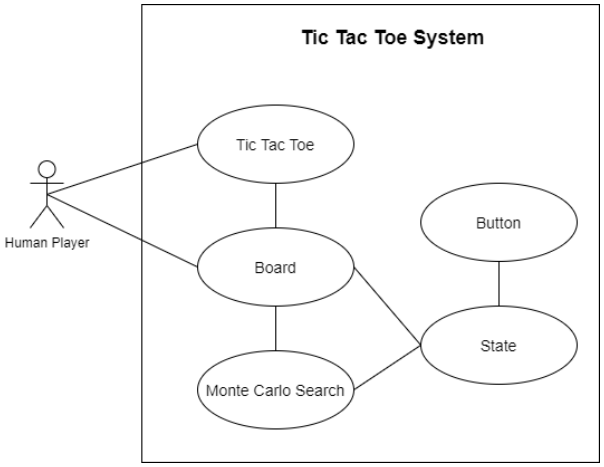
Tic Tac Toe with Monte Carlo and UCT evaluation

Andris Docaj

The goal of the project was to create a computer program which would be able to play the game of Tic Tac Toe using the ideas made popular by AlphaGo Zero. This means that the computer would have to use Monte Carlo techniques to analyze the game search tree, and an upper confidence bound (UCT) function to determine which square to choose. The function that is commonly used has the following form,

$$UCT = {WinValue}/{VisitValue} + \sqrt{{2\ln(TotalSimNumber)}/{VisitValue}}$$

Where the WinValue is the number of wins a particular square contains after a specific number of simulations. VisitValue is the number of times a square has been visited. TotalSimNumber is the total number of simulations, which for our case is one thousand. Instead of using 2 in the square root, I found best results using the constant 0.00017, where win values had a more predominant factor.
For this particular project I was the sole contributor and thoroughly enjoyed the experience. The potential users of the created software include any person who enjoys solving puzzles and applying their mind to logic games. The use case diagram of the project looks as follows,



The use case Tic Tac Toe is responsible for taking the human player's preference in terms of board size, or to determine who makes the first move, the human player or the computer. The interaction between human player and Tic Tac Toe object occurs through a frame. The information acquired from the user is passed on to the Board use case. The Tic Tac Toe use case contains default values if the human player refuses to make any selection. More options can be added to the Tic Tac Toe use case, but the only basic options that presently exist deal with board size, and move order.
The Board use case takes the information provided by the Tic Tac Toe use case and applies it. If a particular board size is chosen by the user, the information is used to display the right board size on the screen. In addition, the Board use case records user moves and provides this information to the State use case. Moreover the Board use case provides the State use case with the proper move order and

marker information also. If the computer is supposed to move first the Board use case passes the information to the Monte Carlo Search use case, so that a best move option can be calculated.
The State use case determines the state of the game, if the game is a win for a particular marker, or a draw. The data about clicked squares from the human player is passed from the Board to the State use case. In addition, the Monte Carlo Search analysis passes information to the State use case on its decision of where the computer will move next.
The Button use case maintains record on the number of wins, and UCT values for each square. The Button use case takes the information from the State use case. The information applied to the Button use case through the State use case comes from the Monte Carlo Search use case. This particular type of information, win and uct values, is altered after a simulation that occurs in the Monte Carlo Search use case. The Button information is accessed from the Monte Carlo Search through the State use case in order to decide the best move for the computer player.
The Monte Carlo Search use case takes information from the Board use case, if the computer needs to make the first move for example. After the analysis is done and the UCT values determined, the computer is able to make its move. The State use case is checked if the move is terminal, a game ending move. In addition, the Monte Carlo Search use case involves the State use case during its simulations to determine when a particular simulation has ended, or it can be ended. In other words, a win or a draw can occur. So, the play-outs in the Monte Carlo Search use case are still quite light, but the State use case information applied makes the button statistics more useful.