Project 4

Rafael Gonzalez

UFID: 74040466

Description of Methods.

1. eatPill: This is an integer return type method that targets all available power pills in the current map and makes the attacker approach the node of the closest one. When there are no power pills left, the attacker will perform the same behavior but with the regular pills instead.

2. nearestDefender: This is a Pair return method that returns a Defender and an integer. According to the passed Boolean parameter upon calling (if true), the method will return the closest non-vulnerable defender to the attacker and the distance as an integer and if false, return the defender that has the shortest pathway to the attacker and the distance as an integer.

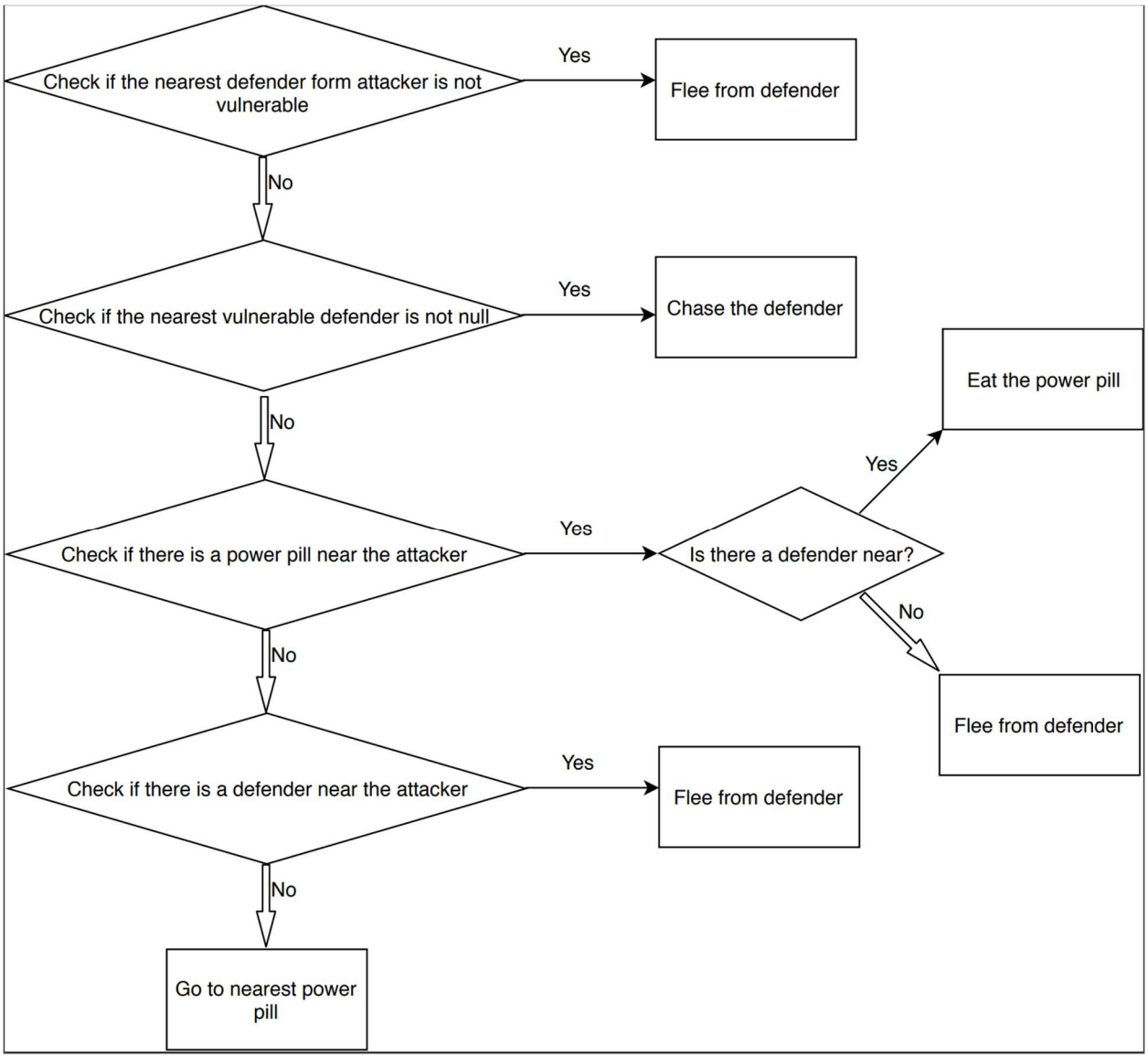
3. powerPillNear: This is a Boolean return type method that will check the neighbor nodes of the attacker for if they are power pills and of so, make the attacker stay in the position before of that of the power pill.

Diagram

Diamond shaped → conditional statement

Rectangle shaped → return statement

Note: Whenever a rectangle shape is reached, the program returns something, and then it goes back to the beginning of the diagram starting off from the last point



“What went right”: Developing a general idea of what were some of the most optimal ways of moving the attacker around according to the surrounding defenders’ locations, whether they were vulnerable or not, when to flee from defender and when to chase, to score the maximum amount of points as possible.

“What went wrong”: When it came to put those ideas into java code, many problems came across that made the initial strategy determined to be questioned. Specially determining that the path distance between the attacker and the closest defender is not the same as the path distance between the closest defender and the attacker, due to the defenders not being able to make 180 degree turns.

This project gave me great insight in the scenario when some ideas that were apparently clear can be easily deceiving, giving as a result having to rethink another approach to the problem. Think about all the possible cases that could happen whenever a step is taken, how to prepare and react to those cases and specially to optimize accordingly. Overall I enjoyed creating some sort of A.I for one of the most classic videogames, Pac-Man.