James Domingo & Diane Margo

Robocode

CS 141

13 March 2018

Team Support

           Robocode was given as a project that the students work on throughout the quarter with another person. James Domingo and Diane Margo paired together and worked on the robot called Team Support. The name originated from another video game called League of Legends and it has nothing to do with supporting other robots. Although supporting other robots would be a cool idea to implement. Both individuals agreed that the robot should specialize in avoiding any conflicts with opposing robots and be able to dodge incoming bullets to increase survivability. In addition, the main purpose of Team Support was to survive as long as possible.

           At the beginning of our code, we included the package name of the individual that turned it in and included two imports. One was to be able to use the robocode classes, and the other was to be able to change the color of the robot. Line 12 is the class header for Team Support and it extends the Robot superclass. The Robot superclass contains several methods for individuals to use and is used as an introduction superclass for new Robocode users.

Furthermore, an integer variable was created for the upcoming for-loop and continues to the public void run() method. The run() method is where the robot’s colors, initial movement, gun movement, and body movement is located. Specifically, for Team Support, we set the color all as red. The for loop in line 21 to 23 is created so that whenever a new battle has started, it immediately travels backwards in hopes of colliding with a wall. If it does not collide with a wall during the for loop, it would be close enough to a wall and continue to the run() program where it constantly moves forwards and backwards. On line 25 to 29, we created a loop that would make the robot keep moving so that it would be able to turn and dodge at the same time.

The first method in the Team Support robocode is from lines 35 to 43. This method is called onHitRobot and its a method that is called when our robot collides with an opposing robot. This method is useful when it collides with another robot. It may not happen all the time, but it is a safety precaution for Team Support robot. The first thing that it does is go into an if statement. It returns the degree of the robot and the degree is something that is greater than -90 degrees and less than or equal to 90 degrees, the robot will back and turn left to attempt to dodge any incoming bullets. If not, then the robot will go ahead 100 and turn left 90 degrees. After this method is finished, it returns back to the run() statement and the while-loop will continue to run.

Lines 48 to 60 consists of our second method in our robot. The second method is named “onScannedRobot” and it is called whenever our robot sees another robot. This method is crucial because it dictates our robot’s actions and each line of code holds significant value when battling other robots. Also, e.getDistance is a call to the superclass Robot and it returns the distance of the scanned robot back to our Team Support robot. Each if statement gets the distance of the robot and fires an appropriate powered bullet. The purpose to make each bullet weaker the farther away the robot is to conserve health points if the bullet misses. So. it’s a lot easier to hit robots the closer they are so shooting a max powered bullet is an optimal choice, which is a max power of 3 for robocode.

Our third method is called onHitByBullet and it is from lines 62 to 67. This method is called when our robot is hit by a bullet. The idea with this method is to try and maneuver around the map in short distance in hopes to make it more difficult for other robots to target our robot. It would have been significantly better if we used other superclasses such as AdvancedRobot because it provides more complex methods and objects. However, with the Robot superclass, this was a simple yet functioning command for our robot that satisfies our condition to dodge incoming bullets.

Lastly, our Team Support robocode had the method called onHitWall. The onHitWall method consists of line 73 to line 78. This method will be called when our robot runs into a wall. The onHitWall method is used in order to avoid the robot to permanently run into a wall without coming up with an alternate route. Team Support’s code when the robot runs into a wall is to move ahead, turn right, and finally move back. The purpose with using 50 units for moving forward and backward is to avoid incoming bullets. In addition, this code helps our robot hover around the edges of the wall.

Overall, Team Support’s robocode was rather simple but accomplished basic dodging mechanics and efficiency in trying to survive the longest. Our robot would try to run away if it hits another robot before being able to counterattack. When Team Support attacks, we prioritized efficiency rather than raw firepower. When our robot gets hit by a bullet, it would maneuver around the map and dodge any further incoming bullet. The combination of our code had the general theme of avoiding unnecessary conflicts which helped us win a number of rounds. The robocode could have definitely been improved to have more advanced codes and enhanced dodging mechanics. Also, the use of other superclasses could have provided the methods needed to survive and beat the opposing robots. One of the codes in AdvancedRobot was calculating the velocity of other robots which would make it significantly easier for the bullets to land on the opponent.

One of the methods we referred to from an API source is the onHitRobot method. We thought that it was a necessity for our robot to have a specific action when it collided with other robots. In addition, we used the getBearing() call inside the onHitRobot method. The onHitRobot and getBearing() code of our robot assists in Team Support’s movements when it comes to battling other robots. Bearing helps identify the location and positioning of the robot and acts accordingly. Our robot would go forward or backward depending on the angle where the robot was hit and then turn to the left.

In order to test our code, you would first need to download Robocode. For Windows operating system, you would need to install a JDK or JRE file. Afterwards, you would need to establish a path variable and it must be included in the directory of the Java executable. After completing and establishing all the path variables and correct files, you should be able to run Robocode. For Mac, you need to download Robocode. Make sure that the Robocode file downloading is under the folder “User/currentuser/directory”. If you are unable to find the path, go to Finder and then press Shift + command + C. You should be able to find the robocode file, double click robocode.command and it will launch Terminal.

Once Robocode loads, click on Robot, Source Editor, and a new window should open. On the new window, click on File, New, Robot, and name your first Robot! You’ll get a new window which you can customize your new robot. For Team Support, copy and paste our code into your editor and then change the package name to what you named your robot! You can test our robot by going to back to the first robocode window, click on Battle, New, and then find your Robocode file with Team Support’s code. Lastly, you add the robocode file and sample robocodes and finally Start Battle. You should see a red robot battling it out with other robots!

There are multiple sources and websites that you can refer to in order to help you build your own robot. One of the robots that stood out during the battle was a robot that would stick to the walls of the map and continually fire at other robots. The pros of this robot was that it did not have to look behind itself to check for other robots. However, the robot is very vulnerable if its against a robot who could freely move around the map and dodge incoming bullets. Lastly, Robocode is very flexible and it adjusts according to each coder who creates their robots. There are also sample robots that could be used as reference robots that could help develop ideas and robot mechanics.