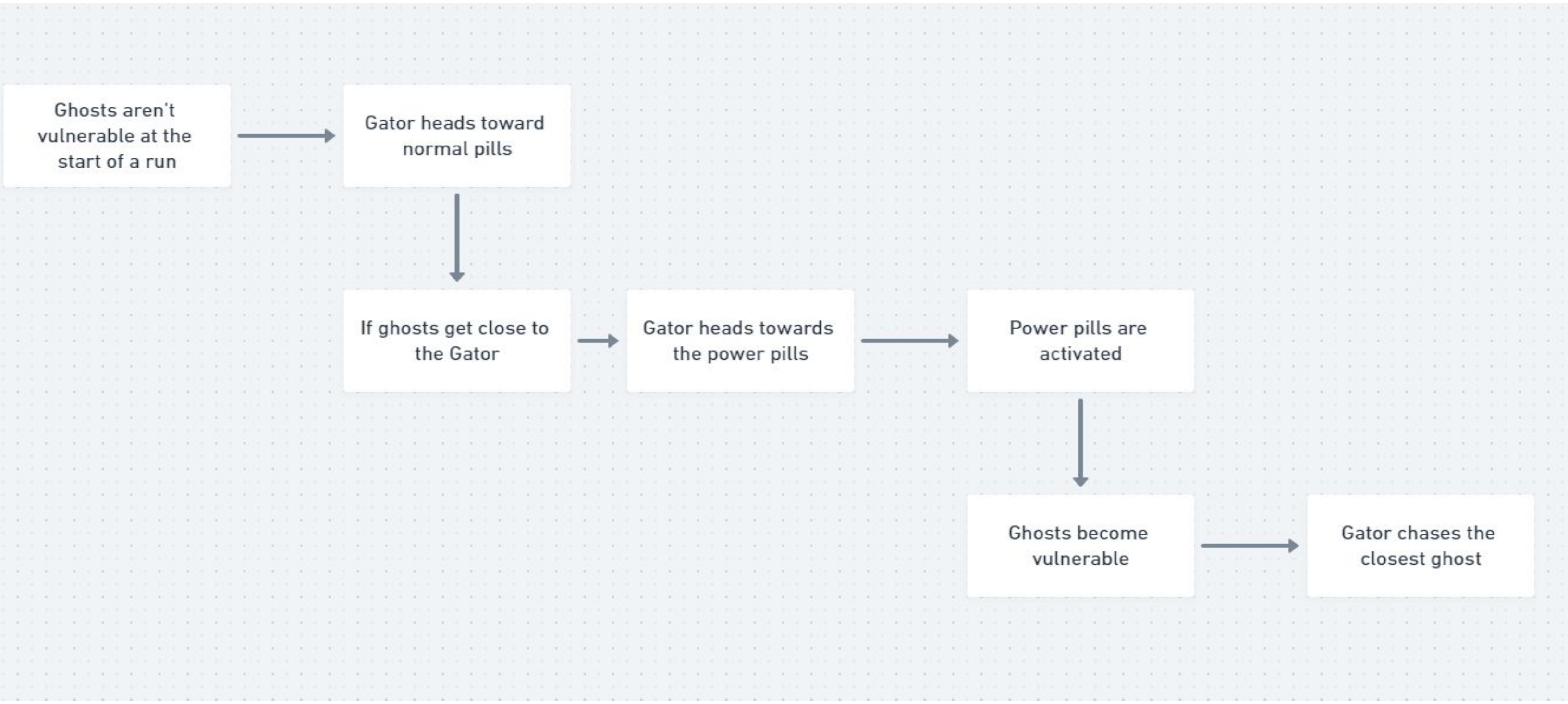
Ghosts aren't vulnerable at the start of a run
Gator heads toward normal pills
If ghosts get close to the Gator
Gator heads towards the power pills
Power pills are activated
Ghosts become vulnerable
Gator chases the closest ghost

At the beginning of the run, the ghosts are invulnerable and the gator is going to the closest normal pill. The gator eats the pills that are closest to it. However, when ghosts get close to the gator, the gator heads towards a power pill. Once the gator eats the power pill, the ghosts become vulnerable and the gator begins to chase them. Throughout the ghosts' vulnerable state, the gator is actively looking to eat them. This was done to maximize the amount of points that can be gained in a single run.

Although given a wide selection of methods, I only needed to implement a few of them into my final product. The most important of these methods consist of "getTargetActor"," getTargetNode", "getNextDir" and "getLocation". Each one of these methods played a pivotal role in the functionality of my code. "getTargetActor" was used to create an object ("closestGhost") of type Defender that mimics the closest ghost to the gator. "getTargetNode" was used to find the closest normal pill to the gator. This method was combined with a list of nodes that was created at the start of the program. "getNextDir" is single-handedly the most important method throughout this entire project. Every direction the gator moved was because of this method. I combined this method with the locations of the pills, powerpills, and ghosts to guide the gator into achieving the maximum amount of points possible. "getLocation" is a basic method that returns the location of a given object; it was used to determine the gator and ghosts' location. One additional method that was useful when designing my controller was "isVulnerable". I needed to check the condition of the ghosts and determine when the gator should chase the ghosts. By using "isVulnerable", I was able to program the gator to chase them once they became vulnerable.

Going into this project, I knew that planning was essential for my success. I started the project by planning out certain behaviors I wanted the gator to follow. My main goal was a success as the gator's primary goal is to eat normal pills until a ghost gets close. However, coming up with an avoidance algorithm proved to be difficult. My gator had the intelligence to chase after pills continuously, but didn't know what to do when ghosts trapped it in. My first train of thought was to have the gator reverse directions when the gator and ghost were about to collide. While this intuitively made sense in my head, the actual execution of the code proved to be different. The gator at times would oscillate in one position because it would change its target ghost every time it switched directions. I ended up scraping the idea and chose a more simplistic approach. Every time a ghost got close to the gator, it would just run away from it and try to eat a power pill. Since a computer is faster than a human, I was able to set the distance between ghost and gator to five nodes away from each other. Humans wouldn't have a quick enough reaction time, but a computer can maintain the distance with pinpoint accuracy. Even though my code surpassed the score requirement, there are definitely more ways I could improve my code. There are many instances where the gator will be trapped in a loop at the corner of a map and instead of chasing the rest of the pills, it would run around in circles until it was eventually caught by a ghost. I had succeeded in achieving the points, but there is much more work to be done if I want to create a highly intelligent controller.

This project was the most difficult project in terms of conceptuality. Starting the project was difficult and there was a significant learning curve to using the methods. I spent around 30 minutes or so planning my code and trying to understand the different methods that were available. While this project may have been one of the most challenging, I thoroughly enjoyed working on it because there were so many ways it could have been completed. It was satisfying to watch the gator move based off of my code. In my opinion, projects like these are the most beneficial because they require the programmer to incorporate all topics of programming and can learn from other programmers based on the logic of their solution.