Noah Renella

Youngjun Woo

Robocode Report: Team Groovy Crew

In our Computer Science 141 class we were given the task to create a robot using robocode to test the limits of our capabilities with coding. Together Noah Renella and Youngjun Woo worked as the groovy crew to create a robot to compete with the best. There were many challenges and speculation to what could happen and what we could create with the robocode engine but we tried different things out to see what we could get. In this report we will go over the inspiration of our robocode, the features we put into our robot, next we will go over what we put into our code and finally, what someone might need to do to test our robocode for themselves.

To start off we did not really know what to go for in creating a robocode robot. We started off by coming up with ideas of what robots could do the best at surviving so we thought of a dodging robot but it would need to learn how to fight back. We then ran a bunch of test runs using the sample robots that robocode came with. To our surprise some of the robots were pretty good. Robocode robots in particular that stood out were the “Walls” robot which was mainly what we were going for, a robot that rides along the edge and shoots to the center, the “Crazy” robocode robot that would have random movements around the map while shooting anything that comes within it scan, and finally one of the most interesting robot for us, the “Corners” bot that would start the game off by going to a corner and shooting everything that passes through its scanner. Each of these robots had their own boons and banes that we tried to put together to balance everything in each robot.

We came up with “CornerCamper” that took all the good parts of each robot and put it into one. We saw that the “Crazy” robot had gotten a lot of points in the overall damage in the game while robots like “Walls” and “Corners” had a lot more points in survivability. We took that massive movement from “Crazy” and “Walls” while keeping the survivability from “Corners” to create “CornerCamper” which would ride the walls to stop at a corner and fire a barrage of bullets to all the robots in the center of the arena for a brief period before it started moving again to the next corner to repeat its cycle.

To start off with our code for our robot “CornerCamper” let us start with the imports. We had many imports but the more important ones we had were the “robocode.ScannedRobotEvent”, “java.awt.Color”, and the “robocode.Robot” import. Our first import “robocode.ScannedRobotEvent” was very important because this would be the basis for all our firing methods, if a robot was scanned then our robot should fire at it or if a robot was scanned in front of us we should move out of its way after crashing into it. The second import “java.awt.Color” is may not be as important as the other ones but when creating a robot it is important to put as much of yourself into the robot as you can so we decided to make our robot a combination of white and black colors so that it could look like a chess board on our robot and finally our last import was “robocode.Robot”. This import is one of our more important ones because it allows us to inherit more codes into our robot like a very important code in our bot getBattleFieldWidth(), and getBattleFieldHeight().

Moving on to the code that makes our robot fight the way it does, we will start off with what is in the public void run method. Our first strand of code is a value called “wallDistance” and we set this value to the greater of two numbers using Math.max from the battle field width and the battle field height. This allows us to always know where our robot is in respect to the walls which it rides along the whole time. Let us say our robot was about to turn right while its on the bottom right corner, it would need to know how far to go to get to the other corner so it will check the width and height and take the larger of the two in order to get to the next corner without missing out on any turns. The next piece of code in the run method is an integer for how much the gun moves each time it takes a turn while at a corner. We experimented with this number many times trying to get a perfect one for what we wanted our robot to do, we found that if its increment was to low it turn way to slow giving it the problem that the “Corners” sample robot had of being to immobile while if we made the number to large it turn to fast making it fire way more shoots than were necessary so we went with a nice rounded number of ten to make it turn just fast enough to get to a corner, unload enough bullets to get damage down and then move to the next corner. Another very important part to our code in the run method is the startup movements that our robot takes only once in the game to set itself up. This would be the turnLeft(getHeading() % 90); piece of code followed by the ahead command which sets up our robot so that it goes straight to a wall regardless of where it may have been looking before. While it is there it sets its gun right so that the rest of the code can run without problems.

After all of this start of the game code we move on to the code that actually makes the robot do what it is supposed to do for the rest of the game so within the infinite while loop we set up a for loop that repeats five times that has nested for loops within it. Each of these for loops repeat nine times so that our gun when at a corner functions like a sentry turret. It goes back and forth shooting whatever is scanned with its radar after the nested for loops are completed it moves to the next part of the code which make it moves forward using the ahead command with a distance equal to the greater number of the wallDistance value. Once it reaches the next corner it turns right and repeats the process of turning the gun left and right again. This is how we came up with the name corner camper, in video games you would refer to someone who stays in one place for long periods of time as a camper and because our robots would stay at each corner for a good amount of time in order to fire shoots into any robot in the center we went with the name “CornerCamper”.

The next part of our code is the events code. These codes function almost like exceptions in which they take over what is going on with our code if any of the events are met. The first one we have is the onHitRobot event in which while we are riding along the walls if we happen to come across any other robots and crash into them our robot will move forward or backwards depend on which side was hit. We thought this would be an import part to our code because if we were going against a robot specialized for ramming other robots we would not want to keep moving into them giving them free points in damage and even worse, killing our robot. After moving back or forward it would continue moving along the walls to do its thing. The next event code we have is the onScannedRobot method which is the most important part to any robot in the game. Simply put if our robot would scan another robot we want it to shoot at them so we did just that but we would check to see if the robot we scanned was close enough to us in order to fire a more powerful shot. We initialized our range as, if the robot is one hundred pixels in range it fires at full power but if they are any other distance, it fires at its base one power.

If you wanted to try out our robot for yourself there are many things you could change to see how it affects the overall robot. First off be careful with changing the gunIncrement value because we set it up so that after the nine loops in the nested for loops the gun would have turned a full 90 degrees left and then do another 90 degrees right, so if any changes were made to the gunIncrement value you would also have to change the number of loops that occur in the nested for loops so that the gun does not stay off centered. Another thing to test out could be the directions in which our robot turns, we initialized it so that it turns right every time but you could replace the turnRights with turnLeft and the turnLeft into turnRights to change the direction that the robot goes. You could also add more distances that the robot takes into account when firing a shot but we found that our robot fires to many shots to begin with so adding more power to different distances could disable our robot a lot faster resulting in a shorter overall life.

In conclusion this robocode project was very interesting to say the least. It began as a nerve wracking thing in creating a robot that could stand to the best but after a couple of test runs with whatever we could come up with we slowly started realizing just how fun it could be to make something come to life with our skills in coding. Sure there are many more improvements we could have made after seeing how our robot performed in the actual tournaments but it was incredible to see how ours compared to the other great robots that the class made.