ECE 8540 Analysis of Tracking Systems

Lab 8 – Simulated behavior-based robot

In this lab, each student is to write a piece of C-code to simulate a behavior-based robot. The robot will operate in the "Emerge" simulated world demonstrated in class. The code for this world can be downloaded at the link given at the course web site.

You need to make changes to the existing code at four places. First, in robots.c, you must add a new function describing your robot. For example:

```
                /* This robot moves towards food, ignoring sharks */

void GreedyRobot(int FoodClosestDistance, /* input - closest food in pixels */
                 int FoodClosestAngle,/* input - angle in degrees towards closest food */
                 int RobotClosestDistance,      /* input - closest other robot, in pixels */
                 int RobotClosestAngle, /* input - angle in degrees towards closest robot */
                 int SharkClosestDistance,      /* input - closest shark in pixels */
                 int SharkClosestAngle, /* input - angle in degrees towards closest shark */
                 int CurrentRobotEnergy, /* input - this robot's current energy (50 - 255) */
                 int *RobotMoveAngle,           /* output - angle in degrees to move */
                 int *RobotExpendEnergy)        /* output - energy to expend in motion */
                                                /* (cannot exceed Current-50) */
{
(*RobotMoveAngle)=FoodClosestAngle;
(*RobotExpendEnergy)=30;
}
```

Your function will use the exact same prototype (list of variables). Simply change the name of the function. Creative and fun names are appreciated.

Second, in globals.h, you will need to add your function to the list of prototypes. For example:

```
void GreedyRobot(int,int,int,int,int,int,int,int *,int *);
```

Again, only the name will be different. Everything else is the same.

Third, in main.c, you can adjust the number of robots, sharks and food in the simulation.

```
TotalFood=30;
TotalSharks=3;
TotalRobots=3;
```

Finally, in emerge.c, you will need to add your function to the list that is called during an iteration. For example:

```
        /**********************************/
        /* ADD YOUR ROBOT FUNCTION(S) HERE */
        /**********************************/

case 3:         /* this is the call to your robot function */
        Megatron(FoodClosestDistance,FoodClosestAngle,
                                RobotClosestDistance,RobotClosestAngle,
                                SharkClosestDistance,SharkClosestAngle,
                                RobotEnergy[i],
                                &RobotMoveAngle,&RobotMoveEnergy);
        break;
```

This lab will be done over two days. Each day, the robots of every student will compete against each other in a set of challenges to determine the survival of the fittest. The challenges will include small groups in a triple-elimination format. We will also study the effect of changing the simulation parameters (food, sharks).

During the second day, each robot must be enhanced (rewritten) by the student to include the use of memory. This will necessitate the use of C static variables to remember information between function calls. This idea will be further explained and discussed at the end of day 1.

The lab due dates are the days of the simulations and are given at the class web site. For both days, you must email me your C-code at least 30 minutes prior to class. Only email me your function (including its prototype/header). Do not email me the rest of the emerge program as I already have that code, obviously. I will recompile everyone's code for the simulation wars.

You must submit a report to Canvas. This report is due by midnight of the due date. The report should briefly describe your design strategy for your robot for both days, and should include your code highlighting which parts execute different strategies.

We will discuss any questions in class or via email.