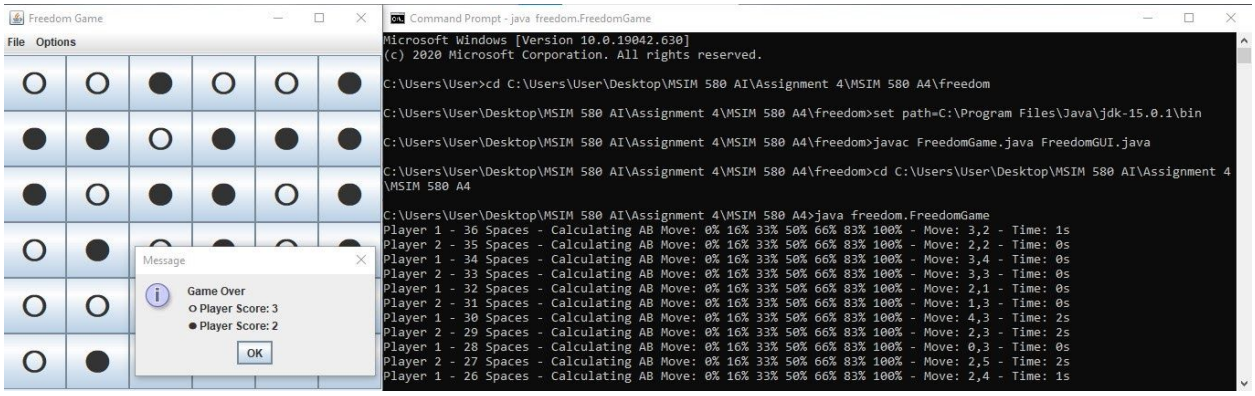
Ben Bissantz
bbiss001@odu.edu
11/24/2020

MSIM 580 Assignment Four: Freedom Game

Overview

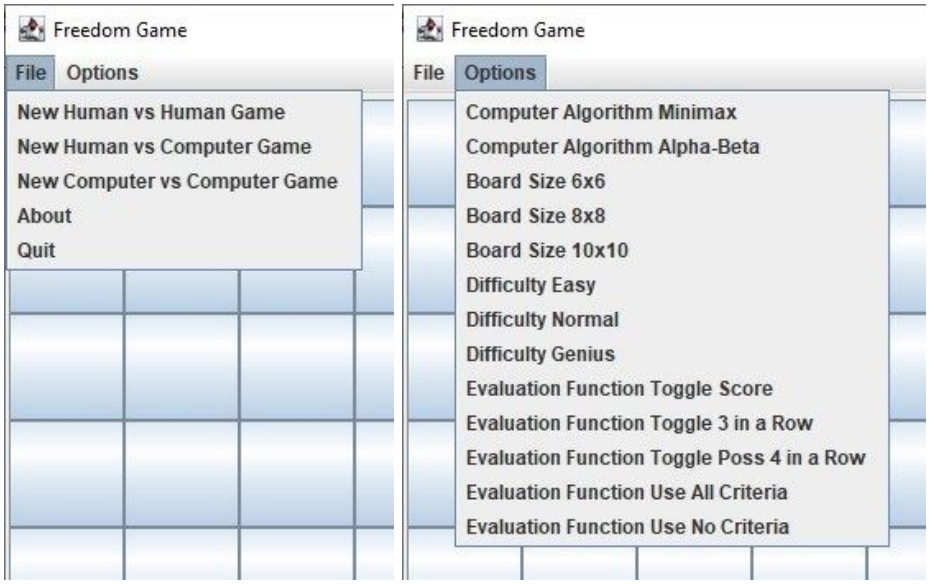
The program to implement a minimax algorithm and alpha-beta pruning algorithm to play the Freedom game consists of two classes. The main class, FreedomGame.java includes the main method which starts the Freedom graphical user interface (GUI) by calling the FreedomGUI.java class. The FreedomGUI.java class contains methods to create the GUI window, add items to the menu bar, reset the game board, toggle the active player, handle computer moves, determine if the game is over, calculate the score, and calculate the utility for a given game state. The FreedomGUI.java class also contains three methods which implement the minimax algorithm and three methods which implement the alpha-beta pruning algorithm.



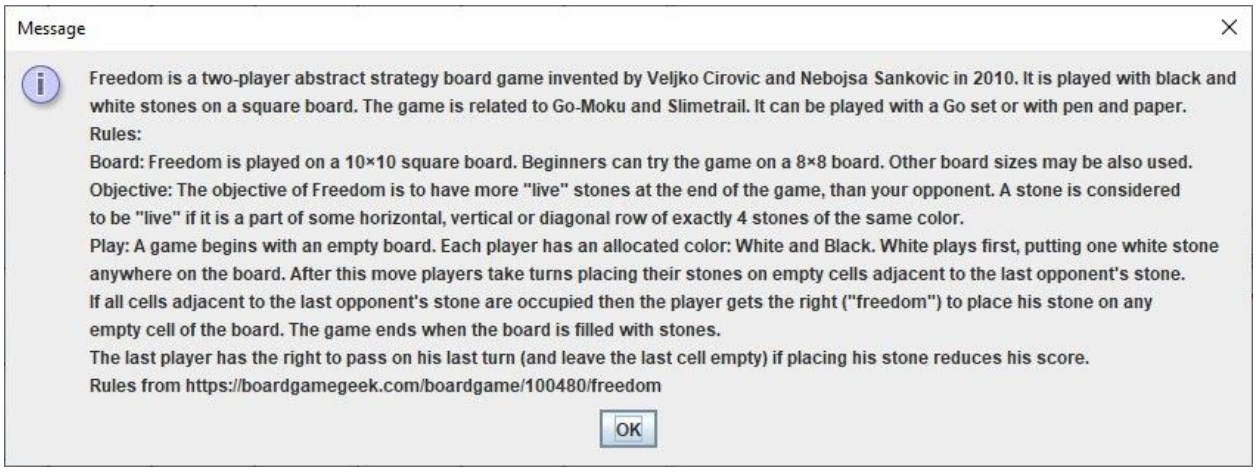
Running the Freedom Game Program from the Command Prompt

Program Execution

When running the program, the user may make selections from the options menu to change how the program is run. The user may select the board size from 6x6, 8x8, or 10x10; the minimax or the alpha-beta pruning algorithm; the difficulty from easy, normal or genius, and the evaluation function components to use. The algorithm, difficulty, and evaluation function components may be changed throughout the game when playing a person vs computer game. Changing the board size and starting a new game will clear the board. The file menu also contains an about option with information and rules for the game. When the user starts a new game the user may select a person vs person, person vs computer, or computer vs computer game.

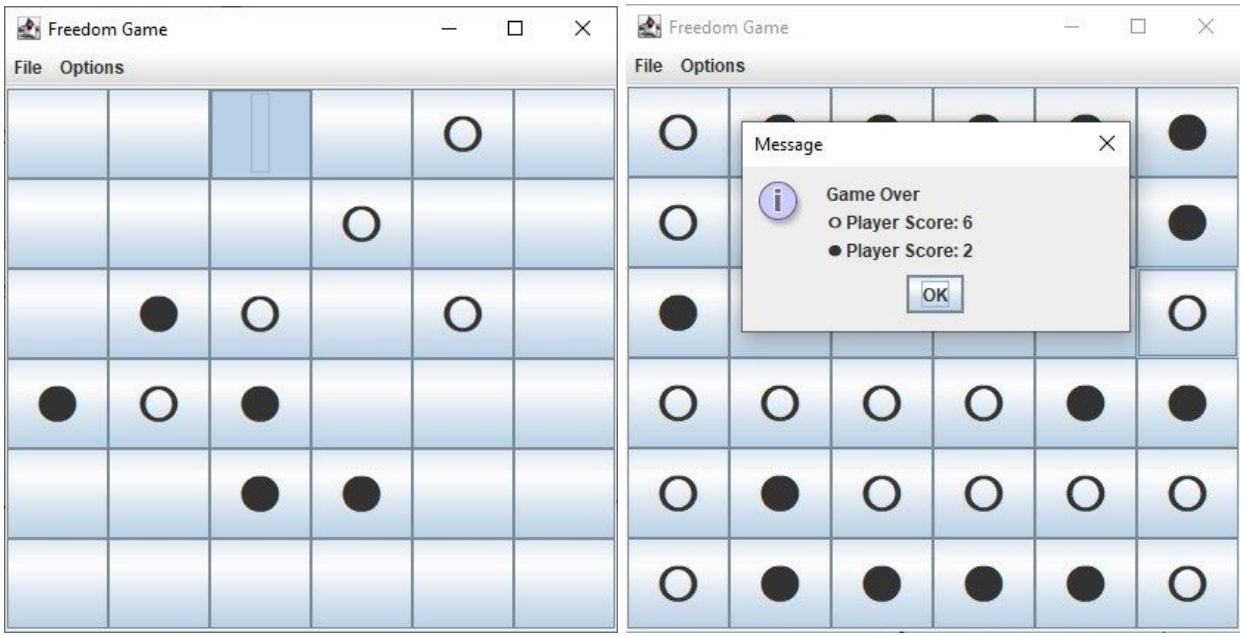


Freedom Game File Menu and Options Menu



Game Rules Displayed from the File Menu

Gameplay alternates between the two players. Human players take turns by selecting a space on the board. Computer players will calculate a place on the board to select. Depending on the difficulty and algorithm selected the computer player may take a few minutes to complete their turn. A transcript of each move is passed to the console as output. In the current implementation a computer player vs computer player does not update the board after each turn and only updates the board once the game has finished.



Example of Gameplay and Message Displaying Score

It is important to mention that because the second player has the option to pass on their final turn, the decision for the final turn is handled differently than all other turns. The final turn for the second player is automated, even if the second player is a human player. The game automatically determines if it is better for the second player to pass or play for their final move and proceeds accordingly to calculate and display the final score.

Algorithms

The minimax algorithm and alpha-beta pruning algorithm are both implemented using three methods each. Each algorithm has its own method which makes the final update to the game board and determines the search depth. Additionally each algorithm had a minimum and a maximum function which are called in an alternating order until the search depth is reached. The utility in each function is calculated by subtracting the result of the evaluation function for the idle player from the result of the evaluation function for the current player.

Search Depth

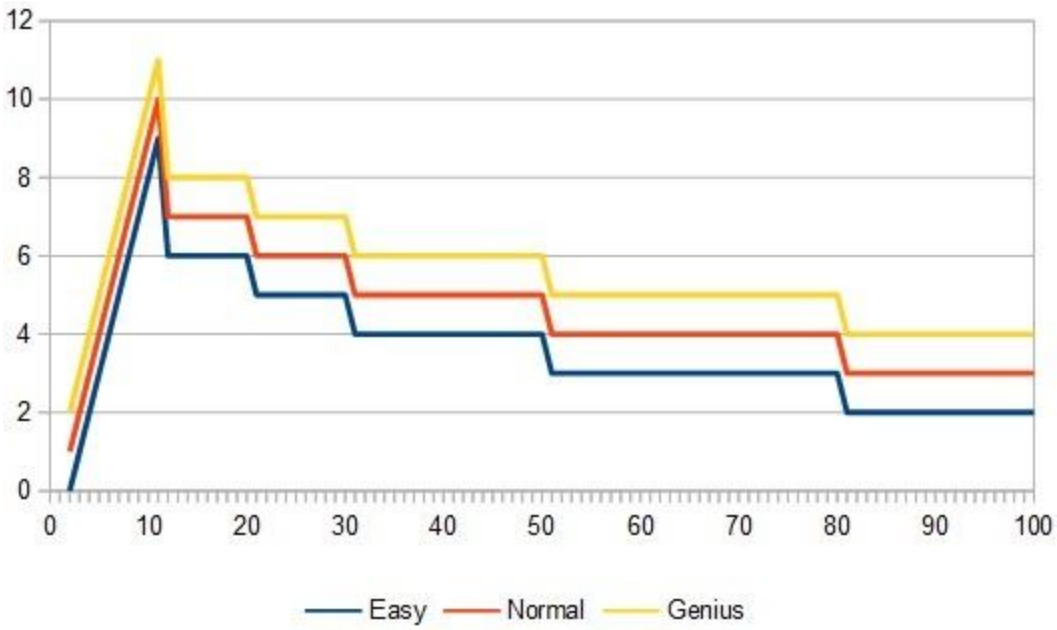
The computer player for the Freedom game has a variable search depth for the number of moves ahead to look based on the number of empty spaces remaining and the difficulty selected. As the number of remaining spaces decreases the search depth periodically increases as the computer player is able to search additional moves in a shorter amount of time. As the end of the game approaches the search depth is set to match the number of empty spaces. If the easy difficulty is selected the search depth is one less than the normal difficulty and if the genius difficulty is selected the search depth is one greater than the normal difficulty. For the

minimax algorithm the search depth is reduced by one additional level to accommodate the additional time needed compared to the alpha-beta pruning algorithm.

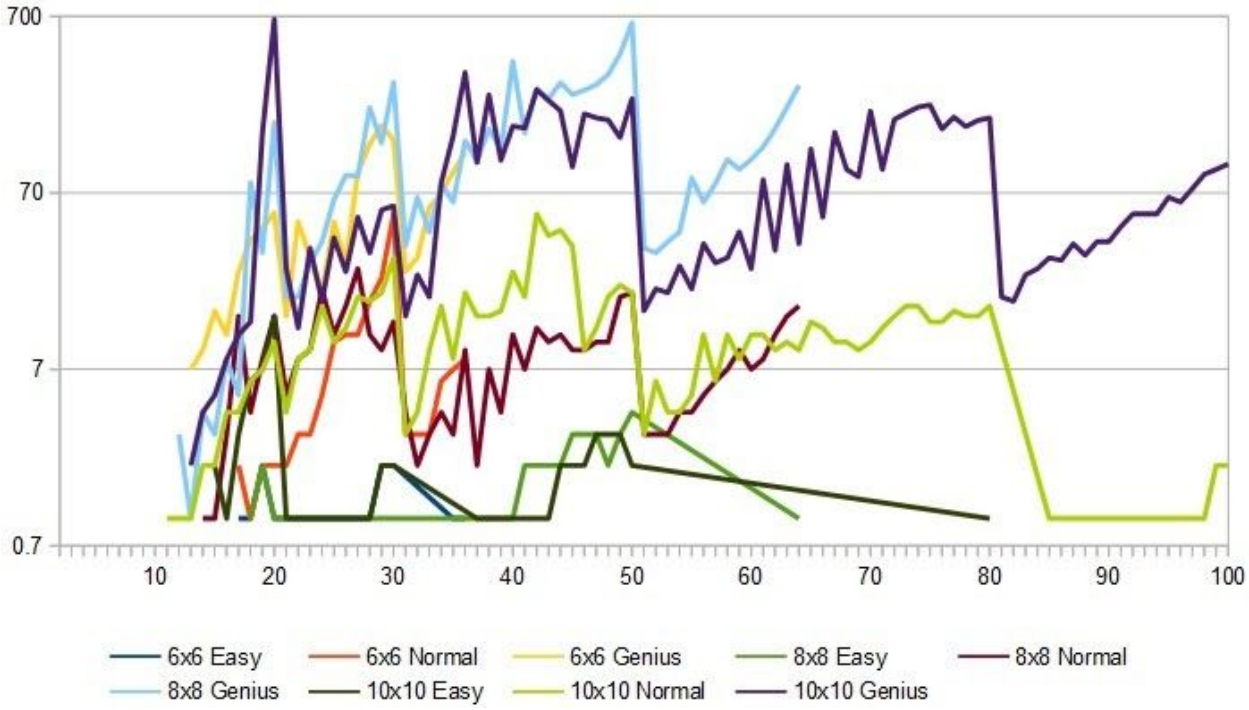
The chart below and table below show how the search depth changes based on the number of empty spaces remaining for each difficulty. The second chart below shows the performance in terms of search time for a given number of empty spaces remaining. As expected the graph shows increases in search time each time the search depth increases. Both charts and the table show information for the alpha-beta pruning algorithm search depth.

Empty Spaces	Easy Search Depth	Normal Search Depth	Genius Search Depth
2-11	Empty Spaces - 2	Empty Spaces - 1	Empty Spaces
12-20	6	7	8
21-30	5	6	7
31-50	4	5	6
51-80	3	4	5
80-100	2	3	4

Search Depth by Difficulty Level



Plot of Search Depth Compared to Empty Spaces Remaining



Logarithmic Plot of Search Time Compared to Empty Spaces Remaining

Evaluation Function

The evaluation function calculates the utility for a given game state in a similar manner to the calculation of the score at the end of the game. The score is calculated by totaling the number of locations a player has exactly four spaces in a row in the horizontal, vertical, and diagonal directions. The evaluation function then finds the number of locations where a player has an open-ended set of four spaces in a row and lastly the evaluation function finds all possible locations a player could get exactly four spaces in a row remaining for a given board state.

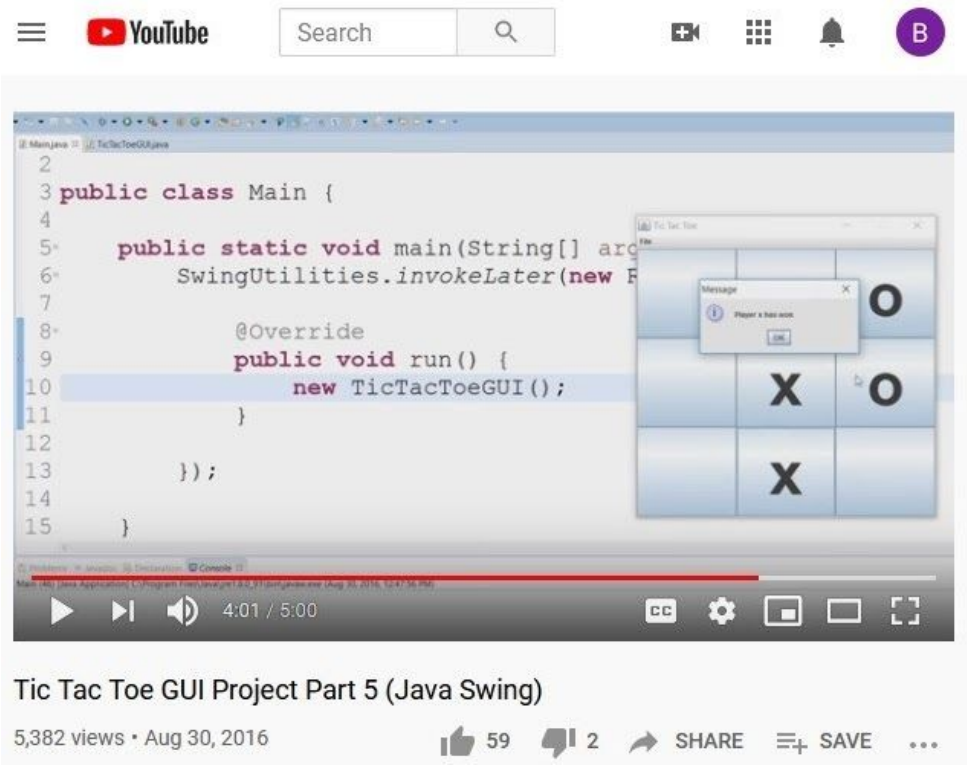
The evaluation function assigns a utility value of four for each point of the player's current score, two for each open-ended three in a row, and one for each possible four in a row. The user may toggle each of the three components of the evaluation function using the options menu. If all components of the evaluation function are toggled off the computer will perform random moves.

Evaluation Function Search Item	Utility Value
Current Score Points	4
Open-Ended Three in a Row	2
Any Possible Four in a Row	1

Evaluation Function Utility Values

Graphical User Interface

The GUI for the Freedom game was written using Java Swing. Portions of the code for the Freedom GUI have been modified from Pedro's Tic Tac Toe Tutorial YouTube Video (https://www.youtube.com/watch?v=YMeVSoNumAg). The code from the Tic Tac Toe Tutorial only stored the game state in the Strings attached to the buttons which represent spaces in the game. The Freedom game only uses the Strings attached to the buttons to display the game. The game state is saved in a two dimensional boolean array which allows the algorithms to run much faster. Additional modifications to the GUI included the ability to change the size of the game board and additional menu options for the functions of the program.



Pedro's Tic Tac Toe Tutorial YouTube Video

Results

In order to analyze the effectiveness of the evaluation function criteria several games were played using the possible combinations of the evaluation criteria at each difficulty level for each board size. The results are shown in the table below with C representing a game the computer player won and H representing a game the human player won.

Current Score	X	X	X		X			
Open Three in a Row	X	X		X		X		
Possible Four in a Row	X		X	X			X	
10x10 Easy Difficulty	C	H	H	H	H	H	H	H
10x10 Normal Difficulty	C	H	H	H	H	H	H	H
10x10 Genius Difficulty	C	H	C	H	H	H	H	H
8x8 Easy Difficulty	H	H	H	H	H	H	H	H
8x8 Normal Difficulty	H	H	H	H	H	H	H	H
8x8 Genius Difficulty	C	H	C	H	H	H	H	H
6x6 Easy Difficulty	H	H	H	H	H	H	H	H
6x6 Normal Difficulty	H	H	H	H	H	H	H	H
6x6 Genius Difficulty	C	H	H	H	H	H	H	H

Results of Human vs Computer Games Using Various Evaluation Functions and Difficulties

The only times the computer player won without one of the three criteria was when using the current score and all possible combinations of four spaces in a row criteria without the open-ended three spaces in a row criteria. Under this condition the computer player won on both the 8x8 and 10x10 puzzles with the genius difficulty. The computer was not able to win any of the games that used only one of the evaluation criteria.

The results of using different combinations of evaluation function criteria show that the three criteria used together were the most effective. The Computer player was most effective on the 10x10 puzzle and the genius level computer player was able to win on all three puzzles using all three criteria. This shows that the evaluation criteria were effective in helping the computer player to play the Freedom game.