World of Snails

(Artificial Intelligence)

Report

NAMAL INSTITUTE

Submitted to: **MR. FAISAL ZEESHAN**
Submitted by: The Dreamers (**G-3**)
Dated: December 2nd, 2020.

Abstract:

Our **World Of Snails** is a two-player Computer Game. On one hand, We have **Human** (Human Player) and on the other, we have **Bot** (Artificially intelligent Computer Player). Both Players fight over a limited area and the 1st Player that successfully able to occupy a little more than half of the available area wins the game. It is simple to play and fun to watch.

Figure 1: Garry-Kasparov Vs Deep-Blue (IBM-Computer)

The development of Artificial intelligence has not only revolutionized the world of computing in general but especially affected Game Development. The Artificially Intelligent games are considerably difficult to develop as they involve complex computational techniques. It is important to note that the development in the field of AI was more of an Evolutionary Process than a Revolutionary process as it has been around from many decades. The AI shocked the World when the Deep Blue (A chess-playing computer developed by IBM.) managed to beat a reigning chess grand master Garry Kasparov (It was the first time a Chess Champion was defeated by a Computer), It was a huge success for AI developers. It is also important to note that it was around 25 years ago and now the AI has considerably advanced. We believe, Our World of Snails (game name) is just like drop in the Ocean of the exciting World of Game Development that uses AI.

Table of Contents

A. INTRODUCTION:

1. Overview:

The World of Snails (name of the game) is a two-player turn-based board game. One player is Human and the other is Bot. This game can be played in two modes, i.e. “Human vs Human” and “Human vs Bot”, The current version of our game is “Human vs Bot”. A Player in the is represented by a Snail of a specific color and the Snail moves (takes turn) on a different grid square, then it is followed by a trail of its respective slime of a specific color that is compatible with the Snail. The player that manages to occupy the majority of squares wins the game.

2. User Manual:

➢Terminologies:

- **Bot:** AI Agent (Artificially Intelligent Computer Player)
- **Human:** Human Player
- **Grid Square:** Single-cell in the 2d grid.
- **Grid World:** Area covered by the 2d grid
- **Slime:** It is a mark left by a Player (Snail) to signify the occupation of a Grid Square.
- **Trail of Slime:** It signifies the occupied area.
- **State:** It is the current state of the game and the game could generally in Continue state or Winning state.

➢Requirements / Installation:

The World of Snails was developed using Python 3.8. It is also hugely dependent on the Arcade (Python Game Framework) for its working because it was build using Arcade. It is necessary to install Python and then install Arcade. It is recommended to use the latest or close to the latest version of Python because Arcade may not work properly with some older versions of Python.

➢Objectives / Rules:

The major objectives of the World of Snails are listed below:

1. The primary objective of the Game is to occupy more squares than the opponent player to Win the game.
2. The World of Snails is a turn based 2 Player Computer Game. It is a board game in which a player can play with either or both of the following 3 strategies:

i- **Attacking Strategy:** While attacking, A Player 1st try to restrict the moment of opponent player to a limited area by stopping its way, after doing that He starts occupying the rest of the empty squares.

ii- **Defensive Strategy:** While Defending, A Player just tries to occupy more squares than its opponent.

iii- **Hybrid Strategy:** A Hybrid strategy is the mixture of the above two strategies in which a Player not only tries to occupy more squares but also at the same time tries to restrict the opponent's Player's movement.

3. On each turn, a Player can move horizontally or vertically.
4. When a Snail moves, it leaves behind a trail of Slime as a sigh of his occupation.
5. When a Snail moves to an Empty Square, 1 is added to his scores.
6. When a Snail tries to move on its own slime, no scores are awarded and snails slips to the farthest place on the trail of slime in the direction of the move.
7. A Player cannot move on the opponent's Slime, This could be very valuable in some cases as this constraint can be used to block the opponent's way and restrict him in a limited space.
8. The winner is announced when all the grid squares are captured or when the majority of grid squares are captured by either of the players then the one with the highest score is declared as the winner.
9. When a Player plays an illegal move then 1 is deducted from his scores as a penalty and He also loses his turn. The turn is also lost when a Player clicks on his own snail but scores stay the same, as this is a meaningless move.

➢Constraints / Assumptions:

The Constrains are listed below:

1. Board size is fixed at 10 X 10.
2. A Player can occupy only a single empty grid square at a time.
3. The area of the Player can be blocked by its opponent.
4. A Player can lose his turn by playing an illegal move.
5. The Human will always take the 1st turn.

➢User Interface (UI):

The UI of World of Snails is not only cool and Beautiful but it is easy to use as well. The following are the screenshots of some of the important views of the screen:

a. Welcome Screen:

Figure 2: Well Come Screen

B. View of the Game at Start:

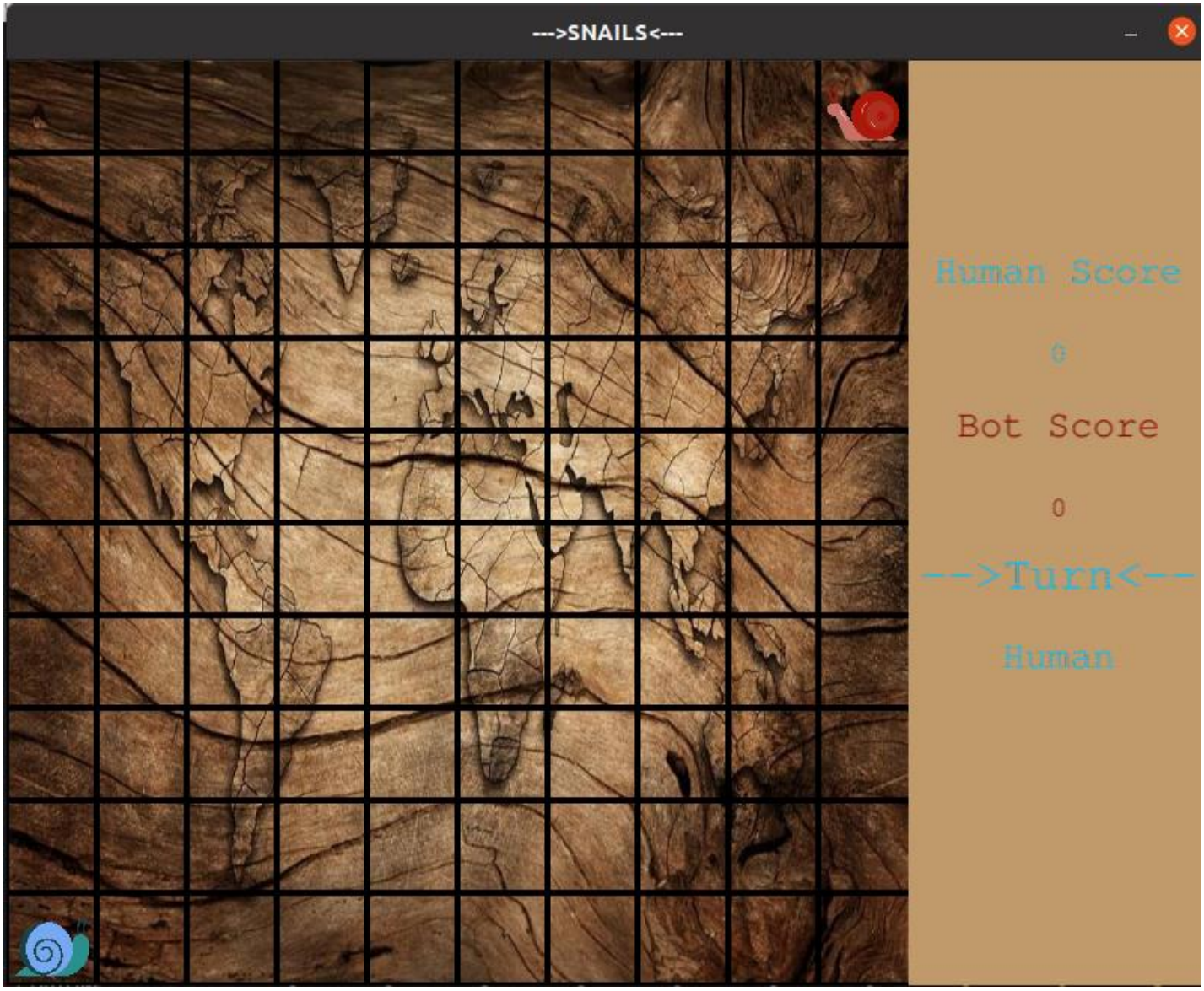

Figure 3: View at the start of the game.

C. View of the Game during the game:

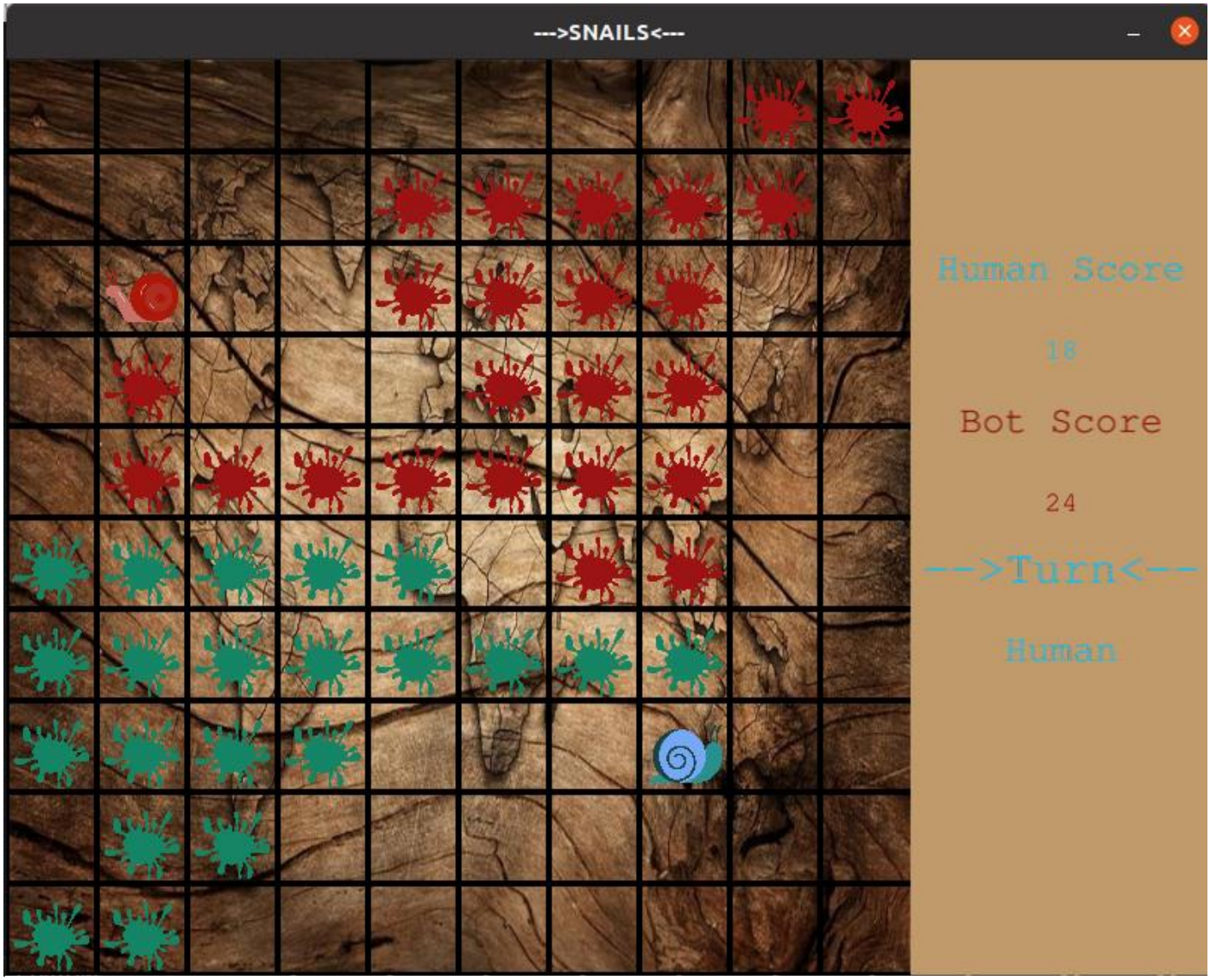

Figure 4: View in the middle of the game

D. View of when Human Wins:

Figure 5: View when Human Player Wins.

E. View of when Bot Wins:

Figure 6: View when Bot (Computer Player) Wins.

F. View when there is a Draw:

Figure 7: View when there is a Draw.

➢Illegal Moves:

The Player is penalized (1 is subtracted from its scores) for playing the following illegal moves.

1. linking on the opponent's slime.
2. Clicking on opponent's snail.
3. Clicking on the area outside the grid world.
4. Jumping a box is not allowed, A Player has to chick on the immediate next grid square from its current position in any direction.
5. Trying to move diagonally.

Note: Clicking on your own snail is not considered an illegal move but no change will occur in the Grid or the scores and the turn will be lost.

➢Legal Moves:

1. Clicking on an empty box next to your current location in a vertical or horizontal direction.
2. Clicking on the box with your trail of slime next to your current location in a vertical or horizontal direction.

➢How to Win:

The human players can use two strategies to give a tough time to AI Player and even or win which are:

1. Try to occupy more squares than your the opponent.
2. Also try to block the way of opponent player to restrict him from occupying more squares.

The 1st Player that manages to gain 51 points wins the game or the Player with a greater number of points after the Grid is completely filled wins the game. A tie (Draw State) is also possible if both players have the same points after the Grid is completely filled.

B. METHODOLOGY AND IMPLEMENTATION:

1. General flow of our code:

The general flow of our code is represented by the following flowchart:

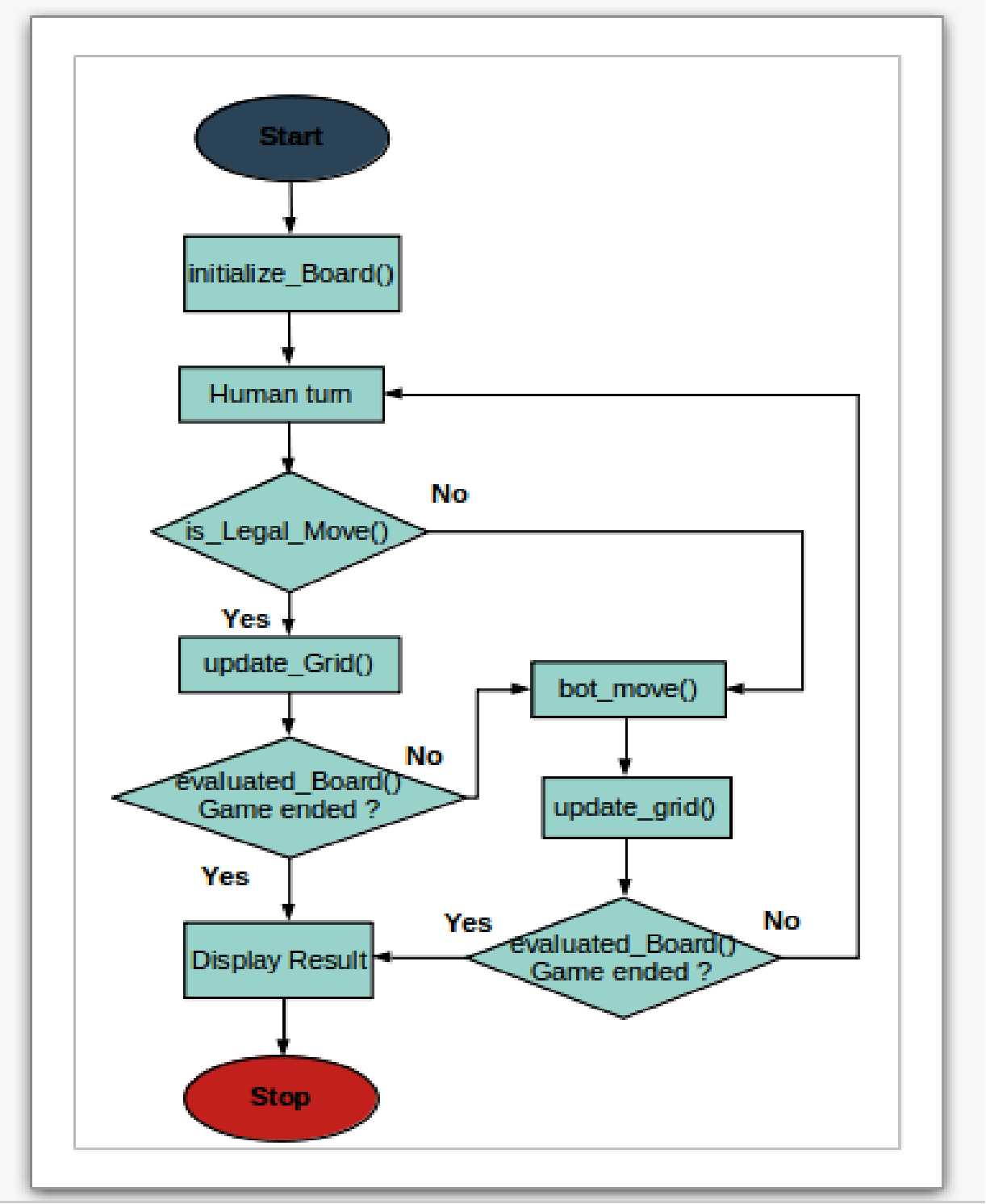

Figure 8: Flow Chart of the code.

2. Functions:

The important functions of our code are listed below:

➢InitailizeBoard () :

This function is responsible for the making of the back-end 2D matrix for the front end Grid World. It also places the players at their initial positions. The Human is placed at the bottom left corner of the screen and the Bot is paced at the top right corner of the screen.

➢EvaluateBoard () :

This function is responsible for the analysis of the **current state** of the game at any point of time during the game. It helps us to determine the winner of the game and also tells us the game is in continue state or not. Upon calling, it returns integer values. It returns 0 for draw state, 1 for Bot win, and -1 for Human Win. If it returns **None,** This means the game is in a continue state.

➢IsMoveLeft () :

This is a Boolean function that checks either an empty box is present on the board or not.

➢Bot_move () :

The Bot plays his moves using this function in such a way that it manually moves the Bot in all four directions and checks where the winning chances are maximum using the heuristic function. Where it gets the maximum winning chances, it moves the Bot in that direction.

I. Heuristic ():

Our heuristic function takes a board as input and returns winning chances at the current location of the Bot. We are getting winning chances by applying three conditions in this function:

1) Firstly, we are checking visited boxes by the bot and adding their count to winning chances.
2) Secondly, unvisited boxes are to be checked in all four directions around the bot and the max of them will be added to winning chances.
3) Lastly, Bot's position is checked whether it is in the central area of the board or not. If it is, then winning chances are incremented by a value of 10. The Bot plays his moves using this function in such a way that it manually moves the Bot in all four directions and checks where the winning chances are maximum using the heuristic function. Where it gets the maximum winning chances, it moves the Bot in that direction.

II. Minimax ():

In our minimax function, implementation is divided into three parts:

1) Firstly, we are checking terminal conditions means base cases for recursion.

2) Secondly, the AI agent (Bot) is supposed to take its turn.

3) Lastly, the Human (Human Player) is supposed to take his turn.

In both the turns, we are getting child boards on every move and then minimax is recursively called on every child board. Here we are using a helper function “childBoard()” which returns the list of child boards according to the current location of Agent either it is AI or Human.

Our logic to make AI agent (Bot) intelligent through minimax Algorithm is that when the terminating depth is reached in the recursive call then we have to call “heuristic()” function on every leaf nodes means child boards and finds where the heuristic returns the maximum value and goes in that direction.

➢ is_Legal_Move () :

This function is responsible to validate the move of a human player. It checks the move is legal or not. Upon calling, It returns True or False for legal and illegal moves respectively. It has three important checks:

1) for moving the Snail onto the opponent's Snail or Trail of Slime.
2) for clicking on an area out of bounds(outside the Grid World)
3) for playing a move by-passing (jumping over) an empty Grid Square.

➢ update_grid () :

This function is responsible to update the Grid World by updating the backed 2D matrix after a legal move by human players. It in a way helps the Snails to move around and leave the slime.

C. ADDITIONAL NOTES:

➢ Heuristic Search:

The **Heuristic search** is used to search for an optimal solution. This method is not always finding the best solution but finding an acceptable or good solution. It is a technique to solve the problem faster rather than other classic methods. It solves the problem in a reasonable time that's one of the reason is used in Artificial Intelligence.

➢ Minimax Algorithm:

The **Minimax Algorithm** is a backtracking or recursive algorithm. It is used in the decision-making of AI-based games. This algorithm helps Computer Player to make the best possible moves in its current state. The Minimax explores the game tree in a depth-first search manner. One part of the minimax acts like a minimizing player and the other as a maximizing player. This algorithm uses recursively searches the game tree to find the most optimal move.

➢ A* Algorithm:

The **A* algorithm** uses Euclidean distance and finds the shortest path from the current location to the immediate location. In this game, it can be used in such a way that it finds the distance from the bot snail to immediate position and also from the human snail. If the bot's distance is less than humans' distance then it will be adding more winning chances in that direction otherwise it will try to immediate position other than this.

D. REFERENCES:

1) Lab Lectures (https://nulms.namal.edu.pk/course/view.php?id=503)
2) https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
3) https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-2-evaluation-function/?ref=rp
4) https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/
5) https://www.youtube.com/watch?v=trKjYdBASyQ&vl=en
6) https://en.wikipedia.org/wiki/Minimax
7) https://www.youtube.com/watch?v=l-hh51ncgDI
8) https://arcade.academy/index.html
9) https://docs.python.org/3.8/
10) https://arcade.academy/examples/starting_template.html
11) https://www.geeksforgeeks.org/a-search-algorithm/
12) https://arcade.academy/tutorials/bundling_with_pyinstaller/index.html