Luke Heary

Introduction

The main goal of this project was to use parallel and distributed systems in order to make N number of robots search and find a target on a grid, and then once the target is found, move all of the robots to surround the target. The robots move by communicating with each other, and trying to figure out which way to move is most efficient in order to move each robot the least amount possible.

Algorithms

When you run the program, you're first asked to input the height, width, and the number of robots you want to search for the target. Once you input that information, the program then calls the createGrid() function, which makes the 2D array for the grid, initializes all of the cells to 0, and then places a random "1" on the grid which represents the target. After that, the grid is returned, and the createRobots() function is called. It then uses the number of robots that the user inputted, and creates a new thread for each robot. Each robot is represented by a "2" on the board, and each robot object is a structure.

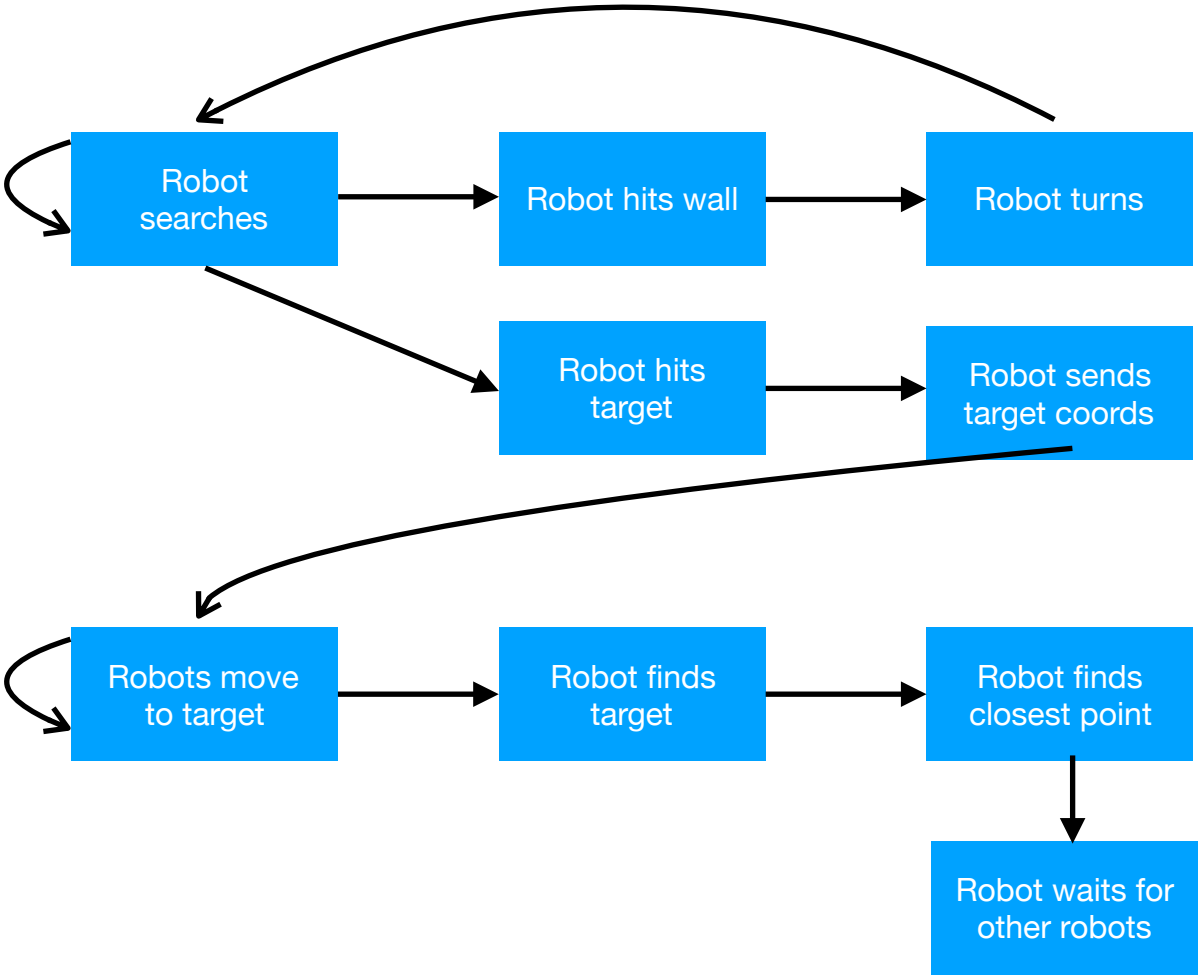
Inside each robot structure is all of the information that the robot needs to function properly. The current row and current column are stored as integers, as well as the initial row and column. There is also a spot in the struct that is used to hold the target row and target column, although they aren't initialized until one robot discovers the target on the board. The robot also has two booleans; touchingRobot and atTarget. The touchingRobot boolean is true if the robot is touching another robot, and the atTarget boolean is true if the robot is touching the target. These two booleans are used when figuring out if all of the robots are in their proper spots and are used to eventually stop the programming once all the robots are around the target.

The way the robot moving algorithm works, is that to start, the algorithm thats the current robot number, divides it by four, and gets the remainder. This will then give you a remainder of either 0, 1, 2, or 3. Based on these four numbers, the robot will then either move north, south, east, or west. The robot then travels in that direction until it either hits a wall or corner, in which it will then switch directions and travel a different way, hits another robot, and using parallel systems and message passing, the robots talk to each other and tell each of the two robots will then turn around and travel in different in directions, or lastly, there's an .125% chance that a robot turns in a random direction in order to give spread the robots in a more diverse direction.

Once a robot finds the target, the target location is then, using distributed systems, is multicasted to all of the other robots. From this point, the robots stop moving in random directions and instead travel towards the robot coming in first from the bottom and the top, so that the robots are in the same row as the target, and then the robots move inward left and right until they either hit the target or hit another robot. From there, it then makes sure that the cardinal directions of the target are filled, so the robots move up, down, or sideways in order to get to these positions. The other robots then try to also work there way in so that they are also in a spot closest to the target.

When the all of the robots are where they should be, the program stops. At the very end, the time it took for all of the robots to search and get to the target is displayed, as well as the number of moves that each robot took in order to get to the target.

State Diagram

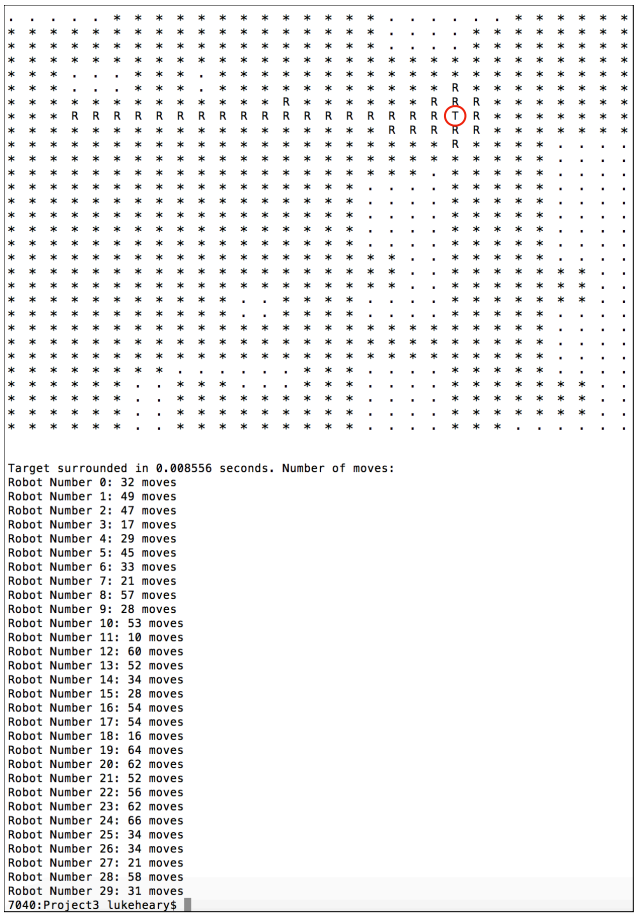


Results

First Test:

Grid Height: 30 cells
Grid Width: 30 cells
Total cells: 900
Number of robots: 30

Time taken: 0.008556

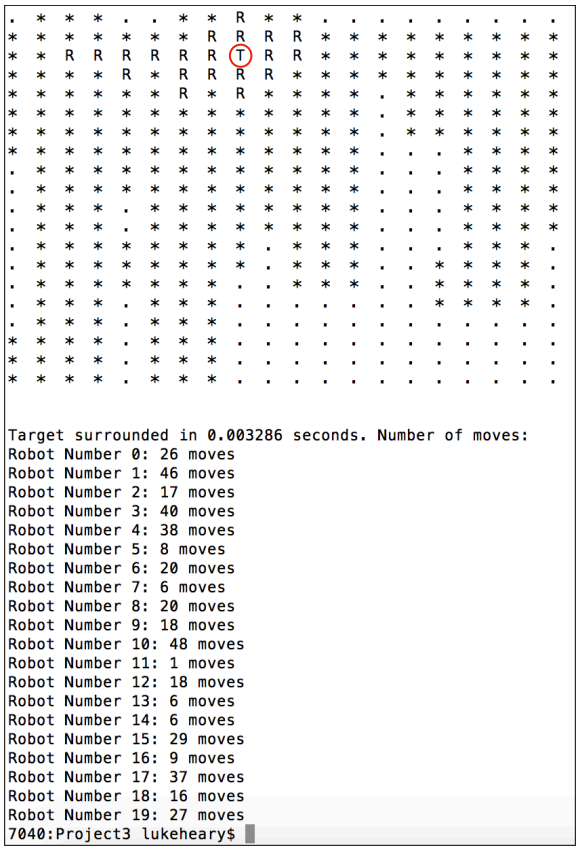


```
Target surrounded in 0.008556 seconds. Number of moves:
Robot Number 0: 32 moves
Robot Number 1: 49 moves
Robot Number 2: 47 moves
Robot Number 3: 17 moves
Robot Number 4: 29 moves
Robot Number 5: 45 moves
Robot Number 6: 33 moves
Robot Number 7: 21 moves
Robot Number 8: 57 moves
Robot Number 9: 28 moves
Robot Number 10: 53 moves
Robot Number 11: 10 moves
Robot Number 12: 60 moves
Robot Number 13: 52 moves
Robot Number 14: 34 moves
Robot Number 15: 28 moves
Robot Number 16: 54 moves
Robot Number 17: 54 moves
Robot Number 18: 16 moves
Robot Number 19: 64 moves
Robot Number 20: 62 moves
Robot Number 21: 52 moves
Robot Number 22: 56 moves
Robot Number 23: 62 moves
Robot Number 24: 66 moves
Robot Number 25: 34 moves
Robot Number 26: 34 moves
Robot Number 27: 21 moves
Robot Number 28: 58 moves
Robot Number 29: 31 moves
7040:Project3 lukeheary$
```

Second Test:

Grid Height: 20 cells
Grid Width: 20 cells
Total cells: 400
Number of robots: 20

Time taken: .003286

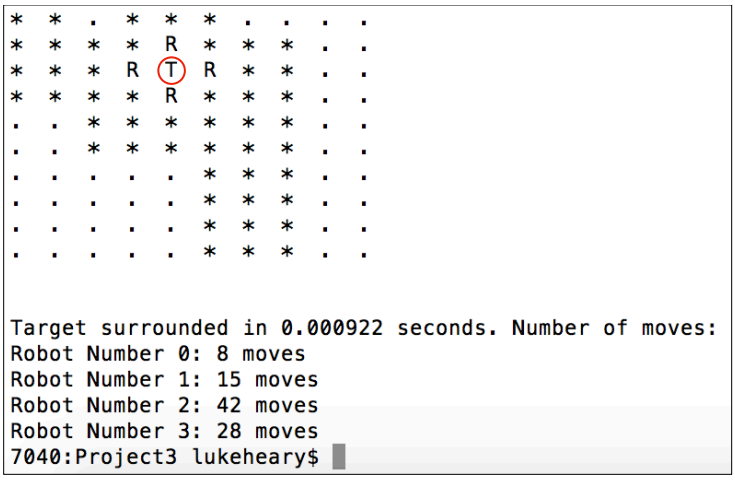


```
Target surrounded in 0.003286 seconds. Number of moves:
Robot Number 0: 26 moves
Robot Number 1: 46 moves
Robot Number 2: 17 moves
Robot Number 3: 40 moves
Robot Number 4: 38 moves
Robot Number 5: 8 moves
Robot Number 6: 20 moves
Robot Number 7: 6 moves
Robot Number 8: 20 moves
Robot Number 9: 18 moves
Robot Number 10: 48 moves
Robot Number 11: 1 moves
Robot Number 12: 18 moves
Robot Number 13: 6 moves
Robot Number 14: 6 moves
Robot Number 15: 29 moves
Robot Number 16: 9 moves
Robot Number 17: 37 moves
Robot Number 18: 16 moves
Robot Number 19: 27 moves
7040:Project3 lukeheary$
```

Third Test:

Grid Height: 10 cells
Grid Width: 10 cells
Total cells: 100
Number of robots: 4

Time taken: .000922 seconds



```
Target surrounded in 0.000922 seconds. Number of moves:
Robot Number 0: 8 moves
Robot Number 1: 15 moves
Robot Number 2: 42 moves
Robot Number 3: 28 moves
7040:Project3 lukeheary$
```

Documented Code

Every function in the program has a comment and description above them, explaining the use of each parameter and telling you what each function returns when it's finished.

Conclusion

Overall, I think the results of my project were very good at demonstrating threads and multicasting. With a larger grid and more robots, it makes sense that the robots were able to find the target in a shorter amount of time, because you have more robots searching, communicating, and a better probability of finding the target sooner. With the smallest grid and less robots, which took the longest amount time, it also makes sense. Less robots makes it harder for the robots to search efficiently, because they have to move more in order to search the entire board.