Autonomous agents
Lecture 6, 20160204

Approaches to robot intelligence
Basic robotic brain processes

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Additional information, HP1.1

- In HP1.1 (a) the parentheses do *not* represent function arguments. Thus, the parentheses can safely be removed, so that

$$v_L(t) = v_0 t / t_1$$

… and similarly (but with t_2) for the speed of the other (right) wheel.

About HP1

- The home problem(s) should be solved ***individually***!
- HP1.2 and HP1.3 illustrate the difficulties in generating behaviors for a robot with very limited sensor capability.
- As stated before, you may (of course) **not** make use (in the robotic brain) of any information that a real robot (with the same sensors as our simulated robots) would not have.
- Specifically, you may **not** use the exact pose (position and heading) of the robot.

About HP1

- In HP1.3 (but *not* in HP1.2), you may use the odometric *estimate* of the pose, but you may not recalibrate it (the problem is supposed to illustrate odometric drift – calibration will be considered in a later home problem).
- Even though the absolute pose of the robot will drift quite strongly, the odometer can still be used for a making a local 90-degree turn.

About HP1

Notes:

(1) In order to follow a wall using a single sensor, one may take several readings (to reduced noise effects), then move a bit, take several readings again, and compare.

If constant ⇔ moving parallel to the wall.

If increasing ⇔ moving towards the wall etc.

(2) The IR sensor readings (RayBasedSensors(k).Reading) are *not* linear functions of the distance. In order to estimate the reading at a given distance, place the robot at that distance from a wall, and take several measurements.

Today's learning goals

- After this lecture you should be able to
  - Explain the difference between classical artificial intelligence and behavior-based robotics
  - Describe the basic ideas behind behavior-based robotics in detail
  - Implement simple robotic behaviors in the form of finite-state machines

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Approaches to machine intelligence

- Classical artificial intelligence (AI)
  - 1950s –
  - Focused on high-level (human) reasoning
- Behavior-based robotics (BBR)
  - 1980s-
  - Focused on basic survival-related behaviors

Approaches to machine intelligence

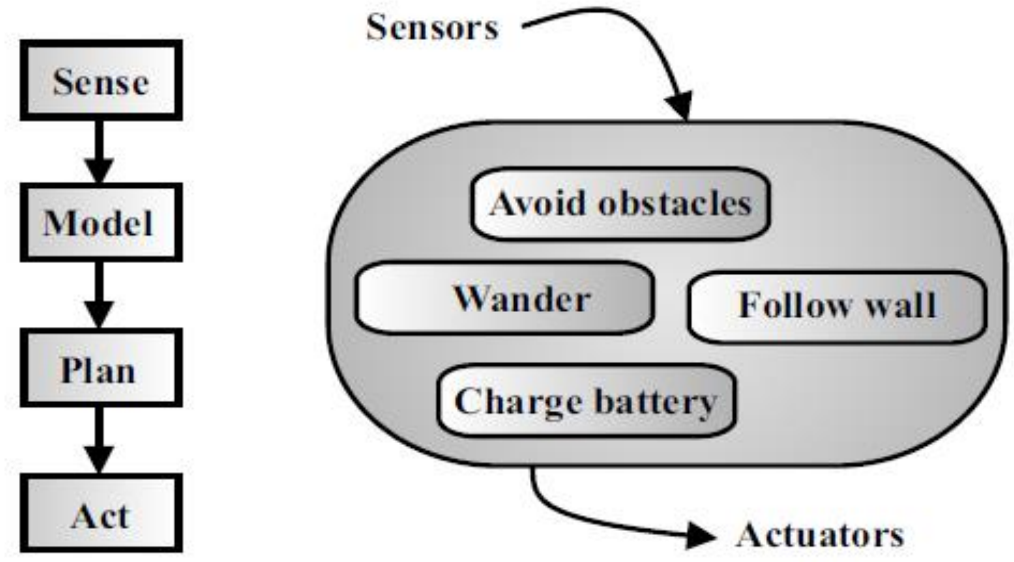


Figure 5.1: *A comparison of the information flow in classical AI (left panel) and in BBR (right panel). For BBR, any number of behaviors may be involved, and the figure only shows an example involving four behaviors.*

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today's learning goals

- After this lecture you should be able to
  - Explain the difference between classical artificial intelligence and behavior-based robotics ✓
  - Describe the basic ideas behind behavior-based robotics in detail
  - Implement simple robotic behaviors in the form of finite-state machines

Behavior-based robotics

- Mostly applied to mobile robots.
- Strongly influenced by (the brains of) relatively simple biological organisms (e.g. Insects).
- Robots are first provided with simple, survival-related behaviors, then more complex behaviors. For a robot, survival entails
  - Avoiding collisions
  - Avoiding to run out of battery energy

Behavior-based robotics

- Reactivity: Quick reactions, without too much deliberation (even though internal states are required for more complex behaviors).
- Basic behaviors form a behavioral repertoire.
- Decision-making system needed to select between behaviors.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today's learning goals

- After this lecture you should be able to
  - Explain the difference between classical artificial intelligence and behavior-based robotics ✓
  - Describe the basic ideas behind behavior-based robotics in detail ✓
  - Implement simple robotic behaviors in the form of finite-state machines

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde
CHALMERS

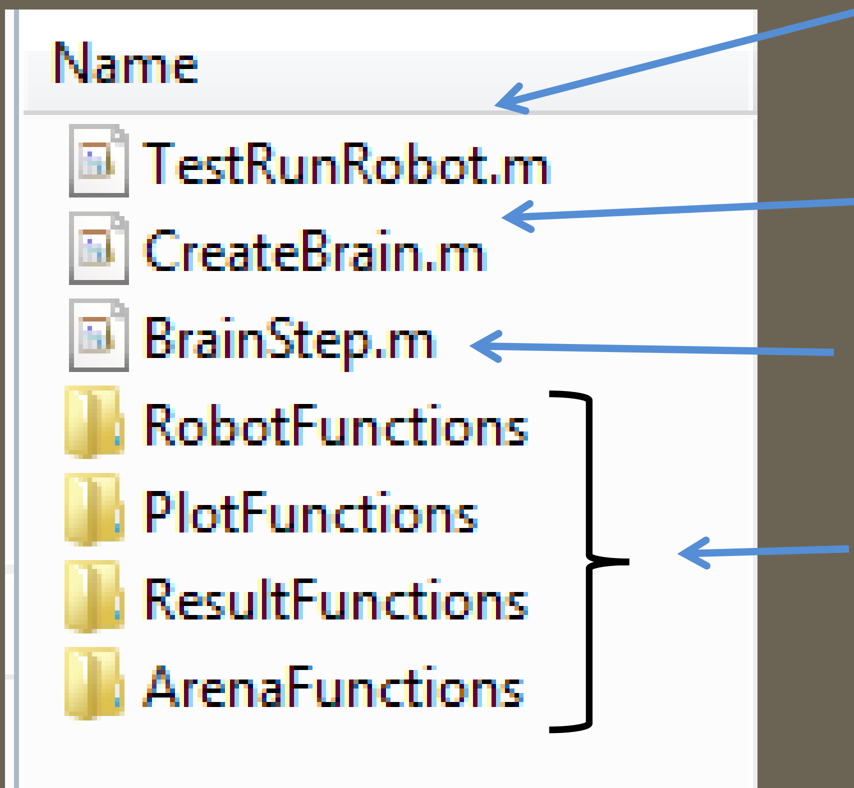
Implementing robot behaviors

- Behavioral architectures
  - Artificial neural networks (ANNs)
  - Finite-state machines (FSMs)
  - Other
- Many architectures have been suggested.
- Some behaviors require a high degree of clarity (as obtained with FSMs), while others are more suitable as black-box units (e.g. ANNs).
- Here we will use FSMs.

ARSim

- The simulator is available on the web page.
- Implement and run the two ARSim examples (code available in Chapter 5)
  - *Wandering*
  - *Navigation*
- Study the various parts of ARSim carefully.

ARSim



Main program

Definition of variables and parameters

Finite-state machine (FSM) (sequence of IF-THEN-ELSE rules)

Normally *not modified!*

ARSim

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% TestRunRobot: a sample program, illustrating the use
% of the ARSim functions (v1.1.9)
%
% (c) Mattias Wahde, 2006, 2007, 2011
%
% Send bug reports to: mattias.wahde@chalmers.se
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Add ARSim to the search path:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if strcmp(computer,'PCWIN')
  path(path,'.\RobotFunctions');
  path(path,'.\ResultFunctions');
  path(path,'.\PlotFunctions');
  path(path,'.\ArenaFunctions');
else
  path(path,'./RobotFunctions');
  path(path,'./ResultFunctions');
  path(path,'./PlotFunctions');
  path(path,'./ArenaFunctions');
end
```

Main program (TestRunRobot.m)

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

ARSim

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate arena:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
arenaSize = [-2 2 -2 2];
arenaObject1 = CreateArenaObject('arenaObject1',[[-1 1]; [-1 0.7]; [-0.7 0.7]; [-0.7 1]]);
arenaObject2 = CreateArenaObject('arenaObject2',[[-2.0 -2.0]; [-1.8 -2.0]; [-1.8 2.0]; [-2.0 2.0]]);
arenaObject3 = CreateArenaObject('arenaObject3',[[-1.8 -2.0]; [1.8 -2.0]; [1.8 -1.8]; [-1.8 -1.8]]);
arenaObject4 = CreateArenaObject('arenaObject4',[[2.0 -2.0]; [1.8 -2.0]; [1.8 2.0]; [2.0 2.0]]);
arenaObject5 = CreateArenaObject('arenaObject5',[[-1.8 2.0]; [1.8 2.0]; [1.8 1.8]; [-1.8 1.8]]);
arenaObject6 = CreateArenaObject('arenaObject6',[[1 -1]; [1 -0.7]; [0.7 -0.7]; [0.7 -1]]);
testArena = CreateArena('arena',arenaSize,[arenaObject1; arenaObject2; arenaObject3;...
                                           arenaObject4; arenaObject5; arenaObject6]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate robot:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%
% Brain:
%%%%%%%%%%%%%%%%%%
brain = CreateBrain;
```

Written by the user.
Problem-specific

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

ARSim

```
%%%%%%%%%%%%%%%%%%
% IR Sensors:
%%%%%%%%%%%%%%%%%%
sensor1RelativeAngle = 0.7854;
sensor2RelativeAngle = -0.7854;
size = 0.0500;
numberOfRays = 3;
openingAngle = 0.5000;
range = 1.0000;
c1 = 0.0300;
c2 = 0.1000;
sigma = 0.0100;
sensor1 = CreateIRSensor('sensor1',sensor1RelativeAngle,size,...
                          numberOfRays,openingAngle,range,c1,c2,sigma);
sensor2 = CreateIRSensor('sensor2',sensor2RelativeAngle,size,...
                          numberOfRays,openingAngle,range,c1,c2,sigma);

%%%%%%%%%%%%%%%%%%%%
% Odometer:
%%%%%%%%%%%%%%%%%%%%
sigma = 0.0001;      % Measures the odometric drift
odometer = CreateOdometer('odometer',sigma);

%%%%%%%%%%%%%%%%%%%%
% Compass:
%%%%%%%%%%%%%%%%%%%%
sigma = 0.0500;
compass = CreateCompass('compass',sigma);
```

Noise level

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

ARSim

```
%%%%%%%%%%%%%%%%%%%%
% Motors:
%%%%%%%%%%%%%%%%%%%%
leftMotor = CreateMotor('leftMotor');
rightMotor = CreateMotor('rightMotor');
mass = 3.0000;
momentOfInertia = 0.2000;
radius = 0.2000;
wheelRadius = 0.1000;

%% Robot with IR sensors:
%  testRobot = CreateRobot('TestRobot',mass,momentOfInertia,radius,wheelRadius, ...
%                      [sensor1 sensor2],[leftMotor rightMotor],[],[],brain);

%% Robot with IR sensors and odometer:
%  testRobot = CreateRobot('TestRobot',mass,momentOfInertia,radius,wheelRadius, ...
%                       [sensor1 sensor2],[leftMotor rightMotor],odometer,[],brain);

%% Robot with IR sensors, odometer, and compass:
   testRobot = CreateRobot('TestRobot',mass,momentOfInertia,radius,wheelRadius, ...
                     [sensor1 sensor2],[leftMotor rightMotor],odometer,compass,brain);
```

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

CHALMERS

ARSim

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Miscellaneous simulation parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
showPlot = true;
plotStep = 10;
recordResults = true;
testRobot.ShowSensorRays = false;
testRobot.ShowOdometricGhost = false;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

position = [-0.2000 0.0000];
heading = 0.0000;
velocity = [0.0000 0.0000];
angularSpeed = 0.0000;

testRobot = SetPositionAndVelocity(testRobot, position, heading, velocity, angularSpeed);
if (~isempty(testRobot.Odometer))
 testRobot.Odometer = CalibrateOdometer(testRobot);
end

if (showPlot)
 plotHandle = InitializePlot(testRobot, testArena);
end
if (recordResults)
 motionResults = InitializeMotionResults(testRobot);
end

maxSteps = 1000;
dt = 0.01;
time = 0.0;
collided = false;
```

Try changing both parameters to *true*

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

ARSim

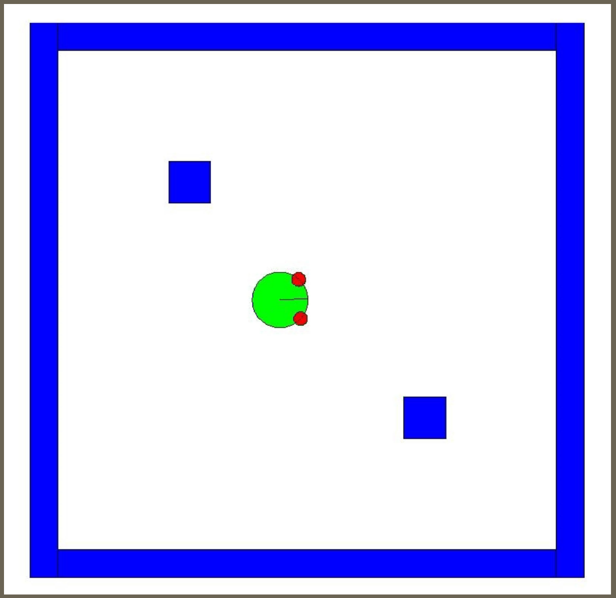
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main loop:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

i=1;
while ((i < maxSteps) & (~collided))
   testRobot.RayBasedSensors = GetRayBasedSensorReadings(testRobot,testArena);
   if (~isempty(testRobot.Odometer))
    testRobot.Odometer = GetOdometerReading(testRobot,dt);
   end
   if (~isempty(testRobot.Compass))
    testRobot.Compass = GetCompassReading(testRobot,dt);
   end
   testRobot.Brain = BrainStep(testRobot, time);
   testRobot = MoveRobot(testRobot,dt);

   time = i*dt;
   if (recordResults)
    motionResults = AddMotionResults(motionResults,time,testRobot);
   end

   i = i + 1;

   collided = CheckForCollisions(testArena, testRobot);

   if ((mod(i,plotStep)==0) && showPlot)
    ShowRobot(plotHandle,testRobot);
   end
end
```

Written by the user.
Problem-specific!

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Example 1: Wandering

- Simple behavior: Aimless exploration of an arena
- Implemented as a finite-state machine



Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Example 1: Wandering

Code listing 5.1: *The CreateBrain function for the wandering example.*

```
function b = CreateBrain;

%% Variables
leftMotorSignal = 0;
rightMotorSignal = 0;
currentState = 0;

%% Parameters:
forwardMotorSignal = 0.5;
turnMotorSignal = 0.7;
turnProbability = 0.01;
stopTurnProbability = 0.03;
leftTurnProbability = 0.50;


b = struct('LeftMotorSignal',leftMotorSignal,...
           'RightMotorSignal',rightMotorSignal,...
           'CurrentState',currentState,...
           'ForwardMotorSignal',forwardMotorSignal,...
           'TurnMotorSignal',turnMotorSignal,...
           'TurnProbability',turnProbability,...
           'StopTurnProbability',stopTurnProbability,...
           'LeftTurnProbability',leftTurnProbability);
```

Set all numerical parameters (related to the robotic brain) here. Do ***NOT*** set parameters at many different places (e.g. in BrainStep).

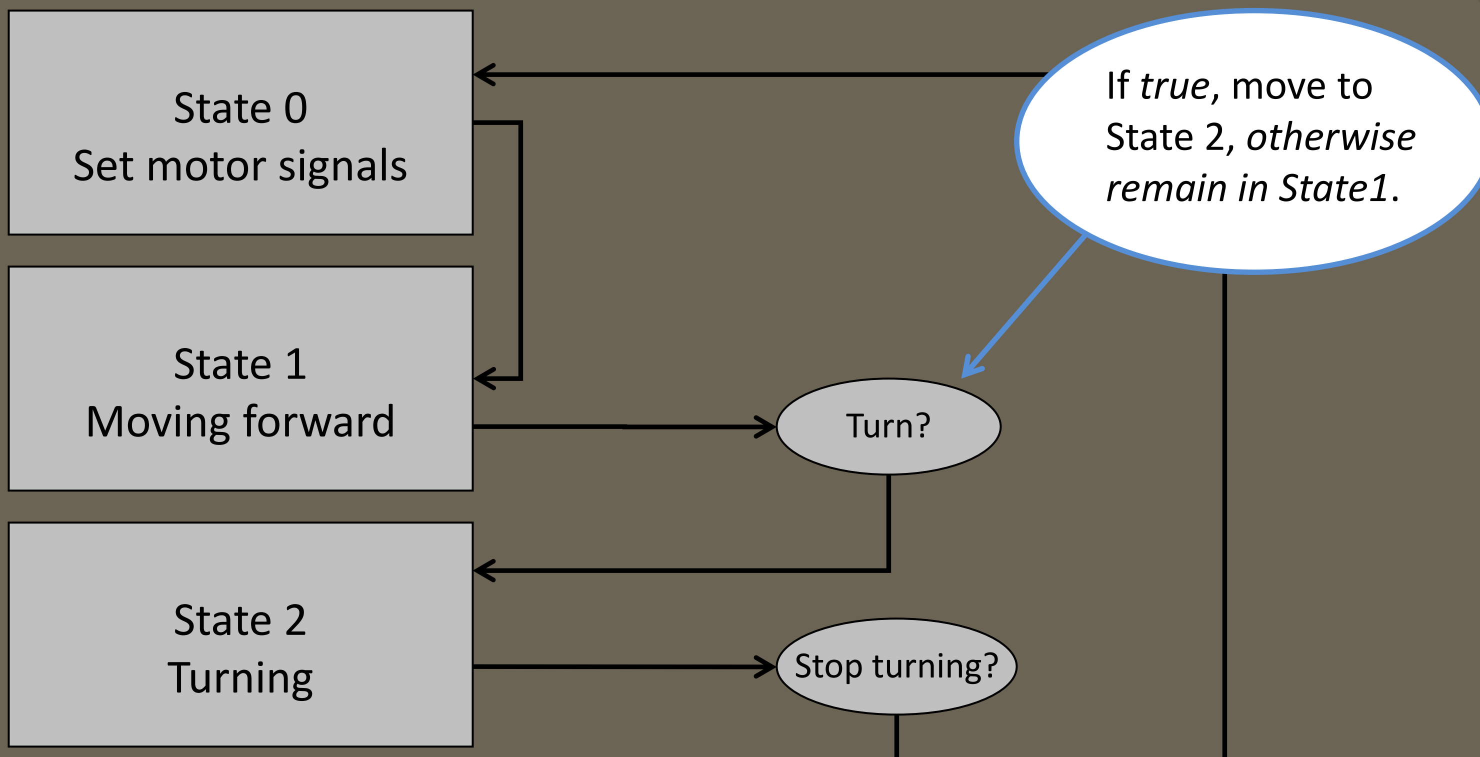
Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Example 1: Wandering

Code listing 5.2: *The BrainStep function for the* wandering *example.*

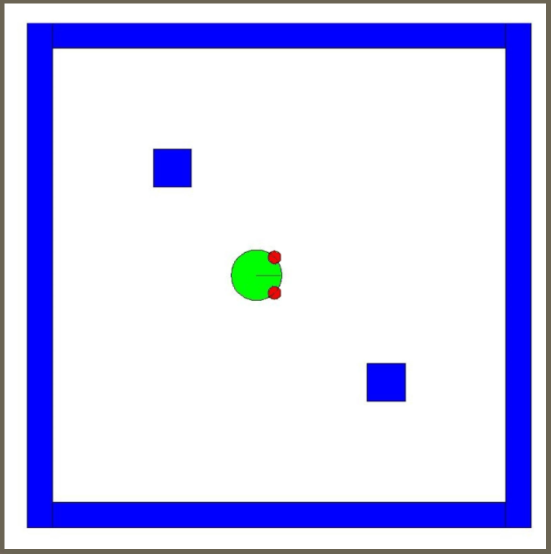
```
function b = BrainStep(robot, time);

b = robot.Brain;

%%%%%%%%%%%%%%%% FSM: %%%%%%%%%%%%%%%%%%%%
if (b.CurrentState == 0) % Forward motion
 b.LeftMotorSignal = b.ForwardMotorSignal;
 b.RightMotorSignal = b.ForwardMotorSignal;
 b.CurrentState = 1;
elseif (b.CurrentState == 1) % Time to turn?
 r = rand;
 if (r < b.TurnProbability)
  s = rand;
  if (s < b.LeftTurnProbability)
   b.LeftMotorSignal  =  b.TurnMotorSignal;
   b.RightMotorSignal = -b.TurnMotorSignal;
  else
   b.LeftMotorSignal  = -b.TurnMotorSignal;
   b.RightMotorSignal =  b.TurnMotorSignal;
  end
  b.CurrentState = 2;
 end
elseif (b.CurrentState == 2) % Time to stop turning?
 r = rand;
 if (r < b.StopTurnProbability)
  b.CurrentState = 0;
 end
end
```

Example 1: Wandering



Example 2: Navigation

- Very simple part of (purposeful) navigation:
  - move a given distance, then stop.
- Can easily be generalized (waypoint sequences, turning...)



Example 2: Navigation

Code listing 5.3: *The CreateBrain function for the* navigation *example.*

```
function b = CreateBrain;

%% Variables:

leftMotorSignal = 0;
rightMotorSignal = 0;
currentState = 0;
initialPositionX = 0; % Arbitrary value here - set in state 0.
initialPositionY = 0; % Arbitrary value here - set in state 0.

%% Parameters:
desiredMovementDistance = 1;
motorSignalConstant = 0.90;
atDestinationThreshold = 0.02;


b = struct('LeftMotorSignal',leftMotorSignal,...
           'RightMotorSignal',rightMotorSignal,...
           'CurrentState',currentState,...
           'InitialPositionX',initialPositionX,...
           'InitialPositionY',initialPositionY,...
           'DesiredMovementDistance',desiredMovementDistance,...
           'MotorSignalConstant',motorSignalConstant,...
           'AtDestinationThreshold',atDestinationThreshold);
```

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Example 2: Navigation

```
function b = BrainStep(robot, time);

b = robot.Brain;

if (b.CurrentState ~= 0)
 deltaX = robot.Odometer.EstimatedPosition(1) - b.InitialPositionX;
 deltaY = robot.Odometer.EstimatedPosition(2) - b.InitialPositionY;
 distanceTravelled = sqrt(deltaX*deltaX + deltaY*deltaY);
end

%%%%%%%%%%%%%%%% FSM: %%%%%%%%%%%%%%%%%%%%%
if (b.CurrentState == 0) % Initialization
 b.InitialPositionX = robot.Odometer.EstimatedPosition(1);
 b.InitialPositionY = robot.Odometer.EstimatedPosition(2);
 b.CurrentState = 1;
elseif (b.CurrentState == 1) % Adaptive motion
 motorSignal = b.MotorSignalConstant*(b.DesiredMovementDistance-distanceTravelled)/...
                              b.DesiredMovementDistance;
 b.LeftMotorSignal = motorSignal;
 b.RightMotorSignal = motorSignal;
 if (abs(b.DesiredMovementDistance - distanceTravelled) < ...
        b.AtDestinationThreshold*b.DesiredMovementDistance)
  b.CurrentState = 2;
  % 'At destination'  % Output for debugging
 end
elseif (b.CurrentState == 2) % At destination
 b.LeftMotorSignal = 0;
 b.RightMotorSignal = 0;
end
```

Computed in every state, except state 0.

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today's learning goals

- After this lecture you should be able to
  - Explain the difference between classical artificial intelligence and behavior-based robotics ✓
  - Describe the basic ideas behind behavior-based robotics in detail ✓
  - Implement simple robotic behaviors in the form of finite-state machines ✓