جامعة بيرزيت

BIRZEIT UNIVERSITY

Faculty of Engineering Technology

Electrical & Computer Engineering Department

ENCS3340, ARTIFICIAL INTELLIGENCE

Project Report

Prepared by:

Name (1): Faten Sultan **ID Number:** 1202750

Name (2): Mohammed Abu Shams **ID Number:** 1200549

Instructor: Dr. Yazan Abu Farha

Date: 18.4.2023

Section: 1 && 2

Program Implementation

Our project talks about implementing a game using min-max algorithm this game called Magnetic Cave. The game involves two players playing versus each other the two players may be played by manual or versus the computer. And so for the computer to play we implemented a function that tries in somehow play in the best way.

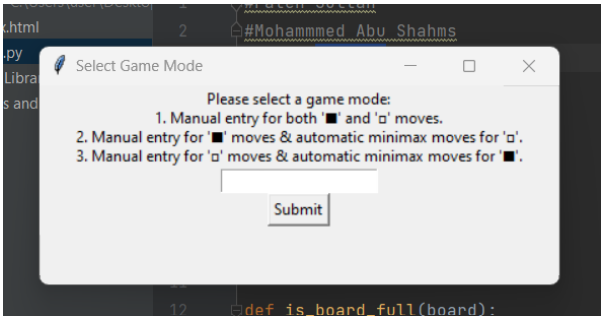
The rules of the game is simple, both players take turns putting their bricks in empty squares. The first player to get 5 of their marks in a row (up, down, across, or diagonally) is the winner. When all 64 squares are full, the game is ended.

The game can be considered as a zero sum game in which one person's gain is equivalent to another's loss, so the net change in wealth or benefit is zero

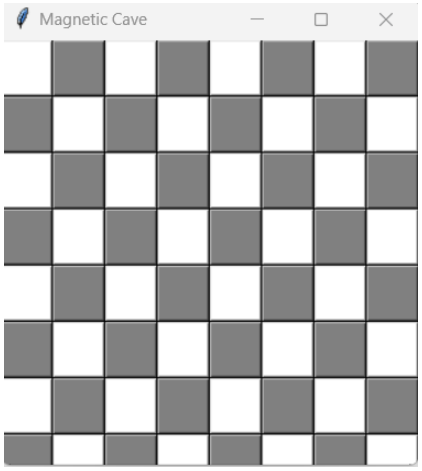
We used Python language to implement our program using PyCharm 2022 Edition.
And for our interface we use Python binding to the Tk GUI toolkit.

How the Program Runs

First, we have to choose in what game mode you want to play versus the computer or another person.



And after you chose the mode of the game and press submit the window will disappear and the board will be appeared in front of you



We implement this board in a way that if you choose either mode 2 or 3 the computer will play with you and will try as much as it can to minimize your winning and maximize its winning using function will be discussed later.

Main Function Used in The Project

1. create_board : we use this function to implant an empty 8*8 board
2. is_board_full(board): check if the game board when does it become full by checking each cell and returning False if any cell contains a space (' ') indicating for empty cell
3. is_valid_move(board, row, col): this function check if whenever the movement is valid by verifying if the movement meet the specifications –conditions that should be taken into consideration.
4. make_move(board, row, col, player): this function makes the move by placing the black and white bricks at the right position only if the movement is within the rules
5. check_win(board, player): this function check if a player has won the game by examining all possible winning configurations on the board.
6. get_best_move(board): This function uses the “minimax” algorithm to find the best move for the computer when the player plays versus the computer based on the current state of the game board.
7. minimax(board, depth, maximizing_player): This function implements the minimax algorithm for determining the best move for the computer player. It recursively evaluates the game board at different depths to determine the optimal score for the current player. This function maximize the evaluation value, while the minimizing player tries to minimize it.

Data Structure Used

We used two dimension array to create 8*8 board. Each element of the list represents a cell on the board and can hold the values '■', '□', or ' ' (empty). The game board is updated and manipulated throughout the code to keep track of the game state.

By exploring all possible moves and their outcomes, the mini-max algorithm evaluates the game board and determines the best move for the computer player at the current state. The get_best_move function utilizes the mini-max algorithm to find the move with the highest score and returns it as the optimal move for the computer player.

The Heuristic Of My Code

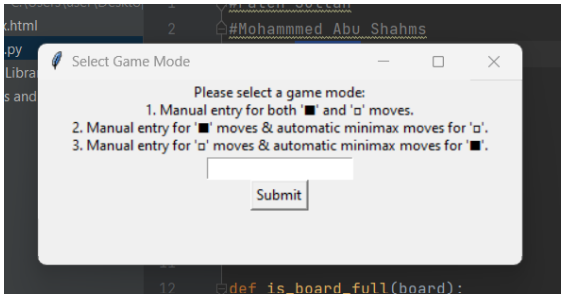
The heuristic code in our project present the implementation of mini-max algorithm. This algorithm is used to determine the best move done by the computer Ai based on the current state in the chess board. It frequently evaluate each possible path and ended up in a certain depth it assigns the score for each path. And ended up playing with the path that has the highest score.

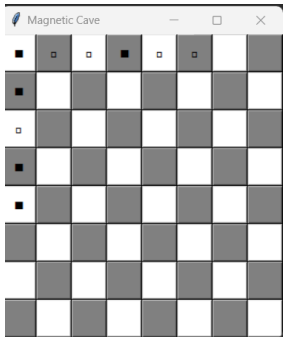
The aim purpose of using the implementation of this heuristic code is to choose the optimal and the best path, by considering all possible moves and their potential outcomes, the minimax algorithm ensures that the computer player chooses the move that leads to the best possible outcome or minimizes the worst-case scenario.

My Tournament with the AI

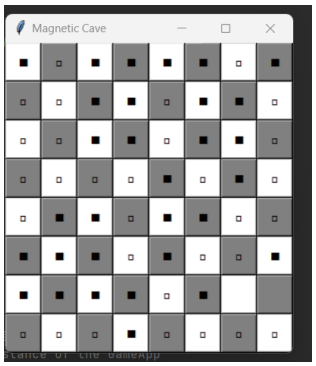
I entered a game versus the computer AI to see the result between each other we played until the game ends

In the first game: I run the code and choose a number from one of the three game modes, and we begin the game

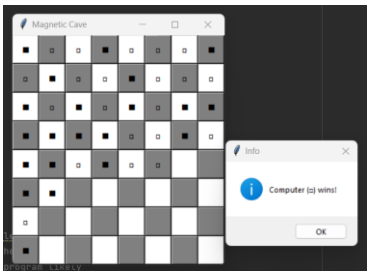


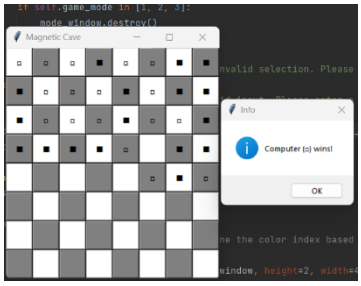


I choose a place to put the brick and we continue playing until we tie
As you can see the computer prevent me from winning by putting its white brick through my winning row.
The computer prevented me to win for many times as you can see in the next graph:

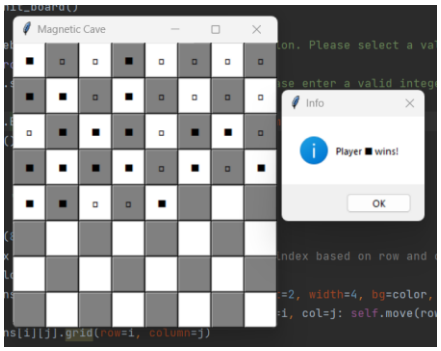


We return to play for several times against each other and in each time we ended up with it beat me as you can see in the graph below:

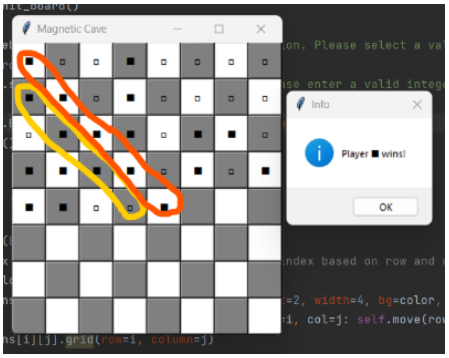




For this time I finally beat them



The only reason for my winning is that I beat it in two winning path. As you can see, it put their brick in the orange path to prevent my wining however since I am I have more than way to win…. I won.



I played another time to see how will win and this is the result

We both played and tried our best in a way that if we can't win at least we can minimize our opponent winning. But however in this situation it beat me and the computer won