

National University of Sciences & Technology (NUST), Baluchistan Campus
(NBC), Department of Computer Science



ARTIFICIAL INTELLIGENCE

“ASSIGNMENT 1”

PROJECT TITLE:

“Simulating Pacman game using AI”

GROUP MEMBERS:

- Mustafa Ali (311297)
- Muhammad Fahad (356324)

CS-19 (5th SEMSTER)

INSTRUCTOR: Dr. Ayesha Maqbool

DATE OF SUBMISSION: 09-10-2021

Table of Contents

Abstract.....	1
1.0 Introduction	1
2.0 Features.....	2
3.0 Methods	2
3.1 Setting up GUI	2
3.2 Setting up Game Engine	2
3.3 Incorporating AI in pacman (TASKS)	2
3.3.1 (Task-1) Rules of the Game:.....	2
3.3.2 (Task-2) Properties of the Environment:	2
3.3.3 (Task-3) Simulation	3
3.3.3.1 Reflexive Agent.....	3
3.3.3.2 Goal-Based Agent.....	4
4.0 Experimental Results	5
4.1 Graphs.....	5
4.2 GUI	7
4.2.1 Reflexive Agent Simulation	8
4.2.2 Goal Based Agent Simulation	9
5.0 Contributions.....	11
6.0 Link of the Simulation and files.....	11

Abstract: The game Pac-Man is a both challengeable and satisfactory video game that has been the focus of some important Artificial Intelligence research. The goal for using the game Pac-Man as a test bed in our experiment is that the Pac-Man game provides a sufficiently rich and challengeable platform for studying the AI in the computer game, and that it is simple enough to permit understanding of its characteristics. To achieve these desired goals we have used multiple informed search methodology. In addition to above, we have incorporated ghost agents to search smartly based on various variables such as distance from packman, food locations etc. Finally, the experimental result from the game Pac-Man is presented to demonstrate the effectiveness and efficiency of these algorithms.

1.0 Introduction

This project was started to apply an array of AI techniques to playing Pac-Man. This project visualizes foundational AI algorithms namely informed state-space search. As Pacman is a zero-sum game which means the pacman will maximize its score by eating all the coins in the maze meanwhile the ghost will try to minimize pacman's score through interception. The algorithm uses adversarial search tree to maximize pacman score. The algorithm in this project processes the data provided by the game state. The game state is defined as the current position of the pac-man on the maze along with its neighboring coin and ghost position. After the game state has passed to the desired algorithm the result is used to increase the state-space search or move the agent if optimal solution is reached.

2.0 Features

Some application-based features include:

- The GUI interface for selection Pac-man agent types, ghost agents and maze layout.
- The results of the game will be displayed on the terminal which is integrated into the GUI screen.
- Ability to run multiple instances of the game to generate cumulative result of the algorithm's working.
- Algorithm visualization on pac-man game.
- Algorithm's effectiveness shown through scoring-based module.
- Ability to switch between goal-based and reflexive based agent pacman

3.0 Methods

3.1 Setting up GUI:

The GUI interface of the project is designed using a python GUI designing tool called PyQt5. PyQt5 designer allows easy drag and drop GUI designing ability and also allows easy conversion of the user interface file into python code.

3.2 Setting up game engine:

The pacman game has been developed using the **tkinter library** in order to display all entities of the pacman world including coins, capsules, pacman, ghosts, mazes etc.

3.3 Incorporating AI in pacman (TASKS):

3.3.2 (Task-1) Rules of the Game:

We defined goals and rules for the Pacman that are:

- Eat all the dots quickly.
- Get as many points as possible.
- Don't die

Some of the rational actions for pacman that would allow it to maximize its goals would include:

- Moving toward coins / food.
- Avoiding ghosts
- Moving toward capsules.

3.3.2 (Task-2) Properties of the Environment:

Before establishing AI in pacman it was important to assess the environment type and the types of agents involved. The environment analysis of the pacman can be shown below:

Pacman Game		
Observable	Fully	Pacman knows all the aspects of environment, as state is maintained
Deterministic	Stochastic	It is deterministic however; action of opponents matters hence stochastic
Episodic/ Sequential	Sequential	Each decision depends on the previous maintained state, hence sequential
Static/ Dynamic	Dynamic	The environment is constantly changing i.e. agents moving, coins reducing etc.
Discrete/ Continues	Discrete	There are clearly defined perceptual experience for the game
Agents	Multi-Agent	Multiple agents such as pacman, ghosts are involved

3.3.3 (Task-3) Simulation

The goal was to solve pacman i.e. able to act rationally without user intervention. Firstly, the project involves pacman being a model-based reflex agent that suggests action based on the previous state.

3.3.3.1 Reflexive Agent

Reflex agents choose an action based on the current perception of the world rather than planning or considering future consequences. This is a good rule in the short term for staying alive but could lead pacman to be trapped in a corner. Below are some rules that pacman follows in Reflexive based agent

- Without any disturbance from the ghost, pacman would go towards the closest food location. In order to calculate the closest food, we have used **breadth first search** algorithm shown below which always finds the shortest path first which gives an optimal solution. Its **space and time complexity is $O(b^d)$** where d is the depth of the tree.
- In this mode, pacman tries to eat all the coins without thinking of the ghosts, as it is reflexive based and not goal based, it does not think of the future but rather is more concerned about the current state which is good for the short run but will eventually end up in a dead end

```

dmap = walls.copy()
stk = utility_functions.Queue()
stk.push(loc)
dmap[loc[0]][loc[1]] = 0
dis = 0
while not stk.isEmpty(): # Using BFS inorder to find the closest coin available
    x , y = stk.pop()
    dis = dmap[x][y] + 1
    if coin[x][y]:
        break;
    for v in [(0, 1) , (1, 0) , (0 , -1) , (-1 , 0)]:
        xn = x + v[0]
        yn = y + v[1]
        if dmap[xn][yn] == False:
            dmap[xn][yn] = dis
            stk.push((xn, yn))
if(coin.count() == 0):
    dis = 1
score = 1 - dis
for ghost in ghosts:
    if ghost.scared_timer == 0: # active ghost poses danger to the pacman
        score -= 100 ** (1.6 - coords_distance(ghost.get_coord(), loc))
    else: # bonus points for having a scared ghost
        score += 25
score -= 30 * coin.count() # bonus points for eating a coin
return score # next_game_state.get_score()

```

3.3.3.2 Goal-Based Agent

Goal-based agents plan by considering "what if" a certain action is taken. This needs a model including elements of the environment that are needed to measure success.

In this mode, the chances of pacman losing is narrowed down to almost 0 because the only goal of the pacman is to win the game and this goal is achieved by the following points

- Same as the reflex based Without any disturbance from the ghost, pacman would go towards the closest food location. In order to calculate the closest food, we have used breadth first search algorithm shown below which always finds the shortest path first which gives an optimal solution. Its space and time complexity is **O(b^d)** where d is the depth of the tree.
- If the goal is to score points, then points need to be measured. Knowing how many points are scored for a given action, as compared to other actions, provides pacman with information for choosing an action.

- Moreover, pacman would avoid a scared ghost, because eating it produces an active ghost in the center and lose the bonus points, so if there is a choice, it would tend to leave scared ghosts alone and take as much advantage of the scared time as possible.

4.0 Experiments/Results/ Discussion

When designing a rational/intelligent agent, we keep in mind PEAS analysis. PEAS analysis of the pacman game is as follows:

Agent → Pacman :

- Performance measure:** average score, win rate, computation time
- Environment:** maze containing coins, power capsules, ghosts
- Sensors:** screen display
- Actuators:** Eating and Moving

This section highlights the experimental results and insights gained from them. The experimental setup includes running a 10-fold iteration 10 times on each agent and then recording the **average score, average win rate, average time taken**. The diagrams below summarize all the experimental results for all the mazes.

4.1 Graphs



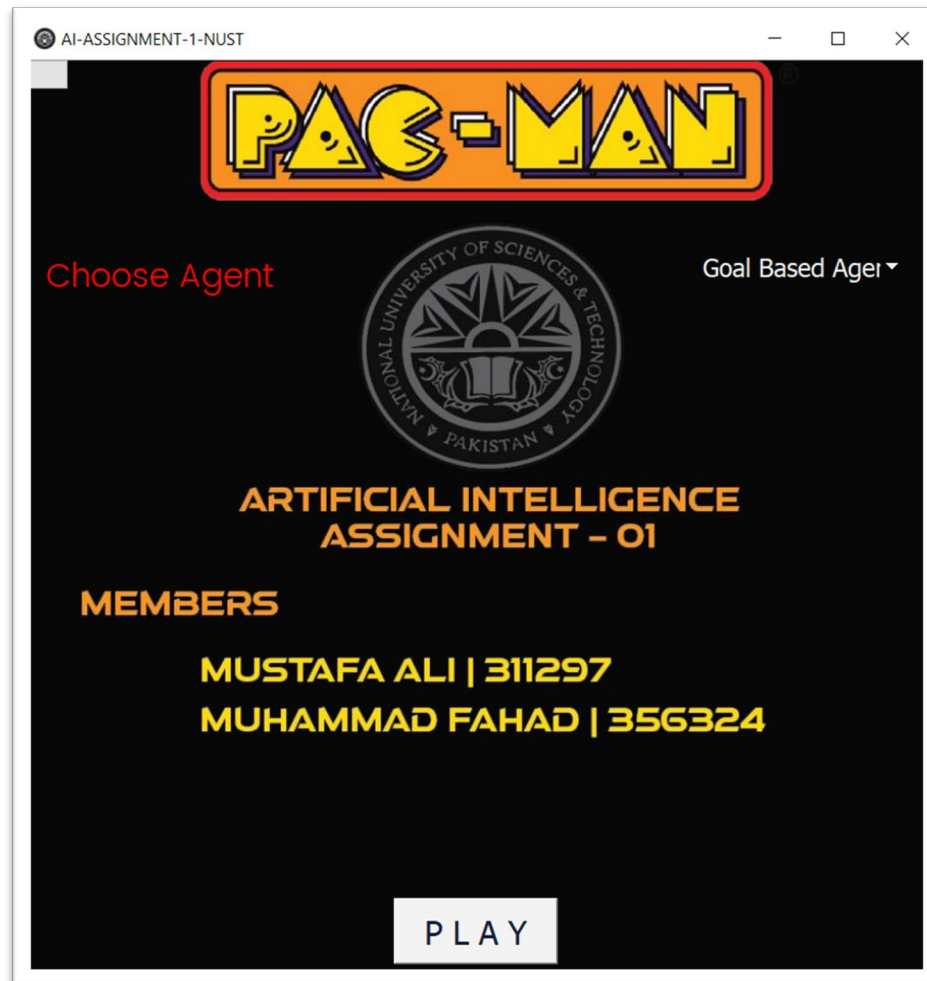
Goal Based Agent

```
Score: 2641 Game Won!  
Score: 2645 Game Won!  
Score: 3273 Game Won!  
Score: 2642 Game Won!  
Score: 3008 Game Won!  
Score: 2611 Game Won!  
Score: 2871 Game Won!  
Score: 2624 Game Won!  
Score: 2460 Game Won!  
Score: 2646 Game Won!  
10 games won out of 10 ---> Win Rate: 100.00  
Average Score: 2742.1  
  
Process took 55.015 seconds
```

Reflex Based Agent

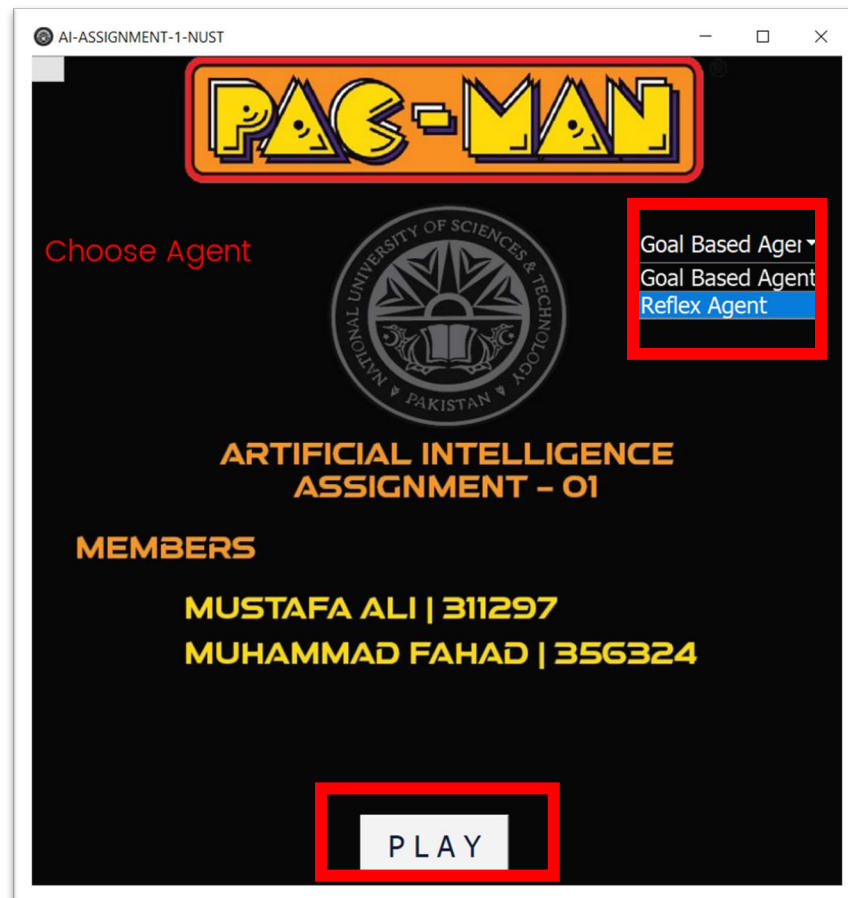
```
Score: 1339 Game Lost  
Score: 26 Game Lost  
Score: 45 Game Lost  
Score: -347 Game Lost  
Score: 2416 Game Won!  
Score: 772 Game Lost  
Score: 648 Game Lost  
Score: -293 Game Lost  
Score: 87 Game Lost  
Score: 2616 Game Won!  
2 games won out of 10 ---> Win Rate: 20.00  
Average Score: 730.9  
  
Process took 8.16 seconds
```


4.2 GUI



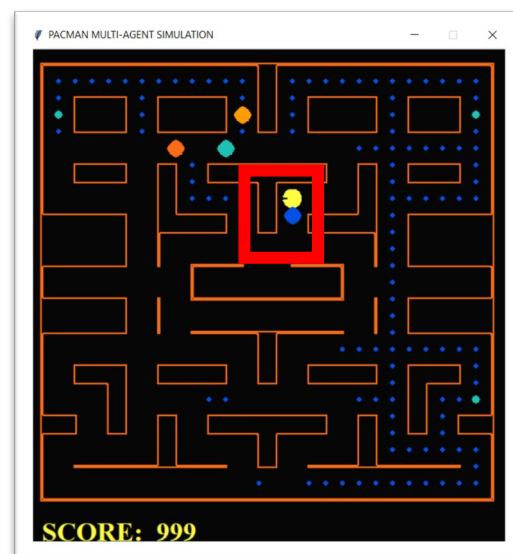
When we run the code it will display Graphic User Interface display, in this game we have Play Button which is used to start the game and also have two agents that are highlighted and listed below:

1. Goal Based Agent
2. Reflexive Agent



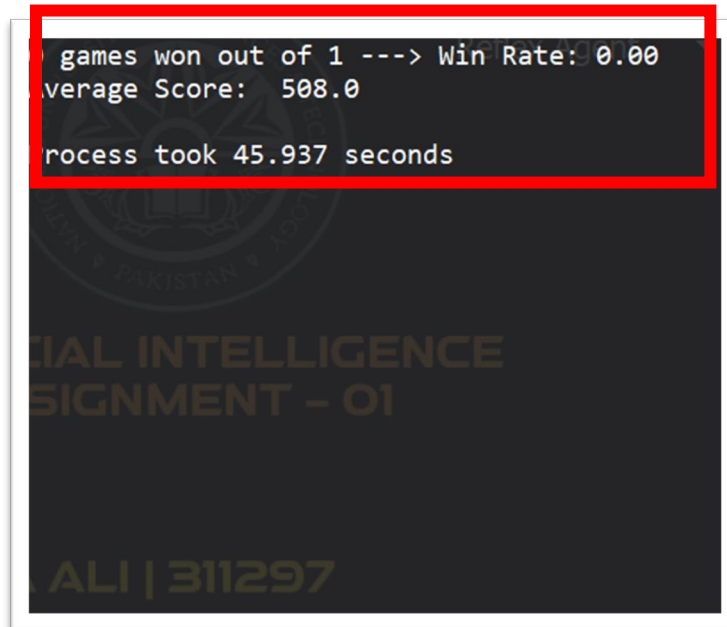
Now we will start the game with Reflexive Agent and observe the output.

4.2.1 REFLEXIVE AGENT SIMULATION



As we can see the agent scored 999 points and tried to ate all the coins, but agent failed to win the game.

REFLEXIVE AGENT SIMULATION – OUTPUT

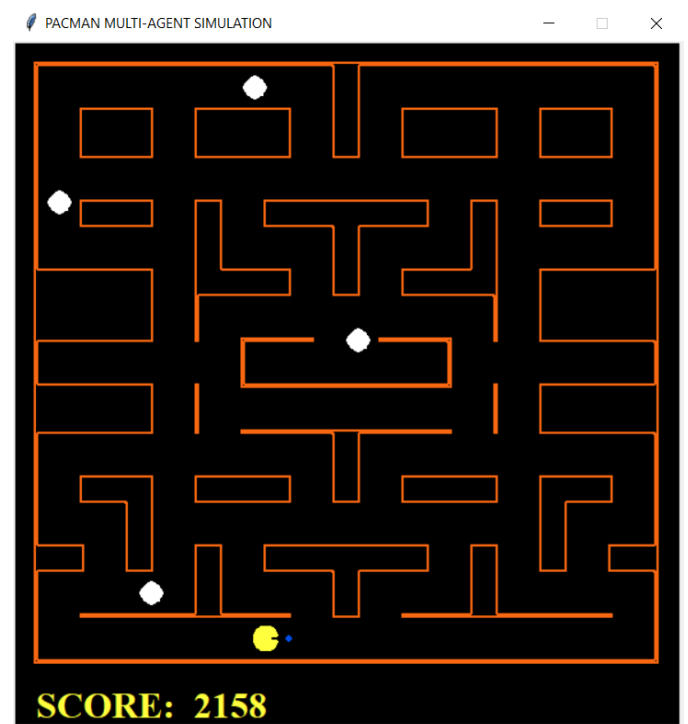
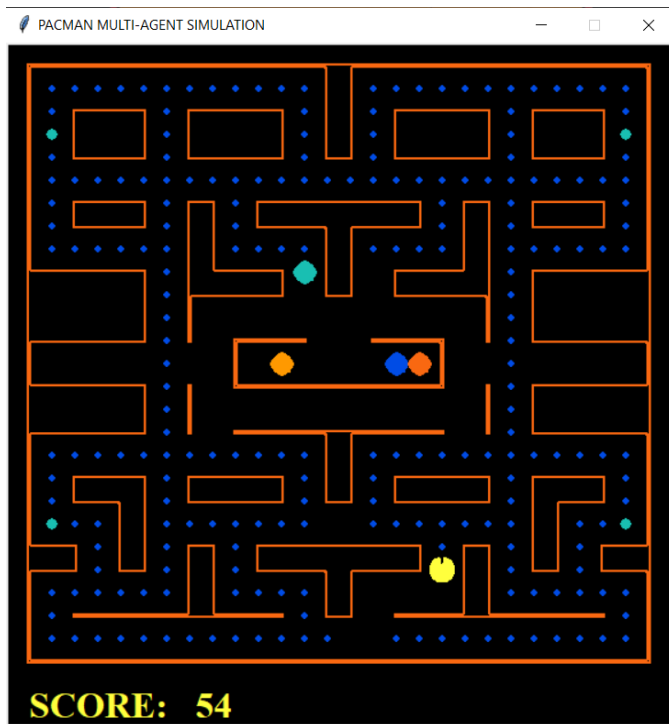


```
games won out of 1 ---> Win Rate: 0.00  
Average Score: 508.0  
Process took 45.937 seconds
```

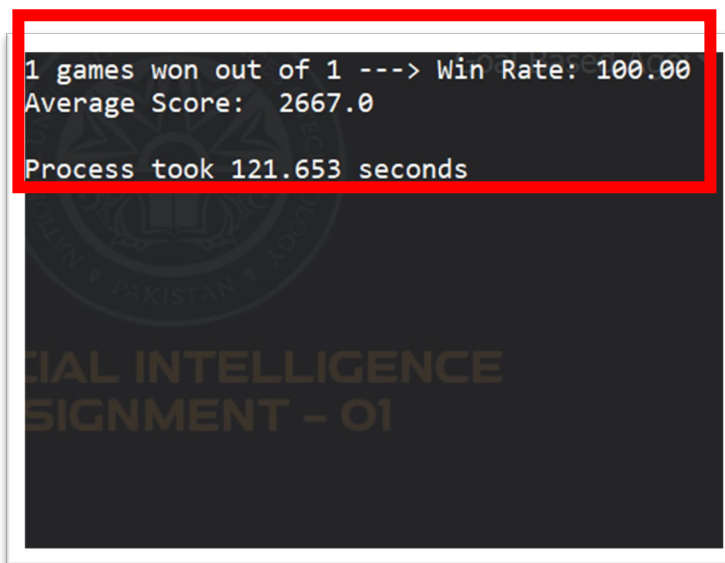
Output will show the statistics of Game winning and the average score of the agent, also it will show that how much time agent took to end this game.

4.2.2 GOAL BASED AGENT SIMULATION

Goal of this agent is to complete the game and score maximum and move smartly to avoid ghost in his way.



GOAL BASED AGENT SIMULATION – OUTPUT



As we can see in the above screen shots **Goal Based Agent** is successfully completed the game by acting smartly and scored maximum.

5.0 Contributions

Mustafa Ali :

- Developing AI for Pacman (Goal-Based Agent)
- Making game rules
- Collection of experimental data
- Coding of the Game and Environment

Muhammad Fahad:

- Developing AI for Pacman (Reflex Based Agent)
- Documentation
- Development of GUI

6.0 LINKS OF THE SIMULATION AND FILES

The video of the simulation is available in the link given below

https://drive.google.com/drive/folders/1Ti84A7KqpAyTsq1HrhfAp_FAaUFVG2NX

Thank You