

## **CS 201 Lab**

### **Event Handling in Robocode**

The previous Robocode lab exercise consisted of moving the robot in a square. The robot was not provoked by the existence of another robot, it just did its square move several times. In a battle, your robot will most likely move, because of some situation or *event*. If your radar detects another robot or if you are being fired upon, you will want your robot to do something.

You will need to install the Robocode files on your lab computer. Follow the instructions in Lab 9 under "Installing Robocode" and "Running Robocode".

## Experiment 1

**Step 1.** Download J07E03.java and copy it to the c:\temp\robocode\robots\sample directory. Open the source code for the robot J07E03 by selecting Robot > Editor. In the robot editor, select File > Open. Select and open J07E03.java. Take some time to understand the code.

In the `run` method, the statement

```
setAdjustRadarForRobotTurn(true);
```

allows the radar to be turned independent of the direction of the tank. Likewise, the statement

```
setAdjustGunForRobotTurn(true);
```

allows the gun to be turned independent of the direction of the tank.

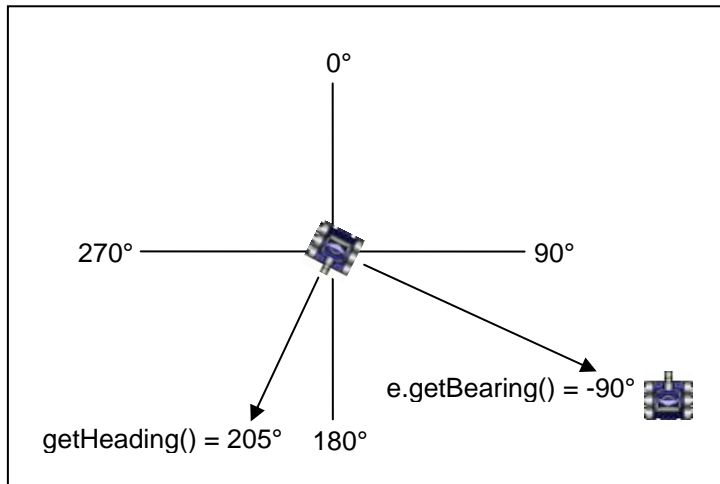
Special methods are called when an *event* occurs during a robot's turn. When the radar detects another robot, the `onScannedRobot` method will be called. The statements in this method will be executed every time a robot is detected by the radar. The robot is reacting to an *event*, which in this case is the event of finding another robot.

One parameter is passed in the `onScannedRobot` method. This parameter is an object, `e`, that contains information about the robot that has been detected. By calling certain methods on this object, information such as the robot's bearing, distance, and name can be retrieved.

The `onScannedRobot` method starts with an `if` statement. The condition of the `if` statement is that the name of the detected robot is "sample.SittingDuck." The name of the robot is retrieved by calling the `getName` method on the object passed by the event, namely `e`.

The `absoluteBearing` is determined by adding the current heading of the tank (the result of calling the `getHeading` method) and the bearing of the detected robot (the result of calling the `getBearing` method on the object that is passed by the event, `e`). The heading of the tank is based on north being  $0^\circ$ . In the figure below, the heading is  $205^\circ$  from the north. The bearing of the detected robot is based on the heading of the main robot. In the figure below,

the bearing is  $-90^\circ$ , means that the detected robot is 90 degrees to the left of the current heading of the robot that is doing the scanning.



The `bearingFromGun` determines the bearing where the gun should be pointed to so it is pointing at the detected robot. The `normalRelativeAngle` method is called to produce a bearing between  $-180^\circ$  and  $180^\circ$ .

The `turnGunRight` method is called to turn the gun toward the detected robot, and the `fire` method fires the gun. The value `1` that is passed in the `fire` method determines the energy of the bullet that is fired, where `3` fires the highest powered bullet.

**Step 2.** Compile and run the robot J07E03 in a battle with the Sitting Duck robot.

**Step 3.** Add a statement to turn the body of the tank toward the detected robot. Before the `fire` method, call the `turnRight` method and pass as the parameter the value that is returned by the `normalRelativeAngle` method, which in turn is passed the value that is returned by the `e.getBearing` method. Compile and run the robot J07E03 in a battle with the Sitting Duck robot.

**Step 4.** After the statement in step 3, add a call to the `ahead` method and pass the value `100` to this method. This should move the robot closer and closer to the detected robot. However, but we don't want the two robots to collide. To prevent this, place the `ahead` method inside an `if` statement where the condition is that the distance of the detected robot has to be more than `100`. The distance of the detected robot can be retrieved by calling the `getDistance` method on the object that is passed by the event, `e`. Compile and run the robot J07E03 in a battle with the Sitting Duck robot.

## Experiment 2

Another event that warrants some action is when the robot is hit by a bullet. The robot above (J07E03) contains an empty `onHitByBullet` method. This is the method that is called if the robot is hit by a bullet from another robot.

**Step 1.** When the robot is hit by a bullet, turn the robot to the right at a 90° angle against the bearing of the bullet. To do this, call the `turnRight` method and pass it the difference between 90 and the value that is returned by calling the `getBearing` method on the object that is passed by the event, `e`. Then add an `ahead` method and pass the value 100. Compile and run the robot, but this time include another robot in addition to the Sitting Duck robot.

**Step 2.** The current code only allows the robot to attack the Sitting Duck robot. Remove the `if` statement in the `onScannedRobot` method that restricts the attack on the Sitting Duck robot to allow attacks on any robot. Compile and run the robot.

**Step 3.** Remove the `ahead` method call in the `onScannedRobot` method. Compile and run the robot and compare the robot's movement with the movement of the robot in step 2.