FINAL PROJECT -- MILESTONE 5 FINAL

=====================

**Introduction:**

- Name of Assignment: Final Project Milestone 5A - Simple Map Method

- Lab team number: 5

- Team Members:

+ Phuoc Nguyen, Khoa Tran (NXT Part)

+ Corey Short, Trevor Davenport (PC Part)

- Approximate number of person hours spent on this: 5 hours

- Code: steam/TEAM/FinalProject/Milestone5/

**Performance Specification**:

Our robot meets all the specification for this milestone. Our Rover is able to Map out the surrounding area by using Locator's calculations of a new pose and combining it with a dead reckoning pose. This is accomplished by making two complete circuits (rectangles) and fixing the pose when the robot reaches a corner.

The standard deviation is big for X because the location (0, 0) and (241, 0) cannot be reached. We have to estimate our first starting point and calculate from that point. In order to prevent the error from building up, we try to fix position along the way so make the calculation more accurate.

**Objective:** The final milestone for this project requires us to locate three bombs on the map and print a screenshot showing that we have found them.

**Mapping Idea:**

In order to map the wall, we will let the robot know the location to go to, then tell the robot to either map left (the scanner will rotate to the left) or right (scanner rotates right). While the robot is moving, we will get back the x, y coordinate from the robot and send it to the PC. When PC receives it, it will map the location of the wall on the GUI. We also have 2 more methods to support our mapping which is Ping (to get the location at the current heading) and Explore (which is the get all the data within a range)

The bomb will be mapped on the GUI as a curve, that’s how we know which one is the bomb and which one is the wall.

**New Classes and Roles:**

\*\*\* For this Milestone 5-B, we implement 2 more values for the enum Message which are *WALL* and *SEND\_MAP*.

* In the PC side: sendMapLeft(), sendMapRight(), drawObstacle()
* When SEND\_MAP is executed, it will send out the x and y coordinate to the NXT brick and asks the NXT to *GO\_TO*. Also, if Map Left is called, the PC also send out 90 degree which is the angle that the NXT should rotate to in order to carry out the action. Otherwise, if Map Right is called, -90 degree will be sent instead. We have 2 functions sendMapLeft() and sendMapRight() to handle this task.
* While the NXT processes the message, it sends back SET\_POSE and WALL to the PC. Those 2 messages are sent continuously when the NXT is running, that's how the PC knows where and when to update its location and the location of the wall. In the PC part, we use DrawWall() method to draw the location of the wall and update the location of the robot by drawing pink dot.
* In the NXT side: sendData(), sendWall()

- When the SEND\_MAP command is received, the scanner is first being rotated to the desired angle 90 or -90. We change the method sendData() so it will take in 2 boolean values to determine whether the robot should send back the wall location or not. If the command is SET\_POSE then the wall location will not be sent. Otherwise, if SEND\_MAP is called then the sendData() will get the echoDistance from the scanner and send back the location of the wall to the PC by using the new function sendWall(). Inside the sendWall() method, based on the current Pose from the navigator and the angle from the current getTachoCount(), we set the current Pose by using PointAt(obstacleDistance , angle + pose.getHeading()), then we send back the x and y from the current pose that we just update to the PC side.

\*\*\* For Milestone 5-C we implement more methods to make the mapping more accurate with less effort, we add 2 more enum values which are *ECHO* and *EXPLORE*.

* In the PC side: sendEcho(), sendPing()
* sendEcho() will send an angle relative to the robot to the NXT brick.
* sendPing() will just ask the NXT brick to explore the map, and send nothing.
* In the NXT side: sendEcho(), sendPing()
* When the *ECHO* message is received, the robot will use the received heading angle to get the location to the wall upon that current heading. The robot will send back the x, y coordinate of the wall by sendWall() method. With that, the PC will receive the message to update the wall location.
* When the *EXPLORE* message is received, the scanner will sweep a specific range (in our code it’s from -90 to 90) and send back the *MAP* message the location of the wall within the sweeping range. With that method, we are able to get more data with only one task. By using this method, we are able to detect the corner and the room. However, it’s not accurate because of the concavity of the rotation of the Head and the location of the wall.

**Detected Map:**

