2020

Artificial Intelligence

ARTIFICIAL INTELLEGENCE FOR GAMES

SUBMITTED TO: FAISAL ZEESHAN

GROUP MEMBERS: |

Mahmood Yousuf (1802011) | Hamdan Ali Baloch (1802037) | Salman Ali (1802010)

ABSTRACT

AI-Game development is the integral part of Artificial Intelligence. Today Gaming Industry is following the AI based games. To make play games with machine or computer here AI agents will be used. AI agent will detect a conceptual flow and will follow it. AI agent either stops opponent or tries to win the game. This report is about AI Agent based gaming. AI Gaming helps to develop mental and phycological skills. Making computer/program intelligent so that it should move intelligently like humans. Many algorithms such as path finding, trees, and heuristic functions can be used for making AI Agents. This project use minimax and heuristic functions to move an AI agent. In this project an AI bot is developed to play with human. AI agent should be enough intelligent to compete with human. AI bot can try to stop opponent to score or it will try to score maximum. Finding optimal solution is the key factor for AI agent. A.I Agent will think as like human brain to choose the best move.

Table of Contents

Introduction

Artificial intelligence is making the machine/program intelligent like humans. Making the machine thinkable like humans. AI is the studying thought process of human beings. Artificial Intelligence is playing a very important role in gaming industry. AI or bot-based game are need of many games. Designing the games with AI is very common now a days. In this report we will discuss how we used AI in our snails' game.

Game Rule and Goals

The rules of snails' game are very simple. Two sprites/player have to move in such a way that they have to cover more and more boxes. With the movement each player will leave a splash behind. The player having maximum number of splashes will win the game.

1. One sprite can move in four ways like it can move only left, right, up and down.
2. At a time one player move to one box means player can't jump on boxes.
3. One player can't move on opponent sprite and splashes.
4. Player can move on its own splashes and this process can be called as slip of sprite. In this case player can make jumps on its own splashes.
5. At the end of game player with maximum splashes will win.
6. Both players will try to cover more and more boxes.
7. In case of any wrong movement the turn of player will be lost.

Game Requirements and Installation

This game is designed in python using its library name arcade. This game needs any python IDE with arcade installed in it. If you want to run the .PY file you will have to attach all the images that are being loaded in the game. You have to make sure that your program and all attached images are present in a single directory.

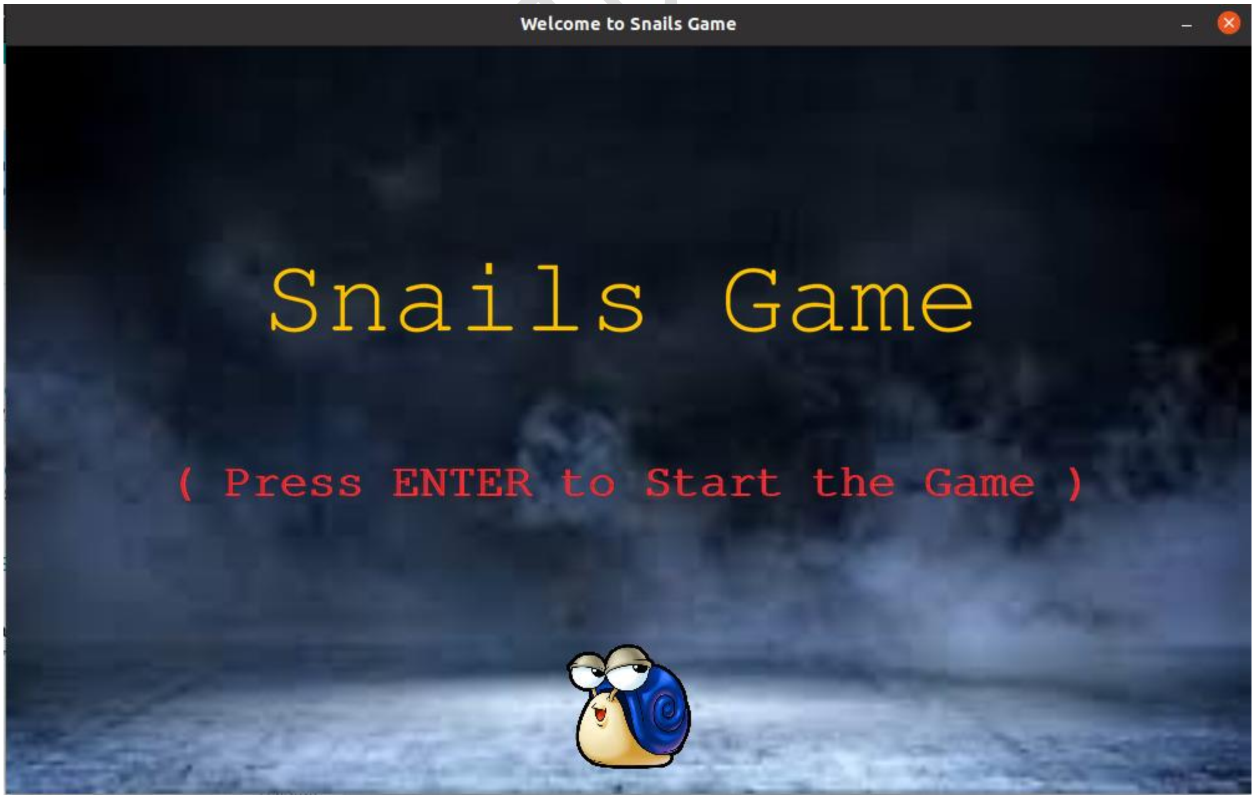
Using the Game

In game we use different terms are given below.

1. AI Agent
2. User Player
3. Grid ()
4. Board ()
5. Start ()
6. Exit ()

How to Start a New Game

When you start/run program then you will get User Interface which will guide you how to start a game. For starting game go through the requirements of User Interface messages.



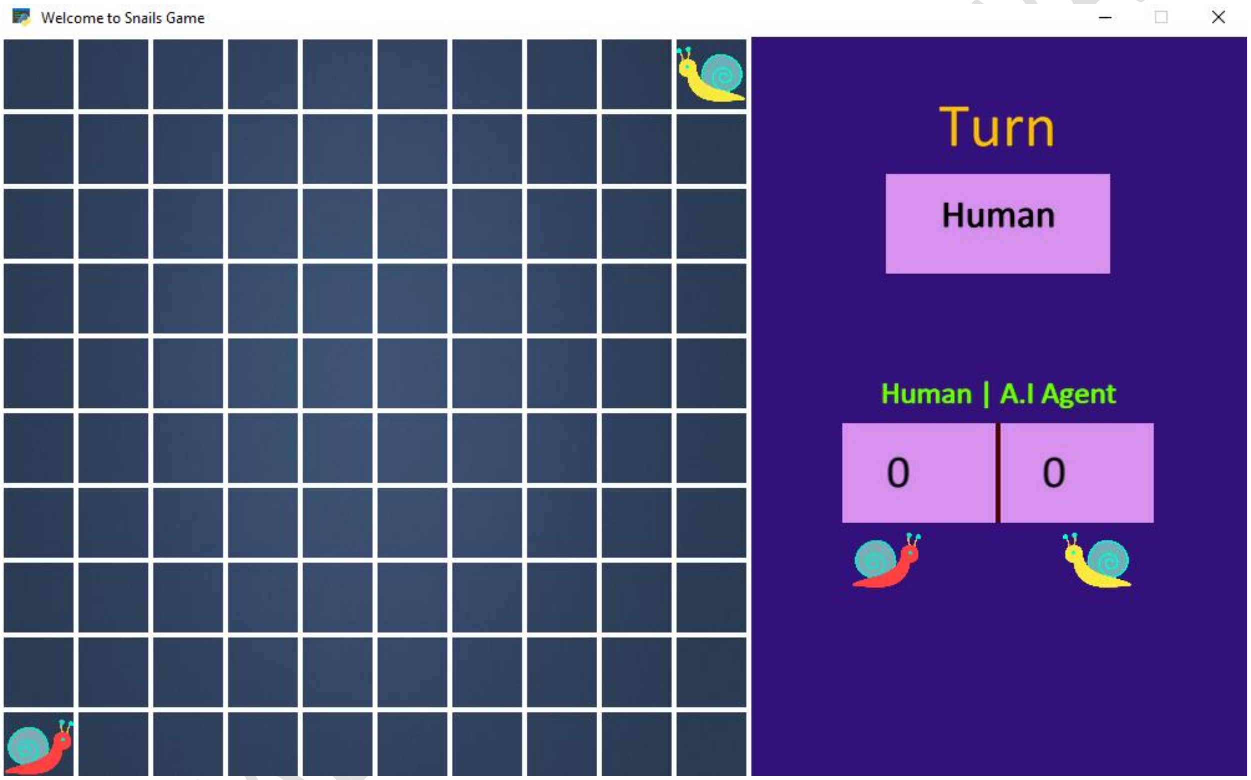
Welcome to Snails Game

Snails Game

(Press ENTER to Start the Game)

User Interface

User Interface of this game is very simple and user friendly. User can follow the guidelines of UI.



Welcome to Snails Game

Turn

Human

Human | A.I Agent

0 | 0

Illegal Moves

In board if you click on any wrong place/area then it will be considered as an illegal move and the turn will lost. Your movement on board have four choice up, down and left right. If you are not clicking on required boxes like up, down, left and right then your turn will be lost and it will be considered as an illegal move.

How to Win

The win of games depends upon number of splashes. At the end of game, the player with maximum number of splashes will be the winner. The getting of maximum splashes depends upon your playing strategies. AI agent is much intelligent to win against human player so the game is about intelligence. intelligence of human vs intelligence of AI Agent.

Literature Review

The review of this game is based on developing AI Agents. AI based games can also be called as Mind games. In real/common world these types of games also called game between human vs computer. Majority of games are launched few years ago are totally AI based games. Today the demand of these type of games is increasing day by day.

Movement

Movement of players and their actions are some basics in AI based games. Players can move in such a way that they don't affect the game rules. In different types of AI game like chess there are different types of moves for any king of sprite for every player. For example, in chess game crown can't move like knight and queen.

Path Finding

During the game it is very important for both human and bot to find a such path that it will defend the attack of opponent. Similarly, the player can find an attacking path for win state. AI Agent must be much intelligent to find a right path to follow and can change the path in different situations of game. Like in AI based games your sprite should not stuck at any stage so that it can make moves to attack.

Decision Making

We may say that the movement of players are fixed. How they can move and can't. But it comes to AI based agent then the concept will change at some points like Agent will have to make decisions. AI agent have to make decision that it make further attacks or go for defensive mode.

AI Agent

An agent can be anything that perceive its environment through sensors and act upon that environment through actuators. An Agent runs in the cycle of **perceiving**, **thinking**, and **acting.** AI agent is a player that can make move like human player. The AI agents act in their environment and make move against other agents(human).

MINIMAX

The minimax algorithm is very popular for teaching AI agents how to play turn-based strategy games. The reason being is that it takes into account all the possible moves that players can take at any given time during the game.

With this information, it then attempts to minimize the opponent player's advantage while maximizing the agents at every turn the AI agent gets to play. There are two player mina and maxi they play for their own perspectives.

Heuristic

Presently, video games are not only designed and developed for fun and entertainment but they also play an important role in education, military, medical, and business sectors. The reason of the mobile game growth is the growing number of mobile phones. **Heuristics** are shortcuts to solutions. The idea of heuristic methods in AI is based on cognitive science or the study of how humans think. Indeed, humans use heuristics all the time to make decisions and solve problems. Likewise, heuristic algorithms are often used in AI to get a computer to find an approximate solution instead of an exact solution.

Methodology and Implementation

Before starting any project, there must be a structure or plan in the mind that what to do and how to do. The plan should be clear and the implementation will be easy and effective. Selecting the tools according to the project is the initial step to any project.

Arcade Setup

As Python is the emerging and widely using programming language. Python is being used in almost all across the globe. So, for this project the most suitable language is considered is Python. In Python we used a library named arcade. Arcade library is basically used for 2d gaming development. Arcade is easy to use. For setting arcade in your computer you have to install any Python IDE. After installing IDE, you have to install pip to get libraries of Python. You have to install arcade library of Python using pip. After installing the arcade, you have to run the PY program on your IDE. You have to make sure that all the attached document of game/program must be in a single directory.

Initialize Board ()

In coding section, we have to code the backend procedures and classes. Here we used some classes of arcade like views. We have followed the modular approach. Initializing the backend board. The backend board is 2d Board which is made by nested lists. The backend board contains initial states of players. The whole remaining board is initializing with all zeros. The backend board values will change with flow of game.

Board [rows] [column]

Row ->

Column

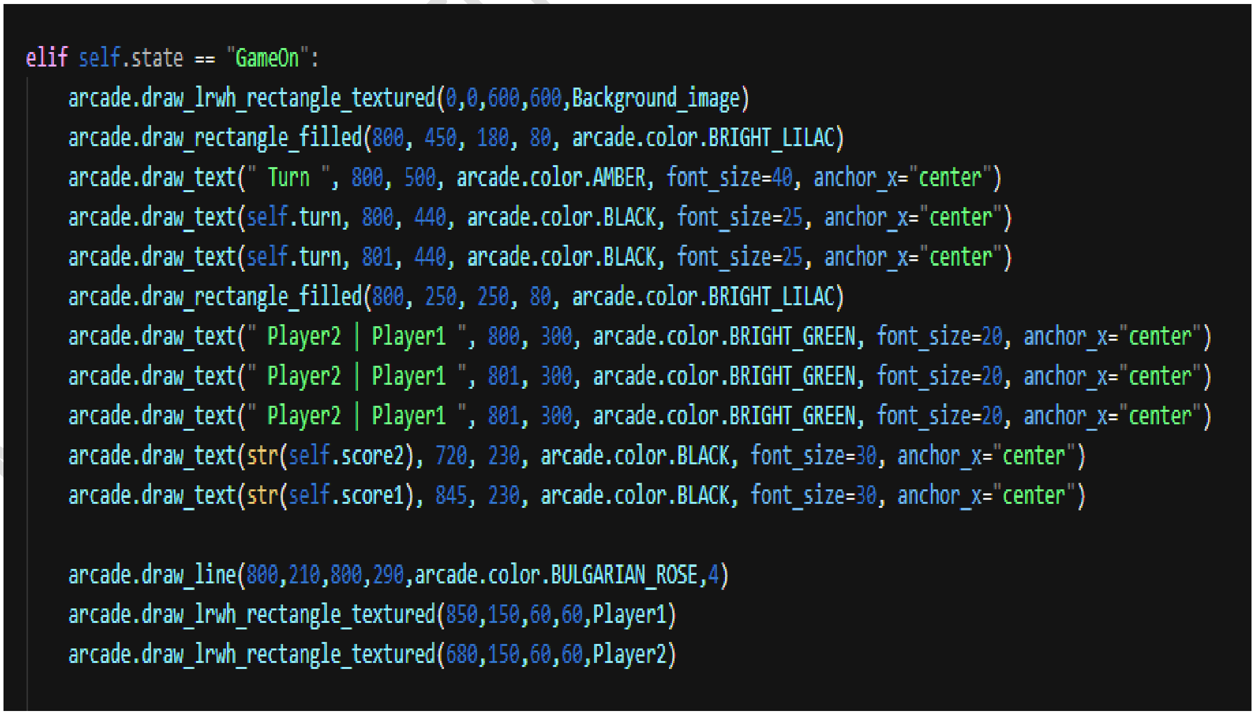
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0

Initialize Grid ()

The frontend grid is connected with the backend 2D Board. Changing on frontend will also results in changing the backend 2D board. The board will display us a UI which contains splashes and sprites of both players.

Frontend Representation

The frontend has total three number of states which are Game Menu, Game ON and Game End. Frontend contains many images, set of lines and rectangles. Frontend is documented properly to guide the user. In frontend there are partitions in which different tasks are represents.

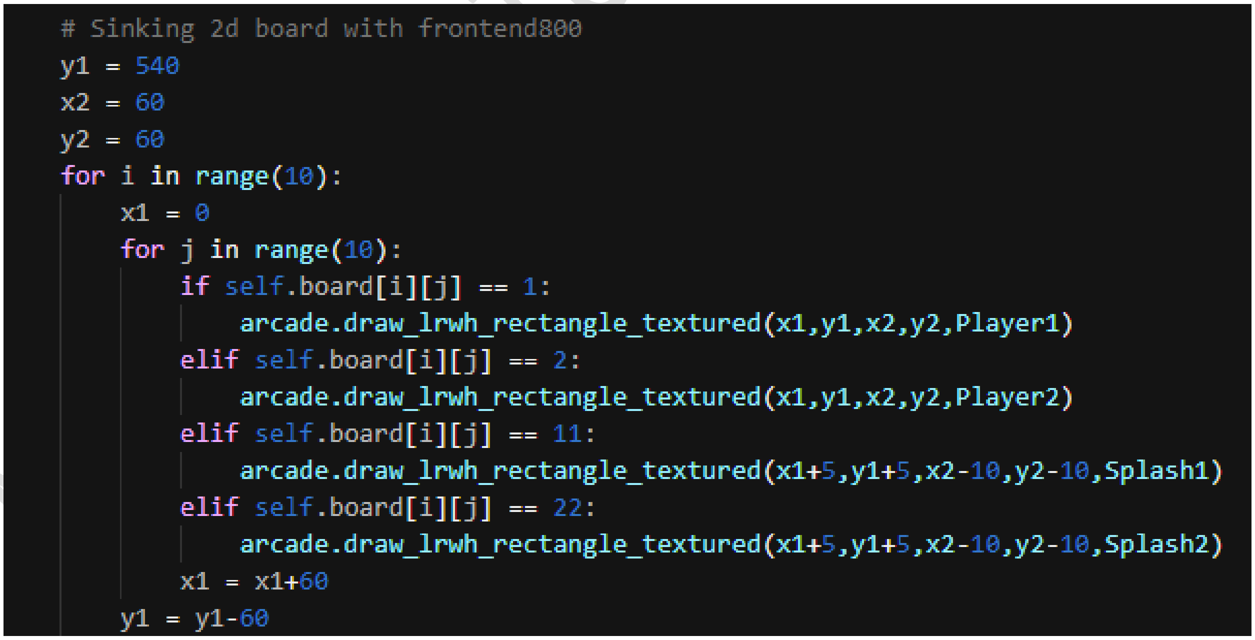


```
elif self.state == "GameOn":
    arcade.draw_lrwh_rectangle_textured(0,0,600,600,Background_image)
    arcade.draw_rectangle_filled(800, 450, 180, 80, arcade.color.BRIGHT_LILAC)
    arcade.draw_text(" Turn ", 800, 500, arcade.color.AMBER, font_size=40, anchor_x="center")
    arcade.draw_text(self.turn, 800, 440, arcade.color.BLACK, font_size=25, anchor_x="center")
    arcade.draw_text(self.turn, 801, 440, arcade.color.BLACK, font_size=25, anchor_x="center")
    arcade.draw_rectangle_filled(800, 250, 250, 80, arcade.color.BRIGHT_LILAC)
    arcade.draw_text(" Player2 | Player1 ", 800, 300, arcade.color.BRIGHT_GREEN, font_size=20, anchor_x="center")
    arcade.draw_text(" Player2 | Player1 ", 801, 300, arcade.color.BRIGHT_GREEN, font_size=20, anchor_x="center")
    arcade.draw_text(" Player2 | Player1 ", 801, 300, arcade.color.BRIGHT_GREEN, font_size=20, anchor_x="center")
    arcade.draw_text(str(self.score2), 720, 230, arcade.color.BLACK, font_size=30, anchor_x="center")
    arcade.draw_text(str(self.score1), 845, 230, arcade.color.BLACK, font_size=30, anchor_x="center")

    arcade.draw_line(800,210,800,290,arcade.color.BULGARIAN_ROSE,4)
    arcade.draw_lrwh_rectangle_textured(850,150,60,60,Player1)
    arcade.draw_lrwh_rectangle_textured(680,150,60,60,Player2)
```

Connectivity of Frontend with Backend

The frontend is mapped with the backend 2D board. Similarly, the other variables in frontend also have their backend variable to keep the flow off changings.



```
# Sinking 2d board with frontend800
y1 = 540
x2 = 60
y2 = 60
for i in range(10):
    x1 = 0
    for j in range(10):
        if self.board[i][j] == 1:
            arcade.draw_lrwh_rectangle_textured(x1,y1,x2,y2,Player1)
        elif self.board[i][j] == 2:
            arcade.draw_lrwh_rectangle_textured(x1,y1,x2,y2,Player2)
        elif self.board[i][j] == 11:
            arcade.draw_lrwh_rectangle_textured(x1+5,y1+5,x2-10,y2-10,Splash1)
        elif self.board[i][j] == 22:
            arcade.draw_lrwh_rectangle_textured(x1+5,y1+5,x2-10,y2-10,Splash2)
        x1 = x1+60
    y1 = y1-60
```

Two Player Game

By two player game we mean human to human game. This game uses two users to play with each other. Every user has to use their own turn. Any player who achieved 49 score will win automatically.

Functions of SLIPS

In this game by slip we mean that if a player clicks or moves on its own splash. Then it has to jump all splashes and have to stop on last one. The slip can be up, down, left and right. Both Player can slip on their own splashes not on opponent splash. In left and right slip we have to change column number of 2D board. Similarly, in up and down slips we have to change the row number accordingly.

Get Position of Mouse Click ()

There is a built-in function in arcade which works on clicking the mouse. When mouse is Clicked then it will generate axis of arcade window. Now for mapping these axes with backend 2D board there is a function named get position.

A.I Agent move ()

This function will call minimax function which will return best move for A.I Agent. After returning the best move from minimax A.I Agent will move according to best move. Then A.I Agent will check left, right, up, down and invalid conditions for next move. Then it will turn to Human

minimax ()

Minimax function will call heuristic function on left, right, up, down move to check the best winning score. By using this best winning score, A.I Agent will try to move according to it intelligently. Minimax uses the heuristic function for best move other wise minimax always choose optimal move which will cost too much time for just single move

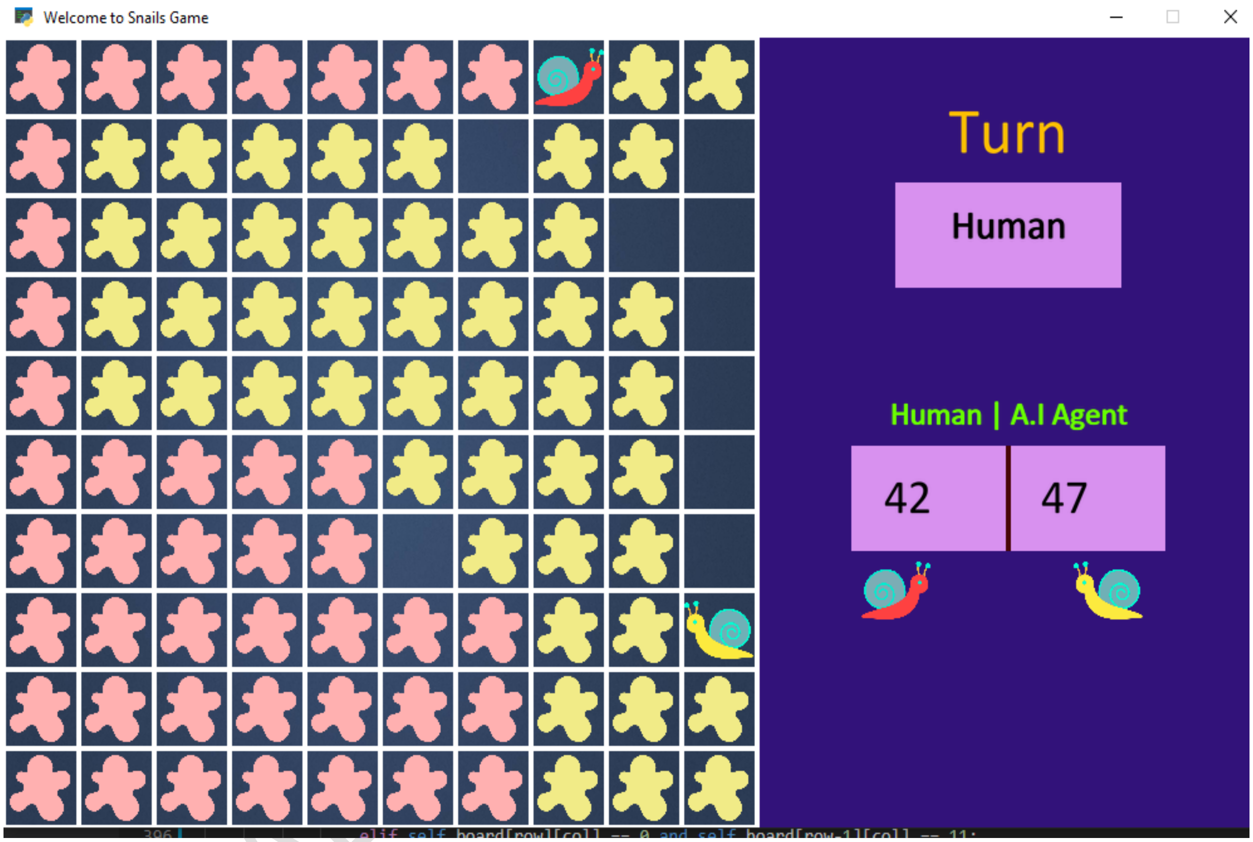
heuristic ()

Heuristic function will check first splashes of A.I Agent in board than it will add to the winning score. Heuristic function will receive direction from where to check the best winning score for A.I Agent. And Heuristic function will add some extra score to winning score if A.I Agent is not on walls of board. Finally, heuristic function will look number of empty boxes on specified direction as passed in argument of heuristic function

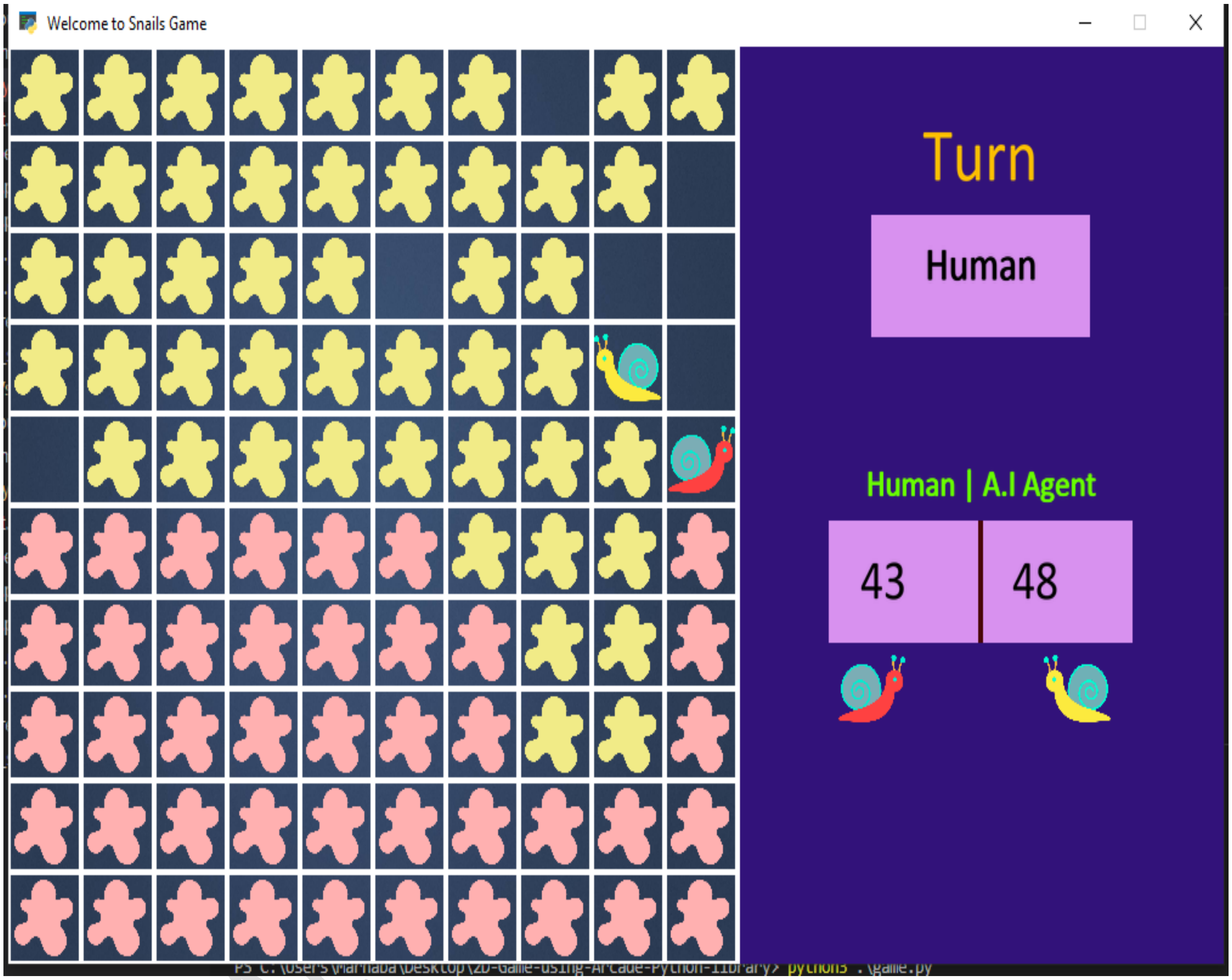
Testing

On testing it was found heuristic function tried best to compete with human player.

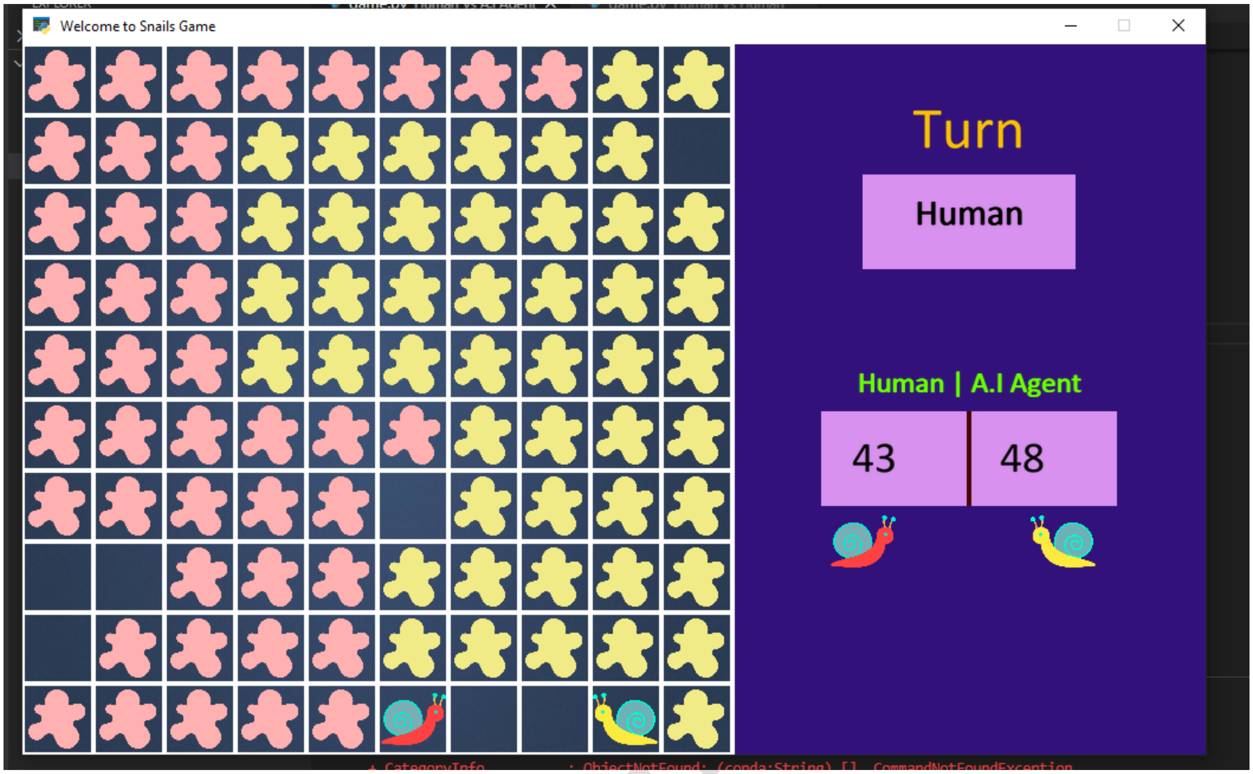
Below some test cases just before wining state for A.I Agent

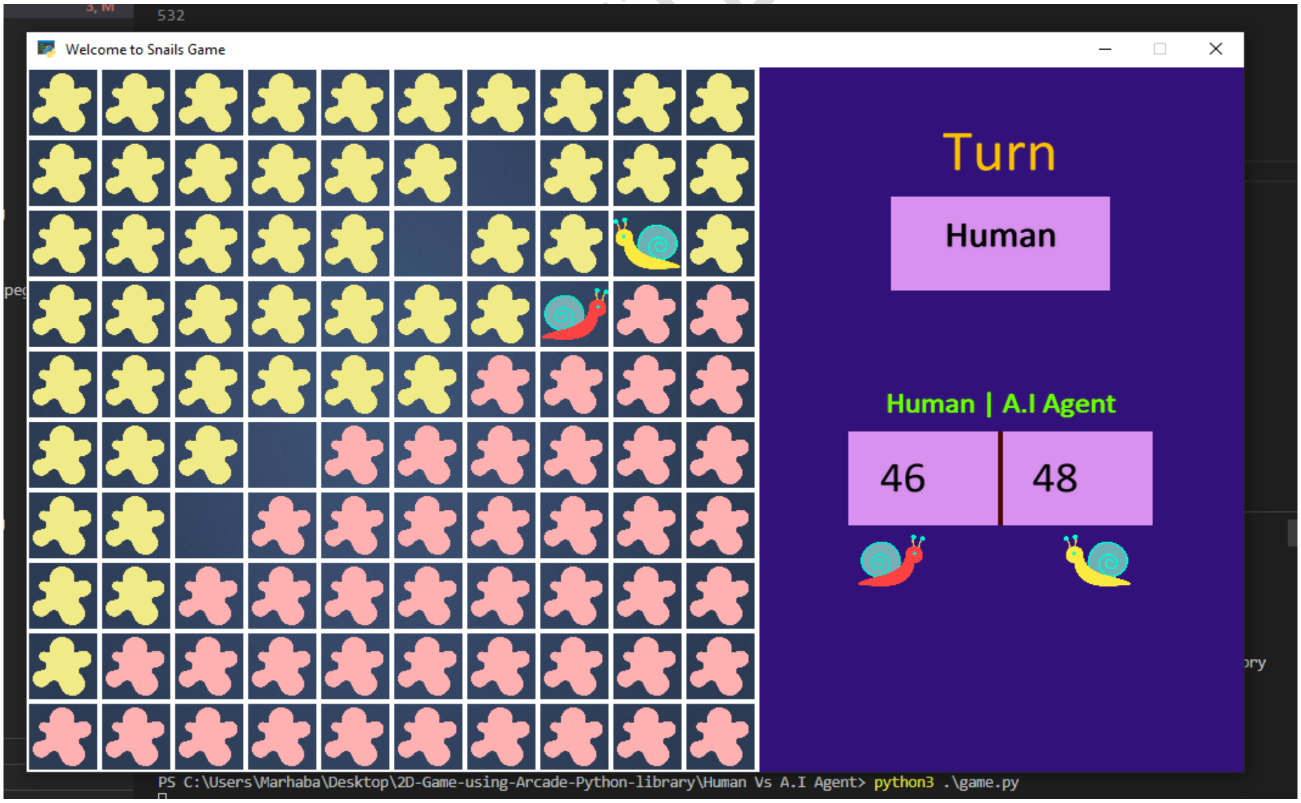


```
elif self.board[row][col] == 0 and self.board[row-1][col] == 11:
```



```
PS C:\Users\Marhaba\Desktop\2D-Game-using-Arcade-Python-library> python3 .\game.py
```





```
PS C:\Users\Marhaba\Desktop\2D-Game-using-Arcade-Python-library\Human Vs A.I Agent> python3 .\game.py
```

Result and Analysis

AI Agent tried his best to compete with human but at some cases A.I Agent fails to compete with human. But at most cases A.I Agent can easily beat Human. The most crucial role of A.I Agent is heuristic function which always look for best move to beat the Human player and Heuristic function helps minimax function to decide which will be the best move for A.I Agent to Beat the Human player at any time in Game.