IEOR 140: Project 2 Milestone 3

Team 6: MoonSoo Choi & Sherman Siu

**Sherman**: coder for GridNavigator, debugger, report contributor

**MoonSoo**: coder for ButtonCounter, debugger, report writer

**Time spent:** approximately 8 hours each

**Project Description/Robot Specification:**

We programmed our robot to read the destination grid coordinate (through buttons), and reach the desired destination. While approaching the destination, a robot may have to establish its traveling directions and appropriate amount of travelling grids. After approaching the destination, robot can repeat this activity as many times as the user wishes to.

Robot user can use buttons of a robot to enter coordinates of a destination; in our project, ButtonCounter class handles this activity. Through this class, the robot will be able to display X and Y values of destination, increase/decrease those values, and save the data for later uses.

**Hardware Design**

In terms of hardware, our team did not make significant changes to the robot throughout project 2. In milestone 2 of project 2, we recognized the significance of locating the light sensors above the ground, as doing so would yield higher quality of performance of light sensors, so we kept this design in this milestone 3 as well.

**Most interesting/challenging/difficult part of the project:**

Probably the most challenging part for milestone 3 was learning and understanding lot more new concepts, including ButtonCounter class and task analysis and class design. For ButtonCounter class, after spending good number of hours, we finally learned how to write one. Nonetheless, task analysis and class design were responsibilities that we had to force ourselves to write out. Both of us did not feel urging need to design a task analysis, but as the length of our projects got longer in this class, we thought it would be necessary to organize a summary of what our programming codes do.

One another interesting mistake we made in our class is the following:

{

oldPoint = currentPoint;

currentPoint = newPoint;

}

From this code, you can probably tell that once this code is executed, the oldPoint and currentPoint variable will all *“draw its reference arrows to”*newPoint variable, which means whatever modifications were made to newPoint variable, oldPoint and currentPoint will **always store the same value as the newPoint**.

**Problem/Task Analysis**

**Software Design**

For Project 2 Milestone 3, we have three classes as following:

ButtonCounter.java

Tracker.java

Milestone3.java

Out of three classes, Tracker class was not modified (i.e. we used the same one from milestone 2). I think ButtonCounter class was quite straightforward, since its only responsibility is to read user input according to what buttons users press, and increase/decrease X-coordinate and Y-coordinate as the user wishes to. Below (Table 1.1 is the list of buttons used in ButtonCounter class)

However, milestone3 was quite a tough cookie. We had to draw out a diagram of different scenario that robot is going to encounter (e.g. which direction should robot travel? How much should robot rotate?), so we ***divided the task into smaller parts***. To be more specific, we first thought about to different directions robot needs to rotate in different scenarios. In designing the robot to travel appropriate amount of distances until it reaches the destination, both of our team members knew that we had to implement some kind of a loop, so wrote out several loops for the robot to travel (i.e. those are part of moveXdistance() and moveYdistance() method of milestone3 class). Finally, we decided to write out a storePoints() class to keep a track of old, current, and new points.

In terms of relations to tasks and subtasks, it is quite obvious which task/subtask is delegated to each class. For example, the ButtonCounter class will ask the user to press the buttons to input their data. Milestone3 class, through an instance variable of a ButtonCounter class, will take the input, and use appropriate (e.g. travel and rotate) methods to approach its destination. Tracker is a baseline class that supports fundamental functions of a robot.

**Table 1.1**: List of Buttons

|  |  |
| --- | --- |
| **Button Number** | Task |
| 2 (Left Button) | X = X + 1; |
| 3 (Left and ENTER) | X = X – 1; |
| 4 (Right Button) | Y = Y + 1; |
| 5 (Right and ENTER) | Y = Y – 1; |
| 8 (Bottom grey button) | EXIT |

**Programs:**

ButtonCounter.java (this is located in Project Two Milestone 3/src/buttoncountpc)

Milestone3.java (located in Project Two Milestone 3/src/com/mydomain)

Tracker.java (this has actually not changed from Milestone 2; also in com/mydomain)

JavaDocs are located in the project folder. They should be right there.