**Group:**

**Umair Rashid – 100312938**

**Ryan Mock – 100272664**

**Annie Phan – 100283449**

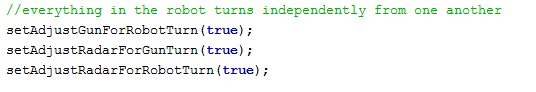
**CLASS: INFO 2313 S10**

**August 6th 2017**

**FINAL REPORT**

Our robot is called ARU Robot which is the first letter of all our names. We started our robot with just basic movement and worked our way towards what we have now. We wanted to create a robot that avoids walls when it starts and tracks the robot’s enemy movements once it finds the robot. The very first thing we did is we made sure that our gun, radar and robot moves independently from one another. We also made a Boolean value to tell us when our robot is moving forward so if that value is true then its moving forward and if its false then its moving back. We change it in the code every time we have the ahead or back statements.



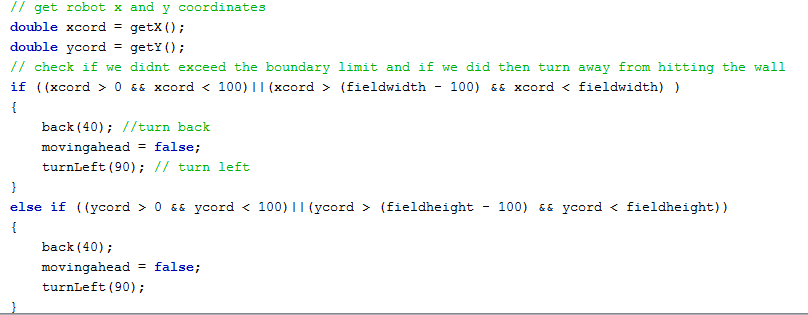


After doing that we wanted our robot to scan the whole 360 degrees when it starts and then if it doesn’t find anyone then move ahead, and then scans again. So basically, keep repeating the same procedure repeatedly until it finds the robot.

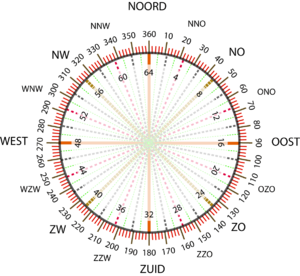


Once we got the robot in the battlefield we realized that the robot sometimes starts too close to the wall and loose unnecessary points so we decided that we would create an imaginary boundary around the robot so it doesn’t cross it and hit the wall when it just starts. We grabbed the battlefield width and height and then whenever the robot is working the program grabs robot’s x and y coordinates and stores in a variable. Once we have this data we just make sure that if x coordinate is less than 100 or there is only 100 pixels remaining between robot and battlefield width, we signal the robot to turn 90 degrees and go back 40 pixels. We did the same thing with the Y coordinates except we used the battlefield height this time.

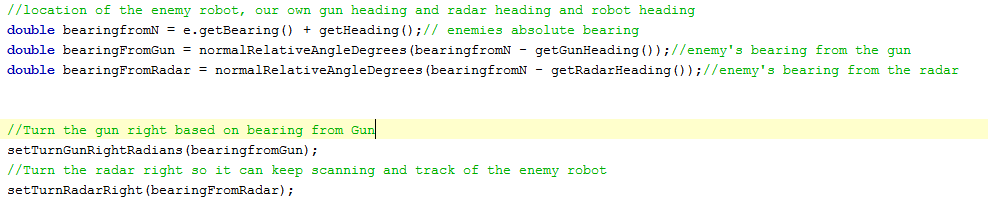




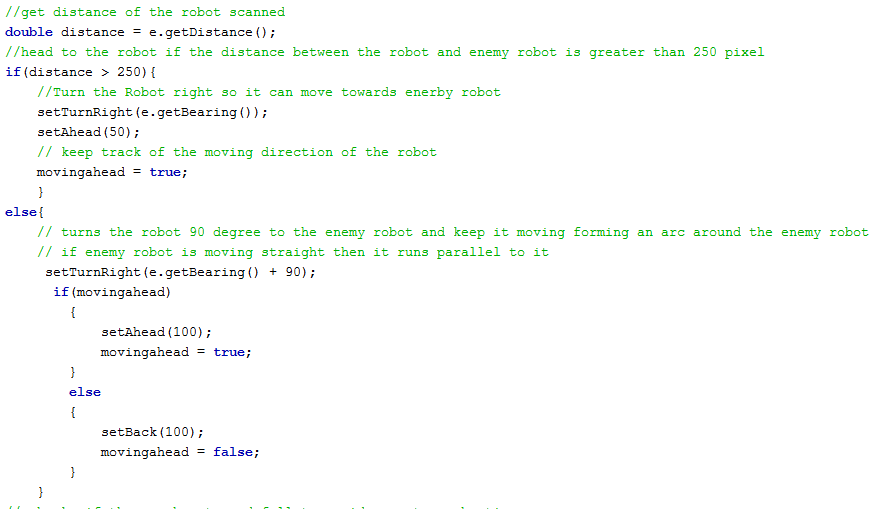
Our next focus was to track the enemy robot consistently so we had to come up with the formula that helps us do that. We had to play with the heading and the bearing to understand how can we make this work. First, we decided to find the absolute bearing or we call it bearing from the north. This calculates the angle of the enemy robot to our robot as if it’s a compass. Like the one in this image:

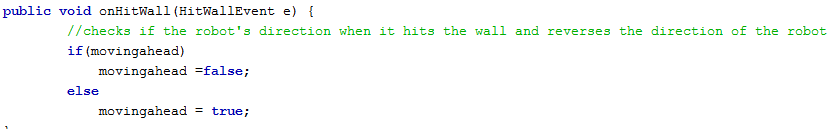


So, if our robot is heading 30 degrees and the enemy robot’s bearing to us is 40 degrees then the absolute angle of the enemy robot is 70 degrees. Then since our radar and gun turns independently we created bearing from Gun and Radar. We basically subtract either gun heading or radar heading from absolute bearing to find out how much they need to turn so for example if the absolute heading is 70 but our gun’s heading 270 that means the answer would be 70 – 270 = -200 and since its -200 and we must get the positive value we use the function normalRelativeAngleDegrees to convert this difference into a positive one. Basically, if the answer is negative this function will add the number to 360 to get the positive angle between 0 and 360. Once we have that we can use this to turn our radar or gun right that amount.

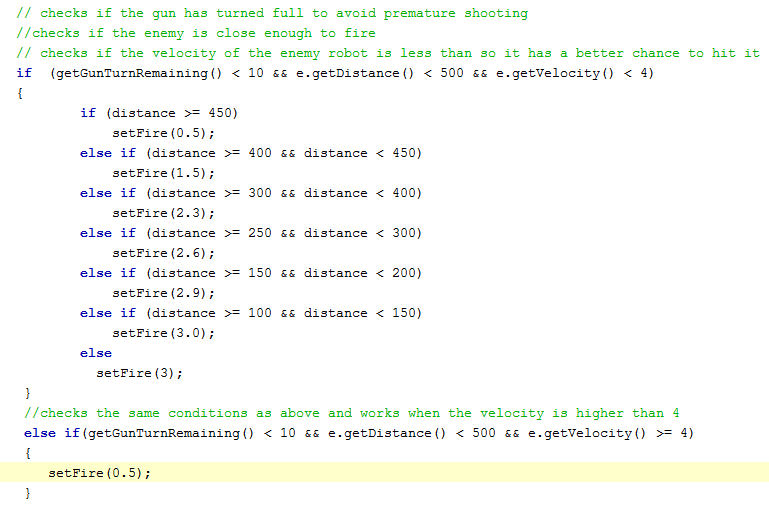


Next step was to make sure that if our robot is too far from the enemy’s robot we want the robot to head towards it. Once it reaches the robot close enough we want our robot to turn 90 degrees to the bearing of enemy robot and create a movement of arc around it. It pretends like the center of an arc is the enemy’s robot and it moves around it once its close enough. One problem we faced we did that is that once the robot hits the wall it doesn’t deflect since its always scanning the robot. So, to overcome it we used the event of hitting the wall to turn our robot based on its original direction. If the robot was moving forward and the Boolean moving forward is true and it hits the wall then this event makes the Boolean moving forward false and then our robot goes the other way.





Further we worked on improving our shooting success. We first realized that when the enemy’s robot is moving fast and our robots fire we tend to miss that pretty much 90% of the time and this causes us to lose energy. So, we created a series of criteria our robot needs to check before it fires. First condition was that robot checks if the gun has turned enough before it fires so it avoid premature shooting, then we checked the distance to make sure the robot is close enough to fire. Lastly, we checked the velocity and if it’s less than 4 then fire based on distance of the enemy’s robot. If the velocity is more than equal to 4 then we fire less powerful but faster bullets so it has a better chance of hitting the target.



Another thing we added is something called later velocity. We got this formula to helps us find our the later velocity of the enemy’s robot and helps us turn the robots gun and fire at the predicted location of the enemy’s robot. We also had to do some conversions since some of the functions are for degrees and the others are for radians.

