Tic Tac Toe

Brian Ramirez

CSC 17C

Date: 6/6/19

Guthub Link: https://github.com/bramirez246/RamirezBrian\_CSC17c\_42511

**Introduction**

Title: Tic Tac Toe

This project is a Tic Tac Toe program, it is a board game in which players take turns to place a character, an ‘X’ or an ‘O’ on a 3x3 grid. The Program begins by prompting the user to select between two game modes. The first mode is human vs. human and the second mode is human vs. CPU. The human vs. human mode is straightforward it will let the first player ‘X’ go first and then just switch the token to ‘O’ and now the second player makes their move. This goes on until one player gets three in a row horizontally, vertically, or diagonally. The game will also end if all nine spaces are filled and result in a tie. After that the players will be prompted to play again. In the human vs. CPU mode the user is assigned to ‘X’ and they go first, then the CPU uses the minimax algorithm to find the best move. So on, until either the computer wins or the game results in a tie.

**Summary**

Project size: about 847 lines

Number of variables: about 10 main variables

Number of methods: 16

This project includes STL containers such as vectors and pairs. It also includes recursion with the minimax algorithm. The project also utilizes the concept of traversing a tree. For example, when the minimax algorithm is called it has to, recursively, try all possible moves for itself and of those moves it branches out to all the possible moves of the user until it reaches the end of each possible board state and scores each board state.

**Pseudo Code (for play\_cpu function)**

*Output the instructions*

*Do while retry == true*

*prompt player if they want to start a game*

*If user enters ‘y’ or ‘n’*

*retry is set to false*

*Else*

*retry is set to true*

*outputs a message saying that input is invalid*

*While play == ‘Y’ or ‘y’*

*Initialize the board*

*Do while gameStatus == ‘N’*

*Display the board*

*If turn is even*

*Then get the human player’s move*

*Increment turn*

*Else it is computer’s turn*

*Set AImove to the move returned by bestMove function*

*Enter the move into board vector*

*Increment turn*

*Set gameStatus equal to value returned by checkWin function*

*Once outside of do while loop display the board and output winner*

*Prompt player if they want to play again*

*Validate input*

*Reset turn variable*

**Objects**

My program has two objects, the Board object and the AI object. The Board object includes a six variables, mainly to hold player/game information. For example, board is a 2d vector that is used to hold the information for the 3x3 grid/matrix, player1 is a pair that holds the first player’s token and name, and play holds the user’s decision when prompted if they want to play. There are 16 functions in this object that utilize the private variables. For instance, initBoard() initializes the vector to fill all the spaces on the board with blanks, displayBoard() displays the game board, and switchTokens() switches the token from the current player to the opposite token (to get ready for the next turn).

|  |
| --- |
| Board |
| * gameStatus: char * play: char * token: char * changeName: char * player1: pair<char, string> * player2: pair<char, string> * board: vector< vector <char> > |
| * initBoard() * outputInstruct() * displayBoard() * getCheckInput() * getPlayers() * outputWinner(char) * checkWin() * switchTokens() * play\_pvp() * getGameStatus() * getToken() * getChangeName() * getPlay() * setPlay(char ) * setChangeName(char ) * setGameStatus(char ) * setToken() |

The AI object has one private variable that is a struct called Move. Move has two int variables to hold the value for row and column. AI has 7 methods that will mostly be utility functions for the AI to make its best move in bestMove().

|  |
| --- |
| AI |
| * Move: int row, col |
| * movesLeft(vector< vector <char> > &) * evaluation(vector< vector <char> > &) * minimax(vector< vector <char> > &) * bestMove(vector< vector <char> > &) * getHumanMove() * outputWinner(char) * play\_cpu() |

**Program**

// Include libraries

#include <cstdlib>

#include <stdlib.h>

#include <iostream>

#include "tictac.h"

#include "AI.h"

using namespace std;

int main(int argc, char\*\* argv) {

// Declare and initialize variables

int choice=0;

bool retry=true;

Board tictac1;

AI tictac2;

// Ask player(s) what mode they would like to play

cout << "Select one of the following modes: " << endl;

cout << "\t1. Human vs. Human \n\t2. Human vs. Comp (you play as 'X')\n" << endl;

do{

cout << "Enter your choice: ";

cin >> choice;

if(choice == 1 || choice == 2)

retry = false;

else{

retry = true;

cout << "Invalid input, try again." << endl;

}

}while(retry==true);

cout << endl;

// There are 2 cases that can happen

// If players choose option 1 then run pvp mode

// if option 2 run player vs cpu mode

switch(choice){

case 1: tictac1.play\_pvp();

break;

case 2: tictac2.play\_cpu();

break;

default: cout << "Invalid input, try again.\n";

break;

}

return 0;

}