Samuel B.

Mayank B.

Big Hero 6 Aftermath

As our last project for our into to Java class, we were asked to create a robot of our own within the Robocode program. With basic Java syntax already learned, we were able to come up with a robot in which my partner and I named “Big Hero 6”. With a name created for our robot, we then proceeded with the development stage; we had to start with the basic movement of our robot. Our plan for the robot’s movement around the battle field was to make the movement predictable; we did this in hopes of throwing off our enemies because we expected our enemies to assume that each robot would move randomly. We also wanted to implement a function where the robot would be able to detect where a wall is and turn itself around to avoid losing health due to crashing into the walls.

Moving forward to the attacking styles of Big Hero 6, we first wanted our robot to be constantly shooting regardless if they hit an enemy or not as our “hook”. We say hook because if a bullet lands on another robot, we then wanted to make this robot our “main enemy” among the other robots and focus on killing them off. We wanted this function to keep looping until all of our enemies have been killed off. Another function we wanted to implement into our robot was to have a “special shot” or a high-damaging bullet. Just like in a console game, once a character reaches a certain amount of energy stored, they’re able to perform their special move which deals a lot of damage to the opponent. Instead of energy, however, we wanted to see if there was something similar in terms of Robocode that we would be able to keep track of until our robot would be able to perform its special move.

Last but not least, how would our robot be able to survive if we don’t instruct it on how to defend itself? Starting off with the basics, we wanted our robot to be perpendicular to the enemy at all times; this allows our robot to maneuver around the enemy/ dodge any incoming bullets from the enemy. Another idea in mind was that after our robot shot, it would move up/down/left/right to essentially be “hidden” from the enemy. This “shoot then move” function was created to make it difficult for the other robot to detect who the bullets are coming from and to the opponent, it would appear as though bullets are just coming from every different angle. Another ability we wanted our robot to have was if it has been shot at least once, it would automatically move up/left/down/right, scan to find the shooter and make them the main target.

With all these visions just on paper, it was time for the implementing stage – the place where we test our abilities to see which of our ideas are unrealistic and which are doable. In the end, our idea for the robot’s movement backfired because when it came down to fight day, we would notice that our robot would periodically glitch out because not only was it scanning for walls, it was also scanning for enemies and any bullets headed its way which messed up the basic movement of the robot. On the other hand, however, we were able to execute the “shoot then move” ability which turned out to be a success. Once the bullet hit an enemy, the enemy would be alert and start searching around to look for the shooter but the shooter would be nowhere to be found. This process continued until the end of the round. Another function that seemed like a good idea on paper but ended up being harmful to us was making our robot contiguously shoot. In Robocode, when a robot shoots, health is taken away from the robot regardless if they hit an enemy or not, however, if the bullet lands on another robot, a percentage of the health taken away is returned back to the shooter. The problem with this rule is that our robot missed more than hitting another robot which caused our robot’s health to drop quicker. On top of this, our idea of determining the “main enemy” was not coded in time for the due date and our robot just ended up shooting at any robot it detected.

APIs Used:

* onScannedRobot()
* onHitByBullet()
* onHitWall()
* onWin()