Nicholas Allaire, A10639753
Marcel Aguiar, A10904034
Son Tang, A11370127

Programming Assignment 2 Write Up

Question 1. *Explain briefly the features you considered while writing the evaluation function and its shortcomings (if any).*

For this evaluation function, we calculated the closest food to the Pacman and made it more valuable than food that are farther away. To make sure that the Pacman does not die, we check to see if the nearest ghost is within 2 manhattan spaces of Pacman, and make the Pacman avoid being within that space. We have also tried calculating the total combined distances of the food, to pull the Pacman closer to areas with food, but this has limited use when it is within a large radius around the foods. We also made pacman try to stay as far away from the ghost as possible, but the Pacman tended to stay in a corner to avoid the ghost.

Question 2. *On larger boards such as openClassic and mediumClassic (the default), you'll find Pacman to be good at not dying, but quite bad at winning. He'll often thrash around without making progress. He might even thrash around right next to a dot without eating it because he doesn't know where he'd go after eating that dot. Can you explain the reason for this behaviour? (Hint - Attempt question 5 before answering this*

It will depend on the given evaluation function. This behavior is probably caused because Pacman cannot see past the states that are immediately around it. Therefore, Pacman does not know where to go to increase the score, and will just thrash around. However, Pacman will be able to see and avoid ghosts that get close, because the score will decrease so it is good at not dying. Another reason may be the priority to stay away from the ghost is too high, and eating food is seen as not that important to Pacman. So it will just try to stay as far from the ghost as possible, even if it means forgoing the nearby food.

Try the following scenario. Press any key to step forward. What do you observe? Explain the behaviour.

The Pacman walks towards the ghost that is nearby rather than going away from it, even if it meant going away from the food. This behavior is probably because the Pacman wants to maximize the score that the game ends with, and since running away from the ghosts is futile and decreases the points further by moving, the Pacman runs toward the nearby ghost to end the game early.

Question 4. *You should find that your ExpectimaxAgent wins about half the time, while your AlphaBetaAgent always loses. Can you explain why the behavior here differs from the minimax case?*

The AlphaBeta Agent probably always loses because the Pacman always assumes the worse, and reacts accordingly. This may cause the Pacman to perform an action that loses the game, as a self-fulfilling prophecy, to minimize point loss. The Expectimax Agent, on the other hand, does not know what the ghosts will do, and calculate the chances of the ghost doing a move that will make Pacman lose or allow Pacman to survive. If it thinks that the ghost will kill it, it will run towards it to minimize point loss, but if it thinks the ghost will let it live, then it will win. Therefore, the Pacman will assume that the ghost will kill it half the time, and allow it to live the other half, and so the Expectimax Agent wins about half the time.

Question 5. *Explain briefly the features you considered while writing the evaluation function and its shortcomings (if any).*

The goal of our evaluation function is to maximize the score. To maximize the score, the Pacman ate nearby food by minimizing the pellets and capsules on the board. To simultaneously ensure the safety and movement of the Pacman moved to minimize the distance to ghosts. We tried minimizing the total food distance to bring Pacman closer to areas with food. However, this had limited use most of the time as the radius for this function to be ineffective is large. We also calculated the distance the ghost is from the Pacman so that the Pacman tries to stay at least 2 spaces away from the nearest ghost. We found that this was made redundant when we tried to maximize the score, which made the Pacman avoid the getting eaten by nearby ghosts by itself. We also made the Pacman look for the closest food. However, this did not work because the Pacman looks at multiple spaces (four directions) and the closest food can be different for each of those spaces, and the Pacman does not know which food to eat if the spaces are equal. We also tried to make Pacman avoid the walls, but it did not do much to help Pacman win or survive. We also found that the Pacman was frozen in a lot of cases where it does not know how to maximize the score, or where the food is. To combat this, we tried to implement a random function with a small priority that made the Pacman move randomly in the case where all the choices are equal. This somewhat worked in lucky cases, but Pacman mostly thrashed back and forth around the same areas. This problem was fixed when we made Pacman want to maximize the distance from the ghosts, who already move randomly. This way Pacman not only always moves, but also has a general direction of where to move.