

ROS Scratch: Enabling Block-Based Robotics

Brian Thomas

Brown University
Department of Computer Science

Robots for Education, May 2, 2011

New Blocks

- New motor command Scratch blocks



- New motor command Scratch blocks

```
(motors %w at %X% power' - broadcastMotors:Power)
(motors left %n right %n' - broadcastMotorsLeft:Right: -100 100)
(motors %w at %X% power for %n sec' t broadcastMotors:PowerSecselapsedfrom: 'forward' 80 1.0)
(motors left %n right %n for %n sec' t broadcastMotorsLeft:Right:Secselapsedfrom: -100 100 1.0)
(motors stop' - broadcastMotors:Stop)
```

robotMotorPowerMenu

```
| menu |
menu = CustomMenu new.
*( ('stopped' 0)
('slow' 25)
('medium' 50)
('fast' 75)
('full power' 100)
) do: [:pair |
menu
add: '( asUTF8. pair second printString. '); pair first localized
action: pair second].
+ menu
```

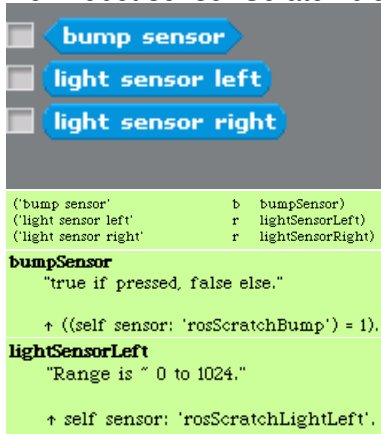
```
{% = code ifTrue: [% ChoiceArgMorph new getOptionSelector: #varNamesMenu; choice: ''].
{% = code ifTrue: [% ChoiceArgMorph new getOptionSelector: #robotMotorDirection; choice:
'forward'].
{% = code ifTrue: [% ChoiceArgMorph new getOptionSelector: #motorDirection].
{% = code ifTrue: [% ChoiceOrExpressionArgMorph new getOptionSelector: #sceneNames; choice:
''].
{% = code ifTrue: [% ExpressionArgMorphWithMenu new numExpression: '80'; menuSelector:
#robotMotorPowerMenu].
{% = code ifTrue: [% ExpressionArgMorphWithMenu new numExpression: '1'; menuSelector:
#listIndexForDeleteMenu].
```

broadcastMotorsLeft: left Right: right

```
self broadcast: 'left ', left storeString, ' right ', right storeString withArgument: 0.
```

New Blocks

- New robot sensor Scratch blocks



The image shows three blue Scratch-style blocks for robot sensors, each with a small square icon to its left. The blocks are labeled 'bump sensor', 'light sensor left', and 'light sensor right'. Below the blocks is a code block with the following content:

```
('bump sensor'          b  bumpSensor)
('light sensor left'    r  lightSensorLeft)
('light sensor right'   r  lightSensorRight)

bumpSensor
  "true if pressed, false else."

  ↑ ((self sensor: 'rosScratchBump') = 1).

lightSensorLeft
  "Range is ~ 0 to 1024."

  ↑ self sensor: 'rosScratchLightLeft'.
```

Making New Scratch Functionality

- Entire filesystem is an image
- Code in system browser

System Browser			
Scratch-Objects	CameraMedia	-- all --	broadcastMotors:Power:
Scratch-Blocks	FilterPack	initialization	broadcastMotors:Power:Se
Scratch-Execution Engine	ImageMedia	accessing	broadcastMotorsLeft:Right
Scratch-Object ID	MovieMedia	looks ops	broadcastMotorsLeft:Right
Scratch-UI-Dialogs	ScratchMedia	sound ops	broadcastMotorsStop
Scratch-UI-Panes	ScratchSpriteMorph	sensing ops robot	robotMotorDirection
Scratch-UI-Watchers	ScratchStageMorph	sensing ops	robotMotorPowerMenu
Scratch-UI-Support	ScriptableScratchMorph	list ops	
Scratch-Paint	SoundMedia	string ops	
Scratch-Sound		motor ops robot	
Scratch-Translation	instance	motor ops	
Scratch-Networking	?	other ops	
	class		

```

broadcastMotorsLeft: left Right: right Secs: secs elapsed: elapsed from: ignored
elapsed > (((secs*1000) asInteger) - 100) ifTrue: [+ self broadcastMotorsStop].
self broadcastMotorsLeft: left Right: right.
  
```

ROS<->Scratch Interface

```
#!/usr/bin/env python
import ...

def parseData(str):
    # Parse text received from Scratch broadcasts
    ...
    if e: # Successfully parsed
        ...
        tank(left,right)

def tank(left, right):
    # Call the tank service for l,r
    ...

def sendScratchSensor(variable, value, scratchSock):
    sendScratchCommand('sensor-update \''+variable+'\' '+value+' ', scratchSock)

def sendScratchCommand(cmd, scratchSock):
    ...
    scratchSock.send(a.toString() + cmd)

def cb_sensorPacket(sp):
    ...
    sendScratchSensor("rosScratchBump", "1" if (sp.bumpLeft or sp.bumpRight) else "0", scratchSock)
    sendScratchSensor("rosScratchLightLeft", str(sp.cliffFrontLeftSignal), scratchSock)
    sendScratchSensor("rosScratchLightRight", str(sp.cliffFrontRightSignal), scratchSock)

def makeConnection():
    global scratchSock
    ...
    scratchSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    ...

def main():
    # Connect to ROS
    ...
    # Receive Scratch commands;
    ...
    makeConnection()
    ...
    data = scratchSock.recv(1024)
    ...
    parseData(data)
    ...
```

ROS<->Scratch Interface

- **Main release:** iRobot Create (Movement and sensing)
- **Experimental release:** iRobot Create + Camera
- **Experimental release:** AR.Drone (Movement only)



Made ROS part more failure-robust

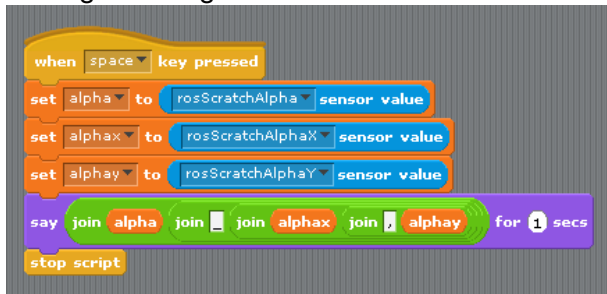
- Automatic node restarts

```
launch
<!-- iCreate -->
<param name="/brown/irobot_create_2_1/port" value="/dev/rfcomm0" />
<node name="irobot_create_2_1" pkg="irobot_create_2_1" type="driver.py" respawn="true" />

<!-- ROS<->Scratch interface -->
<node name="ros_scratch_1create" pkg="ros_scratch" type="ros_scratch_1create.py" respawn="true" output="screen" />
</launch>
```

Demos for New Blocks

- AR tag following

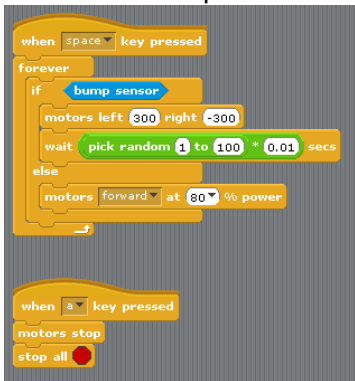


A Scratch script for AR tag following. The script starts with a yellow 'when space key pressed' block. This is followed by three orange 'set' blocks: 'set alpha to rosScratchAlpha sensor value', 'set alphax to rosScratchAlphaX sensor value', and 'set alphay to rosScratchAlphaY sensor value'. These three blocks are connected to a purple 'say' block with the text 'join alpha join [] join alphax join [] alphay' and a duration of '1 secs'. The script ends with a yellow 'stop script' block.

```
when space key pressed
set alpha to rosScratchAlpha sensor value
set alphax to rosScratchAlphaX sensor value
set alphay to rosScratchAlphaY sensor value
say join alpha join [ ] join alphax join [ ] alphay for 1 secs
stop script
```

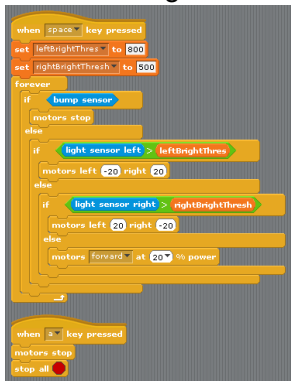
Demos for New Blocks

- Enclosure escape



Demos for New Blocks

- Line following



```
when space key pressed
set leftBrightThresh to 800
set rightBrightThresh to 500
forever
  if bump sensor
    motors stop
  else
    if light sensor left > leftBrightThresh
      motors left -20 right 20
    else
      if light sensor right > rightBrightThresh
        motors left 20 right -20
      else
        motors forward at 20% 90 power
  end
end
when a key pressed
motors stop
stop all
```

The image shows a Scratch script for line following. It starts with a 'when space key pressed' event block. This is followed by two 'set' blocks: 'set leftBrightThresh to 800' and 'set rightBrightThresh to 500'. A 'forever' loop contains an 'if bump sensor' block. If the bump sensor is triggered, 'motors stop'. Otherwise, there are two nested 'if' blocks. The first checks 'if light sensor left > leftBrightThresh' and sets 'motors left -20 right 20'. The second checks 'if light sensor right > rightBrightThresh' and sets 'motors left 20 right -20'. If neither condition is met, the 'else' block sets 'motors forward at 20% 90 power'. Below the 'forever' loop is a 'when a key pressed' event block that triggers 'motors stop' and 'stop all'.

Demos for New Blocks

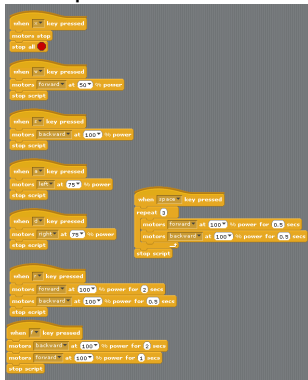
- Basic motor control

The image displays four separate Scratch scripts for motor control, each triggered by a 'when key pressed' event. The scripts are as follows:

- Script 1:**
 - when **3** key pressed
 - motors stop
 - stop all
- Script 2:**
 - when **5** key pressed
 - motors forward at **80** % power
 - wait **1** secs
 - motors backward at **75** % power
 - wait **1** secs
 - motors left at **50** % power
 - wait **1** secs
 - motors right at **25** % power
 - wait **1** secs
 - motors stop
 - stop script
- Script 3:**
 - when **6** key pressed
 - motors left **-100** right **100**
 - wait **1** secs
 - motors left **50** right **-75**
 - wait **1** secs
 - motors stop
 - stop script
- Script 4:**
 - when **0** key pressed
 - motors backward at **25** % power for **4.0** secs
 - motors left at **50** % power for **2.0** secs
 - motors right at **75** % power for **2.0** secs
 - motors forward at **100** % power for **1.0** secs
 - stop script
- Script 5:**
 - when **7** key pressed
 - motors left **-100** right **100** for **1.0** secs
 - motors left **80** right **50** for **2.0** secs
 - stop script

Demos for New Blocks

- Teleoperation



The image shows a Scratch script for teleoperation, consisting of several 'when key pressed' blocks. Each block contains a 'motors' block with specific parameters and a 'stop script' block. The blocks are as follows:

- when **W** key pressed: motors stop, stop script.
- when **A** key pressed: motors forward at **20%** to power, stop script.
- when **S** key pressed: motors backward at **100%** to power, stop script.
- when **L** key pressed: motors left at **75%** to power, stop script.
- when **R** key pressed: motors right at **75%** to power, stop script.
- when **Space** key pressed: repeat 1 times: motors forward at **100%** to power for **0.5** secs, motors backward at **100%** to power for **0.5** secs, stop script.
- when **F** key pressed: motors forward at **100%** to power for **2** secs, motors backward at **100%** to power for **0.5** secs, stop script.
- when **B** key pressed: motors backward at **100%** to power for **2** secs, motors forward at **100%** to power for **0.5** secs, stop script.

Wiki instructions

Installation

There are two major ways to install this software. The simpler of the two is to simply download a copy of our virtual machine², which has all of the software pre-installed, and run the machine inside of [VirtualBox](#).

A more advanced installation below requires the user to download and install packages in Ubuntu or another ROS and Scratch compatible operating system.

Modified Scratch

Install Scratch (sudo aptitude install scratch). Obtain the ros-scratch.zip package and unzip it on your \$PATH. Modify the path within the executable wrapper ros-scratch to reflect the path to the file ScratchSourceCode¹2.4.image.

ROS and ros_scratch

[Install ROS](#), then install brown-ros-pkg by [checking it out of SVN](#) and placing it on \$ROS_PACKAGE_PATH.

Changes from original Scratch

For developers looking to incorporate the changes made in our Scratch image into their own Scratch images, the following files in the virtual filesystem have been modified:

- Scratch-Objects/ScratchSpriteMorph2(class)/block\ specs/blockSpecs
- Scratch-Objects/ScriptableScratchMorph2(class)/block\ specs/blockSpecs
- Scratch-Objects/ScriptableScratchMorph2(instance)/sensing\ ops\ robot/ (entire folder)
- Scratch-Objects/ScriptableScratchMorph2(instance)/motor\ ops\ robot/ (entire folder)
- Scratch-Blocks/CommandBlockMorph2(instance)/accessing/uncoloredArgMorphFor:

The End

- Questions?
- Comments?

The screenshot displays the Scratch IDE interface with a script titled "line-follower" for a sprite named "Sprite 1". The script is written in Scratch blocks and is designed to control a robot's movement based on light sensor input.

Script Details:

- When clicked:** Triggers the start of the program.
- When 22438 key pressed:** Resets the "leftBrightThresh" variable to 800 and the "rightBrightThresh" variable to 500.
- When Sprite 1 clicked:** Starts a 3-second timer.
- Start 3 secs:** Initiates a "forever" loop.
- Forever Loop:**
 - Repeat 10 times:** A sub-loop that repeats the following actions 10 times:
 - Broadcast:** Broadcasts a message.
 - Broadcast and wait:** Waits for the broadcast to be received.
 - When I receive:** Triggers a "forever if" loop.
 - Forever if:** Checks for a "key pressed" event. If it occurs, it triggers a "forever" loop.
 - Forever Loop (if):**
 - if:** Checks if "light sensor left" is greater than "leftBrightThresh". If true, it sets "motors left" to 20 and "right" to 20.
 - else:** Checks if "light sensor right" is greater than "rightBrightThresh". If true, it sets "motors left" to 20 and "right" to 20.
 - else:** Sets "motors forward" to 20 and "power" to 20.

- Stop script:** Ends the program.
- Stop all:** Stops all scripts on the stage.
- Motor Commands:** At the bottom of the script, there are several "motors forward" and "motors left" blocks with numerical values (e.g., 80, 200, 100) and "to power" settings.

Blank

This slide intentionally left blank.