Maxwell Giancaterino
Project User Manual
NETS 150

Overview

The main part of my project is located in the TurnSolver package. This set of source files is designed as an interface for creating 2-player turn-based games with a turn solver that uses minimax to make moves for one player. Someone wanting to use this package would need to implement two interfaces in particular: GameGraphics and GameState

GameState handles the rules of your game like legal moves, win conditions, and game state evaluation. This last part is pretty important. When you implement this interface, you must provide a function that will return a double according to the value of the current state of the game for the bot. This will be used by the minimax algorithm to find the most favorable move for the bot. A better description of each method can be found in the interface itself.

GameGraphics handles the graphical portion of your game such as drawing your game, prompting user input for moves and giving messages for when the game is over. Again, a better description of each method can be found in the interface itself.

There is technically a third interface, Turn, that requires no method implementations and is just used internally by your implementation to keep all types coherent.

Once you implement these three interfaces successfully, you can instantiate your implementation of GameGraphics and hand it the provided Game class run method. This will run your game with the bot making moves against you.

Usage

To see this turn solver in action, I've also implemented two concrete games by doing exactly what I described in the previous section. These games are tic-tac-toe and Connect Four, and are located in the packages TicTacToe and ConnectFour respectively.

The tic-tac-toe game is generalizable to any n-by-n board. You can run the game by running TicTacToe.Play [n]. It defaults to standard 3x3 tic-tac-toe, but this can be changed the one optional command-line argument.

The Connect Four game is generalizable to any n-by-n board with a winning length of toWin. You can run the game by running ConnectFour.play [n] [toWin]. n defaults to 7 and toWin defaults to 4. These can be changed with the optional command-line arguments.

These are some screenshots of the games: