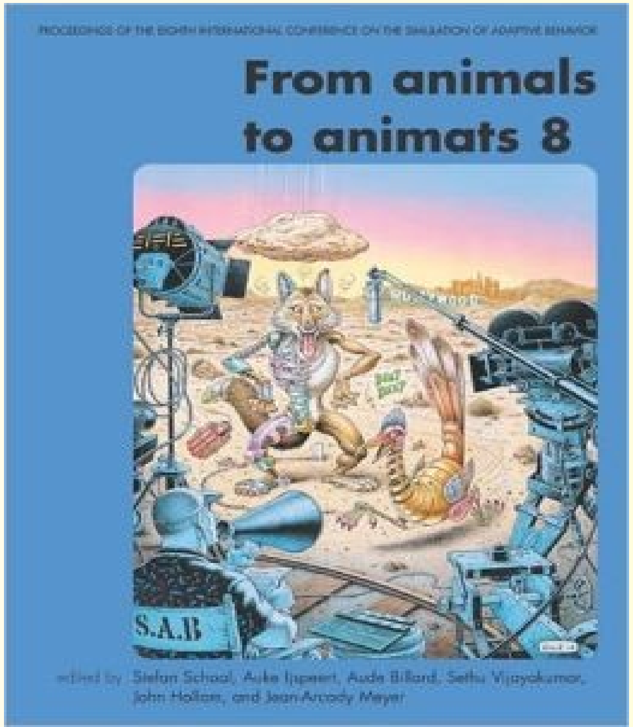
Reactive Control

Robotics

Reactive systems.. a Beginning

- Once upon a time, there were deliberative systems
- and they...
- then some young turks looked upon the flaws and said...
- so reactive systems were born.

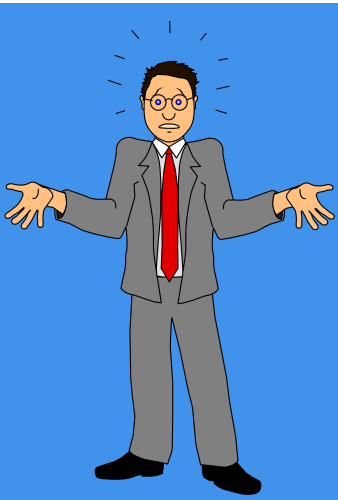
Reactive Systems Influences

- Heavily influenced by ideas from biology.
  - Animal behaviors are simple
  - We can model them and create effective mechanical animals
  - 
PROCEEDINGS OF THE EIGHTH INTERNATIONAL CONFERENCE ON THE SIMULATION OF ADAPTIVE BEHAVIOR
From animals to animats 8
S.A.B
edited by Stefan Schaal, Auke Ijspeert, Aude Billard, Sethu Vijayakumar, John Hallam, and Jean-Arcady Meyer

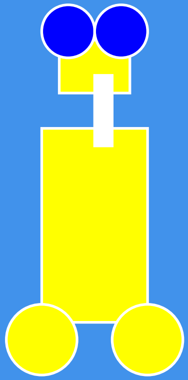
Some of those influences

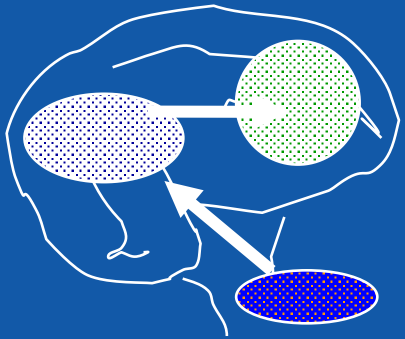
- A Quick look at some of those influences follows
  - Originally from R Murphy: AI Robotics.

Marr’s Computational Theory

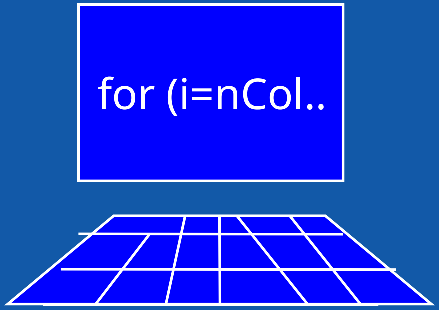


Level 1:
What is the *phenomena*
we’re trying to represent?

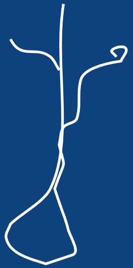




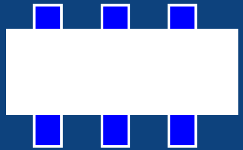
Level 2:
How it be represented as
a *process with inputs/outputs*?



for (i=nCol..



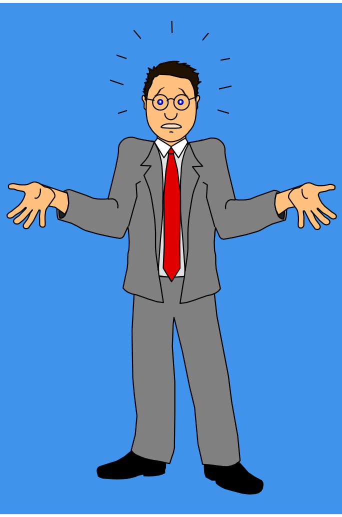
Level 3:
How is it *implemented*?



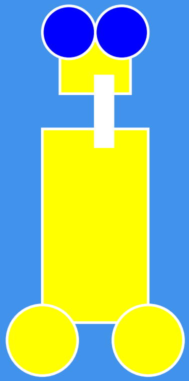
Level 1: Existence Proof

Goal: how to make line drawings of objects?

people can do this by age 10, computers should



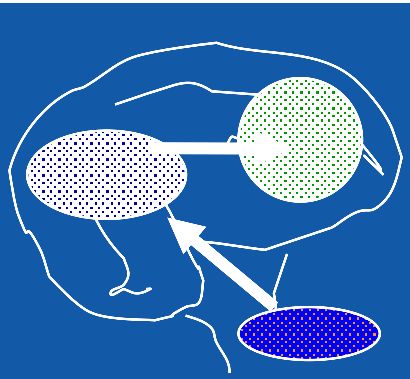
Level 1:
What is the *phenomena*
we’re trying to represent?



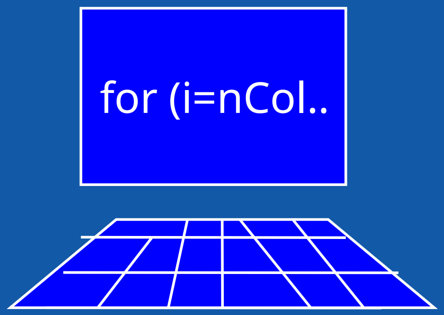
Level 2: Inputs, Outputs, Transforms

light [redacted] drawing

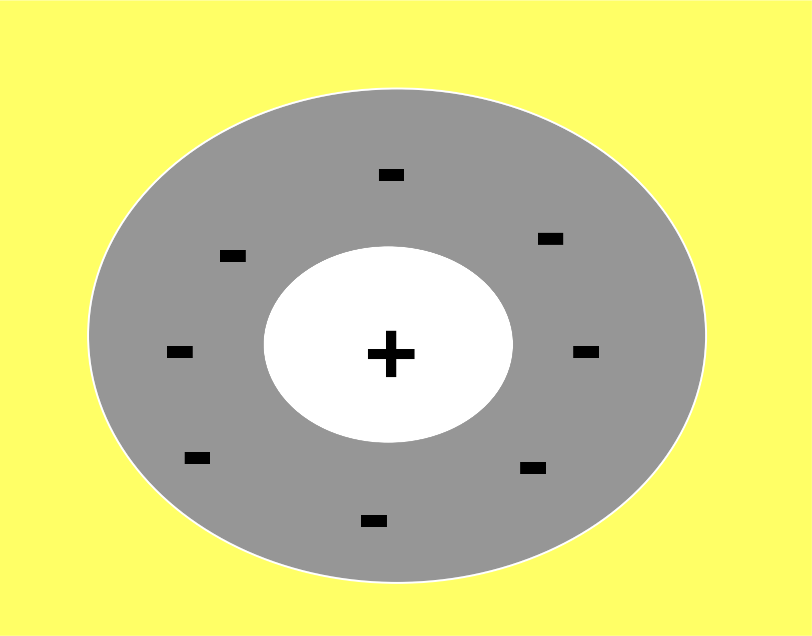
light retina (gradient) lines (edges) [redacted] drawing



Level 2:
How it be represented as
a *process with inputs/outputs*?



Level 3: Implementation

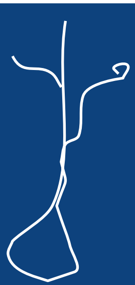


Center Surround Cell
in retinal ganglion

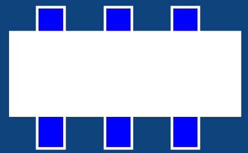
-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1

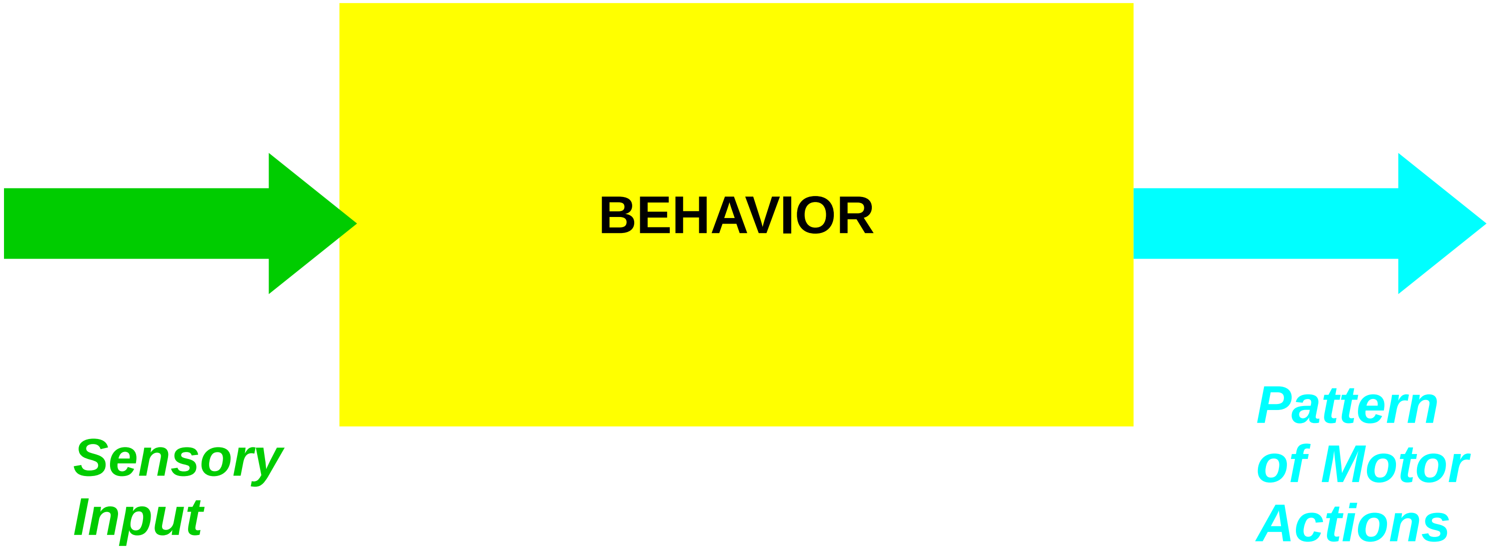
Sobel Edge Detector
in computer vision



Level 3:
How is it *implemented*?



Behavior Definition (graphical)



Types of Behaviors

- **Reflexive**

  stimulus-response, often abbreviated S-R

- **Reactive**

  learned or “muscle memory”

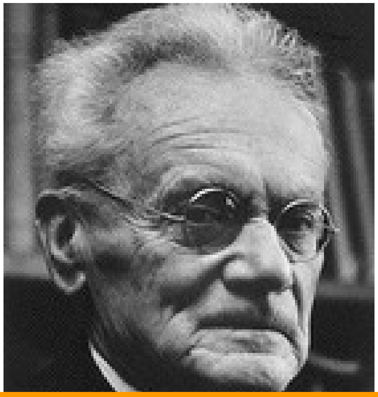
- **Conscious**

  deliberately stringing together

WARNING Overloaded terms:
Roboticists often use “reactive behavior” to mean purely reflexive,
And refer to reactive behaviors as “skills”

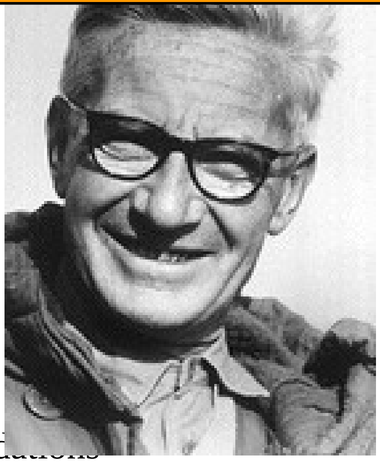
Ethology: Coordination and Control of Behaviors

Nobel 1973 in physiology or medicine



INNATE RELEASING MECHANISMS

Tinbergen



www.nobel.se

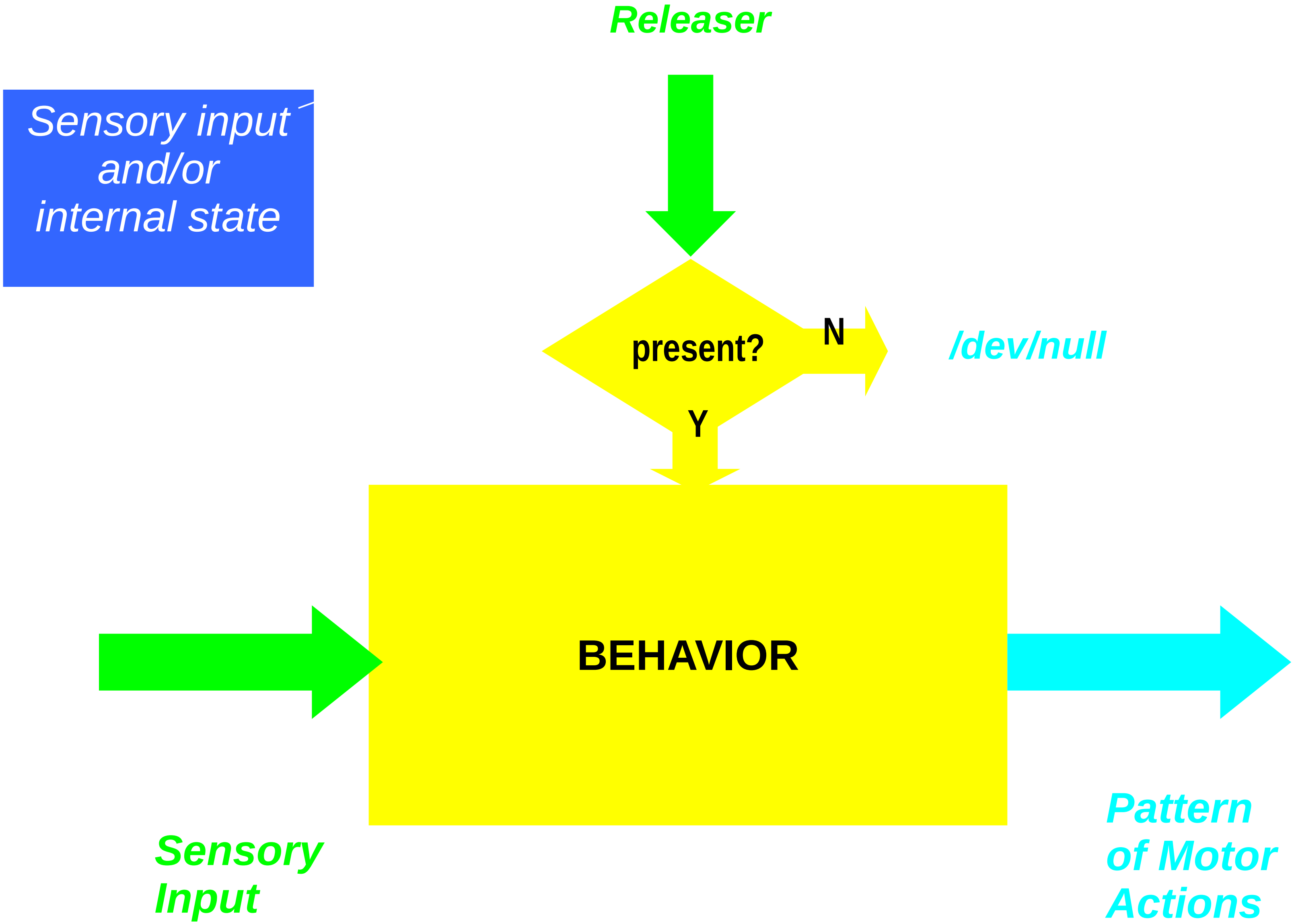
Arctic Terns

Arctic terns live in Arctic (black, white, gray environment, some grass) but adults have a red spot on beak

When hungry, baby pecks at parent’s beak, who regurgitates food for baby to eat

How does it know its parent?

Innate Releasing Mechanisms



Example: Hide Behavior

- Programmed in C++, << 100 LOC
- shows

  taxis *(oriented relative to light, wall, niche)*

  fixed action pattern *(persisted after light was off)*

  reflexive *(stimulus, response)*

  impliciting sequencing

  use of internal state

Example: Cockroach Hide

- light goes on, the cockroach turns and runs
- when it gets to a wall, it follows it
- when it finds a hiding place (thigmotrophic), goes in and faces outward
- waits until not scared, then comes out
- *even if the lights are turned back off earlier*

Reflexive Behaviors S-R

- light goes on, the cockroach turns and runs
- when it gets to a wall, it follows it
- when it finds a hiding place (thigmotrophic), goes in and faces outward
- waits until not scared, then comes out
- *even if the lights are turned back off earlier*

Fixed Pattern Actions

- light goes on, the cockroach turns and runs
- when it gets to a wall, it follows it
- when it finds a hiding place (thigmotrophic), goes in and faces outward
- waits until not scared, then comes out
- *even if the lights are turned back off earlier*

Exhibits Taxis

- light goes on, the cockroach turns and runs

to light

- when it gets to a wall, it follows it

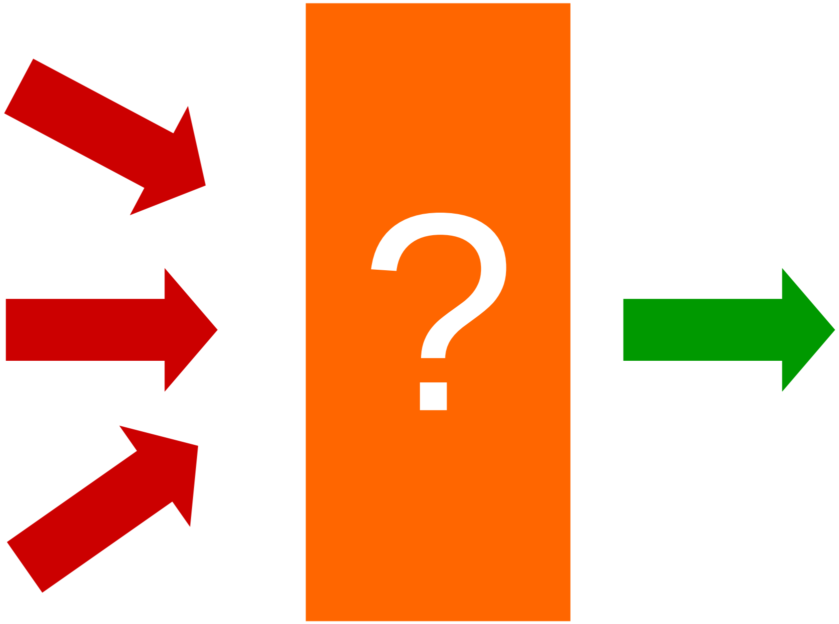
to wall

- when it finds a hiding place (thigmotrophic), goes in and faces outward

to niche

- waits until not scared, then comes out
- *even if the lights are turned back off earlier*

What happens when there’s a conflict from concurrent behaviors?



- Equilbrium
  Feeding squirrels- feed, flee: hesitate in-between
- Dominance
  Sleepy, hungry: either sleep or eat
- Cancellation
  Sticklebacks defend, attack: build a nest

Gibson’s Ecological Approach

- *Acting* and *sensing* co-evolved as agent survived in a particular *environment.* The environment *affords* the agent what it needs to survive.

- The perception needed to release or guide the “right action” is directly in the environment, not inferred or memorized

  Ex. Red on Artic Terns== food

  Ex. flat surface at just over knee level - sitting

- Percepts are called *affordances* or said to be obtained through *direct perception*

Gibsonian Affordances

- How do you know you’re going fast in a car? Or in a space movie?
- How do animals know when to mate?
- How do mosquitoes know to bite in the most tender areas?
- What should you do when you think you’re being stalked by a mountain lion?
- What’s your favorite fishing lure?

One key to reactive systems

- The world is its own best model
    - don't remember anything about the world just use sensor data.

Reactive Systems

- Behavioral
  - Sense => plan => act
- Reactive
  - sense(<)=>act
  - sense(<)=>act
  - sense(<)=>act
  - each part of the controller directly tries to act on its sense data if releaser is applicable.

Easy Reactivity

- An easy reactive robot
  - all sense=>act behaviors are mutually exclusive
    - note disagreement about term “behavior”
    - no two will try to trigger competing actions
  - must have separate controller for every useful partition of sensor space
  - two bump sensors have what partition?
  - two ir distance sensors?
  - a sonar ring of 12 sensors?

simple robot example

- robot w/ two bump sensors
  - left bump
  - right bump
- control
  - if right bump pressed, turn left
  - if left bump pressed, turn right
  - if both pressed, back up and turn left.
  - if neither pressed, go forward.
- sensor space fully partitioned.
- problems?

mutually exclusive is intractable

- Mutually exclusive conditions quickly become intractable
    - even if you setup a lookup table for fastest processing
        - 4 dimensional array for 4 IR sensors
        - each element for relevant sensor range
        - faster than a big bunch of conditions (if else if etc)
        - huge
        - difficult to design
        - faster than deliberative.

Common problems

- Common problems for simple reactive robots
    - box canyon problem.
    - oscillation problem.
- two common answers
    - use randomness (sparingly)
    - remember one step back
        - not representing world, just robot's actions.

Consolidating sensor input

- fully partitioned input intractable
    - consolidate to make tractable
    - work through example from page 165 on board.

Action Selection

- What about non-mutually exclusive actions?
    - need action selection mechanism
    - this is about control architectures after all
    - Action selection
        - choosing which action should dominate at this time step
        - Command Arbitration
            - choose exactly one action
        - Command Fusion
            - merge two or more
            - good for vector based navigation.

need for parallelism

- each individual controller (behavior) needs to run in parallel
    - so either parallel processing or multitasking
    - all have to process each timestep
    - then controller selects

subsumption

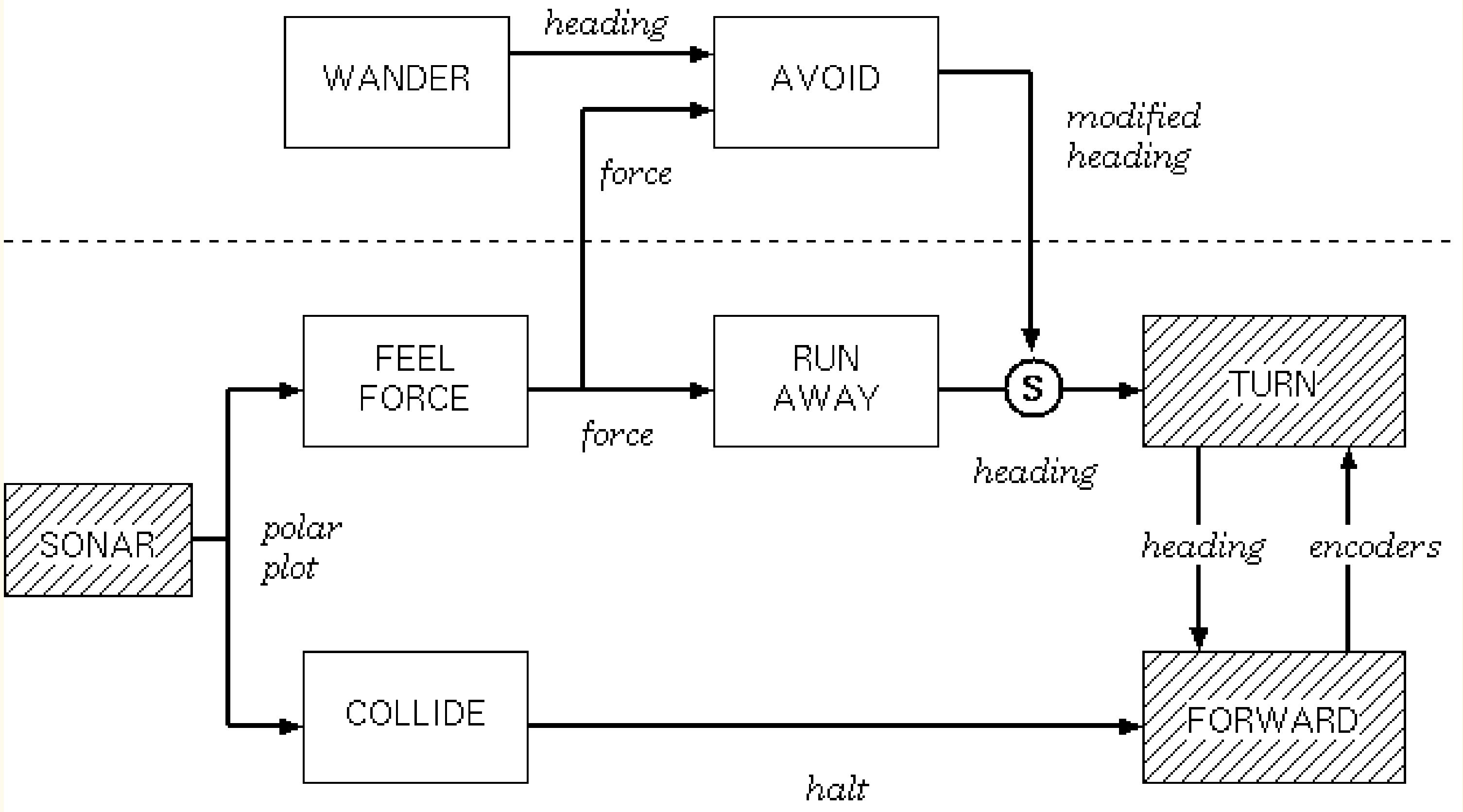
- Subsumption architecture
    - one of first and best known reactive architectures
    - Rod Brooks (MIT)
    - stirring up trouble
        - gets you tenure at MIT

subsumption description

- subsumption is layered architecture
  - higher layers can override lower layers in two ways
  - inhibition: the outputs of a behavior are turned off; the module receives input, does its computation – then nothing
  - suppression: the inputs to the module are turned off. no input so no computation

Subsumption Example

- image from Robyn Murphy's intro to AI robotics



Potential Fields

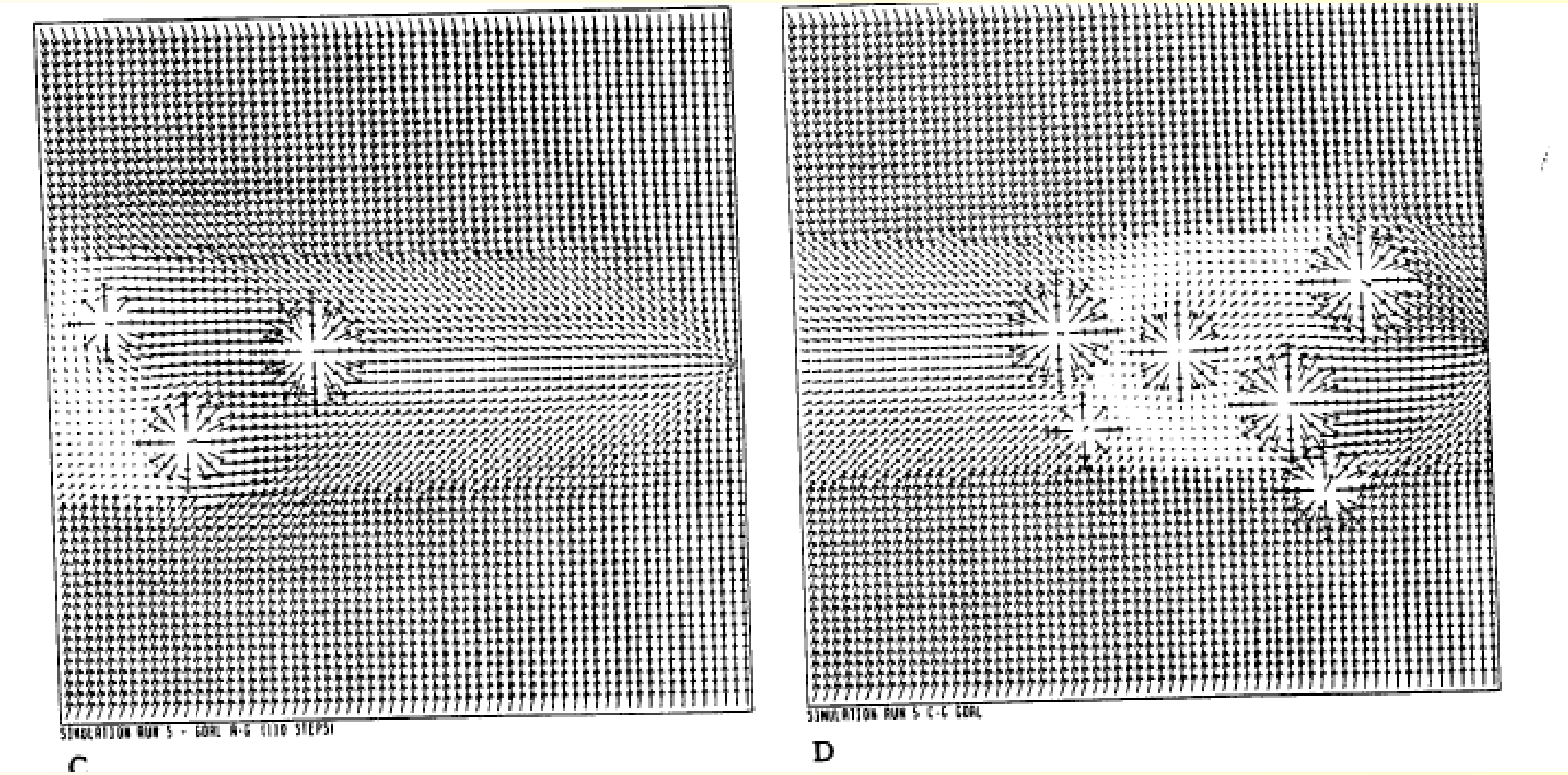
- Potential Fields is another popular reactive robot control technique
- often abbreviated pfields
    - developed by Ron Arkin (Georgia Tech)
    - primarily used for robot navigation
    - clever techniques to get around that.
        - but focus on navigation here
    - output of behavior is a vector
        - in 2d space very easy to model

Potential Fields

- Each behavior has as output a vector
    - several primitive vectors are basics of theory
    - can combine for more fields
    - from each (active) behavior robot 'feels' a force from vector
        - magnitude and direction.
    - when several behaviors all produce vectors of force, final output (behavior selection) is done by vector summation.
    -

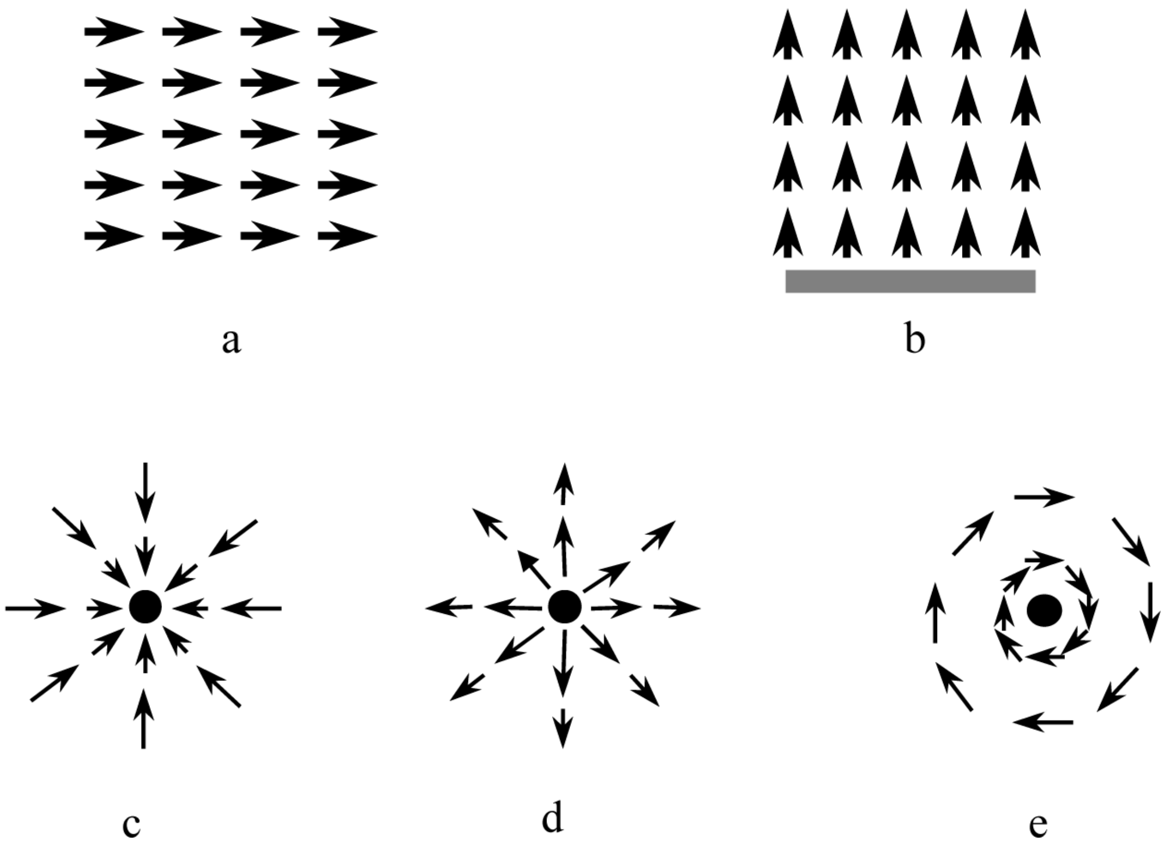
Potential fields

- Even though behavior is a single vector,
  - visualize by calculating vectors for every spot in entire environment


SIMULATION RUN 5 - GOAL A-G (110 STEPS)
C
SIMULATION RUN 5 C-G GOAL
D

Five primitive pfields

- a)uniform
- b)perpendicula
- c) attraction
- d) repulsion
- e)tangential
- sometimes: random



Using primitive pfields

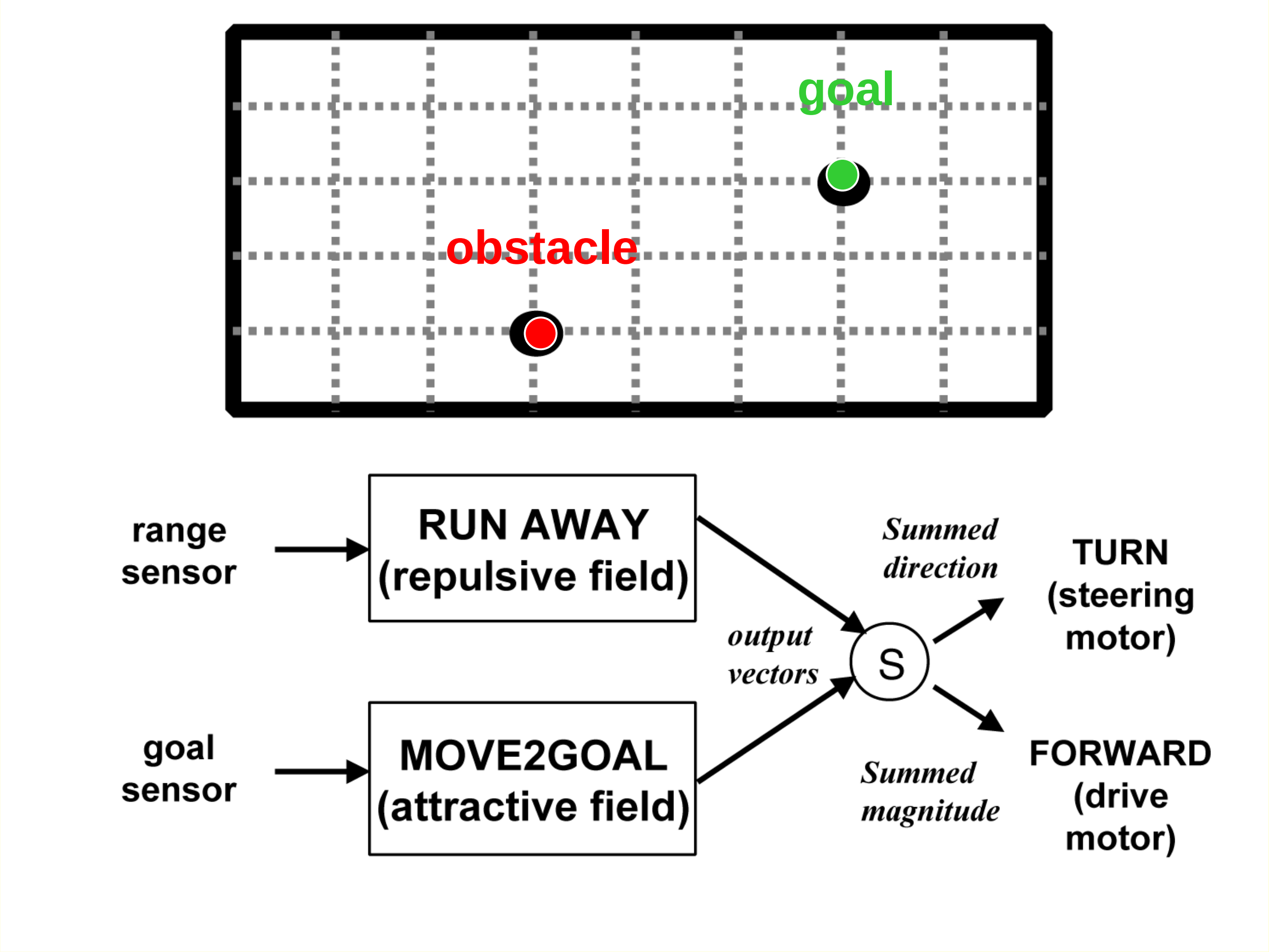
- How would we use
    - a)uniform
    - b)perpendicular
    - c) attraction
    - d) repulsion
    - e)tangential
    - random

More questions for you

- What field would you use for
    - moving toward the light
    - avoiding obstacles ?

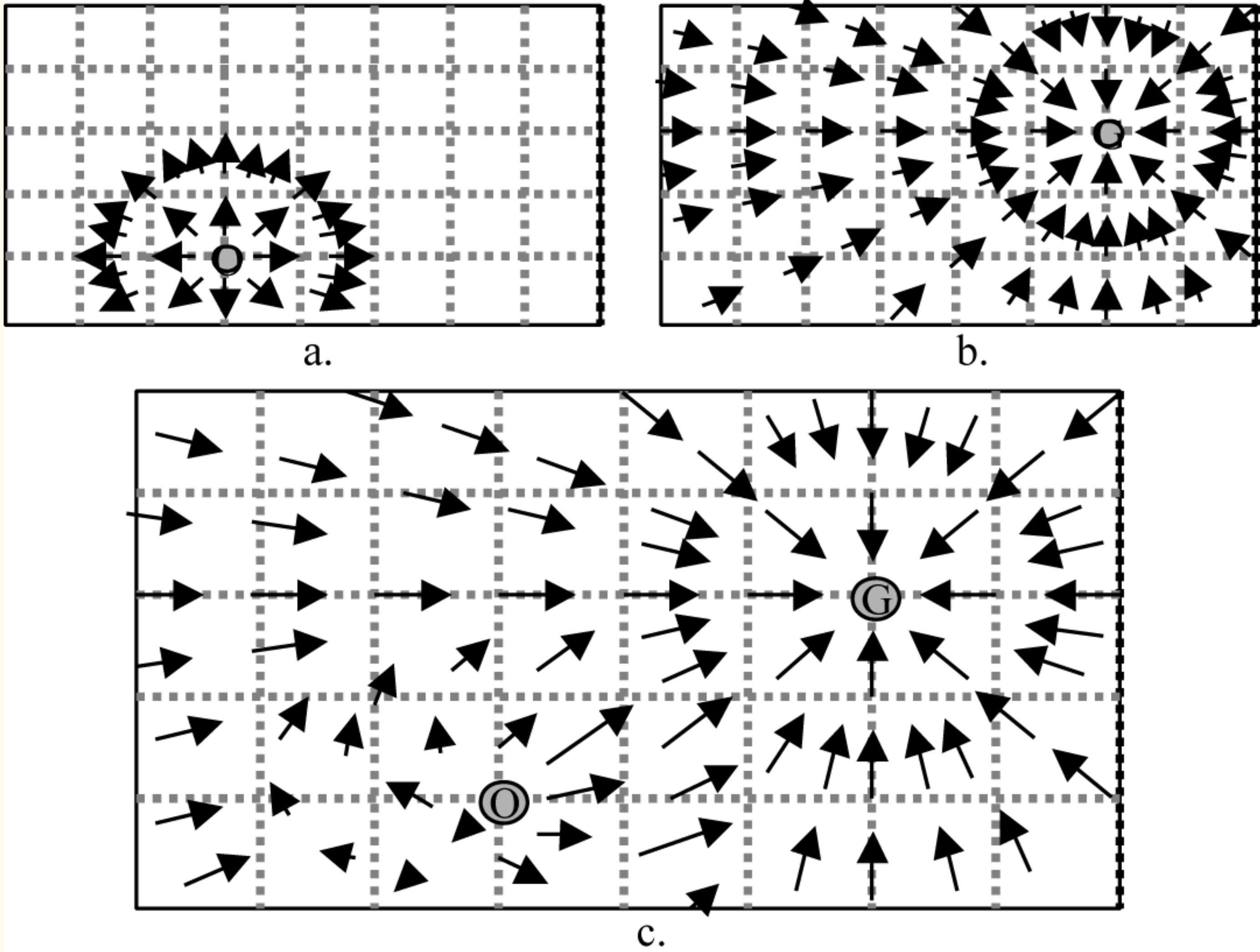
Combining fields

- Get more interesting behavior by combining more than one. Example: From AI Robotics
-



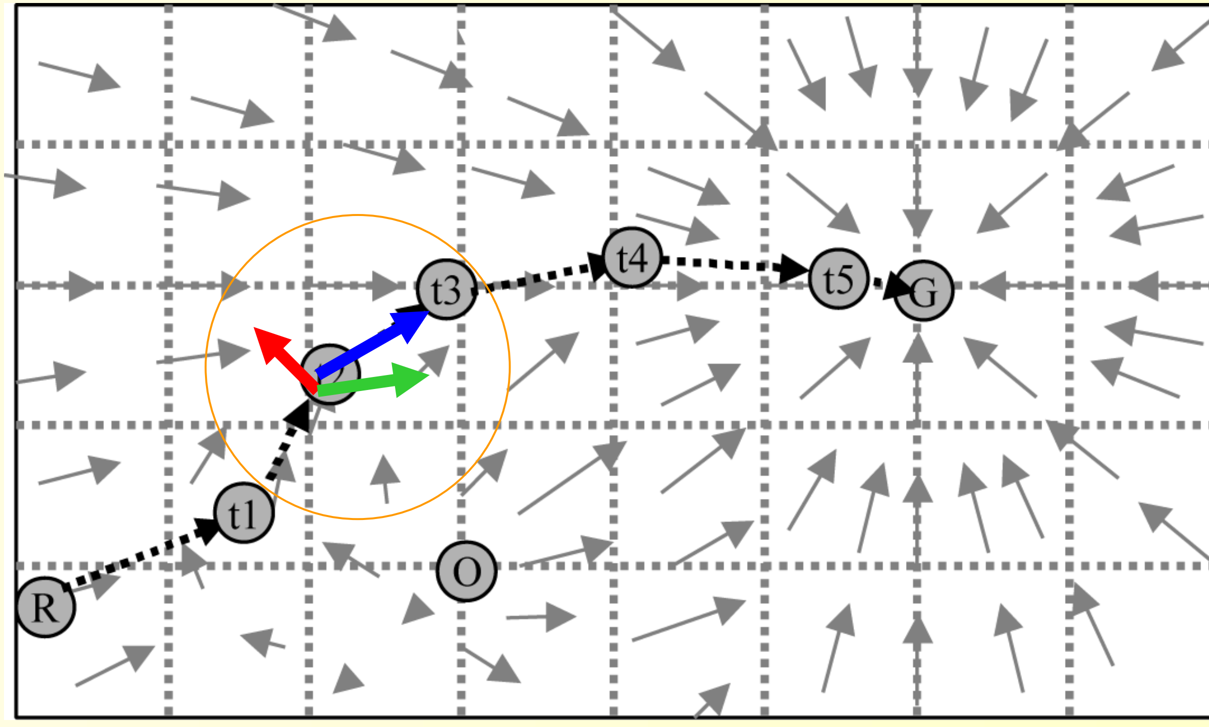
The visualized fields

- using pfields to visualize the final output
- goal: visible from 10 ft
- obstacle from 5 ft



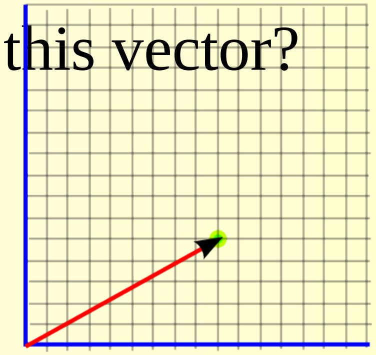
Final Trajectory

- Visualize whole field
    - but robot feel two vectors at any timestep.
    - sums them up to get final move vector



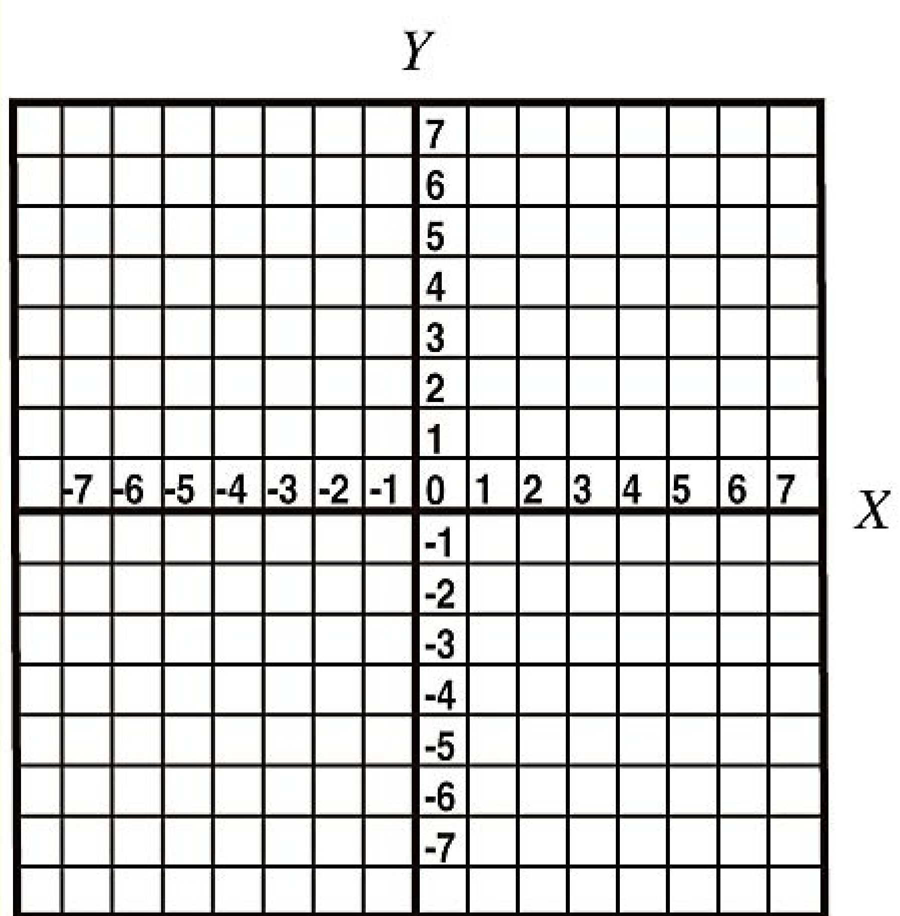
Implementing Pfields

- When a robot receives a sensor input
  - we'll represent vector magnitude and direction as two numbers
    - x change, y change
    - rise and run
    - so what is the representation of this vector?



Representing robot in the field

- Let the robot be at origin
  - so vector back and to left might be [-5,-5]



combining behaviors

- Combine behaviors with simple vector addition
  - with our representation of vectors
  - add both x change values, result is new xchange value
  - same with y change
  - example
    - light finding robot with three light sensors
    - left, right, center
    - left 34 (lots of light)
    - center 45 (still lots of light)
    - right 110 (medium amount of light)

sensor input mapped to behavior

- Light Sensor input

left light sensor value	**vector output**
0-50	[-10, 0]
51-75	[-5, 0]
76-110	[-3, 0]
111-150	[-1, 0]

Right sensor value	**vector output**
0-50	[10, 0]
51-75	[5, 0]
76-110	[3, 0]
111-150	[1, 0]

What about the final result?

- do out center lookup table on board
    - what is resulting behavior from just these attraction vectors?
    -

better turn now

- Now you have the vector,
  - in omni-directional robot great, else turn
  - dot product of two vectors a.b
    - |a| * |b| Cos(theta)
      - where |a| is the magnitude of a
    - solve for theta
      - theta = arccos(a.b/(|a|*|b|))
      - unit vector discussion.

Calculating the turn angle

- a.b is also
  - a[x]*b[x] + a[y]*b[y] so
  - theta = arccos((a[x]*b[x] + a[y]*b[y])/ (|a|*|b|))
  - now use forward vector as a and calculated vector as b; calculate turn – then go forward by |b|

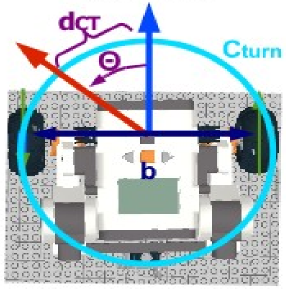
Moving through the turn

ack: These next slides based on tutorial from shelden Robotics

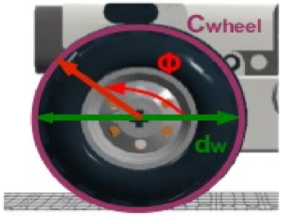
- So you've calculated a turn angle of theta
  - How do you calculate the turn itself?

Making Precise Turns: Understanding the Relationships between the Robot and its Wheels

Top View



Side View



Terms Defined

b	Robot Wheelbase
C_{turn}	Robot Circumference of turn
	$C_{turn} = b \times \pi$
Θ	Desired Turn Angle
d_{CT}	Distance along C_{turn}
d_w	Wheel Diameter
C_{wheel}	Wheel Circumference
	$C_{wheel} = d_w \times \pi$
Φ	Motor Rotation
360°	degrees around a circle

We want the robot to turn a set number of degrees (Θ) by turning the power wheels in opposite directions. Θ can also be expressed in terms of its fractional value of the whole circumference and it is also equal to the distance along the circumference we want to turn the robot.

$$(\Theta / 360) = (d_{CT} / C_{turn})$$

And we can solve this for the Distance along the circumference to turn the wheels.

$$d_{CT} = (\Theta / 360°) \times (C_{turn})$$

We want to be able to program this in terms of wheel rotations or degrees.

Recognize that the distance along the Robot's Turning Circumference (d_{CT}) is the same distance the robot's wheels must turn along the Circumference of each Wheel (C_{wheel}) to make the turn.

This leads to the following relationship:

Number of Rotations to Program $= (d_{CT}) / (C_{wheel})$ in NXT-G

Number of Degrees to Program $= (360°) \times (d_{CT}) / (C_{wheel})$ in NXT-G or RoboLab

And if we substitute the terms that make d_{CT} into the above equations, they simplify rather nicely.

Number of Rotations to Program $= (C_{turn}) / (C_{wheel}) \times (\Theta) / (360°)$ in NXT-G

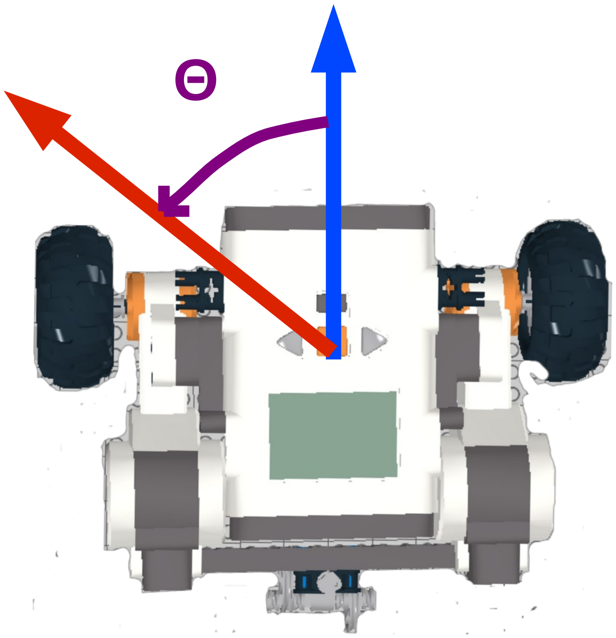
Number of Degrees to Program $= (C_{turn}) / (C_{wheel}) \times (\Theta)$ in NXT-G or RoboLab

Summary

Based on our mission, we know how many degrees we need to turn the robot, or our desired turn angle **Θ.**

And **Θ** corresponds to a fraction (**dCT**) of our robot's turning circumference (**Cturn**):

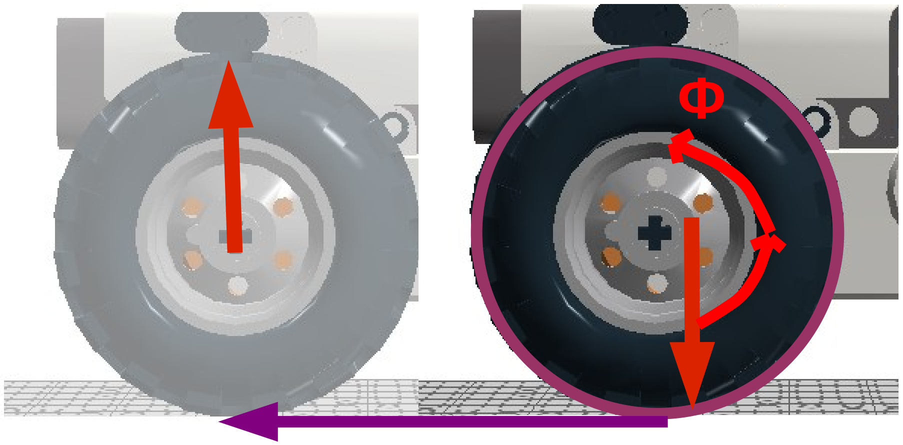
$$\mathbf{dCT} = \mathbf{Cturn} \times \frac{\mathbf{\Theta}}{\mathbf{360^\circ}}$$



And **dCT** is **also** a fraction of **Cwheel**

$$\textbf{\# of Wheel Motor Rotations} = \frac{\mathbf{dCT}}{\mathbf{Cwheel}}$$

$$\textbf{\# of Wheel Motor Degrees} = \mathbf{360^\circ} \times \frac{\mathbf{dCT}}{\mathbf{Cwheel}}$$



Summary – 2 Combining terms

Based on our mission, we know how many degrees we need to turn the robot, or our desired turn angle **Θ.**

And **Θ** corresponds to a fraction (**dCT**) of our robot's turning circumference (**Cturn**):

And **dCT** is **also** a fraction of **Cwheel**

Substituting **dCT**'s factors into the # of Rotations or Degrees has some interesting results.

$$\mathbf{dCT} = \mathbf{Cturn} \times \frac{\Theta}{360°}$$

$$\textbf{\# of Wheel Motor Rotations} = \frac{\mathbf{dCT}}{\mathbf{Cwheel}} = \frac{\mathbf{Cturn}}{\mathbf{Cwheel}} \times \frac{\Theta}{360°}$$

$$\textbf{\# of Wheel Motor Degrees} = 360° \times \frac{\mathbf{dCT}}{\mathbf{Cwheel}} = 360° \times \frac{\mathbf{Cturn}}{\mathbf{Cwheel}} \times \frac{\Theta}{360°}$$

Summary – 2 Combining terms

Based on our mission, we know how many degrees we need to turn the robot, or our desired turn angle **Θ.**

And **Θ** corresponds to a fraction (**dCT**) of our robot's turning circumference (**Cturn**):

And **dCT** is **also** a fraction of **Cwheel**

Substituting **dCT**'s factors into the # of Rotations or Degrees has some interesting results that make all the math EASIER.

$$\mathbf{dCT} = \mathbf{Cturn} \times \frac{\Theta}{360°}$$

$$\textbf{\# of Wheel Motor Rotations} = \frac{\mathbf{dCT}}{\mathbf{Cwheel}} = \frac{\mathbf{Cturn}}{\mathbf{Cwheel}} \times \frac{\Theta}{360°}$$

$$\textbf{\# of Wheel Motor Degrees} = 360° \times \frac{\mathbf{dCT}}{\mathbf{Cwheel}} = \cancel{360°} \times \frac{\mathbf{Cturn}}{\mathbf{Cwheel}} \times \frac{\Theta}{\cancel{360°}}$$

$$\textbf{\# of Wheel Motor Degrees} = \frac{\mathbf{Cturn}}{\mathbf{Cwheel}} \times \Theta$$

Reading Assignment