and added by the specified specific coefficients. The same action is taken for the opposing player and subtracted from the other total. Return as result score evaluation_2(board, piece): -For the current player, the numbers of consecutive two pieces, consecutive three pieces, consecutive four pieces are multiplied by the specified specific coefficients and added to total score. The same action is taken for the opposing player and subtracted from the other total. And values of the positions of the current player's pieces on the board are also added to total score. Return as result score evaluation_3(board, piece): -For the current player, the numbers of consecutive two pieces, consecutive three pieces, consecutive four pieces are multiplied by the specified specific coefficients and added to total score. The same action is taken for the opposing player and subtracted from the other total. Return as result score minimax_1(board, depth,maximizingPlayer): -to find the best move for the AI 1, the depth limited game is simulated with the help of evaluation 1 and returns the

		piece column of the optimal state minimax_2(board, depth,maximizingPlayer): -to find the best move for the AI 2, the depth limited game is simulated with the help of evaluation 2 and returns the piece column of the optimal state minimax_3(board, depth,maximizingPlayer): -to find the best move for the AI 3, the depth limited game is simulated with the help of evaluation 3 and returns the piece column of the optimal state

h1, h2, and h3 Explanation

---

evaluation heuristic 1(AI1): For AI (maximizing player), we thought it an advantage to have a lot of sequential double and triple groups of their pieces. That's why 10 points were added to the score for each group of doubles. 1000 points were added to the score for each group of three. For the minimizing player, 10 points were subtracted from the score for each group of doubles and 1000 points were subtracted from the score for each group of three.

evaluation heuristic 2(AI2): As a result of our research, we learned that it is an advantage to start the game from the columns in the middle in the Connect Four game and to collect the player's pieces there during the game to win the game. For this reason, we have given different values to the squares on the board depending on whether it is in the middle or in the corner. For AI, we tried the approach of adding extra points to score according to the position of the pieces and subtracting the opponent's points from score. Again, as with other heuristics, we thought it was an advantage to have a large number of consecutive pairs and three tracks for AI (maximizing player). Thus, 10 points were added to the score for each group doubles. 1000 points were added to the score for each group of three. For the minimizing player, 10 points were subtracted from the score for each group of doubles, and 1000 points were subtracted from the score for all three groups. Additionally, 100000 points have been added to the score for cases where AI pieces are consecutive quads. When the same situation was found in the opponent player, 100000 points were subtracted from the score.

evaluation heuristic 3(AI3): For AI (maximizing player), we thought it an advantage to have a lot of sequential double and triple groups of their pieces. That's why 10 points were added to the score for each group of doubles. 1000 points were added to the score for each group of three. For the minimizing player, 10 points were subtracted from the score for each group of doubles and 1000 points were subtracted from the score for each group of three. In addition, 100000 points were

added to the score for cases where AI's pieces were in consecutive quadruples. When the same situation was found in the opposing player, 100000 points were subtracted from the score.

Finally, A2 has the best heuristic method. Won the games against AI1 and AI3. He eventually played AI2 with the human player. And AI2 won most games against us.

Ply Number:

We try the minimax algorithm with 1,2,3,4 and 5 ply number. Maximum ply number for the best move is 5 for AI1, AI2 and AI3 in along with the maximum time. When we increase the number of ply more, the game progresses very slowly because the calculation time increases too much.

complexity of AI :

column number, number of initial states: 7

ply number	complexity
1	7^1
2	7^2
3	7^3
4	7^4
5	7^5