UNIVERSITY OF LINCOLN

School of Computer Science

MSc Robotics and Autonomous Systems

CMP9794M

ADVANCED ARTIFICIAL INTELLIGENCE

Assessment-2-Resit

Sujith Karthikaiselvan

Student ID: 25916153

Submission Date:

TABLE OF CONTENT

ADVANCED ARTIFICIAL INTELLIGENCE

ASSESSMENT-2-RESIT

***Abstract*- Games have become more popular nowadays people spend most of their time playing video games. Arcade games are a type of video game where you have to collect rewards by avoiding the obstacle in your path. Where motion planning plays a major role in the creation of games. This AI assessment is to make the tallon move in the given environment using the MDP tool and collect rewards and avoid the meanies and pits in the fully and partially observable condition.**

KEYWORDS- Tallon, Markov Decision process, Directions, Utility, Value iteration, Policy Iteration,Q-learning, Fully and partial observability.

INTRODUCTION:

Motion planning for stochastic games like Pac man and so involves MDP(Markov Decision Process)[1]. Motion planning should work on both the conditions fully observable and partially observable. Where in the fully observable condition the character can see the environment clearly but in the partially observable condition the character can only till the specified visibility level.

MDP(Markov Decision Process) is a non-deterministic model where the agent acts on the given environment which is fully known[2]. It is used to model stoichastic environment

Creating a game involves motion planning like moving in any one of the four directions(north, south, east, and west). The motion planning should also be in the way that it should avoid the obstacle and collect the reward or reach the desired destination. Deciding which direction to take also plays a major role because the right path can lead to a reward others may mislead to the obstacle that is present in the environment.

In this assessment we have to make the tallon move in fully and partially observable condition.

Solution-A consist of the method used to develop the code for the tallon to move in the arena as long as possible in the fully observable condition. In solution-B the code is conducted for the partially observable condition. Where the visibility level of the tallon is reduced. And tallon can only view the environment to a certain level. In solution-C the size of the arena is changed and the scores are observed. Not only the size of the arena is increased also the number of pits in the environment is also increased.

RELATED WORKS

In this the researcher have considered the human as the player and the reward function is distributed evenly at the start of the game[3]. MDP method is used equential decision making tasks are commonly modeled as Markov decision processes (MDPs). Though MDPs are often addressed with reinforcement learning algorithms [7][3]. In this the MDP method is used in the agent as same as the tallon in our assignment. And the agent moves in the environment without any collosion with the obstacle.

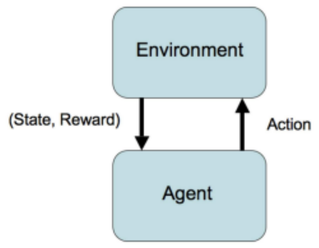


Figure 1:Simple working demo.

"*Learning to intercept opponents in first person shooter games*" by Tastan, B., Chang, Y. and Sukthankar, G[5] this paper examines the opponent interception, in which the goal of the bot is to reliably apprehend the opponent[5]. In this inverse reinforcement learning is used to track the opponents behaviour, Particle filter to find the enemies state over multiple time horizon and planner is used to intercept the best path.

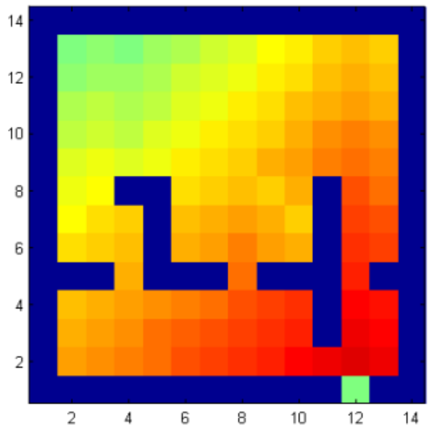


Figure 2: Final value map

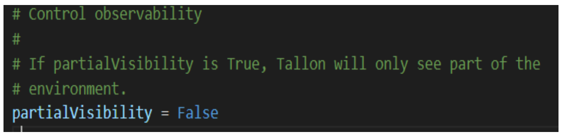
The above figure shows the final value map of the game in which the opponents repeated movement from the spawn to the end are calculated. This is helpful for the player to predict the next move of the player which is crucial in the shooting game that helps to score more point than the opponent. Which can also be used in the tallon program to find the random movements of the meanies.

"*Creating pro-level AI for a real-time fighting game using deep reinforcement learning*" by Oh, I., Rho, S., Moon, S., Son, S., Lee, H. and Chung, J[3]. implements the concept of learning the enemies move and training its performance according to the repeated action of the opponent.

SOLUTION-A

In the first part of the assessment we have to make the tallon to move in the environment by avoiding the meanies and the pits in fully observable condition. Where the tallon can view the environment without any restrictions.

Fully observable means that the agent can determine the systems state at all cost of time and predict the next move with the help of the information it gained in the environment.



```
# Control observability
#
# If partialVisibility is True, Tallon will only see part of the
# environment.
partialVisibility = False
```

Figure 3: Fully observable condition.

The above figure indicates that the observability can be changed by setting it to true and false conditions.

Fully Observability: After the system is fully visible by using the optimum values. It is easy for the Tallon to make decisions. In this condition the view for tallon is wide and can locate the meanies, pits and the bonus easily. The visibility level is set only if the partial visibility condition is true.

MDP's solution or policy is represented as π. But we would require an optimum policy.

$$\pi^* = \arg\max_{\pi} U^{\pi}(s)$$

Utility for every value are found this is used in MDP to find the best decision.

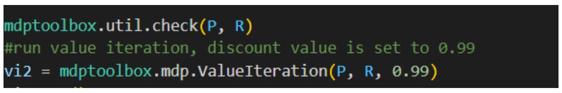
The steps of the tallon are decided in four directions north, south, east, and west. Tallon moves towards the coordinates of the bonus point that is present in the environment. The coordinates of the pit and the meanies are also found so that the tallon could avoid them and move towards the bonus points.

There are many techniques to solve this problem such as Policy iteration, value iteration, and Q-Learning.

The utility values are used in the value and the policy iteration which gives an optimum value for the problem.

The grid

Value Iteration: It is used to find the optimal value of MDP and the policy value. In the value iteration, it starts with an arbitrary value, and then the iteration is continued till it reaches an optimum value where the Tallon decision is decided appropriately. In this, we have given the arbitrary value as 0.99. After that, the policy iteration is done to find the position of the Tallon.



```
mdptoolbox.util.check(P, R)
#run value iteration, discount value is set to 0.99
vi2 = mdptoolbox.mdp.ValueIteration(P, R, 0.99)
```

Figure 4: MDP tool

In this MDP toolbox is used to find the utility, Value and policy iteration values.

Policy iteration: In policy iteration policy estimation and evaluation are done. In the improvement part, every time a new policy is created and in policy evaluation, an initial policy is created which is used in the improvement. The iteration will terminate if there is no further improvement. Value iteration and the Policy iteration are used to display the position of the tallon in the Grid and the next position of the tallon is also found with the help of Value and Policy Iteration. It also gives the detail of the coordinates of the nearest bonus, pit, and meanies so that it is easy to know the information about the environment.

Reinforcement learning: Is a branch of machine learning that studies how intelligent agents should respond in a given environment to maximize the concept of cumulative reward.

Q-Learning: Q-learning is an active reinforcement learning method that does not require the use of a model.

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

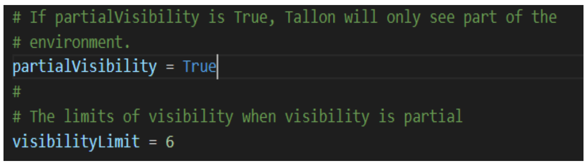
As Q-learning is a active reinforcement learning it should have updated action and state. So that it could used to find the next best step for Tallon.

The direction of the Tallon is North, South, East, and west which are decided based on the movement of the meanies so that it avoids it. And it also depends on the pit in the arena. The motion planning is easy in the case of fully observable as the Tallon has the knowledge about the environment. But in the case of partial visibility, it is vice versa.

SOLUTION-B

There are two types of observability they are Partial Observability and Fully Observability.

Partial Observability: The system dynamics are determined by an MDP. The POMDP framework can be used to simulate a wide range of real-world sequential decision processes. Robot navigation issues, machine maintenance, and general planning under uncertainty are all examples of applications. It is a stochastic process where the system can only see one step ahead. Optimal utility values and policy are found so that the tallon can decide a direction to move towards the bonus points avoiding meanies and pit.



```
# If partialVisibility is True, Tallon will only see part of the
# environment.
partialVisibility = True
#
# The limits of visibility when visibility is partial
visibilityLimit = 6
```

Figure 5: Partially visible.

In the above image, the partial visibility is set to true so that the Tallon can only view the environment to a certain level. And the visibility level is also set to 6 so that tallon can only see through 6 grids.

Deterministic policy: - A deterministic policy is a function that connects states and actions. The best deterministic strategy is the one that maximises the expected discounted sum of rewards if the agent follows it[5]. It is flexible to solve the problems in polynomial time. And can predict he next step of the problem.

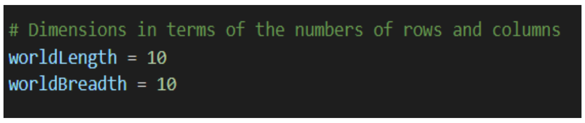
Non-deterministic policy: - A non-deterministic policy is defined as a function that translates each state S to a non-empty collection of actions represented by S(a). When the MDP is in state S, the agent can choose to perform any action a ∈ Π(s)[5]. It cannot determine its next step of the problem and it is not suitable for polynomial times.

In our case, we have taken the Partially Observable condition. Where the Tallon can only sense the bonus to a certain level. Where it is already coded. Whenever Tallon moves to a certain grid the next bonus coordinate which is nearer to the tallon is displayed but when tallon moves away from the bonus it does not move towards the bonus it moves randomly. As it does not know any information about the bonus in the partial observation.

The observability condition only affects when the size of the arena differs. As the number of meanies increases when tallon starts the game.

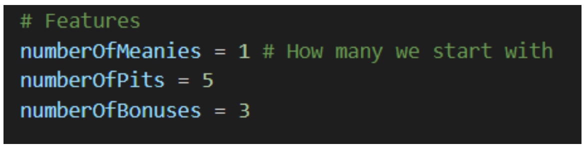
SOLUTION-C

Section-C is to change the size of the arena of the tallon and to increase the number of pits and the bonus station. So in our case, we have considered three conditions where in the first case it is a 10X10 matrix in this the tallon can move easily and it doesn't take much time for the tallon to collect the bonus in the arena. But when the number of pits increases the game becomes more difficult as the meanies also increase.



```
# Dimensions in terms of the numbers of rows and columns
worldLength = 10
worldBreadth = 10
```

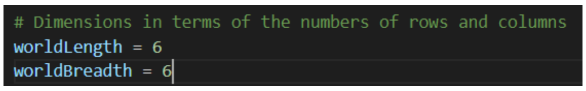
Figure 6: Changing the size of the Arena(10X10).



```
# Features
numberOfMeanies = 1 # How many we start with
numberOfPits = 5
numberOfBonuses = 3
```

Figure 7: Changing the number of Pits and Meanies.

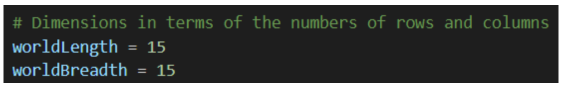
The size of the arena and the number of meanies spawning and the pit can be increased or decreased in the configuration section of the code as shown in the above two figures.



```
# Dimensions in terms of the numbers of rows and columns
worldLength = 6
worldBreadth = 6
```

Figure 8: Arena size 6X6

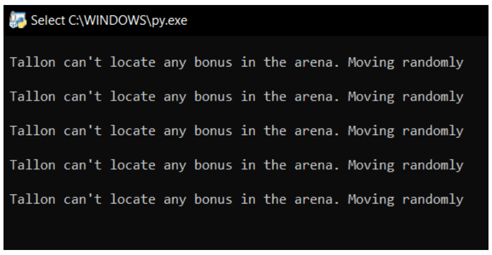
In case 2 the arena is reduced from a 10X10 matrix to a 6X6 matrix. This is where the real problem occurs. 6X6 makes it difficult for the tallon to move and also to avoid the meanies and the pits as there is no place for the tallon to move. Increasing the number of the pit in the 6X6 arena will make the tallon too difficult to move.



```
# Dimensions in terms of the numbers of rows and columns
worldLength = 15
worldBreadth = 15
```

Figure 9: Arena size 15X15

In case 3 we consider the 15X15 matrix, In this, the tallon moves freely in the arena because there is a lot of gap between the tallon, meanies, and pits. Increasing the number of pits and the speed of the gameplay will not affect tallon as the arena size is huge.



```
Select C:\WINDOWS\py.exe
Tallon can't locate any bonus in the arena. Moving randomly
Tallon can't locate any bonus in the arena. Moving randomly
Tallon can't locate any bonus in the arena. Moving randomly
Tallon can't locate any bonus in the arena. Moving randomly
Tallon can't locate any bonus in the arena. Moving randomly
```

Figure 10: Tallon can't locate the bonus

Whenever there is an increase in the size of the arena it makes it difficult for tallon to locate the bonus and the time required for tallon to collect the bonus also increases this is seen in the 15X15 matrix as shown in the figure 6. But it is not the same in the 10X10 and 6X6 matrix as the arena size is small compared to 15X15. It also makes it difficult for tallon to avoid meanies and pits in a small arena.

Increasing the number of pits in the partially observable true condition the difficulty level also increases as the tallon can only see certain grids but it is the opposite in the false condition.

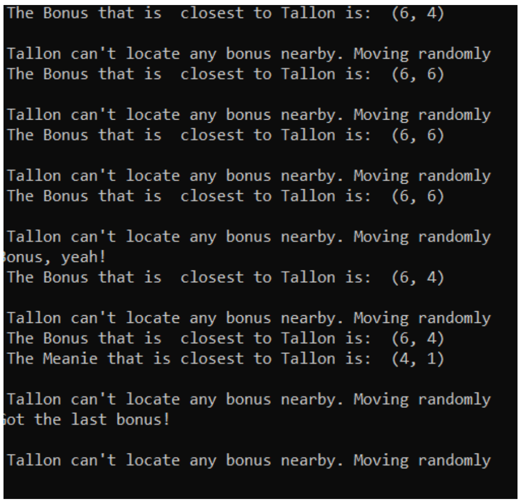
The change in size of the arena, iteration values and visibility outputs are presented in the appendices.

CONCLUSION

In this assessment, we have considered motion planning for the Tallon to move in the environment by avoiding the meanies and the pits. We have considered two conditions for visibility like fully and partially observable. By changing the visibility we can see that the Tallon motion planning becomes difficult. Tallon's output is taken for the different arenas like 10X10, 6X6, and 15X15. From that, we can see that the talon moves freely in the bigger arena but it is difficult to move in the smaller arena. In a bigger arena like 15X15, it is difficult for the talon to locate the meanies, bonus, and pit. But in the small arena, it is easy for the talon to locate the meanies, pits, and the bonus. Increasing the number of pits affects the talon movement in the small arena it does not have much effect on the bigger arena.

APPENDIX:

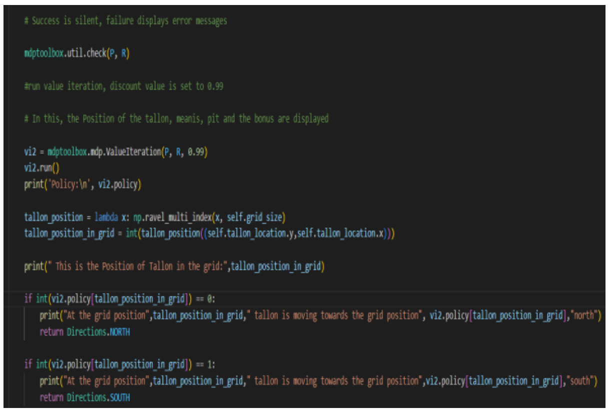
Task - 1



```
The Bonus that is  closest to Tallon is:  (6, 4)

Tallon can't locate any bonus nearby. Moving randomly
The Bonus that is  closest to Tallon is:  (6, 6)

Tallon can't locate any bonus nearby. Moving randomly
The Bonus that is  closest to Tallon is:  (6, 6)

Tallon can't locate any bonus nearby. Moving randomly
The Bonus that is  closest to Tallon is:  (6, 6)

Tallon can't locate any bonus nearby. Moving randomly
onus, yeah!
The Bonus that is  closest to Tallon is:  (6, 4)

Tallon can't locate any bonus nearby. Moving randomly
The Bonus that is  closest to Tallon is:  (6, 4)
The Meanie that is closest to Tallon is:  (4, 1)

Tallon can't locate any bonus nearby. Moving randomly
ot the last bonus!

Tallon can't locate any bonus nearby. Moving randomly
```

Figure 11: Tallon can't locate the bonus

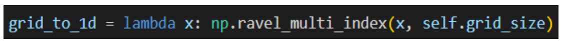
The above image indicates the Partial Observability of the Tallon as it moved away from the bonus it cannot locate the bonus. When it approaches the bonus, the bonus coordinates are known and Tallon moves towards it.



```python
# Success is silent, failure displays error messages

mdptoolbox.util.check(P, R)

#run value iteration, discount value is set to 0.99

# In this, the Position of the tallon, meanis, pit and the bonus are displayed

vi2 = mdptoolbox.mdp.ValueIteration(P, R, 0.99)
vi2.run()
print('Policy:\n', vi2.policy)

tallon_position = lambda x: np.ravel_multi_index(x, self.grid_size)
tallon_position_in_grid = int(tallon_position((self.tallon_location.y,self.tallon_location.x)))

print(" This is the Position of Tallon in the grid:",tallon_position_in_grid)

if int(vi2.policy[tallon_position_in_grid]) == 0:
    print("At the grid position",tallon_position_in_grid," tallon is moving towards the grid position", vi2.policy[tallon_position_in_grid],"north")
    return Directions.NORTH

if int(vi2.policy[tallon_position_in_grid]) == 1:
    print("At the grid position",tallon_position_in_grid," tallon is moving towards the grid position",vi2.policy[tallon_position_in_grid],"south")
    return Directions.SOUTH
```

Figure 12: Value, Utility and Policy iteration code.

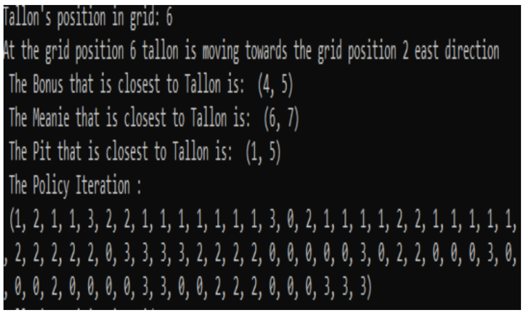
The above image displays the code that is used to find the utility values for finding the value iteration and to display the position of the tallon in the grid.



```python
grid_to_1d = lambda x: np.ravel_multi_index(x, self.grid_size)
```

Figure 13: Code to convert grid to 1D

The line of code in the figure is used to convert the grid to one dimension. This conversion increases the processing speed of the code.

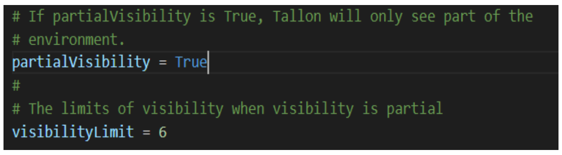


```
Tallon's position in grid: 6
At the grid position 6 tallon is moving towards the grid position 2 east direction
 The Bonus that is closest to Tallon is:  (4, 5)
 The Meanie that is closest to Tallon is:  (6, 7)
 The Pit that is closest to Tallon is:  (1, 5)
 The Policy Iteration :
 (1, 2, 1, 1, 3, 2, 2, 1, 1, 1, 1, 1, 1, 1, 3, 0, 2, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1,
, 2, 2, 2, 2, 2, 0, 3, 3, 3, 3, 2, 2, 2, 2, 0, 0, 0, 0, 0, 3, 0, 2, 2, 0, 0, 0, 3, 0,
, 0, 0, 2, 0, 0, 0, 0, 3, 3, 0, 0, 2, 2, 2, 0, 0, 0, 3, 3, 3)
```

Figure 14: Coordinates of bonus, meanies, pit and policy values.

In the above figure, the policy value is calculated using the Value iteration and utility and the closest coordinates of meanies, pits, and bonuses are also calculated.

TASK-2



```python
# If partialVisibility is True, Tallon will only see part of the
# environment.
partialVisibility = True
#
# The limits of visibility when visibility is partial
visibilityLimit = 6
```

Figure 15 : Setting partial visibility and the visibility level.

This is used to change the visibility from fully observable to partially observable.

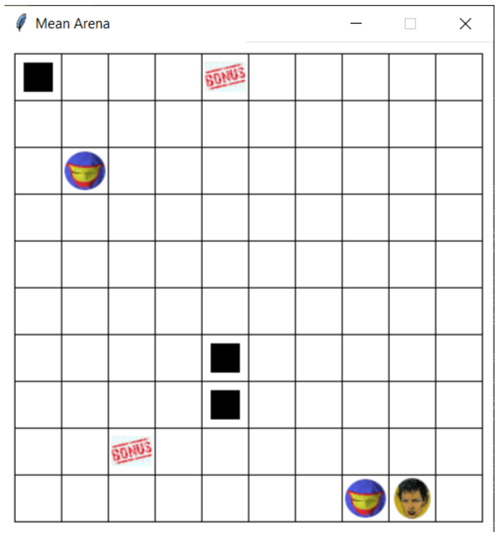
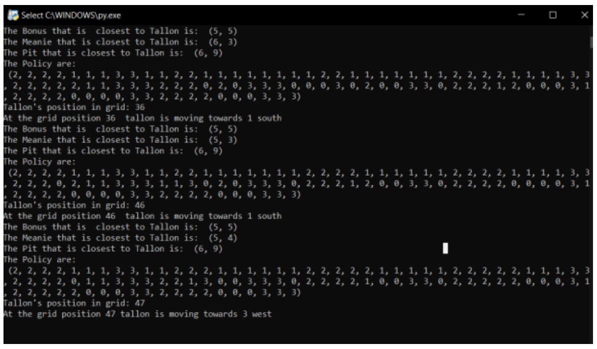


Figure 16: 10X10 Partially observable.

We have considered 10X10 matrix for the partial visibility.



```
Select C:\WINDOWS\py.exe
The Bonus that is  closest to Tallon is:  (5, 5)
The Meanie that is closest to Tallon is:  (6, 3)
The Pit that is closest to Tallon is:  (6, 9)
The Policy are:
 (2, 2, 2, 2, 1, 1, 1, 3, 3, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 3, 3
, 2, 2, 2, 2, 2, 1, 1, 3, 3, 3, 2, 2, 2, 0, 2, 0, 3, 3, 3, 0, 0, 0, 3, 0, 2, 0, 0, 3, 3, 0, 2, 2, 2, 1, 2, 0, 0, 0, 3, 1
, 2, 2, 2, 2, 0, 0, 0, 0, 3, 3, 2, 2, 2, 2, 0, 0, 0, 3, 3, 3)
Tallon's position in grid: 36
At the grid position 36  tallon is moving towards 1 south
The Bonus that is  closest to Tallon is:  (5, 5)
The Meanie that is closest to Tallon is:  (5, 3)
The Pit that is closest to Tallon is:  (6, 9)
The Policy are:
 (2, 2, 2, 2, 1, 1, 1, 3, 3, 1, 1, 2, 2, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 3, 3
, 2, 2, 2, 0, 2, 1, 1, 3, 3, 3, 1, 1, 3, 0, 2, 0, 3, 3, 3, 0, 2, 2, 2, 1, 2, 0, 0, 3, 3, 0, 2, 2, 2, 2, 0, 0, 0, 0, 3, 1
, 2, 2, 2, 2, 0, 0, 0, 0, 3, 3, 2, 2, 2, 2, 0, 0, 0, 3, 3, 3)
Tallon's position in grid: 46
At the grid position 46  tallon is moving towards 1 south
The Bonus that is  closest to Tallon is:  (5, 5)
The Meanie that is closest to Tallon is:  (5, 4)
The Pit that is closest to Tallon is:  (6, 9)
The Policy are:
 (2, 2, 2, 2, 1, 1, 1, 3, 3, 1, 1, 2, 2, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 1, 1, 1, 3, 3
, 2, 2, 2, 2, 0, 1, 1, 3, 3, 3, 2, 2, 1, 3, 0, 0, 3, 3, 3, 0, 2, 2, 2, 2, 1, 0, 0, 3, 3, 0, 2, 2, 2, 2, 2, 0, 0, 0, 3, 1
, 2, 2, 2, 2, 2, 0, 0, 0, 3, 3, 2, 2, 2, 2, 0, 0, 0, 3, 3, 3)
Tallon's position in grid: 47
At the grid position 47 tallon is moving towards 3 west
```

Figure 17: Output for partially visible.

TASK -3

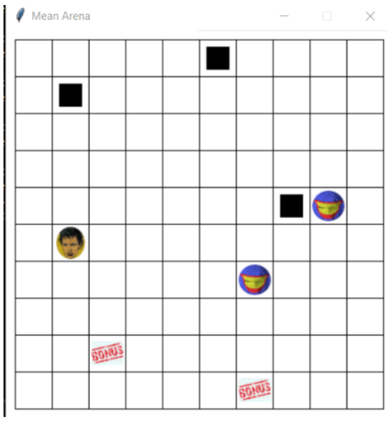
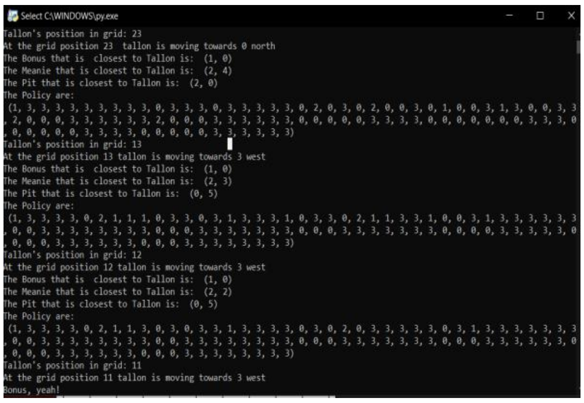


Figure 18: 10X10 Matrix

As mentioned in the section-c 10X10 matrix the tallon could move and locate the bonus point with the provided visibility level.



```
Select C:\WINDOWS\py.exe
Tallon's position in grid: 23
At the grid position 23  tallon is moving towards 0 north
The Bonus that is  closest to Tallon is:  (1, 0)
The Meanie that is closest to Tallon is:  (2, 4)
The Pit that is closest to Tallon is:  (2, 0)
The Policy are:
 (1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 3, 3, 3, 0, 3, 3, 3, 3, 3, 0, 2, 0, 3, 0, 2, 0, 0, 3, 0, 1, 0, 0, 3, 1, 3, 0, 0, 3, 3
, 2, 0, 0, 0, 3, 3, 3, 3, 3, 3, 2, 0, 0, 0, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 0
, 0, 0, 0, 0, 0, 3, 3, 3, 3, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 3)
Tallon's position in grid: 13
At the grid position 13 tallon is moving towards 3 west
The Bonus that is  closest to Tallon is:  (1, 0)
The Meanie that is closest to Tallon is:  (2, 3)
The Pit that is closest to Tallon is:  (0, 5)
The Policy are:
 (1, 3, 3, 3, 3, 0, 2, 1, 1, 1, 0, 3, 3, 0, 3, 1, 3, 3, 3, 1, 0, 3, 3, 0, 2, 1, 1, 3, 3, 1, 0, 0, 3, 1, 3, 3, 3, 3, 3, 3
, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 3, 3, 3, 3, 3, 0
, 0, 0, 0, 3, 3, 3, 3, 3, 3, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3)
Tallon's position in grid: 12
At the grid position 12 tallon is moving towards 3 west
The Bonus that is  closest to Tallon is:  (1, 0)
The Meanie that is closest to Tallon is:  (2, 2)
The Pit that is closest to Tallon is:  (0, 5)
The Policy are:
 (1, 3, 3, 3, 3, 0, 2, 1, 1, 3, 0, 3, 0, 3, 3, 1, 3, 3, 3, 3, 0, 3, 0, 2, 0, 3, 3, 3, 3, 3, 0, 3, 1, 3, 3, 3, 3, 3, 3, 3
, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 3, 3, 3, 3, 3, 3, 0, 0, 0, 3, 3, 3, 3, 3, 3, 0
, 0, 0, 0, 3, 3, 3, 3, 3, 3, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3)
Tallon's position in grid: 11
At the grid position 11 tallon is moving towards 3 west
Bonus, yeah!
```

Figure 19: 10X10 matrix iteration values for Partial visibility = True

The above figure gives the coordinates of the closest bonus, meanies and the pit in the 10X10 matrix.

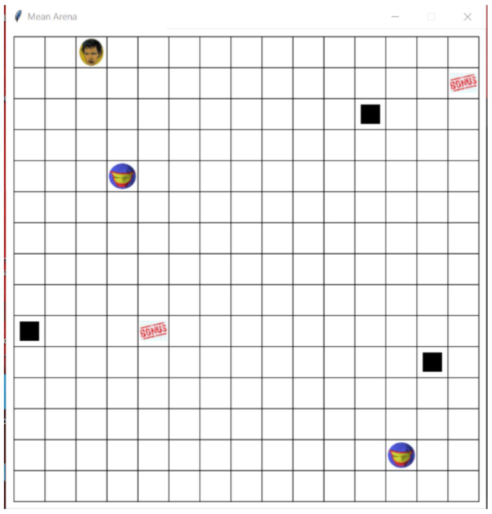
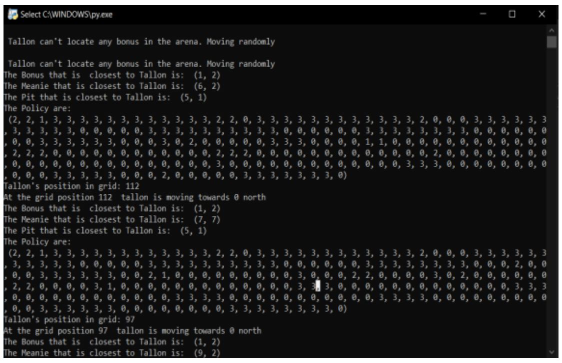


Figure 20: 15X15 Matrix

The above figure is 15X15 matrix in this the tallon has lot of space to move as the arena size is huge.



```
Select C:\WINDOWS\py.exe

 Tallon can't locate any bonus in the arena. Moving randomly

 Tallon can't locate any bonus in the arena. Moving randomly
The Bonus that is  closest to Tallon is:  (1, 2)
The Meanie that is closest to Tallon is:  (6, 2)
The Pit that is closest to Tallon is:  (5, 1)
The Policy are:
 (2, 2, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 0, 0, 0, 3, 3, 3, 3, 3, 3
, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0
, 0, 0, 3, 3, 3, 3, 3, 3, 0, 0, 0, 3, 0, 2, 0, 0, 0, 0, 0, 3, 3, 3, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 3, 3, 3, 3, 3, 0, 0, 0, 2, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 0)
Tallon's position in grid: 112
At the grid position 112  tallon is moving towards 0 north
The Bonus that is  closest to Tallon is:  (1, 2)
The Meanie that is closest to Tallon is:  (7, 7)
The Pit that is closest to Tallon is:  (5, 1)
The Policy are:
 (2, 2, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 0, 0, 0, 3, 3, 3, 3, 3, 3
, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 0, 0, 0, 2, 0, 0
, 0, 0, 3, 3, 3, 3, 3, 3, 0, 0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 2, 2, 0, 0, 0, 0, 3, 0, 2, 0, 0, 0, 0, 0, 0
, 2, 2, 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3
, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 3, 3, 3, 0)
Tallon's position in grid: 97
At the grid position 97  tallon is moving towards 0 north
The Bonus that is  closest to Tallon is:  (1, 2)
The Meanie that is closest to Tallon is:  (9, 2)
```

Figure 21: 15X15 matrix iteration value.

Figure shows the coordinates of the bonus and meanies it indicates that the location of bonus in 15X15 arena because of the visibility level.

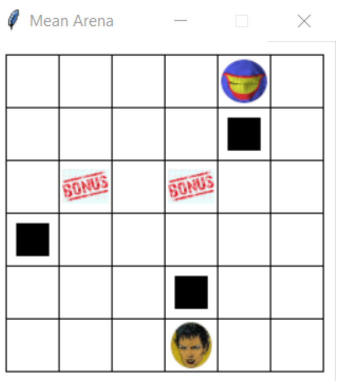
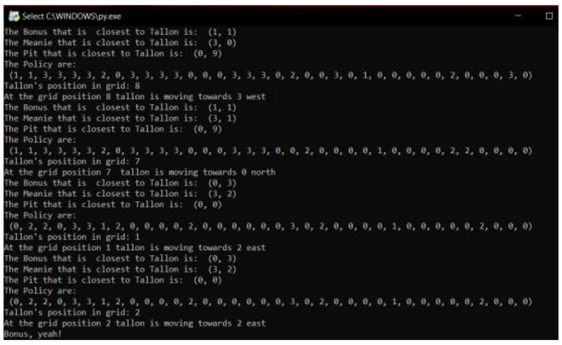


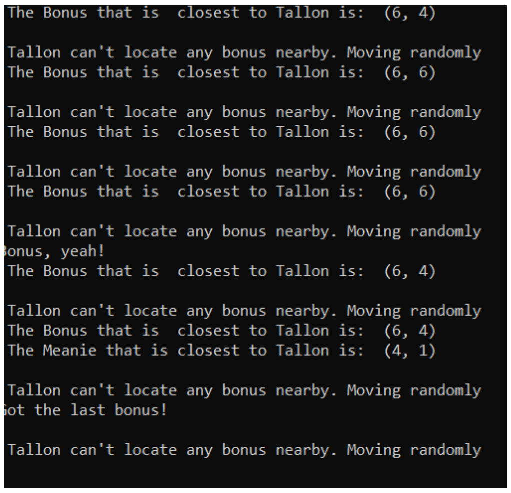
Figure 22: 6X6 Matrix



```
Select C:\WINDOWS\py.exe
The Bonus that is  closest to Tallon is:  (1, 1)
The Meanie that is closest to Tallon is:  (3, 0)
The Pit that is closest to Tallon is:  (0, 9)
The Policy are:
 (1, 1, 3, 3, 3, 3, 2, 0, 3, 3, 3, 3, 0, 0, 0, 3, 3, 3, 0, 2, 0, 0, 3, 0, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 3, 0)
Tallon's position in grid: 8
At the grid position 8 tallon is moving towards 3 west
The Bonus that is  closest to Tallon is:  (1, 1)
The Meanie that is closest to Tallon is:  (3, 1)
The Pit that is closest to Tallon is:  (0, 9)
The Policy are:
 (1, 1, 3, 3, 3, 3, 2, 0, 3, 3, 3, 3, 0, 0, 0, 3, 3, 3, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0)
Tallon's position in grid: 7
At the grid position 7  tallon is moving towards 0 north
The Bonus that is  closest to Tallon is:  (0, 3)
The Meanie that is closest to Tallon is:  (3, 2)
The Pit that is closest to Tallon is:  (0, 0)
The Policy are:
 (0, 2, 2, 0, 3, 3, 1, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 3, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0)
Tallon's position in grid: 1
At the grid position 1 tallon is moving towards 2 east
The Bonus that is  closest to Tallon is:  (0, 3)
The Meanie that is closest to Tallon is:  (3, 2)
The Pit that is closest to Tallon is:  (0, 0)
The Policy are:
 (0, 2, 2, 0, 3, 3, 1, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 3, 0, 2, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0)
Tallon's position in grid: 2
At the grid position 2 tallon is moving towards 2 east
Bonus, yeah!
```

Figure 23: 6X6 Matrix iteration values for partial visibility = True

As mentioned before it is difficult for the tallon to move in 6X6 even if the visibility is true or false.



```
The Bonus that is  closest to Tallon is:  (6, 4)

Tallon can't locate any bonus nearby. Moving randomly
The Bonus that is  closest to Tallon is:  (6, 6)

Tallon can't locate any bonus nearby. Moving randomly
The Bonus that is  closest to Tallon is:  (6, 6)

Tallon can't locate any bonus nearby. Moving randomly
The Bonus that is  closest to Tallon is:  (6, 6)

Tallon can't locate any bonus nearby. Moving randomly
onus, yeah!
The Bonus that is  closest to Tallon is:  (6, 4)

Tallon can't locate any bonus nearby. Moving randomly
The Bonus that is  closest to Tallon is:  (6, 4)
The Meanie that is closest to Tallon is:  (4, 1)

Tallon can't locate any bonus nearby. Moving randomly
ot the last bonus!

Tallon can't locate any bonus nearby. Moving randomly
```

Figure 24: Output window for 15X15 matrix when the tallon is too far from bonus and obstacles.

As the arena 15X15 is huge tallon cant locate the bonus in the environment**.**

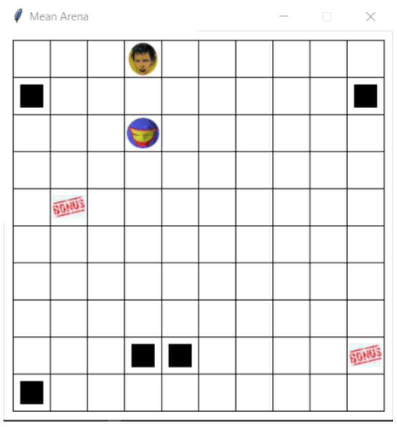


Figure 25: Increase in the number of pit from 3 to 5 in 10X10 matrix.

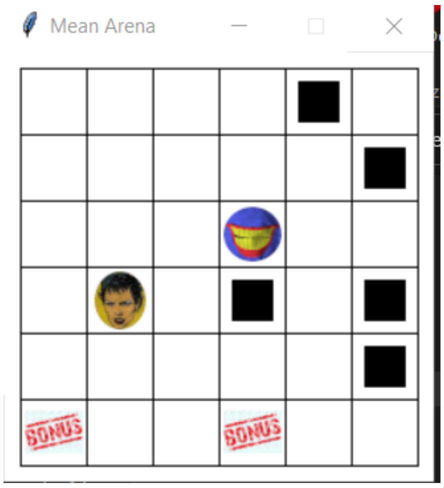


Figure 26: Increase in the number of pit from 3 to 5 in 6X6 matrix.

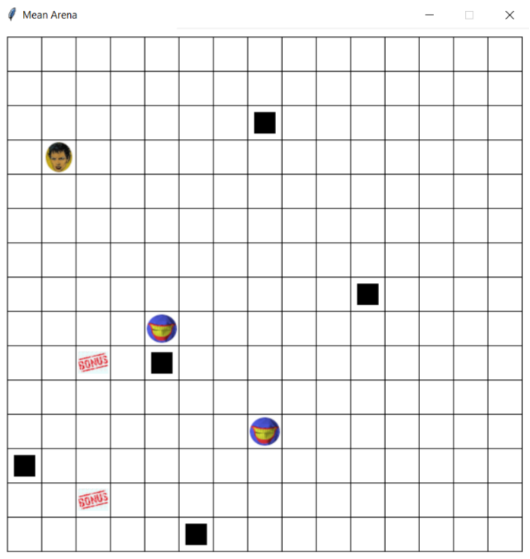


Figure 27: Increase in the number of pit from 3 to 5 in 15X15 matrix.

Increasing the number of the pit does not affect the bigger environment it only affects the smaller one. In our case, it affects the 6X6 matrix.

REFERENCES:

1]Winterer, L., Junges, S., Wimmer, R., Jansen, N., Topcu, U., Katoen, J.P. and Becker, B., 2017, December. Motion planning under partial observability using game-based abstraction. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)* (pp. 2201-2208). IEEE.

2] Brechtel, S., Gindele, T. and Dillmann, R., 2011, October. Probabilistic MDP-behavior planning for cars. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)* (pp. 1537-1542). IEEE.

3] Oh, I., Rho, S., Moon, S., Son, S., Lee, H. and Chung, J., 2021. Creating pro-level AI for a real-time fighting game using deep reinforcement learning. *IEEE Transactions on Games*.

4] Knox, W.B. and Stone, P., 2008, August. Tamer: Training an agent manually via evaluative reinforcement. In *2008 7th IEEE international conference on development and learning* (pp. 292-297). IEEE.

5] Bozkurt, A.K., Wang, Y., Zavlanos, M.M. and Pajic, M., 2021, May. Model-free reinforcement learning for stochastic games with linear temporal logic objectives. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 10649-10655). IEEE.

6] Tastan, B., Chang, Y. and Sukthankar, G., 2012, September. Learning to intercept opponents in first person shooter games. In *2012 IEEE Conference on Computational Intelligence and Games (CIG)* (pp. 100-107). IEEE.

[7] Sutton, R., 1998. R. and A. Barto, Reinforcement learning.