Ultimate Tic Tac Toe Game

We made an ultimate tic tac toe game with an Ai (MiniMax & Alpha-Beta Pruning).

We created two classes (Board & Ultimate).

Class Board :

1. Array of character with size of 9 which presents the indices of the game board.
2. Two vectors (X,O) : each of them saves the moves of the player in the active board.
3. Constructor initializes the boards (1-9).
4. Undo function : takes two parameters (int index, bool turn), replace the index with a character that represents the position of the index and pops out an element from the vector (depends on the turn).
5. Play function : takes two parameters (int index, bool turn), edits the board by replacing the index with (X,O) depends on the turn.
6. Score function : returns the score of the current board such as :
7. ‘1’ represents X won.
8. ‘-1’ represents O won.
9. ‘0’ represents Tie.
10. Empty function : returns a vector with all the available moves.
11. Check function : takes one parameter (int index), checks if the current index is available or not.
12. PrintUp function : prints the first row.
13. PrintMid function : prints the second row.
14. PrintDown function : prints the last row.
15. CheckXO function : takes one parameter (int index), returns ‘1’ if the index in the board is (X), returns ‘-1’ if the index in the board is (O) and returns ‘zero’ if the index is empty.
16. CalculaterSmall function : calculates the average of the current board by doing this:

First of all, stores the positions of X and O in array , then check if there is any close spot to win like “1, zero ,1” (in the array 1 is X, zero is empty) and adds 2 to the average.

if there is a spot like (zero, 1 ,zero) adds 1 to the average and then returns the total average of the current board.

Class Ultimate :

1. We defined an array (B) of type “Board” with size of 9 which represents the single boards on the screen.
2. We defined a Boolean “turn” which represents the current turn.
3. We defined 3 maps of type <int, int> :
4. Wx : stores which board ‘X’ closed.
5. Wo : stores which board ‘O’ closed.
6. Tie : stores which board has no available indices to play on (full board).
7. Undo function : takes three parameters (int Board, int Index, int Turn), this function uses the parameter (Board)-1 to call the correct small board in “B”, then it calls the function undo in the object B (Board) and sends Index and Turn as parameters.
8. Constructor Ultimate : initialize the turn with 1 and initialize the maps (Wx,Wo,Tie) to -1.
9. Print function : prints each single board on the screen.
10. Play function : takes three parameters (int Board, int Index, bool T), this function uses the parameter (Board)-1 to call the correct small board in “B”, then it calls the function play in the object B (Board) and sends Index and T as parameters.
11. CheckerBoard function : loops all the boards in ‘B’ and checks if the board is closed or not by calling the score function in ‘B’ :
12. If it was zero assign one to map Tie with key (iterator+1).
13. If it was 1 assign one to map Wx with key (iterator+1).
14. If it was -1 assign one to map Wo with key (iterator+1).
15. else, assign -1 to Wx & Wo & Tie with key (iterator+1).
16. CheckerIndex : takes two parameters (int Board, int Index), checks if the Index on the B[Board-1] is available by calling check function in object ‘B’ and returns its value.
17. AvailableBoard : takes one parameter (int ind), checks if the score of B[ind-1] is equal 10 (available to play) return true, otherwise return false.
18. BigWin function : checks if there are any winner in the game by using the maps (Wx, Wo, Tie) :
19. If the winner was X return 2000.
20. If the winner was O return -2000.
21. If there is no winner in the game loop and check all the indices if there is a -1 return 100, which means the board is available to play, else return zero which means Tie.
22. Calculater function (it isn’t perfect) : we defined an int AVG and array of integers with size of 9 (Scores[9]), and calls the function CheckerBoard (to update{ Wx, Wo, Tie } to get the final board status), loops from 0 to 9 and do this:
23. If Wx[a+1] equals 1, assign 15 to Scores[a].
24. If Wo[a+1] equals 1, assign -15 to Scores[a].
25. If Tie[a+1] equals 1, assign -999999 (which means this board is closed) to Scores[a].
26. Else calls the function CalculaterSmall from B[a] and assign the results to Scores[a].

Loops from 0 to 9 and if Scores[i] is not equal to -999999 add the Scores[i] value to AVG , finally returns AVG.

1. All function :takes one parameter (int index), define a vector of type int with size zero called emp and calls the function AvailableBoard with parameter index if the result was true returns emp, otherwise calls the Empty function from B[index-1] , assign it to emp and returns emp.
2. Minimax function (Alpha Beta Pruning ) : takes six parameters (int table, int Index, int Depth, int alpha, int beta, bool Turn).

First of all, we defined int Max = - INT\_MAX, Min = INT\_MAX, the base case is if the Depth = zero, it returns the result of calling the function Calculator.

If the Turn was one:

1. Calls the function play in the object B[table] and sends Index and Turn.
2. Calls the CheckerBoard function.
3. We defined an int (Score) and saved the result of calling the BigWin function in it, if the Score != 100 return the Score.
4. We defined a vector of int (emp), we check if the board is available by calling AvailableBoard function and pass the Index as a parameter, if it is true :

* Assign B[Index-1].Empty() to emp.
* Loops on the emp elements { for(auto x : emp)}, defined int v and assign the result of the Minmax(Index-1, x, Depth-1, alpha, beta, !Turn).
* Inside the loop; if the Depth != 1, calls the Undo(Index, x, !Turn).
* Inside the loop; calls the CheckerBoard function, assign the minimum value between v & Min to Min, Min = min(v,Min), assign the minimum value between beta & Min to beta,

beta = min(beta,Min).

* If (beta <= alpha) breaks from the loop.
* Return Min.

If the result of the AvailableBoard function was false :

* Loops from 1-9, we defined a vector of int (em) and assign the result of calling the All(iterator).
* Loops on the emp elements { for(auto x : emp)}, defined int v and assign the result of the Minmax(Index-1, x, Depth-1, alpha, beta, !Turn).
* Inside the loop; if the Depth != 1, calls the Undo(Index, x, !Turn).
* Inside the loop; calls the CheckerBoard function, assign the minimum value between v & Min to Min, Min = min(v,Min), assign the minimum value between beta & Min to beta,

beta = min(beta,Min).

* If (beta <= alpha) breaks from the loop.
* Return Min.

If turn was zero:

1. Calls the function play in the object B[table] and sends Index and Turn.
2. Calls the CheckerBoard function.
3. We defined an int (Score) and saved the result of calling the BigWin function in it, if the Score != 100 return the Score.
4. We defined a vector of int (emp), we check if the board is available by calling AvailableBoard function and pass the Index as a parameter, if it is true :

* Assign B[Index-1].Empty() to emp.
* Loops on the emp elements { for(auto x : emp)}, defined int v and assign the result of the Minmax(Index-1, x, Depth-1, alpha, beta, !Turn).
* Inside the loop; if the Depth != 1, calls the Undo(Index, x, !Turn).
* Inside the loop; calls the CheckerBoard function, assign the maximum value between v & Max to Max, Max = max(v,Max), assign the maximum value between alpha & Max to alpha, alpha = max(alpha,Max).
* If (beta <= alpha) breaks from the loop.
* Return Max.

If the result of the AvailableBoard function was false :

* Loops from 1-9, we defined a vector of int (em) and assign the result of calling the All(iterator). Loops on the emp elements { for(auto x : emp)}, defined int v and assign the result of the Minmax(Index-1, x, Depth-1, alpha, beta, !Turn).
* Inside the loop; if the Depth != 1, calls the Undo(Index, x, !Turn).
* Inside the loop; calls the CheckerBoard function, assign the maximum value between v & Max to Max, Max = max(v,Max), assign the maximum value between alpha & Max to alpha, alpha = max(alpha,Max).
* If (beta <= alpha) breaks from the loop.
* Return Max.

1. Start function: basically it starts the game, ask the user to input which board he want to play on and on which index.

We defined a bool called taken which represents if the board was available to play or not.

Loops with true if taken was true this means the board is not available to play so feel free to choose a new board.

After you choose an index to play on the board calls the function play on the chosen board and pass the index.

Calls the CheckerBoard function to update the maps (Wx,Wo,Tie) , if the BigWin was 10 print X IS THE CHAMPION and breaks , if BigWin was -10 print O IS THE CHAMPION and breaks , if BigWin was zero print Tie and breaks, else assign !turn to turn.

Define int best and assign it to -9999999 , int ind and assign it to -1, vector<int> emp.

If AvailableBoard(Index) was true:

1. Assign B[Index – 1].empty() to emp.
2. Loops on the emp elements { for(auto x : emp)}, define int v , assign Minmax(Index – 1, x , 6 , -999999 , 999999 , turn) to it.
3. Calls the Undo function and pass the Index, x, turn.
4. Calls the CheckerBoard function.
5. If v < best assign v to best and x to ind.

If AvailableBoard(Index) was false:

1. Loops from 1 to 9.
2. Calls the All (iterator) and assign it to em.
3. Assign B[Index – 1].empty() to emp.
4. Loops on the emp elements { for(auto x : emp)}, define int v , assign Minmax(Index – 1, x , 6 , -999999 , 999999 , turn) to it.
5. Calls the Undo function and pass the Index, x, turn.
6. Calls the CheckerBoard function.
7. If v < best assign v to best and x to ind.

Then prints where the AI move was, and call the function play and pass (Index, ind, turn), call the function Print to print the board.

Assign !turn to turn, calls the function CheckerBoard to update the maps , and then checks if the board is taken or not by using if statement depends on the maps.

Finally if the BigWin was 10 print X IS THE CHAMPION and breaks , if BigWin was -10 print O IS THE CHAMPION and breaks , if BigWin was zero print Tie and breaks.

In the main just defined an object from Ultimate and call the function Start().

By : Somaia Omar Hasan AlQiam & Muhammad Ramadan Farraj.

Supervisor : Dr.Malak Abdullah.