How Do You Make a Program Wait?

How Do You Make a Program Wait? Pre-Quiz

1. **What is an algorithm?**

2. **Can you think of a reason why it might be inconvenient to program your robot to always go a precise distance?**

3. **What is a stimulus? Can you think of a stimulus the robot could detect? What sensor would it use to detect it?**

How Do You Make a Program Wait? Pre-Quiz

1. What is an algorithm?

   An *algorithm* is a clear and specific procedure for solving a problem in a finite number of steps.

2. Can you think of a reason why it might be inconvenient to program your robot to always go a precise distance?

   If the distance is very far, this could take up a lot of time. Inconsistencies in the robot itself (power, battery charge) can also cause the robot to not always go the exact distance you program it.

2. What is a stimulus? Can you think of a stimulus the robot could detect? What sensor would it use to detect it?

   A *stimulus* is something that is sensed by a robot or animal and causes it to act. For the purposes of the lesson, it is sensed by the robot and this may cause it to act in a different manner.

   Example stimulus & sensor: The robot could detect a wall in front of it with a touch sensor or an ultrasonic sensor.

Day 1: Programming Using Wait Blocks

(50 minutes)

Objective

To learn to use conditional commands.

In today's lesson, we will investigate:

- Why it is helpful to use conditional commands in programming
- How to use wait blocks to program a LEGO robot to respond to the presence of a *stimulus*

Review: What is an algorithm?

- An *algorithm* is a clear and specific procedure for solving a problem in a finite number of steps.
- *Addition algorithm*: A systematic process that always produces the correct answer when numbers are added:

$$\begin{array}{r} {\scriptstyle 1\ \ 1\ \ \ } \\ 1\,2\,3 \\ \underline{+\,7\,8\,9} \\ 9\,1\,2 \end{array}$$

- Do This: For worksheet question 1, try the algorithm (as above) for adding these two numbers:

    345 + 176 = _______

Did the algorithm result in the same answer for everyone?

Programming as an Algorithm

- Programming is designing an algorithm to solve a problem.
- You need to have commands that are clear and precise because the robot will follow them *exactly*.
- Each step is important. If you make an error in any step, the robot makes that same error!
- Do This: *Write down the detailed steps for the addition algorithm you just performed.* For worksheet question 2, start with step 1, “Write both numbers one above the other,” and so on.
- Let’s discuss what everyone has written down so we understand how the algorithm can be written down in steps so that a computer could execute them.

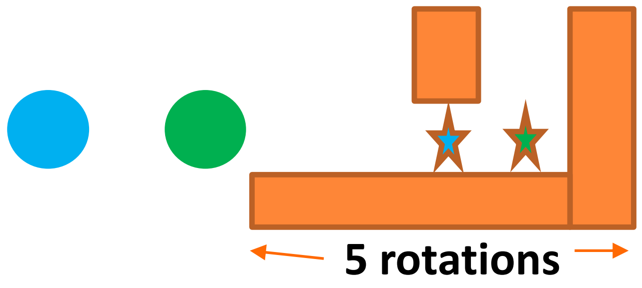
Designing Good Algorithms

- Good algorithms should be *flexible*.

- Algorithms are useful because they can be used to solve many similar problems, not just a specific problem.
- For example, the addition algorithm gives the correct sum—no matter what numbers are added.

Designing Good Algorithms (continued)

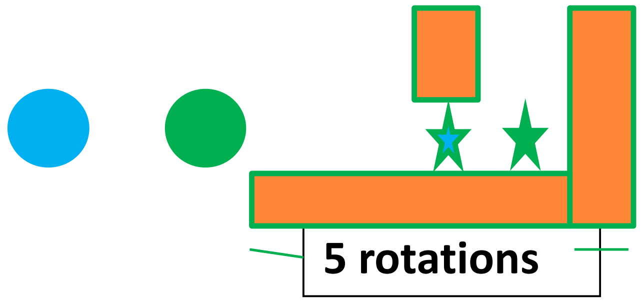
- So far, we have only learned about programming in terms of exact distances.
- Imagine we are programming the robot to go through a maze. We could achieve one step of this by saying "I want my robot to go five rotations forward then turn left."



- But what happens if we start the robot a little too far back (at the blue circle instead of the green)? The robot will turn too early (at the blue star instead of the green)!
- Do This: Describe such problems on worksheet question 3.

Designing Good Algorithms (continued)

- As a result, that program is not very helpful because it will only work in one specific case.
- But what if we could program the robot to turn when it senses the wall? The robot could start from any distance and still make the turn at the correct time.
- In other words, the algorithm we design would work in a variety of situations, which means it would be far more useful.



Conditional Commands

- It turns out we can design this kind of algorithm by using *conditional commands*.
- A conditional command is a command whose action depends on a condition being satisfied.
- Many types of conditional commands exist.
  - *Example*: If you see a stop sign, then stop!
  - This is conditional because the action (stop!) depends on a condition (seeing a stop sign).

Conditional Commands (continued)

- We will be focusing on conditional statements using until. Example: “Play at recess until you hear the bell ring. Then go back to class”
- Notice, that you do not have to keep track of exactly how long you can play at recess; you will know to go to class when you hear the bell ring.

1. Play at recess
2. Wait until you hear the bell
3. Go to class

- Do This: Write a sequence of similar steps to tell a robot to stop when it bumps into a wall. (worksheet question 4)
- Let’s discuss what each of you have written down.

Conditional Commands (continued)

Maze Demonstration

Let’s do a simple demo of how it is easier to use conditional commands than specific commands.

- **Do This:** Set up a simple maze: Use tape on the floor to mark off a 2-foot wide track that ends at a wall 5 feet away.
- Blindfold a student volunteer to serve as the “robot.”
- Have another student volunteer command the “robot” through the maze, making a left turn at the end when the route terminates at the wall.
  1. First try using only one of the two commands at a time: “move forward (or backward) X steps” or “turn left.”
  2. Then try using only one of the two commands at a time: “move forward with your hands stretched UNTIL you sense the wall” or “turn left.”

Conditional Commands (continued)
Maze Demonstration

As a group, discuss your thoughts:

- How did using the conditional command change the robot's success with the instructions?
- Was the robot volunteer able to move faster with the conditional command?
- Note that the student volunteer used his/her sense of "touch" to navigate the maze. If he had used his "eyes" (light sensor), then the task could have been performed much faster!
- Likewise, a robot could use a variety of sensors to perform the same task.
- During the next class, we will learn about programming the LEGO taskbot using conditional commands.

Day 2: Programming with Wait Blocks

(50 minutes)

Today, when programming your robot, use the conditional command ideas we discussed during the last class. If you want to make your robot keep moving forward and stop when it bumps into a wall, add a touch sensor to it and use the conditional command "until."

The robot should go forward until the touch sensor hits the wall. Then the robot should stop.

1. Move forward
2. Wait until the touch sensor is pressed
3. Stop

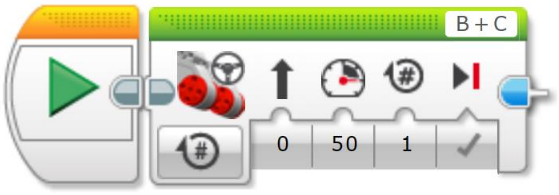
Notice that now the robot does not have to keep track of how far it should move forward!

Next, we will develop an NXT program to implement this.

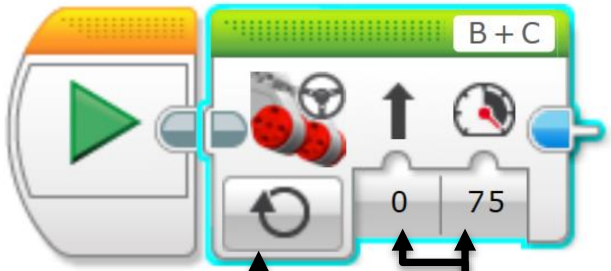
Programming with Wait Blocks (continued)

1. Move forward

- **We already know which block to use for the first command – a Move Steering block.**
- **We want the direction to be forward, and the steering to be in the middle.**



- **But we have a problem:** ***What should we set the duration to?***
- **We don't know exactly how many rotations or seconds it will take the robot to hit the wall.**
- **So, let's select "ON" from the drop-down menu for unlimited turn.**



Set to "ON"
for unlimited turn.

Set steering to 0, and speed to 75.

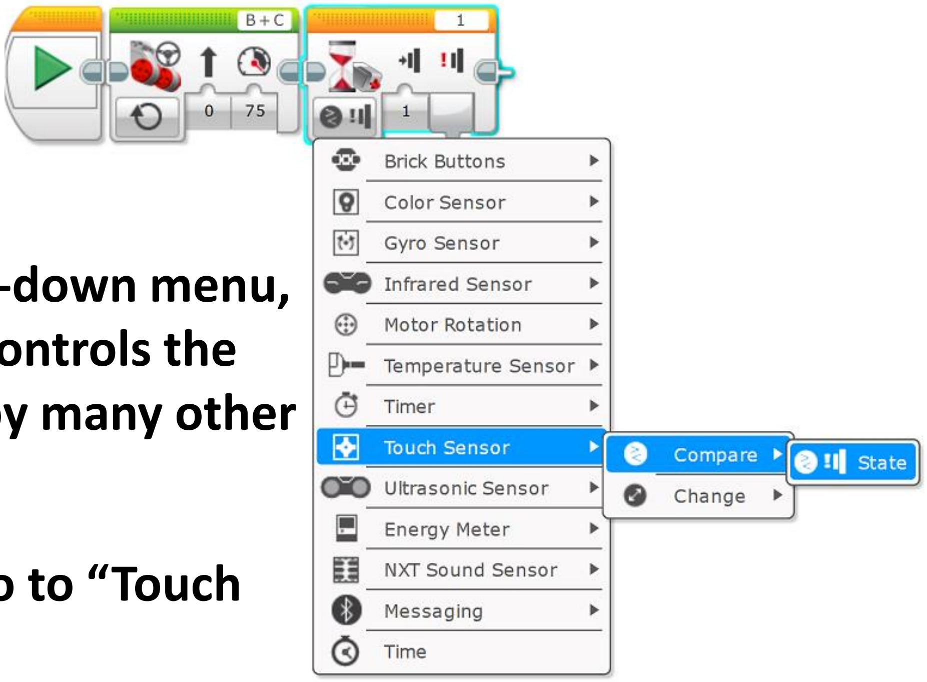
Programming with Wait Blocks (continued)

2. Wait until the touch sensor is pressed

- **For the second part of the program, we need to use a new kind of block – the wait block.**
- **Wait blocks tell the robot to wait until a specified *stimulus* occurs before going on to the rest of the program.**
- **A *stimulus* is an action that can be perceived by the robot that causes it to move on to the next part of the program.**
- **In this case, we want to wait until the wall presses in the touch sensor (when it bumps into it) and then stop the robot.**
- **So, the touch sensor being pressed is our stimulus.**

Programming with Wait Blocks (continued)

- The wait block is orange in color and has an hour-glass icon on it. Drag it into your code right next to the "Move Steering Icon".



- Notice that if you click on the drop-down menu, you can change the stimulus that controls the wait block to an action perceived by many other sensors.

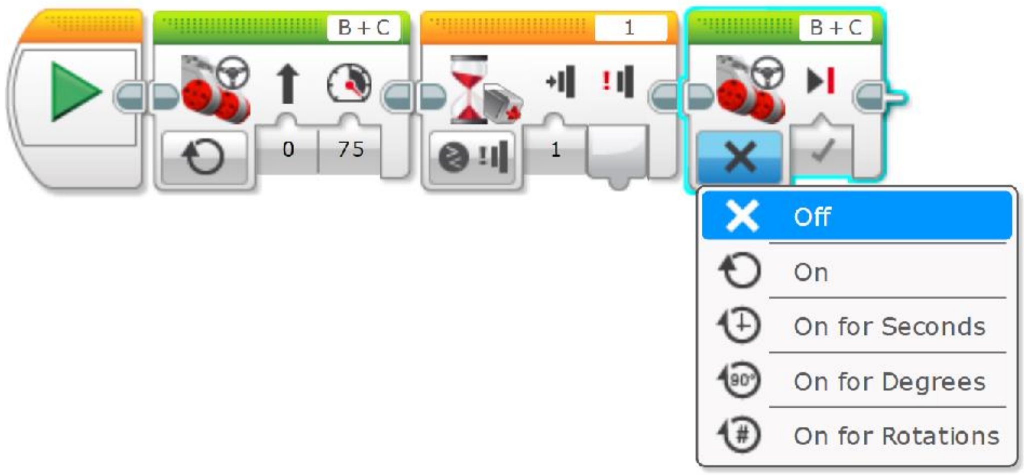
- Next, from the drop down menu, go to "Touch Sensor"→ "Compare" → "State".

- Also notice that the wait block shows the touch sensor connected to Port 1. Make sure that the touch sensor is connected to Port 1 on your robot (You can always change it).

Programming with Wait Blocks (continued)

3. Stop

- Once the touch sensor has been pressed, we want the robot to stop.
- We can achieve this by simply dragging a Move Steering block and setting the option to "OFF".
- Our program is now complete: *The robot will go forward for unlimited and wait until the touch sensor is pressed. When this happens, the robot will stop and the program will end.*
- Notice that nowhere in the program did we tell the robot how far it needed to go forward until it hit the wall.



Review: How to Use Wait Blocks

- Try to break each part of the task you are trying to achieve into a conditional statement using "until" followed by "then." (*In the example*: The robot should go forward until the touch sensor hits the wall. Then the robot should stop.)
- Whatever comes before the "until" should come before your wait block. (*In the example*: "The robot should go forward.")
- The wait block should depend on the condition following "until." (*In the example*: The touch sensor hits the wall.) We program this by dragging down a wait block and selecting the correct sensor from the dropdown menu.
- The move block that comes before the wait block should have duration set to UNLIMITED. This makes the robot keep moving until it is told to stop or change direction.
- Whatever follows "then" should come after the wait block.

Implementing the Simple Program

- **Do This:** **Complete questions 4 and 5 on the worksheet with the logic of the program and make sketches of the EV3 blocks to implement it.**
- **The next task is to implement the program on the LEGO taskbot.**
- **Then complete the worksheet, including listing the program using the EV3 software. Then download it to the computer and check it out.**
- **Discuss your findings as a group.**
  Was navigating the maze much faster and accurate?

How Do You Make a Program Wait? Post-Quiz

1. **What is an algorithm?**

2. **Can you think of a reason why it might be inconvenient to program your robot to always go a precise distance?**

3. **What is a stimulus? Can you think of a stimulus the robot could detect? What sensor would it use to detect it?**

How Do You Make a Program Wait? Post-Quiz

1. **What is an algorithm?**

   An *algorithm* is a clear and specific procedure for solving a problem in a finite number of steps.

2. **Can you think of a reason why it might be inconvenient to program your robot to always go a precise distance?**

   If the distance is very far, this could take up a lot of time. Inconsistencies in the robot itself (power, battery charge) can also cause the robot to not always go the exact distance you program it.

3. **What is a stimulus? Can you think of a stimulus the robot could detect? What sensor would it use to detect it?**

   A *stimulus* is something that is sensed by a robot or animal and causes it to act. For the purposes of the lesson, it is sensed by the robot and this may cause it to act in a different manner.

   Example stimulus & sensor: The robot could detect a wall in front of it with a touch sensor or an ultrasonic sensor.

Vocabulary

algorithm: A clear and specific procedure for solving a problem in a finite number of steps.

conditional command: A command in which the completion of an action depends on a condition being satisfied.
For example, if I see a stop sign [condition], I stop [action].

stimulus: Something that rouses or incites to activity.
For the purposes of the lesson, it is an action that can be perceived by the robot that causes it to move on to the next part of the program.

Images Sources

Slide 1: Baseball player girl waiting; source: Microsoft® clipart: http://office.microsoft.com/en-us/images/results.aspx?qu=baseball+player&ex=1#ai:MP900422161|mt:2|

Slide 7: Woman doing yoga stretch; source: Microsoft® clipart : http://office.microsoft.com/en-us/images/results.aspx?qu=yoga&ex=1#ai:MC900439917|

Slide 10: Boy holding stop sign; source: Microsoft® clipart: http://office.microsoft.com/en-us/images/results.aspx?qu=stop&ex=1#ai:MP900422690|mt:2|

Device and programming images from LEGO MINDSTORM EV3 User's Guide

Screen captures and diagrams by author