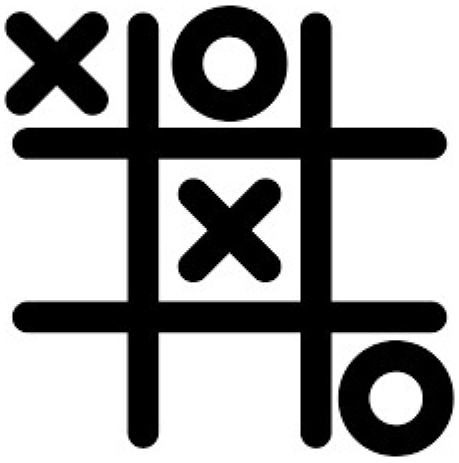
Tic-Tac-Toe game

Algorithms & Data Structures

SET08122



Strident: Konstantin Tomov

ID:40087091

Introduction.

The aim of the project is to create a tic tac toe game with as many features as possible, using only ‘C’ programing language. The game must be designed to be efficient and use appropriate algorithms and data types to achieve its goals. It is equipped with responsive interface and menus, using the arrow kays for navigation and consol. At the currant stage of development, only the 2-player mode is fully completed, giving the user the ability to play verses another player until someone wins or the players hove a draw. After the end of each game the players can review and analyze their game using the replay function. Overall the program run smooth without significant delays thanks to custom algorithm and matrix storing system.

Design

The design of the software began with the game aspect and the play board used for this game. The combination of the board needed for the game and the limited user interface of a terminal, made the task of creating an easy and enjoyable environment very hard. A complex solution of arrow controls and menus wore devised in order to make the controls very easy and instinctive for the user.

After the UI design was finished, the next step was to start working on the functionality of the game. The original plan was to use custom struct to store the board and pleased symbols (X and O) on it. That solution was overcomplicated and could potentially lead to a lot of problems, therefore a simpler 3 by 3 metrics was created to hold all the symbols as the actual matric already represent the board. Also, the matrix gives the ability to control the size of the board.

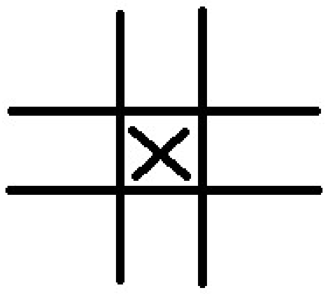
The 3rd major design choice was the algorithm used to analyses who is the winner if there is one. After some research done to find if there is a specifically optimized algorithm used for this purpose, a custom one was created to fit the need of a quack check for winner at the end of every move. This Algorithm I more noticeable and efficient when the board start increasing in size.

Algorithm explanation.

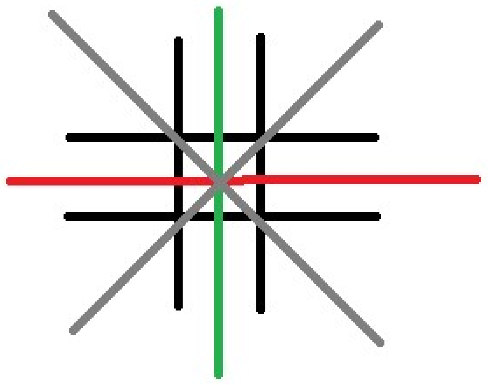
After the player makes a move there is the possibility of him winning. That means that the placement of the symbol on the map can be used as a starting position of the search for a winner instead of checking every possibility on the map following the rules of the game. When a symbol is pleased by a player, following the rules he can win only if there is 3 horizonal, vertical or diagonal symbols. Knowing the position of the pleased Symbol on the matric, the algorithm needs to check only the 3 directions connected to that point, thus cutting the search possibilities to one 3rd (33.3%) of the total possibilities.

Pleased Symbol

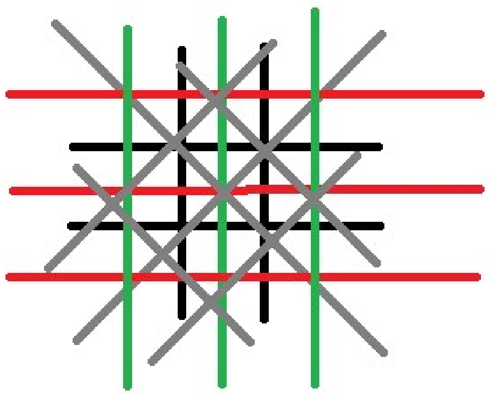
by the player.



Algorithm check



100% board check



The final design choice was the type of data structure needed to store the game for reversing moves the replay. The initial logical choice was to use stack to store the player moves, that way when popped it can be used to easily reverse to the previous move. Unfortunately, that data type was not very well soothed for the replay function as access to the first item of the stack is required. That meant that the stack was replaced by a linked list in the form of another matrix, storing the sequence, coordinates and the player that made the move.

Enhancements

There are three features that could be developed and implemented in the project.

The first of them is variable size of the playing board. On the currant stage this is not too hard to implement as the only thing needed to achieve it is to change the size of the matrix making the game board. Also, that means that an option function should be added to the Manu.

The second feature to be added is the 1 player option. Giving the user the opportunity to play against an AI. This is hard to implement as there must be also option to choose the strength of the AI opponent as if the AI is making the best moves possible the only game result will be losing or draw.

And the last optional future could be ranking board with the ability to save and replay all the games of the users. In order to achieve this the games must be saved on an external file or database and there must be customization of the users that saves at leas the user name and his/her victories/scores. Although the design allows for this to be implemented, it will be a lot of additional code and work.

Critical Evaluation

The main two features that work very well are the victory algorithm and its efficiency and the user control method. The algorithm and how it works is explained on page 2. It is 66% more effective on a field with size 3x3 and the efficiency incises exponentially as the size of the field increases. The control implementation is very instinctive and fast, making it feel more like a real game that a terminal program.

The worst implemented part of the project is the separation of the code into fiction and the use of object orientation need to be increased a lot. That will make it easier to read and add new code, as well as cut out some repeating code that could be reused from functions.

Personal Evaluation

The project was very interesting as I have never played tic-tac-toe or made a game. Although I have programed in 'C' before making the terminal fell like an interactive and easy to use environment, was very hard and took most of the time spend on the project. The actual game is easy with just a few rules that give me the opportunity to concentrate more in the code and the quality of how the program interact with the person, than trying to understand how the game works and trying to implement it.

I feel like I could have done some more functionality easily and very quickly like making the game board variable in size, as I have already set up the code to do that. Unfortunately, I ran out of time before I could have achieved everything that I have planned because of poor time management on my part. The main problem I had with time management was underestimating the amount of time it takes to research, learn and implement new skills.