Sensing the World and Making Decisions

Week #5
Prof. Ryan Kastner
Scribbler’s Internal Sensors

- Previous lecture you learned Scribbler’s internal sensors
  1) Stall
    *Why:* It could be stuck against a wall!!
  2) Time
    *Why:* Knowing the time is important to have more complex robot behaviors!!
  3) Battery Level
    *Why:* So you can detect when to change the batteries!!
Scribbler’s External Sensors

- Scribbler also come equipped with a suite of external sensors (exteroceptors) that can sense various things in the environment.

- These various things can be seen as inputs and Scribbler perform different tasks depending on them.
Scribbler’s External Sensors

1) Camera

*Why:* It can take a still picture of whatever the robot is seeing

2) Light Sensors

*Why:* Scribbler detect variations in the ambience light in a room

3) Proximity Sensors

*Why:* So Scribbler can detect objects on the front and on its sides
Getting to Know Sensors

- It is important to know
  - How to access the information reported by them;
  - What this information looks like.

- Try

  \[ \text{senses()} \]
Camera

- Camera is located on the Fluke dongle

- To take pictures, use
  
  $\text{takePicture()}$

  $\text{takePicture(“color”)}$

  $\text{takePicture(“gray”)}$

- To show pictures, use
  
  $p = \text{takePicture()}$

  $\text{show} \ (p)$
Camera

- Alternatively you can use
  \[\text{show(takePicture())}\]

- You can do many different things with these pictures, but you might want to save them first:
  \[\text{savePicture(p, "NAME.jpg")}\]

- Exercise: Assume that Scribbler got lost, write a program so Scribbler turns around, takes pictures and shows them so you can locate it
while timeRemaining(30):
    show(takePicture())
    turnLeft(0.5, 0.2)

- Do you know how many pictures it took?

    $N = 0$

    while timeRemaining(30):
        show(takePicture())
        turnLeft(0.5, 0.2)
        $N = N + 1$

    print $N$
Can you create an animated GIF using these images?

```python
Pics = []
while timeRemaining(30):
    pic = takePicture()
    show(pic)
    Pics.append(pic)
    turnLeft(0.5, 0.2)
    savePicture(Pics, "NAME.gif")
```

This code uses **Lists** which we will learn at the end of this lecture.
To obtain values of light sensors, use

```plaintext
getLight()
getLight(<POSITION>)
getLight('left') OR getLight(0)
```

- The values being reported can be in the range of [0...5000]
- Low values imply bright light
Light Sensors on Scribbler

- Move your robot around, and see its values with `senses()` command

- Also try:

  ```python
  L, C, R = getLight()
  print L
  ```
Light Sensors on Fluke

- Camera on the fluke has a brightness sensor
  
  $getBright()$
  
  $getBright(<POSITION>)$

- The values being reported by these sensors can vary depending on the view of the camera

- Higher values imply bright segments while lower values imply darkness
Light Sensors on Fluke

- Important Note:
  - `getLight` reports the amount of ambient light being sensed by the robot (including the light above the robot)
  - `getBright` is an average of the brightness obtained from the image seen from the camera

_These can be used in many different ways!_
Scribbler has two infrared (IR) sensors on the front of the robot.

To obtain values of the front IR sensors, use:

- `getIR()`
- `getIR(<POSITION>)`

IR sensors return either a 1 or a 0.
- 1 implies that there is nothing in close proximity of the front of that sensor.
Fluke has three additional IR obstacle sensors

To obtain values of the obstacle IR sensors, use

\[
getObstacle()
\]

\[
getObstacle(<POSITION>)
\]

The values reported by these sensors range from 0 to 7000.

\[
A \ 0 \ implies \ there \ is \ nothing \ in \ front \ of \ the \ sensor
\]
Lists in Python

- List is a sequence of objects
- These objects could be anything: numbers, letters, strings, images etc.

- Lists are very useful way of collecting a bunch of information
- Python provides many useful operations and functions that enable manipulation of lists
Lists in Python

- Try these:
  
  #Empty List
  
  []

  \[N = [7, 14, 17, 20, 27]\]

  Cities = [“New York”, “Moscow”]
Lists in Python

- **Try these:**
  ```python
  >>> N = [7, 14, 17, 20, 27]
  >>> Cities = [“New York”, “Dar es Salaam”, “Moscow”]
  >>> FamousNumbers = [3.1415, 2.718, 42]
  >>> SwankyZips = [90210, 33139, 60611, 10036]
  >>> MyCar = [“Toyota Prius”, 2006, “Purple”]
  >>> len(N)
  >>> len(L)
  >>> N + FamousNumbers
  >>> SwankyZips[0]
  >>> SwankyZips[1:3]
  >>> 33139 in SwankyZips
  True
  >>> 19010 in SwankyZips
  False
  ```
Try these:

```python
>>> SwankyZips
[90210, 33139, 60611, 10036]

>>> SwankyZips.sort()
>>> SwankyZips
[10036, 33139, 60611, 90210]

>>> SwankyZips.reverse()
>>> SwankyZips
[90210, 60611, 33139, 10036]

>>> SwankyZips.append(19010)
>>> SwankyZips
[90210, 60611, 33139, 10036, 19010]
```
Inputs in Python

- Using the input function, you can input some values into your Python programs:

  >>> N = input("Enter a number: ")
Enter a number: 42

  >>> print N
42
Remembering Python Functions

- Basic syntax for defining new commands/functions:
  ```python
def <FUNCTION NAME>(<PARAMETERS>):
    <SOMETHING>
    ...
    <SOMETHING>
  ```

- Writing functions that return values:
  ```python
def triple(x):
    # Returns x*3
    return x * 3
  ```