Project 4 Design and Post-Mortem Vince Velocci

Summary of attacker behavior:

Depending on the state of the game, the attacker will do one of five things: chase after a vulnerable defender, flee the closest non-vulnerable defender, eat a power pill, move towards a power pill, remain still in the immediate vicinity of a power pill, and move towards the closest pill (with the intention of eating it). One can view which behavior the attacker conducts as being dependent on seven conditions, in a hierarchy, which will now be described (see also the attached diagram on page 3 of this document).

The most important condition controlling the action of the attacker is whether a non-vulnerable defender is close to the attacker (1). If (1) is true, and there are either no power pills or the closest power pill is far away from the attacker (2), the attacker will simply flee the defender. If either (1) or (2) are not true, the program will then check to see if there are any vulnerable defenders (3). If (3) is true, the attacker will chase after (and eat) the closest one. If (3) is not true, the program will then check to see if there are any power pills available (4). If there are, the program will check to see if the attacker is close to a power pill and to a (non-vulnerable) defender (5). The defender will be non-vulnerable because (3) is checked before (4) or (5). If (5) is true, the attacker will eat the power pill and subsequently chase after vulnerable defenders. If (5) is false, the program will check to see if the attacker is close to a power pill and that defenders are far away from the attacker (6). If that is true, the attacker will remain still in the vicinity of the power pill, waiting to eat the power pill until a defender is about to make contact with the attacker. If (6) is false, the attacker will simply move towards the closest power pill. If (4) is false (no power pills are available), the program will check to see if the attacker is closer to a pill than to a defender, or if the defenders are far enough away (7). If true, the attacker will move towards the closest pill and eat it. If false, the defender will flee the defender that is close to the attacker.

Identification of successes and failures:

Overall, the attacker was successful at executing the strategy of moving extremely close to power pills and not eating them until a defender was also very close by. The attacker was also successful at chasing down vulnerable defenders while also evading non-vulnerable defenders. There were times when the attacker was so successful at evading the closest defender, that it would continue to be chased around in circles despite the presence of pills which could have been eaten while still evading the defenders. That can obviously be considered as a failure by the attacker, but it can also be considered a success because sometimes the attacker will be chased around in circles for so long, the defenders will stop pursuing the attacker and instead move in a different direction, thereby allowing the attacker to proceed to the nearest pills.

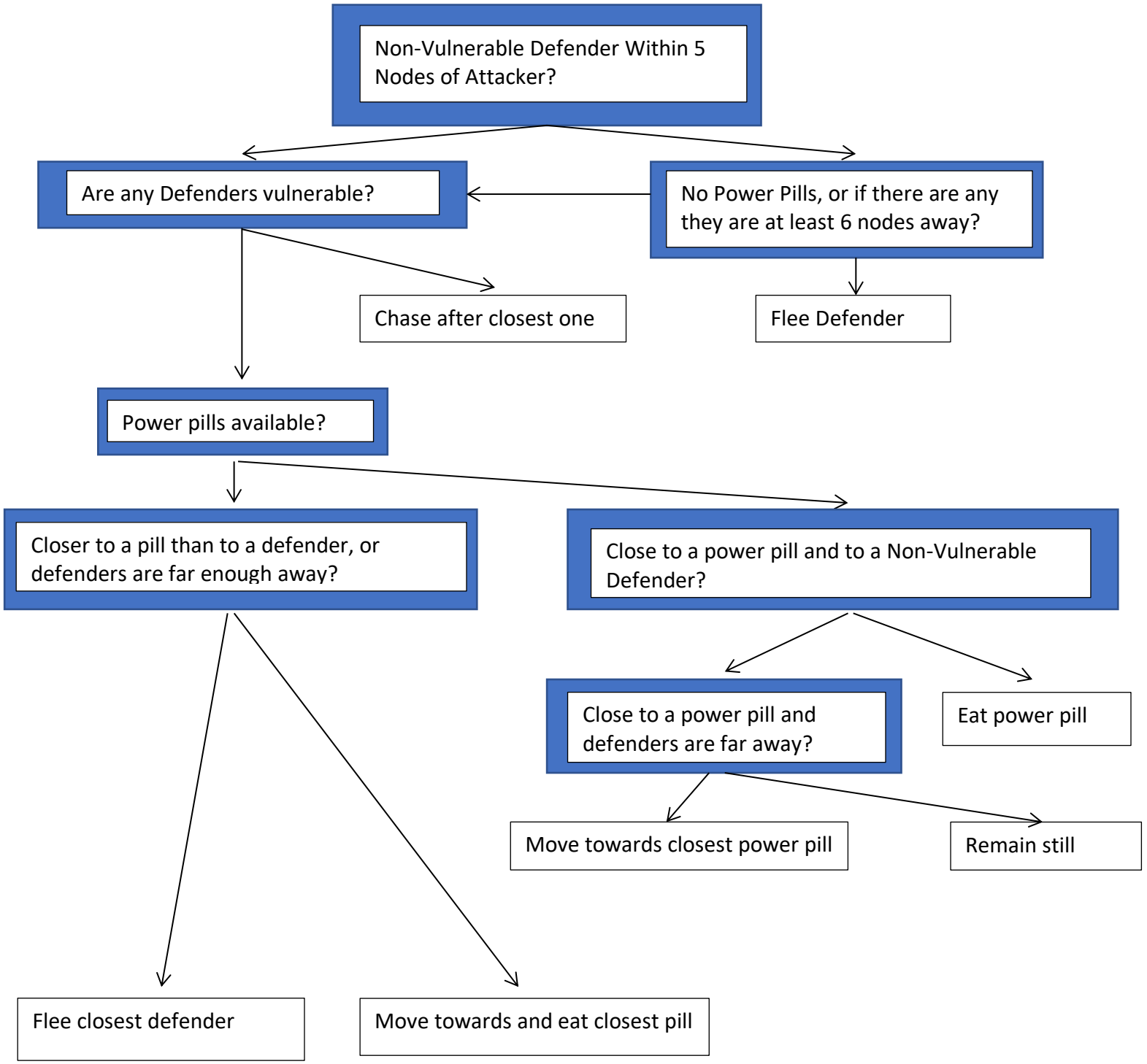
In addition to the issue with the attacker going around in circles, one thing I couldn't implement effectively was a way for the attacker to evade both the closest (non-vulnerable)

defender and the second-to-closest (non-vulnerable) defender. I tried including an if statement that checked for the presence of two nearby defenders and returned a direction that allowed the attacker to flee both defenders, but it resulted in a lower score, so I left it out. Upon reviewing the attacker's performance visually, it was clear that there were times when the attacker was being "double-teamed" at times. The attacker should have tried to evade both defenders well in advance but didn't.

Related to the problem of the attacker being double-teamed, the attacker would sometimes evade a defender in such a way that eventually the attacker would back itself into a corner, allowing a second defender to come in and double-team it. I'm sure with time, as with all things, I would have figured out a way to program the attacker to avoid those kinds of situations, but I decided not to devote much time to it.

Reflection:

This project was noticeably different from the previous three in that I wasn't tasked with writing multiple methods meant to produce a prescribed output. Rather, I was tasked with using an existing codebase with fully-programmed classes and interfaces to control an entity in order to maximize a score. Rather than playing the Pac-Man-inspired game myself, I would be programming the computer to play it. This resulted in a very open-ended project. In addition to being mindful of the code I wrote, I had to be creative and strategize how to best use the methods provided in the codebase, given the constraints. At first, I was very intimidated by this project. How could I possibly program the attacker to achieve a score of 6400 when I, a human, was struggling just to get to 1000? Aren't humans supposed to be better than computers at playing video games (unless you use a deep learning algorithm)? After forming a strategy, implementing it, and watching the attacker use the strategy, I then tried using the same strategy myself and managed to get a score of 4000! Although at first I was dreading this project, I eventually found it to be much fun. I enjoyed trying different things, regardless of how the score came out, and I enjoyed building upon a strategy that was already earning me the required number of points. I was especially entertained while watching my attacker's highest-scoring game!


Non-Vulnerable Defender Within 5 Nodes of Attacker?
Are any Defenders vulnerable?
No Power Pills, or if there are any they are at least 6 nodes away?
Chase after closest one
Flee Defender
Power pills available?
Closer to a pill than to a defender, or defenders are far enough away?
Close to a power pill and to a Non-Vulnerable Defender?
Close to a power pill and defenders are far away?
Eat power pill
Move towards closest power pill
Remain still
Flee closest defender
Move towards and eat closest pill