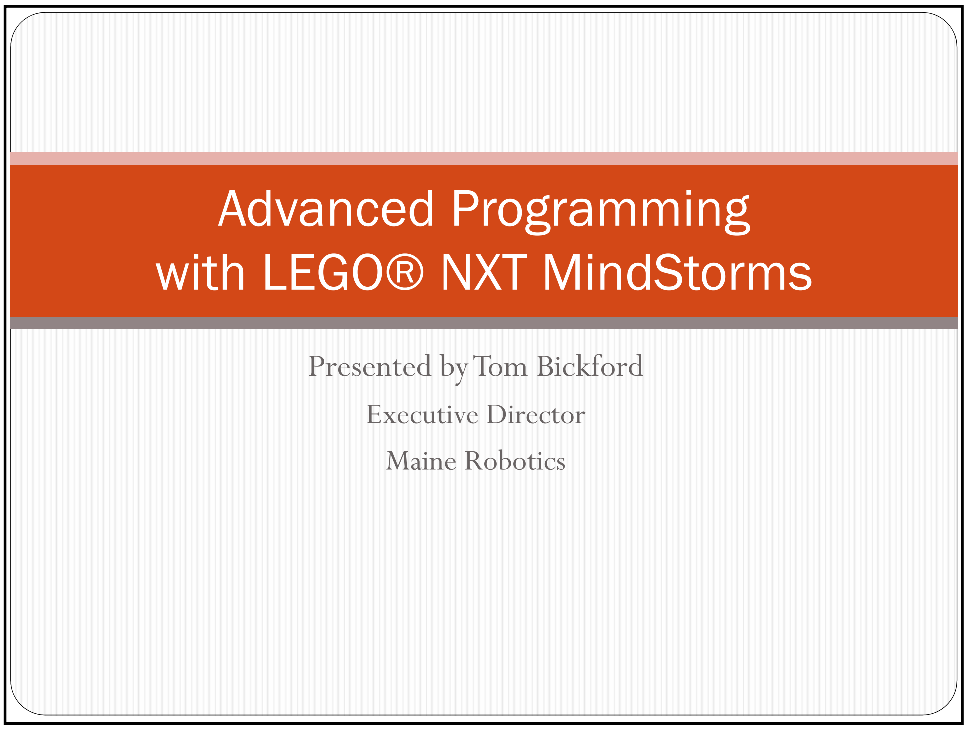
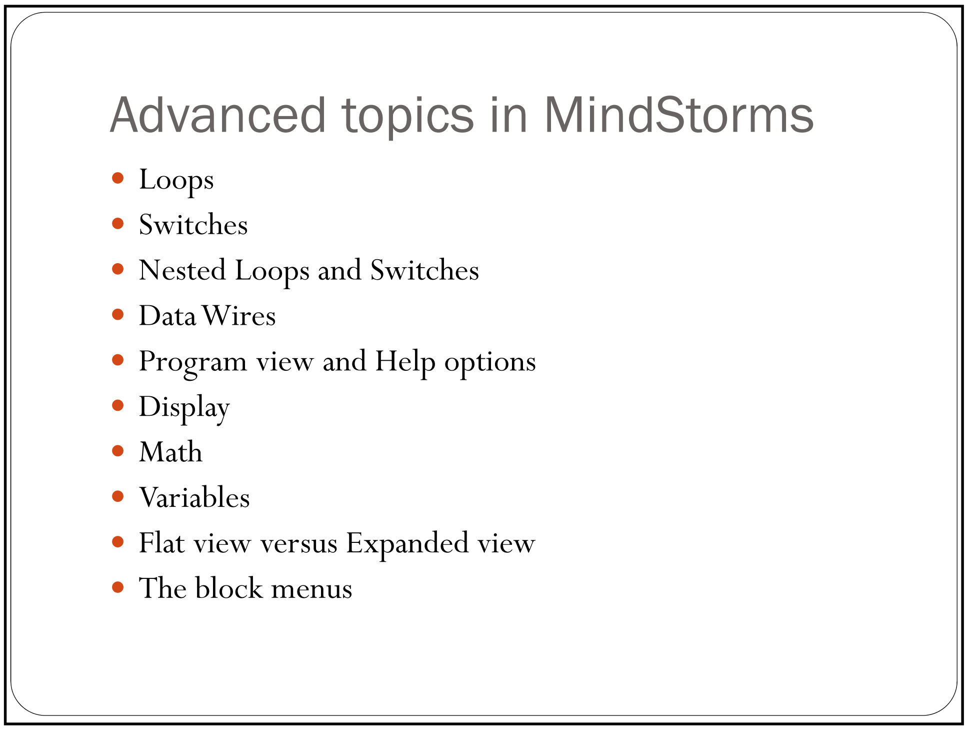


Advanced Programming with LEGO® NXT MindStorms

Presented by Tom Bickford

Executive Director

Maine Robotics



Advanced topics in MindStorms

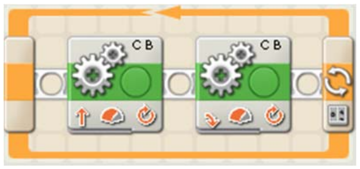
- Loops
- Switches
- Nested Loops and Switches
- Data Wires
- Program view and Help options
- Display
- Math
- Variables
- Flat view versus Expanded view
- The block menus

The Loop

- The loop provides an opportunity to repeat any of the instruction blocks inside the loop
- The loop will stop here in this line until the condition of the loop is true
- Conditions:
  - Forever (endless loop)
  - Sensor (example: Do an activity until the touch is pushed)
  - Time (loop for 20 seconds)
  - Count (loop 8 times)
  - Logic (loop while variable < 10)

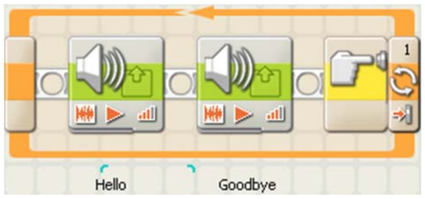
The Loop, go in a square

- Go straight
- Turn (0.6 rotation on simple NXT = 90° turn)
- Repeat 4 times



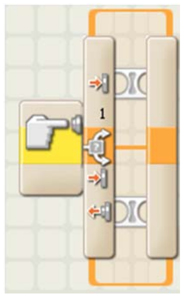
The Loop, sensor

- Pick a sensor for the loop (in this case touch)
- The loop will continue until the condition is true (pushed, released, or bumped for a touch sensor)
- This program will say “Hello”, “Goodbye” over and over until the touch sensor is pushed



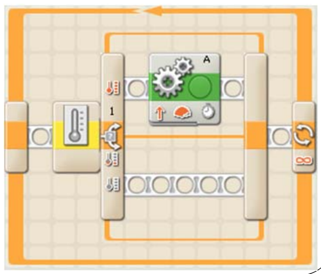
The Switch

- Switches allow for conditional assessments of the NXT environment.
- Conditions:
  - Sensor (all the usual to pick from)
  - Value (logic, number or text)
- If the condition is true, then do the top line of programming
- If the condition is false, then do the bottom line of programming
- Equivalent to an IF THEN command
- Switches ask the question only once. To continually ask if the condition is true you must imbed in a LOOP



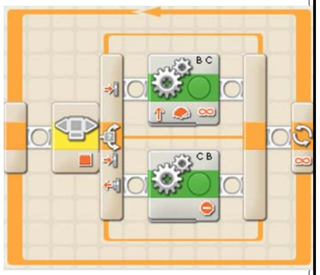
The Switch, terrarium control
(switch nested in loop)

- Using a temperature sensor inside a terrarium and a 9 volt motor (LEGO or otherwise); this program will turn on an exhaust motor to keep the terrarium temperature moderate
- This loop is set to infinite
- Temperature Switch
- IF temp > 100° then turn on motor, ELSE don't



The Switch, remote control
(switch nested in loop)

- By holding the NXT in the hand and having a wire leading to the chassis, you can use the NXT buttons as sensors, shown here.
- IF the orange ENTER button is pressed, then the motors run forward, ELSE stop motors
- NOTE: set motor control to infinite. Most programming blocks will stop the program until the block is complete. So don't use "go 1 rotation" or you'll have to wait for the rotation to finish before reading for the switch condition again.

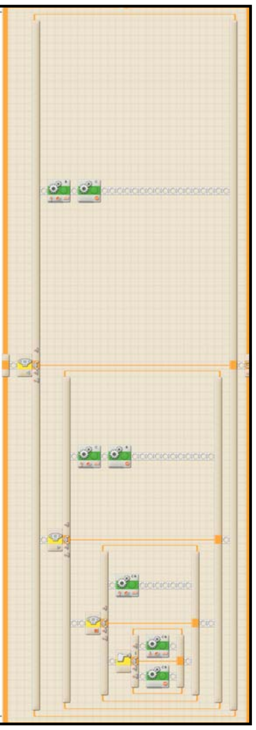


Remote Control
(Multiple nested switches)

- What if you want to use the arrow keys to turn right and left and the enter key to go forward and a touch sensor if you want to go backwards?
- You use multiple nested switches inside of a loop
- IF right arrow, turn right
  - ELSE if left arrow, turn left
    - ELSE if enter button, go straight
      - ELSE if touch sensor on 1, go back
        - ELSE shut off motors

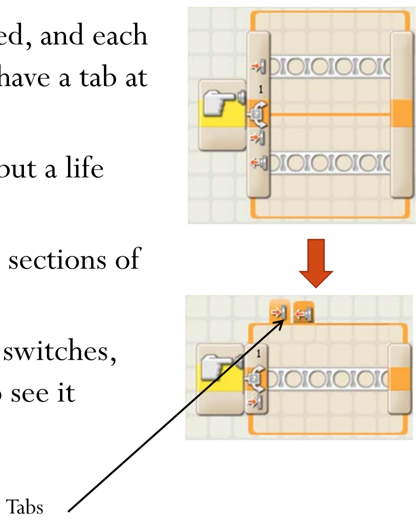
Remote Control
(Multiple nested switches)

- As you start to nest multiple loops and switches inside of each other you find that the program gets "away from you" visually. You can no longer see it all.
- Each switch splits the screen in half and gets smaller and smaller (or the empty space gets bigger and bigger)
- Use the Lower Right help/view window (question mark/magnifying glass) to see all of the program. Click around to see parts that aren't shown on the major screen.



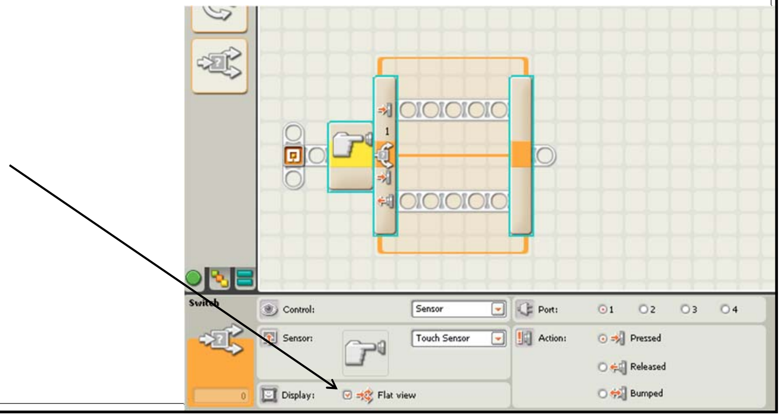
Switches and Flat View

- Switches can have Flat View disabled, and each condition for the switch will then have a tab at the top of the block
- Not a big saver in small programs but a life saver in large programs
- Toggle the tabs to see the different sections of code inside the block sections
- You can nest switches inside other switches, but you have to display flat view to see it



Turning on/off Flat View

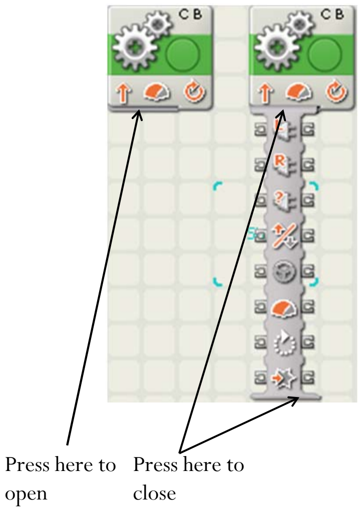
- Each Switch has a FLAT VIEW checkbox in the control area for the switch



Control: Sensor
Sensor: Touch Sensor
Display: ☑ Flat view
Port: [✓ 1] 2 3 4
Action: [✓ Pressed] Released Bumped

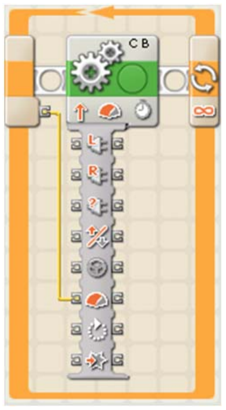
Data Wires

- We use Data Wires to pass information around inside of a program. This is easier than using variables and accomplishes much of the same function
- Data wires can go between blocks and are connected at the Data Terminals (normally hidden)
- Shown here are the same move blocks with the Data Terminal hidden and shown

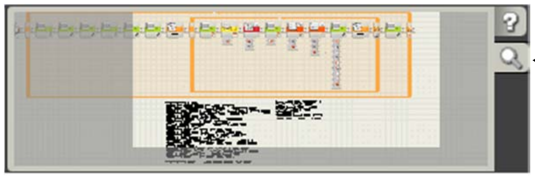


Linking the Loop to the Motors

- Lets run a loop from 0 to 100
- And lets make the Motor Power level equal the Loop counter (0 to 100) which would match the range of the motor power level
- You need to go to the LOOP block and click the "show counter" option at the bottom
- Now open up the motor Data Terminal and use the cursor to click between the LOOPS counter button and the motor speed control
- Set the motor duration to 0.1 second
- Now the robot will start at a stand still (count = power = 0)
- With each pass through (0.1 seconds) the count will increase until you get to at least 20 (the motors won't run with minimal voltage) and then start to move, slowly at first, then faster.
- The program will continue until count = power = 100, then stop



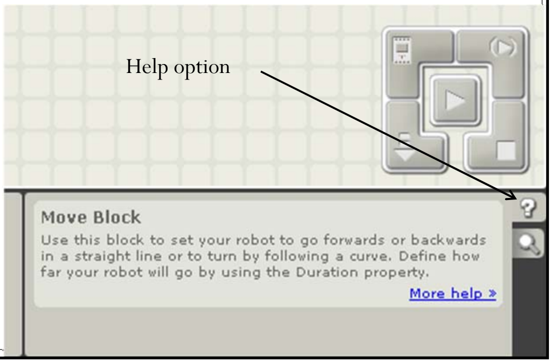
Finding your way in a big program



- Here we see a large program that is bigger than will display on the screen. The white area is what is showing on the large screen, the rest is not displayed. Simply click anywhere on this small window and you will change what appears on the large screen.

Using the HELP options

- Click on block, such as the MOVE block and then go to the lower right where the help window is (click on ? If the window shows the program outline.
- Don't be afraid to use the HELP or MORE HELP features, it is important



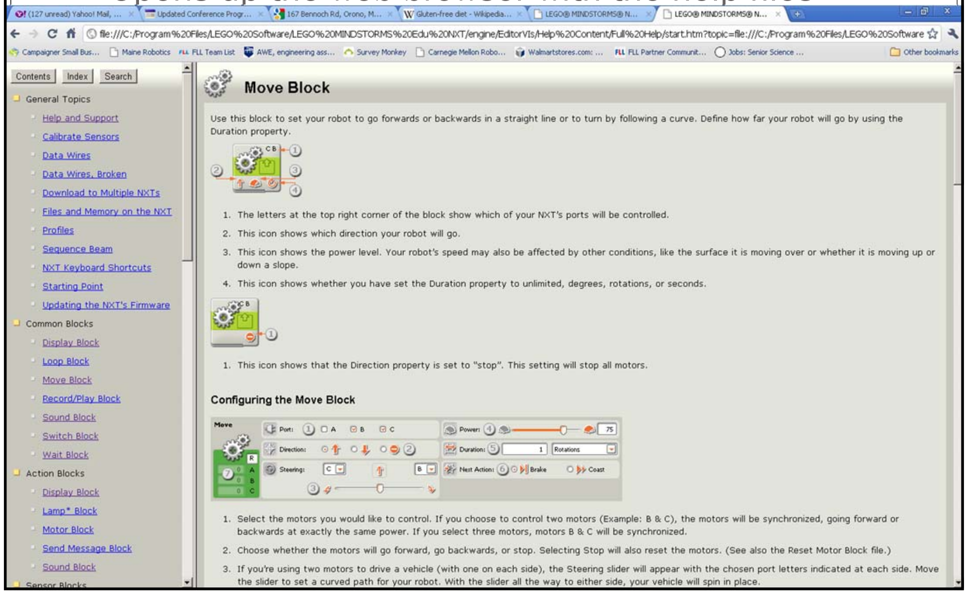
Move Block

Use this block to set your robot to go forwards or backwards in a straight line or to turn by following a curve. Define how far your robot will go by using the Duration property.

More help »

MORE HELP?

Opens up the web browser with the help files



Move Block

Use this block to set your robot to go forwards or backwards in a straight line or to turn by following a curve. Define how far your robot will go by using the Duration property.

1. The letters at the top right corner of the block show which of your NXT's ports will be controlled.
2. This icon shows which direction your robot will go.
3. This icon shows the power level. Your robot's speed may also be affected by other conditions, like the surface it is moving over or whether it is moving up or down a slope.
4. This icon shows whether you have set the Duration property to unlimited, degrees, rotations, or seconds.

1. This icon shows that the Direction property is set to "stop". This setting will stop all motors.

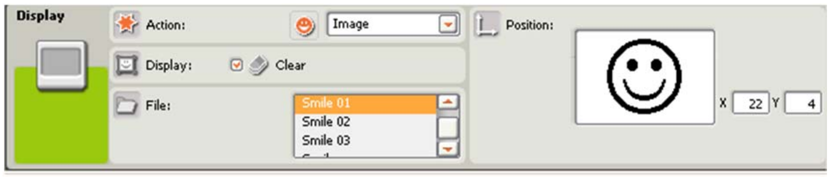
Configuring the Move Block

1. Select the motors you would like to control. If you choose to control two motors (Example: B & C), the motors will be synchronized, going forward or backwards at exactly the same power. If you select three motors, motors B & C will be synchronized.
2. Choose whether the motors will go forward, go backwards, or stop. Selecting Stop will also reset the motors. (See also the Reset Motor Block file.)
3. If you're using two motors to drive a vehicle (with one on each side), the Steering slider will appear with the chosen port letters indicated at each side. Move the slider to set a curved path for your robot. With the slider all the way to either side, your vehicle will spin in place.

Display Block

- The display block is one of the more helpful programming features
- You can display pictures, text, dots or lines
- One thing you can't display are numbers, so numbers have to be converted to text prior to displaying
- The display block is 100 pixels wide and 64 pixels tall
- If you are displaying text, there are 8 lines of text

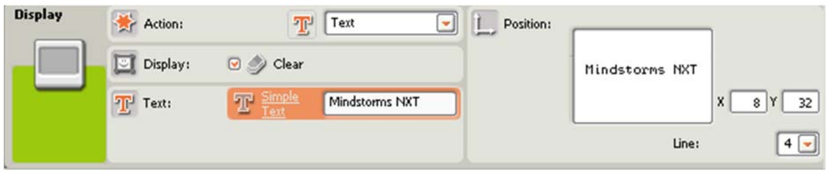
Display Block: Image



Action: Image
Display: Clear
File: Smile 01, Smile 02, Smile 03
X 22 Y 4

- You can select which of the images to show on the display and where to position the top left corner of the image
- You can also tell it to clear the screen before displaying the current image

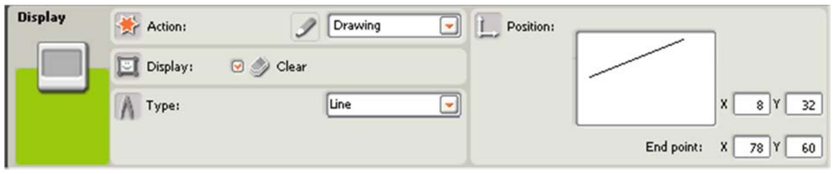
Display Block: Text



Action: Text
Display: Clear
Text: Simple Text — Mindstorms NXT
Mindstorms NXT
X 8 Y 32
Line: 4

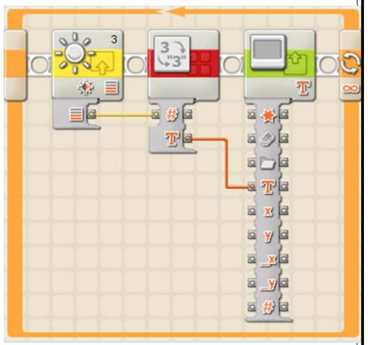
- You can type in what text you want to appear on the screen and which line you want it to be written on (total of 8 lines)
- You can also tell it to clear the screen before displaying the current text is written

Display Block: Drawing


Display
Action: Drawing
Display: Clear
Type: Line
Position: X 8 Y 32
End point: X 78 Y 60

- You can draw on the screen as well and you have 3 choices: point, line or circle
- You can also tell it to clear the screen before displaying the current text is written
- Drawing on the screen is a useful way to show data from sensors

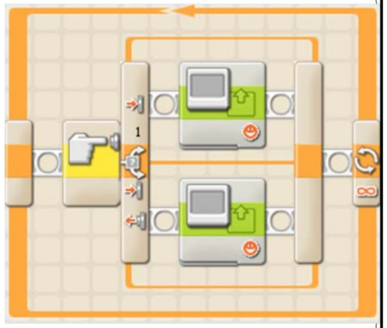
Display Block
Showing values from a sensor



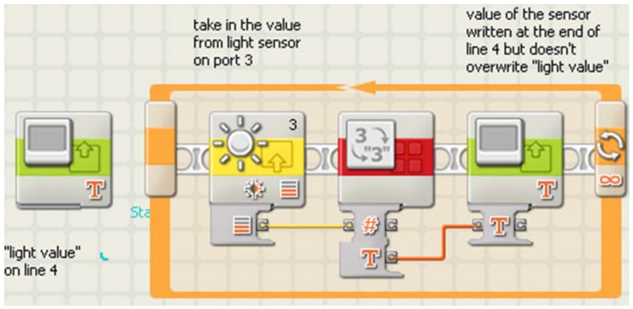
- Here we take a LIGHT SENSOR block (found under the Complete Palette
- Connect it to a NUMBER TO TEXT block
- Connect that to a DISPLAY block's Text data terminal
- You have to tell the Display block that you want to display TEXT and which line you want to display it on

Display Block
Showing different images

- Here is a LOOP with a touch sensor SWITCH
- If touched it will show a smiley face
- If not touched it will show a frown face

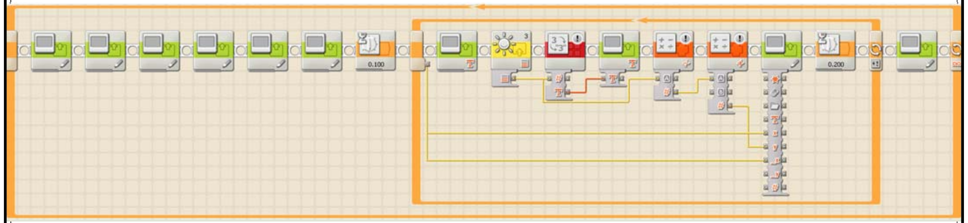


Display using multiple display blocks


take in the value from light sensor on port 3
value of the sensor written at the end of line 4 but doesn't overwrite "light value"
"light value" on line 4
3
"3"

- Here we use one block outside of the loop to write "light value" on line 4
- Then we keep writing the value of the sensor at the end of line 4 and we get:
- "light value 39" on line 4

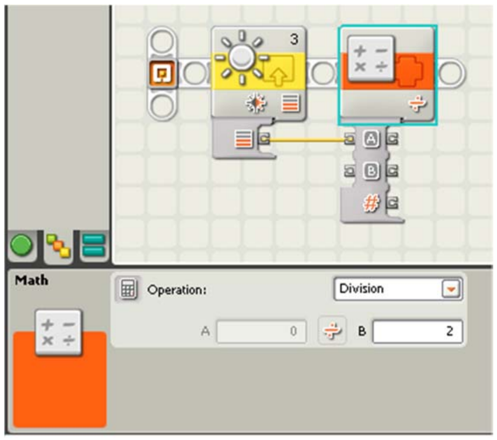
Using Display to show sensor data graphically



- Using a loop, we build a program that graphs a vertical line, whose height is equal to the light sensors' value divided by 2 (LS is from 0-100, but display only shows 0-64)
- And we use the count value of the loop for our x axis position (0-100)
- Each pass draws a vertical line from y=10 to y=10+value with x=count
- There is also a pause/wait in each pass to make the information useful. We can set the timer to any amount depending on if we want to measure for a minute or a day or a week.
- The blocks at the beginning simply give us vertical grid lines as reference with 10 pixels between them (or 20 points on the 0-100 LS scale)

The Math Block

- The math block allows us to do the following:
  - Addition
  - Subtraction
  - Multiplication
  - Division
  - Absolute Value
  - Square Root
- Values can be put in at the program OR taken from other blocks in the program (sensors, counts, variables, etc)



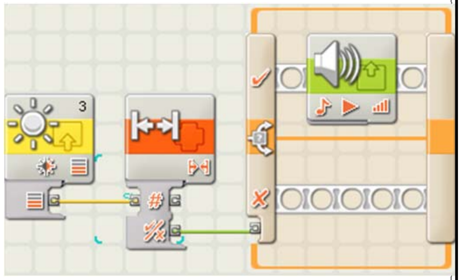
Operation: Division

A 0 B 2

Shown here: The value of the light sensor is being divided by 2, then the value would be available on the outside of the Math block by connecting to the # data terminal

Range Block

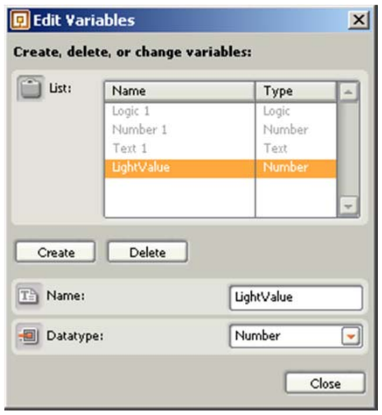
- You can use the Range Block to select a course of action if a value falls within a certain range.
- It doesn't do anything by itself and must be paired with two things
  - An input number and
  - An output logic value (true/false)



- This program reads the value on the light sensor and checks if it is in a certain range
- If it is it plays a note otherwise it doesn't

Variables

- Go to *Edit* and *Define Variables* under the menu at the top of the page
- Select Create from the pop up window and give the variable a name. On the example to the right we've name it "LightValue" and set it to a Number type (could also be Text or Logic)



Edit Variables

Create, delete, or change variables:

List:

Name	Type
Logic 1	Logic
Number 1	Number
Text 1	Text
LightValue	Number

Create | Delete

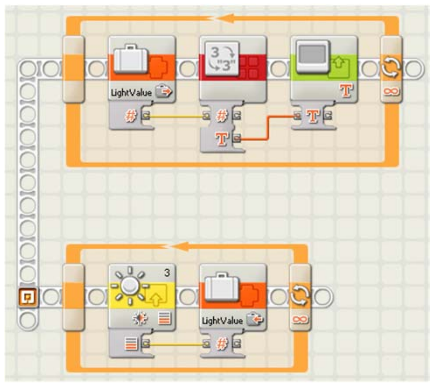
Name: LightValue

Datatype: Number

Close

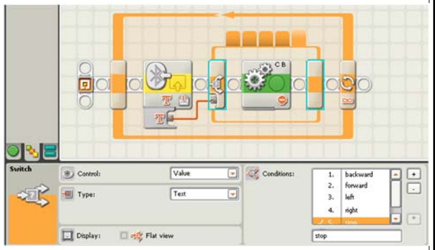
Variables in the Program

- In this program we see TWO strands running at the same time
- The lower strand is looping and assigning the value of the LS to the variable "LightValue" and
- The upper strand is looping and displaying the value of the variable in the NXT window



Switches with multiple discrete options

- In this example, the NXT will receive a Bluetooth text message and feed this as a value to a Switch
- The Switch has been defined with multiple text messages to compare this value to (forward, back, left, right, and stop)
- In order to define multiple switch options you MUST deselect the flat view, then use the + or – options to add or delete options
- You should select one of the values as default (no new value), in this case we selected stop


Switch

Control: Value

Type: Text

Display: Flat view

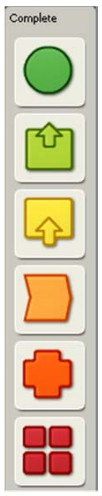
Conditions:

1. backward
2. forward
3. left
4. right
5. stop

stop

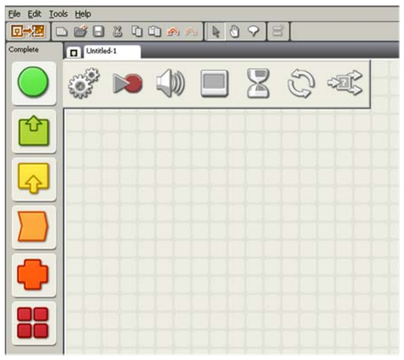
The Complete Pallette

- There are six sub menus on the Complete palette
  - Common Palette
  - Action Palette
  - Sensor Palette
  - Flow Palette
  - Data Palette
  - Advanced Palette



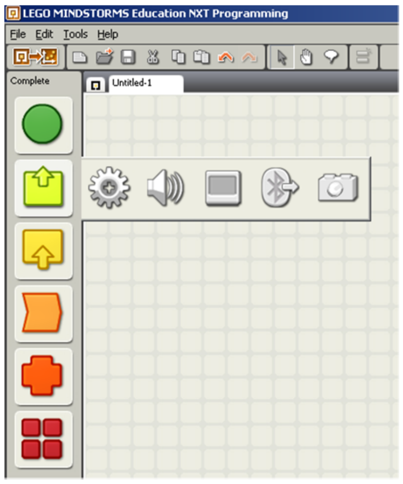
Common Palette

- The seven common palette blocks are:
  - Move
  - Record/Play (don't use)
  - Sound
  - Display
  - Wait for
  - Loop
  - Switch



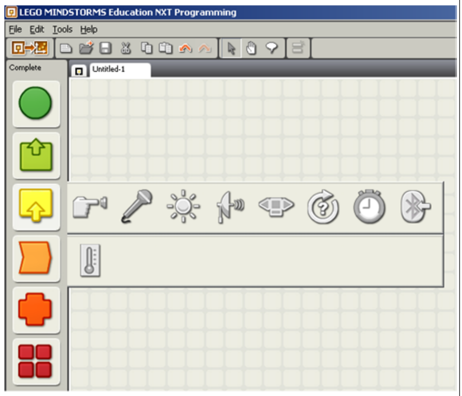
Action Palette

- The five action palette blocks:
  - Motor
  - Sound
  - Display
  - Send message
  - Lamp (can use motor as well)
- Most of this can be accessed through the common palette, although there are slight differences



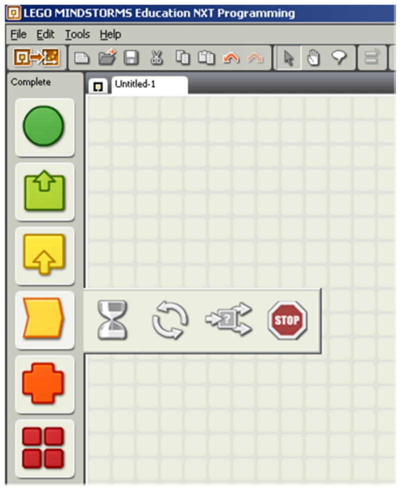
Sensor Palette

- The nine sensor blocks:
  - Touch sensor
  - Sound sensor
  - Light sensor
  - Distance sensor
  - NXT buttons
  - Rotation sensor
  - Timer
  - Receive message
  - Temperature
- All of these are used as data sources for other parts of your program



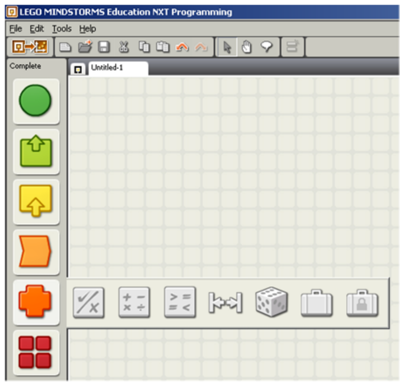
Flow Palette

- The four Flow blocks are:
  - Wait
  - Loop
  - Switch
  - Stop program
- The stop program block can be used any where in the program to end the whole program



Data Palette

- The seven Data blocks
  - Logic
  - Math
  - Compare
  - Range
  - Random
  - Variable (read and write)
  - Constant



Advanced Palette

- The 10 Advanced blocks
  - Number to text
  - Text
  - Keep alive
  - File access
  - Calibrate sensor
  - Reset motor
  - Start data-logging
  - Stop data-logging
  - Bluetooth connection
  - Compass sensor

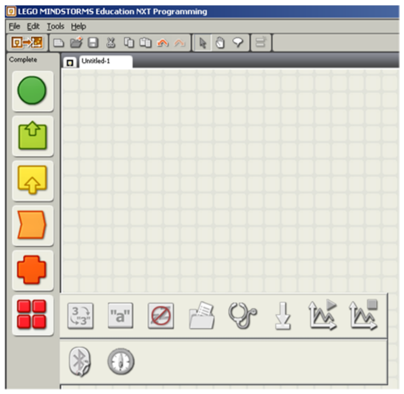


Image files

- The NXT-g software (MindStorms) uses RIC image files, which is a raster based image format.
- NXTRICeditV2 is available on line to edit your own images
- These image files need to be in the program file directory /engine/pictures in order to be automatically available with the Display block

Sound files

- The sound files used by the NXT are RSO files
- There is a utility called WAV2RSO that is available to convert WAV files to RSO and RSO to WAV files
- This would allow students to record their own sounds for use on the NXT

#