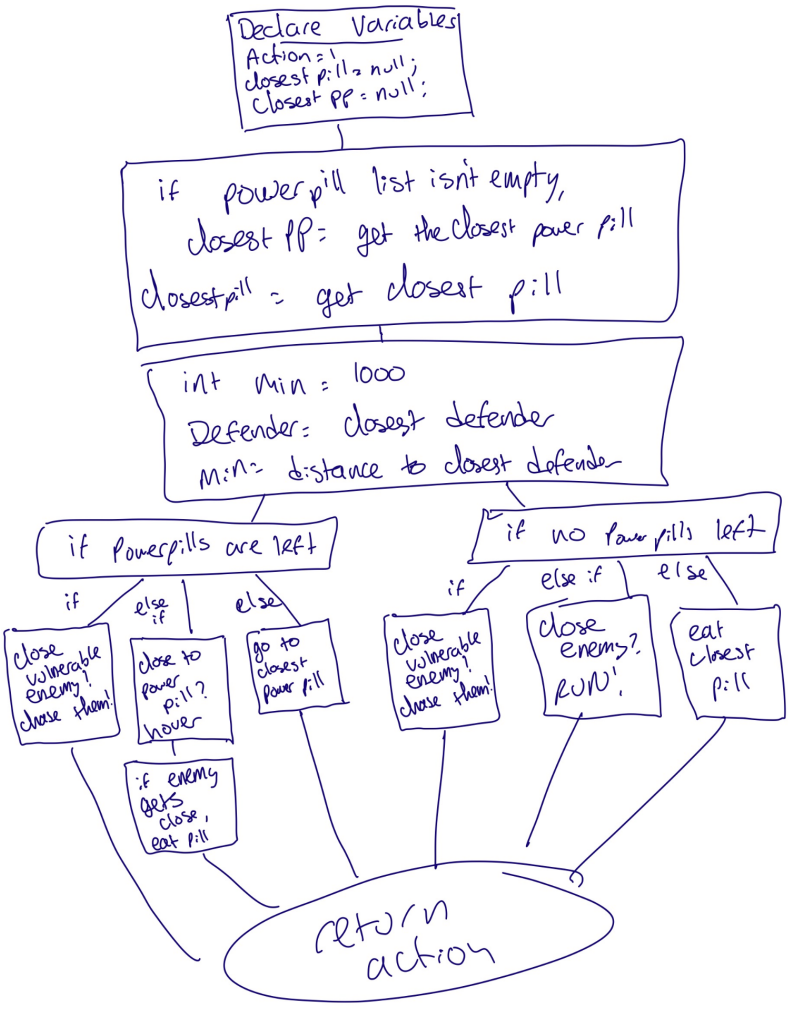
Design & Post-Mortem

Max Ross

Diagram(s) and description of attacker behavior and methods with description (100-300 words)



First, the variables are declared and defined- action, closestPill, and closestPP. Then we check if the PowerPill list isn't empty- if so, we set closestPP to the closest power pill available. We set closestpill to the closest pill. Next, we set min to 1000 to identify the minimum distance between the attacker and closest defender. It is set high because we have to shoot down, not up. We set the defender to the closest defender and min to the distance between them and us. Then, we have 2 if statements that decide where the code will take us- are there power pills left or not? If so, we are greeted with an if, if else, and else. The if checks if there is a close vulnerable enemy- if so, chase them. The if else sees that if there's no close vulnerable enemies and it is

next to a power pill, it will hover near it. If an enemy gets near, it will eat the pill and enact the if statement to chase them. Else, it will continue on to the next powerpill. On the other side, we have an if statement for no power pills left. Once again, an if, else if, and else. If there's a close vulnerable enemy, chase them, else if run from them because they're not vulnerable, else eat the closest pill because there are no enemies near. Then return the action.

Identifications of successes ("what went right") and failures ("what went wrong") (~300 words)

What went wrong: First off, the attacker kept getting stuck and I wasn't sure why. He would just stop for a moment and I had to analyze why he was stopped. The reasoning behind this was that while a defender wasn't on the board, the method I created to find the closest defender wasn't functional. This forced me to change my approach on looking for the closest defender and modify it to use a pre-made method that allowed me to search for any target, then cast it to a defender, which was previously just an actor. This made it usable because it was a defender object and now had access to the defender methods, and was the closest one to the attacker. Next, the priority of actions also went wrong. I had to thoroughly think out and write down what each if statement did to decide where it should go and if its placement should be changed. Another thing that went wrong was the closest power pill method- for a while I couldn't find the error until I discovered that I had to check if the list of power pills was empty. What went right? The order of the if statements after re-organizing them and the direction the code should go in. I decided to split the project into 2 parts- if there were power pills available and if not. Then, I organized the if statements under those parts into their respective position to decide what the best move for the attacker would be at evert moment. After countless reformations of this part of the project, the logic was sound and to its best extent received a higher score than required for 100%, therefore proving successful. The attacker also was successful in running from defenders when they got close as well as hovering near power pills to achieve the highest amount of ghosts eaten per pill.

Reflection on project (one per student; 100-300 words)

The project was overall very fun and eye-opening. It showed me that the logic behind the code was as important if not more important than the code itself. To come up with a structure for pac-man to get the most points involved analyzing the different possibilities for success, then putting them into action to make it a reality. The hardest part was definitely debugging the code. While it was difficult to come up with a plan of action, the bugs that left me hanging were tedious and required more effort than expected. This was one of the better projects in the class because it encapsulated many of the ideas we had previously learned as well as new logic.