COEN 166 Artificial Intelligence

Lab Assignment #5: MultiAgent I

Gagan Gupta **SCU#00001478479**

Problem: Reflex Agent

Function 1:

```
   def evaluationFunction(self, currentGameState, action):
      # Useful information you can extract from a GameState (pacman.py)
      successorGameState = currentGameState.generatePacmanSuccessor(action)
      newPos = successorGameState.getPacmanPosition()
      newFood = successorGameState.getFood()
      newGhostStates = successorGameState.getGhostStates()
      newScaredTimes = [ghostState.scaredTimer for ghostState in newGhostStates]

      MAX_VAL=float("inf")
      MIN_VAL=-MAX_VAL

      #Find distance to closest food and ghost
      foodDistance=[manhattanDistance(pellet,newPos) for pellet in newFood.asList()]
      ghostDistance=[manhattanDistance(ghost,newPos) for ghost in
successorGameState.getGhostPositions()]
      if len(foodDistance)==0:
         return MAX_VAL
      minFD=min(foodDistance)
      minGD=min(ghostDistance)
      foodCount=newFood.count()

      #If any ghost is too close then we return min
      if minGD<=2:
         return MIN_VAL
      #If the ghost is next to food and far enough away from the ghost we get the food
      elif currentGameState.hasFood(newPos[0],newPos[1]):
         return MAX_VAL
      #Otherwise we return a score expression balancing distance from the ghost.
      #distance from closest food, and total food count
      else:
         return minGD/2-minFD-foodCount
```

Comment:

Purpose: The Evaluation Function function is meant to give a potential Pacman action a score based on the environmental conditions and position of Pacman, ghosts, and food pellets resulting from that action (where a higher score means a more optimal action).

Design: We input the current game state and the action we plan to take from there and return a float score for that action (scaling of this score it relative to all other calculated scores). We start by obtaining the successor's game state from the action and the current game state and use the new game state to get pacman's potential new position, new food locations (in a 2D array format), new ghost states, and new scared ghost times. We then initialize a min and max value to return if we are in a super optimal or super bad state due to the action. This evaluation function focuses on the position of food, position of ghosts, and the number food pellets left. We make two arrays which contain the distances to all the food in one and all the ghosts in the other. We then only take the minimum values of each array as we only care about the closest food pellet and the closest ghost (we also get a food pellet count value with a simple count call on food positions). We then go to the final check of the function. The first check is to see if the ghost is too close to pacman due to the action and if it is, we return MIN_VAL for that action. The following check is to see if taking the given action gains us a food pellet and if it does, we return MAX_VAL as we want the all the pellets to win. The else statement is returning a score expression balancing distance from the ghost, distance from closest food, and total food count (more distance from the ghost is better, less distance from food is better, and less food count is better).

Revisions to evaluationFunction:

1. Made the score heavily dependent of the position of the ghosts as those are the only source the pacman can die/lose from
2. Made the score heavily favor pellets directly next to the ghost given that no ghost is too close (too close in this case is 2 units or less) because collecting all pellets is the goal for the agent
3. Given that no pellets are adjacent and no ghosts are close, the score bases itself on proceeding to the closest pellet that is far enough away from a ghost