# Automated Pacman Game – Report 2

## Introduction

Within this project I have created a Pacman game that is fully automated. I have achieved this by making an agent that will consider the layout of the world around it and make decisions on its movement based on what it thinks the best outcome could be. I created this type of agent with the ability to take into consideration all its possible future actions and make a decision for movement based on what it could see in front of it. This will lead to a 'smarter' agent that will be more effective at completing the game efficiently.

## Description – Strategy

To complete the task of making pacman win on the 'smallGrid' map and 'mediumClassic' I decided that the first thing I needed to do was assign values to all the locations on the map, depending what was in the location. To do this I accessed its location through the api and created a dictionary to hold this location and it's assigned value, this value is important as it will help pacman decide which is the best way to go. The next task was to find the transition value for each possible move pacman could make: I would calculate the maximum expected utility for North, South, East, West and 'Wait', which meant pacman stayed in the same place. To do this I multiplied the value of the location by the probability it would move in that direction (lecture 4). When I had this value for each direction, I stored them in the dictionary and updated the map with the values I currently can calculate.

After I had an initial value for the spaces around the pacman I applied the bellman equation with each of the values that are currently stored in the dictionary which allows me to access the utilities of the other locations in the map. When I had these values, it was then time to reapply maximum expected utility, this would allow me to calculate the best possible path to gain the most utility, or in terms of the game, go to the most food / capsule locations. The final thing to do was check that the move was legal then, if so, move in in that direction.

After these calculations have been made and pacman makes a move, the whole process starts again on the updated version of the map. This was key when a ghost was involved in the game as their movements directly affect the route pacman will choose.

## Description – Methodology

When implementing my strategy I ran into multiple problems, the first of which was finding the utility of each of the directions. My first method of finding the utilities was to set the utility at the same place where it is assigned its value in the 'updateFoodInMap' function. I found that this was unsuccessful as whenever I wanted to access the utility, I had to state its location as well. I could foresee problems with this method so decided that the best way to assign utilities was to create a dictionary to hold its value then access record when I wanted to know the utility of a certain object. I saw that pacman would not be able to access utilities if they were set at the same location as the drawing of the map.

Another challenge I had to face was that pacman would traverse the map up until he reached the space before the food (on map smalMDPGrid). After investigation I found the problem was within the utility that I set for the food. As this was at 2 when I first implemented the strategy, it meant that pacman would not loose enough utility per ‘Wait’ for it to be economical to finish game. To fix this I increased the utility of the food and increased the discount value, this meant that it made sense for pacman to get the food and finish the game. I saw pacman lost most of the time on ‘smallGrid’ when it used this strategy as it would often get eaten by the ghost waiting for last move.

I also had to change the methodology a few times within the bellman equation, I originally set the loops to 10. I saw that pacman would often still run into ghosts if the number of loops was too low, this was because it had not considered all the possibilities enough times so make a reliable decision. After seeing pacman fail more often than not in ‘smallGrid’ with this number of loops, I increased this number to 100 as this should find all of the possible routes easily. I saw that pacman failed often on ‘smallGrid’ when it used this strategy as it would often stop for a considerable time while it was calculating all the possibilities. This meant that the ghost would often catch up to it and the game would be over. I finally decided that 25 loops were a sufficient amount of loops for the ‘smallGrid’ map as it allowed pacman to make the correct decision and not wait in place for too long calculating all the moves.

Another problem with my code arose when I started testing ‘smallGrid’ over multiple times. I was finding that the food items were already set to seen and so pacman would not want to go into those locations, thus meaning that he would loose as the ghost would eventually catch up with him. To fix this I had to clear the ‘seen’ variables from the ‘mapValues’, this meant that nothing would be considered seen when the game reloaded.

A bug that I came across towards the end of my development comes from the starting of mediumClassic map. As I had been doing all my testing in the ‘smallMDPGrid’ or in ‘smallGrid’ I only encountered the problem that mediumClassic does not run due to the locations outside of the size of the smallGrid not being in the dictionary until very late in the process.

## Tests

The tests I conducted throughout the process of creating the game range from testing the ability to move without errors, to increasing likelihood of success with ghosts. Test 1 is pacman playing the game on layout “smallGrid”, this test is looking to see if the pacman can navigate in a map successfully without any errors; this is using the same class that I submitted for coursework 1. Test 2 is testing if I can read the utilities of each location when setting it in the same place as grid creation. Test 3 is played on layout ‘smallGrid’, and tests to see if the dictionary is accessible and contains the right utilities for the right object it is, this is testing the setting and getting of utilities. Tests 4, 5 and 6 are all testing different utilities for the various objects in the game. Test 4 is for the utility of the food, test 5 is the discount value / reward for staying in place and moving and test 6 is for the utility of the ghost. These tests will be closely related as a balance is needed between them to make pacman successful.

Test 7 is testing the amount of times the bellman equation should be ran, I am testing this on ‘smallGrid’ and will aim to find the optimum number of loops to get the best path for pacman. Test 8 is to find out how many times pacman will win the game over 10 iterations of the game on ‘smallGrid’. Test 9 is finding out how well pacman performs over 10 iterations on ‘mediumClassic’.

## Results

For Test 1 the results are as follows. When running the agent in any layout pacman can navigate successfully without crashing, although without any purpose thus we can conclude that the agent is only smart enough to recognise walls and to remove any illegal options from its next choice.

For Test 2, I tried to run the agent with the utility set in the grid formation. The pacman could not read the utility of the food in the map as it was not accessible without knowing the exact location of each food item. Thus, this test has failed.

For Test 3, having unsuccessfully implemented the utility into the game I decided to change my method and use a dictionary to store the varying locations and their utilities. When I had this implemented in my code you could see each utility in its place on the grid, meaning that it was accessible for pacman to see. When these utilities were available to pacman it would make one move in the direction of the highest utility. When running this in 'smallMDP' there was some success as pacman would reach the food but depending on what the discount value and reward where he could spend a long time to reach the goal.

Having seen the results of test 3 (smallMDPGrid), it became obvious the utility of each space was key for the success of pacman, so this is what I was testing in test 4. Having started with the utility for food at 1, I soon realised that this was too low, considering the reward for movement was near 1. After this initial test lead pacman to not needing to leave the game to gain a similar amount of utility. This lead to test 5 (smallMDPGrid) where I was changing the reward and discount value for each of the moves pacman made, to make it fit in line with food value of 1, I lowered the reward to 0.5 leaving the discount at 0.3, this lead the pacman to not wanting to move as there was little difference between moving and not. When I increased the food value to 10, it stopped caring about the value of the ghost as the reward for getting the food was worth taking the risk. Finally, after a few changes I settled on value for food to be 3, reward to be 0.2 and discount to be 0.5. For test 6 I used the map 'smallGrid', this is so I can see the effect of the ghost on the utilities. This utility started at 0 but I soon realised this was the same value that I was assigning the seen spaces so pacman would still go there on some occasions. I decreased the value to -4  as to make sure that any route with the ghost in it should not be considered.

When I was testing the amount of times the bellman equation should be run (test 7), I started at 10 as I thought this would allow pacman to definitely find the best route to the food. However, when I ran the program on 'smallGrid' I saw that pacman wouldn't always choose the best path on 10 iterations. Seeing these results, I increased the iterations to 100 then decreased down to 50 as when there was so many iterations pacman often waited in place while it calculated the best route, allowing the ghost to catch up and ending the game.

For test 8, when I had the utilities set correctly, I saw that pacman won the first game successfully, but all the other games resulted in a loss. I found that as I was not clearing the 'seen' objects, all of the food utility was set to 0. This meant pacman would not go to this value and wait in the location it was in until the ghost came and ended the game. When I cleared the seen values and ran the game 10 times it won the game roughly 6 out of 10 times on 'smallGrid'. Pacman wouldn't win every game as sometimes when collecting the food in the middle of the map the ghost would appear behind it and then end the game, other times pacman would turn back into pacman when coming from collecting that piece of food.

For test 9, I was originally not able to run the game on map ‘mediumClassic’, this is because the ‘getHeight()’ and ‘getWidth()’ functions were the wrong way around when running the value iteration. To fix this I made sure that ‘getWidth()’ was always checked first as when creating the map in ‘setValue’ I am first asking for the [y] then [x] which is the height.

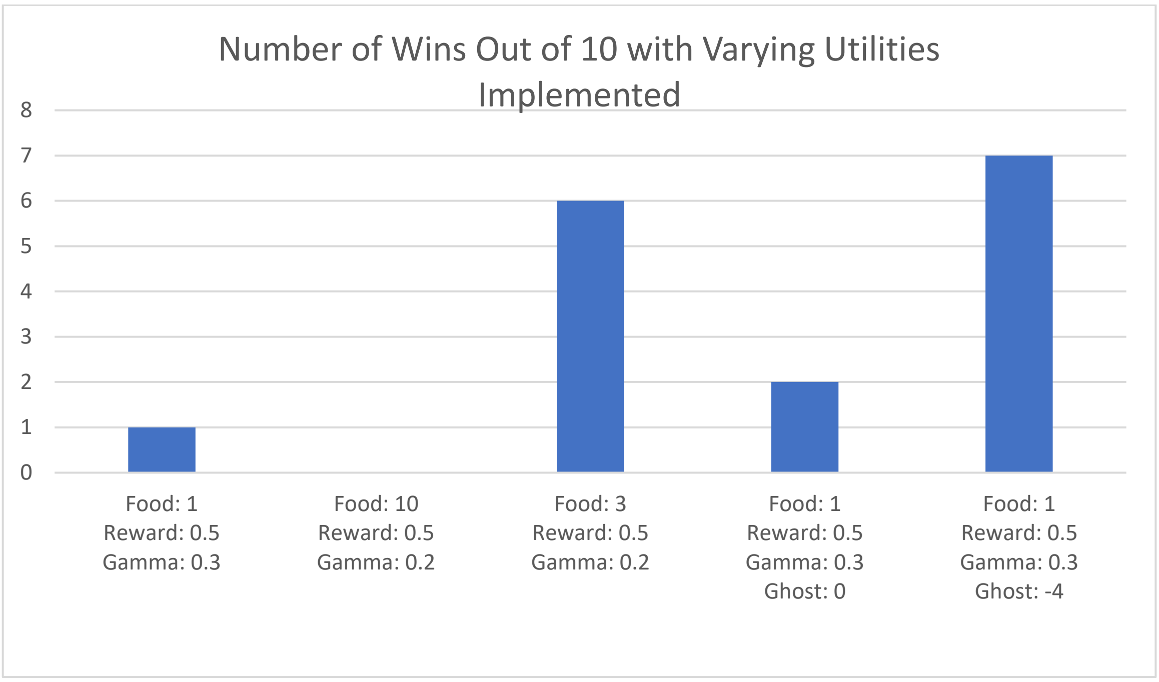

*Table 1 - Average number of wins per test (test 4,6 and 6)*

| Test Name | Intention | Outcome | Pass / Fail | Comments | Changes | No of times Tested |
|---|---|---|---|---|---|---|
| Movement Test – No Ghosts | Pacman to move through map, changing direction when running into a wall | Pacman moves successfully without any system errors | Pass | N/A | N/A | 2 |
| Utility Searching | Pacman to access utilities of each objects spread across the map | Pacman does not see utility. Utilities not shown on grid | Fail | Unable to see utilities | Change method for accessing utilities | 10 |
|  | Pacman to access utilities of each objects spread across the map | Pacman sees utility. Utilities shown on grid | Pass | When implemented dictionary to hold values, utilities set and accessible | N/A | 5 |
| Changing of utilities – No Ghosts | Pacman to run into the areas of the map with food | Pacman doesn't always pick up the food | Fail | (Food: 1, Reward: 0.3, Discount: 0.5) Pacman doesn't always pick up food as utility of staying is just as beneficial | Change utility for food | 10 |
|  | Pacman to run into the areas of the map with food | Pacman always goes for food no matter what is in front of it | Fail | (Food: 10, Reward: 0.3, Discount: 0.5, Ghost: 0) As pacman will always go for food it will still go through ghosts as the utility is worth the risk | Change utility of food | 10 |
|  | Pacman to run into the areas of the map with food | Pacman finds the food successfully on 'smallMDPGrid' | Pass | (Food: 3, Reward: 0.2, Discount: 0.5) | Add ghosts to equation | 10 |
| Changing of utilities - Ghost | Pacman to run into the areas of the map with food and avoid ghosts | Pacman doesn't move away from ghost | Fail | (Food: 3, Reward: 0.2, Discount: 0.5, Ghost: 0) As the utility of the ghost isn't low enough, pacman will still use that location | Decrease ghost utility | 10 |
|  | Pacman to run into the areas of the map with food and avoid ghosts | Pacman moves away from ghost | Pass | (Food: 3, Reward: 0.2, Discount: 0.5, Ghost: -4) Pacman will not move away from ghosts if this value doesn't update as it will not think it is getting closer to a ghost | N/A | 10 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Bellman Equation Iterations | For pacman to make the right decision in a suitable amount of time | Pacman stalls when making a decision | Fail | (Iterations: 100)As there is so many iterations it takes time for pacman to run all the iterations | Decrease iterations | 5 |
| | For pacman to make the right decision in a suitable amount of time | Pacman doesn't always avoid the ghost | Fail | (Iterations: 10) Pacman will sometimes run into the ghost as it hasn't considered the complete outcome | Increase iterations | 10 |
| | For pacman to make the right decision in a suitable amount of time | Pacman avoids ghost and collects food | Pass | (Iterations: 10)Pacman avoids ghosts and wins most of the time when running individual game | N/A | 10 |
| Winning the game enough times on 'smallGrid' | To see if pacman would win at least 4/10 games in smallGrid | Pacman wins first game but fails in the rest of the games | Fail | As I am not clearing the values at the end of the game the food is set to seen when the next game starts | Clear all mapValues at the end | 15 |
| | To see if pacman would win at least 4/10 games in smallGrid | Pacman wins roughly 6/10 | Pass | Pacman wins the majority of games. | N/A | 10 (x5) |
| Running the game on 'mediumClassic' | To see pacman's performance on 'mediumClassic' | The game does not currently load on this or any map | Fail | 'getHeight()' and 'getWidth()' functions were the wrong way around when running the value iteration | Ensure widths are checked first at all points | 20 |
| | To see pacman's performance on 'mediumClassic' | Pacman navigates around the map | Pass | Pacman now traverses the map but is not often successful in winning the game | Change utilities | 10 |
| Winning the game enough times on 'mediumClassic' | To see if pacman would win at least 2/10 games in mediumClassic | Pacman is successful 2/10 times | | (Food: 3, Reward: 0.2, Discount: 0.8, Ghost: -10) Pacman is not very successful with the utilities currently implemented. Also, to make performance better could set a closest distance allowed to ghost to stop pacman being near ghosts at all | | 10 (x5) |