**Assignment 2**

|  |  |
| --- | --- |
| **Maximum marks** | **100** |
| **Weight** | **15% of the total marks for the course** |
| **Submission mode** | **Electronic, using subversion** |
| **Estimated time** | **20 hours** |
| **Penalty** | **10% per day** |

The assignment will be updated on a regular basis. Clarifications and modifications will be added as and when required. Note that answers should be submitted electronically, using subversion. You will be required to commit one or more files for each question, with specific file names. If you want to commit additional files, for functions that you may have written, that's fine. Just remember to commit the specified files for each question, since that is what we will be looking for as a starting point for your solution.

This is a group assignment and should be completed in groups of 2, unless permission has been obtained from the course convenor to work by yourself. Both people in the group must contribute equally to the assignment. It is strongly recommended that both people in the group work collaboratively, rather than split the workload and work in isolation. Please note that a engn2219 student can only form a group with another engn2219 student, and not with a comp6719 student. Likewise, a comp6719 student can only form a group with another comp6719 student, and not with a engn2219 student.

Note that all work submitted must be entirely your own work. Make certain you use quality information and that you carefully reference all the material that you use. It is unacceptable to cut and paste another author's work and pass it off as your own. Anyone found doing this, from whatever source, will get a zero for the assignment. Any material that you wish to quote should have the source clearly referenced. Read the school's policy on plagiarism, to be found in the Research School of Computer Science Student Handbook.

It is strongly suggested that you start working on the report right away. You can commit as many times as you want. Only the most recent commit will be assessed. Check that your files work on the student system before committing. Please ensure that you copy your committed files to an appropriate folder in your directory on the student system as well.

Note that the ENGN2219 assignment is for the undergraduate students and the COMP6719 assignment for the postgrad students. The two are not interchangeable. If you find problems with this assignment (e.g., something is not clear, or there are broken links), please get in contact with the course lecturer immediately.

**Task - Mobile Robotic Sensors Deployment Simulation  
First posted: Sun 27 Apr 2014 20:21:00 EST   
Last modified: Fri May 9 16:40:47 EST 2014   
Marks: 100**

Mobile robotic sensors can be used for perimeter detection, in applications such as tracking chemical spills, radiation leaks, oil spills and forest fires. In this assignment, we will look at using mobile robots for perimeter surveillance.

Mobile robotic sensors are to be deployed to monitor the perimeter of a protected area. This could, for example, be a remote town or an area within a nature reserve. The area is rectangular in shape. Each sensor has a sensor range, which is a circle of specified radius, within which it can sense other objects. Each sensor also has a specified maximum speed at which it can move.

The sensors are introduced into the area one after the other at random points within a square whose origin is at the centre of this area. A sensor will not be introduced at a location that is already occupied by another sensor. Each sensor enters the area at a random time with respect to the entry of the previous sensor. When introduced, each sensor is aware of its initial co-ordinates, as well as the length and width of the protected area.

There are other objects present in the area. These can be square, rectangular or circular in shape. While moving, the sensors need to skirt around any such objects. If an object is located on the perimeter of the area, then the sensors need to cover the inner perimeter of such an object.

A sensor should not collide with another sensor or with an object in the area. In such a situation, the sensor (or sensors) will be destroyed. You may assume that you will be provided with enough sensors to cover all of the specified perimeter. A sensor can only pass on information to another sensor, if they are physically located within each others sensor ranges. The sensors can use appropriate strategies to optimize the time taken to complete their deployment.

Your task is to simulate the movement of the sensors from their initial position to the perimeter of the area, such that their combined sensor ranges covers all of the perimeter. The time limit for the simulation, that is, the maximum number of time steps for which the game can run, is specified. The simulation ends when either all of the sensors have arrived at their final positions or the time limit has been exceeded..

The initial configuration is to be read from a file, which has the following information in the specified order:

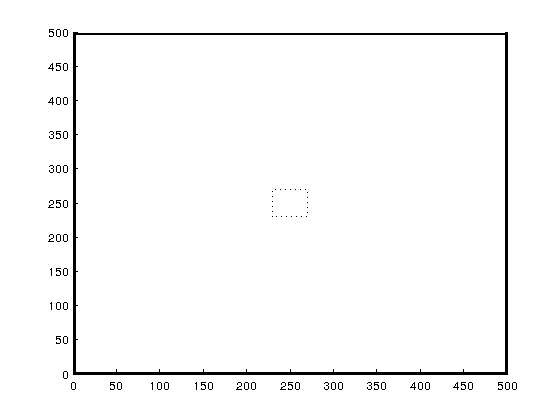
* Length and width of the rectangular area (in kilometres).
* Number of objects in the area (k, a positive integer).
* For each object, object type (square/rectangle/circle) and size (side for square, length and width for rectangle and radius for circle, in metres) and co-ordinates of the centre of the object, all in metres. Eg.

square 10 25 30

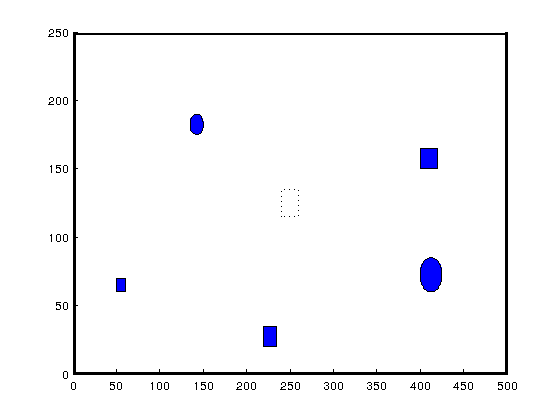
* Side of the entry square (in metres).
* Number of sensors (a positive integer).
* Range of each sensor (radius, in metres).
* Maximum speed of each sensor (in metres/sec).
* Time limit for each sensor's entry (number of secs, an integer).
* Time limit for simulation (number of secs, an integer).

Except where stated, all the values are real numbers, with an accuracy to two decimal places. A line starting with a '#' should be treated as a comment and ignored. You should ask the user for the file name of the configuration file. You should read this file and extract the required parameters from the file. Include error checking, so that you report an error if you don't get a sufficient number of input parameters, or you are given incorrect input (eg, negative numbers, where it should be positive). If there are errors, report the errors and exit. Some sample inputs are given below:

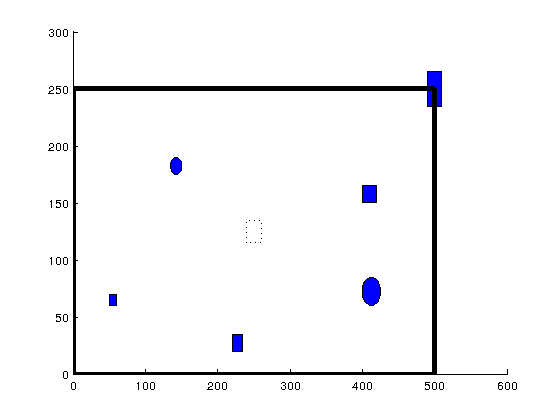
* A [configuration](http://cs.anu.edu.au/student/engn2219/assignments/ass2/engn2219/sample-input-1.txt) with no obstacles:

.

* A [configuration](http://cs.anu.edu.au/student/engn2219/assignments/ass2/engn2219/sample-input-2.txt) with obstacles, all inside the area:

.

* A [configuration](http://cs.anu.edu.au/student/engn2219/assignments/ass2/engn2219/sample-input-3.txt) with obstacles, with one being on the perimeter:

.

In your simulation, you may assume that a single time step is equal to one second. Draw the positions of the sensors at each time step. You can use the draw now function command for this purpose. You can use the pause command to force a pause between plots. Remember to erase the previous positions when you draw the new positions. Explore how this can be done (you can use a number of different approaches and is quite easy to do. For example, create a new plot each time, although this is inefficient).

Draw the perimeter as a thick black line and the objects in the area as filled objects in blue color. Draw each of the sensors with a small filled red circle to denote its position and a red circle around it to show its sensor range. You can use the rectangle function to draw the rectangles and squares (and even the circles), if you wish. Any approach that works is fine.

Design your program keeping in mind functionality, the potential for reuse and efficiency of code. Your program should be in a file called sensors.m. You may use as many functions as required. Functions that you think can be reused must be located in separate files. Document your code as appropriate. Write a short report on your approach to solving the problem and on the design of your program and save this in a html or pdf file with name design.html or design.pdf, respectively.

**Marking Scheme**

The marks will be allocated as follows:

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Description** | **Marks** |
| Your solution and design | A write up of your approach to solving the problem and of your program design as comments in the main program file. Good program construction; appropriate use of data structures and functions; and efficiency of code, including use of builtin operators and functions, where appropriate and error checking. Good program design will involve the creation of appropriate functions. | 25 |
| Documentation | Appropriate documentation. This includes program and function descriptions, author details and appropriate commenting of code. Also include the appropriate time spent, in hours, and your feedback, if any, on the assignment. | 10 |
| Coding Style | This includes use of indentation to make the code easy to read; descriptive variable names; and avoiding the use of global variables. | 5 |
| The simulation | The simulation works correctly. It has appropriate and descriptive title and labels. If your simulation works only in the case where there are no objects in the area, you will get up to 35 marks. 5 marks will be reserved for good strategy. | 60 |