Improvements

Assignment 3

Jacob Rosengren: 100581854

INFR 4320 - Artificial Intelligence in Games

You will be marked on creativity of your game, satisfying the specifications such as the efficiency improvements, documentation (name, date, program description, descriptive variable name & constants, etc.) and correct behaviour.

* Generation
  + Used a struct called Board to store; the state of the board (state), possible valid moves (children), the worth of the board (worth), and if the board is a winning board (Won).
    - The boards state is an array of 9 int’s. (array<int, 9>)
    - The children is a vector of int’s, each int corresponds to the child's position in the Boards vector; as there are 765 boards, the number will be between 0 and 764. A board can’t have more than 9 children.
  + Generated all boards at the beginning and stores the values into a vector called Boards.
    - This takes 0.8 to 1.2 seconds.
  + Only generated 765 boards, the smallest possible number of unique boards.
    - Saves on memory and wasted looping.
    - Though I had to have 2 different states of the current board because the boards generated in the vector don’t care about rotation of flipping, and the board would rotate to the new board after every move.
* Minimax
  + The minimax will search through the nodes based on the boards position in the vector, not the board state.
    - Int position = index of the board in vector Boards.
    - Int alpha = passes through the max worth found.
    - Int beta = passes through the min worth found.
    - Bool maximizingPlayer = is the player trying to maximize or minimize.
  + The minimax will only loop through the boards children; as the children are the only valid states the board can become.
  + Didn’t pass the vector of boards through the minimax function, instead stored it where the function was able to access it; though passing a reference would have been just as good.
    - I was first passing through an entire tree everytime so it took 5 second to generate, without passing through a copy or with using a reference, it has reduced the time to 0.0005~0.002 seconds.
  + Used alpha-beta pruning to cut down on the amount of processing I needed to do.