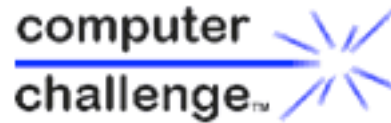


Lego Robot Tutorials

Touch Sensors



Bumper Cars with a Touch Sensor

With a touch sensor and some robot programming, you can make your robot search its way around the room. It can back up and turn around when it hits something, like bumper cars. Go to the Web site to see a robot bumper car in action!

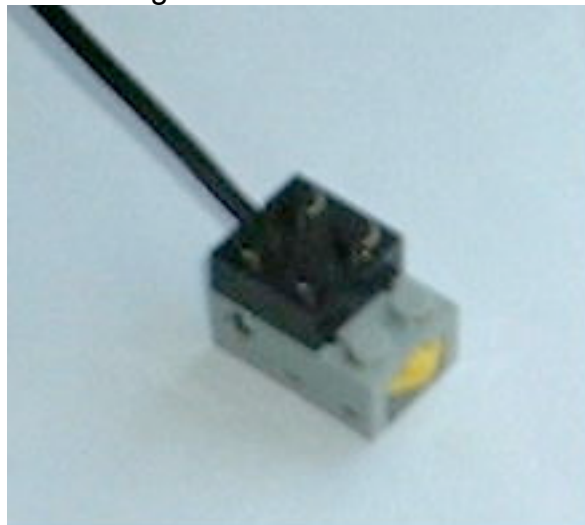


Before starting this activity, you should *Build a Robot* and know how to *Program Your Robot to Move*.

Touch Sensors

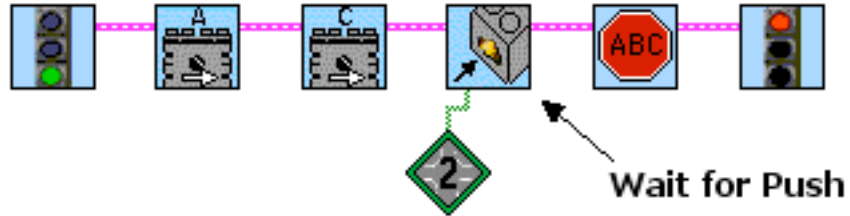
Touch sensors let your robot sense contact. Use a bumper in front of the touch sensor so your robot can sense contact over a larger area.

Touch sensors connect to one of the three input ports numbered 1, 2, and 3 on the RCX. Your robot program can read the touch sensor and determine if the yellow button is pushed in or out. Then the program can take appropriate action, such as backing up and turning around.





Wait for Push

Suppose you want your robot to go forward until it bumps into something, then stop. You can use a touch sensor and the **Wait for Push** function to do this:

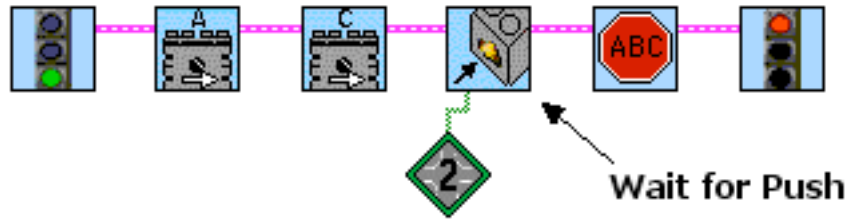




You can read this program as:

- Begin the program
- Start motors A and C forward (go forward)
- Wait until the touch sensor connected to input  is pushed in (keep going until you bump into something)
- Stop all motors
- End the program

You can find the Wait for Push function by clicking the **Wait For**  button on the **Functions Palette**. This brings up the subpalette of wait functions. With wait functions, it is the program that waits at that step, not the robot. The robot keeps going until you tell it to stop.

Set the Input Port

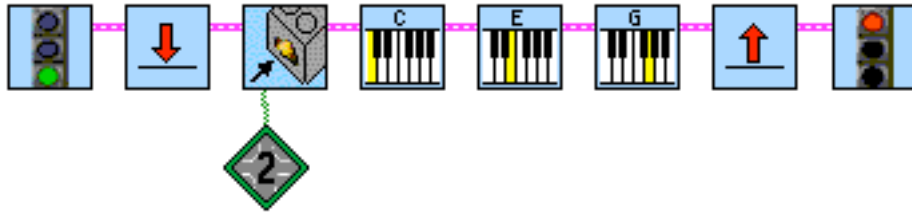


 is a modifier. It modifies the actions of the **Wait for Push** function by telling it which input the touch sensor is connected to. You could also have told it to use input 1 or 3. To find the modifiers, click the **Modifiers** button  on the **Functions Palette**.

If you don't use any modifier, the input *defaults* to input 1. Use Context Help to determine the modifiers for each function, where to connect them, and the default values.


Loops

Suppose you want your program to do the same actions over and over again. For example, you may want your robot to play a musical chord each time the touch sensor is pushed. You do this with a programming structure called a loop:



You can read this program as:

1. Begin the program

2. Start a loop 

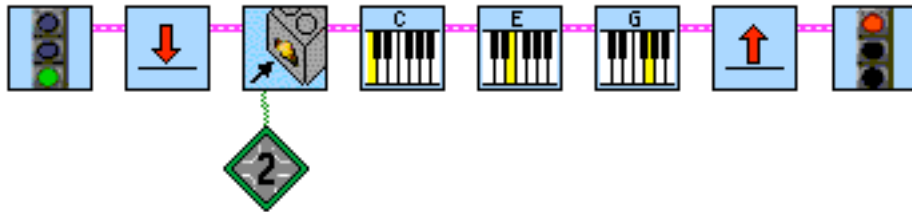
a. Wait until the touch sensor on input 2 is pushed

b. Play the notes C, E, G (a C major chord)

c. Go to step 2 

3. End the program

Loops




When the **Red Jump**  is executed, the program goes to the **Red Land**



as shown by the gray arrow (which is not part of the program). The **Red Land** doesn't do anything itself, so the program just goes to the next step, **Wait for Push**. If the **Land** is before the **Jump** in the program they make a loop, because what is between them executes over and over.

To find the Jumps and Lands, click the **Structures** button  on the

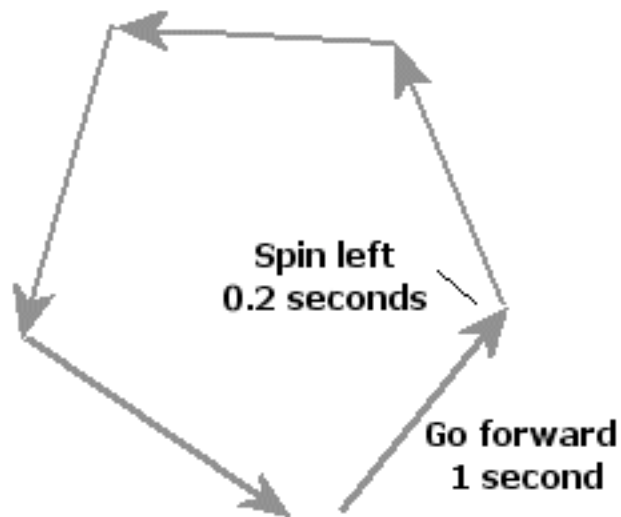
functions palette, then the **Jumps**  button. A **Jump** always goes to the **Land** of the same color. Use other colors if you have more than one pair in a program.

Algorithms

An algorithm is a list of steps for solving a problem, something like a recipe. For example, here is an algorithm that keeps a robot going around in a big circle:

Bumper Car Algorithm

1. Go forward 1 second
2. Spin left for 0.2 seconds
3. Go to step 1

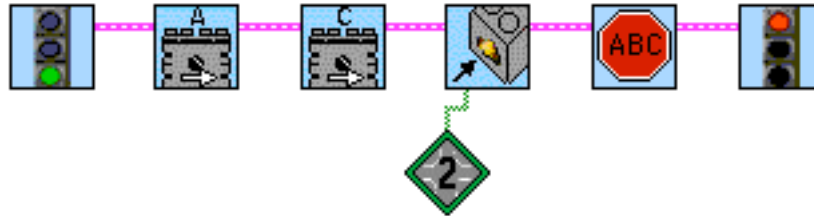


Brainstorming a Bumper Car Algorithm

You need an algorithm in mind before you start writing a program. Otherwise, you don't know what you are trying to do! Can you create your own bumper car algorithm? View the Bumper Car video again . Brainstorm with your teammates to create a list of steps: 1, 2, 3 for what you want your robot to do. After you come up with your own idea, go to page 7 of this tutorial on the Web site and click "possible answer" for one possible solution.

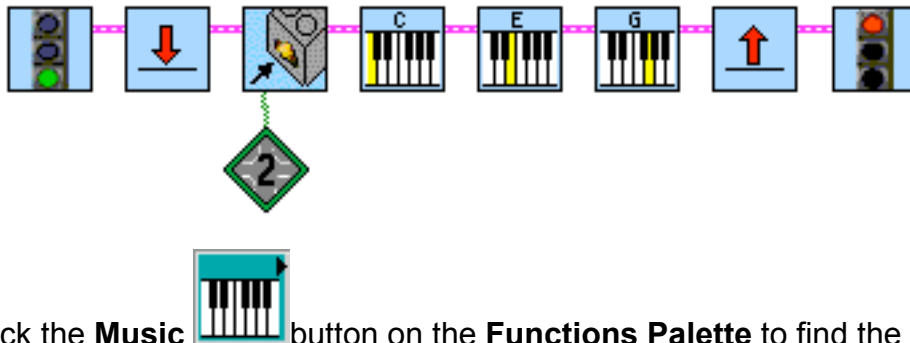
Challenge 1: Bump and Stop


Enter the sample program into RoboLab, download it, and run it. Does your robot stop when it bumps into a wall or other object? If not, check that the touch sensor is connected to port 2 and that the bumper is working.



Challenge 2: Make Beautiful Music

Enter the sample program that plays musical notes each time the touch sensor is pushed. Download and test it. Does it work?




TIP: Click the **Music**  button on the **Functions Palette** to find the music functions.

EXTRA CHALLENGE: For those who are musically inclined, try programming your own song. Use Context Help to explore the different music functions, which include different note lengths, octaves, and recorded music.

Challenge 3: Bumper Cars

Write a bumper car program for your robot. Your robot will go forward until it hits something, then back up, turn, and head off in a new direction.

TIP: Start with the touch sensor music program from Challenge 2. Replace the music functions with motor control functions that *implement* the algorithm you developed.

EXTRA CHALLENGE: Use **Wait for Random Time**  to add randomness to how your robot turns. Use Context Help to learn how to use this function. Grab a numeric constant to set the maximum time to turn.