CIS 700 – Programming & Problem Solving
Fall 2013
Project #2: Mosquito

Problem description

The *Culex computerus* mosquito exists in a two dimensional square world and is nocturnal. At night it flies at constant speed (1 meter per second) in a straight line for a short segment (1 meter), and then randomly chooses a new direction for the next segment. Each mosquito moves independently of other mosquitoes, and they are small enough that they do not collide when their paths cross. Mosquitoes are initially distributed randomly within the world. The world also includes obstacles, which are straight walls of zero width. When a mosquito encounters an obstacle or world boundary, it chooses a new random direction away from the obstacle/boundary and starts a new 1 meter segment.

Your job is to corral the mosquitoes using a circular device (a "collector") with a diameter of 1 meter that will capture any mosquitoes that fly over it. Your tool for this job is a set of lights that you can place and move around the world, and turn on and off over time. Mosquitoes are attracted to the lights, and their flying behavior changes.

A mosquito will be attracted to the brightest light. Because all lights have the same intensity, this means they are attracted to the nearest light that is switched on and visible in a straight line from the mosquito. (Obstacles obscure lights if they are between a mosquito and a light.) However, a mosquito is not attracted at all to a dim light -- lights that are more than 20 meters away have no effect on mosquitoes.

When a mosquito is about to choose a new direction, it will choose a direction biased by the nearest light (within 20 meters). The new direction will be within 30 degrees of the direction of the light. For example, suppose the mosquito is at (50,50), that the light is at (50,60), and that there are no obstacles. Then the new direction will be a random angle chosen uniformly from between 60 degrees and 120 degrees using conventional polar coordinates. The mosquito will travel 1 meter in the chosen direction, and then repeat the process of selecting a direction. When a mosquito gets very close to a light, this process may take the mosquito past the light. (We'll assume the light is a point source, and that mosquitoes don't collide with the light.)

The world is assumed to be a Cartesian square with opposing corners at (0,0) and (100,100). You will also know the endpoints of zero or more obstacles (you may assume that the obstacles do not partition the world), and a maximum number *L* of lights you get to use.

Given the location of obstacles and the value of *L*, your program should first specify the location of the collector and the initial location of the lights. The collector and lights cannot overlap an obstacle, another light, or a world boundary.

Then, in each step of the simulation, you are given the location of all the mosquitoes and can move each light up to one unit in any direction. The goal is to attract the mosquitoes to the lights, and then bring them to the collector.

The score of your solution will either be the number of mosquitoes caught in a set number of rounds (simulated time), or the time taken to collect a certain percentage of the mosquitoes; this will be discussed later in today's class. We will do several runs and average them to compensate for variability due to the randomness.

Setting up the simulator

Download version 1.0 of the simulator as an Eclipse project from Blackboard. To set it up:

1. Create a new Eclipse project called “Mosquito”.
2. Copy all of the content from the tar file that you downloaded into the workspace directory and refresh the Eclipse project.
3. Add the six .jar files to the project's Java Build Path. After a while, the project errors should disappear.
4. Create a "Java Application" Run Configuration for the project:
    - main class = mosquito.sim.GameEngine
    - program arguments = mosquito.xml gui

When you run the project, you should see the simulator UI launch.

- You can configure the Number of Lights, Percentage of Mosquitoes to Capture, Maximum Number of Rounds, and Number of Mosquitoes on the right-hand side.
- At first, the Player should be "g0 Random Player" and Board should be "boards/Blank.xml"
- Click "Begin New Game" to set up the Lights and Collector.
- Click "Step" to move ahead one unit of time.
- Click "Play" to move continuously (configure the Delay with the slider at the bottom right)
- Click "Pause" to temporarily pause the game.
- Click "Resign" to quit.

Creating your own player

Create a class called mosquito.gX.GroupXPlayer where X is your group number. This class must:

- extend mosquito.sim.Player
- implement the getName, startNewGame, getLights, updateLights, and getCollector methods

See mosquito.g0.RandomPlayer for an example.

The simulator calls startNewGame at the start of a new game and passes it a Set containing the locations of the obstacles in the world, as well as the maximum number of lights you may use. The return value is a Set of Line2D objects that will be drawn on the board. Note that these lines have no effect on the game and do not change during the game; they are only there so that you can get a visualization of your player's behavior.

The simulator then calls getLights and getCollector to allow you to set the initial placement of the lights and the collector (which you cannot move after it is placed). Then, on each step of the simulation, updateLights is called. You can move each light up, down, left, or right, or to a specific location; however, if the light moves more than one unit, the simulation will end.

To add your Player to the application, add the name of your class to mosquito.xml in the "mosquito.classes" entry (if you have more than one, the class names should be separated by whitespace) .

Note: To do logging/debugging, do *not* use System.out.println or System.err.println. Rather, create a Logger instance (see RandomPlayer for an example) and then call its trace, debug, info, warn, or error method and pass the String to appear in the console.