2D SNAIL GAME

December 6, 2021

ARTIFICIAL INTELLIGENCE LAB
CS-340
Mr. Faisal Zeeshan

Acknowledgement

We would like to express the deepest gratitude to Mr. Faisal Zeeshan, our Lab instructor, for his help, guidance and encouragement in our work and far beyond. Without his numerous suggestions and immense knowledge, this 2D game would never have been completed. Without his open-mindedness, our hard-work would not have been so rewarding. We appreciate him who gave us an eye opening opportunity to develop this 2D snail game using the artificial intelligence.it is highly appreciated that he gave us extra time for discussing our problems.

Table of Contents

List of Figures

Abstract

Artificial Intelligence (AI) is a growing industry in Computer Science. And AI-game development is the most significant part of it. Artificial Intelligence has revolutionized the game industry. Instead of playing game with humans, it is possible to play advanced games with computer. We enable computers and machines to be intelligent at an extent level so that they may play with humans using developed algorithms like mini-max, heuristic search and path finding trees. The main aim of AI-agent in each game is to stop opponent to take less points and tries to get maximum points for itself. This report is also about an AI based game named as "2D Snail Game". Mini-max and Heuristic search algorithms are used to develop this specific game. This game can be played between two humans or an human with computer (AI-agent). AI-agent will try to cover the maximum point and stop the human to get less points. In short, AI-agent will follow all the possible paths where it can get maximum.

1 Problem Statement:

In the course of Artificial Intelligence, we had to develop a game named as "2D Snail Game". First We have to develop 2P game means two humans can play with each other according to their own intelligence. Then we have to develop an AI-agent who can play with humans intelligently. This game was already developed but we have to redevelop it using our AI algorithms.

2 Introduction:

Artificial Intelligence (AI) games have revolutionized the game industry. Now a days mostly games are built and developed using AI agents. Computers are trained to play against human's intelligence. Based on algorithms and human's psychology, AI-agents of games take decisions and try to win over humans at each step and if it is impossible to win for them then they try to decrease the chances of winning for humans. This report is about the development of 2D snail game. This game is also developed on the basis of artificial intelligence. 2D Snail Game is a two player game and players may be two humans or an AI-agent.

3 Game Overview:

Game development was an amazing experience for all of us in artificial intelligence Lab course. We have to develop a 2D Snail game. In the first phase, simple two player game has been developed. it means only two humans can play the game with each other at a time. This game was developed on the grid of 10 rows and 10 columns.
In the next phase of development, we have to implement artificial intelligence where human can play game with computer (AI-agent). This was much difficult and logical than 2P game. This phase was totally developed on the basis of artificial intelligence. In this part, we have to make computer so much intelligent that it should always win from human.

4 Requirements to Play 2D Snail game:

This game is developed in the latest version of Python using arcade library which is commonly used to develop front-end of 2D games.
The given below things are required to play the game;

1. First you gave to install Visual stdio code.
2. Then you have to install Python latest version and integrate it with VS Code.
3. Then you have to pip install arcade library to run the given code to play 2D Snail game.

4.1 Game Terminologies:

Grid:	2 Dimensional square having 10 rows and 10 columns
Grid Cell/Box:	A single square within a grid.
Board:	Front-end grid of 10 rows and 10 columns.
Sprite:	Object representing any player in game.
Splash:	Object to represent the covered area of a player.
AI agent	Computer which is enough intelligent to play with human

4.2 Rules and Regulations:

2D Snail game is a two player game. Two players can be either both humans or an human with AI-agent. There is a specified grid and both players have to cover maximum grid to win the game. When a player moves from one position to other, it leaves a specific splash which tells other player that this area is covered and it can't move to that position. The player who covers maximum paths will win the game.

1. 2D Snail game has total hundred number of grids.
2. Initial position of 1st player will be at the bottom left but inside the specified grid.
3. Initial position of 2nd player will be at the top right but inside the specifies grid.

4. A player can move one step in single turn and step can be towards left, right, up or down.
5. A player can't move diagonally.
6. A player can't jump from one position to other. Instead of jumping, it has to cover all the position to reach that specific position.
7. A player can't move to position or splash of opponent.
8. A player can move on its own splashes but it will slip to last splash towards moving side.
9. A player can use mouse to click to move its sprite.
10. A player can lose its turn by clicking on opponent's position.
11. Both players have to cover maximum position to win the game.
12. When there is be no position or box of grid, the game will be ended.
13. Each legal turn and filling of empty position will add a point to total points of that player.
14. A player with maximum number of covered position will win the game.

4.3 Limitations of Game:

There are certain limitations of this 2D Snail game;

1. This game can be played on the grid having 10 rows and 10 columns.
2. You can't play this game on mobile because this is a computer game.
3. You have to install above given requirements to play this game. Python language is compulsory for this game.
4. Player or sprite can't jump from opponent's sprite to move from one position to other.
5. If any player tries to cover or click outside of the grid then its turn would be wasted.

6. Sprite can cover only one box near to its four directions.
7. You can't play with keyboard. You have to use mouse to move your sprites in the game.

5 Literature View:

The main part of the game was the development of AI-agent. The literature view of the game is the ideas of implementing the AI based game using different types of artificial intelligence algorithms. This view differentiates between natural or human intelligence and artificial intelligence. AI based games are generally played between an human and a computer. Today, most of the games are developed on the basis of artificial intelligence.
Here are some algorithms which is used and can be used to develop this 2D Snail game;

5.1 Finding the best path:

Computer plays game itself in AI based games using its own artificial intelligence. Movement of AI agents are very important and cautious. Like humans, AI agent should take care the rules and regulations of the game. It should not violate any rule and movement.
For example, AI should not move to the position of human and should not move outside of the defined grid.
Winning is the first priority for every player who plays the game. A player who plays with intelligence whether it is human or AI agent. The most difficult task or step in any game is to find the path from which a player can easily win or reduce the chances of winning for opponent. AI agent finds the best path where it defends the attack of human. Similarly AI agent checks all possible paths and finds nd chooses the best path through which AI bot or agent can win the game easily. AI agent should be clever and intelligent that it should be attacked by human at any stage rather it should attack on every step.

5.2 Decision making on the best path:

Once the AI agent finds all the possible paths where it can win. But sometimes it has to make decisions at every turn of human. It takes decision

that whether it should play attacking or defensive game. And it updates the possible paths at every decision which leads it to the winning state.

5.3 Minimax:

Most AI based games are self developed means they use the algorithms of artificial intelligence and f0r the betterment and to increase the winning chances for AI agent, we use search tree based algorithms and minimax is one of the best algorithm being used to develop all AI based game.
In an n-player game, generally a two-player game, a minimax algorithm is a recursive method for deciding the next move. Each position or condition of the game has a monetary value. A position evaluation function is used to calculate this value, which reflects how good it would be for a player to attain that position. The player then makes a move that maximises the position's minimal value as a result of the opponent's probable next moves.[1] Below given picture is how minimax algorithm works; [3]

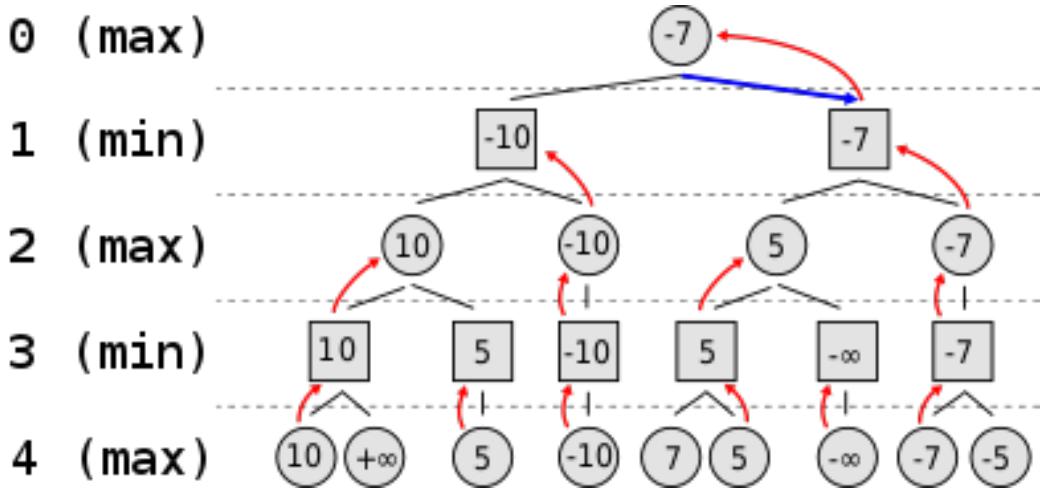

Figure 1: How minimax algorithm work?

5.4 Heuristic:

A heuristic is a strategy for solving a problem quicker than traditional approaches or for estimating a solution when traditional methods fail. We frequently exchange one of optimality, completeness, accuracy, or precision

for speed, therefore this is a form of shortcut.[2] Heuristic is generally used in humans because they can think limited. Heuristic algorithms are often used in AI to make computer intelligent to find an approximate solution instead of an exact solution. In heuristic algorithm, AI does not find all the possible path to the winning states, rather instead of going depth, it finds paths till a given level and tries to stuck the human and decreases the chances of winning for human and increase the chances for itself.
Heuristic Search algorithm use different kinds of techniques in artificial intelligence. it is impossible to define all the techniques but given below is the list;[4]

- Hill Climbing
- Stimulating Annealing heuristic search
- Breadth-first search
- Constraint Satisfaction problem

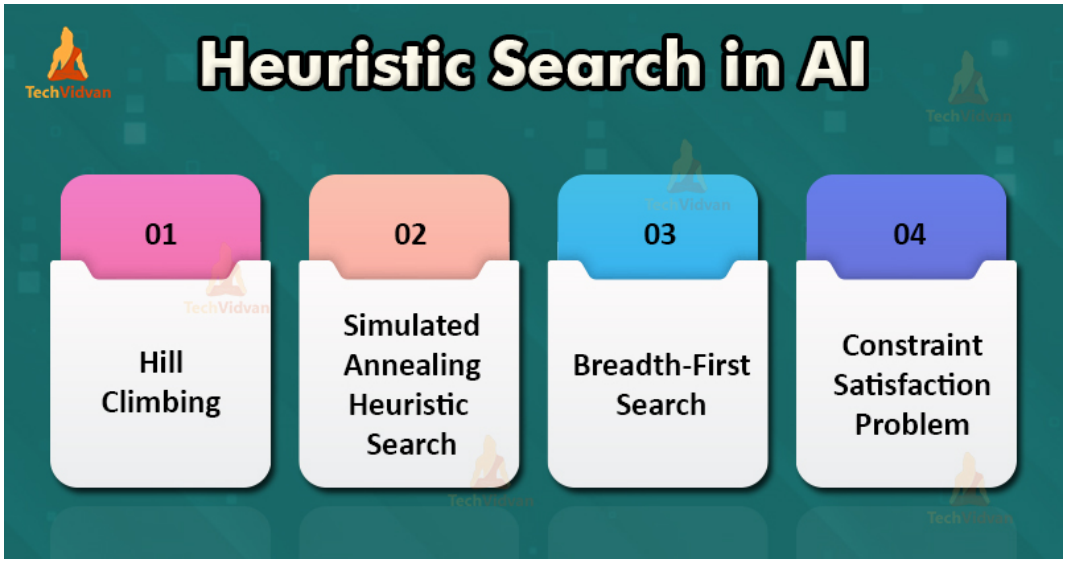

Figure 2: Techniques Used in Heuristic Search

Above mentioned algorithms and techniques are used in the development of 2D Snail game. But for better results, we can use more algorithms like Alpha Beta Pruning, A* algorithm etc.

5.5 Alpha Beta Pruning:[5]

A modified variant of the minimax method is alpha-beta pruning. It's a way for improving the minimax algorithm. Pruning is a strategy for computing the proper minimax choice without inspecting each node of the game tree.Alpha-beta pruning may be done at any level in a tree, and it can occasionally prune the entire sub-tree as well as the tree leaves.
There are two parameters which can be defined as;

- Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -.
- Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +.

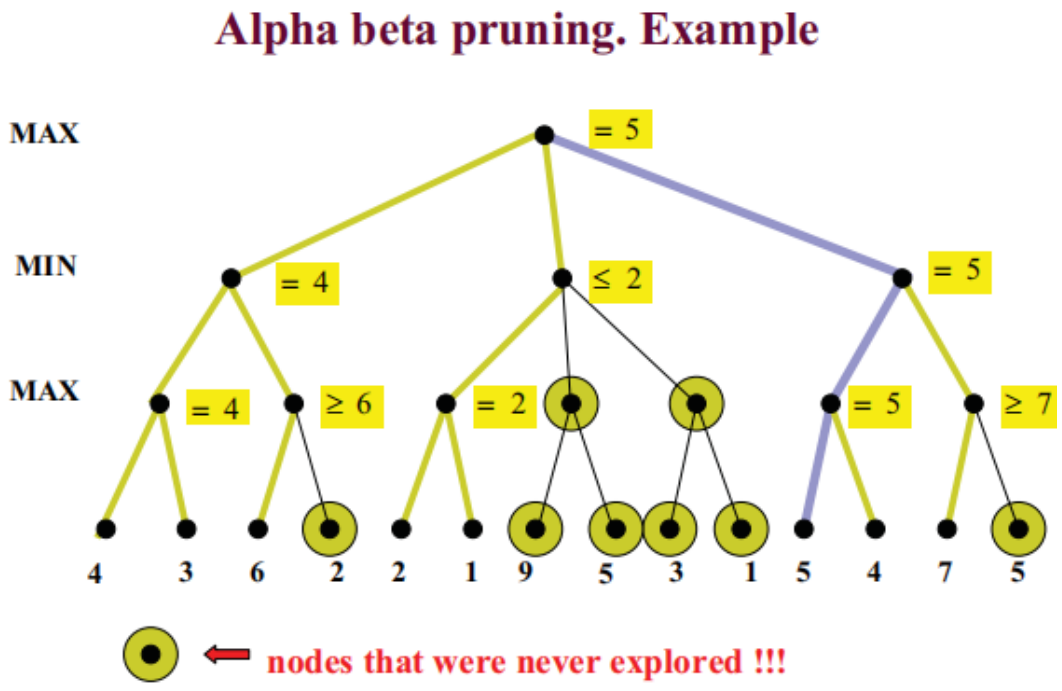

Figure 3: Alpha Beta Pruning example

5.5.1 Limitations of Alpha Beta Pruning:

- It does not solve all the problems associated with the original minimax algorithm.
- Requires a set depth limit, as in most cases, it is not feasible to search the entire game tree.
- Though designed to calculate the good move, it also calculates the values of all the legal moves.

5.6 A* algorithm:[6]

The A* Search algorithm is one of the most widely used path-finding and graph traversal techniques. Unlike other traversal techniques, A* Search algorithms have "brains." What this means is that it is a sophisticated algorithm that distinguishes itself from other algorithms.It's also worth noting that many games and web-based maps employ this approach to effectively locate the shortest path (approximation).
Dijkstra is a special case of A* Search Algorithm, where h = 0 for all nodes.

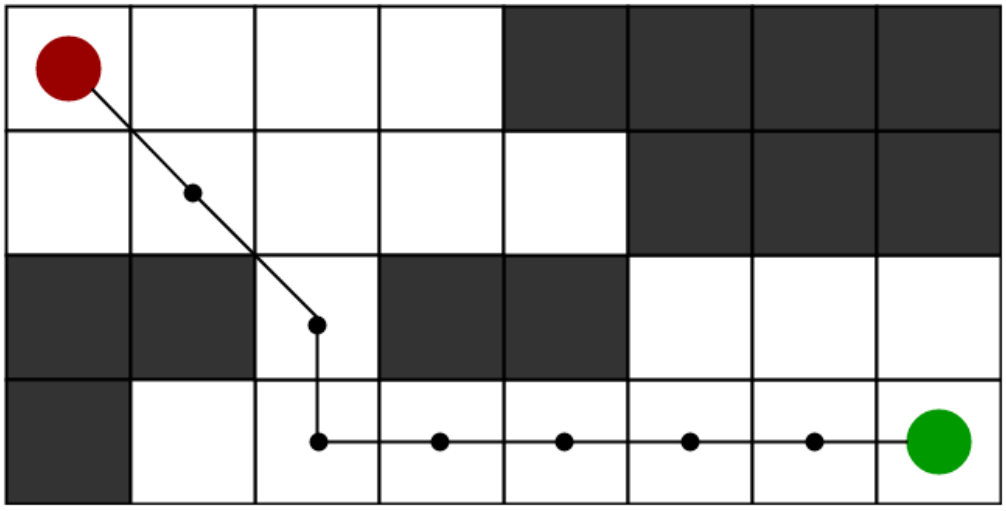

Figure 4: A* search algorithm example

5.6.1 Limitations of A* algorithm:

A* Search Algorithm doesn't produce the shortest path always, as it relies heavily on heuristics / approximations to calculate – h.

6 Methodology and Implementation:

Once a problem is given, its first step is to define the problem. What is real problem or what we have to develop? and how to find the optimal solution for that specific problem. Here it was given a project of developing 2D Snail game using artificial intelligence. When the problem has been identified then next step is to adopt optimal method and techniques to solve the problem. Before implementation, we have to make plans because when the plans are cleared then it is easy to implement the project. On the other side we have to set up specific tools on which the project is to be implemented.
Following methods and tools are used to develop 2D Snail Game;

6.1 Visual Stdio Code:

To develop 2D snail game we have to write code. For this we had installed Visual stdio code. Visual Studio Code is a lightweight yet capable source code editor for Windows, mac-OS, and Linux that runs on desktop. It features JavaScript support built-in, as well as a large ecosystem of extensions for other languages (such as C++, C, Java, and Python).

6.2 Python and Arcade Library:

Python is a programming language which is almost being used in every field of computer science. Python is used for web development, game development, machine learning, implementation of GUI's and to solve scientific and numerical problems. To develop this game, we selected python language. Python has open source libraries. Arcade is easy-to-learn python which is commonly used to develop 2D games. To use this library we have to install it using pip install command in VS code IDE of python. Once the library has been installed then you have to make a folder and a file in it with the extension of ".py" which means that we are writing the code for python. Once you start using the file of python, it is ensures that specific directory should be opened in VS code.

6.3 Front-end Board Initialization:

After setting up an IDE for python, we started coding for the game. First of all, we imported the installed library of arcade. We set up a board of size 10x10 means 10 rows and 10 columns of equal width and height at the left side of the window. This board is front-end visualization of game. We placed one sprite for player1 at bottom left of the board and second sprite for player2 at top right of the board.

```
def on_draw(self):

    arcade.start_render()
    arcade.draw_lrwh_rectangle_textured(0, 0, SCREEN_WIDTH, SCREEN_HEIGHT, self.Background)
    for x in range(0, 631, 63):
        arcade.draw_line(x, 0, x, 630, arcade.color.WHITE, 1.5)
    for y in range(0, 631, 63):
        arcade.draw_line(0, y, 630, y, arcade.color.WHITE, 1.5)

    arcade.draw_text("SNAILS Game", 530, 663, arcade.color.WHITE, 40,bold=True,font_name="ALGERIAN")

    arcade.draw_text("Player Turn: A", 700, 510, arcade.color.WHITE, 22,bold=True,font_name="ALGERIAN")
    arcade.draw_text("Player Turn: B", 1050, 510, arcade.color.WHITE, 22,bold=True,font_name="ALGERIAN")
    if self.count % 2 == 0:
        arcade.draw_text("Player Turn: A", 700, 510, arcade.color.CYAN, 22,bold=True,font_name="ALGERIAN")
    else:
        arcade.draw_text("Player Turn: B", 1050, 510, arcade.color.CYAN, 22,bold=True,font_name="ALGERIAN")
    # arcade.draw_text("Player A Score:", 700, 470, arcade.color.YELLOW, 20)
    # arcade.draw_text("Player B Score:", 1050, 470, arcade.color.YELLOW, 20)
    arcade.draw_text(str(self.scoreA), 800, 450, arcade.color.WHITE, 25,bold=True,font_name="ALGERIAN")
    arcade.draw_text(str(self.scoreB), 1150, 450, arcade.color.WHITE, 25,bold=True,font_name="ALGERIAN")
    if self.count % 2 == 0:
        arcade.draw_text(str(self.scoreA), 800, 450, arcade.color.CYAN, 25, bold=True,font_name="ALGERIAN")
    else:
        arcade.draw_text(str(self.scoreB), 1150, 450, arcade.color.CYAN, 25,bold=True,font_name="ALGERIAN")

    arcade.draw_texture_rectangle(992, 510, 180, 120, self.vs)
    # self.snailD.draw()


    self.player_list.draw()
    self.player1_list.draw()
```

Figure 5: code for Front end Board of the game

6.4 Back-end Grid Initialization:

We developed a grid of 10 rows and 10 columns same like game board but it is not visualized to the users of the game. This is developed just to remember the states and position of the sprites and check the conditions of the game. Initially, the whole grid contains zero in each box except two boxes where

the sprites are placed on the board. Those two are initialized by the value of 1.
When the sprite is moved or clicked on the front-end board, the values at the back-end grid will be changed according to the position of sprites. When the sprite moves from the previous position, it will leave a splash at back and grid will put the value of 10 in that specific box.
In short terms, with the movement of front-end board, the values in back-end grid will also be changed.

0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0

Figure 6: Back-end Grid Initialization

6.5 Front-end Game Window:

After the initialization of front-end board and back-end grid, we worked on the designing of game window to make it user friendly. We designed front-end board to the left of the window and designed player's scores to the right

side of of board. The more front-end of game window is user friendly, the more users enjoy to play the game.
Here is the code for Designing the front-end game window;

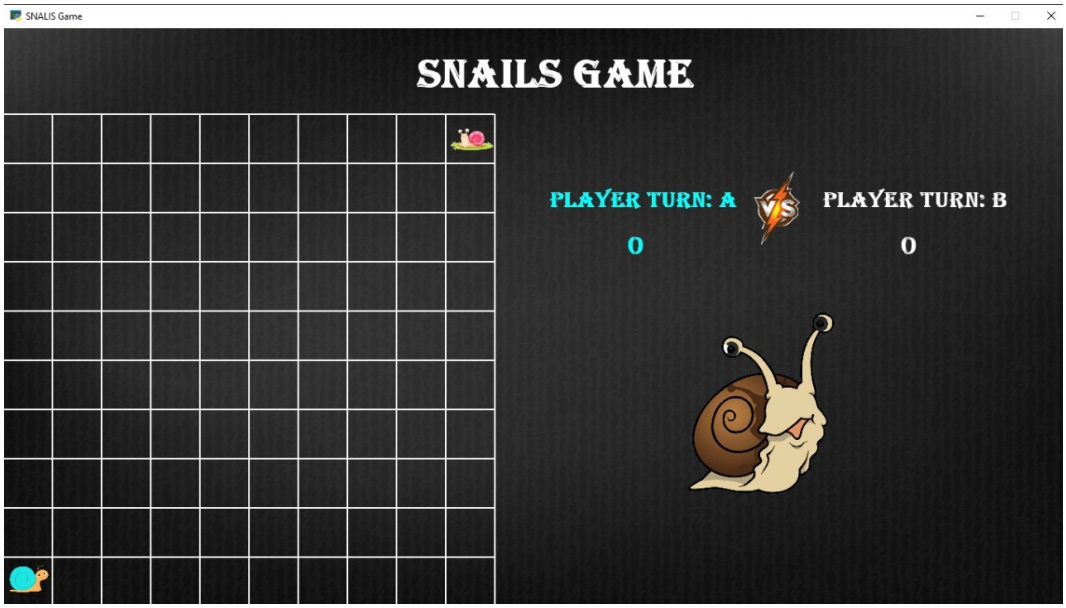

Figure 7: Game Window for users

6.6 Functions of Game:

Everything has been set up and designed now it was the time to implement the basics of games that what rules should be adopted and how it should be played.

6.6.1 GameView():

The complete code of game is developed in Object Oriented Programming. We have used a class of GameView() in which we implemented all the functions of game whether it is front-end or back-end.

6.6.2 On-mouse-press():

We have used the function of MouseClick() to move the sprite from one position to an other position. This is a built-in function of arcade library.

When a player click on any position of the grid, if it is valid move then the sprite will be moved to the clicked position. The sprite will leave a splash to the previous position which means that opponent can't move to that position. Use of valid mouse-click is very important in this game.

6.6.3 direction-slip():

There is a rule in this game that you can't move to previous position but there is a term used in this game is Slip function. It means when you click on the previous position of the board, it will be checked that if there is splashed behind that sprite then sprite will slip to the last splash. This is slippery function.

6.6.4 Is-Legal-Move():

When any player will click on the board, it will be checked that the move of player is legal or not legal. If it clicks on the position of opponent player then it is illegal. Clicking on diagonal of the board is illegal move. Clicking on the other player's sprite is illegal move. Jumping is illegal move.

6.6.5 Game-over():

When there is no empty box on the front-end board and number of zeros (0's) is none in back-end grid then the game will end. The player with maximum number of splashes on the front-end board and maximum number of 10 or 20 in back-end grid will win the game. There is also a possibility of draw, if both players have equal number of points.

6.6.6 Eval-board():

This function checks after each turn that which player is winning. It updates the board with scores and checks that game is finished or not. If game is not finished then checks the turn and again updates the board.

6.7 AI-agent Move and Algorithms:

If a human is playing with AI-agent the AI-agent will check the best move where it can either win from human and minimize the winning possibility of human. There are certain algorithms and functions are developed in this

game which will return the best move to AI-agent and then agent move according to it. AI-agent changes its move according to the intelligence of human. It means it checks all the possible winning states at every move.

6.7.1 Minimax():

In an n-player game, generally a two-player game, a minimax algorithm is a recursive method for deciding the next move. Each position or condition of the game has a monetary value. A position evaluation function is used to calculate this value, which reflects how good it would be for a player to attain that position. The player then makes a move that maximises the position's minimal value as a result of the opponent's probable next moves.[1]

6.7.2 Heuristic():

The initial splashes of A.I Agent in the board will be checked by the heuristic search algorithm, which will then, at that point, be added to the winning score. The heuristic search algorithm will be advised where to search for the best winning score for the A.I Agent. If the A.I. agent isn't on the board's dividers, the Heuristic search algorithm will add some additional scores to the winning score. At long last, the heuristic search algorithm will look for a particular number of empty boxes in the predefined direction as given in the heuristic search algorithm's parameters.

7 Game User Interfaces:

When the code is run, the gameView() function will display the window of the game.

7.1 Starting Screen

First it displays a screen that is an opening window of the game.

Figure 8: This is the starting screen of the game.

7.2 Instructions Screen

After clicking on the starting screen then it comes the instructions screen where the instructions of games is labeled. "How to play game" is also defined at the instruction scree.

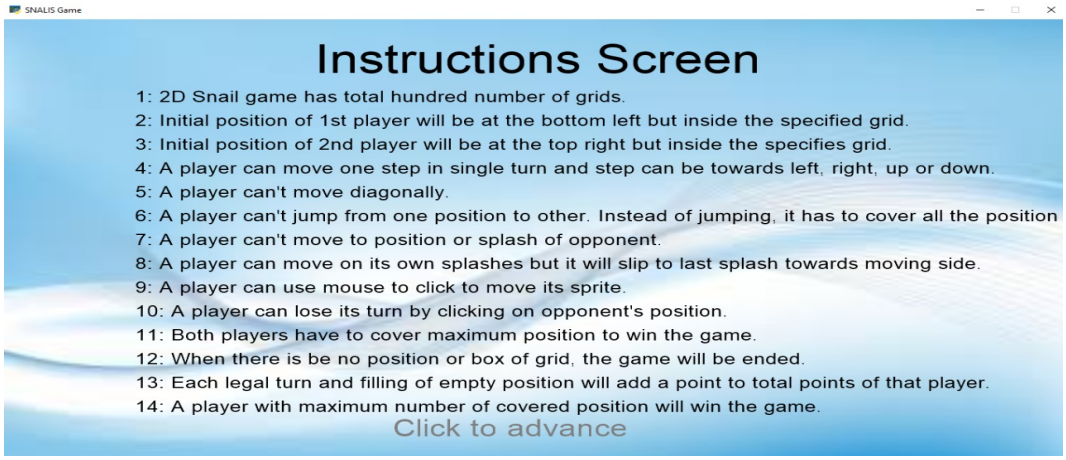

Figure 9: This is the Instruction screen of the game.

7.3 Game Window

After clicking on the option to start game, it is asked the type of game whether it should be two players game or AI-agent game. After selecting the game type then the game window will be opened. Game window interface is user-friendly and easy to understand and play the game.

Figure 10: This is the board of game where players can play.

7.4 Winning Screen

At the end of the match, there will be a window to show the scores of each player and to define who won the match. From the same window you can also play again.

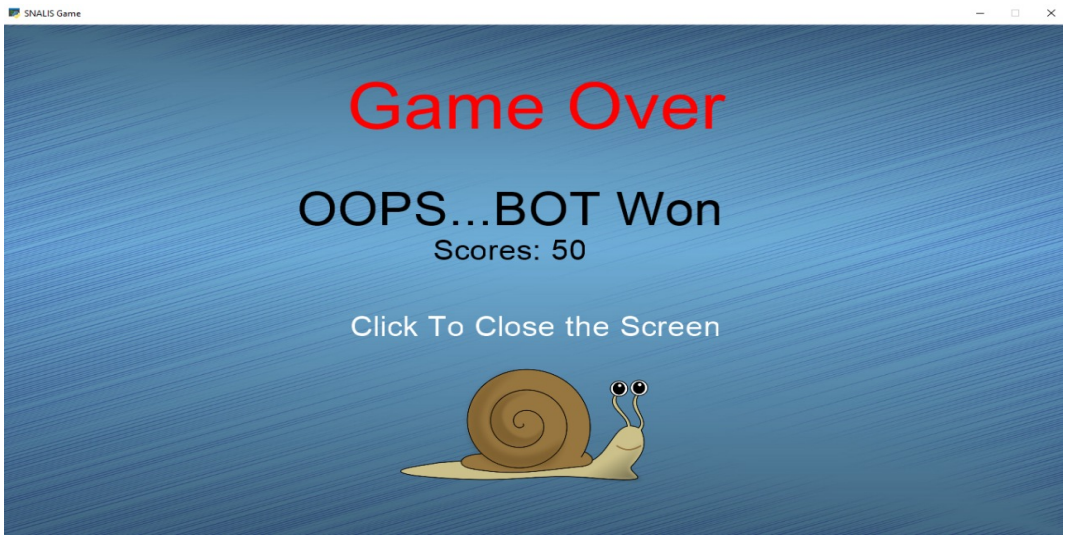

Figure 11: This is the scoring screen of the game.

8 Testing and Analysis of 2D Snail Game:

8.1 Test1:

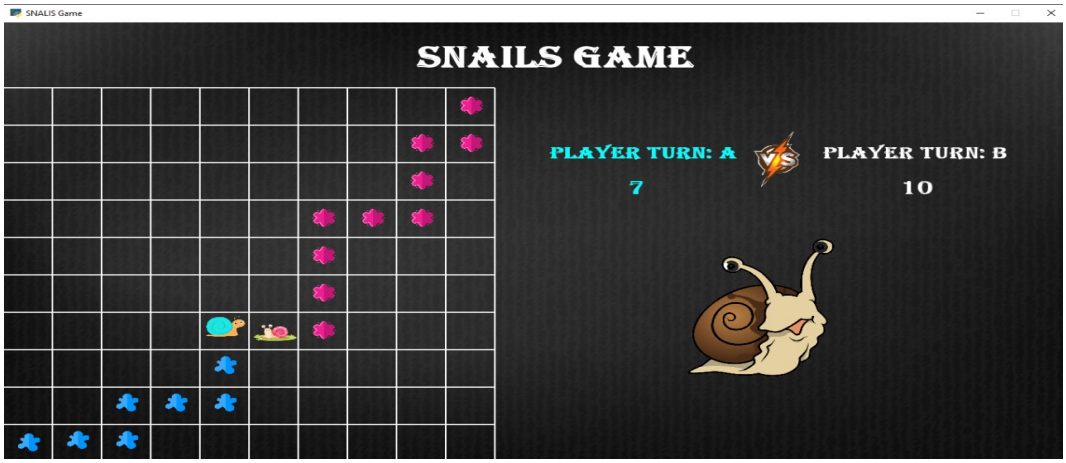

Figure 12: This shows the game is in running state.

8.2 Test2:

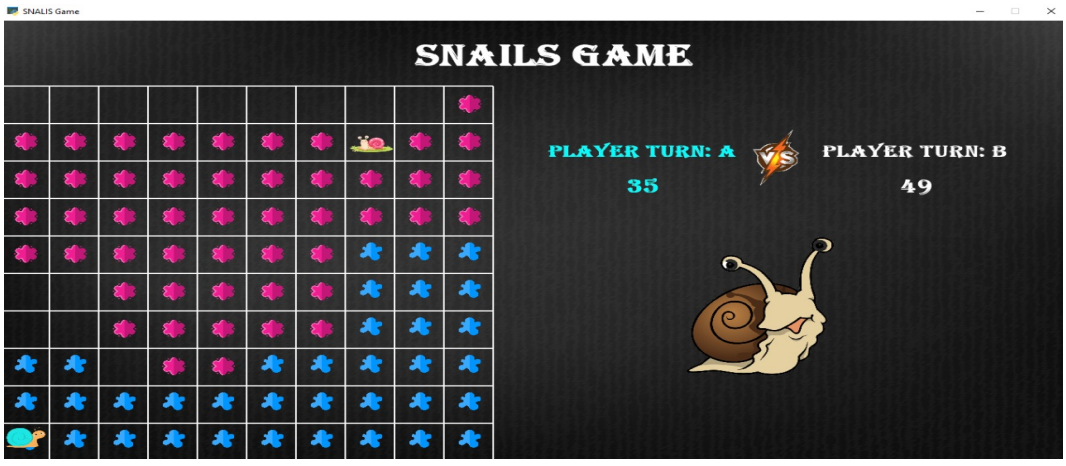

Figure 13: This shows the game is in running state.

8.3 Test3:

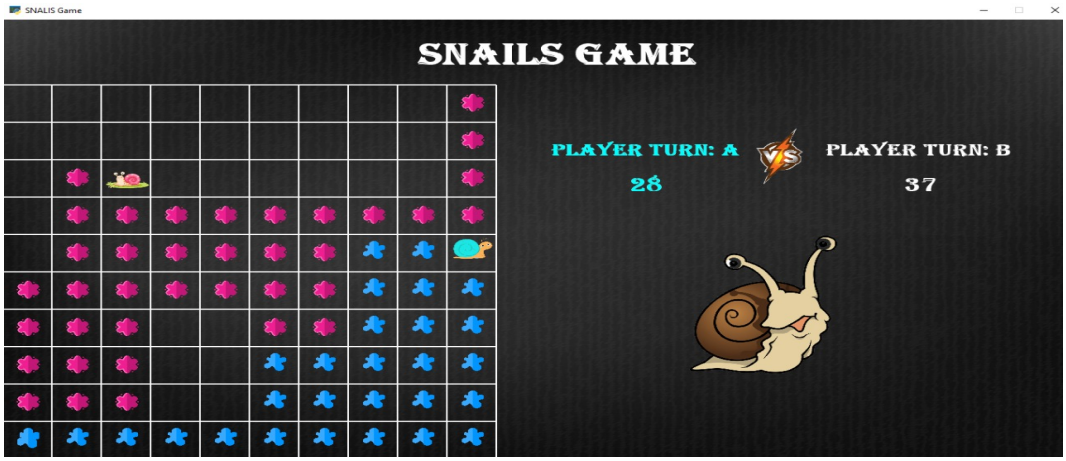

Figure 14: This shows the game is in running state.

8.4 Test4:

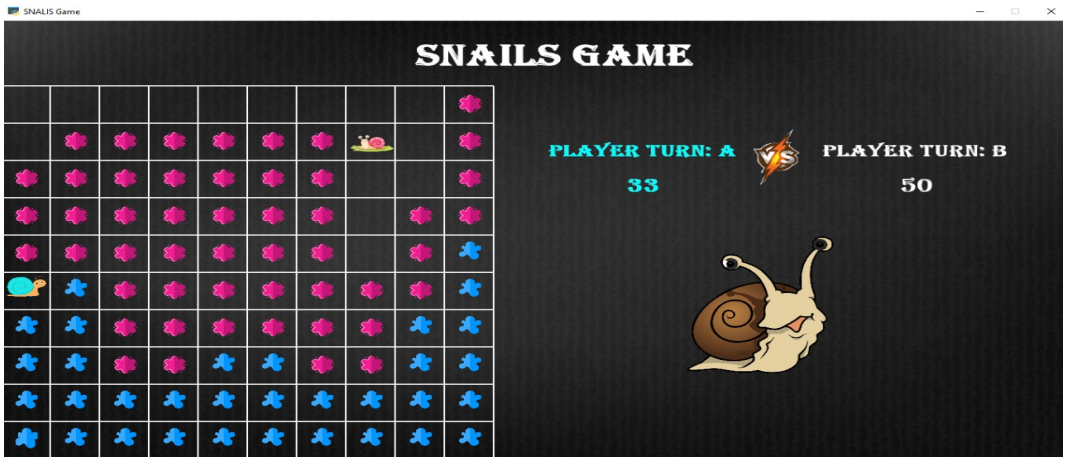

Figure 15: This shows the game is in running state.

9 Conclusion

This report is proposed of development of a 2D Snail game. We tried to make it user friendly and used the concept of artificial intelligence. It is possible now that you can play this game with computer because we made computer so much intelligent that it can play with the intelligence of human. There is an option to play this game between two humans. Minimax and heuristic algorithms have been used to implement the AI in the game.

10 References

1. https://en.wikipedia.org/wiki/Minimax
2. https://data-flair.training/blogs/heuristic-search-ai/
3. https://upload.wikimedia.org/wikipedia/commons/thumb/6/6f/Minimax.svg/400px-Minimax.svg.png
4. https://techvidvan.com/tutorials/ai-heuristic-search/
5. https://www.javatpoint.com/ai-alpha-beta-pruning
6. https://www.geeksforgeeks.org/a-search-algorithm/