CIS 1.5 Fall 2009 Homework V

Overview

- This is the fifth homework assignment for CIS 1.5.
- The entire assignment will be worth 15 points, that is 15% of your semester grade.
- **It is due by midnight on Friday December 4th** and must be submitted by email (as below).
- **Follow these emailing instructions:**
  1. Create a mail message addressed to *parsons@sci.brooklyn.cuny.edu* with the subject line **CIS 1.5 HW5**.
  2. Write your name, that is the name under which you registered for the course, in the email (when I get an email from `deathmetal@aol.com` or `pinkprincess@yahoo.com`, I can usually guess whose program it is, but that is not as good as *knowing* whose program it is).
  3. Attach ONLY the **.cpp** files as I tell you below.
  4. Use a **zip** utility to bundle all your files together and send them as ONE attachment to the email.
     on a PC: use **WinZip**
     on a Mac: use **File - Create Archive...**
     on Linux: use **zip**
  5. Failure to follow these instructions will result in points being taken away from your grade. The number of points will be in proportion to the extent to which you did not follow instructions... (which can make it a lot harder for me to grade your work — grrrr!)

Description

For this project, you will write another robot simulator, one that is a bit more sophisticated than the previous one we looked at.

The simulator will include three robots, two *pusuers* and one *evader*. The part that controls the *evader* is rather similar to the Roomba simulator we looked at earlier in the semester — the user controls a robot and drives it around the simulated world.

The pursers are a new feature. They are two robots that chase down the evader, at each turn moving to try and get closer to the evader.

Start from the code in the file `evader.cpp` which is on the course website, on the page for Unit V. STart by compiling and running this code.

1. Declare two arrays `xPursuer` and `yPursuer` each of which will hold two integers. `xPursuer` will hold the x coordinates of the two Pursuer robots, and `yPursuer` will hold the y coordinates.
   (1 point)

2. Initialise the x and y coordinates of the two pursuers randomly. You need to make sure that these values aren't the same as the the location of the evader (but the two pursuers can have the same location as each other).
   (1 point)

3. Modify the code for `displayPosition` so that it prints a `P` in the location of each pursuer.
   (1 point)

4. Write a function `whichDirection` which chooses the direction that the pursuer moves in. The function should take the x and y coordinate of a pursuer and the evader and return a `char` which indicates the direction (`n`, `s`, `e` or `w`) that the pursuer should move in. It should move towards the evader.

For example, if the pursuer has a larger y coordinate than the evader, the pursuer should move south (`s`).

The function will be more interesting if it has a random element (for example sometimes it makes the pursuer move in the `x` direction and sometimes it makes the pursuer move in the `y` direction, but this is optional).
(2 points)

5. Write a function `movePursuer` which moves an individual pursuer. The function should take as arguments a character and the `x` and `y` coordinates of the pursuer. These last two arguments should be reference parameters.

   You might want to base your function on `moveEvader`.

   Use `movePursuer` and `whichDirection` to make the pursuers move.
   (2 points)

6. Write a function `gotcha` which detects if one of the pursuers is at the same location as the evader. The function should take as arguments the locations of one pursuer and the evader and return a `bool`.

   If the pursuer has caught the evader, the function should print a message.

   Use `gotcha` to end the program when a pursuer catches the evader.
   (2 points)

7. Count the number of moves that the evader makes.
   (1 point)

8. The user's score is the number of moves that the evader manages to stay away from the pursuers. Print the score out after every move that the evader makes.
   (1 point)

9. Add another option `j` to the list of moves that the user is offered.

   If the user selects `j`, then the evader jumps four grid squares in a random direction. (The user can use jump in a last ditch attempt to avoid the pursuers.)

   A jump reduces the user's score by 5 points.
   (3 points)

10. Make the program end if the user's score becomes less than zero.
    (1 point)

Submission

- You will be submitting one file: **evader.cpp**
- Make sure that you have a comment at the top of the file that contains the name of the file, your name, "CIS 1.5 HW5" and the submission date (December 4th, 2009).
- The subject line of your email should say: CIS 1.5 HW5
- The body of your email should contain your name.