Myro Overview

Below is a chapter by chapter summary of all the Myro features introduced in this text. For a more comprehensive listing of all the Myro features you should consult the C++/Myro Reference Manual.

Chapter 1

```
#include "Myro.h"
```
This preprocessor directive inserts into your program all the robot commands available in the C++/Myro library. We will use this whenever we intend to write programs that use the robot.

```
connect();
```
This command establishes a wireless communication connection with the robot and reports when the connection has been made.

```
disconnect();
```
This command disconnects the wireless connection to your robot.

```
robot.beep(<TIME>, <FREQUENCY>);
```
Makes the robot beep for `<TIME>` seconds at frequency specified by `<FREQUENCY>`.

```
robot.getName()
```
Returns the name of the robot.

`robot.setName(<NEW NAME>);`
Sets the name of the robot to `<NEW NAME>`. The new name should be enclosed in double quotes, no spaces, and not more than 16 characters long. For example: `setName("Bender")`.

`robot.gamepad();`
Enables manual control of several robot functions and can be used to move the robot around.[1]

Chapter 2

`robot.backward(SPEED);`
Move backwards at `SPEED` (value in the range -1.0…1.0).

`robot.backward(SPEED,SECONDS);`
Move backwards at `SPEED` (value in the range -1.0…1.0) for a time given in `SECONDS`, then stop.

`robot.forward(SPEED);`
Move forward at `SPEED` (value in the range -1.0..1.0).

`robot.forward(SPEED,TIME);`
Move forward at `SPEED` (value in the range -1.0…1.0) for a time given in seconds, then stop.

`robot.motors(LEFT,RIGHT);`
Turn the left motor at `LEFT` speed and right motor at `RIGHT` speed (value in the range -1.0…1.0).

`robot.move(TRANSLATE, ROTATE);`
Move at the `TRANSLATE` and `ROTATE` speeds (value in the range -1.0…1.0).

[1] The gamepad is not supported by the current C++/Myro interface.

`robot.rotate(SPEED);`
Rotates at `SPEED` (value in the range -1.0…1.0). Negative values rotate right (clockwise) and positive values rotate left (counter-clockwise).

`robot.stop();`
Stops the robot.

`robot.translate(SPEED);`
Move in a straight line at `SPEED` (value in the range -1.0…1.0). Negative values specify backward movement and positive values specify forward movement.

`robot.turnLeft(SPEED);`
Turn left at `SPEED` (value in the range -1.0…1.0)

`robot.turnLeft(SPEED,SECONDS);`
Turn left at `SPEED` (value in the range -1.0..1.0) for a time given in seconds, then stops.

`robot.turnRight(SPEED);`
Turn right at `SPEED` (value in the range -1.0..1.0)

`robot.turnRight(SPEED,SECONDS);`
Turn right at `SPEED` (value in the range -1.0..1.0) for a time given in seconds, then stops.

`wait(TIME)`
Pause for the given amount of `TIME` seconds. `TIME` can be a decimal number.

Chapter 3

`speak(<something>);`
The computer converts the text in `<something>` to speech and speaks it out. `<something>` is also simultaneously printed on the screen. Speech generation is done synchronously. That is, anything following the speak command is done only after the entire thing is spoken.

`speak(<something>, 0);`
The computer converts the text in `<something>` to speech and speaks it out. `<something>` is also simultaneously printed on the screen. Speech generation is done asynchronously. That is, execution of subsequent commands can be done prior to the text being spoken.

`timeRemaining(<seconds>)`
This is used to specify timed repetitions in a while-loop (see below).

Chapter 4

`randomNumber()`
Returns a random number in the range 0.0 and 1.0. This is an alternative Myro function that works just like `(double) rand() / RAND_MAX`, which uses the `rand()` function from the C++ `cstdlib` library (see below).

`askQuestion(MESSAGE-STRING)`
A dialog window with `MESSAGE-STRING` is displayed with choices: "`Yes`" and "`No`". Returns `"Yes"` or `"No"` depending on what the user selects.

`askQuestion(MESSAGE-STRING, LIST-OF-OPTIONS)`[2]
A dialog window with `MESSAGE-STRING` is displayed with choices indicated in `LIST-OF-OPTIONS`, which is string of labels separated by commas. Returns option string depending on what the user selects.

`currentTime()`
The current time, in seconds from an arbitrary starting point in time, many years ago.

`robot.getStall()`
Returns `true` if the robot is stalled when trying to move, `false` otherwise.

[2] This description may not reflect the final C++/Myro API.

`robot.getBattery()`
Returns the current battery power level (in volts). It can be a double number between 0.0 and 9.0 with 0.0 indicating no power and 9.0 being the highest. There are also LED power indicators present on the robot. The robot behavior becomes erratic when batteries run low. It is then time to replace all batteries.

Chapter 5

`robot.getBright(<POSITION>)`
Returns the current value in the `<POSITION>` light sensor. `<POSITION>` can either be one of `"left"`, `"center"`, `"right"` or one of the numbers 0, 1, 2.

`robot.getGamepad(<device>)`
`robot.getGamepadNow(<device>)`
Returns a `double` vector indicating the status of the specified `<device>`. `<device>` can be `"axis"` or `"button"`. The `getGamepad` function waits for an event before returning values. `getGamepadNow` immediately returns the current status of the device.

`robot.getIR(<POSITION>)`
Returns the current value in the `<POSITION>` IR sensor. `<POSITION>` can either be one of `"left"` or `"right"` or one of the numbers 0, 1.

`robot.getLight(<POSITION>)`
Returns the current value in the `<POSITION>` light sensor. `<POSITION>` can either be one of `"left"`, `"center"`, `"right"` or one of the numbers 0, 1, 2. The positions 0, 1, and 2 correspond to the left, center, and right sensors.

`robot.getObstacle(<POSITION>)`
Returns the current value in the `<POSITION>` IR sensor. `<POSITION>` can either be one of `"left"`, `"center"`, `"right"` or one of the numbers 0, 1, or 2.

`savePicture(<picture>, <file>);`
`savePicture(<picture vector>, <file>);`
Saves the picture in the file specified. The extension of the file should be "`.gif`" or "`.jpg`". If the first parameter is a vector of pictures, the file name

should have an extension "`.gif`" and an animated GIF file is created using the pictures provided.

```
robot.senses();
```
Displays Scribbler's sensor values in a window. The display is updated every second.

```
show(<picture>);
```
Displays the picture in a window. You can click the left mouse anywhere in the window to display the (x, y) and (r, g, b) values of the point in the window's status bar.

```
robot.takePicture()
robot.takePicture("color")
robot.takePicture("gray")
```
Takes a picture and returns a `picture` object. When no parameters are specified, the picture is in color.

```
cout << <vector>
cout << <list>
```
Print vector `<vector>` or list `<list>`.

Chapters 6 & 7

No new Myro features were introduced in these chapters.

Chapter 8[3]

```
GraphWin()
GraphWin(<title>, <width>, <height>)
```
Returns a graphics `window` object. It creates a graphics window with title,

[3] The graphics and sound capabilities of the C++/Myro interface are under development, and so this description might not reflect the final interface. Consult the documentation for more information.

`<title>` and dimensions `<width>` x `<height>`. If no parameters are specified, the window created is 200x200 pixels.

```
<window>.close();
```
Closes the displayed graphics window <window>.

```
<window>.setBackground(<color>);
```
Sets the background color of the window to be the specified color. `<color>` can be a named color (Google: color names list), or a new color created using the `color_rgb` command (see below).

```
color_rgb(<red>, <green>, <blue>)
```
Creates a new RGB color object using the specified `<red>`, `<green>`, and `<blue>` values. The values can be in the range 0..255.

```
Point(<x>, <y>)
```
Creates a `Point` object at (`<x>`, `<y>`) location in the window.

```
<point>.getX()
<point>.getY()
```
Returns the x and y coordinates of the `Point` object `<point>`.

```
Line(<start point>, <end point>)
```
Creates a `Line` object starting at `<start point>` and ending at `<end point>`.

```
<line>.getP1()
<line>.getP2()
```
Returns an endpoint (`Point` object) of a `Line` object.

```
Circle(<center point>, <radius>)
```
Creates a `Circle` object centered at `<center point>` with radius `<radius>` pixels.

```
Rectangle(<point1>, <point2>)
```
Creates a `Rectangle` object with opposite corners located at `<point1>` and `<point2>`.

`Oval(<point1>, <point2>)`
Creates an `Oval` object in the bounding box defined by the corner points `<point1>` and `<point2>`.

```
Polygon(<point1>, <point2>, <point3>,…)
Polygon([<point1>, <point2>, …])
```
Creates a `Polygon` object with the given points as vertices.

`Text(<anchor point>, <string>)`
Creates a `Text` object anchored (bottom-left corner of text) at `<anchor point>`. The text itself is defined by `<string>`.

`Image(<centerPoint>, <file name>)`
Creates an `Image` centered at `<center point>` from the image file `<file name>`. The image can be in GIF, JPEG, or PNG format.

All of the graphics objects respond to the following commands:

`<object>.draw(<window>);`
Draws the `<object>` in the specified graphics window `<window>`.

`<object>.undraw();`
Undraws `<object>`.

```
<line-object>.getP1()
<line-object>.getP2()
```
Returns the end points of the `<line-object>`.

`<object>.getCenter()`
Returns the center point of the `<object>`.

```
<object>.setOutline(<color>);
<object>.setFill(<color>);
```
Sets the outline and the fill color of the `<object>` to the specified `<color>`.

`<object>.setWidth(<pixels>);`
Sets the thickness of the outline of the `<object>` to `<pixels>`.

`<object>.move(<dx>, <dy>);`

Moves the object `<dx>`, `<dy>` from its current position.

The following sound-related functions were presented in this chapter.

```
<robot/computer object>.beep(<seconds>, <frequency>);
<robot/computer object>.beep(<seconds>, <f1>, <f2>);
```

Makes the robot or computer beep for `<seconds>` time at frequency specified. You can either specify a single frequency `<frequency>` or a mix of two: `<f1>` and `<f2>`.

`robot.playSong(<song>)` ;

Plays the <song> on the robot.

```
readSong(<filename>)
Reads a song object from <filename>.
```

`song2text(song)`

Converts a `song` to text format.

```
makeSong(<text>)
text2song(<text>)
```

Converts `<text>` to a `song` format.

Chapter 9[4]

`getBlob()`

Return a three-element vector of integers of the number of pixels that matched the blob, the average x and y locations of blob pixels.

```
getHeight(<picture>)
getWidth(<picture>)
```

Returns the height and width of the `<picture>` object (in pixels).

[4] The graphics capabilities of the C++/Myro interface are under development, and so this description might not reflect the final interface.

`getPixel(<picture>, x, y)`
Returns the pixel object at `x, y` in the `<picture>`.

`getPixels(<picture>)`
Returns an iterator of type `Pixels`, which returns one `Pixel` at a time from `<picture>`.

```
getRGB(pixel)
getRed(<pixel>)
getGreen(<pixel>)
getBlue(<pixel>)
```
Returns the RGB values of the `<pixel>`. `getRBG` returns a `color_rgb` object.

`getBlob()`
Returns a three-element `vector<int>` containing the number of on-pixels and the average x- and y-locations of blob pixels. See chapter text for example.

`makeColor(<red>, <green>, <blue>)`
Creates a `color_rgb` object with the given `<red>`, `<green>`, and `<blue>` values (all of which are in the range [0..255]).

```
makePicture(<file>)
makePicture(<width>, <height>)
makePicture(<width>, <height>, <color>)
```
Creates a `picture` object either by reading a picture from a `<file>`, or of the given `<width>` and `<height>`. If `<color>` is not specified, the picture created has a white background.

`pickAColor()`
Creates an interactive dialog window to select a color visually. Returns the color object corresponding to the selected color.

`pickAFile()`
Creates an interactive dialog window that allows user to navigate to a folder and select a file to open. Note: it cannot be used to create new files.

```
repaint();
repaint(<picture>);
```
Refreshes the displayed `<picture>`.

```
savePicture(<picture>, <file>);
savePicture(<picture vector>, <gif file>);
```
Saves the `<picture>` in the specified file (a GIF or JPEG as determined by the extension of the `<file>`: `.gif` or `.jpg`). `<picture vector>` is saved as an animated GIF file.

```
setColor(<pixel>, <color_rgb>);
setRed(<pixel>, <value>);
setGreen(<pixel), <value>);
setBlue(<Pixel>, <value>);
```
Sets the color of `<pixel>` to specified `<color_rgb>` or `<value>`.

```
setPixel(<picture>, x, y, <color_rgb>);
setPixel(<picture>, x, y, <pixel>);
```
Sets the pixel at `x, y` in `<picture>` to specified `<color_rgb>` or color of `<pixel>`.

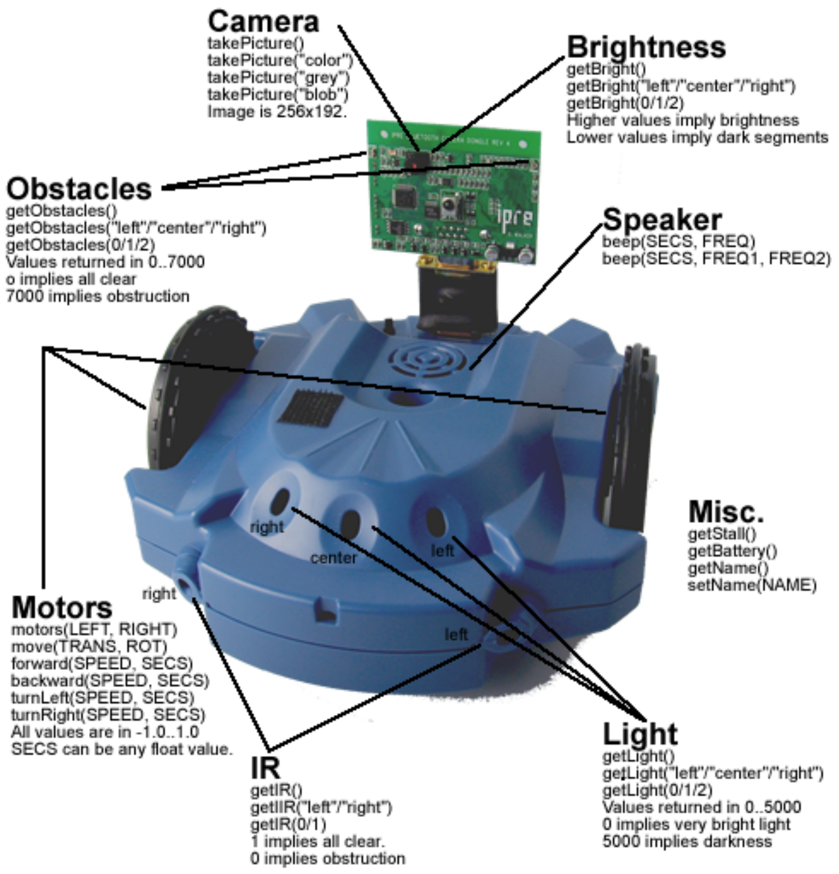
```
show(<picture>);
show(<picture>, <name>);
```
Displays the `<picture>` on the screen in a window named `<name>` (string).

```
takePicture()
takePicture("gray")
takePicture("blob")
```
Takes a picture from the Scribbler camera. It is a color picture by default, or grayscale (`"gray"`), or a filtered image based on the defined blob (`"blob"`). See chapter text for examples.

Chapters 10 – 12

No new Myro features were introduced in these chapters.

Scribbler: Myro Reference



Camera
```
takePicture()
takePicture("color")
takePicture("grey")
takePicture("blob")
Image is 256x192.
```

Brightness
```
getBright()
getBright("left"/"center"/"right")
getBright(0/1/2)
Higher values imply brightness
Lower values imply dark segments
```

Obstacles
```
getObstacles()
getObstacles("left"/"center"/"right")
getObstacles(0/1/2)
Values returned in 0..7000
o implies all clear
7000 implies obstruction
```

Speaker
```
beep(SECS, FREQ)
beep(SECS, FREQ1, FREQ2)
```

Misc.
```
getStall()
getBattery()
getName()
setName(NAME)
```

Motors
```
motors(LEFT, RIGHT)
move(TRANS, ROT)
forward(SPEED, SECS)
backward(SPEED, SECS)
turnLeft(SPEED, SECS)
turnRight(SPEED, SECS)
All values are in -1.0..1.0
SECS can be any float value.
```

IR
```
getIR()
getIIR("left"/"right")
getIR(0/1)
1 implies all clear.
0 implies obstruction
```

Light
```
getLight()
getLight("left"/"center"/"right")
getLight(0/1/2)
Values returned in 0..5000
0 implies very bright light
5000 implies darkness
```