

Getting Started Guide

Go Direct® Sensors and Web VPython

Introduction

This guide describes how to write Web VPython programs to connect to and gather data from most [Vernier Go Direct devices](#). This guide was written for those with some familiarity and experience with Python coding. Expert coding skills are not required to create interesting, useful Web VPython programs that use Go Direct sensors. But basic knowledge of Python, its syntax, and functions is required to make use of this guide.

Table of Contents

Introduction

- Table of Contents
- Python Editor
- Compatible Go Direct Devices

Overview of Web VPython

Getting Started

- Hardware and Software Requirements
- Web VPython Account
- Tour of a Simple Web VPython Program

Using Web VPython with a Go Direct Device

- ex3-mod-size-with-Go-Direct Example Program
- Running a Web VPython Program with a Go Direct Device
- Go Direct Device Selection and Connection

Example Programs

- Basic Starter Programs
- Go Direct Sensor Applications

Go Direct Library Functions in Web VPython

- `gdx.open()`
- `gdx.select_sensors()`
- `gdx.start()`
- `gdx.read()`
- `gdx.vp_close_is_pressed()`
- `gdx.vp_collect_is_pressed()`
- `gdx.vp_get_slider_period()`
- `gdx.collectFor()`
- `gdx.vp_rate()`

```
gdx.vp_vernier_canvas()
```

Tips for using Web VPython

Displaying Programs as a List in Your Account

Moving Programs between Web VPython and Installed VPython

Channels Available on Go Direct Devices

Troubleshooting and Support

Troubleshooting

Support

Web VPython Help

Web Resources for Python and VPython

Python Editor

This guide was written for the web-based Web VPython editor, available at <https://webvpython.org/>. A separate guide for using Go Direct devices with installed Python with the VPython library is available here: [Getting Started with Vernier Go Direct Sensors and VPython](#).

Compatible Go Direct Devices

Most Vernier Go Direct devices are compatible with the Web VPython editor. However, some of the more complex Go Direct devices are not easy to use with do-it-yourself programs in Python or other programming languages. The following devices are **not** compatible with Web VPython:

- Spectrometers, such as the Go Direct SpectroVis Plus
- Go Direct Mini GC
- Go Direct Polarimeter
- Go Wireless Heart Rate
- Go Direct Cyclic Voltammetry System

Go Direct devices that can be used with Web VPython, but require advanced programming for calibration and/or analysis are:

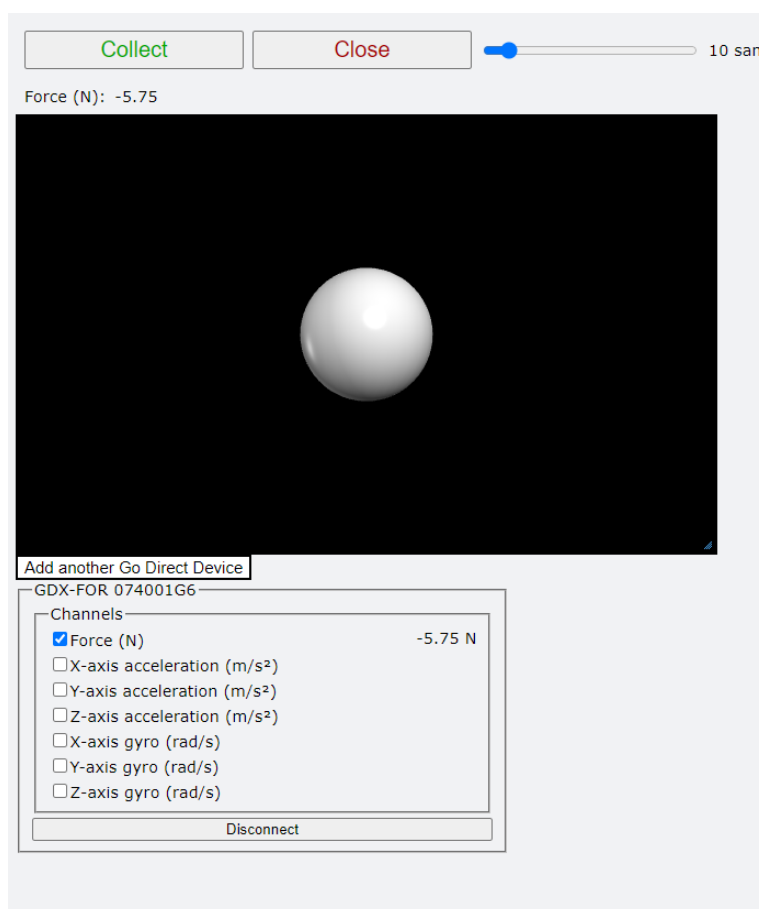
- Go Direct Blood Pressure
- Ion-selective electrodes, such as Go Direct Ammonium Ion-Selective Electrode
- Go Direct Optical Dissolved Oxygen Probe
- Timing/event-based devices, such as Go Direct Photogate, Go Direct Drop Counter, and Go Direct Projectile Launchers

Overview of Web VPython

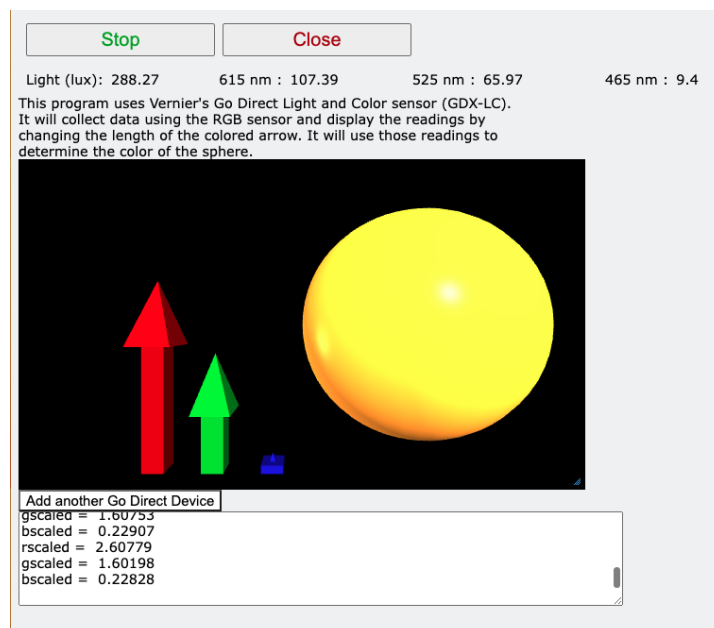
Web VPython (formerly known as Glowscript) is an easily accessible, browser-based coding platform that bundles a Python editor and the VPython library together, allowing users to easily create 3D graphics, animations, and simulations using Python programming. It was developed to simplify the creation of physics animations and simulations in an education setting. Web VPython runs in a modern browser on computers (Windows and macOS) and Chromebooks. Because it runs in a browser, no installation is required to run Web VPython; Internet access is required.

In 2022, Vernier Science Education worked with Bruce Sherwood, the lead developer of Web VPython, to build the connection between our Go Direct or “GDX” devices and Web VPython. We would like to thank Bruce for helping us with this project, but also for creating such an amazing, free, programming environment. For more on the history of VPython, see <https://brucesherwood.net/?p=136>.

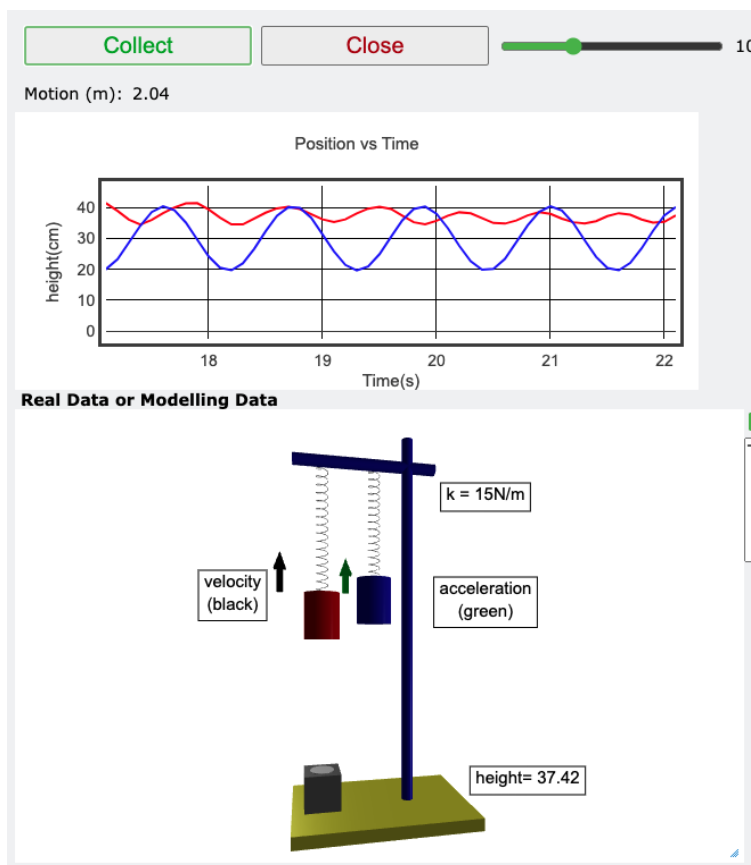
Here are some sample screenshots from Web VPython using Go Direct sensors:



Simple data collection program using Go Direct Force and Acceleration (GDX-FOR)



Color matching program using Go Direct Light and Color (GDX-LC)



Simple harmonic motion program that compares collected data from Go Direct Motion Detector (GDX-MD) and a theoretical model

Getting Started

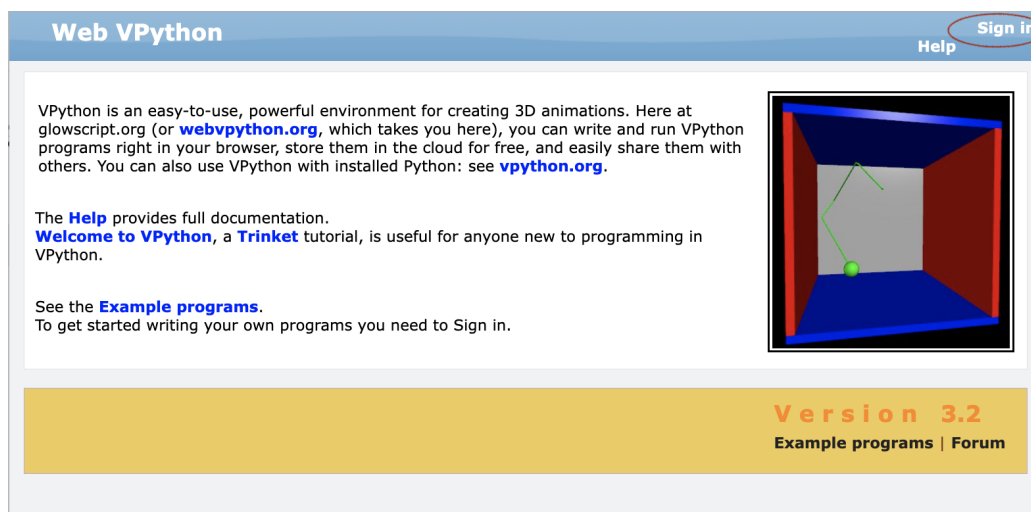
Hardware and Software Requirements

To use Web VPython with a Go Direct device, you will need the following:

- Compatible Vernier Go Direct device
- Windows® 10 (or newer) computer, macOS® computer, or Chromebook
- Access to webvpython.org using the Chrome browser
 - Note: Google Chrome is the recommended browser. Other browsers may work with Web VPython but will not with Go Direct devices.
- Web VPython account

Web VPython Account

The Web VPython platform is free and available at www.webvpython.org. In order to create, copy, or modify programs, you will need an account. Web VPython uses Google accounts for account management; you must either sign in using an existing Google account or create one.



*The **Sign in** button is in the upper right corner of the Web VPython homepage*

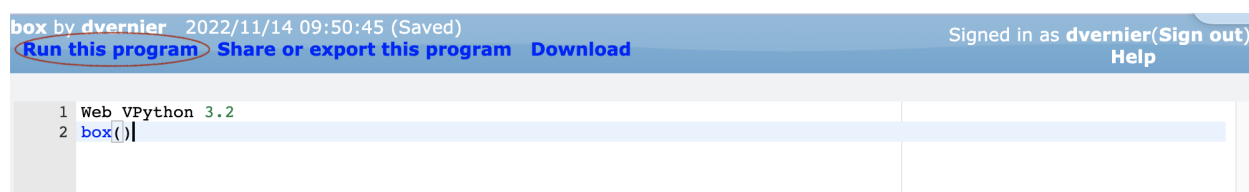
Your work in Web VPython is automatically saved to your account online. Your account includes rudimentary file management tools to organize your programs in folders, as well as rename, copy, and delete programs.

Viewing the example programs found in the “Example programs” on the homepage is helpful in understanding what can be done in Web VPython.

Tour of a Simple Web VPython Program

Once you have created a Web VPython account, you are ready to start programming. Try the following:

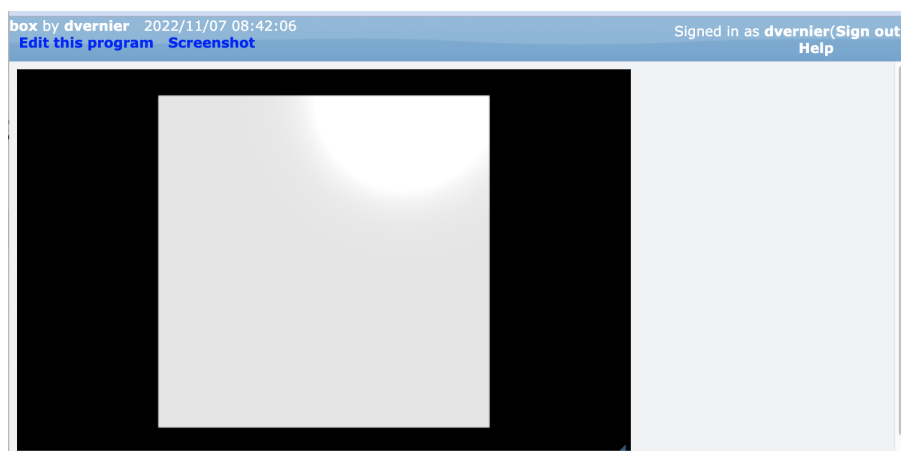
1. Go to www.webvpython.org
2. Sign in to your account.
3. Go to your programs: Click on your sign-in name
4. Click **Create New Program**.
5. Name your new program “box” and click **Create**.
6. In line 2 in the editor, write the following code: **box()**
7. Click **Run this program** from the menu.



Run this program is among a short list of options in the program menu at the top

Your code will be replaced by the output of your program, as shown in the figure below. While consisting of only a single line, your code...

- Created a “canvas” where 3D objects can appear, move, and interact with each other
- Set up rules and methods for changing the view of objects in the canvas
- Added a light source to the canvas to aid in viewing depth
- Added a box to the canvas



One line of code in Web VPython can accomplish a lot

With a Web VPython canvas, you can control the view in many ways:

- To rotate the view, drag with the right button or Ctrl-drag.

Getting Started with Go Direct Sensors and Web VPython

- To zoom, drag with the middle button, or Alt/Option depressed, or use scroll wheel.
- On a two-button mouse, middle is left + right. To pan left/right and up/down, Shift-drag.
- On a touch screen: pinch/extend to zoom, swipe or two-finger rotate.

Web VPython does all of the programming work behind the scenes to create the canvas and 3D object elements in this simple box program. That means that students can spend more time working on the interesting code and less on setup and formatting.

If you would like to explore a few more examples of simple Web VPython programs, try the following programs from our [Example Programs](https://vnr.st/glowscript) (<https://vnr.st/glowscript>):

- ex1-create-object
- ex2-modify-object

Using Web VPython with a Go Direct Device

It is easy to incorporate Go Direct sensor data into your Web VPython program. By importing the Vernier Web VPython library and using the appropriate functions, you can connect to and gather data from a Go Direct device and then use that data in your own custom animation or graph.

ex3-mod-size-with-Go-Direct Example Program

```
ex3-mod-size-with-Go-Direct by vernier-web-vpython (read only) 2023/06/27 15:35:19
Run this program Share or export this program Download

1 Web VPython 3.2
2 """
3 Control the radius of a VPython sphere with Go Direct sensor data.
4
5 For information on the 'gdx' functions, see the Getting Started Guide at:
6 https://github.com/VernierST/godirect-examples/tree/main/web_vpython
7
8 - Connect a Go Direct sensor via USB
9 - Click 'Share or export this program'
10 - Click 'Download as HTML'
11 - Double-click the HTML file to run the program
12 """
13
14 get_library('https://unpkg.com/@vernier/godirect/dist/webVPython.js')
15
16 gdx.open(connection='usb')
17 gdx.select_sensors()
18 gdx.vp_vernier_canvas()
19 gdx.start()
20
21 ball = sphere(color=color.red)
22
23 while not gdx.vp_close_is_pressed():
24     rate(gdx.vp_rate())
25     if gdx.vp_collect_is_pressed():
26         measurements = gdx.read() # 'measurements' is a list with one data point per sensor
27         if measurements is None: # if there are no measurements available, loop back
28             continue
29         ball.radius = measurements[0] # index out the sensor's data point from the list
```

Starter Web VPython with Go Direct sensor **Note:** This example can be found at <https://vnr.st/glowscript>.

Take a look at the example program, called “ex3-mod-size-with-Go-Direct”, and review each line of code:

Web VPython 3.2

This first line is required for all Web VPython programs.

"""

Control the radius of a VPython sphere with Go Direct sensor data.

For information on the 'gdx' functions, see the Getting Started Guide at:
https://github.com/VernierST/godirect-examples/tree/main/web_vpython

- Connect a Go Direct sensor via USB

- Click 'Share or export this program'
- Click 'Download as HTML'
- Double-click the HTML file to run the program

"""

This text is not run, it is merely comments/instructions for people using this program. In Web VPython, lines can be changed into comments by either three quote marks `"""` or `#`.

```
get_library("https://unpkg.com/@vernier/godirect/dist/webVPython.js")
```

The `get_library` function imports the JavaScript library that allows Web VPython to communicate with our Go Direct devices. **Note:** You must have internet connectivity to use this library.

```
gdx.open(connection='usb')
```

The `gdx.open` function creates an initial connection to a Go Direct device. Change the `'usb'` to `'ble'` to connect wirelessly using Bluetooth.

```
gdx.select_sensors()
```

The `gdx.select_sensors()` function specifies which sensor channels of your Go Direct device to turn on. When the argument is left blank, the default channel for the sensor is used. **Note:** You can also select the channel from the Web VPython canvas while the program is running.

```
gdx.vp_vernier_canvas()
```

The `gdx.vp_vernier_canvas()` function adds a number of useful data collection objects to the VPython canvas. For instance, the default objects (when the argument is left blank) include: Collect/Stop and Close buttons, a data collection rate slider, a live meter readout, and a channel setup box.

```
gdx.start()
```

The `gdx.start()` function sets the data collection period (in milliseconds) and starts the Go Direct device collecting data. If the argument is left blank it will set the data collection period to 100 ms. You can set the period with an argument, e.g. `gdx.start(period=500)`. **Note:** When the program is running, you can modify the sample period by using the data collection rate slider on the VPython canvas.

```
ball = sphere(color=color.red)
```

The `sphere()` object is a standard VPython function that creates a 3D sphere. By changing the values in the argument, it is easy to change the ball's position, size, color, or other properties.

```
while not gdx.vp_close_is_pressed():  
    rate(gdx.vp_rate())  
    if gdx.vp_collect_is_pressed():
```

This code shows the data-collection loop. The `while` loop monitors the Close button on the canvas. If the Close button is pressed, the loop is terminated and the program ends. As long as the Close button has *not* been pressed, the program monitors the Collect/Stop button. When the Collect button is pressed, data collection occurs. When the Stop button is pressed, data collection is stopped. You can start and stop data collection as many times as necessary. The `rate(gdx.vp_rate())` statement is a VPython function that is required in any animation loop. It prevents the program from monopolizing the browser when nothing is happening in the loop.

```
measurements = gdx.read()
```

The `gdx.read()` function reads a data point from all of the sensor channels that you have selected. In this example program, that data is stored in a variable called 'measurements'. The function `gdx.read()` returns data as a 1-dimensional list. **Note:** If only 1 channel is active, `gdx.read()` still returns a list, just a list with one value.

```
if measurements is None:  
    continue
```

This code handles the situation when there is no data to read. For instance, if the Go Direct device is turned off or an error in the code prevents it from connecting, there might be no data to be read. In such an event, this code has the program skip to the end of the data collection loop.

```
ball.radius = measurements[0]
```

In this example code, the sensor reading from the Go Direct device is used to change the size of the ball shown on the canvas. When the sensor reading increases, the box grows larger; when it decreases, the ball shrinks. **Note:** the `ball.radius` function sets the size, i.e. radius, of the ball object. Since the `measurements` variable is a list and the

first value in that list has an index value of 0, `measurements[0]` returns the first value in the list - which is then set as the radius of the ball.

More detailed information for all of the functions can be found [below](#).

Running a Web VPython Program with a Go Direct Device

In order to run the `ex3-mod-size-with-Go-Direct`, you will need to create a copy of it in your own Web VPython account.

1. Sign into your account, go to your programs, and click on **Create New Program**.
2. There are a few options on how to create your own copy of the program:
 - Copy the program line-by-line.
 - Copy the text below and paste it in. **Note:** This is just the whole program with the first line left out.

```
get_library('https://unpkg.com/@vernier/godirect/dist/webVPython.js')
```

```
gdx.open(connection='usb')
gdx.select_sensors()
gdx.vp_vernier_canvas()
gdx.start()
```

```
ball = sphere(color=color.red)
```

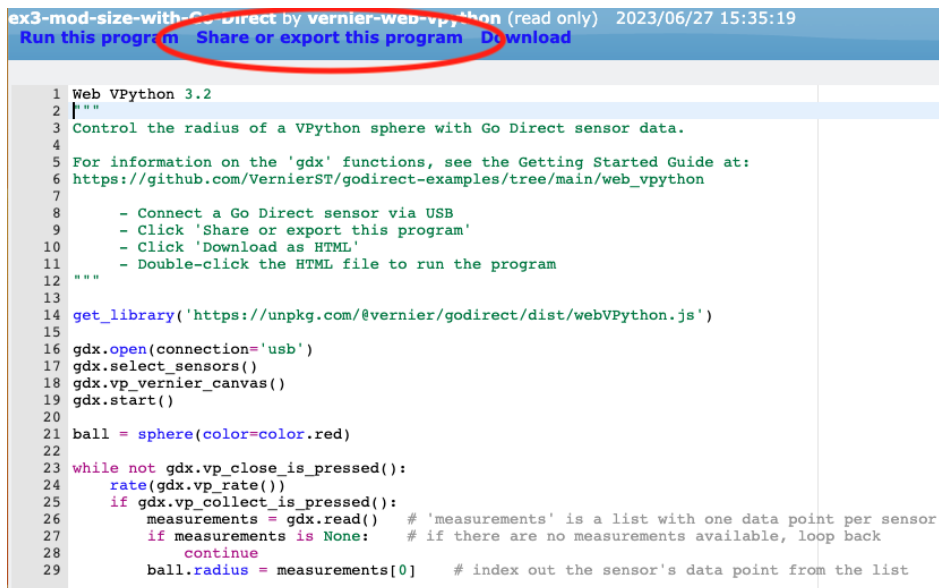
```
while not gdx.vp_close_is_pressed():
    rate(gdx.vp_rate())
    if gdx.vp_collect_is_pressed():
        measurements = gdx.read()
        if measurements is None:
            continue
        ball.radius = measurements[0]
```

- Go to the public Vernier example programs at <https://vnr.st/glowscript> and find the `ex3-mod-size-with-Go-Direct` program. Click on **View** to see the program and copy all the text and paste it into your new program.

With a copy of the program in your account, you will see **Run this program** in the editor. That option will only work for VPython code that does *not* include Go Direct devices. Your browser will not let you directly run a Web VPython program that connects to a Go Direct device for internet security reasons. Instead, you must download your program as an HTML file and then later run it.

To run a Web VPython program that connects to a Go Direct device:

1. Choose **Share or export this program** and a new web page will open. Note the whitespace at the bottom of this page. If this area is blank, the program did not compile correctly and there is a bug in the code. Go back to the editor to fix the code.

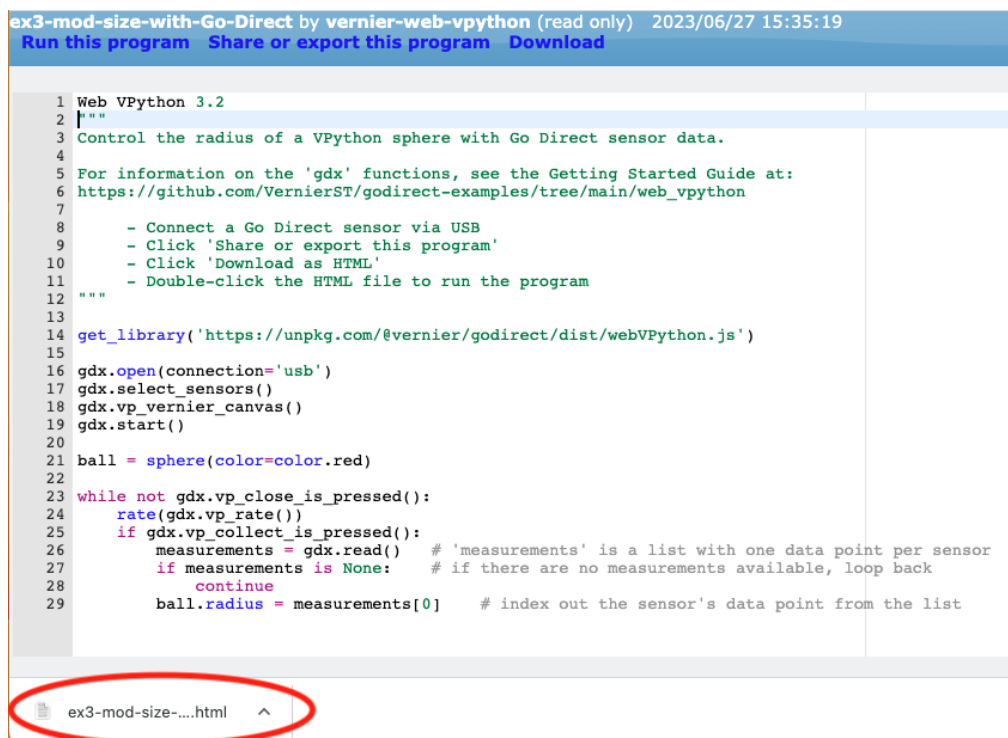


```
ex3-mod-size-with-Go-Direct-by-vernier-web-vpython (read only) 2023/06/27 15:35:19
Run this program Share or export this program Download

1 Web VPython 3.2
2 """
3 Control the radius of a VPython sphere with Go Direct sensor data.
4
5 For information on the 'gdx' functions, see the Getting Started Guide at:
6 https://github.com/VernierST/godirect-examples/tree/main/web_vpython
7
8 - Connect a Go Direct sensor via USB
9 - Click 'Share or export this program'
10 - Click 'Download as HTML'
11 - Double-click the HTML file to run the program
12 """
13
14 get_library('https://unpkg.com/@vernier/godirect/dist/webVPython.js')
15
16 gdx.open(connection='usb')
17 gdx.select_sensors()
18 gdx.vp_vernier_canvas()
19 gdx.start()
20
21 ball = sphere(color=color.red)
22
23 while not gdx.vp_close_is_pressed():
24     rate(gdx.vp_rate())
25     if gdx.vp_collect_is_pressed():
26         measurements = gdx.read() # 'measurements' is a list with one data point per sensor
27         if measurements is None: # if there are no measurements available, loop back
28             continue
29         ball.radius = measurements[0] # index out the sensor's data point from the list
```

*Click **Share or export this program** to compile and download your program*

2. Click **Download as HTML** to download your program as an HTML file that your browser can run offline. **Note:** If you are using a Chrome browser, an icon for the downloaded file will usually appear at the bottom left of the browser and you can just click on that to run it.
3. Once the html file has downloaded to your computer, click on the html file to open and run it.



```
ex3-mod-size-with-Go-Direct by vernier-web-vpython (read only) 2023/06/27 15:35:19
Run this program Share or export this program Download

1 Web VPython 3.2
2 """
3 Control the radius of a VPython sphere with Go Direct sensor data.
4
5 For information on the 'gdx' functions, see the Getting Started Guide at:
6 https://github.com/VernierST/godirect-examples/tree/main/web_vpython
7
8 - Connect a Go Direct sensor via USB
9 - Click 'Share or export this program'
10 - Click 'Download as HTML'
11 - Double-click the HTML file to run the program
12 """
13
14 get_library('https://unpkg.com/@vernier/godirect/dist/webVPython.js')
15
16 gdx.open(connection='usb')
17 gdx.select_sensors()
18 gdx.vp_vernier_canvas()
19 gdx.start()
20
21 ball = sphere(color=color.red)
22
23 while not gdx.vp_close_is_pressed():
24     rate(gdx.vp_rate())
25     if gdx.vp_collect_is_pressed():
26         measurements = gdx.read() # 'measurements' is a list with one data point per sensor
27         if measurements is None: # if there are no measurements available, loop back
28             continue
29         ball.radius = measurements[0] # index out the sensor's data point from the list
```

ex3-mod-size-....html

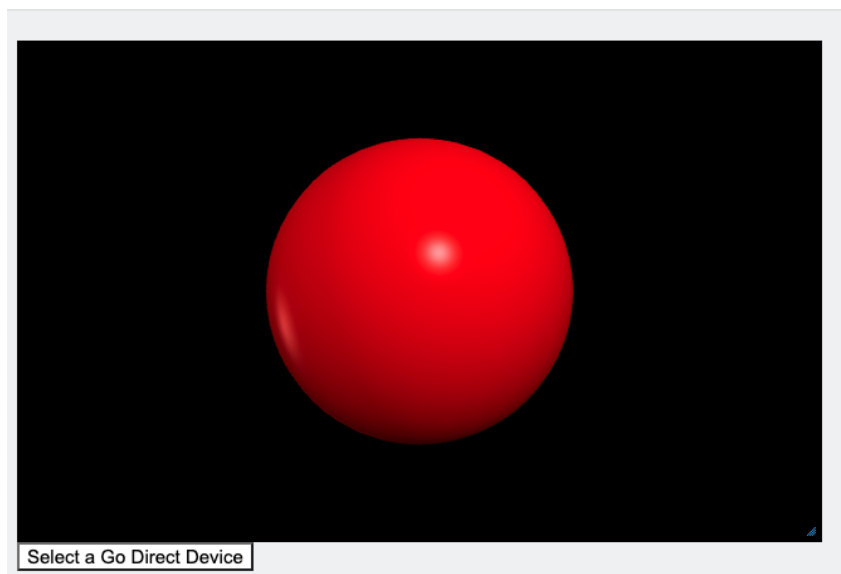
In the Chrome browser, downloads such as the html file you just downloaded will be indicated at the bottom of the window.

Notes about this process:

- Although downloading and running an HTML file may seem a clumsy way to run a program, it does offer one advantage: the created executable HTML file can be used by anyone, even if they have never heard of Python or Web VPython. It can be shared like any other file (email, online folder, etc).
- One downside of the download-as-HTML system: As you create, edit, and tweak programs, you will generate many similarly-named HTML files in your Downloads folder. Periodic clean-up and organization is essential.

Go Direct Device Selection and Connection

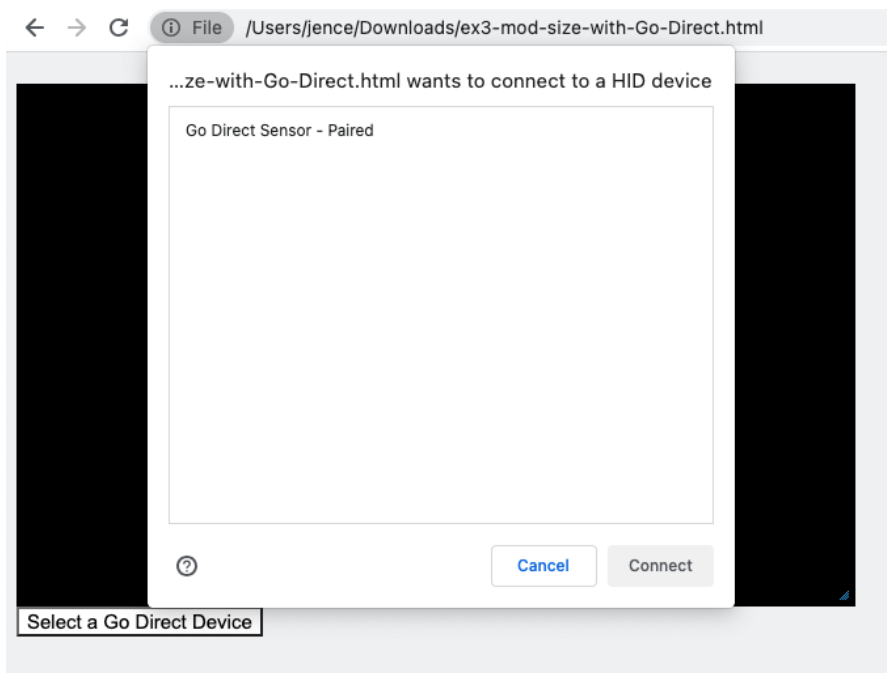
When running the VPythonGettingStartedUSB program, you will see a simple black canvas with a red ball and a **Select a Go Direct Device** button.



ex3-mod-size-with-Go-Direct launch screen

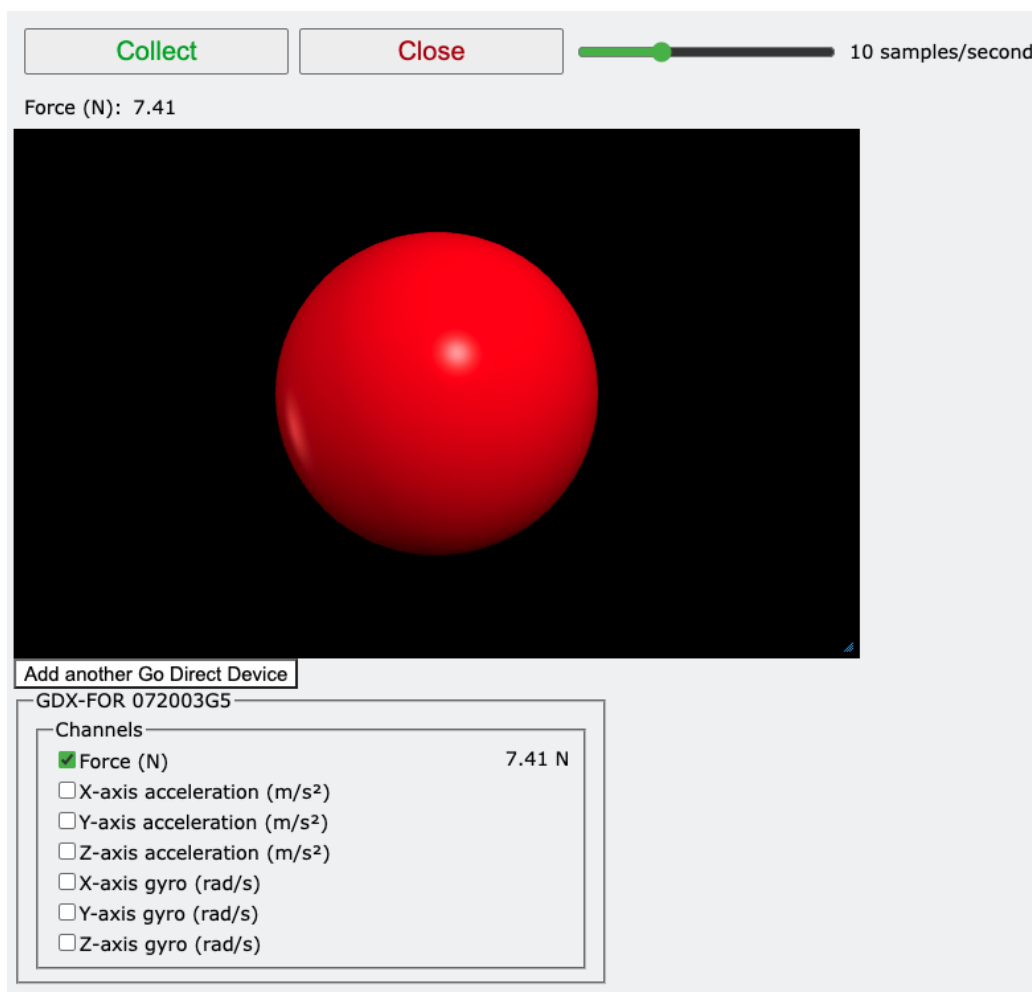
The process for connecting to a Go Direct device while the program is running is the following:

1. Click **Select a Go Direct Device** button. A dialog window will open, displaying all available Go Direct devices that your browser can find. If you are connecting via USB, often only one device will be listed.
2. Click on the name of the device to which you want to connect and click the **Connect** button.



The Chrome browser's device selection dialog window

In the `ex3-mod-size-with-Go-Direct` example program, once a Go Direct device is connected, you will have a canvas like the one below, including: Collect/Stop button, Close button, slider for data collection rate, a live, “meter” display of the sensor reading, and a channel setup box showing which channel of the Go Direct device is being used.



ex3-mod-size-with-Go-Direct example program with a Go Direct Force and Acceleration (GDX-FOR) sensor connected

Try the following:

- Click on the **Collect** button to start data collection. Your sensor reading will control the size of the ball.
- Click on the **Stop** button when you want to stop data collection. You can start and stop data collection as often as you like.
- Click on **Close** to properly shut down the connection between the Go Direct device and the program.

Example Programs

Examples for using Web VPython with Go Direct devices at: <https://vnr.st/glowscript>

Basic Starter Programs

The following programs are similar to the ex3-mod-size-with-Go-Direct example and are helpful in learning to use Web VPython with GDX Sensors.

File Name	Notes
ex1-create-object	Demonstrates the ease of creating a VPython object, such as a sphere, box, and cone. Note: This example does not use Go Direct sensors.
ex2-modify-object	Demonstrates how to modify a VPython object's attributes. Note: This example does not use Go Direct sensors.
ex3-mod-size-with-Go-Direct	Demonstrates how to control the size of a VPython object with Go Direct sensor data, described in the previous section.
ex4-mod-position-with-Go-Direct	Demonstrates how to use a Go Direct sensor to control the position of a sphere within a translucent rod.
ex5-chart-with-Go-Direct	Demonstrates how to plot Go Direct sensor data to a scrolling chart.
ex6-set-collection-duration	Demonstrates how to collect data for a user-specified duration using the <code>gdx.collectFor()</code> function.
ex7-download-to-file	Demonstrates how to capture data collected from your Go Direct sensor and download it as a .csv file.
ex8-upload-from-file	Demonstrates how to upload Go Direct sensor data from a .csv file for visualization or analysis.
ex9-chart-modeling-data	Demonstrates how to plot both Go Direct sensor data and theoretical data together. This scenario is useful in testing a model of a physical phenomena. The model, generated from the equation, can be refined to better reflect the empirical data collected.

Go Direct Sensor Applications

The following programs were written for individual Go Direct sensors and allow some functionality or visualization with those sensors that may be difficult or impossible to replicate in our Graphical Analysis software.

- **GDX-LC-color-match** - This program reads the red, green, and blue sensor channels of the Go Direct Light and Color sensor (GDX-LC) to determine the currently “seen” color and changes the color of a sphere to match.
- **GDX-FOR-ForceVectors** - This program reads the accelerometer channels of a Go Direct Force and Acceleration sensor (GDX-FOR) and changes the orientation of an arrow on the canvas to match the orientation of the force sensor. Additionally, it scales the length of the arrow with the force measured by the force sensor.
- **GDX-ACC-control-tilt** and **GDX-HD-control-tilt** - These two, similar programs use the accelerometer channels in a Go Direct Acceleration and Go Direct Hand Dynamometer, respectively, to control the orientation of an object on the canvas.
- **GDX-FOR-live-freebody-diagram** - The program draws a simple free-body diagram for a ring suspended by two strings, from which a weight is hung. By connecting one of the strings to a Go Direct Force and Acceleration sensor (GDX-FOR) and reading its force and orientation, the program can dynamically update the other two forces (magnitude and direction) acting on the ring.
- **GDX-MD-simple-harmonic-oscillator** - This program uses a Go Direct Motion Detector (GDX-MD) to compare measured position data from a mass on a spring with theoretical model data.
- **MappingElectricPotentialDemonstration** - This program demonstrates plotting 2-D and 3-D field maps in Web VPython. Since no sensor is connected or required, you can just Run this program without having to download it as an HTML file.

Go Direct Library Functions in Web VPython

The Go Direct Library in Web VPython includes functions for connecting to, collecting data from, and closing connections to Go Direct devices. Additionally, the library also creates a canvas that can include elements such as a sensor channel selection box, start/collect and close buttons, data collection rate slide, a visualization canvas, and a graph of sensor data. Below you will find details about each of the Go Direct functions in the library.

`gdx.open()`

This function opens a connection to a Go Direct device. The connection can be made via USB or wirelessly via Bluetooth:

- `gdx.open(connection='usb')`
- `gdx.open(connection='ble')`

Note: Shortened and upper-case versions of connection method work as well, e.g. `gdx.open('usb')` or `gdx.open('BLE')`.

Note: When connecting via Bluetooth, each available sensor will be preceded by a triangle icon that indicates the Bluetooth radio signal strength. The darker it is, the stronger the Bluetooth signal is.



An optional argument parameter is `device_to_open`. If the `device_to_open` parameter is left out, the `gdx.open()` function finds all available Go Direct devices, displays the list, and prompts the user to select the devices to connect.

You can use `device_to_open` with your Go Direct device name(s):

```
gdx.open(connection='ble', device_to_open="GDX-FOR 071000U9, GDX-HD  
151000C1")
```

If the device name or names is used as the argument in the `gdx.open()` function you still need to manually select the device to connect. However, this allows you to know which device is the first device and which is the second (if you are opening two devices, for example).

`gdx.select_sensors()`

The `gdx.select_sensors()` function chooses with sensor channels in the Go Direct device to turn on and collect data from. In most cases, simply using `gdx.select_sensors()` without an

argument will suffice. In this case, your program will select the default channel on the Go Direct device. If you wish to select other channels, you can do so from the Channel Selection box in the canvas while the program is running.

If you want to specify the channel or channels on the sensor to be read, argument format is a list of sensor channels, such as:

```
gdx.select_sensors([5])
```

If you want to use multiple channels, separate the channel numbers with commas:

```
gdx.select_sensors([1,2])
```

If you are connecting to multiple Go Direct devices, use a 2-dimensional list:

```
gdx.select_sensors([[1,2,3],[1]])
```

In this example, the argument `[[1,2,3],[1]]` enables sensor channels 1, 2, and 3 for the first device and sensor channel 1 for the second device.

`gdx.start()`

The `gdx.start()` function starts the Go Direct device collecting data. If this function's argument is left blank, a default sampling period of 100 milliseconds, or 10 readings per second, will be used. To specify a different sampling rate, enter the sampling period (in ms) in the argument of the `gdx.start()` function:

```
gdx.start(50)
```

Some additional details about the `gdx.start()` function:

- The same sampling rate is used for all selected sensor channels.
- The data collection rate slider on the canvas can also be used to adjust the sampling rate while the program is running, up to 30 samples/second.
- Sampling at a period that is less than 10 milliseconds (or greater than 100 samples/second) may be problematic because javascript/python infrastructure used to gather data will unlikely be able to move data that fast.
- Be aware not to use Web VPython's `print()` function in your data collection loop as it will bog down the data collection.

`gdx.read()`

The `gdx.read()` function will take single point readings from the selected sensors at the specified period and return the readings as a one dimensional list. For instance, the first item in

the list is `measurements[0]`, the second is `measurements[1]`, etc. **Note:** index 0 is the first value in the list, index 1 is the second, and so on.

Place this function in the data collection loop and make sure the loop can iterate fast enough to keep up with the sampling period, i.e. do not add functions such as Web VPython's `print()` function that might slow the loop's execution.

`gdx.vp_close_is_pressed()`

The function `gdx.vp_close_is_pressed()` monitors the state of the VPython canvas Close button and returns either `True` or `False`. When true, this function will call `gdx.stop()` and `gdx.close()` to stop data collection and disconnect the device.

`gdx.vp_collect_is_pressed()`

The function `gdx.vp_collect_is_pressed()` monitors the state of the VPython canvas Collect/Stop button and returns either `True` or `False`. When Collect is clicked, the function will call `gdx.start()`. When Stop is clicked, a `gdx.stop()` is called.

`gdx.vp_get_slider_period()`

The `gdx.vp_get_slider_period()` function returns the period of data collection in milliseconds. It is useful when parts of your program, such as a custom graph, need to know the data collection rate or period.

`gdx.collectFor()`

For some programs, you may want to collect data for a set duration of time, rather than manually start and stop data collection. The `gdx.collectFor()` function allows you to do so. The argument of the function sets the data collection duration in seconds. For example, `gdx.collectFor(5)` means that the program would collect data for 5 seconds once the Collect button is pressed.

[Example program ex6-set-collection-duration](#) shows how to use the `gdx.collectFor()` function.

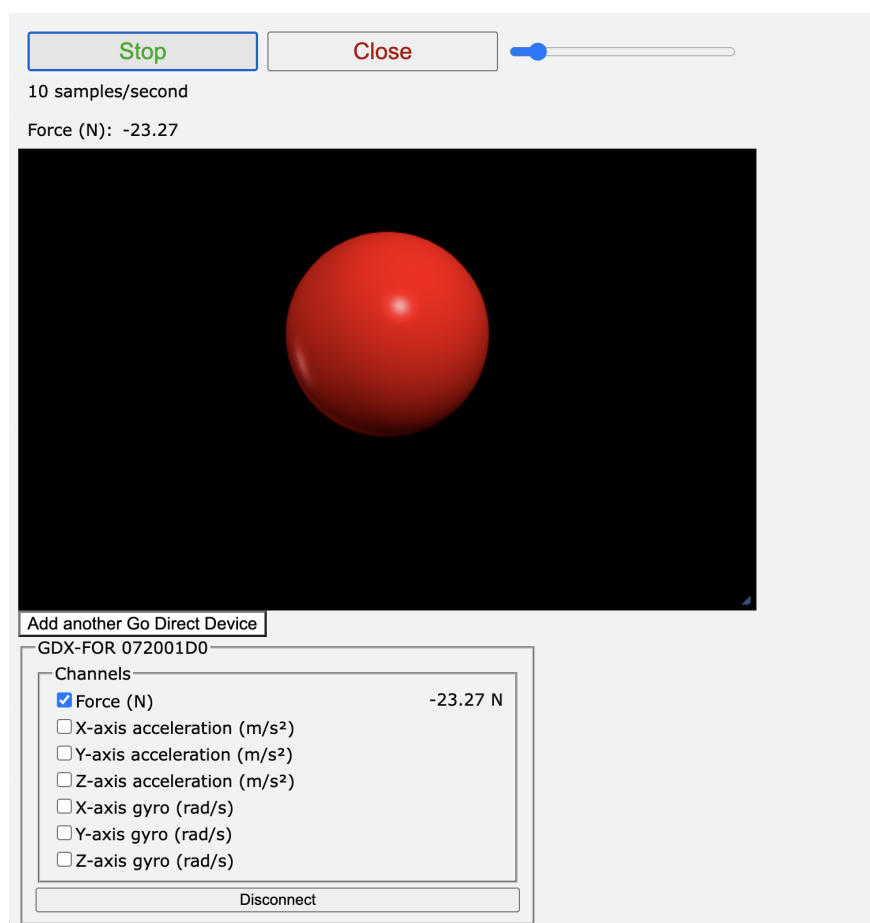
`gdx.vp_rate()`

All Web VPython programs include a `rate()` statement. [rate\(\)](#) is a Web VPython function that slows the program loop so that any animations or graphs run at a reasonable speed for the user. The rate at which the program loop runs is especially important if you are including data

collection from a Go Direct device. Once you use `gdx.start()`, the Go Direct device begins collecting data and sends it to your Web VPython program. If your program loops too quickly or too slowly, the program may lose data or may not receive data from the Go Direct device. The `gdx.vp_rate()` function chooses an appropriate value for the `rate()` function so that your program loops at a rate compatible with the sampling rate you have set either through the argument in `gdx.start()` or through the data collection rate slider on the canvas.

`gdx.vp_vernier_canvas()`

The `gdx.vp_vernier_canvas()` function contains a number of important options and features for programs involving data collection. With no arguments, the function will set up the canvas in Web VPython like this:



Default configuration output for `gdx.vp_vernier_canvas()`

The `gdx.vp_vernier_canvas()` function makes it simple to create a data collection program with a Go Direct device. The function creates a Collect button to start data collection; the Collect

button changes to Stop when pressed and is then used to stop data collection. You can start and stop data collection repeatedly while the program is running. Additionally, the function creates a Close button that will nicely shut down the Go Direct device's connection with the program.

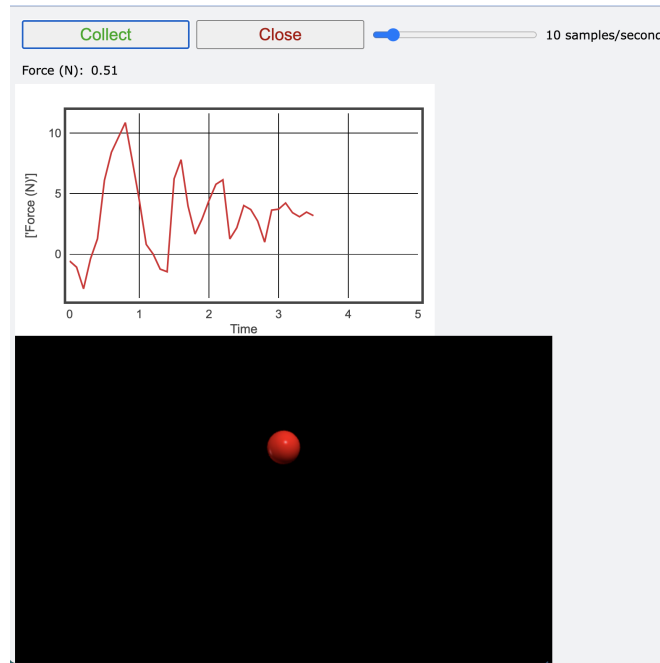
The `gdx.vp_vernier_canvas()` function creates live meters for the selected sensor channels in the space below the Collect/Stop button.

The `gdx.vp_vernier_canvas()` function also creates a data collection rate slider to the right of the buttons. The default data collection rate on the slider is 10 samples per second. It can be moved as small as 1 sample/sec and as large as 30 samples/sec. Faster data collection rates are possible but must be hard-coded into your program using an argument in `gdx.start()`.

Once the program is connected to a sensor, `gdx.vp_vernier_canvas()` creates a Channel Setup box below the visualization canvas. This box allows you to select sensor channels in the connected Go Direct device. If you do not want the Channel Setup box, you can remove it by adding an argument to `gdx.vp_vernier_canvas()`:

```
gdx.vp_vernier_canvas(channel_setup=False)
```

An optional element of `gdx.vp_vernier_canvas()` is a scrolling graph. Add this graph by adding `chart=True` to the `gdx.vp_vernier_canvas()` argument. For example, using the function `gdx.vp_vernier_canvas(channel_setup=False, chart=True)` would remove the Channel Setup box but add the scrolling graph, as shown below.



Edit the `gdx.vp_vernier_canvas()` Arguments to add or remove elements from the canvas

The default configuration for `gdx.vp_vernier_canvas()` function is:

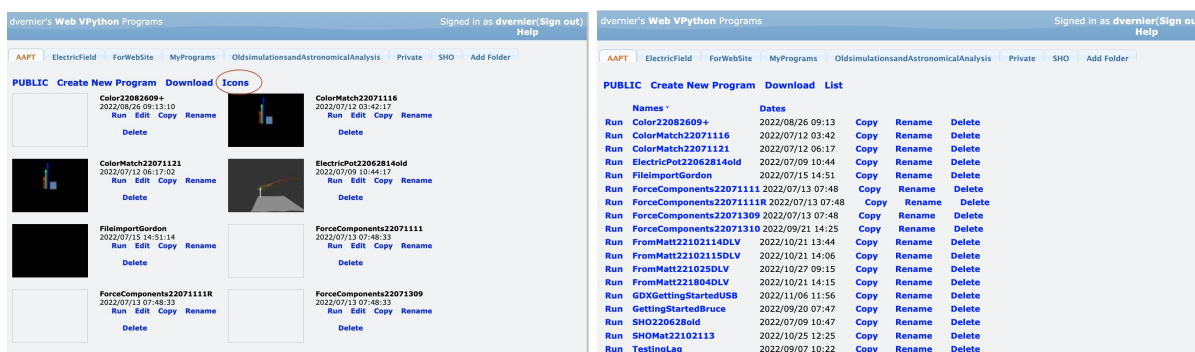
```
gdx.vp_vernier_canvas(buttons=True, meters=True, slider=True,  
channel_setup=True, chart=False)
```

Tips for using Web VPython

As you work with the Web VPython, you will find that there are a few practices you can adopt to make your work easier. The following tips should help.

Displaying Programs as a List in Your Account

By default Web VPython displays programs as icons. If you click on the word **Icons**, shown below, the icons change to a list of programs.



Programs in your account are displayed as icons by default but can be shown in a list

The primary advantage of the List view is that you can sort the programs by name or date, by clicking on the column heading.

Note: You can also create folders and rename/delete programs to organize the programs stored in your account.

Moving Programs between Web VPython and Installed VPython

Go Direct devices can also be used with installed Python and the VPython library on your computer; for details, see [Getting Started with Vernier Go Direct Sensors and VPython](#).

Occasionally you may want to move code from Web VPython and run it on an installed version of Python for development; you may find that the richer, more fully featured debugging tools in installed Python are helpful. If you want to move a Web VPython program into a format that can be run in installed Python, do the following:

1. Comment out the top two lines of your Web VPython code, by doing adding a # to the start of the first two lines.

```
#Web VPython 3.2  
#get_library("https://unpkg.com/@vernier/godirect/dist/webVPython.js")
```
2. Add these three lines:


```
from vpython import *
from.gdx import.gdx
gdx = gdx.gdx()
```

Likewise, moving a program from an installed Python into a format that can be run in Web VPython requires similar steps.

1. Add these two lines to the top of your code:

```
Web VPython 3.2
get_library("https://unpkg.com/@vernier/godirect/dist/webVPython.js")
```

2. Comment out these three lines:

```
# from vpython import *
# from.gdx import.gdx
# gdx = gdx.gdx()
```

Between installed Python and Web VPython, there are a few minor differences in how the VPython library is used.

- The `channel_setup` parameter is not available in installed Python. Instead you choose sensor channels from the console when the program is run. This means that the parameter options in the `gdx.vp_vernier_canvas()` function are a little different.
 - In Web VPython the parameter are: `gdx.vp_vernier_canvas(buttons=True, meters=True, slider=True, channel_setup=True)`
 - In installed Python, the parameters are:
`gdx.vp_vernier_canvas(buttons=True, meters=True, slider=True, chart=True, cvs=True)`
- Make sure to include the `rate()` statement in the Web VPython data collection loop. The `rate()` statement does not appear to be as important when working within installed VPython.

Channels Available on Go Direct Devices

If you use the `gdx.select_sensors()` function without any arguments, you get the default channel. For most Go Direct devices, this default sensor channel is channel 1. There are a few exceptions. Additionally, sometimes it is helpful to know which sensor is assigned to each channel. You can find a list of each Go Direct device and its sensor channels on the Vernier website: <https://www.vernier.com/til/16315>

There are a few Go Direct devices that have some unusual sensor channel numbers or unique data collection modes listed below.

- Go Direct Motion Detector (GDX-MD) - The Motion Detector has three channels, of which only one can be active at a time:
 - 5 Motion - This is the default sensor channel.

- 6 Motion (cart) - This sensor channel is a bit more sensitive than the default channel and was designed for tracking the motion of a dynamics cart.
 - 7 Motion with TC - This sensor channel adjusts the position measurement by compensating for the ambient temperature. This is the channel to use if you are collecting data in a location that is more than a few degrees warmer or colder than 20° C.
- Go Direct Rotary Motion Sensor (GDX-RMS) - The Rotary Motion Sensor has two sensor channels, of which only one can be active at a time:
 - 5 Angle - This is the default sensor channel.
 - 6 Angle (high resolution) - This sensor channel reports angle at a higher resolution than the default channel but maxes out at 7.5 rev/sec (versus 30 rev/sec with the default channel).
- Go Direct Sound (GDX-SND) - The Sound sensor default channel is channel 1 sound pressure. But it is **not** compatible with Web VPython because it requires a higher sampling rate than Web VPython can accommodate. The other channels are compatible with Web VPython.
 - 1 Sound Pressure - not compatible with Web VPython
 - 2 Sound Level (A weighted)
 - 3 Sound Level (C weighted)
 - 4 Wave Amplitude

Troubleshooting and Support

If your Web VPython program is not working properly, try the following troubleshooting steps. If you are still having problems, please contact us using the support options listed.

Troubleshooting

- If you are having trouble, you may first check out our [FAQ for Python Troubleshooting](#) article.
- To run a program you must click **Share or export this program** and then **Download as HTML**. Before clicking on **Download as HTML**, look at the bottom of the page to make sure there is code in the window. If not, there is an error. To locate the error, go back to the editor and click **Run this program**. You may receive a helpful error message. Often, commenting out portions of your code to simplify your program until it works and then step-by-step uncommenting them will help you identify the troublesome code.
- If you suspect the problems arise from the code connecting to/collecting data from a Go Direct sensor, try rewriting your program to remove the Go Direct library and replace it with fake data.
- If you get a message like No Web HID Support when running a program, you may be using the wrong browser. Make sure to use Chrome.
- If you click on Run this program and get the following error:

```
TypeError: Cannot read properties of undefined (read'__argnames__')
```

It is likely you clicked **Run this program**. Instead, click **Share or export this program** and the **Download as HTML** procedure to use the program.

- In June 2023, the Go Direct Web VPython library was updated to improve performance. This may cause previously working programs to crash. Please refer to [TIL 17633](#) to find instructions to modify your program to work with the updated library.

Support

Additional help can be found by reaching out:

- Email: support@vernier.com
- [Vernier Science Education](#) website chat window
- Call: 503-277-2299 or 1-888-VERNIER
- Post a question about the Go Direct library at:
<https://github.com/VernierST/godirect-examples/issues>
- Post a question about Web VPython on the Web VPython forum at:
<https://groups.google.com/g/glowscript-users>

Web VPython Help

In the top right corner of the Web VPython window, there is a very helpful Help link. You will find information on any of the objects you can add to your program, like boxes, spheres, arrows, helices, etc, as well as the details about canvases.

The Web VPython examples at <https://www.glowscript.org/#/user/GlowScriptDemos/folder/Examples/> are amazing. You can run them, you can view and copy the code. Studying them is a great way to see what is possible and to learn programming tricks.

Web Resources for Python and VPython

If you are totally new to Python, here are some generally helpful links for getting started with Python.

- [Python for Beginners](#)
- [Official Python FAQs](#)

Here are some websites with great information about Web VPython:

- [Matter and Interactions](#), a physics textbook written by Bruce Sherwood and Ruth Chabay, creators of Web VPython
- [Rhett Allain](#), college physics professor and blogger, who often includes computational solutions and investigations in his work
- [VPython Tour - YouTube](#)