*Purpose:*

The lab creates and compares two distinct methods of makeMove() that, in a virtual scenario, directs a Roomba to move and clean dirty tiles in a room containing randomly generated objects. The experiment focuses on the presence and involvement of the infrared sensor.

*Procedure:*

Two unique makeMove() methods will clean the room with different strategies. The efficiency of the Roombas will be tested via 60 executions of code with four room configurations (15 trials per) with varying size and the amounts of objects generated. For each room configuration, the average of the percentage of the room cleaned for the Roomba will be recorded and analyzed.

*Programs & Explanation* (Class headings and brackets *not* included for brevity):

Method 1: Diagonal Movement, No Infrared Sensors

Method 1 focuses purely on diagonal movement and turning. Whenever the Roomba’s frontBumper or wallSensor hits an object, the Roomba will turn in a randomly-selected diagonal direction. The strategy relies on the Roomba moving in multiple diagonal directions and the diagonal paths covering a large amount of distance. Weaknesses include the Roomba moving back and forth in the same directions due to ill probability. Note, the method this.WallSensor was removed from the program to simplify and focus the procedure and code on the infrared sensor.

**if**(**this**.turning && **this**.getDirection() == newDir)

**this**.turning = **false**;

**if**(**this**.frontBumper) {

turning = **true**;

**double** rand = Math.*random*();

**if**(rand < .26)

newDir = Direction.***NORTHWEST***;

**else** **if** (rand > .25 && rand < .5)

newDir = Direction.***NORTHEAST***;

**else** **if** (rand > .49 && rand < .75)

newDir = Direction.***SOUTHEAST***;

**else** **if** (rand > .74)

newDir = Direction.***SOUTHWEST***;

**return** Move.***TURNCOUNTERCLOCKWISE***;

}

**if**(**this**.turning)

{ **return** Move.***TURNCLOCKWISE***; }

**return** Move.***FORWARD***;

Method 2: Diagonal Movement, Multiple Infrared Sensors

Method 2 has the same diagonal movement capabilities of Method 1, but Method 2 adds multiple if statements testing for different values of infraredSensor. If the Roomba is 50, 100, or 200 units from the nearest object according to the infraredSensor, the Roomba will turn into a new diagonal direction. This strategy relies on the same strength of diagonal movement coverage as Method 1, but the Roomba will change directions more spontaneously. It is unsure, but possible, that the heavy use of infrared sensors and direction change will help prevent the Roomba from repeating an already clean path (a precaution that Method 1 does not account for). Note, the method this.WallSensor was removed from the program to simplify and focus the procedure and code on the infrared sensor.

**if**(**this**.turning && **this**.getDirection() == newDir)

**this**.turning = **false**;

**if**(**this**.frontBumper) {

turning = **true**;

**double** rand = Math.*random*();

**if**(rand < .26)

newDir = Direction.***NORTHWEST***;

**else** **if** (rand > .25 && rand < .5)

newDir = Direction.***NORTHEAST***;

**else** **if** (rand > .49 && rand < .75)

newDir = Direction.***SOUTHEAST***;

**else** **if** (rand > .74)

newDir = Direction.***SOUTHWEST***;

**return** Move.***TURNCOUNTERCLOCKWISE***;

}

**if**(**this**.infraredSensor == 50) {

turning = **true**;

**double** rand = Math.*random*();

**if**(rand < .26)

newDir = Direction.***NORTHWEST***;

**else** **if** (rand > .25 && rand < .5)

newDir = Direction.***NORTHEAST***;

**else** **if** (rand > .49 && rand < .75)

newDir = Direction.***SOUTHEAST***;

**else** **if** (rand > .74)

newDir = Direction.***SOUTHWEST***;

**return** Move.***FORWARD***;

}

**if**(**this**.infraredSensor == 100) {

turning = **true**;

**double** rand = Math.*random*();

**if**(rand < .26)

newDir = Direction.***NORTHWEST***;

**else** **if** (rand > .25 && rand < .5)

newDir = Direction.***NORTHEAST***;

**else** **if** (rand > .49 && rand < .75)

newDir = Direction.***SOUTHEAST***;

**else** **if** (rand > .74)

newDir = Direction.***SOUTHWEST***;

**return** Move.***FORWARD***;

}

**if**(**this**.infraredSensor == 200) {

turning = **true**;

**double** rand = Math.*random*();

**if**(rand < .26)

newDir = Direction.***NORTHWEST***;

**else** **if** (rand > .25 && rand < .5)

newDir = Direction.***NORTHEAST***;

**else** **if** (rand > .49 && rand < .75)

newDir = Direction.***SOUTHEAST***;

**else** **if** (rand > .74)

newDir = Direction.***SOUTHWEST***;

**return** Move.***FORWARD***;

}

**if**(**this**.turning)

{ **return** Move.***TURNCLOCKWISE***; }

**return** Move.***FORWARD***;

*Data*

|  |  |  |
| --- | --- | --- |
| Room Configuration\* | Method #1 Average (%) | Method #2 Average (%) |
| Size: Big (800 by 600 pixels)  Object *#*: Low (6 objects) | 14.8 | 32.1 |
| Size: Big(800 by 600 pixels)  Object *#*: High (12 objects) | 13.9 | 27.1 |
| Size: Small (700 by 500 pixels)  Object *#*: Low(6 objects) | 16.9 | 35.6 |
| Size: Small (700 by 500 pixels)  Object *#*: High (12 objects) | 15.9 | 32.1 |
| Overall Average (the averaged value of the means of the four experiments). | 15.4 | 31.7 |

*Conclusion:*

The experiment tested the effect of infrared sensors on the Roomba’s cleaning ability in a virtual simulator. Method 1’s program moved the Roomba in diagonal directions, only switching between directions when the Roomba’s front bumper sensor hit an object or wall. Method 2’s program moved the Roomba in diagonal directions, reorienting itself when either the front bumper sensor impacted an object or wall or when the infrared sensor read that the Roomba was 50, 100, or 200 units from the nearest wall or object. The data concludes that Method 2’s program, with an overall average of 31.7% cleaned tiles per room, surpasses the cleaning ability of Method 1, almost doubling Method 1’s overall average of 15.4% cleaned tiles per room; the hypothesis that Method 2’s use of infrared sensors and changing diagonal movement proved true as the Roomba did not retrace its path as often as Method 1 and significantly improved Method 2’s overall performance. Both methods scored the highest percentages of cleaning when moving through a small room with a low number of objects. The lowest percentages occurred when the two methods undertook trials in a large room and with a high number of objects. Errors include possible logic errors or bugs that may have occurred without notice and the erratic bugging of the randomization of objects that may have skewed the data. Further experiments include testing and comparing the specific distances of the infrared sensor (for values such as 50, 100, 200) and their cleaning efficiencies.