Lecture 11:
Simulated Tag and Robot Tag prep

Professor Carr Everbach

Course web page:
http://www.swarthmore.edu/NatSci/ceverba1/Class/e5/E5Index.html
Remember...

- Tuesday (today) and Wednesday: Wizards available (Hicks 213) from 7:00-10:00 p.m. for robot tag prep.

- Videos of robot races posted on E5 website. Please add a link to the video of your group’s ‘bot on your website with a description of what happened and what you learned.

- Robot tag in class three weeks from today in class. Only one lab after Thanksgiving to get everything working.

- Today: Matt Zucker, Computer Engineering (& robotics)
simtag: experimenting with strategy

The simulated tag routines on Moodle are designed to help you work on strategy without worrying about robot communications issues.

%program simtag_init
% sets up the variables and robot board for simulating E5 robot Tag.
clear
clf
numbots = input('Enter number of robots to play simulated Tag (including yours): ');
mybot = input('Enter robot number of your robot: ');
itbot = input('Enter robot number of "it" robot: ');
for index = 1:numbots
    x(index) = rand(); % random starting position for each robot
    y(index) = rand();
    vx(index) = rand()*0.03; % random starting x-velocity
    vy(index) = rand()*0.03; % random starting y-velocity
    scell = num2str(index); % convert robot numbers to strings
    robotnum(index) = {scell}; % save string in cell array for plot labeling
end
plotbots; % plot the robots
% program plotbots.m
% plots a graph of the simulated robot Tag
% written by E. Carr Everbach, 3 November 2011
quiver(x,y,vx,vy,'ko') % plot all robots
hold on
quiver(x(mybot),y(mybot),vx(mybot),vy(mybot),'go','filled'); % mybot in green
quiver(x(itbot),y(itbot),vx(itbot),vy(itbot),'ro','filled') % "it" robot in red
text(x,y-0.04,robotnum); % put robot numbers under circles
axis([0 1 0 1])
axis square
hold off
pause(0.4)
simtag.m:

% program simtag
% sets up and runs simulated robot Tag for E5
% written by E. Carr Everbach, 3 November 2011

simtag_init; % initialize all robots, their starting positions, and starting velocities
N = input('Enter number of steps to play (300): ');

lasttagged = itbot; % variable to prevent tagbacks
count = 1; % counter for tagbacks
tone = sin(2*pi*1:1000); % for tag beeps

for index = 1:N
    Q = onestep(x,y,vx,vy,numbots); % tick forward one step
    x = Q(1,:); y = Q(2,:); vx = Q(3,:); vy = Q(4,:); % unpack updated variables
    plotbots; % plot all the robots in their updated positions
    % itbot behavior:
    dists = sqrt((x(itbot)-x).^2 + (y(itbot) - y).^2); % calculate distances from IT
function Q = onestep(x,y,vx,vy,numbots)
% Q = onestep(x,y,vx,vy)
% where Q = [x; y; vx; vy]
% Updates positions of all robots in x-y plane, reflecting them from the
% boundaries when they cross

for index = 1:numbots % for each robot
  % x step:
  vx(index) = vx(index)*(1 + (rand-0.5)/100); % add some randomness
  x(index) = x(index) + vx(index); % step forward in x
  if x(index)<0
    x(index) = 0; % stop robot at bottom boundary
    vx(index) = -vx(index); % reverse sign of x-velocity
  end
  if x(index)>1
    x(index) = 1; % stop robot at top boundary
    vx(index) = -vx(index); % reverse sign of x-velocity
  end
% y step:
    vy(index) = vy(index)*(1 + (rand-0.5)/100); % add some randomness
    y(index) = y(index) + vy(index); % step forward in y
    if y(index)<0
        y(index) = 0; % stop robot at left boundary
        vy(index) = -vy(index); % reverse sign of x-velocity
    end
    if y(index)>1
        y(index) = 1; % stop robot at right boundary
        vy(index) = -vy(index); % reverse sign of x-velocity
    end
end
Q = [x;y;vx;vy]; % pass out modified variables
end
% program simtag
% sets up and runs simulated robot Tag for E5
% written by E. Carr Everbach, 3 November 2011

simtag_init; % initialize all robots, their starting positions, and starting velocities
N = input('Enter number of steps to play (300): ');

lasttagged = itbot; % variable to prevent tagbacks
count = 1; % counter for tagbacks
tone = sin(2*pi*1:1000); % for tag beeps

for index = 1:N
    Q = onestep(x,y,vx,vy,numbots); % tick forward one step
    x = Q(1,:); y = Q(2,:); vx = Q(3,:); vy = Q(4,:); % unpack updated variables
    plotbots; % plot all the robots in their updated positions
    dists = sqrt((x(itbot)-x).^2 + (y(itbot) - y).^2); % calculate distances from IT
simtag.m continued:

```matlab
[sorted, bots] = sort(dists); % sort the distances to each robot
if (sorted(2) < 0.03 & bots(2) ~= lasttagged) % considered Tagged if closer than 0.03 and not tagback
    itbot = bots(2); % new "it" robot
    count = 1; % reset counter
    fprintf('New IT %d
', itbot); % print number of new itbot
else % simple chase behavior of "it" robot: chase closest one
    itspeed = sqrt(vx(itbot)^2 + vy(itbot)^2); % current speed of "it" robot
    vx(itbot) = itspeed*(x(bots(2)) - x(itbot))/sorted(2); % make itbot's vx toward closest bot
    vy(itbot) = itspeed*(y(bots(2)) - y(itbot))/sorted(2); % make itbot's vy toward closest bot
```

Recall from vector math:

\[
\vec{A} = A\hat{u}_A \quad \text{where unit vector has components}
\]

\[
\hat{u}_A = u_x\hat{i} + u_y\hat{j}
\]

So it follows that

\[
\hat{u}_A = \frac{\vec{A}}{A}
\]
if mybot ~= itbot  % flee behavior
    myspeed = sqrt(vx(mybot)^2 + vy(mybot)^2);  % current speed of my robot
    mydist = sqrt((x(itbot) - x(mybot))^2 + (y(itbot) - y(mybot))^2);  % distance from mybot to itbot
    if(mydist < 0.2)
        vx(mybot) = -0.01*(x(itbot) - x(mybot))/mydist;  % aim away from itbot
        vy(mybot) = -0.01*(y(itbot) - y(mybot))/mydist;
    end
    else
        disp('My Robot is IT!')
disp('My Robot is IT!')
%soundsc(tone); % play Tag tone
dists = sqrt((x(mybot)-x).^2 + (y(mybot) - y).^2); % calculate distances from itbot
[sorted, bots] = sort(dists); % sort the distances to each robot
mymybot = sqrt(vx(itbot)^2 + vy(itbot)^2); % current speed of my robot
vx(mybot) = myspeed*(x(bots(2)) - x(mybot))/sorted(2); % make mybot's vx toward closest bot
vy(mybot) = myspeed*(y(bots(2)) - y(mybot))/sorted(2); % make mybot's vy toward closest bot
if x(mybot) >= 1
    vx(mybot) = -0.1;
end
if y(mybot) >= 1
    vx(mybot) = -0.1;
end
if x(mybot) <= 0
    vy(mybot) = 0.1;
end
if y(mybot) <= 0
    vy(mybot) = 0.1;
end
function Q = getcoords
% usage:  Q = getcoords where Q = [robot; xposition; yposition]
% returns scaled coordinates of all robots from “eye in the sky”
% Uses judp.m to read position of ARToolKitPlus tracking of robots in E5
msg = judp('receive',4951,10000000);
sm = [char(msg),':']  % append terminator
rindex = find(smsg==':'); % find all the colons
pindex = find(smsg=='>'); % find all the >
cindex = find(smsg==','); % find all the commas
for index = 1:length(rindex)-1
    robot(index) = str2num(smsg(rindex(index)+1:pindex(index)-1));
    xposition = str2num(smsg(pindex(index)+1:cindex(index)-1));
    yposition = str2num(smsg(cindex(index)+1:rindex(index+1)-1));
    Q = [robot'; xposition'; yposition'];
end

You’ll need to call “Q = getcoords” each time around your program’s “event loop” to find out where all the robots are currently located. If you save the previous positions, you can subtract them vectorially from the current positions and obtain vx and vy for each ‘bot.