1 Webots API Documentation

1.1 Important Terminology

arg.: Short for ”argument.” Used for listing arguments expected by the blocks.

con.: Short for ”connection.” Indicates that a block should be connected to the right of another block. The value returned by a block will be fed into the block to its left.

Initialize Blocks: The initialize blocks are the setupMotor, initializeGyro, initializeGPS, and initializeOtherSensor blocks.

Motor/Sensor Name: Each motor/sensor has a name that must be supplied to the initialize blocks - do not type quotes when supplying the name. The default names of the motors and sensors are listed in the descriptions of the initialize blocks. To manually find the name of the sensor, follow these steps:

1. If the Scene Tree is not already open, press “Ctrl+t” to open it.
2. In the Scene Tree, click on the drop-down arrow for the bullet point that says “DEF ROBOT Pioneer3dx (PROTO).”
3. Click on the drop-down arrow for the bullet point that says “extensionSlot.”
4. A list of sensors will appear with their respective names listed in quotes.
5. To rename a sensor, click the drop-down arrow for the sensor you want to rename. Click on the bullet point that says “name.” Use the window that appears below to change the name.
Motor/Sensor Variable: For each motor, distance sensor, and light sensor that you use, you must create a corresponding variable to represent and reference those motors/sensors. Go to the “Variables” category, select “Create new variable...,” and name your variable accordingly (e.g. “leftMotor,” “distSensor,” etc.). Supply this variable to the initialize blocks, which will bind a sensor/motor to the variable. We will refer to this variable as a motor variable or sensor variable, but remember that it is just a regular variable. You do not need to set your motor/sensor variables to any specific value - simply supply it to the various motor/sensor blocks as instructed.

1.2 Block Descriptions

Motors

setupMotor

```
arg 1
motorObj
```

```
arg 2
motorName
```

Description: Initializes a motor and binds it to a motor variable. Must be used before a motor can be referenced by other motor blocks.

Parameters

- arg 1: The motor variable to be bound to the motor.
- arg 2: The name of the motor you wish to bind - do not enter quotes. The names of the default robot’s left and right motor are ”left wheel” and ”right wheel” respectively.

Example:

Binds the variable leftMotor to the robot’s motor for its left wheel. When you use other motor blocks and want to specify the robot’s left motor, you would then use the leftMotor variable.

setSpeed

```
arg 1
motorObj
```

```
arg 2
% 
```

Description: Sets a motor at a specified percentage of its maximum speed. Uses positive percentage for turning forward and a negative percentage for turning backward.

Parameters

- arg 1: The motor variable of the motor you want to spin.
• arg2: The percentage of the motor’s maximum speed you want to run the motor at. Accepts positive or negative integers or decimals from -100 to 100.

**Example:**

![setSpeed block](image)

Spins the motor that is bound to the `leftMotor` variable backwards at 50% speed.

**resetEncoders**

![resetEncoders block](image)

**Description:** Resets the specified motor’s encoder count (see `getEncoders` block for more information).

**Parameters**

• arg 1: The motor variable of the motor whose encoder count you want to reset.

**Example:**

![resetEncoders block](image)

Resets the encoder count of the motor bound to the `leftMotor` variable.

**getEncoders**

![getEncoders block](image)

**Description:** As the motor revolves throughout the simulation, it keeps track of how many degrees it has rotated with an encoder count. Rotating forward increases this count and rotating backward lowers this count. This block returns the specified motor’s encoder count.

**Parameters**

• con 1: The block that you want to feed the encoder count to. Accepts any block with a right connection.

• arg 1: The motor variable of the motor whose encoder count you want to retrieve.

**Example:**

![print block](image)

Prints out the encoder count of the variable bound to the `leftMotor` variable.
Time

**delay**

arg 1

Description: Suspends the program for a specified number of milliseconds. Note that the simulation will continue to run - only the program is suspended. This means that if a motor was rotating when the program arrived at the delay block, it will continue to rotate through the duration of the delay.

Parameters

- arg 1: The number of milliseconds to delay for. Accepts positive integers or decimals.

Example:

Suspends the program for 1 second.

**get_Time**

con 1

Description: Retrieves the number of milliseconds elapsed since the last resetTime block. If there were no prior resetTime blocks, the number of milliseconds elapsed since the beginning of the simulation is returned.

Parameters

- con 1: The block that you want to feed the returned time to.

Example:

Prints out the number of milliseconds elapsed since the last resetTime block or the last Start block if there were no prior resetTime blocks.

**resetTime**
Description: Resets the timer that counts the number of milliseconds that have elapsed since the last resetTime or Start block - whichever was most recent. Note that this block will not reset the timer shown to the right of simulation’s play button. It only resets an internal counter.

Sensors

The worlds you downloaded off GitHub automatically come with a Pioneer 3dx robot. This robot comes with 5 sensors:

- Gyro sensor named “gyro”
- GPS named ”gps”
- Distance sensor named ”distance sensor” and facing forward
- Light sensor named ”light sensor” and facing down
- Camera named ”camera” and facing forward

initializeGyro

arg 1

Description: Initializes the gyro sensor. This block must be used before any other gyro sensor blocks can be used.

Parameters

- arg 1: The name of the gyro sensor - do not enter quotes. By default, the name of the gyro sensor is ”gyro” - see the preface of the Sensors section to see how to find and/or change the name.

Example:

Initializes a gyro sensor with the name “gyro”.

initializeGPS

arg 1

Description: Initializes the GPS. This block must be used before any other GPS blocks can be used.

Parameters
• arg 1: The name of the GPS - do not enter quotes. By default, the name of the GPS is ”gps” - see the preface of the Sensors section to see how to find and/or change the name.

Example:

```
initializeGPS gps
```

Initializes a GPS with the name “gps”.

**initializeOtherSensor**

```
initializeOtherSensor lightSensor sensor1
```

**Description:** Initializes a light or distance sensor and binds that sensor to a sensor variable. **Must** be used before any other light/distance sensor blocks can be used.

**Parameters**

• arg 1: Sensor type.

• arg 2: The name of the light/distance sensor - do not enter quotes. By default, the light sensor is named ”light sensor” and the distance sensor is named ”distance sensor” - see the preface of the Sensors section to see how to find and/or change the name.

• arg 3: The sensor variable to be bound to the sensor.

Example:

```
initializeOtherSensor distanceSensor distance sensor myDistSensor
```

Initializes a distance sensor named “distance sensor” and binds it to the variable *myDistSensor*.

**getGyroVals**

```
getGyroVals
```

**Description:** Retrieves the robot’s angle heading from the gyro sensor. The angle heading is a count that starts at 0, increases by 1 for every degree the robot rotates counterclockwise, and decreases by 1 for every the robot rotates clockwise.

**Parameters**

• arg 1: The block to feed the angle heading to.

Example:
getGpsVals

Description: Retrieves the robot’s position from the GPS sensor.

Parameters
• arg 1: The block to feed the angle heading to.

Example:

getDist

Description: Retrieves the distance seen by the robot’s distance sensor in centimeters.

Parameters
• arg 1: The sensor variable for the distance sensor you want to probe.

Example:

getLight

Description: Retrieves the brightness value from the robot’s downward-facing light sensor (brightness of the floor directly under the robot). Returns a value from 0 to 255, with 0 meaning dark and 255
meaning light.

**Parameters**

- arg 1: The sensor variable for the light sensor you want to probe.

**Example:**

```
print getLight myLightSensor
```

Prints out how light/dark the floor is beneath the robot.

---

**Camera**

To simplify camera operations, Webots has added a system called recognition colors. Certain objects in your world can be assigned a recognition color, which consists of 3 different numbers from 0 to 1 that represent a color (an RGB value). To do so, follow these steps:

1. Press “Ctrl+t” to open the Scene Tree.
2. Click the drop-down arrow for the object whose recognition color you want to set.
3. Double click on the “recognitionColors” attribute.
4. Edit the red, green, and blue value in the window that appears below to set the recognition color.

Note that an object’s recognition color is completely separate from its actual color - a blue box can have an orange recognition color. In turn, you can retrieve a list of all objects with a recognition color that are in the camera’s view. You can then retrieve those objects’ position and recognition color. We will refer to objects with a recognition color as recognition objects.

**initializeCamera**

```
initializeCamera arg1 arg2
```

**Description:** Initializes a camera on the robot and binds it to a sensor variable.

**Parameters**

- arg 1: The sensor variable to be bound to the camera.
- arg 2: The name of the camera.

**Example:**

```
initializeCamera myCam camera
```

Initializes a camera named “camera” and binds it to the variable *myCam.*
getRecognitionObjects

Description: Returns a list of all recognition objects in the camera’s view.

Parameters
- con 1: The block to which you want to feed the list of recognition objects.
- arg 1: The sensor variable of the camera you want to probe.

Example:

```
print getRecognitionObjects myCam
```

Prints a list of all recognition objects in the camera’s view.

getNumRecognitionObj

Description: Returns the number of recognition objects in the camera’s view. This value is also attainable by finding the length of the list returned by the getRecognitionObjects block.

Parameters
- con 1: The block to which you want to feed the number of recognition objects.
- arg 1: The sensor variable of the camera you want to probe.

Example:

```
print getNumRecognitionObj myCam
```

Prints the number of recognition objects in the camera’s view.

getObjColor

Description: Returns the recognition color of a recognition object in the form of a list containing 3 values from 0 to 1 (RGB).
Parameters

• con 1: The block to which you want to feed the recognition color.

• inp 1: The recognition object whose color you want to retrieve. Accepts a recognition object (an element of the list returned by the getRecognitionObjects block).

Example:

The first line retrieves a list of all recognition objects in the camera’s view and stores it in the variable myList. The second line prints out the recognition color of the recognition object at index 0 of the list (first element).

getObjPos

con 1

inp 1

Description: Returns the position of a recognition object relative to the position of the robot. The position is returned as a list of 3 decimals representing XYZ coordinates with the robot at the origin.

Parameters

• con 1: The block to which you want to feed the recognition object’s position.

• inp 1: The recognition object whose position you want to retrieve. Accepts a recognition object (an element of the list returned by the getRecognitionObjects block).

Example:

The first line retrieves a list of all recognition objects in the camera’s view and stores it in the variable myList. The second line prints out the relative position of the recognition object at index 0 of the list (first element).

Other

Start

Description: Your programs must begin with a Start block, or else several internal variables will not be set.
End

**Description:** The *End* block is not necessary, but can be placed at the end of the program for aesthetic purposes.

`print`

**Description:** Prints out a value to the console window with an automatic newline character at the end.

**Parameters**

- `inp 1`: The value to be printed. Accepts anything.

**Example:**

```
print "hello"
print 1
```

Prints out “Hello” and then “1” on the line below in the console window.