CS3105 Practical 1

160021428

15/03/2019

1. Overview

Following the notes and specification provided, a design that successfully makes use of fractional progress has been implemented to find the most optimal points to get to the goal. However, to maneuver around severe C shape obstacles BUG like implementation was used rather than the suggested winding and unwinding mechanic from the notes as it was simpler to implement and easier to understand. In addition, using the results gathered from the numerous tests conducted against both the provided and implemented planners the Overview questions have also been successfully answered. To run the program, just run the included Robot jar file and select the Fractional button to use the Fractional progress implementation. To get the same results from the tests conducted in this report, make sure to use the same move parameters set below.

2. Free space (Part 1)

2.1 *Move Parameters*

All the tests conducted for both Parts (Free Space & Obstacle Navigation) make use of a sensor density of 100, a range of 150, and a robot radius of 40, with a goal size of 40. This has been found to be the parameters that return the most accurate data and the least unexpected results not caused by the system or algorithm used. With a robot radius too small, the robot begins to shake and cause unnecessary turns when using fractional progress to get to the goal.

2.2 *Completeness*

To test the completeness of the planning system, several different orientations were against the planner to see if the robot can make it to the goal in any given orientation. An example of this can be seen in Appendix A.4, wherein the robot successfully reaches the goal at a simple orientation. Further tests and examples can be seen in Appendix A. The same tests have also been conducted on the Euclidean planner that show that it is also Complete when in free space.

2.3 *Graphical User Interface*

A Fractional Progress button has been added to the Planner section of the UI that allows for the Non-Greedy planner to be selected by the user. In addition, the total turn value of the robot and the number of steps taken have also been added to the information panel (in blue), as can be seen in Figure A of Section 2.2, to provide more detail to the journey of the robot.

In Comparison to the greedy Euclidean planner, the planned arc path of the non-greedy planner has shown itself to be more inefficient when the goal is directly in front of the robot. After analyzing and conducting a number of tests, the non-greedy planner seems to make slight adjustments to get the point with the most fractional progress. This, however, does not necessarily entail the point that is closest to the goal. In contrast, the greedy planner always prioritizes the point with the least obstacle potential to the goal which is very efficient for movement in free space when the goal is placed directly in front of the robot. This difference in priority can be seen from each of the arc drawings and travel statistics in Appendix B. This also seems to be the case for any goal position that is in front of the robot, not just directly in front of it.

If the goal is positioned anywhere behind the robot however, fractional progress is more efficient at turning and has a better path smoothness rating compared to the greedy planner but still takes a similar number of steps. The greedy planner’s inefficiency is due to it prioritizing points with the least obstacle potential, which in free space is 0. Therefore, it doesn’t necessarily pick the most efficient path to the goal and in turn does not turn as smoothly (see Appendix C for a comparison between the two planners with different goal locations).

2.4 *Potential and motion mechanisms*

To select the point with the most fractional progress, the fractional progress of each candidate point is to be taken with by using the formula

(Estimated future cost) / (Past cost + Estimated future cost)

wherein the past cost is the length of the first arc between the current point and the candidate point. The estimated future cost on the other hand is the sum of the remaining two arcs and the point’s obstacle potential. Summing in this obstacle potential allows the fractional progress to increase or decrease depending on the distance of the obstacle from the robot, thus giving the robot some form of obstacle navigation.

2.5 *Efficiency*

Given a straight path to the goal, the Euclidean planner seems to take a more efficient path than the non-greedy planner. As can be seen in Appendix C the fractional progress planner seems to be making quite a number of turns to take the path with the most fractional progress even with a path straight to the goal. This in turn increases the smoothness rating of the path showing that the path taken is not the most efficient to get to the goal. Due to the extra turns taken to find the point that makes the most fractional progress, the number of steps taken slightly increases by 1 to 5 steps. However, as stated previously, it is more efficient than the greedy planner when the path to the goal requires more turns to make. In the figures in Appendix C.3 and Appendix C.4, it can be seen that the robot using Fractional Progress has a smaller total turn value of 37.4 and a smoothness rating ranging between 1.2-1.4, whereas the Euclidean robot has a turn value of 58 and a smoothness rating of 1.5 and above.

2.6 *Set-up superiority*

In terms of overall efficiency, the non-greedy planner seems to be the more efficient choice between the two. Although the greedy planner is more efficient when travelling to a goal in front of the robot, the difference in steps taken is very small, whereas the difference in terms of turn value is drastic for a goal that requires more turns to get to.

3. Obstacle Navigation (Part 2)

3.1 *Obstacle Courses*

Despite being more efficient at certain aspects of free space travel, the Euclidean planner is only able to maneuver around the easy obstacle course with at these move parameters. It is unable to traverse the medium and hard obstacle courses as it gets stuck in a spinning loop and sometimes runs out of moves inside the C shape in the hard course (see Appendix D). This is due to its design wherein it tries to move to the point that gets it closer to the goal with the least obstacle potential, which is what causes it to end up spinning as it tries to move to the point with the least obstacle potential and eventually brings it to free space, then it tries to move back to the goal repeating the process. This is phenomenon is known as the local minimum problem.

Unlike the greedy field planner, the non-greedy implementation has been extended to determine when it is necessary to take an alternative route to reach the goal. Instead of just moving towards the point that makes the most fractional progress, the planner now also considers the obstacle distance of each sample point to determine the most viable point under a certain threshold. By implementing this, the non-greedy field planner can now maneuver around obstacles including the hard obstacle course which can be seen in Appendix D. To further test the capabilities of the implemented obstacle navigation, varying degrees of C-shaped obstacles were created and added as buttons to the GUI. The first shape the robot is tested against is the Circular C, which can be seen in Appendix E.1 and is the least severe C-shaped obstacle. When tested against this obstacle, the robot is successfully able to maneuver around it and get to the goal (see the rest of Appendix E for the other custom obstacle courses).

3.2 *Potential and motion mechanisms*

As briefly stated in section 3.1, the planner has been implemented with a method of dealing with the local minimum problem by determining the most viable point to move towards by considering the obstacle potential of each candidate point. This is based off the POTBUG implementation that combines BUG-like maneuvering around obstacles using obstacle potential. In the figures of Appendices D and E, it can be observed that the robot follows the shape of the obstacle until the path from it to the obstacle is deemed as clear. This action is done whenever the obstacle potential of a point crosses the set threshold of the robot. The default threshold is set as:

(radius of the robot *1.5)

which has been found to be the threshold with the best and most consistent results. If an obstacle goes over the threshold, the robot will attempt to "wind" and "unwind". Although this implementation does not make use of winding or unwinding, the actions will still be referred to as such, due to the lack of a better name. Once the robot is in its winding and unwinding state, it will attempt to look for one of three points from its sample points that are under the obstacle potential threshold and potentially select one to move towards.

The first of the three points is the unwinding point, wherein the robot can move safely away from the obstacle and move towards the goal. This is determined by a checking if the distance of the hit-point from the goal is greater than the distance of the current candidate point to the goal. The logic used to justify this is if the distance of the hit-point from the goal is greater than the current candidate point to the goal, this implies that there is a point that is closer to the goal than the obstacle it is winding around. Therefore, implying that the robot is on the other side of the obstacle. This, however, does not always guarantee that the robot has successfully wound around the object as there are some structures such as the C-shaped obstacle found in the hard obstacle course that cause some exceptional problems (see Figure D.5). Where the robot is on that figure has its rightmost point out in free-space which is closer to the goal than the current hit-point which is on its leftmost sensor. This in the robots logic, implies that moving towards the point furthest away from the hit-point and closer to the point closest to the goal will get it to free-space and eventually, the goal. This undoubtably will cause the robot to loop around inside the C shape that it is currently trying to wind out of. To deal with this, two variables called intensity and sensitivity have been introduced. Sensitivity is the difference between the

(current heading of the robot – the heading of the goal)

which will give the value that determines whether the robot is looking directly at the goal. The closer the value is to zero, the closer it is to looking directly at the goal. On the other hand, Intensity is formulated by:

The absolute value of (radius * sensitivity)

which determines how much the robot will stick to the obstacle by adding its value to the distance of the candidate point to the goal. The intensity increases as the robot looks away from the goal, hence, forcing the robot to wind around the obstacle until it reaches a point on the obstacle where it can look near enough to the goal that the intensity gets close enough to zero. Since an unwind point is determined by finding a point that is closer to the goal than the hit-point, adding the intensity to the closest point to the goal will prevent the point from being added as a viable unwind point.

The second of the three points is the wind point, which is referred to as the point that has an obstacle potential that is closest to the threshold without going over it. This point is very simple as it just measures from among all the candidate points, which among them has the greatest obstacle potential and selects that as the wind point. This point allows the robot to trace along the outline of the obstacle without getting too close until an unwind point is found.

The last of the three points is just the point with the most fractional progress. This point will be used as a last resort for the robot to move towards in the event that there are no viable winding or unwinding points found within the robot’s sensor.

3.3 *Efficiency & Planner type superiority*

Due to the Euclidean planner being unable to successfully maneuver around even the medium obstacle course, the planner with the more efficient obstacle navigation must be the non-greedy planner. Given an easier obstacle course, the greedy planner may still be able to successfully move around the obstacles as it has very minimal obstacle navigation, but is still more inefficient in terms of path smoothness, the total turn value, and the number of steps taken. As can be seen in the Figures in Appendix F.

3.4 *Evaluation*

Although the unwinding works as expected on the custom C-shaped obstacles and the pre-built obstacle courses with the given settings, some smaller sizes of the robot and sensor range may cause the measured threshold to not catch obstacles that are too close to the robot. This is caused by sensor ranges that are smaller than the threshold. Since the threshold is the (radius of the robot * 1.5) a sensor range that is too small will cause certain sample points to cross the radius and make the robot take an invalid point. This can potentially be fixed in the future by selecting a more dynamic threshold that is not too dependent on the radius of the robot. If however, the sensor range is large enough but still quite small, the robot may not be able to wind due to fractional progress already having a small portion of obstacle navigation already implemented, causing the robot to be pushed just far enough away from the obstacle to not cross over the threshold (see Appendix G for examples of exceptional cases).

4. Overview Questions

4.1 *Overview Question 1*

The reason potential fields suffer from the local minimum problem is because they are entirely dependent on the starting position of the robot, the position of the obstacle, and the position of the goal. For instance, using the Euclidean planner, the robot gets stuck inside the C curve obstacle of the hard obstacle course because it always attempts to take the shortest path possible without considering the shape of the obstacle. It innately has some form of obstacle navigation as it can move around objects, but it does not have the sense to move around an obstacle blocking its way which causes it to fall into the local minimum problem.

4.2 *Overview Question 2*

As stated in the evaluation section of Section 3, there are certain exceptional cases which may trigger the current implementation of POTBUG to fail and fall into a local minimum trap. One of these exceptional cases may be when the robot enters a D-shaped obstacle and it is inside of it with the goal over the D. This may cause the method of unwinding or detaching from the object to fail and fall into a loop. Perhaps in the future there could be some other method of countering this.

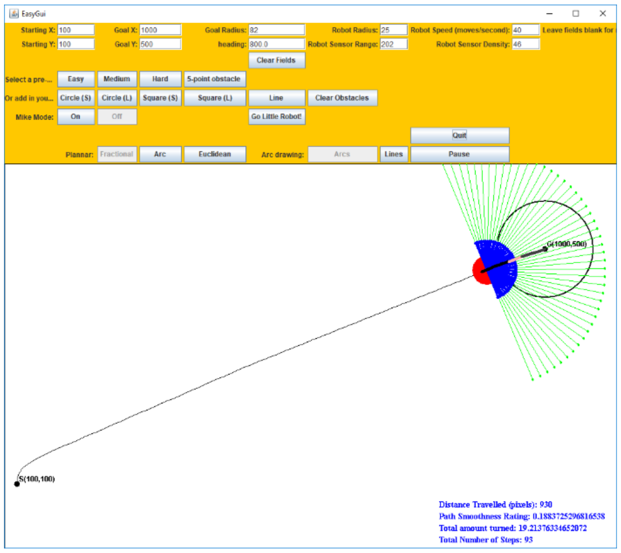
5. Conclusion

To conclude, the implementation of a non-greedy planner has been successful overall and is able to pass all the obstacles that has been tested against it. I believe a substantial number of tests have been used to show that this implementation is viable and works as intended. Although winding has not been implemented, there are certain aspects that seem quite interesting and can potentially be implemented to improve this in the future.

6. Appendix

Appendix A

Fractional Progress Completeness

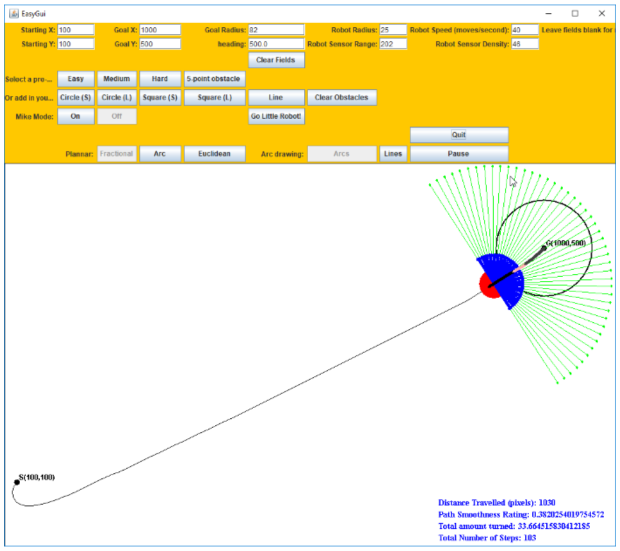


Starting X: 100 Goal X: 1000 Goal Radius: 82 Robot Radius: 25 Robot Speed (moves/second): 40 Leave fields blank for

Starting Y: 100 Goal Y: 500 heading: 800.0 Robot Sensor Range: 202 Robot Sensor Density: 46

Distance Travelled (pixels): 930
Path Smoothness Rating: 0.1883725296816538
Total amount turned: 19.21376334652072
Total Number of Steps: 93

Appendix A.1: Heading 800

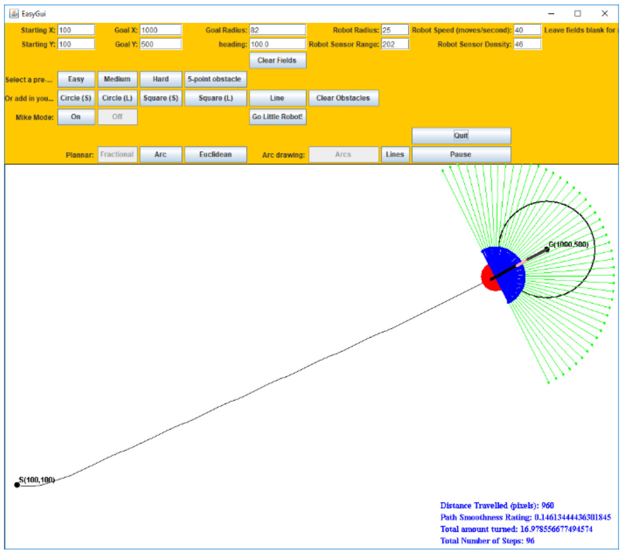


Starting X: 100 Goal X: 1000 Goal Radius: 82 Robot Radius: 25 Robot Speed (moves/second): 40 Leave fields blank for

Starting Y: 100 Goal Y: 500 heading: 500.0 Robot Sensor Range: 202 Robot Sensor Density: 46

Distance Travelled (pixels): 1030
Path Smoothness Rating: 0.3820254019754572
Total amount turned: 33.664515830412185
Total Number of Steps: 103

Appendix A.2: Heading 500

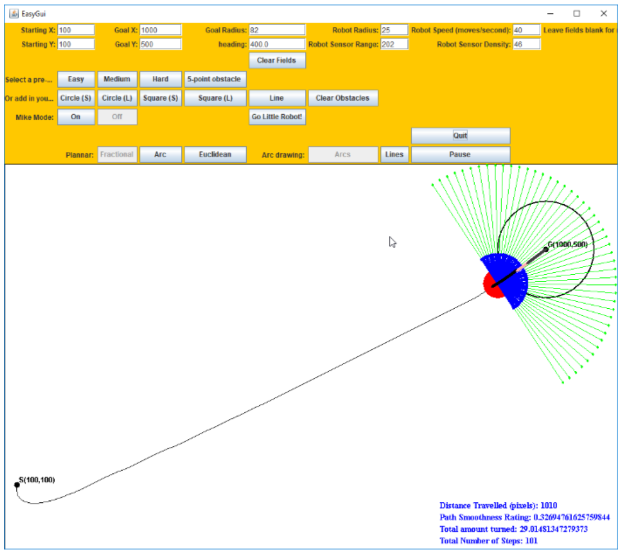


Starting X: 100 Goal X: 1000 Goal Radius: 82 Robot Radius: 25 Robot Speed (moves/second): 40 Leave fields blank for

Starting Y: 100 Goal Y: 500 heading: 100.0 Robot Sensor Range: 202 Robot Sensor Density: 46

Distance Travelled (pixels): 960
Path Smoothness Rating: 0.14613444436301845
Total amount turned: 16.978556677494574
Total Number of Steps: 96

Appendix A.3: Heading 400



Starting X: 100 Goal X: 1000 Goal Radius: 82 Robot Radius: 25 Robot Speed (moves/second): 40 Leave fields blank for

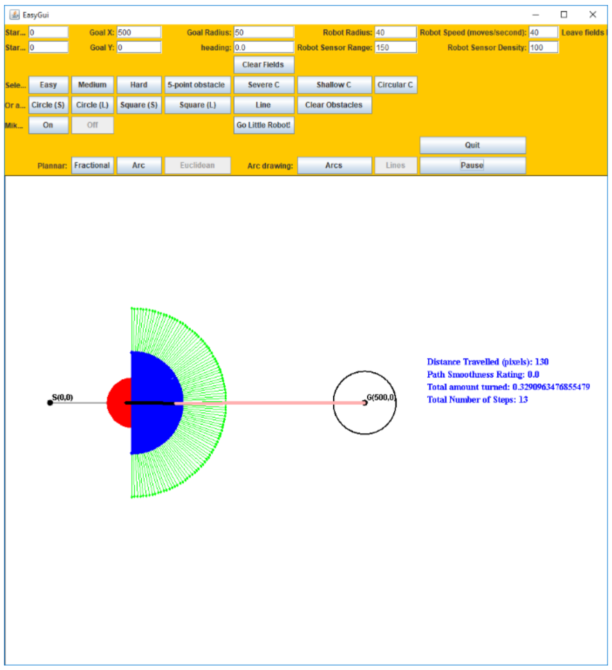
Starting Y: 100 Goal Y: 500 heading: 400.0 Robot Sensor Range: 202 Robot Sensor Density: 46

Distance Travelled (pixels): 1010
Path Smoothness Rating: 0.32694761625759844
Total amount turned: 29.01481347279373
Total Number of Steps: 101

Appendix A.4: Heading 100

Appendix B

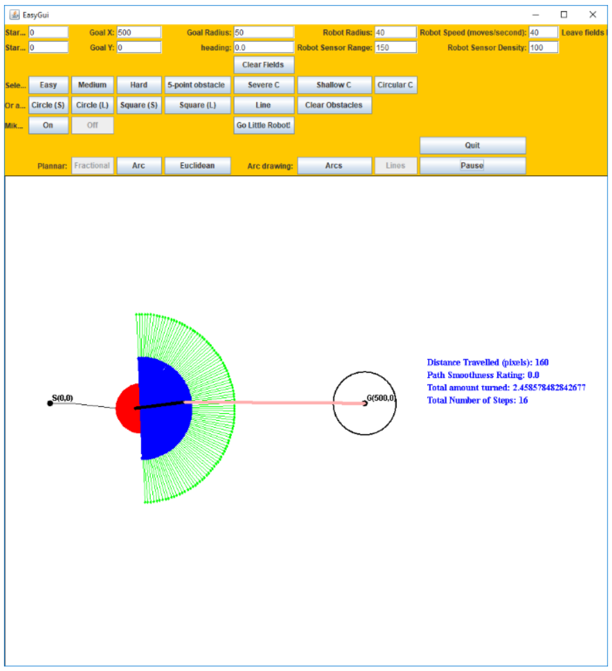
Arc Drawing comparison (Free space)



Star... 0 Goal X: 500 Goal Radius: 50 Robot Radius: 40 Robot Speed (moves/second): 40 Leave fields
Star... 0 Goal Y: 0 heading: 0.0 Robot Sensor Range: 150 Robot Sensor Density: 100

Distance Travelled (pixels): 130
Path Smoothness Rating: 0.0
Total amount turned: 0.3290963476855479
Total Number of Steps: 13

Appendix B.1: Euclidean Arc Drawing



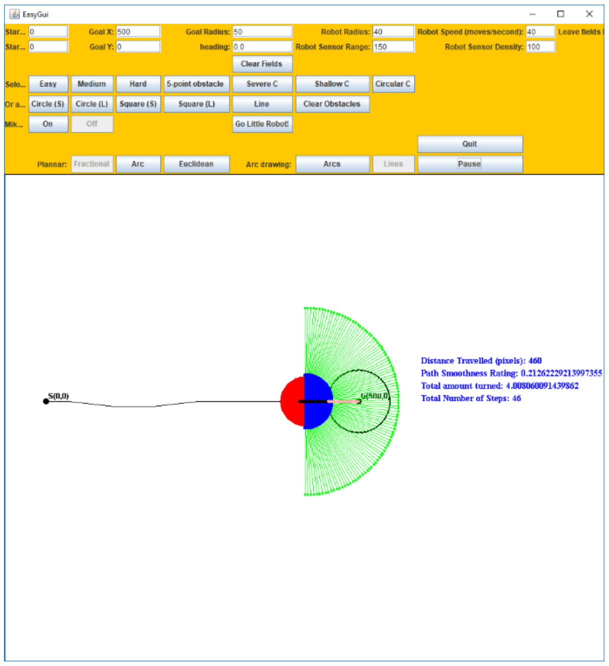
Star... 0 Goal X: 500 Goal Radius: 50 Robot Radius: 40 Robot Speed (moves/second): 40 Leave fields
Star... 0 Goal Y: 0 heading: 0.0 Robot Sensor Range: 150 Robot Sensor Density: 100

Distance Travelled (pixels): 160
Path Smoothness Rating: 0.0
Total amount turned: 2.458578482842677
Total Number of Steps: 16

Appendix B.2: Fractional Progress Arc Drawing

Appendix C

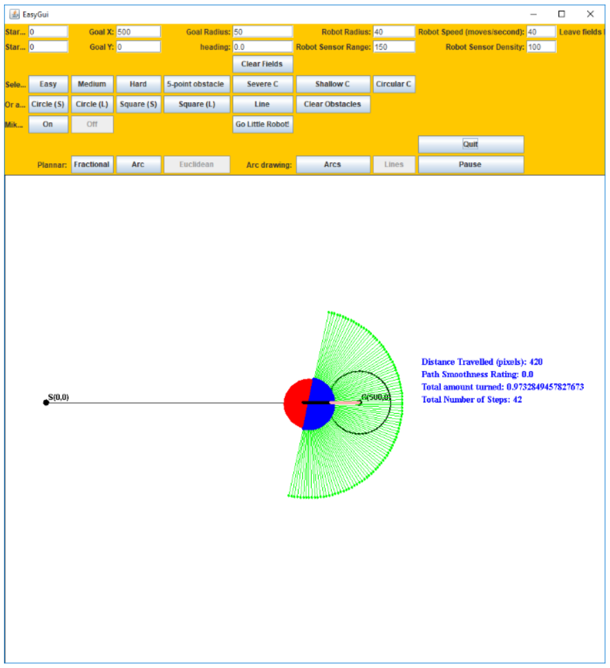
Moving to the goal in Free Space



Star... 0 Goal X: 500 Goal Radius: 50 Robot Radius: 40 Robot Speed (moves/second): 40 Leave fields
Star... 0 Goal Y: 0 heading: 0.0 Robot Sensor Range: 150 Robot Sensor Density: 100

Distance Travelled (pixels): 460
Path Smoothness Rating: 0.21262229213997355
Total amount turned: 4.008060091439862
Total Number of Steps: 46

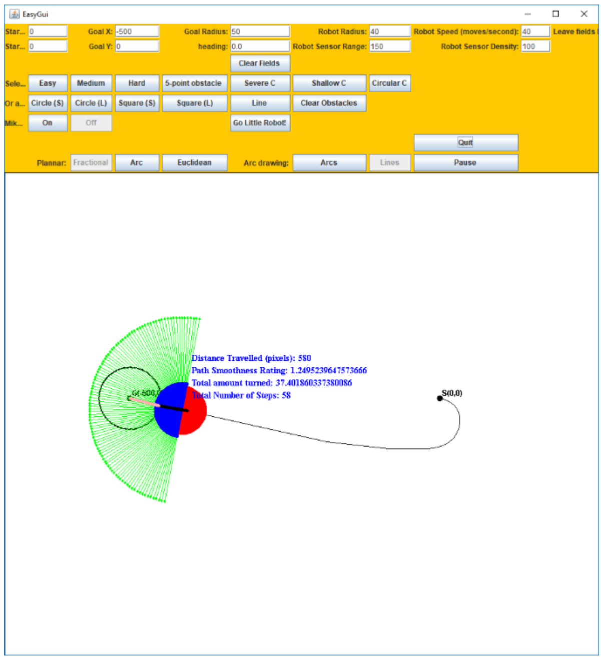
Appendix C.1: Straight path (Fractional)



Star... 0 Goal X: 500 Goal Radius: 50 Robot Radius: 40 Robot Speed (moves/second): 40 Leave fields
Star... 0 Goal Y: 0 heading: 0.0 Robot Sensor Range: 150 Robot Sensor Density: 100

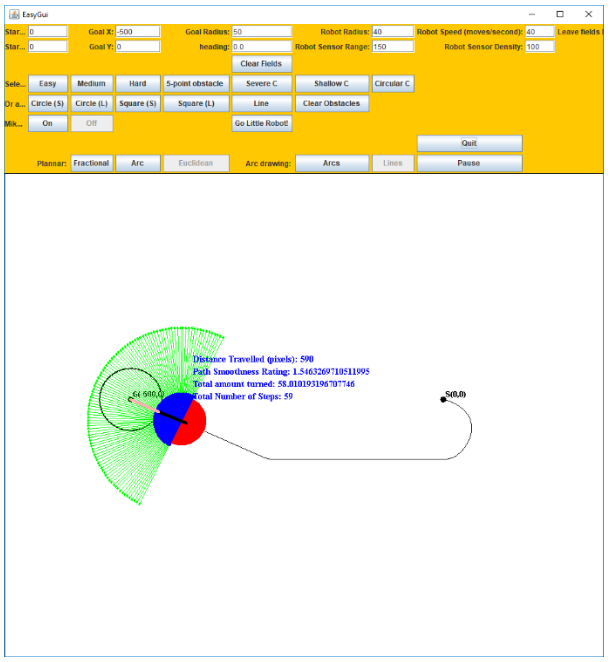
Distance Travelled (pixels): 420
Path Smoothness Rating: 0.0
Total amount turned: 0.9732849457827673
Total Number of Steps: 42

Appendix C.2: Straight path (Euclidean)


```
EasyGui
Star... 0        Goal X: -500    Goal Radius: 50     Robot Radius: 40            Robot Speed (moves/second): 40
Star... 0        Goal Y: 0       heading: 0.0        Robot Sensor Range: 150     Robot Sensor Density: 100
```

```
Distance Travelled (pixels): 580
Path Smoothness Rating: 1.2495239647573666
Total amount turned: 37.401860337380086
Total Number of Steps: 58
```

Appendix C.3: 180 degree turn (Fractional)

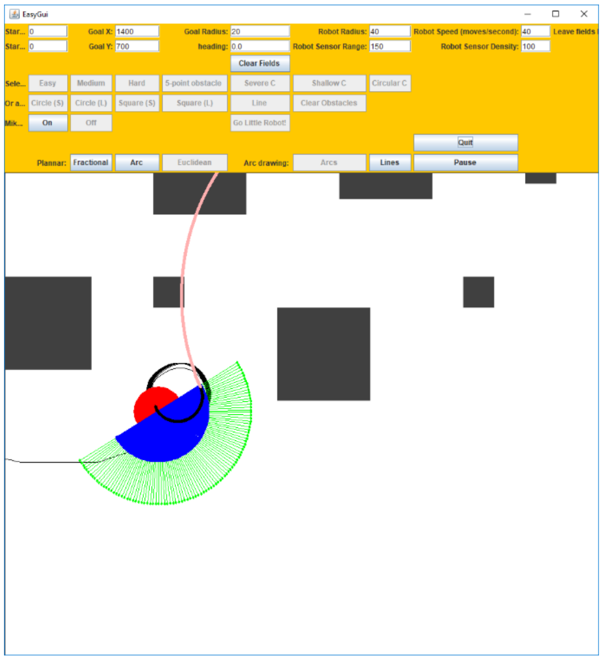

```
EasyGui
Star... 0        Goal X: -500    Goal Radius: 50     Robot Radius: 40            Robot Speed (moves/second): 40
Star... 0        Goal Y: 0       heading: 0.0        Robot Sensor Range: 150     Robot Sensor Density: 100
```

```
Distance Travelled (pixels): 590
Path Smoothness Rating: 1.5463269710511995
Total amount turned: 58.010193196707746
Total Number of Steps: 59
```

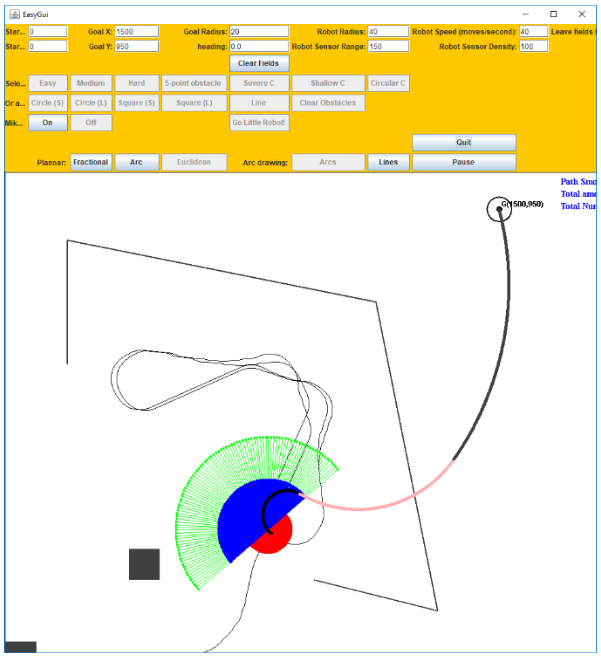
Appendix C.4: 180 degree turn (Euclidean)

Appendix D

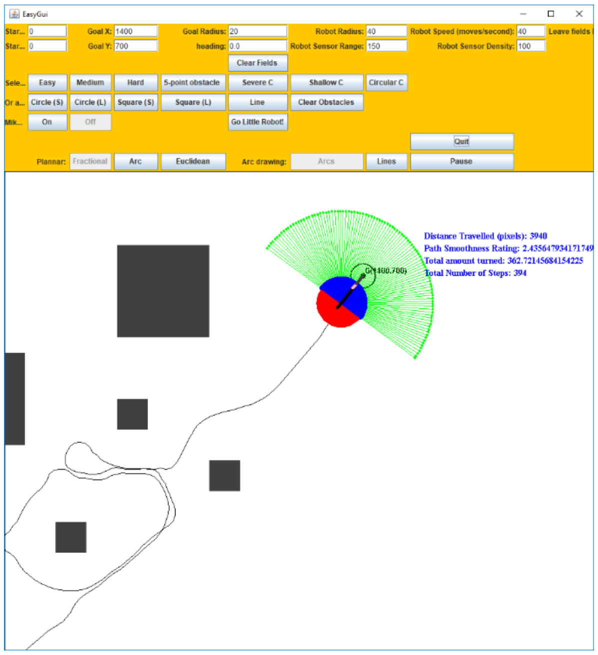
Obstacle courses


```
EasyGui
Star... 0        Goal X: 1400    Goal Radius: 20     Robot Radius: 40            Robot Speed (moves/second): 40
Star... 0        Goal Y: 700     heading: 0.0        Robot Sensor Range: 150     Robot Sensor Density: 100
```

Appendix D.1: Medium Obstacle course (Euclidean)


```
EasyGui
Star... 0        Goal X: 1500    Goal Radius: 20     Robot Radius: 40            Robot Speed (moves/second): 40
Star... 0        Goal Y: 950     heading: 0.0        Robot Sensor Range: 150     Robot Sensor Density: 100
```

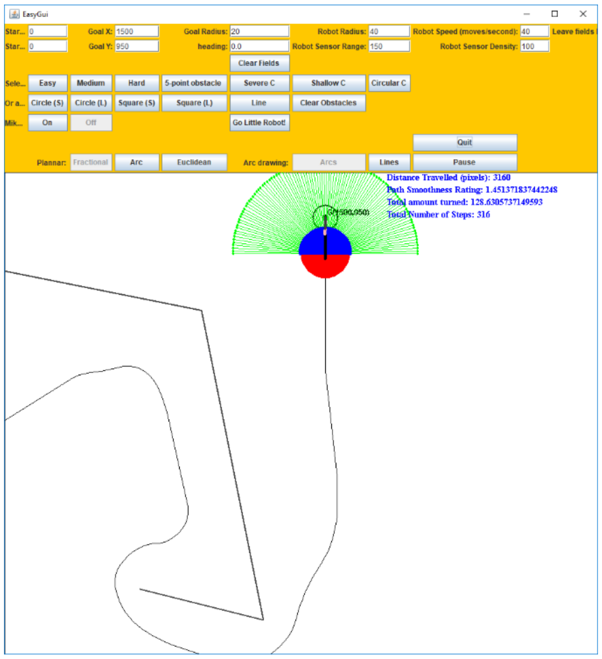
Appendix D.2: Hard Obstacle course (Euclidean)



Star... 0
Star... 0
Goal X: 1400
Goal Y: 700
Goal Radius: 20
heading: 0.0
Robot Radius: 40
Robot Sensor Range: 150
Robot Speed (moves/second): 40
Robot Sensor Density: 100
Leave fields
Sele...
Or a...
Mik...
Plannar:
Arc drawing:

Distance Travelled (pixels): 3940
Path Smoothness Rating: 2.435647934171749
Total amount turned: 362.72145684154225
Total Number of Steps: 394

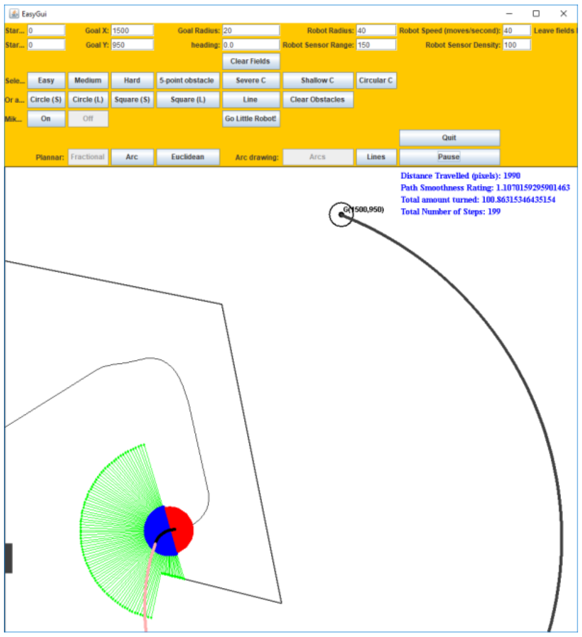
Appendix D.3: Medium Obstacle course (Fractional)



Star... 0
Star... 0
Goal X: 1500
Goal Y: 950
Goal Radius: 20
heading: 0.0
Robot Radius: 40
Robot Sensor Range: 150
Robot Speed (moves/second): 40
Robot Sensor Density: 100
Leave fields
Sele...
Or a...
Mik...
Plannar:
Arc drawing:

Distance Travelled (pixels): 3160
Path Smoothness Rating: 1.451371837442248
Total amount turned: 128.6305737149593
Total Number of Steps: 316

Appendix D.4: Hard Obstacle Course (Fractional)



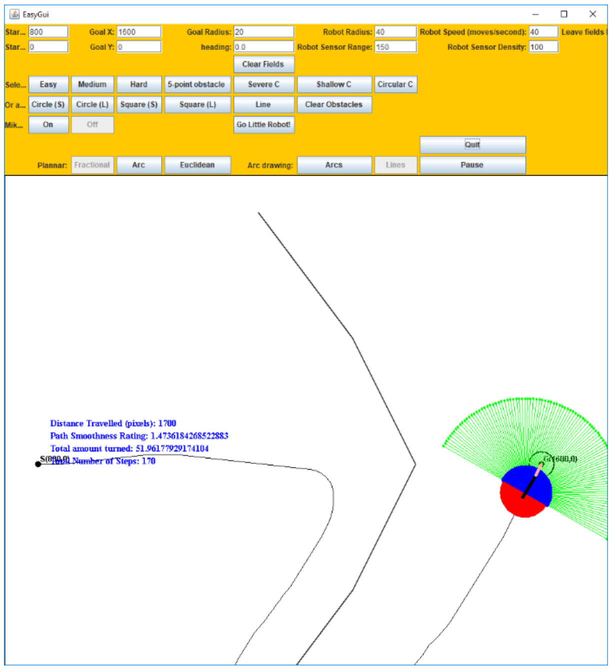
Star... 0
Star... 0
Goal X: 1500
Goal Y: 950
Goal Radius: 20
heading: 0.0
Robot Radius: 40
Robot Sensor Range: 150
Robot Speed (moves/second): 40
Robot Sensor Density: 100
Leave fields
Sele...
Or a...
Mik...
Plannar:
Arc drawing:

Distance Travelled (pixels): 1990
Path Smoothness Rating: 1.1070159295901463
Total amount turned: 100.86315346435154
Total Number of Steps: 199

Appendix D.5: Exceptional Case (Fractional)

Appendix E

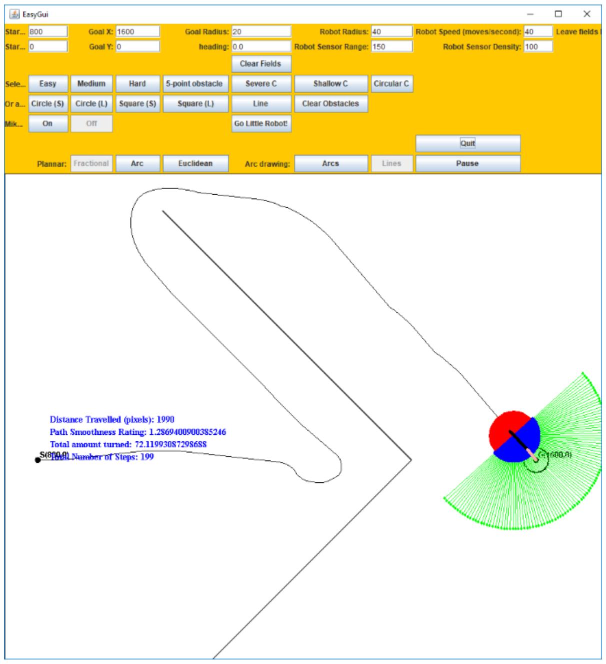
Custom obstacle Courses



Star... 800　Goal X: 1600　Goal Radius: 20　Robot Radius: 40　Robot Speed (moves/second): 40　Leave fields
Star... 0　Goal Y: 0　heading: 0.0　Robot Sensor Range: 150　Robot Sensor Density: 100

```
Distance Travelled (pixels): 1700
Path Smoothness Rating: 1.4736184268522883
Total amount turned: 51.96177929174104
Total Number of Steps: 170
```

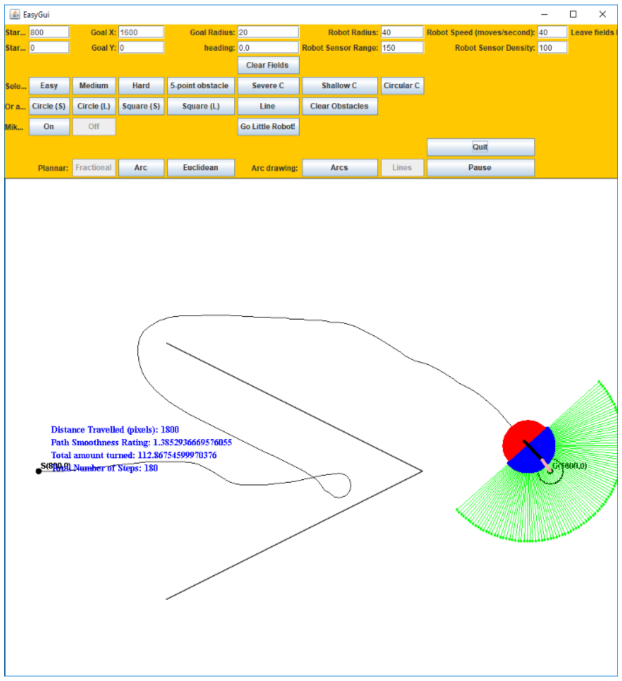
Appendix E.1: Circular C obstacle



Star... 800　Goal X: 1600　Goal Radius: 20　Robot Radius: 40　Robot Speed (moves/second): 40　Leave fields
Star... 0　Goal Y: 0　heading: 0.0　Robot Sensor Range: 150　Robot Sensor Density: 100

```
Distance Travelled (pixels): 1990
Path Smoothness Rating: 1.2869400900385246
Total amount turned: 72.11993087298688
Total Number of Steps: 199
```

Appendix E.2: Shallow C obstacle



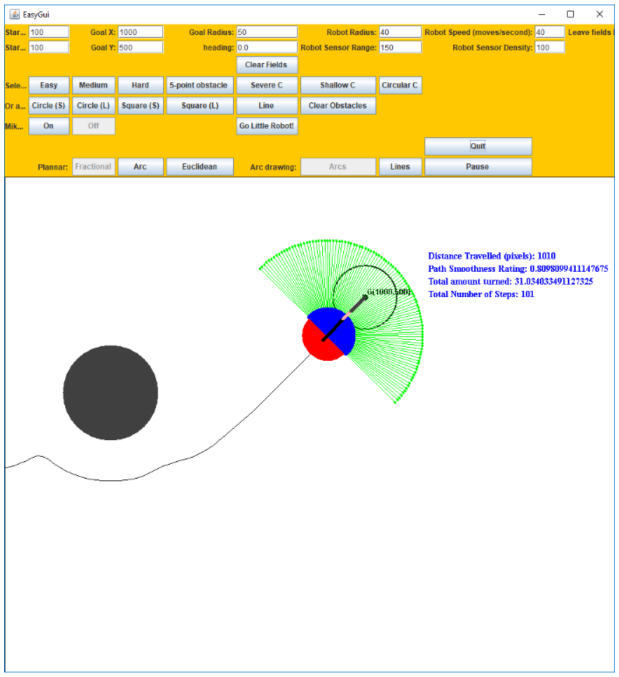
Star... 800　Goal X: 1600　Goal Radius: 20　Robot Radius: 40　Robot Speed (moves/second): 40　Leave fields
Star... 0　Goal Y: 0　heading: 0.0　Robot Sensor Range: 150　Robot Sensor Density: 100

```
Distance Travelled (pixels): 1800
Path Smoothness Rating: 1.3852936669576055
Total amount turned: 112.86754599970376
Total Number of Steps: 180
```

Appendix E.3: Severe C obstacle

Appendix F

Easy Obstacle Course Comparison

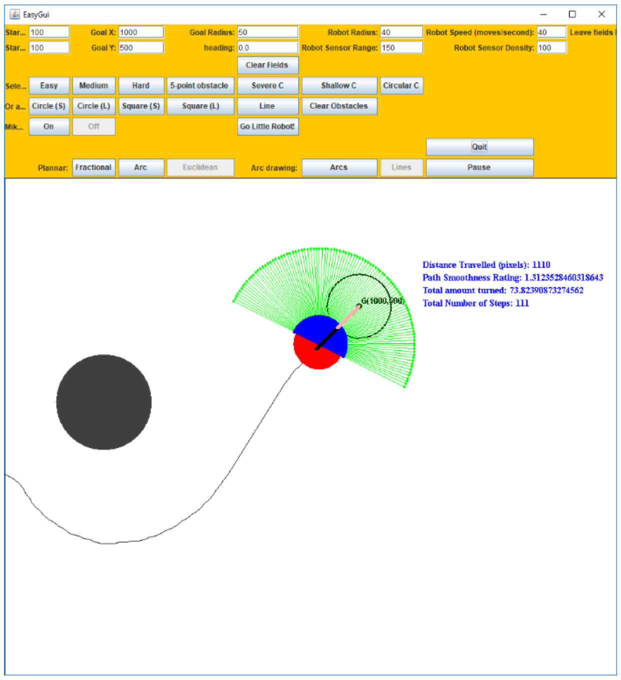


```
EasyGui
Star... 100    Goal X: 1000   Goal Radius: 50           Robot Radius: 40    Robot Speed (moves/second): 40   Leave fields
Star... 100    Goal Y: 500    heading: 0.0   Robot Sensor Range: 150        Robot Sensor Density: 100
Plannar: Fractional   Arc   Euclidean      Arc drawing: Arcs   Lines
```

```
Distance Travelled (pixels): 1010
Path Smoothness Rating: 0.8098099411147675
Total amount turned: 31.034033491127325
Total Number of Steps: 101
```

Clear Fields
Sele... Easy Medium Hard 5-point obstacle Severe C Shallow C Circular C
Or a... Circle (S) Circle (L) Square (S) Square (L) Line Clear Obstacles
Mik... On Off
Go Little Robot!
Quit
Pause
G(1000,500)

Appendix F.1: Easy Obstacle Course (Fractional)



```
EasyGui
Star... 100    Goal X: 1000   Goal Radius: 50           Robot Radius: 40    Robot Speed (moves/second): 40   Leave fields
Star... 100    Goal Y: 500    heading: 0.0   Robot Sensor Range: 150        Robot Sensor Density: 100
Plannar: Fractional   Arc   Euclidean      Arc drawing: Arcs   Lines
```

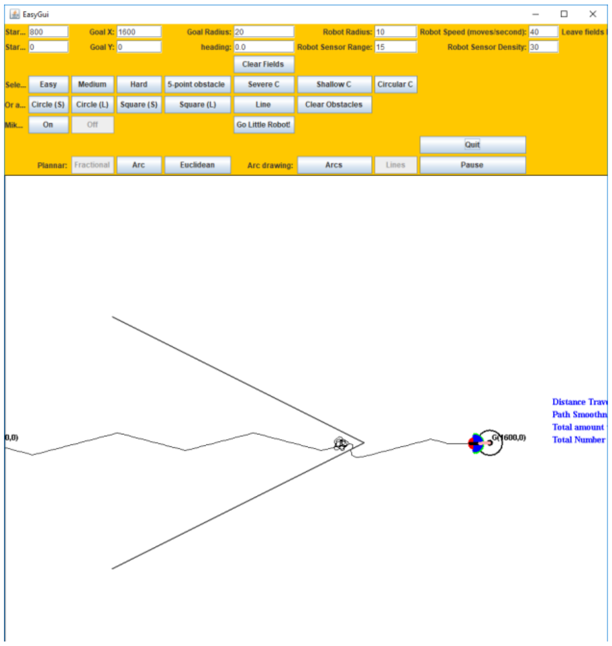
```
Distance Travelled (pixels): 1110
Path Smoothness Rating: 1.3123528460318643
Total amount turned: 73.82390873274562
Total Number of Steps: 111
```

Clear Fields
Sele... Easy Medium Hard 5-point obstacle Severe C Shallow C Circular C
Or a... Circle (S) Circle (L) Square (S) Square (L) Line Clear Obstacles
Mik... On Off
Go Little Robot!
Quit
Pause
G(1000,500)

Appendix F.2: Easy Obstacle Course (Euclidean)

Appendix G

Exceptional cases/settings that cause errors

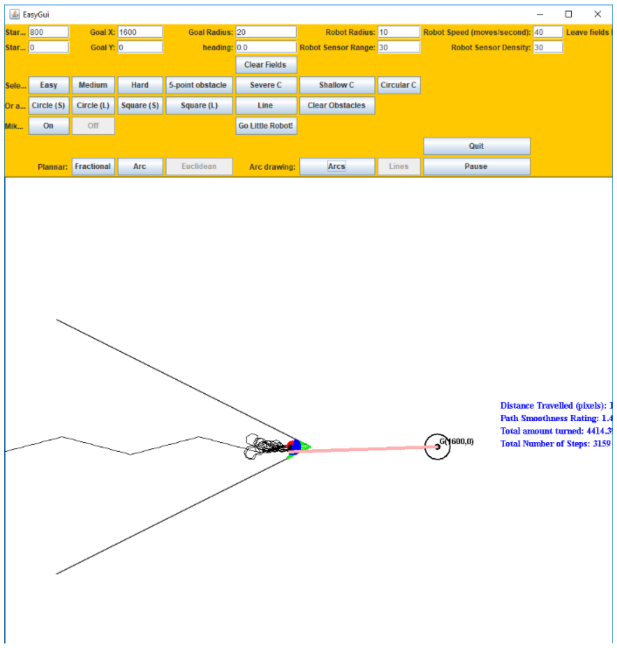


```
EasyGui
Star... 800    Goal X: 1600   Goal Radius: 20           Robot Radius: 10    Robot Speed (moves/second): 40   Leave fields
Star... 0      Goal Y: 0      heading: 0.0   Robot Sensor Range: 15         Robot Sensor Density: 30
Plannar: Fractional   Arc   Euclidean      Arc drawing: Arcs   Lines
```

```
Distance Trave
Path Smoothn
Total amount
Total Number
```

Clear Fields
Sele... Easy Medium Hard 5-point obstacle Severe C Shallow C Circular C
Or a... Circle (S) Circle (L) Square (S) Square (L) Line Clear Obstacles
Mik... On Off
Go Little Robot!
Quit
Pause
0,0)
G(1600,0)

Appendix G.1: Radius of 10 and sensor range of 15



```
EasyGui
Star... 800    Goal X: 1600   Goal Radius: 20           Robot Radius: 10    Robot Speed (moves/second): 40   Leave fields
Star... 0      Goal Y: 0      heading: 0.0   Robot Sensor Range: 30         Robot Sensor Density: 30
Plannar: Fractional   Arc   Euclidean      Arc drawing: Arcs   Lines
```

```
Distance Travelled (pixels): 1
Path Smoothness Rating: 1.4
Total amount turned: 4414.3
Total Number of Steps: 3159
```

Clear Fields
Sele... Easy Medium Hard 5-point obstacle Severe C Shallow C Circular C
Or a... Circle (S) Circle (L) Square (S) Square (L) Line Clear Obstacles
Mik... On Off
Go Little Robot!
Quit
Pause
G(1600,0)

Appendix G.2: Radius of 10 and sensor of 30