Report of Pacman Capture the Flag

Team LCL

Zidu Chen zchen96@ucsc.edu

Zhiyue Li zli148@ucsc.edu

Jack Liu dliu34@ucsc.edu

Pac-Man was a Stand-alone game that everyone played when they were young. However, in this project, Pac-Man's grid world divides into two factions, blue and red, so that two groups can compete with each other. Each group is made up of two pac-man agents. The basic rule of this game is Red agents must defend the red food while trying to eat the blue food. When on the red side, a red agent is a ghost. When crossing into enemy territory, the agent becomes a pac-man. The same rule applies for the blue agents, it is just the other way around.

This is where the game became interesting, not only we have to consider on behavior of pac-man, but also on ghost. After several times of try out using baseline team, we were inspired to use one agent to do offense, and the other one to do defense. We had an aggressive approach first, thinking that it might bring us more points, which was to use both of them to do offense, but it turned out that defense is just as important as offense. The result was surprisingly bad. Since they had the same algorithm, they would choose the same route to get to where they want. As a result, The defensive team on the other side simply ate two of my agents while we had no agents to stop their offensive agent.

To win this competition, we have to either eat all but two of their food, or we end up with a higher score than their agents in 1200 steps. Since the steps are limited, and our model for offensive player is to try to eat as much food as possible while avoiding collide with their defensive player, we let our agent to eat the closest food first so that we do not waste our steps. Another important part in this game is the capsules in the game. It can make our agents stronger so that they are not afraid of ghosts. In other words, once our agent eat the capsules, we can not waste our steps on avoiding enemy ghosts. We had evaluation function in our program where it evaluates each step that agents can possibly take. In the end, the agents would choose the best step with the highest score. In our evaluation function, we made our agent want to eat the capsules the most, then food. If our agent eat capsules successfully, it would not avoid ghost, otherwise, when agent sees ghosts, it will run away and give up on eating food right away.

For defensive player, We wanted to stop their offensive player as soon as possible so their offensive player had no chance of eating our food. Even though eating opponent does not add score to the scoreboard, for defensive player, it wants to eat opponents the most so they can protect the food on our side. And it leads to a problem, where should the agent go when there is no enemy invade our side? For baseline team, the agent would take random steps. However, it costs problem that if it goes too far, then it takes more steps to catch the enemy and we may lose several of our food. Based on this observation, we decided to keep the agent running around the midpoint food. Since baseline agent want to eat the closest food first, and midpoint food is the closest one for them, once we defend around it, baseline team can not get the food at all. There is another good thing to let defensive player gets around the midpoint, if other people's agent tries to choose another path to avoid mid-path, our agent is close enough to either go up or down to catch their offensive player.

The algorithm we choose for offensive agent is minimax with the alpha-beta pruning because it works very well in competing situations since it was designed for zero-sum games. The minimax algorithm calculates the value of all the next actions for both teams and chooses the best action among them. After applying minimax algorithm, our agent can go toward the food while avoids the opponent's ghost. We modified the reflex defensive agent for our defensive agent. Our defensive agent will stay before the food that is closest to the border. Once an invader appears, the agent would simply change its target and take the invader down. It is not a complex solution, but it works excellent on stopping the enemy from getting points.

One of the greatest obstacles we encountered was the implementation of evaluation function which calculates the value for next action. To make the offensive agent eats the food while avoids opponent's agents, we need to get the information of current status and calculates a numeric value. However, there are so many parameters to be considered, such as the distance to the closest food and capsule, the agent's current mode, either pacman or ghost, and whether the opponent's agents are ghosts or pacmans. Since we've got these values, the problem is how to combine these values together and calculates an appropriate value which tells the agent which is the best action it should apply. The most difficult thing in this progress is to determine the weight of each value. For example, if a ghost is on the way to the food, whether the agent should take the risk of meeting ghost to eat the food? If the weight of the ghost is too high, the agent would only take the action which keeps the longest distance with the ghost. If the weight of the food is too high, the agent would go straight towards the food and ignore the ghost, under this situation, if the opponent places a ghost on the way to the closest food, our offensive agent would never eat the food. It was extremely difficult to find a good balance between them. We've tried multiple times and we finally come up with our final solution. However, the solution didn't meet our expectations because it didn't win as much as we expected. We think the problem is we only have one algorithm for all situations. If we write different algorithm for various situations, our agent's performance should be better.

Overall, it was a fun project. Due to our underestimation and the amount of wrong approaches that we took, there were a lot of things that were waiting for us to learn. When we were deciding the algorithm to use for the agents, we tried several different ones. None of them met our expectations. After few trials, we realized that each algorithm is merely one of the ways that humans depend on when choosing actions. Human beings have multiple algorithms and they switch between each other all the time seamlessly. An agent with only one algorithm implemented still cannot be referred to as an artificial intelligent agent. When we were designing the evaluation function, we were aiming to find the perfect evaluation function. After few hours of trying, we finally realized that there are no perfect evaluation function. You always have to sacrifice something to get something else. Thus, we had a debate of what should we sacrifice and came up with our final evaluation function. After everything that we have done, we refreshed our understanding of the AI, and the project. We learned a lot from it.