SWARM ROBOTICS

PART 1:

INTRO AND BLOCKLY / WAGGLE

Dr. Andrew Vardy
COMP 4766 / 6912
Department of Computer Science
Memorial University of Newfoundland
St. John's, Canada

SWARM ROBOTICS?

- "Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment."
  - [Șahin, E. (2004). Swarm robotics: From sources of inspiration to domains of application. In International workshop on swarm robotics (pp. 10-20). Springer, Berlin, Heidelberg.]

- “Swarm robotics is the study of how **a large number of relatively simple physically embodied agents** can be designed such that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment.”

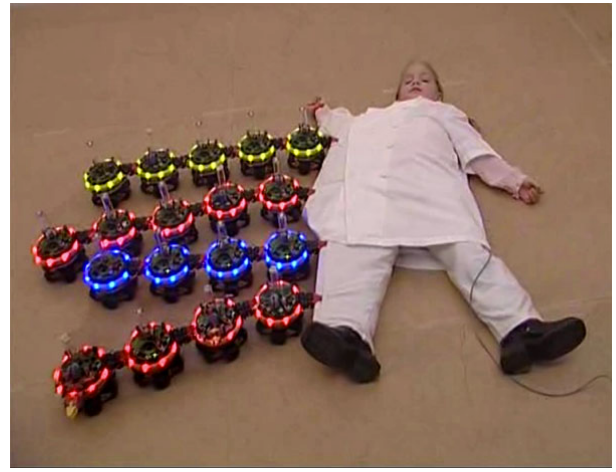
- “Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment.”

- “Swarm robotics is the study of how a large number of relatively simple ~~physically embodied agents~~ **robots** can be designed such that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment.”

Kilobots: 1024

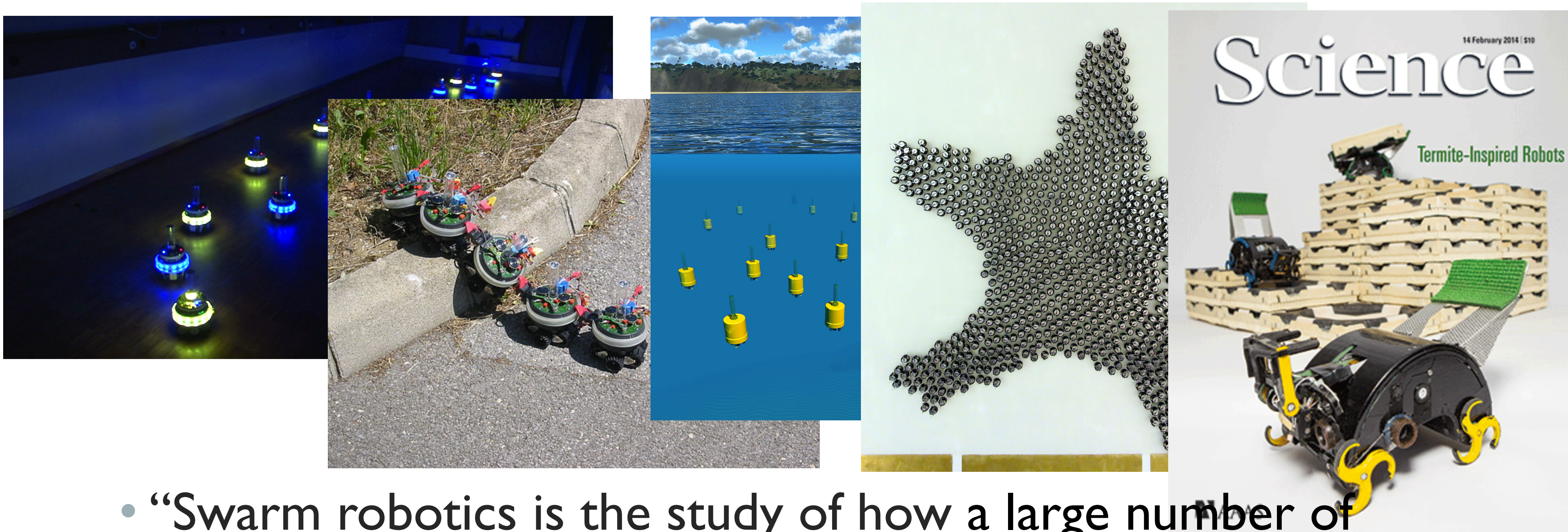
Many limited robots

S-bots: 10-20



Fewer, more capable robots

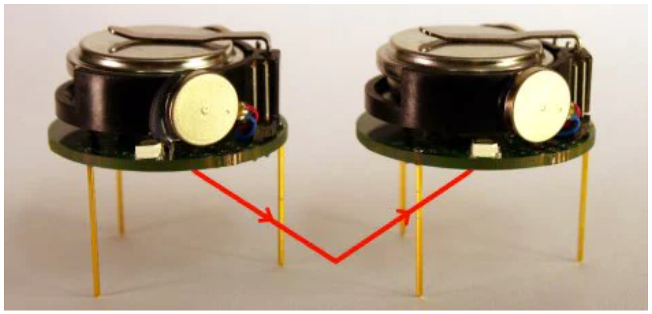
- “Swarm robotics is the study of how a large number of relatively simple ~~physically embodied agents~~ **robots** can be designed such that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment.”

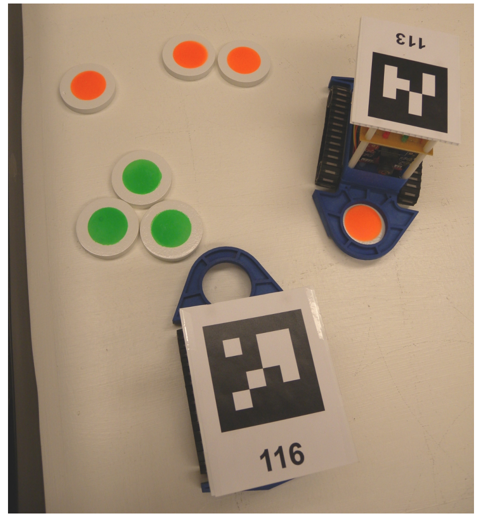


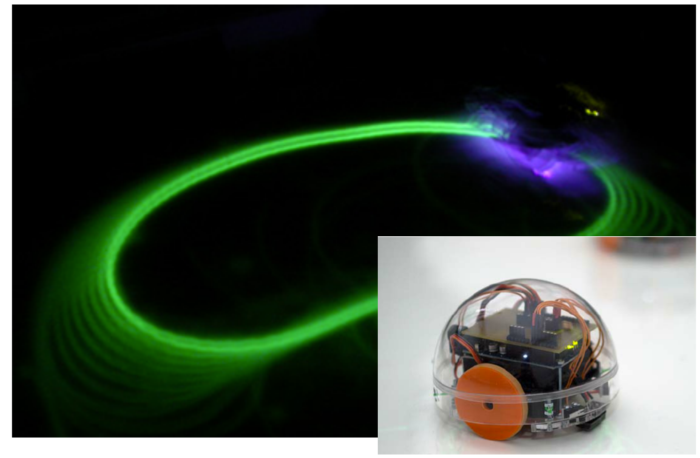
- “Swarm robotics is the study of how **a large number of relatively simple physically embodied agents** can be designed such that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment.”

Indirect communication (perceive results of others actions)

Direct communication (local)







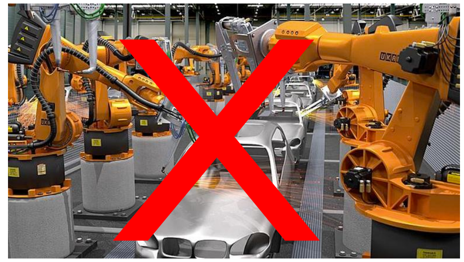
- "Swarm robotics is the study of how **a large number of relatively simple physically embodied agents** can be designed such that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment."

CHARACTERISTICS OF SWARM ROBOTICS

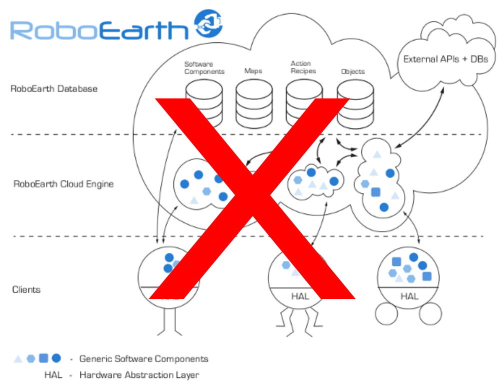
- Swarm Robotics: A multi-robot system with the following characteristics:
  - robots are **autonomous**;
  - robots are **situated** in the environment and can act to modify it
  - robots' **sensing and communication capabilities are local**
  - robots **do not have access to centralized control** and/or to global knowledge
  - robots **cooperate** to tackle a given task

- Brambilla, M., Ferrante, E., Birattari, M., & Dorigo, M. (2013). Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1), 1-41.

- robots are **autonomous**;



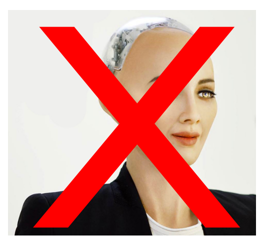
- robots are **situated** in the environment and can act to modify it



- robots' **sensing and communication capabilities are local**

- robots **do not have access to centralized control** and/or to global knowledge

- robots **cooperate** to tackle a given task



- Definition:
  - “Swarm robotics is the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from the local interactions among the agents and between the agents and the environment.”

- Characteristics:
  - robots are **autonomous**;
  - robots are **situated** in the environment and can act to modify it
  - robots’ **sensing and communication capabilities are local**
  - robots **do not have access to centralized control** and/or to global knowledge
  - robots **cooperate** to tackle a given task

What robots exist that adhere to this definition and exhibit these characteristics?

NATURAL ROBOTS: THE HONEYBEES

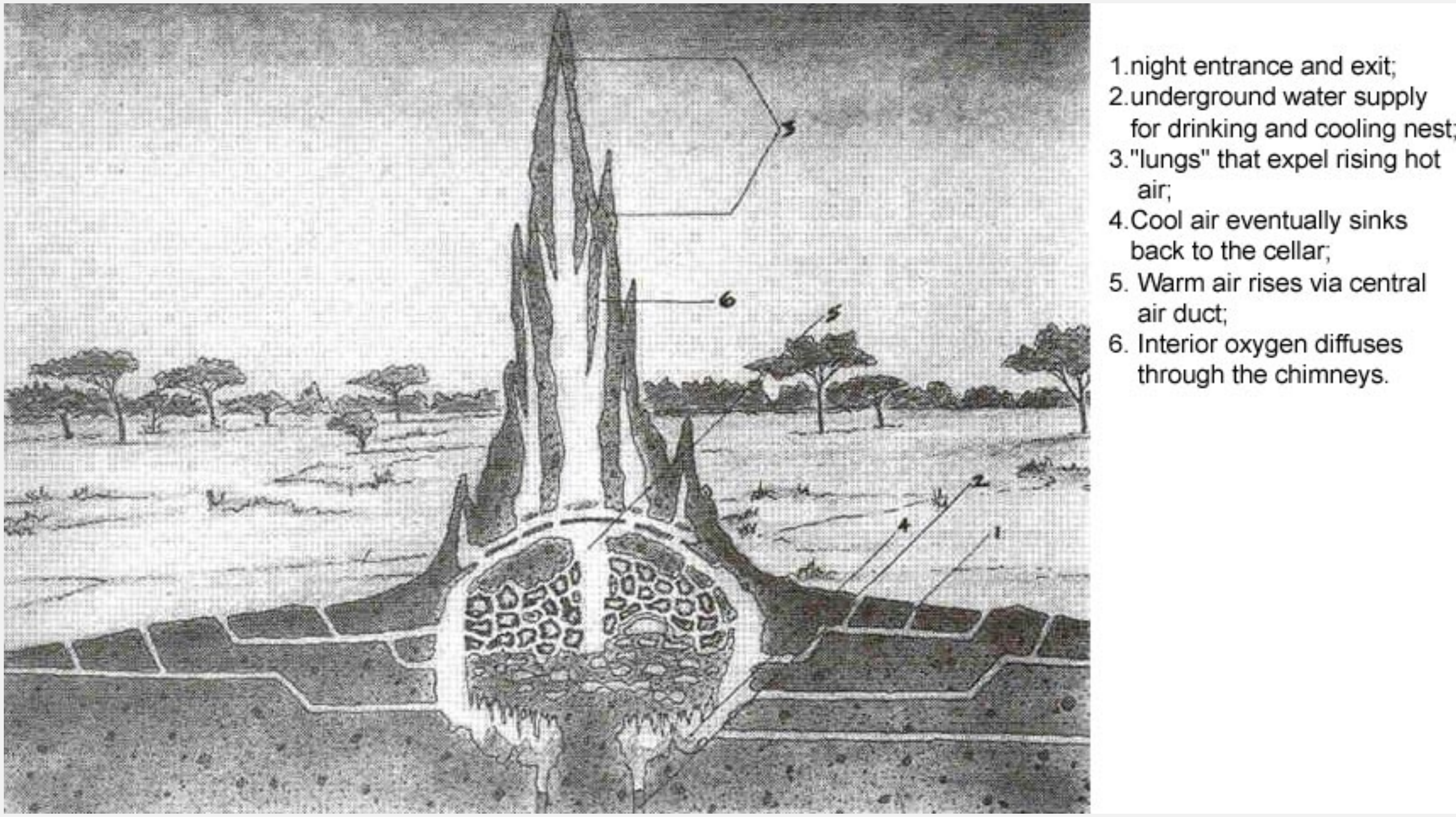
- Democratic nest selection
  - Seeley, T. D. (2010). *Honeybee democracy*. Princeton University Press.
- Waggle dance communication

NATURAL ROBOTS: THE ANTS

- Extremely successful family
  - 15 - 20% of terrestrial biomass
- Leafcutter ants
  - Invented agriculture millions of years before us!

NATURAL ROBOTS: THE TERMITES

- Termite mounds taller than a computer scientist
  - Construct extremely sophisticated structures…



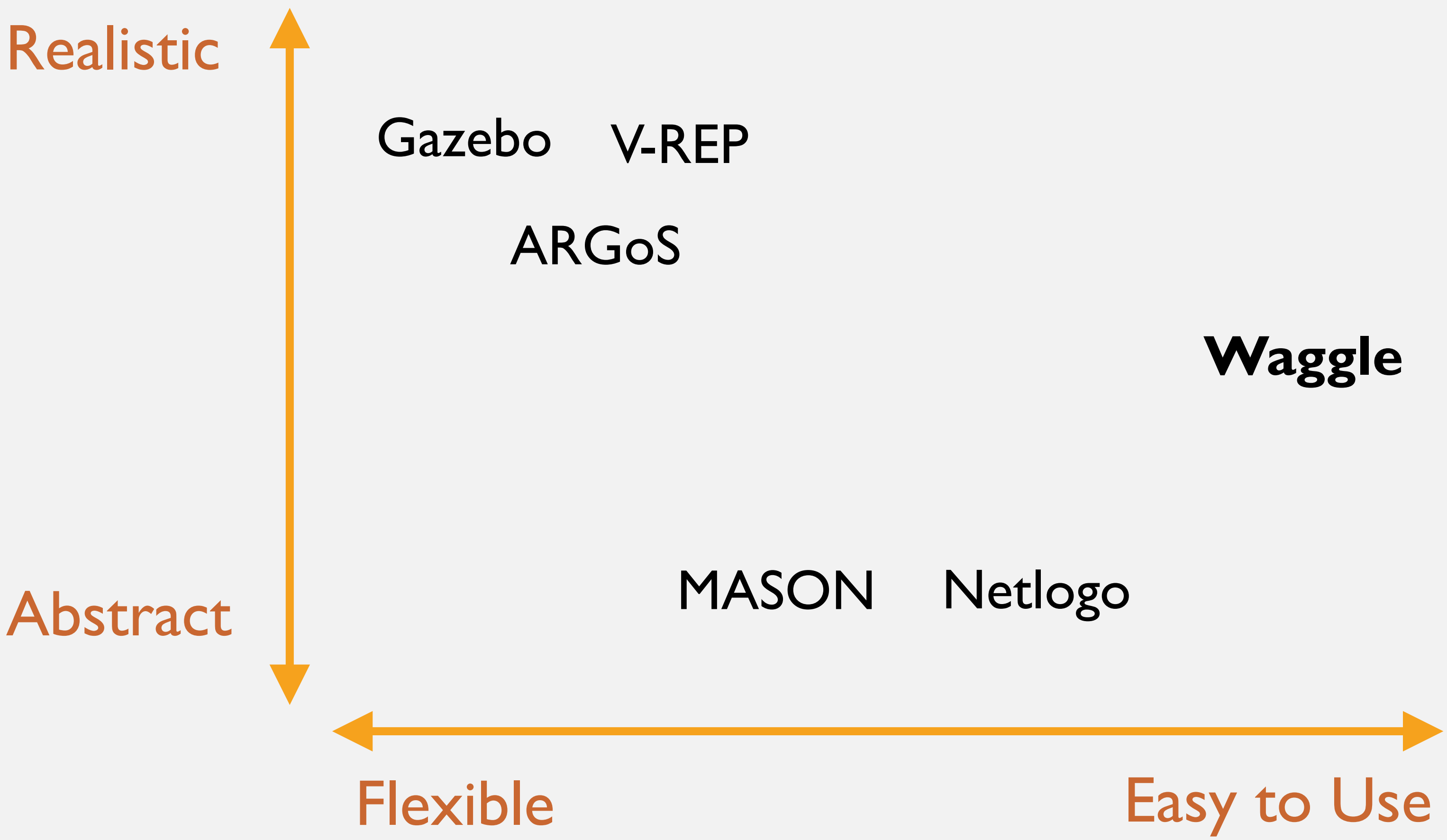
Sophisticated architecture and engineering from insects with small brains, poor vision, and no central coordination

OUR TOOLS:

BLOCKLY AND WAGGLE

WAGGLE

- We will be using Waggle, an online simulation and programming environment
  - 2-D Physics: Matter.js
  - Visual programming: Blockly
- There are other tools for simulating swarms of robots:
  - Agent-based simulators: Netlogo, MASON
  - Robot simulators: ARGoS, V-REP, Gazebo
- Why another simulation tool?


Realistic
Gazebo
V-REP
ARGoS
Waggle
Abstract
MASON
Netlogo
Flexible
Easy to Use

BLOCKLY

- A library for building visual programming apps
  - Started by Google, now open-source
  - Used for hundreds of educational apps on the web and on Android / iOS devices
- Programming for kids?
  - Yes
  - ”But we’re not kids”
  - Blockly is an easily learned language for people with various levels of programming experience

EXERCISE #1:

BLOCKLY GAMES

5 MINUTES

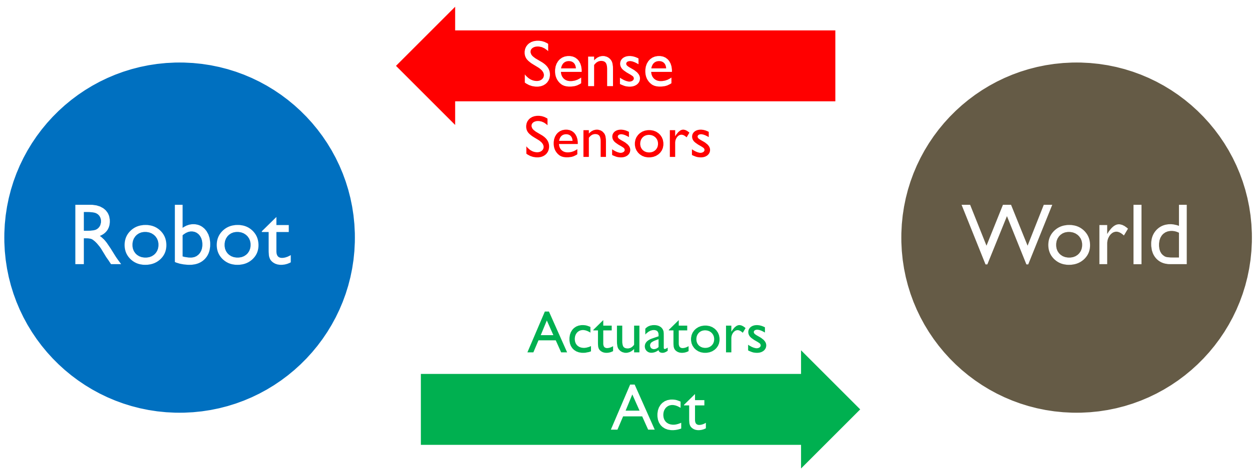
- Go to Blockly Games:
  - https://blockly-games.appspot.com/
- Complete "Puzzle"
- Complete as many levels as you can of "Maze"

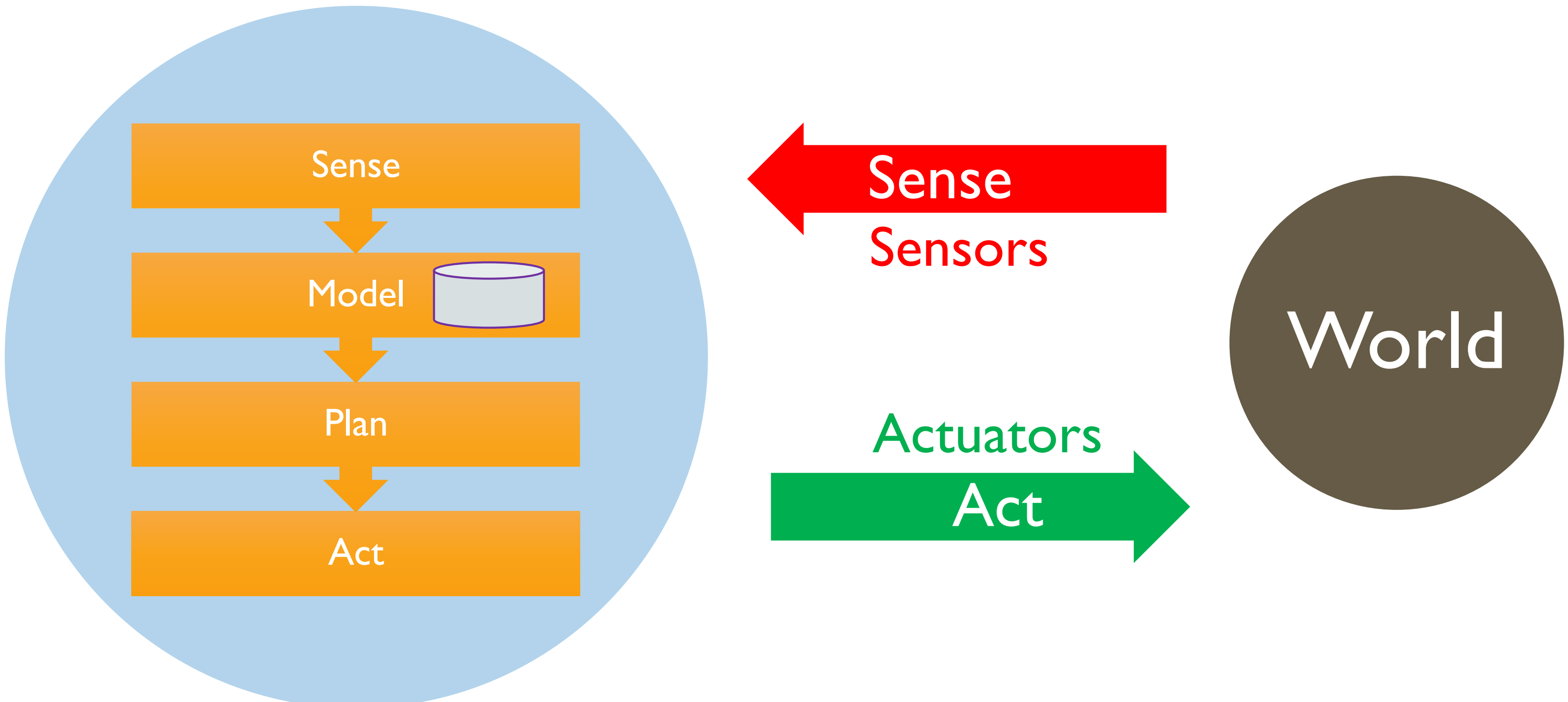
- Experienced programmers:
  - There are interesting challenges as you progress into the higher levels

ROBOTICS CONCEPTS

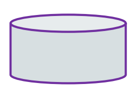
THE SENSE / ACT CYCLE

- Robots exist in a constant cycle of perception (via sensors) and movement (via actuators)



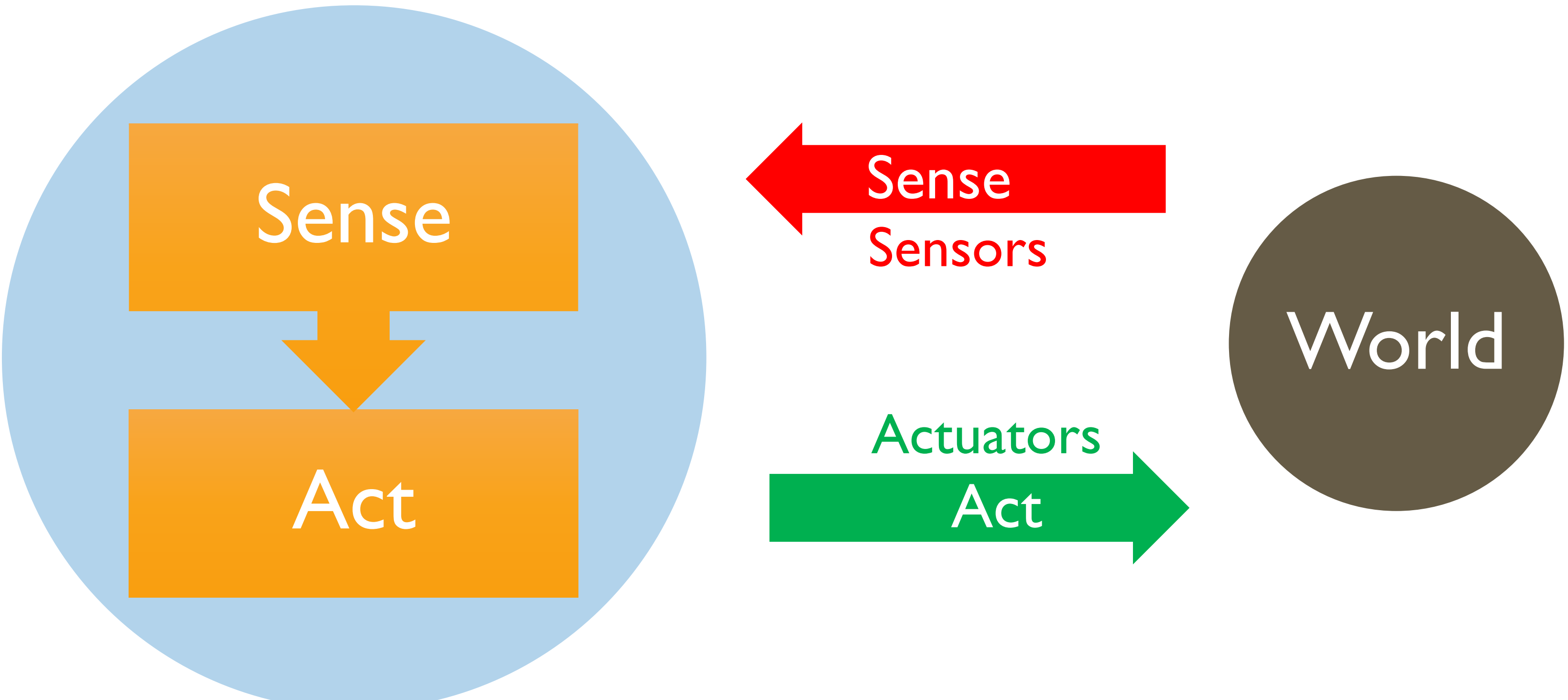


- The robot's controller may be very **deliberative**
  - **Carefully build a model of the world, plan a trajectory, then execute it**



- Or the controller may be **reactive**
  - **There is a direct mapping of sensor states to actions**
  - **The model (if any) is small**

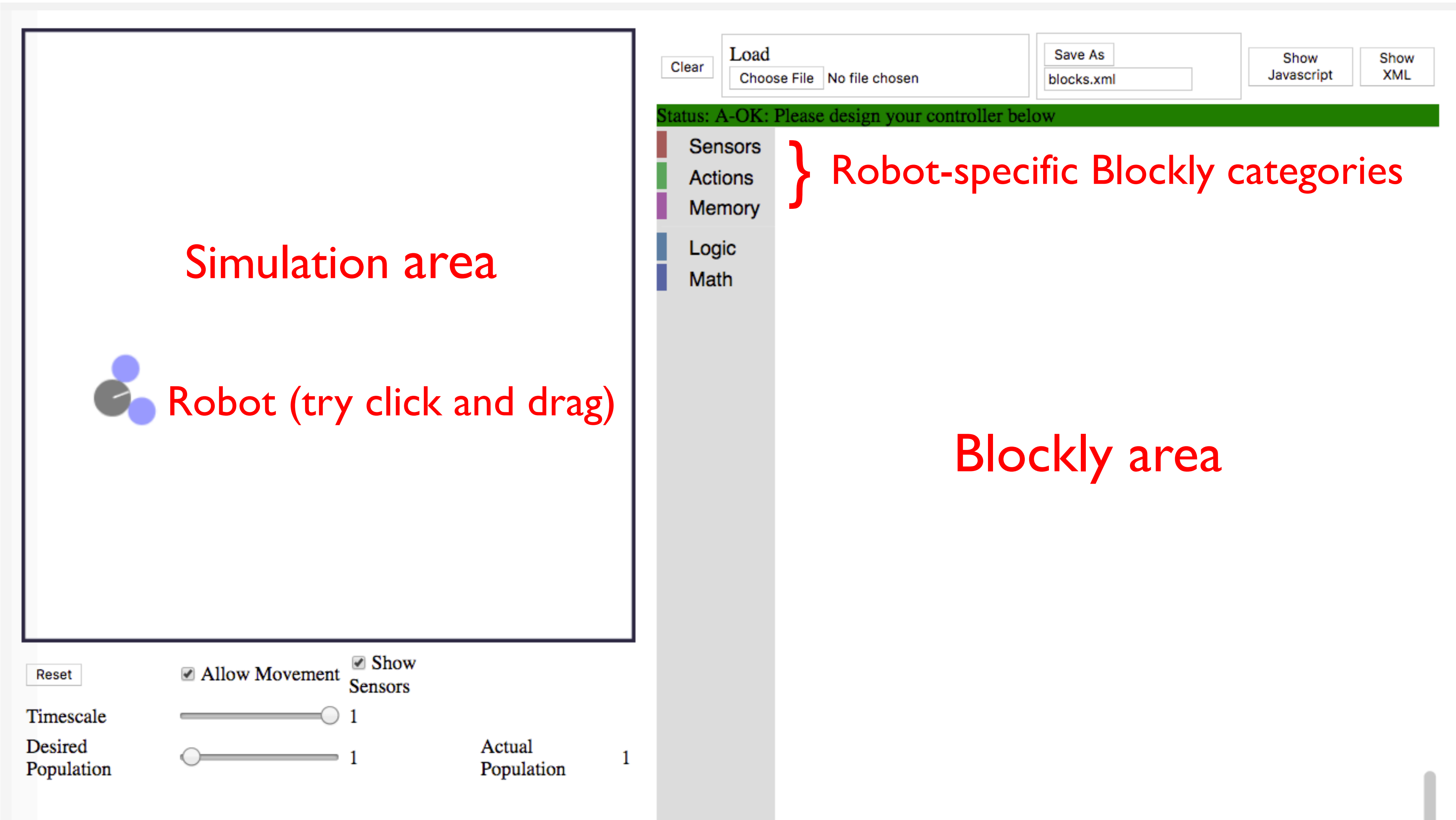
We will be using reactive controllers



TUTORIAL LEVEL

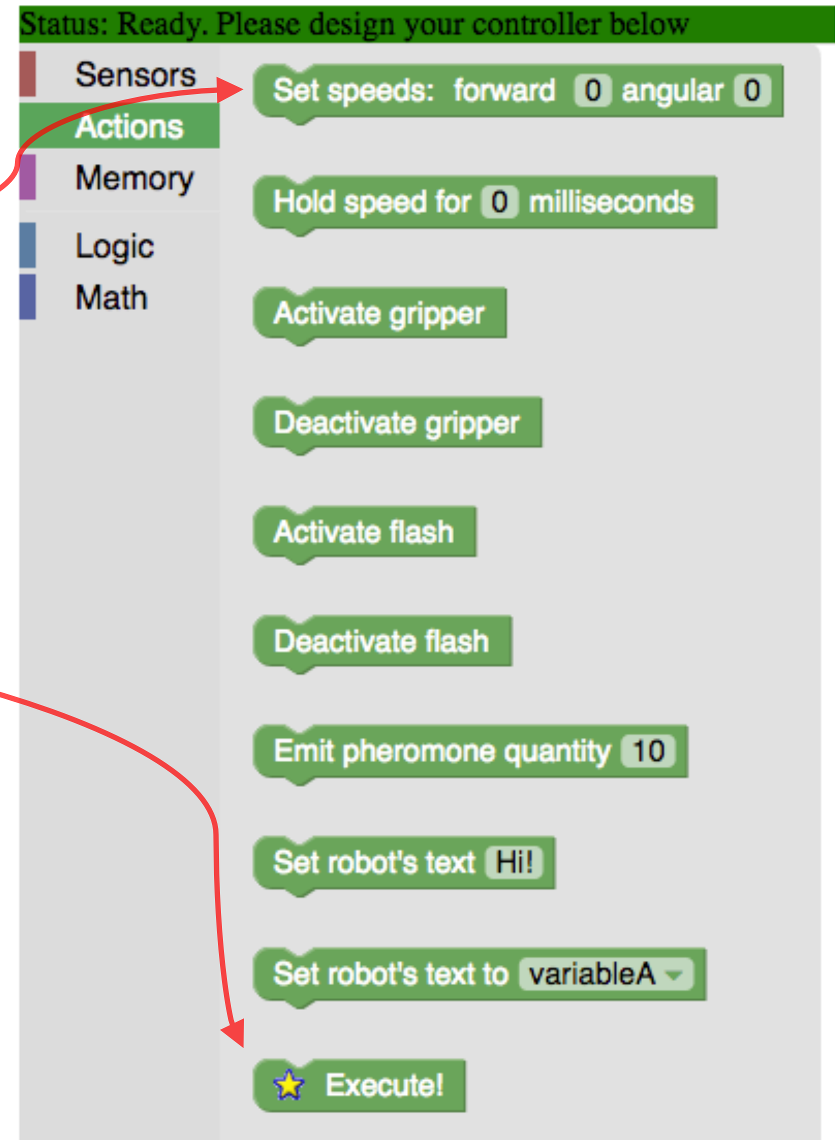
THE WAGGLE PAGE

- Go to the main Waggle page:
  - http://bots.cs.mun.ca/waggle/
  - A previous version of these slides are available here
- Select the **Tutorial** level
- Take note of the following features…

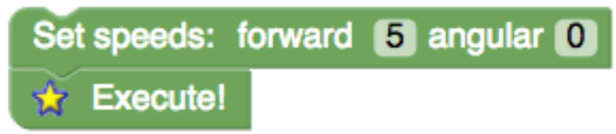

Clear
Load
Choose File No file chosen
Save As
blocks.xml
Show Javascript
Show XML
Status: A-OK: Please design your controller below
Sensors
Actions
Memory
Logic
Math
} Robot-specific Blockly categories
Simulation area
Robot (try click and drag)
Blockly area
Reset
Allow Movement
Show Sensors
Timescale
1
Desired Population
1
Actual Population
1

- Within the Blockly area, click on the "Actions" category
- Now click-and-drag on "Set speeds" and pull it into the open white area
- Click on the '0' next to "forward" and change it to some number in the range [-10, 10]
  - Robot does not move
- Click-and-drag "Execute" beneath "Set speeds" so that they connect like this:

  - Now the robot moves!



OUR FIRST CONTROLLER!

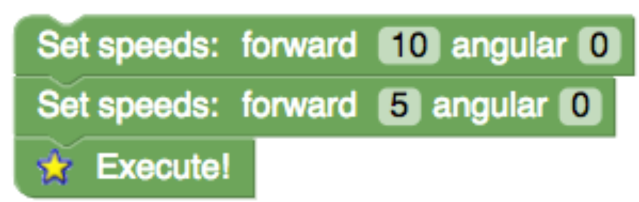
- So this is our first controller: 

- A controller senses the world, then acts
  - But this controller is so basic, it doesn't even do any sensing
- The controller is executed by the simulator many times per second
  - There is no need for a "loop", the controller gets called over-and-over by default

DEFERRED EXECUTION

- Notice how the robot did not move without "Execute"

  - The controller uses a **deferred execution** model, meaning that nothing happens until "Execute". The controller is executed top-to-bottom and each block in the "Actions" category sets something about the action the robot will take. However, no action is taken until "Execute".
- e.g. The following controllers all behave the same because the speeds set by the blocks on the bottom overwrite the speeds set above:



EXERCISE #2:

BASIC MOVEMENT

5 MINUTES

- Experiment with "Set speeds" to derive controllers that travel in...
  - Straight lines
  - Circles
- Note the importance of 

- Experiment with positive / negative speeds
  - Try the following:
    - Counter-clockwise circle, moving forwards
    - Counter-clockwise circle, moving backwards

EXERCISE #2: REVIEW

- Move forward (speed 10):

- Turn clockwise (speed 10):

  - Set angular speed negative for counter-clockwise turns

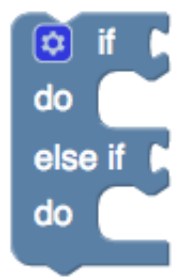
- Move in a tight clockwise circle:

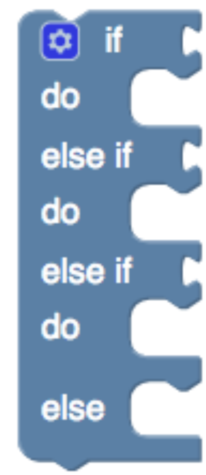
  - Reduce angular speed for a larger circle; Reduce forward speed for a smaller circle
- Use negative values for backward motion and counter-clockwise turns

THE CONDITIONAL BLOCK

- Under "Logic" select the conditional block: 

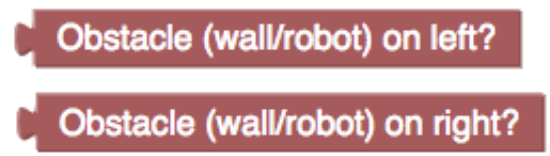
- Connected to the top part will be a condition block which must evaluate to True or False
  - If the condition block is True, the block(s) within the "do" will execute
- Click on the star to customize this block. Here are some possibilities:

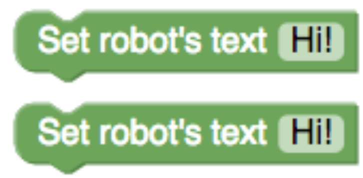




CONDITIONAL + SENSORS

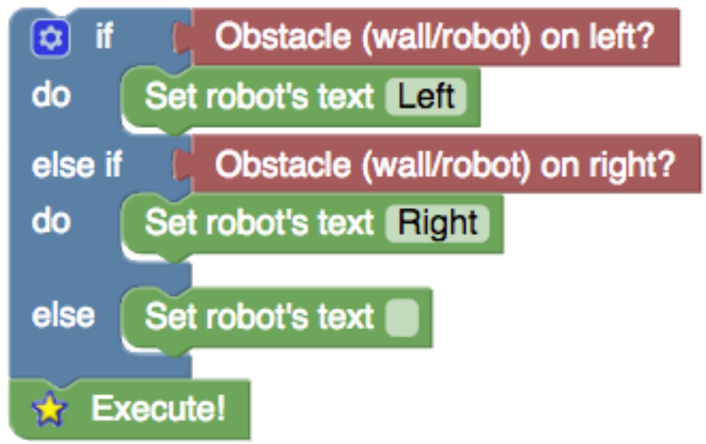
- Create an if – else if – else conditional block: 

- Select the "Sensors" category and drag in these two blocks: 

- Now under the "Actions" category drag in two of these: 

- Also under "Actions" drag in an 

- Connect the blocks together like this:



- Note that you have to customize the text for each "Set robot's text" block
- Drag the robot around, bumping into the walls to test it

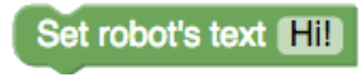
EXERCISE #3:

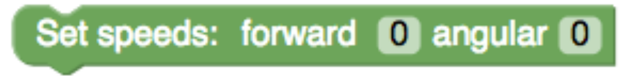
OBSTACLE AVOIDANCE

5 MINUTES

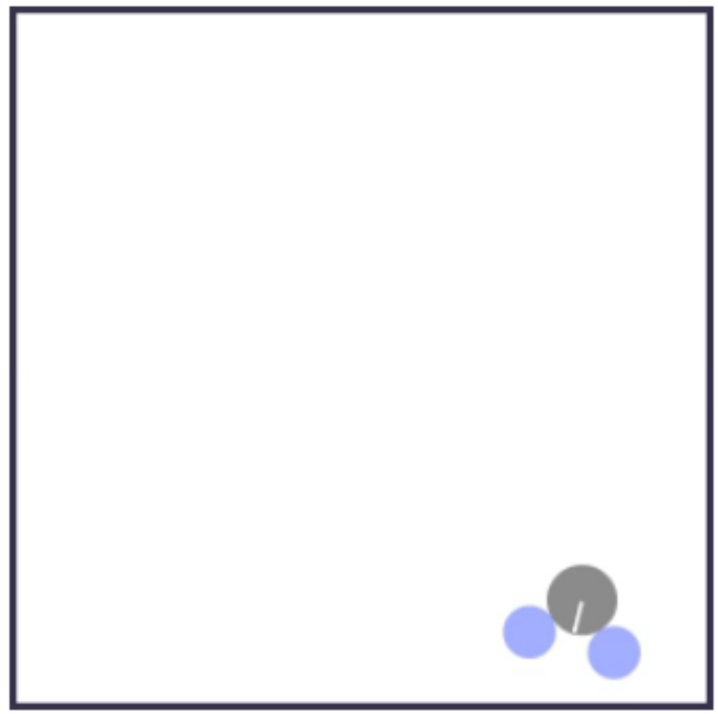
Conditions:

Should work with any number of robots

- Replace the 

blocks with 

blocks and adjust the speeds to achieve this:



- Note: Should still work when bouncing against walls on the right

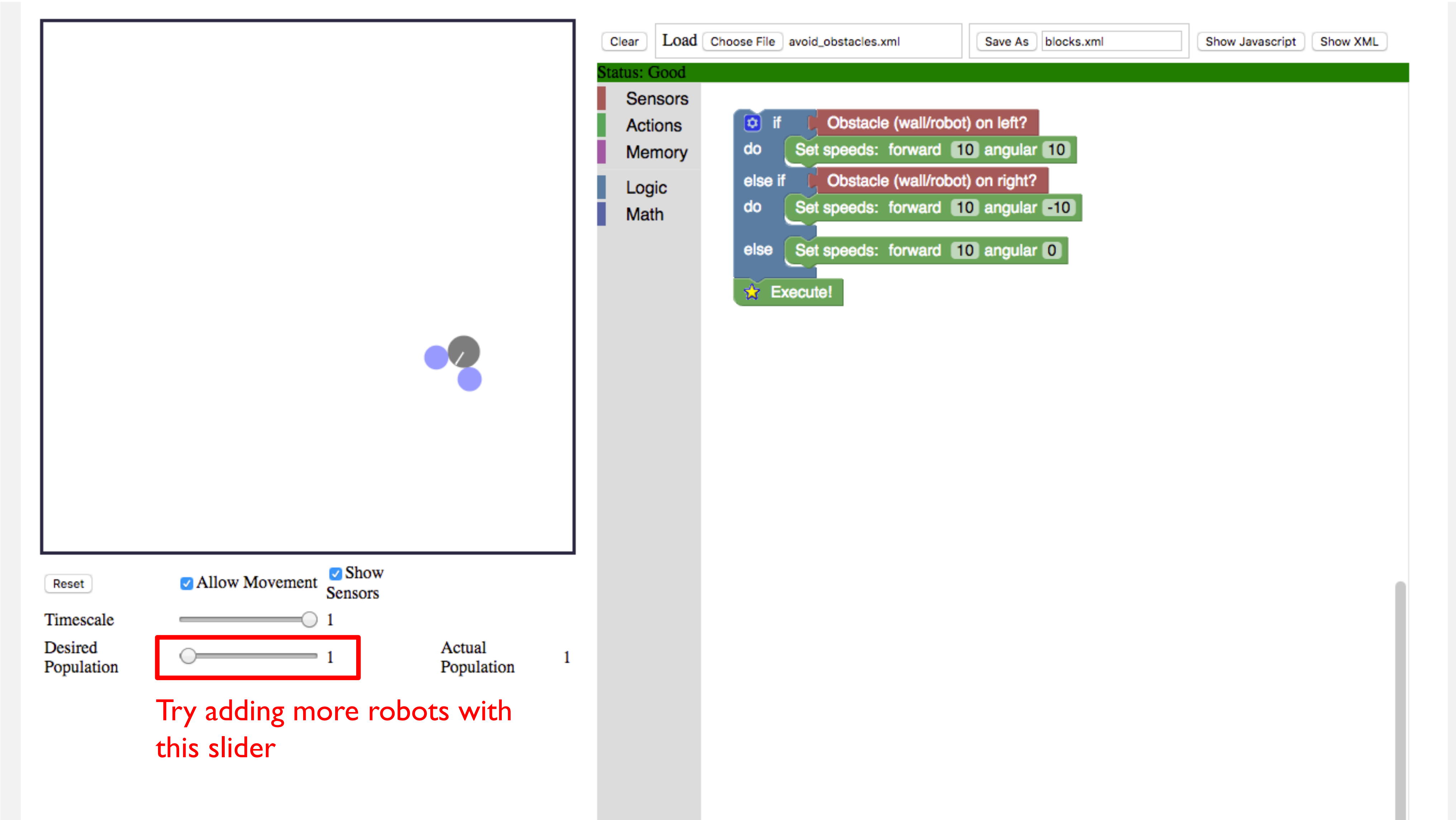
EXERCISE #3: REVIEW

- Your solution should look something like this:
- You could reduce the speeds a little, but the robot might hit the wall (not a big problem)

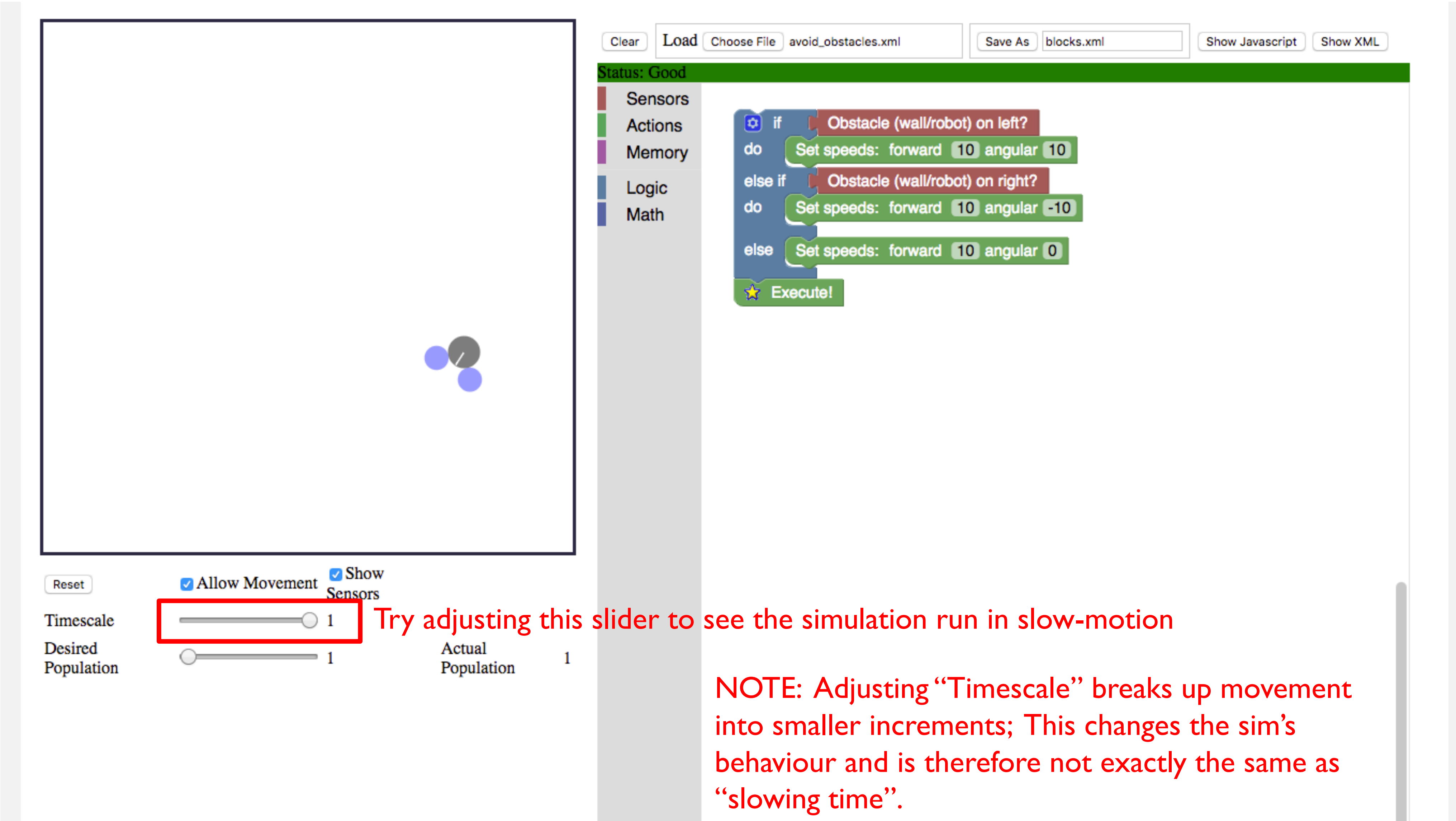
WAGGLE:
TOUR OF FEATURES AND BLOCKS

WAGGLE FEATURE / BLOCK TOUR

- Continuing on the Tutorial level, we will introduce some further features of the waggle page...

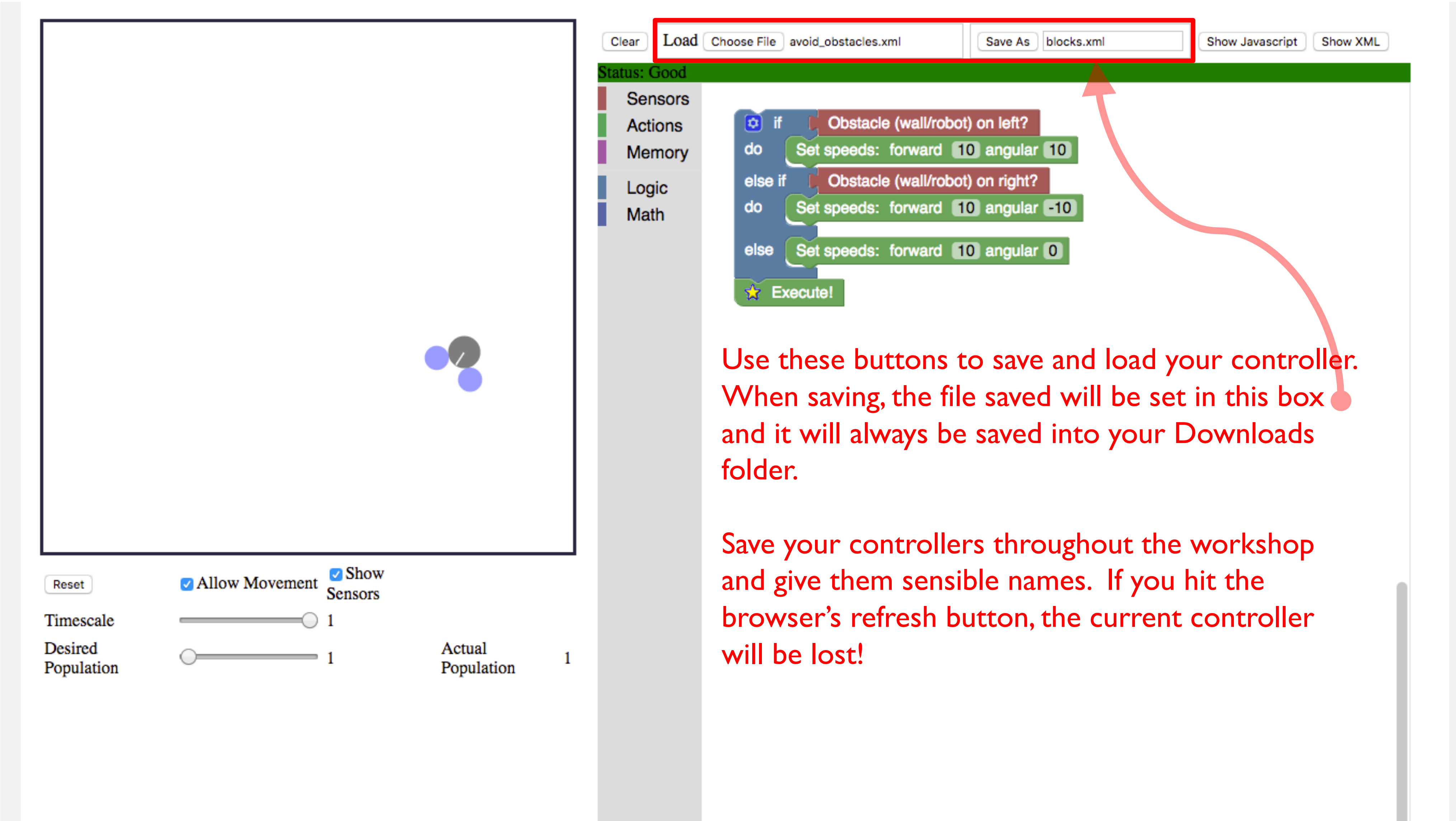


Try adding more robots with this slider



Try adjusting this slider to see the simulation run in slow-motion

NOTE: Adjusting "Timescale" breaks up movement into smaller increments; This changes the sim's behaviour and is therefore not exactly the same as "slowing time".

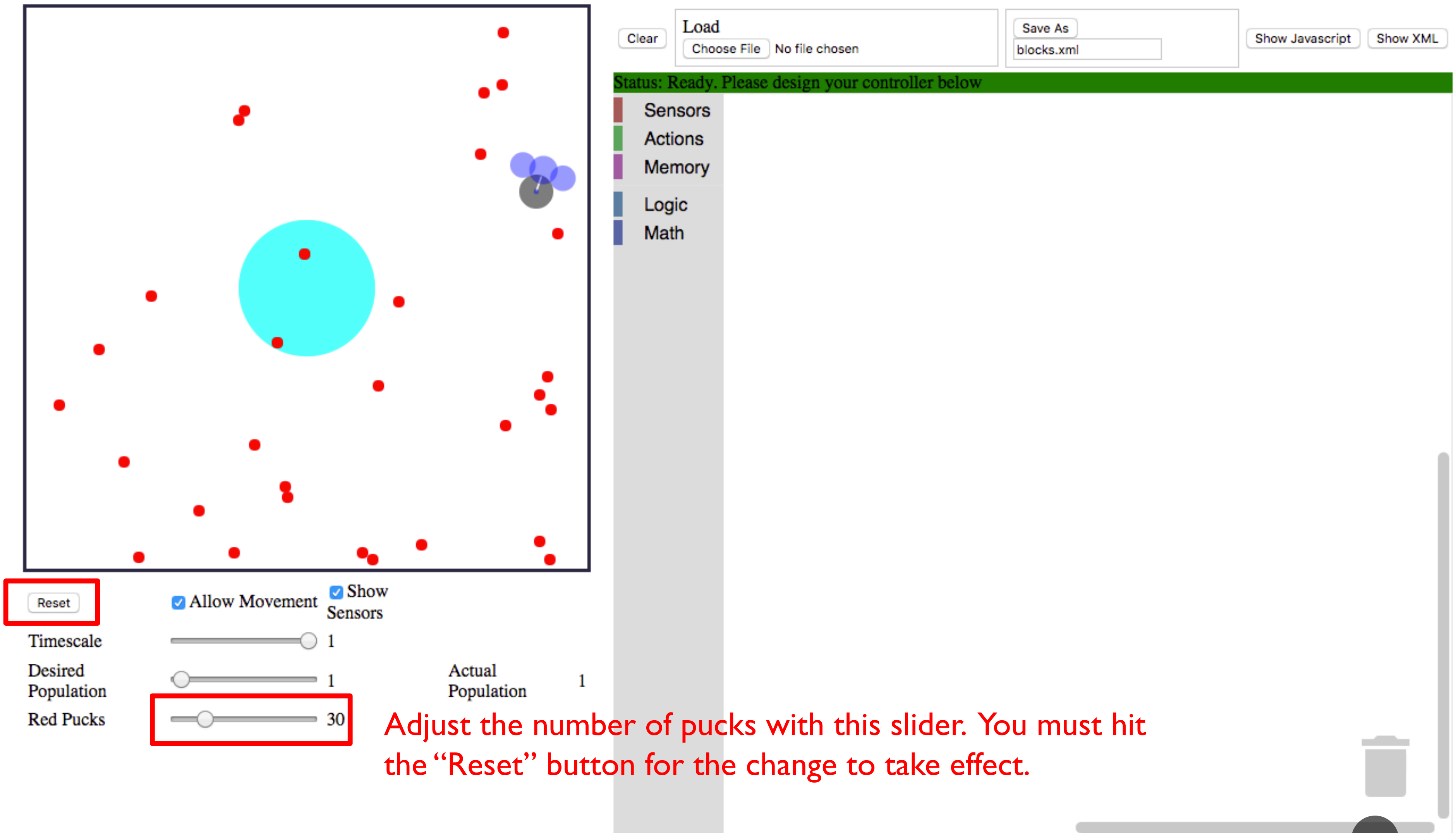


Use these buttons to save and load your controller. When saving, the file saved will be set in this box and it will always be saved into your Downloads folder.

Save your controllers throughout the workshop and give them sensible names. If you hit the browser's refresh button, the current controller will be lost!

WAGGLE FEATURE / BLOCK TOUR

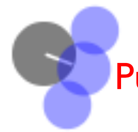
- Our next major topic is **Object Clustering**
- Prior to introducing the swarm robotics perspective on this, we will take a tour through the features and blocks needed for clustering
- Go to the main Waggle page (or just hit your browser's back button):
  - http://bots.cs.mun.ca/waggle/
- Select the **Pre-clustering** level



Adjust the number of pucks with this slider. You must hit the “Reset” button for the change to take effect.

SENSORS

- Sensors are attached to the robot in different configurations; Here is the configuration we will focus on for now:



- Note that the obstacle sensors cannot sense pucks and the puck sensor cannot sense obstacles
- When a sensor detects its target object, the sensor is shaded red, otherwise it remains blue

THE GRIPPER

- The robot can grasp a sensed puck by creating a virtual spring between itself and the puck

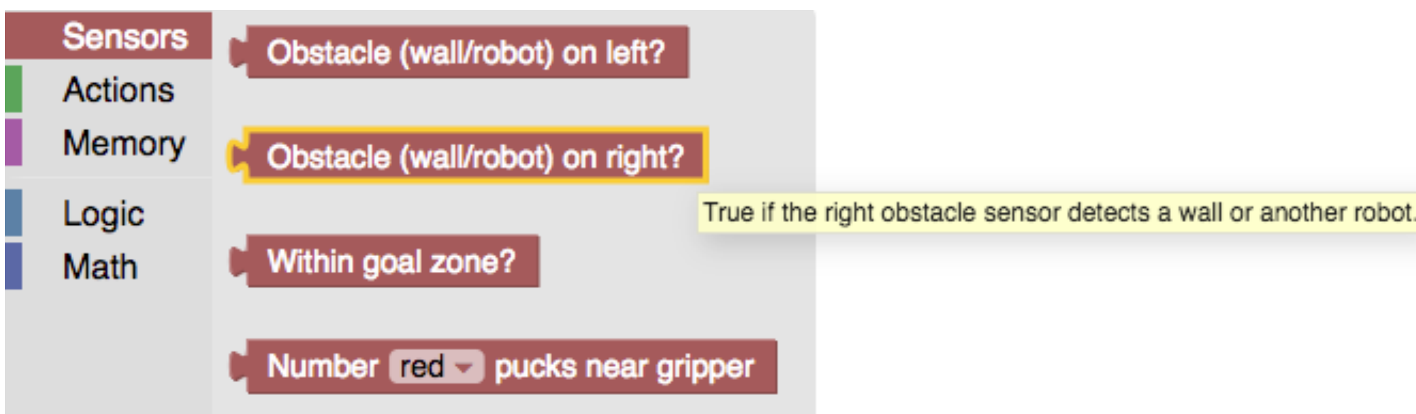
- Note that the sensor does not detect a grasped puck

DEMO: PUCK COUNT

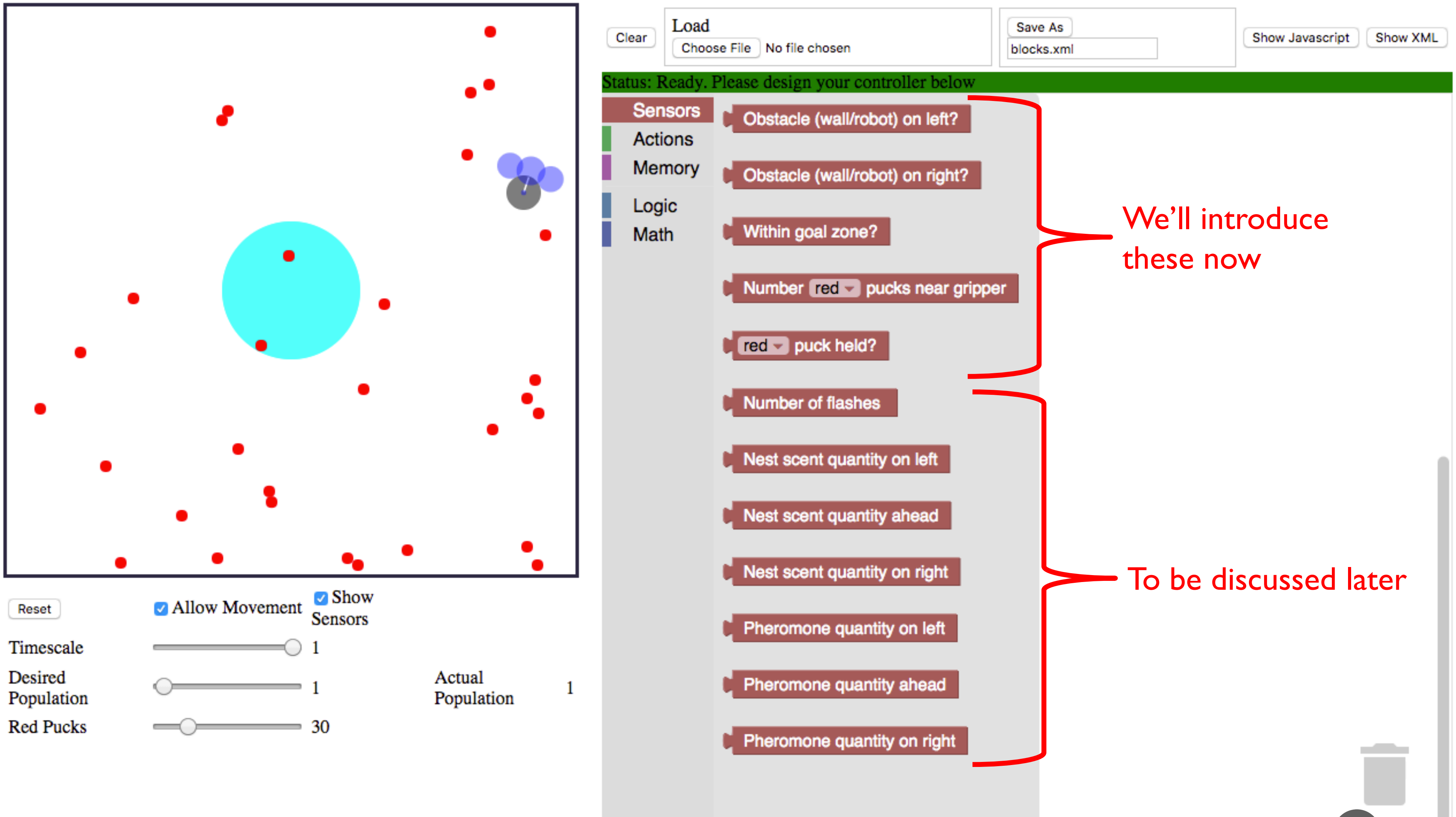
- The main Waggle page has demo controllers that you can load:
  - http://bots.cs.mun.ca/waggle/
- Download `grip_and_count.xml`
- Now go into the Pre-clustering level, click on "Choose File" and load it
- Drag the robot around; As soon as it comes close to a puck it will grip it
- Notice how it counts (at least up to 4) the number of pucks within sensor's radius, but not including the puck held

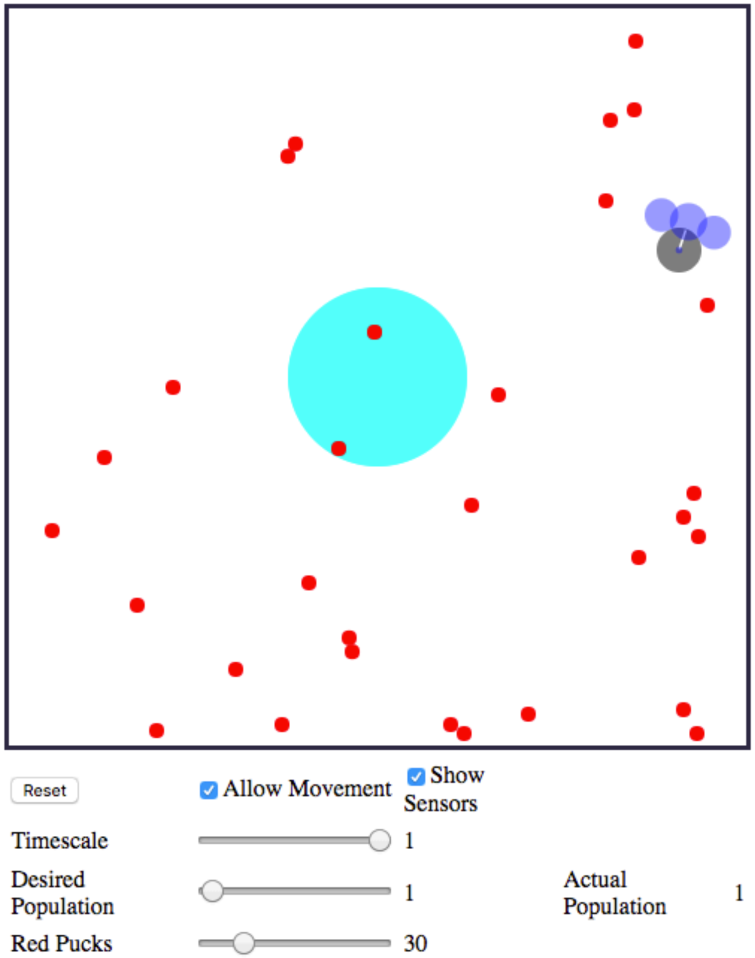
BLOCKLY CATEGORIES

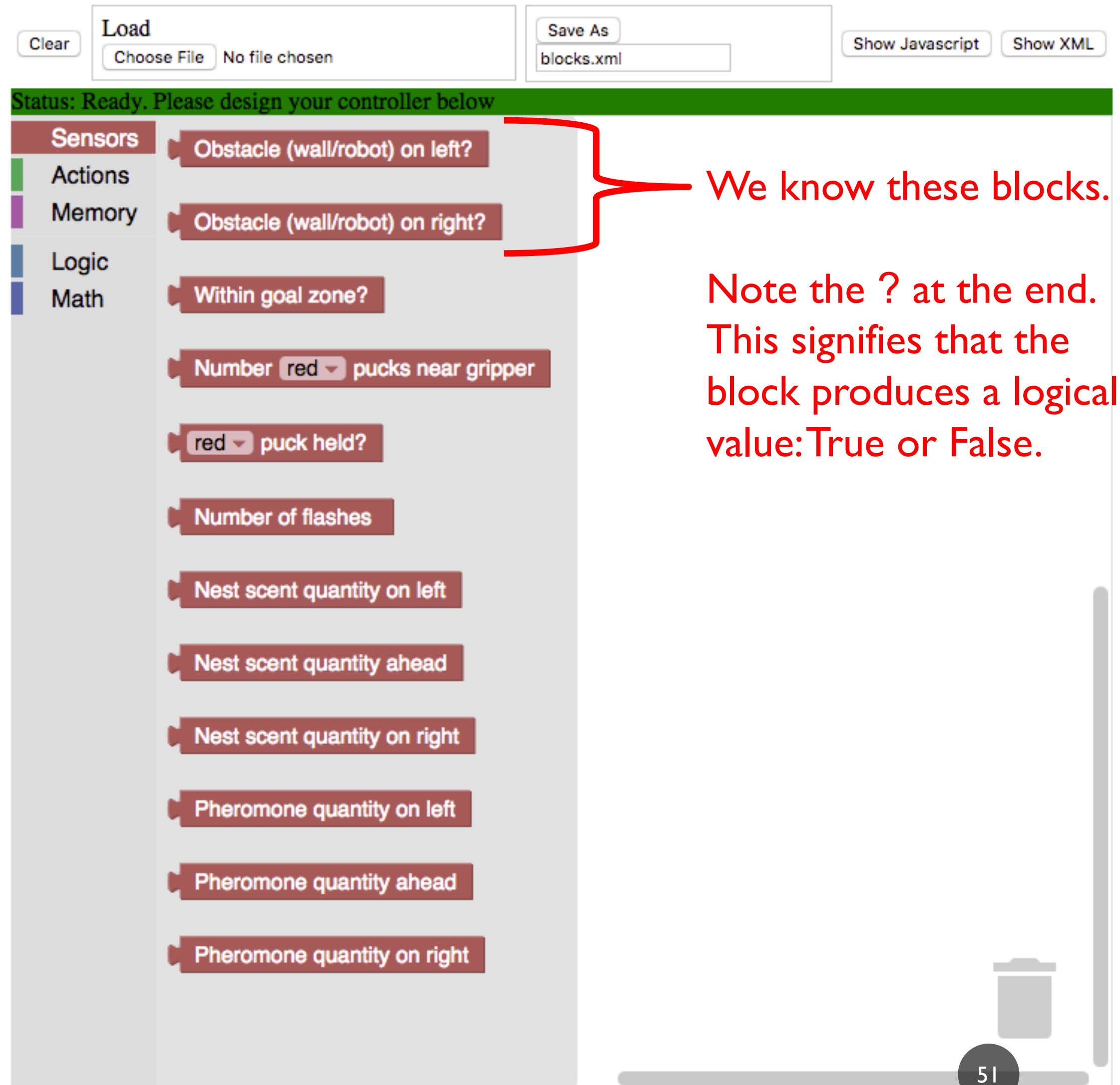
- We will now tour through the various blocks available
- You can always get a helpful pop-up on any block by hovering your mouse above it:



“SENSORS” CATEGORY

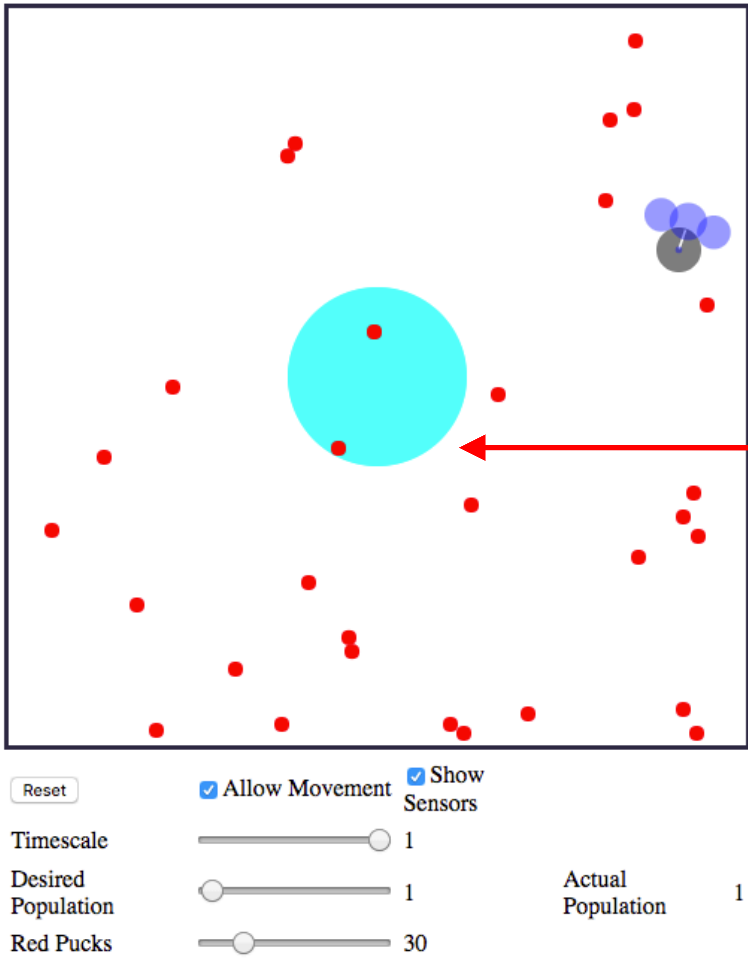

Reset
Allow Movement
Show Sensors
Timescale
1
Desired Population
1
Actual Population
1
Red Pucks
30
Clear
Load
Choose File
No file chosen
Save As
blocks.xml
Show Javascript
Show XML
Status: Ready. Please design your controller below
Sensors
Actions
Memory
Logic
Math
Obstacle (wall/robot) on left?
Obstacle (wall/robot) on right?
Within goal zone?
Number red pucks near gripper
red puck held?
Number of flashes
Nest scent quantity on left
Nest scent quantity ahead
Nest scent quantity on right
Pheromone quantity on left
Pheromone quantity ahead
Pheromone quantity on right
We'll introduce these now
To be discussed later

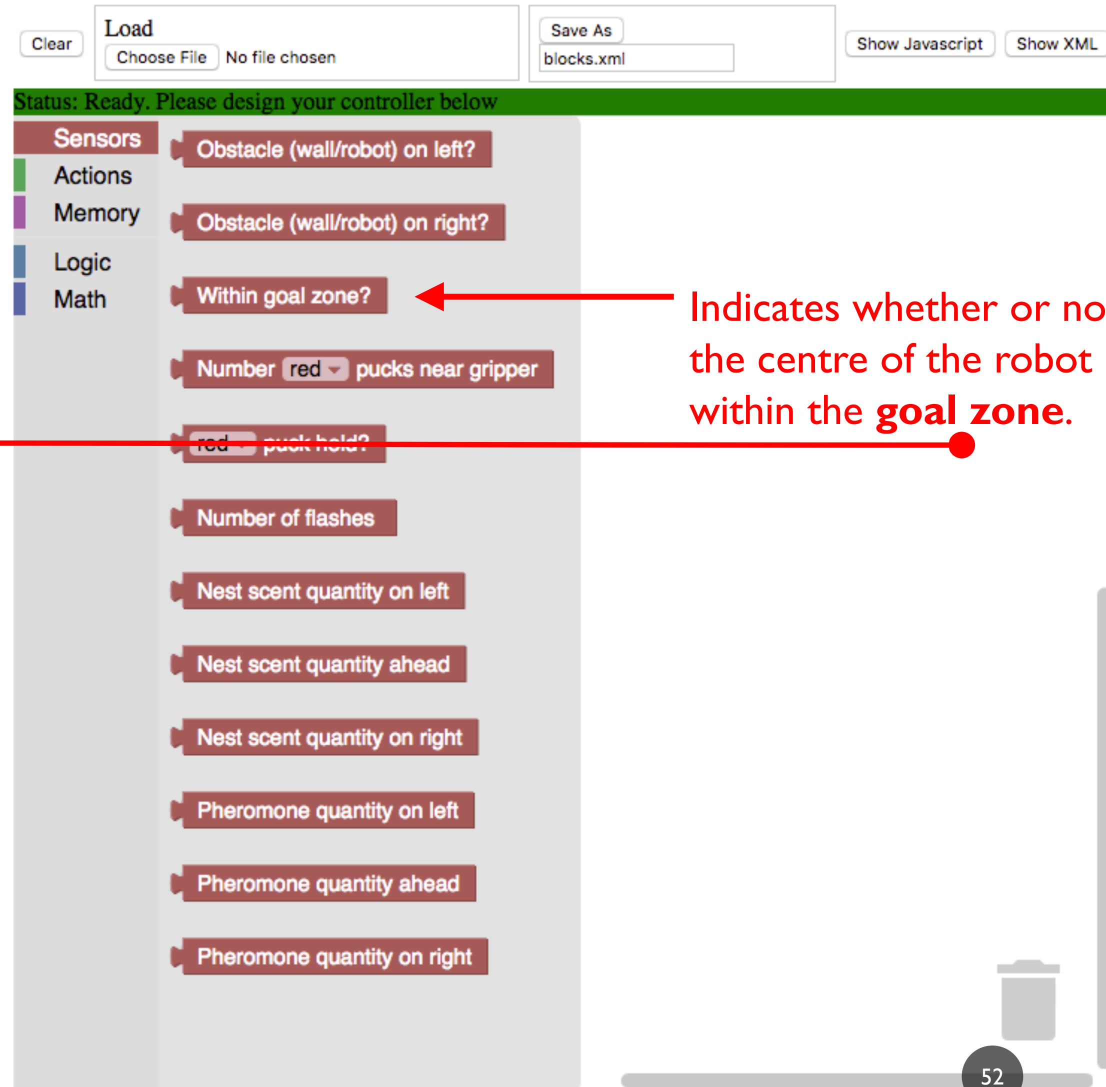




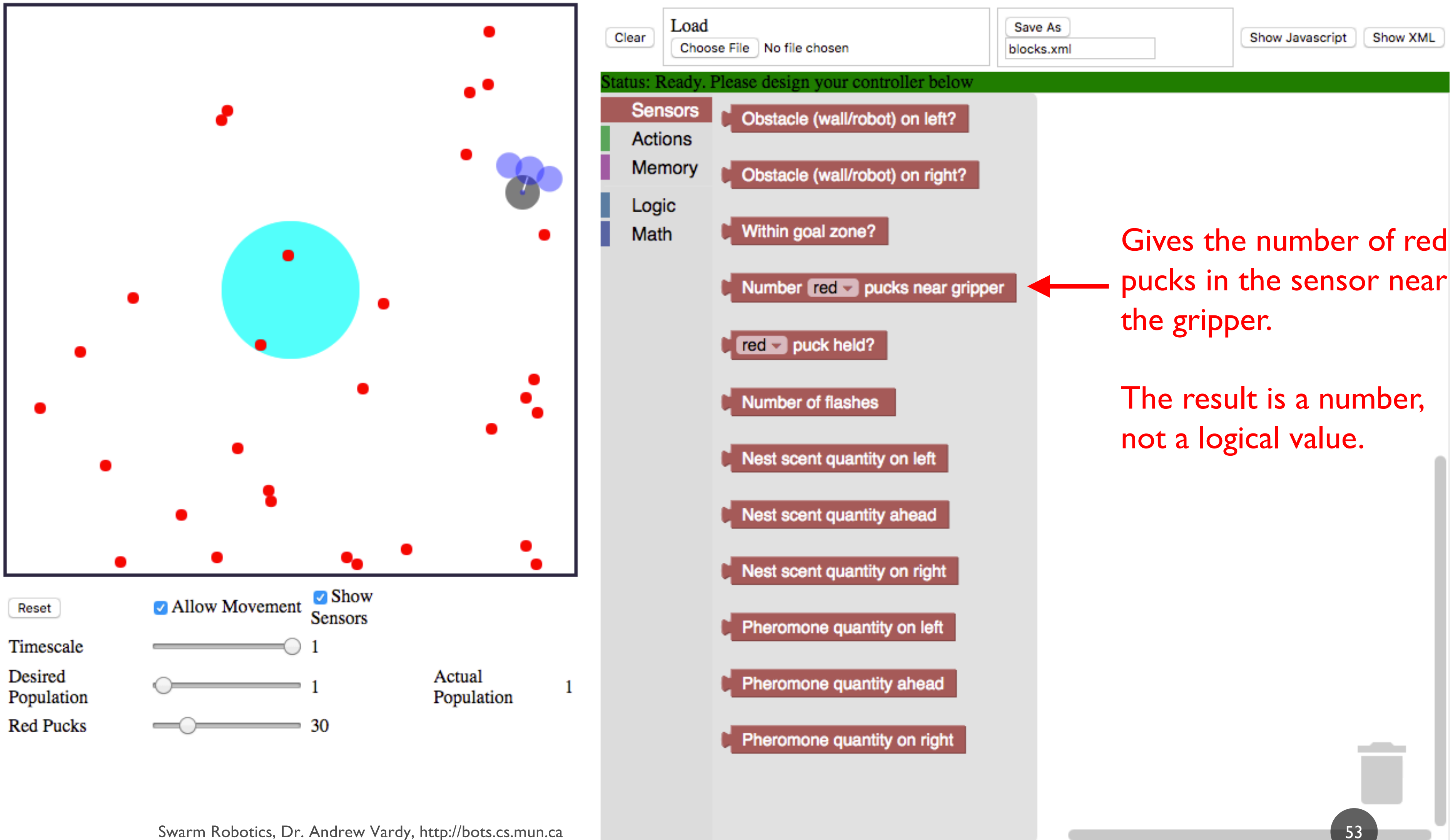
We know these blocks.

Note the ? at the end. This signifies that the block produces a logical value: True or False.



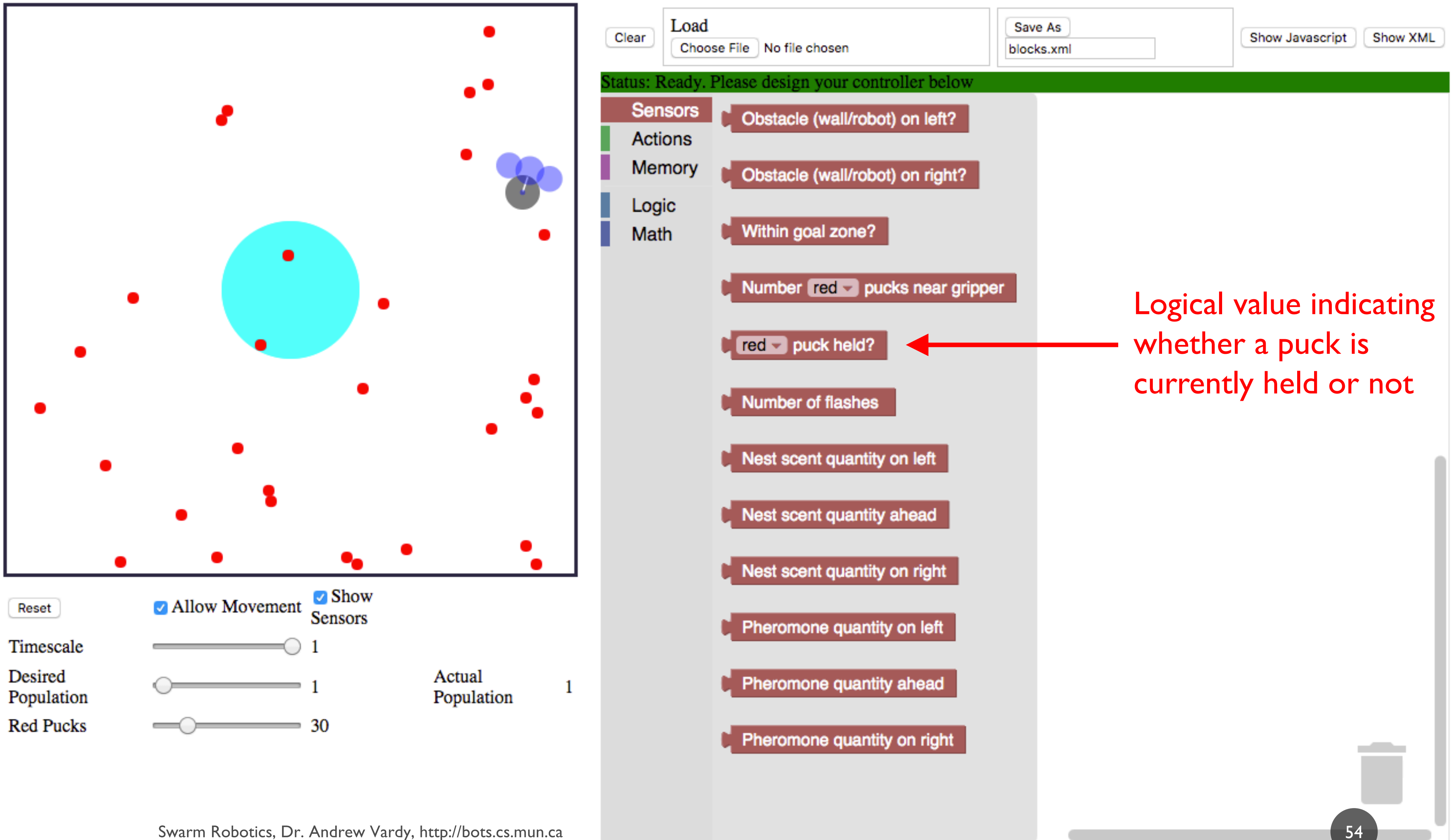


Indicates whether or not the centre of the robot is within the **goal zone**.



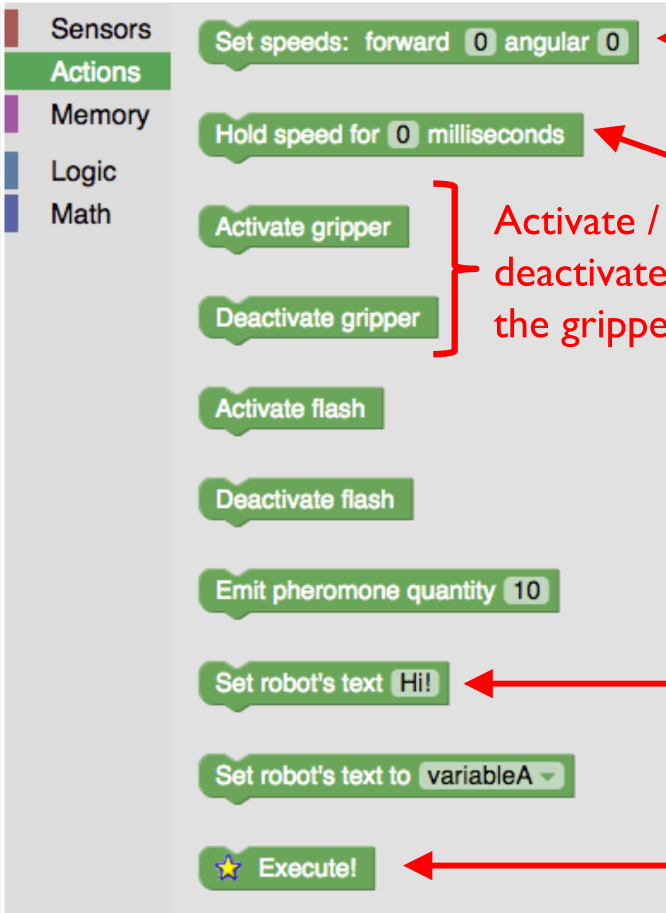
Gives the number of red pucks in the sensor near the gripper.

The result is a number, not a logical value.



Logical value indicating whether a puck is currently held or not

“ACTIONS” CATEGORY



Sets forward and angular speeds (as we've seen)

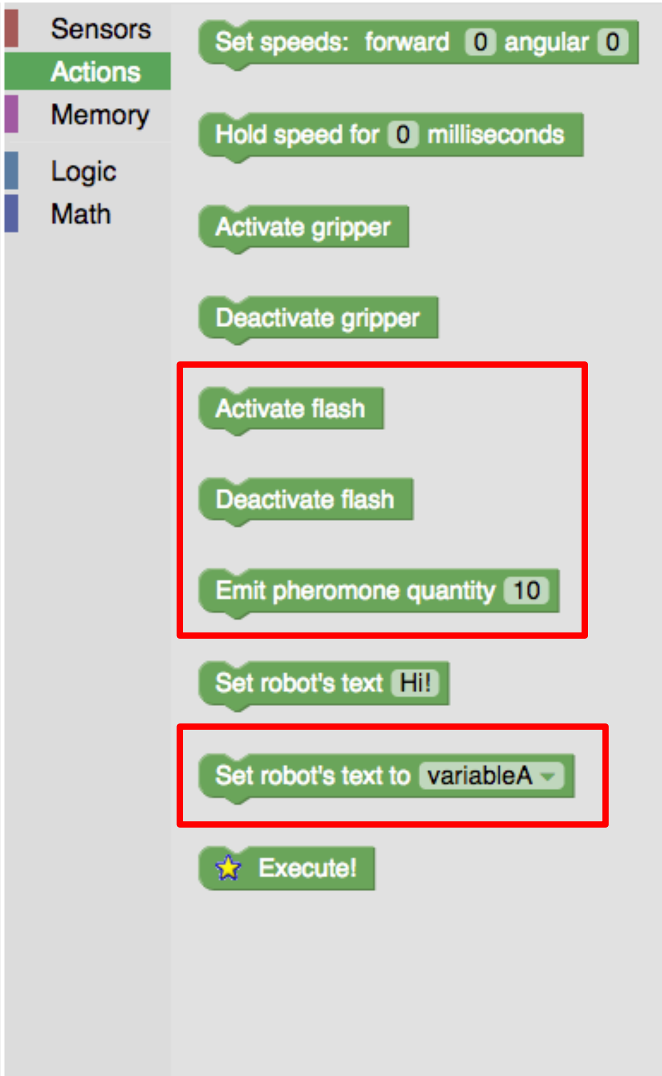
After "Execute", holds the last commanded speed for this number of milliseconds. Very useful for extended actions such as backing away from something or turning by a desired angle.

Note that while a speed is held, the controller is essentially frozen and cannot receive sensor data.

Activate / deactivate the gripper

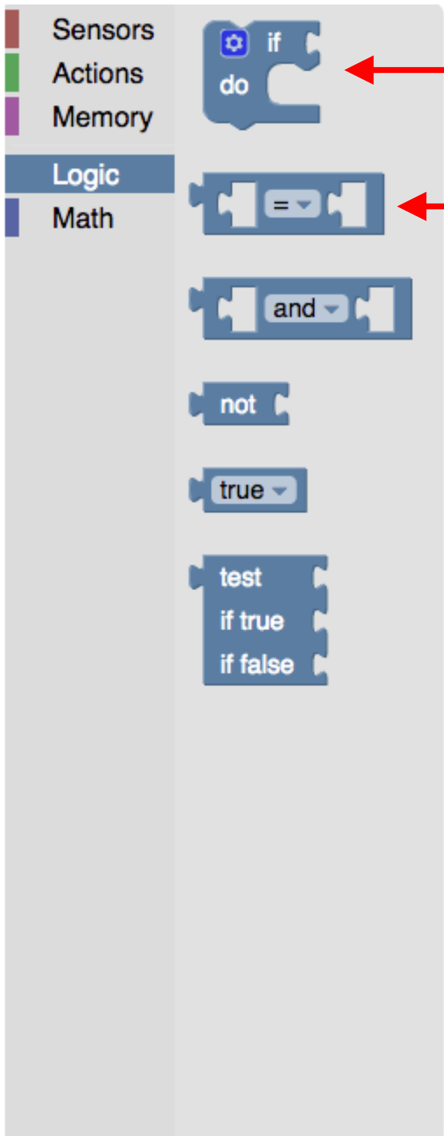
Displays the given text on top of the robot

Causes the controller to execute for one time step



Remaining action blocks will be described later

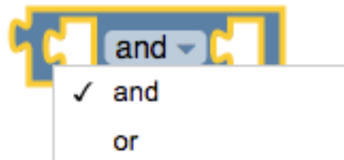
“LOGIC” CATEGORY



The conditional block. Configure by clicking star. **Super important!**

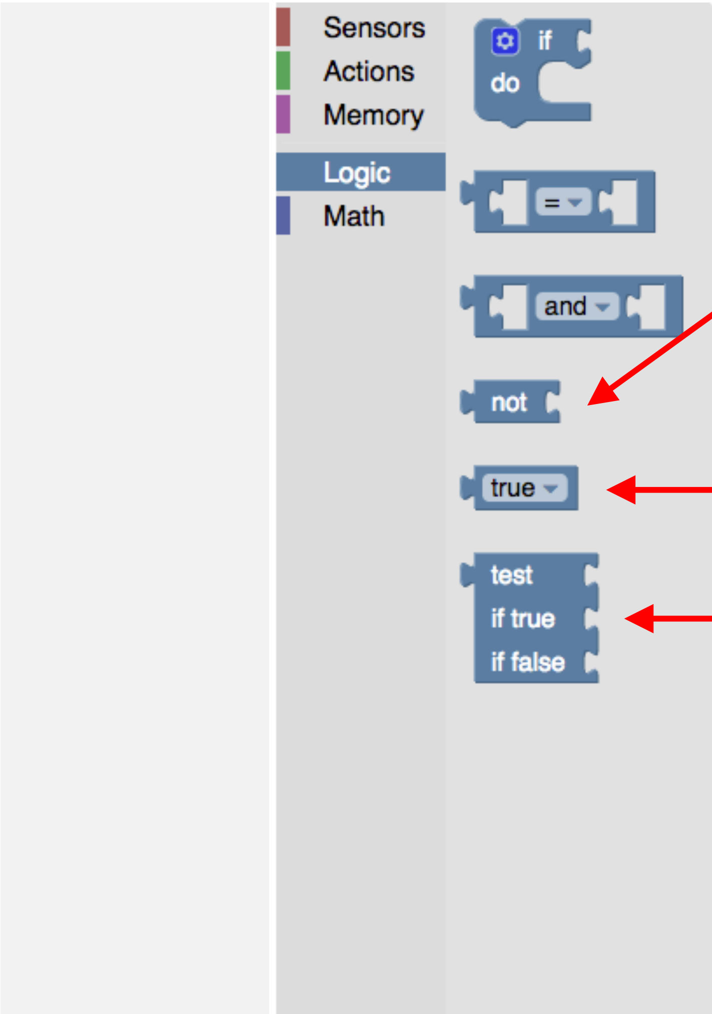
Relational operator. Compares the two number blocks and produces a logical value. By default tests numbers for equality, but this can be configured:

Compares logical values. The two slots are for the logical blocks to compare. By default, produces True only if both values are True, but this can be configured:



and: True only if both inputs are True

or: True if one or both inputs are True



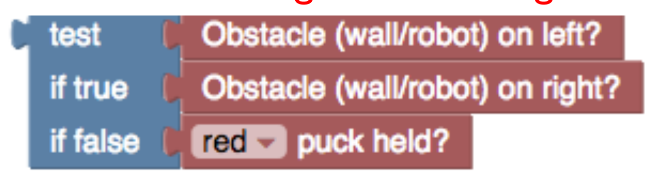
Negates a logical value:

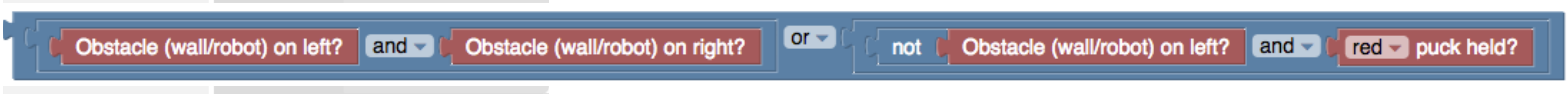
- True becomes False
- False becomes True

Always produces True (or False if selected)

(I've never used this... but you might like it)

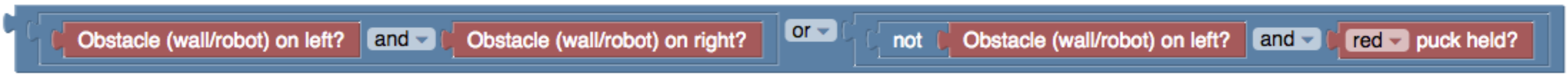
If the "test" condition is true, returns the same logical value as the "if true" condition. Otherwise, returns the same logical value as the "if false" condition. e.g. the following commands are equivalent:



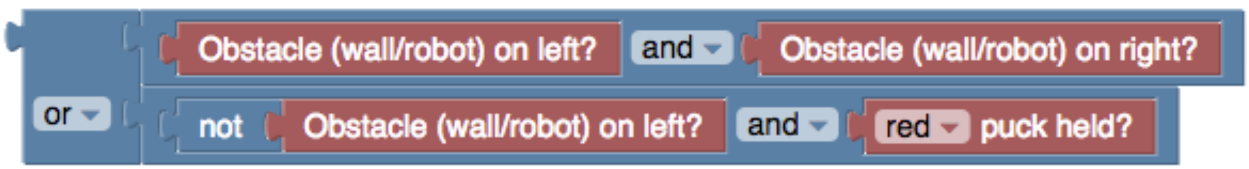


BOOLEAN EXPRESSIONS

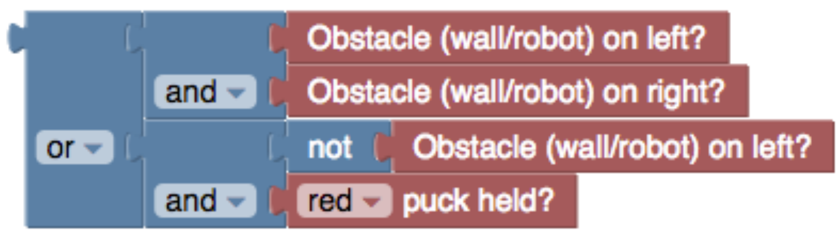
- You will sometimes need to form long expressions like this:



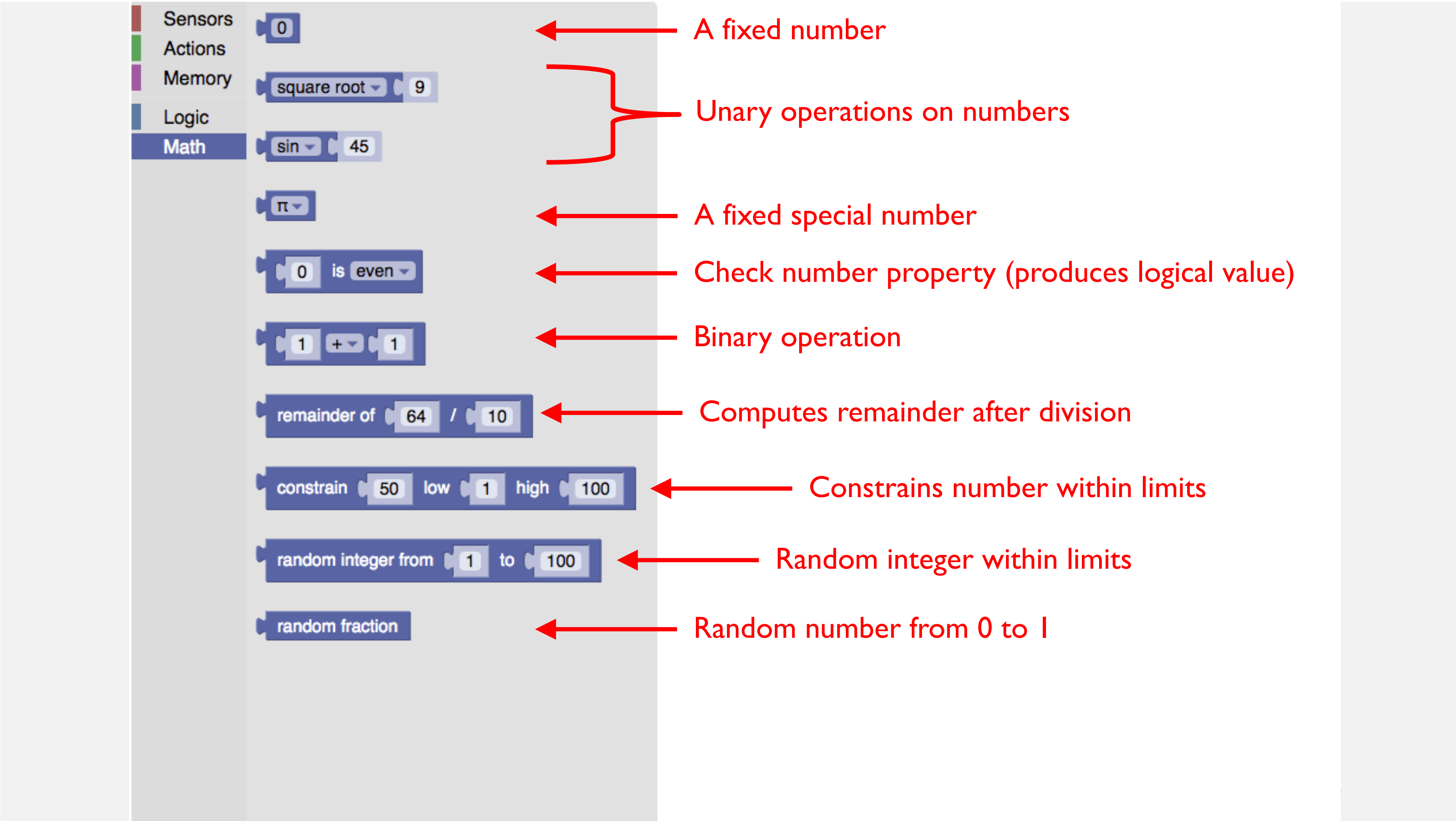
- By right-clicking and selecting "External Inputs" you can re-format like so...



- Applying "External Inputs" to the sub-blocks we get to this form...



“MATH” CATEGORY


Sensors
Actions
Memory
Logic
Math
0
square root 9
sin 45
π
0 is even
1 + 1
remainder of 64 / 10
constrain 50 low 1 high 100
random integer from 1 to 100
random fraction
A fixed number
Unary operations on numbers
A fixed special number
Check number property (produces logical value)
Binary operation
Computes remainder after division
Constrains number within limits
Random integer within limits
Random number from 0 to 1

EXERCISE #4:

DON'T BUMP THE PUCKS!

10 MINUTES

Conditions:

1 robot

- Start with the obstacle avoidance controller:

- In the pre-clustering level create/load this controller; Increase the number of pucks to 100 and hit "Reset"
  - The robot bumps into the pucks
- Your goal is to create a controller that avoids the pucks, bumping into them as little as possible

EXERCISE #4:

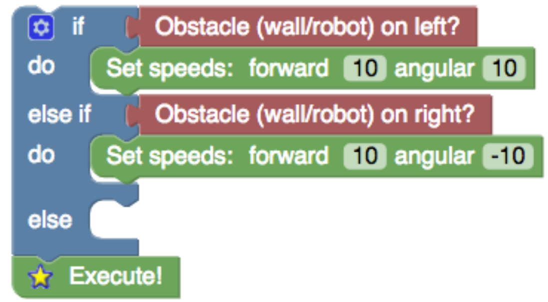
DON'T BUMP THE PUCKS!

10 MINUTES

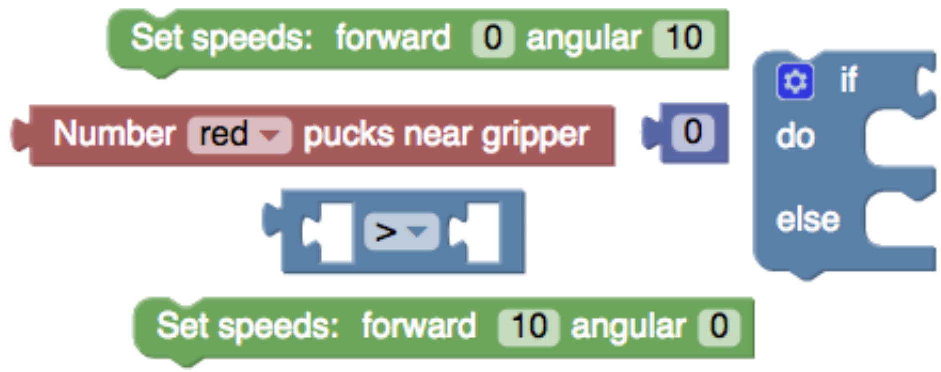
Conditions:

1 robot

- Modify the controller as follows:



- Rearrange the following blocks to go into the empty slot:



- The resulting controller won't be perfect, but it should collide with fewer pucks

EXERCISE #4:

DON'T BUMP THE PUCKS!

10 MINUTES

Conditions:

1 robot

- Your controller will still probably collide with the occasional puck
- Try rearranging your controller and tuning parameters to see if you can further reduce collisions
- Experiment with the use of "Hold speed" which keeps the robot going at the commanded speed for the given number of milliseconds

EXERCISE #5 / A6, TASK 1

FORAGING

10 MINUTES

Conditions:

10 or more robots

- Write a new controller that wanders about, looking for pucks. Upon encountering a puck it should pick it up. If a robot carrying a puck encounters the goal zone, it should drop it.
- You will need the following blocks (along with others already seen):

- CHALLENGES:
  - Robots may collide with pucks already in the goal zone. How can we handle this?
  - If using fewer robots, they may not explore the environment fully. How can we handle this?